Article

The International Journal of Robotics Research
1–15
© The Author(s) 2018
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364918783061
journals.sagepub.com/home/ijr
SAGE

Learning to reconstruct 3D structures for occupancy mapping from depth and color information

Vitor Guizilini and Fabio Ramos

Abstract

Real-world scenarios contain many structural patterns that, if appropriately extracted and modeled, can be used to reduce problems associated with sensor failure and occlusions while improving planning methods in such tasks as navigation and grasping. This paper devises a novel unsupervised procedure that models 3D structures from unorganized pointclouds as occupancy maps. Our methodology enables the learning of unique and arbitrarily complex features using a variational Bayesian convolutional auto-encoder, which compresses local information into a latent low-dimensional representation and then decodes it back in order to reconstruct the original scene, including color information when available. This reconstructive model is trained on features obtained automatically from a wide variety of scenarios, in order to improve its generalization and interpolative powers. We show that the proposed framework is able to recover partially missing structures and reason over occlusions with high accuracy while maintaining a detailed reconstruction of observed areas. To combine localized feature estimates seamlessly into a single global structure, we employ the Hilbert maps framework, recently proposed as a robust and efficient occupancy mapping technique, and introduce a new kernel for reproducing kernel Hilbert space projection that uses estimates from the reconstructive model. Experimental tests are conducted with large-scale 2D and 3D datasets, using both laser and monocular data, and a study of the impact of various accuracy–speed trade-offs is provided to assess the limits of the proposed methodology.

Keywords

Occupancy models, Hilbert maps, scene reconstruction, variational inference, convolutional auto-encoders

1. Introduction

In this day and age, the task of collecting information from the environment is no longer an issue, as standard sensors are able to output millions of points in a fraction of a second. The challenge now is to store and interpret all the data in a way that can be exploited by both human beings and machines. One crucial task is the generation of environment models (i.e. maps) that are able to distinguish between occupied and unoccupied areas in 3D space. The knowledge of which areas can be safely traversed and which would result in a collision is of key importance for applications ranging from grasping and object manipulation to obstacle avoidance and autonomous navigation.

Initial models would simply discretize the space (Elfes, 1989; Nüchter et al., 2007), maintaining an equally sized grid to store independent information about each particular area. This approach, however, is very memory-intensive and does not take into account spatial relationships between cells. OctoMaps (Hornung et al., 2013) use a tree-like structure to recursively divide the space as necessary; this results in a massive decrease in memory requirement and processing time for grid-like occupancy mapping. Gaussian process occupancy maps (O'Callaghan et al., 2009) and Gaussian process implicit surfaces (Dragiev et al., 2011) both address spatial dependency by producing a continuous probabilistic function that maps location to occupancy values. However, they both scale cubically in relation to the number of training points, which limits their applicability to larger datasets, especially when 3D information is involved. Sparse approximations are commonly used to increase efficiency (Kim and Kim, 2015; Jadidi et al., 2017); however, these come with significantly more complex models, and at the cost of accuracy in the final reconstruction results.

A more recent approach, Hilbert maps (Ramos and Ott, 2015), projects data points into a higher-dimensional reproducing kernel Hilbert space, based on a series of simple kernel functions, and then uses linear classifiers to produce

School of Information Technologies, The University of Sydney, Australia

Corresponding author:
Vitor Campanholo Guizilini, School of Information Technologies, The University of Sydney, Australia.
Email: vitor.guizilini@sydney.edu.au

a continuous probabilistic occupancy function. The resulting framework maintains the spatial relationship between inputs (capable of better data interpolation) and also does not require space discretization (can be queried at arbitrary resolutions). Over the last couple of years, significant efforts have been made in extending this framework to address larger datasets (Guizilini and Ramos, 2016), the learning of more complex features (Guizilini and Ramos, 2017b), and dynamic environment modeling (Senanayake et al., 2016). However, these works are restricted in the type of features that can be learned from data. Guizilini and Ramos (2016) use a squared-exponential kernel, which restricts the resulting features to ellipsoids, while Guizilini and Ramos (2017b) propose the addition of a planar surface kernel and devise a methodology to determine which feature is better suited to each portion of the environment.

The main contribution of this paper is the development of a novel methodology that allows the learning of unique and arbitrarily complex 3D features to represent different structures in the environment, based on a convolutional variational auto-encoder (CVAE) trained on a series of features extracted from previously observed maps, including color information when available. The convolutional neural network architecture has already been successfully applied to several areas of computer vision, i.e. image classification (He et al., 2016), object detection (Ren et al., 2015), semantic segmentation (Ghiasi and Fowlkes, 2016), and inpainting (Pathak et al., 2016), and is now transitioning into 3D information, as more volumetric capturing techniques (Newcombe et al., 2011) and large-scale repositories (Wu et al., 2015) become publicly available. The number of feature patterns available in the 3D space is exponentially larger, making it much more difficult to design representative models of these features manually. Thus, the ability to learn relevant patterns automatically, based on historical data, and use the resulting representative models to reconstruct newly observed areas of the input space, becomes even more attractive.

In particular, we will be focusing on convolutional auto-encoders (CAEs) (Masci et al., 2011; Pu et al., 2016), owing to their ability to reason over spatial information to produce latent low-dimensional representations; and their variational counterpart (Doersch, 2016), which infuses Bayesian probabilistic inference into a deep learning framework to produce a generative model for new structures. To the best of our knowledge, this is the first time that deep variational auto-encoders are used to learn complex environment features for 3D occupancy mapping, thus significantly reducing gaps in the environment model due to occlusions, sparse data, and partially observed objects. Within the Hilbert maps framework, the benefits of such an approach are:

- *More detailed reconstructions*. The learned features are not restricted to any single predetermined shape, and so can better adapt to more complex structures.
- *Sparser representation*. Each feature is able to cover a larger portion of the environment, meaning that fewer clusters are necessary for an accurate reconstruction.
- *Better reasoning over data gaps*. The model uses its learned filters to reconstruct unobserved parts of the environment, and thus is better able to deal with partial occlusions and sensor failure.

The remainder of this paper is structured as follows. We start by describing the theoretical background necessary to implement the proposed technique, including an overview of the Hilbert maps framework, the concept of auto-encoders, and their extension to a convolutional variational scenario. We then proceed to introduce the proposed methodology, detailing its various steps, such as automatic feature extraction, the reconstructive model used to recover partially observed structures, and the final occupancy mapping framework. Experimental results are then presented and discussed, alongside comparisons with other occupancy mapping techniques, both qualitatively and quantitatively. Finally, we conclude the paper with a summary of its contributions and discuss potential directions for future work within this newly proposed framework.

This paper improves on the original work of Guizilini and Ramos (2017a) by introducing the use of data collected from visual sensors, particularly monocular cameras, obtained using structure-from-motion techniques (Waechter et al., 2014) based solely on image frames, without the need of intrinsic calibration or initial pose estimates. Furthermore, we show how the proposed framework can be used to learn both color and depth information simultaneously from extracted features, and then include these estimates during the reconstruction process. Lastly, we provide cross-comparisons between models trained on data from different sensor types (i.e. reconstructing monocular data using a model trained on laser data, and vice versa), showing that the proposed framework is able to abstract over sensor differences and still produce accurate reconstructions of partially observed structures that outperform current state-of-the-art occupancy mapping techniques.

2. Theoretical background

This section provides a brief overview of the two main techniques used to create the proposed 3D scene reconstruction algorithm. The Hilbert maps framework organizes and indexes available data, extracting clusters that contain potential feature information. The convolutional and variational auto-encoders are responsible for processing the said information, training a generative model that encodes input data into a low-dimensional latent feature vector and then decodes it back to reconstruct different portions of the input space.

2.1. Hilbert maps

Ramos and Ott (2015) proposed a novel framework for scene reconstruction, in which real-world complexity is represented in a linear fashion by projecting spatial coordinates into a high-dimensional feature vector. This high-dimensional representation is known as a *Hilbert space* (Sansone, 2012) and, furthermore, if point evaluation in this space is a continuous linear functional (i.e. if $||f - g||$ is small for functions f and g, then $|f(x) - g(x)|$ is also small for all x), then it becomes a *reproducing kernel Hilbert space* (Schölkopf et al., 2015). The dot product of these feature vectors can be used to approximate popular kernels found in the literature (Rasmussen and Williams, 2005); by operating only in terms of kernel evaluations, we never have to perform calculations explicitly in this high-dimensional (and potentially infinite) feature space.

We assume a training dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathcal{R}^D$ is a point in the D-dimensional space and $y_i = \{-1, +1\}$ is its corresponding occupancy state. For example, in a laser scanner, the return distances are treated as occupied points, while the space traversed by each beam is randomly sampled (e.g. once every meter, at arbitrary locations) and treated as unoccupied points. Note that uncertainty can be naturally encoded into the framework by using labels with values between $[-1, 1]$, with 0 indicating no occupancy information; however, here only binary labels are considered, without loss of generality. These input points are projected into the reproducing kernel Hilbert space using a feature vector function $\Phi(\mathbf{x})$, and a classifier is trained in this high-dimensional space to produce a discriminative model $p(y|\mathbf{x}, \mathbf{w})$, where $\mathbf{w}$ are the parameters of this classifier. It can be shown (Komarek, 2004) that linear separators are almost always adequate to separate classes in high-dimensional spaces, making simple classifiers sufficiently accurate for this task. The result is a compact and efficient model, whose parameters can be trained in an online fashion and then queried at constant time to produce probabilistic occupancy value estimates at arbitrary resolutions.

2.2. Convolutional auto-encoders

The main purpose of unsupervised learning methods is to extract useful features in unlabeled data, removing input redundancies while preserving the essential aspects, which are then used to produce robust and discriminative representations. Within this context, the *encoder–decoder* paradigm is arguably the most commonly used (Hinton and Salakhutdinov, 2006); in this paradigm, the input is first projected into a lower-dimensional space (*encoded*) and then expanded to reproduce the initial data (*decoded*). Techniques using this paradigm include: low-complexity coding and decoding machines (Hochreiter and Schmidhuber, 1999), auto-encoders (Ranzato et al., 2007), energy-based models (LeCun et al., 2006), and restricted Boltzmann machines (Hinton, 2002).

Recently, deep architectures have taken over most learning tasks (Deng and Yu, 2014), providing an unprecedented level of pattern recognition and data abstraction that is inspired by biological systems. In particular, convolutional neural networks excel with visual information (Ciresan et al., 2011), because they preserve the input's spatial locality and neighborhood information in latent higher-level feature representations. In contrast to fully connected deep architectures, which do not scale well to high-dimensional inputs in terms of computational complexity, the number of free parameters in a convolutional neural network does not depend on input dimensionality, since they are locally shared in each layer.

The concept of CAEs originated in the work of Masci et al. (2011), as a hierarchical unsupervised feature extractor that uses stochastic gradient descent to learn good convolutional neural network initializations, thus avoiding distinct local minima in highly non-convex objective functions. It takes an input $\mathbf{x} \in \mathcal{R}^D$ and first maps it to a latent representation $\mathbf{h} \in \mathcal{R}^{D'}$, using a deterministic function

$$\mathbf{h}^k = f(\mathbf{x}, \theta^k) = \sigma\left(\mathbf{x} * W^k + b^k\right) \tag{1}$$

with parameters $\theta = \{W, b\}$ and activation function σ (the symbol $*$ denotes convolution). The same process is repeated for each of the k channels, producing a latent multi-channel representation H. The resulting "encoded" vector is then used to reconstruct the input via a reverse mapping

$$\mathbf{r} = f'(H, \theta'^k) = \sigma\left(\sum_{k \in H} \mathbf{h}^k * \tilde{W}^k + c^k\right) \tag{2}$$

with the parameters $\theta' = \{\tilde{W}, c\}$ usually constrained such that $\tilde{W} = W^{\mathrm{T}}$, i.e. the same weights are used for encoding the input and decoding the latent representation. These parameters are optimized by minimizing an appropriate cost function over the training set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, usually the mean squared error between input and reconstructed states

$$E(\theta) = \frac{1}{2N} \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{r}_i||^2 \tag{3}$$

Similarly to standard neural networks, the back-propagation algorithm is applied to compute the gradient of the error function with respect to the parameters. For equation (3), this can be easily obtained by convolution operations using

$$\frac{\partial E(\theta)}{\partial W^k} = \mathbf{x} * \delta\mathbf{h}^k + \mathbf{h}^k * \delta\mathbf{r} \tag{4}$$

where $\delta\mathbf{h}$ and $\delta\mathbf{r}$ are gradients of the hidden and reconstructed states, respectively. The weights can then be updated using standard stochastic gradient descent, in mini-batches composed of training data. Furthermore, several layers can be stacked to form a deeper hierarchy, with each layer receiving as input the latent representation from the previous layer (Figure 1(a)). A max-pooling layer (Scherer et al., 2010) is often introduced between convolutional layers to generate translation invariant results, in which the latent representation is down-sampled by a constant factor by taking the maximum value over non-overlapping sub-regions. This extra step both reduces computational complexity and improves filter selectivity, since now the activation of each neuron is determined by a region of interest, rather than a single value. During decoding, the max-pooling operation is reversed by resizing the latent vector, using such techniques as bilinear interpolation or nearest neighbors.

2.3. Variational auto-encoders

Even though it is commonly classified as of the same type, the variational auto-encoder actually has little to do with classical auto-encoders (Doersch, 2016). In a nutshell, a variational auto-encoder aims to optimize the parameters θ of a function $f(z, \theta)$ so that, when a set of latent variables z is sampled from a probability distribution $P(z)$, there is a high probability that $f(z, \theta)$ will resemble the input points $X = \{\mathbf{x}_i\}_{i=1}^{N}$ from the dataset. In other words, it aims to optimize the probability of X according to

$$P(X) = \int f(z, \theta) P(z)\, \mathrm{d}z = \int P(X|z, \theta) P(z)\, \mathrm{d}z \tag{5}$$

In this scenario, z are the latent variables that encode input data into a low-dimensional manifold and $f(z, \theta)$ is substituted by a neural network, as seen in Figure 1(b). Furthermore, we also assume that $P(z) \sim \mathcal{N}(\mathbf{0}, I)$, following the intuition that any D-dimensional distribution can be generated by taking a set of D variables that are normally distributed and mapping them through a sufficiently complicated function, such as $f(z, \theta)$.

This posterior probability, however, is intractable, so a variational approach is taken to provide an analytical approximation, in the form of a lower bound that can be used for efficient calculations. We start by defining a new function $Q(z|X)$, which takes a value of X and returns a distribution over z values that are likely to produce X back. With some rearranging (Doersch, 2016), here omitted for brevity, the lower bound to be optimized becomes

$$\begin{aligned}\mathcal{L} &= \log P(X) - KL[Q(z|X)\,||P(z|X)] \\ &= E[\log P(X|z)] - KL[Q(z|X)\,||P(z)]\end{aligned} \tag{6}$$

In this equation, in the first line, we are attempting to maximize $P(X)$ while simultaneously minimizing the Kullback–Leibler (KL) divergence $KL[Q(z|X)\,||P(z|X)]$ (i.e. trying to approximate the two functions). In the second line, it is possible to see some similarities to standard auto-encoders, since Q essentially “encodes" X into z, while P is “decoding" z in order to reconstruct X. When applying stochastic gradient descent to the right hand side of equation (6), it is usually assumed that $Q(z|X) = \mathcal{N}(z|\mu(X, \gamma), \Sigma(X, \gamma))$, where μ and Σ are arbitrary functions with parameters γ that can be learned directly from data (Σ is also constrained to be a diagonal matrix). The KL-divergence to be optimized can then be simplified to

$$\begin{aligned}&KL[\mathcal{N}(\mu(X), \Sigma(X))\,||\mathcal{N}(0, I)] \\ &= \frac{1}{2}\left(\mathrm{tr}(\Sigma(X)) + \mu(X)^{\mathrm{T}} \mu(X) - k - \log||\Sigma(X)||\right)\end{aligned} \tag{7}$$

where k is the dimensionality of the distribution. Since getting a good estimate of $E[\log P(X|z)]$ in equation (6) would involve passing many samples of z through f, it is common practice to take one sample of z and treat $P(X|z)$ for that z as a good approximation of $E[\log P(X|z)]$. The resulting equation can then be reparameterized (Doersch, 2016) to allow closed-form gradient calculation, if we assume that both distributions $P(z)$ and $Q(z|X)$ are continuous. The resulting lower bound can then be maximized to produce the optimal parameters θ that represent our neural network weights, using standard back-propagation techniques. Note that, since $P(z)$ follows a unit Gaussian distribution, it is possible to generate artificial outputs by sampling from this distribution and then decoding the resulting latent variables.

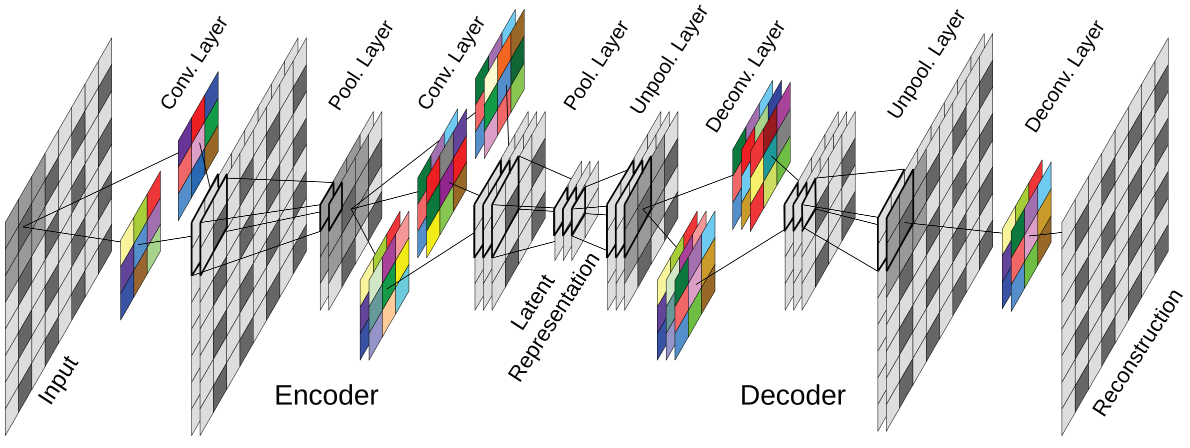
3. Methodology

This section describes how the techniques previously mentioned can be combined to produce a robust 3D scene reconstruction framework, capable of learning complex and unique features for different portions of the environment while probabilistically reasoning over missing information and occlusions. The proposed framework consists of three steps:

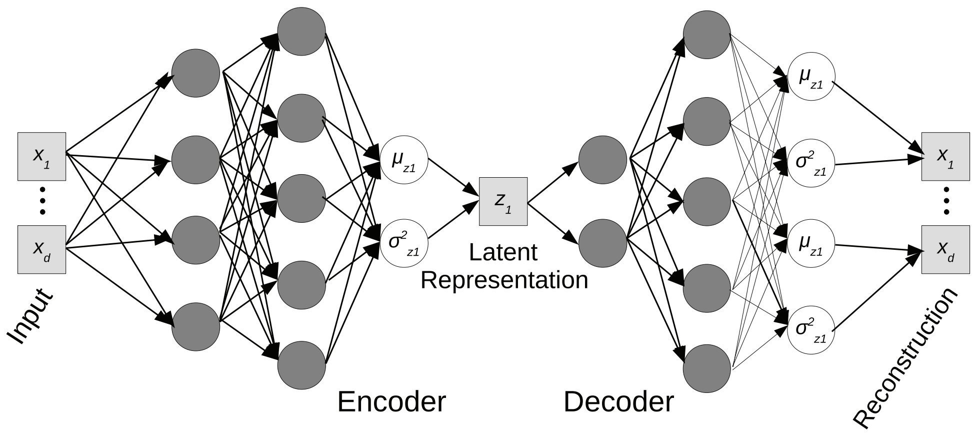
- *Automatic feature extraction*:
  - *Clustering.* The input data are clustered to produce roughly uniformly spaced points.
  - *Feature extraction.* The clusters are used to generate feature vectors for different portions of the input space.
- *Reconstructive model*:
  - *Training.* Observed feature vectors are used to train a CVAE model, minimizing the reconstructed error.
  - *Inference.* The trained CVAE model is used to reconstruct new feature vectors.
- *Occupancy mapping*:
  - *Training.* Observed reconstructed feature vectors are used to update the weights of a Hilbert map.
  - *Inference.* The trained Hilbert map is used to infer the occupancy state of new reconstructed feature vectors.

3.1. Automatic feature extraction

We start with an unorganized pointcloud $\mathcal{D} = \{\mathcal{X}, \mathbf{y}\} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ containing the spatial coordinate of each point



(a) Convolutional Auto-Encoder.



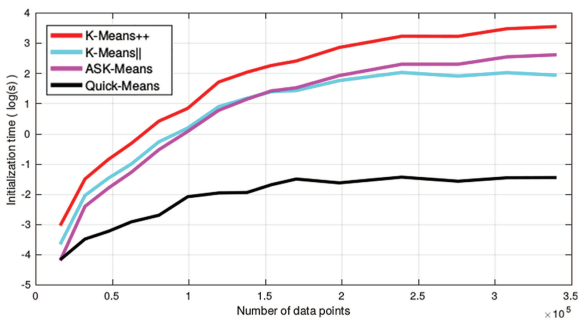
(b) Variational Auto-Encoder.

Fig. 1. Visual representations of the deep learning architectures used in this paper.

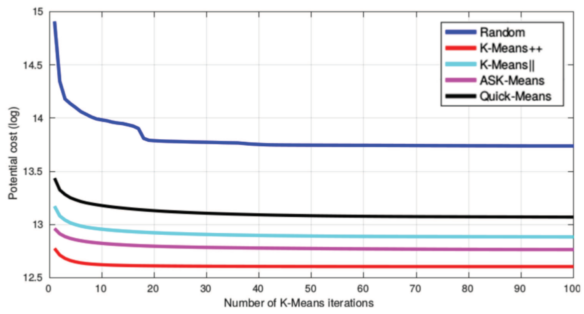
and its corresponding occupancy value, as described in Section 2.1. This unorganized pointcloud is clustered, to produce different local structures that will serve as features and together describe the environment as a whole. Guizilini and Ramos (2017b) proposed an alternative to the standard k-means++ initialization algorithm (Arthur and Vassilvitskii, 2007), which selects the starting cluster seeds for further optimization, i.e. using the standard k-means algorithm (Kanungo et al., 2002). This technique was shown to outperform the k-means++ algorithm in terms of speed, while allowing automatic determination of the number of clusters necessary to describe the environment properly, given a distance threshold.

Here, we build on the ASK-Means algorithm described in Guizilini and Ramos (2017b) and propose Quick-Means, an extension that further improves its computational speed without significantly compromising accuracy, as can be seen in Figure 2. The key insight is that, for the particular application at hand, we are not interested in precise cluster locations, but rather in regularly placed clusters, which are within a given inner radius threshold r_i given a distance metric $d(.,.)$, here chosen to be the Euclidean distance. Because of this, there is no need to calculate sampling probabilities for each point, only relative distances; this significantly improves computational speed. Furthermore, it accommodates cluster overlapping by using an outer radius threshold $r_\text{o} > r_\text{i}$, as a way to increase feature complexity and model structures from different perspectives. Pseudocode for the Quick-Means initialization algorithm can be found in Algorithm 1.

Once the M clusters are obtained, the next step is to encode the information contained in each of them, so it can be used as input by the reconstructive model. Here this is done by generating a grid $G_m = \{\mathbf{x}_i, y_i\}_{i=1}^d$ around the cluster (Figure 3(b)), at a certain resolution r_G and with dimensionality $d = \lceil r_G^{-1} \rceil^D$ (unless noted otherwise, a value of $r_G = 0.1$ m was used throughout this paper). Each coordinate is populated with the most common occupancy value for data points that fall within that cell (-1 for unoccupied,



(a) Initialization time (random values are negligible).



(b) Potential cost $\left(\underset{S}{\text{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} ||\mathbf{x} - \boldsymbol{\mu}_i||^2\right)$ during k-means.

Fig. 2. Comparison of different clustering initialization techniques: Random, k-means++ (Arthur and Vassilvitskii, 2007), k-means|| (Bahmani et al., 2012), ASK-Means (Guizilini and Ramos, 2017b), and the proposed Quick-Means algorithm (average of 50 runs with different random seeds).

Algorithm 1 Quick-Means initialization algorithm.

```
Require: pointcloud $\mathcal{X}$ with $N$ points
         radial inner $r_i$ and outer $r_o$ thresholds
         minimum number of points per cluster $k$
         distance function $d(\cdot,\cdot)$
Ensure: clusters $\mathcal{C}$
 1: $t \leftarrow N$ % Number of available points
 2: $v_i \leftarrow \{0,1,2,\dots,N\}$ % Aux. index vector
 3: $v_j \leftarrow \{0,1,2,\dots,N\}$ % Aux. index vector
 4: $v_b \leftarrow \{0,0,0,\dots,0\}_{i=1}^{N}$ % Aux. Boolean vector
 5: $\mathcal{C} \leftarrow \{\}$ % Empty cluster set
 6: while $t > 0$ do
 7:     $\mathbf{x}_* \leftarrow \mathcal{X}[v_i[\text{rand}(0,t)]]$
 8:     $\mathcal{M} \leftarrow \{\mathbf{x} \mid d(\mathbf{x}_*,\mathbf{x}) < r_o\}, \forall \mathbf{x} \in \mathcal{X}[v_i[0,1,\dots,t]]$
 9:     if $|\mathcal{M}| > k$ then
10:         $\mathcal{C} \leftarrow \mathcal{M}$
11:     end if
12:     for $\mathbf{m} \in \mathcal{M}$ do
13:         $i \leftarrow$ index of $\mathbf{m}$ in $\mathcal{X}$
14:         if $v_b[i] == 0$ and $d(\mathbf{m},\mathbf{x}) < r_i$ then
15:             $v_b[i] = 1, j \leftarrow v_j[i]$
16:             $v_i[j] = v_i[--t]$
17:             $v_j[t] = v_j[v_i[j]] = j$
18:         end if
19:     end for
20: end while
```

0 for unknown, and +1 for occupied)[1]. These grid representations act as the kernels k produced by each extracted cluster, correlating the occupancy values of unobserved points $\mathbf{x}_*$ to observed data as defined by

$$k(\mathbf{x}_*, G) = \begin{cases} 0 & \text{if } \mathbf{x}_* \text{ is outside } G \\ y_j^G \text{ for } \mathbf{x}_j^G \text{ closest to } \mathbf{x}_* & \text{otherwise} \end{cases} \tag{8}$$

The feature vector Φ that projects an input point $\mathbf{x}$ into the reproducing kernel Hilbert space (see equation (15)) is given by a vector containing the kernel values produced by all extracted clusters $\mathcal{G}$, as shown in equation (9). For computational reasons, we enforce sparsity by calculating only the kernels related to a subset of the closest clusters and ignoring the influence of all the others

$$\Phi(\mathbf{x}, \mathcal{G}) = \begin{bmatrix} k(\mathbf{x}, G_1) \\ k(\mathbf{x}, G_2) \\ \vdots \\ k(\mathbf{x}, G_M) \end{bmatrix} \tag{9}$$

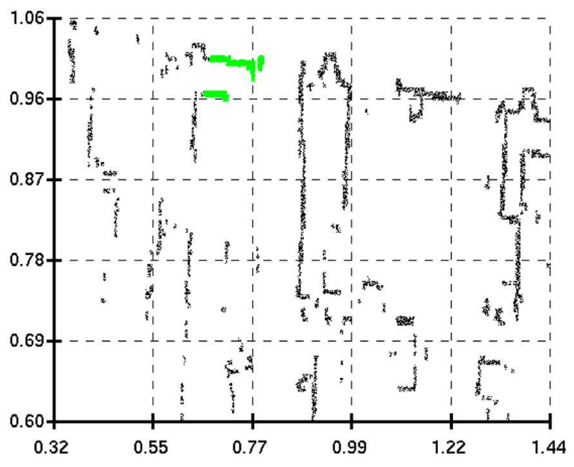
Furthermore, to increase symmetry during the reconstruction process, the data points in each grid G_m are transformed to an aligned grid representation H_m (see equation (10)) by: translation of $\bar{G}_m$ to a zero-median position; rotation so their sorted eigenvectors V_m are, from the largest to the smallest eigenvalue, aligned with the coordinate system; and scaling by s so their largest dimension becomes unitary (Figure 3(c)). These transformations are stored to be used later, in order to recover the original grid representation from an aligned reconstructed grid

$$H_m = \frac{1}{s}\left((G_m - \bar{G}_m)V_m\right) \tag{10}$$

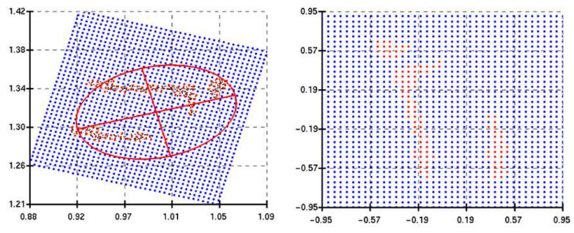
3.2. *Reconstructive model*

Strictly speaking, the feature vector Φ previously described could be inserted directly into the Hilbert maps framework, combining individual cluster information to produce occupancy values for any input point. However, this naive approach does not address such issues as interpolation, extrapolation, data gaps, partial occlusion, and so forth. Because of this, we propose the use of a reconstructive model that, as the name implies, takes these feature vectors and reconstructs the observed structure based on how it should look, according to prior information collected from other datasets containing similar objects (see Figure 3).

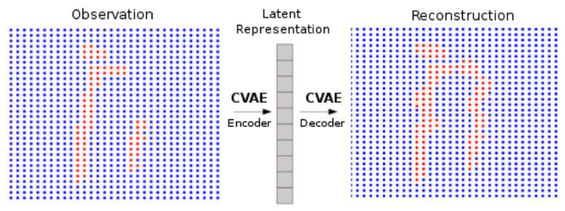
Here, this reconstructive model is a CVAE, which combines the concepts found in Sections 2.2 and 2.3 in a single framework. The use of convolutional layers preserves spatial relationships and greatly decreases the number of training parameters, while variational approximations are able to encode information in a significantly smaller latent representation, as will be demonstrated experimentally. Given an aligned reconstructed grid H as input, the resulting encoded



(a) Unorganized point cloud (green dots indicate data from the selected cluster).



(b) Extracted cluster information. **(c)** Aligned grid representation.



(d) A partial observation of the original structure is encoded into a latent representation and then decoded back.

Fig. 3. The 2D feature extraction and reconstruction process.

vectors (i.e. the latent variables z, as introduced in equation (5)) are generated from the probabilistic distribution

$$Q(\mathbf{z}|H) = \mathcal{N}\left(\mathbf{z}|\boldsymbol{\mu}(H), \boldsymbol{\sigma}^2(H)\right) \tag{11}$$

3.2.1. Learning occupancy values. As a simple 2D numerical example, we start with a $32 \times 32 \times 1$ aligned grid representation, as shown in Figure 3(c) (for now only occupancy values are considered; color reconstruction will be addressed on the next subsection). During the encoding process, this grid goes through a series of convolutional layers, with different kernel filter sizes, a stride of 1, a max-pooling of 2, and a rectified linear unit activation function (Glorot et al., 2011). The use of max-pooling is important because it both decreases computational cost and allows the next layers to extract patterns on higher scales (see Figure 1(a)). The number of channels, conversely, increases as more kernels filters are used to process the same input simultaneously, emulating a larger number of possible useful patterns that can be learned during the training process. For the example at hand, the first layer produces 32 channels and each one afterwards doubles this value, up to a maximum of 256 channels.

Once this process is complete, the output $\mathbf{y}$ of the last convolutional layer is used to produce the mean and variance values (see equation (11)) that compose the latent representation $\mathbf{z}$, as depicted in Figure 1(b). Here, the generation of these mean and variance values is defined as

$$\boldsymbol{\mu}(H) = \text{Fully}(\mathbf{y}) \tag{12}$$

$$\boldsymbol{\sigma}(H) = \text{Softplus}(\text{Fully}(\mathbf{y})) \tag{13}$$

where Fully(.) is a single-layer fully connected neural network in which all inputs contribute to the calculation of each output, and Softplus(.) is an activation function that applies the nonlinearity $\log(1 + \exp(\,.\,))$ to each input, thus ensuring positive variances. Note that, while $\mathbf{y}$ is shared in the calculation of both mean and variance values, each one has its own fully connected neural network Fully(.), which is not shared between variables, even though they use the same input information.

During the decoding process, the latent representation goes through deconvolutional layers, with similar properties to their convolutional counter-parts, and is expanded using an unpooling operation (i.e. the input grid is resized to its required output dimensions) (Zeiler and Fergus, 2014). The resulting reconstructed grid H'_m has the same size as the original representation, and contains occupancy estimates according to the CVAE model, normalized between values of $[0, 1]$ using a Sigmoid(.) activation function. An example of this process can be seen in Figure 3(d), in which a partial observation is encoded into its latent representation and then decoded back to produce a reconstruction of the observed structure, including its missing parts. This reconstructed aligned grid can be transformed back to its original shape G'_m by reversing the stored transformations for that cluster (equation (14)) and is then used to produce the reconstructed feature vector $\Phi(\mathbf{x}, \mathcal{G}')$, as shown in equation (9)

$$G'_m = sV_m^{\mathrm{T}} H'_m + \bar{G}_m \tag{14}$$

The CVAE model is trained using mini-batches of reconstructed aligned feature vectors obtained from clusters extracted from the training datasets (Figure 3(a)), which are assumed to contain similar structures to the ones found in the evaluation dataset. As explained in Section 2.3, the reconstruction error (i.e. similarity between input and output) is minimized alongside the KL-divergence (i.e. the variational approximation is as close as possible to the true distribution). To avoid over-fitting, a dropout (Srivastava et al., 2014) value of 0.8 was introduced in the last layer. Dropout is a form of regularizer in which a percentage of neurons are randomly “switched off", or set to zero.

3.2.2. Learning color information. When color information is available (i.e. if visual data were used to generate the input pointcloud), it is possible to incorporate these values into the reconstructive model itself, which will then be able to estimate color and occupancy states simultaneously during the reconstruction process. Following the simple 2D example from before, the aligned input grid representation now has shape $32 \times 32 \times 4$, with each channel respectively storing occupancy values and RGB color information, all normalized to have a range between [0, 1]. Since only occupied points have a corresponding color value, from triangulated matched features, during grid generation, unoccupied and unknown cells receive $(0, 0, 0)$ (i.e. black). This information serves only as a placeholder, so that the full grid can be used as an input for the neural network, and has no purpose during the reconstruction process.

The CVAE reconstructive model remains the same, albeit now receiving as input a four-channel aligned grid, containing occupancy and RGB values, rather than a one-channel aligned grid containing only occupancy values. This extra information is convolved during the encoding process and projected into a low-dimensional latent feature vector representation, defined by equations (12) and (13). This latent representation is then decoded to produce a reconstructed output grid with the same dimensionality as the input, containing both occupancy and RGB estimates, also normalized using the Sigmoid(.) activation function to have range between [0, 1]. Note that, while occupancy values represent a state distribution indicating the probability that each cell is occupied, color values indicate continuous functions that quantify each of the primary colors for each cell.

3.3. *Occupancy mapping*

The reconstructed feature vectors $\Phi(\mathbf{x}, \mathcal{G}')$ obtained previously are used to classify the input space, according to the Hilbert maps methodology described in Section 2.1. Here we employ a logistic regression classifier, in which the probability of occupancy for a query point $\mathbf{x}_*$ is given by

$$p\big(y_* = 1 | \Phi'(\mathbf{x}_*), \mathbf{w}\big) = \frac{1}{1 + \exp\big(\mathbf{w}^{\mathrm{T}} \Phi'(\mathbf{x}_*)\big)} \tag{15}$$

where the dependencies on $\mathcal{G}'$ were removed for notation clarity. To optimize the weight parameters $\mathbf{w}$ based on information contained in $\mathcal{D}$, we minimize the *regularized negative log-likelihood* function

$$\mathrm{RNLL}(\mathbf{w}) = \sum_{i=1}^{N} \big(1 + \exp\big(-y_i \mathbf{w}^{\mathrm{T}} \Phi(\mathbf{x}_i)\big)\big) + R(\mathbf{w}) \tag{16}$$

in which $R(\mathbf{w})$ is a regularization term, such as the elastic net ($R(\mathbf{w}) = \lambda_1 ||\mathbf{w}||^2_{L2} + \lambda_2 ||\mathbf{w}||^1_{L1}$, where λ_1 and λ_2 are, respectively, the shrinkage and sparseness parameters). A useful property of equation (16) is its suitability for *stochastic gradient descent* optimization (Bottou, 2010), in which information contained in each point, or batch of points, provides one small step toward a local minimum, calculated as

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \frac{\delta}{\delta \mathbf{w}} \mathrm{RNLL}(\mathbf{w}) \tag{17}$$

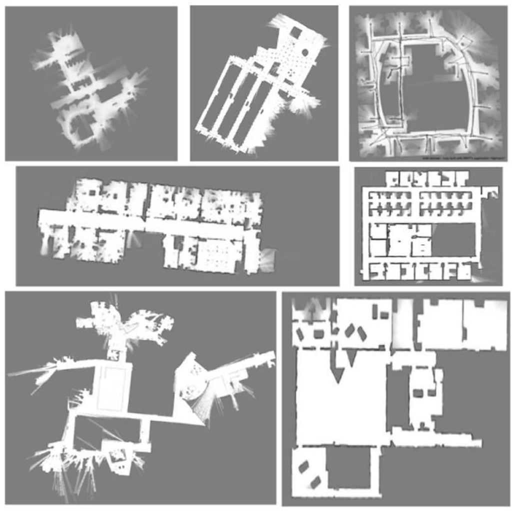
where $\eta > 0$ is the learning rate, usually kept constant or asymptotically decaying with the number of iterations. The main benefit of this training methodology is that the entire dataset never has to be touched at the same time, which might be infeasible, owing to sheer size and memory requirements. Also, note that this technique lends itself naturally to online learning, since new information can be added to the current model by incrementally performing the stochastic update step given by equation (17).

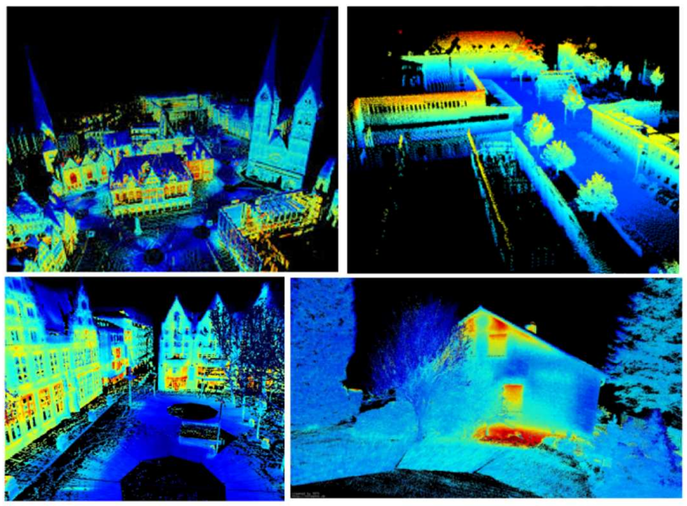
If color information is available, it can be incorporated in the occupancy model to produce more detailed representations of the reconstructed environment. Here we employ a simple averaging system between all color estimates that fall closest to the same grid cell (see equation (8)), although other techniques might produce better results through weighted interpolation (Shepard, 1968). During isosurface calculation, only areas deemed occupied (i.e. having an occupancy probability above a certain threshold) receive this color information, thus producing a textured reconstruction of the environment, including occluded and partially observed structures.

4. Experimental results

The proposed framework was tested using 2D and 3D datasets collected from various publicly available repositories, as depicted in Figure 4. The 2D datasets were obtained using single slice laser range sensors, while the 3D datasets were obtained using both rotating laser range sensors and monocular rolling shutter cameras. Laser data from various scans, collected at different locations during navigation, were combined using SLAM6D (Nüchter et al., 2007) to produce a single global pointcloud. Monocular data from sequences of frames were combined using Structure-from-Motion with Sparse Bundle Adjustment (Waechter et al., 2014), to optimize camera poses and 3D feature locations by minimizing reprojection errors. No color information was used in laser data, while monocular data were assigned an RGB color for each occupied point (unoccupied points were always assigned a placeholder color of $(0, 0, 0)$, i.e. black).

After one dataset is selected for evaluation, the other ones are clustered 20 times according to Section 3.1 with different random seeds, to produce grid representations of observed structures. These grid representations are then used to train the reconstructive model described in Section 3.2. (Different models were trained for each configuration, one for 2D laser data, another for 3D laser data, and a third one for monocular data) Afterwards, the reconstructed feature vectors are used to train a Hilbert map that produces occupancy values for any point in the input space, as depicted in Section 3.3. Finally, the evaluation dataset is clustered, its grid representations are reconstructed, and the resulting feature vectors are used to generate the occupancy values that describe the newly observed environment. During evaluation, different random seeds for clustering were


(a) 2D indoor datasets (laser). Available at `https://www.mrpt.org/robotics_datasets`.


(b) 3D outdoor datasets (laser). Available at `http://kos.informatik.uni-osnabrueck.de/3Dscans/`.

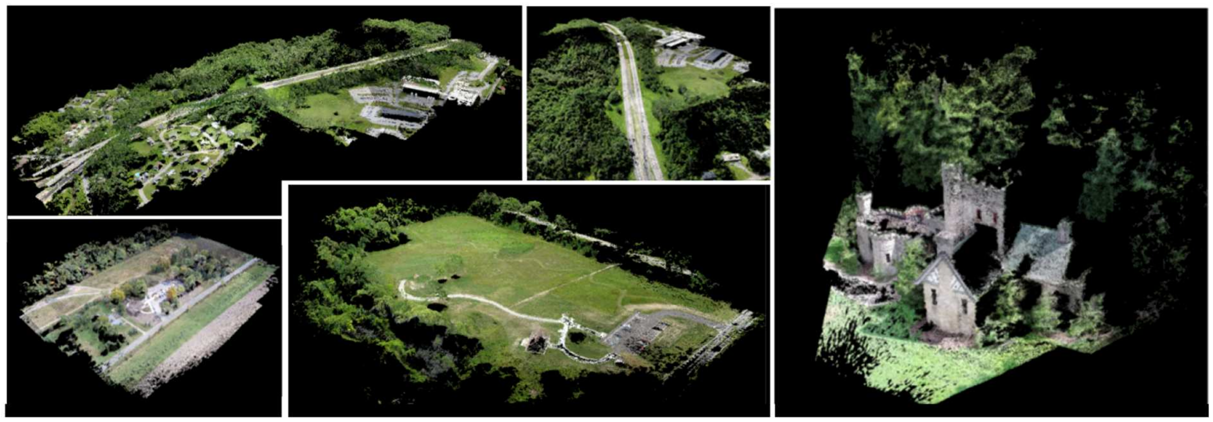

(c) 3D outdoor datasets (monocular). Available at `https://github.com/OpenDroneMap`.

Fig. 4. Examples of datasets used during training and evaluation in this paper, collected from various sources.

considered, and there were no significant changes in overall results.

To test the reconstructive powers of the proposed framework, random portions of the evaluation dataset were removed, as a way to simulate data gaps and partial occlusions. The objective is to maintain a detailed representation of observed structures (both occupied and unoccupied), while also using this available information to reconstruct any gaps. To this end, the following aspects were evaluated:

- *Latent dimension.* The effects of changing the number of latent dimensions in the encoded vector;
- *Cluster size.* The effects of increasing the average cluster size, which produces each extracted feature;
- *Gap ratio.* The effects of changing the relative size of data gaps, in relation to average cluster size.

As a baseline, we selected a grid size of 64, a latent dimension of 25 or 100 for 2D or 3D input data (multiplied by two if color information was also being encoded), a network depth of four convolutional layers (with kernel filter sizes of 9, 7, 5, and 5, max-pooling of 2, and dropout of 0.8 on the last layer) and a gap ratio of 50% in relation to an average cluster size of 5 m. The classification results, for 2D and 3D datasets, respectively, can be found in Tables 1–3. For comparison purposes, we provide results obtained using both a convolutional variational auto-encoder (CVAE-HM) as the reconstructive model and a standard convolutional auto-encoder (CAE-HM). Furthermore, we provide results obtained using the LARD-HM framework described by Guizilini and Ramos (2016) for both 2D and 3D scenarios, Gaussian process occupancy maps (O'Callaghan et al., 2009) for 2D datasets, and OctoMaps (Hornung et al., 2013) for 3D datasets. The Gaussian process occupancy map framework is known for its ability to reason over data gaps, while OctoMaps is considered the state-of-the-art in 3D occupancy mapping. In all cases, an area is considered occupied if its occupancy probability is greater than 60% and unoccupied if this probability is smaller than 40% (values in between are considered unknown).

Table 1. Classification results for 2D laser datasets.

Method	Data observed Precision	Recall	Data gaps Precision	Recall
GPOM	90.60%	80.83%	34.61%	29.94%
LARD-HM	80.44%	99.70%	53.58%	51.71%
CAE-HM	77.82%	74.89%	45.22%	42.45%
CVAE-HM	96.64%	94.75%	90.86%	86.29%

Table 2. Classification results for 3D laser datasets.

Method	Data observed Precision	Recall	Data gaps Precision	Recall
OctoMap	95.59%	94.99%	16.85%	21.94%
LARD-HM	86.08%	96.58%	40.20%	38.62%
CAE-HM(MO)	54.74%	56.08%	32.01%	31.33%
CAE-HM(L)	61.92%	64.53%	36.20%	34.26%
CVAE-HM(MO)	84.77%	83.91%	76.48%	74.69%
CVAE-HM(L)	92.91%	89.68%	86.02%	81.27%

L: laser; M: monocular; O: occupancy.

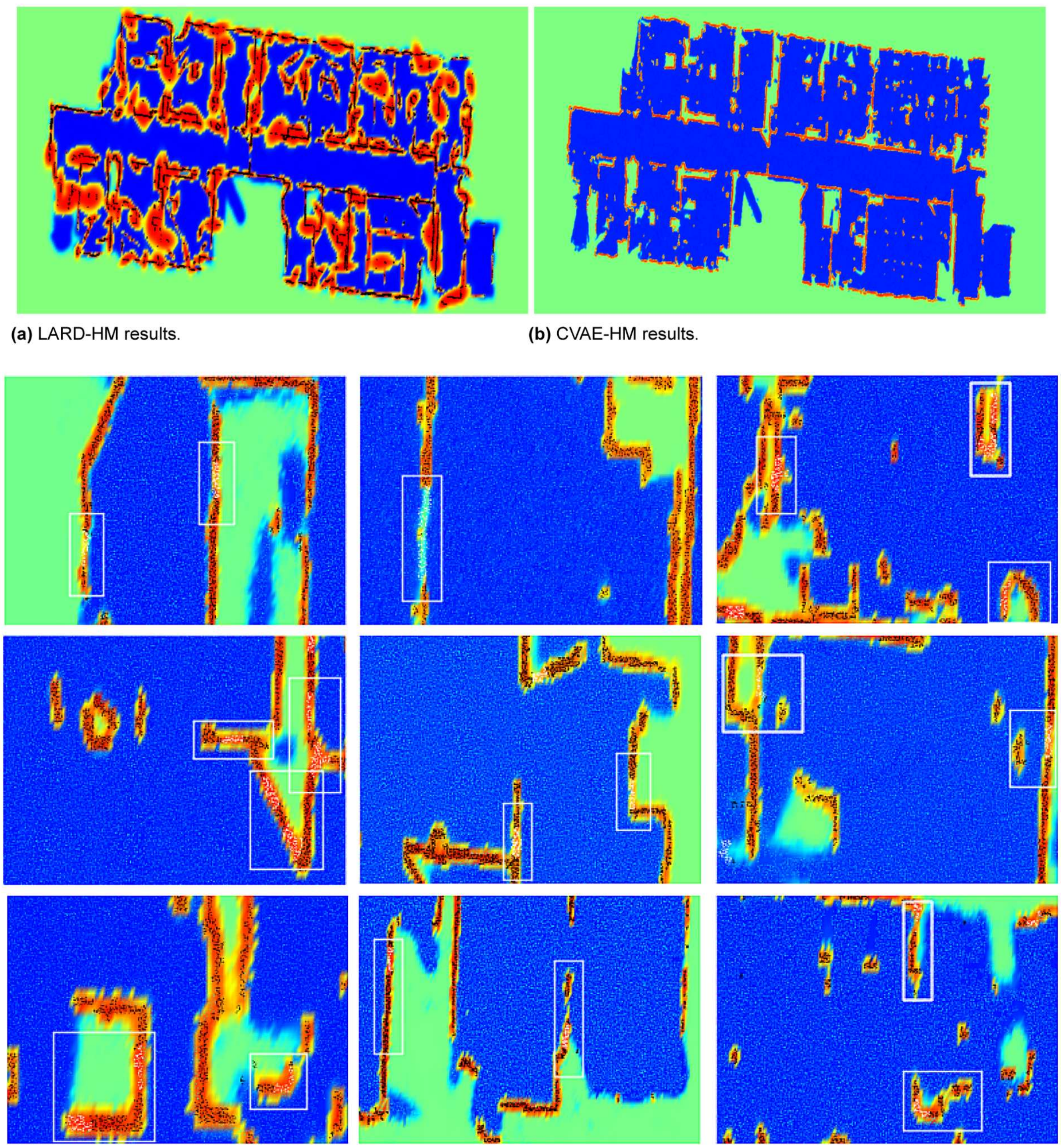
Table 3. Classification results for 3D monocular datasets.

Method	Data observed Precision	Recall	Data gaps Precision	Recall
OctoMap	93.87%	93.21%	18.33%	20.32%
LARD-HM	87.41%	96.02%	38.74%	39.44%
CAE-HM(L)	50.98%	53.72%	31.18%	29.42%
CAE-HM(MO)	58.91%	60.09%	31.66%	28.79%
CAE-HM(MC)	59.78%	59.11%	32.13%	30.94%
CVAE-HM(L)	78.64%	75.90%	69.58%	73.39%
CVAE-HM(MO)	88.92%	84.11%	84.36%	76.88%
CVAE-HM(MC)	90.07%	85.56%	84.59%	77.21%

C: color; L: laser; M: monocular; O: occupancy.

The precision-recall metric (Powers, 2011) was selected for evaluation, because it depicts different aspects of results (i.e. the ratios between true or false positives and negatives), thus providing a better understanding of how occupied and unoccupied areas are reconstructed. As shown in Tables 1–3, the proposed method is able to achieve a considerable higher reconstruction rate of data gaps in relation to other techniques, while still providing competitive rates when dealing with observed information (see Figures 5–7). Interestingly, using the model trained on 3D laser data (L) to reconstruct 3D monocular data (M), and vice versa, did not significantly affect the results. This indicates that the proposed framework is able to abstract over sensor differences, and even changes in environment structures, and produce features that can be applied equally to a wide variety of scenarios without further training. No color information (C) was used during these cross-tests, only occupancy values (O); however, for 3D monocular data, both models (with and without color information) were generated, and the introduction of color actually produced slightly more accurate overall results. We attribute this behavior to a tendency in similar structures to also share similarities in color; this facilitates the generation of similar shapes during the reconstruction process, since these two properties are estimated simultaneously.

Additionally, these results show that the introduction of a variational component into the CAE framework is crucial in achieving satisfactory reconstruction results, especially with smaller latent dimensions. A comparison of the effects of changing this dimensionality can be found in Figure 8, where we see that CAE-HM requires more dimensions to converge and still achieves worse results, in terms of both observed information and data gaps. Interestingly, the reconstruction rate of data gaps starts to decrease after the latent dimensionality reaches a certain value, a phenomenon we attribute to over-fitting, since a higher-dimensional vector is better able to fit training data and thus learns to model gaps as part of the reconstructed



(a) LARD-HM results.

(b) CVAE-HM results.

(c) Enlarged areas of CVAE-HM results. Black dots indicate occupied areas, light blue dots indicate unoccupied areas, and white dots indicate structure gaps. Blue, red, and green shades, respectively, indicate unoccupied, occupied, and unknown areas.

Fig. 5. The 2D scene reconstruction results from Table 1. (a) The LARD-HM framework is used; it is clear that it is unable to reconstruct such large-scale features in detail. (b) The same dataset is reconstructed using the proposed CVAE-HM framework. (c) Enlarged areas are shown, depicting regions in which there were data gaps (white dots) to be reconstructed.

features. Fine-tuning the network topology, as well as the introduction of recently proposed normalization techniques (Klambauer et al., 2017), would probably address this shortcoming and produce better results; however, this was not explored here and is left for future work.

Another comparison was made in relation to average cluster size, which dictates the size of features extracted from the environment and, by extension, their complexity. Figure 5(a) shows that the standard LARD-HM framework is already unable to reconstruct environments with an average cluster size of 5 m, while CVAE-HM is able to achieve a much more detailed representation. Figure 9 shows results for CVAE-HM under different average cluster sizes, for both observed information and data gaps. As

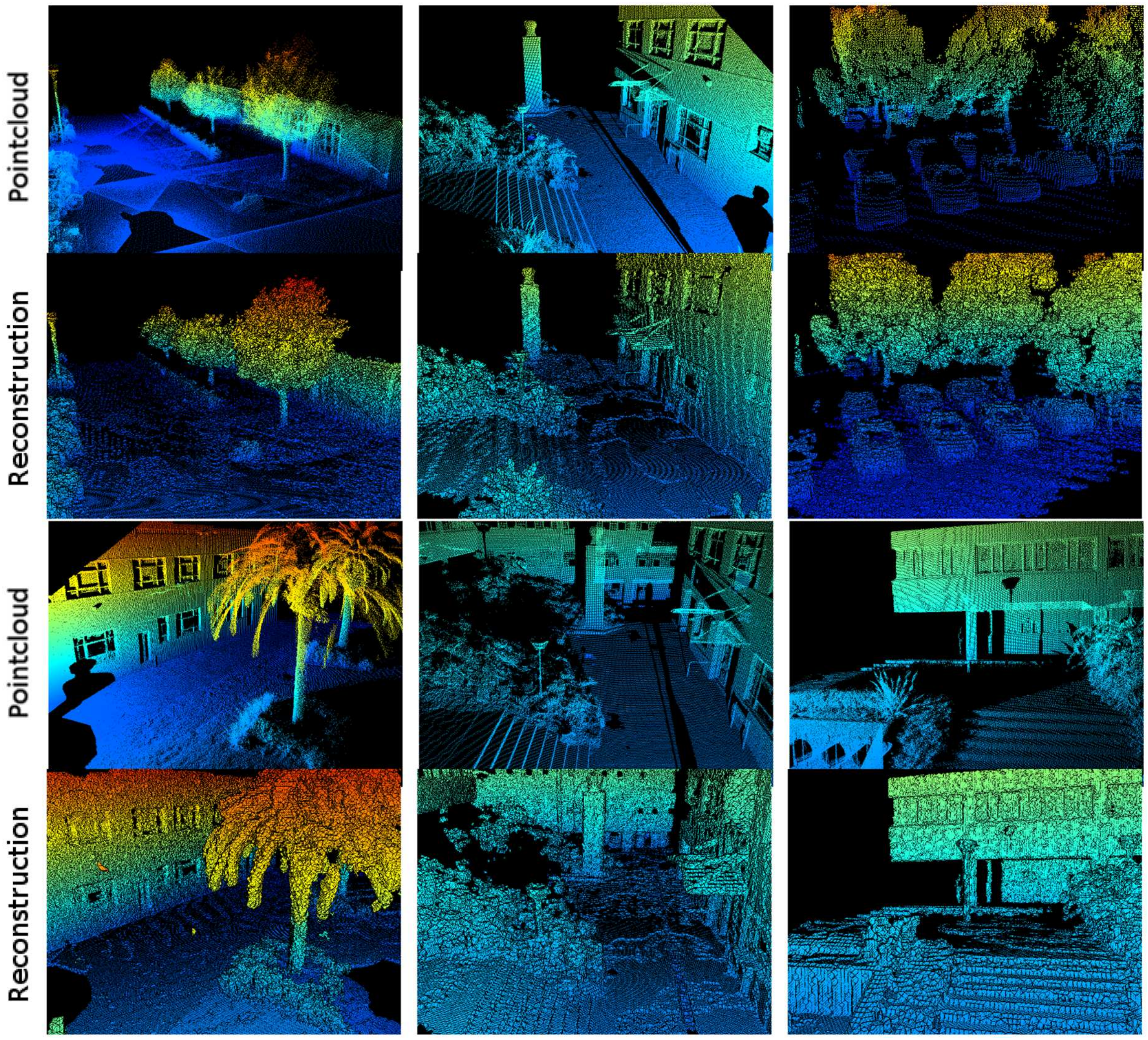


Fig. 6. The 3D scene reconstruction results from Table 2, using CVAE-HM on laser data. Note how the reconstructive model is able to reason over sparse pointclouds to produce a more solid representation of structures, and also extrapolates available information to areas not covered by sensors. It is also capable of completing objects based on partial views, such as cars (top right image) and trees (top left and right images). Interestingly, it learns to consistently fill in gaps produced by shadows, completing partially occluded structures (top and bottom left images).

expected, smaller cluster sizes produce better classification results, since the structures to be learned are simpler; however, they are also only able to reconstruct smaller gaps. As the average cluster size increases, larger gaps can be reconstructed; however, the reconstructive model itself starts to suffer, because it is unable to learn such complex features in the first place. As an empirical observation, we estimate that the gap size should be roughly equal to half the average cluster size for an optimal reconstruction. Note that these cluster sizes are much larger than those usually found in the literature for similar tasks, such as object detection and scene reconstruction (Song and Xiao, 2014; Lai et al., 2014; Ruhnke et al., 2010).

Lastly, in Figures 6 and 7, we can see 3D reconstruction results in areas that were not deliberately removed from the evaluation dataset, such as shadows and partial occlusions, for both 3D laser and monocular data. Owing to the large abundance of similar objects in the training dataset, and large enough features to encompass a significant portion of the structure, the reconstructive model was able to reason over sparser areas and data gaps to recover original unobserved shapes, including color information when available. The use of a higher resolution to generate the grid representations would most probably result in more detailed reconstructions; however, owing to the high computational cost and memory requirements, this assumption was not explored here and is left for future work. In fact, the occupancy maps depicted here take roughly 5 s to produce from raw input pointcloud, using a modern GPU-enabled computer (Titan Pascal X), making the proposed approach

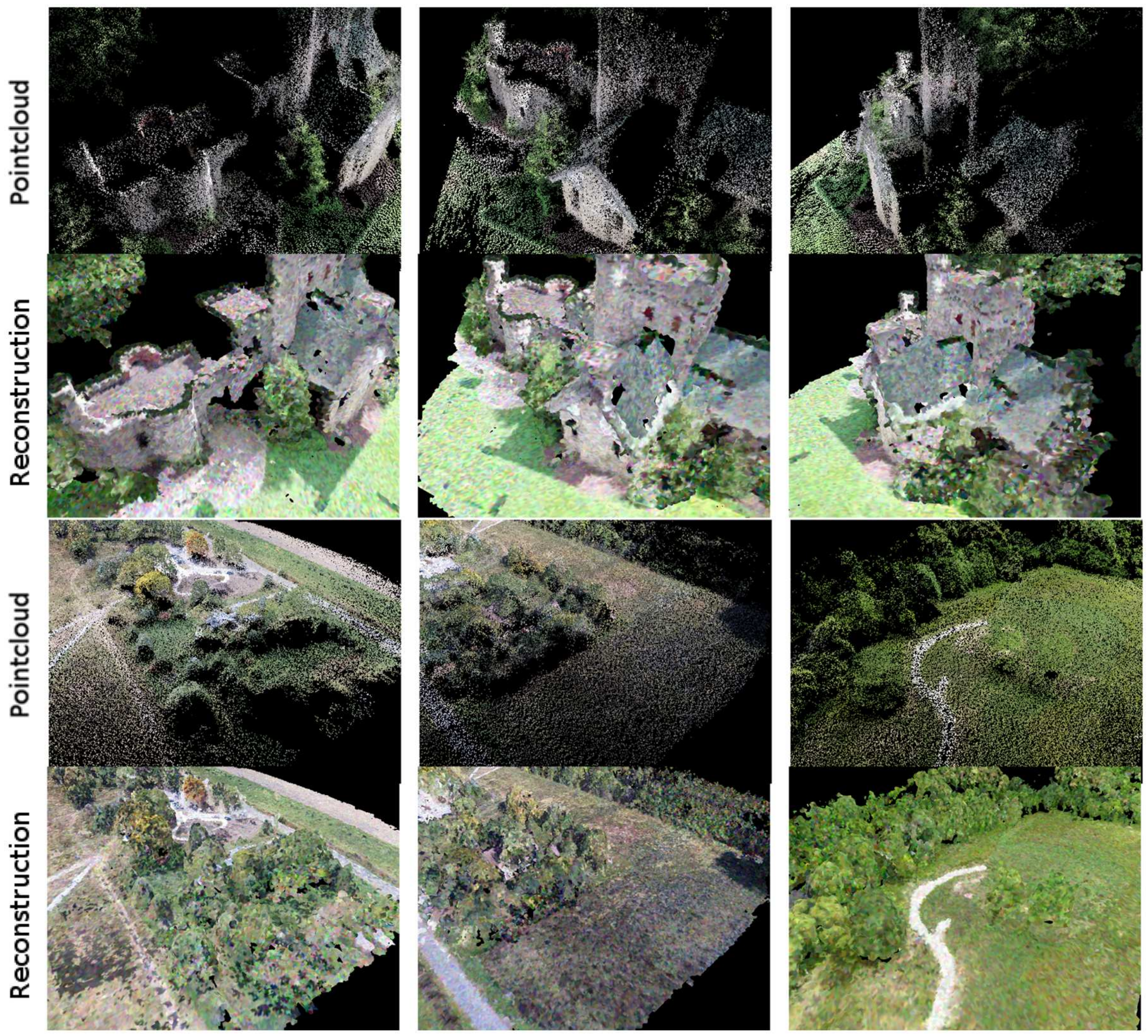


Fig. 7. The 3D scene reconstruction results from Table 3, using CVAE-HM on monocular data. Here, both occupancy values and RGB colors are learned, and the reconstructive model is able to extrapolate this information to areas not covered by sensors or without enough texture for visual feature extraction.

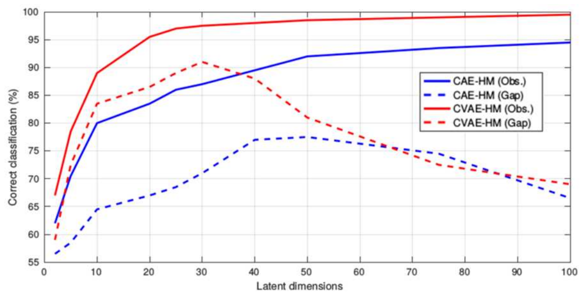


Fig. 8. Effects of changing the number of latent dimensions in CAE-HM and CVAE-HM, for the 3D test dataset.

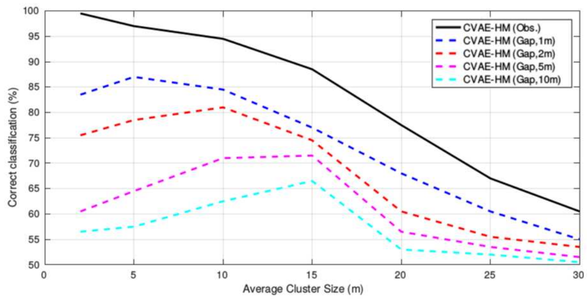


Fig. 9. Effects of changing the size of gaps in CVAE-HM for different average cluster sizes, for the 3D test dataset.

still incapable of online estimation. This is mostly because of the high computational cost and memory requirements of 3D convolution, which will also be the subject of future research.

5. Conclusion

This paper introduces a novel methodology for 3D occupancy mapping, which utilizes CVAEs to learn a low-dimensional manifold of observed structures. Once this reconstructive model is trained, new structures can be encoded and then decoded to produce occupancy estimates, that are then combined using the Hilbert maps framework. While not achieving the level of detail currently found in other state-of-the-art reconstruction techniques when dealing with observed information, the proposed methodology is able to consistently reason over data gaps and partial occlusions with an accuracy significantly greater than any of the other techniques considered here.

Future work will address neural network over-fitting while focusing on performance and level of detail, particularly through the use of continuous convolutions and the sparse representation recently introduced by Riegler et al. (2016). The use of different loss functions, as described by Pathak et al. (2016), will also be explored as a way to improve convergence speed and level of detail. Furthermore, we will explore color reconstruction when this information is not available in the first place (i.e. estimating color values in laser data based on a model trained on monocular data), as a way to improve environment representation without introducing extra sensors during inference time. From an application perspective, the proposed framework will be deployed in autonomous navigation tasks, providing a means to reconstruct unobserved areas of the environment, owing to sensor failures or occlusions, and thus improve the performance of path planning algorithms.

Acknowledgments

The GPU cards used to conduct this work were donated by the NVIDIA corporation.

Funding

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

ORCID iD

Vitor Guizilini https://orcid.org/0000-0002-8715-8307

Notes

1. Only integer values for occupancy were used in this work; however, the framework can be trivially modified to reason over occupancy probabilities in the range $[-1, +1]$.

References

Arthur D and Vassilvitskii S (2007) k-means++: The advantages of careful seeding. In: *ACM-SIAM symposium on discrete algorithms (SODA)*, New Orleans, LA, 7–9 January 2007, pp. 1027–1035. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Bahmani B, Moseley B, Vattani A, et al. (2012) Scalable k-means++. *Proceedings of the VLDB Endowment* 5(7): 622–633.

Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: *Proceedings of the international conference on computational statistics (COMPSTAT)* (eds. Y Lechevallier and G Saporta), Paris, France, 22–27 August 2010, pp. 177–186. Berlin: Springer-Verlag.

Ciresan D, Meier U, Masci J, et al. (2011) Flexible, high performance convolutional neural networks for image classification. In: *22nd international joint conference on artificial intelligence (IJCAI)*, Barcelona, Spain, 19–22 July 2011. Palo Alto, CA: AAAI Press.

Deng L and Yu D (2014) Deep learning: Methods and applications. *Foundations and Trends in Signal Processing* 7(3–4): 197–387.

Doersch C (2016) Tutorial on variational autoencoders. *arXiv* arXiv:1606.05908.

Dragiev S, Toussaint M, and Gienger M (2011) Gaussian process implicit surfaces for shape estimation and grasping. In: *IEEE international conference on robotics and automation (ICRA)*, Shanghai, China, 9–13 May 2011, pp. 2845–2850. Piscataway, NJ: IEEE.

Elfes A (1989) *Occupancy grids: A probabilistic framework for robot perception and navigation*. Ph.D. Thesis, Carnegie Mellon University, USA.

Ghiasi G and Fowlkes C (2016) Laplacian pyramid reconstruction and refinement for semantic segmentation. In: Leibe B, Matas J, Sebe N, and Welling M (eds.) *Computer Vision: ECCV 2016* (*Lecture Notes in Computer Science*, vol. 9907). Cham: Springer.

Glorot X, Bordes A, and Bengio Y (2011) Deep sparse rectifier neural networks. *Proceedings of Machine Learning Research* 15: 315–323.

Guizilini V and Ramos F (2016) Large-scale 3D scene reconstruction with Hilbert maps. In: *IEEE international conference on intelligent robots and systems (IROS)*, Daejeon, South Korea, 9–14 October 2016, pp. 3247–3254. Piscataway, NJ: IEEE.

Guizilini V and Ramos F (2017a) Learning to reconstruct 3D structures for occupancy mapping. In: *Robotics: Science and systems* (eds. N Amato, S Srinivasa, N Ayanian, et al.), Cambridge, MA, USA: MIT Press, 12–16 July 2017.

Guizilini V and Ramos F (2017b) Unsupervised feature learning for 3D scene reconstruction with occupancy maps. In: *31st AAAI conference on artificial intelligence*, San Francisco, CA, USA, 4–10 February 2017. Palo Alto, CA: AAAI Press.

He K, Zhang X, Ren S, et al. (2016) Deep residual learning for image recognition. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, Las Vegas, NV, USA, 27–30 June 2016, pp. 770–778. Piscataway, NJ: IEEE.

Hinton G (2002) Training products of experts by minimizing contrastive divergence. *Neural Computation* 14(8): 1771–1800.

Hinton G and Salakhutdinov R (2006) Reducing the dimensionality of data with neural networks. *Science* 5786(313): 504–507.

Hochreiter S and Schmidhuber J (1999) Feature extraction through LOCOCODE. *Neural Computation* 11(3): 679–714.

Hornung A, Wurm K, Bennewitz M, et al. (2013) Octomap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots* 34(3): 189–206.

Jadidi M, Miro J, and Dissanayake G (2017) War ped Gaussian processes occupancy mapping with uncertain inputs. *IEEE Robotics and Automation Letters* 2(2): 680–687.

Kanungo T, Mount D, Netanyahu N, et al. (2002) An efficient *k*-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(7): 881–892.

Kim S and Kim J (2015) GPmap: A unified framework for robotic mapping based on Gaussian processes. In: Mejias L, Corke P, and Roberts J (eds.) *Field and Service Robotics* (*Springer Tracts in Advanced Robotics*, vol. 105). Cham: Springer, pp. 319–332.

Klambauer G, Unterthiner T, Mayr A, et al. (2017) Self-normalizing neural networks. *arXiv* arXiv:1706.02515.

Komarek P (2004) *Logistic regression for data mining and high-dimensional classification*. Ph.D. Thesis, Carnegie Mellon University, USA.

Lai K, Bo L, and Fox D (2014) Unsupervised feature learning for 3D scene labeling. In: *IEEE international conference on robotics and automation (ICRA)*, Hong Kong, China, 31 May–7 June 2014, pp. 3050–3057. Piscataway, NJ: IEEE.

LeCun Y, Chopra S, Hadsell R, et al. (2006) *A Tutorial on Energy-Based Learning*. Cambridge, MA: MIT Press.

Masci J, Meier U, Ciresan D, et al. (2011) Stacked convolutional auto-encoders for hierarchical feature learning. In: Honkela T, Duch W, Girolami M, et al. (eds.) *Artificial Neural Networks and Machine Learning: ICANN 2011* (*Lecture Notes in Computer Science*, vol. 6791). Berlin: Springer, pp. 52–59.

Newcombe R, Izadi S, Hilliges O, et al. (2011) KinectFusion: Real-time dense surface mapping and tracking. In: *10th IEEE international symposium on mixed and augmented reality*, Basel, Switzerland, 26–29 October 2011, pp. 127–136. Piscataway, NJ: IEEE.

Nüchter A, Lingemann K, Hertzberg J, et al. (2007) 6D SLAM—3D mapping outdoor environments. *Journal of Field Robotics* 24(8): 699–722.

O'Callaghan S, Ramos F, and Durrant-Whyte H (2009) Contextual occupancy maps using Gaussian processes. In: *IEEE international conference on robotics and automation (ICRA)*, Kobe, Japan, 12–17 May 2009, pp. 1054–1060. Piscataway, NJ: IEEE.

Pathak D, Krähenbühl P, Donahue J, et al. (2016) Context encoders: Feature learning by inpainting. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, Las Vegas, NV, USA, 27–30 June 2016, pp. 2536–2544. Piscataway, NJ: IEEE.

Powers DMW (2011) Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies* 2(1): 37–63.

Pu Y, Gan Z, Henao R, et al. (2016) Variational autoencoder for deep learning of images, labels and captions. In: *30th international conference on neural information processing systems (NIPS'16)*, Barcelona, Spain, 5–10 December 2016, pp. 2360–2368. Red Hook, NY: Curran Associates Inc.

Ramos F and Ott L (2015) Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. In: *Robotics: Science and systems* (eds. LE Kavraki, D Hsu, and J Buchli), Rome, Italy: MIT Press Cambridge, MA, USA, 13–17 July 2015.

Ranzato M, Huang F, and LeCun Y (2007) Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, Minneapolis, MN, USA, 17–22 June 2007. Piscataway, NJ: IEEE.

Rasmussen C and Williams C (2005) *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press.

Ren S, He K, Girshick R, et al. (2015) Faster R-CNN: Towards real-time object detection with regional proposal networks. In: *Advances in neural information processing systems (NIPS)*, Montréal, Canada: MIT Press Cambridge, MA, USA, 7–12 December 2015.

Riegler G, Ulusoy A, and Geiger A (2016) Octnet: Learning deep 3D representations at high resolutions. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, Honolulu, HI, USA, 21–26 July 2017, pp. 6620–6629. Piscataway, NJ: IEEE.

Ruhnke M, Steder B, Grisetti G, et al. (2010) Unsupervised learning of compact 3D models based on the detection of recurrent structures. In: *IEEE international conference on intelligent robots and systems (IROS)*, Taipei, Taiwan, 18–22 October 2010, pp. 2137–2142. Piscataway, NJ: IEEE.

Sansone G (2012) *Orthogonal Functions: Revised English Version*. New York, NY: Dover Books on Mathematics.

Scherer D, Muller A, and Behnke S (2010) Evaluation of pooling operations in convolutional architectures for object recognition. In: Diamantaras K, Duch W, and Iliadis LS (eds.) *Artificial Neural Networks: ICANN 2010* (*Lecture Notes in Computer Science*, vol. 6354). Berlin: Springer, pp. 92–101.

Schölkopf B, Muandet K, Fukumizu K, et al. (2015) Computing functions of random variables via reproducing kernel Hilbert space representations. *Statistics and Computing* 25(4): 755–766.

Senanayake R, Ott L, Callaghan S, et al. (2016) Spatio-temporal Hilbert maps for continuous occupancy representation in dynamic environments. In: *Advances in neural information processing systems (NIPS)*, Barcelona, Spain, 5–10 December 2016, pp. 3925–3933. Red Hook, NY: Curran Associates Inc.

Shepard D (1968) A two-dimensional interpolation function for irregularly-spaced data. In: *23rd ACM national conference*, Las Vegas, NV, USA, 27–29 August 1968, pp. 517–524. New York, NY: ACM.

Song S and Xiao J (2014) Sliding shapes for 3D object detection in depth images. In: Fleet D, Pajdla T, Schiele B, et al. (eds.) *Computer Vision: ECCV 2014* (*Lecture Notes in Computer Science*, vol. 8694). Cham: Springer. pp. 634–651.

Srivastava N, Hinton G, Krizhevsky A, et al. (2014) Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15: 1929–1958.

Waechter M, Moehrle N, and Goesele M (2014) Let there be color! Large-scale texturing of 3D reconstructions. In: Fleet D, Pajdla T, Schiele B, et al. (eds.) *Computer Vision: ECCV 2014* (*Lecture Notes in Computer Science*, vol. 8693). Cham: Springer, pp. 836–850.

Wu Z, Song S, Khosla A, et al. (2015) 3D ShapeNets: A deep representation for volumetric shapes. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, Boston, MA, USA, 7–12 June 2015, pp. 1912–1920. Piscataway, NJ: IEEE.

Zeiler M and Fergus R (2014) Visualizing and understanding convolutional networks. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. In: Fleet D, Pajdla T, Schiele B, et al. (eds.) *Computer Vision: ECCV 2014* (*Lecture Notes in Computer Science*, vol. 8689). Cham: Springer, pp. 818–833