Learning to Reconstruct 3D Structures for Occupancy Mapping

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

***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 tasks such as navigation and grasping. This paper devises a novel unsupervised procedure that is able to learn 3D structures from unorganized point clouds as occupancy maps. Our framework enables the learning of unique and arbitrarily complex features using a Bayesian Convolutional Variational Auto-Encoder that compresses local information into a latent low-dimensional representation and then decodes it back in order to reconstruct the original scene. This reconstructive model is trained on features obtained automatically from a wide variety of scenarios to improve its generalization and interpolative powers. We show that the proposed framework is able to recover partially missing structures and reason over occlusion with high accuracy, while maintaining a detailed reconstruction of observed areas. To seamlessly combine this localized feature information into a single global structure, we employ a Hilbert Map, recently proposed as a robust and efficient occupancy mapping technique. Experimental tests are conducted in large-scale 2D and 3D datasets, and a study on the impact of various accuracy/speed trade-offs is provided to assess the limits of the proposed framework.**

I. 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 this data, in a way that can be exploited by both humans 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 the 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 [8, 24], maintaining an equally-sized grid that stores independent information about each particular area. This approach, however, is very memory-intensive and it does not take into account spatial relationships between cells. OctoMaps (OM) [17] use a tree-like structure to recursively divide the space as necessary, which results in a massive decrease in memory requirement and processing time for grid-like occupancy mapping. Gaussian Process Occupancy Maps (GPOM) [25] and Gaussian Process Implicit Surfaces (GPIS) [7] 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.

A more recent approach, Hilbert Maps (HM) [26], projects data points into a higher-dimensional space and uses simple linear classifiers to produce a continuous probabilistic occupancy function. The resulting framework both maintains 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 year, significant effort has been made to extend this framework to address larger datasets [11] and the learning of more complex features [12]. However, these works are restricted in the type of features that can be learned from data. In [11] a squared-exponential kernel is used, which restricts the resulting features to ellipsoids, while [12] proposes the addition of a planar surface kernel and devises 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 Bayesian convolutional neural network (CNN) trained on a series of features extracted from previously observed maps. Such learning architecture has already been successfully applied to several areas of computer vision, i.e. image classification [13], object detection [28] and semantic segmentation [9]; and is now transitioning into tridimensional information, as more 3D capturing techniques [23] and large-scale repositories [36] become available.

In particular, we will be focusing on Convolutional Auto-Encoders [22] due to their ability to reason over spatial information to produce latent low-dimensional representations; and their variational counterpart [6], that infuses Bayesian probabilistic inference into the Deep Learning framework to produce a generative model for new structures. To the best of our knowledge, this is the first time Deep Variational Auto-Encoders are derived to learn complex environment features for 3D occupancy mapping, significantly reducing gaps in the model as a result of occlusions. Within the Hilbert Maps framework, the benefits of such approach are:

- **More detailed reconstructions**. The learned features are not restricted to any single 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 a good reconstruction.
- **Better reasoning over data gaps**. The model uses its learned filters to reconstruct unobserved parts of the environment, thus dealing with partial occlusions and sensor failure.

II. 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 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.

A. Hilbert Maps

In [26] a novel framework for scene reconstruction was proposed, 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 the *Hilbert space* [31] 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* (RKHS) [33].

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, while the discretized space traversed by each beam is treated as unoccupied. These input points are projected into the RKHS using a feature function $\Phi(\mathbf{x})$, and a classifier is trained in this higher-dimensional space to produce a discriminative model $p(y|\mathbf{x}, \mathbf{w})$, where $\mathbf{w}$ are the parameters of this classifier. Due to the common wisdom that linear separators are almost always adequate to separate classes in high-dimensional spaces [19], a simple classifier can be used for this task.

B. Convolutional Auto-Encoders

The main purpose of unsupervised learning methods is to extract useful features in unlabeled data, removing input redundancies while preserving its essential aspects, that are then used to produce robust and discriminative representations. Within this context, the *encoder-decoder* paradigm is arguably the most commonly used [15], in which 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 (LOCOCODE) [16], Auto-Encoders [27], Energy-Based Models [21] and Restricted Boltzmann Machines (RBM) [14].

Recently, deep architectures have taken over most learning tasks [5], providing an unprecedented level of pattern recognition and data abstraction that is inspired by biological systems. In particular, convolutional neural networks (CNNs) excel with visual information [4], because they preserve the input's spatial locality and neighborhood information in latent higher-level feature representations. In contrast to fully connected deep architectures, that do not scale well to high-dimensional inputs in terms of computational complexity, the number of free parameters in a CNN does not depend on input dimensionality, since they are locally shared in each layer.

The concept of Convolutional Auto-Encoders (CAE) originated in [22], as a hierarchical unsupervised feature extractor that uses stochastic gradient descent to learn good CNN 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:

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

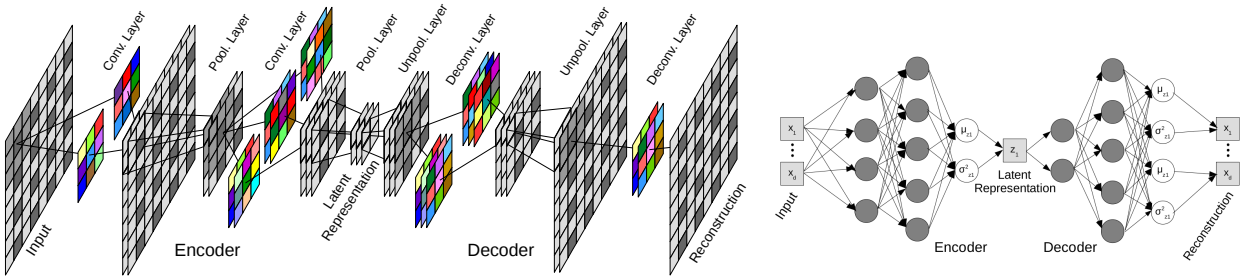
with parameters $\theta' = \{\tilde{W}, c\}$ usually constrained such that $\tilde{W} = W^T$, i.e. the same weights are used for encoding the input and decoding the latent representation. These parameters are optimized by minimizing an appropriate cost function over the training set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, usually the mean squared error (MSE) value 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 backpropagation algorithm is applied to compute the gradient of the error function with respect to the parameters. For Eq. 3, this can be easily obtained by convolution operations using the following formula:

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

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 (Fig. 1a). A max-pooling layer [32] 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.



(a) Convolutional Auto-Encoder.

(b) Variational Auto-Encoder.

Fig. 1: Visual representations of the Deep Learning architectures used in this paper.

C. Variational Auto-Encoders

Even though commonly classified as of the same type, the Variational Auto-Encoder (VAE) actually has little to do with classical auto-encoders [6]. In a nutshell, a VAE 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 our dataset. In other words, it aims to optimize the probability of X according to:

$$P(X) = \int f(z, \theta) P(z) dz = \int P(X|z, \theta) P(z) dz. \quad (5)$$

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 Fig. 1b. 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 [6], here omitted for brevity, the lower bound to be optimized becomes:

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

In the above equation, on the first line we are attempting to maximize $P(X)$ while simultaneously minimizing the KL-divergence $KL[Q(z|X)||P(z|X)]$ (i.e. trying to approximate the two functions). On 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. This 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.

III. 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 is 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.

A. Automatic Feature Extraction

We start with an unorganized point cloud $\mathcal{D} = \{\mathcal{X}, \mathbf{y}\} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ containing the spatial coordinate of each point and its corresponding occupancy value, as described in Sec. II-A. This unorganized point cloud is clustered, to produce different local structures that will serve as features and together describe the environment as a whole. In [12] an alternative to the standard k-means++ initialization algorithm [1] was proposed, that selects the starting cluster seeds for further optimization, i.e. using the standard k-means algorithm [18]. This technique was shown to outperform k-means++ in terms of speed while allowing the automatic determination of the number of clusters necessary to properly describe the environment, given a distance threshold.

Here, we build upon the ASK-Means algorithm described in [12] and propose Quick-Means, an extension that further improve its computational speed without significantly compromising accuracy, as it can be seen in Fig. 2. The key

Algorithm 1 Quick-Means initialization algorithm

Require: point cloud $\mathcal{X}$ with N points
radial inner r_i and outer r_o thresholds
minimum number of points per cluster k
distance function $d(.,.)$
Ensure: clusters $\mathcal{C}$

$t \leftarrow N$ % Number of available points
$v_i \leftarrow \{0, 1, 2, \ldots, N\}$ % Aux. index vector
$v_j \leftarrow \{0, 1, 2, \ldots, N\}$ % Aux. index vector
$v_b \leftarrow \{0, 0, 0, \ldots, 0\}_{i=1}^{N}$ % Aux. boolean vector
$\mathcal{C} \leftarrow \{\}$ % Empty cluster set
while $t > 0$ **do**
  $\mathbf{x}_* \leftarrow \mathcal{X}[v_i[rand(0, t)]]$
  $\mathcal{M} \leftarrow \mathbf{x} \mid d(\mathbf{x}_*, \mathbf{x}) < r_o \ , \forall \mathbf{x} \in \mathcal{X}[v_i[0, 1, \ldots, t]]$
  if $|\mathcal{M}| > k$ **then**
    $\mathcal{C} \leftarrow \mathcal{M}$
  end if
  for $\mathbf{m} \in \mathcal{M}$ **do**
    $i \leftarrow$ index of $\mathbf{m}$ in $\mathcal{X}$
    if $v_b[i] == 0$ **and** $d(\mathbf{m}, \mathbf{x}) < r_i$ **then**
      $v_b[i] = 1 \ , \ j \leftarrow v_j[i]$
      $v_i[j] = v_i[\text{--}t]$
      $v_j[t] = v_j[v_i[j]] = j$
    end if
  end for
end while

insight is that, for the particular application at hand, we are not interested in precise cluster locations, but rather at regularly-placed clusters, that are within a given inner radius threshold r_i given a distance metric $d(.,.)$. Furthermore, it accommodates cluster overlapping by using an outer radius threshold $r_o > r_i$, as a way to increase feature complexity and model structures from different perspectives. Pseudo-code for the Quick-Means initialization algorithm can be found in Alg. 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 (Fig. 3b), at a certain resolution r_G and with dimensionality $d = \lceil r_G^{-1} \rceil^D$. Each coordinate is populated with the most common occupancy value for data points that fall within that cell (-1 for unoccupied, 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:

$$k(\mathbf{x}_*, G) = \begin{cases} 0 & \text{if } \mathbf{x}_* \text{ is outside } G \\ y_j^G \text{ for } \mathbf{x}_j^G \text{ closest to } \mathbf{x}_* & \text{otherwise,} \end{cases} \quad (7)$$

The feature vector Φ that projects an input point $\mathbf{x}$ into the RKHS (see Eq. 14) is given by a vector containing the kernel values produced by all extracted clusters $\mathcal{G}$, as shown in Eq. 8. 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:

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

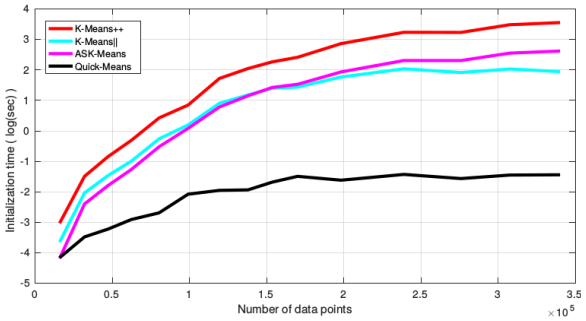
$$\Phi(\mathbf{x}, \mathcal{G}) = \begin{bmatrix} k(\mathbf{x}, G_1) \\ k(\mathbf{x}, G_2) \\ \vdots \\ k(\mathbf{x}, G_M) \end{bmatrix}. \quad (8)$$

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 Eq. 9) by: translation of $\bar{G}_m$ to a zero-median position; rotation so their sorted eigenvectors V_m are, from largest to smallest eigenvalue, aligned with the coordinate system; and scaling by s so their largest dimension becomes unitary (Fig. 3c). These transformations are stored to be used later, in order to recover the original grid representation from an aligned reconstructed grid:

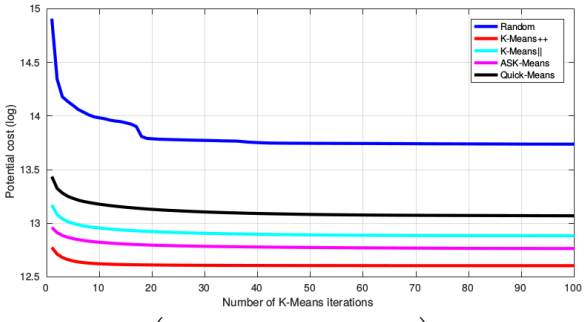
$$H_m = \frac{1}{s}\left((G_m - \bar{G}_m)V_m\right). \quad (9)$$

B. 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 issues like interpolation, extrapolation, data gaps, partial occlusion and so forth. Because of that, we propose the use of a reconstructive model that, as the name implies, takes these feature vectors and reconstructs the

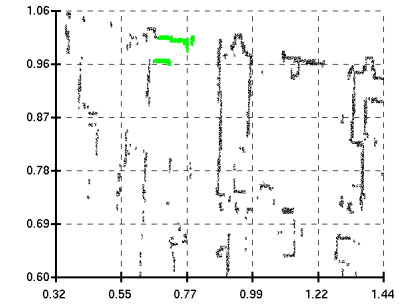


(a) Initialization time (random values are negligible).

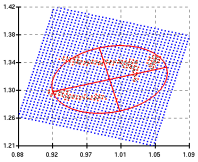


(b) Potential cost $\left(\underset{S}{\text{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} ||\mathbf{x} - \boldsymbol{\mu}_i||^2\right)$ during k-means.

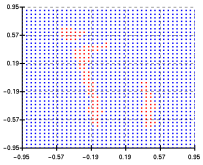
Fig. 2: Comparison between different clustering initialization techniques: Random, k-means++ [1], k-means|| [2], ASK-Means [12] and the proposed Quick-Means algorithm (average of 50 runs with different random seeds).



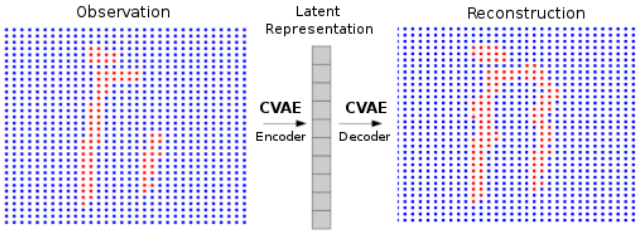
(a) Unorganized point cloud (green dots indicate data from the selected cluster)



(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: 2D feature extraction and reconstruction process.

observed structure based on how it should look like, according to prior information collected from other similar datasets.

Here, this reconstructive model is a Convolutional Variational Auto-Encoder (CVAE), that combines the concepts found in Sec. IIb-c into a single framework. The use of convolutional layers preserves spatial relationships and greatly decreases the number training parameters, while variational approximations are able to encode information into a significantly smaller latent representation, as it will be shown during experiments. Given an aligned reconstructed grid H as input, the resulting encoded vectors (i.e. the latent variables z, as introduced in Eq. 5) are generated from the following probabilistic distribution:

$$Q(\mathbf{z}|H) = \mathcal{N}\left(\mathbf{z}|\boldsymbol{\mu}(H), \boldsymbol{\sigma}^2(H)\right). \tag{10}$$

As a simple 2D numerical example, we start with a $32 \times 32 \times 1$ aligned grid representation as shown in Fig. 3c. During the encoding process, this grid goes through a series of convolutional layers, each with a kernel size of 5×5, stride of 1, max-pooling of 2 and a ReLU activation function [10]. 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 Fig. 1a). The number of channels, on the other hand, increases as more kernels filters are used to simultaneously process the same input. For the example at hand, the first layer produces 64 channels and each one afterwards doubles this value, up to a maximum of 256.

Once this process is complete, the output $\mathbf{y}$ of the last convolutional layer is used to produce the mean and variance values (see Eq. 10) that compose the latent representation $\mathbf{z}$, as depicted in Fig. 1b. Here, the generation of these mean and variance values is defined as:

$$\boldsymbol{\mu}(H) = Fully(\mathbf{y}) \tag{11}$$

$$\boldsymbol{\sigma}(H) = Softplus(Fully(\mathbf{y})), \tag{12}$$

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 $Fully(.)$, that 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 as their convolutional counter-parts, and is expanded using an unpooling operation (i.e. the input grid is resized to its required output dimensions). The resulting reconstructed grid H'_m has the same size as the original representation, and contains occupancy estimates according to the CVAE model. An example of this process can be seen in Fig. 3d, 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 (Eq. 13) and is then used to produce the reconstructed feature vector $\Phi(\mathbf{x}, \mathcal{G}')$, as shown in Eq. 8:

$$G'_m = sV_m^T H'_m + \bar{G}_m. \tag{13}$$

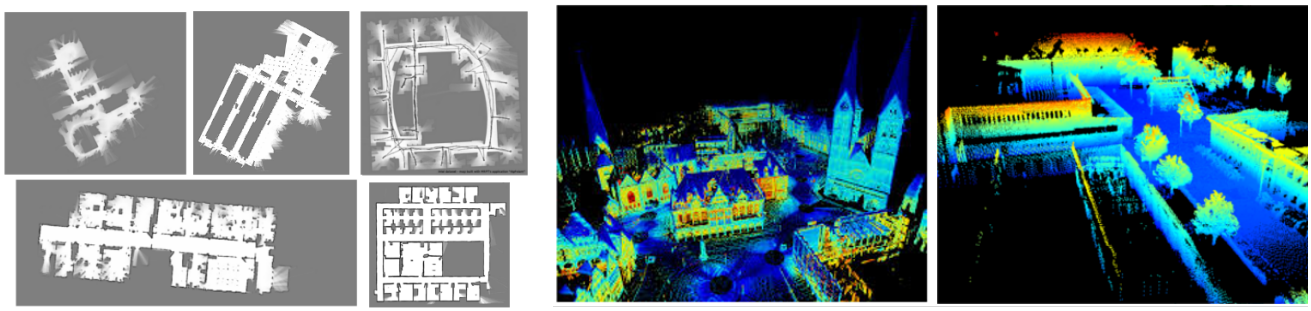
The CVAE model is trained using mini-batches of reconstructed aligned feature vectors, obtained from clusters extracted from the training datasets, which are assumed to contain similar structures to the ones found in the evaluation dataset. As explained in Sec. II-C, the reconstruction error (i.e. similarity 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 [35] value of 0.7 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.

C. 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 Sec. II-A. Here we employ a Logistic Regression (LR) classifier, in which the probability of occupancy for a query point $\mathbf{x}_*$ is given by:

$$p(y_* = 1|\Phi'(\mathbf{x}_*), \mathbf{w}) = \frac{1}{1 + \exp\left(\mathbf{w}^T \Phi'(\mathbf{x}_*)\right)}, \tag{14}$$

where the dependencies on $\mathcal{G}'$ were removed for notation clarity. To optimize the weight parameters $\mathbf{w}$ based on information



(a) 2D indoor datasets.

(b) 3D outdoor datasets.

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

contained in $\mathcal{D}$, we minimize the *Regularized Negative Log-Likelihood* (RNLL) function:

$$RNLL(\mathbf{w}) = \sum_{i=1}^{N} \left(1 + \exp\left(-y_i \mathbf{w}^T \Phi(\mathbf{x}_i)\right)\right) + R(\mathbf{w}), \quad (15)$$

in which $R(\mathbf{w})$ is a regularization function, used to prevent overfitting and promote sparseness in $\mathbf{w}$. An useful property of Eq. 15 is its suitability for *Stochastic Gradient Descent* (SGD) optimization [3], in which information contained in each point, or batch of points, provides one small step towards a local minimum, calculated as such:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \frac{\delta}{\delta \mathbf{w}} RNNL(\mathbf{w}), \quad (16)$$

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 due to sheer size and memory requirements.

IV. Experimental Results

The proposed framework was tested using 2D and 3D datasets collected from various public repositories available on the Internet, as depicted in Fig. 4. After one dataset is selected for evaluation, the other ones are clustered 20 times according to Sec. III-A with different random seeds, to produce grid representations of observed structures. These grid representations are then used to train the reconstructive model described in Sec. III-B (two different models were trained, one for 2D and another for 3D datasets). Afterwards, the reconstructed feature vectors are used train a Hilbert Map that produces occupancy values for any point in the input space, as depicted in Sec. III-C. 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.

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, that produces each extracted feature.
- **Gap ratio.** The effects of changing the relative size of data gaps, in relation to average cluster size.

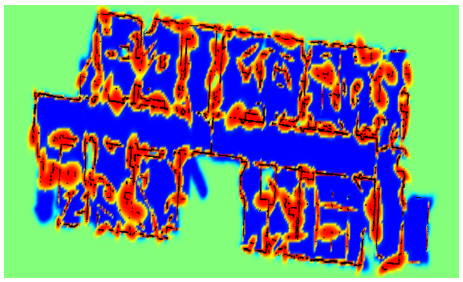
As a baseline, we selected a grid size of 64, a latent dimension of $25/100$ for 2D/3D input data, a network depth of 4 convolutional layers (with kernel sizes $9/7/5/5$, max-pooling of 2 and dropout of 0.8 on the last one) and a gap ratio of 50% in relation to an average cluster size of 5 meters. The classification results, for 2D and 3D datasets, can be respectively found in Tables I and II. 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 in [11] for both 2D and 3D scenarios; and Gaussian Process Occupancy Maps (GPOM) [25] for 2D datasets and OctoMaps [17] for 3D datasets. The GPOM 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

TABLE I: Classification results for 2D datasets.

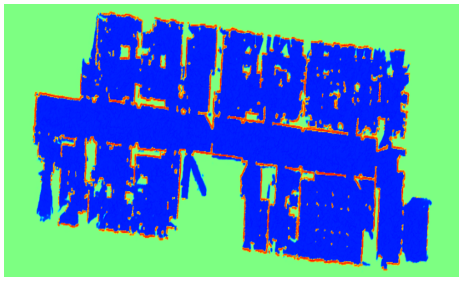
Method	Data Observed		Data Gaps	
	Precision	Recall	Precision	Recall
GPOM	90.60%	80.83%	34.61%	29.94%
LARD-HM	80.44%	99.70%	53.58%	51.71%
CAE-HM	77.82%	74.89%	45.22%	42.45%
CVAE-HM	96.64%	94.75%	90.86%	86.29%

TABLE II: Classification results for 3D datasets.

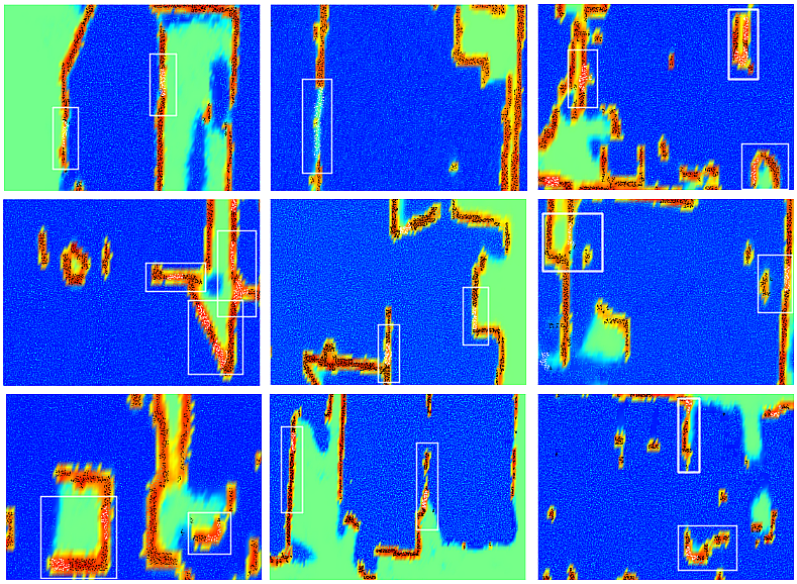
Method	Data Observed		Data Gaps	
	Precision	Recall	Precision	Recall
OctoMap	95.59%	94.99%	16.85%	21.94%
LARD-HM	86.08%	96.58%	40.20%	38.62%
CAE-HM	61.92%	64.53%	36.20%	34.26%
CVAE-HM	92.91%	89.68%	86.02%	81.27%



(a) LARD-HM results.



(b) CVAE-HM results.



(c) Zoomed in 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 unkonwn areas.

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

occupied if its occupancy probability is higher than 60% and unoccupied if this probability is smaller than 40% (values in between are considered unknown).

As shown in Tables I and II, 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 Figs. 5 and 8). Additionally, these results show that the introduction of a variational component into the Convolutional Auto-Encoder framework is crucial in achieving satisfactory reconstruction results, especially with smaller latent dimensions. A comparative study showing the effects of changing this dimensionality can be found in Fig. 6, where we see that CAE-HM requires more dimensions to converge and still achieves worse results both in terms of 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 ignore the presence of information gaps. Fine-tuning the network topology would probably address this shortcoming and produce better results, however this was not explored here and is left for future work.

Another comparative study was done in relation to average cluster size, that dictates the size of features extracted from the environment and, by extension, their complexity. Fig. 5a shows that the standard LARD-HM framework is already unable to reconstruct environments with an average cluster size of $5m$, while CVAE-HM is able to achieve a much more detailed

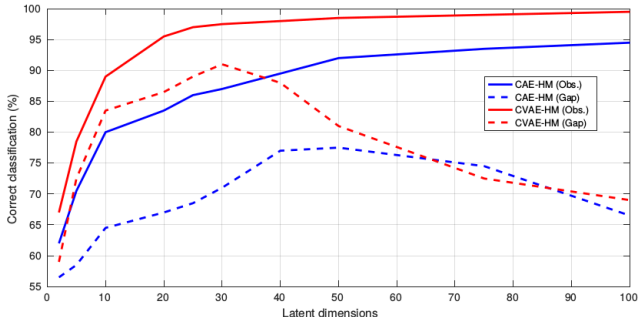


Fig. 6: Effects of changing the number of latent dimensions in CAE-HM and CVAE-HM.

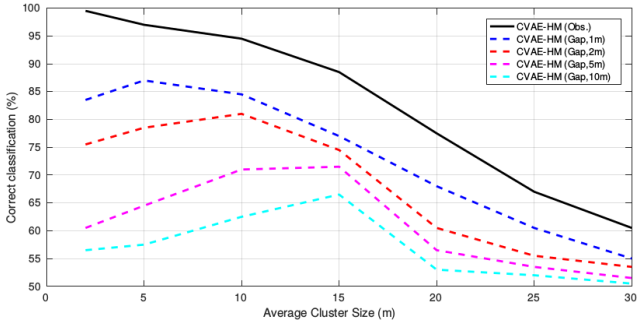


Fig. 7: Effects of changing the size of gaps in CVAE-HM for different average cluster sizes.

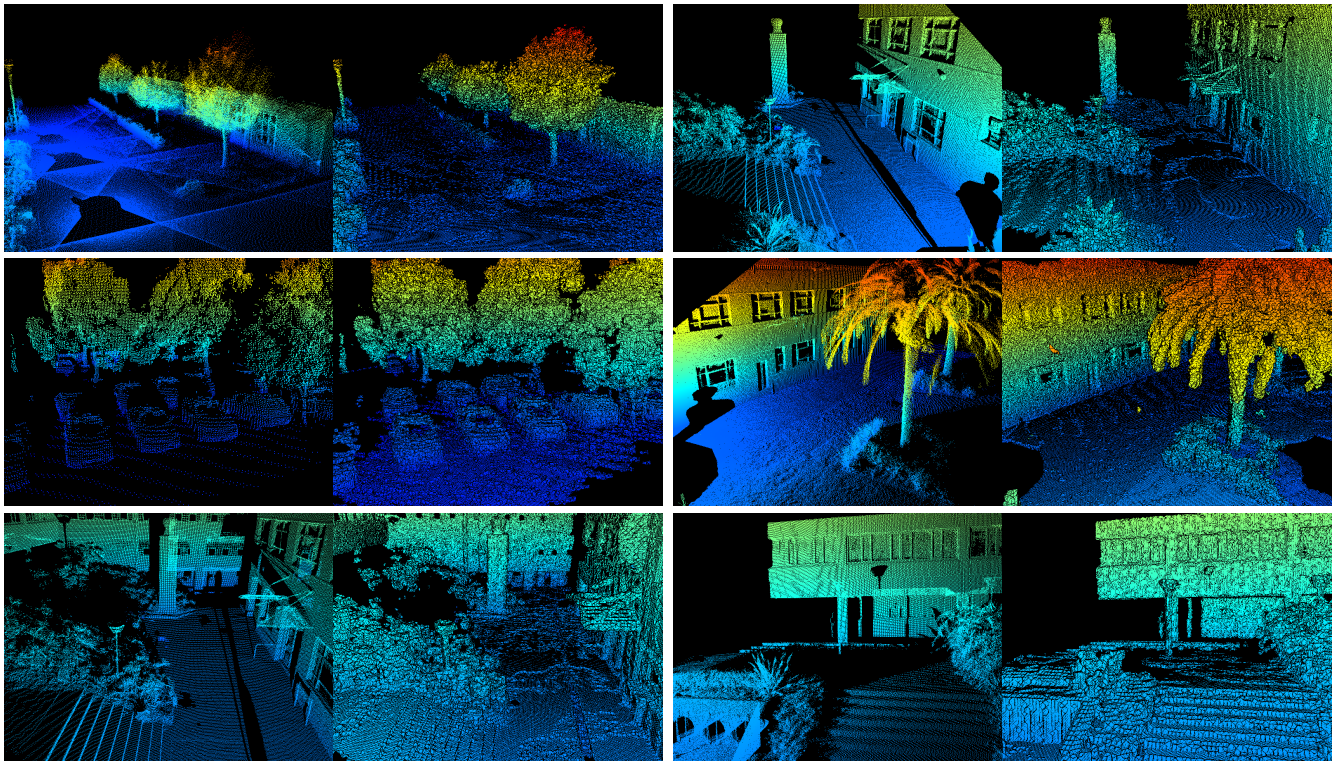


Fig. 8: Visual representation of the 3D scene reconstruction results from Table II, using CVAE-HM. Note how the reconstructive model is able to reason over sparse point clouds to produce a more solid representation of structures, and also extrapolates available information to areas not covered by sensors. It is also capable to complete objects based on partial views, like cars (middle left image) and trees (top left and middle right images). Interestingly, it learns to consistently fill in gaps produced by shadows, completing partially occluded structures (i.e. bottom left and top right images).

representation. In Fig. 7 we show results for CVAE-HM under different average cluster sizes, both for observed information and data gaps. As expected, smaller cluster sizes produce better classification results, since the structures to be learned are simpler, however they are also able to reconstruct only smaller gaps. As the average cluster size increases, larger gaps are able to 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 the ones usually found in the literature for similar tasks, such as object detection and scene reconstruction [34, 20, 30].

Lastly, in Fig. 8 we can see 3D reconstruction results in areas that were not deliberately removed from the evaluation dataset, such as shadows and partial occlusions. Due to the high 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 the original unobserved shape. The use of a higher resolution to generate the grid representations would most likely result in more detailed reconstructions, however due to the high computational cost and memory requirements this assumption was not explored here and is left for future work.

V. Conclusion

This paper introduced a novel methodology for 3D occupancy mapping, that utilizes Convolutional Variational Auto-Encoders 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 higher 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 in [29].

Acknowledgements

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

References

[1] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007.

[2] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. In *Proceedings of the VLDB Endowment*, volume 5, pages 622–633, 2012.

[3] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the International Conference on Computational Statistics (COMPSTAT)*, pages 177–186, 2010.

[4] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[5] L. Deng and D. Yu. Deep learning: Methods and applications. Technical report, Now Publishers, 2014.

[6] C. Doersch. Tutorial on variational autoencoders. Technical report, arXiv preprint arXiv:1606.05908, 2016.

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

[8] A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Carnegie Mellon University, Pittsburgh PA, USA, 1989.

[9] G. Ghiasi and C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

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

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

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

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

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

[16] S. Hochreiter and J. Schmidhuber. Feature extraction through lococode. *Neural Computation*, 11(3):679–714, 1999.

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

[18] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(7):881–892, 2002.

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

[20] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3050–3057, 2014.

[21] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. *A Tutorial on Energy-Based Learning*. The MIT Press, 2006.

[22] J. Masci, U. Meier, D. Ciresan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature learning. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 52–59, 2011.

[23] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[24] A. Nuchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d slam: 3d mapping outdoor environments. *Journal of Field Robotics (JFR)*, 24(8-9):699–722, 2007.

[25] S. O'Callaghan, F. Ramos, and H. Durrant-Whyte. Contextual occupancy maps using gaussian processes. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1054–1060, 2009.

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

[27] M. Ranzato, F. Huang, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[28] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with regional proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[29] G. Riegler, A. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Computing Research Repository (CoRR)*, 2016.

[30] M. Ruhnke, B. Steder, G. Grisetti, and W. Burgard.

Unsupervised learning of compact 3d models based on the detection of recurrent structures. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2137–2142, 2010.

[31] G. Sansone. *Orthogonal Functions: Revised English Version*. Dover Books on Mathematics, 2012.

[32] D. Scherer, A. Muller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2010.

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

[34] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 634–651, 2014.

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

[36] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.