Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Unsupervised Feature Learning for 3D Scene Reconstruction with Occupancy Maps

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

Abstract

This paper addresses the task of unsupervised feature learning for three-dimensional occupancy mapping, as a way to segment higher-level structures based on raw unorganized point cloud data. In particular, we focus on detecting planar surfaces, which are common in most structured or semi-structured environments. This segmentation is then used to minimize the amount of parameters necessary to properly create a 3D occupancy model of the surveyed space, thus increasing computational speed and decreasing memory requirements. As the 3D modeling tool, an extension to Hilbert Maps (Ramos and Ott 2015) recently proposed in (Guizilini and Ramos 2016) was selected, since it naturally uses a feature-based representation of the environment to achieve real-time performance. Experiments conducted in simulated and real large-scale datasets show a substantial gain in performance, while decreasing the amount of stored information by orders of magnitude without sacrificing accuracy.

Introduction

Nowadays, the task of obtaining information from the surrounding environment is no longer an issue in mobile robotics. Stereo and RGBD cameras are able to provide scale-aware per-pixel dense point clouds that include color information, while 3D laser range sensors produce snapshots of surrounding structures at sub-degree resolution and sub-centimeter accuracy. Millions of points can be obtained in a fraction of a second creating challenges to store and process this vast amount of data in an efficient and useful manner. Furthermore, points-clouds are what is called "low-level information", meaning that they encode patterns that could be used to describe the same environment in a much more compact way. This is how the human brain works, by introducing high-level patterns that cluster large segments of the input data into semantically meaningful classes or features.

In (Lai, Bo, and Fox 2014) the authors explore unsupervised learning of features based on a virtual training dataset, extending to three-dimensional space the state-of-the-art 2D classifier Hierarchical Matching Pursuit, or HMP (Bo, Ren, and Fox 2011; Ren and Ramanan 2013). An alphabet of local scans is constructed in (Ruhnke et al. 2010) to describe the point cloud based on recurrent surfaces, thus benefiting from repetitive environments that share similar structures. Sparse coding is employed in (Ruhnke et al. 2013) to achieve massive decreases in memory storage requirements, while producing 3D surface models that outperform the widely used Octomap (Hornung et al. 2013). Alternatively, the work in (Deuge et al. 2013) focuses on outdoor datasets, developing a novel technique for the regular sampling of densely irregular Velodyne scans that allows unsupervised learning of relevant features, as opposed to standard hand-crafted features (Johnson 1997). Recently, much work has been done on using clustering for unsupervised deep learning (Dosovitskiy et al. 2014), such as convolutional clustering in (Dundar, Jin, and Culurciello 2016) to minimize redundancy in learned features and spherical K-means in (Coates and Ng 2012) to learn a dictionary of features to represent the original structures.

Inspired by the above works, in this paper we propose a novel clustering technique that takes into account the orientation of each cluster, in addition to its spatial coordinates[1], and apply this methodology to 3D scene reconstruction problems. We use as input an unorganized point cloud composed of occupied points (i.e. obtained by laser scans), indicating structures captured by the sensor. Unoccupied points are generated by randomly sampling the beams that produced occupied points (a ratio of 1 point / 2 meters was used throughout the paper). Cluster orientation is calculated based on statistical information obtained from this point cloud, and used to produce planar surface features, composed of clusters that share a similar normal vector.

These planar surface features are capable of describing the environment in a much more compact way without sacrificing accuracy. In particular, we show how this compact representation can be used by the Hilbert Maps framework to greatly decrease the dimensionality of the feature vector used to project observations into the reproducing kernel Hilbert space. The original paper (Ramos and Ott 2015) discussed 2D applications of this framework, while (Guizilini and Ramos 2016) introduced the concept of localized length-scales, in which the position of each feature is learned based on available data. Here, we propose learning the shape of each feature alongside its position. The contri-

[1] A C++ demo of the proposed algorithm is available in https://bitbucket.org/vguizilini/cvpp

butions of this paper are as follows:

- A novel method for K-means initialization that outperforms K-means++ (Arthur and Vassilvitskii 2007) speedwise, while maintaining similar final potential values and automatically selecting the optimal number of clusters.
- An iterative clustering technique that merges clusters with similar properties, creating super-clusters that combine the statistical information of its constituents.
- A Planar Surface covariance function, that can be used by the Hilbert Maps framework to model features in conjunction with the Squared Exponential covariance function (Guizilini and Ramos 2016).

Feature Learning

We proposed feature learning in two steps: spatial and orientation. Initially the available data is clustered spatially, based on Euclidean distance between points. This cluster set is then analyzed statistically to produce the mean and variance for each cluster. Orientation is then obtained based on this statistical analysis, and the initial cluster set is refined using this new information, to produce the features that will be used for scene reconstruction.

Spatial Clustering

The k-means algorithm (Lloyd 1982), despite its simplicity (or thanks to it), has stood the test of time and remains as the most widely used technique for unsupervised clustering. It takes as input the dataset $\mathcal{X}$ to be clustered and the initial cluster positions $\mathcal{C}$ and returns as output the optimized cluster positions $\mathcal{C}'$ that minimize the potential function:

$$\phi = \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{c} \in \mathcal{C}} ||\mathbf{x} - \mathbf{c}||^2. \quad (1)$$

Over the years, several extensions have been proposed to address some of the limitations of the original approach, namely: 1) scalability to large datasets; 2) initial cluster positioning $\mathcal{C}$; and 3) number k of clusters. The *mini-batch* extension, recently proposed by (Sculley 2010), addresses the scalability issue by subdividing the dataset into smaller batches, and performing optimization using stochastic gradient descent. Initialization heuristics (Arthur and Vassilvitskii 2007; Bahmani et al. 2012) are able to provide better starting points for cluster positioning that avoid local minima, and hypothesis-testing techniques (Pelleg and Moore 2000; Hamerly and Elkan 2004) automatically elect the optimal value of k during optimization.

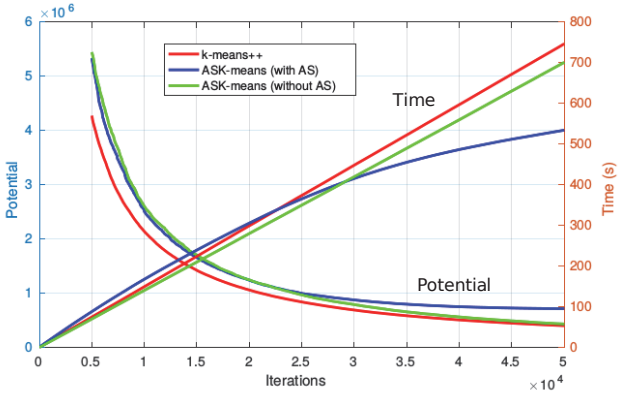


Figure 1: Comparison between different K-means initialization techniques ($n = 2156525$ and $d = 3$, obtained from laser scans in an outdoor environment). Average values of 10 runs with different random seeds.

We propose a novel initialization heuristic, ASK-means (*Automatic Stop K-means*), that also addresses the issue of selecting the optimal number of clusters (pseudo-code can be found in Alg. 1). It builds upon K-means++ (Arthur and Vassilvitskii 2007), in the sense that it uses the shortest distance to nearest cluster $d(\mathbf{x})^2$ as the sampling probability (line 5), however it avoids calculating the cumulative sum of distances $\sum_{\mathbf{x} \in \mathcal{X}} d(\mathbf{x})^2$ by producing instead two random values: one for the index of point to be tested and another for the probability of selection (lines 11-12). To decrease the influence of randomness, this process is repeated a given number of times and the point with highest distance is selected as the new cluster center (lines 9-19). Every time a new cluster center is selected, all points closer to it than a given threshold are removed from the test set (lines 21-28), thus decreasing the amount of points available for the next iteration. The process is then repeated to produce a new cluster center until there are no more available points.

A quantitative comparison between ASK-means and K-means++ can be found in Fig. 1, both in terms of processing time and potential value per iteration. As expected, processing time is linear for K-means++ as it is for ASK-means without automatic stop (lines 23-27 are removed from Alg. 1). When automatic stop is introduced (with a threshold of 0.1% the maximum distance between points), the algorithm becomes slightly slower during the first iterations due to the extra computational cost. However this extra cost is eventually compensated by the decrease in the number of available points, resulting in a speed increase of around 40%. K-means++ still produces better potential values, followed closely by ASK-means without automatic stop. When automatic stop is introduced, the decrease in available points produces more evenly spaced clusters, but at the cost of higher potential values. After initialization, the *mini-batch* K-means algorithm (Sculley 2010) was able to achieve similar potential values (within 10%) for both approaches, with the same number of iterations.

Orientation Clustering

Once spatial clustering is complete, the next step is to re-cluster this set based on orientation information, obtained statistically. Each cluster $C_i \in \mathcal{C}$ is defined by a collection of points $C_i = \{\mathbf{x} = \{x^0, x^1, \ldots, x^d\}\}_{j=1}^{N_i}$, with mean vector $\boldsymbol{\mu}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{x}_j$ and covariance matrix Σ_i with elements $\sigma_i^{rc} = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} \sum_{k=1}^{N_i} (x_j^r - \mu_i^r)(x_k^c - \mu_k^c)$, where r and c are row and column indexes of Σ_i.

Since we are looking for planar surfaces, it is natural to

Algorithm 1 ASK-Means initialization algorithm

Require: dataset $\mathcal{X}$ with n points
maximum number of clusters $\bar{m}$
number of random samples per iteration s
distance function $\phi(.,.)$, distance threshold t
Ensure: cluster centers $\mathcal{C}$

$v \leftarrow n$ % Initialise number of available points
$m \leftarrow 1$ % Initialise number of clusters
$\mathbf{q} \leftarrow \{0, 1, \ldots, n\}$ % Index vector for candidate points
$C \leftarrow$ random point picked from $\mathcal{X}$ % Initialise cluster vector
$\mathbf{d} \leftarrow \min_c \phi(\mathcal{X}, \mathcal{C})$ % Distance vector to nearest cluster
while $m < \bar{m}$ **and** $v > 0$ **do**
  $r \leftarrow max(\mathbf{d})$ % Maximum distance to nearest cluster
  $\mathbf{p} \leftarrow \emptyset$ % Candidate indexes for new cluster
  for i = 1 to s **do**
    while *true* **do**
      $a \leftarrow$ *random_integer*$(0, t)$
      $b \leftarrow$ *random_real*$(0, r)$
      if $d_{q_a} > b$ **then**
        $\mathbf{p} \leftarrow q_a$, **break**
      end if
    end while
  end for
  $j \leftarrow \arg\max(d_i$, i $\in \mathbf{p})$ % Highest distance index
  $\mathcal{C} \leftarrow \mathcal{X}_j$ % Add highest distance point as next cluster
  $m \leftarrow m + 1$ % Increment number of clusters
  for i = 1 to v **do** % Update distance vector
    $d_{q_i} \leftarrow min(d_{q_i}, \phi(\mathbf{x}_{q_i}, \mathbf{c}_m))$
    if $d_{q_i} < t$ **then** % Remove if too close to clusters
      $v \leftarrow v - 1$
      $q_i = q_v$
      $i \leftarrow i - 1$
    end if
  end for
end while

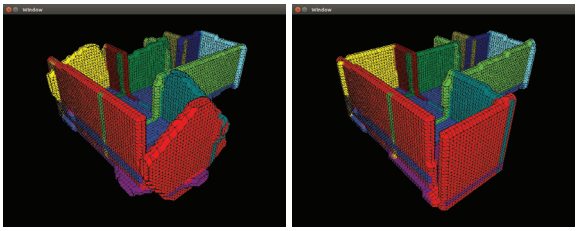
assume that one dimension (thickness) will be significantly smaller than the others, acting as the normal vector. This normal vector is defined as the eigenvector $\mathbf{u}$ associated with the smallest eigenvalue λ calculated from each Σ_i. An example of initial clustering and orientation calculation for a simple 2D dataset can be found in Fig 3b. The modeling results obtained from this same dataset, using LARD-HM (Guizilini and Ramos 2016), is shown in Fig. 3c.

Orientation clustering is then performed using a distance-based search method, with a threshold on the maximum spatial separation d_d on the localized length-scale space and on the maximum angular deviation d_a between normal vectors. These two calculations are as follows:

$$d_d(\boldsymbol{\mu}_j, \Sigma_j, \boldsymbol{\mu}_k, \Sigma_k) = \sqrt{(\boldsymbol{\mu}_j - \boldsymbol{\mu}_k)\Sigma^{-1}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_k)^T} \quad (2)$$

$$d_a(\mathbf{u}_j, \mathbf{u}_k) = cos^{-1}(\mathbf{u}_j \cdot \mathbf{u}_k), \quad (3)$$

where $\Sigma = (\sqrt{\Sigma_j} + \sqrt{\Sigma_k})^2$ is the weighted length-scale between two clusters. This formula was selected due to its intuitive way of providing a separation threshold, that is independent of the covariance values themselves. A value of 2, for example, indicates that the boundaries of 95% certainty (two standard deviations) for each cluster are in close proximity. Since it is faster to compute, the angular deviation threshold is used as an initial filter, and the square root of covariance matrices can be precomputed for efficiency.



(a) Without orientation alignment (b) With orientation alignment

Figure 2: Effects of 3D orientation alignment.

When two clusters are deemed close enough, their points are merged together to produce a new cluster, with mean and covariance values calculated based on this new collection of points. The same process is repeated iteratively, until there are no more changes in the number of clusters. The results of orientation clustering on a simple 2D dataset can be seen in Fig. 3d, where the 100 clusters from Fig. 3b were reduced to 11, each one correctly representing a wall in the environment. Note that the LARD-HM framework has issues dealing with such sparse datasets since it uses Euclidean distance to determine nearest neighbors for training and querying.

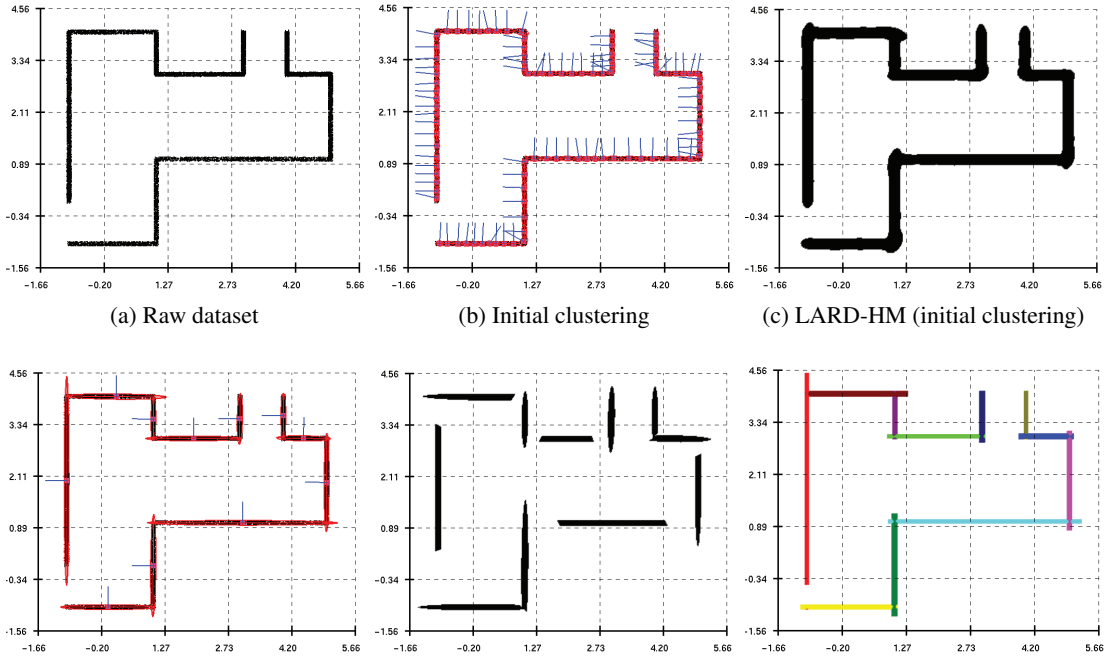
Alignment in 3D Space

So far we were only concerned with the orientation of the normal vector (i.e. smallest eigenvector), since it determines the similarity between planar surfaces. However, for three-dimensional spaces the orientation of other eigenvectors are also relevant, determining the alignment of the surface in relation to the point cloud it describes. The effects of such misalignment can be found in Fig. 2a (the extrusion of the dataset shown in Fig. 3), where several walls are correctly placed but wrongly rotated on the normal axes.

Because of that, we propose an extra step that corrects theses misalignments, taking place after the orientation clustering. In this step, the normal axis of each cluster is maintained, while the other two rotate according to the Rodrigues' rotation formula:

$$R(\mathbf{u}, \theta) = \begin{bmatrix} c + u_0u_0(1-c) & -u_2s + u_1u_0(1-c) & u_1s + u_2u_0(1-c) \\ u_2s + u_0u_1(1-c) & c + u_1u_1(1-c) & -u_0s + u_2u_1(1-c) \\ -u_1s + u_0u_2(1-c) & u_0s + u_1u_2(1-c) & c + u_2u_2(1-c) \end{bmatrix}, \quad (4)$$

where $\mathbf{u}$ is the rotation axis, $c = cos(\theta)$ and $s = sin(\theta)$. This is a convex optimization problem on θ that minimizes the spatial separation between each point and its respective cluster center. Since each individual point does not have a covariance matrix, the weighted length-scale matrix in Eq. 2 becomes simply $\Sigma = \Sigma_i$. Once the optimal angle θ is determined, the covariance matrix of each cluster is rotated by this amount in relation to the normal vector to produce the final cluster set $\mathcal{C}$, that will be used for scene reconstruction.



(a) Raw dataset (b) Initial clustering (c) LARD-HM (initial clustering)

(d) Orientation clustering (e) LARD-HM (orientation clustering) (f) SHAPE-HM (orientation clustering, colored by cluster)

Figure 3: 2D example of orientation clustering. Pink dots represent cluster centers, red ellipses indicate covariance matrices within two standard deviations, and blue lines depict normal vectors.

Scene Reconstruction

In this section we show how to use the cluster set $\mathcal{C}$ from Section to produce a 3D model of the environment that estimates the probability of occupancy at any point in the input space. We exploit the presence of planar surface features to greatly decrease the number of parameters required to properly generate this 3D model, thus increasing computational speed and decreasing memory requirements.

Hilbert Maps Overview

Initially proposed in (Ramos and Ott 2015), Hilbert Maps is a technique that represents real-world complexity in a linear fashion by operating on a high-dimensional feature vector, that projects observations into a reproducing kernel Hilbert space (RKHS). Assuming a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathcal{R}^3$ is a point in the three-dimensional space and $y_i = \{-1, +1\}$ is a classification variable that indicates the occupancy property of $\mathbf{x}_i$, the probability of non-occupancy for a query point $\mathbf{x}_*$ is given by:

$$p(y_* = -1|\Phi(\mathbf{x}_*), \mathbf{w}) = \frac{1}{1 + \exp\left(\mathbf{w}^T \Phi(\mathbf{x}_*)\right)}, \quad (5)$$

where $\phi(\mathbf{x}_*)$ is the feature vector and $\mathbf{w}$ are the weight parameters, that describe the discriminative model $p(y|\mathbf{x}, \mathbf{w})$. To estimate the optimal weight parameters $\bar{\mathbf{w}}$ we minimize the regularized negative log-likelihood (NLL) function:

$$NLL(\mathbf{w}) = \sum_{i=1}^{N} \log\left(1 + \exp\left(-y_i \mathbf{w}^T \Phi(\mathbf{x}_i)\right)\right) + R(\mathbf{w}). \quad (6)$$

where $R(\mathbf{w})$ is a regularization function such as elastic net.

A useful property of Eq. 6 is its suitability for stochastic gradient descent (SGD) optimization (Bottou 2010), in which the information contained in each point provides one small step towards a local minimum, given by:

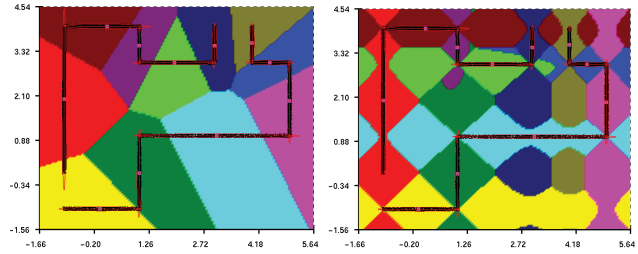
$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t A_t^{-1} \frac{\delta}{\delta \mathbf{w}} NNL(\mathbf{w}), \quad (7)$$

where $\eta > 0$ is the learning rate and the matrix A is a preconditioner to accelerate convergence rate (in most cases, A can be set to the identity matrix). 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 Eq. 7.

Nonstationary Length-Scales

In (Guizilini and Ramos 2016) the authors introduced the concept of nonstationary length-scales to the Hilbert Maps framework, in which scale varies throughout the input space. This variation is determined by the data distribution and is calculated using a statistical analysis similar to what is described in Section . Each one of the M clusters $\mathbf{c} \in \mathcal{C}$ acts as an extra dimension in the RKHS, and contributes to the feature vector in the following manner:

$$\phi(\mathbf{x}) = \begin{bmatrix} k(\mathbf{x}, \mathbf{c}_1, \Sigma_1) \\ k(\mathbf{x}, \mathbf{c}_2, \Sigma_1) \\ \vdots \\ k(\mathbf{x}, \mathbf{c}_M, \Sigma_M) \end{bmatrix}. \quad (8)$$



(a) Without length-scale weighting (b) With length-scale weighting

Figure 4: Effects of length-scale weighting in nearest neighbors search (compare to Fig. 3e-f).

To enforce sparsity, only the k closest clusters of $\mathbf{x}$ are selected to produce the feature vector $\phi(\mathbf{x})$, following the intuition that points far away will have less impact on each other's estimates. In (Guizilini and Ramos 2016) a standard kd-tree (Muja and Lowe 2009) is maintained to efficiently calculate nearest neighbors using Euclidean distance as the measure of proximity. While this approximation is enough for a highly dense cluster distribution, it suffers as sparsity increases during the orientation clustering process, leading to wrong nearest neighbor associations (see Fig. 4). Because of that, we employ an anisotropic search for nearest neighbors based on (Pereira and Andreazza 2010), in which the covariance matrix of each cluster is used to split the input space and produce a tree-like structure that is maintained for efficient queries.

Planar Covariance Function

In Eq. 8, $k(\mathbf{x}, \mathbf{c}, \Sigma)$ is commonly referred to as the covariance function, that defines the relationship between points (i.e. how they influence each other in the input space). The Squared Exponential covariance function (Eq. 9) is arguably the most common one, and is used in (Guizilini and Ramos 2016) to model the influence of each cluster in different portions of the input space during training and inference.

$$k_{sq}(\mathbf{x}_i, \mathbf{x}_j, \Sigma) = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)\Sigma^{-1}(\mathbf{x}_i - \mathbf{x}_j)^T\right). \tag{9}$$

However, if a particular cluster is known to belong to a planar surface (i.e. it was produced from a sufficiently large number of clusters with similar normal vectors), it is possible to achieve better modeling results with a covariance function that takes into account this information. Here we introduce the Planar Surface covariance function, defined as:

$$k_{ps}(\mathbf{x}_i, \mathbf{x}_j, \Sigma) = \begin{cases} 1 & \text{if } d_k < \lambda_k \mid k = \{1, 2, 3\} \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

where $d = U(\mathbf{x}_i - \mathbf{x}_j)$, with $U = \{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$ being the eigenvectors of Σ, and λ_k are the corresponding eigenvalues for U. This transformation is necessary in order to align the distance vector in relation to the eigenvectors, so they can be properly compared to the eigenvalues. Fig. 3f shows the result of using this covariance function on a simple 2D dataset.

Experiments

In this section we validate the proposed algorithm using three different datasets: *Room*, a simulated indoor environment; *Corridor*, a real indoor environment; and *Outdoor*, a real outdoor environment[2]. Clustering results from these datasets can be found in Table 1, where: n is the number of occupied points; V is the approximate volume covered in m^3; z is the initial number of clusters; and w and o (where applicable) are the number of final clusters dedicated to planar surfaces and other objects, respectively. In all experiments, ASK-means (Sec.) was used with a threshold of 0.1% the maximum distance between points.

Table 1: Statistics from the three datasets used in this paper.

	n	V	z	w	o
Room	582463	7653	8120	42	$--$
Corridor	759328	31521	7637	83	$--$
Outdoor	228090	86546	10560	23	1750

As it can be observed, there was a considerable decrease in the number of points necessary to describe each environment, of around two orders of magnitude from n to z and another two from z to w. This information was then used as input for the Hilbert maps framework, using the planar surface covariance function (Eq. 10) to model clusters that fall within this category and the squared exponential covariance function (Eq. 9) to model all other clusters (this distinction was done only for the *Outdoor* dataset, that represents a less structured environment).

The scene reconstruction results for each dataset, using only the orientation clustering statistic information, are depicted in Fig. 5, first for the entire scene (two leftmost columns) and then for particular zoomed-in areas. As it can be seen, the orientation clustering technique presented here was able to correctly identify virtually all the planar surfaces present in the scene, and furthermore, it was able to correctly distinguish between different planar surfaces. The introduction of unoccupied points, obtained by randomly sampling the empty space between the sensor and each occupied point, served to shape these planar surfaces into more intricate structures (i.e. windows in the first row). When these unoccupied points were not available, planar surfaces with similar orientations tended to cluster together, filling up gaps due to limited sensor coverage or resolution (i.e. ground in the second and third rows).

In Fig. 6 classification results using the proposed algorithm, henceforth referred to as PS-HM (Planar Surface Hilbert Maps), are shown in comparison to the original algorithm, LARD-HM (Localized Automatic Relevance Determination Hilbert Maps), and Octo-Map (Hornung et al. 2013). Note that, for structured environments, PS-HM was able to achieve a better correct classification percentage for every training/testing ratio, even though only around 1% of

[2]Both real datasets were obtained from http://kos.informatik.uni-osnabrueck.de/3Dscans/

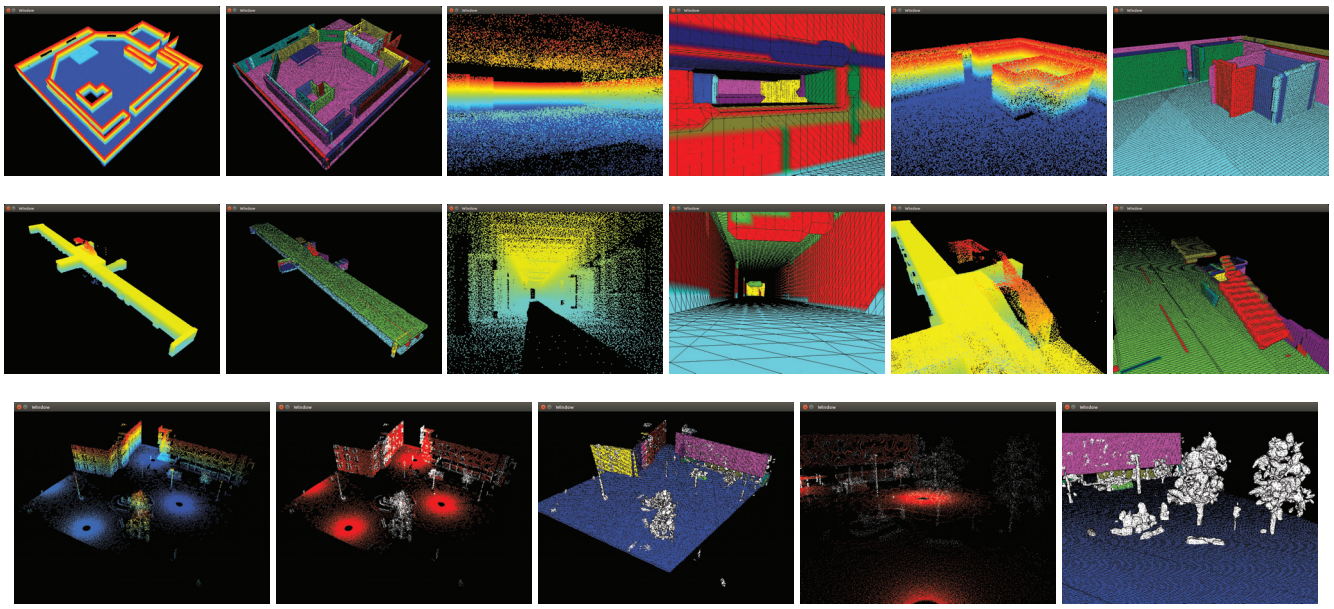


Figure 5: 3D scene reconstruction results (the rows depict different datasets in this order: *Room*, *Corridor* and *Outdoor*). Each planar surface cluster is colored with a different random color, while other clusters are in white. In the third row points belonging to planar surfaces are colored red, while the ones belonging to other objects are in white.

the original cluster points were effectively used. This translated into a higher processing speed during training and query, proportional to the decrease in number of clusters, and lower memory storage requirements. The ability of PS-HM to fill in gaps by merging statistical information from different clusters also contributes to its higher correct classification scores when dealing with sparse data. It is worth noting, as sparsity increases the threshold for ASK-Means used the select the number of clusters was also increased by the same percentage, to account for points further away.

Empirical tests indicate that orientation clustering consumes around 20% of the spatial clustering processing time. However, this step only has to be done once, when new data is acquired, and thus is overshadowed by the speed gains during training and query. The segmentation between planar surfaces and other structures also serves as a preprocessing step for other techniques, decreasing the number of candidate points. When there is a predominance of non-planar surfaces, PS-HM converges back to LARD-HM, since spatial clustering will have a smaller effecting of merging information and therefore most clusters will be modeled by the standard squared exponential covariance function. This can be seen in the results from the *outdoor* dataset, which contains non-structured objects that were not merged during orientation clustering.

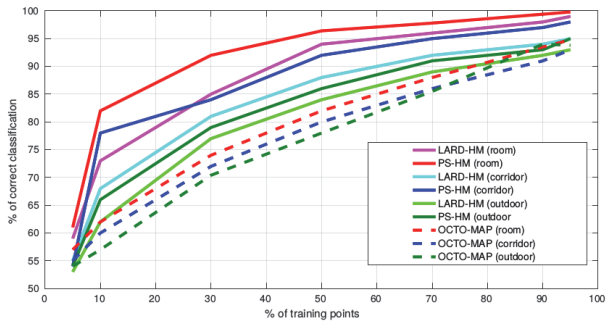


Figure 6: Classification results using the proposed algorithm (PS-HM), with orientation clustering, in comparison to the original algorithm (LARD-HM) and Octo-Map, for different ratios of training/testing points.

Conclusion

This paper introduced a novel clustering technique that takes into account the orientation of points in the input space, in order to detect the presence of planar surfaces. Its goal is to decrease the amount of information required for a proper modeling of the environment, using state-of-the-art 3D scene reconstruction algorithms. Results show that the proposed technique is able to decrease by two orders of magnitude the amount of clusters required for an accurate 3D scene reconstruction, thus increasing computational speed and decreasing memory requirements. Furthermore, it is also able to reliably segment planar surfaces from other objects in non-structured environments, which can then be more efficiently processed by other techniques to detect more complex shapes. Future work will focus on algorithm speed, and the detection and modeling of other feature types based on statistical information and unsupervised learning.

Acknowledgements

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

References

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

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

Bo, L.; Ren, X.; and Fox, D. 2011. Hierarchical matching pursuit for image classification: Architecture and fast algorithms. In *Advances in Neural Information Processing Systems (NIPS)*.

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

Coates, A., and Ng, A. 2012. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*, volume 7700. Springer Berlin Heidelberg, 2nd edition. 561–560.

Deuge, M. D.; Quadros, A.; Hung, C.; and Douillard, B. 2013. Unsupervised feature learning for classification of outdoor 3d scans. In *Proceedings of the Australasian Conference on Robotics and Automation (ACRA)*.

Dosovitskiy, A.; Springenbert, J.; Riedmiller, M.; and Brox, T. 2014. Discriminative unsupervised feature learning with convolutional neural networks. *Computing Research Repository (CoRR)*.

Dundar, A.; Jin, J.; and Culurciello, E. 2016. Convolutional clustering for unsupervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

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

Hamerly, G., and Elkan, C. 2004. Learning the k in k-means. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, 281–288.

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

Johnson, A. 1997. *Spin-Images: A Representation for 3-D Surface Matching*. Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh PA, USA.

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

Lloyd, S. 1982. Least-squares quantization in pcm. In *IEEE Transactions on Information Theory*, volume 28, 129–136.

Muja, M., and Lowe, D. 2009. Fast approximate nearest neighbours with automatic algorithm configuration. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 4, 331–340.

Pelleg, D., and Moore, A. 2000. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the International Conference on Machine Learning (ICML)*, 727–734.

Pereira, E., and Andreazza, C. 2010. Anisotropic k-nearest neighbor search using covariance quadtree. *Mecanica Computacional (Computational Geometry)* 24(60).

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

Ren, X., and Ramanan, D. 2013. Histograms of sparse codes for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ruhnke, M.; Steder, B.; Grisetti, G.; and Burgard, W. 2010. 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)*, 2137–2142.

Ruhnke, M.; Bo, L.; Fox, D.; and Burgard, W. 2013. Compact rgbd surface models based on sparse coding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1429–1435.

Sculley, D. 2010. Web-scale k-means clustering. In *Proceedings of the International Conference on World Wide Web (WWW)*, volume 19, 1177–1178.