Learning Structured Latent Space Encoding for Part Based Generative Modeling of 3D Objects

Cormac OMeadhra Stephen Landers

Abstract

Real-world operation of autonomous systems necessitates modeling and inference in 3D environments, that are complex, varied and highly cluttered. This work achieves increased generalizability through the use of a part based generative surface model that is capable of generating high fidelity surfaces from an un-ordered point cloud. The expectation-maximization algorithm is employed to learn the latent correspondences between the input data and the inferred parts at test-time, enabling the approach to better generalize to unseen parts by focusing on capturing surface patch regularity rather than part distributions.

1. Introduction

Real-world operation of autonomous systems necessitates modeling and inference in 3D environments. Point clouds are a widely used representation of 3D environments that are compatible with modern 3D sensors, such as RGB-D and LIDAR. However, the point clouds produced by these sensors are typically large ($> 10^5$ points) and, as they are samples from 2D surfaces, highly redundant due to missing information about local neighborhoods and shared surfaces. From a representational efficiency stand-point, it is desirable to identify a model of the more compact surface from which the point cloud was sampled.

To that end, this work aims to develop a part-based generative surface model that is capable of generating high fidelity surfaces from an un-ordered point cloud. Given a point cloud $\mathcal{X}$ representing an object model $\mathcal{O}$, consisting of parts $\{P_i\}$, such that $\cup_i P_i = \mathcal{O}$, we wish to construct a generative model $g_\phi(f_\theta(\mathcal{X}_i))$, acting on a subset of the input point cloud $\mathcal{X}_i \subseteq \mathcal{X}$, where $f_\theta(\mathcal{X}_i)$ is an encoder that generates a latent encoding $\mathbf{z}_i$ of the input and $g_\phi(\cdot)$ is a decoder that outputs a surface model S. The parts, $\{P_i\}$, are learned such that their union forms a surface S that well models the generating surface of the input point set. As such, the parts are not necessarily semantically meaningful, but a semantically meaningful decomposition can be enforced when labeled training data is provided. In the absence of semantic-part-segmented data, we train the the generative model in an unsupervised manner, with a heuristic based labeling, to minimize reconstruction error as detailed in section 3.

We are interested in exploring the effect of employing the expectation-maximization (EM) algorithm to determine point-set partitioning at test time, such that a set of J identical part models each operate on a set of, potentially partially, overlapping point clouds. Thus, at training time, we learn a relatively small autoencoder model that captures local structure for a J subsets of the input point cloud. We hypothesize that removing the need for the network to learn input-point partitioning will enable increased model capacity and therefore fidelity.

2. Related Work

While several representations of 3D objects are commonly used, point clouds are the most easily obtained from common commercially available sensors. Learning features directly on point clouds is a desirable tool for analysing point clouds. However, unlike images, point clouds do not exist on a structured lattice and any function that operates on point clouds must be invariant to permutation of the input points. A successful solution to this problem was found in the form of PointNet and its successors (Qi et al., 2017), which utilize point-wise features and max-pooling to achieve a permutation invariant feature extractor. Subsequently, the requirement for point-wise feature extraction has been relaxed to enable learning of 3D convolutional filters as a direct analogy to the 2D image case (Xu et al., 2018). These 3D convolutional filters enable efficient extraction of features from unordered point clouds, which can be leveraged to create an expressive latent space encoding.

The most direct method for generating models of 3D objects from latent codes is to output a point cloud. However, other representations, such as triangulated meshes, are better suited to many applications, e.g. graphics and computational geometry. Recently, approaches have been presented to predict mesh models from a latent space encoding, using, for example, planar patch deformation (Groueix et al., 2018) to map set of 2D planes onto the surface of a 3D object. However, these approaches have demonstrated limited fidelity. An alternative strategy that has yield much higher fidelity is to utilize the decoder as a classifier and implicitly encode the object surface as a classification boundary (Mescheder et al., 2018). A mesh model can then be obtained using standard reconstruction techniques.

Due to the structure inherent in part-based models, it is desirable to embed explicit knowledge of this structure into the point-cloud encoding architecture. This can be achieved through the use factorized graphical model in the form of a conditional random field, from which the latent space distributions can be extracted using mean-field inference (Huang et al., 2015). Improved performance has been observed through the use of VAE architecture, with additional structure enforced in the latent space. Specifically, the VAE latent encoding is combined with a learned Binomial existence distribution to model the presence of parts within an object instance, which enables capturing discontinuous part removal and addition actions (Nash & Williams, 2017). Generative Adversarial Networks (GANs) offer an alternative strategy to VAEs for generative modeling. Li et al. propose a multi-stage framework that utilizes a part-based bounding box GAN to achieve an object skeleton latent encoding followed by a second stage, which learns to fill the bounding boxes based on the part encodings. While the GAN provides an alternative optimization formulation, a general principle from this approach is the introduction of part-based skeletons into the latent space to yield increased accuracy by progressively growing the model complexity. As an alternative strategy to combining parts within the same latent space, Dubrovina et al. propose learning disjoint sub-spaces for each part, which can be linearly combined to reconstruct a composite object. Furthermore, Dubrovina et al. tackle the problem of part alignment through the use of a spatial transformer network (Jaderberg et al., 2015), which learns to transform parts from a canonical frame to the desired object pose.

The problem of part-based 3D modelling has strong parallels to 3D indoor scene synthesis. However, indoor scenes often exhibit greater diversity in relative part placement, grouping and orientation, which requires additional latent space structure (Li et al., 2019). Transferring these techniques to the problem of the 3D part-based modeling can enable increased robustness and diversity in the reconstructed models.

Variational autoencoders (VAE) are a widely used technique for generating latent space encoding of an input distribution (Kingma & Welling, 2013). The VAE learns a mapping from a data distribution to a Normal distribution with diagonal covariance over the latent variables. The mapping is learned by minimizing a lower bound on the reconstruction error and the deviation of the latent space distribution from the desired diagonal Gaussian. Achieving a diagonal covariance is referred to as disentangling the latent variables. This can be achieved more reliably through the use of constrained optimization, where the deviation between the diagonal latent prior and the learned latent distribution is bounded (Burgess et al., 2018). This concept can be extended further through the introduction hierarchies of independent variables, which can achieve a more expressive latent space factorization (Esmaeili et al., 2018).

Several surface model variations were considered for this work. Ultimately, we elected to use an explicit surface model achieved through a parametric surface function.

Implicit Surface Models Learning implicit models of object surfaces has recently been shown to achieve high-fidelity surface models (Mescheder et al., 2018; Park et al., 2019). One particularly interesting implicit representation is using a signed distance field (SDF) (Park et al., 2019). Given a network $f_{\text{SDF}}(x, \theta)$ that outputs a signed distance field from a surface S, we can extract the surface via the level-set $f_{\text{SDF}}(x, \theta) = 0$. Training such a network simply requires regressing to an SDF learned over the input shape. We follow the approach of Park et al. (2019) to construct an SDF for each mesh in the input dataset and use a cost function of the form

$$L_{\text{SDF}}(f_{\text{SDF}}(x, \theta), s) = |\text{clamp}(f_{\text{SDF}}, \tau) - \text{clamp}(s, \tau)| \quad (1)$$

where $\text{clamp}(x, \tau) = \min(\tau, \max(-\tau, x))$ and τ is a parameter controlling the maximum extent of the SDF.

Explicit Surface Models While implicit surface modeling has the benefit that it outputs the distance field directly which is useful for the E-step calculation (**??**), explicit surface modeling has the benefit of directly outputting a usable shape model such as a mesh. We follow the approach of Groueix et al. (2018) and minimize the symmetric Chamfer distance between the learned mesh, $f_{\text{MESH}}(x, \theta)$, and the ground truth S. The loss function is evaluated over a point cloud $\tilde{\mathcal{X}}$ resampled from $f_{\text{MESH}}(x, \theta)$ and a point cloud $\mathcal{S}$ resampled from S.

$$L_{\text{MESH}}(\tilde{\mathcal{X}}, \mathcal{S}) = \sum_{s \in \mathcal{S}} \min_{x \in \tilde{\mathcal{X}}} \|x - s\|_2^2 + \sum_{x \in \tilde{\mathcal{X}}} \min_{s \in \mathcal{S}} \|x - s\|_2^2 \quad (2)$$

Primitive Surface Models The final approach we consider is learning primitive based models. Specifically, we consider superellipsoids which implicitly define a surface by the equation

$$(|\, x \,|^r + |\, y \,|^r)^{t/r} + |\, z \,|^t = 1 \quad (3)$$

A significant benefit to this model is that it only requires the prediction of two parameters r, t. The loss function is identical to L_{MESH}, but with points resampled from the surface of the primitive in this case.

3. Methodology

3.1. Probabilistic Surface Model

We wish to model a surface in $\mathbb{R}^3$, which we represent in parameterized form

$$\mathcal{S}_j(\mathbf{\Theta}) = f_j(u, v; \mathbf{\Theta}) = \mathcal{S}_j : \mathbb{R}^2 \to \mathbb{R}^3 \tag{4}$$

where the parameters $\boldsymbol{\theta}$ define the chart from a two-dimensional space on which u, v are defined. We model a point cloud, $\mathcal{X}$, as a set of noise-corrupted samples from a uniform distribution over the surface $\mathcal{S}(\mathbf{\Theta})$

$$\mathbf{x} = \mathbf{s}_{\min} + \mathbf{n}\epsilon \tag{5}$$

where $\mathbf{s}_{\min}$ is the point closest to $\mathbf{x}$ lying on $\mathcal{S}$, $\mathbf{n}$ is the unit-normal vector at $\mathbf{s}_{\min}$ and ϵ is zero-mean Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The distribution over point cloud samples can then be expressed as (dropping the dependency on $\mathbf{\Theta}$ for brevity)

$$p(\mathbf{x}) = p(\mathbf{x}|\mathbf{s}_{\min};)p(\mathbf{s}_{\min}) = \mathcal{N}(d_{\min}(\mathbf{x}), \sigma^2)\frac{1}{|\mathcal{S}|} \tag{6}$$

where $d_{\min}(\mathbf{x}) = \|\mathbf{x} - \mathbf{s}_{\min}\|_2$ and $|\mathcal{S}|$ is the surface area of $\mathcal{S}$. We note that the modelling the noise as being directed normal to the surface is a choice made for mathematical convenience, in that it results in a tractable form for the point likelihood function.

We further elect to model $S(\mathbf{\Theta})$ as the union of J surface parts $S_i(\boldsymbol{\theta}_i)$, where $\mathbf{\Theta} = \{\boldsymbol{\theta_1, \ldots, \theta_J}\}$. Each parts S_i is associated with a corresponding probabilistic model $p_i(x)$, which is defined according to Equation (6). By constraining the surface parts to be disjoint, i.e. $x \in S_i \Rightarrow x \notin S_j \; \forall j \neq i$, we can express the union of surface parts as a sum

$$p(\mathbf{x}) = \sum_{j=1}^{J} \pi_j p_j(\mathbf{x}) = \sum_{j=1}^{J} \frac{1}{|\mathcal{S}|} \exp\left(-\frac{d^2_{\min,j}(\mathbf{x})}{\sigma^2}\right) \tag{7}$$

where $\{\pi_j\}$ are a set of normalizing coefficients, which are given by $\pi_j = \frac{|\mathcal{S}_j|}{|\mathcal{S}|}$. in order to achieve the desired uniform surface distribution.

3.2. Learning Surface Models via the EM algorithm

Given a point set $\mathcal{X}$, the parameters $\mathbf{\Theta}$ can be learned through maximum likelihood estimation, which is achieved by maximizing the log-likelihood of the probabilistic surface model, $\log p(\mathbf{x}; \mathbf{\Theta})$. However, due to the presence of the sum inside the log, this objective is difficult to optimize.

The expectation-maximization (EM) algorithm (Dempster et al., 1977) provides an iterative method for optimizing the log-likelihood. We proceed by introducing binary, latent correspondence variables, z, and using the *complete* data log-likelihood to optimize a lower bound

$$\ell(\mathbf{x}) = \sum_{i=1}^{N} \log \sum_{\mathbf{z}} p(\mathbf{x}_i, z) \geq \sum_{i=1}^{N}\sum_{j=1}^{J} \gamma_{ij} \log \frac{p(\mathbf{x}_i, z = j)}{\gamma_{ij}}$$

where $\gamma_{ij} = q(z = j|\mathbf{x}_i)$ and $q(z|\mathbf{x})$ is a proxy posterior distribution over the latent correspondence variables. It can be shown (Bishop, 2006) that the lower bound is tight when the γ_{ij} terms are selected as

$$\gamma_{ij} = p(z_j|\mathbf{x}_i, \mathbf{\Theta}) = \frac{\hat{p}_k(\mathbf{x}; \boldsymbol{\theta}_j)}{\sum_k \hat{p}_k(\mathbf{x}; \boldsymbol{\theta}_k)} \tag{8}$$

where $\hat{p}_k(\mathbf{x}; \boldsymbol{\theta}_k) = \exp\left(-\frac{d^2_{\min,k}(\mathbf{x})}{\sigma^2}\right)$ However, in the case of distributions with non-compact support, eq. (7) this will assign N points to each of the J surface parts. Under an alternative interpretation of EM, the E-step can be viewed as minimizing the divergence between the true latent variable posterior $p(z|\mathbf{x})$ and the proxy $q(z|\mathbf{x})$(Neal & Hinton, 1998). By placing constraints on families of valid posteriors $q(z|\mathbf{x})$, we can retain the guarantees of monotonically increasing the lower bound, while achieving significant sparsification of the assignment distribution γ_{ij}.

Maximal Assignment In maximal assignment, we set

$$\gamma_{ij} = 1 \text{ if } j = \arg\max_k \hat{p}_k(\mathbf{x}; \boldsymbol{\theta}_k) \tag{9}$$

and zero otherwise. This results in each point being assigned to a single part, which maximizes numerical efficiency and avoids the need for the distance function $d_{\min}(\cdot)$ to reason about point-wise weights. However, with this approach it is more difficult to impose inter-part smoothness constraints, which may result in a more jagged final surface.

δ-responsibility Assignment To overcome this risk of a non-smooth output surface, we propose to select the parts with the largest assignments such that a fraction δ of the responsibility is accounted for. We denote this set as Δ_i. The latent distribution is then

$$\gamma_{ij} = \frac{\hat{p}_k(\mathbf{x}; \boldsymbol{\theta}_j)}{\sum_{k \in \Delta_i} \hat{p}_k(\mathbf{x}; \boldsymbol{\theta}_k)} \text{ if } j \in \Delta_i \tag{10}$$

and zero otherwise. A benefit of this approach is that as the parts are no-longer disjoint, it becomes possible to enforce smoothness constraints during the training stage (Williams et al., 2018). However, this also necessitates the reasoning about point-wise weights, which will be discussed later.

M-Step Given the responsibilities γ_{ij}, the maximization step becomes

$$\Theta = \arg\max_{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J} \sum_{i=1}^{N}\sum_{j=1}^{J} \gamma_{ij} \log \hat{p}_k(\mathbf{x}; \boldsymbol{\theta}_j) \tag{11}$$

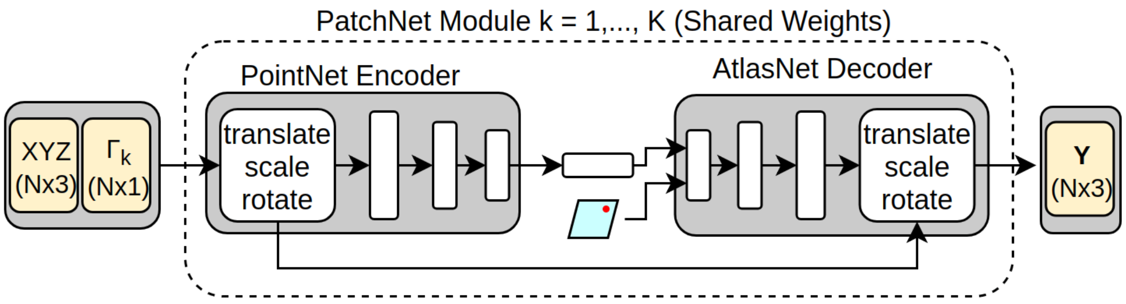

Figure 1. Proposed network architecture. From left to right, input point cloud and point-wise weights are transformed and passed through a PointNet (Qi et al., 2016) encoder to achieve a feature representation of the patch points. The surface is then reconstructed by passing samples from a 2D grid along with the patch feature through a decoder network.

Each of the $\boldsymbol{\theta}_j$ can be optimized independently, giving the following part-wise updates

$$\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta}_j} \sum_{i=1}^{N} \gamma_{ij} \left(\|\mathbf{x} - \mathbf{s}_{\min}\|_2^2 + \sigma^2 \log |\mathcal{S}_j(\boldsymbol{\theta}_j)| \right) \tag{12}$$

The first term of this objective seeks to find the minimum-error interpolating surface and the second term acts as a surface area regularizer. The noise variance, σ^2, controls the regularization, which is as expected.

3.3. Learning Surface Patches

In this work, we follow Groueix et al. (2018) and use an auto-encoder to model the function $f_j(u, v; \boldsymbol{\theta}_j)$. Specifically, we train an encoder modeled after the PointNet (Qi et al., 2016) with a latent space of size 1024. However, in order to assess the increased capacity of the proposed model, we train smaller PointNet encoders with reduced latent space codes. The encoder $g^{(e)}_{\boldsymbol{\theta}_j}(\mathcal{X}_j)$ learns a local surface code describing an individual surface part defined by $\mathcal{X}_j$.

Each of the J latent codes are then passed through identical decoder networks, with tied weights, which output the learned surface representation $h^{(d)}_{\boldsymbol{\theta}_j}$. The decoder consists of a stack of fully-connected layers, mapping from the latent code of size L to the output size, which corresponds to a point in $\mathbb{R}^3$. An illustration of the proposed network is shown in Figure 1.

We train this network using the squared Chamfer distance loss function, which is given by

$$d_{CH}(\mathbf{x}, \mathcal{S}) = \min_{\mathbf{s} \in \mathcal{S}} \|\mathbf{x} - \mathbf{s}\|_2^2 \tag{13}$$

The Chamfer distance computes an approximation of the normal distance to the surface as shown in Figure 2.

In order to prevent the network overfitting to the training data, we regularize the surface area of the reconstructed

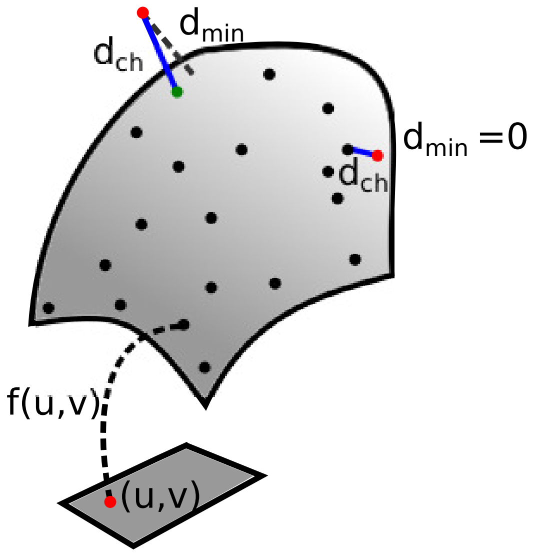

Figure 2. Visualization of the parametric surface representation and the relationship between the Chamfer distance and the normal surface distance.

patch. Given the parametric surface representation $f_j(u, v)$, the surface area of a patch is given by

$$\begin{aligned} |\mathcal{S}_j| &= \int_{(0,1)^2} \sqrt{\left\|\frac{\partial f_j}{\partial u}\right\|_2^2 \left\|\frac{\partial f_j}{\partial v}\right\|_2^2 - \left(\frac{\partial f_j}{\partial u}^T \frac{\partial f_j}{\partial v}\right)^2} \, \mathrm{d}u\mathrm{d}v \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \sqrt{\|f_{u,j}\|_2^2 \, \|f_{v,j}\|_2^2 - \left(f_{u,j}^T f_{v,j}\right)^2} \end{aligned} \tag{14}$$

where we made use of the Monte Carlo estimate of the surface integral evaluated using the N samples drawn from the 2D latent space when reconstructing the surface $\mathcal{S}_j$.

Given this choice of regularized training loss, we see that

Algorithm 1 Patch Initialization, **K-init**

```
Input: Point cloud $\mathcal{X}$, Target # patches $J$,
       Initial # patches $N_{\text{INIT}}$
Result: Point-part labels $\Gamma_i$
Init:   $\Gamma_{\text{init}} = \text{k-means}(N_{\text{INIT}}, \mathcal{X})$
While  $N_{\text{patches}} > J$
       Get patch $k$ with min $C_k = N_k + \omega_\lambda \lambda_{\min}$
       Merge patch $k$ with neighbor with minimum $C_j$
```

Algorithm 2 Patch EM

```
Input: Point Cloud $\mathcal{X}$, # patches$J$
Result: $K$-patch surface reconstruction $X_{\text{recon}}$, $\{S_k\}$
Init:   $\Gamma_{\text{init}} = \text{k-init}(K, X)$
While  not converged:
  M-step:
    $X_r = \bigcup_k \text{PatchNet}(X, \Gamma(k))$        Equation (12)
  E-step:
    $\Gamma(k) = p(z_k|\mathcal{X}, \mathbf{\Theta})$        Equation (8)
```

the network is trained to optimize the same function as in Equation (12) from the M-step of the EM algorithm, where the regularization parameter can be interpreted as the variance of the Gaussian sampling noise.

3.4. Initialization Strategies

In order to initialize the EM algorithm and to provide the network with suitable training data, we utilize a heuristic based partitioning algorithm. The algorithm The algorithm pseudocode is presented in Algorithm 1. The algorithm proceeds by learning a k-means partitioning of the data with some number of patches N_{INIT}, which is greater than the target number of patches J. The k-means patches are progressively merged until the target number of patches has been reached. The merging strategy selects patches according to a weighting function defined by the weighted sum the patch size N_k and $\lambda_{\min}$, the minimum eigenvalue of the patch sample covariance. The term ω_λ is a mixing term. An example partitioning for a chair model taken from the ShapeNet dataset is shown in fig. 3.

3.5. Algorithm

The full EM algorithm used to fit a patch model requires relatively few steps. Given and point cloud and a target number of patches, initialization is performed using the k-init algorithm. The initialization is the passed through the learned surface parameterization network, which outputs a new surface parameterization (M-step). The responsibility weights are then recomputed (E-step) and the process is repeated until the responsibility converges. The pseudocode for the algorithm is given in Algorithm 2.

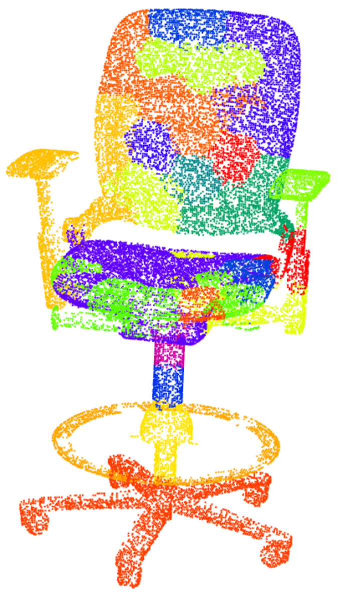
Figure 3. Heuristic patch initialization using the K-init algorithm with $K = 25$.

4. Experiments

4.1. Dataset

Evaluation was performed on the ShapeNet dataset (Chang et al., 2015), which provides a $40k$ instances over 13 differed objects classes. The ShapeNet dataset was used in the modified form provided by (Groueix et al., 2018), which extracts the subset of each shape that is visible by raycasting from an external viewpoint. This removes points lying on the interior of the object which could give the point cloud volumetric properties. The ShapeNet training data used considers the classes plane, bench, cabinet, car, chair, monitor, lamp, speak, firearm, couch, table cellphone and watercraft.

4.2. Results

The training loss, presented in the Figure 4, shows near identical performance across training and validation data for both networks. This suggests that at the scales of the patches used, the objects show significant regularity. This is in keeping with the central hypothesis of this work - that at a local level, surfaces exhibit substantial similarities, despite stark global differences. Figure 4 also demonstrates the expected result that increasing the number of trainable parameters in the network results in improved performance. However, we note that the performance for the smaller network remains competitive even at a substantially reduced size.

A qualitative evaluation for the small network is presented in Figure 5 and Figure 6. These figures demonstrate that

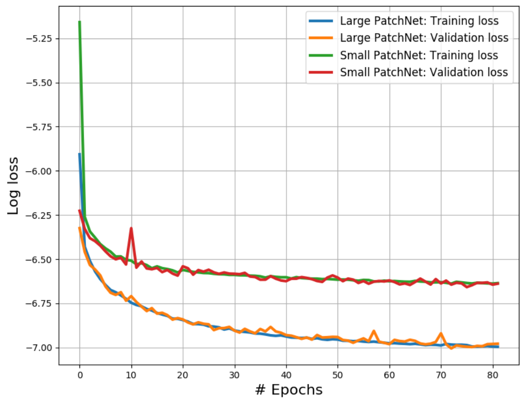

Figure 4. Training loss for the small and large variations of the auto-encoder architecture. Note the similarity between the test and validation error.

the training loss achieved in Figure 4 corresponds to strong reconstruction performance, which the majority of the data structure being preserved. However, we note that the reconstructed surfaces exhibit some deviation from the input point clouds, which can be attributed to two short-comings of this version of the proposed algorithm. Firstly, in the hard-partitioning approach, patches are independent, which does not impose any constraints on smoothness across patch transitions; using the δ-responsibility E-step may alleviate this. Secondly, the noise variance σ^2 was set to zero for this trials, which results in the networks being unregularized. This may result in overfitting to some of the training data. However, since the training and validation losses in Figure 4 are very similar, it does not seem likely that this is a significant issue.

5. Conclusion

Surface reconstruction can be achieved in an efficient and generalizable manner using a collection of small, learned patch primitives. By encoding these patches through a neural network, a flexible representation is achieved that is capable of generalizing across classes ranging from planes to cars. The flexibility of the approach can be further increased through the use of the EM algorithm to learn the latent correspondences between the patches and the input data. The validity of the approach was demonstrated quantitatively and qualitatively on a modified version of the ShapeNet dataset. Further work is required to evaluate the performance of the complete algorithm and to determine the optimal architecture and hyperparameter selection.

References

Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Dubrovina, A., Xia, F., Achlioptas, P., Shalah, M., and Guibas, L. Composite shape modeling via latent space factorization. *arXiv preprint arXiv:1901.02968*, 2019.

Esmaeili, B., Wu, H., Jain, S., Bozkurt, A., Siddharth, N., Paige, B., Brooks, D. H., Dy, J., and van de Meent, J.-W. Structured disentangled representations. *stat*, 1050:12, 2018.

Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 216–224, 2018.

Huang, H., Kalogerakis, E., and Marlin, B. Analysis and synthesis of 3d shape families via deep-learned generative models of surfaces. In *Computer Graphics Forum*, volume 34, pp. 25–38. Wiley Online Library, 2015.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. Spatial transformer networks. In *Advances in neural information processing systems*, pp. 2017–2025, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., and Guibas, L. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):52, 2017.

Li, M., Patil, A. G., Xu, K., Chaudhuri, S., Khan, O., Shamir, A., Tu, C., Chen, B., Cohen-Or, D., and Zhang, H. Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)*, 38(2):12, 2019.

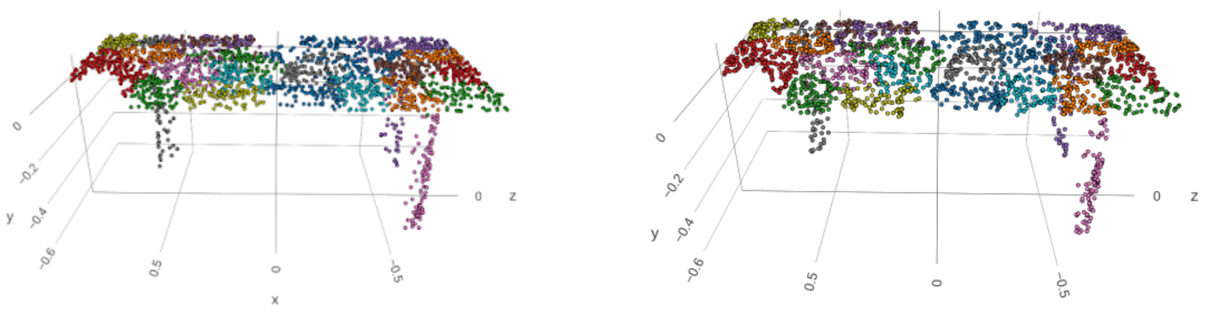

Figure 5. Table class training data input (left) and reconstruction (right). Colored points correspond to different patches.

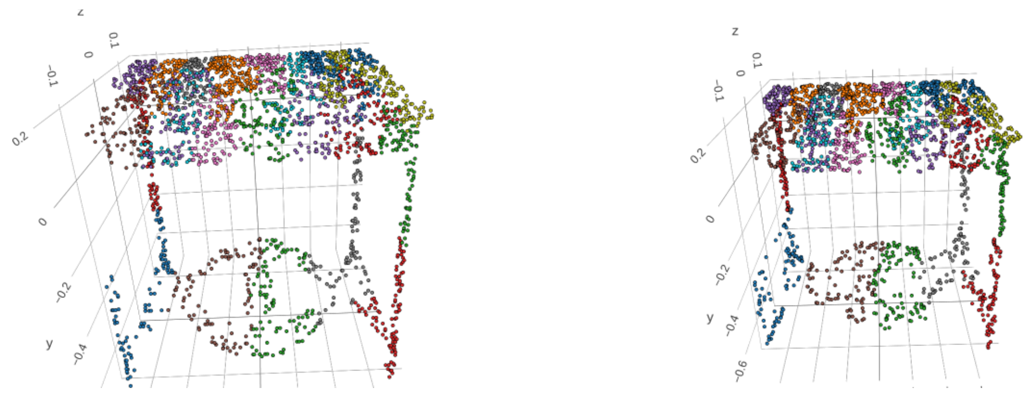

Figure 6. Table class validation data input (left) and reconstruction (right). Colored points correspond to different patches.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. *arXiv preprint arXiv:1812.03828*, 2018.

Nash, C. and Williams, C. K. The shape variational autoencoder: A deep generative model of part-segmented 3d objects. In *Computer Graphics Forum*, volume 36, pp. 1–12. Wiley Online Library, 2017.

Neal, R. M. and Hinton, G. E. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pp. 355–368. Springer, 1998.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. *arXiv preprint arXiv:1901.05103*, 2019.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. 2016.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pp. 5099–5108, 2017.

Williams, F., Schneider, T., Silva, C., Zorin, D., Bruna, J., and Panozzo, D. Deep geometric prior for surface reconstruction. *arXiv preprint arXiv:1811.10943*, 2018.

Xu, Y., Fan, T., Xu, M., Zeng, L., and Qiao, Y. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 87–102, 2018.

A. Alternative Grouping Strategies

Initializing our EM point cloud groupings randomly or using k-means causes the point cloud to be split along arbitrary lines. It would be ideal if this initial grouping made an attempt to adhere to the natural groupings humans perceive. While human perceptual groupings are a complex topic and a complete treatment is out of the scope of this project, one simple way we group points is by looking for smooth surfaces. The goal of this section is to produce an initialization method that will group some parts of a point cloud by estimating how well they are represented by smooth surfaces, resulting in a more natural looking split.

Perceptual grouping This method operates by building up groupings from a low level. It can build up arbitrarily large groupings given that it can do the following:

- Smooth surfaces can be detected in small patches of the point cloud.
- Smooth surfaces can be detected in pairs of small patches of the point cloud.

The method for detecting smooth surfaces will be covered in the next section. Assuming that such a method exists, we can break the point cloud into small arbitrary pieces and look for smooth surfaces in those pieces. We then attempt to combine each of these pieces with its neighbors, making pairwise smooth surfaces. Once we have done this, we can build larger pieces by looking for pairwise smooth surfaces that share the small smooth surfaces that we detected. The collection of all pairwise smooth surfaces that are connected by small smooth surface subsets is a perceptual grouping, and could grow to represent very complex surfaces given that the smaller pieces of those complex surfaces fit our smooth surface model.

To formalize this concept, given

$$\{s_i : s_i \in surfaces\} \tag{15}$$

and

$$\{p_k : p_k \cup \{s_i, s_j\}, i \neq j, p_k \in surfaces\}, \tag{16}$$

a perceptual grouping is defined as

$$\cup \{p_k : \exists\{p_m, s_i\} \mid k \neq m, s_i \subset p_k, s_i \subset p_m\}. \tag{17}$$

Smooth surface model There are many ways to represent a 3D surface. Here are the criteria we had for this model:

- This surface model should take inputs in 2 dimensions and give values in the third. This prevents the surface from representing geometry that is more complex than we intend. For example, we do not want to be able to represent a sphere, because we do not expect small patches of a larger object to resemble spheres.
- Fitting the surface to a point cloud should be efficient. For that reason, we restrict ourselves to surfaces that can be produced by sequences of linear solutions.

To meet these criteria, we fit a surface to a point cloud using the following method:

1. We fit a plane to the point cloud using the standard $Ax + By + Cz + D = 0$ equation.
2. We use the solved plane parameters to create a new reference frame aligned with the plane. To recover the other two axes (the plane normal is the third), we find the most direct rotation that brings the Z-axis to meet our normal, and rotate the X and Y axes by the same rotation. The origin of our reference frame is the mean of the point cloud, projected onto the plane.
3. We transform the point cloud into this reference frame. To standardize the size of the point cloud, we divide by the largest of the standard deviations of the three axes. We don't want to apply different scales for each axis, as this would change the curvature of the point cloud.
4. We solve the system $Ax^2 + By^2 + Cxy + Dx + Ey + F = z$. In order to limit ourselves to surfaces without heavy curvature, we throw out any point clouds where A, B, or C have an absolute value over 0.1.

All linear solves are performed either using standard linear least squares, or SVD when the target values are all zero.

Determining if a point cloud is a smooth surface We start by determining what small point cloud patches to attempt to fit smooth surfaces to. We do the following:

1. We split up the point cloud using k-means M times. This lets us look at M different splits of the point cloud.
2. We fail a patch immediately if it does not have enough points. This limit is set based on expected density. In our case, for a point cloud with 2000 points, this limit was set to 15.
3. We use RANSAC to fit our smooth surface to the patch. If 75% of the points in the patch fit the model, we pass it. We consider a point as fitting the model if the Z value predicted by the model is within a threshold of the Z value of the point in the surface's reference frame. The threshold is computed as a function of the 3D size of the patch, and in general needs to require that the surface is much less thick than it is long or wide.
4. We compute the eigenvalues of the covariance matrix of the patch, and reject any patches where more than one of the values is similar in size to the inlier threshold. This allows us to avoid fitting surfaces to lines.

Once we have a list of patches that were successfully represented by smooth surfaces, we do the following:

1. We make a neighbor list for each patch that includes the N closest other patches. In our implementation, we used 30 for N.
2. For every pair of patches that are neighbors, we combine their points and use RANSAC to fit a surface to them. The pair is considered a surface if 95% of the points are inliers.
3. We then find the unions of pairs with patch subsets in common to produce our perceptual groupings.

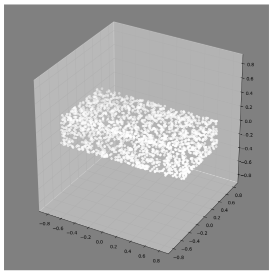
Figure 7. Input box point cloud.

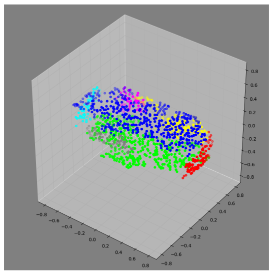
Figure 8. Box point cloud and perceptual split.

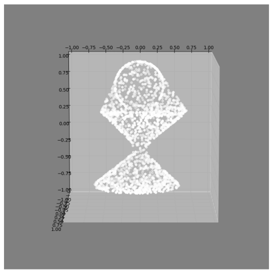
Figure 9. Input hourglass point cloud.

Sampling of results

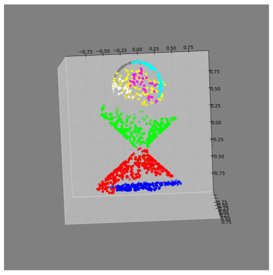
Figure 10. Hourglass point cloud and perceptual split.

Future work The perceptual grouping method rarely crosses perceptual lines, but it does happen. This is possible when a smooth surface can be found in two patches from two different surfaces that have a gap in between. This could be solved by adding a requirement for surfaces to have their points evenly distributed.

There are portions of each point cloud which are ignored by this method, as random sampling is involved in finding these surfaces. Also, we make the assumption that each patch only contains a single surface. A more principled method of splitting the point cloud into patches or a method for extracting multiple distinct surfaces from a patch could help to alleviate this.

The PartNet data set used to test this method has many instances of double walling, where two versions of the same surface exist side by side. This is difficult for our method to deal with, due to the assumption that each patch only has one surface. Removing this requirement should help in this case.

Long, thin objects, such as the stem of a desk lamp, are not well represented by these surfaces. Adding additional types of objects such as cylinders may help.