PointGMM: a Neural GMM Network for Point Clouds

Amir Hertz   Rana Hanocka   Raja Giryes   Daniel Cohen-Or

Tel Aviv University

Abstract

Point clouds are a popular representation for 3D shapes. However, they encode a particular sampling without accounting for shape priors or non-local information. We advocate for the use of a hierarchical Gaussian mixture model (hGMM), which is a compact, adaptive and lightweight representation that probabilistically defines the underlying 3D surface. We present PointGMM, a neural network that learns to generate hGMMs which are characteristic of the shape class, and also coincide with the input point cloud. PointGMM is trained over a collection of shapes to learn a class-specific prior. The hierarchical representation has two main advantages: (i) coarse-to-fine learning, which avoids converging to poor local-minima; and (ii) (an unsupervised) consistent partitioning of the input shape. We show that as a generative model, PointGMM learns a meaningful latent space which enables generating consistent interpolations between existing shapes, as well as synthesizing novel shapes. We also present a novel framework for rigid registration using PointGMM, that learns to disentangle orientation from structure of an input shape.

1. Introduction

Point clouds are a common and simple representation of 3D shapes. They can be directly obtained through scanning devices, which sample the surface of a 3D object. A major drawback of the point cloud representation is the inherent dependence on the particular sampling pattern, making it sensitive to occlusions, noise and sparse sampling. Moreover, they are *unordered* and *irregular*, and each individual point sample does not carry any non-local information [7, 28, 34, 43].

The Gaussian mixture model (GMM) is an alternative representation for 3D shapes [4, 13]. Unlike the point cloud, which uses an arbitrary specific discrete set of samples, GMMs use a fixed and prescribed number of Gaussians that probabilistically define the underlying 3D surface. GMMs are a compact and lightweight representation, which excel in representing a sparse and non-uniform point cloud. They are inherently *adaptive*: representing intricate details with
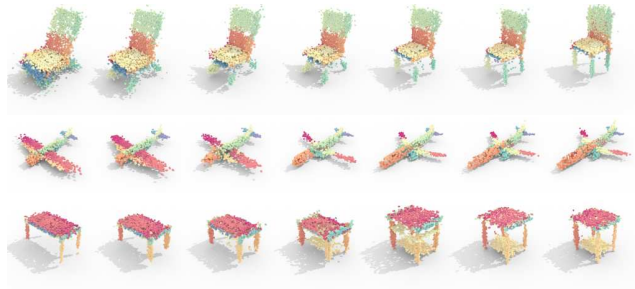

Figure 1: Shape interpolation using a PointGMM generative model. The consistent coloring in the hGMM leaf nodes suggests that an interpretable partitioning has been learned, without any supervision.

more Gaussians, while large smooth regions can be represented with a smaller number of Gaussians.

In this work, we present the use of GMMs as an intermediate and compact representation for point cloud processing with a neural network, called *PointGMM*. PointGMM is trained over a set of shapes to learn a *class-specific prior*. For a given point cloud, PointGMM learns a set of Gaussians which are characteristic of the shape class, and also coincide with the input point cloud. In other words, PointGMM provides additional geometric information which is otherwise missing from the input point cloud. Moreover, PointGMM is a *structured* representation, where subsets of Gaussians represent semantic spatial regions of the shape. This provides a means to consistently parse a diverse set of shapes, which is learned without any explicit ground-truth labels (Figure 1). We demonstrate the advantage of a learned GMM representation for the task of partial point cloud registration and shape generation.

We show that a major challenge in training a neural network to directly partition shapes into distinct regions for each GMM is a loosely defined objective, which is highly susceptible to local minima. To alleviate this issue, we learn in a *coarse-to-fine* manner, through a hierarchical GMM (hGMM) framework. We propose a network which learns to subdivide the input points into distinct groups that are modeled by GMMs at different shape scales. This promotes and encourages more effective learning; since the GMMs at

the bottom of the hierarchy concentrate on learning smaller spatial regions, while the top-level GMMs learn larger regions.

Although the input point cloud is unordered, PointGMM learns to output GMMs in a consistent and meaningful order.

This significant property motivates us to use PointGMM as a generative model trained on collection of shapes. We show that the learned latent space enables generating meaningful interpolations between existing shapes, as well as synthesizing novel shapes. Moreover, we present a novel framework for rigid registration using PointGMM, which learns to disentangle orientation from the input shape. To facilitate registration of partial shapes, we train the network to receive partial shapes, and generate GMMs as if they were complete. This enables PointGMM to perform non-trivial registrations which implicitly considers the overlap between missing regions. We compare to state-of-the-art rigid registration approaches on partial shapes with large rotation angles demonstrating the applicability of PointGMM.

Finally, another important property of the hGMM representation is that it enables a more efficient loss function compared to voxels $\mathcal{O}(n^3)$ or point clouds $\mathcal{O}(n^2)$ which are bound by the resolution of the grid or points, respectively. By contrast, the complexity of a GMM loss function depends on the number (k) of Gaussians $\mathcal{O}(n \cdot k)$ $(k \ll n)$, and the hGMM loss complexity is $\mathcal{O}(n \cdot \log k)$ [14].

2. Related work

Deep networks for irregular 3D shapes. Pointnet [34] first proposed a neural network which operates directly on irregular and unordered point clouds through 1×1 convolutions followed by global (max) pooling. Several follow-up works proposed incorporating local neighborhood information. For example, Pointnet++ [35] used a hierarchical network to capture local information. PointCNN [28] proposed learning a χ-transformation on point clouds, without using global pooling. Wang et al. [43] build Euclidean neighborhoods by building a graph from point samples. SPLAT-Net [39] represents points in a high-dimensional lattice. Another class of approaches propose irregular convolution and pooling operators on 3D meshes [21] or treats shapes as a graph [32]. For more details on learning on non-Euclidean data (*geometric deep learning*) refer to [7].

GMMs for 3D data. Ben-Shabat et al. [4] suggested using GMMs to represent 3D shapes, by pre-defining partitions of the point cloud and calculating the associated GMM. The parameters of the GMM are used as input features to a neural network for the task of classification, which was later extended in [5] for normal estimation. Unlike previous works, PointGMM uses the networks loss function to learn how to best partition the point cloud for the task at hand. Moreover, instead of using the parameters of the

GMM as features, the learned GMMs are used directly as the 3D shape representation.

Eckart et al. [13] also propose using a hierarchical GMM representation for the task of shape registration.

Yet, their hGMM is generated without learning. Thus, unlike our learned hGMM, it does not include any prior of a training set, nor any implicit information about missing parts. Moreover, [13] does not provide a latent space as PointGMM does, which enables shape generation.

Generative models for 3D shapes. In recent years, various works proposed leveraging the power of deep generative models for 3D shape generation. Achlioptas et al. [1] pioneered a deep generative model which directly generated sparse and irregular point clouds. SO-NET [27] used hierarchical feature structures for generative as well as discriminative tasks. A generative adversarial network for point clouds was proposed in [26]. Recently, StructureNet [31] use graph neural networks to jointly learn geometry and inter-part training on a collection of shapes. SDM-NET proposed generating deformable mesh parts using a VAE [18]. Surface Networks [25] propose a generative model for 3D surfaces via a Dirac operator. Although not a generative model, Tulsiani et al. [41] demonstrated that learning to fit primitives to shapes is an effective approach for exploiting visual similarities in the data. Our work proposes a novel approach for shape generation via a learned hGMM partition. The learned GMMs correspond to consistently segmented regions across a diverse set of shapes, without using any explicit correspondence in the loss function.

Shape registration. Shape registration is a well-studied problem with a variety of different techniques proposed over the years. There are two main classes of registration approaches: estimating global (rigid) or local (non-rigid) transformations between two potentially partial shapes. A popular technique for rigid registration is to apply RANSAC to find three matching points [17, 10]. In 4PCS [2] a technique for filtering the number of trials required, greatly improves efficiency. Often, rigid registration is followed by non-rigid registration for additional refinement. A popular approach for local registration is ICP [6, 11] and its many variants [36, 38]; which can also be used to compute a rigid transformation.

Recently, there have been efforts to apply deep neural networks for the task of rigid [40] and non-rigid [20, 19] registration, which can be faster and more robust than classic techniques.

PointNetLK [3] proposed a recurrent PointNet framework for rigid registration, which is more robust to initialization and missing parts compared to ICP.

Instead of learning to align one shape to another, we train PointGMM to orient shapes into a canonical pose, thereby indirectly calculating the transformation between two shapes.

3. Method

Our solution aims at representing a point cloud by a set of GMMs. To this end, we design an encoder-decoder based framework that generates a set of GMMs for given a point cloud. The encoder generates a latent vector from the input point cloud, and the decoder reconstructs a GMM from the latent vector. To train the network, we propose a maximum-likelihood based loss function that maximizes the probability that the point cloud was generated by the estimated Gaussians. To improve performance and avoid converging to local minima, we learn a hierarchical GMM (hGMM) instead of a single scale GMM. Our method follows Eckart *et al.* [14], who demonstrated that hGMM is an effective tool for representing 3D point clouds.

In the following, we describe the hGMM representation (Section 3.1) using the notation of [14], and the PointGMM architecture (Section 3.2). Finally, we describe how Point-GMM can be used for shape generation (Section 3.3) and registration (Section 3.4).

3.1. Hierarchical GMM

The hGMM can be viewed as a tree of Gaussians where the child of each node are a refined Gaussian mixture of their parent. The first level (root) is composed from a mixture of J (overlapping) weighted 3-dimensional multivariate Gaussians $\Theta_j^{l=1} = \{\pi_j, \mu_j, \Sigma_j\}$. Given a point cloud X of size N, its probability of being generated by $\Theta^{l=1}$ is:

$$p\left(X|\Theta^{l=1}\right) = \prod_{i=1}^{N} p\left(X_i|\Theta^{l=1}\right) = \prod_{i=1}^{N}\sum_{j=1}^{J} \pi_j \mathcal{N}\left(X_i|\Theta_j^{l=1}\right).$$
(1)

Based on this probability, we define a set of C latent variables c_{ij} that represent the binary associations between a point X_i and a Gaussian $\Theta_j^{l=1}$ in the mixture model. We calculate the posterior for all $c_{ij} \in C$ given $\Theta^{l=1}$ by

$$\gamma_{ij}^{l=1} \stackrel{\text{def}}{=} E\left[c_{ij}\right] = \frac{\pi_j p\left(X_i|\Theta_j^{l=1}\right)}{\sum_{j'=1}^{J} \pi_{j'} p\left(X_i|\Theta_{j'}^{l=1}\right)}.$$
(2)

The next levels in the hierarchical model are defined recursively. Given a point cloud X, At the root, we calculate the probabilities $p\left(X_i|\Theta_j^{l=1}\right)$ and the posteriors $\gamma_{ij}^{l=1}$ for each point and Gaussian in the mixture. Each $\Theta_j^{l=1}$ can then be refined as another Gaussian mixture of size $\hat{J}$:

$$p\left(X|\gamma^{l=1}, \Theta^{l=2}\right) = \prod_{i=1}^{N}\sum_{j=1}^{\hat{J}} \pi_j^{l=2||1} p\left(X_i|\Theta_j^{l=2||1}\right)$$
(3)

where the superscript $l = 2|1$ indicates the selection of Gaussian parameters at level 2 given the parent node at level 1. Each refinement term is a Gaussian mixture that satisfies $\sum_{j=1}^{\hat{J}} \pi_j^{l=2||1} = 1$. In the recursive step, we follow a
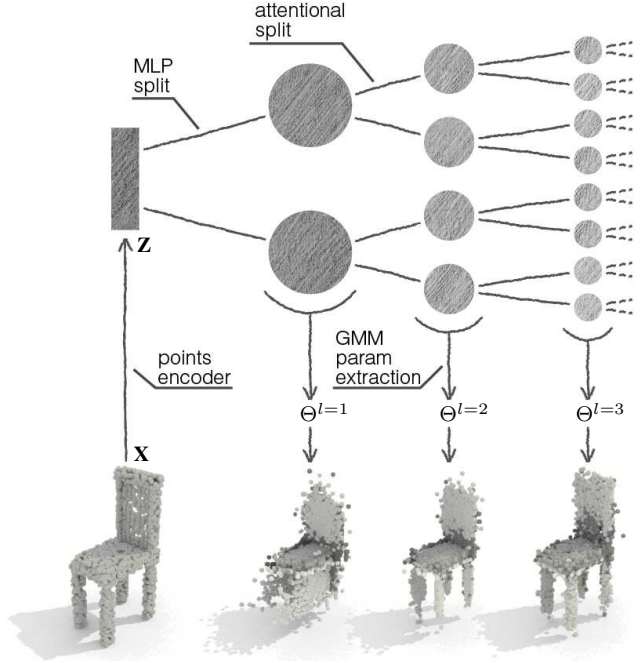

Figure 2: *Method Overview*. PointGMM learns a hierarchical GMM representation of the input X. Each depth d of the tree corresponds to a group of GMMs (with parameters Θ^d) representing the input distribution at different resolutions.

hard partitioning scheme, where we assign each point in the point cloud to a mixture model. For a point with posteriors $\gamma_{ij}^{l=d-1}$ in level $d-1$, the point is assigned to the parent with the highest expectation in level $d - 1$, *i.e.*, that corresponds to $\arg\max_{j} \gamma_{ij}^{l=d-1}$.

Log-likelihood. Finally, given a point cloud X, the log-likelihood for each depth in the hierarchy is given by

$$\ell_{hGMM}\left(X|\Theta^{l=d}\right) =$$
$$\begin{cases} \log p\left(X|\Theta^{l=1}\right) & \text{if } d = 1 \\ \log p\left(X|\gamma^{l=d-1}, \Theta^{l=d}\right) & \text{else.} \end{cases}$$
(4)

hGMM sampling. When sampling from the hGMM, we refer to the leaf nodes of the tree at level D as a single mixture model to sample from. Therefore, each leaf weight is scaled by its ancestors' weights, *i.e.,* the fixed leaf weight becomes

$$\hat{\pi}_j = \prod_{d=1}^{D} \pi_j^{l=d||d-1}.$$
(5)

3.2. PointGMM

PointGMM proposes a novel encoder-decoder network that builds an hGMM tree from an input point cloud. We demonstrate two different use cases of PointGMM: shape

generation and registration. The encoder architecture is task-specific, and we use the same PointGMM decoder for both tasks.

The PointGMM decoder network is illustrated in Figure 2. First, an input point cloud X goes through an encoder network resulting in a latent vector Z. Then, the embedding is used by the decoder to generate the hGMM representation. Starting at the root (with latent vector Z), the decoder generates the nodes in the hGMM tree in a top-down *splitting* process. For each node, the network generates a feature vector, which is used to extract the GMM parameters.

MLP split. For every depth in the tree, the decoder splits each node into children. We apply a multi-layer perceptron (MLP) on each node. The output of the MLP is a feature map which is used later to extract the GMM parameters. Specifically, the MLP creates a feature vector h that represents each Gaussian in the mixture. Starting at the root, the initial latent vector is the encoder output Z.

Attentional split. After the first MLP split, we have the first level in the hGMM that represents a single Gaussian mixture. Since we want to *combine* information between siblings, *i.e.,* children of the same parent, we utilize a self-attention module [42]. This enables effective information passing between siblings to generate a higher quality mixture. Following the first MLP split into children, each child will split recursively into new children using attentional splits.

To calculate the self-attention, we follow [29, 42], and compute for each node j, its query $Q_j \in \mathbb{R}^{d_k}$, key $K_j \in \mathbb{R}^{d_k}$ and value $V_j \in \mathbb{R}^h$. This is done using three fully connected layers such that $Q_j = f_Q(j)$, $K_j = f_K(j)$, $V_j = f_V(j)$, which share the same parameters for all nodes of the same depth. The output of these layers are used for calculating the attention weights between the siblings $Si(j)$ of each node j, which are given by $\hat{\alpha}_j = \left\{ \dfrac{Q_{\hat{j}}^T K_{\hat{j}}}{\sqrt{d_k}} \mid \hat{j} \in Si(j) \right\}$.

As in regular attention, we transform this vector into a distribution by $\alpha_j = softmax(\hat{\alpha}_j)$. The node descriptor before splitting is $\sum_{\hat{j} \in Si(j)} \alpha_{j,\hat{j}} V_{\hat{j}}$, which is splitted into its children by a MLP with one hidden layer as described above. We repeat this process of attention followed by a MLP split until reaching the pre-determined depth.
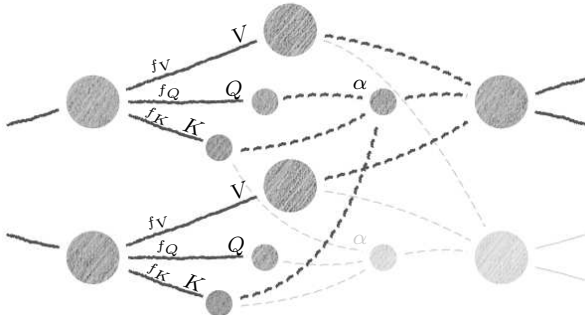
GMM parameter extraction. The children of each node in the hGMM tree corresponds to a GMM. Each child node j contributes one Gaussian $N(\mu_j, \Sigma_j)$ with weight π_j to the GMM. We extract the Gaussian parameters from the feature vector h of each node by applying an MLP with one hidden layer. The output of this MLP $\in \mathbb{R}^{16}$, corresponds to the parameters $\left\{ \hat{\pi}_j \in \mathbb{R}, \ \mu_j \in \mathbb{R}^3, \ \hat{U}_j \in \mathbb{R}^{3 \times 3}, \ \sqrt{\lambda_j} \in \mathbb{R}^3 \right\}$, which are used afterwards to create the parameters $\Theta_j = \{\pi_j, \mu_j, \Sigma_j\}$ of each Gaussian in the GMM.

We ensure that the mixture weights sum to probability 1 by applying the *softmax* function to all the node weights π_i in the group of siblings. The covariance Σ_j is calculated from the eigen-decomposition $\Sigma_j = U_j^{-1} D_j U_j$ where D_j is diagonal with the vector λ_j as its diagonal and U_j is a unitary matrix resulting from applying the Gram-Schmidt orthogonalization process on $\hat{U}_j$. This decomposition is required since we would like to restrict Σ to be a positive definite matrix (PSD). This characteristic is guaranteed as the decomposition is of positive real eigenvalues and their eigenvectors (columns of U).

hGMM loss. As mentioned above, the loss function for optimizing PointGMM is, naturally, the negative log-likelihood (ℓ_{hGMM}) of a given point cloud X under the networks output parameters, that is:

$$\mathcal{L}_{hGMM}(X, \Theta) = -\frac{1}{|X|} \sum_{d=1}^{D} \ell_{hGMM}\left(X | \Theta^{l=d}\right). \quad (6)$$

ℓ_{hGMM} are summed over all depths until reaching the the finer level. In this way, the network is restricted to represent the shape by each of the GMMs in each level. This creates a spatial connection between the tree nodes, where children give a finer description of their Gaussian parent.

3.3. Shape generation

PointGMM can be trained for shape generation using a variational auto encoder (VAE) framework [24]. In our experiments, we adopt the PointNet architecture [34] for the encoder. PointNet encodes a given point cloud $X \in \mathbb{R}^{N \times d}$ by employing an MLP on each point. PointNet maintains the order invariance property by applying a global max-pooling operator over each dimension resulting in an order invariant embedding $\hat{Z}$.

This embedding contains the parameters Z_μ and Z_σ, which are used for generating the latent vector $Z = Z_\mu + \epsilon Z_\sigma$, where $\epsilon \sim \mathcal{N}(0, I)$. This makes our model *generative*, enabling a smooth latent space which can randomly be sampled.

The encoding Z is the input to the PointGMM decoder described above, which outputs the hGMM parameters. The
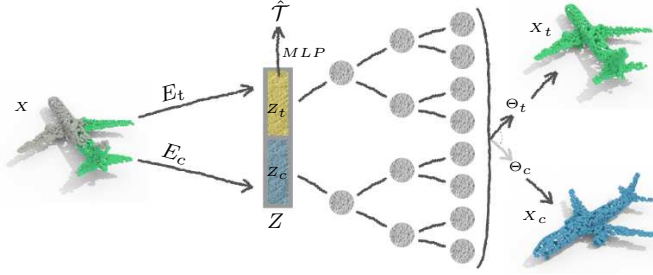
Figure 3: *Registration overview.* The input point cloud X (green) is disentangled into two different embeddings: transformation (Z_t) and the shape (Z_c); via two parallel encoders E_t and E_c.

shape generation loss is composed of $\mathcal{L}_{hGMM}$ (6) and the KL (Kullback- Leibler) divergence with respect to the Normal distribution to encourage a continuous latent space

$$\mathcal{L}_g = \mathcal{L}_{hGMM}(X, \Theta) + \gamma \mathcal{L}_{KL}\left[\mathcal{N}\left(Z_\mu, Z_\sigma^2\right) \| \mathcal{N}(0, I)\right]. \tag{7}$$

To generate novel shapes, we can sample or interpolate vectors from the learned latent distribution. Latent vectors are decoded by PointGMM to generate the hGMM representation. Given the probabilistic representation defined by the hGMM, we can sample the underlying surface of the synthesized shape in any desired resolution.

3.4. Shape registration

For the registration task, we would like to find a transformation to align two partial point clouds. The core assumption of our approach is that for each shape in our data, there is a canonical form. Thus, our registration strategy is based on finding the transformation of each given shape with respect to its canonical form.

Data processing. To train our network to learn a canonical position, we use an aligned dataset, ShapeNet [8], for training. We use partially sampled and rotated shapes as an input to our network, generated by the following simple steps. First, we sample the shape surface to get the canonical point cloud X_c. Then, the shape is randomly rotated with angle ϕ around the z axis, creating X_r. The partial point cloud is obtained by sampling a subset of the points. Finally, we translate the point cloud to the center by v resulting in the partial and full point clouds X and $X_t = X_r + v$. To summarize, we obtain the input point cloud X together with point clouds X_c, X_t and transformation $\mathcal{T} = \{\phi, v\}$, which satisfy $X_t = \mathcal{T} \cdot X_c$. These inputs are used as a supervision to our method as detailed below.

Training. Our registration framework (illustrated in Figure 3) aims to disentangle the shape from the transformation of an input point cloud. First, two different encoders E_t

(*transformation*) and E_c (*canonical*) process the input point cloud X in parallel. The transformation encoder E_t learns to map the Cartesian coordinates of X to a latent vector Z_t that is *agnostic* to the shape. On the other hand, the shape encoder E_c learns to embed rotation-invariant features [9] of X to a latent vector Z_c, which is agnostic to the orientation. The final latent vector Z is a concatenation of both the shape and rotation vectors $Z = Z_t \oplus Z_c$.

The network learns to disentangle the *shape* and *transformation* of the input point cloud in two different passes. In the *transformation pass*, we estimate the (transformed) complete shape X_t from the (transformed) partial input X. In this scenario the latent vector Z is a concatenation of the output of the transformation encoder and the shape encoder $Z_t \oplus Z_c$. The PointGMM decoder D builds the hGMM of the transformed shape from the latent vector $\Theta_t = D(Z_t \oplus Z_c)$. In addition, the encoding Z_t is passed to an MLP with one hidden layers, which generates an estimate $\hat{\mathcal{T}} = \left\{\hat{\phi}, \hat{v}\right\}$ of the transformation $\mathcal{T}$. Therefore, the loss used for the transformation pass is given by

$$\mathcal{L}_t = \mathcal{L}_{hGMM}(X_t, \Theta_t) + \mathcal{L}\left(\hat{\mathcal{T}}, \mathcal{T}\right). \tag{8}$$

The transformation loss $\mathcal{L}\left(\hat{\mathcal{T}}, \mathcal{T}\right)$ penalizes differences between translation v using the L_1-norm and rotation ϕ using cosine similarity. Thus, it is defined as

$$\mathcal{L}\left(\hat{\mathcal{T}}, \mathcal{T}\right) = \gamma_1 \mathcal{L}_1(\hat{v}, v) + \gamma_2 \mathcal{L}_{cos}\left(\hat{\phi}, \phi\right). \tag{9}$$

In the *shape* pass, we estimate the (canonical) complete shape X_c from the (transformed) partial input X. Since the input features of X to E_c are rotation-invariant, the *shape* latent vector Z_c is agnostic to the orientation. Since we want the network to disentangle the shape from the orientation, we do not use the transformation encoder, which gets Cartesian coordinates to estimate the transformation, and set the transformation latent vector with zeros $\vec{0}$. This leaves the final latent vector Z to be the output of the shape encoder concatenated with zeros $\vec{0} \oplus Z_c$. The PointGMM decoder D builds the hGMM of the transformed shape from the latent vector $\Theta_c = D\left(\vec{0} \oplus Z_c\right)$. Clearly, in the shape pass, there is no MLP-transformation update. Therefore, the loss for the shape pass is given by

$$\mathcal{L}_c = \mathcal{L}_{hGMM}(X_c, \Theta_c). \tag{10}$$

Our proposed architecture learns to align shapes into a global canonical orientation. Thus, we can easily obtain the alignment between two shapes by inverting the transformation of one and applying it to the other.

4. Experiments

In all experiments we train our network on shapes from the Shapenet dataset [8], where each network is trained on

hGMM ablation

	hGMM
	$k = 512$
	$k = 128$
	$k = 32$
	GMM
	$k = 512$
	$k = 128$
	$k = 32$

Attention ablation

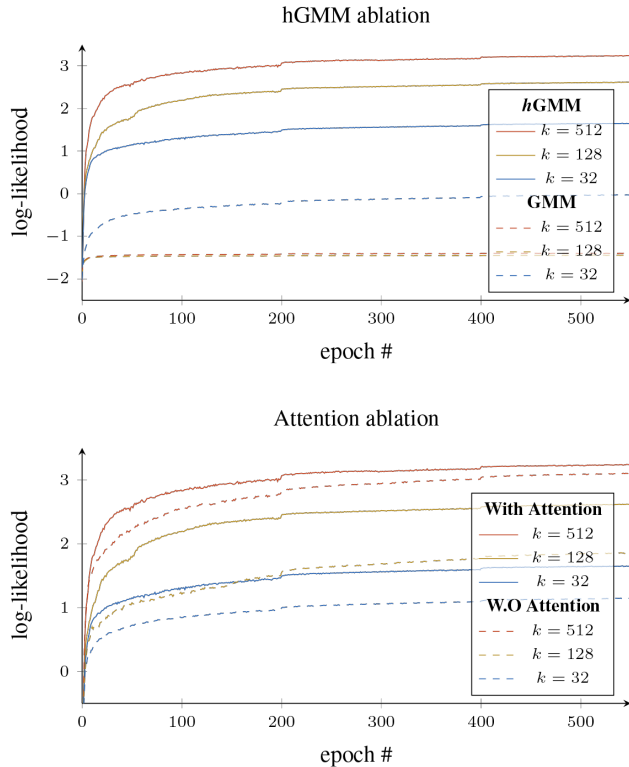|	**With Attention**
	$k = 512$
	$k = 128$
	$k = 32$
	W.O Attention
	$k = 512$
	$k = 128$
	$k = 32$

Figure 4: *Ablation study.* Top: significance of hGMM compared to vanilla GMM. Bottom: impact of using attention.

a single class.

We train each network for 1000 epochs using Adam optimizer [23] with a learning rate (lr) of 10^{-4} and lr decay of 0.5 applied in intervals of 200 epochs. The value V_j, key K_j and query Q_j have dimension 512, 64 and 64 respectively. All the networks use the following hGMM tree form: starting with the first split of the root to 8 children, with additional 3 splits to 4 children each. This results in 8, 32, 128 and 512 Gaussian mixtures in each level.

Our PyTorch [33] implementation as well as the pretrained models are available at https://github.com/amirhertz/pointgmm.

4.1. Shape generation evaluation

Following the VAE framework (see Section 3.3), we train class-specific networks on the chair, table and airplane datasets. We train with latent vector Z of size 256 and a weight of 1 for the $\mathcal{L}_{KL}$ applied with decay of 0.98 every 100 epochs. We use the trained networks to make interpolations between shapes from the dataset (see Figure 1). In addition, we sample from the latent space to generate new shapes (see Figure 5). Quantitative results based on the protocol of [16] are in the supplementary material, as well as additional qualitative results.

shape	Baseline	Point Decoder	PointGMM
chair	0.0612	0.1185	**0.0382**
car	0.1514	0.2073	**0.0447**
airplane	0.1688	0.1817	**0.0447**

Table 1: Decoder ablation. The registration results improve when adding a PointGMM decoder (compared to baseline), and excel compared to a vanilla point decoder.

Ablation Study. We perform an ablation study to highlight the contributions of each component of PointGMM. We use the chair dataset on the shape generation task. First, we examine the influence of the hierarchical architecture compared to a vanilla GMM architecture. In the vanilla GMM setting there is no hierarchy, instead the decoder generates all the GMMs once in the last layer of the network. We plot the log-likelihood of the hGMM and vanilla GMM nodes vs. epochs in Figure 4a for $k = 32$, 128 and 512 respectively. Observe the use of hGMM is crucial in preventing getting caught in a local minima. Namely, notice that the vanilla GMM with $k = 32$ Gaussians was able to learn something, indicating the solution is indeed a coarse-to-fine learning procedure. Moreover, as we increase the number of Gaussians, the hGMM network improves, while the vanilla GMM becomes even more susceptible to local minima (performance decreases). This phenomenon is also common in image GANs, where a coarse to fine methodology was found useful to help stabilize training [15].

We also verify the importance of the attention module within our network architecture by conducting the experiments for the different depths as before while the attention modules are disabled. We can see in Figure 4b that adding attention does improve the log-likelihood during training, especially in the shallower cases (fewer Gaussians). This shows that when the number of Gaussians is limited, it is even more crucial to enable information passing across the hGMM nodes using the attention module.

4.2. Shape registration evaluation

In the registration experiments, we assume the direction of gravity is known (a common feature of real scanning devices), resulting in partial point clouds in any position or orientation around the z axis.

We test our registration approach (see Section 3.4) by training three class-specific networks on chairs, airplanes and cars. We replace the tables class due to symmetry which makes the registration evaluation ambiguous. We use weights of 20 and 10 for the translation $\mathcal{L}_1$ and rotation $\mathcal{L}_{cos}$ losses respectively. The dimensions of Z_c and Z_t are 256 and 128, respectively. In all the training scenarios, the input point clouds are uniformly sampled from only 30% to 80% of the underlying shape area and are randomly rotated
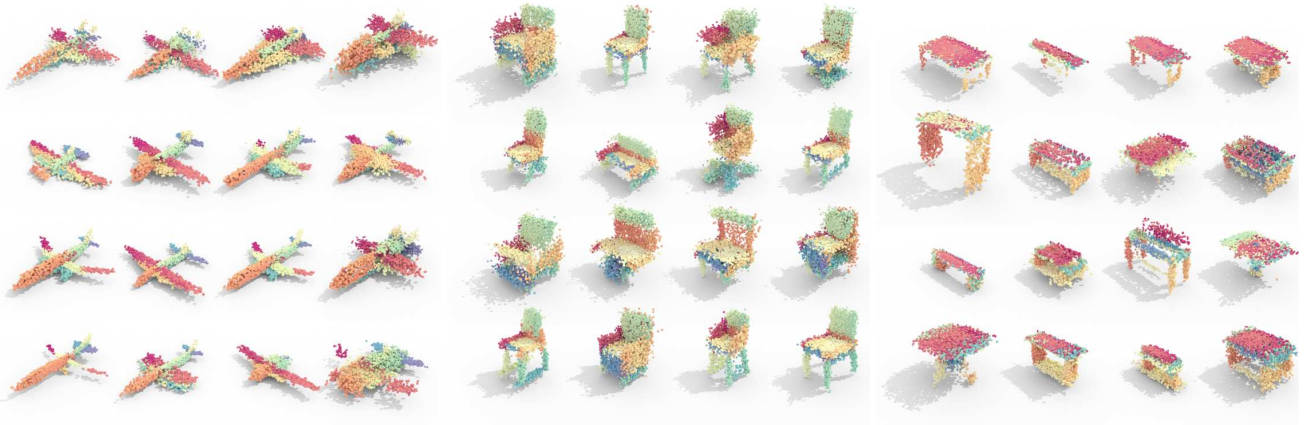
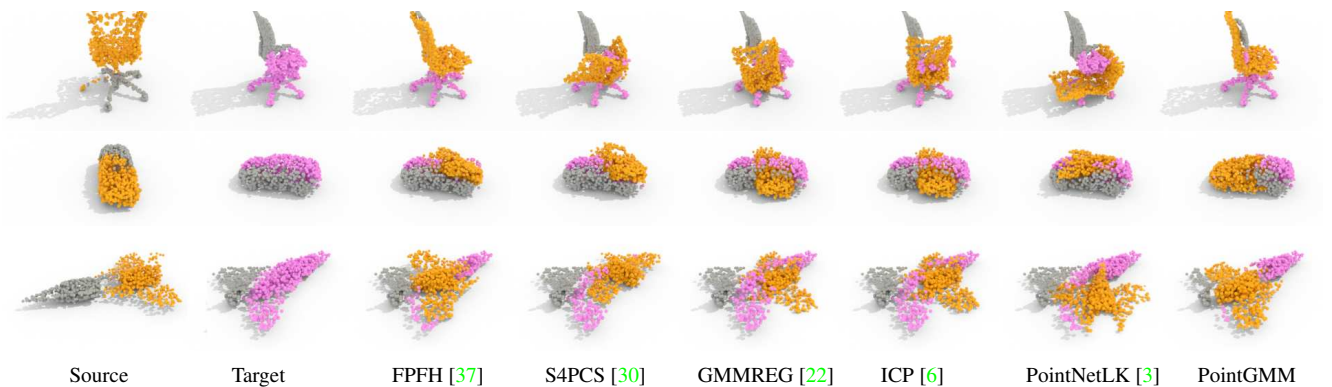Figure 5: Randomly sampled shapes using a PointGMM generative model.

| Source | Target | FPFH [37] | S4PCS [30] | GMMREG [22] | ICP [6] | PointNetLK [3] | PointGMM |

Figure 6: Qualitative results from the rigid registration comparison.

around the z axis, for example, see *source* and *target* point clouds in Figure 6.

Evaluation. We evaluate the performance of each network on four test sets covering two cases: *medium* ($\leq 30°$) and *large* rotations ($\leq 180°$). We also test different two different ranges of surface area coverage: 50% to 80% and 30% to 50%.

Each test consists of 1000 pairs of point clouds, where each pair is sampled from the same shape and are added with a random Gaussian noise of $\sigma = 0.02$. All test shapes are unseen during training.

Given a pair of source and target point clouds, we compute the rigid transformation from source to target. In order to evaluate alignment accuracy, we compute the mean squared error per point in the transformed source point cloud to the ground-truth target point cloud.

Comparisons. We run the same registration tests on five additional approaches. Two of them, *RANSAC-fast point feature histograms* (FPFH [37]) and *Super 4-points Congruent Sets (S4PCS [30])* are global registration approaches. We also compare to point set registration approaches: the

ubiquitous *Iterative Closest Point* (ICP [6]) and a GMM-based approach *robust point set registration using GMMs* (GMMREG [22]). Lastly, we compare to a recent deep learning approach PointNetLK [3], which we adapt for our test by training it on the same Shapenet dataset.

The quantitative MSE evaluation results are reported in Table 2 and qualitative examples in Figure 6. Our method achieves out performs the other approaches, most notably when large regions are missing with large rotation angles. Observe that the gap between our method and the point set approaches is small in the cases with medium rotations ($\leq 30°$) and larger sampling coverage (50% to 80%).

Ablation study. We perform two additional ablation comparisons to demonstrate the utility of PointGMM as a decoder within the registration framework (Figure 3).

As a *baseline*, we remove the decoder and train only the transformation encoder E_t to output the canonical transformation of an input point cloud. Thus, this baseline network is trained by the transformation loss $\mathcal{L}_{\mathcal{T}}$ (Equation 8), without the disentanglement hGMM losses.

We were also interested in comparing our approach

shape	max rotation (°)	sampling coverage %	RANSAC FPFH [37]	S4PCS [30]	GMMREG [22]	ICP [6]	PointNetLK [3]	Ours PointGMM
chair	30	50 - 80	0.2113	0.3343	0.0434	0.0430	0.1665	**0.0226**
chair	30	30 - 50	0.2804	0.3500	0.0842	0.0824	0.2617	**0.0496**
chair	180	50 - 80	0.2481	0.3479	0.2586	0.2578	0.2768	**0.0232**
chair	180	30 - 50	0.3132	0.3732	0.2829	0.2817	0.3386	**0.0574**
car	30	50 - 80	0.1352	0.2344	0.0399	0.04003	0.0566	**0.0246**
car	30	30 - 50	0.2134	0.2573	0.0884	0.08774	0.1647	**0.0552**
car	180	50 - 80	0.1754	0.2411	0.2134	0.2134	0.2288	**0.0290**
car	180	30 - 50	0.2357	0.2593	0.2354	0.2350	0.2548	**0.0702**
airplane	30	50 - 80	0.0765	0.1254	0.0632	0.0661	0.0798	**0.0312**
airplane	30	30 - 50	0.1501	0.1637	0.1052	0.1070	0.1301	**0.0490**
airplane	180	50 - 80	0.1485	0.1768	0.1983	0.1979	0.2023	**0.0350**
airplane	180	30 - 50	0.1961	0.2084	0.2293	0.2302	0.2308	**0.0489**

Table 2: Quantitative comparisons for rigid registration on partial shapes.

against a simple point-based decoder. Therefore, we replaced PointGMM with an implementation of PointNet auto-encoder [1]. This decoder applies a 4 layer MLP network on an input vector Z to output a point cloud with 2048 points. In this setting, the hGMM loss is replaced by the dual Chamfer distance.

We ran the decoder ablation on random rotations ($\leq$ 180°) with sampling area coverage between 30% to 80% in Table 1. Observe that using PointGMM performs better than without a decoder at all (baseline). On the other hand, vanilla point decoder did not always do better than the baseline (in fact, worse than the baseline in the chairs set). We believe this is due to the fact that the PointNet decoder struggles to generalize to the ill-posed problem of completing partial-shapes, while PointGMM can define the complete shape with better certainty using a probabilistic model. In other words, the probabilistic hGMM framework naturally supports handling partial shapes by predicting large variances in uncertain regions.

5. Conclusion

We have introduced a novel framework to transform point clouds into a hGMM representation. This representation has various desirable properties, including representing the shape probabilistically and in a coarse to fine manner. This facilitates coping with shapes that have missing parts or non-uniform sampling. The coarse levels capture global information of the shape better, while the probabilistic framework may alleviate randomness in the sampling process.

The key idea of representing a point cloud using a probabilistic framework was demonstrated to be useful for con-

structing a generative model for 3D objects, a problem which is known to be challenging. A common approach for this task directly synthesizes a point cloud, which often leads to a fixed number of points and fixed (uniform) sampling. Using PointGMM to learn hGMM for shape generation enables sampling the learned distribution to any desired resolution and possibly even non-uniformly.

Note that our model struggles to generate small sharp details (see Figure 5). We believe that one source of the problem is the fact that we use a VAE-based model, which is known to produce non-sharp results. Accordingly, incorporating adversarial, or other *detail-sensitive*, losses is expected to improve the representation of finer details.

An intriguing property that was revealed in our representation is that it provides an interpretable and consistent partitioning of a shape class within the hGMM leaf nodes (Figure 1). This implies that the learned hGMM is structured and captures the shape information. We exploited this inherent property to disentangle the shape from the orientation, and demonstrated how to use this effectively for the task of rigid registration.

We believe that PointGMM can be utilized in other shape analysis tasks such as detection, segmentation and reconstruction, to name a few. Note that hGMM is a more compact representation compared to the whole point cloud. Thus, such a conversion may allow a more efficient processing in these tasks and serve as an alternative to various sampling methods proposed to improve processing time [12].

Acknowledgement

---

[1] www.github.com/charlesq34/pointnet-autoencoder

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017. 2

[2] Dror Aiger, Niloy J Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust pairwise surface registration. In *ACM transactions on graphics (TOG)*, volume 27, page 85. Acm, 2008. 2

[3] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7163–7172, 2019. 2, 7, 8

[4] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters*, 3(4):3145–3152, 2018. 1, 2

[5] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. Nesti-net: Normal estimation for unstructured 3d point clouds using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10112–10120, 2019. 2

[6] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992. 2, 7, 8

[7] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 1, 2

[8] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[9] Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang Lin. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4994–5002, 2019. 5

[10] Chu-Song Chen, Yi-Ping Hung, and Jen-Bo Cheng. Ransac-based darces: A new approach to fast automatic registration of partially overlapping range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1229–1234, 1999. 2

[11] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992. 2

[12] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to sample. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2760–2769, 2019. 8

[13] Benjamin Eckart, Kihwan Kim, and Jan Kautz. Hgmr: Hierarchical gaussian mixtures for adaptive 3d registration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 705–721, 2018. 1, 2

[14] Benjamin Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. Accelerated generative models for 3d point cloud data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5497–5505, 2016. 2, 3

[15] Karras et al. Progressive growing of gans for improved quality, stability, and variation. *arXiv:1710.10196*, 2017. 6

[16] Yang et al. Pointflow: 3d point cloud generation with continuous normalizing flows. In *ICCV*, 2019. 6

[17] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2

[18] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao(Richard) Zhang. SDM-NET: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019)*, 38(6):243:1–243:15, 2019. 2

[19] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Unsupervised cycle-consistent deformation for shape matching. *Computer Graphics Forum*, 38(5):123–133, 2019. 2

[20] Rana Hanocka, Noa Fish, Zhenhua Wang, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Alignet: Partial-shape agnostic alignment via unsupervised learning. *ACM Trans. Graph.*, 38(1):1:1–1:14, 2018. 2

[21] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: A network with an edge. *ACM Trans. Graph.*, 38(4):90:1–90:12, July 2019. 2

[22] Bing Jian and Baba C Vemuri. Robust point set registration using gaussian mixture models. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1633–1645, 2010. 7, 8

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 4

[25] Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Joan Bruna. Surface networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2540–2548, 2018. 2

[26] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018. 2

[27] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018. 2

[28] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018. 1, 2

[29] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018. 4

[30] Nicolas Mellado, Dror Aiger, and Niloy J Mitra. Super 4pcs fast global pointcloud registration via smart indexing. In *Computer Graphics Forum*, volume 33, pages 205–215. Wiley Online Library, 2014. 7, 8

[31] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575*, 2019. 2

[32] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017. 2

[33] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 6

[34] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 1, 2, 4

[35] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 2

[36] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3dim*, volume 1, pages 145–152, 2001. 2

[37] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009. 7, 8

[38] Aleksandr Segal, Dirk Hähnel, and Sebastian Thrun. Generalized-icp. *Robotics: Science and Systems*, 2009. 2

[39] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[40] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015. 2

[41] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643, 2017. 2

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia

Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 4

[43] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):146, 2019. 1, 2