Appendix : Learning Shape Abstractions by Assembling Volumetric Primitives

Shubham Tulsiani[1], Hao Su[2], Leonidas J. Guibas[2], Alexei A. Efros[1], Jitendra Malik[1]
[1]University of California, Berkeley [2]Stanford University
[1]{shubhtuls, efros, malik}@eecs.berkeley.edu, [2]{haosu, guibas}@cs.stanford.edu

A1. Consistency Loss via Differentiable Surface Sampling

We first re-iterate the consistency loss which aims for the assembled shape to be subsumed by ground-truth object and does so by sampling points on the surface of each primitive and penalizing the squared distance field w.r.t the ground-truth object at the sampled points.

$$L_2(\{(z_m, q_m, t_m)\}, O) = \sum_m \mathbb{E}_{p \sim S(\bar{P}_m)} \|\mathcal{C}(p; O)\|^2$$

Recall that to sample a point p on the surface of $\bar{P}_m$, one can equivalently sample p' on the surface of the untransformed primitive P_m and then rotate, translate p' according to (q_m, z_m).

$$p \sim S(\bar{P}_m) \equiv \mathcal{T}(\mathcal{R}(p', q_m), t_m); p' \sim S(P_m)$$

Note that the untransformed primitive P_m is an origin-centered cuboid parametrized by $z_m \equiv (w_m, h_m, d_m)$ – the dimensions along the three canonical axes. We now aim to show that we can derive gradients for z_m given gradients for a $p' \sim S(P_m)$. We do so using a re-parametrization trick where we aim to decouple the random sampling process from the parameters. The process of sampling from a cuboid's surface requires developing tow aspects. The first concerns how to sample from a particular face. The second required component is to decide which face to sample from – we show that we can sample equally from all faces if we assign sample importance weights.

Sampling Along a Face. Let us assume we want to to sample a point from a given cuboid face – say the one on the plane $x = w$. The process of sampling a point on this face can be written as -

$$u_x = 1; u_y \sim [-1, 1]; u_z \sim [-1, 1];$$
$$p' = (u_x w_m, u_y h_m, u_z d_m)$$

We can similarly define a sampling process for the other cuboid faces. Note that the coordinates of the sampled point p' are linear the primitive parameters and that given the random coefficients (u_x, u_y, u_z), it is straightforward to compute $\frac{\partial p'}{\partial z_m}$.

Importance Weights per Sample. To sample uniformly on the cuboid surface, we need to sample on each face with a probability proportional to its area. Unfortunately, it is unclear how this sampling process can be decoupled from the parameters p_m. Instead, we can simply sample equally from all faces and assign each sample an importance weight proportional to the area of the face it was sampled from. Under this implementation, the consistency loss as previously defined can be implemented as:

$$L_2(\{(z_m, q_m, t_m)\}, O) = \sum_m \sum_{p, w_p \sim S(\bar{P}_m)} w_p \|\mathcal{C}(p; O)\|^2$$

Here w_p is the importance weight of the corresponding sample and is proportional to the area of the face where it was drawn from. Note that w_p is also differentiable w.r.t z_m, therefore making consistency loss described differentiable w.r.t the predicted primitive parameters.

A2. Gradient computation for primitive existence probabilities p_m

We provide more details on how we use the REINFORCE algorithm to compute gradients for the primitive existence probabilities p_m. We discuss a more general implementation compared to the text – we assume $p_m \equiv (p_m^0, p_m^1)$ represents the probability of the available choices for the primitive – p_m^0 is the probability that the primitive does not exist and p_m^1 is the probability that it is a cuboid. Note that this can be modified to add other choices *e.g.* a third choice that the primitive is a cylinder.

Let l denote the loss $L(\{(z_m, q_m, t_m)\}, O)$ incurred for the particular sampling of z_m^e. Let r denote the external parsimony reward obtained if a primitive is sampled as not existing. Using these, the gradient for the predicted probability p_m is computed as -

$$\frac{\partial L}{\partial p_m^i} = \begin{cases} l - \mathbb{1}(z_m^e = 0)r, & \text{if } z_m^e = i \\ 0, & \text{otherwise} \end{cases}$$

It is typically also advised to subtract a 'baseline' (mean reward in the ideal scenario) to reduce the variance of the estimated gradients so we incorporate it using $l - b$ instead of l in the equation above where b is the running average of the losses incurred.

A3. Architecture and Initialization

Architecture and Hyper-parameters. Our network has five 3D convolution layers (with ReLU) of kernel 3, padding 1, stride 2. The number of channels (initially 4) are doubled after every layer. These are followed by 2 fc layers (with ReLU) with 100 units each, followed by a final layer to predict primitive parameters. We use ADAM for optimization and choose M to be more than the double the number of expected parts. M=20 for chairs, but was reduced to 15, 12 for planes, animals (to reduce iteration time).

Initialization. We initially bias z_m s.t. primitives are small cubes and $p_m = 0.9$ so every primitive is initially used but all other layer parameters/weights (and primitive positions, rotations *etc*.) are initialized randomly (our training discovers consistent solutions despite a random initialization because of commonalities in the data).

A4. Visualization

Shape COSEG Results. We visualize our parsings obtained on the 'chairs' subset of the Shape COSEG data. Figure 1 shows the meshes in the dataset and Figure 2 shows our inferred representations for these. Figure 3 visualizes the obtained unsupervised parsing of the original mesh. Figure 4 shows the annotated ground-truth labels and Figure 5 shows our predictions obtained by assigning each primitive a label from the available ground-truth labels.

Shape Embedding. We show in Figure 6, Figure 7, Figure 8 and Figure 9 the high-resolution images corresponding to the embeddings shown in the original text.

Figure 1: Meshes in the Shape COSEG dataset.

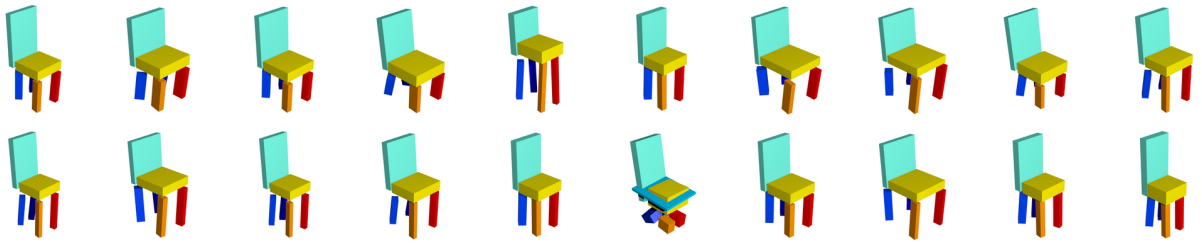

Figure 2: Inferred primitive based representations using our learned CNN on ShapeNet.

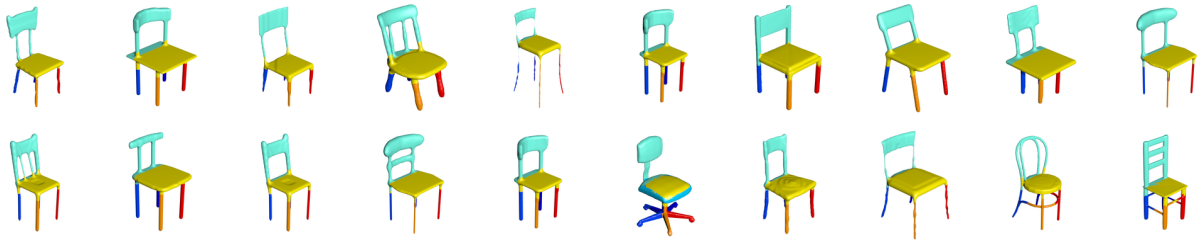

Figure 3: Primitive labels projected onto the original mesh.

Figure 4: Annotated Ground-truth part labels.

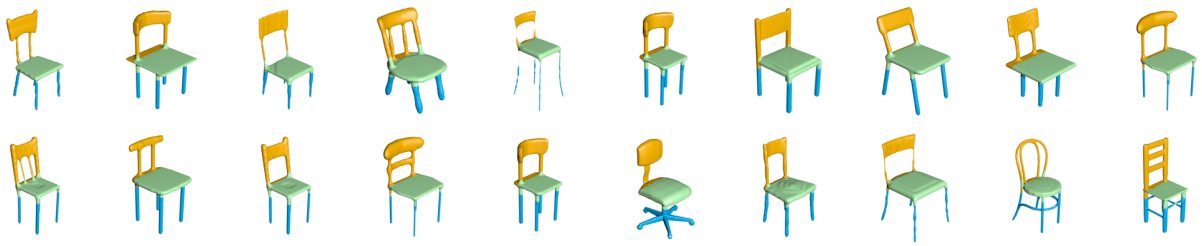

Figure 5: Our inferred part labels on the Shape COSEG dataset, obtained by assigning each primitive to a ground-truth label.

Figure 6: Embedding using voxel IoU as distance metric.

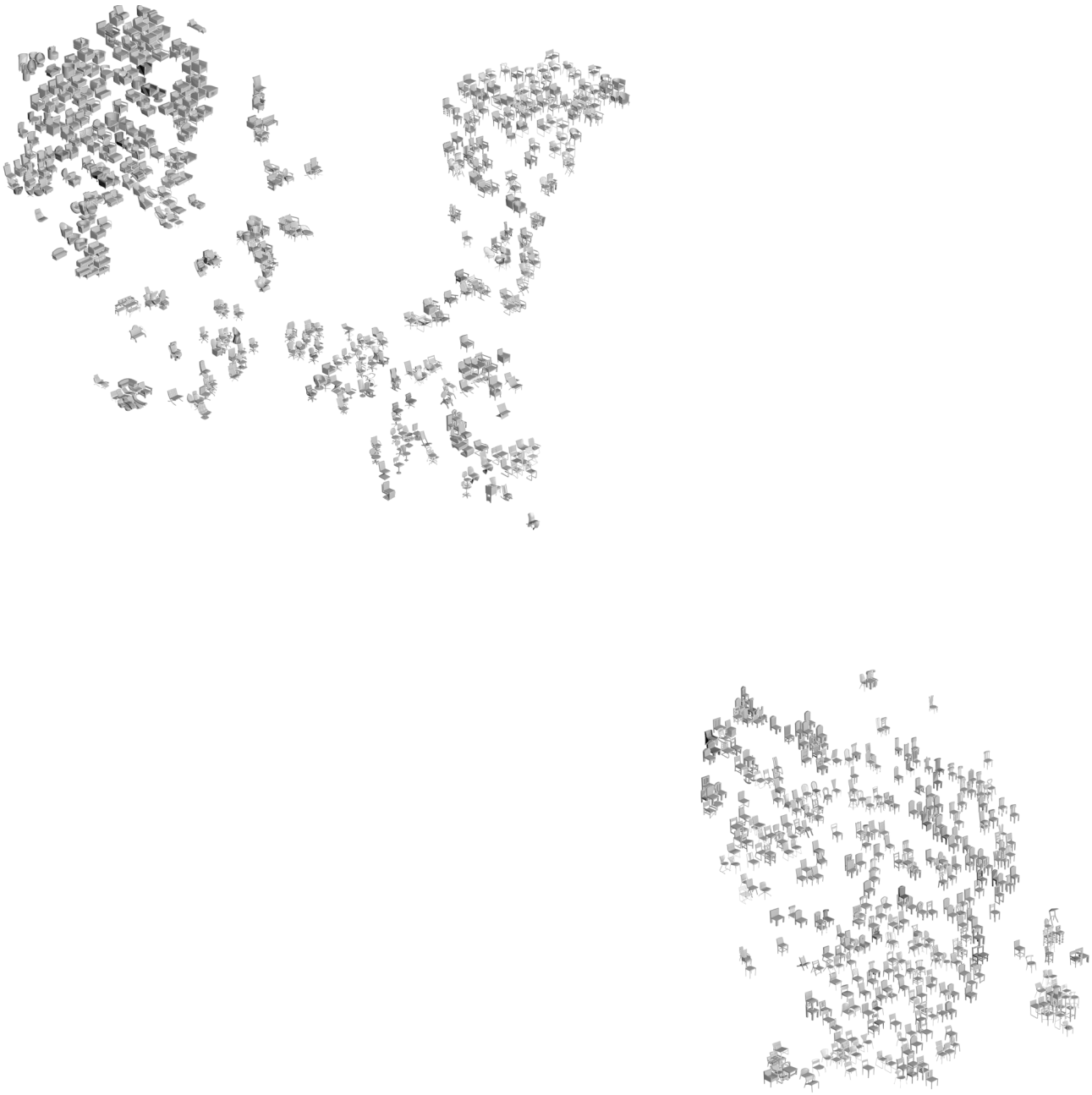

Figure 7: Embedding using squared euclidean distance in primitive based representation space.

Figure 8: Embedding using configurations of two specific primitives (chair back, seat) to compute the distances.

Figure 9: Embedding using the orientation of a specific primitive (chair back) to compute distances.