Supplemental Material
Scan2Mesh: From Unstructured Range Scans to 3D Meshes

1. Network Architecture Details

Figure 1 shows the detailed specification of our network architecture. Convolutions are specified by (kernel size, stride, padding), and are each followed by ReLUs. For both graph neural networks, each fully connected layer (except the last) is followed by an ELU, and within each pair of fully connected layers, we use a dropout of 0.5 during training and batch normalization following each pair. The node-to-edge and edge-to-node message passing operations are as defined in the main paper, through concatenation and summation, respectively.

# Verts	Train		Inference	
	Time(s)	Memory(GB)	Time(s)	Memory(GB)
100	0.15	0.38	0.13	0.07
200	1.15	1.44	1.04	0.12
300	4.62	3.29	4.32	0.27
400	15.8	5.96	14.24	0.55

Table 1: Time and memory during training and inference time for joint vertex-edge prediction as the number of predicted vertices grows. Note time measurements include a CPU hungarian algorithm computation (which currently dominates for larger number of vertices), and memory includes all allocated device memory.

2. Learned Feature Space

In Figure 2, we visualize the t-SNE of the latent vectors learned from our Scan2Mesh model trained for shape completion. We extract the latent vectors of a set of test input partial scans (the 256-dimensional vector encoding) and use t-SNE to visualize the feature space as a 2-dimensional embedding, with images of the partial scans displayed ac-

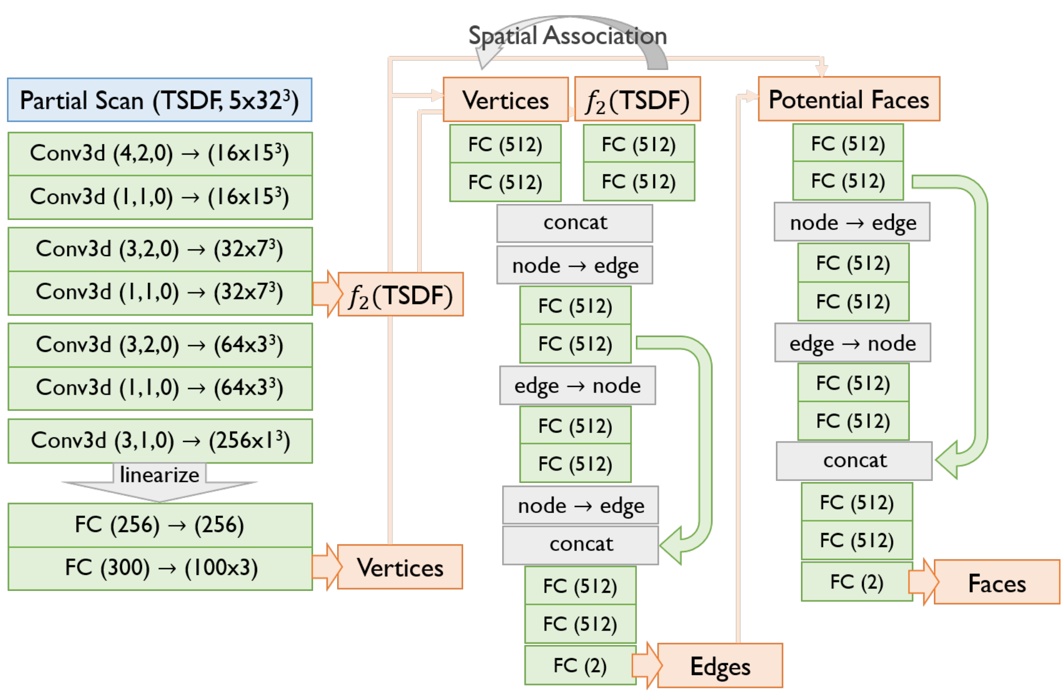

Figure 1: Network architecture specification for our mesh generation approach.

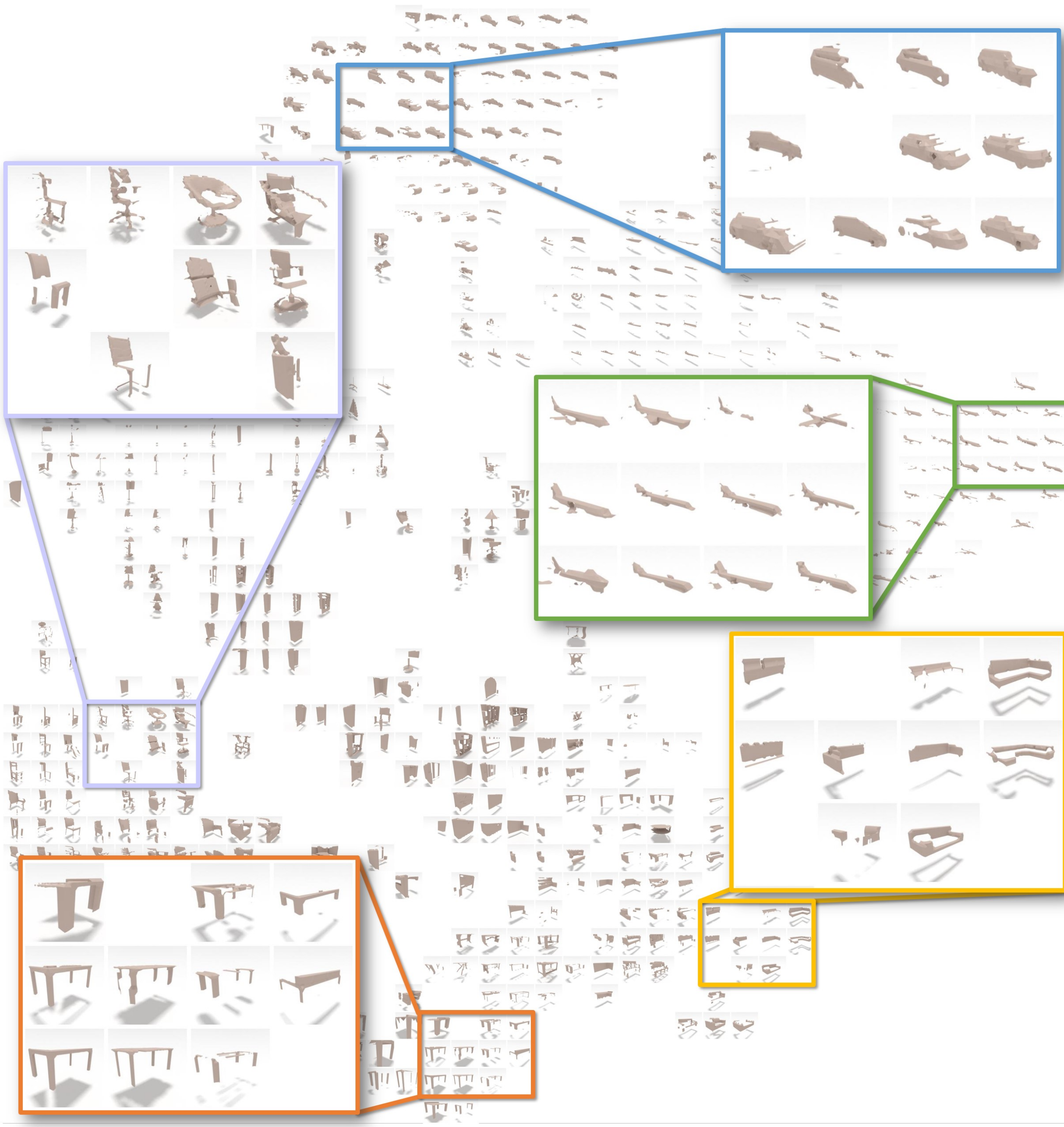

Figure 2: t-SNE visualization of the latent features learned from our Scan2Mesh model trained on shape completion. Features corresponding to input partial scans are visualized, with objects of similar geometric structure lying close together in this space.

cordingly. Our model learns to cluster together shapes of similar geometric structure.

3. Shape Generation

In the main paper, we demonstrate our mesh generation approach on the task of scan completion of shapes; we can also apply it to other tasks such as shape generation. Here, instead of learning an encoder for TSDF scans of shapes, we train a variational autoencoder [2] to produce mesh vertices and edges (on the same 8-class training set of ShapeNet [1] objects). Figure 3 shows shapes generated by drawing random samples from a unit normal distribution, along with nearest neighbor ground truth objects.