REIN: Flexible Mesh Generation from Point Clouds

Rangel Daroya Rowel Atienza Rhandley Cajote
University of the Philippines Diliman
Electrical and Electronics Engineering Institute
{rangel.daroya, rowel, rhandley.cajote}@eee.upd.edu.ph

A. Appendix

A.1. Network Details

Algorithm 1 shows the summary of the edge generation process as inspired by GraphRNN. From edges, the faces are formed by defining a face on any three vertices that are connected by edges. Algorithm 2 enumerates the steps to determine faces from the edge predictions.

Algorithm 1 Edge Generation Process

Initialize RNN-based f_{trans} and f_{out}
Initialize encoder ϕ of autoencoder
Input: Probability distribution P_{θ_i} parameterized by θ_i, start token SOS, end token EOS, empty graph state
Input: point cloud $x \in \mathcal{X}$ where $\mathcal{X}$ is the space of all input point clouds
Output: $\vec{e} \in \mathcal{E}$ where $\mathcal{E}$ is a sequence of edge predictions for all vertices in x
Output: sequence of edge predictions $\mathcal{E} \in \mathbb{E}$ for input point cloud x where $\mathbb{E}$ is the space of all edge predictions for all point clouds $\mathcal{X}$
$\vec{e}_1 = \texttt{SOS}$
$i = 1$
$h_1 = h'$
$z = \phi(x)$
repeat
  $i = i + 1$
  $h_i = f_{trans}(h_{i-1}, \vec{e}_{i-1}, z)$
  $\theta_i = f_{out}(h_i)$
  $\vec{e}_i \sim P_{\theta_i}$
until $\vec{e}_i$ is EOS
Return $\mathcal{E} = (\vec{e}_1, \vec{e}_2, ..., \vec{e}_i)$

Table 1 shows the network implementation details. The network is detailed by indicating the shape of the input and the shape of the output for each layer.

A.2. Meshlab Parameters

Table 2 shows the default parameters for Butterfly and Midpoint subdivisions. The parameters were used to process the meshes prior to training and testing. Table 3 shows the parameters used for normal estimation, BPA, and PSR. These were used to perform surface reconstruction on input point clouds.

Butterfly subdivision works under the assumption that mesh surfaces are originally curved. As a point is added to a surface, the planes adjacent to the new point have varied surface normal directions. The midpoint subdivision does not change the direction of the plane normals. Midpoint simply adds points midway between two existing points without altering the plane. Meshlab was used to perform both butterfly and midpoint subdivisions on the datasets.

Algorithm 2 Face Generation Process

Input: edge predictions $\mathcal{E}$ for point cloud x
Input: vertices $v_i \in \mathcal{V}$ for point cloud x
Initialize set adj
Output: set of faces $f_i \in \mathcal{F}$ for mesh $\mathcal{M}$ with vertices $\mathcal{V}$
for $e_{ab} \in \mathcal{E}$ **do**
  add $\{b\}$ to $adj[a]$
  add $\{a\}$ to $adj[b]$
end for
$i = 0$
for $e_{ab} \in \mathcal{E}$ **do**
  for c in $adj[a]$ and $adj[b]$ **do**
    $i = i + 1$
    $f_i = \text{sorted}(a, b, c)$
  end for
end for
Set f_1 as the reference
for $f_j \in (f_2, f_3, ..., f_i)$ **do**
  Flip order of vertices in f_j if in conflict with f_{j-1}
end for
Return $\mathcal{F} = (f_1, f_2, ..., f_i)$

A.3. Autoencoder Results

Figure 1 illustrates REIN's autoencoder performance. The autoencoder is shown to satisfactorily recreate an input point cloud.

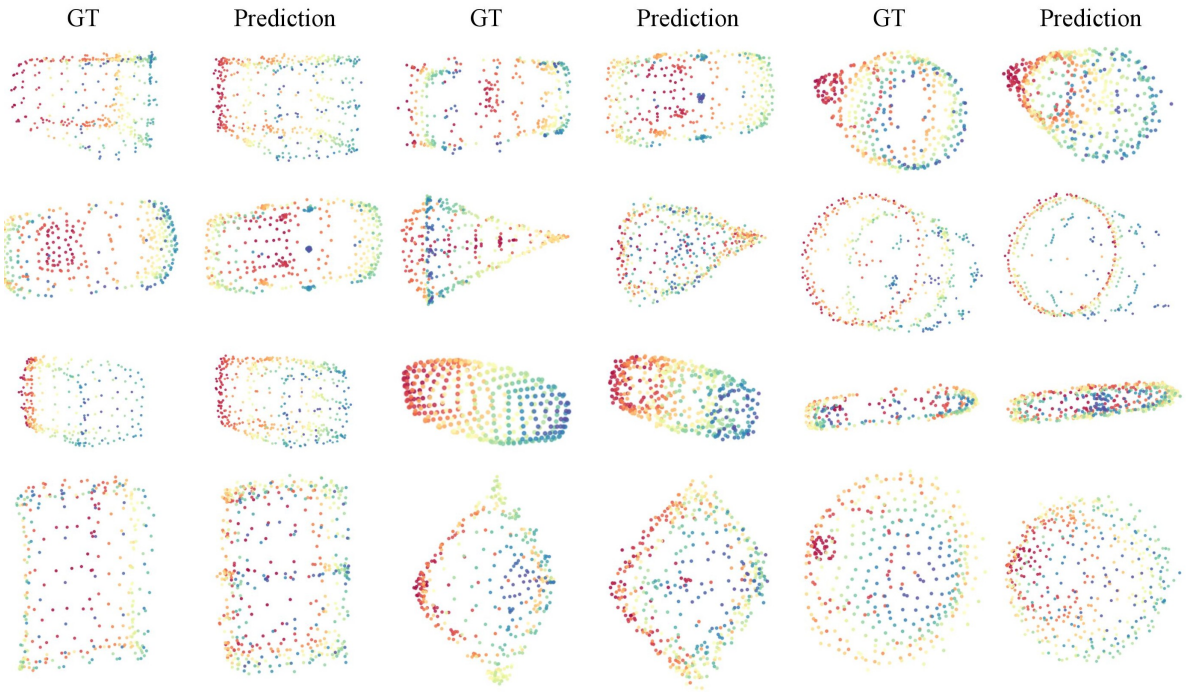

Figure 1. Sample point cloud autoencoder results of REIN on ShapeNet.

Table 1. Network details and corresponding layer (input, input shape → output shape) for the autoencoder, *State RNN*, and *Edge RNN*. All Conv1D are followed by ReLU and BN, while all GRU have 4 layers. The latent vector in the autoencoder is denoted by z.

AUTOENCODER
$x \in \mathbb{R}^{3xn}, z \in \mathbb{R}^{64}, n \in \mathbb{Z}^+$
$p = RandomSample(x, (3, n) \rightarrow (1, 3, 500))$
$p = Conv1D(p, (1, 3, 500) \rightarrow (1, 64, 500))$
$p = Conv1D(p, (1, 64, 500) \rightarrow (1, 64, 500))$
$p = Conv1D(p, (1, 64, 500) \rightarrow (1, 64, 500))$
$p = Conv1D(p, (1, 64, 500) \rightarrow (1, 64, 500))$
$p = MaxPool(p, (1, 64, 500) \rightarrow (1, 64, 1))$
$z = Reshape(p, (1, 64, 1) \rightarrow (1, 64))$
$p = ReLU(FC(z, (1, 64) \rightarrow (1, 64)))$
$p = ReLU(FC(p, (1, 64) \rightarrow (1, 64)))$
$p = FC(p, (1, 64) \rightarrow (1, 1500))$
$p = Reshape(p, (1, 1500) \rightarrow (1, 3, 500))$
$p \in \mathbb{R}^{1x3x500}$
STATE RNN
$x \in \mathbb{R}^{3xn}, z \in \mathbb{R}^{64}, n \in \mathbb{Z}^+$
$h = Concat(x, z)$
$h = FC(h, (1, n, 67) \rightarrow (1, n, 64))$
$h = GRU(h, (n, 64) \rightarrow (n, 128))$
$h = ReLU(FC(h, (1, n, 128) \rightarrow (1, n, 64)))$
$h = FC(h, (1, n, 64) \rightarrow (1, n, 16))$
$h \in \mathbb{R}^{1xnx16}$
EDGE RNN
$n \in \mathbb{Z}^+, m \in \mathbb{Z}^+$
$y = Zeros((n, 500, 1))$
$y = FC(y, (n, 500, 1) \rightarrow (n, 500, 8))$
$y = GRU(y, (m, 8) \rightarrow (m, 16))$
$y = ReLU(FC(y, (n, n, 16) \rightarrow (n, n, 8)))$
$y = FC(y, (n, n, 8) \rightarrow (n, n, 1))$
$y \in \mathbb{R}^{nxnx1}$

Table 2. Parameters used for Butterfly and Midpoint Subdivisions in Meshlab.

BUTTERFLY SUBDIVISION	
PARAMETER	VALUE
ITERATIONS	1
EDGE THRESHOLD (ABS)	0.0042521
EDGE THRESHOLD (%)	0.425213
MIDPOINT SUBDIVISION	
PARAMETER	VALUE
ITERATIONS	1
EDGE THRESHOLD (ABS)	0.010095
EDGE THRESHOLD (%)	1.00953

Table 3. Parameters used in Meshlab to apply BPA and PSR.

POINT NORMAL ESTIMATION	
PARAMETER	VALUE
NUM NEIGHBORS	10
VIEWPOINT POS	(0,0,0)
BALL PIVOTING ALGORITHM	
PARAMETER	VALUE
PIVOTING BALL RADIUS	0 (WOLRD UNIT), 0%
CLUSTERING RADIUS	20%
ANGLE THRESHOLD	90°
POISSON SURFACE RECONSTRUCTION	
PARAMETER	VALUE
OCTREE DEPTH	6
SOLVER DIVIDE	6
SAMPLES PER NODE	1
SURFACE OFFSETTING	1

A.4. Additional Results

Figure 2 shows additional results on ShapeNet Hull dataset. The number of vertices for the mesh outputs vary from 8 to 300. Figures 3 and 4 show additional results

on the ShapeNet Midpoint dataset. Similar to previous sample figures, BPA struggles in reconstructing areas with few vertices. PSR is prone to constructing closed surfaces around dense points. Reconstructions from PSR form separate closed surfaces around clusters. BPA and PSR can form good reconstructions in cases where the distribution of vertices are uniform. This can be seen in some sample cases in Figure 3. REIN is shown to perform well compared to BPA and PSR.

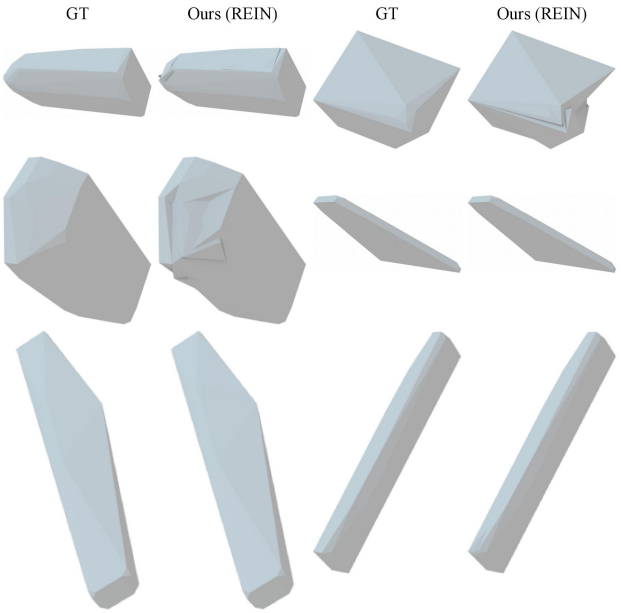

Figure 2. Sample results of REIN on ShapeNet Hull where number of vertices vary from 8 to 300.

Figures 5, 6, and 7 show qualitative results on the ShapeNet Patched dataset. Note that the results on ShapeNet Patched were obtained by using the network trained on ShapeNet Wrapped.

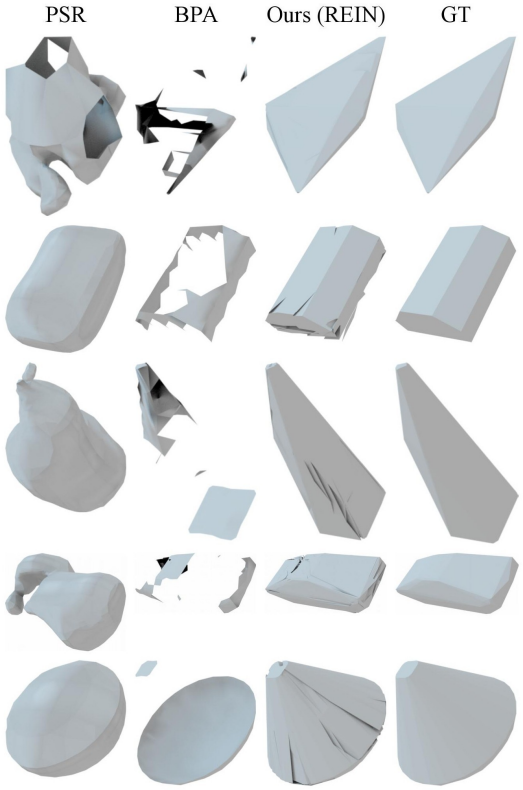

Figure 3. Sample results of REIN on ShapeNet Midpoint.

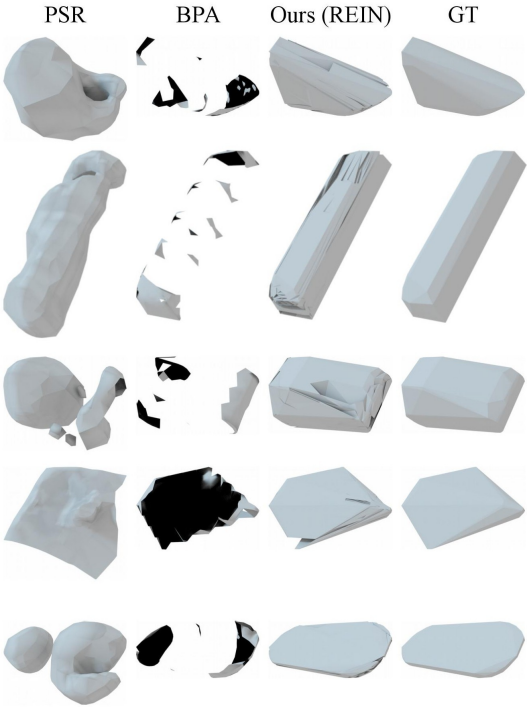

Figure 4. Sample results of REIN on ShapeNet Midpoint.

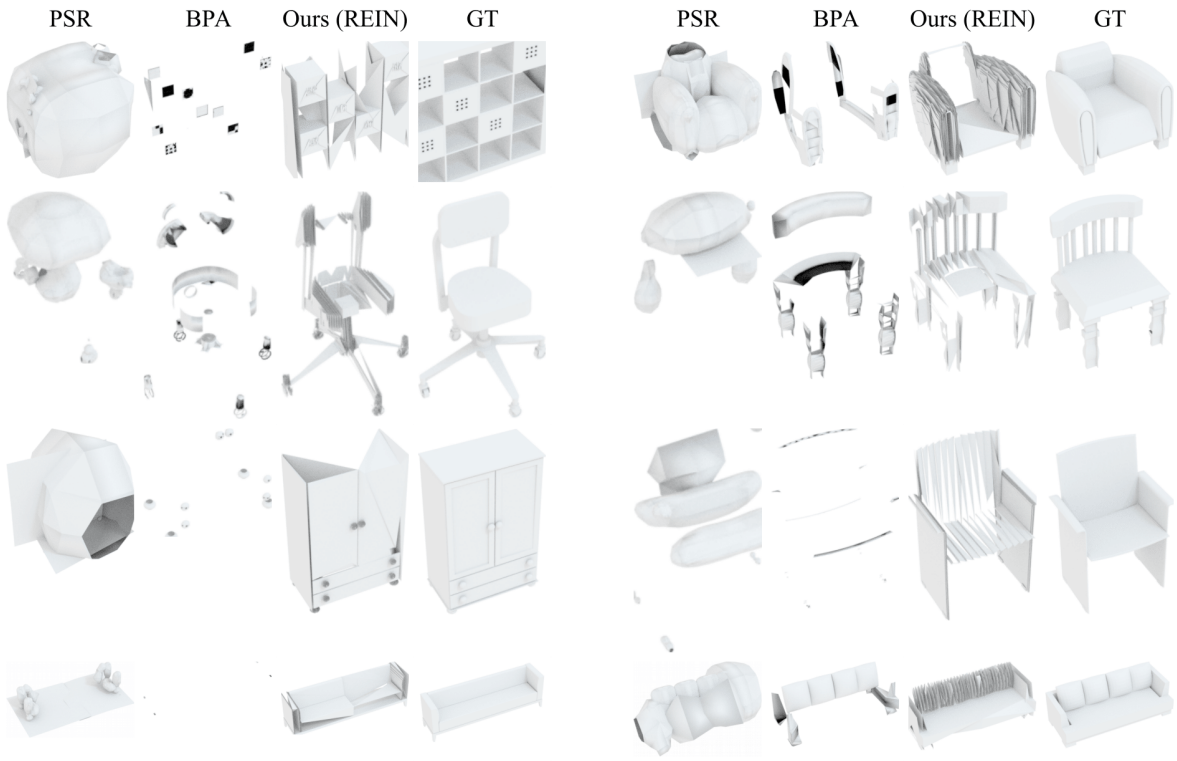

Figure 5. Sample results of REIN on ShapeNet Patched.

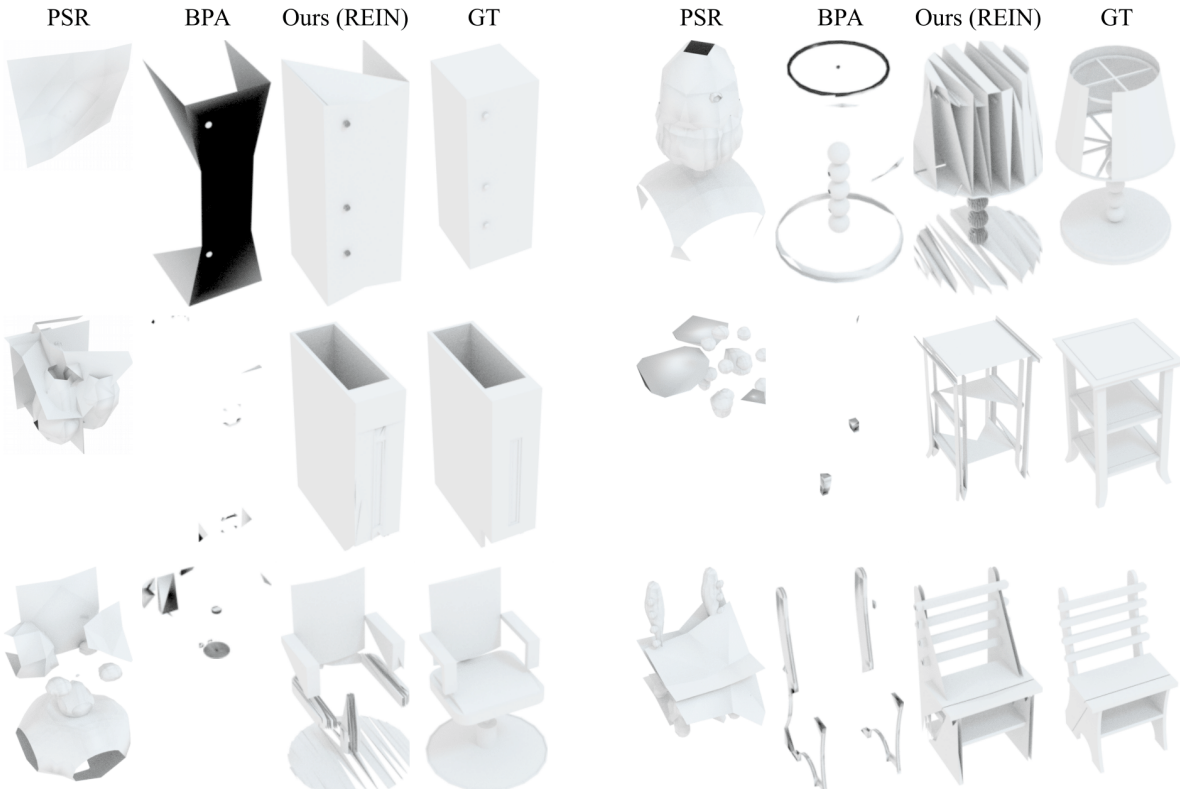

Figure 6. Sample results of REIN on ShapeNet Patched.

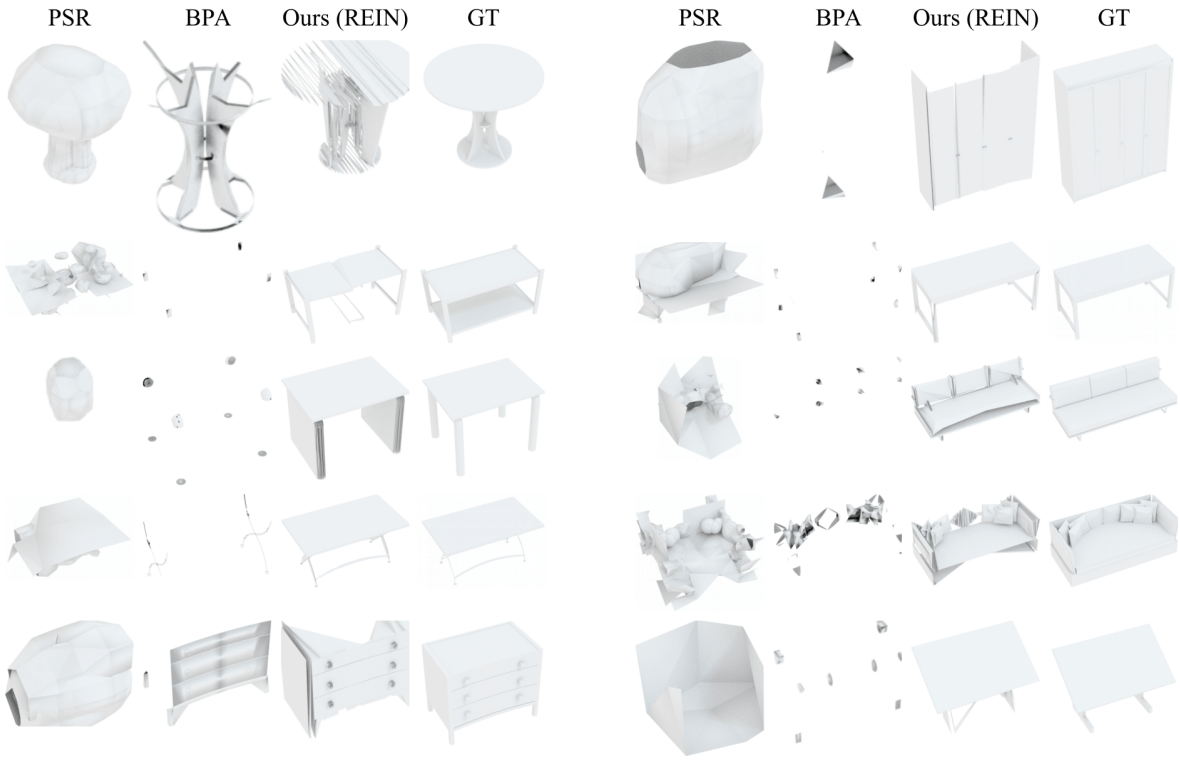

Figure 7. Sample results of REIN on ShapeNet Patched.