A Supplementary material

A.1 Details of network architecture

In this section, we provide details of the autoencoder, discriminator, and translator networks in LOGAN.

A set abstraction layer of PointNet++ is denoted as $SA(M, r, N, [l_1, ..., l_d])$ where M is the number of local patches, r is the radius of balls that bound the patches, N is the number of sample points selected in each patch, $[l_1, ..., l_d]$ are the widths of fully-connected layers used in local PointNet.

A fully connected layer is denoted as $FC(l)$ where l is its width. A dropout layer is denoted as $DROPOUT(p)$ where p is the probability that each element is dropped. We use $RELU$, BN and $MAXPOOL$ to represent ReLU activation layer, batch normalization layer and max pooling layer, respectively.

The architectures of our autoencoder, translator and discriminator are shown in Figure 1.

A.2 Visualization of latent space

As a supplement to the visualization of latent spaces in the paper, we show in Figure 2 the joint embedding of latent sub-vectors constructed by different autoencoders for *chair-table* . Same with the paper, the embedding is generated by t-SNE with hamming distance after discretize the values into integers.

A.3 Additional chair-table transform results

We show more examples of *chair* $\rightarrow$ *table* and *table* $\rightarrow$ *chair* transforms in Figure 3 and Figure 4, respectively.

A.4 Additional tall-short table transform results

We show more examples of *tallTable* $\rightarrow$ *shortTable* and *shortTable* $\rightarrow$ *tallTable* transforms in Figure 5 and Figure 6, respectively.

A.5 Additional results on the datasets of P2P-NET

We show more examples of our method for non-trivial transformation task on the datasets of P2P-NET, including *skeleton*$\rightarrow$*shape*(Figure 7, Figure 8), *scan*$\rightarrow$*shape*(Figure 9, Figure 10), *cross-sectional profiles*$\rightarrow$*shape*(Figure 11, Figure 12).

A.6 Additional font style/content transfer results

We show more examples of $A \leftrightarrow H$, $G \leftrightarrow R$, $M \leftrightarrow N$, *regularA/H* $\leftrightarrow$ *italicA/H*, *thinG/R* $\leftrightarrow$ *thickG/R* , and *wideM/N* $\leftrightarrow$ *narrowM/N* transfer in Figure 13, Figure 14, Figure 15, Figure 16, Figure 17 and Figure 18, respectively.

A.7 Additional armchair-armless chair transform results

We show more examples of *armchair* $\rightarrow$ *armless chair* and *armless chair* $\rightarrow$ *armchair* transforms in Figure 19 and Figure 20. To obtain the mesh for the transformed point cloud, we first compute the difference between the original point cloud and the output transformed point cloud. Suppose A is the point cloud with armrest and B is the point cloud without armrest, the difference can be computed as $D = \{p \in A | \min_{q \in B} ||p - q||_2^2 > \lambda\}$. The difference point cloud usually contain less points than optimal, therefore we then expand the difference point cloud by $D' = \{p \in A | \min_{q \in D} ||p - q||_2^2 < \lambda\}$. λ is set to 0.004 for arm removal and 0.008 for arm addition. We assume the chair is symmetric so we only take half of the difference point cloud (i.e. one arm only), denoted as P_{arm}. To remove an arm, we simply delete all the vertices and faces near P_{arm}. To add an arm, we first retrieve an arm from the database, then put the arm into position with respect to P_{arm}. Before arm retrieval, we normalize all arms (including P_{arm}) by putting each arm point cloud inside a unit box by scaling separately in X, Y and Z directions. We then retrieve top 10 candidate arms by their chamfer distance to P_{arm}, and randomly select one to put into the original mesh. To put the selected arm into the original mesh, we scale and reposition the selected arm, so that its bounding box is the same with P_{arm}.

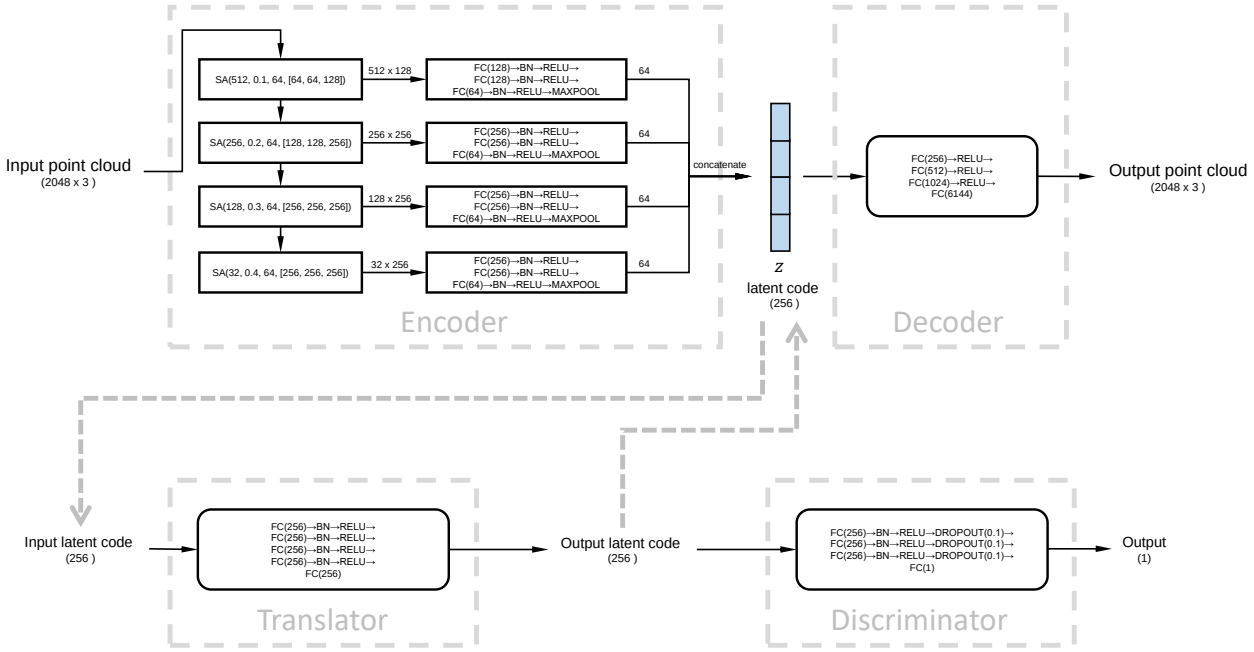

Figure 1: The architectures of our autoencoder, translator and discriminator.

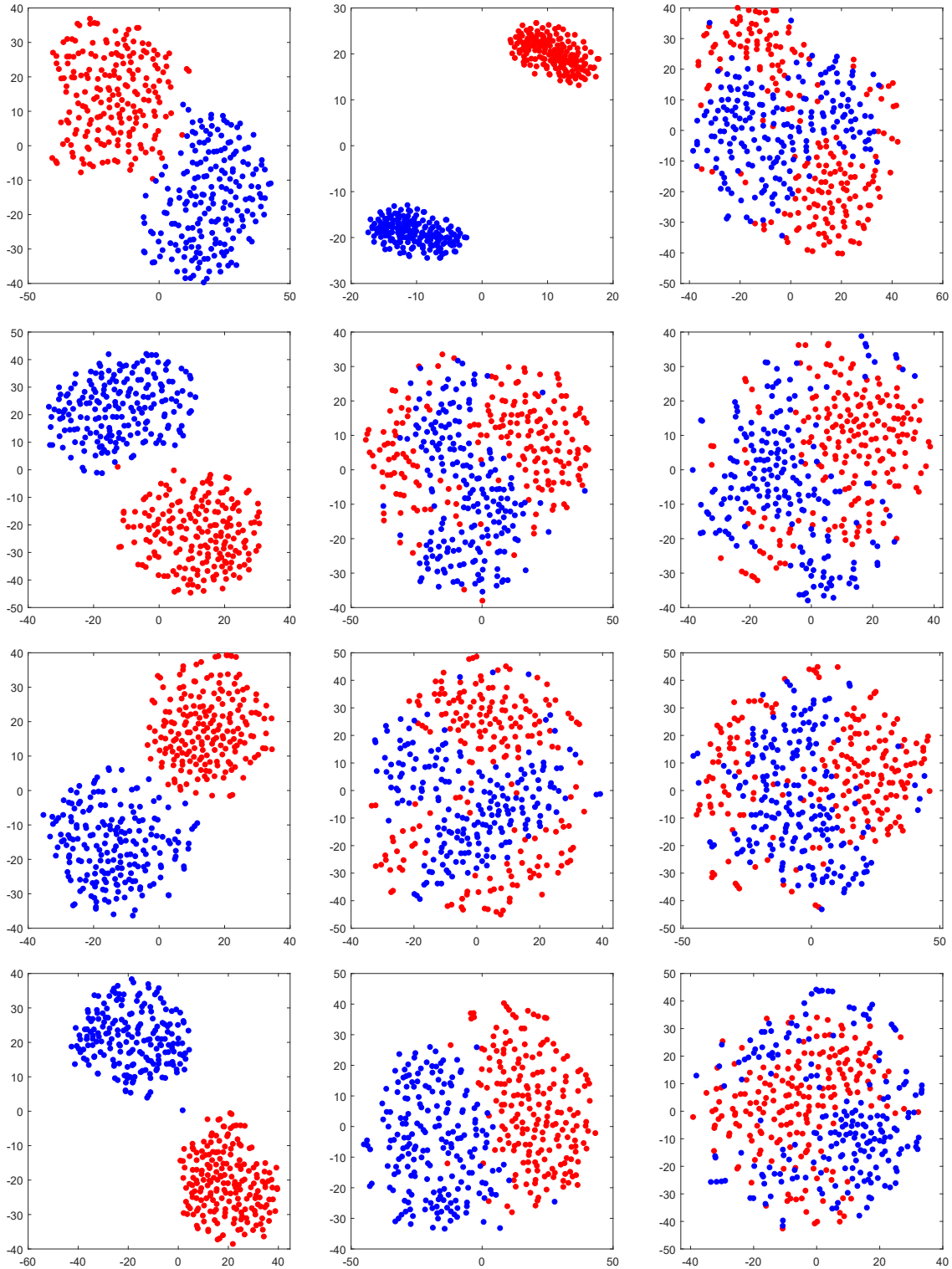

Figure 2: Joint embedding of the latent sub-vectors generated by our autoencoder(c) for chairs (red) and tables (blue). From top down, we show embedding result for z_1, z_2, z_3, z_4. For comparison, we also provide the embedding result for the corresponding sub-vectors in the codes generated by the two baseline autoencoders(a,b) described in Section 4.1 of the paper.

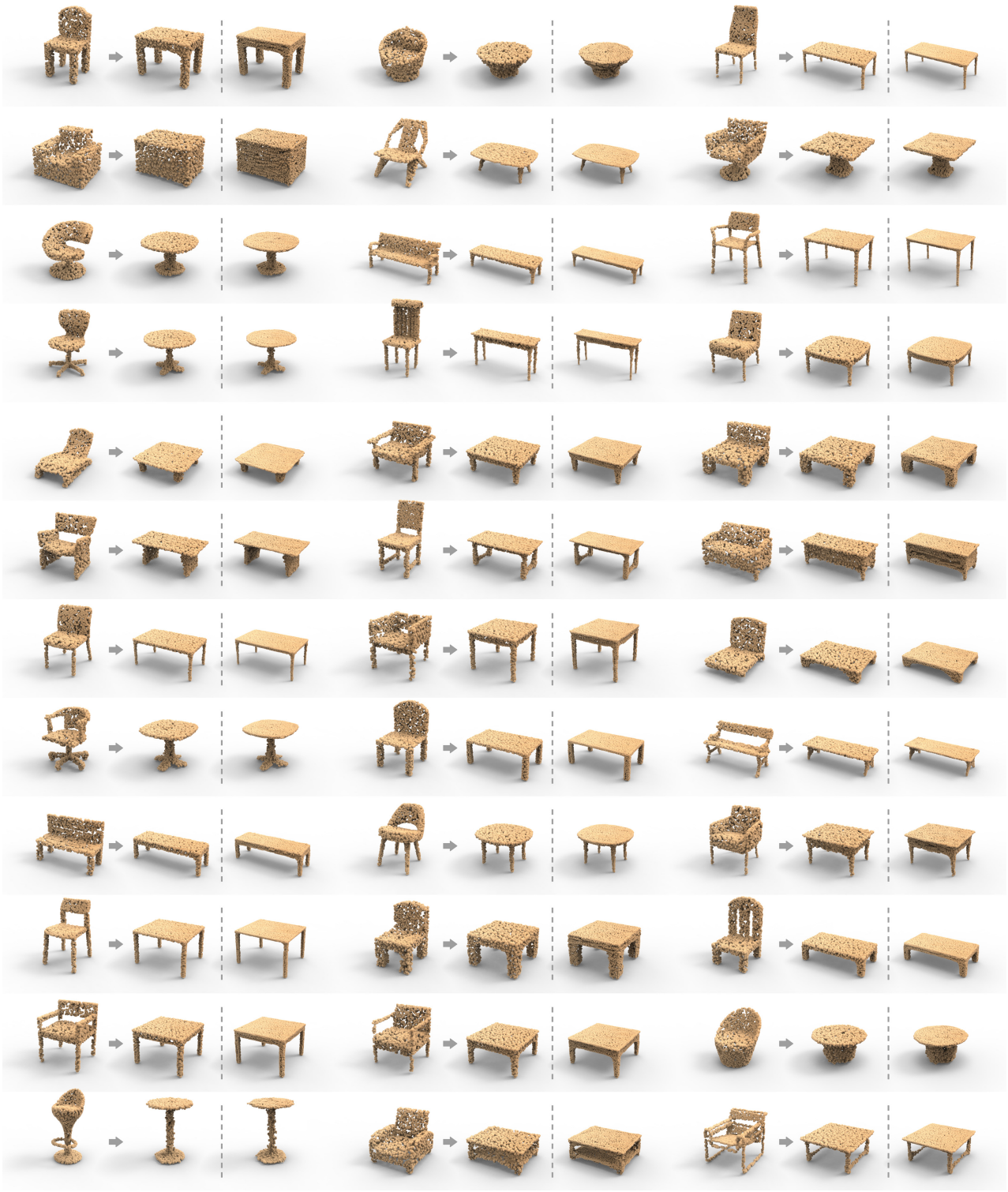

Figure 3: Results for *chair* → *table*. In each group (of 3 shapes), from left to right: input point cloud; output point cloud (2048 points); higher-resolution output point cloud (8192 points).

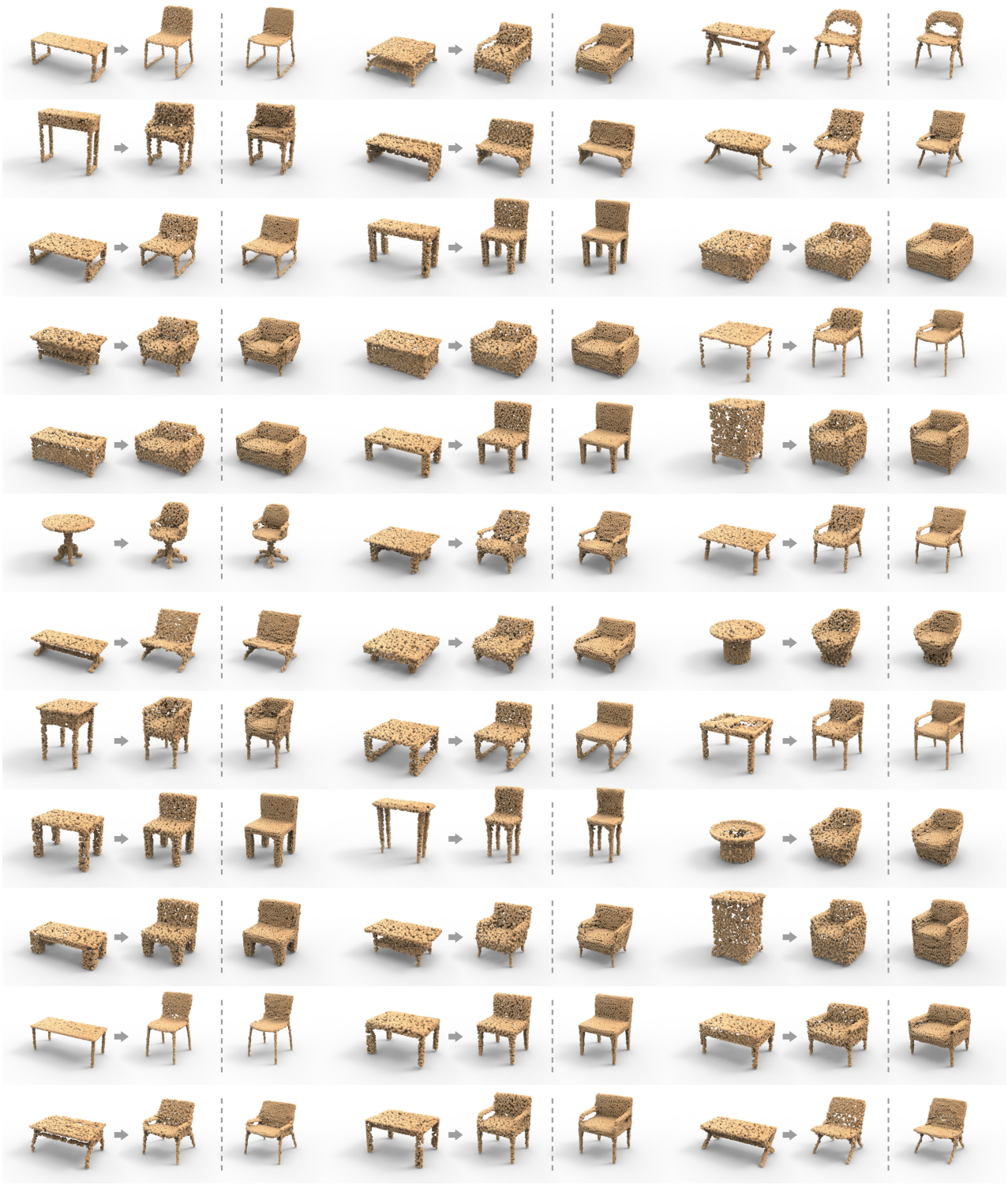

Figure 4: Results for *table* → *chair*. In each group (of 3 shapes), from left to right: input point cloud; output point cloud (2048 points); higher-resolution output point cloud (8192 points).

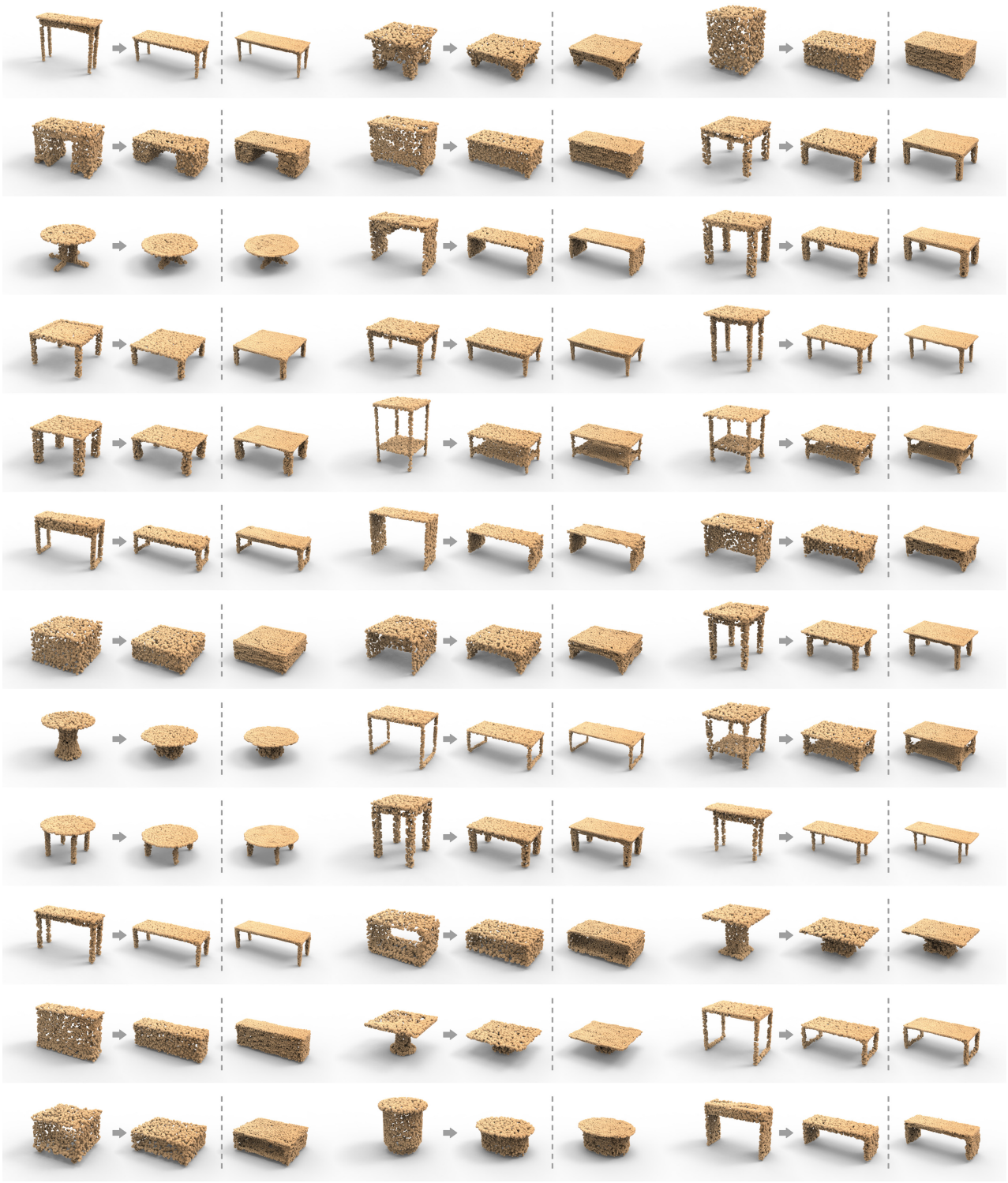

Figure 5: Results for *tall* → *short table*. In each group (of 3 shapes), from left to right: input point cloud; output point cloud (2048 points); higher-resolution output point cloud (8192 points).

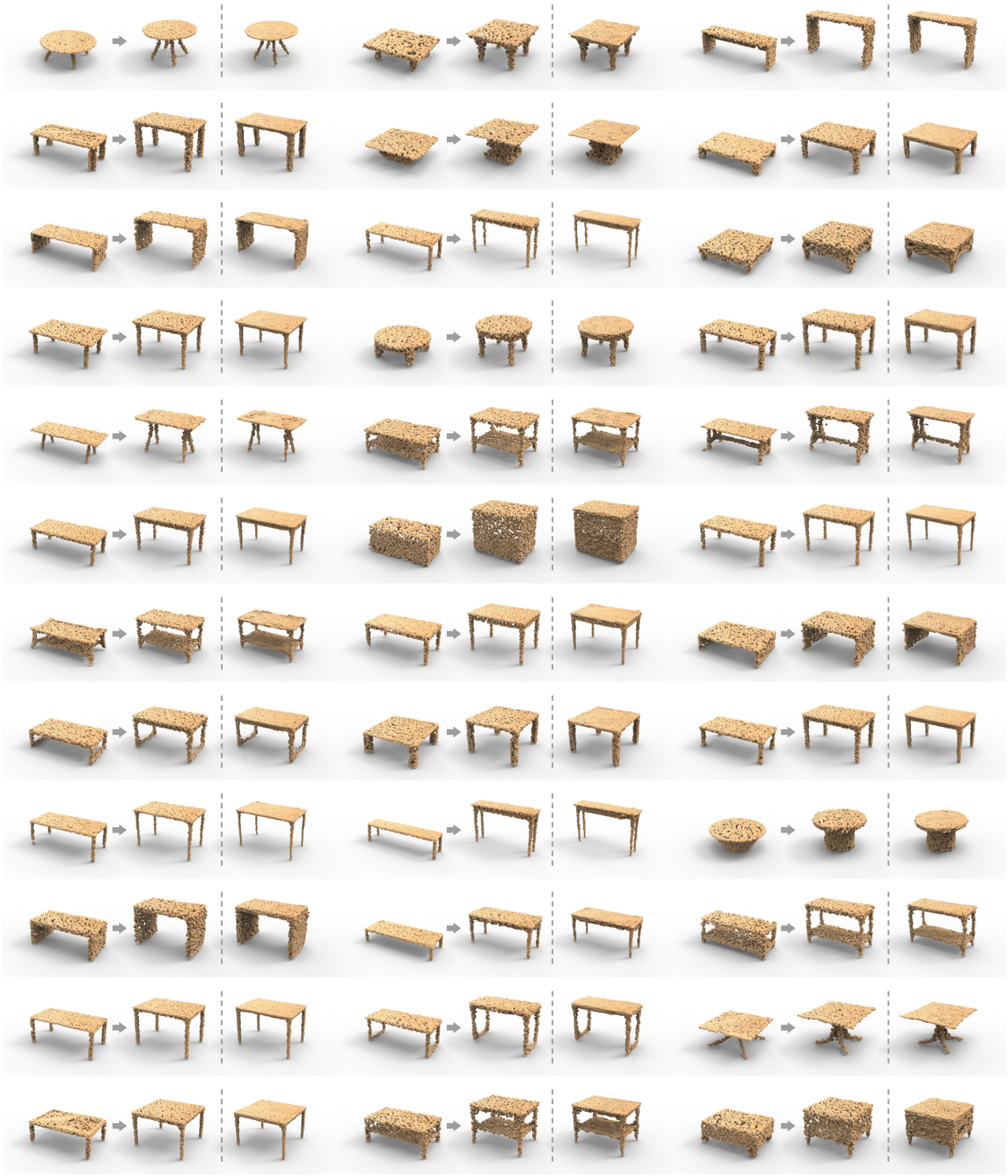

Figure 6: Results for *short* → *tall table*. In each group (of 3 shapes), from left to right: input point cloud; output point cloud (2048 points); higher-resolution output point cloud (8192 points).

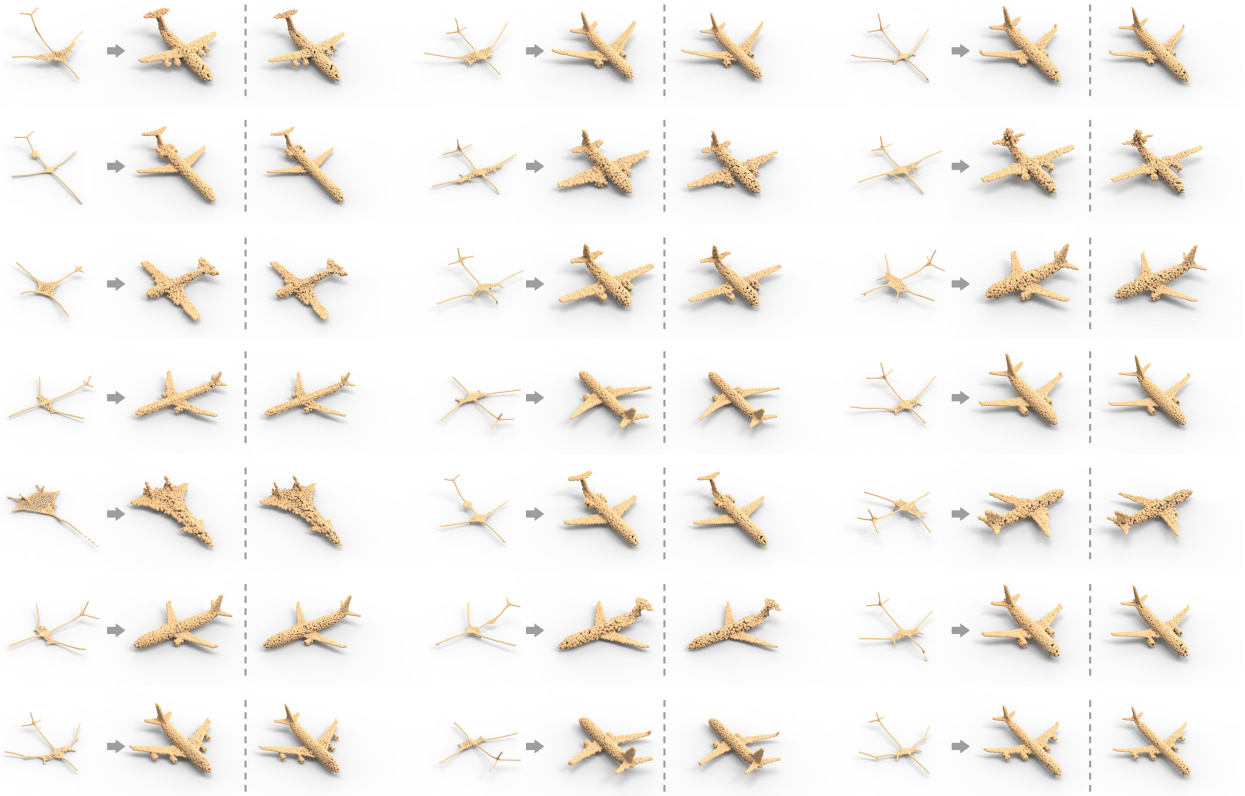

Figure 7: Results for *airplane skeleton* → *airplane*.

Figure 8: Results for *chair skeleton* → *chair*.

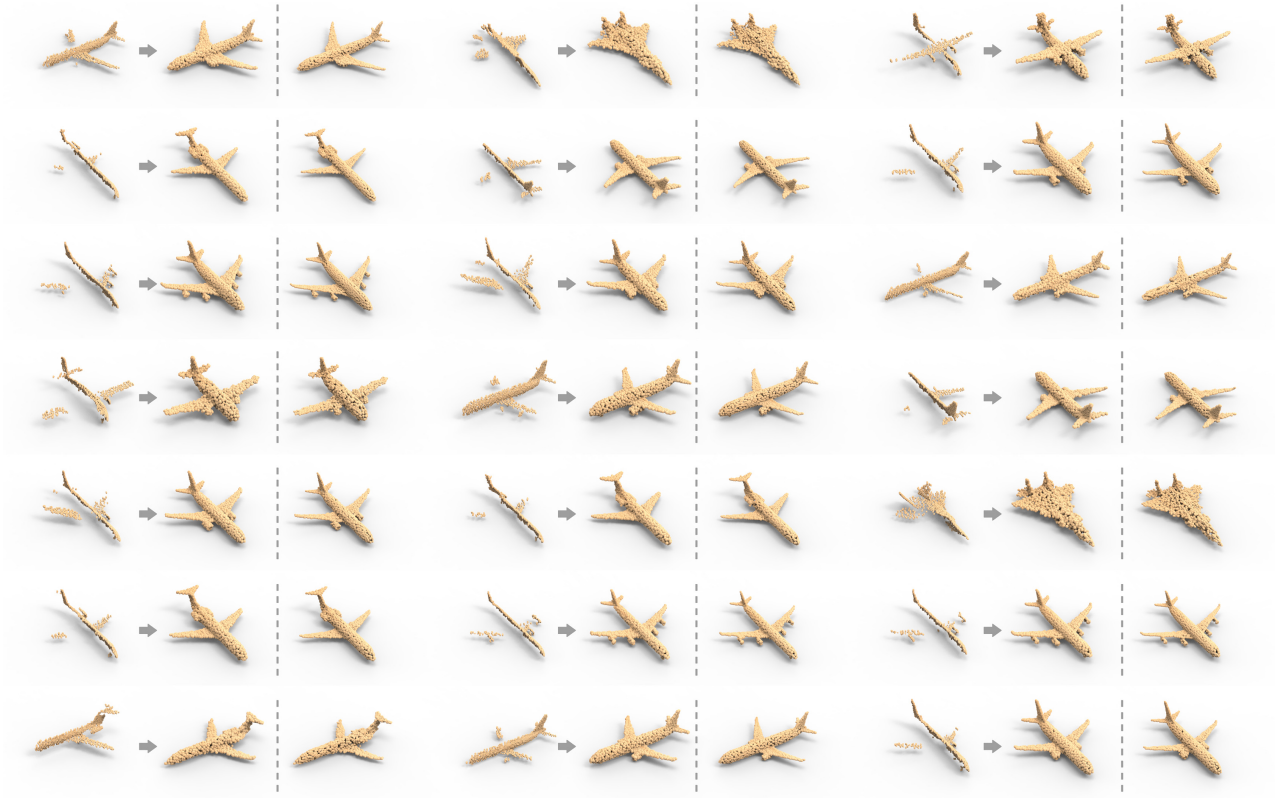

Figure 9: Results for *airplane scan* → *airplane*.

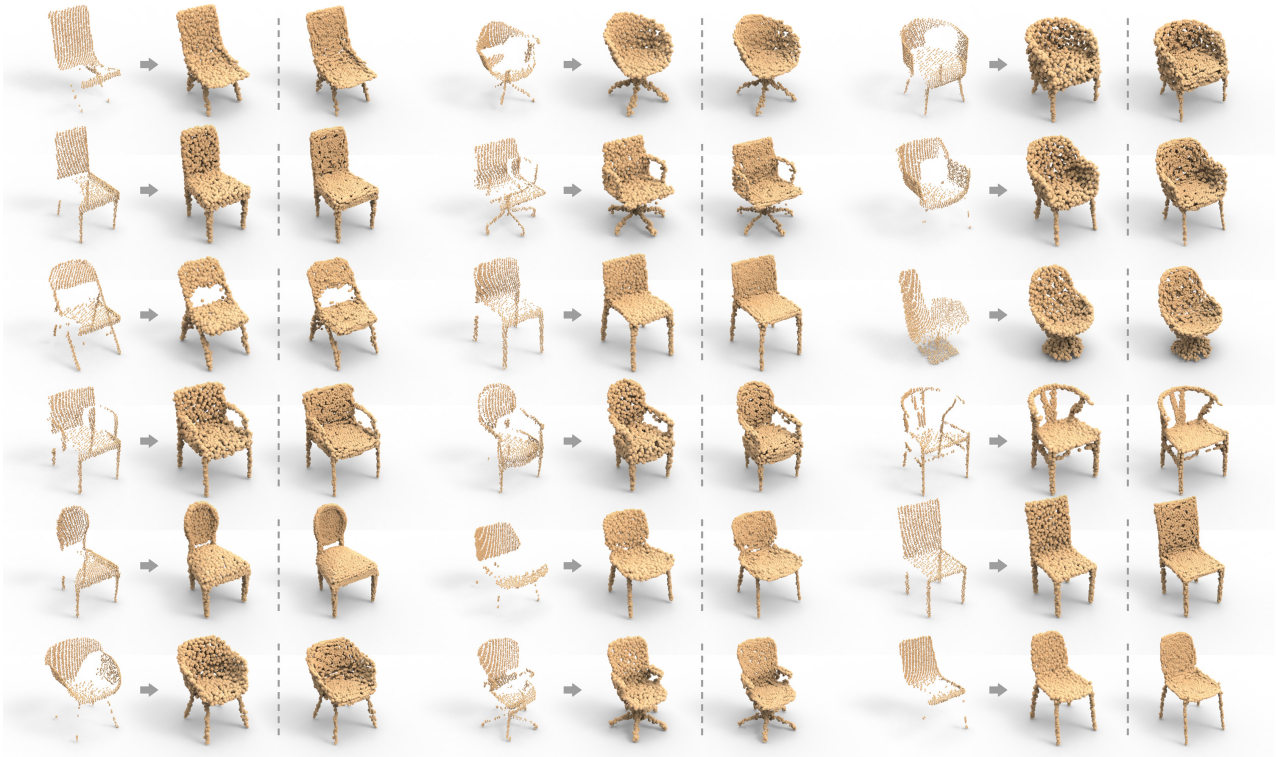

Figure 10: Results for *chair scan* → *chair*.

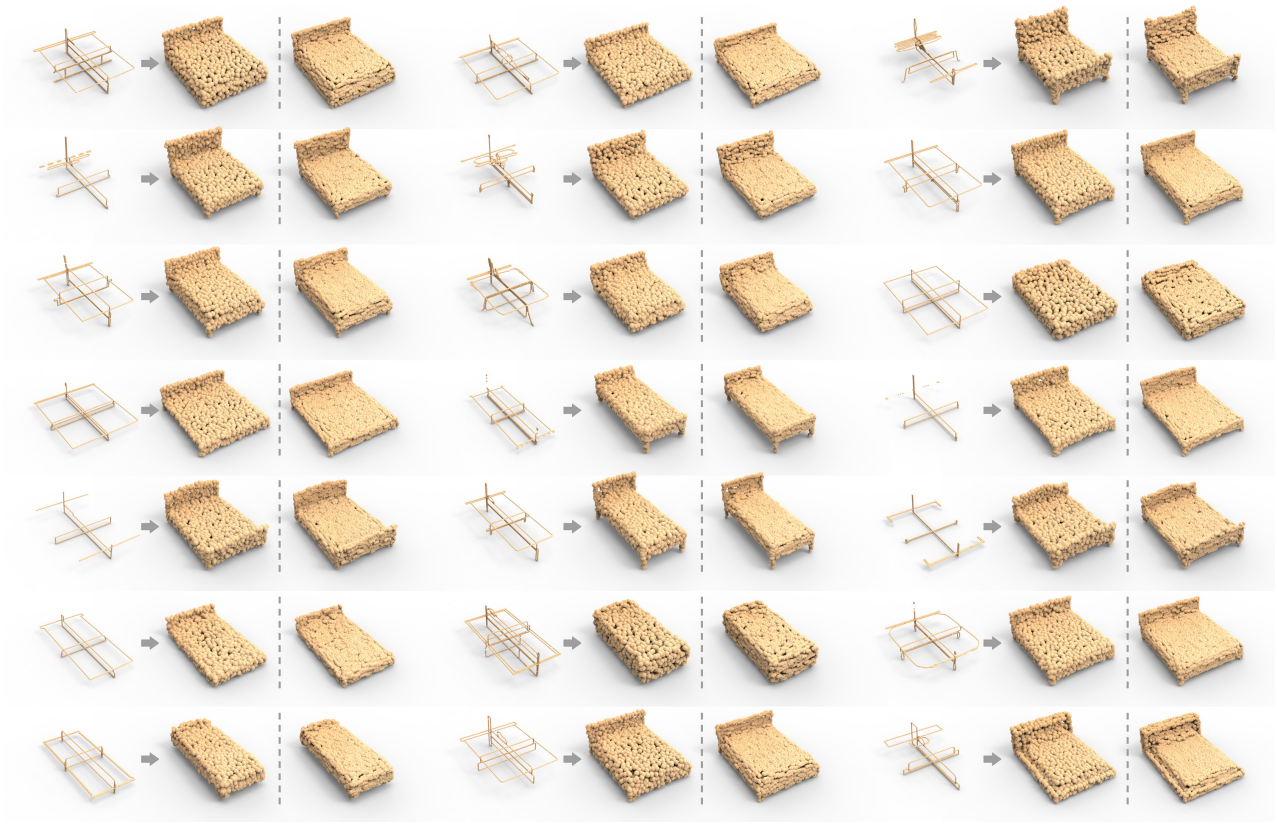

Figure 11: Results for *bed cross-sectional profiles* → *bed*.

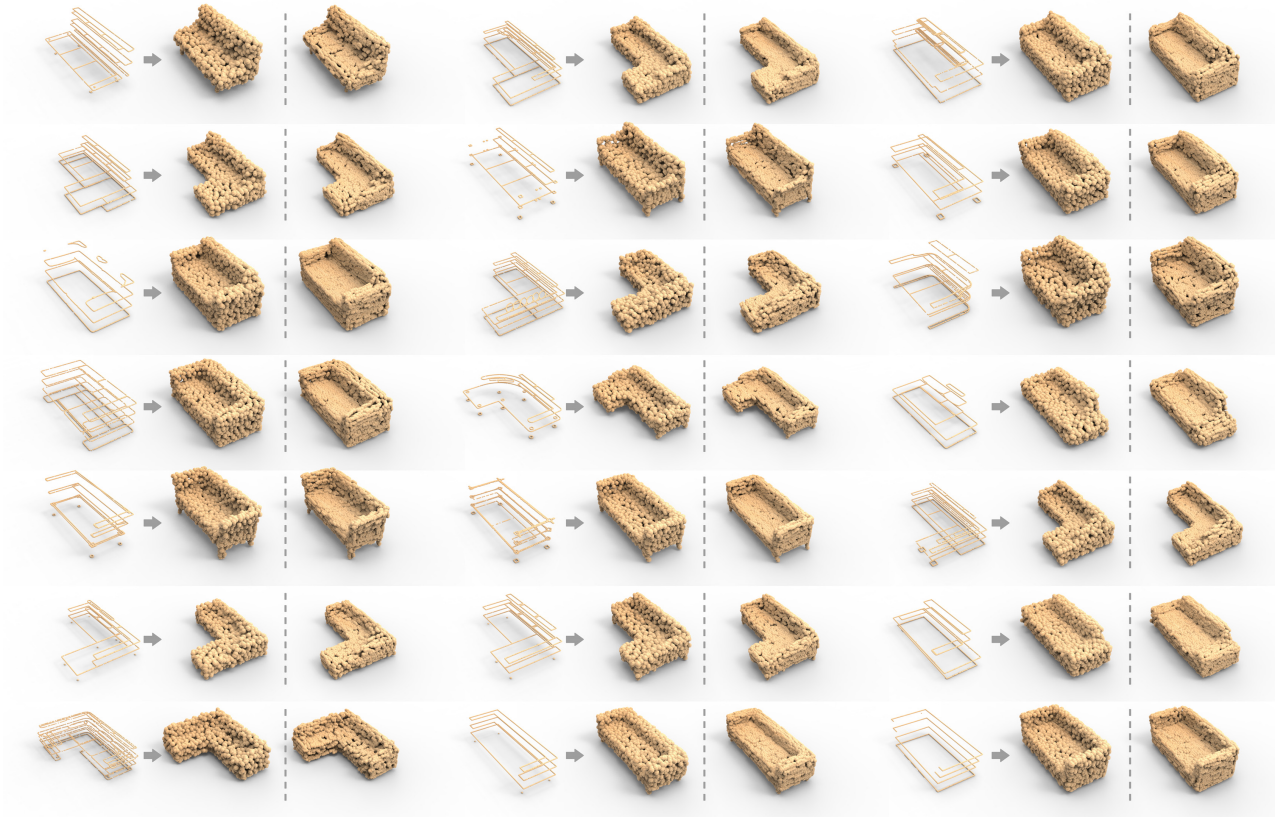

Figure 12: Results for *sofa cross-sectional profiles* → *sofa*.

Figure 13: Results of different methods for $A \leftrightarrow H$. Note that the input letters of the same row in Figure 13 ($A \leftrightarrow H$), Figure 14 ($G \leftrightarrow R$) and Figure 15 ($M \leftrightarrow N$) are from the same font.

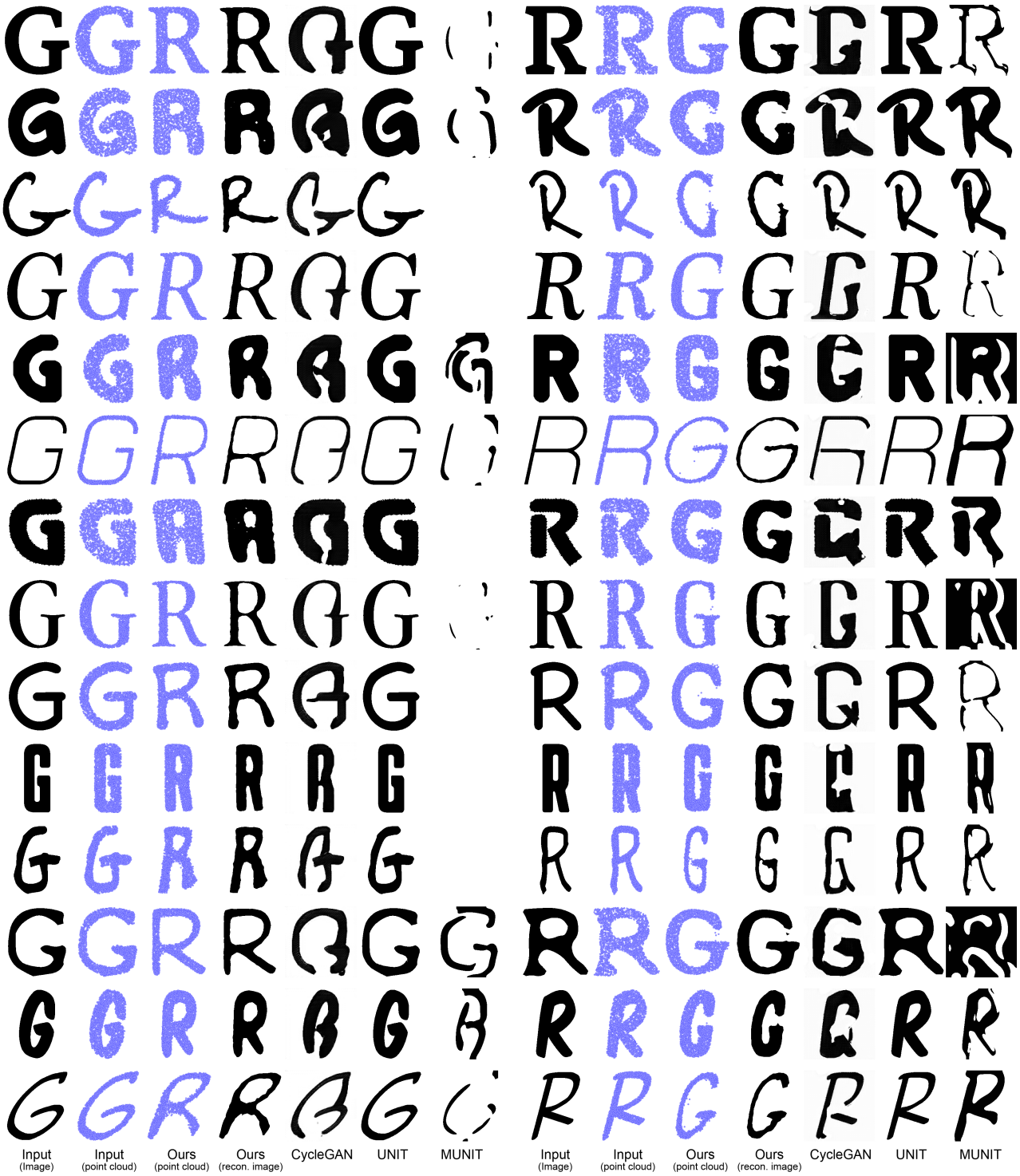

Figure 14: Results of different methods for $G \leftrightarrow R$. Note that the input letters of the same row in Figure 13 ($A \leftrightarrow H$), Figure 14 ($G \leftrightarrow R$) and Figure 15 ($M \leftrightarrow N$) are from the same font.

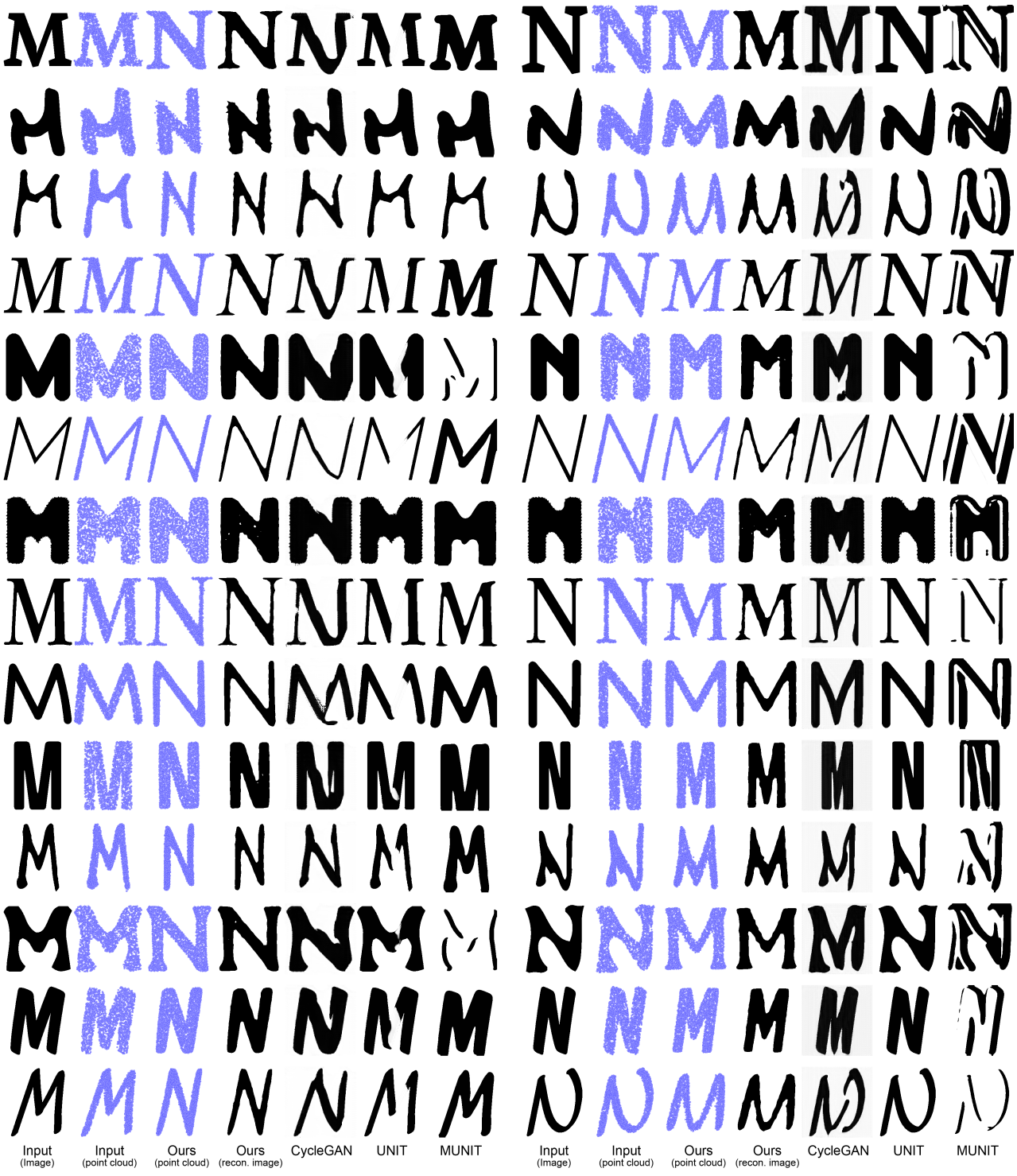

Figure 15: Results of different methods for $M \leftrightarrow N$. Note that the input letters of the same row in Figure 13 ($A \leftrightarrow H$), Figure 14 ($G \leftrightarrow R$) and Figure 15 ($M \leftrightarrow N$) are from the same font.

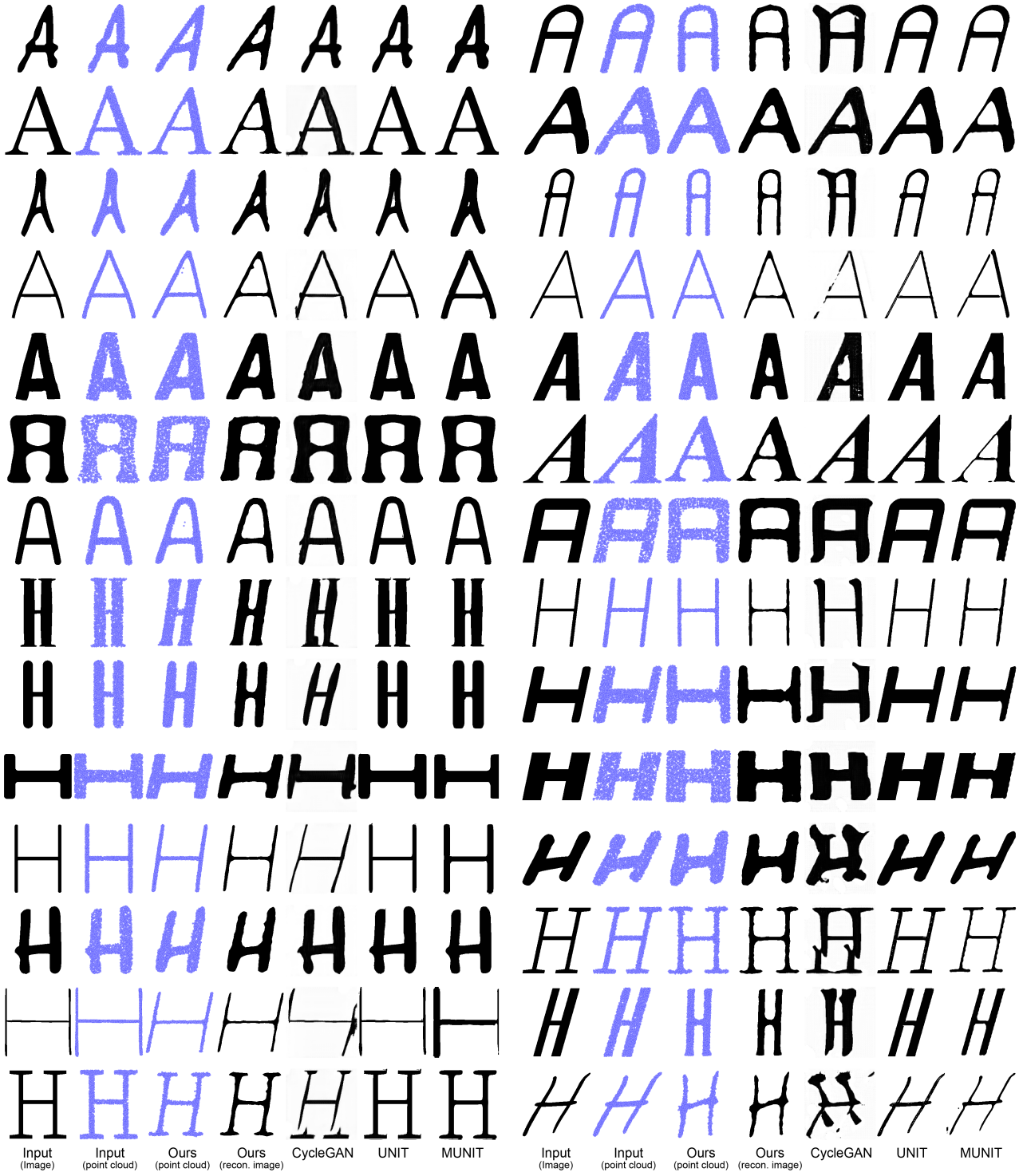

Figure 16: Results of different methods for *regularA/H* $\leftrightarrow$ *italicA/H*.

Figure 17: Results of different methods for *thinG/R* ↔ *thickG/R*.

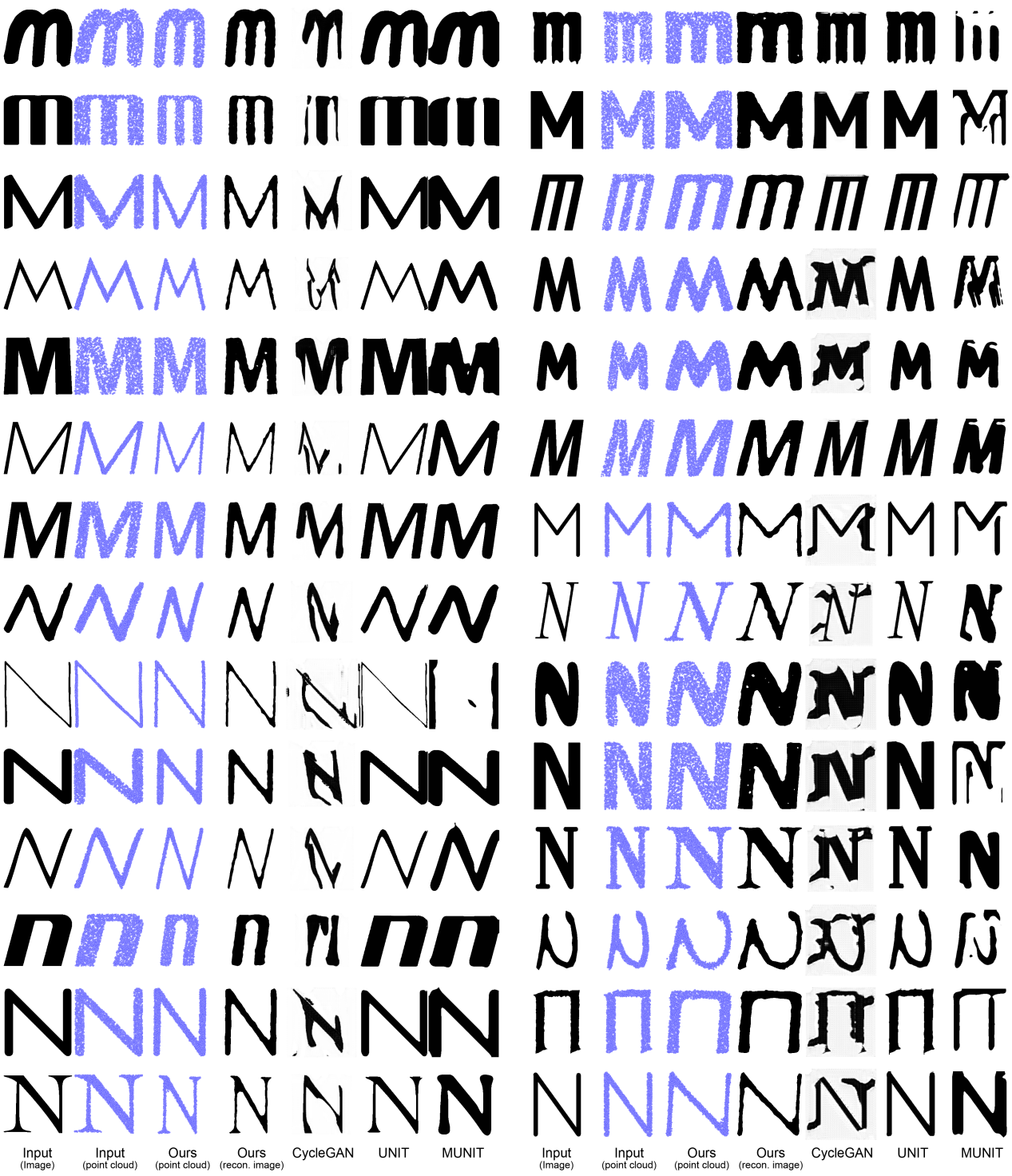

Figure 18: Results of different methods for *wideM/N* ↔ *narrowM/N*. Note that we align the letters by height for better visualization.

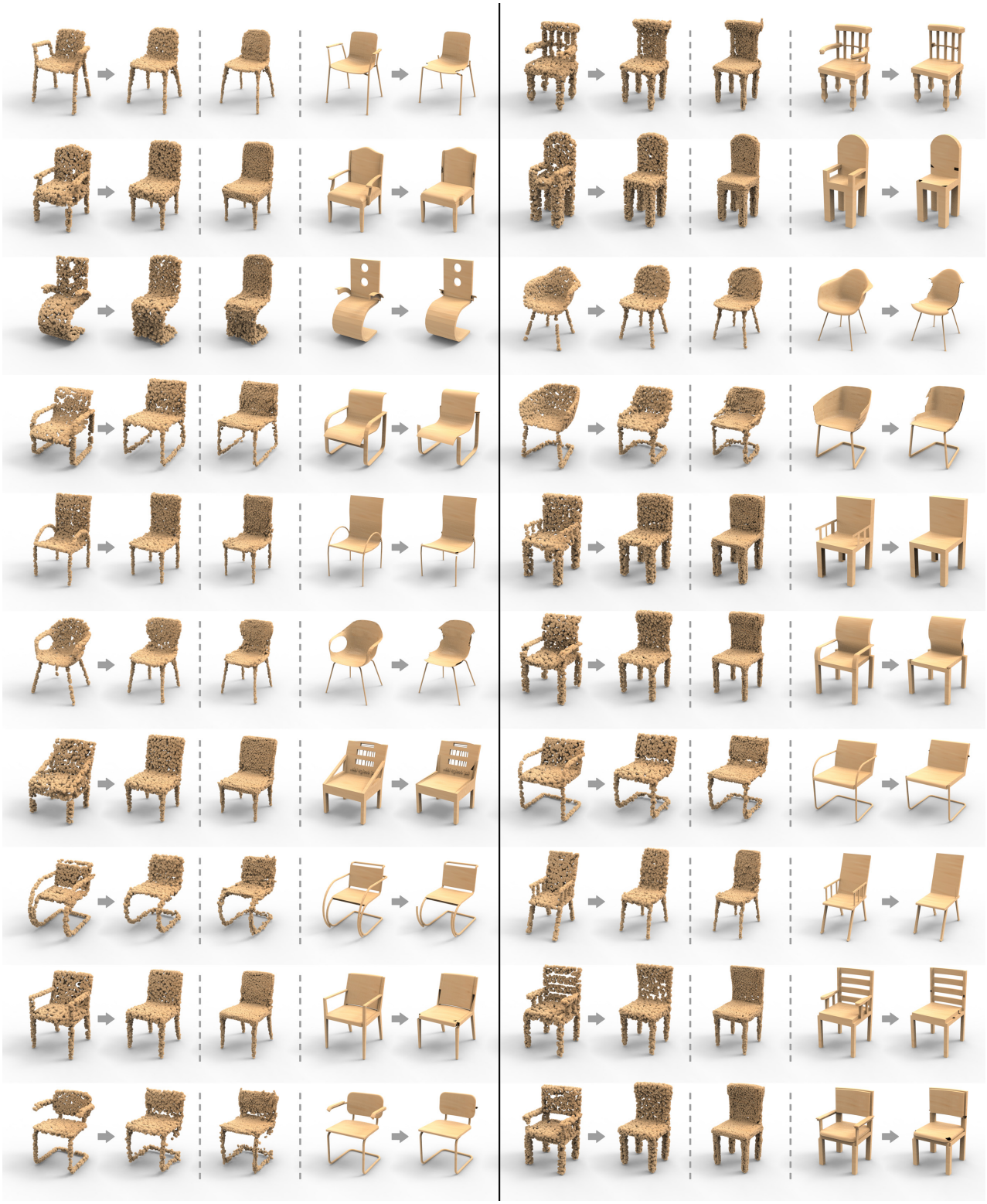

Figure 19: Results for *armchair* → *armless chair*. From left to right in each group (of 5 shapes): (1) input point cloud; (2) output point cloud (2048 points); (3) higher-resolution output point cloud (8192 points); (4) original mesh of input point cloud; (5) result of part removal guided by the transforms.

Figure 20: Results for *armless chair* → *armchair*. From left to right in each group (of 5 shapes): (1) input point cloud; (2) output point cloud (2048 points); (3) higher-resolution output point cloud (8192 points); (4) original mesh of input point cloud; (5) result of part addition guided by the transforms.