# StructEdit: Learning Structural Shape Variations Supplementary Material

Kaichun Mo[*1] Paul Guerrero[*2] Li Yi[3] Hao Su[4] Peter Wonka[5] Niloy J. Mitra[2,6] Leonidas Guibas[1,7]

[1]Stanford University [2]Adobe Research [3]Google Research [4]UC San Diego
[5]KAUST [6]University College London [7]Facebook AI Research

## A. More Dataset Details

We provide statistics of the PartNet [3, 2] dataset, as well as the synthetic dataset, and show a few samples from each. Additionally, we discuss the procedural shape generation pipeline that we use to create the synthetic dataset.

### A.1. Dataset Statistics

In our experiments, we use four datasets. A dataset of 4,871 chairs, 5,099 tables, and 862 cabinets in PartNet [3]. Additionally, we create a synthetic dataset of 57,600 chairs, stools, and sofas, where we have a ground-truth correspondence between the deltas in the neighborhoods of different source shapes. For each dataset, we use the same hierarchical bounding box representation. Figure 1 show examples of each dataset and more statistics are summarized in Table 1.

Table 1. **Dataset Statistics.** We show number of shapes, average tree depth, average leaf count for each dataset, and the neighborhood size (train time/test time), *i.e.* the number of shape deltas for each source shape.

| | #shapes | tree depth | #leafs | $\lvert\mathcal{N}\rvert$ |
|---|---|---|---|---|
| PartNet chair | 4871 | 4.039 | 11.097 | 100/20 |
| PartNet table | 5099 | 5.127 | 7.537 | 100/20 |
| PartNet furniture | 862 | 4.522 | 14.377 | 100/20 |
| Synthetic | 57600 | 3.667 | 10.111 | 96/96 |

### A.2. Synthetic Dataset Generation

In this section, we introduce the procedural generation pipeline of the synthetic dataset. In the procedural generation, we explicitly create shape deltas for each source shape. This gives us knowledge of the ground truth correspondences between the shape deltas of different source shapes. We use this ground truth to quantitatively evaluate our edit transfer performance.

A chair shape consists of four basic components: a back, a seat, an optional pair of arms and a leg base with possibly different types of stretcher bars connecting four legs. We randomly sample 8 global parameters for each shape: $(w_{leg}, h_{leg}, w_{seat}, d_{seat}, h_{seat}, h_{back}, w_{back}, d_{back})$ where $w_{leg}$ and $h_{leg}$ are leg width and height, $w_{seat}$, $d_{seat}$, and $h_{seat}$ are seat width, depth and height, and finally, $w_{back}$, $d_{back}$, and $h_{back}$ refer to back width, depth and height. All parameters for the other parts are deterministically derived based on the eight global parameters or assigned with fixed values. For example, chair arm depth is half of the seat depth and all stretcher bars have a fixed height of 0.03. Combinations of values for the 8 global parameters give us a large set of source shapes.

We create structural variations for each of these shapes by changing the structure of individual parts. For each shape, we create 4 variants for back (*e.g.* with or without back bars, with vertical bars or horizontal bars), 2 variants for legs


Figure 1. **Dataset Examples.** We show examples from each of the four datasets we use in our experiments. Chairs, tables and furniture are from the PartNet dataset, while the synthetic dataset is procedurally generated. Colors correspond to part semantics.

[*]: indicates equal contributions.

(*e.g.* short or long), 3 variants for arms (*e.g.* with or without arms, different layouts for armrest and arm support), and 4 variants for leg stretchers (*e.g.* squared layout, H-like layout, or X-like layout). In total, we make $4 \times 2 \times 3 \times 4 = 96$ structural variants for the same shape. A few examples of these variants are shown in Figure 2. We normalize all generated shapes within a unit sphere. In this procedural dataset, corresponding variants of different source shapes have the same index. Thus, for two variants with index $i$ and $j$ of two shapes $A, B$: $(A_i, A_j)$ and $(B_i, B_j)$, we can define a ground-truth for the edit transfer as $(A_j - A_i) + B_i = B_j$.

In real chairs, we do not always have correspondences between all possible shape variations. For example, a delta that makes the legs shorter does not have a correspondence in a sofa that does not have legs. To model these differences in the delta neighborhoods of our synthetic chairs, we divide them into three sub-types: 19,200 chairs, 19,200 sofas, and 19,200 stools. The creation of sofa shapes and stool shapes follow the same procedural generative grammar as chair shapes, except that we remove the leg base for sofas and remove chair back and arms for stools. For each of these sub-types, the dataset comprises of 200 groups of shapes, each with 96 structural variations. Between two sub-types, a known subset of the deltas does not have a correspondence. For example, deltas that modify the legs of chairs do not have a correspondence in sofas. We manually set the correspondence of these deltas to the identity edit (i.e. the delta that does not change the shape).

We have released the code for procedural shape generation pipeline and the generated synthetic dataset.

## B. Network Architecture Details

In our architecture, individual encoders and decoders share a similar architecture, unless noted otherwise in the main paper. In our experiments, we found that the total number of layers in the encoders and decoders has a significant adverse effect on the performance of the cVAE, especially since the recursive traversal depends on the depth of the shape tree. To keep the number of layers low, we use a relatively simple architecture for all individual encoders and decoders (unless noted otherwise): a multi layer perceptron (MLP) that has two layers. We also add a skip connection [1] that starts at the input and is added to the output to shorten the information path. This simple architecture is illustrated in Figure 3.

Figure 2. **Synthetic chair variations.** We show a few examples of the 96 structural variations of a bench, including variations to the legs, base, backrest, and armrests.

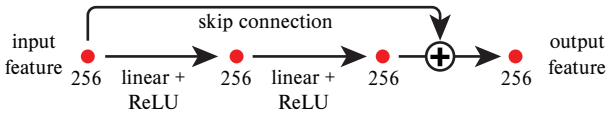


Figure 3. **Typical encoder/decoder architecture.** Unless noted otherwise we use this type of architecture for our individual encoders and decoders. The red dots are feature vectors in $\mathbb{R}^{256}$, arrows correspond to operations.

## C. Additional Experiments

We provide an ablation study for the network architecture design choices and more experimental results with comparisons to our baselines. Note that methods that only encode geometry and not structure, such as methods that represent objects as voxels, point clouds, or implicit functions, are not suitable baselines for our method. The domain we work on consists of both geometry and structure, where structure is an abstraction of geometry. Methods that work on geometry only have a fundamentally different goal than our method. Their outputs, being geometry only, cannot be compared fairly to our output that combines geometry and structure. For this reason we only compare to methods that work on the same domain as ours in our experiments.

### C.1. Ablation Study

We perform an ablation of four design choices in our architecture: our extensive use of skip connections, using group normalization, using a separate classifier to determine of added nodes are leafs, and encoding deltas of box parameters, instead of modified boxes. For each ablation, we

Table 2. **Ablation.** We compare four ablated versions of our method to the full version in the last row. Each row shows the geometric and structural reconstruction error for both geometric and structural neighborhoods, as described in Section 4 of the paper.

| | $\mathcal{N}^g$ | | $\mathcal{N}^s$ | |
|---|---|---|---|---|
| | $E_r^{\text{geo}}$ | $E_r^{\text{st}}$ | $E_r^{\text{geo}}$ | $E_r^{\text{st}}$ |
| No Skip Conn. | 0.900 | 0.201 | 0.713 | 0.364 |
| No Group Norm. | 0.749 | 0.083 | 0.525 | 0.142 |
| No Leaf Class. | 0.759 | 0.087 | 0.533 | 0.171 |
| No Box Deltas. | 1.737 | 0.083 | 1.766 | 0.142 |
| Full | 0.754 | 0.082 | 0.531 | 0.136 |

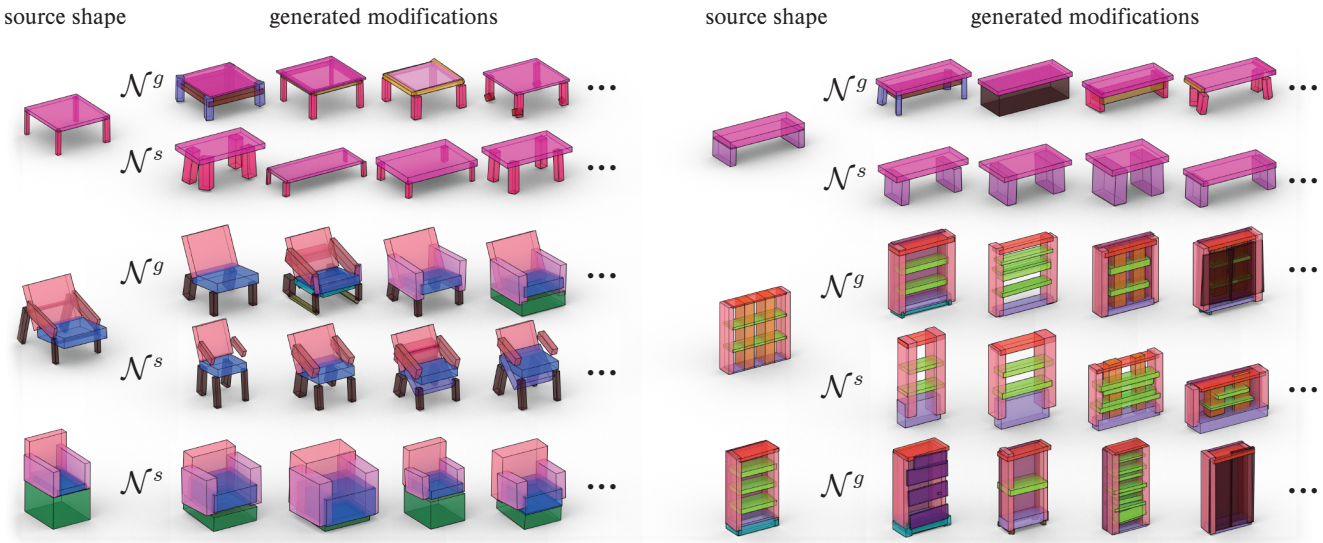


Figure 4. **Edit Generation Examples.** Several examples of variations are generated for the source shapes on the left. We show variations generated with both geometric neighborhoods $\mathcal{N}^g$ and structural neighborhoods $\mathcal{N}^s$.

Table 3. **Edit Reconstruction.** We compare the geometric and structural reconstruction errors for the identity baseline (ID), StructureNet (SN), StructureNet-edge (SNe), and StructEdit (SE) on both geometric neighborhoods $\mathcal{N}^g$ and structural neighborhoods $\mathcal{N}^s$.

| | | $\mathcal{N}^g$ | | | | $\mathcal{N}^s$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | chair | table | furn. | avg. | chair | table | furn. | avg. |
| $E_r^{\text{geo}}$ | ID | 1.000 | 1.000 | 1.001 | 1.000 | 1.000 | 1.000 | 0.999 | 1.000 |
| | SN | 0.886 | 0.972 | 0.875 | 0.911 | 0.656 | 0.492 | 0.509 | 0.553 |
| | SNe | 0.893 | 1.118 | 0.895 | 0.969 | 0.658 | 0.575 | 0.517 | 0.583 |
| | SE | **0.755** | **0.805** | **0.798** | **0.786** | **0.531** | **0.414** | **0.434** | **0.459** |
| $E_r^{\text{st}}$ | ID | 0.946 | 0.940 | 0.951 | 0.945 | 1.107 | 1.341 | 1.124 | 1.191 |
| | SN | 0.264 | 0.370 | 0.388 | 0.340 | 0.734 | 1.469 | 0.915 | 1.039 |
| | SNe | 0.273 | 0.380 | 0.415 | 0.356 | 0.711 | 1.475 | 0.843 | 1.010 |
| | SE | **0.082** | **0.151** | **0.139** | **0.124** | **0.136** | **0.246** | **0.183** | **0.188** |

evaluate the reconstruction and generation performance on the chairs dataset. From these four design choices, the skip connections have the largest positive impact on the structure of shape deltas, while encoding box deltas instead of absolute boxes has the largest positive effect on the geometry. Table 2 shows the performance for each ablated variant of our method.

## C.2. Edit Reconstruction

We add a version of StructureNet with relationship edges (SNe) as additional baseline to the edit reconstruction results in Table 3. The StructureNet version with edges tends to have a slightly lower performance, since the relationship edges require reconstructing additional information.

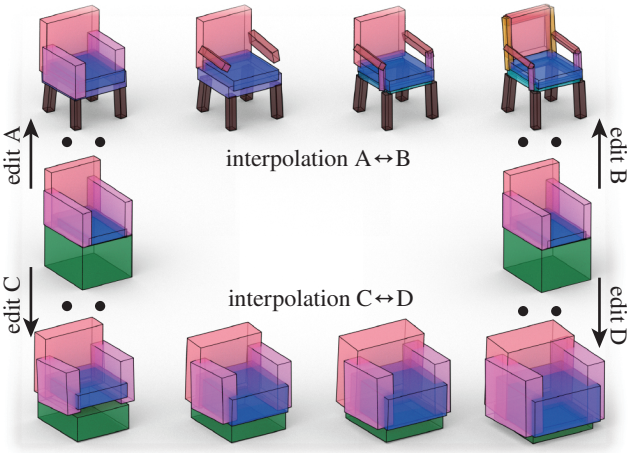


Figure 5. **Edit interpolation.** Edit A and B on the left are interpolated with edit B and D on the right. Intermediate steps of the interpolated edits are applied to the source shape (middle row) to get the interpolated shapes in the top and bottom rows. Note how changes in both geometry and structure are interpolated smoothly.

## C.3. Edit Generation

Full edit generation metrics, including separate quality and coverage errors and the additional StructureNet baseline that includes relationship edges, are given in Table 4. We also show more qualitative results in Figure 4.

## C.4. Edit Interpolation

Our latent space of edits has all the benefits that are enabled by a smooth latent space, such as the ability to interpolate between two edits. In Figure 5, we show two examples

Table 4. **Edit Generation.** We compare the delta distribution generated by our method to several baselines. We evaluate on three PartNet subsets, using both geometric and structural neighborhoods. The quality, coverage and aggregated errors are shown, using both geometric and structural distances. Our method benefits from the consistency of delta distributions, resulting in an improved performance.

| $\mathcal{N}^g$ | | chair | table | furniture | avg. |
|---|---|---|---|---|---|
| $E_{\mathrm{q}}^{\mathrm{geo}}$ / $E_{\mathrm{c}}^{\mathrm{geo}}$ / $E_{\mathrm{qc}}^{\mathrm{geo}}$ | Identity | 0.846 / 0.976 / 1.822 | **0.789** / 0.974 / 1.763 | **0.704** / 0.980 / 1.684 | **0.780** / 0.977 / 1.756 |
| | StructureNet-0.2 | 0.826 / 0.934 / 1.760 | 1.008 / 1.068 / 2.076 | 0.735 / 0.891 / 1.626 | 0.856 / 0.964 / 1.821 |
| | StructureNetEdge-0.2 | 0.838 / 0.942 / 1.781 | 1.127 / 1.197 / 2.324 | 0.766 / 0.932 / 1.698 | 0.910 / 1.024 / 1.934 |
| | StructureNet-0.5 | 0.857 / 0.865 / 1.722 | 1.092 / 0.975 / 2.068 | 0.744 / 0.815 / 1.558 | 0.898 / 0.885 / 1.783 |
| | StructureNetEdge-0.5 | 0.863 / 0.874 / 1.737 | 1.127 / 1.052 / 2.178 | 0.775 / 0.869 / 1.644 | 0.922 / 0.931 / 1.853 |
| | StructureNet-1.0 | 0.940 / 0.828 / 1.768 | 1.270 / 0.918 / 2.189 | 0.789 / 0.765 / **1.554** | 1.000 / 0.837 / 1.837 |
| | StructureNetEdge-1.0 | 0.933 / 0.832 / 1.765 | 1.168 / 0.987 / 2.156 | 0.816 / 0.810 / 1.626 | 0.973 / 0.877 / 1.849 |
| | StructEdit (Ours) | **0.789** / **0.804** / **1.593** | 0.834 / **0.821** / **1.655** | 0.806 / **0.755** / 1.561 | 0.810 / **0.793** / **1.603** |
| $E_{\mathrm{q}}^{\mathrm{st}}$ / $E_{\mathrm{c}}^{\mathrm{st}}$ / $E_{\mathrm{qc}}^{\mathrm{st}}$ | Identity | **0.281** / 1.000 / 1.281 | **0.215** / 1.000 / 1.215 | **0.288** / 1.000 / 1.288 | **0.261** / 1.000 / 1.261 |
| | StructureNet-0.2 | 0.300 / 0.781 / 1.081 | 0.248 / 0.630 / 0.878 | 0.316 / 0.698 / 1.015 | 0.288 / 0.703 / 0.991 |
| | StructureNetEdge-0.2 | 0.318 / 0.789 / 1.108 | 0.352 / 0.763 / 1.115 | 0.313 / 0.667 / 0.980 | 0.328 / 0.740 / 1.068 |
| | StructureNet-0.5 | 0.324 / 0.547 / 0.871 | 0.284 / 0.445 / 0.729 | 0.314 / 0.559 / 0.873 | 0.307 / 0.517 / 0.824 |
| | StructureNetEdge-0.5 | 0.343 / 0.555 / 0.898 | 0.332 / 0.585 / 0.918 | 0.313 / 0.516 / 0.829 | 0.330 / 0.552 / 0.882 |
| | StructureNet-1.0 | 0.388 / 0.363 / 0.751 | 0.347 / 0.321 / 0.667 | 0.336 / 0.390 / 0.726 | 0.357 / 0.358 / 0.715 |
| | StructureNetEdge-1.0 | 0.408 / 0.372 / 0.780 | 0.348 / 0.405 / 0.753 | 0.335 / 0.382 / **0.717** | 0.364 / 0.386 / 0.750 |
| | StructEdit (Ours) | 0.299 / **0.259** / **0.559** | 0.299 / **0.225** / **0.524** | 0.518 / **0.223** / 0.741 | 0.372 / **0.236** / **0.608** |
| $\mathcal{N}^s$ | | chair | table | furniture | avg. |
| $E_{\mathrm{q}}^{\mathrm{geo}}$ / $E_{\mathrm{c}}^{\mathrm{geo}}$ / $E_{\mathrm{qc}}^{\mathrm{geo}}$ | Identity | 0.651 / 0.978 / 1.629 | **0.499** / 0.980 / 1.479 | 0.467 / 0.980 / 1.446 | 0.539 / 0.979 / 1.518 |
| | StructureNet-0.2 | **0.557** / 0.751 / 1.308 | 0.501 / 0.707 / 1.208 | **0.450** / 0.793 / 1.243 | **0.502** / 0.750 / 1.253 |
| | StructureNetEdge-0.2 | 0.564 / 0.754 / 1.318 | 0.549 / 0.782 / 1.331 | 0.458 / 0.772 / 1.231 | 0.524 / 0.769 / 1.293 |
| | StructureNet-0.5 | 0.571 / 0.670 / 1.241 | 0.516 / 0.587 / 1.103 | 0.451 / 0.684 / 1.135 | 0.513 / 0.647 / 1.160 |
| | StructureNetEdge-0.5 | 0.575 / 0.675 / 1.250 | 0.549 / 0.660 / 1.209 | 0.458 / 0.674 / 1.132 | 0.527 / 0.670 / 1.197 |
| | StructureNet-1.0 | 0.611 / **0.621** / 1.232 | 0.548 / 0.509 / 1.057 | 0.456 / 0.561 / 1.017 | 0.538 / 0.564 / 1.102 |
| | StructureNetEdge-1.0 | 0.606 / 0.624 / 1.230 | 0.551 / 0.572 / 1.123 | 0.463 / 0.570 / 1.032 | 0.540 / 0.589 / 1.129 |
| | StructEdit (Ours) | 0.581 / 0.637 / **1.218** | 0.501 / **0.499** / **1.000** | 0.521 / **0.494** / **1.015** | 0.534 / **0.543** / **1.078** |
| $E_{\mathrm{q}}^{\mathrm{st}}$ / $E_{\mathrm{c}}^{\mathrm{st}}$ / $E_{\mathrm{qc}}^{\mathrm{st}}$ | Identity | 0.437 / 1.000 / 1.437 | 0.303 / 1.000 / 1.303 | **0.442** / 1.000 / 1.442 | 0.394 / 1.000 / 1.394 |
| | StructureNet-0.2 | 0.693 / 0.773 / 1.466 | 2.218 / 1.267 / 3.484 | 0.598 / 0.816 / 1.414 | 1.169 / 0.952 / 2.121 |
| | StructureNetEdge-0.2 | 0.805 / 0.845 / 1.650 | 2.944 / 1.792 / 4.736 | 0.653 / 0.853 / 1.506 | 1.467 / 1.163 / 2.631 |
| | StructureNet-0.5 | 0.888 / 0.485 / 1.373 | 2.518 / 0.781 / 3.300 | 0.613 / 0.590 / 1.204 | 1.340 / 0.619 / 1.959 |
| | StructureNetEdge-0.5 | 0.993 / 0.540 / 1.534 | 2.793 / 1.208 / 4.002 | 0.692 / 0.627 / 1.319 | 1.493 / 0.792 / 2.285 |
| | StructureNet-1.0 | 1.413 / 0.350 / 1.763 | 3.099 / 0.523 / 3.622 | 0.750 / 0.417 / 1.167 | 1.754 / 0.430 / 2.184 |
| | StructureNetEdge-1.0 | 1.516 / 0.392 / 1.908 | 3.047 / 0.792 / 3.839 | 0.821 / 0.478 / 1.298 | 1.794 / 0.554 / 2.348 |
| | StructEdit (Ours) | **0.323** / **0.286** / **0.609** | **0.271** / **0.180** / **0.451** | 0.454 / **0.222** / **0.676** | **0.349** / **0.229** / **0.579** |

of interpolations between different edits. The examples show that both geometric and structural changes are interpolated smoothly.

### C.5. Edit Transfer on the Synthetic Dataset

Qualitative results comparing StructEdit to StructureNet for edit transfer on the synthetic dataset are shown in Figure 6. The edit that transforms source shape A into the modified shape (first two columns) is transferred to source shape B (third column). On the synthetic dataset, we have a ground truth for the result of the edit transfer, shown in the fourth column. StructEdit (SE, last column) explicitly encodes edits, and can thus benefit from the large degree of consistency between the neighborhoods of deltas around different source shapes, giving us a significantly more accurate edit transfer than than StructureNet (SN). Note that we do not use the ground truth transferred edit during training. We do not use any kind of supervision for the mapping between the shape deltas of different source shapes. Our intuition is that the increased accuracy of the edit transfer is a result of tendency of networks to compress information in their latent space. Due to the consistency of the shape deltas around different source shapes in our datasets, a consistent layout of shape deltas in the latent spaces around different source shapes is the layout that uses the least amount of information. A similar effect is observed in several other unsupervised methods [5, 4].

## D. Implementation Details

In the following, we give additional details for the StructureNet baseline, and for two applications shown in the main paper: editing raw point clouds and cross-modal analogies.

### D.1. StructureNet Baseline

StructureNet learns a latent space of shapes (as opposed to shape edits), where we use simple arithmetics to transfer edits. We compare the result of our edit transfer: $S'_l = S_k + d(S_k, e(S_i, \Delta S_{ij}))$ to the following edited shape in the StructureNet baseline:

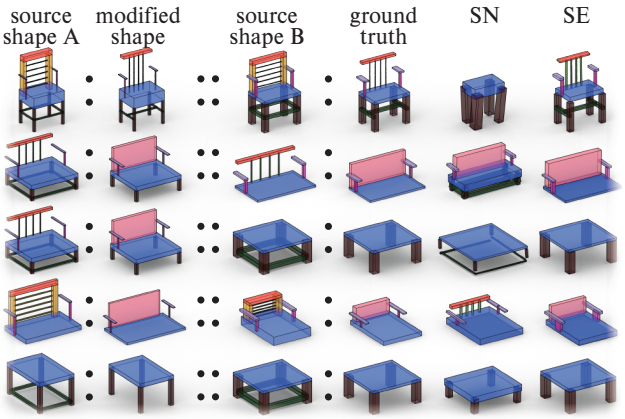


Figure 6. **Edit Transfer on the synthetic dataset.** Edits from source shape A are transferred to source shape B. Our edits (SE) more faithfully recover the ground truth modified shape.

$$S_l^{SN} = d^{SN}\Big(e^{SN}(S_k) + \big(e^{SN}(S_j) - e^{SN}(S_i)\big)\Big), \quad (1)$$

where in both ours and the StructureNet baseline, the edit is transferred from source shape $S_i$ to source shape $S_k$, resulting in shape $S_l$. Here, $e^{SN}$ and $d^{SN}$ denote the StructureNet encoder and decoder. Unlike our method, the difference vector in latent space $e^{SN}(S_j) - e^{SN}(S_i)$ is not encouraged to represent a semantically similar edit when applied to different source shapes, which results in the lower performance we can see in our experiments (Figures 3, 4 and 5 of the main paper). Alternative data-driven edit transfer methods typically use the same latent space arithmetics, but cannot handle structure.

### D.2. Generating Edits of Raw Point Clouds

In this application, we transform the point cloud into a structured shape using an existing method, find variations for the structured shape, and then transfer the corresponding shape deltas back to modify the point cloud. For the transformation into a shape, we first perform panoptic segmentation using the method described in PartNet [3], giving us part semantic and instance labels for each point. The semantics allow us to create a hierarchy among the part instances, and the instance labels give us part bounding boxes. After an edit, point cloud segments can either be transformed with the bounding box modifications or deleted, depending on the modification of the corresponding part. Added parts are transformed back into a point cloud by sampling their surface with a fixed number of points.

### D.3. Cross-modal Analogies

To transfer an edit defined by a pair of images to a point cloud, both images are transformed into structured shapes using the method described in StructureNet [2]: an encoder maps the images into the latent space of a pre-trained StructureNet. Once we have structured shapes for both images, we use their difference as shape delta. This delta is then transferred to the shape obtained from the point cloud using our learned latent space. The conversion between point clouds and shapes is handled as described in the previous application.

## References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[2] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *ACM Transactions on Graphics (TOG), Siggraph Asia 2019*, 38(6):Article 242, 2019. 1, 5

[3] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2019. 1, 5

[4] Chun-Yu Sun, Qian-Fang Zou, Xin Tong, and Yang Liu. Learning adaptive hierarchical cuboid abstractions of 3d shape collections. *ACM Trans. Graph.*, 38(6):241:1–241:13, Nov. 2019. 4

[5] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 4