ShapeWalk: Compositional Shape Editing through Language-Guided Chains

Supplementary Material

Code and Dataset

Our dataset, generation code, model checkpoints and training scripts will be made available upon publication at:

`https://shapewalk.github.io/`

Appendix Overview

This supplementary document is organized as follows:

- Section 1 provides additional analysis of our latent editor models.
- Section 2 provides additional implementation details about our dataset generation and model training processes.
- Section 3 gives shape reconstruction examples.
- Section 4 showcases edit chain samples from our dataset, including edit chains for additional **table** and **vase** classes, and dataset statistics.
- Section 5 illustrates the sampling process for the 3D2VS latent diffusion model.

1. Analysis

We differentiate in the additional results in this section between $\text{CD} - \text{REC}$ and $\text{CD} - \text{REAL}$ metrics, which respectively measure the distance to the reconstructed edited shape and the real edited shape. The latter metric is more relevant for our task, as it measures the final editing ability of the model (reported in the main paper).

1.1. Chained Shape Editing

We provide detailed results for the chained shape editing task described in the main paper, in Table 2. We report baseline results for our proposed chained shape editing task, for chain lengths $|\mathcal{P}| \in \{10, 15, 20\}$. Overall, while no clear trend is observed for the Chamfer Distance, we observe that the average edit error for $\mathcal{L}_2$ distance decreases with longer chains. This could be explained by the fact that longer chains contain more sequences of fine-grained edits which are less likely to make the feature representations diverge abruptly. On the other hand, shorter chains are more likely to contain sequences of high-magnitude edits, which amplify the cumulative prediction error. In that sense, a parallel could be made with the long-horizon prediction problem in trajectory forecasting, where the prediction error accumulates over time [2].

1.2. Oracle Editing

We provide detailed results for the chained shape editing task when oracle magnitudes and directions are used, in Table 2. Overall, the biggest gain for PC-AE based models is observed when using oracle directions. This suggests that the main bottleneck for our models is the edit direction prediction task, which is more challenging than predicting the magnitude from the input prompt.

1.3. Predicting Edit Magnitudes

Our decoupled latent editor models predict both a normalized edit direction $\hat{v}_{ij}$ and an edit magnitude $\hat{m}_{ij}$ for each edit. We use this property to analyze the correlation between the predicted edit magnitudes and the ground truth edit intensities extracted from our metadata. We extract decoupled edit vector predictions from our models on the *random* subset, and compare them to the ground truth edit intensities. In order to do that, we min-max normalize the predicted edit magnitudes and bin them in the $[1, 9]$ interval. Ground-truth edit intensities are extracted from the ground-truth edit vectors, similarly by discretizing the magnitude of changes in the same interval. We plot confusion matrices comparing these predicted discretized edit magnitudes to the ground truth edit intensities in Figure 1. We show results for three decoupled baselines: LATEFUSION_{1024}, LATEFUSION_{512}, and $\text{OURS}_{512\times4}$.

Overall, we observe a strong correlation between the predicted edit magnitudes and the ground truth edit intensities across all models. However all models tend to deviate from the ideal diagonal magnitude predictions. We remark also that the $\text{OURS}_{512\times4}$ model tends to deviate from the diagonal and avoids predicting the maximum possible magnitude range. One way to alleviate these biases could be to introduce explicit supervision to the edit magnitude prediction task using the ground truth edit magnitude labels. For example, a contrastive loss could incentivize the model to predict edit magnitudes that align with the ground truth ranking of edit intensities.

2. Implementation Details

2.1. Shape Chain Generation

We detail in Figure 9 the process of sampling parameter chains from the parameter tree. Each parameter vector θ_i can be represented as a dependency tree. We start by sampling a parameter tree θ_0 using regressed parameters from a real **source** shape in the *realistic* subset. We then successively sample parameters to edit by interpolating towards a **target** shape in the *realistic* setting[1].

[1] In the *random* setting, the source shape and subsequent parameters are sampled randomly.

Model	decoupled?	$\|\mathcal{P}\|$ = 10						$\|\mathcal{P}\|$ = 15						$\|\mathcal{P}\|$ = 20					
		F$_{\text{CD}-\text{Rec}}$	A$_{\text{CD}-\text{Rec}}$	F$_{\text{CD}-\text{Real}}$	A$_{\text{CD}-\text{Real}}$	F$_{\mathcal{L}_2}$	A$_{\mathcal{L}_2}$	F$_{\text{CD}-\text{Rec}}$	A$_{\text{CD}-\text{Rec}}$	F$_{\text{CD}-\text{Real}}$	A$_{\text{CD}-\text{Real}}$	F$_{\mathcal{L}_2}$	A$_{\mathcal{L}_2}$	F$_{\text{CD}-\text{Rec}}$	A$_{\text{CD}-\text{Rec}}$	F$_{\text{CD}-\text{Real}}$	A$_{\text{CD}-\text{Real}}$	F$_{\mathcal{L}_2}$	A$_{\mathcal{L}_2}$
LateFusion$_{1024}$	✗	1.751	1.431	3.006	2.756	1.913	1.707	2.150	1.579	2.941	2.925	1.366	1.241	1.591	0.944	2.620	2.182	1.054	**0.805**
LateFusion$_{512}$	✗	1.802	1.413	3.031	2.801	2.018	1.770	1.804	1.469	2.824	2.755	1.309	1.221	1.414	1.005	2.301	2.272	1.059	0.880
LateFusion$_{256}$	✗	1.807	1.486	3.026	2.928	2.079	1.810	2.034	1.380	2.852	2.718	1.342	1.203	1.737	1.018	2.743	2.306	1.106	0.838
Ours$_{512\times8}$	✗	2.240	1.665	3.680	3.169	2.311	2.022	2.471	1.664	3.347	2.956	1.591	1.351	1.571	0.992	2.599	2.341	1.207	0.937
Ours$_{512\times4}$	✗	1.751	1.494	3.236	2.859	2.168	1.884	2.378	1.540	3.098	2.850	1.432	1.271	1.659	1.087	2.636	2.401	1.166	0.898
LateFusion$_{1024}$	✓	1.773	1.452	3.066	2.794	2.002	1.734	1.704	1.290	2.787	2.667	**1.212**	**1.184**	1.502	1.049	2.279	2.242	**0.999**	0.817
LateFusion$_{512}$	✓	1.687	1.430	**2.694**	**2.698**	**1.883**	**1.672**	1.978	1.328	3.089	2.749	1.300	1.190	2.149	1.307	3.224	2.676	1.170	0.900
LateFusion$_{256}$	✓	2.057	1.555	3.071	2.824	2.043	1.743	2.306	1.630	3.342	2.987	1.443	1.305	2.397	1.394	3.514	2.734	1.170	0.923
Ours$_{512\times8}$	✓	1.827	1.358	3.243	2.858	2.097	1.809	1.982	1.331	2.918	2.679	1.359	1.234	**1.331**	**0.865**	**2.187**	2.214	1.035	0.826
Ours$_{512\times4}$	✓	**1.647**	**1.348**	3.028	2.826	1.961	1.765	**1.698**	**1.240**	**2.713**	**2.582**	1.328	1.211	1.450	0.884	2.268	**2.163**	1.053	0.821

Table 1. **Chained shape editing ablation.** We report detailed baseline results for our proposed chained shape editing task, for chain lengths $|\mathcal{P}| \in \{10, 15, 20\}$, using the PC-AE trained latent editors. Both the average final error and average edit error are reported for the Chamfer Distance (CD) and $\mathcal{L}_2$ distance ($\mathcal{L}_2$) metrics. We differentiate between distances to the reconstructed edited shape (CD − Rec) and distances to the real edited shape (CD − Real), which is the most relevant metric for our task. We highlight in grey the model we select for our qualitative results.

Model	decoupled?	$\|\mathcal{P}\|$ = 10						$\|\mathcal{P}\|$ = 15						$\|\mathcal{P}\|$ = 20					
		F$_{\text{CD}-\text{Rec}}$	A$_{\text{CD}-\text{Rec}}$	F$_{\text{CD}-\text{Real}}$	A$_{\text{CD}-\text{Real}}$	F$_{\mathcal{L}_2}$	A$_{\mathcal{L}_2}$	F$_{\text{CD}-\text{Rec}}$	A$_{\text{CD}-\text{Rec}}$	F$_{\text{CD}-\text{Real}}$	A$_{\text{CD}-\text{Real}}$	F$_{\mathcal{L}_2}$	A$_{\mathcal{L}_2}$	F$_{\text{CD}-\text{Rec}}$	A$_{\text{CD}-\text{Rec}}$	F$_{\text{CD}-\text{Real}}$	A$_{\text{CD}-\text{Real}}$	F$_{\mathcal{L}_2}$	A$_{\mathcal{L}_2}$
LateFusion$_{1024}$	✓	1.773	1.452	3.066	2.794	2.002	1.734	1.704	1.290	2.787	2.667	1.212	1.184	1.502	1.049	2.279	2.242	0.999	0.817
+ $\mathbb{O}$ Magnitude	✓	1.118	0.941	2.323	2.242	1.592	1.437	1.037	0.910	2.106	2.142	0.978	0.996	0.608	0.631	1.442	1.892	0.784	0.714
+ $\mathbb{O}$ Direction	✓	0.688	0.360	2.109	2.054	0.822	0.551	0.360	0.279	1.903	1.944	0.410	0.329	0.137	0.341	1.358	1.747	0.207	0.316
LateFusion$_{512}$	✓	1.687	1.430	2.694	2.698	1.883	1.672	1.978	1.328	3.089	2.749	1.300	1.190	2.149	1.307	3.224	2.676	1.170	0.900
+ $\mathbb{O}$ Magnitude	✓	1.206	1.017	2.328	2.300	1.628	1.445	1.192	1.015	2.241	2.311	1.027	1.027	0.777	0.762	1.727	2.039	0.816	0.731
+ $\mathbb{O}$ Direction	✓	0.699	0.386	2.136	2.082	0.805	0.561	0.507	0.356	2.134	2.020	0.500	0.384	0.094	0.345	1.347	1.776	0.172	0.319
LateFusion$_{256}$	✓	2.057	1.555	3.071	2.824	2.043	1.743	2.306	1.630	3.342	2.987	1.443	1.305	2.397	1.394	3.514	2.734	1.170	0.923
+ $\mathbb{O}$ Magnitude	✓	1.150	0.952	2.337	2.280	1.636	1.457	1.272	1.104	2.477	2.429	1.159	1.085	0.684	0.734	1.583	2.072	0.770	0.736
+ $\mathbb{O}$ Direction	✓	0.719	0.423	2.185	2.108	0.846	0.610	0.747	0.449	2.267	2.112	0.695	0.468	0.267	0.463	1.422	1.804	0.309	0.402

Table 2. **Oracle editing results.** We report detailed baseline results when oracle magnitudes and directions are used for the chained shape editing task, on all LateFusion models. We differentiate between distances to the reconstructed edited shape (CD − Rec) and distances to the real edited shape (CD − Real), which is the most relevant metric for our task. Overall, the biggest gain for PC-AE based models is observed when using oracle directions.

When boolean parameters with children are triggered, we sample a random value for each child parameter. For scalar parameters with a continuous domain, we sample a random value from a discretized version of the parameter's range. After generating each parameter vector θ_i, we pass it through a geometry checker (implemented in [12]) ensuring that the resulting shape is valid. If the shape is invalid, the whole chain is discarded and we start over from a different starting pair. Otherwise, the corresponding mesh is synthesized using the shape program ϕ_Θ and added to the chain.

2.2. Text Instructions Generation

We detail in Figure 2 the process of synthesizing text instructions for the chained shape editing task. Starting from an edit vector, we map the edit intensity and the parameter name to a predefined vocabulary set which is randomly sampled. Optionally, the generated instruction is paraphrased using a pre-trained language model. Our instruction generation method is completely automatic and does not require any human annotation or additional data.

2.3. Architecture details

Rendered View Feature Extractor. We use a ResNet-50 [7] model pre-trained on ImageNet [5] to extract rendered view features from the input shapes. Since all synthetic meshes are centered at the origin and aligned, we do not require multi-view features to ensure invariance to rotation. Every mesh is rotated around the z-axis by an angle $\theta = \frac{\pi}{8}$, assigned a base RGB color and rendered from a static viewpoint using the trimesh [4] library.

Autoencoders. We encode pointclouds using a PC-AE [1] model pre-trained on the ShapeNet [3] dataset with a latent dimension $d = 256$. We also use a 3D2VS [17] model also pre-trained on ShapeNet with a latent dimension $d = 512 \times 8$. Note that our method is agnostic to the choice of autoencoder model, and can be adapted to any other shape or pointcloud representation.

PC-AE Latent Editors. We detail here the architecture of our latent editor models.

- **LateFusion$_X$** is composed of a shape latent encoder,

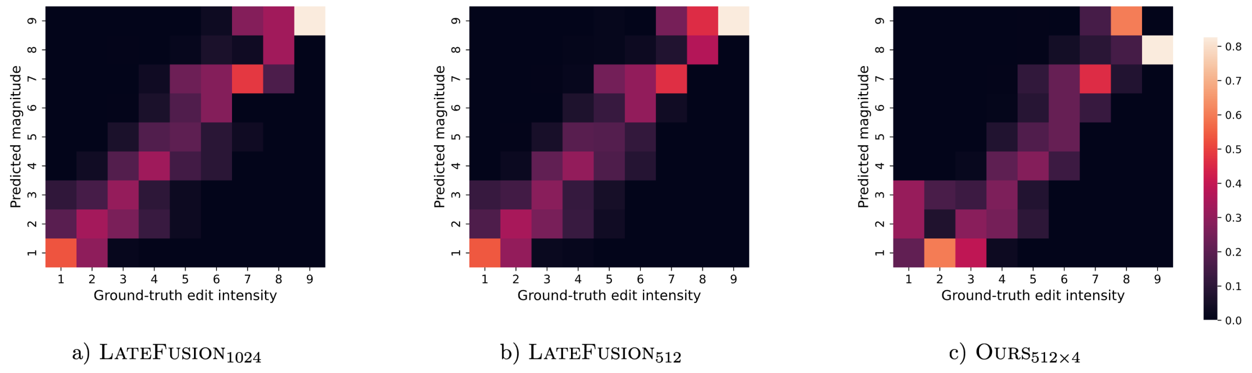

Figure 1. **Comparing predicted edit magnitude to ground truth edit intensity.** We plot confusion matrices comparing the predicted edit magnitudes binned in the interval $[1, 9]$ to the ground truth edit intensities extracted from our metadata. We show results for three decoupled baselines: LateFusion$_{1024}$, LateFusion$_{512}$, and Ours$_{512\times4}$. Overall, we observe a strong correlation between the predicted edit magnitudes and the ground truth edit intensities. However, a bias can be observed in the Ours$_{512\times4}$ model, which tends to deviate from the diagonal predictions, and avoids predicting the maximum possible magnitude range.

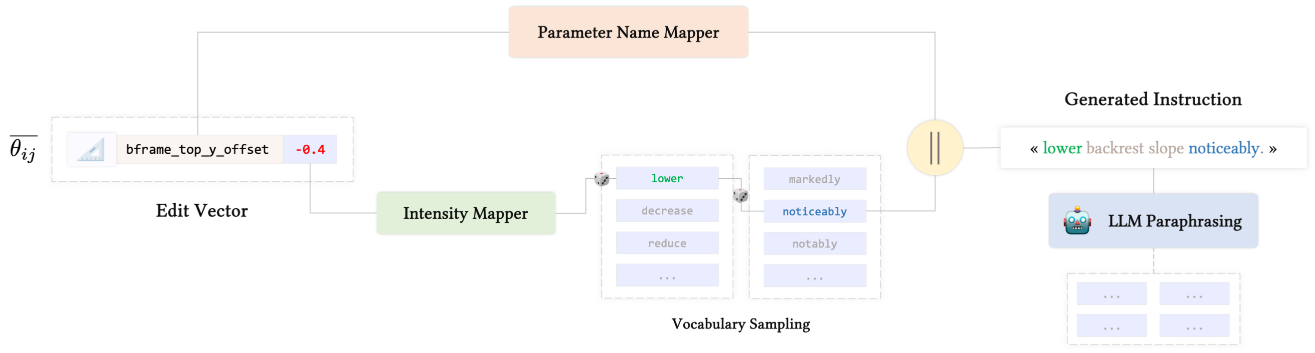

Figure 2. **Synthesizing edit instructions.** Our method generates synthetic text instructions by first applying rule-based generation to translate parameter changes into natural language. The edit intensity is mapped to a natural language intensity depending on the parameter type. To add diversity, the vocabulary describing the edit is randomly sampled, edit directions are randomly inverted, and a paraphrasing transformer model is optionally employed to augment the final instructions.

an edit direction prediction module, and an edit magnitude prediction module. The *shape latent encoder* is a 2-layer MLP with hidden dimensions $[256, 256]$. We use ReLU activations for all layers, and a linear activation for the output layer. The *edit direction prediction module* is a 4-layer MLP with hidden dimensions $[X, 256, 256, 256]$. Since we normalize the predicted edit vector, we remove the final bias term from the output layer. We use batch normalization [8] for all layers except the output layer. The *edit magnitude prediction module* is a 3-layer MLP with hidden dimensions $[256, 128, 64]$ respectively.
For all modules, we use dropout [14] with a probability of 0.1 for all layers except the output layer.

- **Ours**$_{512\times X}$ only uses the edit direction prediction module which is an X-layer MLP with hidden dimensions of the size of the latent space of the autoencoder. We use batch normalization [8] for all layers except the output layer, dropout with a probability of 0.2, and ReLU activations for all layers. We separately predict the edit magnitude and edit direction using the same modules as LateFusion.

We illustrate in Figure 4 the architecture of the latent editor diffusion model.

Text Encoder Model. We use a pre-trained BERT [6] model to encode the language instructions into a 768-dimensional feature vector. We employ the base uncased model with 12 layers, 12 self-attention heads, and $110M$ parameters. Training with larger instances of BERT or with LLM-extracted text features could potentially improve the generalization ability of our models on unseen language instructions.

3D2VS Latent Editors. We fine-tune a latent diffusion

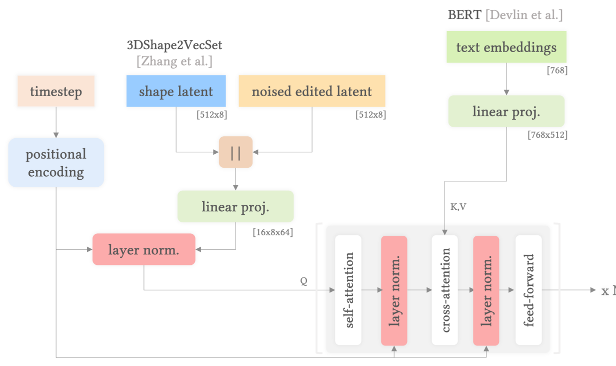

Figure 3. **Diffusion-based latent editor.** We illustrate the architecture of the transformer-based latent editor diffusion model used to edit 3D2VS [17] latents. || indicates concatenation of the input features.

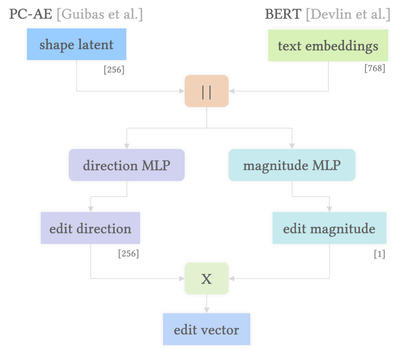

Figure 4. **MLP-based latent editor.** We illustrate the architecture of the MLP-based latent editor used to edit PC-AE [1] latents, for the decoupled magnitude and direction prediction models. || indicates concatenation of the input features.

model trained in the space of a frozen 3D2VS [17] auto-encoder. The best performing model uses a depth of 24 layers with 8 channels and a latent dimension of 512 $\times$ 8, and conditions the model on text features by feeding them as keys and values to the cross-attention layers. We illustrate in Figure 3 the architecture of the latent editor diffusion model.

2.4. Training

Shape Sampling. We sample $N = 4096$ points from each synthesized mesh using the triangle point sampling scheme [16]. Pointclouds sampled from synthetic shapes are centered at the origin, normalized to the unit sphere, and rescaled alongside each axis to align with ShapeNet statistics. A minor downside of normalizing shapes is that the model will only be able to learn to apply edits relative to the scale of the input shapes.

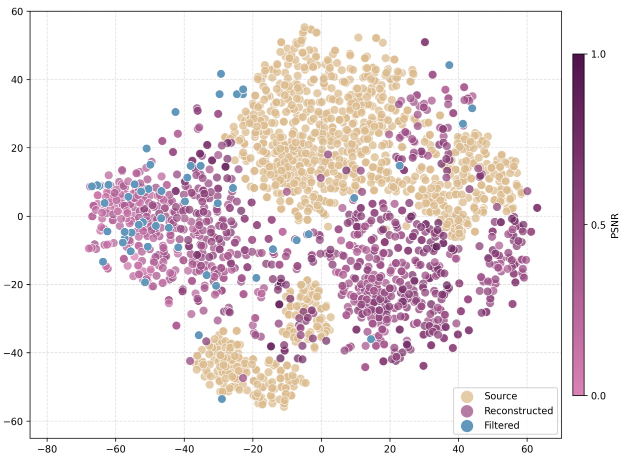

Figure 5. **t-SNE embeddings of shape reconstructions.** We plot t-SNE embeddings [15] of the rendered view features of the input shapes and their reconstructions in the space of the rendered view feature extractor ϕ_R. We also provide the PSNR ratio between input and reconstructed shape rendered views.

PC-AE Latent Editor. We train our models using the Adam optimizer [10] with a learning rate of 1×10^{-4}, and an effective bath size of 128. The learning rate is linearly increased from 1×10^{-6} to 1×10^{-4} during the first 8 epochs, and then gradually decayed back to 1×10^{-6} following a cosine annealing schedule [11]. We train our models on two NVIDIA V100 GPUs with 32GB of memory each. Training on the full *realistic* set takes around 60 minutes, with pre-extracted text and pointcloud features. Almost all models converge within 50 epochs, and we use the best model checkpoint based on the validation loss for all experiments.

3D2VS Latent Editor. To fine-tune the latent diffusion models, we use gradient clipping and a half-cycle cosine after a warmup of 20 epochs. We train the models for 200 epochs with a batch size of 32 on eight NVIDIA V100 GPUs with 32GB of memory each. Training and sampling strategies are the same as in [9, 17].
All text and pointcloud features are pre-extracted and cached to disk to speed up training.

3. Shape Reconstructions

We illustrate in Figure 12 the top-8 and bottom-8 shape reconstructions from the 3DCoMPaT [13] dataset into the GeoCode representation, ranked by rendered view feature similarity. Overall, we observe that the reconstructions are of high quality, and that the model is able to capture the main shape features of the input shapes. The worst reconstructions are discarded from the shape interpolation process as they do not generally lead to realistic reconstructed shapes.

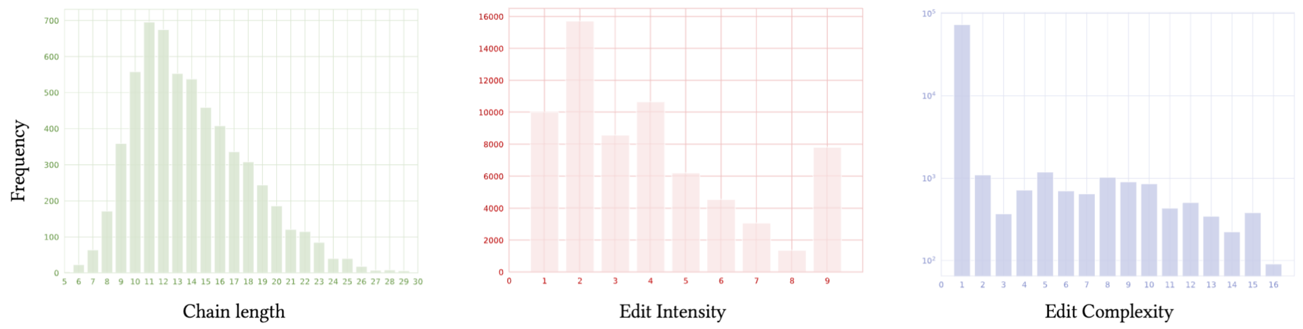

Figure 6. **ShapeWalk dataset statistics.** We plot the distribution of the number of shapes per chain (*left*), of the intensity of edits (*middle*), and of the complexity of edits (*right*). For the edit complexity, note that the frequency is provided in log-scale. The vast majority of edits in the *realistic* subset are granular, affecting a single shape parameter.

In order to further investigate the quality of the reconstructions, we compute t-SNE embeddings [15] of the rendered view features of the input shapes and their corresponding reconstructions in the space of ϕ_R, the rendered view feature extractor. We show the results in Figure 5. While reconstructed (in purple) and source shapes (in yellow) from the 3DCoMPaT [13] dataset form two distinct clusters, we observe that the reconstructions are generally close to the input shapes in the feature space. The filtered reconstructions (in blue) are more likely to be outliers in the feature space, and are thus discarded from the shape interpolation process to avoid unrealistic shape interpolations.

4. Dataset Insights

4.1. Shape Chains

In Figures 10, and 11, we showcase sampled shape chains from our dataset truncated to the first $N = 9$ shapes. For each chain edge, we show the corresponding language instruction describing the parameter changes necessary to transition from one shape to the next.

4.2. Dataset Statistics

In Figure 6, we plot the distribution of the number of shapes per chain (*left*), of the intensity of edits (*middle*), and of the complexity of edits (*right*) in the *realistic* subset[2]. The edit complexity corresponds to the number of shape parameters affected by the edit. For edit complexity, the frequency is provided in log-scale. We observe that the vast majority of edits in the *realistic* subset are granular, and affect a single shape parameter. Chain lengths range from $N = 6$ to $N = 29$ shapes, with an expected length around $N = 15$ shapes. The intensity of edits is not uniformly distributed and skews heavily towards low-intensity edits, which is expected.

[2]Note that we do not show statistics for the *random* subset as all parameters including chain length and edit intensity are static and determined at generation.

5. Diffusion-based Editor Generation Process

We illustrate in Figure 7 the process of generating a shape chain using a diffusion-based latent editor model. We showcase the generation of a shape chain from a source shape to a target shape using our latent editor operating in the space of the 3D2VS autoencoder, illustrated in Figure 3. We show 32 intermediate shapes generated by the diffusion model for each sampling step.

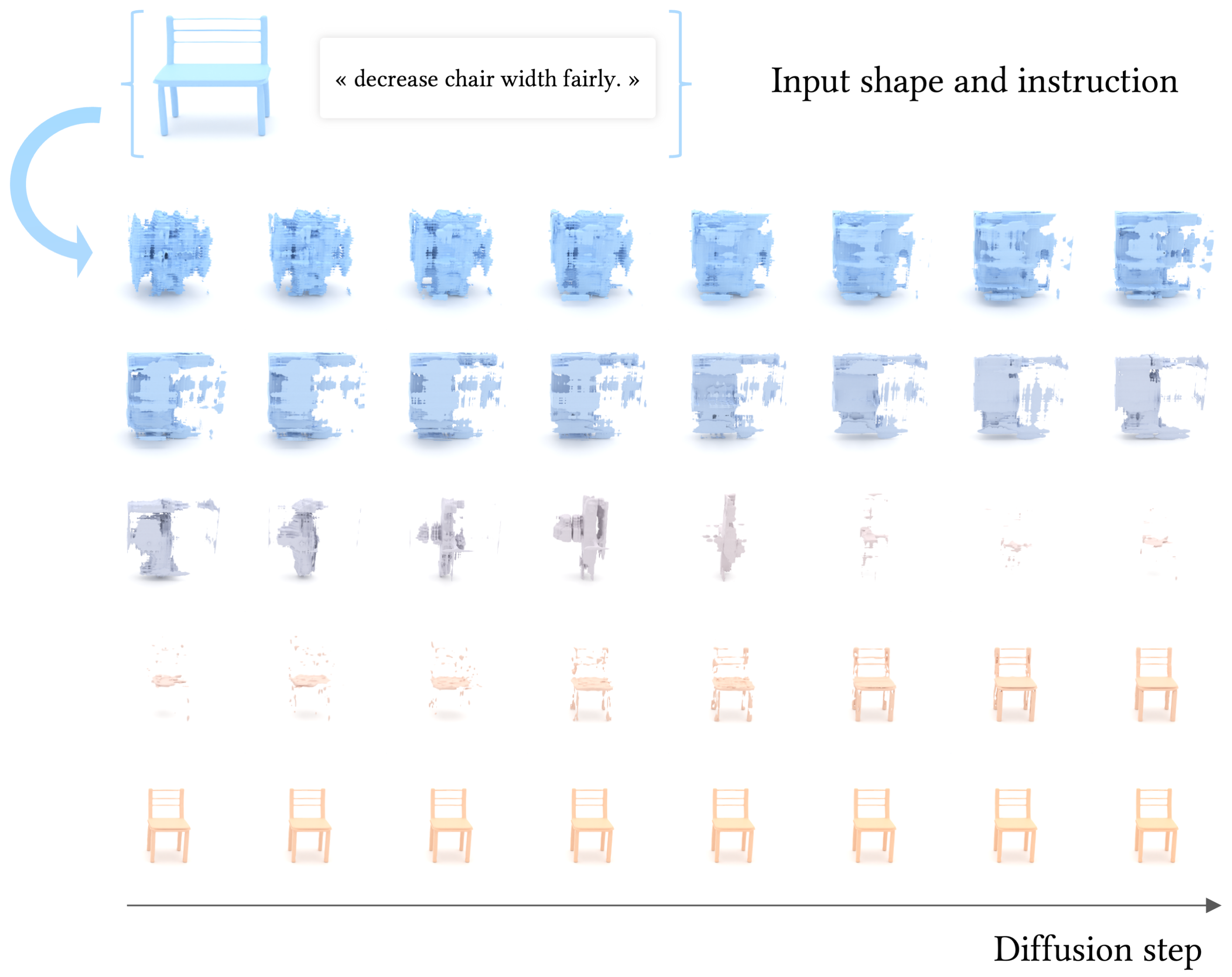

Figure 7. **Diffusion-based shape chain generation.** We illustrate in this figure the process of generating a shape using a diffusion-based latent editor model, for a source shape (**blue**) and a text instruction, to obtain a final target shape (**orange**) after $T = 32$ diffusion steps.

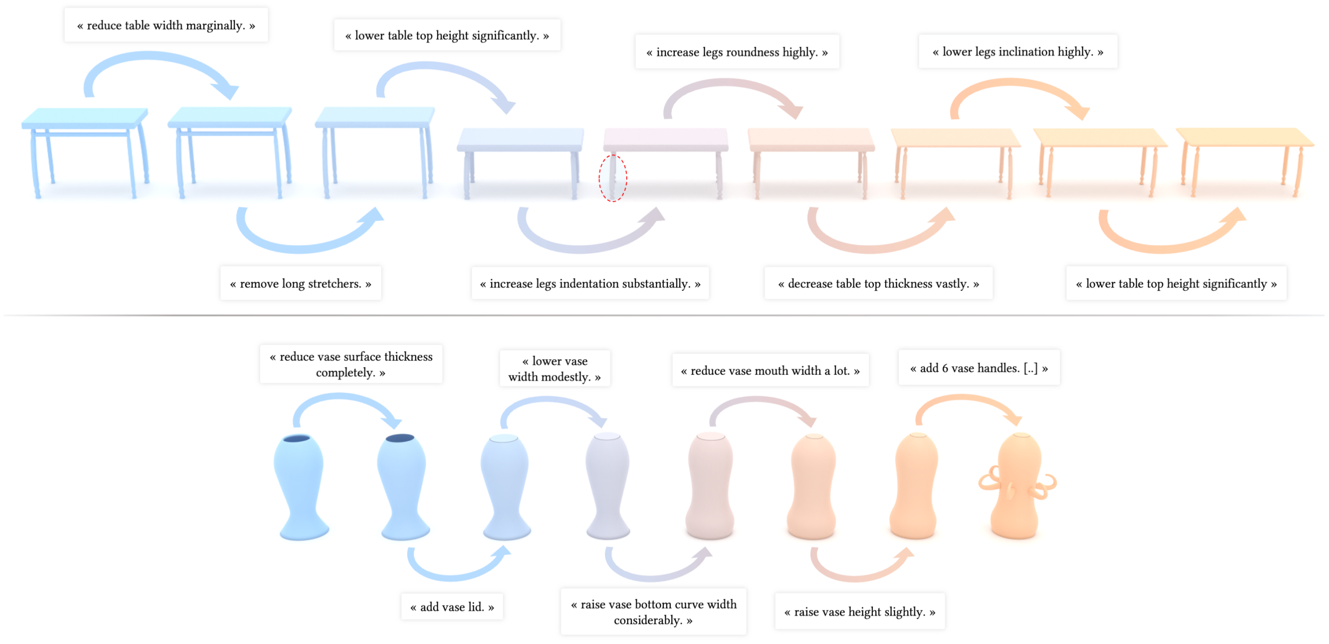
« reduce table width marginally. »
« remove long stretchers. »
« lower table top height significantly. »
« increase legs indentation substantially. »
« increase legs roundness highly. »
« decrease table top thickness vastly. »
« lower legs inclination highly. »
« lower table top height significantly »
« reduce vase surface thickness completely. »
« add vase lid. »
« lower vase width modestly. »
« raise vase bottom curve width considerably. »
« reduce vase mouth width a lot. »
« raise vase height slightly. »
« add 6 vase handles. [..] »

Figure 8. **Chain samples for additional classes.** We showcase in this figure additional chain samples for the *realistic* subset, for the *table* and *vase* classes. Our method is able to generate realistic shape chains for a variety of classes, given support for the corresponding shape programs. In our case, GeoCode [12] supports the classes we use in this work. Extending our method to new classes will require extending these underlying shape programs.

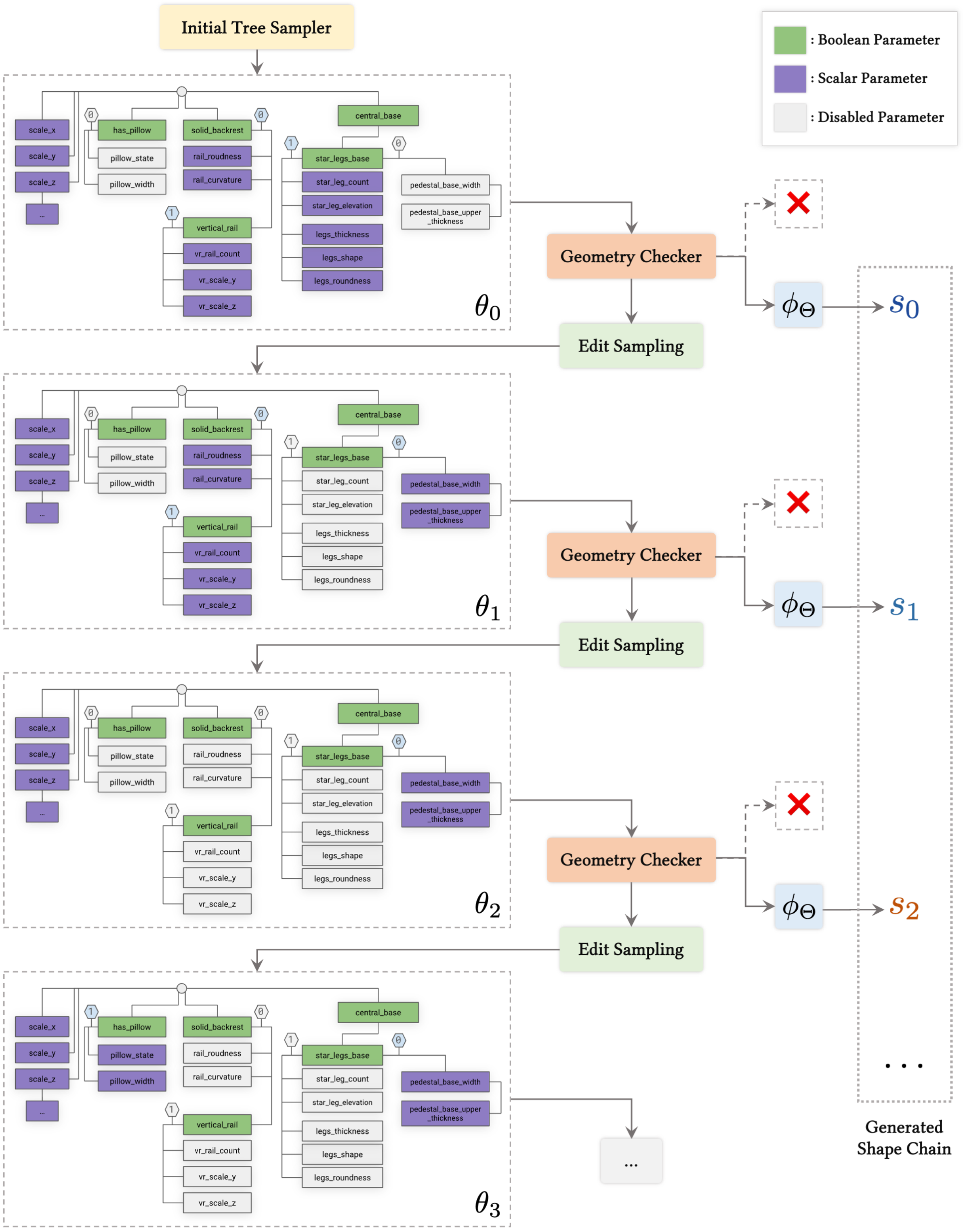

Figure 9. **Parameter tree sampling.** We detail in this figure the process of sampling parameter chains from the parameter tree. Each parameter vector θ_i can be represented as a dependency tree. We start by sampling a random parameter tree θ_1 from the root node of the tree (or using regressed parameters from a real shape in the *realistic* subset). Edit parameters are then sampled randomly (or using interpolation for the *realistic* set) from the tree until the desired chain length is reached (or until the target shape is reached).

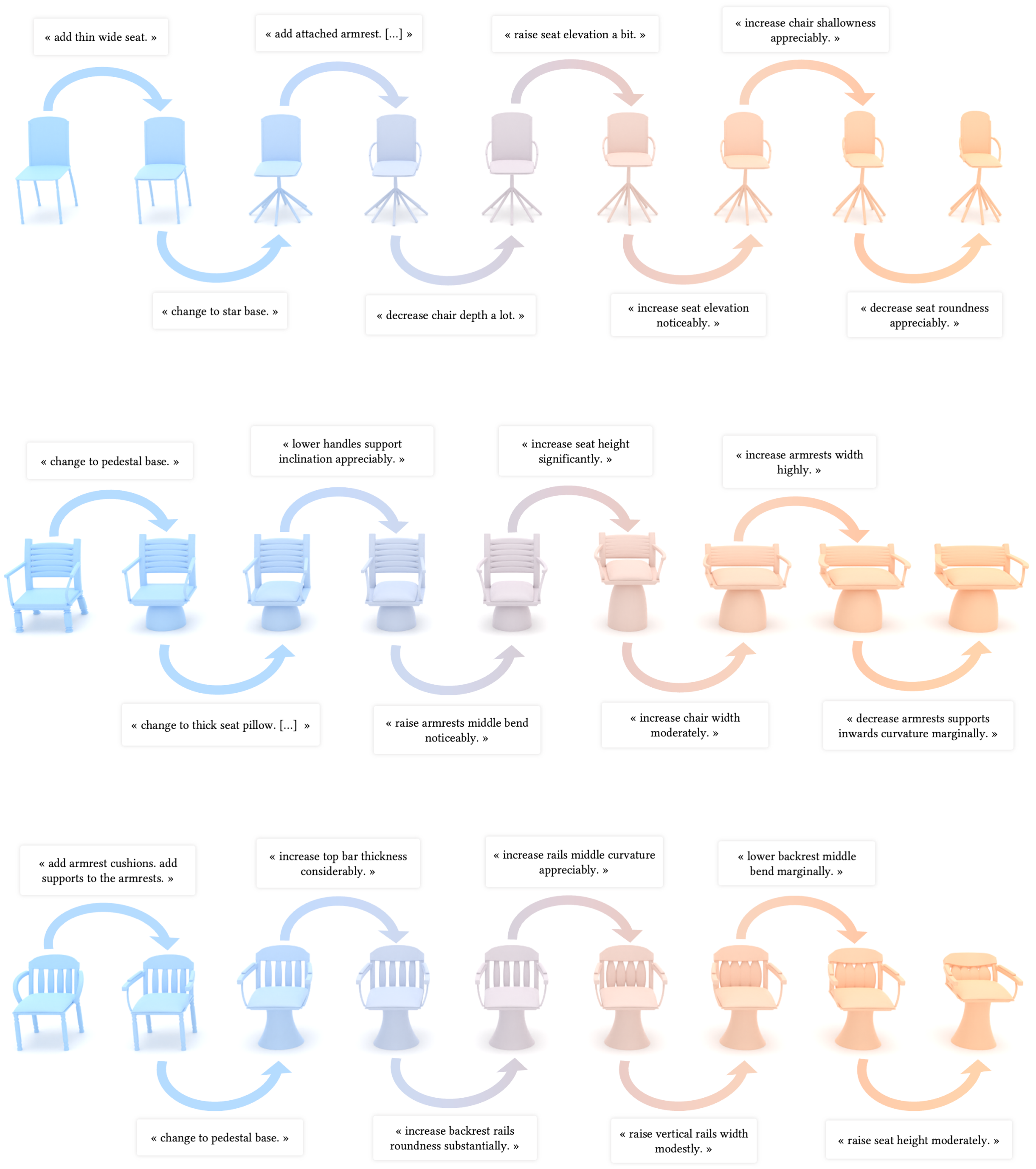
« add thin wide seat. »
« change to star base. »
« add attached armrest. [...] »
« decrease chair depth a lot. »
« raise seat elevation a bit. »
« increase seat elevation noticeably. »
« increase chair shallowness appreciably. »
« decrease seat roundness appreciably. »
« change to pedestal base. »
« change to thick seat pillow. [...] »
« lower handles support inclination appreciably. »
« raise armrests middle bend noticeably. »
« increase seat height significantly. »
« increase chair width moderately. »
« increase armrests width highly. »
« decrease armrests supports inwards curvature marginally. »
« add armrest cushions. add supports to the armrests. »
« change to pedestal base. »
« increase top bar thickness considerably. »
« increase backrest rails roundness substantially. »
« increase rails middle curvature appreciably. »
« raise vertical rails width modestly. »
« lower backrest middle bend marginally. »
« raise seat height moderately. »

Figure 10. **Additional dataset samples.** We showcase sampled chains from our dataset truncated to the first $N = 9$ shapes. We color shapes based on their proximity to the starting shape (**blue**), and the ending shape (**orange**). For each chain edge, we show the corresponding language instruction describing the parameter changes necessary to transition from one shape to the next. All shapes are normalized to the unit cube and centered at the origin.

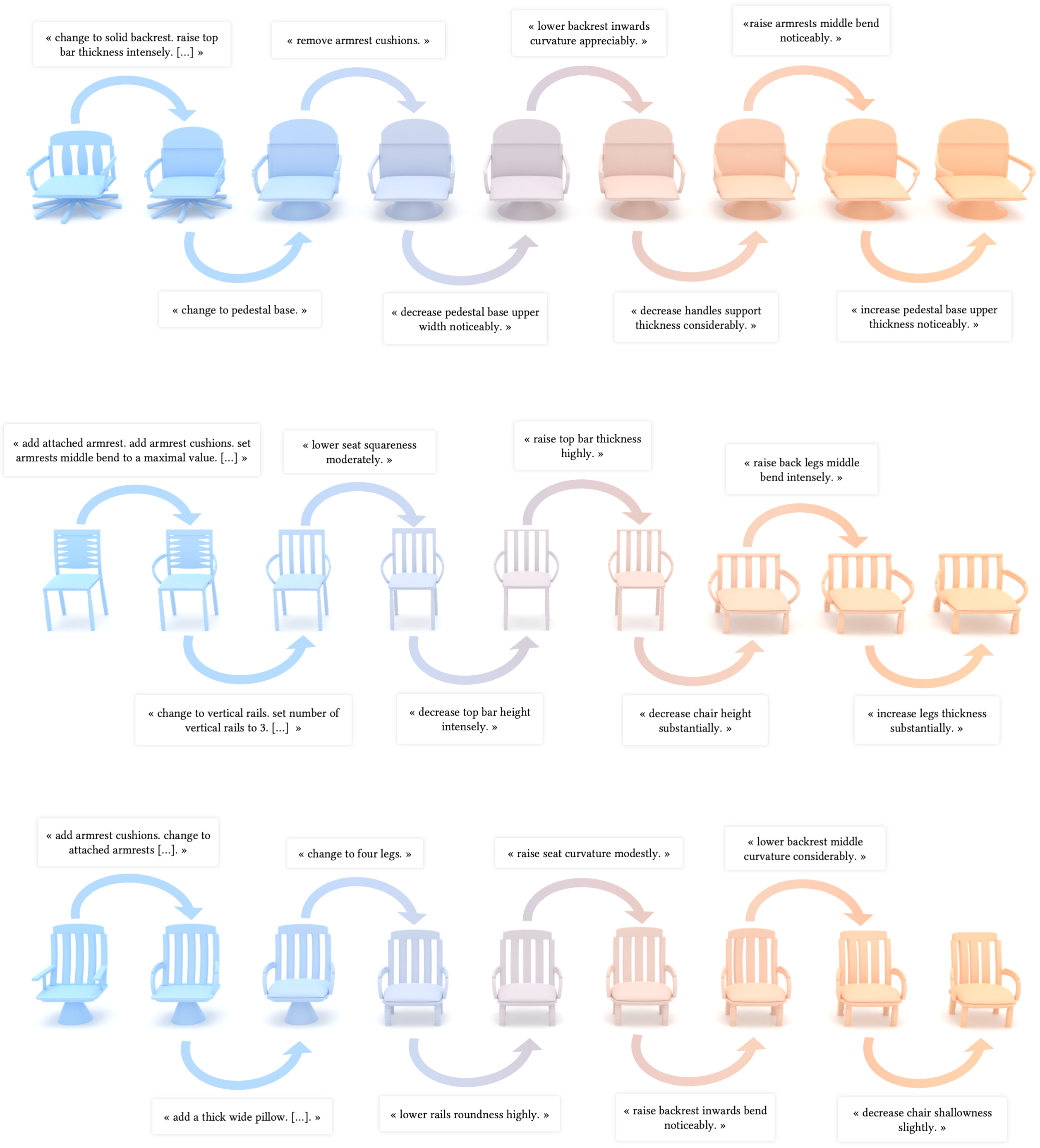
« change to solid backrest. raise top bar thickness intensely. [...] »
« change to pedestal base. »
« remove armrest cushions. »
« decrease pedestal base upper width noticeably. »
« lower backrest inwards curvature appreciably. »
« decrease handles support thickness considerably. »
«raise armrests middle bend noticeably. »
« increase pedestal base upper thickness noticeably. »
« add attached armrest. add armrest cushions. set armrests middle bend to a maximal value. [...] »
« change to vertical rails. set number of vertical rails to 3. [...] »
« lower seat squareness moderately. »
« decrease top bar height intensely. »
« raise top bar thickness highly. »
« decrease chair height substantially. »
« raise back legs middle bend intensely. »
« increase legs thickness substantially. »
« add armrest cushions. change to attached armrests [...]. »
« add a thick wide pillow. [...]. »
« change to four legs. »
« lower rails roundness highly. »
« raise seat curvature modestly. »
« raise backrest inwards bend noticeably. »
« lower backrest middle curvature considerably. »
« decrease chair shallowness slightly. »

Figure 11. **Additional dataset samples.** We showcase sampled chains from our dataset truncated to the first $N = 9$ shapes. We color shapes based on their proximity to the starting shape (**blue**), and the ending shape (**orange**). For each chain edge, we show the corresponding language instruction describing the parameter changes necessary to transition from one shape to the next. All shapes are normalized to the unit cube and centered at the origin.

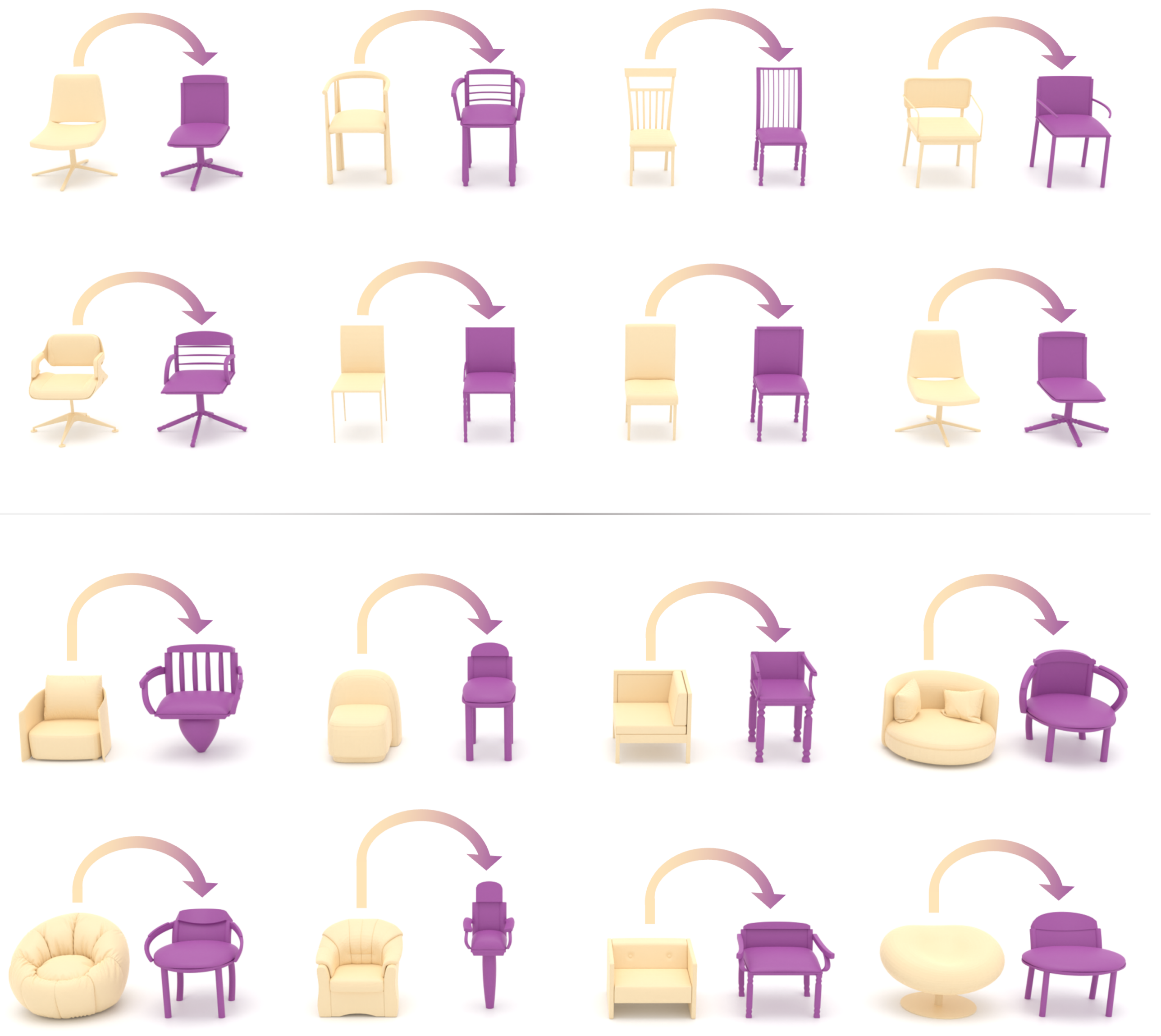

Figure 12. **Shape reconstruction examples.** We illustrate in this figure the original shapes from the 3DCoMPaT [13] dataset (**yellow**), and their corresponding reconstructions into the GeoCode representation (**purple**). In the top two rows, we show the top-8 reconstructions ranked by rendered view feature similarity, and the bottom-8 reconstructions ranked by rendered view feature similarity in the bottom two rows. While top reconstructions are generally of high quality, the bottom reconstructions are discarded from the shape interpolation process as they do not generally lead to realistic reconstructed shapes. We discard these shapes to avoid unrealistic shape interpolations in our shape chain generation process.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018. 2, 4

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016. 1

[3] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. In *arXiv*, 2015. 2

[4] Mike Dawson-Haggerty. Trimesh [computer software]. https://github.com/mikedh/trimesh, 2019. 2

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 3

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 2

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 3

[9] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proc. NeurIPS*, 2022. 4

[10] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015. 4

[11] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. 4

[12] Ofek Pearl, Itai Lang, Yuhua Hu, Raymond A. Yeh, and Rana Hanocka. Geocode: Interpretable shape programs. In *arXiv preprint arxiv:2212.11715*, 2022. 2, 7

[13] Habib Slim, Xiang Li, Mahmoud Ahmed Yuchen Li, Mohamed Ayman, Ujjwal Upadhyay Ahmed Abdelreheem, Suhail Pothigara Arpit Prajapati, Peter Wonka, and Mohamed Elhoseiny. 3DCoMPaT++: An improved large-scale 3d vision dataset for compositional recognition. In *arXiv preprint arxiv:2212.11715*, 2023. 4, 5, 11

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 3

[15] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 4, 5

[16] Eric W. Weisstein. Triangle point picking. MathWorld–A Wolfram Web Resource. 4

[17] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics*, 42(4):1–16, 2023. 2, 4