ChangeIt3D: Language-Assisted 3D Shape Edits and Deformations Supplemental Material

Panos Achlioptas[1,2], Ian Huang[2], Minhyuk Sung[3], Sergey Tulyakov[1], and Leonidas Guibas[3]

[1] Snap Inc.
[2] Stanford University
[3] KAIST

A ShapeTalk Data Set

A.1 Curation of ShapeTalk

A.2 Variations among Annotators and Comparison to Full Descriptions

Figure 3 includes examples among classes not shown in the Main paper. We include here the responses given by two annotators per pair; demonstrating qualitatively the degree that annotators vary in their responses both in terms of content, or the perceived importance of a visual difference (as indicating by similar spotted differences given in different order).

Figure 4 shows representative annotations collected with our **pilot** AMT study where we ask the annotators to describe in detail the changes/transformations that one needs to apply to a shown source shape to make it look more similar to a designated target. Typically, such descriptions are vague and highly under specific ignoring key details that are necessary to change to make the source shape more similar to the target, prompting to opt for the task of difference *enumeration*.

A.3 Data Set Analysis

Table 1 includes the number of total collected utterances, and corresponding submissions included per class, along with the number of its class's unique shapes in ShapeTalk.

As can be seen in Table 2, the distribution of language usage varies from category to category. For categories that have more complex partonomy (e.g. "pistol", "chair", "skateboard"), the part references are used more, and holistic shape references are less common. However, for instance, the "bathtub" category shows a case where language more holistic compared to other categories, and language used on parts is less common than other categories. Please see Main

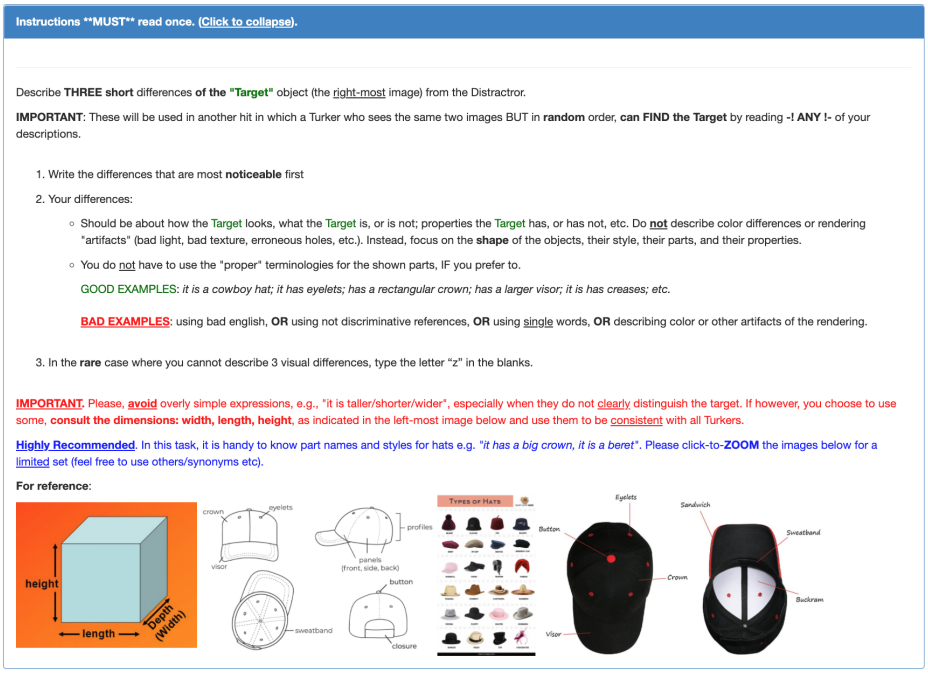

Fig. 1: **Instructions given to Annotators.** Each shape class was presented with a specialized instruction-set highlighting common part names, and/or styles for the underlying objects (shown examples concern 'hats'). For each shape classes we requested a distinct number of maximum differences to be reported (between three and five) according to the visual complexity of the class and its underlying density/variations of objects.

paper, Section 3 for a detailed explanation of the different linguistic categories used.

A similar analysis regarding the average number of tokens (words) used by annotators across the different shape classes is presented in Table 3.

Fig. 2: **Example of a pair of shapes and prompts as shown to annotators.**

Table 1: **Volume** of the final (merged) shape classes of ShapeTalk.

Class	Unique Shapes	Unique Submissions	Utterances
Airplane	2,722	11,130	44,450
Bag	139	834	3,327
Bathtub	663	3,073	12,095
Bed	747	2,985	14,870
Bench	1,657	6,633	25,618
Bookshelf	816	3,263	16,114
Bottle	492	1,968	5,897
Cabinet	246	984	3,860
Cap	208	1,263	3,775
Chair	6,608	15,210	74,743
Clock	580	2,320	9,231
Display	1,174	4,909	14,703
Dresser	1,690	6,740	33,567
Faucet	640	2,560	10,192
Flowerpot	623	2,492	9,867
Guitar	753	3,013	12,024
Knife	424	1,696	6,783
Lamp	2,312	13,642	53,956
Mug	208	832	2,488
Person	94	564	2,820
Pistol	302	1,208	4,832
Scissors	80	480	1,438
Skateboard	152	912	2,724
Sofa	3,065	13,025	51,932
Table	8,176	21,316	88,975
Trash bin	343	1,372	5,346
Vase	823	3,290	13,074
Total	**35,737**	**127,714**	**528,701**

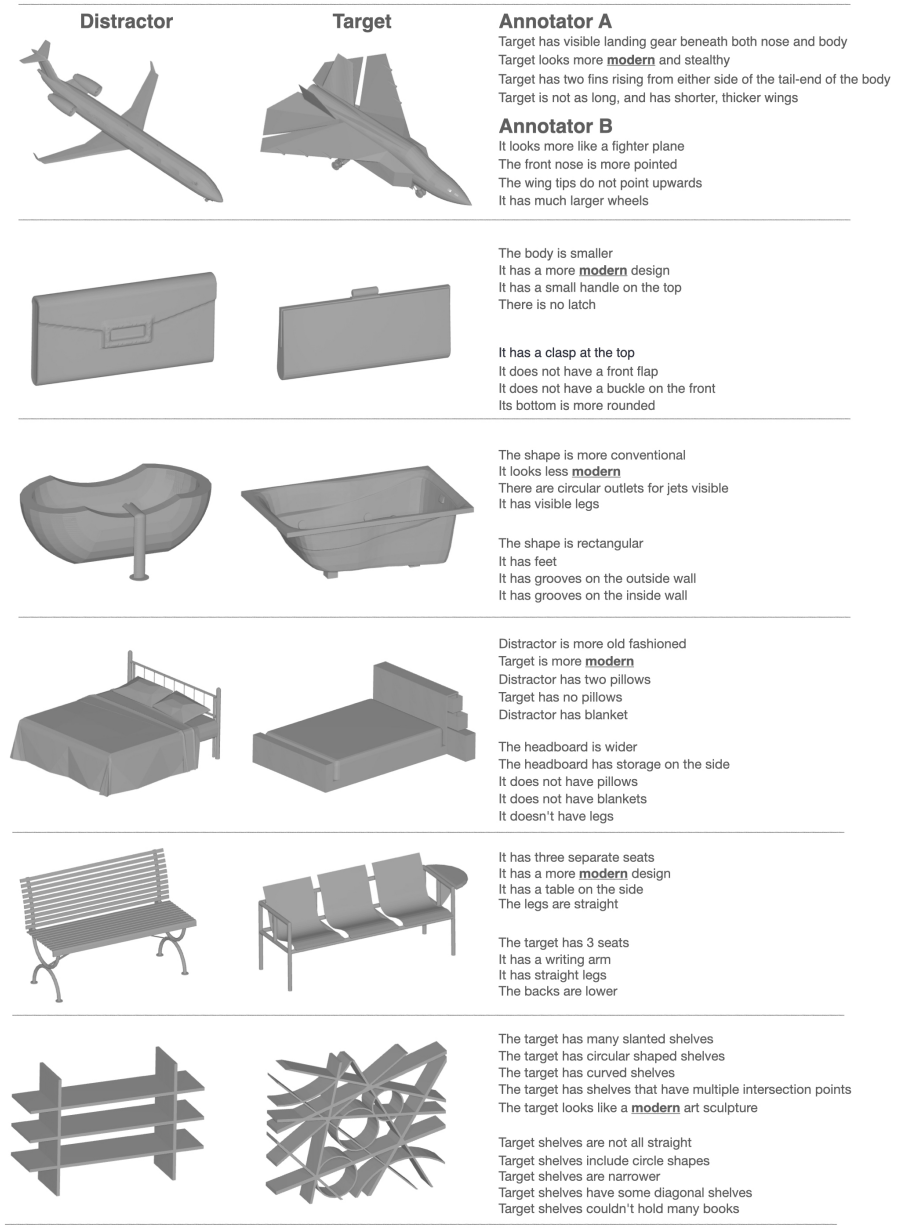

Fig. 3: **ShapeTalk annotations across different classes and annotators.** The word *modern* is included in at least one response.

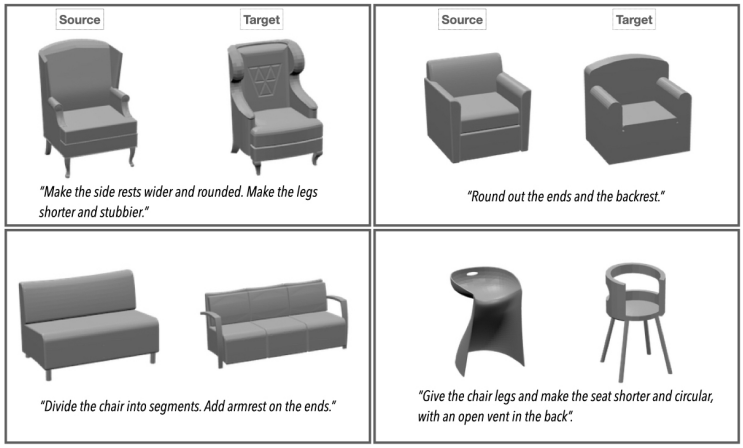

Fig. 4: **Examples from our *pilot* study**. In this pilot, Turkers were asked to describe in language how to transform one shape (source, left-most of each pair) into a target (right-most shape of each pair). Examples of derived descriptions are shown under each shape pair. Such descriptions tend to be significantly under- specific highlighting the hardness of this task.

Table 2: The proportion of utterances containing different types of information, analyzed based on the words that they contain.

class	part	local	holistic	stylistic	dimension	geometric
Airplane	0.87	0.02	0.13	0.02	0.31	0.33
Bag	0.63	0.10	0.32	0.10	0.28	0.21
Bathtub	0.51	0.10	0.43	0.05	0.32	0.37
Bed	0.80	0.02	0.19	0.05	0.34	0.12
Bench	0.86	0.04	0.14	0.05	0.32	0.22
Bookshelf	0.69	0.04	0.30	0.09	0.32	0.15
Bottle	0.81	0.07	0.19	0.04	0.54	0.21
Cabinet	0.65	0.03	0.34	0.05	0.32	0.17
Cap	0.75	0.02	0.24	0.07	0.35	0.23
Chair	0.88	0.05	0.11	0.08	0.33	0.27
Clock	0.59	0.15	0.39	0.09	0.21	0.30
Display	0.90	0.05	0.09	0.01	0.44	0.28
Dresser	0.76	0.04	0.22	0.06	0.35	0.17
Faucet	0.93	0.05	0.06	0.03	0.36	0.30
Flowerpot	0.67	0.05	0.31	0.04	0.41	0.26
Guitar	0.87	0.09	0.10	0.09	0.17	0.23
Knife	0.87	0.13	0.12	0.11	0.26	0.33
Lamp	0.83	0.04	0.17	0.06	0.38	0.30
Mug	0.75	0.02	0.24	0.02	0.55	0.24
Person	0.72	0.02	0.28	0.03	0.13	0.11
Pistol	0.91	0.07	0.08	0.07	0.34	0.32
Scissors	0.67	0.28	0.08	0.11	0.40	0.27
Skateboard	0.91	0.04	0.09	0.03	0.44	0.26
Sofa	0.85	0.03	0.15	0.05	0.35	0.19
Table	0.76	0.06	0.23	0.06	0.41	0.24
Trash bin	0.71	0.20	0.27	0.08	0.25	0.29
Vase	0.73	0.06	0.25	0.03	0.55	0.21
All	**0.81**	**0.05**	**0.18**	**0.06**	**0.36**	**0.25**

Table 3: **Average number of tokens and utterances and Hard contexts** per shape class of ShapeTalk. The reported average (last row) is weighted by the number of utterances collected per each class.

Class	Number-Tokens	Hardness Prc.
Airplane	6.82	50.18
Bag	6.58	49.95
Bathtub	6.40	49.19
Bed	5.47	49.89
Bench	5.81	48.08
Bookshelf	5.90	49.50
Bottle	6.69	49.96
Cabinet	5.80	49.04
Cap	6.36	49.85
Chair	6.58	46.32
Clock	6.05	49.81
Display	6.61	49.88
Dresser	5.62	49.67
Faucet	6.61	49.80
Flowerpot	5.90	49.63
Guitar	7.69	49.87
Knife	6.55	49.99
Lamp	6.32	52.96
Mug	5.25	49.84
Person	6.52	50.00
Pistol	7.34	50.00
Scissors	5.70	50.56
Skateboard	6.27	49.85
Sofa	5.81	48.71
Table	6.26	72.63
Trash bin	6.59	48.75
Vase	6.02	49.73
Weighted Mean	**6.27**	**53.28**

B Details on Listening and Editing Systems and Data

B.1 Dataset Preprocessing (w.r.t. Language of ShapeTalk)

For all deep learning based experiments involving ShapeTalk *i.e.* Neural Listening and Language-assisted Editing we remove from the dataset utterances with *more* tokens than the 95% percentile (more than twelve words). Moreover, we replace with a special token (<UNK>) rare words; those that appear in fewer than two utterances across an underlying train split of ShapeTalk.

B.2 Details on the Train/Test/Val Splits

Our neural-listeners and editors operate with inputs that are comprised by *pairs* of shapes coupled with a discriminative utterance for each of their designated *target*. Our pretrained generative networks on the other hand expect a simpler input, namely, a single shape. In order to build fair (and least biased) train/test/val splits and link together in our ChangeIt3DNet framework: i) an editor module that learns how to change the *distractor* shape to conform more with the language uttered to separate it from a target, together with, ii) a guiding pretrained, neural-listener that is trained to classify the *target* shape based on discriminative language for it (against a distractor), and finally, operate in (i) and (ii) with latent representations encoding a shape based on a pretrained 3D generative model; we had to consider different approaches. If data spillage was not an actual bottleneck the fairest/simplest approach would be to keep everything between all test/train/val splits disjoint, across *all* three neural components. However, given that ShapeTalk was not built (nor should be) with a *single* specific train/test/val split in mind; we opted for the least wasteful, and still fair, data separation: A) we built the generative shape networks with train/test/val splits each comprised by separate (disjoint with each other) shapes. B) for the neural-listeners we separate the ShapeTalk shape-pairs according to the splits we made in (A) according to *where the target shape of each pair belongs*, i.e., we ignore the distractor when deciding the split of a shape-pair/language datum. Last, C) when we train the editor module of ChangeIt3DNet we again use the same train-split we used for (B) but we exclude first the training pairs for which the *distractor* comes from the test split of (A). Thus when reporting neural-listening accuracy (B) we use and report pairs with *unseen* target shapes per the 3D generator. When reporting editing performance, we similarly use *unseen* per the 3D generator, distracting shapes coupled with unseen language that was describing a corresponding target in (B).

B.3 Multi-class learning

We train and extract latent shape representations from 3D shape AutoEncoders that are trained simultaneously with *all* classes of ShapeTalk; without specializing the creation of a 3D latent space concerning solely a single class. Aside of logistical convenience, and the increased generality of such an underlying latent

space, this choice directly enables us to train neural listeners with all utterances of ShapeTalk, of all classes, with a single neural-listening system. Despite the fact that per class-specializing 3D AutoEncoders tend to attain slightly better reconstructions than our multi-class AEs; a neural listener trained with the utterances concerning all classes of ShapeTalk tends to attain on average higher accuracy, in many cases up to more than 10% better than if trained only with the latents/utterances coming from the single underlying class. This performance boost is especially pronounced in classes with small number of shapes and corresponding discriminative language; implying that there exist learnable geometric language in ShapeTalk that is *transferable* across classes and which can act as a regularization in data spare regimes. Because of this phenomenon, in all our reported numbers, including those concerning editing we use both a 3D shape-encoder and a corresponding neural-listener trained with *all* classes/language of ShapeTalk.

C Neural Listening-Comprehension

C.1 Main Neural Listening Architecture

We use the *context-free* variant of ShapeGlot [2] operating with 256D latent representations extracted by a corresponding 3D shape generator (a PC-AE [1], or ImNet [4], or SGF [3]). This listening variant is fast to train, high-performing (as show in ShapeGlot and PartGlot [8]) and enables us to measure the compatibility of an *arbitrary* number of shapes with a given utterance. In terms of hyper-parameters, we use a word embedding with 128D, and a visual projection-layer that transforms each input latent code to 128D. For this projection-layer we use a 2-layer deep MLP network with 128 neurons on each layer, where each layer is closed with a ReLU non-linearity [11] and a batch-normalization layer [7]. Moreover, we regularize these two layers with 0.2 dropout probability [15]. We also use a two hidden-layer MLP-based classification-head that operates on a multimodal fused representation of the visual signal and the transformed/latent representation of each discriminative utterance (see next). The classification head is comprised by 100, and 50 neurons per hidden (FC/ReLU/batch-normalization) layer and its final (two) output neurons are implemented with a fully-connected layer. We note that we apply weight-decay (0.005 weight) on the L2-norm of all FC layers of our listener. To transform the input tokens into a single linguistic latent representation we experimented with two approaches (LSTM-based and Transformed-based) each operating and optimizing the underlying word-embedding vectors. In the LSTM-based setup, we use a unidirectional LSTM [6] cell with 128D hidden neurons for which we initialize its hidden state with the 128 projected visual representation of each stimulus (we unroll the LSTM twice; once per shape-utterance of a context). We extract from the LSTM a resulting multimodal representation by collecting the maximum values of its output vectors per each token of the utterance; before passing them to the aforementioned classification head. For the Transformer-based result, we use a transformer encoder instead of an LSTM to encode an utterance and concatenate

the transformer's final output (upon reaching <eos>) with the projected visual representation, before passing their concatenation to the listener's classification head. We use a transformer with 128D embedding, 4 layers deep, each with 4 attention heads [17].

Note. All our pre-trained models, evaluation and training code will be released upon publication with detailed documentation to promote reproducibility.

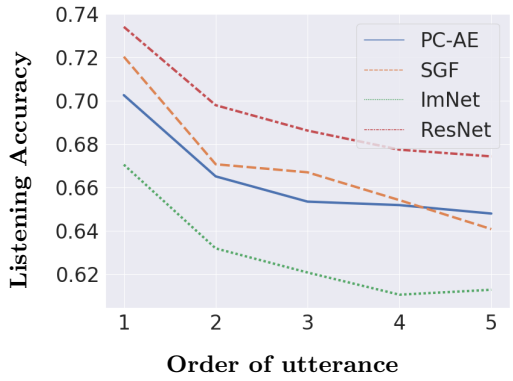

Fig. 5: **Listening accuracy, per decreasing saliency of an utterance (order it was given), with different visual shape encoders**. For our main neural listening architecture, we vary the latent visual representations the neural listener is trained/tested with. This figure shows a positive correlation between the order an utterance was given by an annotator and the capacity of the neural listener to generalize to it and accurately predict its underlying target. Together with the three 3D encoders described in the Main paper (PC-AE, SGF, ImNet), we also include here the accuracy based on a 2D ResNet-32 encoder pretrained on ImageNet operating with a single monochromatic rendering of each shape from a fixed point of view ('ResNet').

C.2 Discussing Listening Performance

Table 4 reports the attained performance with each of the three 3D latent representations across each class. It also includes the attained performance of the main listener when operating with a 2D-based latent representation. Specifically, when using a ResNet-32 pretrained on ImageNet to extract the a 512D latent representation for a shape, based on a 2D monochromatic rendering of it (from a fixed angles across all shape-classes). Also, we include the performance of a pretrained CLIP [13] model operating with the same renderings and without fine-tuning it. It is interesting to note that learning to discriminate fine-grained shape details with 2D image representations of shapes has advantages: the performance advantage of using latent representations extracted via a image-based encoder pretrained on ImageNet operating on 2D renderings of shapes was first

Table 4: **Neural comprehension performance (% accuracy) per shape class**. The reported average (last row) is weighted by the number of utterances collected per each class.

Class	ResNet	ImNet	PC-AE	SGF	CLIP
Airplane	69.9	65.4	67.6	69.9	51.0
Bag	48.4	55.0	55.3	61.3	54.8
Bathtub	67.1	56.3	65.6	68.4	52.3
Bed	71.1	63.3	68.7	67.3	51.9
Bench	70.4	66.7	71.0	71.5	52.1
Bookshelf	61.0	58.4	61.8	62.4	52.1
Bottle	64.9	67.4	62.8	67.8	54.3
Cabinet	71.4	61.8	56.9	62.3	52.7
Cap	71.0	60.8	68.3	61.8	54.1
Chair	79.0	71.3	75.0	75.3	53.5
Clock	63.4	56.7	63.6	63.3	55.9
Display	70.7	60.3	66.1	64.1	51.9
Dresser	73.4	64.0	65.4	67.2	52.0
Faucet	64.2	56.4	58.6	61.3	51.1
Flowerpot	65.7	61.6	66.2	64.6	53.3
Guitar	63.7	58.3	59.7	62.7	54.5
Knife	60.8	48.7	58.8	58.3	50.6
Lamp	64.7	58.5	60.8	63.5	52.0
Mug	58.1	62.1	64.1	64.9	50.3
Person	49.3	19.0	39.2	34.3	60.5
Pistol	60.6	58.6	62.6	63.3	50.6
Scissors	50.7	41.5	56.3	51.4	48.8
Skateboard	68.4	58.2	60.5	66.2	47.1
Sofa	75.3	65.3	70.7	69.9	52.5
Table	74.4	67.3	70.2	70.9	53.1
Trash bin	60.9	61.3	63.2	68.3	51.6
Vase	67.1	65.2	66.0	69.0	53.6
Weighted-mean	72.2	65.2	68.7	69.6	52.6

demonstrated in a ShapeGlot-based study [2], and now is further confirmed with our ShapeTalk (using such image-based latents to train/test our listener gives overall a 2.6% improvement over SGF). As noted, in Section 4 of the Main paper, CLIP's performance is close to random guessing in ShapeTalk, with a notable exception the class of "Person" for which presumably the original dataset CLIP was trained on contains a non-trivial amount of references. Last, it is worth re-emphasizing here that ShapeTalk appears to be a challenging dataset, and thus, despite using modern deep-learning tools in our approach their overall attained performance is far from perfect - suggesting that perhaps several breakthroughs are necessary to attain human-level visio-linguistic understanding of common 3D shapes.

D Editing Experiments

We note that unlike the practice we followed for 3D shape generators and neural-listeners (see also Subsection B.3) when training a ChangeIt3DNet we use exclusively the corresponding data of each underlying class we are concerned with i.e., we train a different ChangeIt3DNet per class.

The shape editor (E) depicted in the architecture diagram of Main paper (Fig. 6), is comprised by two shallow MLP-based layers that are responsible for finding i) the latent *direction* that needs to be followed and *added* to the input shape latent to achieve the requested change, and ii) for finding a scalar value that vector-multiplies the found direction i.e., a value acting as the *magnitude* that captures how far the direction should be followed. Aside of providing as input to E the 'source' latent which we wish to change (a distractor shape in ShapeTalk data), we also encode and provide as input a corresponding utterance that is *incompatible* with the source and which indicates the apply to it (this utterance take from ShapeTalk was given by an annotator to describe a different target-shape in a context involving the input source). To encode the utterance we use the same LSTM-based architecture described in Subsection C.1. We concatenate (fuse) this linguistic representation with a 2-layer deep MLP-ReLU network with $[256, 256]$ neurons operating on the shape-latent, and acting as a projection layer; before passing the concatenated representation to the final two shallow MLPs (with [256, 256] and [256, 1] neurons each) responsible for the finding of the aforementioned latent direction and its magnitude. We note that the found latent direction is unit-normed normalized before multiplied with the corresponding learned magnitude.

The loss function used to optimize E is comprised by two terms: a cross-entropy term produced by the frozen pretrained listener (L); acting on the original vs. the edited latent codes, with the edited latent indicated as the ground-truth target (for the same input utterance used to make the edit), and a second term that controls how far the edit is from the input latent; given as the L2 distance between the two vectors. We control the balance between these two terms via a single scalar hyper-parameter, which we denote as γ which scales down the relative importance of the second term compared to the first ($\gamma in [0, 1]$). Figure 6 shows the effect different values of γ across our main evaluation metrics have when using the SGF backbone (note the x-axis across all metrics being referred to as the 'identity-γ' since this hyper-parameter directly affects the preservation of the identity between the edited and input latent shapes). The findings presented in Figure 6 are intuitive: the larger values we use for γ the more we preserve the identity between the input/output as reflected in both geometric metrics (GD, l-GD), while also naturally improving in the CP metric, since we apply a more conservative (per L2-distance) change. Of course, this improvements come at the expense of the utterance-edit listening compatibility as measures in LAB in bottom-right sub-figure. Last, in this figure we note how the ChangeIt3DNet trained with *Self-Contrast* (see also Main Section 5) improves the GD, l-GD and CP metrics at a slight expense for the LAB. This is also an intuitive finding, since when self contrast is off, i.e., we contrast in

Table 5: **Quantitative comparisons when using different backbone 3D-shape generators in *ChangeIt3D*.**

Backbone	**GD** (↓)	**LAB** (↑)	**CP** (↓)	***l*-GD** (↓)
ImNet [4]	0.494	62.4	6.8	0.993
PC-AE [1]	0.399	71.5	5.1	0.847
SGF [3]	**0.356**	**77.7**	**3.8**	**0.818**

the cross-entropy loss term, the edited latent with the actual target latent from the relevant context in ShapeTalk; we force in many cases the discovery of large in magnitude edits to compensate for the fact that many ShapeTalk pairs are significantly different (approx. half of ShapeTalk is made with Easy pairs).

Table 5 reports the evaluation metrics for ChangeIt3DNet (*Decoupled* and with *Self-Contrast* when using three different backbone architectures for the 3D shape AEs (with a $\gamma = 0.4$). Across the board all metrics favor SGF. We note however, that such *direct* comparison is less robust to interpretation since i) different backbones have their own prior quality in shape reconstructions which is an orthogonal problem to the editing, and ii) LAB is measured with a dedicated (different) neural-listener operating with the latents of the underlying 3D AE; which as shown in Table 4 have different performance for the listening task. Last, we note that to measure the l-GD, GD based on Chamfer distance, and CP, we rely on a single final shape representation, for all backbones, that of 3D pointclouds with 2048 points per shape. PC-AE and SGF directly output 3D pointclouds; for ImNet we use iso-surface extraction to convert the output implicites to meshes that we then convert o 3D pointclouds by surface sampling.

Finally, we note that for the Monolithic baseline (Section 6) we introduce an extra symbol in our vocabulary to denote the concatenation of the utterances given by a single annotator ($< merge >$) and build its architecture based on 3D pointcloud (PointNet-based [12]) shape encoder which encodes the source (distractor) shape at 2048 point resolution, and an LSTM-based utterance-encoder similar to the one described in Subsection C.1. This utterance-encoder processes the entire sequence of *all* 'merged' utterances given by an annotator in the context of a ShapeTalk shape-pair, in the same order. The outputs of the two encoding networks are concatenated and are further processed by an MLP (ReLU-batch-normed-based) decoder with $[256, 256, 512, 2048 \times 3]$ neurons which is trained to reconstruct the corresponding target shape from ShapeTalk according to a Chamfer-distance loss-function.

D.1 More Qualitative Results for Editing

Figures 7, 8 show qualitative examples of language-assisted editing based on ChangeIt3DNet when using an IM-Net and PC-AE shape encoder, respectively. While 8 shows that utterance-compatible changes can be done in the PC-AE latent space, 7 shows that this can be done with comparable reconstruction quality with the IM-Net backbone, and the added benefit in this case of attaining high-quality meshes as well.

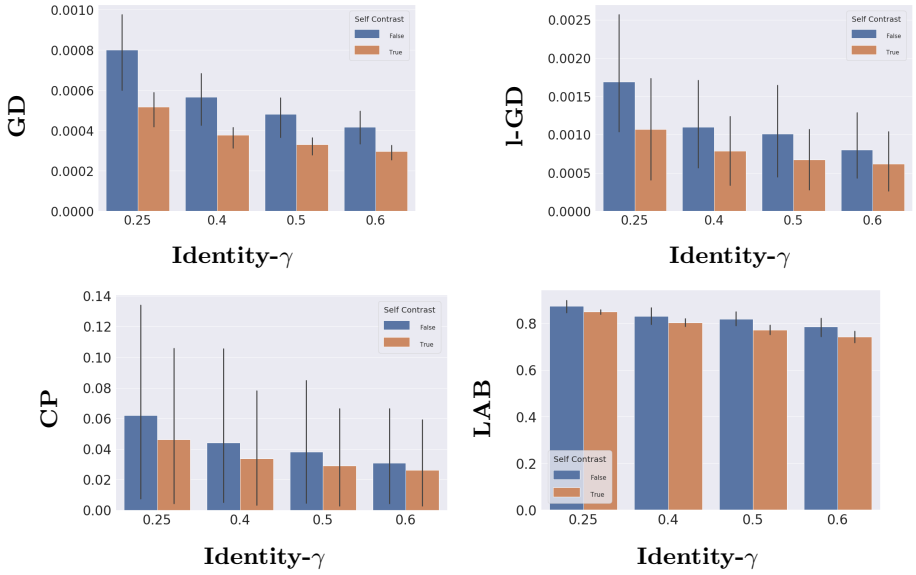

Fig. 6: GD, l-GD, CP and LAB scores as the weight of the identity penalty varies in the training loss of ChangeIt3DNet. We find that as the requirement for identity preservation becomes more stringent, a trade-off between GD and l-GD begins to happen with increasing LAB-scores, indicating tension between the two underlying objectives. The histograms are average scores across the metrics over three classes (chairs, tables and lamps) and the bars reflect the variance among them.

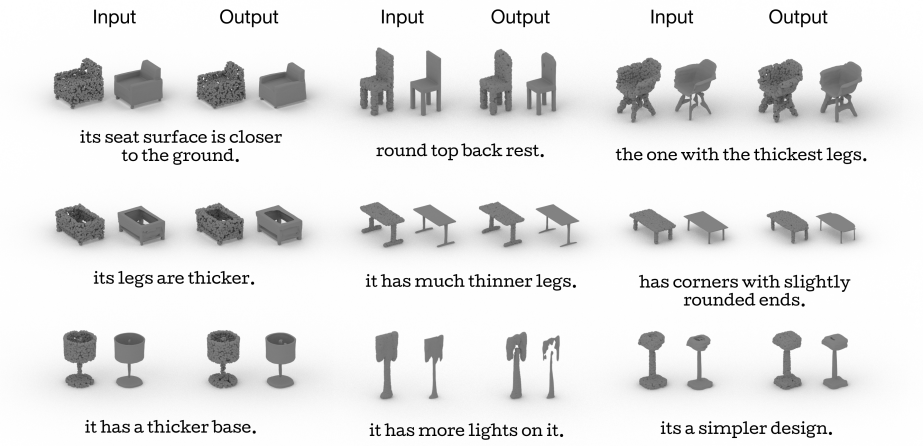

Fig. 7: Qualitative examples of language-assisted edits in the IM-Net latent space. Note that we show both the pointcloud and the mesh for each input-output pair, and that the 2048 points in each pointcloud are sampled on the reconstructed mesh.

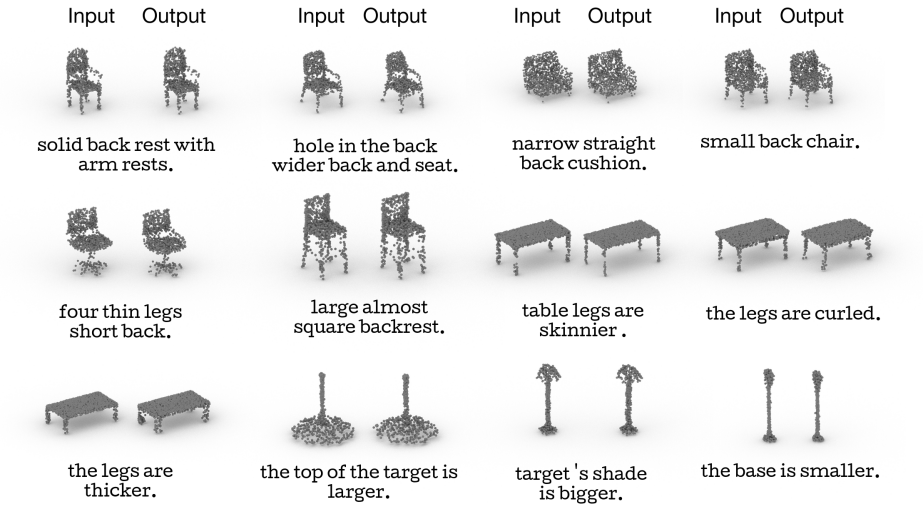

Fig. 8: Qualitative examples of language-assisted edits in the **PC-AE** latent space.

Figure 9 shows **failure cases** of editing with ChangeIt3DNet via an SGF backbone, complementing the qualitative results presented in the Main paper (see Main Figure 4). As we can see in the examples presented in this figure our language-assisted editing task and approach can suffer from a variety of different challenges. From editing 'entanglement' (changing more elements than

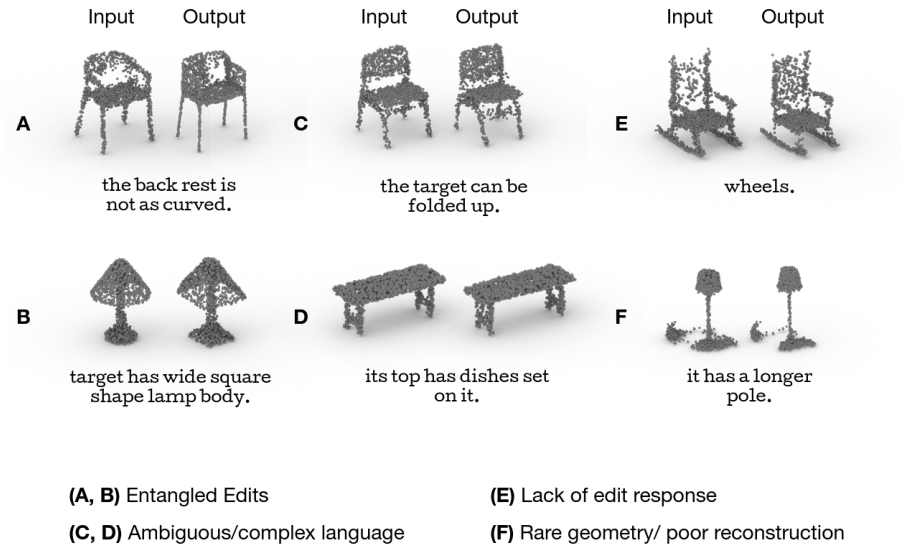

Fig. 9: Qualitative examples demonstrating **failure cases** of language-assisted edits in the SGF latent space via ChangeIt3DNet. Oftentimes the resulting edits successfully apply the requested command, but also affect other geometric elements not mentioned and expected to change in language (*Entangled Edits*). The method can also struggle when the input language is *complex or ambiguous* in meaning. Finally, there are cases where the requested edit is difficult to be discovered in the latent 'neighborhood' of the underlying object (*wheels* for a reclining chair); or, the default output shape-reconstruction of the AE is poor (example F) making the edit task hard in ab initio.

those described in the input language), to a more fundamental inability to understand and apply ambiguous or complex language, or, more basic problems resulting from the a-priori inability of the 3D backbone to reconstruct an object (especially when its geometry is rarer within the dataset). These results point towards several promising directions our method can be improved in future work. Ranging from using more disentangled, part-aware latent spaces e.g, [5,10,14]; to altogether lifting the premise of operating inside a latent space, and instead work directly on the primal object space with appropriate deformation-aware handles e.g., [16,9].

D.2 Retrieval-Oriented Results:

Lastly, Figure 10 shows a few qualitative examples for the retrieval-based baseline, where the nearest neighbor of shapes seen during training is retrieved based on the predicted edit vector predicted within the ShapeGF latent space. We find that though the retrievals oftentimes satisfy the language instruction, it does so without preserving other parts of the input object (a good example of this can be seen in the first chair in the middle column of Figure 10). This is a testament to

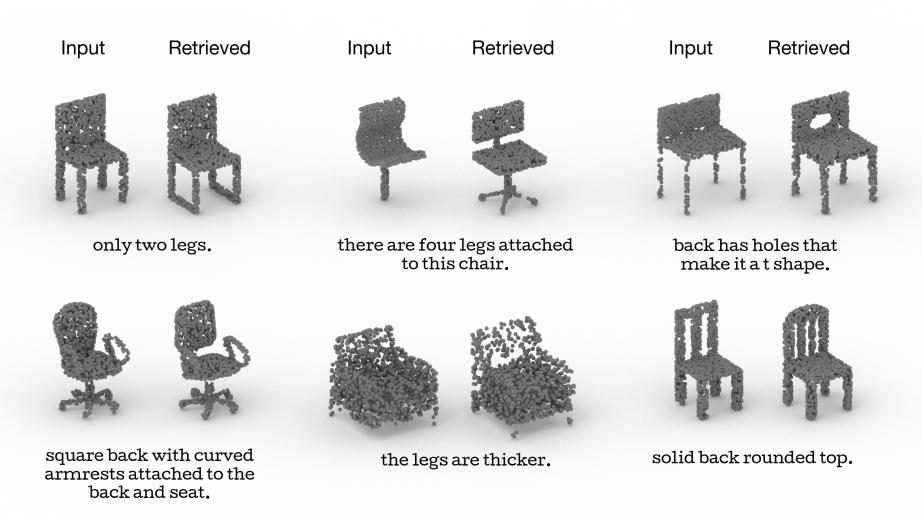

Fig. 10: Qualitative examples of language-driven **retrieval** in the ShapeGF latent space.

the sparsity of the object space with respect to fine-grained language-articulable differences between shapes.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.J.: Learning representations and generative models for 3D point clouds. In: International Conference on Machine Learning (ICML) (2018)
2. Achlioptas, P., Fan, J., Hawkins, R.X., Goodman, N.D., Guibas, L.J.: ShapeGlot: Learning language for shape differentiation. In: International Conference on Computer Vision (ICCV) (2019)
3. Cai, R., Yang, G., Averbuch-Elor, H., Hao, Z., Belongie, S.J., Snavely, N., Hariharan, B.: Learning gradient fields for shape generation. In: European Conference on Computer Vision (ECCV) (2020)
4. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
5. Dubrovina, A., Xia, F., Achlioptas, P., Shalah, M., Groscot, R., J., G.L.: Composite shape modeling via latent space factorization. International Conference on Computer Vision (ICCV) (2019)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation (1997)
7. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (ICML) (2015)
8. Koo, J., Huang, I., Achlioptas, P., Guibas, L.J., Sung, M.: PartGlot: Learning shape part segmentation from language reference games. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2022)
9. Liu, M., Sung, M., Mech, R., Su, H.: DeepMetaHandles: Learning deformation meta-handles of 3d meshes with biharmonic coordinates. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
10. Mo, K., Zhu, S., Chang, A.X., Yi, L., Tripathi, S., Guibas, L.J., Su, H.: PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
11. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: International Conference on Machine Learning (ICML) (2010)
12. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3D classification and segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
13. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. Computing Research Repository (CoRR) abs/2103.00020 (2021)
14. Schor, N., Katzir, O., Zhang, H., Cohen-Or, D.: CompoNet: Learning to generate the unseen by part synthesis and composition. International Conference on Computer Vision (ICCV) (2019)
15. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research (JMLR) (2014)
16. Sung, M., Jiang, Z., Achlioptas, P., Mitra, N.J., Guibas, L.J.: DeformSyncNet: Deformation transfer via synchronized shape deformation spaces. In: ACM SIGGRAPH Asia (2020)

17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NeurIPS) (2017)