CS766 - Midterm Report: Shape Editing using Language Instructions

Shape Shifters

March 2024

1 Introduction

Utilizing tools like 3D scanners and photogrammetry software greatly facilitates the reverse engineering process, enabling the replication or rapid prototyping of objects through 3D printing, benefiting fields such as product design and architectural modeling. However, these tools are often not without some level of inaccuracy, generating fragmented point clouds due to noise or device calibration issues. The subsequent manual cleanup in software platforms like SolidWorks®, Onshape®, Blender®, etc. demands considerable time and a steep learning curve for users to master their intricacies.

In recent years, there has been interest in utilizing language a universally understood means of communication—to generate e.g. [1], segment e.g. [2], and manipulate e.g. [3] 3D models. We are interested in better understanding the application of 3d object manipulation to improve the precision of point clouds and streamlining the design process for creating diverse functional object variations tailored to a wide range of design applications. Specifically, we plan to explore utilizing the ShapeTalk dataset of "discriminative utterances produced by contrasting the shapes of common 3D objects" [3] and the ChangeIt3D framework which utilizes a "3D generative model of shapes" [3] to modify 3D point clouds based on the description a user provides on how to deform the model and analyze how it performs on broken point clouds due to noise or imperfect calibration.

The Change-It-3D architecture includes 3-phases, i) encode the 3D shape into a latent vector using an auto encoder, ii) a shape editing module which proposes edits to the shape based on the language instruction and iii) a discriminator or listener module which identifies whether shape-editing module produce the necessary changes in the shape latent vector. In this project, we will be exploring the discriminator and shape-editing module. For the first phase, we will present initial results on training the listener architecture on different choice of text-encoders and auto-encoders can impact listener accuracy. For the second phase, we will utilize a pre-trained listener and and auto-encoder architecture to modify an existing 3D shape. The shape editing module would receive a text prompt along with a pretrained embedding from the autoenocder which will

be modified to generate the target shape. This module would be trained using the frozen listener architecture used in the first stage of training. We include a figure from [3] representing the complete workflow of ChangeIt3D in **Figure 1**

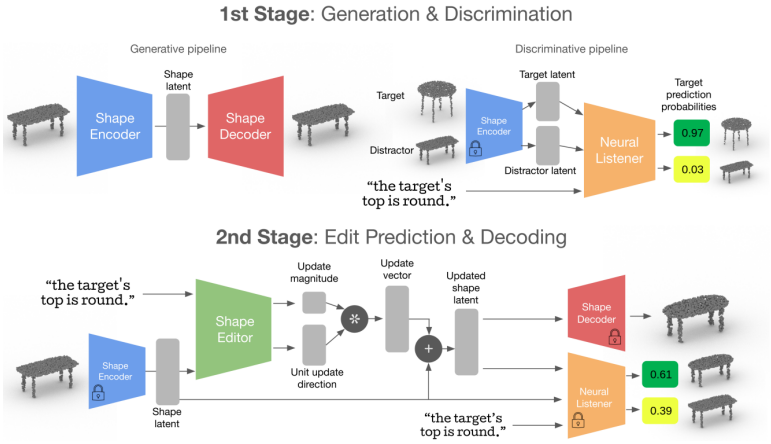


Figure 1: The overall pipeline of changeit3d to modify the structure of a shape based on the 3D text query. i) The shape auto-encoder is trained to produce a latent vector of the shape and is trained with reconstruction loss. ii) The discriminator is tasked with classifying the difference between two 3D shapes and identify which one better matches the text-query . iii) The shape editing module receives the shape embedding and the text to modify the shape which used to propose a vector edit to the shape.[3]

2 Related Work

Over the last few-years, the deep-learning community has witnessed an increasing interest in connecting different modalities of representation to increase the richness of our interaction with the world.

CLIP [4] was one of the first methods to connect language and images by proposing a common embedding space for the two modalities. Subsequent works like DALL-E [5] extend ideas from CLIP to apply it to generating and modifying images based on a text prompt.

A parallel trajectory of research has emerged in the realm of three-dimensional (3-D) modeling, exemplified by projects like CLIP-Forge [6], Dream-Fields [7], and Shape-Crafter [8], which built text conditioned 3-D generation models. However, these endeavors have primarily focused on the direct generation of 3-D shapes, they have not placed significant emphasis on nuanced shape editing, particularly regarding specific components of the shapes. Although some efforts have been made to establish connections between language and the structural features of 3-D shapes, leveraging localization techniques for direct shape

editing remains an under-explored avenue.

In this project, we explore a way of editing 3-D shapes using a multi-modal model to propose edits to the 3-D shapes. Similar to multi-modals like CLIP, we propose an edit in the latent space and utilize a shape-decoder to recover the shape in the original 3-D space.

3 Summary of Progress

For training our model we utilize the Shape-Talk dataset [9] which contain shapes from the Shape-Net dataset. The complete shape-talk dataset contains 30 object classes with over $536K$ constrastive utterances. To make training feasible on an cloud source computing environment like Google-Colab we use a sub-sample 4 *classes* from this dataset across the lamp, vase, mug, and bottle classes. This reduces the size of our dataset to 4882 utterances, 9979 unique communication contexts, and 3835 point clouds. We also made this choice to accommodate our experiments in the real-world exploring shape modifications on point clouds using a 3D scanner. For the purposes of this report, we conduct an analysis of distribution of data across the 4 categories and report the performance of training on the dataset which we planned to use for our 'real-world' experiments. We also have analyzed the dataset and splits used in the various pretrained models outlined in the initial ChangeIt3D paper [3]. In the rest of the document we will discuss the specifics about the dataset, training, initial results, and future steps.

4 Experiments

4.1 Dataset Analysis

Terminology: Throughout this analysis we will refer to **communication context** as a pair of objects provided in an image based dataset. The left-most object in the image set is referred to as the *source* or *distractor* object and the right-most is referred to as the *target* object. These communication contexts were presented to **annotators** who, with a given prompt, generate referential language to distinguish the target from the distractor, i.e. what is different about it from the source. Together the referential language and the communication contexts comprise the core dataset used for training the neural listener for the discriminative component of the framework, which, given an utterance and a pair of input shapes, assigns high probability to the most utterance-compatible shape within the pair [3].
To further contextualize this **Figure 2** illustrates an example distractor (left) and a target (right). One annotator provided the referential language *"The target has a distinctive top shaped like a pointy crown."* based off of this communication context.

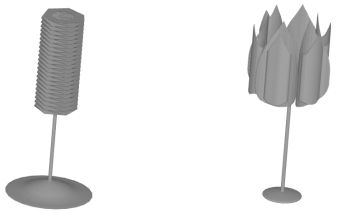


Figure 2: Distractor Lamp Object (Left) and Target Lamp Object for the query "The target has a distinctive top shaped like a pointy crown"

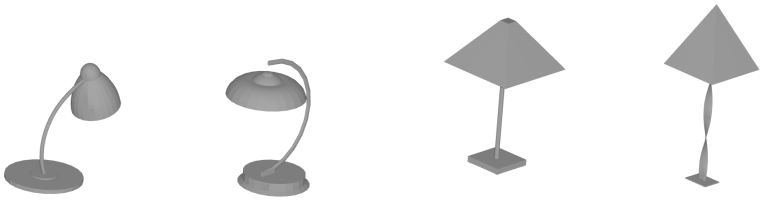
We will also refer to autoencoders (AE) which is a type of neural network architecture that learns to reproduce its input, which in our scenario would be the 3D point cloud or the image of an object. In the context of this work we utilize 3-pretrained-shape generators. A **point cloud autoencoder (PC-AE)** [10] which down-samples point clouds through applying a number of 1-D convolutions and then using a decoder to regenerate the form, a **residual networks (ResNet101)** [11] which seeks to circumvent the issue of noncontinuous sampling of 3d point data through learning a gradient field from a set of sampled points from the shape of its log-density, and , **implicit neural methods, (ImNet)** [12] which uses a binary classifier to dictate if a point lies on the object or not.

Dataset Structure: The dataset is comprised of 3 key components: images depicting a communication context, utterances provided by annotators, and point cloud representations used for training the autoencoder (AE) for 3D shape representation.

The images, representing the communication contexts, have different source databases including ShapeNet [13], ModelNet [14], and PartNet [15]. The different sources serve to help diversify the type of objects, i.e. classes used.

A given communication context is organized and assigned a specific class, the class represents what type of object is getting evaluated, e.g. lamp, shelf, mug, etc. The contexts are also assigned a level of difficulty: Easy vs Hard. A communication context is considered *easy* if the two objects in the context have less visually similar pairs and considered *hard* if the objects have high visual similarity. 48.6 percent of contexts are classified as hard. **Figures 3a and 3b** display a randomly selected hard vs easy categorized communication context from the lamp class.

The language dataset component was built using input from 2161 annotators who in total entered 536596 utterances across 73799 distinct communication contexts. An utterance is the word-based input provided by the annotators, indicating key shape based differences between the target and distractor. Each



(a) Hard Communication Context. This was paired with an utterance of "The shade is hanging from a wire"

(b) Easy Communication Context. This was paired with an utterance of "The target has a helix shaped lamp rod."

Figure 3: Sample Hard vs Easy Communication Contexts from the Lamp Class in Our Dataset

(a) LSTM: Test (b) LSTM:Validation (c) LSTM:Train

(d) Transformer: Test (e) Transformer:Validation (f) Transformer: Train

Figure 4: Training/Validation and Test Accuracy Curves for the Listener an Metric Comparison of both the LSTM and Transformer on our custom training dataset. Notice that both the architectures with default parameters overfit during training. We plan to investigate the parameters to identify the best performing model for the second phase of the project.

word in an utterance is indicated in the dataset as a token. Across all utterances there were 9251 unique tokens and an average of 6.28 tokens per utterance. Annotators were instructed to report between 3 and 5 utterances in order of most noticeable difference to least noticeable difference [9]. The order in which the utterance was input dictates its assigned *saliency* in the dataset, with 0 being what was listed first by the annotator for the context and should be the most salient/apparent difference. On average per communication context annotators documented 4.12 utterances. There is a fairly even split of utterances between hard vs easy contexts; 49.5 percent of utterances were for hard contexts. Utterances for hard contexts had on average 6.49 tokens while easy contexts had on average 6.08 contexts.

4.2 Training Details

As highlighted in Figure 1, there are three 3 primary training phases of the pipeline:

1. Training the generative 3D shape networks/autoencoders using traditional reconstruction losses, we may refer to this as the Generative pipeline or Module-1.

2. Training the Neural Listener using the cross entropy loss to identify which of two 3D shapes matches a text query. In this phase we use the encoded latents of the target and distractor generated from the previous training in the generative pipeline.

3. Train the ShapeEditor module to edit shapes using the pretrained auto-encoder and discriminative neural listener.

We have access to all utterances for the contexts based on these image pairings that were used with the pretrained neural listener. However, because we are limiting our training to 4 classes this brings down the total number of utterances available to *4882.* Training, testing, and validation did not include examples where utterences where long, i.e. excluded utterences of 12 or more tokens. Rare tokens in the dataset (words that appear in fewer than 2 utterances) were replaced in the training split of the data with a special token: $< UNK >$ [9].

For point cloud data, we have a subset of pointclouds from the initial dataset consisting of data across 10 classes. A break down of available point cloud files is noted in **Appendix A, Figure 10.** However, we will focus on just the data in the vase, mug, bottle, and lamp classes yielding 3835 point clouds, see **Figure 5b** for specifics.

Module-1: Shape-Encoder: PC-AE [10], ResNet101 [11], and ImNET [12] were all trained with the same point cloud files containing the (x,y,z) coordinates, normalized between -1 and 1, representing the shape/surface of object across all 30 classes. Split details are noted in **Appendix A, Table 1** and sample visualizations of point clouds are included in **Appendix A, Figure**

11. These auto encoders are trained on all the shapes in the ShapeTalk dataset containing $> 50,000$ shapes. Considering the large scale of data required for training this module we utilize a pre-trained auto-encoder for our experiments.

Module-2: Shape Discriminator/Listener: The system uses a variant of ShapeGlot [9] that does not take additional contextual information into account when processing language and shapes. It uses 256-dimensional latent representations of the 3D shapes generated by the one of the three shape encoders: PC-AE [10], ResNet101 [11], and ImNET [12].

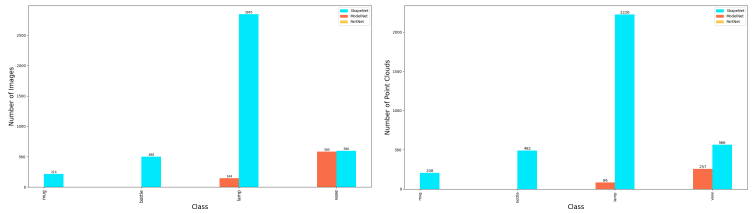
Hyper-parameters and Model Components: 128-dimensional vectors are used as word embeddings to represent each token in the input utterances. The Visual Projection Layer transforms the 256D latent space representations into a 128D space using a two-layer MLP with 128 neurons each to make them compatible with the outputs of language encoder [16]. All layers use batch normalisation and ReLU activation function. Finally these two layers are regularized with 0.2 dropout probability. A multi hidden-layer MLP-based classification-head is used. It operates on a multimodal concatenated representation of the visual signal and the latent representation of each utterance. The layers use fully connected operations followed by ReLU activation and batch normalization. The bias free output neurons are also implemented with a fully connected layer.

The transformation of the input tokens to the single linguistic latent space representation is experimented with two methods based on a LSTM and the attention based transformer architecture which are described below. We chose the following architectures as they explore two constrastive mechanisms of encoding textual input which can influence future design decisions.

- **Long short-term memory (LSTM):** The LSTM based model, as used in ShapeGlot [9], is used for language encoding to capture sequential dependencies in the language data. The inputs are processed sequentially with the tokens being processed in the order they occur in the sentence. A unidirectional LSTM cell with 128D hidden states is initialized with the visual representation. The LSTM is run for each shape-description pair, and the maximum values of its output vectors are collected per token and passed to the classification head to produce a single output representing the probability of the first shape matching the text query.

- **Transformer:** The LSTM is replaced with a Transformer based encoder [3] as described in PartGlot [17]. The transformer, in contrast to the LSTM, can process all inputs and once. It has 128 dimensional encodings with two layers, each having two attention heads. This multi-headed mechanism enables it to use positional encoding and can process the entire sequence at once utilising an attention mechanism, weighing different parts of the input sequence. The output at the end of the sequence is concatenated with the visual representation and passed to the classification

head to produce a single output representing the probability of the first shape matching the text query



(a) Images (b) Point Cloud

Figure 5: Data used in testing, training, and validation splits

4.3 Initial Results

We compare the results of two of the backbones - i) The baseline LSTM architecture and ii) The Transformer architecture for encoding a sentence. The results of training are reported in Figure 4. In the training distribution we notice that both architectures perform well with an accuracy of 80%. However, the smaller size of our dataset leads to over-fitting which results in poor performance on the test dataset. Further, by comparing the testing accuracy between the LSTM and the Transformer it appears that the transformer model overfits more than the LSTM model on the same dataset. Although this could be traced to the larger number of parameters in the transformer, sequential processing in LSTM and can be corrected through a better choice of hyper-parameters, this merits further investigation before the final report. Within each of the curves, all three of the encoders (Im-NET, PC-AE and ResNet101 perform equally well with no significant variation in the training and testing curves.

5 Next Steps

5.1 Investigating the Listener Architecture

Our trained listener module faces severe over-fitting due to the sub-sampled data-distribution used with only 4-classes. To better bridge the gap between the training and validation distribution we'll experiment with weight regularization, dropout and other techniques to better combat over-fitting.

5.2 Shape-Editing Module

Our immediate next step is to investigate the shape-editing module to modify the 3D shape using the text query using a pre-trained listener architecture. We'll utilize the 3D shape along with the point cloud embedding and utilize

utterances from Shape-Talk dataset to modify the 3-D shape. This module would be trained in a generative fashion using adversarial loss to fool the frozen listener module trained in the previous stage. We'll experiment with our trained listener module and the pre-trained listener module trained across all classes by the authors of [3].

Real World Experiments In order to test the the application of ChangeIt3D framework for real-world applications, 3d scan data was collected. The 3D scan was collected using the Creaform Handyscan 700, used courtesy of the UW Makerspace. A 3D printed vase shown in Figure 6, was selected from the UW Makerspace because this is one of the object classes provided by the ShapeTalk data set.

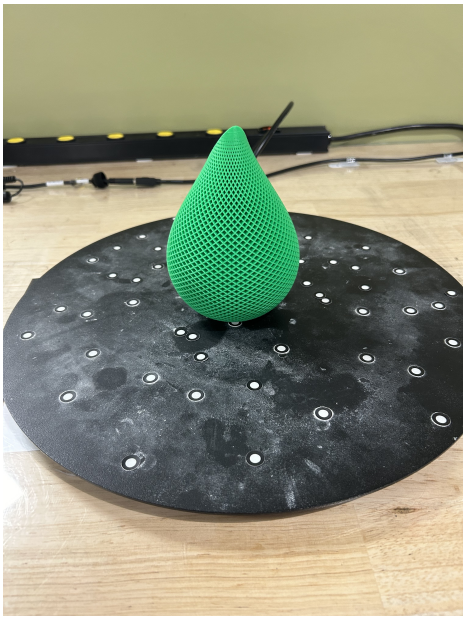


Figure 6: Scanning set up for a 3D printed vase at the UW Makerspace.

A single scan of the 3D printed vase was captured as shown in Figure 7. The Creaform Handyscan 700 is a very accurate scanner with a accuracy up to 0.04 mm. However, the geometry of objects can lead to holes and other imperfections of the model during the scanning process. Notice the holes within the cross hatched pattern in Figure 7 and the large hole on the bottom of the vase. These are due to the scanner not being able to have a direct line of sight to points on the surface of the object.

In order to use this data with the ChangeIt3D framework, the Mesh needs to be converted to a point cloud, down sampled and scaled to [-1, 1] which can be seen in Figure 8. The smaller holes due to the crosshatch pattern are no

longer visible, but the over all shape of the vase remains with the large hole on the bottom. We believe this is sufficient to test the framework.

Once the entire ChangeIt3D framework has been trained, we will integrate this data and other 3d scans from the vase, bottle, mug, and lamp classes. We aim to determine if the ChangeIt3d framework is an effective language based text editor on real world data.

6 Contributions by Team-Member

We list out the contributions by team member. The contributors are listed in no particular order.

- **Keshav**
  1. Major: Module-2: Shape Discriminator/Listener, Training, Cloud Compute Setup
  2. Minor: Training Results
- **Abhinav**:
  1. Major: Experiment formulation/Training Results, Introduction,
  2. Minor: Dataset-Analysis, Related Work
- **Laura**
  1. Major: Dataset Analysis, First half of Training Details up to 'Module-2: Shape Discriminator/Listener ' paragraph, Apendix
  2. Minor: Summary of progress, Introduction
- **Kevin**
  1. Major: Next Steps, References

References

[1] R. Liu, R. Wu, B. V. Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick, *Zero-1-to-3: Zero-shot one image to 3d object*, 2023. arXiv: 2303.11328 [cs.CV].

[2] A. Abdelreheem, I. Skorokhodov, M. Ovsjanikov, and P. Wonka, *Satr: Zero-shot semantic segmentation of 3d shapes*, 2023. arXiv: 2304.04909 [cs.CV].

[3] P. Achlioptas, I. Huang, M. Sung, S. Tulyakov, and L. Guibas, "ShapeTalk: A language dataset and framework for 3d shape edits and deformations," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

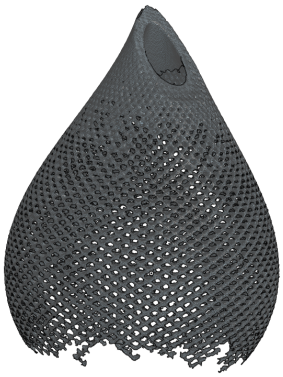


Figure 7: Captured stl of Figure 6 using the Creaform Handyscan 700. Notice the major hole on the bottom of the vase.

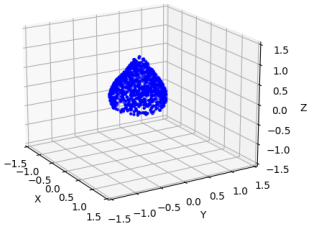


Figure 8: Point Cloud with 1000 points sampled from the stl shown in 7. The complex smaller holes shown in the Figure 7 are no longer visible due to down sampling.

[4] A. Radford, J. W. Kim, C. Hallacy, *et al.*, *Learning transferable visual models from natural language supervision*, 2021. arXiv: 2103.00020 [cs.CV].

[5] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, *Hierarchical text-conditional image generation with clip latents*, 2022. arXiv: 2204.06125 [cs.CV].

[6] A. Sanghi, H. Chu, J. G. Lambourne, *et al.*, *Clip-forge: Towards zero-shot text-to-shape generation*, 2022. arXiv: 2110.02624 [cs.CV].

[7] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole, *Zero-shot text-guided object generation with dream fields*, 2022. arXiv: 2112.01455 [cs.CV].

[8] R. Fu, X. Zhan, Y. Chen, D. Ritchie, and S. Sridhar, *Shapecrafter: A recursive text-conditioned 3d shape generation model*, 2023. arXiv: 2207.09446 [cs.CV].

[9] P. Achlioptas, J. Fan, R. Hawkins, N. Goodman, and L. Guibas, "ShapeGlot: Learning language for shape differentiation," in *International Conference on Computer Vision (ICCV)*, 2019.

[10] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, *Learning representations and generative models for 3d point clouds*, 2018. arXiv: 1707.02392 [cs.CV].

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv:1512.03385*, Dec. 2015, Microsoft Research. "arXiv": 1512.03385 (cs.CV).

[12] Z. Chen and H. Zhang, *Learning implicit fields for generative shape modeling*, 2019. arXiv: 1812.02822 [cs.GR].

[13] A. X. Chang, T. A. Funkhouser, L. J. Guibas, *et al.*, "Shapenet: An information-rich 3d model repository," *CoRR*, vol. abs/1512.03012, 2015. arXiv: 1512.03012. [Online]. Available: http://arxiv.org/abs/1512.03012.

[14] Z. Fang, X. Li, X. Li, S. Zhao, and M. Liu, *Modelnet-o: A large-scale synthetic dataset for occlusion-aware point cloud classification*, 2024. arXiv: 2401.08210 [cs.CV].

[15] K. Mo, S. Zhu, A. X. Chang, *et al.*, "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding," *CoRR*, vol. abs/1812.02713, 2018. arXiv: 1812.02713. [Online]. Available: http://arxiv.org/abs/1812.02713.

[16] P. Achlioptas, I. Huang, M. Sung, S. Tulyakov, and L. Guibas, *Changeit3d: Language-assisted 3d shape edits and deformations*, Supplemental Material, 2020.

[17] J. Koo, I. Huang, P. Achlioptas, L. J. Guibas, and M. Sung, "Partglot: Learning shape part segmentation from language reference games," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

A Supplemental Data and Visualizations

Object Class	3D Shape Generators			Neural Listener			ChangeIt3D		
	Train	**Test**	**Val**	**Train**	**Test**	**Val**	**Train**	**Test**	**Val**
airplane	2313	272	137	37551	4369	2237	38004	3829	2298
bag	118	14	7	2765	330	165	2805	303	130
bathtub	564	66	34	10142	1161	615	9935	1098	563
bed	634	75	38	12581	1493	750	12133	1462	908
bench	1408	166	83	21665	2568	1277	21608	2542	1194
bookshelf	693	82	41	13619	1610	805	13556	1528	852
bottle	418	49	25	4978	581	296	4896	560	264
bowl	197	23	12	2299	268	141	2233	199	130
cabinet	209	25	12	3246	398	188	3154	414	238
cap	176	21	11	3167	382	199	3238	340	138
chair	5616	661	331	62973	7512	3692	62392	7540	3932
clock	492	58	30	7807	913	479	7764	928	409
display	997	117	60	12451	1453	737	12475	1396	712
dresser	1436	169	85	28437	3351	1687	27316	3375	1961
faucet	543	64	33	8547	1009	520	8385	1108	468
flowerpot	529	62	32	8360	982	506	8127	1008	458
guitar	640	75	38	9938	1170	591	9876	1128	653
helmet	118	14	8	1403	167	96	1378	207	77
knife	360	42	22	5740	669	351	5710	612	364
lamp	1965	231	116	45583	5323	2675	44913	4919	2709
mug	176	21	11	2103	250	132	2142	192	137
person	79	9	6	2359	270	180	2436	223	150
pistol	256	30	16	4000	463	247	3997	414	232
plant	238	28	15	2804	323	179	2588	329	202
scissors	67	8	5	1201	144	89	1228	117	89
skateboard	129	15	8	2296	265	141	2192	241	188
sofa	2605	306	154	43923	5177	2558	43985	4964	2386
table	6949	818	409	74960	8912	4427	74390	8411	4201
trashbin	291	34	18	4530	520	270	4549	481	245
vase	699	82	42	11074	1308	665	10556	1366	596
Total	**30915**	**3637**	**1839**	**452502**	**53341**	**26895**	**447961**	**51234**	**26884**

Table 1: Counts of Inputs for Training, Testing, and Validation for the the 3D Shape Generators (PC-AE, ResNet101, and ImNet), for the Neural Listener for the Discriminative pipeline, and ChangeIt3D for the

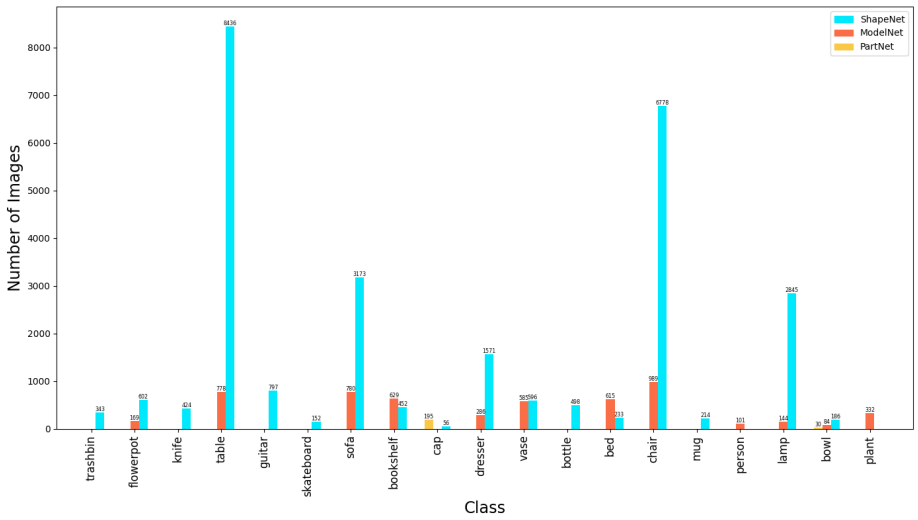


Figure 9: Number of Images in our Available DataSet by Class and Data Source

The image data that we have access to is comprised of 19 classes across all 3 data-sources (i.e. ShapeNet, ModelNet, and PartNet). However, for training purposes we are using contexts generated from image pairs in just the lamp, mug, bottle, and vase classes.

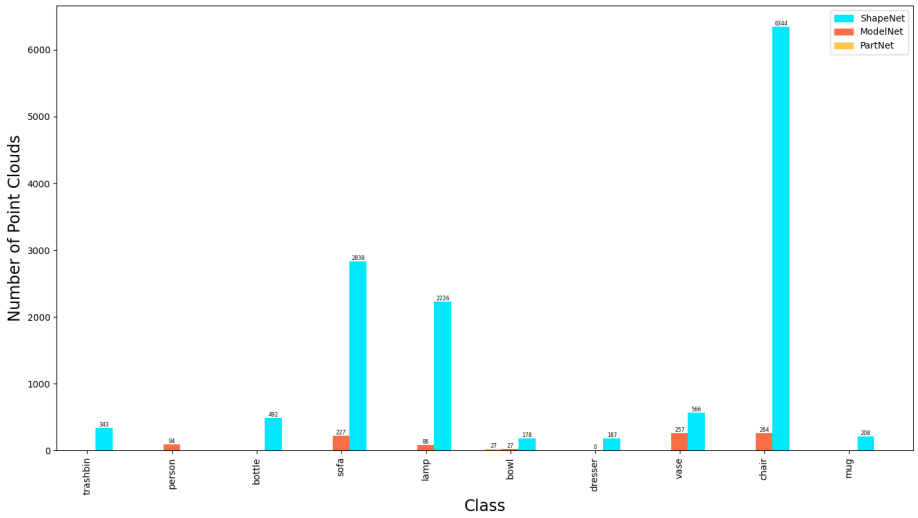


Figure 10: Number of Point Cloud Files in our Available DataSet by Class and Data Source

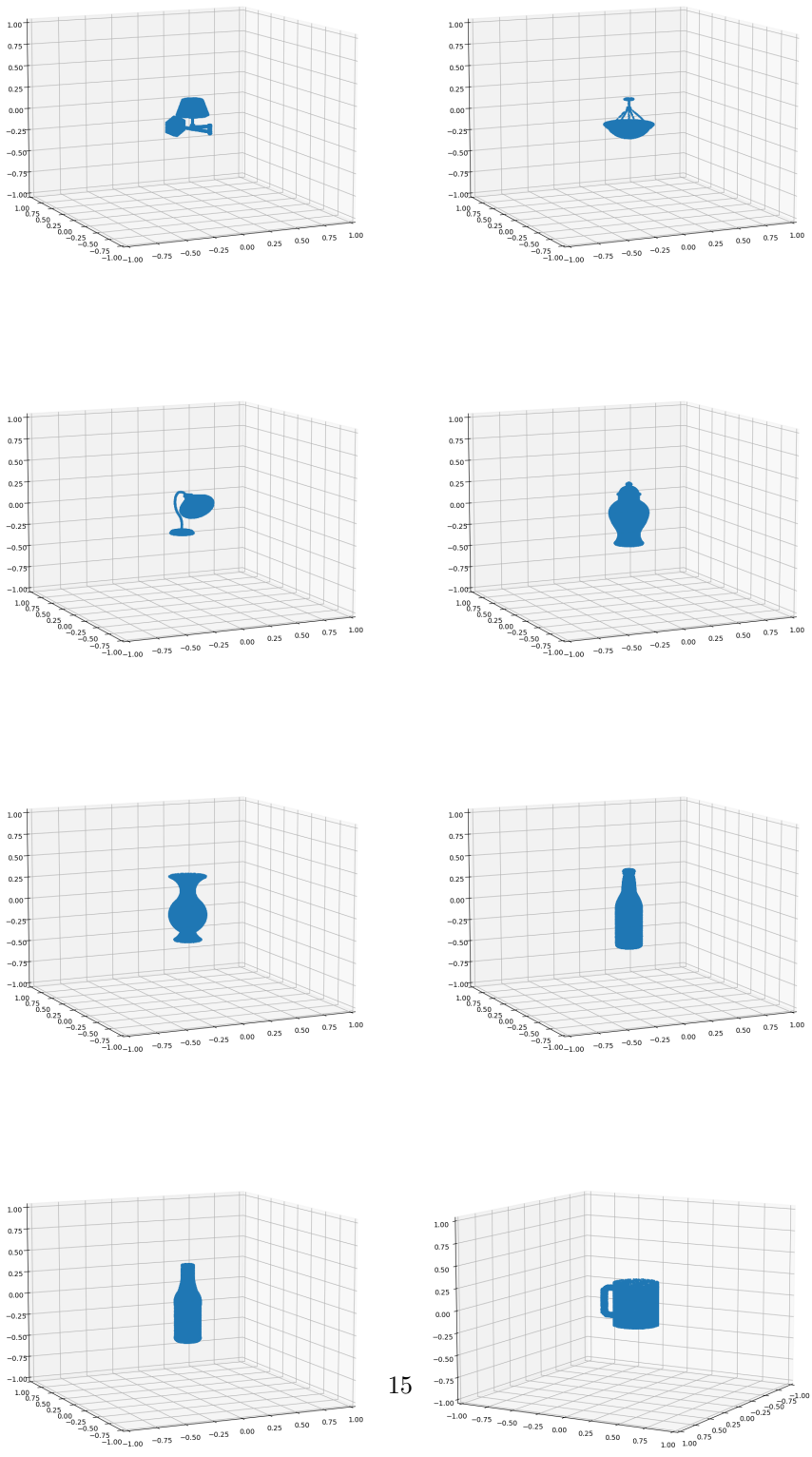


Figure 11: Visualizations of Sample Point Clouds from the Lamp, Vase, Bottle, & Mug Classes