Learning Shape-to-Shape Transformation

by

Kangxue Yin

B.Eng., Chang'an University, 2012

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2020

Approval

Name:	**Kangxue Yin**
Degree:	**Doctor of Philosophy (Computing Science)**
Title:	**Learning Shape-to-Shape Transformation**
Examining Committee:	**Chair:** Angel Chang Assistant Professor
	Hao(Richard) Zhang Senior Supervisor Professor
	Hui Huang Supervisor Adjunct Professor
	Daniel Cohen-Or Supervisor Adjunct Professor
	Manolis Savva Internal Examiner Assistant Professor School of Computing Science
	Matthias Zwicker External Examiner Professor Department of Computer Science University of Maryland
Date Defended:	**14 May 2020**

Abstract

Many problems in computer graphics and geometric modeling, e.g., skeletonization, surface completion, and shape style transfer, can be posed as a problem of shape-to-shape transformation. In this thesis, we are interested in learning general-purpose shape transform, e.g., between 3D objects and their skeletons, between chairs and tables, and between letters of two different font styles, etc. With a point-based shape representation, we explore the problem of learning general-purpose shape-to-shape transformation, under two different settings: i). having shape-level supervision, ii). unsupervised.

We present P2P-NET, a deep neural network, for learning shape transform under shape-level supervision. It is trained on paired shapes from the source and target domains, but without relying on point-to-point correspondences between the source and target point sets(i.e., point-level supervision). The architecture of the P2P-NET is that of a bi-directional point displacement network, which transforms a source point set to a prediction of the target point set with the same cardinality, and vice versa, by applying point-wise displacement vectors learned from data.

For an unsupervised setting, we introduce LOGAN, a deep neural network aimed at learning general-purpose shape transforms from unpaired shape domains. It consists of an autoencoder to encode shapes from the two input domains into a common latent space, where the latent codes are overcomplete representations for shapes. The translator is based on a generative adversarial network (GAN), operating in the latent space, where an adversarial loss enforces cross-domain translation while a feature preservation loss ensures that the right shape features are preserved for a natural shape transform.

We conduct ablation studies to validate each of our key designs and demonstrate superior capabilities in shape transforms on a variety of examples over baselines and state-of-the-art approaches. Several different applications enabled by our general-purpose shape transform solutions are presented to highlight the effectiveness, versatility, and potential of our networks in solving a variety of shape-to-shape transformation problems.

Keywords: Shape transform; shape-to-shape transformation; domain translation; shape-to-shape translation; geometric deep learning

Acknowledgements

Obtaining my doctoral degree was an unforgettable and wonderful journey. With the help and support of friends and colleagues along the way, these past 5 years have been challenging but immensely rewarding.

First and foremost, I would like to thank my senior supervisor, Dr. Richard Zhang, for his support, encouragement, and guidance throughout my PhD program. He has encouraged me to work on impactful and challenging research topics, and has taught me how to find good research problems. Richard is a responsible and patient supervisor, as well as an insightful researcher, all of which I hope to one day emulate.

I also want to thank my co-supervisors, Dr. Hui Huang and Dr. Daniel Cohen-Or, for the help and guidance they provided me—especially in the early stages of my PhD research. I started my collaboration with my three mentors Richard, Hui and Daniel even before I started my PhD. They have taught and trained me in different aspects, and have collectively made me become a mature researcher.

In addition, I would like to thank the rest of my thesis committee: Dr. Manolis Savva, Dr. Matthias Zwicker, and Dr. Angel Chang, for their insightful comments, suggestions, and questions.

I am appreciative of the great research fellows and friends in GrUVi lab I have gotten to know, particularly: Zhiqin Chen, Wallace Pinto Lira, Akshay Gadi Patil, Fengen Yu, Zili Yi, Johannes Merz, Manyi Li, Chenyang Zhu, Rui Ma, Ali Mahdavi-Amiri, Jon Liu, Changqing Zou, Han Liu, Xi Ao, Jaime Vargas, Warunika Ranaweera, Zeinab Sadeghipour Kermani, among many others.

Last but not least, I give my special appreciation to my fianceé, Jingwen Zhang, the most beautiful lady in the world, for the companionship, support, and trust she gave me during the most difficult times of obtaining my PhD. Thank you for always being there for me.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem statement and motivation

Shape transform is one of the most fundamental and frequently encountered problems in computer graphics and geometry processing. Many tasks, e.g., skeletonization, surface completion, and shape style transfer, can be considered as a shape transform problem. With much interest in geometric deep learning in the graphics community today, it is natural to explore whether a machine can learn shape transforms, either under a supervised setting or an unsupervised setting. Specifically, given two sets of shapes from two different domains $\mathscr{X}, \mathscr{Y}$ with some commonalities, can machine learn to transform a shape X of domain $\mathscr{X}$ into a shape Y belonging to domain $\mathscr{Y}$ while preserving common shape attributes shared by the two domains? Some attempts of the community in learning shape transforms have been made to surface detail transfer [1], decorative styles [2], and piece-wise rigid transformations of 3D objects [3]. Different than these works, the goal of this thesis is to develop *general-purpose* methods for learning shape transforms.

A similar domain translation problem in computer vision is general-purpose image-to-image translation, which has drawn much interest in computer vision and computer graphics. However, most of the successes of image-to-image translation [4, 5, 6, 7] have been achieved only on transforming or transferring stylistic image features or textures, not shapes. These image translation tasks include translating between night and day images, summer and winter images, different artistic styles, etc. Recently, later than the development of the methods for learning shape transform in this thesis, geometric deformation for general-purpose image translation is realized by [8, 9]. These methods can jointly warp images and transfer style from one domain to another. However, they are not designed for dealing with significant shape transformations, such as turning a chair into a table, or shape skeletonization.

Without prior assumption on the properties of the shape transforms, our deep neural networks are expected to learn highly-customized features for different shape domains. These shape transforms span a wide spectrum of applications. As shown in Figure 1.1, some examples include transforms between 3D objects and their skeletons, incomplete and

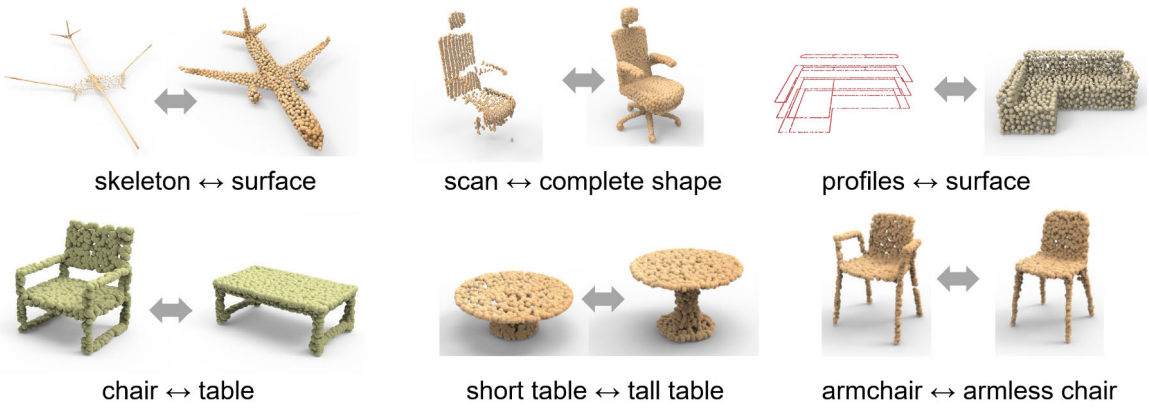


Figure 1.1: Examples of shape transforms between two domains.

completed object scans, 2D cross-sectional profiles and 3D shapes, chairs and tables, etc. Learning general-purpose shape transform will provide a reusable solution for these types of shape-to-shape transformation problems, or provide a good initialization to customized solutions.

Before starting to explore the problem of learning general-purpose shape transform, a shape representation should be selected. Point set, as a shape representation, has the advantages of simplicity and versatility. Compared to volumetric representations or triangle meshes, it is more efficient and natural in representing shapes of mixed dimensionalities, such as meso-skeletons, partial scans, and object contours. With the emergence of affordable 3D acquisition devices, point cloud data is widely captured, accumulated, and processed by many useful techniques, e.g., in point cloud filtering [10], resampling [11], and surface reconstruction [12, 13], among others. Furthermore, the deep learning techniques for learning shape features from point clouds have been sufficiently developed and will provide us useful building blocks for learning shape transforms from two sets of shapes represented by point clouds.

In this thesis, we present two designs of deep neural networks for learning domain translation of shapes represented by point sets. Our goal is to develop *general-purpose* neural networks that are capable of learning shape transformations between point sets from two domains. The network is trained to map point sets from one domain to another, where the point sets can be in 2D or 3D Euclidean spaces. As we aim for general-purpose neural networks, no part of the networks should be tailor-made to specific transformation tasks. Moreover, we should not alter the network architecture when dealing with different pairs of transformation domains.

Under a supervised setting, paired shapes of two domains are required. We are interested in designing deep neural networks which will be trained by paired point sets with some commonalities, but not by their point-wise mapping. The commonality can be that the two point sets were sampled from the same shape(e.g., scan and complete surface), or the two

point sets are similar in terms of some local or global geometric features(e.g., chair and table). In many examples, there is no clear point-to-point correspondence between the two point sets. Not requiring point-wise correspondences can significantly expand our capacity to collect training data.

Under an unsupervised setting, training shapes from two domains are provided without pairing relationships. In real-world, unpaired datasets are prevalent. For example, it is interesting to learn transformations between a set of chairs and a set of tables, but there does not exist a natural pairing relationship between chairs and tables. Real 3D scans of indoor scenes captured by humans [14] and 3D scenes created by artists [15] are two datasets that cannot be well-paired. Learning geometric transformations directly from such unpaired datasets will provide new insights and benefits to practical problems, such as surface reconstruction, SLAM, etc. We are interested in developing a deep neural network aiming at learning general-purpose shape transforms from unpaired domains. The network is trained on two sets of shapes, e.g., tables and chairs, or letters of two different font styles, each represented by a point cloud. There is neither a pairing between shapes in the two domains to guide the shape translation, nor any point-wise correspondence between any shapes.

1.2 Challenges

In this section, we discuss three major challenges lying in learning shape transform: (1).Lack of point-level supervision, (2).Representation of shape transform, and (3).Feature preservation and alteration.

1.2.1 Lack of point-level supervision

Unlike image-to-image translation, where there exist pixel-to-pixel correspondences, implied by pixel coordinates, between a source image and its target image, we usually do not have point-level correspondences between training shapes. Actually, for many types of shape transforms, point-to-point correspondences do not exist between the two domains, which stops us from developing techniques that rely on point-level supervision. In some examples, such as the transforms between skeleton and 3D shape, or between cross-sectional profiles and 3D shape, we have pairing relationship between shapes from two domains. This pairing relationship can provide us shape-level supervision, which may be strong enough to train our networks to learn the transform if the networks are well designed to utilize the information. While if shape-level pairing relationship between two domains does not exist, for example, when transforming chairs into tables, an unsupervised solution is required to automatically discover the commonality between two domains and learn to alter the source shapes in a natural way.

1.2.2 Representation of shape transform

In image translation, pixel values are stored on a regular grid. An image can be naturally represented by a matrix, and the image translation can be modeled by convolution operations. Unlike images, a 3D shape is a set of 2D manifolds, or a set of points on the 2D manifolds after discretizing it by sampling. Shape transform itself is a function that takes one shape from the source domain as input and produces another shape that is in the target domain as output. Most of the time, the function is non-contiguous and cannot be modeled as an injective mapping. For example, when transforming a chair into a table, some points may be removed, and some points may be added, which makes the function highly discontinuous. When transforming a skeleton into the surface of an object, it requires a one-to-multiple mapping where a skeletal point is mapped onto points lying on a closed profile curve. Complicated cases of the transform functions motivate us to design general-purpose representations for shape transforms. In our proposed solution P2P-NET with shape-level supervision, we use point-wise displacements to represent the learned shape transform. While the point-wise displacements do not exist in our training data, our network learns to generate the displacements after trained under shape-level supervision. In our proposed solution LOGAN with an unsupervised setting, we represent shape transform by a fully-connected network in a common latent space of the two shape domains.

1.2.3 Feature preservation and alteration

Given two sets of shapes, it is challenging for a neural network to learn what are the right shape attributes to preserve with the transform, as there are many ambiguities and conflicts in them, especially when the two sets of shapes are unpaired. It poses a fundamental challenge: what features of the source shape are to be preserved/altered is unknown to the machine — it must depend on the given shape domains, and our network must learn it. For example, when the task is to transform a bicycle into a motorbike, humans know that the size of wheels and overall contour of the bike would better be preserved, but such commonalities are challenging for machine to detect and utilize. When the training shapes are paired, the common shape attributes are more or less clear and tractable. However, when the training shapes are unpaired, ambiguities and conflicts will make the problem much more difficult. Feature disentanglement is one of the most significant challenges in learning shape transforms when the dataset is unpaired. In our proposed solution, LOGAN, with an unsupervised setting, we present a novel feature preservation mechanism for accelerating feature disentanglement in a common latent space of the two domains.

1.3 Overview of the thesis

In this section, we provide overviews of the two methods we propose for learning shape transform under shape-level supervision and without supervision.

1.3.1 Point-to-Point Displacement Network for Shape Transform

Under the setting that the pairing relationship between shapes in the two input domains is available, we propose P2P-NET, a general-purpose deep neural network which learns geometric transformations between point-based shape representations from two domains, e.g., meso-skeletons and surfaces, partial and complete scans, etc. P2P-NET is trained on paired shapes from the source and target domains, but without relying on point-to-point correspondences between the source and target point sets. The architecture of the P2P-NET is that of a bi-directional point displacement network, which transforms a source point set to a prediction of the target point set with the same cardinality, and vice versa, by applying point-wise displacement vectors learned from data. The training loss combines two uni-directional geometric losses, each enforcing a shape-wise similarity between the predicted and the target point sets, and a cross-regularization term to encourage consistency between displacement vectors going in opposite directions. We develop and present several different applications enabled by our general-purpose bidirectional P2P-NET to highlight the effectiveness, versatility, and potential of our network in solving a variety of point-based shape transformation problems.

1.3.2 Unpaired Shape Transform in Latent Overcomplete Space

Under the setting that the pairing relationship between shapes from two domains is unavailable, we introduce LOGAN, a deep neural network aimed at learning general-purpose shape transforms from unpaired domains. The network is trained on two sets of shapes, e.g., tables and chairs, while there is neither a pairing between shapes from the domains as supervision nor any point-wise correspondence between any shapes. Once trained, LOGAN takes a shape from one domain and transforms it into the other. Our network consists of an autoencoder to encode shapes from the two input domains into a common latent space, where the latent codes concatenate multi-scale shape features, resulting in an overcomplete representation. The translator is based on a generative adversarial network (GAN), operating in the latent space, where an adversarial loss enforces cross-domain translation while a feature preservation loss ensures that the right shape features are preserved for a natural shape transform. We conduct ablation studies to validate each of our key network designs and demonstrate superior capabilities in unpaired shape transforms on a variety of examples over baselines and state-of-the-art approaches. We show that LOGAN is able to learn what shape features to preserve during shape translation, either local or non-local, whether content or style, depending solely on the input domains for training.

1.4 Contributions

In this thesis, we present two deep learning frameworks for learning shape-to-shape transformation for paired and unpaired domains. With shape-level supervision, we introduce

P2P-NET, short for point-to-point displacement network, for learning shape-to-shape transformation for paired domains. The main contributions of P2P-NET include:

- It is the first general-purpose deep neural network designed to learn transformations between point-based shape representations.
- The bidirectionality of the network with a novel cross regularization loss to mutually enhance two directional transforms.

Pairing shapes in two domains sometimes is impractical. Therefore, we introduce a latent overcomplete GAN, i.e., LOGAN, for learning shape-to-shape transformations for unpaired shape domain. The main contributions of LOGAN include:

- It is the first deep neural network for general-purpose, unpaired shape-to-shape transformation.
- It is the first translation network that can perform both content and style transfers between shapes, without changing the network's architecture or any of its hyperparameters. Owing to the overcomplete, multi-scale representations it learns, LOGAN adapts its feature preservation solely based on the two input domains for training.
- It puts forth the interesting concept of implicit feature disentanglement, enabled by the overcomplete representation, which may hold potential in other application settings.

1.5 Thesis organization

The thesis is organized in the following way: In the next chapter (Chapter 2) for background discussion, we will discuss related literature in three main related fields to our topic. In Chapter 3, we will introduce our proposed P2P-NET for learning shape transform with shape-level supervision, describe the techniques, and provide the results. In Chapter 4, we will introduce our proposed LOGAN for learning shape transform under an unsupervised setting, describe the techniques, and provide the results. Finally, in Chapter 5, we will give our conclusion and discuss interesting directions for future work.

Related publications. This thesis includes previously published material. The following is a list of the papers and their corresponding chapters:

- The point-to-point displacement network described in Chapter 3 appeared in the paper: Yin, Kangxue, Hui Huang, Daniel Cohen-Or, and Hao Zhang. P2P-NET: Bidirectional point displacement net for shape transform. ACM Transactions on Graphics (SIGGRAPH). 2018 Jul 30;37(4):1-3.

- The network LOGAN for unpaired shape transform in latent overcomplete space described in Chapter 4 appeared in the paper: Yin, Kangxue, Zhiqin Chen, Hui Huang, Daniel Cohen-Or, and Hao Zhang. LOGAN: unpaired shape transform in latent overcomplete space. ACM Transactions on Graphics (SIGGRAPH Asia). 2019 Nov 8;38(6):1-3.

Chapter 2

Background

In this chapter, we discuss previous works in some of the most related areas to our thesis. We start from shape transform in geometry processing by discussing some heuristic-based approaches for single shape transformation tasks, such as skeletonization, surface reconstruction, and cross-sections reconstruction. After that, we review the deep learning techniques for learning shape features from point clouds. Finally, we discuss image-to-image translation for both paired domains and unpaired domains.

2.1 Shape transform in geometry processing

Geometry processing is a large area where many problems can be considered as a problem of shape transform. In this section, we mainly focus on these topics that are directly related to the example shape-to-shape transformation tasks that our general-purpose networks are tested on.

Skeletonization. Skeleton of a shape is a compact representation for the topology and geometry of the shape. There are multiple definitions for medial skeletons of 3D shapes. Blum's medial axis [16] is defined as the locus of centers of maximally inscribed balls in the shape. Directly applying the original definition of medial axis transform(MAT) by Blum for extracting skeletons from complex shapes is not practical. Many skeletonization methods use a definition of skeleton based on a grassfire analogy of medial axis transform [17]: considering the shape as a piece of grassland, the fire started at the boundary of the grassland and spreads at a uniform speed. The points where the wavefronts meet and quench can be treated as a medial skeleton. Laplacian-based mesh contraction [18] is an example that uses this analogy.

We refer to [19] for a detailed survey on shape skeletonization. As we use a point-based representation in the thesis, we focus exclusively on works developed for skeleton extraction from point clouds. For example, Sharf et al. [20] extract curve skeleton from point cloud with an algorithm involving multiple competing fronts that evolve in a coarse-to-fine man-

ner inside the volume defined by the point cloud. Assuming a cylindrical shape prior and accurate point normals to compensate for missing data, Tagliasacchi et al. [21] present a method that directly works on incomplete point clouds. Cao et al. [22] developed a robust Laplacian contraction method for point cloud skeletonization, where the contraction is realized by local Delaunay triangulation and topological thinning. Huang et al. [23] introduce an iterative contraction procedure to extract L1-medial curve skeleton from 3D point cloud.

Surface reconstruction and completion. Many surface reconstruction methods take an input point cloud as input and estimate an implicit function that smoothly interpolates over data missing regions. Among them, Hoppe et al. [24] reconstruct the surface from the point cloud by determining the zero set of an estimated signed distance field. Carr et al. [12, 25] reconstruct the shape by fitting Radius basis functions(RBFs) to the given data points. Poisson surface reconstruction [26, 13, 27] estimates an indicator function from the point cloud by solving a Poisson equation, where the gradient of the indicator function is constrained at all input points . Davis et al. [28] introduce a diffusion-based method to extend the signed distance field to fill the holes existing in the input scan.

Example-based methods are capable of dealing with larger holes, though it requires explicit hole detection and proper exemplars from the incomplete input surface or from a database. Sharf et al. [29] introduce a context-based method that analyzes the input surface and iteratively fills the hole by copying patches from the input surface. As context-based surface completion may synthesize geometric patches that are very different from those on the original surface, Harary et al. [30] impose a coherence objective term that encourages every local neighborhood of the filled hole to be similar to some patches from the input mesh. To convert incomplete surface scans into a complete 3D model, Pauly et al. [31] retrieve for a suitable 3D shape from a database to provide a template for regions of missing data. The retrieved shapes are then warped to conform with the input scan.

While implicit function-based methods and example-based methods focus on geometry completion via smooth interpolation and/or detail transfer, skeleton-driven surface reconstruction is able to deal with significant missing data under the guidance of the shape skeleton. To deal with the cases where large portions of the data may be missing in the point data, after skeletonization, Tagliasacchi et al. [21] reconstruct geometry around the shape skeleton by extending the skeleton with medial sheets. Wu et al. [32] introduce a schematic surface reconstruction method for architectural scenes represented by 3D point clouds, where the scene is represented by a network of planar transport and profile curves. Yin et al. [33] deal with more general shape topology and geometry with an operation called "morfit" which sweeps along shape skeleton to reconstruct the shape as a combination of generalized cylinders. Lazar et al. [34] propose a solution for topology-aware surface reconstruction, where the topological constraints are formulated as convex constraints over an integer vector.

Style analysis and manipulation for 3D shapes. Earlier work in shape style analysis leverage structural similarity between shapes of the same class. For example, Xu et al. [35] analyze a set of 3D shapes at part level and define the shape style by anisotropic part scales. Kalogerakis et al. [36] define part and shape styles by latent variables of a probabilistic model for component-based shape structure trained on a set of compatibly segmented shapes of the same object class. Huang et al. [37] classify object styles with a distance metric, learned via a multi-label semi-supervised approach, that captures the geometric similarity within an object class

Given a set of 3D shapes spanning multiple object classes and styles, Lun et al. [38] introduce a method that mimics human perceptions of stylistic similarity by learning the relative importance of different metrics from crowdsourced data. Lim et al. [39] explore deep metric learning for perceiving style similarities of 3D shapes with a multiview representation for 3D shapes. With a sequence of element-level operations, Lun et al. [40] transfer the style of an exemplar shape to the target shape while preserving the functionality of the target shape. Recently, Yu et al. [41] co-analyze projected feature lines of 3D shapes with a semi-supervised method and backproject the learned style features onto the 3D shapes. Ma et al. [42] use the concept of shape analogy in IQ test, where three shapes A, B, C are presented to the subject and the subject is asked to select an output D such that the transformation from C to D mimics the transformation from A to B. They recast 3D shape style transfer in terms of shape analogy which consist of a set of simple transformations, and compute an optimal set of transformations that form a shape analogy satisfying the criteria of simplicity, compactness, and continuity.

2.2 Deep learning on point sets

Early research on 3D shape analysis and 3D computer vision with deep learning mainly focused on volumetric representation [43, 44, 45, 46] and multi-view representations [47, 48]. However, these representations always lead to redundant computation, which limits the size and capability of deep learning models. Compared to them, point cloud as a 3D shape representation is more lightweight, thus can retain better geometric details.

The well-known challenges that have to be faced in learning shape features from point clouds include sparse input, spatial irregularity, and invariance to point permutation. While the points in a point cloud are unordered, the output of the neural network has to be consistent when the same point clouds of different permutations are given as input. The sparsity and irregularity of points make it difficult to directly use conventional techniques such as CNNs or RNNs from other domains to learn shape features for point clouds. In this section, three main categories of works will be discussed: i). point-wise feature pooling, ii). convolution on points, iii). space partition tree-based methods.

Point-wise feature pooling. The order of points in a point cloud is arbitrary. The key challenge in learning shape features from a set of points is making the network and its output invariant to point permutation. A pioneering work that tackles this challenge is PointNet [49] which directly consumes unorganized points as input and process each point with weight-sharing multilayer perceptrons. The key idea of PointNet is learning a high-dimensional spatial encoding of each point, and then aggregate all individual point features to a global point cloud feature vector by max-pooling, which is a symmetric function. PointNet is followed by PointNet++ [50], which enables hierarchical learning on point sets. PointNet++ uses PointNet to extract local shape feature vectors from a batch of points. The input point set goes through a patch-wise feature transform followed by feature aggregation, so as to serve the tasks of shape classification and segmentation. For the purpose of reasoning about the structural dependencies of local regions in point clouds, [51] proposes SRN-PointNet++ that uses a structural relation network (SRN) module to exploits geometrical and locational relations of neighboring points. Another multi-scale variant of PointNet, by Guerrero et al. [52], is adapted for estimating local shape properties, such as normals and curvatures.

PointWeb [53] also uses max pooling to aggregate point-wise features into a global feature vector. However, it enhances feature aggregation in local point patches by learning adaptive feature adjustment from a fully-connected graph of local point patches. Sun et al. [54] developed a variant of PointNet that achieves rotation-invariance by transforming point cloud into a unique local coordinate system. Xie et al. [55] developed ShapeContextNet that adopts handcrafted shape context features in building neural layers. Yang et al. [56] present Point Attention Transformers (PATs) which use a parameter-efficient Group Shuffle Attention (GSA) for permutation invariance feature transform. The Group Shuffle Attention is developed based on Multi-Head Attention [57], but is faster for computation.

Convolution on points. The main challenge of developing Convolutional neural networks for point clouds is that the points in a point cloud are unorganized and irregular. Unlike images, the points are unordered, and there does not exist a regular grid on point clouds, which naturally allows convolution operators to operate on. As the key to the success of CNNs is that the convolution operators are able to leverage spatially-local correlation in data represented in dense grids, directly applying convolutions on the features associated with points in a point cloud will result in loss of geometric information and invariance to point permutation.

A straightforward method to enable CNN on point clouds is to project points on a regular grid or split the point cloud by a regular grid. Hua et al. [58] present a point-wise convolution operator for point clouds where nearest neighbors are queried on the fly for each point and binned into cells that locate in a regular 3D grid. By stacking a set of point-wise convolution layers together, they can build fully convolutional neural networks

for point cloud classification and segmentation. Mao et al. [59] present an Interpolated Convolution(InterpConv) operation which interpolates point features from an irregular point cloud by approximating an interpolation function and assign the interpolated value to neighboring kernel weights located on a regular grid. The operator is shown to be permutation and sparsity invariant and can directly handle irregular inputs. Atzmon et al. [60] transfer Euclidean volumetric convolution to point clouds by using a pair of operators that they call extension and restriction. The extension operator is essentially an operator that converts point cloud to volumetric function by radius basis functions(RBFs). The restriction operator converts volumetric function back to point cloud by sampling. Inspired by high-dimensional Gaussian filtering technique [61] where convolutions can efficiently be performed on sparse data in high dimensions, Su et al. [62] projects point cloud as a sparse set of samples in a high-dimensional lattice which is regular and where convolutions are then applied on the regular high-dimensional lattice.

We can extend convolutional operations from regular grids to irregular point sets by modeling continuous convolution operations. In concurrent works by Wang et al. [63] and Hermosilla et al. [64], convolution operation on irregular point cloud is approximated by Monte Carlo integration [65]. The convolutional kernels are represented as parametric functions(modeled by multilayer perceptrons) of points in local neighborhood. Based on Monte Carlo approximation approach, Xu et al. [66] apply a density scale learned from an MLP to re-weight the learned weight matrices. SpiderCNN [67] represents the convolution kernel as a product of a simple step function that captures local geodesic information and a Taylor polynomial defined on neighboring points that ensure the kernels are complex enough to capture the intrinsic local geometric variations. Thomas et al. [68] present Kernel Point Convolution (KPConv) that uses a set of kernel points, to define the convolution matrix as a weighted sum of the matrices on kernel points in a local spherical neighborhood. By learning local shifts applied to the kernel points, KPConv is able to involve spatial deformation and adapts the shape of its kernels for different regions of the input cloud.

Different from the methods that focus on modeling continuous kernel weight, Li et al. [69] present a χ-conv operator which simultaneously weights and permutes the input features carried on the irregular point cloud. The points in a local patch are then cast into a latent and potentially canonical order before applying convolution on them. The convolution aggregates point features into a representative point selected by downsampling.

Another alternative formulation for feature learning from point cloud is to learn convolution operations over graphs. Wang et al. [70] build a k-NN graph over the point cloud, where spectral convolution is carried out in the neighborhood of each point of the downsampled version of the point cloud. The standard max pooling is replaced with a recursive clustering and pooling strategy, which is designed for aggregating information within clusters of in spectral domain. Edge-conditioned filter networks [71] avoid the spectral domain and formulates a convolution-like operation on graph signals in the spatial domain. Their kernel

weights are dynamically generated for each specific input sample based on the corresponding edge labels.

Space partition tree-based methods. The main drawback of using 3D regular grids for representing 3D shapes for feature learning is its weak scalability. In computer graphics, many data representations have better scalability than a regular 3D grid, such as k-d tree [72], octree [73], BSP tree [74], constructive solid geometry [75], etc.

It is possible to define the neural network by following a space partition tree. For example, Klokov el al. [76] present a Kd-network that uses k-d tree to form the computational graph and define learnable parameters on different partition levels. At train time, point clouds are downsampled or upsampled to have the fixed size $N = 2^D$, where D is the depth of the k-d tree. A hierarchical deep model is learned in a feed-forward bottom-up fashion, where on each node of the k-d tree features of two child nodes are aggregated by concatenation followed by a non-linear transform. In another k-d tree-based method, Zeng et al. [77] exploit local and global contextual cues imposed by the k-d tree and calculate the representation vectors progressively along the k-d tree for feature aggregation. Gadelha et al. [78] directly use k-d tree to sort the points in a point cloud and optimize the point ordering by minimizing PCA reconstruction error after PCA analysis. With the points been sorted, they can process them by fully-connected layers. In a follow-up work [79], they sort point cloud in a hierarchical and locality-preserving order such that feed-forward processing through 1D convolutions and coarse-to-fine analysis are enabled.

2.3 Image-to-image translation

Image-to-image translation is a domain translation problem that is related to shape-to-shape transformation. Before researchers started working on general-purpose image translation [4, 5, 6, 80], earlier works for image translation mainly focused on learning style or texture transfer [81, 82, 83, 84]. Recently, some works jointly warp images and transfer styles from one domain to another [8, 9]. In this section, we discuss image translation methods for both paired domains and unpaired domains.

Paired domains. To translate images from one domain to another, researchers leverage adversarial learning to get plausible output images. Compared to directly measuring the difference between two images [85, 86], adversarial learning with GANs [87] often learns a more adaptive error metric through the discriminator. Given paired images from two input domains, Isola et al. [4] use a conditional GAN [88] with a U-Net [89]-based generator, and a PatchGAN classifier as the discriminator, for image translation. The generator takes an input image from the source domain and tries to output an image that is in the target domain, while the discriminator is fed by a pair of input and output images and tells

whether the pair is real or fake. [4] assumes that image-to-image translation is deterministic. However, many image-to-image translation problems are ambiguous – one image from the source domain may correspond to multiple images in the target domain. To this end, Zhu et al. [90] models a conditional distribution for the output image given the input image, which can be sampled to get multiple output images. While the image resolutions dealt by the above approaches are limited, Wang et al. [7] synthesize high-resolution images from semantic label maps using a conditional GAN framework with a coarse-to-fine generator, a multi-scale discriminator, and an improved adversarial loss.

Unpaired domains. Collecting paired training images can be time-consuming or even impractical. For example, when the task is translating winter photos into summer photos, asking for a large number of paired photos of the same scenery in the same camera view is unrealistic. Therefore unsupervised learning framework for general-purpose image-to-image translation is needed. To this end, CycleGAN [5], DualGAN [6], and DiscoGAN [91] use the idea of dual learning [92] from natural language processing, where the translation networks for two directions are jointly trained. A cycle-consistency loss or a dual learning loss are applied to enforce the similarity between image A and the image $g(f(A))$, where f is the function that maps images from the source domain to the target domain, g is the function that maps images from the target domain to the source domain. These approaches succeed at image translation with low-level appearance shift, but often fail when the translation requires geometric deformation. Gokaslan et al. [93] show that the problem can be relieved by considering more a global context with a dilated convolutional discriminator [94], a more context-aware generator, and a multi-scale perceptual loss [95]. In another thread of works, UNIT [80] assumes the images from two domains can be mapped into a shared latent space. Two coupled GANs are trained to translate images by encoding them into the shared latent space and decode their latent codes as images in the target domain. MUNIT [96] assumes that an image latent code can be decomposed into a content code that is shared by different domains, and a style code that is unique to a specific domain. Image translation is then posed as replacing the style code of one image in the source domain with a style code of the target domain. FUNIT [97] achieves few-shot, unsupervised image-to-image translation via adaptive instance normalization technique [98].

Concurrent to the development of the methods for learning shape transform in this thesis, some works in the field of image-to-image translation can learn geometric deformation for image translation. For example, CariGANs [99] detect landmarks from human face images, and train a GAN to deform the landmarks to drive image deformation for photo-to-caricature translation. Wu et al. [100] use a similar landmark-based technique for more image domains. The two methods rely on landmarks on the images; thus, they are not general-purpose. More recently, later than the development of the methods for learning shape transform in this thesis, Katzir et al. [8] perform general-purpose image translation

along a deep feature hierarchy in a cascaded deep-to-shallow fashion, so as to achieve geometric deformation in the image. In GANHopper, Lira et al. [9] perform general-purpose image translation through multiple hops, where each hop produces an in-between image that resembles weighted hybrids between images from the two domains. As a result, it allows geometric deformation as well. The above methods can jointly warp images and transfer style from one domain to another. However, they are not designed for dealing with significant shape transformations, such as turning a chair into a table, or shape skeletonization.

Chapter 3

Point-to-Point Displacement Network for Shape Transform

3.1 Introduction

Point primitives and point-based processing have attracted considerable interest from the computer graphics community for many years [101]. As a fundamental shape representation, point sets are more compact than voxels and more flexible than polygon meshes. They are immediately available as the default output from most 3D shape acquisition devices. Recently, deep neural networks have been designed to learn global and multi-scale point set features for shape classification and segmentation [49, 50], as well as geometry (e.g., normal or curvature) estimation [52]. Image-driven generative models have also been trained to reconstruct point-based 3D object representations from single or multi-view images [102, 103, 104].

In this paper, we are interested in exploring how deep neural networks can benefit a new class of problems in point-based graphics: *geometric transformations* between shapes represented by point sets. These shape transforms span a wide spectrum of applications. Some examples include transforms between shape skeletons and surfaces, incomplete and completed object scans, 2D contours and 3D shapes, and simplified and detailed surfaces. Our goal is to develop a *general-purpose* neural network that is capable of learning geometric transformations between point sets from two domains. Recently, in computer vision, there has been a great deal of interest in solving a similar problem for images, namely, designing general-purpose, end-to-end image-to-image translation networks [4, 105, 5, 6].

Most successes on generic image translation have been achieved for tasks leading to "stylistic" changes in images, without geometrically transforming the image content. These tasks include translating between night and day images, artistic styles, material properties, etc. Under this setting, some recent works, such as CycleGAN [5] and DualGAN [6] can train their translators *without* paired images, but they both rely on loss functions that measure pixel-to-pixel differences. Thus, while the images are not paired, the pixels are. However,

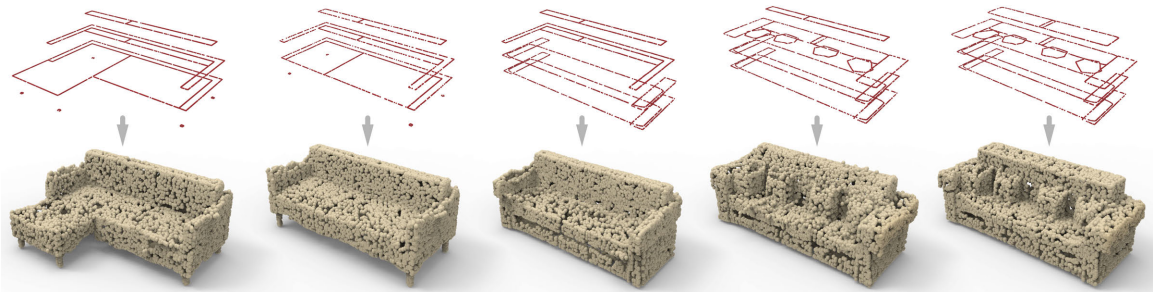


Figure 3.1: We develop a general-purpose deep neural network which learns geometric transformations between point sets, e.g., from cross-sectional profiles to 3D shapes, as shown. User can edit the profiles to create an interpolating sequence (top). Our network transforms all of them into point-based 3D shapes.

when the translation involves geometric changes, these methods have all encountered clear obstacles.

In our work, the distinction, as well as the key novelty, of the problem is that transforming geometry *is* the goal. While pixel correspondences can be trivially defined between two images depicting the same content (e.g., night vs. day photos of the same location), pairing points which represent geometrically different shapes is, in general, far from straightforward. Even if the two sets of points were sampled from the same shape, e.g., a cross-sectional profile representation vs. that of the whole (see Fig. 3.1), there may not always be a clear point-to-point correspondence between them. In our first attempt to develop a general-purpose transformation network for point sets, and in contrast to CycleGAN and DualGAN, we rely on paired shapes in training, but do not require paired points.

Specifically, we design a *point-to-point displacement* network, coined *P2P-NET*, which transforms an input point set to an output set with the same cardinality, by applying *point-wise* displacement vectors learned from data. The P2P-NET is trained under a weakly-supervised setting, where paired point sets which share some commonality are provided, but not their point-wise mapping. In most of the applications considered, the commonality is that the two point sets were sampled from the same shape. However, in many applications, there is no clear point-to-point correspondence between the two point sets, or the two sets could contain point samples acquired under different view settings or at different time instants. Not requiring point-wise correspondences can significanly expand our capacity to collect training data for P2P-NET.

Given two domains of point set data, $\mathscr{X}$ and $\mathscr{Y}$, we introduce a *bidirectional* architecture to learn the two transformations $\mathscr{X}$-to-$\mathscr{Y}$ and $\mathscr{Y}$-to-$\mathscr{X}$, simultaneously, as shown in Fig. 3.2. Given a source point set X, the (bidirectional) P2P-NET learns to predict a set of displacement vectors $\mathcal{I}_X$ that are applied to X to obtain the predicted point set $\hat{Y} = X + \mathcal{I}_X$. One objective of training the P2P-NET, defined by a *geometric loss*, is to

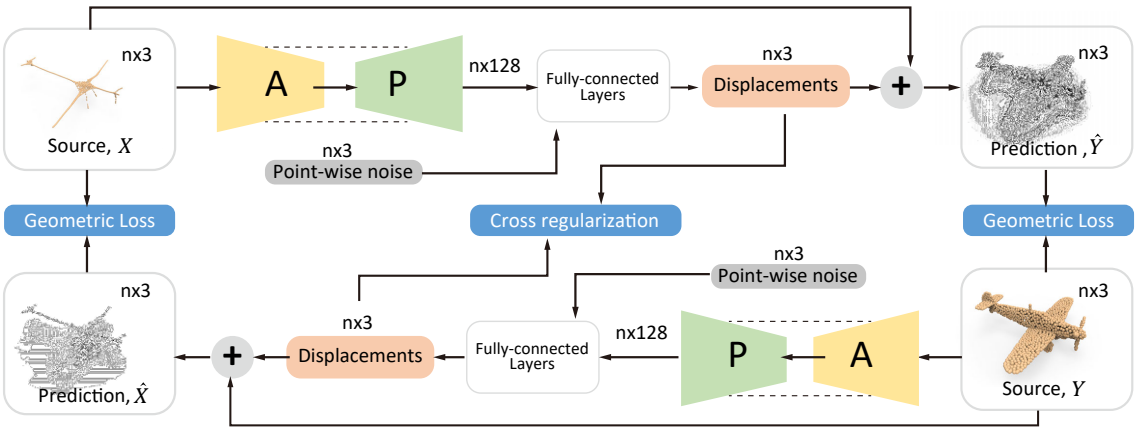


Figure 3.2: Network architecture of our bidirectional P2P-NET, which transforms a source point set to a prediction of the target point set with the same cardinality. Note that blocks A and P represent set abstraction layers and feature propagation layers, respectively, of PointNET++.

make the prediction $\hat{Y}$ as close as possible to the target *shape* represented by the point set Y. At the same time and along the opposite direction, the network also learns to predict displacement vectors $\mathcal{I}_Y$, such that $Y + \mathcal{I}_Y$ is as close as possible to X, shape-wise. In addition, we define a *cross regularization* loss which couples and *mutually enhances* the two directional sub-networks, by encouraging *parallelism* between the two sets of displacement vectors $\mathcal{I}_X$ and $\mathcal{I}_Y$. Our bidirectional P2P-NET is trained with a combined loss consisting of two (directional) geometry terms and one cross regularization term. None of the loss terms requires a point-wise correspondence between X and Y.

Along each direction of the P2P-NET, the network takes a set of 3D points as input and first learns a multi-scale feature per point through set abstraction and feature propagation layers of PointNet++ [50]. The learned feature vector for each point is then *concatenated* with an *independent* Gaussian noise vector. The set of "noise-augmented" feature vectors are fed into a set of fully connected layers, which produce a set of 3D displacement vectors, one per input point, as shown in Fig. 3.2.

P2P-NET learns point set transforms *implicitly*. It does not learn how to directly displace each point in the source shape to its corresponding point in the target shape, since such point-to-point correspondences are not provided by training data. Instead, our network learns a mapping from point features (encoded as in PointNET++) to displacement vectors, which are applied to the source shape. As a result, nearby points in the source shape, which often possess similar point features, tend to be mapped to similar displacement vectors, which would impose a certain "rigidity" on the shape transforms that P2P-NET can learn. The augmentation of independent per-point noise into P2P-NET alleviates this problem by providing added degrees of freedom to the displacements of individual points, allowing the network to learn a richer variety of transforms.

The main contributions of our work include:

- The first general-purpose deep neural network designed to learn transformations between point-based shape representations.
- Bidirectionality of the network with a novel cross regularization loss to mutually enhance two directional transforms.
- Training losses defined without point-to-point correspondence, allowing the network to be trained under weak supervision.

We demonstrate how noise augmentation, cross regularization, and bidirectionality improve the performance of our P2P-NET, as it carries out its learning tasks. We develop and present several different applications enabled by our general-purpose bidirectional P2P-NET to highlight the effectiveness, versatility, and potential of our network in solving a variety of point-based shape transformation problems. These include transforms between skeletons and shapes, between skeletons and scans, between partial and complete scans, and finally between cross-sectional profiles and 3D shapes.

3.2 Related work

The literature on point-based graphics and the application of deep neural networks to solve graphics problems is vast. In this section, we only cover the most relevant works to P2P-NET.

Point processing. Point cloud, as a 3D shape representation, has shown its advantages and applications in geometry modeling [106, 107], rendering [108, 109], and computer animation [110, 111]. With the emergence of affordable 3D acquisition devices, point cloud data is widely captured, accumulated, and processed by many useful techniques, e.g., in point cloud filtering [10], resampling [11], and surface reconstruction [12, 13, 112], among others.

Point set displacement. Several previous works have taken a displacement-based approach to process point sets. One such example is point set skeletonization, where point samples gradually converge from a shape's surface to a skeletal structure. The main challenge lies in how to deal with missing data over the latent surface, e.g., when the point scan was acquired from a single view. Tagliasacchi et al. [21] propose a generalized rotational symmetry axis (ROSA) to extract curve skeletons from incomplete point clouds. Cao et al. [22] apply a Laplacian-based contraction to extract curve skeletons. Huang et al. [23] introduce L_1-medial skeletons by adapting L_1-medians locally to an incomplete point set representing a 3D shape. Our P2P-NET leads to a data-driven approach to point cloud skeletonization and the bidirectional network is also trained for a novel task: *skeleton-to-shape* transform.

Another example is surface completion by evolving point samples to gradually fill missing data over the latent surface. Representative methods include point cloud consolidation via LOP [113] and WLOP [114, 115]. A more recent work that bridges point cloud skeletonization and consolidation [116] spreads point samples regularly to cover gaps over surface regions via a joint optimization of surface and structure samples. Point cloud resampling can also be applied for edge enhancements [11]. P2P-NET offers a data-driven approach to surface completion, via point displacements, which offers an alternative to other learning-based surface completion techniques, such as the recent work by Dai et al. [117].

Neural networks for point processing. Neural networks excel in learning global features. A key development in connecting point sets to neural networks is PointNet [49], which directly consumes unorganized point samples as input. This is followed by PointNet++ [50], which enables hierarchical learning on point sets. In both cases, the input point set goes through point-wise or patch-wise feature transform followed by feature aggregation, either globally or locally, so as to serve the tasks of shape classification and segmentation (i.e., patch classification). Another multi-scale variant of PointNet, by Guerrero et al. [52], is adapted for estimating local shape properties, such as normals and curvatures. Sung et al. [118] demonstrate the usefulness of PointNets for component suggestion in part-based shape assembly.

Our P2P-NET is designed to solve a different class of problems, namely, point displacement based shape transforms. While P2P-NET does employ the set abstraction and feature propagation layers of PointNet++ [50] for feature learning, it combines the learned features with noise and trains a bidirectional network, with a novel loss term combining shape approximation and cross regularization, to obtain point displacement vectors.

There have been several recent attempts at developing deep generative networks for point-based shapes. Fan et al. [102] design and train a neural network as a conditional sampler, which is capable of predicting multiple plausible 3D point clouds from a single input image. Multiple point clouds from different views have also been constructed as intermediate shape representations for the purpose of generating 3D shapes from 2D images and/or sketches [104, 103]. Gadelha et al. [78] synthesize point-based 3D shapes in the space of shape coefficients, using a generative adversarial network (GAN). They build a KD-tree to spatially partition the points and then conduct PCA analysis to derive a linear shape basis and optimize the point ordering.

To the best of our knowledge, P2P-NET is the first deep neural network designed to learn geometric transformations between point-based shape representations.

Learning transformations. Several classical vision problems need to account for spatial transformations in images, e.g., recognizing objects undergoing deformations [119, 120] and synthesizing images under novel views [121, 122], among others. Both works are representative of applying deep neural networks for their respective tasks. There has been considerable

less effort on learning geometric transforms for 3D shapes. Recent attempts have been made to learn to transfer surface details [1], decorative styles [2], and to predict piecewise rigid transformations of 3D objects [3]. In contrast, our work aims to develop a *general-purpose* neural network for learning transformations between point-based 3D shapes.

Paired vs. unpaired training data. Analogous to our shape transform problem, is general-purpose image-to-image translation [4, 105, 5, 6], where point displacements can be regarded as a counterpart to pixel-to-pixel transforms. An important feature of some of these recent works [105, 5, 6] is that the training does not require paired images. On the other hand, these works, which rely on deep generative neural networks such as GANs, have only shown success in color and texture transforms, e.g., for altering painting styles or material properties of imaged objects. Training these networks to deform objects geometrically in images has remained an unresolved challenge. In contrast, P2P-NET is designed to learn geometric transforms between 3D shapes; it requires paired training data, but not paired points.

Bidirectionality vs. cycle consistency. Our design of the bidirectional P2P-NET drew inspirations from dual learning [6] and the use of cycle consistency loss [5]. What is common about these works is that they all learn transforms between two domains. The cycle consistency loss is a clever way of dealing with the challenge of not having paired training data from the two domains. However, P2P-NET is not built on a cyclic loss. With paired training data for P2P-NET, we can afford to define the two directional geometry losses, without the inverse mappings. At the same time, the bi-directionality between the two transforms is taken advantage of, since we define the extra cross regularization loss to enhance the training. Another distinction lies in how the loss functions are defined: the cycle consistency loss measures pixel-to-pixel differences, while our geometry loss measures a *shape-wise* difference between two point sets.

3.3 Bidirectional P2P-NET

3.3.1 Network architecture

The architecture of bidirectional P2P-NET is illustrated in Fig. 3.2. Our training set consists of paired point sets $\{X, Y\}$, with prior relations among them. However, the transformations between two sets X and Y are latent and difficult to model explicitly. For instance, X can be a single-view point scan of a chair, and Y contains complete surface samples of the same chair. As another example, shown in Fig. 3.3, a 2D point set sampled from a dog shape is transformed to represent the shape of a cat and vice versa.

To realize a bidirectional architecture on unordered point sets, we develop a geometric loss (Section 3.3.2) that is order-invariant. Since X and Y are not in dense correspondences, i.e., point-wise, we need to further regularize the loss to balance the mapping and the global

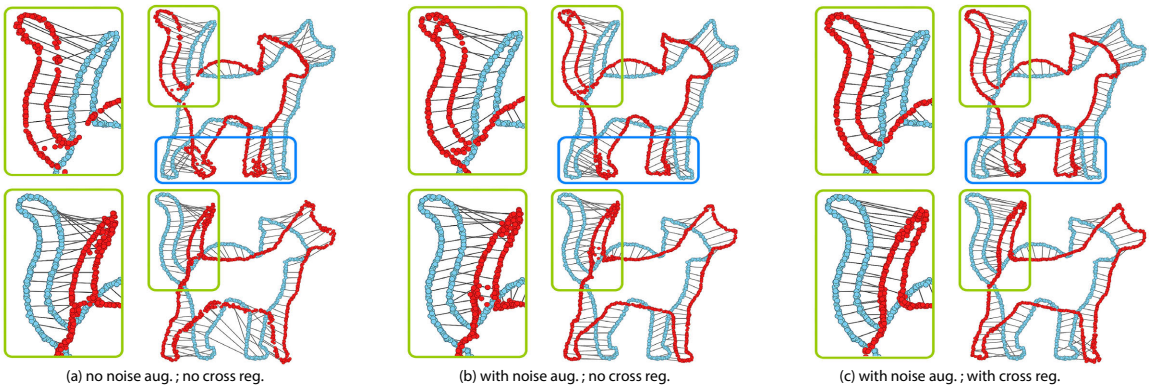


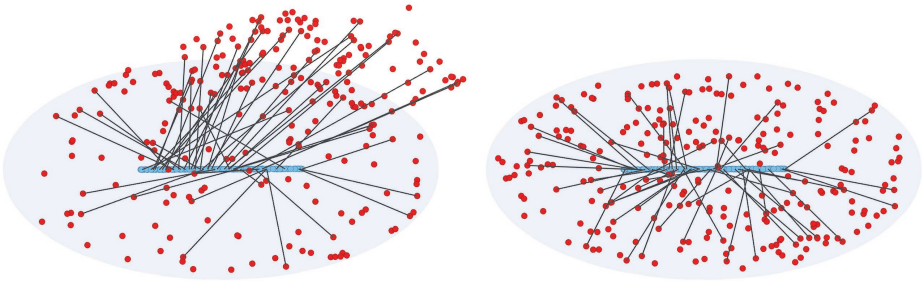
Figure 3.3: An ablation study on cat-dog transforms. P2P-NET was trained on a dataset synthesized by randomly rotating and scaling a pair of 2D point sets representing the shapes of a dog and a cat, respectively. Top row: dog to cat. Bottom row: cat to dog. The source shape is always shown in blue and prediction in red, and only 10% of the displacements are displayed. Light green and blue boxes highlight areas with visible improvements.

distribution of the displacements. To this end, the loss of bidirectional transformations is tightly coupled with a cross regularization (Section 3.3.3) that maximizes the parallelism between displacements from X-to-Y and displacements from Y-to-X.

P2P-NET consists of two network branches in two opposite directions. At each branch, the network first learns a multi-scale feature for each point, using layers of PointNet++ [50]. The input point set is down-sampled and point features are constructed in multiple levels with set abstraction layers (marked with A in Fig. 3.2). Point-wise multi-scale features are then produced with the feature propagation layers (marked with P in Fig. 3.2). Next, the multi-scale point-wise feature vectors are concatenated with the same number of noise vectors, one per point. Each noise vector is an independent Gaussian noise vector of length 32. Finally, the feature-noise vectors are fed to a set of fully connected layers that output displacement vectors $\mathcal{I}_X$. In the end, the network yields the predicted point set $\hat{Y} = X + \mathcal{I}_X$. See Appendix 3.6.1 for a more detailed description of the network architecture.

Noise augmentation in our P2P-NET adds new dimensions to the feature vectors. These newly added (noise) dimensions inject new degrees of freedom, when the network learns to map point features to displacements during training. This effectively neutralizes an "over-fitting" of the point displacements to point features and enables more variation in the displacements. The appended noise vectors are independent to each other, allowing each point to train for its own variation to further improve the versatility of P2P-NET. Furthermore, since the noise introduces stochasticity into the network, we can feed an input point set multiple times during testing, to obtain a dense output point set; see Fig. 3.9 for an example.

Fig. 3.4 illustrates the effect of noise augmentation using a toy example, where P2P-NET is trained to transform points along a straight line to points distributed over an elliptical



(a) no noise aug. (b) with noise aug.

Figure 3.4: Noise augmentation allows P2P-NET to learn to transform points along a straight line (blue dots in the center) to points distributed over an elliptical disk (red dots) more effectively. For a clearer visualization, we only show 20% of the displacement vectors (black lines) which are randomly chosen. Note also that the cross regularization term is not employed.

disk. The network was trained on 1,000 line-disk pairs with random scales and orientations. Since P2P-NET only learns a mapping from point features to displacements, it intrinsically respects the smooth variation of the point features along the straight line. However, the transform task at hand sets a conflicting goal, which is enforced by the geometry loss in the network: map smoothly varying point features to "non-smooth" displacement vectors, so that the output points can be well-distributed over a disk. Without noise augmentation, P2P-NET would respect the point features relatively more rigidly. As shown in Fig. 3.4(a), the network struggles to fulfill the two conflicting goals and produces many similar displacement vectors, causing some points to overshoot over the disk boundary. In contrast, with noise augmentation, the points have added degrees of freedom to be mapped to non-smooth displacement vectors to minimize the geometry loss. The final result is a significantly better point distribution over the disk without overshooting, as shown in Fig. 3.4(b).

In addition to the geometric losses defined and enforced at the two input/output ends of the bidirectional P2P-NET, the aforementioned cross regularization over the point displacements strengthens the coupling between the two directional networks. The ablation study shown in Fig. 3.3 demonstrates the impact of both noise augmentation and cross regularization on a less toyish example: transforming between points representing the shapes of dogs and cats.

3.3.2 Geometric Losses

To measure the geometric difference between the predicted and target point sets, the network is trained with a loss that consists of two terms. One term penalizes points that do not match with the target shape, and the other term measures the discrepancy of the local point density between two corresponding point sets.

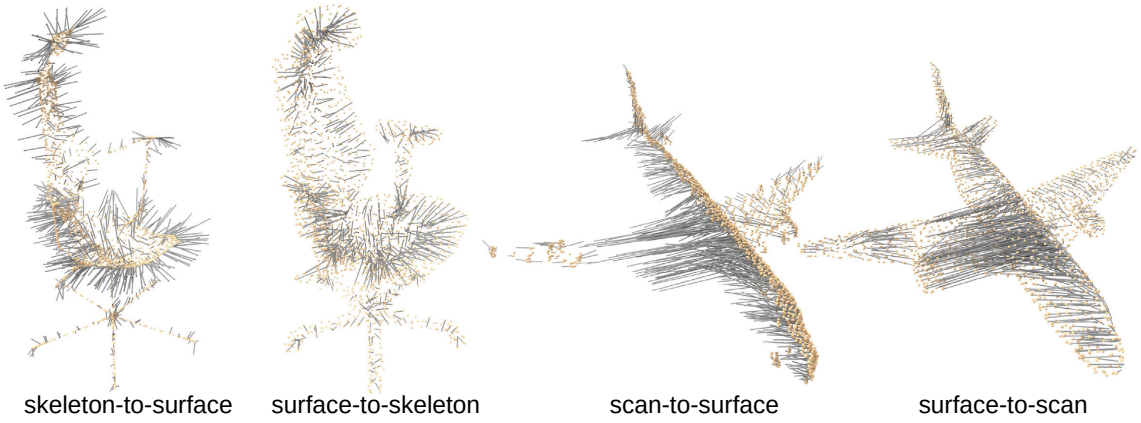


Figure 3.5: Visualization of vectors (grey lines) depicting point-wise displacements learned by P2P-NET for various domain mappings, where the source point sets are rendered in orange. Note that for ease of visualization, only 30% of the vectors are displayed and we do not show the predicted target point sets.

The shape matching loss computes the sum of differences between the shape of transformed point set $\hat{Y} = X + \mathcal{I}_X$ and the shape of target point set Y, vice versa between $\hat{X} = Y + \mathcal{I}_Y$ and X, by searching the closest point from target point set for each displaced source point:

$$L_{\text{shape}}(\hat{Y}, Y) = \sum_{p \in Y} \min_{q \in \hat{Y}} d(p, q) + \sum_{q \in \hat{Y}} \min_{p \in Y} d(p, q),$$

where $d(p, q)$ measures $L2$ distance between points p and q.

This symmetric shape matching term is close to the Hausdorff distance between shapes, except that we compute the sum of closest distances, instead of their maxima. The summation operation makes the loss function differentiable w.r.t. the displaced points, and encourages the displaced point set to match the target tightly.

In addition, we also compute a density loss. For each point p in target point set Y, we define local density measures w.r.t. Y and $\hat{Y}$, respectively, using two k-D vectors ($k = 8$ by default):

$$\begin{aligned}
&[\ d(p, N_1(Y, p)) \quad d(p, N_2(Y, p)) \quad ... \quad d(p, N_k(Y, p))\ ], \\
&[\ d(p, N_1(\hat{Y}, p)) \quad d(p, N_2(\hat{Y}, p)) \quad ... \quad d(p, N_k(\hat{Y}, p))\ ].
\end{aligned}$$

Here we denote $N_i(Y, p)$ as the i-th closest point to p from the same target point set Y, and $N_i(\hat{Y}, p)$ is the i-th closest point to p from the predicted point set $\hat{Y}$.

These two k-D vectors encode density measures of Y and $\hat{Y}$ in small neighborhoods for each point $p \in Y$. The density of the predicted point set $\hat{Y}$ resembles the density of target set Y, if and only if the density vectors of $\hat{Y}$ are similar to that of Y. Therefore, the density

loss is defined as the integration of distances between density vectors of $\hat{Y}$ and Y over all points in Y:

$$L_{\text{density}}(\hat{Y}, Y) = \frac{1}{k} \sum_{p \in Y} \sum_{i=1}^{k} |d(p, N_i[Y, p]) - d(p, N_i[\hat{Y}, p])|.$$

With a setting of single X-to-Y transformation network, the geometric loss function is then as follows:

$$L_{X \to Y}(\mathcal{D}) = \sum_{\{X,Y\} \in \mathcal{D}} \Big(\; L_{\text{shape}}(\hat{Y}, Y) \;\; + \;\; \lambda L_{\text{density}}(\hat{Y}, Y) \; \Big),$$

and similarily to the other direction:

$$L_{Y \to X}(\mathcal{D}) = \sum_{\{X,Y\} \in \mathcal{D}} \Big(\; L_{\text{shape}}(\hat{X}, X) \;\; + \;\; \lambda L_{\text{density}}(\hat{X}, X) \; \Big),$$

where $\mathcal{D}$ denotes our training set, with a weight $\lambda = 1$ by default.

3.3.3 Cross Regularization

We couple the transformations X-to-Y and Y-to-X by a cross regularization over their displacement vectors $\mathcal{I}_X$ and $\mathcal{I}_Y$. The key observation is that when $\mathcal{I}_X$ and $\mathcal{I}_Y$ are encouraged to be parallel to each other, the two transformations can be mutually enhanced, with a more uniform distribution of the displacement mapping.

The regularization term maximizes the parallelism between $\mathcal{I}_X$ and $\mathcal{I}_Y$, without having paired displacements. For each point $p \in X$, or each point $q \in Y$, the displacements are associated with 6D vectors $[p,\ p + \mathcal{I}_X(p)]$ or $[q + \mathcal{I}_Y(q),\ q]$, respectively. The regularization works in 6D in a similar manner as computing L_{shape}. That is:

$$\begin{aligned} L_{\text{reg}}(X, Y) \;\; &= \;\; \sum_{p \in X} \min_{q \in Y} d([p,\ p + \mathcal{I}_X(p)],\;\; [q + \mathcal{I}_Y(q),\ q]) \\ &+ \;\; \sum_{q \in Y} \min_{p \in X} d([p,\ p + \mathcal{I}_X(p)],\;\; [q + \mathcal{I}_Y(q),\ q]). \end{aligned}$$

Minimizing L_{reg} in 6D results in maximizing the parallelism between 3D displacement vectors with two opposite directions from bidirectional transformations. See Fig. 3.3(c) that depicts the enhancement on a 2D toy example by adding cross regularization. More elaborated evaluation on the results is provided in Section 3.4.

Given the displacement regularization that couples the transformations X-to-Y and Y-to-X, the network is trained with a loss function that sums three terms:

$$L_{X \to Y}(\mathcal{D}) + L_{Y \to X}(\mathcal{D}) + \mu \sum_{\{X,Y\} \in \mathcal{D}} L_{\text{reg}}(X, Y), \tag{3.1}$$

with a balancing parameter μ set as 0.1 by default.

We minimize the loss (3.1) with an Adam optimizer. The learning rate is set as 1e-3 and decays to 1e-4 at discrete intervals during training.

3.4 Experimental results and applications

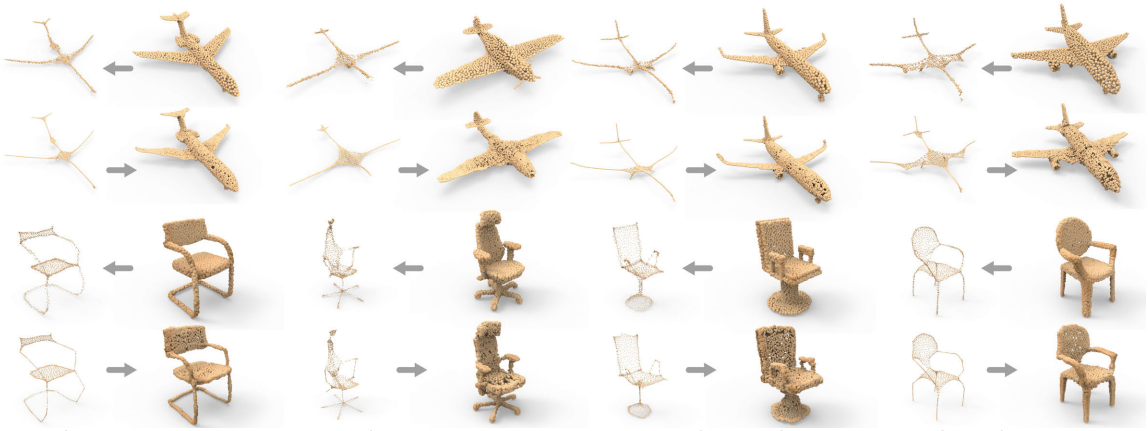
We conduct experiments to demonstrate the capability of P2P-NET in learning geometric transforms between point sets in various domains. Throughout the experiments, the network was trained using different datasets and for different domain pairs separately, but always with the same default network settings as described in Section 3.3 and Appendix 3.6.1. There is no hyperparameter or architecture tuning for any specific domains or any specific datasets. All the results are presented without any post-processing.

3.4.1 Meso-skeleton, surface, and single-view point scan

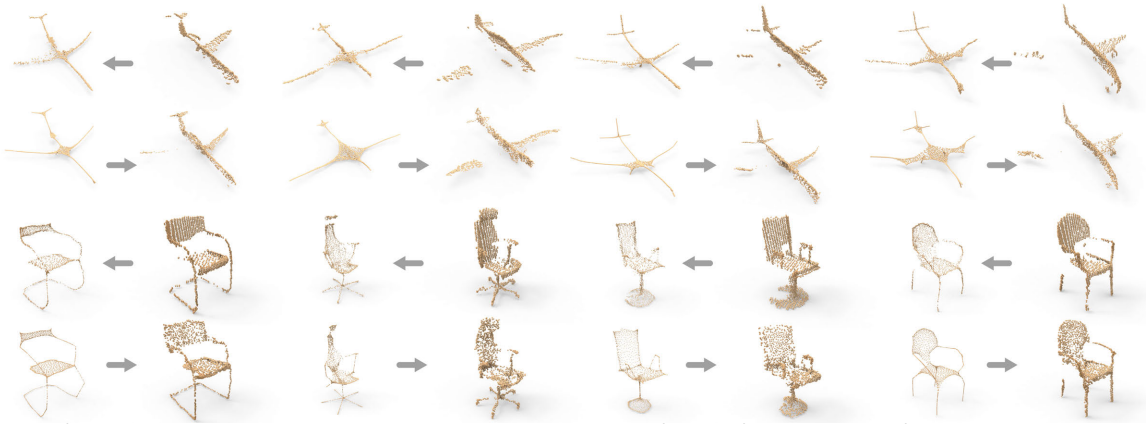
In many cases, a mapping from one domain to another is easy, but the inverse is a lot more difficult. For example, synthesizing point scans from 3D shapes is easy, but surface completion is hard. Skeleton extraction from 3D shapes may have been considered as a solved problem [19], but synthesizing shape surfaces from skeletons is an unresolved challenge. Our network is able to learn to solve ill-posed inverse mapping problems (e.g., skeleton-to-surface) by using training data synthesized by an algorithm designed for the easier transform (e.g., surface-to-skeleton). In this section, we demonstrate transformations among meso-skeletons, surface samples, and single-view point scans using P2P-NET.

Given a set of 3D shapes, we convert them to surface samples with Poisson disk sampling [123]. By taking the surface samples as input, meso-skeletons of the shapes are obtained using a contraction-based approach [22]. To show the robustness of our network to shape occlusion, we also synthesize single view point scans with a Kinect simulator [124, 125] applied to 3D shapes. We use the chair and airplane datasets of ModelNet40 [43] as original 3D shapes, and sample each point set to the size of 2,048. The chair dataset contains 889 training and 100 test examples, while the airplane dataset contains 626 training and 100 test examples.

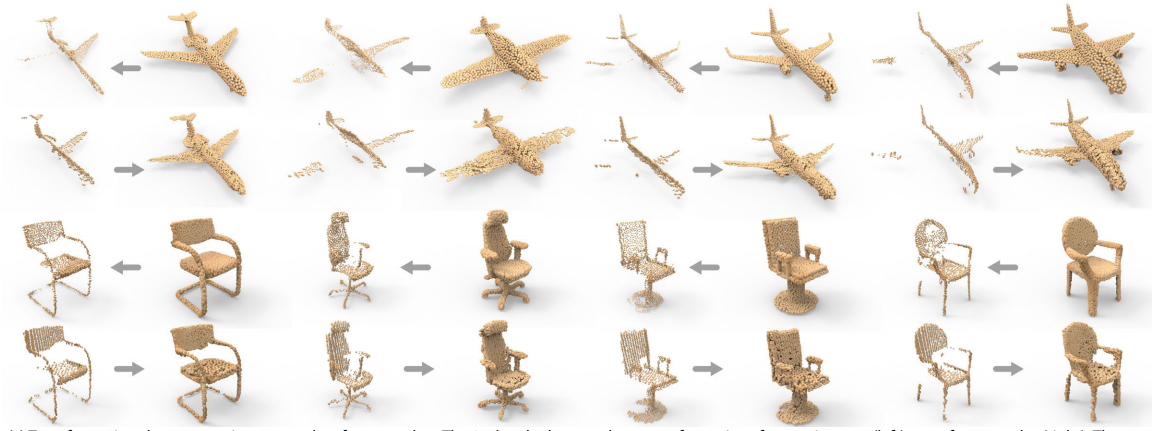
With the synthesized surface samples, meso-skeletons, and single-view point scans, we tested our method on three pairs of transformations among the three different types of point sets, i.e., meso-skeleton vs. surface, meso-skeleton vs. point scan, and point scan vs. surface. In Fig. 3.6, the visual results of the three pairs of transformations are provided with eight distinctive examples chosen from the test set. Note that, in order to obtain transformed surface point samples, we feed the same input in eight passes to P2P-NET and integrate the network outputs to produce a dense final result. To obtain point scans or meso-skeletons, we only feed the input once to the network.



(a) Transformations between meso-skeleton and surface samples. The 2nd and 4th rows show transformations from meso-skeleton (left) to surface samples (right). The 1st and 3rd rows show transformations from surface samples (right) to meso-skeleton (left).



(b) Transformations between meso-skeleton and point scan. The 2nd and 4th rows show transformations from meso-skeleton (left) to point scan (right). The 1st and 3rd rows show transformations from point scan (right) to meso-skeleton (left).



(c) Transformations between point scan and surface samples. The 2nd and 4th rows show transformations from point scan (left) to surface samples (right). The 1st and 3rd rows show transformations from surface samples (right) to point scan (left).

Figure 3.6: A gallery of point set transformations among meso-skeletons, shape surfaces, and single-view point scans via our network P2P-NET. Note that, to obtain the transformed surface point samples, we feed the same input eight times to the network and integrate the network outputs to produce a dense point set.

To convey that our network is able to learn a shape transform, we show the closest training examples retrieved for the inputs over the eight test examples in Appendix 3.6.2. We also provide quantitative evaluations for the three pairs of transformations in Section 3.4.3.

Figure 3.7: Testing P2P-NET on real chair scans (middle) captured by Kinect v2. The completed point cloud is shown on the right.

Fig. 3.5 visualizes point-wise displacements produced by P2P-NET to offer a glimpse of what the network learned. We re-emphasize that the network was not trained on any point-wise mapping between paired shapes nor with any displacement vectors. Yet, it appears that the learned displacements are well-localized and reflect what a properly devised transformation algorithm would produce.

In Fig. 3.7, we show that the trained P2P-NET is capable of converting real point scans of chairs captured by a Kinect v2 to complete shapes. Note that during the capture, the Kinect sensor was placed to roughly align with the camera view used in data synthesis.

The results shown in Fig. 3.6 demonstrate the potential of P2P-NET for possible applications. In Fig. 3.8, we show such an example for shape editing and synthesis. After combining the meso-skeletons of different shapes into a new meso-skeleton, our network can convert the synthesized meso-skeleton into a new point-set shape. Moreover, the result of transforming point scans to surface samples offers the promise of applying P2P-NET for scan completion. To extend P2P-NET to a full-fledged scan completion network, one would require a multi-view assembly of the network or adding view prediction and rotation layers, which are out of the scope of this paper.

3.4.2 2D cross-sectional profiles and 3D shapes

Planar cross-sectional profiles are widely used in computer-aided design and geometric modeling. Transforming from 2D cross sections to 3D shapes is an interesting test for our neural network. For this experiment, we use the sofa and bed datasets of ModelNet40. The sofa dataset contains 680 training and 100 test examples. Similarly, in the bed dataset, there are

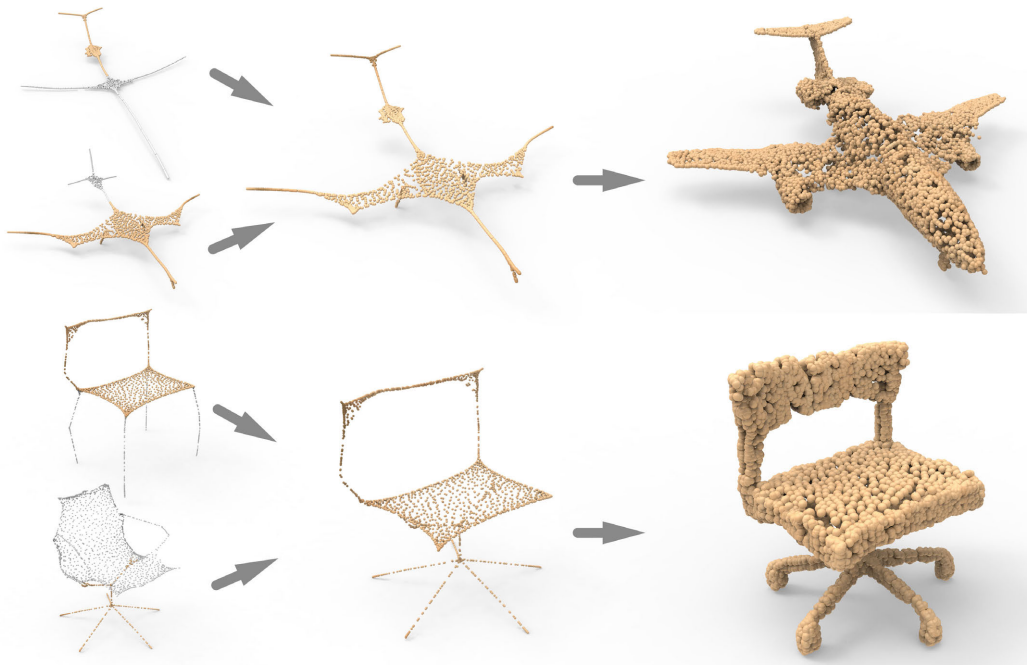


Figure 3.8: After editing and combining the point sets of meso-skeletons, our network is able to generate new shapes (right) from the new meso-skeletons.

515 training and 100 test examples. We cut each sofa with four parallel planes to obtain four parallel cross sections, and cut each bed with three orthogonal planes to obtain three orthogonal cross sections. We sample each set of cross sections uniformly to acquire a point set consisting of 2,048 points. Each point set of cross sections is then paired with the point set of mesh samples of the same sofa or bed object.

We visualize the results obtained on eight typical test examples of the sofa and bed in Fig. 3.9. We can observe that the transformation results obtained with feeding the network a single input pass exhibit non-uniformity and missing regions. However, after feeding the input over eight passes and integrating the network outputs, the resulting dense point sets are complete and smooth overall. This demonstrates the stability of the transformation prediction by P2P-NET.

It is also interesting to observe that, in some cases, the dense outputs produced by multiple passes of P2P-NET can better convey shape details than the ground truth data, e.g., see column (c) in Fig. 3.9 for the long pillow in the fifth row and the slats in the crib in the second to last row, in contrast to their counterparts in column (d). One reason is that the ground truth is only at 1/8 of the resolution, compared to the dense results. On the other hand, the level of surface details produced by the network are not copied from the training set, since all training data are at a low resolution of 2,048 points which do not well reflect the surface details. The produced details should be attributed to the point transforms learned by P2P-NET.

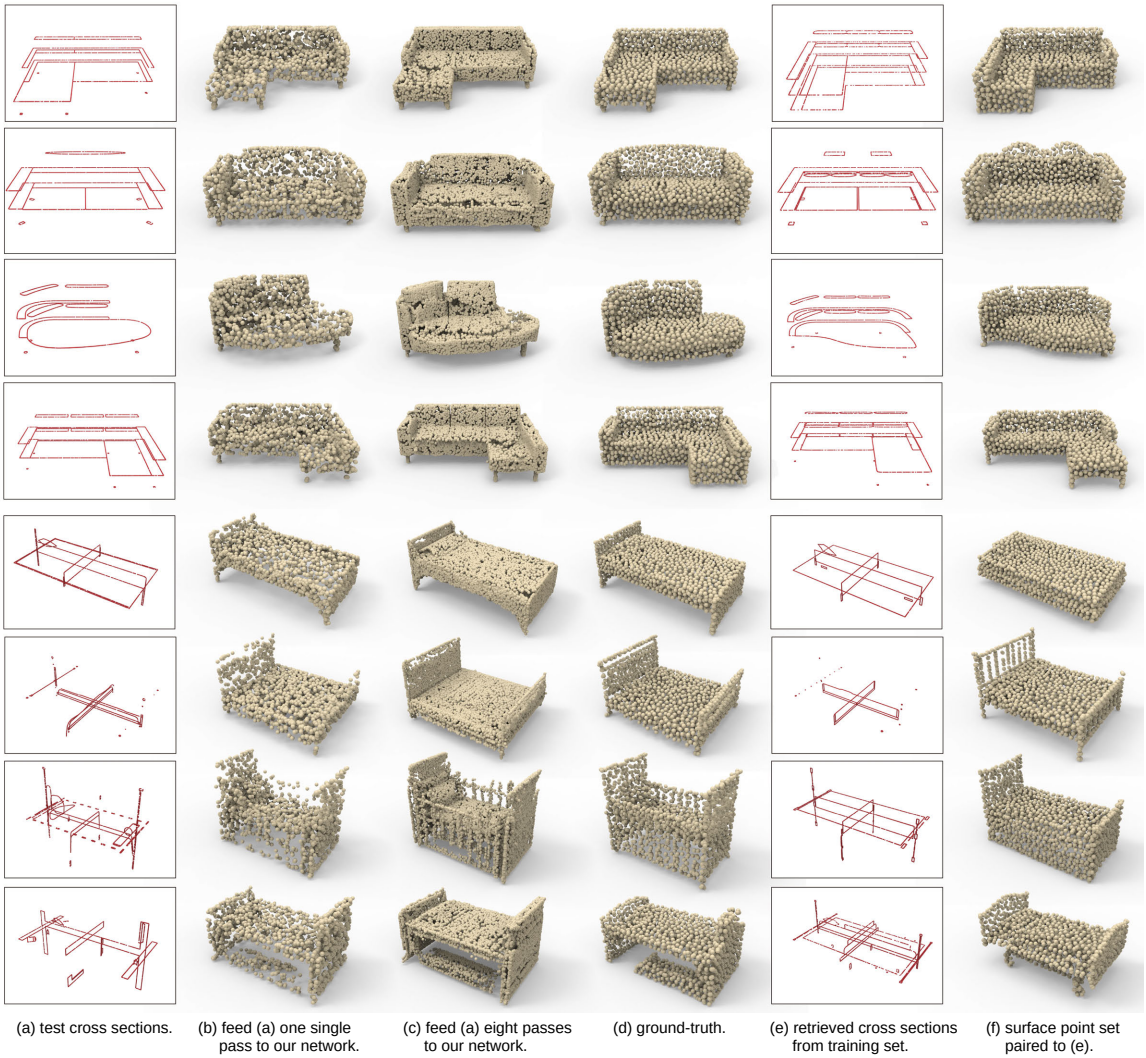


Figure 3.9: Transformations from 2D cross-sectional profiles (a) to 3D object surfaces (b) and (c). In addition to ground-truths (d), we also provide the closest 2D cross-sectional profiles (e) retrieved from the training set, and their corresponding surface point sets (f).

To further demonstrate that the network has learned a proper transform, we retrieved the closest training cross sections with the test cross sections as query inputs, and show the retrieved cross sections and their paired sampled meshes in Fig. 3.9. The retrieval was carried out using the distance measure:

$$D_{\text{retrieve}}(P,Q) = \sum_{p\in P} \min_{q\in Q} \|p-q\| + \sum_{q\in Q} \min_{p\in P} \|p-q\|\,,$$

where P and Q are two point sets of cross sections.

We can observe that the retrieved cross sections are generally not close to the queries. This is clearly evident in the last three rows of Fig. 3.9. Admittedly, it is far from trivial

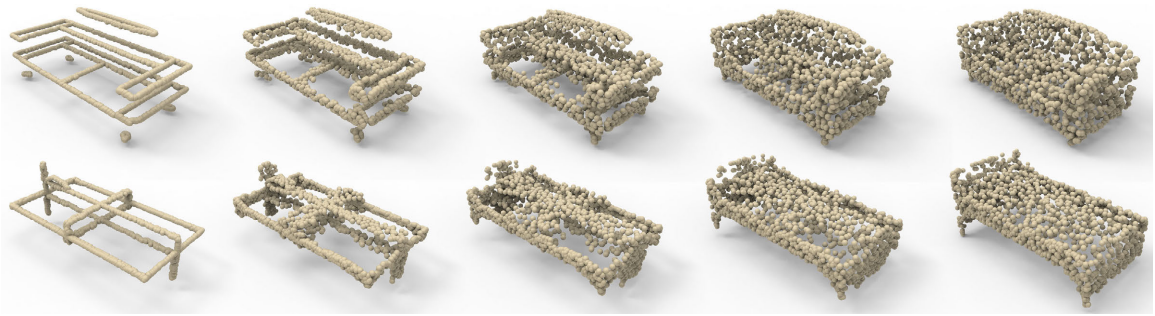


Figure 3.10: Visualization of point displacements learned by P2P-NET, which transform cross-sectional profiles into surface samples. We scaled the displacements, from left to right, by factors of 0.05, 0.25, 0.5, 0.75, 1.0, respectively, to obtain a morphing sequence.

to come up with an accurate similarity distance measure for cross-sectional profiles. To confirm that the retrieved results are reasonable, we have manually examined all the training examples in the dataset and found no other cross sections to be visually closer to those shown in the figure.

In Fig. 3.10, we visualize the displacement vectors learned for the current transform. Since mappings from cross-sectional profiles to 3D shapes are much less predictable and coherent, instead of showing the displacement vectors explicitly like in Fig. 3.5, we show a morphing sequence following the displacements.

What is common between skeleton-to-shape and profile-to-shape transforms is that one domain has an *easy-to-edit* shape abstraction. It is quite common to perform user edits on skeletons and curve profiles. After that, it would be quite desirable to be able to directly convert the edited shape abstractions to whole shapes. Like the example shown in Fig. 3.8, we also experimented with editing 2D cross-sections and then transforming the edits to 3D point-set shapes using P2P-NET. A visual result showing an interpolating sequence is provided in Fig. 4.1. This further demonstrates the potential of our network in shape synthesis applications.

3.4.3 Quantitative evaluation

We evaluated the performance of our P2P-NET quantitatively on the four datasets used in Sections 3.4.1 and 3.4.2. For the purpose of measuring the errors, the original shapes and their point sets were normalized, so that the diagonal lengths of their bounding boxes are equal to 1. It should be remembered that all the training and test point sets are sampled to the size of 2,048. The performances were measured with three error metrics, as detailed below.

Point separation rate. Given a predicted point set and the ground-truth point set, each point searches for its closest point from the opposite set. If the distance from a point p to its closest point q in the opposite set is greater than 0.02, we consider the point p as a

Table 3.1: Quantitative evaluation of our network with different settings and error metrics. In the head of the table, 'ns' stands for noise augmentation, 'rg' stands for cross regularization, and '+/-' indicates enable/disable.

Dataset	Source	Target	mean of separation rate			mean of curvature diff.			mean of normal diff.		
			ns-rg-	ns+rg-	ns+rg+	ns-rg-	ns+rg-	ns+rg+	ns-rg-	ns+rg-	ns+rg+
airplane	skeleton	surface	1.4%	**0.6%**	**0.6%**	0.084	0.063	**0.062**	0.575	0.390	**0.389**
	surface	skeleton	**0.3%**	0.4%	**0.3%**	0.076	0.076	**0.075**	-	-	-
	scan	skeleton	2.1%	**2.0%**	**2.0%**	0.068	0.068	**0.066**	-	-	-
	skeleton	scan	2.9%	**2.3%**	2.4%	**0.051**	0.052	0.052	0.677	0.617	**0.601**
	scan	surface	1.6%	**1.2%**	1.3%	0.074	0.063	**0.061**	0.495	0.444	**0.418**
	surface	scan	1.3%	**1.1%**	1.3%	**0.056**	0.057	0.057	0.669	0.670	**0.667**
chair	skeleton	surface	12.0%	**7.5%**	**7.5%**	0.096	0.082	**0.080**	0.686	0.620	**0.617**
	surface	skeleton	5.3%	5.3%	5.3%	0.061	0.061	**0.060**	-	-	-
	scan	skeleton	15.5%	15.8%	**15.4%**	0.053	0.053	**0.051**	-	-	-
	skeleton	scan	18.8%	17.6%	**17.5%**	**0.052**	0.054	**0.052**	0.590	0.584	**0.562**
	scan	surface	10.9%	**6.6%**	6.7%	0.092	**0.083**	0.084	0.613	0.557	**0.553**
	surface	scan	5.2%	5.2%	**4.9%**	0.057	0.057	**0.056**	0.553	0.556	**0.552**
sofa	cross sec.	surface	22.0%	**9.8%**	10.8%	0.084	0.066	**0.065**	0.626	**0.457**	0.458
	surface	cross sec.	9.6%	9.2%	**8.9%**	**0.059**	0.060	**0.059**	-	-	-
bed	cross sec.	surface	14.5%	3.0%	**2.9%**	0.084	**0.056**	**0.056**	0.544	**0.380**	**0.380**
	surface	cross sec.	**11.0%**	11.8%	11.5%	0.058	0.058	0.058	-	-	-

separated point. We call the percentage of separated points among all points in the two sets, the separation rate. We compute a separation rate for every test example and report the mean in Table 3.1.

Curvature difference. We estimate a curvature indicator for each point p as $\lambda_0/(\lambda_0 + \lambda_1 + \lambda_2)$, where $\lambda_0 \leq \lambda_1 \leq \lambda_2$ are the eigenvalues of a 3×3 covariance matrix of a local point patch around point p. The size of the local patch is 0.3% of the size of whole point set. For each point p, we compute the absolute difference of curvature indicator with its closest point q in the opposite point set. Finally, we compute the mean difference of all points.

Normal difference. Similar to the curvature indicator, we estimate a PCA normal for each point, and measure the radian of the angle between the normal of a point p and the normal of its closest point q in the opposite set. We compute the mean radian of all points.

For each pair of transformations on each dataset, we trained the networks for 200 epoches on a Nvidia Titan Xp GPU that takes approximately 5∼8 hours to finish the process. During the testing phase, when the target point set is not from surface samples, we feed the source point set in one single pass to the network and obtain output point sets of size 2,048. When the target point set is from surface samples, we feed the source point set in eight passes to the network to obtain an integrated dense output of size 16,384. Four pairs of transformations and three different settings of the network were tested and reported in Table 3.1. The three different network settings are: no noise augmentation and no cross regularization (ns-rg-), no noise augmentation but with cross regularization (ns-rg+), and the setting with both options on (ns+rg+).

Note that the separation rate measures the tightness of the match between the predicted point set and the ground-truth point set. The smaller the value is, the tighter the match.

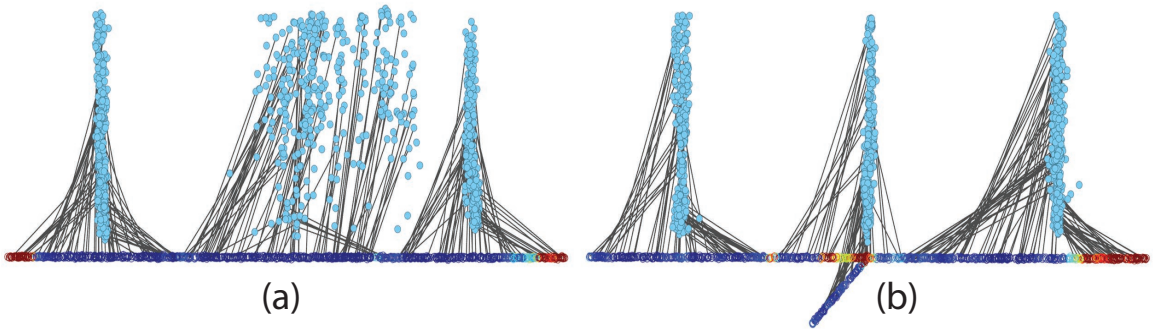


Figure 3.11: P2P-NET cannot be properly trained to map a long horizontal line to three vertical bars, since points near the mid-section possess similar point features. Our method fails to associate points with similar features with dissimilar displacements to clearly form the middle bar (a). However, a small added protrusion (b) can serve to disambiguate these point features, leading to different displacements to produce the middle bar. Colorings of the points reflect their features, after a 1D embedding using PCA.

The results in Table 3.1 show that noise augmentation helps to reduce the separation rate, i.e., to make the predicted points to match more tightly to the ground-truth point sets. Adding the cross-regularization does not further reduce the separation rate. However, as shown in Table 3.1, adding the cross-regularization term does achieve lower error rates overall, in terms of curvature difference and normal difference. Since these two measures reflect how well local geometric properties of the point set are preserved, the quantitative results demonstrate that cross-regularization is effective in enhancing the local geometric properties of the predicted point set.

3.5 Discussion and limitation

By design, P2P-NET is a general-purpose point-to-point displacement network, in that no parts of the network are tailor-made to specific transformation tasks. Moreover, we do not alter the network architecture, when dealing with different pairs of transformation domains. The network is trained to map point sets from one domain to another, where the point sets can be in 2D or 3D spaces. As we demonstrated, the mapping can also lift 2D profiles to 3D shapes; see Fig. 3.9. Since the mapping is applied in a feature space, the learned transform is agnostic to the dimensionality of the point sets. Interestingly, the point displacements, which are one-to-one, are learned without training data on point-wise mapping or displacement vectors. All we provide are pairs of point-set shapes, which may even have different cardinalities.

P2P-NET is bidirectional, which may be reminiscent of networks trained under cycle consistency [5, 6]. However, there is no cyclic consistency in P2P-NET; the bidirectionality is used to form a cross-regularization which exploits the two mappings to enhance the mapping distribution. There is an intriguing, and seemingly "dual", relation between P2P-

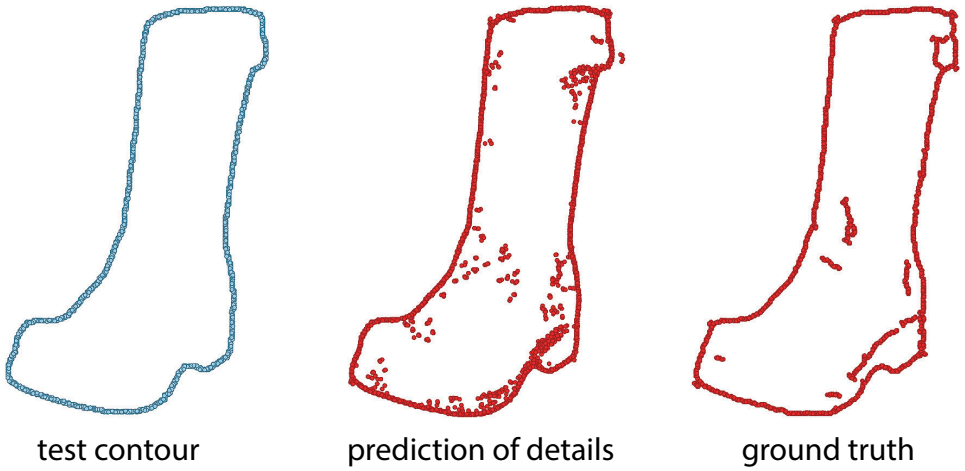


Figure 3.12: P2P-NET cannot be properly trained for non-deterministic point set transforms, e.g., adding details to a shape contour.

NET and CycleGAN [5]. P2P-NET is trained on paired shapes, while CycleGAN learns from unpaired images. But in CycleGAN, there is pixel-to-pixel correspondence between the pair of training images; P2P-NET does not require point-to-point correspondence between the training point-sets. CycleGAN is trained to learn how to transform pixel values, *in place*, while Point-NET is trained to displace points from one shape representation to another.

We reiterate that our work is only a first attempt at designing a general-purpose shape transformation network. By no means should one expect P2P-NET to work effectively for all transformation tasks. The network is inherently limited by its current architecture, training loss, and optimization scheme for the network parameters. In what follows, we provide a non-exhaustive list of such limitations to explore the behavior and limit of our method.

Ambiguous feature-to-displacement mapping. In the absence of point-to-point correspondences between the training source and target shapes, P2P-NET must learn point set transforms implicitly. Architecturally, P2P-NET first turns the input points into Point-NET++ features. It then learns to map these point features to displacement vectors to minimize the training loss. As a result, P2P-NET should be trained with examples, where the implicit relation between point features and displacements is *unambiguous*. The network should not be expected to learn to associate points possessing similar features with *different* displacements. In reality, however, ambiguous feature-to-displacement mappings may be unavoidable for many transformation tasks. They may be characteristic of an entire class of transformations or occur only for some shape pairs or only over a portion of the shapes. Any such case may potentially lead to failure cases by P2P-NET.

To provide a simple illustration, consider a 2D example of learning to map points along a long horizontal line to three vertical bars, as shown in Fig. 3.11. We trained P2P-NET using more than 1,000 examples of source and target pairs in random orientations and

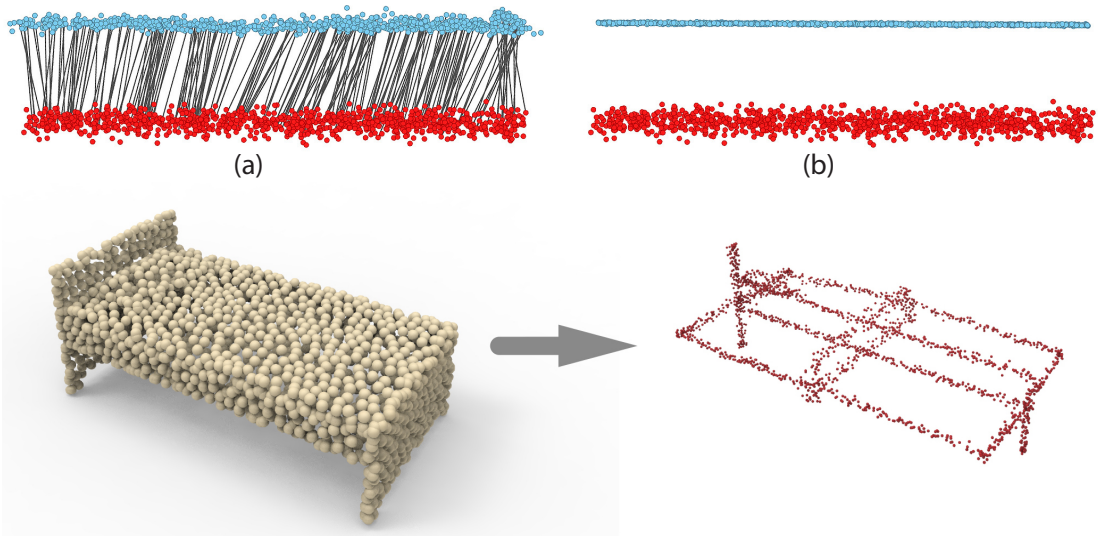


Figure 3.13: P2P-NET cannot be trained to accurately and cleanly predict thin structures. Top: from a noisy input (red), as shown in (a), in contrast to the ground truth (b). Bottom: from lower-resolution point set inputs.

scales. However, regardless of how many training examples we employed, the network still cannot map the mid-section of the line to the middle bar, since points near the mid-section all possess similar features. As shown in (a), P2P-NET could only learn to associate these points with similar displacements. To verify that the crux of the problem is the ambiguity, we added a small protrusion under the source line, so that points near the mid-section can be better distinguished by PointNET++ features. As can be seen in (b), P2P-NET now does a much better job of learning the proper transform.

Ambiguous point transforms. Some point set transforms may exhibit shape-level ambiguities, as shown in Fig. 3.12. In this task, we learn displacements from a shape silhouette to its interior details. The training set contains more than 1,000 examples of adding different details (via edge maps) to different boot shapes. To minimize the training loss, P2P-NET is only able to learn to displace to an *average* of the target points, leading to an erroneous outcome.

Decorrelation of displacements. The training loss adopted by P2P-NET is predominantly a point-to-shape distance measure. It does not account for intrinsic properties of the input shape. This immediately implies that P2P-NET is generally unable to learn such properties, so as to preserve them in the output point clouds. The point displacement vectors predicted by P2P-NET are not correlated or controlled by the shape properties, since the network predicts a displacement vector for each point independently. Fig. 3.13 shows that P2P-NET is unlikely to reproduce thin lines, when all the training data contain clean, thin line structures. Fig. 3.14 shows that P2P-NET cannot be well trained to produce highly

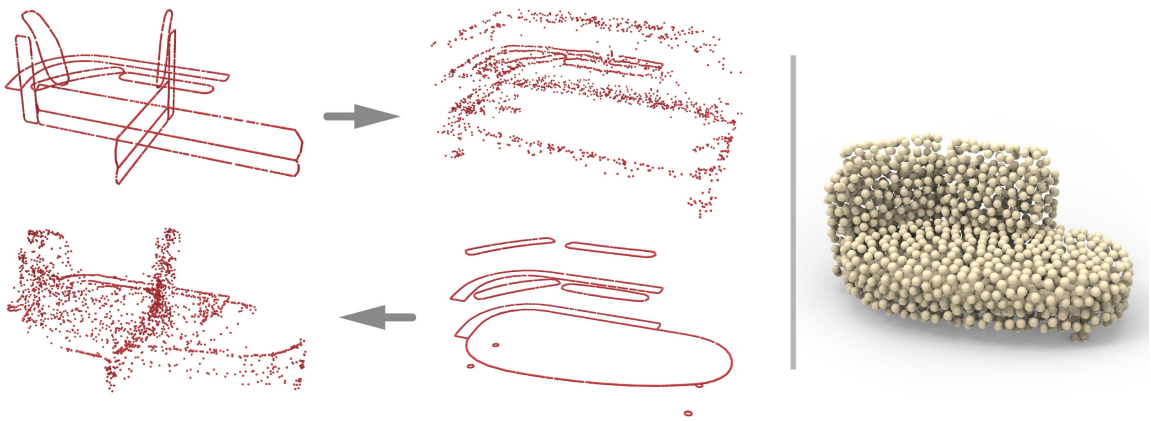


Figure 3.14: P2P-NET cannot be trained to produce highly structured point sets cleanly, e.g., to transform between orthogonal profiles and parallel profiles for the same 3D shape (sofa on the right).

structured point sets, where the transform is between orthogonal profiles and parallel profiles of the same shapes. By the same token, P2P-NET is not part-aware, i.e., it is unlikely to preserve part structures of the input shapes. For example, it cannot transform clean rooms into messy rooms, by displacing or adding point-set objects. Currently, only uncorrelated point-wise displacements are learned.

Future work. In addition to addressing the limitations discussed so far, a network capable of transforming point sets *hierarchically* is likely to produce more fine-grained results and adapt to more domains. We would also like to consider *transitive* transformations, where a source shape reaches a target via a sequence of two networks through an intermediate shape. On this shape, the points can be upsampled, consolidated, filtered or undergo any other processing operation. This may also be generalized to combining and composing transformations. Finally, an intriguing avenue for future research would be to relax the need for paired shapes, and replace it with an unsupervised or weakly supervised setting to train a general-purpose network for point set transforms.

3.6 Appendix

3.6.1 Details of network architecture

In this appendix, we provide details of the set abstraction layers, feature propagation layers, and fully connected layers in P2P-NET. We use the same notations as in [50]. A set abstraction layer of PointNet++ is denoted as $SA(K, r, [l_1, .., l_d])$, where K is number of local patches, r is radius of balls that bound the patches, $[l_1, .., l_d]$ are widths of fully connected layers used in local PoinNet. A feature propagation layer is denoted as $FP([l_1, .., l_d])$, where $[l_1, .., l_d]$ are widths of fully connected layers used inside the layer. A fully connected layer is denoted as $FC(l)$, where l is its width. Note that we disabled dropout for the FC layers.

For all experiments shown, we used the same A-P layers:

$$\begin{aligned}&\text{input} \rightarrow SA(1024, 0.1, [64, 64, 128]) \rightarrow SA(384, 0.2, [128, 128, 256])\\ &\rightarrow SA(128, 0.4, [256, 256, 512]) \rightarrow SA(1, 1.0, [512, 512, 1024])\\ &\rightarrow FP([512, 512]) \rightarrow FP([512, 256]) \rightarrow FP([256, 128])\\ &\rightarrow FP([128, 128, 128]) \rightarrow \text{feature}\end{aligned}$$

We also used the same fully connected layers:

$$[\text{feature, noise}] \rightarrow FC(128) \rightarrow FC(64) \rightarrow FC(3) \rightarrow \text{displacements}$$

3.6.2 Closest training examples

In Figs. 3.15 and 3.16, we show the closest models from the training set that are retrieved for the test examples used in Section 3.4.1. The retrieval was done by searching for a training example having the closest surface samples to an input test example. To measure the difference between surface samples, we use the distance measure $D_{\text{retrieve}}(P, Q)$ as described in Section 3.4.2.

Figure 3.15: Closest training examples for the test airplanes in Fig. 3.6.

Figure 3.16: Closest training examples for the test chairs in Fig. 3.6.

Chapter 4

Unpaired Shape Transform in Latent Overcomplete Space

4.1 Introduction

Shape transform is one of the most fundamental and frequently encountered problems in computer graphics and geometric modeling. With much interest in geometric deep learning in the graphics community today, it is natural to explore whether a machine can learn shape transforms, particularly under the *unsupervised* setting. Specifically, can a machine learn to transform a table into a chair or vice versa, in a natural way, when it has only seen a set of tables and a set of chairs, without any pairings between the two sets?

In recent years, the intriguing *unpaired domain translation* problem has drawn much interest in computer vision and computer graphics, e.g., the domain transfer network (DTN) [126], CycleGAN [5], DualGAN [6], MUNIT [96], among others [80, 127, 128, 129, 130]. However, most success on unpaired image-to-image translation has been achieved only on transforming or transferring *stylistic* image features, not shapes. A symbolic example is the CycleGAN-based cat-to-dog transfiguration which sees the network only able to make minimal changes to the cat/dog shapes [5]. The recently developed P2P-NET [131] is able to learn general-purpose shape transforms via point displacements. While significant shape changes, e.g., skeleton-to-shape or incomplete-to-complete scans, are possible, the training of P2P-NET is *supervised* and requires *paired* shapes from two domains.

In this paper, we develop a deep neural network aimed at learning *general-purpose* shape transforms from *unpaired domains*. The network is trained on two sets of shapes, e.g., tables and chairs or different letters, each represented using a point cloud. There is neither a pairing between shapes in the two domains to guide the shape translation nor any point-wise correspondence between any shapes. Once trained, the network takes a point-set shape from one domain and transforms it into the other.

Without any point correspondence between the source and target shapes for the transform, one of the challenges is how to properly "normalize" the shapes, relating them so

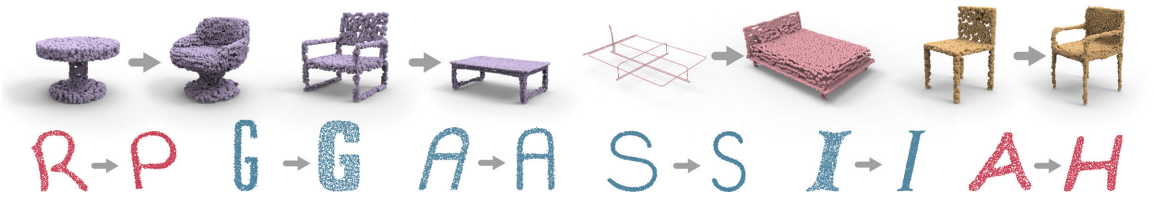


Figure 4.1: We present LOGAN, a deep neural network which learns *general-purpose* shape transforms from *unpaired* domains. By altering only the two input data domains for training, without changing the network architecture or any hyper-parameters, LOGAN can transform between chairs and tables, from cross-sectional profiles to surfaces, as well as adding arms to chairs. It can also learn both style-preserving content transfer (letters $R \rightarrow P$, $A \rightarrow H$, in different font styles) and content-preserving style transfer (wide to narrow S, thick to thin I, thin to thick G, and italic to non-italic A.)

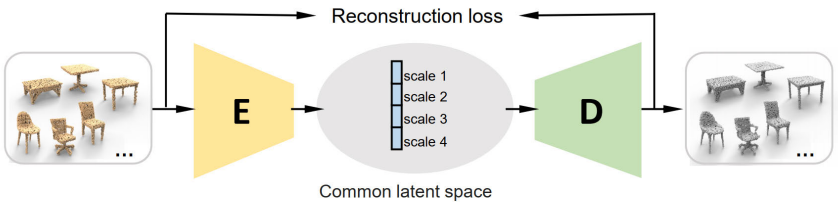
as to facilitate their translation. To this end, we perform shape translation in a *common latent* space shared by the source and target domains, rather than on the point-set shapes directly. The latent space is obtained by an *autoencoder* trained prior to shape transform; see Figure 4.2(a).

More importantly, a proper shape transform from chairs to tables should not translate a given chair to any table, but to a table that is clearly from that particular input chair. Hence, some features of the chair that are also common to tables should be preserved during a chair-table translation while the other features can be altered. This poses a key challenge: what features are to be preserved/altered is unknown — it must depend on the given shape domains and our network must learn it without supervision. To this end, our network is designed with two novel features to address this challenge:

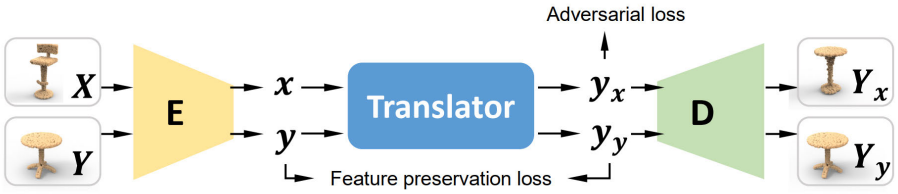
- Our autoencoder encodes shape features at *multiple scales*, which is common in convolutional neural networks (CNNs). However, unlike conventional methods which aggregate the multi-scale featuers, e.g., in PointNET++ [50], we *concatenate* the multi-scale features to produce a latent code which is "*overcomplete*". Specifically, the input shape can be reconstructed using only parts of the (overcomplete) code corresponding to different feature scales; see Figure 4.5.

  Our intuition is that performing shape transforms in the latent space formed by such overcomplete codes, where multi-scale feature codes are *separated*, would facilitate an *implicit disentangling* of the preserved and altered shape features — oftentimes, these features are found at different scales.

- In addition, our chair-to-table translator is not only trained to turn a chair code to a table code, but also trained to turn a table code to the *same* table code, as shown in Figure 4.2(b). Our motivation for the second translator loss, which we refer to as the *feature preservation loss*, is that it would help the translator preserve table features (in an input chair code) during chair-to-table translation.



(a) Autoencoder encodes shapes into overcomplete latent codes.



(b) Translator networks in latent space.

Figure 4.2: Overview of our network architecture, which consists of an autoencoder (a) to encode shapes from two input domains into a common latent space which is overcomplete, and a GAN-based translator network (b) designed with an adversarial loss and a loss to enforce feature preservation.

Figure 4.2 shows an overview of our network architecture, with shape translation operating in a latent space produced by an overcomplete autoencoder. The translator network itself is built on the basic framework of generative adversarial networks (GANs), guided by an adversarial loss and the feature preservation loss. We call our overall network a *latent overcomplete GAN*, or LOGAN for short, to signify the use of GANs for shape-to-shape translation in a common, overcomplete, latent space. It is also possible to train a dual pair of translators between the source and target domains, reinforcing the results with an additional cyclic loss [5].

Overall, LOGAN makes the following key contributions:

- To the best of our knowledge, it is the first deep model trained for general-purpose, unpaired shape-to-shape translation.
- It is the first translation network that can perform *both* content and style transfers between shapes (see Figure 4.3), without changing the network's architecture or any of its hyper-parameters. Owing to the overcomplete, multi-scale representations it learns, LOGAN adapts its feature preservation solely based on the two input domains for training.
- It puts forth the interesting concept of implicit feature disentanglement, enabled by the overcomplete representation, which may hold potential in other application settings.

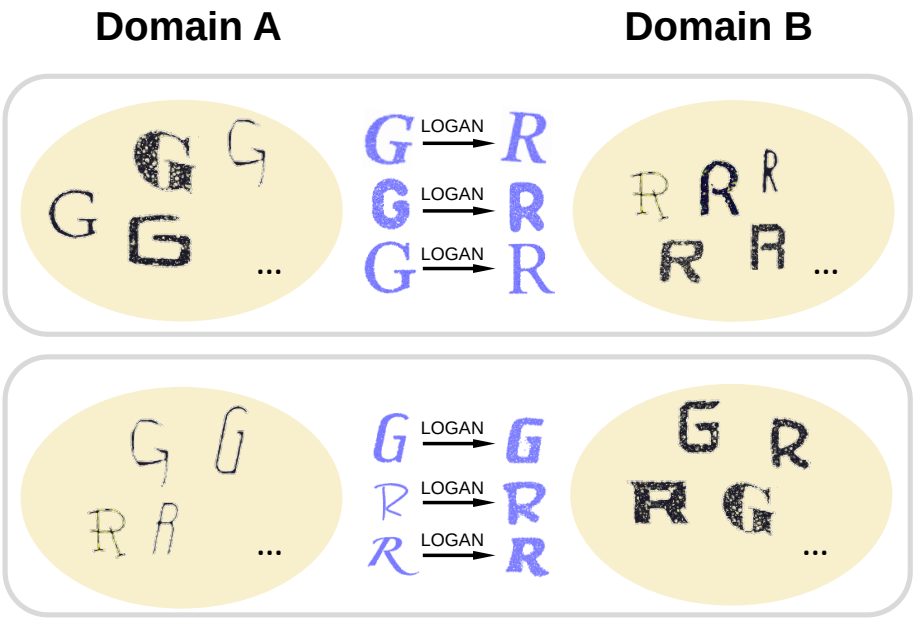


Figure 4.3: Depending solely on the (unpaired) input training domains, our network LOGAN can learn both content transfer (top row: from letters G to R, in varying font styles) and style transfer (bottom row: from thin to thick font strokes), without any change to the network architecture. In this example, 2D letters are represented by dense clouds of 2D points.

We conduct ablation studies to validate each of our key network designs: the autoencoder, the multi-scale and overcomplete latent codes, as well as the feature preservation loss. We demonstrate superior capabilities in unpaired shape transforms on a variety of examples over baselines and state-of-the-art approaches. We show that LOGAN is able to learn what shape features to preserve during shape transforms, either local or non-local, whether content or style, etc., depending solely on the input domain pairs; see Figure 4.1.

4.2 Related work

Computing image or shape transforms is a fundamental problem in visual data processing and covers a vast amount of literature. In the classical setting for shape transforms, source shapes are deformed into target shapes anchored on corresponding points or parts. The key is how to formulate and compute deformation energies to ensure detail preservation [132, 133], structure preservation [134, 135], or topology variation [136, 137]. On the other hand, our work is related to learning general-purpose, cross-domain image/shape transforms. As such, we mainly cover learning-based methods from vision and graphics that are most closely related to our approach.

Unpaired image-to-image translation A wave of notable works on unsupervised/unpaired cross-domain image translation have emerged in 2017. In DTN, Taigman et al. [126] train

a GAN-based domain transfer network which enforces consistency of the source and generated samples under a given function f. For example, f can capture face identities, allowing their network to generate identity-preserving emojis from facial images. CycleGAN [5] and DualGAN [6] both train dual translators with a cyclic loss to address the challenge of unpaired image-to-image translation. However, by performing the translations on images directly, based on pixel-wise losses, these methods perform well on transferring stylistic images features, but poorly on shape transforms.

Dong et al. [138] train a conditional GAN to learn shared global features from two image domains and to synthesize plausible images in either domain from a noise vector and a domain label. To enable image-to-image translation, they separately train an encoder to learn a mapping from an image to its latent code, which would serve as the noise input to the conditional GAN. In UNIT, Liu et al. [80] assume that corresponding images from two domains can be mapped to the *same code* in a shared latent space. Based on this assumption, they train two GANs, coupled with weight sharing, to learn a joint distribution over images from two unpaired domains. By sharing weight parameters corresponding to high level semantics in both the encoder and decoder networks, the coupled GANs are enforced to interpret these image semantics in the same way.

Architecturally, there are some similarities between LOGAN and UNIT [80]. Both networks take inputs from two domains, map them into a latent space, and enforce some notion of "self-reconstruction". However, LOGAN does not make the shared latent space/code assumption: it is not aiming to map the two inputs into the same latent code. Moreover, the notion of self-reconstruction in UNIT is realized by a variational autoencoder (VAE) loss and the VAE is trained together with the GANs. In contrast, LOGAN trains its autoencoder and translator networks separately, where the notion of self-reconstruction is applied to latent codes in the translators, via the feature preservation loss.

Identity loss in domain translation Our feature preservation loss is equivalent, in form, to the *identity loss* in the DTN of Taigman et al. [126] for reinforcing face identity preservation during emoji generation; it was later utilized in CycleGAN [5] as an additional regularization term for color preservation. In our work, by enforcing the same loss in latent space, rather than on images directly, and over the multi-scale overcomplete codes in LOGAN, we show that the loss can play a critical role in feature preservation for a variety of shape transformation tasks. The preserved features can be quite versatile and adapt to the input domain pairs.

Disentangled representations for content generation Disentangled image representations have been utilized to produce many-to-many mappings so as to improve the diversity of unsupervised image-to-image translation [139, 96]. Specifically, in MUNIT, a multi-modal extension of UNIT [80], Huang et al. [96] relax the assumption of fully shared latent space between the two input domains by postulating that only part of the latent space, the *con-*

tent, can be shared whereas the other part, the *style*, is domain-specific. Their autoencoders are trained to encode input images into a disentangled latent code consisting of a content part and a style part. During image translation, a fixed content code is recombined with a random style code to produce diverse, style-transferred target images. Most recently, Press et al. [140] learn disentangled codes in a similar way, but for a different kind of unsupervised content transfer task, i.e., that of adding certain information, e.g., glasses or facial hair, to source images.

In contrast, LOGAN does not learn a disentangled shape representation explicitly. Instead, our autoencoder learns a multi-scale representation for shapes from both input domains and explicitly assigns encodings at different shape scales to sub-vectors of the latent codes. Specifically, codes for shapes from different domains are *not* enforced to share any sub-codes or content subspace; the codes are merely constructed in the same manner and they belong to a common latent space. Feature preservation during shape translation is enforced in the translator networks, with the expectation that our overcomplete latent representation, with feature separation, would facilitate the disentangling of preserved and altered features. Note that in other contexts and for other applications using CNNs, there have been works, e.g., [141, 15, 142], which also encode and concatenate multi-scale features.

Learning shape motions and transforms Earlier work on spatial transformer networks [119] allows deep convolutional models to learn invariance to translation, scale, rotation, and more generic shape warping for improved object recognition. Byravan and Fox [3] develop a deep neural network to learn rigid body motions for robotic applications, while deep reinforcement learning has been employed to model controllers for a variety of character motions and skills [143]. For shape transforms, Berkiten et al. [1] present a metric learning approach for analogy-based mesh detail transfer. In P2P-NET, Yin et al. [131] develop a point displacement network which learns transforms between point-set shapes from two *paired* domains.

More closely related to our work is the VAE-CycleGAN recently developed by Gao et al. [130] for unpaired *shape deformation transfer*. Their network is trained on two unpaired animated mesh sequences, e.g., animations of a camel and a horse or animations of two humans with different builds. Then, given a deformation sample from one set, the network generates a shape belonging to the other set which possesses the same pose. One can view this problem as a special instance of the general shape transform problem. Specifically, it is a pose-preserving shape transform where the source meshes (respectively, the target meshes) model different poses of the same shape and they all have the same mesh connectivity. LOGAN, on the other hand, is designed to be a general-purpose translation network for point-set shapes, where much greater geometric and topological variations among the source or target shapes are allowed.

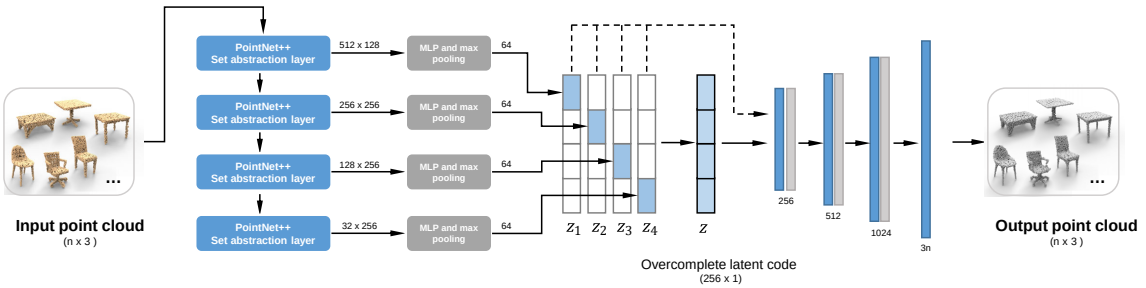


Figure 4.4: Architecture of our multi-scale, overcomplete autoencoder. We use the set abstraction layers of PointNet++ to produce point features in different scales and aggregate them into four sub-vectors: z_1, z_2, z_3, and z_4. The four sub-vectors are padded with zeros and summed up into a single 256-dimensional latent vector z that is overcomplete; the z vector can also be seen as a *concatenation* of the other four sub-vectors. During training, we feed all the five 256-dimensional vectors to the decoder. In the decoder, the blue bars represent fully-connected layers; grey bars represent ReLU layers.

Technically, the VAE-CycleGAN of Gao et al. [130] encodes each input set into a *separate* latent space and trains a CycleGAN to translate codes between the two latent spaces. 3D models can then be recovered from the latent codes by the VAE decoder. In contrast, LOGAN encodes shapes from both input domains into a common latent space and performs shape translations in that space. To enable generic shape transforms, the key challenge we address is learning what shape features to preserve during the translation.

Deep learning for point-set shapes Recently, several deep neural networks, including PointNET [49], PointNET++ [50], PCPNET [52], PointCNN [69], and PCNN [144], have been developed for feature learning over point clouds. Generative models of point-set shapes [102, 145, 146, 147] and supervised, general-purpose point-set transforms [131] have also been proposed. To the best of our knowledge, our work represents the first attempt at learning general shape transforms from unpaired domains. While LOGAN relies on PointNET++ for its multi-scale feature encodings, it produces an overcomplete latent code via feature concatenation rather than feature aggregation.

4.3 Method

Given two sets of unpaired shapes $\mathcal{X}$ and $\mathcal{Y}$, our goal is to establish two mappings $\mathcal{M}_{\mathcal{X}\to\mathcal{Y}} : \mathcal{X} \mapsto \mathcal{Y}$ and $\mathcal{M}_{\mathcal{Y}\to\mathcal{X}} : \mathcal{Y} \mapsto \mathcal{X}$, to translate shapes between the two domains. The translation should be natural and intuitive, with emphasis given to the preservation of common features. We make no special assumptions on the two domains, except that certain common features exist in both domains. Note that such features can be both local and global in nature, and hence are difficult to define or model directly. Therefore, we employ deep neural networks to implicitly learn those features.

4.3.1 Overview of networks and network loss

As shown in Figure 4.2, our network comprises of two parts that are trained in separate steps. First, an autoencoder is trained. The multi-scale encoder (Sec. 4.3.2) E takes point clouds from both domains as input, and encodes them into compact latent codes in a common latent space. The decoder D decodes the latent codes back into point clouds. After training, the autoencoder produces the over-complete latent codes for the input shapes, denoted by $\mathcal{Z}_{\mathcal{X}}$ and $\mathcal{Z}_{\mathcal{Y}}$, where $\mathcal{Z}_{\mathcal{X}} = \{E(X) \mid X \in \mathcal{X}\}$ and $\mathcal{Z}_{\mathcal{Y}} = \{E(Y) \mid Y \in \mathcal{Y}\}$.

The second part of our network is a latent code translator network that transforms between $\mathcal{Z}_{\mathcal{X}}$ and $\mathcal{Z}_{\mathcal{Y}}$. It consists of two translators: $T_{\mathcal{X}\to\mathcal{Y}} : \mathcal{Z}_{\mathcal{X}} \mapsto \mathcal{Z}_{\mathcal{Y}}$ and $T_{\mathcal{Y}\to\mathcal{X}} : \mathcal{Z}_{\mathcal{Y}} \mapsto \mathcal{Z}_{\mathcal{X}}$. We only show $T_{\mathcal{X}\to\mathcal{Y}}$ in the figure for simplicity. The translators take latent codes in both domains as input, and treat them differently with two different loss functions. Once trained, given a shape $X \in \mathcal{X}$, its translated shape in domain $\mathcal{Y}$ is obtained by $Y_x = D(T_{\mathcal{X}\to\mathcal{Y}}(E(X)))$.

We use three loss terms for the translator network to create a natural and feature-preserving mapping:

- *Adversarial loss*: We take $T_{\mathcal{X}\to\mathcal{Y}}$ in Figure 4.2(b) as an example. Given $x \in \mathcal{Z}_{\mathcal{X}}$, the network performs the translation and an adversarial loss is applied on the translated latent-code. The discriminator would judge the output codes from the ground-truth codes in $\mathcal{Z}_{\mathcal{Y}}$, to ensure that the distribution of the translator outputs matches the target distribution of $\mathcal{Z}_{\mathcal{Y}}$.

- *Feature preservation loss*: Given $y \in \mathcal{Z}_{\mathcal{Y}}$, since this network serves only $\mathcal{X} \to \mathcal{Y}$ transfer, the output still falls in $\mathcal{Z}_{\mathcal{Y}}$, and we use a feature preservation loss (identity loss) to enforce the output of the network to be similar to the original input y. Feature preservation loss is the key for our network to learn meaningful mappings, since it encourages the translator to keep most portions of the code intact and only changes the parts that are really important for the domain translation.

- *Cycle-consistency loss*: The two loss terms above play critical roles in our translator network and already allow the generation of satisfactory results. However, we may introduce a third loss term, the cycle-consistency loss or simply, the cycle loss, to further regularize the translation results.

  The cycle loss pushes each shape to reconstruct itself after being translated to the opposite domain and then translated back to its original domain. The term encourages the mappings $T_{\mathcal{X}\to\mathcal{Y}}$ and $T_{\mathcal{Y}\to\mathcal{X}}$ to be one-to-one. It further reduces the possibility that shapes in one domain only map to a handful of shapes in the opposite domain, i.e., mode collapse.

Note that our translator network still maintains a cross-domain similarity, which results from a joint effort of the feature preservation and adversarial losses — the former loss forces

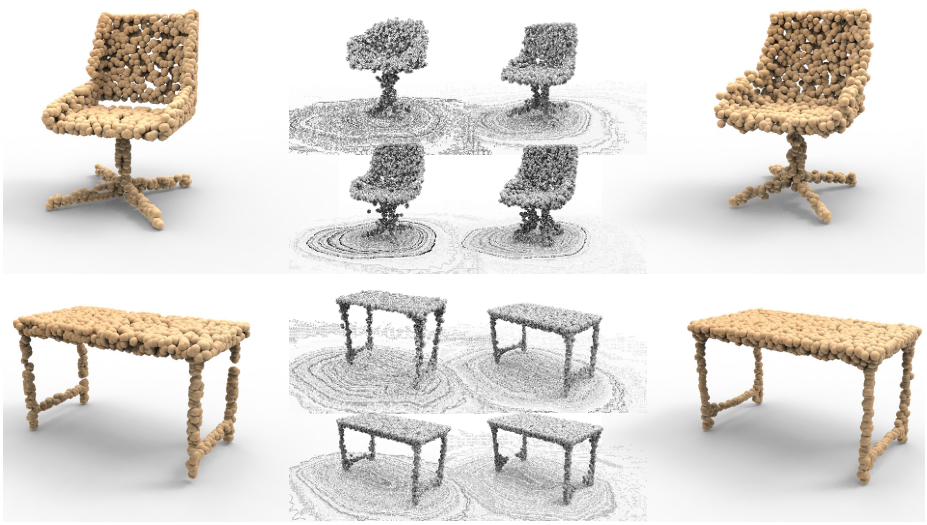


Figure 4.5: Our autoencoder encodes each test point cloud (a) into 5 latent vectors (z, $z_1, \ldots, z_4$), as shown in Figure 4.4, and decodes them back to point clouds. The decoder output (c) for the overcomplete latent code z exhibits better reconstruction than those from the other sub-vectors (b). Yet, all the reconstructions in (b) and (c) resemble the input shapes well, demonstrating the overcompleteness of our latent codes.

the result to be similar to the source shape, while the latter loss ensures plausibility in the target domain. The cycle loss only serves a supportive role in our network. Without cycle-consistency loss, our network is still able to suppress mode collapse. The multi-scale encoder pushes the codes from both domains to share the common space in different scales, bringing large distribution overlap and making it hard to collapse in $\mathcal{Z}_\mathcal{X} \to \mathcal{Z}_\mathcal{Y}$ without collapsing in $\mathcal{Z}_\mathcal{Y} \to \mathcal{Z}_\mathcal{Y}$, and the latter is unlikely to happen due to the feature preservation loss. In our experiments, cycle loss becomes prominent when the size of the dataset is very small. Details of the translator network and the loss functions can be found in Sec. 4.3.3.

4.3.2 Multi-scale overcomplete autoencoder

Our multi-scale autoencoder is depicted in Figure 4.4. The input to our encoder is a set of n points. It passes through four set abstraction layers of PointNet++ [50] with increasing sampling radius. The output point features from each of the four layers are further processed by an MLP and a max-pooling layer to form a single 64-dimensional sub-vector. We pad the 4 sub-vectors z_1, z_2, z_3, z_4 from the 4 branches with zeros to make them 256-dimensional vectors, and sum them up to get an overcomplete 256 dimensional latent vector z. The detailed network structure can be found in the supplementary material.

During training, we feed the padded sub-vectors and the overcomplete latent vector to the same decoder. Similar to [145], our decoder is a multilayer perceptron (MLP) with 4 fully-connected layers. Each of the first 3 fully-connected layers are followed by a ReLU layer

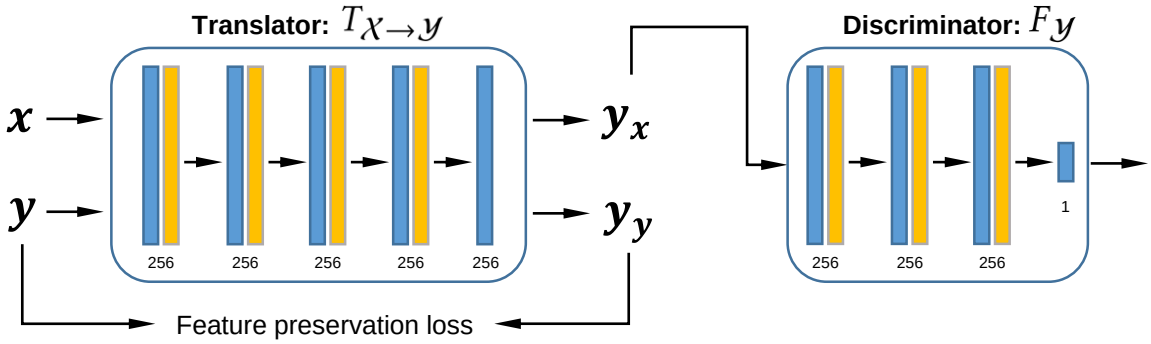


Figure 4.6: Architecture of our translator network. The blue bars represent fully-connected layers; orange bars represent BN-ReLU layers.

for non-linear activation. The last fully-connected layer outputs an $n \times 3$ point cloud for each input 256-dimensional vector. As shown in Figure 4.5, our decoder is able to reconstruct point clouds from the 5 vectors simultaneously. The quality of reconstruction from z is higher than the 4 sub-vectors as it contains most information. The loss function of the autoencoder considers all the 5 point clouds reconstructed from the 5 vectors:

$$L_{\text{AE}} = L_z^{\text{rec}} + \lambda_1 \sum_{i=1}^{4} L_{z_i}^{\text{rec}}, \tag{4.1}$$

where λ_1 is a scalar weight set to 0.1 by default. L_z^{rec} and $L_{z_i}^{\text{rec}}$ denote reconstruction losses for code z and z_i. When training the autoencoder, the input point cloud contains 2,048 points. Our decoder produces 2,048 points from each of the 5 latent vectors. Note that we choose the Earth Mover's Distance (EMD) [148] to define the reconstruction losses for all the 5 latent vectors, since EMD has been found to produce less noisy outputs compared to Chamfer Distance [102].

Note that our current choices of the latent code length (256) and number of scales (4) are both empirical. We tested autoencoding using shorter/longer codes as well as scale counts from two to six. Generally, short codes do not allow the multi-scale shape features to be well encoded and using too few scales compromises the translator's ability to disentangle and preserve the right features during cross-domain translation. On the other hand, employing latent codes longer than 256 or increasing the number of scales beyond 4 would only introduce extra redundancy in the latent space, which did not improve translation results in our experiments.

4.3.3 Feature-preserving shape transform

Our translators $T_{\mathcal{X}\to\mathcal{Y}}$ and $T_{\mathcal{Y}\to\mathcal{X}}$ work in the common latent space. Similar to the decoder, they are implemented as MLPs with 5 fully-connected (FC) layers, as shown in Figure 4.6. The detailed network structure can be found in the supplementary material.

The two discriminators $F_{\mathcal{X}}$ and $F_{\mathcal{Y}}$ work in the latent space as well. They are implemented as MLPs with 3 FC hidden layers where each of them are followed by a BN layer and a ReLU layer, as shown in Figure 4.6. We adopt WGAN [149] in our implementation. To that end, we directly take the result of the output layer without sigmoid activation.

In the common latent space, the translators and discriminators work directly over the over-complete latent code, $x \in \mathcal{Z}_{\mathcal{X}}$ and $y \in \mathcal{Z}_{\mathcal{Y}}$, as they already contain the information of the sub-vectors. For simplification, in the following part of this section, we will only explain the loss function in details for $T_{\mathcal{X}\to\mathcal{Y}} : \mathcal{Z}_{\mathcal{X}} \mapsto \mathcal{Z}_{\mathcal{Y}}$. The opposite direction can be derived directly by swapping x and y in the equations. The loss function for $T_{\mathcal{X}\to\mathcal{Y}}$ is,

$$L_{\mathcal{X}\to\mathcal{Y}} = L^{\text{WGAN}}_{\mathcal{X}\to\mathcal{Y}} + \alpha L^{\text{FP}}_{\mathcal{X}\to\mathcal{Y}} \tag{4.2}$$

where α is a scalar weight set to 20 by default. Similar to WGAN [150], the adversarial loss for $T_{\mathcal{X}\to\mathcal{Y}}$ is defined as:

$$L^{\text{WGAN}}_{\mathcal{X}\to\mathcal{Y}} = \mathbb{E}_{y\sim\mathbb{P}(\mathcal{Z}_{\mathcal{Y}})}[F_{\mathcal{Y}}(y)] - \mathbb{E}_{x\sim\mathbb{P}(\mathcal{Z}_{\mathcal{X}})}[F_{\mathcal{Y}}(y_x)] + \lambda_2 L_{GP} \tag{4.3}$$

where $y_x = T_{\mathcal{X}\to\mathcal{Y}}(x)$ is the output of translation for x. L_{GP} is the gradient penalty term introduced by [150] for regularization. λ_2 is a scalar weight set to 10 by default. During training, our discriminator $F_{\mathcal{Y}}$ aims to maximize the adversarial loss, while our translator $T_{\mathcal{X}\to\mathcal{Y}}$ aims to minimize it.

As shown in Figure 4.6, our feature preservation loss is defined in the latent space. For translator $T_{\mathcal{X}\to\mathcal{Y}}$, it is defined as the $L1$ distance between the input vector $y \in \mathcal{Z}_{\mathcal{Y}}$ and its translated output vector $y_y = T_{\mathcal{X}\to\mathcal{Y}}(y)$:

$$L^{\text{FP}}_{\mathcal{X}\to\mathcal{Y}} = \mathbb{E}_{y\sim\mathbb{P}(\mathcal{Z}_{\mathcal{Y}})}[\|y - T_{\mathcal{X}\to\mathcal{Y}}(y)\|_1] \tag{4.4}$$

Training our translators with the loss function $L_{\mathcal{X}\to\mathcal{Y}}$ or $L_{\mathcal{Y}\to\mathcal{X}}$ is able to produce reasonable result as shown in Figure 4.8 (g). However, having a cycle-consistency term could further improve the result by encouraging one-to-one mapping between the two input domains. For latent code $x \in \mathcal{Z}_{\mathcal{X}}$, applying $T_{\mathcal{X}\to\mathcal{Y}}$ followed by $T_{\mathcal{Y}\to\mathcal{X}}$ should produce a code similar to itself: $T_{\mathcal{Y}\to\mathcal{X}}(T_{\mathcal{X}\to\mathcal{Y}}(x)) \approx x$. Similarly for $y \in \mathcal{Z}_{\mathcal{Y}}$ we have: $T_{\mathcal{X}\to\mathcal{Y}}(T_{\mathcal{Y}\to\mathcal{X}}(y)) \approx y$. Thus, the cycle-consistency loss term is defined as,

$$\begin{aligned} L_{\text{Cycle}} = {} & \mathbb{E}_{x\sim\mathbb{P}(\mathcal{Z}_{\mathcal{X}})}[\|T_{\mathcal{Y}\to\mathcal{X}}(T_{\mathcal{X}\to\mathcal{Y}}(x)) - x\|_1] \\ & + \mathbb{E}_{y\sim\mathbb{P}(\mathcal{Z}_{\mathcal{Y}})}[\|T_{\mathcal{X}\to\mathcal{Y}}(T_{\mathcal{Y}\to\mathcal{X}}(y)) - y\|_1] \end{aligned} \tag{4.5}$$

The overall loss function is defined as:

$$L_{\text{Overall}} = L_{\mathcal{X}\to\mathcal{Y}} + L_{\mathcal{Y}\to\mathcal{X}} + \beta L_{\text{Cycle}} \tag{4.6}$$

where β is a scalar weight set to 20 by default. With the overall loss function, the goal of training the translators is to solve:

$$T^*_{\mathcal{X}\to\mathcal{Y}}, T^*_{\mathcal{Y}\to\mathcal{X}} = \arg\min_{T}\max_{F} L_{\text{Overall}} \tag{4.7}$$

where F denotes $\{F_{\mathcal{X}}, F_{\mathcal{Y}}\}$, T denotes $\{T_{\mathcal{X}\to\mathcal{Y}}, T_{\mathcal{Y}\to\mathcal{X}}\}$. In Sec. 3.4, we perform an ablation study to show that all loss terms play active roles in achieving high-quality results.

4.3.4 Training details and point upsampling

We train the AE and the translator networks separately in two steps, since an insufficiently trained autoencoder can misguide the translators into poor local minima. In our experiments, we train the autoencoder for 400 epochs with an Adam optimizer (learning rate = 0.0005, batch size = 32). After that, we train the translator networks with Adam and the training schema of [150] for 600 epochs. We set the number of discriminator iterations per generator iteration to two. The batch size we set for training the translator networks is 128. The learning rate starts at 0.002 and is halved after every 100 training epochs, until reaching 5e-4. Assuming each of the datasets of two domains contains 5000 shapes, the training of autoencoder takes about 20 hours and the training of two translators takes about 10 mins on a NVIDIA Titan Xp GPU.

By default, we train LOGAN on point clouds each of which contains 2,048 points and the networks output point clouds at the same resolution. However, we allow the addition of an *upsampling layer* to our network to produce higher-resolution outputs when desired. Specifically, we attach a single upsampling layer to the *second-to-last* layer of the trained decoder. We train the upsampling layer independently after the shape translation has been complete, so it would not affect the translation results. As shown in Figure 4.7, the upsampling layer recombines the neural outputs from the second-to-last layer and predicts m local displacement vectors for each point in the output point cloud. As a result, it splits each point into m points and turns the sparse point cloud of size n into a dense point cloud of size mn. The upsampling layer is a fully-connected layer which is followed by a sigmoid function. We scale the output of the sigmoid function to make it lie in the interval $[-0.05, 0.05]$.

To train the upsampling layer, we sample 16,384 points from each training shape as the ground truth (GT). The loss function is defined as the distance between the dense point cloud of size mn and the dense GT point cloud. To reduce computation and memory costs, we randomly select 4,096 points from each of the two dense point clouds, and compute the EMD between the downsampled point clouds as an approximation of the EMD between the dense point clouds. Note that sometimes one of the two domains does not require upsampling (e.g., meso-skeletons). In that case, we train the upsampling layer with only the dataset of the other domain. The upsampling layer is trained independently for 80 epochs, with an Adam optimizer (learning rate = 0.0005, batch size = 32). Assuming that each of the two

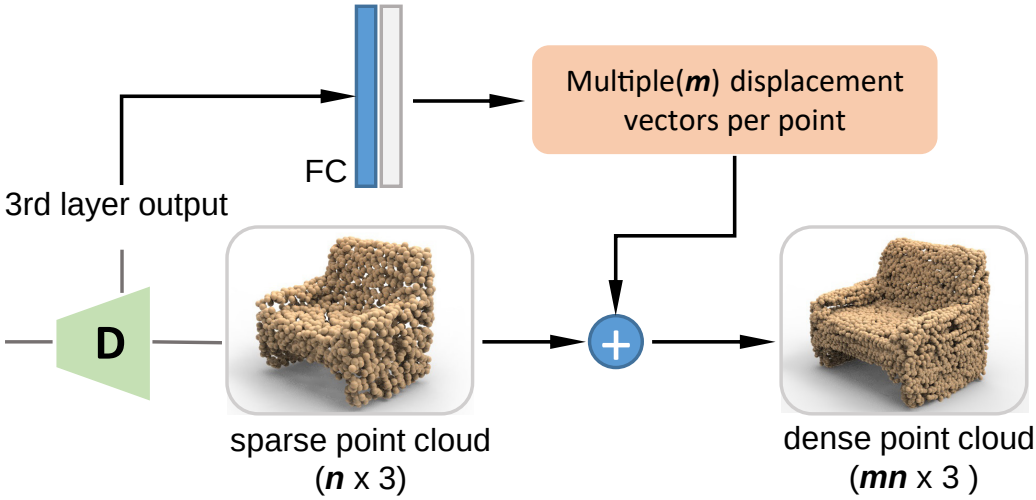


Figure 4.7: Architecture of the upsampling layer of our network after shape translation. We predict m local displacement vectors for each of the n points in the sparse point cloud, which results in a dense set of mn points.

domains contains 5,000 shapes, training the upsampler takes about 5 hours on a NVIDIA Titan Xp GPU.

4.4 Results and evaluation

To demonstrate the capability of LOGAN in learning unpaired shape transforms, we conduct experiments including ablation studies and comparisons to baselines and state-of-the-art techniques. Throughout the experiments, specifically for all results shown in Sections 4.4.1, 4.4.2, 4.4.3 and 4.4.4, LOGAN was trained with the *same* default network settings as described in Section 3.3 and the appendix; there is no hyperparameter or architecture tuning for any specific input datasets. The only tuning was applied when training LOGAN on the small datasets of [130], to avoid overfitting. All visual results are presented without any post-processing.

4.4.1 Shape transform results and ablation studies

The first domain pair we tested LOGAN on is the chair and table datasets from ShapeNet Core [151], which contains mesh models. The chair dataset consists of 4,768 training shapes and 2,010 test shapes, while the table dataset has 5,933 training and 2,526 test shapes. We normalize each chair/table mesh to make the diagonal of its bounding box equal to unit length and sample the normalized mesh uniformly at random to obtain 2,048 points for our point-set shape representation. If upsampling is required, we sample from each training mesh another set of 16,384 points with Poisson disk sampling [123] to form the training

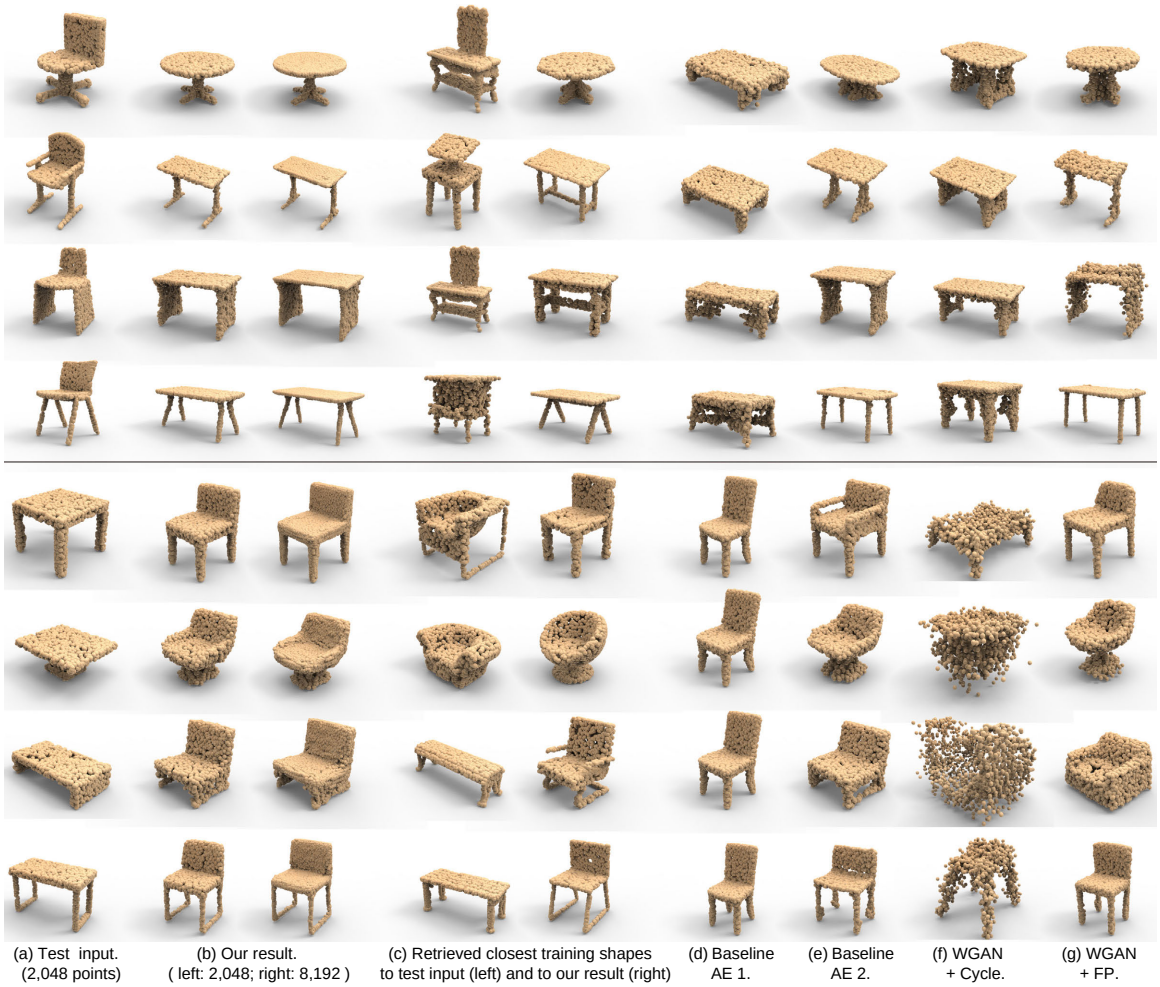


Figure 4.8: Comparing chair-table translation results using different network configurations. Top four rows: chair → table. Rest: table → chair. (a) Test input. (b) LOGAN results with and without upsampling. (c) Retrieved training shapes from the *target* domain which are closest to the test input (left) and to our translator output (right). The retrieval was based on EMD between point clouds at 2,048 point resolution. Note that the chair dataset from ShapeNet has some benches mixed in, which are retrieved as "tables." (d) Baseline AE 1 as autoencoder + our translator network. (e) Baseline AE 2 ($\lambda_1 = 0$) + our translator network. (f) Our autoencoder ($\lambda_1 = 0.1$) + WGAN & Cycle loss. (g) Our autoencoder ($\lambda_1 = 0.1$) + WGAN & feature preservation (FP) loss.

dataset for the upsampler. But note that the autoencoder and the translator networks are always trained by point clouds of size 2,048.

Comparing autoencoding. With the chair-table domain pair, we first compare our autoencoder, which produces multi-scale and overcomplete latent codes, with two baseline alternatives:

- In Baseline AE 1, we apply the original PointNet++, as described in [50], as the encoder to produce latent vectors of length 256 (the same as in our autoencoder) and use the same decoder as described in Section 4.3.2. With this alternative, there is no separate encoding of mulit-scale features (into sub-vectors as in our case) to produce an overcomplete latent code; features from all scales are aggregated.

- In Baseline AE 2, we set $\lambda_1 = 0$ in the loss function (4.1) of our autoencoder. With this alternative, the autoencoder still accounts for shape features from all scales (via the vector z), but the impact of each sub-vector (one of the z_i's, $i = 1, \ldots, 4$) for a specific feature scale is diminished.

An examination on reconstruction errors of the three autoencoders, based on the Earth Mover's Distance (EMD), reveals that our autoencoder may not be the best at reconstructing shapes. However, the main design goal of our autoencoder is to facilitate shape translation between unpaired domains, not accurate self-reconstruction.

In Figure 4.8 (b, d, e), we show that with the same translator network but operating in different latent spaces, our autoencoder leads to the best cross-domain transforms, compared to the two baselines.

With Baseline AE 1, the translator is unable to preserve input features and can suffer from mode collapse. With a multi-scale overcomplete code z, Baseline AE 2 clearly improves results, but it can still miss input features that should be preserved, e.g., more global features such as the roundness at the top (row 1), the oblique angles at the legs (row 4), and more local features such as the bottom slat between the legs (row 8); it could also add erroneous features such as armrests (row 5) and extra holes (row 7).

In contrast, with a more overcomplete and multi-scale encoding into the latent space by using all five vectors $(z, z_1, \ldots, z_4)$, our default autoencoder produces the most natural table-chair translations. This is likely attributed to a better disentangling of the preserved and altered features in the latent codes.

Note that the results shown in Figure 4.8 were casually picked as representative examples to demonstrate the capability of our network. Larger sets of randomly selected outputs can be found in the supplementary material and they reveal the same trend: our method consistently outperforms the baselines.

Comparison to retrieval results. As a sanity check, we show that our network indeed *generates* shape transforms; it does not simply retrieve a training shape from the target domain. In Figure 4.8, column (c), we show retrieved training shapes from the target domain that are the closest (based on EMD) to the test input and to our translator output. It is quite evident that in general, these training shapes are far from being similar to outputs from our translator, with a couple of exceptions shown in rows 4 and 5. In these rare cases, it so happens that there are similar shapes in the target training set.

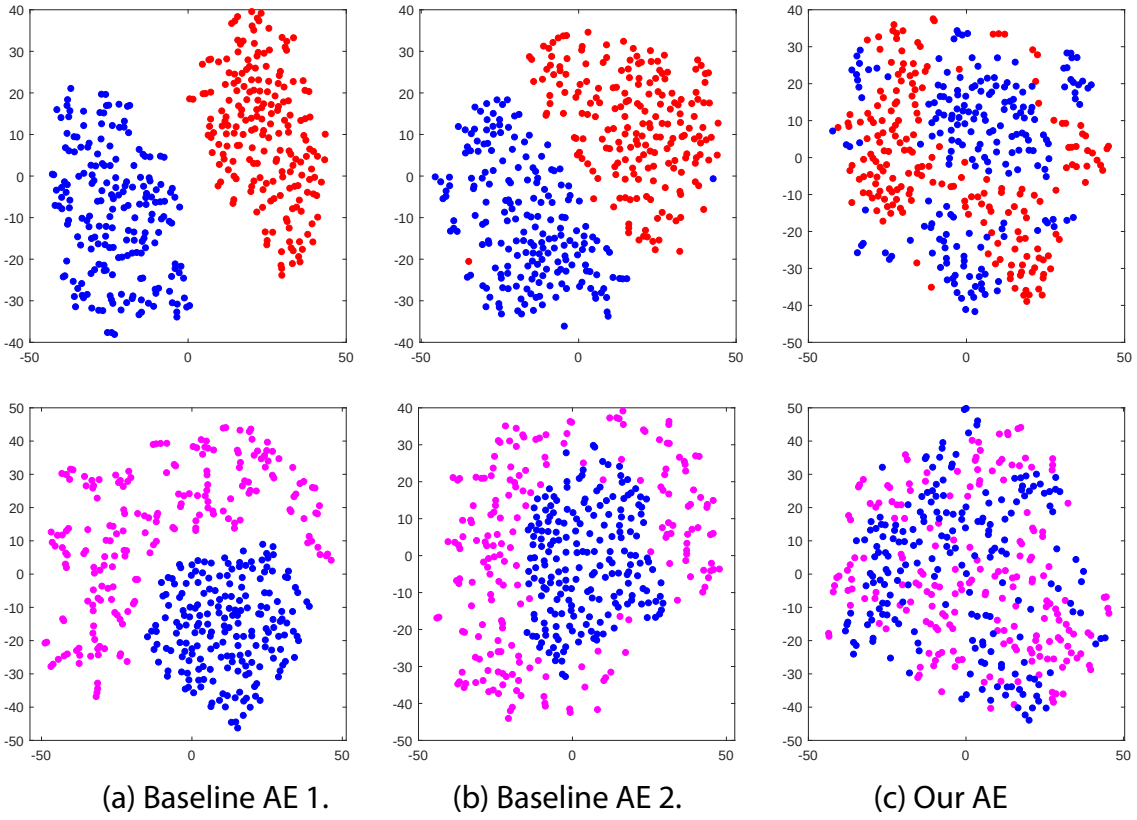


Figure 4.9: Visualizing joint embeddings of table and chair latent codes produced by three AEs. Top row: red = chair, blue = table, *before* translation. Bottom: magenta = chair *after* translation; blue = original table. Our default AE brings the chairs and tables closer together in common latent space.

Joint embedding of latent codes. Figure 4.9 visualizes the common latent spaces constructed by the three AEs by jointly embedding the chair and table latent codes. For each domain, we standardize every latent dimension by moving the mean to 0 and scaling the values to a standard deviation of 1.0. We discretize the values in each dimension by multiplying it by 3 and rounding it to an integer. Finally, we measure distances between all the latent codes using Hamming distance and embed the codes into 2D space via t-SNE.

We can observe that, compared to the two baselines, our default AE brings the chairs and tables closer together in the latent space, before the translation, effectively "tangling" together the distributions of the generated latent codes to better discover their common features. During translation, the overcomplete codes facilitate an implicit disentanglement of the preserved and altered features. After chair→table translation, the chair codes are closer to the tables in all three cases, but our default network clearly produces a better coverage of the target (table) domain. This makes the translated chair latent codes more plausible in the table domain, which can explain, in part, a superior performance for the translation task.

Figure 4.10: Unpaired shape transforms between armchairs and armless chairs. The first two rows show results of armrest removal by LOGAN, while the last two rows show insertion. On the right, we show the mesh editing results guided by the learned point cloud transforms.

Comparing translator settings. In the second ablation study, we fix our autoencoder as presented in Figure 4.4, but change the translator network configuration by altering the loss function into two baseline alternatives: WGAN loss + Cycle loss and WGAN loss + feature preservation (FP) loss. Note that our default network LOGAN has all three losses. It is quite evident, from the visual results in Figure 4.8, that the feature preservation loss has significant positive impact on cross-domain translation, while the cycle loss provides additional regularization for improved results.

Part removal/insertion. Chair-table translations mainly involve transforms between local shape structures. As another example, we show that LOGAN is able to learn to remove/add armrests for the chair dataset; see Figure 4.10. The dataset split was obtained from the chairs in ShapeNet Core by a hand-crafted classifier. It contains 2,138 armchairs and 3,572 armless chairs, where we used 80% of the data for training and the rest for testing. The results demonstrate that our network can work effectively on part-level manipulation, as it learns which parts to alter and which parts to preserve purely based on the observation of the input shapes from the two domains, without supervised training. In addition, the insertion/removal of the armrest parts are carried out naturally.

We show in Figure 4.10 (right) that we can use the learned point cloud transforms to guide mesh editing. We remove/add mesh parts according to the difference between the

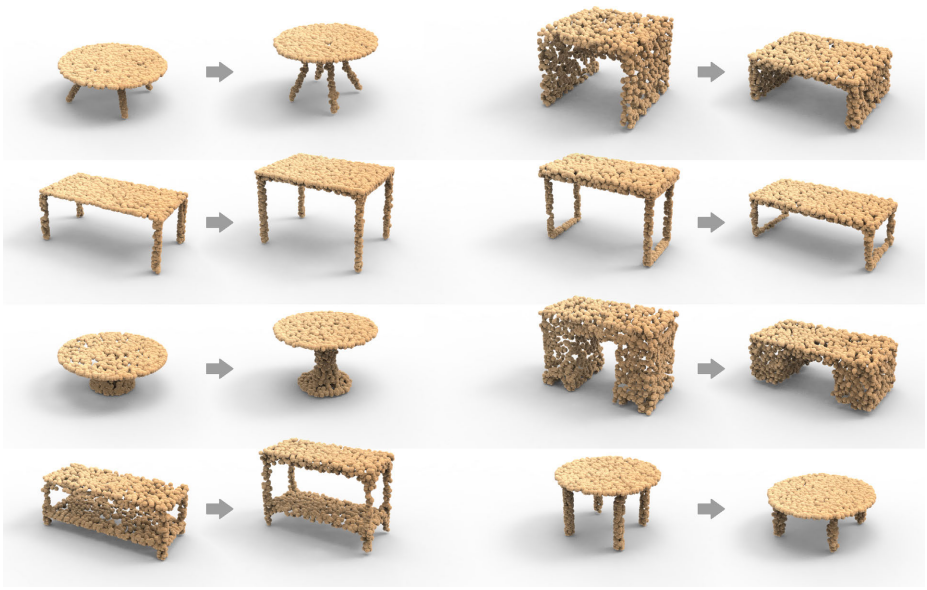


Figure 4.11: Unpaired shape transforms between tall and short tables. Left: increasing height. Right: decreasing height.

original and transformed point clouds. The mesh parts are retrieved from the ShapeNet part dataset [152] by Chamfer distance. Details of retrieval and more examples can be found in the supplementary material.

Transforming global 3D shape attribute. At last, we show that LOGAN is able to learn to increase/decrease heights of tables, which can be considered as a style for 3D shapes, as shown in Figure 4.11. To obtain such a suitable dataset, we sort the height-width ratios of all the tables from ShapeNet Core dataset. The first 3,000 tables are selected as tall tables, while the last 3,000 as short tables. From each of the two sets, we randomly choose 500 shapes as the test set, and the rest are used for training. The results demonstrate that LOGAN is able to alter global attributes of 3D shapes.

4.4.2 Unpaired style/content transfer and comparisons

Most deep networks for unpaired cross-domain translation operate on images, aiming for content-preserving style transfer [5, 6, 80, 96]. We conduct an experiment to compare LOGAN with these state-of-the-art networks on a *Font* dataset we collected. The dataset consists of 7,466 fonts of English letters. For each letter, we produce a rendered 256^2 image by normalizing the letter to make the longest edge of its bounding box equal to 248. Then we obtain a point cloud by uniformly sampling 2,048 points over the pixels inside each letter shape.

In the first test, *style transfer*, we train the networks to translate between different styles of various fonts, while keeping the source and target letters the same. In Figures 4.12, we

Figure 4.12: Comparisons on content-preserving style transfer, i.e., *regularA/H-italicA/H*, *thinG/R-thickG/R*, and *wideM/N-narrowM/N* translations, by different methods. First two rows: regular-to-italic; middle two rows: thin-to-thick; last two rows: wide-to-narrow. From left to right: input letter images; corresponding input point clouds; output point clouds from LOGAN; images reconstructed from our results; output images of CycleGAN; outputs from UNIT [80]; outputs from MUNIT [96]. For *wideM/N-narrowM/N* we align the letters by height for better visualization.

show examples including *regularA/H-italicA/H*, *thinG/R-thickG/R*, and *wideM/N-narrowM/N*. Since most fonts in our collected dataset do not have paired regular and italic types, nor paired regular and boldface types, we split the dataset ourselves. To split the dataset for *regularA/H-italicA/H*, we simply sort the fonts by checking how vertical the letter "I" is for each font. The first 2,500 fonts with more vertical "I"s are regarded as *regularA/H*s, while the last 2,500 as *italicA/H*s. Similarly, we split the set of all *G*s and *R*s in the dataset simply by sorting them based on how many pixels are inside the letter shapes. The first 2,500 letters with more interior pixels are regarded as *thickG/R*s, and the last 2,500 as *thinG/R*s. To obtain the dataset for *wideM/N-narrowM/N*, we sort *M*s and *N*s by looking at their width-height ratios. Finally, we randomly selected 500 fonts from each of the above sets to serve as test set for the specific tasks, while using the rest for training. An ideal translator

Figure 4.13: Comparisons on style-preserving content transfer, i.e., *A-H*, *G-R*, and *M-N* translations, by different methods, including ground truth.

Table 4.1: Quantitative comparisons on *A-H*, *G-R*, and *M-N* translations by different unpaired cross-domain translation networks. Mean squared error (MSE) and intersection over union (IOU) are measured against ground-truth target letters and averaged over the testing split of the respective datasets. Better-performing numbers are highlighted in boldface.

	A↔H		G↔R		M↔N	
	MSE	IOU	MSE	IOU	MSE	IOU
CycleGAN	0.246	0.385	0.229	0.412	0.266	0.383
UNIT	0.253	0.376	0.264	0.377	0.295	0.348
MUNIT	0.280	0.286	0.358	0.171	0.363	0.292
Ours	**0.195**	**0.490**	**0.213**	**0.472**	**0.207**	**0.506**

would only change the specific style of an input letter while keeping the letter in the same font.

The second comparison is on style-preserving *content transfer*, via letter translation tasks on three subsets: *A-H*, *G-R*, and *M-N*. To obtain the *G-R* dataset, from the 7,466 pairs of uppercase *G*s and *R*s, each in the same font, we randomly selected 1,000 to serve as

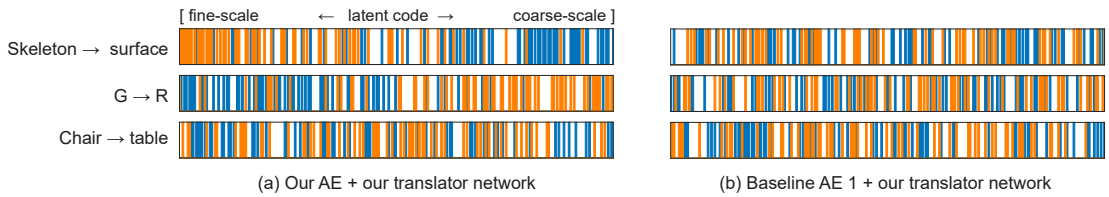


Figure 4.14: Visualizing "disentanglement" in latent code preservation (blue color) and alteration (orange color) during translation, where each bar represents one dimension of the latent code. Orange bars: top 64 latent code dimensions with the largest average changes. Blue bars: dimensions with smallest code changes.

the testing set while using the rest for training. The datasets for *A-H* and *M-N* are obtained in the same way. For these task, we expect an ideal translator to transform samples between the two domains by changing the letter (content) only, while preserving its style, e.g., font width, height, thickness, skewness, etc.

We compare LOGAN with three unpaired image translation networks: the original CycleGAN [5] which translates images directly, as well as UNIT [80] and MUNIT [96], both of which utilize shared latent spaces. While LOGAN operates on point clouds, the other networks all input and output images. We trained each of the four networks for 15 hours for each of the *regularA/H-italicA/H*, *thinG/R-thickG/R*, and *wideM/N-narrowM/N* translations; and 27 hours for each of the *A-H*, *G-R*, and *M-N* translations. To help with comparison, we convert the output point clouds from LOGAN to images: for each point in a given point cloud, we find all its neighbors within r pixels away, and then fill the convex hull of these points; we used $r = 10$ in our tests.

Results for content-preserving style translation are shown in Figures 4.12, while Figure 4.13 compares results for style-preserving content transfer. More results can be found in appendix and the supplementary material. We observe that CycleGAN, UNIT, and MUNIT are unable to learn transforms between global shape structures, which are necessary for letter translations and certain style translations.

Overall, our network can adaptively learn which features (content vs. style) to preserve and which to transfer according to the training domain pairs. We also provide quantitative comparisons in Table 4.1 for letter translations since we have ground-truth target letters. These results again demonstrate the superiority of LOGAN.

4.4.3 Implicit disentanglement over latent codes

We examine how our latent space representations may help disentangle preserved vs. altered shape features during cross-domain translation. In Figure 4.14, we plot latent code dimensions with the largest (top 64 out of 256 dimensions, in orange color) and smallest changes (bottom 64, in blue color) for three translation tasks: airplane skeleton→ surface

Table 4.2: Quantitative comparisons between different autoencoder and translator configurations, on transformation tasks from P2P-NET. Reported errors are averaged over two categories per domain pairs (see Figure 4.15), and are measured against ground-truth target shapes from the P2P-NET dataset. Since P2P-NET did not come with an upsampling layer, for a fair comparison, all point cloud results are obtained at the same resolution of 2,048 points.

	Skeleton ↔ Shape		Scan ↔ Shape		Profiles ↔ Surface	
	Chamfer	EMD/n	Chamfer	EMD/n	Chamfer	EMD/n
PointNET++ autoencoder (AE) + Our translator with all three losses	5.30	0.071	5.30	0.079	9.29	0.099
Our AE ($\lambda_1 = 0$) + Our translator with all three losses	2.24	0.048	2.42	0.051	**3.08**	**0.061**
Our AE ($\lambda_1 = 0.1$) + Our translator with only WGAN + Cycle losses	14.06	0.098	16.74	0.127	17.06	0.116
Our AE ($\lambda_1 = 0.1$) + Our translator with only WGAN + FP losses	2.22	0.047	2.53	0.054	3.37	0.064
LOGAN: Our AE ($\lambda_1 = 0.1$) + Our translator with all three losses	**2.11**	**0.046**	**2.18**	**0.048**	3.09	**0.061**
P2P-NET: Supervised method with paired domains	0.44	0.020	0.66	0.060	1.36	0.056

(Section 4.4.4), $G \rightarrow R$, and chair→table. Note that the plots reflect the *mean* magnitude of code changes in each dimension, over the respective *whole test sets*.

We can observe that our network automatically learns the right features to preserve (e.g., more global or coarser-scale features for airplane skeleton→ surface and more local features for $G \rightarrow R$), solely based on the input domain pairs. For the chair→table translation, coarse-level and fine-level features are both impacted. Compared to PointNET++ encoding (baseline AE 1), our default autoencoder with overcomplete codes better disentangles parts of the latent codes that are preserved vs. altered— this is more pronounced for the first two examples and less so for the chair-table translation.

4.4.4 Comparison with supervised P2P-NET

In Figure 4.15, we show unpaired cross-domain shape transform results obtained by LOGAN, on several domain pairs from the recent work P2P-NET [131], where the specific input shapes are also from their work. We compare these results to P2P-NET, as well as results from other network configurations as done in Figure 4.8. Note that these shape transforms, e.g., cross-sectional profiles to shape surfaces, are of a completely different nature compared to table-chair translations. Yet, our network is able to produce satisfactory results, as shown in column (b), which are visually comparable to results obtained by P2P-NET, a supervised method.

Both LOGAN and P2P-NET aim to learn general-purpose cross-domain transforms between point-set shapes. P2P-NET works on paired domains but without explicit feature preservation, while LOGAN is trained on unpaired data but enforces a feature preservation loss in the GAN translator. The results show that LOGAN is able to preserve the right global features for skeleton/scan-to-shape translations. At the same time, some finer details, e.g., the swivel chair legs and the small bump near the back of the fuselage in row 1, can also

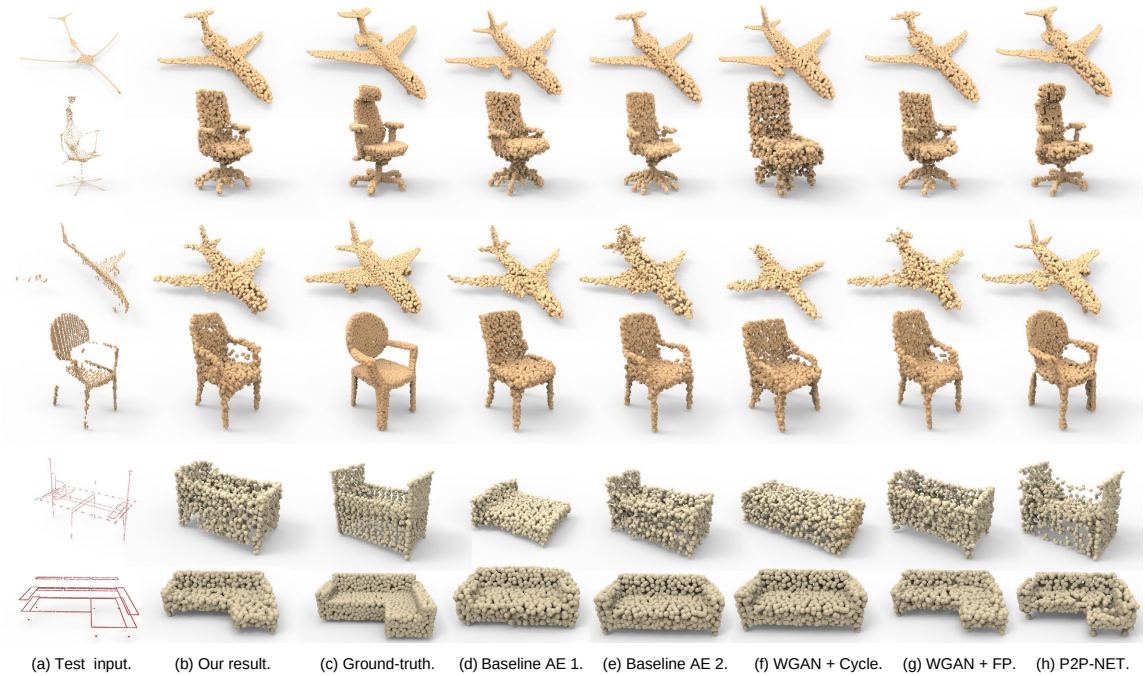


Figure 4.15: Comparisons between various network configurations, (supervised) P2P-NET, and ground truth targets, on shape transform examples from P2P-NET: skeleton→shape (rows 1-2), scan→surface (rows 3-4), and (cross-sectional) profiles→surface (rows 5-6). All point clouds have 2,048 points.

be recovered. However, the unsupervised LOGAN cannot quite match P2P-NET in this regard; see the back of the swivel chair.

Since P2P-NET is supervised, ground-truth target shapes are available to allow us to quantitatively measure the approximation quality of the translation results. As shown in Table 4.2, our default LOGAN network achieves the best quality, compared to other baseline alternatives, but still falls short of the supervised P2P-NET.

4.4.5 Comparison with unpaired deformation transfer

The latent VAE-CycleGAN developed by Gao et al. [130] was designed for the specific task of unpaired deformation transfer on meshes. In this last experiment, we further test the generality of our shape transformation network, by training it on datasets from [130]. In Figure 4.16, we compare results obtained by LOGAN and results from [130] as provided by the authors. The point clouds for training were obtained by uniformly sampling 2,048 points from the corresponding meshes. Note that the *horse* → *camel* dataset contains a total of 384 shapes in the training set; *fit* → *fat* contains 583 shapes. Since these datasets are significantly smaller than those from the previous experiments, we adjusted the hyperparameter λ_2 to 40 in order to better avoid overfitting, and increase the number of training epochs for translators to 1,200.

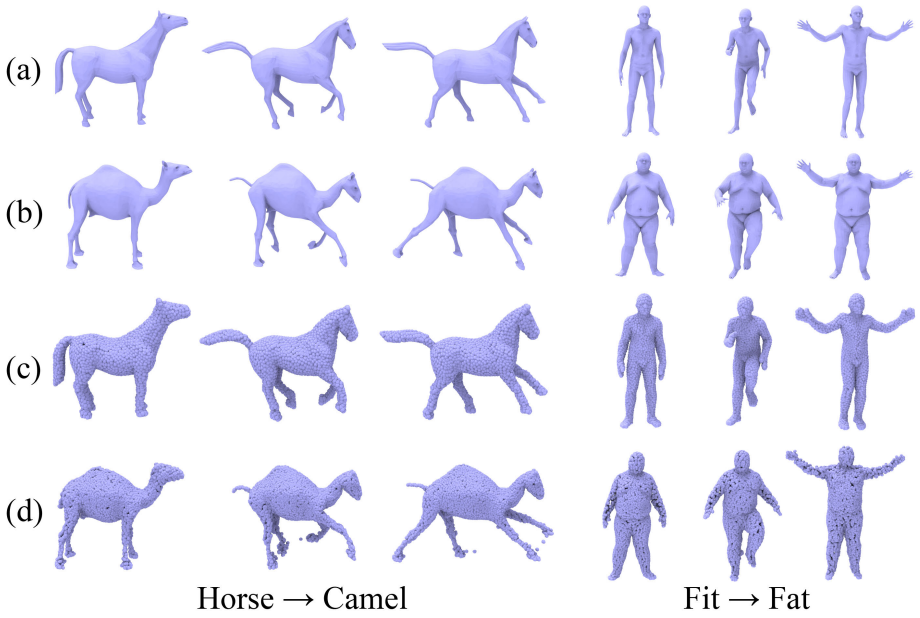


Figure 4.16: Comparison with unpaired deformation transfer [130], demonstrating that LOGAN can also accomplish the task. (a) Input meshes; (b) output meshes by Gao et al. [130]; (c) input point clouds of size 2,048 sampled from (a); (d) output point clouds upsampled to 8,192 points.

The results show that, *qualitatively*, LOGAN, which is designed as a general-purpose cross-domain shape transform network, is also able to learn to preserve pose-related features and achieve pose-preserving shape transform, like Gao et al. [130]. However, since our current implementation of the network is limited in training resolution (at 2,048 points), the visual quality of the generated point clouds does not quite match that of their mesh outputs.

4.5 Discussion and limitation

Shape transform, or shape-to-shape translation, is a basic problem in geometric modeling. We believe that it is as fundamental to computer graphics as image-to-image translation is to computer vision. We develop a deep neural network for learning generic cross-domain shape translations. The key challenge posed is how to ensure that the network can learn shape transforms between two domains *without* any paired shapes. Our motivation is that for most modeling tasks, especially those involving 3D models, it is difficult to find pre-existing paired data due to a lack of 3D models to begin with. On the other hand, synthesizing paired 3D data is unrealistic since 3D modeling is generally a non-trivial task; this is the very reason why we seek learning methods to automate the process.

We regard our work as only making a first step towards generic, unpaired cross-domain shape transform, and it still has quite a number of limitations. First, due to the inherent

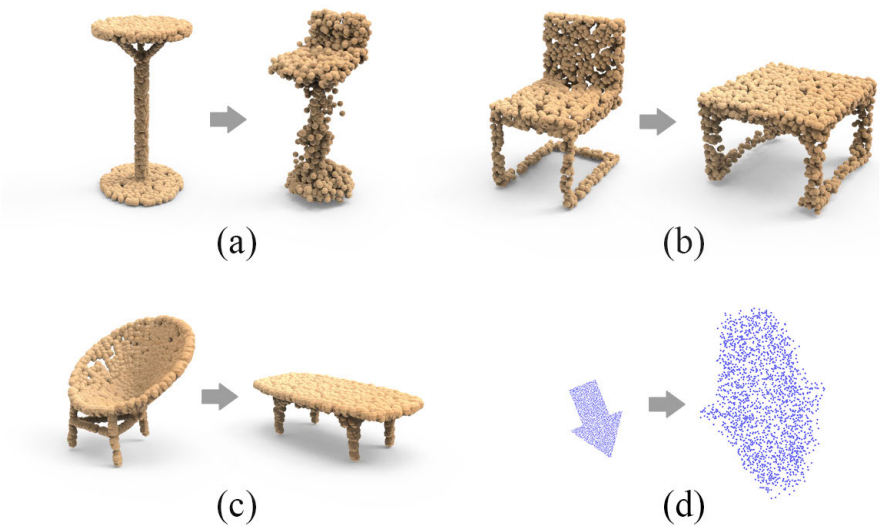


Figure 4.17: Several failure cases by LOGAN: (a) scattered point clouds; (b-c) unnatural transform results due to lack of commonalities in the target domain; (d) failed translation between shapes differing in global scales.

nature of point cloud representations, the output shapes from our network are not necessarily clean and compact. The points can be scattered around the desired locations, especially when there are thin parts; see Figure 4.17(a) and some results in Figure 4.13.

Second, due to our assumption of shared commonalities between the input domains, if an input shape in one domain cannot find sufficient commonalities in the other domain, our network cannot learn a natural translation for it; see Figures 4.17(b-c). In such cases, the adversarial loss plays a more dominant role. We can observe the impact of this loss in rows 1-3 of Figure 4.8. These chair→table translation results by LOGAN cannot retain the squared tops, which may be judged by some as an unnatural transform; the reason is that most tables in the training set have rectangular tops. Similarly, the result in Figure 4.17(b) is not a complete failure as the output table did preserve the square top as well as certain leg features. Simply removing the chair back would result in an unusual table.

Third, performing translations in a common space and measuring the feature preservation loss entry-by-entry imply that we implicitly assume a "scale-wise alignment" between the input shapes. That is, the common features to be preserved should be in the same scales. Figure 4.17 (d) shows a result from LOGAN which was trained to translate between arrow shapes of very different scales; the result is unnatural due to a lack of that scale-wise alignment. Last but not least, as a consequence of employing latent space transforms, our method is unable to output shape correspondences between the source and target shapes as prior works could.

In future work, we would like to consider other overcomplete, concatenated shape encodings where the different representations reflect other, possibly semantic, aspects of the shapes, beyond their multi-scale features. We would also like to expand and broaden the

scope of shape transforms to operations such as shape completion, style/content analogy, and more. Finally, semi-supervision or conditional translations [96] to gain more control on the transform tasks are also worth investigating.

4.6 Appendix

4.6.1 Details of network architecture

We provide the detailed architectures of the autoencoder, discriminator, and translator networks in LOGAN in Figure 4.18. Let us denote a set abstraction layer of PointNet++ as $SA(M, r, N, [l_1, ..., l_d])$ where M is the number of local patches, r is the radius of balls that bound the patches, N is the number of sample points selected in each patch, $[l_1, ..., l_d]$ are the widths of fully-connected layers used in local PointNet. A fully connected layer is denoted as $FC(l)$ where l is its width. A dropout layer is denoted as $DROPOUT(p)$ where p is the probability that each element is dropped. We use $RELU$, BN and $MAXPOOL$ to represent

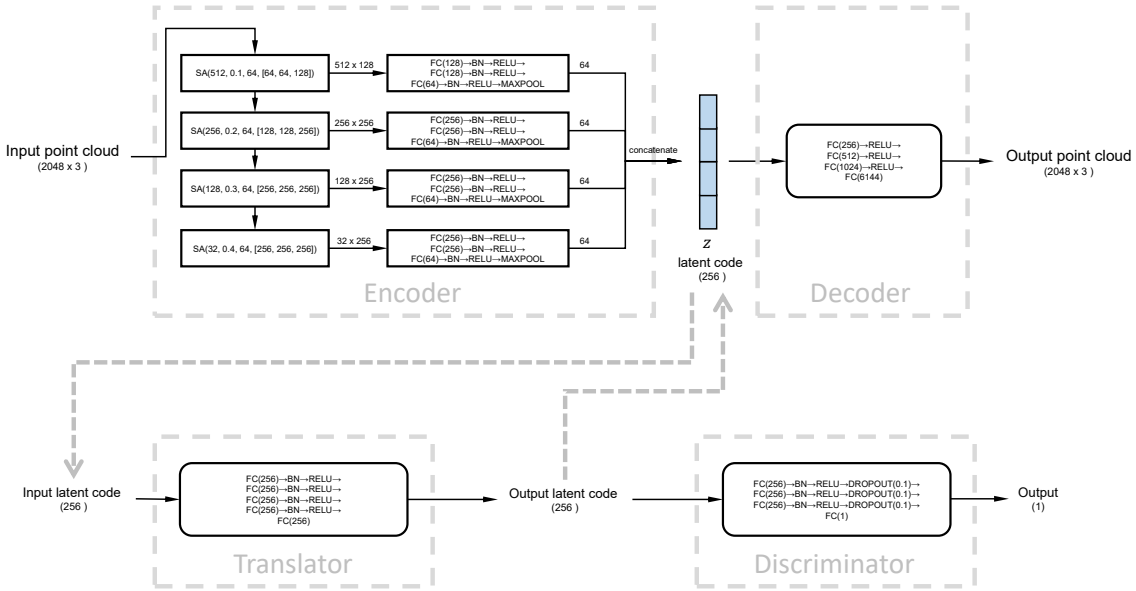


Figure 4.18: The architectures of our autoencoder, translator and discriminator.

4.6.2 Visualization of latent space

As a supplement to the visualization of latent spaces in section 4.4, we show in Figure 4.19 the joint embedding of latent sub-vectors constructed by different autoencoders for *chair-table* . Same with section 4.4, the embedding is generated by t-SNE with hamming distance after discretizing the values into integers.

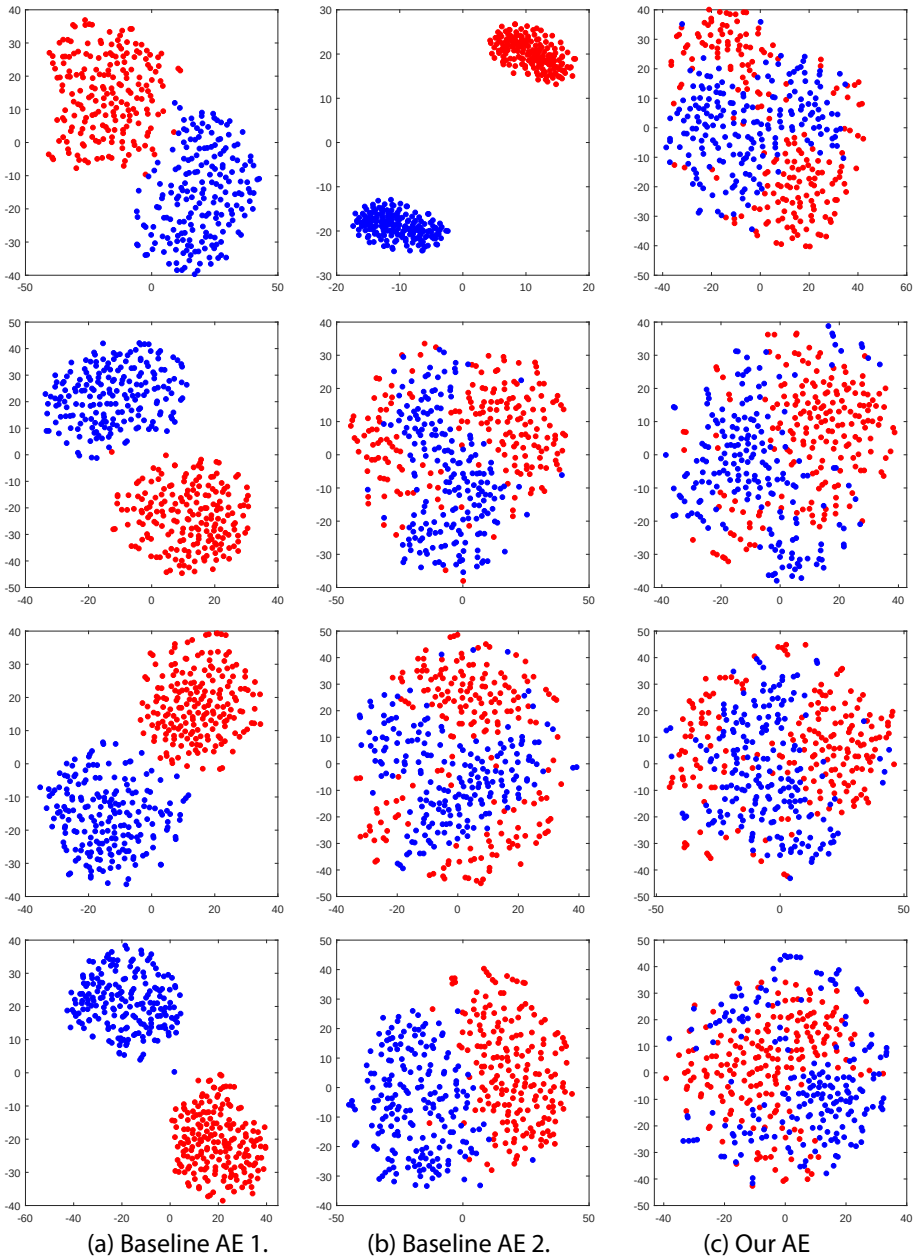


Figure 4.19: Joint embedding of the latent sub-vectors generated by our autoencoder(c) for chairs (red) and tables (blue). From top down, we show embedding result for z_1, z_2, z_3, z_4. For comparison, we also provide the embedding result for the corresponding sub-vectors in the codes generated by the two baseline autoencoders(a,b) described in Section 4.4 of the paper.

4.6.3 Additional Results

We show more visual results for *chair* $\rightarrow$ *table*, *table* $\rightarrow$ *chair*, *tallTable* $\rightarrow$ *shortTable*, *shortTable* $\rightarrow$ *tallTable*, $G \leftrightarrow R$, *thinG/R* $\leftrightarrow$ *thickG/R* in Figure 4.20, Figure 4.21, Figure 4.22, Figure 4.23, Figure 4.24 and Figure 4.25 respectively.

Figure 4.20: Results for *chair* → *table*. In each group (of 3 shapes), from left to right: input point cloud; output point cloud (2048 points); higher-resolution output point cloud (8192 points).

Figure 4.21: Results for *table* → *chair*. In each group (of 3 shapes), from left to right: input point cloud; output point cloud (2048 points); higher-resolution output point cloud (8192 points).

Figure 4.22: Results for *tall* → *short table*. In each group (of 3 shapes), from left to right: input point cloud; output point cloud (2048 points); higher-resolution output point cloud (8192 points).

Figure 4.23: Results for *short* → *tall table*. In each group (of 3 shapes), from left to right: input point cloud; output point cloud (2048 points); higher-resolution output point cloud (8192 points).

Figure 4.24: Results of different methods for $G \leftrightarrow R$.

Figure 4.25: Results of different methods for $thinG/R \leftrightarrow thickG/R$.

Chapter 5

Conclusions and Future Work

Shape-to-shape transformation is an important problem in computer graphics and geometric modeling. We believe that it is as fundamental to computer graphics as image-to-image translation is to computer vision. We regard our works as making the first two steps towards generic, cross-domain shape transforms. In this chapter, we conclude the thesis with a summary of the main contributions of our methods for learning shape transforms. Then, we discuss possible directions for future work.

5.1 Conclusions

We have presented P2P-NET and LOGAN, which are deep neural networks for learning shape transforms under two different settings. They are general-purpose in that no parts of the networks are tailor-made to specific transformation tasks. Moreover, we do not change the network architecture when dealing with different pairs of transformation domains.

P2P-NET is the first deep neural network designed to learn general-purpose transformations between point-based shape representations. It is a point-to-point displacement network trained under shape-level supervision to map point sets from one domain to another. The learned transform is agnostic to the dimensionality of the point sets, as the mapping is predicted from the feature space of the shape. The point sets can be in 2D or 3D spaces. As we demonstrated in Figure 3.1, the mapping can also lift 2D cross-sectional profiles to 3D shapes.

It is interesting that the shape transformations learned by P2P-NET are represented by point displacements that do not exist in the training set. That schema is generic and may generalize to other application domains as well. Furthermore, the bidirectionality of P2P-NET is achieved with a novel cross regularization loss to mutually enhance two directional transforms, which may also apply to other applications.

LOGAN is the first deep neural network designed for general-purpose, unpaired shape-to-shape translation. For most geometric modeling tasks, it is actually difficult to find pre-existing paired data. On the other hand, synthesizing paired 3D data is unrealistic since 3D

modeling is generally a non-trivial task. That is our motivation for developing unsupervised networks for learning shape-to-shape translation. Without changing the network's architecture or any of its hyperparameters, LOGAN can perform both content and style transfers between shapes. That is partially owing to the overcomplete, multi-scale representations it learns. LOGAN adapts its feature preservation solely based on the two input domains for training.

There is an implicit assumption in LOGAN that shapes from the two domains should share some commonalities. In general, these commonalities may be latent; they may reflect global or local features and represent either content or style. The key is for the network to learn the right commonalities and keep them during shape translations, in a way that is adaptive to the input domain pairs and the input shapes. Our network is designed with several important features to accomplish this: autoencoding shapes from two domains into a common latent space; the feature preservation loss; and perhaps most importantly, the use of multi-scale and overcomplete latent codes. It puts forth the interesting concept of implicit feature disentanglement, enabled by the overcomplete representation, which may hold potential in other application settings.

5.2 Future work

Our ultimate goal is to have machine learning assist in any transformative design tasks carried out by a graphics artist/modeler, where there are no ground truth targets, only a target domain as inspiration. The thesis only makes a first step towards this direction, laying a foundation for follow-up research.

We would like to consider transitive transformations for future work, where a source shape reaches a target via a sequence of two networks through an intermediate shape. To transform a 3D shape of a dog to a cat, one may first change the dog pose to a cat pose, then transfer the geometric style of cat to the reposed dog. On the other hand, the cat-dog example also reflects an idea of decoupling the shape structure with geometric style. We are interested in exploring how the structure prior learned from a set of 3D shapes will help the network learn shape transformation, especially for the purpose of style or content transfer.

The shape transform learned by P2P-NET, from paired shapes, is represented by point displacements which do not exist in our training set. We are interested in extending such learning schema to other application domains. For example, an image-to-image translation can be represented as a shift-map [153] that shifts the pixels on source image to obtain a target image. A neural network can generate such shift-map after trained by paired images.

Feature disentanglement is the main challenge we encountered in learning shape transform between unpaired domains. We are interested in exploring more ideas in addressing the challenge. Attention mechanism would be a good choice as it may allow the network to focus

on the right shape regions to extract features from. An explicit latent code disentanglement enforced by a regularization term is another direction we would try.

Finally, we are interested in investigating conditional shape-to-shape transformation for gaining more control over the transformation tasks. For example, when transforming a chair into a table, the user may want to ask the target table to have a rectangular top explicitly. By providing an artist such options to enforce more controls, it will make a more practical tool for shape modeling.

Bibliography

[1] Sema Berkiten, Maciej Halber, Justin Solomon, Chongyang Ma, Hao Li, and Szymon Rusinkiewicz. Learning detail transfer based on geometric features. In *Computer Graphics Forum (Eurographics)*, volume 36, pages 361–373, 2017.

[2] Ruizhen Hu, Wenchao Li, Oliver van Kaick, Hui Huang, Melinos Averkiou, Daniel Cohen-Or, and Hao Zhang. Co-Locating Style-Defining Elements on 3D Shapes. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 36(3):33:1–33:15, 2017.

[3] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

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

[6] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2017.

[7] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8798–8807, 2018.

[8] Oren Katzir, Dani Lischinski, and Daniel Cohen-Or. Cross-domain cascaded deep feature translation. *arXiv preprint arXiv:1906.01526*, 2019.

[9] Wallace Lira, Johannes Merz, Daniel Ritchie, Daniel Cohen-Or, and Hao Zhang. Ganhopper: Multi-hop gan for unsupervised image-to-image translation. *arXiv preprint arXiv:2002.10102*, 2020.

[10] Niloy J Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *Symposium on Geometry Processing*, pages 322–328, 2003.

[11] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. Edge-aware point set resampling. *ACM Transactions on Graphics (TOG)*, 32(1):9:1–9:12, 2013.

[12] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, pages 67–76, 2001.

[13] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29:1–29:13, 2013.

[14] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] Harry Blum et al. A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form*, 19(5):362–380, 1967.

[17] Frédéric Leymarie and Martin D. Levine. Simulating the grassfire transform using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (1):56–75, 1992.

[18] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 27(3):1–10, 2008.

[19] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 3d skeletons. In *Eurographics State of the Art Report*, 2016.

[20] Andrei Sharf, Thomas Lewiner, Ariel Shamir, and Leif Kobbelt. On-the-fly curve-skeleton computation for 3d shapes. In *Computer Graphics Forum*, volume 26, pages 323–328. Wiley Online Library, 2007.

[21] Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. Curve skeleton extraction from incomplete point cloud. In *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, pages 1–9. 2009.

[22] Junjie Cao, Andrea Tagliasacchi, Matt Olson, Hao Zhang, and Zhinxun Su. Point cloud skeletons via laplacian based contraction. In *Proc. of the IEEE International Conference on Shape Modeling and Applications (SMI)*, pages 187–197. IEEE, 2010.

[23] Hui Huang, Shihao Wu, Daniel Cohen-Or, Minglun Gong, Hao Zhang, Guiqing Li, and Baoquan Chen. L1-medial skeleton of point cloud. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 32(4):65–1, 2013.

[24] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *ACM Transactions on Graphics (Special Issue of SIGGRAPH Asia)*, pages 71–78, 1992.

[25] Jonathan C Carr, Richard K Beatson, Bruce C McCallum, W Richard Fright, Tim J McLennan, and Tim J Mitchell. Smooth surface reconstruction from noisy range data. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 119–ff, 2003.

[26] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. *Symposium on Geometry Processing*, pages 61–70, 2006.

[27] Hugues Hoppe. Poisson surface reconstruction and its applications. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 10–10, 2008.

[28] James Davis, Stephen R. Marschner, Matt Garr, and Marc Levoy. Filling holes in complex surfaces using volumetric diffusion. In *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*, pages 428–441, 2002.

[29] Andrei Sharf, Marc Alexa, and Daniel Cohen-Or. Context-based surface completion. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 23(3):878–887, 2004.

[30] G. Harary, A. Tal, and E. Grinspun. Context-based coherent surface completion. *ACM Transactions on Graphics (TOG)*, 33(1):5:1–5:12, 2014.

[31] M. Pauly, N. J. Mitra, J. Giesen, M. Gross, and L. Guibas. Example-based 3D scan completion. *Symposium on Geometry Processing*, pages 23–32, 2005.

[32] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M. Seitz. Schematic surface reconstruction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1498–1505, 2012.

[33] Kangxue Yin, Hui Huang, Hao Zhang, Minglun Gong, Daniel Cohen-Or, and Baoquan Chen. Morfit: interactive surface reconstruction from incomplete point clouds with curve-driven topology and geometry control. *ACM Transactions on Graphics (Special Issue of SIGGRAPH Asia)*, 33(6):202–1, 2014.

[34] Roee Lazar, Nadav Dym, Yam Kushinsky, Zhiyang Huang, Tao Ju, and Yaron Lipman. Robust optimization for topological surface reconstruction. *ACM Transactions on Graphics (TOG)*, 37(4):1–10, 2018.

[35] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhi-Quan Cheng. Style-content separation by anisotropic part scales. In *ACM Transactions on Graphics (Special Issue of SIGGRAPH Asia)*, pages 1–10. 2010.

[36] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):1–11, 2012.

[37] Qi-Xing Huang, Hao Su, and Leonidas Guibas. Fine-grained semi-supervised labeling of large shape collections. *ACM Transactions on Graphics (TOG)*, 32(6):1–10, 2013.

[38] Zhaoliang Lun, Evangelos Kalogerakis, and Alla Sheffer. Elements of style: learning perceptual shape style similarity. *ACM Transactions on graphics (TOG)*, 34(4):1–14, 2015.

[39] Isaak Lim, Anne Gehre, and Leif Kobbelt. Identifying style of 3d shapes using deep metric learning. In *Computer Graphics Forum*, volume 35, pages 207–215. Wiley Online Library, 2016.

[40] Zhaoliang Lun, Evangelos Kalogerakis, Rui Wang, and Alla Sheffer. Functionality preserving shape style transfer. *ACM Transactions on Graphics (TOG)*, 35(6):1–14, 2016.

[41] Fenggen Yu, Yan Zhang, Kai Xu, Ali Mahdavi-Amiri, and Hao Zhang. Semi-supervised co-analysis of 3d shape styles from projected lines. *ACM Transactions on Graphics (TOG)*, 37(2):1–17, 2018.

[42] Chongyang Ma, Haibin Huang, Alla Sheffer, Evangelos Kalogerakis, and Rui Wang. Analogy-driven 3d style transfer. In *Computer Graphics Forum*, volume 33, pages 175–184. Wiley Online Library, 2014.

[43] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[44] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[45] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3577–3586, 2017.

[46] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 36(4):72, 2017.

[47] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5648–5656, 2016.

[48] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 945–953, 2015.

[49] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[50] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, 2017.

[51] Yueqi Duan, Yu Zheng, Jiwen Lu, Jie Zhou, and Qi Tian. Structural relational reasoning of point clouds. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 949–958, 2019.

[52] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. Pcpnet learning local shape properties from raw point clouds. In *Computer Graphics Forum*.

[53] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. PointWeb: Enhancing local neighborhood features for point cloud processing. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5565–5573, 2019.

[54] Xiao Sun, Zhouhui Lian, and Jianguo Xiao. SRINet: Learning strictly rotation-invariant representations for point cloud classification and segmentation. In *ACM International Conference on Multimedia*, pages 980–988, 2019.

[55] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4606–4615, 2018.

[56] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3323–3332, 2019.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[58] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[59] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 1578–1587, 2019.

[60] Atzmon Matan, Maron Haggai, and Lipman Yaron. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 37(4):1–12, 2018.

[61] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762. Wiley Online Library, 2010.

[62] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2530–2539, 2018.

[63] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2589–2597, 2018.

[64] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 37(6), 2018.

[65] Roger Eckhardt, Stan Ulam, and Jon Von Neumann. the monte carlo method. *Los Alamos Science*, (15):131, 1987.

[66] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3D point clouds. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[67] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. SpiderCNN: Deep learning on point sets with parameterized convolutional filters. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

[68] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. *Proc. of the International Conference on Computer Vision (ICCV)*, 2019.

[69] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 828–838, 2018.

[70] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 52–66, 2018.

[71] M. Simonovsky and N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 29–38, 2017.

[72] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[73] Donald Meagher. Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer, 10 1980.

[74] Joaquín Huerta, Miguel Chover, Ricardo Quirós, Roberto Vivó, and José Ribelles. Binary space partitioning trees: a multiresolution approach. In *Proceedings. 1997 IEEE Conference on Information Visualization (Cat. No. 97TB100165)*, pages 148–154. IEEE, 1997.

[75] Aristides AG Requicha and Herbert B Voelcker. Constructive solid geometry. 1977.

[76] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3D point cloud models. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 863–872, 2017.

[77] Wei Zeng and Theo Gevers. 3dcontextnet: K-d tree guided hierarchical learning of point clouds using local and global contextual cues. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018.

[78] Matheus Gadelha, Subhransu Maji, and Rui Wang. Shape generation using spatially partitioned point clouds. In *Proc. of The British Machine Vision Conference (BMVC)*, 2017.

[79] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3d point cloud processing. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 103–118, 2018.

[80] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 700–708, 2017.

[81] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4990–4998, 2017.

[82] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[83] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 386–396, 2017.

[84] YiChang Shih, Sylvain Paris, Connelly Barnes, William T Freeman, and Frédo Durand. Style transfer for headshot portraits. *ACM Transactions on Graphics (TOG)*, 33(4):148, 2014.

[85] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 694–711. Springer, 2016.

[86] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 1511–1520, 2017.

[87] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[88] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[89] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[90] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 465–476, 2017.

[91] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1857–1865. JMLR. org, 2017.

[92] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 820–828, 2016.

[93] Aaron Gokaslan, Vivek Ramanujan, Daniel Ritchie, Kwang In Kim, and James Tompkin. Improving shape deformation in unsupervised image-to-image translation. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 649–665, 2018.

[94] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[95] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[96] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018.

[97] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 10551–10560, 2019.

[98] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 1501–1510, 2017.

[99] Kaidi Cao, Jing Liao, and Lu Yuan. Carigans: unpaired photo-to-caricature translation. *ACM Transactions on Graphics (Special Issue of SIGGRAPH Asia)*, 37(6):1–14, 2018.

[100] Wayne Wu, Kaidi Cao, Cheng Li, Chen Qian, and Chen Change Loy. Transgaga: Geometry-aware unsupervised image-to-image translation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8012–8021, 2019.

[101] Markus Gross and Hanspeter Pfister. *Point-Based Graphics*. Morgan Kaufman, 2007.

[102] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[103] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *Proc. of AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[104] Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 3d shape reconstruction from sketches via multi-view convolutional networks. In *3*.

[105] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 700–708, 2017.

[106] Mark Pauly, Richard Keiser, Leif P Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 22(3):641–650, 2003.

[107] Nina Amenta and Yong Joo Kil. Defining point-set surfaces. *ACM Transactions on Graphics (TOG)*, 23(3):264–270, 2004.

[108] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 9(1):3–15, 2003.

[109] Jeffrey P Grossman and William J Dally. Point sample rendering. In *Rendering techniques' 98*, pages 181–192. Springer, 1998.

[110] Matthias Müller, Richard Keiser, Andrew Nealen, Mark Pauly, Markus Gross, and Marc Alexa. Point based animation of elastic, plastic and melting objects. In *Symposium on Computer Animation*, pages 141–151, 2004.

[111] Martin Wicke, Denis Steinemann, and Markus Gross. Efficient animation of point-sampled thin shells. In *Computer Graphics Forum*, volume 24, pages 667–676. Citeseer, 2005.

[112] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, 36(1):301–329, 2017.

[113] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. In *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, volume 26, pages 22:1–22:6, 2007.

[114] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. In *ACM Transactions on Graphics (Special Issue of SIGGRAPH Asia)*, volume 28, pages 176:1–176:7, 2009.

[115] Reinhold Preiner, Oliver Mattausch, Murat Arikan, Renato Pajarola, and Michael Wimmer. Continuous projection for fast l1 reconstruction. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 33(4):47:1–47:13, July 2014.

[116] Shihao Wu, Hui Huang, Minglun Gong, Matthias Zwicker, and Daniel Cohen-Or. Deep points consolidation. *ACM Transactions on Graphics (Special Issue of SIGGRAPH Asia)*, 34(6):176:1–176:13, 2015.

[117] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[118] Minhyuk Sung, Hao Su, Vladimir G Kim, Siddhartha Chaudhuri, and Leonidas Guibas. Complementme: Weakly-supervised component suggestions for 3d modeling. *ACM Transactions on Graphics (Special Issue of SIGGRAPH Asia)*, 36(6):226:1–226:12, 2017.

[119] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 2017–2025, 2015.

[120] Shrinivasan Sankar and Adrien Bartoli. Model-based active learning to detect an isometric deformable object in the wild with a deep architecture. *Computer Vision and Image Understanding*, 171:69–82, 2018.

[121] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 286–301, 2016.

[122] Konstantinos Rematas, Chuong H Nguyen, Tobias Ritschel, Mario Fritz, and Tinne Tuytelaars. Novel views of objects from a single image. *IEEE transactions on pattern analysis and machine intelligence*, 39(8):1576–1590, 2016.

[123] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 18(6):914–924, 2012.

[124] Jeannette Bohg, Javier Romero, Alexander Herzog, and Stefan Schaal. Robot arm pose estimation through pixel-wise part classification. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3143–3150. IEEE, 2014.

[125] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Blensor: blender sensor simulation toolbox. *Advances in visual computing*, pages 199–208, 2011.

[126] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017.

[127] Yedid Hoshen and Lior Wolf. Nam: Non-adversarial unsupervised domain mapping. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018.

[128] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Proc. of Machine Learning Research*, 2018.

[129] Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. In *Proc. of the International Conference on Machine Learning (ICML)*, 2018.

[130] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. In *ACM Transactions on Graphics (TOG)*, volume 37, pages 1–15. ACM, 2018.

[131] Kangxue Yin, Hui Huang, Daniel Cohen-Or, and Hao Zhang. P2P-NET: Bidirectional point displacement net for shape transform. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 37(4):Article 152, 2018.

[132] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Chris Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Symposium on Geometry Processing*, pages 175–184. ACM, 2004.

[133] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 papers*, pages 80–es. 2007.

[134] Niloy Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, Vladimir Kim, and Qi-Xing Huang. Structure-aware shape processing. In *SIGGRAPH Asia 2013 Courses*, pages 1:1–1:20, 2013.

[135] Youyi Zheng, Hongbo Fu, Daniel Cohen-Or, Oscar Kin-Chung Au, and Chiew-Lan Tai. Component-wise controllers for structure-preserving shape manipulation. In *Computer Graphics Forum*, volume 30, pages 563–572. Wiley Online Library, 2011.

[136] Ibraheem Alhashim, Honghua Li, Kai Xu, Junjie Cao, Rui Ma, and Hao Zhang. Topology-varying 3D shape creation via structural blending. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 33(4):Article 158, 2014.

[137] Ibraheem Alhashim, Kai Xu, Yixin Zhuang, Junjie Cao, Patricio Simari, and Hao Zhang. Deformation-driven topology-varying 3d shape correspondence. *ACM Transactions on Graphics (Special Issue of SIGGRAPH Asia)*, 34(6):1–13, 2015.

[138] Hao Dong, Paarth Neekhara, Chao Wu, and Yike Guo. Unsupervised image-to-image translation with generative adversarial networks. In *Proc. of the International Conference on Machine Learning (ICML)*, 2017.

[139] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Proc. of the European Conference on Computer Vision (ECCV)*, volume 1, page 5, 2018.

[140] Ori Press, Tomer Galanti, Sagie Benaim, and Lior Wolf. Emerging disentanglement in auto-encoder based unsupervised image content transfer. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019.

[141] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 2366–2374, 2014.

[142] Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-scale convolutional neural networks for time series classification. *CoRR*, abs/1603.06995, 2016.

[143] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4), 2018.

[144] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (TOG)*, 37(4), 2018.

[145] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *Proc. of the International Conference on Machine Learning (ICML)*, 2018.

[146] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 4541–4550, 2019.

[147] Maciej Zamorski, Maciej Zięba, Piotr Klukowski, Rafał Nowak, Karol Kurach, Wojciech Stokowiec, and Tomasz Trzciński. Adversarial autoencoders for compact representations of 3d point clouds. *Computer Vision and Image Understanding*, 193:102921, 2020.

[148] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision (IJCV)*, 40(2):99–121, 2000.

[149] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 214–223, 2017.

[150] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 5767–5777, 2017.

[151] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing 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. *CoRR*, abs/1512.03012, 2015.

[152] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (Special Issue of SIGGRAPH Asia)*, 35(6):1–12, 2016.

[153] Yael Pritch, Eitam Kav-Venaki, and Shmuel Peleg. Shift-map image editing. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 151–158. IEEE, 2009.