LOGAN: Unpaired Shape Transform in Latent Overcomplete Space

KANGXUE YIN, Simon Fraser University
ZHIQIN CHEN, Simon Fraser University
HUI HUANG, Shenzhen University
DANIEL COHEN-OR, Tel Aviv University
HAO ZHANG, Simon Fraser University

We present LOGAN, a deep neural network aimed at learning generic shape transforms from *unpaired* domains. The network is trained on two sets of shapes, e.g., tables and chairs, but there is neither a pairing between shapes in the two domains to supervise the shape translation 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 encode *multi-scale* shape features in an *overcomplete* manner. 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 various 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 our network is able to learn what shape features to preserve during shape translations, either local or non-local, whether content or style, depending solely on the input domain pairs.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Shape modeling**; **Shape analysis**.

Additional Key Words and Phrases: Shape transform, unsupervised learning, unpaired domain translation, generative adversarial network, multi-scale point cloud encoding

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) [Taigman et al. 2017], CycleGAN [Zhu et al. 2017], DualGAN [Yi et al. 2017], MUNIT [Huang et al. 2018], among others [Almahairi et al. 2018; Gao et al. 2018; Hoffman et al. 2018; Hoshen and Wolf 2018; Liu et al. 2017]. 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 [Zhu et al. 2017]. The recently developed P2P-NET [Yin et al. 2018] 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.

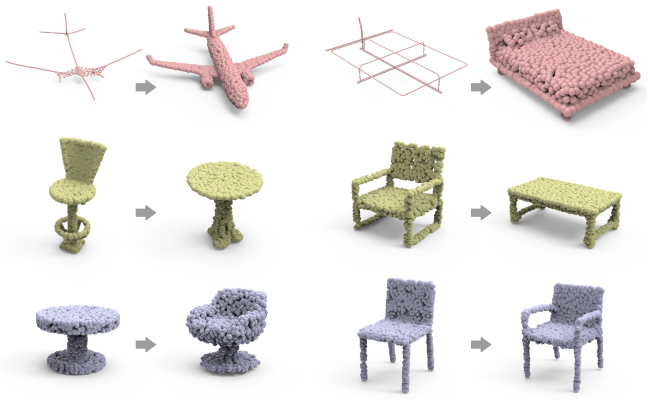
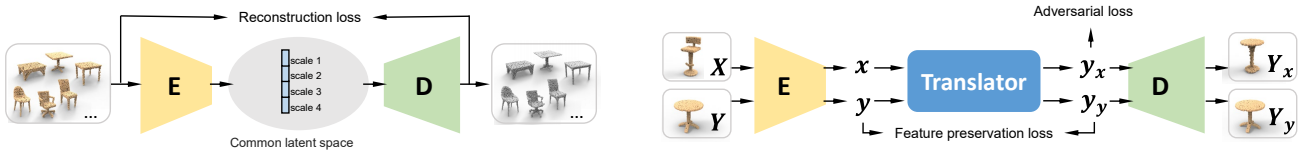


Fig. 1. Our deep neural network, LOGAN, learns shape transforms from *unpaired* domains. The same network can transform from skeletons to shapes, from cross-sectional profiles to surfaces, between chairs and tables; it can also add/remove parts, such as the armrests of a chair.

In this paper, we develop a deep neural network aimed at learning *generic* shape transforms from *unpaired domains*. The network is trained on two sets of shapes, e.g., tables and chairs, 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; the shapes may even have different point counts. Once trained, the network takes a point-set shape from one domain and transforms it into the other.

Without any point-to-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 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 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



(a) Autoencoder encodes shapes into overcomplete latent codes.

(b) Translator networks in latent space.

Fig. 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.

network must learn it without supervision. Our network is designed with two novel features to accomplish this:

- Our autoencoder encodes shape features at *multiple scales*, which is a common practice in convolutional neural networks (CNNs). However, unlike conventional approaches which aggregate the multi-scale featuers, e.g., in PointNET++ [Qi et al. 2017b], we *concatenate* the multi-scale features to produce a latent code with potential *redundancy*; we call such a latent code "*overcomplete*". Our motivation is that an overcomplete code for the shape transform would leave *more degree-of-freedom* to the translator network to facilitate an *implicit disentangling* of the preserved and altered shape features.
- 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 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.

Figure 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 [Zhu et al. 2017].

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 3.

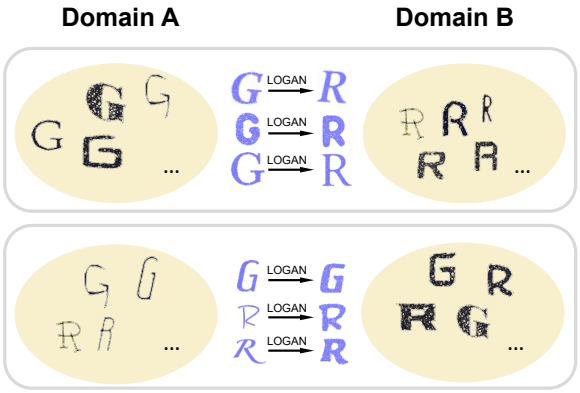


Fig. 3. Depending on the input (unpaired) domains, our network LOGAN can learn both content transfer (top row: from letters G to R, in varying font styles) and style transfer (bottom row: from thin to thick font strokes), without any change to the network architecture. In this example, 2D letters are represented by dense clouds of 2D points.

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 [Sorkine et al. 2004], structure preservation [Mitra et al. 2013], or topology variation [Alhashim et al. 2014]. On the other hand, our work falls into the category of 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. [2017] 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 [Zhu et al. 2017] and DualGAN [Yi et al. 2017] 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. [2017] 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 concatenated with a class/domain label. To enable image-to-image translation, they separately train an encoder to learn a mapping from an image to its

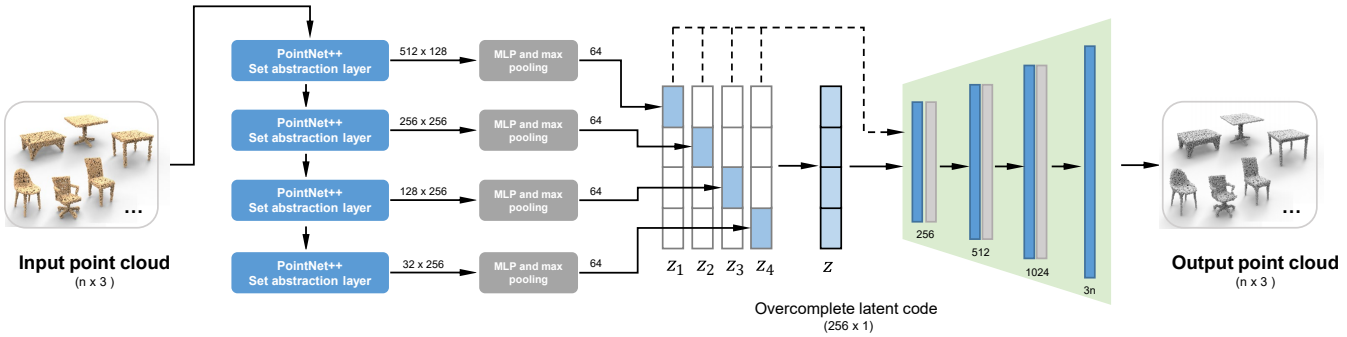


Fig. 4. Architecture of our multi-scale, overcomplete autoencoder. We use the set abstraction layers of PointNet++ [Qi et al. 2017b] to produce point features in different scales and aggregate them into four sub-vectors: z_1, z_2, z_3, and z_4. The four sub-vectors are padded with zeros and summed up 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.

latent code, which would serve as the noise input to the conditional GAN to generate a target image. In UNIT, Liu et al. [2017] 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 [Liu et al. 2017]. Both networks take inputs from two domains, map them into in 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. [2017] for reinforcing face identity preservation during emoji generation; it was later utilized in CycleGAN [Zhu et al. 2017] 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 [Huang et al. 2018; Lee et al. 2018]. Specifically, in MUNIT, a multi-modal extension of UNIT [Liu et al. 2017], Huang et al. [2018] relax the assumption of fully shared latent space between the two input domains by postulating that only part of the latent space, the *content*, 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. [2019] 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 an overcomplete latent representation would facilitate the disentangling of preserved and altered features.

Learning shape motions and transforms. Earlier work on spatial transformer networks [Jaderberg et al. 2015] allows deep convolutional models to learn invariance to translation, scale, rotation, and more generic shape warping for improved object recognition. Byravan and Fox [2017] 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 [Peng et al. 2018]. For shape transforms, Berkiten et al. [2017] present a metric learning approach for analogy-based mesh detail transfer. In P2P-NET, Yin et al. [2018] 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. [2018] 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.

Technically, the VAE-CycleGAN of Gao et al. [2018] 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 [Qi et al. 2017a], PointNET++ [Qi et al. 2017b], PCPNET [Guerrero et al. 2018], PointCNN [Li et al. 2018], and PCNN [Atzmon et al. 2018], have been developed for feature learning over point clouds. Generative models of point-set shapes [Achlioptas et al. 2018; Fan et al. 2017] and supervised, general-purpose point-set transforms [Yin et al. 2018] 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.

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 in a natural and intuitive manner, with special emphasis given to the preservation of common features. Other than assuming that such common features exist in both domains, we make no other assumption on them or their properties. 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.

3.1 Overview of networks and network loss

As shown in Figure 2, our network comprises of two parts that are trained in separate steps. First, an autoencoder is trained. The multi-scale encoder (Sec. 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 $\mathcal{Z}_{\mathcal{X}} = E(\mathcal{X})$ and $\mathcal{Z}_{\mathcal{Y}} = E(\mathcal{Y})$.

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*: Take $T_{\mathcal{X}\to\mathcal{Y}}$ in Figure 2 (b) for 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 tells the output codes from the ground-truth codes in $\mathcal{Z}_{\mathcal{Y}}$, to ensure 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 the cycle loss only acts as 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 common spaces 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. 3.3.

3.2 Multi-scale overcomplete autoencoder

Our multi-scale autoencoder is depicted in Figure 4. The input to our encoder is a set of n points. It passes through four set abstraction layers of PointNet++ [Qi et al. 2017b] with increasing sampling radius. The output point features from each of the four layers are further processed by 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 over-complete latent vector to the same decoder. Similar to [Achlioptas et al. 2018], our decoder is a multilayer perceptron (MLP) with 4

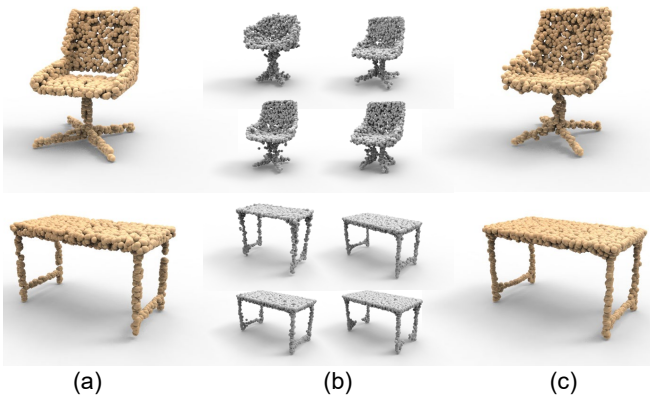


Fig. 5. Our autoencoder can encode an input point cloud (a) into 5 latent vectors shown in Figure 4, and decode them back to point clouds. The decoding output for the overcomplete latent code (c) is better than the 4 sub-vectors (b).

fully-connected layers. Each of the first 3 fully-connected layers are followed by a ReLU layer 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 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:

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

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. We use the earth mover distance (EMD) to measure the reconstruction losses.

Note that our current choices of the latent code length (256) and number of scales (4) were both experimental. 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.

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 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 6. We adopt WGAN [Arjovsky et al. 2017] in our implementation. To that end, we directly take the result of the output layer without sigmoid activation.

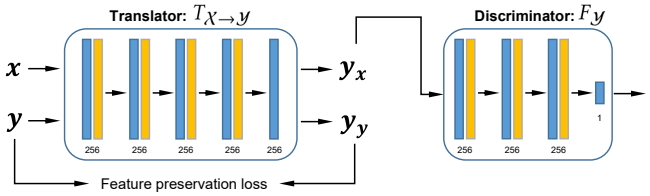


Fig. 6. Architecture of our translator network. The blue bars represent fully-connected layers; orange bars represent BN-ReLU layers.

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,

$$L_{\mathcal{X}\to\mathcal{Y}} = L_{\mathcal{X}\to\mathcal{Y}}^{\text{WGAN}} + \alpha L_{\mathcal{X}\to\mathcal{Y}}^{\text{FP}} \tag{2}$$

where α is a scalar weight set to 20 by default. Similar to WGAN [Gulrajani et al. 2017], the adversarial loss for $T_{\mathcal{X}\to\mathcal{Y}}$ is defined as:

$$L_{\mathcal{X}\to\mathcal{Y}}^{\text{WGAN}} = \mathbb{E}_{y\sim\mathbb{P}(\mathcal{Z}_{\mathcal{Y}})}[F_{\mathcal{Y}}(y)] - \mathbb{E}_{x\sim\mathbb{P}(\mathcal{Z}_{\mathcal{X}})}[F_{\mathcal{Y}}(y_x)] + \lambda_2 L_{GP} \tag{3}$$

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 [Gulrajani et al. 2017] 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 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 or transformed output vector $y_y = T_{\mathcal{X}\to\mathcal{Y}}(y)$:

$$L_{\mathcal{X}\to\mathcal{Y}}^{\text{FP}} = \mathbb{E}_{y\sim\mathbb{P}(\mathcal{Z}_{\mathcal{Y}})}[\|y - T_{\mathcal{X}\to\mathcal{Y}}(y)\|_1] \tag{4}$$

Training our translators with the loss function $L_{\mathcal{X}\to\mathcal{Y}}$ or $L_{\mathcal{Y}\to\mathcal{X}}$ has already produced reasonable result as shown in Figure 7. 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,

$$\begin{aligned} L_{\text{Cycle}} = {} & \mathbb{E}_{x\sim\mathbb{P}(\mathcal{Z}_{\mathcal{X}})}[\|T_{\mathcal{Y}\to\mathcal{X}}(T_{\mathcal{X}\to\mathcal{Y}}(x)) - x\|_1] \\ & + \mathbb{E}_{y\sim\mathbb{P}(\mathcal{Z}_{\mathcal{Y}})}[\|T_{\mathcal{X}\to\mathcal{Y}}(T_{\mathcal{Y}\to\mathcal{X}}(y)) - y\|_1] \end{aligned} \tag{5}$$

The overall loss function is defined as:

$$L_{\text{Overall}} = L_{\mathcal{X}\to\mathcal{Y}} + L_{\mathcal{Y}\to\mathcal{X}} + \beta L_{\text{Cycle}} \tag{6}$$

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:

$$T_{\mathcal{X}\to\mathcal{Y}}^*, T_{\mathcal{Y}\to\mathcal{X}}^* = \arg\min_T \max_F L_{\text{Overall}} \tag{7}$$

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. 4, we perform an ablation study to show that all loss terms play active roles in achieving high-quality results.

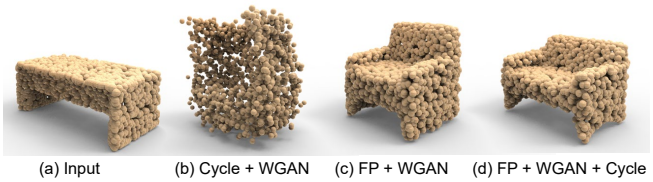


Fig. 7. Given an input table (a), WGAN + cycle loss (b) cannot translate it into a reasonable chair. Without the cycle loss, FP loss + WGAN can generate a reasonable output (c). Using the 3 losses together produces the output chair (d) that is visually most similar to the input table.

3.4 Training details

We train the autoencoder and the translator networks separately in two steps, since an insufficiently trained autoencoder can misguide the translators to 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 [Gulrajani et al. 2017] for 600 epochs. We set the number of discriminator iterations per generator iteration as two. The batch size we set for training the translator networks is 128. The learning rate starts from 0.002 and decays to 5e-4 during training. 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.

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, the network was trained with the *same* default network settings as described in Section 3 and the supplementary material. Unless otherwise specified, there is no hyperparameter or architecture tuning for any specific input datasets. All visual results are presented without any post-processing.

4.1 Shape transform results and ablation studies

The first domain pair on which we test our network is the chair and table datasets from ShapeNet [Chang et al. 2015], which contain mesh models. The chair dataset consists of 4,768 training shapes and 2,010 test shapes, while the table dataset has 5,933 training shapes 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. All output point clouds, e.g., in Figures 8-11, are in the same resolution of 2,048 points.

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 [Qi et al. 2017b], 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 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 (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 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 8 (a-d), 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 autoencoding by 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.

Joint embedding of latent codes. In Figure 9, we visualize the common latent spaces constructed by the three autoencoders 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 with a constant 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 autoencoder brings the chairs and tables closer together in the latent space since it is able to better discover their common 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, which can explain, in part, its superior performance for the translation task.

Comparing translator settings. In the second ablation study, we fix our autoencoder as presented in Figure 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 8, that the feature preservation loss has the most significant positive impact on cross-domain translation, while the cycle loss provides additional regularization for improved results.

Fig. 8. Comparing chair-table translation results (all in 2,048 points) using different network configurations. Top four rows: chair-to-table. Rest: table-to-chair. (a) Test input. (b) Result by default LOGAN. (c) Baseline AE 1 as autoencoder + our translator network. (d) Baseline AE 2 ($\lambda_1 = 0$) + our translator network. (e) Our autoencoder ($\lambda_1 = 0.1$) + WGAN & Cycle loss. (f) Our autoencoder ($\lambda_1 = 0.1$) + WGAN & feature preservation (FP) loss. (g) Most similar shape retrieved using EMD, from the target training domain. Note that the chair dataset from ShapeNET have some benches mixed in, which were retrieved as "tables."

Part insertion/removal. As an example of transferring or editing local shape structures, we show that LOGAN is able to learn to add/remove armrests for the chair dataset; see Figure 11. The dataset split was obtained from the chairs in ShapeNET by a handcrafted 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 seeing the input shapes from the two domains, without supervised training. In addition, the insertion/removal of the parts are carried out naturally.

For additional results and comparisons, including large and extended galleries like those presented in Figures 8 and 10, please refer to the supplementary material.

4.2 Comparison with supervised P2P-NET

In Figure 10, we show unpaired cross-domain shape transform results obtained by LOGAN, on several domain pairs from the recent

	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

Table 1. 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 10), and are measured against ground-truth target shapes from the P2P-NET dataset.

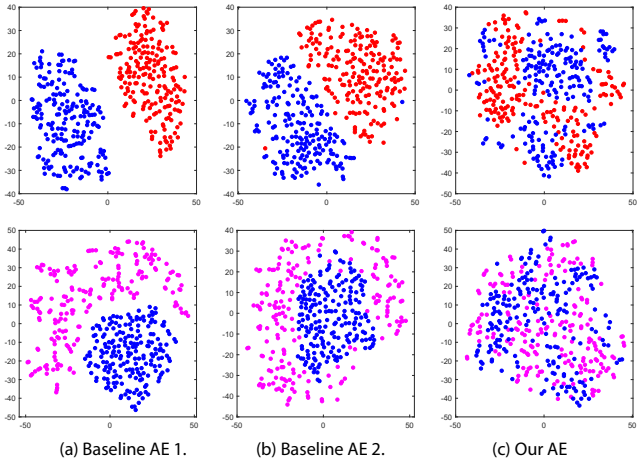


Fig. 9. Visualizing joint embeddings of table and chair latent codes produced by three autoencoders. Top row: red = chair, blue = table, *before* translation. Bottom: magenta = chair *after* translation; blue = original table. Our default AE brings the chairs and tables closer together in common latent space.

work P2P-NET [Yin et al. 2018], where the specific input shapes were also from their work. We compare these results to P2P-NET, as well as results from other network configurations as done in Figure 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 generic 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 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 1, our default LOGAN

	R→G		G→R	
	MSE	IOU	MSE	IOU
CycleGAN	0.234	0.413	0.248	0.411
UNIT	0.264	0.377	0.265	0.376
MUNIT	0.362	0.235	0.341	0.051
LOGAN (ours)	**0.210**	**0.466**	**0.216**	**0.477**

Table 2. Quantitative comparisons on *G-R* translation 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 *G-R* dataset. Better-performing numbers are highlighted in boldface.

network achieves the best quality, compared to other baseline alternatives, but still falls short of the supervised P2P-NET.

4.3 Unpaired style/content transfer and comparisons

Most deep networks for unpaired cross-domain translation work on images, aiming for content-preserving style transfer [Huang et al. 2018; Liu et al. 2017; Yi et al. 2017; Zhu et al. 2017]. 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, we train the networks to translate between regular letters of *G* and *R* (referred to as *thinG*s and *thinR*s), in various fonts, and their boldface versions (referred to as *thickG*s and *thickR*s), respectively, where stroke thickness is regarded as a style. Since most fonts in our collected dataset do not have paired regular and boldface types, 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 of them as *thinG/R*s. Then we randomly selected 500 *thinG/R*s and 500 *thickG/R*s to serve as test inputs while using the rest for training. In this case, an ideal translator would only change the stroke thickness of an input letter while keeping the letter in the same font.

The second test would assess the networks' capabilities for style-preserving *content transfer*, via a *G-R* translation task. From the 7,466 pairs of uppercase *G*s and *R*s, each in the same font, we randomly

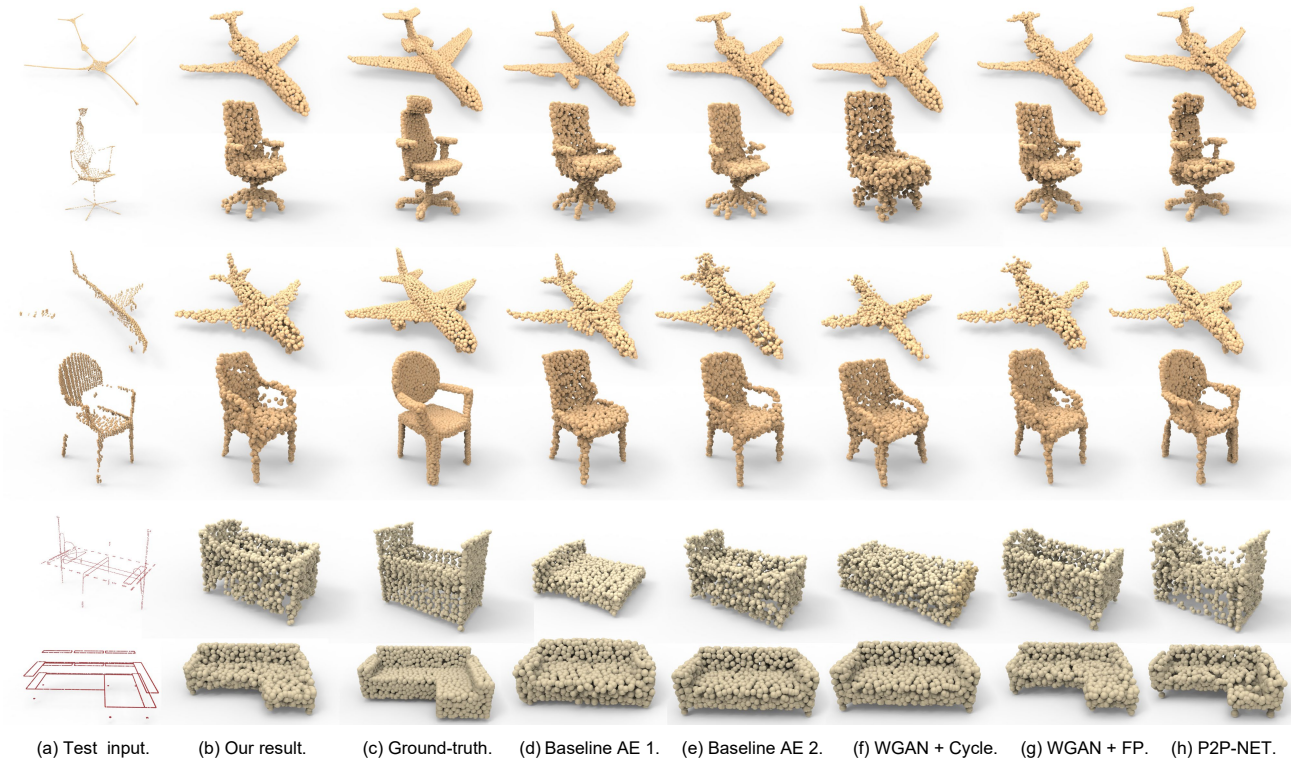


Fig. 10. Comparisons between various network configurations and (supervised) P2P-NET, on shape transform examples from P2P-NET: skeleton→shape (rows 1-2), scan→surface (rows 3-4), and (cross-sectional) profiles→surface (rows 5-6). Ground truth target shapes are shown as well. All results are in 2,048 points.

Fig. 11. Unpaired shape transforms between armchairs and armless chairs. First two rows show armrest removal while the last two rows show addition.

selected 1,000 to serve as the testing set while using the rest for training, without pairing any of the *G*s with *R*s in the same font. For this task, we expect an ideal translator to transform samples between *G* and *R* by changing the letter (the content) only, while preserving its style, e.g., font width, height, thickness, skewness.

We compare our method with three unpaired image translation networks: the original CycleGAN [Zhu et al. 2017] which translates images directly, as well as UNIT [Liu et al. 2017] and MUNIT [Huang et al. 2018], both of which translate latent codes. While our network LOGAN takes point clouds as inputs, the other networks all input and output images. We trained each of the four networks for 15 hours for *thinG/R-thickG/R* translation and 27 hours for *G-R* translation. To help with comparison, we convert the output point clouds from LOGAN to images as follows: 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 experiments.

Results from *thinG/R-thickG/R* translations are shown in Figures 12. All the networks performed decently for this style transfer task, which is expected out of CycleGAN, UNIT, and MUNIT. A close examination reveals that LOGAN tends to produce letter strokes with more regular thickness compared to the other networks.

Figure 13 shows comparison results for the style-preserving content transfer between *G*s and *R*s. We observe that CycleGAN, UNIT, and MUNIT are unable to learn transforms between global shape structures, which would be necessary for a *G-R* translation. In particular, the common latent code assumption of UNIT would not hold, hence the network was seemingly forced to copy the input. With a disentanglement between content and style codes, the results by MUNIT are different from those from UNIT, but they are still limited to localized changes to the inputs. This is because MUNIT

Fig. 12. Comparisons on a style transfer, i.e., *thinG/R-thickG/R* translation, by different methods. First four rows: thin-to-thick; last four rows: thick-to-thin. 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 trained on images; outputs from UNIT [Liu et al. 2017]; outputs from MUNIT [Huang et al. 2018].

Fig. 13. Comparisons on a style-preserving content transfer task, i.e., *G-R* translation, by different methods. Each two rows show a pair of translations between a *G* and an *R* in the same font.

is designed for content preservation, but there is no appropriate content features to preserve for the *G-R* translation task.

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 2 for *G-R* translation since we have ground-truth target letters. These results again demonstrate the superiority of LOGAN.

4.4 Comparison with unpaired deformation transfer

The latent VAE-CycleGAN developed by Gao et al. [2018] 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 [Gao et al. 2018]. In Figure 14, we compare results obtained by LOGAN and results from [Gao et al. 2018] as provided by the authors. The point clouds for training were obtained by uniformly sampling 2,048 points from the corresponding meshes. The output point clouds contain the same number of points. Note that the *horse* → *camel* dataset contains a total of 384 shapes in the training set; *hand* → *pants* contains 342 shapes; *fit* → *fat* contains 583 shapes; and *flamingo* → *human* only contains 102 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 to 1,200.

The results show that LOGAN is able to learn to preserve pose-related features and achieve pose-preserving shape transform, like Gao et al. [2018]. However, since our current implementation of the network is limited by the point resolution (to 2,048), both the input and output point clouds have lower surface quality as reflected by missing surface details and scattered noise.

4.5 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 15, 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: skeleton→shape, *G*→*R*, and chair→table. The plot is produced on the test sets and based on the mean magnitude of code changes.

We can observe that our network automatically learns the right features to preserve (e.g., more global or coarser-scale features for skeleton→shape and more local features for *G*→*R*), solely based on the input domain pairs. For the chair→table translation, coarse- and

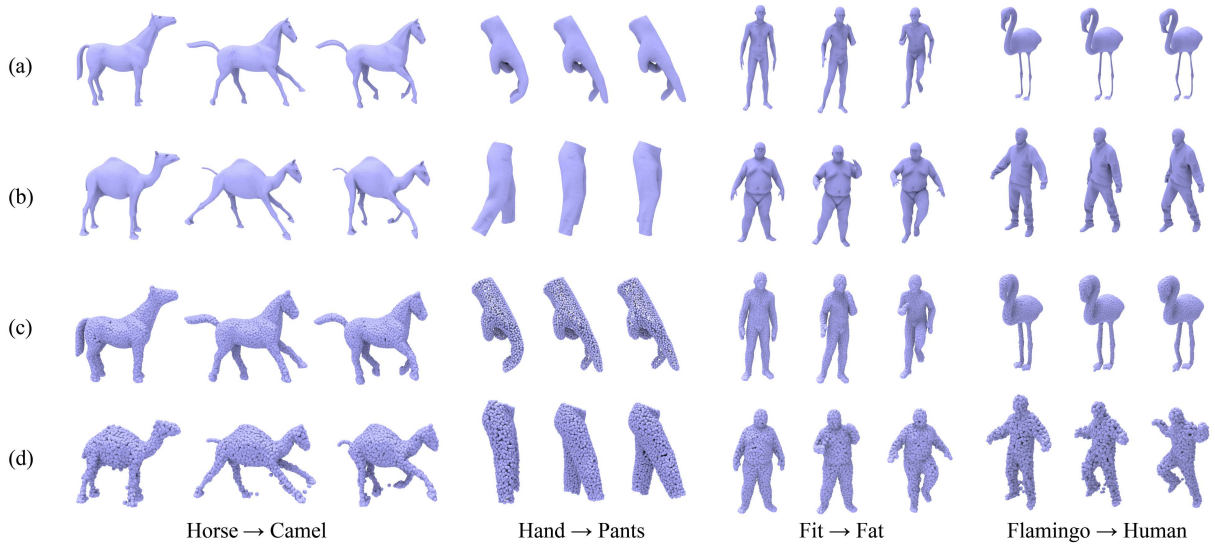


Fig. 14. Comparisons with unpaired deformation transfer [Gao et al. 2018]. (a) Input meshes; (b) output meshes by Gao et al. [2018]; (c) input point clouds sampled from (a) using 2,048 points; (d) output point clouds also with 2, 048 points, by LOGAN, our generic cross-domain shape transform network.

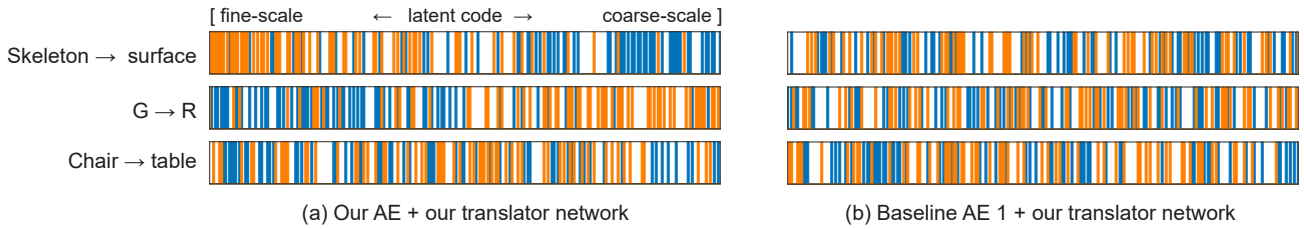


Fig. 15. Visualizing "disentanglement" in latent code preservation (blue color) and alteration (orange color) during translation, where each bar represents one dimension of the latent code. Orange bars: top 64 latent code dimensions with the largest average changes. Blue bars: dimensions with smallest code changes.

fine-level level features are both impacted. In addition, compared to PointNET++ encoding, our default autoencoder with overcomplete codes better disentangles parts of the latent codes that are preserved vs. altered. Note that unlike MUNIT [Huang et al. 2018], LOGAN does not impose any explicit code separation/disentanglement, the implicit disentanglement is likely the result of having more degrees-of-freedom in the overcomplete latent codes.

5 DISCUSSION, LIMITATION, AND FUTURE WORK

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 modelnig is generally a non-trivial task; this is the very reason why we seek learning methods to automate the process.

Cross-domain shape translation is not suitable for all domain pairs, e.g., (chair, airplane) or (human body, face). Hence, there is an implicit assumption 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

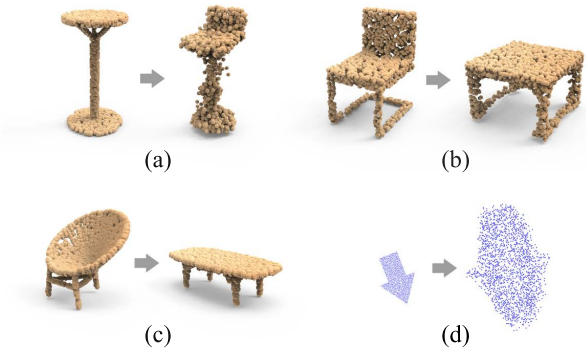


Fig. 16. Several failure cases by LOGAN: (a) scattered point clouds; (b-c) unnatural transform results due to lack of commonalities in the target domain; (c) failed translation between shapes differing in global scales.

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.

Another aspect of our work is the separation of autoencoder training from the latent cross-domain translation network. Unlike most previous works on unpaired cross-domain image translation, e.g., [Huang et al. 2018; Liu et al. 2017], we do not use a combined loss. Similar to Gao et al. [2018], we train the autoencoder and translator separately. We also believe that the separation facilitates training of the GAN translators and leads to improved results.

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 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 16(a) and some results in Figure 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 16(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 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 16(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.

Last but not least, 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 16 (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.

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 [Huang et al. 2018] to gain more control on the transform tasks are also worth investigating.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018. Learning Representations and Generative Models for 3D Point Clouds. In *Proc. of ICML*.

Ibraheem Alhashim, Honghua Li, Kai Xu, Junjie Cao, Rui Ma, and Hao Zhang. 2014. Topology-Varying 3D Shape Creation via Structural Blending. *ACM Trans. on Graph* 33, 4 (2014), Article 158.

Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. 2018. Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data. In *Proc. of ICML*.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *Proc. of ICML*. 214–223.

Matan Atzmon, Haggai Maron, and Yaron Lipman. 2018. Point Convolutional Neural Networks by Extension Operators. *ACM Trans. on Graph* 37, 4 (2018).

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

Arunkumar Byravan and Dieter Fox. 2017. SE3-Nets: Learning Rigid Body Motion using Deep Neural Networks. In *Proc. of ICRA*. IEEE, 173–180.

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. 2015. ShapeNet: An Information-Rich 3D Model Repository. *CoRR* abs/1512.03012 (2015).

Hao Dong, Paarth Neekhara, Chao Wu, and Yike Guo. 2017. Unsupervised image-to-image translation with generative adversarial networks. In *Proc. of ICML*.

Haoqiang Fan, Hao Su, and Leonidas Guibas. 2017. A point set generation network for 3d object reconstruction from a single image. In *Proc. of CVPR*.

Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. 2018. Automatic unpaired shape deformation transfer. In *ACM Trans. on Graph*, Vol. 37. ACM, 1–15.

Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. 2018. PCPNET: Learning Local Shape Properties from Raw Point Clouds. In *Computer Graphics Forum (Eurographics)*, Vol. 37. Wiley Online Library, 75–85.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of Wasserstein GANs. In *Proc. of NeurIPS*. 5767–5777.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. 2018. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *Proc. of Machine Learning Research*.

Yedid Hoshen and Lior Wolf. 2018. NAM: Non-Adversarial Unsupervised Domain Mapping. In *Proc. of ECCV*.

Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. 2018. Multimodal Unsupervised Image-to-Image Translation. In *Proc. of ECCV*.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. 2015. Spatial transformer networks. In *Proc. of NeurIPS*. 2017–2025.

Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. 2018. Diverse image-to-image translation via disentangled representations. In *Proc. of ECCV*, Vol. 1. 5.

Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. PointCNN: Convolution On X-Transformed Points. In *Proc. of NeurIPS*. 828–838.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. 2017. Unsupervised image-to-image translation networks. In *Proc. of NeurIPS*. 700–708.

Niloy Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, Vladimir Kim, and Qi-Xing Huang. 2013. Structure-aware Shape Processing. In *SIGGRAPH Asia 2013 Courses*. 1:1–1:20.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills. *ACM Trans. on Graph* 37, 4 (2018).

Ori Press, Tomer Galanti, Sagie Benaim, and Lior Wolf. 2019. Emerging Disentanglement in Auto-Encoder Based Unsupervised Image Content Transfer. In *Proc. of ICLR*.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of CVPR*.

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of NeurIPS*.

Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Chris Rössl, and Hans-Peter Seidel. 2004. Laplacian surface editing. In *Symp. on Geom. Proc.* ACM, 175–184.

Yaniv Taigman, Adam Polyak, and Lior Wolf. 2017. Unsupervised cross-domain image generation. In *Proc. of ICLR*.

Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. In *Proc. of ICCV*.

Kangxue Yin, Hui Huang, Daniel Cohen-Or, and Hao Zhang. 2018. P2P-NET: Bidirectional Point Displacement Net for Shape Transform. *ACM Trans. on Graph* 37, 4 (2018), Article 152.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. of ICCV*.