C-Flow: Conditional Generative Flow Models for Images and 3D Point Clouds

Albert Pumarola[1,*] Stefan Popov[2] Francesc Moreno-Noguer[1] Vittorio Ferrari[2]
[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain
[2]Google Research, Zürich, Switzerland

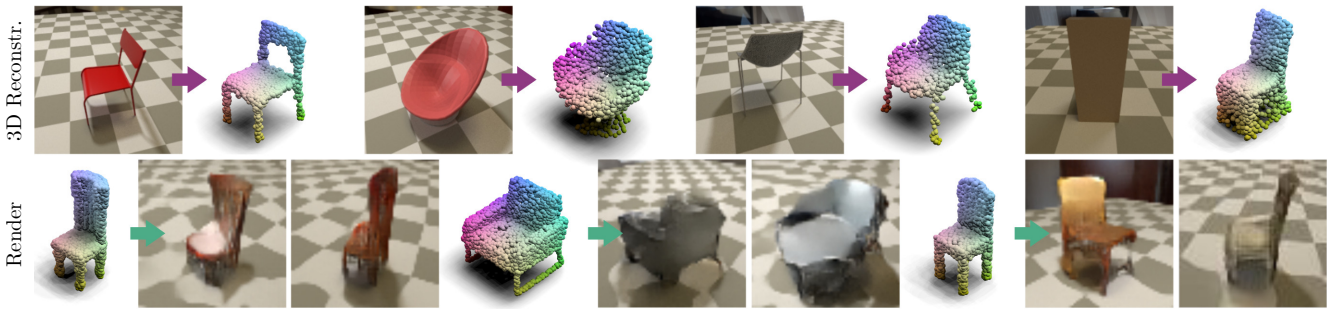


Figure 1: We propose C-Flow, a conditioning scheme for flow-based generative models applicable to many different domains. The figure shows the results of modeling the conditional distributions *image* $\leftrightarrow$ *3D point cloud*. In the top row we apply this model for 3D reconstruction (*image* $\rightarrow$ *point cloud*), and in the bottom row for rendering new images (*point cloud* $\rightarrow$ *image*). Our model allows sampling multiple times from this conditional distribution to generate several renderings of the same point cloud.

Abstract

Flow-based generative models have highly desirable properties like exact log-likelihood evaluation and exact latent-variable inference, however they are still in their infancy and have not received as much attention as alternative generative models. In this paper, we introduce C-Flow, a novel conditioning scheme that brings normalizing flows to an entirely new scenario with great possibilities for multi-modal data modeling. C-Flow is based on a parallel sequence of invertible mappings in which a source flow guides the target flow at every step, enabling fine-grained control over the generation process. We also devise a new strategy to model unordered 3D point clouds that, in combination with the conditioning scheme, makes it possible to address 3D reconstruction from a single image and its inverse problem of rendering an image given a point cloud. We demonstrate our conditioning method to be very adaptable, being also applicable to image manipulation, style transfer and multi-modal image-to-image mapping in a diversity of domains, including RGB images, segmentation maps and edge masks.

1. Introduction

Generative models have become extremely popular in the machine learning and computer vision communities. Two main actors currently prevail in this scenario, Variational Autoencoders (VAEs) [26] and especially Generative Adversarial Networks (GANs) [17]. In this paper we focus on a different family, the so-called flow-based generative models [13], which remain under the shadow of VAEs and GANs despite offering very appealing properties. Compared to other generative method, flow-based models build upon a sequence of reversible mappings between the input and latent space that allow for (1) exact latent-variable inference and log-likelihoood evaluation, (2) efficient and parallelizable inference and synthesis and (3) useful and simple data manipulation by operating directly on the latent space.

The main contribution of this paper is a novel approach to condition normalizing flows, making it possible to perform multi-modality transfer tasks which have so far not been explored under the umbrella of flow-based generative models. For this purpose, we introduce C-Flow, a framework consisting of two parallel flow branches, interconnected across their reversible functions using conditional coupling layers and trained with an invertible cycle consistency. This scheme allows guiding a source domain towards a target domain guaranteeing the satisfaction of the aforementioned properties of flow-based models. Conditional inference is then implemented in a simple manner, by (exactly) embedding the source sample into its latent space, sampling a point from a Gaussian prior, and then propagating them through the learned normalizing flow. For example, for the application of synthesizing multiple plausible photos given a semantic segmentation mask, each image is generated by jointly propagating the segmentation embedding and a random point drawn from a prior distribution across the learned flow.

*Work done while interning at Google.

Our second contribution is a strategy to enable flow-based methods to model unordered 3D point clouds. Specifically, we introduce (1) a re-ordering of the 3D data points according to a Hilbert sorting scheme, (2) a global feature operation compatible with the reversible scheme, and (3) an invertible cycle consistency that penalizes the Chamfer distance. Combining this strategy with the proposed conditional scheme we can then address tasks such as shape interpolation, 3D object reconstruction from an image, and rendering an image given a 3D point cloud (Fig. 1).

Importantly, our new conditioning scheme enables a wide range of tasks beyond 3D point cloud modeling. In particular, we are the first flow-based model to show mapping between a large diversity of domains, including image-to-image, pointcloud-to-image, edges-to-image segmentation-to-image and their inverse mappings. Also, we are the first to demonstrate application in image content manipulation and style transfer tasks.

We believe our conditioning scheme, and its ability to deal with a variety of domains, opens the door to building general-purpose and easy to train solutions. We hope all this will spur future research in the domain of flow-based generative models.

2. Related Work

Flow-Based Generative Models. Variational Auto-Encoders (VAEs) [26] and Generative Adversarial Networks (GANs) [17] are the most studied deep generative models so far. VAEs use deep networks as function approximators and maximize a lower bound on the data log-likelihood to model a continuous latent variable with intractable posterior distribution [42, 47, 28]. GANs, on the other hand, circumvent the need for dealing with likelihood estimation by leveraging an adversarial strategy. While GANs' versatility has made possible advances in many applications [23, 37, 61, 11, 6, 1], their training is unstable [41] and requires careful hyper-parameter tuning.

Flow-based generative models [13, 43] have received little attention compared to GANs and VAEs, despite offering very attractive properties such as the ability to estimate exact log-likelihood, efficient synthesis and exact latent-variable inference. Further advances have been proposed in RealNVP [14] by introducing the affine coupling layers and in Glow [25], through an architecture with 1x1 invertible convolutions for image generation and editing. These works have been later applied to audio generation [36, 24, 56, 45], image modeling [49, 19, 8] and video prediction [27].

Some recent works have proposed strategies for conditioning normalizing flows by combining them with other generative models. For instance, [29, 19] combine flows with GANs. These models, however, are more difficult to train as adversarial losses tend to introduce instabilities. Similarly, for the specific application of video prediction, [27] enforces an autoregressive model onto the past latent variables to predict them in the future. Dual-Glow [49] uses a conditioning scheme for MRI-to-PET brain scan mapping by concatenating the prior distribution of the source image with the latent variables of the target image.

In this paper, we introduce a novel mechanism to condition flow-based generative models by enforcing a source-to-target coupling at every transformation step instead of only feeding the source information into the target prior distribution. As we show experimentally, this enables fine-grained control over the modeling process (Sec. 7).

Modeling and reconstruction of 3D Shapes. The success of deep learning has spurred a large number of discriminative approaches for 3D reconstruction [9, 38, 51, 46, 18, 58]. These techniques, however, only learn direct mappings between output shapes and input images. Generative models, in contrast, capture the actual shape distribution from the training set, enabling not only to reconstruct new test images, but also to sample new shapes from the learned distribution. There exist several works along this line. For instance, GANs have been used in Wu *et al*. [54] to model objects in a voxel representation; Hamu *et al*. [4] used them to model body parts; and Pumarola *et al*. [39] to learn the manifold of geometry images representing clothed 3D bodies. Auto-encoders [12, 48] and VAEs [15, 3, 30, 20] have also been applied to model 3D data. More recently, Joon Park *et al*. [32] used auto-decoders [5, 16] to represent shapes with continuous volumetric fields. All previous techniques are not bijective, and thus, not directly applicable to our model.

PointFlow [57] is the only approach that uses normalizing flows to model 3D data. They learn a generative model for point clouds by first modeling the distribution of object shapes and then applying normalizing flows to model the point cloud distribution for each shape. This strategy, however, cannot condition the shape, preventing PointFlow from being used in applications such as 3D reconstruction and rendering. Also, its inference time is very high, as point clouds are generated one point at a time, while we generate the entire point cloud in one forward pass.

3. Flow-Based Generative Model

Flow-based generative models aim to approximate an unknown true data distribution $\mathbf{x} \sim p^*(\mathbf{x})$ from a limited set of observations $\{\mathbf{x}^{(i)}\}_{i=1}^N$. The data is modeled by learning an invertible transformation $\mathbf{g}_{\boldsymbol{\theta}}(\cdot)$ mapping a latent space with tractable density $p_{\boldsymbol{\vartheta}}(\mathbf{z})$ to $\mathbf{x}$:

$$\mathbf{z} \sim p_{\boldsymbol{\vartheta}}(\mathbf{z}), \quad \mathbf{x} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}), \tag{1}$$

where $\mathbf{z}$ is a latent variable and $p_{\boldsymbol{\vartheta}}(\mathbf{z})$ is typically a Gaussian distribution $\mathbf{N}(\mathbf{z}; 0, \mathbf{I})$. The function $\mathbf{g}_{\boldsymbol{\theta}}$, commonly known as a normalizing flow [43], is bijective, meaning that given a data point $\mathbf{x}$ its latent-variable $\mathbf{z}$ is computed as:

$$\mathbf{z} = \mathbf{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{x}), \tag{2}$$

where $\mathbf{g}_{\boldsymbol{\theta}}^{-1}$ is composed of a sequence of K invertible transformations $\mathbf{g}^{-1} = \mathbf{g}_1^{-1} \circ \mathbf{g}_2^{-1} \circ \cdots \circ \mathbf{g}_K^{-1}$ defining a mapping between $\mathbf{x}$ and $\mathbf{z}$ such that:

$$\mathbf{x} \triangleq \mathbf{h}_0 \overset{\mathbf{g}_1^{-1}}{\longleftrightarrow} \mathbf{h}_1 \overset{\mathbf{g}_2^{-1}}{\longleftrightarrow} \mathbf{h}_2 \cdots \overset{\mathbf{g}_K^{-1}}{\longleftrightarrow} \mathbf{h}_K \triangleq \mathbf{z}, \tag{3}$$

The goal of generative models is to find the parameters $\boldsymbol{\theta}$ such that $p_{\boldsymbol{\theta}}(\mathbf{x})$ best approximates $p^*(\mathbf{x})$. Explicitly modeling such probability density function is usually intractable, but using the normalizing flow mapping of Eq. (1) under the change of variable theorem, we can compute the exact log-likelihood for a given data point $\mathbf{x}$ as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log p_{\boldsymbol{\vartheta}}(\mathbf{z}) + \log \left| \det(\partial \mathbf{z} / \partial \mathbf{x}) \right| \tag{4}$$

$$= \log p_{\boldsymbol{\vartheta}}(\mathbf{z}) + \sum_{i=1}^{K} \log \left| \det(\partial \mathbf{h}_i / \partial \mathbf{h}_{i-1}) \right| \tag{5}$$

where $\partial \mathbf{h}_i / \partial \mathbf{h}_{i-1}$ is the Jacobian matrix of $\mathbf{g}_i^{-1}$ at $\mathbf{h}_{i-1}$ and the Jacobian determinant measures the change of log-density made by $\mathbf{g}_i^{-1}$ when transforming $\mathbf{h}_{i-1}$ to $\mathbf{h}_i$. Since we can now compute the exact log-likelihood, the training criterion of flow-based generative model is directly the negative log-likelihood over the observations. Note that optimizing over the actual log-likelihood of the observations is more stable and informative than doing it over a lower-bound of the log-likelihood for VAEs, or minimizing the adversarial loss in GANs. This is one of the major virtues of flow-based approaches.

4. Conditional Flow-Based Generative Model

Given a true data distribution $(\mathbf{x}_\text{A}, \mathbf{x}_\text{B}) \sim p^*(\mathbf{x}_\text{A}, \mathbf{x}_\text{B})$. Our goal is to learn a model for $\mathbf{x}_\text{B} \sim p^*(\mathbf{x}_\text{B}|\mathbf{x}_\text{A})$ to map sample points from domain A to domain B. For example, for the application of 3D reconstruction, $\mathbf{x}_\text{A}$ would be an image and $\mathbf{x}_\text{B}$ a 3D point cloud. To this end, we propose a conditional flow-based generative model extending [14, 25]. Our L-levels model, learns both distributions with two bijective transformations $\mathbf{g}_{\boldsymbol{\theta}}$ and $\mathbf{f}_{\boldsymbol{\phi}}$ (Fig. 2):

$$\mathbf{z}_\text{A} \sim p_{\boldsymbol{\vartheta}}(\mathbf{z}_\text{A}), \quad \mathbf{z}_\text{B} \sim p_{\boldsymbol{\varphi}}(\mathbf{z}_\text{B}) \tag{6}$$

$$\mathbf{x}_\text{A} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}_\text{A}), \quad \mathbf{x}_\text{B} = \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{z}_\text{B}|\mathbf{z}_\text{A}) \tag{7}$$

$$\mathbf{z}_\text{A} = \mathbf{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{x}_\text{A}), \quad \mathbf{z}_\text{B} = \mathbf{f}_{\boldsymbol{\phi}}^{-1}(\mathbf{x}_\text{B}|\mathbf{x}_\text{A}) \tag{8}$$

where $\mathbf{z}_\text{A}$ and $\mathbf{z}_\text{B}$ are latent-variables, and $p_{\boldsymbol{\vartheta}}(\mathbf{z}_\text{A})$ and $p_{\boldsymbol{\varphi}}(\mathbf{z}_\text{B})$ are tractable spherical multivariate Gaussian distributions with learnable mean and variance. Note that conditioning on $\mathbf{z}_A$ or $\mathbf{x}_A$ is equivalent, as they are related by a bijective transformation.

We then define the mapping $\mathcal{M}$ to sample $\mathbf{x}_\text{B}$ conditioned on $\mathbf{x}_\text{A}$, as a three-step operation:

$$\mathbf{z}_\text{A} = \mathbf{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{x}_\text{A}) \qquad \text{encode condition } \mathbf{x}_\text{A} \tag{9}$$

$$\mathbf{z}_\text{B} \sim p_{\boldsymbol{\varphi}}(\mathbf{z}_\text{B}) \qquad \text{sample latent-variable } \mathbf{z}_\text{B} \tag{10}$$

$$\mathbf{x}_\text{B} = \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{z}_\text{B}|\mathbf{z}_\text{A}) \qquad \text{generate } \mathbf{x}_\text{B} \text{ cond. on } \mathbf{x}_\text{A} \tag{11}$$

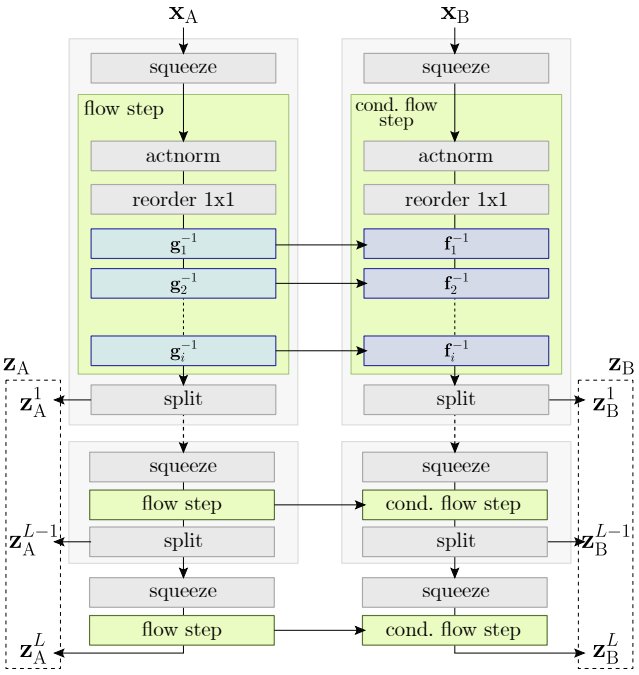


Figure 2: **The C-Flow model** consists of two parallel flow branches mutually interconnected with conditional coupling layers. This scheme allows sampling $\mathbf{x}_\text{B}$ conditioned on $\mathbf{x}_\text{A}$. For a detailed description on functions in grey refer to [25].

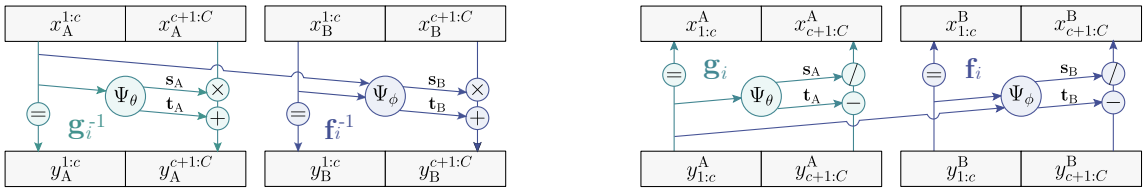
In the following subsections we describe how this conditional framework is implemented. Sec. 4.1 discusses the foundations of the conditional coupling layer we propose to map source to target data using invertible functions, and how its Jacobian is computed. Sec. 4.2 describes the architecture we define for the practical implementation of the coupling layers. Sec. 4.3 presents an invertible cycle consistency loss introduced to further stabilize the training process. Finally, in Sec. 4.4 we define the total training loss.

4.1. Conditional Coupling Layer

When designing the conditional coupling layer we need to fulfill the constraint that each transformation has to be bijective and tractable. As shown in [13, 14], both these issues can be overcome by choosing transformations with triangular Jacobian. In this case their determinant is calculated as the product of diagonal terms, making the computation tractable and ensuring invertibility. Motivated by these works, we propose an extension of their coupling layer to account for cross-domain conditioning. A schematic of the proposed layers is shown in Fig. 3. Formally, let us define $y \triangleq h_i$ and $x \triangleq h_{i-1}$. We then write the invertible function $\mathbf{f}^{-1}$ to transform a data point $\mathbf{x}_\text{B}$ based on $\mathbf{x}_\text{A}$ as follows:

$$\begin{cases} y_\text{B}^{1:c} & = x_\text{B}^{1:c} \\ y_\text{B}^{c+1:C} & = x_\text{B}^{c+1:C} \odot \exp\left(s\left(x_\text{A}^{1:c}, x_\text{B}^{1:c}\right)\right) + t\left(x_\text{A}^{1:c}, x_\text{B}^{1:c}\right) \end{cases},$$

where C is the number of channel dimensions in both data points, $\odot$ denotes element-wise multiplication and s and t



(a) Forward Propagation (b) Backward Propagation

Figure 3: **Conditional coupling layer for forward and backward propagation.** Given two input tensors $\mathbf{x}_\text{A}$ and $\mathbf{x}_\text{B}$, the proposed conditional coupling layer transforms the second half of $\mathbf{x}_\text{B}$ conditioned on the first halves of $\mathbf{x}_\text{A}$ and $\mathbf{x}_\text{B}$. The first halves of all tensors are not updated. By sequentially concatenating these bijective operations we can transform data points x into their latent representation y (forward propagation) and vice versa (backward propagation).

are the scale and translation functions $(\mathbb{R}^c, \mathbb{R}^c) \mapsto \mathbb{R}^{C-c}$. We set $c = C/2$ in all experiments. For $\mathbf{f}$ it is not strictly necessary to split $\mathbf{x}_\text{A}$ to ensure bijectiveness. However, by doing so we highly reduce the computational requirements.

The inverse $\mathbf{f}$ of the conditional coupling layer is:

$$\begin{cases} x_\text{B}^{1:c} &= y_\text{B}^{1:c} \\ x_\text{B}^{c+1:C} &= \left(y_\text{B}^{c+1:C} - t\left(y_\text{A}^{1:c}, y_\text{B}^{1:c}\right)\right) \odot \exp\left(s\left(x_\text{A}^{1:c}, x_\text{B}^{1:c}\right)\right), \end{cases} \tag{12}$$

and its Jacobian:

$$\frac{\partial y_\text{B}}{\partial x^\top} = \begin{bmatrix} \mathbf{I}_c & \mathbf{0} \\ \frac{\partial y_\text{B}^{c+1:C}}{\partial (x^{1:c})^\top} & \text{diag}\left(\exp\left(s\left(x_\text{A}^{1:c}, x_\text{B}^{1:c}\right)\right)\right) \end{bmatrix},$$

where $\mathbf{I}_c \in \mathbb{R}^{c \times c}$ is an identy matrix. Since the Jacobian is a triangular matrix, its determinant can be calculated efficiently as the product of the diagonal elements. Note that it is not required to compute the Jacobian of the functions s and t, enabling them to be arbitrarily complex. In practice, we implement these functions using a convolutional neural network $\Psi(\cdot)$ that returns both $\log(\mathbf{s})$ and $\mathbf{t}$.

4.2. Coupling Network Architecture

We next describe the architecture of $\Psi_{\boldsymbol{\theta}}(\cdot)$ and $\Psi_{\boldsymbol{\phi}}(\cdot)$ used to regress the affine transform applied at every conditional coupling layer at each $\mathbf{g}_i$ and $\mathbf{f}_i$ respectively. We build upon the stack of three 2D convolution layers proposed by [25]. The first two layers have a filter size of 3×3 and 1×1 with 512 output channels followed by actnorm [25] and a ReLU activation. The third layer regresses the final scale and translation by applying a 2D convolutional layer with filter size 3×3 initialized with zeros such that each affine transformation at the beginning of training is equivalent to an identity function.

For the transformation $\mathbf{g}_i^{-1}(x_\text{A})$ we exactly use this architecture, but for $\mathbf{f}_i^{-1}(x_\text{B}|x_\text{A})$ we extend it to take into account the conditioning x_A. Concretely, in $\mathbf{f}_i^{-1}$, x_B is initially transformed by two convolution layers, like the first two of $\mathbf{g}_i^{-1}$. Then, x_A is adapted with a channel-wise affine transform implemented by a 1×1 convolution. Finally, its output is added to the transformed x_B. To ensure a similar contribution of x_A and x_B their activations are normalized with actnorm so that they operate in the same range. A final 3×3 convolution regresses the conditional coupling layer operators $\log(\mathbf{s}_\text{B})$ and $\mathbf{t}_\text{B}$.

4.3. Invertible Cycle Consistency

We train our model to maximize the log-likelihood of the training dataset. However, likewise in GANs learning [34, 22], we found beneficial to add a loss encouraging the generated and real samples to be similar in L1. To do so, we exploit the fact that our model is made of bijective transformations, and introduce what we call an invertible cycle consistency. This operation can be summarized as follows:

$$\{\mathbf{x}_\text{A}, \mathbf{x}_\text{B}\} \xrightarrow{\mathbf{g}^{-1}, \mathbf{f}^{-1}} \{\mathbf{z}_\text{A}, \mathbf{z}_\text{B}\} \rightarrow \{\mathbf{z}_\text{A}, \hat{\mathbf{z}}_\text{B}\} \xrightarrow{\mathbf{f}} \hat{\mathbf{x}}_\text{B}. \tag{13}$$

Concretely, the data points observations ($\mathbf{x}_\text{A}$, $\mathbf{x}_\text{B}$) are initially mapped into their latent variables ($\mathbf{z}_\text{A}$, $\mathbf{z}_\text{B}$), where each variable is composed of an L-level stack. As demonstrated in [14] the first levels encode the high frequencies (details) in the data, and the last levels the low frequencies.

We then resample the first $L-1$ dimensions of $\mathbf{z}_\text{B}$ from a Gaussian distribution, *i.e.* $\mathbf{z}_\text{B} = [\mathbf{z}_1, \ldots, \mathbf{z}_L] \rightarrow \hat{\mathbf{z}}_\text{B} = [\mathbf{N}(0, \mathbf{I})_1, \ldots, \mathbf{N}(0, \mathbf{I})_{L-1}, \mathbf{z}_L]$. By doing this, $\hat{\mathbf{z}}_\text{B}$ is only retaining the lowest frequencies of the original $\mathbf{z}_\text{B}$.

As a final step, we invert $\mathbf{f}^{-1}$, to recover $\hat{\mathbf{x}}_\text{B} = \mathbf{f}(\hat{\mathbf{z}}_\text{B}|\mathbf{z}_\text{A})$ and penalize its L1 difference w.r.t the original $\mathbf{x}_\text{B}$. What we are essentially doing is to force the model to use information from the condition $\mathbf{x}_\text{A}$ so that the recover sample $\hat{\mathbf{x}}_\text{B}$ is as similar as possible to the original $\mathbf{x}_\text{B}$. Note that if reconstructed $\hat{\mathbf{z}}_\text{B}$ based on the entire latent variable, the recovered sample would be identical to the original $\mathbf{x}_\text{B}$ because $\mathbf{f}$ is bijective, and this loss would be meaningless.

4.4. Total Loss

Formally, denoting the training pairs of observations as $\{\mathbf{x}_\text{A}^{(i)}, \mathbf{x}_\text{B}^{(i)}\}_{i=1}^N$, the model parameters are learned by minimizing the following loss function:

$$\frac{1}{N}\sum_{i=1}^{N}\left[-\log p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_\text{A}^{(i)}, \mathbf{x}_\text{B}^{(i)}) + \lambda \left\| \mathbf{x}_\text{B}^{(i)} - \hat{\mathbf{x}}_\text{B}^{(i)} \right\|_1\right] \tag{14}$$

The first term maximizes the joint likelihood of the data observations. With our design, it also maximizes the conditional likelihood of $\mathbf{x}_\text{B}|\mathbf{x}_\text{A}$ and thus forces the model to

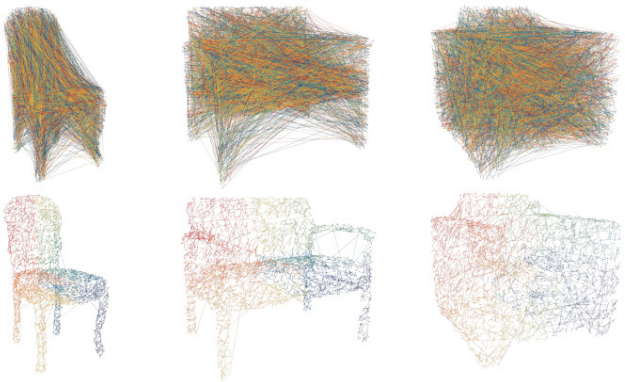


Figure 4: **Sorting 3D point clouds.** Point clouds corresponding to three different chairs. The colored line connects all points based on their ordering. **Top:** Unordered. **Bottom:** Applying the proposed sorting strategy. Note how the coloring is consistent across samples even for point clouds with different topology.

learn the desired mapping. To show this, we apply the law of total probability and we factor it into:

$$-\sum_{i=1}^{N}\log p_{\boldsymbol{\theta}}(\mathbf{x}_{\mathrm{A}}^{(i)})-\sum_{i=1}^{N}\log p_{\boldsymbol{\phi}}(\mathbf{x}_{\mathrm{B}}^{(i)}|\mathbf{x}_{\mathrm{A}}^{(i)}) \tag{15}$$

Due to the diagonal structure of the Jacobians, the marginal likelihood of $\mathbf{x}_{\mathrm{A}}$ depends only on $\boldsymbol{\theta}$ (first sum), while the conditional of $\mathbf{x}_{\mathrm{B}}|\mathbf{x}_{\mathrm{A}}$, only on $\boldsymbol{\phi}$. Maximizing the joint likelihood thus maximizes both likelihoods independently.

The second term in (14) minimizes the cycle consistency loss. λ is a hyper-parameter balancing the terms. This loss is fully differentiable, and we provide details on how we optimize it in Sec. 6.

5. Modeling Unordered 3D Point Clouds

The model described so far can handle input data represented on regular grids but it fails to model unordered 3D point clouds, whose lack of spatial neighborhood ordering prevents convolutions from being applied. To process point clouds with deep networks, a common practice is to apply *symmetry operations* [40] that create fixed-size tensors of global features describing the entire point cloud sample. These operations require extracting point-independent features followed by a max-pool, which is not invertible and not applicable to normalizing flows. Another alternative would be the graph convolutional networks [55], although their high computational cost makes them not suitable for our scheme of multiple coupling layers. We propose a three-step mechanism to enable modeling 3D point clouds:

(i) Approximate Sorting with Space-Filling Curves. C-Flow is based on convolutional layers which require input data with a local neigboorhood consistent across samples. To fulfill this condition on unordered point clouds, we propose to sort them based on proximity. As discussed in [40], for high dimensional spaces it is not possible to produce a perfect ordering stable to point perturbations. In this paper we therefore consider using the approximation provided by the Hilbert's space-filling curve algorithm [21]. For each training sample, we project its points into a 3D Hilbert curve and reorder them based on their ordering along the curve (Fig. 4). Notice that not only we can establish a neighborhood relationship but also a semantically-stable ordering (*e.g.* in Fig. 4 the chair's right-leg is always blue). To the best of our knowledge there is no previous work using such preprocessing for point clouds.

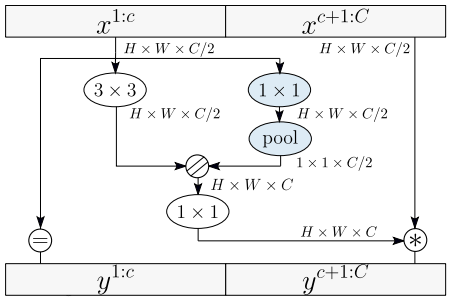


Figure 5: **Approximating global features in point clouds.** When dealing with point clouds (reordered and reshaped to a $H \times W \times 3$ size and using $c = C/2$) we approximate, with operations in blue, global features in coupling layers while still being invertible. $\circledast$ stands for affine transformation where the first $C/2$ input channels are the scale and the other half the translation.

(ii) Approximating Global Features. Hilbert Sort is not sufficient to model 3D data because of a major issue: it splits the space into equally sized quadrants and the Hilbert curve will cover all points in a quadrant before moving to the next. As a consequence, two points that were originally close in space, but lie near the boundaries of two different quadrants, will end up far away in the final ordering. To mitigate this effect we extend the proposed coupling network architecture (Sec. 4.2) with an approximate but invertible version of the global features proposed in [40] that describe the whole point cloud. Concretely, we first resample and reshape the reordered point cloud to form $H \times W \times 3$ matrices (in practice we use the same size as that of the images). Then we approximate the global descriptors of [40] through a 1×1 convolution to extract point-independent features followed by a max-pool applied only over the first half of the point cloud features $x^{1:c}$ (Fig. 5). The coupling layer remains bijective because during the backward propagation the approximated global features can be recovered using a similar strategy as in Eq. (12).

(iii) Symmetric Chamfer Distance for Cycle Consistency. For the specific case of point clouds, we observed that when penalizing the invertible cycle consistency with L1 the model converged to a mean Hilbert curve. Therefore, for point clouds, we substitute L1 by the symmetric Chamfer distance , which computes the mean Euclidean distance between the ground-truth point cloud $\mathbf{x}_{\mathrm{B}}$ and the recovered $\hat{\mathbf{x}}_{\mathrm{B}}$.

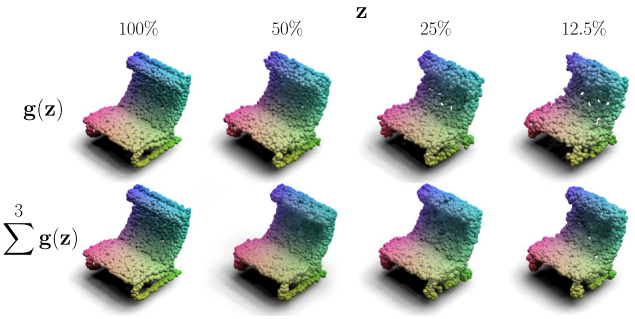


Figure 6: **Embedding 3D points clouds. Top:** Reconstruction with partial embeddings. **Bottom:** Reconstruction with three iterations of backward propagations of partial embeddings.

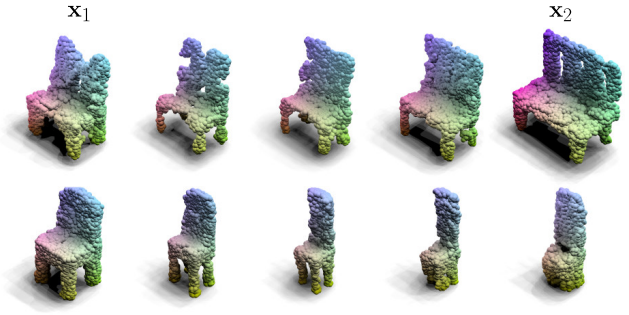


Figure 7: **Interpolation.** Results of interpolating two 3D point clouds $\mathbf{x}_1$ and $\mathbf{x}_2$ in the learned latent space.

Method	100%	50%	25%	12.5%
C-Flow* ≡ Glow [25]	0.00	0.39	0.39	0.39
C-Flow* + Sort	0.00	0.19	0.21	**0.22**
C-Flow* + Sort + GF-Coupling	**0.00**	**0.14**	**0.18**	0.31
AtlasNet-Sph. [18]		0.75		
AtlasNet-25 [18]		0.37		
DeepSDF [32]		0.20		

Table 1: **Representing 3D point clouds**. Chamfer distance when recovering point clouds with partial embeddings. For all C-Flow* we change the embedding size at test, with no further training. The percentages are with respect to the input dimension (4096). For AtlasNet and DeepSDF we provide the results from [32].

6. Implementation Details

Due to memory restrictions, we train with image samples of 64×64 resolution. For 3D point clouds, to maintain the same architecture as in images, we reshape each point cloud sample (list of 64^2 points) to 64×64. At test time we also regress 64^2 3D points per forward pass. Our implementation builds upon that of Glow [25]. We use Adam with learning rate $1e^{-6}$, $\beta_1 = 0.85$, $\beta_2 = 0.007$ and batch size 4. The multi-scale architecture consists of $L = 4$ levels with 12 flow steps per level ($K = 4 * 12$ in Eq. (3)) each and $2\times$ squeezing operations. For conditional sampling we found additive coupling ($s(\cdot) = 1$) to be more stable during training than affine transformation. The prior distributions $p_{\boldsymbol{\theta}}(\mathbf{z}_{\mathrm{A}})$ and $p_{\boldsymbol{\varphi}}(\mathbf{z}_{\mathrm{B}})$ are initialized with mean 0 and variance 1. The rest of weights are randomly initilized from a normal distribution with mean 0 and std 0.05. $\lambda = 10$ in Eq. (14). As in previous likelihood-based generative models [33, 25], we observed that sampling from a reduced-temperature prior improves the results. To do so, we multiply the variance of $p_{\boldsymbol{\varphi}}(\mathbf{z}_{\mathrm{B}})$ by $T = 0.9$. The model is trained with 4 GPUs P-100 for 10 days.

7. Experimental Evaluation

We next evaluate our system on diverse tasks: (1) Modeling point clouds (Sec. 7.1), (2) 3D reconstruction and rendering (Sec. 7.2), (3) Image-to-image mapping in a variety of domains and datasets (Sec. 7.3), and (4) Image manipulation and style transfer (Sec. 7.4).

7.1. Modeling 3D Point Clouds

We evaluate the potential of our approach to model 3D point clouds on ShapeNet [7]. For this task, we do not consider the full conditioning scheme and only use one of the branches of C-Flow in Fig. 2, which we denote as C-Flow*.

In our first experiment we study the representation capacity of unknown shapes, formally defined as the ability to retain the information after mapping forward and backward between the original and latent spaces. For this purpose, we first map a real point cloud $\mathbf{x}$ to the latent space $\mathbf{z} = \mathbf{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{x})$. The full-size embedding $\mathbf{z} = [\mathbf{z}_1, \ldots, \mathbf{z}_L]$ has as many dimensions as the input (HWC). Then we progressively remove information from $\mathbf{z}$ by replacing their left-most l components with samples drawn from a Gaussian distribution, *i.e.* $\hat{\mathbf{z}} = [\mathbf{N}(0, \mathbf{I})_1, \ldots, \mathbf{N}(0, \mathbf{I})_l, \mathbf{z}_{l+1}, \ldots, \mathbf{z}_L]$. Note that the embedding size $L - l$ can be set at test time with no need to retrain, making tasks like point cloud compression straightforward. Finally we map back this embedding to the original point cloud space $\hat{\mathbf{x}} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})$ and compare to $\mathbf{x}$.

Tab. 1 reports the Chamfer Distance (CD) for different embedding sizes. CD is computed by densely resampling the input mesh with up to 10^7 vertices. The plain version of C-Flow* (no conditioning, no sorting, no global features) is equivalent to Glow [25]. This version is consistently improved when introducing the sorting and global features strategies (Sec. 5). The error decreases gracefully as we increase the embedding size, and importantly, when using the full size embedding we obtain a perfect recovering (Fig. 6-top). This is a virtue of the bijective models, and is not a trivial property. Tab. 1 also reports the numbers of AtlasNet [18] and DeepSDF [32], showing that our approach achieves competitive results. This comparison is only indicative as the representation used are inherently different ([18] parametric and [32] continuous surface).

Recall that the left-most components in $\mathbf{z}$ encode the shape high details. We exploit this property to generate point clouds with an arbitrarily large number of points by performing multiple backward propagations ($\hat{\mathbf{x}} = \mathbf{g}_{\theta}(\hat{\mathbf{z}})$) of a partial embedding $\hat{\mathbf{z}}$ (Fig. 6-bottom). Every time we propagate, we recover a new set of 3D points allowing to progressively improve the density of the reconstruction.

Another task that can be addressed with C-Flow is shape interpolation in the latent space (Figure 7).

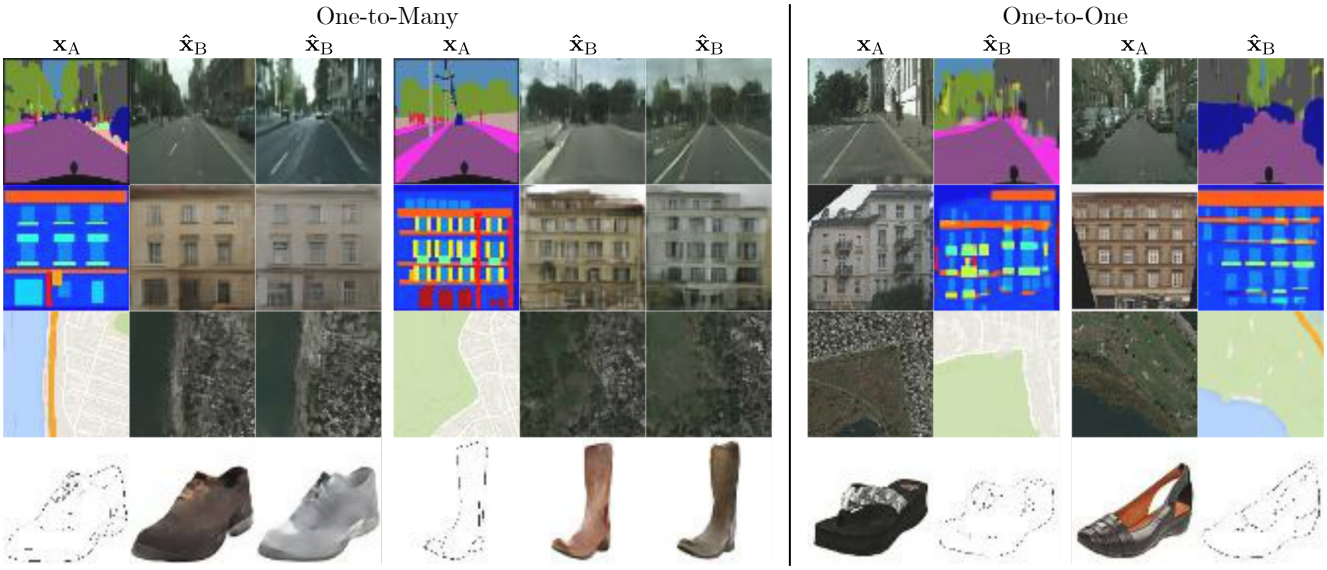


Figure 8: **Image-to-Image.** Results from 64×64 image-to-image mappings on a variety of domains. $\mathbf{x}_A$: source image; $\hat{\mathbf{x}}_B$: generated image in the target domain. The examples on the left correspond to target domains with high variability that when sampled multiple times generate different images. In the examples on the right the target domain has a small variability and the sampling becomes deterministic.

	Image → PC	Image ← PC	
Method	CD↓	BPD↓	IS↑
3D-R2N2 [9]	0.27	-	-
PSGN [15]	0.26	-	-
Pix2Mesh [51]	0.27	-	-
AtlasNet [18]	**0.21**	-	-
ONet [30]	0.23	-	-
C-Flow	0.86	4.38	1.80
C-Flow + Sort	0.52	**2.77**	2.41
C-Flow + Sort + GF-Coupling	0.49	2.87	**2.61**
C-Flow + Sort + GF-Coupling + CD	**0.26**	-	-

Table 2: **3D Reconstruction and rendering.** ↓: the lower the better, ↑: the higher the better. C-Flow is the first approach able to render images from point clouds. The same model can be used to perform 3D reconstruction from images. The results of all other methods are obtained from their original papers.

	C-Flow			C-Flow + cycle		
Method	BPD↓	SSIM↑	IS↑	BPD↓	SSIM↑	IS↑
segmentation → street views	3.21	0.37	1.80	**3.17**	**0.42**	**1.94**
segmentation ← street views	3.25	0.33	2.19	**3.05**	**0.36**	**2.23**
structure → facades	3.55	0.24	**1.92**	**3.54**	**0.26**	1.69
structure ← facades	3.55	**0.31**	**2.05**	3.55	0.30	2.01
map → aerial photo	3.65	**0.19**	1.52	3.65	0.17	**1.62**
map ← aerial photo	3.65	0.54	1.95	3.65	**0.57**	**1.97**
edges → shoes	1.70	0.66	2.40	**1.68**	**0.67**	**2.43**
edges ← shoes	1.65	0.64	1.61	1.65	**0.65**	**1.69**

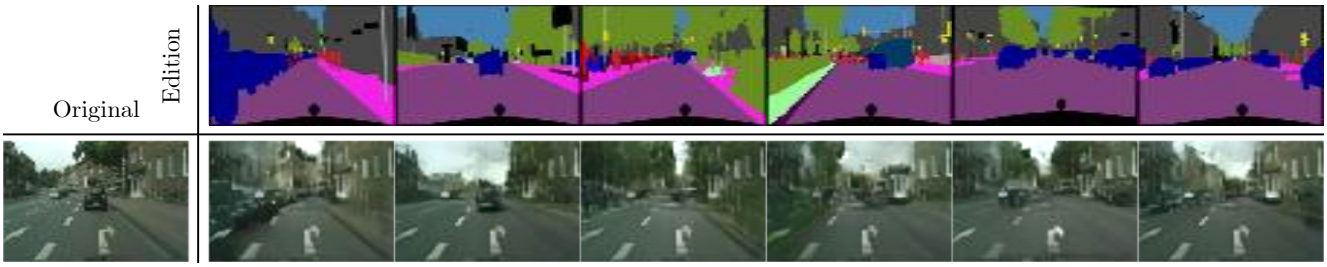
Table 3: **Conditional image-to-image generation.** Evaluation of C-Flow (plain) and C-Flow + cycle consistency loss in image-to-image mapping.

7.2. 3D Reconstruction & rendering

We next evaluate the ability of C-Flow to model the conditional distributions (1) *image* → *point cloud* , which enables to perform 3D reconstruction from a single image; and (2) *point cloud* → *image*, which is its inverse problem of rendering an image given a 3D point cloud. Fig. 1 shows qualitative results on the *Chair* class of ShapeNet. In the top row our model is able to generate plausible 3D reconstructions of unknown objects even under strong self-occlusions (top-right example). The second row depicts results for rendering, which highlights another advantage of our model: it allows sampling multiple times from the conditional distribution to produce several images of the same object which exhibit different properties (*e.g.* viewpoint or texture).

In Table 2 we compare C-Flow with other single-image 3D reconstruction methods 3D-R2N2 [9], PSGN [15], Pix2Mesh [51], AtlasNet [18] and ONet [30]. We evaluate 3D reconstruction in terms of the Chamfer distance (CD) with the ground truth shapes. Our approach (last row) performs on par with [9, 15, 51] and it is slightly below the state-of-the-art techniques specifically designed for 3D reconstruction [18, 30].

With the same model, we can also render images from point clouds. To the best of our knowledge, no previous work can perform such mapping. While a few approaches do render point clouds [31, 2, 35], they hold on strong assumptions of knowing the RGB color per point and the camera calibration to project the point cloud onto the image plane. Table 2 also reports an ablation study about the different operations we devised to handle 3D point clouds, namely sorting the point cloud (Sort), approximating global features (GF-Coupling) and inverse cycle consistency with chamfer distance (CD). In this case, evaluation is reported using Inception Score (IS) [44] and Bits Per Dimension (BPD) which is equivalent to the negative log2-likelihood typically used to report flow-based methods performance. Results show a performance boost when using each of these components, and especially when combining them.



(a) Image content manipulation

(b) Style transfer

Figure 9: **Other applications.** Sample results on 64×64 image manipulation and style transfer. The model was not retrained for these tasks, and we used the same training weights to perform image-to-image in Fig. 8.

7.3. Image-to-Image mappings

We evaluate the ability of C-Flow to perform multi-domain image-to-image mapping: *segmentation* $\leftrightarrow$ *street views* trained on Cityscapes [10], *structure* $\leftrightarrow$ *facade* trained on CMP Facades [50], *map* $\leftrightarrow$ *aerial photo* trained on [22] and *edges* $\leftrightarrow$ *shoes* trained on [59, 60, 22]. Fig. 8-left shows mappings in which the target domain has a wide variance and multiple sampling generates different results (*e.g.* a semantic segmentation map can map to several grayscale images). Fig. 8-right examples have a target domain with a narrower variance, and despite multiple samplings the generated images are very similar (*e.g.* given an image its segmentation is well defined).

Table 3 reports quantitative evaluations using Structural Similarity (SSIM) [53], and again BPD and IS. When introducing the invertible cycle consistency loss (Sec. 4.3) the model does not improve its compression abilities (BPD) but improves in terms of structural similarity (SSIM) and semantic content (IS). It is worth mentioning that while GANs have shown impressive image-to-image mapping results, even at high resolution [52], ours is the first work that can address such tasks using normalizing flows.

7.4. Other Applications

Finally, we demonstrate the versatility of C-Flow being the first flow-based method capable of performing style transfer and image content manipulation (Fig. 9). Importantly, the model was *not retrained* for these specific tasks, and we use the same parameters learned to perform image-to-image mappings (Sec. 7.3). For image manipulation we use the weights of *segmentation* $\rightarrow$ *street view* and for style transfer those of *edges* $\leftrightarrow$ *shoes*. Formally, let the domain A to be the structure (*e.g.* segmentation mask) and the domain B to be the image (*e.g.* street view). Then, image manipulation is achieved via three operations:

$$\mathbf{z}_{\mathrm{B}}^{1} = \mathbf{f}_{\phi}^{-1}(\mathbf{x}_{\mathrm{B}}^{1}|\mathbf{x}_{\mathrm{A}}^{1}) \quad \text{encode original image } \mathbf{x}_{\mathrm{B}}^{1} \tag{16}$$

$$\mathbf{z}_{\mathrm{A}}^{2} = \mathbf{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{x}_{\mathrm{A}}^{2}) \quad \text{encode desired structure } \mathbf{x}_{\mathrm{A}}^{2} \tag{17}$$

$$\mathbf{x}_{\mathrm{B}}^{2} = \mathbf{f}_{\phi}(\mathbf{z}_{\mathrm{B}}^{1}|\mathbf{z}_{\mathrm{A}}^{2}) \quad \text{synthesise new image } \mathbf{x}_{\mathrm{B}}^{2} \tag{18}$$

Note that we are no longer conditioning based only on A, as in Sec. 7.3, now the synthesised image is jointly conditioned on A (for structure) and B (for texture).

To perform style transfer, we first transform the content image into its structure $\mathbf{x}_{\mathrm{A}}^{2}$. For instance, in Fig. 9-bottom, the content of the *shoe* is initially mapped onto its *edge* structure with the *shoes* $\rightarrow$ *edges* weights. Then, we apply the same procedure as we did for image manipulation using the *edges* $\rightarrow$ *shoes* weights, setting $\mathbf{x}_{\mathrm{A}}^{1}$ to be the structure of the content image and $\mathbf{x}_{\mathrm{B}}^{1}$ the style image.

8. Conclusions

We have proposed C-Flow, a novel conditioning scheme for normalizing flows. This conditioning, in conjunction with a new strategy to model unordered 3D point clouds, has made it possible to address 3D reconstruction and rendering images from point clouds, problems which so far, could not be tackled with normalizing flows. Furthermore, we demonstrate C-Flow to be a general-purpose model, being also applicable to many more multi-modality problems, such as image-to-image translation, style transfer and image content edition. To the best of our knowledge, no previous model has demonstrated such an adaptability.

Acknowledgements

This project was done during an internship at Google. It is also partially supported by the EU project TERRINET: The European robotics research infrastructure network H2020-INFRAIA-2017-1-730994.

References

[1] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Generative adversarial networks for extreme learned image compression. In *ICCV*, 2019. 2

[2] Kara-Ali Aliev, Dmitry Ulyanov, and Victor S. Lempitsky. Neural point-based graphics. *arXiv preprint arXiv:1906.08240*, 2019. 7

[3] Timur Bagautdinov, Chenglei Wu, Jason Saragih, Pascal Fua, and Yaser Sheikh. Modeling facial geometry using compositional vaes. In *CVPR*, 2018. 2

[4] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. In *SIGGRAPH Asia*, 2018. 2

[5] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *PMLR*, 2017. 2

[6] Sergi Caelles, Albert Pumarola, Francesc Moreno-Noguer, Alberto Sanfeliu, and Luc Van Gool. Fast video object segmentation with spatio-temporal gans. *arXiv preprint arXiv:1903.12161*, 2019. 2

[7] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 6

[8] Hung-Jen Chen, Ka-Ming Hui, Szu-Yu Wang, Li-Wu Tsao, Hong-Han Shuai, and Wen-Huang Cheng. Beautyglow: On-demand makeup transfer framework with reversible generative network. In *CVPR*, 2019. 2

[9] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016. 2, 7

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 8

[11] Enric Corona, Albert Pumarola, Guillem Alenyà, Francesc Moreno-Noguer, and Gregory Rogez. Ganhand: Predicting human grasp affordances in multi-object scenes. In *CVPR*, 2020. 2

[12] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *CVPR*, 2017. 2

[13] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. In *ICLR*, 2014. 1, 2, 3

[14] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *ICLR*, 2017. 2, 3, 4

[15] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017. 2, 7

[16] Jicong Fan and Jieyu Cheng. Matrix completion by deep matrix factorization. *Neural Networks*, 2018. 2

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 1, 2

[18] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. *CVPR*, 2018. 2, 6, 7

[19] Aditya Grover, Christopher D. Chute, Rui Shu, Zhangjie Cao, and Stefano Ermon. Alignflow: Cycle consistent learning from multiple domains via normalizing flows. *arXiv preprint arXiv:1905.12892*, 2019. 2

[20] Paul Henderson and Vittorio Ferrari. Learning single-image 3D reconstruction by generative modelling of shape, pose and shading. *IJCV*, 2019. 2

[21] David Hilbert. *Über die stetige Abbildung einer Linie auf ein Flächenstück*. 1935. 5

[22] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016. 4, 8

[23] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 2

[24] Sungwon Kim, Sang-gil Lee, Jongyoon Song, and Sungroh Yoon. Flowavenet: A generative flow for raw audio. *ICML*, 2018. 2

[25] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018. 2, 3, 4, 6

[26] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 1, 2

[27] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*, 2019. 2

[28] Christoph Lassner, Gerard Pons-Moll, and Peter V. Gehler. A generative model of people in clothing. In *ICCV*, 2017. 2

[29] Rui Liu, Yu Liu, Xinyu Gong, Xiaogang Wang, and Hongsheng Li. Conditional adversarial generative flow for controllable image synthesis. In *CVPR*, 2019. 2

[30] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 2, 7

[31] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *CVPR*, 2019. 7

[32] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CVPR*, 2019. 2, 6

[33] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018. 6

[34] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 4

[35] Francesco Pittaluga, Sanjeev J Koppal, Sing Bing Kang, and Sudipta N Sinha. Revealing scenes by inverting structure from motion reconstructions. In *CVPR*, 2019. 7

[36] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP*, 2018. 2

[37] Albert Pumarola, Antonio Agudo, Aleix M. Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: One-shot anatomically consistent facial animation. *IJCV*, 2019. 2

[38] Albert Pumarola, Antonio Agudo, Lorenzo Porzi, Alberto Sanfeliu, Vincent Lepetit, and Francesc Moreno-Noguer. Geometry-aware network for non-rigid shape prediction from a single view. In *CVPR*, 2018. 2

[39] Albert Pumarola, Jordi Sanchez, Gary Choi, Alberto Sanfeliu, and Francesc Moreno-Noguer. 3DPeople: Modeling the Geometry of Dressed Humans. In *ICCV*, 2019. 2

[40] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 5

[41] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, 2015. 2

[42] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019. 2

[43] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015. 2

[44] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016. 7

[45] J. Serrà, S. Pascual, and C. Segura. Blow: a single-scale hyperconditioned flow for non-parallel raw-audio voice conversion. *NeurIPS*, 2019. 2

[46] Ayan Sinha, Asim Unmesh, Qixing Huang, and Karthik Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. In *CVPR*, 2017. 2

[47] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NeurIPS*, 2015. 2

[48] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In *CVPR*, 2018. 2

[49] Haoliang Sun, Ronak Mehta, Hao Zhou, Zhichun Huang, Sterling Johnson, Vivek Prabhakaran, and Vikas Singh. Dual-glow: Conditional flow-based generative model for modality transfer. *arXiv preprint arXiv:1908.08074*, 2019. 2

[50] Radim Tyleček and Radim Šára. Spatial pattern templates for recognition of objects with regular structure. In *GCPR*, 2013. 8

[51] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018. 2, 7

[52] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 8

[53] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004. 8

[54] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NeurIPS*, 2016. 2

[55] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019. 5

[56] Masataka Yamaguchi, Yuma Koizumi, and Noboru Harada. Adaflow: Domain-adaptive density estimator with application to anomaly detection and unpaired cross-domain translation. In *ICASSP*, 2019. 2

[57] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. *ICCV*, 2019. 2

[58] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018. 2

[59] Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014. 8

[60] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016. 8

[61] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2