Single View Point Cloud Generation via Unified 3D Prototype

Yu Lin, Yigong Wang, Yi-fan Li, Zhuoyi Wang, Yang Gao, Latifur Khan
The University of Texas at Dallas, USA
yxl163430, yxw158830, yli, zxw151030, yxg122530, lkhan@utdallas.edu

Abstract

As 3D point clouds become the representation of choice for multiple vision and graphics applications, such as autonomous driving, robotics, etc., the generation of them by deep neural networks has attracted increasing attention in the research community. Despite the recent success of deep learning models in classification and segmentation, synthesizing point clouds remains challenging, especially from a single image. State-of-the-art (SOTA) approaches can generate a point cloud from a hidden vector, however, they treat 2D and 3D features equally and disregard the rich shape information within the 3D data. In this paper, we address this problem by integrating image features with 3D prototype features. Specifically, we propose to learn a set of 3D prototype features from a real point cloud dataset and dynamically adjust them through the training. These prototypes are then integrated with incoming image features to guide the point cloud generation process. Experimental results show that our proposed method outperforms SOTA methods on single image based 3D reconstruction tasks.

Introduction

Fueled by recent development in 3D acquisitions, 3D sensors are becoming increasingly available in various applications, such as LiDARs and RGB-D cameras. In contrast to a simple RGB image, 3D data offers much richer geometric, shape, and scale information, which is proven advantageous for numerous applications. Among the 3D data representations, point clouds are becoming more and more popular as they can capture a much higher resolution than the voxel grids, and show more sophisticated representations such as meshes. It would be extremely valuable if we can construct a point cloud merely from one single image. For instance, since 3D sensors are still much more extravagant comparing to traditional cameras, generating point clouds from RGB images can significantly reduce the data acquisition budget. Also, self-driving may benefit from this technique since complete 3D shapes can provide more information regarding surroundings of the object.

Numerous methods have been proposed to solve this 3D representation synthesis task. For example, *3D-R2N2* (Choy et al. 2016) utilizes a 3D LSTM to generate the voxel representation, given one or multiple views of an object. *HoloGAN* (Nguyen-Phuoc et al. 2019) separates the shape and appearance of an object, then generate its 2D projection from an unseen angle. Despite such great success, these approaches mainly focus on the regularly sampled data, which limits their usage in practice.

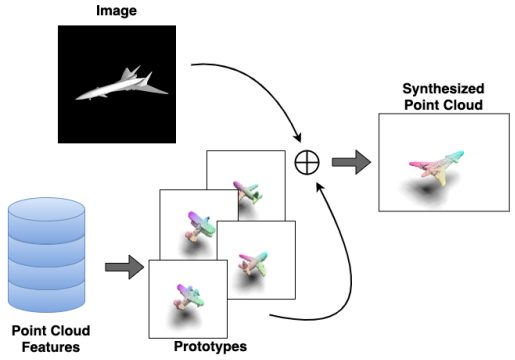

Figure 1: The intuition of our framework: Combining 2D image feature and 3D prototype knowledges.

As a flexible and powerful representation, the point cloud receives more and more attention from the research community. For instance, *PointSetNet* (Fan, Su, and Guibas 2017) generates a point cloud from one image, with the help from a two-branch network architecture and MoN loss function. *FoldingNet* (Yang et al. 2018) is proposed to reconstruct a point set by deforming a canonical 2D grid onto the underlying 3D object surface. Furthermore, *AtlasNet* (Groueix et al. 2018) takes one more step in this direction, which deforms multiple 2D configurations onto the target surface. Although significant progresses has been made, previous methods either focus on the self-reconstruction task tremendously, or overlook the handy shape information concealed in the point clouds themselves.

Motivated by these aforementioned limitations and recent advances of the 3D shape representation learning, we believe that model performance could be significantly improved using the strong shape prior introduced by a point cloud dataset.

In this paper, we focus on designing a novel architec-

ture that allows efficient image-to-point-cloud generation, and connects the gap between 2D and 3D features. To address those problems, we introduce our **T**hree **D**imensional **P**rototype **Net**work (**TDPNet**), which is a unified point cloud generation framework based on 3D prototypes. To be more concrete, we first extract a set of features from point cloud datasets via a SOTA representation learning framework, such as *PointNet* (Qi et al. 2017a) and *PointwiseCNN* (Hua, Tran, and Yeung 2018). Then, we obtain a set of 3D prototype features with a clustering algorithm (*e.g. KMeans*). Note that these prototypes can be considered as 3D shape priors to our proposed framework. For a given image, we build a set of fused vectors based on 2D features, 3D features, and random 2D grids. The 2-manifold is deformed onto a point cloud patch using a simple MLP. Our framework contains multiple MLP decoders to handle different prototype features. This setting encourages diverse reconstruction and avoids mode collapse. All the point patches are collected into the final point cloud. Compared to previous research such as *CSRNet* (Kar et al. 2015) and *RealPoint3D* (Zhang et al. 2019), which restrict their model template on a specific domain, our approach is more flexible and efficient.

To sum up, our contributions in this paper are listed as follows:

- A deep learning based framework is proposed to solve the image-to-point-cloud generation problem;
- We bridge the gap between 2D image features and 3D shape features in the hidden space;
- A unified 3D prototypes schema is designed to efficiently utilize the rich structural 3D information;
- We conduct extensive experiments to verify the effectiveness of our method, both quantitatively and qualitatively;

Related Works

Multi-view Reconstruction

Challenges about 3D reconstruction has been well studied based on the multi-view geometry (MVG) in the previous literatures (Hartley and Zisserman 2003). Traditional approaches include motion structures (**SfM**) (Schonberger and Frahm 2016), simultaneous localization and mapping (**SLAM**) (Cadena et al. 2016). Even though these methods have achieved promising performances, they are limited by the coverage multiple views images (Wang et al. 2018; Zhu et al. 2020). Recently, a number of SOTA frameworks leverage deep neural networks to learn a 3D shape from multiple images (Gadelha, Maji, and Wang 2017; Rezende et al. 2016; Tulsiani et al. 2017; Choy et al. 2016). Most of these approaches require no 3D ground truth (GT) labels for supervision, while additional signals (*e.g.*, contextual information or camera matrix) are required. In practice, these requirements could limit their usages dramatically. Moreover, it is favorable if we can reconstruct the complete 3D shape from only one RGB image (single view reconstruction).

Single-view 3D Reconstruction

Predicting a complete 3D shape via only one image is a long-standing conundrum. Furthermore, this problem is ill-posed, and prior knowledge is mandatory because an RGB image solely contains deficient information for a complex 3D model (Fan, Su, and Guibas 2017).

With the rapid advance of deep learning techniques, lots of deep learning frameworks explored the land of 3D reconstruction. *3D-ShapeNet* (Chang et al. 2015) is amongst the first framework utilizing deep networks to predict multiple 3D solutions from a single partial view. *3D-EPN* (Dai, Ruizhongtai Qi, and Nießner 2017) firstly predicted a 32^3 voxel grids and synthesized a higher resolution model with deep networks. The resolution of the voxel representation gradually increased from 32^3 to 60^3 (Gwak et al. 2017; Yang et al. 2017). Recently, an interesting GAN framework, so-called *HoloGAN* (Nguyen-Phuoc et al. 2019), is proposed to learn the unseen 2D projections of a 3D shape based on a sculpted 32^3 voxel volume. Even great progress has made, the aforementioned methods are restricted to the voxel representation and suffered from scalable issues.

On the other hand, 3D objects tend to be more complete and natural by using point clouds and such representation forms a nicer shape for neural networks. In recent years, *PointSetNet* (Fan, Su, and Guibas 2017) combined an MoN loss and a powerful two-branch architecture to restore point clouds from one image. *PCGAN* (Li et al. 2018) theoretically solved the applicable problem of GAN on the point cloud. *AtlasNet* (Groueix et al. 2018) trained a set of manifold decoders that are applicable to both point cloud self-recontruction and image-to-point-cloud synthesization. A contemporary method, so-called *SSPNet* (Navaneet et al. 2020), generates point cloud from one image by enforcing geometric and pose cycle consistency. Nevertheless, it has a strong assumption that each image has its corresponding sihouette (image mask). We observed that those frameworks treat 2D images and 3D shapes equally. Consider the fact that the valuable information embedded in 3D shapes compensate for the missing part of 2D images, mixing 2D and 3D features seem to be more reasonable.

Deep Learning in Point Clouds

As a preferred representation for many CV applications, many deep learning approaches emerged to analyze the property of point clouds. Shared MLP and convolutional-based networks are two main categories in this domain. *PointNet* (Qi et al. 2017a) is the pioneer of the first schema, which learns shape descriptors from point clouds via a sharing function. Successors follow the same philosophy achieved amazing performance in different applications (Qi et al. 2017b; Yang et al. 2019b; Wang et al. 2020).

Shared MLP methods generally ignored the local information around each point. Convolutional-based networks are introduced to tackle this issue. Specifically, they learned a convolutional kernel covers a small region and use it to aggregate information from 3D shapes (Hua, Tran, and Yeung 2018; Simonovsky and Komodakis 2017). Unfortunately, it's striking to observe that convolutional based approaches are hard to train and struggled by overfitting. Without loss of generality, we adopt *PointNet* as the 3D shape encoder in our framework.

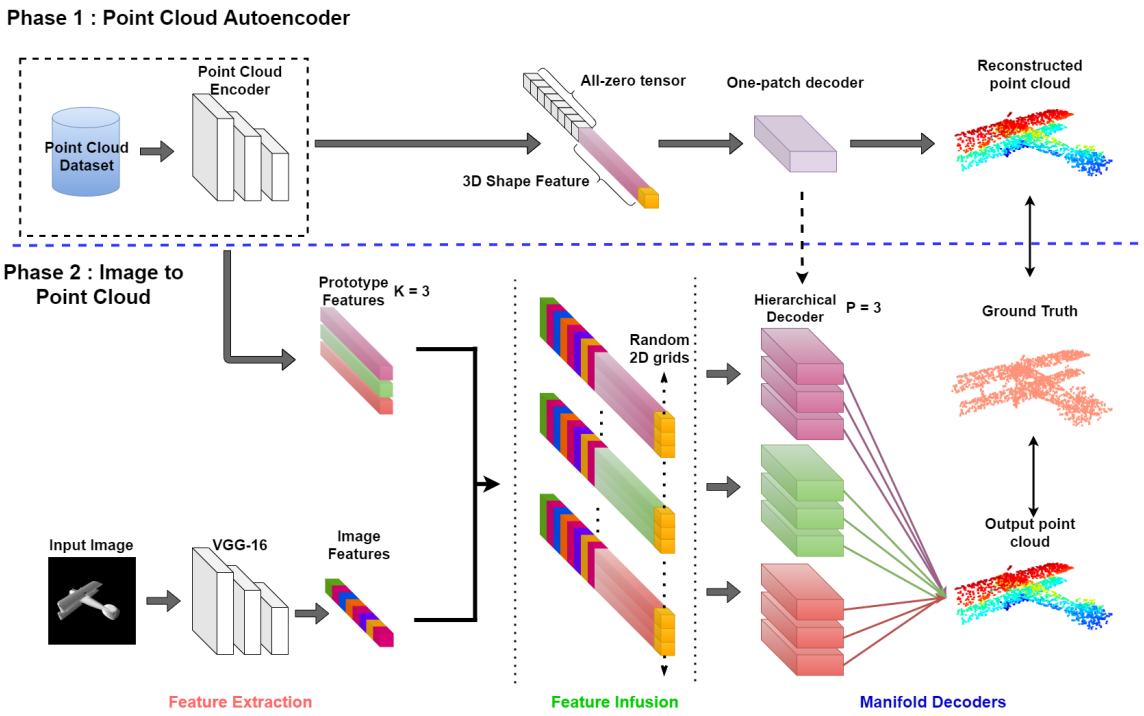

Figure 2: Approach overview: A two-phase single-view point cloud reconstruction solution. **a)** We firstly warm-up the network by solving a point cloud self reconstruction problem. Namely, we trained a point cloud autoencoder in this phase. **b)** We build the actual image-to-point-cloud network in the 2nd phase. K prototype features are computed using the trained point cloud encoder and *KMeans* clustering. Notice that it can be extended to a multi-class version by repeating this operation every class. We then infuse the image feature and prototypes with random 2D grids and feed them to a hierarchical decoder to construct the final point cloud. The decoder has K MLP clusters and each contains P one-patch decoders ($K = 3, P = 3$ in this figure).

Framework

Overview

The objective of our framework is to reconstruct a complete point cloud from a single 2D projection, with the aids from existing point cloud datasets. A point cloud is presented as $S = \{p_i\}_{i=1}^{N}$, where N denotes its cardinality and p_i is a point in the 3D Euclidean space with coordinate (x_i, y_i, z_i). Based on our observation and existing literature, we found that $N = 2048$ is sufficient to preserve the major structure of given 3D object (Chang et al. 2015).

An image is a 2D projection of a 3D shape, while it contains limited information about its source. This missing information is crucial for a successful reconstruction. We introduce a set of 3D prototype features $T = \bigcup_{c \in C} \{t_i^{(c)}\}_{i=1}^{K}$ to compensate the information loss. Here C is the set of classes, K is the set size, and t_i is a prototype feature derived from a point cloud dataset. Note that we use the same point cloud dataset both phases for simplicity in Fig 2, while a comprehensive external resource, like ShapeNet, is welcome in the 1st phase. Let I be the input image and $f(\cdot)$ be the predefined image feature extractor (*e.g.* VGG-16). Our goal is to learn a neural network $G(\cdot\|\theta)$ such that the distance between the synthesized point cloud and the ground truth is minimized. The objective is formulated as:

$$\arg\min_{\theta} D(S, G(f(I) \oplus T \| \theta)) \tag{1}$$

where $\theta = \{\phi, \rho\}$ denotes network parameter: ϕ is the parameter of the feature extractor and ρ belongs to the manifold decoders. $D(\cdot, \cdot)$ is the distance function and two common choices of this distance metric are Chamfer Distance (CD) and Earth Mover Distance (EMD). Mathematically speaking, CD and EMD between two sets of points are formulated as following:

$$d_{\mathrm{CD}}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2 \tag{2}$$

$$d_{\mathrm{EMD}}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2 \tag{3}$$

where $\phi : S_1 \to S_2$ is a bijection. Although these two metrics are widely used by different frameworks, each of them has its own concentration during the generating process, thus leads to different 3D shapes when being used as loss function. EMD favors the shapes close to the "mean-shape" of the given category (Fan, Su, and Guibas 2017). Consider a set of airplanes, the model always outputs a cabin will get a better score. In contrast, CD tends to cover all components while leading to a splashy shape that blurs the object's geometric structure. Noted that the sum term in both equations, such computation is expensive and is another hint we should keep the number of points, N, reasonable in practice.

Architecture and Workflow

Our framework, TDPNet, has two training phases as illustrated in Fig 2. The goal of such design is to prepare the point cloud encoder for prototype extraction and to help the decoder gain the ability to reconstruct point clouds from an isolated 3D feature in the 1st phase. We later build the actual image-to-point-cloud pipeline in the 2nd phase, where the real image features and 3D prototypes are combined and decoded.

To be concrete, we train a point cloud autoencoder in the first step. The encoder may either be a pointwise MLP or a convolution-based network (We adopt *PointNet* in this paper). The one-patch decoder is a simple MLP (1538-512-256-128) comprised of ReLU non-linearities on the first three layers and tanh on the last output layer. This decoder will be later used to initialize our hierarchical decoder. Finally, since the image feature I_{feat} is not available now, we mask out that part with an all-zero tensor. The loss function of this autoencoder may either be CD or EMD.

For the 2nd phase, let's start from a simple scenario, where all inputs are from the same category (*e.g.*, airplane). We need to generate K prototype features with the trained point cloud encoder. A clustering algorithm, such as *KMeans*, is applied to obtain K clusters. We then initialize the prototype features by the centroid of each cluster. Notice that we need to repeat this operation for every category when facing a general multi-class problem setting. Nevertheless, the centroids are insufficient for a realistic reconstruction, we will apply a Froze-Finetune training strategy on those prototypes, which is explained in the next subsection.

Our hierarchical decoder contains K MLP clusters and every cluster has P one-patch decoders. We used $K = 3, P = 3$ in Fig 2. We concatenate the image feature I_{feat} with corresponding 3D prototypes $T^{(c)} = \{t_i^{(c)}\}_{i=1}^{K}$ to obtain K fused vectors. Recall that there are K MLP clusters and each of them will handle one fused vector. Each fused vector is replicated P times and endowed with randomly sampled 2D grids. With this configuration, we allow each prototype to contribute to the final result democratically and dedicated to different local regions. In the end, all the $K \times P$ patches produced by the decoder are collected onto the final point cloud. The pseudocode for this training process is presented below.

Algorithm 1: Phase 1 Training

input: A point cloud dataset $S = \{s_i\}_{i=1}^{m}$

for *Number of training epochs* **do**
  for $batch \leftarrow 1$ **to** $\lfloor m/batch\_size \rfloor$ **do**
    Compute 3D features for $S_{\{batch\}}$;
    Concatenate 3D features with dummy image features and 2D random grids ;
    Generate $\hat{S}_{\{batch\}}$ from the fused vector;
    Compute $d_{CD}(\hat{S}_{\{batch\}}, S_{\{batch\}})$;
    Update the network ;
  end
end

Algorithm 2: Phase 2 Training

input: A paired image & point cloud dataset $D = \{I_i, s_i\}_{i=1}^{n}$

Generate K prototypes with KMeans;
Initialize all MLPs with the 1st phase decoder;
for *Number of training epochs* **do**
  if *epoch* $<$ *frozen_period* **then**
    Froze the prototype
  else
    Activate prototype tuning
  end
  for $batch \leftarrow 1$ **to** $\lfloor n/batch\_size \rfloor$ **do**
    Compute 2D features from $I_{\{batch\}}$;
    Concatenate 2D features with 3D prototypes and 2D random grids ;
    Generate $\hat{S}_{\{batch\}}$ from the fused vector ;
    Compute $d_{CD}(\hat{S}_{\{batch\}}, S_{\{batch\}})$;
    Update the network ;
  end
end

Dynamic 3D Prototype

In this section, we demonstrate how we generate the 3D prototype and why it's important to use a frozen-finetune training schema.

We initialize the 3D prototypes, $T = \bigcup_{c \in C} \{t_i^{(c)}\}_{i=1}^{K}$, by the *KMeans* centroids of a collection of point cloud features. Recall that we trained a point cloud AE in the 1st phase and initialized our hierarchical decoder with its one-patch decoder. Indeed, a multiple-patches decoder is acceptable in the point cloud AE, whereas the performance won't be impaired heavily and it could cause negative effects to the initialization. To reduce the requirement of computational resources and minimize the training time, we stay with the one-patch decoder. Since the clustering algorithm won't change the feature space, we conclude that the centroid captured meaningful information and can be decoded by the one-patch decoder. Examples of extracted prototypes are presented in the experiment section. The one-patch decoder is capable of reconstructing a complete point cloud without any image feature. In other words, the network learns the mechanism to incorporate the 2D features with 3D prototypes in the 2nd phase. The prototypes seem to be random noise and lead to model collapse provided that 1st phase does not exist (Mejjati et al. 2018), yet it's also not advisable to keep the prototype untouched during the training. For example, an external dataset provides rich shape information while its underlying distribution may not be consistent with the prototype distribution of the 2nd phase.

To overcome aforementioned problems, we propose to froze the prototypes for first few epochs. *UAGAN* (Mejjati et al. 2018) embraced this strategy to balance the generator and the discriminator in a GAN. The idea behind this op-

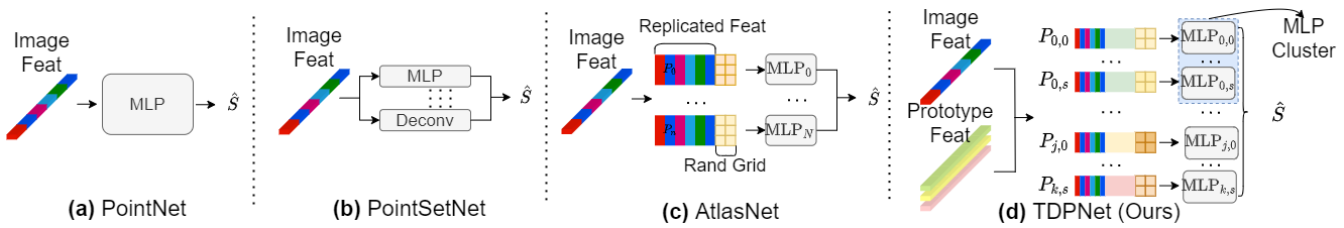

Figure 3: Comparison of four SOTA image-to-point-cloud decoders. PointNet and PointSetNet both use a single image feature and have no surface assumption, while PointSetNet is able to generate multiple plausible shapes thanks to the deconv branch and MoN loss function. The AtlasNet tries to deform multiple 2D grids onto local 2-manifold but they still stay on the single image feature. Our TDPNet is capable of fusing 2D and 3D information (K prototype features). Each prototype controls an MLP cluster and every cluster contains P MLP components, each of which can fold a distinct 2D grid onto a specific local point set.

eration is general and intuitive: Mode collapse is caused by the joint training of one or more auxiliary components. The alleviation of it is allowing update parameters for only one component in the early stage. With the same idea, We froze the prototype for the first 30 epochs and allow them to be fine-tuned in the rest epochs, thus they can capture the correct information in the target dataset.

Hierarchical Manifold Decoder

Following the **AtlasNet** (Groueix et al. 2018) convention, generating a point cloud can be considered as generating a surface of a 3D shape. The surface (shape) of a 3D object is a differentiable 2-manifold that embedded in the ambient 3D Euclidean space: $M^2 \in \mathbb{R}^3$. A point cloud is considered as a sampled discrete subset of the surface $S = \{p_i \in M^2 \cap \mathbb{R}^3\}$.

Before we dive into the reconstruction process, let us first start with some basic concepts (Zhao et al. 2019):

Definition 1 (Diffeomorphisms)
A **diffeomophism** is an invertible, differentiable map between two differentiable surfaces.

Definition 2 (Chart)
Consider an open set $U \in \mathbb{R}^2$. A **chart** C is a diffeomorphism $C : M^2 \to U \in \mathbb{R}^2$ that maps an open neighborhood in 3D space to its 2D embedding.

Definition 3 (Parameterization)
Given a chart C, let $\Psi \equiv C^{-1} : \mathbb{R}^2 \to M^2$ be the inverse of this chart. Ψ is called a **parameterization**.

Definition 4 (Atlas)
A set of charts with images covering the 2-manifold is called an **atlas**: $A = \cup_i C_i(p_i)$

With these definitions, we conclude that a 2D point set can be deformed to a surface with a parameterization Ψ. In the other words, we are not learning an exact mapping from the hidden vector to a point set $\hat{S}$, but trying to find function(s) $\Psi(U|\rho)$ to generate the 2-manifold, such that $\Psi(U|\rho) \approx S$. ρ is a lower-dimensional parameterization of these functions such that $|\rho| < |S|$.

It has been proved that "*Given that C^{-1} exists, arbitrary 3D surfaces can be reconstructed if ψ is approximated by a 3-layer MLP*" (Groueix et al. 2018). Based on this theorem and the universal approximation theorem (Csáji et al. 2001), we are able to state that a point cloud S can be universally reconstructed up to a precision ϵ via an MLP with H hidden units.

With these definitions and theorems, previous point cloud decoder networks can be categorized based on their architecture. As presented in Fig 3, *PointNet* (Qi et al. 2017a) could be extended to an image-to-point-cloud network naturally by swapping the point cloud encoder with an image feature extractor, and replace the FC-layer with an MLP decoder. *PointSetNet* (Fan, Su, and Guibas 2017) improves this architecture by adding a deconvolutional branch and hierarchically combining the output from FC-branch into the final result. However, both of them lack the grid structure and their decode functions depend upon a single latent feature. In other words, these two frameworks have the assumption $U = \emptyset$. *AtlasNet* (Groueix et al. 2018) is an advanced version of *FoldingNet* (Yang et al. 2018). They shared the same intuition of manifold deformation, whereas *FoldingNet* restricts itself to one manifold, and *AtlasNet* deforms multiple 2D grids with MLPs.

Although *AtlasNet* performs very well in the point cloud self-reconstruction task, it assumes that 2D features and 3D features have the same impact on the result point cloud, which is not true in practice. Our framework addresses this problem by combining 2D features and 3D prototypes together. Therefore, our framework is a generalization of *AtlasNet*, which can be obtained by setting all the prototypes to zero ($T = \emptyset$). Noted that our framework contains one MLP cluster per prototype, thus a prototype can affect several regions if desired.

Experiments

Setting

We evaluate our method quantitatively and qualitatively on different challenging tasks, such as single category image-to-point-cloud generation, multiple category image-to-point-cloud generation and multiple plausible shapes generation.

Data Two datasets are used for evaluation in this paper: ModelNet (Wu et al. 2015), and ShapeNet (Chang et al.

	PointSetNet	PointFlow	AtlasNet 1 Patch	AtlasNet 32 Patches	TDPNet K4P4	TDPNet K8P4
Airplane	6.48 / 36.63	7.96 / **14.47**	6.38 / 21.33	5.94 / 21.22	**5.40** / 19.42	5.44 / 17.13
Bathtub	13.16 / 53.35	33.34 / 21.18	11.36 / 20.55	12.06 / 14.94	10.64 / **14.93**	**9.54** / 14.96
Bed	11.80 / 42.49	10.07 / 15.09	10.14 / 19.28	9.16 / 32.87	7.80 / 13.39	**7.27** / **13.45**
Chair	14.81 / 42.14	11.16 / **15.35**	11.03 / 22.16	9.47/ 16.92	9.86 / 17.43	**8.74** / 17.55
Desk	18.75 / 47.43	28.83 / **23.63**	21.14 / 32.87	21.67 / 34.87	**16.18** / 27.32	18.59 / 31.04
Dresser	18.89 / 55.88	12.15 / 15.52	13.11 / 17.58	10.35 / 14.39	**9.88** / **14.27**	10.18 / 14.71
Monitor	16.49 / 43.91	10.88 / **14.82**	12.88 / 21.59	11.38 / 18.33	10.51 / 16.06	**10.05** / 16.41
Sofa	12.56 / 45.56	9.56 / 14.86	8.66 / 17.03	8.09 / 16.26	**7.59** / **14.36**	8.11 / 15.10
Table	15.46 / 43.69	9.72 / **14.94**	10.49 / 18.45	8.06 / 16.54	7.97 / 15.91	**7.48** / 16.11
Toilet	13.88 / 45.85	12.78 / **16.12**	9.87 / 19.89	9.39 / 21.38	**8.92** / 19.36	9.12 / 20.02

Table 1: Single-View Reconstruction (per category) for ModelNet dataset, trained on each category. The results of each framework are reported in format "CD / EMD". Chamfer Distance is multiplied by 10^3 and Earch Mover Distance is multiplied by 10^2 for better visualization. Both metrics are computed on 2048 points. Best results are bolded.

	Airplane	Chair	Car
PointSetNet	3.36 / 34.71	6.35 / 45.15	8.63 / 52.39
PointFlow	4.12 / 12.17	8.95 / 14.73	8.42 / 11.50
AtlasNet	2.82 / **11.39**	6.67 / **13.81**	4.42 / 11.39
TDPNet	**2.34** / 13.85	**6.32** / 14.87	**4.20** / **11.18**

Table 2: Single-View Reconstruction (per category) for ShapeNet dataset, trained on each category. The results are organized with the same format of Tab 1. *AtlasNet 1 patch* and *TDPNet K4P4* are omitted.

2015). For the ModelNet dataset, we borrow the processed data from *MVCNN* (Su et al. 2015), which contains 4,899 CAD models across 10 categories and each model is accompanied by 12 2D projections. Regarding ShapeNet, we sampled 3 categories, which totally contains 14,355 CAD models. We then render 12 views of each 3D shape based on the Blinn-Phong shading formula with a black environmental map (Blinn 1977). The single RGB image of each CAD model in both datasets is chosen from corresponding 12 2D projections randomly. Both datasets are divided into a 80/20 train/test split randomly.

Baselines We compare our proposed TDPNet with three SOTA frameworks. **PointSetNet** (Fan, Su, and Guibas 2017), **AtlasNet**(Groueix et al. 2018) and **PointFlow** (Yang et al. 2019a). Only the first two methods claim that they have the capacity to generate the point cloud from a single image. Nevertheless, *PointFlow* solved the point cloud reconstruction task from the perspective of statistics and achieved promising results. Thus, we include this method to study its capacity for the image-to-point-cloud task. For a fair comparison, all the images features are extracted by a VGG-16 and we provide an additional run of AtlasNet with 32 patches, which is equal to the maximum number of MLP decoders in our framework, $K = 8, P = 4$.

Evaluation Metrics We evaluated the synthesized point cloud by comparing it to ground truth shapes using two criteria: Chamfer Distance and Earth Mover Distance. Formulas and physical meanings of these two criteria are presented in Equation 2 and 3, respectively. As we will show in later sections, although these two numerical metrics have limitations, they unveil different insights to the performance of all models (Yang et al. 2019a).

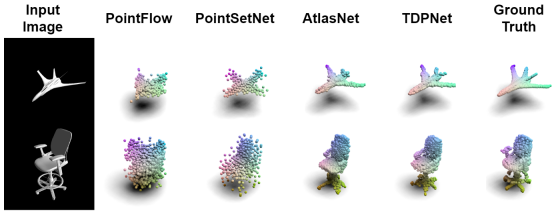

Figure 4: Examples of qualitative comparison among different method. From left to right: Input image, *PointFlow*, *PointSetNet*, *AtlasNet*, TDPNet (Ours) and Ground Truth.

Implementation Before training, the input point clouds are aligned to a common ground plane and size normalized. Data augmentation like random rotation and jitter are applied. All the RGB images are center cropped and resize to 224×224. To train our network, we use an ADAM optimizer with an initial learning rate of 10^{-3} and a batch size of 32 for the first phase. We still use an ADAM optimizer in the second phase but set the initial learning rate as 10^{-4}. We arranged 100 epochs per training stage and froze the prototype for 30 epochs.

3D Shape Reconstruction from RGB Images

We first report quantitative results for the single category image-to-point-cloud generation in Tab 1 (ModelNet) and Tab 2 (ShapeNet). Notice that the networks are trained on each category **separately**. We observe that for single view reconstruction, our proposed method consistently achieves better CD and competitive EMD in every categories. Additionally, we can see that our approach is significantly better than *AtlasNet* with the same number of decoders.

The results of our framework trained on all categories of ModelNet are in Tab 3. Although our performance downgrade slightly comparing to the single-category tests, we still achieve better scores in most categories. Strictly speaking, it's not a fair comparison because our framework uses label information to decide which prototypes to use.

PointSetNet and *PointFlow* are two extremes in this task. *PointSetNet* performs moderately in CD but has intolerable EMDs. *PointFlow* achieves amazing EMDs with unstable CD. Considering the visualization in Fig 4, the results reveal the shortcomings of these two metrics. CD favors

	PointSetNet	PointFlow	AtlasNet P32	TDPNet K8P4
Airplane	20.92 / 42.54	14.28 / 24.52	8.03 / 33.56	**5.68 / 21.74**
Bathtub	44.83 / 53.05	21.11 / 20.25	16.17 / 37.27	**8.73 / 18.17**
Bed	18.77 / 49.82	12.37 / 17.65	8.11 / **15.08**	**7.50** / 17.90
Chair	23.12 / 44.99	17.54 / 27.41	14.10 / 27.52	**9.52 / 23.85**
Desk	27.79 / 49.06	31.72 / 27.97	21.32 / 41.49	**16.61 / 27.84**
Dresser	54.45 / 56.65	12.15 / **15.52**	17.64 / 22.37	**9.74** / 20.35
Monitor	31.83 / 50.88	15.05 / 24.22	13.08 / 20.76	**9.09 / 20.01**
Sofa	16.59 / 50.19	14.09 / 18.97	10.10 / **16.88**	**7.64** / 20.30
Table	22.45 / 47.88	10.91 / 20.77	9.23 / 25.74	**7.03 / 18.36**
Toilet	23.29 / 49.67	17.14 / **27.28**	9.89 / 30.17	**8.96** / 27.75

Table 3: Single-View Reconstruction (per category) for ModelNet, trained on all categories. The results are in format "CD / EMD" and they are scaled by 10^3 and 10^2, respectively.

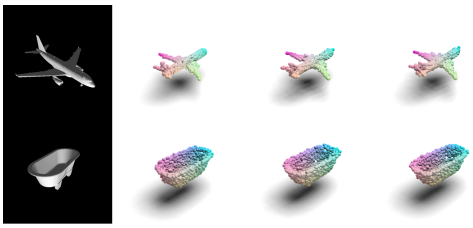

Figure 5: Multiple predictions for a single input image. Note that the input view can be a 2D projection from a different angle, while we can still reconstruct the 3D shape correctly.

a splashy result that covers more regions, whereas EMD prefers a "mean-shape" that roughly matches every instance in a given category (Yang et al. 2019a; Fan, Su, and Guibas 2017). Based on these observations and previous literatures, we suggest that **a model is better if it has better CDs and moderate EMDs**. Adequate EMD guranantees that the result is in the right category and small CD make sure all regions are recovered properly.

Generating Multiple Plausible Clouds

The random sampled coordinates from the 2-manifold naturally allows prediction of different shapes, given the same input image. This model behavior is valuable because of the ambiguous 2D to 3D construction behavior (Sung et al. 2018). Fig 4 shows examples of a set of predictions given one image. We observed that our network can reveal its uncertainty about the shape or the ambiguity in the input.

Ablation Study

Prototypes and Decoders

We report the result of different settings of K and P in Tab 4. Notice how our approach generally improves as we increase the number of prototypes and MLP decoders. Another interesting observation is that our approach usually obtains more benefits from the increase of prototypes compared to the increase of MLP decoders. Our approach consistently outperforms *AtlasNet* when the number of decoders is equal ($K \times P = 32$), which further justified the effectiveness of combining 2D and 3D features. Finally, we observed that adding extravagant decoders doesn't necessarily improve the model performance.

	P=1	P=2	P=4	P=8
K=2	6.41	6.28	5.70	5.63
K=4	6.15	6.09	5.40	5.66
K=8	6.08	5.82	5.44	5.41
K=16	5.73	5.41	5.25	5.07

Table 4: Chamfer Distance measured on ModelNet-airplane with different hyper-parameter setting. The result are multiplied by 10^3 for better visualization.

	Froze	Finetune	Froze-Finetune
Airplane	6.12	5.76	5.44
Bathtub	11.37	10.84	9.54

Table 5: Chamfer Distance ($\times 10^3$) measured on 2 ModelNet categories with different training strategies. We adopt configuration $K = 8, P = 4$ in this experiment.

Frozen-Finetune Training

In this section, we evaluate the effectiveness of the Frozen-Finetune training. Tab 5 shows the quantitative results. The performance of *Frozen* and *Finetune* are approximately the same since the network may recognize the first one as constants and the finetuned version as random noise. On the other hand, our approach is capable of avoiding mode collapse and utilizing the 3D shape information.

Fig 6 visualizes two samples of prototypes. We combined the prototype with a dummy image feature and feed it to the 1st phase decoder. Although this may not be the optimal visualization because we trained the prototype features to incorporate with **real** image features, we can still find that the finetuned prototype looks closer to the category mean-shape, whereas the frozen version tends to be more sparse in the space.

Conclusion

In this paper, we introduced a unified framework for generating point clouds from an image. Our approach bridges the gap between 2D and 3D features by introducing a flexible 3D prototype mechanism. The superiority of our framework compared to SOTA methods is demonstrated by both quantitative metrics and qualitative figures. Moreover, the proposed method is a general framework that can be easily extended to new 3D reconstruction techniques (e.g. point cloud representation) with few modifications.

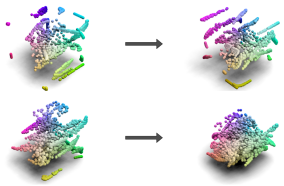

Figure 6: Sampled prototypes on ModelNet-bathtub. Left are the initial centroids and Right are the finetuned prototypes. See context for more information.

Acknowledgements

The research reported herein was supported in part by NSF awards DMS-1737978, DGE-2039542, OAC-1828467, OAC-1931541, DGE-1906630; and an IBM faculty award (Research).

References

Blinn, J. F. 1977. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, 192–198.

Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; and Leonard, J. J. 2016. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics* 32(6): 1309–1332.

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. 2015. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* .

Choy, C. B.; Xu, D.; Gwak, J.; Chen, K.; and Savarese, S. 2016. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, 628–644. Springer.

Csáji, B. C.; et al. 2001. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary* 24(48): 7.

Dai, A.; Ruizhongtai Qi, C.; and Nießner, M. 2017. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5868–5877.

Fan, H.; Su, H.; and Guibas, L. J. 2017. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 605–613.

Gadelha, M.; Maji, S.; and Wang, R. 2017. 3d shape induction from 2d views of multiple objects. In *2017 International Conference on 3D Vision (3DV)*, 402–411. IEEE.

Groueix, T.; Fisher, M.; Kim, V. G.; Russell, B. C.; and Aubry, M. 2018. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 216–224.

Gwak, J.; Choy, C. B.; Garg, A.; Chandraker, M.; and Savarese, S. 2017. Weakly supervised generative adversarial networks for 3d reconstruction. *arXiv preprint arXiv:1705.10904* 2.

Hartley, R.; and Zisserman, A. 2003. *Multiple view geometry in computer vision*. Cambridge university press.

Hua, B.-S.; Tran, M.-K.; and Yeung, S.-K. 2018. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 984–993.

Kar, A.; Tulsiani, S.; Carreira, J.; and Malik, J. 2015. Category-specific object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1966–1974.

Li, C.-L.; Zaheer, M.; Zhang, Y.; Poczos, B.; and Salakhutdinov, R. 2018. Point cloud gan. *arXiv preprint arXiv:1810.05795* .

Mejjati, Y. A.; Richardt, C.; Tompkin, J.; Cosker, D.; and Kim, K. I. 2018. Unsupervised attention-guided image-to-image translation. In *Advances in Neural Information Processing Systems*, 3693–3703.

Navaneet, K.; Mathew, A.; Kashyap, S.; Hung, W.-C.; Jampani, V.; and Babu, R. V. 2020. From Image Collections to Point Clouds with Self-supervised Shape and Pose Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1132–1140.

Nguyen-Phuoc, T.; Li, C.; Theis, L.; Richardt, C.; and Yang, Y.-L. 2019. Hologan: Unsupervised learning of 3d representations from natural images. In *Proceedings of the IEEE International Conference on Computer Vision*, 7588–7597.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, 5099–5108.

Rezende, D. J.; Eslami, S. A.; Mohamed, S.; Battaglia, P.; Jaderberg, M.; and Heess, N. 2016. Unsupervised learning of 3d structure from images. In *Advances in neural information processing systems*, 4996–5004.

Schonberger, J. L.; and Frahm, J.-M. 2016. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4104–4113.

Simonovsky, M.; and Komodakis, N. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3693–3702.

Su, H.; Maji, S.; Kalogerakis, E.; and Learned-Miller, E. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, 945–953.

Sung, M.; Su, H.; Yu, R.; and Guibas, L. J. 2018. Deep functional dictionaries: Learning consistent semantic structures on 3d models from functions. In *Advances in Neural Information Processing Systems*, 485–495.

Tulsiani, S.; Zhou, T.; Efros, A. A.; and Malik, J. 2017. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2626–2634.

Wang, N.; Zhang, Y.; Li, Z.; Fu, Y.; Liu, W.; and Jiang, Y.-G. 2018. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 52–67.

Wang, Z.; Wang, Y.; Lin, Y.; Delord, E.; and Latifur, K. 2020. Few-Sample and Adversarial Representation Learning for Continual Stream Mining. In *Proceedings of The Web Conference 2020*, 718–728.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1912–1920.

Yang, B.; Wen, H.; Wang, S.; Clark, R.; Markham, A.; and Trigoni, N. 2017. 3d object reconstruction from a single depth view with adversarial learning. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 679–688.

Yang, G.; Huang, X.; Hao, Z.; Liu, M.-Y.; Belongie, S.; and Hariharan, B. 2019a. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, 4541–4550.

Yang, J.; Zhang, Q.; Ni, B.; Li, L.; Liu, J.; Zhou, M.; and Tian, Q. 2019b. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3323–3332.

Yang, Y.; Feng, C.; Shen, Y.; and Tian, D. 2018. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 206–215.

Zhang, Y.; Liu, Z.; Liu, T.; Peng, B.; and Li, X. 2019. RealPoint3D: An efficient generation network for 3D object reconstruction from a single image. *IEEE Access* 7: 57539–57549.

Zhao, Y.; Birdal, T.; Deng, H.; and Tombari, F. 2019. 3D point capsule networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1009–1018.

Zhu, H.; Wei, H.; Li, B.; Yuan, X.; and Kehtarnavaz, N. 2020. A Review of Video Object Detection: Datasets, Metrics and Methods. *Applied Sciences* 10(21): 7834.