PVDECONV: POINT-VOXEL DECONVOLUTION FOR AUTOENCODING CAD CONSTRUCTION IN 3D

Kseniya Cherenkova⋆†   *Djamila Aouada*⋆   *Gleb Gusev*†

⋆ SnT, University of Luxembourg   † Artec3D

ABSTRACT

We propose a Point-Voxel DeConvolution (*PVDeConv*) module for 3D data autoencoder. To demonstrate its efficiency we learn to synthesize high-resolution point clouds of 10k points that densely describe the underlying geometry of Computer Aided Design (CAD) models. Scanning artifacts, such as protrusions, missing parts, smoothed edges and holes, inevitably appear in real 3D scans of fabricated CAD objects. Learning the original CAD model construction from a 3D scan requires a ground truth to be available together with the corresponding 3D scan of an object. To solve the gap, we introduce a new dedicated dataset, the CC3D, containing 50k+ pairs of CAD models and their corresponding 3D meshes. This dataset is used to learn a convolutional autoencoder for point clouds sampled from the pairs of 3D scans - CAD models. The challenges of this new dataset are demonstrated in comparison with other generative point cloud sampling models trained on ShapeNet. The CC3D autoencoder is efficient with respect to memory consumption and training time as compared to state-of-the-art models for 3D data generation.

Index Terms— CC3D, point cloud autoencoder, CAD models generation, Scan2CAD.

1. INTRODUCTION

Recently, deep learning (DL) for 3D data analysis has seen a boost in successful and competitive solutions for segmentation, detection and classification [1], and real-life applications, such as self-driving, robotics, medicine, and augmented reality. In industrial manufacturing, 3D scanning of fabricated parts is an essential step of product quality control, when the 3D scans of real objects are compared to the original Computer Aided Design (CAD) models. While most consumer solutions for 3D scanning are good enough for capturing the general shape of an object, artifacts can be introduced in the parts of the object that are physically inaccessible for 3D scanning, resulting in the loss of sharp features and fine details.

This paper focuses on recovering scanning artifacts in an autoencoding data-driven manner. In addition to presenting a new point cloud autoencoder, we introduce a new 3D dataset, referred to as *CC3D*, which stands for *CAD Construction*
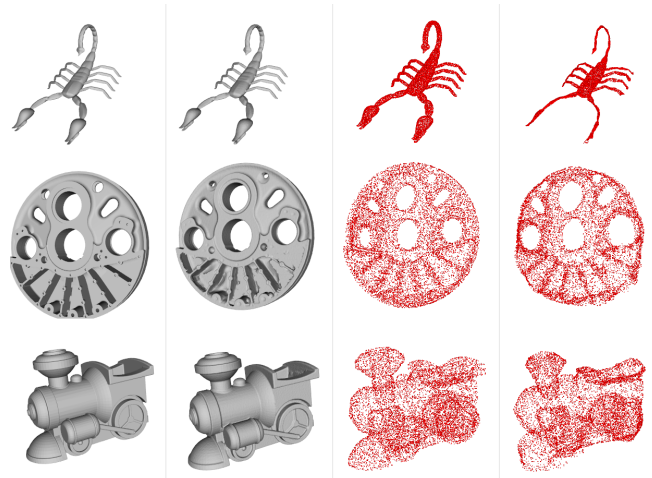

Fig. 1. Examples of CC3D data: From left to right, CAD models, corresponding 3D scans, 10k input point clouds and results of the proposed autoencoder.

in 3D. We further provide an analysis focused on real 3D scanned data, keeping in mind real-world constraints, i.e., variability, complexity, artifacts, memory and speed requirements. The first two columns in Fig. 1 give some examples from CC3D data; the CAD model and its 3D scanned version in triangular mesh format. While the most recent existing solutions [2, 3, 4, 5] on 3D data autoencoders mostly focus on low-resolution data configuration (approximately 2500 points), we see it more beneficial for real data to experiment in higher-dimension. It is what brings the important 3D object details into the big data learning perspective.

Several publicly available datasets related to CAD modelling, such as ModelNet [6], ShapeNet [7], and ABC [8], have been released in the last years. The summary of the features they offer can be found in Table 1. These datasets have boosted the research on deep learning on 3D point clouds mainly.

Similarly, our CC3D dataset should support research efforts in addressing real-world challenges. Indeed, this dataset provides various 3D scanned objects, with their ground-truth CAD models. The models collected in CC3D dataset are not restricted to any object's category and/or complexity. 3D scans offer challenging cases of missing data, smoothed ge-

ometry and fusion artefacts in the form of varying protrusions and swept holes. Moreover, the resolution of 3D scans is typically high with more than 100k faces in the mesh.

In summary, the contributions of this paper include: (1) A 3D dataset, CC3D, a collection of 50k+ aligned pairs of meshes, a CAD model and its virtually 3D scanned counterpart with corresponding scanning artifacts; (2) A CC3D autoencoder architecture on 10k point clouds learned from CC3D data; (3) A Point-Voxel DeConvolution (*PVDeConv*) block for the decoder part of our model, combining point features on coarse and fine levels of the data.

The remainder of the paper is organized as follows: Section 2 reviews relevant state-of-the-art works in 3D data autoencoding. In Section 3 we give a brief overview of the core components our work is built upon. Section 4 describes the main contributions of this paper in more details. In Section 5 the results and comparison with related methods are presented. Section 6 gives the conclusions.

2. RELATED WORK

The choice of DL architecture and 3D data representation is usually defined by existing practices and available datasets for learning [9]. Voxel-based representations have pioneered 3D data analysis, applying 3D Convolution Neural Network (CNN) directly on a regular voxel grid [10]. Despite the improved models in terms of memory consumption, e.g., [11], their inability to resolve fine object details remains the main limiting factor in practical use.

Other works introduce convolutions directly on graph structures, e.g., [12]. They attempt to generalize DL models to non-Euclidean domains such as graphs and manifolds [13], and offer the analogs of pooling/unpooling operations as well [14]. However, they are not applicable for learning on real unconstrained data as they require either meshes to be registered to a common template, or inefficiently deal with meshes of up to several thousand faces, or are specific to segmentation or classification tasks only.

Recent advances in developing efficient architectures for 3D data analysis are mainly related to point cloud based methods [15, 16]. Decoders [17, 2, 18, 3, 19] have made point clouds a highly promising representation for 3D object generation and completion using neural networks. Successful works in generative adversarial network (GAN) (e.g.,[20]), show the applicability of different GAN models operating on the raw point clouds.

In this paper, we comply with the common autoencoder approach, i.e., we use a point cloud encoder to embed the point cloud input, and design a decoder to generate a complete point cloud from the embedding of the encoder.

3. BACKGROUND AND MOTIVATION

We herein present the fundamental building blocks that comprise the core of this paper, namely, the point cloud, metric on it, and the DL backbone. All together, these elements make the CC3D autoencoder perform efficiently on high-resolution 3D data.

A point cloud S can be represented as $S = \{(p_k, f_k)\}$, where each p_k is the 3D coordinates of the kth input point, and f_k is the feature corresponding to it, and the size of f_k defines the dimensionality of the points feature space. Note that while it is straightforward to include auxiliary information (such as points' normals) to our architecture, in this paper we exclusively employ the xyz coordinates of p_k as the input data.

We base on Point-Voxel Convolution (PVConv), a memory efficient architecture for learning on 3D point cloud presented in [21]. To the best of our knowledge, it is the first development of autoencoder based on PVCNN as the encoder. Briefly, PVConv combines the fine-grained feature transformation on points with the coarse-grained neighboring feature aggregation in the voxel space of point cloud. Three basic operations are performed in the coarse branch, namely, voxelization, followed by voxel-based 3D convolution, and the devoxelization. The point-based branch aggregates the features for each individual with multilayer perceptron (MLP), providing high resolution details. The features from both branches are aggregated into a hidden feature representation.

The formulation of convolution in both voxel-based and point-based cases is the following:

$$y_k = \sum_{x_i \in N(x_k)} K(x_k, x_i) \times F(x_i), \qquad (1)$$

where for each center x_k, and its neighborhood $N(x_k)$, the neighboring features $F(x_i)$ are convolved with the kernel $K(x_k, x_i)$. The choice of PVCNN is due to its efficiency in training on high-resolution 3D data. Indeed, it makes it a good candidate for working with real-life data. As it is stated in [21], PVConv combines advantages of point-based methods, reducing memory consumption, and voxel-based, improving the data locality and regularity.

For the loss function, Chamfer distance [22] is used to measure the quality of the autoencoder. It is a differentiable metric, invariant to permutation of points in both ground-truth and target point clouds, S_G and S, respectively. It is defined as follows:

$$d_{CD}(S, S_G) = \sum_{x \in S} \min_{y \in S_G} \|x - y\|^2 + \sum_{y \in S_G} \min_{x \in S} \|x - y\|^2.$$
$$(2)$$

As it follows from its definition, no correspondence or equal number of points in S and S_G is required for the computation of d_{CD}, making it possible to work within different resolutions for the encoder and decoder.

4. PROPOSED AUTOENCODING OF 3D SCANS TO CAD MODELS

This paper studies the problem of 3D point cloud autoencoding in a deep learning setup, and in particular, the choice of the architecture of a 3D point cloud decoder for efficient reconstruction of point clouds sampled from corresponding pairs of 3D scans and CAD models.

4.1. CC3D dataset

The CC3D dataset of 3D CAD models was collected from a free online service for sharing CAD designs [23]. In total, the collected dataset contains 50k+ models in STEP format, unrestricted to any category, with varying complexity from simple to highly detailed designs (see examples in Fig. 1). These CAD models are converted to meshes, and each mesh was virtually scanned using proprietary 3D scanning pipeline developed by Artec3D [24]. The typical size of the resulting scans is in the order of 100K points and faces, while the meshes converted from CAD models are usually more than an order of magnitude lighter.

In order to illustrate the uniqueness of our dataset, Table 1 summarizes the available CAD-like datasets and semantic information they provide. Unlike ShapeNet [7] and ModelNet [6], the CC3D dataset is a collection of 3D objects unrestricted to any category, with the complexity varying from very basic to highly detailed models. One of the most recent datasets, the ABC dataset [8] would have been a valuable collection due to its size for our task if it had contained 3D scanned models alongside with ground-truth CAD objects. The availability of CAD-3D scan pairings, the high-resolution of meshes and variability of the models make the CC3D dataset stand out among other alternatives. The CC3D dataset will be shared with the research community.

4.2. CC3D Autoencoder

Our decoder is a modified version of PVCNN, where we cut the final classification/segmentation layer. The proposed

Dataset	#Models	CAD	Curves	Patches	Semantics	Categories	3D scan
CC3D (ours)	50k+	✓	✗	✗	✗	✗	✓
ABC [8]	1M+	✓	✓	✓	✗	✗	✗
ShapeNet [7]	3M+	✗	✗	✗	✓	✓	✗
ShapeNetCore [7]	51k+	✗	✗	✗	✓	✓	✗
ShapeNetSem [7]	12k	✗	✗	✗	✓	✓	✗
ModelNet [6]	151k+	✗	✗	✗	✓	✓	✗

Table 1. Summary of datasets with CAD-like data. Note that only ABC and CC3D offer CAD models in b-rep (boundary representation) format in addition to triangular meshes.
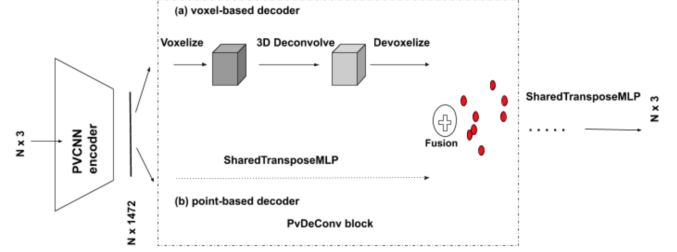
Fig. 2. Overview of CC3D autoencoder architecture and PVDeConv module. The features from coarse voxel-based and fine point-based branches are fused to be unwrapped to the output point cloud.

PVDeConv structure is depicted in Fig. 2. The fine point-based branch is implemented as shared transposed MLP, allowing to maintain the same number of points throughout the autoencoder, while the coarse branch allows the features to be aggregated at different voxel grid resolutions, thus modelling the neighborhood information at different scales.

The PVDeConv block consists of 3D volumetric deconvolutions to aggregate the features, dropout, the batch normalization and the nonlinear activation function after each 3D deconvolution. Features from both branches are fused at the final level and MLP to produce the output points.

The transposed 3D convolution operator, used in PVDeConv, multiplies each input value element-wise by a learnable kernel, and sums over the outputs from all input feature channels. This operation can be seen as the gradient of 3D convolution, although it is not an actual deconvolution operation.

5. EXPERIMENTS

We evaluate the proposed autoencoder by training first on our CC3D dataset, and then on the ShapeNetCore [7] dataset.

5.1. Training on CC3D

Dataset. CC3D dataset is randomly split into three non-intersecting folds: 80% for training, 10% for testing and 10% for validation. Ground-truth point clouds are generated by uniformly sampling $N = 10k$ points on the CAD models surfaces, while the input point clouds are sampled in the same manner from corresponding 3D scans of the models. The data is normalized to (0, 1).

Implementation Details. The encoder follows the structure in [21], the coarse blocks are ((64, 1, 32), (64, 2, 16), (128, 1, 16), 1024), where triplets describe voxel-based convolutional PVConv block in terms of number of channels, number of blocks, and voxel resolution. The last number describes the resulting embedding size for the coarse part, and being combined with shared MLP cloud blocks = (256, 128), gives the feature embedding size of 1472. The decoder coarse blocks are ((128, 1, 16), (64, 2, 16), (64, 1, 32), 128), where
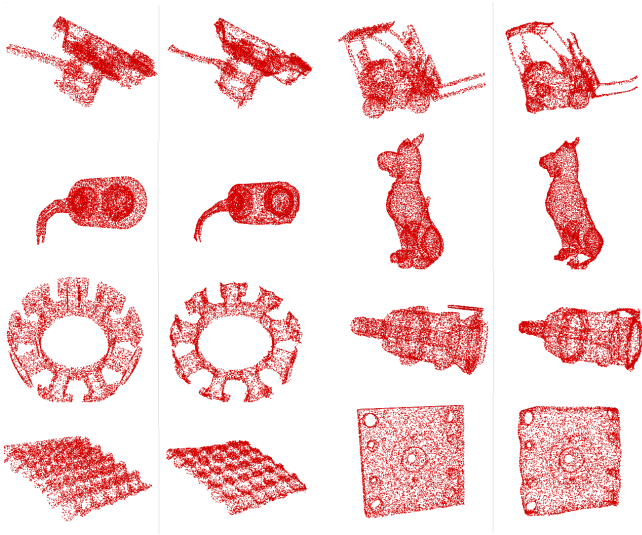
Fig. 4. Chamfer distance distribution for our autoencoder. On test set of CC3D for point clouds of size $N = 10$k, mean Chamfer distance is 1.26×10^{-3} with standard deviation of 0.794×10^{-3}. ShapeNetCore test set with $N = 2.5$k, it is 0.804×10^{-3} with standard deviation 0.766×10^{-3}.

Fig. 3. Results of our autoencoder on CC3D data with 10k points for input and output. The left of each pair of results is the input point cloud of 10k, the right is the autoencoder reconstruction of 10k points.

the triplets are PVDeConv concatenated with decoder point-based fine blocks of size (256, 128).

   Training setup. The autoencoder is trained with Chamfer loss for 50 epochs on two Quadro P6000 with batch size 80 in data parallel mode. The overall training takes approximately 15 hours. The best model is chosen based on the validation set.

   Evaluation. The qualitative results of our autoencoder on the CC3D data are presented in Fig. 3. We notice that the fine details are captured in these challenging cases.

Method	Chamfer distance, 10^{-3}
AtlasNet [2]	1.769
FoldingNet [17]	1.648
PCN [19]	1.472
TopNet [3]	0.972
Ours	**0.804**

Table 2. CC3D autoencoder results on ShapeNetCore dataset: comparison against previous works ($N = 2.5$k).

5.2. Training on ShapeNetCore

To demonstrate the competitive performance of our CC3D autoencoder, we train it on the ShapeNetCore dataset following the train/test/val split of [3], with the number of sampled point $N = 2500$ for a fair comparison. Since we do not have scanned models for the ShapeNet data, we add a 3% Gaussian noise to each point's location. The rest of the training setup is replicated from the CC3D configuration. The final metric is the mean Chamfer distance averaged per model across all classes. The numbers for other methods are reported from [3]. The results of the evaluation of our method against state-of-the-art methods are shown in Table 2. We note that
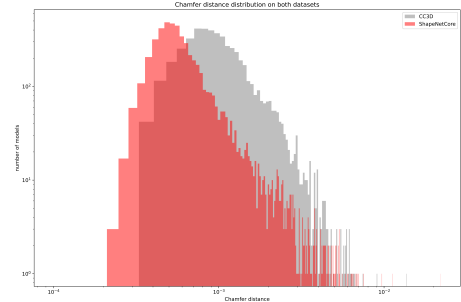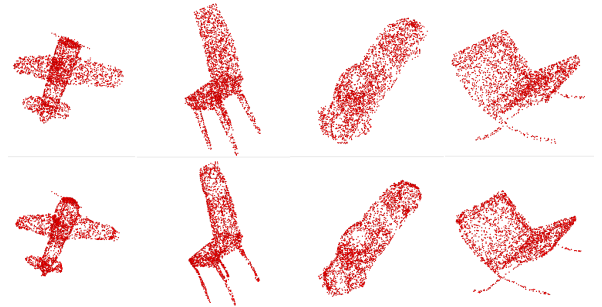

Fig. 5. Results of our autoencoder on ShapeNetCore data. The top row is the input 2.5k point clouds, the bottom is the reconstruction of our autoencoder.

our result surpasses the previous works by a significant margin. Qualitative examples on ShapeNetCore data are given in Fig. 5. The distribution of distances given in Fig. 4 implies that CC3D dataset presents advanced challenges for our autoencoder, where it performs at 1.26×10^{-3} average Chamfer distance, while it reaches 0.804×10^{-3} on ShapeNetCore.

6. CONCLUSIONS

In this work, we proposed a Point-Voxel Deconvolution (*PVDeConv*) block for a fast and efficient deconvolution on 3D point clouds. It was used in combination with a new dataset, CC3D, for autoencoding 3D Scans to their corresponding synthetic CAD models. The CC3D dataset offers pairs of CAD models and 3D scans, totaling to 50k+ objects. Our CC3D autoencoder on point clouds is memory and time efficient. Furthermore, it demonstrates superior results compared to existing methods on ShapeNet data. As future work, different types of losses will be investigated to improve the sharpness on edges, such as quadric [5]. Testing the variants of CC3D autoencoder with different configurations of stacked PVConv and PVDeConv layers will also be considered. Finally, we believe that the CC3D dataset itself could assist in real 3D scanned data analysis with deep learning methods.

7. REFERENCES

[1] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun, "Deep learning for 3d point clouds: A survey," *ArXiv*, vol. abs/1912.12033, 2019.

[2] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry, "Atlasnet: A papier-mâché approach to learning 3d surface generation," *CoRR*, vol. abs/1802.05384, 2018.

[3] Lyne P Tchapmi, Vineet Kosaraju, S. Hamid Rezatofighi, Ian Reid, and Silvio Savarese, "Topnet: Structural point cloud decoder," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[4] Isaak Lim, Moritz Ibing, and Leif Kobbelt, "A convolutional decoder for point clouds using adaptive instance normalization," *CoRR*, vol. abs/1906.11478, 2019.

[5] Nitin Agarwal, Sung-Eui Yoon, and M. Gopi, "Learning embedding of 3d models with quadric loss," *CoRR*, vol. abs/1907.10250, 2019.

[6] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1912–1920.

[7] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu, "Shapenet: An information-rich 3d model repository," 2015.

[8] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo, "Abc: A big cad model dataset for geometric deep learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[9] Eman Ahmed, Alexandre Saint, Abd El Rahman Shabayek, Kseniya Cherenkova, Rig Das, Gleb Gusev, Djamila Aouada, and Björn E. Ottersten, "Deep learning advances on different 3d data representations: A survey," *ArXiv*, vol. abs/1808.01462, 2018.

[10] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 922–928.

[11] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[12] Soubhik Sanyal Anurag Ranjan, Timo Bolkart and Michael J. Black, "Generating 3D faces using convolutional mesh autoencoders," in *European Conference on Computer Vision (ECCV)*. 2018, pp. 725–741, Springer International Publishing.

[13] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, July 2017.

[14] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or, "Meshcnn: A network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 90, 2019.

[15] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *CoRR*, vol. abs/1706.02413, 2017.

[16] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, Oct. 2019.

[17] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian, "Foldingnet: Interpretable unsupervised learning on 3d point clouds," *ArXiv*, vol. abs/1712.07262, 2017.

[18] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari, "3d point-capsule networks," *CoRR*, vol. abs/1812.10775, 2018.

[19] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert, "Pcn: Point completion network," in *3D Vision (3DV), 2018 International Conference on*, 2018.

[20] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabás Póczos, and Ruslan Salakhutdinov, "Point cloud GAN," *CoRR*, vol. abs/1810.05795, 2018.

[21] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han, "Point-voxel cnn for efficient 3d deep learning," in *Advances in Neural Information Processing Systems*, 2019.

[22] Haoqiang Fan, Hao Su, and Leonidas Guibas, "A point set generation network for 3d object reconstruction from a single image," 07 2017, pp. 2463–2471.

[23] "3dcontentcentral," https://www.3dcontentcentral.com.

[24] "Artec3d," https://www.artec3d.com/.