Dynamic Graphs For Point Cloud Completion

Leron Julian
18-889 Final Project Report
Carnegie Mellon University
ljulian@andrew.cmu.edu

Abstract

Point clouds, commonly used to represent shapes and scenes, are often incomplete due to low sensor resolution by the imaging device or occlusion between the object and sensor. As a result, performing further analysis on these incomplete point clouds results in sub optimal predictions. Many works have tackled the issue of point cloud completion using a learning framework that directly predicts a complete point cloud given an incomplete input while leaving it up to the model to estimate the underlying shape of the incomplete input. This project will investigate how adding dynamic graphs into the learning pipeline better helps the model understand the overall structure of the input and leads to a more accurate reconstructed point cloud. Results from this project show that by adding dynamic graphs within the learning pipeline leads to a more compact completed point cloud and better metric results.

1. Introduction

Point clouds, a scattered collection of points, can be efficiently used to represent shapes and objects in both 2D and 3D. Point cloud data on real-world objects and scenes can be recovered from devices such as a LiDAR sensor or depth sensor and then be converted to other surface representations to visualize scenes and objects in 3D. A common limitation of using point clouds is that capturing real-world data often results in missing points or "gaps" in the point cloud representation. This is often caused by occlusion between the scene and the sensor, lighting differences, noise, or limited sensor resolution. These incomplete point clouds will therefore result in poor or incorrect analysis such as segmentation or classification as well as poor reconstruction in 3D.

Image inpainting, commonly used in 2D images, refers to the process of filling-in missing data in a designated region of an image or object [2]. As shown in Figure 1 missing pixels (in black) are filled in with the correct pixel values for the pixel locations.

Figure 1. a.) Ground truth image. b.) Image with missing pixel values. c.) Inpainted image where missing pixels are filled with assumed value.

However, for missing points in point cloud representations, inpainting, commonly referred to as point cloud completion, can be used as a method to fill points and complete the cloud with a high order of accuracy. An example of point cloud completion is shown in Figure 2.

Many works have tackled the issue of point cloud completion using a deep learning framework that directly predicts a completed point cloud given an incomplete input while leaving it up to the model to estimate the underlying shape of the incomplete input. My project hypothesizes that by implementing a method to directly allow the model to learn the underlying shape of the incomplete point cloud will result in a better predicted point cloud as output. Therefore, this project will investigate how adding dynamic graphs into the learning pipeline better helps the model understand the overall structure of the input and leads to a more accurate reconstructed point cloud.

2. Related Work

For the overall task of point cloud completion, 3 main categories have been investigated: Geometry-Based Methods, Alignment-Based Methods, and Learning-Based Methods. In this project, I will be focusing on learning-based methods for point cloud completion in which previous works commonly used encoder-decoder models. A major limitation to these methods however is that they do not take into consideration the underlying shape/structure of the point clouds which inhibits the generality of point cloud completion, especially for real-world large scenes.

Zhang et. al [9] presented a view-guided solution for the task of point cloud completion by taking the missing

Figure 2. Incomplete and Complete Point Cloud

crucial global structure information from an extra single-view RGB image. Inferring the underlying shape of a point cloud from an RGB image is not always feasible due to the fact that RGB images are not always available in real-world scenarios. Also, factors such as lighting differences can affect objects in the RGB image which will in turn affect this model.

Yuan et. al [8] tackled the task of point cloud completion through a novel Point Completion Network (PCN) that directly operates on raw point clouds without any structural assumptions of the incomplete point cloud. Through this encoder-decoder network, PCN is able to produce high-quality coarse and dense point clouds given an incomplete point cloud as input. The major limitation of this implementation is that PCN does not assume anything about the underlying structure of the point clouds. I believe that taking the underlying shape of the incomplete input into consideration will result in a better completed point cloud. As a result, PCN will be used as my base network in which I will add dynamic graphs into the learning pipeline to investigate how adding these will better help the model understand the overall structure of the input and lead to a more accurate reconstructed point cloud.

3. Method

The methods for this project will incorporate a base model with improvements to benefit the task of point cloud completion.

3.1. Base Model

Given that PCN will be used as the base model for this project, the fine details of the model can be read in their paper [8]. However, the model takes as input an incomplete point cloud X in 3D (x,y,z). Random points are removed from the point cloud to create an incomplete point cloud such that there are only 512 points in the input. The model then produces 2 outputs: a coarse output (Y_{coarse}) with

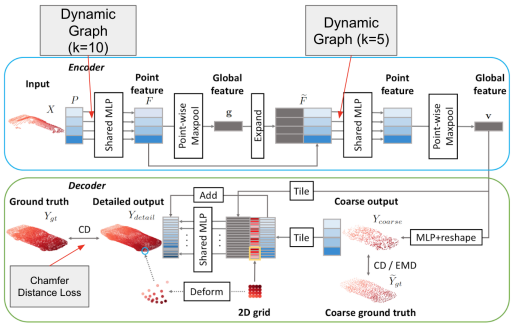

Figure 3. Proposed network with $k = 10$ neighbors and $k = 5$ neighbors throughout.

Figure 4. Visualization of how dynamic graphs aggregates local neighbor points.

m points and a dense output point cloud (Y_{dense}) with **n** points. For this project **m** $= 1024$ and **n** $= 8192$. The benefit of this is that more dense point clouds can be predicted which produces better predictions on point cloud analysis tasks. As stated earlier, PCN does not make any assumptions about the underlying shape of the incomplete point cloud in which this project will improve using dynamic graphs into the PCN learning pipeline.

3.2. Dynamic Graphs

The dynamic graph implemented in this projected will be based off of Dynamic Graph CNN for Learning on Point Clouds [7] in which a directed graph $\mathcal{G}$ is used to represent a local neighborhood in the point cloud structure. $\mathcal{G}$ can be represented as any neighborhood algorithm, however for this project k-nearest neighbor (k-NN) is used as the directed graph. In the learning pipeline, channel-wise aggregation is used on the edge features of on the graph. A visualization is shown in figure 4. The whole idea behind this is that instead of generating point features directly from their embeddings in an MLP, the dynamic graph generates edge features that describe the relationships between a point and its neighbors.

3.3. Proposed Model

To this end, my proposed model will incorporate these dynamic graphs throughout the learning pipeline of the base PCN network. A figure of this pipeline is visualized in figure 3 with $k = 10$ neighbors within the first dynamic

graph addition and $k = 5$ neighbors within the second dynamic graph addition. All other implementations of the PCN model remains the same with $\mathbf{m} = 1024$ points for (Y_{coarse}) and $\mathbf{n} = 8192$ points for (Y_{dense})

3.4. Naive Model

The naive model for this project is used to compare the results on point cloud completion on a simple model and to analyze how the predicted complete point cloud is affected with the addition of dynamic graphs throughout the learning pipeline. The naive model is modeled as a simple encoder-decoder network of shared multi-layer perceptrons (MLPs) [3] with the same channel dimensions as PCN.

4. Experiments

A few experiments have been analyzed on this project to test how adding dynamic graphs improves the task of point cloud completion.

4.1. Data

For this project, the datasets that will be used for training and testing is the Shapenet [1] dataset and real-world point clouds captured from my iPhone 13 Pro LiDAR sensor. Although the ShapNet dataset contains numerous object categories, only 3 were chosen. For the real world data captured from an iPhone, 3 single objects were tested: an office chair, a piano, and a shoe box.

4.2. Loss Function

For this project, the loss function will be the difference between the predicted point cloud and the ground truth point cloud. Due to the nature of point clouds, the loss function needs to be invariant to the point cloud permutations. Therefore, Chamfer distance (CD) [4] is used as the loss function between the predicted point cloud P_1 and the ground truth point cloud P_2. The equation for CD is expressed as the equation:

$$CD\left(P_1, P_2\right) = \frac{1}{|P_1|} \sum_{x \in P_1} \min_{y \in P_2} \|x - y\|_2 + \frac{1}{|P_2|} \sum_{y \in P_2} \min_{x \in P_1} \|y - x\|_2 \tag{1}$$

Therefore, the complete loss function for training is:

$$L\left(Y_{coarse}, Y_{dense}, Y_{truth}\right) = CD\left(Y_{coarse}, \tilde{Y}_{truth}\right) + \alpha\, CD\left(Y_{dense}, Y_{truth}\right) \tag{2}$$

Where $\tilde{Y}_{truth}$ is the predicted coarse point cloud and α is a hyper-parameter that determines how much does the dense prediction influences the overall loss.

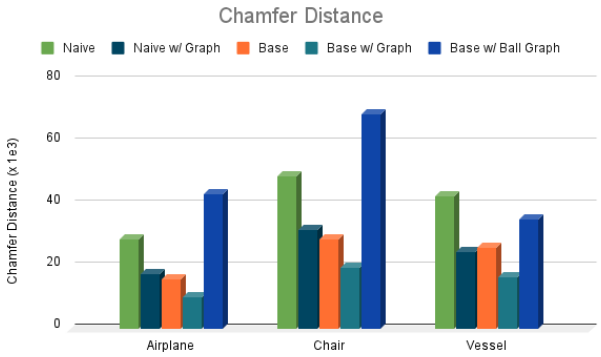

Figure 5. CD results on ShapeNet data.

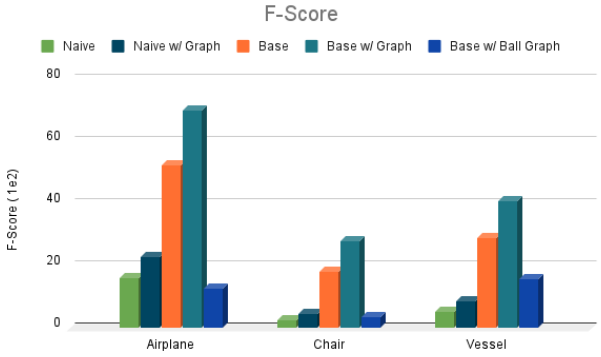

Figure 6. F-Score results on ShapeNet data.

4.3. Training Details

The models were trained and tested on the shapenet dataset with further testing on real-world data from the iPhone 13 Pro LiDAR sensor. The model was trained on 3 object categories: airplane, chair, and vessel and tested on the same categories on unseen data. The models were trained for 400 epochs with Adam optimizer and a learning rate of 0.0001 with a decay rate of 0.7 and a batch size of 32.

5. Results

The results of how adding dynamic graphs within the learning pipeline will be analyzed based on qualitative results as well as quantitative results in terms of CD and F-score.

5.1. ShapeNet Results

Again, the models were trained across 3 categories within the ShapeNet dataset: airplane, chair, and vessel. In terms of CD, a lower CD value is better while a higher F-score is better. As shown in figures 5 and 6 adding k-NN dynamic graphs to the base model allowed the predicted com-

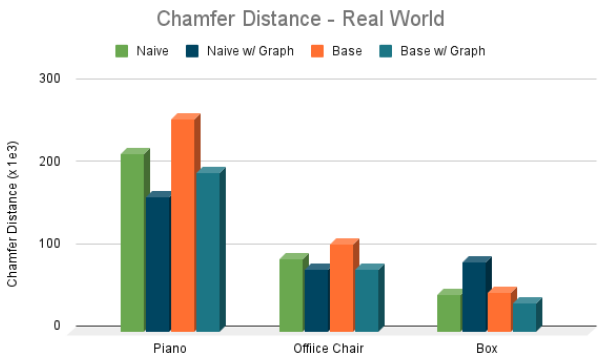

Figure 7. CD results on real-world data.

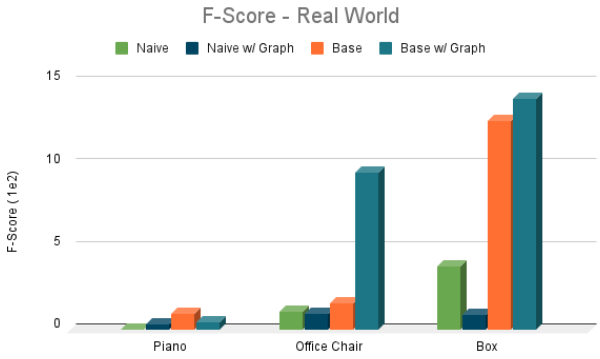

Figure 8. F-Score results on real-world data.

plete point cloud to achieve better metric results compared to the other models. Even adding k-NN dynamic graphs to the naive model resulted in better quantitative results. Also, to test the results on another neighboring function, ball-query [5] was used which looks at points within a certain radius as opposed to a certain number of k points. However, I believe my implementation of the ball-query algorithm may have been incorrect which resulted in very bad predictions. Qualitative results are shown in figures 9-15. It can clearly be seen that adding dynamic graphs to the learning pipeline resulted in a better complete point cloud compared to previous works.

5.2. Real-World Results

Results on real-world data captured from an iPhone LiDAR system is used which includes single objects of an office chair, a piano, and a shoe box. Random points are selected and removed from the input such that there are 512 points used in the incomplete point cloud as input. Results on the real-world dataset based on CD and F-Score is shown in figure 7 and 8. Based on CD, the results vary on category on which model performs the best. Again, these models are only trained on 3 object categories on synthetic data which means that they will not always generalize well on real-world data across different categories. Also, my capturing of objects with my iPhone LiDAR was not the greatest due to the fact that it is not as easy to capture a LiDAR scan of a scene as it is with a regular RGB image. Further data capturing and training on more data should ultimately alleviate this issue.

6. Conclusion

Overall, this project investigated how adding dynamic graphs into the learning pipeline better helps the model understand the overall structure of the input and leads to a more accurate reconstructed point cloud when given an incomplete point cloud as input. Results from this project show that by adding dynamic graphs within the learning pipeline leads to a more compact completed point cloud and better metric results than other methods that leaves it up to the model to estimate the underlying geometry of the object. Although my model did not perform well on real-world data, I believe that it would perform better given better data to train on. Also, it would be interesting to investigate how other dynamic graph implementations such as KPConv [6] affects learning a completed point cloud.

Figure 9. Input incomplete point cloud.

Figure 10. Ground truth complete point cloud.

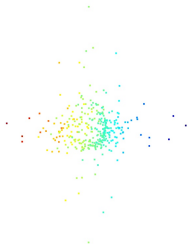

Figure 11. Naive prediction.

Figure 12. Naive prediction with k-NN Dynamic Graph.

Figure 13. Base model prediction

Figure 14. Base model prediction with k-NN Dynamic Graph.

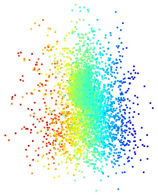

Figure 15. Base model prediction with ball query Dynamic Graph.

References

[1] Shapenet. https://shapenet.org/. Accessed: 2022-03-14. 3

[2] Image inpainting. pages 315–316, 2006. 1

[3] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 3

[4] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2463–2471, 2017. 3

[5] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 4

[6] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6410–6419, 2019. 4

[7] 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.*, 38(5), oct 2019. 2

[8] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737, 2018. 2

[9] Xuancheng Zhang, Yutong Feng, Siqi Li, Changqing Zou, Hai Wan, Xibin Zhao, Yandong Guo, and Yue Gao. View-guided point cloud completion. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15885–15894, 2021. 1