PÁZMÁNY
Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

Tutored Research Project II.

Artificial intelligence-based sensor system for water rescue and disaster management

Csatári, Dominik
Computer Science Engineering MSc

Supervisor: Dr. Hatvani, Janka
Faculty Mentor: Dr. Cserey, György Gábor

2024

Declaration of Authenticity

I, the undersigned Dominik Csatári, student of the Faculty of Information Technology and Bionics at the Pázmány Péter Catholic University, hereby declare that I have prepared the present Tutored Research Project myself without any unauthorized help, and that I have only used the sources specified in the Tutored Research Project. I have clearly marked all parts that I have literally taken from other sources or have rewritten or translated while keeping the meaning of the original text, also indicating the source thereof. I have not submitted this Tutored Research Project to any other study programs.

_____

Dominik Csatári

Contents

Chapter 1

Introduction

1.1. Background

This Tutored Research Project 2 is motivated by the sinking of the cruise ship Hableány in the Danube, and aims to implement a system that will make it much easier to locate and rescue victims in a disaster, even in difficult circumstances. The whole of this project, is to create such a system. The whole project consists of more phases. The scope of this semester was to explore possible methods to densify sparse point clouds or complete partial point clouds.

A point cloud can be created from a measurement made with a sonar or a LiDAR sensor. Real-life data is usually incomplete or too sparse. These issues could limit classification and segmentation tasks on the data. To counter this there is an area of research called shape completion. With the properties of neural networks it tries to complete shapes based on their structure. Newer models even try to create detailed outputs so no information is lost. These networks promise a lot of opportunity. After completion a point cloud will have a better structure, giving more feedback of a sunken ship, or a collapsed bridge or finding mines in the water. These are just some current research ideas focused on underwater shape completion, but shape completion itself has many other application areas.

1.2. Outline

The summary of the chapters of the thesis work:

Related Methods This chapter describes the related works regarding shape completion.

Methods This chapter describes the tasks I have implemented and the necessary tools I used.

Results This chapter describes the results I have gathered from performing the tasks introduced in the previous chapter.

Summary In this chapter, I summarize my results, draw conclusions, and outline future plans for this project.

Chapter 2

Related Works

2.1. Shape Completion

In this section I will briefly talk about the problem statement for which shape completion offers a solution to. I will start with an introduction to point clouds, then work my way down to explaining the necessity of these architectures.

Point clouds are grouped points having 3 values (x, y, z), their coordinates. They sometimes can have a $4th$ dimension, intensity. In our use case it is not that important as the transformed points are already trusted in a sense that they are involved in the model. In future work, however, this information could be used for differentiating materials. There is a basic rule in image processing or in information theory in general, called data processing inequality. It takes a random variable Z in a Markov Chain of $X \to Y \to Z$, meaning Z only depends on Y and is independent from X. This can be expressed with the joint probability

$$p(x, y, z) = p(x)p(y \mid x)p(z \mid y) = p(y)p(x \mid y)p(z \mid y). \tag{2.1}$$

This means that there is no process (in this isolated scenario) of Y, which can improve its information (I) regarding X. With mutual information we can state the following:

$$I(X; Y) \geq I(X; Z) \tag{2.2}$$

This means that post-processing cannot increase information. [1]

This statement fits most image processing algorithms or if we look at the 3D point cloud applications as an array of the point cloud's segmentation, it also fits for example interpolation techniques. However, recent works of the last decade show, that in post-processing we can gather information from X with different methods and with this knowledge acquired we are able to increase the output object's information from our existing general knowledge. If a model like this behaves accordingly, we have a great probable assumption on X's wider information, rather than just a hallucinations of sort. These methods are for example super resolution, with its newly found and wildly spread methodology or point cloud completion in case of 3D point cloud studies. [2] Point cloud completion methods main tasks are making denser point clouds from a coarse input or generating parts of the object's point cloud, which were obscured at the time of creating the point cloud. Most of these processes are built on encoder-decoder architecture having a global feature vector created from a sparse point cloud input from which an improved point cloud is generated. The pooling method most of these processes use, creates a great core for the object. In Section 2.2. I will introduce some architectures made for shape completion.

2.2. Architectures

In this section I am going to introduce the general purpose of an encoder-decoder architecture, then I will discuss various methods in more details.

2.2.1 Encoder-Decoder

In shape completion studies, neural networks are extremely useful due to their ability to capture a wide range of variations. More precisely, the encoder–decoder architecture is used in most cases.

The encoder – decoder architecture is usually used for sequence-to-sequence learning. Its two main parts are the encoder and the decoder. The encoder from the input creates a vector of sort which is the input for the decoder. Than the decoder creates an output sequence which is similar to the input sequence, however, some parts are changed due to the contribution of weights in each layer of this architecture. In our case the output point cloud will consist of more points. The structure can recognize more complex relationships within the given point cloud. [3]

2.2.2 Point Completion Network

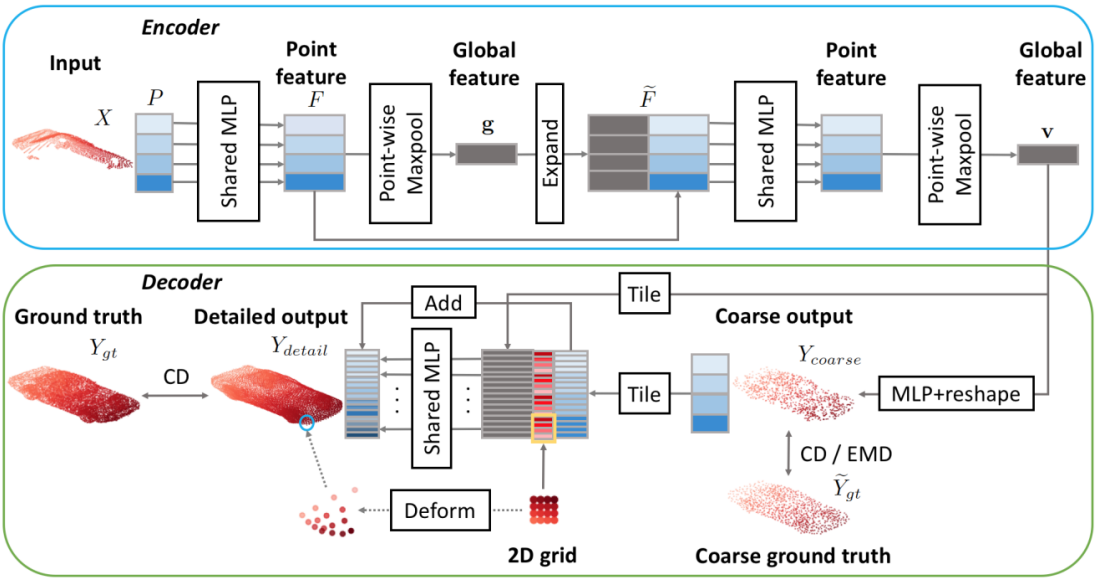

Figure 2.1: Encoder-Decoder architecture of PCN. Source: [4]

Architecture

Point Completion Network(PCN) is an early model for creating denser point clouds from the input in the form of an encoder – decoder. It is a tool for shape completion for point cloud inputs. It was developed because there was a high demand for completing unfinished partial or incomplete point clouds. Incompleteness in this case means barely or not recognizable structure of a point cloud. Before this method, the primary approach for shape completion involved voxelizing the 3D point clouds and then applying convolutional networks to the resulting voxel grid. However, these methods were limited by the resolution of the grid. Additionally, directly using the 3D point cloud input requires less memory and as it operates on raw point cloud data it keeps the initial geometric information. [4]

For 3D point clouds these networks face two complex task: creating a great feature extractor and loss function as it needs to be permutation invariant. Direct cases of the tasks mentioned above are for example robustness to noise and sparsity and generalization. The problem statement is as follows: We have a sparse 3D point cloud, X, gathered with a sensor such as LiDAR or sonar. Y is the point cloud representation of the real scene, a dense and complete structure. Our goal is to predict Y based on X. For this

purpose, this model suggests a neural network which gives a generic solution for multiple categories. [4]

As mentioned above, the structure of this model is based on the encoder – decoder architecture. From the input point cloud the network creates a feature vector $\boldsymbol{v}$ which has k dimensions ($k = 1024$). For creating $\boldsymbol{v}$, first some multilayer perceptron (MLP) converts X to $\boldsymbol{F}$, from which maxpooling (MP) extracts a global feature vector $\boldsymbol{g}$ containing the coarse, important geometric information. As it is shown in Figure 2.1, $\boldsymbol{g}$ is concatenated with $\boldsymbol{F}$. After the shown MLP and MP layers the final $\boldsymbol{v}$ feature vector is created. Vector $\boldsymbol{v}$ is the output of the encoder and the input for the decoder. The decoder is a multistage point generation pipeline. This pipeline is unique as it is effective at predicting a sparse set of points that represents the global geometry and also it performs quite well at predicting local surfaces. This is performed in two different stages in the model, estimating the coarse (Y_{coarse}) and the detailed (Y_{detail}) shape. [4]

For creating detailed output the network uses a local folding operation. It takes an input q_i from Y_{coarse}, and the global feature vector $\boldsymbol{v}$. Than in creates a grid of points centered around q_i. Afterwards, with the help of a shared MLP a 3D local patch is created from the 2D grid input. [4]

Loss Function

As I have mentioned in the introduction of PCN, it is a challenging task to create an appropriate loss function given we are working with 3D points. There are two loss functions that are used in most of the cases. These functions measure the distance between the ground truth and the output. These are the Chamfer Distance (CD) and the Earth Mover's Distance (EMD). [4]

$$d_{CD}(S_1, S_2) = \frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \frac{1}{|S_2|} \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2 \tag{2.3}$$

$$d_{EMD}(S_1, S_2) = \min_{\phi : S_1 \to S_2} \frac{1}{|S_1|} \sum_{x \in S_1} \|x - \phi(x)\|_2^2 \tag{2.4}$$

Where $\phi : S_1 \to S_2$ is a bijection.

Chamfer Distance shown in Equation 2.3, calculates the average closest point distances. S_1 and S_2 are the two cloud points. PCN uses the symmetric version of it CD. It is also

important to state, that this distance measure does not require S_1 and S_2 to be the same size. This means a really dense point cloud of a rough shape can have a small distance with an object. It is a part of the loss function that is great for determining rough differences between the approximated skeleton and the ground truth. For telling apart detailed differences between point clouds, Earth Mover's Distance can be used. The general form of EMD is shown in Equation 2.4. EMD finds a bijection which has the minimum distance between the two sets. In practice, it is too expensive to find the optimal bijection, that is why in most cases they use an iterative approach. Unlike CD, here the two sets are required to have the same size. The loss function is the weighted sum of these two methods. Distance for the coarse and detailed solutions is measured separately. [4]

In summary, CD is high when the global structure is different, while EMD is more evenly distributed, thus panelizes details as well. In the model's metric space EMD requires one-to-one correspondence while CD requires one-to-many, which is easier to deliver, that is why generally CD is smaller.

Dealing with noise

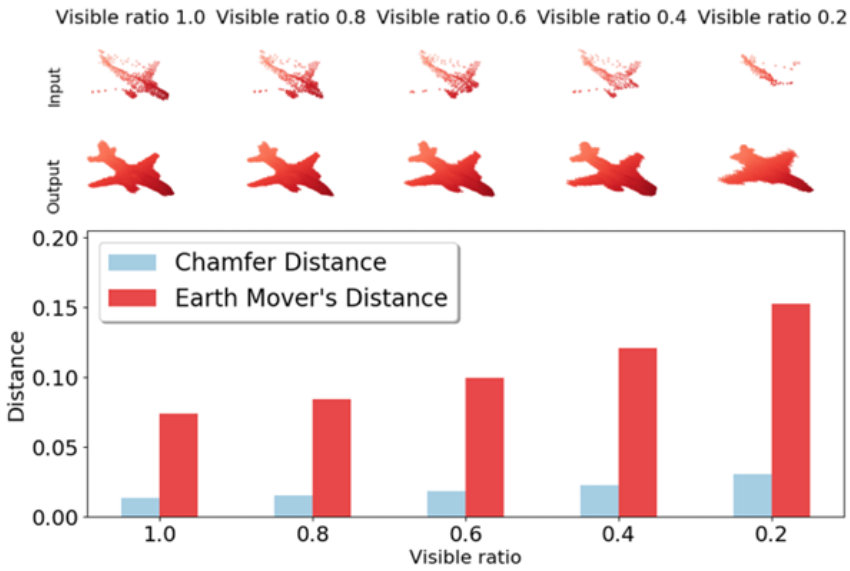

Figure 2.2: CD and EMD losses over inputs with decreasing visibility. Source: [4]

PCN promises robustness to noise, given its structure. The noise in their own measurement is a Gaussian noise with a standard deviation of 0.01. The results they have measured are shown in Figure 2.2 with the corresponding input and output. In this

scenario PCN did well, however for various other objects they can get way worse predictions. For example, for inputs where they have multiple objects or the object has a thin structure, this architecture performs poorly. This can be seen in Appendix A in Figure A.1.

2.2.3 Variational Relational Point Completion Network

Architecture

VRCNet has a unique structure compared to other models used in the field of shape completion. It has two important stages: probabilistic modeling and relational enhancement. Both of these sub-networks are built on a encoder-decoder architecture, referenced as PMNet and RENet. This network was created to perform shape completion focused on detailed outputs. Most methods generate global shape skeletons while this method tries to improve local details as well. VRCNet promises great generalizability and robustness for real–world data. [5]

Probabilistic modeling

The network focuses on predicting the sparse skeleton, as fine details and local features could only be recovered if the core of the model is set. It is important in cases, when we only have a partial input. For this general purpose of predicting the skeleton of a shape, thin method proposes PMNet structure. This prediction process is different from other methods, as it uses probabilistic modeling to predict embedded global features and learnt latent distributions. This structure is assisted with a dual-path parallel pipeline. So both reconstruction and completion is available. Reconstruction is done with a variatonal autoencoder. The completion path works on a similar basis. The two paths share weights at multiple steps but they do not at the distribution inference layers. Both work as an autoencoder should, both paths learn to approximate the coarse structure from the global features and the latent distribution. From this we have the generated skeleton of the shape from a sparse or incomplete input. [5]

Relational Enhancement network

This part of the whole network focuses on founding local relations, features which could be enhanced in the final output of the model. This structure applies a Point Self-Attention kernel, defines a neighborhood and observes relational structures on these sets. This is the core idea behind RENet. Its structure is based on a hierarchical encoder-decoder

architecture and uses Edge-aware Feature expansion module which I will introduce in Subsection 2.2.4. [5]

Loss Function

As there are various parts of the system, different loss functions are needed. It has three main parts, for reconstruction path the KL divergence is used, for reconstruction loss they chose Chamfer Distance which I have described in Equation 2.3. Completion path and Relational paths share the same idea here, the whole loss funciton is a weighted sum of these methods. [5]

Results

Their experience was evaluated with CD and F-score as the CD metric in itself could be misleading. The results of CD losses on different categories with multiple models is attached in Appendix A in Figure A.4. Also, I have attached Figure 2.3 which shows how different methods performed. As their results show, this methods based on their claims outperforms each existing methods on various categories and even on real-world data. [5]

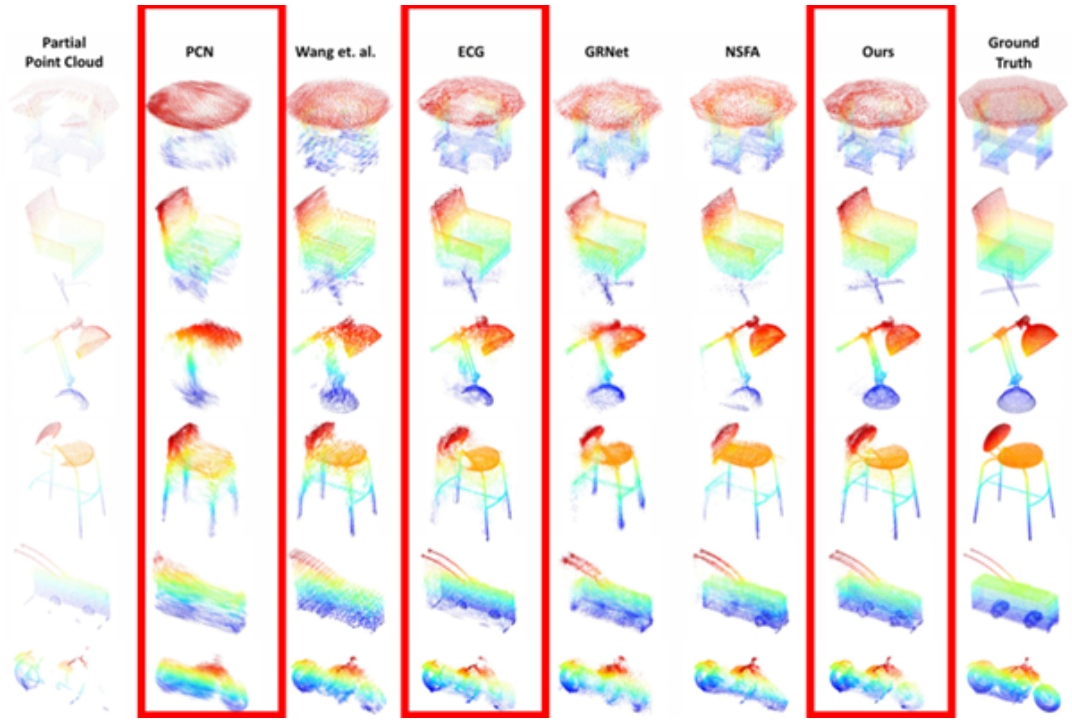

Figure 2.3: This figure compares results of different Shape Completion Methods. The results of PCN, ECG and VRCNet are highlighted. Source: [5]

2.2.4 Edge-aware Point Cloud Completion with Graph Convolution

This method created for the same shape completion task as the others presented previously, but promises edge-aware point completion with Graph convolution. This architecture also consists of two stages: skeleton generation for the input shape and resolution enhancement with correctly added points. This way the details or information of the output can be increased. [6]

As described in Subsection 2.2.2, this method also generates global features and a coarse skeleton of the shape from the input, just as PCN. Then, local features are created with a deep hierarchical encoder with graph convolution. It also implements an Edge-aware Feature Expansion which preserves local details at the upsampling steps rather than just using folding mechanism which was the case with PCN. [4] [6]

Choice of using graph convolution was convenient, as this methodology can be used with unordered points, having a flexible structure to learn neighborhoods, giving an opportunity to have a better understanding of the fine structure. The idea of skeleton generation and detail refinement structures are similar to the previously shown. However, the graph structure applies two strategies to define neighborhoods: ball query for distance-based ordering and k-nearest-neighbors for connectivity optimization. These are just the main elements of the model, the whole structure has a higher complexity. The results of this method over different categories is shown in Figure A.4 and in Figure 2.3. [6]

Chapter 3

Methods

In this Chapter I am going to briefly summarize the methods that I used and the datasets that I have met during this research. Also i am going to introduce some other tools that I have used.

3.1. Datasets

3.1.1 ModelNet40

For proof of concept, most of the pretrained models were trained with this dataset, due to the diversity it shows in a simple form. The objects themselves are simple, as each object is represented with 2048 point. As the name suggests, it includes 40 categories, ranging from household objects to vehicles and planes. It is great for showcasing the possibilities of a network. [7]

3.1.2 ShapeNet

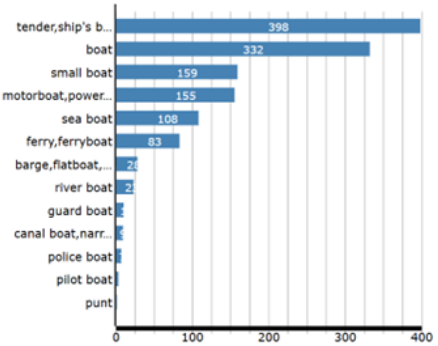

Figure 3.1: This graph shows the diversity of Shapenet's taxonomy of boats as a subgroup of watercraft. Source: [8]

ShapeNet is a widely used large-scale dataset, having 55 common object categories and 51,300 unique 3D models. The high diversity of objects is reflected in the taxonomy. For example, the common object category *watercraft* is divided into 7 subcategories and includes nearly 2000 objects in total. The diversity of this common category is shown on the following Figure 3.1. The main categories of watercrafts is attached in Appendix A in Figure A.2. [8]

3.1.3 My dataset

For this dataset I have gathered 50 *stl* files of ships, boats and ship parts. An *stl* file is a mesh object, consisting of a series of 3D vertices, and lists of vertices that describe individual polygons, the faces of the object. For generating the 3D point clouds, I have chosen the middle points of each triangle, because the original vertices were ordered too regularly. From this, I can control the number of points each point cloud contains. The size depends on how many points represent the shape or the object. These 3D Point clouds are the ground truths. From this I have created both partial and sparse clouds as inputs. In case of sonar data a partial point cloud will mean that the object is scanned from a single direction, or it is laying on the riverbed. For creating partial point clouds I have written a function `filtered_points = cut(points, plane_normal, plane_point, cut_direction)`, where

- **points** is the 3D point cloud
- **plane_normal** is the normal vector (n_x, n_y, n_z) of the plane which cuts the point cloud
- **plane_point** is a point (p_1, p_2, p_3) on the plane
- **cut_direction** determines which side of the plane to keep. Valid values are 'l' for left side and 'r' for right side.

In Figure 3.2 I have attached 2 plots, showcasing what the partial point cloud look like next to the original. I have generated this point cloud with a plane_normal $(1, 1, 1)$ and with a plane_point being $(0, 0, 0)$.

For creating sparse point clouds I have written a function which keeps a given percentage of the input's point cloud. Also for some of the points I have kept, I gave an offset value of a given distance, to generate some noise. The number of points with offset can be given. As an example, I have attached Figure A.3 in Appendix A, which shows a newly generated sparse point cloud with half of the points having a small offset.

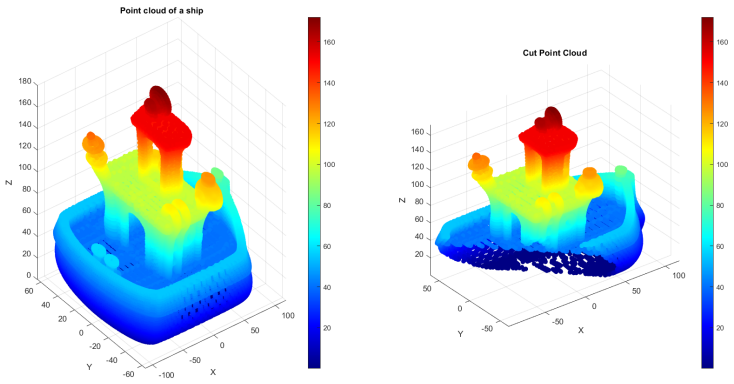

Figure 3.2: On the **left** the original point cloud of the ship can be seen, while on the **right** the newly created partial point cloud is shown.

3.2. PCN

For PCN I have implemented a library on Python called Learning3D. [9] It offered a network to train, and also a pretrained model for testing. I have applied both techniques. For testing I have chosen to test in the following way:

1. **Testing sparse input from ModelNet40.** I have tested different sparsity levels, than evaluated with the metric CD loss to compare how close was my estimation to the ground truth. The testset was based on categories, that the network had learnt on, but the objects themselves were different.

2. **Testing incomplete input from ModelNet40.** For this exercise I have utilized the cut function described in Section 3.1. After creating the incomplete structures I have tested PCN's pretrained model at this completion task. I have also evaluated on CD loss.

3. **Testing on my dataset.** For this I chose examples from my dataset covering all the types included in it by structural mean. I have also evaluated using CD loss.

In addition to standard evaluations, I will also qualitatively assess the accuracy of the approximated structure from a viewer's perspective. It is an important step, as a low CD loss in itself does not mean a good fit between the ground truth and the generated point cloud.

Chapter 4

Results

In this Chapter I am going to describe the results I have gathered by completing the tasks I have proposed in Chapter 3.

4.1. Testing sparse input from ModelNet40

For this task I have expected the model to perform well for most of the objects, as the coarse of the model will still be fed to the network. I tested at three levels: 80, 400, and 800 points. I did not choose fewer points, as they would not have provided a clear representation of the structure. I did not choose more points, as above 800, the structure becomes too clear for the network to recognise. I have tested on all of the categories. For showcasing the results I have chosen two structures. Their ground truth 2048 point point clouds can be seen in Figure 4.1 and n Figure 4.2.

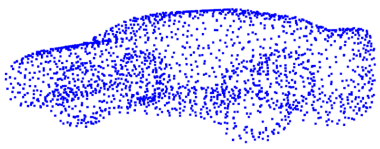

Figure 4.1: Ground Truth of the first object

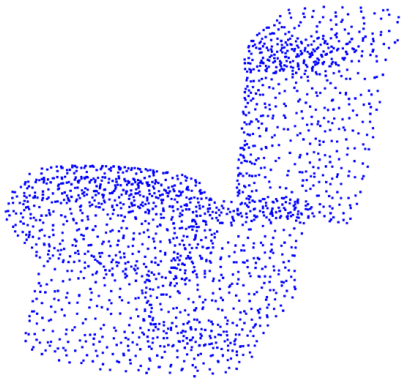

Figure 4.2: Ground Truth of the second object

The CD losses for different inputs are included in Table 4.1. The Chamfer Distance

between the ground truth and the generated point cloud decreases by increasing the input point cloud's sizes. This is the expected behaviour. However this could very well be the cause of the fact, that the network has less points to predict as it could choose the input as a coarse for its prediction if the network learnt to do that at training phase. The network does not create similar CD losses compared to the results summarized in Figure A.4. This is because this network learnt on a smaller dataset (with objects having less points) than the one I have introduced in Chapter 2. The results met my expectations. From the experiment, I have attached two figures. One of the car in Figure 4.3 and one in Appendix A in Figure A.5.

	80 points	**400 points**	**800 points**
Car	**0.0949**	**0.0489**	**0.0407**
Toilet	**0.0844**	**0.0446**	**0.0373**

Table 4.1: This table summarizes the CD losses for the two categories I chose to show.

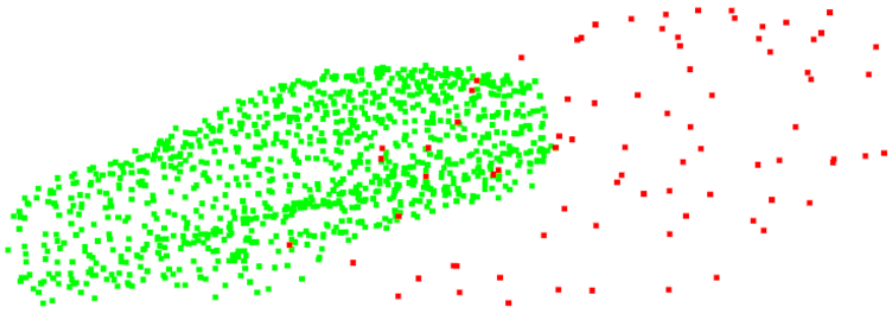

Figure 4.3: This Figure shows the 80point input for the car and generated point cloud from it. The red point cloud is the input and the green is the output.

4.2. Testing incomplete input from ModelNet40

In this task I have created partial inputs with the modified version of cut function. Before I cut, the input was 2048 points. The results of CD losses were: **0.2063** for the car, **0.1701** for the toilet and **0.1771** for an object with low complexity (table). These distances are higher than the distances measured in Section 4.1. Working with partial input is a harder task in general, the network shows no sign of creating a great coarse

for the model either. The partial inputs are included in Appendix A in Figures A.6, A.7 and A.8.

4.3. Testing on my dataset

From my dataset I chose to show 2 results which represent the whole behaviour of the network. My input was the sparse point cloud of the ship shown in Section 3.1. I chose not to have any offsets for the points. The input was 1024 points. I have attached the result of it in Figure 4.4. The CD loss was **0.5310**. As this is an unseen category and a complex structure the result fits my expectation. In the figure the result is sideways compared to the input.

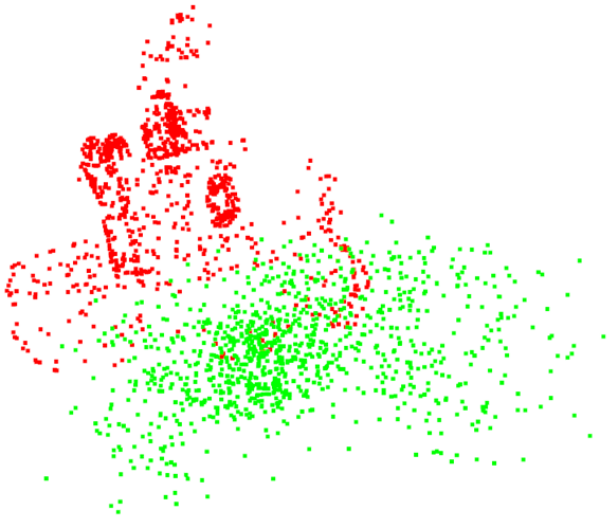

Figure 4.4: Figure of the input ship (red) and the reconstructed point cloud (green). The reconstructed point cloud is sideways.

For another test, I have chosen a general input, a partial part of a boat's hull. The CD loss was **0.5271**. The images of the input and the generated point cloud is attached in Appendix A in Figure A.9.
As these inputs were an unseen category and had bigger distances between each points the network failed to generalize. This behaviour might be caused because of the nature of this network, introduced in Figure A.1.

Chapter 5

Summary

5.1. Conclusion

From the proposed methods I have gathered, I have implemented the baseline model PCN. On this network I have tried two datasets, ModelNet40 and my own dataset. On the first dataset I have tested both sparse an partial inputs. For sparse inputs, the network did well, generated great dense point clouds. The outputs were great representations of the objects even when the input only had 80 points. I did the experiment on all categories, to showcase the results I chose 2 objects.

For incomplete inputs the model did worse. It had a hard time associating the input's structure with previously learnt knowledge. As this network is not able to define neighborhoods and smaller structural details, this behaviour was expected.

From my dataset I have tried 2 inputs. One sparse input from a boat and one partial input from a boat's hull. Neither of these did well on this network as they were unseen categories. However the boat's hull is a general shape, for which I have expected better results. These results show the limitations of this model.

5.2. Future plans

In the future, I plan to train a model using the ShapeNet dataset. Given the limitations of the current network, I want to experiment with different models, such as VRCNet or ECG. There are also newer networks that are not open source at the moment, but if they become available, I will implement them as well and apply them to real-world data. Additionally, I want to experiment with different techniques during the training phase, such as using data augmentation by applying various cuts to the point cloud.

Bibliography

[1] N. J. Beaudry and R. Renner, *An intuitive proof of the data processing inequality*, arXiv:1107.0740 [quant-ph], Sep. 2012. DOI: 10.48550/arXiv.1107.0740. [Online]. Available: http://arxiv.org/abs/1107.0740 (visited on 06/02/2024).

[2] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, *Image Super-Resolution via Iterative Refinement*, arXiv:2104.07636 [cs, eess], Jun. 2021. DOI: 10.48550/arXiv.2104.07636. [Online]. Available: http://arxiv.org/abs/2104.07636 (visited on 06/03/2024).

[3] Ahmadsabry, *A Perfect guide to Understand Encoder Decoders in Depth with Visuals*, en, Jun. 2023. [Online]. Available: https://medium.com/@ahmadsabry678/a-perfect-guide-to-understand-encoder-decoders-in-depth-with-visuals-30805c23659b (visited on 06/02/2024).

[4] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, *PCN: Point Completion Network*, en, arXiv:1808.00671 [cs], Sep. 2019. [Online]. Available: http://arxiv.org/abs/1808.00671 (visited on 06/02/2024).

[5] L. Pan, X. Chen, Z. Cai, *et al.*, "Variational Relational Point Completion Network," en, in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA: IEEE, Jun. 2021, pp. 8520–8529, ISBN: 978-1-66544-509-2. DOI: 10.1109/CVPR46437.2021.00842. [Online]. Available: https://ieeexplore.ieee.org/document/9577912/ (visited on 06/02/2024).

[6] L. Pan, "ECG: Edge-aware Point Cloud Completion with Graph Convolution," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4392–4398, Jul. 2020, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2020.2994483. [Online]. Available: https://ieeexplore.ieee.org/document/9093117/ (visited on 06/02/2024).

[7] *Papers with Code - ModelNet Dataset*, en. [Online]. Available: https://paperswithcode.com/dataset/modelnet (visited on 06/02/2024).

[8] A. X. Chang, T. Funkhouser, L. Guibas, *et al.*, *ShapeNet: An Information-Rich 3D Model Repository*, arXiv:1512.03012 [cs], Dec. 2015. DOI: 10.48550/arXiv.1512.03012. [Online]. Available: http://arxiv.org/abs/1512.03012 (visited on 06/02/2024).

[9] V. SaRoDe, *Vinits5/learning3d*, original-date: 2020-03-21T21:37:11Z, May 2024. [Online]. Available: https://github.com/vinits5/learning3d (visited on 06/02/2024).

Appendix A

Appendix

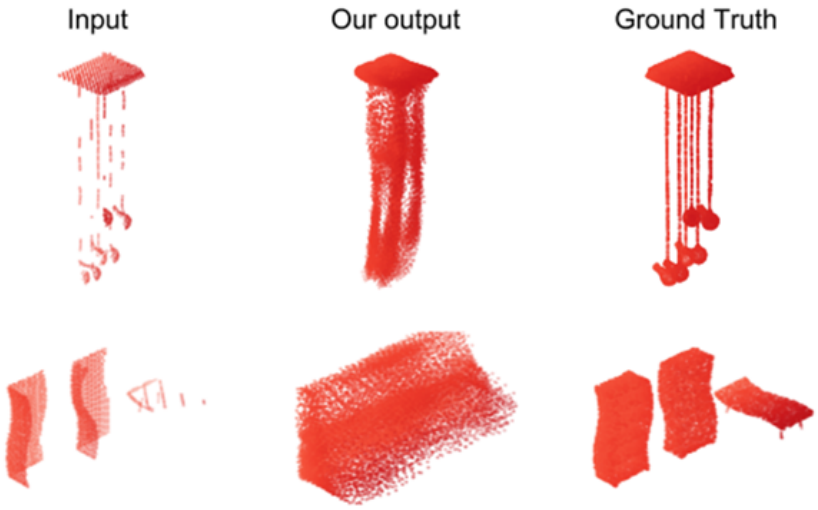

Figure A.1: Scenarios with multiple objects and thin structures. For these examples PCN has a hard time of reconstructing the original shapes. Source: [4]

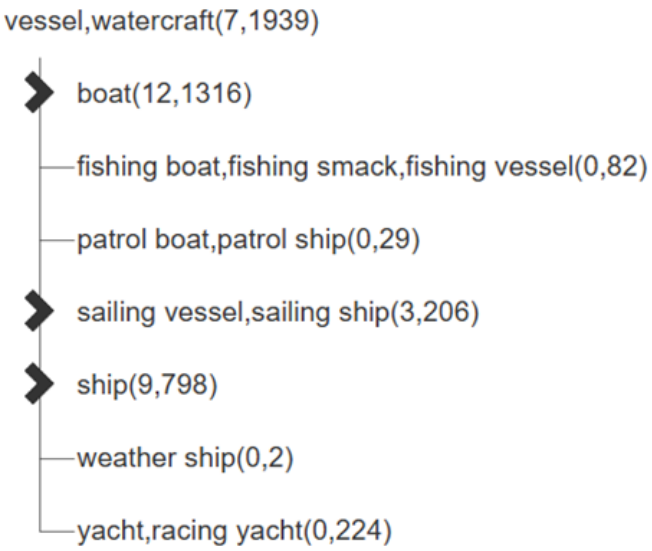

Figure A.2: This list shows the diversity of Shapenet's taxonomy of the common category watercraft. Source: [8]

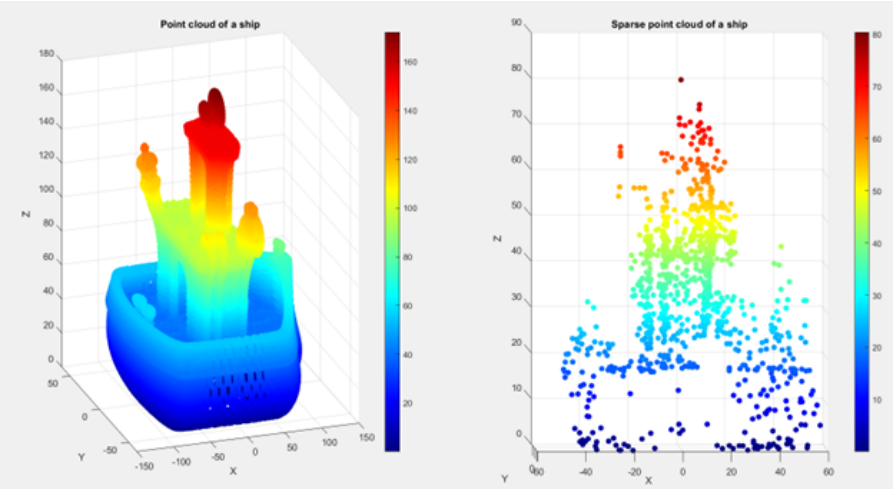

Figure A.3: On the **left** the original point cloud of the ship can be seen, while on the **right** the newly created sparse point cloud is shown with some points having an offset value.

Method	*airplane*	*cabinet*	*car*	*chair*	*lamp*	*sofa*	*table*	*watercraft*	*bed*	*bench*	*bookshelf*	*bus*	*guitar*	*motorbike*	*pistol*	*skateboard*	Avg.
PCN [30]	2.95	4.13	3.04	7.07	14.93	5.56	7.06	6.08	12.72	5.73	6.91	2.46	1.02	3.53	3.28	2.99	6.02
TopNet [21]	2.72	4.25	3.40	7.95	17.01	6.04	7.42	6.04	11.60	5.62	8.22	2.37	1.33	3.90	3.97	2.09	6.36
MSN [14]	2.07	3.82	2.76	6.21	12.72	4.74	5.32	4.80	9.93	3.89	5.85	2.12	0.69	2.48	2.91	1.58	4.90
Wang et. al. [23]	1.59	3.64	2.60	5.24	9.02	4.42	5.45	4.26	9.56	3.67	5.34	2.23	0.79	2.23	2.86	2.13	4.30
ECG [15]	1.41	3.44	2.36	4.58	6.95	3.81	4.27	3.38	7.46	3.10	4.82	1.99	0.59	2.05	2.31	1.66	3.58
GRNet [28]	1.61	4.66	3.10	4.72	5.66	4.61	4.85	3.53	7.82	2.96	4.58	2.97	1.28	2.24	2.11	1.61	3.87
NSFA [31]	1.51	4.24	2.75	4.68	6.04	4.29	4.84	3.02	7.93	3.87	5.99	2.21	0.78	1.73	2.04	2.14	3.77
VRCNet (Ours)	**1.15**	**3.20**	**2.14**	**3.58**	**5.57**	**3.58**	**4.17**	**2.47**	**6.90**	**2.76**	**3.45**	**1.78**	**0.59**	**1.52**	**1.83**	**1.57**	**3.06**

Figure A.4: This table compares results of different methods including PCN, ECG and VRCNet on different categories. The results are Chamfer Distances multiplied with a constant of 10^4. Source: [5]

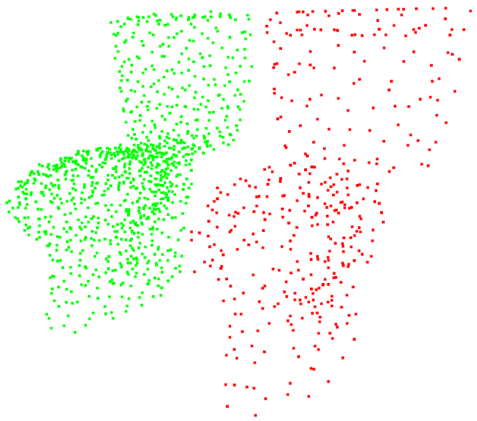

Figure A.5: This Figure shows the 400 points input for the toilet and generated point cloud from it. The red point cloud is the input and the green is the output.

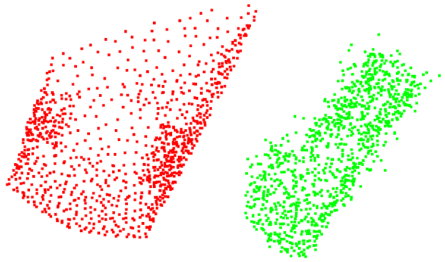

Figure A.6: Partial point cloud of a car. The red point cloud is the input and the green point cloud is the generated output.

Figure A.7: Partial point cloud of a toilet. The red point cloud is the input and the green point cloud is the generated output.

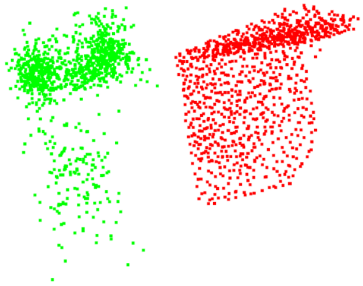

Figure A.8: Partial point cloud of a table. The red point cloud is the input and the green point cloud is the generated output.

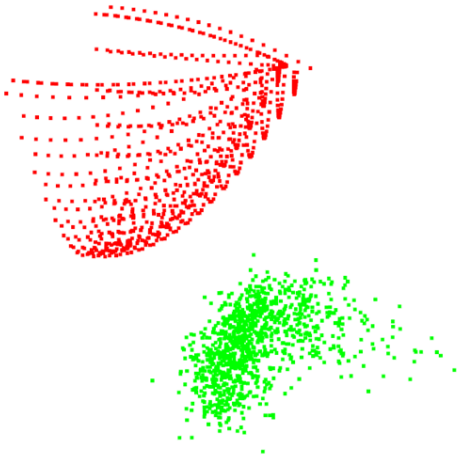

Figure A.9: Reconstruction of a partial input. The input is red and the generated point cloud is green. The green result is sideways.