DEEP LEARNING BASED POINT CLOUD PROCESSING AND COMPRESSION

A Dissertation
IN
Electrical and Computer Engineering
and
Telecommunication and Computer Networking

Presented to the Faculty of the University
of Missouri–Kansas City in partial fulfillment of
the requirements for the degree

DOCTOR OF PHILOSOPHY

by
ANIQUE AKHTAR

M.S., Koç University, Istanbul, Turkey, 2015
B.S., Lahore University of Management Sciences, Lahore, Pakistan, 2012

Kansas City, Missouri
2022

DEEP LEARNING BASED POINT CLOUD PROCESSING AND COMPRESSION

Anique Akhtar, Candidate for the Doctor of Philosophy Degree

University of Missouri–Kansas City, 2022

ABSTRACT

A point cloud is a 3D data representation that is becoming increasingly popular. Recent significant advances in 3D sensors and capturing techniques have led to a surge in the usage of 3D point clouds in virtual reality/augmented reality (VR/AR) content creation, as well as 3D sensing for robotics, smart cities, telepresence, and automated driving applications. With an increase in point cloud applications and improved capturing technologies, we now have high-resolution point clouds with millions of points per frame. However, due to the large size of a point cloud, efficient techniques for the transmission, compression, and processing of point cloud content are still widely sought.

This thesis addresses multiple issues in the transmission, compression, and processing pipeline for point cloud data. We employ deep learning solution to process 3D dense as well as sparse point cloud data for both static as well as dynamic contents. Employing deep learning on point cloud data which is inherently sparse is a challenging task. We propose multiple deep learning-based frameworks that address each of the following

problems:

1. **Point Cloud Compression Artifact Removal.** V-PCC is the current state-of-the-art for dynamic point cloud compression. However, at lower bitrates, there are unpleasant artifacts introduced by V-PCC. We propose a deep learning solution for V-PCC artifact removal by leveraging the direction of projection property in V-PCC to remove quantization noise.

2. **Point Cloud Geometry Prediction.** The current point cloud lossy compression and processing techniques suffer from quantization loss which results in a coarser sub-sampled representation of the point cloud. We solve the problem of points lost during voxelization by performing geometry prediction across spatial scale using deep learning architecture.

3. **Point Cloud Geometry Upsampling.** Loss of details and irregularities in point cloud geometry can occur during the capturing, processing, and compression pipeline. We present a novel geometry upsampling technique, PU-Dense, which can process a diverse set of point clouds including synthetic mesh-based point clouds, real-world high-resolution point clouds, real-world indoor LiDAR scanned objects, as well as outdoor dynamically acquired LiDAR-based point clouds.

4. **Dynamic Point Cloud Interpolation.** Dense photorealistic point clouds can depict

real-world dynamic objects in high resolution and with a high frame rate. Frame interpolation of such dynamic point clouds would enable the distribution, processing, and compression of such content. We also propose the first point cloud interpolation framework for photorealistic dynamic point clouds.

5. **Inter-frame Compression for Dynamic Point Clouds.** Efficient point cloud compression is essential for applications like virtual and mixed reality, autonomous driving, and cultural heritage. We propose a deep learning-based inter-frame encoding scheme for dynamic point cloud geometry compression.

In each case, our method achieves state-of-the-art results with significant improvement to the current technologies.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Graduate Studies, have examined a dissertation titled “Deep Learning Based Point Cloud Processing and Compression,” presented by Anique Akhtar, candidate for the Doctor of Philosophy degree, and hereby certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Zhu Li, Ph.D., Committee Chair
Department of Computer Science & Electrical Engineering

Sejun Song, Ph.D., Co-discipline Advisor
Department of Computer Science & Electrical Engineering

Cory Beard, Ph.D.
Department of Computer Science & Electrical Engineering

Masud Chowdhury, Ph.D.
Department of Computer Science & Electrical Engineering

Anthony Caruso, Ph.D.
Department of Physics & Astronomy

CONTENTS

ILLUSTRATIONS

TABLES

ACKNOWLEDGEMENTS

The past four years at UMKC have been an invaluable and memorable experience for me. When I first started my PhD at UMKC in 2018, I knew very little about multimedia, vision, and compression. Over the following years I have made remarkable progress and performed state-of-the-art research in point cloud communication, processing, and standardization. I have been very fortunate to witness the standardization process of next-generation point cloud coding standard, as well as participate in this process. I would have not been able to make this journey without the help and support of many people, to whom I feel deeply indebted.

I would like to express my sincere gratitude to my advisor, Dr. Zhu Li who has given me all the help from the absolute first day I began working under his supervision. His insightful input pushed me to hone my reasoning and carried my research to a higher level. Apart from my Ph.D. supervisor, I would like to express my gratitude to the rest of my Ph.D. committee members Dr. Sejun Song, Dr. Cory Beard, Dr. Masud Chowdhury, and Dr. Anthony Caruso for their advice, support, and guidance throughout my degree.

I would also like to thank my fellow lab mates Paras Maharjan, Zhaobin Zhang, Dewan Fahim Noor, Raghunath Sai Puttagunta, Birendra Katharia, Yangfun Sun, Li Li, Hongcheng Jiang, Wei Jia, Rijun Liao, Chris Henry, and Sajid Umair for the fun-time we spent together, for the sleepless nights that enabled us to complete tasks before deadlines, for stimulating the discussions, and for happy distractions to rest my mind outside of my research.

Finally, I would like to thank my family to be so supportive during my doctoral journey and providing me with guidance and help every step of the way.

CHAPTER 1

INTRODUCTION

Recent significant advances in 3D sensors and capturing techniques have led to a surge in the usage of 3D point clouds in virtual reality/augmented reality (VR/AR) content creation and communications, as well as 3D sensing for robotics, smart cities, telepresence [29], and automated driving applications [9]. A 3D point cloud can efficiently represent volumetric visual data such as 3D scenes and objects using a collection of discrete points with 3D geometry positions and other attributes (e.g., color, reflectance). Point cloud data offers advantages over polygonal meshes because it is more flexible and has real-time processing potential, since there is no need to process, store, or transfer surface topological information. With an increase in point cloud applications and improved capturing technologies, we now have high-resolution point clouds with millions of points per frame.

Based on their usage, point clouds can be categorized into point cloud scenes and point cloud objects. Point cloud scenes are dynamically acquired and are typically captured by LIDAR sensors. One example of a dynamic point cloud would be LIDAR sensors mounted atop a vehicle for mobile mapping and autonomous navigation purposes [45]. Point cloud objects can be further subdivided into static objects and dynamic objects. A static point cloud is a single object, whereas a dynamic point cloud is time-varying, where each instance of a dynamic point cloud is a static point cloud. Dynamic time-varying point clouds are used in AR/VR, volumetric video, and telepresence and can be generated using 3D models, i.e. CGI, or captured from real-world scenes using various methods such as multiple cameras with depth sensors surrounding the object and capturing movement over time.

A volumetric video such as a dynamic point cloud provides an immersive media experience. A dynamic point cloud describes a 3D object using its geometry, respective attributes, as well as any temporal changes. Temporal information in the dynamic point cloud is included in the form of individual capture instances, much like 2D video frames. A dynamic point cloud can be viewed from any angle or viewpoint because it includes a complete 3D scene. This six degrees-of-freedom (6DoF) [2] viewing capability makes the dynamic point cloud essential for any AR or VR application. A single instance of a dynamic point cloud captured by 8i [59] could contain as many as one million points.

Point cloud data is inherently sparse. point cloud scenes are usually captured by LiDAR which tend to be sparser than the volumetric point cloud objects. However, the size of all these categories of point clouds are still really large and is one of the main problems faced during the point cloud transmission and processing. Efficient techniques for the transmission, compression and processing of point cloud content are still widely sought.

In our work, we offer the following contributions in the compression, transmission and processing pipeline:

Point cloud compression artifact removal. V-PCC encoding is the current state-of-the-art method for dynamic point cloud compression that has been selected by MPEG to be developed into a standard. However, quantization noise during V-PCC encoding results in severe quality degradation because it introduces compression artifacts. V-PCC is based on the projection of the point cloud patches to 2D planes and encoding the sequence as 2D texture and geometry patch sequences. However, the resulting quantization errors from coding can introduce compression artifacts, which can be very unpleasant for the quality of experience (QoE). We developed a novel out-of-the-loop point cloud geometry artifact removal solution that can significantly improve reconstruction quality without additional bandwidth cost. Our novel framework consists of a point cloud sampling scheme, an artifact removal network, and an aggregation scheme. The point cloud sampling scheme employs a cube-based neighborhood patch extraction to divide the point cloud into patches. The geometry artifact removal network then processes these patches to obtain artifact-removed patches. The artifact-removed patches are then merged together using an aggregation scheme to obtain the final artifact-removed point cloud. We employ 3D deep convolutional feature learning for geometry artifact removal that jointly recovers both the quantization direction and the quantization noise level by exploiting projection and quantization prior. The simulation results demonstrate that the proposed method is highly effective and can considerably improve the quality of the reconstructed point cloud.

Point cloud geometry prediction. During compression and transmission, point cloud often suffer from quantization noise which results in lower Level-of-Detail (LoD) point clouds. We solve the problem of points lost during voxelization by performing geometry prediction across spatial scale using deep learning architecture. We perform an octree-type upsampling of point cloud geometry where each voxel point is divided into 8 sub-voxel points and their occupancy is predicted by our network. This way we obtain a denser representation of the point cloud while minimizing the losses with respect to the ground truth. Our results show that our geometry prediction scheme can significantly improve the PSNR of a point cloud, therefore, making it an essential post-processing scheme for the compression-transmission pipeline. This solution can serve as a crucial prediction tool across scale for point cloud compression, as well as display adaptation.

Point Cloud Upsampling. Loss of details and irregularities in point cloud geometry can occur during the capturing, processing, and compression pipeline. It is essential to address these challenges by being able to upsample a low Level-of-Detail (LoD) point cloud into a high LoD point cloud. Current upsampling methods suffer from several weaknesses in handling high-resolution large-scale photo-realistic point clouds. We present a novel geometry upsampling technique, PU-Dense, which is specifically designed for dense real-world high-resolution point clouds. PU-Dense employs a 3D multiscale architecture using sparse convolutional networks that hierarchically reconstruct an upsampled point cloud geometry via progressive rescaling and multiscale feature extraction. The framework employs UNet type architecture that downscales the point cloud to a bottleneck and then upscales it to a higher level-of-detail (LoD) point cloud. PU-Dense introduces a novel Feature Extraction Unit that incorporates multiscale spatial learning by employing filters at multiple sampling rates and field of views. Qualitative and quantitative experimental results show that our method significantly outperforms the state-of-the-art approaches by a large margin while having much lower inference time complexity. Where most work struggles to operate on real-world data, our method efficiently learns the underlying surface and recreates a high level of detail upsampled point cloud.

Dynamic Point Cloud Interpolation. Dense photorealistic point clouds can depict real-world dynamic objects in high resolution and with a high frame rate. Frame interpolation of such dynamic point clouds would enable the distribution, processing, and compression of such content. We propose a first point cloud interpolation framework for photorealistic dynamic point clouds. Given two consecutive dynamic point cloud frames, our framework aims to generate intermediate frame(s) between them. The proposed deep learning framework has three major components: the encoder module, the fusion network, and the multi-scale point cloud synthesis module. The encoder module extracts multi-scale features from two consecutive frames. The fusion network employs a novel 4D feature learning technique to merge the multi-scale features from consecutive frames. Finally, the multi-scale point cloud synthesis module hierarchically reconstructs the interpolated point cloud intermediate frame at different resolutions. We evaluate our framework on high-resolution point cloud datasets used in MPEG, JPEG Pleno, and AVS standards. The quantitative and qualitative results demonstrate the effectiveness of the proposed method.

Deep Learning-based Point Cloud Compression . Efficient point cloud compression is essential for applications like virtual and mixed reality, autonomous driving, and cultural heritage. We propose a deep learning-based inter-frame encoding scheme for dynamic point cloud geometry compression. We propose a lossy geometry compression scheme that predicts the latent representation of the current frame using the previous frame by employing a novel prediction network. Our proposed network utilizes sparse

convolutions with hierarchical multiscale 3D feature learning to encode the current frame using the previous frame. We employ *convolution on target coordinates* to map the latent representation of the previous frame to the downsampled coordinates of the current frame to predict the current frame's feature embedding. Our framework transmits the residual of the predicted features and the actual features by compressing them using a learned probabilistic factorized entropy model. At the receiver, the decoder hierarchically reconstructs the current frame by progressively rescaling the feature embedding. We compared our model to the state-of-the-art Video-based Point Cloud Compression (V-PCC) and Geometry-based Point Cloud Compression (G-PCC) schemes standardized by the Moving Picture Experts Group (MPEG). Our method achieves more than 91% BD-Rate (Bjøntegaard Delta Rate) reduction against G-PCC, more than 62% BD-Rate reduction against V-PCC intra-frame encoding mode, and more than 52% BD-Rate savings against V-PCC P-frame-based inter-frame encoding mode using HEVC.

CHAPTER 2

POINT CLOUD COMPRESSION ARTIFACT REMOVAL

2.1 Introduction

A 3D point cloud can efficiently represent volumetric visual data such as 3D scenes and objects using a collection of discrete points with 3D geometry positions and other attributes (e.g., color, reflectance). Point cloud data offers advantages over polygonal meshes because it is more flexible and has real-time processing potential, since there is no need to process, store, or transfer surface topological information. With an increase in point cloud applications and improved capturing technologies, we now have high-resolution point clouds with millions of points per frame.

A volumetric video such as a dynamic point cloud provides an immersive media experience. A dynamic point cloud describes a 3D object using its geometry, respective attributes, as well as any temporal changes. Temporal information in the dynamic point cloud is included in the form of individual capture instances, much like 2D video frames. A dynamic point cloud can be viewed from any angle or viewpoint because it includes a complete 3D scene. This six degrees-of-freedom (6DoF) [2] viewing capability makes the dynamic point cloud essential for any AR or VR application. A single instance of a dynamic point cloud captured by 8i [59] could contain as many as one million points. Approximately 30 bits are used to represent the geometry (x,y,z), and 24 bits represent the color (r,g,b). The size of a single instance can be approximated as 6 Mbytes, which translates to a bitrate of 180 Mbytes per second without compression for a 30-fps dynamic point cloud. The high data rate is one of the main problems faced by dynamic point clouds, and efficient compression technologies to allow for the distribution of such content are still widely sought.

The current state-of-the-art dynamic point cloud compression algorithm is the video-based point cloud compression (V-PCC) method [38] which has been selected and developed for standardization by MPEG for dynamic point clouds. Under the V-PCC standard, a point cloud is first projected onto its bounding box patch by patch. Then, the patches are packed into a video for compression. During the video compression, the reconstructed geometry may suffer severe quality degradation due to the quantization errors. Blocking artifacts or compression artifacts are often introduced in compressed media due to distortion which is introduced by lossy compression techniques [94]. The V-PCC coded point cloud yields excellent reproduction without noticeable artifacts at high or moderate bitrates. However, at low bitrates, the reconstructed point cloud suffers from visually annoying artifacts due to coarse quantization. Fig. 1 shows two versions of a point cloud encoded at different bitrates. As can be seen, there is no visible blocking artifact in the point cloud coded at a higher bitrate, while severe blocking artifacts exist in the one coded at a lower bitrate. Since blocking artifacts significantly degrade the visual quality of the reconstructed point cloud, it is desirable to identify these artifacts and remove them from the reconstructed point cloud.

This work proposes the first deep-learning-based geometry artifact removal algorithm for the V-PCC standard for dynamic point clouds. Ours is a pioneering work in

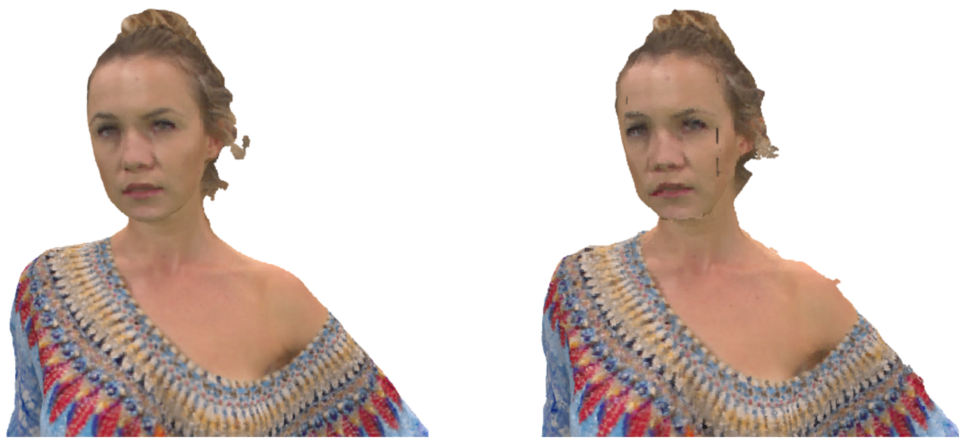

(a) Higher bitrate (b) Lower bitrate

Figure 1: Blocking effects in a point cloud coded at different bitrates using V-PCC encoding.

V-PCC artifact removal. The proposed framework offers the following contributions:

- We present a projection-aware 3D sparse convolutional neural network-based framework for point cloud artifact removal. Our sparse convolutional network learns an embedding and then regresses over this embedding to learn the quantization noise. Experimental results show that our method significantly improves the quality of the V-PCC reconstructed point cloud in terms of both objective evaluations and visual comparison.

- We observe that the geometry distortion of the V-PCC reconstructed point cloud exists only in the direction of the V-PCC projection. We exploit this prior knowledge to learn both the direction and level of quantization noise by limiting the degree of freedom of the learned noise. We employ Chamfer distance as our loss function and use MSE-PSNR as our quality evaluation metrics.

- We identify a *patch correspondence mismatch problem* that arises due to a difference in the number of points in the original geometry and the V-PCC reconstructed geometry. To solve this, we propose a sampling and aggregation scheme using a cube-centered neighbor search algorithm to find a better correspondence between the reconstructed geometry (after V-PCC encoding) and the original geometry (before V-PCC encoding). The sampling and aggregation scheme makes our method scalable to larger point clouds since the framework is not dependent on the number of points in a point cloud.

2.2 Background

In 2017, MPEG issued a call for proposals on Point Cloud Compression (PCC) to target an international standard for PCC [3]. As a result of this call, multiple proposals were submitted to MPEG. Since then, MPEG has been evaluating and improving the performances of the proposed technologies. MPEG has selected two technologies for PCC: Geometry-based PCC (G-PCC) [4] for static point cloud data as well as for dynamically acquired LIDAR point cloud data, and video-based point cloud compression (V-PCC) [5] for dynamic content. G-PCC employs octree in its coding scheme, whereas V-PCC projects point clouds onto 2D surfaces and then uses state-of-the-art HEVC video encoding to encode dynamic point clouds. However, V-PCC does introduce compression artifacts, primarily when encoded with a low bitrate.

To the best of our knowledge, compression artifact removal in V-PCC has not been studied so far. However, compression artifact removal techniques and deblocking have been extensively studied in image and video coding. Since V-PCC also employs state-of-the-art HEVC video coding, there is potential to learn from the video compression artifact removal techniques and use them for V-PCC artifact removal. The current state-of-the-art compression artifact removal techniques are based on deep learning. There have been several previous studies using residual networks [133] and GANs [30], as well as efforts employing memory-based deep learning architecture [101]. All of these works are limited to a single image and do not utilize information from previous frames. Recent works, however, have exploited temporal information in restoration tasks to improve video compression artifact removal [40, 72].

Point cloud deep learning has been attracting increasing attention, especially in the last five years [43]. PointNet [84] was among the earliest deep learning architectures for point cloud learning and employed pointwise fully connected layers followed by max pooling. This architecture was further improved into PointNet++ [85] by adding hierarchical learning that could learn local features with a better contextual scale. Wang et al. [116] proposed an octree-based CNN for 3D shape classification that performs 3D CNN operations on the octants of the octree data structure. PointCNN [67] achieved state-of-the-art results using a convolutional neural network on raw 3D point clouds. To perform convolution on raw point clouds, PointCNN makes the input permutation invariant by learning a transformation matrix using fully connected layers. SparseConvNet [36] by Facebook was among the first sparse convolutional neural networks that achieved state-of-the-art results on point clouds. SparseConvNet introduced submanifold sparse convolutions that exploited the sparse nature of point clouds and ensured that the convolutions would not “dilate” the data. Since sparse convolutions are memory-efficient and the data remain sparse throughout the network, deeper architectures can be used for point clouds. MinkowskiNet [20] is another such implementation that employs sparse convolutions for 3D point cloud learning. Recent works have also explored newer architectures for point cloud learning [110, 134].

The problem of point cloud denoising is an active research field which originated in the early 1990s. The works can be broadly divided into two categories: **optimization-based methods** and **deep-learning-based methods**. The optimization-based methods include techniques such as moving least squares (MLS)-based methods [42, 81], locally optimal projection (LOP)-based methods [68], sparsity-based methods [74, 94], non-local similarity-based methods [21], and graph-based methods [24, 32]. However, the current state-of-the-art solutions are all deep learning-based methods. PU-Net [129] employed deep learning to learn multi-level features for point cloud denoising. This work was further improved to EC-Net [127], which added edge-awareness to the network to further improve the results, especially around the edges. PU-GAN [64] utilized generative adversarial networks (GANs) with patch-based learning for point cloud denoising. PUGeo-Net [87] incorporated discrete differential geometry into deep learning to learn underlying geometric structures from given sparse point clouds. These methods work well for synthetic noise (e.g. Gaussian noise), and some have even been tested on real-world noise

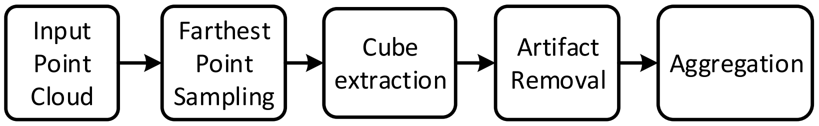


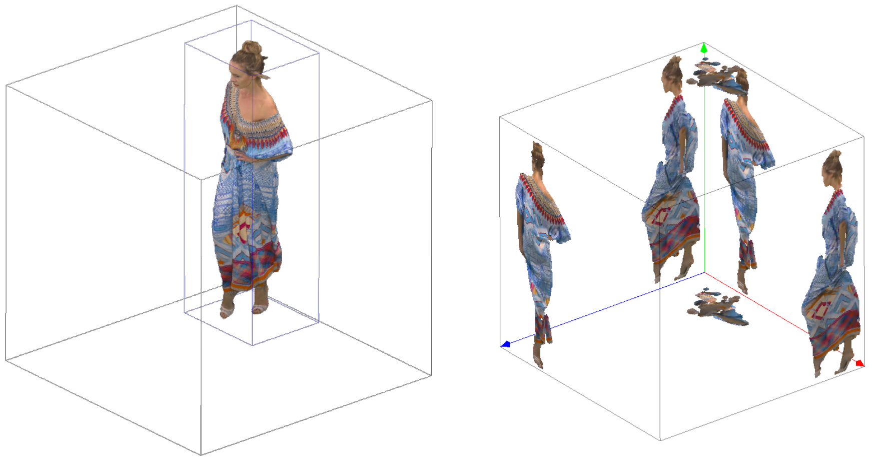
Figure 2: System Model.

introduced during point cloud capture. However, these methods are not optimized to work on compression artifact removal because of the nature of the quantization noise introduced during V-PCC.

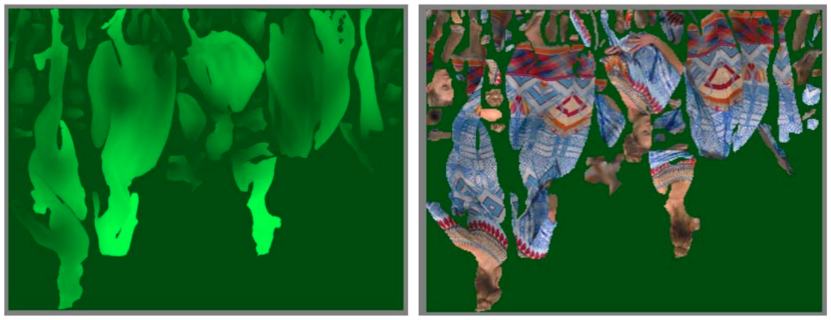
Similarly, some work focuses on point cloud inpainting [28, 48, 131], wherein portions of point clouds lost during point cloud capture are completed. However, these methods do not work for compression artifact removal in V-PCC. In the last year, there has been notable work performed on deep learning solutions for point cloud compression [52, 73, 108, 115]. However, these solutions are still immature, and the standardized V-PCC is still widely used. There has also been some work conducted with respect to the improvement of the V-PCC standard [62].

2.3 System Model

As described earlier, the quantization noise from V-PCC coding can result in compression artifacts that can cause considerable degradation to the quality of the reconstructed point cloud. Our goal is to perform deep learning-based geometry artifact removal. We make our framework scalable to larger point clouds. We propose a sampling and aggregation scheme in which the point cloud is divided into smaller patches, and then these patches are passed through a geometry compression artifact removal network.



(a) V-PCC projection onto ”bounded box” planes. Image from [1]



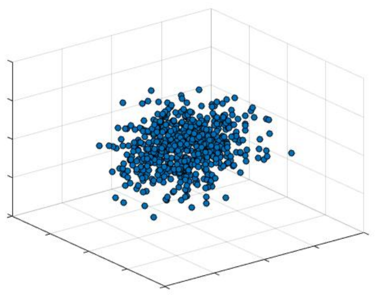
(b) V-PCC patch packing onto a 2D grid. Example of geometry (left) and texture (right) images. Image from [97].

Figure 3: V-PCC: Example of 3D-to-2D projection. Although the figures show both geometry and texture information, in our work we are only concerned with geometry artifact removal.

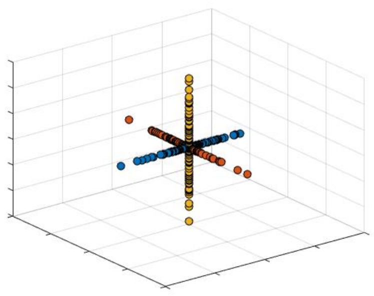
Afterward, we combine the artifact-removed patches to form an artifact-removed point cloud. The system model is shown in Fig. 2. Taking advantage of the sparse nature of point clouds, we employ the sparse convolutional network [36] which uses submanifold sparse convolutions [37] for our 3D learning.

2.3.1 Problem Formulation

V-PCC attempts to leverage existing video compression codecs for point cloud geometry and texture compression. V-PCC converts the point cloud into a set of different video sequences, one for geometry and one for texture information. In our work, we are only concerned with the noise introduced in the geometry compression. The video-generated bitstreams and the metadata needed to interpret these videos are then multiplexed together to generate the final point cloud V-PCC bitstream. V-PCC maps the input point cloud to a regular 2D grid by first decomposing the point cloud into a set of patches and then mapping these patches independently to a 2D grid using orthogonal projection. This process is shown in Fig. 3a. V-PCC iteratively divides the point cloud into smaller patches to avoid auto-occlusions and to generate patches with smooth boundaries. To generate these patches, the normal for each point is first estimated. An initial clustering is obtained by associating each point to one of the six cube-oriented planes. More precisely, each point is associated with the plane that has the closest normal (i.e., the dot product of the point normal and the plane normal is maximum). This initial clustering is then refined by iteratively updating the cluster index by taking into account the point's normal and the neighboring point's cluster index. In this way, all the points in a refined patch are associated with a single plane. The majority of the points in the point cloud are projected to the cube plane that is closest to the normal of that point. This projection is only along one of the axes $(x, y,$ or $z)$. These patches are then projected onto the 2D grid using a process called packing to obtain a frame with texture and another with geometry. The final video sequence frames for geometry and texture are shown in Fig. 3b.



(a) Gaussian Noise



(b) V-PCC Quantization Noise

Figure 4: Comparison of two different types of noises in a point cloud.

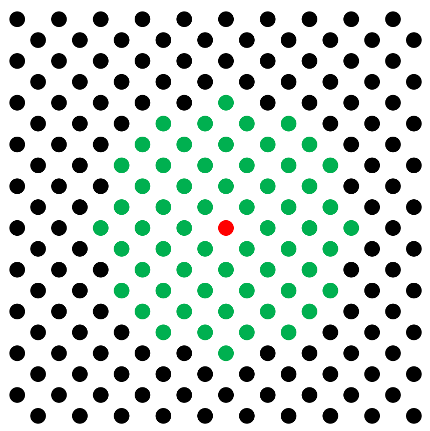
Afterward, these projected video frames are encoded by leveraging video compression techniques. Since these compression techniques are lossy, compression artifacts are often introduced due to quantization noise affecting the point cloud geometry. However, the artifact noise introduced in V-PCC geometry is only in one direction, as shown in Fig. 4. This is because each point is projected to only one plane, and therefore the artifact noise in that point is only in the direction of that plane. ***This means that quantization noise introduced in each point is only along one of the axes (x, y, or z)***. We leverage this property to learn the quantization noise level and quantization noise direction introduced by the V-PCC codec in the point cloud geometry. Since the quantization noise is along one of the axes, we make sure that our learned quantization noise for each point is also along a single axis. We exploit this prior knowledge of quantization noise direction to limit the degree of freedom of the learned quantization noise. We use the learned quantization noise to remove geometry artifacts from the reconstructed point cloud and improve its PSNR.

2.3.2 Sampling

The size of a point cloud can vary greatly, from point clouds with only a few thousand points to point clouds with millions of points. We propose a patch-based sampling and aggregation scheme to make our framework scalable to all sizes of point clouds. We sample a large point cloud into smaller neighborhood patches to ensure efficiency for practical application by offering affordable memory consumption on a cube basis, along with parallel processing.

For each extracted patch, we need to find the patch in the original/ground-truth point cloud and the corresponding patch in the V-PCC reconstructed/noisy point cloud. A lossy low bitrate V-PCC reconstructed point cloud usually has fewer points than the original point cloud, consequently making the reconstructed point cloud sparser. Traditional patch-based point cloud deep learning models employ k-nearest neighbor (k-NN) search algorithms to obtain patches. However, when the number of points in the reconstructed point cloud differs from the number of points in the original point cloud, the k-NN search for extracting a neighborhood patch would not work because it would result in a different patch surface area for the two point clouds. Consider the example of $k = 61$: using k-NN to extract a patch of 61 points would occupy a much larger area in a sparser point cloud as compared with a dense point cloud. We call this the *patch correspondence mismatch problem* and show an example of it in Fig. 5.

To solve the *patch correspondence mismatch problem*, we propose a cube-centered neighborhood search algorithm wherein we extract all the points inside a fixed cube volume. We employ farthest point sampling (FPS) [26] to sample points on the


(a) Dense surface

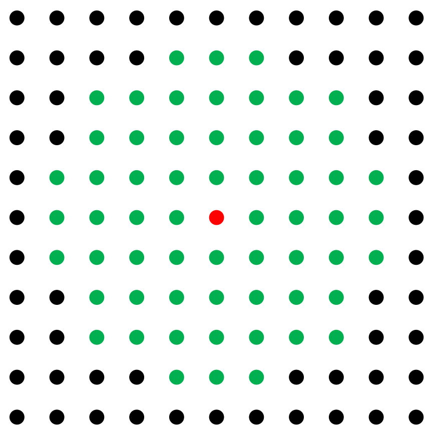

(b) Sparse surface

Figure 5: Patch correspondence mismatch problem for $k = 61$. The k-NN search covers a smaller surface on a denser point cloud as compared to a sparser point cloud.

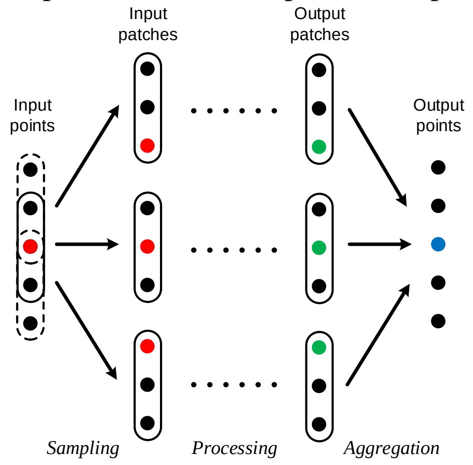


Figure 6: An example sampling and aggregation scheme. The red point is the input point that is sampled into three patches. These three patches are processed to obtain three processed green points. The three green points are then aggregated to form the blue output point.

noisy point cloud and then extract cube patches around the sampled points. We obtain the noisy point cloud patch and the associated ground truth patch of the same cube volume extracted from the same location from both point clouds.

Farthest point sampling is employed to sample N points over the point cloud, and a cube neighborhood around these points is used to extract neighborhood patches. N patches are extracted using the formula:

$$N = \frac{n * C}{k} \tag{2.1}$$

where n represents the total number of points in the point cloud, k is the approximate average number of points in the neighborhood patch, and C is a variable used to control the average number of overlapping patches per point. More points can be sampled by increasing the value of C, which would result in a larger average number of overlapping patches per point. Each of the sampled points is used as a center point for a cube and all the points inside the cube are extracted to form an input neighborhood patch. The geometry of the points inside the patch is zero-centered and then normalized between zero and the length of the cube side. These smaller input patches are fed to our 3D U-Net architecture as shown in Fig. 7. Depending on the value of C, each point is sampled into multiple input patches and processed to obtain output patches. Therefore, we obtain multiple processed points for each point. We employ an aggregation scheme to merge the output patches to obtain the final output point cloud. An example of this can be found in Fig. 6.

2.3.3 Aggregation

Once we have the artifact-removed output patches, we aggregate them back together to form the final point cloud. Post-processing is performed on the output patches

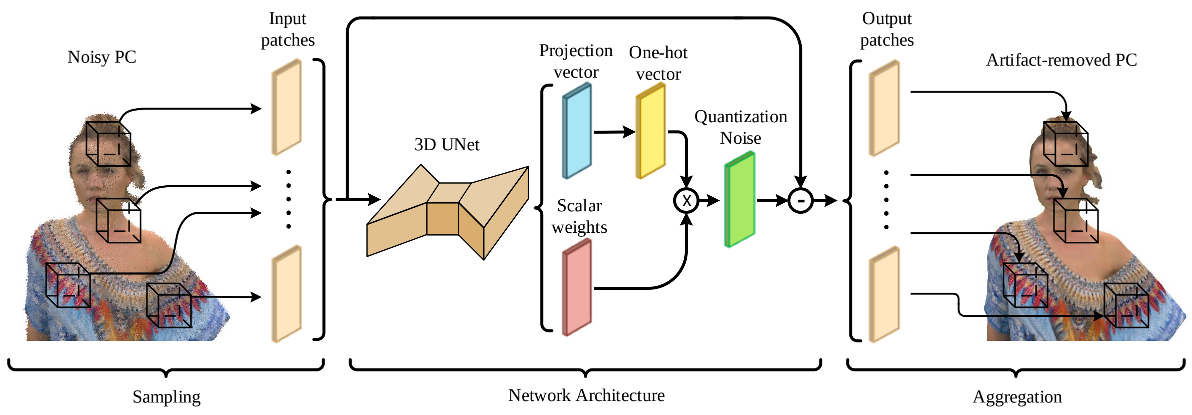


Figure 7: Overview of the proposed point cloud artifact removal scheme. The input point cloud is divided into smaller patches that are fed into a sparse U-Net, which produces the projection vector and the scalar weights for each patch. The projection vector and the scalar weights are used to calculate the quantization noise that is then removed from the reconstructed point cloud patch. The output patches are then aggregated to obtain the artifact-removed point cloud.

for which the normalization is removed, and they are moved back to their original locations as shown in Fig. 7. Depending on the value of C, we obtain many overlapping patches: therefore, each point receives multiple geometry values from different patches. Each input point is sampled into multiple patches, resulting in multiple clean point outputs for each input point. The geometries of these clean output points are mean aggregated to obtain the final clean output point.

One example sampling and aggregation scheme is shown in Fig. 6. The number of input points is $n = 5$, the number of patches sampled is $N = 3$, and each patch is of size $k = 3$. For reference, we can examine the red input point. This input point is sampled into three patches and processed to obtain three processed green points. We then take the mean of the three green processed points to obtain the blue aggregated output point.

2.3.4 Network Architecture

As explained in the previous section, we employ a cube-based patch sampling algorithm, so the number of points in the input patch varies. Traditional deep-learning-based point cloud processing networks work on a fixed number of input points to the network. We propose a fully convolutional 3D network that functions for variable input patch size. We employ sparse convolutional networks to create a fully convolutional network that offers the advantage of using a different number of points per patch, making the cube neighborhood patch extraction viable. Recall that our goal is to take in a 3D input patch of a V-PCC reconstructed point cloud and learn the per point quantization noise. The output of our 3D deep learning architecture is per point scalar weights and a per point projection vector. We use these to calculate the quantization noise.

3D U-Net. We employ submanifold sparse convolutions [37] that use a 3D sparse convolutional network [36]. Sparse convolutions exploit the sparse nature of a point cloud and are much more memory efficient. Furthermore, sparse submanifold convolutions make sure the network does not "dilate" the sparse data and maintain the same sparsity level throughout the network. This helps us build and train deeper architectures like U-Net [93].

U-Net architecture has been widely used in biomedical image segmentation tasks and usually employs 2D convolutions. We implemented a sparse convolution-based 3D U-Net architecture, the details of which are shown in Fig. 8. The architecture takes in a

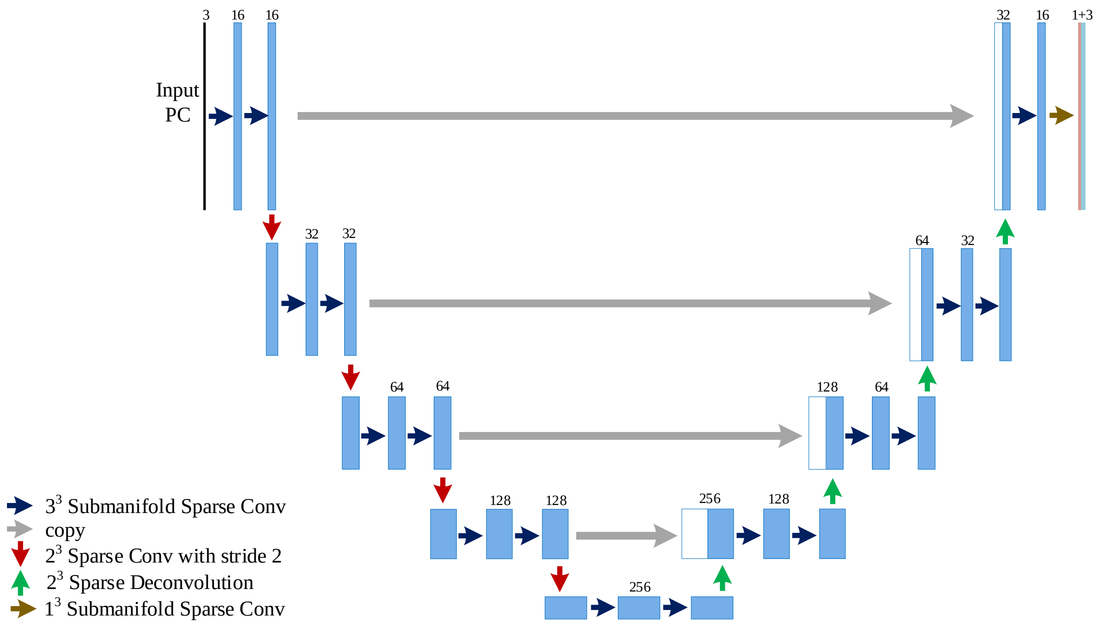

3
16
16
Input PC
32
32
64
64
128
128
256
256
128
128
64
64
32
32
16
1+3
3^3 Submanifold Sparse Conv
copy
2^3 Sparse Conv with stride 2
2^3 Sparse Deconvolution
1^3 Submanifold Sparse Conv

Figure 8: 3D U-Net Network Architecture.

3-dimensional geometry input patch, and the output is 4-dimensional: 3 dimensions for the projection vector and 1 dimension as a scalar weight. We use 3x3x3 (3^3) submanifold sparse convolutions in each layer. We employ 2x2x2 (2^3) convolutions with a stride of 2 for each *downsampling*, whereas 2x2x2 (2^3) deconvolution is used for *upsampling*. The U-Net architecture typically consists of two paths: the encoder path and the decoder path. The encoder path captures the context of the point cloud, producing feature maps using strided convolutions to downsample. In the encoder path, with each downsampling, the number of points decreases but the feature dimension is doubled. The decoder path employs 3D deconvolution to upsample the point cloud. U-Net combines the information from the encoder path with that of the decoder path to obtain both the contextual information and localized information. U-Net only contains convolutional layers and does not employ any dense layers, making the network fully convolutional. This offers the added advantage of working with patches of variable input size.

Quantization Noise Calculation. The projection vector yields the direction of the quantization noise, whereas the scalar weight provides the level of the quantization noise. In V-PCC, a point is typically projected in one direction (x, y, or z); therefore, the quantization noise for each point is in a specific direction. Hence, it makes more sense to use a one-hot encoded projection vector for each point. We convert the projection vector into a one-hot vector by performing one-hot encoding using the maximum of the projection vector:

$$Z_i(j) = \begin{cases} 1 & \text{if } j = \underset{j}{\operatorname{argmax}} V_i(j) \\ 0 & \text{else} \end{cases} \tag{2.2}$$

Where i is the point number, j is the dimension, i.e. $j \in \{x, y, z\}$-axis, and $Z_i(j)$ is the one-hot vector, whereas $V_i(j)$ is the projection vector.

Once we have the one-hot vector, we multiply it with the scalar weight to learn the quantization noise. The per point quantization noise is then removed from the input patch to obtain an artifact-removed output patch. This is also illustrated in Fig. 7.

Loss Function. Our loss function is calculated by comparing the artifact-removed output patch to the ground truth patch. The loss function is applied to each patch before aggregation. We use Chamfer distance as the loss function in our architecture:

$$L_{CD}(P_O, P_G) = \sum_{x \in P_O} \min_{y \in P_G} ||x - y||_2^2 + \sum_{y \in P_G} \min_{x \in P_O} ||x - y||_2^2 \tag{2.3}$$

Where L_{CD} is the Chamfer distance loss function, P_G is the ground truth patch and P_O is the output artifact-removed patch calculated using input patch, projection vector, and scalar weights. Intuitively, the first term measures an approximate distance between each output point to the target surface, whereas the second term rewards even coverage of the output point cloud and penalizes any gaps.

2.4 Simulation Results

We perform extensive simulations and show both the objective results and the visual comparison of our framework. Since this is the first work on V-PCC artifact removal, we have nothing with which to compare our work. However, we do show considerable improvement in the quality of the point cloud and perform multiple ablation studies to gain insight into the problem and explore alternative methods. For our simulation, we use the values of $k = 10000$ and $C = 20$.

2.4.1 Dataset

We use the 8i voxelized full bodies dataset from 8i labs [59], which contains up to one million points per point cloud and is widely used by MPEG. The 8i dataset includes multiple sequences of point clouds. Each sequence has multiple point clouds representing 10 seconds of video captured at 30 fps for a total of 300 frames. We use two sequences for training (*longdress, loot*) and three sequences for testing (*redandblack, soldier, queen*).

We use three different bitrates to encode these point clouds using V-PCC, shown in Table 1. We label these bitrates as $br1$, $br2$, and $br3$ from the highest bitrate to the lowest bitrate, respectively.

2.4.2 Objective Evaluation

We use mean-squared-error (MSE) point-to-point PSNR (dB) as well as point-to-point Hausdorff PSNR (dB) as our objective metrics, which are calculated using MPEG's PC error tool [106]. We refer to MSE PSNR as simply PSNR in the rest of the section. We also measure the BD-Rate (Bjöntegaard Delta Rate) [15] improvement to determine how much savings our method achieves. The PSNR of the reconstructed point clouds obtained from the V-PCC encoder is measured before and after our artifact removal technique. The results are shown in Table 2. As can be determined for all three bitrates, our artifact removal technique considerably improves the PSNR as well as Hausdorff PSNR of the reconstructed point cloud.

We follow the MPEG common test condition to calculate the BD-rate using the PSNR metric. We compute the point-to-point distance for each frame of a sequence and obtain a total score by averaging across all frames. The final objective score was obtained by averaging across all three test sequences. We obtain a BD-rate savings of 11.3%. We also show the PSNR results for each 8i point cloud sequence separately for each bitrate in Table 3. We can also observe PSNR improvement for the bitrates for each point cloud sequence. From the results, we observe that higher PSNR improvement is achieved at a lower bitrate. This is because lower bitrates suffer from higher quantization noise, which our system can efficiently remove.

Table 1: Different bitrates used in simulation

Bitrate label	Actual bitrate
br1	0.01866 bpp
br2	0.01632 bpp
br3	0.01502 bpp

Table 2: Average PSNR results tested on three 8i sequences

	PSNR (dB)		Hausdorff PSNR (dB)	
Bitrate	**Noisy PC**	**Cleaned PC**	**Noisy PC**	**Cleaned PC**
br3	59.62	60.47	37.02	37.21
br2	61.84	62.36	39.58	39.64
br1	64.20	64.53	41.15	41.20
BD-rate savings:		11.3 %		

2.4.3 Visual Results

Visual results for point cloud artifact removal are shown in Fig. 9 and Fig. 10 for two different sequences: *queen* and *soldier*. We show the original (ground truth) point cloud, V-PCC reconstructed point cloud, and the artifact-removed point cloud for three different bitrates of V-PCC encoding. To visualize the point cloud, we first compute the normals for each point using 100 neighboring points; then, we set the shading to vertical and view the point cloud as a mesh. In this way, we can observe the point cloud geometry, which is more intuitive than vertex-color rendered images. However, since we use normals to visualize the point cloud, some points might appear black due to flipped

Table 3: Simulation Results For Each Individual Sequence

		PSNR (dB)		
Test PC	**Bitrate**	**Noisy PC**	**Cleaned PC**	**Improvement**
Queen	br3	60.23	60.79	0.56
	br2	62.35	62.90	0.55
	br1	64.94	65.21	0.27
RedAndBlack	br3	59.44	60.38	0.94
	br2	61.62	62.18	0.56
	br1	63.90	64.27	0.37
Soldier	br3	59.20	60.25	1.05
	br2	61.53	62.01	0.48
	br1	63.76	64.12	0.36

normals, as shown from the right foot in Fig. 9. We also plot the error map based on the point-to-point (P2point) D1 distance between decoded point clouds and ground truth to visualize the error distribution.

We can see that our method improves the quality of the point cloud, especially on the edges and surface of the point cloud. Although V-PCC performs well in quantitative objective comparison, its reconstructed point clouds contain noticeable artifacts when the bitrate is low. Our method removes these artifacts and considerably improves the visual quality of the point cloud. An interesting observation is that our artifact-removed point cloud compensates for some broken parts in the V-PCC reconstructed point cloud.

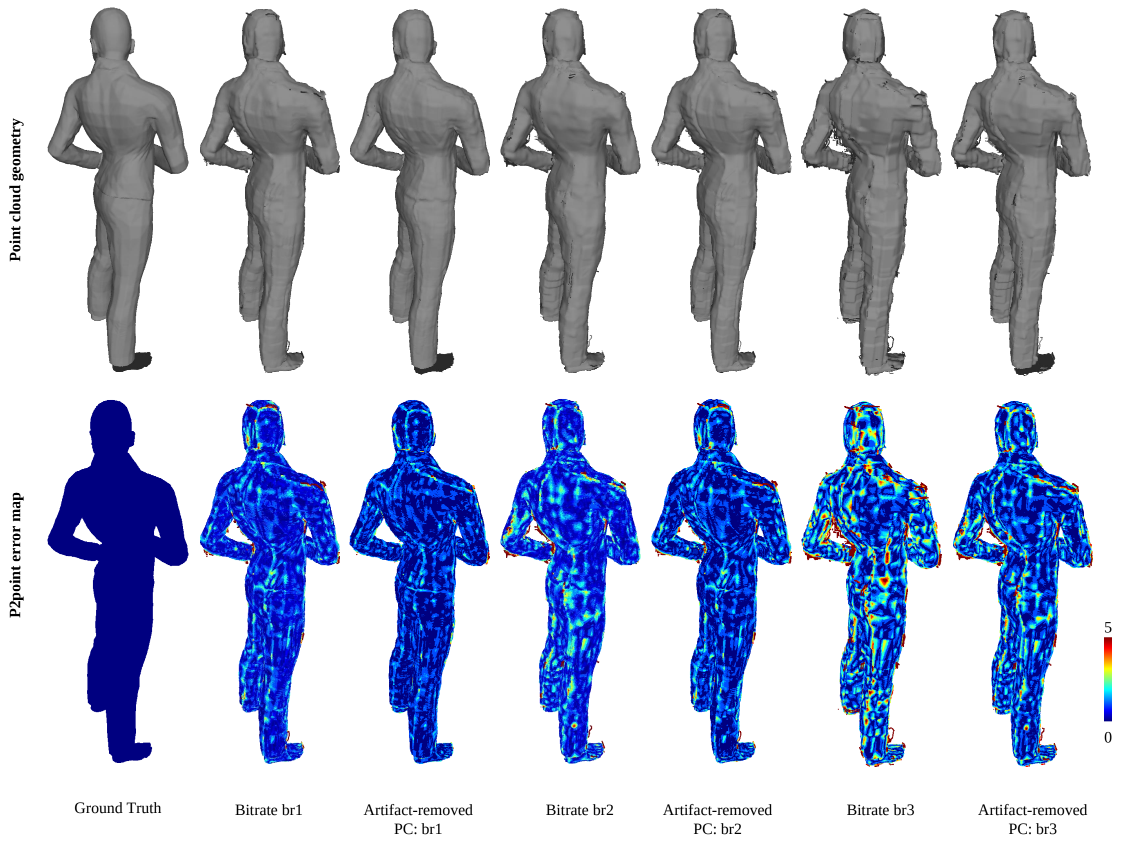


Figure 9: Visual comparison of 'queen' showing ground truth, three different V-PCC reconstructed point clouds, and their corresponding artifact-removed point cloud. We show the geometry as well as point-to-point error map.

2.4.4 Quantization Noise Calculation

In this section, we compare our quantization noise calculation method with alternative methods. We use the V-PCC encoded bitrate of $br3$ for this experiment. Our current structure outputs 1-dimensional scalar weights and a 3-dimensional projection vector. We convert the projection vector to a one-hot encoded vector and multiply it by the scalar weights to calculate the quantization noise. After removing the quantization

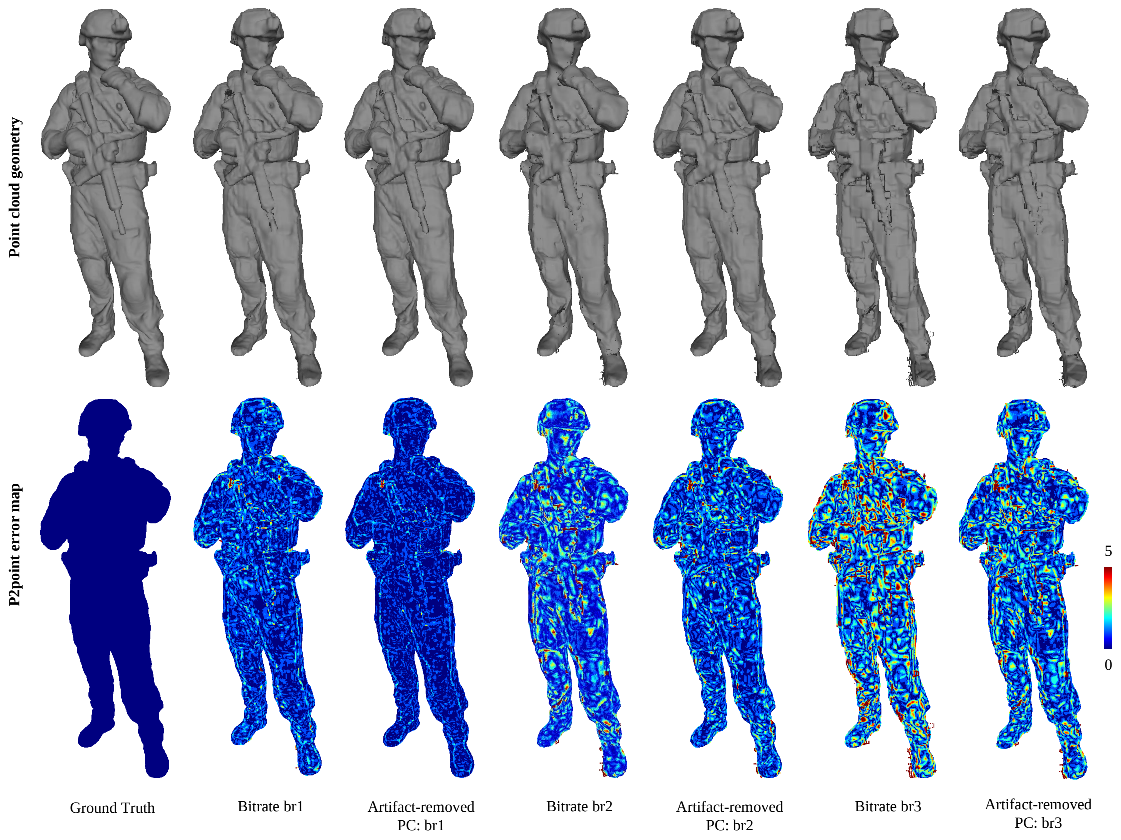


Figure 10: Visual comparison of 'soldier' showing ground truth, three different V-PCC reconstructed point clouds, and their corresponding artifact-removed point clouds. We show the geometry as well as point-to-point error map.

noise from the input patch, Chamfer loss is used to train the network. We label this **Our Method** and compare it with two alternative methods. **Method 1:** The network outputs 3-dimensional quantization noise that is directly removed from the input patch without any post-processing, and then Chamfer loss is used to train the network. **Method 2:** The network outputs 1-dimensional scalar weights and a 3-dimensional projection vector. Projection vectors are directly multiplied by the scalar weights to find quantization noise without converting them to a one-hot vector first. After removing the quantization noise from the input patch, Chamfer loss is used to train the network.

To summarize, in *Method 1* the network directly outputs the quantization noise, whereas in *Method 2* we remove the one-hot encoding part from our original architecture. The results of *Our Method*, *Method 1*, and *Method 2* are compared in Table 4.

Table 4: Different quantization noise calculation methods

		PSNR (dB)		
Test PC	**Noisy PC**	**Our Method**	**Method 1**	**Method 2**
Queen	60.23	**60.79**	60.35	60.41
RedAndBlack	59.44	**60.38**	59.92	60.02
Soldier	59.20	**60.25**	59.76	59.93
Average	59.62	**60.47**	60.01	60.12

The results of Method 1 show that learning quantization noise directly from the network yields poor results. Similarly, comparison of our method with Method 2 shows that converting the projection vector to a one-hot vector before calculating the quantization noise considerably improves our results. This also shows that utilizing the prior knowledge of quantization noise which is only in the direction of the projection is beneficial to the learning of quantization noise.

2.4.5 Sampling and Aggregation Schemes

Currently, we employ the Farthest Point Sampling technique to sample points and extract a neighborhood around these points using cube extraction. Since there are overlapping neighborhood patches, we perform a mean aggregation scheme to obtain the final artifact-removed point cloud. We compare our method to a non-overlapping **octree-based** [8] cube division method. In the octree-based method, the point cloud is partitioned into cube nodes, and artifact removal is applied to each node. The differences between our method and the octree-based method are: 1) the patch sampling is performed using an octree; 2) there is no aggregation scheme, since the sampled patches are non-overlapping. The results of the comparison are shown in Table 5. We can observe from the results that our overlapping cubes-based sampling method substantially outperforms the octree-based sampling method.

Table 5: Different sampling and aggregation schemes

		PSNR (dB)	
Test PC	**Noisy PC**	**Our Method**	**Octree-based**
Queen	60.23	**60.79**	60.38
RedAndBlack	59.44	**60.38**	59.97
Soldier	59.20	**60.25**	59.80
Average	59.62	**60.47**	60.05

2.4.6 Choosing the Value of C

As described earlier, C is the variable used to control the average number of overlapping patches per point. A higher value of C would result in a larger number of randomly sampled patches. To further study our sampling scheme, we perform a hyper-parameter optimization experiment for the parameter C. We perform the simulation on the three test sequences of our 8i dataset (*redandblack, soldier, queen*) and then plot the combined results. We vary the value of C and measure the PSNR results as well as the

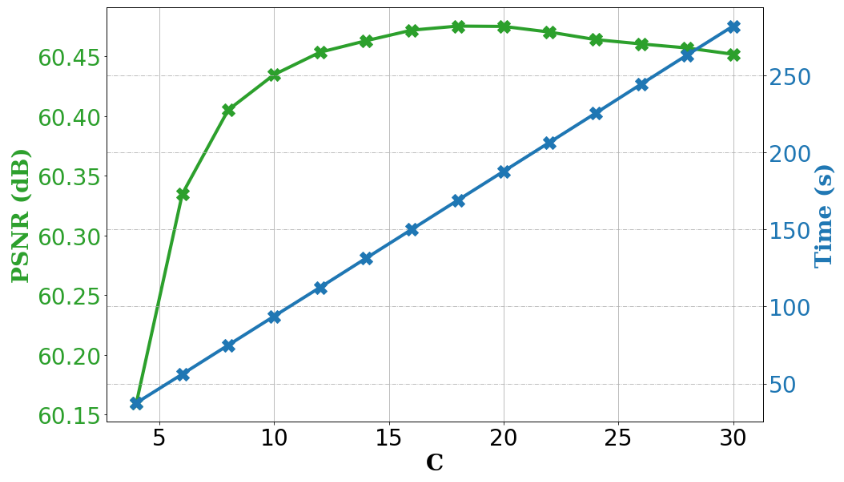


Figure 11: PSNR (dB) and Time (s) complexity for different values of C.

computation time. Results of this experiment are shown in Fig. 11. As can be observed, the PSNR increases with the value of C: PSNR is maximal at $C = 18$ and starts to slightly decrease after that. The computation time is calculated as the average time to process a single 8i point cloud on an NVIDIA GeForce GTX 1080 Ti GPU. The computation time includes sampling, forward propagation through the network, and aggregation. As Fig. 11 shows, the computation time increases linearly with the value of C.

2.5 Conclusion

V-PCC encoding is the current state-of-the-art method for dynamic point cloud compression that has been selected by MPEG to be developed into a standard. However, quantization noise during V-PCC encoding results in severe quality degradation because it introduces compression artifacts. This work presents a first-of-its-kind deep learning-based point cloud geometry compression artifact removal framework for V-PCC encoded dynamic point clouds. We leverage the prior knowledge that during V-PCC, the quantization noise is introduced only in the direction of the point cloud projection. We employ a 3D sparse convolutional neural network to learn both the direction and the magnitude of geometry quantization noise. To make our work scalable, we propose a cube-centered neighborhood extraction scheme with a sampling and aggregation method to extract small neighborhood patches from the original point cloud. These patches are passed through the network to remove compression artifacts and then aggregated to form the final artifact-removed point cloud. Experimental results show that our method considerably improves the V-PCC reconstructed point cloud's geometry quality in both objective evaluations and visual comparisons.

CHAPTER 3

POINT CLOUD GEOMETRY PREDICTION

3.1 Introduction

Point clouds are being readily used in augmented and virtual reality experiences, as well as 3D sensing for smart cities, robotics, and automated driving applications [9]. Therefore, point cloud capturing, transmission, and processing are essential for these use cases. However, point cloud representation requires a large amount of data which is not always feasible for transmission. Efficient compression technologies are in high demand to make point cloud transmission, storage, and processing more proficient [8]. Therefore, in 2017 MPEG issued a call for proposals on Point Cloud Compression (PCC), and since then MPEG has been evaluating and improving the performances of the proposed technologies [97].

For natural captured 3D sensor signals, scene geometry needs an efficient representation that is scalable in Level-of-Detail (LoD) as well as efficient in compression. MPEG has selected two technologies for PCC: Geometry-based PCC (G-PCC) for dynamically acquired LiDAR point cloud data and for static point cloud data, and video-based point cloud compression (V-PCC) for dynamic content [97]. G-PCC employs octree in its coding scheme, whereas, V-PCC projects point cloud into 2D cube surfaces and then uses state-of-the-art HEVC video encoding to encode dynamic point clouds. Octree has been widely used in processing as well as compression of point clouds [95, 135]. In

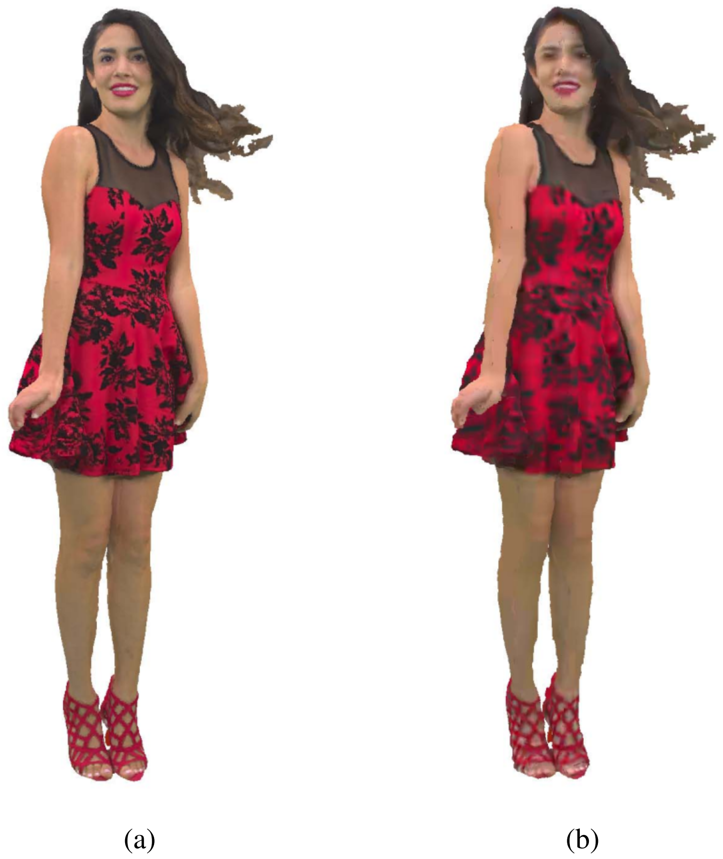


Figure 12: (a) Original (uncompressed) point cloud, (b) Reconstructed point cloud suffering from quantization noise.

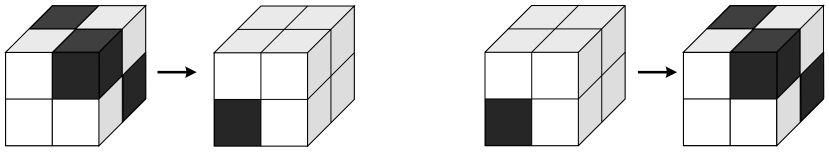
Octree a node is subdivided into eight child-nodes and the occupancy of each child-node is decided by whether it has points or not. A linear model based PCC approach has been proposed in [136].

Deep learning for point cloud solutions have also matured with PointNet [84] among the earlier works utilizing fully connected layers. This work was further extended into PointNet++ [85] by introducing hierarchical feature learning. Octree based voxelized deep learning solutions have also been proposed that remained state-of-the-art in the past

[116]. Recently, sparse tensors and sparse convolutions have been explored in point cloud deep learning [37]. Sparse convolution leverages the inherent sparsity of point cloud which makes them memory efficient and enables deeper architecture to be built for point cloud learning. Submanifold sparse convolutional network [36] was the first to use sparse convolutions followed by Minkowski Engine [20].

The current compression and transmission schemes often suffer from quantization noise resulting in a lower LoD reconstructed point cloud as shown in Fig. 12. Due to quantization, the neighboring points in a voxelized point cloud are merged to form a single voxel resulting in a coarser point cloud with fewer points as shown in Fig. 13a. Leveraging this fact, we use octree voxel subdivisions to predict the occupancy of the empty neighboring voxels with a deep learning model as shown in Fig. 13b. This makes our architecture a point cloud geometry prediction scheme to upsample a lower Level-of-Detail (LoD) point cloud into a higher LoD point cloud without any overhead to the compression-transmission pipeline. We use sparse convolution by employing Minkowski Engine with a UNet like structure employing inception-residual network blocks. To the best of our knowledge, this is the only work on point cloud upsampling that specifically targets the quantization loss during the compression-transmission pipeline.

Both the objective and subjective results show that we significantly improve the quality of the point cloud. Since our technique is a post-processing step, there is no transmission overhead or a bit rate cost to achieve this gain. Another use case for this technique is in display adaptation, when zooming in a point cloud this technique can help super resolve details for display adaptation.



(a) (b)

Figure 13: (a) Voxel merging due to quantization. (b) Upsample using voxel prediction.

3.2 System Model

3.2.1 Problem Formulation

Quantization is a necessary step in most compression-transmission pipelines. As a consequence of quantization, the neighboring voxelized points get merged into one voxel. Depending on the compression rate, the quantization step-size (*qs*) can determine the number of points lost and the LoD of the reconstructed point cloud. The quantization loss is modeled by:

$$\hat{X} = \left\lfloor \frac{X}{qs} \right\rfloor \times qs \tag{3.1}$$

where qs is the quantization step-size, X is the original point cloud and $\hat{X}$ is the quantized point cloud. This quantization results in duplicate points that are removed during the compression process. One example of $qs = 2$ is shown in Fig. 13a. Our goal is to reproduce these lost points by predicting the occupancy of the neighboring empty voxels given the coarser low LoD point cloud. An example of how each voxel would be upsampled using voxel prediction is shown in Fig. 13b.

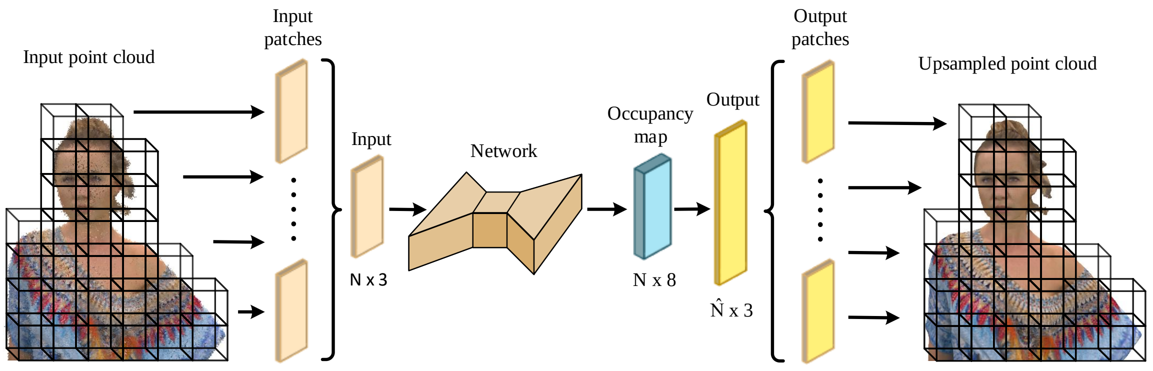


Figure 14: System Model.

3.2.2 Network Architecture

Our system model is shown in Fig. 14. Generally, a point cloud can be large with millions of points. To feed the point cloud to the network and make our system scalable, we subdivide the point cloud into smaller cube patches and feed each cube patch to the network. The input patch is a voxelized geometry and is of dimension $N \times 3$, where N is the number of voxels in the input cube patch and 3 are the x,y,z coordinates. The output of our network is a $N \times 8$ occupancy map for the $2 \times 2 \times 2 = 8$ voxels encompassing the input voxel. We use this occupancy map to generate a denser point cloud $\hat{N} \times 3$, where $N \leqslant \hat{N} \leqslant N \times 8$. It should be noted that our output predicted occupancy map can be more than 8 voxels depending on the qs and amount of upsampling needed. e.g. For $qs = 4$ we can employ an occupancy map of $4 \times 4 \times 4 = 64$. We aggregate the output patches back together to form the upsampled point cloud.

We use sparse tensors and sparse convolution using Minkowski Engine [20]. We employ UNet type architecture [93] with three Inception-residual network blocks (IRB)

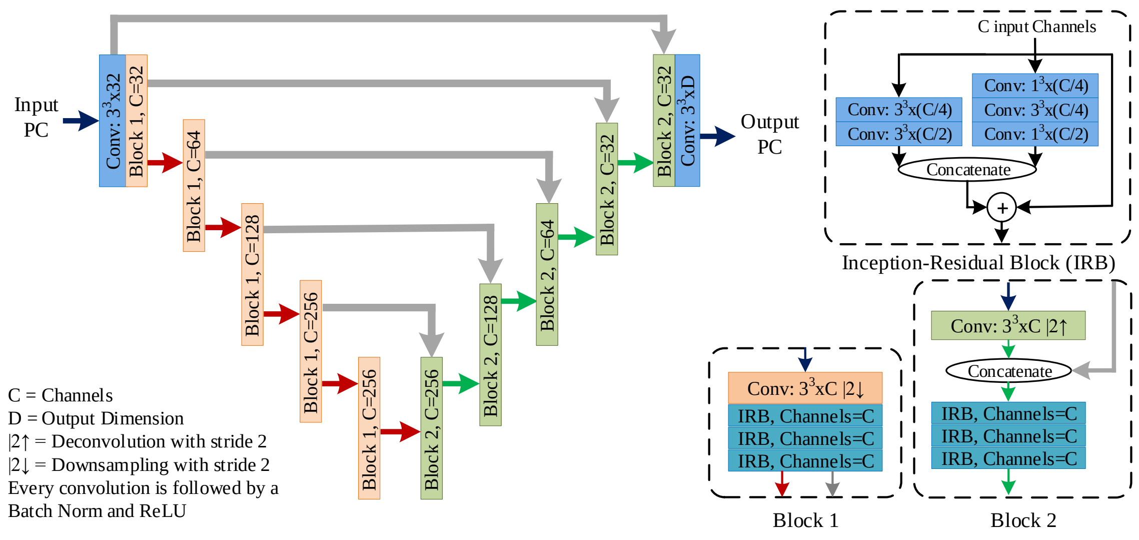


Figure 15: Network Architecture.

[100] per layer as shown in Fig. 15. We use a binary cross-entropy classification loss to compare the occupancy map prediction from the network and the ground truth (original) point cloud.

3.3 Simulation Results

An input cube patch size of $128 \times 128 \times 128$ voxels is used. **Dataset:** the system model is simulated on 8i voxelized full bodies dataset [59] that is being used in the MPEG standardization. The training is performed on two sequences (longdress, loot) and testing on the 3 sequences (redandblack, soldier, queen). Each sequence has hundreds of point clouds with each point cloud having up to a million points each.

We perform both objective and subjective evaluations. We ran experiments for

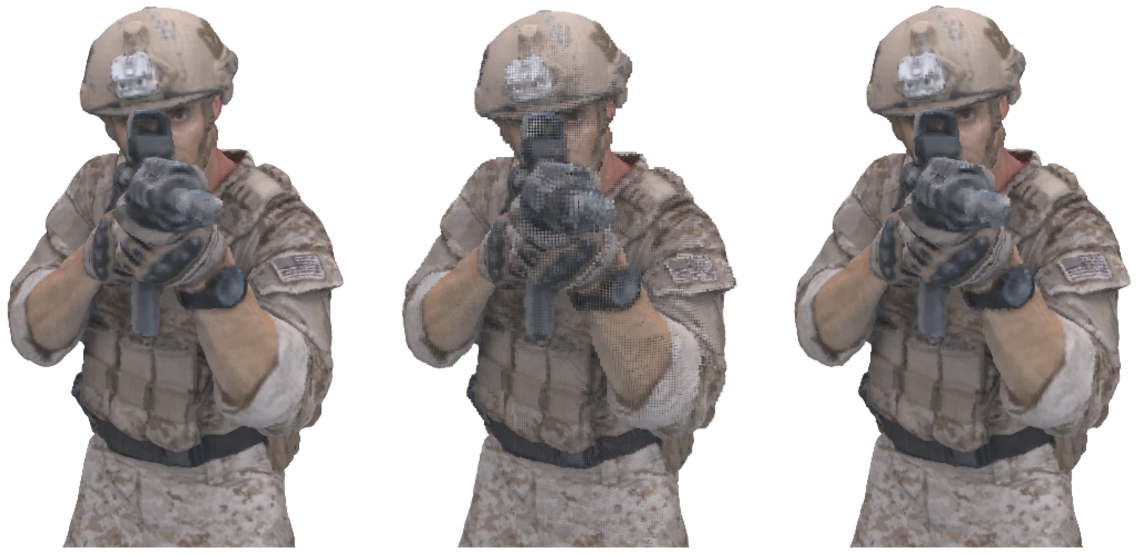


Figure 16: (a) Original point cloud, (b) Quantized point cloud with *qs* = 2, (c) Upsampled point cloud.

three $qs = 4/3, 2, 4$. These qs are being used in both MPEG PCC. For $qs = 4/3$ and $qs = 2$, we predict 8 neighboring voxels. However, for $qs = 4$, we increased our receptive field to include $4 \times 4 \times 4 = 64$ neighboring voxels. Which means the output of the network for $qs = 4$ is $N \times 64$. We use D1 geometry PSNR quality metric that is adopted by MPEG [106].

The results of the simulations are shown in Table 6. *Input PC* is the reconstructed point cloud after compression pipeline with a specific quantization step and the *Output PC* is the output of our network. As can be seen from the table, we see a significant improvement in the PSNR of the point cloud. Since our method is a post-processing and adds no overhead on the compression-transmission pipeline. We get on average a 8.8678 dB improvement in the quality of a reconstructed point cloud of $qs = 2$ without any

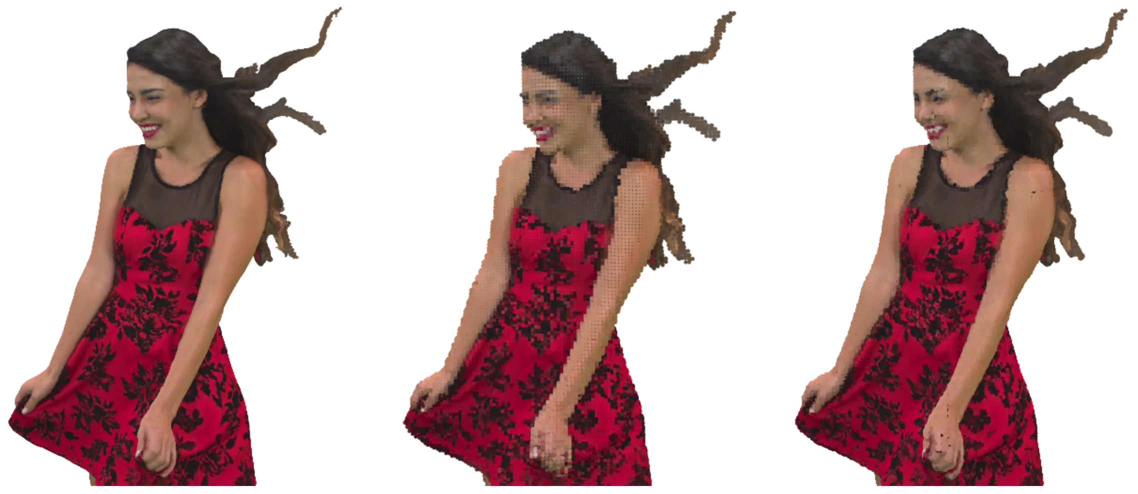


Figure 17: (a) Original point cloud, (b) Quantized point cloud with $qs = 4$, (c) Upsampled point cloud.

bit-rate cost.

Table 6: Average PSNR (dB) results.

qs	Input PC	Output PC	Difference
4/3	64.6646	73.8630	9.1984
2	63.2080	72.0758	8.8678
4	58.0077	65.1890	7.1813

The visual results are shown in Fig. 16 and Fig. 17 for $qs = 2$ and $qs = 4$ respectively. As can be seen our approach significantly improves the quality of the point cloud both objective as well as in subjective evaluations.

3.4 Conclusion

Point cloud compression is a necessary step for point cloud transmission, storage, and processing. However, during compression and transmission, point cloud suffers from quantization noise which results in lower Level-of-Detail (LoD) point clouds. In this work, we propose a deep learning-based point cloud geometry prediction scheme that takes a lower LoD point cloud and upsamples it into a higher LoD point cloud. We use octree to encompass each voxel and its neighboring voxels from the lower LoD point cloud into 8 voxels (or more). Then we learn an occupancy map for each of these voxels using a deep learning architecture. Based on the occupancy map, we generate a higher LoD point cloud by populating the empty voxels. The simulation results show that our method significantly improves the PSNR of the reconstructed point cloud geometry without adding any transmission overhead to the compression-transmission pipeline. This makes our method highly efficient and ideal post-processing step in decoding, as well as super-resolving point cloud for display adaptation.

CHAPTER 4

PU-DENSE: SPARSE TENSOR-BASED POINT CLOUD GEOMETRY UPSAMPLING

4.1 Introduction

There has been a surge in the usage of 3D point clouds in augmented/virtual reality (AR/VR), telepresence [29, 80], surveillance and autonomous driving [8, 9] which has been accompanied by significant advances in 3D sensors and capturing techniques [63]. With the rapid advancement of 3D point cloud acquisition technologies, such as LiDAR (Light Detection And Ranging) sensors, high precision point cloud representations have become affordable. Moreover, the recent advances in GPU power capabilities have enabled real-time rendering and visualization of dense 3D point clouds. Recent advances in point cloud compression by groups like MPEG [97] and JPEG Pleno [82] have enabled the efficient transfer of larger point clouds. These developments have now allowed the capture and utilization of very high definition real-world point clouds with millions of points per frame.

Based on their usage, point clouds can be categorized into point cloud scenes and point cloud objects. Point cloud scenes are dynamically acquired and are typically captured by LiDAR sensors. LiDAR sensors mounted on top of a vehicle for mobile mapping and autonomous navigation [45] are examples of a dynamically acquired point cloud scene. Point cloud objects can be further subdivided into static objects and dynamic objects. A static point cloud is a single object, whereas a dynamic point cloud is time-varying, where each instance of a dynamic point cloud is a static point cloud. Dynamic time-varying point clouds are used in AR/VR, volumetric video, and telepresence and can be generated using 3D models, i.e., CGI, or captured from real-world scenes using various methods such as multiple cameras with depth sensors surrounding the object and capturing movement over time. Advancement in sensor technologies has allowed the capture of photo-realistic point clouds with millions of points per object. These real-world high-resolution point clouds tend to be dense and pose complicated challenges in point cloud processing due to their size. In this paper, we introduce a novel point cloud geometry upsampling technique that does not just work on synthetic point clouds but can also process a diverse set of point clouds including dense high-resolution photo-realistic point clouds, real-world LiDAR scanned objects, as well as dynamically acquired outdoor LiDAR point clouds.

Recently, there have been considerable advances in point cloud upsampling along with advances in problems closely resembling point cloud upsampling like point cloud completion [54, 103, 117, 120] and point cloud denoising [25, 47, 132, 137]. Point cloud upsampling methods can be broadly categorized as optimization-based methods [10,49,50,68,83] and deep learning-based methods [7,65,88,125,128,130]. Optimization-based methods usually fit local geometry by utilizing geometry priors that only work well for low precision smooth surfaces. Optimization-based methods are computationally intensive and are difficult to scale to larger point clouds. Deep learning-based methods like PU-Net [130], 3PU [125], and PU-GAN [65] can effectively learn point cloud structures from data. However, these methods use kNN search-based patch selection for neighborhood feature aggregation. Raw points within the patch can be in any order and are processed directly by fully connected layers without considering the relative point location in the overall 3D representation or the point's distance from its neighbors. Furthermore, these methods are also memory hungry and computationally intensive, which is why they are limited to fixed small input sizes. Therefore, the current state-of-the-art in point cloud upsampling fails to build deeper architectures with large receptive fields that can effectively learn discriminative features, and be able to efficiently work on denser point clouds that have a large number of points.

To resolve these issues, we propose a novel PU-Dense architecture that can upsample synthetic point clouds, real-world scanned sparse point clouds as well as dense photo-realistic point clouds. The proposed framework offers the following contributions:

- PU-Dense employs a UNet [93] type encoder-decoder architecture that hierarchically reconstructs an upsampled point cloud via progressive rescaling and multiscale feature extraction. PU-Dense introduces a novel feature extraction (FE) unit with Inception-Residual Block (IRB) and a 3D Dilated Pyramid Block (3D-DPB) to extract 3D multiscale features with different field-of-views in a computationally efficient manner.

- PU-Dense is a fully convolutional geometry upsampling network that is translation invariant and has a variable input size that takes advantage of the sparse nature of point clouds and employs sparse convolutions [20] that tend to be memory efficient. Rather than a distance-based loss function, PU-Dense employs a memory-efficient binary voxel classification loss and utilizes 3D data representation that enables efficient learning of 3D point features. While the previous state-of-the-arts limit their input to a fixed number of points between a few hundred and a few thousand points, our method can process millions of points per iteration with variable input size.

- Rather than creating our own synthetic dataset like the previous works, we intend to use a standardized dataset. We train our network on ShapeNet while testing on ShapeNet as well as real-world photo-realistic point cloud datasets like MPEG's 8i Voxelized Surface Light Field (8iVSLF), JPEG Pleno's 8i Voxelized Full Bodies (8iVFB), Queen by technicolor, real-world LiDAR scanned objects from ScanObjectNN dataset, and dynamically acquired outdoor LiDAR dataset from KITTI. Experimental results show that our method considerably outperforms the previous works in not just synthetic point cloud upsampling but also dense photo-realistic point clouds as well as sparse LiDAR-based datasets. We show that the proposed method is robust against noise. Moreover, the results show that PU-Dense is faster as well as more memory efficient when compared with the other state-of-the-art point cloud upsampling.

4.2 Related Work

4.2.1 Optimization-based Upsampling Methods

Earlier works formulated point cloud upsampling as an optimization problem. A pioneering solution proposed by Alexa *et al.* [10] computed a Voronoi diagram on the moving least squares (MLS) surface. Lipman *et al.* [68] developed a parametrization-free method using a locally optimal projection operator (LOP) to resample points and reconstruct surfaces based on an L1 norm. This work was subsequently improved to weighted LOP [49] and continuous LOP [83] methods to consolidate point sets with noise, outliers, and nonuniformities. However, LOP-based methods' performance suffers when upsampling sharp edges and corners because they assume points are sampled from smooth surfaces. Huang *et al.* [50] introduced an edge-aware (EAR) approach that is designed to preserve sharp features by first resampling away from edges and then progressively approaching edges and corners. These optimization-based methods are not data-driven, assume insufficient priors, and often require additional attributes. Furthermore, these methods are computationally intensive so they are not scalable to high-resolution point clouds with a large number of points.

4.2.2 Deep Learning in Point Clouds

Recent advances in deep learning have seen a lot of success in point cloud processing using deep learning models [43]. The raw format of point cloud lacks point order and has an irregular structure which brings new challenges in employing deep learning solutions for point cloud processing.

Grid-based architectures. Pioneer works [102, 122] have tried to extend 2D convolutional neural networks to 3D space by voxelizing the point cloud into uniform voxels and applying 3D convolutions to them. However, due to the sparsity of point clouds, most of the computations are wasted on empty voxels. Projection methods [57, 61, 98, 99] project 3D data onto 2D planes and then process these 2D planes. However, these projection-based methods are not as effective as processing the 3D point cloud.

Point-based methods. Raw point cloud data acquired from 3D sensors are in the form of unordered points. Point-based methods process raw point cloud data using permutation invariant feature extraction networks. PointNet [84] was among the first deep learning solutions that can directly process raw 3D point cloud data. They employed pointwise fully connected layers and symmetric max-pooling to make the process permutation invariant. This work was subsequently improved to PointNet++ [85] that introduced hierarchical learning to learn more meaningful and discriminative features. PointCNN [67] was among the first works to use the convolutional layers on raw point cloud data. PointCNN proposed a permutation invariant $\mathcal{X}$-Conv that employs MLP layers to learn the permutation followed by a convolutional layer. The VoxelNet [138] divides point clouds using voxel partition and then uses PointNets to learn point-wise features. However, point-based methods are computationally expensive, which imposes severe constraints on building larger networks or their applicability on point clouds with a large number of points.

New point cloud data representation. There has been significant work on graph-based models which can operate on unordered 3D data. Graphs can be constructed from 3D point clouds in a variety of ways [58, 60, 118] which have shown promising results in point cloud processing. Octree-based convolutional neural networks were introduced in [116] that converted the data into an octree data representation and employed convolutions on octants of the octree data structure. SparseConvNet [36] by Facebook was among the first sparse convolutional neural networks that achieved state-of-the-art results on point clouds. SparseConvNet employed submanifold sparse convolutions [37] that exploited the sparse nature of point clouds and ensured that the convolutions would not “dilate” the data. MinkowskiNet [20] is another such implementation that employs sparse convolutions for 3D point cloud learning. Sparse convolutions exploit the inherent sparsity of point cloud data and store point cloud data in sparse tensors where convolutions are only performed on the voxels that are occupied. This makes the sparse convolutions much more computationally efficient allowing for a much deeper architecture to be built and the ability to process hundreds of thousands of points in a single inference time. We employ sparse convolutions using Minkowski Engine in our work.

4.2.3 Deep Learning-based Upsampling Methods

PU-Net [130] was the pioneer deep learning upsampling work on point cloud that uses PointNet++ for feature extraction. PU-Net uses multi-branch MLPs to expand features with a joint reconstruction and repulsion loss to generate uniform point clouds. PU-Net operates on small patch level and does not consider the spatial relations among the points which results in the output lacking fine-grained high-resolution geometry structures. EC-Net [128] intended to improve PU-Net work and introduced a point-to-edge distance loss, which can help preserve the edges. However, EC-Net requires the tedious work of labeling the point cloud data with annotated edge and surface information. Wang *et al.* [125] proposed 3PU-Net which introduced a patch-based progressive upsampling inspired by image super-resolution methods. 3PU learns point-wise features similar to the methods before and employs cascaded 2x upsampling networks. Each subnet in 3PU appends a 1D code to the features which limits each subnet upsampling to a factor of 2. However, multiple subnets can be progressively employed to get a large upsampling factor, say 16 times. 3PU only supports an upsampling factor in powers of 2 and also requires careful step-by-step training. PU-GAN [65] introduced a generative adversarial network for point cloud upsampling. Their performance improvement mainly comes from the introduction of a discriminator. PUGeo-Net [88] introduced a geometric-centric approach where they upsample point clouds by learning the first and second fundamental forms of the local geometry. However, their method needs additional supervision in the form of normals, which many point clouds like those generated by LiDAR sensors do not come with.

However, these deep learning-based upsampling methods use PointNet++ type architecture where the raw unordered points are stored in 2D arrays and kNN search is employed for neighborhood feature aggregation without considering the point location in the overall 3D representation. Point-based approaches perform upsampling by splitting scenes into smaller chunks, effectively restricting the model's ability to learn from global context. These methods employ deep learning operations that are usually used for 2D images to learn 3D features, which tends to be an inefficient approach. Furthermore, all these methods are limited to a small fixed input size and employ patch-based scaling to be able to process larger point clouds. Additionally, even after using smaller patches, these methods suffer from memory issues and are computationally intensive, especially their loss functions. This limits these architectures to a relatively shallower network which decreases their ability to learn discriminative features. Due to these reasons, the previous upsampling methods are often limited to and tested on small mesh-based point clouds and give poor results when applied on dense high-resolution real-world scanned point clouds such as the ones used in AR/VR or the MPEG standards.

Note that recently there has been some point cloud upsampling work that has emerged in parallel to our work. Recently, Qian *et al.* [86] proposed PU-GCN that uses a multi-level feature extraction using an inception-based graph convolutional network. They employ shuffling rather than duplicating features to expand the feature space for upsampling. Dis-PU [66] proposes to disentangle the upsampling task using two cascaded sub-networks, a dense generator, and a spatial refiner, to obtain both distribution uniformity and proximity-to-surface. Inspired by Meta-SR from image super-resolution, Meta-PU [124] was proposed to support point cloud upsampling of arbitrary scale factors with a single model. Meta-PU proposes a meta-subnetwork to dynamically adjust the weights of their residual graph convolution (RGC) upsampling network for different scaling factors. The upsampling network outputs a dense maximum points point cloud which is downsampled using farthest point sampling (FPS) to the desired ratio. However, this whole process is computationally intensive and inefficient when employed to point clouds with a large number of points. More recently, Flexible-PU [89] has been proposed that can also work for an arbitrary upsampling ratio using a lightweight neural network. Flexible-PU explicitly involves the local neighborhood information in the learning process. They generate new samples by an affine combination of neighboring points projected onto the tangent plane which are further refined by a self-attention-based refinement module.

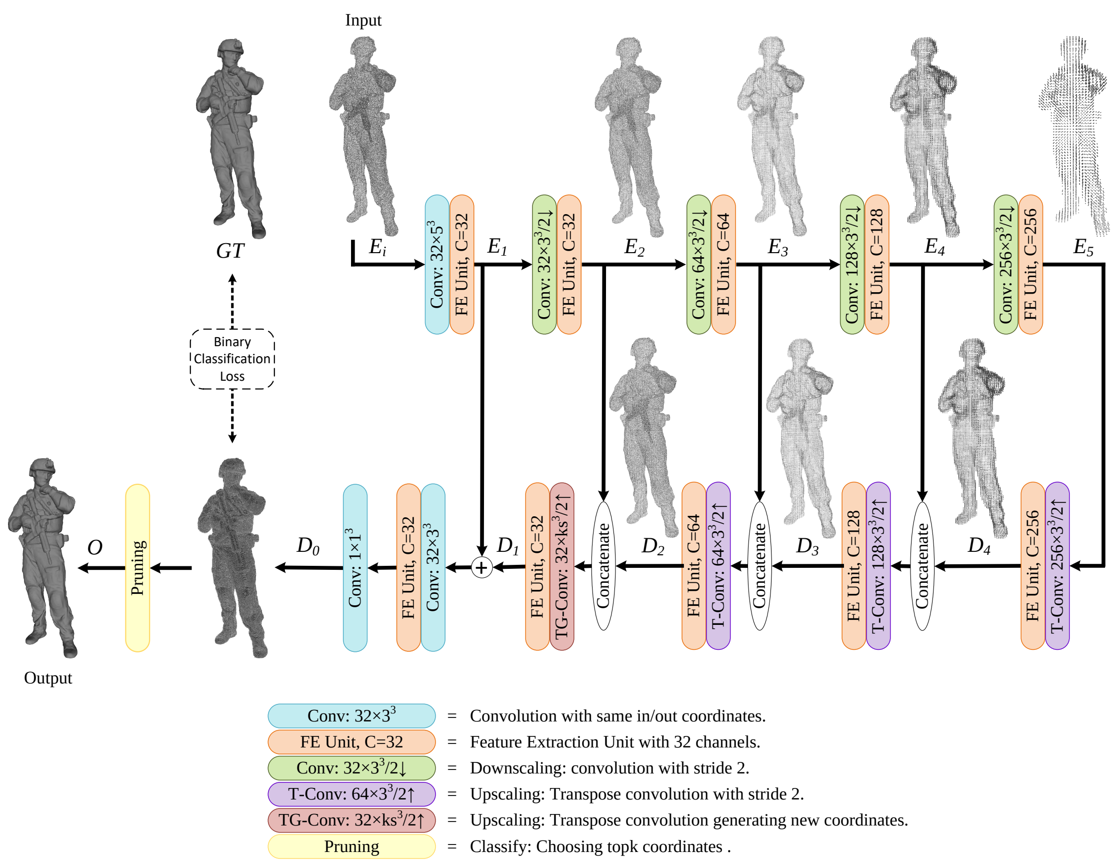


Figure 18: PU-Dense network architecture consisting of encoder and decoder branches with skip connections in the middle. E_i is the input to the network that gets downscaled in the encoder branch by using convolutions of stride 2. The decoder side upscales the tensor by using transpose convolutions. *TG-Conv*: Transpose convolution generating new coordinates is employed in the last upscaling layer of the decoder side to populate additional coordinates around the existing coordinates. PU-Dense employs a voxel-based binary cross-entropy loss to compare decoder output D_0 with ground truth GT. Finally, pruning is applied to classify D_0 by choosing top k coordinates with the largest features. For convolutions, the terminology 32×3^3 denotes a kernel size of 3^3 with a channel size of 32. An example of tensor sizes throughout the network is shown in Table 7.

4.3 PU-Dense

Given a lower level-of-detail (LoD) point cloud $\mathcal{X} = \{x_i \in \mathbb{R}^3\}_{i=1}^{M}$ with M points and an upsampling ratio R, we seek to generate a denser higher LoD point cloud $\mathcal{X}_R = \{x_i^r \in \mathbb{R}^3\}_{i,r=1}^{M,R}$ that is distributed uniformly over the underlying surface.

4.3.1 Preprocessing

Converting a point cloud from its raw format to a 3D volumetric representation is called *voxelization*. PU-Dense employs voxelization that allows it to use 3D convolutions to learn 3D features, which are more consistent and accurate in feature representation compared to the previous works where kNN is employed on unordered points to aggregate neighborhood features. Since we are only working with geometry and not other attributes, we assign feature f to each coordinate where $f(x, y, z) = 1$, if the voxel is occupied, and $f(x, y, z) = 0$ otherwise. We represent each input point cloud using a data tensor with a set of coordinates $C = \{(x_i, y_i, z_i)\}_i$ and their associated features $F = \{f(x_i, y_i, z_i)\}_i$.

4.3.2 The Network

PU-Dense is the first fully convolutional geometry upsampling network which makes it translation-invariant with variable input size. PU-Dense employs a multiscale U-Net [93] encoder-decoder type architecture built on Minkowski engine [20] utilizing sparse convolutions. Sparse convolutions exploit the inherent sparsity of point cloud data and are much more memory efficient. Sparse convolution is defined in [20] as:

$$f_u^{out} = \sum_{i \in N_3(u, C^{in})} W_i f_{u+i}^{in} \quad \text{for} \quad u \in C^{out}, \tag{4.1}$$

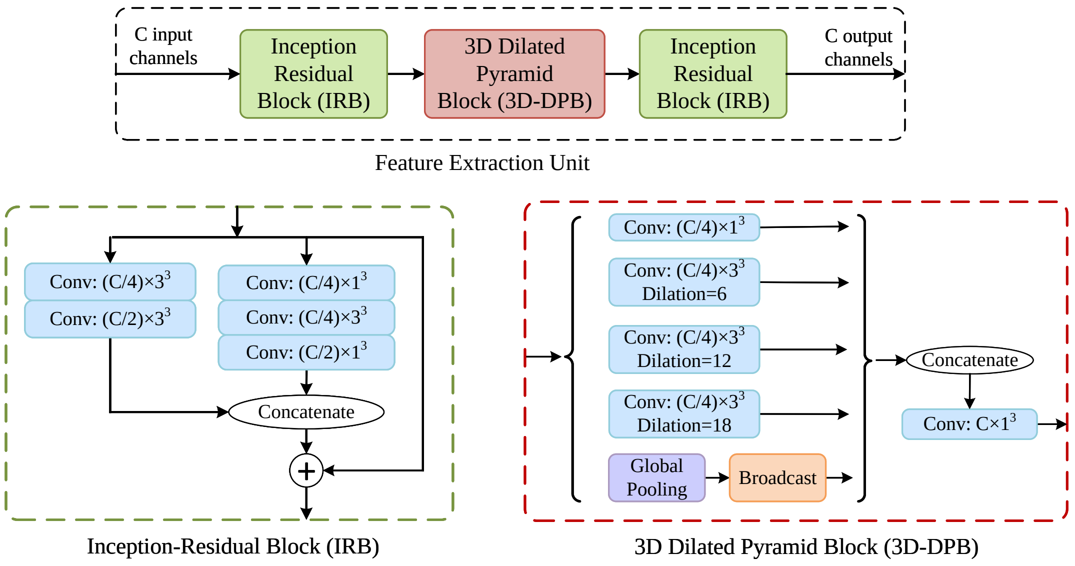

C input channels
Inception Residual Block (IRB)
3D Dilated Pyramid Block (3D-DPB)
Inception Residual Block (IRB)
C output channels
Feature Extraction Unit
Conv: (C/4)×3³
Conv: (C/2)×3³
Conv: (C/4)×1³
Conv: (C/4)×3³
Conv: (C/2)×1³
Concatenate
Inception-Residual Block (IRB)
Conv: (C/4)×1³
Conv: (C/4)×3³ Dilation=6
Conv: (C/4)×3³ Dilation=12
Conv: (C/4)×3³ Dilation=18
Global Pooling
Broadcast
Concatenate
Conv: C×1³
3D Dilated Pyramid Block (3D-DPB)

Figure 19: *FE Unit*: Feature Extraction Unit. Each FE Unit consists of two Inception-Residual Blocks (IRB) and one 3D Dilated Pyramid Block (3D-DPB). FE Unit efficiently learns 3D multiscale spatial features by employing multiple sampling rates and receptive field.

where N_3 is a set of offsets that define the shape of a kernel and $N_3(u, C^{in}) = \{i|u + i \in C^{in}, i \in N_3\}$ is the set of offsets from the current center, u, that exists in C^{in}. C^{in} and C^{out} are the input and output coordinates. f_u^{in} and f_u^{out} are the input and output features at location u. W_i denotes the kernel value at offset i.

The proposed network architecture, shown in Fig. 18, has an encoder downscaling network and a decoder upscaling network, with residual connections in between. For convolutions, the terminology 32×3^3 denotes a kernel size of 3^3 with a channel size of 32. The PU-Dense architecture employs a total of four different types of sparse convolutions: (i) Convolution, (ii) Downscaling Convolution, (iii) Transpose Convolution, and (iv) Transpose Convolution generating new coordinates. Employing these four different kinds of sparse convolutions, PU-Dense builds a 3D multiscale spatial architecture employing hierarchical learning using an encoder-decoder architecture.

Convolution (Referred as *Conv*). PU-Dense employs sparse convolutions with the same in/out coordinates ($C^{in} = C^{out}$). The coordinates of the sparse tensor stay the same after passing through these kernels and only the feature changes.

Downscaling convolution (Referred as *Conv*$/2 \downarrow$): PU-Dense employs sparse convolution with stride of two to halve the scale of each geometric dimension.

Transpose convolution (Referred as *T-Conv*$/2 \uparrow$): Sparse transpose convolution with a stride of two is used to map the tensor coordinates back to their previous upscaled coordinates.

Transpose convolution generating new coordinates (Referred as *TG-Conv*$/2 \uparrow$): This is a special case of sparse transpose convolution where the convolution generates new coordinates before aggregating features [44]. The new coordinates are generated on the voxels the kernel covers during convolution around the input coordinates. PU-Dense employs *TG-Conv* in the last upscaling layer of the decoder with a variable kernel size of ks. Variable kernel size is used here to optimize the architecture according to the resolution of the point cloud and the upsampling ratio. A larger ks is suitable for a sparser point cloud and would result in a larger number of output coordinates (C^{out}) generated. In this work, we employ a kernel size of $ks = 5$ for upsampling ratios of 4x and 8x.

Table 7 shows an example of sparse tensor sizes in the PU-Dense network during 4x upsampling. Since a $ks = 5$ is used in *TG-Conv*, the size of the new generated

Table 7: An example of tensor sizes in PU-Dense architecture (Fig. 18) for 4x upsampling using $ks = 5$ for *longdress* PC from 8i.

Tensor	Size of coordinates (C)	Size of features (F)
GT	830,397	1
E_i	207,599	1
E_1	207,599	32
E_2	139,244	32
E_3	52,612	64
E_4	14,440	128
E_5	3,623	256
D_4	14,440	256
D_3	52,612	128
D_2	139,244	64
D_1	4,379,676	32
D_0	4,379,676	1
O	830,397	1

coordinates in D_1 is $4{,}379{,}676$. PU-Dense employs **Pruning** to choose the top$k$ features and their corresponding coordinates from D_0 to form the final output tensor O, where k is $830{,}397$ in this particular example. *Pruning* is implemented on D_0 which has a feature length of 1 making it easier to choose the top$k$ features and their corresponding coordinates to create the output tensor O.

4.3.3 Feature Extraction Unit

We introduce a novel feature extraction unit containing two *Inception-Residual Blocks* and a single *3D Dilated Pyramid Block* as shown in Fig. 19.

Inception-Residual Block (IRB) is inspired by Inception-ResNet architecture

[100] which was originally proposed for 2D images. IRB has been implemented with great success using 3D sparse convolutions in earlier works [113]. Similar to 2D Inception-ResNet architecture, IRB employs filters of different sizes on the same feature scale while squeezing the feature domain and adding a residual link to the inception block. The IRB block employs 1x1x1 convolutions which do not learn any spatial patterns but do learn patterns across the depth (cross channel) of each occupied voxel in the sparse tensor. Combining kernels of different sizes (1x1x1 and 3x3x3) allows the block to learn patterns across the depth (channel) as well across spatial patterns. This combined with a residual link leads to a computationally efficient layer which much like its 2D counterpart, Inception-ResNet architecture, is more discriminative and converges faster [100].

3D Dilated Pyramid Block (3D-DPB) is inspired by the success of spatial pyramid pooling [46] as well as pyramid blocks in image segmentation tasks [18] which showed that it is effective to resample features at different scales for accurately and efficiently classifying regions of an arbitrary scale. In 3D-DPB, we employ multiple parallel 3D sparse convolutions with different dilation rates to implement a 3D pyramid architecture. Dilation convolution, also called atrous convolution [19], allows us to arbitrarily enlarge the field-of-view of filters at any convolutional layer. A 3D global pooling is also implemented, which is followed by broadcasting the pooled feature to all the occupied coordinates. The features from all 3D-DPB layers are then concatenated followed by a 1x1x1 convolution to refine the features. 3D-DPB probes the incoming sparse tensors with filters at multiple sampling rates and effective fields-of-views, thus capturing objects as well as geometric context at multiple scales. 3D-DPB with different dilation rates effectively captures multi-scale information, allowing us to extract denser feature maps by arbitrarily enlarging the receptive field.

4.3.4 Loss Function

The loss functions used in previous works usually employ distance-based metrics like chamfer distance, reconstruction loss, or repulsion loss, which tend to be memory inefficient and fail to learn accurate reconstruction. This is one of the reasons why the previous works are limited to processing small point clouds or small patches (usually less than 1024 points). PU-Dense implements an efficient voxel-based binary occupancy classification loss [56, 113] that allows us to process millions of points at a time. During training, PU-Dense applies Binary Cross Entropy (BCE) loss on the output of the decoder (D_0) and compares the occupied voxels to the ground truth point cloud (GT) as shown in Fig. 18. BCE loss during training is calculated using the following formula:

$$L_{BCE} = -\frac{1}{N}\sum_{i}(x_i log(p_i) + (1 - x_i) log(1 - p_i)) \tag{4.2}$$

where x_i is the voxel label that is either occupied (1) or empty (0) in the GT point cloud. p_i is the probability of the voxel being occupied and is calculated using a *sigmoid* function applied to the decoder output D_0.

4.3.5 Scalability

To the best of our knowledge, all current deep learning-based point cloud upsampling models are constrained to a fixed number of input points and suffer from memory issues. These models can typically process a fixed number of points at one moment, e.g., PU-Net: 1024 points, EC-Net: 1024 points, PU-GAN: 256 points, PUGeo-Net: 256 points. All these methods employ patch-based point cloud processing methods. However, PU-Net is limited to smaller point clouds. EC-Net, PU-GAN, and 3PU have extended their patch-based approaches to partition larger point clouds into smaller overlapping patches to operate on the individual patch separately. These patches are upsampled independently, then merged together. The merged pointset is then resampled using farthest point sampling to obtain uniform point distribution despite overlapping regions. Because of all the pre-processing and post-processing involved, these methods work well on smaller point clouds but are very inefficient when applied to point clouds with a large number of points.

Our method is the first fully convolutional upsampling method and, therefore, has variable input point cloud size. Moreover, since we employ efficient sparse convolutions and also have a memory-efficient loss function, we can process a much larger number of points. To make the network further scalable to even bigger point clouds, we employ a simple kd-tree partition that divides the point cloud into smaller non-overlapping point clouds that can be processed separately as well as in parallel.

4.4 Experimental Results

We performed extensive experiments, quantitatively and qualitatively, compared our methodology with state-of-the-art point cloud upsampling methods, and evaluated various aspects of our model. We perform comprehensive experiments to test PU-Dense on different scenarios: (1) Test the proposed method on real-world scanned objects, (2) Show visual results for mesh surface reconstruction of the upsampled point clouds, (3) Test the robustness of our method against Gaussian noise, (4) Show comparative results for inference time and trainable parameters for different methods, (5) Perform ablation study to show the effectiveness of different components of PU-Dense. Further experimental details, extended results, and additional visual results can be found in the supplemental material.

4.4.1 Dataset

We train our network on *ShapeNet* dataset [17] while test it on *ShapeNet*, *8iVFB v2* [23], *8iVSLF* [59], *Technicolor*, *ScanObjectNN* [109], and KITTI [34] datasets. We evaluate the performance of our approach on a diverse set of point clouds in terms of spatial density and content type. *ShapeNet* is a mesh-based dataset, *ScanObjectNN* dataset is a real-world scanned object dataset containing sparse point cloud objects, *KITTI* is a dynamically acquired outdoor LiDAR dataset, whereas *8iVFB v2*, *8iVSLF*, and *Technicolor* are real-world captured dense photo-realistic dynamic point cloud datasets for immersive communication used in MPEG [96] and JPEG [82] point cloud compression standardization.

- **ShapeNet**. We randomly select ≈ 24000 3D mesh models from the core dataset of ShapeNet. We sampled the mesh model into point clouds by randomly generating points on the surfaces of the mesh, then randomly rotated and quantized the point cloud to 7-bit precision. The size of the point cloud in this dataset is between $5\,000$

to $50\,000$ points per point cloud.

- **8iVFB v2**. JPEG Pleno's 8i Voxelized Full Bodies dataset [23]. This is a dynamic voxelized point cloud dataset where each sequence represents a 10 second long video captured at 30 fps with a total of 300 frames. Each frame has a resolution of 1024 with 10-bit precision. We use four sequences from this dataset: *Longdress*, *Loot*, *Red and Black*, and *Soldier* containing about $770\,000$, $790\,000$, $730\,000$, and $1\,060\,000$ points per frame respectively.

- **8iVSLF**. MPEG's 8i Voxelized Surface Light Field dataset [59] of dynamic point clouds. It has a resolution of 4096 with 12-bit precision. We use two sequences from this dataset *Boxer* and *Thaidancer* containing about $3\,130\,000$ and $3\,490\,000$ points respectively.

- **Technicolor**. We employ the dynamic point cloud sequence *Queen* produced by Technicolor (https://www.technicolor.com/fr) with about 1 million points per frame.

- **ScanObjectNN**. [109] This is a real-world scanned object dataset which contains $\approx 15,000$ objects that are categorized into 15 categories with 2902 unique object instances. Compared with other datasets, this is a low-resolution sparse point cloud dataset with 2048 points per point cloud. We compare visual results for *ScanObjectNN* later in this section.

- **KITTI**. [34] This is a dynamically acquired dataset captured by LiDAR for autonomous driving. This is a sparse dataset that captures outdoor scenes. We perform visual comparisons between different methods when evaluated on KITTI dataset and show the results later in this section.

4.4.2 Implementation Details

We use kernel size, $ks = 5$ in our experiments. Unlike previous work where small patches had to be extracted from the training dataset and fed into the network, we trained our network with a batch size of 8 by feeding eight *ShapeNet* point clouds into our network each iteration. We implemented the proposed framework in PyTorch with Minkowski Engine [20].

4.4.3 Evaluation Metrics

Previous works have employed different quality assessment metrics; some works have even introduced their novel evaluation metrics. We consider commonly-used evaluation metrics that compare the reconstructed point cloud to the ground truth point cloud to quantitatively evaluate the performance of different methods. These methods are Chamfer distance (CD), point-to-point (D1) based mean squared error peak signal-to-noise ratio (D1 PSNR), point-to-plane (D2) based mean squared error peak signal-to-noise ratio (D2 PSNR), and point-to-point (D1) based Hausdorff PSNR. *MSE D1 PSNR*, *MSE D2 PSNR*, and *Hausdorff PSNR* has been adopted by both MPEG and AVS standards as an evaluation metric for point cloud quality [106]. We obtain the geometry PSNRs using the *pc_error* MPEG tool [107].

Table 8: Extended comparative results (CD (10^{-2}) and MSE PSNR).

Dataset	Upsampling Method	4x		8x	
		CD (10^{-2}) ↓	PSNR (dB) ↑	CD (10^{-2}) ↓	PSNR (dB) ↑
	Downsampled PC	108.18	64.63	199.94	61.96
ShapeNet	3PU	76.36	68.65	149.20	65.37
	PU-GAN	49.41	70.64	174.58	64.88
	PU-GCN	48.15	70.90	65.81	69.59
	Dis-PU	36.23	72.19	55.62	70.23
	PU-Dense (Ours)	**18.82**	**75.24**	**30.52**	**73.11**
	Downsampled PC	114.63	64.38	222.91	61.49
8iVFB	3PU	67.04	69.41	105.43	66.83
	PU-GAN	45.60	70.92	117.66	66.19
	PU-GCN	46.30	70.96	63.71	69.78
	Dis-PU	32.47	72.72	51.68	70.59
	PU-Dense (Ours)	**19.38**	**75.05**	**33.18**	**72.57**
	Downsampled PC	286.67	73.17	600.34	70.00
8iVSLF	3PU	135.41	78.98	202.82	76.78
	PU-GAN	121.52	79.58	231.39	76.34
	PU-GCN	103.84	80.28	138.81	79.17
	Dis-PU	91.99	81.16	129.79	79.52
	PU-Dense (Ours)	**58.38**	**83.93**	**102.82**	**81.79**
	Downsampled PC	106.69	64.69	196.46	62.04
Queen	3PU	57.13	70.19	90.90	67.55
	PU-GAN	41.67	71.43	110.42	66.36
	PU-GCN	42.67	71.24	59.47	70.01
	Dis-PU	30.29	73.08	47.32	70.90
	PU-Dense (Ours)	**15.76**	**75.93**	**25.45**	**73.76**

We define the Chamfer distance between PC A and PC B as:

$$d_{CD}(A,B) = \sum_{x \in A} \min_{y \in B} ||x-y||_2^2 + \sum_{y \in B} \min_{x \in A} ||x-y||_2^2 \tag{4.3}$$

Intuitively, the first term measures an approximate distance from each upsampled point to the target surface, and the second term rewards an even coverage of the upsampled surface and penalizes gaps.

The geometry quality metrics point-to-point MSE PSNR is obtained from a normalization factor and the mean squared error (MSE), as defined in (4.4) which is computed from PC A to PC B as well as in the opposite direction. The PSNRs of the two directions are then combined to obtain a single symmetric PSNR value with the maximum pooling function, as defined in (4.5).

$$PSNR_{A,B}^{MSE} = 10 \log_{10} \left(\frac{p_s^2}{d_{A,B}^{MSE}} \right) \tag{4.4}$$

$$PSNR^{MSE} = min(PSNR_{A,B}^{MSE}, PSNR_{B,A}^{MSE}) \tag{4.5}$$

In (4.4), p_s is signal peak and $d_{A,B}^{MSE}$ is the average squared error (i.e., MSE) between all points in PC A and their corresponding nearest neighbor point in PC B. The point-to-point MSE is computed from the point-to-point distance or error $\overrightarrow{e}(i,j)$ between each point in PC A and its nearest neighbor in PC B, $d_{A,B}^{Po2Po}$;

$$d_{A,B}^{MSE} = \frac{1}{N_A} \sum_{\forall a_i \in A} d_{A,B}^{Po2Po}(i) \quad \text{, with} \tag{4.6}$$

$$d_{A,B}^{Po2Po} = ||\overrightarrow{e}(i,j)||_2^2 \tag{4.7}$$

As far as the signal peak p_s in (4.4) is concerned, the largest diagonal (LD) distance of the PC bounding box is typically used for non-voxelized data. For *ShapeNet*, *8iVFB* and *Queen* datasets we use $p_s = 1024$. For *8iVSLF* dataset we use $p_s = 4096$. In the D2 PSNR, MSE is still based on the distance between each point to its nearest neighbor but this distance is now computed from the projection of the point-to-point error vector $\vec{e}(i,j)$ along the normal vector of the underlying surface at point j in PC B.

Hausdorff PSNR is derived in a similar symmetrical manner but instead of using point-to-point MSE distance, the Hausdorff distance is employed. The Hausdorff distance is defined as the largest distance between all the points in point cloud A and their nearest neighbor in reference point cloud B, thus defining the Hausdorff distance as:

$$d_{A,B}^{Haus} = \max_{\forall i \in A} d^{A,B}(i) \tag{4.8}$$

4.4.4 Comparison with State-of-the-arts

We were unable to run optimization-based methods like **EAR** on large point clouds like the 8iVFB dataset. **PU-Net** [130] is an earlier work, the code of which is not scalable to larger point clouds, so we did not include it in our comparisons. We chose four state-of-the-art upsampling methods that were scalable to larger point clouds and have so far shown the best results in point cloud upsampling: **PU-GAN** [65], **3PU** [125], **PU-GCN** [86] and **Dis-PU** [66]. Their models are trained with the author-released code, and all settings are the same as stated in their papers. These networks were originally trained on relatively sparser point clouds generated from small synthetic mesh-based objects. To make the comparison fairer, we retrained these networks on the same data as our model, i.e., ShapeNet, and then tested them on our test datasets. We generated about 24000 patches from ShapeNet dataset for training purposes. For 3PU a patch size of 1024

Table 9: Quantitative comparison with state-of-the-art approaches using D2 PSNR for 4x and 8x upsampling.

Dataset	Upsampling Method	D2 PSNR (dB)	
		4x	8x
8iVFB	3PU	72.71	69.02
	PU-GAN	74.21	67.96
	PU-GCN	75.66	74.05
	Dis-PU	76.70	74.79
	PU-Dense (Ours)	**79.05**	**76.45**

was used for 8x upsampling and 2048 for 4x upsampling, For PU-GAN, PU-GCN, and Dis-PU a patch size of 256 was used for 4x as well as 8x upsampling.

4.4.5 Objective Evaluation

For *Chamfer Distance (CD)*, lower values are better whereas for *D1 PSNR*, *D2 PSNR* and *Hausdorff PSNR* higher values are better. We compare our work with other state-of-the-art methods and display the results in Table 8, Table 9, and Table 10. In Table 8 we show the *Chamfer Distance (CD)* and *D1 PSNR* comparison results on four different datasets for both 4x as well as 8x upsampling for all five compared methods. The comparative results show that PU-Dense outperforms other state-of-the-arts by substantially decreasing the CD while significantly improving the D1 PSNR. Among the previous methods, PU-GCN and Dis-PU perform decently well on all the datasets. Dis-PU performs the best among the previous works. PU-GCN and Dis-PU perform similarly for 8x upsampling, however, Dis-PU performs much better than PU-GCN at 4x upsampling. Overall, PU-Dense outperforms all of these methods.

Table 10: Quantitative comparison with state-of-the-art approaches using Hausdorff PSNR for 4x and 8x upsampling.

Dataset	Upsampling Method	Hausdorff PSNR (dB)	
		4x	8x
8iVFB	Downsampled PC	51.95	49.10
	3PU	45.56	45.51
	PU-GAN	46.97	41.38
	PU-GCN	52.29	49.48
	Dis-PU	52.86	49.63
	PU-Dense (Ours)	**52.97**	**49.65**

We measure the point-to-plane D2 PSNR using the normals of the ground truth point clouds using 8iVFB dataset and show the results in Table 9. We can see that PU-Dense outperforms the state-of-the-art on D2 metric also. The Hausdorff PSNR is also calculated on 8iVFB dataset, the results of which are shown in Table 10. Hausdorff distance tends to be sensitive to outliers because the metric distance corresponds to the greatest of all the distances from a point in one point cloud to the closest point in the other point cloud (original to reconstructed, and vice-versa) [55]. The Hausdorff PSNR decreases for both PU-GAN and 3PU whereas PU-GCN, Dis-PU, and PU-Dense improve the Hausdorff PSNR. We see that PU-Dense outperforms the state-of-the-arts in this metric too. While the distance-based distortion loss function tends to add outliers in the upsampling process, PU-Dense generates points closest to the actual point cloud surface with limited outliers. This is because the new points generated by PU-Dense are generated using *TG-Conv*, which generates new points in the neighborhood of the already occupied voxels. This neighborhood is determined by the kernel size ks.

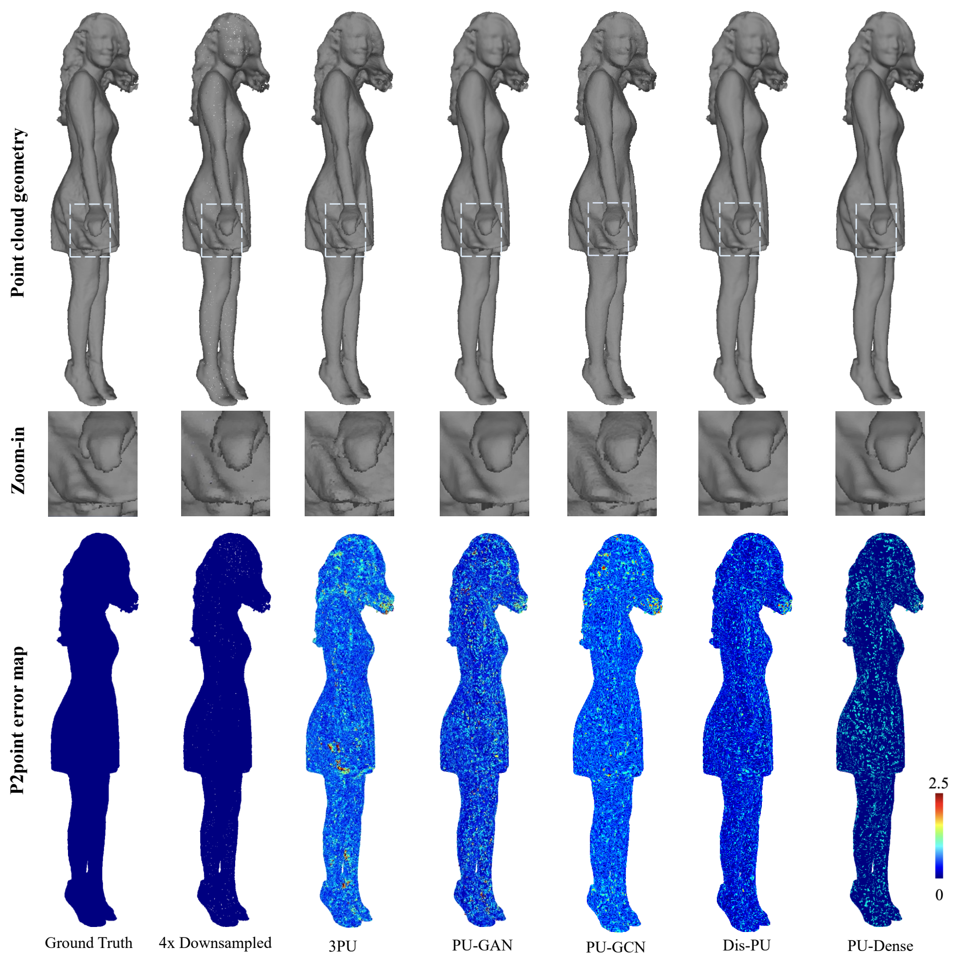


Figure 20: 4x upsampling visual results on sequence *Red and Black*.

4.4.6 Visual Comparisons

To visualize large photo-realistic point clouds like 8i sequences, it is not possible to show point-level visual upsampling results. Therefore, for these point clouds, we visualize their geometry by calculating their normals and with vertical shading. We also plot the error map based on the point-to-point (P2point) D1 distance between the point cloud and its ground truth to visualize the error distribution. We show the visual results of our experiments for 4x upsampling in Fig. 20 and for 8x upsampling in Fig. 21 using a point cloud frame from sequence *Red and Black* and *Longdress* respectively. We can see that our method generates limited outliers while populating points very close to the surface of the ground truth point cloud. This results in a high-resolution point cloud with considerably better quality than the other state-of-the-arts. Since PU-Dense uses *TG-Conv* to generate new points around the currently occupied voxels, there are limited outliers generated as is evident in Fig. 20 and Fig. 21. We also show the zoomed-in point cloud geometry views. The results of our proposed method are more precise, sharper, and cleaner, especially around key positions, such as corners and edges. We show further visual results on different datasets later in this section. We show further visual results on additional datasets in the next sections. Additional visual results and a description of how these figures are generated are also provided in the supplemental materials.

4.4.7 Evaluation on Real-World Scanned Objects

We also examined the performance of the proposed method on real-world scanned object dataset with 2048 points from *ScanObjectNN*. We show the comparative visual results of upsampling a point cloud from *ScanObjectNN* dataset in Fig. 22. *ScanObjectNN* dataset is captured through LiDAR and consists of sparse point clouds. We visualize the upsampled point cloud as well as the reconstructed mesh surface using the ball-pivoting algorithm for the input point cloud and each of the upsampled point clouds. As can be seen in Fig. 22, the proposed method (PU-Dense) works well on even sparse object point

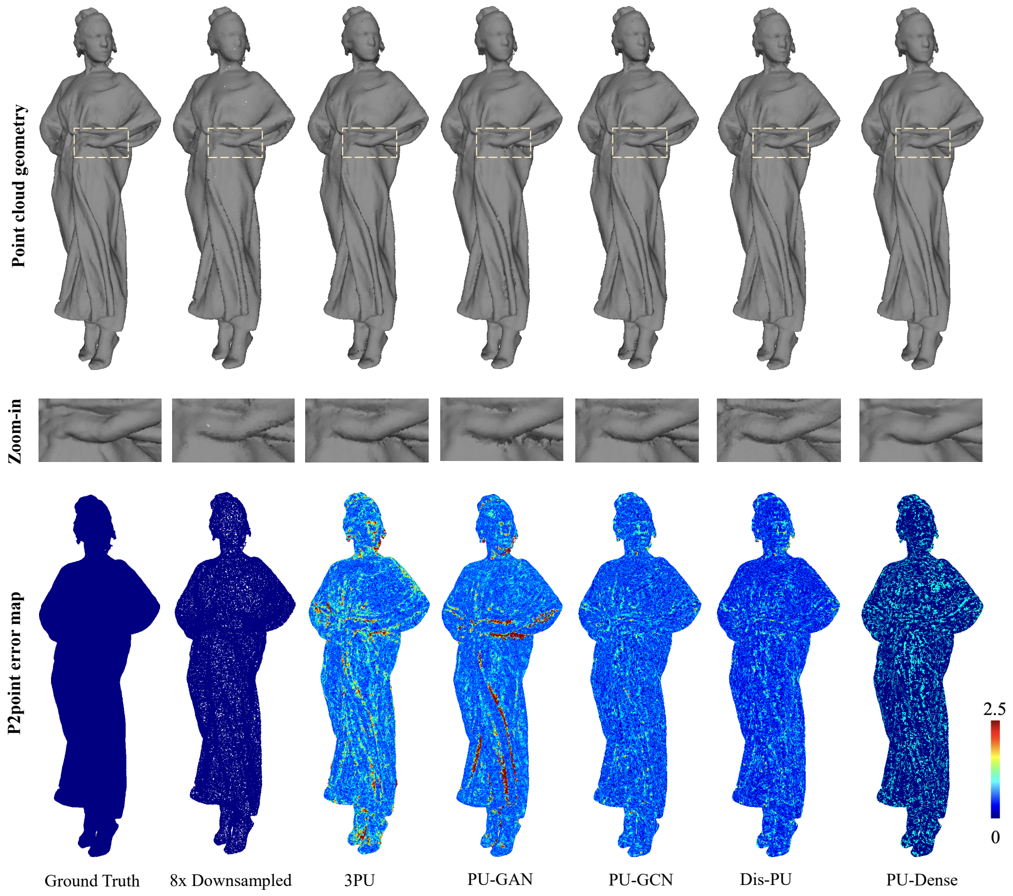


Figure 21: 8x upsampling visual results on sequence *Longdress*.

cloud datasets. In the upsampled point cloud generated by PU-Dense, the points generated are a lot more structured. This is due to the generation of new points using TG-Conv with binary voxel classification and pruning that generates newer points around the previously occupied voxels. We can notice that the holes in the point cloud get inpainted by the other methods, whereas, PU-Dense keeps the original shape of the point cloud. The reconstructed mesh surface quality for PU-dense is also better compared with the state-of-the-art. We can see PU-GCN and Dis-PU performs better than 3PU and PU-GAN on

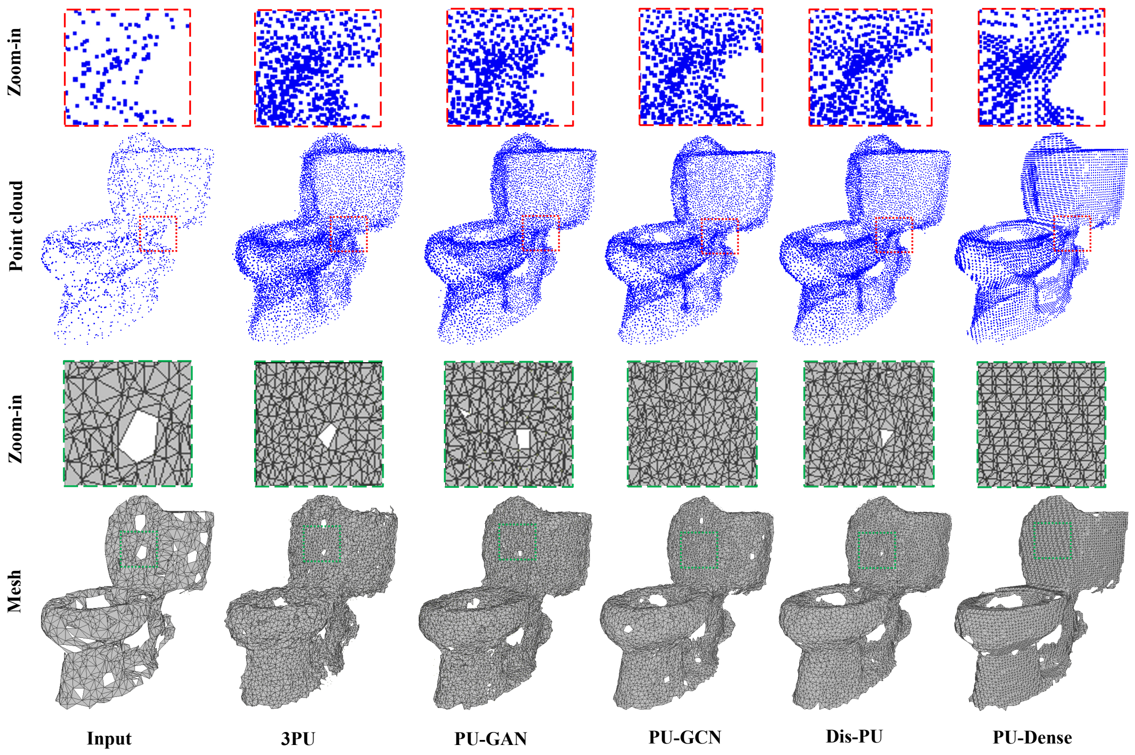


Figure 22: Visual comparison for 4x upsampling on ScanObjectNN dataset. Input is the real-scanned point cloud object with 2048 points. The rest of the five are the output for each of the methods: 3PU, PU-GAN, PU-GCN, Dis-PU, and PU-Dense (Ours). The top row shows the zoomed-in regions from the point cloud. The third row is the zoomed-in mesh surface from the mesh reconstructed surface from corresponding point clouds using the ball-pivoting algorithm.

this dataset.

4.4.8 Dynamically Acquired Outdoor LiDAR Dataset

We examine the performance of PU-Dense, as well as 3PU, PU-GAN, PU-GCN, and Dis-PU on dynamically-acquired outdoor LiDAR dataset from KITTI [34] used for autonomous driving. This dataset is much more sparse compared to 8iVFB, 8iVSLF,

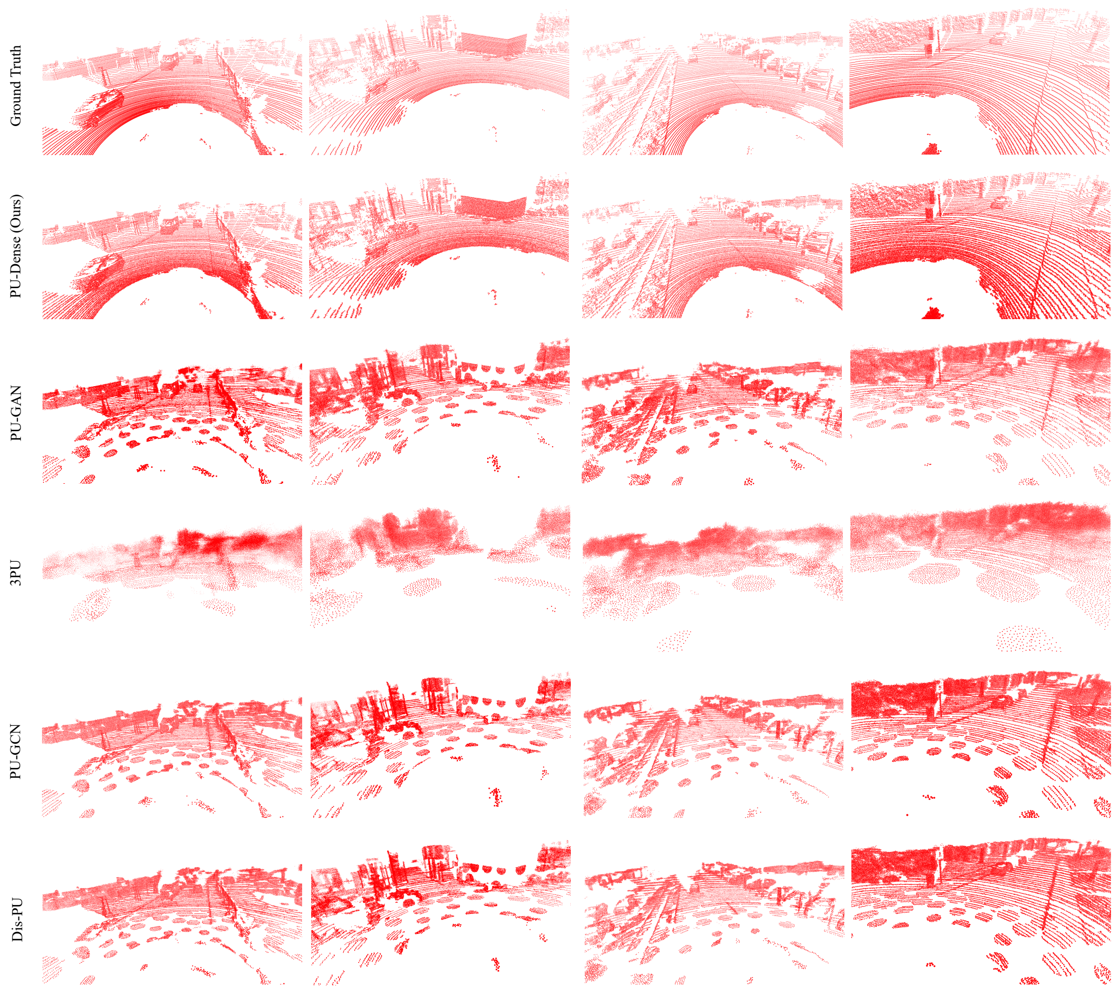


Figure 23: Visual comparisons for outdoor LiDAR dataset from KITTI for 4x upsampling.

and Queen datasets and also has non-uniform point distribution. We show the visual results from four street point clouds in Fig. 23. Even for sparse and non-uniform point clouds from the real-world LiDAR dataset, our proposed method can significantly improve the quality by upsampling these point clouds. In our experiments, all five methods are trained on regularly sampled and relatively dense ShapeNet dataset. Therefore, it is a considerable achievement for PU-Dense to be able to generalize to a sparse non-uniform dataset. Whereas the performance of the other methods suffers when evaluated on the KITTI dataset. This is because PU-Dense is fully convolutional making it translation-invariant that can process point clouds with a different number of points and sparsity levels. Whereas the other methods are patch-based solutions where the sparsity of the patch greatly influences the results. When the networks were trained on ShapeNet, the patch size was much smaller compared to an outdoor LiDAR point cloud. We can see that the patch-based upsampling results in the clustering of the points together within the patch. We believe this might be due to uneven sampling because of the *farthest point sampling* employed in these patch-based methods where not enough patches are sampled closer to the LiDAR resulting in clustering. PU-GCN and Dis-PU perform better than 3PU and PU-GAN on the KITTI dataset when trained on ShapeNet.

4.4.9 Mesh Generation

Fig. 24 shows the visualized results of 3D surface reconstruction using the ball-pivoting algorithm for the ground truth, input point cloud, and the upsampled point clouds from 3PU, PU-GAN, PU-GCN, Dis-PU, and PU-Dense on ShapeNet dataset. In the 3D mesh reconstruction task, the result is greatly influenced by the density as well as the quality of the upsampled point cloud. We can see the effectiveness of our proposed upsampling method by surface reconstruction in Fig. 24. For both 4x and 8x upsampling, surfaces reconstructed from PU-Dense upsampled point clouds are a lot more structured and exhibit richer geometry details. PU-Dense can recover more details and better preserve

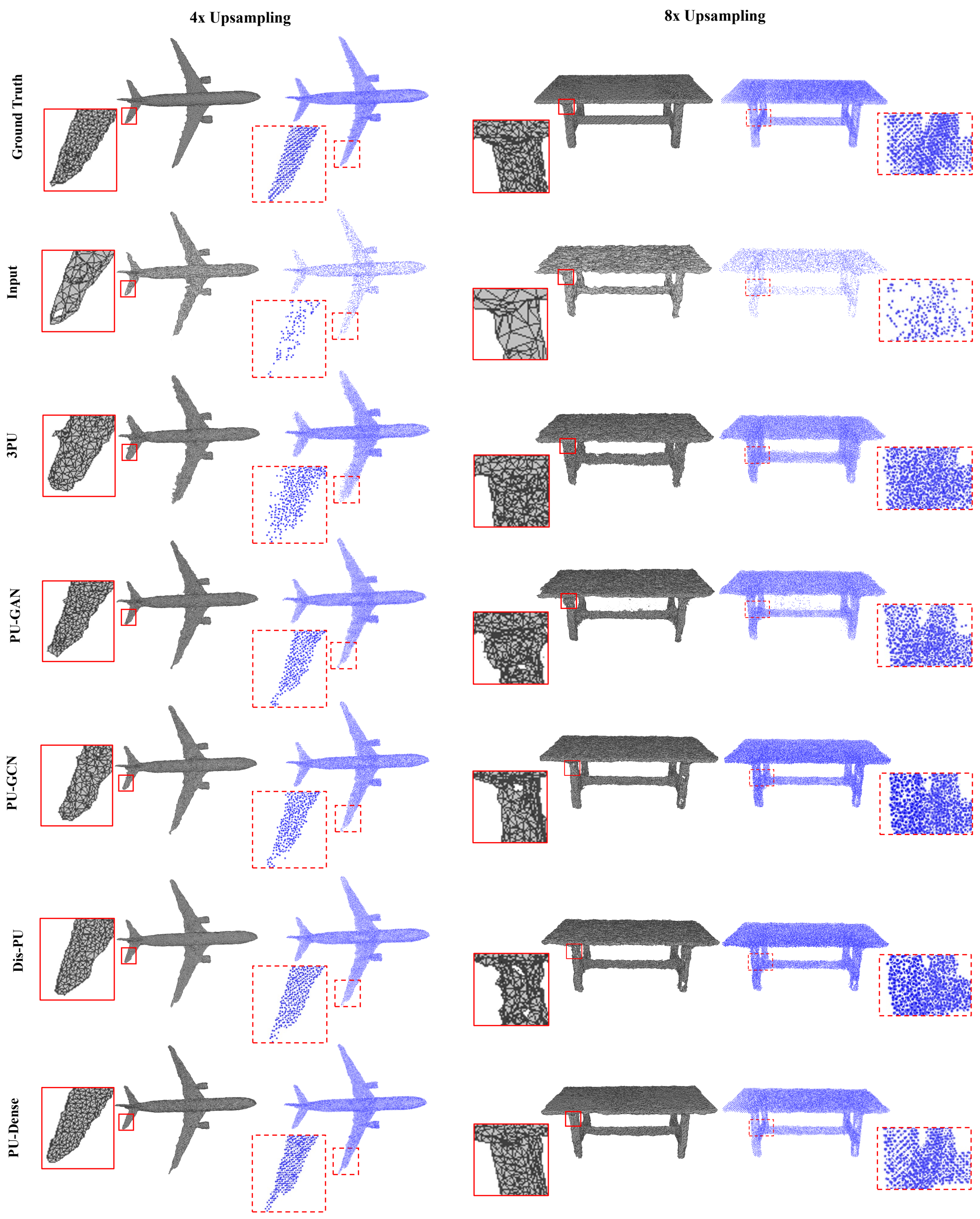


Figure 24: Visual comparisons for mesh reconstruction using ball-pivoting algorithm for 4x and 8x upsampling on ShapeNet dataset.

the smoothness of smooth regions as well as preserve the sharp edges without creating outliers. Although we notice that PU-GCN, and Dis-PU surface reconstruction qualities are much better compared to 3PU and PU-GAN. We also notice that PU-GAN produces a lot more outliers when it comes to 8x upsampling as seen in both Fig. 21 and Fig. 24.

4.4.10 Evaluation on Noisy Data

In this section, we evaluate the robustness of different methods to noise. Fig. 26 quantitatively compares the PU-Dense, 3PU, and PU-GAN on different levels of noise using ShapeNet dataset for 4x upsampling. We plot D1 MSE PSNR for each method under 7 levels of Gaussian noise. It can be seen that the performance of all methods decreases as the noise level increases. Nevertheless, the proposed method consistently achieves the best performance under each noise level. For PU-Dense, we notice the sharpest decrease when the noise is initially added and then the quality consistently decreases with an increase in noise. The robustness against noise can be further increased by augmenting noise during training of PU-Dense, something that was not employed during our training. We further show visual upsampling results for the PU-Dense on various noisy point clouds in Fig. 25. We visualize both 4x and 8x upsampling with 0%, 1%, and 2% Gaussian noise. We observe that even after adding noise, the upsampled point clouds visually look similar to the noise-free upsampled point cloud, demonstrating the robustness of the proposed method against noise.

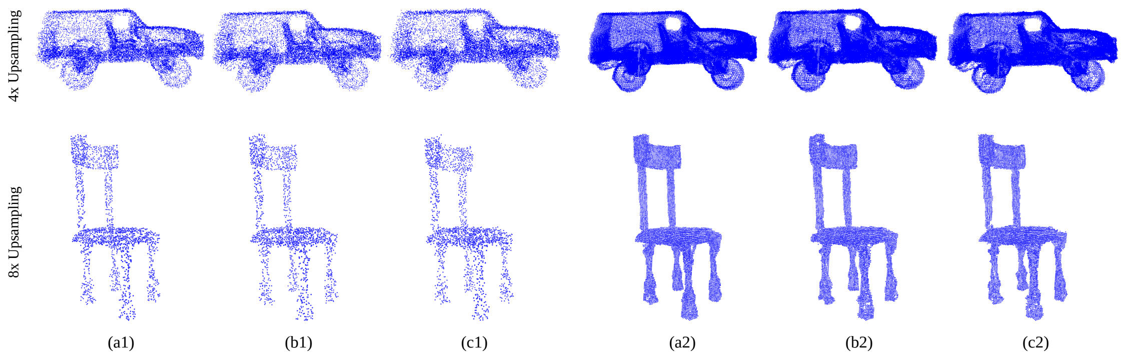


Figure 25: Visual results for PU-Dense on noisy data for 4x and 8x upsampling. Left: (a1), (b1), and (c1) are the sparse inputs with 0%, 1%, and 2% Gaussian noise, respectively. Right: (a2), (b2), and (c2) are the upsampled results from (a1), (b1), and (c1) respectively.

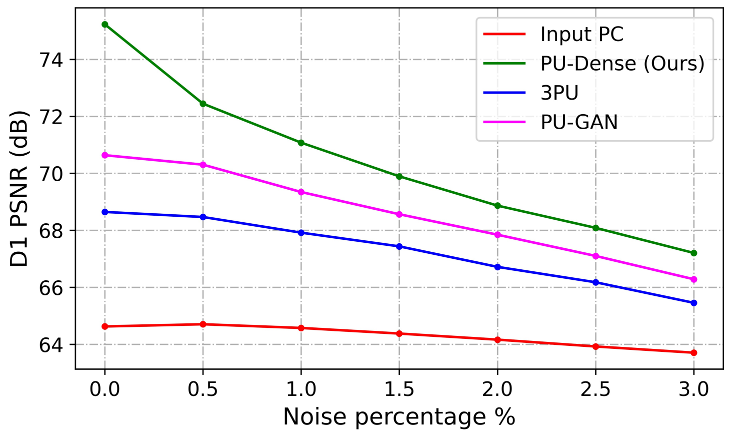


Figure 26: Quantitative comparisons on data with various levels of noise for 4x upsampling using ShapeNet dataset.

4.4.11 Inference Time and Trainable Parameters

We compare the average computational time as well as average inference time to 4x upsample a single 8iVFB point cloud in Table 11. The time in the experiments is calculated as the average time to process a single 8iVFB point cloud on NVIDIA GeForce

GTX 1080 Ti GPU. The computation time includes the whole end-to-end pipeline including the pre-processing and post-processing while the inference time includes the time required by the network to infer on all the patches. Note that the implementation of *3PU*, *PU-GAN*, *PU-GCN*, and *Dis-PU* is not optimized for larger point clouds. Since *3PU*, *PU-GAN*, *PU-GCN*, and *Dis-PU* are patch-based solutions, most of the computational time is spent in pre-processing and post-processing. This includes employing farthest point sampling to sample smaller overlapping patches from the surface of the point cloud and then processing these individual patches. After the processing, these patches are merged and then downsampled by again employing farthest point sampling. *PU-Dense* is a lot faster compared to other state-of-the-art in processing point clouds, because it employs efficient sparse convolutions, has limited pre-processing and post-processing, and employs an efficient binary voxel-classification loss all of which enables PU-Dense to process a large point cloud (up to 1 million points) in a single iteration without running out of GPU memory (NVIDIA GeForce GTX 1080 Ti).

We also compare the number of trainable parameters in each of the upsampling methods in Table 12. We can see that *PU-Dense* has many more trainable parameters and is much larger than the other networks. This is partly because the loss functions in other methods often employ very large matrix multiplications (e.g., Chamfer loss) that consume a lot of memory. Another advantage of PU-Dense is that it utilizes sparse convolution which takes advantage of the sparse nature of the point cloud and is, therefore, memory efficient. This allows PU-Dense to build a much deeper and wider model which was not possible in the previous works. PU-Dense is the first deeper point cloud upsampling

Table 11: Quantitative comparison: Average evaluation time per point cloud for 4x upsampling on 8iVFB dataset.

Upsampling Method	Inference Time (sec)	Computation Time (min)
3PU	123.76	24.49
PU-GAN	31.55	23.60
PU-GCN	**19.64**	23.20
Dis-PU	34.13	23.77
PU-Dense (Ours)	40.76	**00.79**

Table 12: Quantitative comparison: Number of trainable parameters.

Upsampling Method	Trainable parameters
3PU	152,054
PU-GAN	541,601
PU-GCN	75,971
Dis-PU	1,046,966
PU-Dense (Ours)	**13,172,441**

network whereas all the previous methods are much shallower. This makes PU-Dense more powerful, with higher discriminative power, as well as a larger receptive field. PU-Dense can process up to a million points per inference due to its memory efficiency.

4.4.12 Ablation Study

We perform a quantitative comparison of different versions of PU-Dense by removing specific components from the PU-Dense pipeline and tabulating the results in Table 13. First, we removed the *3D-DPB* from our *FE Unit* and replaced it with another *IRB*. This network is shown as *PU-Dense w/o 3D-DPB* in Table 13 and contains three *IRB*

Table 13: Ablation Study: removing specific components from the PU-Dense pipeline for 4x upsampling.

Dataset	Upsampling Method	CD (10^{-2})	MSE PSNR (dB)
8iVFB	PU-Dense w/o IRB, 3D-DPB	26.14	74.01
	PU-Dense w/o IRB	23.88	74.50
	PU-Dense w/o 3D-DPB	23.99	74.42
	PU-Dense	**19.38**	**75.05**

units in each *FE Unit*. Then, in the second method, we removed the *IRB* units from the *FE Unit* and replaced them with *3D-DPB*. We call this *PU-Dense w/o IRB*. Then, in the third method, we removed both the *IRB* unit as *3D-DPB* unit from the *FE Unit* and replaced them with multiple *ResNet* blocks with approximately the same number of parameters as in the original *FE Unit*. This network is shown as *PU-Dense w/o IRB, 3D-DPB* in Table 13 and contains three *ResNet* blocks per *FE Unit*. We can see that the complete PU-Dense pipeline gives the best performance and removing any one individual component greatly reduces the results, meaning that each component contributes to PU-Dense's efficacy. This also shows that employing both *3D-DPB* as well as *IRB* units for feature extraction yields the best results.

4.5 Limitations

In this section, we discuss the limitations of our proposed method and the future improvements that could be made to PU-Dense. We employ sparse tensors with 3D sparse convolutions along with a binary voxel classification loss. This makes PU-Dense computationally efficient allowing us to build a much larger network, process a dynamic number of input points, as well as allow PU-Dense to process high-resolution dense photo-realistic point clouds with millions of points. However, there is a downside to this approach. As a preprocess, we convert point clouds into sparse tensors by converting them into a 3D volumetric representation through voxelization. Sparse tensors allow us to only operate 3D convolutions on the occupied voxels while ignoring the empty voxels. We can populate the empty voxels using TG-Conv around an already occupied voxel and then prune them using binary voxel classification loss. The size of the kernel (ks) determines how far the newly occupied voxel could be from an already occupied voxel. This method works well for upsampling dense photo-realistic point clouds as well as sparse point clouds. However, this method would not be able to fill out larger holes that cannot be covered by the kernel size (ks). PU-Dense is not able to produce points at a location that has no nearby points within ks distance. For this reason, the current architecture would not work for inpainting tasks. However, this limitation can be easily rectified by employing multiple TG-Conv and pruning layers. We can simply replace all the T-Conv with TG-Conv + Pruning to be able to adapt this architecture for inpainting tasks.

4.6 Conclusion

In this paper, we propose a novel point cloud upsampling method called *PU-Dense* that can upsample both synthetic as well as real-world captured point clouds including

LiDAR scanned as well as dense high-resolution point clouds. PU-Dense incorporates hierarchical learning via progressive rescaling and multiscale feature extraction. PU-Dense utilizes a voxelized 3D representation with sparse convolutions backbone and introduces a novel *feature extraction (FE)* unit that contains *Inception-Residual Blocks* and a *3D Dilated Pyramid Block* to extract 3D multiscale features with different field-of-view in a computationally efficient manner. PU-Dense employs a memory-efficient voxel-based binary occupancy classification loss that can better reconstruct point clouds from features. Experimental results show that PU-Dense outperforms other state-of-the-arts by a significant margin on synthetic datasets, real-world LiDAR scanned datasets, as well as dense photo-realistic point clouds for immersive communication. While the other state-of-the-arts struggle to efficiently process high-resolution point clouds with a large number of points, PU-Dense can effectively learn 3D features and produce higher level-of-detail upsampled point clouds for both synthetic as well as real-world datasets. PU-Dense generates limited outliers while populating points very close to the surface of the ground truth point cloud resulting in a high-resolution point cloud with considerably better quality than the other state-of-the-arts. Furthermore, PU-Dense is more robust against Gaussian noise compared to other methods.

CHAPTER 5

DYNAMIC POINT CLOUD INTERPOLATION

5.1 Introduction

Point clouds can be categorized as point cloud scenes and point cloud objects. Point cloud scenes are typically dynamically acquired using LiDAR (light detection and ranging) sensors and are commonly used in autonomous vehicles [8]. Point cloud objects can be further subdivided into static objects and dynamic objects. A static object is a single object, whereas a dynamic object is time-varying, where each instance of a dynamic point cloud is a static point cloud. Dynamic time-varying point clouds are used in AR/VR, volumetric video, and telepresence and can be generated using 3D models, i.e., CGI, or captured from real-world scenes using various methods such as multiple cameras with depth sensors surrounding the object and capturing movement over time. Frame rates of LiDARs are generally 10 to 20 Hz, resulting in a lower resolution, spatially and temporally sparse point cloud [9]. However, dynamic point clouds are denser photo-realistic point clouds that contain a lot more points with high data rates. For example, a single instance of dynamic point cloud captured by 8i [59] contains between 1 million to 4 million points per frame which translates to a bitrate of around 1 Gbytes per second without compression for a 30 fps dynamic point cloud. The high data rate is one of the main problems faced by dynamic point clouds, and efficient interpolation techniques to synthesize intermediate frames would help in the distribution, processing, and compression of such content [6].

Video frame interpolation is commonly utilized in frame rate conversion, novel view synthesis, video streaming, and video compression pipeline to generate high frame rate videos from low frame rate ones (e.g. from 30 Hz to 240 Hz). Typical video frame interpolation methods [13, 77–79] perform two tasks: motion estimation, usually optical flow, and pixel synthesis. However, these methods cannot be directly applied to point clouds since 3D point clouds are unstructured and unordered. There are no direct correspondences between points in two point clouds like pixels in two images. The sparsity and large size of point clouds further complicate the point cloud interpolation problem. There has been limited work on 3D point cloud interpolation and all work has been performed on dynamically acquired LiDAR-based point cloud scenes. While optical flow represents 2D pixel movements on the image plane, 3D scene flow represents per point 3D movement. Optical flow can be considered the projection of scene flow into 2D. FlowNet3D [70] is a pioneering work of deep learning-based 3D scene flow estimation. FlowNet3D proposed a flow embedding layer to model the motion of points in different point cloud scenes. Following FlowNet3D, FlowNet3D++ [119] proposed geometric constraints in the form of point-to-plane distance and angular alignment to further improve the accuracy of scene flow estimation. There have been further works [39, 76, 121] that explore point cloud scene flow estimation or interpolation. PointINet [71] estimates bi-directional 3D scene flows, performs frame warping, followed by point fusions and intermediate point cloud generation. Even though these methods make a decent baseline in point cloud scene flow estimation, they are only limited to LiDAR-based point cloud scenes and are not applicable to photo-realistic dynamic point clouds.

Compared to dense dynamic point clouds, LiDAR point clouds tend to be sparser, with a lower frame rate and a smaller number of points. Since point cloud scenes are dynamically acquired, there is more scene rigidity as well as point correspondences. However, photo-realistic dynamic point clouds tend to be denser where the object moves and changes shape making the point correspondences difficult and thus the scene flow methods are inapplicable to dynamic point clouds. Moreover, the large size of the photo-realistic dynamic point clouds makes the task further challenging [7]. To address these issues we propose a first of its kind dynamic point cloud interpolation framework. Given two consecutive dynamic point cloud frames, our framework aims to generate intermediate frame(s) between them. We propose three different modules: the encoder network, the fusion network, and the multi-scale point cloud synthesis module. The encoder module extracts features from frames at four different scales. The fusion network takes features at different scales from consecutive frames, concatenates them into 4D features, then utilizes 4D convolutions to merge consecutive frame features. Finally, the multi-scale point cloud synthesis module hierarchically interpolates the target frame at different resolutions.

5.2 Point Cloud Interpolation

In this section, we first introduce the overall architecture of the proposed point cloud interpolation network and then explain the details of the key components of our framework. The overall system model is shown in Fig. 27. Given two point cloud frames $F^1 \in \mathbb{R}^{N\times 3}$ and $F^3 \in \mathbb{R}^{N\times 3}$, the goal is to predict the intermediate point cloud frame $\hat{F}^2$.

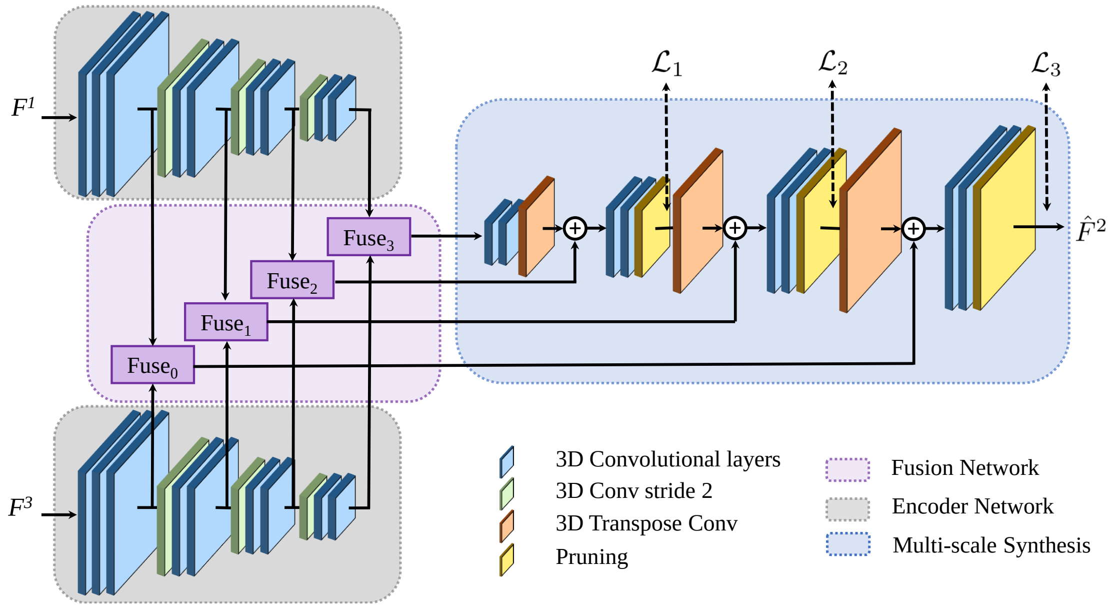


Figure 27: System Model. F^1 and F^3 are the input frames while $\hat{F}^2$ is the interpolated frame.

5.2.1 Network

The proposed framework contains three different modules: the encoder network, the fusion network, and the multi-scale point cloud synthesis module. We use sparse tensors and sparse convolution using Minkowski Engine [20]. The input frames are pre-processed where they are voxelized and converted into sparse tensors.

The Encoder Module. We employ the encoder module for multi-scale point cloud feature extraction. This module learns the point cloud features at four different scales for both frames F^1 and F^3. Both encoder modules shown in Fig. 27 are identical and share the same weights. As shown in the evaluation results, a pre-trained encoder module performs much better than learning the weights of the encoder along with the rest of the network. We pre-train our encoder module in a typical encoder-decoder architecture using reconstruction loss (binary voxel occupancy loss) on a static point cloud objects dataset (ShapeNet).

The Fusion network. The fusion network utilizes a novel 4D fuse block that merges features from two consecutive point cloud frames into a single feature. The fusion module takes features at four different scales from frames F^1 and F^3. Different scale features are processed individually by a fuse block as shown in Fig. 27. The goal here is to merge features at the same scale together. The individual fuse block is shown in Fig. 28. In the fuse block, the features pass through 3D convolutions and then are concatenated in the 4^{th} dimension resulting in the feature size of $(x, y, z, 2)$. Where x, y, z is the size of the *x, y* and *z* coordinates respectively. The 4D features are then passed through 4D convolutions so the inter-frame features could be learned. Afterward, a 4D convolution with stride in only the 4^{th} dimension $(stride = [1, 1, 1, 2])$ is applied. This convolution acts as a learnable pooling in a single dimension where the resulting feature size becomes $(x, y, z, 1)$. This is converted back into 3D features so the 3D convolutions can be applied. Since 4D convolutions tend to have higher computational complexity and extra memory consumption, we tend to limit the amount of 4D convolutions we employ.

Multi-scale Point Cloud Synthesis Module. The fused features at four different scales are fed into the multi-scale point cloud synthesis module that hierarchically interpolates the intermediate frame $\hat{F}^2$. Sparse transpose convolutions are used to upscale the features. After each upscaling, the features from the fuse block of the same scale are added to the synthesis module network. Upscaling using transpose convolution generates

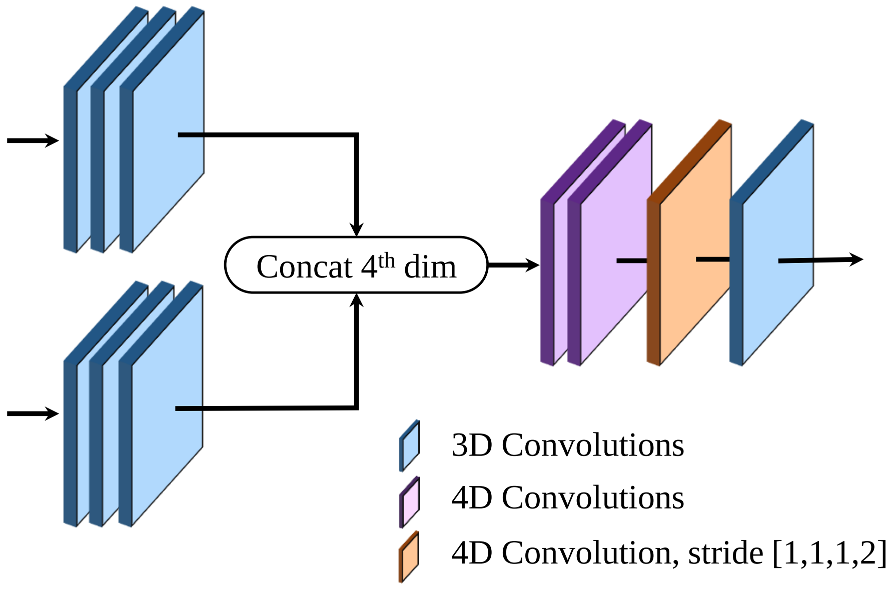


Figure 28: A single fuse block from the fusion network.

a lot of new points. To choose the best points and dispose of invalid points, pruning is performed. The pruning layer implements classification by converting $N \times C$ features to $N \times 1$ features and choosing the best features above a threshold.

5.2.2 Loss Function

Rather than upscaling the features directly to the full scale, our network synthesizes the interpolated point cloud at different resolutions by employing multiple loss functions. The multi-scale synthesis module produces the interpolated point cloud at three different resolutions. As shown in Fig. 27, we train our network using three different loss functions:

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 \quad , \tag{5.1}$$

We perform voxel classification using binary cross-entropy loss to compare the voxel occupancy prediction from the network and the ground truth (original) point cloud. The ground truth point cloud is downscaled to three resolutions, one for each loss function.

5.3 Evaluation Results

In this section, we will go over our datasets, the training environment, evaluation metrics employed, and both the quantitative and visual results of our dynamic point cloud interpolation framework.

5.3.1 Datasets

The encoder network is pretrained before being utilized in the framework. The encoder is pretrained on **ShapeNet** dataset which is a static objects dataset. We randomly selected ≈ 24000 3D mesh models from the core dataset of ShapeNet. The mesh model is sampled into point clouds by randomly generating points on the surfaces of the mesh, then randomly rotated and quantized the point cloud to 7-bit precision. For dynamic point clouds we employed sequences *loot, longdress* and *redandblack* from JPEG Plenos 8i Voxelized Full Bodies dataset (**8iVFB v2**) [23] with about a million points per frame. We also used sequences *basketball* and *exercise* from MPEGs 8i Voxelized Surface Light Field dataset (**8iVSLF**) [59] with about 2.6 million points per frame. Finally we used the sequence *Queen* produced by **Technicolor** (https://www.technicolor.com/fr) with about 1 million points per frame. We train on *loot* and *longdress* sequences and test on the rest of the four sequences.

5.3.2 Training

The encoder architecture is pretrained on ShapeNet dataset. We train the rest of the framework on *loot* and *longdress* sequences. To be able to feed our network three frames each containing about a million points per frame, we subdivide the point cloud using kd-tree partitioning. Points are extracted from the same locations within the cube across multiple frames. In our training, we use a kd-tree depth of 4 to divide each frame into 16 cubes. During evaluation, the whole point cloud can be fed into the network.

5.3.3 Evaluation Metrics

We consider two commonly-used evaluation metrics that compare the reconstructed point cloud to the ground truth point cloud to quantitatively evaluate the performance of our method. These metrics are Chamfer distance (CD) and point-to-point (D1) based mean squared error peak signal-to-noise ratio (MSE PSNR). MSE PSNR has been adopted by both MPEG and AVS standards as an evaluation metric for dynamic point cloud quality [106]. We obtain the point-to-point geometry PSNRs using MPEG's *pc_error* tool [107].

5.3.4 Objective Evaluation and Visual Results

We compare our method with different variations of our framework as well as with **Identity** where we simply duplicate the first point cloud frame as the intermediate point cloud. The objective results of our evaluations are shown in Table 14. **Ours** is the framework described in this paper. **Ours-w/o Pretrained** is the framework where

Method	*redandblack*		*queen*		*basketball*		*exercise*		Avg	
	CD↓	PSNR↑	CD↓	PSNR↑	CD↓	PSNR↑	CD↓	PSNR↑	CD↓	PSNR↑
Identity	1623.69	53.68	45.29	70.75	113.65	71.73	146.54	71.19	482.29	66.84
Ours	**386.22**	**56.88**	**30.44**	**76.08**	**80.96**	**74.54**	**110.20**	**75.48**	**151.96**	**70.75**
Ours-w/o Pretrained	502.86	55.44	35.64	74.81	90.37	73.30	117.62	74.34	186.63	69.47
Ours-Single Loss	575.91	55.40	36.64	74.00	89.74	73.03	121.54	73.37	205.96	68.95
Ours-Fuse3D	865.72	54.64	37.01	73.64	100.18	72.14	125.47	73.12	282.10	68.39

Table 14: Evaluation results of our interpolation method using Chamfer Distance (CD (10^{-2})) and MSE PSNR (dB).

the Encoder is not pretrained on ShapeNet. **Ours-Single Loss** framework employs only a single loss ($\mathcal{L}_3$) and no longer employs losses $\mathcal{L}_1$ and $\mathcal{L}_2$. **Ours-Fuse3D** framework utilizes 3D convolutions in the fuse block rather than the 4D convolutions. In this method, the features are simply added and 3D convolutions are employed. As can be seen, our method considerably performs better than *Identity*. Furthermore, the results show that each sub-component of our framework is essential and improves the results considerably. The results show that the fusion using 4D convolutions is much more efficient in learning the intermediate frame features compared to 3D convolutions.

An example of visual results for sequence *redandblack* is shown in Fig. 29. We can see that our method generates limited outliers while populating points very close to the surface of the ground truth point cloud. This results in a high-resolution point cloud with considerably better quality than the *Identity* where the previous frame is used.

5.4 Conclusion

In this work, we propose the first dynamic point cloud interpolation framework for dense high-resolution point clouds. While the previous point cloud interpolation methods

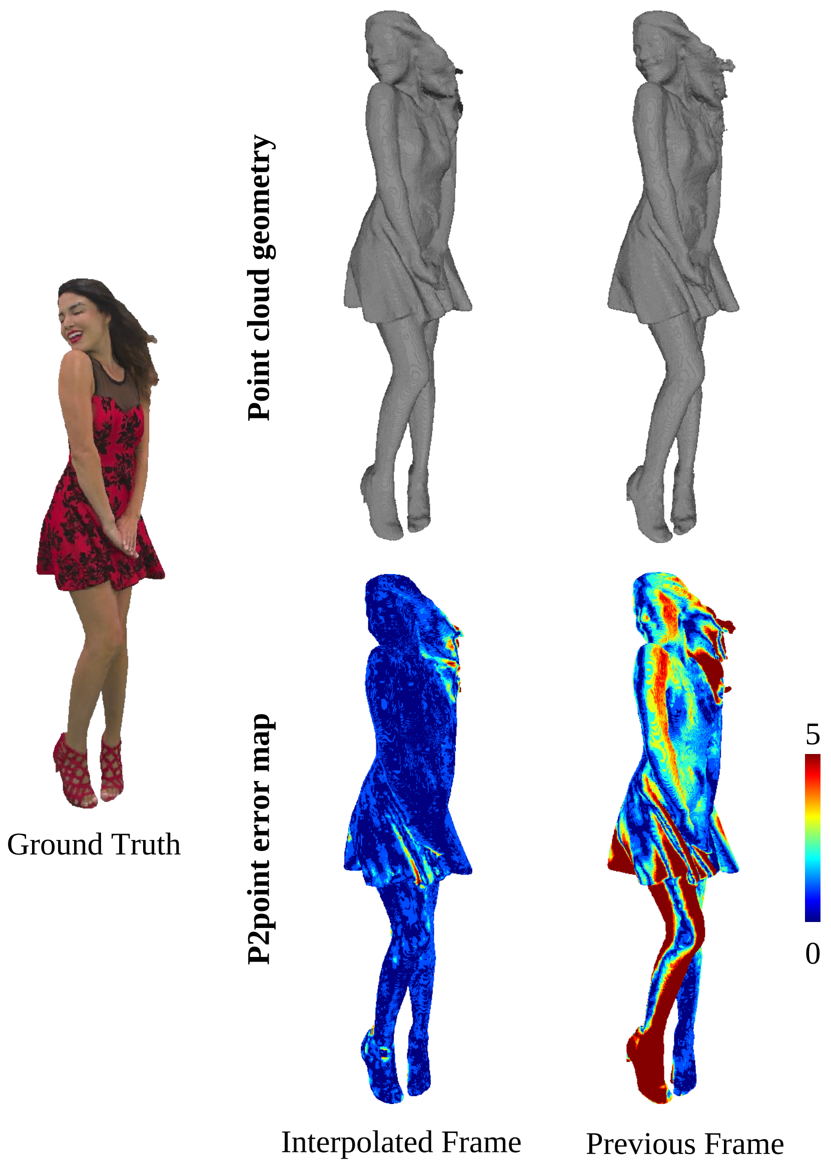


Figure 29: Visual Results on *Redandblack* dataset.

are limited to point cloud scenes, our framework is able to process and interpolate frames on a high-resolution dynamic point cloud. We employ a pretrained multi-scale encoder module to extract features at multiple scales. The encoder module is pre-trained on a static object dataset (ShapeNet). We introduce a novel 4D feature fusion module that utilizes 4D learning to merge 3D features from two consecutive frames at multiple scales. Finally, our multi-scale point cloud synthesis module hierarchically reconstructs the interpolated point cloud frame at different resolutions. We test our framework on a diverse set of high-resolution dynamic point cloud sequences. The evaluation results validate our network design and demonstrate the effectiveness of our method.

CHAPTER 6

INTER-FRAME COMPRESSION FOR DYNAMIC POINT CLOUD GEOMETRY CODING

6.1 Introduction

A point cloud (PC) is a 3D data representation that is essential for tasks like virtual reality (VR) and mixed reality (MR), autonomous driving, cultural heritage, etc. PCs are a set of points in 3D space, represented by their 3D coordinates (*x, y, z*) referred to as the *geometry*. Each point may also be associated with multiple *attributes* such as color, normal vectors, and reflectance. Depending on the target application and the PC acquisition methods, the PC can be categorized into point cloud scenes and point cloud objects. Point cloud scenes are typically captured using LiDAR sensors and are often dynamically acquired. Point cloud objects can be further subdivided into static point clouds and dynamic point clouds. A static PC is a single object, whereas a dynamic PC is a time-varying PC where each instance of a dynamic PC is a static PC. Dynamic time-varying PCs are used in AR/VR, volumetric video streaming, and telepresence and can be generated using 3D models, i.e., CGI, or captured from real-world scenarios using various methods such as multiple cameras with depth sensors surrounding the object. These PCs are dense photo-realistic point clouds that can have a massive amount of points, especially in high precision or large-scale captures (millions of points per frame with up to 60 frames per second (FPS)). Therefore, efficient point cloud compression (PCC) is particularly important to enable practical usage in VR and MR applications.

The Moving Picture Experts Group (MPEG) has approved two PCC standards [38, 97]: Geometry-based Point Cloud Compression (G-PCC) [4] and Video-based Point Cloud Compression (V-PCC) [5]. G-PCC edition 1 includes octree-geometry coding as a generic geometry coding tool and a predictive geometry coding (tree-based) tool which is more targeted toward LiDAR-based point clouds. G-PCC is still developing a triangle meshes or triangle soup (trisoup) based method to approximate the surface of the 3D model. V-PCC on the other hand encodes dynamic point clouds by projecting 3D points onto a 2D plane and then uses video codecs, e.g., High-Efficiency Video Coding (HEVC), to encode each frame over time. MPEG has also proposed common test conditions (CTC) to evaluate test models [96]. For quantitative evaluations, CTC employs point-to-point (D1) and point-to-plane (D2) quality metrics.

Deep learning solutions for image and video encoding have been widely successful [69]. Recently, similar deep learning-based PCC methods [14, 27, 31, 41, 51, 90–92, 112–114, 126] have been shown to provide significant coding gains over traditional methodologies. Point cloud compression represents new challenges due to the unique characteristics of PC. For instance, the unstructured representation of PC data, the sparse nature of the data, as well as the massive number of points per PC, specifically for dense photo-realistic PC, makes it difficult to exploit spatial and temporal correlation. The current deep learning-based PCC solutions are all intra-prediction methods for static point clouds and fail to utilize inter-prediction coding gains by predicting the current frame using previously decoded frames. Following MPEG's PCC category guidelines, our work seeks to target dense dynamic point clouds used for VR/MR and immersive telecommunications. Sparse dynamically acquired LiDAR-based point clouds are a very different point cloud category that is out of the scope of this work. Ours is the first solution for inter-prediction for lossy point cloud geometry encoding using deep learning and has the following novelties:

- We propose a novel deep learning-based framework for point cloud geometry inter-frame encoding similar to P-frame encoding in video compression.
- We propose a novel predictor module that learns a feature embedding of the current PC frame from the previous PC frame. The network utilizes hierarchical multiscale feature extractions and employs ”*convolution on target coordinates*” to map latent features from the previous frame to the downsampled coordinates of the current frame to learn the current frame's feature embedding.
- To the best of our knowledge, our method is the first deep learning-based method to outperform V-PCC inter-frame mode across all bitrates. Experimental results show our method achieves more than 91% BD-Rate gains against G-PCC (octree), more than 84% BD-Rate gains against G-PCC (trisoup), more than 34% BD-Rate gains against state-of-the-art deep learning-based point cloud geometry compression method, more than 62% BD-Rate gains against V-PCC intra-frame mode, and more than 52% BD-Rate gains against V-PCC P-frame-based inter-frame mode which uses HEVC.

6.2 Background

Our research is most closely related to three research topics: point cloud geometry compression, deep learning-based video inter-frame coding, and deep learning-based point cloud compression.

Prior non-deep learning-based point cloud geometry compression mostly includes *octree-based, triangle mesh-based*, and *3D-to-2D projection-based* methodologies. **Octree-based methods** are the most widely used point cloud encoding methods [75, 95, 111]. Octree provides an efficient way to partition the 3D space to represent point clouds and is especially suitable for lossless coding. In these methods, the volumetric point cloud is recursively divided into octree decomposition until it reaches the leaf nodes. Then the occupancy of these nodes can be compressed through an entropy context modeling conditioned on neighboring and parent nodes. Thanou et al. [104, 105] implemented octree-based encoding for time-varying point clouds that can predict graph-encoded octree structures between adjacent frames. MPEG's G-PCC standard [97] also employs an octree-based compression method known as *octree geometry codec* and is specifically devoted to sparse point clouds. G-PCC encoding can further be complemented by triangle meshes (a.k.a., triangle soups) which are locally generated together with the octree to terminate the octree decomposition prematurely. This helps reconstruct object surfaces with finer spatial details and is known as the *trisoup geometry codec* [16].

3D-to-2D projection-based methods. Traditional 2D image and video coding have demonstrated outstanding efficiency and have been widely used in standards which have motivated works to project 3D objects to multiple 2D planes and leverage popular image and video codecs for compact representation. MPEG's V-PCC [38] standard is one such 3D-to-2D projection-based solution that is specifically designed for dense, as well as, dynamic PCs. The V-PCC standard projects the points and the corresponding attributes onto planes and then uses a state-of-the-art video codec, such as HEVC, to encode point clouds. V-PCC has both intra-frame coding as well as inter-frame coding where the previously decoded frames are employed to encode the next frames. We have also had some other works specifically for **dynamic point cloud compression** [22,33,35]. However, their results are still lacking and not comparable to V-PCC.

Deep learning-based models for image and video encoding can learn an optimal non-linear transform from data along with the probabilities required for entropy coding the latent representation into a bitstream in an end-to-end fashion. For image compression, autoencoders [11] were initially adopted and the best results were achieved by employing variational autoencoders with side information transmission and applying an autoregressive model [12]. Deep learning solutions for video compression methods usually employ 3D autoencoders, frame interpolation, and/or motion compensation via optical flow. 3D autoencoders are an extension of deep learned image compression. Frame interpolation methods use neural networks to temporally interpolate between frames in a video and then encode the residuals [79]. Motion compensation via optical flow is based on estimating and compressing optical flow which is applied with bilinear warping to a previously decoded frame to obtain a prediction of the frame currently being encoded [78]. Current deep learning-based PCC takes inspiration from the deep learning-based image compression methods but so far has not been able to implement inter-frame prediction models commonly used in video encoding. Our work is the first method that takes inspiration from the frame interpolation-based methods in video encoding to perform inter-frame encoding for dynamic point clouds.

Deep learning-based Geometry PCC can be broadly categorized into: *voxelization-based methods, octree-based methods, point-based methods*, and *sparse tensors-based methods*. **Voxelization-based methods** were employed in the earlier approaches, including Quach et al. [90], Wang et al. [114], Guarda et al. [41] and Quach et al. [91]. These methods voxelizes the PC and then divide it into smaller blocks typically of $64 \times 64 \times 64$ voxels. Then 3D convolutions are applied using autoencoder architectures to compress these blocks into latent representations. These methods usually employ a focal loss or a weighted binary cross-entropy loss to train their model. However, these methods also have to process empty voxels which are usually the majority of the voxels and are, therefore, computational and memory inefficient.

Octree-based methods employ octree representation to encode the PCs leading to better consumption of storage and computation. These methods employ entropy context modeling to predict each node's occupancy probability conditioned on its neighboring and parent nodes. MuSCLE [14] and OctSqueeze [51] employ Multi-Layer Perceptrons (MLPs) to exploit the dependency between parent and child nodes. VoxelContextNet [92] employs both neighbors and parents as well as voxelized neighborhood points as context for probability approximation. Recently, OctAttention [27] has been introduced that increases the receptive field of the context model by employing a large-scale transformer-based context attention module to estimate the probability of occupancy code.

All of these methods encode the point cloud in a lossless manner and show promising results, particularly on sparse LiDAR-based point clouds.

Point-based methods directly process raw point cloud data without changing their representation or voxelizing them. They typically employ PointNet [84] or PointNet++ [85] type architectures that process raw point clouds using point-wise fully connected layers. These methods are typically patch-based methods that employ farthest point sampling to subsample and a knn search to find per point feature embedding to build an MLP-based autoencoder. However as seen in some of these works [31, 53, 126], the coding efficiency of such point-wise models is still relatively low and fails to generalize to large-scale dense point clouds. Furthermore, these methods require a lot of pre and post-processing making the encoding process computationally inefficient.

Recent **sparse convolution-based methods** [112, 113] have shown really good results especially for denser photo-realistic point clouds. Sparse convolutions exploit the inherent sparsity of point cloud data for complexity reduction allowing for very large point clouds to be processed by a deeper sparse convolutional network. This allows the network to better capture the characteristics of sparse and unstructured points and better extraction of local and global 3D geometric features. However, all of these works employ intra-frame encoding for static point clouds. We employ sparse convolution-based autoencoder architecture similar to [113] and design a sparse convolutional inter-frame prediction module that encodes the next PC frame using the previously decoded PC frame similar to P-frame prediction in video encoding.

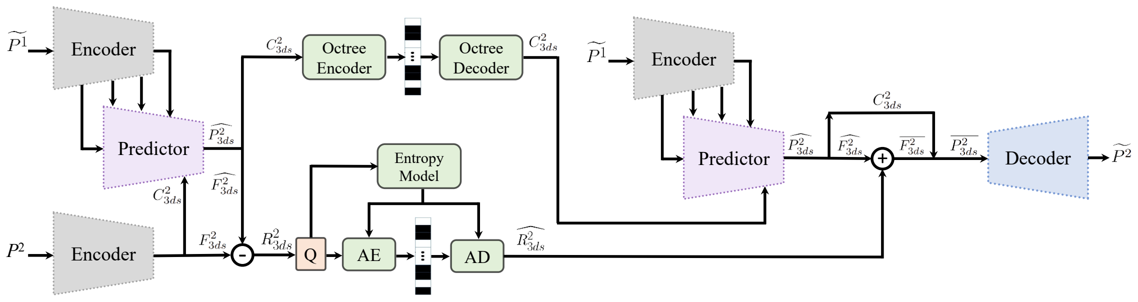


Figure 30: System Model. The previously decoded frame $\widetilde{P^1}$ is employed to encode a feature embedding of the current frame P^2. Multiscale features from $\widetilde{P^1}$ and three-times downsampled coordinates C^2_{3ds} from P^2 are passed to the Predictor network to learn a feature embedding $\widehat{P^2_{3ds}} = \{C^2_{3ds}, \widehat{F^2_{3ds}}\}$. Current frame's three-times downsampled coordinates C^2_{3ds} are transmitted in a lossless manner using octree encoder. The predicted downsampled features $\widehat{F^2_{3ds}}$ and the original downsampled features F^2_{3ds} are subtracted to obtained the residual features R^2_{3ds}. The residual is transmitted in a lossy manner using a learned entropy model. The same Encoder and Predictor module is used throughout the system.

6.3 Proposed Method

The proposed lossy inter-frame point cloud geometry compression framework is illustrated in Fig. 30. We employ sparse tensors and sparse convolutions to decrease the computational complexity of the network so it can process two PC frames. The solution is inspired by the PCGCv2 [113] multiscale point cloud geometry compression (PCGC) work. PCGCv2 is an intra-frame point cloud compression scheme suitable for static point clouds. Our inter-frame prediction scheme uses an encoder and decoder network similar to PCGCv2 along with a novel prediction network to predict a feature embedding for the current PC frame from the previous PC frame. We calculate the residual between the predicted and ground truth features and transmit the residuals along with the three-times downsampled coordinates. The three-times downsampled coordinates are losslessly encoded by an octree encoder using G-PCC [4], whereas the residual features are encoded in a lossy manner using factorized entropy model to predict the probability distribution for arithmetic coding. It should be noted that in our system, the encoder and prediction network is present both at the transmitter as well as the receiver. We train the networks with joint reconstruction and bit-rate loss to optimize rate distortion. We provide a detailed description of all our modules in subsequent discussions.

6.3.1 Problem Formulation and Preprocessing

We adopt sparse convolutions for low-complexity tensor processing and build our system using Minkowski Engine [20]. Each point cloud frame is converted into a sparse tensor P. Each point cloud tensor $P = \{C_n, F_n\}_n$ is represented by a set of coordinates $C = \{(x_n, y_n, z_n)\}_n$ and their associated features $F = \{f(x_n, y_n, z_n)\}_n$. Only the occupied coordinates are kept in a sparse tensor. To initialize the input point cloud as geometry only, we assign feature $f(x, y, z) = 1$ to each occupied coordinate. Given a dynamic point cloud with multiple frames, P^i, our goal is to convert them into a latent representation with the smallest possible bitrate. We use P-frame encoding where the current frame is encoded using the prediction from the previous frame. We denote the Encoder network as E, and the Decoder network as D.

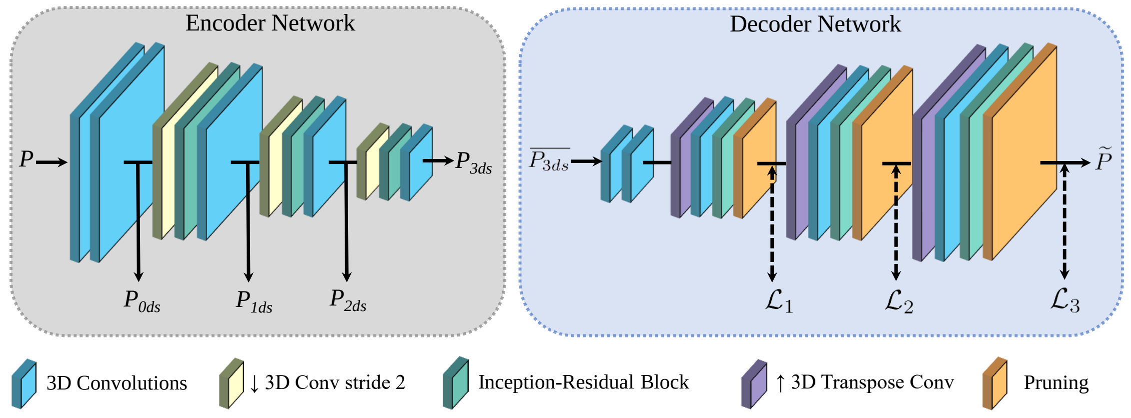


Figure 31: Encoder and Decoder Network. The encoder network takes the original point cloud sparse tensor P, and creates sparse features at four different scales: P_{0ds}, P_{1ds}, P_{2ds}, and P_{3ds}. Where $3ds$ denotes three-times downsampled sparse tensor. The decoder network takes the three-times downsampled sparse tensor and hierarchically reconstructs the original point cloud by progressively rescaling. The decoder upsamples the sparse tensor followed by a pruning layer to prune false voxels.

6.3.2 Encoder, Decoder, and Pruning

Our encoder and decoder network is shown in Fig. 31. We utilize the Inception-Residual Block (IRB) from PCGCv2 [113] for feature extraction in all our networks. We employ a multiscale re-sampling with downscaling at the encoder and upscaling at the decoder. This helps exploit the sparsity of the PC while encoding 3D geometric structural variations into feature attributes of the latent representation. The encoder helps us obtain PC tensors at four different scales capturing multiscale features at different level of details: $P_{0ds}, P_{1ds}, P_{2ds}, P_{3ds} = E(P)$. Where P_{ids} represents a sparse tensor P that has been downsampled i times. The decoder receives a three-times downsampled PC tensor and upsamples it hierarchically to reconstruct the original PC tensor by employing a different reconstruction loss at each scale: $\widetilde{P} = D(P_{3ds})$. Decoder employs transpose convolution to upsample the PC tensor. Encoder and decoder architecture can on their own be used for intra-frame PC compression, the details of which are available from PCGCv2 [113].

The geometry at the decoder is reconstructed by employing a pruning layer to prune false voxels and extract true occupied voxels using binary classification after each upscaling. One example of a pruning layer is shown in Fig. 32. In this example, the input sparse tensor P_a has coordinates C_a of shape $139{,}244 \times 3$ and their corresponding features of shape $139{,}244 \times 64$. We pass P_a through a convolution of channel size 1 to obtain sparse tensor P_b with features F_b of shape $139{,}244 \times 1$. From F_b we select the topk features (in this example $k = 52{,}612$) and their corresponding coordinates using binary classification. The false coordinates and their corresponding features are then pruned from P_a to obtain P_c. During training, the binary voxel classification loss is applied to P_b to learn the proper point cloud reconstruction.

6.3.3 Overall System Model

The overall working of the proposed method is shown in Fig. 30. In our work, we denote the current PC frame as P^2 while the previously decoded PC frame is denoted by $\widetilde{P^1}$. The same encoder and prediction module are used throughout the system to decrease the number of parameters. Previously decoded frame $\widetilde{P^1}$ is passed through the encoder to obtain multiscale features $P^1_{0ds}, P^1_{1ds}, P^1_{2ds}, P^1_{3ds}$. The current frame P^2 is also passed through the encoder to obtain three-times downsampled tensor containing coordinates and features: $P^2_{3ds} = \{C^2_{3ds}, F^2_{3ds}\}$. Current frame's three-times downsampled coordinates and

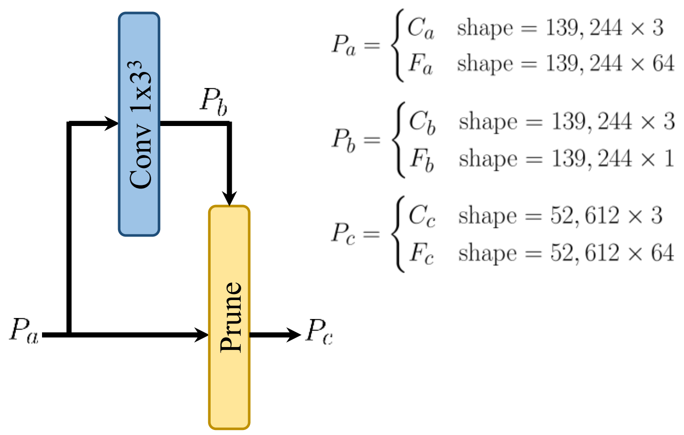

Conv 1x3^3
P_b
Prune
P_a
P_c
$P_a = \begin{cases} C_a & \text{shape} = 139,244 \times 3 \\ F_a & \text{shape} = 139,244 \times 64 \end{cases}$
$P_b = \begin{cases} C_b & \text{shape} = 139,244 \times 3 \\ F_b & \text{shape} = 139,244 \times 1 \end{cases}$
$P_c = \begin{cases} C_c & \text{shape} = 52,612 \times 3 \\ F_c & \text{shape} = 52,612 \times 64 \end{cases}$

Figure 32: Example of pruning layer with input sparse tensor P_a and output sparse tensor P_c. Binary classification is applied to P_b to chose the top voxels and prune false voxels from P_a to obtain P_c.

the multiscale features from the previous frame are passed to the prediction network to obtain current frame's predicted three-times downsampled tensor $\widehat{P^2_{3ds}} = \{C^2_{3ds}, \widehat{F^2_{3ds}}\}$. The predicted downsampled features $\widehat{F^2_{3ds}}$ and the original downsampled features F^2_{3ds} are subtracted to obtain the residual features R^2_{3ds}. The residual is transmitted in a lossy manner using a factorized entropy model [11]. The current frame's three-times downsampled coordinates C^2_{3ds} are transmitted in a lossless manner using an octree encoder like G-PCC [4]. Three-times downsampled coordinates C^2_{3ds} is much smaller than the original geometry (e.g. for the 8iVFB dataset, the C^2_{3ds} is about 16 times smaller than C^2). At the receiver, the previously decoded frame $\widetilde{P^1}$ and the three-times downsampled coordinates C^2_{3ds} are used to predict $\widehat{P^2_{3ds}}$. The residual $\widehat{R^2_{3ds}}$ is added with $\widehat{P^2_{3ds}}$ to obtain the current frame's three-times downsampled tensor representation $\overline{P^2_{3ds}}$. The decoder progressively rescales $\overline{P^2_{3ds}}$ to obtain the current decoded frame $\widetilde{P^2}$.

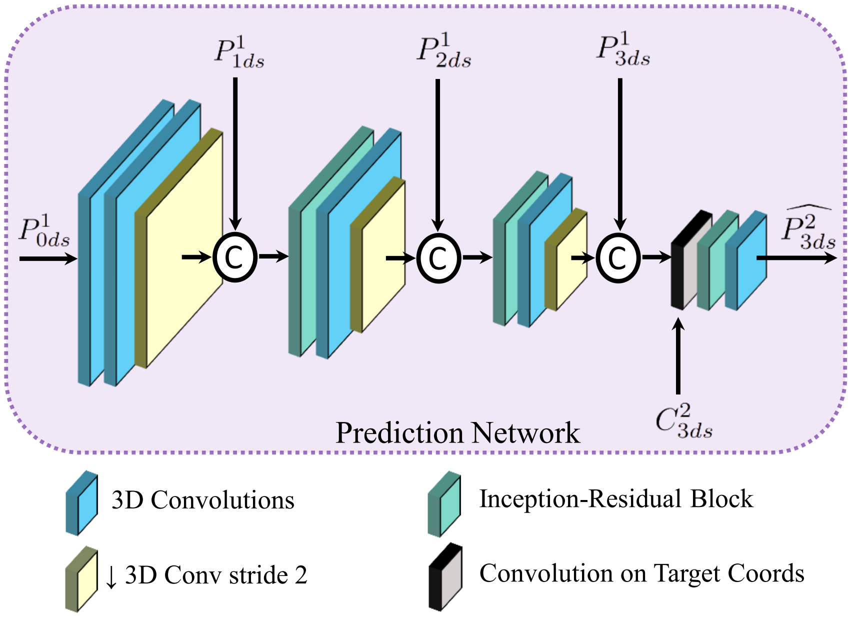


Figure 33: Prediction network. Takes in four multiscale features from previous frame and the three-times downsampled coordinates of the current frame (C^2_{3ds}) to learn current frame's feature embedding $\widehat{P^2_{3ds}}$.

6.3.4 Prediction Network

We propose a novel deep learning-based inter-frame prediction network that can predict the latent representation of the current frame from the previously reconstructed frame as shown in Fig. 33. This way the network performs motion estimation between consecutive frames to measure the current frame's feature embedding. The multiscale features from the previous frame, $P^1_{0ds}, P^1_{1ds}, P^1_{2ds}, P^1_{3ds}$, and the three downsampled coordinates from the current frame, C^2_{3ds}, are fed to the prediction network to obtain current frame's predicted three-times downsampled tensor $\widehat{P^2_{3ds}} = \{C^2_{3ds}, \widehat{F^2_{3ds}}\}$. The prediction network downscales the input three times while concatenating it with the corresponding

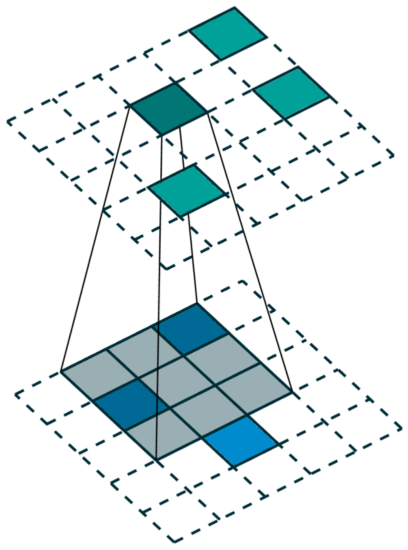


Figure 34: Example of convolution on target coordinates in 2D. Where the blue are the input cooridnates and the green are the output coordinates. (figure taken from [20]).

scale features. Finally we employ a ***convolution on target coordinates*** to obtain features for $\widehat{P^2_{3ds}}$. *Convolution on target coordinates* help us translate the latent features from the downsampled coordinates of P^1, i.e., C^1_{3ds} to the downsampled coordinates of P^2, i.e., C^2_{3ds}. *Convolution on target coordinates* can be viewed as a convolution with arbitrary input and output coordinates where the features from input coordinates get convolved with the convolutional kernel and the output is only retained at the output coordinates. An example of *Convolution on target coordinates* is shown in Fig. 34. During *convolution on target coordinates*, the features are mapped from the input coordinates C^1_{3ds} to the output coordinates C^2_{3ds} after applying sparse convolution. We use a kernel size of $3 \times 3 \times 3$ for the *convolution on target coordinates*.

6.3.5 Training

During training, we optimize the Lagrangian loss, i.e.,

$$J_{loss} = R + \lambda D \tag{6.1}$$

Where R is the compressed bit rate and D is the distortion loss. We employ binary cross-entropy loss for voxel occupancy classification as the distortion loss at the decoder. We employ three binary cross-entropy losses at three different scales such that the total distortion loss is:

$$D = \mathcal{L}_1(\widetilde{P}_{2ds}, P_{2ds}) + \mathcal{L}_2(\widetilde{P}_{1ds}, P_{1ds}) + \mathcal{L}_3(\widetilde{P}, P) \tag{6.2}$$

Where the ground-truth P_{2ds} and P_{1ds} are obtained by voxel or quantization-based down-sampling of the original point cloud P.

The three downsampled coordinates C^2_{3ds} are transmitted losslessly using Octree encoder in G-PCC [4] and consumes a very small amount of bits (i.e., around 0.024 bpp for 8iVFB dataset). We subtract the three downsampled predicted features $\widehat{F^2_{3ds}}$ from the original three downsampled features F^2_{3ds} to obtain the residual features R^2_{3ds}. The residual features are quantized during inference, while during training, uniform noise is used to approximate the quantization [11]. Quantized residual features are encoded by an arithmetic encoder where a fully factorized probabilistic entropy model [12] is used to learn the probability distribution of each feature where the bitrate (R) is lower bounded by its information entropy.

6.4 Experimental Results

6.4.1 Dataset and Training

We used a total of eight different sequences for dynamic point cloud datasets: *longdress, loot, redandblack*, and *soldier* sequences from JPEG Plenos 8i Voxelized Full Bodies dataset (**8iVFB** v2) [23], ***queen*** sequence from Technicolor (https://www.technicolor.com/fr), and *basketball, exercise*, and *model* sequences from MPEGs **Owlii** Dynamic Human Textured Mesh Sequence Dataset [123]. We converted all the dynamic datasets to a voxel depth of 10 bits. The 8iVFB and queen datasets have about 300 frames whereas Owlii dataset has 64 frames.

We train the proposed inter-frame encoding method in an end-to-end manner using dynamic point cloud datasets. During training, *longdress, loot* and *queen* sequences were used; while sequences *redandblack, soldier, basketball, exercise*, and *model* were used during testing. To decrease computational complexity during training, we divide the PC frames into smaller chunks by applying the same kd-tree partition on two consecutive frames. During inference time we used whole point clouds. We use seven different λ values to cover a wide range of bit rates.

6.4.2 Performance Evaluation.

For a fair comparison, we closely follow MPEG's common test conditions (CTC) [96]. We compare our method to the state-of-the-art deep learning intra-frame encoding PCGCv2 [113], MPEG's G-PCC (octree as well as trisoup) [4] methods, as well as

MPEG's video-based V-PCC method (inter and intra-frame encoding) [5]. We utilize G-PCC's latest reference implementation TMC13-v14 and for V-PCC the latest implementation TMC2-v17 is employed that uses HEVC video codec. We employ V-PCC inter-frame low-delay setting which involves P-frame encoding for a fair comparison to our P-frame encoding scheme. We employ MPEG's point-to-point distance (D1) mean squared error (MSE) based Peak Signal-to-Noise Ratio (PSNR) as our evaluation metrics. *Bits per point* (bpp) is used to measure the compression ratio. We plot rate-distortion curves and calculate the BD-Rate (Bjøntegaard Delta Rate) [15] gains over different methods.

Table 15 shows the BD-Rate gains of the proposed method over the state-of-the-art. The lower the BD-Rate value, the more the improvement is. Our method achieves significant gains compared to G-PCC with an average of 91.68% BD-Rate gains against G-PCC (octree), 84.41% BD-Rate improvement over G-PCC (trisoup). Compared to the deep learning-based model PCGCv2, we achieve a 34.08% BD-Rate improvement. Compared to the V-PCC, we achieve a 62.69% BD-Rate improvement over intra-frame encoding mode and 52.44% BD-Rate improvement over inter-frame encoding mode. To the best of our knowledge, our method is the first deep learning-based method to outperform V-PCC inter-frame mode across all rates for dense photo-realistic point clouds.

The rate-distortion curves for each test sequence and their average is plotted in Fig. 36. Fig. 37 shows the zoomed-in version of Fig. 36. As can be seen, our method performs considerably better than G-PCC (octree) and G-PCC (trisoup) and has significant coding gains compared with the deep learning-based model PCGCv2. It should be noted that compared with PCGCv2, our method performs much better at higher PSNR and still

Table 15: BD-Rate gains against the state-of-the-art methods using D1 distortion measurements.

	G-PCC (octree)	G-PCC (trisoup)	PCGCv2 [113]	V-PCC intra	V-PCC inter
basketball	-92.01	-87.80	-32.45	-60.46	-48.82
exercise	-91.70	-87.02	-35.44	-62.08	-48.30
model	-89.86	-83.24	-33.69	-61.93	-51.80
redandblack	-91.42	-81.25	-28.31	-59.33	-55.58
soldier	-92.75	-82.61	-40.16	-66.51	-43.60
Average	-91.68	-84.41	-34.08	-62.69	-52.44

performs better than PCGCv2 at lower PSNRs. This is because both the proposed method and PCGCv2 transmit the three downsampled coordinates in a lossless manner and their corresponding features in a lossy manner. However, at lower PSNRs, most of the bits are consumed by coordinates (i.e., around 0.024 bpp) which constitutes the majority of the bitrate. At higher PSNR values most of the bitrates are due to features. Our inter-frame prediction network transmits only the residual of the features and, hence, can significantly decrease the feature bits transmitted leading to much higher gains at higher PSNR and bitrates.

Compared with V-PCC, we can see that we achieve a much higher PSNR for the same bitrate for all of the sequences and bitrates. As expected, the V-PCC inter-frame encoding mode performs better than V-PCC intra-frame encoding mode. The sequences that

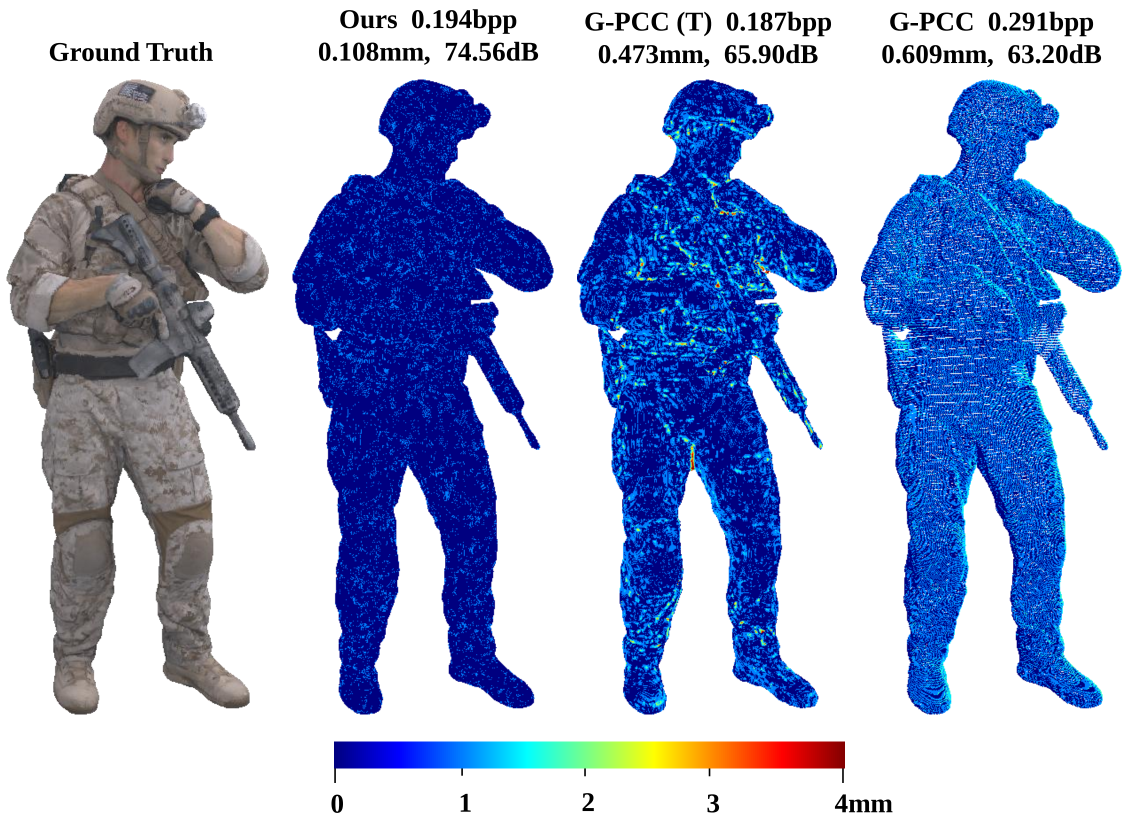


Figure 35: Qualitative visual comparison of sequence "soldier" for different methods. The color error map describes the point-to-point distortion measured in mm, and the numbers above represent the bitrate, mean error measured in mm, and D1 PSNR.

have the most movement (i.e., redandblack) the V-PCC inter and V-PCC intra modes perform pretty similarly whereas the sequence with the least amount of movement (i.e., soldier) the V-PCC inter-frame encoding method performed much better than V-PCC intra-frame encoding method. We can see a similar pattern between our proposed inter-frame method and PCGCv2 which is an intra-frame method. We see that our proposed inter-frame method has the most improvement over PCGCv2 on soldier sequence and the least improvement over PCGCv2 on redandblack sequence. Our Prediction module maps the

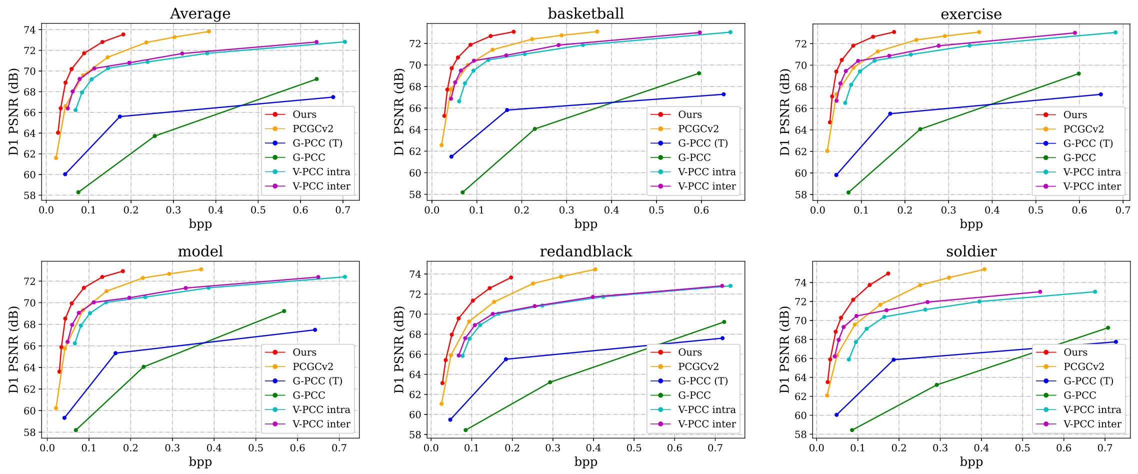


Figure 36: Rate-distortion curves comparison with the state-of-the-arts plotted for five different sequences and their average.

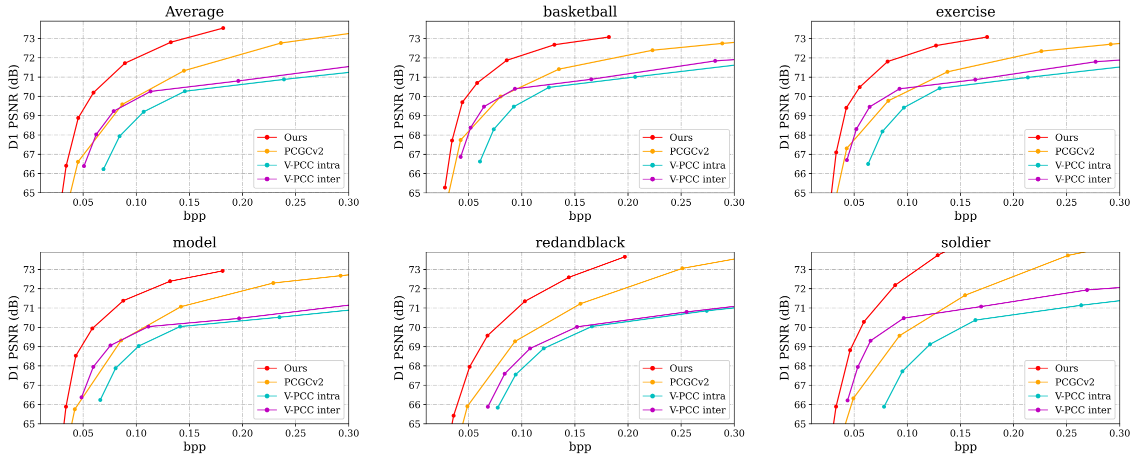


Figure 37: Zoomed-in version of Fig. 36.

features extracted from the previous frame to the coordinates extracted from the current frame. In this way, when the motion between adjacent frames is small, the performance is significantly improved.

Qualitative comparison with G-PCC and our proposed method is presented in Fig.

Table 16: Average runtime of different methods using 8iVFB v2 PCs.

	G-PCC (O)	G-PCC (T)	PCGCv2 [113]	Ours
Enc (s)	1.13	6.15	0.258	0.364
Dec (s)	0.44	5.01	0.537	0.714

35, using point clouds colored by reconstruction error.

6.4.3 Runtime Comparison

We compare the runtime of different methods in Table 16. We use an Intel Core i9-11900F CPU and an Nvidia GeForce GTX 3090 GPU. G-PCC runtime is computed for the highest bitrate on a CPU. While both PCGCv2 and Our method utilize the GPU. Due to the diversity in platforms, e.g., CPU vs. GPU, Python vs. C/C++, etc, the running time comparison only serves as the intuitive reference to have a general idea about the computational complexity. As can be seen, our method experiences a slight increase in runtime due to processing two PC frames at a time. However, the increased complexity is still minimal given that our network is an inter-frame prediction scheme. PCGCv2 has about 778 thousand parameters, whereas, the proposed method has about $2{,}033$ thousand parameters which is still a relatively small network. The runtime complexity can be optimized by migrating to a C++ implementation and simplifying the framework.

6.4.4 Ablation Study: Block Size

Even though in our evaluations, we have used the full point cloud during inference. We wanted to see the effects on PSNR and bitrate of dividing the point cloud into smaller

Table 17: Partitioning the point cloud into smaller number of blocks. Tested on soldier sequence

# of blocks	PSNR	bpp
1	74.56	0.1944
2	74.52	0.1987
4	74.48	0.2055
8	74.35	0.2158

blocks for encoding. The purpose is to demonstrate that if needed, a large point cloud can be partitioned into blocks for processing. During the encoding, we save the *coordinate bitstream, feature bitstream, number of points*, and *the entropy model header information* into four different files. Overall bitrate is decided by the collective size of these files. Once we divide the point cloud into blocks, each block would be encoded separately into four different files so we should expect to see a higher overhead involved. kd-tree partitioning is employed to divide each point cloud into multiple blocks and encoded the blocks independently. The results of this experiment on *soldier* sequence are shown in Table 17. We notice that partitioning the point cloud into smaller blocks decreases the PSNR slightly. However, the difference is minimal. We also notice that the bitrate increases a bit but that could possibly be from the overhead of saving the information in lots of files (e.g. for 8 # of blocks, we have a total of 24 files encoded, whereas, for 1 block, we have a total of 4 files encoded). It is possible to merge these files into a single file to decrease the overhead. However, that is out of the scope of the current work.

6.5 Conclusion

In this work, we propose a deep learning-based inter-frame compression scheme for dynamic point clouds that encodes the current frame using the previously decoded previous frame. We employ an encoder to obtain multi-scale features and a decoder to hierarchically reconstruct the point cloud by progressive scaling. We introduce a novel prediction network module that predicts the latent representation of the current frame by mapping the latent features of the previous frame to the downsampled coordinates of the current frame using *convolution on target coordinates*. We encode and transmit the residual of the predicted features and the actual features. We employ sparse convolutions to reduce the space and time complexity which allows our network to process two consecutive point cloud frames. Experimental results show more than 91% BD-Rate gains over the state-of-the-art MPEG G-PCC (octree), more than 84% BD-Rate gains over G-PCC (trisoup), more than 34% BD-Rate gains over intra-frame network PCGCv2, more than 62% BD-Rate improvement over MPEG V-PCC intra-frame encoding mode, and more than 52% BD-Rate improvement over MPEG V-PCC inter-frame encoding mode.

REFERENCE LIST

[1] Coding of moving pictures and audio. Document ISO/IEC JTC1/SC29/WG11 w18892. [Online].

[2] 6DoF vs 3DoF, 2014. https://packet39.com/blog/2018/02/25/3dof-6dof-roomscale-vr-360-video-and-everything-in-between/ [Online].

[3] Call for Proposals for Point Cloud Compression (V2), 2017. document ISO/IEC JTC1/SC29/WG11 MPEG, N 16763, 3D Graphics, Apr. 2017, [Online].

[4] MPEG-PCC-TMC13: Geometry Based Point Cloud Compression G-PCC, 2021. https://github.com/MPEGGroup/mpeg-pcc-tmc13.

[5] MPEG-PCC-TMC2: Video Based Point Cloud Compression VPCC, 2022. https://github.com/MPEGGroup/mpeg-pcc-tmc2.

[6] Akhtar, A., Gao, W., Li, L., Li, Z., Jia, W., and Liu, S. Video-Based Point Cloud Compression Artifact Removal. *IEEE Transactions on Multimedia 24* (2022), 2866–2876.

[7] Akhtar, A., Gao, W., Zhang, X., Li, L., Li, Z., and Liu, S. Point Cloud Geometry Prediction Across Spatial Scale using Deep Learning. In *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)* (2020), IEEE, pp. 70–73.

[8] Akhtar, A., Kathariya, B., and Li, Z. Low Latency Scalable Point Cloud Communication. In *2019 IEEE International Conference on Image Processing (ICIP)* (2019), IEEE, pp. 2369–2373.

[9] Akhtar, A., Ma, J., Shafin, R., Bai, J., Li, L., Li, Z., and Liu, L. Low Latency Scalable Point Cloud Communication in VANETs using V2I Communication. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)* (2019), IEEE, pp. 1–7.

[10] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. T. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (2003), 3–15.

[11] Ballé, J., Laparra, V., and Simoncelli, E. P. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704* (2016).

[12] Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436* (2018).

[13] Bao, W., Lai, W.-S., Ma, C., Zhang, X., Gao, Z., and Yang, M.-H. Depth-aware video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 3703–3712.

[14] Biswas, S., Liu, J., Wong, K., Wang, S., and Urtasun, R. MuSCLE: Multi Sweep Compression of LiDAR using Deep Entropy Models. *Advances in Neural Information Processing Systems 33* (2020), 22170–22181.

[15] Bjontegaard, G. Calculation of average PSNR differences between RD-curves. *VCEG-M33* (2001). https://cir.nii.ac.jp/crid/1574231874057880192.

[16] Cao, C., Preda, M., Zakharchenko, V., Jang, E. S., and Zaharia, T. Compression of sparse and dense dynamic point clouds - methods and standards. *Proceedings of the IEEE 109*, 9 (2021), 1537–1558.

[17] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012* (2015).

[18] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence 40*, 4 (2017), 834–848.

[19] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 801–818.

[20] Choy, C., Gwak, J., and Savarese, S. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 3075–3084.

[21] Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K. Image Denoising by Sparse 3-D Transform-domain Collaborative Filtering. *IEEE Transactions on Image Processing 16*, 8 (2007), 2080–2095.

[22] De Queiroz, R. L., and Chou, P. A. Motion-compensated compression of dynamic voxelized point clouds. *IEEE Transactions on Image Processing 26*, 8 (2017), 3886–3895.

[23] d'Eon, E., Harrison, B., Myers, T., and Chou, P. A. 8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006* (2017).

[24] Dinesh, C., Cheung, G., and Bajic, I. V. 3D point cloud denoising via bipartite graph approximation and reweighted graph laplacian. *arXiv preprint arXiv:1812.07711* (2018).

[25] Dinesh, C., Cheung, G., and Bajić, I. V. Point Cloud Denoising via Feature Graph Laplacian Regularization. *IEEE Transactions on Image Processing 29* (2020), 4143–4158.

[26] Eldar, Y., Lindenbaum, M., Porat, M., and Zeevi, Y. Y. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing 6*, 9 (1997), 1305–1315.

[27] Fu, C., Li, G., Song, R., Gao, W., and Liu, S. OctAttention: Octree-based Large-scale Contexts Model for Point Cloud Compression. *arXiv preprint arXiv:2202.06028* (2022).

[28] Fu, Z., Hu, W., and Guo, Z. Point cloud inpainting on graphs from non-local self-similarity. In *2018 25th IEEE International Conference on Image Processing (ICIP)* (2018), IEEE, pp. 2137–2141.

[29] Fuchs, H., State, A., and Bazin, J.-C. Immersive 3D Telepresence. *Computer 47*, 7 (2014), 46–52.

[30] Galteri, L., Seidenari, L., Bertini, M., and Del Bimbo, A. Deep generative adversarial compression artifact removal. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 4826–4835.

[31] Gao, L., Fan, T., Wan, J., Xu, Y., Sun, J., and Ma, Z. Point cloud geometry compression via neural graph sampling. In *2021 IEEE International Conference on Image Processing (ICIP)* (2021), IEEE, pp. 3373–3377.

[32] Gao, X., Hu, W., and Guo, Z. Graph-Based Point Cloud Denoising. In *2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM)* (2018), IEEE, pp. 1–6.

[33] Garcia, D. C., Fonseca, T. A., Ferreira, R. U., and de Queiroz, R. L. Geometry Coding for Dynamic Voxelized Point Clouds using Octrees and Multiple Contexts. *IEEE Transactions on Image Processing 29* (2019), 313–322.

[34] Geiger, A., Lenz, P., and Urtasun, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 3354–3361.

[35] Gomes, P. Graph-based Network for Dynamic Point Cloud Prediction. In *Proceedings of the 12th ACM Multimedia Systems Conference* (2021), pp. 393–397.

[36] Graham, B., Engelcke, M., and van der Maaten, L. 3D Semantic Segmentation With Submanifold Sparse Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018), pp. 9224–9232.

[37] Graham, B., and van der Maaten, L. Submanifold Sparse Convolutional Networks. *arXiv preprint arXiv:1706.01307* (2017).

[38] Graziosi, D., Nakagami, O., Kuma, S., Zaghetto, A., Suzuki, T., and Tabatabai, A. An overview of ongoing point cloud compression standardization activities: video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing 9* (2020), e13.

[39] Gu, X., Wang, Y., Wu, C., Lee, Y. J., and Wang, P. HPLFlowNet: Hierarchical Permutohedral Lattice FlowNet for Scene Flow Estimation on Large-scale Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 3254–3263.

[40] Guan, Z., Xing, Q., Xu, M., Yang, R., Liu, T., and Wang, Z. MFQE 2.0: A new approach for multi-frame quality enhancement on compressed video. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[41] Guarda, A. F., Rodrigues, N. M., and Pereira, F. Adaptive deep learning-based point cloud geometry coding. *IEEE Journal of Selected Topics in Signal Processing 15*, 2 (2020), 415–430.

[42] Guennebaud, G., and Gross, M. Algebraic Point Set Surfaces. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, Association for Computing Machinery, p. 23–es.

[43] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021 43*, 12, 4338–4364.

[44] Gwak, J., Choy, C., and Savarese, S. Generative Sparse Detection Networks for 3D Single-shot Object Detection. *arXiv preprint arXiv:2006.12356* (2020).

[45] Haala, N., Peter, M., Kremer, J., and Hunter, G. Mobile LiDAR mapping for 3D point cloud collection in urban areas-A performance test. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci 37* (2008), 1119–1127.

[46] He, K., Zhang, X., Ren, S., and Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *EEE Transactions on Pattern Analysis and Machine Intelligence 37*, 9 (2015), 1904–1916.

[47] Hermosilla, P., Ritschel, T., and Ropinski, T. Total Denoising: Unsupervised learning of 3D point cloud cleaning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 52–60.

[48] Hu, W., Fu, Z., and Guo, Z. Local frequency interpretation and non-local self-similarity on graph for point cloud inpainting. *IEEE Transactions on Image Processing 28*, 8 (2019), 4087–4100.

[49] Huang, H., Li, D., Zhang, H., Ascher, U., and Cohen-Or, D. Consolidation of Unorganized Point Clouds for Surface Reconstruction. *ACM Transactions on Graphics (TOG) 28*, 5 (2009), 1–7.

[50] Huang, H., Wu, S., Gong, M., Cohen-Or, D., Ascher, U., and Zhang, H. Edge-aware point set resampling. *ACM Transactions on Graphics (TOG) 32*, 1 (2013), 1–12.

[51] Huang, L., Wang, S., Wong, K., Liu, J., and Urtasun, R. Octsqueeze: Octree-structured entropy model for LiDAR compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 1313–1323.

[52] Huang, T., and Liu, Y. 3D Point Cloud Geometry Compression on Deep Learning. In *Proceedings of the 27th ACM International Conference on Multimedia* (2019), pp. 890–898.

[53] Huang, T., and Liu, Y. 3D Point Cloud Geometry Compression on Deep Learning. In *Proceedings of the 27th ACM International Conference on Multimedia* (2019), pp. 890–898.

[54] Huang, Z., Yu, Y., Xu, J., Ni, F., and Le, X. PF-Net: Point fractal network for 3D point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 7662–7670.

[55] Javaheri, A., Brites, C., Pereira, F., and Ascenso, J. A generalized Hausdorff distance based quality metric for point cloud geometry. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)* (2020), IEEE, pp. 1–6.

[56] Jia, W., Li, L., Akhtar, A., Li, Z., and Liu, S. Convolutional Neural Network-based Occupancy Map Accuracy Improvement for Video-based Point Cloud Compression. *IEEE Transactions on Multimedia* (2021), 1–1.

[57] Kanezaki, A., Matsushita, Y., and Nishida, Y. RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 5010–5019.

[58] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[59] Krivokuća, M., Chou, P., and Savill, P. 8i voxelized surface light field (8iVSLF) dataset. In *ISO/IEC JTC1/SC29/WG11 MPEG, input document m42914* (2018).

[60] Landrieu, L., and Simonovsky, M. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 4558–4567.

[61] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 12697–12705.

[62] Li, L., Li, Z., Liu, S., and Li, H. Efficient Projected Frame Padding for Video-Based Point Cloud Compression. *IEEE Transactions on Multimedia 23* (2021), 2806–2819.

[63] Li, L., Li, Z., Zakharchenko, V., Chen, J., and Li, H. Advanced 3D Motion Prediction for Video-Based Dynamic Point Cloud Compression. *IEEE Transactions on Image Processing 29* (2020), 289–302.

[64] Li, R., Li, X., Fu, C.-W., Cohen-Or, D., and Heng, P.-A. Pu-gan: a point cloud upsampling adversarial network. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 7203–7212.

[65] Li, R., Li, X., Fu, C.-W., Cohen-Or, D., and Heng, P.-A. PU-GAN: a point cloud upsampling adversarial network. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 7203–7212.

[66] Li, R., Li, X., Heng, P.-A., and Fu, C.-W. Point Cloud Upsampling via Disentangled Refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 344–353.

[67] Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. PointCNN: Convolution On X-Transformed Points. In *Advances in Neural Information Processing Systems* (2018), vol. 31, pp. 820–830.

[68] Lipman, Y., Cohen-Or, D., Levin, D., and Tal-Ezer, H. Parameterization-free projection for geometry reconstruction. *ACM Transactions on Graphics (TOG) 26*, 3 (2007), 22–32.

[69] Liu, D., Li, Y., Lin, J., Li, H., and Wu, F. Deep learning-based video coding: A review and a case study. *ACM Computing Surveys (CSUR) 53*, 1 (2020), 1–35.

[70] Liu, X., Qi, C. R., and Guibas, L. J. FlowNet3D: Learning Scene Flow in 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 529–537.

[71] Lu, F., Chen, G., Qu, S., Li, Z., Liu, Y., and Knoll, A. PointINet: Point Cloud Frame Interpolation Network. *arXiv preprint arXiv:2012.10066* (2020).

[72] Lu, G., Ouyang, W., Xu, D., Zhang, X., Gao, Z., and Sun, M.-T. Deep kalman filtering network for video compression artifact reduction. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 568–584.

[73] Matsuzaki, K., and Tasaka, K. Binary Representation for 3D Point Cloud Compression based on Deep Auto-Encoder. In *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)* (2019), IEEE, pp. 489–490.

[74] Mattei, E., and Castrodad, A. Point cloud denoising via moving rpca. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 123–137.

[75] Mekuria, R., Blom, K., and Cesar, P. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology 27*, 4 (2016), 828–842.

[76] Mittal, H., Okorn, B., and Held, D. Just go with the flow: Self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 11177–11185.

[77] Niklaus, S., and Liu, F. Context-aware synthesis for video frame interpolation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 1701–1710.

[78] Niklaus, S., Mai, L., and Liu, F. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 670–679.

[79] Niklaus, S., Mai, L., and Liu, F. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 261–270.

[80] Orts-Escolano, S., Rhemann, C., Fanello, S., Chang, W., Kowdle, A., Degtyarev, Y., Kim, D., Davidson, P. L., Khamis, S., Dou, M., et al. Holoportation: Virtual 3D teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (2016), pp. 741–754.

[81] Öztireli, A. C., Guennebaud, G., and Gross, M. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 493–501.

[82] Perry, S. JPEG Pleno Point Cloud Coding Common Test Conditions v3. *ISO/IEC JTC1/SC29/WG1 N 86044* (2020).

[83] Preiner, R., Mattausch, O., Arikan, M., Pajarola, R., and Wimmer, M. Continuous projection for fast L1 reconstruction. *ACM Trans. Graph. 33*, 4 (2014), 47–1.

[84] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 652–660.

[85] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems* (2017), pp. 5099–5108.

[86] Qian, G., Abualshour, A., Li, G., Thabet, A., and Ghanem, B. PU-GCN: Point Cloud Upsampling using Graph Convolutional Networks. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021), pp. 11683–11692.

[87] Qian, Y., Hou, J., Kwong, S., and He, Y. PUGeo-Net: A Geometry-centric Network for 3D Point Cloud Upsampling. *arXiv* (2020), arXiv–2002.

[88] Qian, Y., Hou, J., Kwong, S., and He, Y. PUGeo-Net: A Geometry-Centric Network for 3D Point Cloud Upsampling. In *European Conference on Computer Vision* (2020), Springer, pp. 752–769.

[89] Qian, Y., Hou, J., Kwong, S., and He, Y. Deep Magnification-Flexible Upsampling Over 3D Point Clouds. *IEEE Transactions on Image Processing 30* (2021), 8354–8367.

[90] Quach, M., Valenzise, G., and Dufaux, F. Learning convolutional transforms for lossy point cloud geometry compression. In *2019 IEEE International Conference on Image Processing (ICIP)* (2019), IEEE, pp. 4320–4324.

[91] Quach, M., Valenzise, G., and Dufaux, F. Improved deep point cloud geometry compression. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)* (2020), IEEE, pp. 1–6.

[92] Que, Z., Lu, G., and Xu, D. VoxelContext-Net: An Octree based Framework for Point Cloud Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 6042–6051.

[93] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241.

[94] Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M., and Beetz, M. Towards 3D point cloud based object maps for household environments. *Robotics and Autonomous Systems 56*, 11 (2008), 927–941.

[95] Schnabel, R., and Klein, R. Octree-based Point-Cloud Compression. *Proc. IEEE Eurographics Symp. Point-Based Graphics (PBG 06) 6* (2006), 111–120.

[96] Schwarz, S., Martin-Cocher, G., Flynn, D., and Budagavi, M. Common Test Conditions for Point Cloud Compression. *Document ISO/IEC JTC1/SC29/WG11 w17766, Ljubljana, Slovenia, [Online]* (2018).

[97] Schwarz, S., Preda, M., Baroncini, V., Budagavi, M., Cesar, P., Chou, P. A., Cohen, R. A., Krivokuća, M., Lasserre, S., Li, Z., et al. Emerging MPEG standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems 9*, 1 (2018), 133–148.

[98] Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H., and Kautz, J. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2530–2539.

[99] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 945–953.

[100] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Thirty-first AAAI Conference on Artificial Intelligence* (2017).

[101] Tai, Y., Yang, J., Liu, X., and Xu, C. MemNet: A Persistent Memory Network for Image Restoration. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 4539–4547.

[102] Tchapmi, L., Choy, C., Armeni, I., Gwak, J., and Savarese, S. SEGCloud: Semantic Segmentation of 3D Point Clouds. In *2017 International Conference on 3D Vision (3DV)* (2017), IEEE, pp. 537–547.

[103] Tchapmi, L. P., Kosaraju, V., Rezatofighi, H., Reid, I., and Savarese, S. TopNet: Structural Point Cloud Decoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 383–392.

[104] Thanou, D., Chou, P. A., and Frossard, P. Graph-based Motion Estimation and Compensation for Dynamic 3D Point Cloud Compression. In *2015 IEEE International Conference on Image Processing (ICIP)* (2015), IEEE, pp. 3235–3239.

[105] Thanou, D., Chou, P. A., and Frossard, P. Graph-based Compression of Dynamic 3D Point Cloud Sequences. *IEEE Transactions on Image Processing 25*, 4 (2016), 1765–1778.

[106] Tian, D., Ochimizu, H., Feng, C., Cohen, R., and Vetro, A. Geometric distortion metrics for point cloud compression. In *2017 IEEE International Conference on Image Processing (ICIP)* (2017), IEEE, pp. 3460–3464.

[107] Tian, D., Ochimizu, H., Feng, C., Cohen, R., and Vetro, A. Geometric distortion metrics for point cloud compression. In *2017 IEEE International Conference on Image Processing (ICIP)* (2017), IEEE, pp. 3460–3464.

[108] Tu, C., Takeuchi, E., Carballo, A., and Takeda, K. Point cloud compression for 3D LiDAR sensor using recurrent neural network with residual blocks. In *2019 International Conference on Robotics and Automation (ICRA)* (2019), IEEE, pp. 3274–3280.

[109] Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, D. T., and Yeung, S.-K. Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data. In *International Conference on Computer Vision (ICCV)* (2019).

[110] Valsesia, D., Fracastoro, G., and Magli, E. Learning Localized Representations of Point Clouds with Graph-Convolutional Generative Adversarial Networks. *IEEE Transactions on Multimedia* (2020).

[111] Vo, A.-V., Truong-Hong, L., Laefer, D. F., and Bertolotto, M. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing 104* (2015), 88–100.

[112] Wang, J., Ding, D., Li, Z., Feng, X., Cao, C., and Ma, Z. Sparse Tensor-based Multiscale Representation for Point Cloud Geometry Compression. *arXiv preprint arXiv:2111.10633* (2021).

[113] Wang, J., Ding, D., Li, Z., and Ma, Z. Multiscale Point Cloud Geometry Compression. In *2021 Data Compression Conference (DCC)* (2021), pp. 73–82.

[114] Wang, J., Zhu, H., Liu, H., and Ma, Z. Lossy point cloud geometry compression via end-to-end learning. *IEEE Transactions on Circuits and Systems for Video Technology* (2021).

[115] Wang, J., Zhu, H., Ma, Z., Chen, T., Liu, H., and Shen, Q. Learned Point Cloud Geometry Compression. *arXiv preprint arXiv:1909.12037* (2019).

[116] Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y., and Tong, X. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–11.

[117] Wang, X., Ang Jr, M. H., and Lee, G. H. Cascaded refinement network for point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 790–799.

[118] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions On Graphics (TOG) 38*, 5 (2019), 1–12.

[119] Wang, Z., Li, S., Howard-Jenkins, H., Prisacariu, V., and Chen, M. FlowNet3D++: Geometric Losses For Deep Scene Flow Estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2020), pp. 91–98.

[120] Wen, X., Li, T., Han, Z., and Liu, Y.-S. Point cloud completion by skip-attention network with hierarchical folding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 1939–1948.

[121] Wu, W., Wang, Z., Li, Z., Liu, W., and Fuxin, L. PointPWC-Net: A Coarse-to-Fine Network for Supervised and Self-Supervised Scene Flow Estimation on 3D Point Clouds. *arXiv preprint arXiv:1911.12408* (2019).

[122] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3D shapenets: A Deep Representation for Volumetric Shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1912–1920.

[123] Xu, Y., Lu, Y., and Wen, Z. Owlii Dynamic human mesh sequence dataset. *ISO/IEC JTC1/SC29/WG11 m41658* (2017).

[124] Ye, S., Chen, D., Han, S., Wan, Z., and Liao, J. Meta-PU: An Arbitrary-Scale Upsampling Network for Point Cloud. *IEEE Transactions on Visualization and Computer Graphics* (2021).

[125] Yifan, W., Wu, S., Huang, H., Cohen-Or, D., and Sorkine-Hornung, O. Patch-based Progressive 3D Point Set Upsampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 5958–5967.

[126] You, K., and Gao, P. Patch-Based Deep Autoencoder for Point Cloud Geometry Compression. In *ACM Multimedia Asia*. Association for Computing Machinery, 2021, pp. 1–7.

[127] Yu, L., Li, X., Fu, C.-W., Cohen-Or, D., and Heng, P.-A. EC-Net: An Edge-aware Point set Consolidation Network. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 386–402.

[128] Yu, L., Li, X., Fu, C.-W., Cohen-Or, D., and Heng, P.-A. EC-Net: an edge-aware point set consolidation network. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 386–402.

[129] Yu, L., Li, X., Fu, C.-W., Cohen-Or, D., and Heng, P.-A. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2790–2799.

[130] Yu, L., Li, X., Fu, C.-W., Cohen-Or, D., and Heng, P.-A. PU-Net: Point Cloud Upsampling Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2790–2799.

[131] Yu, Y., Huang, Z., Li, F., Zhang, H., and Le, X. Point Encoder GAN: A deep learning model for 3D point cloud inpainting. *Neurocomputing 384* (2020), 192–199.

[132] Zeng, J., Cheung, G., Ng, M., Pang, J., and Yang, C. 3D Point Cloud Denoising using Graph Laplacian Regularization of a Low Dimensional Manifold Model. *IEEE Transactions on Image Processing 29* (2019), 3474–3489.

[133] Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing 26*, 7 (2017), 3142–3155.

[134] Zhang, M., You, H., Kadam, P., Liu, S., and Kuo, C.-C. J. PointHop: An Explainable Machine Learning Method for Point Cloud Classification. *IEEE Transactions on Multimedia* (2020).

[135] Zhang, X., Gao, W., and Liu, S. Implicit Geometry Partition for Point Cloud Compression. In *Data Compression Conference (DCC)* (2020), pp. 73–82.

[136] Zhang, X., Gao, W., and Liu, S. Linear Model Based Geometry Coding for Lidar Acquired Point Clouds. In *Data Compression Conference (DCC)* (2020), pp. 406–406.

[137] Zhou, H., Chen, K., Zhang, W., Fang, H., Zhou, W., and Yu, N. DUP-Net: Denoiser and Upsampler Network for 3D Adversarial Point Clouds Defense. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 1961–1970.

[138] Zhou, Y., and Tuzel, O. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 4490–4499.

VITA

Anique Akhtar was born on December 14, 1990 in Bahawalpur, Pakistan. Anique Akhtar received his B.Sc. degree in Electrical and Electronic Engineering from Lahore University of Management Sciences (LUMS), Lahore, Pakistan, in 2012. He received his M.Sc. degree in Electrical Engineering from Koç University, Istanbul, Turkey, in 2015. Since 2018, he has been pursuing a Ph.D. in Electrical and Computer Engineering at the University of Missouri-Kansas City. He was the recipient of the annual award for outstanding doctoral student in Electrical and Computer Engineering for the academic year 2021-22.

During his study at University of Missouri-Kansas City, Mr. Anique published 5 conference papers, 3 journal papers, and filed 2 patents. Mr. Anique has worked at HERE Technologies during the summer of 2018, and for six months in 2019. He worked at Tencent Media Lab in the summer of 2020. In 2022, he worked at Qualcomm as a Support Engineer.

Mr. Anique's research interest includes deep learning solutions for point cloud processing, data compression, video coding, immersive communication, and machine learning.