4DSR-GCN: 4D Video Point Cloud Upsampling using Graph Convolutional Networks

Lorenzo Berlincioni
lorenzo.berlincioni@unifi.it
MICC, Università degli Studi di Firenze
Italy

Stefano Berretti
stefano.berretti@unifi.it
MICC, Università degli Studi di Firenze
Italy

Marco Bertini
marco.bertini@unifi.it
MICC, Università degli Studi di Firenze
Italy

Alberto Del Bimbo
alberto.delbimbo@unifi.it
MICC, Università degli Studi di Firenze
Italy

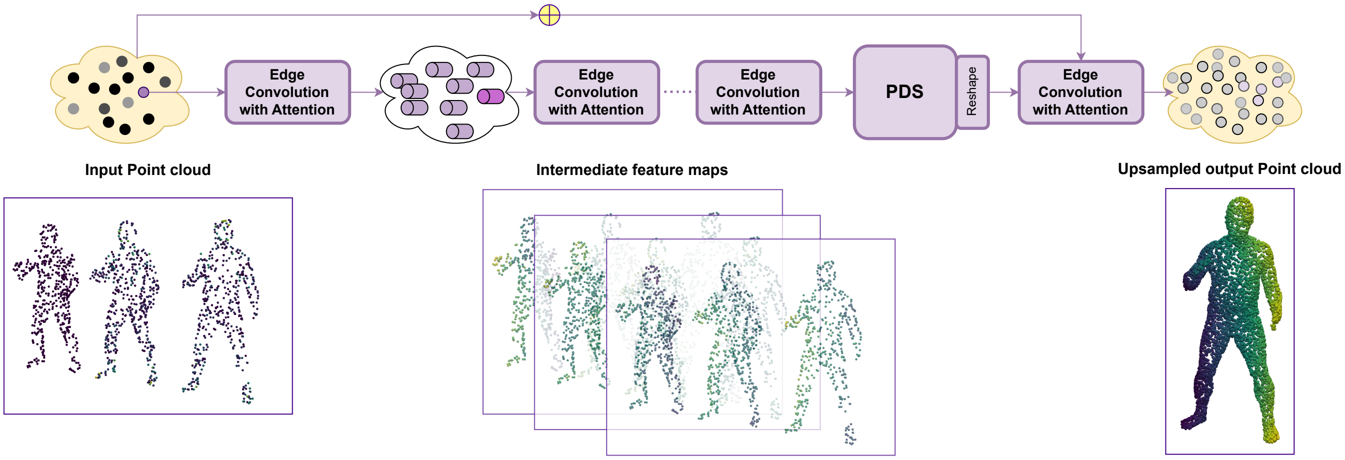
Edge Convolution with Attention
Edge Convolution with Attention
Edge Convolution with Attention
PDS
Reshape
Edge Convolution with Attention
Input Point cloud
Intermediate feature maps
Upsampled output Point cloud

Figure 1: Schematic architecture of the Generator GCN used for our upsampling model: The input point clouds (three point clouds are shown on the left) are first processed by a set of Edge Convolution with Attention layers cascaded to generate an embedding; the Parallel Double Sampling (PDS) layer is then used for upsampling, and the output is summed at the end with the input following a residual-like schema (the upsampled point cloud given as output is shown on the right).

ABSTRACT

Time varying sequences of 3D point clouds, or 4D point clouds, are now being acquired at an increasing pace in several applications (*e.g.*, personal avatar representation, LiDAR in autonomous or assisted driving). In many cases, such volume of data is transmitted, thus requiring that proper compression tools are applied to either reduce the resolution or the bandwidth. In this paper, we propose a new solution for upscaling and restoration of time-varying 3D video point clouds after they have been heavily compressed. Our model consists of a specifically designed Graph Convolutional Network that combines Dynamic Edge Convolution and Graph Attention Networks for feature aggregation in a Generative Adversarial setting. We present a different way to sample dense point clouds with the intent to make these modules work in synergy to provide each node enough features about its neighbourhood in order to later on generate new vertices. Compared to other solutions in the literature that address the same task, our proposed model is capable of obtaining comparable results in terms of quality of the reconstruction, while using a substantially lower number of parameters (≃ 300KB), making our solution deployable in edge computing devices.

McGE '23, October 29, 2023, Ottawa, ON, Canada
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0278-5/23/10.
https://doi.org/10.1145/3607541.3616824

CCS CONCEPTS

• **Computing methodologies** → **Point-based models**; **Neural networks**.

KEYWORDS

Time varying 3D point clouds, 3D upscaling, Graph Attention Network, Generative Adversarial setting, Super Resolution

ACM Reference Format:
Lorenzo Berlincioni, Stefano Berretti, Marco Bertini, and Alberto Del Bimbo. 2023. 4DSR-GCN: 4D Video Point Cloud Upsampling using Graph Convolutional Networks. In *Proceedings of the 1st International Workshop on Multimedia Content Generation and Evaluation: New Methods and Practice*

(McGE ’23), October 29, 2023, Ottawa, ON, Canada. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3607541.3616824

1 INTRODUCTION

In light of emerging applications such as Augmented and Virtual Reality (AR/VR), there is a rising interest in capturing the real world in 3D at high-resolution. For real time applications in dynamic settings, such as 3D sensing for robotics, telepresence, automated driving applications using LiDAR, this technology might need high-resolution point clouds with up to millions of points per frame. After taking into consideration the average point-cloud video, under some constraints such as keeping the identity of a human subject recognizable, we observe that the size of a single instance, which is a single frame, can be approximated as ∼10 Mbytes, which translates to a bitrate of ∼300 Mbytes per second without compression for a 30fps 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 an important area of research. One result in this direction is represented by the Point Cloud Compression standard specifications that include video-based PCC (V-PCC) and geometry-based PCC (G-PCC) [9] as released in 2020 by the The Moving Picture Expert Group (MPEG).

Given these premises, our task is to perform *upscaling* and *artifact removal* of sparsely populated 3D point cloud videos. The terms *upscaling* and *artifact removal* are usually found in image/video super resolution literature and so they might not have an immediate translation in the 3D context. We will use the term *upscale* to indicate the operation by which the total number of vertices of an input point cloud is increased; by using *artifact removal*, instead, we will imply the correct reconstruction process after some sort of compression or subsampling has been performed on an input point cloud. As shown in Figure 2, a high compression rate can achieve acceptable bandwidth requirements with a huge decrease in fidelity. For some applications, for example where the user experience is important, the identity of the subject must be maintained or, for autonomous driving, such a low-resolution is not acceptable.

Recent approaches that tackled this task, such as [39], employed a strategy that uses long sequences of input frames and a large encoder-decoder model. As we will detail below, we followed a different approach.

In this paper, we pose the upscaling problem in a Generative Adversarial setting using two architectural modules: the EdgeConvolution [43] and the Graph Attention Network (GAT) [38]. In particular, the input point clouds are modeled as graphs and processed by a Graph Convolutional Network (GCN). The convolution operation has been performed using a EdgeConv module: this module incorporates local neighborhood information, can be stacked to learn global shape properties, and affinity in feature space captures semantic characteristics over potentially long distances in the original embedding. While this module was used for CNN-based high-level tasks on point clouds, including classification and segmentation, the GAT has been used for feature aggregation performing an attentioned learned mean of the neighbourhood features instead of simply averaging it out. Experiments have been performed on the FAUST 4D dataset [2] also in comparison with state-of-the-art solutions. Overall, our method shown upscaling reconstructions that are comparable with those reported in the literature, while using a lower number of input frames and an architecture with a much lower number of parameters. This opens the way to the deployment of our architecture in edge computing devices.

The main contributions of our work can be summarized as follows:

- We propose a new architecture for time varying point cloud upscaling that combines together a PointNet [32], used as a Discriminator Network in a GAN, and a Generator that makes use of Edge Convolution on the input graphs derived from the point clouds and a graph attention mechanism for aggregating the features of the local neighbourhood. The resulting GAN architecture represents a setting that, to our knowledge, has not been tried before for this task;
- The proposed solution demonstrates a clear advantage over the existing methods in the capability of producing upscaled 3D point clouds with comparable accuracy but using a way lower number of parameters in the architecture. Finally, the inference time is compatible with an online application of the method handling a stream of input frames.

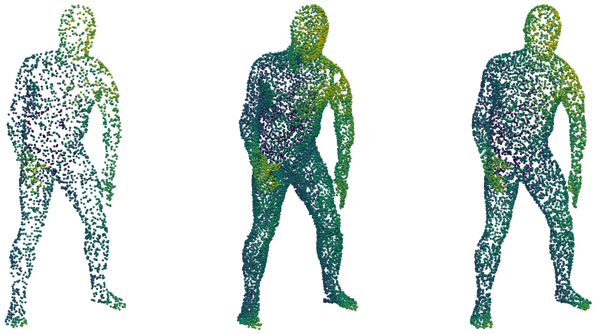

Figure 2: *Left:* Sample of an input low-resolution point cloud with ∼3K vertices. *Center:* our model reconstruction with ∼12K vertices. *Right:* Ground truth point cloud with ∼12K vertices.

2 RELATED WORK

Numerous studies have been conducted with the goal of reconstructing a 3D model given inputs in various possible forms: a mesh, a 3D point cloud, a collection of voxels or an implicit function. Some of these works focused on the use of a 3D point cloud as an input [5, 36]. Others, instead, used a discretized version based on *voxels*, such as [7, 41], or directly tried to reconstruct a mesh [21, 40].

Point cloud upsampling was first approached using optimization based solutions, while deep learning based methods were applied only more recently. Methods from both these categories are summarized below.

Optimization-based methods. One of the first work addressing point sets upsampling was proposed by Alexa *et al.* [1]. In their approach, points at vertices of a Voronoi diagram were interpolated in the local tangent space. Lipman *et al.* [23], presented a Locally Optimal Projection (LOP) operator performing points resampling and surface reconstruction using L1-median. The LOP operator showed satisfactory results even in the case the input point set was affected by noise and outliers. An improved version of the LOP approach aiming to address the density problem of the upscaled point set was then proposed by Huang *et al.* [13]. Overall, good results were demonstrated by these works though their applicability scope was limited by the smoothness assumption of the underlying surface, which is rarely matched by data acquired with real scanners. To overcome such limitation, in [14] Huang *et al.* proposed an edge-aware point set resampling solution that first resamples away from edges, then progressively approaches edges and corners. One limitation of this method is the dependence of the quality of the results from the normals accuracy at the points, and the need for a careful tuning of the parameters. A point representation method based on volumetric voxelization was introduced by Wu *et al.* [45]. As a preliminary operation, they proposed to fuse consolidation and completion in one coherent step. However, the goal of this operation was on filling large holes, so that global smoothness is not enforced, making the method sensitive to large noise. All these methods are not driven by the data, rather they strongly rely on some priors.

Deep-learning based methods. Only recently, methods have adopted deep architectures to directly learn from point sets. This was mainly due to the inherent difficulty of such data, where points are unordered and do not follow any regular-grid structure in their spatial arrangement. To circumvent such difficulty, some methods converted point clouds to other 3D representations, based on graphs [3, 25] or volumetric grids [4, 26, 35, 45]. The PointNet [31] and Point Net++ [32] were the first successful attempts to directly process point clouds for classification and segmentation purposes using a hierarchical feature learning architecture that captures both local and global geometry contexts. Other networks that were proposed for high-level analysis of point clouds focusing on global or mid-level attributes of point clouds include [12, 17, 20, 30, 42]. Local shape properties, like normal and curvature in point clouds, were estimated by the network proposed in [11]. Interesting network architectures were also proposed for 3D reconstruction from 2D images [5, 10, 22]. For example, Fan *et al.* [5] addressed the problem of 3D reconstruction from a single image, generating a straightforward form of output–point cloud coordinates. The 4D extension of the resulting Point Set Generation Network (PSGN-4D) was used in several studies as a baseline for comparison.

One of the first work aiming to perform point cloud upsampling was proposed by Yu *et al.* [48]. They introduced the PU-Net that learns per point features at multiple scales, and expands the set of points using a Multi-layer Perceptron (MLP) with multiple branches. However, to learn multi-layer features the input point sets were downsampled, thus potentially causing a loss of resolution. In [47], the same authors proposed an edge-aware network for point set consolidation (EC-Net) that used a specific loss to encourage learning to consolidate points for edges. On the negative side, a very expensive edge-notation was needed for training the EC-Net. In the work of Yifan *et al.* [46], a progressive network (3PU) was proposed that duplicates the input point patches over multiple steps. The progressive architecture of 3PU makes its training computationally expensive. More data were also required to supervise the middle stage outputs of the network. A Generative Adversarial Network designed to learn upsampled point distributions (PU-GAN) was proposed by Li *et al.* [19], with the main performance improvement obtained by the discriminator. Qian *et al.* [34] proposed to upsample points by learning the first and second fundamental forms of the local geometry. However, their PUGeo-Net needs additional supervision in the form of normals. The PU-GCN proposed by Qian *et al.* [33] performed upsampling by leveraging on an Inception based module to extract multi-scale information, and using a GCN-based upsampling module to capture local point information. This has the main advantage of not needing for additional annotations, like edges, normals, point clouds at intermediate resolutions, *etc.*, while also avoiding the use of a sophisticated discriminator.

Recently, more and more works shifted the attention towards *4D reconstruction*, where a sequence of 3D objects is reconstructed from time-varying point clouds given as inputs [18, 28].

In the Occupancy Network (ONet) proposed by Mescheder *et al.* [27], a 3D object was described using a continuous function that indicates which sub-sets of the 3D space the object occupies, and an iso-surface retrieved by employing the Marching Cube algorithm. Tang *et al.* [37] learned a temporal evolution of the 3D human shape through spatially continuous transformation functions among cross-frame occupancy fields. To this end, they established, in parallel, the dense correspondence between predicted occupancy fields at different time steps via explicitly learning continuous displacement vector fields from spatio-temporal shape representations. Niemeyer *et al.* [29] introduced a learning-based framework for object reconstruction directly from 4D data without predefined templates. The proposed OFlow method calculates the integral of a motion field of 3D points in a 3D point cloud specified in space and time to implicitly represent trajectories of all the points in dense correspondences between occupancy fields. Vu *et al.* [39] proposed a network architecture, called RFNet-4D, that jointly reconstructs objects and their motion flows from 4D point clouds. It is shown that jointly learning spatial and temporal features from a sequence of point clouds can leverage individual tasks, leading to improved overall performance. To this end, a temporal vector field learning module using unsupervised learning approach for flow estimation was designed that, in turn, leveraged by supervised learning of spatial structures for object reconstruction. Jiang *et al.* [15] introduced a compositional representation that disentangles shape, initial state, and motion for a 3D object that deforms over a temporal interval. Each component is represented by a latent code via a trained encoder. A neural Ordinary Differential Equation (ODE) is used to model the motion: it is trained to update the initial state conditioned on the learned motion code, while a decoder takes the shape code and the updated state code to reconstruct the 3D model at each time stamp. An Identity Exchange Training (IET) strategy is also proposed to encourage the network to learn decoupling each component. With respect to the above solutions, our approach is characterized by a specific design that combines two GCNs to work in an adversarial setting. The resulting architecture proved to be flexible in the number of frames used as inputs and conjugated effective reconstructions with inference times that are compatible with online execution.

3 PROPOSED METHOD

3.1 Problem statement

We consider a sequence of point clouds in the 3D space. Each point cloud can be regarded as a frame of a 4D video at time t. In the following, we consider n point cloud frames *fused* together forming a time varying point cloud as an unordered lists of $\{x, y, z, t\}$ points. Our task is to *upscale*, a term borrowed from the 2D image super-resolution domain, each of the point cloud (frame) of the input sequence F_t and get a more detailed one by leveraging the information of the previous $n-1$ low-resolution point cloud frames (*i.e.*, $F_{t-1}, \ldots, F_{t-n+1}$).

More in detail, given a buffer composed of n previous frames, the input point cloud P_i is defined as:

$$P_i = \{p_{-n+1}, p_{-n+2}, ..., p_0\}; P_i \in \mathbb{R}^{4\times L\times n}, \tag{1}$$

where each low-resolution point cloud p_i is composed of L points:

$$p_i \in \mathbb{R}^{4\times L}. \tag{2}$$

We are interested in learning a map $f(P_i, \theta)$ from $P_i \longrightarrow P_T$, where P_T is the target point cloud and it represents the zeroth frame upscaled to have $H = S \times L \times n$ points, with S being the *scale factor*:

$$P_T \in \mathbb{R}^{3\times H}. \tag{3}$$

We note the target frame has just the three spatial components $\{x, y, z\}$. Our proposed method makes use of message passing Graph Networks, different neighbourhood sampling techniques and Generative Adversarial training. More in detail, our architecture has been developed starting from [32]. The employed architecture works on unordered lists of $\{x, y, z, t\}$ points, representing the last n frames *fused* together, using two GCNs in an adversarial setting. The discriminator is based on [32], while the generator improves on the same architecture. In particular, we used different neighbors sampling techniques that were developed with the intent of collecting, for each point, features contemporaneously of its immediate neighborhood and also from furthest vertices of the whole point cloud without making the computation too expensive. The fully convolutional nature of our generator network allows us to potentially train and test at different input and output resolutions.

3.2 Edge Convolution and GAT

The basic module composing our generator network is made of the combination of Edge Convolution [43] and GAT [38]. The Edge Convolution allows us to perform message passing over a dynamic graph in which the edges are updated as the point cloud changes. The GAT side is used to perform an attentional aggregation over the features collected from the dynamic local neighbourhood. This is in contrast with much more common choices for aggregation such as *max* or *average*. We refer to this combination module as *Edge Convolution with Attention.*

3.3 Parallel Double Sampling (PDS) module

The core of the generator side of the architecture is the Parallel Double Sampling (PDS) module that performs two different graph convolutions using two different sets of sampled points. A simplified illustration of this module is presented in Figure 3. For each point, two sets of operations are performed in a parallel fashion. The first set, is a pipeline composed of:

- **Radius filtering**: For each vertex, a filtering step leaves as neighbors, with the capability of passing messages, only those vertices that belong to a sphere of radius r, centered on the vertex;
- **Furthest Point Subsampling**: We used the Furthest Point Subsampling (FPS) algorithm in [32] to sample temporarily, a fraction s of the original points that are the farthest away, inside the radius, from a starting point;
- **Convolution**: Graph convolution is applied over the remaining vertices, **independently of their number**, and their features are aggregated.

The second set of operations, performed in parallel to the first one, is composed of:

- **K-NN**: A fixed number of k closest vertices is selected as neighbors;
- **Convolution**: Graph convolution is applied over the vertices, and their aggregated features.

Finally, the two sets of features are concatenated and fed to a linear layer that maps $2 \times Channels_{in} \longrightarrow Channels_{out}$.

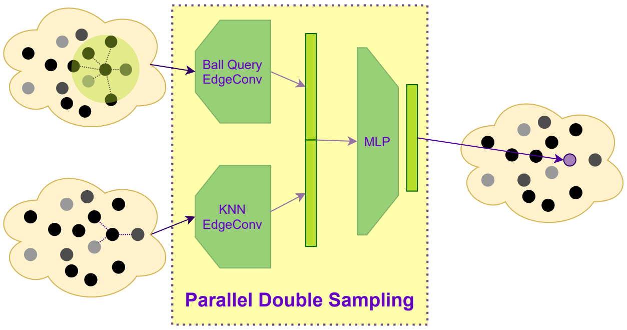

Figure 3: Schematic representation of the proposed Parallel Double Sampling (PDS) module.

3.4 Our Architecture

The proposed architecture is composed of two Graph GCNs working in an adversarial setting [8]. It is illustrated in the bottom of Figure 4. Basically, the point cloud given as input is processed as a graph using message passing based convolution.

3.5 Discriminator

The discriminator is inspired by the PointNet++ architecture [32], since it also targets a classification task. We used the same structure that progressively reduces the number of points using *max-pooling* operations and finally a sequence of linear layers before the output as shown in the bottom part of Figure 4.

3.6 Generator

The generator side of the model is instead built as an initial sequence of Edge Convolution with Attention modules followed by our Parallel Double Sampling (PDS) module. It is also inspired by the PointNet++ architecture [32] but undergoing major changes as detailed in Section 3.3. In the upper part of Figure 4, a simplified visualization of the PDS generator is presented. The generator is composed of multiple Graph Convolutions with Attention followed by a single PDS. The intuition behind this choice is to collect various features for each node, using different neighborhood sampling techniques. Once the original node has been enriched with the local features, the PDS will use them to generate multiple new vertices according to the scale factor. Finally, this new vertices position is summed with the closest one that originated it, in a sort of residual fashion (see Figure 1).

The generator loss L_G is composed of an adversarial component L_{Adv} coming from the Discriminator, a full reference reconstruction loss computed as the Chamfer distance ($L_{C_{dist}}$) between the restored point cloud and the original one, and an additional Density loss L_{Dnt}. We used the LSGAN from [24] loss for our training, which assumes the form:

$$L_{Adv} = \min_G L\left(G\right) = \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[\left(D\left(G\left(\mathbf{z}\right)\right) - c\right)^2\right], \tag{4}$$

for the generator G, and:

$$\min_D L\left(D\right) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\left[\left(D\left(\mathbf{x}\right) - b\right)^2\right] + \tag{5}$$

$$+\frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[\left(D\left(G\left(\mathbf{z}\right)\right) - a\right)^2\right], \tag{6}$$

for the discriminator D.

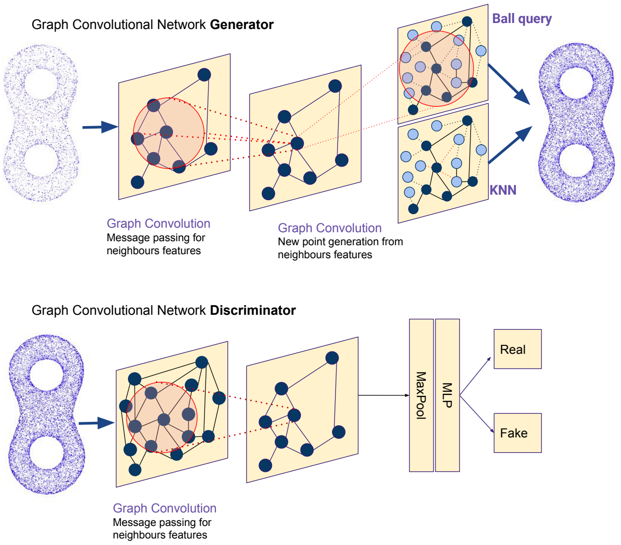
Graph Convolutional Network **Generator**
Ball query
KNN
Graph Convolution
Message passing for neighbours features
Graph Convolution
New point generation from neighbours features
Graph Convolutional Network **Discriminator**
MaxPool
MLP
Real
Fake
Graph Convolution
Message passing for neighbours features

Figure 4: Schematic representation of the proposed GCN architecture. *Top*: Generator architecture; *Bottom*: Discriminator architecture.

3.7 Loss functions

The model is trained end-to-end using multiple losses. Beside the adversarial component L_{Adv}, we also computed the point-to-set Chamfer distance, $L_{C_{dist}}$, between the reconstructed point cloud and the target one and, similarly to [44], we take into account the *neighbourhood* of each point. That is, for each reconstructed point $p_r \in P_r$, we find the closest point $p_t \in P_t$ in the target point cloud, and compute both the distance between them and the difference in terms of local neighbors:

$$L_{C_{dist}}(P_r, P_t) = \sum_{r\in P_r} \min_{t\in P_t} ||r - t||_2^2 + \sum_{t\in P_t} \min_{r\in P_r} ||r - t||_2^2. \tag{7}$$

We define a vertex p *neighbourhood* density $Dnt(p)$ as the normalized sum of its neighbours in a given radius:

$$Dnt(p \in P) = \frac{1}{N_{max}} \sum_{n\in Ball_p} 1, \tag{8}$$

$$L_{Dnt}(P_r, P_t) = \sum_{r\in P_r} \min_{t\in P_t} ||Dnt(r) - Dnt(t)||_2^2 + \tag{9}$$

$$+ \sum_{t\in P_t} \min_{r\in P_r} ||Dnt(r) - Dnt(t)||_2^2. \tag{10}$$

The generator final loss is therefore given by:

$$L_G = \lambda_1 L_{C_{dist}} + \lambda_2 L_{Dnt} + \lambda_3 L_{Adv}, \tag{11}$$

where values for λ_i have been empirically determined ($\lambda_1 = 1.0, \lambda_2 = 0.5, \lambda_3 = 0.1$).

4 EXPERIMENTS

The proposed solution for point clouds upscaling has been evaluated in a comprehensive set of qualitative (Section 4.3) and quantitative (Section 4.4) experiments. An ablation study aiming to evidence the relevance of different components of our architecture is also reported in Section 4.5.

4.1 Implementation details

Our model was implemented using the PyTorch Geometric (PyG) library [6]. This library is specifically designed for Graph Neural Networks. The two networks are implemented as two Message Passing Networks put in an adversarial setting. Both the Discriminator and the Generator are optimized with Adam, using the standard learning rate $lr = 1e^{-4}$ and betas $\beta_1 = 0.9, \beta_2 = 0.999$, using a linear decaying scheduler that drops the learning rate to 1/10th every 10 epochs. Other hyperparameters, such as the radii for the *Ball Query* for the FPS sampling ($r_{small} = 0.06$, $r_{large} = 0.1$) and the number of neighbours for the *KNN* sampling ($n_{neighbours} = 9$) were empirically determined trough grid search.

4.1.1 Augmentation. The training data was augmented using different operations. Each sequence of input point clouds and its relative ground truth point cloud was randomly flipped along any of its axes, per point random noise was added, and finally a random scale along any axes was applied in a range between [0.9, 1.1]. As a form of augmentation, we also exploited the fully convolutional nature of the generator architecture; similar to the case of 2D image super-resolution, where patches of the target high resolution image are used in the training, we randomly feed a 3D slice of the video instead of the full body. As a further form of augmentation during training, a time inversion inside the sequence was also applied.

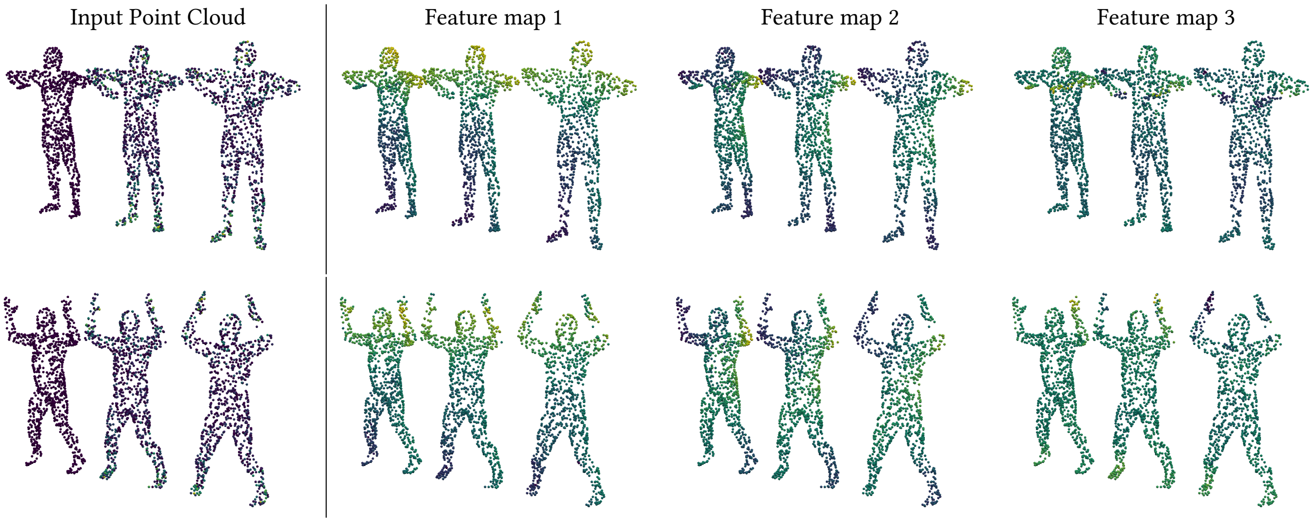

Figure 5: The top and bottom row show: (*left*) the point clouds of a three-frame input sequence with movements. Colors indicate the movement of a point with respect to the previous frame; (*right*) different features obtained from subsequent edge graph convolutional layers of the proposed architecture as a response to the three-frame sequence shown on the left. It can be noted the response of layers seems to shift from spatial (feature map 1) to temporal (feature map 3) details.

4.2 Dataset

To evaluate our proposed solution, we used the Dynamic FAUST (D-FAUST) dataset [2]. It contains animated meshes for 129 sequences of 10 human subjects (5 females and 5 males) with various motions such as "shake hips", "punching", running on "spot", or "one leg jump". In order to compare with other methods, we used the train/test split proposed in [29]. For each sequence, at training time, we randomly pick an index and then subsample the following frames according to the model's frame rate. We trained multiple models at different frame rates. We also followed the evaluation setup used in [29]. Specifically, for each evaluation, we carried out two case studies: *seen individuals but unseen motions* (*i.e.*, test subjects were included in the training data but their motions were not given in the training set); and *unseen individuals but seen motions* (*i.e.*, test subjects were found only in the test data but their motions were seen in the training set).

4.3 Qualitative results

Some qualitative results of the proposed upscaling approach are given in Figures 6 and 7. In Figure 6, the input low-resolution frame, our reconstruction point cloud and the ground truth are given from left to right. A second example is shown in Figure 7, where the input frame, our reconstruction and the ground truth are compared both in terms of point clouds (top) and in terms of mesh reconstruction using the Poisson algorithm (bottom). Additional qualitative results are given as videos in the supplementary material.

To give some insights on the behaviour of the network layers, we inspected the response of the various convolutional layers given an input point cloud, and visualized them. As an example, on the left of Figure 5, three frames of an input point cloud are shown (the frames are taken at three consecutive times, t_0, $t_0 + 1$ and $t_0 + 2$). Points in the clouds are colored to highlight their movement with respect to the previous frame. On the right of Figure 5, instead, the response features of different layers are visualized (the depth of the layers increases from left to right). It is interesting to note as, similarly to CNNs, depth correlates to complexity: The first convolutional layers seem to have strong response for large physical parts of the human subject, while the later ones focus more on time and movement.

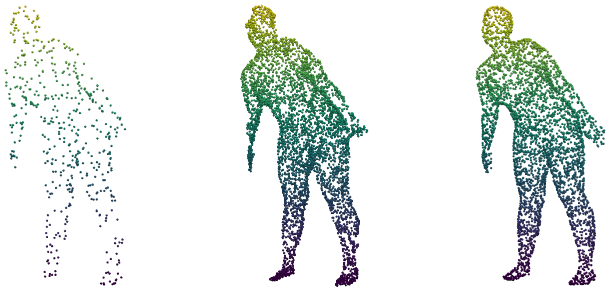

Figure 6: *Left:* Sample of a single frame from an input low resolution point cloud with ~512 vertices, *Center:* reconstruction obtained with our proposed solution; *Right:* Ground truth point cloud.

4.4 Quantitative results

To measure the quality of the reconstructed point set, we applied the standard Chamfer Distance (CD), a point-to-set metric, In doing so, we follow the same protocol as reported in [39], where the CD was used as a reconstruction metric for measuring the dissimilarity between a point and a point set.

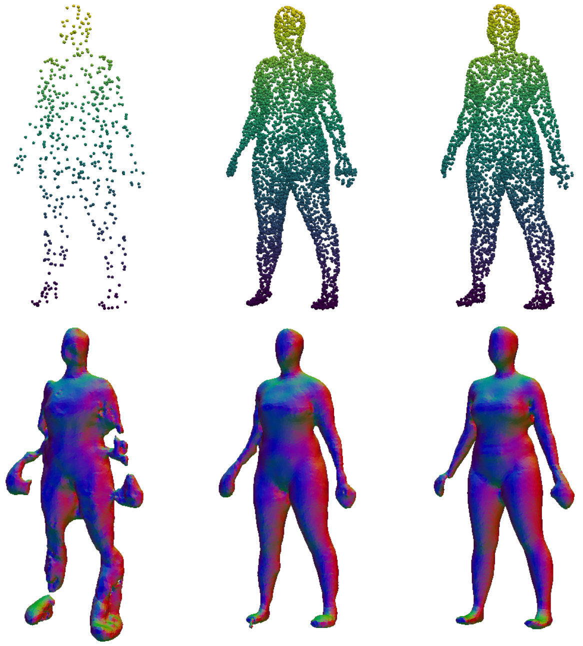

Figure 7: *Top:* Point cloud visualization. *Bottom:* Mesh reconstruction using Poisson surface reconstruction from [16]. *Left:* Sample obtained from a single frame of an input low-resolution point cloud with 1024 vertices; *Center:* Model reconstruction using our proposed approach; *Right:* Ground truth point cloud.

4.4.1 Compared Methods. We compared our approach with respect to six state-of-the-art solutions in the literature for 4D reconstruction from point cloud sequences, namely PSGN 4D, ONet 4D, OFlow, LPDC, 4DCR, and RFNet-4D. The PSGN 4D extends the PSGN approach [5] to predict a 4D point cloud, *i.e.*, the point cloud trajectory instead of a single point set. The ONet 4D network is an extension of ONet [27] to define the occupancy field in the spatio-temporal domain by predicting occupancy values for points sample in space and time. The OFlow network [29] assigns each 4D point an occupancy value and a motion velocity vector and relies on the differential equation to calculate the trajectory. The LPDC [37] learned a temporal evolution of the 3D human shape through spatially continuous transformation functions among cross-frame occupancy fields. The 4DCR solution [15] used a compositional representation that disentangles shape, initial state, and motion for a 3D object that deforms over a temporal interval. Finally, RFNet-4D [39] jointly reconstructs objects and their motion flows from 4D point clouds.

4.4.2 Results. Tables 1 and 2 report results for our solution and for the other methods as given in [39]. For our method (last line in the tables), we used 3 frames for upscaling at 60fps with a scale factor of ×4 starting from low-resolution point clouds composed of 1024 vertices. For the *unseen individual and seen motion* protocol in Table 1, our approach achieved the second best score. From Table 2, it can be observed that our method reached a reconstruction error of similar magnitude with respect to the two best performing methods, *i.e.*, RFNet-4D and LPDC. It is worth noting that RFNet-4D obtained the reported error using a larger number of input frames (*i.e.*, 17 against 3 to 8 as used in our tests). It was not possible to test the RFNet-4D with our setting because the code was not publicly available.

Method	Chamfer Distance $\times 10^{-3} \downarrow$
PSGN-4D [5]	0.6877
ONet-4D [27]	0.7007
OFlow [29]	0.2741
4DCR [15]	0.2220
LPDC [37]	0.2188
RFNet-4D [39]	**0.1594**
Ours	0.1758

Table 1: Reconstruction accuracy for the *unseen individuals and seen motions* protocol. We report the Chamfer distance (lower is better). Results for the best and second best performing methods are given in bold and underlined, respectively. Our approach scored the second best accuracy.

Method	Chamfer Distance $\times 10^{-3} \downarrow$
PSGN-4D [5]	0.6189
ONet-4D [27]	0.5921
OFlow [29]	0.1773
4DCR [15]	0.1667
LPDC [37]	0.1526
RFNet-4D [39]	**0.1504**
Ours	0.1638

Table 2: Reconstruction accuracy for the *seen individuals and unseen motions* protocol. We report the Chamfer distance (lower is better). Results for the best and second best performing methods are given in bold and underlined, respectively. Our approach results in the third best performance.

In Table 3, we report the inference time, in seconds, for various different configurations of our model. All the measurements correspond to experiments executed on an Nvidia 2080Ti GPU. The values reported in the table evidence that our approach can open the way to real-time upscaling. As reported in [39], their method used 17 inout frames to reconstruct an output frame, while our range of frames is between 3 (for models using larger input point clouds) and 8 (for smaller inputs) due to memory constraints at training time.

4.5 Ablation Studies

In this section, we present ablation studies to verify different aspects in the design of our model. In particular, we verified each of the introduced components in our architecture for 4D point clouds reconstruction by comparing the percentage decrease of the model when some particular features are removed. We performed a first set of experiments by using a stream of input point clouds at 60fps and with 256 points per frame; on this stream, we performed upscaling from subsets of consecutive 3 frames, using an

Method	Input size	Upscale ×	Inference time (s) ↓
Ours	1024	3	0.103
Ours	1024	2	0.089
Ours	512	4	0.046
Ours	512	2	0.039
Ours	256	8	0.034
Ours	256	4	0.030
LDPC [37]	-	-	0.44
Oflow [27]	-	-	0.95
RFNet-4D [39]	-	-	0.24

Table 3: Inference time for different configurations of our model using a three-frame buffer. Every test was performed on an Nvidia2080Ti. For the other models it must be noted that they used a 17 frame input sequence to output a frame.

upscale factor of ×2. From Table 4, we can notice that by removing individual components of our architecture, the performance of the model significantly and consistently decreases. In particular, we removed the attention aggregation module and we substituted it with a more common *mean* aggregation. We also ablated the impact of the Density loss and the adversarial component.

Variant	Chamfer $\times 10^{-3}$ ↓	% wrt F. Featured
No Attention	1.193	+3.11%
No Density Loss	1.226	+5.96%
No Adversarial Loss	1.213	+4.84%
Ours Fully Featured	1.157	-

Table 4: Ablation study for our model using 256 input points, 3 frames, 60fps, and upscale factor ×2.

In Table 5, we repeated the above ablation experiments using a different setup. In this case, the frame rate is changed to 30fps, the input resolution to 512 points per frame, and we performed upscaling using a factor of ×4.

Variant	Chamfer $\times 10^{-3}$ ↓	% wrt F. Featured
No Attention	0.5856	+2.82%
No Density Loss	0.6433	+12.95%
No Adversarial Loss	0.5930	+4.13%
Ours Fully Featured	0.5695	-

Table 5: Ablation study for our model using 512 input points, 3 frames, 30fps, and upscale factor ×4.

Also in this case, ablating the density loss term results into the most significant decrease in the accuracy of the upscaled model. It is also interesting to observe that, while the percentage increment in the Chamfer distance when removing the attention layer and the adversarial loss shows small differences between the two tables, this is not the case for the density loss: removing this term has a much larger impact on the results in Table 5 (~ +13%) than in Table 4 (~ +6%).

4.5.1 Importance of the temporal information. A question that arises with the proposed solution is the actual impact of having the time buffer compared to using just the last point cloud as an input. To compare these two solutions, we feed our model with the same frame repeated n times. In this way, we keep the comparison fair by not changing the input size and the amount of starting points but only the *information* contained within it. We refer to this setup as *Static Sequence*, whilst we use the term *Dynamic* to refer the proposed procedure that uses n different frames. In Table 6, we report some comparative results between the two ways of using the frames in a sequence. It can be observed that there is useful information in the time and movement of the cloud. Just like in a 2D video, the same frame repeated n times does not contain the same amount of useful data for reconstruction as n different subsequent frames.

Sequence	Input size	Frms	×	Chamfer x 10^{-3} ↓
Static	256	3	4	2.876
Dynamic	256	3	4	1.109
Static	256	4	3	2.825
Dynamic	256	4	3	0.745
Static	512	3	2	1.851
Dynamic	512	3	2	0.677

Table 6: Ablation study for our model using the aforementioned sequences at different resolutions. It shows how the dynamic approach performs consistently better than the static one.

5 CONCLUSIONS

In this paper, we presented a fully convolutional graph-based approach for video point clouds upscaling using a novel and different approach with respect to most of the state-of-the-art models. Our proposed method is comparable with state-of-the-art solutions in terms of upsampling performance using a lighter architecture allowing the deployment on edge devices with limited computational capabilities. As a possible future development could be the realease as an update for older LiDAR devices or to allow faster 3D point cloud streaming by only transmitting/sampling a subset of the original points. While our method tackles the problem in a different way bringing some advantages, it still has some limitations and drawbacks:

- Training time and memory footprint. Not relying on an encoder-decoder model implies having the whole point cloud at every stage of the network in memory. This slows down the training and limits the number of input frames;
- Results for the reconstruction accuracy are comparable with those reported in the state-of-the-art, though a bit lower.

6 ACKNOWLEDGMENTS

This work was partially supported by the European Commission under European Horizon 2020 Programme, grant number 951911 - AI4Media

REFERENCES

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. 2003. Computing and rendering point set surfaces. *IEEE Trans. on Visualization and Computer Graphics* 9, 1 (2003), 3–15. https://doi.org/10.1109/TVCG.2003.1175093

[2] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2017. Dynamic FAUST: Registering Human Bodies in Motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *Int. Conf. on Learning Representations (ICLR)*.

[4] Angela Dai, Charles Ruizhongtai Qi, and Matthias Niessner. 2017. Shape Completion Using 3D-Encoder-Predictor CNNs and Shape Synthesis. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[5] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. 2017. A Point Set Generation Network for 3D Object Reconstruction From a Single Image. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[6] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[7] R. Girdhar, D.F. Fouhey, M. Rodriguez, and A. Gupta. 2016. Learning a Predictable and Generative Vector Representation for Objects. In *European Conf. on Computer Vision (ECCV)*.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, Vol. 27. https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[9] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai. 2020. 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. https://doi.org/10.1017/ATSIP.2020.12

[10] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 2018. A Papier-Mâché Approach to Learning 3D Surface Generation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[11] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. 2018. PCPNet Learning Local Shape Properties from Raw Point Clouds. *Computer Graphics Forum* 37, 2 (2018), 75–85. https://doi.org/10.1111/cgf.13343

[12] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. 2018. Pointwise Convolutional Neural Networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[13] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. 2009. Consolidation of Unorganized Point Clouds for Surface Reconstruction. *ACM Trans. on Graphics* 28, 5 (dec 2009), 1–7. https://doi.org/10.1145/1618452.1618522

[14] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao (Richard) Zhang. 2013. Edge-Aware Point Set Resampling. *ACM Trans. on Graphics* 32, 1, Article 9 (feb 2013), 12 pages. https://doi.org/10.1145/2421636.2421645

[15] Boyan Jiang, Yinda Zhang, Xingkui Wei, Xiangyang Xue, and Yanwei Fu. 2021. Learning Compositional Representation for 4D Captures With Neural ODE. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. 5340–5350.

[16] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson Surface Reconstruction. In *Eurographics Symposium on Geometry Processing (SGP '06)*. Eurographics Association, Goslar, DEU, 61–70.

[17] Roman Klokov and Victor Lempitsky. 2017. Escape From Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models. In *IEEE Int. Conf. on Computer Vision (ICCV)*.

[18] Vincent Leroy, Jean-Sebastien Franco, and Edmond Boyer. 2017. Multi-View Dynamic Shape Refinement Using Local Temporal Integration. In *IEEE Int. Conf. on Computer Vision (ICCV)*.

[19] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2019. PU-GAN: A Point Cloud Upsampling Adversarial Network. In *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*.

[20] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. PointCNN: Convolution On X-Transformed Points. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 31.

[21] Yiyi Liao, Simon Donné, and Andreas Geiger. 2018. Deep Marching Cubes: Learning Explicit Surface Representations. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2916–2925. https://doi.org/10.1109/CVPR.2018.00308

[22] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. 2018. Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction. In *AAAI Conf. on Artificial Intelligence*, Vol. 32.

[23] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. 2007. Parameterization-Free Projection for Geometry Reconstruction. *ACM Trans. on Graphics* 26, 3 (jul 2007), 22–es. https://doi.org/10.1145/1276377.1276405

[24] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. 2016. Least Squares Generative Adversarial Networks. *2017 IEEE International Conference on Computer Vision (ICCV)* (2016), 2813–2821.

[25] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. 2015. Geodesic Convolutional Neural Networks on Riemannian Manifolds. In *IEEE Int. Conf. on Computer Vision (ICCV) Workshops*.

[26] Daniel Maturana and Sebastian Scherer. 2015. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 922–928. https://doi.org/10.1109/IROS.2015.7353481

[27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3D reconstruction in function space. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. 4460–4470.

[28] Armin Mustafa, Hansung Kim, Jean-Yves Guillemaut, and Adrian Hilton. 2016. Temporally Coherent 4D Reconstruction of Complex Dynamic Scenes. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[29] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. 2019. Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics. In *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*.

[30] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. 2018. Frustum PointNets for 3D Object Detection From RGB-D Data. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[31] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[32] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 30.

[33] Guocheng Qian, Abdulellah Abualshour, Guohao Li, Ali Thabet, and Bernard Ghanem. 2021. PU-GCN: Point Cloud Upsampling Using Graph Convolutional Networks. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. 11683–11692.

[34] Yue Qian, Junhui Hou, Sam Kwong, and Ying He. 2020. PUGeo-Net: A Geometry-Centric Network for 3D Point Cloud Upsampling. In *European Conf. on Computer Vision (ECCV)*. 752–769.

[35] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. 2017. OctNet: Learning Deep 3D Representations at High Resolutions. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[36] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas Guibas. 2017. Frustum PointNets for 3D Object Detection from RGB-D Data. (11 2017).

[37] Jiapeng Tang, Dan Xu, Kui Jia, and Lei Zhang. 2021. Learning Parallel Dense Correspondence from Spatio-Temporal Descriptors for Efficient and Robust 4D Reconstruction. *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)* (2021), 6018–6027.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[39] Tuan-Anh Vu, Duc Thanh Nguyen, Binh-Son Hua, Quang-Hieu Pham, and Sai-Kit Yeung. 2022. RFNet-4D: Joint Object Reconstruction and Flow Estimation from 4D Point Clouds. In *European Conf. on Computer Vision (ECCV)*. 36–52.

[40] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018. Pixel2mesh: Generating 3d mesh models from single rgb images. In *European Conf. on Computer Vision (ECCV)*. 52–67.

[41] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN: Octree-Based Convolutional Neural Networks for 3D Shape Analysis. *ACM Trans. on Graphics* 36, 4, Article 72 (jul 2017), 11 pages. https://doi.org/10.1145/3072959.3073608

[42] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. 2018. SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[43] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)* 38, 5 (2019), 1–12.

[44] Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. 2021. Density-aware chamfer distance as a comprehensive metric for point cloud completion. *arXiv preprint arXiv:2111.12702* (2021).

[45] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[46] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. 2019. Patch-Based Progressive 3D Point Set Upsampling. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*.

[47] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. EC-Net: an Edge-aware Point set Consolidation Network. In *European Conf. on Computer Vision (ECCV)*.

[48] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. PU-Net: Point Cloud Upsampling Network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.