Point Cloud Generation Using Deep Adversarial Local Features for Augmented and Mixed Reality Contents

Sohee Lim, *Student Member, IEEE*, Minwoo Shin, *Student Member, IEEE*, and Joonki Paik, *Senior Member, IEEE*

***Abstract*—We present a generative model-based point cloud generation method using deep adversarial local features. The proposed generative adversarial network (GAN) can reduce computational load and increase the accuracy in three-dimensional (3D) acquisition, reconstruction, and rendering processes. To train the proposed GAN, we first optimize the latent space using an autoencoder to extract local features. The training process provides an accurate estimation of local context from the latent variables and robust point cloud generation. The main contribution of this work is a novel deep learning-based 3D point cloud generation, which significantly reduces computational load to render augmented reality (AR) and mixed reality (MR) contents. Additional contribution in the deep learning field is twofold: i) The autoencoder in the proposed network avoids the vanishing gradient problem using hierarchically linked features in different layers, and ii) the complexity of the network is significantly reduced by removing the transformation network that estimates the affine transformation matrix of the point cloud.**

***Index Terms*—Augmented reality (AR), mixed reality (MR), point cloud, generative adversarial network (GAN).**

I. Introduction

RECENT advances in three-dimensional (3D) imaging technology makes various applications possible such as 3D movies, augmented reality (AR), virtual reality (VR), mixed reality (MR), and interactive games. Along with this trend, point cloud-based 3D representation attracts growing attention. A point cloud is a set of points in a 3D space to describe or represent the surface of an object, and can be directly obtained by a 3D scanner without additional conversion processes.

In particular, a point cloud can be obtained as raw data and can provide spatial information with higher resolution than other methods. Since 3D scanners become popular, and computing power rapidly increases, the point cloud is used in various fields including: AR, VR, robotics, and autonomous driving. AR and MR particularly require an interaction between virtual and real objects, and the demand for 3D models of various virtual objects is increasing for deeper immersion and realism.

For several decades, 3D imaging technologies have matured under the umbrella of high-end consumer markets and been applied with great success, mainly in medical and industrial applications. Thanks to advances in highly integrated semiconductor devices, new 3D technologies are now able to meet the needs of consumer market [1]. For example, mobile devices such as Sony Xperia XZ1 and Apple iPhone-X are equipped with 3D scanning cameras [2], [3], and compression technology is being developed to handle a large amount of point cloud data to support 3D applications for an immersive VR experience for consumers [4].

Typical consumer electronics products using 3D imaging and sensing technologies include: game accessories with a leap motion gesture controller, software-based AR applications in mobile devices, drones and robots. In particular, since interactions between a virtual and real objects are important, the demand for 3D models of various virtual objects is increasing for deeper immersion and realism. Implementation of 3D models in computer graphics requires huge processing power and time. To solve that problem, deep learning-based point cloud generation and reconstruction become an active research field in the sense of both accuracy and computational efficiency.

To overcome the limitation of 3D reconstruction using multiple 2D images, Cho *et al.* used a time-of-flight (TOF) depth camera to generate a virtual human actor [5]. In this context, Kim *et al.* generated 3D video using a TOF depth sensor, and Tsai *et al.* proposed a real-time 2D-to-3D video conversion system respectively in [6], [7].

On the other hand, rendering a full 3D model using computer graphics requires tremendous processing power and time. To solve these problems, deep learning-based point cloud generation and reconstruction approaches are actively being conducted. Early deep learning models cannot deal with permutation-invariant and transformation-invariant features in the point cloud, which is an unordered set of 3D points. To overcome the limitations, Herrera *et al.* proposed a machine learning approach to 2D-to-3D video

Manuscript received August 21, 2020; revised January 15, 2021, March 30, 2021, July 26, 2021, and October 10, 2021; accepted January 4, 2022. Date of publication January 7, 2022; date of current version April 8, 2022. This work was supported in part by the Institute of Civil-Military Technology Cooperation Program funded by the Defense Acquisition Program Administration and Ministry of Trade, Industry and Energy of Korean Government under Grant 19CM5119; in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by Korea Government (MSIT) under Grant 2021-0-01341; and in part by the National Research and Development Program through the National Research Foundation of Korea (NRF) funded by Ministry of Science and ICT under Grant 2020M3F6A1110350. *(Corresponding author: Joonki Paik.)*

The authors are with the Department of Image, Chung-Ang University, Seoul 06974, South Korea (e-mail: sohee@cau.ac.kr; minwoo03d0@cau.ac.kr; paikj@cau.ac.kr).

Digital Object Identifier 10.1109/TCE.2022.3141093

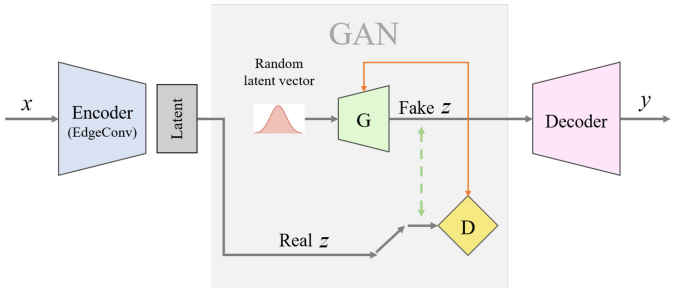

Fig. 1. The framework of LatentGAN-based point cloud generation model [18]. The LatentGAN first trains the AE to learn a latent vector, and then trains the generative model in the trained latent space, which is easier than to train a vanilla GAN. This framework provides superior reconstruction results and better coverage of the data distribution.

conversion [8], and Qi *et al.* proposed a deep learning-based 3D point cloud classification and segmentation methods [9], [10]. Furthermore, related studies are actively being conducted [11]–[15].

In particular, graph-based networks using geometric information have been very successful in classification and segmentation of 3D point clouds [16], [17].

Deep learning frameworks that perform data generation typically include variational autoencoders (VAEs) or generative adversarial networks (GANs) [19], [20]. The proposed model can generate realistic point clouds with higher resolution for real-world applications. The major reason for using a GAN is the higher resolution than a VAEs by using a clearly guessed density function even if it does not exactly match the density of the actual data [21]. In other words, the point cloud generation task using GANs are advantageous for the synthesis and analysis of 3D data to construct AR/MR contents and 3D information by providing a clear probability distribution of the point cloud. Unfortunately, the training process of GAN is very unstable [22]. To solve that problem, Achlioptas *et al.* introduced LatentGAN, which trains GAN by building a latent space that is stably learned by an autoencoder as shown in Fig. 1 [18]. Achlioptas' method uses the PointNet-based autoencoder, which generates undesired artifacts because of losing local features [9].

To avoid that problem, we adopted a linked dynamic graph CNN (LDGCNN) in the encoder to capture the local features as shown in Fig. 2 [16]. The LDGCNN uses an edge convolutional (EdgeConv) layer, which was originally proposed by Wang *et al.* and links hierarchical features from different layers [11], [16]. Major advantage of using EdgeConv is local neighborhood information aggregation and its combination with global shape information. In addition, it is possible to reduce artifacts in the generated point cloud by capturing semantically similar structures in the latent space despite being far from the original input space.

The remainder of the paper is organized as follows: Section II summarizes related works. Section III presents the proposed LatentGAN using local features. Section IV shows experimental results, and Section V concludes the paper.

II. Related Work

This section briefly summarizes general deep learning-based approaches for point cloud classification and segmentation and generative models for point cloud generation.

A. Deep Learning for Point Cloud Classification and Segmentation

Qi *et al.* proposed PointNet for non-ordered 3D point cloud classification and segmentation [9]. Point clouds have two properties: permutation-invariance and transformation-invariance. The PointNet directly processes points in the point cloud in a permutation-invariant manner, applies multi-layer perceptron (MLP) to independently extract a feature for each point, and then uses the max pooling layer to capture the global features. To address transformation-invariance, PointNet spatially aligns the input point cloud and features by adding a transformation network that constructs the affine transformation matrix.

Although the original PointNet works well for classification and segmentation of point cloud, it does not consider the relationship between points and their neighbors. Therefore, it is difficult to hold the local structure of the data by processing each point independently, which leads to incorrect segmentation due to the loss of local features.

DGCNN uses an edge convolutional (EdgeConv) layer to extract local geometric features from the relationship between a point in the point cloud and its neighboring points by applying K-nearest neighbors (KNN) [11]. The EdgeConv layer dynamically updates the graph for each layer. This convolution operation not only finds neighbors in the Euclidean space but also clusters similar shapes in the feature space. In other words, semantic context can be understood and incorporated.

Both PointNet and DGCNN include a transformation module to estimate the affine transformation matrix of the point cloud. Since this module doubles the network size, the vanishing gradient problem is unavoidable. To solve this problem, LDGCNN links hierarchical features from dynamic graphs and replaces the transformation network with MLP [16]. This method successfully avoids the vanishing gradient problem by reducing the size of the network.

B. Generative Model for Point Cloud Generation

Recently, generative models were used to generate point cloud. Achlioptas *et al.* proposed LatentGAN, which is the first work using GAN to generate point cloud. The LatentGAN trains GAN with a compact latent representation through a pre-trained autoencoder [18]. Zamorski *et al.* extended the variational autoencoder (VAE) to adversarial autoencoder (AAE) to generate a point cloud [23]. As mentioned earlier, these two methods use PointNet-based encoders, which cannot capture the local features to understand the shape of the data. Sun *et al.* proposed pointGrow, which is an auto-regressive model that generates a point cloud by modeling joint 3D spatial distribution in a point-by-point manner [24]. PointGrow is sensitive to missing parts and rotations since it generates points in a prespecified order. Therefore, it becomes inaccurate when point cloud is deformed without a proper restoration

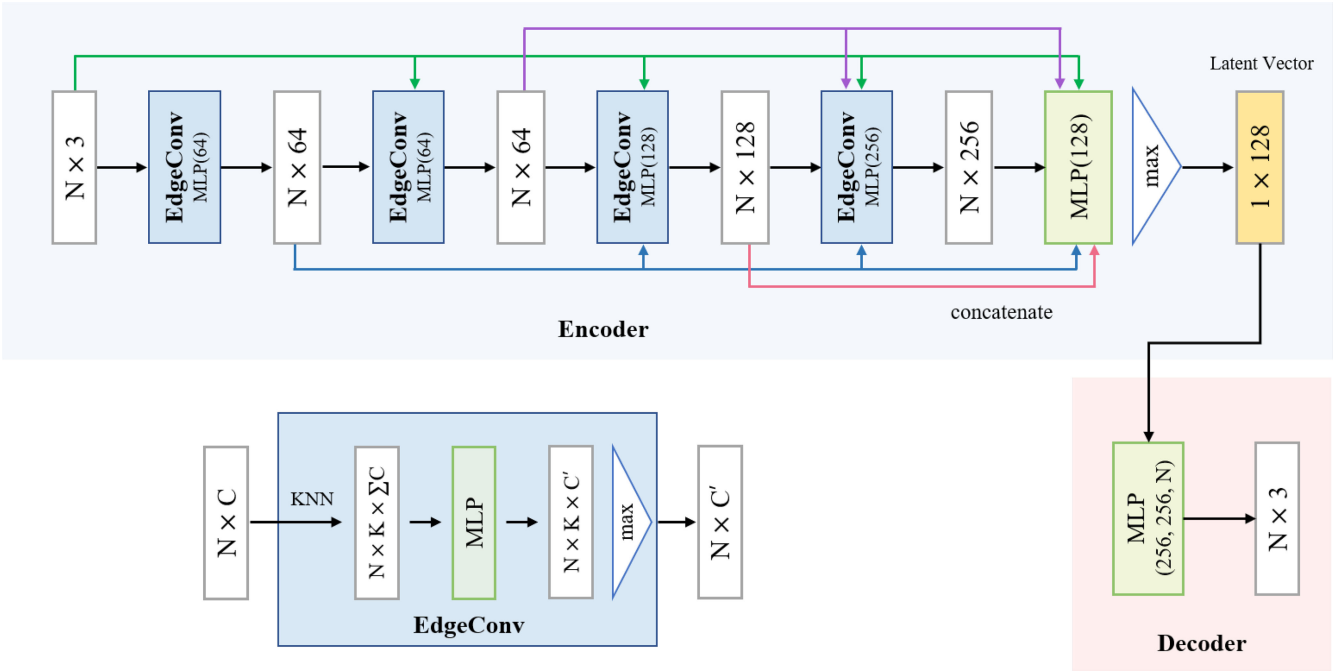

Fig. 2. The architecture of the proposed autoencoder. Both input and output of the autoencoder have N points. The EdgeConv layer extracts local edge features from the aggregated features and their neighbors, which consist of features from the input to current layers [16]. Multi-layer perceptron (MLP) shares parameters for all points and represents a symmetric function. Shortcut connections are used aggregated features to extract multi-scale features, and max polling is used to get a global feature.

or compensation. Yang *et al.* proposed PointFlow using VAE with continuously normalizing flows [25]. PointFlow is a flow-based method, and its distribution estimation performance is relatively inferior to generative models such as GAN and VAE. For that reason, it cannot be used for 3D rendering applications for AR/MR because it cannot control the shape, and, as a result, generates only one point at a time.

Our method is inspired by DGCNN, LDGCNN and LatentGAN [11], [16], [18]. Existing LatentGAN can neither learn local features nor share weights by applying PointNet to encode latent variables. Furthermore, it is difficult to generate a realistic shape since a data shape is ignored, which makes learning difficult. To solve this problem, we propose a model that generates a point cloud by replacing the autoencoder with LDGCNN as proposed in [16]. The LDGCNN-based autoencoder builds the optimized latent space using EdgeConv which dynamically extracts local geometric structures.

III. Proposed Method

Based on the original LatentGAN [18], the proposed 3D point cloud generation method considers the local structure by replacing the autoencoder with LDGCNN [16]. LatentGAN consists of an autoencoder and GAN modules. The former is trained to build a stable latent space by providing a point cloud. A new 3D model is then generated by learning the GAN in the latent space obtained through the encoder. Before describing the details of the proposed method, we will briefly introduce EdgeConv.

Terminologies and notations used in this paper are summarized in Table I.

TABLE I
Notations Used in This Paper

Notations	Descriptions
$\mathbb{R}^d$	d-dimensional Euclidean space
a	Scalar
$\mathbf{a}$	Vector
$\mathbf{A}$	Matrix or Point cloud
$\mathcal{G}$	Graph
$\mathcal{V}$	The set of nodes in a graph
$\mathcal{E}$	The set of edges in a graph
$\lvert\cdot\rvert$	Ordinal number
$\lVert\cdot\rVert_2$	L2-norm, Euclidean distance
$x \sim p_x$	x follows probability distribution p_x
$\mathbb{E}_x[f(x)]$	Expected value of $f(x)$ in a distribution with x as a variable
$KL(\cdot\|\|\cdot)$	KL-divergence between the two distributions

A. EdgeConv

The proposed EdgeConv is a convolution operation inspired by PointNet and graph convolution network (GCN) to extract local geometric relationships between a point and its neighbors in the point cloud. EdgeConv generates a local neighborhood graph by considering K neighboring points in the point cloud in the same manner that a convolutional neural network (CNN) considers adjacent pixels in a uniform, Euclidean grid, such as an image. The EdgeConv goes through each layer of the network, and the graph is dynamically updated to learn global shape properties.

A set of N 3D points in $\mathbb{R}^3$ is denotes as:

$$\mathbf{P} = \left\{\mathbf{p}_i = (x_i, y_i, z_i) \,|\, i = 1, \ldots, N\right\} \subset \mathbb{R}^3. \tag{1}$$

The graph $\mathcal{G}$ consists of a vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. More specifically, $\mathcal{V}$ represents a set of points in the point cloud as a vertex of the graph, and $\mathcal{E}$ is a set of edges connecting the

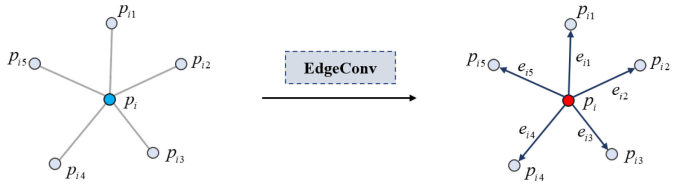

Fig. 3. EdgeConv operation. The $\mathbf{p}_i$ and $\{\mathbf{p}_{i1}, \ldots, \mathbf{p}_{i5}\}$ respectively represent the center point and its neighbors. The directed edges from neighbors to center are denoted by $\{\mathbf{e}_{i1}, \ldots, \mathbf{e}_{i5}\}$. The output of EdgeConv is calculated by aggregating the edge features.

corresponding pair of vertices. To express the local structure of the point cloud $\mathbf{P}$, KNN is applied. For the MLP operation, the edges $\mathbf{e}_{ij}$ are calculated by the distance between the center point $\mathbf{p}_i$ and its k neighboring points $\mathbf{p}_{ij}$ to construct a directed graph $\mathcal{G}$. The graph layer and its related variables are represented as:

$$\begin{aligned} &\mathcal{G} = (\mathcal{V}, \mathcal{E}), \\ &\mathcal{V} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\} \subset \mathbb{R}^C, \\ &\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_N\} \subset \mathcal{V} \times \mathcal{V}, \text{ where } \mathbf{e}_i = (\mathbf{e}_{i1}, \ldots, \mathbf{e}_{ik}), \\ &\mathbf{e}_{ij} = \mathbf{p}_{ij} - \mathbf{p}_i, \end{aligned} \tag{2}$$

where C represents the dimension of input points of the layer. Although each point in the point cloud is a 3D vector, its dimension changes through each layer.

The advantage of applying KNN is that we can construct a local graph in both Euclidean and feature spaces. After constructing the local graph, local features can be extracted through the EdgeConv layer. The input of EdgeConv is a local graph of a center point $\mathbf{p}_i$ and the output is a local feature $\mathbf{l}_i$:

$$\begin{aligned} \mathbf{l}_i &= \max_{j\,:\,(i,j)\in\mathcal{E}} h\big(\mathbf{p}_i, \mathbf{e}_{ij}\big) \\ &= \max\big\{h(\mathbf{p}_i, \mathbf{e}_{i1}),\ \ldots,\ h(\mathbf{p}_i, \mathbf{e}_{iK})\big\}, \end{aligned} \tag{3}$$

where $h : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^{C'}$ is a feature extraction function in the form of an MLP, and C' represents the number of channels for output hidden feature vector, $h(\mathbf{p}_i, \mathbf{e}_{ij})$. The hidden feature vector has the center point $\mathbf{p}_i$ and an edge vector $\mathbf{e}_{ij}$, which combines the global shape structure captured at the coordinates of the center point $\mathbf{p}_i$ and local neighborhood information captured in the form of edges $\mathbf{e}_{ij}$. Each vertex set represents a center point. Edge set is a subset of $\mathcal{V} \times \mathcal{V}$, whose elements represent the relationships between the center point and its K neighbors. The max-pooling operation is a symmetric function that can extract the most principal feature among all edge, regardless of the order of neighbors.

B. Autoencoder

The autoencoder is based on the framework of LDGCNN that is optimized for DGCNN [11], [16]. LDGCNN removes the transformation network in DGCNN, and extracts transformation invariant features using MLP. In addition, hierarchical features of different dynamic graphs are combined to calculate useful edge vectors and to avoid the vanishing gradient problem.

The autoencoder loss function compresses data into a lower-dimensional latent space $\mathbf{Z} \subseteq \mathbb{R}^d$. Therefore, we train the encoder $E : \mathbf{P} \to \mathbf{Z}$ and decoder $D : \mathbf{Z} \to \mathbf{P}$ to minimize reconstruction errors between original $\mathbf{p}_i$ and its reconstruction $\hat{\mathbf{p}}_i$.

The loss for reconstruction of the autoencoder uses Chamfer distance (CD) and Earth Mover's distance (EMD). These two metrics can compare unordered point sets. EMD transforms a distribution to another one. Given two subsets of the same size $\mathbf{P}, \mathbf{Q} \subseteq \mathbb{R}^3$, their EMD is defined as:

$$EMD(\mathbf{P}, \mathbf{Q}) = \min_{\phi:\mathbf{P}\to\mathbf{Q}} \sum_{\mathbf{p}\in\mathbf{P}} \|\mathbf{p} - \phi(\mathbf{p})\|_2, \tag{4}$$

where $\phi(\mathbf{p})$ is a bijection, which maps a point $\mathbf{p} \in \mathbf{P}$ to its closest point $\mathbf{q} \in \mathbf{Q}$. Since estimation of EMD using a deep neural network is computationally expensive, we use a CD as the loss of the autoencoder. The CD measures the squared distance between a point in one set and its nearest neighbor in the other set:

$$CD(\mathbf{P}, \mathbf{Q}) = \sum_{\mathbf{p}\in\mathbf{P}} \min_{\mathbf{q}\in\mathbf{Q}} \|\mathbf{p} - \mathbf{q}\|_2^2 + \sum_{\mathbf{q}\in\mathbf{Q}} \min_{\mathbf{p}\in\mathbf{P}} \|\mathbf{p} - \mathbf{q}\|_2^2. \tag{5}$$

C. LatentGAN

GANs generate realistic data by competing with generator G, which generates data with the same distribution as real data, and discriminator D, which distinguishes real data from generated data. This is a minmax problem of G and D [20]:

$$\begin{aligned} \min_G \max_D \mathrm{V}(D, G) = \mathbb{E}_{\mathbf{x}\sim p_r}\big[\log D(\mathbf{x})\big] \\ + \mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}\big[\log\big(1 - D\big(\tilde{\mathbf{x}}\big)\big)\big], \end{aligned} \tag{6}$$

where p_r is a distribution of real data x, and p_g is model distribution implicitly defined by $\tilde{\mathbf{x}} = G(\mathbf{z})$, $\mathbf{z} \sim p(\mathbf{z})$. The input to the generator, $\mathbf{z}$, is sampled from noise distribution p, such as Gaussian distribution. D learns that $D(\mathbf{x}) = 1$ for real data $\mathbf{x}$ and $D(G(\mathbf{z})) = 0$ for generated data $G(\mathbf{z})$. In other words, D learns to distinguish between real and generated data while G generates data that is similar to the real data, and learns that $D(G(\mathbf{z})) = 1$.

LatentGAN acts on the bottleneck code of a pretrained autoencoder. After learning the GAN, the generated code is converted to a point cloud using the decoder of the autoencoder. Achlioptas *et al.* showed that learning a latent code with reduced dimensions generates a better point cloud generation than learning a raw point cloud [18].

In actual experiments, WGAN-GP was used for more stable learning [26]. The loss function is defined as:

$$\begin{aligned} L = \mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}\big[D\big(\tilde{\mathbf{x}}\big)\big] - \mathbb{E}_{\mathbf{x}\sim p_r}[D(\mathbf{x})] \\ + \lambda \mathbb{E}_{\hat{\mathbf{x}}\sim p_{\hat{x}}}\Big[\big(\big\|\nabla_{\hat{\mathbf{x}}} D\big(\hat{\mathbf{x}}\big)\big\|_2 - 1\big)^2\Big], \end{aligned} \tag{7}$$

where D is a set of 1-Lipschitz function, p_r is a distribution of real data $\mathbf{x}$, p_g is a model distribution implicitly defined as $\tilde{\mathbf{x}} = G(\mathbf{z})$, $\mathbf{z} \sim p(\mathbf{z})$, and $p_{\hat{x}}$ is distribution of random sample $\hat{\mathbf{x}}$. A penalty was imposed on the gradient norm for a random sample $\hat{\mathbf{x}} \sim p_{\hat{x}}$ to give a constraint.

Finally, the proposed method extracts the bottleneck code of the autoencoder from the input point cloud, performs generation using GAN as the bottleneck code, and then generates a point cloud through the process of reconstructing to a decoder.

IV. Experimental Results

In this section, experimental results of the proposed generative model are described. Section IV-A introduces metrics to evaluate the performance of point cloud generation. Evaluation criteria provided by Achlioptas *et al.* are used to evaluate fidelity and diversification between samples including Jensen-Shannon Divergence, Coverage, and Minimum Matching Distance [18].

A. Metrics

Let $\mathbf{P}$ be the set of reference point clouds and $\mathbf{Q}$ be the generated point clouds with $|\mathbf{P}| = |\mathbf{Q}|$.

- *Jensen-Shannon Divergence (JSD):* a measure of the distance between two probability distributions P_r and P_g, defined as:

$$JSD\big(P_r \| P_g\big) = \frac{KL(P_r \| M) + KL\big(P_g \big\| M\big)}{2}, \tag{8}$$

  where $M = \frac{1}{2}(P_r + P_g)$. P_r and P_g are approximated by discretizing the space into 28^3 regular voxels and assigning each point to one of them. The JSD between marginal distributions is defined in the Euclidean 3D space by assuming that point cloud data are axis-aligned and a canonical voxel grid is in the ambient space. One can measure the similarity between locations of two point clouds, and grid-resolution affects the granularity of measurements. The JSD is a symmetric enhanced version of the Kullback-Leibler Divergence using the average of differences between the mean distribution. For that reason, the JSD is more appropriate to make a balanced tessellation of voxels.
- *Coverage (COV):* measure of generative capabilities in terms of richness of generated samples from the model. For two point cloud sets $\mathbf{P}$ and $\mathbf{Q}$, the coverage is defined as a fraction of point clouds in $\mathbf{Q}$ that are, in the given metric, the nearest neighbor to some point cloud in $\mathbf{P}$.

$$\mathrm{COV}(P_r, P_g) = \frac{\big|\big\{\arg\min_{\mathbf{P}\in P_r} D\big(\mathbf{P}, \mathbf{Q}\big|\mathbf{Q} \in P_g\big)\big\}\big|}{|P_r|}, \tag{9}$$

  where $D(\cdot, \cdot)$ can be either CD or EMD.
- *Minimum Matching Distance (MMD):* Since COV only considers the nearest point clouds into account and does not depend on the distance between the matchings, additional metric was introduced. For point cloud sets $\mathbf{P}$ and $\mathbf{Q}$, MMD is a measure of the fidelity of $\mathbf{P}$ with respect to $\mathbf{Q}$. For each point cloud in the reference set, the distance to its nearest neighbor in the generated set is computed and averaged:

$$\mathrm{MMD}\big(P_r, P_g\big) = \frac{1}{|P_r|} \sum_{\mathbf{P}\in P_r} \min_{Q\in P_g} D(\mathbf{P}, \mathbf{Q}), \tag{10}$$

  where $D(\cdot, \cdot)$ can be either CD or EMD.

In addition, 1-nearest neighbor accuracy (1-NNA) measures the similarity between different shape distributions. However, we did not mention 1-NNA since JSD, COV, and MME sufficiently cover the objective performance.

TABLE II
Reconstruction Capabilities of the Models

Category	Method	MMD-CD	MMD-EMD
Airplane	AE [18]	0.8	5.8
	Ours	**0.4**	**4.2**
Car	AE [18]	0.9	6.5
	Ours	**0.6**	**5.1**
Chair	AE [18]	1.3	7.2
	Ours	**0.7**	**5.3**

MMD-CD scores are multiplied by 10^3; MMD-EMD scores are multiplied by 10^2. The lower the better.

B. Implementation Details

For the experiment, we used ShapeNet dataset including airplane, car and chair [27]. Since the proposed work is focused on the reconstruction of consumer products, such as furniture and automobiles, ShapeNet's taxonomy is the most suitable. In addition, ShapeNet keeps updated large-scale dataset of 3D shapes with rich annotation and can cover most of other datasets. We trained models with point clouds from a single object class and worked with train, validation, and test sets of ratio [85%, 5%, 10%].

Hyperparameters of the network used for the experiment in the proposed method are as follows: The autoencoder uses 2048×3 point clouds with neighbors $K = 20$ as input. The encoder is a LDGCNN-based network with four EdgeConv layers to extract geometric features [16]. The EdgeConv layer uses shared fully-connected layers (64, 64, 128, 256) and each layers includes ReLU and batch normalization. The decoder transforms the latent vector to the output of 2048×3 point cloud by using three fully-connected layers (256, 256, 2048) with ReLU except for the last layer. We used Adam with initial learning rate of 0.001 to optimize the autoencoder, and the batch size of 50 [28].

The network structures of the generator and discriminator is the same to that of LatentGAN [18]. We used WGAN-GP with gradient penalty regularizer $\lambda = 10$ [26]. The generator consists of two fully connected layers (128, 128) with ReLU, and the discriminator consists of two fully-connected layers (256, 512) with sigmoid activation. We used Adam with initial learning rate of 0.0001, $\beta_1 = 0.5$, and batch of size 100 [28]. The latent vector was drawn by a spherical Gaussian of 128 dimensions with zero mean and 0.2 units of standard deviation.

C. Reconstruction Capabilities

For the experiment, we evaluated the reconstruction capabilities of the proposed autoencoders using test data. Table II shows the MMD-CD and MMD-EMD between the reconstructed point clouds and their corresponding ground-truth in the test set of the chair category. The autoencoders trained with the CD loss on training data of the chair category. Fig. 4 shows the ground truth, reconstruction results of PointNet-based autoencoder and LDGCNN-based autoencoder, respectively [9], [16]. As shown in the result, the ground truth has non-uniform point cloud while PointNet-based autoencoder reconstructs point cloud better than the ground truth. However there still exist holes and discontinuous areas. On the other

Fig. 4. Examples of reconstructed point clouds by autoencoder. From top to bottom: airplane, car, and chair. From left to right: ground truth, PointNet-based autoencoder [18], and our LDGCNN-based autoencder.

TABLE III
COMPARISON OF THE NUMBER OF PARAMETERS BETWEEN DIFFERENT AUTOENCODER NETWORKS

Method	#Parameters (M)
AE [18]	1.77
Ours	**0.55**

hand, the proposed method reconstructs better edges and more continuous planar regions than the ground truth and PointNet. Because LDGCNN maintains the shape of the point cloud well through local geometrical features and can understand the context of the semantically similar structures, it can uniformly reconstruct the sparse input. As shown in Table III, the model size of our LDGCNN-based autoencoder is smaller than that of PointNet-based autoencoder.

D. Generative Capabilities

Table IV compares the generation results of our method with existing generative models including: RawGAN and LatentGAN [18]. All pre-trained autoencoders use CD for the reconstruction loss. Although the proposed algorithm has poor performance when compared to LatentGAN in the airplane case, it performs the best in the sense of objective measures using LDGCNN-based autoencoder with the smaller model size. Fig. 5 shows some models of the new point cloud generated using our autoencoder. The well-trained latent representations generate good-looking samples based on embeddings created by performing interpolation or simple linear algebra. Given that the complex wing structure of an airplane has been somewhat realistically constructed, it can be seen that extracting regional geometric features is effective in learning the distribution of data.

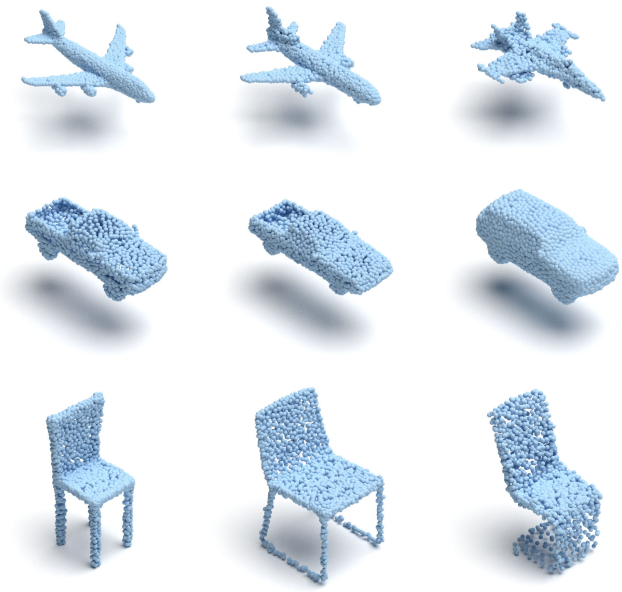

Fig. 5. Examples of point clouds generated by our model. From top to bottom: airplane, car, and chair. The processing time of airplane, car, and chair data are 29.89ms for 4045 point sets, 50.72ms for 7497 point sets, and 45.73ms for 6778 point sets, respectively.

Fig. 6. Examples of a generated point cloud applied to an AR applications.

V. CONCLUSION

In this paper, we adopted an autoencoder using local geometric information that builds optimized latent space to generate a new 3D point clouds. The use of local geometric information enables semantically similar structures to be captured in the feature space of deep layers, helping to create a more robust point cloud than conventional methods when the noise is present or a part of data are lost.

Also, through interpolation, one of the most important features of latent space produces a smooth transition between two

TABLE IV
EVALUATING GENERATIVE CAPABILITIES

Category	Method	JSD (↓)	MMD (↓)		COV (%, ↑)	
			CD	MMD	CD	MMD
Airplane	RawGAN [18]	7.44	0.261	5.47	42.72	18.02
	LatentGAN (CD) [18]	4.62	0.239	4.27	43.21	21.23
	LatentGAN (MMD) [18]	**3.61**	0.269	**3.29**	**47.90**	**50.62**
	PointFlow [25]	4.92	**0.217**	**3.24**	**46.91**	**48.40**
	Ours (CD)	**4.41**	**0.229**	4.42	43.13	21.09
Car	RawGAN [18]	12.8	**1.27**	8.74	15.06	9.38
	LatentGAN (CD) [18]	4.43	1.55	6.25	38.64	18.47
	LatentGAN (MMD) [18]	**2.21**	1.48	**5.43**	39.20	**39.77**
	PointFlow [25]	**0.87**	**0.91**	**5.22**	**44.03**	**46.59**
	Ours (CD)	3.53	1.32	6.04	**39.46**	19.25
Chair	RawGAN [18]	11.5	2.57	12.8	33.99	9.97
	LatentGAN (CD) [18]	4.59	2.46	8.91	41.39	25.68
	LatentGAN (MMD) [18]	2.27	2.61	**7.85**	40.79	**47.69**
	PointFlow [25]	**1.74**	**2.42**	**7.87**	**46.83**	**46.98**
	Ours (CD)	**2.09**	**2.42**	8.35	**43.53**	24.37

MMD-CD scores are multiplied by 10^3; MMD-EMD scores and JSD are multiplied by 10^2. ↑: the higher the better, ↓: the lower the better. The best scores are highlighted in bold red, and second scores are highlighted in bold blue.

distinct types of same-class objects using only simple addition and subtraction operations. Our model is attractive in that it can change the characteristics of various objects such as the shape and legs of the chair at once.

The LDGCNN-based autoencoder has a longer runtime than PointNet-based autoencoder because the each EdgeConv layer uses a non-parallel KNN to find neighbors. In addition, since the latent representation is described as a floating-point number, compression efficiency is limited. Further work includes the replacement of the network, which extracts local geometric information but is lighter than LDGCNN, and the integration of autoencoder and LatentGAN modules. Since an autoencoder has a data compression capability, the amount of data for the latent representation can be greatly reduced [29].

Generating a 3D point cloud through deep learning suggests that we can easily have a realistic 3D model without having any expertise in computer graphics. However, generation of a realistic 3D content requires a large amount of training data. Since there are few 3D training datasets, the proposed method can easily generate 3D datasets instead real acquisition using a 3D sensor.

In addition, the generated 3D point cloud can be used in various applications. Fig. 6 shows an example of the 3D model to render various 3D objects in AR application for interior design and driving simulation, to name a few. For an example, a consumer can choose a piece of furniture that matches their surroundings through the simulation by generating 3D models, which helps the consumer's decision to purchase the product. For another example, a gamer can apply the proposed method to the interactive game by creating various 3D objects using point clouds [30].

REFERENCES

[1] P. Cambou. "3D Imaging and Sensing Technologies: An Expected Explosion Within the Consumer Market Segment." May 2017. [Online]. Available: https://www.i-micronews.com/3d-imaging-sensing-technologies-an-expected-explosion-within-the-consumer-market-segment (accessed Jan. 2, 2021).

[2] "Specifications for XPERIA XZ1." [Online]. Available: https://www.sony.co.uk/electronics/support/mobile-phones-tablets-mobile-phones/xperia-xz1/specifications (accessed Oct. 10, 2021).

[3] "iPhone X—Technical Specifications." [Online]. Available: https://support.apple.com/kb/SP770 (accessed Oct. 10, 2021).

[4] L. Cui and E. S. Jang, "Palette-based color attribute compression for point cloud data," *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 6, pp. 3108–3120, 2019.

[5] J.-H. Cho, S.-Y. Kim, Y.-S. Ho, and K. H. Lee, "Dynamic 3D human actor generation method using a time-of-flight depth camera," *IEEE Trans. Consum. Electron.*, vol. 54, no. 4, pp. 1514–1521, Nov. 2008.

[6] S.-Y. Kim, J.-H. Cho, A. Koschan, and M. A. Abidi, "3D video generation and service based on a TOF depth sensor in MPEG-4 multimedia framework," *IEEE Trans. Consum. Electron.*, vol. 56, no. 3, pp. 1730–1738, Aug. 2010.

[7] S.-F. Tsai, C.-C. Cheng, C.-T. Li, and L.-G. Chen, "A real-time 1080p 2D-to-3D video conversion system," *IEEE Trans. Consum. Electron.*, vol. 57, no. 2, pp. 915–922, May 2011.

[8] J. L. Herrera, C. R. del-Blanco, and N. García, "A novel 2D to 3D video conversion system based on a machine learning approach," *IEEE Trans. Consum. Electron.*, vol. 62, no. 4, pp. 429–436, Nov. 2016.

[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[10] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2017, pp. 5099–5108.

[11] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.

[12] J. Li, B. M. Chen, and G. Hee Lee, "So-Net: Self-organizing network for point cloud analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9397–9406.

[13] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 87–102.

[14] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2018, pp. 820–830.

[15] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9621–9630.

[16] K. Zhang, M. Hao, J. Wang, C. W. de Silva, and C. Fu, "Linked dynamic graph CNN: Learning on point cloud via linking hierarchical features," 2019, *arXiv:1904.10014*.

[17] Y. Zhang and M. Rabbat, "A graph-CNN for 3D point cloud classification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2018, pp. 6279–6283.
[18] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas, "Learning representations and generative models for 3D point clouds," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 40–49.
[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 9–10.
[20] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2014, pp. 2672–2680.
[21] "What are the Pros and Cons of Generative Adversarial Networks vs Variational Autoencoders?" [Online]. Available: https://www.quora.com/What-are-the-pros-and-cons-of-Generative-Adversarial-Networks-vs-Variational-Autoencoders (accessed Oct. 10, 2021).
[22] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," 2016, *arXiv:1606.03498*.
[23] M. Zamorski *et al.*, "Adversarial autoencoders for compact representations of 3D point clouds," *Comput. Vis. Image Understand.*, vol. 193, Apr. 2020, Art. no. 102921.
[24] Y. Sun, Y. Wang, Z. Liu, J. Siegel, and S. Sarma, "PointGrow: Autoregressively learned point cloud generation with self-attention," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2020, pp. 61–70.
[25] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3D point cloud generation with continuous normalizing flows," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 4541–4550.
[26] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., 2017.
[27] A. X. Chang *et al.*, "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.
[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[29] K. Matsuzaki and K. Tasaka, "Binary representation for 3D point cloud compression based on deep auto-encoder," in *Proc. IEEE 8th Global Conf. Consum. Electron. (GCCE)*, 2019, pp. 489–490.
[30] J.-P. Virtanen *et al.*, "Interactive dense point clouds in a game engine," *ISPRS J. Photogrammetry Remote Sens.*, vol. 163, pp. 375–389, May 2020.

Sohee Lim (Student Member, IEEE) was born in Daejeon, South Korea, in 1995. She received the B.S. degree in electronics and control engineering from Hanbat National University, South Korea, in 2019. She is currently pursuing the M.S. degree in digital imaging engineering with Chung-Ang University. Her research interests include 3D vision, and deep learning.

Minwoo Shin (Student Member, IEEE) was born in Buyeo, South Korea, in 1992. He received the B.S. degree in electronics and information engineering from Konyang University, South Korea, in 2017, and the M.S. degree in image science from Chung-Ang University, South Korea, in 2019, where he is currently pursuing the Ph.D. degree in image science. His research interests include camera calibration and visual odometry.

Joonki Paik (Senior Member, IEEE) was born in Seoul, South Korea, in 1960. He received the B.Sc. degree in control and instrumentation engineering from Seoul National University, Seoul, in 1984, and the M.Sc. and Ph.D. degrees in electrical engineering and computer science from Northwestern University, Evanston, IL, USA, in 1987 and 1990, respectively. From 1990 to 1993, he was with Samsung Electronics, where he designed the image stabilization chip sets for consumer camcorders. Since 1993, he has been a Faculty Member with Chung-Ang University, Seoul, where he is currently a Professor with the Graduate School of Advanced Imaging Science, Multimedia, and Film. From 1999 to 2002, he was a Visiting Professor with the Department of Electrical and Computer Engineering, The University of Tennessee, Knoxville, TN, USA. Since 2005, he has been the Head of the National Research Laboratory in the field of image processing and intelligent systems. In 2008, he was a Full-Time Technical Consultant for the System LSI Division, Samsung Electronics, where he developed various computational photographic techniques, including an extended depth of field systems. From 2005 to 2007, he was the Dean of the Graduate School of Advanced Imaging Science, Multimedia, and Film and the Director of the Seoul Future Contents Convergence Cluster established by the Seoul Research and Business Development Program. He was a recipient of the Chester-Sall Award from the IEEE Consumer Electronics Society, the Academic Award from the Institute of Electronic Engineers of Korea, and the Best Research Professor Award from Chung-Ang University. He has served the Consumer Electronics Society of IEEE as a member of the Editorial Board. He is currently a member of the Presidential Advisory Board for Scientific/Technical Policy with the Korean Government and is a technical consultant for the Korean Supreme Prosecutors Office for computational forensics.