remote sensing

MDPI

Article

Deep Learning-Based Point Upsampling for Edge Enhancement of 3D-Scanned Data and Its Application to Transparent Visualization

Weite Li [1,*], Kyoko Hasegawa [2], Liang Li [2], Akihiro Tsukamoto [3] and Satoshi Tanaka [2]

[1] Graduate School of Information Science and Engineering, Ritsumeikan University, Kusatsu 525-8577, Shiga, Japan
[2] College of Information Science and Engineering, Ritsumeikan University, Kusatsu 525-8577, Shiga, Japan; hasegawa@media.ritsumei.ac.jp (K.H.); liliang@fc.ritsumei.ac.jp (L.L.); stanaka@is.ritsumei.ac.jp (S.T.)
[3] Graduate School of Integrated Arts and Sciences, Tokushima University, Tokushima 770-8502, Tokushima, Japan; tsukamoto.akihiro@tokushima-u.ac.jp
* Correspondence: is0290fh@ed.ritsumei.ac.jp

Abstract: Large-scale 3D-scanned point clouds enable the accurate and easy recording of complex 3D objects in the real world. The acquired point clouds often describe both the surficial and internal 3D structure of the scanned objects. The recently proposed edge-highlighted transparent visualization method is effective for recognizing the whole 3D structure of such point clouds. This visualization utilizes the degree of opacity for highlighting edges of the 3D-scanned objects, and it realizes clear transparent viewing of the entire 3D structures. However, for 3D-scanned point clouds, the quality of any edge-highlighting visualization depends on the distribution of the extracted edge points. Insufficient density, sparseness, or partial defects in the edge points can lead to unclear edge visualization. Therefore, in this paper, we propose a deep learning-based upsampling method focusing on the edge regions of 3D-scanned point clouds to generate more edge points during the 3D-edge upsampling task. The proposed upsampling network dramatically improves the point-distributional density, uniformity, and connectivity in the edge regions. The results on synthetic and scanned edge data show that our method can improve the percentage of edge points more than 15% compared to the existing point cloud upsampling network. Our upsampling network works well for both sharp and soft edges. A combined use with a noise-eliminating filter also works well. We demonstrate the effectiveness of our upsampling network by applying it to various real 3D-scanned point clouds. We also prove that the improved edge point distribution can improve the visibility of the edge-highlighted transparent visualization of complex 3D-scanned objects.

Keywords: transparent visualization; point upsampling; 3D-scanned point cloud; opacity-based edge highlighting; 3D edges; deep learning

check for updates

Citation: Li, W.; Hasegawa, K.; Li, L.; Tsukamoto, A.; Tanaka, S. Deep Learning-Based Point Upsampling for Edge Enhancement of 3D-Scanned Data and Its Application to Transparent Visualization. *Remote Sens.* **2021**, *13*, 2526. https://doi.org/10.3390/rs13132526

Academic Editor: Angel D. Sappa

Received: 6 May 2021
Accepted: 25 June 2021
Published: 28 June 2021

1. Introduction

The recent development of 3D-scanning technology has enabled the rapid and accurate recording of complex objects in the real world. The 3D-scanned data exist in the form of a large-scale point cloud, which records the complex 3D structure of the object. When the scanned object has internal 3D structures and the acquired point cloud records the internal structures as well as the surficial shapes, the complexity of the 3D structure becomes significant. To analyze the entire 3D structure of this complex point cloud data, point-based transparent visualization [1] is effective. Transparent visualization enables us to observe the internal structures and the surficial shapes simultaneously. However, transparent visualization can provide too much information, which may increase clutter and decrease visibility. To solve this problem, Kawakami et al. [2] proposed opacity-based edge highlighting, which combines the edge-highlighting technique with transparent visualization based on stochastic point-based rendering (SPBR) [1,3,4]. Highlighting 3D edges improves the transparent visualization of 3D-scanned complex objects to a great extent.

The above approach, i.e., highlighting 3D edges in transparent visualization, assumes that the points are sufficiently dense and uniform along the edges. Unfortunately, this assumption does not always hold for 3D-scanned point data. Therefore, upsampled points are needed to realize the assumed situation. However, the upsampling of point data is difficult because each constituent point does not have information on the neighboring points. This lack of information makes it difficult for us to generate new points between points. Therefore, we propose using the deep learning technique for effective and reliable upsampling. Once a sufficiently dense and uniform point cloud is generated on 3D edges, we can execute the steps in opacity-based edge highlighting; we use the opacity control mechanism enabled by SPBR to increase opacity, that is, the brightness of the edge area. Thus, the new contributions of this paper can be summarized as follows. (1) We propose a deep neural network suitable for point upsampling that focuses on the point proliferation of 3D edges. (2) We propose applying our neural network to realize comprehensible transparent visualization with high-quality edge highlighting.

The problem of upsampling point clouds is essentially similar to the super-resolution problem of images. However, because point clouds do not have any spatial order or regular structure, unlike the image space represented by a regular grid, simple interpolation between the input points does not give satisfactory results. Early approaches tried various optimization strategies to generate upsampled point clouds without using deep learning models to solve this problem. For example, Alexa et al. [5] upsampled points by referring to Voronoi diagrams, which require surface-smoothing assumptions and are computed on a moving least squares surface. Subsequently, the locally optimal projection (LOP) operator [6,7] was shown to be effective for point resampling and surface reconstruction based on the L_1 median. The L_1 median is defined as any point that minimizes the sum of Euclidean distances to all points in the point cloud, especially for point clouds with noisy or outlier points.

In recent years, deep neural networks have achieved outstanding performances in various point cloud-processing tasks, including object detection [8,9], point cloud completion [10,11], shape classification [12–14], and semantic scene segmentation [15,16]. In the field of point cloud upsampling, Yu et al. [17] proposed the first deep learning algorithm for point cloud upsampling, which works on patches by learning multilevel per-point features and expands the point cloud by multibranch convolution. Yifan et al. [18] proposed a patch-based progressive point cloud upsampling method, which can gradually upsample the input to a relatively large upsampling rate. Recently, Li et al. [19] proposed a GAN-based framework to generate a high-quality upsampled point cloud. Existing methods have achieved well-performing results on synthetic datasets such as ShapeNet [20] and ModelNet40 [21]. However, for large-scale 3D-scanned point clouds with sparse points, especially for the edge data extracted based on point feature values in our previous work [2], the points generated during upsampling are usually not concentrated near the edges, and their visibility cannot be effectively improved due to the additional noise generated by upsampling. Therefore, in this paper, we propose an upsampling network adapted to the 3D edges of 3D-scanned point cloud data, which exhibits good robustness to various types of edges. The proposed method can be applied to both synthetic point cloud edges and scanned point cloud edges.

This paper is organized as follows. In Section 2, we briefly review the eigenvalue-based point feature values used in edge extraction. In Section 3, we review SPBR and opacity-based edge highlighting, which can relate edge extraction to comprehensible transparent visualization. In Section 4, we describe the steps in our method, which realizes the comprehensible transparent visualization of 3D-scanned point cloud data with edge highlighting. Details of our deep learning neural network are also explained in this section. In Section 5, we offer experiments that demonstrate the effectiveness of our method by applying it to real 3D-scanned point cloud data. We show that our neural network successfully increases the point density in the 3D-edge regions and improves the connectivity of the edge lines. We also demonstrate that our method improves the visibility of the transparent visualization of the 3D-scanned objects. In Section 6, we discuss the experimental results. In Section 7, we summarize our achievements and describe future work.

2. Eigenvalue-Based 3D Feature Values

To achieve edge extraction of the target point cloud, we adopt a statistical method [22–24]. The method uses an appropriate eigenvalue-based 3D feature value to extract high curvature areas, which are regarded as 3D edges of the point-based surface. For a local spherical region centered at each point, variances and covariances of point distributions are numerically calculated, and the local 3D structure tensor [25] is defined. Then, the 3D feature value is calculated by using the three eigenvalues of the tensor, and the value is assigned to the central point.

To date, several kinds of eigenvalue-based 3D feature values have been proposed [25]. In our work, we adopt change-of-curvature C_λ and linearity L_λ:

$$C_\lambda = \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \tag{1}$$

$$L_\lambda = \frac{\lambda_1 - \lambda_2}{\lambda_1} \tag{2}$$

where λ_1, λ_2, and λ_3 are the eigenvalues of the 3D structure tensor with $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$. C_λ measures the minimal extension of the local point distribution that shows the degree of deviation from the tangential plane at each point, and L_λ measures the difference between the two independent-directional largest extensions of the local point distribution.

The limitation of the abovementioned edge extraction method is that its reliability highly depends on the quality of the target point cloud. In visualization, a nonuniform point distribution leads to unclear edges, and a local deficiency of points leads to disconnection of the edge lines. Therefore, an upsampling of the extracted 3D edges of the point data is required. Our proposed upsampling method is explained in Section 4.

3. Methods for Transparent Visualization and Edge Highlighting

In this section, we briefly review the two methods that we use in our study. In Section 3.1, we explain the stochastic point-based rendering [1,3,4], which is the point-based high-quality transparent visualization method. In Section 3.2, we explain the opacity-based edge-highlighting method [2], which visually highlights 3D edges based on local opacity control. These methods are both suitable for 3D-scanned point clouds. Their combined use realizes a comprehensible transparent visualization of 3D-scanned point clouds with edge highlighting.

3.1. Stochastic Point-Based Rendering (SPBR)

Traditionally, transparent visualization of a 3D-scanned object has been realized by using alpha blending after transforming point cloud data into a polygon mesh. However, the polygonization of point cloud data often fails for noisy 3D-scanned data. In addition, the computational cost of alpha blending increases rapidly as the data size becomes large. The reason for the high computational cost is that alpha blending includes the depth sorting process of a great number of polygons. Therefore, in our work, we use SPBR, which is directly applicable to point cloud data and does not require a depth sorting of rendering primitives. In SPBR, the surface opacity of a visualized point-based object is determined by its point density. The surface opacity α of any local surface is controllable based on a mathematically well-defined opacity formula [1,3].

3.2. Opacity-Based Edge Highlighting

In our work, we adopt opacity-based edge highlighting. The concept of the method is to relate the local feature value to the local surface opacity. Namely, we assign higher opacity to points with larger feature values, that is, points on 3D edges. As the feature value, we adopt change-of-curvature C_λ and linearity L_λ (see Section 2). Below, we represent the feature value by f, and we normalize f such that its maximal value becomes one on the target point-based 3D-scanned surface. Since the 3D-edge regions tend to have larger f than the surrounding areas, the regions are given higher opacity and visualized more brightly.

Reference [2] proposed three types of opacity functions, which relate the feature value f to opacity α:

The type (a) function is the simplest but works robustly. This opacity function defines the simplest binary relation between feature value f and opacity α. We assign a constant high opacity α_{max} to points with f larger than a user-defined threshold f_{th}. We regard the points assigned α_{max} as edge points. We assign a constant low opacity α_{min} to points with f less than f_{th} and regard these points as non-edge points.

The type (b) function shows 3D edges as thinner lines compared to type (a). This opacity function is suitable for highlighting sharp 3D edges with high curvatures within narrow local areas. The relation of 3D feature value f and opacity α is defined as follows:

$$\alpha(f) = \frac{\alpha_{\text{max}} - \alpha_{\text{min}}}{(1 - f_{\text{th}})^d}(f - f_{\text{th}})^d + \alpha_{\text{min}} \tag{3}$$

where f_{th} is the threshold value of f, over which the corresponding local regions are regarded as 3D edges. Minimal opacity α_{min} and maximal opacity α_{max} are assigned to points with $f < f_{\text{th}}$ and $f = 1$, respectively. In the intermediate range, $f_{\text{th}} \le f < 1$, and the opacity $\alpha(f)$ increases as f becomes larger. Parameter d controls the speed of the increase in opacity. Usually, we set d in a range of $1.0 \le d \le 3.0$.

The type (c) function enables us to visualize the soft 3D edges. We define a soft 3D edge as a region that is less angled but has comparably higher curvatures than the surrounding areas. The relation of the 3D feature value f and opacity α is realized by the following opacity function:

$$\alpha(f) = \begin{cases} \frac{\alpha_{\text{max}} - \alpha_{\text{min}}}{(F_{\text{th}} - f_{\text{th}})^d}(f - f_{\text{th}})^d + \alpha_{\text{min}} & (f_{\text{th}} \le f < F_{\text{th}}) \\ \alpha_{\text{max}} & (F_{\text{th}} \le f < 1) \end{cases} \tag{4}$$

In formula (4), F_{th} and f_{th} are the two threshold values with $f_{\text{th}} < F_{\text{th}}$. The 3D-edge regions are defined as areas with $f \ge f_{\text{th}}$ as in type (a) and (b) functions. α gradually increases for $f_{\text{th}} \le f < F_{\text{th}}$ and becomes constant for $f \ge F_{\text{th}}$.

3.3. Resampling for Controlling Point Density

For the opacity-based edge highlighting explained in Section 3.2, we need to control opacity α. Since our visualization is executed by stochastic point-based rendering, we can control α, that is, by adjusting the local point density. In other words, we need to execute resampling of the given point cloud data.

In reference [2], downsampling is executed by randomly eliminating points, and upsampling is executed by copying randomly selected points. This strategy works well if the point distribution has local uniformity and does not have a local deficiency in the 3D-edge regions. However, this assumption does not always hold, as we described in Section 1. Therefore, we perform point upsampling using our deep learning-based neural network to recover the assumed condition. Details are explained in Section 4.

4. Proposed Method for Edge-Highlighting Visualization

In this section, we present our method for the edge-highlighting transparent visualization of 3D-scanned point cloud data. In Section 4.1, we summarize the steps of the method. In Section 4.2, we describe our deep learning neural network for point upsampling of the 3D-edge regions.

4.1. Steps of the Proposed Method

Our proposed method is executed in the following steps:

1. **Random downsampling of the 3D-edge regions:** Execute downsampling for points in the 3D-edge regions. We randomly eliminate points with $f > f_{\text{th}}$ such that the resultant point distribution obeys the selected opacity function (type (a), (b), or (c)). Points with $f < f_{\text{th}}$ are eliminated;
2. **Deep learning-based upsampling of the 3D-edge regions:** Execute the deep learning-based upsampling for the points obtained in STEP 1;
3. **Point integration and visualization:** Merge the original 3D-scanned points, which include points of the non-edge regions, with the upsampled edge points obtained in STEP 2. Then, stochastic point-based rendering is applied to the integrated point dataset. In this step, we obtain a transparent image of the target 3D-scanned point cloud data with clear edge highlighting.

In STEP 1, which is different from the original opacity-based edge highlighting [2], we regard α as the probability of keeping points in the downsampling. Correspondingly, we set $\alpha_{\max} = 1$, which is the maximal probability. Then, the opacity function is regarded as the probability function describing the feature value dependence of the probability. For f with $\alpha = \alpha_{\max} = 1$, the original points are fully kept. For f with $\alpha = 0$, the points are absolutely removed. For f with $0 < \alpha < 1$, the points are removed with probability $1 - \alpha$.

In STEP 2, details of the deep learning network are described in the next section. STEP 2 recovers the local distributional uniformity of the 3D-edge points and fills the point deficiency. After upsampling, the point distribution remains similar to that described by the selected opacity function. Even though the original feature of the opacity function is weakened, it is still true that the selection of type (a) is robust, the selection of type (b) is suitable for sharp edges, and the selection of type (c) is suitable for soft edges.

The integration of the two point datasets in STEP 3 is easy. All we have to do is simply appending the edge points acquired in STEP 2 to the original 3D-scanned points.

4.2. Proposed Upsampling Network

4.2.1. Overview

In our work, we aim to upsample the edge areas in 3D-scanned point clouds. For our purpose, the training data have to contain sufficient information on 3D edges. Generally, dense and continuous edge data of 3D-scanned point clouds are considered the best training data. However, it is challenging to collect large numbers of fine 3D-scanned point cloud edges for training in practice. We also attempt to use the edge data of simple models as the training dataset. However, since the edge data contain fewer local features, our upsampling network cannot learn enough features, which causes most of the new points generated in the upsampling results to be clustered in the high point density region of the initially extracted edge data. Therefore, in our training phase, we adopt training strategies similar to those of [17,19], which use polygon data to generate high-precision point cloud data for training. Specifically, we cut each polygon mesh data used for training to generate numerous local patches. To generate training point cloud data with uniform point distribution and fine detail retention, Poisson disk sampling (PDS) [26] is used to generate points on these patches as ground truth $\mathcal{T}$. Then, the ground truth data are downsampled to generate sparse input point cloud data $\mathcal{P} = \left\{p_i \in \mathbb{R}^{3\times1}\right\}_{i=1}^{N}$ with N points. As illustrated in Figure 1, the network consists of a generator and a discriminator, and the discriminator guides the generator training. Continuous adversarial training by alternating between the two models eventually makes the generator better able to perform the upsampling task. Specifically, for each input point cloud data $\mathcal{P}$, the goal of the generator is to produce dense and uniformly distributed point clouds $S = \{s_i \in \mathbb{R}^{3\times1}\}_{i=1}^{rN}$, where p_i and s_i are the coordinates of 3D points and r is the upsampling rate. The discriminator finds the fake data generated by the generator. Finally, we propose a patch-based training strategy with a joint loss function formed by adversarial loss, reconstruction loss, and repulsion loss.

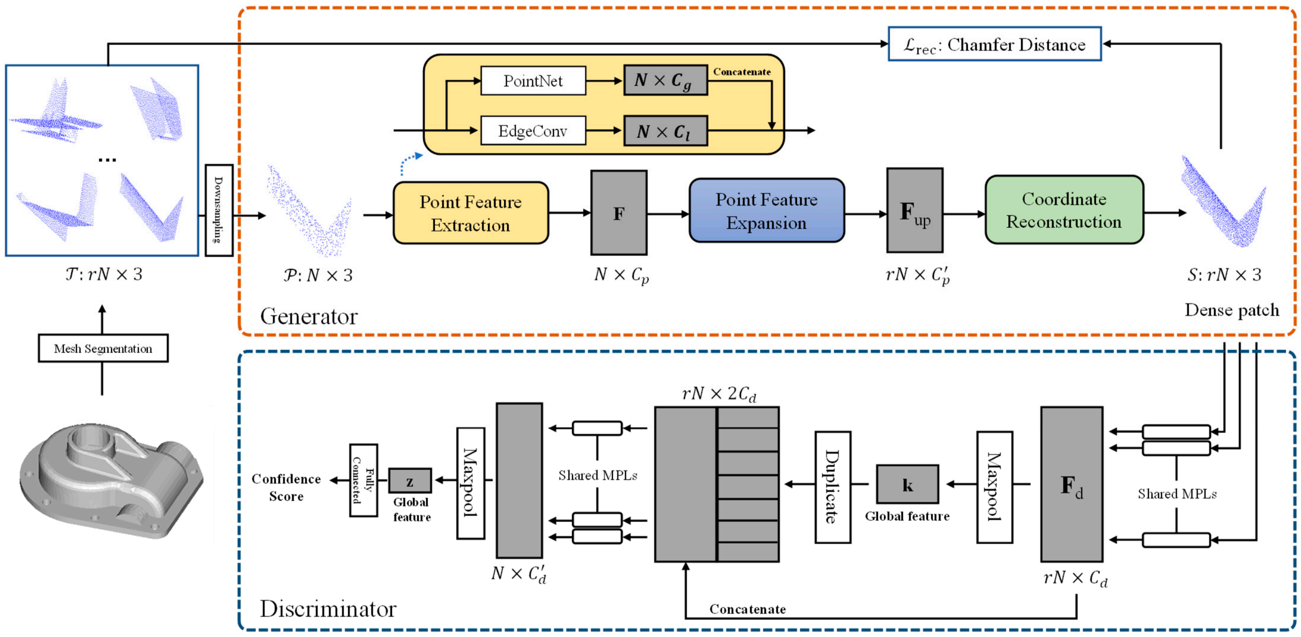

Figure 1. The architecture of the proposed upsampling network. Note that N is the number of points in input patch $\mathcal{P}$, r is the upsampling rate, and C_g, C_l, C_p, C_p', C_d, and C_d' are the numbers of feature channels. Given a sparse input patch $\mathcal{P}$ with N points, we generate a dense patch S with rN points in the generator, which consists of feature extraction, feature expansion, and coordinate reconstruction. The goal of the discriminator is to distinguish whether its input is produced by the generator.

4.2.2. Preparing the Training Data and Ground Truth

Before training the upsampling network, we preprocess all the collected polygon mesh data to generate input point cloud data and ground truth for training (see Section 5.1 for details). First, for each polygon mesh data, we randomly find 200 vertices on its surface and then grow patches around these vertices that cover 5% of the polygon mesh surface. For all generated patches, we use the PDS method to generate the point cloud data $\mathcal{T}$ with rN points and regard it as the ground truth. Finally, for each $\mathcal{T}$, we randomly select N points as the input data $\mathcal{P}$ for the network in the training phase.

4.2.3. Generator

After obtaining the sparse input point cloud data, we establish a dense point cloud generation network as a generator to densify the sparse input point cloud $\mathcal{P}$ into a dense point cloud S. As shown at the top of Figure 1, the generator consists of three modules to process input data $\mathcal{P}$, feature extraction, feature expansion, and coordinate reconstruction.

Point Feature Extraction. Feature extraction is important in the processing of discrete point cloud data, especially sparse edge point clouds. To extract the complete edge features, we propose a point feature extraction module to simultaneously extract the global feature and the context information inside local regions. PointNet [12] is effective for extracting global features of point clouds and performs well in various point cloud processing tasks. Thus, we adopt a multilayer perceptron(MLP) structure of dimension (32, 64, 64), similar to PointNet, to process each point and obtain global features with a size of $N \times C_g$ by max-pooling the output of a set of MLPs. However, only using global features cannot represent local geometric information. PointNet++ [27] is very effective and widely used for local feature extraction. For example, EC-Net, [28,29] adopts PointNet++ to extract features from the input point cloud data. However, within each local region, PointNet++ still extracts the features of each point independently without considering the relationship between neighboring points. For 3D point cloud data with a small number of points, PointNet++ feature extraction is efficient, but for large-scale 3D-scanned point cloud data, which usually contain millions or even tens of millions of points, sampling and finding neighboring points in PointNet++ consumes more memory and time as the number of points increases. Therefore, a fast and lightweight feature extraction module becomes necessary in our work since the main objective of our study is to upsample 3D-scanned point clouds. Inspired by DGCNN [30], we define the local neighborhood in the feature space and adopt a set of edge convolutions to extract local features. Given a sparse patch $\mathcal{P}$ with the size of $N \times 3$ as input, we compute the edge features of each point by applying MLPs with dimensions of (32, 64, 128) and obtain the local feature with a size of $N \times C_l$ after max-pooling among neighboring edge features, where C_l is the number of feature channels. The local neighborhood is computed by k-nearest neighbors search in feature space and dynamically updated due to different feature outputs for each layer. Then, we concatenate the local features and global features to obtain the concatenated feature $\mathbf{F}$ with a size of $N \times C_p$ and pass it to the next step for feature expansion.

Point Feature Expansion. The purpose of feature expansion is to establish the mapping from known points to more points. At present, the mainstream feature extraction methods can be roughly divided into three categories: interpolation-based methods [27], reshaping-based methods [31], and folding-based methods [32]. Interpolation-based methods usually influence feature expansion through the point interpolation relationship, but in some cases, the interpolation relationship between point clouds is unclear. Additionally, the reshaping-based method usually first expands the dimensions of the input features through a deep network such as a set of MLPs or fully connected (FC) layers and then generates the target features through a simple reshaping operation. However, the expanded features obtained in this way are closer to the input features and thus affect the upsampling quality. For example, the finally generated new points tend to be gathered near the original points. Therefore, in our work, we adopted the point feature expansion method in [19], which is a folding-based method. Compared with other feature expansion methods, the folding-based method is more flexible and has good performance in multiple applications [11,32]. In particular, the folding-based method not only avoids tedious multistep training but also promotes the generation of fine-grained information. This can save considerable memory and time for the upsampling task of 3D-scanned point cloud data and produce more refined results.

Coordinate Reconstruction. For an expanded feature with a size of $rN \times C'_p$, we regress the 3D coordinates through a series of fully connected layers of dimension (64, 3) on the feature of each point and finally output a dense patch S with a size of $rN \times 3$.

4.2.4. Discriminator

For the input data of the discriminator, the discriminator distinguishes whether it is produced by the generator, thus guiding the generator training to achieve better upsampling performance. In our work, we build a discriminator based on PCN [33] that contains two PointNet layers to extract the global feature using a pointwise multilayer perceptron with shared weights and a max-pooling operation. Specifically, for input point cloud data with a size of $rN \times 3$, we first extract a feature vector $\mathbf{F}_\mathrm{d}$ with a size of $rN \times C_d$ using an MLP of dimension (32, 64). Then, the global feature $\mathbf{k}$ is extracted from the feature vector $\mathbf{F}_\mathrm{d}$ using the max-pooling operation. We combine feature vector $\mathbf{F}_\mathrm{d}$ with global feature $\mathbf{k}$ as input to the second PointNet layer. Similar to the first PointNet layer, we obtain the new feature vector through another MLP of dimension (128, 256) and perform max-pooling to obtain the final global feature $\mathbf{z}$. Finally, the confidence scores are regressed through a set of fully connected layers of dimension (256, 1). If the confidence score is close to 0, the discriminator anticipates that the input data are most likely from the generator.

4.2.5. Loss Function

To move the generated point cloud closer to the ground truth, we adopt the Chamfer distance [34] as the reconstruction loss to evaluate the similarity between the point cloud generated by the generator and the ground truth. Specifically, the reconstruction loss is the Chamfer distance between the generated dense point cloud data S and the ground truth $\mathcal{T}$:

$$\mathcal{L}_{\text{rec}} = d_{CD}(S, \mathcal{T}) = \sum_{s_i \in S} \min_{t_i \in \mathcal{T}} \|s_i - t_i\|_2^2 + \sum_{t_i \in \mathcal{T}} \min_{s_i \in S} \|s_i - t_i\|_2^2 \tag{5}$$

where s_i is the point in the generated dense point cloud data S, t_i is the point in the ground truth $\mathcal{T}$, and $\|\cdot\|_2$ denotes the L2 norm of a vector.

The reconstruction loss moves the generated dense point cloud to fit the input point cloud better. Furthermore, to improve the local uniformity of the generated point cloud, we add the repulsion loss in [17] to more uniformly distribute the generated points rather than distributing close to the original points:

$$\mathcal{L}_{\text{rep}} = \sum_{i=0}^{rN} \sum_{i' \in K(i)} \eta(\|s_{i'} - s_i\|_2)\omega(\|s_{i'} - s_i\|_2) \tag{6}$$

where $K(i)$ is the index set of the k-nearest neighbors of point s_i. In our experiment, we set $k = 5$. $\eta(r) = -r$ is a decreasing function, and $\omega(r) = e^{-r^2/h^2}$ [6,7] is a fast-decaying weight function; they are adopted to penalize s_i when it is too close to its neighboring points in $K(i)$. Here, h is a hyperparameter of ω, and we set it to 0.03 in our experiment.

Both $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{rep}}$ calculate the pointwise distance, which ignores high-order properties generated by the point cloud, such as continuity. Therefore, we adopt the least squares method in [35] as our adversarial loss to train the generator and discriminator adversarially. For the generator, the aim is to fool the discriminator by minimizing $\mathcal{L}_{\text{gan}}(G)$ to generate S as close as possible to the ground truth:

$$\mathcal{L}_{\text{gan}}(G) = \frac{1}{2}[D(S) - c]^2 \tag{7}$$

where D is the discriminator that aims to distinguish whether its input is generated by the generator by minimizing $\mathcal{L}_{\text{gan}}(D)$, which is defined as follows:

$$\mathcal{L}_{\text{gan}}(D) = \frac{1}{2}[D(\mathcal{T}) - b]^2 + \frac{1}{2}[D(S) - a]^2 \tag{8}$$

where a, b, and c are hyperparameters. As suggested by [1], we set $a = 0$, $b = 1$, and $c = 1$ in our experiment.

Altogether, we train the network end-to-end by minimizing $\mathcal{L}_{\text{G}}$ for the generator and $\mathcal{L}_{\text{D}}$ for the discriminator:

$$\mathcal{L}_{\text{G}} = \alpha\mathcal{L}_{\text{rec}} + \beta\mathcal{L}_{\text{rep}} + \gamma\mathcal{L}_{\text{gan}}(G) \tag{9}$$

$$\mathcal{L}_D = \mathcal{L}_{\text{gan}}(D) \tag{10}$$

where α, β, and γ are the weights and empirically set as 10, 0.02, and 0.5. During the training process, the generator and discriminator are optimized alternatively.

5. Upsampling and Visualization Results and Evaluation of the Proposed Method

In this section, we show experiments applying the proposed upsampling network to real 3D-scanned data. In Section 5.1, we demonstrate the robustness of the upsampling network under different types of edges. In Section 5.2, we describe the test sets and the network training. We also summarize the parameters used in our experiments. In Section 5.3, we show the results of applying the trained network for upsampling 3D-edge regions of real 3D-scanned point clouds. We also demonstrate that our upsampling of the 3D edges successfully improves the visibility of the transparent visualization of complex 3D-scanned objects. In Section 5.4, we describe the solution strategy for treating noisy 3D-scanned data. In Section 5.5, we show the effect of the proposed method on soft edges.

5.1. Verifying the Robustness of Upsampling Networks on Simulated Edges

Since the edges of the simple model have better recognizability than the real 3D-scanned point cloud edges, to observe the effect of upsampling more intuitively, we first show the upsampling result on simulated edges in this section. The application of the proposed method to real 3D-scanned data is shown in Section 5.3. We simulate the possible situations in realistic 3D-scanned edge data by modifying the point density, removing some edges, and performing other operations. Specifically, as shown in Figure 2b, we first use the PDS method to generate the point cloud data with 50,000 points on the surface of Joint and Fandisk [36] and select the change-of-curvature as the feature value, combined with the type (a) function to extract the 3D edges. Then, to enrich the types of edge shapes, we modify the 3D edges of Joint and Fandisk differently.

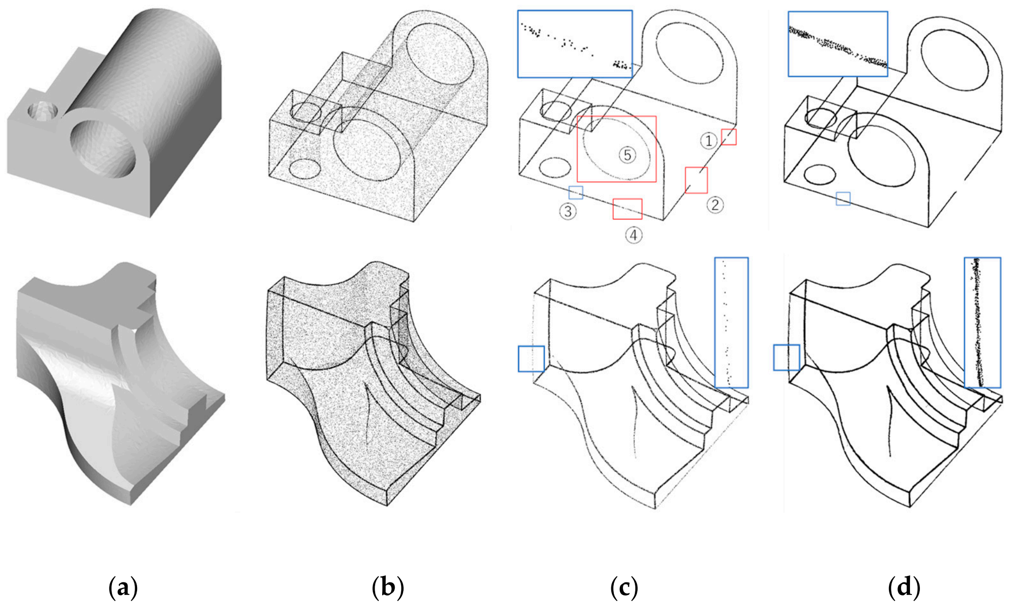

Figure 2. The upsampling results of the edge data for Joint (top) and Fandisk (bottom); (**a**) are the original polygon model, (**b**) are the original point cloud and extracted 3D edges, (**c**) are the modified edge data, and (**d**) are the upsampling results of the modified edge data.

For the edges in Joint, we simulate the absence of a point cloud in the real scene due to occlusion by artificially removing points on the edge. For example, in region 1 and region 2 (regions in red rectangles in Figure 2c), we remove all points in a short edge and a longer edge. To preserve the edge's continuity at region 3 and region 4, we remove most points in a short edge and a longer edge. In region 5, we remove 70% of the points in a circular edge. For the edges in Fandisk, we simulate the uneven point density of the 3D-scanned data in a real scene by adjusting the point density of different portions. For example, for the edge data of different Fandisk areas, we adjust the point density from 5% to 80% of the initially extracted point density.

Table 1 shows the number of points and density of the original point cloud, initially extracted edges, modified edges, and upsampling result. The upsampling results are shown in Figure 2d. The upsampling network produces numerous new points on the low point density edges in Joint and Fandisk. These edges form complete edges after upsampling.

In addition, numerous new points are also generated around edges, which improves the visibility of the overall edge shape. However, for regions 1 and 2 of the edges in Joint, the upsampling network cannot learn any information in these regions and cannot generate new points. In practice, most upsampling strategies have such a limitation. The network cannot know whether the gaps in the original data are due to missing points or if the shape of the object contains gaps. Moreover, we consider the result shown in Figure 2d to be reliable since not all gaps should be connected for realistic 3D edges. Generally, undesired gaps can be eliminated by interactively inserting support points in the gap before upsampling. Therefore, we consider that our upsampling network can be stable in most scenarios and generate numerous new points along the edges to improve the edge continuity. However, for regions where information is completely missing, new points cannot be generated to connect the edges since feature information cannot be learned.

Table 1. The number of points and density of the Joint and Fandisk data.

Objects	Scale [m]	Data	Number of Points	Density (10^6) [Points/m^3]
Joint	$0.93 \times 1.24 \times 1.18$	Original point cloud	50,000	1.55
		Extracted edges	2283	1.41
		Modified edges	1072	1.19
		Upsampling result	4290	2.99
Fandisk	$1.29 \times 1.40 \times 0.72$	Original point cloud	50,000	2.05
		Extracted edges	2518	1.95
		Modified edges	1441	1.62
		Upsampling result	5764	2.69

5.2. Datasets and Implementation Details

Since the goal of our work is to upsample the edges, redundant curved surfaces and large areas of planes in the training model will affect the quality of edge upsampling. Therefore, we selected 80 polygon mesh models from the released PU-GAN datasets [19], most of which contain more edge shapes and fewer curved surfaces. We selected 65 of these models as the training dataset and 15 models as the test set. As we explained in Section 4.2, we adopted a patch-based training strategy. By default, we cropped 200 patches for each training model and set the number of patch points $N = 512$ and the upsampling rate $r = 4$. For the optimization, we trained the network for 200 epochs using the Adam algorithm [37] with a batch size of 24 and set the learning rate of the generator and the discriminator as 0.001 and 0.0001, respectively. We implemented our network using TensorFlow and trained it on an NVIDIA GeForce RTX 2080 Ti GPU. In the testing phase, we adopted the farthest point sampling method to find the vertices and generated a local patch containing N points for each test model. Then, we generated a dense patch through the generator and combined all the upsampled results as the final output. We selected different feature values and functions to extract edges depending on the target objects. In the transparent visualization, we set $L_R = 100$, and to fairly compare the visualization results of different methods, we unified the number of points for all edge data.

To quantitatively evaluate the upsampling quality of the 3D edges, we employed three evaluation metrics. First, we adopted the Hausdorff distance [38] and cloud-to-cloud [39] distance to evaluate the similarity between the edge data and the upsampled results; the smaller the distance was, the closer the upsampled edges were to the real edges. Then, for the edge data extracted with different feature values f, we also calculated the same feature value for the upsampled edge data to evaluate the upsampling quality for the edge portion.

In the verification experiment, we first extracted 3D edges for the testing models by using the change-of-curvature with $f_{\text{th}} = 0.25$ and randomly selected 8192 points as the ground truth (see Figure 3b). Then, we used Poisson disk sampling to sample 2048 points as input.

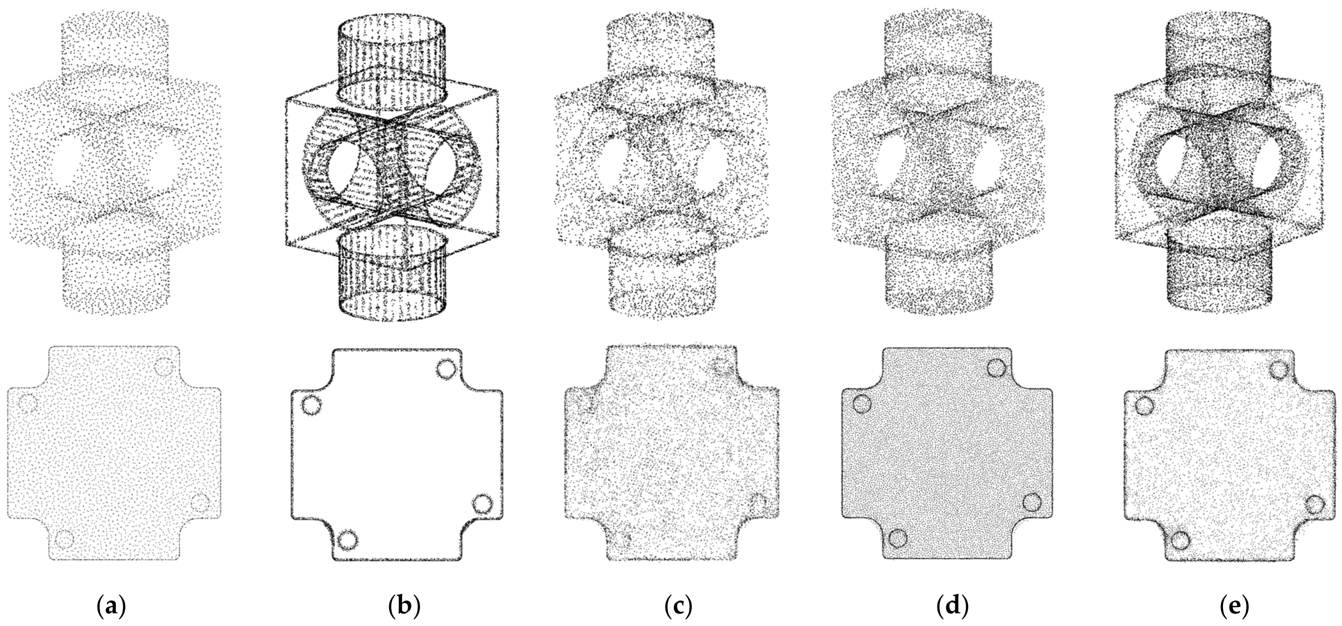

Figure 3. Qualitative verification of block (top) and cover rear (bottom). (**a**) Inputs, (**b**) ground truth, (**c**) PU-NET, (**d**) PU-GAN, (**e**) our proposed method.

We qualitatively and quantitatively compared our proposed method with the existing deep learning networks for point upsampling: PU-NET [10] and PU-GAN [19]. We show the upsampling results for block [36] and cover rear [36] with different methods in Figure 3. Compared with the other two methods, our results can generate more points at the edges. Table 2 shows the results of the quantitative comparison. Our proposed method performs optimally on all metrics.

Table 2. Quantitative verification of the effectiveness of our method.

Objects	**Methods**	**Ratio of Edge Points ($C_\lambda \geq 0.25$)**	**Cloud-to-Cloud Distance (10^{-2})**	**Hausdorff Distance**
Block	PU-NET	65.26%	3.19	0.31
	PU-GAN	58.53%	3.42	0.32
	Proposed method	82.37%	1.63	0.29
Cover rear	PU-NET	62.62%	1.51	0.72
	PU-GAN	53.77%	1.71	0.72
	Proposed method	79.83%	1.15	0.67

5.3. Application to Real 3D-Scanned Data

This section demonstrates applying the proposed upsampling network to extract 3D edges of 3D-scanned data. We also show the application of the upsampling results to edge-highlighting visualization. Then, we compare the ability of our network with existing networks for upsampling point cloud data.

5.3.1. Results of Upsampling and Visualization for 3D-Edge Regions

Figure 4 shows the 3D laser-scanned point cloud of the former Nakajima residence (24,074,424 points). The residence is a traditional Southeast Asian folk house that is preserved by the Ritto History Museum in the Shiga Prefecture of Japan. Due to limited scanning conditions, there are numerous missing parts in the roof portion. This leads to poor recognizability of the extracted edges. Figure 5a shows the result of extracting 3D edges (red parts). Standard binary extraction is adopted, using change-of-curvature C_λ

as the feature value f and regarding the regions with $f > f_{\text{th}}$ as the 3D-edge regions. In Figure 5a, we hardly recognize the 3D edges as connected lines.

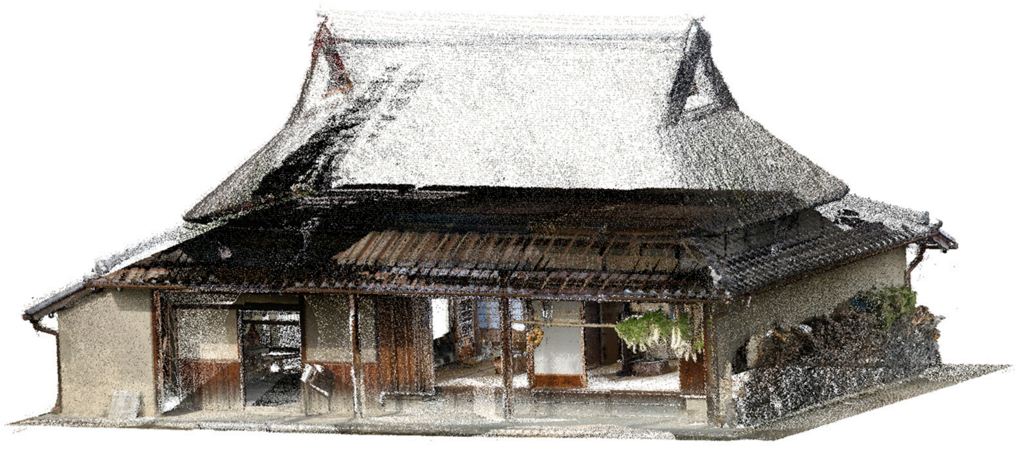

Figure 4. The 3D laser-scanned data of the former Nakajima Residence, which is a traditional Southeast Asian folk house (24,074,424 points).

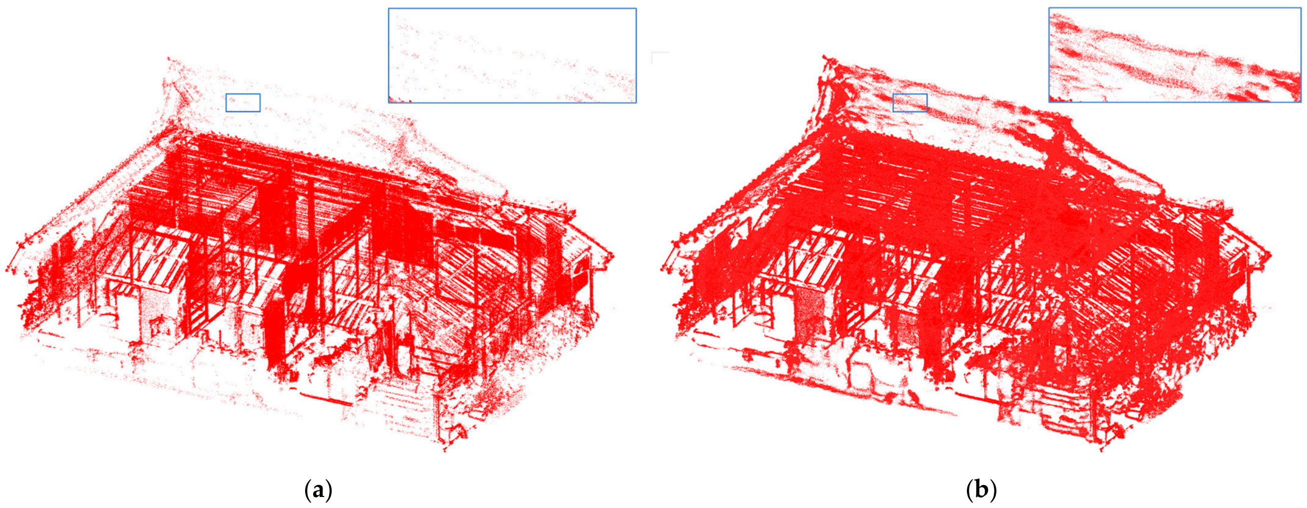

Figure 5. Extraction of edge points from 3D-scanned data in Figure 4. Standard binary extraction is adopted using change-of-curvature C_λ as the feature value f and regarding the regions with $f > f_{\text{th}}$ as the 3D-edge regions. The threshold parameter f_{th} is set to 0.25. (**a**) shows the initially extracted points, and (**b**) shows the result of applying our upsampling network.

The creation of the edge points in Figure 5a corresponds to STEP 1 of our method (see Section 4.1), adopting the type (a) function and setting $\alpha_{\max} = 1$. Figure 5b shows the result of applying our upsampling network to the edge points in Figure 5a. This upsampling corresponds to STEP 2 of our method. In Figure 5b, we can see that numerous new points are generated in the 3D-edge regions. The generated new points improve the point-distributional uniformity on the edges and improve the continuity of the edge lines. In addition, the improved edge lines, which are made denser and more continuous, provide a clearer understanding of the roof shape. Table 3 shows the number of points and density of the original point cloud, the initially extracted edges, and the upsampling edges.

Table 3. The number of points and density of the Nakajima residence data.

Data	Scale [m]	Number of Points	Density (10^5) [Points/m^3]
Original point cloud		24,074,424	8.46
Initially extracted edges	10.47 × 15.13 × 7.18	3,821,874	3.33
Upsampling edges		15,287,496	6.34

Figure 6, which shows transparent edge-highlighting visualizations by SPBR, demonstrates STEP 3 in our method. Figure 6a is created based on the integrated point dataset composed of the original 3D-scanned points and the edge points in Figure 5a. Figure 6b shows a similar visualization in which the edge points in Figure 5b are used instead. Figure 6b, which shows the result of our method after executing the three steps, improves the comprehensibility of the transparent visualization.

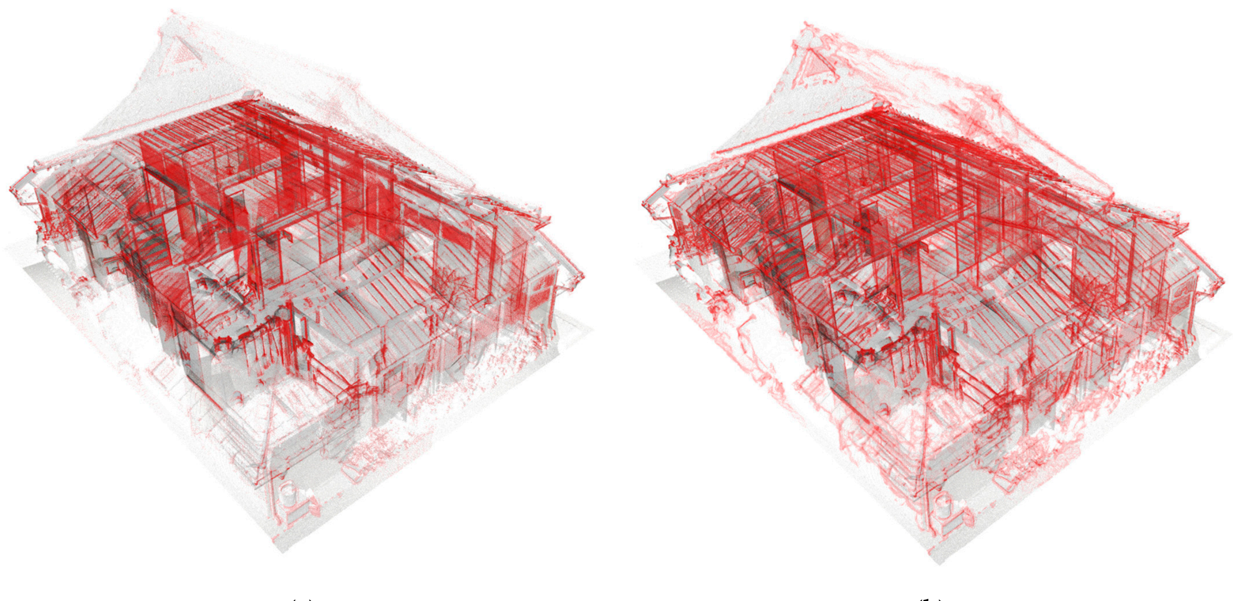
(**a**) (**b**)

Figure 6. Edge highlighting transparent visualizations of the 3D-scanned data in Figure 4. (**a**) is created based on the integrated point dataset composed of the original 3D-scanned points and the extracted edge points in Figure 5a. (**b**) shows a similar visualization in which the upsampled edge points in Figure 5b are used instead.

Let us show the experimental results of upsampling another point cloud, the laser-scanned data of a gymnasium. This laser-scanned object has many piles, which all appear as 3D edges in visualization. We adopt change-of-curvature as the feature value f and select the type (b) function to extract the 3D edges. The extracted edges are shown in Figure 7a. Although the edges of the roof portion are extracted well, the 3D edges of the front wall and surrounding objects are mostly incomplete. However, by upsampling the extracted 3D edges, we can update the disconnected edges to connect them (see the enlarged images shown in the rectangles). Table 4 shows the number of points and density of the original point cloud, the initially extracted edges, and the upsampling edges.

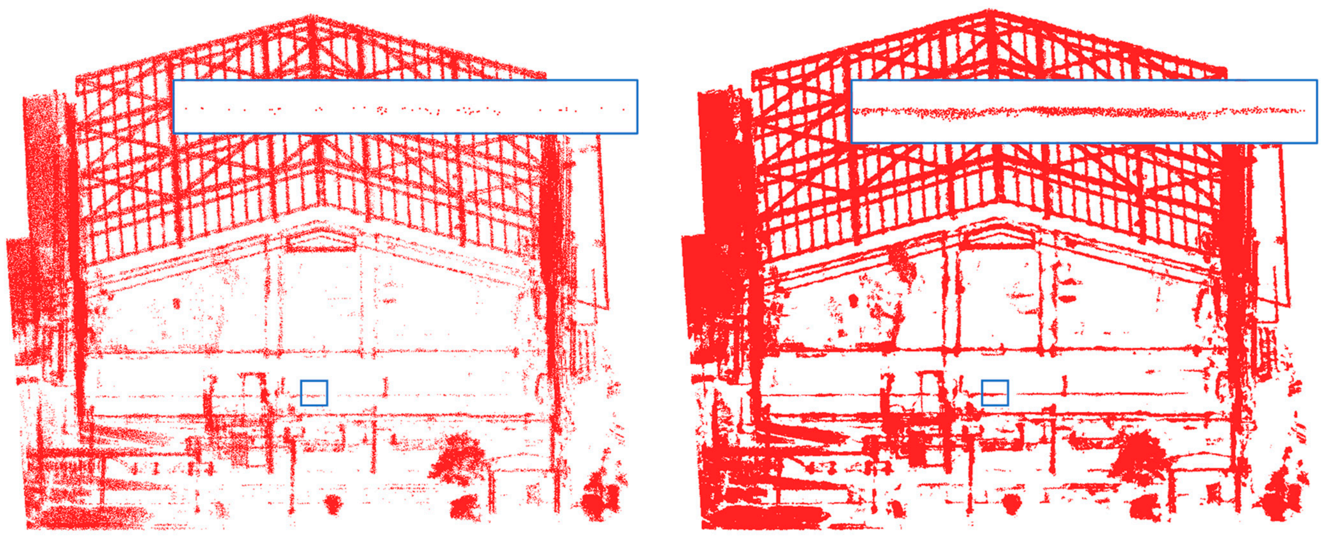
(**a**) (**b**)

Figure 7. (**a**) Points on the extracted 3D edges of the gymnasium using the change-of-curvature and type (b) function. (**b**) Points obtained by executing our upsampling for the points in (**a**). The parameters are set as follows: $f_{\text{th}} = 0.3$, $\alpha_{\min} = 0.2$, $\alpha_{\max} = 1.0$, and $d = 3.0$. Each large rectangle shows the enlarged image of the area indicated by the corresponding small rectangle.

Table 4. The number of points and density of the gymnasium data.

Data	**Scale [m]**	**Number of Points**	**Density (10^3) [Points/m^3]**
Original point cloud		5,234,550	5.21
Initially extracted edges	$25.59 \times 43.42 \times 20.04$	1,211,452	4.38
Upsampling edges		4,845,808	4.84

Figure 8a shows the result of transparent fused visualization, which is created by applying SPBR to the fused point dataset of the original laser-scanned points and the edge points in Figure 7a. Figure 8b shows a similar transparent fused visualization, for which the edge points in Figure 7b obtained by our upsampling network are used. We can see that the connectivity of the 3D edges is improved in Figure 8b, and the edge-highlighting effect is improved.

The experimental results presented in this subsection demonstrate that our method successfully increases the point density and connectivity of the 3D edges extracted from a 3D-scanned point cloud. This effect improves the visibility and comprehensibility of complex 3D-scanned objects in their transparent visualization. The goal of our work is to increase the point density in the edge regions, especially in the inhomogeneous sparse edge regions. As shown in Tables 3 and 4, our method can effectively increase the density of the edge regions, which also proves the effectiveness of the proposed method.

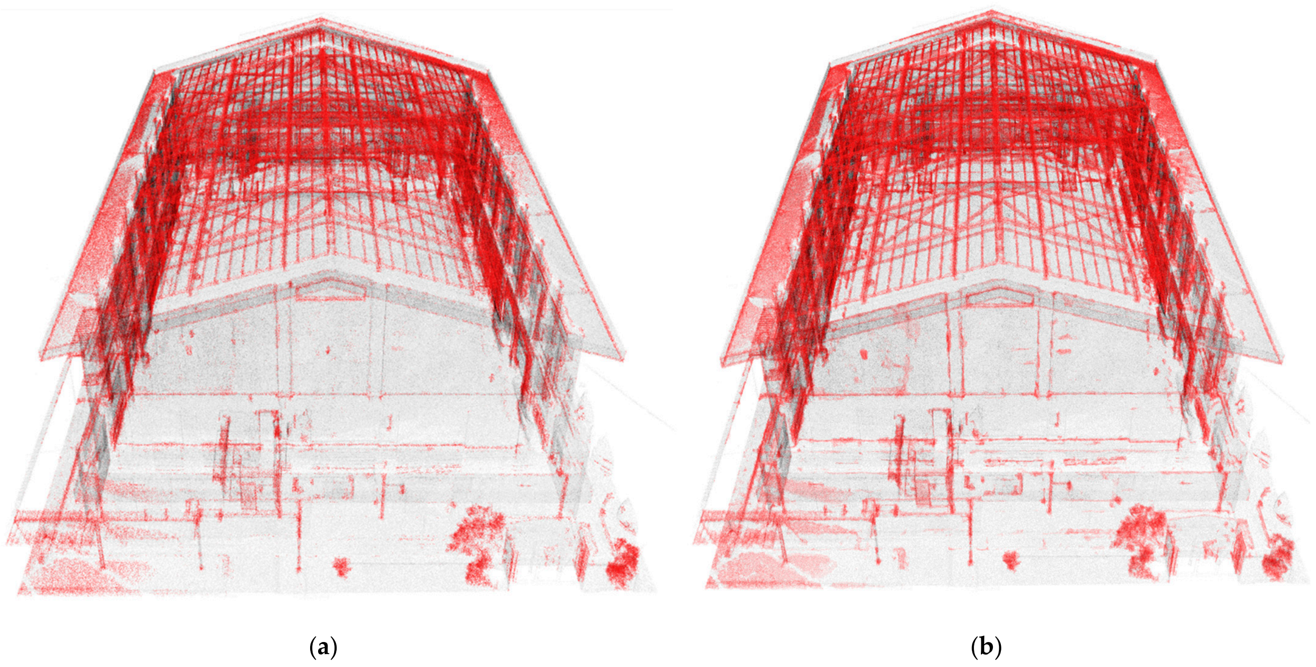

Figure 8. Fused transparent visualization of the original 3D-scanned point cloud with the extracted edge points. (**a**) shows the fusion with the edge points before upsampling, and (**b**) shows the fusion after the upsampling using our proposed network.

5.3.2. Comparison with Existing Upsampling Networks

In this subsection, we compare the upsampling results of our proposed network with those of the existing deep learning networks for point upsampling. We demonstrate that our proposed network, which is designed and trained for upsampling 3D edges, works better. For the experiments, we first use a typical computer aided design (CAD) model, as shown in Figure 9a, which is selected from ShapeNetCore [40]. To extract the 3D edges of the CAD model, we generate 50,000 points on its surface and then adopt the change-of-curvature C_λ as the feature value f with the type (a) function to extract the edges. The extracted edges are shown in Figure 9b.

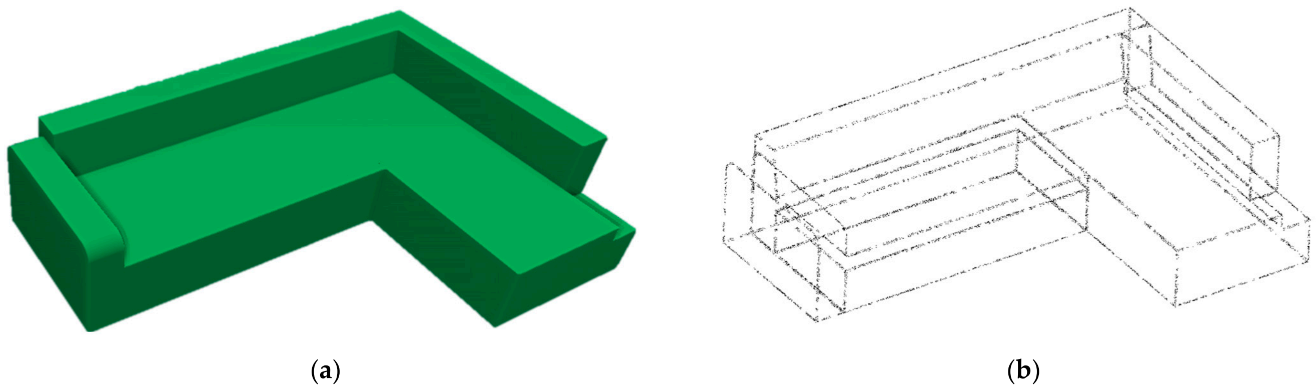

Figure 9. (**a**) CAD model of the Resort Sofa Bed. (**b**) shows the extracted edge points by adopting change-of-curvature and the type (a) function with $f_{\text{th}} = 0.2$, $\alpha_{\max} = 1.0$.

Figure 10 compares the results of upsampling the point cloud in Figure 9b by applying (a) PU-NET, (b) PU-GAN, and (c) our proposed network. For PU-NET and PU-GAN, since they are proposed to be mainly applied to the upsampling task of the overall point cloud, in the upsampling task for the edge portion, the results contain numerous points beyond the edges. In contrast, our network is more discriminative for edges and can generate new points at the correct edges.

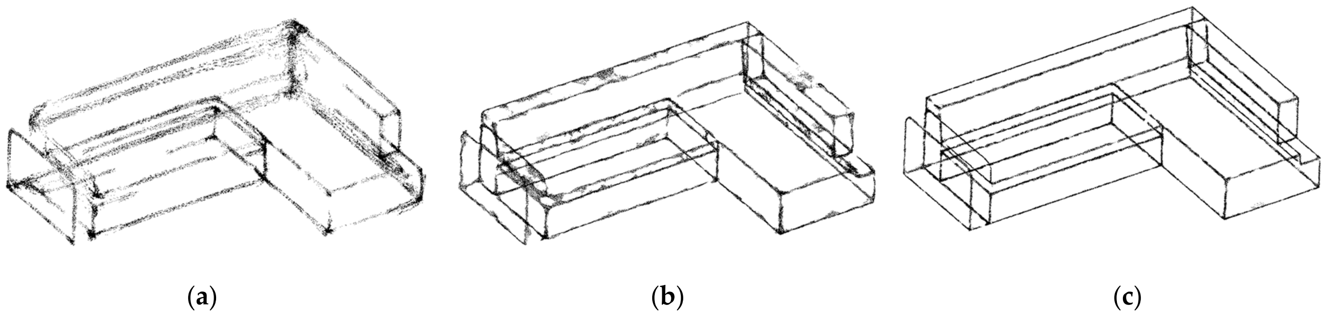

Figure 10. Results of the upsampling when using (**a**) PU-NET, (**b**) PU-GAN, and (**c**) our proposed deep learning network.

To evaluate the point feature extraction module in our proposed network, including the global and local feature extractors, we removed each of them from the network and generated upsampling results for the extracted edge points in Figure 9b. Figure 11 shows the upsampling results. In the case of using only local features, the generated new points tend to cluster more in one place on the edges, especially at the corners, where this clustering effect is more pronounced. This is due to the local features reflecting only the features in the neighborhoods, and the lack of relationships between the neighborhoods cause the points in the generated upsampled edges to cluster more in the respective local neighborhoods. Conversely, the generated upsampled edges cannot present clear edge lines when using only global features. Due to the lack of local geometric information, the generated new points cannot be clustered around the edges and even fail to form a continuous edge in some regions. Thus, as shown in Figure 10c, our proposed full point feature extraction module performs the best. The absence of either the local features or the global features reduces the overall performance, which shows that both of them contribute in the network.

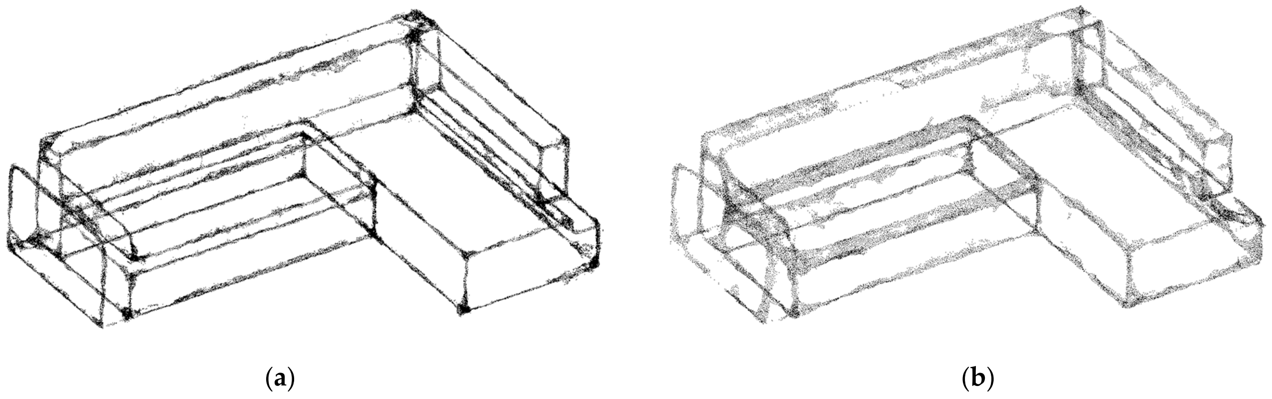

Figure 11. Ablation study on point feature extraction module. (**a**) shows the upsampling result using only local features, and (**b**) shows the upsampling result using only global features.

Then, we use our 3D-scanned data of a campus building (see Figure 12), for which we know the ground truth of the 3D edges well. Figure 13 shows the edge points extracted from the points in Figure 12. In the following, we use these edge points as the input data of the upsampling. The edge points are extracted according to the direction in STEP 1 (see Section 4.1). The type (a) function is selected, and the change-of-curvature C_λ is adopted as the feature value f with a threshold value $f_{\text{th}} = 0.25$. We set $\alpha_{\text{max}} = 0.23$, which means that the downsampling is executed excessively for our upsampling tests. We investigate the recovery of the edge points when our proposed network and the existing networks are applied. In particular, we focus on the point recovery in the selected areas shown in the rectangles.

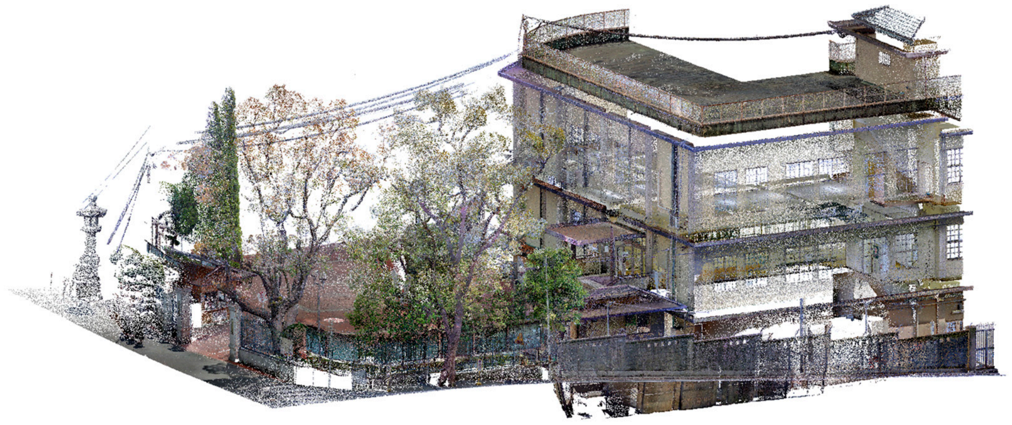

Figure 12. 3D laser-scanned data used in our comparative experiments of deep learning networks (10,480,242 points). The scanned target is the campus building of Kyoto Women's University, Japan.

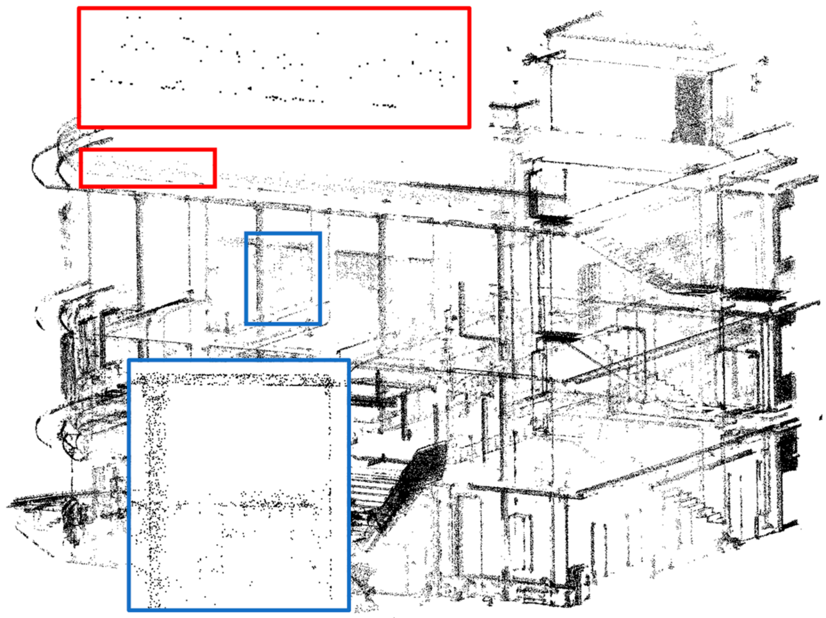

Figure 13. Edge points extracted from the 3D laser-scanned points in Figure 12. We use these edge points for our comparative study of the upsampling. Each large rectangle shows the enlarged image of the area indicated by the small rectangle with the same shape.

Figure 14 compares the results of upsampling the point cloud in Figure 13 by applying (a) PU-NET, (b) PU-GAN, and (c) our proposed network. All three networks generate new points on and around the edges to increase the point density. However, the existing networks tend to create many extra points in non-edge regions. This is because they cannot distinguish between edge and non-edge regions. In contrast, our proposed network intensively creates points in the edge regions. In the three images in Figure 14, we selected the areas where the differences mentioned above were most pronounced (see the enlarged images in the rectangles). In the enlarged images in Figure 14c, the number of points in the non-edge region increased somewhat, but the ratio is much smaller than in Figure 14a,b. As a result, our proposed network works better for increasing point density on the 3D edges and improving their connectivity.

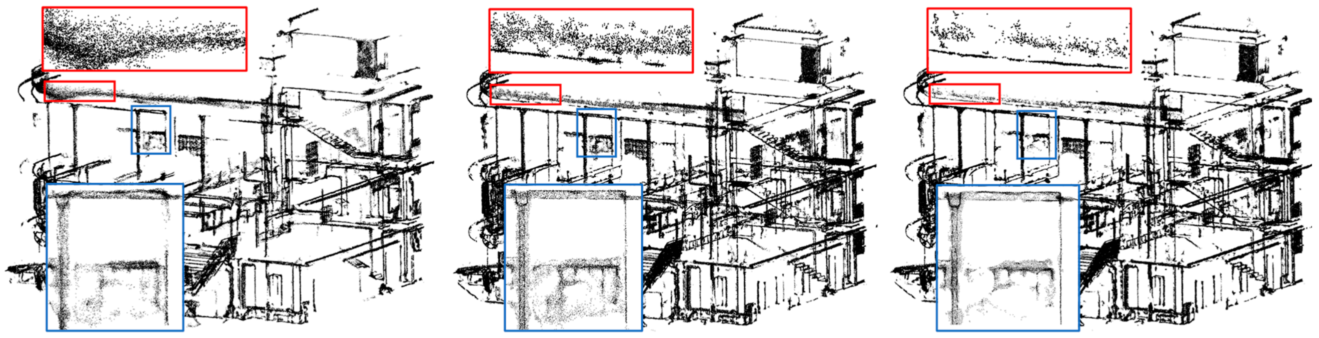

Figure 14. Results of the upsampling when using (**a**) PU-NET, (**b**) PU-GAN, and (**c**) our proposed deep learning network. Each large rectangle shows the enlarged image of the area indicated by the small rectangle with the same shape, corresponding to Figure 13.

It should also be noted here that our network is robust for the measurement noise included in the 3D-scanned data. The noise distributes the 3D-scanned points out of the edge regions. Our network tends to ignore such outliers. In addition, we can also use the statistical outlier removal (SOR) filter together with our network for very noisy data (see the following subsection for details).

We compare the performance of the three upsampling networks in Figure 15, which shows the frequency distribution of the point number with respect to the feature value. We computed the feature value f for the results obtained by PU-NET, PU-GAN, and our proposed method. We adopted change-of-curvature C_λ as the feature value f and computed it at each point in every upsampling result. We set $f_{\text{th}} = 0.25$ in the feature value-based edge extraction to extract the edge points. This means that we regard points with $C_\lambda < 0.25$ as nonedge points. Therefore, the successful upsampling of the edge points should have the following features: (i) points are intensively created within the edge region of $C_\lambda \geq 0.25$. (ii) The number of points created in the non-edge region of $C_\lambda < 0.25$ is sufficiently small. In Figure 15, we can see that the proposed method satisfies both (i) and (ii) best. A more quantitative comparison of the three networks is summarized in Table 5. First, the ratio of edge points with $C_\lambda \geq 0.25$ is higher than 90% only in our method. Second, the Hausdorff and cloud-to-cloud distances between the point clouds before and after upsampling are smallest in our method, which means that our upsampling retains the edge regions specified by the initially extracted edge extraction best.

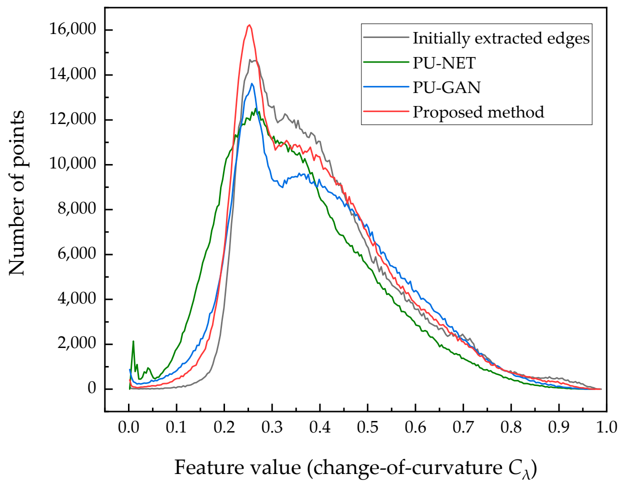

Figure 15. Frequency distribution of the point number with respect to the feature value (change-of-curvature C_λ) for the initially extracted edge points and the upsampled edge points by using the three networks.

Table 5. Quantitative comparisons of the upsampling networks.

	Ratio of Edge Points ($C_\lambda \geq 0.25$)	**Cloud-to-Cloud Distance (10^{-2})**	**Hausdorff Distance**
PU-NET	75.00%	4.42	0.89
PU-GAN	84.02%	3.53	1.44
Proposed method	92.71%	2.65	0.64

5.4. Use of Statistical Outlier Removal (SOR) Filter for Noisy Data

For noisy data, it is effective to use a statistical outlier removal filter (SOR filter) [41], which eliminates points that are sufficiently distant from other points further than the average interpoint distance. The SOR filter works well in a combined manner with our upsampling network.

For the 3D-scanned data in Figure 16, we executed experiments to investigate the effects of the SOR in its combined use with our upsampling network. First, we extracted edge points using linearity L_λ as the feature value ($f_{\text{th}} = 0.25$) and adopting the type (b) function and then increased the extracted points solely using our upsampling network. The result is shown in Figure 17. Figure 17a shows the obtained edge points, and Figure 17b shows their transparent fused visualization with the original 3D-scanned points in Figure 16. In Figure 17a, we can see that many outlier points are extracted incorrectly, suffering from measurement noise. These outlier points deteriorate the visibility of the edge-highlighted transparent visualization (see Figure 17b). In contrast, Figure 18 shows the improved results from using the SOR filter together with our upsampling network. The edge points in Figure 18a are created by applying the SOR filter before and after upsampling. In Figure 18a, we can see that the outlier points are significantly suppressed compared to Figure 17a. Figure 18b shows an edge-highlighted transparent fused visualization similar to that in Figure 17b, using the improved edge points in Figure 18a. We can see that the suppression of the outlier points results in improved visibility of the edge-highlighting transparent visualization, although the total number of edge points is reduced.

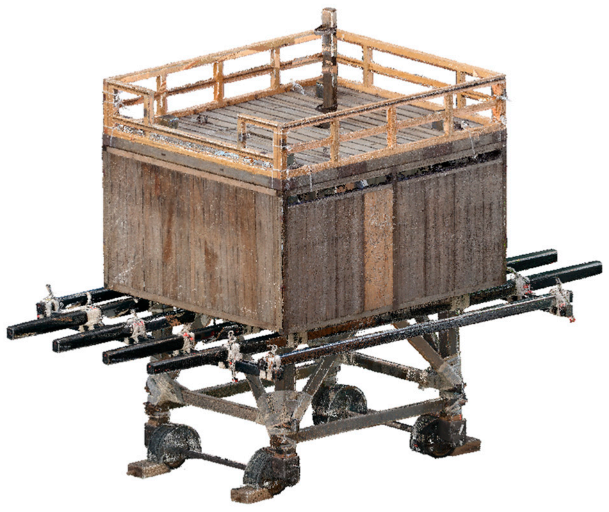

Figure 16. 3D-laser-scanned point cloud of the Hachiman-Yama float used in the Gion Festival in Kyoto City, Japan (7,866,197 points).

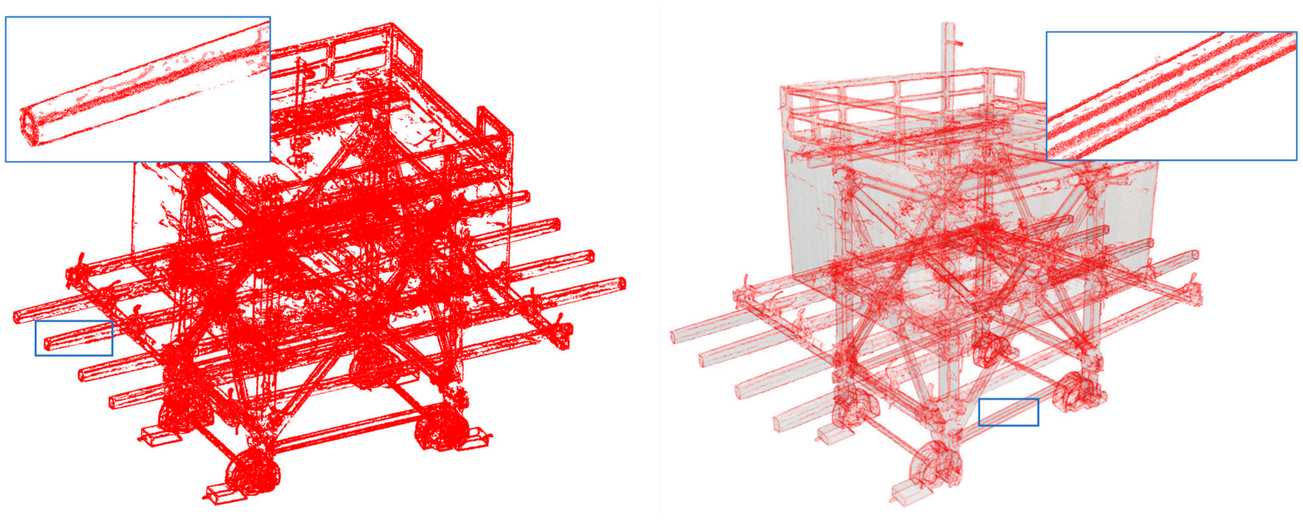
(**a**) (**b**)

Figure 17. (**a**) Edge points of the point cloud data in Figure 16. The set of edge points is created by extracting points by using linearity L_λ and adopting the type (b) function and then applying our upsampling network. (**b**) Edge-highlighted transparent visualization using the points of (a) and the original 3D-scanned points in Figure 16.

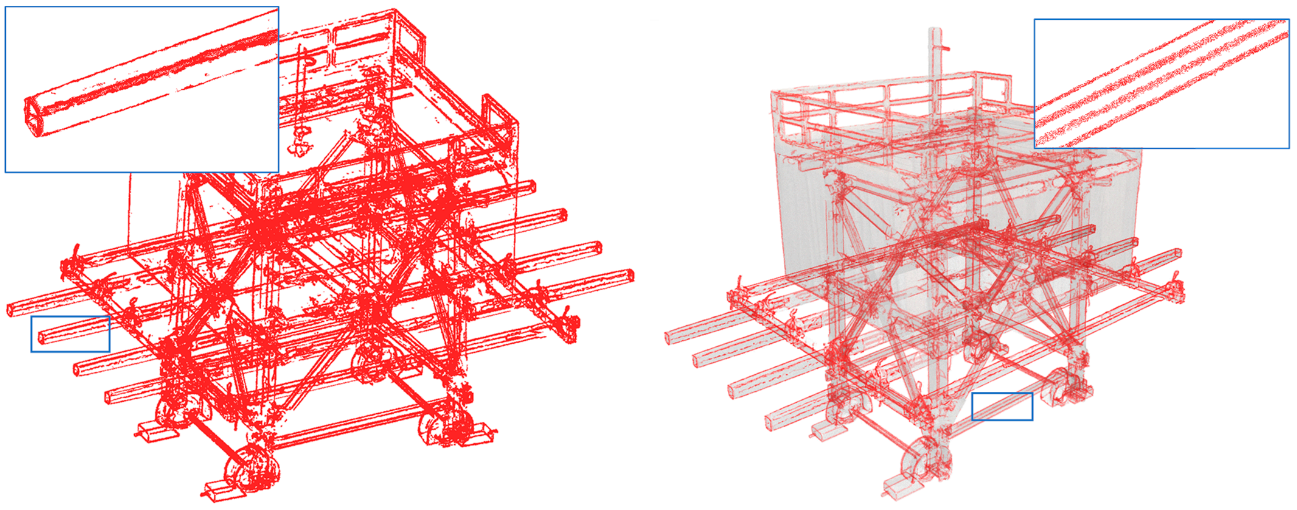
(**a**) (**b**)

Figure 18. (**a**) Edge points improved by the combined use of the SOR filter with our upsampling network. (**b**) Edge-highlighted transparent visualization using the points of (a) and the original 3D-scanned points in Figure 16.

In Figure 19, we compare the frequency distribution of the point number with respect to the feature value for the initially extracted edge points, upsampled edge points without using the SOR filter, and the upsampled edge points using the SOR filter. By using the SOR filter, the distributional shape of the upsampled points becomes similar to the initially extracted edge points. This means that the negative influence of the measurement noise on the upsampling is reduced successfully.

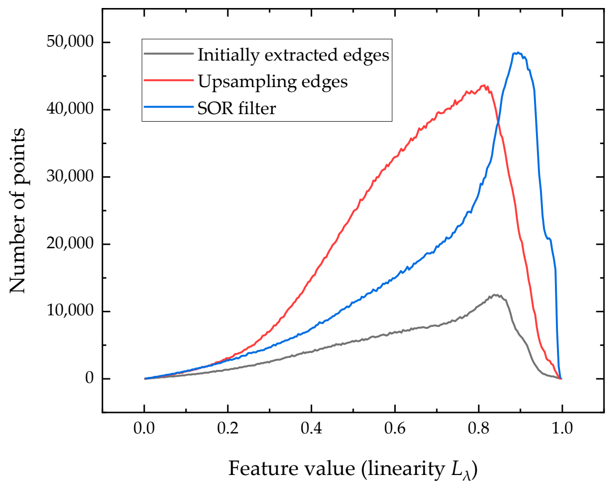

Figure 19. Frequency distribution of the point number with respect to the feature value (linearity L_λ) for the initially extracted edge points, upsampled edge points without using the SOR filter, and the upsampled edge points using the SOR filter.

Table 6 shows the experimental results of the quantitative evaluation of the effects of the SOR filter. The first data column compares the number of initially extracted edge points, upsampled edge points without applying the SOR filter, and upsampled edge points applying the SOR filter. By using the SOR filter, the number of edge points is reduced by 24 percent. However, the number is still sufficiently larger than the initially extracted edge points. The second data column compares the ratio of the edge points with $L_\lambda \geq 0.25$. We can see that upsampling with the SOR filter realizes the largest ratio. Although the increased ratio is only 2 percent compared with the initially extracted edge points, it reflects the elimination of the outlier points demonstrated in Figure 18. The third data column compares the average feature value at the edge points. We can see that the value is largest when using the SOR filter, which means that the real edge regions are most intensively extracted.

Table 6. Quantitative evaluation of the effects of the SOR filter.

	Number of Edge Points	**Ratio of Edge Points ($L_\lambda \geq 0.25$)**	**Average L_λ of the Edge Points**
Initially extracted edges	1,200,278	95.32%	0.64
Upsampling edges	4,801,112	95.67%	0.66
SOR filter	3,661,036	97.17%	0.73

5.5. Visibility Improvement of Soft Edges Using Our Upsampling Network

Soft or rounded edges often characterize the 3D structure of objects as well as sharp edges. As described in Section 3.2, the type (c) function is available for extracting soft edges. When using the type (c) function, edges are represented by the gradation of point density, which leads to the gradation of brightness in the visualized image. However, the gradient becomes blurred if the local point density is insufficient. In such a case, we should upsample the points on soft edges. Our proposed upsampling also works well for upsampling soft edges.

We extracted the soft edges for 3D-scanned data of the Japanese armor in Figure 20a. The surface of this armor has many unsharp concave and convex local areas, that is, soft edges. We extracted the soft edges using the type (c) function with change-of-curvature C_λ as the feature value f. Figure 20b shows the extracted edge points before upsampling. There are many local regions where the soft edges are not clearly visible; for example, see the enlarged image in the rectangle. The point density gradation does not express the local concavity and convexity well. This leads to unclear visualization of the soft edges. In contrast, Figure 20c shows the edge points increased by our upsampling network, which learned the soft-edge features beforehand. The enlarged image in the rectangle, which clips the same part as that in Figure 20b, shows the clearer point density gradation. This successfully makes the soft edges more visible.

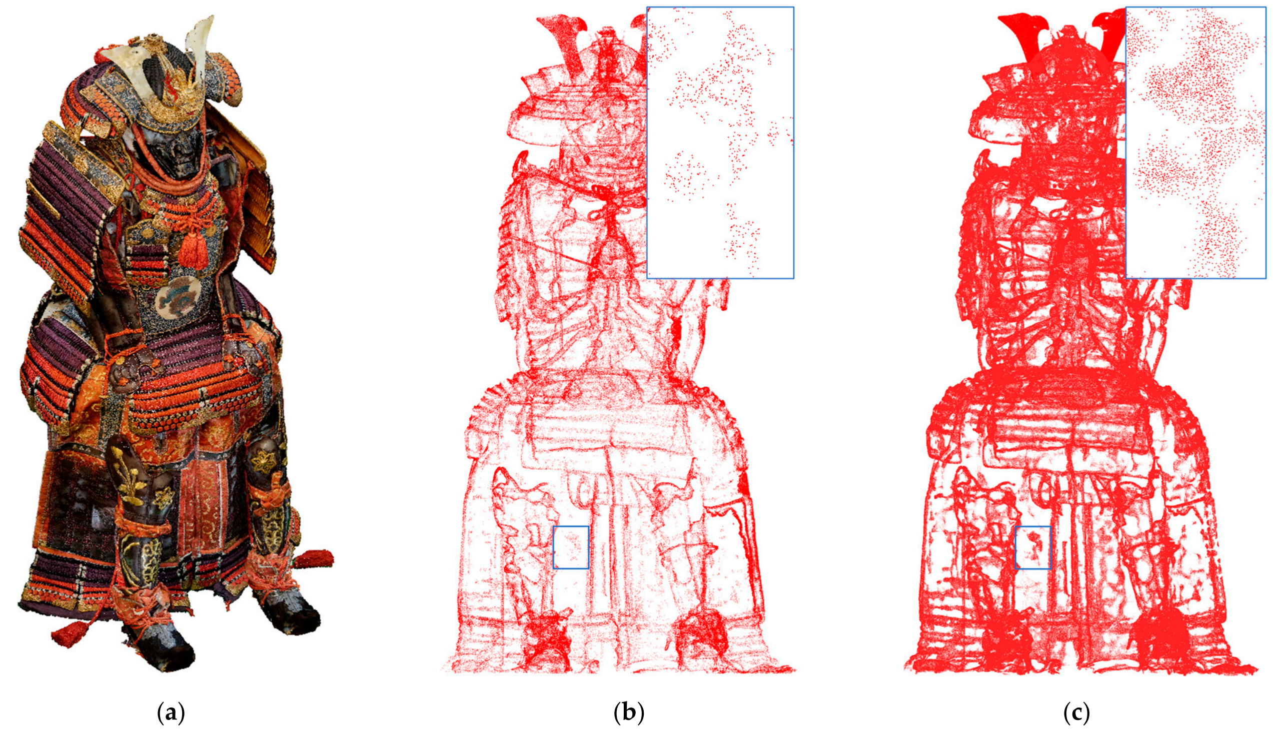

Figure 20. Upsampling experiment for 3D-scanned data with soft edges. The scanned target is the Japanese armor. Image (**a**) shows the 3D-scanned point cloud (9,094,466 points) acquired by our photogrammetric 3D scanning. Image (**b**) shows the extracted edge points (2,571,474 points) by adopting change-of-curvature and the type (c) function with $f_{\text{th}} = 0.03$, $F_{\text{th}} = 0.2$, $\alpha_{\text{min}} = 0.2$, $\alpha_{\text{max}} = 1.0$, and $d = 2.0$. Image (**c**) shows the soft-edge points increased by our upsampling network (10,285,896 points).

Figure 21 shows the results of the edge-highlighted transparent visualization. We fused the extracted edge points with the original 3D-scanned points in Figure 20a and applied SPBR for the integrated point dataset. Figure 21a shows the result of using the initially extracted edges, i.e., the edge points in Figure 20b, and Figure 21b shows the result of using the upsampled edge points in Figure 20c. The edge points in Figure 20b are multiplied by simple copying, resulting in the same number of points as the edge points in Figure 20c. (In SPBR, point multiplication based on simple point copying is allowed for increasing opacity [1].) By comparing Figure 21a and Figure 21b, we can see that the latter visualizes the soft edges more clearly. This improvement is due to the clear gradation of the point density realized by our proposed network. Thus, our upsampling network works effectively to improve the comprehensibility of the transparent visualization of an object with soft edges.

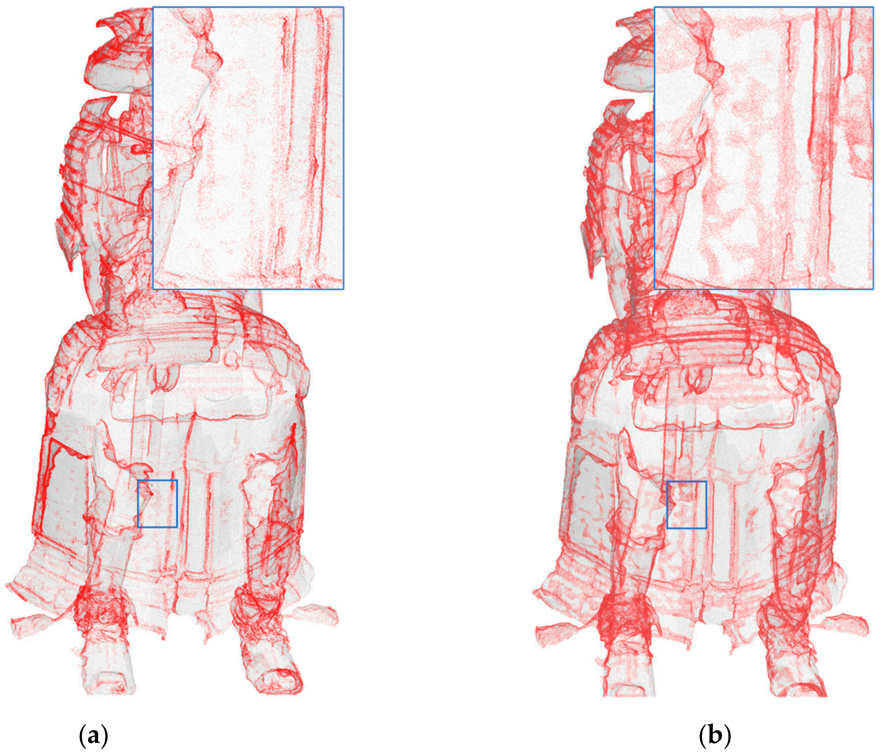

Figure 21. Edge highlighting by fusing extracted edge points with the 3D-scanned points in Figure 20a. Figure (**a**) shows the result of fusing the initially extracted edge points in Figure 20b, and Figure (**b**) shows the results of fusing the upsampled edge points in Figure 20c.

6. Discussion

We successfully demonstrate the effectiveness of the proposed upsampling method on the 3D edges of point cloud data. Applying the proposed method to the upsampling task on synthetic edges and scanned edges, we find that the GAN network-based upsampling method is effective in improving the visibility of edge regions in the point cloud. Our results in transparent visualization show that the edge regions are more visible compared with the conventional point cloud upsampling network.

Compared with our previous work [2], which uses 3D edges obtained by feature value extraction, the proposed deep learning-based edge upsampling method can learn the features of edges in the neural network and form more continuous and dense edges by expanding the features of edges. We also demonstrate in Section 5.3 that our method can enhance the visibility of 3D edges in both opaque and transparent visualization on real 3D-scanned data. Compared with the conventional upsampling networks of PU-NET [17] and PU-GAN [19], our method can focus more on the edge regions of point clouds and generate more edge points. In addition, our results in Section 5 also show that our method achieves better performance for large-scale 3D-scanned point clouds.

For the 3D-edge upsampling task, we consider that the ideal training data should consist of high-precision edges extracted from high-density scanned point clouds. However, it is challenging to collect a large training set of high-precision 3D-edge data at present because real 3D-scanned point clouds generally have a relatively large number of missing portions, causing extracted edges with low accuracy. Therefore, in our work, a patch-based training strategy is adopted to obtain the sparse input point cloud. As shown in Section 5, for real 3D-scanned point clouds, the extracted edges also generally contain many noise points in non-edge regions; thus, we consider that the sparse input point clouds, which are generated by randomly selecting points, can simulate the edges in real 3D-scanned data relatively well. In addition, by applying this approach to synthetic and real edges, we find that using the generated sparse point cloud in the training phase is effective; thus, we consider that the sparse point cloud generated by the polygon patches can effectively obtain the edge features of the original data.

7. Conclusions

In this paper, we proposed a deep learning-based upsampling method that is suitable for intensively increasing the number of edge points of 3D-scanned data. We applied the proposed methods to various real 3D-scanned data and demonstrated its effectiveness compared to the existing upsampling networks. Our edge upsampling network works well for both sharp and soft edges. We also showed that the creation of outlier points is significantly reduced by the combined use of the SOR filter. Thus, for a variety of 3D-scanned point cloud data, our upsampling network works well for increasing the point density in the edge regions. In addition, the network successfully intensively improves the point-distributional uniformity and connectivity in the edge regions. Therefore, it is possible to improve the visibility of the edge-highlighting transparent visualization realized by the recently proposed opacity-based edge highlighting [2].

There are still limitations to our method. The quality of the upsampled edges depends on the original 3D-scanned point cloud, especially the point distribution in its edge regions. For example, in cases where points are partially very sparse or largely lacking in the edge regions, our method cannot completely compensate for the insufficiency. In the future, we plan to investigate a multi-object training strategy that combines features from the original scanned point cloud with features from the edge data, which will encourage the network to better distinguish between edge and non-edge regions.

We believe the proposed method enables easier recognition of complex 3D-scanned data and contributes to dynamic/static visual analyses. We plan to apply our edge-highlighting transparent visualization method to the collision-warning visualization of dynamical/moving objects. 3D edges are likely to be involved in collisions, and thus, edge-highlighting visualization leads to efficient collision detection. For example, our recent visual collision simulation of a festival float and street obstacles [42] will become faster and more visible.

Author Contributions: Conceptualization, W.L., K.H., L.L. and S.T.; methodology, W.L., K.H., L.L. and S.T.; software, W.L.; validation, W.L.; formal analysis, W.L.; investigation, W.L.; resources, A.T.; data curation, W.L. and K.H.; writing—original draft preparation, W.L.; writing—review and editing, W.L., K.H., L.L. and S.T.; visualization, W.L.; supervision, L.L. and S.T.; project administration, S.T.; funding acquisition, S.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by JSPS KAKENHI Grant No. 19KK0256 and the AY2020 Program for Asia-Japan Research Development, Ritsumeikan University.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the Ritto History Museum, Y. Kitao of Kyoto Women's University, the Hachiman-Yama Preservation Society, the Tokushima Castle Museum, and Shrewd Design Co. Ltd. for their cooperation in executing the 3D scanning.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tanaka, S.; Hasegawa, K.; Okamoto, N.; Umegaki, R.; Wang, S.; Uemura, M.; Okamoto, A.; Koyamada, K. See-Through Imaging of Laser-Scanned 3D Cultural Heritage Objects Based on Stochastic Rendering of Large-Scale Point Clouds. In Proceedings of the ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, Prague, Czech Republic, 12–19 July 2016; Volume III-3.
2. Kawakami, K.; Hasegawa, K.; Li, L.; Nagata, H.; Adachi, M.; Yamaguchi, H.; Thufail, F.I.; Riyanto, S.; Tanaka, S.; Brahmantara. Opacity-based edge highlighting for transparent visualization of 3D scanned point clouds. *ISPRS Ann. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2020**, *5*, 373–380. [CrossRef]
3. Tanaka, S.; Hasegawa, K.; Shimokubo, Y.; Kaneko, T.; Kawamura, T.; Nakata, S.; Ojima, S.; Sakamoto, N.; Tanaka, H.T.; Koyamada, K. Particle-Based Transparent Rendering of Implicit Surfaces and its Application to Fused Visualization. In Proceedings of the Eurographics Conference on Visualization (EuroVis), Vienna, Austria, 5–8 June 2012.

4. Uchida, T.; Hasegawa, K.; Li, L.; Adachi, M.; Yamaguchi, H.; Thufail, F.I.; Riyanto, S.; Okamoto, A.; Tanaka, S. Noise-robust transparent visualization of large-scale point clouds acquired by laser scanning. *ISPRS J. Photogramm. Remote. Sens.* **2020**, *161*, 124–134. [CrossRef]
5. Alexa, M.; Behr, J.; Cohen-Or, D.; Fleishman, S.; Levin, D.; Silva, C. Computing and rendering point set surfaces. *IEEE Trans. Vis. Comput. Graph.* **2003**, *9*, 3–15. [CrossRef]
6. Lipman, Y.; Cohen-Or, D.; Levin, D.; Tal-Ezer, H. Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph.* **2007**, *26*, 22. [CrossRef]
7. Huang, H.; Li, D.; Zhang, H.; Ascher, U.; Cohen-Or, D. Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph.* **2009**, *28*, 1–7. [CrossRef]
8. Zhou, Y.; Tuzel, O. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 4490–4499.
9. Shi, S.; Wang, X.; Li, H. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2019; pp. 770–779.
10. Huang, Z.; Yu, Y.; Xu, J.; Ni, F.; Le, X. PF-Net: Point Fractal Network for 3D Point Cloud Completion. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2020; pp. 7659–7667.
11. Wen, X.; Li, T.; Han, Z.; Liu, Y.-S. Point Cloud Completion by Skip-Attention Network With Hierarchical Folding. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2020; pp. 1936–1945.
12. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. Pointnet: Deep Learning on Point Sets for 3d Classification and Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2017; pp. 652–660.
13. Zhang, Y.; Rabbat, M. A Graph-CNN for 3D Point Cloud Classification. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2018; pp. 6279–6283.
14. Liu, Y.; Fan, B.; Xiang, S.; Pan, C. Relation-Shape Convolutional Neural Network for Point Cloud Analysis. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2019; pp. 8887–8896.
15. Landrieu, L.; Simonovsky, M. Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2018; pp. 4558–4567.
16. Wang, W.; Yu, R.; Huang, Q.; Neumann, U. SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA; pp. 2569–2578.
17. Yu, L.; Li, X.; Fu, C.-W.; Cohen-Or, D.; Heng, P.A. PU-Net: Point Cloud Upsampling Network. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2018; pp. 2790–2799.
18. Yifan, W.; Wu, S.; Huang, H.; Cohen-Or, D.; Sorkine-Hornung, O. Patch-based Progressive 3D Point Set Upsampling. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–21 June 2019; pp. 5958–5967.
19. Li, R.; Li, X.; Fu, C.-W.; Cohen-Or, D.; Heng, P.-A. PU-GAN: A Point Cloud Upsampling Adversarial Network. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2019; pp. 7203–7212.
20. Chang, A.X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H. Shapenet: An information-rich 3d model repository. *arXiv* **2015**, arXiv:1512.03012 2015.
21. Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; Xiao, J. 3D ShapeNets: A deep representation for volumetric shapes. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2015; pp. 1912–1920.
22. West, K.F.; Webb, B.N.; Lersch, J.R.; Pothier, S.; Triscari, J.M.; Iverson, A.E. Context-Driven Automated Target Detection in 3D Data. In Proceedings of the Automatic Target Recognition XIV, Orlando, FL, USA, 21 September 2004.
23. Rusu, R.B. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *KI Künstliche Intell.* **2010**, *24*, 345–348. [CrossRef]
24. Weinmann, M.; Jutzi, B.; Mallet, C. Semantic 3D scene interpretation: A framework combining optimal neighborhood size selection with relevant features. *ISPRS Ann. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2014**, *II-3*, 181–188. [CrossRef]
25. Jutzi, B.; Gross, H. Nearest neighbour classification on laser point clouds to gain object structures from buildings. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2009**, *38*, 4–7.

26. Corsini, M.; Cignoni, P.; Scopigno, R. Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes. *IEEE Trans. Vis. Comput. Graph.* **2012**, *18*, 914–924. [CrossRef]
27. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Comput. Vis. Pattern Recognit.* **2017**, arXiv:1706.02413 2017.
28. Yu, L.; Li, X.; Fu, C.-W.; Cohen-Or, D.; Heng, P.A. EC-Net: An Edge-Aware Point Set Consolidation Network. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; Springer Science and Business Media LLC: Berlin/Heidelberg, Germany, 2018; pp. 398–414.
29. Chen, N.; Liu, L.; Cui, Z.; Chen, R.; Ceylan, D.; Tu, C.; Wang, W. Unsupervised Learning of Intrinsic Structural Representation Points. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2020; pp. 9118–9127.
30. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.* **2019**, *38*, 1–12. [CrossRef]
31. Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; Guibas, L. Learning Representations and Generative Models for 3d Point Clouds. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 40–49.
32. Yang, Y.; Feng, C.; Shen, Y.; Tian, D. FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2018; pp. 206–215.
33. Yuan, W.; Khot, T.; Held, D.; Mertz, C.; Hebert, M. Pcn: Point Completion Network. In Proceedings of the 2018 International Conference on 3D Vision (3DV), Verona, Italy, 5–8 September 2018; pp. 728–737.
34. Fan, H.; Su, H.; Guibas, L. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2017; pp. 2463–2471.
35. Mao, X.; Li, Q.; Xie, H.; Lau, R.Y.; Wang, Z.; Paul Smolley, S. Least Squares Generative Adversarial Networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2017; pp. 2794–2802.
36. Visionair. Available online: http://www.infra-visionair.eu (accessed on 21 March 2021).
37. Kingma, D.P.; Ba, J. Adam: A method for Stochastic Optimization. In Proceedings of the International Conference Learn Represent (ICLR), San Diego, CA, USA, 5–8 May 2015.
38. Berger, M.; Levine, J.A.; Nonato, L.G.; Taubin, G.; Silva, C.T. A benchmark for surface reconstruction. *ACM Trans. Graph.* **2013**, *32*, 1–17. [CrossRef]
39. Lague, D.; Brodu, N.; Leroux, J. Accurate 3D comparison of complex topography with terrestrial laser scanner: Application to the Rangitikei canyon (N-Z). *ISPRS J. Photogramm. Remote Sens.* **2013**, *82*, 10–26. [CrossRef]
40. ShapeNetCore. Available online: https://shapenet.org (accessed on 4 June 2021).
41. Rusu, R.B.; Cousins, S. 3d is Here: Point Cloud Library (pcl). In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 1–4.
42. Li, W.; Shigeta, K.; Hasegawa, K.; Li, L.; Yano, K.; Adachi, M.; Tanaka, S. Transparent Collision Visualization of Point Clouds Acquired by Laser Scanning. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 425. [CrossRef]