Point2color: 3D Point Cloud Colorization Using a Conditional Generative Network and Differentiable Rendering for Airborne LiDAR

Takayuki Shinohara[1], Haoyi Xiu[2], Masashi Matsuoka[3]
Tokyo Institute of Technology
Yokohama, Japan
{shinohara.t.af[1],xiu.h.aa[2],matsuoka.m.ab[3]}@m.titech.ac.jp

Abstract

Airborne LiDAR observations are very effective for providing accurate 3D point clouds, and archived data are becoming available to the public. In many cases, only geometric information is available in the published 3D point cloud observed by airborne LiDAR (airborne 3D point cloud), and geometric information alone is not readable. Thus, it is important to colorize airborne 3D point clouds to improve visual readability. A scheme for 3D point cloud colorization using a conditional generative adversarial network (cGAN) was proposed, but it is difficult to apply to airborne LiDAR because the method is for artificial CAD models. Since airborne 3D point clouds are spread over a wider area than simple CAD models, it is important to evaluate them spatially in two-dimensional (2D) images. Currently, the differentiable renderer is the most reliable method to bridge 3D and 2D images. In this paper, we propose an airborne 3D point cloud colorization scheme called point2color using cGAN with points and rendered images. To achieve airborne 3D point cloud colorization, we estimate the color of each point with PointNet++ and render the estimated colored airborne 3D point cloud into a 2D image with a differentiable renderer. The network is then trained by minimizing the distance between real color and colorized fake color. The experimental results demonstrate the effectiveness of point2color using the IEEE GRSS 2018 Data Fusion Contest dataset with lower error than previous studies. Furthermore, an ablation study demonstrates the effectiveness of using a cGAN pipeline and 2D images via a differentiable renderer. Our code will be available at GitHub.

1. Introduction

3D measurement by airborne LiDAR can acquire 3D point clouds quickly and accurately over a wide area and is widely used to create digital terrain maps (DTMs) [6] and digital surface maps (DSMs) [11]. The 3D point cloud

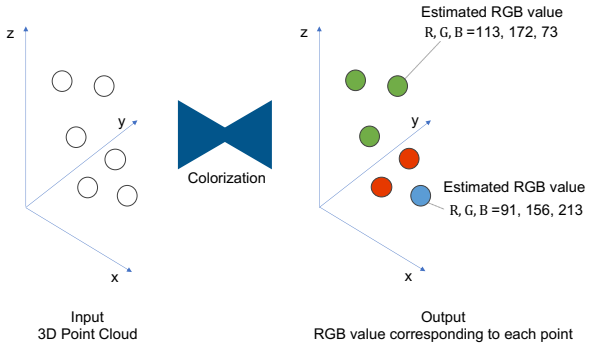

Figure 1. 3D point cloud colorization. The output is the color (RGB) corresponding to each point.

obtained by aircraft LiDAR (airborne 3D point cloud) can be used not only for 3D visualization but also for generating 3D maps by adding semantic information to each point. The process of assigning semantic information requires visual readability, and color information of the airborne 3D point cloud is necessary to improve the visual readability. Additionally, automatic analysis of 3D point clouds using deep learning methods such as PointNet [46] and PointNet++ [47] has been studied in recent years, and the color information of points has been reported to be effective in deep learning methods by benchmarks [15, 9]. However, airborne 3D point clouds that can be used for open data often do not have color information [1, 52, 56]. Therefore, to make effective use of the airborne 3D point cloud available in open data, we need to develop a point colorization method (Figure 1).

Image colorization has always been a research focus in the field of computer vision and computer graphics. At present, image colorization methods can be roughly divided into two categories: convolutional neural network (CNN)-based colorization [7, 25, 19] and conditional generative adversarial network (cGAN)-based colorization [23, 42, 3]. They both solve the same regression task of minimizing

the RGB value of each pixel estimated by the deep learning model and the true RGB value. CNN-based colorization [7], which solves only the regression task, has the disadvantage that it can intrinsically only estimate the gray average color. Therefore, an image colorization method using cGAN [42] was proposed. The cGAN allows for more advanced learning by trying to judge whether the fake data colored by the model or real data are given as ground truth. cGAN overcomes the disadvantage of CNN-based colorization. Some image colorization methods were applied to 3D point cloud colorization for a simple 3D CAD model [35, 4]. These 3D point cloud colorization methods use a PointNet [46]-based model with cGAN. Although PointNet is effective for simple 3D shapes, it is difficult to apply these previous studies to airborne 3D point clouds because it is difficult to apply PointNet to data containing various objects, such as airborne 3D point clouds. Comparing 3D point cloud colorization with image colorization, 3D point cloud colorization is a difficult task because images have grayscale hints, while points have only geometric information. Therefore, we need to devise a new method to color 3D point clouds. Since the placement of color in the airborne 3D point cloud viewed from above is important, it is necessary to train the model by projecting the airborne 3D point cloud in two dimensions as well as the points.

In recent years, many methods for projecting 3D data, such as meshes, voxels, and 3D point clouds, to 2D images using differentiable rendering have been studied in the field of computer vision. OpenDR [38] and neural mesh rendering (NMR) [28] compute conventional rasterization in the forward computation and approximate gradients in the backward computation. Recently, a soft rasterizer [36] and a differentiable interpolation-based renderer (DIB-R) [5] have proposed a differentiable renderer by viewing rasterization as a stochastic process in which the color of each pixel depends on multiple mesh faces. Differentiable ray tracing methods provide a more photorealistic image at the expense of increased computational complexity [32, 43]. Differentiable 3D point cloud rendering stores points in a voxel grid that is explored using ray termination probabilities and limits the resolution [20]. Differentiable rendering allows a deep learning model to render a colored 3D point cloud into an image, and errors in the loss function can be computed without interruption by backpropagation.

We propose an airborne 3D point cloud colorization schema (point2color) using conditional generative adversarial networks (cGANs) [40]-based pix2pix [23] and differentiable renderers. Point2color performs airborne 3D point cloud colorization using pairs of geometric information and its ground truth color. Point2color does not necessarily restore the original color information of the object to improve visibility but creates a reasonable color that conforms to the real world. Point2Color consists of a generator that performs colorization, a differentiable renderer that converts the colored points into a 2D image, and discriminators that judge whether the color from the generator is fake or a ground truth real color. Our generator respects PointNet++ [47], a renderer is based on a soft rasterizer [36], and the discriminators are a PointNet [47]-based model for points and a simple CNN model for rendered images. The major contributions of this research can be summarized as follows:

- We show the airborne 3D point cloud colorization schema (point2color) using a conditional generative adversarial network (cGAN) and differentiable rendering.
- The airborne 3D point cloud not only evaluates the color point-by-point but also trains to minimize the loss function for each image by projecting it in 2D with a differentiable renderer, given the characteristics of spatially distributed objects.
- Point2color obtains a lower MAE than previous studies on the GRSS 2018 Data Fusion Contest dataset.

2. Related Study

2.1. 3D Deep Learning

As a pioneering work that directly applies deep learning models to 3D point clouds, PointNet [46] employs MLP to learn the features of individual points and a symmetric function (e.g., max pooling) to extract global features. PointNet cannot capture the relationship between the local and global structure of the points. To solve this problem, PointNet++ [47] was developed by constructing a hierarchical neural network that applies convolutional operations via sampling layers and grouping layers. Following these two models, many researchers proposed various deep learning models for 3D point clouds based on PointNet-like architectures [34, 24, 49].

In the field of airborne 3D point clouds, recent studies have applied deep learning models. A 1D fully convolutional network was proposed to directly handle 3D coordinates and three corresponding spectral features extracted from 2D georeferenced images for each point [57]. Furthermore, a PointNet++-based deep neural network for airborne 3D point clouds was proposed [54]. In addition, some previous works explore dealing with graph structure [55], local and global information of points using attention [53] or context encoding [33].

2.2. Colorization

Traditional colorization methods often require human hints such as scribbles and reference images as guidance. These methods [18, 22, 39, 37, 14] mainly used handcrafted

features, including low-level handcrafted features or edges and high-level scene or location categories. The limitation of these works is that they do not have the generalization ability of images in different scenes.

Recently, deep learning-based methods have been utilized to address this problem. They are based on CNN to learn mapping from grayscale images to color images. The initial method used was a deep learning method that colors the image based on features extracted from different patches [7]. Recently, some conditional generative adversarial network (cGAN) [40]-based models have been proposed for image colorization. The cGAN is based on GAN [12] proposed to generate fake data in an unsupervised manner. The GAN contains a generator that learns to produce realistic fake data and a discriminator that judges whether data are fake data generated by the generator or real data sampled from the training data. The objective of GAN is training to minimize the Jensen–Shannon (JS) divergence between fake and real data. To stabilize convergence during GAN training, some advanced divergences between fake data and real data were proposed, such as Wasserstein distance [2] and Wasserstein distance with a gradient penalty [13]. The cGAN is a method of inputting conditional data to GANs. In the case of image colorization, a grayscale image is input as a condition, and a generator attempts to make a colored image. Compared to traditional colorization methods with CNN, the cGAN-based image colorization methods [42] minimize the various divergences between the colorized fake images and the real images in the ground truth, leading to a substantial improvement in the results. Additionally, pix2pix [23] proposed a cGAN-based pipeline to archive not only image colorization but also a general image-to-image translation problem. The experimental results demonstrated that the vividness of colorized images was enhanced due to GAN.

Moreover, two methods to achieve 3D point cloud colorization using deep learning-based methods were proposed [35, 4]. These methods are based on the pix2pix [23] pipeline using cGAN. The generator attempts to predict the color of each point using PointNet [46], and the PointNet-based discriminator attempts to judge fake color from the generator or real ground truth color. These methods can be used to colorize simple CAD models. However, these methods assume that only one simple 3D object, such as a table, chair, or teapot, is input. Therefore, it is difficult to apply these methods when various objects, such as roads, buildings, and vegetation, are mixed, as in the case of airborne 3D point clouds.

2.3. Differential Rendering

Currently, it is not a realistic option to prepare a large quantity of 3D ground truth data. Therefore, deep learning methods for 3D structures that use only 2D images to train a model are being explored [28, 27, 50, 48]. From the 3D structure output by the model, it is easy to generate a 2D image (called rendering) and calculate the difference between the generated image and the ground truth image. In this case, we assume that the camera parameters are known from which position and direction the ground truth image data are taken, and when rendering, we set a virtual camera similar to that position and direction to generate the image. The neural network of the prediction model is trained by backpropagation of the difference in the output image, but the error cannot be backpropagated unless the rendering is differentiable. For example, if the 3D structure is represented by a triangular mesh, the output image should change smoothly. For example, when a 3D structure represented by a triangular mesh is rendered using a rasterization technique, the pixel values change discontinuously depending on whether the mesh is placed in the center of the pixels. The output does not change smoothly in response to changes in the position of the mesh, so it is not differentiable, as in [28]. Thus, a differentiable renderer needs to be developed for 2D supervised learning. Recent deep learning methods show differentiable rendering for 3D data. In the case of mesh representation [28, 27], the most common approach is to assume some initial shape and to reconstruct the 3D structure by deforming it (e.g., by first assuming a sphere as the mesh structure and then continuously deforming it into the desired object). The disadvantage of this approach is that it is difficult to reconstruct objects with different topologies (e.g., holes, multiple objects). Another drawback of mesh representation is that it is not easy to handle data with complex structures consisting of vertices and faces in a deep learning framework. Another drawback of DRC [50], which uses voxel representation, is that the memory requirement increases with the cube of the resolution as the resolution increases. The previous works of [21, 30, 41] show the differentiable rendering method for 3D point clouds. These methods project reconstructed 3D point clouds using a differentiable renderer to obtain 2D images during supervision. In recent years, some libraries for differentiable renderers [26, 45, 51] have emerged that are capable of performing the differentiable rendering described above.

3. Proposed Method

To the best of our knowledge, our point2color is the first work on airborne 3D point cloud colorization. We use a conditional generative adversarial network (cGAN) to estimate the color of each point and the differentiable rendering to assist the colorization ability. The details of the point2color are described below.

3.1. Problem Statement

Figure 2 illustrates our colorization problem definition. Our model takes an airborne 3D point cloud $\boldsymbol{P} \in \mathbb{R}^{N\times 3}$ as input and estimates its missing color (fake color) $\boldsymbol{C}_{\text{fake}} \in \mathbb{R}^{N\times 3}$. First, we estimate fake color $\boldsymbol{C}_{\text{fake}}$ corresponding to an input set of points $\boldsymbol{P}$ using generator (G). Here, each $\boldsymbol{P}$ has geometric information (x, y, z), and $\boldsymbol{C}_{\text{fake}}$ has color information (RGB). Next, we feed $\boldsymbol{C}_{\text{fake}}$ into a differentiable renderer to project into a 2D fake image ($\boldsymbol{I}_{\text{fake}} \in \mathbb{R}^{H\times W\times 3}$). Finally, we employ two discriminators. The first is the point discriminator to judge fake color $\boldsymbol{C}_{\text{fake}}$ with $\boldsymbol{P}$ or real color ($\boldsymbol{C}_{\text{real}} \in \mathbb{R}^{N\times 3}$) with $\boldsymbol{P}$. The second is the image discriminator to judge fake image $\boldsymbol{I}_{\text{fake}}$ or real image ($\boldsymbol{I}_{\text{real}} \in \mathbb{R}^{H\times W\times 3}$).

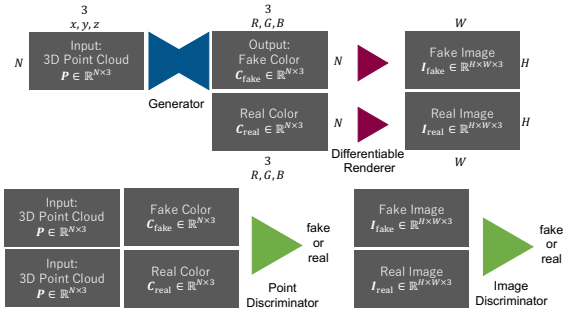

Figure 2. Problem statement of our point2color. Our generator estimates the fake color ($\boldsymbol{C}_{\text{fake}}$) of the input airborne 3D point cloud ($\boldsymbol{P}$). Then, $\boldsymbol{C}_{\text{fake}}$ are fed into a differentiable renderer to project 2D fake image $\boldsymbol{I}_{\text{fake}}$. Finally, two discriminators called a point discriminator and an image discriminator judge whether the image is fake or real.

3.2. Proposed Network

An overview of point2color is shown in Figure 3. Point2color is based on a cGAN that estimates the color corresponding to the input airborne 3D point cloud. It consists of three components: the main generator for colorization, a differentiable renderer, and two discriminators. All components are trained end-to-end. Each component is described in detail below.

3.2.1 Generator

To solve the problem of airborne 3D point cloud colorization, we use the generator to map the points with geometric coordinates ($\boldsymbol{P}$) into fake colors ($\boldsymbol{C}_{\text{fake}}$). Since it is necessary to obtain an output that corresponds one-to-one to the input $\boldsymbol{P}$, we use an autoencoder [17]-based network in which the input and output have a one-by-one correspondence. As an autoencoder-based architecture to deal with the 3D coordinates of input $\boldsymbol{P}$, we use the PointNet++ [47]-based model. Additionally, in the image processing field, some previous works [58] solved the image colorization problem using an autoencoder-based architecture.

First, we show the encoder that extracted features from the input airborne 3D point cloud ($\boldsymbol{P}$). In an encoder, the input is passed through a series of layers that progressively downsample until a bottleneck layer. In the downsampling layer, the representative point is determined by furthest point sampling (FPS), and the information of the points around the representative point is collapsed. By stacking these downsampling layers, the encoder can extract features hierarchically.

Next, we show the decoder that predicts the color of each point from extracted features from the encoder. The decoder performs upsampling three times on the features extracted by the encoder and adds color information corresponding to each input $\boldsymbol{P}$. Due to the limitation of downsampling in the encoder, much low-level or high-frequency information is lost. For many image colorization problems, a large amount of low-level information is shared between input and output, and it would be desirable to shuttle this information directly across the network. To bypass low-level information from the encoder into the decoder, we use a skip connection. It combines the upsampled features with the features assigned by skip connections and performs feature extraction using MLP on the combined features. In the above structure, the decoder estimates the fake color ($\boldsymbol{C}_{\text{fake}}$) corresponding to each point.

The generator network architecture is shown in Figure 3. The encoder architecture uses the coordinates (x, y, z) of input $\boldsymbol{P}$ and consists of three downsampling layers. The numbers of points for each downsample are 8,192, 4,046, and 2,028 from the shallow to deep layers. The convolutional filter size is 1×1, and the numbers of filters are 64, 128, and 256. The decoder architecture consists of three upsampling layers to the encoder. After the upsampling layers, the final layer uses the fully connected layer to estimate the color information (RGB) of each point. In this case, the RGB value is normalized in $[0, 1]$ by the sigmoid function.

3.2.2 Differentiable renderer

The 3D point cloud differentiable renderer first projects onto a 2D image under the given transformation matrix. A point with color is projected and added into an image. Although the projection process is not differentiable, we approximate derivatives using the subderivative. Therefore, we use the differentiable rendering method proposed in the soft renderer [36] to produce a fake image ($\boldsymbol{I}_{\text{fake}}$) and a real image ($\boldsymbol{I}_{\text{real}}$).

3.2.3 Discriminators

We use two discriminators to enforce the natural colorization result. The aim of discriminators is to determine

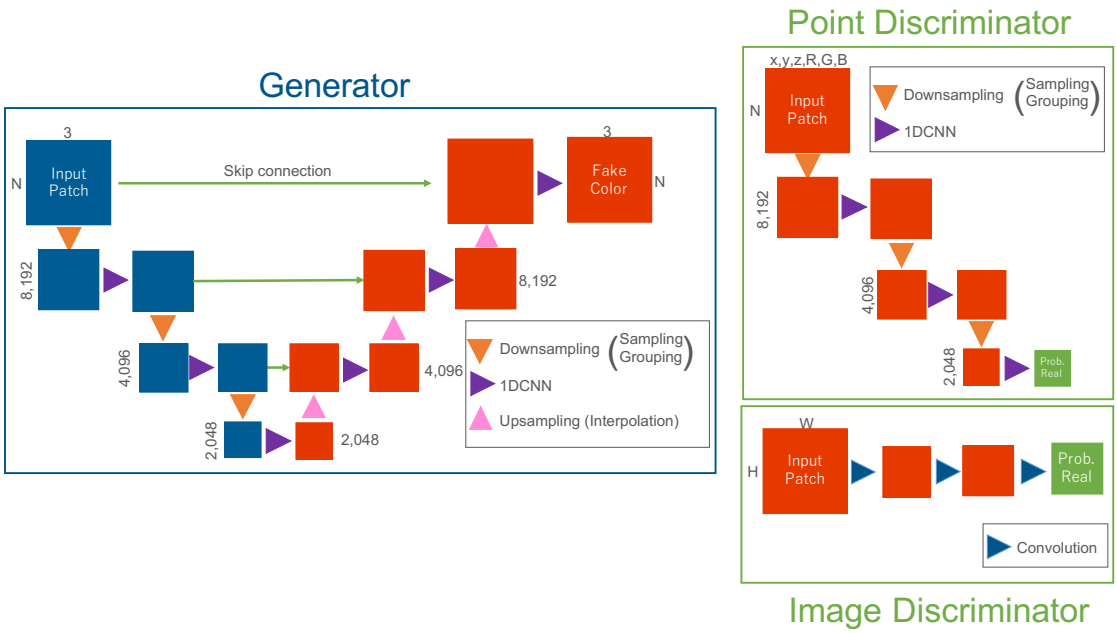

Figure 3. Network overview. Given an airborne 3D point cloud $\boldsymbol{P}$ as input, our model starts with color estimation of each point using a PointNet++ [47]-based generator. The generator has an encoder-decoder architecture that includes downsampling and convolution-based encoders and decoders estimating colors corresponding to input points. We then project the estimated fake color of the input 3D point cloud $\boldsymbol{C}_{\text{fake}}$ into fake image $\boldsymbol{I}_{\text{fake}}$ via a differentiable renderer. Finally, the point discriminator judges fake color $\boldsymbol{C}_{\text{fake}}$ or real color $\boldsymbol{C}_{\text{real}}$, and the image discriminator judges fake images $\boldsymbol{I}_{\text{fake}}$ or real images $\boldsymbol{I}_{\text{real}}$. The point discriminator is based on PatchGAN [23] with PointNet++ [47], and the image discriminator is based on PatchGAN [23] with a simple CNN. All networks train end-to-end.

whether the data is real (ground truth) or fake and colored by the generator. In the original GAN, the effect of the discriminator is defined as matching the real and fake probability distributions. As a result, the fake data from the generator can maintain the same color distribution as the real ground truth. We use two discriminators as shown as below.

Point Discriminator The point discriminator (D_{P}) network architecture is shown in Figure 3. The point discriminator uses the PointNet++ [47]-based PatchGAN architecture that judges the fake color by the generator $\boldsymbol{C}_{\text{fake}}$ and real color $\boldsymbol{C}_{\text{real}}$ by the ground truth. We use PointNet++ [47] with three downsampling layers and a fully connected layer to judge fake or real. The point discriminator evaluates three-dimensional color relationships.

Image Discriminator The image discriminator (D_{I}) network architecture is shown in Figure 3. The image discriminator uses the simple CNN-based PatchGAN [23] architecture. The image discriminator judges the colorized image ($\boldsymbol{I}_{\text{fake}}$) by the generator and ground truth color image ($\boldsymbol{I}_{\text{real}}$). Based on pix2pix [23], we use a 70×70 PatchGAN architecture. The image discriminator evaluates the placement of the objects as seen from above, such as aerial photos.

3.3. Loss Formulation

We use four loss functions: pointwise loss, pixelwise loss, point GAN loss, and image GAN loss.

Pointwise loss The pointwise loss emphasizes the pointwise fidelity of the estimated color by the generator. We use the L1 distance between fake color and real color defined as:

$$L_{\text{L1}}^{\text{point}} = \mathbb{E}[||\boldsymbol{C}_{\text{fake}} - \boldsymbol{C}_{\text{real}}||_1], \tag{1}$$

where $\boldsymbol{C}_{\text{fake}}$ and $\boldsymbol{C}_{\text{real}}$ represent the estimated fake color and ground truth real color.

Pixelwise loss The pixelwise loss emphasizes the rendered image fidelity of the estimated color by the generator. The loss uses the L1 distance between fake and real defined as:

$$L_{\text{L1}}^{\text{image}} = \mathbb{E}[||\boldsymbol{I}_{\text{fake}} - \boldsymbol{I}_{\text{real}}||_1], \tag{2}$$

where $\boldsymbol{I}_{\text{fake}}$ and $\boldsymbol{I}_{\text{real}}$ represent the rendered image from the estimated color by generator $\boldsymbol{C}_{\text{fake}}$ and the ground truth colorful rendered image from $\boldsymbol{C}_{\text{real}}$.

Point GAN loss We use a point discriminator ($D_{\text{P}}(*)$) that judges fake color ($\boldsymbol{C}_{\text{fake}}$) or real color ($\boldsymbol{C}_{\text{real}}$). As the loss function between fake color and real color, we use the Wasserstein distance [2] defined as:

$$L_G^{\text{point}} = -\mathbb{E}[D_{\text{P}}(\boldsymbol{C}_{\text{fake}})], \tag{3}$$

$$L_D^{\text{point}} = \mathbb{E}[D_{\text{P}}(\boldsymbol{C}_{\text{fake}})] - \mathbb{E}[D_{\text{P}}(\boldsymbol{C}_{\text{real}})]. \tag{4}$$

Image GAN loss We use an image discriminator ($D_{\text{I}}(*)$) that judges fake images ($\boldsymbol{I}_{\text{fake}}$) or real images ($\boldsymbol{C}_{\text{real}}$). As the loss function between a fake image and a real image, we use the Wasserstein distance [2], defined as:

$$L_G^{\text{image}} = -\mathbb{E}[D_{\text{I}}(\boldsymbol{I}_{\text{fake}}))], \tag{5}$$

$$L_D^{\text{image}} = \mathbb{E}[D_{\text{I}}(\boldsymbol{I}_{\text{fake}})] - \mathbb{E}[D_{\text{I}}(\boldsymbol{I}_{\text{real}})]. \tag{6}$$

Total loss The total loss function of the generator includes Equation 1, 2, 3, and 5, which is given by:

$$L_G = \lambda L_{\text{L1}}^{\text{point}} + \lambda L_{\text{L1}}^{\text{image}} + L_G^{\text{point}} + L_G^{\text{image}}. \tag{7}$$

Here, the λ is weight for L1 loss. Moreover, the total loss function of the discriminator includes Equations 4 and 6, which is given by:

$$L_D = L_D^{\text{point}} + L_D^{\text{image}}. \tag{8}$$

To preserve the Lipschitz continuity, we fix the weights of each discriminator to a small fixed range $[-0.01, 0.01]$ after every gradient update on the discriminators.

4. Experimental Result

4.1. Experimental Setup

The weight in total loss (λ) in Equation 7 was set to 100 in pix2pix [23]. In the training processes, we used the ADAM optimizer [29] with $\beta_1 = 0.99$ and $\beta_2 = 0.999$. During the training process, we set the rendering image size to 128×128. PyTorch [44] was used to implement the training process and image discriminator, PyTorch Geometric [10] was used to implement PointNet++-based a generator and a point discriminator, and PyTorch3D [26] was used to implement differentiable rendering for airborne 3D point clouds.

4.2. Dataset

We used the pair of airborne 3D point clouds and aerial photos of an outdoor scene called the 2018 IEEE GRSS Data Fusion Contest (DFC2018) [16] to conduct our experiments (Figure 4). DFC2018 contains a wide variety of objects, such as houses, ground, trees, roads, and an American football field (Figure 4 (a)). DFC2018 (Figure 4 (b)) was acquired by the National Center for Airborne Laser Mapping using an Optech Titan MW with an integrated camera, and the density was approximately 5 points/m^2. Since DFC2018 contains aerial photos of the same area as the airborne 3D point cloud, aerial photos were used to add color to each point to create the real ground truth color. PDAL [8] was used to create the real color of each point from the aerial photo. Additionally, the real color values of the points were normalized to 0-1. The original DFC2018 data were separated by hundreds of meters. Due to its large size, it had to be split into small patches to be computed by the GPU. In this study, we split the airborne 3D point cloud with real color into 30 m^2. The target for the actual loss calculation is the central 25 m^2 and the outside area was used to provide context information for colorization. In addition, because this airborne 3D point cloud contained noise, such as points under the ground, we removed isolated points as noise, and we subsampled points in each patch into $20,000$ using open3D [59]. We normalized the geometric value of patches to $[-0.5, 0.5]$. To train and evaluate the model, we divided all patch data into two datasets: the training data used to train our model and the test data used for evaluation. The test area is shown in the blue rectangle in Figure 4. Additionally, we used fivefold cross validation during the training process. All computations in this work were carried out by using the TSUBAME3.0 supercomputer at the Tokyo Institute of Technology.

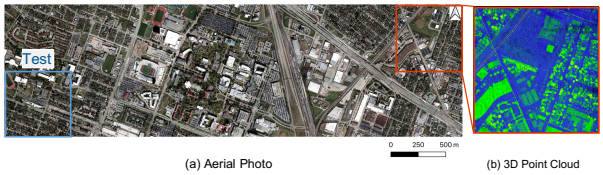

Figure 4. Experimental dataset used in this paper. We used an airborne 3D point cloud, published in the 2018 IEEE GRSS Data Fusion Contest [16]. The test area is indicated as a blue rectangle.

4.3. Point Colorization Result

Training the model on the generator and two discriminators took approximately three days on a TSUBAME3.0 with one NVIDIA P100 GPU until 100 epochs. Here, we showed quantitative and qualitative evaluations of the colorization results by the trained model.

4.3.1 Quantitative Evaluation

To measure the performance, we chose to employ pointwise mean absolute error (MAE) in RGB space normalized to 0-1. The MAE for the whole test data was computed by taking the mean of the absolute error of the fake color and real color on a point level for each color channel. The training results for point2color showed that the MAE was 0.10 in the test data.

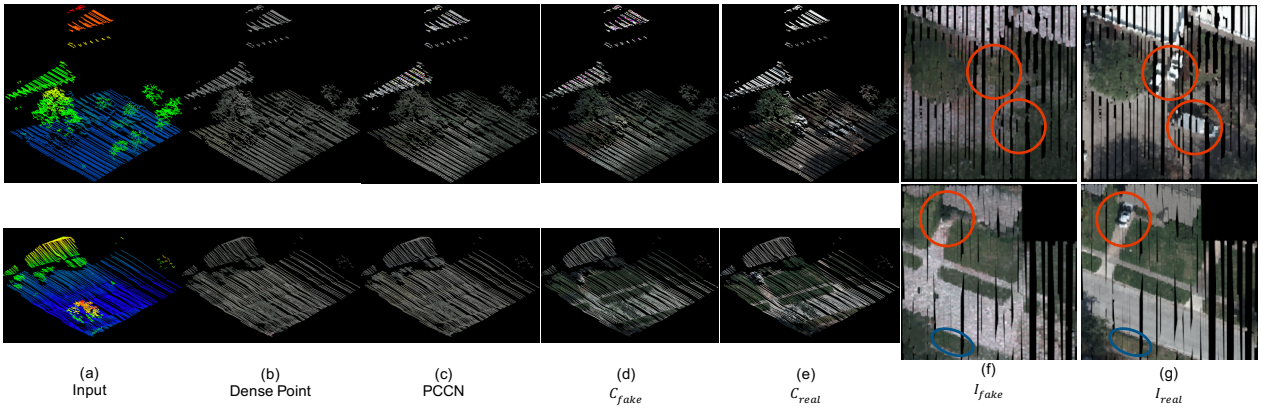

Figure 5. Visual results from the tasks of colorization example sampled from the test data. (a) Input airborne 3D point cloud colored by height. (b) Fake color with DensePoint [4]. (c) Fake color with PCCN [35]. (d) Fake color with point2color. (e) Ground truth (real color). (f) Rendered fake color with point2color. (g) Rendered real color. The circles indicate colorization failure cases.

Next, we compared to previous point colorization methods [4, 35] based on PointNet [46]-based cGAN. DensePoint [4] with vanilla PointNet showed that MAE is 0.25 in the test data. PCCN [35] with modified PointNet showed that MAE is 0.22 in the test data. Compared to these previous studies, point2color achieved a lower MAE. This is because the method of previous studies on a simple 3D object cannot handle complex objects observed by airborne LiDAR. In particular, the largest problem is that these methods do not combine local and global features effectively since PointNet is used as the generator for colorization in these methods.

4.3.2 Qualitative Evaluation

Some of the typical results using the DFC2018 dataset are shown in Figure 5. The colored points generated by point2color (Figure 5 (d)) and its rendered images (Figure 5 (f)) contained colors that were vibrant. Generally, the colored points had a tendency to nearly replicate the ground truth color (Figure 5 (e)). Additionally, the rendered images nearly replicated the ground truth rendered image (Figure 5 (g)). As shown in the lower figure, coloring was possible even with blank input. In addition, point2color distinguished and colored the grass from the road. Overall, point2color improved the legibility and made it easier to understand the objects.

Next, we compared the colorization results with previous point colorization methods [4, 35] with PointNet [46] for CAD data. Comparing densePoint [4] with point2color, point2color obtained more vivid colors. Comparing PCCN [35] with point2color, point2color also showed more vivid colors. Point2color using a PointNet++ [47]-based generator and a differentiable renderer was more effective for capturing large-scale 3D point clouds observed by airborne LiDAR than PointNet [46]-based methods [4, 35].

Point2color was estimated the relatively reasonable color of the input airborne 3D point cloud in the shown samples; however, there were some failure cases, shown as circles in Figure 5. We saw missing colors in cars, as shown by orange circles in Figure 5. This was because deep learning methods for 3D point clouds are sensitive to the size of the training data, so car-like small objects with a small number of points tended to be ignored. Additionally, another drawback was that point2color tended to colorize points in colors that are most frequently seen. For example, the bare ground was colored grass-like green, shown as a blue circle in Figure 5. This was most likely due to the significantly larger number of points with bare ground than points with grass. In the future, we need to develop new methods to enhance color variation and to deal with small objects while generating more realistic colors.

4.3.3 Ablation Study

To further demonstrate the effectiveness of point2color, we quantitatively analyzed different settings of point2color. Basically, we used four models to exclude some parts from the original structure.

Model 1: $L_{\mathrm{L1}}^{\mathrm{point}}$. We dropped the rendered images $L_{\mathrm{L1}}^{\mathrm{image}}$, $L_{D,G}^{\mathrm{point}}$, and $L_{D,G}^{\mathrm{point}}$ from point2color and its corresponding discriminators to analyze the ability of baseline pointwise regression.

Model 2: $L_{\mathrm{L1}}^{\mathrm{point}} + L_{\mathrm{L1}}^{\mathrm{image}}$. We added the image loss (L_{L1}^{image}) to Model 1 to analyze the effect of the rendered image.

Model 3: $L_{\mathrm{L1}}^{\mathrm{point}} + L_{D,G}^{\mathrm{point}}$. We added the point-based GAN loss ($L_{D,G}^{\mathrm{point}}$) to Model 1 and its corresponding point discriminator to analyze the effect of the GAN mechanism. This Model 3 replaces the PointNet-based cGAN methods

[4, 35] with PointNet++.

Model 4: $L_{\mathrm{L1}}^{\mathrm{point}} + L_{D,G}^{\mathrm{image}}$. We added the image-based GAN loss ($L_{D,G}^{\mathrm{image}}$) to Model 1 and its corresponding image discriminator to analyze the effect of the GAN mechanism for the rendered image.

Figure 6 (a)-(d) shows the results from ablation settings. First, if we used only pointwise L1 loss (Figure 6 (a)), the system tended to colorize all points with a dark green color. Consistent with the trend reported in pix2pix [23], L1 loss alone estimated average color. Second, if we used pointwise and pixelwise L1 loss (Figure 6 (b)), we were able to find white colored ground easier than Model 1. However, the roof points had a similar color to the trees. Third, if we used pointwise L1 loss and point GAN loss (Figure 6 (c)), the system tended to enhance the sharpness of the colorization result compared to Models 1 and 2 with regression task. Compared to the result of the cGAN with PointNet [4, 35] (Figure 5 (b), (c)), Method 3 using PointNet++ and cGAN resulted in more natural colors. The assumption that PointNet++ is effective in outdoor scenes has been shown to be reasonable. Finally, if we used pointwise L1 loss and image GAN loss (Figure 6 (d)), the system also tended to enhance the sharpness of the colorization result. Compared to Model 3, Model 4 was able to recognize the shade of a tree by using contextual information outside of the target. Generally, by adding each GAN loss function, natural colorization was possible. As shown in Figure 6 (e), the full (point2color) had the best performance compared with the four settings.

The quantitative analysis using MAE for the whole test data is summarized in Table 1. First, if image L1 loss and two discriminators were removed (Model 1), the system tended to generate samples with the highest MAE compared to the full model (point2color). Second, we also train the system with image and point L1 losses (Model 2) using the same data as Model 1. The MAE of Model 2 was higher than Model 1. Third, point GAN loss and the corresponding point discriminator (Model 3) promoted the baseline model to produce lower error than regression models alone (Models 1 and 2). Finally, image GAN loss and the corresponding image discriminator (Model 4) produced lower error than Models 1, 2, and 3. These effects of GAN were consistent with pix2pix [23] and GAN-based image colorization methods [31]. In conclusion, each component of point2color is indispensable, and the full model (point2color) achieved the lowest MAE.

5. Conclusion

In this study, we proposed a colorization schema called point2color for airborne 3D point clouds using a conditional generative adversarial network (cGAN). With the GRSS Data Fusion Contest 2018 dataset, point2color was able to consistently produce a colored airborne 3D point cloud with a lower mean absolute error than previous methods. In addition, qualitative evaluation showed that it was possible to estimate fake colors close to the real color. Moreover, the colorization result by the ablation model showed all of our proposed components.

However, point2color tends to generate color frequently in the training data. Additionally, the PointNet++-based generator tended to ignore small objects when colorized. We need to solve limitations of our method.

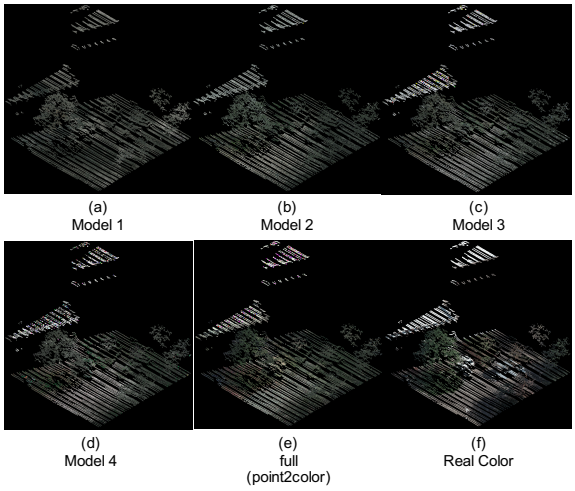

Figure 6. Comparison of colorization examples of different ablation study settings and full models (point2color). (a) Fake color with Model 1 ($L_{\mathrm{L1}}^{\mathrm{point}}$). (b) Fake color with Model 2 ($L_{\mathrm{L1}}^{\mathrm{point}}$ + $L_{\mathrm{L1}}^{\mathrm{image}}$). (c) Fake color with Model 3 ($L_{\mathrm{L1}}^{\mathrm{point}}$ + $L_{D,G}^{\mathrm{point}}$). (d) Fake color with Model 4 ($L_{\mathrm{L1}}^{\mathrm{point}}$ + $L_{D,G}^{\mathrm{image}}$). (e) Fake color with full model (point2color). (f) Real color.

Model	$L_{\mathrm{L1}}^{\mathrm{point}}$	$L_{\mathrm{L1}}^{\mathrm{image}}$	$L_{D,G}^{\mathrm{point}}$	$L_{D,G}^{\mathrm{image}}$	**MAE**
1	✓	-	-	-	0.23
2	✓	✓	-	-	0.18
3	✓	-	✓	-	0.15
4	✓	-	-	✓	0.12
full	✓	✓	✓	✓	0.10

Table 1. Ablation study performance of colorization result for test data. The MAE shown in the table is the average value for the whole test data.

Acknowledgment

We thank Data Fusion Contest 2018 for providing dataset. We used TSUBAME3.0 supercomputer at Tokyo Institute of Technology. This research was supported in a part by KAKENHI (19H02408) and "the Tokyo Metropolitan Resilience Project" of the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of the Japanese Government and the National Research Institute for Earth Science and Disaster Resilience (NIED).

References

[1] Het actueel hoogtebestand nederland 3(ahn3). https://downloads.pdok.nl/ahn3-downloadpage/. (Accessed on 02/17/2021). 1

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proc. ICML*, pages 214–223, 2017. 3, 5, 6

[3] M. G. Blanch, M. Mrak, A. F. Smeaton, and N. E. O'Connor. End-to-end conditional gan-based architectures for image colourisation. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2019. 1

[4] Xu Cao and Katashi Nagao. Point cloud colorization based on densely annotated 3d shape dataset. In *International Conference on Multimedia Modeling*, pages 436–446. Springer, 2019. 2, 3, 7, 8

[5] Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances In Neural Information Processing Systems*, 2019. 2

[6] Ziyue Chen, Bingbo Gao, and Bernard Devereux. State-of-the-art: Dtm generation using airborne lidar data. *Sensors*, 17(1):150, 2017. 1

[7] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proc. ICCV*, pages 415–423, 2015. 1, 2, 3

[8] PDAL Contributors. Pdal point data abstraction library, Nov. 2018. 6

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 1

[10] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 6

[11] Stephan Gehrke, Kristian Morin, Michael Downey, Nicolas Boehrer, and Thomas Fuchs. Semi-global matching: An alternative to lidar for dsm generation. In *Proceedings of the 2010 Canadian Geomatics Conference and Symposium of Commission I*, volume 2, 2010. 1

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NeurIPS*, pages 2672–2680, 2014. 3

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Proc. NeurIPS*, pages 5767–5777, 2017. 3

[14] Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin Ng, and Huang Zhiyong. Image colorization using similar images. In *Proc. ACM MM*, pages 369–378, 2012. 2

[15] Timo Hackel, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, pages 91–98, 2017. 1

[16] Saurabh Prasad; Bertrand Le Saux; Naoto Yokoya; Ronny Hansch. 2018 ieee grss data fusion challenge – fusion of multispectral lidar and hyperspectral data. 2020. 6

[17] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. 4

[18] Yi-Chin Huang, Yi-Shin Tung, Jun-Cheng Chen, Sung-Wen Wang, and Ja-Ling Wu. An adaptive edge detection based colorization algorithm and its applications. In *Proc. ACM MM*, pages 351–354, 2005. 2

[19] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Trans. on Graphics*, 35(4):110, 2016. 1

[20] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2

[21] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. *arXiv preprint arXiv:1810.09381*, 2018. 3

[22] Revital Ironi, Daniel Cohen-Or, and Dani Lischinski. Colorization by example. In *Proc. EGSR*, pages 201–210, 2005. 2

[23] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proc. CVPR*, pages 1125–1134, 2017. 1, 2, 3, 5, 6, 8

[24] Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018. 2

[25] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proc. ECCV*, pages 694–711, 2016. 1

[26] Justin Johnson, Nikhila Ravi, Jeremy Reizenstein, David Novotny, Shubham Tulsiani, Christoph Lassner, and Steve Branson. Accelerating 3d deep learning with pytorch3d. In *SIGGRAPH Asia 2020 Courses*, pages 1–1. 2019. 3, 6

[27] Hiroharu Kato and Tatsuya Harada. Learning view priors for single-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9778–9787, 2019. 3

[28] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3907–3916, 2018. 2, 3

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2014. 6

[30] K L Navaneet, Priyanka Mandikal, Varun Jampani, and Venkatesh Babu. Differ: Moving beyond 3d reconstruction with differentiable feature rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 18–24, 2019. 3

[31] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. In *Proc. CVPR*, pages 6874–6883, 2017. 8

[32] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018. 2

[33] Xiang Li, Lingjing Wang, Mingyang Wang, Congcong Wen, and Yi Fang. Dance-net: Density-aware convolution networks with context encoding for airborne lidar point cloud classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 166:128–139, 2020. 2

[34] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018. 2

[35] Jitao Liu, Songmin Dai, and Xiaoqiang Li. Pccn: Point cloud colorization network. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3716–3720. IEEE, 2019. 2, 3, 7, 8

[36] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717, 2019. 2, 4

[37] Xiaopei Liu, Liang Wan, Yingge Qu, Tien-Tsin Wong, Stephen Lin, Chi-Sing Leung, and Pheng-Ann Heng. Intrinsic colorization. *ACM Trans. on Graphics*, 27(5):1–9, 2008. 2

[38] Matthew M Loper and Michael J Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169. Springer, 2014. 2

[39] Qing Luan, Fang Wen, Daniel Cohen-Or, Lin Liang, Ying-Qing Xu, and Heung-Yeung Shum. Natural image colorization. In *Proc. EGSR*, pages 309–320, 2007. 2

[40] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2, 3

[41] KL Navaneet, Priyanka Mandikal, Mayank Agarwal, and R Venkatesh Babu. Capnet: Continuous approximation projection for 3d point cloud reconstruction using 2d supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8819–8826, 2019. 3

[42] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. Image colorization using generative adversarial networks. In *International conference on articulated motion and deformable objects*, pages 85–94. Springer, 2018. 1, 2, 3

[43] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Trans. Graph.*, 38(6), Nov. 2019. 2

[44] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[45] MS Prasad, KJ Reid, and HH Murray. Kaolin: processing, properties and applications. *Applied clay science*, 6(2):87–119, 1991. 3

[46] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 2, 3, 7

[47] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++ deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5105–5114, 2017. 1, 2, 4, 5, 7

[48] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019. 3

[49] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019. 2

[50] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017. 3

[51] Julien Valentin, Cem Keskin, Pavel Pidlypenskyi, Ameesh Makadia, Avneesh Sud, and Sofien Bouaziz. Tensorflow graphics: Computer graphics meets deep learning. 2019. 3

[52] N. Varney, V. K. Asari, and Q. Graehling. Dales: A large-scale aerial lidar data set for semantic segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 717–726, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society. 1

[53] Congcong Wen, Xiang Li, Xiaojing Yao, Ling Peng, and Tianhe Chi. Airborne lidar point cloud classification with global-local graph attention convolution neural network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 173:181–194, 2021. 2

[54] Lukas Winiwarter, Gottfried Mandlburger, Stefan Schmohl, and Norbert Pfeifer. Classification of als point clouds using end-to-end deep learning. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 87(3):75–90, 2019. 2

[55] H. Xiu, T. Shinohara, and M. Matsuoka. Dynamic-scale graph convolutional network for semantic segmentation of 3d point cloud. In *2019 IEEE International Symposium on Multimedia (ISM)*, pages 271–2717, 2019. 2

[56] Zhen Ye, Yusheng Xu, Rong Huang, Xiaohua Tong, Xin Li, Xiangfeng Liu, Kuifeng Luan, Ludwig Hoegner, and Uwe Stilla. Lasdu: A large-scale aerial lidar dataset for semantic labeling in dense urban areas. *ISPRS International Journal of Geo-Information*, 9(7):450, 2020. 1

[57] Mohammed Yousefhussien, David J Kelbe, Emmett J Ientilucci, and Carl Salvaggio. A fully convolutional network for semantic labeling of 3d point clouds. *arXiv preprint arXiv:1710.01408*, 2017. 2

[58] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Proc. ECCV*, pages 649–666, 2016. 4

[59] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 6