Wasserstein Generative Adversarial Network for Depth Completion with Anisotropic Diffusion Depth Enhancement

TRI MINH NGUYEN¹, MYUNGSIK YOO²
¹Department of Information Communication Convergence Technology, Soongsil University, Seoul, Korea (e-mail: trim@soongsil.ac.kr)
²School of Electronic Engineering, Soongsil University, Korea (e-mail: myoo@ssu.ac.kr)

ABSTRACT The objective of depth completion is to generate a dense depth map by upsampling a sparse one. However, irregular sparse patterns or the lack of groundtruth data caused by unstructured data make depth completion extremely challenging. Sensor fusion using both RGB and LIDAR sensors can help produce a more reliable context with higher accuracy. Compared with previous approaches, this method takes semantic segmentation images as additional input and develops an unsupervised loss function. Thus, when combined with supervised depth loss, the depth completion problem is considered as semi-supervised learning. We used an adapted Wasserstein Generative Adversarial Network architecture instead of applying the traditional autoencoder approach and post-processing process to preserve valid depth measurements received from the input and further enhance the depth value precision of the results. Our proposed method was evaluated on the KITTI depth completion benchmark, and its performance in depth completion was proven to be competitive.

INDEX TERMS Depth completion, LIDAR sparse depth, anisotropic diffusion, generative adversarial network, Wasserstein GAN

I. INTRODUCTION

DENSE depth maps are a fundamental feature for many vision functions installed in autonomous vehicles, such as 3D reconstruction, 3D object detection, and localization. LIDAR is a depth sensor that works on a similar principle to that of RADAR in terms of calculating the distance to an object by measuring the reflection of a signal from that object. However, instead of using radio waves, LIDAR uses an array of multiple laser beams irradiated at a high frequency, and it can rotate 360°when mounted on a vehicle. Despite its high accuracy and long-distance measurement, the depth data collected by the LIDAR sensor frequently appear to be of low resolution and are unstructured. This occurs because certain regions of a target object do not reflect the laser to the sensor, and thus, the depth data of these regions are invalid, or the regions are identified as "holes" in the depth data. Depth completion is aimed at producing an accurate dense depth map from sparse input LIDAR data.

In the last few years, deep learning methods, based on a variety of architectures, including convolutional neural networks (CNNs) and both supervised and unsupervised learning, have achieved promising results in computer vision and image processing tasks. Deep learning has also been applied to depth completion, achieving excellent results. For instance, Jaritz et al. [1] and Hue et al. [2] constructed two encoders to extract each modality’s features and one decoder for fusing them using the late fusion strategy to generate a dense depth map. Researchers who proposed methods such as FusionNet [3] and PENet [4] designed a two-branch autoencoder architecture network to extract the local and global structures of an input scene. Each branch can generate a semi-dense depth map, and they applied a multiple stage fusion strategy with learned confidence coefficients to produce the final depth map.

However, the aforementioned approaches address depth completion as supervised learning and require a significant amount of data with labels to sufficiently generalize the scene structure and achieve reliable results. This raises a non-trivial problem because groundtruth depth data are difficult to obtain in a realistic scenario, and only 30% of the depth
values in the groundtruth data are valid. Moreover, existing approaches [1] [2] [3] [4] consider only the RGB images corresponding to the sparse depth map as additional input, and, although RGB sensors are very sensitive to the lighting condition of the environment, the images that they capture can downgrade the accuracy of the model. Therefore, to address this problem, we treat the depth completion problem in a semi-supervised manner by combining both supervised and unsupervised learning. Furthermore, generative adversarial network (GAN)–based methods in image processing, such as those presented in [5] [6] [7] [8], have recently achieved outstanding results for both unsupervised and semi-supervised learning because GANs focus on learning the distribution of data rather than their numerical density value. GANs can generalize the data represented in a low-dimensional manifold.

In summary, the main contributions of our study are as follows:

- We successfully adapted Wasserstein GAN (WGAN) for solving the depth completion task as a semi-supervised learning problem.
- We designed a two-branch generator and effectively improved the results by taking semantic segmentation images as additional input, followed by a smoothness penalization loss.
- Finally, we enhanced the accuracy of the depth value by directly replacing the valid depth value obtained from sparse input; we also used an anisotropic diffusion process for the gradient-smoothing of the depth image.

II. RELATED WORK

In this section, several related works on depth completion are discussed. We focus on the handling of sparse depth images using a deep learning method with a single depth image or multiple modalities, in particular RGB images.

Sparse depth only. The problem of filling sparse data with the help of a groundtruth sample is well-known in the field of computer vision. Similar techniques, such as image inpainting, denoising, and super resolution, can be viewed as part of the depth completion problem, facilitating various solutions. The implementation of a bilateral filter or a similar interpolation technique has been used as a handcrafted strategy to upsample sparse input. Ku et al. realized a noteworthy achievement [9] using only these traditional methods. They successfully produced a dense output by analyzing the morphological structure of an input image and generating the final output from an intermediate representation. However, even in a reasonable result, the proportion of error in the LIDAR frame remains significant. Using the camera pose projection approach, LIDAR data can be projected onto the image plane and converted into semi-structured data; subsequently, a CNN can be applied. Jaritz et al. [10] and Ma et al. [11] utilized a multi-layer deep network and used additional mask encoding bits, "0" and "1", as valid and invalid data points, respectively.

Uhrig et al. [12] demonstrated that sparsity-invariant convolutions were capable of handling sparse data. They created a validity mask and preserved it on each layer of the deep network to allow normalized convolution operations, which was followed by a max-pooling operation. They also propagated a confidence mask using a second convolution for each layer to normalize and create a mask for the next layer. A similar method was proposed by Eldesokey et al. [13]. The authors of HMS-Net [14] developed a multi-variation of layers and proposed a new function for concatenating, upsampling, and including a sparse input map.

Multiple modalities. To date, the use of a combination of multiple modalities, such as RGB images or different data from different sensors, has led to significant performance improvement. Fusion strategies remain an open question for future studies. In our study, an RGB image was used as additional input to the network. We next list several recent studies on combining different modalities to enhance depth completion performance.

Schneider et al. [15] preserved the sharp boundary of an object in the depth output. They processed pixel-wise labeling to separate different objects and used a geodesic distance to maintain the edges. Ma et al. [16] [17] applied residual blocks (ResNet) to a deep network and used the RGB-D data as the input. Moreover, a self-supervised learning method, which requires additional temporal information, was chosen by Ma et al. [17] as the learning method. They introduced a two-branch network that can combine RGB data and sparse LIDAR frames in the same feature space to improve performance. Zhang et al. [18] proposed a different strategy for combining inputs: late fusion. They first attempted to estimate the surface normals by processing RGB data and subsequently combined the outcomes with the sparse depth data to obtain the final depth map. Similarly, in [19], surface normals were also used to enhance the depth completion result. In CFCNet [20], semantic correlation features between RGB and depth information were utilized. The authors of CG-Net [21] proposed combining the features of multiple modalities using a cross-guidance module.

Generative Adversarial Network. The development of GAN [22] and its variants opened a new door in the field of image processing and computer vision. In DCGAN [23], multiple convolutional layers were concatenated to create a deep neural network to extract image features efficiently. BigGANs [24], a modification of GANs that results in a large-scale model, are flexibly designed to implement a trade-off between variety and fidelity. GANs have also been applied to medical image processing [6], [25], semi-supervised classification [26], and semantic segmentation of remote sensing images [27] to improve the feature extraction capability of the network.

III. METHODOLOGY

In the first part, we provide an overview of our proposed method; next, we briefly discuss the difficulty involved in training the original GAN and the way WGAN compensates
for these difficulties. Subsequently, we introduce the details of our proposed architecture for depth completion and the loss function of the model.

In the second part, we describe our post-processing procedure.

### A. PROPOSED ARCHITECTURE

1) Architecture overview

The overall depth completion architecture proposed in this paper is shown in Fig. 1. The backbone is based on the WGAN architecture with a generator (G) and critic (C). The generator takes input from different sensors and is trained to generate a dense depth map. By contrast, the critic attempts to evaluate the realness of the generated sample. After training, the output of the generator passes through a post-processing stage for enhancing performance.

The architectural details are explained next.

2) From GAN to WGAN

GAN [22], first introduced by Goodfellow in 2014, consists of two different models called the generator and discriminator. The generator attempts to create a fake sample that is as similar as possible to the real sample. Meanwhile, the discriminator acts as a classifier to determine whether the sample is fake. As a result, the generator and discriminator play a minimax game to improve themselves. The loss function of a GAN is

$$\min_{G} \max_{D} \mathbb{E}_{x \sim p_r(x)} \log D(x) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D(G(z))),$$

(1)

where $x$ and $z$ are the real data with distribution $p_r(x)$ and the noise input with distribution $p_z(z)$, respectively; $G(\cdot)$ and $D(\cdot)$ represent the output of the generator and discriminator, respectively.

For GANs, the discriminator must be trained before training the generator. This results in a dilemma:

- If the discriminator is not appropriately trained, it will not provide any meaningful feedback to the generator.

Consequently, the loss function cannot reflect the errors from the generated sample.

- If the discriminator is well trained, the gradient from the loss function will be approximately zero, and thus, the learning will be excessively slow or even stop.

WGAN [28] is an version of the original GAN proposed in 2017 by Martin et al. to overcome the problems of GANs and make the training more stable. Instead of using binary cross entropy as a loss function, the authors proposed a new measurement of the distance between two probability distributions called the earth mover’s (EM) distance or Wasserstein distance. The EM distance is defined as the minimum energy required to transform the shape of one probability distribution to that of another distribution. As mentioned in [28], EM can provide a smoother gradient, which is essential for a stable learning process. The Wasserstein distance is

$$W(P_r, P_s) = \inf_{\delta \in \Pi(P_r, P_s)} \mathbb{E}_{(x,y) \sim \delta} || x - y ||.$$

(2)

where $P_r$ and $P_s$ represent the distributions of $x$ and $y$, which denote the real and generated samples, respectively. $\Pi(P_r, P_s)$ is the set of all possible joint distributions between $P_r$ and $P_s$, and $\delta$ describes the energy of one transformation to make the two distributions similar.

However, because all possible joint distributions of $P_r$ and $P_s$ are highly intractable, it is difficult to compute the inf (infimum) of the energy of the transportation plan. Based on the Kantorovich–Rubinstein duality [29], the author modified (2) to obtain a more sophisticated equation:

$$W(P_r, P_s) = \frac{1}{K} \sup_{\|f\|_K \leq K} \mathbb{E}_{x \sim P_r} f(x) - \mathbb{E}_{x \sim P_s} f(x),$$

(3)

where $\sup$ (supremum) is the maximum value and function $f(x)$ must satisfy $\|f\|_K \leq K$, which is k-Lipschitz continuous.

Suppose function $f$ is parameterized by a learnable parameter $\omega$: $f_\omega \in W$; then, the loss function for WGAN is

$$W(P_r, P_s) = \frac{1}{K} \sup_{\|f\|_K \leq K} \mathbb{E}_{x \sim P_r} f_\omega(x) - \mathbb{E}_{x \sim P_s} f_\omega(x).$$

(4)

In terms of the zero-sum game between the generator and discriminator, the loss function is changed to

$$L_{WGAN} = \min_{G} \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim P_r} [D(x)] - \mathbb{E}_{x \sim P_g} [D(G(x))],$$

(5)

where $\mathcal{D}$ is a set of 1-Lipschitz functions and $x$ is a consecutive sample from the distribution of the real sample $P_r$ and the distribution of the generated sample $P_g$, respectively.

In particular, the discriminator is trained to learn a k-Lipschitz function to compute the Wasserstein distance. As the loss function becomes smaller through training, the Wasserstein distance decreases, and the generator can generate a sample that is more similar to the real one. As mentioned in [28], the discriminator no longer discriminates between the fake and real samples; it outputs a number representing the realness and fakeness of a sample. Consequently, the discriminator was changed to the critic.
FIGURE 2. Overview of the generator architecture. It is composed of two branches having different purposes. The upper branch with the RGB input attempts to predict a depth map from the color image. The lower branch attempts to upsample the sparse depth map. (1)–(4) denote the concatenated function between the output of each decode layer in the RGB branch with the corresponding output of each encode layer in the sparse LIDAR branch.

3) Architecture details

Sensor fusion. Data collected from the LIDAR sensor not only retrieve accurate range information of objects surrounding a vehicle but are also unaffected by weather conditions; however, LIDAR data significantly lack the scene structure and object features required for visual recognition. By contrast, the RGB camera can easily capture the differences between shapes and colors, and a specific object can be identified based on these features; however, the camera is not an active sensor, and therefore its use is severely restricted to certain weather conditions. In brief, sensor fusion is a process aimed at overcoming the limitations of each sensor by fusing data from different levels to obtain more reliable and higher precision data. Ultimately, these fused data can be used for further processes in many different perception tasks.

Semantic segmentation. The semantic segmentation image of the scene is used as additional input to the model. Because the intensity of a pixel in an RGB image is highly sensitive to light conditions, the output depth map tends to be discontinued at the surface of the object when multiple fusion stages are performed with the RGB image. To compensate for this problem, we concatenated the semantic segmentation image with the sparse depth map. The semantic segmentation image is the output from DeepLabV3+ [32].

Generator. We designed an architecture similar to that of PENet [4], which has been proven to be highly efficient. However, as shown in Fig. 2, we reduced some of the layers in the encoder to prevent model overfitting. The architecture is composed of two branches having similar autoencoder models for each branch and skip connections [30] to prevent the vanishing gradient problem from occurring in the deep layer. The upper branch takes the RGB image as input and then attempts to predict the depth map and a confidence weight map. The lower branch takes the sparse depth image concatenated with the segmentation map obtained from the RGB data as input and then attempts to predict the depth map and its corresponding confidence weight map. Finally, these depth maps are fused together using a fusion strategy similar to that applied in FusionNet [3] to create a final dense depth map. We denote the depth map and confidence map predicted from the RGB image as $D_{rgb}$ and $C_{rgb}$, and those from the sparse depth image as $D_d$ and $C_d$, respectively. The final depth map $D$ is:

$$D(x, y) = e^{C_{rgb}(x, y)} \cdot D_{rgb}(x, y) + e^{C_d(x, y)} \cdot D_d(x, y) + e^{C_{rgb}(x, y)}$$

where $(x, y)$ denoted the pixel’s position.

In addition, when a traditional transpose convolution layer is used in the decoder, the output suffers heavily from the checkerboard artifact, as shown in Fig. 3. We address this problem by applying resize convolution layers [31] consisting of a nearest-neighbor upsampling layer followed by a convolution layer.

Critic. As mentioned previously, the critic of WGAN outputs a score or scalar that represents the fakeness or realness of the input. Therefore, we designed a critic, as shown in Fig. 4, that has an architecture similar to that of the discriminator of DCGAN [23] with certain modifications. In particular, we replaced the final sigmoid layer with a convolution layer to output a scalar. Moreover, we replaced the BatchNorm layer with an InstanceNorm layer because BatchNorm tends to normalize through batches, which is not suitable for our model, because weight penalization should be performed for each sample independently, not for the entire batch.

FIGURE 3. Traditional transpose convolutional layer causes heavy checkerboard artifact.
Supervised loss.

\[ L_{\text{depth}}(d, gt) = \|1_{\{gt > 0\}} \odot (d - gt)\|_1, \tag{7} \]

4) Loss function

**Supervised loss.** As in the study by Carvalho et al. [33], we used \( L_1 \) to minimize the depth loss between the output depth map and the groundtruth. The depth loss is

\[ L_{\text{depth}}(d, gt) = \|1_{\{gt > 0\}} \odot (d - gt)\|_1, \tag{7} \]

where \( d \) denotes the predicted depth map, \( gt \) denotes the groundtruth depth map and \( \|1_{\{gt > 0\}}\| \) denotes a valid depth pixel in the groundtruth data.

**Unsupervised loss.** Inspired by [34], we adopted a smoothness penalization loss:

\[ L_{\text{smooth}} = \frac{1}{N} \sum_{i=1}^{N} (\|\nabla^2_x d_i\|e^{-\|\nabla^2_x s_i\|} + \|\nabla^2_y d_i\|e^{-\|\nabla^2_y s_i\|}), \tag{8} \]

where \( \nabla^2_x \) and \( \nabla^2_y \) represent the second-order derivative in the \( x \) and \( y \) direction, respectively, \( d_i \) and \( s_i \) denote the \( i^{th} \) pixel of the predicted depth image and semantic segmentation image, respectively, and \( N \) is the total number of pixels.

As mentioned earlier, we used the semantic segmentation image to compensate for the depth discontinuity problem when fusing with the RGB image. The intensity of pixels clustered as the same object class in the semantic segmentation image will be equal, whereas two pixels belonging to two adjacent regions will have a significant difference in intensity. The exponential expression acted as a stop function to prevent over smoothing the actual edges detected by the semantic segmentation. Particularly, if the gradient of the pixel \( i^{th} \) in the semantic segmentation image is zero or close to zero, it may belong to the object’s surface, and the exponential expression \( e^{(-)} \) term will be close to one; therefore, the gradient of corresponding pixel \( i^{th} \) in the depth image will be reduced in the optimization process. By contrast, if the gradient of the pixel \( i^{th} \) in the semantic segmentation image is large, it may represent the boundary between an object and the background, which may have a significant difference in depth value; therefore, the exponential expression \( e^{(-)} \) term will be close to zero and the depth gradient of the corresponding pixel’s position \( i^{th} \) in the depth image will not be affected.

In summary, the critic attempts to maximize the following loss function:

\[ L_{\text{critic}} = L_{\text{WGAN}} \tag{9} \]

The generator attempts to minimize the overall loss:

\[ L_{\text{generator}} = L_{\text{WGAN}} + L_{\text{depth}} + L_{\text{smooth}}. \tag{10} \]

**B. POST PROCESSING**

1) Depth preservation

As argued in [35], the actual input depth measured by a LIDAR sensor cannot be preserved in the final output depth map because of the downsampling and upsampling steps in the learning phase. To compensate for this problem, the authors of [35] replaced the depth value from the original input directly in the output depth map and then performed a spatial propagation procedure to smooth the gradient of the depth value in neighboring pixels. However, according to our observation, it is not suitable to replace all the pixels because
the results mostly degrade after the replacement. As shown in Fig. 6, we analyzed the histogram of the depth error between the input sparse depth and the groundtruth in Figure 6(a) and between the prediction result and the groundtruth in Figure 6(b). The tail of the histogram of the depth error between the sparse input depth and the groundtruth is longer than the other, implying that the amount of depth error in the sparse input depth is more significant. As a result, the performance was downgraded when we replaced all pixels from sparse input depth to the prediction result.

Subsequently, we investigated the part of the input depth image that could be chosen to preserve the depth value. First, the input image was equally segmented into different ratios as different cases. Second, we separately preserved the depth input by replacing the pixel of the sparse input in each case to the predicted result and repeated with different samples from the dataset. Finally, we selected the case having the smallest average of root mean squared error (RMSE) of depth value. In addition, we included 1000 samples in our analysis.

- Case 1: We vertically divided into two equal parts.
- Case 2: We vertically divided into three equal parts.
- Case 3: We vertically divided into four equal parts.
In case three, because the portion of preserving the sparse input became too small, the mean of RMSE was only slightly changed only slightly.

After statistically analyzing three different cases, we decided only to replace only the output depth value with only the region comprising the lower one-third of the original input depth map. We argued that because the measurement in this region is closer to the sensor; therefore, the laser beams can a have better reflection to the sensor and suffer lesser noise of from the surrounding environment than the rest remaining upper part of the input depth image, which is far from the LIDAR sensor.

In the next section, we will introduce an image smoothing method and apply it to the resulting depth image because after the replacing step, the gradient value of the depth image becomes non-smooth and discontinuity.

2) Depth smoothing via anisotropic diffusion

Anisotropic diffusion [36] constitutes an adaptive filter aimed at smoothing the image target without causing significant loss of its content, such as its edges, lines, or other details that are crucial to image interpretation.

Image smoothing applies an adaptation of Newton’s law of heating. Therefore, based on the heat equation, Perona and Malik [36] modified it by adding a conductivity function \( c(x,y,t) \) to control the diffusion and prevent over-smoothing of the edge or the boundary of the target object. The equation of anisotropic diffusion for smoothing in a continuous domain is

\[
\frac{\partial I(x,y,t)}{\partial t} = \text{div}[c(x,y,t) \nabla I(x,y,t)]
\]

(11)

\[
= c(x,y,t) \cdot \Delta I + \nabla c \cdot \nabla I,
\]

(12)

where the initial condition at time \( t = 0 \) is given by

\[
I(x,y,0) = I_0(x,y).
\]

Here, \( \text{div} \) is the divergence operator, \( \Delta \) and \( \nabla \) denote the Laplacian operator and the gradient, respectively, \( t \) is the time parameter, \( I(x,y,0) \) is the original image, and \( I(x,y,t) \) is image \( I \) at time \( t \). If \( c(x,y,t) \) is a constant, Eq. (11) becomes the heat equation or isotropic diffusion.

To apply it to the discrete domain of the image, the authors discretized the anisotropic diffusion to:

\[
I_{p}^{t+1} = I_{p}^{t} + \lambda \sum_{q \in \eta_p} c(\nabla I_{p,q}^{t}) \nabla I_{p,q}^{t}.
\]

(14)

where \( p \) is the pixel position, \( I_{p}^{t} \) is the intensity of pixel \( I \) at iterator \( t \) or at time \( t \) as in the continuous domain, and \( \eta_p \) is the spatial neighborhood of pixel \( p \), as shown in Fig. 10. \( \lambda = \frac{1}{\eta_p} = 0.25 \) is the stable diffusion coefficient, and \( c(\cdot) \) represents diffusivity functions:

\[
c(\nabla I) = \begin{cases} 
\left(-\frac{\|
abla I\|}{K}\right)^2 & \nabla I \leq K \\
0 & \text{otherwise}
\end{cases}
\]

(15)

where \( K \) is the threshold parameter for the gradient magnitude controlling the rate of the diffusion process.

IV. EXPERIMENTS AND RESULTS

A. EXPERIMENT SETUP

1) Dataset and Metrics

For training and evaluation, we used the KITTI dataset and KITTI benchmark [37]. The dataset contains a total of 85898 training samples, 1000 samples for verification, and 1000 samples for testing. Moreover, all the LIDAR point cloud data were projected onto a 2D plane and synchronized with the corresponding RGB frame. When the data are projected to the image plane, the sparse depth image contained 5% valid pixels, and the groundtruth depth map contained 16% valid pixels. The size of each LIDAR frame and RGB frame was 1216 × 352, as this size was maintained in the training process.

We used four standard metrics in the comparative evaluation of model performance: root mean squared error (RMSE [mm]), root mean squared error of the inverse depth (1/RMSE [1/km]), mean absolute error (MAE [mm]), and mean absolute error of the inverse depth (1/MAE [1/km]).

2) Implementation details

Model configuration. We used the Pytorch [38] framework to implement the proposed architecture. We used the ADAM optimizer [39] for both the generator and critic, but with different hyperparameters: for the generator \( \beta_1 = 0.9, \beta_2 = 0.99 \), and the learning rate was 0.0004, whereas for the critic \( \beta_1 = 0.0, \beta_2 = 0.9 \), and the learning rate was 0.0001. Our model was trained for the first 20 epochs with loss defined as (9) and (10) and decreased by half for the subsequent epochs.

Post processing. Two parameters were defined for the anisotropic diffusion process. \( t \) and \( K \) in (14) and (15) were

| Region | Mean (RMSE) |
|--------|-------------|
| A      | 827         |
| B      | 787         |
| C      | 745         |
| D      | 741         |

TABLE 3. Mean RMSE of 1000 samples when replacing with different regions of input in case 3

FIGURE 10. Spatial neighbors of \( p \): they are the four pixels surrounding it.
TABLE 4. Comparison of different depth completion architectures on the KITTI Dataset

| Method                  | RMSE [mm] | MAE [mm] | iRMSE [1/km] | iMAE [1/km] | Run time [s] |
|-------------------------|-----------|----------|--------------|-------------|-------------|
| ScaffNet-FusionNet [40] | 1121.89   | 282.96   | 3.32         | 1.17        | 0.03        |
| Spade2D [41]            | 1035.29   | 248.32   | 2.60         | 0.98        | 0.04        |
| CSPN [35]               | 1019.64   | 279.64   | 2.93         | 1.15        | 1           |
| Sparse-to-Dense [16]    | 954.36    | 288.64   | 3.21         | 1.35        | 0.04        |
| NConv [13]              | 829.98    | 233.26   | 2.60         | 1.03        | 0.02        |
| PwP [42]                | 777.05    | 235.17   | 2.42         | 1.13        | 0.1         |
| FusionNet [3]           | 772.87    | 215.02   | 2.19         | 0.93        | 0.02        |
| NLSPN [43]              | 741.68    | 199.59   | 1.99         | 0.84        | 0.22        |
| GuideNet [44]           | 736.24    | 218.83   | 2.25         | 0.99        | 0.14        |
| PENet [3]               | 730.08    | 210.55   | 2.17         | 0.94        | 0.03        |
| Our                     | 725.43    | 209.42   | 2.26         | 0.95        | 0.09        |

the stopping criterion and the gradient threshold controlling diffusion rate, respectively. We conducted series of experiments on the KITTI Dataset to select the appropriate value for those parameters.

For the gradient threshold $K$, if $K$ was too large ($K \gg \text{pixel gradient } \nabla I$), the diffusion process can be over smooth, which made the result became blurry and lost all of the details; whereas for $K$ was too small ($K \ll \text{pixel gradient } \nabla I$), the diffusion process might become saturation in early iteration, which made the result appeared to be the same as the original. We plotted the histogram of depth gradient for 1000 thousand samples from testset and most of the pixels had the depth gradient distribution in the range 0 - 40 as shown in Fig. 11. We decided to select the value $K = 30$ and used it in our experiment.

Overestimating the stopping criteria $t$ can lead to the over-smoothing of the result, while underestimating can create visible noise artifacts. We used the root mean error (RMSE) metric to observe how each iteration affects performance. If the difference in the RMSE value is $\pm 1\%$ between two consecutive iterations $t$ and $t + 1$, the algorithm will stop at $t$ iteration. As shown in Fig. 12, the anisotropic diffusion process stopped at $7^{th}$ iteration.

B. RESULTS EVALUATION

Table 4 shows the comparison results on the KITTI benchmark of 10 widely used models presented in published or archived papers. According to our analysis, in the NConv [13] and Sparse-to-Dense [16] models, the details in the upper part of the output dense depth image are noisy and poor. ScaffNet [40] only used the synthetic data for training and evaluation, which might be not effective when evaluating with real scenario data. The results of the CSPN [35] and Sparse-to-Dense [16] models still include some regions that are incompletely filled. FusionNet [3] introduced a guidance map from the RGB image; however, it had been proved to be unnecessary and slightly downgraded the performance. In conclusion, our model generates a sample with high fine-grained density. Figure 13 shows a comparison of the results obtained by our model and the different depth completion models.

C. ABLATION STUDY

In an ablation study, we conducted a series of experiments to show the effectiveness of combining different losses and multiple inputs.

We compared models having the same architecture but different input and loss functions. Table 5 shows the performance of the variant architectures denoted as $V_1$, $V_2$, $V_3$, $V_4$, $V_5$ and $V_6$, and Fig. 14 shows the visual illustration results.

Multiple modalities. In $V_1$, we only use depth image LIDAR with depth supervised loss, the result shows many regions, and the upper part of the image cannot be properly filled because only small amount of pixels have the groundtruth value. Next, in $V_2$, we combined with RGB
image, the result showed more finer details and structure with better precision.

**Loss function.** We continued to explore the effectiveness of $\mathcal{L}_{\text{smooth}}$ by comparing $V_3$ and $V_5$. As the RGB image is highly sensitive to surrounding light, the performance of $V_3$ only increased slightly when training under two losses, and the result had discontinuity at some regions and the object’s boundary. In $V_5$, the semantic segmentation image was concatenated with the sparse input, and based on the RMSE, the combination of multiple inputs and losses can create a plausible result. Finally, $V_4$ and $V_6$ proved that the postprocessing stage with depth value replacement and anisotropic diffusion significantly increased performance in terms of RMSE.

| Denoted | Input | Loss function | Post processing | RMSE |
|---------|-------|---------------|-----------------|------|
| $V_1$   | Sparse | $\mathcal{L}_{\text{depth}}$ |               | 811.23 |
| $V_2$   | Sparse + RGB | $\mathcal{L}_{\text{depth}}$ |               | 756.01 |
| $V_3$   | Sparse + RGB | $\mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{smooth}}$ | ✓ | 754.23 |
| $V_4$   | Sparse + RGB | $\mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{smooth}}$ | ✓ | 731.85 |
| $V_5$   | Sparse + RGB + Segmentation | $\mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{smooth}}$ | ✓ | 743.66 |
| $V_6$   | Sparse + RGB + Segmentation | $\mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{smooth}}$ | ✓ | 725.43 |

**V. CONCLUSION**

We proposed an architecture using a WGAN that takes multiple inputs to generate a dense depth map. We formulated the problem in a semi-supervised manner and concatenated the semantic segmentation with the sparse depth LIDAR image to compensate for the RGB image’s sensitivity to lighting conditions in the surrounding environment. We further adapted the anisotropic diffusion in the postprocessing stage and achieved a competitive result. Moreover, our experimental results demonstrated the efficiency of the proposed architecture as compared with other approaches. In the future, we intend to investigate the means of enforcing the Lipschitz constant of the WGAN architecture and construct an edge stopping function in anisotropic diffusion through image statistical analysis.
VI. ACKNOWLEDGMENT

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education, Science and Technology (NRF-2021R1A2B5B01002559).

REFERENCES

[1] M. Jaritz et al., “Sparse and dense data with CNNs: Depth completion and semantic segmentation,” in Intl Conf. on 3D Vis. (3DV), vol. 2018, IEEE, 2018, pp. 52-60.

[2] J. Hua and X. Gongi, “A normalized convolutional neural network for guided sparse depth upsampling,,” in IJCAI, 2018, pp. 2283-2290.

[3] W. Van Gansbeke et al., “Sparse and Noisy LiDAR Completion with RGB Guidance and Uncertainty,” in 16th Intl Conf. on Mach. Vis. Appl. (MVA), 2019, pp. 1-6, doi: 10.23919/7939.8757939.

[4] M. Hu, S. Wang, B. Li et al., 2021, PENet: Towards precise and efficient image guided depth completion. arXiv preprint arXiv:2103.00783.

[5] , A. Brock et al., 2018, Large Scale GAN Training for High Fidelity Natural Image Synthesis. arXiv Preprint ArXiv:1809.11096.

[6] H. Salehinejad et al., “Generalization of deep neural networks for chest pathology classification in x-rays using generative adversarial networks,” in IEEE Intl Conf. on Acoust., Speech and Signal Process. (ICASSP),2018, pp. 990-994.

[7] G. Cheng et al., “Remote sensing image scene classification: Benchmark and state of the art.” Proc. IEEE, vol. 105, no. 10, 2017, pp. 1865-1883. doi: 10.1109/JPROC.2017.2675998.

[8] D. Lin et al., “MARTA GANs: Unsupervised representation learning for remote sensing image classification.” IEEE Geosci. Remote Sens. Lett., vol. 14, no. 11, pp. 2092-2096, 2017.

[9] J. Ku et al., “In defense of classical image processing; Fast depth completion on the CPU,” in 15th Conf. on Comput. and Robot Vis. (CRV), vol. 2018. IEEE, 2018, May, pp. 16-22.

[10] M. Jaritz et al., “Sparse and dense data with CNNs: Depth completion and semantic segmentation,” in Intl Conf. on 3D Vis. (3DV), vol. 2018, IEEE, 2018, pp. 52-60.

[11] F. Ma, S. Karaman, 2018, “Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image,” in IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018 pp. 4796-4803.

[12] J. Uhrig et al., “Sparsity invariant CNN,” in Intl Conf. on 3D Vis. (3DV),IEEE, 2017, pp. 11-20.

[13] , A. Eldesokey et al., 2018, Propagating Confidences through CNNs for Sparse Data Regression. arXiv Preprint ArXiv:1805.11913.

[14] Z. Huang et al., “Hms-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion.” IEEE Trans. Image Process., vol. 29, pp. 3429-3441, 2019.

[15] N. Schneider et al., “Semantically guided depth upsampling, in Ger. Conf. on Pattern Recognit.” Cham: Springer, 2016, pp. 37-48.

[16] F. Ma S. Karaman, “Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image.” in IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 4796-4803.

[17] F. Ma et al., “Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera,” in Intl Conf. on Robot. and Autom. (ICRA), vol. 2019. IEEE, 2019, pp. 3288-3295.

[18] Y. Zhang T. Funkhouser, “Deep depth completion of a single rgb-d image,” in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2018. pp. 175-185.

[19] J. Qiu et al., 2019, Deeplidar, “Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image,” in Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition, pp. 3313-3322.

[20] , Y. Zhong et al., 2019, Deep rgb-d Canonical Correlation Analysis for Sparse Depth Completion. arXiv Preprint ArXiv:1906.08967.

[21] Z. Zhang Y. Pang, “CGNet: cross-guidance network for semantic segmentation,” Science China Information Sciences, vol. 63, no. 2, pp. 1-16, 2020.

[22] I. Goodfellow et al., “Generative adversarial nets,” Adv. Neural Inf. Process. Syst., vol. 27, 2014.

[23] , A. Radford et al., 2015, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv Preprint ArXiv:1511.06434.

[24] , A. Brock et al., 2018, Large Scale GAN Training for High Fidelity Natural Image Synthesis. arXiv Preprint ArXiv:1809.11096.

[25] N. A. Tenenholtz et al., “Medical image synthesis for data augmentation and anonymization using generative adversarial networks,” in Intl Workshop on Simul. and Synth. in Med. Imaging. Cham: Springer, 2018, pp. 1-11.

[26] T. Salimans et al., “Improved techniques for training GANs.” Adv. Neural Inf. Process. Syst., vol. 29, pp. 2234-2242, 2016.

[27] Y. Tao et al., “GAN-assisted two-stream neural network for high-resolution remote sensing image classification. Remote Sens.,” vol. 9, no. 12, p. 1328, 2017.

[28] M. Arjovsky, S. Chintala, L. Bottou, “Wasserstein generative adversarial networks,” in Intl Conf. on Mach. Learn., PMLR, 2017, pp. 214-223.
[29] C. Villani, 2009, Optimal Transport: Old and New, vol. 338. Berlin: Springer.
[30] G. Huang et al., “Densely connected convolutional networks,” in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2017, pp. 4700-4708.
[31] A. Odena et al., “Deconvolution and checkerboard artifacts.” Distill, vol. 1, no. 10, p. e3, 2016.
[32] L. C. Chen et al., “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in Proc. Eur. Conf. on Computer Vision (ECCV), 2018, pp. 801-818.
[33] M. Carvalho et al., “On Regression Losses for Deep Depth Estimation,” in 25th IEEE Intl Conf. on Image Process. (ICIP), IEEE, 2018, Oct., pp. 2915-2919.
[34] C. Zhao et al., “Masked GAN for Unsupervised Depth and Pose Prediction with Scale Consistency,” IEEE Trans. Neur. Net. Learn., 2020.
[35] X. Cheng et al., “Learning depth with a convolutional spatial propagation network,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 10, pp. 2361-2379, 2019.
[36] P. Perona J. Malik, “Scale-space and edge detection using anisotropic diffusion.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 7, pp. 629-639, 1990.
[37] A. Geiger et al., “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in IEEE Conf. on Comput. Vis. and Pattern Recognit., IEEE, 2012, pp. 3354-3361.
[38] A. Paszke et al., “Pytorch: An imperative style, high-performance deep learning library,” Adv. Neural Inf. Process. Syst., vol. 32, pp. 8026-8037, 2019.
[39] D. P. Kingma and J. Ba, 2014, Adam: A Method for Stochastic Optimization. arXiv Preprint ArXiv:1412.6980.
[40] A. Wong et al., “Learning topology from synthetic data for unsupervised depth completion,” IEEE Robot. Autom. Lett., vol. 6, no. 2, pp. 1495-1502, 2021.
[41] M. Jaritz et al., “Sparse and dense data with CNNs: Depth completion and semantic segmentation,” in Intl Conf. on 3D Vis. (3DV), IEEE, 2018, pp. 52-60.
[42] Y. Xu et al. “Depth completion from sparse lidar data with depth-normal constraints,” in Proc. IEEE/CVF Intl. Conf. on Computer Vision, 2019, pp. 2811-2820.
[43] J. Park et al. “Non-local spatial propagation network for depth completion,” in Computer Vision–ECCV 2020:16th European Conference, Glasgow, UK, August 23–28, 2020, Proc. Part XIII 16 (pp. 120-136). Springer International Publishing
[44] J. Tang et al., “Learning guided convolutional network for depth completion,” IEEE Transactions on Image Processing, vol. 30, pp. 1116-1129, 2020.

TRI MINH NGUYEN received the B.Eng. degree in computer engineering from the University of Information Technology, Vietnam National University—Ho Chi Minh City, Ho Chi Minh City, Vietnam, in 2019. He is currently pursuing the master’s degree with Soongsil University. His research interests include Visible Light Communication (VLC).

MYUNGSIK YOO received the B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, South Korea, in 1989 and 1991, respectively, and the Ph.D. degree in electrical engineering from The State University of New York at Buffalo, New York, in 2000. He was a senior research engineer at Nokia Research Center, Burlington, MA. He is currently a full-time professor at School of Electronic Engineering, Soongsil University, Seoul. His research interests include visible light communications, cloud computing, and Internet protocols.

---

VOLUME 4, 2016