Article

Gradient-Applied Weighted Loss for Details of 3D Shape in Single-View Reconstruction

Jiho Lee[1], **Taehyeon Kim**[2], **Gihwan Lee**[3], **and Yoonsik Choe**[3]*

[1] Hyundai Motor Namyang Research Center, Hyundai Motor Co. Inc., Hwaseong 18280, Korea; jiholee526@hyundai.com (J.L.)
[2] Contents Convergence Research Center, Korea Electronics Technology Institute, Seoul 03924, Korea; taehyeon.kim@keti.re.kr (T.K.)
[3] Department of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, Korea; lgh1101@yonsei.ac.kr (G.L.), yschoe@yonsei.ac.kr (Y.C.)
* Correspondence: yschoe@yonsei.ac.kr

Abstract: There has been considerable research on reconstructing 3D shapes from single-view images; however, preserving the detailed information of the input image remains difficult. In this paper, we propose the application of a gradient map to train a network, aimed at improving the visual quality of fine-grained details such as the thin and tiny components of generated shapes. Each gradient map was created from the original voxel data, and each value represented the amount of information per volume. Here, the gradient map was defined by several methods that mathematically quantify and represent the detailed structure of an object. By applying this map to the loss function in training, we could induce the network to intensively train partial details, such as thin and narrow parts. We demonstrated that the detailed information was well-recovered when a weight that is proportional to the gradient value was applied to the loss. Furthermore, it is expected that our method will contribute to the development of 3D technologies related to the construction of virtual space for simulation and new customer experience.

Keywords: 3D shapes; Tingle-viewreconstruction; gradient map; fine-grained

1. Introduction

In the past, many studies on deep neural networks for 3D data have emerged, and they had the potential to be applied to many fields such as 3D geometry analysis [1–3], 3D synthesis [4–9], and 3D-assisted image analysis [10]. Unlike 2D images, 3D representations, such as voxel grids [11–15], point clouds [8], polygonal meshes [16–19], and multi-view images [1,20,21], are diverse; thus, many methods have been proposed to handle each representation. However, most of these representations are not compatible with the existing neural networks for 1D and 2D data and have limitations in terms of resolution. To address these limitations, studies on the process of implicit representation have been conducted. Implicit representation indicates whether it is inside or outside the surface (occupancy probability [22]) or its distance from the surface (signed distance function, SDF [23,24]) for each coordinate. This creates a binary classification problem for each point.

In this paper, we focus on the single-view 3D reconstruction (SVR) problem among 3D networks, which involves the generation of 3D shapes from a single image. Single-view reconstruction is a difficult problem because it requires learning the geometrical relationship between images and objects, and a single-view image has insufficient information for 3D, such as depth or unseen shape. Despite many advances [25],[26],[27], 3D shapes generated by these networks are still visually unsatisfactory. They generate overall shape to some degree but not in detail, because they extract one-dimensional feature vectors from 2D single images, with loss of much information. For example, IM-NET [26], a state-of-the-art work, which uses an implicit field to improve the visual

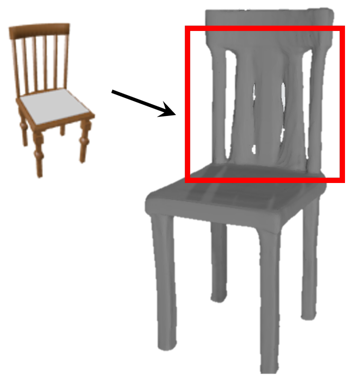

Figure 1. The input(left)[28] and output(right) of IM-NET[26]

quality of the result, was proposed. However, as shown in Figure 1, it is vulnerable to fine-grained details such as tiny holes and thin rods in objects and can easily fail to produce them when reconstructing 3D shapes. Moreover, DISN [27] attempts to resolve this problem and construct a better shape, but it still falls far short of the desired result. Furthermore, it takes too much time to train just the camera pose estimation, even longer than following signal distance function (SDF) prediction. This is one of the major issues in 3D deep neural networks that demands an inordinate amount of time for learning and generation because of the large amount of data.

To address the aforementioned problems, we introduced a method to improve the ability to capture the details of an object, without modification by complicating the network. In training, we weighed loss differently according to the spatial gradient value; thus, the larger the value, the larger the penalty applied. The algorithm was based on the assumption that fine-grained regions have a larger gradient value than other smooth regions. Note that we would simply extract the gradient values from the training data and revise only the loss function. Subsequently, the learning progressed at different levels in each part of the shape, which provided the network more flexibility in training the parameters. In other words, the neural networks could be made to train not only global shapes but also fine-grained information. We then determined how to define methods that mathematically quantify the gradient and reflect the topological details well. Once the spatial gradient data was prepared, we applied it to the loss function along with its original dataset. This enabled us to generate improved results without any time-consuming additional training.

In essence, the main contributions of this study are as follows:

- If local information is used as additional input data, the same level of the performance can be achieved as the result of using high-resolution training data, even with low-resolution. In the case of a network that uses 3D data as a training data, the computation and training time increase sharply as the resolution rises. Therefore, this advantage brings great **computational efficiency** to learning the generative network.
- Since gradient map as the local information is used in learning the network, it is possible **to enhance the implicit representation power**. When the 3D topology is represented by an implicit field, the location information of each point is represented by a function value. Here, the detailedness can be improved by adjusting the learning speed independently by using the gradient value corresponding to each point.

2. Related Works

2.1. Implicit fields

Based on a wide variety of 3D representations, there have been different approaches to improve the visual quality of the results in single-view reconstruction. Remarkable progress has been made, but most works fail to recover detailed information from input images. To solve this limitation, the function of learning an implicit field is introduced for generative shape modeling [22–24,26]. An implicit field is defined by a continuous function over space; therefore, it assigns a value to each point in the 3D space. We can use this to reconstruct the 3D polygonal meshes generating the isosurfaces by marching cubes [29].

There are different ways to define implicit fields depending on the network. For a closed shape, Chen et al. [26] and Mescheder et al. [22] defined the inside/outside field F of the shape using a simple binary classification problem:

$$F(p) = \begin{cases} 0 & \text{if point } p \text{ is outside the surface} \\ 1 & \text{if point } p \text{ is inside the surface} \end{cases} \tag{1}$$

The mapping parameters are trained to determine the occupancy using a deep network composed of multi-layer perceptrons (MLPs) and nonlinearities.

Meanwhile, Xu et al. adopted an alternative implicit representation called SDF to improve the performance of recovering local details. An SDF encodes the signed distance of each point sample in 3D from the surface. In other words, an iso-surface is defined as $S_0 = \{\mathbf{p}|SDF(\mathbf{p}) = 0\}$. Subsequently, a point is outside the surface if $s > 0$, and inside if $s < 0$.

Among the works using Implicit fields, Chen et al. introduced an implicit field decoder, called IM-NET [26], for shape generation, aimed at improving the visual quality of the generated shapes. As previously explained, an implicit field assigns a value to each voxel in space, and Chen et al. performed this assignment using a binary classifier. In summary, IM-NET takes the encoded feature vector and point coordinates as inputs and predicts the occupancy as the output.

When IM-NET is used in single-view reconstruction, it passes through two stages, IM-AE and IM-SVR, which are commonly composed of an encoder and decoder (auto-encoder) trained in sequence. (Figure 5) In IM-AE, the encoder is trained to create feature vectors from 3D input objects in the ShapeNet 3D dataset [28]. Thereafter, IM-NET generates a 3D shape from the feature vector and learns to obtain a better result by comparing the result with the ground truth, which is the initial input. As this process is repeated, the encoder learns to create a feature vector that represents the object well. The final latent feature vector created as a result of training is called a z-vector, which is used as the ground truth in training another encoder in IM-SVR.

Likewise, an encoder and a decoder are trained in IM-SVR, and the overall learning process is similar. However, one difference is that the encoder of SVR intakes a 2D image, whereas that of AE encodes 3D data. Subsequently, the encoder outputs the shape feature vectors, minimizing the loss with the z-vector from the previous stage. Since SVR shares the decoder with AE, the more similar the encoded feature vector, the better the decoded predicted result. After both stages of training, SVR can reconstruct a 3D shape from a 2D input image. Despite this progress, the shapes produced by IM-NET are still unsatisfactory in terms of visual quality and recovery of local details. We propose a new method in which the loss for updating the occupancy of each voxel is weighted differently according to the 3D gradient to improve the visual quality, and we demonstrate its results through an experiment using IM-NET.

3. Proposed Method

3.1. Preliminary Study

An implicit field is a continuous scalar field over a 3D space (x, y, z). Hence, the gradient of the implicit field $F(x, y, z)$ is represented as :

$$\nabla F(x, y, z) = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z} \right) \tag{2}$$

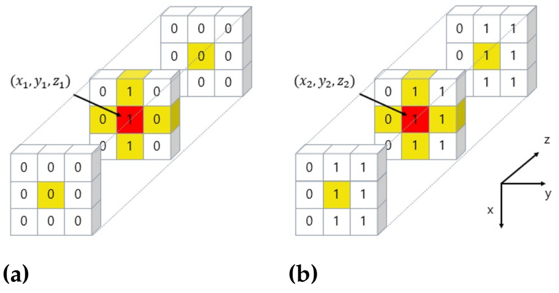

Figure 2. Example of voxel gradient calculation: (a) A point on a line in the x direction and (b) a point on a surface of a normal vector in the y direction

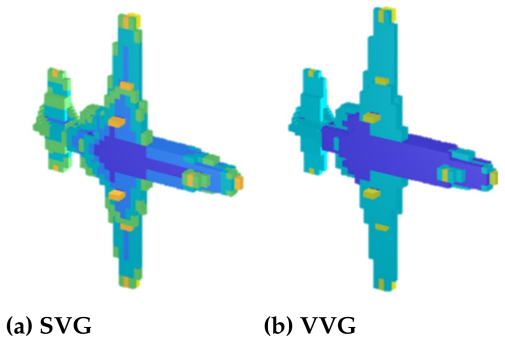

Figure 3. Voxelized gradient map (a) created from scalar voxel gradient (b) created from vector voxel gradient in which yellow cells represent higher weights than blue cells. Both methods were normalized to the same magnitude range.

Note that IM-NET assigns occupancy values to voxels that have discrete properties. We can treat ∂x, ∂y, and ∂z as any small integer h. Then, equation (2) can be expressed as follows with regard to one voxel of coordinate (x, y, z) :

$$\frac{\partial F(x,y,z)}{\partial x} \approx \frac{F(x+h,y,z) - F(x,y,z)}{h} \tag{3}$$

Likewise, we can approximate the Laplacian and obtain discrete Laplacians on implicit fields:

$$\begin{aligned}\triangle F(x,y,z) \approx \frac{F(x+h,y,z)+F(x-h,y,z)}{h^2} + \frac{F(x,y+h,z)+F(x,y-h,z)}{h^2} \\ + \frac{F(x,y,z+h)+F(x,y,z-h)}{h^2} - \frac{6F(x,y,z)}{h^2}\end{aligned} \tag{4}$$

In our experiment, we modified the 3D discrete Laplacians above to develop a methodology to quantify the details of 3D shapes. Modified Laplacians are used to recover the fine-grained details by varying the learning rate for each voxel.

3.2. Concept

Previous works on implicit representation contain a feature that differentiates them from explicit representation methods. A network that outputs the explicit representation as a voxel, mesh, and point cloud generates an overall shape for the input image at once. Conversely, a network generating implicit representation creates a 0 or 1 for each 3D coordinate $p(x, y, z)$ as an input. The entire shape is created when generating a high-resolution object, which enables the network to operate even with low memory or to process efficiently by parallel processing. However, it is difficult to consider comprehensive information of the entire shape. This problem is clearly revealed in the

Algorithm 1: Algorithm of proposed method

(a) AE

```
Input: 3D training data $\boldsymbol{X_{3D}}$
Initialize the model parameter $\theta_{EN-AE}$, $\theta_{DE}$
N = $n(\boldsymbol{X_{3D}})$
$R_{3D}$ = the resolution of $\boldsymbol{X_{3D}}$
while $epoch < epoch_{max}$ do
    while $i < N, x < R_{3D}, y < R_{3D}, z < R_{3D}$ do
        3D point coordinate $p_i = (x, y, z)$
        prediction = $F_\theta(p_i)$
        binary ground truth = $o_{p_i}$
        weight $\boldsymbol{g(p_i)} = g(x, y, z)$
          $= |F(x+2, y, z) - F(x, y, z)| + |F(x-2, y, z) - F(x, y, z)| + |F(x, y+2, z) - F(x, y, z)|$
          $+ |F(x, y-2, z) - F(x, y, z)| + |F(x, y, z+2) - F(x, y, z)| + |F(x, y, z-2) - F(x, y, z)|$
          Loss $+= \boldsymbol{g(p_i)} \| o_{p_i} - F_\theta(p_i) \|_2^2$
    end
    Update the model parameters $\theta_{EN-AE}$, $\theta_{DE}$
end
Output: $\mathbb{Z}$ = encoded feature vector after the training is completed
```

(b) SVR

```
Input: 2D training data $\boldsymbol{X_{2D}}$, $\mathbb{Z}$
Initialize the model parameter $\theta_{EN-SVR}$
N = $n(\boldsymbol{X_{2D}})$
$R_{2D}$ = the resolution of $\boldsymbol{X_{2D}}$
while $epoch < epoch_{max}$ do
    while $i \leq N$ do
        encoded feature vector = $G(X^i_{2D})$
        ground truth = $\mathbb{Z}_i$
        Loss $+= \| \mathbb{Z}_i - \mathrm{G}(\mathrm{X}^i_{2D}) \|_2^2$
    end
    Update the model parameters $\theta_{EN-SVR}$
end
Output: 3D shape
```

generation of fine-grained and detailed structures. To address this, we propose a loss function that considers the context information around the input coordinate.

In this paper, we hypothesize that the main cause of the failure to restore fine-grained local areas is the data imbalance of the object. In general, a small area occupies a small proportion compared to the whole area, so the entire network is trained by focusing on a thickened or wide range. The proposed loss function gives a weight of 1 for the overall region and a weight of gradient value $g(p)$ for the fine region with regard to each point p in 3D space, which increases the number of data by $g(p)$ times. The magnitude of the weight depends on the degree of detail, and it can be quantified by the gradient $g(p)$ we suggest.

A gradient map is created with a scalar or vector form of voxel gradients. First, the gradient map can be depicted as a scalar field, representing the rate of change in the topology, and equation

$$\begin{aligned} g_s(x,y,z) = {} & |F(x+h,y,z) - F(x,y,z)| + |F(x-h,y,z) - F(x,y,z)| + |F(x,y+h,z) - F(x,y,z)| \\ & + |F(x,y-h,z) - F(x,y,z)| + |F(x,y,z+h) - F(x,y,z)| + |F(x,y,z-h) - F(x,y,z)| \end{aligned} \tag{5}$$

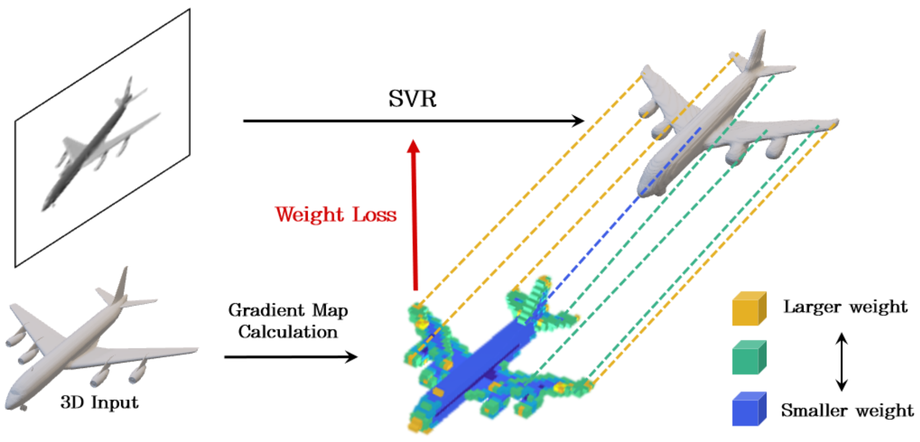

Figure 4. Simple conceptual framework of gradient map application

which is derived from equation (4), can be used to calculate the scalar voxel gradient. The gradient value is defined as the sum of the differences in the occupancy value between the central voxel and the surrounding voxels. Hence, with regard to the object of which the implicit value is $F(x, y, z)$, the voxel gradient is expressed as another scalar field.

Figure 2(a) shows the voxel gradient of a linear structure that is extended in the x-direction, computed as $g_s(x_1, y_1, z_1) = 4$. If we apply the same method to the surface that has a normal vector in the -y direction, $g_s(x_2, y_2, z_2) = 1$. (Figure 2(b)) The larger gradient value implies that the thin line has a more fine-grained structure than the plane. In other words, a larger weight is applied to point (x_1, y_1, z_1) than point (x_2, y_2, z_2). Similarly, the gradient values of the other coordinates are calculated, and those compose another scalar field of the same voxel size, which we call a gradient map. For example, Figure 3(a) shows the gradient map created from scalar voxel gradient.

- **SVG (Scalar Voxel Gradient-based method)** : As explained in equation(5), it is the method using scalar voxel gradient. The gradient value of each point is determined by the sum of the difference of implicit values between the point and surrounding points. For example in Figure 2, the gradient values of central point are (a)4 and (b)1. The detailed explanation is well described in Algorithm 1.
- **VVG (Vector Voxel Gradient-based method)** : It is the method using vector voxel gradient. We notice that if there are more 2-component in the vector gradient of a certain point, it is inferred to be fine-grained part as line or flat structure. Hence, the larger weight proportional to the number of 2-component is given to the point when training. Therefore, the gradient map made by VVG is inclined to have discontinuous color distribution as the weight is given exclusively to the fine-grained parts which we are interested in, while no weight is given to the points which have no 2-component in the gradient vector.

3.3. Weighted Loss

Our strategy is to diversify the learning rate of each occupancy value by weighting the loss functions differently, according to which local parts of the voxel data are fine-grained and what level of detail description is required. That is, the larger the gradient value, the larger is the weight given to each coordinate. Figure 4 shows a simple conceptual framework and the principle of applying a gradient map to the SVR task. The gradient map is constructed from 3D training data, and each gradient value is reflected in the corresponding part of the object in the training stage of SVR. In the figure, a larger weight is multiplied by the loss function of the implicit value of the fine-grained parts that correspond to the yellow (larger $g(p)$) parts on the gradient map (the wheels, edge

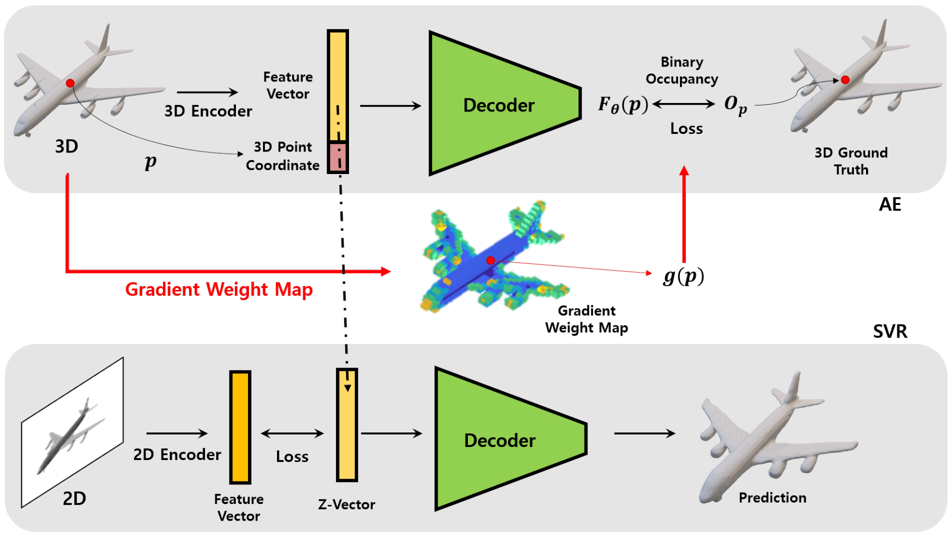

Figure 5. The network structure of IM-NET for SVR task with our gradient weight map. Shadowed boxes at the top(AE) and the bottom(SVR) represent the original network structure of IM-NET [26]. This flow chart shows how and when the designed gradient weight map is applied in our algorithm. This shows how the gradient map was applied to the loss function. The lighter part of the gradient map represents a higher gradient value, and this is applied to the loss function as a larger weight

of wings, tails of the plane). Meanwhile, the implicit value of monotonous parts that have a small $g(p)$ is trained by a small weight.

Figure 5 shows how and where the gradient map specifically reflected on the loss function in former method. As explained in Related Works, two auto-encoders were used in previous technique for the SVR task and the training of AE precedes the training of SVR, as the latter is trained with the z-vector as its ground truth. The entire gradient map was created from the 3D training dataset. The decoder was trained so that it could predict the correct implicit field value $F_\theta(p)$ of each point. The gradient value $g(p)$ was multiplied by the MSE loss function between the prediction and ground truth. Let θ be the trained parameter, o_p, the predicted occupancy of points p, and $F_\theta(p)$, the binary ground truth. Then, we have

$$L(\theta) = \sum_p g(p) \| \, o_p - F_\theta(p) \, \|_2^2. \tag{6}$$

Algorithm 1 provides a detailed description of the overall proposed algorithm.

After completing the training of AE, SVR is trained so that the 2D image is encoded to the feature vector with regard to the same object, following the fully-trained feature vector in AE. In other words, the tendency to preserve details is reflected by the weighted loss of the gradient map in AE and is transferred to the SVR. The weighted loss relieves some of the data imbalance problems, thereby improving the restorative performance of the structure.

4. Experiment

In this section, we applied a voxelized gradient weight map to IM-NET [26] and compared its performance with that of the original network to test the effectiveness of our theory. We used ShapeNet [28] for this experiment and evaluated our method on five representative categories: airplanes, cars, chairs, rifles, and tables. Each category contained 3,236, 5,996, 5,422, 1,897, and 6,807 in the training set and 809, 1,500, 1,356, 475, and 1,702 in the test set. In the evaluation, we chose 100 samples from each category and calculated the mean metrics for the qualitative results.

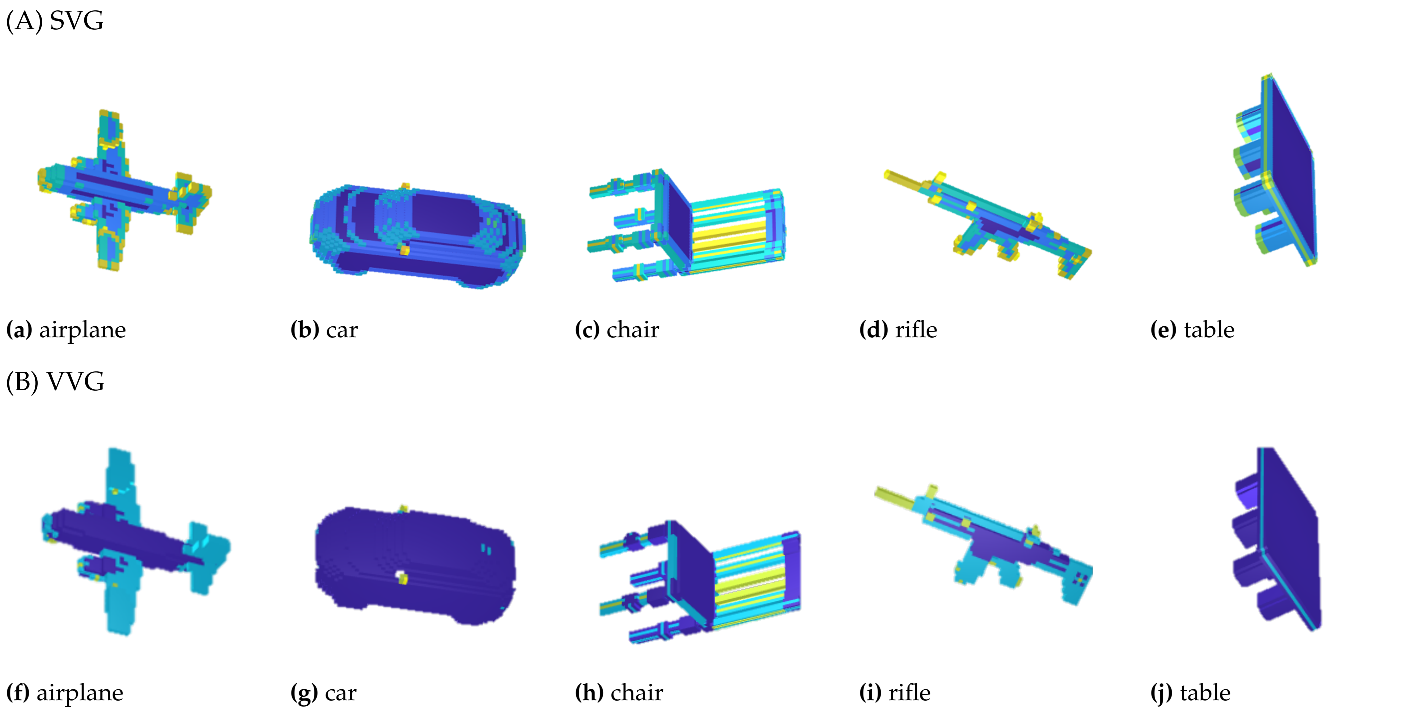

Figure 6. Gradient weight map of the objects in each category. It should be noted that highlighted areas include in (a) the nose, wheels, edges of wings, fins, and tail; in (b) the side mirrors; in (c) the legs and splats; in (d) the muzzle, barrel, sights, edges of grip, magazine, and stock; and in (e) the legs and edges. Note that each object has a different maximum value. In addition, it is easily seen that the weights are distributed discontinuously in VVG(f j) compared to SVG(a e)

For equivalent comparisons, all methods were implemented using Python in Linux under the same conditions: a computer equipped with an Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz, two 16-GB RAM, and a GeForce RTX 3090 graphic card.

4.1. Gradient Weight Map

To implement SVG, we simply applied the method using the total sum of the differences between the implicit field values of the x, y, and z directions surrounding the central voxel, as represented by equation (5) with $h = 2$. In other words, we calculated a new 64^3 gradient scalar field using the equation from the same size of the binary implicit field function, which represents the inside and outside of an object. In addition, as mentioned in Proposed Method, another gradient map for VVG is created such that if it is 2-component in vector gradient, the large weight value proportional to the number of 2 is given to each point. Therefore, unlike SVG, VVG applies the weights to fine-grained parts exclusively, so the weight distribution is slightly discontinuous compared to SVG. Figure 6 shows the gradient weight map calculated from the data, where the edges, thin structures, and small parts of the objects are highlighted with high values (bright color). In this way, we calculated the entire set of gradient maps of the training set and put these data into the training stage as another input.

4.2. Training

As explained in Related Works, we need to train both auto-encoders, AE and SVR, and put the 64^3 gradient map data input into the first stage where the 64^3 to 256^3 transformation is learned and a latent vector is created. In this process, the value of the gradient map, which has a one-to-one correspondence with the input voxel, is applied as a weight to the loss function when the implicit field value of each point is learned. Note that the maximum value of the gradient map was 6 according to equation (5), and we normalized the weight such that the maximum weight value became 16. We expected this method to maintain the generating performance of the entire global shape and also

Metric	Model	Plane	Car	Chair	Rifle	Table
Mean IoU(×100)	Original[26]	57.58	82.77	48.19	49.21	53.11
	SVG	59.14	**83.63**	**49.22**	**50.37**	55.48
	VVG	**59.82**	82.70	48.88	49.70	**55.50**
MSE(×0.1)	Original[26]	99.33	231.33	499.68	89.52	553.64
	SVG	96.25	**218.65**	**494.61**	**88.30**	**525.88**
	VVG	**95.89**	233.37	497.99	89.17	532.95

Table 1: 3D reconstruction errors with 64 sampling resolutions. The best performance numbers are shown in bold.

vary the learning speed of the local parts to improve the recovery performance of the details.

For the 3D shape, we trained the network in hierarchical order [30] to obtain the voxel models at different resolutions ($16^3, 32^3$, and 64^3). The AE encoders were trained on 64^3 resolution data for the same number of 200 epochs for each resolution. SVR encoders, which generate 3D results from 2D images, were trained for 1,000 epochs. The experiment was conducted in a relatively challenging environment, as the network was trained on five categories simultaneously.

4.3. Inference and Result

Because the resolution of the binary implicit field of the ground truth was 64^3, we generated a voxel output of the same size to calculate the metrics. The voxel implicit field data were extracted before the voxel passed through the marching cube. We calculated the IoU and MSE for quantitative analysis by comparing these voxel values with the ground truth. We chose the first 100 objects for each tested class and calculated their IoU and MSE. Each object in the dataset has 24 views of 2D images, hence, the network generates the same number of 3D outputs depending on each input image. Hence, we calculated and showed the average metric for a total of $24 \times 100 = 2,400$ 3D results. Table 1 evaluates the reconstruction results using two common evaluation metrics, IoU and MSE. Both of our methods outperformed the original reconstruction network in all five categories and with regard to both metrics. In most cases, better performance was shown in IoU when the weight was given evenly(SVG) rather than when it was given intensively only to the local part(VVG).

Qualitative evaluations are shown in Figure 7, where the models implemented with the gradient weight map generally perform better. SVRs with a gradient weight map represent details well(the wheels of the plane, the stabilizer of the jet, the side mirrors of the car, the seats of the convertible car, separated splats of the chair, and the trigger of pistol) whereas the original method either roughly generates or fails to recover the details. Moreover, original SVR is inclined to be dragged into a common form of training data. This is because most planes have bent wingtips and four table legs. In contrast, SVR with a gradient weight map deals with the deviation of the input from the generality of the training data. Nevertheless, all three are not adequate enough to reconstruct the objects in the rifle class, and we infer that this is because the number of training data is insufficient, compared with other classes. To compare two proposed methods, while SVG generates a uniform overall surface better, VVG produces even smaller and finer details. Therefore, it is important to properly converge the two techniques in order to achieve the two effects in a balanced way for the purpose, and further analysis of this issue is left to the future work. Furthermore, the training time was measured to determine the amount of penalty incurred in terms of learning time when the gradient map was given as an additional input. It took 32,507 s for the original network and 33,156 s for our method to train the AE for the entire dataset used in the experiment. Therefore, it is confirmed that there is almost no time penalty required to input the gradient map for learning. This result is easily predictable and straightforward, as there is no structural change in the model, and there is no change in the number of parameters in the network. Yielding improved results without computational complexity is clearly one of the powerful advantages of our algorithm.

5. Conclusion

We suggest a simple modification to enhance the performance of the SVR model, especially for the improved reconstruction of fine-grained boundaries. The gradient map can be easily plugged into the loss function in various forms, not only as a scaling factor in this experiment, thereby diversifying the learning rate considering the topological property of the object. In other words, the network is allowed to focus more on detailed parts while maintaining the quality of the overall global shape.

Recently, implicit fields have been actively used in shape generation because of their prominent performance. One of the merits of our method is that the gradient map can be used not only for binary implicit fields but also for the SDF [24], as the gradient values can be calculated from the implicit fields to construct the gradient map. Naturally, it takes time to create a gradient map by applying the designed operations at each point of the data in the training set. For example, it takes approximately one day to calculate a gradient map for all 64^3-resolution data consisting of 35,019 objects. The operation time increases rapidly when the training set is large, especially as the resolution increases. Nevertheless, this time cost is relatively small compared with the training time of most methods dealing with 3D data.

Modulating the loss function with the gradient map preserves local details, and the result would be further improved by methods that consider the geometric relations of the points. However, our method does not target the diagonal change. That is, as our method only considers the gradient in the x, y, and z directions, it is difficult to capture structures such as a thin line that has a vector of two or three directional components. Moreover, the shape which has the maximum gradient weight in our method is an occupied point surrounded by an empty space or a void in an object, which is not an actual shape to be weighted. Further study is needed to find a method that captures fine and thin shapes better.

Our future work will include investigating the optimized design of a gradient map, applying directional information in topology. Moreover, our method is not limited to 3D reconstruction but can also be used in encoding and generating 2D images, thus providing variation and flexibility in training speed. We expect that applying a gradient weight map has the potential for further 3D deep machine learning studies. We envision that our method will contribute to building delicate virtual spaces and configuring efficient simulation environments. Ultimately, it is expected to lower production costs for simulation to raise production efficiency in industries, and allow the general public to experience satisfactory services.

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

Funding: This research received no external funding.

Acknowledgments: This research is supported by Ministry of Culture, Sports and Tourism and Korea Creative Content Agency(Project Number: R2020040238).

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

Abbreviations

The following abbreviations are used in this manuscript:

IoU	Intersection over Union
MSE	Mean Square Error
SVR	Single-view Reconstruction
SVG	Scalar Voxel Gradient
VVG	Vector Voxel Gradient

References

1. Su, H.; Maji, S.; Kalogerakis, E.; Learned-Miller, E. Multi-view convolutional neural networks for 3d shape recognition. Proceedings of the IEEE international conference on computer vision, 2015, pp. 945–953.
2. Kalogerakis, E.; Averkiou, M.; Maji, S.; Chaudhuri, S. 3D shape segmentation with projective convolutional networks. proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 3779–3788.
3. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. Pointnet: Deep learning on point sets for 3d classification and segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.
4. Kar, A.; Tulsiani, S.; Carreira, J.; Malik, J. Category-specific object reconstruction from a single image. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1966–1974.
5. Rock, J.; Gupta, T.; Thorsen, J.; Gwak, J.; Shin, D.; Hoiem, D. Completing 3d object shape from one depth image. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2484–2493.
6. Lun, Z.; Gadelha, M.; Kalogerakis, E.; Maji, S.; Wang, R. 3d shape reconstruction from sketches via multi-view convolutional networks. 2017 International Conference on 3D Vision (3DV). IEEE, 2017, pp. 67–77.
7. Dibra, E.; Jain, H.; Oztireli, C.; Ziegler, R.; Gross, M. Human shape from silhouettes using generative hks descriptors and cross-modal neural networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4826–4836.
8. Fan, H.; Su, H.; Guibas, L.J. A point set generation network for 3d object reconstruction from a single image. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 605–613.
9. Wu, J.; Zhang, C.; Xue, T.; Freeman, W.T.; Tenenbaum, J.B. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 82–90.
10. Kittler, J.; Hilton, A.; Hamouz, M.; Illingworth, J. 3D assisted face recognition: A survey of 3D imaging, modelling and recognition approachest. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops. IEEE, 2005, pp. 114–114.
11. Girdhar, R.; Fouhey, D.F.; Rodriguez, M.; Gupta, A. Learning a predictable and generative vector representation for objects. European Conference on Computer Vision. Springer, 2016, pp. 484–499.
12. Zhu, R.; Kiani Galoogahi, H.; Wang, C.; Lucey, S. Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image. Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 57–65.
13. Tulsiani, S.; Zhou, T.; Efros, A.A.; Malik, J. Multi-view supervision for single-view reconstruction via differentiable ray consistency. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2626–2634.
14. Wu, J.; Zhang, C.; Zhang, X.; Zhang, Z.; Freeman, W.T.; Tenenbaum, J.B. Learning shape priors for single-view 3d completion and reconstruction. Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 646–662.
15. Yang, G.; Cui, Y.; Belongie, S.; Hariharan, B. Learning single-view 3d reconstruction with limited pose supervision. Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 86–101.
16. Groueix, T.; Fisher, M.; Kim, V.G.; Russell, B.C.; Aubry, M. A papier-mâché approach to learning 3d surface generation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 216–224.
17. Sinha, A.; Unmesh, A.; Huang, Q.; Ramani, K. Surfnet: Generating 3d shape surfaces using deep residual networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 6040–6049.
18. Wang, N.; Zhang, Y.; Li, Z.; Fu, Y.; Liu, W.; Jiang, Y.G. Pixel2mesh: Generating 3d mesh models from single rgb images. Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 52–67.
19. Yang, Y.; Feng, C.; Shen, Y.; Tian, D. Foldingnet: Point cloud auto-encoder via deep grid deformation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 206–215.
20. Arsalan Soltani, A.; Huang, H.; Wu, J.; Kulkarni, T.D.; Tenenbaum, J.B. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1511–1519.

21. Lin, C.H.; Kong, C.; Lucey, S. Learning efficient point cloud generation for dense 3d object reconstruction. proceedings of the AAAI Conference on Artificial Intelligence, 2018, Vol. 32.
22. Mescheder, L.; Oechsle, M.; Niemeyer, M.; Nowozin, S.; Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4460–4470.
23. Dai, A.; Ruizhongtai Qi, C.; Nießner, M. Shape completion using 3d-encoder-predictor cnns and shape synthesis. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5868–5877.
24. Park, J.J.; Florence, P.; Straub, J.; Newcombe, R.; Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 165–174.
25. Petersen, F.; Bermano, A.H.; Deussen, O.; Cohen-Or, D. Pix2vex: Image-to-geometry reconstruction using a smooth differentiable renderer. *arXiv preprint arXiv:1903.11149* **2019**.
26. Chen, Z.; Zhang, H. Learning implicit fields for generative shape modeling. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5939–5948.
27. Xu, Q.; Wang, W.; Ceylan, D.; Mech, R.; Neumann, U. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. *arXiv preprint arXiv:1905.10711* **2019**.
28. Chang, A.X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; others. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* **2015**.
29. Lorensen, W.E.; Cline, H.E. Marching cubes: A high resolution 3D surface construction algorithm. *ACM siggraph computer graphics* **1987**, *21*, 163–169.
30. Häne, C.; Tulsiani, S.; Malik, J. Hierarchical surface prediction. *IEEE transactions on pattern analysis and machine intelligence* **2019**, *42*, 1348–1361.

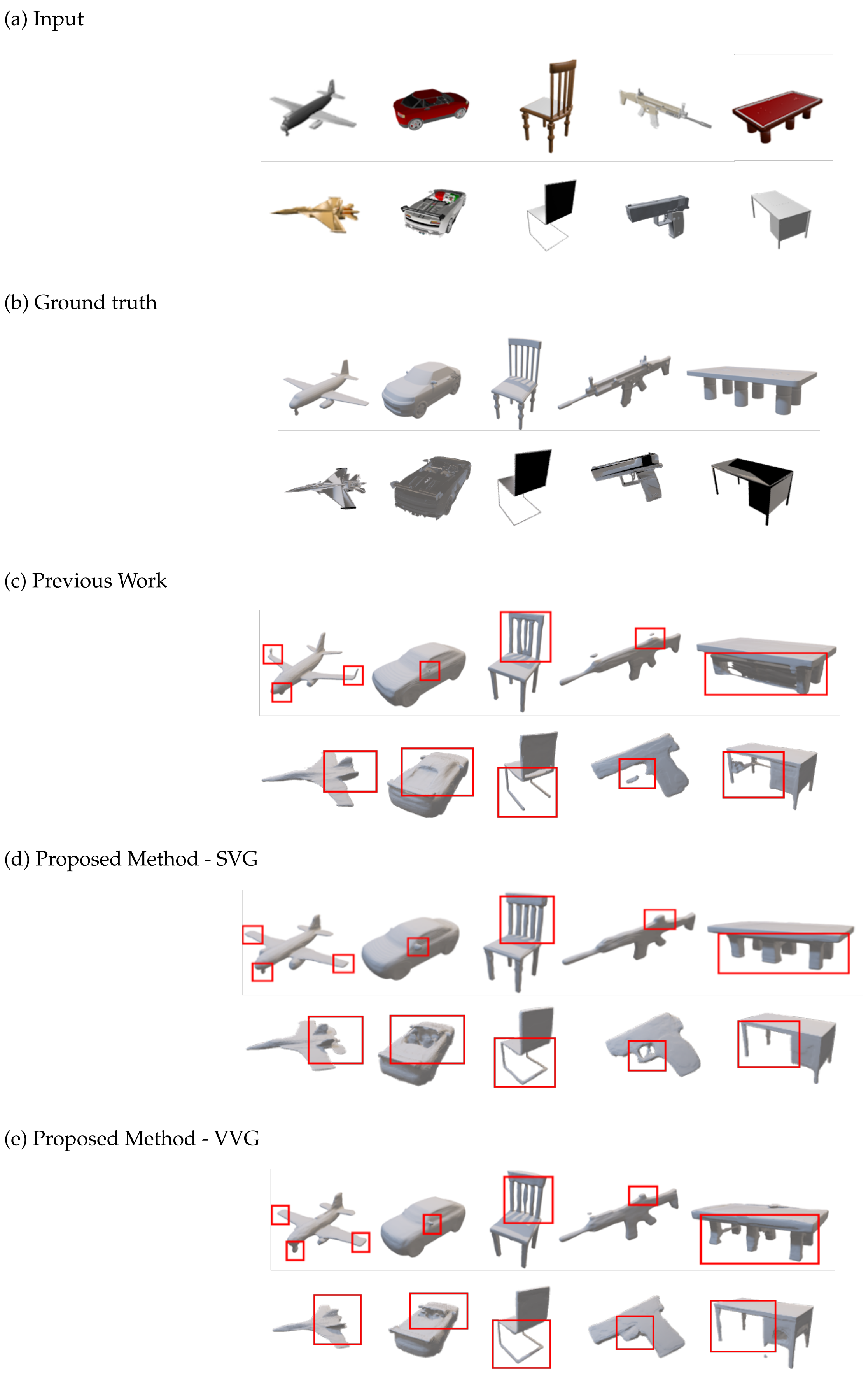

Figure 7. 3D shape generation results sampled at 256^3 resolution. One generated shape from each category is shown for each model. (a) 2D input single-view image, (b) ground truth, (c) models reconstructed by original IM-SVR, (d) models reconstructed by IM-SVR with gradient weight map created by SVG, (e) models reconstructed by IM-SVR with gradient weight map created by VVG