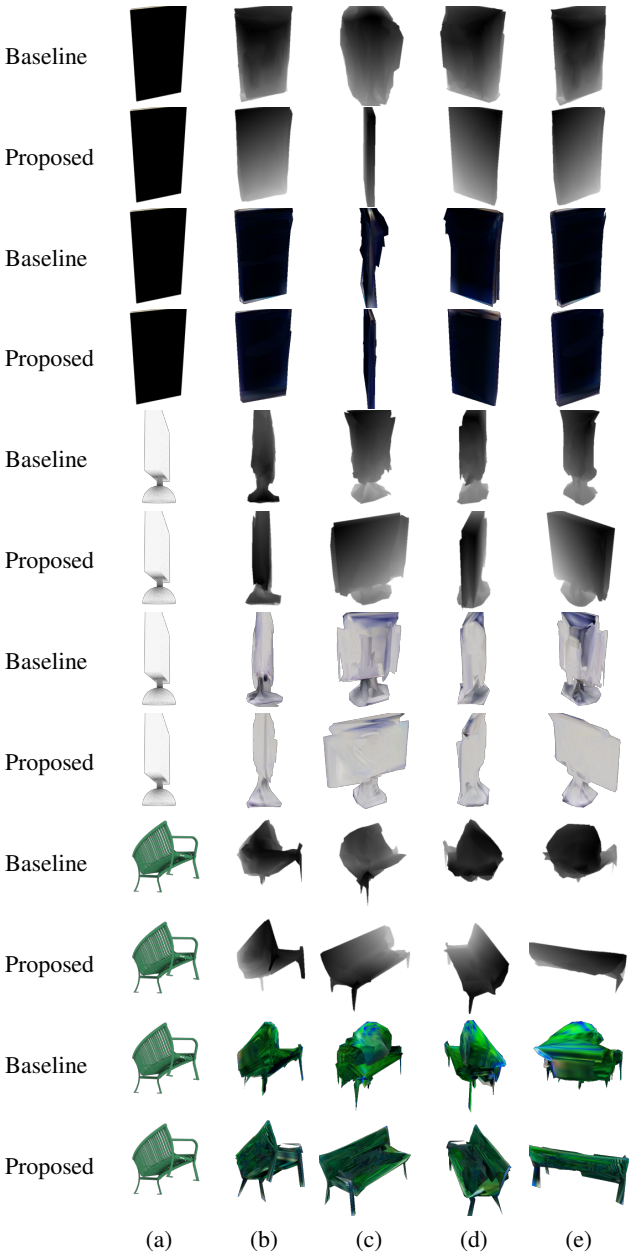


Figure 8. Examples on the ShapeNet dataset in single-view training. Panels (a–e) are the same as in Figure 4. This figure corresponds to Section 4.1.1.

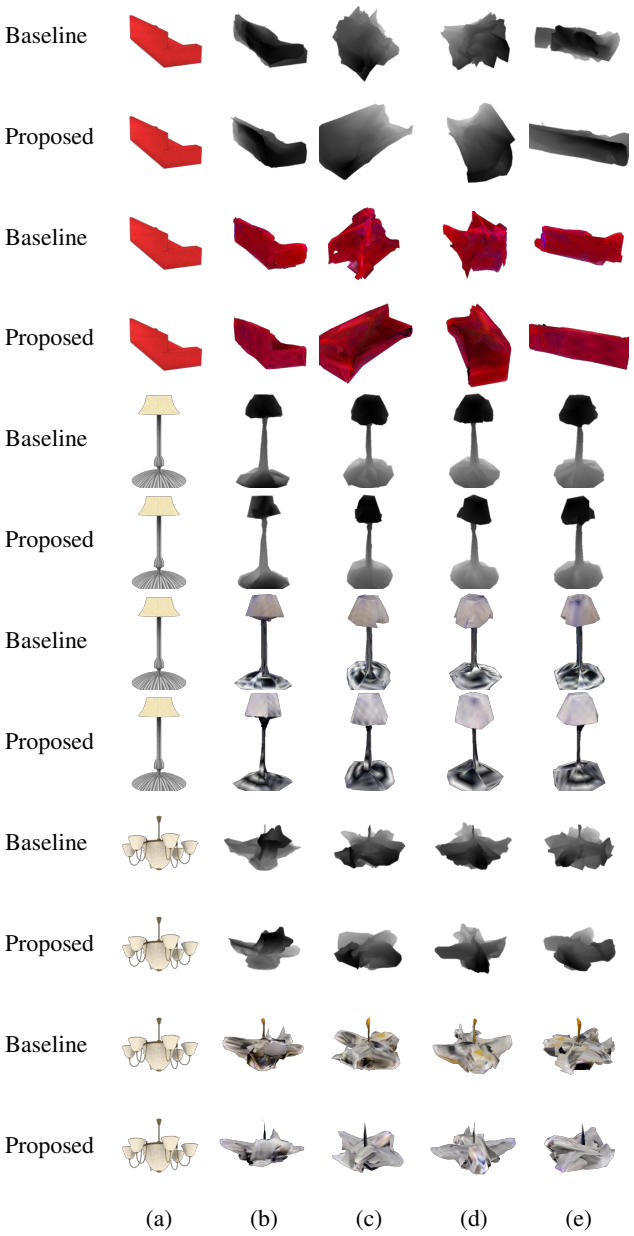


Figure 9. Examples on the ShapeNet dataset in single-view training. Notation of (a–e) is the same as in Figure 4. This figure corresponds to Section 4.1.1.

# A. Appendix

## A.1. Additional examples of single-view training

Figures 8 and 9 show some reconstruction examples of *phone*, *display*, *bench*, *sofa*, and *lamp* categories on the ShapeNet dataset using single-view training. These figures correspond to the description in Section 4.1.1. The difference among categories and the use of texture prediction can be examined from these figures.

## A.2. Performance of each category by multi-view training

In Section 4.1.2, only the average performance in all categories on the ShapeNet dataset has been reported. Table 7 shows reconstruction accuracy of each category.

## A.3. Additional examples of multi-view training

Figures 10 and 11 show results from our best performing models using multi-view training ($N_v = 20$) for those interested in the

| $N_v$ | VPL | TP | airplane | bench | dresser | car | chair | display | lamp | speaker | rifle | sofa | table | phone | vessel | all |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | | | .615 | .467 | .658 | .767 | .512 | .465 | .476 | .631 | .603 | .588 | .517 | .622 | .557 | .575 |
| 2 | ✓ | | .619 | .476 | .667 | .770 | .518 | .467 | .477 | .631 | .598 | .590 | .529 | .687 | .556 | .583 |
| 3 | | | .631 | .495 | .673 | .775 | .537 | .499 | .490 | .639 | .624 | .599 | .535 | .672 | .573 | .596 |
| 3 | ✓ | | .638 | .507 | .674 | .779 | .543 | .497 | .491 | .638 | .625 | .607 | .552 | .680 | .574 | .600 |
| 5 | | | .654 | .530 | .696 | .787 | .554 | .539 | .502 | .657 | .642 | .623 | .564 | .721 | .589 | .620 |
| 5 | ✓ | | .662 | .542 | .699 | .792 | .565 | .533 | .502 | .656 | .647 | .629 | .571 | .725 | .593 | .624 |
| 10 | | | .673 | .570 | .712 | .795 | .582 | .572 | .510 | .671 | .660 | .640 | .585 | .750 | .606 | .641 |
| 10 | ✓ | | .681 | .577 | .718 | .798 | .584 | .570 | .508 | .673 | .661 | .648 | .593 | .761 | .606 | .644 |
| 20 | | | .688 | .593 | .722 | .799 | .597 | .599 | .512 | .678 | .665 | .651 | .595 | .766 | .615 | .652 |
| 20 | ✓ | | .691 | .598 | .724 | .802 | .601 | .597 | .505 | .680 | .664 | .656 | .607 | .775 | .613 | .655 |
| 2 | | ✓ | .614 | .469 | .663 | .768 | .511 | .475 | .477 | .624 | .605 | .582 | .523 | .649 | .560 | .579 |
| 2 | ✓ | ✓ | .618 | .481 | .666 | .772 | .522 | .476 | .480 | .631 | .607 | .589 | .529 | .678 | .557 | .585 |
| 20 | | ✓ | .685 | .588 | .723 | .799 | .597 | .589 | .508 | .674 | .663 | .648 | .603 | .762 | .615 | .650 |
| 20 | ✓ | ✓ | .701 | .585 | .723 | .802 | .604 | .593 | .502 | .659 | .661 | .667 | .609 | .746 | .631 | .653 |

Table 7. IoU of multi-view training on the ShapeNet dataset dataset. This table corresponds Section 4.1.2. VPL: proposed view prior learning. TP: texture prediction.

state-of-the-art performance on the ShapeNet dataset. As can be seen in the figure, high-quality 3D models with textures can be reconstructed without using 3D models for training. The mean IoU of our method without texture prediction is 65.5, and the mean IoU of our method with texture prediction is 65.0. In contrast to single-view training, texture prediction does not improve the performance in multi-view training for large $N_v$.

## A.4. Evaluation using CD and EMD

In addition to intersection over union (IoU), Chamfer distance (CD) and earth mover's distance (EMD) are also often used for evaluation of 3D reconstruction. Table 8, 9, 10, 11, and 12 show CD and EMD in the experiment of Table 2 in the paper. We computed CD and EMD from points uniformly sampled from surfaces and volumes.

CDs and EMD$_v$ correlate well to IoUs, which also validates our proposed method. EMD$_s$ seems strange because EMD is greatly affected by spatial density of points and our method often generates spatially imbalanced surfaces. However, this imbalance hardly affects visual quality because it is often made by folding surfaces inside shapes. EMD$_v$, which is computed from spatially uniform points, shows similar performance as other metrics.

## A.5. Discriminators and optimization

Table 13 shows the performances of the discriminators in Table 1 (c–d) in single-view training. The discriminator in Table 1 (d) does not work well in all cases. This table corresponds to the description in Section 4.1.3.

## A.6. Internal pressure in multi-view training

In Section 4.1.4, we validated the effect of the internal pressure loss in single-view training. Table 14 shows that this loss is also effective in multi-view training. This experiment was conducted without texture prediction and view prior learning.

## A.7. Loss functions of silhouettes

To compare two silhouette images, we used multi-scale cosine distance in Eq. 3 and intersection over union (IoU) of silhouettes in Eq. 4. Table 15 shows comparison of these loss functions in multi-view training ($N_v = 2$) on ShapeNet dataset. Additionally, sum of squared error of two silhouette images is compared. This result indicates that non-standard loss functions described in this paper are not so effective.

## A.8. Modification of neural mesh renderer

In our implementation, we compute the differentiation of a renderer in a different way from [18]. Their method is not stable when $\delta_i^x$ in [18] is very small. Furthermore, the computation time is significant because a very large number of pixels is involved in computing the gradient with respect to one pixel. The approximate differentiation described in this section solves both problems.

Suppose three pixels are aligned horizontally, as shown in Figure 12 (a). Their coordinates are $(x_{i-1}, y_{i-1})$, $(x_i, y_i)$, and $(x_{i+1}, y_{i+1})$, and their colors are $p_{i-1}$, $p_i$, and $p_{i+1}$, respectively. Pixel $i$ is located on a polygon, and its three vertices projected onto a 2D plane are $(x_1^v, y_1^v)$, $(x_2^v, y_2^v)$, and $(x_3^v, y_3^v)$. Then $(x_i, y_i)$ can be represented as their weighted sum $(x_i, y_i) = w_1(x_1^v, y_1^v) + w_2(x_2^v, y_2^v) + w_3(x_3^v, y_3^v)$. Let $\mathcal{L}$ be the loss function of the network. When the gradient with respect to pixel $(\frac{\partial \mathcal{L}}{\partial x_i}, \frac{\partial \mathcal{L}}{\partial y_i})$ is obtained, the gradient with respect to the vertices of the polygon $(\frac{\partial \mathcal{L}}{\partial x_1^v}, \frac{\partial \mathcal{L}}{\partial y_1^v})$, $(\frac{\partial \mathcal{L}}{\partial x_2^v}, \frac{\partial \mathcal{L}}{\partial y_2^v})$, $(\frac{\partial \mathcal{L}}{\partial x_3^v}, \frac{\partial \mathcal{L}}{\partial y_3^v})$ can be computed using $w_1$, $w_2$, $w_3$, and the chain rule.

We assume that when pixel $i$ moves to the right by $\Delta x_i$, the pixel colors change, as shown in Figure 12 (b). Concretely, the color of pixel $i$ changes to $p_i + (p_{i-1} - p_i)\Delta x$ and the color of pixel $i+1$ changes to $p_{i+1} + (p_i - p_{i+1})\Delta x$. Then, $\frac{\partial p_i}{\partial x_i} = p_{i-1} - p_i$ and $\frac{\partial p_{i+1}}{\partial x_i} = p_i - p_{i+1}$. Let $g_i$ be the gradient of the loss function back-propagated to pixel $i$. Concretely, $g_i = \frac{\partial \mathcal{L}}{\partial p_i}$.

| VPL | CC | TP | airplane | bench | dresser | car | chair | display | lamp | speaker | rifle | sofa | table | phone | vessel | all |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 2.07 | 9.69 | 9.07 | 4.91 | 5.01 | 8.72 | 5.94 | 11.49 | 2.80 | 9.05 | 10.14 | 4.36 | 4.41 | 6.74 |
| ✓ | | | 2.33 | 6.69 | 7.73 | 3.40 | 4.94 | 8.51 | 7.81 | 9.24 | 2.48 | 7.28 | 5.88 | 5.48 | 3.76 | 5.81 |
| ✓ | ✓ | | 2.06 | 5.45 | 6.58 | 2.83 | 4.24 | **5.11** | 6.31 | 9.57 | 2.02 | 5.65 | 6.56 | 2.99 | 3.17 | 4.81 |
| | | ✓ | 1.94 | 8.17 | 6.42 | 4.17 | 3.96 | 6.78 | **5.24** | 7.34 | 2.59 | 7.65 | 6.21 | 4.28 | 3.95 | 5.29 |
| ✓ | | ✓ | 1.58 | 3.97 | **4.66** | 2.92 | **3.24** | 5.48 | 5.36 | **6.15** | 1.82 | 5.19 | **3.55** | **2.59** | 3.23 | 3.83 |
| ✓ | ✓ | ✓ | **1.52** | **3.94** | 5.19 | **2.76** | 3.42 | 5.15 | 5.37 | 6.38 | **1.50** | **4.87** | 3.64 | 2.61 | **2.85** | **3.78** |

Table 8. Evaluation using CD. Points are uniformly sampled from surfaces. This table corresponds to Table 2.

| VPL | CC | TP | airplane | bench | dresser | car | chair | display | lamp | speaker | rifle | sofa | table | phone | vessel | all |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 2.39 | 9.52 | 4.61 | 2.66 | 5.17 | 12.29 | 4.74 | 3.86 | 3.96 | 8.32 | 5.30 | 5.93 | 3.92 | 5.59 |
| ✓ | | | 2.56 | 5.93 | 2.04 | 1.05 | 3.88 | 6.77 | 6.47 | 3.20 | 3.02 | 6.29 | 3.77 | 3.39 | 2.79 | 3.93 |
| ✓ | ✓ | | 2.24 | 4.34 | 1.82 | 0.76 | 3.07 | 4.37 | 5.58 | **2.51** | 2.94 | 4.38 | 3.49 | 2.08 | 2.85 | 3.11 |
| | | ✓ | 2.21 | 8.04 | 2.51 | 2.66 | 4.81 | 5.35 | **4.58** | 3.23 | 3.54 | 7.70 | 4.67 | 2.03 | 3.53 | 4.22 |
| ✓ | | ✓ | 1.74 | 3.74 | 1.84 | 0.72 | 2.47 | **4.02** | 4.69 | 2.72 | 2.60 | 4.20 | 2.97 | **1.50** | 2.52 | 2.75 |
| ✓ | ✓ | ✓ | **1.66** | **3.71** | **1.72** | **0.71** | **2.40** | 4.04 | 4.59 | 2.68 | **2.20** | **3.95** | **2.89** | 1.64 | **2.31** | **2.65** |

Table 9. Evaluation using CD. Points are uniformly sampled from volumes. This table corresponds to Table 2.

| VPL | CC | TP | airplane | bench | dresser | car | chair | display | lamp | speaker | rifle | sofa | table | phone | vessel | all |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 14.8 | 23.2 | **24.9** | **21.1** | **21.5** | **22.2** | 23.2 | **24.9** | 17.4 | 24.0 | **23.9** | 22.8 | 19.2 | **21.8** |
| ✓ | | | 14.8 | 23.6 | 29.3 | 22.4 | 22.3 | 25.0 | 24.4 | 29.2 | **16.9** | 26.1 | 24.7 | 27.1 | 19.4 | 23.5 |
| ✓ | ✓ | | 14.5 | 22.8 | 25.2 | 21.6 | 22.9 | 22.6 | **23.1** | 25.2 | 17.0 | **23.9** | 24.8 | **20.9** | **18.3** | **21.8** |
| | | ✓ | 14.7 | 23.1 | 27.0 | 26.0 | 23.3 | 22.3 | 24.0 | 28.2 | 17.3 | 24.2 | 25.5 | 26.9 | 19.7 | 23.3 |
| ✓ | | ✓ | 14.3 | 25.7 | 31.6 | 23.0 | 24.3 | 32.8 | 24.1 | 31.0 | 17.1 | 26.8 | 25.2 | 28.7 | 19.1 | 24.9 |
| ✓ | ✓ | ✓ | **14.2** | **22.5** | 30.7 | 24.3 | 23.0 | 30.2 | 24.1 | 30.2 | 17.0 | 24.3 | 24.9 | 31.5 | 19.0 | 24.3 |

Table 10. Evaluation using EMD. Points are uniformly sampled from surfaces. This table corresponds to Table 2.

| VPL | CC | TP | airplane | bench | dresser | car | chair | display | lamp | speaker | rifle | sofa | table | phone | vessel | all |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0.21 | 1.61 | 2.17 | 0.93 | 1.27 | 3.65 | 0.58 | 1.67 | 0.27 | 2.08 | 1.52 | 2.23 | 0.61 | 1.45 |
| ✓ | | | 0.20 | 0.92 | 1.07 | 0.44 | 0.94 | 1.12 | **0.53** | 1.32 | 0.20 | 1.25 | 1.00 | **0.45** | 0.48 | 0.76 |
| ✓ | ✓ | | 0.19 | 0.79 | 1.12 | **0.37** | 0.85 | 1.49 | 0.87 | **1.17** | 0.22 | 1.28 | 1.13 | 0.96 | 0.48 | 0.84 |
| | | ✓ | 0.20 | 1.17 | 1.31 | 0.92 | 1.20 | 1.36 | 0.63 | 1.57 | 0.26 | 1.90 | 1.30 | 0.71 | 0.57 | 1.01 |
| ✓ | | ✓ | **0.17** | **0.66** | 1.08 | 0.42 | 0.82 | 1.01 | 0.65 | 1.35 | 0.19 | 1.25 | **0.84** | 0.47 | 0.48 | 0.72 |
| ✓ | ✓ | ✓ | **0.17** | **0.66** | **1.06** | 0.41 | **0.78** | **1.00** | 0.62 | 1.35 | **0.18** | **1.08** | 0.85 | 0.50 | **0.45** | **0.70** |

Table 11. Evaluation using EMD. Points are uniformly sampled from volumes. This table corresponds to Table 2.

Then, the gradient of $x_i$ is

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x_i} &= \frac{\partial \mathcal{L}}{\partial p_i}\frac{\partial p_i}{\partial x_i} + \frac{\partial \mathcal{L}}{\partial p_{i+1}}\frac{\partial p_{i+1}}{\partial x_i} \\ &= g_i(p_{i-1} - p_i) + g_{i+1}(p_i - p_{i+1}) \\ &= (g_i^p)^{\text{right}}. \end{aligned} \tag{6}$$

In the case where pixel $i$ moves to the left, we can compute the


Figure 10. Examples of thirteen categories on the ShapeNet dataset by multi-view training ($N_v$ = 20) without texture prediction. Panels (a–e) are the same as in Figure 4.


Figure 11. Examples of thirteen categories on the ShapeNet dataset by multi-view training ($N_v$ = 20) with texture prediction. Panels (a–e) are the same as in Figure 4.

gradient in a similar manner. Thus,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x_i} &= \frac{\partial \mathcal{L}}{\partial p_i}\frac{\partial p_i}{\partial x_i} + \frac{\partial \mathcal{L}}{\partial p_{i-1}}\frac{\partial p_{i-1}}{\partial x_i} \\ &= g_i(p_i - p_{i+1}) + g_{i-1}(p_{i-1} - p_i) \\ &= (g_i^p)^{\text{left}}. \end{aligned} \tag{7}$$

The problem is whether to use $(g_i^p)^{\text{right}}$ or $(g_i^p)^{\text{left}}$. When $x_i$ moves to the right, the decrease in $\mathcal{L}$ is proportional to $(d)^{\text{right}} = -(g_i^p)^{\text{right}}$. When $x_i$ moves to the left, the decrease in $\mathcal{L}$ is proportional to $(d)^{\text{left}} = (g_i^p)^{\text{left}}$. We define the gradient differently according to the following three cases.

- When $\max((d)^{\text{right}}, (d)^{\text{left}}) < 0$, the loss increases by mov-

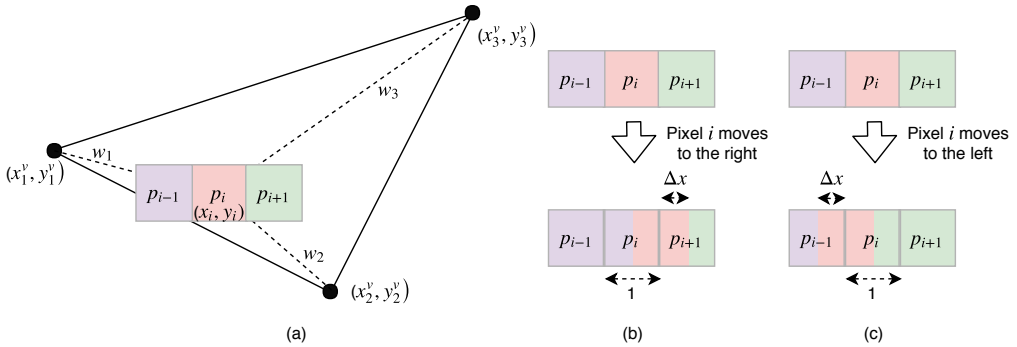


Figure 12. Our assumptions on the differentiation of a renderer.

| VPL | CC | TP | IoU | $CD_s$ | $CD_v$ | $EMD_s$ | $EMD_v$ |
|---|---|---|---|---|---|---|---|
| | | | .403 | 6.74 | 5.59 | **21.8** | 1.45 |
| ✓ | | | .490 | 5.81 | 3.93 | 23.5 | 0.76 |
| ✓ | ✓ | | .505 | 4.81 | 3.11 | **21.8** | 0.84 |
| | | ✓ | .434 | 5.29 | 4.22 | 23.3 | 1.01 |
| ✓ | | ✓ | .508 | 3.83 | 2.75 | 24.9 | 0.72 |
| ✓ | ✓ | ✓ | **.513** | **3.78** | **2.65** | 24.3 | **0.70** |

Table 12. Comparison of IoU, CD and EMD. This table is summary of Table 2, 8, 9, 10, and 11. Subscripts $s$ and $v$ mean that points are uniformly sampled from surfaces and volumes respectively. VPL: proposed view prior learning. CC: class conditioning in the discriminator. TP: texture prediction. CD and EMD is lower is better.

| Discriminator | Optimization | Texture | IoU |
|---|---|---|---|
| None | - | | .403 |
| Table 1 (c) | Gradient reversal | | .505 |
| Table 1 (c) | Iterative | | .514 |
| Table 1 (d) | Gradient reversal | | *.403** |
| Table 1 (d) | Iterative | | *.403** |
| None | - | ✓ | .434 |
| Table 1 (c) | Gradient reversal | ✓ | .513 |
| Table 1 (c) | Iterative | ✓ | .510 |
| Table 1 (d) | Gradient reversal | ✓ | *.434** |
| Table 1 (d) | Iterative | ✓ | *.434** |

Table 13. Evaluation of the discriminators in Table 1 (c–d). *No meaningful improvement was observed by tuning $\lambda_d$.

| Supervision | $N_v$ | Internal pressure | IoU |
|---|---|---|---|
| Single-view | 1 | | .387 |
| Single-view | 1 | ✓ | .403 |
| Multi-view | 20 | | .648 |
| Multi-view | 20 | ✓ | .652 |

Table 14. Effect of internal pressure loss.

ing the pixel $i$. Therefore, in this case, we define $\frac{\partial \mathcal{L}}{\partial x_i} = 0$.

- When $0 \leq \max((d)^{\text{right}}, (d)^{\text{left}})$ and $(d)^{\text{left}} < (d)^{\text{right}}$, the loss decreases more by moving pixel $i$ to the right. In this case, we define $\frac{\partial \mathcal{L}}{\partial x_i} = (g_i^p)^{\text{right}}$.
- When $0 \leq \max((d)^{\text{right}}, (d)^{\text{left}})$ and $(d)^{\text{right}} < (d)^{\text{left}}$, it is better to move pixel $i$ to the left. In this case, we define $\frac{\partial \mathcal{L}}{\partial x_i} = (g_i^p)^{\text{left}}$.

The gradient with respect to $y_i$ is defined in a similar way.

| Loss function | IoU |
|---|---|
| Multi-scale cosine distance (Eq. 3, $N_s = 5$) | .575 |
| Multi-scale cosine distance (Eq. 3, $N_s = 1$) | .567 |
| Interesection over union (Eq. 4) | .552 |
| Sum of squared error | **.579** |

Table 15. Comparison of silhouette loss functions.

## A.9. Experimental settings

### A.9.1 Optimizer

We used the Adam optimizer [19] in all experiments. In our ShapeNet experiments, the Adam parameters were set to $\alpha = 4e-4, \beta_1 = 0.5, \beta_2 = 0.999$. In the PASCAL experiments, the parameters were set to $\alpha = 2e-5, \beta_1 = 0.5, \beta_2 = 0.999$. The batch size is set to 64 in our ShapeNet experiments, and set to 16 in our PASCAL experiments.

### A.9.2 Encoder, decoder and discriminator

We used the ResNet-18 architecture [10] for the encoders in all experiments. The weights of the encoder were randomly initialized in the ShapeNet experiments. The weights were initialized using the weights of the pre-trained model from [10] in the PASCAL experiments.

We generated a 3D shape and texture image by deforming a pre-defined cube. The number of vertices on each face of the cube is $16 \times 16$, and the vertices on the edge of the cube are shared within two faces. The total number of vertices is 1352. The size of a texture image on each face is $64 \times 64$ pixels. The shape decoder outputs the coordinates of the vertices of this cube, and the texture decoder outputs six texture images. Figures 13 and 14 show the architecture of the shape decoders used in the ShapeNet

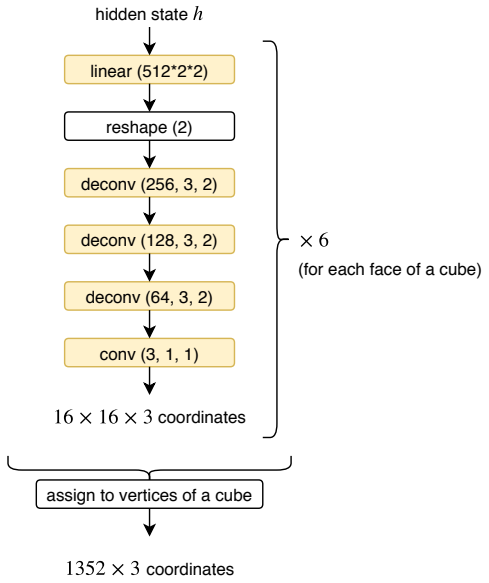


Figure 13. Architecture of the shape decoder used in the ShapeNet experiments. The $16 \times 16$ vertices on each face of the cube are separately generated, and they are merged into 1352 vertices. The dimension of the input vector is 512. All linear and deconvolution layers except the last one are followed by ReLU nonlinearity.

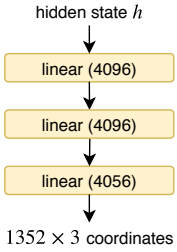


Figure 14. Architecture of the shape decoder used in the PASCAL experiments. The dimension of the input vector is 512. All linear layers except the last one are followed by ReLU nonlinearity.

and PASCAL experiments. Figure 15 shows the architecture of the texture decoder used in all experiments.

Figures 16 and 17 show the architectures of the discriminators in the ShapeNet and PASCAL experiments.

The layers used in the architecture figures are as follows:

- linear$(a)$ is an affine transformation layer. $a$ is the number of feature maps.
- conv$(a, b, c)$ is a 2D convolution layer. The number of feature maps is $a$, the kernel size is $b \times b$, and the stride size is $c \times c$.
- deconv$(a, b, c)$ is a 2D deconvolution layer. The number of feature maps is $a$, the kernel size is $b \times b$, and the stride size is $c \times c$.
- reshape$(a)$ reshapes a vector into feature maps of size $a \times a$.
- tile$(a)$ tiles a vector into feature maps of size $a \times a$.

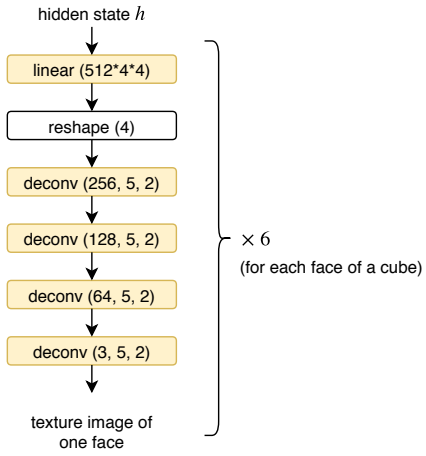


Figure 15. Architecture of the texture decoder used in all experiments. A texture image of size $64 \times 64$ is generated separately for each face of a cube. The input vector has 512 dimensions. All linear and deconvolution layers except the last one are followed by Batch Normalization [12] and ReLU nonlinearity.

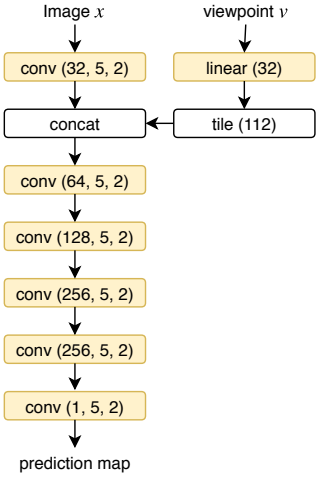


Figure 16. The architecture of the discriminator used in the ShapeNet experiments. The size of the input image is $224 \times 224$. A viewpoint is represented by a three-dimensional vector of the elevation, azimuth, and distance to the object. Spectral Normalization [25] is applied to all convolution and linear layers. All convolution layers except the last one are followed by LeakyReLU nonlinearity.

- concat$(\cdot)$ stacks two feature maps.

### A.9.3 Other hyperparameters

Table 16 and Table 17 show the number of training iteration and the weights of loss terms in ShapeNet and PASCAL experiments.

| Training type | $N_v$ | Texture prediction | View prior learning | #training iteration | $\lambda_c$ | $\lambda_d$ | $\lambda_p$ |
|---|---|---|---|---|---|---|---|
| single-view | 1 | | | 50000 | - | - | 0.0001 |
| single-view | 1 | ✓ | | 50000 | 0.5 | - | 0.0001 |
| single-view | 1 | | ✓ | 100000 | - | 0.2 | 0.0001 |
| single-view | 1 | ✓ | ✓ | 100000 | 0.5 | 2 | 0.0001 |
| multi-view | 2, 3, 5, 10, 20 | | | $25000N_v$ | - | - | 0.0001 |
| multi-view | 2, 20 | ✓ | | $25000N_v$ | 0.1 | - | 0.0001 |
| multi-view | 2, 3, 5, 10, 20 | | ✓ | $50000N_v$ | - | 0.03 | 0.0001 |
| multi-view | 2, 20 | ✓ | ✓ | $50000N_v$ | 0.1 | 0.3 | 0.0001 |

Table 16. Hyperparameters used in the ShapeNet experiments.

| Training type | Texture prediction | View prior learning | #training iteration | $\lambda_c$ | $\lambda_d$ | $\lambda_p$ |
|---|---|---|---|---|---|---|
| category-agnostic | | | 15000 | - | - | 0.00003 |
| category-agnostic | ✓ | | 15000 | 0.01 | - | 0.00003 |
| category-agnostic | | ✓ | 50000 | - | 2 | 0.00003 |
| category-agnostic | ✓ | ✓ | 250000 | 0.01 | 0.5 | 0.00003 |
| category-specific | | | 5000 | - | - | 0.00003 |
| category-specific | ✓ | | 5000 | 0.01 | - | 0.00003 |
| category-specific | | ✓ | 40000 | - | 2 | 0.00003 |
| category-specific | ✓ | ✓ | 80000 | 0.01 | 0.5 | 0.00003 |

Table 17. Hyperparameters used in the PASCAL experiments.

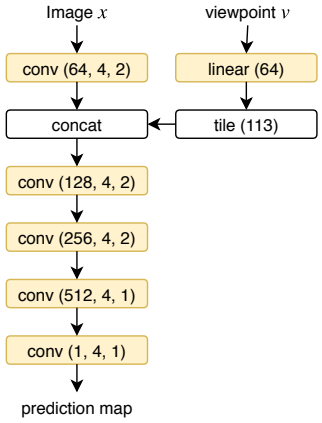


Figure 17. Architecture of the discriminator used in the PASCAL experiments. The size of the input image is $224 \times 224$. A viewpoint is represented by a $3 \times 3$ rotation matrix. All convolution layers except the last one are followed by LeakyReLU nonlinearity.