# Vision Transformer for NeRF-Based View Synthesis from a Single Input Image Supplementary Materials

## 1. Video Results

We include video results as a webpage (`https://cseweb.ucsd.edu/~viscomp/projects/VisionNeRF/supplementary.html`). In each video, we compare the input, SRN [11], PixelNeRF [13], ours and ground-truth. The renderings of SRN and PixelNeRF are provided by the authors of PixelNeRF on request.

## 2. Network Architecture

**Transformer encoder.** We adopt the pretrained ViT-b16 model from Wightman [12] as our transformer encoder. The transformer encoder has 12 layers ($J$=12), where each layer uses the LayerNorm [1] for normalization and GELU [4] for activation. The positional embedding is resized to the same size as the input image (e.g., $128 \times 128$ for the ShapeNet dataset, and $64 \times 64$ for the NMR dataset).

**Convolutional decoder.** We provide the architecture details of our convolutional decoder in Table 1. Our convolutional decoder takes the tokens $f^3$, $f^6$, $f^9$, and $f^{12}$ [9] as input and generate multi-level features $\mathbf{W}_G^3, \mathbf{W}_G^6, \mathbf{W}_G^9$, $\mathbf{W}_G^{12}$, as shown in Fig. 4 of the main paper. We then resize all feature maps $\mathbf{W}_G$ to the size of $\frac{H}{2} \times \frac{W}{2}$ via bilinear interpolation. Finally, we adopt a two-layer CNN module (see Table 2) to generate the global feature representation $\mathbf{W}_G'$.

**2D CNN $\mathcal{G}_L$.** We use the ResBlocks in Fig. 1 to generate the local feature representation $\mathbf{W}_L$. The local feature $\mathbf{W}_L$ and global feature $\mathbf{W}_G'$ are then concatenated as our hybrid feature representation $\mathbf{W}$.

**NeRF MLP.** We utilize the hierarchical rendering [8] to include detailed geometry and appearance. We use 6 ResNet blocks with width 512 for both coarse and fine stages to process the global and local features.

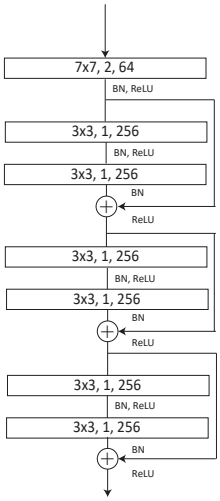


Figure 1. **Illustration of the ResBlocks.** We use three ResBlocks as our 2D CNN module to extract local feature representation from the input image. The numbers in each layer denote the kernel size, stride, and output channels, respectively. BN denotes BatchNorm, and ReLU is the nonlinear activation function.

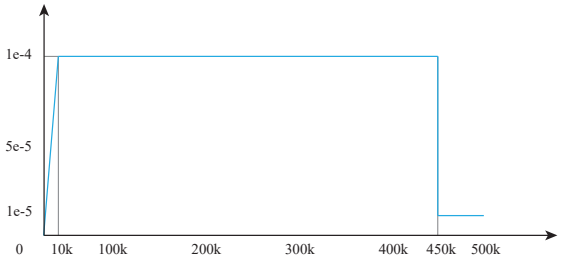


Figure 2. **Illustration of the learning rate strategy for the MLP module.** We first linearly increase the learning rate to $1e-4$ for the first 10k step. Then, we set a learning rate decay of scale 0.1 at 450k steps.

## 3. Implementation Details

**Learning rate.** We set the initial learning rate to be $10^{-4}$ for the MLP and $10^{-5}$ for ViT and the 2D CNN module.

| $\#j$ | layer | kernel | stride | dilation | in | out | activation | input |
|---|---|---|---|---|---|---|---|---|
| 3 | Conv0-0 | $1 \times 1$ | 1 | 1 | 768 | 96 | N/A | $f^3$ |
| | TransConv0-1 | $4 \times 4$ | 4 | 1 | 96 | 96 | N/A | Conv0-0 |
| | Conv0-2 | $3 \times 3$ | 1 | 1 | 96 | 512 | N/A | TransConv0-1 |
| 6 | Conv1-0 | $1 \times 1$ | 1 | 1 | 768 | 192 | N/A | $f^6$ |
| | TransConv1-1 | $2 \times 2$ | 2 | 1 | 192 | 192 | N/A | Conv1-0 |
| | Conv1-2 | $3 \times 3$ | 1 | 1 | 192 | 512 | N/A | TransConv1-1 |
| 9 | Conv2-0 | $1 \times 1$ | 1 | 1 | 768 | 384 | N/A | $f^9$ |
| | Conv2-1 | $3 \times 3$ | 1 | 1 | 384 | 512 | N/A | Conv2-0 |
| 12 | Conv3-0 | $1 \times 1$ | 1 | 1 | 768 | 768 | N/A | $f^{12}$ |
| | Conv3-1 | $3 \times 3$ | 2 | 1 | 768 | 768 | N/A | Conv3-0 |
| | Conv3-2 | $3 \times 3$ | 1 | 1 | 768 | 512 | N/A | Conv3-1 |

Table 1. **Architecture of the convolutional decoder.** $\mathbf{Conv}$ denotes convolution layer. $\mathbf{TransConv}$ denotes transposed convolution layer.

| Layer | kernel | stride | dilation | in | out | activation | input |
|---|---|---|---|---|---|---|---|
| Conv0 | $3 \times 3$ | 1 | 1 | 1024 | 512 | ReLU | $\mathbf{W}_G$ |
| Conv1 | $3 \times 3$ | 1 | 1 | 512 | 256 | ReLU | conv0 |

Table 2. **Architecture of the last two layers for generating global feature representation $\mathbf{W}'_G$.**

To improve training stability, we use a warm-up schedule to increase the learning rate linearly from 0 for the first 10k steps. We decay the learning rate by a factor of $0.1$ at 450k steps. Our learning rate schedule for the NeRF MLP is plotted in Fig. 2.

**Inference speed.** We ran 300 forward steps and averaged them to obtain the per-step inference time. PixelNeRF takes $1.35s$ and ours takes $1.7s$. Our method has slightly more latency.

# 4. Experiment Configurations

We provide the detailed experiment setups in Sec. 4 of the main paper.

## 4.1. Category-specific View Synthesis.

**SRN [11] and PixelNeRF [13]:** We obtain the PSNR and SSIM from Table 2 in [13]. We use the pre-generated results provided by the authors of [13] (on request) to calculate the LPIPS score.

**CodeNeRF [5]:** We obtain the PSNR and SSIM of the unposed input from Table 2 in [5]. As the pre-generated results are not available, and the source code of CodeNeRF does not provide optimization stages for unposed inputs, we are not able to calculate the LPIPS score.

**FE-NVS [3]:** We obtain the PSNR and SSIM of the unposed input from Table 1 in [3]. We obtain the LPIPS score from the authors on request. As FE-NVS does not provide source code to reproduce any qualitative results, we crop the high-resolution images from their paper to show the comparison in Fig. 6 of the main paper.

Note that all the methods except CodeNeRF uses view 64 as input, while CodeNeRF uses view 82 as input for evaluation in their paper. As the source code for CodeNeRF does not include unposed inference, we are not able to generate the full evaluation using view 64.

## 4.2. Category-agnostic View Synthesis

Similar to category-specific view synthesis, we obtain the numbers of SRN [11] and PixelNeRF [13] from Table 4 of [13]. We obtain the numbers of FE-NVS [3] from Table 4 of [3]. Qualitative results for SRN and PixelNeRF are generated using the pre-generated results from [13]. The detailed categorical numbers of SRT [10] and FWD [2] are provided by their authors. For qualitative results, we obtain the images from SRT project website and from the author of FWD. Note that SRT does not provide the full renderings of the dataset.

## 4.3. View Synthesis on Real Images

We use the real car images provided by PixelNeRF and the Stanford Cars dataset [7] for evaluation. We remove the background using the PointRend [6] segmentation and resize the images to $128 \times 128$ resolution. We assume the input camera pose is an identity matrix and apply rotation matrices to simulate $360^\circ$ views. We use the source code and pre-trained model of PixelNeRF to generate the results for PixelNeRF, where we synthesize the renderings at the same camera poses.

# 5. Additional Results

## 5.1. Generalization on Unseen objects.

For unseen categories, we feed an image of a real mug to the category-agnostic model as shown in Fig. 3. While the training data do not include mugs, our method is able to predict reasonable novel views.

(a) Input (b) View 1 (c) View 2

Figure 3. **Results for unseen real data.** We run our category-agnostic model on a real image (a) and render two viewpoints from the right (b) and left (c). Our method is able to predict a reasonable geometry even though mugs are not presented in the training data.

### 5.2. Results on Category-specific and Category-agnostic View Synthesis

We include extra results on category-specific view synthesis in Fig. 4 and 5, results on category-agnostic view synthesis in Fig. 6-12. Note that we choose the target views where most pixels are not visible in the input view to better compare the rendering quality of each method on occluded regions. The video results (in GIF format) are also provided on website (`https://cseweb.ucsd.edu/~viscomp/projects/VisionNeRF/supplementary.html`).

## References


[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Ang Cao, Chris Rockwell, and Justin Johnson. Fwd: Real-time novel view synthesis with forward warping and depth. *CVPR*, 2022.

[3] Pengsheng Guo, Miguel Angel Bautista, Alex Colburn, Liang Yang, Daniel Ulbricht, Joshua M. Susskind, and Qi Shan. Fast and explicit neural view synthesis. In *WACV*, 2022.

[4] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[5] Wonbong Jang and Lourdes Agapito. CodeNeRF: Disentangled neural radiance fields for object categories. In *ICCV*, 2021.

[6] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image segmentation as rendering. In *CVPR*, 2020.

[7] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[8] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[9] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021.

[10] Mehdi S. M. Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lucic, Daniel Duckworth, Alexey Dosovitskiy, Jakob Uszkoreit, Thomas Funkhouser, and Andrea Tagliasacchi. Scene Representation Transformer: Geometry-Free Novel View Synthesis Through Set-Latent Scene Representations. *CVPR*, 2022.

[11] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *NeurIPS*, 2019.

[12] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[13] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.


Figure 4. **Category-specific view synthesis results on cars.**


Figure 5. **Category-specific view synthesis results on chairs.**

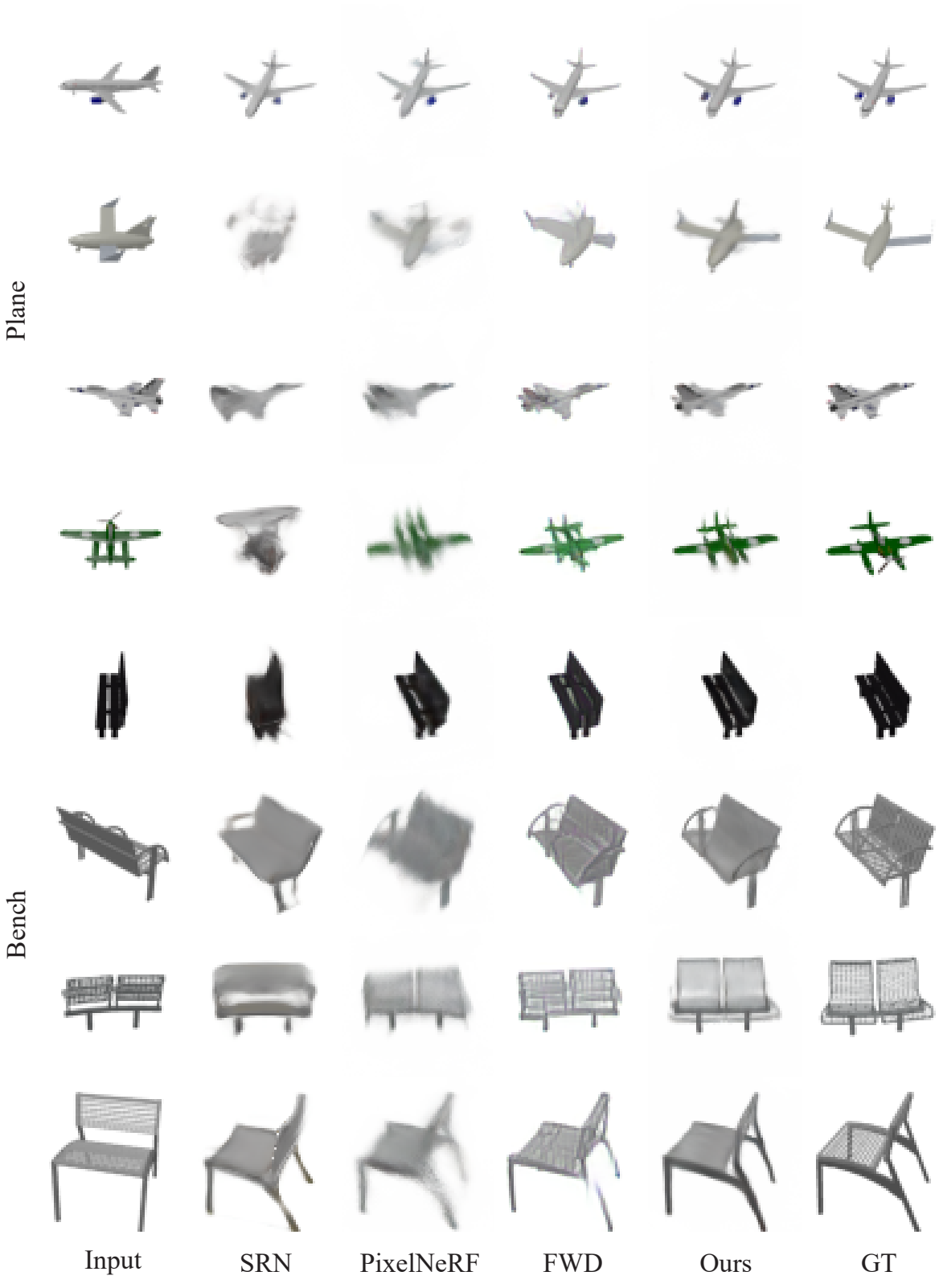


Figure 6. **Category-agnostic view synthesis results on the NMR dataset.**

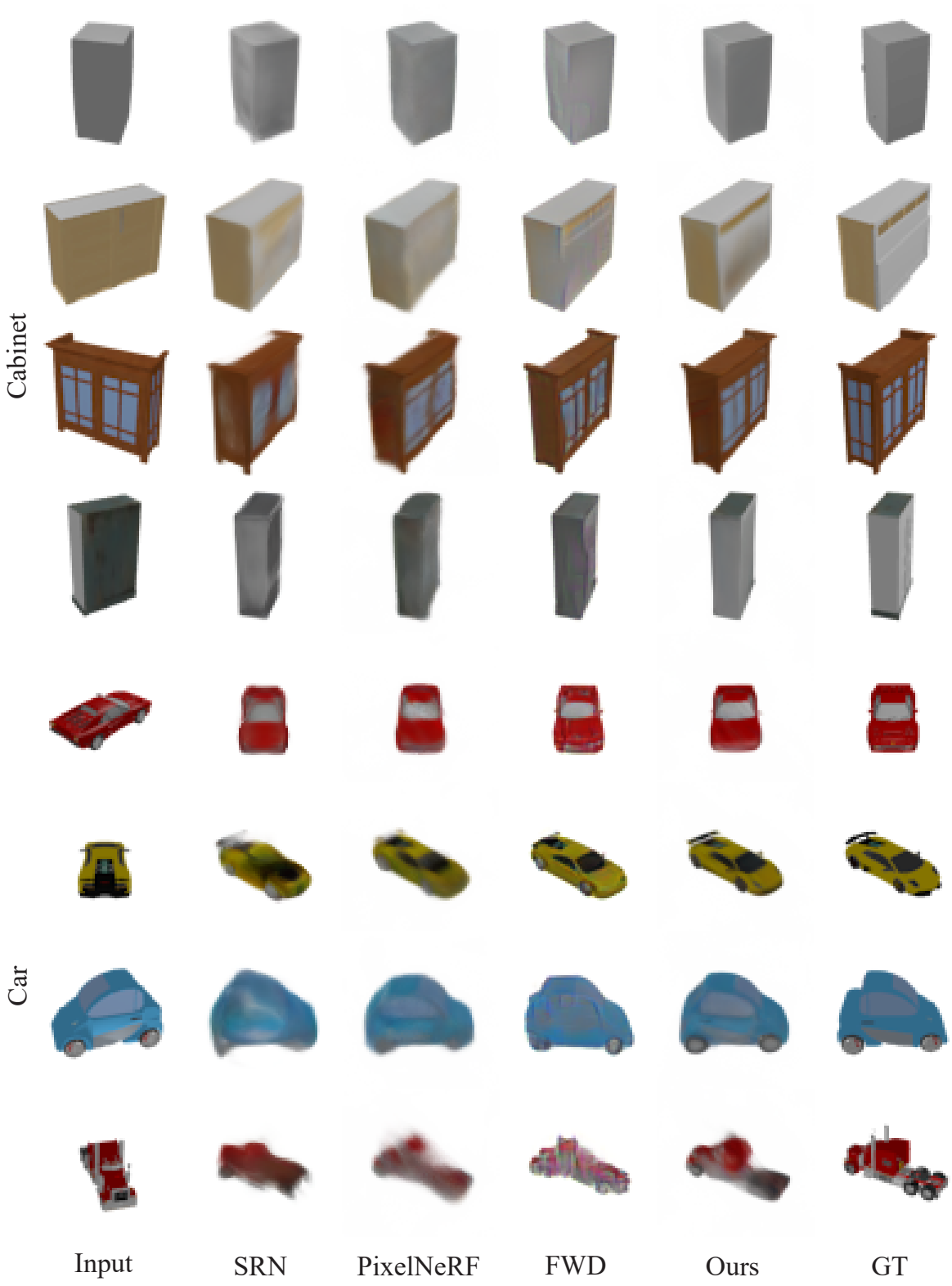


Figure 7. **Category-agnostic view synthesis results on the NMR dataset.**


Figure 8. **Category-agnostic view synthesis results on the NMR dataset.**

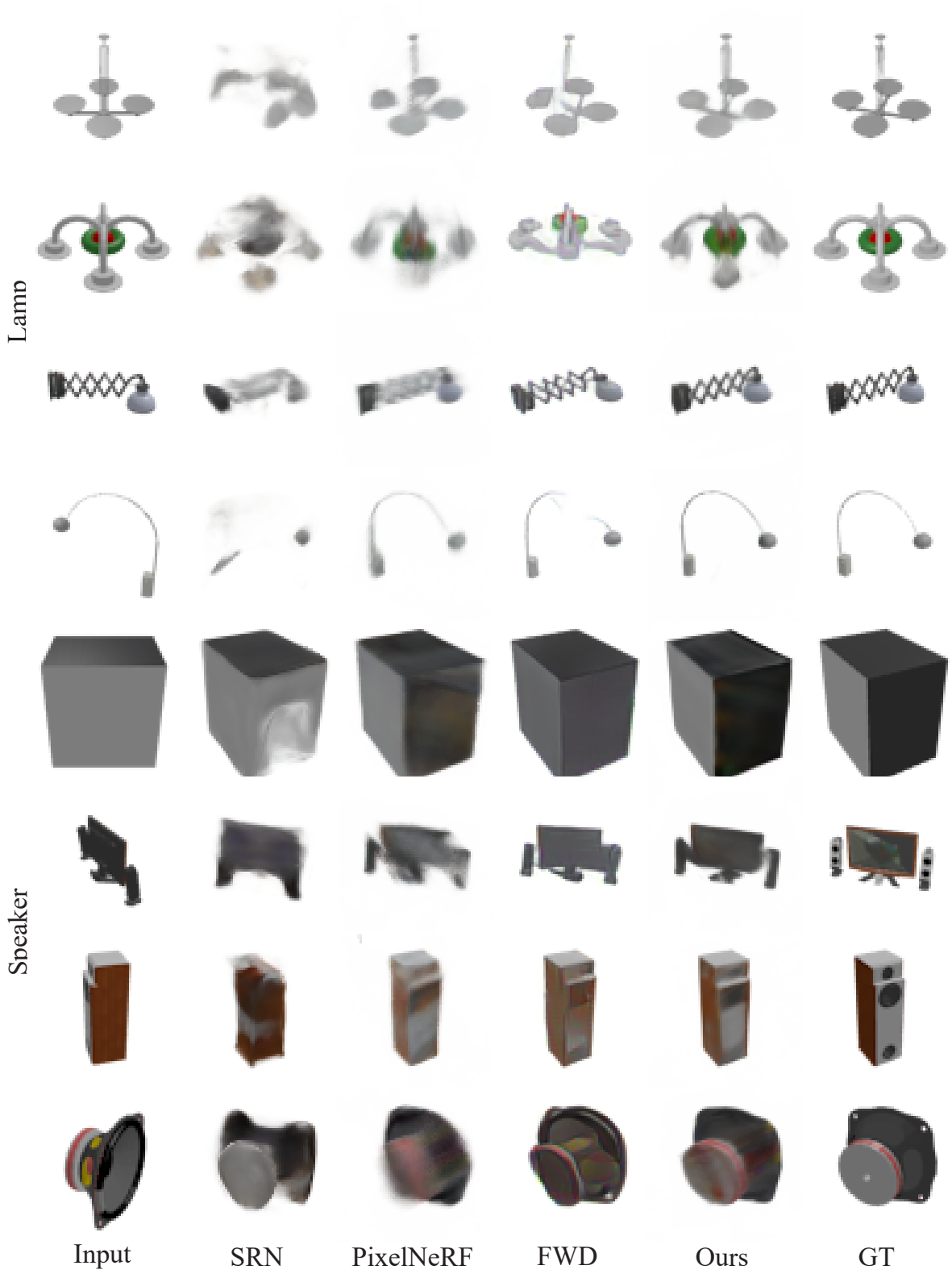


Figure 9. **Category-agnostic view synthesis results on the NMR dataset.**

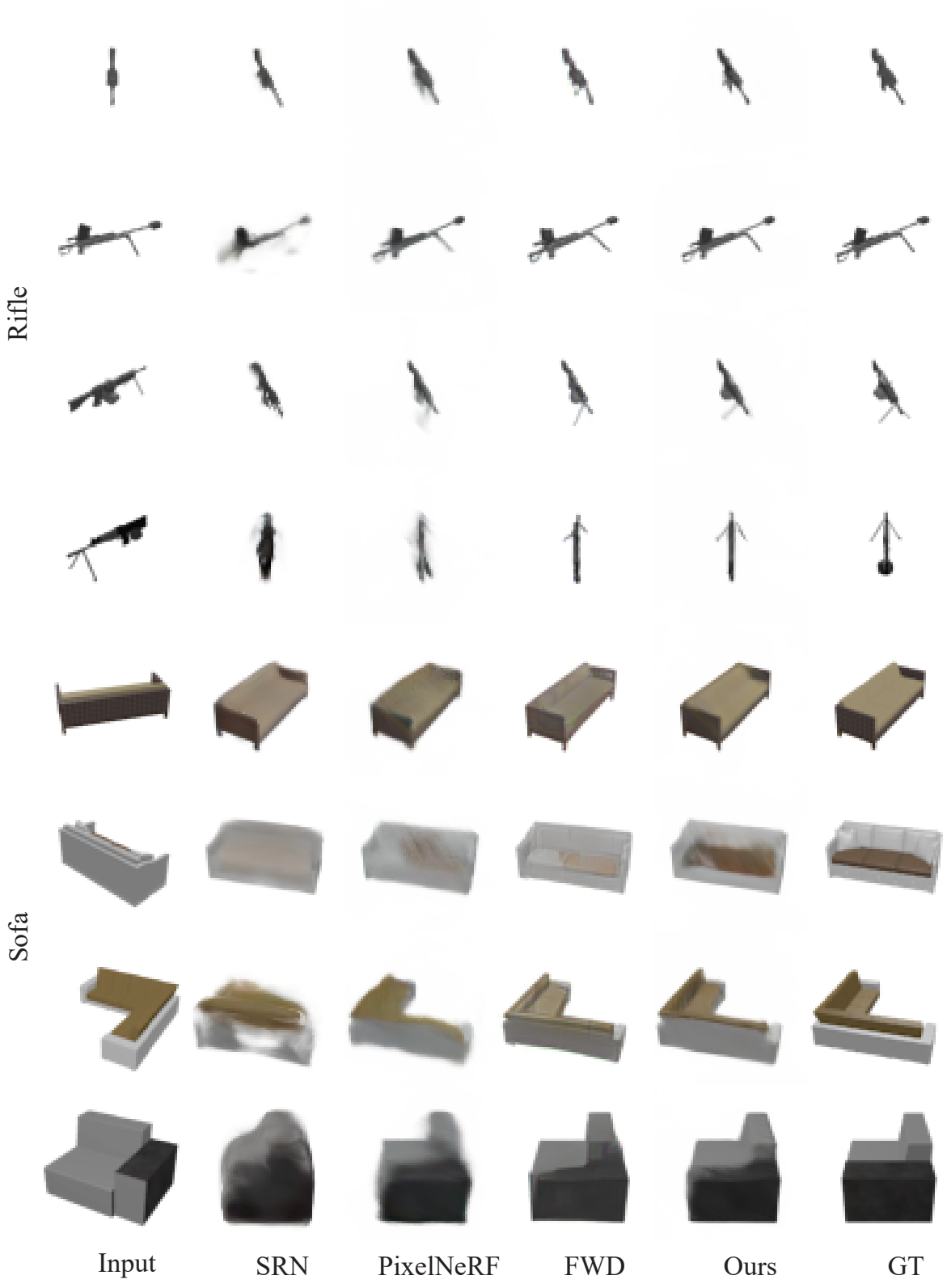


Figure 10. **Category-agnostic view synthesis results on the NMR dataset.**

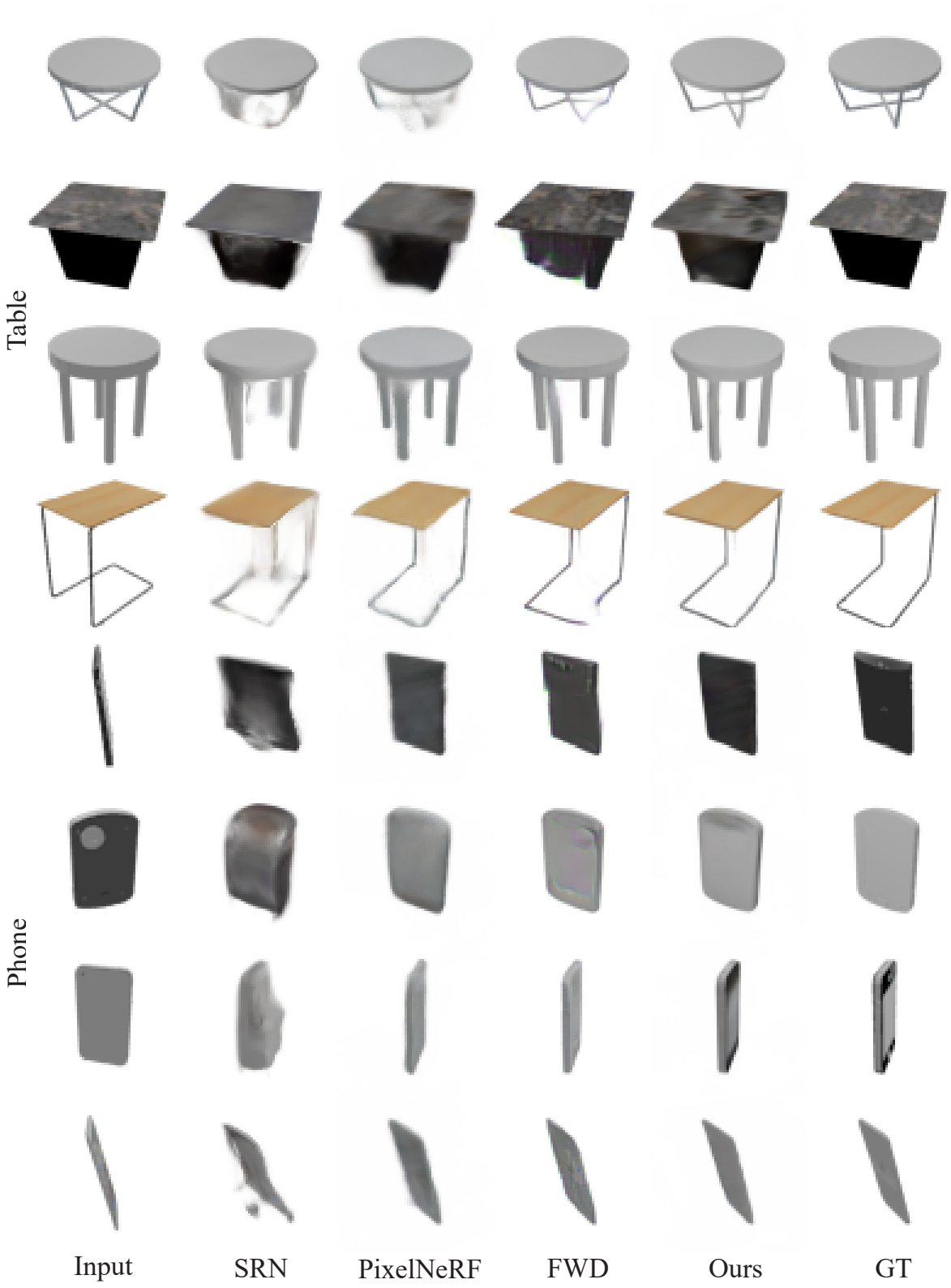


Figure 11. **Category-agnostic view synthesis results on the NMR dataset.**

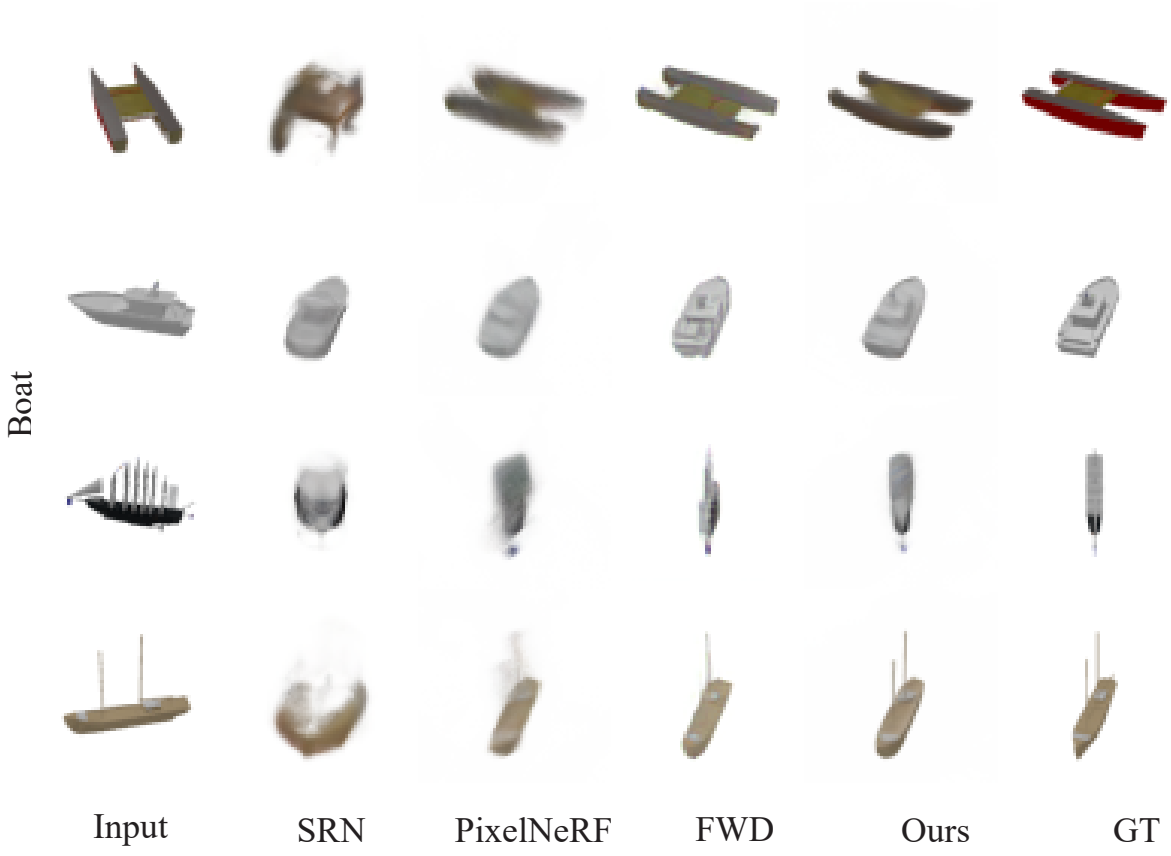


Figure 12. **Category-agnostic view synthesis results on the NMR dataset.**