FWD: Real-time Novel View Synthesis with Forward Warping and Depth

Ang Cao, Chris Rockwell, Justin Johnson
University of Michigan, Ann Arbor
{ancao,cnris,justincj}@umich.edu

We show more visualizations and comparisons in the supplementary and attached videos. Please watch the results videos for more results. We also provide implementation details, license of other methods and datasets, as well as other experiments results. See the following contents label for more information.

Contents

1. Model Architectures and Training details

As stated in the paper, our model consists of spatial feature network f, depth network d, view-dependent feature MLP ψ, neural point cloud renderer π, fusion transformer T and refinement module R. We show architecture details.

Spatial Feature Network f. The spatial feature network f contains 8 ResNet blocks, with output channels of 32, 32, 32, 64, 64, 64,64, 61 and no downsampling. Each ResNet block utilizes 3×3 /stride 1/padding 1 convolution followed by instance norm and ReLU. Similar to [1, 2], spectral normalization is utilized for each convolution for stable training. The input images are padded by 0 to fit network.

Depth Network d. We utilize a classical U-Net for depth refinement, consisting 8 downsampling blocks and 8 upsampling blocks. We pad constant zero to make the input have feasible shape. For PatchMatchNet [3] to estimate the initial depths, we follow the original pipeline, in which we downsample the input images into 4 scales and predict the depths in a coarse-to-fine manner.

View-dependent feature MLP ψ. The relative view direction change vector is first passed through a two-layer MLP without 16 and 32 output features perspectively to get a 32-dims feature embedding. Then this 32 dims feature embedding is concatenated with the original 64 dims feature vector and passed through another two-layer MLP with output 64 and 64 output features. We use ReLU as activation function following MLP and no normalization layers.

Neural point cloud renderer π. The point cloud renderer is implemented in Pytorch3D [4], which takes a point cloud and pose P and project it to a 150×200 feature map, where each pixel has 64-dims feature. The renderer fills zero with 64-dims for pixels which are invisible.

The blending weight α of the 3D point x for pixel l is

$$\alpha = 1 - \text{torch.clamp}(\sqrt{\frac{s}{r^2}}, \text{min} = 1e-3, \text{max} = 1.0), \tag{1}$$

where s is the Euclidean distance between point x's splatted center and pixel l; r is the radius of point x in NDC coordinate. We set $r = 1.5$ pixels in our experiments. To render the value of each pixel, we employ alpha-compositing to blend all feasible points. The rendered feature F_l of pixel l is:

$$F_l = \sum_{i=1}^{K} \alpha_i F_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{2}$$

where F_i, α_i is the feature and alpha of 3D point i. The rendering is based on points' depths and we blend the Top K points with the nearest depths. We use $K = 16$ in our paper, meaning we blend at most 16 points to get the results of one pixel.

We compute D_l, the depth of pixel l using the blending weights by:

$$D_l = \sum_{i=1}^{K} \alpha_i d_i \tag{3}$$

where d_i is the depth of point i.

Fusion transformer T. The Key and Value are feature vectors from multiple views, with the shape as NView$\times N \times C$,

where NView is the number of input views (3 in our experiments), N is the batch size, which is $H \times W$, and C is the feature dimension, which is 64. The query is a learnable token with the shape as $1 \times C$, and expanded to N batch. The transformer is a multi-head attention with 4 heads, projecting key into 16 dims embedding and output 64 dims vectors.

Refinement module R. We use a ResNet decoder as our refinement module, which consists 8 ResNet blocks, in which we downsample at the 3rd layers and upsample at 6th and 7th layers. The output feature dims are 64, 128, 256, 256, 128, 128, 128, 3.

Two-stage training for FWD-U. We find that a two-stage training scheme would slightly improve the performance of FWD-U model (0.4dB). In stage one, we first train the depth networks by constructing RGB point clouds for each input view and projecting them to the target view via differentiable point cloud rendering. We directly aggregate all the point clouds from various input views and get the rendered RGB images at target views. The depth network is trained by photometric loss between rendered views and target images. This step works as a simple unsupervised MVS scheme and gives a proper initialization of the depth network. We then train the whole model with the initialized depth network in stage two. This two-stage training scheme's intuition is that jointly training depth network and other components from scratch are unstable, and our first training stage could give a initialization for depth network.

2. License Discussions

We discuss the licenses of assets.

2.1. Dataset

ShapeNet Dataset [?] . We conduct our experiments on the ShapeNet dataset and we cite the paper [**?**] as required by the author. More specifically, we download the data from NMR, which is hosted by DVR [5] authors.

DTU MVS Dataset [6]. We conduct our experiments on the DTU MVS dataset. This dataset doesn't include any license. On the other hand, we cite the paper [6] which is required by the paper.

2.2. Methods

PixelNeRF [7] . We evaluate the official code of PixelNeRF [7] for comparison. The code is hosted in the github page https://github.com/sxyu/pixel-nerf, which uses the BSD 2-Clause "Simplified" License.

IBRNet [8] . We use the official code of IBRNet [8] for comparison, which is hosted at https://github.com/googleinterns/IBRNet. This project has the Apache License 2.0.

MVSNeRF [9]. We use the official code of MVSNeRF [9] for comparison, which is hosted at https://github.com/apchenstu/mvsnerf with MIT License.

SynSin [2]. The code of SynSin [2] is hosted at https://github.com/facebookresearch/synsin with Copyright (c) 2020, Facebook All rights reserved.

Stable View Synthesis (SVS) [10]. We use the code hosted at https://github.com/isl-org/StableViewSynthesis for evaluation, with MIT License and Copyright (c) 2021 Intel ISL (Intel Intelligent Systems Lab).

DeepBlending [11]. We implement it according to the code at https://github.com/Phog/DeepBlending. This work is under Apache License.

PixelNeRF-DS[12]. We use the number reported in [12].

Pytorch3D. We use the code from Pytorch3D [**?**]: https://github.com/facebookresearch/pytorch3d for our differentiable renderer. The code is licensed with BSD 3-Clause License.

3. User Study Details

As detailed in the paper, we conduct user study to evaluate perceptual quality of synthesized images. We provide more information about user study here.

We employ the standard A/B test paradigm as our user study format, which asks workers to select the closest result to a ground truth image between two competing results. Results of method A and B and ground truth target view are available during test. All views in the test set (690 views in total) are evaluated and each view is judged by 3 workers.

All tests are conducted using thehive.ai, a website similar to Amazon Mechanical Turk. Workers were given instructions and examples of tests, and then they were given a test to identify whether they understand the task. Only workers passing the test were allowed to conduct A/B test. Three images of the same view are shown in the test, where the first image is the ground truth target views and the rest two images are synthesized images. Workers are asked to select "left" or "right" image to indicate their preference. Results of method A and B are randomly placed for every test for fairness. We show the instructions and examples below.

4. Additional Results

Again, please see attached videos for comparison between ours and other methods.

4.1. Time statistics.

We show times spent on each component of FWD-D model during a single forward pass in Table 1. As shown in the Table, only 30 percent and 12 percent of times are spent on the Rendering process and Fusion process, indicating that our renderer and Transformer are highly-efficient.

Moreover, we compare the time distributions of our method with PixelNeRF and IBRNet in Table 2.

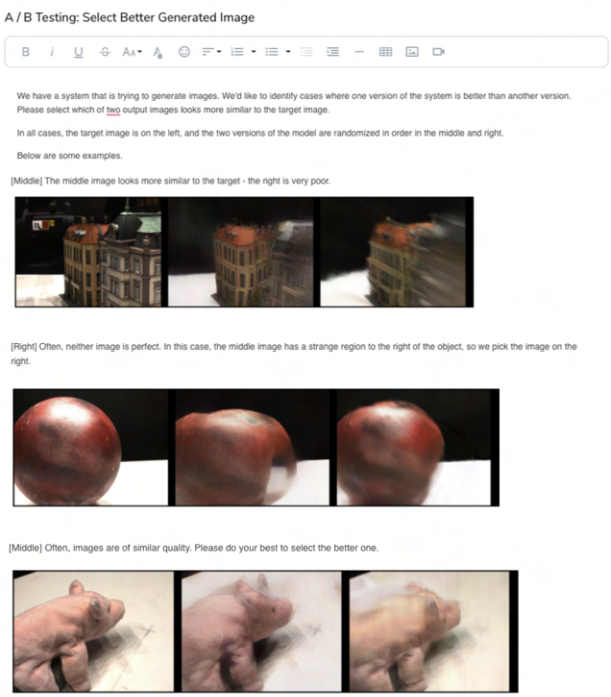

Figure 1. **Instructions of A/B test.** We show the instructions of A/B test used in our paper.

Table 1. **Inference time of FWD-D.** We count times used for each stage in single forward pass with a batch size of 24.

	Constr.	Render	Fusion	Refine.	Total
Time (ms)	40.39	172.57	70.11	302.59	585.66
Percent	6.90	29.47	11.97	61.67	100.00

Table 2. **Inference time between different methods.** We study the time distributions for each method to render a total scene (46 views) for PixelNeRF, IBRNet and FWD. Feature Encoder refers feature extractions for input views, including feature encoding for PixelNeRF and IBRNet, feature encoding and depth regression for FWD. Renderer refers to the time spent for synthesizing each view.

model	Feature Encoder	Renderer	Total
PixelNeRF	0.696s	30min	30min
IBRNet	0.87s	163.2s	173s
FWD	0.085s	1.10s	1.19s

4.2. Other design ablations.

In Table 3, we conduct more ablations for fusion and view-dependent feature MLP on FWD-D model.

IBRNet's Multi-view Feature Aggregation (IBR MFA). We replace our's fusion Transformer with IBRNet's multi-view feature aggregation mechanism. In IBR MFA, feature vectors are fused using variance as the global pooling operator. The per-element mean and variance are computed from feature vectors to capture global information and then concatenated with each feature vector, which contains local information. A small shared MLP is used to integrate both local and global information, resulting in multi-view aware feature vectors and corresponding weights. Another MLP is used to map the weighted feature vector into a final feature vector. See IBRNet's paper for more information.

View dependence feature without depth (VD w/o depth). Our current view-dependent feature MLP takes a 4-dimension feature vector as input, the first three of which is relative view change and the last of which is depth. In this ablation model, we only use the first three-dimension as inputs and ignore the depth information.

Table 3. **Ablation Studies.** We show more ablation study on FWD-D.

Model	PSNR	SSIM	LPIPS
Full model	21.98	0.791	0.208
w/o Transformer	20.95	0.748	0.241
w/o View dependence	21.16	0.769	0.212
IBR MFA	21.47	0.785	0.210
VD w/o depth	21.56	0.785	0.213

4.3. View number ablations.

In Table 4, we explore how the synthesis quality and speeds change with varying input view numbers. With increasing input view numbers, we can observe that our method can achieve higher quality while requiring more computations and times to process input views. In particular, the time spent on MVS module and fusion Transformer is significantly increased with increasing input views. However, our method is still much more efficient than other baseline methods with varying input view number.

4.4. Failure Cases.

We show failure cases for our method in Figure 2 for FWD model. One distinct failure pattern is the artifact cloud shown in the figure, which occasionally happens for several challenging viewpoints and scenes. Several reasons jointly cause this artifact. 1) At several challenging viewpoints which are very far away from input views, imperfect depth estimations would cause significant misalignments in the target view. Also, regions in the target view are not fully visible in inputs. This situation is challenging for the fusion module to give reasonable fused feature maps. 2) Moreover, the refinement module couldn't faithfully inpaint and modify the fused feature maps caused by reason 1), since the training data is limited. The limited training data makes the model easy to overfit and cannot generalize very well for several test scenes. We believe that training our model on a

Table 4. We investigate the novel view synthesis performance and speed when provided with more input views.

	3 views				6 views				9 views			
	PSNR	SSIM	LPIPS	FPS	PSNR	SSIM	LPIPS	FPS	PSNR	SSIM	LPIPS	FPS
PixelNeRF	19.24	0.687	0.399	0.025	20.29	0.725	0.372	0.013	20.91	0.75	0.348	0.009
IBRNet	18.86	0.695	0.387	0.265	20.93	0.780	0.312	0.176	21.30	0.805	0.285	0.125
FWD-U	17.42	0.598	0.341	35.4	18.11	0.623	0.323	15.0	18.93	0.648	0.304	8.5
FWD	20.15	0.721	0.259	35.4	21.40	0.758	0.235	15.0	21.98	0.779	0.218	8.5
FWD-D	21.98	0.791	0.208	43.2	22.54	0.802	0.199	29.5	22.95	0.816	0.188	22.8

Figure 2. **Failure Cases.** We show some failure cases for FWD model.

large-scale dataset would effectively resolve these artifacts.

4.5. Synthesis Results.

We show more synthesis results in the following. For comparison, we show results of the same scene and views. We first show the results of FWD-U in Figure 3, FWD in Figure 4 and FWD-D in Figure 5. We also show baseline results: PixelNeRF in Figure 6, IBRNet in Figure 7, MVS-NeRF in Figure 8, FVS in Figure 9 and Blending-R in Figure 10. Again, please see attached videos for comparison between ours and other methods.

Figure 3. **Efficient view synthesis from very sparse views for FWD-U.** We show the view synthesis results with 3 input views on DTU MVS test dataset for FWD-U trained with unsupervised depths.

Figure 4. **Efficient view synthesis from very sparse views for FWD.** We show the view synthesis results with 3 input views on DTU MVS test dataset for FWD.

Figure 5. **Efficient view synthesis from very sparse views for FWD-D.** We show the view synthesis results with 3 input views on DTU MVS test dataset for FWD-D.

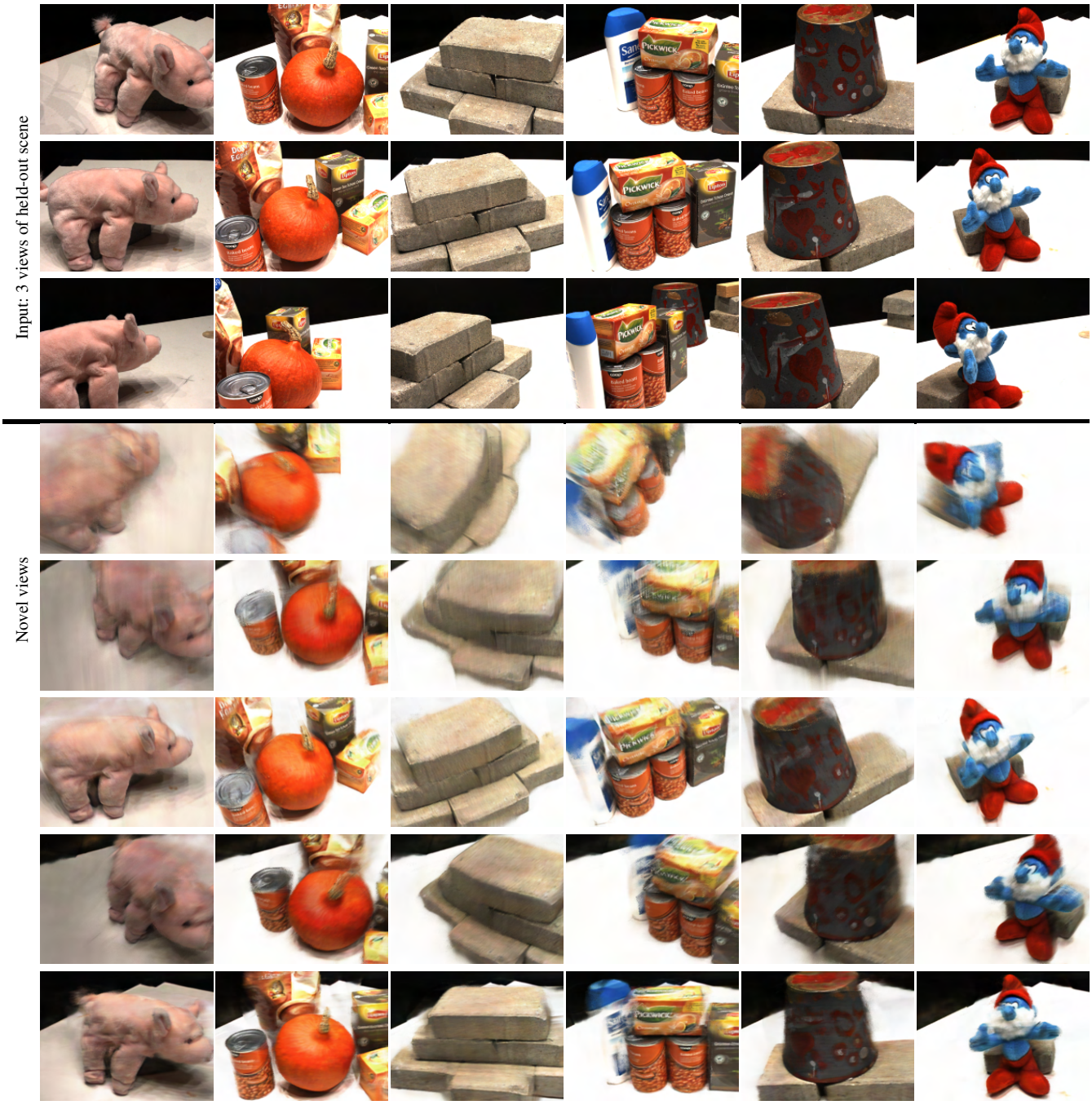

Figure 6. **Efficient view synthesis from very sparse views for PixelNeRF [7].** We show the view synthesis results with 3 input views on DTU MVS test dataset for PixelNeRF.

Figure 7. **Efficient view synthesis from very sparse views for IBRNet [8].** We show the view synthesis results with 3 input views on DTU MVS test dataset for IBRNet.

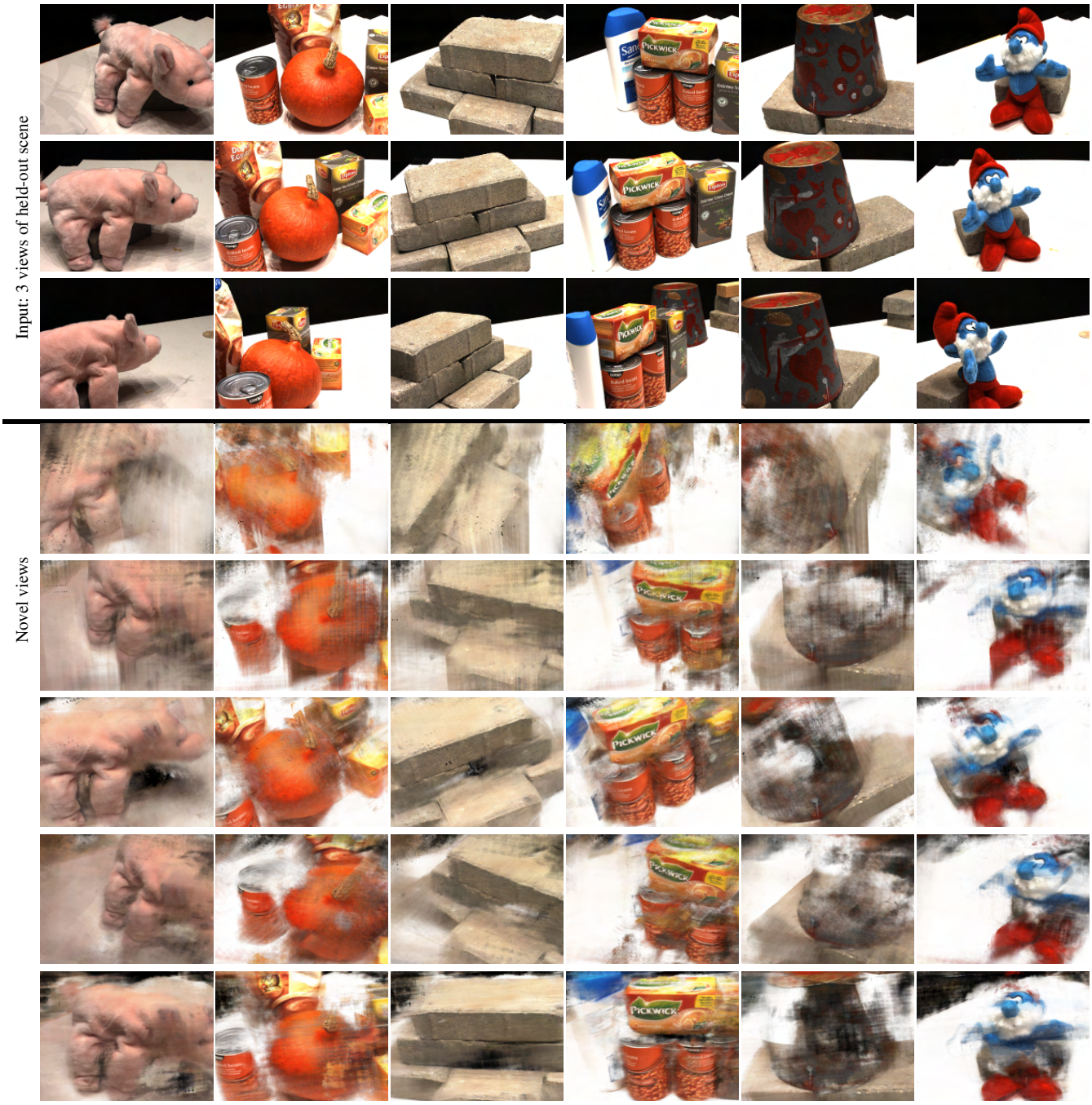

Figure 8. **Efficient view synthesis from very sparse views for MVSNeRF.** We show the view synthesis results with 3 input views on DTU MVS test dataset for MVSNeRF.

Figure 9. **Efficient view synthesis from very sparse views for FVS.** We show the view synthesis results with 3 input views on DTU MVS test dataset for FVS.

Figure 10. **Efficient view synthesis from very sparse views for Blending-R.** We show the view synthesis results with 3 input views on DTU MVS test dataset for Blending-R.

References

[1] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *ICLR*, 2019. 1

[2] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson, "Synsin: End-to-end view synthesis from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7467–7477, 2020. 1, 2

[3] F. Wang, S. Galliani, C. Vogel, P. Speciale, and M. Pollefeys, "Patchmatchnet: Learned multi-view patchmatch stereo," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14194–14203, 2021. 1

[4] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari, "Accelerating 3d deep learning with pytorch3d," *arXiv preprint arXiv:2007.08501*, 2020. 1

[5] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3504–3515, 2020. 2

[6] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl, "Large-scale data for multiple-view stereopsis," *IJCV*, vol. 120, no. 2, pp. 153–168, 2016. 2

[7] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelnerf: Neural radiance fields from one or few images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4578–4587, 2021. 2, 8

[8] Q. Wang, Z. Wang, K. Genova, P. P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser, "Ibrnet: Learning multi-view image-based rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2021. 2, 9

[9] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14124–14133, 2021. 2

[10] G. Riegler and V. Koltun, "Stable view synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 2

[11] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," vol. 37, no. 6, pp. 257:1–257:15, 2018. 2

[12] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised nerf: Fewer views and faster training for free," *arXiv preprint arXiv:2107.02791*, 2021. 2