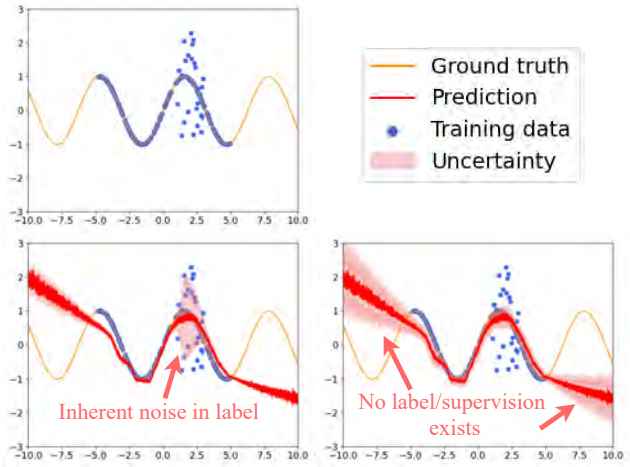
Supplementary Materials

1. Why Use Uncertainty for Self-supervised MVS?

In Bayesian deep learning, the uncertainty is categorized into two types [2]: aleatoric and epistemic uncertainty. Aleatoric uncertainty models the inherent noise in the training data, and epistemic uncertainty accounts for what is not included in the training data. As shown in Fig. 1, a toy example of aleatoric and epistemic uncertainty is provided. In Fig. 1(a), aleatoric uncertainty models the regions which have noisy labels. In Fig. 1(b), it shows that epistemic uncertainty models what current model ignores, for example, the regions without certain supervision or label.

In previous works [9, 1, 7, 11], self-supervised MVS methods are built on intuitive assumptions, aiming at involving extra priors into the self-supervision loss. It can be further viewed as an attempt to increase the certain supervision signals in self-supervision, which is proved by extensive experiments to be effective. Whereas, from an opposite viewpoint, we rethink the effect of uncertain supervision signals modeled by *epistemic uncertainty* in this work. Since epistemic uncertainty models the ignorance of supervision, it can provide us a more comprehensive and explainable understanding of self-supervision. In analogy to the epistemic uncertainty in Fig. 1(b), the uncertainty in self-supervision can guide our skepticism to the limitations of current self-supervised MVS, which is further discussed in the Introduction part of the manuscript.



(a) Aleatoric Uncertainty **(b) Epistemic Uncertainty**

Figure 1. A toy example to understand aleatoric and epistemic uncertainty.

2. Modified Backbone Network for Uncertainty Estimation

In this section, we introduce the modified backbone network for estimating the aforementioned aleatoric uncertainty and epistemic uncertainty, following a classical configuration proposed by [8]. As shown in Fig. 2, the illustration of the modified backbone architecture is presented. The aleatoric uncertainty map is directly predicted by a 6-layer CNN, whose detailed architecture is further listed in Table 1. The epistemic uncertainty map is estimated via a statistical Bayesian model by sampling T times. Though traditional Bayesian models can offer a mathematically grounded framework to estimate model uncertainty, they are usually attached with prohibitive computational cost. Hence, Monte-Carlo Dropout (MC-Dropout) [3] attempts to alleviate the huge cost in computation, casting dropout training in deep neural networks as approximate Bayesian inference in deep Gaussian process. In Monte-Carlo Dropout, the inference is done by training a model with dropout, and by also performing dropout at test time to sample from the approximate posterior. Detailed theoretical evidence is also provided in [3]. Following an open implementation[1] of [8], we append dropout layers on the bottleneck layers of the 3D U-Net in MVSNet [12] and CascadeMVSNet [5], which are applied as backbone in our proposed self-supervised MVS framework. If MVSNet is applied as backbone, the MC-Dropout is embedded in the 3D U-Net of MVSNet, whose details are provided in Table 2. If CascadeMVSNet is applied as backbone, the modified 3D U-Net with MC-Dropout shares the same architecture as Table 2 shows. Since CascadeMVSNet has multiple stages, the MC-Dropout layers are only activated on the first stage. Because too many dropout layers may result into strong regularization effect in self-supervised training, and make the model trapped in a trivial solution. To guarantee the convergence of self-supervised training in MVS, the number of dropout layers is limited.

[1] https://github.com/pmorerio/dl-uncertainty

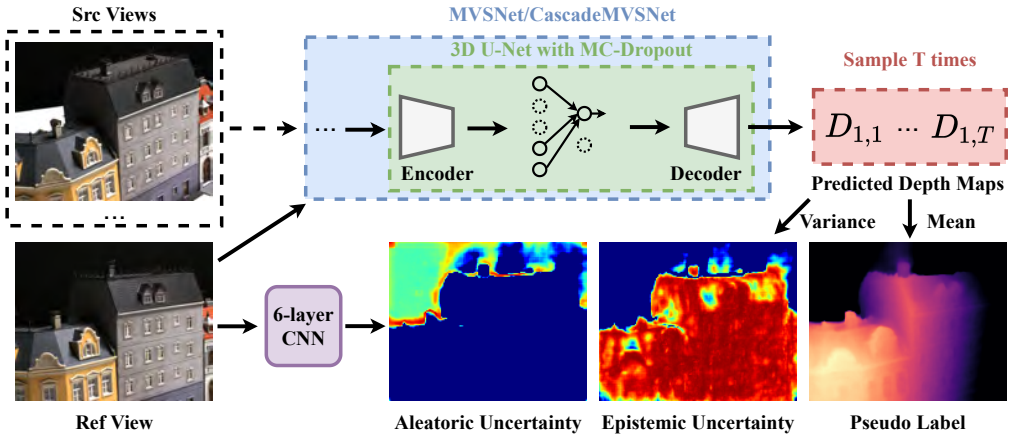


Figure 2. Illustration of the modified architecture of backbone network for estimation aleatoric uncertainty and epistemic uncertainty.

Name	Layer	Output Size
Input	-	H×W×3
Conv_0	ConvBR,K=3×3,S=1,F=16	H×W×16
Conv_1	ConvBR,K=3×3,S=1,F=32	H×W×32
Conv_2	ConvBR,K=3×3,S=1,F=128	H×W×128
Conv_3	ConvBR,K=3×3,S=1,F=256	H×W×256
Conv_4	ConvBR,K=3×3,S=1,F=32	H×W×32
Conv_5	Conv,K=3×3,S=1,F=1	H×W×1

Table 1. Network structure of the 6-layer CNN utilized to estimate *aleatoric uncertainty*. Denote that 3D convolution as "Conv", 3D deconvolution as "DeConv", batch normalization as "B", ReLU as "R" in the column of "Layer". "K" is the kernel size, "S" is the stride and "F" is the output channels. "H" and "W" represent the height and width, respectively.

Name	Layer	Output Size
Input	-	H×W×32
Conv_0	ConvBR,K=3×3,S=1,F=8	H×W×8
Conv_1	ConvBR,K=3×3,S=2,F=16	H/2×W/2×16
Conv_2	ConvBR,K=3×3,S=1,F=16	H/2×W/2×16
Conv_3	ConvBR,K=3×3,S=2,F=32	H/4×W/4×32
Conv_4	ConvBR,K=3×3,S=1,F=32	H/4×W/4×32
Conv_5	ConvBR,K=3×3,S=2,F=64	H/8×W/8×64
Conv_6	ConvBR,K=3×3,S=1,F=64	H/8×W/8×64
Drop_6	Dropout,Rate=0.5	H/8×W/8×64
Deconv_7	DeConvBR,K=3×3,S=2,F=32	H/4×W/4×32
Drop_7	Dropout,Rate=0.5	H/4×W/4×32
Shortcut_8	Deconv_7 + Conv4	H/4×W/4×32
Deconv_9	DeConvBR,K=3×3,S=2,F=16	H/2×W/2×16
Shortcut_10	Deconv_9 + Conv2	H/2×W/2×16
Deconv_11	DeConvBR,K=3×3,S=2,F=8	H×W×8
Shortcut_12	Deconv_11 + conv0	H×W×8
Conv_13	Conv,K=3×3,S=1,F=1	H×W×1

Table 2. Network structure of modified 3D U-Net in MVSNet, embeded with *Monte-Carlo Dropout* to estimate *epistemic uncertainty*. Denote that 3D convolution as "Conv", 3D deconvolution as "DeConv", batch normalization as "B", ReLU as "R" in the column of "Layer". "K" is the kernel size, "S" is the stride, "F" is the output channels and "Rate" means the dropout rate. "H" and "W" represent the height and width, respectively.

3. Implementation Details

Backbone: We directly adopt the concise open implementations of MVSNet[2] and CascadeMVSNet[3] as the bacbone in our porposed U-MVS framework. Following the suggestions proposed by [8], the backbone architecture is embedded with MC-Dropout and a 6-layer CNN for uncertainty estimation, which is introduced in the previous section. In default, the other network settings follow the original open implementation.

RGB2Flow Module: In the RGB2Flow module, we utilize a self-supervised method[4] to train an optical flow estimation network, PWC-Net [10], from the scratch on DTU dataset. The two-view pairs for estimating optical flow are selected by combining the reference view with each of the source views in the multi-view pairs provided by MVSNet [12]. After self-supervisedly pretraining the PWC-Net, it is able to predict the optical flow from RGB images in the RGB2Flow module. No extra ground truth is used in this module.

Uncertainty Estimation: As discussed in the manuscript, MC-Dropout is only activated during estimating the uncertainty maps. We proceed the forward propagation on the network with random MC-Dropout for $T = 20$ times, which can be viewed as sampling T different model weights. Following the procedure of [8], the mean and variance of T sampled depth maps are respectively treated as the pseudo label and uncertainty map, as shown in Fig. 2.

Training and testing: The whole training process is conducted on 4 RTX 2080Ti GPU. No ground truth depth

[2] https://github.com/xy-guo/MVSNet_pytorch

[3] https://github.com/alibaba/cascade-stereo/tree/master/CasMVSNet

[4] https://github.com/lliuz/ARFlow

	DTU			Intermediate of Tanks&Temples			Advanced of Tanks&Temples		
Sup	Accuracy	Completeness	Overall	Precision	Recall	F-score	Precision	Recall	F-score
✓	0.325	0.385	0.355	47.62	74.01	56.84	29.68	35.24	31.12
×	0.354	0.3535	0.3537	45.45	78.52	57.15	24.22	44.46	30.97

Table 3. Performance comparison of our self-supervised method and supervised method on DTU evaluation set, intermediate and advanced partition set of Tanks&Temples. CascadeMVSNet [5] is utilized as the backbone. Under the metrics of DTU benchmark, the smaller the value the better the performance; Under the metrics of Tanks&Temples, the larger the value the better the performance.

maps are used in the training phase[5]. In default, the hyper-parameter settings for self-supervision follow the bconfiguration of Unsup_MVS [9]. In the training phase, the image resolution is set to 640×512. Due to the limitation of memory, the batch size is set to 1 per GPU. The model is trained on the DTU training set as [12]. If MVSNet is selected as backbone, the model is firstly trained for 10 epochs in the self-supervision pretraining stage, and the model is further trained for 10 epochs in the pseudo label post-training stage. If CascadeMVSNet is selected as backbone, the self-supervision pretraining stage requires 16 epochs for training the model, and the pseudo label post-training stage requires 16 epochs for further training. We utilize Adam optimizer with a learning rate of 1e-3 which is decreased by 0.5 times every 2 epochs. In the testing phase, the depth maps on all views of the scene are predicted. After depth estimation, 3D point cloud is reconstructed from the multi-view depth maps and images [4]. The test setting is also the same as the aforementioned open implementations.

4. Discussion

4.1. Quantitative evaluation of uncertainty estimates.

Ideally, the aforementioned uncertainty should be inversely correlated with accuracy. To provide a quantitative evaluation of uncertainty estimates, we provide further experimental results in Fig. 3 following [6]. As suggested by the authors, in order to assess the capability of the uncertainty measure to predict whether a prediction is (in)correct, the depth predictions on all pixels are ranked in decreasing order of confidence. Then the per-pixel error rate of the depth predictions are computed. As shown in the figure, the abscissa represented the percentage of selected pixels ranked by the uncertainty, which is also called "density"[6]. The ordinate in the figure shows the absolute error rate of the selected pixels according its density. It is noted that as the density/uncertainty increases, the absolute error rate increases as well. We can find that the aforementioned uncertainty is inversely correlated with accuracy, which supports the idea of rejecting invalid supervisions according to uncertainty estimates.

[5]The code will be released on Github in the future

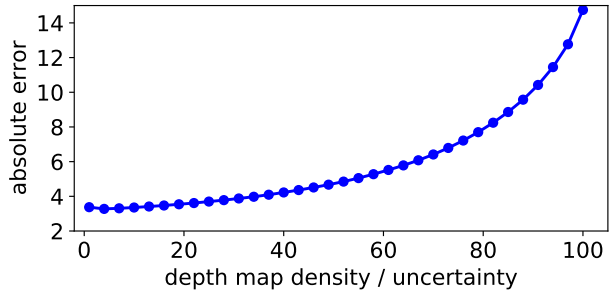


Figure 3. Quantitative evaluation results of uncertainty estimates.

4.2. Can Self-supervised Methods "Outperform" Supervised Methods?

In Table. 3, we provide a direct comparison of our proposed self-supervised MVS framework and supervised method on several benchmarks with the same backbone of CascadeMVSNet [5], such as DTU, intermediate and advanced partition in Tanks&Temples. On the third row of Table 3, the performance of supervised method is picked from the original paper and the official leaderboard of Tanks&Temples. On the fourth row of Table 3, we provide the performance of our proposed self-supervised method under the same metrics. The performance of our proposed unsupervised method on intermediate[6] and advanced[7] partition of Tanks&Temples benchmark can be found on the official website.

From the quantitative comparison in Table 3, we can find: the Accuracy/Precison of supervised method is better than our unsupervised method; Whereas, the Completeness/Recall of self-supervised method is better than the supervised one; Moreover, the Overall/F-score of self-supervised method is competitive with the supervised method. It demonstrates that each of the supervised and self-supervised methods has its own advantages. For supervised training, the results are more precise and accurate on each point of the reconstructed 3D point cloud, compared with self-supervised method. Because there is an inevitable loss of detailed information in self-supervision signal built upon cross-view correspondence between discrete pixels.

[6]The submission is named as *6956-ss-mvs-test* on `https://www.tanksandtemples.org/leaderboard/IntermediateF/`

[7]The submission is named as *6956-self-sup-mvs* on `https://www.tanksandtemples.org/leaderboard/AdvancedF/`

For our self-supervised MVS framework, the advantage is that the reconstructed 3D model can retain more integral parts, compared with supervised training. It shows that our self-supervised framework can excavate depth information from abundant correspondence priors from the multi-view images which can cover more integral parts of the 3D object.

4.3. Visualization of the Reconstructed 3D Models

We visualize the reconstructed 3D point clouds from DTU evaluation set and Tanks&Temples test set respectively in Fig. 4, Fig. 5 and Fig. 6.

5. Limitations

1) The computation and time consumption for uncertainty estimation is enormous. Though the MC-Dropout can alleviate the prohibitive computational cost of Bayesian model during estimating the uncertainty maps, it still requires to sample $T = 20$ times, which means the model is forward propagated for 20 more times. It shows that the uncertainty estimation may be 20 times slower than a normal forward propagation. As a result, the uncertainty estimation phase of our proposed U-MVS framework may be time-consuming. In future work, a fast and light-weight method for uncertainty estimation is necessary.

2) Extension in a semi-supervised framework is ignored. As discussed in Section 4.2 and Table 3, each of the supervised method and our proposed self-supervised framework has its own advantage: the supervised training can reconstruct more accurate details of 3D model, whereas self-supervised method can retain integral parts in 3D reconstruction, producing more complete results. The combination of these two methods may provide further improvements in the performance of 3D reconstruction, such as semi-supervised learning, in future works.

References

[1] Yuchao Dai, Zhidong Zhu, Zhibo Rao, and Bo Li. Mvs2: Deep unsupervised multi-view stereo with multi-view symmetry. In *2019 International Conference on 3D Vision (3DV)*, pages 1–8. IEEE, 2019.

[2] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

[3] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[4] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 873–881, 2015.

[5] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2495–2504, 2020.

[6] Xiaoyan Hu and Philippos Mordohai. A quantitative evaluation of confidence measures for stereo vision. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2121–2133, 2012.

[7] Baichuan Huang, Can Huang, Yijia He, Jingbin Liu, and Xiao Liu. Mˆ 3vsnet: Unsupervised multi-metric multi-view stereo network. *arXiv preprint arXiv:2005.00363*, 2020.

[8] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[9] Tejas Khot, Shubham Agrawal, Shubham Tulsiani, Christoph Mertz, Simon Lucey, and Martial Hebert. Learning unsupervised multi-view stereopsis via robust photometric consistency. *arXiv preprint arXiv:1905.02706*, 2019.

[10] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[11] Hongbin Xu, Zhipeng Zhou, Yu Qiao, Wenxiong Kang, and Qiuxia Wu. Self-supervised multi-view stereo via effective co-segmentation and data-augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[12] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018.

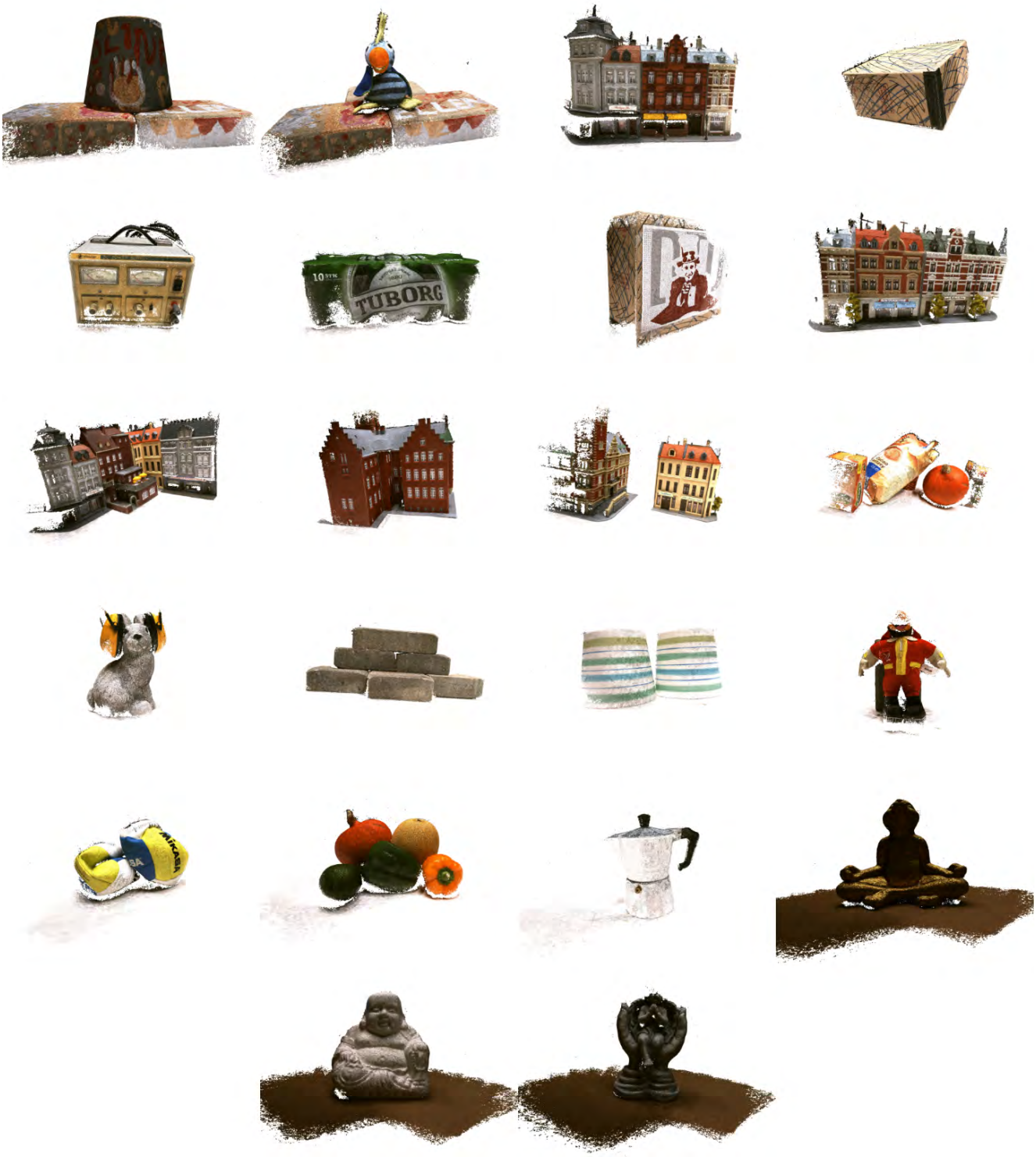


Figure 4. Visualization of all scenes in DTU evaluation set.

Figure 5. Visualization of all scenes on the *intermediate* partition of Tanks&Temples dataset.

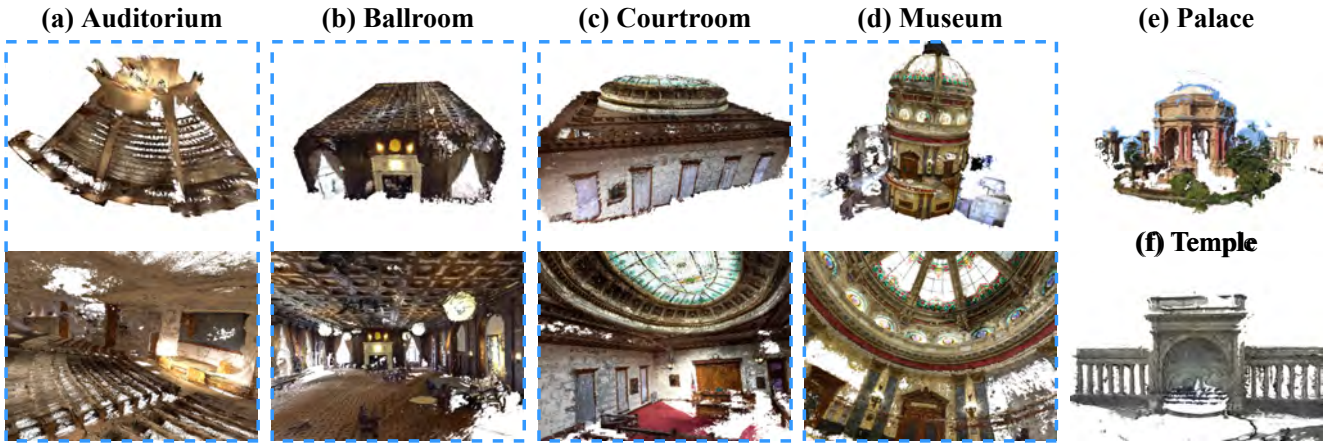


Figure 6. Visualization of all scenes on the *advanced* partition of Tanks&Temples dataset.