Supplementary Material: PatchMatch-RL: Deep MVS with Pixelwise Depth, Normal, and Visibility

1. Network Architecture

We present the network architecture of our PatchMatch-RL (Table 1). We refer the readers to the original text for detailed information on how each component is used. Our method contains a total of 149K parameters.

Name	Parameters	Input	Output Size
$\mathcal{I}$	-	-	$3\times H\times W$
Feature Extraction			
Conv0_0	CBR, k=7, s=1, d=2	$\mathcal{I}$	$8\times H\times W$
Conv0_1	CBR, k=3, s=1, d=1	Conv0_0	$8\times H\times W$
Conv1_0	CBR, k=5, s=2, d=1	Conv0_1	$16\times H/2\times W/2$
Conv1_1	CBR, k=3, s=1, d=1	Conv1_0	$16\times H/2\times W/2$
Conv1_2	CBR, k=3, s=1, d=1	Conv1_1	$16\times H/2\times W/2$
Conv2_0	CBR, k=5, s=2, d=1	Conv1_2	$32\times H/4\times W/4$
Conv2_1	CBR, k=3, s=1, d=1	Conv2_0	$32\times H/4\times W/4$
Conv3_0	CBR, k=5, s=2, d=1	Conv2_1	$64\times H/8\times W/8$
Conv3_1	CBR, k=3, s=1, d=1	Conv3_0	$64\times H/8\times W/8$
ConvL3	Conv, k=1, s=1, d=1	Conv3_1	$64\times H/8\times W/8$
ConvL3_2	UpConv, k=1, s=1, d=1	ConvL3	$32\times H/4\times W/4$
ConvL2	Conv, k=1, s=1, d=1	ConvL2_1	$32\times H/4\times W/4$
ConvL2_1	UpConv, k=1, s=1, d=1	ConvL2	$16\times H/2\times W/2$
ConvL1	Conv, k=1, s=1, d=1	ConvL1_2	$16\times H/2\times W/2$
$\mathcal{F}^3$	ConvL3		$64\times H/8\times W/8$
$\mathcal{F}^2$	ConvL3_2 + ConvL2		$32\times H/4\times W/4$
$\mathcal{F}^1$	ConvL2_1 + ConvL1		$16\times H/2\times W/2$
Attentional Projection			
$\mathcal{A}_i^q$	Conv1d, k=1, s=1, d=1	$\mathcal{F}_i$	$\|\|\mathcal{F}_i\|\|\times H/2^i\times W/2^i$
$\mathcal{G}_i$	$\sum_q \mathcal{A}_q \cdot (\mathcal{F}_q^r \mathcal{F}_{H^{r\to s}\cdot q}^r)$		$4\times H/2^i\times W/2^i$
View Scorer			
View_0$_i$	CR, k=5, s=1, d=1	$\mathcal{G}_i$ & GeomP	$8\times H/2^i\times W/2^i$
View_1$_i$	CR, k=3, s=1, d=1	View_1$_i$	$16\times H/2^i\times W/2^i$
View_2$_i$	Conv, k=1, s=1, d=1	View_2$_i$	$1\times H/2^i\times W/2^i$
$\hat{\mathcal{V}}_i$	Sigmoid	View_2$_i$	$1\times H/2^i\times W/2^i$
Recurrent Cost Regularization			
RCN_0$_i$	Conv, k=3, s=1, d=1	$\mathcal{G}_i^{\mathcal{V}}$ & ζ	$8\times H/2^i\times W/2^i$
RCN_1$_i$	Conv, k=1, s=1, d=1	RCN_0$_i$	$8\times H/2^i\times W/2^i$
RCN_2$_i$	Conv, k=1, s=1, d=1	RCN_1$_i$	$8\times H/2^i\times W/2^i$
$\mathcal{Z}_i$	Conv1d, k=1, s=1, d=1	RCE_2$_i$	$1\times H/2^i\times W/2^i$

Table 1. **Network Architecture Details.** CBR stands for 2D convolution with batch normalization and ReLU activation. UpConv represents bilinear upsampling followed by 2D convolution. k is kernel size, s is stride, and d is dilation. GeomP represents the encoded geometric prior, which is explained in detail in Section 2

2. Geometric Priors

We describe our geometric priors: scale, incident-angle, and triangulation angle differences. Figure 1 illustrates how these are computed. These geometric priors are used in the visibility estimation process. The scale difference is computed as a ratio given by the source projection distance to the oriented point divided by the reference projection distance to the oriented point. The incident-angle is obtained from the angle between the normal value of the oriented point and the ray from the center of the source camera to the oriented point. The triangulation angle is calculated as the angle between the projection ray of the reference and source camera centers to the oriented point. For the images and camera pose, we estimate the scale difference as the source projection distance divided by the reference projection distance. In order to further augment the prior information, we use positional encoding to augment the raw prior values. We use 5 encodings of sines and cosines of increasing scale.

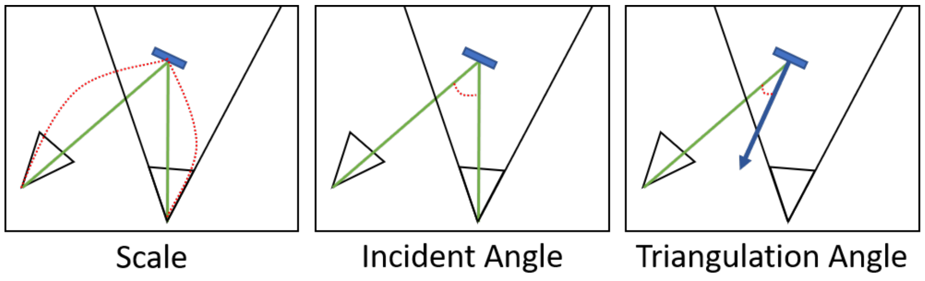

Figure 1. **Visualization of geometric priors.** From the left, the scale difference is given by the ratio between the camera to point distances. The incident angle is given by the angle between the rays from the camera to the point. Triangulation angle is given by the angle between the oriented point tangent normal and the ray from the source camera to the point.

3. Support Window Dilation

Since features extracted from Convolutional Neural Networks (CNN) are locally smooth, using direct neighboring pixels as the supporting window prevents the neighbors from providing distinguishing information. Table 2 describes the impact of dilation in the support window. We show that in both 2cm and 5cm benchmarks, having no dilation will reduce the reconstruction accuracy (-2.7%, -1.8%) and the reconstruction is highly incomplete (-10.1%, -7.2%).

Model		Accuracy / Completeness / F1 Train 2CM	Accuracy / Completeness / F1 Train 5cm
No dilation	$(\beta = 1)$	73.4 / 52.1 / 60.1	88.7 / 71.6 / 78.6
Ours	$(\beta = 3)$	**76.1 / 62.2 / 67.8**	**90.5 / 78.8 / 83.3**

Table 2. **Comparison of performance on the support window dilation.** We compare our system (using the supporting region with dilation of $\beta = 3$) to the system that uses direct neighboring pixels as the supporting region. For each threshold, the model with higher F_1 score is marked in bold.

4. Fusion Parameters

We use different fusion parameter settings for the different benchmarks because the number of images in each set of the ETH3D high-res benchmark is much smaller than the number of images in each set of the Tanks and Temples benchmark. Given reference pixel $p \in I_{ref}$ and corresponding source pixel $q = H_{\omega_p}^{ref\rightarrow src} \cdot p$, we first define the projection distance threshold τ_{proj} as the distance between the projected pixel of the corresponding source depth and the reference pixel (*i.e.* $||p - H_{\omega_q}^{src\rightarrow ref} \cdot q||_2$), relative depth threshold τ_{rel} as the relative error of the source depth and the reference depth (*i.e.*, $(d_p^{ref} - d_q^{ref})/d_p^{ref}$), and angle threshold τ_{angle} as the angular difference between the oriented point normal values of the corresponding source and reference oriented points (*i.e.* $\arccos \mathbf{n}_p^T \cdot \mathbf{n}_q$). The source pixel is marked as consistent if the projection distance, relative error, and angular difference are less than τ_{proj}, τ_{rel}, and τ_{angle} respectively.

For all scenes of the ETH3D high-res benchmark, we use $\tau_{proj} = 1px$, $\tau_{rel} = 1\%$, and $\tau_{angle} = 30°$ and keep the depth values that are consistent across at least 1 source view. For all scenes of the Tanks and Temples benchmark, we use $\tau_{proj} = 1px$, $\tau_{rel} = 1\%$, and $\tau_{angle} = 30°$. For depth consistency checks, we use 2 source views for indoor scenes (*Auditorium, Ballroom, Courtroom, Museum*), 3 source views for outdoor scenes (*Palace, Temple, Train, Lighthouse, Playground*), and 5 source views for object scenes (*Family, Francis, Horse, M60, Panther*). The depth maps for each image are then averaged over the consistent views into a point cloud.

Additionally, we fine-tune our fusion parameters to estimate the accuracy-completeness trade-off point, where the accuracy remains at least 90% for the 2cm benchmark. Table 3 shows how this trade-off point compares to our setup. By enforcing a stricter fusion parameter, our completeness drops heavily in both 2cm and 5cm benchmarks (-29.8%, -30.1%), making the overall reconstruction F_1 score drop accordingly (-31.5%, -29.7%). This implies that our system does not estimate points as accurately as the methods that use high-resolution, hand-crafted features (e.g. ACMH and ACMM [4]).

		Accuracy / Completeness / F1	
Model		**Train 2CM**	**Train 5cm**
Acc-90	$(0.5px/0.5\%/15°/2$ Views$)$	90.5 / 32.4 / 46.3	97.1 / 48.7 / 63.6
Ours	$(1px/1.0\%/30°/1$ View$)$	**76.1 / 62.2 / 67.8**	**90.5 / 78.8 / 83.3**

Table 3. **Accuracy-Completness trade-off.** We compare the accuracy completeness trade off by enforcing stricter thresholds for higher accuracy. Acc-90 represents the setting where the fusion parameter was fine tuned to achieve near 90% accuracy in the 2cm benchmark. *Ours* denotes the settings we use in the experiments. The values next to the model name are $\tau_{proj}, \tau_{rel}, \tau_{angle}$, and number of consistent source views for each pixel. For each setting, the model with higher F_1 score is marked in bold.

5. Training Rewards

We provide the reward plots for the training to show how the rewards increase over time. Figure 2 shows the mean and the standard deviation of the training rewards per half-epochs and per step. We show that the reward increases over time as we train for more epochs. Due to the nature of the policy gradient algorithm, we also observe the variance of rewards increasing.

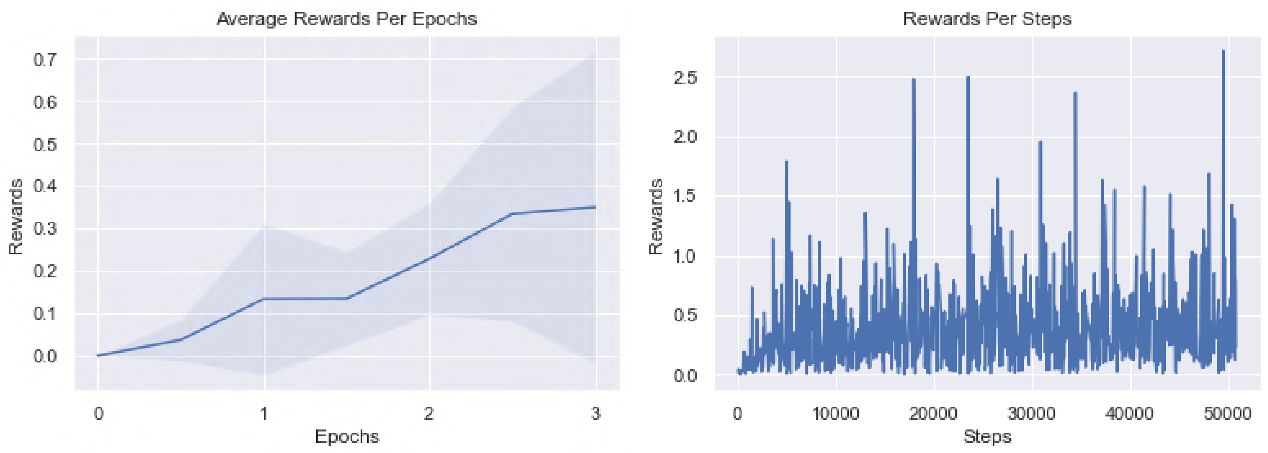

Figure 2. **Rewards visualization.** We visualize average rewards per pixel for each half-epoch and step. The shaded area in the left plot represents the standard deviation for each half-epoch.

6. Formal Definition of Markov Decition Process

We first define the formal definitions of MDP by letting $(s^i \in \mathbb{S}, a^i \in \mathbb{A}, r^i \in \mathbb{R})$ to denote the state space, the action space, and the reward space at time i for each pixel. For sake of simplicity, we let all states contain static sub-states of camera poses and images. Our s^i is a set of candidates propagated using the neighboring pixels. Our a^i is a sampling of the candidate. The state transition function $P_{a^i}(s^i, s^{i+1})$ is given by PatchMatch propagation, which generates a new set of candidates based on a nearest neighbor search of selected candidates. The reward r^i is given by comparing the selected candidates and the ground truth.

Unlike the training of the view selection function, which uses the original REINFORCE algorithm without modification, we use the cross entropy between the reward and the cost distributions formed by each candidates for training the candidate scoring function. This is possible because ground truth is available.

7. Analysis of Runtime

We compare our method with PatchMatchNet [3], which achieves state-of-the-art runtime and memory usage for learned MVS. PatchMatchNet and our work have major differences at runtime due to the use of support windows and the number of depth candidate evaluations at each coarse-to-fine stage. Both PatchMatchNet and our method contain 3 stages (excluding the refinement stage), where stages 1, 2, and 3 represent solving depth maps at scale $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$ respectively. Stage 1 takes the longest time since the target resolution is the highest. In stage 1, PatchMatchNet evaluates 8 candidates, while we evaluate 26 candidates for each pixel. This is because PatchMatchNet samples 8 candidates from perturbing upsampled depth values from the previous stage without propagation, while we test 5 perturbed candidates and 8 propagated candidates for two iterations. On our hardware, PatchMatchNet takes 0.177s to evaluate 8 candidates at stage 1, and the average depth map evaluation is about 0.022s. We take 4.54s per evaluating 13 candidates at stage 1, and each depth map evaluation takes about 0.349s on average. Since we evaluate 9 supporting pixels for each center pixel, it takes around 0.038s for each supporting pixel of each depth map. Another factor is that we use more images: 10 source images instead of 7. Table 4 shows an in-depth overview of runtime per image on the ETH3D benchmark.

Stage	Resolution	Num Iterations	Num Candidates	Runtime
3	$^H/_8 \times {}^W/_8$	8	13	4.89s
2	$^H/_4 \times {}^W/_4$	2	13	3.57s
1	$^H/_2 \times {}^W/_2$	2	13	4.54s

Table 4. **Runtime Analysis per stage.** We measure the average runtime per stage for each images in ETH3D benchmark dataset. We use image resolutions of 1920×1280 with 10 source images for each view.

8. Additional qualitative results

Figure 3 and Figure 4 show our point cloud reconstructions of the ETH3D [2] High-res train and test sets respectively. Figure 5 and Figure 6 show our results for the Tanks and Temples [1] dataset.

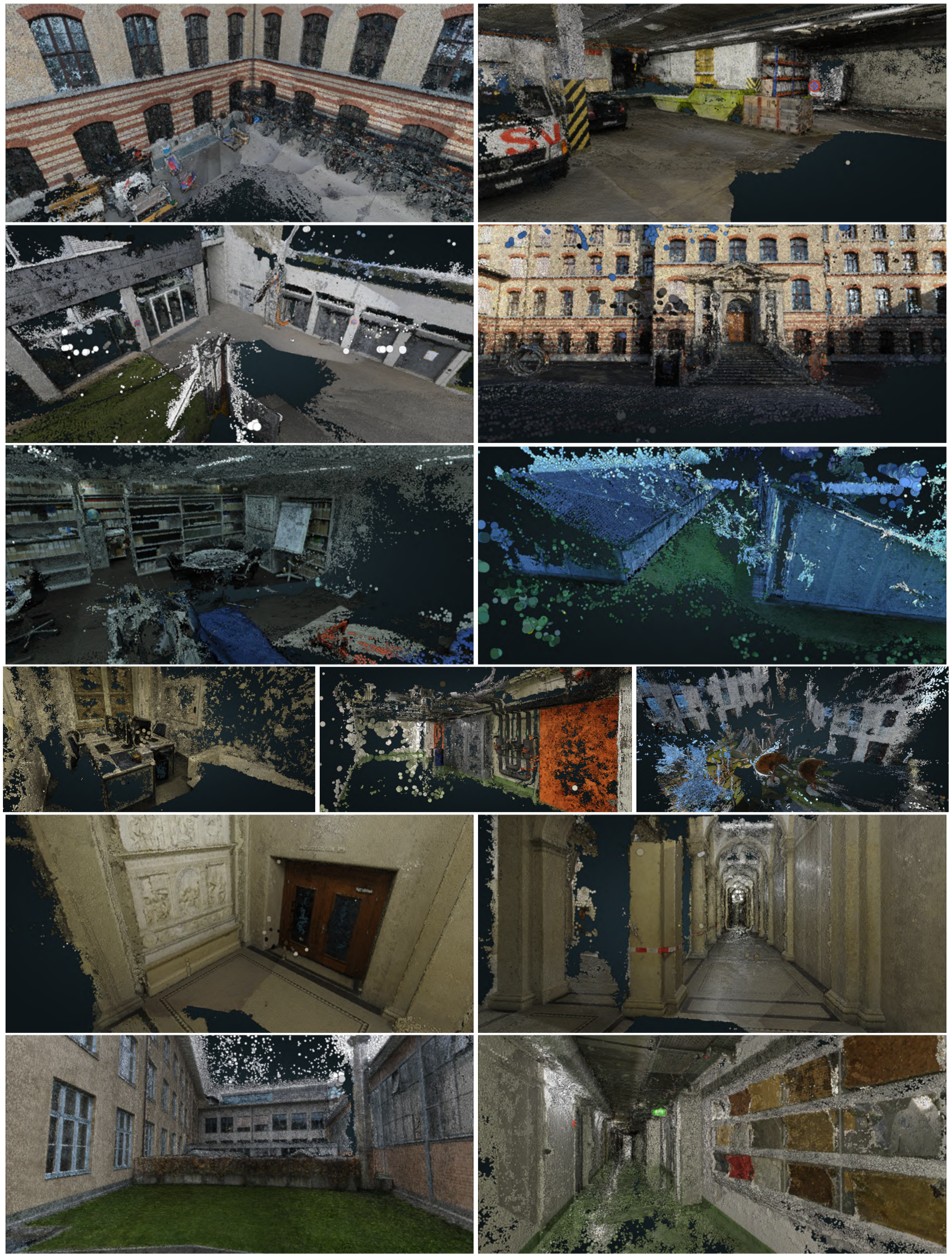
Figure 3. **Additional Qualitative Results for ETH3D High-Res [2] Train set.**

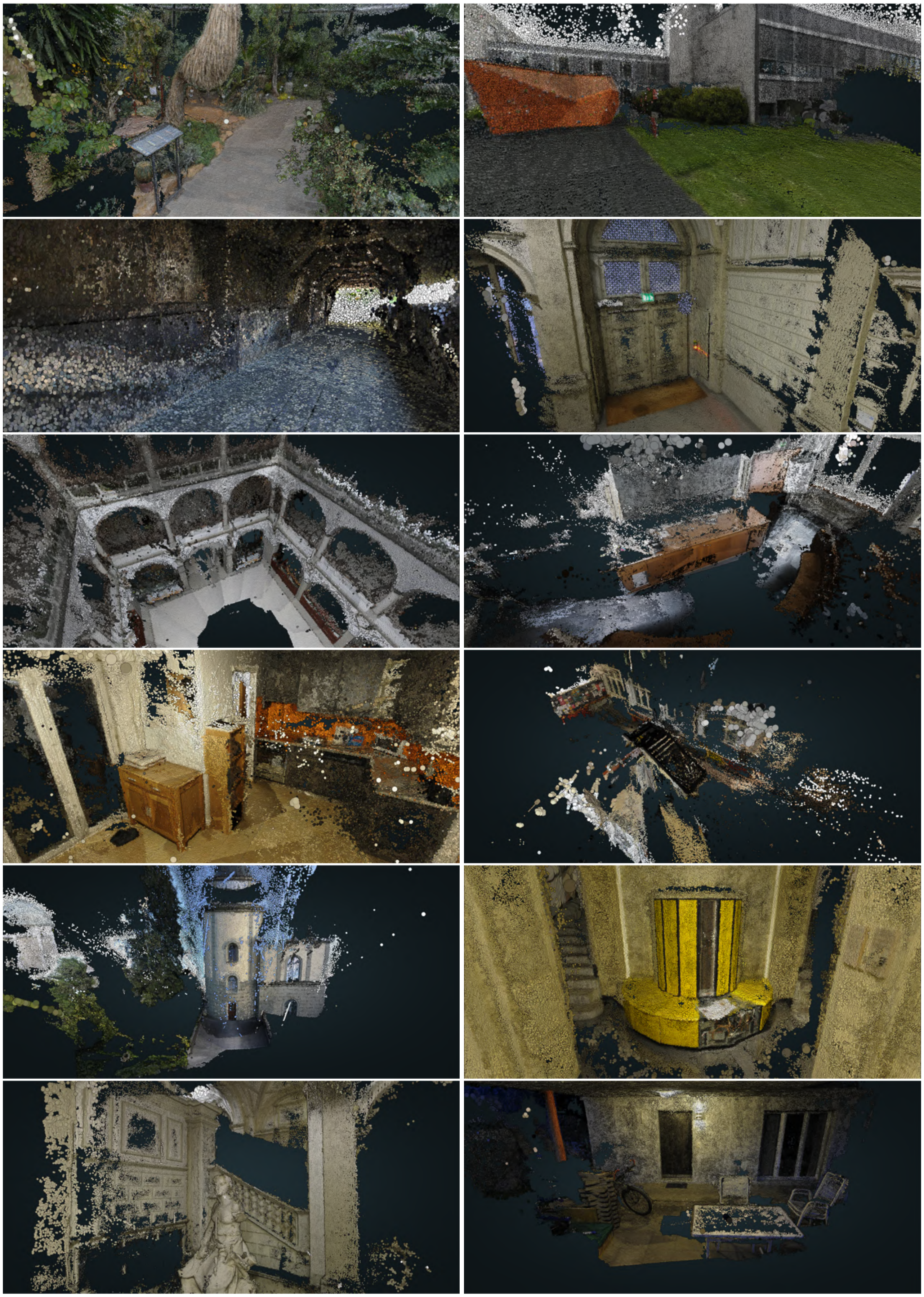

Figure 4. **Additional Qualitative Results for ETH3D High-Res [2] Test set.**

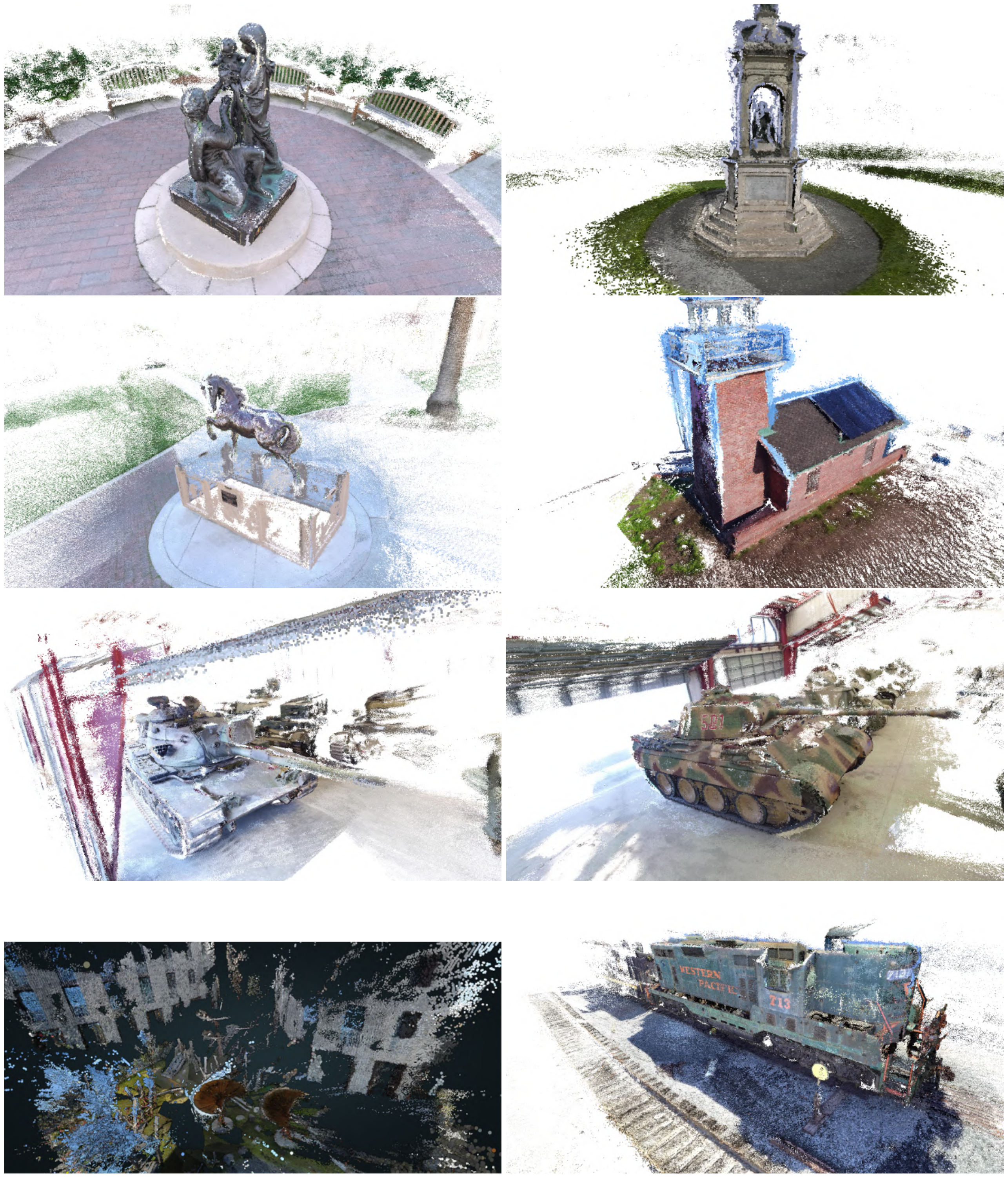

Figure 5. **Additional Qualitative Results for Tanks and Temples [1] Intermediate set.**

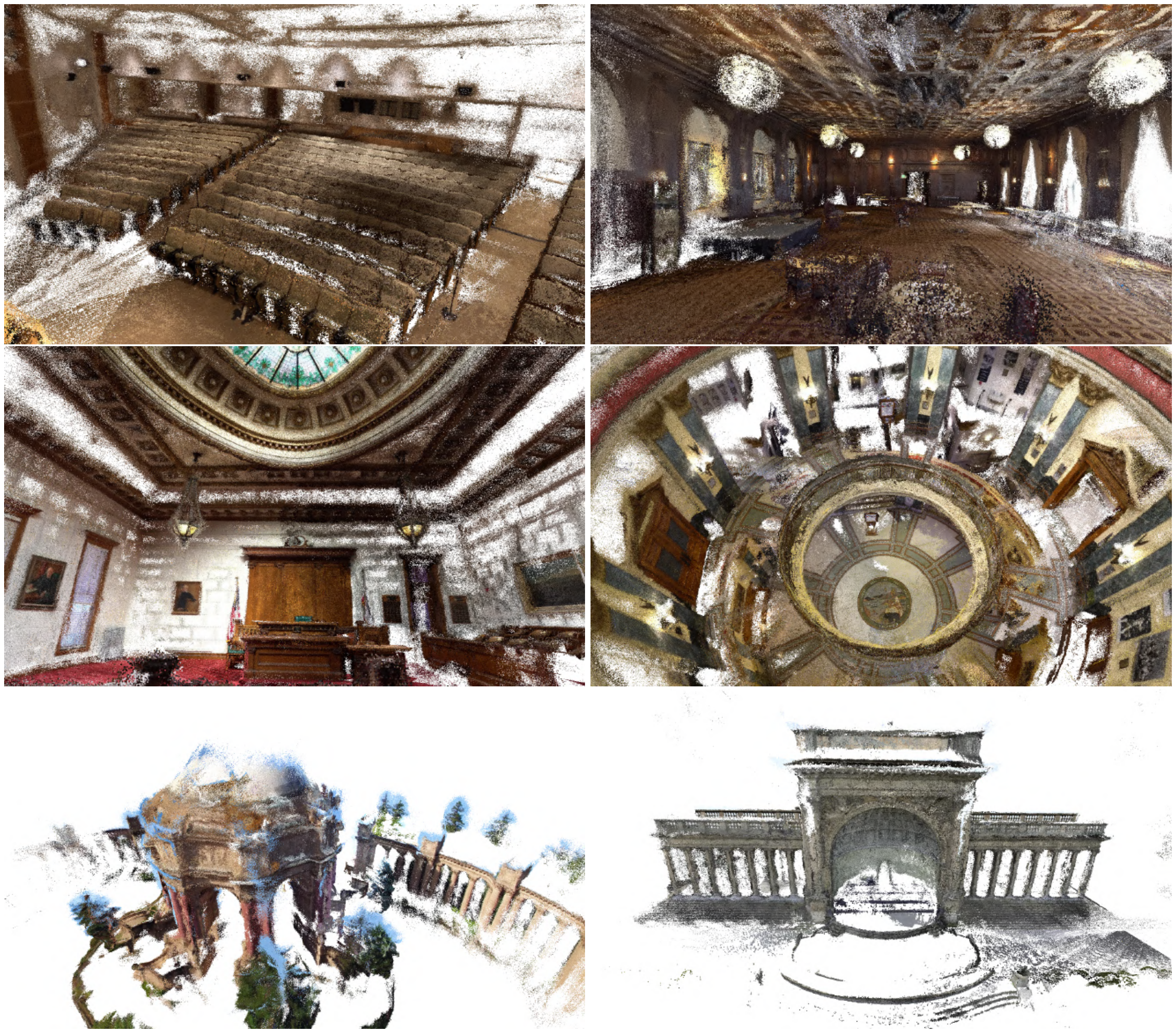
Figure 6. **Additional Qualitative Results for Tanks and Temples [1] Advanced set.**

References

[1] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 4, 7, 8

[2] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4, 5, 6

[3] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo, 2020. 4

[4] Qingshan Xu and Wenbing Tao. Multi-scale geometric consistency guided multi-view stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5483–5492, 2019. 3