Hierarchical Prior Mining for Non-local Multi-View Stereo Supplementary Material

Chunlin Ren[1] Qingshan Xu[2] Shikun Zhang[1] Jiaqi Yang[1*]
[1] Northwestern Polytechnical University [2] Nanyang Technological University
{renchunlin, zhangshikun}@mail.nwpu.edu.cn; qingshan.xu@ntu.edu.sg;
jqyang@nwpu.edu.cn

A. Review of ACMH

In this section, we briefly review the state-of-the-art diffusion-like propagation MVS framework ACMH [10], which is the baseline method of HPM-MVS. ACMH follows the classical four-step pipeline of the PatchMatch MVS method and improves hypothesis propagation and multi-view matching cost evaluation. Therefore, we focus on these two parts of ACMH.

Hypothesis Propagation. Following the idea proposed in Gipuma [1], ACMH first divides all the pixels in the reference image into a red-black checkerboard pattern, which uses black pixels as candidates to update all the red pixels in parallel and vice versa. Then, ACMH expands eight fixed positions in Gipuma into four V-shaped areas and four long strip areas and samples eight reasonable hypotheses for the center pixel to be updated. Finally, eight sampled hypotheses are set to a current candidate hypothesis set, $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i | i = 0 \cdots 8\}$.

Multi-View Matching Cost Evaluation. Given a view selection set $\boldsymbol{I}$ and a current candidate hypothesis set $\boldsymbol{\Theta}$ for the center pixel, ACMH first calculates their corresponding matching costs and embeds them into a cost matrix $\boldsymbol{M}$ which is used for better view selection,

$$\boldsymbol{M} = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,N-1} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{8,1} & m_{8,2} & \cdots & m_{8,N-1} \end{bmatrix}, \tag{1}$$

where N is the number of views in the set $\boldsymbol{I}$ and $m_{i,j}$ is the matching cost for hypothesis θ_i scored by the view I_j. Then, ACMH employs a voting decision in each column to obtain an appropriate view set $\boldsymbol{S}_t$. However, different views in $\boldsymbol{S}_t$ serve different purposes and they should contribute different weights. Therefore, the final multi-view aggregated matching cost is defined as,

$$c_{photo}(\theta_i) = \frac{\sum_j w_j \cdot m_{i,j}}{\sum_j w_j}, \tag{2}$$

where w_j is the weight of view I_j. Finally, the hypothesis with the minimum multi-view matching cost is utilized to modify the center pixel.

B. Fusion

After generating all the depth maps, a fusion step is used to merge them into a point cloud. We use different fusion settings for different datasets concerning the difference of the numbers of images in Tanks & Temples dataset [2] and ETH3D benchmark [5].

Fusion on Tanks & Temples Datasets. We follow a fusion method similar to [4, 10] to merge hypotheses into a point cloud. First, the hypotheses of each image are projected into neighboring images to generate corresponding matches. Next, a consistent match is defined as satisfying the following fixed consistency constraints: the relative depth error ε less than 0.01, the angle between normals θ less than 10° and the reprojection error ψ less the 2 pixels. If there exist $\tau_{src} \geq 2$ source images whose corresponding matches satisfy the above conditions, these hypotheses are projected into the world coordinate and averaged to a 3D point.

Fusion on ETH3D Benchmark. We employ a flexible fusion method proposed by [9], which defines a dynamic consistency constraint. Specifically, in at least $\tau_{src} = 1$ source image, if the relative depth error ε is less than 0.01, the normal deviation angle θ is less than 30° and the reprojection error ψ is less than 2 pixels, the dynamic consistency constrain for pixel x in image I_i is defined as,

$$c(x) = \sum_{j=0 \wedge j \neq i}^{N-1} c_{i,j}(p), \tag{3}$$

$$c_{i,j}(p) = e^{-(\lambda_d \cdot \varepsilon + \lambda_n \cdot \theta + \lambda_r \cdot \psi)}, \tag{4}$$

[1]Corresponding author

where $\lambda_d = 200$, $\lambda_n = 10$ and $\lambda_{re} = 1$ are weights. In the meantime, a counter n will be used to record the number of dynamical consistency calculations. At last, if the consistency metric satisfies the condition $c(x) > 0.3n$, the reliable hypothesis will be projected as a 3D point.

C. Algorithm

Algorithm 1: HPM-MVS pipline

Input : multi-view images with camera parameters
Output: a 3D point cloud

for *each image* **do**
  set reference image and source images;
  random initialization hypotheses;
  for *iteration i = 1 to* T_{photo} **do**
    hypothesis propagation via NESP;
    update hypotheses via multi-view matching cost with photometric consistency;
    refinement via Eq. 4;
  end
end
for *each image* **do**
  set reference image and source images;
  random initialization hypotheses;
  for *scale j = 0 to* T_{scale} **do**
    downsample the hypothesis map to scale *j*;
    construct prior via Eq. 5 and 6;
    upsample the prior to the original scale;
    for *iteration i = 1 to* T_{prior} **do**
      hypothesis propagation via NESP;
      update hypotheses via multi-view matching cost with prior assistance;
      refinement via Eq. 4;
    end
  end
end
for *each image* **do**
  set reference image and source images;
  initialization via the obtained hypotheses;
  for *iteration i = 1 to* T_{geom} **do**
    hypothesis propagation via NESP;
    update hypotheses via multi-view matching cost with geometric consistency;
    refinement via Eq. 4;
  end
end
for *each image* **do**
  back project hypotheses and generate points;
end

Algorithm 1 is the overall pipeline of HPM-MVS. To ensure the overall efficiency of the algorithm, $\{T_{photo}, T_{prior}, T_{geom}\}$ is set to $\{3, 3, 2\}$, respectively. T_{scale} is a dynamic variable that depends on the current image's resolution,

$$T_{scale} = \max(0, \left\lceil \log_2 \left[\frac{\max(len, wid)}{1000} \right] \right\rceil), \quad (5)$$

where len and wid represent the length and width of the image.

Table 1: Runtime of depth map generation for a 3200 × 2130 image.

Method	Time (s)
COLMAP [4]	60.87
ACMM [10]	19.68
ACMP [11]	**13.14**
ACMMP [9]	22.33
HPM-MVS$_{\text{fast}}$	<u>16.72</u>
HPM-MVS	21.86

Table 2: Runtime of different stages of HPM-MVS$_{\text{fast}}$ for a 3200 × 2130 image.

Stage	Time (s)	Ratio (%)
Basic MVS with NESP	3.78	22.61
Planar Prior Model Construction (low scale)	2.46	14.71
Basic MVS with NESP & Prior Assistance	2.45	14.65
Basic MVS with NESP & Geometric Consistency	8.03	48.03
Total Time	16.72	-

Table 3: Runtime of different stages of HPM-MVS for a 3200 × 2130 image.

Stage	Time (s)	Ratio (%)
Basic MVS with NESP	3.92	17.93
Planar Prior Model Construction (low scale)	2.53	11.57
Basic MVS with NESP & Prior Assistance	1.99	9.10
Planar Prior Model Construction (medium scale)	0.88	4.03
Basic MVS with NESP & Prior Assistance	1.96	8.97
Planar Prior Model Construction (high scale)	0.72	3.29
Basic MVS with NESP & Prior Assistance	1.88	8.60
Basic MVS with NESP & Geometric Consistency	7.98	36.51
Total Time	21.86	-

D. Runtime Performance

In this section, we conduct a runtime analysis experiment. For evaluation, we set the input image size to 3200 × 2130 and test all methods on the same computer.

Table 1 summarizes the runtime of depth map generation for different PatchMatch MVS methods. One can see that COLMAP [4] is significantly more time-consuming than others, because the sequential propagation of COLMAP only updates one pixel horizontally and vertically at a time. Although our HPM-MVS and HPM-MVS$_{\text{fast}}$ are less efficient than ACMP [11] due to the extensible strategy in the hypothesis propagation and the additional upsampling operation, they are still faster than ACMMP [9] and are in an acceptable range. Overall, our methods can strike a good trade-off in terms of accuracy, completeness and efficiency.

Table 4: Point cloud evaluation on ETH3D [5] benchmark at different thresholds ($1cm$, $2cm$, $5cm$ and $10cm$).

	Method	$1cm$	$2cm$	$5cm$	$10cm$
Train.	*i) Traditional*				
	COLMAP [4]	**84.34** / 38.65 / 51.99	**91.85** / 55.13 / 67.66	**97.09** / 69.91 / 80.50	**98.75** / 79.47 / 87.61
	PCF-MVS [3]	73.99 / 62.31 / 67.32	84.11 / 75.73 / 79.42	92.44 / 85.52 / 88.66	95.98 / 90.42 / 92.98
	ACMM [10]	82.85 / 57.91 / 67.58	90.67 / 70.42 / 78.86	96.31 / 80.91 / 87.68	98.12 / 86.40 / 91.70
	ACMP [11]	82.17 / 59.78 / 68.72	90.12 / 72.15 / 79.79	95.96 / 82.23 / 88.32	97.97 / 87.15 / 92.03
	ACMMP [9]	83.17 / 63.27 / 71.57	91.03 / 77.27 / 83.42	96.12 / 88.48 / 92.03	97.96 / 93.19 / 95.46
	ii) Learning				
	PatchMatchNet [7]	48.67 / 53.37 / 49.92	64.81 / 65.43 / 64.21	82.44 / 76.85 / 78.67	89.98 / 83.28 / 85.70
	IterMVS [6]	59.24 / 50.23 / 53.45	79.79 / 66.08 / 71.69	88.32 / 72.43 / 78.79	96.35 / 82.62 / 88.60
	MVSTER [8]	53.08 / 62.51 / 57.19	68.08 / 76.92 / 72.06	84.79 / 87.68 / 86.03	91.97 / 91.91 / 91.73
	HPM-MVS$_{\text{fast}}$	82.95 / 57.36 / 67.39	91.17 / 73.20 / 80.86	96.51 / 86.22 / 90.89	98.23 / 92.41 / 94.97
	HPM-MVS	82.93 / **65.16** / **72.73**	90.66 / **79.50** / **84.58**	96.13 / **89.98** / **92.88**	97.97 / **95.59** / **96.22**
Test	*i) Traditional*				
	COLMAP [4]	83.75 / 50.90 / 61.27	91.97 / 62.98 / 73.01	96.75 / 75.74 / 83.96	98.25 / 84.54 / 90.40
	PCF-MVS [3]	72.70 / 70.10 / 70.95	82.15 / 79.29 / 80.38	89.12 / 86.77 / 87.74	92.12 / 91.26 / 91.56
	ACMM [10]	82.11 / 64.35 / 70.80	90.65 / 74.34 / 80.78	96.30 / 83.72 / 89.14	98.05 / 88.77 / 92.96
	ACMP [11]	82.64 / 66.24 / 72.30	90.45 / 75.58 / 81.51	95.71 / 84.00 / 89.01	97.47 / 88.71 / 92.62
	ACMMP [9]	84.22 / 70.81 / 76.02	91.91 / 82.10 / 85.89	96.61 / 90.39 / 93.24	98.05 / 94.67 / 96.27
	ii) Learning				
	PatchMatchNet [7]	54.60 / 67.07 / 59.84	79.71 / 77.46 / 73.12	85.22 / 86.83 / 85.85	91.98 / 92.05 / 91.91
	IterMVS [6]	63.12 / 63.77 / 62.79	84.73 / 76.49 / 80.06	90.18 / 80.67 / 84.88	96.92 / 88.34 / 92.29
	MVSTER [8]	63.29 / **73.25** / 67.16	77.09 / 82.47 / 79.01	89.32 / 89.25 / 88.84	94.21 / 92.71 / 92.30
	HPM-MVS$_{\text{fast}}$	**84.59** / 67.86 / 74.30	**92.50** / 80.25 / 85.35	**97.01** / 90.38 / 93.40	**98.32** / 94.89 / 96.51
	HPM-MVS	84.12 / 72.60 / **77.28**	92.13 / **83.25** / **87.11**	96.81 / **91.62** / **94.02**	98.11 / **95.41** / **96.69**

Tables 2 and 3 list the runtime of each stage for HPM-MVS$_{\text{fast}}$ and HPM-MVS, respectively. Based on the results, the following conclusions can be drawn: 1) the time consumption of planar prior model construction takes a small proportion; 2) the auxiliary time of planar prior construction and prior assistance is continuously reduced because it is a process of continuous optimization.

E. More Evaluation Results

In order to show a more comprehensive demonstration of the excellent results of our methods, we supplement the point cloud evaluation results on the ETH3D [5] benchmark at more thresholds. As shown in Table 4, both HPM-MVS and HPM-MVS$_{\text{fast}}$ can produce impressive results.

F. Visualization of Point Clouds

Figs. 1 and 2 show the reconstructed point clouds on the Tanks & Temples [2] datasets. Figs. 3, 4, 5 and 6 present the qualitative results on the ETH3D [5] benchmark.

References

[1] 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. 1

[2] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4):1–13, 2017. 1, 3, 5, 6

[3] Andreas Kuhn, Shan Lin, and Oliver Erdler. Plane completion and filtering for multi-view stereo reconstruction. In *Proceedings of the German Conference on Pattern Recognition*, pages 18–32. Springer, 2019. 3

[4] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Proceedings of the IEEE European Conference on Computer Vision*, pages 501–518. Springer, 2016. 1, 2, 3

[5] Thomas Schops, Johannes L Schonberger, 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3260–3269, 2017. 1, 3, 7, 8, 9, 10

[6] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, and Marc Pollefeys. Itermvs: Iterative probability estimation for efficient multi-view stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8606–8615, 2022. 3

[7] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 14194–14203, 2021. 3

[8] Xiaofeng Wang, Zheng Zhu, Guan Huang, Fangbo Qin, Yun Ye, Yijia He, Xu Chi, and Xingang Wang. Mvster: Epipolar transformer for efficient multi-view stereo. In *Proceedings of*

the IEEE European Conference on Computer Vision, pages 573–591. Springer, 2022. 3

[9] Qingshan Xu, Weihang Kong, Wenbing Tao, and Marc Pollefeys. Multi-scale geometric consistency guided and planar prior assisted multi-view stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1, 2, 3

[10] 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. 1, 2, 3

[11] Qingshan Xu and Wenbing Tao. Planar prior assisted patchmatch multi-view stereo. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12516–12523, 2020. 2, 3

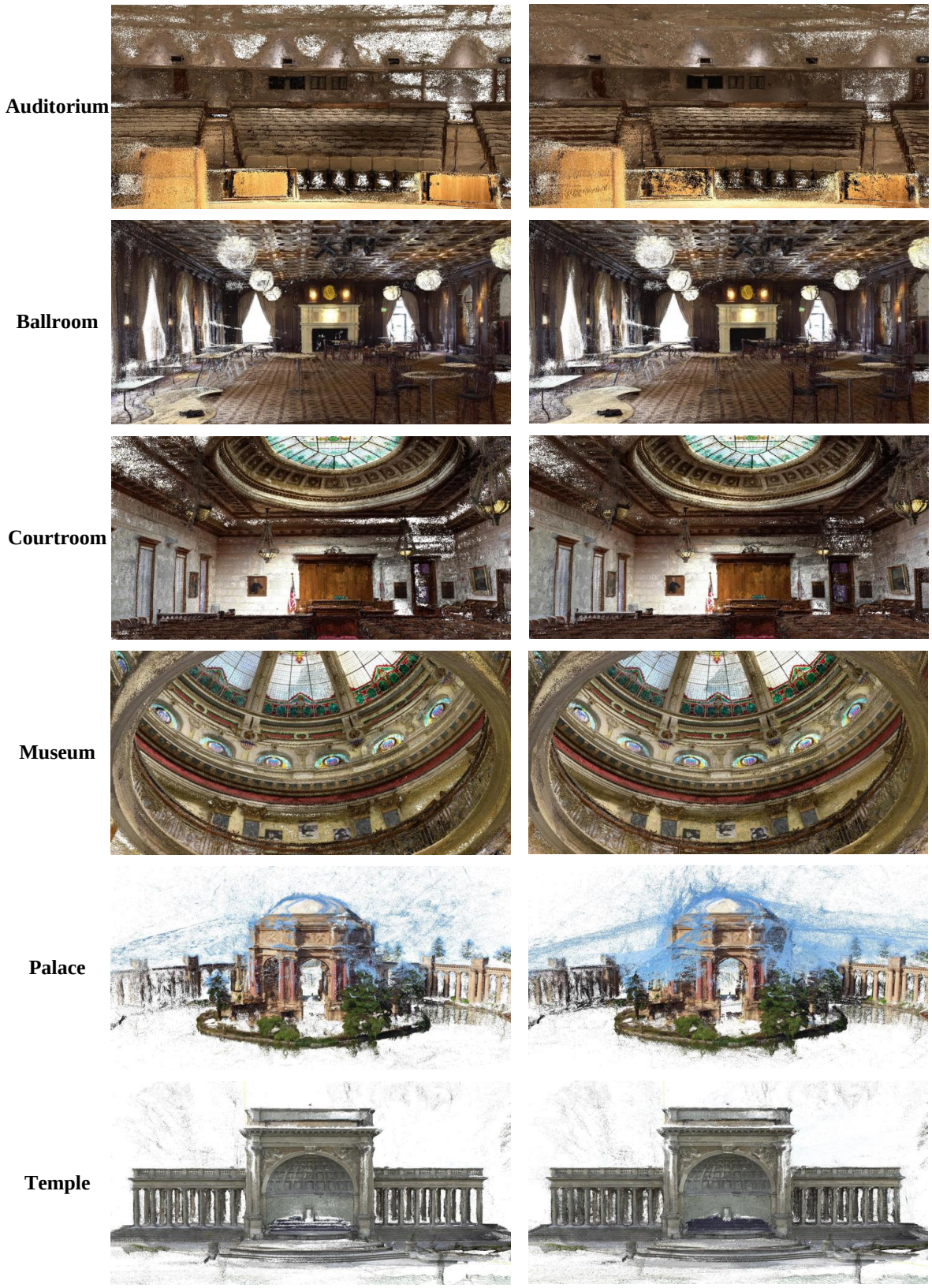

Figure 1: Reconstruction results on Tanks & Temples [2] Advanced set. Left: HPM-MVS$_{\text{fast}}$. Right: HPM-MVS

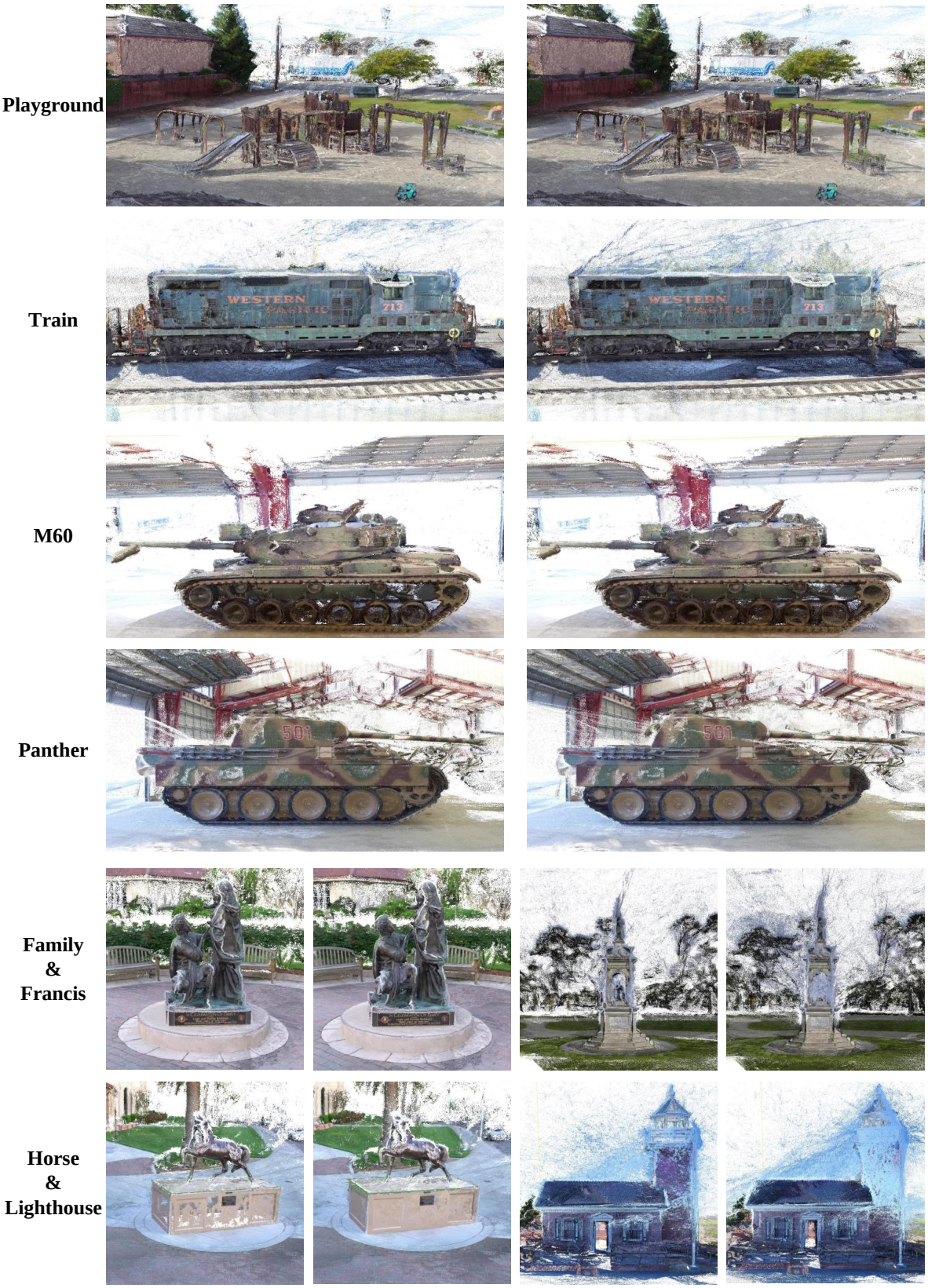

Figure 2: Reconstruction results on Tanks & Temples [2] Intermediate set.

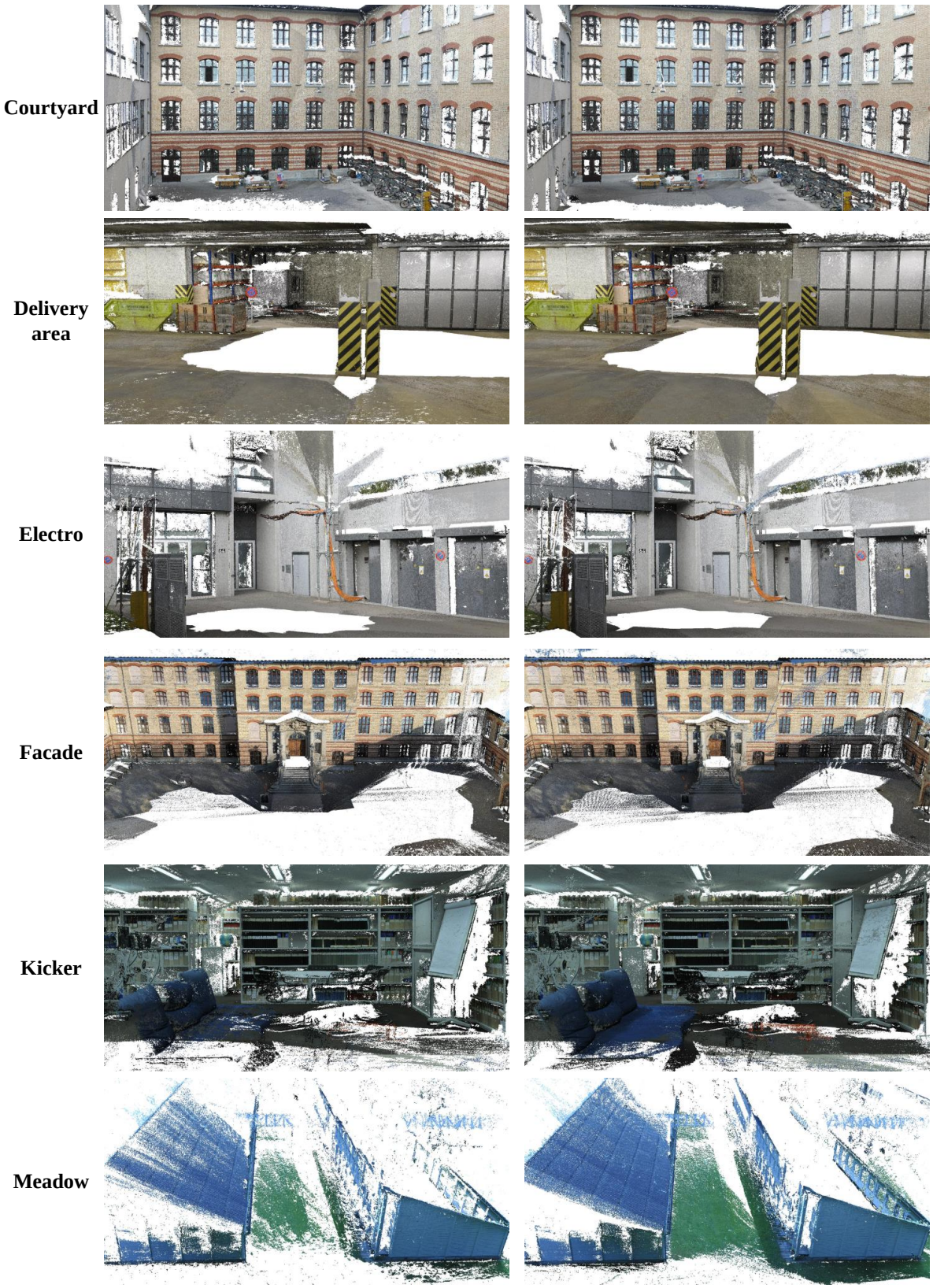

Figure 3: Reconstruction results on ETH3D [5] Training set (1/2).

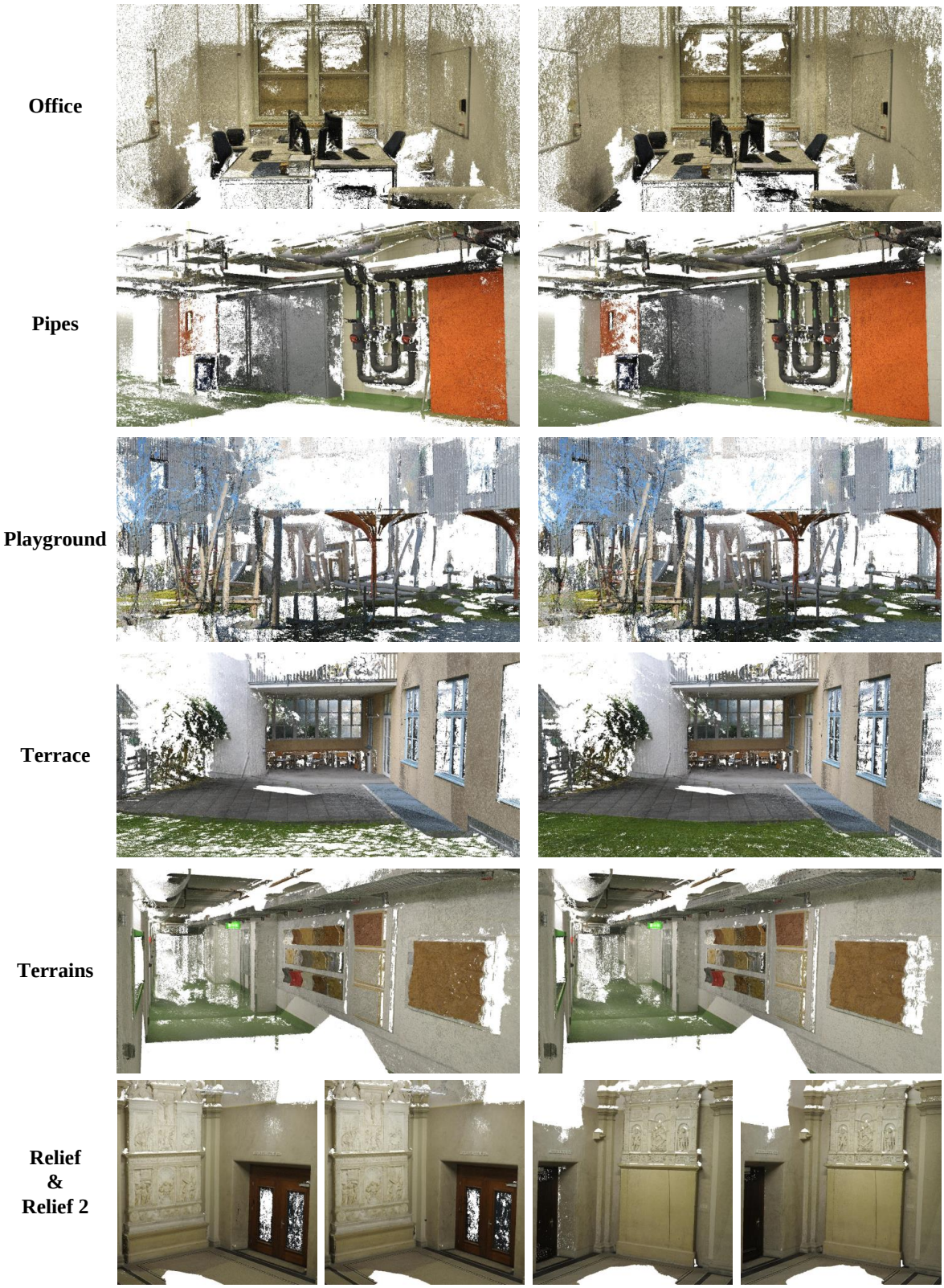

Figure 4: Reconstruction results on ETH3D [5] Training set (2/2).

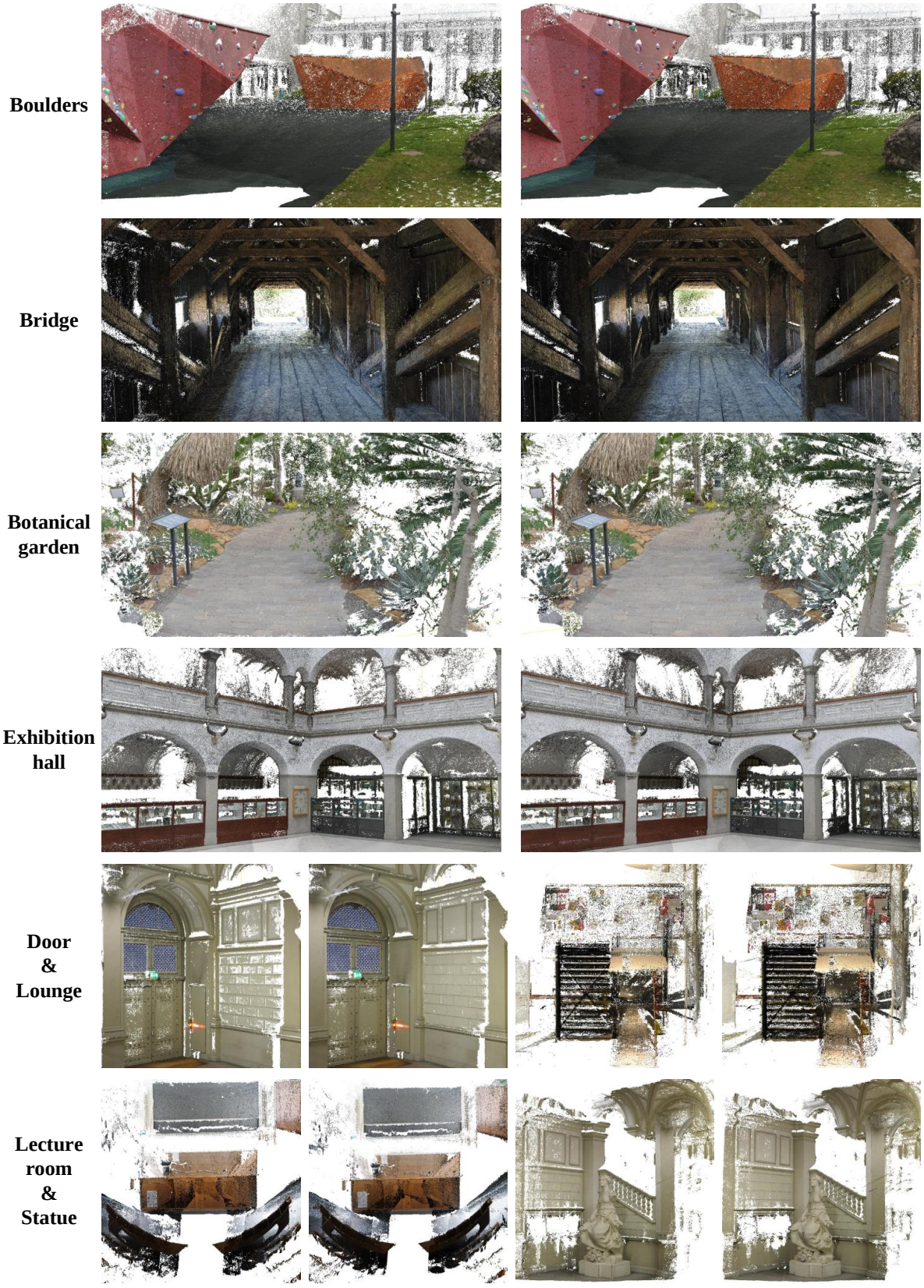

Figure 5: Reconstruction results on ETH3D [5] Test set (1/2).

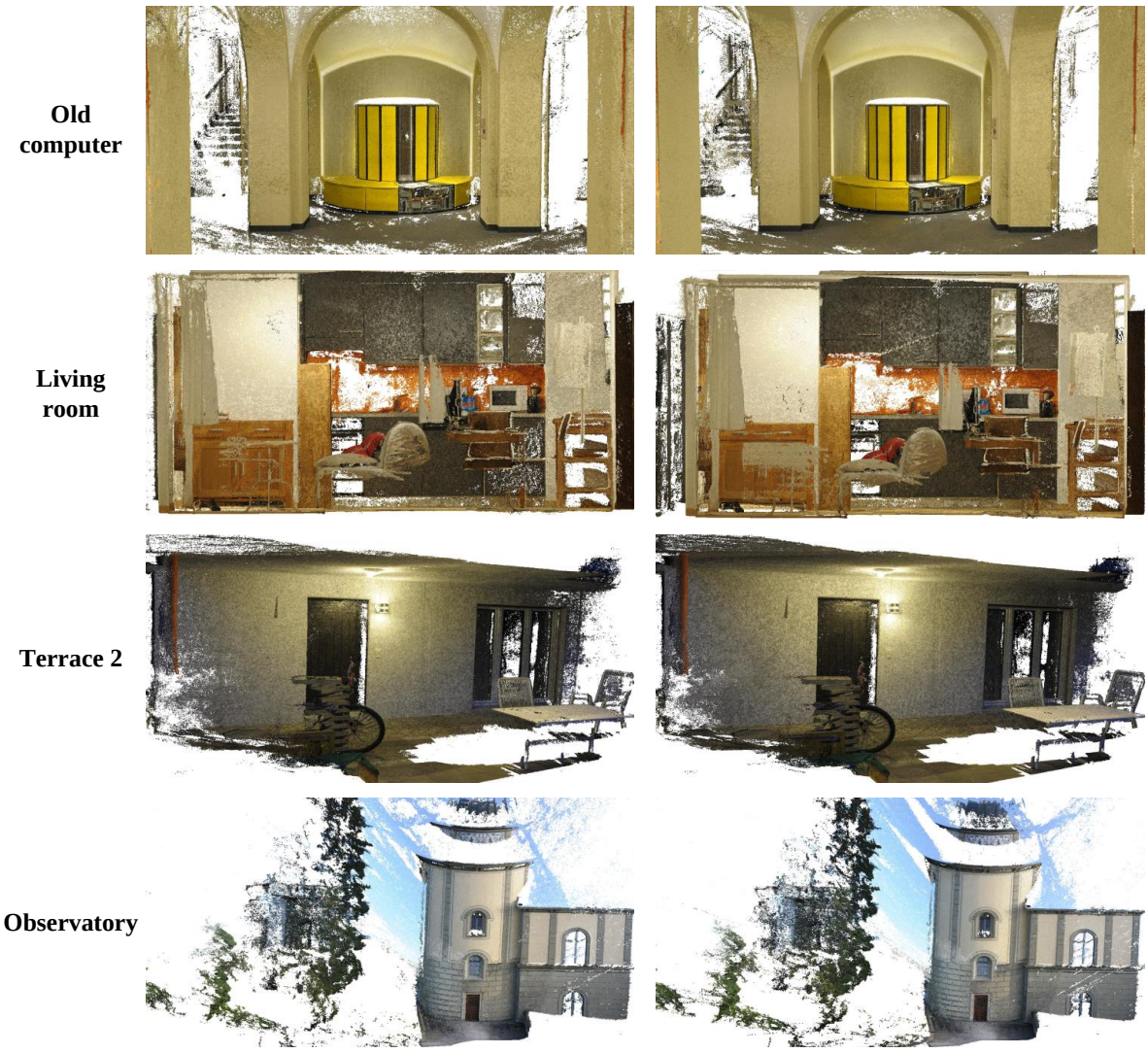

Figure 6: Reconstruction results on ETH3D [5] Test set (2/2).