Supplementary Material: Initialization and Alignment for Adversarial Texture Optimization

This supplementary is structured as follows:

- Sec. A provides details regarding the scenes;
- Sec. B describes implementation details;
- Sec. C provides more quantitative results;
- Sec. D gives more ablation studies;
- Sec. E shows more qualitative results.

A UofI Texture Scenes Details

We provide detailed scene statistics in Tab. S1. As mentioned in Sec. 4.1, we collect a dataset of 11 scenes: four indoor and seven outdoor scenes. This dataset consists of a total of 2807 frames, of which 91, 2052, and 664 are of resolution 480×360, 960×720, and 1920×1440 respectively. For each scene, we reserve 10% of the views for evaluation. In contrast, prior work [24] "select(s) 10 views uniformly distributed from the scanning video" for evaluation while using up to thousands of frames for texture generation. The studied setting is hence more demanding.

B Implementation Details

B.1 Network Structure for `TexSmooth`

As mentioned in Sec. 3.2, we utilize a convolutional neural network for the discriminator D in Eq. (13). Specifically, we follow [24]'s code release[3] to utilize five convolutional layers with structures (6, 64, 2), (64, 128, 2), (128, 128, 2), (128, 128, 1), (128, 1, 1), where (in, out, s) indicates the number of input channels, the number of output channels, and stride respectively. All layers employ a 4×4 kernel and use `VALID` padding. Regarding activation functions, the first four layers contain leakyReLUs while the last one uses a sigmoid activation.

B.2 Hyperparameters

The weight for the $\mathcal{L}_1$ loss in Eq. (13) is $\lambda = 10$. We decay it by a factor of 0.8 every 960 steps. We use Adam [26] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rates for the texture and discriminator are set to $1e^{-3}$ and $1e^{-4}$ respectively. We run `AdvOptim` for 4000 iterations for AdvTex-C and our `TexSmooth` stage, following the code release.[3] For `AdvTex-B`, `AdvOptim` runs for 50000 iterations as stated in their paper [24].

[3] https://github.com/hjwdzh/AdversarialTexture

Table S1: **Statistics of UofI Texture Scenes.** For each column, the format is indicated below the header. We reserve 10% views for evaluation. "Angular Diff" measures the the angular differences between test set view directions and their nearest neighbour in the training sets.

	Mesh #Faces	#Views total (test)	Resolution value (#count)	Angular Diff avg (min/max)
1	77,528	160 (16)	1920×1440 (160)	2.04° (0.38°/4.26°)
2	104,684	195 (20)	960×720 (195)	0.85° (0.06°/2.46°)
3	132,196	94 (10)	1920×1440 (3) 480×360 (91)	4.58° (1.14°/13.8°)
4	65,832	43 (5)	960×720 (43)	3.48° (0.12°/6.66°)
5	143,108	584 (59)	960×720 (584)	1.28° (0.19°/3.18°)
6	168,664	372 (38)	1920×1440 (372)	1.45° (0.36°/3.13°)
7	77,627	233 (24)	960×720 (233)	3.24° (0.67°/12.6°)
8	80,240	352 (36)	960×720 (352)	1.60° (0.27°/6.89°)
9	199,976	347 (35)	960×720 (347)	1.81° (0.21°/4.55°)
10	69,484	129 (13)	1920×1440 (129)	1.12° (0.19°/2.59°)
11	149,176	298 (30)	960×720 (298)	1.13° (0.27°/2.60°)

B.3 `AdvOptim` Reimplementation

As mentioned in Sec. 4.1, we re-implement `AdvOptim` using PyTorch based on the official TensorFlow (TF) code. To verify the correctness of our implementation, we compare results from the official code release and our re-implemented version on the Chair dataset from [24], which contains 35 scanned chairs. Specifically, similar to Sec. 4.1, we reserve 10% of the views from each scan for evaluation, resulting in 14,836 training views and 1663 test views. As can be seen from the 1st *vs*. 2nd row in Tab. S2, we observe 0.830 *vs*. 0.828 SSIM, 0.163 *vs*. 0.167 LPIPS, 0.068 *vs*. 0.069 S_3, and 2.052 *vs*. 2.047 Grad. This verifies the correctness of our re-implementation.

B.4 Determine $\mathcal{T}$ resolution

To determine the H and W of the texture $\mathcal{T}$, we use the following three steps: 1) for each of the $|\mathcal{Y}|$ planes mentioned in Sec. 3.1's "Overlap Detection", we use a resolution of 512^2, 1024^2, or 2048^2 based on whether the major RGB resolution's larger side is 480, 960, or 1920; 2) all $|\mathcal{Y}|$ planes are concatenated to obtain the texture $\mathcal{T}$'s H and W; 3) For a fair comparison, AdvTex baselines use the same resolution as ours.

Table S2: **Quantitative results on Chair dataset**. We report in the form of `mean`±`std`.

		SSIM↑	LPIPS↓	S_3 ↓	Grad↓
1	TF	0.830±0.072	0.163±0.072	0.068±0.020	2.052±1.267
2	PyTorch	0.828±0.072	0.167±0.073	0.069±0.020	2.047±1.275
3	`TexInit` + `TexSmooth`	0.827±0.073	0.167±0.073	0.068±0.019	2.071±1.289

C More Quantitative Results

C.1 Scene-level Quantitative Results

We provide scene-level quantitative results in Tab. S5. As can be seen from the number for the Best or 2nd-Best results (last two rows), our technique improves upon all baselines. Specifically, ours dominates the count (ours *vs*. runner-up) for "best" (17 *vs*. 14), "2nd-best" (14 *vs*. 13), and "best or 2nd-best" (31 *vs*. 18).

C.2 Results on Existing Dataset

We apply our `TexInit` + `TexSmooth` on the Chair dataset. We directly utilize the provided conformal mapping to ensure a fair comparison. Quantitative results are shown in 2nd *vs*. 3rd row of Tab. S2. Our method performs on par with the baseline. This is expected as we do not observe geometry and images of those scans to be misaligned. Beneficially and as expected, the proposed technique doesn't harm the result if geometry and images are well aligned.

C.3 Robustness to Inaccurate Camera Poses

To verify the robustness of the proposed pipeline, we conduct studies by deliberately adding more noise to camera poses in the training split. Concretely, given a camera pose with rotation (r_x, r_y, r_z) (represented in Euler angles) and translation (t_x, t_y, t_z), we add uniformly-sampled noise, *i.e.*, we have $\widehat{r}_x = r_x + \epsilon_{r_x}$, where $\epsilon_{r_x} \sim \mathcal{U}(-0.05 \cdot |r_x|, 0.05 \cdot |r_x|)$, and analogously $\widehat{r}_y, \widehat{r}_z, \widehat{t}_x, \widehat{t}_y, \widehat{t}_z$. We apply AdvTex-B/C and ours on data with these corrupted camera poses. Tab. S3 corroborates the robustness of the method: we outperform baselines on SSIM (0.456 *vs*. 0.452, ↑ is better), LPIPS (0.472 *vs*. 0.490, ↓ is better), and sharpness S_3 (0.142 *vs*. 0.153, ↓ is better).

D More Ablations

D.1 Unary *vs*. Pairwise Cues

As stated in Eq. (6), we use pure-unary cues for `TexInit`. To verify this design choice, we ablate with a setup where `TexInit` considers both unary and pairwise

Table S3: **Evaluation with noise added to camera poses on UofI Texture Scenes**. Results are in the form of mean±std. Ours is the most robust.

		SSIM↑	LPIPS↓	S_3 ↓	Grad↓
1-1	AdvTex-B	0.378±0.158	0.503±0.091	0.160±0.040	9.193±4.562
1-2	AdvTex-C	0.452±0.191	0.490±0.082	0.153±0.072	**7.949**±4.364
2	Ours	**0.456**±0.196	**0.472**±0.078	**0.142**±0.050	8.177±4.441

cues. Concretely, we consider the following optimization problem:

$$\mathbf{t}^* = \underset{\mathbf{t}}{\operatorname{argmax}} \sum_{i=1}^{|M|} \psi_i(t_i) + \sum_{(i,j)\in\mathcal{A}} \psi_{i,j}(t_i, t_j), \tag{S1}$$

where $\mathcal{A}$ is the adjacency used in Eq. (2). Here ψ_i refers to the *unary* cues in Eq. (6) while $\psi_{i,j}$ captures *pairwise* ones. Therefore, besides $\mathtt{C}_1$, $\mathtt{C}_2$, and $\mathtt{C}_3$ discussed in Sec. 3.1, we take into account another cue $\mathtt{C}_4$:

$$\psi_{i,j}(t_i, t_j) \doteq \psi_{i,j}^{\mathtt{C}_4}(t_i, t_j). \tag{S2}$$

• **Explicit adjacency encouragement** ($\mathtt{C}_4$). Intuitively, adjacent triangles $\mathtt{Tri}_i$ and $\mathtt{Tri}_j$ maintain smoothness if they are assigned textures from the same frame. We encourage this choice using $\psi_{i,j}^{\mathtt{C}_4}(t_i, t_j) = \mathbb{1}(t_i = t_j)$ and set $\omega_4 > 0$.

In our experiments, we set $\omega_4 = 1$ for `TexInit`. The `TexSmooth` stage remains the same. Quantitative results are shown in Column 1-1 *vs*. 1-2 in Tab. S6. We do not observe a big difference when integrating pairwise cues. Concretely, when comparing pairwise *vs*. unary only cues, we have 0.601 *vs*. 0.602 (SSIM), 0.305 *vs*. 0.309 (LPIPS), 0.120 *vs*. 0.120 (S_3), and 6.872 *vs*. 6.871 (Grad), which corroborate our design.

D.2 View Sparsity Analysis

To understand whether our framework is sensitive to the sparsity of training views, we conduct an ablation: 1) we reserve 10% of the views, which are uniformly sampled, for evaluation. As a consequence, we have 90% of all views that can be used for training; 2) within those 90% of all views, we again uniformly sample images for optimization every k views. We use $k \in \{1, 2, 3, 4, 5\}$. Note, the results reported in Tab. 1, Tab. 2, and Tab. S5 are situations where $k = 1$, namely all 90% views are used. Quantitative results are reported in Column 1-1 to Column 5 in Tab. S6. As expected, when the number of training views decreases, we observe the results' quality to drop. However, our framework is robust to the view sparsity as the performance gap is small. Specifically, we observe best *vs*. worst results as 0.604 *vs*. 0.580 (SSIM), 0.303 *vs*. 0.316 (LPIPS), 0.120 *vs*. 0.127 (S_3), and 6.859 *vs*. 7.026 (Grad). The reason that $k = 2$ performs slightly better than $k = 1$ is that 1) $k = 2$ still provides dense enough views for `TexInit` and `TexSmooth`; 2) the initialization from `TexInit` can contain less seams as less views are considered.

Table S4: **Patchwise alignment on UofI Texture Scenes**. Results are in the form of `mean`±`std`. $X \times Y$ in the '#Patches' column denotes the number of patches created by splitting along height (X) and width (Y). See Fig. S1 for qualitative results.

	#Patches	SSIM↑	LPIPS↓	S_3 ↓	Grad↓
1	1×1	**0.602**±0.189	**0.309**±0.086	**0.120**±0.058	**6.871**±4.342
2	2×2	0.556±0.184	0.330±0.069	0.123±0.055	7.122±4.306
3	4×4	0.545±0.175	0.338±0.064	0.122±0.054	7.163±4.257

(a) Patch 1x1 (b) Patch 2x2 (c) Patch 4x4

Fig. S1: **Patch-wise alignment on UofI Texture Scenes**. Highlighted issues: sofa and plant colors are mapped to the wall in (b,c).

D.3 Patchwise Alignment

We assess the results of patch-wise alignment in our pipeline. Tab. S4 and Fig. S1 verify: a patch-wise method ignores global content and is inferior.

E More Qualitative Results

E.1 Alignment Visualizations

As mentioned in Sec. 3.2, we utilize the Fourier transformation to align ground-truth image and rendering from $\mathcal{T}_{\text{init}}$. We show qualitative results of the alignment for all 11 scenes in Fig. S2. As can be seen clearly, our module successfully mitigates the misalignment, verifying the efficacy.

E.2 Complete Qualitative Results

We display qualitative results for the six scenes that are not shown in the main submission in Fig. S3. Compared to AdvTex-C, our `TexInit` + `TexSmooth` framework largely reduces artifacts (Fig. S3e *vs*. Fig. S3f), yielding more perceptual similarity while maintaining more sharpness (Fig. S3g).

In addition, we provide an HTML page in the supplementary to display more rendering comparisons from those 10% evaluation views.

Table S5: **Scene-level quantitative results on UofI Texture Scenes.** We use **bold** and underline to mark best and 2nd-best results for each row respectively. If one value's difference from its higher-rank counterpart is no larger than 0.001, we treat them as the same.

		1	2	3	4	5	6	7	8	9	10
		L2Avg	ColorMap	TexMap	MVSTex	AdvTex-B	AdvTex-C	$\mathcal{T}_{\text{init}}$ Only	w/o $\mathcal{T}_{\text{init}}$	w/o `Align`	Ours
Scene 1	SSIM↑	0.866	0.831	0.608	0.718	0.774	0.834	0.769	0.863	0.845	**0.876**
	LPIPS↓	0.235	0.361	0.387	0.260	0.227	0.261	0.251	0.215	0.228	**0.187**
	S_3 ↓	**0.042**	0.058	0.088	0.073	0.064	0.051	0.070	0.043	0.046	**0.041**
	Grad↓	1.806	2.085	3.232	2.977	2.349	2.073	2.414	1.816	1.936	**1.724**
Scene 2	SSIM↑	0.584	0.428	0.384	0.494	0.486	0.541	0.521	0.592	0.532	**0.626**
	LPIPS↓	0.319	0.653	0.433	**0.221**	0.288	0.290	0.271	0.231	0.294	0.226
	S_3 ↓	0.227	0.342	0.253	0.172	0.172	0.180	0.190	**0.165**	0.174	**0.164**
	Grad↓	8.910	11.09	11.75	10.28	9.909	9.499	9.749	8.488	9.707	**8.067**
Scene 3	SSIM↑	**0.651**	0.594	0.465	0.575	0.534	0.605	0.561	0.630	0.585	0.636
	LPIPS↓	0.362	0.533	0.459	**0.301**	0.383	0.344	0.354	0.324	0.348	0.309
	S_3 ↓	0.135	0.170	0.188	0.129	0.153	**0.121**	0.132	0.128	0.124	**0.121**
	Grad↓	5.721	5.707	8.203	6.836	7.468	6.051	6.566	6.118	6.238	**5.659**
Scene 4	SSIM↑	0.217	0.182	0.168	0.155	0.133	0.187	0.176	0.198	0.180	**0.244**
	LPIPS↓	0.656	0.845	0.468	**0.370**	0.575	0.540	0.430	0.516	0.500	0.440
	S_3 ↓	0.310	0.441	0.168	**0.112**	0.152	0.143	0.118	0.141	0.128	0.121
	Grad↓	16.79	18.35	16.68	**15.78**	18.05	16.07	16.76	15.95	16.13	16.09
Scene 5	SSIM↑	**0.480**	0.430	0.397	0.412	0.422	0.448	0.422	0.474	0.430	0.472
	LPIPS↓	0.427	0.615	0.391	**0.268**	0.343	0.351	0.308	0.332	0.321	0.293
	S_3 ↓	0.326	0.403	0.267	0.213	0.225	0.227	0.223	0.220	0.210	**0.205**
	Grad↓	10.38	11.80	11.83	10.62	10.85	9.359	11.18	**9.230**	10.12	9.807
Scene 6	SSIM↑	**0.684**	0.615	0.330	0.392	0.509	0.598	0.480	0.637	0.591	0.642
	LPIPS↓	0.374	0.615	0.674	0.553	0.386	0.403	0.419	**0.343**	0.408	0.363
	S_3 ↓	**0.058**	0.099	0.122	0.109	0.129	0.066	0.127	0.064	0.075	0.067
	Grad↓	**3.020**	3.552	4.925	4.790	4.500	3.435	4.476	3.360	3.551	3.256
Scene 7	SSIM↑	**0.423**	0.378	0.343	0.346	0.362	0.384	0.374	0.411	0.373	0.396
	LPIPS↓	0.465	0.650	0.475	**0.347**	0.436	0.442	0.356	0.448	0.392	0.367
	S_3 ↓	0.280	0.385	0.262	0.215	0.216	0.245	0.220	0.239	0.203	**0.200**
	Grad↓	12.16	13.98	13.27	12.93	13.27	12.09	13.65	**11.60**	12.11	11.91
Scene 8	SSIM↑	**0.846**	0.808	0.265	0.629	0.679	0.815	0.740	0.839	0.816	0.842
	LPIPS↓	0.269	0.498	0.680	0.396	0.328	0.286	0.349	0.259	0.278	**0.250**
	S_3 ↓	0.060	0.084	0.135	0.117	0.096	0.063	0.080	0.058	0.060	**0.055**
	Grad↓	2.062	2.489	4.627	4.405	3.181	2.275	2.877	2.070	2.257	**2.011**
Scene 9	SSIM↑	**0.545**	0.502	0.306	0.373	0.376	0.450	0.383	0.500	0.469	0.508
	LPIPS↓	0.483	0.709	0.509	**0.371**	0.460	0.520	0.419	0.474	0.454	0.451
	S_3 ↓	0.194	0.254	0.197	0.146	0.146	0.163	**0.132**	0.159	0.138	0.143
	Grad↓	6.988	8.042	9.815	8.751	8.582	7.514	8.380	**6.765**	6.985	6.814
Scene 10	SSIM↑	**0.853**	0.800	0.493	0.718	0.716	0.810	0.744	0.828	0.816	0.839
	LPIPS↓	0.260	0.403	0.492	0.305	0.282	0.244	0.267	0.204	0.240	**0.198**
	S_3 ↓	**0.043**	0.059	0.088	0.072	0.099	0.051	0.081	0.047	0.049	0.045
	Grad↓	**2.159**	2.476	3.740	3.366	3.295	2.564	3.081	2.416	2.421	2.283
Scene 11	SSIM↑	**0.563**	0.520	0.374	0.428	0.454	0.518	0.443	0.546	0.513	0.544
	LPIPS↓	0.398	0.510	0.399	**0.292**	0.350	0.337	0.334	0.307	0.345	0.312
	S_3 ↓	0.229	0.278	0.198	0.167	0.178	0.177	0.179	0.165	0.167	**0.158**
	Grad↓	**7.742**	8.100	10.04	9.435	9.057	7.946	9.878	7.684	8.225	7.957
Count	Best	14	0	0	9	0	1	1	5	0	**17**
	2nd-Best	4	1	0	0	0	2	4	13	3	**14**

Table S6: **Scene-level ablation results on UofI Texture Scenes.** The fractions below each header indicate the portion of training views utilized during optimization. Note, we do not consider the already-reserved 10% views for evaluation. Namely, 1/1 means we use all 90% training views. For Column 1-1, we use pairwise cues for `TexInit` while only unary cues are utilized for the remaining columns. We report `mean`±`std`.

		1-1 1/1-pairwise	1-2 1/1	2 1/2	3 1/3	4 1/4	5 1/5
Scene 1	SSIM↑	0.876	0.876	0.873	0.868	0.831	0.826
	LPIPS↓	0.186	0.187	0.185	0.187	0.216	0.213
	S_3 ↓	0.040	0.041	0.042	0.044	0.058	0.063
	Grad↓	1.727	1.724	1.742	1.770	2.045	2.120
Scene 2	SSIM↑	0.622	0.626	0.624	0.617	0.614	0.610
	LPIPS↓	0.219	0.226	0.218	0.215	0.221	0.215
	S_3 ↓	0.162	0.164	0.160	0.161	0.161	0.159
	Grad↓	8.070	8.067	8.053	8.105	8.162	8.124
Scene 3	SSIM↑	0.637	0.636	0.657	0.630	0.601	0.639
	LPIPS↓	0.307	0.309	0.293	0.312	0.314	0.265
	S_3 ↓	0.120	0.121	0.117	0.125	0.133	0.112
	Grad↓	5.639	5.659	5.445	5.706	6.008	5.435
Scene 4	SSIM↑	0.237	0.244	0.258	0.240	0.219	0.204
	LPIPS↓	0.442	0.440	0.440	0.444	0.466	0.417
	S_3 ↓	0.123	0.121	0.122	0.128	0.128	0.114
	Grad↓	16.21	16.09	16.09	16.33	16.00	16.45
Scene 5	SSIM↑	0.473	0.472	0.471	0.473	0.473	0.474
	LPIPS↓	0.294	0.293	0.288	0.288	0.290	0.287
	S_3 ↓	0.206	0.205	0.205	0.205	0.205	0.203
	Grad↓	9.853	9.807	9.866	9.830	9.850	9.889
Scene 6	SSIM↑	0.644	0.642	0.643	0.627	0.621	0.618
	LPIPS↓	0.364	0.363	0.351	0.351	0.355	0.353
	S_3 ↓	0.066	0.067	0.070	0.073	0.074	0.073
	Grad↓	3.245	3.256	3.284	3.349	3.375	3.352
Scene 7	SSIM↑	0.400	0.396	0.407	0.400	0.400	0.411
	LPIPS↓	0.367	0.367	0.368	0.366	0.366	0.354
	S_3 ↓	0.201	0.200	0.199	0.200	0.201	0.194
	Grad↓	11.88	11.91	11.74	11.94	11.93	11.62
Scene 8	SSIM↑	0.837	0.842	0.827	0.816	0.814	0.815
	LPIPS↓	0.249	0.250	0.257	0.271	0.270	0.268
	S_3 ↓	0.055	0.055	0.060	0.065	0.065	0.065
	Grad↓	2.035	2.011	2.115	2.209	2.224	2.224
Scene 9	SSIM↑	0.509	0.508	0.506	0.498	0.490	0.489
	LPIPS↓	0.440	0.451	0.441	0.441	0.441	0.428
	S_3 ↓	0.143	0.143	0.142	0.141	0.142	0.140
	Grad↓	6.788	6.814	6.787	6.844	6.886	6.852
Scene 10	SSIM↑	0.839	0.839	0.839	0.826	0.802	0.801
	LPIPS↓	0.194	0.198	0.189	0.202	0.221	0.219
	S_3 ↓	0.044	0.045	0.046	0.051	0.067	0.069
	Grad↓	2.256	2.283	2.282	2.379	2.599	2.636
Scene 11	SSIM↑	0.540	0.544	0.537	0.530	0.518	0.523
	LPIPS↓	0.297	0.312	0.310	0.311	0.315	0.311
	S_3 ↓	0.155	0.158	0.157	0.159	0.160	0.159
	Grad↓	7.890	7.957	8.034	8.085	8.204	8.106
Aggregated	SSIM↑	0.601±0.189	0.602±0.189	0.604±0.184	0.593±0.184	0.580±0.180	0.583±0.182
	LPIPS↓	0.305±0.086	0.309±0.086	0.304±0.086	0.308±0.084	0.316±0.082	0.303±0.074
	S_3 ↓	0.120±0.058	0.120±0.058	0.120±0.056	0.123±0.055	0.127±0.051	0.123±0.049
	Grad↓	6.872±4.364	6.871±4.342	6.859±4.321	6.959±4.355	7.026±4.233	6.983±4.296

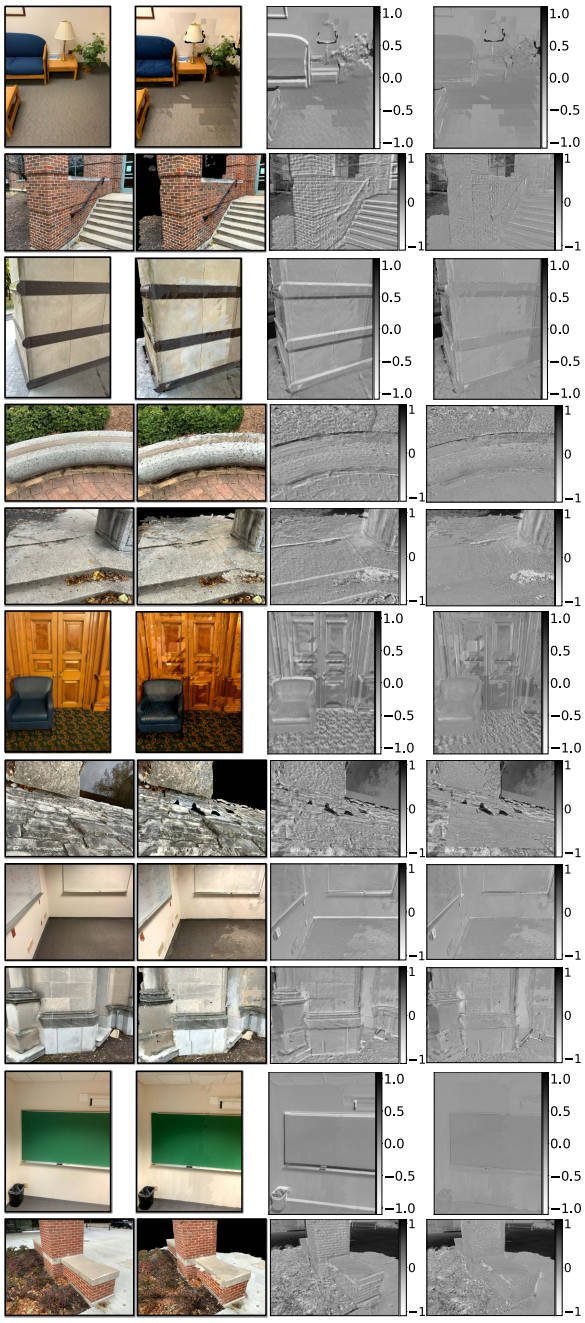
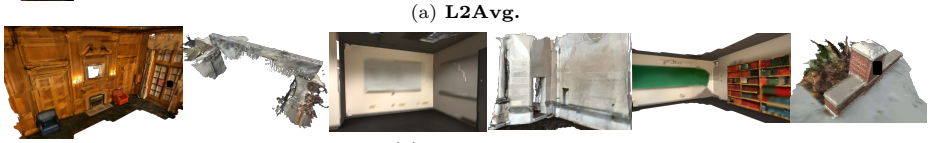
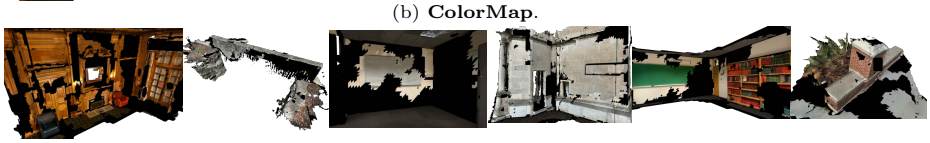


Fig. S2: **Effect of alignment module.** From top to bottom, we show alignment results for Scene 1 to 11 respectively. For each scene, from left to right, we display the ground-truth image (GT), rendering from $\mathcal{T}_{\text{init}}$, difference between GT and rendering before alignment, and difference after alignment.

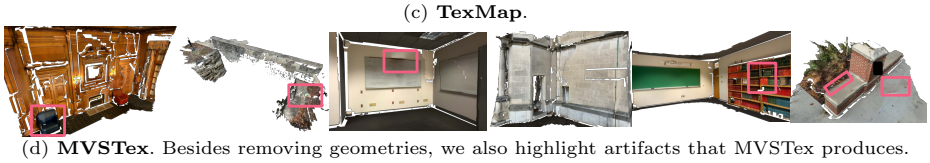
(a) **L2Avg.**



(b) **ColorMap**.

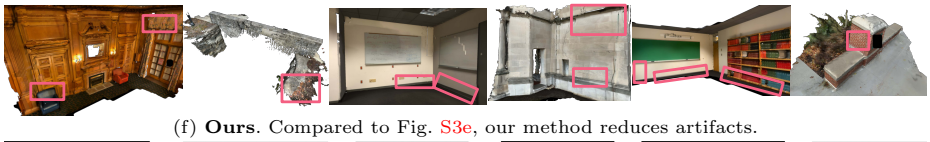


(c) **TexMap**.

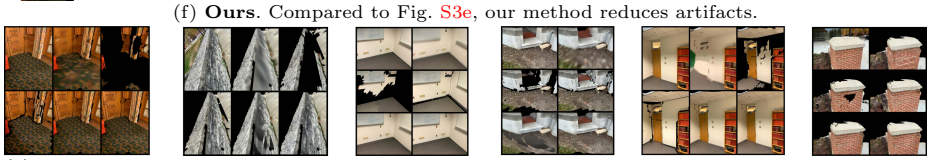


(d) **MVSTex**. Besides removing geometries, we also highlight artifacts that MVSTex produces.

(e) **AdvTex-C**. We highlight artifacts: 1) Scene 6: chair's texture is mapped to the door in the background and the curtain's pattern breaks; 2) Scene 7: leaf colors are fused into the ground; 3) Scene 8: the boundary between lower gray and upper white areas are fused; 4) Scene 9: the color on the wall breaks; 5) Scene 10: on the right: the texture of the bookshelf's lower part is mapped to the floor; on the left: the texture of carpet is mapped to the wall and the wall's color breaks; 6) Scene 11: bricks' cracks are mixed together.



(f) **Ours**. Compared to Fig. S3e, our method reduces artifacts.



(g) **Highlights**. From left to right and top to bottom, we show ground-truth image as well as renderings at the same camera pose with texture from ColorMap, TexMap, MVSTex, AdvTex-C, and ours respectively. **1) Compared to MVSTex:** Besides our method producing more complete geometry, we observe 1.1) Scene 6: MVSTex produces an apparent split of the color on the carpet; 1.2) Scene 7: MVSTex produces purple-like color at the far-end of the wall; 1.3) Scene 8: for MVSTex, the top-middle of the image has an apparent color seam on the wall; 1.4) Scene 9: MVSTex produces an apparent color seam at the far-end of the red ceramic ground; 1.5) Scene 10: MVSTex produces apparent color seams on the door; 1.6) Scene 11: MVSTex generates large color seams on the meadow. **2) Compared to AdvTex-C:** 2.1) Scene 6: the carpet's pattern of our method is sharper and the color is smoother; 2.2) Scene 7: our method generates a sharper pattern of the wall; 2.3) Scene 8: our method produces a sharper boundary between the lower gray and upper white areas of the wall; 2.4) Scene 9: our method produces sharper boundaries of the architecture and there are no color breaks on the ground; 2.5) Scene 10: our reconstruction has a sharper boundary for the door while AdvTex-C fused the door and wall; 2.6) Scene 11: our method generates sharper cracks on the wall.

Fig. S3: **Qualitative results on UofI Texture Scenes.** For each method, we show results for Scene 6 to 11 from left to right. Best viewed in color and zoomed-in.