

Supplementary Material

EllipssianNet: Image-guided Sampling of 2D Gaussians for Gaussian Splatting

A. Ablation study

To evaluate the effectiveness of EllipssianNet, we conducted an ablation study across two architectural variations: (1) a variant without the Gradient Decoder, and (2) a variant where the Covariance Decoder directly estimates the full 2×2 covariance matrix σ . We denote them as 'w/o Grad' and ' 2×2 Cov' respectively. The variants were compared against the default EllipssianNet architecture, denoted as 'Default' to assess the impact of network design on approximating 2D Gaussians. For all three cases, the training loss weights $\lambda_C, \lambda_\Sigma, \lambda_G$ in Equation 6 were set to 1. To ensure convergence, covariance values were normalized: for the Default and w/o Grad, the Cholesky components were log-transformed, while for the 2×2 Cov variant, the values were normalized relative to the image size.

We created 10K unseen data for this ablation study. Each model variant predicted the center, covariance and gradient maps. For each model variant, we computed the Mean Squared Error (MSE) across different output components to quantitatively assess their accuracy. For covariance, the models recovered the full 2×2 covariance matrix from the estimated values with de-normalizing, and MSE was calculated accordingly.

The results, summarized in Table 6, show that training with Cholesky components yields significantly higher accuracy compared to directly predicting the full 2×2 covariance matrix. In addition, incorporating the Gradient as an auxiliary output further improves performance by enriching feature representations prior to the final prediction.

B. Implementation details of EllipssianNet

B.1. Cleaning Step

For the cleaning step, which converts Figure 7c into Figure 7d, we use the predicted center map, \hat{C} , judging the validity of each ellipse. For this, we first compute the fast-decaying function (given in Equation 1) for every ellipse. Figure 13a visualizes the functions together with bounding boxes that encompass the areas containing function values that exceed a certain threshold. We then apply a Non-Maximum Suppression-inspired filtering to the bounding boxes.

In Figure 13a, consider the bounding box (bbox) located at the upper-right corner, which is redrawn in Figure 13b. The same bbox is placed at \hat{C} , as shown in Figure 13c. The 2D bboxes are linearized into vectors and normalized. Then, their dot product is computed. The result with Fig-

Table 6. MSE comparison of different EllipssianNet architectures.

Model	Center ↓	Covariance ↓	Total ↓
Default	0.000243	0.000090	0.000334
w/o Grad	0.000268	0.000105	0.000373
2×2 Cov	0.000268	0.001109	0.001377

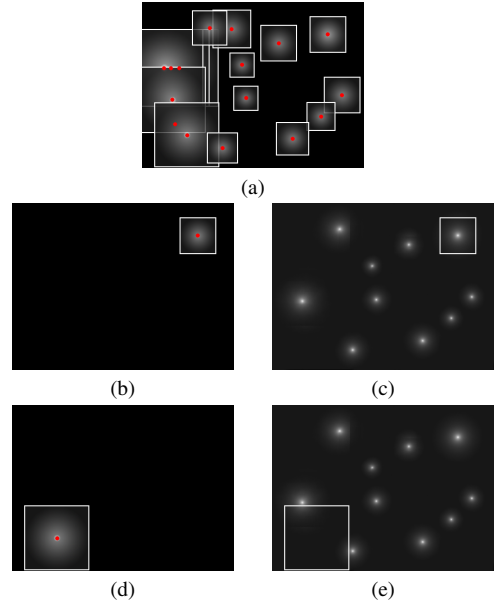


Figure 13. Cleaning unnecessary ellipses: (a) Fast-decaying functions and their bounding boxes. (b) A local maximum. (c) The bounding box in (b) is placed at \hat{C} . (d) Another local maximum. (e) The bounding box in (d) is placed at \hat{C} .

ure 13b and Figure 13c would be close to one, and the ellipse is considered valid.

In contrast, consider another example presented in Figure 13d and Figure 13e. In this example, the dot product would be far smaller than one. Then, the ellipse centered on the red dot is taken as invalid and then discarded.

B.2. Image pre-processing

To ensure robustness and efficiency, we incorporated a carefully designed pre-processing stage. When an input image requires more Gaussians than the model's capacity, such as in large or overly complex scenes, the final quality may degrade. To mitigate this, we perform pre-processing that preserves fidelity with minimal computation.

Table 7. Rendering quality metrics (PSNR, MS-SSIM, LPIPS) for Gaussian-based image representations across datasets.

Method	Metric	Kodak	DIV2K	Image-GS
FR	PSNR \uparrow	25.46	23.60	22.12
	MS-SSIM \uparrow	0.91	0.89	0.80
	LPIPS \downarrow	0.42	0.43	0.47
CSR	PSNR \uparrow	25.81	24.16	22.63
	MS-SSIM \uparrow	0.92	0.91	0.83
	LPIPS \downarrow	0.40	0.40	0.43
EGS	PSNR \uparrow	27.17	25.62	23.95
	MS-SSIM \uparrow	0.93	0.92	0.85
	LPIPS \downarrow	0.36	0.35	0.40
Ours	PSNR \uparrow	27.54	25.84	24.80
	MS-SSIM \uparrow	0.92	0.92	0.87
	LPIPS \downarrow	0.34	0.33	0.36

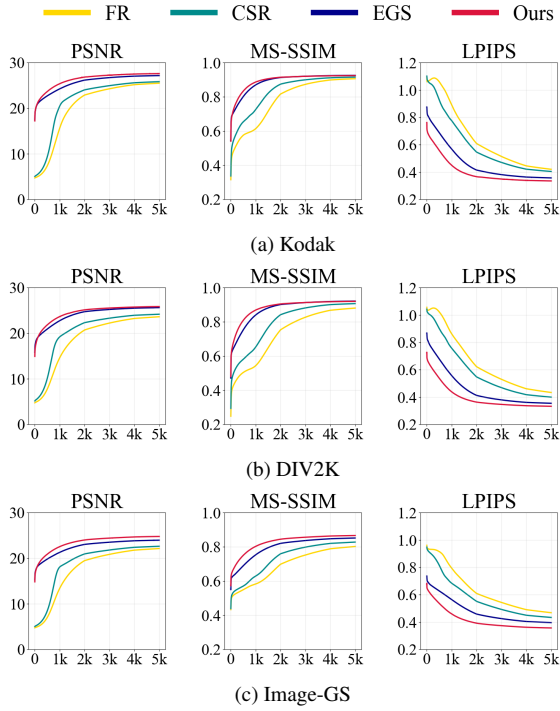


Figure 14. Change in rendering quality metrics (PSNR, MS-SSIM, LPIPS) over optimization steps across datasets.

Input image resolution All datasets in this paper were created at a fixed resolution of 480×640 (H \times W). Consequently, EllipssianNet was optimized for inference at this resolution. In order to support input images of varying sizes, each image is first resized to 480×640 before being processed by EllipssianNet. The resulting 2D Gaussians are then mapped back to the original image size, with both their positions and scales adjusted accordingly.

Image complexity control The maximum number of polygons in the training data was limited to 500, as de-

scribed in Section 3.2. If an image required more Gaussians, the reconstruction quality could degrade. To mitigate this issue, we applied a simple blurring step to overly complex images. Image complexity was measured using a spatial frequency scalar defined as:

$$S = \frac{\sum_{u,v} r(u,v)M(u,v)}{\sum_{u,v} M(u,v)} \quad (7)$$

Where $M(u,v)$ denotes the Fast Fourier Transform magnitude of the grayscale input image $I(x,y)$, and $r(u,v)$ is the frequency radius, given by:

$$r(u,v) = |\mathcal{F}\{I(x,y)\}| \quad (8)$$

$$r(u,v) = \sqrt{(u - W/2)^2 + (v - H/2)^2}$$

when the spatial frequency scalar exceeded a threshold, blurring was applied, up to a maximum of two iterations. We heuristically set the threshold to 50.

C. Evaluation details of Gaussian image

The rendering qualities are reported using three metrics, PSNR, MS-SSIM and LPIPS in Table 7, where our method consistently outperforms two random position based baselines. Compared to Ellipssian-guided sampling (EGS), which initializes isotropic Gaussians, our method consistently achieves superior overall quality. While EGS on the Kodak dataset demonstrated a slightly higher MS-SSIM score, this can be attributed to its relatively low resolution, whereas our approach maintains robust performance across all datasets. In addition, Figure 14 illustrates the change in rendering quality metrics over optimization steps. Across all three datasets, our method consistently delivers higher rendering quality from the early stages and achieves faster convergence compared to the baselines, as reflected in all reported metrics.

D. Implementation details of 3DGS SLAM

Figure 15 shows the detailed architecture of our 3DGS SLAM system. We use DROID-SLAM [39] for tracking, as was done in IG-SLAM [34] and DROID-Splat [12], whereas Gaussian optimization is done by the mapping thread. Our system runs an additional thread for sampling, and our core contribution lies in its Gaussian sampling strategy.

D.1. Tracking details

Our tracking thread manages the graph of keyframes. From the incoming RGB image stream, a new keyframe is selected if its average optical flow relative to the previous

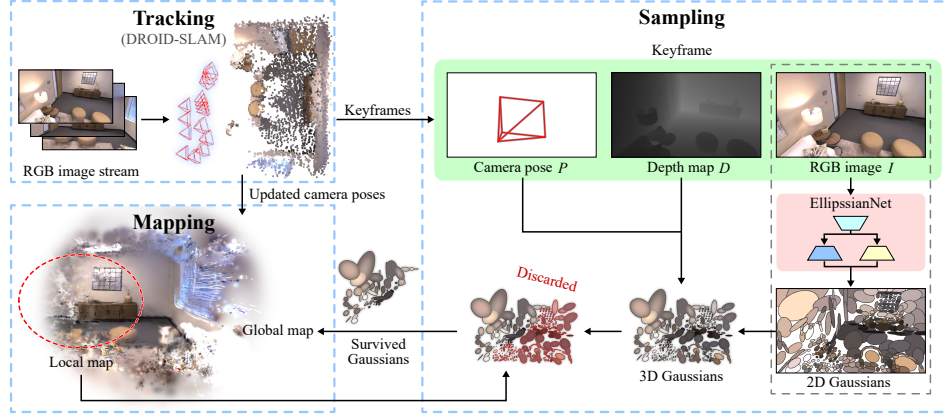


Figure 15. Architecture of the proposed 3DGS-based SLAM system, highlighting the details of three main threads.

keyframe exceeds a threshold. For each keyframe, the camera pose and depth map are estimated. As shown in Figure 15, the keyframe’s data passed to the sampling thread includes the camera pose, P , the depth map, D , and the RGB image, I . The tracking thread also performs Bundle Adjustment (BA) using the keyframe graph, and the camera poses updated via BA are passed to the mapping thread. For BA-based tracking, a covisibility frame graph $(\mathcal{V}, \mathcal{E})$ is dynamically constructed on every frame, where the nodes include the current frame and the latest keyframes. A new keyframe $\{P, D, I\} \in \mathcal{V}$ is added when its average optical flow relative to the previous keyframe exceeds a predefined threshold, indicating a different viewpoint. An edge $(i, j) \in \mathcal{E}$ is then established between frames if they are temporally adjacent or exhibit a low average optical flow, indicating substantial overlap. Here, the optical flow between two frames is computed by back-projecting pixels $\mathbf{p}_i \in \mathbb{R}^{H \times W \times 2}$ from one frame and then reprojecting them into the other with their relative camera pose $P_{ij} = P_j \circ P_i^{-1}$:

$$\mathbf{p}_{ij} = \Pi(P_{ij} \circ \Pi^{-1}(\mathbf{p}_i, D_i)) \quad (9)$$

where $\Pi(\cdot)$ denotes the camera projection function that maps 3D points in the camera-space coordinates to their corresponding pixel coordinates in the image, and $\Pi^{-1}(\cdot)$ is its inverse.

Instead of relying on handcrafted features, DROID-SLAM uses a deep learning approach to extract dense and robust visual features directly from the input image I . Initially, \mathbf{p}_{ij} is estimated using a linear motion model, which provides a rough prediction of the pixel correspondence between images I_i and I_j . Then, visual features are extracted from the input images and used to compute pixel-wise correlations around the initial estimate. From these correlations, a convolutional GRU network then infers a revision term \mathbf{r}_{ij} along with an associated confidence weight \mathbf{w}_{ij} to determine how much the initial correspondence should be updated. This revised correspondence $\mathbf{p}_{ij}^* = \mathbf{p}_{ij} + \mathbf{r}_{ij}$ is subsequently used to minimize the reprojection error E , thereby computing the optimal camera pose P^* and depth

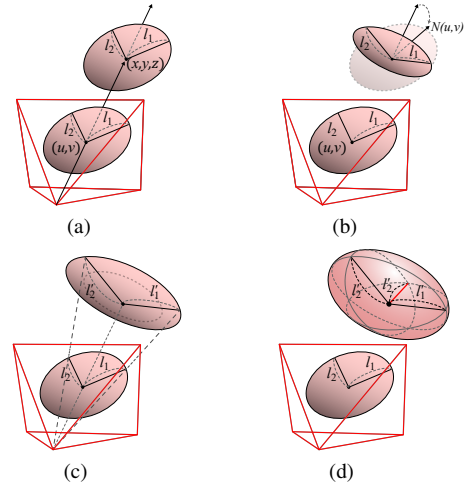


Figure 16. Lifting into the 3D world space: (a) The ellipse center is back-projected into the world space. (b) The ellipse is rotated using the normal map derived from the depth map. (c) The major and minor axes are scaled to ensure correct projection onto the image plane. (d) The third axis is added to complete a 3D Gaussian.

$$E = \sum_{(i,j) \in \mathcal{E}} \|\mathbf{p}_{ij}^* - \Pi(P_{ij}^* \circ \Pi^{-1}(\mathbf{p}_i, D_i^*))\|_{\Sigma_{ij}}^2 \quad (10)$$

Here, $\|\cdot\|_{\Sigma_{ij}}^2$ denotes the Mahalanobis distance weighted by Σ_{ij} , a diagonal matrix derived from the confidence weights \mathbf{w}_{ij} . This weighting ensures that correspondences with lower confidence have a reduced influence on the overall error.

By adopting this deep learning-based tracking approach, our system achieves more accurate and stable performance compared to conventional handcrafted techniques, even under challenging conditions such as varying illumination, texture, and dynamic scenes.

D.2. Sampling details

In the sampling thread, EllipssianNet estimates 2D Gaussians from a keyframe’s RGB image. It produces well-

Table 8. Mapping performance analysis.

Dataset	Method	Map/It.↓	Map/Fr.↓
Replica	Mono	20.50ms	3.07s
	Photo	15.18ms	-
	Splat	-	-
	Ours	10.11ms	1.32s
TUM	Mono	12.67ms	1.90s
	Photo	8.91ms	-
	Splat	-	-
	Ours	11.74ms	1.66s
ScanNet	Ours	5.50ms	0.82s

Table 9. Overall time overhead (f = input frame; kf = keyframe).

Thread	Elapsed time (ms)		
	Replica	TUM	ScanNet
Tracking (ms/f)	60.03	50.55	60.33
Sampling (ms/kf)	160.75	176.67	228.67
Mapping (s/kf)	1.32	1.66	0.82

distributed 2D Gaussians which match the RGB frequency of the input image. These Gaussians are then lifted to 3D space and further processed before being passed to the mapping thread.

In Figure 7f, the geometric parameters of each ellipse include its center position (u, v) , orientation, major axis l_1 , and minor axis l_2 . Lifting it into 3D space begins by back-projecting the center, (u, v) , into the world space using the depth map, D , and the camera pose, P , provided by the tracking thread. Let d denote $D(u, v)$ and \mathbf{K} denote the camera intrinsic matrix. Then, $d\mathbf{K}^{-1}(u, v, 1)^\top$ represents the camera-space point, and it is transformed into the world space using P . In Figure 16a, (x, y, z) represents the world space coordinates of the center.

From the depth map, D , we derive the normal map, N . The world space ellipse is rotated to align with the surface normal, $N(u, v)$. See Figure 16b. Then, as depicted in Figure 16c, the major and minor axes, l_1 and l_2 , are scaled to l'_1 and l'_2 , respectively, so that the footprint of the world space ellipse’s projection can be as close as possible to the original 2D ellipse. Note that the 3D Gaussian requires a third axis along the normal direction. In the current implementation, it is set to l'_2 , as shown in Figure 16d. Then, the lifting process is completed to generate a 3D Gaussian in the world space.

Before adding the 3D Gaussians into the *global map*, redundancy is tested. For this, the sampling thread asks the mapping thread for a *local map*. Being a subset of the global map, it stores the 3D Gaussians, which are located in the current keyframe’s field of view. In order to test if the newly-lifted Gaussians overlap with the Gaussians in the local map, we utilize Faiss [6, 14], a library for effi-

Table 10. Sampling performance analysis.

Processing stage	Elapsed time (ms)		
	Replica	TUM	ScanNet
EllipsoidNet	19.93	25.27	21.93
2D G. generation	33.92	39.89	47.93
3D G. generation	6.13	7.02	5.97
Local map handling	57.09	54.13	80.42
Overall sampling	160.75	176.67	228.67

cient similarity search and dense vector clustering, as done in Point-SLAM [32] and SAR-SLAM [24].

Overlapping Gaussians are discarded. If the number of survived Gaussians exceeds the threshold (80 in the current implementation), they are added to the global map. Otherwise, the entire set is removed because the tiny holes, which would be filled by them, can also be filled during the optimization process.

D.3. Mapping details

The mapping thread is in charge of integrating the 3D Gaussians passed by the sampling thread into the global map, and it follows the standard way of 3DGS-based mapping, which is an iterative optimization combined with densification. When idle, i.e., when no request for a local map is received from the sampling thread, the global map is consistently refined using keyframes.

When the tracking thread performs Bundle Adjustment, it optimizes the camera poses of the involved keyframes. In order to maintain global consistency, the updated camera poses are passed to the mapping thread, which uses them to transform the Gaussians extracted from the keyframes. In addition, the Gaussians densified using the transformed Gaussians are also transformed accordingly.

E. Evaluation details of 3DGS SLAM

E.1. System performance analysis

Table 8 compares mapping performance across SLAM methods. The baseline results are taken from the article of MGS-SLAM [49], whose code is unavailable and did not use the ScanNet dataset. Since their evaluation was conducted on an NVIDIA GeForce RTX 3090, we used the same GPU. Our method, which accounts for the full process of sampling and optimization, achieves the fastest mapping time per frame (1.32s on Replica and 1.66s on TUM).

Table 9 shows the time taken by all three threads of our SLAM system. Sampling and mapping run on keyframes only, whereas tracking runs on every input frame. Our framework lets the threads execute in parallel, and the mapping thread runs at a much slower rate, which is standard in modern SLAM methods. Table 9 demonstrates that our method is well-suited for real-time applications. The sampling thread’s performance is further analyzed in Table 10.

Table 11. Rendering performance on Replica over frames.
The best results are highlighted as First, Second and Third.

Method	Metric	R0	R1	R2	O0	O1	O2	O3	O4	Avg.
Mono	PSNR \uparrow	25.85	25.36	22.01	32.76	32.90	23.37	26.63	24.03	26.61
	SSIM \uparrow	0.813	0.797	0.831	0.912	0.908	0.836	0.859	0.867	0.853
	LPIPS \downarrow	0.232	0.347	0.381	0.221	0.180	0.305	0.208	0.334	0.276
Photo	PSNR \uparrow	26.78	28.96	30.38	32.55	36.25	29.30	30.81	29.46	30.57
	SSIM \uparrow	0.771	0.874	0.912	0.924	0.941	0.911	0.910	0.913	0.918
	LPIPS \downarrow	0.170	0.118	0.093	0.128	0.080	0.114	0.102	0.108	0.093
Splat	PSNR \uparrow	24.77	27.66	27.10	32.73	31.90	26.70	26.48	27.58	28.12
	SSIM \uparrow	0.765	0.831	0.848	0.912	0.913	0.870	0.859	0.891	0.861
	LPIPS \downarrow	0.185	0.169	0.146	0.115	0.131	0.154	0.118	0.138	0.145
Ours	PSNR \uparrow	29.01	30.34	32.89	34.53	37.81	29.80	31.32	31.28	32.12
	SSIM \uparrow	0.850	0.896	0.942	0.928	0.958	0.915	0.929	0.934	0.919
	LPIPS \downarrow	0.172	0.135	0.117	0.156	0.111	0.155	0.113	0.126	0.136

Table 12. Rendering performance on Replica over keyframes.

Method	Metric	R0	R1	R2	O0	O1	O2	O3	O4	Avg.
OTF-NVS	PSNR \uparrow	30.33	24.25	31.88	31.47	36.91	31.00	33.46	23.61	30.36
	SSIM \uparrow	0.920	0.861	0.946	0.933	0.959	0.943	0.946	0.903	0.926
	LPIPS \downarrow	0.145	0.249	0.145	0.183	0.151	0.156	0.135	0.229	0.174
Ours	PSNR \uparrow	29.82	30.91	33.55	34.97	38.72	30.55	31.87	31.44	32.73
	SSIM \uparrow	0.870	0.901	0.945	0.935	0.962	0.920	0.933	0.933	0.925
	LPIPS \downarrow	0.164	0.140	0.118	0.159	0.102	0.154	0.109	0.136	0.135

Table 13. Rendering performance on TUM.

The best results are highlighted as First, Second and Third.

Method	Metric	fr1/desk	fr2/xyz	fr3/office	Avg.
Mono	PSNR \uparrow	17.46	15.54	19.76	17.58
	SSIM \uparrow	0.657	0.651	0.742	0.683
	LPIPS \downarrow	0.392	0.352	0.333	0.359
Photo	PSNR \uparrow	19.75	20.44	16.09	18.76
	SSIM \uparrow	0.701	0.696	0.588	0.661
	LPIPS \downarrow	0.263	0.189	0.413	0.289
Splat	PSNR \uparrow	19.60	24.45	18.51	20.85
	SSIM \uparrow	0.742	0.834	0.687	0.754
	LPIPS \downarrow	0.285	0.158	0.394	0.279
Ours	PSNR \uparrow	19.97	23.79	21.07	21.61
	SSIM \uparrow	0.716	0.815	0.751	0.761
	LPIPS \downarrow	0.333	0.190	0.321	0.281

Table 14. Rendering performance on TUM.

Method	Metric	fr1/desk	fr2/xyz	fr3/office	Avg.
OTF	PSNR \uparrow	20.28	26.21	21.52	22.67
	SSIM \uparrow	0.761	0.861	0.810	0.811
	LPIPS \downarrow	0.342	0.197	0.293	0.277
Ours	PSNR \uparrow	21.24	24.83	22.04	22.70
	SSIM \uparrow	0.755	0.836	0.775	0.789
	LPIPS \downarrow	0.300	0.166	0.310	0.259

E.2. Novel View Rendering

Table 11 and Table 12 report the mapping quality on the Replica dataset, evaluated on non-keyframes and keyframes, respectively. Table 13 and Table 14 report the results on the TUM dataset under the same conditions. Table 15 and Table 16 provide the corresponding results for the ScanNet dataset. Overall, our method demonstrates robust reconstruction performance in most scenes.

E.3. Tracking accuracy

Our approach does not aim to improve tracking performance relative to DROID-SLAM, as our primary contribution is an adaptive Gaussian distribution that significantly improves mapping quality and efficiency for real-time SLAM applications. However, for completeness, we present the tracking results using the Absolute Trajectory Error (ATE) on the Replica [36] and TUM [37] datasets in Table 17 and 18.

F. Qualitative Evaluation

To qualitatively assess our novel view rendering quality, we compared our method to MonoGS [27], Photo-SLAM [13] and Splat-SLAM [33] on Replica and TUM and ScanNet datasets. We exclude OTF-NVS [28] from this qualitative comparison because its official code does not support rendering from non-keyframes, making it difficult to generate the appropriate frames.

Table 15. Rendering performance on ScanNet.
The best results are highlighted as First, Second and Third.

Method	Metric	01	03	04	05	08	13	15	16	Avg.
Mono	PSNR \uparrow	20.61	21.98	19.10	18.33	20.91	26.64	18.47	19.68	20.53
	SSIM \uparrow	0.770	0.818	0.704	0.733	0.785	0.812	0.744	0.705	0.750
	LPIPS \downarrow	0.530	0.382	0.553	0.502	0.520	0.552	0.443	0.484	0.512
Photo	PSNR \uparrow	16.49	19.05	15.67	18.98	19.71	24.68	10.67	14.33	17.22
	SSIM \uparrow	0.727	0.754	0.577	0.759	0.749	0.789	0.55	0.623	0.682
	LPIPS \downarrow	0.465	0.356	0.529	0.351	0.409	0.396	0.545	0.525	0.460
Splat	PSNR \uparrow	18.28	23.64	21.65	20.18	22.44	25.53	21.10	23.17	21.76
	SSIM \uparrow	0.678	0.824	0.707	0.741	0.751	0.709	0.742	0.753	0.726
	LPIPS \downarrow	0.556	0.285	0.381	0.331	0.436	0.420	0.395	0.303	0.403
Ours	PSNR \uparrow	26.19	22.19	21.31	23.57	22.23	30.21	17.21	23.17	23.26
	SSIM \uparrow	0.860	0.809	0.732	0.827	0.796	0.862	0.730	0.800	0.802
	LPIPS \downarrow	0.358	0.386	0.434	0.338	0.474	0.410	0.544	0.328	0.409

Table 16. Rendering performance on ScanNet.

Method	Metric	01	03	04	05	08	13	15	16	Avg.
OTF	PSNR \uparrow	22.16	23.45	24.59	21.68	26.09	28.80	23.69	22.74	24.15
	SSIM \uparrow	0.829	0.862	0.833	0.838	0.866	0.870	0.863	0.813	0.847
	LPIPS \downarrow	0.412	0.300	0.299	0.364	0.327	0.346	0.323	0.353	0.341
Ours	PSNR \uparrow	28.20	22.81	23.13	24.60	22.84	30.56	17.93	23.95	24.25
	SSIM \uparrow	0.885	0.820	0.786	0.848	0.817	0.870	0.748	0.822	0.825
	LPIPS \downarrow	0.333	0.373	0.384	0.320	0.464	0.412	0.512	0.305	0.388

Table 17. Tracking performance on Replica (ATE in cm).

Method	R0	R1	R2	O0	O1	O2	O3	O4	Avg.
Mono	9.75	67.90	38.30	45.63	24.52	45.27	11.52	62.57	38.18
Photo	0.39	1.08	0.22	3.82	5.21	2.38	0.44	3.92	2.18
Splat	1.77	1.04	0.97	1.33	2.51	1.98	2.56	2.11	1.78
OTF-NVS	5.72	30.62	5.95	7.50	17.93	7.81	1.51	51.38	16.05
Ours	1.22	0.83	0.67	1.02	0.38	1.22	0.69	1.12	0.89

Table 18. Tracking performance on TUM (ATE in cm).

Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Mono	3.38	4.57	3.18	3.71
Photo	1.59	0.95	2.28	1.61
Splat	2.14	0.24	1.99	1.46
OTF-NVS	9.91	1.13	93.33	34.79
Ours	4.34	0.58	5.82	3.58

From Figure 17 to 35, we provide two rendering results: (1) the standard version, and (2) a version in which the 3D Gaussians are uniformly scaled by a factor of 0.2 just for visualizing their distribution. Additionally, each image in-

cludes zoomed-in regions that clearly highlight key scene details and performance differences.

As evident from each result, our method reconstructs sharper edges and recovers finer details compared to the other methods. Moreover, the manipulated Gaussian-scale renderings illustrate that our Gaussians are more appropriately distributed throughout the scene, unlike other techniques where Gaussians are predominantly clustered along edges and corners. Our Gaussians also exhibit anisotropic shapes that capture local representation, whereas other methods are unable to achieve this. This further validates the effectiveness of our adaptive Gaussian placement strategy in achieving more accurate and efficient scene representations.

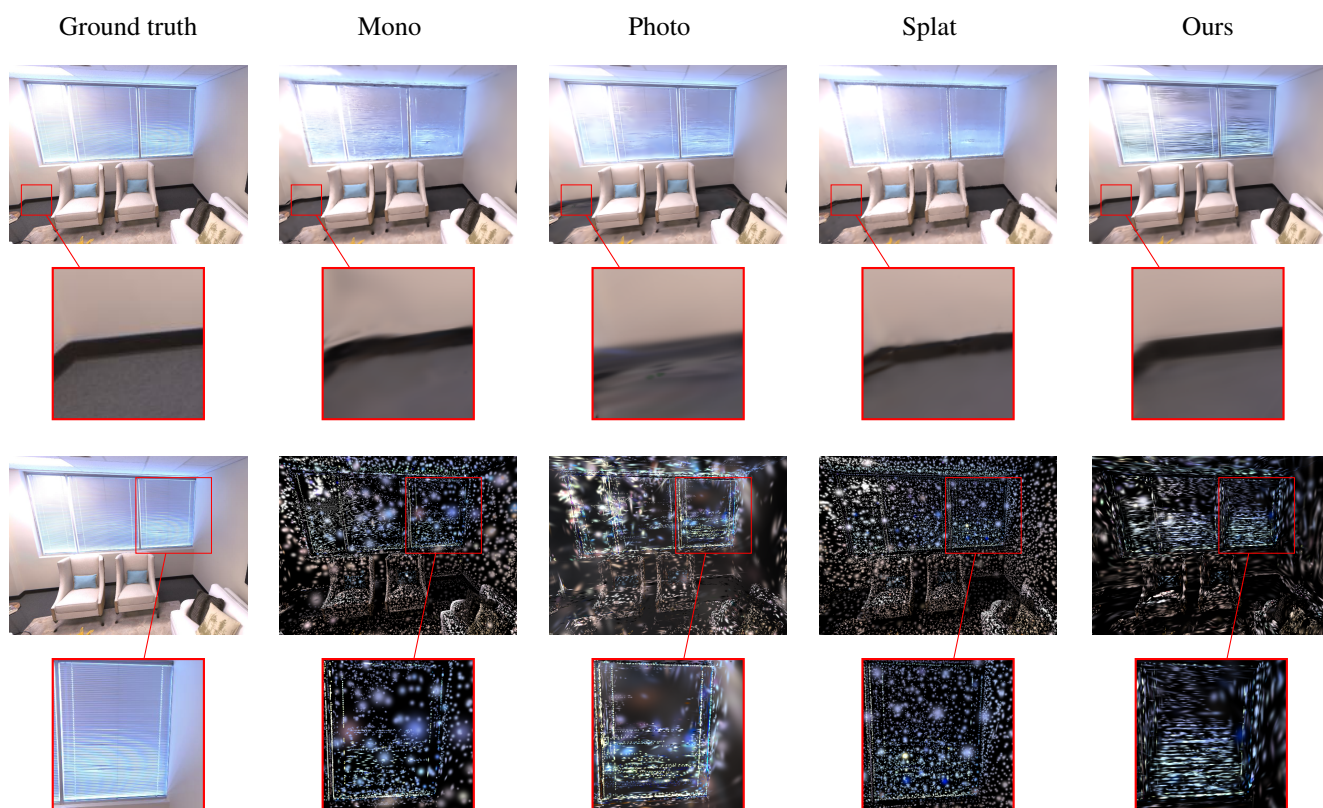


Figure 17. Qualitative evaluation on Replica R0.

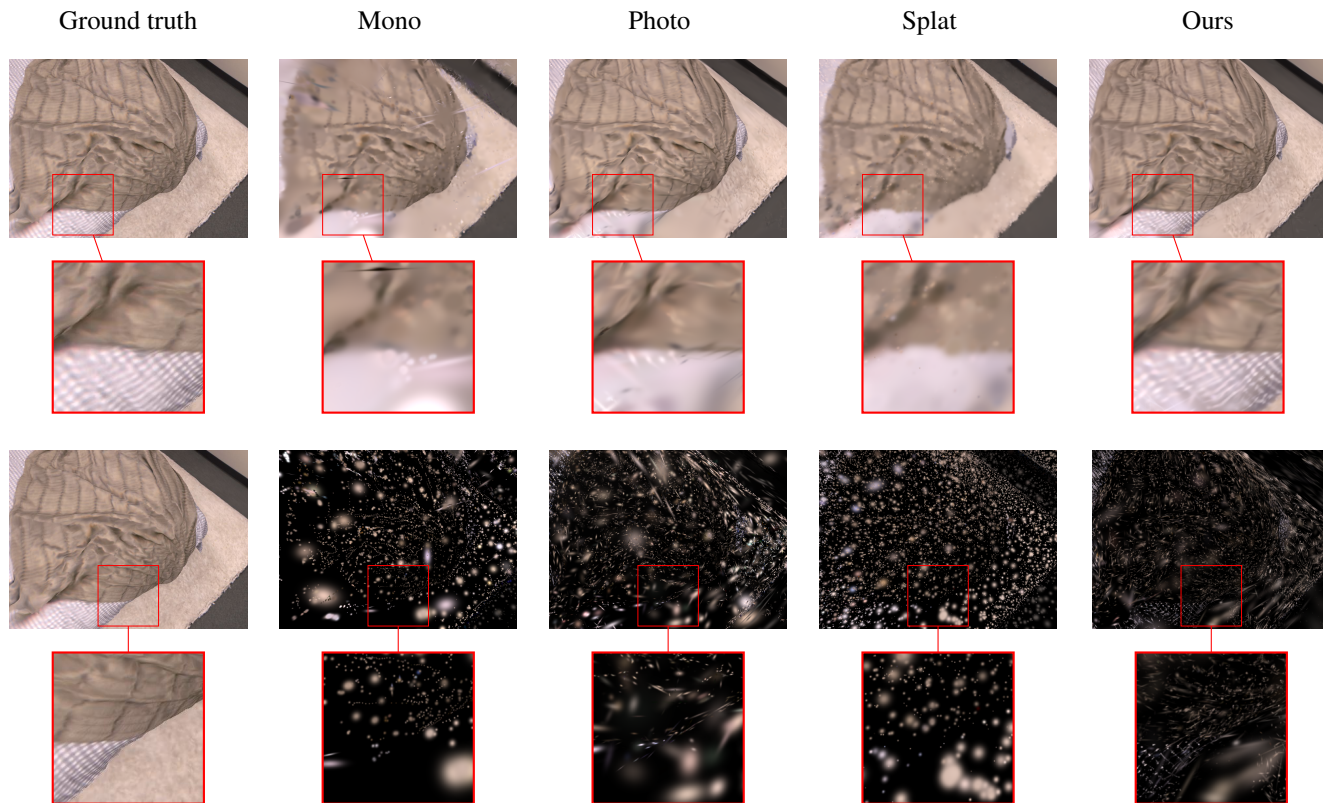


Figure 18. Qualitative evaluation on Replica R1.

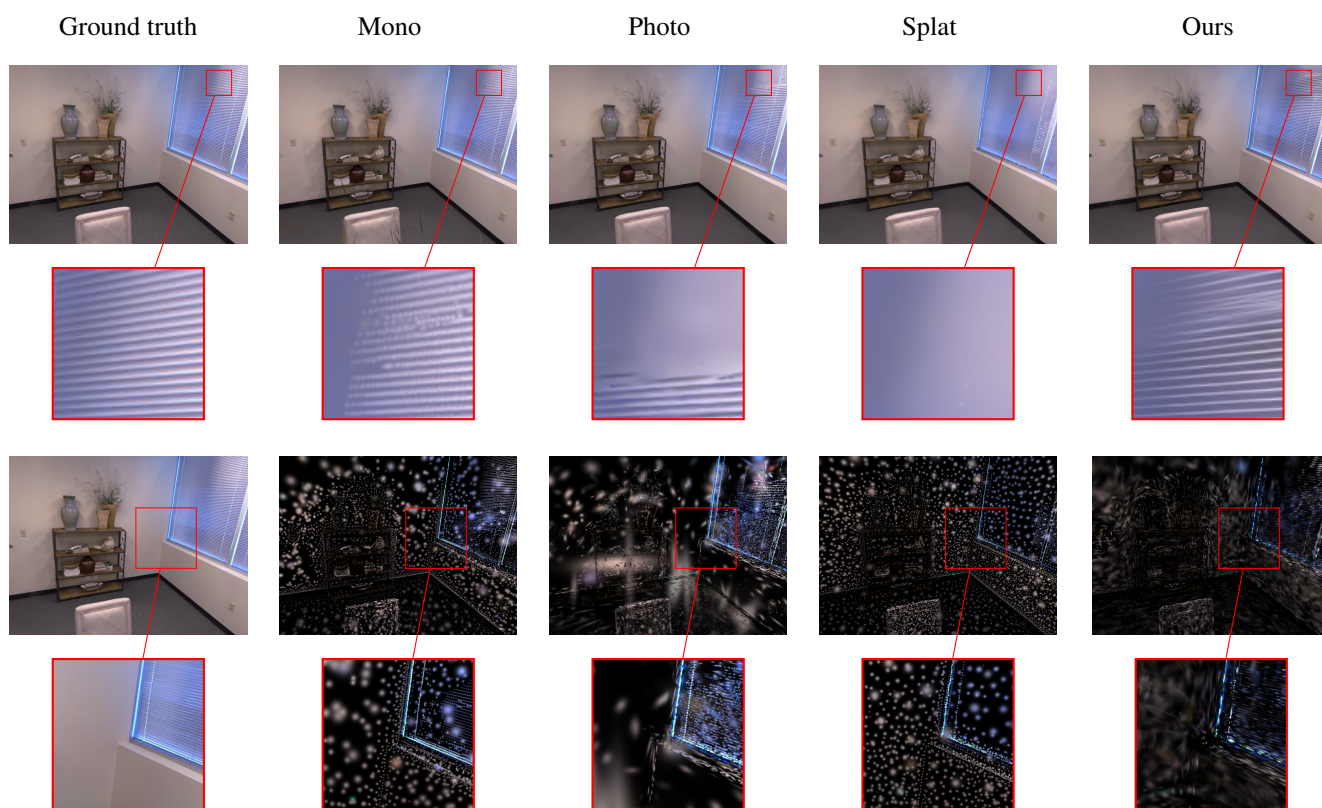


Figure 19. Qualitative evaluation on Replica R2.

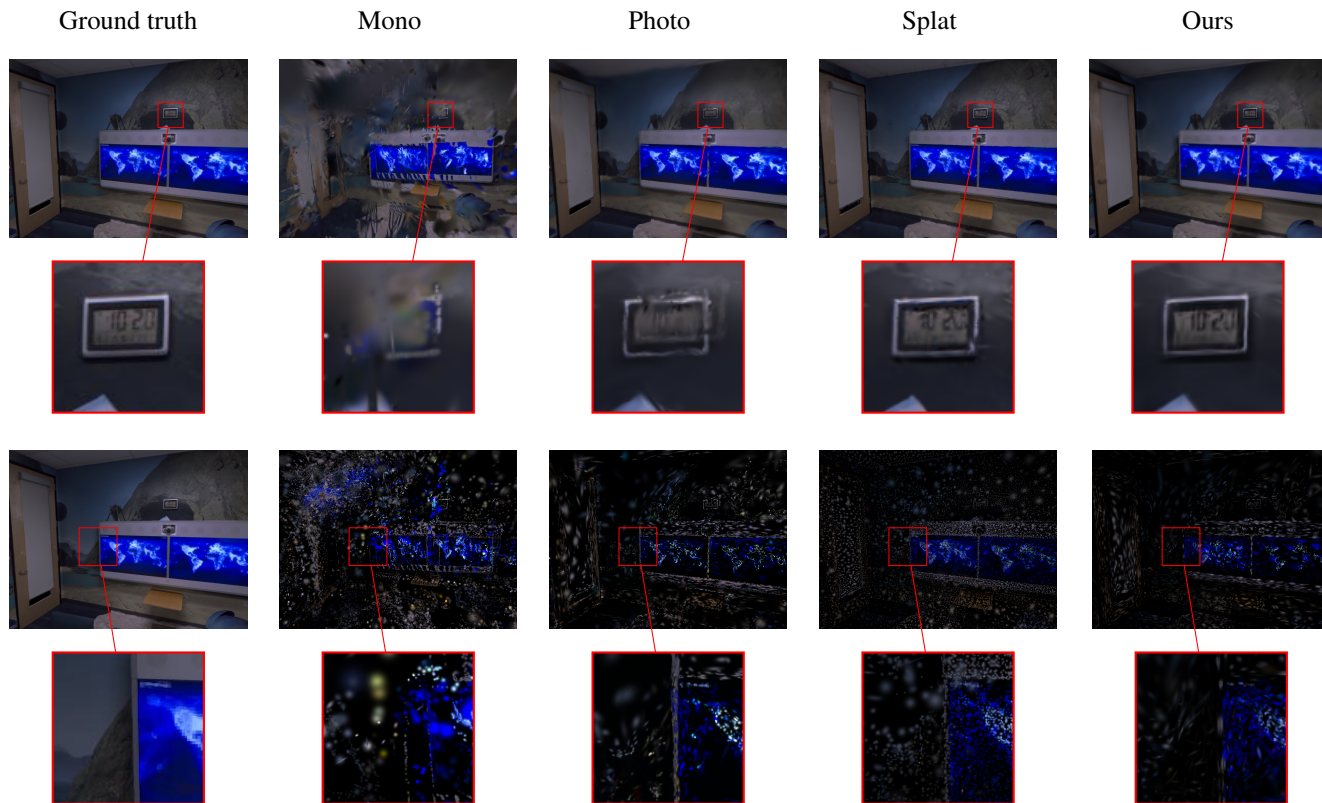


Figure 20. Qualitative evaluation on Replica O0.

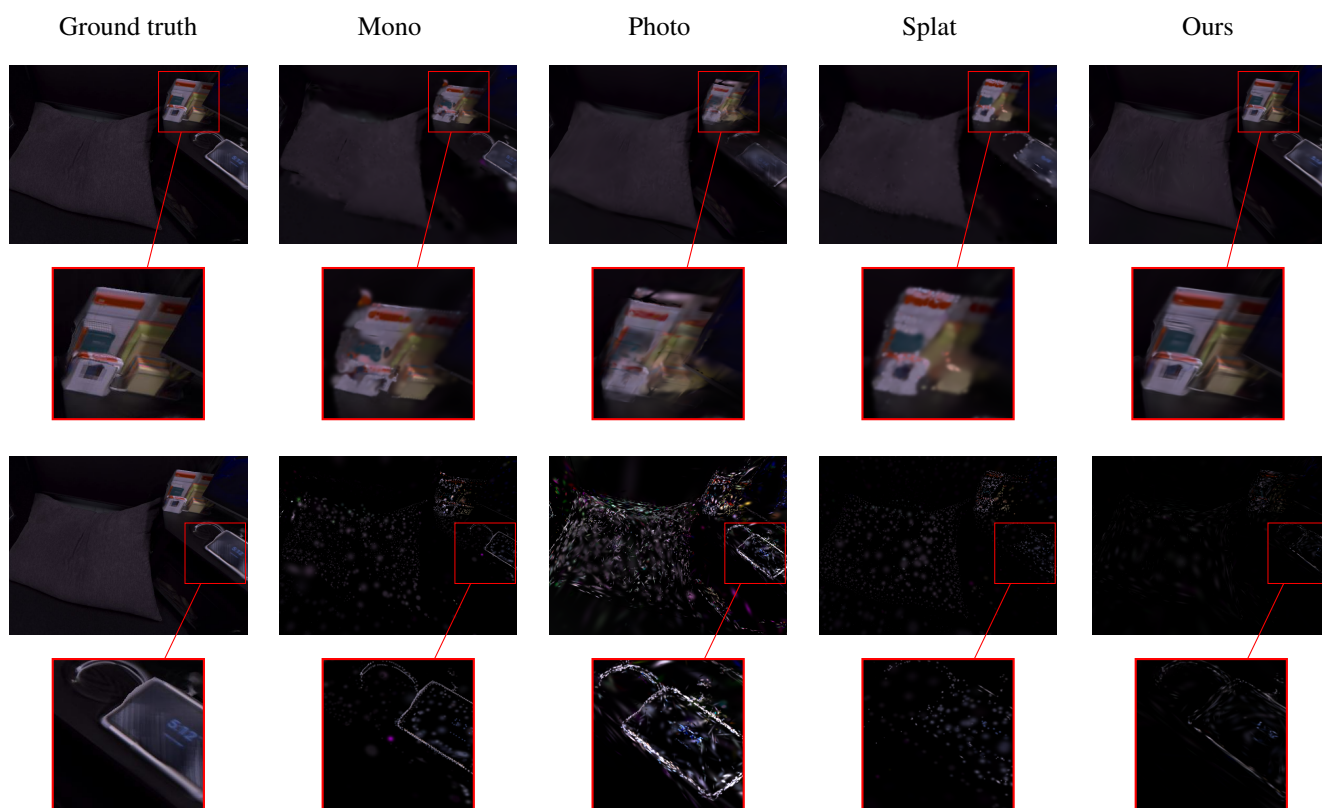


Figure 21. Qualitative evaluation on Replica O1.

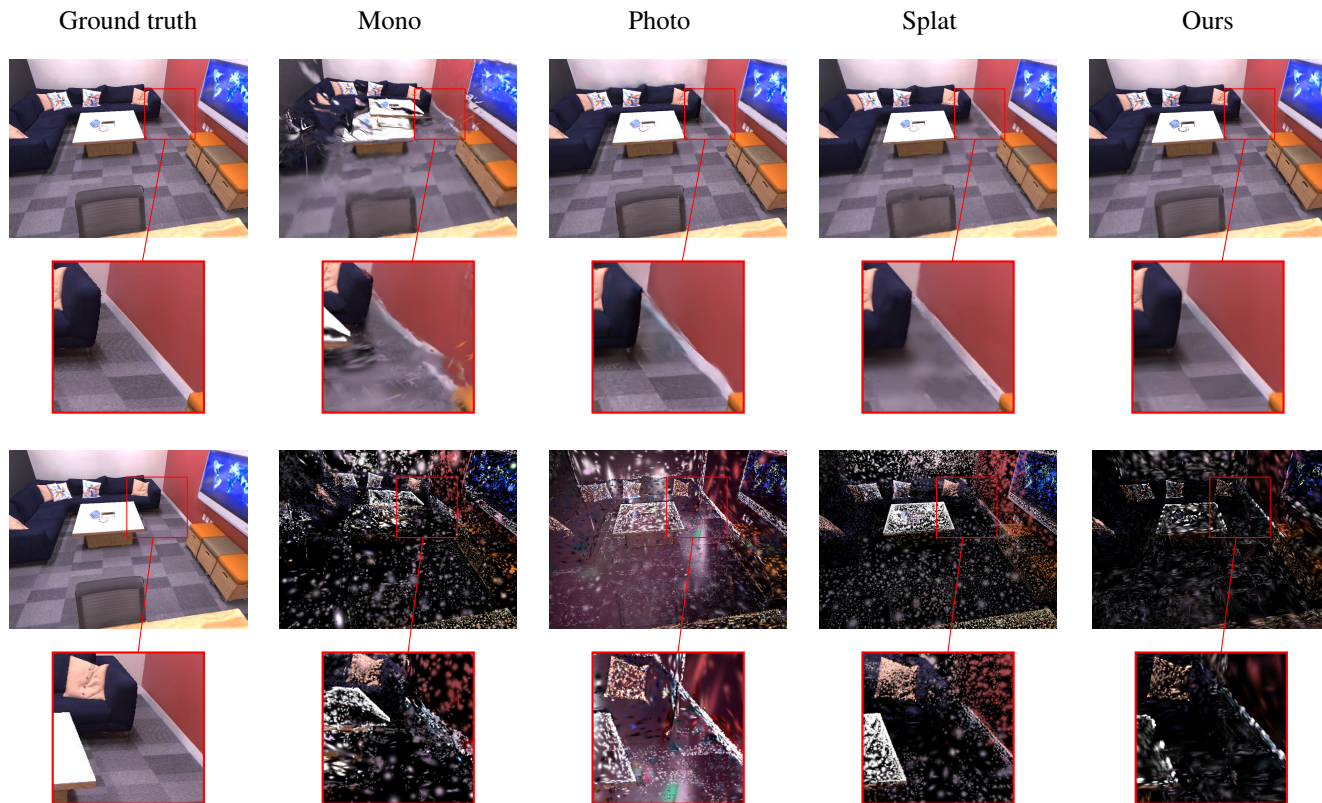


Figure 22. Qualitative evaluation on Replica O2.

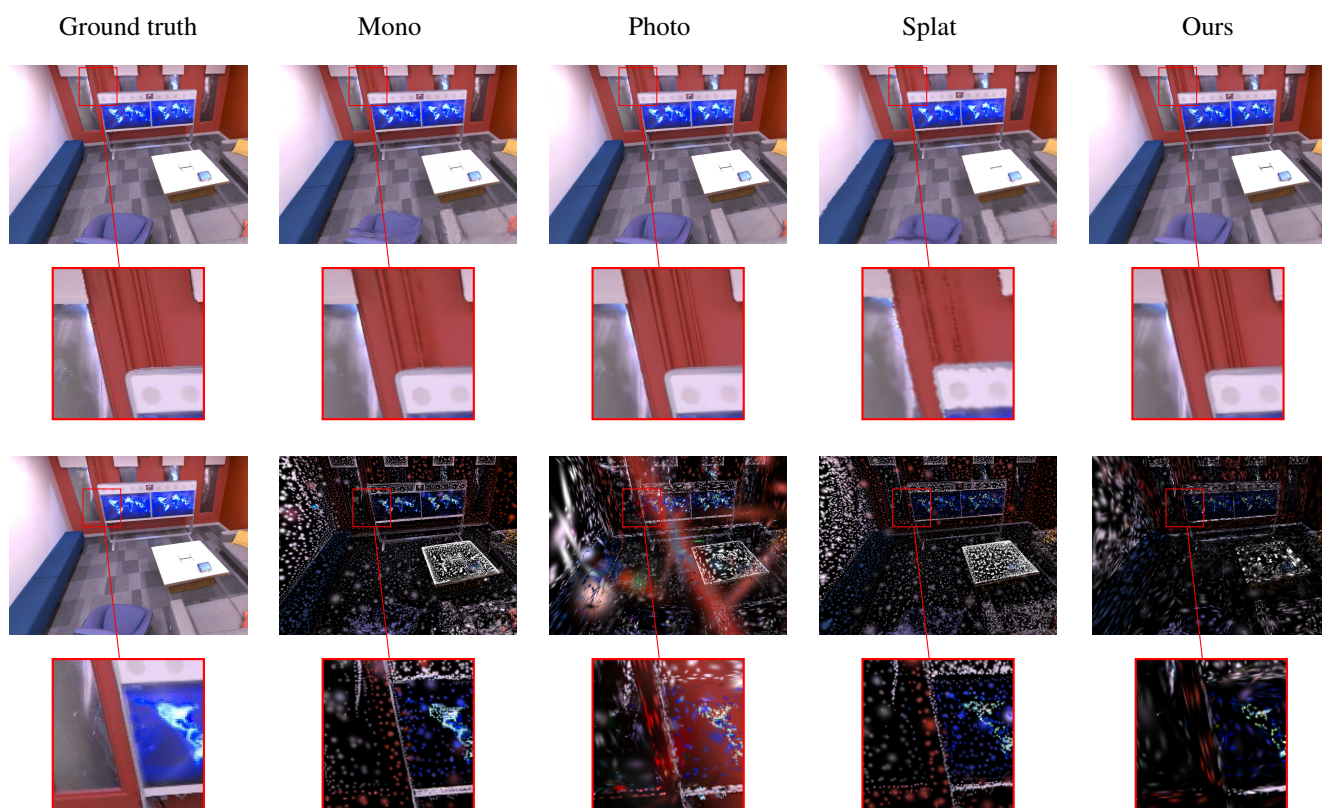


Figure 23. Qualitative evaluation on Replica O3.

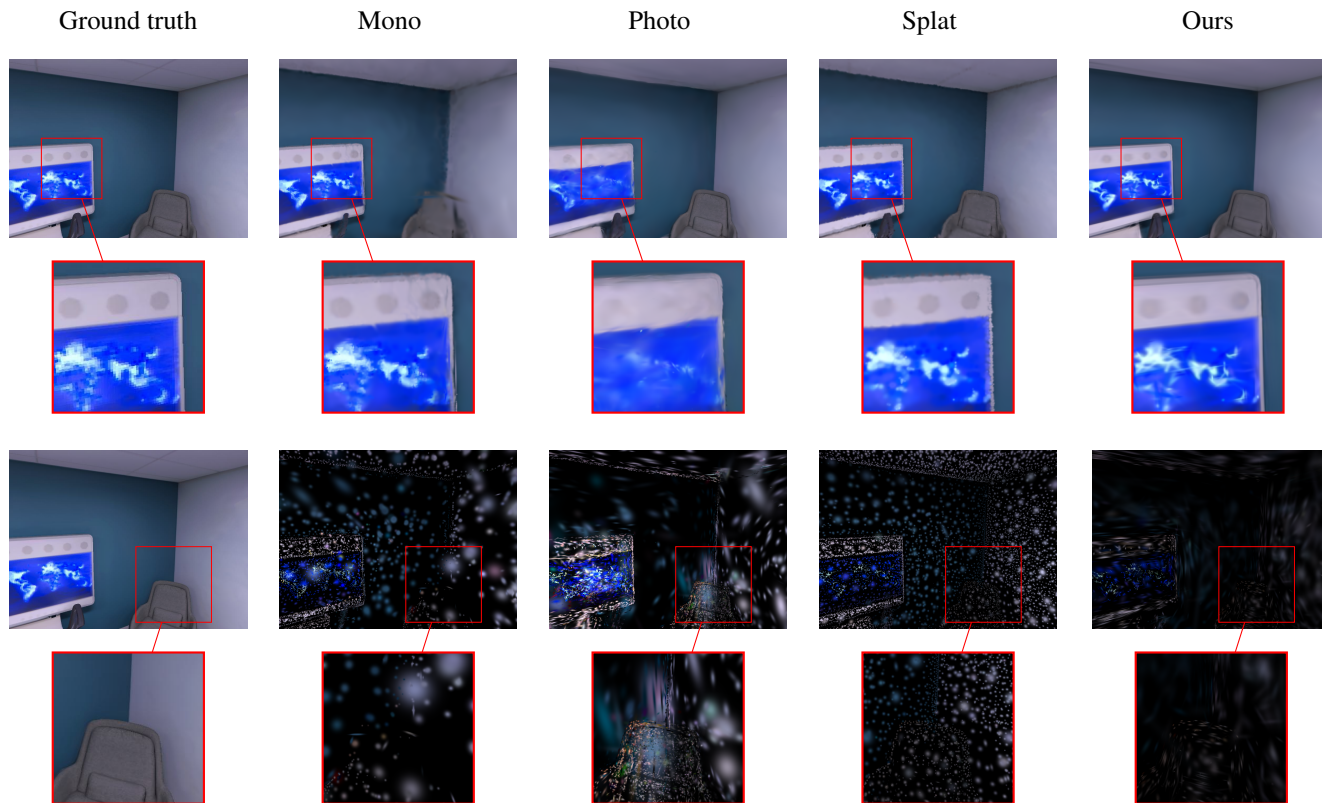


Figure 24. Qualitative evaluation on Replica O4.

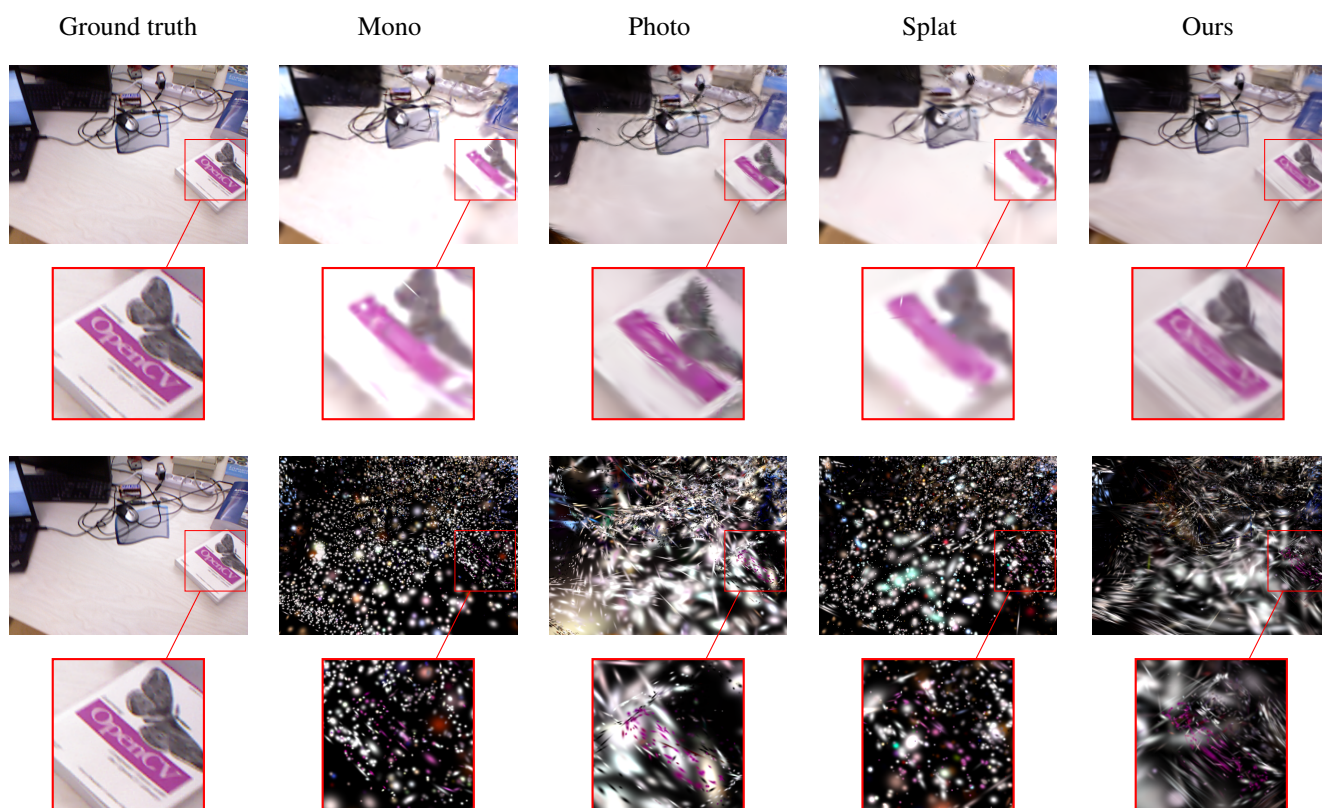


Figure 25. Qualitative evaluation with TUM fr1/desk.

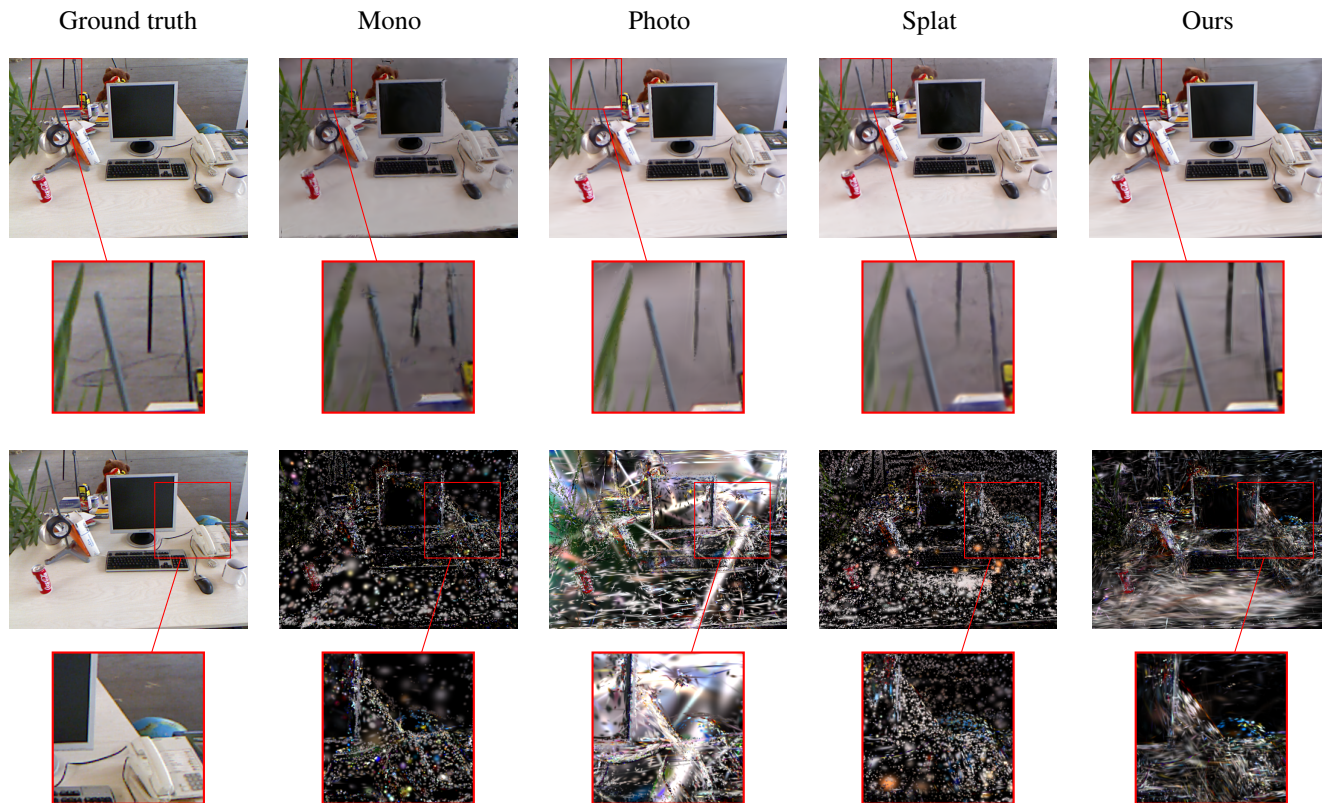


Figure 26. Qualitative evaluation with TUM fr2/xyz.

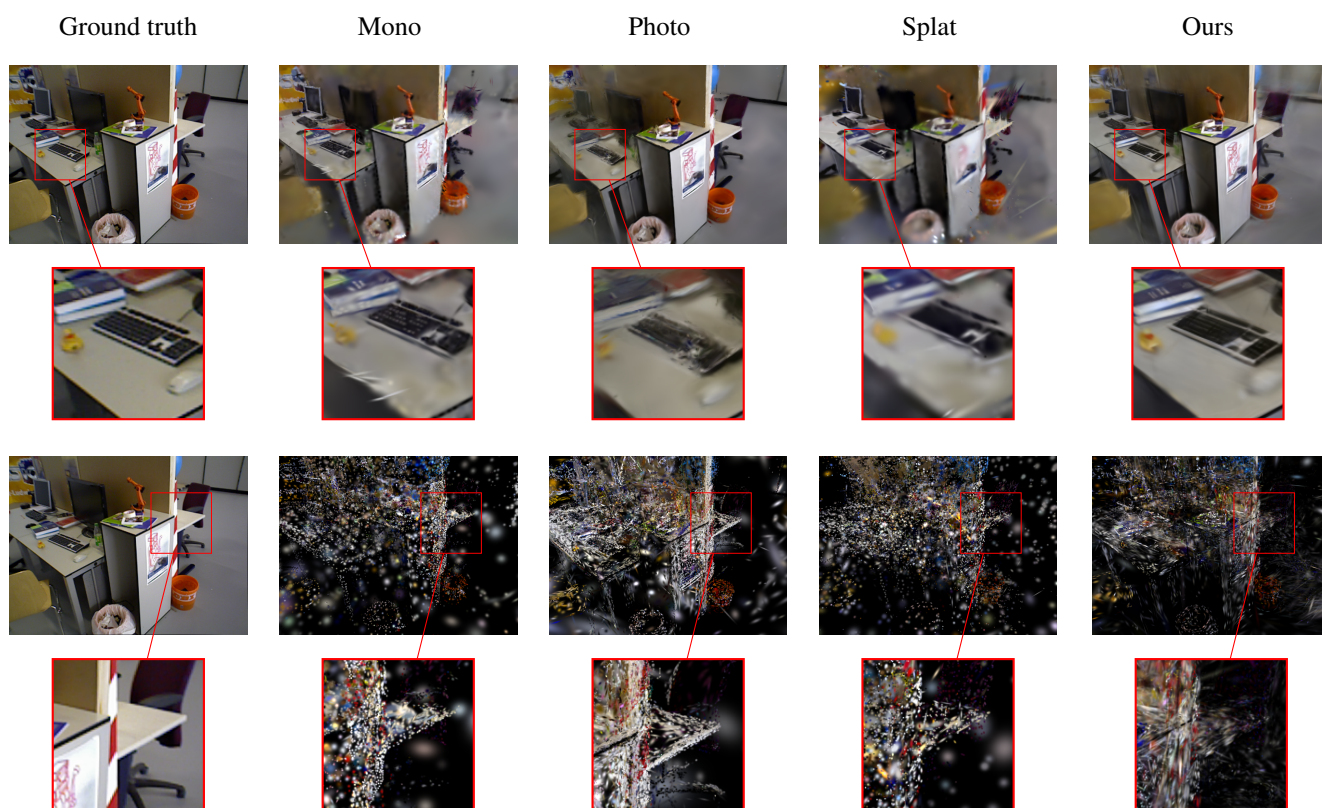


Figure 27. Qualitative evaluation with TUM fr3/office.

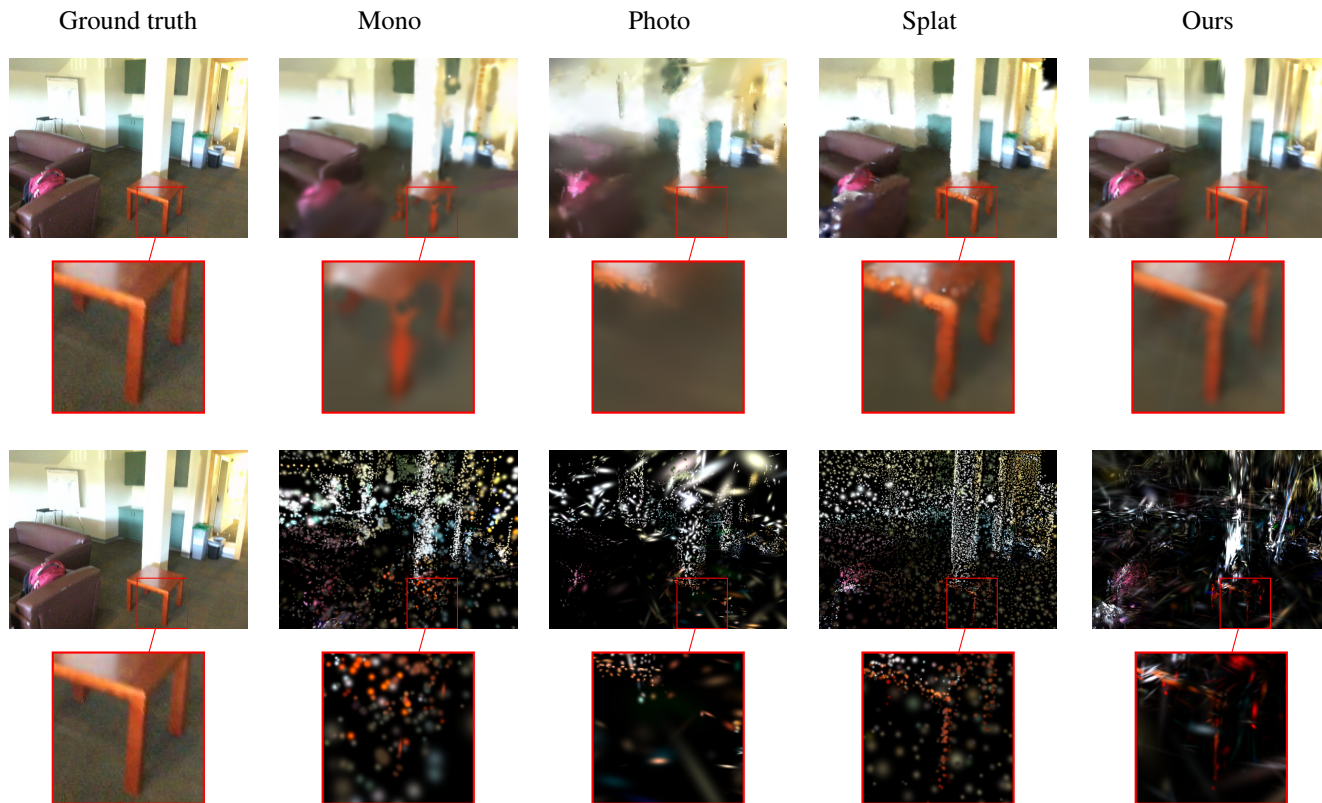


Figure 28. Qualitative evaluation on ScanNet 01.

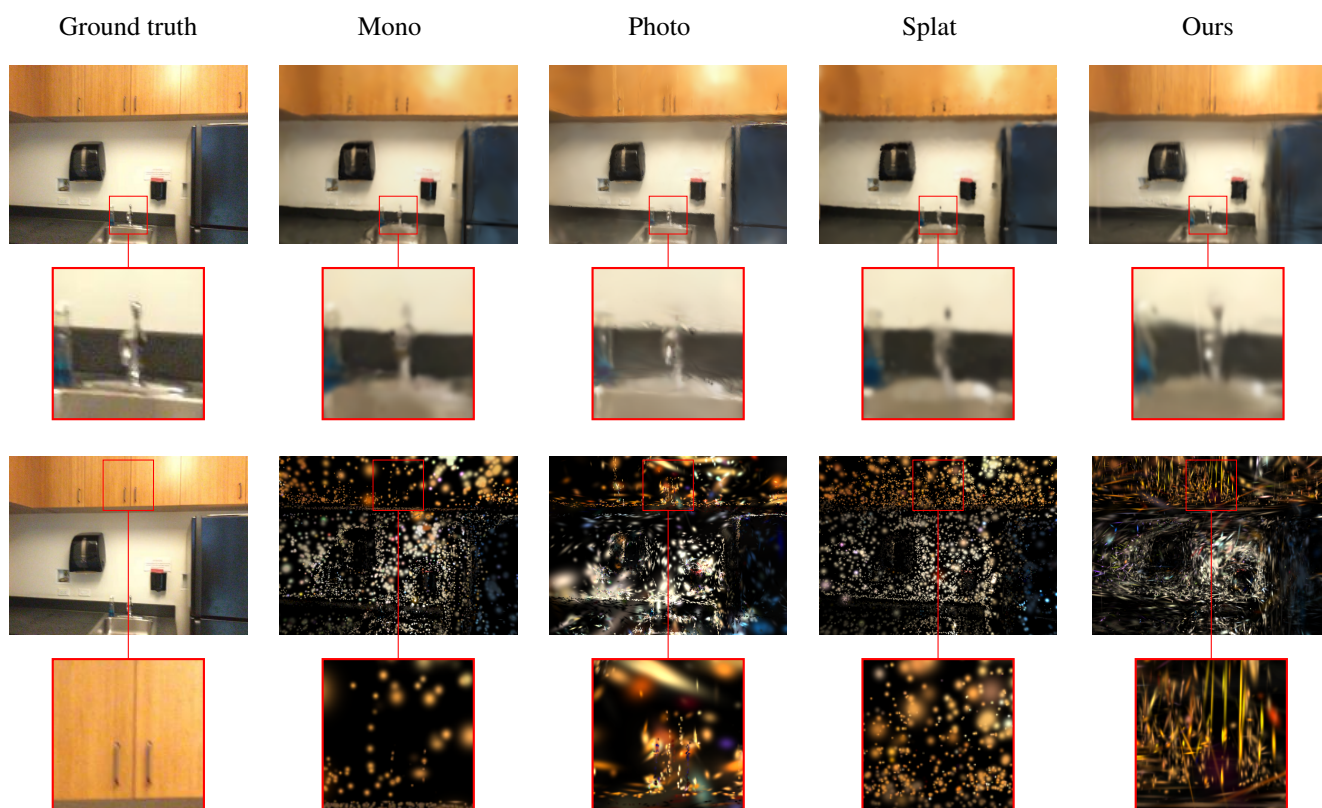


Figure 29. Qualitative evaluation on ScanNet 03.

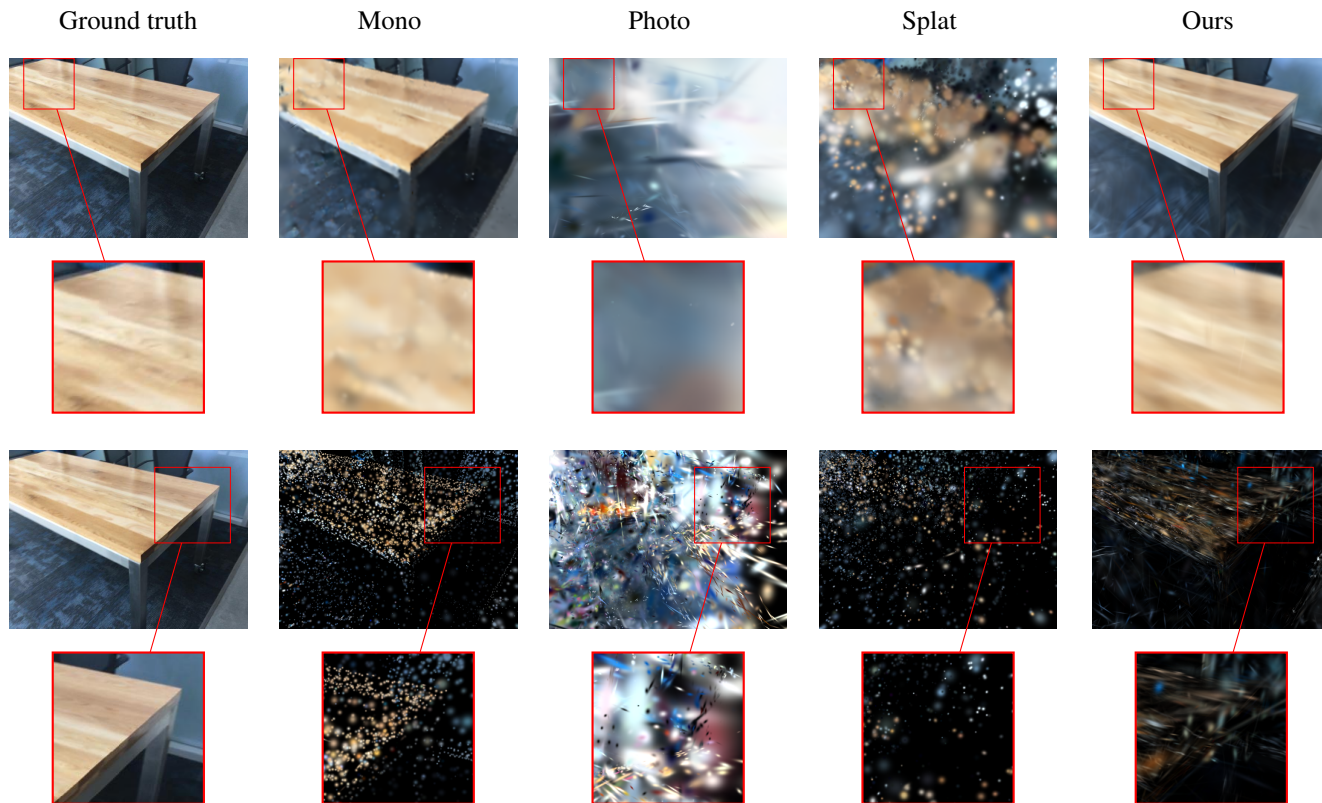


Figure 30. Qualitative evaluation on ScanNet 04 (Photo-SLAM failed tracking).

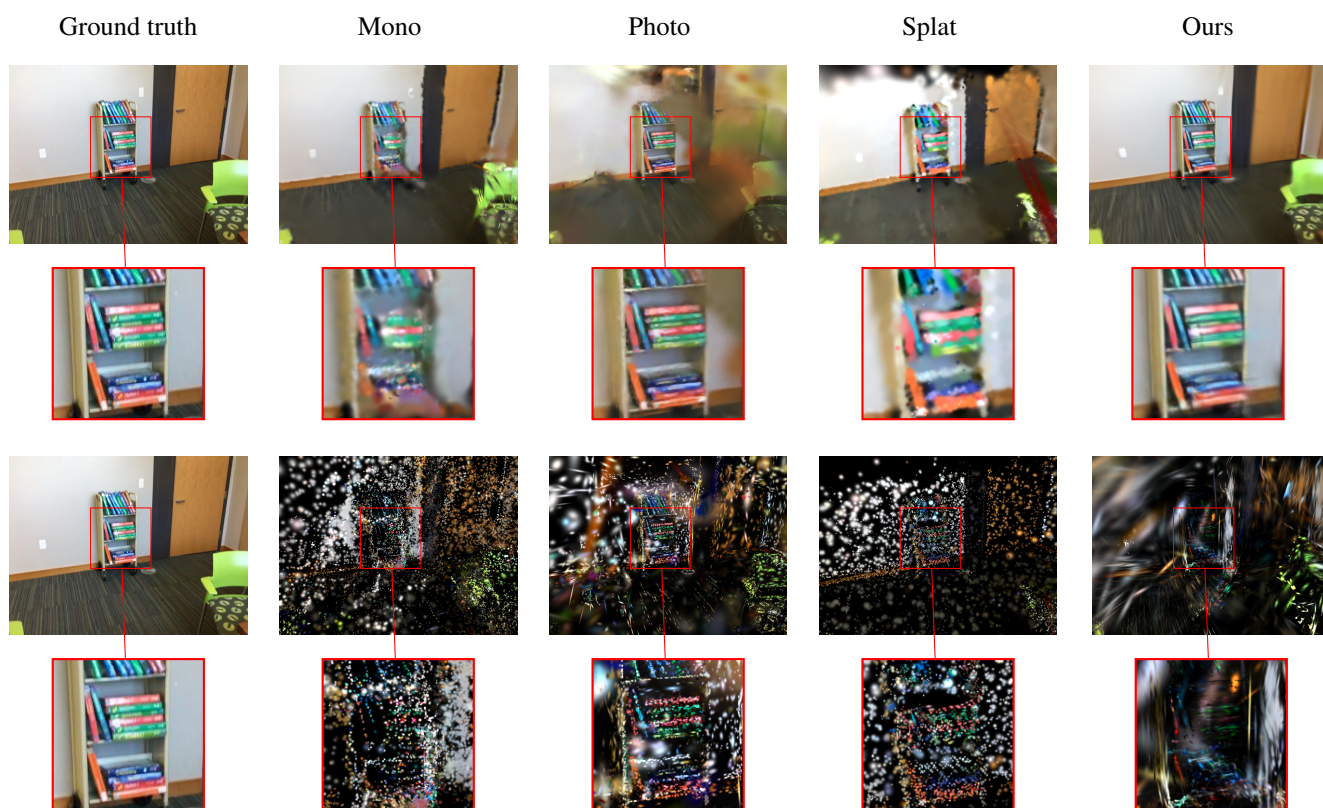


Figure 31. Qualitative evaluation on ScanNet 05.

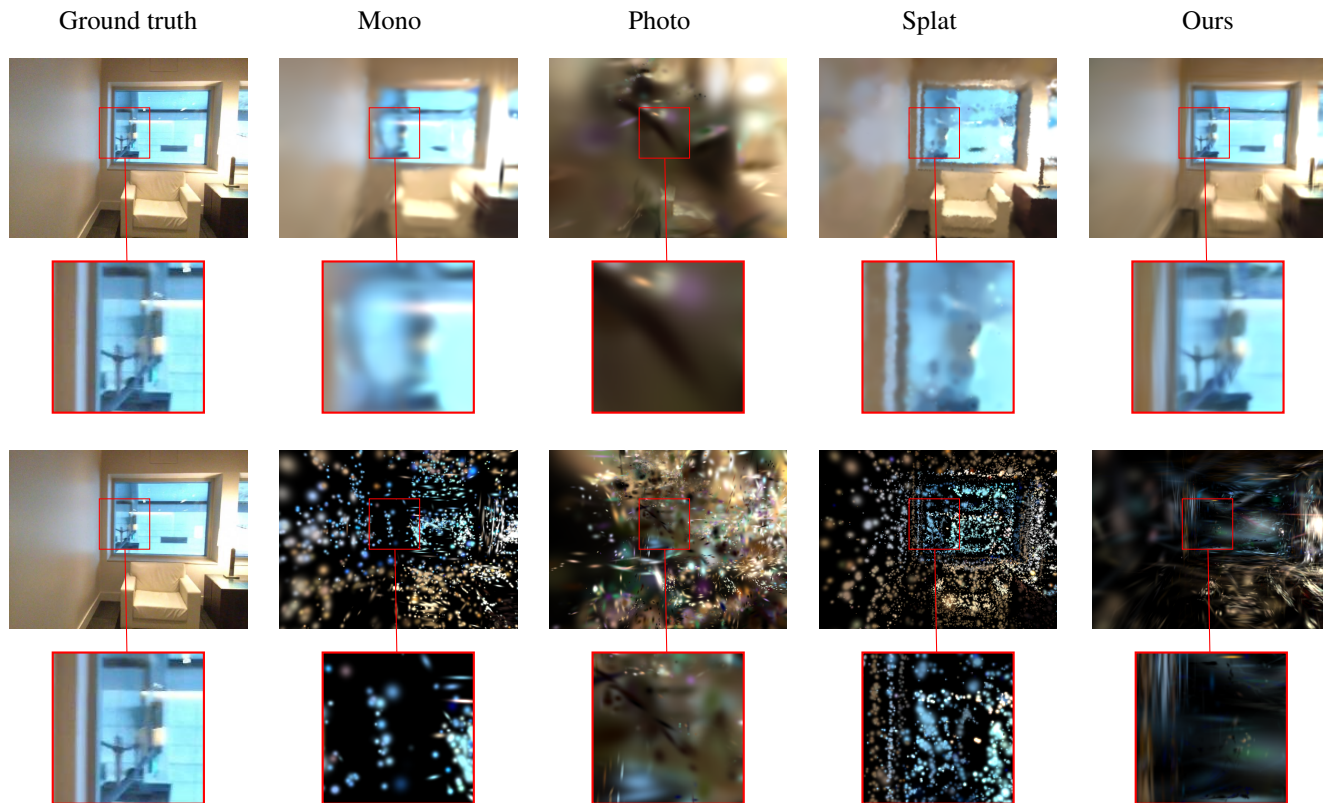


Figure 32. Qualitative evaluation on ScanNet 08 (Photo-SLAM failed tracking).



Figure 33. Qualitative evaluation on ScanNet 13 (Photo-SLAM failed tracking).

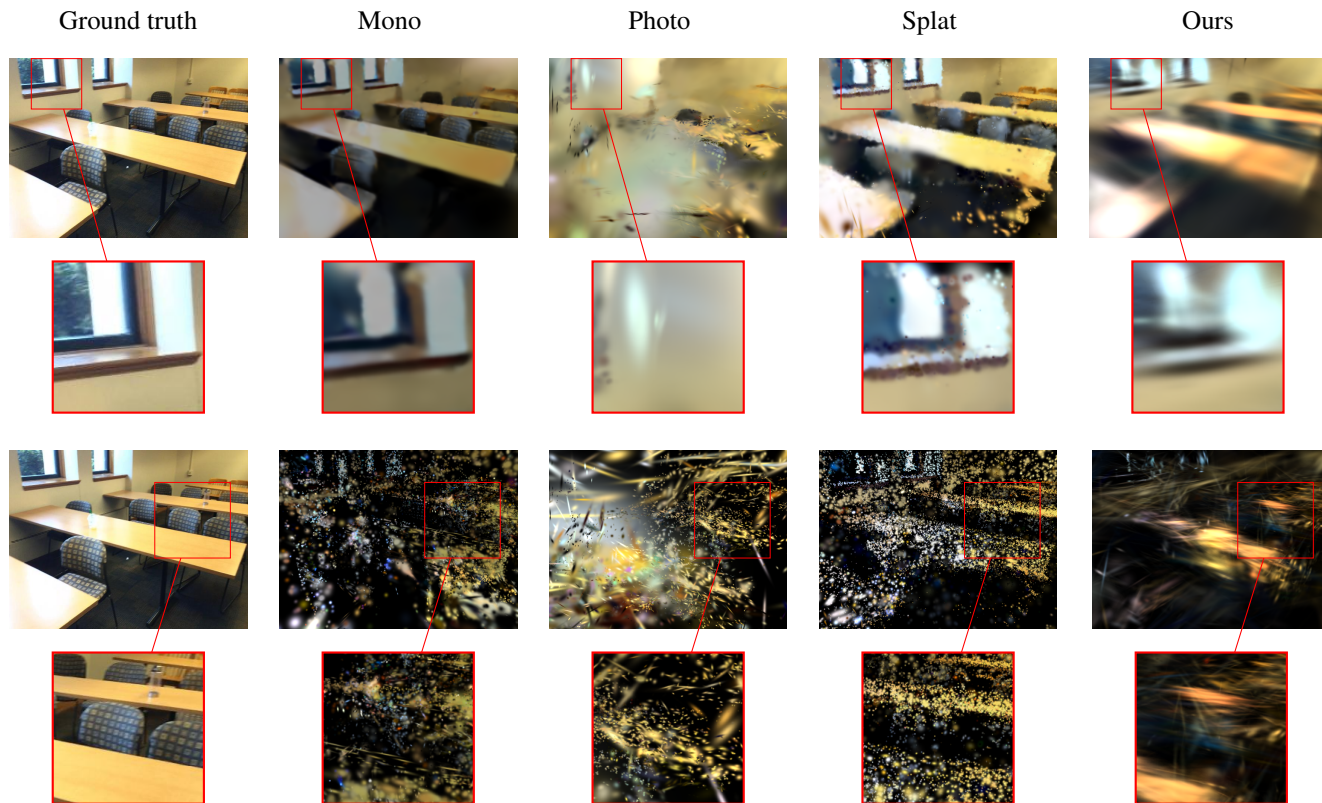


Figure 34. Qualitative evaluation on ScanNet 15.

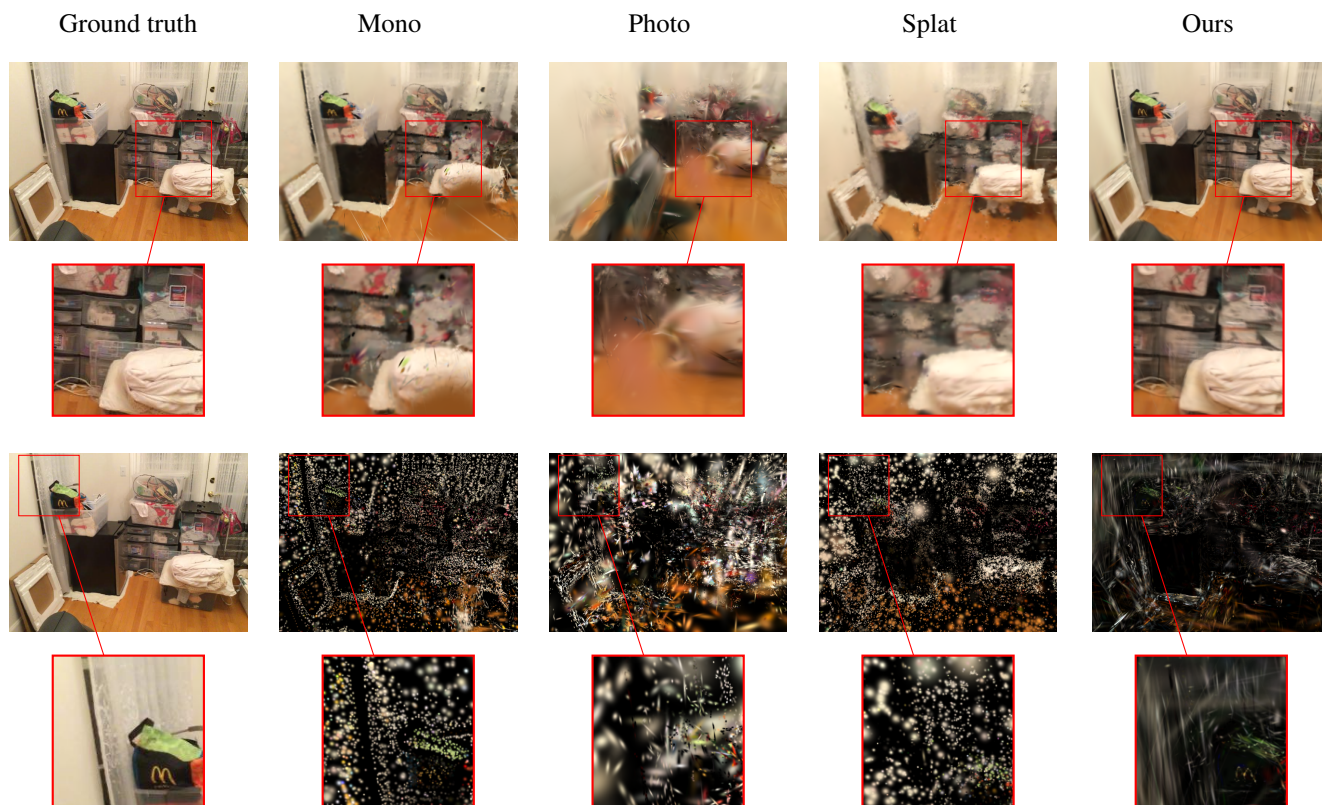


Figure 35. Qualitative evaluation on ScanNet 16 (Photo-SLAM failed tracking).