A Constrained Optimization Approach for Gaussian Splatting from Coarsely-posed Images and Noisy Lidar Point Clouds

Jizong Peng[1*], Tze Ho Elden Tse[2*], Kai Xu[2], Wenchao Gao[1], Angela Yao[2]
[1]dConstruct Robotics [2]National University of Singapore
{jizong.peng,wehchao.gao}@dconstruct.ai {eldentse,kxu,ayao}@comp.nus.edu.sg

In this supplemental document, we provide:
- details of dataset acquisition and pre-processing (Sec A);
- derivation of intrinsic refinement (Sec B);
- details of exposure compensation module (Sec C);
- additional analysis on test time adaptation (Sec D);
- details of line intersection-based depth estimation (Sec E);
- extended implementation details and discussion (Sec F);
- supplementary experimental results on GarageWorld dataset (Sec G);
- additional qualitative comparison on Waymo dataset (Sec H).

A. Dataset acquisition and pre-processing

In this section, we provide the configuration, calibration, and synchronization details of our SLAM hardware setup as well as the main pre-processing steps for the captured images.

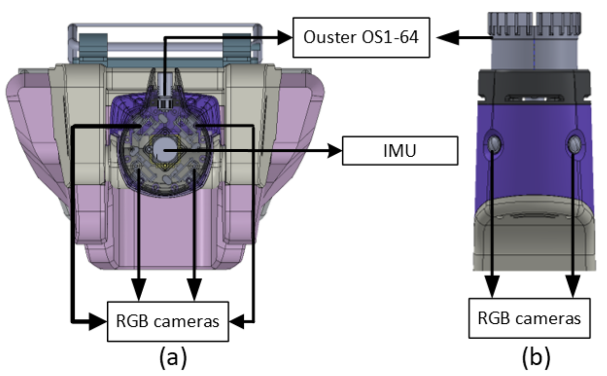

Figure 1. Key components of our SLAM hardware setup. The device includes a 64-channel mechanical Lidar on top, with four RGB cameras positioned at various angles to provide a complete 360-degree view. An IMU sensor is located directly beneath the Lidar. (a) System top-down view (b) front view.

*Equal contribution

Data acquisition: As illustrated in Fig. 1, Our self-developed device comprises an Ouster OS1-64 Lidar, four Decxin AR0234 cameras with wide-angle lenses, and a Pololu UM7 IMU. The Lidar is positioned at the top, with the IMU located directly beneath it. The IMU provides 9-DOF inertial measurements including rigid body orientation, angular velocity, and acceleration. These data are used in Lidar odometry optimization and Lidar points deskewing. The four wide-angle cameras are utilized to capture RGB features for 3D scene reconstruction.

The four cameras are parameterized using fisheye models. We calibrate them by employing OpenCV [1], where the intrinsic and distortion parameters of each camera are computed based on a calibration checkerboard. We then utilize the approach proposed in [6] to perform extrinsic parameter calibration, which describes the relationship between the Lidar and the four cameras.

An FSYNC/FSIN (frame sync) signal is utilized for time synchronization among multiple camera sensors, operating at 10 Hz, which results in the same capture frequency per camera. This sync signal consists of a pulse that goes high at the beginning of each frame capture to trigger all four camera shutters simultaneously.

The Ouster Lidar system works at 10 Hz, while the UM7 IMU provides data at a rate of 200 Hz. Unlike the hardware synchronization methods employed between cameras, the synchronization between the IMU and Lidar, as well as between the cameras and Lidar, is achieved solely through software, where timestamp data is utilized to align the outputs from the various sensors. Software synchronization is convenient and cost-effective, whereas the relatively noisy timestamps may result in less-than-optimal IMU pre-integration in SLAM and cause misalignment in point-cloud colorization.

With this SLAM setup, we present a new dataset which covers various environments, including three complex indoor scenes as well as a large-scale outdoor scene. We show a qualitative overview of this dataset in Fig. 2 and present the key statistics in Table 1.

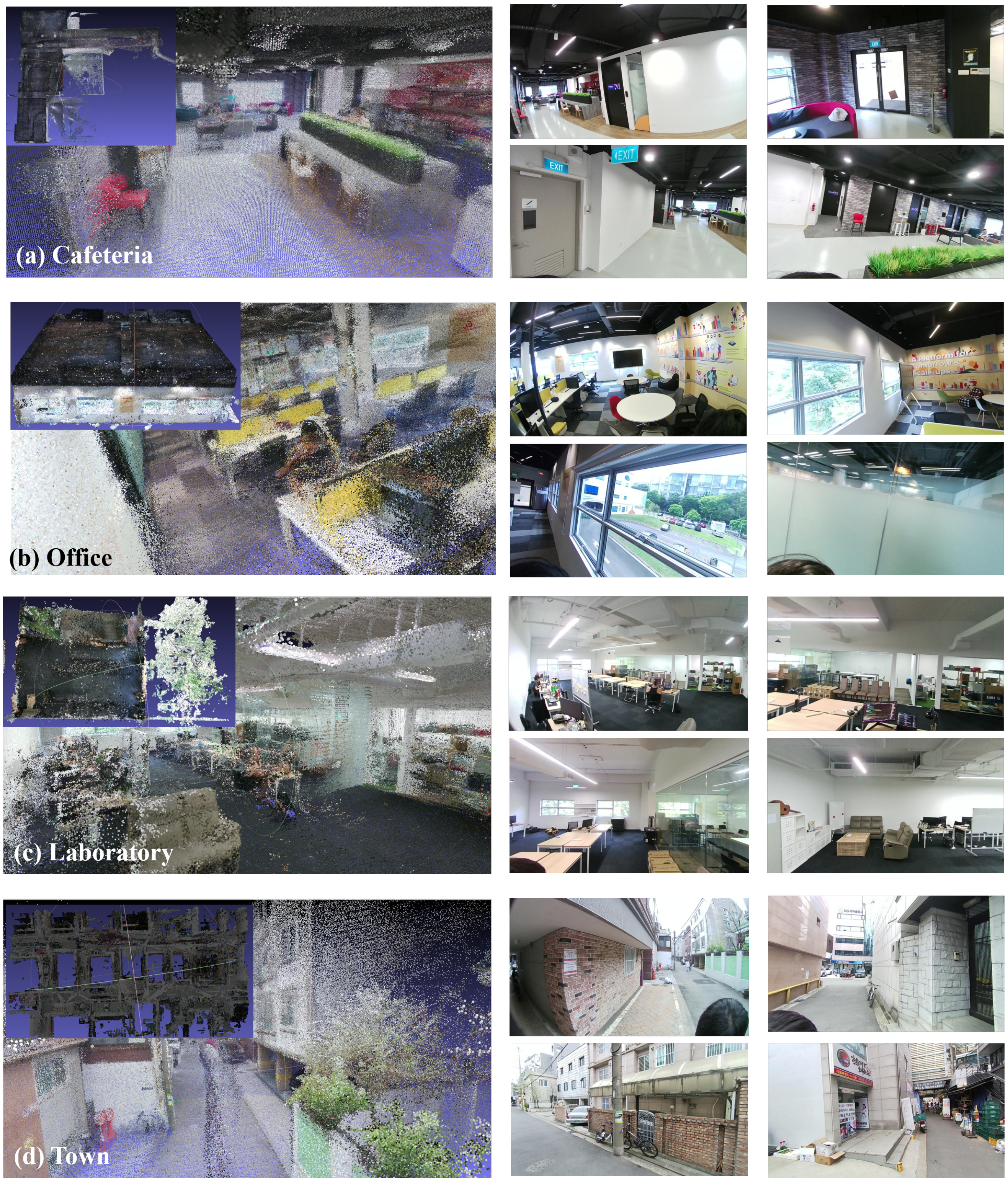

Figure 2. Qualitative examples of our proposed dataset. With our self-developed SLAM hardware system, we capture a new dataset comprising four scenes, including Cafeteria, Office, Laboratory and Town.

Table 1. Key statistics of our proposed dataset.

Scene	scene type	frame num	key-frame num	scene dimension	pcd size
Cafeteria	indoor	5788	1260	20×8	4 millions
Office	indoor	8184	1760	15×18	45 millions
Laboratory	indoor	5360	1216	15×9	57 millions
Town	outdoor	8816	1932	85×45	39 millions

Data pre-processing: We first undistort the wide-angle images based on the estimated intrinsic and distortion parameters from camera calibration, which produces prospectively correct images with a large FoV of 97°. To reduce any influence of dynamic objects on our 3D reconstruction process, we employ a publicly available `Yolo v8` model [13] that detects and spatially localizes passengers in these images. We exclude all pixels within their bounding boxes from further processing.

Our dataset offers dense point clouds for each scene, with a point number of 4-57 million points. As they often overpass the GPU memory limitation, we downsample the point cloud to a voxel size of 0.05 m in all experiments.

B. Derivation of intrinsic refinement

Most research employing 3DGS assumes the prior availability of accurate camera intrinsic parameters [4, 7, 10]. However, this assumption is difficult to fulfill, especially with SLAM devices that are equipped with multiple wide-angle cameras. Inaccurate intrinsic parameter estimates often lead to blurred reconstructed images, particularly at the image boundaries, as shown in Fig. 7 of the main paper. This issue is most severe in setups with multiple cameras, significantly degrading the quality of reconstruction. Despite its importance, this problem is frequently overlooked by the research community. We tackle this by enhancing the 3DGS rasterizer to refine imprecise camera intrinsic parameters during joint reconstruction optimization. This enhancement is achieved through an analytical solution, where the backward pass of the rasterization can be expressed as:

$$\partial\mathcal{L}/\partial f_x = \partial\mathcal{L}/\partial u \times \partial u/\partial f_x, \quad \partial\mathcal{L}/\partial f_y = \partial\mathcal{L}/\partial v \times \partial v/\partial f_y;$$
$$\partial\mathcal{L}/\partial c_x = \partial\mathcal{L}/\partial u \times \partial u/\partial c_x, \quad \partial\mathcal{L}/\partial c_y = \partial\mathcal{L}/\partial v \times \partial v/\partial c_y.$$

Following the chain rule, the initial terms in each equation are the partial derivatives from the loss to the uv variables, representing the screen coordinates of Gaussian ellipses. These derivatives are precomputed using the differentiable rasterizer. The subsequent terms are the derivatives of uv with respect to the intrinsic parameters, which have analytical solutions expressed as:

$$\partial u/\partial f_x = \vec{u}^x_{\text{cam}}/\vec{u}^z_{\text{cam}}; \quad \partial u/\partial c_x = 1$$
$$\partial v/\partial f_y = \vec{u}^y_{\text{cam}}/\vec{u}^z_{\text{cam}}; \quad \partial v/\partial c_y = 1$$

where $\vec{u}_{\text{cam}}$ represents the Gaussian mean in camera space, with its components $\vec{u}^x_{\text{cam}}$, $\vec{u}^y_{\text{cam}}$ and $\vec{u}^z_{\text{cam}}$ corresponding to the x, y and z dimensions, respectively.

C. Exposure compensation module

Our captures were taken in *uncontrolled* settings, where significant variations in lighting conditions exist during the data acquisition. Training directly with these images can introduce the floaters and degrade the scene geometry [3, 9]. To address this, we introduce an efficient exposure compensation module to handle issues related to illumination and exposure, drawing inspiration from [14] and [15]. We hypothesize that the variations in illumination are region-specific and affect the image's brightness in a gradual manner. Thus, our objective is simply to correct the illumination aspect of the images using a *learnable* and *low-frequency* offset.

In particular, for an image $I \in \mathbb{R}^{3\times h\times w}$, we initially transform it from the RGB color space to the YCbCr color space [12], denoted as $I_{\text{YCbCr}} \in \mathbb{R}^{3\times h\times w}$. In this transformed space, the first dimension $I_{\text{Y}} \in \mathbb{R}^{1\times h\times w}$ corresponds to the image luminance, representing brightness. The second and third dimensions, I_{Cb} and I_{Cr}, capture the chrominance, thereby defining the color context of the image. Our learnable offset $\Delta^{2\times h\times w}$ is applied on the luminance dimension of the image as a small affine transformation. This compensates for region-specific inconsistency caused by either lighting condition or auto-exposure changes, as follows:

$$I_{\text{Y}}^{'} = \Delta_{[0:1]} \times I_{\text{Y}} + \Delta_{[1:2]}. \tag{1}$$

We then obtain our resulting image by projecting the I_{YCbCr} with modified $I_{\text{Y}}^{'}$ back to the original RGB color space. The parameter Δ is defined per training image and is generated by a compact neural network implemented using `tinycudann` [11]. This network consists of a multi-resolution hash-encoding grid and a one-layer MLP. We set `n_lelves` to 2 with a base resolution of 8×8, which ensures that Δ can only capture coarse spatial information. More importantly, we further smooth Δ with a low-pass Gaussian filter with a large kernel size of 51×51 pixels. We illustrate our exposure compensation scheme in Fig. 3.

Since these offsets are not available for test images, we learn Δ per test image during the test-time optimization, together with the refinement of camera poses.

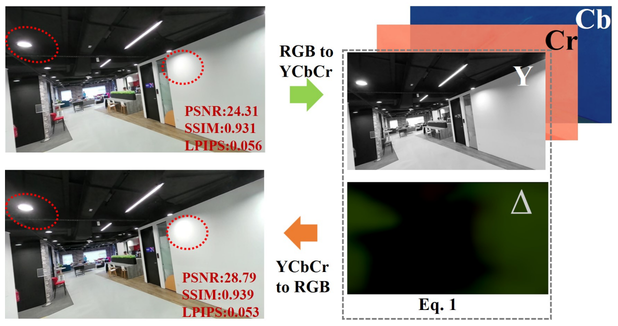

Figure 3. Illustration on our proposed exposure compensation module. In this approach, we project the image into YCbCr color space and only modify the channel representing illumination with a learnable low-frequency Δ. We observe that Δ mainly highlights strong lighting regions.

D. Ablations on test-time adaptation

For each test image, we keep the 3DGS parameters constant while refining the camera pose and learning the exposure compensation with low-frequency offset. Table 2 provides a detailed analysis of the impact these components have on the experimental results. Without applying either technique, we observe poor quality in novel-view renderings, primarily due to camera pose mismatches. Test-time pose optimization helps align the rendered image with the actual ground truth image, improving all visual metrics across both evaluated scenes, particularly the Cafeteria scene. On the other hand, using only exposure compensation did not significantly enhance metrics related to visual and structural similarities (SSIM and LPIPS), though it did moderately increase the Peak Signal-to-Noise Ratio (PSNR). As expected, this exposure correction module addresses exposure errors but struggles to capture high-frequency details. Combining both techniques results in the best reconstruction performance in the tested scenes.

Table 2. Ablations on test-time adaptation. **Pose** denotes pose refinement while **Expo.** represents exposure correction module.

Methods		Cafeteria			Laboratory		
Pose	**Expo.**	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓
✗	✗	19.80	0.7752	0.1144	22.52	0.8984	0.0939
✗	✓	22.65	0.7872	0.1102	27.93	0.9065	0.0881
✓	✗	23.04	0.9026	0.0876	22.67	0.9017	0.0933
✓	✓	28.58	0.9156	0.0820	28.18	0.9101	0.0875

E. Line intersection-based depth estimation

We compute the depth of two matched key-point pair by determining the intersection point of two lines defined by the camera origins and their view directions. We first consider two lines, l_1 and l_2, in 3D space, with origins $\vec{o}_1$ and $\vec{o}_2$, and directions $\vec{d}_1$ and $\vec{d}_2$, respectively. Our objective is to find the points on lines $\vec{l}_1$ and $\vec{l}_2$, parameterized by the scalars, t and s:

$$\vec{l}_1(t) = \vec{o}_1 + t \cdot \vec{d}_1,$$
$$\vec{l}_2(s) = \vec{o}_2 + s \cdot \vec{d}_2,$$

so that the distance between these two points $||\vec{l}_2(s) - \vec{l}_1(t)||^2$ are minimized (0 if the two lines intersect).

A necessary condition for this minimization is that the vector $\vec{l}_2(s) - \vec{l}_1(t)$ must be perpendicular to both $\vec{d}_1$ and $\vec{d}_2$, which can be expressed as:

$$\left(\vec{l}_2(s) - \vec{l}_1(t)\right) \cdot \vec{d}_1 = (\vec{x}_{21} + s \cdot \vec{d}_2 - t \cdot \vec{d}_1) \cdot \vec{d}_1 = 0,$$
$$\left(\vec{l}_2(s) - \vec{l}_1(t)\right) \cdot \vec{d}_2 = (\vec{x}_{21} + s \cdot \vec{d}_2 - t \cdot \vec{d}_1) \cdot \vec{d}_2 = 0,$$

where $\vec{x}_{21} = \vec{o}_2 - \vec{o}_1$ denotes the vector between the two origins. These conditions can be derived by setting the first-order gradient of the distance function to zero. By applying these two conditions, one can obtain the analytic solution, resulting in the following expressions:

$$t = \frac{||\vec{d}_2||^2 \cdot \vec{x}_{21} \cdot \vec{d}_1 - \vec{x}_{21} \cdot \vec{d}_2 \cdot (\vec{d}_1 \cdot \vec{d}_2)}{||\vec{d}_1 \cdot \vec{d}_2||^2 - ||\vec{d}_1||^2 ||\vec{d}_1||^2}, \quad (2)$$

$$s = -\frac{||\vec{d}_1||^2 \cdot \vec{x}_{21} \cdot \vec{d}_2 - \vec{x}_{21} \cdot \vec{d}_1 \cdot (\vec{d}_1 \cdot \vec{d}_2)}{||\vec{d}_1 \cdot \vec{d}_2||^2 - ||\vec{d}_1||^2 ||\vec{d}_1||^2}. \quad (3)$$

In our setting, for each pair of matched points, the origins $\vec{o}_1$ and $\vec{o}_2$ are defined as the camera centers. The directions $\vec{d}_1$ and $\vec{d}_2$ represents the vectors from these camera centers towards their respective image planes, determined by the uv coordinates, image dimensions, and intrinsic parameters. Note that the camera origins and directions are expressed in the world coordinates. Given t and s, we can compute the depth of these two matched pixels by applying the viewing matrix and extracting the z-axes element. We disregard matched pairs where t or s is zero or negative, as the line intersection must be in front of both cameras. Additionally, we ignore pairs with very small angles (less than $2°$) between $\vec{d}_1$ and $\vec{d}_2$, as this makes Eqs. 2 and 3 unstable due to very small denominators.

F. Extended implementation details and discussion

In this section, we provide implementation details of our proposed constrained-optimization based method, as well as the comparison approaches.

All experiments were conducted on a machine equipped with an Intel-14900K CPU and an NVIDIA 4090 GPU. Our

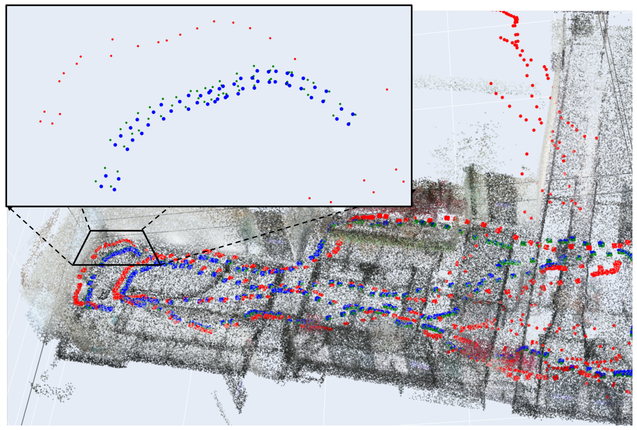

Figure 4. Camera pose visualization for the Cafeteria scene. Red and green points represent the trajectories estimated by 3DGS-COLMAP and 3DGS-COLMAP$^{\triangle}$, respectively. 3DGS-COLMAP fails to capture the geometry structure, while our method, shown in blue, converges very similarly to 3D-COLMAP$^{\triangle}$ but with better visual quality (as shown in Table 1 in the main paper).

framework is based on the open-source differentiable rasterizer [8, 17], with modifications to accommodate non-centric images, enable differentiable depth rendering, and ensure differentiability in both extrinsic and intrinsic parameters. To facilitate optimization and avoid sub-optimal solutions, we employed a cosine learning rate decay strategy with restarts. Specifically, we increased the learning rate and performed the decay three times during the optimization process, starting at the 1st, `max_iter`/6, and `max_iter`/2 iterations. Considering that the point clouds roughly capture the scene geometry, we disabled the pruning operation during optimization for all experimental variants, while enabling Gaussian point densification starting 67% of its training.

In the following, we provide the implementation details

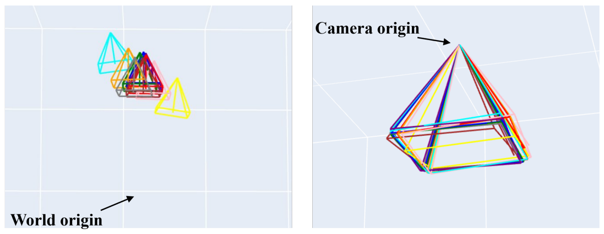

Figure 5. Qualitative comparison for two camera pose optimization approaches. Different colors represent camera poses at various time during optimization. **Left:** camera poses are optimized by rotating around the world origin (Eq. 6 in main paper). **Right:** camera poses are rotated around the initial camera origin (Eq. 7 in main paper). Our proposed approach on the right demonstrates better optimization robustness.

on comparing methods:

- **3DGS-COLMAP**: We enhanced this widely-used baseline by associating the camera information with RGB images. This was achieved by modifying the `database` file of generated by COLMAP software, with the intrinsic estimations as a prior.
- **3DGS-COLMAP$^{\triangle}$**: The next method takes the initial camera poses as additional priors and perform rig-based bundle adjustment. This is achieved using COLMAP's `point_triangulator` and `rig_bundle_adjuster` interfaces.
- **CF-3DGS** [5]: This approach incrementally estimates the camera poses based solely on visual images, which utilizes two distinct Gaussian models: a local model and a global model. The local Gaussian model calculates the relative pose differences between successive images, while the global Gaussian model aims to model the entire scene and refine the camera poses derived from the local model. Since this approach is designed for a monocular camera configuration and requires video-like input, we provide our images on a per-camera basis to ensure compatibility. Unexpectedly, this method failed to capture the geometry after processing approximately 10 images, resulting in significantly poor rendering. This issue is primarily due to our key frames having a moderate covisibility threshold. Additionally, the frames exhibit a repetitive block pattern and feature plain surfaces in many scenes, which impede this visual-based method from accurately estimating the camera poses.
- **MonoGS** [10]: Similar to the previous approach, this technique incrementally reconstructs the scene while simultaneously estimating camera positions by optimizing for photometric loss and depth inconsistency loss. In our experiments, we found that this baseline faces a similar challenge as CF-3DGS, specifically, a difficulty in accurately capturing the true geometry from a diverse and uncontrolled set of images. Consequently, it produces entirely empty images after processing about 15 images across all tested scenes. We interrupt and restart the training when it fails completely, continuing this process until the method can provide a test score on our designated set of test images
- **InstantSplat** [4]: This approach uses 3D foundation models to generate a dense and noisy point cloud, which is then optimized along with the camera extrinsics. Originally designed for sparse-view synthesis, we found it challenging to handle more than 30 images due to GPU memory limitations. To adapt this method to our context, we selected a sequence of 30 images, consisting of 29 consecutive training images and one test image strategically placed in the middle. The individual test score is computed on the sub-model, which requires one minute of pre-processing and 50 seconds of training time. We

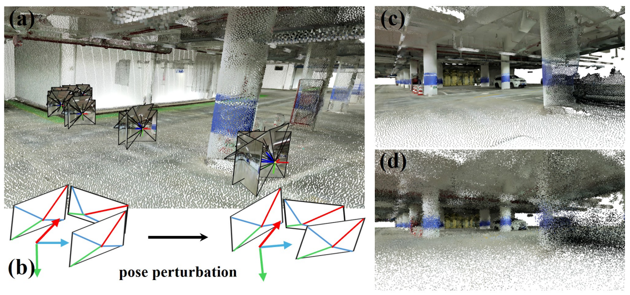

Figure 6. Illustration of the `Garage World` dataset with four undistorted cameras oriented in various directions. (b) We perturbed the camera poses using pose decomposition. (c) and (d) show the point cloud both before and after the introduction of perturbations.

report the test score based on the average of multiple sub-models.

- **LetsGo** [2]: Similar to ours, this approach proposed to integrate high-quality point cloud and camera poses with enhanced 3DGS technology. We follow their open-sourced implementation[1], default training parameters, and test it on different sequences of GarageWorld and Waymo datasets.
- **StreetGS** [16]: The last multi-modality method aims to reconstruct dynamic driving scenes with dynamic and static Lidar point clouds and high quality camera poses. Similar to our 3DGS-COLMAP baselines, this baseline first refines the camera poses using COLMAP and then optimize each camera pose independently during the reconstruction. We follow the default setting in their open-sourced implementation[2] to test this method on both Waymo and GarageWorld datasets, except that we only reconstruct the static scene while ignoring the moving objects.

We present in Fig. 4 the pose estimation results from 3DGS-COLMAP, 3DGS-COLMAP$^{\triangle}$, and our method. As illustrated, 3DGS-COLMAP fails in this scene due to repeated block patterns and plain surfaces. We also show in Fig. 5 the qualitative examples for two different camera pose refinement approaches. We observe that using the Eq. 7 in the main paper results in a more stable optimization trajectory.

G. Extended experimental setup and results on `GarageWorld` dataset

We are particularly interested in `GarageWorld` [2] dataset due to its high relevance to our work. We conducted extensive experiments on this dataset to validate the robustness of our proposed method. Unlike our collected dataset,

[1]https://github.com/zhaofuq/LOD-3DGS
[2]https://github.com/zju3dv/street_gaussians

Table 3. Quantitative comparisons on the (perturbed) Garage World dataset. We show that our proposed method can consistently improve the performance despite large perturbations.

Noise Level	Method	Group 0			Group 6		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
-	**3DGS**	25.43	0.8215	0.2721	21.23	0.7002	0.4640
	Ours	26.06	0.8325	0.2606	23.76	0.7779	0.3537
0.3°	**3DGS**	23.17	0.7595	0.4033	21.00	0.6979	0.5085
	Ours	25.12	0.8060	0.3110	23.06	0.7515	0.4004
0.6°	**3DGS**	22.07	0.7388	0.4645	20.58	0.6874	0.5359
	Ours	23.09	0.7594	0.3995	21.94	0.7160	0.4611

this dataset provides *highly accurate* camera poses and *very clean* point cloud but with only one fisheye camera. Fortunately, four pinhole images are undistorted from the same wide-angle image with fixed view directions: Front, Left, Right, and Up, as shown in Fig. 6 (a). We therefore consider this dataset as an image collection from multiple-camera setup and decompose the camera poses into device-center and camera-to-device transformations. We further test our method against 3DGS baseline [8] on two sequences, Group 0 and Group 6, randomly drawn from the campus scene. This extended experimental results are shown in both Table 3 and Fig. 7.

The first two rows of Table 3 show the experimental results under the ideal conditions. Due to the high-quality camera poses and the clean point cloud, the reconstruction performance for 3DGS reaches a PSNR score of 25.43 and 21.23 dB for both scenes. Notably, our proposed method consistently outperforms the baseline across both scenes and all visual metrics. The rendered test images exhibit clearer edges and more detailed context, which can be attributed to our method's ability to mitigate even subtle intrinsic and extrinsic errors encountered during time-intensive acquisitions with complex hardware.

Our next series of experiments aim to demonstrate the robust capabilities of our proposed method using a dataset with varying levels of perturbation. To achieve this, we introduce Gaussian noise to both the device-center and camera-to-device poses, as well as to the point cloud, creating synthetic datasets with degradations. This process is illustrated in Fig. 6 (b) and (d).

The third and fourth rows of Table 3 present experimental results under conditions of mild degradation. Both camera-to-device and device transformations were adjusted with a random Gaussian noise limited to 0.3°in their orientations. Additionally, random Gaussian noise confined to 0.01 m was added to the initial point cloud. This noise negatively affects the 3DGS baseline performance, whereas our proposed method shows quality improvements by 1.95 dB, 4.65%, and 9.23% across the three visual criteria. In the second scene, there is an enhancement of 2.06 dB, 5.36%, and 10.8%.

The final two rows in Table 3 represent the performance of both methods under greater perturbations, with orientations disturbed up to 0.6°. Our method enhances reconstruction performance in both evaluated scenes, particularly for the LPIPS metric, and maintains credible rendering quality despite the challenging conditions.

H. Qualitative comparison on Waymo dataset

As shown in Fig. 8, we present qualitative comparisons of our proposed method against state-of-the-art multimodal 3DGS approaches which integrate cameras, Lidars, and inertial sensors. We show that our method can better reconstruct scene geometries, as evidenced by straight rendered streetlights, and achieves a higher level of detail in the final rendering. These improvements demonstrate the effectiveness of our approach in capturing fine-grained structural and textural information, leading to a more realistic and visually consistent representation of the scene.

References

[1] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000. 1

[2] Jiadi Cui, Junming Cao, Yuhui Zhong, Liao Wang, Fuqiang Zhao, Penghao Wang, Yifan Chen, Zhipeng He, Lan Xu, Yujiao Shi, et al. Letsgo: Large-scale garage modeling and rendering via lidar-assisted gaussian primitives. *arXiv preprint arXiv:2404.09748*, 2024. 6

[3] François Darmon, Lorenzo Porzi, Samuel Rota-Bulò, and Peter Kontschieder. Robust gaussian splatting. *arXiv preprint arXiv:2404.04211*, 2024. 3

[4] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, Zhangyang Wang, and Yue Wang. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds, 2024. 3, 5

[5] Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A. Efros, and Xiaolong Wang. Colmap-free 3d gaussian splatting. In *CVPR*, 2024. 5

[6] Arturo de la Escalera Fernando García Jorge Beltrán, Carlos Guindel. Automatic extrinsic calibration method for lidar and camera sensor setups. *IEEE Transactions on Intelligent Transportation Systems*, 2022. 1

[7] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat track & map 3d gaussians for dense rgb-d slam. In *CVPR*, 2024. 3

[8] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM ToG*, 2023. 5, 6

[9] Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, et al. Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5166–5175, 2024. 3

[10] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *CVPR*, 2024. 3, 5

[11] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022. 3

[12] Hideki Noda and Michiharu Niimi. Colorization in ycbcr color space and its application to jpeg images. *Pattern recognition*, 40(12):3714–3720, 2007. 3

[13] Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi. Real-time flying object detection with yolov8. *arXiv preprint arXiv:2305.09972*, 2023. 3

[14] Christian Reiser, Richard Szeliski, Dor Verbin, Pratul P. Srinivasan, Ben Mildenhall, Andreas Geiger, Jonathan T. Barron, and Peter Hedman. Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *SIGGRAPH*, 2023. 3

[15] Yuehao Wang, Chaoyi Wang, Bingchen Gong, and Tianfan Xue. Bilateral guided radiance field processing. *ACM TOG*, 2024. 3

[16] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street Gaussians: Modeling Dynamic Urban Scenes with Gaussian Splatting. In *ECCV*, 2024. 6

[17] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for Gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024. 5

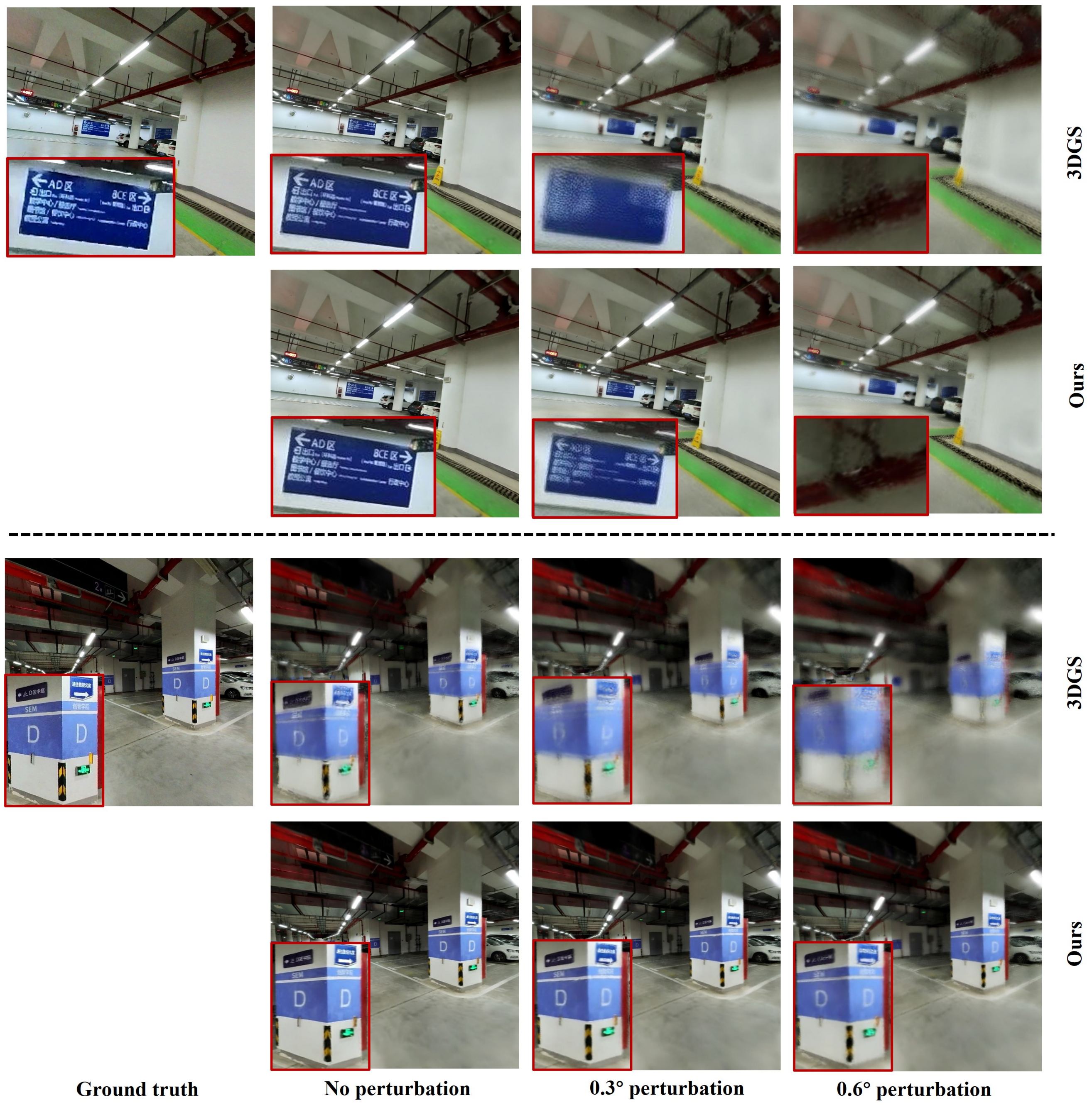

Figure 7. Qualitative comparison of our constrained optimization approach with the 3DGS baseline. The top and bottom respectively show clean and perturbed scenes for groups 0 and 6 at different levels. We show that our method enhances visual quality in the presence of camera pose errors and maintains better quality even without noise injection.

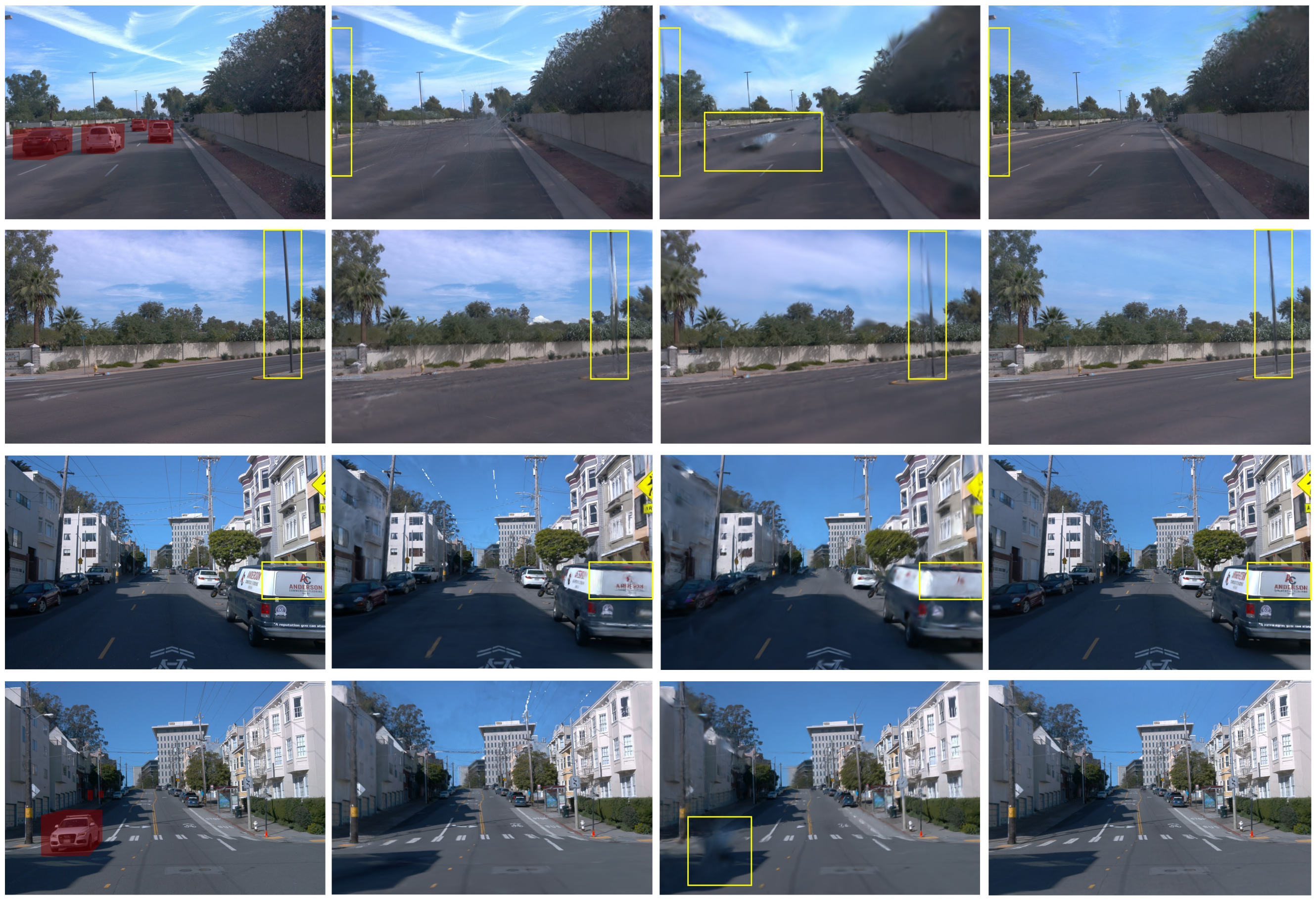

Figure 8. Qualitative comparison of our constrained optimization approach with multimodal methods. We overlay the dynamic object mask on the ground truth images to highlight the static regions on which our metrics are computed. We show that our proposed method offers better scene geometry and rendering details compared with state-of-the-art approaches.