# Supplementary Material – Gaussian Splatting Map Registration with Orthographic Bird's-Eye-View Renderings

H. Leblond[1] G. Simon[1] R. Martins[2] C. Demonceaux[2] M.-O. Berger[1]
[1]Université de Lorraine, INRIA, LORIA, France
[2]Université Bourgogne Europe, CNRS, ICB, France
{hugo.leblond, gilles.simon, marie-odile.berger}@inria.fr
{renato.martins, cedric.demonceaux}@ube.fr

Experiments were conducted on a Linux workstation running Ubuntu 22.04 LTS with an Intel Xeon W-2265 CPU (12 cores, 24 threads, 3.5–4.8 GHz), 125 GB RAM, and an NVIDIA A5000 GPU (24 GB VRAM).

## 1. Cross-GS Scene Representation Characteristics

From Table 1, we observe clear differences in how each Gaussian Splatting (GS) variant represents a scene. LightGaussian produces fewer splats and a smaller memory footprint compared to 3DGS, while maintaining moderate opacity and density. SuGaR, on the other hand, generates more numerous, finer-scaled, and denser splats, resulting in the largest memory usage. These differences are consistent across both the *Truck* and *Caterpillar* scenes.

Such variations in splat number, scale, density, and memory help explain why cross-variant registration is more challenging. When submaps are reconstructed with different GS variants, differences in point distribution, opacity, and scale can lead to misalignments or reduced matching performance. Nonetheless, understanding these characteristics provides important context for evaluating registration methods: LightGaussian prioritizes compactness, SuGaR emphasizes fine-grained detail, and 3DGS sits in between. These insights clarify why intra-variant registration tends to be easier than cross-variant registration, and they also help explain why our method maintains strong performance even when submaps originate from architectures with differing GS representations.

To complement these quantitative measures, the Figure 1 shows RGB renderings and density maps for the *Truck* and *Caterpillar* scenes under different GS methods. These visuals highlight structural, appearance, and density differences that result from reconstruction method choices and can impact downstream tasks such as cross-model registration.

## 2. Illumination Changes Qualitative Results

Figure 2 presents the two scenes *Galerie Europa* and *Ludwigskirche* from the NeRF-OSR dataset [10] used to evaluate the effect of illumination variations on scene registration. For each scene, we show images captured under varying lighting conditions along with their corresponding Gaussian Splatting reconstructions. These visualizations serve to illustrate the experimental setup and provide a qualitative reference for how submaps appear under varying illumination, supporting the analysis of registration performance across lighting changes.

| Scene | Method | #GS | Mean Opacity | Mean Scale | Density | Memory (MB) |
|---|---|---|---|---|---|---|
| Truck | 3DGS | 1,583,455 | 0.18 | 0.018 | 0.037 | 392.7 |
| Truck | LightGaussian | 973,498 | 0.31 | 0.028 | 0.053 | 241.4 |
| Truck | SuGaR | 2,112,252 | 0.35 | 0.005 | 0.019 | 523.8 |
| Caterpillar | 3DGS | 1,117,949 | 0.21 | 0.027 | 0.064 | 277.3 |
| Caterpillar | LightGaussian | 589,930 | 0.36 | 0.040 | 0.093 | 146.3 |
| Caterpillar | SuGaR | 2,062,182 | 0.295 | 0.008 | 0.030 | 511.4 |

Table 1. Statistics of different GS methods across two scenes.

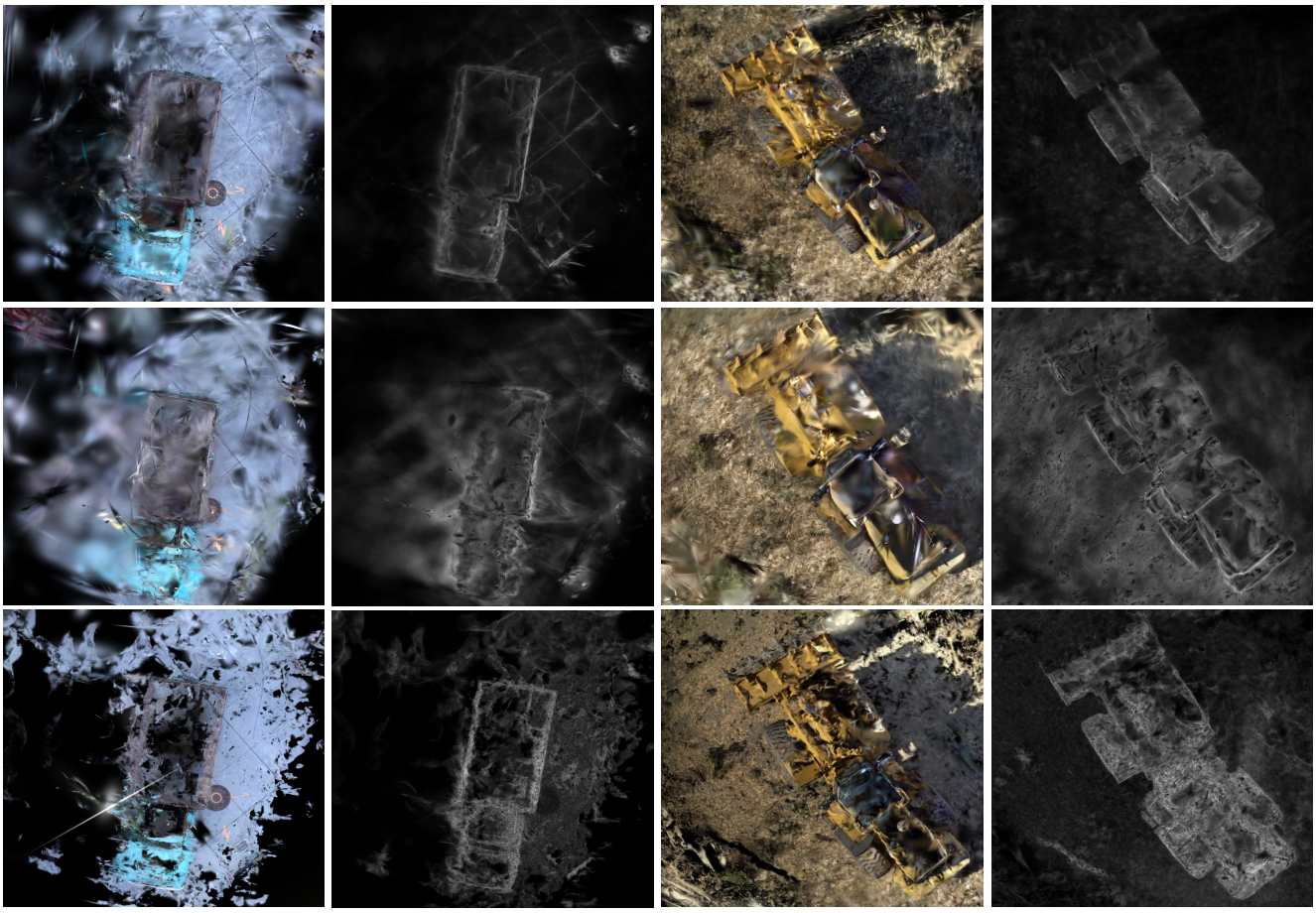

Figure 1. Qualitative comparison of different Gaussian Splatting reconstructions on the *Caterpillar* and *Truck* scenes from the Tanks and Temples [7] dataset with 0.2 overlap. For each scene, the first row shows the RGB rendering and density visualization from the original 3DGS model [6], the second row shows the same views for the more compact LightGaussian [2] model, while the third row shows the same views for the SuGaR [5]. These visualizations highlight structural and appearance differences that impact cross-model registration.

Table 2. Processing times and image sizes for the smallest and largest BEV images generated from the Tanks and Temples dataset [7]. The smallest and largest images correspond to the *Francis* and *Truck* scenes, respectively, when the dataset is sampled at 60% overlap. For each scene, we report the smallest and largest images along with corresponding processing times and peak GPU memory usage.

| Scene | BEV Image Size 1 (px) | BEV Image Size 2 (px) | Processing Time 1 (s) | GPU memory footprint (MB) |
|---|---|---|---|---|
| Francis (smallest) | 1172 × 1617 | 429 × 464 | 7.7 | 1064 |
| Truck (largest) | 3860 × 3424 | 3205 × 3290 | 13.2 | 1813 |

## 3. Scalability Analysis

The Tanks and Temples dataset [7] contains large-scale outdoor scenes captured with high-resolution images. Since the 3D reconstructions produced by COLMAP are not in absolute metric scale, the BEV resolution should be interpreted relatively; for reference, the average BEV image size is 2458 × 2464 pixels. Table 2 reports the results on this dataset, showing the image sizes and processing times for the scenes with the smallest and largest BEV images.

To further analyze scalability, we also conduct a complementary experiment where the BEV image resolution is varied for a fixed scene in order to study the impact on memory footprint, GPU consumption, and inference time. Table 3 illustrates the effect of BEV resolution on image size, GPU memory usage, and processing time for the *Barn* scene. While memory consumption grows rapidly with resolution due to the quadratic increase in image size, processing time increases more grad-

Figure 2. Examples of illumination variation in two scenes from the NeRF-OSR dataset. The first two columns show the *Galerie Europa* scene at different times of day, and the last two columns show the *Ludwigskirche* scene under similarly varying lighting conditions. Each column displays a scene image along with its corresponding Gaussian Splatting (GS) representation.

Table 3. Processing times, image sizes, and resolutions for the *Barn* scene from the Tanks and Temples dataset [7] sampled at 60% overlap. For each BEV resolution, we report the generated image size, processing time, and peak GPU memory usage.

| BEV Resolution | BEV Image 1 Size (px) | BEV Image 2 Size | Processing Time (s) | GPU memory footprint (MB) |
|---|---|---|---|---|
| 0.0015 | 10350 × 8965 | 8415 × 10175 | 26.6 | 5968 |
| 0.002 | 7762 × 6723 | 6311 × 7631 | 20.1 | 3663 |
| 0.005 | 3105 × 2689 | 2524 × 3052 | 12.2 | 1332 |
| 0.05 | 310 × 268 | 252 × 305 | 10.8 | 1223 |
| 0.5 | 31 × 21 | 25 × 30 | 10.9 | 1222 |

ually, exhibiting sublinear scaling relative to image size. This indicates that, although high-resolution BEV images require more memory, the computational cost does not increase proportionally, making high-resolution reconstruction feasible when sufficient GPU memory is available. Overall, these results highlight the scalability of the approach, as larger scenes and higher resolutions can be handled with only modest impact on inference time.

We evaluated our approach on two datasets with very different environments: the urban **Block_1** part of the MatrixCity dataset [8] and the desert BASEPROD-UAV dataset [3, 4]. For both datasets, we applied the same splitting strategy with a 40% overlap and transformed the data using the method described in the main paper. **Block_1** spans roughly 1.2 $km^2$, while BASEPROD-UAV covers approximately 0.12 $km^2$. To keep the BEV images manageable, we reduced the resolution to 0.1 m per pixel, resulting in image sizes of approximately 12k $\times$ 10k for Block_1 and 3200 $\times$ 2700 for BASEPROD-UAV—still sufficiently high for city- and terrain-scale data, respectively.

Despite the differences in scale, appearance, and environmental conditions, our approach performs robustly across both datasets. In the urban scenario of **Block_1**, registration achieves a Relative Translation Error (RTE) of 0.0001 and a Relative Rotation Error (RRE) of 0.043. On the outdoor desert scenes of BASEPROD-UAV, the method generalizes well, yielding an RTE of 0.0041 and an RRE of 0.446. Figure 3 illustrates the scene extent of **Block_1** and BASEPROD-UAV along with a corresponding BEV image.

## 4. Impact of Noisy Gravity Vector

### 4.1. Impact of Gravity Vector Noise on Registration Metrics

Perturbation experiments with synthetic noise indicate that fixed-magnitude errors of 1° and 2° with random orientation on both scenes reduce the success ratio from 0.899 to 0.717 (Table 4), yet VP-based estimates remain reliable due to their consistency across images of the same scene. This consistency maintains BEV alignment and mitigates the effect on rotation estimation. To validate this interpretation, we added identical noise to both gravity vectors (Table 4), which caused only a minor reduction in success ratio. These results indicate that the alignment of BEV images depends primarily on relative consistency rather than absolute accuracy, and that errors not originating from image-based zenith detection can reduce the

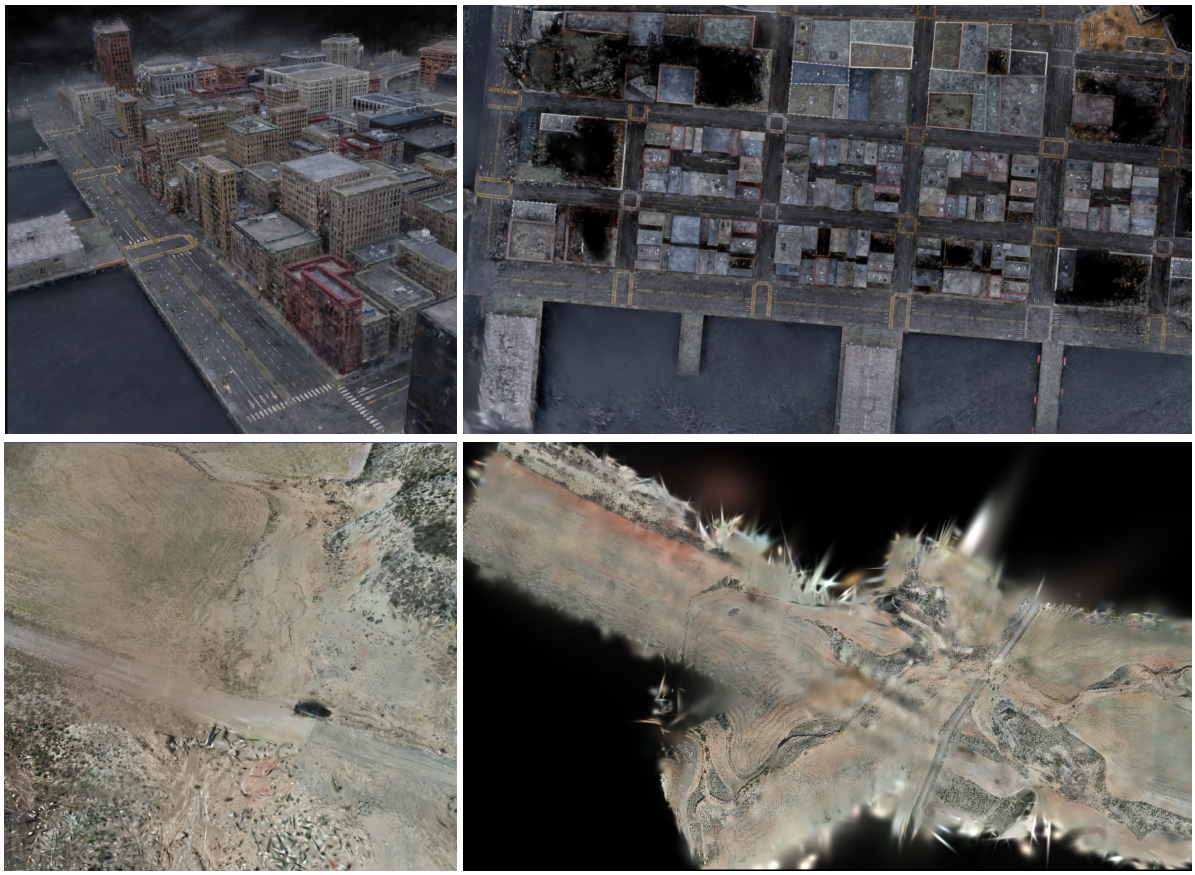

Figure 3. Left: 3DGS representation of a part of the scene created from `block_1` and BASEPROD data. Right: Example of a BEV generated at a resolution of 0.1 m per pixel.

Table 4. Impact of gravity vector noise on registration metrics for Tanks and Temples [7].

| | Independent noise | | | Identical noise | | |
|---|---|---|---|---|---|---|
| Noise (°) | SR↑ | RTE↓ | RRE↓ | SR↑ | RTE↓ | RRE↓ |
| 0 | 0.974 | 0.029 | 0.147 | 0.974 | 0.029 | 0.147 |
| 1 | 0.899 | 0.068 | 0.125 | 0.949 | 0.035 | 0.112 |
| 2 | 0.717 | 0.094 | 0.086 | 0.949 | 0.039 | 0.156 |

number of valid correspondences, slightly lowering the success ratio compared to the *Ours + VP* configuration.

### 4.2. Qualitative Analysis of Vanishing Points Estimation

In this section, we show the reliability of our vanishing point (VP)–based gravity estimation. The zenith vector is computed for each image in a sequence, but its accuracy depends on the complexity of the image. Examples of challenging cases include situations where the detected vertical segments are either scarce or very short, or when the scene contains directions that are close to vertical (e.g., barriers or fences). We thus use a robust estimate of the gravity vector and select a representative gravity by computing the medoid of all estimates. It is defined as the vector minimizing the sum of cosine distances to all other estimates in the sequence.

Figure 4 illustrates this process for three sequences (*Barn*, *Ignatius*, *Caterpillar*) of increasing complexity. Each blue curve shows (left) the angular deviation in degrees of the estimated VPs from the ground-truth zenith along the sequence, while the red point indicates the medoid-estimate that best represents the entire set. The images on the right show the detected segments corresponding to the zenith in the image corresponding to the medoid. The blue curve is computed from the ground

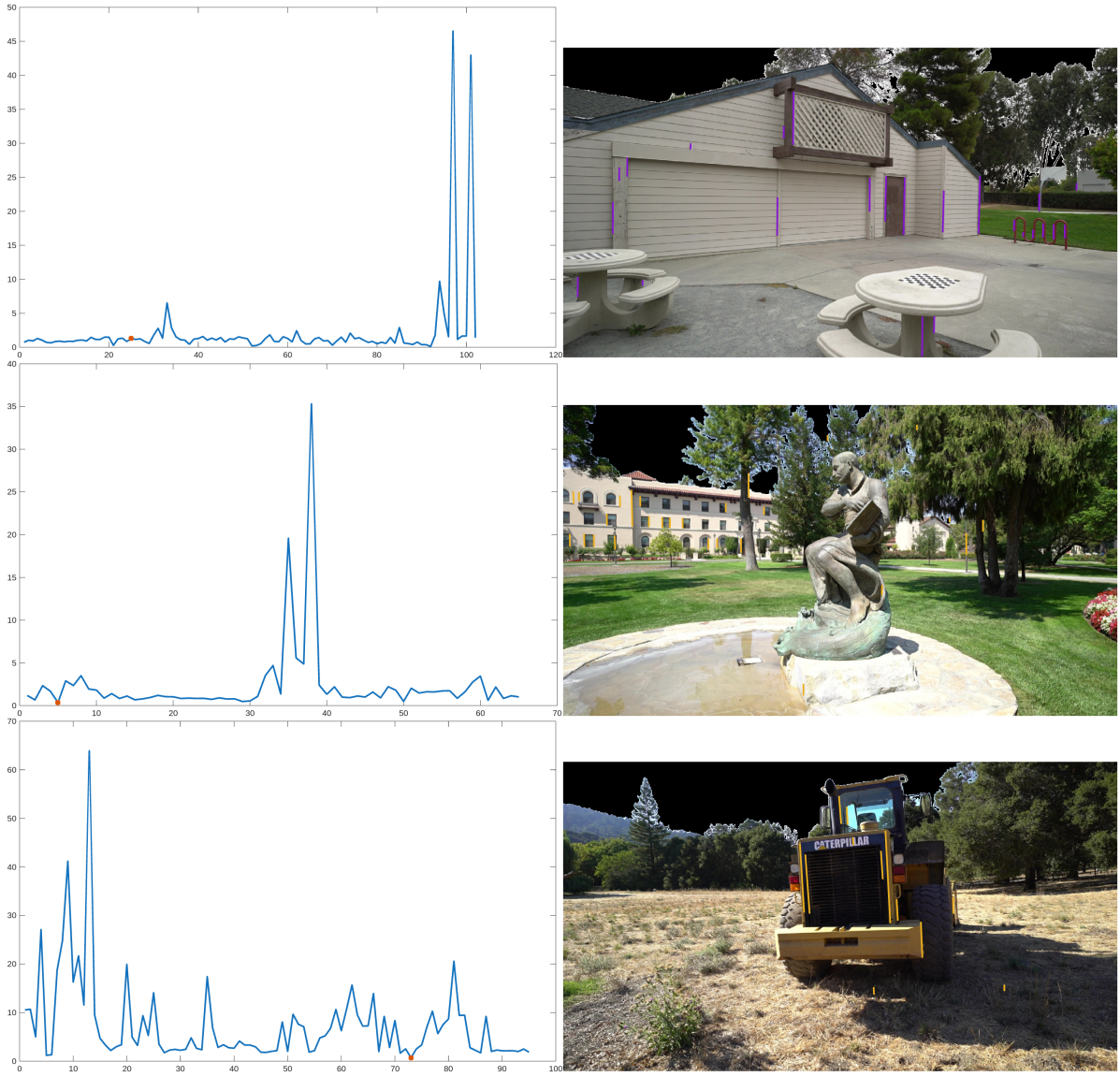


Figure 4. Medoid-based vanishing point (VP) estimation for three sequences of increasing complexity: *Barn*, *Ignatius*, and *Caterpillar*. Left: angular deviation (in degrees) of each estimated VP from the ground-truth zenith; the red point indicates the medoid, representing the most consistent estimate across the sequence. Right: detected segments corresponding to the zenith in the medoid image. Despite challenges such as sparse or ambiguous vertical structures, the medoid VP provides a stable, representative gravity vector without relying on ground-truth information. Better seen when zooming.

truth gravity vectors, but the medoid is computed directly from the estimated VP. In a relatively easy case as Barn, the estimated gravity vectors are accurate for almost all the image sequences. Ignatius is more challenging since there are only a few vertical structures in the background. In addition, some trunks of trees are also detected as vertical lines. This leads to a less accurate gravity vector. Caterpillar is even more challenging with very few vertical structures. The figure demonstrates that medoid VP estimates are close to the ground truth and that the medoid provides a stable, representative gravity vector, highlighting the robustness of our approach without relying on ground-truth information.

## 5. Implementation Details

### Rendering Thresholds

To prevent excessively large renders caused by outlier points in the GS maps, we discard Gaussians that are significantly distant from the median position of each GS scene centroid. Specifically, the geometric consistency threshold is set to **6.0** for the **ScanNet dataset** (metric scale) and **10.0** for the **Tanks and Temples dataset** (COLMAP-scaled). In addition, we apply a **KNN filter** with a neighborhood size of 20 and a radius of 1 to remove isolated Gaussians that could introduce noise in the final render.

### Matching Details

We adopt **SuperPoint** [1] for keypoint detection and description, followed by **LightGlue** [9] for feature matching, processing the density and RGB renders in parallel. To balance speed and accuracy, SuperPoint is configured with a maximum of **2048 keypoints**. For feature matching, LightGlue is parametrized with the following settings: *n_layers* = 9, *flash* = True, *mp* = False, *depth_confidence* = 0.95, *width_confidence* = 0.99, and *filter_threshold* = 0.1. The matches obtained from density and RGB modalities are then fused to compute a joint set of correspondences.

## References


[1] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description, 2018. 6

[2] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps, 2023. 2

[3] Levin Gerdes and Carlos Jesús Pérez del Pulgar Mancebo. Baseprod-uav, 2025. 3

[4] Levin Gerdes, Tim Wiese, Raúl Castilla-Arquillo, Laura Bielenberg, Martin Azkarate, Hugo Leblond, Felix Wilting, Joaquín Cortés, Alberto Bernal, Santiago Palanco, and Carlos Perez-del Pulgar. Baseprod: The bardenas semi-desert planetary rover dataset. *Scientific Data*, 11:1–15, 2024. 3

[5] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *CVPR*, 2024. 2

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4), 2023. 2

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

[8] Yixuan Li, Lihan Jiang, Linning Xu, Yuanbo Xiangli, Zhenzhi Wang, Dahua Lin, and Bo Dai. Matrixcity: A large-scale city dataset for city-scale neural rendering and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3205–3215, 2023. 3

[9] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed, 2023. 6

[10] Viktor Rudnev, Mohamed Elgharib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Nerf for outdoor scene relighting. In *European Conference on Computer Vision (ECCV)*, 2022. 1