ODGS-SLAM: Omnidirectional Gaussian Splatting SLAM

Stefan Spiss Joey Hieronimy Marcel Ritter Matthias Harders
Interactive Graphics and Simulation Group, University of Innsbruck
stefan.spiss@student.uibk.ac.at

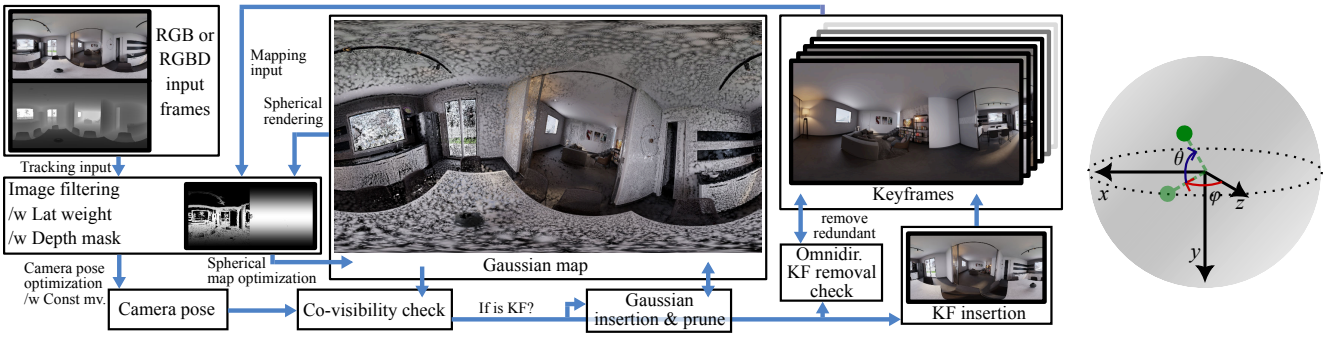

Figure 1. (Left:) Overview of the ODGS-SLAM system. (Right:) Following [22], any ray can be represented in the spherical coordinate system $\mathbb{S}^2$ with the angles $\phi \in [-\pi, \pi]$ and $\theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

Abstract

This work presents ODGS-SLAM, an omnidirectional simultaneous localization and mapping (SLAM) system utilizing 3D Gaussian Splatting (3DGS) as the unified representation for tracking and mapping. Thus, it reconstructs scene geometry from panoramic image sequences (RGB or RGBD) via splats while also estimating the camera poses. Such a framework is important to understand the full surrounding, e.g.*, for augmented reality applications or autonomous systems. We extended existing 3DGS-SLAM methods to handle omnidirectional input by including closed-form gradients for mapping and camera pose estimation, utilizing an equirectangular projection model. To reduce memory footprint, a key frame removal procedure based on graph analysis is proposed, enabling the application to handle larger input sizes. For evaluation, we provide a dataset of controlled real-world and synthetic test scenes (indoor and outdoor), employing a custom developed virtual camera lens. An extensive evaluation shows that, for camera tracking, the proposed method achieves statistically significant lower ATE RMSE scores compared to a recent omnidirectional SLAM system, as well as other 3DGS-SLAM frameworks, while reaching a similar mapping performance.*

1. Introduction

Simultaneous Localization and Mapping (SLAM) is a long-standing challenge in robotics and computer vision, with applications in autonomous systems [1] and augmented reality [36]. Its objective is to create a 3D map of the environment while determining the sensor's position within it using different sensor modalities. Visual SLAM (V-SLAM) relies solely on cameras, offering advantages such as portability, low cost, and passive sensing [31, 46, 54]. A key challenge is how to represent the map efficiently for tracking while maintaining dense and detailed information of the surroundings. Recent neural implicit representations like NeRFs [30] and 3D Gaussian Splatting (3DGS) [21] have enabled dense map representations, starting with iMAP [41] and evolving into 3DGS-based SLAM systems [15, 20, 29, 55, 58], improving mapping quality, tracking accuracy, and runtime performance.

However, these methods target perspective cameras with limited field-of-views (FoVs), which creates blind spots and limits tracking robustness. These limitations are addressed by omnidirectional sensing, improving environmental awareness for applications like obstacle avoidance [54] and scene understanding as, *e.g.*, in object-level SLAM [23]. While omnidirectional SLAM systems exist [16, 23, 54], to the best of our knowledge, none fully integrate omnidirectional vision with 3DGS.

We introduce ODGS-SLAM, the first omnidirectional

3DGS-based SLAM system (see an overview in Fig. 1). Built upon [29, 55] and integrating ODGS [22], the back-propagation pipeline is extended by analytical gradients w.r.t. the camera position for omnidirectional camera tracking, a weighting scheme in loss calculations to mitigate equirectangular distortion, and a novel key-frame removal strategy exploiting the rotational invariance of omnidirectional overlap to reduce memory consumption. We evaluate on a new dataset of real-world and synthetic scenes including realistic fisheye distortions in the rendered data. The evaluation shows superior tracking performance compared to other 3DGS-based and feature-based omnidirectional SLAM methods. Source code and dataset will be made available online (please contact authors for the link).

2. Related Work

SLAM has made great advancements since its first introduction in [37]. Classical approaches can be found in [3, 9] and more modern ones are summarized in [5, 19]; with RGB or RGBD images employed as input. V-SLAM approaches are, *e.g.*, provided in [28, 34, 44]. Newer approaches focus on deep learning, as recently surveyed, *e.g.*, in [31].

Neural SLAM uses a dense differentiable scene description based on NeRFs [30], where a position- and angle-dependent continuous color- and density-field is computed via a multi-layer perceptron. Since the first introduction [41], improvements have focused on hierarchical techniques [18, 62], neural implicit maps [24], volumetric fusion [56], and coordinate networks with global bundle adjustment [48]. RGB-only input was enabled in [63] based on [62]. A real-time optimizer was introduced in [60] with efficient loop closure and full bundle adjustment, supporting monocular, stereo, and RGBD setups. We also support RGB and RGBD and target near real-time performance, but use omnidirectional images and 3DGS for mapping.

3DGS SLAM represents scene geometry via 3D Gaussians [21], enabling fast rendering, scalability, and direct gradient-based optimization. The first methods introduced key innovations: memory-efficient key framing [29], coarse-to-fine pose refinement [55], and sub-map based scalability [58]. We base our work on [29] for its RGB/RGBD support and available implementation, while adopting the constant velocity assumption from [58]. Recent methods improve 3DGS SLAM by neural radiance priors to densify the map [26], by domain adaptions [49], by learned scene priors and a tri-plane neural field architecture to achieve real-time performance [13], or by multiple agents reconstructing globally from distributed cameras [57].

Omnidirectional SLAM addresses FoV limitations of perspective cameras by incorporating wide-angle or multi-camera images, providing advantages in complex navigation scenarios [23]. A modular multi-camera approach was presented in [47] reaching real-time performance through hierarchical data integration. OmniSLAM [54] is the most closely related work to ours, employing a wide-baseline multi-camera rig with four ultra-wide FoV cameras for 360° coverage, visual odometry [35] with dense depth maps from omnidirectional stereo matching [52], and loop closure for global consistency. But, it uses traditional volumetric fusion rather than 3DGS for dense mapping. Other approaches extend traditional SLAM frameworks to omnidirectional scenarios [50] or use stitched panoramic images [16].

Omnidirectional Neural Implicit Representations combine omnidirectional input with neural geometric representations. NeRFs are extended in [33] for omnidirectional outdoor urban images as captured for, *e.g.*, Google Street View, in [12] to work with distortions introduced by two fisheye lenses covering the full scene, and in [14] by introducing methods to address parallax distortion effects of panoramic images. 3DGS is adjusted for omnidirectional usage in [25] where Gaussians are directly projected from 3D space onto a sphere, in [2] where Gaussians are projected first onto a tangent plane and then wrapped around a sphere, and in [22] where a hybrid approach of the former is realized by projecting from tangent space onto the sphere in a second step. We follow the latter approach as the two step projection provides a better computational performance while still maintaining high accuracy.

Omnidirectional Image Stitching and Depth Estimation: Many omnidirectional methods rely on stitched panorama images and depth information [4]. Monocular approaches like [27, 64] estimate depth from single panoramic images, while stereo methods [40, 52, 53] use multiple cameras with overlapping FoV. Recent advances focus on real-time performance via optimized networks [8, 17], applicable in robotics and autonomous driving.

3. Method

3.1. 3D Gaussian Splatting

Following [29], our SLAM system employs 3DGS [21] to represent the scene as a set of N anisotropic 3D Gaussians $\mathcal{G}_i$, $i \in N$, defined by a covariance matrix $\mathbf{\Sigma}_i \in \mathbb{R}^{3\times3}$, its center $\boldsymbol{\mu}_i$, and opacity α_i. Instead of using spherical harmonics to represent view-dependent radiance, each Gaussian is assigned one single color $\mathbf{c}_i$, to prioritize speed over rendering quality. For rendering, the 3D Gaussians are projected onto the image plane, given the camera pose $\mathbf{T}_\mathrm{c}$ in world coordinates and the projection π. Images are rendered by reprojecting Gaussians to 2D, ordering them front to back. The color of a pixel $\mathbf{p}=(u,v)$ is determined as:

$$\mathbf{c}_\mathrm{p}(\mathbf{p}) = \sum_{i\in N} \mathbf{c}_i w_i(\mathbf{p}) \prod_{j=1}^{i-1} (1 - w_j(\mathbf{p})), \qquad (1)$$

with $w_i(\mathbf{p})$ determining the contribution of the i-th Gaussian to the pixel position $\mathbf{p}$. Similarly, the depth $d_\mathrm{p}(\mathbf{p})$ can

be synthesized, by replacing $\mathbf{c}_i$ in Eq. (1) with the distance d_i from the camera center to $\boldsymbol{\mu}_i$. Analogously, the silhouette $s_\mathrm{p}(\mathbf{p})$ can also be computed, by replacing $\mathbf{c}_i$ with 1. The latter denotes the coverage of each pixel with Gaussians $\mathcal{G}_i$. Now, a color image $I(\mathcal{G}, \mathbf{T}_\mathrm{c})$, a depth image $D(\mathcal{G}, \mathbf{T}_\mathrm{c})$, and a silhouette image $S(\mathcal{G}, \mathbf{T}_\mathrm{c})$ can be rendered for any camera position $\mathbf{T}_\mathrm{c}$ from the Gaussians $\mathcal{G}_i$. The rendering process is fully differentiable. The Gaussian parameters can be optimized to minimize the error between the rendered images and provided RGB/RGBD input frames via their derived gradients. As in [29], the extrinsic camera parameters for a respective frame are included in the optimization. This enables the usage of the same Gaussian map representation for mapping as well as for camera tracking; see Sec. 3.2.

Adaptive density control is run every $\mathrm{it}_\mathrm{adc} = 150$ iterations [21] and Gaussians with α_i below threshold $\epsilon_\alpha = 0.7$ are deleted. Note that in the further explanations below, index i of all Gaussian parameters is omitted for readability.

For more details see Supplementary (Sec. S1.1).

Omnidirectional Gaussian Splatting: To support 360° panorama input frames, the projection operation π as well as the calculation of $\mathbf{J}$ has to be adapted from the original 3DGS [21]. To this end, we extend ODGS [22] to support the optimization of the camera extrinsic parameters. The captures of an omni-directional camera system are represented in a spherical coordinate system $\mathbb{S}^2$. Thus, each ray refers to a location on a unit sphere with azimuth angle ϕ and elevation angle θ; see Fig. 1 (right). The normalization $\hat{\boldsymbol{\mu}} = \boldsymbol{\mu}/||\boldsymbol{\mu}||$ yields the projections onto the unit sphere with angular coordinates:

$$\phi_\mu = \sin^{-1}(\hat{\mu}_y), \quad \theta_\mu = \tan^{-1}(\hat{\mu}_x/\hat{\mu}_z). \tag{2}$$

The angles (ϕ, θ) are transformed, using image width W and height H, to yield $\boldsymbol{\mu}_\mathrm{o}$ in centered image space:

$$\boldsymbol{\mu}_\mathrm{o} = \pi_o(\hat{\boldsymbol{\mu}}) = \left(\frac{W}{2\pi}\phi_\mu + \frac{W}{2}, -H/\pi \cdot \theta_\mu + \frac{H}{2}\right)^T. \tag{3}$$

The 3D covariance matrix $\boldsymbol{\Sigma}$ is then projected onto the 2D image plane [22]. First, each $\boldsymbol{\Sigma}$ is projected onto a tangential plane of the unit sphere, using a perspective camera with unit focal length and principal ray from the sphere center, as well as the respective Gaussian $\boldsymbol{\mu}$. The transformation $\mathbf{T}_{\boldsymbol{\mu}}$ rotates the camera to point to $\boldsymbol{\mu}$ dependent on transformations given by ϕ_μ and θ_μ:

$$\mathbf{T}_{\boldsymbol{\mu}} = \mathbf{T}_{\theta_\mu} \times \mathbf{T}_{\phi_\mu} \tag{4}$$

$$= \begin{bmatrix} \cos\phi_\mu & 0 & -\sin\phi_\mu \\ \sin\theta_\mu \sin\phi_\mu & \cos\theta_\mu & \sin\theta_\mu \cos\phi_\mu \\ \cos\theta_\mu \sin\phi_\mu & -\sin\theta_\mu & \cos\theta_\mu \cos\phi_\mu \end{bmatrix}. \tag{5}$$

The Jacobian for perspective projection yields

$$\mathbf{J}_p = \begin{bmatrix} \frac{1}{||\boldsymbol{\mu}||} & 0 & 0 \\ 0 & \frac{1}{||\boldsymbol{\mu}||} & 0 \end{bmatrix}, \tag{6}$$

for aligned coordinates. Due to the adaptive density control of 3DGS [21], small covariance values of the Gaussians are assumed. Thus, the distortion introduced by projection from the tangential planes on the unit sphere is neglected. The resulting covariances on the tangent planes are directly mapped in the equirectangular image plane. To compensate for the distortion from the equirectangular projection a horizontal scaling factor of $\sec(\theta) = \frac{1}{\cos(\theta)}$ is applied, resulting in the distortion correction matrix $\mathbf{Q}_o$. Scaling $\mathbf{S}_o$ maps from angles to image pixels. The updated Jacobian is:

$$\mathbf{J}_o = \mathbf{S}_o \mathbf{Q}_o \mathbf{J}_p \mathbf{T}_{\boldsymbol{\mu}} \tag{7}$$

$$= \begin{bmatrix} w\frac{\cos\phi_\mu}{\cos\theta_\mu} & 0 & -w\frac{\sin\phi_\mu}{\cos\theta_\mu} \\ h\sin\theta_\mu \sin\phi_\mu & h\cos\theta_\mu & h\sin\theta_\mu \cos\phi_\mu \end{bmatrix},$$

$$\mathbf{Q}_o = \begin{bmatrix} \sec\theta_\mu & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{S}_o = \begin{bmatrix} \frac{W}{2\pi} & 0 \\ 0 & \frac{H}{\pi} \end{bmatrix}, \tag{8}$$

with $w = \frac{W}{2\pi||\boldsymbol{\mu}||}$ and $h = \frac{H}{\pi||\boldsymbol{\mu}||}$. Finally, Eq. (1) and the respective equations for depth and silhouette are used with $\boldsymbol{\mu}_\mathrm{o}$ and $\boldsymbol{\Sigma}'_{\mathrm{o,2D}}$ to create fully differentiable, equirectangular panorama images for color I_o, depth D_o, and silhouette S_o.

3.2. SLAM

Following Matsuki *et al*. [29], the SLAM-system is built around the 3D Gaussian representation and the differentiable renderer; see Sec. 3.1. It performs three main steps, tracking, key-framing, and mapping; see Fig. 1 (left). Here, these steps are described in more detail with focus on the proposed extensions for omni-directional input images.

Error Functions and Preprocessing: Given a set of N Gaussians $\mathcal{G}$ and the k-th input frame (either $\mathcal{F}^k_\mathrm{RGB} = \{\tilde{I}^k\}$ or $\mathcal{F}^k_\mathrm{RGBD} = \{\tilde{I}^k, \tilde{D}^k\}$) the parameters of the scene representation and the rendering pipeline can be optimized. Depending on the task, *i.e*., tracking or mapping, and given input RGB/RGBD, different error functions, preprocessing steps, masks, and weights are combined. To ensure that the Gaussians visually represent the input frame, a photometric residual between the rendered image $I(\mathcal{G}, \mathbf{T}^k_\mathrm{c})$, at camera pose $\mathbf{T}^k_\mathrm{c}$, and the input $\tilde{I}^k$ is defined:

$$E^k_\mathrm{pho} = \frac{1}{WH}\sum_{x=1}^{W}\sum_{y=1}^{H} f(x,y)\left\| I(\mathcal{G}, \mathbf{T}^k_\mathrm{c})_{x,y} - \tilde{I}^k_{x,y}\right\|_1, \tag{9}$$

with the pixel-wise weighting function $f(x, y)$; see below. In case of RGBD input, a geometric error is calculated by:

$$E^k_\mathrm{geo} = \frac{1}{WH}\sum_{x=1}^{W}\sum_{y=1}^{H} f(x,y)\left\| D(\mathcal{G}, \mathbf{T}^k_\mathrm{c})_{x,y} - \tilde{D}^k_{x,y}\right\|_1, \tag{10}$$

with $\bar{D}^k$ the ground truth depth. Further, to avoid too elongated Gaussians that can lead to tracking errors, an isotropic

regularization term is utilized. For each $\mathcal{G}_i$ it considers the axis-wise deviation of the scaling vector $\mathbf{s}_i$ (Eq. (S5)) from its mean $\bar{\mathbf{s}}_i$. With j as index for the three axes of the scaling vector, the isotropic regularization is calculated as:

$$E_{\text{iso}} = \frac{1}{3N} \sum_{i=1}^{N} \sum_{j=1}^{3} ||\mathbf{s}_{i,j} - \bar{\mathbf{s}}_i||_1. \tag{11}$$

Further, to compensate for varying lighting conditions, rendered images $I(\mathcal{G}, \mathbf{T}_{\text{c}}^k)$ are exposure-corrected by:

$$I_{\text{exp}}^k = e^{a_k} I(\mathcal{G}, \mathbf{T}_{\text{c}}^k) + b_k, \tag{12}$$

with optimizable correction parameters a_k and b_k, specific to every input frame. A pixel mask M_{int} can be applied to filter out all pixels for which the sum of the three color channels is smaller than a threshold $\lambda_{\text{rgb}} = 0.01$. To focus on areas with varying pixel intensity, *e.g.*, for tracking, a gradient mask M_{grad} can be applied. This mask is valid for all gradients that are larger than the median gradient of the whole image, scaled by an edge threshold $\lambda_{\text{edge}} = 1.1$. Gradients are calculated from the grayscale image utilizing a Scharr filter. In addition, for RGBD frames, a depth mask M_{depth} can be setup to filter all pixels with depth $d(x, y)$ outside of a valid range $d_{\text{min}} < d(x, y) \leq d_{\text{max}}$, with $d_{\text{min}} = 0.01$ and $d_{\text{max}} = 100$. This helps with filtering objects on the horizon.

Equirectangular images significantly stretch regions at the poles. Thus, a latitude weight scaling the pixel errors based on their distance from the poles [42] is applied to both E_{pho}^k and E_{geo}^k. It is calculated for pixel (x, y) via:

$$w_{\text{lat}}(y) = \cos((y/H - 1/2)\pi). \tag{13}$$

The rendered silhouette images $\mathcal{S}(\mathcal{G}, \mathbf{T}_{\text{c}}^k)$ can be used as pixel-wise weighting, to ensure pixels contribute to the error depending on their Gaussian coverage. Also, it can be employed to create a silhouette mask M_{sil}, filtering pixels with $s_{\text{p}}(x, y) \leq s_{\text{min}}$, with minimum opacity $s_{\text{min}} = 0.95$.

Differentiable Camera Pose Estimation: To estimate the camera position $\mathbf{T}_{\text{c}}^k$ for a new frame $\mathcal{F}^k$, gradients of the camera pose in the rasterization process of 3DGS are calculated and back-propagated to minimize the loss in $\mathcal{L}_{\text{track}}$; see Eqs. (17) and (18). They can be derived analytically, following GS-SLAM [55]. Using the chain rule, the gradients are calculated independently for the photometric and geometric errors E_{pho}^k and E_{geo}^k, respectively. Dropping frame index k and using previously defined color and blending factor, the gradient w.r.t. $\mathbf{T}_{\text{c}}^k$ for one pixel is obtained for E_{pho}^k as:

$$\begin{aligned} \frac{\partial E_{\text{pho}}}{\partial \mathbf{T}_{\text{c}}} &= \frac{\partial E_{\text{pho}}}{\partial \mathbf{c}_{\text{p}}} \frac{\partial \mathbf{c}_{\text{p}}}{\partial \mathbf{T}_{\text{c}}} \\ &= \frac{\partial E_{\text{pho}}}{\partial \mathbf{c}_{\text{p}}} \sum_{i \in N} \left(\frac{\partial \mathbf{c}_{\text{p}}}{\partial \mathbf{c}_i} \frac{\partial \mathbf{c}_i}{\partial \mathbf{T}_{\text{c}}} + \frac{\partial \mathbf{c}_{\text{p}}}{\partial w_i} \frac{\partial w_i}{\partial \mathbf{T}_{\text{c}}} \right). \end{aligned} \tag{14}$$

Here, the term $\frac{\partial \mathbf{c}_{\text{p}}}{\partial \mathbf{c}_i} \frac{\partial \mathbf{c}_i}{\partial \mathbf{T}_{\text{c}}}$ corresponds to view-dependent colors. As specular highlights are not intrinsic to the object's surface, this term is neglected for faster computation. Splitting remaining terms further, yields derivatives with respect to the Gaussian means $\boldsymbol{\mu}'_{\text{o}i}$ and covariances $\boldsymbol{\Sigma}'_{\text{o,2D}i}$:

$$\begin{aligned} \frac{\partial E_{\text{pho}}}{\partial \mathbf{c}_{\text{p}}} \left(\frac{\partial \mathbf{c}_{\text{p}}}{\partial w_i} \frac{\partial w_i}{\partial \mathbf{T}_{\text{c}}} \right) = & \frac{\partial E_{\text{pho}}}{\partial \mathbf{c}_{\text{p}}} \frac{\partial \mathbf{c}_{\text{p}}}{\partial w_i} \cdot \\ & \left(\frac{\partial w_i}{\partial \boldsymbol{\Sigma}'_{\text{o,2D}i}} \frac{\partial \boldsymbol{\Sigma}'_{\text{o,2D}i}}{\partial \mathbf{T}_{\text{c}}} + \frac{\partial w_i}{\partial \boldsymbol{\mu}'_{\text{o}i}} \frac{\partial \boldsymbol{\mu}'_{\text{o}i}}{\partial \mathbf{T}_{\text{c}}} \right). \end{aligned} \tag{15}$$

As shown by Yan *et al*. [55], $\frac{\partial \boldsymbol{\mu}'_{\text{o}i}}{\partial \mathbf{T}_{\text{c}}}$ is the deterministic component for the calculation of $\mathbf{T}_{\text{c}}$. Following them, we ignore the covariance-term for efficiency. Finally, $\mathbf{T}_{\text{c}}$ is split up in rotation $\mathbf{R}_{\text{c}}$ and translation $\mathbf{t}_{\text{c}}$. The respective gradients are derived from $\frac{\partial \boldsymbol{\mu}'_{\text{o}i}}{\partial \mathbf{T}_{\text{c}}}$; see Supplementary (Sec. S1.2) for details.

The gradient of E_{geo}^k with respect to $\mathbf{T}_{\text{c}}^k$ for the depth value of one pixel is derived similarly:

$$\begin{aligned} \frac{\partial E_{\text{geo}}}{\partial \mathbf{T}_{\text{c}}} &= \frac{\partial E_{\text{geo}}}{\partial d_{\text{p}}} \frac{\partial d_{\text{p}}}{\partial \mathbf{T}_{\text{c}}} \\ &= \frac{\partial E_{\text{geo}}}{\partial d_{\text{p}}} \sum_{i \in N} \left(\frac{\partial d_{\text{p}}}{\partial d_i} \frac{\partial d_i}{\partial \mathbf{T}_{\text{c}}} + \frac{\partial d_{\text{p}}}{\partial w_i} \frac{\partial w_i}{\partial \mathbf{T}_{\text{c}}} \right). \end{aligned} \tag{16}$$

Finally, the derivative with respect to w_i is also very similar to the color term in Eq. (15). Again, the gradient is split into a rotation and translation term; see Supplementary (Sec. S1.2).

Initialization: At the start, the system creates Gaussians by random sampling of $\lfloor WH/k_{\text{init}} \rfloor$ pixels in the equirectangular input image, with $k_{\text{init}} = 32$. If available, depth data is used to determine the distance of the Gaussians from the camera. Otherwise, they are placed on the unit sphere with a small random offset within ± 0.025. The parameters of the initial Gaussians are optimized for $n_{\text{init}} = 1050$ iterations to minimize the mapping loss $\mathcal{L}_{\text{map}}$, see Eqs. (20) and (21), only for the initial input frame without exposure correction. Then, the next frame is loaded to start the tracking phase.

Tracking: In the tracking module, new input frames are loaded and the camera pose is estimated. The initial pose $\mathbf{T}_{\text{c init}}^k$ for frame k is calculated using a constant motion assumption. A weighted average of the last three frames' relative motions (with weights $[0.5, 0.3, 0.2]$) is added to the pose of frame $k-1$. Then, the new position is optimized via the 3DGS rendering pipeline by repeatedly minimizing the tracking losses with respect to $\mathbf{T}_{\text{c}}^k$:

$$\mathcal{L}_{\text{track}}(\mathcal{F}_{\text{RGB}}^k) = E_{\text{pho}}^k. \tag{17}$$

For each frame, this optimization is run for $\text{it}_{\text{track}} = 50$ iterations or until the relative pose change is below a threshold $\epsilon_{\text{pose-conv}} = 10^{-5}$. Further, $I(\mathcal{G}, \mathbf{T}_{\text{c}}^k)$ is exposure corrected

via Eq. (12). The affine brightness parameters a_k and b_k are optimized during tracking, too. In addition, M_{int} and M_{grad} are computed from $\tilde{I}^k$ and used to filter I^k_{exp} and $\tilde{I}^k$. Finally, the pixel-wise weighting $f(x,y) = w_{\text{lat}}(y)s_{\text{p}}(x,y)$ is applied (see Eqs. (1), (9) and (13)).

For RGBD input, the depth error is included as well and the two terms are mixed using $\lambda_{\text{pho}} = 0.95$ as follows:

$$\mathcal{L}_{\text{track}}(\mathcal{F}^k_{\text{RGBD}}) = \left(\lambda_{\text{pho}} E^k_{\text{pho}} + (1-\lambda_{\text{pho}}) E^k_{\text{geo}}\right). \quad (18)$$

Depth images $D(\mathcal{G}, \mathbf{T}^k_{\text{c}})$ and $\tilde{D}^k$ are weighted with $f(x,y) = w_{\text{lat}}(y)$. Masks M_{depth} and M_{sil} are applied to filter very close, far away, or little contributing pixels.

Key Framing: Using every input frame for the optimization of the Gaussian map is not feasible due to limited memory. Thus, it is essential to select a set of active key frames $\mathcal{W}_{\text{k}}$ based on inter-frame co-visibility. First, we apply strategies for initial key frame selection, insertion, and removal from the active window similar to [29]. A new frame is marked if the overlap of observed Gaussians in both frames is large; *i.e.*, co-visibility measure $\tau_{\text{cov}} > 0.9$. When adding a new key frame, random Gaussians are initialized following the same approach as outlined above. For RGBD input the depth of Gaussians is set to the rendered depth at the current camera position perturbed by a small random factor. For RGB they are placed on the unit sphere. Then, an initial map refinement is run for $\text{it}_{\text{kf}} = 10$ iterations.

In addition, ODGS-SLAM introduces a novel key frame removal strategy that leverages properties of omnidirectional imaging. Panoramic images captured from similar positions with different orientations largely contain the same scene content. Therefore, spatially close key frames can be dropped. They are detected employing a graph-based approach. All key frames except the ones in the active tracking window $\mathcal{W}_k$ are filtered using a KD-tree [11] to identify pairs that are at most τ_{dist} apart. The resulting pairs are then evaluated for rotation-invariant image similarity through a spherical warping process. For two candidate key frames $\mathcal{F}^i$ and $\mathcal{F}^j$ with rotation matrices $\mathbf{R}_i$ and $\mathbf{R}_j$, a rotation-aligned similarity score $S(\tilde{I}^i, \tilde{I}^j)$ between the two input images $\tilde{I}^i$ and $\tilde{I}^j$ is estimated. First, the spherical coordinate directions $\mathbf{v}^i(\phi,\theta)$ are calculated for each pixel in $\tilde{I}^i$. Then, the relative rotation between the two key frames is applied:

$$\mathbf{v}'(\phi,\theta) = \left(\mathbf{R}_i \mathbf{R}_j^{-1}\right) \mathbf{v}^i(\phi,\theta). \quad (19)$$

The resulting directions are re-projected back to equirectangular coordinates. With the resulting map, image $\tilde{I}^i$ is then aligned with image $\tilde{I}^j$ via bilinear interpolation, producing $\tilde{I}^{i\prime}$. Finally, score $S(\tilde{I}^i, \tilde{I}^j)$ is calculated using a pixel-wise L_1 norm, between $\tilde{I}^{i\prime}$ and $\tilde{I}^j$.

All key frame pairs with $S(\tilde{I}^i, \tilde{I}^j) < \tau_{\text{sim}}$ are marked as redundant. Based on this an undirected graph is set up where nodes represent key frames and edges connect mutually redundant pairs. A greedy algorithm selects removable key frames, based on the score $R^k = 0.5 \cdot A^k + C^k$. Here, the temporal age A^k of key frame k is measured by its index in the chronologically ordered list, and C^k is its connectivity within the redundancy graph; *i.e.*, number of redundant neighbors. Iterating through the sorted list, for each unvisited node, all its redundant neighbors are removed. This ensures that the final selection retains a subset of key frames with minimal redundancy, while prioritizing more recent and better-connected observations. An example of the resulting redundancy graph is presented in Fig. 2, showing the spatial distribution of key frames, their redundancy relationships, and the result of the pruning process.

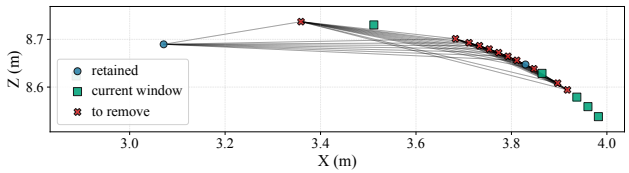

Figure 2. Key frame redundancy graph constructed from spatial proximity and rotation-aligned similarity: retained (blue), removed (red), and key frames belonging to the current tracking window (green). Edges illustrate mutual similarity.

Mapping: The mapping module creates and maintains an accurate 3D representation of the observed environment. In ODGS-SLAM, the Gaussian map of the union of recently observed key frames in the active window and two randomly selected key frames from the set of past key frames, $\mathcal{W} = \mathcal{W}_{\text{k}} \cup \mathcal{W}_{\text{r}}$, is optimized. The former ensures that the currently observed region is properly mapped, while the latter prevents the system from forgetting other parts of the map. In one mapping step, all the Gaussian parameters are optimized using the following loss over N_{f} frames in $\mathcal{W}$, with isometric mixing factor $\lambda_{\text{iso}} = 10$:

$$\mathcal{L}_{\text{map}}(\mathcal{F}^k_{\text{RGB}}) = \left(\frac{1}{N_{\text{f}}} \sum_{k=1}^{N_{\text{f}}} E^k_{\text{pho}}\right) + \lambda_{\text{iso}} E_{\text{iso}}. \quad (20)$$

Again, images $I(\mathcal{G}, \mathbf{T}^k_{\text{c}})$ are exposure corrected, masked with M_{int}, and weighted pixel wise via $f(x,y) = w_{\text{lat}}(y)$. Further, depth errors are included in the loss function if RGBD data is available:

$$\mathcal{L}_{\text{map}}(\mathcal{F}^k_{\text{RGBD}}) = \left(\frac{1}{N_{\text{f}}} \sum_{k=1}^{N_{\text{f}}} \left(\lambda_{\text{pho}} E^k_{\text{pho}} + (1-\lambda_{\text{pho}}) E^k_{\text{geo}}\right)\right) + \lambda_{\text{iso}} E_{\text{iso}}. \quad (21)$$

Here, depth inputs are weighted using w_{lat}. Further, very close as well as far away pixels are filtered using M_{depth}. The masking and weighting of depth inputs follows the tracking loss calculation, except for silhouette mask M_{sil}, which is omitted for mapping. Finally, adaptive density control is applied every it_{adc}-*th* iteration of the mapping.

4. Dataset

Although there are open datasets available in the context of SLAM [38, 39] and omnidirectional SLAM [6, 7, 10], these are either not omnidirectional or do not cover all required features for a thorough evaluation. RGB images, depth and positional data are required. Further, a comprehensive set of sequences, with explorations and varying complexity in movement and scene structure should be available; with sequences designed for testing and debugging. Moreover, identical sequences captured with omnidirectional as well as perspective cameras are needed, enabling direct comparisons between SLAM systems. For these reasons, we opted to create our own dataset, comprising real world and rendered scenes, for indoor and outdoor; see Tab. 1.

Table 1. Dataset overview; for resolution, f denotes fisheye, s stitched, o panorama, p perspective; sequences are exploration (Ex), isolated direction (ID), retracing (Rt), circular motion (C).

Scene	Camera	Resolution	Depth	Sequences
Indoor-Real	Insta360 Pro	$6\times1440\times1920f$	n	Ex, ID, Rt
		$1920\times960s$	y	
	Insta360 X4	$3840\times1920o$	n	Ex, ID, Rt
Indoor-Room	Synthetic	$6\times1440\times1920f$	y	Ex, ID, Rt, C
		$1920\times960s$	y	
		$3840\times1920o$	y	
		$1440\times1920p$	y	
Outdoor	Synthetic	$6\times1440\times1920f$	y	Ex
		$3840\times1920o$	y	
		$1440\times1920p$	y	

Indoor real world sequences were captured using two omnidirectional camera systems mounted on a remote-controlled robotic platform (*myAGV*). An *Insta360 Pro* and an *Insta360 X4* were chosen to cover professional and consumer hardware, comprising six and two fisheye cameras, respectively. For the *Insta360 X4* two mounting setups were employed; directly on the robot and with a vertical extension. Synthetic data was rendered in Blender via a custom developed lens extension to create realistic fisheye distortions based on the intrinsic parameters of the *Insta360 Pro*; calibrated using MC-Calib [32]. Two available Blender scenes were utilized: the *Italian Flat* [45] and the virtual urban [61] scene. Rendering resolutions were chosen to match the real cameras. Additionally, panoramic images and depth maps were generated, single fisheye camera outputs stitched, and pinhole cameras rendered. The camera motion sequences include longer *exploration paths* (Ex), as well as shorter evaluation sequences: *isolated directional movement* (ID), *retracing* (Rt), *i.e.*, traversing same trajectory twice with 180° turn in between, and *circular movement* (C). More details on the dataset can be found in the Supplementary (Sec. S3), together with example images (Figs. S3 to S6).

5. Evaluation

5.1. Setup

ODGS-SLAM was evaluated extensively on the generated dataset. In particular, its performance in tracking and mapping, as well as computational performance was compared with three other baseline methods for 20 different camera movement sequences in the four different scenes included in the dataset. Additionally, the contribution of selected system components was analyzed in an ablation study.

Tracking accuracy is measured using the root mean square error (RMSE) of the absolute trajectory error (ATE). Mapping quality is evaluated by comparing rendered key frames against corresponding ground truth views using three pixel-wise image error metrics: peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM) [51], and learned perceptual image patch similarity (LPIPS) [59]. Computational performance is assessed by measuring per-frame processing time for tracking and mapping, together with GPU and RAM memory utilization. Means of three to five runs per method are reported.

For the comparisons, firstly two 3DGS-SLAM systems for perspective cameras were selected: GS-SLAM (MonoGS) [55] and Gaussian-SLAM (GASL) [58]. This allows to isolate the effect of omnidirectional imagery and the extensions particular to ODGS-SLAM w.r.t. SLAM performance. Secondly, the feature-based omnidirectional SLAM method PatchMatch-Stereo-Panorama (PMSP) [43] was included to compare to a more traditional omnidirectional SLAM method employing ORB-features.

All experiments were conducted on workstations equipped with a NVIDIA RTX 4090 GPU, an Intel Core i7 9700K CPU, and at least 32 GB of RAM. Several hyper-parameters for ODGS-SLAM were already indicated above; the full set is listed in the Supplementary (Sec. S2). For all comparison methods, parameters were set to closely match the ODGS-SLAM ones, to ensure a fair comparison. In particular, for GASL the default configuration was applied. For MonoGS the Replica dataset parameters were used. These are the same in ODGS-SLAM; with the exception of the initial point cloud density, which was increased to create 2.67× more points, thus reflecting the larger amount of pixels present in omnidirectional input.

Input image size was set to 1920 × 960 pixels for omnidirectional and 720 × 960 pixels for perspective data; employing a FoV of ≈ 97°. The outdoor scene had to be run at half the input resolution, since no system was capable of handling the large amount of frames and scale of the map at full resolution. Furthermore, for the outdoor scene the tracking iteration and learning rate for the camera pose were increased, reflecting the larger scene scale. Note that depending on the input data, not all methods could be run on all settings; MonoGS and GASL both require perspective

Table 2. RMSE ATE (m) for RGB/RGBD synthetic sequences. Note: * is excluded from average calculation.

Method	Ex-O*	Ex1	Ex1s	Ex2	Rt	C	ID-XYZ	Avg
				RGB				
MonoGS	39.555	1.335	-	0.986	0.384	0.490	0.121	0.663
PMSP	**2.365**	0.081	0.069	**0.138**	0.099	0.074	0.095	0.093
O-SLM	36.070	**0.008**	**0.021**	0.355	**0.004**	**0.006**	**0.012**	**0.068**
				RGBD				
GASL	-	0.640	-	0.987	0.846	8.844	0.157	2.295
MonoGS	20.999	0.845	-	1.159	0.339	0.447	0.105	0.579
O-SLM	**0.571**	**0.013**	0.128	**0.025**	**0.002**	**0.001**	**0.002**	**0.029**

Table 3. RMSE ATE (m) for RGB/RGBD Insta360 Pro sequences.

Method	ID-Z	Rt	ID-X	Ex1	Ex2	Avg
			RGB			
MonoGS	0.028	0.439	0.109	0.582	0.394	0.310
PMSP	0.092	0.096	0.340	**0.038**	0.047	0.122
O-SLM	**0.020**	**0.036**	**0.026**	0.044	0.028	**0.031**
			RGBD			
O-SLM	0.020	0.037	0.129	0.052	0.038	0.055

images, GASL in addition needs RGBD data, and PMSP only works on RGB input.

5.2. Quantitative Evaluation

Tracking: Detailed tracking results (*i.e.*, RMSE ATE in meters) are reported in Tabs. 2 to 4, for all sequences in different scenes and setups in the created dataset (note that ODGS-SLAM is shortened in the tables to *O-SLM*, due to space). Further, the header rows indicate the respective sequences, and averages are shown in the last columns.

It can be observed that ODGS-SLAM manages to outperform other methods in almost all of the sequences. For statistical evaluation of the results, first Shapiro-Wilk tests were carried out; data largely did not follow a normal distribution, wherefore non-parametric tests were applied. Kruskal-Wallis H tests indicated that there is a significant difference ($p < .001$) between the methods for the cases RGB synthetic ($\chi^2(2) = 45.77$), RGB Insta360 Pro ($\chi^2(2) = 24.21$), and RGBD synthetic ($\chi^2(2) = 38.91$). Post-Hoc Dunn's tests, with Bonferroni corrected alpha of 0.017, indicated statistically significant improvement of ODGS-SLAM over the other methods. Similarly, a Mann-Whitney U test was performed to compare between ODGS-SLAM and PSMP for the Insta360 X4 sequences. The results indicated a significant difference, $U = 135$, $p < .000001$, with ODGS-SLAM having significantly lower RMSE compared to PSMP. Note that the outdoor scene (Ex-O) was excluded from statistical test since only ODGS-SLAM with RGBD and PMSP with RGB successfully track the full sequence.

Mapping: In the mapping evaluation, all 3DGS-based SLAM methods were compared. The mean results grouped by the different scenarios are listed in Tab. 5. Note that PMSP was excluded, since it only outputs a dense point cloud. The maps were evaluated at the end of each sequence, comparing all selected key frames with the resulting renders at the same positions. Overall, ODGS-SLAM rendering performance is comparable to the other methods.

Table 4. RMSE ATE (m) for Insta360 X4 sequences, separated for direct and extension mount.

Method	ID-Z	ID-XZ	Rt	Ex	Avg
		Direct mount			
PMSP	0.086	0.090	0.054	0.043	0.068
O-SLM	**0.042**	**0.008**	**0.037**	**0.042**	**0.032**
		Extension mount			
PMSP	0.176	0.071	0.067	0.276	0.148
O-SLM	**0.095**	**0.008**	**0.028**	**0.116**	**0.062**

Memory usage: Finally, for all sequences, the CPU and GPU memory utilized by the front- and backend, and the tracking and mapping times were measured. Mean values for the different scenarios are reported in Tab. 6.

Table 5. Mean rendering results across different scenarios. Arrows indicate if lower or higher values are better.

Scenario	Method	PSNR ↑	SSIM ↑	mSSIM ↑	LPIPS ↓	mLPIPS ↓
360 Pro RGB	MonoGS	22.50	**0.83**	**-**	**0.34**	-
	O-SLM	**24.35**	0.67	0.83	0.42	0.22
360 Pro RGBD	*O-SLM*	23.19	0.62	0.77	0.48	0.28
X4 (ext) RGB	*O-SLM*	23.98	0.75	0.82	0.32	0.19
X4 (dt) RGB	*O-SLM*	22.08	0.59	0.77	0.45	0.23
Synth. RGB	MonoGS	25.99	0.86	-	0.32	-
	O-SLM	**28.45**	**0.89**	-	**0.21**	-
Synth. RGBD	GASL	**32.36**	**0.96**	-	**0.09**	-
	MonoGS	27.65	0.87	-	0.27	-
	O-SLM	29.37	0.88	-	0.22	-

Ablation Study: To analyze the impact of the different modules on the overall system, an ablation study was performed. In particular, the constant movement, large depth masking, key frame (KF) removal, and latitude weighting were investigated, by disabling them one at a time. The first 750 frames of exploration one (Ex1) in the synthetic room scene was utilized as test sequence. The results shown in Tab. 7 indicate that without latitude weighting the system would be much more efficient, but the tracking performance is much worse without. Further, from the mean GPU memory utilization, it can be seen, that the key frame removal is very efficient without much runtime overhead.

Parameter Study: Finally, to analyze the influence of different parameters on the system performance a parame-

Table 6. Mean performance metrics across different scenarios, with frontend (FE) and backend (BE). Lower is better.

Scenario	Method	FE RAM (MB)	FE GPU (MB)	BE RAM (MB)	BE GPU (MB)	Track time (ms)	Map time (ms)
360 Pro RGB	MonoGS	1984.78	**4544.1**	**1542.97**	**4555.7**	299.9	**871.7**
	PMSP	-	-	-	-	**43.0**	-
	O-SLM	**1833.82**	5358.4	1596.62	5414.0	1085.6	3051.2
360 Pro RGBD	*O-SLM*	1930.82	7296.7	1776.71	7508.6	1356.5	2489.6
X4 (ext) RGB	PMSP	-	-	-	-	**58.9**	-
	O-SLM	1921.21	6910.0	1648.02	7025.8	1219.6	3131.9
X4 (dt) RGB	PMSP	-	-	-	-	**56.8**	-
	O-SLM	1943.45	7788.2	1689.93	7953.0	1243.7	2843.8
Synth. RGB	MonoGS	1985.73	**4428.9**	**1532.01**	**4440.8**	283.0	**825.7**
	PMSP	-	-	-	-	**43.8**	-
	O-SLM	**1910.49**	6685.4	1644.15	6757.8	1274.5	3126.8
Synth. RGBD	GASL	-	-	-	-	709.8	2911.6
	MonoGS	2487.49	**5815.6**	**1774.10**	**5835.4**	**505.4**	**821.2**
	O-SLM	**2030.25**	10315.0	1864.31	10666.3	1709.0	2752.7

Table 7. Results of the ablation runs.

Metric	Unit	Control	No const. mov.	No depth mask	No KF removal	No lat. weight
FE RAM	MB	2032.6	2018.1	2032.3	2318.4	**2006.3**
BE RAM	MB	1811.5	1814.2	1837.7	2143.4	**1781.8**
FE GPU	MB	7023.9	7398.5	8077.8	8072.9	**6036.8**
BE GPU	MB	7906.8	8216.6	8942.7	8857.2	**6676.8**
Track time	s	1.532	1.596	1.672	1.540	**1.412**
Map time	s	2.495	2.591	2.807	2.543	**2.260**
RMSE ATE	m	0.007	**0.006**	0.028	0.007	0.030
PSNR (↑)	db	27.84	27.48	26.55	**28.99**	28.33
SSIM (↑)	/	0.817	0.867	0.856	**0.885**	0.881
LPIPS (↓)	/	0.254	0.262	0.275	0.236	**0.223**

ter study was setup. As key parameters, varying image size, point cloud sampling rate k_{run}, tracking iterations it_{track}, and densification/tracking-convergence thresholds τ_{dens} and $\epsilon_{\text{pose-conv}}$ were identified. The study was run in the synthetic room exploration sequence (Ex1). For RGBD input, reducing the image size and increasing k_{run}, τ_{dens}, and $\epsilon_{\text{pose-conv}}$ achieved FPS 0.967, tracking time $0.184\,\text{s}$, mapping time $0.902\,\text{s}$, RMSE ATE $0.032\,\text{m}$, PSNR 24.65, SSIM 0.809, and LPIPS 0.249. Using the same parameters with RGB input, resulted in FPS 1.925, tracking time $0.112\,\text{s}$, mapping time $0.833\,\text{s}$, RMSE ATE $0.029\,\text{m}$, PSNR 25.26, SSIM 0.837, and LPIPS 0.220. The full study results are reported in the Supplementary (Sec. S6).

6. Discussion and Conclusion

This work introduces ODGS-SLAM, the first omnidirectional SLAM method using 3DGS for tracking and mapping. It extends existing 3DGS-based SLAM approaches to panoramic RGB/RGBD input by integrating an omnidirectional 3DGS pipeline, closed-form gradients for camera pose optimization, lateral weighting to address equirectangular distortion, and a novel graph-based key frame removal strategy exploiting rotational invariance in omnidirectional imagery for memory footprint reduction.

The system is evaluated and compared to two other 3DGS based SLAM methods and an omnidirectional feature based SLAM using our newly generated dataset. The latter includes fisheye, perspective and panoramic image sequences, of real-world and rendered indoor and outdoor scenes. Additional results including trajectories, depth maps, and error visualizations are provided in the Supplementary.

Results show that ODGS-SLAM achieves statistically significant better ATE RMSE scores, compared to all other methods, while reaching comparable mapping performances. Nevertheless, the main limitation of ODGS-SLAM is its tracking and mapping runtime, with overall tracking times of over one second. This likely results from the larger input resolution and the additional processing overhead due to the equirectangular imaging. Nevertheless, the parameter study shows that ODGS-SLAM can reduce runtime at the cost of higher RMSE ATE. Even with this trade-off, the RMSE ATE remains about an order of magnitude lower than that of other 3DGS-SLAM methods while delivering comparable rendering quality (see Tabs. 2 and 5). While longer computation times are a general concern with 3DGS based SLAM methods, this needs addressing in future work. Further, several methods faced difficulties with the outdoor scene. ODGS-SLAM RGB fails due to scale-drift; MonoGS struggles with rotation estimation (RGBD) or fails completely (RGB) and GASL runs out of memory (see in Fig. S9). In future, we intent to explore the performance in outdoor environments.

One of our key contributions, the key frame removal, became necessary due to the large GPU memory requirements. This is especially critical for our omnidirectional approach, where larger maps are created, covering more of the surroundings, than the other 3DGS-SLAM methods.

We conclude, that ODGS-SLAM establishes a foundation for omnidirectional 3DGS-based SLAM, demonstrating the benefits of full 360° sensing for tracking accuracy in dense neural SLAM systems.

Acknowledgments

The authors acknowledge partial use of AI tools for code and scripts, primarily for evaluation of results. These tools were also employed for grammar/spelling corrections. Additionally, manual derivation of gradients was verified. Any output from AI systems was critically reviewed and validated.

References

[1] Basheer Al-Tawil, Thorsten Hempel, Ahmed Abdelrahman, and Ayoub Al-Hamadi. A review of visual SLAM for robotics: evolution, properties, and future applications. *Frontiers in Robotics and AI*, 11, 2024. 1

[2] Jiayang Bai, Letian Huang, Jie Guo, et al. 360-GS: Layout-guided Panoramic Gaussian Splatting For Indoor Roaming. In *IEEE International Conference on 3D Vision (3DV)*, pages 1042–1053, 2025. 2

[3] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006. 2

[4] Josep Bosch, Klemen Istenio, Nuno Gracias, et al. Omnidirectional Multicamera Video Stitching Using Depth Maps. *IEEE Journal of Oceanic Engineering*, 45(4):1337–1352, 2020. 2

[5] Cesar Cadena, Luca Carlone, Henry Carrillo, et al. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016. 2

[6] N. Carlevaris-Bianco, A. Ushani, and R. Eustice. University of Michigan North Campus Long-Term (NCLT) vision and LiDAR dataset. https://robots.engin.umich.edu/nclt/, 2016. 6

[7] D. Caruso, J. Engel, and D. Cremers. Omni-LSD-SLAM dataset: Omnidirectional monocular SLAM sequences with ground truth. https://cvg.cit.tum.de/data/datasets/omni-lsdslam, 2015. 6

[8] Jiaxi Deng, Yushen Wang, Haitao Meng, et al. OmniStereo: Real-time Omnidirectional Depth Estimation with Multiview Fisheye Cameras. In *CVPR*, pages 1003–1012, 2025. 2

[9] Hugh F. Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006. 2

[10] Giulio Fontana, Matteo Matteucci, and Domenico G. Sorrenti. *Rawseeds: Building a Benchmarking Toolkit for Autonomous Robotics*, pages 55–68. Springer, 2013. 6

[11] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977. 5

[12] Kai Gu, Thomas Maugey, Sebastian Knorr, et al. OmniNeRF: Neural Radiance Field from 360° Image Captures. In *ICME*, pages 1–6, 2022. 2

[13] Zhen Hong, Bowen Wang, Haoran Duan, et al. SP-SLAM: Neural Real-Time Dense SLAM With Scene Priors. *preprint arXiv:2501.06469*, 2025. 2

[14] Ching-Yu Hsu, Cheng Sun, and Hwann-Tzong Chen. Moving in a 360 World: Synthesizing Panoramic Parallaxes from a Single Panorama. *preprint arXiv:2106.10859*, 2021. 2

[15] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-SLAM: Real-Time Simultaneous Localization and Photorealistic Mapping for Monocular, Stereo, and RGB-D Cameras. In *CVPR*, pages 21584–21593, 2024. 1

[16] Shunping Ji, Zijie Qin, Jie Shan, et al. Panoramic SLAM from a multiple fisheye camera rig. *ISPRS Journal of Photogrammetry and Remote Sensing*, 159:169–183, 2020. 1, 2

[17] Hao Jiang, Rui Xu, Mingdao Tan, et al. RomniStereo: Recurrent Omnidirectional Stereo Matching. *IEEE Robotics and Automation Letters*, 9(3):2458–2465, 2024. 2

[18] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. ESLAM: Efficient Dense SLAM System Based on Hybrid Representation of Signed Distance Fields. *CVPR*, pages 17408–17419, 2023. 2

[19] Iman Abaspur Kazerouni, Luke Fitzgerald, Gerard Dooly, et al. A survey of state-of-the-art on visual SLAM. *Expert Syst. Appl.*, 205:117734, 2022. 2

[20] 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*, pages 21357–21366, 2024. 1

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG*, 42(4):1–14, 2023. 1, 2, 3

[22] Suyoung Lee, Jaeyoung Chung, Jaeyoo Huh, and Kyoung Mu Lee. ODGS: 3D Scene Reconstruction from Omnidirectional Images with 3D Gaussian Splattings. In *NeurIPS*, pages 57050–57075, 2024. 1, 2, 3

[23] Feng Li, Rui Wang, and Cheng Bai. MCOO-SLAM: Multi-Camera Omnidirectional Object-level SLAM for Robust Localization. *arXiv preprint arXiv:2506.15402*, 2025. 1, 2

[24] Heng Li, Xiaodong Gu, Weihao Yuan, et al. Dense RGB SLAM with Neural Implicit Maps. In *ICLR*, 2023. 2

[25] Longwei Li, Huajian Huang, Sai-Kit Yeung, and Hui Cheng. OmniGS: Fast Radiance Field Reconstruction Using Omnidirectional Gaussian Splatting. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2260–2268, 2025. 2

[26] Mingrui Li, Shuhong Liu, Tianchen Deng, et al. DenseSplat: Densifying Gaussian Splatting SLAM with Neural Radiance Prior. *preprint arXiv:2502.09111*, 2025. 2

[27] Yuyan Li, Zhixin Yan, Ye Duan, et al. PanoDepth: A Two-Stage Approach for Monocular Omnidirectional Depth Estimation. *arXiv preprint arXiv:2202.01323*, 2022. 2

[28] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A Comprehensive Survey of Visual SLAM Algorithms. *Robotics*, 11(1), 2022. 2

[29] Hidenobu Matsuki et al. Gaussian Splatting SLAM. In *CVPR*, pages 18039–18048, 2024. 1, 2, 3, 5

[30] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2

[31] Saad Mokssit, Daniel Bonilla Licea, Bassma Guermah, and Mounir Ghogho. Deep Learning Techniques for Visual SLAM: A Survey. *IEEE Access*, 11:20026–20050, 2023. 1, 2

[32] François Rameau, Jinsun Park, Oleksandr Bailo, et al. MC-Calib: A generic and robust calibration toolbox for multi-camera systems. *Computer Vision and Image Understanding*, 217:103353, 2022. 6

[33] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, et al. Urban Radiance Fields. In *CVPR*, pages 12922–12932, 2022. 2

[34] Ali Rida Sahili, Saifeldin Hassan, Saber Muawiyah Sakhrieh, et al. A Survey of Visual SLAM Methods. *IEEE Access*, 11:139643–139677, 2023. 2

[35] Hochang Seok and Jongwoo Lim. ROVO: Robust Omnidirectional Visual Odometry for Wide-baseline Wide-FOV Camera Systems. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 6344–6350, 2019. 2

[36] Xingdong Sheng, Shijie Mao, Yichao Yan, and Xiaokang Yang. Review on SLAM algorithms for Augmented Reality. *Displays*, 84:102806, 2024. 1

[37] Randall C. Smith and Peter Cheeseman. On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986. 2

[38] Julian Straub, Thomas Whelan, Lingni Ma, et al. The Replica Dataset: A Digital Replica of Indoor Spaces. *preprint arXiv:1906.05797*, 2019. 6

[39] Jrgen Sturm, Nikolas Engelhard, Felix Endres, et al. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012. 6

[40] Xiaojie Su, Shimin Liu, and Rui Li. Omnidirectional Depth Estimation With Hierarchical Deep Network for Multi-Fisheye Navigation Systems. *IEEE Trans. on Intelligent Transportation Systems*, 24(12):13756–13767, 2023. 2

[41] Edgar Sucar, Shikun Liu, Joseph Ortiz, et al. iMAP: Implicit Mapping and Positioning in Real-Time. *ICCV*, pages 6209–6218, 2021. 1, 2

[42] Yule Sun, Ang Lu, and Lu Yu. Weighted-to-Spherically-Uniform Quality Evaluation for Omnidirectional Video. *IEEE Sign. Process. Letters*, pages 1408–1412, 2017. 4

[43] Hartmut Surmann, Marc Thurow, and Dominik Slomma. PatchMatch-Stereo-Panorama, a fast dense reconstruction from 360° video images. *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 366–372, 2022. 6

[44] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ Trans. Comput. Vis. Appl.*, 9:16, 2017. 2

[45] Flavio Della Tommasa. Italian Flat. https://www.malapixel.com/mal-house.html, accessed: 31.03.2025. 6

[46] Fabio Tosi, Youmin Zhang, Ziren Gong, Erik Sandström, Stefano Mattoccia, Martin R. Oswald, and Matteo Poggi. How NeRFs and 3D Gaussian Splatting are Reshaping SLAM: A Survey, 2024. 1

[47] Steffen Urban and Stefan Hinz. MultiCol-SLAM - A modular real-time multi-camera SLAM system. *CoRR*, abs/1610.07336, 2016. 2

[48] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-SLAM: Joint Coordinate and Sparse Parametric Encodings for Neural Real-Time SLAM. *CVPR*, pages 13293–13302, 2023. 2

[49] Kailing Wang, Chen Yang, Yuehao Wang, et al. EndoGSLAM: Real-Time Dense Reconstruction and Tracking in Endoscopic Surgeries using Gaussian Splatting. *preprint arXiv:2403.15124*, 2024. 2

[50] Senbo Wang, Jiguang Yue, Yanchao Dong, et al. Real-time Omnidirectional Visual SLAM with Semi-Dense Mapping. In *IEEE Intelligent Vehicles Symposium, IV, Changshu, Suzhou, China*, pages 695–700, 2018. 2

[51] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. 6

[52] Changhee Won, Jongbin Ryu, and Jongwoo Lim. OmniMVS: End-to-End Learning for Omnidirectional Stereo Matching. In *ICCV*. arXiv, 2019. 2

[53] Changhee Won, Jongbin Ryu, and Jongwoo Lim. End-to-End Learning for Omnidirectional Stereo Matching With Uncertainty Prior. *IEEE TPAMI*, 43(11):3850–3862, 2021. 2

[54] Changhee Won et al. OmniSLAM: Omnidirectional Localization and Dense Mapping for Wide-baseline Multi-camera Systems. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 559–566, 2020. 1, 2

[55] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. GS-SLAM: Dense Visual SLAM with 3D Gaussian Splatting. In *CVPR*, pages 19595–19604, 2024. 1, 2, 4, 6

[56] Xingrui Yang, Hai Li, Hongjia Zhai, et al. Vox-Fusion: Dense Tracking and Mapping with Voxel-based Neural Implicit Representation. *IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR)*, pages 499–507, 2022. 2

[57] Vladimir Yugay, Theo Gevers, and Martin R. Oswald. MAGiC-SLAM: Multi-Agent Gaussian Globally Consistent SLAM. *arXiv preprint arXiv:2411.16785*, 2024. 2

[58] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-SLAM: Photo-realistic Dense SLAM with Gaussian Splatting, 2024. 1, 2, 6

[59] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*, pages 586–595. IEEE, 2018. 6

[60] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, et al. GO-SLAM: Global Optimization for Consistent 3D Instant Reconstruction. *ICCV*, pages 3704–3714, 2023. 2

[61] Zichao Zhang et al. Benefit of large field-of-view cameras for visual odometry. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 801–808, 2016. 6

[62] Zihan Zhu, Songyou Peng, Viktor Larsson, et al. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In *CVPR*, pages 12776–12786, 2022. 2

[63] Zihan Zhu, Songyou Peng, Viktor Larsson, et al. NICER-SLAM: Neural Implicit Scene Encoding for RGB SLAM. *IEEE International Conference on 3D Vision (3DV)*, pages 42–52, 2024. 2

[64] Nikolaos Zioulis, Antonis Karakottas, Dimitrios Zarpalas, et al. OmniDepth: Dense Depth Estimation for Indoors Spherical Panoramas. In *ECCV*. arXiv, 2018. 2