IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Pittsburgh, USA, 2026.

MonoEM-GS: Monocular Expectation–Maximization Gaussian Splatting SLAM

Evgenii Kruzhkov[1] and Sven Behnke[1]

***Abstract*— Feed-forward geometric foundation models can infer dense point clouds and camera motion directly from RGB streams, providing priors for monocular SLAM. However, their predictions are often view-dependent and noisy: geometry can vary across viewpoints and under image transformations, and local metric properties may drift between frames. We present MonoEM-GS, a monocular mapping pipeline that integrates such geometric predictions into a global *Gaussian Splatting* representation while explicitly addressing these inconsistencies. MonoEM-GS couples Gaussian Splatting with an *Expectation–Maximization* formulation to stabilize geometry, and employs ICP-based alignment for monocular pose estimation. Beyond geometry, MonoEM-GS parameterizes Gaussians with multi-modal features, enabling in-place open-set segmentation and other downstream queries directly on the reconstructed map.**

We evaluate MonoEM-GS on 7-Scenes [1], TUM RGB-D [2] and Replica [3], and compare against recent baselines.

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) is a fundamental problem in robotics. Monocular SLAM is particularly attractive because it requires only a single RGB camera, avoiding additional sensing hardware. Traditional monocular SLAM systems (e.g., ORB-SLAM2 [4]) rely primarily on visual features and geometric optimization, including feature matching, pose estimation, local bundle adjustment, and loop closure. However, common limitations of traditional monocular SLAM are feature-poor regions and insufficient motion between frames.

Recently, feed-forward networks [5], [6], [7], [8], [9], [10] that directly infer dense point clouds and camera poses from the frames have attracted significant attention for visual SLAM. By extending the RGB input with additional geometric priors, new SLAM methods achieve dense monocular mapping in scenarios that are challenging for traditional pipelines. Moreover, some models are trained to produce metric–scale priors.

However, a key limitation is that the predicted geometry can be noisy and inconsistent: the noise may vary across camera views and can even change for the same image under simple transformations such as vertical flipping. In addition, local metric properties (e.g., relative scale or segment lengths) can drift between consecutive frames. These inconsistencies pose a challenge for current approaches [11], [12], [13], as they accumulate and degrade global consistency and smoothness of the reconstructed map.

[1]All authors are with the Autonomous Intelligent Systems group, Computer Science Institute VI – Intelligent Systems and Robotics – and the Center for Robotics and the Lamarr Institute for Machine Learning and Artificial Intelligence, University of Bonn, Germany; ekruzhkov@ais.uni-bonn.de

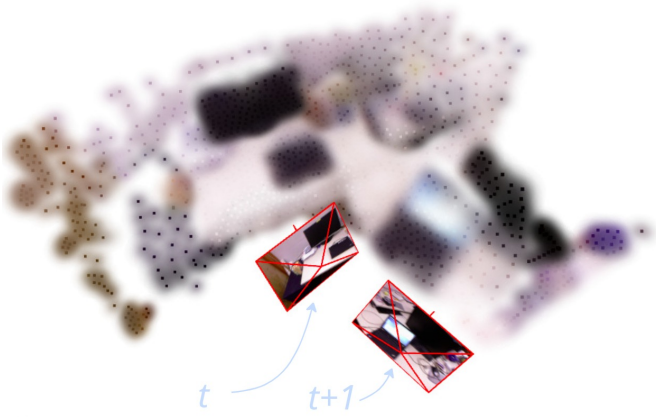

Fig. 1. MonoEM-GS constructs a map in which each Gaussian aggregates measurements over time. Despite contradictory point-cloud predictions at timestamps t and $t{+}1$, the Gaussians remain consistent, with only a single update step needed to incorporate new observations.

In this work, we propose MonoEM-GS, a monocular approach that constructs the map by maximizing the predicted measurements' likelihood. MonoEM-GS addresses the inconsistent predictions by merging them into the map in accordance with an incremental expectation–maximization (EM) pipeline and estimates the camera pose with respect to the current map [14], [15], [16]. We represent the map as a Gaussian Mixture Model (GMM) and employ Gaussian Splatting (GS) [17] to find challenging misalignments. To the best of our knowledge, MonoEM-GS is the first approach that studies the combination of EM and GS within a single monocular SLAM system.

Since geometric foundation models are integrated into the pipeline, they implicitly extract rich scene information. However, most SLAM systems do not explicitly attach this information to the resulting map, limiting what can be queried or computed directly from it. MonoEM-GS goes beyond pure geometry by parameterizing Gaussians with multi-modal features, enabling a variety of downstream applications to operate directly on the map. This is feasible because our Gaussian representation is sparser, allowing us to store high-dimensional features with no noticeable overhead. Figure 1 demonstrates the Gaussians that were incrementally created by MonoEM-GS.

We summarize our contributions as follows:

- We present **MonoEM-GS**, a monocular mapping system that reconstructs point clouds and meshes with improved geometric consistency and supports in-place open-set segmentation directly on the map.

- We show that **Gaussian Splatting** and **Expectation–Maximization** can be integrated within a single pipeline.
- We demonstrate that **ICP**-based alignment can be used to obtain monocular pose estimates.

II. Related Works

The closest line of work to ours is learned-geometry monocular SLAM, where the measurements are predicted from monocular RGB by a feed-forward geometry model. MASt3R-SLAM [13] builds monocular SLAM on two-view predicted reconstructions. VGGT-SLAM [11] extends this setting to an arbitrary number of frames and studies point-cloud alignment on the $\mathrm{SL}(4)$ manifold. VGGT2-SLAM [12] further extends VGGT-SLAM [11] and targets improved spatial alignment of internal submaps. ViSTA-SLAM [18] also regresses geometry from two views using symmetric associations; unlike [11], [12], it assigns one pose-graph node per view. VGGT-Long [19] targets large-scale monocular SLAM while avoiding graph-based bundle adjustment.

As demonstrated by the aforementioned works, predicted geometry can serve as measurements for a SLAM system. However, such predictions are often noisy and view–dependent. In contrast to prior works, we mitigate this by representing the map as a mixture model of parameterized Gaussians that are incrementally updated to increase the measurement likelihood. Overall, we target more geometrically consistent reconstructions by applying an EM-based [20] update to the predicted measurements and optimize the camera pose with respect to the map.

3D Gaussians in the form of Gaussian Splatting (GS) [17] have been successfully used for novel-view rendering within SLAM systems [21], [22], [23]. However, high reconstruction quality and low latency are typically achieved when additional geometric priors are available (e.g., depth sensors or LiDAR). HI-SLAM2 [24] and SING3R-SLAM [25] follow this direction; in particular, SING3R-SLAM [25] uses predicted geometry as a prior. In these methods, Gaussians are optimized primarily with novel-view synthesis objectives and updated via multiple per-frame optimization steps. Although GS demonstrates strong performance in photorealistic rendering, it doesn't generally represent a probabilistic scene model and may overfit to the views.

In contrast, we combine EM with GS and treat the map as a Gaussian mixture. Our incremental EM-based map update aggregates incoming measurements in a single step per frame, while GS is used for improved spatial data association. We don't optimize photorealistic rendering but employ rendering-based data association for the monocular predictions.

A wide range of models (VGGT [5], DUSt3R [6], MASt3R [7], MUSt3R [8], Pi3-X [9], CUT3R [10], DA3 [26], etc.) can generate geometric priors. In this work, we use the MapAnything [27] framework to estimate the observable geometry from consecutive RGB frames. We choose [27] because it is trained for metric-scale reconstruction and uses a DINO-based backbone, whose features are useful for downstream tasks. Moreover, it provides a unified interface, enabling other 3D reconstruction models to be swapped in interchangeably.

Works [11], [12], [13] produce dense point clouds, whereas our representation is sparse. As a result, we can additionally attach DINO [28], [29] features to each Gaussian with minimal memory overhead, enabling downstream tasks to operate directly on our reconstructions. Related multi-domain mapping has been demonstrated in OMCL [30], OVO [31], and RayFronts [32]. Unlike our approach, OMCL [30] performs open-vocabulary Monte Carlo localization using map features, while OVO [31] relies on poses estimated by an external localization method.

III. Method

We propose an approach for geometrically consistent monocular mapping, illustrated in Figure 2 for each new frame. Our method represents the scene as a set of 3D Gaussians, which serve both as a mixture-model [20] map and as the primitives for Gaussian splatting rendering [17]. The Gaussian parameters are accumulated across multiple consecutive observations predicted from the corresponding RGB views (Section III-A). We target monocular mapping with geometrically consistent reconstructions, reducing artifacts and shape ambiguities that can arise in SLAM pipelines that rely on feed-forward geometric predictions. We assume calibrated inputs (known intrinsics), since calibration is inexpensive and improves the geometric predictions.

The proposed method starts with map initialization (Section III-B). For each new frame, it then localizes by aligning the new prediction to the current map (Section III-C) and updates the map (Section III-D) by optimizing the Gaussian parameters to maximize the measurement likelihood.

A. Model Inference

We use the MapAnything [27] 3D reconstruction model to estimate the corresponding scene geometry from consecutive RGB frames, since it provides metric-scale reconstructions and a general-purpose feature backbone. From each inference, we extract the camera pose, the predicted point cloud, the associated per-point colors and DINOv3 [28], [29] features. We additionally compute per-frame surface normals for the point cloud. Let $\mathcal{O}_t^N = \{o_t^n\}_{n=1}^N$ denote the set of inferred observations at timestamp t, where N is the size of the set.

We found that the following heuristics improve the geometry predicted from RGB observations.

We run the model on a set of calibrated RGB images (known intrinsics) consisting of the latest image at time t; a fixed number of previous images stored in a FIFO buffer $\mathcal{S}$; and an anchor image. The anchor is the first image of the dataset with known initial pose.

We provide the model with only the anchor pose. We do not provide poses for the other images in the input set, even if those poses were estimated earlier. This avoids over-constraining the model and improves its geometry predictions.

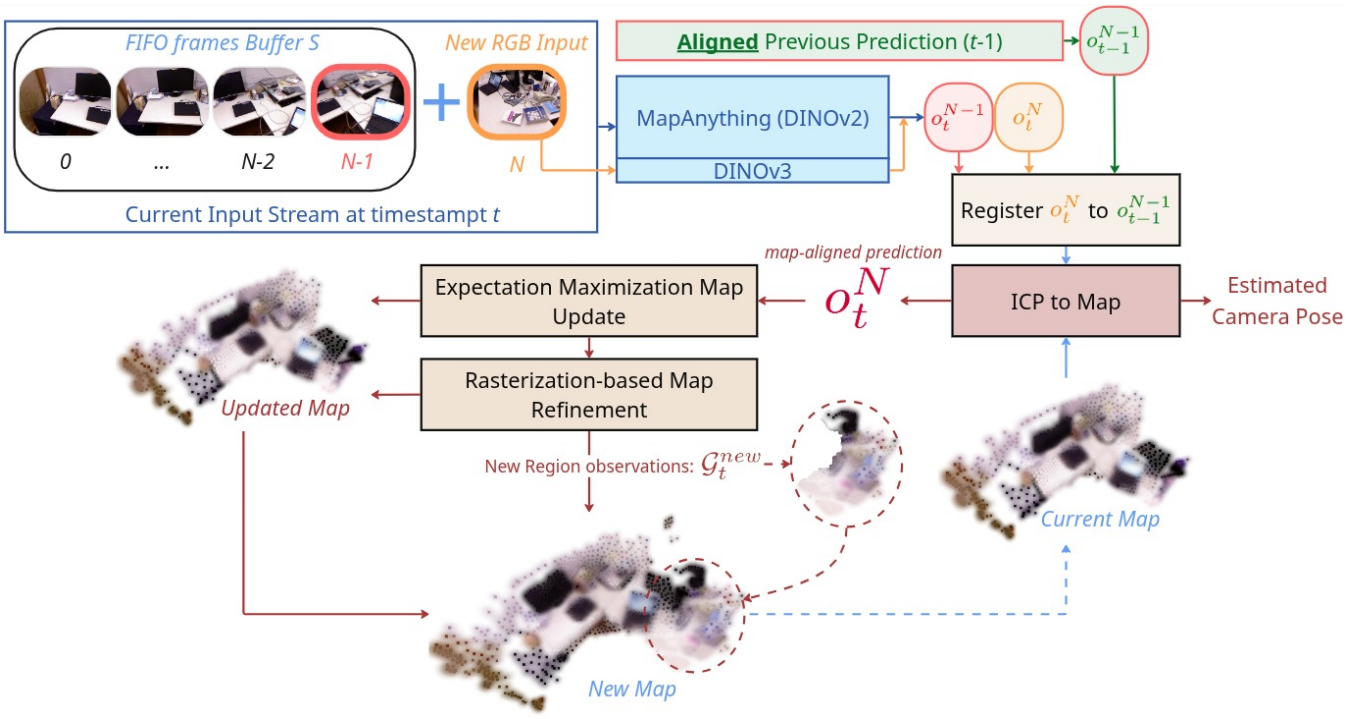

Fig. 2. Single iteration of processing a new input image by MonoEM-GS. For each new RGB input, MapAnything [27] produces a dense point cloud prediction o_t^N. We align it with the new prediction for frame N-1: o_t^{N-1} using ICP [15], [16]. Since o_t^{N-1} and o_{t-1}^{N-1} are created from the same frame at different timestamps, we employ the found correspondences between o_t^{N-1} and o_t^N to register o_t^N to o_{t-1}^{N-1}. After the initial registration has been found, we perform colored ICP [14] alignment of o_t^N to the map and estimate the camera pose with respect to the map. The aligned prediction o_t^N is fused into the map and appended to the FIFO buffer.

At each timestamp t, we integrate the previous frame, $t-1$, into the map, because its prediction improves when the input set includes the newer image at t. In other words, the newest image helps refine the geometry of the preceding one. Since the relative transformation between consecutive frames can be computed from the model prediction, we can temporarily estimate the pose of the frame t relative to frame $t-1$ without introducing a one-frame pose output delay.

For simplicity, without loss of generality, we re-index time by assigning timestamp t to the frame that is integrated into the map, and we refer to this frame as the "current frame".

MapAnything [27] currently provides pre-trained weights for the DINOv2 [28] model, while DINOv3 [29] offers more advanced capabilities for downstream applications. Since a naive encoder swap is suboptimal, we currently extract DINOv3 [28] features in parallel for the corresponding predictions. However, future releases of [27] may be based on DINOv3 [29].

B. Initialization

To initialize the map, we fill an observation buffer $\mathcal{S}$ using a fixed input stride, and then construct the initial map from the merged predictions. For the construction, we cluster [33] the predicted 3D points into K clusters based on their coordinates and use the cluster centroids as Gaussian means. The number of clusters K is set automatically by dividing the total number of predicted points by a user-defined constant λ and the number of inferred frames.

For each cluster k, we treat its centroid as the Gaussian mean $\boldsymbol{\mu}_k$ and estimate its covariance $\boldsymbol{\Sigma}_k$ from the coordinates of its M nearest neighbors points x_m. Each Gaussian is additionally parameterized by a color $\mathbf{c_k}$, a unit surface normal $\bar{\mathbf{n}}_k$, and a DINOv3 [29] feature $\mathbf{f}_k$, computed from the same neighbors using Mahalanobis distance kernel weights $w_{km} = \exp(-\frac{1}{2}\mathrm{d}_{\Sigma_k}(\mu_k, \mathrm{x}_m))$:

$$
\begin{aligned}
&\alpha_{km} = \frac{w_{km}}{\sum_{m=1}^{M} w_{km}}, \quad \bar{\mathbf{n}}_k = \frac{\sum_{m=1}^{M} \bar{n}_m \alpha_{km}}{\| \sum_{m=1}^{M} \bar{n}_m \alpha_{km} \|_2}, \\
&\mathbf{f}_k = f_{k^*}, \qquad k^* = \arg\max_m \frac{f_m \sum_{m=1}^{M} f_m \alpha_{km}}{\| \sum_{m=1}^{M} f_m \alpha_{km} \|_2}, \\
&\mathbf{c}_k = \sum_{m=1}^{M} \mathrm{C}_m \alpha_{km},
\end{aligned}
\tag{1}
$$

Our $\mathcal{M}$ consists of a set of parameterized Gaussians $\mathcal{M} = \{(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \mathbf{c_k}, \bar{\mathbf{n}}_k, \mathbf{f_k})\}_{k=1}^{K}$.

After initialization, the buffered frames are considered aligned with the map because it was directly constructed from their predictions. For DINOv3 [29] features, we assign to each Gaussian the feature closest to the average feature of the neighboring points. The features are fixed after initialization and not updated together with the rest of the parameters.

C. Localization

We build the input stream from the buffered set $\mathcal{S}$ and the current image, predict the observations $\mathcal{O}_t$ of size $|\mathcal{O}_t| = |\mathcal{S}| + 1$, and estimate the pose of the current frame with

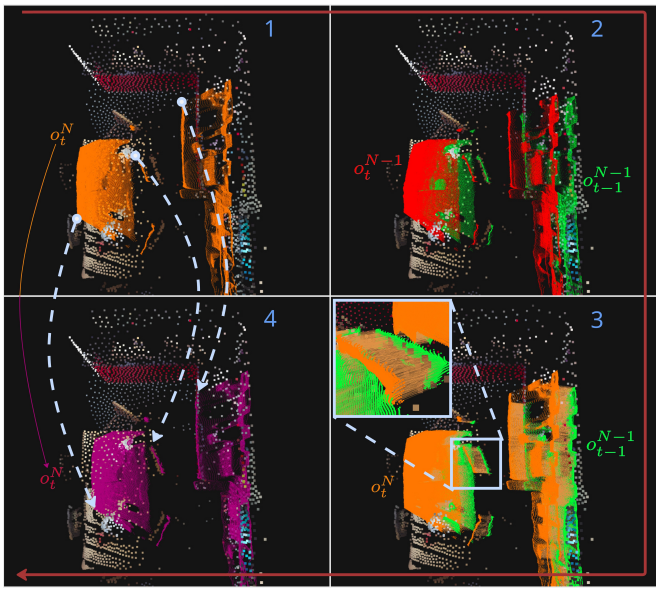

Fig. 3. Views of the current prediction alignment with the map. View 1: misaligned current prediction o_t^N; View 2: predictions o_t^{N-1} and o_{t-1}^{N-1} for the same buffered image at different timestamps; View 3: the found associations between o_t^N and o_{t-1}^{N-1}; View 4: current prediction o_t^N aligned to the map.

respect to the map $\mathcal{M}$ (following Section III-A, the captured frame is omitted).

The newly inferred prediction o_t^N can be arbitrarily misaligned with $\mathcal{M}$. For the buffered frames, however, we store their previous predictions, which are already aligned with the map (either because they were used during initialization or because they were aligned in earlier iterations). Importantly, the last buffered image always appears in two consecutive inference steps. Therefore, the predictions o_{t-1}^{N-1} and o_t^{N-1} are generated from the same input image and provide a one-to-one correspondence.

However, we require associations between predictions o_t^N and o_{t-1}^{N-1}. To obtain them, we first compute closest-point correspondences between o_t^N and o_t^{N-1} using ICP [15], [16] and thus derive the required point-wise associations between o_t^N and o_{t-1}^{N-1}. This yields a coarse registration of the current prediction o_t^N to the map $\mathcal{M}$.

Next, we select a map region (submap) $\mathcal{M}_t^*$ for the precise alignment. We employ the previously aligned point cloud of o_{t-1}^N to compute an approximate bounding box of the currently observable region. For efficiency on large maps, we query only the unique voxelized coordinates of Gaussians that fall inside this box, and then retrieve all Gaussians associated with the selected voxels. The retrieved Gaussians constitute the required submap.

The current frame pose is refined using colored ICP [14] without scale estimation between o_t^N and the submap. As a result, the predicted point cloud for the current timestamp t is tightly aligned with the map. We depict all mentioned predictions in Figure 3

D. Mapping

Expectation–Maximization Map Update. After aligning o_t^N to the map, we integrate it into $\mathcal{M}_t^*$ using an incremental expectation–maximization (EM-based) [20] update of the Gaussian mixture model [20], which increases the likelihood of the incoming measurements. For every point $\mathbf{x}_{o_t^N}$ in o_t^N, we find its K nearest Gaussians from $\mathcal{M}_t^*$. Assuming the aligned measurements and the map have similar local geometry, we fuse into the map only those points whose average normal distances $d_{\mathrm{n}}^{M\times K}$ and Mahalanobis distance $d_{\boldsymbol{\Sigma}_k}^{M\times K}(\boldsymbol{\mu}_k, \mathbf{x}_{o_t^N})$ satisfy thresholds τ_{n} and τ_Σ:

$$\begin{aligned} &d_{\mathrm{n}}^{M\times K} = \bar{\mathbf{n}}_k \bar{\mathrm{n}}_m, \qquad \frac{1}{K}\sum_k^K [d_{\mathrm{n}}^{M\times K}] \geq \tau_{\mathrm{n}} \\ &\min_k (d_{\boldsymbol{\Sigma}_k}^{M\times K}(\boldsymbol{\mu}_k, \mathbf{x}_{o_t^N})) \leq \tau_\Sigma \end{aligned} \tag{2}$$

where M is the total number of points in o_t^N. The weakly explained predicted points that don't satisfy τ_n and τ_Σ are omitted at this step and will be fused during "Rasterization-based Refinement".

For the rest of the points M^*, we compute Gaussian–measurement responsibilities $r^{M^*\times K}$ (normalized measurement–likelihood weight) over the neighbors based on cosine similarity $d_{\cos}^{M^*\times K}$ of the DINOv3 [29] features, local normal consistency $d_{\mathrm{n}}^{M^*\times K}$ and Mahalanobis distance $d_{\boldsymbol{\Sigma}_k}^{M^*\times K}$:

$$\begin{aligned} p_{det}^{ik} &= -\log|\boldsymbol{\Sigma}_k| \\ p^{ik} &= -\frac{1}{2}(d_{\boldsymbol{\Sigma}_k}^{ik})^2 + p_{det}^{ik} + \kappa_1(d_{\mathrm{n}}^{ik} - 1) + \kappa_2(d_{\cos}^{ik} - 1), \\ r^{ik} &= \frac{\exp p^{ik}}{\sum_{k=1}^K \exp p^{ik}}, \quad \pi_i = \max_k(p^{ik}) - \max_k(p_{det}^{ik}) \end{aligned} \tag{3}$$

for $i \in M^*, k \in K$, where k is the k-th nearest Gaussian for point i; κ_1 and κ_2 are constants. We integrate into the map only the observations with $\pi_i \leq \tau_\pi$, i.e. only the observation for which the best estimated explanation is at least τ_π close to the best Gaussian.

Using the responsibilities as the weights, we estimate new parameters for the Gaussians $\mathcal{M}_{t,new}^*$ from the current measurements o_t^N (online M-step) [20]. The final parameters are estimated [34] as a running average:

$$\mathcal{M}_t^* \leftarrow (1-\alpha_t)\mathcal{M}_t^* + \alpha_t \mathcal{M}_{t,new}^*, \; \alpha_t = \frac{\delta}{\alpha_{t-1} + \delta} \tag{4}$$

where $\alpha_0 = 1$ and $\delta \in (0, 1)$ is a per-Gaussian isotropy score that quantifies how symmetrically the assigned measurements support Gaussian k around its center in the plane orthogonal to the normal n_k. In practice, for each tangential principal axis of Gaussian k, we split the assigned points into the positive and negative half-spaces according to their projection onto that axis, and compute $\delta_k = 2\min(\sum_+ r^{ik}, \sum_- r^{ik})$. Thus, δ_k is large when both sides of the principal axis receive similar total responsibility and small when the support is concentrated on only one side. Note that the projection is used only to assign points

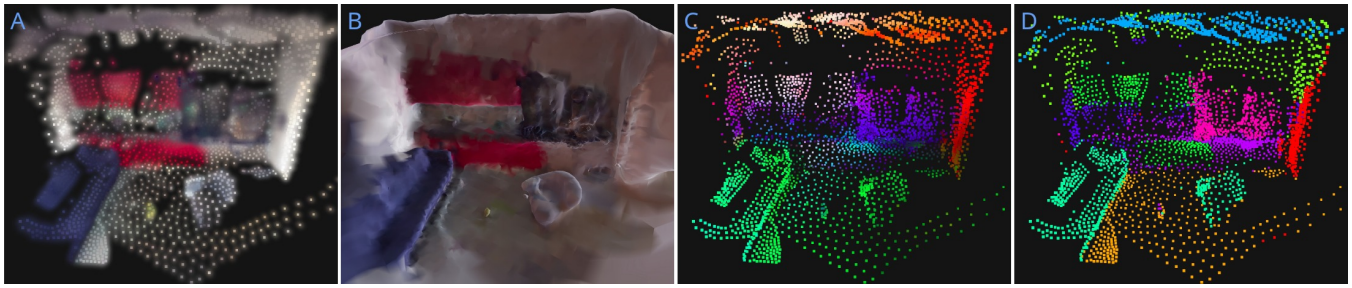

Fig. 4. Multi-domain scene reconstructions produced by MonoEM-GS. A: Gaussian Splatting created by the proposed approach; B: Mesh reconstructed from Gaussian centers and normals; C: DINOv3 [29] feature map; D: Open-set 3D semantic map.

to the positive and negative groups, and it uses the full responsibility values computed in Equation (3).

Rasterization-based Refinement. Many measurements could be omitted in the previous step either because they correspond to a new scene region or due to significant noise in the prediction. To determine possible missed correspondences, we construct new parameterized Gaussians $\mathcal{G}_t = \{(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g, \mathbf{c}_g, \bar{\mathbf{n}}_g, \mathbf{f}_g)\}$ from the predictions not fused by EM following Equation (1), and perform Gaussian splatting rendering [35] for both $\mathcal{G}_t$ and $\mathcal{M}_t^*$, setting constant opacities and using the refined camera pose at timestamp t and known intrinsics.

The rasterized $\mathcal{M}_t^{*R}$ produces expected depth, which we use to determine previously unmapped regions. A subset of $\mathcal{G}_t$ which was rasterized to pixels without valid depth is considered to be the observations of the new regions $\mathcal{G}_t^{new}$. For the rest of the rendering $\mathcal{G}_t^R$, we estimate associations with $\mathcal{M}_t^{*R}$, finding the $K_{2\text{D}}$ closest candidates on the image plane from $\mathcal{M}_t^{*R}$ for each $\mathcal{G}_t^R$. For each found correspondence, we estimate the 2D Mahalanobis distance on the image plane $d_{\Sigma_\mathcal{M}+\Sigma_\mathcal{G}}(\mathcal{M}_t^{*R}, \mathcal{G}_t^R)$ and cosine similarity $d_{cos}(\mathcal{M}_t^{*R}, \mathcal{G}_t^R)$.

We assume that correspondences with $d_{\Sigma_\mathcal{M}+\Sigma_\mathcal{G}} < \tau_{\Sigma 2\text{D}}$ and for which the rendered depth of the predictions is higher than that of the map belong to the same surface and can be merged. Those $\mathcal{G}_t^R$ which do not satisfy both $\tau_{\Sigma 2\text{D}}$ and the depth condition are considered new front observations and appended to $\mathcal{G}_t^{new}$.

The satisfied correspondences are merged with weights:

$$\omega = \frac{\exp(-\frac{1}{2} d_{\Sigma_\mathcal{M}+\Sigma_\mathcal{G}})}{\sum_{K_{2\text{D}}} \exp(-\frac{1}{2} d_{\Sigma_\mathcal{M}+\Sigma_\mathcal{G}})} d_{cos}(\mathcal{M}_t^{*R}, \mathcal{G}_t^R) \qquad (5)$$

where $\sum_{K_{2\text{D}}}$ is the sum over all $K_{2\text{D}}$ correspondences for each Gaussian in $\mathcal{G}_t^R$.

Next, we merge the components of $\mathcal{G}_t$ and $\mathcal{M}_t$. First, for each Gaussian in $\mathcal{M}_t$, we average the Gaussians of $\mathcal{G}_t$ associated with it with corresponding weights ω. The resulting averaged Gaussians $\bar{\mathcal{G}}_t$ have a one-to-one correspondence with $\mathcal{M}_t$ and, similarly to Equation (4), we update the map:

$$\mathcal{M}_t^* \leftarrow (1-\gamma_t)\mathcal{M}_t^* + \gamma_t \bar{\mathcal{G}}_t, \ \gamma_t = \frac{\max_\mathcal{G}(\omega)}{\alpha_t + \max_\mathcal{G}(\omega)} \qquad (6)$$

Observations which were not fused with the map in both of the previous steps, $\mathcal{G}_t^{new}$, are appended to the map, extending it. After the full localization and mapping step, the current frame with the aligned predictions is added to the FIFO buffer.

TABLE I
MONOEM-GS PARAMETERS.

Parameter	Value	Parameter	Value
Buffer size $\lvert\mathcal{S}\rvert$	10	λ	128
Normal distance τ_n	0.5	κ_1	5
τ_Σ	4.6	κ_2	5
Gaussians per point K	16	τ_π (in log-space)	8
Gaussians per point $K_{2\text{D}}$	3	$\tau_{\Sigma 2D}$	4.6

IV. EXPERIMENTS

Following MASt3R-SLAM [13] and VGGT-SLAM [11], we evaluate on the TUM RGB-D [2] and 7-Scenes [1] datasets. In addition, we evaluate the mapping on both datasets. For pose evaluation, we report the RMSE of the absolute trajectory error (ATE) and the relative scale error of the estimated trajectory, computed using evo [36] with alignment. For mapping quality, we report accuracy, completion, normal consistency, and F1-score at a 0.2 m threshold. To obtain ground-truth scene geometry, we fuse the provided ground-truth depth images using the ground-truth poses for each dataset.

Figure 4 illustrates the multiple out-of-the-box map representations supported by MonoEM-GS, highlighting its usability for a variety of robotics applications. Following the evaluation protocol of OMCL [30] and OVO [31], we report semantic segmentation performance on the Replica [3] dataset using the dataset-provided ground-truth (GT) scene.

We specify parameters used by MonoEM-GS in Table I. All baseline methods are evaluated with their default parameters and, when supported, with calibrated intrinsics. All results are obtained on an NVIDIA RTX 3090 GPU with 24 GB of memory. We process every tenth image on all datasets.

A. Pose Estimation and Reconstruction

For trajectory and reconstruction evaluation, we compare against similar RGB-based baselines and report the results in Table II. Since each Gaussian in our representation aggregates multiple observations, our map is naturally sparser. To ensure a fair comparison of completeness, we densify the representation by rendering additional points (for each pixel

TABLE II

MAPPING AND LOCALIZATION EVALUATION ON 7-SCENES [1] AND TUM RGB-D [2] DATASETS.

Methods	7-Scenes [1]						TUM RGB-D [2]					
	Meters [m]			Percentages [%]			Meters [m]			Percentages [%]		
	Acc. ↓	Comp. ↓	Trajectory / ATE RMSE ↓	F1 ↑	Normals ↑	Scale ↑	Acc. ↓	Comp. ↓	Trajectory / ATE RMSE ↓	F1 ↑	Normals ↑	Scale ↑
VGGT-SLAM [11]	**0.06**	0.27	0.07	70.7	34.0	47.2	0.15	0.27	0.05	63.2	46.2	74.8
VGGT-SLAM2 [12]	0.09	**0.11**	0.07	92.8	55.5	57.7	0.11	0.21	0.04	79.6	62.2	65.8
MASt3R-SLAM [13]	0.11	0.16	**0.05**	85.0	41.7	**88.4**	0.08	0.19	**0.03**	83.8	44.2	**86.1**
MapAnything [27]	0.13	0.14	0.1	84.7	35.4	71.9	0.15	0.24	0.13	65.9	38.9	75.4
MonoEM-GS (ours)	0.07	**0.10**	0.08	**93.2**	**59.8**	77.3	**0.06**	**0.15**	0.12	**86.8**	**65.4**	85.2

1st best 2nd best 3rd best

using the dataset resolution) from the estimated poses. For normal-consistency evaluation, we estimate normals for the baselines from their final point clouds, whereas our method estimates and stores normals during mapping, as described in Section III-B. Although MapAnything [27] is not a SLAM method, we include it in the evaluation because MonoEM-GS uses its predictions and MapAnything [27] can estimate camera poses from the input RGB sequence. For MapAnything [27], we process 25 images sampled at equal temporal stride in a single batch to obtain its predictions, whereas the other methods operate sequentially on the stream. We chose 25 images because this is the largest batch that fits in GPU memory on our hardware.

All evaluated approaches achieve similar performance, with only small differences between the top two methods. The proposed method is consistently ranked first or second across most metrics. It outperforms the MapAnything [27] results on all evaluated sequences and metrics. Moreover, MonoEM-GS and MASt3R-SLAM [13] maintain the most correct and consistent metric scale across all sequences.

7-Scenes [1] contains small indoor scenes, whereas TUM RGB-D [2] includes room-scale environments. Compared to optimization-based baselines, our pose estimates are less accurate in both environments; nevertheless, the final reconstruction-quality metrics remain comparable.

However, the proposed approach yields more geometrically consistent reconstructions (Figure 5). While the baselines recover many correct 3D points, they also accumulate erroneous predictions over time, which leads to ambiguous structures. In contrast, we achieve higher normal consistency, which is important for reliable mesh extraction.

Figure 5 compares point clouds and meshes produced by VGGT-SLAM2 [12] and MonoEM-GS. We chose VGGT-SLAM2 [12] because it achieves the second-best normal-consistency score in Table II and is designed for improved measurement alignment. For our method, we use the normals estimated and stored during mapping. For VGGT-SLAM2 [12], we estimate normals from its final map, with manually tuned parameters to obtain the best result.

As shown in Figure 5, VGGT-SLAM2 [12] produces a denser point cloud, but the geometry contains inconsistent and ambiguous shapes. In contrast, our map is sparser but exhibits clearer structure, with more unambiguous walls and surfaces. The mesh extracted from VGGT-SLAM2 [12] contains substantial noise, whereas ours is cleaner. Overall, our method achieves mapping-quality metrics comparable to the baselines while producing more consistent geometry. An additional demonstration of our reconstruction results is provided in Figure 6.

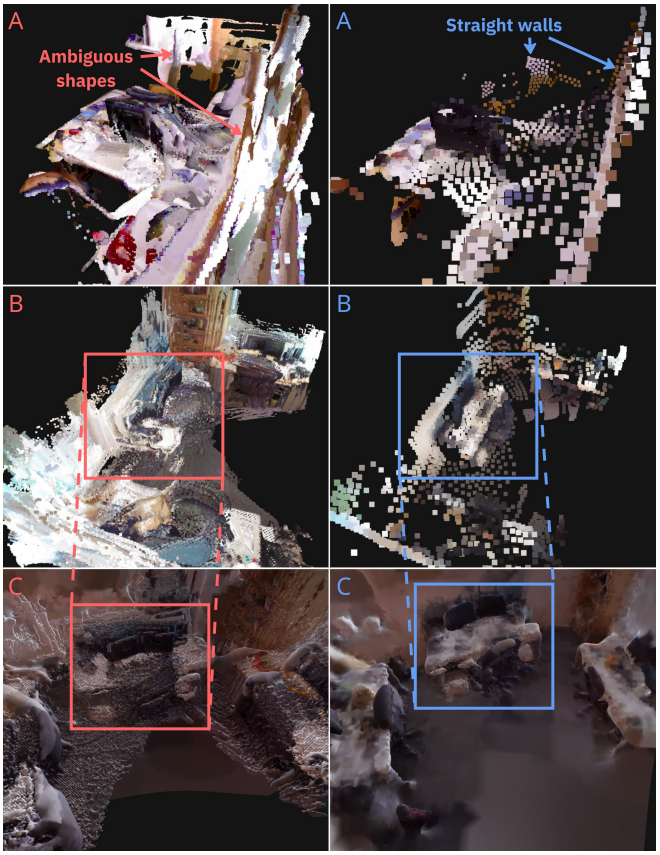

Fig. 5. Qualitative comparison of VGGT-SLAM2 [12] (left) and MonoEM-GS (right) maps. A–A: VGGT-SLAM2 [12] point cloud and MonoEM-GS Gaussian centers on the TUM RGB-D [2] "desk2" sequence; B–B: VGGT-SLAM2 [12] point cloud and MonoEM-GS Gaussian centers on the 7-Scenes [1] "office" sequence; C–C: meshes reconstructed for both methods from the point clouds in B and B, respectively.

B. Semantic Segmentation

We report average 3D semantic segmentation metrics on the photo-realistic Replica dataset [3], which contains room-scale environments. We directly use the DINOv3 [29]

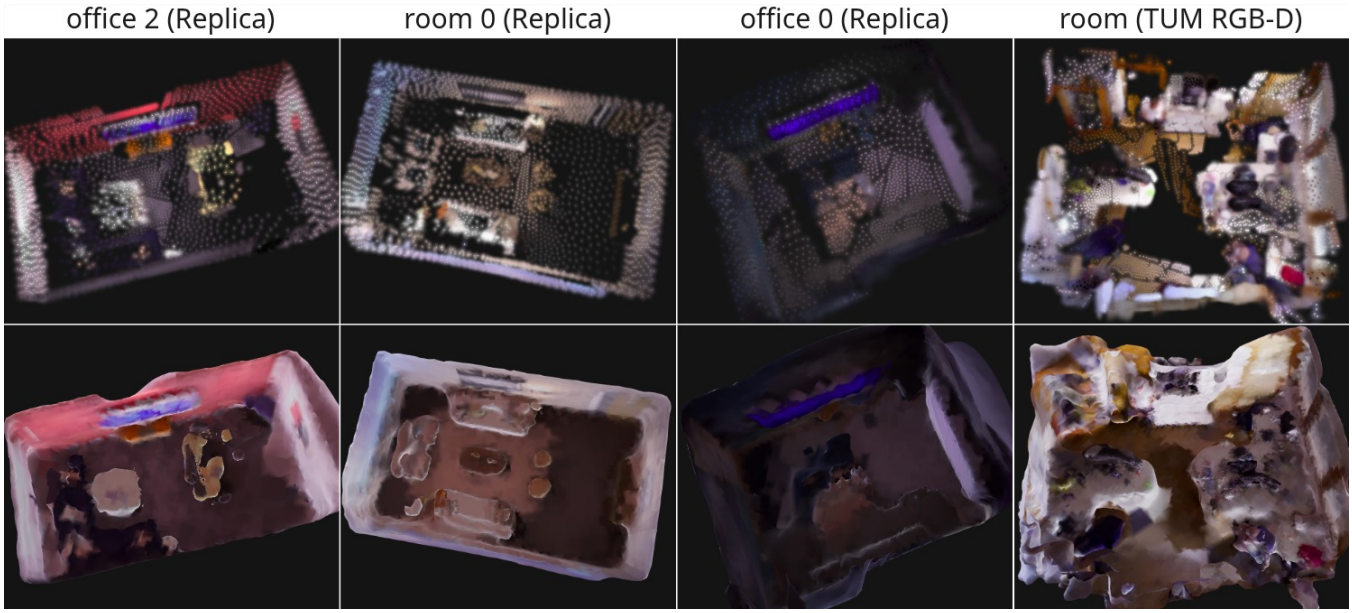

Fig. 6. Reconstructed Gaussians (upper row) and corresponding meshes (bottom row) created from the Gaussian centers and normals.

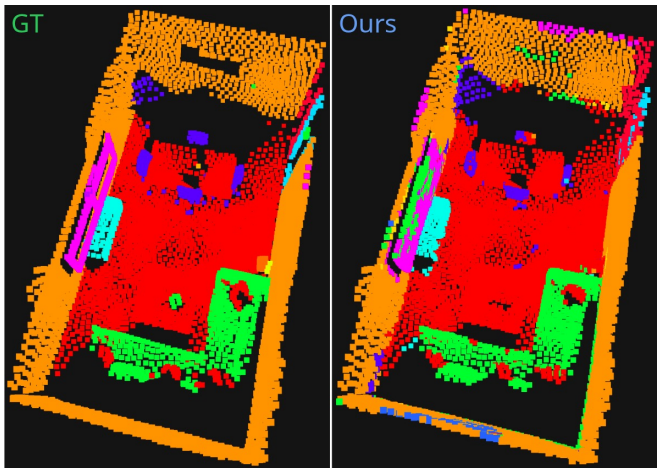

Fig. 7. MonoEM-GS Gaussian centers colored by their predicted classes (right) and the nearest ground-truth points colored by their GT classes (left).

TABLE III

REPLICA 3D [3] SEMANTIC SEGMENTATION EVALUATION.

Methods	**mIoU** [%]	**f-mIoU** [%]	**Acc** [%]	**Trajectory / ATE RMSE** [cm]
ConceptGraphs [37]	16.7	35.75	33.7	N/A
RayFronts [32]	27.7	43.47	54.5	N/A
OVO/ORB-SLAM2 [31]	27.1	60.2	39.1	**1.9**
OML [30]	**32.1**	49.6	**56.2**	35
MonoEM-GS (ours)	31.5	**63.2**	47.4	13.1

1st best 2nd best 3rd best

features stored in our map and classify them with the corresponding text encoder. For each scene, we encode the dataset-provided class names with the text encoder and assign each Gaussian the class whose text embedding has the highest cosine similarity to its DINOv3 [29] feature. Quantitative results are reported in Table III.

Note that our method estimates the trajectory as described in Section III, whereas other approaches either use ground-truth poses (N/A), rely on an independent localization system (OVO [31]/ORB-SLAM2 [4]), or decouple mapping from localization (OMCL [30]).

We achieve the best frequency-weighted mean IoU (f-mIoU) and the second-best mIoU. Although our mIoU is comparable to OMCL [30], our f-mIoU is substantially higher because our predictions are more accurate on frequent classes but weaker on rare classes. In contrast, OMCL [30] attains a similar mIoU with a lower f-mIoU, suggesting more balanced performance across rare and frequent classes, but less consistent accuracy on the dominant classes.

Accuracy measures the fraction of points that are classified correctly and is influenced by rare classes. OMCL [30] achieves the highest accuracy, while our method and the second-best baseline correctly classify approximately half of the points. Figure 7 visualizes our predicted class for each Gaussian center and the ground-truth class of the nearest GT point to that center.

Our trajectory error on Replica [1] (Table III) is consistent with that on TUM RGB-D [2] (Table II). Although OVO [31] achieves better localization accuracy, it uses RGB-D input and relies on an external localization method. In contrast, we outperform OMCL [30], which also performs localization using only monocular input.

V. CONCLUSIONS

In this paper, we introduced MonoEM-GS, a method for consistent monocular mapping. Overall, within a single monocular pipeline, MonoEM-GS reconstructs point clouds and meshes, stores multi-domain features, supports in-place open-set segmentation, and represents the scene as a set of 3D Gaussians whose centers and covariances define a

probabilistic spatial density. MonoEM-GS mitigates temporal measurement inconsistencies and increases the average measurement likelihood. Compared to closely related methods, MonoEM-GS produces clearer geometry and smoother meshes. However, MonoEM-GS is currently limited to small (room-scale) scenes due to the lack of loop closure. It may also integrate incorrect predicted measurements that lie significantly in front of the mapped surfaces.

ACKNOWLEDGMENT

This work has been partially funded by the Federal Ministry of Research, Technology and Space of Germany (BMFTR) under grant no. 01IS22094A WestAI and within the Robotics Institute Germany, grant no. 16ME0999.

REFERENCES

[1] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi, "Real-Time RGB-D Camera Relocalization," in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013, pp. 173–179.

[2] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[3] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, *et al.*, "The Replica Dataset: A Digital Replica of Indoor Spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[4] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[5] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "VGGT: Visual Geometry Grounded Transformer," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5294–5306.

[6] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "DUSt3R: Geometric 3D Vision Made Easy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 697–20 709.

[7] V. Leroy, Y. Cabon, and J. Revaud, "Grounding Image Matching in 3D with MASt3R," in *European Conference on Computer Vision*. Springer, 2024, pp. 71–91.

[8] Y. Cabon, L. Stoffl, L. Antsfeld, G. Csurka, B. Chidlovskii, J. Revaud, and V. Leroy, "MUSt3R: Multi-view Network for Stereo 3D Reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025, pp. 1050–1060.

[9] Y. Wang, J. Zhou, H. Zhu, W. Chang, Y. Zhou, Z. Li, J. Chen, J. Pang, C. Shen, and T. He, "Pi3: Permutation-Equivariant Visual Geometry Learning," *arXiv preprint arXiv:2507.13347*, 2025.

[10] Q. Wang, Y. Zhang, A. Holynski, A. A. Efros, and A. Kanazawa, "Continuous 3D Perception Model with Persistent State," *arXiv preprint arXiv:2501.12387*, 2025.

[11] D. Maggio, H. Lim, and L. Carlone, "VGGT-SLAM: Dense RGB SLAM Optimized on the SL(4) Manifold," *arXiv preprint arXiv:2505.12549*, 2025.

[12] D. Maggio and L. Carlone, "VGGT-SLAM 2.0: Real-Time Dense Feed-forward Scene Reconstruction," *arXiv preprint arXiv:2601.19887*, 2026.

[13] R. Murai, E. Dexheimer, and A. J. Davison, "MASt3R-SLAM: Real-time Dense SLAM with 3D Reconstruction Priors," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 16 695–16 705.

[14] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored Point Cloud Registration Revisited," in *Proceedings of the IEEE international Conference on Computer Vision*, 2017, pp. 143–152.

[15] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm," in *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*. IEEE, 2001, pp. 145–152.

[16] Y. Chen and G. Medioni, "Object Modelling by Registration of Multiple Range Images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.

[17] B. Kerbl, G. Kopanas, T. Leimkühler, G. Drettakis, *et al.*, "3D Gaussian Splatting for Real-time Radiance Field Rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[18] G. Zhang, S. Qian, X. Wang, and D. Cremers, "ViSTA-SLAM: Visual SLAM with Symmetric Two-view Association," *arXiv preprint arXiv:2509.01584*, 2025.

[19] K. Deng, Z. Ti, J. Xu, J. Yang, and J. Xie, "VGGT-Long: Chunk it, Loop it, Align it–Pushing VGGT's Limits on Kilometer-scale Long RGB Sequences," *arXiv preprint arXiv:2507.16443*, 2025.

[20] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intelligent Robotics and Autonomous Agents series. MIT Press, 2005. [Online]. Available: https://books.google.de/books?id=2Zn6AQAAQBAJ

[21] H. Matsuki, R. Murai, P. H. J. Kelly, and A. J. Davison, "Gaussian Splatting SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[22] E. Sandström, G. Zhang, K. Tateno, M. Oechsle, M. Niemeyer, Y. Zhang, M. Patel, L. Van Gool, M. Oswald, and F. Tombari, "Splat-SLAM: Globally Optimized RGB-only SLAM with 3D Gaussians," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 1680–1691.

[23] N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten, "SplaTAM: Splat Track & Map 3D Gaussians for Dense RGB-D SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 357–21 366.

[24] W. Zhang, Q. Cheng, D. Skuddis, N. Zeller, D. Cremers, and N. Haala, "Hi-SLAM2: Geometry-aware Gaussian SLAM for Fast Monocular Scene Reconstruction," *IEEE Transactions on Robotics*, vol. 41, pp. 6478–6493, 2025.

[25] K. Li, M. Niemeyer, S. Wang, S. Gasperini, N. Navab, and F. Tombari, "SING3R-SLAM: Submap-based Indoor Monocular Gaussian SLAM with 3D Reconstruction Priors," *arXiv preprint arXiv:2511.17207*, 2025.

[26] H. Lin, S. Chen, J. Liew, D. Y. Chen, Z. Li, G. Shi, J. Feng, and B. Kang, "Depth Anything 3: Recovering the Visual Space from Any Views," *arXiv preprint arXiv:2511.10647*, 2025.

[27] N. Keetha, N. Müller, J. Schönberger, L. Porzi, Y. Zhang, T. Fischer, A. Knapitsch, D. Zauss, E. Weber, N. Antunes, *et al.*, "MapAnything: Universal Feed-Forward Metric 3D Reconstruction," *arXiv preprint arXiv:2509.13414*, 2025.

[28] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, *et al.*, "DINOv2: Learning Robust Visual Features without Supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[29] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, *et al.*, "DINOv3," *arXiv preprint arXiv:2508.10104*, 2025.

[30] E. Kruzhkov, R. Memmesheimer, and S. Behnke, "OMCL: Open-vocabulary Monte Carlo Localization," *IEEE Robotics and Automation Letters*, vol. 11, no. 3, pp. 2698–2705, 2026.

[31] T. B. Martins, M. R. Oswald, and J. Civera, "Open-Vocabulary Online Semantic Mapping for SLAM," *IEEE Robotics and Automation Letters*, 2025.

[32] O. Alama, A. Bhattacharya, H. He, S. Kim, Y. Qiu, W. Wang, C. Ho, N. Keetha, and S. Scherer, "RayFronts: Open-set Semantic Ray Frontiers for Online Scene Understanding and Exploration," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 5930–5937.

[33] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The Faiss library," 2024.

[34] D. West, "Updating Mean and Variance Estimates: An Improved Method," *Communications of the ACM*, vol. 22, no. 9, pp. 532–535, 1979.

[35] V. Ye, R. Li, J. Kerr, M. Turkulainen, B. Yi, Z. Pan, O. Seiskari, J. Ye, J. Hu, M. Tancik, and A. Kanazawa, "gsplat: An Open-source Library for Gaussian Splatting," *Journal of Machine Learning Research*, vol. 26, no. 34, pp. 1–17, 2025.

[36] M. Grupp, "evo: Python package for the evaluation of odometry and SLAM." https://github.com/MichaelGrupp/evo, 2017.

[37] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, *et al.*, "ConceptGraphs: Open-vocabulary 3D Scene Graphs for Perception and Planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 5021–5028.