Gaussian Mapping for Evolving Scenes

Vladimir Yugay[1]*, Thies Kersten[1]*, Luca Carlone[3], Theo Gevers[1], Martin R. Oswald[1], Lukas Schmid[2,3]
[1]University of Amsterdam [2]University of Technology Nuremberg [3]Massachusetts Institute of Technology
vladimiryugay.github.io/game

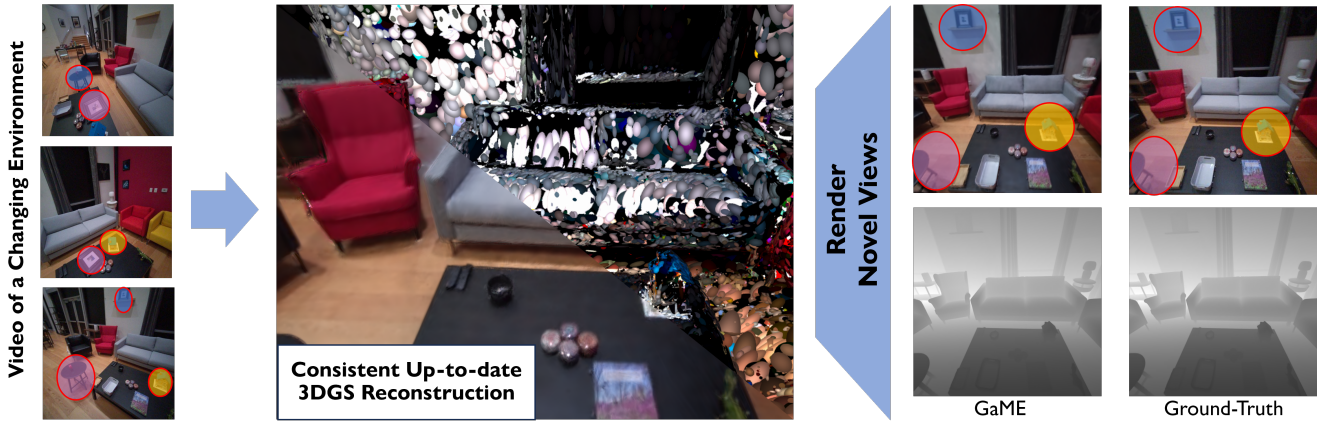

Figure 1. **GaME** is a dense mapping method capable of novel view synthesis. Given a single-camera posed RGBD input stream of an evolving environment, *i.e.*, structural changes happening outside of the camera view but during data capture, GaME reconstructs a consistent up-to-date 3D Gaussian map that can be rendered from previously unseen viewpoints. We showcase the high-fidelity 3D Gaussian map of a real-world environment, where our method effectively detects changes and accurately integrates them into the map. Our model accurately renders fine-grained objects that were moved during the recording, highlighted in blue, yellow, and purple.

Abstract

Mapping systems with novel view synthesis (NVS) capabilities, most notably 3D Gaussian Splatting (3DGS), are widely used in computer vision, as well as in various applications, including augmented reality, robotics, and autonomous driving. However, many current approaches are limited to static scenes. While recent works have begun addressing short-term dynamics (motion within the camera's view), long-term dynamics (the scene evolving through changes out of view) remain less explored. To overcome this limitation, we introduce a dynamic scene adaptation mechanism to continuously update 3DGS to reflect the latest changes. Since maintaining consistency remains challenging due to stale observations disrupting the reconstruction process, we further propose a novel keyframe management mechanism that discards outdated observations while preserving as much information as possible. We thoroughly evaluate Gaussian Mapping for Evolving Scenes (GaME) on both synthetic and real-world datasets, achieving a 29.7% improvement in PSNR and a 3×-improvement in L1 depth error over the most competitive baseline.

1. Introduction

Visual mapping reconstructs a consistent 3D representation of an environment from monocular or multi-view imagery. It is foundational for perception and decision-making in autonomous driving, mobile robotics, and AR/VR, where spatial understanding is required for navigation, planning, and interaction. Recent advances in visual dense reconstruction [20, 25, 28], have improved robustness and accuracy in complex, real-world scenes. Nevertheless, significant challenges persist in handling changing environments.

Recent mapping systems have been enhanced with novel view synthesis (NVS) capabilities [11, 18], enabling them to render realistic views of scenes, allowing for detailed scene exploration and the creation of high-quality virtual environments. These approaches typically assume that the scene is *static*, such that optimization over multiple frames is well-conditioned. However, real-world environments are predominantly dynamic and include both *short-term* dynamic effects (*i.e.*, things moving *within* view of the camera) as well as *long-term* dynamics (*i.e.*, the scene evolving through changes *outside* the view of the camera; for a detailed definition see [29]). While some of the most recent NVS approaches address short-term dynamic ob-

jects [36, 42], long-term dynamics remain less explored. As a result, modern reconstruction methods with NVS capabilities struggle to capture changes as the scene evolves during data capture but outside the camera view, resulting in corrupted and erroneous reconstruction.

During dense mapping of *long-term dynamic* scenes, systems must continuously update the 3D representation to capture the scene's evolution and maintain reliable operation. This is especially challenging compared to classical *multi-session* mapping, as changes can occur *at any time* during data collection. Recently, initial methods have started to address this challenge. A first approach is presented in Panoptic Multi-TSDFs [27], which builds semantically consistent submaps and reasons about changes at the submap level. The following works [6, 22, 28] similarly focus on object-level mapping to handle evolving scenes. However, all these methods rely on map representations that cannot easily be re-rendered, preventing their use in AR/VR or digital maps where realistic rendering is essential.

This work addresses the challenge of NVS mapping using 3DGS in evolving, long-term dynamic scenes. The key insight of our method is that two main problems prevent successful 3DGS mapping in changing environments: an outdated map and stale observations. We address these limitations by introducing a dynamic scene adaptation (DSA) mechanism that detects changes and integrates them into the 3DGS map, as well as a keyframe management system that removes stale observations while minimizing the loss of information. We make the following contributions:

- We present GaME, the first NVS-capable mapping system addressing evolving, long-term dynamic scenes.
- A Dynamic Scene Adaptation (DSA) mechanism to incrementally update a 3DGS model.
- An efficient keyframe management strategy for accurate 3D reconstruction through long-term scene changes.
- We thoroughly evaluate GaME on synthetic and real-world data, showing a performance increase of 325% in depth and 29% in color rendering over the most competitive baselines. We release the code open-source.

2. Related Work

3D Change Detection. The goal of change detection is to handle long-term dynamic effects, *i.e.*, changes to the scene occurring *outside* the view of the sensor. Typically, this is addressed in a *multi-session* setting. Once a map for each session is constructed, changes can be identified through *scene differencing* [1, 5, 13, 26]. To achieve an object-level change understanding, this has been extended using semantic information [15], where recent trends focus on the extraction of more specialized object features, including learned shape descriptors [7, 26, 32], neural object representations [6, 43], and language embeddings [24]. Most related to us, Lu et al. [16] recently presented a 3D Gaussian Splatting-based [12] approach by re-rendering the scene to newly collected views and using EfficientSAM [35] for 2D change detection. However, the assumption that the scene does not change during each session and offline processing is highly limiting, as changes in evolving scenes can occur *at any time* during the mapping process. In contrast, our approach is designed to operate online and does not impose any limitations on when the scene changes.

Online Reconstruction of Evolving Scenes. Recently, methods addressing the modeling of evolving scenes in an online fashion have emerged. A first approach is presented by Schmid et al. [27], which generates local semantically consistent submaps and incrementally reasons about changes at the submap level. Fu et al. [6] proposes neural object descriptors to build an object-level pose graph and detect changes in the graph configuration. This has recently been extended with $SE(3)$-equivariant descriptors [8]. Qian et al. [22] present a frame-to-map-tracking approach, using a probabilistic update rule to detect object-level changes, which is further extended to a variational factor-graph approach in [23]. A unified short- and long-term dynamic reconstruction formulation is presented in Khronos [28], where objects are reconstructed locally and changes detected using a library of rays. Nonetheless, these methods rely on map representations that do not support realistic rendering. In contrast, GaME provides scene reconstruction capable of real-time color and depth rendering.

Dynamic Gaussian Splatting. 3D Gaussian Splatting [12] has revolutionized novel view synthesis (NVS) by enabling photorealistic, real-time rendering at over 100 FPS. Compared to neural radiance fields [19], 3DGS is significantly more memory-efficient and faster to optimize. The problem of dynamic or 4D GS has attracted broad interest. A series of works [3, 17, 34, 38] optimize a canonical set of Gaussians from the initial frame and model temporal variations through a deformation field. However, these methods are limited to short video sequences, as they cannot introduce new Gaussians after the initial frame. Another class of approaches [4, 10, 39] directly models temporal Gaussians that can exist over subsets of frames. Despite their improved flexibility, these methods still require offline optimization and multi-view input, which makes scalability a significant bottleneck for high-quality dynamic reconstruction. In contrast, our mapping operates online using only a single RGB-D camera.

Online Gaussian Mapping. Most related to us, 3D Gaussian Splatting has sparked a wave of online RGB-D mapping methods [9, 11, 18, 37, 40] that adopt 3D Gaussians as their primary scene representation. While these methods perform well in static scenes, they struggle in dynamic environments. More recent approaches [36, 42] tackle *short-term* dynamics within the camera's view by incorporating segmentation models to suppress transient objects, such as

humans or small moving elements. However, they remain limited in addressing *long-term* scene changes, as stale observations corrupt the optimization process. In contrast, GaME is designed to robustly handle evolving environments by filtering outdated information and maintaining high rendering quality throughout reconstruction.

3. Background: 3D Gaussian Splatting

3D Gaussian splatting (3DGS) [12] is an effective method for representing 3D scenes with novel-view synthesis capability. This approach is notable for its speed, without compromising the rendering quality. In Kerbl et al. [12], 3D Gaussians are initialized from sparse Structure-from-Motion points and are further optimized using differentiable rendering from multiple views. Each Gaussian is parameterized by mean $\mu \in \mathbb{R}^3$, covariance $\Sigma \in \mathbb{R}^{3\times3}$, opacity $o \in \mathbb{R}$, and RGB color $C \in \mathbb{R}^3$. The mean of a projected (splatted) 3D Gaussian in the 2D image plane μ^I is computed as follows:

$$\mu^I = \pi\big(P(T_{wc}\mu_{\text{homogeneous}})\big) \ , \tag{1}$$

where $T_{wc} \in SE(3)$ is the world-to-camera transformation, $P \in \mathbb{R}^{4\times4}$ is an OpenGL-style projection matrix, $\pi : \mathbb{R}^4 \to \mathbb{R}^2$ is a projection to pixel coordinates. The 2D covariance Σ^I of a splatted Gaussian is:

$$\Sigma^I = JR_{wc}\Sigma R_{wc}^T J^T \ , \tag{2}$$

where $J \in \mathbb{R}^{2\times3}$ is an affine transformation from Zwicker et al. [44], $R_{wc} \in SO(3)$ is the rotation component of world-to-camera transformation T_{wc}. We refer to Zwicker et al. [44] for further details about the projection matrices. The color C along one channel ch at a pixel i is influenced by m ordered Gaussians an rendered as:

$$C_i^{ch} = \sum_{j\le m} C_j^{ch}\,\alpha_j \prod_{k<j}(1-\alpha_k),$$
$$\text{where} \quad \alpha_j = o_j\, e^{-\sigma_j}, \qquad \sigma_j = \tfrac{1}{2}\,\Delta_j^\top \Sigma_j^{-1}\Delta_j. \tag{3}$$

where $\Delta_j \in \mathbb{R}^2$ is the offset between the pixel coordinates and the 2D mean of a splatted Gaussian. The parameters of the 3D Gaussians are iteratively optimized by minimizing the photometric loss between rendered and training images. During optimization, C is encoded with spherical harmonics $SH \in \mathbb{R}^{15}$ to account for direction-based color variations. Covariance is decomposed as $\Sigma = RSS^TR^T$, where $R \in SE(3)$ and $S = \text{diag}(s) \in \mathbb{R}^{3\times3}$ are rotation and scale, respectively, to preserve the covariance positive semi-definite property during gradient-based optimization.

4. Method

GaME builds and maintains a 3D map capable of novel view synthesis of a static environment undergoing structural changes happening outside the sensor's view at any time. Our method processes posed depth and color images from a single monocular RGB-D sensor. For each keyframe, GaME triggers the Dynamic Scene Adaptation (DSA) module to incorporate changed, removed, or added geometry in the global map. Simultaneously, the keyframe management system updates the observations to ensure they reflect the scene's up-to-date state and do not disrupt the optimization process. An overview of our approach is shown in Fig. 2.

4.1. Online Mapping

The primary goal of our method is to create a 3DGS map that is consistent with the most recent observations. Unlike previous work [27] operating on the point clouds, we chose not to model objects as separate entities, as this is prohibitively expensive in a 3DGS setup. Moreover, accurately optimizing individual objects represented as 3DGS is hard due to the nature of the splatting mechanism (3), which inherently correlates all Gaussians. Instead, we propose to build a singular representation consisting of a set of 3D Gaussians $\{\mathbf{G}_i\}_{i=1}^N$ and extract changed geometry on an 'as needed' basis during *Dynamic Scene Adaptation* (Sec. 4.2). During online mapping, we optimize the 3DGS parameters of the scene using a set of keyframes for supervision, minimizing the loss:

$$L = L_{\text{color}}(\hat{I}, I) + L_{\text{depth}}(\hat{D}, D), \tag{4}$$

where I is the original image, $\hat{I}$ is the rendered image, D and $\hat{D}$ are the measured and reconstructed depth maps. The color loss and depth losses are defined as:

$$L_{\text{color}}(\hat{I}, I) = (1-\lambda)\,\frac{1}{K}\sum_p \big|\hat{I}(p) - I(p)\big| + \lambda\,\big(1 - \text{SSIM}(\hat{I}, I)\big), \tag{5}$$

$$L_{\text{depth}}(\hat{D}, D) = \frac{1}{K}\sum_p |\hat{D}(p) - D(p)|, \tag{6}$$

where K is the number of rendered pixels in the rendered image or depth map, $p \in \mathbb{Z}^2$ denotes the pixel coordinates, and SSIM is a structure similarity loss [33], and $\lambda \in [0, 1]$ is the weight balancing the L1 and SSIM terms. We additionally include an isotropic regularization term [40] to prevent degenerate Gaussian shapes.

4.2. Dynamic Scene Adaptation

In addition to reconstructing newly explored regions of the scenes, GaME addresses three potential scenarios: the addition, movement, or removal of rigid geometry outside of the camera view. While GaME can be readily combined with object re-localization techniques [8, 26, 32], object tracking

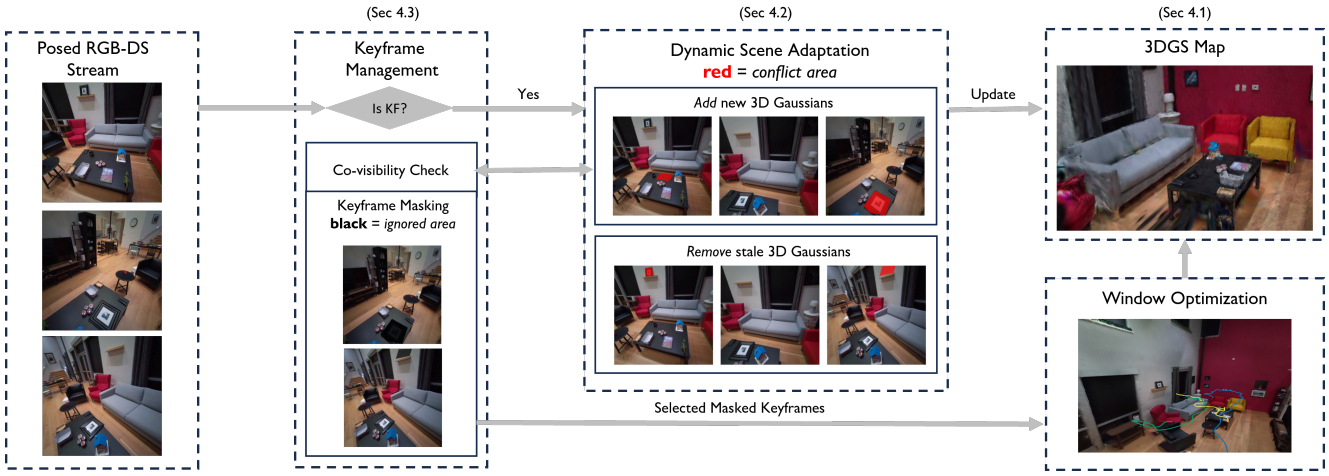

Figure 2. **GaME Architecture.** Given a segmented RGB-D input stream, the keyframe management system selects keyframes and triggers the dynamic scene adaptation (DSA) module. DSA first integrates newly observed geometry, then removes outdated geometry using covisible keyframes from the 3D Gaussian Splatting map. The keyframe manager then masks stale regions, and the mapping system uses the processed keyframes for local covisibility window optimization.

does not affect NVS, and the problem can thus be simplified into two core operations: *Add* and *Remove*.

Add. As the first step, DSA utilizes the current observations to detect changes in the scene where new geometry has been added. They are defined as already reconstructed Gaussians where the observed depth is closer to the camera than the rendered depth from the model:

$$\mathcal{G}_{\text{add}} = \Big\{ \mathbf{G}_i \,\Big|\, \alpha(\mathbf{p}) \geq \epsilon_{\text{opacity}} \wedge \hat{D}(\mathbf{p}) > D(\mathbf{p}) + \epsilon_{\text{depth}} \Big\}, \tag{7}$$

where $\alpha(\mathbf{p})$ represents rendered opacity at pixel $\mathbf{p} \in \mathbb{Z}^2$, $\epsilon_{\text{opacity}}, \epsilon_{\text{depth}} \in \mathbb{R}$ are the thresholds accounting for the rendering noise.

On the next step, DSA dynamically seeds new Gaussian primitives in newly explored regions to reconstruct the complete scene. Following [40], such regions $\mathcal{R}_{\text{new}}$ are characterized by low opacity, a large rendered depth overshoot, or high color error:

$$\mathcal{R}_{\text{new}} = \Big\{ \mathbf{p} \in \mathbb{Z}^2 \,\Big|\, \alpha(\mathbf{p}) < \epsilon_{\text{opacity}} \vee \hat{D}(\mathbf{p}) - D(\mathbf{p}) > \tau \cdot \text{median}(L_{\text{depth}}) \vee L_{\text{color}}(\mathbf{p}) > \epsilon_{\text{seed}} \Big\}. \tag{8}$$

where $L_{\text{depth}}(\mathbf{p}) = |\hat{D}(\mathbf{p}) - D(\mathbf{p})|$ and $L_{\text{color}}(\mathbf{p}) = |\hat{I}(\mathbf{p}) - I(\mathbf{p})|$ denote the per-pixel depth and color errors, and $\tau \in \mathbb{R}_{>0}$, $\epsilon_{\text{seed}} \in \mathbb{R}$ are thresholds. The method lifts the input RGB-D frame in $\mathcal{R}_{\text{new}}$ into 3D space and uses it as the means for new Gaussians. After that, the new Gaussians are optimized against the current keyframe for a small number of iterations (see Sec. 4.1) before the *Remove* step.

Remove. On the next step, DSA detects stale Gaussians that need to be removed from the scene representation. Specifically, changed parts of the 3DGS model can be identified as those with high opacity, but disagreeing visually or geometrically with the new observations:

$$\mathcal{G}_{\text{remove}} = \Big\{ \mathbf{G}_i \,\Big|\, \alpha(\mathbf{p}) \geq \epsilon_{\text{opacity}} \wedge L_{\text{color}}(\mathbf{p}) > \epsilon_{\text{color}} \wedge \hat{D}(\mathbf{p}) < D(\mathbf{p}) - \epsilon_{\text{depth}} \Big\} \tag{9}$$

where $\epsilon_{\text{color}} \in \mathbb{R}$ accounts for color rendering noise. There are two notable changes compared to the *Add* condition. First, the sign of the depth criterion is inverted, reflecting areas where the rays of the observation would penetrate into the current model, thus indicating that the model geometry can no longer be present. This also avoids spurious detections where objects are simply occluded rather than absent. Second, we leverage the high visual fidelity of 3DGS as another conflict signal by adding a color term. In contrast to most long-term mapping methods, which rely solely on 3D information [5, 13, 23, 26–28], this allows GaME to also detect changes that are not geometrically significant, such as a picture removed from the wall.

Naively removing the $\mathcal{G}_{\text{remove}}$ breaks the consistency of the scene. Specifically, when only parts of the objects that need to be removed are seen in the current frame. To address this, DSA renders $\mathcal{G}_{\text{remove}}$ to each covisible keyframe KF and verifies whether the color and depth are consistent with the model:

$$\mathcal{R}^{\text{KF}}_{\text{remove}} = \Big\{ \mathbf{p} \in \mathbb{Z}^2 \,\Big|\, L_{\text{color}}(\mathbf{p}) < \epsilon_{\text{color}}, \; L_{\text{depth}}(\mathbf{p}) < \epsilon_{\text{depth}}, \; \alpha(\mathbf{p}) \geq \epsilon_{\text{opacity}} \Big\}. \tag{10}$$

Figure 3. **Illustration of *Add* and *Remove* operations.** The input disagrees with the rendered model (red). For disappearance (top), the conflicting region is projected to previous keyframes for removal (red), where complete object consistency is enforced through the object mask (blue). This allows GaME to extract complete objects even under partial observations and occlusion. When a new object appears on the scene (bottom), new Gaussians are added (red) and the area of the new object is marked as stale in previous keyframes to prevent the contamination of the optimization process.

$\mathcal{R}^{\text{KF}}_{\text{remove}}$ highlights the areas of previous keyframes where they observe the Gaussians that are removed. However, the detected conflicting areas might not accurately cover the objects due to occlusions and sensing noise. To address this, DSA uses dense SAM [14] masks of the keyframes, and selects all masks that are intersecting with the rendering of $\mathcal{G}_{\text{remove}}$. Then the selected masks are used to segment the Gaussians that additionally need to be removed $\mathcal{G}^{\text{KF}}_{\text{remove}}$. Our method uses FlashSplat [30] to optimally assign masks to the Gaussians, providing multiview consistency making our method robust to the noisy depth and color renderings. Notably, SAM masks may overlap and carry no semantic labels; their sole purpose is to delineate the complete silhouette of object that needs to be removed. This allows DSA to remove whole objects, even when only parts of them were visible in a current frame.

Finally, the joint set, $\mathcal{G}_{\text{remove}} \cup \bigcup_{\text{KF}} \mathcal{G}^{\text{KF}}_{\text{remove}}$, is removed from the global Gaussian model, and all the Gaussians are optimized for a small number of iterations to integrate the changes. The DSA process is illustrated in Fig. 3.

4.3. Keyframe Management

Since jointly optimizing Gaussians using all frames from a video stream is computationally infeasible, we maintain a smaller set of keyframes W_k. Effective keyframe management (KM) aims to select non-redundant keyframes that observe the same region while spanning a wide baseline to enforce stronger multiview constraints. Following [11, 40], keyframes are selected every time a frame exceeds a translation $\theta_{\text{translation}} \in \mathbb{R}$ or a rotation $\theta_{\text{rotation}} \in \mathbb{R}$ threshold.

Covisibility. Upon triggering *Add* or *Remove* operations by DSA, all covisible keyframes are retrieved by reprojecting 3D points from the current frame and selecting frames where these points are not occluded. This approach differs from conventional 3DGS methods [18, 36], which determine covisibility based on the number of Gaussians used to re-render the keyframes. We found projection to be more reliable than re-rendering in multi-room environments, as Gaussians from different rooms may appear visible due to alpha blending, leading to erroneous covisibility estimates.

Keyframe Masking. In evolving scenes, excluding outdated visual information plays a critical role, as the scene may change over time, making previous observations detrimental for the 3DGS optimization process. However, discarding entire keyframes that observe stale geometry can result in losing useful information. While some parts of the scene may change, other regions often remain stable and can provide valuable signals for 3DGS optimization. Therefore, rather than discarding entire keyframes as stale, GaME

selectively ignores only the stale areas within them.

To achieve this, before seeding new Gaussians in $\mathcal{R}_{\text{new}}$ or removing $\mathcal{G}_{\text{remove}}$ and $\mathcal{G}^{\text{KF}}_{\text{remove}}$, our keyframe management system renders them onto covisible keyframes and assesses conflicts via photometric and geometric losses Eq. (4). High-error regions correspond to areas of the map that were changed but are not reflected in the observations. To mitigate rendering noise [12], these regions are refined into object-aligned areas before being marked as ignored: for *Remove*, using precomputed SAM [14] masks that intersect the conflict region; for *Add*, via morphological closing of the reprojected depth conflict. During optimization, all keyframes with any SAM mask that is sufficiently covered by the erroneous area are ignored. For the rest, GaME optimizes the losses from Eq. (4) using the masked version of losses. We visualize the keyframes with masked out stale areas on Fig. 4.

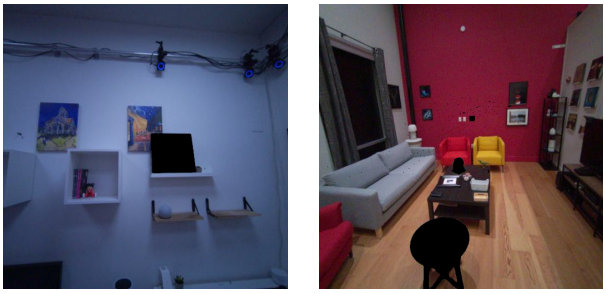

Figure 4. **Visualizations of the outdated keyframes.** After detecting the changes, our keyframe management masks out the areas of covisible keyframes observing changed geometry.

5. Experiments

Datasets. We test our method on the Flat [27] dataset, which consists of two RGB-D sequences captured in a synthetic environment with significant changes occurring between. We further evaluate on the Aria [21] dataset to assess performance on real-world data. We select two recordings from two rooms each that have undergone long-term changes. Finally, the TUM-RGBD [31] dataset is used to assess performance on three different static scenes following the protocol of [18].

Evaluation Metrics. To assess rendering quality, we compute PSNR, SSIM [33] and LPIPS [41]. Rendering metrics on all the datasets are evaluated by rendering full-resolution images along the ground-truth trajectory. We assess the depth error using the L1 norm in centimeters. In the tables, results are color-coded to indicate ranking: **best**, second-best, and third-best.

Baselines. We compare GaME with state-of-the-art 3DGS online reconstruction systems MonoGS [18] and SplaTAM [11]. We compare to DG-SLAM [16], a recent dynamic 3DGS method, to assess the ability of dynamic 3DGS SLAM to handle evolving scenes. Wild-GS [42] cannot be run with ground-truth poses, as it relies on an off-the-shelf depth estimator that may be inconsistent with them. At the same time, it fails to track the camera poses on the Aria and Flat datasets. For this reason, we were not able to compare our method with it.

Methods	PSNR [dB] ↑	SSIM ↑	LPIPS ↓	Depth L1 [cm] ↓
SplaTAM [11]	15.88 / 12.69	0.48 / 0.30	0.55 / 0.70	21.16 / 59.91
MonoGS [18]	21.24 / 21.33	0.77 / 0.77	0.40 / 0.40	30.95 / 29.91
DG-SLAM [16]	13.72 / 13.70	0.59 / 0.60	0.74 / 0.74	73.76 / 73.90
GaME (Ours)	**24.55 / 24.26**	**0.93 / 0.93**	**0.14 / 0.14**	**6.9 / 7.9**

Table 1. **Rendering performance on the Flat dataset.** Our method significantly outperforms other baselines thanks to dynamic scene adaptation and keyframe management system. Metrics are reported for input / novel views.

Evolving Scene Evaluation Protocol. The goal of mapping evolving scenes is to render only the most up-to-date reconstruction without prior information on when the scene was changed. In addition, the system should be able to accurately render both the views it observed (input views) and unobserved frames (novel views). To achieve this, we merge RGB-D captures from every scene in the Aria and Flat datasets into a single continuous sequence to recreate this real-world behavior. For rendering evaluation, every 10th frame from each scene's last RGB-D sequence is held out for novel view synthesis testing. The remaining 90% of frames are used to evaluate input view synthesis. To isolate the mapping performance, ground-truth poses are used for GaME and all baselines.

5.1. Evolving Scene Mapping Results

We run GaME on the Flat and Aria datasets to evaluate rendering quality in evolving scenes, shown in Tabs. 1 and 2. The Flat dataset is designed to test mapping under scene changes and includes more substantial long-term dynamics. We note that all losses are computed over complete images, where changed areas typically occupy a smaller part. Nonetheless, we observe notable differences in rendering performance on both datasets, where GaME is the only method that accurately adapts the reconstruction to even subtle scene changes without compromising rendering quality. This granularity is shown in qualitative results in Fig. 5. Beyond high rendering quality, GaME is able to accurately resolve changes also for challenging cases such as small cutlery on the table and flat paintings on the walls.

5.2. Ablation Studies

Add and Remove operations are important for Dynamic Scene Adaptation. We ablate the components of DSA in Tab. 3, showing that its presence significantly improves performance. While differences between individual DSA variants are less pronounced in final rendering quality, due to keyframe optimization correcting finer detail, combining both operations consistently yields the best performance.

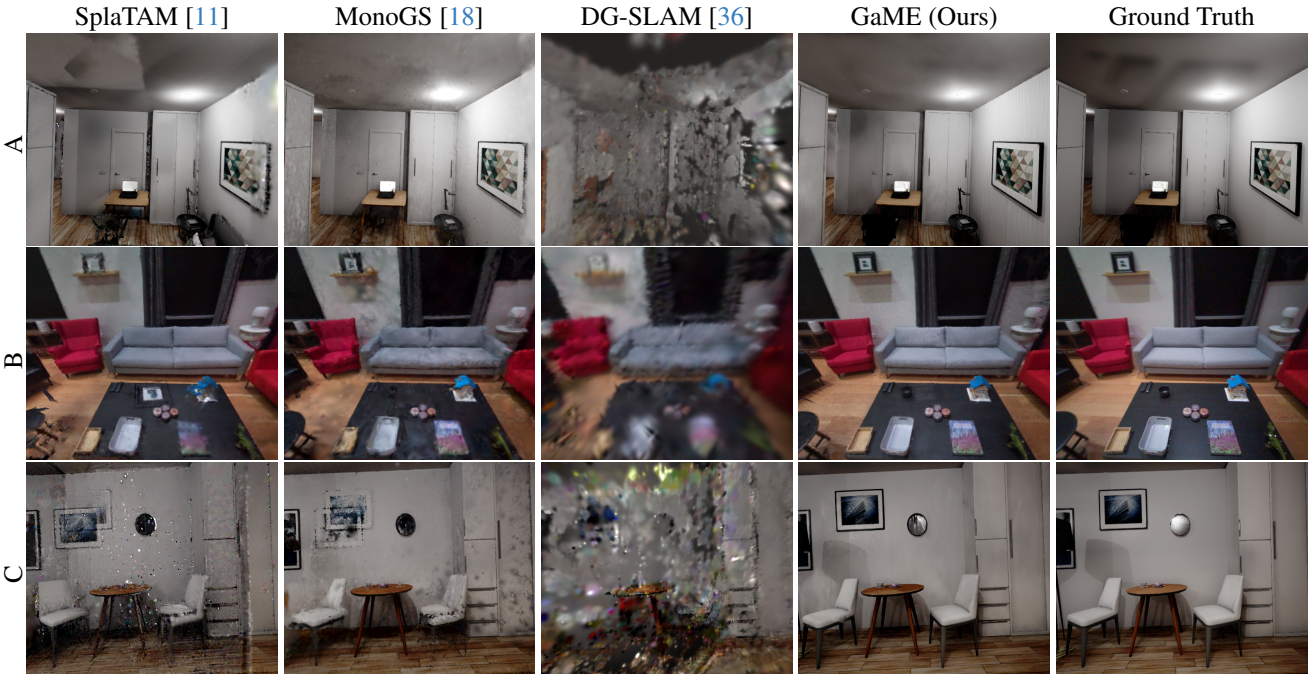

Figure 5. **Qualitative Results.** Comparison across different long-term scene changes. (A) A black office chair appears in the scene; (B) the toy house and chair are moved, the picture is moved from the table to the shelf; (C) the cutlery on the table is replaced, the painting and the right chair are moved. GaME is the only method that captures the scene evolution and preserves high rendering quality.

Methods	Scene	PSNR [dB] ↑	SSIM ↑	LPIPS ↓	Depth L1 [cm] ↓
SplaTAM [11]	room0	1.80 / 16.75	0.78 / 0.45	0.24 / 0.44	4.12 / 17.79
	room1	22.94 / 17.90	0.81 / 0.54	0.22 / 0.44	2.84 / 15.90
	Avg	22.37 / 17.33	0.80 / 0.50	0.23 / 0.44	3.48 / 16.80
MonoGS [18]	room0	25.28 / 25.19	0.78 / 0.78	0.28 / 0.27	5.15 / 5.09
	room1	23.12 / 23.02	0.84 / 0.84	0.24 / 0.24	4.99 / 5.01
	Avg	24.20 / 24.11	0.81 / 0.81	0.26 / 0.26	5.07 / 5.05
DG-SLAM [16]	room0	15.78 / 15.63	0.58 / 0.58	0.76 / 0.76	67.60 / 69.05
	room1	12.62 / 12.44	0.65 / 0.64	0.70 / 0.71	55.47 / 57.03
	Avg	14.20 / 14.04	0.62 / 0.61	0.73 / 0.74	61.54 / 63.04
GaME (Ours)	room0	**31.54 / 31.48**	**0.95 / 0.95**	**0.14 / 0.14**	**0.69 / 0.69**
	room1	**31.23 / 31.97**	**0.95 / 0.95**	**0.10 / 0.10**	**1.75 / 1.80**
	Avg	**31.39 / 31.23**	**0.95 / 0.95**	**0.12 / 0.12**	**1.22 / 1.24**

Table 2. **Rendering performance on the Aria dataset.** Our method significantly outperforms other baselines thanks to dynamic scene adaptaion and keyframe management system. Metrics are reported for input / novel views.

Method	PSNR [dB] ↑	SSIM ↑	LPIPS ↓	Depth L1 [cm] ↓
No DSA	21.25 / 21.08	0.87 / 0.87	0.2 / 0.2	29.20 / 27.20
Add	23.34 / 23.07	**0.92 / 0.91**	**0.15 / 0.15**	11.00 / 11.20
Remove	24.13 / 23.89	**0.93 / 0.93**	**0.14 / 0.14**	9.50 / 8.20
Add and Remove (Ours)	**24.55 / 24.26**	**0.93 / 0.93**	**0.14 / 0.14**	**6.9 / 7.9**

Table 3. **Dynamic Scene Adaptation (DSA) ablation on the Flat dataset.** Combining *Add* and *Remove* operations gives the best performance. Metrics are reported for input / novel views.

Our Keyframe Management system preserves useful information. In Tab. 4, we compare keeping all keyframes against ignoring stale keyframes and ignoring only stale regions. We observe that the naive approach to 3DGS mapping in evolving scenes by discarding conflicting keyframes can leave parts of the scene severely under-constrained, leading to the worst outcome. Ignoring changes (no filtering) achieves clear background rendering (note that the background accounts for the majority of pixels and dominates the losses), but inevitably results in artifacts and inconsistencies for changed regions, as seen in Fig. 5. Our proposed approach highlights the importance of retaining information about the background *and* resolving conflicts in keyframes for 3DGS optimization.

Method	PSNR [dB] ↑	SSIM ↑	LPIPS ↓	Depth L1 [cm] ↓
No KF Filtering	23.57 / 23.31	0.92 / 0.92	0.15 / 0.14	6.70 / 7.20
Full KF Filtering	13.72 / 13.34	0.58 / 0.57	0.56 / 0.55	764.40 / 763.30
Partial KF Filtering (Ours)	**24.55 / 24.26**	**0.93 / 0.93**	**0.14 / 0.14**	**6.9 / 7.9**

Table 4. **Keyframe management ablation on the Flat dataset.** Retaining keyframes is essential to constrain the background, but conflicting regions must be accurately filtered to avoid artifacts.

GaME can handle noisy poses. We evaluate the camera poses on the Aria dataset with an external SLAM [18] system and use the estimated poses and keyframes instead of ground truth in Tab. 5. GaME is robust even when the pose estimation is not precise, exhibiting only a marginal performance drop under noisy pose conditions.

GaME can reconstruct static scenes. While our method is specifically designed to reconstruct evolving scenes, it should not lose the ability to reconstruct static scenes. Results on the TUM-RGBD dataset shown in Tab. 6 show that GaME performs on par with state-of-the-art systems. This

Camera Poses	PSNR [dB] ↑	SSIM ↑	LPIPS ↓	Depth L1 [cm] ↓
Ground-truth	**31.54 / 31.48**	**0.95 / 0.95**	**0.14 / 0.14**	**0.69 / 0.69**
Estimated	31.45 / **31.53**	**0.95 / 0.95**	**0.14 / 0.14**	0.70 / 0.70

Table 5. **Effect of noisy poses on Aria room0.** GaME is robust to camera pose noise, showing no significant impact on performance when using estimated poses instead of ground truth. Camera poses and keyframes are estimated with an off-the-shelf method [18].

Method	desk	xyz	office	Average
SplaTAM [11]	**20.92**	**21.03**	**21.61**	**21.19**
MonoGS [18]	17.41	15.09	19.93	17.48
GaME (Ours)	20.18	21.01	20.71	20.63

Table 6. **Rendering performance on the TUM-RGBD dataset.** PSNR [dB]↑ is reported for input views.

Metric	SplaTAM [11]	MonoGS [18]	DG-SLAM [36]	GaME (Ours)
FPS ↑	0.13	**4.21**	0.25	0.52

Table 7. **Runtime analysis on the Flat dataset.** FPS is calculated by dividing the mapping time by the total number of frames. All metrics are profiled on an NVIDIA RTX 3090 GPU.

Method	PSNR [dB] ↑	SSIM ↑	LPIPS ↓	Depth L1 [cm] ↓
No Refinement	23.70 / 23.46	0.92 / 0.92	0.16 / 0.16	**7.2 / 6.6**
With Refinement	**24.55 / 24.26**	**0.93 / 0.93**	**0.14 / 0.14**	7.9 / 6.6

Table 8. **Final refinement ablation on the Flat dataset.** While refinement can further improve rendering quality, GaME already converges well during incremental processing.

suggests that the introduced DSA and keyframe management are robust to noise and false positives in model updates and masking, which could deteriorate performance. In the future, GaME could well be combined with tracking and registration methods to improve performance, however, this is currently beyond the scope of this work.

GaME is an online mapping system. While addressing scene changes requires additional computation time, our method performs on par with standard 3DGS reconstruction systems as shown in Tab. 7. While the run time is not yet real-time, our approach is of an incremental nature with partial optimization performed on every new keyframe. We believe that future improvements in 3DGS optimization speed will translate well to GaME without incurring notable extra cost to handle evolving scenes. Optionally, our approach lends itself to further final refinement. Both options are shown in Tab. 8. While refinement can potentially improve the final rendering quality, we note that the model is already well converged after incremental processing.

GaME is robust. Since the main focus of our work is on the architectural challenges when handling evolving scenes in 3DGS and not image-based change detection, we opted for a comparably simple threshold-based approach. While

Hyperparameter	PSNR Mean (↑)	PSNR Std. Dev. (↓)	Depth L1 [cm] Mean (↓)	Depth L1 [cm] Std. Dev. (↓)
ϵ_{depth}	31.52	0.21	0.68	0.03
ϵ_{color}	31.58	0.14	0.69	0.01
$\epsilon_{\text{opacity}}$	31.56	0.11	0.68	0.01

Table 9. **Hyperparameters robustness on the Aria room0 dataset.** GaME is robust to hyperparameter variation.

GaME is readily extendable with more complex methods such as [16], we already find good performance and that GaME is robust w.r.t. critical hyperparameters. First, most of our hyperparameters have an interpretable meaning, making them intuitive and requiring minimal effort to adjust. For instance, to determine $\epsilon_{\text{depth}}, \epsilon_{\text{color}}$, we inspected the rendering error in color and depth caused by adding a small object (e.g., a cup on the table) to the scene. For keyframe selection, we adopted standard rotation and translation thresholds [11, 40]. We experimentally show GaME to be robust w.r.t. hyperparameters by varying key parameters $\epsilon_{\text{depth}}, \epsilon_{\text{color}}, \epsilon_{\text{opacity}}$ with steps from the set of $\{-20\%, -10\%, 10\%, 20\%\}$ and report the mean and standard deviation of PSNR and Depth L1 error on the Flat dataset in Tab. 9. The minimal variation in scores confirms that our method does not rely on sensitive parameter tuning.

6. Limitations

While GaME currently demonstrates robust rendering performance in evolving long-term dynamic scenes, several notable limitations exist. First, it does not yet handle short-term dynamic objects, which would result in inefficient addition and removal. Extending the dynamic scene adaptation mechanism to consider both dynamics is an exciting future direction. Second, we primarily study mapping with external pose tracking. Although sparse odometry systems can reject many changes as outliers [2], change-aware tracking mechanisms are an exciting future direction. Finally, while GaME builds a consistent model of the scene at the final time, it would be interesting to explore a complete 4D representation of the history of all changes.

7. Conclusions

We presented GaME, the first online mapping system for evolving scenes with novel view synthesis capabilities. GaME utilizes novel dynamic scene adaptation operations to detect and correct conflicts in the incrementally built 3DGS model. Our keyframe management method furthermore appropriately ignores stale areas in keyframes, retaining useful information while correcting for changed regions, resulting in a well-conditioned 3DGS optimization process. We thoroughly evaluate GaME on synthetic and real-world datasets, demonstrating 29.7% improvement in PSNR, over 3×-improvement in L1 depth error, and artifact-free novel view synthesis in evolving scenes. The code and data are released open-source.

Acknowledgements. This work was supported by TomTom, the University of Amsterdam, Amazon, ARL DCIST program, and the allowance of Top Consortia for Knowledge and Innovation (TKIs) from the Netherlands Ministry of Economic Affairs and Climate Policy.

References

[1] Rareş Ambruş, Nils Bore, John Folkesson, and Patric Jensfelt. Meta-rooms: Building and maintaining long term spatial models in a dynamic world. pages 1854–1861. IEEE, 2014. 2

[2] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE transactions on robotics*, 37(6):1874–1890, 2021. 8

[3] Devikalyan Das, Christopher Wewer, Raza Yunus, Eddy Ilg, and Jan Eric Lenssen. Neural parametric gaussians for monocular non-rigid object reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10715–10725, 2024. 2

[4] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d-rotor gaussian splatting: Towards efficient novel-view synthesis for dynamic scenes. In *Proc. SIGGRAPH*, 2024. 2

[5] Marius Fehr, Fadri Furrer, Ivan Dryanovski, Jürgen Sturm, Igor Gilitschenski, Roland Siegwart, and Cesar Cadena. Tsdf-based change detection for consistent long-term dense reconstruction and dynamic object discovery. pages 5237–5244. IEEE, 2017. 2, 4

[6] Jiahui Fu, Yilun Du, Kurran Singh, Joshua B Tenenbaum, and John J Leonard. Robust change detection based on neural descriptor fields. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2817–2824. IEEE, 2022. 2

[7] Jiahui Fu, Chengyuan Lin, Yuichi Taguchi, Andrea Cohen, Yifu Zhang, Stephen Mylabathula, and John J Leonard. Planesdf-based change detection for long-term dense mapping. *IEEE Robotics and Automation Letters*, 7(4):9667–9674, 2022. 2

[8] Jiahui Fu, Yilun Du, Kurran Singh, Joshua B Tenenbaum, and John J Leonard. Neuse: Neural se (3)-equivariant embedding for consistent spatial understanding with objects. 2023. 2, 3

[9] Huajian Huang, Longwei Li, Cheng Hui, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2

[10] Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. A compact dynamic 3d gaussian representation for real-time dynamic view synthesis. page 394–412, Berlin, Heidelberg, 2024. Springer-Verlag. 2

[11] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2, 5, 6, 7, 8

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

[13] Giseop Kim and Ayoung Kim. Lt-mapper: A modular framework for lidar-based lifelong mapping. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7995–8002. IEEE, 2022. 2, 4

[14] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 5, 6

[15] Edith Langer, Timothy Patten, and Markus Vincze. Robust and efficient object change detection by combining global semantic information and local geometric verification. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8453–8460. IEEE, 2020. 2

[16] Ziqi Lu, Jianbo Ye, and John Leonard. 3dgs-cd: 3d gaussian splatting-based change detection for physical object rearrangement. *IEEE Robotics and Automation Letters*, 2025. 2, 6, 7, 8

[17] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *IEEE International Conference on 3D Vision*, 2024. 2

[18] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2, 5, 6, 7, 8

[19] 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. *Commun. ACM*, 65(1):99–106, 2021. 2

[20] Riku Murai, Eric Dexheimer, and Andrew J. Davison. MASt3R-SLAM: Real-time dense SLAM with 3D reconstruction priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. 1

[21] Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Carl Yuheng Ren. Aria digital twin: A new benchmark dataset for egocentric 3d machine perception, 2023. 6

[22] Jingxing Qian, Veronica Chatrath, Jun Yang, James Servos, Angela P Schoellig, and Steven L Waslander. Pocd: Probabilistic object-level change detection and volumetric mapping in semi-static scenes. 2022. 2

[23] Jingxing Qian, Veronica Chatrath, James Servos, Aaron Mavrinac, Wolfram Burgard, Steven L Waslander, and Angela P Schoellig. Pov-slam: Probabilistic object-aware variational slam in semi-static environments. 2023. 2, 4

[24] Adam Rashid, Chung Min Kim, Justin Kerr, Letian Fu, Kush Hari, Ayah Ahmad, Kaiyuan Chen, Huang Huang, Marcus Gualtieri, Michael Wang, et al. Lifelong lerf: Local

3d semantic inventory monitoring using fogros2. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7740–7747. IEEE, 2024. 2

[25] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021. 1

[26] Joseph Rowell, Lintong Zhang, and Maurice Fallon. Lista: Geometric object-based change detection in cluttered environments. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3632–3638. IEEE, 2024. 2, 3, 4

[27] Lukas Schmid, Jeffrey Delmerico, Johannes L. Schönberger, Juan Nieto, Marc Pollefeys, Roland Siegwart, and Cesar Cadena. Panoptic Multi-TSDFs: a Flexible Representation for Online Multi-resolution Volumetric Mapping and Long-term Dynamic Scene Consistency. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8018–8024, 2022. ICRA. 2, 3, 6

[28] Lukas Schmid, Marcus Abate, Yun Chang, and Luca Carlone. Khronos: A unified approach for spatio-temporal metric-semantic slam in dynamic environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2024. 1, 2, 4

[29] Lukas Schmid, José María Martínez Montiel, Shoudong Huang, Daniel Cremers, José Neira, and Javier Civera. Dynamic and deformable SLAM. In *SLAM Handbook. From Localization and Mapping to Spatial Intelligence*. Cambridge University Press, 2026. 1

[30] Qiuhong Shen, Xingyi Yang, and Xinchao Wang. Flashsplat: 2d to 3d gaussian splatting segmentation solved optimally. *European Conference of Computer Vision*, 2024. 5

[31] 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)*, 2012. 6

[32] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Nießner. Rio: 3d object instance relocalization in changing indoor environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7658–7667, 2019. 2, 3

[33] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 3, 6

[34] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, 2024. 2

[35] Yunyang Xiong, Bala Varadarajan, Lemeng Wu, Xiaoyu Xiang, Fanyi Xiao, Chenchen Zhu, Xiaoliang Dai, Dilin Wang, Fei Sun, Forrest Iandola, et al. Efficientsam: Leveraged masked image pretraining for efficient segment anything. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16111–16121, 2024. 2

[36] Yueming Xu, Haochen Jiang, Zhongyang Xiao, Jianfeng Feng, and Li Zhang. DG-SLAM: Robust dynamic gaussian splatting SLAM with hybrid pose optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2, 5, 7, 8

[37] 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*, 2024. 2

[38] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2

[39] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *International Conference on Learning Representations (ICLR)*, 2024. 2

[40] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting, 2023. 2, 3, 4, 5, 8

[41] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[42] Jianhao Zheng, Zihan Zhu, Valentin Bieri, Marc Pollefeys, Songyou Peng, and Iro Armeni. Wildgs-slam: Monocular gaussian splatting slam in dynamic environments. *arXiv preprint arXiv:2504.03886*, 2025. 2, 6

[43] Liyuan Zhu, Shengyu Huang, Konrad Schindler, and Iro Armeni. Living scenes: Multi-object relocalization and reconstruction in changing 3d environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28014–28024, 2024. 2

[44] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 3