

WildGS-SLAM: Monocular Gaussian Splatting SLAM in Dynamic Environments

Jianhao Zheng^{1*} Zihan Zhu^{2*} Valentin Bieri² Marc Pollefeys^{2,3} Songyou Peng² Iro Armeni¹
¹Stanford University ²ETH Zürich ³Microsoft
wildgs-slam.github.io

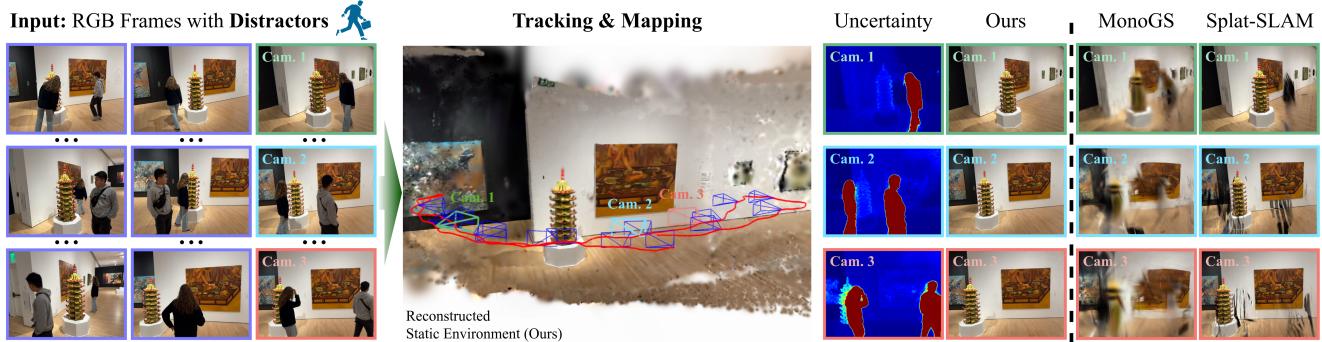


Figure 1. **WildGS-SLAM**. Given a monocular video sequence captured in the wild with dynamic distractors, our method accurately tracks the camera trajectory and reconstructs a 3D Gaussian map for static elements, effectively removing all dynamic components. This approach enables high-fidelity rendering even in complex, dynamic scenes. The illustration presents the final 3D Gaussian map, the camera tracking trajectory (in red), and view synthesis comparisons with baseline methods.

Abstract

We present *WildGS-SLAM*, a robust and efficient monocular RGB SLAM system designed to handle dynamic environments by leveraging uncertainty-aware geometric mapping. Unlike traditional SLAM systems, which assume static scenes, our approach integrates depth and uncertainty information to enhance tracking, mapping, and rendering performance in the presence of moving objects. We introduce an uncertainty map, predicted by a shallow multi-layer perceptron and DI-NOV2 features, to guide dynamic object removal during both tracking and mapping. This uncertainty map enhances dense bundle adjustment and Gaussian map optimization, improving reconstruction accuracy. Our system is evaluated on multiple datasets and demonstrates artifact-free view synthesis. Results showcase *WildGS-SLAM*'s superior performance in dynamic environments compared to state-of-the-art methods.

1. Introduction

Simultaneous Localization and Mapping (SLAM) in dynamic environments is a fundamental challenge in computer vision, with broad applications in autonomous navigation, augmented reality, and robotics. Traditional SLAM systems [7, 33, 47] rely on assumptions of scene rigidity, making them vulnerable to tracking errors in dynamic scenes

where objects move independently. Although some recent approaches [4, 36, 54] incorporate motion segmentation, semantic information, and depth-based cues to handle dynamic content, they often struggle to generalize across scenes with varied and unpredictable motion patterns. This issue is especially acute in real-world scenarios where dynamic distractors, occlusions, and varying lighting conditions introduce significant ambiguity for SLAM systems.

Uncertainty-aware methods have recently gained attention in scene reconstruction and view synthesis, particularly for handling complex environments with partial occlusions, dynamic objects, and noisy observations. For instance, NeRF *On-the-go* [38] and WildGaussians [24] introduced uncertainty estimation to improve the rendering quality of neural radiance fields in real-world scenarios, enabling enhanced view synthesis in the presence of motion and varying light conditions. Such approaches provide valuable insights into modeling ambiguities and have shown strong results in highly dynamic environments. However, they focus on sparse-view settings and require camera poses as input.

To address these limitations, we propose a novel SLAM approach, namely **WildGS-SLAM**, that leverages a 3D Gaussian Splatting (3DGS) representation, designed to perform robustly in highly dynamic environments using only monocular RGB input. Similar to [24, 38], our method takes a purely geometric approach. It integrates uncertainty-aware tracking and mapping, which removes dynamic distractors ef-

* Equal contribution.

fectively without requiring explicit depth or semantic labels. This approach enhances tracking, mapping, and rendering while achieving strong generalizability and robustness across diverse real-world scenarios. Results showcase its improved performance over prior work in both indoor and outdoor scenes, supporting artifact-free rendering and high-fidelity novel view synthesis, even in challenging settings.

Specifically, we train a shallow multi-layer perceptron (MLP) given 3D-aware, pre-trained DINOv2 [57] features to predict per-pixel uncertainty. The MLP is trained incrementally as input frames are streamed into the system, allowing it to dynamically adapt to incoming scene data. We leverage this uncertainty information to enhance tracking, guiding dense bundle adjustment (DBA) to prioritize reliable areas. Additionally, during mapping, the uncertainty predictions inform the rendering loss in Gaussian map optimization, helping to refine the quality of the reconstructed scene. By optimizing the map and the uncertainty MLP independently, we ensure maximal performance for each component. To evaluate our method in diverse and challenging scenarios, we collect a new dataset including indoor and outdoor scenes.

Our main contributions are as follows:

- A monocular SLAM framework, namely WildGS-SLAM, utilizing a 3D Gaussian representation that operates robustly in highly dynamic environments, outperforming existing dynamic SLAM methods on a variety of dynamic datasets and on both indoor and outdoor scenarios.
- An uncertainty-aware tracking and mapping pipeline that enables the accurate removal of dynamic distractors without depth or explicit semantic segmentation, achieving high-fidelity scene reconstructions and tracking.
- A new dataset, namely Wild-SLAM Dataset, featuring diverse indoor and outdoor scenes, enables SLAM evaluation in unconstrained, real-world conditions. This dataset supports comprehensive benchmarking for dynamic environments with varied object motions and occlusions.

2. Related Work

2.1. Traditional Visual SLAM

Most traditional visual SLAM [7, 22, 32, 33] methods assume static scenes, however, the presence of dynamic objects can disrupt feature matching and photometric consistency, leading to substantial tracking drift. To address this, many approaches enhance robustness in dynamic environments by detecting and filtering out dynamic regions, focusing on reconstructing the static parts of the scene. Common approaches to detect dynamic objects include warping or reprojection techniques [3, 36, 41], off-the-shelf optical flow estimators [46, 63], predefined class priors for object detection or semantic segmentation [18, 43], or hybrids of these strategies [2, 4, 42, 53]. Notably, ReFusion [36] requires RGB-D input and uses a TSDF [5] map representation, leveraging

depth residuals to filter out dynamic objects. DynaSLAM [2] supports RGB, RGB-D, and stereo inputs, leveraging Mask R-CNN [11] for semantic segmentation with predefined movable object classes, and detects unknown dynamic objects in RGB-D mode via multi-view geometry.

To our knowledge, no existing traditional SLAM methods support monocular input without relying on prior class information, likely due to the sparse nature of traditional monocular SLAM, which limits the use of purely geometric cues for identifying dynamic regions. Our SLAM approach, however, leverages a 3D Gaussian scene representation to provide dense mapping, enabling support for monocular input without prior semantic information.

2.2. Neural Implicit and 3DGS SLAM

Recently, Neural Implicit Representations and 3D Gaussian Splatting (3DGS) have gained substantial interest in SLAM research, as they offer promising advancements in enhancing dense reconstruction and novel view synthesis. Early SLAM systems like iMAP [45] and NICE-SLAM [68] pioneered the use of neural implicit representations, integrating mapping and camera tracking within a unified framework. Subsequent works have further advanced these methods by exploring various optimizations and extensions, including efficient representations [17, 23, 49], monocular settings [1, 64, 69], and the integration of semantic information [25, 59, 67]. The emergence of 3D Gaussian Splatting (3DGS) [20] introduces an efficient and flexible alternative representation for SLAM and has been adopted in several recent studies [9, 12, 14, 19, 26, 28, 37, 55, 66]. Among these, MonoGS [30] is the first near real-time monocular SLAM system to use 3D Gaussian Splatting as its sole scene representation. Another notable advancement is Splat-SLAM [39], the state-of-the-art (SoTA) in monocular Gaussian Splatting SLAM, offering high-accuracy mapping with robust global consistency. These methods excel in tracking and reconstruction but typically assume static scene conditions, limiting their robustness as performance degrades significantly in dynamic environments.

Some methods have focused explicitly on handling dynamic environments. Most approaches extract a dynamic object mask for each frame before passing it to the tracking and mapping components. DG-SLAM [54], DynaMon [40], and RoDyn-SLAM [16] combine segmentation masks with motion masks derived from optical flow. DDN-SLAM [27] employs object detection combined with a Gaussian Mixture Model to distinguish between foreground and background, checking feature reprojection error to enhance tracking accuracy. However, these approaches rely heavily on prior knowledge of object classes and depend on object detection or semantic segmentation, limiting their generalizability in real-world settings where dynamic objects may be unknown *a priori* and difficult to segment.

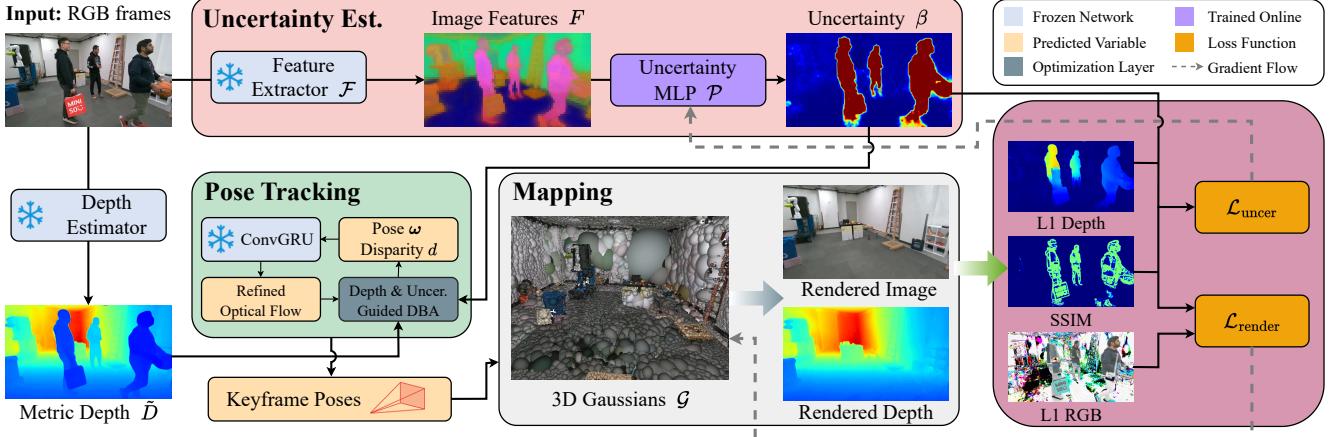


Figure 2. System Overview. WildGS-SLAM takes a sequence of RGB images as input and simultaneously estimates the camera poses while building a 3D Gaussian map \mathcal{G} of the static scene. Our method is more robust to the dynamic environment due to the uncertainty estimation module, where a pretrained DINOv2 model [57] is first used to extract the image features. An uncertainty MLP \mathcal{P} then utilizes the extracted features to predict per-pixel uncertainty. During the tracking, we leverage the predicted uncertainty as the weight in the dense bundle adjustment (DBA) layer to mitigate the impact of dynamic distractors. We further use monocular metric depth to facilitate the pose estimation. In the mapping module, the predicted uncertainty is incorporated into the rendering loss to update \mathcal{G} . Moreover, the uncertainty loss is computed in parallel to train \mathcal{P} . Note that \mathcal{P} and \mathcal{G} are optimized independently, as illustrated by the gradient flow in the gray dashed line. Faces are blurred to ensure anonymity.

In contrast, our method is purely geometric even with monocular input. While similar works such as NeRF *On-the-go* [38] and WildGaussians [24] demonstrate distractor removal in dynamic environments, they are primarily designed for sparse-view settings with known camera poses. Inspired by these approaches, we also leverage the pre-trained 2D foundation model DINOv2 [35] and use an MLP to decode them into an uncertainty map. We extend this framework to tackle the challenging sequential SLAM setting, integrating specific design components that enable robust tracking and high-fidelity mapping within our 3DGs backend.

A concurrent work, MonST3R [61], introduced a feed-forward approach for estimating scene geometry in the presence of motion. It detects moving objects by thresholding the difference between the predicted optical flow [51] and the reprojection flow, estimated using an extended version of DUSt3R [50]. However, this approach is limited to short sequences, and its use of point clouds as the scene representation does not support view synthesis.

3. Method

Given a sequence of RGB frames $\{I_i\}_{i=1}^N$ captured in a dynamic environment, WildGS-SLAM tracks the camera pose while reconstructing the static part of the scene as a 3D Gaussian map (Sec. 3.1). To mitigate the adverse impact of moving objects in tracking and eliminate them from the 3D reconstruction, we utilize DINOv2 features [35] and a shallow MLP to decode them to per-pixel uncertainty (Sec. 3.2). We further introduce how this uncertainty is integrated into the optical-flow-based tracking component (Sec. 3.3). In parallel with tracking, we leverage this uncertainty to pro-

gressively expand and optimize the 3D Gaussian map with uncertainty-aware loss functions (Sec. 3.4). The overview of WildGS-SLAM is in Fig. 2.

3.1. Preliminary on 3D Gaussian Splatting

We utilize a 3D Gaussian representation [20] to reconstruct the static part of the scanned environment. The scene is represented by a set of anisotropic Gaussians $\mathcal{G} = \{g_i\}_{i=1}^K$. Each Gaussian g_i contains color $c_i \in \mathbb{R}^3$, opacity $o_i \in [0, 1]$, mean $\mu_i \in \mathbb{R}^3$, and covariance matrix $\Sigma_i \in \mathbb{R}^{3 \times 3}$.

Rendering. We follow the same rendering approach as in the original 3DGs [20] but omit spherical harmonics to speed up optimization, as in [30, 58]. Given a camera-to-world pose ω and the projection function Π_c that maps 3D points onto the image frame, the 3D Gaussians can be "splatted" onto the 2D image plane by projecting the mean μ and covariance matrix Σ as $\mu' = \Pi_c(\omega^{-1}\mu)$ and $\Sigma' = \mathbf{J}\mathbf{R}\Sigma\mathbf{R}^T\mathbf{J}^T$, where \mathbf{J} is the Jacobian of the linear approximation of the projective transformation and \mathbf{R} is the rotation component of ω . The opacity of a Gaussian g_i at pixel x' is:

$$\alpha_i = o_i \exp\left(-\frac{1}{2}(x' - \mu'_i)^T \Sigma'^{-1} (x' - \mu'_i)\right). \quad (1)$$

The rendered color \hat{I} and depth \hat{D} at pixel x' are obtained by blending the 3D Gaussians \mathcal{G}' overlapping with this pixel, sorted by their depth relative to the camera plane:

$$\hat{I} = \sum_{i \in \mathcal{G}'} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad \hat{D} = \sum_{i \in \mathcal{G}'} \hat{d}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \quad (2)$$

where \hat{d}_i is the z-axis depth of the center of g_i . This process is fully differentiable, enabling incremental map updates as new frames are streamed (we discuss details in Sec. 3.4).

3.2. Uncertainty Prediction

WildGS-SLAM’s main contribution is to eliminate the impact of moving distractors in both mapping and tracking. To achieve this, we use an uncertainty prediction component inspired by [24, 38] while additionally incorporating our own custom depth uncertainty loss during training. For each input frame, we extract DINOv2 [35] features and utilize an uncertainty MLP, trained on-the-fly with streamed frames, to predict a per-pixel uncertainty map that mitigates the impact of distractors in both tracking and mapping.

Feed-Forward Uncertainty Estimation. Given an input image I_i , we use a pre-trained DINOv2 feature extractor \mathcal{F} to derive image features, $F_i = \mathcal{F}(I_i)$. Instead of the original DINOv2 model [35], we use the finetuned version from [57], which injects 3D awareness into the model. The features are used as input to a shallow uncertainty MLP \mathcal{P} to predict an uncertainty map $\beta_i = \mathcal{P}(F_i)$. We bilinearly upsample β_i to the original input frame resolution, which is then used in both tracking (Sec. 3.3) and mapping (Sec. 3.4).

Uncertainty Loss Function. For the uncertainty loss functions, we adopt the modified SSIM loss and two regularization terms from NeRF *On-the-go* [38], along with the L1 depth loss term:

$$\mathcal{L}_{\text{depth}} = |\hat{D}_i - \tilde{D}_i|_1, \quad (3)$$

where $\mathcal{L}_{\text{depth}}$ represents the L1 loss between the rendered depth \hat{D}_i and the metric depth \tilde{D}_i , as estimated by Metric3D v2 [13]. We find that this additional depth signal effectively improves the model’s ability to distinguish distractors, enhancing the training of the uncertainty MLP. Therefore the total uncertainty loss is:

$$\mathcal{L}_{\text{uncer}} = \frac{\mathcal{L}'_{\text{SSIM}} + \lambda_1 \mathcal{L}_{\text{uncer_D}}}{\beta_i^2} + \lambda_2 \mathcal{L}_{\text{reg_V}} + \lambda_3 \mathcal{L}_{\text{reg_U}}, \quad (4)$$

where λ_* are hyperparameters, $\mathcal{L}'_{\text{SSIM}}$ is the modified SSIM loss, $\mathcal{L}_{\text{reg_V}}$ minimizes the variance of predicted uncertainty for features having high similarity, and the last term $\mathcal{L}_{\text{reg_U}} = \log \beta_i$ prevents β_i from being infinitely large. Please refer to NeRF *On-the-go* [38] for details on $\mathcal{L}'_{\text{SSIM}}$, $\mathcal{L}_{\text{reg_V}}$, and $\mathcal{L}_{\text{reg_U}}$. We use $\mathcal{L}_{\text{uncer}}$ to train \mathcal{P} in parallel with map optimization (Sec. 3.4).

3.3. Tracking

Our tracking component is based on the recent method DROID-SLAM [47] with the incorporation of depth and uncertainty into the DBA to make the system robust in dynamic environments. The original DROID-SLAM [47] uses a pre-trained recurrent optical flow model coupled with a DBA layer to jointly optimize keyframe camera poses and disparities. This optimization is performed over a frame graph, denoted as $G = (V, E)$, where V represents the selected keyframes and E represents the edges between keyframes.

Following [39, 60], we incorporate loop closure and online global BA to reduce pose drift over long sequences.

Depth and Uncertainty Guided DBA. Different from [39, 47, 60], we integrate the uncertainty map estimated by \mathcal{P} into the BA optimization objective to deal with the moving distractors. In addition, we utilize the metric depth estimated by Metric3D V2 [13] to stabilize the DBA layer, since \mathcal{P} is trained online and can not always give accurate uncertainty estimation, especially during the early stages of tracking. For each newly inserted keyframe I_i , we first estimate its monocular metric depth \tilde{D}_i and add it to the DBA objective alongside optical flow:

$$\begin{aligned} \arg \min_{\omega, d} & \sum_{(i,j) \in E} \left\| \tilde{p}_{ij} - \Pi_c(\omega_j^{-1} \omega_i \Pi_c^{-1}(p_i, d_i)) \right\|_{\Sigma_{ij}/\beta_i^2}^2 \\ & + \lambda_4 \sum_{i \in V} \left\| M_i(d_i - 1/\tilde{D}_i) \right\|^2, \end{aligned} \quad (5)$$

The first term is the uncertainty-aware DBA objective where \tilde{p}_{ij} is the predicted pixel position of pixels p_i projected into keyframe j by the estimated optical flow; this is iteratively updated by a Convolutional Gated Recurrent Unit (ConvGRU) [47]. Π_c represents the projection from 3D points to 2D image planes, ω_i is the camera-to-world transformation for keyframe i , d_i is the optimized disparity, and $\|\cdot\|_{\Sigma_{ij}/\beta_i^2}$ is the Mahalanobis distance [31] which weighs the error terms by confidence matrix Σ_{ij} from the flow estimator [47] and our uncertainty map β_i . As a result, pixels associated with moving objects will have minimal impact on the optimization in DBA.

The second term is a disparity regularization term to encourage $1/d_i$ to be close to the predicted depth for all i in the graph nodes V . We find that this regularization term can stabilize the pose estimation, especially when the uncertainty MLP \mathcal{P} has not converged to provide reliable uncertainty β_i while moving objects are dominant in the image frames. M_i is a binary mask that deactivates disparity regularization in regions where \tilde{D}_i is unreliable, computed via multi-view depth consistency (details provided in the supplementary).

3.4. Mapping

After the tracking module predicts the pose of a newly inserted keyframe, its RGB image I , metric depth \tilde{D} , and estimated pose ω will be utilized in the mapping module to expand and optimize the 3DGS map. Given a new keyframe processed in the tracking module, we expand the Gaussian map to cover newly explored areas, using \tilde{D}_i as proxy depth, following the RGBD strategy of MonoGS [30]. Before optimization, we also actively deform the 3D Gaussian map if the poses of previous keyframes are updated by loop closure or global BA, as in Splat-SLAM [39].

Map update. After the map is expanded, we optimize the Gaussians for a fixed number of iterations. We maintain a

local window of keyframes selected by inter-frame covisibility, similar to MonoGS [30]. At each iteration, we randomly sample a keyframe with at least 50% probability evenly distributed for the keyframes in the local window, while all the other keyframes share the remaining probability equally. For a selected keyframe, we render the color \hat{I} and depth \hat{D} image by Eq. (2). The Gaussian map \mathcal{G} is optimized by minimizing the render loss $\mathcal{L}_{\text{render}}$:

$$\mathcal{L}_{\text{render}} = \frac{\lambda_5 \mathcal{L}_{\text{color}} + \lambda_6 \mathcal{L}_{\text{depth}}}{\beta^2} + \lambda_7 \mathcal{L}_{\text{iso}}, \quad (6)$$

where the color loss $\mathcal{L}_{\text{color}}$, combines L1 and SSIM losses as follows:

$$\mathcal{L}_{\text{color}} = (1 - \lambda_{\text{ssim}}) \|\hat{I} - I\|_1 + \lambda_{\text{ssim}} \mathcal{L}_{\text{ssim}}. \quad (7)$$

Unlike the loss function for static scenes, here we incorporate the uncertainty map β , which serves as a weighting factor for $\mathcal{L}_{\text{color}}$ and $\mathcal{L}_{\text{depth}}$, minimizing the influence of distractors during mapping optimization. Additionally, isotropic regularization loss \mathcal{L}_{iso} [30] constrains 3D Gaussians to prevent excessive elongation in sparsely observed regions.

At each iteration, we also compute $\mathcal{L}_{\text{uncer}}$ given the rendered color and depth image as in Eq. (4). $\mathcal{L}_{\text{uncer}}$ is then used to train the uncertainty MLP \mathcal{P} in parallel to the map optimization. As shown in [38], it is crucial to separately optimize the 3D Gaussian map and the uncertainty MLP. Therefore, we detach the gradient flow from $\mathcal{L}_{\text{uncer}}$ to the Gaussians \mathcal{G} , as well as from $\mathcal{L}_{\text{render}}$ to \mathcal{P} .

4. Experiments

4.1. Experimental Setup

Datasets. We evaluate our approach on the Bonn RGB-D Dynamic Dataset [36] and TUM RGB-D Dataset [44]. To further assess performance in unconstrained, real-world settings, we introduce the Wild-SLAM Dataset, comprising two subsets: Wild-SLAM MoCap and Wild-SLAM iPhone. The Wild-SLAM MoCap Dataset includes 10 RGB-D sequences recorded with an Intel RealSense D455 camera [15] in a room equipped with an OptiTrack [34] motion capture system, providing ground truth trajectories. The Wild-SLAM iPhone Dataset comprises 7 non-staged RGB sequences recorded with an iPhone 14 Pro. Since ground truth trajectories are not available for this dataset, it is used solely for qualitative experiments. The Wild-SLAM MoCap Dataset provides RGB-D frames at 720×1280 resolution, while the Wild-SLAM iPhone Dataset offers RGB frames at 1440×1920 . For efficiency, in our experiments, we downsample these to 360×480 and 360×640 , respectively. Dataset details are offered in supplementary.

Baselines. We compare WildGS-SLAM with the following 13 methods. (a) *Classic SLAM methods*: DSO [7], ORB-SLAM2 [32], and DROID-SLAM [47]; (b) *Classic SLAM*

methods dealing with dynamic environments: Refusion [36] and DynaSLAM [2]; (c) *Static neural implicit and 3DGs SLAM systems*: NICE-SLAM [68], MonoGS [30], and Splat-SLAM [39]; (d) *Concurrent neural implicit and 3DGs SLAM systems dealing with dynamic environments*: DG-SLAM [54], RoDyn-SLAM [16], DDN-SLAM [27], and DynaMoN [40]; (e) the very recent *feed-forward approach* MonST3R [61]; and (f) a concurrent deep SLAM framework for dynamic videos: MegaSaM [29]. To address its substantial VRAM usage (65 frames requiring 33 GB), we adapt the model by integrating a custom sliding-window inference strategy, enabling SLAM-style sequential input processing (referred to as MonST3R-SW; implementation details provided in the supplementary). We include two versions of DynaSLAM [2] in our experiments. The first, DynaSLAM (RGB), uses only monocular input without leveraging geometric information or performing inpainting. The second, DynaSLAM (N+G), utilizes RGB-D input, incorporates geometric information, and performs inpainting. Since not all methods are open-sourced and run on all sequences, we provide detailed sources for each baseline method’s metrics in supplementary.

Metrics. For camera tracking evaluation, we follow the standard monocular SLAM pipeline, aligning the estimated trajectory to the ground truth (GT) using evo [8] with Sim(3) Umeyama alignment [48], and then evaluate (ATE RMSE) [44]. Note that, although our tracking optimizations are performed solely on keyframe images, we recover camera poses for non-keyframes and evaluate the complete camera trajectory. Please refer to the supplementary for details. Additionally, we employ PSNR, SSIM [52], and LPIPS [62] metrics to evaluate the novel view synthesis quality.

4.2. Mapping, Tracking, and Rendering

Wild-SLAM Mocap Dataset. We begin our evaluation on the newly captured Wild-SLAM MoCap dataset. As shown in Table 1, our method significantly outperforms other baselines on average. The only exception is a slight increase in tracking error in the Person sequence, where the single person moves in a simple pattern, making it easier for DynaSLAM [2] to use semantic segmentation to mask out dynamic regions within the frames. All other sequences contain various types of distractors beyond humans, with different forms of occlusion. Although Refusion [36] and DynaSLAM (N+G) [2] use geometric approaches leveraging raw depth information to identify dynamic objects, they still underperform compared to our WildGS-SLAM with only monocular input. While MonST3R [61] has demonstrated strong performance on short sequences, extending it to longer sequences with a sliding window approach (MonST3R-SW), even with substantial overlap, still leads to significant tracking errors.

We further evaluate rendering results in Fig. 3, Table 2, and Fig. 4. In Fig. 3, we render from the input view to

| Method | ANYmal1 | ANYmal2 | Ball | Crowd | Person | Racket | Stones | Table1 | Table2 | Umbrella | Avg. |
|---------------------------|------------|------------|------------|------------|--------|------------|------------|------------|------------|------------|-------------|
| <i>RGB-D</i> | | | | | | | | | | | |
| Refusion [36] | 4.2 | 5.6 | 5.0 | 91.9 | 5.0 | 10.4 | 39.4 | 99.1 | 101.0 | 10.7 | 37.23 |
| DynaSLAM (N+G) [2] | 1.6 | 0.5 | 0.5 | 1.7 | 0.5 | 0.8 | 2.1 | 1.2 | 34.8 | 34.7 | 7.84 |
| NICE-SLAM [68] | F | 123.6 | 21.1 | F | 150.2 | F | 134.4 | 138.4 | F | 23.8 | - |
| <i>Monocular</i> | | | | | | | | | | | |
| DSO [7] | 12.0 | 2.5 | 1.0 | 88.6 | 9.3 | 3.1 | 41.5 | 50.6 | 85.3 | 26.0 | 32.99 |
| DROID-SLAM [47] | 0.6 | 4.7 | 1.2 | 2.3 | 0.6 | 1.5 | 3.4 | 48.0 | 95.6 | 3.8 | 16.17 |
| DynaSLAM (RGB) [2] | 0.6 | 0.5 | 0.5 | 0.5 | 0.4 | 0.6 | 1.7 | 1.8 | 42.1 | 1.2 | 5.19 |
| MonoGS [30] | 8.8 | 51.6 | 7.4 | 70.3 | 55.6 | 67.6 | 39.9 | 24.9 | 118.4 | 35.3 | 47.99 |
| Splat-SLAM [39] | 0.4 | 0.4 | 0.3 | 0.7 | 0.8 | 0.6 | 1.9 | 2.5 | 73.6 | 5.9 | 8.71 |
| MonST3R-SW [61] | 3.5 | 21.6 | 6.1 | 14.4 | 7.2 | 13.2 | 11.2 | 4.8 | 33.7 | 5.5 | 12.12 |
| MegaSAM [29] | 0.6 | 2.7 | 0.6 | 1.0 | 3.2 | 1.6 | 3.2 | 1.0 | 9.4 | 0.6 | 2.40 |
| WildGS-SLAM (Ours) | 0.2 | 0.3 | 0.2 | 0.3 | 0.8 | 0.4 | 0.3 | 0.6 | 1.3 | 0.2 | 0.46 |

Table 1. **Tracking Performance on our Wild-SLAM MoCap Dataset** (ATE RMSE ↓ [cm]). Best results are highlighted as **first**, **second**, and **third**. All baseline methods were run using their publicly available code. For DynaSLAM (RGB), initialization is time-consuming for certain sequences, and only keyframe poses are generated and evaluated. ‘F’ denotes tracking failure.

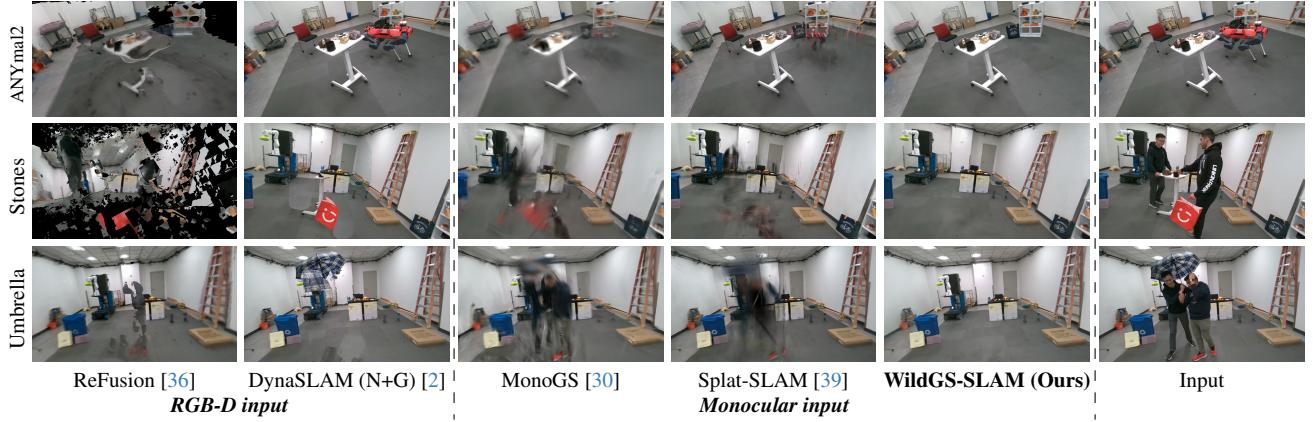


Figure 3. **Input View Synthesis Results on our Wild-SLAM MoCap Dataset.** Regardless of the distractor type, our method is able to remove distractors and render realistic images. Faces are blurred to ensure anonymity.

| | ANYmal1 | ANYmal2 | Ball | Crowd | Person | Racket | Stones | Table1 | Table2 | Umbrella | Avg. |
|---------------------------|-----------------------------|--|--|--|--|--|--|--|--|--|--|
| <i>Monocular</i> | | | | | | | | | | | |
| Splat-SLAM [39] | PSNR ↑ SSIM ↑ LPIPS ↓ | 19.71 0.786 0.313 | 20.32 0.800 0.278 | 17.68 0.702 0.294 | 16.00 0.693 0.356 | 18.58 0.754 0.298 | 16.45 0.699 0.301 | 17.90 0.711 0.291 | 17.54 0.717 0.312 | 11.45 0.458 0.650 | 16.65 0.667 0.362 |
| WildGS-SLAM (Ours) | PSNR ↑ SSIM ↑ LPIPS ↓ | 21.85 0.807 0.211 | 21.46 0.832 0.230 | 20.06 0.754 0.191 | 21.28 0.802 0.176 | 20.31 0.801 0.189 | 20.87 0.785 0.186 | 20.52 0.768 0.185 | 20.33 0.788 0.185 | 19.16 0.728 0.209 | 20.03 0.766 0.303 |
| | | | | | | | | | | | 20.59 0.783 0.210 |

Table 2. **Novel View Synthesis Evaluation on our Wild-SLAM MoCap Dataset.** Best results are in bold.

demonstrate our distractor removal capability. In comparison to other baselines, our method produces artifact-free, realistic renderings of the static scene. To evaluate novel view synthesis, we capture additional images of static scenes as part of the dataset. For each dynamic sequence, we select a subset of static scene images that match the coverage of the dynamic sequence. The results in Table 2 and Fig. 4 showcase that our method has the best novel view synthesis performance thanks to our uncertainty aware mapping.

Wild-SLAM iPhone Dataset. We further evaluate our approach on in-the-wild sequences captured using an iPhone RGB camera. Fig. 5 presents rendering results compar-

isons, along with visualizations of the uncertainty map of our method and the dynamic mask from MonST3R [61]. Our method achieves the best rendering results with an accurate uncertainty map, even able to assign higher uncertainty to the shadows of distractors. In contrast, MonST3R [61] depends heavily on the performance of pretrained models, which may lead to missed detections of entire dynamic objects. On the other hand, MegaSAM [29] leverages only neighboring frames, lacking enough multi-view information, which results in less reliable motion masks.

Bonn RGB-D Dynamic Dataset [36]. Tracking results, presented in Table 3, demonstrate that our method achieves

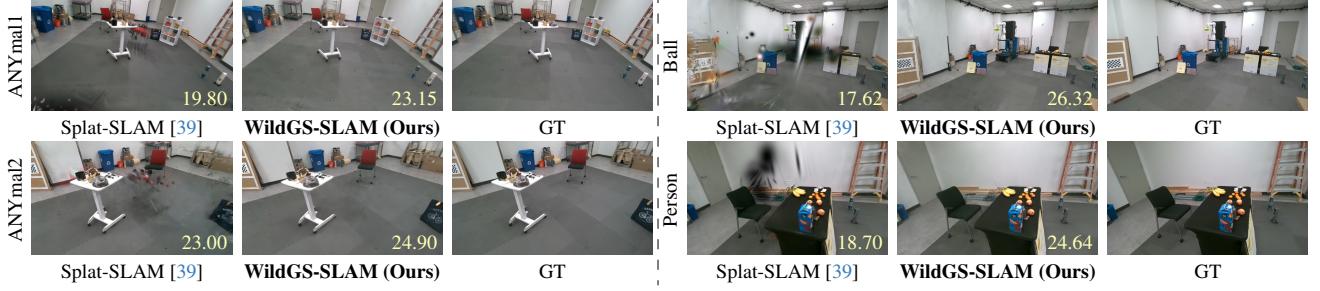


Figure 4. **Novel View Synthesis Results on our Wild-SLAM MoCap Dataset.** PSNR metrics (\uparrow) are included in images.

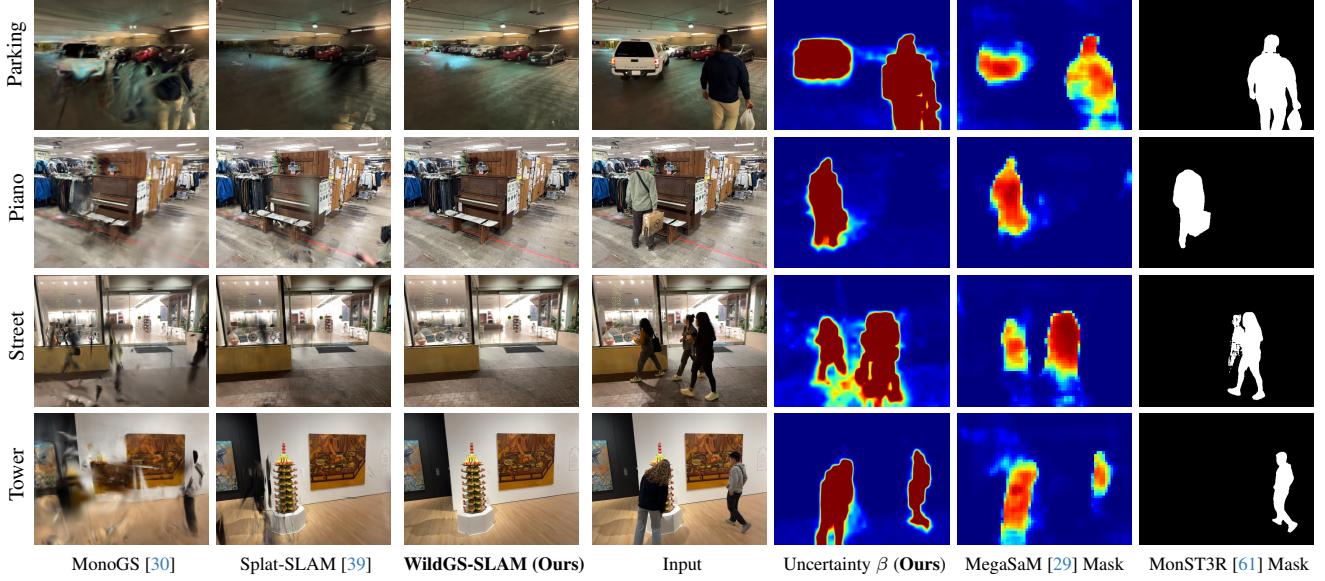


Figure 5. **Input View Synthesis Results on our Wild-SLAM iPhone Dataset.** We only show rendering results of monocular methods, as depth images are unavailable in this dataset. Note that our uncertainty map appears blurry, as DINOv2 outputs feature maps at 1/14 of the original resolution, and for mapping we also downsample to 1/3 of the original resolution, in order to maintain SLAM system efficiency. For a high-resolution, sharper uncertainty map, the resolution can be increased at the cost of some efficiency; further details and results are provided in the supplementary materials. Faces are blurred to ensure anonymity.

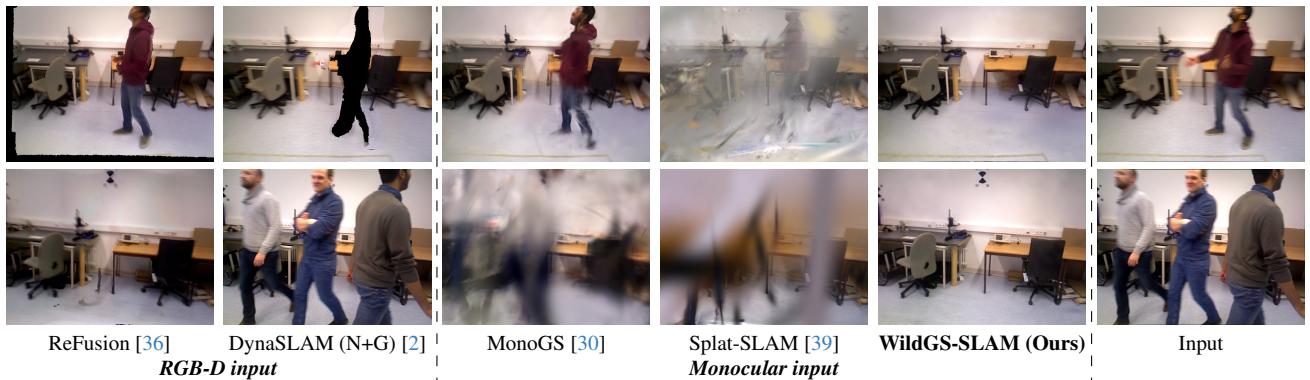


Figure 6. **View Synthesis Results on Bonn RGB-D Dynamic Dataset [36].** We show results on the Balloon (first row) and Crowd (second row) sequences. For Balloon, ReFusion [36] fails to remove the person from the TSDF, and DynaSLAM(N+G)[2] struggles with limited static information from multiple views, resulting in partial black masks. In Crowd, DynaSLAM(N+G)[2] cannot detect dynamic regions, defaulting the original image as the inpainted result. In contrast, ours achieves superior rendering even with motion blur in the input.

| Method | Balloon | Balloon2 | Crowd | Crowd2 | Person | Person2 | Moving | Moving2 | Avg. |
|---------------------------|---------|----------|-------|--------|--------|---------|--------|---------|-------------|
| <i>RGB-D</i> | | | | | | | | | |
| ReFusion [36] | 17.5 | 25.4 | 20.4 | 15.5 | 28.9 | 46.3 | 7.1 | 17.9 | 22.38 |
| ORB-SLAM2 [32] | 6.5 | 23.0 | 4.9 | 9.8 | 6.9 | 7.9 | 3.2 | 3.9 | 6.36 |
| DynaSLAM (N+G) [2] | 3.0 | 2.9 | 1.6 | 3.1 | 6.1 | 7.8 | 23.2 | 3.9 | 6.45 |
| NICE-SLAM [68] | 24.4 | 20.2 | 19.3 | 35.8 | 24.5 | 53.6 | 17.7 | 8.3 | 22.74 |
| DG-SLAM [54] | 3.7 | 4.1 | - | - | 4.5 | 6.9 | - | 3.5 | - |
| RoDyn-SLAM [16] | 7.9 | 11.5 | - | - | 14.5 | 13.8 | - | 12.3 | - |
| DDN-SLAM (RGB-D) [27] | 1.8 | 4.1 | 1.8 | 2.3 | 4.3 | 3.8 | 2.0 | 3.2 | 2.91 |
| <i>Monocular</i> | | | | | | | | | |
| DSO [7] | 7.3 | 21.8 | 10.1 | 7.6 | 30.6 | 26.5 | 4.7 | 11.2 | 15.0 |
| DROID-SLAM [47] | 7.5 | 4.1 | 5.2 | 6.5 | 4.3 | 5.4 | 2.3 | 4.0 | 4.91 |
| MonoGS [30] | 15.3 | 17.3 | 11.3 | 7.3 | 26.4 | 35.2 | 22.2 | 47.2 | 22.8 |
| Splat-SLAM [39] | 8.8 | 3.0 | 6.8 | F | 4.9 | 25.8 | 1.7 | 3.0 | - |
| DynaMoN (MS) [40] | 6.8 | 3.8 | 6.1 | 5.6 | 2.4 | 3.5 | 1.4 | 2.6 | 4.02 |
| DynaMoN (MS&SS) [40] | 2.8 | 2.7 | 3.5 | 2.8 | 14.8 | 2.2 | 1.3 | 2.7 | 4.10 |
| MonST3R-SW [61] | 5.4 | 7.2 | 5.4 | 6.9 | 11.9 | 11.1 | 3.3 | 7.4 | 7.3 |
| MegaSaM [29] | 3.7 | 2.6 | 1.6 | 7.2 | 4.1 | 4.0 | 1.4 | 3.4 | 3.51 |
| WildGS-SLAM (Ours) | 2.8 | 2.4 | 1.5 | 2.3 | 3.1 | 2.7 | 1.6 | 2.2 | 2.31 |

Table 3. **Tracking Performance on Bonn RGB-D Dynamic Dataset** [36] (ATE RMSE ↓ [cm]). DDN-SLAM [27] is not open source and does not report its RGB mode results on this dataset. DynaSLAM (RGB) [2] consistently fails to initialize or experiences extended tracking loss across all sequences and therefore cannot be included in the table. ‘F’ indicates failure.

| Method | f3/ws | f3/wx | f3/wr | f3/whs | Avg. |
|---------------------------|-------|-------|-------|--------|--------|
| <i>RGB-D</i> | | | | | |
| ReFusion [36] | 1.7 | 9.9 | 40.6* | 10.4 | 15.7* |
| ORB-SLAM2 [32] | 40.8 | 72.2 | 80.5 | 72.3 | 66.45 |
| DynaSLAM (N+G) [2] | 0.6 | 1.5 | 3.5 | 2.5 | 2.03 |
| NICE-SLAM [68] | 79.8 | 86.5 | 244.0 | 152.0 | 140.57 |
| DG-SLAM [54] | 0.6 | 1.6 | 4.3 | - | - |
| RoDyn-SLAM [16] | 1.7 | 8.3 | - | 5.6 | - |
| DDN-SLAM (RGB-D) [27] | 1.0 | 1.4 | 3.9 | 2.3 | 2.15 |
| <i>Monocular</i> | | | | | |
| DSO [7] | 1.5 | 12.9 | 13.8 | 40.7 | 17.23 |
| DROID-SLAM [47] | 1.2 | 1.6 | 4.0 | 2.2 | 2.25 |
| MonoGS [30] | 1.1 | 21.5 | 17.4 | 44.2 | 21.05 |
| Splat-SLAM [39] | 2.3 | 1.3 | 3.9 | 2.2 | 2.43 |
| DynaMoN (MS) | 1.4 | 1.4 | 3.9 | 2.0 | 2.18 |
| DynaMoN (MS&SS) [40] | 0.7 | 1.4 | 3.9 | 1.9 | 1.98 |
| DDN-SLAM (RGB) [27] | 2.5 | 2.8 | 8.9 | 4.1 | 4.58 |
| MonST3R-SW [61] | 2.2 | 27.3 | 13.6 | 19.8 | 15.73 |
| MegaSaM [29] | 0.6 | 1.5 | 2.6 | 1.8 | 1.63 |
| WildGS-SLAM (Ours) | 0.4 | 1.3 | 3.3 | 1.6 | 1.63 |

Table 4. **Tracking Performance on TUM RGB-D Dataset** [44] (ATE RMSE ↓ [cm]). We present sequences with higher dynamics here; see the supplementary materials for tracking results on other dynamic sequences. For methods without complete scene coverage in the original reports, results obtained by running their open-source code are marked with ‘*’. If open-source code is unavailable, scenes without results are marked with ‘-’. DynaSLAM (RGB) [2] consistently fails to initialize or experiences extended tracking loss across all sequences and therefore cannot be included in this table.

the highest overall performance, underscoring its robustness. In contrast, DynaMoN [40] first performs initial camera tracking and then refines it offline, which is time-consuming. Meanwhile, DynaSLAM (N+G) [2] and DDN-SLAM [27] rely on depth sensor data combined with semantic segmentation. In Fig. 6, we show rendered images from the input view. Our method successfully removes dynamic objects while achieving realistic rendering with minimal artifacts.

TUM RGB-D Dataset [44]. Our tracking method achieves the best performance on all sequences, as shown in Table 4. Rendering results are included in the supplementary.

| | Wild-SLAM | Bonn | TUM |
|-----------------------------------|-------------|-------------|-------------|
| (a) w/o Uncertainty Mask β | 3.89 | 5.11 | 1.91 |
| (b) w/o L1 Depth Loss in Eq. (4) | 0.50 | 2.37 | 1.83 |
| (c) YOLOv8 + SAM Mask | 3.06 | 2.37 | 1.65 |
| (d) w/o Disparity Reg. in Eq. (5) | 10.97 | F | 2.9 |
| WildGS-SLAM (Ours) | 0.46 | 2.31 | 1.63 |

Table 5. **WildGS-SLAM Ablation Study** (ATE RMSE ↓ [cm]). For each dataset, we report the average tracking error. ‘F’ indicates that the method fails on at least one sequence within that dataset.

4.3. Ablation Study

We ablate our key design choices in Table 5. For (c), we pass predefined distractor types to YOLOv8* to generate bounding boxes, then apply the Segment Anything Model (SAM) [21] for segmentation within each box. It achieves similar results on Bonn and TUM datasets, as their distractors are primarily humans. Our method outperforms all other variants, confirming the effectiveness of our design choices.

5. Conclusion

In this paper, we introduced WildGS-SLAM, a novel SLAM approach designed to handle dynamic environments through a purely geometric framework. By leveraging a shallow MLP to predict per-pixel uncertainty based on pre-trained 3D-aware features, our method efficiently isolates static and dynamic scene elements, enabling robust tracking and rendering. Through extensive evaluations on both newly collected and existing datasets, we demonstrated that WildGS-SLAM achieves state-of-the-art performance in dynamic SLAM tasks, excelling in both tracking and novel view synthesis.

Limitation. Our method’s uncertainty predictor is trained on-the-fly with input frames, making it challenging to recognize distractors when a limited number of views capture the same regions. Introducing motion priors could improve handling of dynamic scenes and enhance tracking robustness.

*<https://github.com/ultralytics/ultralytics.git>

Acknowledgements. The authors thank Sayan Deb Sarkar, Tao Sun, Ata Celen, Liyuan Zhu, Emily Steiner, Jikai Jin, Yiming Zhao, Matt VanCleave, Tess Ruby Horowitz Buckley, Stanley Wang for their help in Wild-SLAM data collection. We thank Aleesa Pitchamarn Alexander for granting permission to release the data collected from the [Spirit House](#) exhibition. We also thank Aleesa Pitchamarn Alexander, Robert M. and Ruth L. Halperin for curating the exhibition, as well as all the participating artists, particularly Dominique Fung, Stephanie H. Shih, and Tammy Nguyen whose art works are prominently captured in the video data.

References

- [1] Thomas Belos, Pascal Monasse, and Eva Dokladalova. Mod slam: Mixed method for a more robust slam without loop closing. In *VISAPP*, 2022. [2](#)
- [2] Berta Bescos, José M. Fácil, Javier Civera, and José Neira. DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes. *IEEE Robotics and Automation Letters (RA-L)*, 2018. [2, 5, 6, 7, 8, 3, 4](#)
- [3] Jiyu Cheng, Yuxiang Sun, and Max Q-H Meng. Improving monocular visual slam in dynamic environments: an optical-flow-based approach. *Advanced Robotics*, 2019. [2](#)
- [4] Shuhong Cheng, Changhe Sun, Shijun Zhang, and Dianfan Zhang. Sg-slam: A real-time rgb-d visual slam toward dynamic scenes with semantic and geometric information. *IEEE Transactions on Instrumentation and Measurement*, 2022. [1, 2](#)
- [5] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *ACM Trans. on Graphics*, 1996. [2](#)
- [6] Alexander Duda and Udo Frese. Accurate detection and localization of checkerboard corners for calibration. In *BMVC*, 2018. [1](#)
- [7] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2017. [1, 2, 5, 6, 8, 3, 4](#)
- [8] Michael Grupp. evo: Python package for the evaluation of odometry and slam. <https://github.com/MichaelGrupp/evo>, 2017. [5](#)
- [9] Seongbo Ha, Jiung Yeon, and Hyeonwoo Yu. Rgbd gs-icp slam. *arXiv preprint arXiv:2403.12550*, 2024. [2](#)
- [10] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International journal of computer vision*, 103:267–305, 2013. [1](#)
- [11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. [2](#)
- [12] Jiarui Hu, Xianhao Chen, Boyin Feng, Guanglin Li, Liangjing Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Cg-slam: Efficient dense rgb-d slam in a consistent uncertainty-aware 3d gaussian field. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2024. [2](#)
- [13] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *arXiv preprint arXiv:2404.15506*, 2024. [4, 6](#)
- [14] 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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. [2](#)
- [15] Intel RealSense. Intel® RealSense™ Depth Camera D455, 2024. [5, 1](#)
- [16] Haochen Jiang, Yueming Xu, Kejie Li, Jianfeng Feng, and Li Zhang. Rodyn-slam: Robust dynamic dense rgb-d slam with neural radiance fields. *IEEE Robotics and Automation Letters (RA-L)*, 2024. [2, 5, 8, 3, 4](#)
- [17] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. *arXiv preprint arXiv:2211.11704*, 2022. [2](#)
- [18] Masaya Kaneko, Kazuya Iwami, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Mask-slam: Robust feature-based monocular slam by masking using semantic segmentation. In *CVPR Workshops*, 2018. [2](#)
- [19] 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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. [2](#)
- [20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. on Graphics*, 2023. [2, 3](#)
- [21] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023. [8](#)
- [22] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, 2007. [2](#)
- [23] Evgenii Kruzhkov, Alena Savinykh, Pavel Karpyshев, Mikhail Kurenkov, Evgeny Yudin, Andrei Potapov, and Dzmitry Tsetserukou. Meslam: Memory efficient slam based on neural fields. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2022. [2](#)
- [24] Jonas Kulhanek, Songyou Peng, Zuzana Kukelova, Marc Pollefeys, and Torsten Sattler. Wildgaussians: 3d gaussian splatting in the wild. In *Advances in Neural Information Processing Systems (NIPS)*, 2024. [1, 3, 4](#)
- [25] Kunyi Li, Michael Niemeyer, Nassir Navab, and Federico Tombari. Dns slam: Dense neural semantic-informed slam. *arXiv preprint arXiv:2312.00204*, 2023. [2](#)
- [26] Linfei Li, Lin Zhang, Zhong Wang, and Ying Shen. Gs3lam: Gaussian semantic splatting slam. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3019–3027, 2024. [2](#)
- [27] Mingrui Li, Jiaming He, Guangan Jiang, and Hongyu Wang. Ddn-slam: Real-time dense dynamic neural implicit slam with joint semantic encoding. *arXiv preprint arXiv:2401.01545*, 2024. [2, 5, 8, 3, 4](#)

- [28] Mingrui Li, Shuhong Liu, Heng Zhou, Guohao Zhu, Na Cheng, Tianchen Deng, and Hongyu Wang. Sgs-slam: Semantic gaussian splatting for neural dense slam. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2024. 2
- [29] Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. Megasam: Accurate, fast, and robust structure and motion from casual dynamic videos. *arXiv preprint arXiv:2412.04463*, 2024. 5, 6, 7, 8
- [30] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 3, 4, 5, 6, 7, 8
- [31] Goeffrey J McLachlan. Mahalanobis distance. *Resonance*, 1999. 4
- [32] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgbd cameras. *IEEE transactions on robotics*, 2017. 2, 5, 8, 3, 4
- [33] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 2015. 1, 2
- [34] OptiTrack. Optitrack - motion capture systems. 5, 1
- [35] Maxime Oquab, Timothée Darcret, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 3, 4, 6
- [36] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2019. 1, 2, 5, 6, 7, 8, 3, 4
- [37] Zhexi Peng, Tianjia Shao, Yong Liu, Jingke Zhou, Yin Yang, Jingdong Wang, and Kun Zhou. Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting. In *SIGGRAPH 2024 Conference Papers*, 2024. 2
- [38] Weining Ren, Zihan Zhu, Boyang Sun, Jiaqi Chen, Marc Pollefeys, and Songyou Peng. Nerf on-the-go: Exploiting uncertainty for distractor-free nerfs in the wild. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 3, 4, 5
- [39] Erik Sandström, Keisuke Tateno, Michael Oechsle, Michael Niemeyer, Luc Van Gool, Martin R Oswald, and Federico Tombari. Splat-slam: Globally optimized rgbd slam with 3d gaussians. *arXiv preprint arXiv:2405.16544*, 2024. 2, 4, 5, 6, 7, 8, 3
- [40] Nicolas Schischka, Hannah Schieber, Mert Asim Karaoglu, Melih Görgülü, Florian Grötzner, Alexander Ladikos, Daniel Roth, Nassir Navab, and Benjamin Busam. Dynamon: Motion-aware fast and robust camera localization for dynamic neural radiance fields. *arXiv e-prints*, pages arXiv–2309, 2023. 2, 5, 8, 3, 4
- [41] Raluca Scona, Mariano Jaimez, Yvan R Petillot, Maurice Fallon, and Daniel Cremers. Staticfusion: Background reconstruction for dense rgbd slam in dynamic environments. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2018. 2
- [42] Rulin Shen, Kang Wu, and Zhi Lin. Robust visual slam in dynamic environment based on motion detection and segmentation. *Journal of Autonomous Vehicles and Systems*, 2024. 2
- [43] João Carlos Virgolino Soares, Marcelo Gattass, and Marco Antonio Meggiolaro. Crowd-slam: visual slam towards crowded environments using object detection. *Journal of Intelligent & Robotic Systems*, 2021. 2
- [44] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgbd slam systems. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2012. 5, 8, 3, 4
- [45] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2
- [46] Yuxiang Sun, Ming Liu, and Max Q-H Meng. Motion removal for reliable rgbd slam in dynamic environments. *Robotics and Autonomous Systems*, 2018. 2
- [47] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgbd cameras. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1, 4, 5, 6, 8, 2, 3
- [48] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 1991. 5, 3
- [49] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
- [50] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3
- [51] Yihan Wang, Lahav Lipson, and Jia Deng. Sea-raft: Simple, efficient, accurate raft for optical flow. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2024. 3
- [52] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing (TIP)*, 2004. 5
- [53] Wenxin Wu, Liang Guo, Hongli Gao, Zhichao You, Yuekai Liu, and Zhiqiang Chen. Yolo-slam: A semantic slam system towards dynamic environment with geometric constraint. *Neural Computing and Applications*, 2022. 2
- [54] Yueming Xu, Haochen Jiang, Zhongyang Xiao, Jianfeng Feng, and Li Zhang. DG-SLAM: Robust Dynamic Gaussian Splatting SLAM with Hybrid Pose Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 1, 2, 5, 8, 3, 4
- [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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2
- [56] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing

- the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10371–10381, 2024. 6
- [57] Yuanwen Yue, Anurag Das, Francis Engelmann, Siyu Tang, and Jan Eric Lenssen. Improving 2d feature representations by 3d-aware fine-tuning. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2024. 2, 3, 4, 6
 - [58] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting. *arXiv preprint arXiv:2312.10070*, 2023. 3
 - [59] Hongjia Zhai, Gan Huang, Qirui Hu, Guanglin Li, Hujun Bao, and Guofeng Zhang. Nis-slam: Neural implicit semantic rgbd slam for 3d consistent scene understanding. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2
 - [60] Ganlin Zhang, Erik Sandström, Youmin Zhang, Manthan Patel, Luc Van Gool, and Martin R Oswald. Glorie-slam: Globally optimized rgb-only implicit encoding point cloud slam. *arXiv preprint arXiv:2403.19549*, 2024. 4
 - [61] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. *arXiv preprint arxiv:2410.03825*, 2024. 3, 5, 6, 7, 8, 4
 - [62] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5
 - [63] Tianwei Zhang, Huayan Zhang, Yang Li, Yoshihiko Nakamura, and Lei Zhang. Flowfusion: Dynamic dense rgbd slam based on optical flow. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2020. 2
 - [64] Wei Zhang, Tiecheng Sun, Sen Wang, Qing Cheng, and Norbert Haala. Hi-slam: Monocular real-time dense mapping with hybrid implicit fields. *IEEE Robotics and Automation Letters*, 2023. 2
 - [65] Yiming Zhao, Taein Kwon, Paul Streli, Marc Pollefeys, and Christian Holz. Egopressure: A dataset for hand pressure and pose estimation in egocentric vision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1
 - [66] Liyuan Zhu, Yue Li, Erik Sandström, Konrad Schindler, and Iro Armeni. Loopsplat: Loop closure by registering 3d gaussian splats. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2025. 2
 - [67] Siting Zhu, Guangming Wang, Hermann Blum, Jiuming Liu, Liang Song, Marc Pollefeys, and Hesheng Wang. Sni-slam: Semantic neural implicit slam. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2
 - [68] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 5, 6, 8, 3, 4
 - [69] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicerslam: Neural implicit scene encoding for rgbd slam. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2024. 2

WildGS-SLAM: Monocular Gaussian Splatting SLAM in Dynamic Environments

Supplementary Material

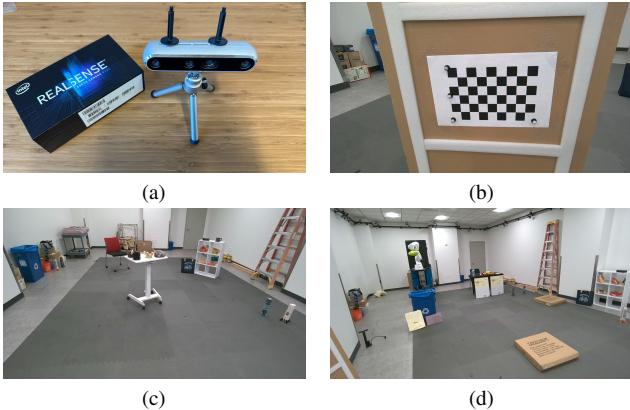


Figure 7. (a) Intel RealSense D455 camera [15]. (b) Calibration board used to align camera reference frame with OptiTrack’s rigid body frame. (c) Static scene 1. (d) Static scene 2.

Abstract

In the supplementary material, we provide additional details about the following:

1. More information about the Wild-SLAM dataset (Sec. 6).
2. Implementation details of WildGS-SLAM and baseline methods (Sec. 7).
3. Additional results and ablations (Sec. 8).

6. Wild-SLAM Dataset

Wild-SLAM MoCap Dataset. This dataset comprises a total of 10 sequences of RGB-D frames featuring various moving objects as distractors, specifically designed for dynamic SLAM benchmarking. Although WildGS-SLAM works with monocular inputs, aligned depth images are included to support the evaluation of other RGB-D baselines or future research. The RGB-D frames were captured using an Intel RealSense D455 camera (Fig. 7a) at a resolution of 720×1280 and a frame rate of 30 fps. All image sequences in this dataset were recorded with a fixed exposure time. The dataset includes two distinct static environment layouts: 2 sequences were captured in one static scene (Fig. 7c), and the remaining 8 sequences were recorded in the other (Fig. 7d). A summary of each sequence is provided in Table 6, while all sequences are presented in the video. The room used for dataset collection was equipped with an OptiTrack motion capture (MoCap) system [34], consisting of 32 OptiTrack PrimeX-13 cameras, to provide the ground truth camera poses. The OptiTrack system operates at 120 fps.

Inspired by [65], to synchronize the OptiTrack system

with the Intel RealSense D455 camera, we positioned the RealSense camera and one of the OptiTrack cameras to observe an iPhone. By switching the iPhone flashlight on and off, we identified the corresponding timestamps in the images captured by both devices that reflected the flashlight’s state change. This allowed us to get the timestamp offset between the two devices. To improve synchronization accuracy, the frame rate of the D455 was increased to 60 fps. We switched the flashlight on and off before and after each sequence recording, obtaining four timestamp offset values. The timestamp offset per recording was calculated as their average. Across all sequences, the average standard deviation among the four timestamp offset values was 5.25 ms, while the time interval between consecutive frames in the captured sequence is 33.33 ms, highlighting the precision of the synchronization.

To track the pose of the D455 camera using the MoCap system, we attached four reflective markers to the camera, defining a rigid body. We then performed a calibration procedure using a calibration board (Fig. 7b) to determine the relative transformation between the MoCap coordinate system and the camera coordinate system for this rigid body. Four reflective markers were carefully placed on the calibration board, enabling the MoCap system to track their 3D positions. These positions were then utilized to compute the locations of the grid corners on the board. Meanwhile, the corresponding 2D pixel coordinates of these grid corners in the camera frame were identified using the method described in [6]. Using this information, the camera poses in the MoCap coordinate system were determined, allowing us to compute the transformation between the rigid body and the camera frame. The calibration process was repeated 19 times, with the camera and the calibration board positioned in different poses for each trial, resulting in 19 transformation matrices. The final transformation between the rigid body and the camera frame was computed by averaging the results across all trials. Specifically, the rotation component was averaged using the chordal L_2 method [10]. The average deviation between an individual estimated transformation matrix and the final averaged transformation is 0.44° for rotation and 0.24 cm for translation.

Wild-SLAM iPhone Dataset. To further assess performance in more unconstrained, real-world scenarios, we captured 7 sequences using an iPhone 14 Pro. These sequences comprise 4 outdoor and 3 indoor scenes, showcasing a variety of daily-life activities such as strolling along streets, shopping, navigating a parking garage, and exploring an art museum. Each sequence provides RGB images at a resolution of 1920×1280 , accompanied by LiDAR depth images at 256×192 .

| Sequence Name | Distractors | Static Environment | Number of Frames | Length of Trajectory [m] |
|---------------|-----------------------------------|--------------------|------------------|--------------------------|
| ANYmal1 | ANYmal Robot | Scene 1 | 651 | 7.274 |
| ANYmal2 | ANYmal Robot | Scene 1 | 1210 | 11.567 |
| Ball | Human, Basketball | Scene 2 | 931 | 11.759 |
| Crowd | Human, Basketball, Bag | Scene 2 | 1268 | 14.189 |
| Person | Human | Scene 2 | 986 | 10.354 |
| Racket | Human, Racket | Scene 2 | 962 | 12.421 |
| Stones | Human, Table, Bag, Gripper, Stone | Scene 2 | 962 | 12.421 |
| Table1 | Human, Table, Gripper, Stone | Scene 2 | 561 | 6.592 |
| Table2 | Human, Table, Gripper, Stone | Scene 2 | 1029 | 11.184 |
| Umbrella | Human, Umbrella | Scene 2 | 458 | 4.499 |

Table 6. Overview of our WildGS-SLAM MoCap Dataset.

resolution. While WildGS-SLAM only requires monocular inputs, the inclusion of LiDAR data facilitates the evaluation of RGB-D baselines and future research. All sequences are showcased in the supplementary video.

Discussion. Both datasets capture humans performing activities. The *Wild-SLAM MoCap Dataset* was recorded in controlled environments, with explicit consent obtained from all participants for publishing, presenting, and sharing the data with the research community. In contrast, the *Wild-SLAM iPhone Dataset* was captured in more unconstrained settings, where we had less control over the presence of bystanders in the scene. While consent was obtained from the primary individuals featured, additional people may occasionally appear in the background. In most cases, these individuals are positioned too far from the camera to be identifiable (occupying very few pixels). Additionally, in the Parking sequence, certain car license plates are visible. To ensure privacy, all sensitive regions, including faces and license plates, have been masked in the data. It is important to note that recordings were conducted in locations where capturing people in public spaces is legally permitted, provided the footage does not target individuals in a way that could be considered intrusive or harassing.

7. Implementation Details

7.1. WildGS-SLAM

Two-Stage Initialization. We use the first 12 keyframes to run the DBA layer for tracking initialization. However, the uncertainty MLP \mathcal{P} has not yet been trained to identify uncertain regions. Hence, we deactivate the uncertainty weight β in Eq. (5) during the first stage of initialization to obtain coarse camera poses. These initial poses are used for map initialization and training of \mathcal{P} . Subsequently, we perform a reduced number of iterations in the DBA layer, with uncertainty weighting activated, to refine the coarse keyframe camera poses from the first stage.

Frame Graph Management. We manage the frame graph as in [47] but enforce the insertion of a new keyframe every 8

frames, independent of the criterion in [47] (average optical flow to the last keyframe larger than a threshold).

Disparity Regularization Mask M . For each newly inserted keyframe i , we project each of its connected keyframes j in the frame graph, i.e., $(i, j) \in E$, onto i using the metric depth \tilde{D}_j and calculate the multi-view depth consistency count as:

$$n_i(u, v) = \sum_{j|(i,j) \in E} \mathbb{1} \left(\frac{|\tilde{D}_i(u, v) - \tilde{D}_{j \rightarrow i}(u', v')|}{\tilde{D}_{j \rightarrow i}(u', v')} < \epsilon \right. \\ \left. \wedge \cos(F_i(u, v), F_j(u', v')) > \gamma \right) \quad (8)$$

where $\mathbb{1}(\cdot)$ is the indicator function, (u', v') is the pixel coordinate in j th frame that falls to (u, v) when re-projected to frame i using \tilde{D}_j , ω_i and ω_j , $\tilde{D}_{j \rightarrow i}(u', v')$ is the projected depth from point (u', v') to frame i , and ϵ is the relative depth threshold. The second condition is to filter out incorrect correspondences that have lower than a threshold γ DINO feature cosine similarity. The depth mask $M_i(u, v)$ is set to 0 if (u, v) has more than one valid correspondence in neighboring frames and $n_i(u, v)$ is less than a threshold.

Final Global BA. After processing all the input frames, we incorporate a final global Bundle Adjustment (BA) module, similar to DROID-SLAM [47], to refine the keyframe poses. The frame graph construction follows the same approach as DROID-SLAM [47]. For the DBA objective during tracking, we retain only the first term of Eq. (5), omitting the disparity regularization term, as sufficient multiview information is already available, and the uncertainty map has converged to a stable state. We include an ablation study in Sec. 8.

Final Map Refinement. After the final global BA, we perform a final refinement of the map using all keyframes, following the same strategy as Splat-SLAM [39] and MonoGS [30]. In the final refinement, we fix the keyframe poses and optimize both the uncertainty MLP and 3D Gaussian map using Eq. (6).

Obtaining Non-keyframe Pose. After completing the final global BA and map refinement, we conduct a motion-only bundle adjustment to estimate non-keyframe poses, similar to the approach in DROID-SLAM [47]. During this optimi-

| Method | Input Type | Dynamic | Open Source | Prior Free | Scene Representation |
|--|------------|---------|-------------|------------|-----------------------|
| <i>Classic SLAM methods</i> | | | | | |
| DSO [7] | RGB | ✗ | ✓ | ✓ | Sparse Point Cloud |
| ORB-SLAM2 [32] | RGB | ✗ | ✓ | ✓ | Sparse Point Cloud |
| DROID-SLAM [47] | RGB | ✗ | ✓ | ✓ | - |
| <i>Classic SLAM methods with dynamic environment handling</i> | | | | | |
| Refusion [36] | RGB-D | ✓ | ✓ | ✓ | TSDF |
| DynaSLAM (RGB) [2] | RGB | ✓ | ✓ | ✗(S) | Sparse Point Cloud |
| DynaSLAM (N+G) [2] | RGB-D | ✓ | ✓ | ✗(S) | Sparse Point Cloud |
| <i>Static neural implicit and 3DGS SLAM methods</i> | | | | | |
| NICE-SLAM [68] | RGB-D | ✗ | ✓ | ✓ | Neural Implicit |
| MonoGS [30] | RGB/RGB-D | ✗ | ✓ | ✓ | 3D Gaussian Splatting |
| SplatSLAM [39] | RGB | ✗ | ✓ | ✓ | 3D Gaussian Splatting |
| <i>Concurrent neural implicit and 3DGS SLAM methods for dynamic scenes</i> | | | | | |
| DG-SLAM [54] | RGB-D | ✓ | ✗ | ✗(S) | 3D Gaussian Splatting |
| RoDyn-SLAM [16] | RGB-D | ✓ | ✗ | ✗(S) | Neural Implicit |
| DDN-SLAM [27] | RGB/RGB-D | ✓ | ✗ | ✗(O) | Neural Implicit |
| DynaMoN (MS) [40] | RGB | ✓ | ✓ | ✓ | Neural Implicit |
| DynaMoN (MS & SS) [40] | RGB | ✓ | ✓ | ✗(S) | Neural Implicit |
| <i>Recent feed-forward methods</i> | | | | | |
| MonST3R [61] | RGB | ✓ | ✓ | ✓ | Dense Point cloud |
| WildGS-SLAM (Ours) | RGB | ✓ | ✓* | ✓ | 3D Gaussian Splatting |

Table 7. **Overview of Baseline Methods.** ‘Dynamic’ indicates whether the method explicitly addresses dynamic scenes. ‘Open Source’ specifies if a public implementation is available. ‘Prior Free’ refers to not using class priors, where ‘O’ represents object detection and ‘S’ denotes semantic segmentation. In all our experiments, we employ the RGB mode of MonoGS [30].

mization, we also deactivate the disparity regularization term in Eq. (5). These poses are further refined using an L1 RGB re-rendering loss, as employed in MonoGS [30], weighted by the uncertainty map.

7.2. Baseline Details

The characteristics of all baseline methods are presented in Table 7. Here we detail the source of each baseline method’s results tabulated in the main paper and include the implementation details of MonST3R-SW (our sliding window extension of MonST3R [61]).

Details for Table 3. For tracking performance on the Bonn RGB-D Dynamic Dataset [36], results for ORB-SLAM2 [32], NICE-SLAM [68], and DDN-SLAM [27] are taken from the DDN-SLAM [27] paper. Results for DROID-SLAM [47] are taken from the DynaMoN [40] paper. Results for DynaSLAM (N+G) [2] and ReFusion [36] are taken from the ReFusion [36] paper. DG-SLAM [54], RoDyn-SLAM [16], and DynaMoN [40] are not open-sourced by the time of submission, therefore we take results from their own paper. The results for DSO [7], MonoGS [30], Splat-SLAM [39], and MonST3R-SW [61] are obtained by running their open-source implementation.

Details for Table 4. For tracking performance on the

TUM RGB-D Dataset [44], results for Refusion [36], DG-SLAM [54], DynaSLAM (N+G) [2], RoDyn-SLAM [16], and DDN-SLAM [27] are based on data reported in their respective papers. Results for ORB-SLAM2 [32], DROID-SLAM [47], and DynaMoN [40] are sourced from the DynaMoN [40] paper. For DSO [7], NICE-SLAM [68], MonoGS [30], Splat-SLAM [39], and MonST3R-SW [61] results were obtained by running their open-source code.

MonST3R-SW. High VRAM usage is required for MonST3R [61], making it impractical to process an entire sequence as input. Instead, we apply a sliding window approach, merging overlapping frames from consecutive windows to form a complete sequence. Specifically, we use a window of 30 frames with a stride of 3, as in the original paper, and maintain an overlap of 25 frames to ensure consistent alignment. We employ Sim(3) Umeyama alignment [48] to integrate each new window’s trajectory with the global trajectory.

8. Additional Experiments

Time Analysis. Table 9 presents the average fps of our method and the baselines. We also provide a fast version to support more efficient processing with minimal loss of accuracy by disabling low-impact processes and reducing



Figure 8. **Input View Synthesis Results on TUM RGB-D Dataset [44]**. We show results on the `freiburg3_walking_static` (first row) and `freiburg3_walking_xyz` (second row) sequences. Our method produces substantially better rendering results.

| Method | f2/dp | f3/ss | fr3/sx | f3/sr | f3/shs | f3/ws | f3/wx | f3/wr | f3/whs | Avg. |
|---------------------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|
| <i>RGB-D</i> | | | | | | | | | | |
| Refusion [36] | 4.9* | 0.9 | 4.0 | 13.2* | 11.0 | 1.7 | 9.9 | 40.6* | 10.4 | 10.73* |
| ORB-SLAM2 [32] | 0.6 | 0.8 | 1.0 | 2.5 | 2.5 | 40.8 | 72.2 | 80.5 | 72.3 | 30.4 |
| DynaSLAM (N+G) [2] | 0.7* | 0.5* | 1.5 | 2.7* | 1.7 | 0.6 | 1.5 | 3.5 | 2.5 | 1.7* |
| NICE-SLAM [68] | 88.8 | 1.6 | 32.0 | 59.1 | 8.6 | 79.8 | 86.5 | 244.0 | 152.0 | 83.6 |
| DG-SLAM [54] | 3.2 | - | 1.0 | - | - | 0.6 | 1.6 | 4.3 | - | - |
| RoDyn-SLAM [16] | - | - | - | - | 4.4 | 1.7 | 8.3 | - | 5.6 | - |
| DDN-SLAM (RGB-D) [27] | - | - | 1.0 | - | 1.7 | 1.0 | 1.4 | 3.9 | 2.3 | - |
| <i>Monocular</i> | | | | | | | | | | |
| DSO [7] | 2.2 | 1.7 | 11.5 | 3.7 | 12.4 | 1.5 | 12.9 | 13.8 | 40.7 | 11.1 |
| DROID-SLAM [47] | 0.6 | 0.5 | 0.9 | 2.2 | 1.4 | 1.2 | 1.6 | 4.0 | 2.2 | 1.62 |
| MonoGS [30] | 112.8 | 1.2 | 6.1 | 5.1 | 28.3 | 1.1 | 21.5 | 17.4 | 44.2 | 26.4 |
| Splat-SLAM [39] | 0.7 | 0.5 | 0.9 | 2.3 | 1.5 | 2.3 | 1.3 | 3.9 | 2.2 | 1.71 |
| DynaMoN (MS) [40] | 0.6 | 0.5 | 0.9 | 2.1 | 1.9 | 1.4 | 1.4 | 3.9 | 2.0 | 1.63 |
| DynaMoN (MS&SS) [40] | 0.7 | 0.5 | 0.9 | 2.4 | 2.3 | 0.7 | 1.4 | 3.9 | 1.9 | 1.63 |
| DDN-SLAM (RGB) [27] | - | - | 1.3 | - | 3.1 | 2.5 | 2.8 | 8.9 | 4.1 | - |
| MonST3R-SW [61] | 51.6 | 2.4 | 28.2 | 5.4 | 36.5 | 2.2 | 27.3 | 13.6 | 19.8 | 20.8 |
| WildGS-SLAM (Ours) | 1.4 | 0.5 | 0.8 | 2.4 | 2.0 | 0.4 | 1.3 | 3.3 | 1.6 | 1.51 |

Table 8. **Tracking Performance on TUM RGB-D Dataset [44]** (ATE RMSE ↓ [cm]). Best results are highlighted as **first**, **second**, and **third**. For methods without complete scene coverage in the original reports, results obtained by running their open-source code are marked with ‘*’. If open-source code is unavailable, scenes without results are marked with ‘-’. DynaSLAM (RGB) [2] consistently fails to initialize or experiences extended tracking loss across all sequences and therefore cannot be included in this table.

iterations. To be more specific, the modifications involve (i) removing the calculation of disparity regularization mask; (ii) optimizing the map \mathcal{G} and the uncertainty MLP \mathcal{P} every 5 keyframes; (iii) skipping the refinement of non-keyframe pose via re-rendering loss; (iv) decrease the number of iterations of final map refinement to 3000. As shown in Table 9, the fast version still outperforms baselines by a clear margin with comparable runtime.

Rendering Results on TUM RGB-D Dataset [44]. Our method effectively removes distractors, as illustrated in Fig. 8. ReFusion [36] struggles to fully eliminate distractors, leading to the presence of multiple ghosting artifacts. DynaSLAM (N+G) [2] exhibits “black holes” due to insufficient multiview information for effective inpainting, while the regions it does manage to inpaint often suffer from noticeable

| Dataset | MonoGS [30] | Splat-SLAM [39] | Ours-full | | Ours-fast | |
|-----------|-------------|-----------------|-----------|-------|-----------|-------|
| | FPS ↑ ATE ↓ | FPS ↑ ATE ↓ | FPS ↑ | ATE ↓ | FPS ↑ | ATE ↓ |
| Wild-SLAM | 2.41 | 47.99 | 2.44 | 8.71 | 0.49 | 0.46 |
| Bonn | 2.98 | 22.80 | 1.99 | - | 0.50 | 2.31 |

Table 9. **Running time evaluation.** For each dataset, we report the average FPS and RMSE of ATE [cm]. We logged the total running time to process a sequence and compute FPS by dividing the total number of processed frames by the total running time. *Ours-full* is the full pipeline presented, while *Ours-fast* is a fast version of WildGS-SLAM.

whitish artifacts. MonoGS [30] and Splat-SLAM [39] exhibit blurry and floating artifacts as they do not explicitly address dynamic environments.

Full Tracking Results on the TUM RGB-D Dataset [44]. We report our performance on the full TUM RGB-D



Figure 9. **Additional Input View Synthesis Results on our Wild-SLAM iPhone Dataset.** Faces are blurred to ensure anonymity.

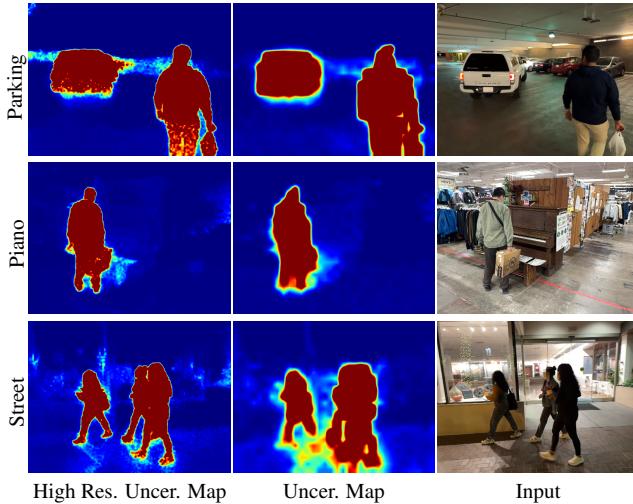


Figure 10. **High Resolution Uncertainty Map.**

Dataset [44] dynamic sequences in Table 8. Our method performs the best on average.

High Resolution Uncertainty Map. In Fig. 10, we present the visualization of high-resolution maps, as referenced in Fig. 5. Achieving higher resolution and sharper uncertainty maps is possible, though it comes at the cost of computational efficiency.

More Results on our Wild-SLAM iPhone Dataset. In addition to the results shown in Fig. 5 of the main paper, we provide additional results in Fig. 9.

Online Uncertainty Prediction. We visualize the online uncertainty prediction for frame 215—with MLP trained before, during, and after the umbrella enters the scene—in Fig. 11. Before (trained until frame 80), the MLP mainly classifies the moving human as a moving distractor since it has never seen the umbrella in the first 80 frames. As the umbrella enters the scene (frame 215), our uncertainty

| | fr1/desk | fr2/xyz | fr3/off | Avg. |
|---------------------------|------------|------------|------------|------------|
| RGB-D | | | | |
| ORB-SLAM2 [32] | 1.6 | 0.4 | 1.0 | 1.0 |
| NICE-SLAM [68] | 2.7 | 1.8 | 3.0 | 2.5 |
| Monocular | | | | |
| DROID-SLAM [47] | 1.8 | 0.5 | 2.8 | 1.7 |
| MonoGS [30] | 3.8 | 5.2 | 2.9 | 4.0 |
| Splat-SLAM [39] | 1.6 | 0.2 | 1.4 | 1.1 |
| WildGS-SLAM (Ours) | 1.7 | 0.3 | 1.4 | 1.1 |

Table 10. **Tracking Performance on TUM RGB-D Dataset (Static) [44]** (ATE RMSE \downarrow [cm]). Best results are highlighted as **first**, second, and third. Results for ORB-SLAM2 [32] and NICE-SLAM [68] are taken from NICE-SLAM [68]. Results for MonoGS [30] and Splat-SLAM [39] are taken from Splat-SLAM [39]. The results for DROID-SLAM [47] are obtained by running their open-source code.

prediction module rapidly identifies it as a moving distractor due to the inconsistency between the Gaussian map and the frame 215. Moreover, the uncertainty estimate stabilizes shortly afterward (frame 451).

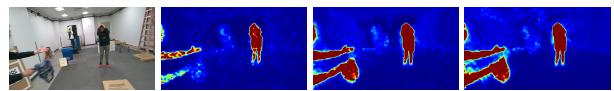


Figure 11. **Online Uncertainty Prediction.**

Pure Static Sequences. To demonstrate the robustness of our method, we also evaluate it on static sequences from the TUM RGB-D Dataset [44], shown in Table 10. Our approach performs on par with state-of-the-art monocular Gaussian Splatting SLAM methods, such as MonoGS [30] and Splat-SLAM [39].

Ablation Study on Disparity Regularization. Table 11 presents an ablation study evaluating the effects of (a) the

| Disp. Reg. Mask M | No Disp. Reg. in Final Global BA | Wild-SLAM | | Bonn | | TUM | | |
|------------------------|-------------------------------------|-----------|-------------|-------------|-------------|-------------|-------------|-------------|
| | | Before BA | After BA | Before BA | After BA | Before BA | After BA | |
| (i) | ✗ | ✗ | 3.12 | 1.95 | 4.34 | 2.56 | 2.17 | 1.86 |
| (ii) | ✓ | ✗ | 2.90 | 1.57 | 3.97 | 2.56 | 1.92 | 1.69 |
| (iii) | ✗ | ✓ | 3.17 | 0.46 | 4.40 | 2.47 | 2.16 | 1.55 |
| (iv) | ✓ | ✓ | 2.92 | 0.46 | 3.89 | 2.31 | 1.94 | 1.63 |

Table 11. **Ablation Study on Disparity Regularization** (ATE RMSE \downarrow [cm]). For each dataset, we report the average tracking error before and after the final global BA. ‘Before BA’ denotes before final global BA. ‘After BA’ denotes after final global BA.

| | Wild-SLAM | Bonn | TUM |
|---------------------------|-------------|-------------|-------------|
| MonST3R Mask | 2.60 | 2.58 | 1.80 |
| YOLOv8 + SAM Mask | 3.06 | 2.37 | 1.65 |
| WildGS-SLAM (Ours) | 0.46 | 2.31 | 1.63 |

Table 12. **Ablation Study on Distractor Estimation.** (ATE RMSE \downarrow [cm]). For each dataset, we report the average tracking error.

disparity regularization mask M used in DBA (Eq. (5)) during on-the-fly capture and (b) the exclusion of the disparity regularization term in the final global BA. Removing M (*rows iii and iv*) has minimal impact on the final global BA, as shown in the ‘After BA’ results. However, the ‘Before BA’ results are significantly degraded, highlighting the multi-view inconsistencies in monocular predictions. Excluding the disparity regularization term in the final global BA (*rows ii and iv*) has no effect on the ‘Before BA’ results (minor deviations are expected due to randomness and initialization) but leads to improved ‘After BA’ performance. This improvement is attributed to the availability of multiple views in the final global BA, which refines depth accuracy compared to monocular predictions. The best results are achieved when M is applied and the disparity regularization term is excluded during the final global BA (*row iv*), validating our design choices.

Ablation Study on Distractor Estimation. We compare various distractor estimation methods and utilize their resulting distractor masks for tracking in WildGS-SLAM. For the MonST3R mask, we aggregate masks from multiple runs because MonST3R supports only a limited number of images per run. The YOLOv8 + SAM mask corresponds to (c) in Table 5; we include it here for a clearer comparison. As shown in Table 12, our method consistently outperforms others, as other approaches struggle to produce accurate enough masks, particularly on our Wild-SLAM dataset, which features diverse and complex distractors.

Ablation Study on Pretrained Models. We conduct an ablation study on the pretrained models, namely the depth estimator and the feature exactor DINOv2 model, as presented in Table 13. Both novel view synthesis and tracking evaluations confirm that Metric3D V2 [13], combined with the finetuned DINOv2 model [57], achieves the best overall performance, validating our design choices.

| Depth Estimator | DINOv2 model | Wild-SLAM | | Bonn | | TUM | |
|------------------|---------------|-----------------|------------------|------------------|------------------|------------------|------------------|
| | | PSNR \uparrow | ATE \downarrow |
| DPTv2 [56] | Original [35] | 20.56 | 0.47 | 2.36 | 1.76 | | |
| DPTv2 [56] | Finetuned[57] | 20.57 | 0.47 | 2.41 | 1.66 | | |
| Metric3D V2 [13] | Original [35] | 20.58 | 0.52 | 2.31 | 1.61 | | |
| Metric3D V2 [13] | Finetuned[57] | 20.58 | 0.46 | 2.31 | 1.63 | | |

Table 13. **Ablation Study on Different Pretrained Models.** For the Wild-SLAM dataset, we report the novel view synthesis results (PSNR \uparrow) and the tracking error (ATE RMSE \downarrow [cm]). For the Bonn and TUM datasets, we report the average tracking error (ATE RMSE \downarrow [cm]).

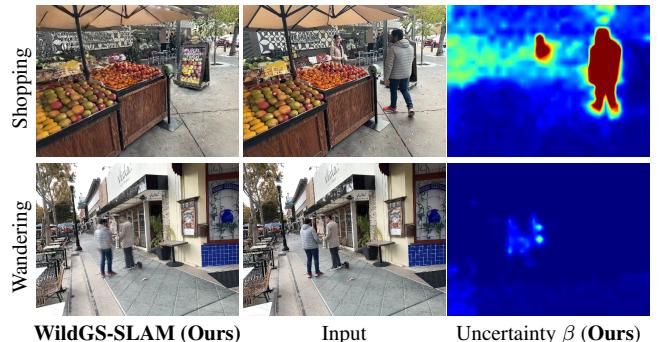


Figure 12. **Failure Cases.** In shopping dataset, patterns on the wall is incorrectly regarded as medium uncertainty because of the difficulty of reconstructing the complicated textures. In wandering, humans are not removed due to the lack of observation of the static scene. Faces are blurred to ensure anonymity.

Failure Cases. In Fig. 12, we present two failure cases of our method. In the first case, while our method successfully removes dynamic objects, it struggles to reconstruct the complex background, leading to a high SSIM loss in Eq. (4). Therefore, the high SSIM loss drives the uncertainty prediction to incorrectly assign higher uncertainty to static regions.

In the second case, the dynamic objects remain stationary in some of the frames and, since all frames are captured from roughly the same camera direction, no earlier frames are available to observe the static scene without the dynamic objects. As a result, the system assigns lower uncertainty to these regions and mistakenly reconstructs the dynamic objects.