MGSO: Monocular Real-time Photometric SLAM with Efficient 3D Gaussian Splatting

Yan Song Hu[1], Nicolas Abboud[1,2], Muhammad Qasim Ali[1], Adam Srebrnjak Yang[1], Imad Elhajj[2], Daniel Asmar[2], Yuhao Chen[1], John S. Zelek[1]

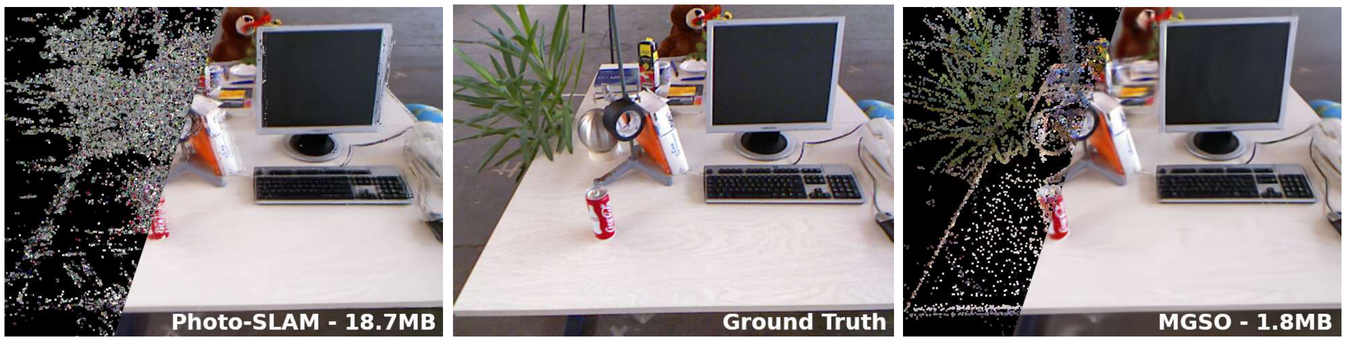


Fig. 1: Qualitative renders of the TUM-RGBD dataset [1] with input point clouds. By initializing 3D Gaussian Splatting (3DGS) [2] with dense, structured point clouds, MGSO produces reconstructions that are memory-efficient and high quality.

Abstract— **Real-time SLAM with dense 3D mapping is computationally challenging, especially on resource-limited devices. The recent development of 3D Gaussian Splatting (3DGS) offers a promising approach for real-time dense 3D reconstruction. However, existing 3DGS-based SLAM systems struggle to balance hardware simplicity, speed, and map quality. Most systems excel in one or two of the aforementioned aspects but rarely achieve all. A key issue is the difficulty of initializing 3D Gaussians while concurrently conducting SLAM. To address these challenges, we present Monocular GSO (MGSO), a novel real-time SLAM system that integrates photometric SLAM with 3DGS. Photometric SLAM provides dense structured point clouds for 3DGS initialization, accelerating optimization and producing more efficient maps with fewer Gaussians. As a result, experiments show that our system generates reconstructions with a balance of quality, memory efficiency, and speed that outperforms the state-of-the-art. Furthermore, our system achieves all results using RGB inputs. We evaluate the Replica, TUM-RGBD, and EuRoC datasets against current live dense reconstruction systems. Not only do we surpass contemporary systems, but experiments also show that we maintain our performance on laptop hardware, making it a practical solution for robotics, A/R, and other real-time applications.**

I. Introduction

Simultaneous localization and mapping (SLAM) is a fundamental task in autonomous robot navigation. It is the process by which a robot constructs a map of an environment while concurrently keeping track of its own location. Accurate self-localization is an essential precursor for advanced mobile robot tasks. Traditionally, SLAM systems provide semantically-poor map representations that are efficient for localization and basic navigation but lack the details needed for complex tasks. For example, sparse point clouds are efficient for localization but lack the surface detail needed for robotic grasping. For these complex robotic tasks, dense, high-fidelity spatial data is increasingly important.

To meet this demand, SLAM systems have evolved to generate dense 3D maps while still simultaneously performing localization. Dense SLAM systems are categorized into two approaches: decoupled and coupled. Decoupled approaches separate tracking from reconstruction, using a traditional SLAM system to provide outputs for a dense reconstruction process. Coupled approaches integrate dense reconstruction with both mapping and tracking, improving map quality but often facing speed bottlenecks, as accurate localization depends on building a high-quality map, which takes time.

A key challenge in decoupled systems is the lack of synergy between SLAM and dense reconstruction components. SLAM algorithms often fail to provide optimal data for high-quality dense reconstruction, compromising overall system performance. To address this challenge, we tailored our SLAM system to meet the specific needs of 3D Gaussian Splatting (3DGS) [2]. 3DGS typically requires an initial point cloud to begin reconstruction, with denser, well-structured initial point clouds leading to improved and faster results [3]. However, traditional feature-based SLAM methods produce sparse point clouds that are not optimal for 3DGS initialization. While RGB-D data could provide dense and accurate point clouds, using a monocular camera is preferable for wider applicability.

[1]Yan Song Hu, N. Abboud, M. Ali, A. Yang, Y. Chen, and J. Zelek are with Vision and Image Processing Lab at the Faculty of System Design Engineering, at the University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada; emails: y324hu,n2abboud,m45ali,asyang,yuhao.chen1, jzelek@uwaterloo.ca
[2] N. Abboud, I. Elhajj and Daniel Asmar are with Vision and Robotics Lab, Maroun Semaan Faculty of Engineering and Architecture, American University of Beirut, 1107 2020, Riad El Solh, Beirut, Lebanon; emails: nfa53,ie05,da20@aub.edu.lb

In this paper, we introduce **Monocular-GSO (MGSO)**, a dense visual SLAM system that performs high-quality online 3D reconstruction in real-time using a single monocular camera. MGSO is a decoupled system that employs photometric SLAM to initialize a 3D Gaussian Splatting (3DGS) module running in parallel, enabling live dense scene reconstruction. The MGSO acronym is a blend of Direct Sparse Odometry (DSO) [4], the photometric SLAM system we built upon, and Gaussian Splatting (GS). In contrast to conventional feature-based SLAM methods that generate sparse point clouds, MGSO is designed to track a dense set of pixels, yielding a denser and well-structured point cloud output. We leverage this dense, structured point cloud to initialize 3D Gaussian Splatting (3DGS) in unmapped areas. Initializing with a high-quality set of points accelerates 3DGS optimization, guiding it toward more compact reconstructions with fewer artifacts and redundancies. As a result, our approach leads to real-time reconstruction with dense 3D maps with high quality and memory compactness.

The main contributions of MGSO are as follows:

- A real-time dense SLAM system that harnesses the synergy between photometric SLAM and 3DGS.
- Our system only requires a monocular camera.
- Experiments show that our system has a combination of speed, map quality, and memory efficiency unmatched by other dense SLAM systems.

II. Related Work

Dense SLAM research has long explored various 3D representations such as signed distance functions [5], dense point clouds [6], and surfel clouds [7][8]. Despite these advancements, efficiently generating high-quality maps for real-time applications remains challenging. Recent innovations in Neural Radiance Fields (NeRFs) [9] and 3D Gaussian Splatting [2] show promise in addressing the issue. These approaches offer high-quality representations that are easy to create and render. Consequently, this section will focus on systems based on these two techniques.

NeRFs represent scenes using a neural network that outputs novel views based on the input camera's position and rotation. They also allow for the incremental learning and updating of their 3D representations through gradient-based optimization [9]. This capability has been effectively applied in pioneering works like iMap [10], and further improved by subsequent systems such as NICE-SLAM [11], Orbeez-SLAM [12], and NeRF-SLAM [13]. However, NeRF-based SLAM systems face two notable challenges: they require predefined scene bounds, which is often impractical in exploratory environments; and their implicit scene representations can be difficult to integrate with other systems. These limitations have spurred the adoption of 3DGS as a more suitable dense reconstructor.

3DGS is an approach to scene representation that models the environment as a large set of 3D Gaussians, which resemble blurry overlapping clouds. Rendering images involves projecting Gaussians onto the camera plane, depth-sorting, and blending them front-to-back. Similar to NeRF, 3DGS allows for gradient-based optimization of parameters by minimizing the discrepancy between rendered and input images. The method will also heuristically clone or prune Gaussians over time. 3DGS offers a boundless and fast-to-render representation compared to NeRF, making it highly suitable for real-time SLAM applications.

Early 3DGS-based SLAM systems, like MonoGS [14], SplaTAM [15], GS-SLAM [16], and Gaussian-SLAM [17], utilize a one-stage approach where tracking and mapping are tightly coupled. This approach introduces a dependency on map refinement before tracking can proceed, which results in slow performance, as shown in Table I. Even newer coupled systems such as CG-SLAM [18], RTG-SLAM [19], and SplatSLAM [20] struggle to run at speeds faster than 20 fps. To allow dense 3DGS-SLAM to operate faster, two-stage systems like Photo-SLAM [21], IG-SLAM [22] and GS-ICP [23] emerged, decoupling the tracking and mapping functions. Furthermore, the majority of current 3DGS systems heavily rely on depth data to perform 3D reconstruction (Table I), making them dependant on RGB-D sensors.

TABLE I: Existing 3DGS SLAM Systems

Name	Type	Possible Sensors	FPS
MonoGS [14]	Coupled	RGB,RGB-D	<5
SplaTAM [15]	Coupled	RGB-D	<5
GS-SLAM [16]	Coupled	RGB-D	>5, <10
Gaussian-SLAM [17]	Coupled	RGB-D	<5
CG-SLAM [18]	Coupled	RGB-D	>15, <20
RTG-SLAM [19]	Coupled	RGB-D	>15, <20
SplatSLAM [20]	Coupled	RGB	<5
GS-ICP [23]	Decoupled	RGB-D	>30
Photo-SLAM [21]	Decoupled	RGB*,RGB-D	>30
IG-SLAM [22]	Decoupled	RGB	>5, <10
MGSO (ours)	Decoupled	RGB	>30

*Both monocular and stereo

Our system, MGSO, improves on existing two-stage 3DGS-based SLAM systems while utilizing only RGB data. It operates at 30 fps or higher, a performance matched only by Photo-SLAM and GS-ICP (see Table I). While MGSO is most similar to Photo-SLAM, which combines 3DGS with ORBSLAM3, we address Photo-SLAM's tendency to create large, memory-inefficient maps. GS-ICP offers exceptional speed but requires depth data to initialize its iterative closest point tracking, whereas our system operates using only RGB data. Unlike IG-SLAM [22], which uses pseudo-depth RGB-D data at the cost of slower performance, MGSO maintains real-time speeds while generating accurate, compact maps.

III. Method

MGSO integrates two core components that operate concurrently: a SLAM module responsible for accurate pose estimation, and a 3D dense reconstruction module for mapping.

A. *SLAM module:*

The tracking backbone of our system is built upon a lineage of visual SLAM approaches originating from Direct Sparse Odometry (DSO) [4]. DSO's key innovation is demonstrating that selective pixel sampling for photometric tracking enhances localization accuracy compared to using all pixels in an image. We chose to build upon DSO because we found that its pixel selection strategy also aligns well with initializing 3DGS. DSO tracks a set of pixels across consecutive frames i and j, optimizing the camera pose ($\boldsymbol{p}$) by minimizing the photometric loss equation below for each pixel tracked,

$$E = \left\| (I_j[\boldsymbol{p}_j] - b_j) - \frac{s_j a_j}{s_i a_i}(I_i[\boldsymbol{p}_i] - b_i) \right\| \tag{1}$$

where I queries the pixel intensity, a and b are variables to account for lighting changes, and s is the camera exposure. The basic principle of this loss equation is to identify pose changes that best match the pixel intensity variation between consecutive frames i and j. The equation is applied at both tracking and mapping levels.

At each frame, our system's tracking process calculates pose changes relative to the latest keyframe, assuming a fixed map. The map of tracked pixels is only adjusted when a keyframe is inserted. A new keyframe is a reference frame that captures a distinct view of the scene relative to existing keyframes. When mapping is done, all current keyframe poses and the map, which consists of the tracked pixel points, are adjusted. Our system then converts the map of tracked pixels into a point cloud map and adds it, along with the keyframe poses, to the dense reconstruction module. We adopt DSO's windowed keyframe management strategy, which generates keyframes when significant changes in the field of view, rotation, or lighting are detected. Older keyframes are removed if the number of keyframes exceeds the window size, which by default is eight, using a distance-based score to ensure a well-distributed set of keyframes in 3D space.

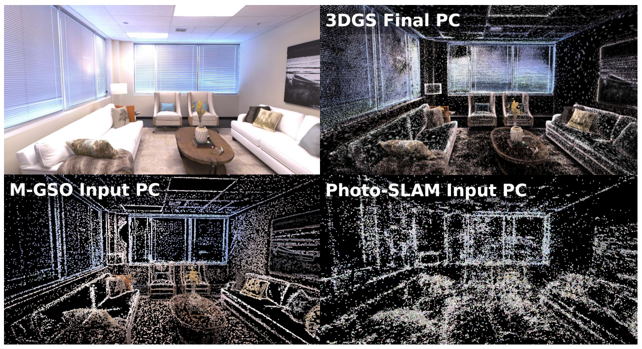


Fig. 3: Comparison of 3DGS point clouds from MGSO and Photo-SLAM on Replica room0. Top left: original frame; top right: Map from original 3DGS after 10,240 iterations with Gaussian size set to 0.1. Bottom: MGSO vs. Photo-SLAM point clouds.

The inspiration for our method is from analyzing the final 3DGS Gaussian position of the original 3DGS (Figure 3). We realized that the final position, colour, and distribution of Gaussians of the final map resembled the point cloud output from DSO (Figure 3). From this observation, we conjectured that initializing 3DGS with photometric SLAM would enhance 3DGS optimization because it would reduce the required optimizations.

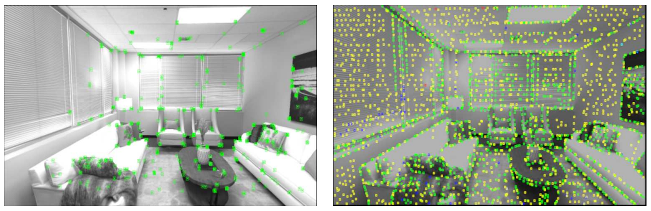


Fig. 4: Tracked points from ORBSLAM3 (left) compared to our system (right). Our system tracks much more points than ORBSLAM3, which results in denser point clouds outputs.

A major aspect of DSO's well-structured dense point cloud is it's pixel selection strategy. DSO does pixel selection by dividing the image into blocks and selecting the highest-gradient pixel above a gradient threshold in each block. It then repeats the process with a lower threshold and larger blocks. This approach not only tracks more pixels in complex areas but also ensures pixel selection in simpler regions. It differs from traditional methods, which typically only track easily recognizable features such as corners and edges. The differences between the two approaches can be observed in Figure 4. This is important because we observed that while completed 3DGS maps have more Gaussians in complex areas, they still maintain some Gaussians in non-complex areas. Furthermore, DSO tracks pixels with high gradients, which are much more common than trackable feature points. Consequently, DSO's output point cloud more closely matches the density of completed 3DGS maps. Our experiments revealed that whileDSO's pixel selection density is optimal for tracking, increasing the pixel selection density enhances 3DGS performance, particularly in low-gradient areas that are challenging for tracking. To address this issue, we modified DSO to include additional tracked pixels not used for pose estimation to increase the output point cloud density (Figure 5). This modification allows the system to have the optimal pixel density for both tracking and 3DGS. Despite these enhancements, flat regions with minimal to no gradients remain sparsely populated with tracked pixels. This is because DSO's pixel tracking system requires at least some gradient for tracking, and as a result, pixels in areas with no gradient are never tracked.

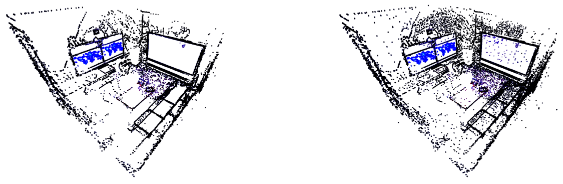


Fig. 5: Original DSO Point cloud (left) compared to our system's point cloud (right). Our system has much more output points, especially in flat low-gradient regions.

We observed that 3DGS performs better with slightly misplaced initialized points in flat areas than with none at all. Therefore, we implemented an interpolation method that estimates point locations in low-gradient regions based on nearby tracked pixels. Our method employs the Delaunay triangulation algorithm [24] to divide the image into a series of triangles using tracked pixels as vertices. The depth of each interpolated point is calculated as the average depth of the triangle's vertices, which generally provides accurate results for pixels on flat surfaces. While feature-based systems like Photo-SLAM also interpolate inactive 2D feature points, our method outperforms theirs due to a higher initial point count and by focusing interpolation on flat areas where it's most accurate, which can be observed in Figure 3.

B. Dense Reconstruction

MGSO employs 3DGS as its dense reconstruction method. Following the original 3DGS, we map the scene using a set of anisotropic Gaussians G. Each Gaussian G_i is modeled with an opacity, rotation, location, scale, and color. We follow Mono-GS' [14] technique of representing color using RGB instead of spherical harmonics because Mono-GS showed this increased speed for minimal impact on reconstruction quality. We render RGB images of the map using the original differentiable tile-based rasterization introduced in 3DGS.

The parameters of each Gaussian are optimized using gradient descent to minimize the photometric loss L:

$$L = |I_r - I_{gt}|(1-\lambda) + SSIM(I_r, I_{gt})\lambda \tag{2}$$

where I_r denotes the rendered image, I_{gt} refers to the captured image, λ is a weighting factor and SSIM [25] represents the structural similarity metric.

In order to improve the speed of our system, we employ Gaussian-pyramid based learning introduced in Photo-SLAM [21] to progressively train the Gaussian map. The pyramid helps accelerate training for live video scenarios. A multi-scale Gaussian pyramid is created by repeatedly smoothing and down-sampling ground-truth image captured by the camera. The photometric loss calculation progresses from using the highest pyramid level for I_{gt} in initial iterations to lower levels as training advances. Furthermore, we use an optimized version of the 3DGS CUDA back-end [26] that is faster than the original.

Our adaptive control strategy periodically densifies and prunes Gaussians every 1000 training iterations to improve map quality over time. We design our strategy around the point clouds returned by our SLAM module, similar to how Photo-SLAM tailored their strategy to ORBSLAM3 [27]. The point clouds generated by our SLAM system are characterized by their high density and uniform coverage. They adapt to the scene's complexity, concentrating points in intricate areas while maintaining representation in simpler regions. When a novel keyframe is processed, we initialize new Gaussians with location and color taken from the point cloud created by the SLAM system.

However, we noticed the emergence of floaters when utilizing the 3DGS's densification and pruning strategies. To mitigate the presence of floaters, we utilized the adaptive control strategies in AbsGS [28]. Thus, as part of our adaptive control strategy we periodically densify Gaussians with high homo-directional view-space position gradients by splitting or cloning them. Large, high-variance Gaussians are split, while small Gaussians in under-reconstructed regions are cloned. Furthermore, we periodically prune Gaussians with low opacity to remove transparent floaters. We use the same splitting and cloning parameters as original 3DGS.

IV. Experiments and Discussions

We evaluate MGSO against the latest state-of-the-art 3DGS dense SLAM systems: MonoGS [14], GlORIE-SLAM [20], Splat-SLAM [15], IG-SLAM [22], and Photo-SLAM [21], to demonstrate our system's combination of high-quality reconstruction, efficient runtime, and compact maps.

A. Implementation and Setup

Datasets: Evaluations are done on the sythetic Replica [29] dataset and real-life EuRoC MAV [30] and TUM-RGBD [1] datasets. These datasets are commonly used to evaluate dense SLAM systems.

Hardware: Results for Replica and re-testing Photo-SLAM were done on an Intel i9-14900K with a NVIDIA RTX 4090. The laptop runs for Replica were done on an Intel i7-12700H with a NVIDIA GeForce RTX 3080 mobile. The results for EuRoC and TUM were done on an Intel i5-12600KF with a NVIDIA RTX 3090.

Experimental Setup: We utilize the default optimization configuration of 3DGS with the exception of adjusting the densification interval to 1000. We configure the SLAM module (DSO [4]) to the default tracking settings. The parameters for increasing SLAM output point density through the inclusion of untracked points were determined through iterative testing.

For our experiments on the EuRoC MAV dataset, we implemented a preprocessing step involving undistorting and cropping the images before inputting them into the SLAM systems. This procedure was necessary to resolve the challenge of aligning poses between the undistorted 3DGS map and the distorted ground truth images. We also re-evaluated Photo-SLAM using this modified dataset, and notably, our new tests showed significant improvements to previously reported performance (Table IV).

Evaluation: To ensure a fair comparison, we evaluated our system by inputting the output 3DGS maps and pose estimation data into the original 3DGS rendering and metric scripts. We evaluate our reconstructions with the standard image quality metrics: PSNR, SSIM [25], and LPIPS [31]. Using an third-party evaluation system rather than built-in metrics offers a more realistic assessment, accounting for real-world factors like potential misalignment between the poses and map. Consistent with other dense SLAM systems, we evaluate on every fifth frame. Photo-SLAM's evaluations are updated using this methodology to ensure consistency. We did ten runs for the Replica and EuRoC dataset and five runs for the TUM-RGBD dataset. Results for other systems were obtained from their respective publications, with the

TABLE II: Reconstruction Results on Replica (cm)

Method	**metric**	**o0**	**o1**	**o2**	**o3**	**o4**	**r0**	**r1**	**r2**	**Avg.**	**Map Size**	**FPS**
	PSNR[dB] ↑	35.22	34.35	**29.58**	28.55	**32.05**	26.75	27.78	29.43	30.46		
Photo-SLAM	SSIM ↑	0.94	0.93	0.91	0.89	0.92	0.79	0.84	0.89	0.89	22.5 MB	>30
	LPIPS↓	**0.21**	**0.23**	**0.26**	**0.26**	**0.22**	**0.31**	0.28	**0.24**	**0.25**		
	PSNR[dB] ↑	35.85	37.15	29.19	**30.44**	30.08	27.71	29.50	31.33	31.41		
MGSO	SSIM ↑	0.94	0.94	0.90	0.90	0.91	0.79	0.86	0.91	0.89	**4.6 MB**	30
	LPIPS↓	0.22	0.25	0.29	**0.26**	0.26	0.33	**0.27**	**0.24**	0.27		
	PSNR[dB] ↑	**36.34**	**38.20**	28.90	30.27	31.41	**28.11**	**30.04**	**31.89**	**31.90**		
MGSO (laptop)	SSIM ↑	**0.95**	**0.96**	0.90	**0.91**	**0.93**	**0.82**	**0.87**	**0.92**	**0.91**	5.2 MB	30
	LPIPS↓	0.24	0.25	0.31	0.27	0.25	0.35	0.29	0.26	0.28		

exception of Mono-GS, whose results were sourced from Splat-SLAM [20].

However, the evaluation process tends to favor slower systems, as 3DGS performs better with extended training times, which may create bias against faster systems. Therefore, readers should consider the differences in speed when interpreting results. Because our system inherits real-time constraint handling from DSO, we decided to constrain our speed to the real-time speeds of videos to enhance the realism of the results. Replica and TUM-RGBD were run at 30 fps while EuRoC was run at 20 fps.

B. Discussion

TABLE III: Absolute Trajectory Error of Tracking on Replica (RMSE in cm)

Method	**r0**	**r1**	**r2**	**o0**	**o1**	**o2**	**o3**	**o4**	**Avg.**
Photo-SLAM	0.58	**0.32**	**5.03**	0.47	0.58	0.35	1.18	0.23	**1.09**
MGSO	**0.35**	1.02	5.93	**0.22**	**0.54**	**0.28**	**0.34**	**0.2**	1.11

1) Localization: While we include tracking results in Table III, tracking performance is not our system's focus. We did not modify the localization aspect of DSO and should inherit its performance. Our system's comparable tracking to Photo-SLAM suggests any rendering differences are not due to localization.

2) Reconstruction Quality: MGSO consistently achieves high PSNR and SSIM across all datasets. In the Replica dataset (Table II), MGSO outperforms Photo-SLAM with a PSNR of 31.406 dB and a much smaller map size of 4.618 Mb, with its mobile version showing even better results (31.896 dB PSNR, 0.906 SSIM). On the EuRoC dataset (Table IV), MGSO further demonstrates superior performance with 22.10 dB PSNR and 0.80 SSIM, compared to Photo-SLAM's 19.68 dB and 0.75 SSIM. Similar trends are observed on the TUM dataset (Table V), where MGSO achieves a higher PSNR and SSIM than Photo-SLAM. MGSO's key advantage is its ability to generate dense, well-structured point clouds, requiring less refinement and resulting in more compact maps—half the size of Photo-SLAM's. This efficient initialization reduces the need for extensive operations like cloning and pruning, leading to faster convergence and fewer reconstruction artifacts. Figure 6 further highlights MGSO's improved rendering of flat surfaces, fewer floating artifacts, and better preservation of edges and thin features, showcasing its capacity to handle complex scenes with more accurate and detailed reconstructions.

TABLE IV: Reconstruction Results on EuRoC

Method	**metric**	**MH**	**V1**	**V2**	**Avg.**	**Mem.**
	PSNR[dB] ↑	18.60	18.30	17.94	18.28	
Photo-SLAM	SSIM ↑	0.65	0.73	0.65	0.68	111.8
	LPIPS↓	0.39	0.44	0.53	0.46	
	PSNR[dB] ↑	**20.75**	**20.26**	**20.31**	**20.44**	
MGSO	SSIM ↑	**0.72**	**0.79**	**0.75**	**0.76**	**8.3**
	LPIPS↓	**0.36**	**0.39**	**0.39**	**0.38**	

Mem. is average map size in Mb

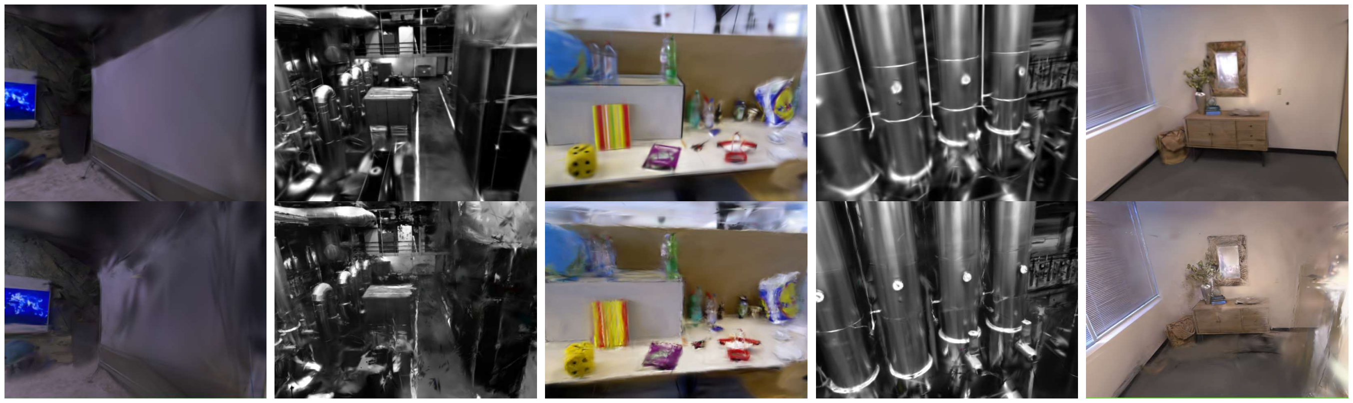
TABLE V: Reconstruction Results on TUM's

Method	**metric**	**fr1**	**fr2**	**fr3**	**Avg.**
	PSNR[dB] ↑	18.01	16.93	17.11	17.35
Photo-SLAM	SSIM ↑	0.65	0.60	0.62	0.63
	LPIPS↓	**0.41**	0.41	0.42	0.41
	PSNR[dB] ↑	**18.07**	**24.10**	**21.61**	**21.26**
MGSO	SSIM ↑	**0.66**	**0.80**	**0.75**	**0.74**
	LPIPS↓	0.45	**0.33**	**0.38**	**0.39**

3) Resource Efficiency and Real-Time Performance: MGSO excels in low memory usage and real-time FPS. On the EuRoC dataset (Table IV), MGSO requires only 8.32 MB, significantly less than Photo-SLAM's 109.73 MB, and just 2.85 MB on the TUM dataset (Table V), compared to Photo-SLAM's 17 MB. All the while, MGSO maintains real-time performance (Tables II,VI) MGSO's structured point clouds allow it to create compact maps with minimal redundant elements, resulting in lower memory consumption. This contrasts with Photo-SLAM's larger map sizes, which require more refinement. Figure 7 underscores MGSO's balance of high FPS with low map size, we are the only system capable of real-time performance with compact maps.

C. Ablations

ted experiments to evaluate the robustness of our system to the frequency of densification and pruning. As shown in table VII, increasing the rate of densification does not improve



(a) MGSO renders flat surfaces well (b) MGSO has less floaters and artifacts (c) MGSO has better edges on difficult scene (d) MGSO renders thin features better (e) MGSO has less floaters and artifacts

Fig. 6: Comparison of difficult novel view renders between MGSO (top) and Photo-SLAM (bottom). Captions describe how MGSO performs better.

TABLE VI: Replica Aggregated Results

Method	PNSR[dB]	Map Size	FPS	GPU Usage
GlORIE-SLAM (GlS)	31.04	114 Mb	0.23	15.22
Mono-GS (MGS)	31.22	6.8 Mb	0.32	14.62
Splat-SLAM (SpS)	**36.45**	6.8 Mb	1.24	17.57
IG-SLAM (IGS)	36.21	14.8 Mb	9.94	16.20
Photo-SLAM (PhS)	30.46	22.5 Mb	**>30***	**3.62**
MGSO (MGSO)	31.41	**4.3 Mb**	**30***	7.98

*System processed data as fast as inputted video stream

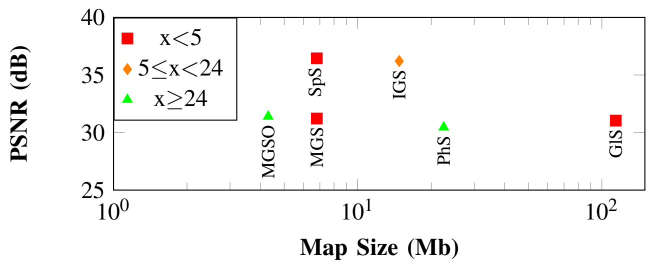


Fig. 7: Plot of Table VI. The 'x' in the legend represents the frames per second. We consider fps>24 (cinema fps standard) as real-time.

reconstruction results and instead reduces the compactness of the final 3DGS map. In fact, at high densification rates, the reconstruction quality diminishes. The observed robustness suggests that our system generates spatially accurate point clouds that effectively capture both complex and simple areas of the scene, without requiring significant adjustments from densification or pruning.

Table VIII demonstrates the importance of dense, well-structured inputs to 3D Gaussians. We deliberately reduced the density of our point clouds by utilizing only half of the points from our tracking system and excluding additional untracked points. This intentional sparsification resulted in a marked decline in reconstruction quality.

TABLE VII: Densify Iteration Ablation

Scene	metric	1024	512	256	128	64	32
o0	PSNR[dB]↑	37.06	37.10	37.02	37.15	37.40	36.11
	Memory(Mb)	6.4	7.2	9.0	13.0	21.4	38.1
r0	PSNR[dB]↑	28.84	28.73	28.86	28.76	28.49	27.73
	Memory(Mb)	4.5	5.3	7.7	12.4	22.2	38.5

Ablation Evaluations done on training images instead of test images

TABLE VIII: Additional Dense Points Ablation

Dataset	metric	o0	o1	o2	r0	r1
Base	PSNR[dB]↑	37.06	38.37	29.81	28.84	30.68
	Memory(Mb)	6.4	4.2	6.0	4.5	5.3
Halved	PSNR[dB]↑	33.48	33.93	27.87	27.30	29.13
	Memory(Mb)	2.0	1.7	2.2	2.1	2.0

Ablation evaluations done on training images instead of test images

V. Conclusions

MGSO integrates real-time photometric SLAM with 3D Gaussian Splatting (3DGS) to achieve dense, high-quality, and memory efficient 3D reconstruction using only a monocular camera. Our approach addressed several challenges in order to harness the natural compatibility of these two techniques. Its proven versatility across various environments without the use of depth sensors makes it optimal for robotics, AR/VR, and digital twin applications. Future research could explore implementing loop closure for global consistency and real-time re-rendering for adaptive scene reconstruction, enhancing MGSO's precision and efficiency in complex, large-scale environments.

ACKNOWLEDGMENT

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) (grant-number, www.nserc-crsng.gc.ca) and the DIDYMOS-XR Horizon Europe project (grant number 101092875–DIDYMOS-XR,www.didymos-xr.eu).

References

[1] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, July 2023. [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[3] Y. S. Hu, D. Mao, Y. Chen, and J. Zelek, "Towards real-time gaussian splatting: Accelerating 3dgs through photometric slam," 2024. [Online]. Available: https://arxiv.org/abs/2408.03825

[4] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[5] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, *et al.*, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 559–568.

[6] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2100–2106.

[7] T. Schöps, T. Sattler, and M. Pollefeys, "Bad slam: Bundle adjusted direct rgb-d slam," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 134–144.

[8] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[10] E. Sucar, S. Liu, J. Ortiz, and A. Davison, "iMAP: Implicit mapping and positioning in real-time," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[11] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[12] C.-M. Chung, Y.-C. Tseng, Y.-C. Hsu, X.-Q. Shi, Y.-H. Hua, J.-F. Yeh, W.-C. Chen, Y.-T. Chen, and W. H. Hsu, "Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9400–9406.

[13] A. Rosinol, J. J. Leonard, and L. Carlone, "Nerf-slam: Real-time dense monocular slam with neural radiance fields," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 3437–3444.

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

[15] N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten, "Splatam: Splat track & map 3d gaussians for dense rgb-d slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 357–21 366.

[16] C. Yan, D. Qu, D. Xu, B. Zhao, Z. Wang, D. Wang, and X. Li, "Gs-slam: Dense visual slam with 3d gaussian splatting," in *CVPR*, 2024.

[17] V. Yugay, Y. Li, T. Gevers, and M. R. Oswald, "Gaussian-slam: Photo-realistic dense slam with gaussian splatting," 2023.

[18] J. Hu, X. Chen, B. Feng, G. Li, L. Yang, H. Bao, G. Zhang, and Z. Cui, "Cg-slam: Efficient dense rgb-d slam in a consistent uncertainty-aware 3d gaussian field," *arXiv preprint arXiv:2403.16095*, 2024.

[19] Z. Peng, T. Shao, Y. Liu, J. Zhou, Y. Yang, J. Wang, and K. Zhou, "Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting," in *ACM SIGGRAPH 2024 Conference Papers*, 2024, pp. 1–11.

[20] E. Sandström, K. Tateno, M. Oechsle, M. Niemeyer, L. V. Gool, M. R. Oswald, and F. Tombari, "Splat-slam: Globally optimized rgb-only slam with 3d gaussians," 2024. [Online]. Available: https://arxiv.org/abs/2405.16544

[21] H. Huang, L. Li, C. Hui, and S.-K. 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.

[22] F. A. Sarikamis and A. A. Alatan, "Ig-slam: Instant gaussian slam," 2024. [Online]. Available: https://arxiv.org/abs/2408.01126

[23] S. Ha, J. Yeon, and H. Yu, "Rgbd gs-icp slam," *ArXiv*, vol. abs/2403.12550, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:268531141

[24] C. L. Lawson, "Transforming triangulations," *Discrete Mathematics*, vol. 3, no. 4, pp. 365–372, 1972. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0012365X72900933

[25] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[26] J. Patas, "Gaussian splatting cuda," https://github.com/MrNeRF/gaussian-splatting-cuda, 2023.

[27] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[28] Z. Ye, W. Li, S. Liu, P. Qiao, and Y. Dou, "Absgs: Recovering fine details for 3d gaussian splatting," 2024.

[29] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[30] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016. [Online]. Available: https://doi.org/10.1177/0278364915620033

[31] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.