
Accelerating SplaTAM for Dense RGB-D SLAM via Feature-Based Tracking and Adaptive Resolution

Galane Basha Namomsa
Carnegie Mellon University
Kigali, Rwanda
gnamomsa@andrew.cmu.edu

Peng Boris Akebuon
Carnegie Mellon University
Kigali, Rwanda
pborisak@andrew.cmu.edu

Matia Mulumba Mukasa
Carnegie Mellon University
Kigali, Rwanda
mmukasa@andrew.cmu.edu

Isaac Thani Moses
Carnegie Mellon University
Kigali, Rwanda
itmoses@andrew.cmu.edu

Abstract

This work proposes three extensions to SplaTAM—an efficient dense RGB-D SLAM system based on 3D Gaussian splatting—focused on reducing inference time without compromising reconstruction quality. We replace gradient-based tracking with a feature-based pipeline using LightGlue combined with SuperPoint, uniform model soup, and greedy model soup strategies. Experiments on the Replica dataset show that the uniform soup variant achieves the best speed-quality trade-off, improving runtime while maintaining comparable reconstruction performance based on SSIM, PSNR, and LPIPS.

1 Motivation

Real-time 3D scene reconstruction is critical for applications in robotics, augmented/virtual reality, and autonomous navigation. Current dense SLAM systems like SplaTAM deliver high-fidelity reconstructions but suffer from computational inefficiencies during tracking and mapping. Accelerating these components not only benefits research but also enables deployment on systems with limited resources, thus broadening the range of potential applications.

2 Objective

The main objective is to reduce SplaTAM’s inference time without compromising the quality of the reconstruction by:

- Incorporating feature-based tracking techniques (inspired by FlashSLAM and LightGlue) to speed up camera pose estimation.
- Employing an adaptive resolution and LOD strategy to dynamically adjust computation based on scene complexity.

3 Literature Review and Background

Recent work by Keetha et al. [4] introduced SplaTAM, demonstrating that explicit 3D Gaussian splatting can achieve high-speed rendering (up to 400 FPS) for dense SLAM. In parallel, FlashSLAM

[7] and LightGlue [5] have shown that fast, feature-based camera tracking can significantly reduce pose estimation time. These studies motivate our approach: by replacing slower gradient-based tracking with a feature-matching pipeline and applying adaptive resolution to concentrate compute where needed, we can enhance SplaTAM’s performance.

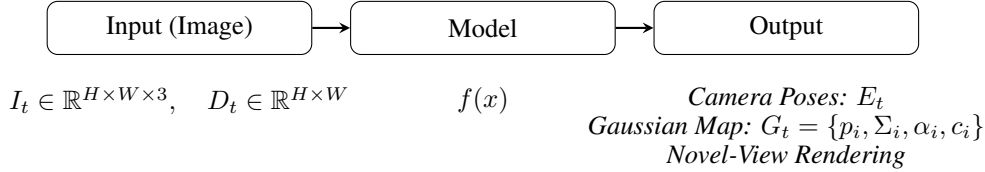
To enhance feature-based tracking, we incorporate a "model soups" approach introduced by Wortsman et al. (2022) [11]. This technique involves averaging weights from multiple fine-tuned models, effectively combining them. Unlike traditional ensembling methods, model soups do not increase inference time, providing a computationally efficient means of improving accuracy and robustness. The adaptive weighting in model soups allows for improved matching accuracy, reducing inference time without compromising reconstruction quality [11].

By combining the adaptive attention of LightGlue and the efficiency of model soups, our proposed enhancements aim to significantly improve SplaTAM’s real-time capabilities, offering an optimized balance between inference speed and reconstruction fidelity.

4 Methodology

4.1 Baseline Methodology

SplaTAM introduces a novel approach to dense RGB-D SLAM by representing the scene as a collection of 3D Gaussians, **facilitating high-fidelity reconstruction from a single unposed RGB-D camera**. It employs a simple online tracking and mapping system tailored to this Gaussian map, utilizing silhouette masks (a binary image that isolates an object from its background by highlighting only its outer contour) to capture scene density effectively. This methodology enables fast rendering and dense optimization, efficiently identifying previously mapped areas and expanding the map by adding more Gaussians to improve the quality of the 3D rendering.



SplaTAM model part can be subdivided into three-step SLAM pipeline: **Camera Tracking**, **Gaussian Densification**, and **Map Update**, optimized via differentiable rendering.

4.1.1 Camera Tracking

The camera pose E_t is estimated by minimizing photometric and depth errors:

$$E_t = \arg \min_E \sum_p (S(p) > \lambda) \left(|D_t(p) - \hat{D}_t(p)| + 0.5 |I_t(p) - \hat{I}_t(p)| \right) \quad (1)$$

where $S(p)$ is the silhouette mask, $D_t(p)$ and $I_t(p)$ are input depth and color values, and $\hat{D}_t(p)$ and $\hat{I}_t(p)$ are the rendered counterparts.

4.1.2 Gaussian Densification

New Gaussians are added in unmapped regions:

$$M(p) = (S(p) < 0.5) + (D_{GT}(p) < D(p)) \cdot (|D(p) - D_{GT}(p)| > \lambda_{MDE}) \quad (2)$$

where $D_{GT}(p)$ is the ground-truth depth.

4.1.3 Map Update

Gaussian parameters (position, scale, color, opacity) are optimized via:

$$G_t = \arg \min_G \sum_{k=1}^t \|\text{Render}(G, E_k) - F_k\|^2 \quad (3)$$

4.2 Loss Function

The SLAM pipeline is trained end-to-end to minimize the per-pixel ℓ_1 reconstruction error between the rendered output from the Gaussian-splat renderer and the RGB-D sensor measurements. The optimization occurs in two stages: mapping and pose tracking.

Core Objective (Mapping): The mapping optimization aims to find the optimal set of 3D Gaussians G_t that minimizes the per-pixel ℓ_1 reconstruction error between the rendered image and the sensor observation over all frames up to time t :

$$G_t = \arg \min_{G'} \sum_{k=1}^t \|\text{Render}(G', E_k) - F_k\|_1 \quad (4)$$

where:

- $G' = \{(p_g, \Sigma_g, c_g, \alpha_g)\}_g$ is the set of all 3D Gaussians (positions, covariances, colors, opacities),
- E_k is the known camera pose (extrinsics) at frame k ,
- F_k is the ground-truth RGB image at frame k from the sensor,
- $\text{Render}(G', E_k)$ is the differentiable Gaussian-splat rendering of the scene from pose E_k , producing a 640×480 image.

Per-Iteration Training Loss. At each mapping iteration, the training objective is composed of two terms: an ℓ_1 photometric loss and an ℓ_1 depth-matching loss. The combined mapping loss is defined as:

$$\mathcal{L}_{\text{map}} = \underbrace{\sum_i \|I_i^{\text{render}} - I_i^{\text{gt}}\|_1}_{\text{Photometric L1}} + \underbrace{\sum_i \|d_i^{\text{render}} - d_i^{\text{gt}}\|_1}_{\text{Depth L1}} \quad (5)$$

where:

- I_i^{render} is the rendered RGB color at pixel i ,
- I_i^{gt} is the ground-truth RGB color from the sensor,
- d_i^{render} is the rendered depth at pixel i ,
- d_i^{gt} is the ground-truth depth from the sensor.

Both loss terms are weighted equally. In practice:

- **Image Loss** = mean ℓ_1 error over RGB channels,
- **Depth Loss** = mean ℓ_1 error over depth,
- **Total Loss** = Image Loss + Depth Loss.

$$\mathcal{L}_k = \underbrace{\frac{1}{N} \sum_p \|F_k(p) - \hat{F}_k(p)\|_1}_{\mathcal{L}_{\text{RGB}}} + \underbrace{\frac{1}{N} \sum_p \|D_k(p) - \hat{D}_k(p)\|_1}_{\mathcal{L}_{\text{Depth}}} \quad (6)$$

Pose Tracking Loss. To refine the camera pose E_t at frame t , the set of Gaussians G_{t-1} from the previous mapping step is held fixed. The pose is optimized by minimizing the combined photometric and depth ℓ_1 loss:

$$E_t = \arg \min_E \sum_i \|I_i^{\text{render}}(G_{t-1}, E) - I_i^{\text{gt}}\|_1 + \sum_i \|d_i^{\text{render}}(G_{t-1}, E) - d_i^{\text{gt}}\|_1 \quad (7)$$

where:

- G_{t-1} is the fixed set of 3D Gaussians from the previous frame,
- E is the candidate pose being optimized,
- $I_i^{\text{render}}(G_{t-1}, E)$ and $d_i^{\text{render}}(G_{t-1}, E)$ are the RGB and depth values rendered at pixel i from pose E ,
- I_i^{gt} and d_i^{gt} are the ground-truth RGB and depth at pixel i .

This loss provides the "Per Iteration Tracking/Image Loss" and "Tracking/Depth Loss" metrics logged during training.

Why ℓ_1 Loss?

- Robust to outliers such as occlusions and specularities compared to ℓ_2 loss,
- Simple and fast to compute, especially efficient in CUDA implementations.

The core of the training objective is the combined photometric and depth ℓ_1 loss, computed across all pixels and frames.

4.3 Dataset and Preparation

We plan to evaluate the proposed system on the Replica dataset [8], which offers high-quality synthetic RGB-D images with precise ground-truth camera trajectories. Pre-processing will include normalization of depth maps, synchronization of RGB and depth streams, and calibration of intrinsic parameters for consistent tracking and mapping.

4.4 Evaluation Metrics

Reconstruction quality and tracking accuracy will be measured using:

Peak Signal-to-Noise Ratio (PSNR). Quantifies image reconstruction fidelity [3]:

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\text{MSE}} \right) \quad (\text{dB}) \quad (8)$$

$$\text{MSE} = \frac{1}{HW} \sum_{h=1}^H \sum_{w=1}^W (I_{\text{gt}}(h, w) - I_{\text{rec}}(h, w))^2 \quad (9)$$

Structural Similarity Index (SSIM): Assesses perceptual structural similarity [10]:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (10)$$

- μ_x, μ_y : local means; σ_x^2, σ_y^2 : variances; σ_{xy} : covariance
- $c_1 = (0.01L)^2$, $c_2 = (0.03L)^2$, with $L = 255$ for 8-bit images

Learned Perceptual Image Patch Similarity (LPIPS): Deep feature-based perceptual similarity [12]:

$$\text{LPIPS} = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|\phi^l(I_{\text{gt}})_{h,w} - \phi^l(I_{\text{rec}})_{h,w}\|_2^2 \quad (11)$$

- ϕ^l : activations at layer l of a pretrained VGG network

Absolute Trajectory Error (ATE RMSE): Measures root mean square error of absolute translational differences after optimal trajectory alignment [9]:

$$\text{ATE}_{\text{RMSE}} = \sqrt{\frac{1}{n} \sum_{i=1}^n \|\mathbf{t}_i - \hat{\mathbf{t}}'_i\|_2^2} \quad (\text{cm}) \quad (12)$$

- $\mathbf{t}_i \in \mathbb{R}^3$: ground truth translation
- $\hat{\mathbf{t}}'_i = \mathbf{S}\hat{\mathbf{t}}_i + \mathbf{p}$: aligned estimate
- $\mathbf{S} \in \mathbb{SO}(3)$, $\mathbf{p} \in \mathbb{R}^3$: optimal rigid transform (Horn’s method)
- n : number of poses

Computational Timing Metrics: All reported in milliseconds (ms), averaged over time:

- **Track Frame** – Total time for pose estimation and feature update:

$$T_{\text{track}} = t_{\text{pose_est}} + t_{\text{feature_update}} \quad (13)$$

- **Track Iter** – Per-iteration optimization time:

$$\bar{T}_{\text{track_iter}} = \frac{1}{N_{\text{iters}}} \sum_{k=1}^{N_{\text{iters}}} t_{\text{LM_step}}^{(k)} \quad (14)$$

- **Map Frame** – Time for full mapping pipeline:

$$T_{\text{map}} = t_{\text{bundle_adjust}} + t_{\text{mesh_update}} \quad (15)$$

- **Map Iter** – Per-iteration mapping update:

$$\bar{T}_{\text{map_iter}} = \frac{1}{M_{\text{iters}}} \sum_{k=1}^{M_{\text{iters}}} t_{\text{BA_step}}^{(k)} \quad (16)$$

5 Baseline & Extensions

5.1 Baseline Slection and Evaluation

We choose SplaTAM [4] as our baseline due to its ease of setup and comprehensive documentation. Although RTG-SLAM was considered [1], its higher computational demands and complex dependencies make SplaTAM a more accessible starting point for running multiple experimentation.

Our approach enhances SplaTAM by incorporating two main components that together reduce inference time while preserving reconstruction quality: feature-based camera tracking and adaptive resolution/level-of-detail (LOD) for mapping.

5.1.1 Feature-Based Camera Tracking

Traditional SplaTAM uses iterative gradient descent to refine camera poses. If each iteration takes time t_{iter} and N_{iter} iterations are required, the total time is:

$$T_{\text{grad}} = N_{\text{iter}} \times t_{\text{iter}}.$$

In contrast, our method uses a feature-based tracker with fixed, predictable steps:

- **Feature Extraction:** Keypoints and descriptors are obtained using SuperPoint [2], which takes time t_{feat} .
- **Feature Matching:** LightGlue [5] matches features between consecutive frames in time t_{match} .
- **Initial Pose Estimation:** Robust correspondences are used to compute a rigid transformation via RANSAC and least-squares fitting, taking time t_{ransac} .
- **Pose Refinement:** A lightweight gradient-based optimization refines the pose in time t_{refine} .

The overall tracking time is then:

$$T_{\text{track}} = t_{\text{feat}} + t_{\text{match}} + t_{\text{ransac}} + t_{\text{refine}}.$$

Since T_{track} is fixed and typically much smaller than the iterative T_{grad} , this replacement leads to significant speed-ups in camera tracking.

5.1.2 Adaptive Resolution and Level-of-Detail (LOD)

In the original mapping stage, every pixel is processed at full resolution, leading to a computation time of:

$$T_{\text{full}} = P \times t_{\text{full}},$$

where P is the total number of pixels and t_{full} is the processing time per pixel at full resolution.

Our adaptive strategy computes an error map by comparing the rendered scene with the RGB-D input. Only a fraction f of the pixels—those with high residual error—are processed at full resolution, while the remaining $(1 - f)$ pixels are processed at a lower cost t_{low} (with $t_{\text{low}} < t_{\text{full}}$). The total time becomes:

$$T_{\text{adaptive}} = f \times P \times t_{\text{full}} + (1 - f) \times P \times t_{\text{low}}.$$

Because t_{low} is much smaller than t_{full} and f is typically less than 1, this adaptive approach reduces the mapping computation substantially.

Combined Effect

The overall inference time of the original system is approximately:

$$T_{\text{orig}} = T_{\text{grad}} + T_{\text{full}},$$

whereas our improved system runs in:

$$T_{\text{improved}} = T_{\text{track}} + T_{\text{adaptive}}.$$

Since both $T_{\text{track}} \ll T_{\text{grad}}$ and $T_{\text{adaptive}} \ll T_{\text{full}}$, the combined effect results in a much lower overall inference time, enabling real-time performance without sacrificing reconstruction quality.

5.2 Implemented Extensions / Experiments

We tested three key extensions to the original SplaTAM tracking module, replacing gradient-based pose optimization with a fast, feature-based tracking pipeline. Each experiment introduces a variation in how local features are extracted and combined before matching with LightGlue.

Experiment 1: SuperPoint + LightGlue for Feature-Based Tracking

In this experiment, we extract 2D keypoints and descriptors from each RGB frame using SuperPoint. These descriptors are then matched between consecutive frames using LightGlue, a fast attention-based feature matcher [5].

From the resulting correspondences, we estimate the relative pose using PnP with RANSAC [7]. This estimated pose is used as an initialization for the subsequent SplaTAM mapping step, where the RGB-D loss is minimized and the 3D Gaussians are updated.

This setup serves as our base feature matching pipeline and allows us to test whether LightGlue+SuperPoint can replace the original tracking module while maintaining or improving performance. This experiment achieved a result comparable to that of the baseline model in all evaluation metrics.

Experiment 2: Uniform Feature Soup + LightGlue

Here, we extend the previous setup by combining multiple local feature extractors into a uniform model soup [11]. We extract descriptors from five pretrained models recommended by FlashSLAM: [7]: **SuperPoint**, **DISK**, **ALIKED**, **SIFT**, and **DoGHardNet**

For each keypoint, we average the descriptors of all extractors to form a single unified representation or a model soup [11]. These soup descriptors are then matched between frames using LightGlue, and pose estimation is carried out as before.

This experiment explores whether combining multiple descriptors improves tracking robustness without increasing inference cost, since LightGlue still operates over one set of averaged descriptors. The method reduced the inference time by keeping the quality metrics similar to the baseline.

Experiment 3: Greedy Feature Soup + LightGlue

Instead of uniformly averaging all descriptors, this experiment uses a greedy selection strategy. For each frame, we evaluate the matching performance of each individual feature extractor (SuperPoint, DISK, ALIKED, SIFT, DogHardNet) when combined with LightGlue.

Starting from the best single extractor, we iteratively add other extractors to the soup only if they improve the matching quality or inference stability. The final descriptor is a weighted average of only the selected extractors.

This greedy soup approach is inspired by the model soups technique [11] and aims to balance speed and quality by only including features that provide additional value.

Other Experiments

Pose Refinement (Not Effective)

To improve inference time and accuracy, we implemented a pose refinement module that optimizes the initial pose estimate $\hat{T} \in SE(3)$ using rendered outputs. The refinement minimizes a weighted loss combining depth and silhouette consistency [13]:

$$\mathcal{L}_{\text{total}} = \lambda_d \mathcal{L}_{\text{depth}} + \lambda_s \mathcal{L}_{\text{silhouette}} \quad (17)$$

Here, $\mathcal{L}_{\text{depth}}$ penalizes depth discrepancies, and $\mathcal{L}_{\text{silhouette}}$ aligns rendered masks with observed silhouettes.

Despite its potential, this module performed worse than the SplaTAM baseline in both accuracy and speed. We found the refinement step to be sensitive to initialization and added unnecessary computational overhead. Further tuning and validation are needed to make this approach viable.

Adaptive Resolution Mapping (Not Effective)

Another experiment explored an adaptive resolution strategy for Gaussian map updates. After dense tracking, we computed a residual heatmap and rendered high-error tiles at full resolution, while others remained at lower scale. The split was controlled by a percentile threshold—tiles above the 70th percentile of residuals were upsampled, and low-error tiles remained at $\frac{1}{2}\times$ scale. Though this method was successful in the original SplaTAM paper, our attempt led to poor results (PSNR ≈ 13), likely due to excessive Gaussian removal and instability in early frames. The residual threshold lagged under rapid lighting changes, and we plan to tune the percentile selection dynamically in future work.

6 Results & Analysis

6.1 Results

Exp 1 – SuperPoint + LightGlue: As shown in Table 2 and Table 3, this configuration performs comparably to the baseline in terms of both quality and speed. The depth L1 (0.488 cm) and LPIPS (0.071) are nearly identical to SplaTAM-F, with minor improvements in some metrics (MS-SSIM = 0.975). However, PSNR slightly drops from 32.56 dB to 32.53 dB, and ATE increases from 0.271 cm to 0.280 cm. Runtime is marginally better than the baseline (tracking per frame: 3.07 ms vs. 3.29 ms), but the overall gain is minimal. This suggests that replacing the gradient-based tracking with SuperPoint+LightGlue is viable but not clearly advantageous.

Exp 2 – Uniform Model Soup: This method stands out in Table 3 for offering the best runtime performance across all experiments (e.g., tracking/frame: **3.04 ms**, mapping/frame: **5.61 ms**). At the same time, Table 1 shows that it maintains similar reconstruction quality to the baseline. LPIPS (0.18), SSIM (0.94), and ATE (0.39 cm) remain within a reasonable range, with only a slight drop in PSNR (28.96 dB vs. 29.23 dB). The experiment confirms that averaging features from diverse extractors can enhance efficiency without a significant sacrifice in accuracy.

Exp 3 – Greedy Model Soup: This adaptive fusion strategy improves upon the uniform soup in quality (e.g., LPIPS reduced from 0.189 to 0.158, MS-SSIM improved from 0.944 to 0.961), but still does not surpass Exp 2 overall. Runtime is slightly slower (tracking/frame: 3.30 ms) and ATE remains higher than the baseline (0.371 cm). While the greedy approach shows potential for dynamic

feature selection, it adds complexity without yielding better trade-offs than the uniform soup in our current setup.

Table 1: Performance Comparison on ROOM_0 (V100 GPU): SplaTAM Baseline vs. Light-Glue+Extractor Variant

Method	ATE (cm)	PSNR (dB)	SSIM	LPIPS	Track Frame (ms)	Track Iter (ms)	Map Frame (ms)	Map Iter (ms)
SplaTAM	0.34	29.23	0.94	0.18	3.29	82.28	6.03	100.55
Exp 2	0.39	28.96	0.94	0.18	3.07	76.87	5.67	94.57

Table 2: Quality evaluation on SplaTAM extensions compared to the SplaTAM-F baseline. All experiments were run on an NVIDIA H100 80GB GPU.

Metric (better = bold)	SplaTAM-F	Exp 1	Exp 2	Exp 3
Depth L1 (cm) ↓	0.494	0.488	0.892	0.820
PSNR (dB) ↑	32.56	32.53	28.97	29.92
LPIPS ↓	0.072	0.071	0.189	0.158
MS-SSIM ↑	0.974	0.975	0.944	0.961
ATE-RMSE (cm) ↓	0.271	0.280	0.396	0.371

Table 3: Speed comparison of SplaTAM-F and its extensions. Experiments 1 and 2, as well as the baseline (Base-F), were run on a Tesla V100-SXM2-32GB GPU.

heightRun	Tracking / Frame (ms)	Tracking / Iteration (ms)	Mapping / Frame (ms)	Mapping / Iteration (ms)
SplaTAM-F	3.29	82.28	6.03	100.55
Exp 1	3.07	76.87	5.67	94.57
Exp 2	3.04	75.98	5.61	93.5
Exp 3	3.30	82.60	6.10	99.9

This is the link to the wandb log of our extension:

<https://wandb.ai/lilina-moa-carnegie-mellon-university/SplaTAM/runs/cw1t861i>

<https://wandb.ai/breaking-b4d/SplaTAM?nw=nwuserlilina-moa>

Overall, Experiment 2 offers the best speed-quality balance across all experiments, as supported by both runtime (Table 3) and quality metrics (Table 2). It is a strong candidate for replacing the baseline in time-sensitive applications.

6.2 Discussion

We visualized evaluation metrics and key findings through tables comparing quality and runtime across experiments (Tables 1-3), and the generated reconstruction output is shown in Figure 1. To ensure fair comparisons, we ran both our baseline (SplaTAM-F) and all extensions on the same GPU models—V100 and H100—since different hardware produced noticeably different results. We also observed that running multiple experiments concurrently on shared GPU resources increased runtime, which limited how many tests we could perform. Despite this, the work remains cohesive and focused on improving tracking speed without compromising reconstruction quality. We plan to address these performance inconsistencies by isolating runs more effectively in future experiments.

7 Future Directions

Future work will focus on:

1. Integrating mixed-precision to further reduce compute time. We propose implementing model quantisation. A simple yet effective first step would be switching computations from float32 to float16, which could substantially reduce both model size and inference time.
2. Extending adaptive strategies to include dynamic Gaussian culling based on the context of the scene.

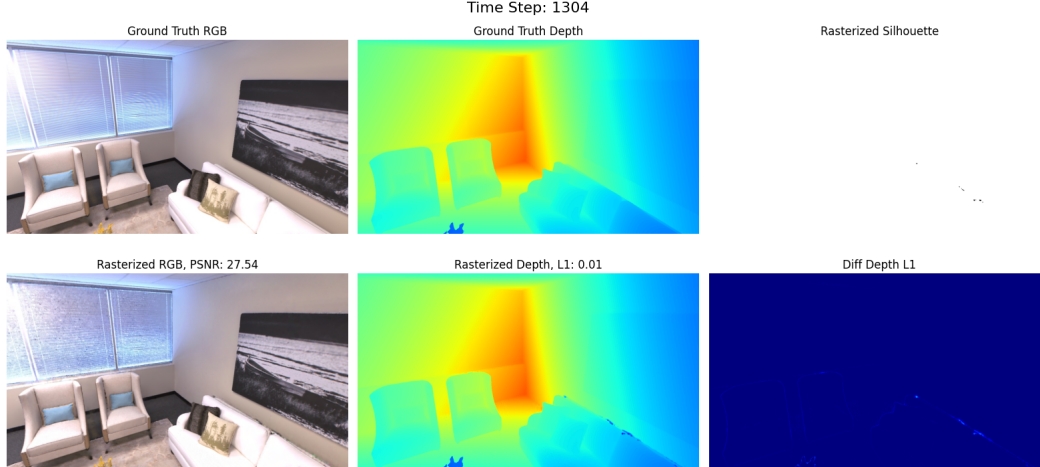


Figure 1: Comparison of ground truth and rasterized outputs for RGB and depth at Time Step 1304, evaluating reconstruction accuracy using PSNR, L1 loss, and depth difference

3. Reducing the resolution gap between the generated 3D reconstruction is indeed a priority. Surely, adaptive resolution could be one way of tackling this issue without sacrificing inference time and computational resource demand. Another technique that is gaining popularity in the field of 3D reconstruction is Super-Resolution Techniques [6]. Some studies in this field have achieved high precision and recall scores.
4. Evaluating the proposed extensions on additional benchmarks such as TUM-RGBD and Replica to validate generalization. TUM-RGBD has become something of a standard benchmark, alongside datasets like Replica, as it has been widely adopted by previous radiance-field-based SLAM methods for testing and evaluation
5. Improving Scene quality: we plan to explore better ways to improve scene quality by refining only the Gaussians in regions with high visual error. This way, we could enhance detail without significantly increasing memory or computation.

8 Conclusion

The goal of this project was to improve the inference speed of SplatAM without reducing the 3D reconstruction quality. We explored three feature-based tracking extensions using LightGlue with SuperPoint, uniform model soup, and greedy model soup strategies. Our results show that the uniform soup (Exp 2) achieves the best trade-off, significantly reducing runtime while maintaining comparable reconstruction quality. While the SuperPoint setup (Exp 1) performed similarly to the baseline and the greedy soup (Exp 3) showed potential, neither outperformed the uniform soup overall. These findings suggest that combining features from multiple extractors is a promising direction for making SLAM faster and more efficient.

Administrative Details

References

- [1] RTG-SLAM: Real-time 3D Reconstruction at Scale using Gaussian Splatting. In *ACM SIGGRAPH 2024 Conference Papers*, 2024. Accessed: Feb. 17, 2025.
- [2] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. *CoRR*, abs/1712.07629, 2017.
- [3] Quang Huynh-Thu and Mohammed Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics letters*, 44(13):800–801, 2008.
- [4] 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. *arXiv preprint arXiv:2312.02126*, 2023.
- [5] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys. Lightglue: Local feature matching at light speed. In *Proceedings of CVPR*, 2023.
- [6] Antigoni Panagiotopoulou, Lazaros Grammatikopoulos, Andreas El Saer, Elli Petsa, E. Charou, Lemonia Ragia, and George Karras. Super-resolution techniques in photogrammetric 3d reconstruction from close-range uav imagery. *Heritage*, 6:2701–2715, 03 2023.
- [7] P. Pham, D. Conover, and A. Bera. Flashslam: Accelerated rgb-d slam for real-time 3d scene reconstruction with gaussian splatting, 2024. Available at <https://flashslam.github.io>.
- [8] J. Straub et al. The replica dataset: A digital replica of indoor spaces, 2019. arXiv:1906.05797.
- [9] Jürgen Sturm, Niko Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. *IROS*, 2012.
- [10] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.
- [11] M. Wortsman et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 17–23 Jul 2022.
- [12] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018.
- [13] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. *CoRR*, abs/1704.07813, 2017.

Team Contributions:

- Galane Basha Namomsa - Baseline setup, Model soup experiments and Report writing.
- Peng Boris Akebuon - LightGLue + Superpoint, Adaptive Resolution experiments and Report writing.
- Matia Mulumba Mukasa - Pose Refinement experiments, Report writing.
- Isaac Thani Moses - Attended the meeting, Report writing.

Our Github Repository

https://github.com/3D-NavRobot/SplaTam_replica