DynaMiTE: Towards robust 2D Gaussian Splatting based camera localization

Sergio De León[1], J-B Hayet[1]
[1]CIMAT

1. Introduction

In this work we tackle the problem of visual simultaneous localization and mapping **(SLAM)**. With the new techniques in the area of novel view synthesis, like Nerf and **Gaussian Splatting**, visual SLAM has acquire new capabilities like photometric reconstruction [3] and semantic scene understanding. Despite the new developments, current algorithms lack of robustness in handling dynamic elements in the scene. In this work we propose a **learned probabilistic module** to disentangle dynamic from static regions that operates continually.

2. Preliminaries

2.1 2D Gaussian Splatting

2D Gaussian Splatting [1] **(2DGS)** is a variant of the 3D original version which aims to solve some of the previous limitations. Instead of use ellipsoids as explicit world representation, 2DGS uses **tangent disks** to the scenes' surfaces, each disk is specified by its central point $\mathbf{p}_c$, two principal tangential vectors $\mathbf{t}_u, \mathbf{t}_v$, and two scalars (s_u, s_v): $\mathbf{p}(u,v) = \mathbf{p}_c + s_u\mathbf{t}_u u + s_v\mathbf{t}_v v$. Based on this parameterization, we can define the normal vector for each disk as $\mathbf{t}_w = \mathbf{t}_u \times \mathbf{t}_v$. This parametrization produce an accurate "***splatting***" of the disks to the image plane by solving for the intersection of three non-parallel planes. This approach fix the view dependent inconsistencies found in 3D Gaussian Splatting, additionally 2DGS enable robust geometry retrieval as depicted in figure 1.

Figure 1: 2DGS view-dependent inconsistency and geometry correction.

As well as in the original version 2DGS trains a scene by minimizing the **photometric loss** between the original frame and the alpha-blended rasterization, given the world Gaussians and the camera poses. To ensure that the disks are in fact align with the surfaces in the scene, 2DGS adds a regularization term to the loss function. The **normal consistency** term leverages the depth signal from an RGB-D camera by aligning the disks normals with the actual normals within the scene.

2.2 Gaussian Processes

Intuitively a Gaussian Process is an extension of a multivariate normal distribution to **infinite number of dimensions** , formally let $\mathcal{F} = \{f(x)\}_{x\in\mathsf{D}}$ be a collection of random variables parametrized by a continuous parameter x. $\mathcal{F}$ is said to be a Gaussian Process if any finite subset of $\mathcal{F}$ follows a **multivariate Gaussian distribution**, that is, for any $x, x' \in \Omega$, we have:

$$f(x), f(x') \sim \mathcal{N}\left(\begin{bmatrix} m(x) \\ m(x') \end{bmatrix}, \begin{bmatrix} k(x,x) & k(x,x') \\ k(x',x) & k(x',x') \end{bmatrix}\right), \tag{1}$$

for some mean and covariance functions $m : \Omega \to \mathbb{R}$, $k : \Omega \times \Omega \to \mathbb{R}$.

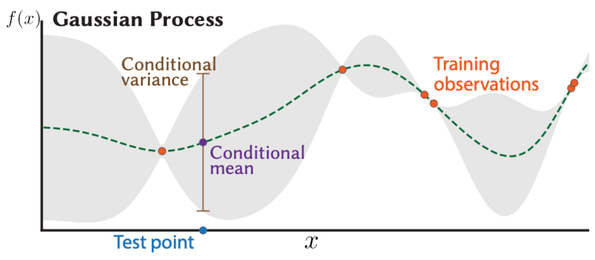

Figure 2: Gaussian process modeling. We can naturally compute the uncertainty of the prediction.

Given a training dataset $\mathcal{T} = \{(x_i, f(x_i))\}_{i=1}^n$, by the **Bayes' theorem** we can compute the **posterior distribution** of a new evaluation point $f(x')$ which is a normal distribution as well, but it requires to compute the inverse of the evaluation matrix of the prior k over the training points, this computation scales as $O(n^3)$, which is **computationally intensive** for the pixels in an image. In our methodology we will follow an approximate approach.

3. Methodology

3.1 2D Gaussian Splatting-based camera localization system

We leverage the modern architecture for SLAM and split the system in two main threads, the **frontend** that keeps track of the camera poses and the **backend** that refine the poses and train the scene with Gaussian Splatting, we add a **third module** that estimates the masks of dynamic elements.

3.2 Frontend

The frontend is responsible for estimating the **camera pose** of the current input frame, -a camera pose is a rigid body transformation $\mathsf{T} \in \mathsf{SE}(3)$ that transforms from a global framework to the local camera coordinates. Does so by solving the following optimization problem

$$\min_{\mathsf{T}\in\mathsf{SE}(3)} ||\mathbf{I} - \bar{\mathbf{I}}(\mathcal{G}, \mathsf{T})||, \tag{2}$$

where $\mathbf{I}, \bar{\mathbf{I}}$ are the input and rasterize frame correspondingly, the later depends on the current Gaussians $\mathcal{G}$, which are assume constant during tracking. This minimization task is performed by the Adam optimizer. The frontend must run at nearly real-time if possible.

3.3 Backend

The backend runs slowly and is responsible for **training the scene with 2DGS** as explained in 2.1, and for refining the camera poses as well. It operates in a sliding window for efficiency.

Since performing Gaussian Splatting optimization with all the input frames will be a computational heavy operation, the system make a **keyframe selection** based on frames spatial distribution and relevant information of the scene show in that frame.

3.4 Dynamics masking module

The aim of this modules is to automatically detect the dynamic elements in the scene. For each new key-frame we compute a robust residual between original and rasterize image given by $r = (1-\mathsf{C})(1-\mathsf{L})(1-\mathsf{S})$ where C, L, S are the contrast, luminescence and structure elements of the SSIM loss. This robust residual captures finer variations between images than just computing the l_1 loss. We assume that this residual is parametrized by some semantic features $z \in \mathbb{R}^d$ like the generated by DINOv2. The dataset given by $\mathcal{D} = \{(z_i, r(z_i))\}$ is use to train a Gaussian process. We use a Matern covariance kernel plus a simple RBF. We adopt a variational approach where a surrogate posterior distribution is approximate by minimizing the ELBO with stochastic gradient descent. To keep a small and compact dataset we actively select the points with the highest values and maintain a sliding window. The final binary mask is produce by thresholding the exponential moving average of the mean predicted value.

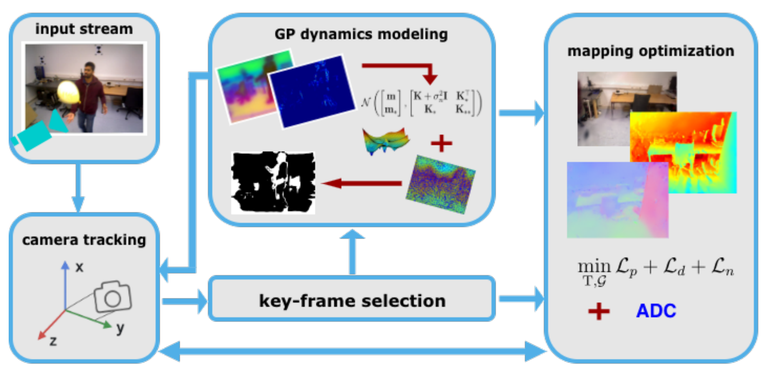

Figure 3: DynaMiTE pipeline

4. Results

4.1 Experiments and dataset

We conducted a set of experiments at the Bonn RGB-D dynamic dataset. We take as baseline WildGS [4]. It is worth mentioning that this method assumes we have access to all frames before computation. We implemented our method in PyTorch and make use of the custom cuda kernels and camera pose derivatives implemented in [2]

4.2 Quantitative results

We report the absolute trajectory error in the following table. Additionally we include an ablation study (indicated as AblationModel) where we turn off the dynamics masking module.

Method	balloon	crowd	moving	person
WildGS	2.8	1.5	1.6	3.1
Ours	21.2	55.2	3.5	30.0
AblationModel	31.2	46.5	26.3	48.8

5. Conclusions

In our ablation study we observe that the dynamics masking module systematically improves the trajectory estimation quantitative and qualitative, in contrast with WildGS that works better by a large margin across all scenes our method performs continual learning by adjusting the parameters of the Gaussan process as it requires by the scene, which is a key feature for autonomous physical agents.

References

[1] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*, 2024.

[2] Hidenobu Matsuki, Gwangbin Bae, and Andrew J. Davison. 4dtam: Non-rigid tracking and mapping via dynamic surface gaussians. *arXiv preprint arXiv:2505.22859v1*, 2025.

[3] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[4] Jianhao Zheng, Zihan Zhu, Valentin Bieri, Marc Pollefeys, Songyou Peng, and Iro Armeni. Wildgs-slam: Monocular gaussian splatting slam in dynamic environments. *arXiv preprint arXiv:2504.03886v1*, 2025.