Semantic & Neural Rendering & SLAM

Research Notes & Literature Review

Shuqi XIAO

July 3, 2024

Outline

3DGS-based SLAM

Timeline

11/2023 - 12/2024

- SplaTAM [1]
- GS-SLAM [2]
- Gaussian-SLAM [3]
- MonoGS [4]

01/2024 - 04/2024

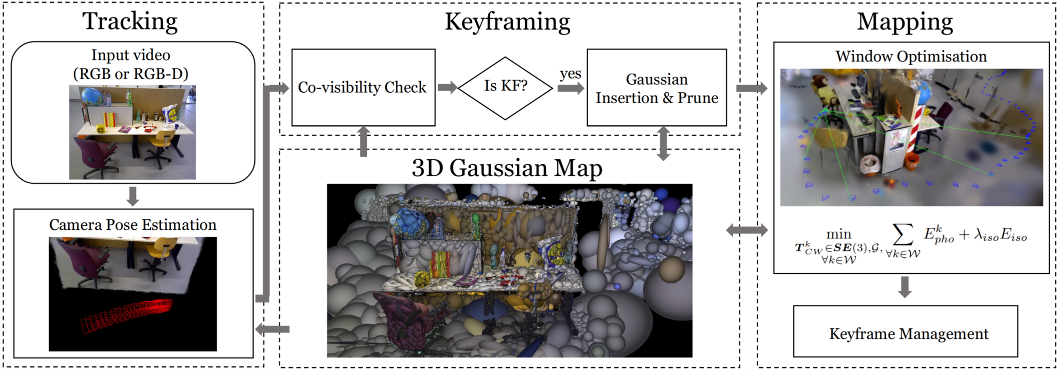
05/2024 - 06/2024

accepted to top conferences

3DGS-based SLAM

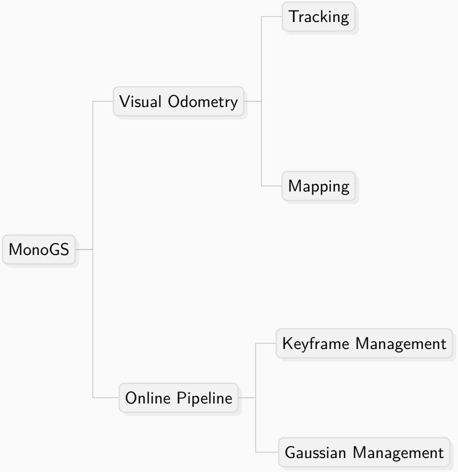
MonoGS

Framework



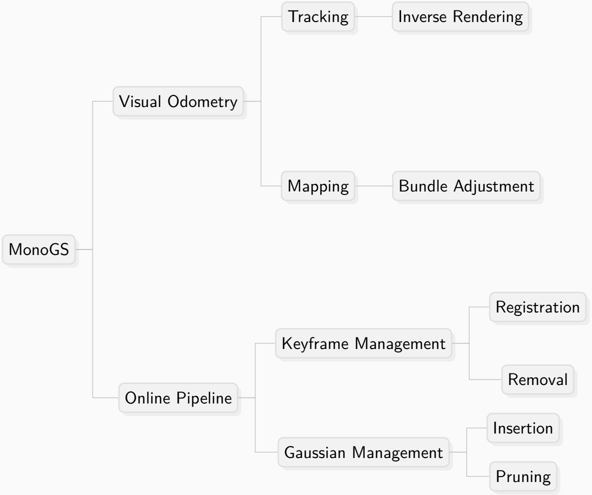
(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Taxonomy of Methodology



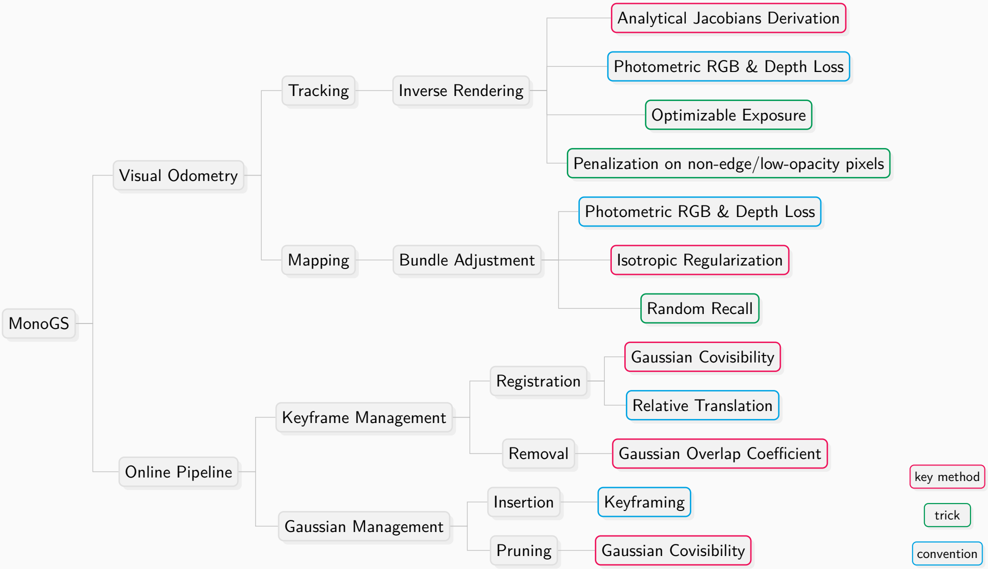
(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Taxonomy of Methodology



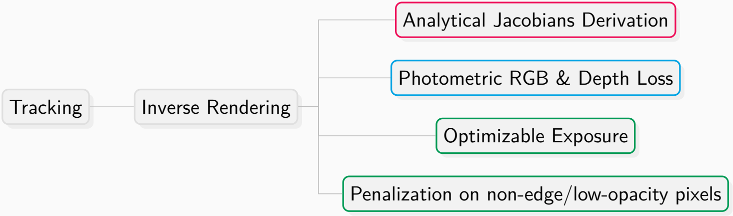
(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Taxonomy of Methodology



(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Overview



Track camera poses by inverse rendering,

- through the extended differentiable rendering pipeline,
- by a direct optimization against fixed 3D Gaussians,
- with some tricks to be more adaptive to brightness and more robust to noise.

key method   trick   convention

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Overview

- Tracking
  - Inverse Rendering
    - Analytical Jacobians Derivation
    - Photometric RGB & Depth Loss
    - Optimizable Exposure
    - Penalization on non-edge/low-opacity pixels

Track camera poses by inverse rendering,

- through the extended differentiable rendering pipeline,
- by a direct optimization against fixed 3D Gaussians,
- with some tricks to be more adaptive to brightness and more robust to noise.

key method trick convention

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Overview

- Tracking
  - Inverse Rendering
    - Analytical Jacobians Derivation
    - Photometric RGB & Depth Loss
    - Optimizable Exposure
    - Penalization on non-edge/low-opacity pixels

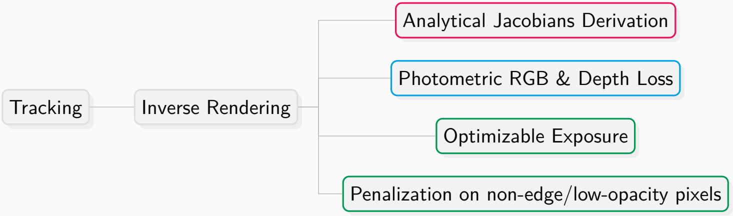
Track camera poses by inverse rendering,

- through the extended differentiable rendering pipeline,
- by a direct optimization against fixed 3D Gaussians,
- with some tricks to be more adaptive to brightness and more robust to noise.

key method   trick   convention

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Overview



Track camera poses by inverse rendering,

- through the extended differentiable rendering pipeline,
- by a direct optimization against fixed 3D Gaussians,
- with some tricks to be more adaptive to brightness and more robust to noise.

key method trick convention

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Derivation of Jacobians i

Firstly, let's review the projection of 3D Gaussians.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Firstly, let's review the projection of 3D Gaussians.

$$\mathcal{N}\left(\mu_w, \Sigma_w\right) \overset{\pi}{\mapsto} \mathcal{N}\left(\mu_i, \Sigma_i\right) \tag{1}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Firstly, let's review the projection of 3D Gaussians.

$$\mathcal{N}\left(\mu_w, \Sigma_w\right) \xrightarrow{\pi} \mathcal{N}\left(\mu_i, \Sigma_i\right) \tag{1}$$

is achieved by

$$\mu_i = \pi\left(\mathbf{T}_{cw} \cdot \mu_w\right) \tag{2}$$

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \tag{3}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Derivation of Jacobians i

Firstly, let's review the projection of 3D Gaussians.

$$\mathcal{N}\left(\mu_w, \Sigma_w\right) \xrightarrow{\pi} \mathcal{N}\left(\mu_i, \Sigma_i\right) \tag{1}$$

is achieved by

$$\mu_i = \pi\left(\mathbf{T}_{cw} \cdot \underbrace{\mu_w}_{\in \mathbb{P}^3,\ \text{3D(world) mean}}\right) \tag{2}$$

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \tag{3}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Derivation of Jacobians i

Firstly, let's review the projection of 3D Gaussians.

$$\mathcal{N}\left(\mu_w, \Sigma_w\right) \overset{\pi}{\mapsto} \mathcal{N}\left(\mu_i, \Sigma_i\right) \tag{1}$$

is achieved by

$$\mu_i = \pi\left(\underbrace{\mathbf{T}_{cw}}_{\in \mathrm{SE}(3),\ \text{camera pose}} \cdot \underbrace{\mu_w}_{\in \mathbb{P}^3,\ \text{3D(world) mean}}\right) \tag{2}$$

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \tag{3}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Derivation of Jacobians i

Firstly, let's review the projection of 3D Gaussians.

$$\mathcal{N}\left(\mu_w, \Sigma_w\right) \overset{\pi}{\mapsto} \mathcal{N}\left(\mu_i, \Sigma_i\right) \tag{1}$$

is achieved by

$$\mu_i = \underbrace{\pi}_{\text{projection}}\left(\underbrace{\mathbf{T}_{cw}}_{\in \mathrm{SE}(3),\ \text{camera pose}} \cdot \underbrace{\mu_w}_{\in \mathbb{P}^3,\ \text{3D(world) mean}}\right) \tag{2}$$

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \tag{3}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Derivation of Jacobians i

Firstly, let's review the projection of 3D Gaussians.

$$\mathcal{N}\left(\mu_w, \Sigma_w\right) \overset{\pi}{\mapsto} \mathcal{N}\left(\mu_i, \Sigma_i\right) \tag{1}$$

is achieved by

$$\mu_i = \pi\left(\mathbf{T}_{cw} \cdot \mu_w\right) \tag{2}$$

$\in \mathbb{P}^3$, 3D(world) mean

$\in \mathrm{SE}(3)$, camera pose

projection

$\in \mathbb{P}^2$, 2D(image) mean

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \tag{3}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Firstly, let's review the projection of 3D Gaussians.

$$\mathcal{N}(\mu_w, \Sigma_w) \xrightarrow{\pi} \mathcal{N}(\mu_i, \Sigma_i) \tag{1}$$

is achieved by

$$\underbrace{\mu_i}_{\in \mathbb{P}^2,\ \text{2D(image) mean}} = \underbrace{\pi}_{\text{projection}} \left(\underbrace{\mathbf{T}_{cw}}_{\in \mathrm{SE}(3),\ \text{camera pose}} \cdot \underbrace{\mu_w}_{\in \mathbb{P}^3,\ \text{3D(world) mean}} \right) \tag{2}$$

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \underbrace{\Sigma_w}_{\in \mathbb{R}^{3\times 3},\ \text{3D(world) covariance}} \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \tag{3}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Derivation of Jacobians i

Firstly, let's review the projection of 3D Gaussians.

$$\mathcal{N}(\mu_w, \Sigma_w) \overset{\pi}{\mapsto} \mathcal{N}(\mu_i, \Sigma_i) \tag{1}$$

is achieved by

$$\underbrace{\mu_i}_{\in \mathbb{P}^2,\ \text{2D(image) mean}} = \underbrace{\pi}_{\text{projection}} \left(\underbrace{\mathbf{T}_{cw}}_{\in \mathrm{SE}(3),\ \text{camera pose}} \cdot \underbrace{\mu_w}_{\in \mathbb{P}^3,\ \text{3D(world) mean}} \right) \tag{2}$$

$$\Sigma_i = \mathbf{J}_\pi \underbrace{\mathbf{R}_{cw}}_{\in \mathrm{SO}(3),\ \text{rotation component of } \mathbf{T}_{cw}} \underbrace{\Sigma_w}_{\in \mathbb{R}^{3\times 3},\ \text{3D(world) covariance}} \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \tag{3}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Derivation of Jacobians i

Firstly, let's review the projection of 3D Gaussians.

$$\mathcal{N}(\mu_w, \Sigma_w) \stackrel{\pi}{\mapsto} \mathcal{N}(\mu_i, \Sigma_i) \tag{1}$$

is achieved by

$$\mu_i = \pi\left(\mathbf{T}_{cw} \cdot \mu_w\right) \tag{2}$$

- μ_w: $\in \mathbb{P}^3$, 3D(world) mean
- $\mathbf{T}_{cw}$: $\in \mathrm{SE}(3)$, camera pose
- π: projection
- μ_i: $\in \mathbb{P}^2$, 2D(image) mean

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \tag{3}$$

- Σ_w: $\in \mathbb{R}^{3\times 3}$, 3D(world) covariance
- $\mathbf{R}_{cw}$: $\in \mathrm{SO}(3)$, rotation component of $\mathbf{T}_{cw}$
- $\mathbf{J}_\pi$: $\in \mathbb{R}^{2\times 3}$, Jacobian of the linear approximation of π

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Firstly, let's review the projection of 3D Gaussians.

$$\mathcal{N}\left(\mu_w, \Sigma_w\right) \stackrel{\pi}{\mapsto} \mathcal{N}\left(\mu_i, \Sigma_i\right) \tag{1}$$

is achieved by

$$\mu_i = \pi\left(\mathbf{T}_{cw} \cdot \mu_w\right) \tag{2}$$

- μ_w: $\in \mathbb{P}^3$, 3D(world) mean
- $\mathbf{T}_{cw}$: $\in \mathrm{SE}(3)$, camera pose
- π: projection
- μ_i: $\in \mathbb{P}^2$, 2D(image) mean

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \tag{3}$$

- Σ_w: $\in \mathbb{R}^{3\times 3}$, 3D(world) covariance
- $\mathbf{R}_{cw}$: $\in \mathrm{SO}(3)$, rotation component of $\mathbf{T}_{cw}$
- $\mathbf{J}_\pi$: $\in \mathbb{R}^{2\times 3}$, Jacobian of the linear approximation of π
- Σ_i: $\in \mathbb{R}^{2\times 2}$, 2D(image) covariance

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

The chain rule,

$$\frac{\partial \mu_i}{\partial \mathbf{T}_{cw}} = \frac{\partial \mu_i}{\partial \mu_c} \frac{\partial \mu_c}{\partial \mathbf{T}_{cw}} \tag{4}$$

$$\frac{\partial \Sigma_i}{\partial \mathbf{T}_{cw}} = \frac{\partial \Sigma_i}{\partial \mathbf{J}_\pi} \frac{\partial \mathbf{J}_\pi}{\partial \mu_c} \frac{\partial \mu_c}{\partial \mathbf{T}_{cw}} + \frac{\partial \Sigma_i}{\partial \mathbf{R}_{cw}} \frac{\partial \mathbf{R}_{cw}}{\partial \mathbf{T}_{cw}} \tag{5}$$

The Lie Algebra,

$$\frac{\partial \mu_c}{\partial \mathbf{T}_{cw}} = \begin{bmatrix} \mathbf{I} & -\mu_c^\times \end{bmatrix} \tag{6}$$

$$\frac{\partial \mathbf{R}_{cw}}{\partial \mathbf{T}_{cw}} = \begin{bmatrix} \mathbf{0} & -\mathbf{R}_{cw}^\times(:,1) \\ \mathbf{0} & -\mathbf{R}_{cw}^\times(:,2) \\ \mathbf{0} & -\mathbf{R}_{cw}^\times(:,3) \end{bmatrix} \tag{7}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Tracking: Derivation of Jacobians ii

The chain rule,

$$\frac{\partial \mu_i}{\partial \mathbf{T}_{cw}} = \frac{\partial \mu_i}{\partial \mu_c} \frac{\partial \mu_c}{\partial \mathbf{T}_{cw}} \tag{4}$$

$$\frac{\partial \Sigma_i}{\partial \mathbf{T}_{cw}} = \frac{\partial \Sigma_i}{\partial \mathbf{J}_\pi} \frac{\partial \mathbf{J}_\pi}{\partial \mu_c} \frac{\partial \mu_c}{\partial \mathbf{T}_{cw}} + \frac{\partial \Sigma_i}{\partial \mathbf{R}_{cw}} \frac{\partial \mathbf{R}_{cw}}{\partial \mathbf{T}_{cw}} \tag{5}$$

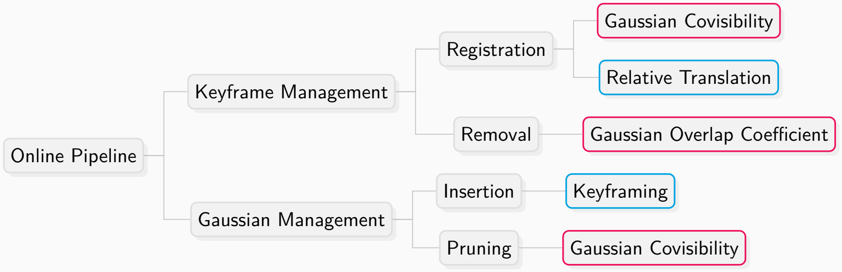
The Lie Algebra,

$$\frac{\partial \mu_c}{\partial \mathbf{T}_{cw}} = \begin{bmatrix} \mathbf{I} & -\mu_c^{\times} \end{bmatrix} \tag{6}$$

$$\frac{\partial \mathbf{R}_{cw}}{\partial \mathbf{T}_{cw}} = \begin{bmatrix} \mathbf{0} & -\mathbf{R}_{cw}^{\times}(:,1) \\ \mathbf{0} & -\mathbf{R}_{cw}^{\times}(:,2) \\ \mathbf{0} & -\mathbf{R}_{cw}^{\times}(:,3) \end{bmatrix} \tag{7}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Online Pipeline: Overview



Keyframe Management:

- Classic strategies, e.g. covisibility & overlap, from DSO [5].
- Off-the-shelf occlusion-aware Gaussian visibility is leveraged to construct metrics.

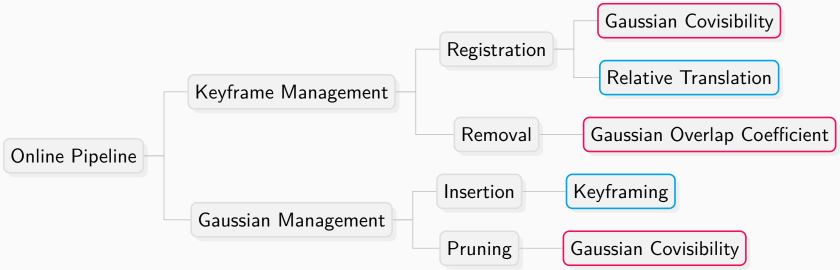
---

key method trick convention

(arXiv, 2016) DSO: Direct Sparse Odometry

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Online Pipeline: Overview



Keyframe Management:

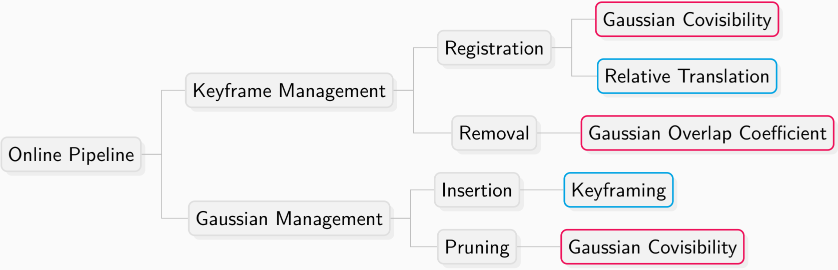
- Classic strategies, e.g. covisibility & overlap, from DSO [5].
- Off-the-shelf occlusion-aware Gaussian visibility is leveraged to construct metrics.

key method | trick | convention

(arXiv, 2016) DSO: Direct Sparse Odometry

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Online Pipeline: Overview



Keyframe Management:

- Classic strategies, e.g. covisibility & overlap, from DSO [5].
- Off-the-shelf occlusion-aware Gaussian visibility is leveraged to construct metrics.

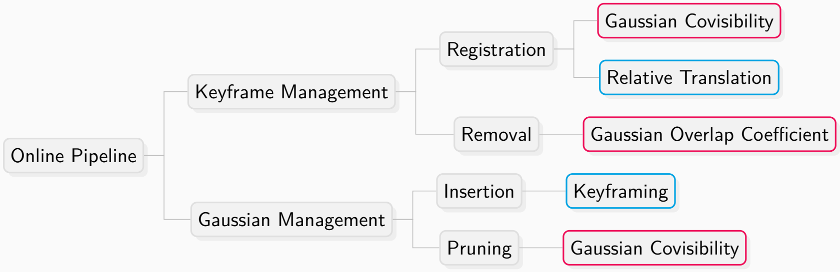
---

key method trick convention

(arXiv, 2016) DSO: Direct Sparse Odometry

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Online Pipeline: Overview



Gaussian Management:

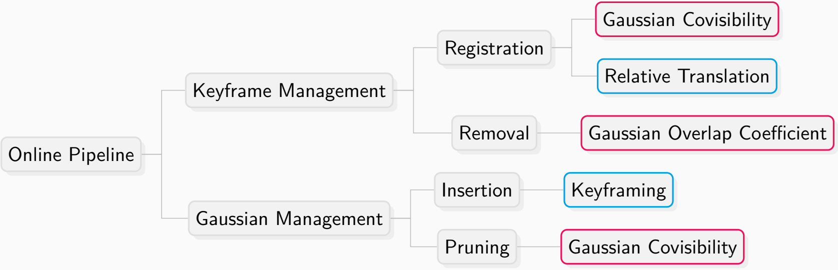
- Insertion: triggered by keyframing, followed by Gaussian initialization.
- Pruning: to remove unstable/incorrect Gaussians by covisibility in a monocular setting.

key method trick convention

(arXiv, 2016) DSO: Direct Sparse Odometry

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Online Pipeline: Overview



Gaussian Management:

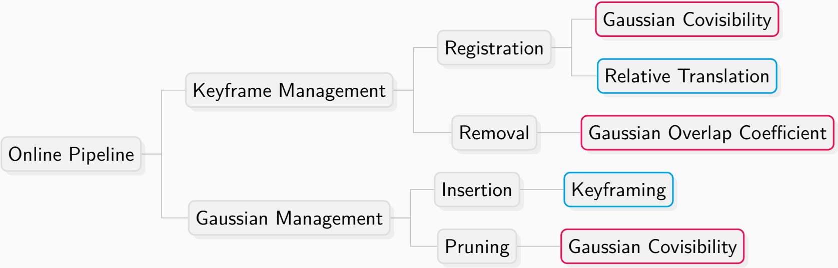
- Insertion: triggered by keyframing, followed by Gaussian initialization.
- Pruning: to remove unstable/incorrect Gaussians by covisibility in a monocular setting.

key method | trick | convention

(arXiv, 2016) DSO: Direct Sparse Odometry

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Online Pipeline: Overview



Gaussian Management:

- Insertion: triggered by keyframing, followed by Gaussian initialization.
- Pruning: to remove unstable/incorrect Gaussians by covisibility in a monocular setting.

---

key method trick convention

(arXiv, 2016) DSO: Direct Sparse Odometry

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Prerequistes

1. What is keyframing or keyframe management?
   - A strategy of selecting and utilizing a crucial subset of frames.
2. Why do we need keyframing?
   - Infeasible to optimize jointly on all frames online.
3. How should we select keyframes?
   - non-redundant and observing the same area.
   - spanning a wide baseline for better multi-view constraints.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Prerequistes

1. What is keyframing or keyframe management?
   - A strategy of selecting and utilizing a crucial subset of frames.
2. Why do we need keyframing?
   - Infeasible to optimize jointly on all frames online.
3. How should we select keyframes?
   - non-redundant and observing the same area.
   - spanning a wide baseline for better multi-view constraints.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Prerequistes

1. What is keyframing or keyframe management?
   - A strategy of selecting and utilizing a crucial subset of frames.
2. Why do we need keyframing?
   - Infeasible to optimize jointly on all frames online.
3. How should we select keyframes?
   - non-redundant and observing the same area.
   - spanning a wide baseline for better multi-view constraints.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Prerequistes

1. What is keyframing or keyframe management?
   - A strategy of selecting and utilizing a crucial subset of frames.
2. Why do we need keyframing?
   - Infeasible to optimize jointly on all frames online.
3. How should we select keyframes?
   - non-redundant and observing the same area.
   - spanning a wide baseline for better multi-view constraints.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Prerequistes

1. What is keyframing or keyframe management?
   - A strategy of selecting and utilizing a crucial subset of frames.
2. Why do we need keyframing?
   - Infeasible to optimize jointly on all frames online.

     (a trade-off between efficiency and accuracy/robustness/...)
3. How should we select keyframes?
   - non-redundant and observing the same area.
   - spanning a wide baseline for better multi-view constraints.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Prerequistes

1. What is keyframing or keyframe management?
   - A strategy of selecting and utilizing a crucial subset of frames.
2. Why do we need keyframing?
   - Infeasible to optimize jointly on all frames online.

     (a trade-off between efficiency and accuracy/robustness/...)
3. How should we select keyframes?
   - non-redundant and observing the same area.
   - spanning a wide baseline for better multi-view constraints.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Prerequistes

1. What is keyframing or keyframe management?
   - A strategy of selecting and utilizing a crucial subset of frames.
2. Why do we need keyframing?
   - Infeasible to optimize jointly on all frames online.

     (a trade-off between efficiency and accuracy/robustness/...)
3. How should we select keyframes?
   - non-redundant and observing the same area.
   - spanning a wide baseline for better multi-view constraints.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Prerequistes

1. What is keyframing or keyframe management?
   - A strategy of selecting and utilizing a crucial subset of frames.
2. Why do we need keyframing?
   - Infeasible to optimize jointly on all frames online.
     (a trade-off between efficiency and accuracy/robustness/...)
3. How should we select keyframes?
   - non-redundant and observing the same area.
   - spanning a wide baseline for better multi-view constraints.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Registration i

If any of the following conditions is true...

Small Gaussian Covisibility — **Condition i, Keyframe Registration**

Gaussian covisibility between the current frame and the previous keyframe drops below a threshold.

$$\frac{\left|\mathrm{v}\left(\mathcal{G}, \mathcal{F}_i\right) \cap \mathrm{v}\left(\mathcal{G}, \mathcal{F}_j\right)\right|}{\left|\mathrm{v}\left(\mathcal{G}, \mathcal{F}_i\right) \cup \mathrm{v}\left(\mathcal{G}, \mathcal{F}_j\right)\right|} < \tau_1 \tag{8}$$

In practice, $\tau_1 = 0.95$.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Registration i

If any of the following conditions is true...

Small Gaussian Covisibility — **Condition i, Keyframe Registration**

Gaussian covisibility between the current frame and the previous keyframe drops below a threshold.

$$\frac{\left|\mathrm{v}\left(\mathcal{G}, \mathcal{F}_i\right) \cap \mathrm{v}\left(\mathcal{G}, \mathcal{F}_j\right)\right|}{\left|\mathrm{v}\left(\mathcal{G}, \mathcal{F}_i\right) \cup \mathrm{v}\left(\mathcal{G}, \mathcal{F}_j\right)\right|} < \tau_1 \tag{8}$$

In practice, $\tau_1 = 0.95$.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Registration i

If any of the following conditions is true...

Small Gaussian Covisibility | **Condition i, Keyframe Registration**

Gaussian covisibility between the current frame and the previous keyframe drops below a threshold.

$$\frac{\left|\mathrm{v}\left(\mathcal{G}, \mathcal{F}_i\right) \cap \underbrace{\mathrm{v}\left(\mathcal{G}, \mathcal{F}_j\right)}_{\subset \mathcal{G},\ \textbf{visible}\text{ Gaussians from frame } j}\right|}{\left|\mathrm{v}\left(\mathcal{G}, \mathcal{F}_i\right) \cup \mathrm{v}\left(\mathcal{G}, \mathcal{F}_j\right)\right|} < \tau_1 \tag{8}$$

In practice, $\tau_1 = 0.95$.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

If any of the following conditions is true...

Small Gaussian Covisibility **Condition i, Keyframe Registration**

Gaussian covisibility between the current frame and the previous keyframe drops below a threshold.

$$\frac{\left| \mathrm{v}\left(\mathcal{G}, \mathcal{F}_i\right) \cap \overbrace{\mathrm{v}\left(\mathcal{G}, \mathcal{F}_j\right)}^{\subset\, \mathcal{G},\ \textbf{visible}\ \text{Gaussians from frame}\ j} \right|}{\left| \mathrm{v}\left(\mathcal{G}, \underbrace{\mathcal{F}_i}_{\text{the previous keyframe}}\right) \cup \mathrm{v}\left(\mathcal{G}, \mathcal{F}_j\right) \right|} < \tau_1 \tag{8}$$

In practice, $\tau_1 = 0.95$.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

If any of the following conditions is true...

Small Gaussian Covisibility **Condition i, Keyframe Registration**

Gaussian covisibility between the current frame and the previous keyframe drops below a threshold.

$$\frac{\left| \mathrm{v}\left(\mathcal{G}, \mathcal{F}_i\right) \cap \overbrace{\mathrm{v}\left(\mathcal{G}, \mathcal{F}_j\right)}^{\subset\, \mathcal{G},\ \textbf{visible}\ \text{Gaussians from frame } j} \right|}{\left| \mathrm{v}\left(\mathcal{G}, \underbrace{\mathcal{F}_i}_{\text{the previous keyframe}}\right) \cup \mathrm{v}\left(\mathcal{G}, \underbrace{\mathcal{F}_j}_{\text{the current frame}}\right) \right|} < \tau_1 \tag{8}$$

In practice, $\tau_1 = 0.95$.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Large Relative Translation — **Condition ii, Keyframe Registration**

Translation from the previous keyframe w.r.t. to the median depth reaches a threshold.

$$\frac{\left\|\mathbf{t}_{\mathcal{F}_i\mathcal{F}_j}\right\|_2}{\bar{D}_{\mathcal{F}_i\mathcal{F}_j}} > \tau_2, \quad \bar{D}_{\mathcal{F}_i\mathcal{F}_j} = \frac{1}{2HW}\sum^{\{\mathcal{F}_i,\mathcal{F}_j\}}\sum_{h=0}^{H}\sum_{w=0}^{W} d(h, w) \tag{9}$$

In practice, $\tau_2 = 0.04$. Additionally, evaluate the Gaussian covisibility only if the relative translation is not too small (> 0.02) for efficiency.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Large Relative Translation **Condition ii, Keyframe Registration**

Translation from the previous keyframe w.r.t. to the median depth reaches a threshold.

$$\frac{\left\| \overbrace{\mathbf{t}_{\mathcal{F}_i\mathcal{F}_j}}^{\in \mathbb{R}^3,\ \text{translation from } \mathcal{F}_i \text{ to } \mathcal{F}_j} \right\|_2}{\bar{D}_{\mathcal{F}_i\mathcal{F}_j}} > \tau_2, \quad \bar{D}_{\mathcal{F}_i\mathcal{F}_j} = \frac{1}{2HW} \sum^{\{\mathcal{F}_i,\mathcal{F}_j\}} \sum_{h=0}^{H} \sum_{w=0}^{W} d(h, w) \tag{9}$$

In practice, $\tau_2 = 0.04$. Additionally, evaluate the Gaussian covisibility only if the relative translation is not too small (> 0.02) for efficiency.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Large Relative Translation **Condition ii, Keyframe Registration**

Translation from the previous keyframe w.r.t. to the median depth reaches a threshold.

$$\frac{\left\| \overbrace{\mathbf{t}_{\mathcal{F}_i \mathcal{F}_j}}^{\in \mathbb{R}^3,\ \text{translation from } \mathcal{F}_i \text{ to } \mathcal{F}_j} \right\|_2}{\underbrace{\bar{D}_{\mathcal{F}_i \mathcal{F}_j}}_{\in \mathbb{R},\ \text{the median depth}}} > \tau_2, \quad \bar{D}_{\mathcal{F}_i \mathcal{F}_j} = \frac{1}{2HW} \sum^{\{\mathcal{F}_i, \mathcal{F}_j\}} \sum_{h=0}^{H} \sum_{w=0}^{W} d(h, w) \tag{9}$$

In practice, $\tau_2 = 0.04$. Additionally, evaluate the Gaussian covisibility only if the relative translation is not too small (> 0.02) for efficiency.
(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Large Relative Translation — **Condition ii, Keyframe Registration**

Translation from the previous keyframe w.r.t. to the median depth reaches a threshold.

$\in \mathbb{R}^3$, translation from $\mathcal{F}_i$ to $\mathcal{F}_j$

$$\frac{\left\| \mathbf{t}_{\mathcal{F}_i\mathcal{F}_j} \right\|_2}{\bar{D}_{\mathcal{F}_i\mathcal{F}_j}} > \tau_2, \quad \bar{D}_{\mathcal{F}_i\mathcal{F}_j} = \frac{1}{2HW} \sum^{\{\mathcal{F}_i,\mathcal{F}_j\}} \sum_{h=0}^{H} \sum_{w=0}^{W} d(h,w) \tag{9}$$

$\in \mathbb{R}$, the median depth

depth of pixel (h, w)

In practice, $\tau_2 = 0.04$. Additionally, evaluate the Gaussian covisibility only if the relative translation is not too small (> 0.02) for efficiency.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Large Relative Translation — **Condition ii, Keyframe Registration**

Translation from the previous keyframe w.r.t. to the median depth reaches a threshold.

$$\frac{\left\| \overbrace{\mathbf{t}_{\mathcal{F}_i\mathcal{F}_j}}^{\in \mathbb{R}^3,\ \text{translation from } \mathcal{F}_i \text{ to } \mathcal{F}_j} \right\|_2}{\underbrace{\bar{D}_{\mathcal{F}_i\mathcal{F}_j}}_{\in \mathbb{R},\ \text{the median depth}}} > \tau_2, \quad \bar{D}_{\mathcal{F}_i\mathcal{F}_j} = \frac{1}{2\,\underbrace{H}_{\text{image height}}\,W} \sum^{\{\mathcal{F}_i,\mathcal{F}_j\}} \sum_{h=0}^{H} \sum_{w=0}^{W} \underbrace{d(h, w)}_{\text{depth of pixel } (h,w)} \tag{9}$$

In practice, $\tau_2 = 0.04$. Additionally, evaluate the Gaussian covisibility only if the relative translation is not too small (> 0.02) for efficiency.
(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Registration ii

Large Relative Translation **Condition ii, Keyframe Registration**

Translation from the previous keyframe w.r.t. to the median depth reaches a threshold.

$$\frac{\left\| \overbrace{\mathbf{t}_{\mathcal{F}_i \mathcal{F}_j}}^{\in \mathbb{R}^3,\ \text{translation from } \mathcal{F}_i \text{ to } \mathcal{F}_j} \right\|_2}{\underbrace{\bar{D}_{\mathcal{F}_i \mathcal{F}_j}}_{\in \mathbb{R},\ \text{the median depth}}} > \tau_2, \quad \bar{D}_{\mathcal{F}_i \mathcal{F}_j} = \frac{1}{2\, \underbrace{H}_{\text{image height}}\, \underbrace{W}_{\text{image width}}} \sum^{\{\mathcal{F}_i, \mathcal{F}_j\}} \sum_{h=0}^{H} \sum_{w=0}^{W} \underbrace{d(h, w)}_{\text{depth of pixel } (h,w)} \tag{9}$$

In practice, $\tau_2 = 0.04$. Additionally, evaluate the Gaussian covisibility only if the relative translation is not too small (> 0.02) for efficiency.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

If any of the following conditions is true...

Beyond Window Capacity **Condition i, Keyframe Removal**

Remove one of previous keyframes

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Removal i

If any of the following conditions is true...

Beyond Window Capacity — **Condition i, Keyframe Removal**

Remove one of previous keyframes

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Removal i

If any of the following conditions is true...

Beyond Window Capacity — **Condition i, Keyframe Removal**

Remove one of previous keyframes

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Removal i

If any of the following conditions is true...

Beyond Window Capacity — **Condition i, Keyframe Removal**

Remove one of previous keyframes that minimize the impact on the overall baseline length.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

If any of the following conditions is true...

Beyond Window Capacity **Condition i, Keyframe Removal**

Remove one of previous keyframes that minimize the impact on the overall baseline length.

$$\mathcal{F}^* = \arg\max_{\mathcal{F}\in\mathcal{W}} \mathrm{l}\left(\mathcal{W} \setminus \{\mathcal{F}\}\right) \tag{10}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

If any of the following conditions is true...

Beyond Window Capacity **Condition i, Keyframe Removal**

Remove one of previous keyframes that minimize the impact on the overall baseline length.

$$\mathcal{F}^* = \operatorname*{arg\,max}_{\mathcal{F}\in\mathcal{W}} \mathrm{l}\left(\mathcal{W} \setminus \{\mathcal{F}\}\right), \quad \mathrm{l}\left(\mathcal{W}\right) = \sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{i} \left\| \mathbf{t}_{\mathcal{F}_i\mathcal{F}_j} \right\| \tag{10}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Removal i

If any of the following conditions is true...

Beyond Window Capacity **Condition i, Keyframe Removal**

Remove one of previous keyframes that minimize the impact on the overall baseline length.

$$\mathcal{F}^* = \underset{\mathcal{F} \in \mathcal{W}}{\arg\max}\ \mathrm{l}\left(\mathcal{W} \setminus \{\mathcal{F}\}\right), \quad \mathrm{l}\left(\mathcal{W}\right) = \sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{i} \left\| \mathbf{t}_{\mathcal{F}_i \mathcal{F}_j} \right\| \tag{10}$$

Remark: for the best multi-view constraints.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Removal ii

Low Gaussian Overlap Coefficient — **Condition ii, Keyframe Removal**

Remove multiple previous keyframes if the "Gaussian overlap coefficient" drops below a threshold.

Szymkiewicz–Simpson coefficient

In practice, $\tau_4 = 0.4$.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Low Gaussian Overlap Coefficient **Condition ii, Keyframe Removal**

Remove multiple previous keyframes if the “Gaussian overlap coefficient” drops below a threshold.

Szymkiewicz–Simpson coefficient

In practice, $\tau_4 = 0.4$.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Removal ii

Low Gaussian Overlap Coefficient — **Condition ii, Keyframe Removal**

Remove multiple previous keyframes if the "Gaussian overlap coefficient" drops below a threshold.

Szymkiewicz–Simpson coefficient

In practice, $\tau_4 = 0.4$.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Removal ii

Low Gaussian Overlap Coefficient **Condition ii, Keyframe Removal**

Remove multiple previous keyframes if the "Gaussian overlap coefficient" drops below a threshold.

$$\frac{|\,\mathrm{v}\left(\mathcal{G}, \mathcal{F}_i\right) \cap \mathrm{v}\left(\mathcal{G}, \mathcal{F}_j\right)|}{\min\left(|\,\mathrm{v}\left(\mathcal{G}, \mathcal{F}_i\right)|, |\,\mathrm{v}\left(\mathcal{G}, \mathcal{F}_j\right)|\right)} < \tau_4 \tag{11}$$

Szymkiewicz–Simpson coefficient

In practice, $\tau_4 = 0.4$.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Keyframe Management: Removal ii

Low Gaussian Overlap Coefficient **Condition ii, Keyframe Removal**

Remove multiple previous keyframes if the "Gaussian overlap coefficient" drops below a threshold.

$$\frac{|\,\mathrm{v}\,(\mathcal{G}, \mathcal{F}_i) \cap \mathrm{v}\,(\mathcal{G}, \mathcal{F}_j)\,|}{\min\left(|\,\mathrm{v}\,(\mathcal{G}, \mathcal{F}_i)\,|, |\,\mathrm{v}\,(\mathcal{G}, \mathcal{F}_j)\,|\right)} < \tau_4 \tag{11}$$

Remark: not observing the same area.

Szymkiewicz–Simpson coefficient

In practice, $\tau_4 = 0.4$.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- Why do we need "Gaussian insertion"?
  - SLAM is for robotic exploration.
- When do we need "Gaussian insertion"?

Keyframing — **Condition i, Gaussian Insertion**

Insertion is triggered for every new keyframe.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Gaussian Management: Insertion

- Why do we need "Gaussian insertion"?
  - SLAM is for robotic exploration.

- When do we need "Gaussian insertion"?

Keyframing — **Condition i, Gaussian Insertion**

Insertion is triggered for every new keyframe.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- Why do we need “Gaussian insertion”?
  - SLAM is for robotic exploration.

- When do we need “Gaussian insertion”?

Keyframing **Condition i, Gaussian Insertion**

Insertion is triggered for every new keyframe.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- Why do we need "Gaussian insertion"?
  - SLAM is for robotic exploration.
- When do we need "Gaussian insertion"?

Keyframing — **Condition i, Gaussian Insertion**

Insertion is triggered for every new keyframe.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- How do we insert Gaussians?
  - Gaussian insertion is Gaussian initialization.

If Depth Available **Gaussian Initialization**

Back-project in a per-pixel, per-Gaussian approach.

If Depth Unavailable **Gaussian Initialization**

Leverage the rendered depth map.

- for pixels with depth: use the rendered depth and assign a "low" covariance.
- for pixels w/o depth: use the median of rendered depth and assign a "high" covariance.

In practice, "low": 0.2σ; "high": 0.5σ, where σ is the standard deviation of the rendered depth map.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- How do we insert Gaussians?
  - Gaussian insertion is Gaussian initialization.

If Depth Available — **Gaussian Initialization**

Back-project in a per-pixel, per-Gaussian approach.

If Depth Unavailable — **Gaussian Initialization**

Leverage the rendered depth map.

- for pixels with depth: use the rendered depth and assign a "low" covariance.
- for pixels w/o depth: use the median of rendered depth and assign a "high" covariance.

In practice, "low": 0.2σ; "high": 0.5σ, where σ is the standard deviation of the rendered depth map.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Gaussian Management: Initialization

- How do we insert Gaussians?
  - Gaussian insertion is Gaussian initialization.

If Depth Available — **Gaussian Initialization**

Back-project in a per-pixel, per-Gaussian approach.

If Depth Unavailable — **Gaussian Initialization**

Leverage the rendered depth map.

- for pixels with depth: use the rendered depth and assign a "low" covariance.
- for pixels w/o depth: use the median of rendered depth and assign a "high" covariance.

In practice, "low": 0.2σ; "high": 0.5σ, where σ is the standard deviation of the rendered depth map.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Gaussian Management: Initialization

- How do we insert Gaussians?
  - Gaussian insertion is Gaussian initialization.

If Depth Available — **Gaussian Initialization**

Back-project in a per-pixel, per-Gaussian approach.

If Depth Unavailable — **Gaussian Initialization**

Leverage the rendered depth map.

- for pixels with depth: use the rendered depth and assign a "low" covariance.
- for pixels w/o depth: use the median of rendered depth and assign a "high" covariance.

In practice, "low": 0.2σ; "high": 0.5σ, where σ is the standard deviation of the rendered depth map.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Gaussian Management: Initialization

- How do we insert Gaussians?
  - Gaussian insertion is Gaussian initialization.

If Depth Available — **Gaussian Initialization**

Back-project in a per-pixel, per-Gaussian approach.

If Depth Unavailable — **Gaussian Initialization**

Leverage the rendered depth map.

- for pixels with depth: use the rendered depth and assign a “low” covariance.
- for pixels w/o depth: use the median of rendered depth and assign a “high” covariance.

In practice, “low”: 0.2σ; “high”: 0.5σ, where σ is the standard deviation of the rendered depth map.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Gaussian Management: Initialization

- How do we insert Gaussians?
    - Gaussian insertion is Gaussian initialization.

If Depth Available — **Gaussian Initialization**

Back-project in a per-pixel, per-Gaussian approach.

If Depth Unavailable — **Gaussian Initialization**

Leverage the rendered depth map.

- for pixels with depth: use the rendered depth and assign a "low" covariance.
- for pixels w/o depth: use the median of rendered depth and assign a "high" covariance.

In practice, "low": 0.2σ; "high": 0.5σ, where σ is the standard deviation of the rendered depth map.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Gaussian Management: Pruning

- Why do we need "Gaussian Pruning"?
  - if depth unavailable, too many incorrect newly inserted Gaussians.

Low Gaussian Opacity — **Condition i, Gaussian Pruning**

Low opacity Gaussians are pruned.

$$\{\mathcal{G}_i \in \mathcal{G} \mid \alpha(\mathcal{G}_i) < \tau_\alpha\} \tag{12}$$

Low Gaussian Covisibility — **Condition ii, Gaussian Pruning**

For "just" inserted Gaussians but unobserved by "some other" keyframes, are pruned out.

If no pruning, although the majority of incorrect Gaussians vanish quickly in following optimization, there are some survivals.

In practice, $\tau_\alpha = 0.7$.

In practice, the pruned Gaussians are inserted in the last 3 keyframes and unobserved by any other 3 keyframes in the sliding window.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- Why do we need "Gaussian Pruning"?
  - if depth unavailable, too many incorrect newly inserted Gaussians.

Low Gaussian Opacity — **Condition i, Gaussian Pruning**

Low opacity Gaussians are pruned.

$$\{\mathcal{G}_i \in \mathcal{G} \mid \alpha(\mathcal{G}_i) < \tau_\alpha\} \tag{12}$$

Low Gaussian Covisibility — **Condition ii, Gaussian Pruning**

For "just" inserted Gaussians but unobserved by "some other" keyframes, are pruned out.

If no pruning, although the majority of incorrect Gaussians vanish quickly in following optimization, there are some survivals.

In practice, $\tau_\alpha = 0.7$.

In practice, the pruned Gaussians are inserted in the last 3 keyframes and unobserved by any other 3 keyframes in the sliding window.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Gaussian Management: Pruning

- Why do we need "Gaussian Pruning"?
  - if depth unavailable, too many incorrect newly inserted Gaussians.

Low Gaussian Opacity — **Condition i, Gaussian Pruning**

Low opacity Gaussians are pruned.

$$\{\mathcal{G}_i \in \mathcal{G} \mid \alpha(\mathcal{G}_i) < \tau_\alpha\} \quad (12)$$

Low Gaussian Covisibility — **Condition ii, Gaussian Pruning**

For "just" inserted Gaussians but unobserved by "some other" keyframes, are pruned out.

If no pruning, although the majority of incorrect Gaussians vanish quickly in following optimization, there are some survivals.

In practice, $\tau_\alpha = 0.7$.

In practice, the pruned Gaussians are inserted in the last 3 keyframes and unobserved by any other 3 keyframes in the sliding window.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Gaussian Management: Pruning

- Why do we need "Gaussian Pruning"?
  - if depth unavailable, too many incorrect newly inserted Gaussians.

Low Gaussian Opacity — **Condition i, Gaussian Pruning**

Low opacity Gaussians are pruned.

$$\{\mathcal{G}_i \in \mathcal{G} \mid \alpha(\mathcal{G}_i) < \tau_\alpha\} \tag{12}$$

Low Gaussian Covisibility — **Condition ii, Gaussian Pruning**

For "just" inserted Gaussians but unobserved by "some other" keyframes, are pruned out.

If no pruning, although the majority of incorrect Gaussians vanish quickly in following optimization, there are some survivals.

In practice, $\tau_\alpha = 0.7$.

In practice, the pruned Gaussians are inserted in the last 3 keyframes and unobserved by any other 3 keyframes in the sliding window.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Mapping: Overview

- Mapping
  - Bundle Adjustment
    - Photometric RGB & Depth Loss
    - Isotropic Regularization
    - Random Recall

- Why do we need mapping in 3DGS SLAM?
  - Local: Optimize newly inserted 3D Gaussians
  - Global: Reconstruct a globally 3D-coherent structure

key method trick convention

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Mapping: Overview

- Mapping
  - Bundle Adjustment
    - Photometric RGB & Depth Loss
    - Isotropic Regularization
    - Random Recall

- Why do we need mapping in **3DGS** SLAM?
  - Local: Optimize newly inserted 3D Gaussians.
  - Global: Reconstruct a globally 3D-coherent structure.

key method trick convention

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Mapping: Overview

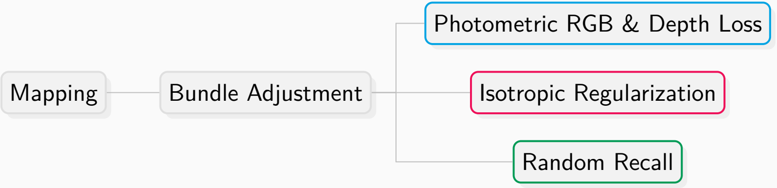
- Mapping
  - Bundle Adjustment
    - Photometric RGB & Depth Loss
    - Isotropic Regularization
    - Random Recall

- Why do we need mapping in **3DGS** SLAM?
  - Local: Optimize newly inserted 3D Gaussians.
  - Global: Reconstruct a globally 3D-coherent structure.

key method trick convention

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Mapping: Overview



- Why do we need mapping in **3DGS** SLAM?
  - Local: Optimize newly inserted 3D Gaussians.
  - Global: Reconstruct a globally 3D-coherent structure.

key method trick convention

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Mapping: Bundle Adjustment & Random Recall

Bundle Adjustment

$$\underset{\mathcal{G},\{\mathbf{T}_{cw}(\mathcal{F}_k)|\mathcal{F}_k\in\mathcal{W}\}}{\operatorname{argmin}} \sum_{\mathcal{F}_k}^{\mathcal{W}} \mathcal{L}_{pho}\left(\mathcal{F}_k\right) \tag{13}$$

Random Recall

Besides $\mathcal{W}$, "some" randomly selected historical keyframes are also leveraged in BA to avoid forgetting the global map.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Bundle Adjustment

$$\underset{\boxed{\mathcal{G}},\{\mathbf{T}_{cw}(\mathcal{F}_k)|\mathcal{F}_k\in\mathcal{W}\}}{\operatorname{argmin}} \sum_{\mathcal{F}_k}^{\mathcal{W}} \mathcal{L}_{pho}\left(\mathcal{F}_k\right) \tag{13}$$

3D Gaussians

Random Recall

Besides $\mathcal{W}$, "some" randomly selected historical keyframes are also leveraged in BA to avoid forgetting the global map.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Bundle Adjustment

$$\underset{\underbrace{\mathcal{G}}_{\text{3D Gaussians}},\ \underbrace{\{\mathbf{T}_{cw}(\mathcal{F}_k) | \mathcal{F}_k \in \mathcal{W}\}}_{\text{camera poses of keyframes in the sliding window}}}{\operatorname{argmin}} \sum_{\mathcal{F}_k}^{\mathcal{W}} \mathcal{L}_{pho}\left(\mathcal{F}_k\right) \tag{13}$$

Random Recall

Besides $\mathcal{W}$, "some" randomly selected historical keyframes are also leveraged in BA to avoid forgetting the global map.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Mapping: Bundle Adjustment & Random Recall

Bundle Adjustment

$$\underset{\underbrace{\mathcal{G}}_{\text{3D Gaussians}},\ \underbrace{\{\mathbf{T}_{cw}(\mathcal{F}_k)|\mathcal{F}_k \in \mathcal{W}\}}_{\text{camera poses of keyframes in the sliding window}}}{\operatorname{argmin}} \sum_{\mathcal{F}_k}^{\mathcal{W}} \mathcal{L}_{pho}\left(\mathcal{F}_k\right) \tag{13}$$

Random Recall

Besides $\mathcal{W}$, "some" randomly selected historical keyframes are also leveraged in BA to avoid forgetting the global map.

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- Why do we need "isotropic regularization"?
  - Observation: isotropic Gaussians behave better than anisotrophic.
  - Analysis: no constraints on the elongation along the viewing ray direction, **even with depth**.

Isotropic Regularization

$$\mathcal{L}_{iso} = \sum_{i=1}^{|\mathcal{G}|} \|\mathrm{s}(\mathcal{G}_i) - \bar{\mathrm{s}}(\mathcal{G}_i)\|_1 , \tag{14}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- Why do we need “isotropic regularization”?
  - Observation: isotropic Gaussians behave better than anisotrophic.
  - Analysis: no constraints on the elongation along the viewing ray direction, **even with depth**.

Isotropic Regularization

$$\mathcal{L}_{iso} = \sum_{i=1}^{|\mathcal{G}|} \|\mathrm{s}(\mathcal{G}_i) - \bar{\mathrm{s}}(\mathcal{G}_i)\|_1, \quad (14)$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- Why do we need "isotropic regularization"?
  - Observation: isotropic Gaussians behave better than anisotrophic.
  - Analysis: no constraints on the elongation along the viewing ray direction, **even with depth**.

Isotropic Regularization

$$\mathcal{L}_{iso} = \sum_{i=1}^{|\mathcal{G}|} \|\mathrm{s}(\mathcal{G}_i) - \bar{\mathrm{s}}(\mathcal{G}_i)\|_1, \tag{14}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- Why do we need "isotropic regularization"?
  - Observation: isotropic Gaussians behave better than anisotrophic.
  - Analysis: no constraints on the elongation along the viewing ray direction, **even with depth**.

Isotropic Regularization

$$\mathcal{L}_{iso} = \sum_{i=1}^{|\mathcal{G}|} \| \mathrm{s}(\mathcal{G}_i) - \bar{\mathrm{s}}(\mathcal{G}_i) \|_1 , \tag{14}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- Why do we need "isotropic regularization"?
  - Observation: isotropic Gaussians behave better than anisotrophic.
  - Analysis: no constraints on the elongation along the viewing ray direction, **even with depth**.

Isotropic Regularization

$$\mathcal{L}_{iso} = \sum_{i=1}^{\overbrace{|\mathcal{G}|}^{\in \mathbb{N},\ \text{total number of Gaussians}}} \|\mathbf{s}(\mathcal{G}_i) - \bar{\mathbf{s}}(\mathcal{G}_i)\|_1, \tag{14}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

- Why do we need "isotropic regularization"?
  - Observation: isotropic Gaussians behave better than anisotrophic.
  - Analysis: no constraints on the elongation along the viewing ray direction, **even with depth**.

Isotropic Regularization

$$\mathcal{L}_{iso} = \sum_{i=1}^{\overbrace{|\mathcal{G}|}^{\in \mathbb{N},\ \text{total number of Gaussians}}} \| \overbrace{\mathbf{s}(\mathcal{G}_i)}^{\in \mathbb{R}^3,\ \text{scale of } i\text{-th Gaussian}} - \bar{\mathbf{s}}(\mathcal{G}_i)\|_1, \tag{14}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Mapping: Isotropic Regularization

- Why do we need "isotropic regularization"?
  - Observation: isotropic Gaussians behave better than anisotrophic.
  - Analysis: no constraints on the elongation along the viewing ray direction, **even with depth**.

Isotropic Regularization

$$\mathcal{L}_{iso} = \sum_{i=1}^{\overbrace{|\mathcal{G}|}^{\in \mathbb{N}\text{, total number of Gaussians}}} \| \overbrace{\mathbf{s}(\mathcal{G}_i)}^{\in \mathbb{R}^3\text{, scale of } i\text{-th Gaussian}} - \overbrace{\bar{\mathbf{s}}(\mathcal{G}_i)}^{\in \mathbb{R}^3\text{, averaged scale of } i\text{-th Gaussian}} \|_1, \quad \bar{\mathbf{s}}(\mathcal{G}_i) = \begin{bmatrix} \left(s(\mathcal{G}_i)^x + s(\mathcal{G}_i)^y + s(\mathcal{G}_i)^z\right)/3 \\ \left(s(\mathcal{G}_i)^x + s(\mathcal{G}_i)^y + s(\mathcal{G}_i)^z\right)/3 \\ \left(s(\mathcal{G}_i)^x + s(\mathcal{G}_i)^y + s(\mathcal{G}_i)^z\right)/3 \end{bmatrix} \tag{14}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Mapping: Conclusion

Overall Optimization for Mapping

$$\underset{\mathcal{G},\{\mathbf{T}_{cw}(\mathcal{F}_k)|\mathcal{F}_k\in\mathcal{W}^+\}}{\operatorname{argmin}} \sum_{\mathcal{F}_k}^{\mathcal{W}^+} \mathcal{L}_{pho}\left(\mathcal{F}_k\right) + \lambda_{iso}\mathcal{L}_{iso} \tag{15}$$

(CVPR Highlight, 2024) MonoGS: Gaussian Splatting SLAM

Semantic 3DGS

Timeline

11/2023 - 12/2024	01/2024 - 04/2024	05/2024 - 06/2024
LEGS [6]	CoSSegGaussians [11]	GOI [15]
LangSplat [7]	Semantic Gaussians [12]	RT-GS2 [16]
Feature 3DGS [8]	Feature Splating [13]	SA-GS [17]
Gaussian Grouping [9]	CLIP-GS [14]	Fast-LGS [18]
Segment Any Gaussian [10]		

Consensus

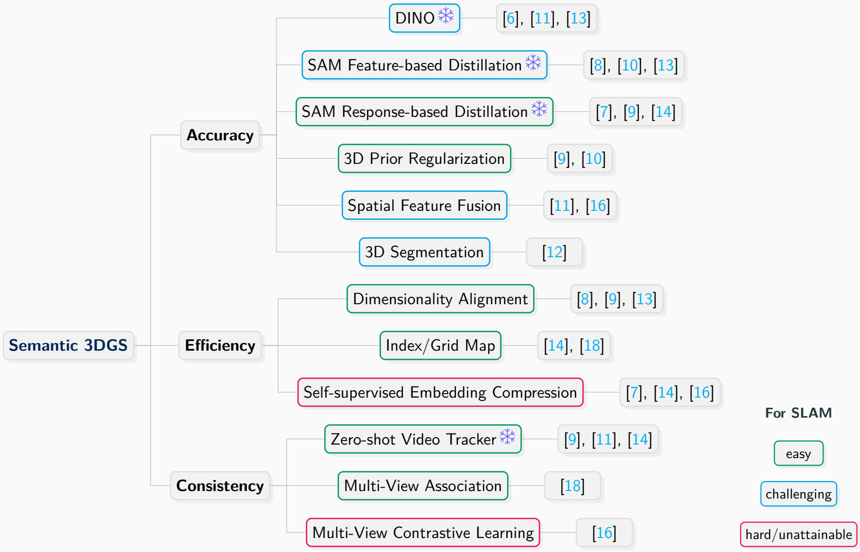
Lift 2D foundation models to 3D scene-specific Gaussians under 2D supervision.

1. 2D foundation models: CLIP, SAM, DINO, etc.
2. Interactivity: manipulation, edit, localization, query, simulation, etc.

Taxonomy

- **Semantic 3DGS**
  - **Accuracy**
    - DINO ❄ — [6], [11], [13]
    - SAM ❄ — [7]–[10], [13], [14]
    - 3D Prior Regularization — [9], [10]
    - Spatial Feature Fusion — [11], [16]
    - 3D Segmentation — [12]
  - **Efficiency**
    - Dimensionality Alignment — [8], [9], [13]
    - Embedding Compression — [7], [14], [16]
    - Index/Grid Map — [14], [18]
  - **Consistency**
    - Zero-shot Video Tracker ❄ — [9], [11], [14]
    - Multi-View Association — [18]
    - Multi-View Self-supervision — [16]

Overview


Semantic 3DGS
Accuracy
DINO
[6], [11], [13]
SAM Feature-based Distillation
[8], [10], [13]
SAM Response-based Distillation
[7], [9], [14]
3D Prior Regularization
[9], [10]
Spatial Feature Fusion
[11], [16]
3D Segmentation
[12]
Efficiency
Dimensionality Alignment
[8], [9], [13]
Index/Grid Map
[14], [18]
Self-supervised Embedding Compression
[7], [14], [16]
Consistency
Zero-shot Video Tracker
[9], [11], [14]
Multi-View Association
[18]
Multi-View Contrastive Learning
[16]
For SLAM
easy
challenging
hard/unattainable

Semantic 3DGS

Feature-3DGS

Key Insights

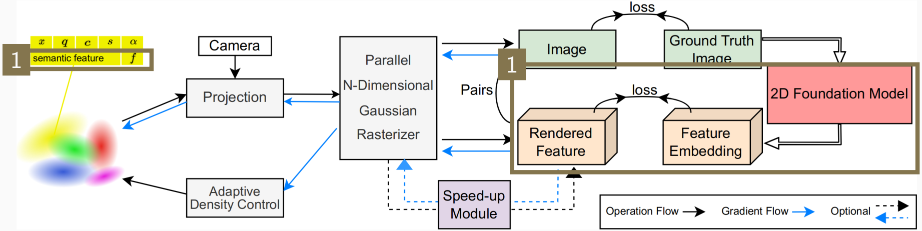


Figure 1: Overview of Feature 3DGS

1 Semantic Rendering Pipeline

Differentiable rendering of Gaussian-wise latent semantic features.

2 Speed-up module

Dimensionality Alignment.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Key Insights

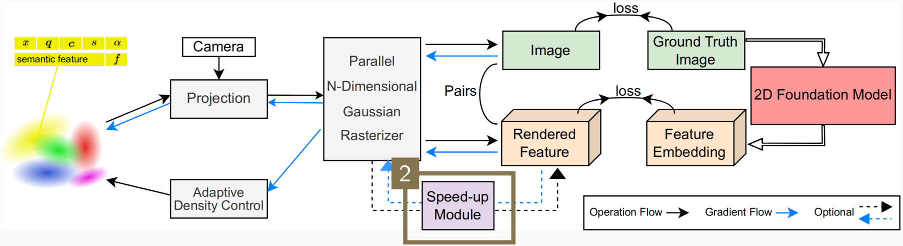


Figure 1: Overview of Feature 3DGS

1. Semantic Rendering Pipeline

   Differentiable rendering of Gaussian-wise latent semantic features.

2. Speed-up module

   Dimensionality Alignment.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

To render semantic embeddings, i.e.

$$\mathbb{R}^{N\times D} \mapsto \mathbb{R}^{H\times W\times D} \tag{16}$$

1. representation
2. projection
3. blending
4. rasterization
5. inverse rendering

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

To render semantic embeddings, i.e.

$$\mathbb{R}^{\underbrace{N}_{\text{number of 3D Gaussians}} \times D} \mapsto \mathbb{R}^{H \times W \times D} \tag{16}$$

1. representation
2. projection
3. blending
4. rasterization
5. inverse rendering

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline

To render semantic embeddings, i.e.

$$\mathbb{R}^{\underbrace{N}_{\text{number of 3D Gaussians}} \times \underbrace{D}_{\text{dimension of semantic embedding}}} \mapsto \mathbb{R}^{H \times W \times D} \tag{16}$$

1. representation
2. projection
3. blending
4. rasterization
5. inverse rendering

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline

To render semantic embeddings, i.e.

$$\mathbb{R}^{\underbrace{N}_{\text{number of 3D Gaussians}} \times \underbrace{D}_{\text{dimension of semantic embedding}}} \mapsto \mathbb{R}^{\underbrace{H}_{\text{image height}} \times W \times D} \tag{16}$$

1. representation
2. projection
3. blending
4. rasterization
5. inverse rendering

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline

To render semantic embeddings, i.e.

$$\mathbb{R}^{\underbrace{N}_{\text{number of 3D Gaussians}} \times \underbrace{D}_{\text{dimension of semantic embedding}}} \mapsto \mathbb{R}^{\underbrace{H}_{\text{image height}} \times \underbrace{W}_{\text{image width}} \times D} \tag{16}$$

1. representation
2. projection
3. blending
4. rasterization
5. inverse rendering

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

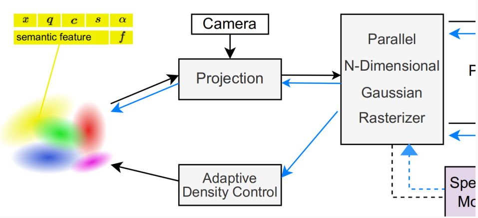
Semantic Rendering Pipeline

To render semantic embeddings, i.e.

$$\mathbb{R}^{\underbrace{N}_{\text{number of 3D Gaussians}} \times \underbrace{D}_{\text{dimension of semantic embedding}}} \mapsto \mathbb{R}^{\underbrace{H}_{\text{image height}} \times \underbrace{W}_{\text{image width}} \times D} \tag{16}$$

5 things,

1. representation
2. projection
3. blending
4. rasterization
5. inverse rendering



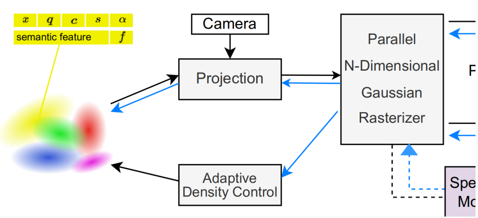
(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

To render semantic embeddings, i.e.

$$\mathbb{R}^{\underbrace{N}_{\text{number of 3D Gaussians}} \times \underbrace{D}_{\text{dimension of semantic embedding}}} \mapsto \mathbb{R}^{\underbrace{H}_{\text{image height}} \times \underbrace{W}_{\text{image width}} \times D} \tag{16}$$

5 things,

1. representation
2. projection
3. blending
4. rasterization
5. inverse rendering



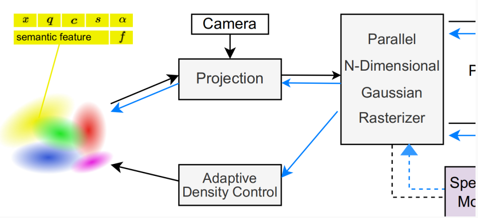
(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

To render semantic embeddings, i.e.

$$\mathbb{R}^{\underbrace{N}_{\text{number of 3D Gaussians}} \times \underbrace{D}_{\text{dimension of semantic embedding}}} \mapsto \mathbb{R}^{\underbrace{H}_{\text{image height}} \times \underbrace{W}_{\text{image width}} \times D} \tag{16}$$

5 things,

1. representation
2. projection
3. blending
4. rasterization
5. inverse rendering



(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

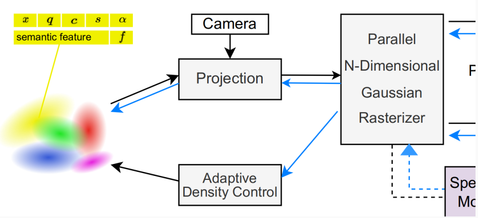
Semantic Rendering Pipeline

To render semantic embeddings, i.e.

$$\mathbb{R}^{\underbrace{N}_{\text{number of 3D Gaussians}} \times \underbrace{D}_{\text{dimension of semantic embedding}}} \mapsto \mathbb{R}^{\underbrace{H}_{\text{image height}} \times \underbrace{W}_{\text{image width}} \times D} \tag{16}$$

5 things,

1. representation
2. projection
3. blending
4. rasterization
5. inverse rendering



(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

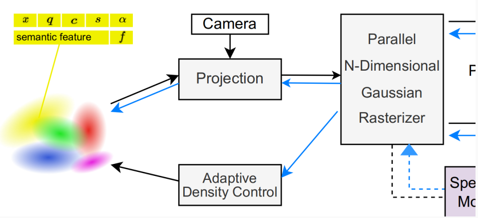
Semantic Rendering Pipeline

To render semantic embeddings, i.e.

$$\mathbb{R}^{\underbrace{N}_{\text{number of 3D Gaussians}} \times \underbrace{D}_{\text{dimension of semantic embedding}}} \mapsto \mathbb{R}^{\underbrace{H}_{\text{image height}} \times \underbrace{W}_{\text{image width}} \times D} \tag{16}$$

5 things,

1. representation
2. projection
3. blending
4. rasterization
5. inverse rendering



(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

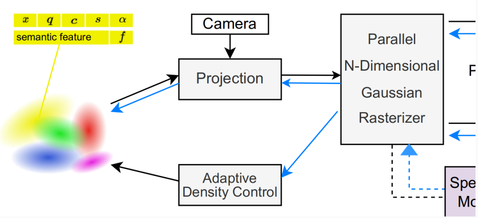
Semantic Rendering Pipeline

To render semantic embeddings, i.e.

$$\mathbb{R}^{\underbrace{N}_{\text{number of 3D Gaussians}} \times \underbrace{D}_{\text{dimension of semantic embedding}}} \mapsto \mathbb{R}^{\underbrace{H}_{\text{image height}} \times \underbrace{W}_{\text{image width}} \times D} \tag{16}$$

5 things,

1. representation
2. projection
3. blending
4. rasterization
5. inverse rendering



(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Representation: 3D Gaussian augmented with a latent embedding.

$$\mathcal{G}_i = \{\mathbf{x}, \mathbf{q}, \mathbf{s}, \alpha, \mathbf{c}, \mathbf{f}\} \tag{17}$$

n: the maximal order of spherical harmonics to represent a color channel. In practice, $n = 4$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline i

Representation: 3D Gaussian augmented with a latent embedding.

$$\underbrace{\mathcal{G}}_{\text{optimizable attributes of a 3D Gaussian}}{}_i = \{\mathbf{x}, \mathbf{q}, \mathbf{s}, \alpha, \mathbf{c}, \mathbf{f}\} \tag{17}$$

n: the maximal order of spherical harmonics to represent a color channel. In practice, $n = 4$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline i

Representation: 3D Gaussian augmented with a latent embedding.

$$\underbrace{\mathcal{G}}_{\text{optimizable attributes of a 3D Gaussian}}{}_{\underbrace{i}_{\in \mathbb{N},\ \text{index of 3D Gaussian}}} = \{\mathbf{x}, \mathbf{q}, \mathbf{s}, \alpha, \mathbf{c}, \mathbf{f}\} \tag{17}$$

n: the maximal order of spherical harmonics to represent a color channel. In practice, $n = 4$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline i

Representation: 3D Gaussian augmented with a latent embedding.

$$\underbrace{\mathcal{G}}_{\text{optimizable attributes of a 3D Gaussian}}\underbrace{{}_{i}}_{\in \mathbb{N},\ \text{index of 3D Gaussian}} = \{\underbrace{\mathbf{x}}_{\in \mathbb{R}^3,\ \text{position}}, \mathbf{q}, \mathbf{s}, \alpha, \mathbf{c}, \mathbf{f}\} \tag{17}$$

---

n: the maximal order of spherical harmonics to represent a color channel. In practice, $n = 4$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline i

Representation: 3D Gaussian augmented with a latent embedding.

$$\underbrace{\mathcal{G}}_{\text{optimizable attributes of a 3D Gaussian}}{}_{\underbrace{i}_{\in\mathbb{N},\ \text{index of 3D Gaussian}}} = \left\{ \underbrace{\mathbf{x}}_{\in\mathbb{R}^3,\ \text{position}}, \underbrace{\mathbf{q}}_{\in\mathrm{SO}(3),\ \text{rotation}}, \mathbf{s}, \alpha, \mathbf{c}, \mathbf{f} \right\} \tag{17}$$

n: the maximal order of spherical harmonics to represent a color channel. In practice, $n = 4$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Representation: 3D Gaussian augmented with a latent embedding.

$$\boxed{\mathcal{G}}_{\boxed{i}} = \left\{ \boxed{\mathbf{x}}, \boxed{\mathbf{q}}, \boxed{\mathbf{s}}, \alpha, \mathbf{c}, \mathbf{f} \right\} \tag{17}$$

optimizable attributes of a 3D Gaussian

$\in \mathbb{N}$, index of 3D Gaussian

$\in \mathbb{R}^3$, scale

$\in \mathrm{SO}(3)$, rotation

$\in \mathbb{R}^3$, position

n: the maximal order of spherical harmonics to represent a color channel. In practice, $n = 4$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Representation: 3D Gaussian augmented with a latent embedding.

$$\underbrace{\mathcal{G}}_{\text{optimizable attributes of a 3D Gaussian}}{}_{\underbrace{i}_{\in \mathbb{N},\ \text{index of 3D Gaussian}}} = \Big\{ \underbrace{\mathbf{x}}_{\in \mathbb{R}^3,\ \text{position}}, \underbrace{\mathbf{q}}_{\in \mathrm{SO}(3),\ \text{rotation}}, \underbrace{\mathbf{s}}_{\in \mathbb{R}^3,\ \text{scale}}, \underbrace{\alpha}_{\in [0,1],\ \text{opacity}}, \mathbf{c}, \mathbf{f} \Big\} \tag{17}$$

---

n: the maximal order of spherical harmonics to represent a color channel. In practice, $n = 4$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline i

Representation: 3D Gaussian augmented with a latent embedding.

$$\mathcal{G}_i = \left\{ \mathbf{x}, \mathbf{q}, \mathbf{s}, \alpha, \mathbf{c}, \mathbf{f} \right\} \tag{17}$$

optimizable attributes of a 3D Gaussian

$\in \mathbb{N}$, index of 3D Gaussian

$\in \mathbb{R}^{3n}$, color

$\in [0, 1]$, opacity

$\in \mathbb{R}^3$, scale

$\in \mathrm{SO}(3)$, rotation

$\in \mathbb{R}^3$, position

n: the maximal order of spherical harmonics to represent a color channel. In practice, $n = 4$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Representation: 3D Gaussian augmented with a latent embedding.

$$\underbrace{\overbrace{\mathcal{G}}^{\text{optimizable attributes of a 3D Gaussian}}}_{}\ {}_{\underset{\in \mathbb{N},\ \text{index of 3D Gaussian}}{i}} = \{\underbrace{\mathbf{x}}_{\in \mathbb{R}^3,\ \text{position}}, \underbrace{\mathbf{q}}_{\in \mathrm{SO}(3),\ \text{rotation}}, \underbrace{\mathbf{s}}_{\in \mathbb{R}^3,\ \text{scale}}, \underbrace{\alpha}_{\in [0,1],\ \text{opacity}}, \underbrace{\mathbf{c}}_{\in \mathbb{R}^{3n},\ \text{color}}, \underbrace{\mathbf{f}}_{\in \mathbb{R}^3,\ \text{semantic feature}}, \} \tag{17}$$

n: the maximal order of spherical harmonics to represent a color channel. In practice, $n = 4$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline ii

Projection: from 3D ellipsoids to 2D ellipses.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Projection: from 3D ellipsoids to 2D ellipses.

$$\mu_i = \pi\left(\mathbf{T}_{cw} \cdot \mu_w\right) \tag{18}$$

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \tag{19}$$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Projection: from 3D ellipsoids to 2D ellipses.

$$\mu_i = \pi\left(\mathbf{T}_{cw} \cdot \underbrace{\mu_w}_{\in \mathbb{P}^3,\ \text{3D(world) mean}}\right) \tag{18}$$

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \tag{19}$$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline ii

Projection: from 3D ellipsoids to 2D ellipses.

$$\mu_i = \pi\left(\underbrace{\mathbf{T}_{cw}}_{\in \mathrm{SE}(3),\ \text{camera pose}} \cdot \underbrace{\mu_w}_{\in \mathbb{P}^3,\ \text{3D(world) mean}}\right) \quad (18)$$

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \quad (19)$$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Projection: from 3D ellipsoids to 2D ellipses.

$$\mu_i = \underbrace{\pi}_{\text{projection}} \left(\underbrace{\mathbf{T}_{cw}}_{\in \mathrm{SE}(3),\ \text{camera pose}} \cdot \underbrace{\mu_w}_{\in \mathbb{P}^3,\ \text{3D(world) mean}} \right) \quad (18)$$

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \quad (19)$$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Projection: from 3D ellipsoids to 2D ellipses.

$$\mu_i = \pi \left(\mathbf{T}_{cw} \cdot \mu_w \right) \quad (18)$$

$\in \mathbb{P}^3$, 3D(world) mean

$\in \mathrm{SE}(3)$, camera pose

projection

$\in \mathbb{P}^2$, 2D(image) mean

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \quad (19)$$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Projection: from 3D ellipsoids to 2D ellipses.

$$\underbrace{\mu_i}_{\in \mathbb{P}^2,\ \text{2D(image) mean}} = \underbrace{\pi}_{\text{projection}} \left(\underbrace{\mathbf{T}_{cw}}_{\in \mathrm{SE}(3),\ \text{camera pose}} \cdot \underbrace{\mu_w}_{\in \mathbb{P}^3,\ \text{3D(world) mean}} \right) \quad (18)$$

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \underbrace{\Sigma_w}_{\in \mathbb{R}^{3\times 3},\ \text{3D(world) covariance}} \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \quad (19)$$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline ii

Projection: from 3D ellipsoids to 2D ellipses.

$$\mu_i = \pi \left(\mathbf{T}_{cw} \cdot \mu_w \right) \quad (18)$$

μ_w: $\in \mathbb{P}^3$, 3D(world) mean

$\mathbf{T}_{cw}$: $\in \mathrm{SE}(3)$, camera pose

π: projection

μ_i: $\in \mathbb{P}^2$, 2D(image) mean

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^\mathrm{T} \mathbf{J}_\pi^\mathrm{T} \quad (19)$$

Σ_w: $\in \mathbb{R}^{3\times 3}$, 3D(world) covariance

$\mathbf{R}_{cw}$: $\in \mathrm{SO}(3)$, rotation component of $\mathbf{T}_{cw}$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Projection: from 3D ellipsoids to 2D ellipses.

$$\mu_i = \pi \left(\mathbf{T}_{cw} \cdot \mu_w \right) \quad (18)$$

$\in \mathbb{P}^3$, 3D(world) mean

$\in \mathrm{SE}(3)$, camera pose

projection

$\in \mathbb{P}^2$, 2D(image) mean

$$\Sigma_i = \mathbf{J}_\pi \mathbf{R}_{cw} \Sigma_w \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \quad (19)$$

$\in \mathbb{R}^{3\times 3}$, 3D(world) covariance

$\in \mathrm{SO}(3)$, rotation component of $\mathbf{T}_{cw}$

$\in \mathbb{R}^{2\times 3}$, Jacobian of the linear approximation of π

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline ii

Projection: from 3D ellipsoids to 2D ellipses.

$$\underbrace{\mu_i}_{\in \mathbb{P}^2,\ \text{2D(image) mean}} = \underbrace{\pi}_{\text{projection}} \left(\underbrace{\mathbf{T}_{cw}}_{\in \mathrm{SE}(3),\ \text{camera pose}} \cdot \underbrace{\mu_w}_{\in \mathbb{P}^3,\ \text{3D(world) mean}} \right) \quad (18)$$

$$\underbrace{\Sigma_i}_{\in \mathbb{R}^{2\times 2},\ \text{2D(image) covariance}} = \underbrace{\mathbf{J}_\pi}_{\in \mathbb{R}^{2\times 3},\ \text{Jacobian of the linear approximation of } \pi} \underbrace{\mathbf{R}_{cw}}_{\in \mathrm{SO}(3),\ \text{rotation component of } \mathbf{T}_{cw}} \underbrace{\Sigma_w}_{\in \mathbb{R}^{3\times 3},\ \text{3D(world) covariance}} \mathbf{R}_{cw}^{\mathrm{T}} \mathbf{J}_\pi^{\mathrm{T}} \quad (19)$$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Blending: α-blending of semantic embeddings.

$$\mathbf{f}(h, w) = \sum_{i=1}^{N} T_i \alpha_i \mathbf{f}_i(h, w), \quad T_i = \prod_{j=1}^{i-1}(1 - \alpha_j) \tag{20}$$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Blending: α-blending of semantic embeddings.

$$\boxed{\mathbf{f}(h, w)} = \sum_{i=1}^{N} T_i \alpha_i \mathbf{f}_i(h, w), \quad T_i = \prod_{j=1}^{i-1}(1 - \alpha_j) \tag{20}$$

semantic feature on pixel (h, w)

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Blending: α-blending of semantic embeddings.

$$\underbrace{\mathbf{f}(h, w)}_{\text{semantic feature on pixel } (h,w)} = \sum_{i=1}^{N} T_i \alpha_i \underbrace{\mathbf{f}_i(h, w)}_{\text{semantic feature of } i\text{-th Gaussian on pixel } (h,w)}, \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{20}$$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Blending: α-blending of semantic embeddings.

$$\underbrace{\mathbf{f}(h,w)}_{\text{semantic feature on pixel } (h,w)} = \sum_{i=1}^{N} T_i \underbrace{\alpha_i}_{\text{opacity of } i\text{-th Gaussian}} \underbrace{\mathbf{f}_i(h,w)}_{\text{semantic feature of } i\text{-th Gaussian on pixel } (h,w)}, \quad T_i = \prod_{j=1}^{i-1}(1-\alpha_j) \tag{20}$$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Blending: α-blending of semantic embeddings.

$$\underbrace{\mathbf{f}(h, w)}_{\text{semantic feature on pixel } (h,w)} = \sum_{i=1}^{N} \underbrace{T_i}_{\text{background opacity for } i\text{-th Gaussian}} \underbrace{\alpha_i}_{\text{opacity of } i\text{-th Gaussian}} \underbrace{\mathbf{f}_i(h, w)}_{\text{semantic feature of } i\text{-th Gaussian on pixel } (h,w)}, \quad T_i = \prod_{j=1}^{i-1}(1-\alpha_j) \tag{20}$$

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline iii

Blending: α-blending of semantic embeddings.

number of the **sorted & visible** subset of 3D Gaussians

$$\mathbf{f}(h, w) = \sum_{i=1}^{N} T_i \, \alpha_i \, \mathbf{f}_i(h, w), \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{20}$$

semantic feature on pixel (h, w)

semantic feature of i-th Gaussian on pixel (h, w)

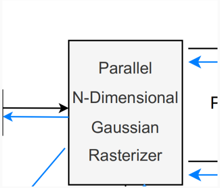
opacity of i-th Gaussian

background opacity for i-th Gaussian

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline iv

Rasterization: tiled and implemented in CUDA.

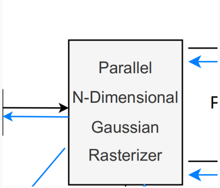


1. Divide the screen space into tiles (CUDA thread blocks).
2. Group the Gaussians by view frustum and tile index.
3. Sort the Gaussians by front-to-back depth order.
4. Blend each pixel within a tile in parallel (CUDA threads).

In practice, 16×16 blocks.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Rasterization: tiled and implemented in CUDA.

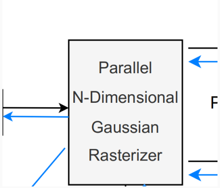


1. Divide the screen space into tiles (CUDA thread blocks).
2. Group the Gaussians by view frustum and tile index.
3. Sort the Gaussians by front-to-back depth order.
4. Blend each pixel within a tile in parallel (CUDA threads).

In practice, 16×16 blocks.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Rasterization: tiled and implemented in CUDA.

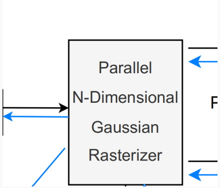


1. Divide the screen space into tiles (CUDA thread blocks).
2. Group the Gaussians by view frustum and tile index.
3. Sort the Gaussians by front-to-back depth order.
4. Blend each pixel within a tile in parallel (CUDA threads).

In practice, 16×16 blocks.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Rasterization: tiled and implemented in CUDA.



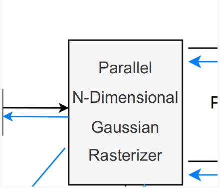
1. Divide the screen space into tiles (CUDA thread blocks).
2. Group the Gaussians by view frustum and tile index.
3. Sort the Gaussians by front-to-back depth order.
4. Blend each pixel within a tile in parallel (CUDA threads).

In practice, 16×16 blocks.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic Rendering Pipeline iv

Rasterization: tiled and implemented in CUDA.

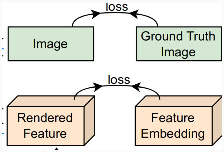


1. Divide the screen space into tiles (CUDA thread blocks).
2. Group the Gaussians by view frustum and tile index.
3. Sort the Gaussians by front-to-back depth order.
4. Blend each pixel within a tile in parallel (CUDA threads).

In practice, 16×16 blocks.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Inverse rendering: guided by image-wise photometric loss,

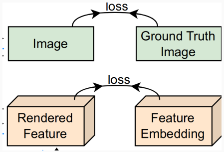


$$\mathcal{L} = \mathcal{L}_{appearance} + \gamma \mathcal{L}_{semantics} \tag{21}$$

In practice, $\gamma = 1, \lambda = 0.2$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Inverse rendering: guided by image-wise photometric loss,

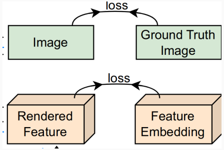


$$\mathcal{L} = \mathcal{L}_{appearance} + \gamma \mathcal{L}_{semantics} \tag{21}$$

In practice, $\gamma = 1$, $\lambda = 0.2$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Inverse rendering: guided by image-wise photometric loss,



$$\mathcal{L} = \mathcal{L}_{appearance} + \gamma \mathcal{L}_{semantics} \tag{21}$$

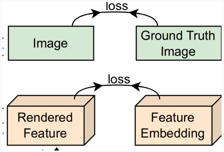
$$\mathcal{L}_{appearance} = (1-\lambda)\mathcal{L}_1\left(\mathbf{C}, \hat{\mathbf{C}}\right) + \lambda \mathcal{L}_{D-SSIM}\left(\mathbf{C}, \hat{\mathbf{C}}\right) \tag{22}$$

(23)

In practice, $\gamma = 1$, $\lambda = 0.2$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Inverse rendering: guided by image-wise photometric loss,



$$\mathcal{L} = \mathcal{L}_{appearance} + \gamma \mathcal{L}_{semantics} \tag{21}$$

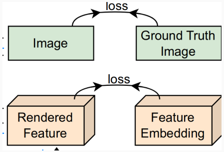
$$\mathcal{L}_{appearance} = (1-\lambda)\mathcal{L}_1\left(\underbrace{\mathbf{C}}_{\text{captured RGB image (GT)}}, \underbrace{\hat{\mathbf{C}}}_{\text{rendered RGB image}}\right) + \lambda \mathcal{L}_{D-SSIM}\left(\mathbf{C}, \hat{\mathbf{C}}\right) \tag{22}$$

(23)

In practice, $\gamma = 1$, $\lambda = 0.2$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Inverse rendering: guided by image-wise photometric loss,



$$\mathcal{L} = \mathcal{L}_{appearance} + \gamma \mathcal{L}_{semantics} \tag{21}$$

captured RGB image (GT)

rendered RGB image

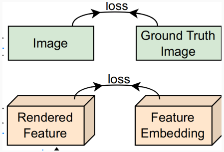
$$\mathcal{L}_{appearance} = (1-\lambda)\mathcal{L}_1\left(\mathbf{C}, \hat{\mathbf{C}}\right) + \lambda \mathcal{L}_{D-SSIM}\left(\mathbf{C}, \hat{\mathbf{C}}\right) \tag{22}$$

$$\mathcal{L}_{semantics} = \mathcal{L}_1\left(\mathbf{F}, \hat{\mathbf{F}}\right) = \sum_{h=1}^{H}\sum_{w=1}^{W} \|\mathbf{f}(h,w) - \hat{\mathbf{f}}(h,w)\|_1 \tag{23}$$

In practice, $\gamma = 1$, $\lambda = 0.2$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Inverse rendering: guided by image-wise photometric loss,



$$\mathcal{L} = \mathcal{L}_{appearance} + \gamma \mathcal{L}_{semantics} \tag{21}$$

$$\mathcal{L}_{appearance} = (1-\lambda)\mathcal{L}_1\Big(\overbrace{\mathbf{C}}^{\text{captured RGB image (GT)}}, \overbrace{\hat{\mathbf{C}}}^{\text{rendered RGB image}}\Big) + \lambda \mathcal{L}_{D-SSIM}\left(\mathbf{C}, \hat{\mathbf{C}}\right) \tag{22}$$

$$\mathcal{L}_{semantics} = \mathcal{L}_1\Big(\underbrace{\mathbf{F}}_{\text{inferred semantic feature map}}, \underbrace{\hat{\mathbf{F}}}_{\text{rendered semantic feature map}}\Big) = \sum_{h=1}^{H}\sum_{w=1}^{W} \|\mathbf{f}(h,w) - \hat{\mathbf{f}}(h,w)\|_1 \tag{23}$$

In practice, $\gamma = 1$, $\lambda = 0.2$.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Motivation

Too **inefficient** to embed naively,

1. High dimension: latent features in large foundation models.
2. Large quantities: millions of Gaussians in a scene.

Solution

1. Compactness: to embed Gaussians with more compact vectors, dim $= D' < D$.
2. Alignment: to align the dimensionalities using a lightweight decoder.

$D = 512$ in CLIP; $D = 256$ in SAM.

In practice, $D' = 128$.

Lightweight decoder: In practice, a 1×1 convolutional layer or a fully-connected layer.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Speed-up Module

Motivation

Too **inefficient** to embed naively,

1. High dimension: latent features in large foundation models.
2. Large quantities: millions of Gaussians in a scene.

Solution

1. Compactness: to embed Gaussians with more compact vectors, $\dim = D' < D$.
2. Alignment: to align the dimensionalities using a lightweight decoder.

$D = 512$ in CLIP; $D = 256$ in SAM.

In practice, $D' = 128$.

Lightweight decoder: In practice, a 1×1 convolutional layer or a fully-connected layer.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Speed-up Module

Motivation

Too **inefficient** to embed naively,

1. High dimension: latent features in large foundation models.
2. Large quantities: millions of Gaussians in a scene.

Solution

1. Compactness: to embed Gaussians with more compact vectors, dim $= D' < D$.
2. Alignment: to align the dimensionalities using a lightweight decoder.

$D = 512$ in CLIP; $D = 256$ in SAM.

In practice, $D' = 128$.

Lightweight decoder: In practice, a 1×1 convolutional layer or a fully-connected layer.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Motivation

Too **inefficient** to embed naively,

1. High dimension: latent features in large foundation models.
2. Large quantities: millions of Gaussians in a scene.

Solution

1. Compactness: to embed Gaussians with more compact vectors, $\dim = D' < D$.
2. Alignment: to align the dimensionalities using a lightweight decoder.

$D = 512$ in CLIP; $D = 256$ in SAM.

In practice, $D' = 128$.

Lightweight decoder: In practice, a 1×1 convolutional layer or a fully-connected layer.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Speed-up Module

Motivation

Too **inefficient** to embed naively,

1. High dimension: latent features in large foundation models.
2. Large quantities: millions of Gaussians in a scene.

Solution

1. Compactness: to embed Gaussians with more compact vectors, $\dim = D' < D$.
2. Alignment: to align the dimensionalities using a lightweight decoder.

$D = 512$ in CLIP; $D = 256$ in SAM.

In practice, $D' = 128$.

Lightweight decoder: In practice, a 1×1 convolutional layer or a fully-connected layer.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Motivation

Too **inefficient** to embed naively,

1. High dimension: latent features in large foundation models.
2. Large quantities: millions of Gaussians in a scene.

Solution

1. Compactness: to embed Gaussians with more compact vectors, $\dim = D' < D$.
2. Alignment: to align the dimensionalities using a lightweight decoder.

$D = 512$ in CLIP; $D = 256$ in SAM.

In practice, $D' = 128$.

Lightweight decoder: In practice, a 1×1 convolutional layer or a fully-connected layer.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

1. Inefficiency
   - "Speed-up module" is not enough,
2. 3D Inconsistency & Inaccuracy
   - 2D foundation models are still 2D.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Limitations

1. Inefficiency
   - "Speed-up module" is not enough,

2. 3D Inconsistency & Inaccuracy
   - 2D foundation models are still 2D.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

1. Inefficiency
   - "Speed-up module" is not enough,

     $\dim = 128$ embedding for millions of Gaussians.
2. 3D Inconsistency & Inaccuracy
   - 2D foundation models are still 2D.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Limitations

1. Inefficiency
   - "Speed-up module" is not enough,
     $\dim = 128$ embedding for millions of Gaussians.
2. 3D Inconsistency & Inaccuracy
   - 2D foundation models are still 2D.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Limitations

1. Inefficiency
   - "Speed-up module" is not enough,
     $\dim = 128$ embedding for millions of Gaussians.
2. 3D Inconsistency & Inaccuracy
   - 2D foundation models are still 2D.

(CVPR Highlight, 2024) Feature 3DGS: Supercharging 3D Gaussian Splatting to Enable Distilled Feature Fields

Semantic 3DGS

LangSplat

Key Insight

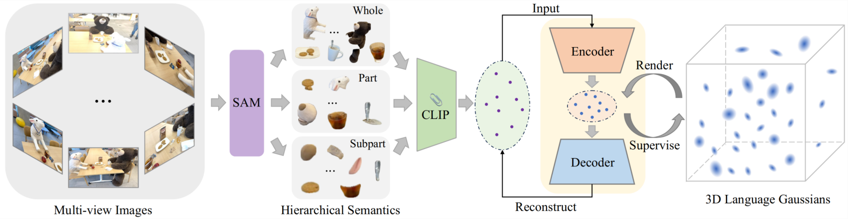


Figure 2: Overview of LangSplat

1. Accuracy: SAM outputs to enhance CLIP features.
   - CLIP: image-aligned training leads to "point-ambiguity".
   - SAM: pixel-aligned & object-centered & multi-granularity.

(CVPR Highlight, 2024) LangSplat: 3D Language Gaussian Splatting

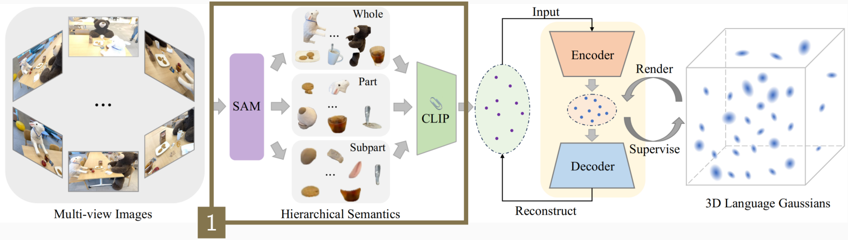


Figure 2: Overview of LangSplat

1 Accuracy: SAM outputs to enhance CLIP features.

- CLIP: image-aligned training leads to "point-ambiguity".
- SAM: pixel-aligned & object-centered & multi-granularity.

(CVPR Highlight, 2024) LangSplat: 3D Language Gaussian Splatting

Key Insight

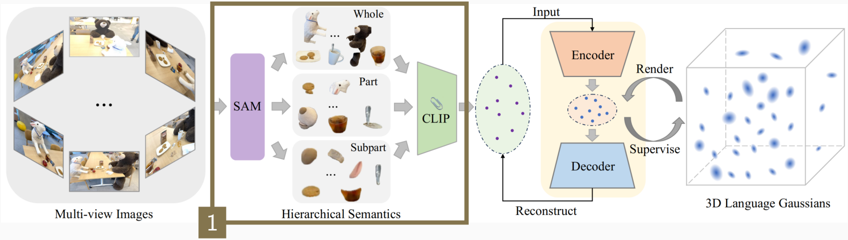


Figure 2: Overview of LangSplat

1 Accuracy: SAM outputs to enhance CLIP features.

- CLIP: image-aligned training leads to "point-ambiguity".
- SAM: pixel-aligned & object-centered & multi-granularity.

(CVPR Highlight, 2024) LangSplat: 3D Language Gaussian Splatting

Key Insight

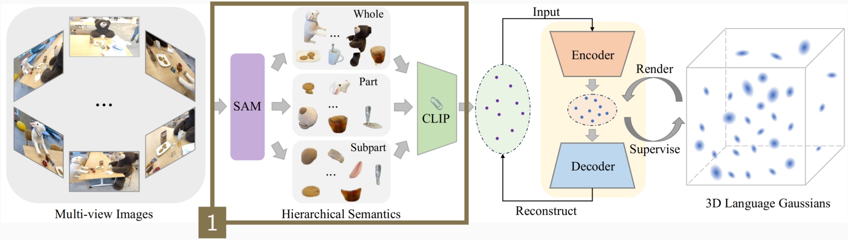


Figure 2: Overview of LangSplat

1 Accuracy: SAM outputs to enhance CLIP features.

- CLIP: image-aligned training leads to "point-ambiguity".
- SAM: pixel-aligned & object-centered & multi-granularity.

(CVPR Highlight, 2024) LangSplat: 3D Language Gaussian Splatting

Key Insight

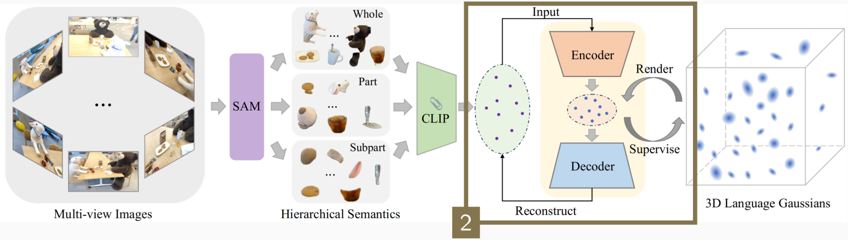


Figure 2: Overview of LangSplat

2 Efficiency: an auto-encoder to compress latent features.

- More complexity and better compression, compared with "speed-up module" in Feature 3DGS [8].

In practice, $\dim = 3$.

(CVPR Highlight, 2024) LangSplat: 3D Language Gaussian Splatting

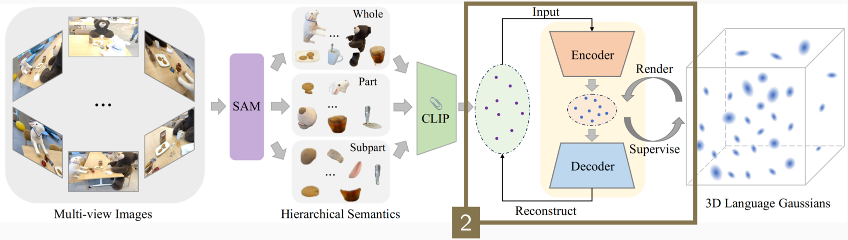


Figure 2: Overview of LangSplat

2 Efficiency: an auto-encoder to compress latent features.

- More complexity and better compression,

  compared with "speed-up module" in Feature 3DGS [8].

In practice, $\mathrm{dim} = 3$.

(CVPR Highlight, 2024) LangSplat: 3D Language Gaussian Splatting

Semantic 3DGS

CLIP-GS

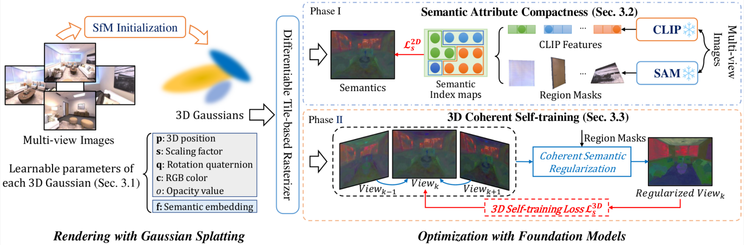


Figure 3: Overview of CLIP-GS

1. Efficiency: unify semantic features within an object by leveraging SAM.
2. Consistency: supervise consecutive frames by video segmentation.

(arXiv 04/2024) CLIP-GS: CLIP-informed Gaussian Splatting for Real-time and View-consistent 3D Semantic Understanding

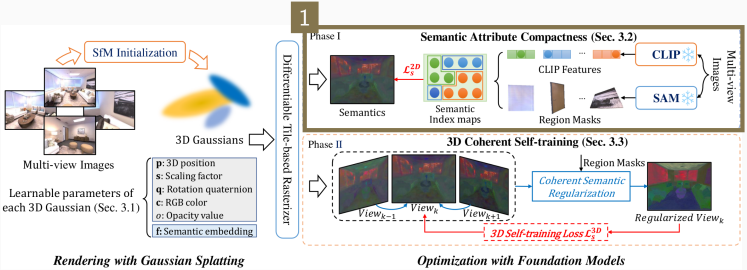


Figure 3: Overview of CLIP-GS

1 Efficiency: unify semantic features within an object by leveraging SAM.

2 Consistency: supervise consecutive frames by video segmentation.

(arXiv 04/2024) CLIP-GS: CLIP-informed Gaussian Splatting for Real-time and View-consistent 3D Semantic Understanding

Key Insight

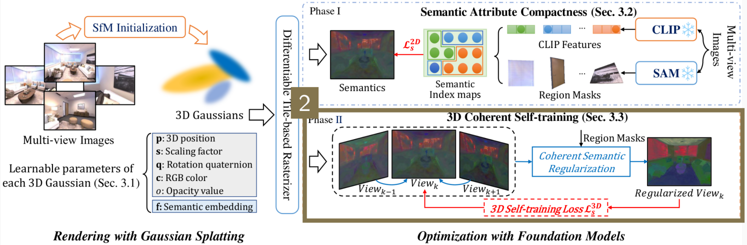


Figure 3: Overview of CLIP-GS

1 Efficiency: unify semantic features within an object by leveraging SAM.

2 Consistency: supervise consecutive frames by video segmentation.

(arXiv 04/2024) CLIP-GS: CLIP-informed Gaussian Splatting for Real-time and View-consistent 3D Semantic Understanding

References

References i

[1] N. Keetha, J. Karhade, K. M. Jatavallabhula, *et al.*, *SplaTAM: Splat, track & map 3d gaussians for dense RGB-d SLAM*, Apr. 16, 2024. arXiv: 2312.02126[cs]. [Online]. Available: http://arxiv.org/abs/2312.02126 (visited on 05/20/2024) (cit. on p. iv).

[2] C. Yan, D. Qu, D. Wang, *et al.*, *GS-SLAM: Dense visual SLAM with 3d gaussian splatting*, Nov. 21, 2023. arXiv: 2311.11700[cs]. [Online]. Available: http://arxiv.org/abs/2311.11700 (visited on 12/26/2023) (cit. on p. iv).

[3] V. Yugay, Y. Li, T. Gevers, and M. R. Oswald, *Gaussian-SLAM: Photo-realistic dense SLAM with gaussian splatting*, Mar. 22, 2024. arXiv: 2312.10070[cs]. [Online]. Available: http://arxiv.org/abs/2312.10070 (visited on 03/27/2024) (cit. on p. iv).

[4] H. Matsuki, R. Murai, P. H. J. Kelly, and A. J. Davison, *Gaussian splatting SLAM*, Apr. 14, 2024. arXiv: 2312.06741[cs]. [Online]. Available: http://arxiv.org/abs/2312.06741 (visited on 05/20/2024) (cit. on p. iv).

[5] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," in *arXiv:1607.02565*, Jul. 2016 (cit. on pp. xxvii–xxix).

[6] J.-C. Shi, M. Wang, H.-B. Duan, and S.-H. Guan, *Language embedded 3d gaussians for open-vocabulary scene understanding*, Nov. 30, 2023. arXiv: 2311.18482[cs]. [Online]. Available: http://arxiv.org/abs/2311.18482 (visited on 06/08/2024) (cit. on pp. xcv–xcvii).

[7] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister, *LangSplat: 3d language gaussian splatting*, Dec. 26, 2023. arXiv: 2312.16084[cs]. [Online]. Available: http://arxiv.org/abs/2312.16084 (visited on 02/23/2024) (cit. on pp. xcv–xcvii).

[8] S. Zhou, H. Chang, S. Jiang, *et al.*, *Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields*, Apr. 8, 2024. arXiv: 2312.03203[cs]. [Online]. Available: http://arxiv.org/abs/2312.03203 (visited on 05/22/2024) (cit. on pp. xcv–xcvii, clxiv, clxv).

[9] M. Ye, M. Danelljan, F. Yu, and L. Ke, *Gaussian grouping: Segment and edit anything in 3d scenes*, Dec. 1, 2023. arXiv: 2312.00732[cs]. [Online]. Available: http://arxiv.org/abs/2312.00732 (visited on 01/02/2024) (cit. on pp. xcv–xcvii).

References ii

[10] J. Cen, J. Fang, C. Yang, *et al.*, *Segment any 3d gaussians*, Dec. 1, 2023. arXiv: 2312.00860[cs]. [Online]. Available: http://arxiv.org/abs/2312.00860 (visited on 03/12/2024) (cit. on pp. xcv–xcvii).

[11] B. Dou, T. Zhang, Y. Ma, Z. Wang, and Z. Yuan, *CoSSegGaussians: Compact and swift scene segmenting 3d gaussians with dual feature fusion*, Jan. 30, 2024. arXiv: 2401.05925[cs]. [Online]. Available: http://arxiv.org/abs/2401.05925 (visited on 06/08/2024) (cit. on pp. xcv–xcvii).

[12] J. Guo, X. Ma, Y. Fan, H. Liu, and Q. Li, *Semantic gaussians: Open-vocabulary scene understanding with 3d gaussian splatting*, Mar. 22, 2024. arXiv: 2403.15624[cs]. [Online]. Available: http://arxiv.org/abs/2403.15624 (visited on 05/20/2024) (cit. on pp. xcv–xcvii).

[13] R.-Z. Qiu, G. Yang, W. Zeng, and X. Wang, *Feature splatting: Language-driven physics-based scene synthesis and editing*, Apr. 1, 2024. arXiv: 2404.01223[cs]. [Online]. Available: http://arxiv.org/abs/2404.01223 (visited on 06/08/2024) (cit. on pp. xcv–xcvii).

[14] G. Liao, J. Li, Z. Bao, *et al.*, *CLIP-GS: CLIP-informed gaussian splatting for real-time and view-consistent 3d semantic understanding*, Apr. 22, 2024. arXiv: 2404.14249[cs]. [Online]. Available: http://arxiv.org/abs/2404.14249 (visited on 05/20/2024) (cit. on pp. xcv–xcvii).

[15] Y. Qu, S. Dai, X. Li, *et al.*, *GOI: Find 3d gaussians of interest with an optimizable open-vocabulary semantic-space hyperplane*, May 27, 2024. arXiv: 2405.17596[cs]. [Online]. Available: http://arxiv.org/abs/2405.17596 (visited on 06/08/2024) (cit. on p. xcv).

[16] M.-B. Jurca, R. Royen, I. Giosan, and A. Munteanu, *RT-GS2: Real-time generalizable semantic segmentation for 3d gaussian representations of radiance fields*, May 28, 2024. arXiv: 2405.18033[cs]. [Online]. Available: http://arxiv.org/abs/2405.18033 (visited on 06/08/2024) (cit. on pp. xcv–xcvii).

[17] B. Xiong, X. Ye, T. H. E. Tse, K. Han, S. Cui, and Z. Li, *SA-GS: Semantic-aware gaussian splatting for large scene reconstruction with geometry constrain*, May 28, 2024. arXiv: 2405.16923[cs]. [Online]. Available: http://arxiv.org/abs/2405.16923 (visited on 06/08/2024) (cit. on p. xcv).

References iii

[18] Y. Ji, H. Zhu, J. Tang, *et al.*, *FastLGS: Speeding up language embedded gaussians with feature grid mapping*, Jun. 3, 2024. arXiv: 2406.01916[cs]. [Online]. Available: http://arxiv.org/abs/2406.01916 (visited on 06/08/2024) (cit. on pp. xcv–xcvii).