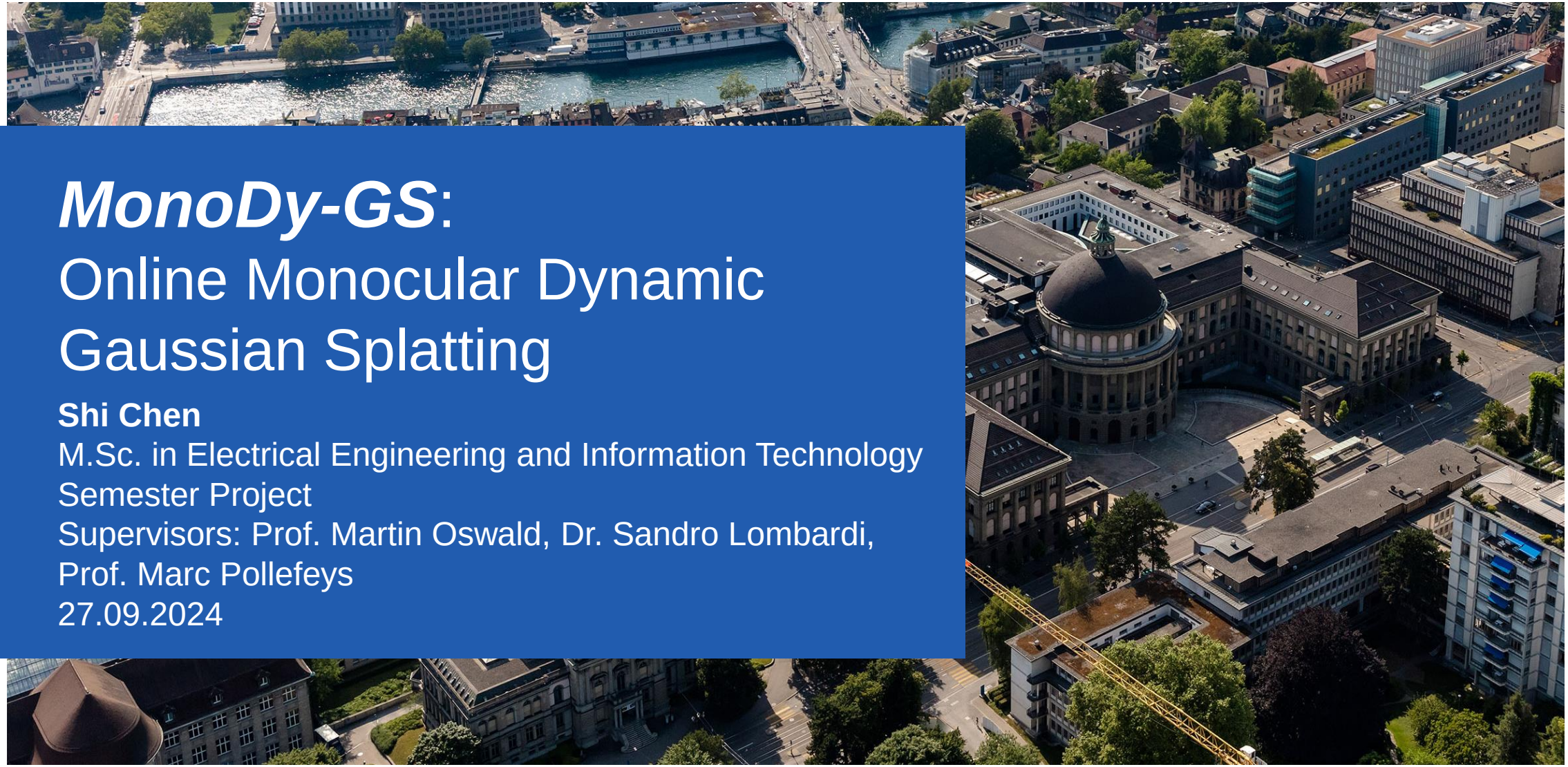
ETH zürich

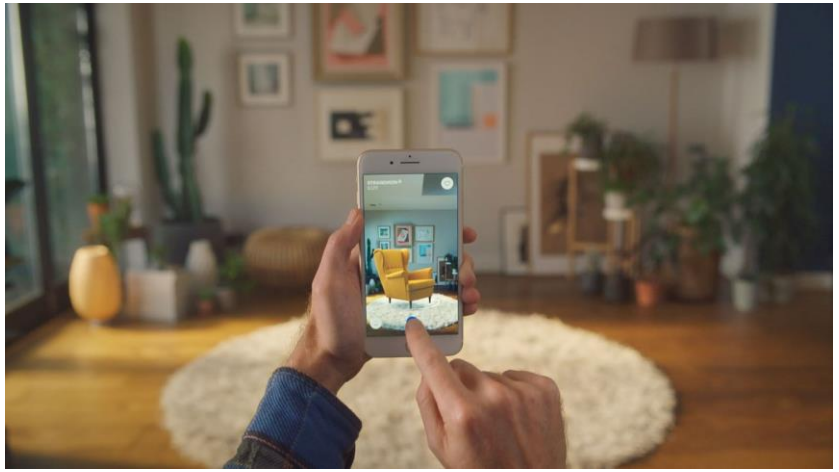


MonoDy-GS: Online Monocular Dynamic Gaussian Splatting

Shi Chen
M.Sc. in Electrical Engineering and Information Technology
Semester Project
Supervisors: Prof. Martin Oswald, Dr. Sandro Lombardi, Prof. Marc Pollefeys
27.09.2024

Motivation: Interactive Mixed Reality

- **Photorealistic** rendering
- **Fast** rendering
- **Dynamic** (4D)
- 4D reconstruction from **monocular** video
- **Online** 4D reconstruction



(https://www.ikea.com/global/en/newsroom/innovation/ikea-launches-ikea-place-a-new-app-that-allows-people-to-virtually-place-furniture-in-their-home-170912/)

(https://pokemongolive.com/de/)

3D Gaussian Splatting [Kerbl 2023]

- ✓ **Photorealistic** rendering
- ✓ **Fast** rendering
- ✗ **Dynamic** (4D)
- ✗ 4D reconstruction from **monocular** video
- ✗ **Online** 4D reconstruction

(https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/)

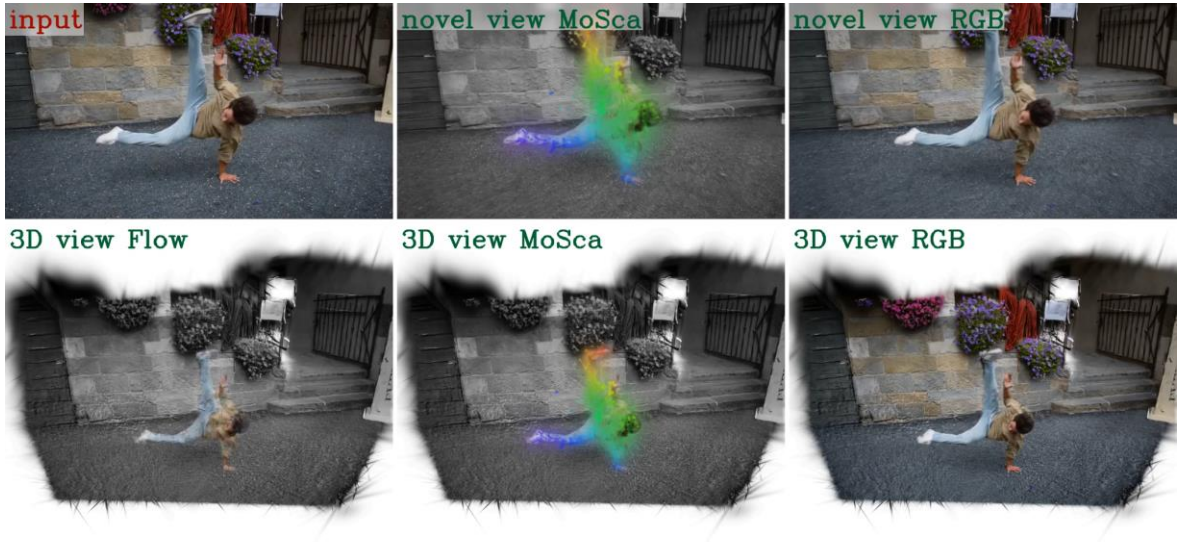
Dynamic 3D Gaussians [Luiten 2024]

- ✓ **Photorealistic** rendering
- ✓ **Fast** rendering
- ✓ **Dynamic** (4D)
- ✗ 4D reconstruction from **monocular** video
- ✓ **Online** 4D reconstruction

(https://dynamic3dgaussians.github.io/)

MoSca [Lei 2024]

- ✓ **Photorealistic** rendering
- ✓ **Fast** rendering
- ✓ **Dynamic** (4D)
- ✓ 4D reconstruction from **monocular** video
- ✗ **Online** 4D reconstruction



(https://www.cis.upenn.edu/~leijh/projects/mosca/)

Goal: **Online Monocular Dynamic** Gaussian Splatting

- ✓ **Photorealistic** rendering
- ✓ **Fast** rendering
- ✓ **Dynamic** (4D)
- ✓ 4D reconstruction from **monocular** video
- ✓ **Online** 4D reconstruction

How?

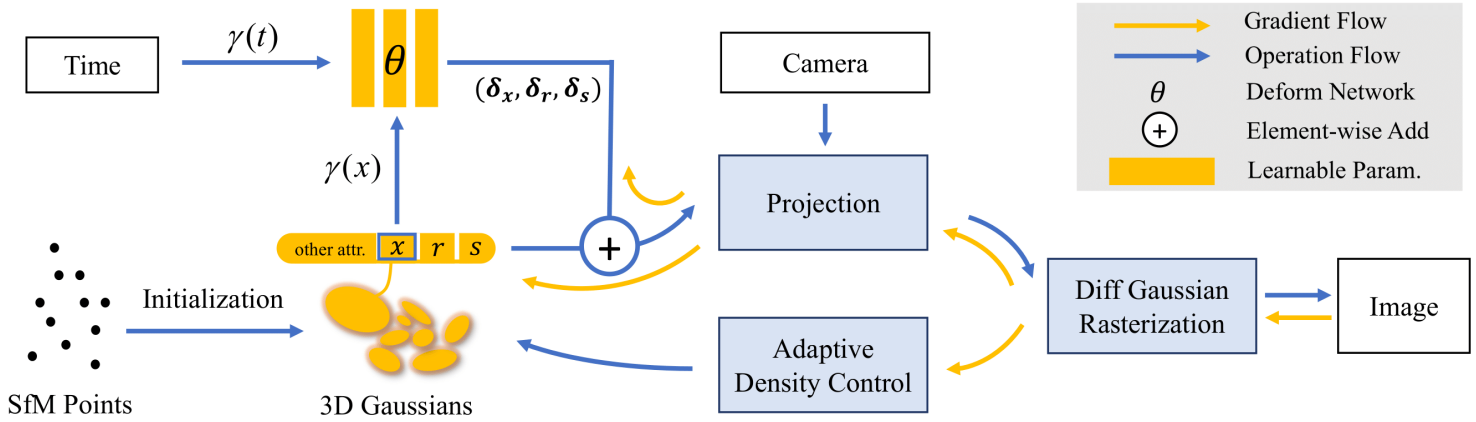
Codebase: MonoGS [Matsuki 2024]

- Gaussian-based SLAM system
- Accurate tracking and mapping
- Multi-process structure:
  - Frontend: online camera tracking, keyframe detection
  - Backend: multiview mapping + camera refinement (BA)
- Assumes **static** scene

(https://rmurai.co.uk/projects/GaussianSplattingSLAM/)

How to handle dynamic scenes using Gaussians?

- Option 1: Canonical Space + Deformation Field
  - Gaussians anchored in shared canonical space
  - MLP to encode dense deformation field
  - \+ Most common
  - \+ Straightforward to implement
  - – Struggles with **complex deformations** (e.g. Topological changes)
  - – **Slow** optimization
  - – Catastrophic **forgetting**

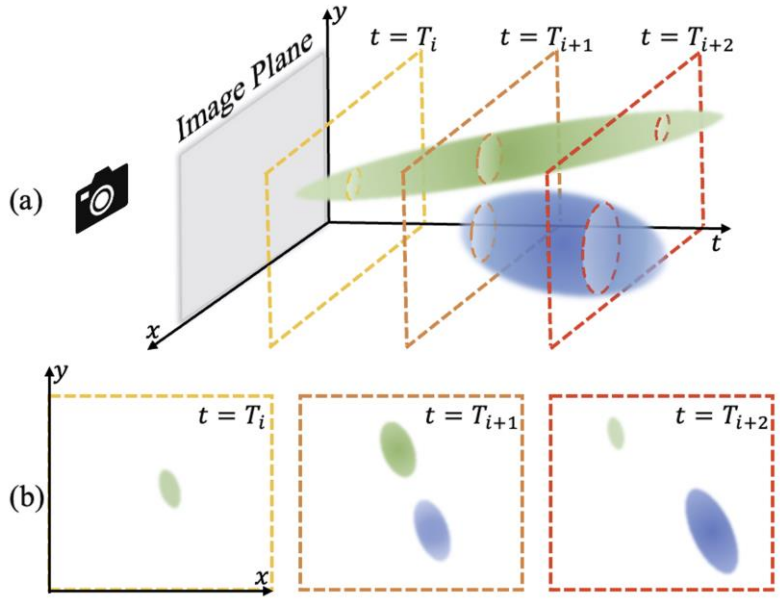


Deformable 3D Gaussians [Yang 2024] (https://ingra14m.github.io/Deformable-Gaussians/)

How to handle dynamic scenes using Gaussians?

- Option 2: 4D Gaussians
  - Extends 3DGS to a 4th dimension of time
  - Gaussians can appear transiently
  + Straightforward to implement
  – Does **not** represent dynamic parts of scene with **consistent set of Gaussians**

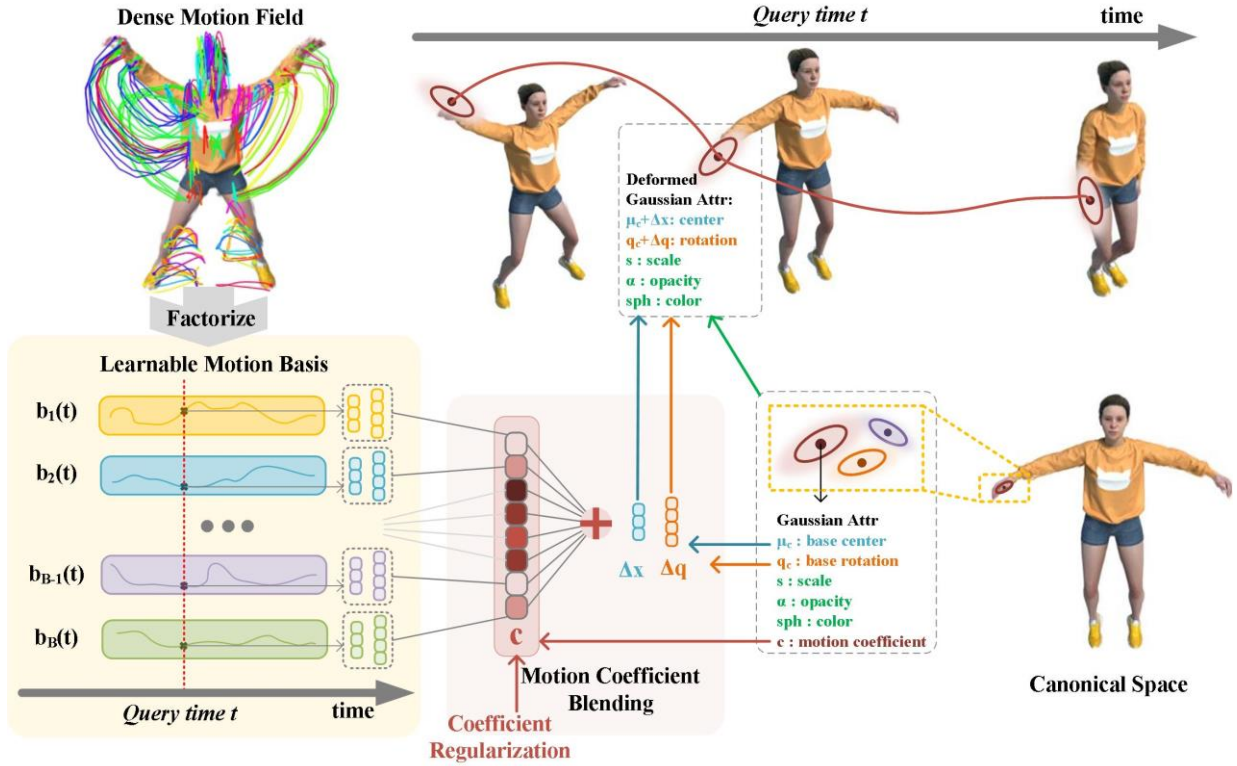
    → Hard to maintain temporal consistency in appearance and geometry



4D Gaussian Splatting [Duan 2024]

How to handle dynamic scenes using Gaussians?

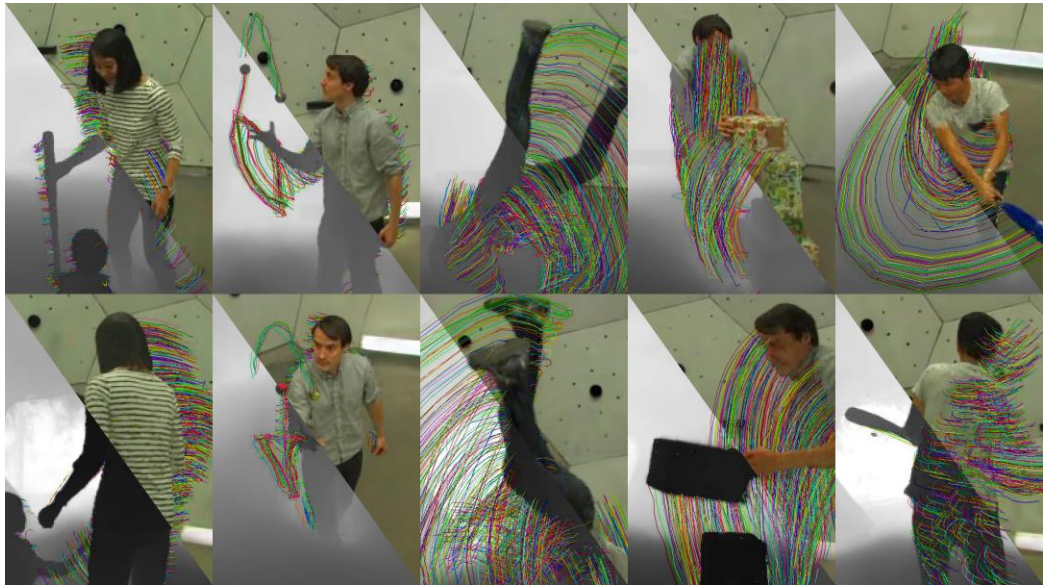
- Option 3: Gaussian Trajectory Decomposition
  - Decomposes each Gaussian's trajectory into combination of fixed or learnable bases
  + Leverages **low-rank** nature of Gaussian dynamics
  – **Compressed** representations of Gaussian dynamics → Compromised accuracy



DynMF [Kratimenos 2024] (https://agelosk.github.io/dynmf/)

How to handle dynamic scenes using Gaussians?

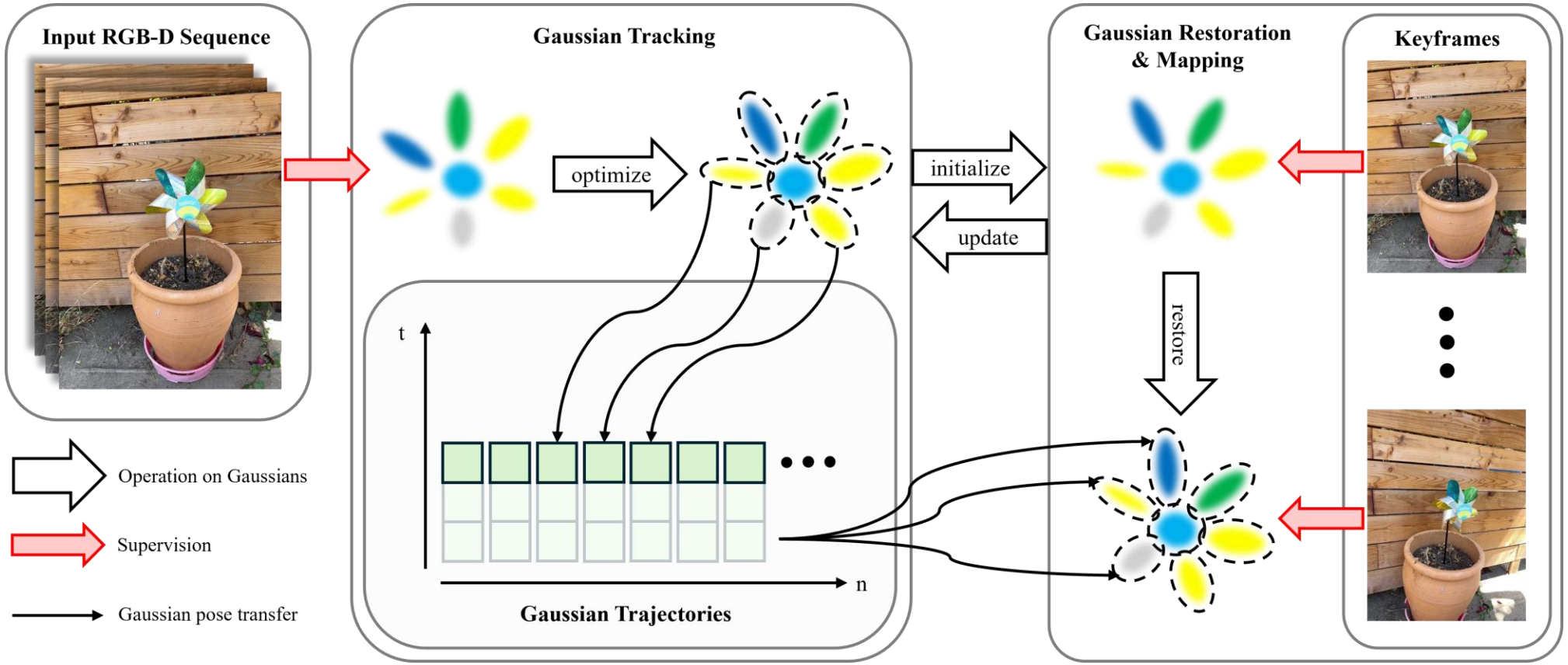
- Option 4: Direct Gaussian Tracking
  - Introduced in *Dynamic 3D Gaussians* [Luiten 2024]
  - Optimizes motion of each Gaussian separately
  + **Fast** optimization
  + **Uncompressed** representation of Gaussian dynamics
  – Requires sufficient **multiview** coverage
  - We adapt it to **monocular** setup by introducing various **priors** to regularize optimization!



Dynamic 3D Gaussians [Luiten 2024] (https://github.com/JonathonLuiten/Dynamic3DGaussians)

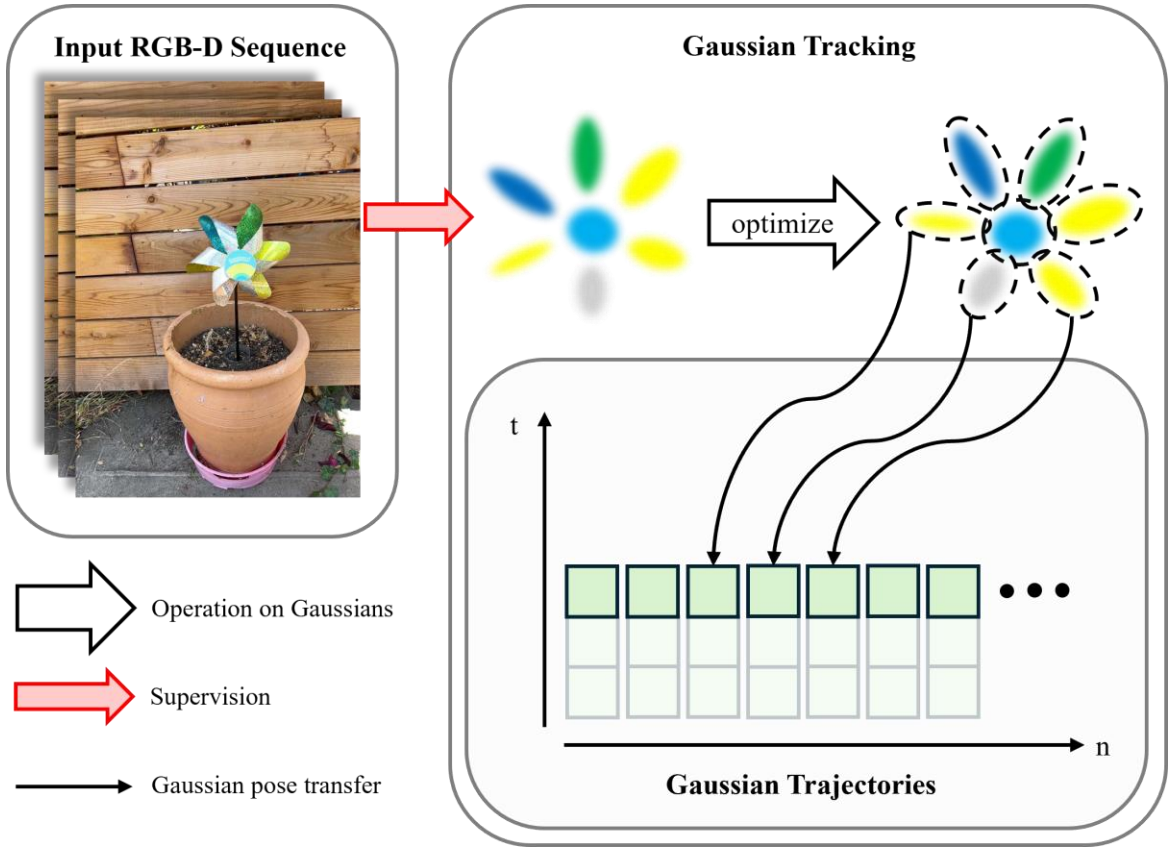
Method Overview

- Assumptions: RGB-D input, known camera poses (applies to all experiments)
- Frontend: **Gaussian Tracking** using every new-coming frame
- Backend: **Gaussian Restoration & Mapping** using keyframes



Gaussian Tracking

- *Gaussian pose* (center position + orientation) optimized through **RGB-D supervision** and various **regularizations**
- *Gaussian Trajectories*: Record of all Gaussian poses at every frame up to the current one



Loss Functions: RGB-D supervision

- Photometric rerendering loss:

Rendered image Observed image

$$\mathcal{L}_{\text{pho}} = \| I(\mathcal{G}, \boldsymbol{T}) - I_{\text{obs}} \|_1$$

- Depth rerendering loss:

Rendered depth Observed depth

$$\mathcal{L}_{\text{depth}} = \| D(\mathcal{G}, \boldsymbol{T}) - D_{\text{obs}} \|_1$$

- Following MonoGS, with $\lambda_{\text{pho}} = 0.9$:

$$\mathcal{L}_{\text{RGB-D}} = \lambda_{\text{pho}} \mathcal{L}_{\text{pho}} + (1 - \lambda_{\text{pho}}) \mathcal{L}_{\text{depth}}$$

Loss Functions: Optical Flow Prior

- Use optical flow estimated by *RAFT* [Teed 2020] as pseudo-GT
- Render forward and backward optical flow from Gaussians at frame $t-1$ and frame t:
  - Contribution of each Gaussian:

$$\hat{f}^i_{\text{fwd}} = \mathbf{J}^i_t \boldsymbol{\mu}^i_t - \mathbf{J}^i_{t-1} \boldsymbol{\mu}^i_{t-1}$$
$$\hat{f}^i_{\text{bwd}} = \mathbf{J}^i_{t-1} \boldsymbol{\mu}^i_{t-1} - \mathbf{J}^i_t \boldsymbol{\mu}^i_t$$

  - alpha-blending:

$$\hat{f}_{\text{fwd}}(\boldsymbol{u}) = \sum_{i=1}^{N} \hat{f}^i_{\text{fwd}} \alpha^i_{t-1}(\boldsymbol{u}) \prod_{j=1}^{i-1} (1 - \alpha^j_{t-1}(\boldsymbol{u}))$$
$$\hat{f}_{\text{bwd}}(\boldsymbol{u}) = \sum_{i=1}^{N} \hat{f}^i_{\text{bwd}} \alpha^i_{t}(\boldsymbol{u}) \prod_{j=1}^{i-1} (1 - \alpha^j_{t}(\boldsymbol{u}))$$

Example of rendered optical flow

Loss Functions: Optical Flow Prior

- Reused native CUDA-optimized rasterizer for acceleration
- Optical flow loss as L1 error between rendered and RAFT-estimated flow:

$$\mathcal{L}_{\text{flow}} = \|\hat{f}_{\text{fwd}} - f_{\text{fwd}}\|_1 + \|\hat{f}_{\text{bwd}} - f_{\text{bwd}}\|_1$$

Example of rendered optical flow

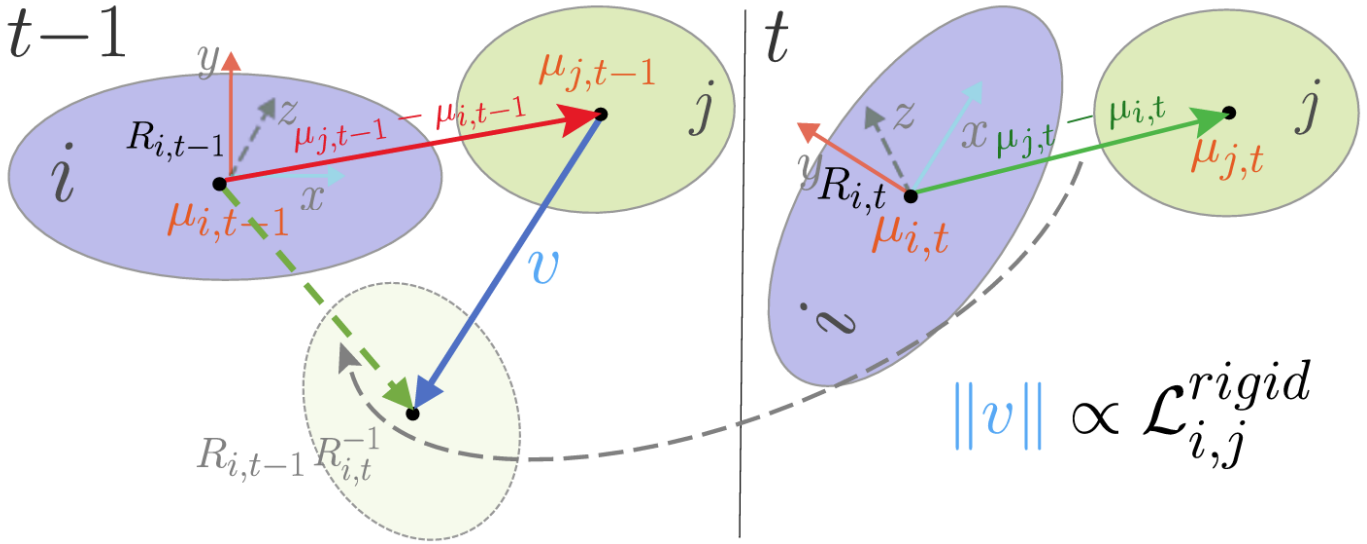
Loss Functions: Physically-Based Priors

We follow *Dynamic 3D Gaussians* [Luiten 2024] to incorporate 3 physically-based priors:

- **Local-rigidity loss** → kNN Gaussians move while following rigid-body transformation:

$$\mathcal{L}_{\text{rigid}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left\| (\boldsymbol{\mu}_{t-1}^{j} - \boldsymbol{\mu}_{t-1}^{i}) - \mathbf{R}_{t-1}^{i} \mathbf{R}_{t}^{i-1} (\boldsymbol{\mu}_{t}^{j} - \boldsymbol{\mu}_{t}^{i}) \right\|_2$$

$$w^{i,j} = \exp\left(-\lambda_w \left\| \boldsymbol{\mu}_{t-1}^{j} - \boldsymbol{\mu}_{t-1}^{i} \right\|_2^2 \right)$$



Dynamic 3D Gaussians [Luiten 2024]

Loss Functions: Physically-Based Priors

We follow *Dynamic 3D Gaussians* [Luiten 2024] to incorporate 3 physically-based priors:

- **Local-rigidity loss** → kNN Gaussians move while following rigid-body transformation:

$$\mathcal{L}_{\text{rigid}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left\| (\boldsymbol{\mu}_{t-1}^{j} - \boldsymbol{\mu}_{t-1}^{i}) - \mathbf{R}_{t-1}^{i} \mathbf{R}_{t}^{i\,-1} (\boldsymbol{\mu}_{t}^{j} - \boldsymbol{\mu}_{t}^{i}) \right\|_2$$

- **Local-rotation similarity loss** → kNN Gaussians maintain similar rotations:

$$\mathcal{L}_{\text{rot}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left\| \boldsymbol{q}_{t}^{j} \boldsymbol{q}_{t-1}^{j\,-1} - \boldsymbol{q}_{t}^{i} \boldsymbol{q}_{t-1}^{i\,-1} \right\|_2$$

Loss Functions: Physically-Based Priors

We follow *Dynamic 3D Gaussians* [Luiten 2024] to incorporate 3 physically-based priors:

- **Local-rigidity loss** → kNN Gaussians move while following rigid-body transformation:

$$\mathcal{L}_{\text{rigid}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left\| (\boldsymbol{\mu}_{t-1}^{j} - \boldsymbol{\mu}_{t-1}^{i}) - \mathbf{R}_{t-1}^{i} \mathbf{R}_{t}^{i\,-1} (\boldsymbol{\mu}_{t}^{j} - \boldsymbol{\mu}_{t}^{i}) \right\|_2$$

- **Local-rotation similarity loss** → kNN Gaussians maintain similar rotations:

$$\mathcal{L}_{\text{rot}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left\| \boldsymbol{q}_{t}^{j} \boldsymbol{q}_{t-1}^{j\,-1} - \boldsymbol{q}_{t}^{i} \boldsymbol{q}_{t-1}^{i\,-1} \right\|_2$$

- **Local-isometry loss** → Distances between kNN Gaussians remain consistent:

$$\mathcal{L}_{\text{iso}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left| \left\| \boldsymbol{\mu}_{t-1}^{j} - \boldsymbol{\mu}_{t-1}^{i} \right\|_2 - \left\| \boldsymbol{\mu}_{t}^{j} - \boldsymbol{\mu}_{t}^{i} \right\|_2 \right|$$

Loss Functions: Physically-Based Priors

We follow *Dynamic 3D Gaussians* [Luiten 2024] to incorporate 3 physically-based priors:

- **Local-rigidity loss** → kNN Gaussians move while following rigid-body transformation:

$$\mathcal{L}_{\text{rigid}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left\| (\boldsymbol{\mu}_{t-1}^{j} - \boldsymbol{\mu}_{t-1}^{i}) - \mathbf{R}_{t-1}^{i} \mathbf{R}_{t}^{i\,-1} (\boldsymbol{\mu}_{t}^{j} - \boldsymbol{\mu}_{t}^{i}) \right\|_2$$

- **Local-rotation similarity loss** → kNN Gaussians maintain similar rotations:

$$\mathcal{L}_{\text{rot}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left\| \boldsymbol{q}_{t}^{j} \boldsymbol{q}_{t-1}^{j\,-1} - \boldsymbol{q}_{t}^{i} \boldsymbol{q}_{t-1}^{i\,-1} \right\|_2$$

- **Local-isometry loss** → Distances between kNN Gaussians remain consistent:

$$\mathcal{L}_{\text{iso}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w^{i,j} \left| \left\| \boldsymbol{\mu}_{t-1}^{j} - \boldsymbol{\mu}_{t-1}^{i} \right\|_2 - \left\| \boldsymbol{\mu}_{t}^{j} - \boldsymbol{\mu}_{t}^{i} \right\|_2 \right|$$

- Total physically-based loss:

$$\mathcal{L}_{\text{phys}} = \lambda_{\text{rigid}} \mathcal{L}_{\text{rigid}} + \lambda_{\text{rot}} \mathcal{L}_{\text{rot}} + \lambda_{\text{iso}} \mathcal{L}_{\text{iso}}$$

Loss Functions: Motion Sparsity Regularization

L1 regularization on all Gaussian motions to enforce sparsity (i.e. Most Gaussians stay static)

$$\mathcal{L}_{\text{sparse}} = \|\Delta\tilde{\boldsymbol{p}}_t\|_1,\ \Delta\tilde{\boldsymbol{p}}_t = [\Delta\tilde{\boldsymbol{p}}_t^1, \Delta\tilde{\boldsymbol{p}}_t^2, \ldots, \Delta\tilde{\boldsymbol{p}}_t^{|\mathcal{G}|}]$$

$$\Delta\tilde{\boldsymbol{p}}_t^i = \begin{bmatrix} \Delta\boldsymbol{\mu}_t^i \\ \lambda_{\boldsymbol{q}}\Delta\boldsymbol{q}_t^i \end{bmatrix}, \quad \Delta\boldsymbol{\mu}_t^i = \boldsymbol{\mu}_t^i - \boldsymbol{\mu}_{t-1}^i, \quad \Delta\boldsymbol{q}_t^i = \boldsymbol{q}_t^i - \boldsymbol{q}_{t-1}^i$$

Concatenation — **Positional change** — **rotational change**

Concatenation → **unified sparsity pattern** for both positional and rotational changes

Rendered image

Positional changes

Rotational changes

Loss Functions: Motion TV Regularization

2D total variation (TV) regularization on rasterized Gaussian translational motions

→ **Piecewise constant** motion fields

- Rasterize Gaussian translational motions into 2D via alpha-blending (native CUDA rasterizer):

$$\Delta\boldsymbol{\mu}_{\text{2D}}(\boldsymbol{u}) = \sum_{i=1}^{N} \Delta\boldsymbol{\mu}_t^i \alpha_{t-1}^i(\boldsymbol{u}) \prod_{j=1}^{i-1} (1 - \alpha_{t-1}^j(\boldsymbol{u}))$$

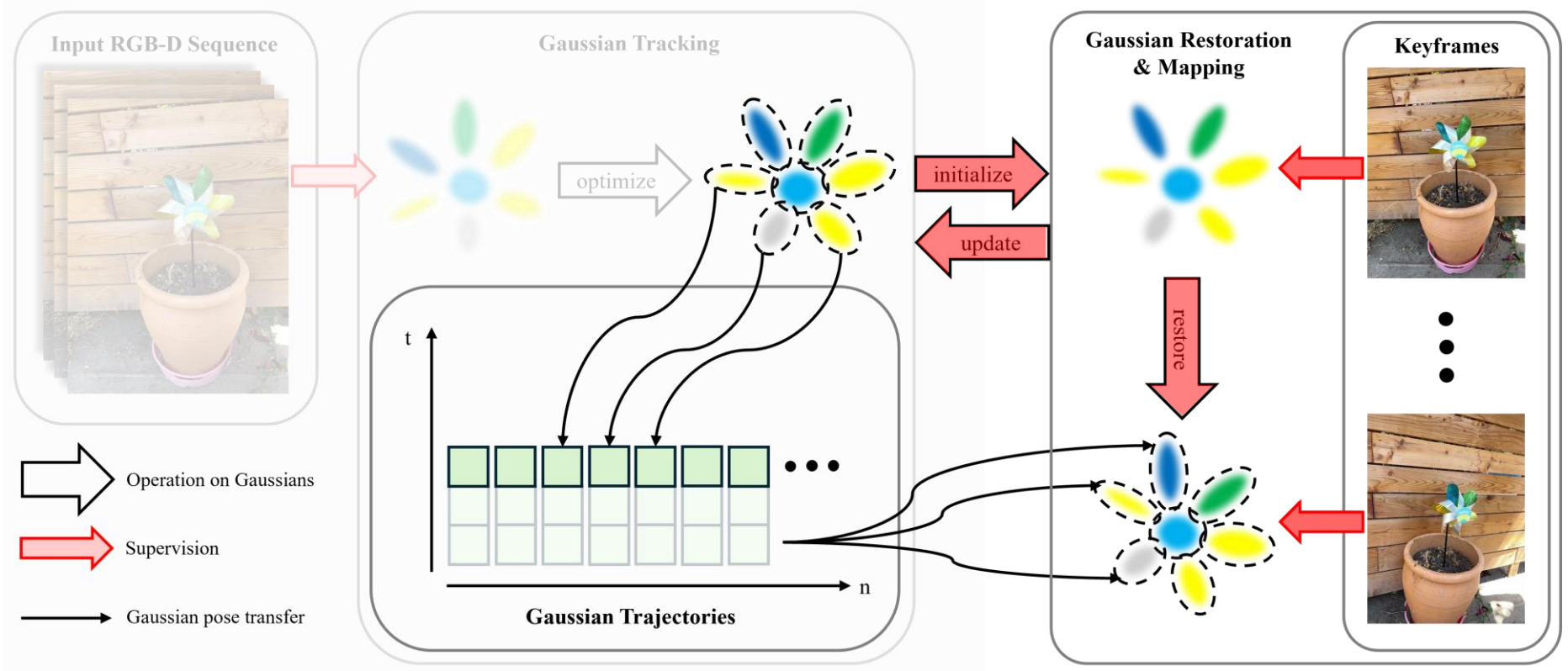
- 2D TV regularization term given as:

$$\mathcal{L}_{\text{TV}} = \sum_{u,v} \left| \frac{\partial \Delta\boldsymbol{\mu}_{\text{2D}}(\boldsymbol{u})}{\partial u} \right| + \left| \frac{\partial \Delta\boldsymbol{\mu}_{\text{2D}}(\boldsymbol{u})}{\partial v} \right|$$

Gaussian Restoration & Mapping

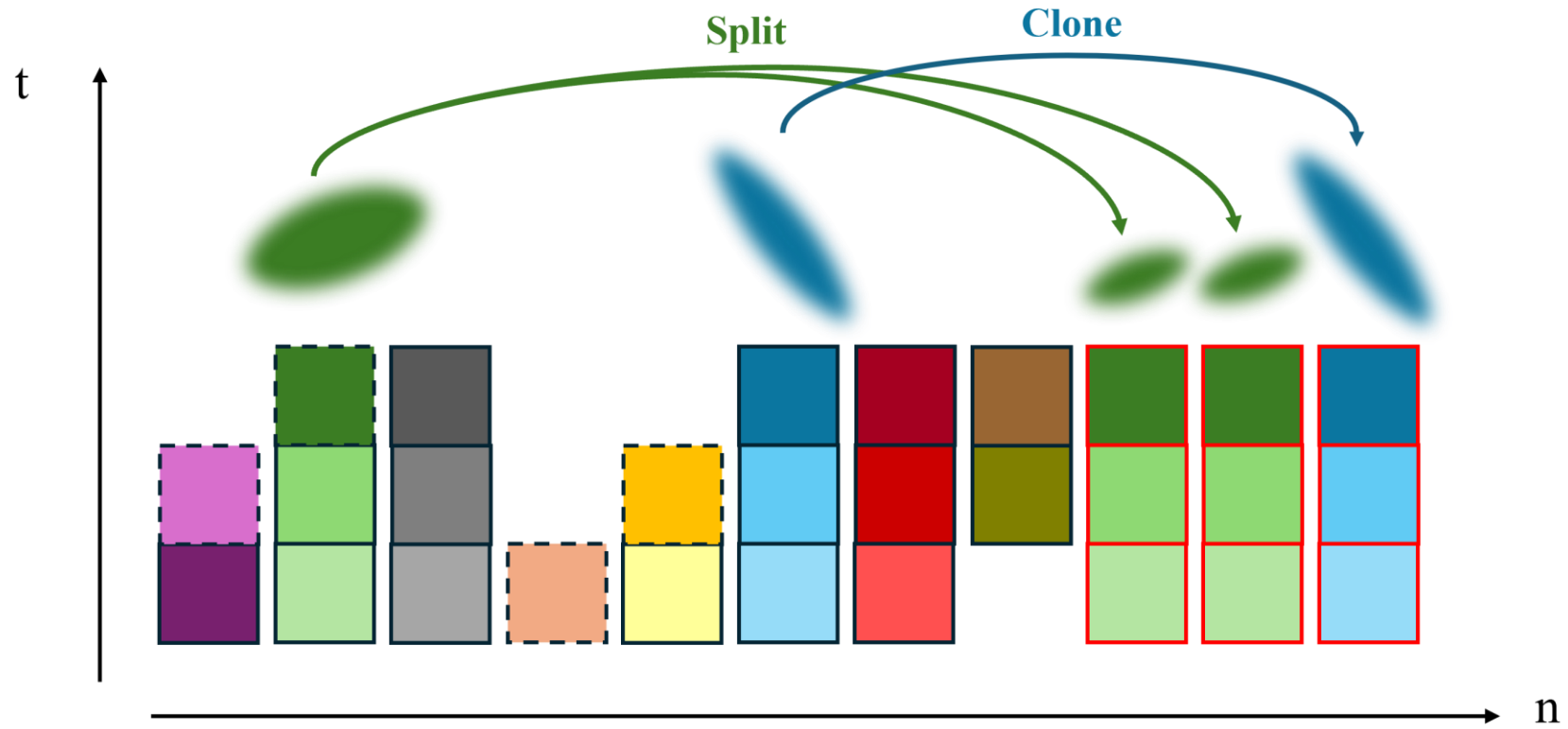
Following MonoGS, every time frontend detects a new keyframe, it requests a *keyframe update:*

1. Frontend sends latest Gaussian attributes & Gaussian trajectories at selected keyframes to backend
2. Backend **restores Gaussian poses** at past keyframes based on latest Gaussians
3. Backend performs **motion-aware multiview mapping**
4. Backend updates frontend with refined Gaussian attributes & trajectories



Gaussian Trajectory Inheritance

- During *Gaussian restoration*, temporal correspondences can be missing because of densification (splitting / cloning)
- To resolve this, we let “child” Gaussians **inherit trajectories** of their “parents”
- Enables restoration of any past frame without loss of density



Motion-Aware Multiview Mapping

- Restored Gaussians share all attributes except for position & orientation with latest Gaussians

  → Temporal consistency of Gaussian appearance

- Similarly to MonoGS, we introduce **isotropic regularization** to prevent overly elongated Gaussians:

$$\mathcal{L}_{\text{iso}} = \sum_{i=1}^{|\mathcal{G}|} \left\| \boldsymbol{s}^i - \bar{\boldsymbol{s}}^i \cdot \mathbf{1} \right\|_1$$

- Overall mapping process can be formulated as:

$$\min_{\substack{\boldsymbol{T}_k, \boldsymbol{p}_k, \mathcal{G}, \\ \forall k \in \mathcal{W}}} \sum_{k \in \mathcal{W}} \mathcal{L}^k_{\text{RGB-D}} + \lambda_{\text{iso}} \mathcal{L}_{\text{iso}}$$

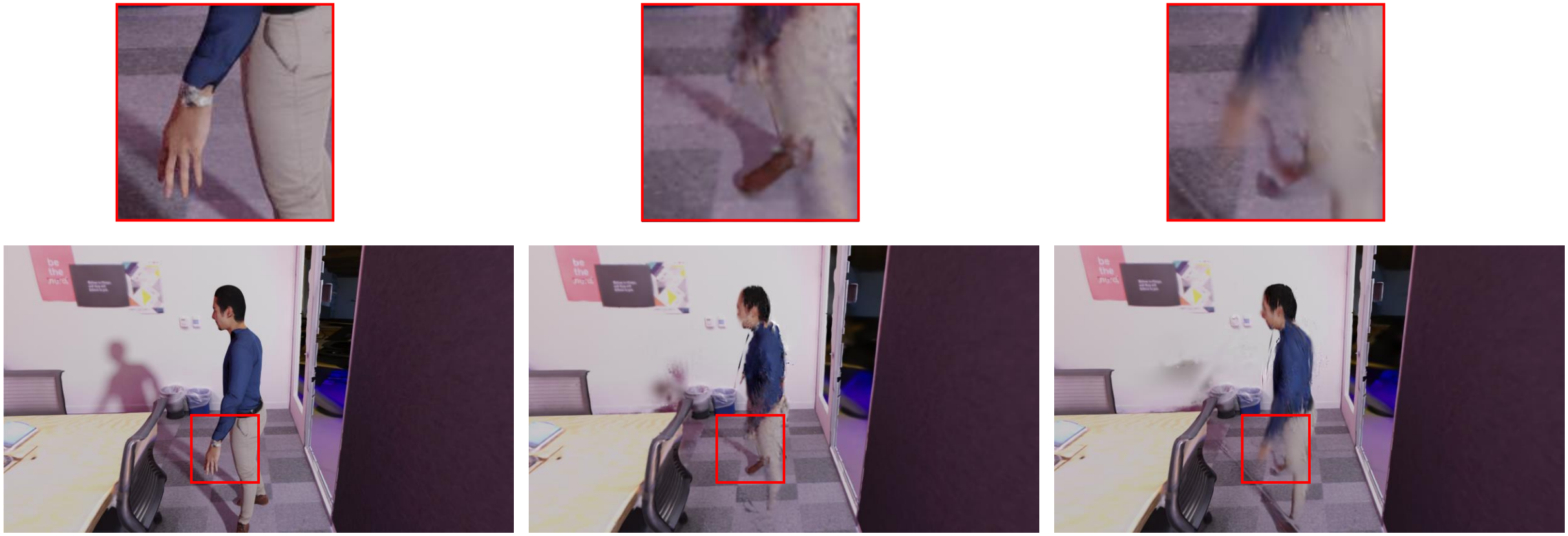
Experiments: Dynamic Replica

- *Dynamic Replica* [Karaev 2023] is a **synthetic** indoor scene dataset with moving humans or animals
- Selected 5 scenes with significant motions
- Used modalities: RGB (left), depth, **optical flow**

(https://dynamic-stereo.github.io/)

Dynamic Replica: Qualitative Evaluation

Our method tracks motion / deformation better



GT | MonoGS (w/ GT camera poses) | **MonoDy-GS (ours)**

Dynamic Replica: Quantitative Evaluation

- Metrics: PSNR, SSIM, LPIPS, evaluated online every fifth frame of training images, averaged across sequence
- MonoDy-GS achieves superior rendering quality over MonoGS for all scenes

Method	Metric	3c30ce	7bf8c1	216ba3	75643c	b907d5	Avg.
MonoGS [5]	PSNR [dB] ↑	19.85	22.95	21.78	21.97	17.06	20.72
	SSIM ↑	0.822	0.858	0.876	0.901	0.728	0.837
	LPIPS ↓	0.268	0.222	0.207	0.181	0.345	0.245
Ours	PSNR [dB] ↑	**22.43**	**27.96**	**23.55**	**24.50**	**19.83**	**23.66**
	SSIM ↑	**0.850**	**0.927**	**0.900**	**0.916**	**0.804**	**0.879**
	LPIPS ↓	**0.222**	**0.114**	**0.148**	**0.155**	**0.269**	**0.182**

Experiments: TUM RGB-D (Dynamic Scenes)

- *TUM RGB-D* [Sturm 2012] is a **real-world** dataset involving motion blur
- Selected 2 sequences with **moving people**: *sitting_halfsphere* & *walking_halfsphere*
- Used modalities: RGB, depth

Example image from *sitting_halfsphere*

TUM RGB-D: Qualitative Evaluation

Our method tracks the hand & head motion better and retains finer details

GT | MonoGS (w/ GT camera poses) | **MonoDy-GS (ours)**

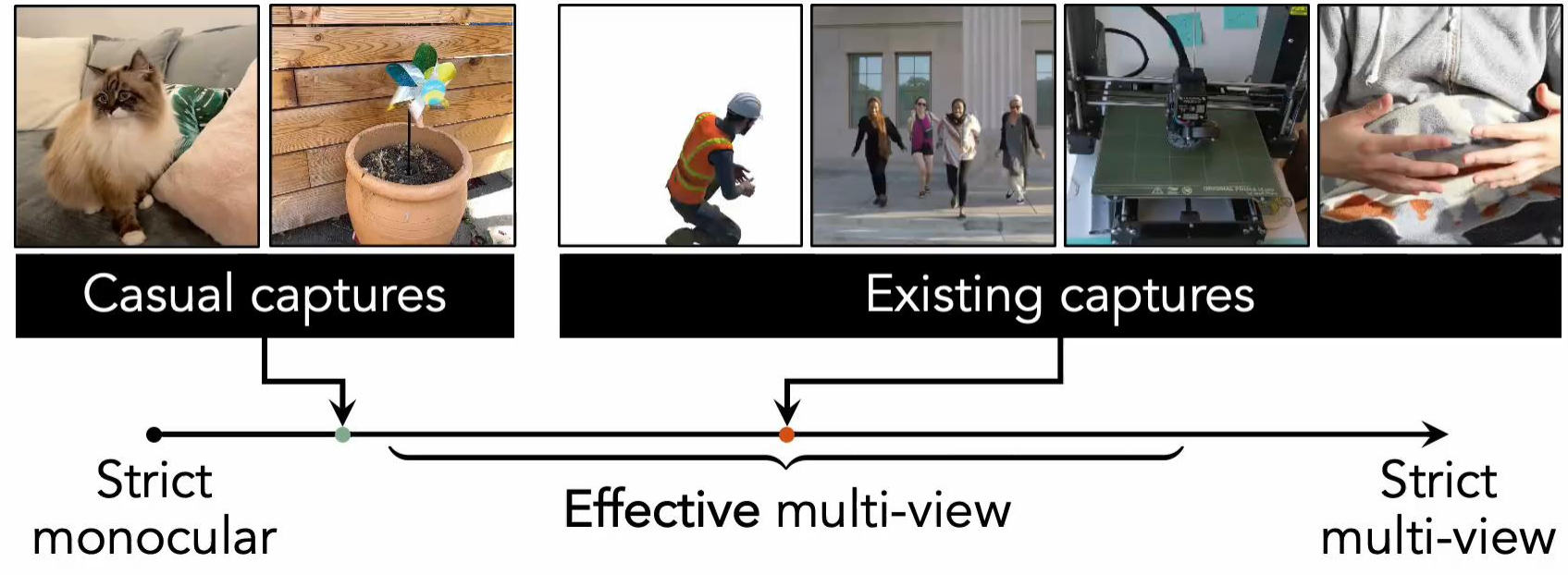
TUM RGB-D : Quantitative Evaluation

- Metrics: PSNR, SSIM, LPIPS, evaluated online every fifth frame of training images, averaged across sequence
- MonoDy-GS achieves superior rendering quality over MonoGS for both scenes

Method	Metric	sitting_halfsphere	walking_halfsphere	Avg.
MonoGS [5]	PSNR [dB] ↑	14.10	12.72	13.41
	SSIM ↑	0.481	0.417	0.449
	LPIPS ↓	0.528	0.558	0.543
Ours	PSNR [dB] ↑	**19.70**	**14.92**	**17.31**
	SSIM ↑	**0.680**	**0.496**	**0.588**
	LPIPS ↓	**0.350**	**0.497**	**0.423**

Experiments: Dycheck iPhone

- *Dycheck iPhone* [Gao 2022] is a **real-world** dataset emulating casual captures using an iPhone
- Challenging: Fast scene motion relative to camera motion ($\bar{v}_{\text{camera}} \approx 0.2\bar{v}_{\text{scene}}$)
  → minimal multiview cues for dynamic parts
- Provides validation videos captured from fixed viewpoints → **Novel View Synthesis** quality evaluation
- Selected 5 scenes with smaller drifts in GT camera poses
- Used modalities: RGB, depth



(https://hangg7.com/dycheck/)

Dycheck iPhone: Qualitative Evaluation (NVS)

Both MonoGS & MonoDy-GS fail to follow the fast motion of the person

GT

MonoGS (w/ GT camera poses)

MonoDy-GS (ours)

Dycheck iPhone: Quantitative Evaluation (NVS)

- **NVS** from fixed viewpoints
- Metrics: mPSNR, mLPIPS (averaged over covisible areas)
- Evaluated only **once per scene** after processing the entire sequence to maximize multiview coverage
- Our method is only almost on par with, or even slightly worse on average than MonoGS
- MonoDy-GS is still **incapable** of capturing **faster scene motion** relative to the camera motion because of the lack of effective multiview cues

Method	Metric	apple	block	paper-windmill	space-out	spin	Avg.
MonoGS [5]	mPSNR [dB] ↑	22.61	**14.91**	**14.63**	14.90	**16.81**	**16.77**
	mLPIPS ↓	0.378	**0.435**	0.350	0.402	**0.277**	**0.368**
Ours	mPSNR [dB] ↑	**23.09**	13.83	14.51	**15.42**	13.81	16.13
	mLPIPS ↓	**0.370**	0.463	**0.345**	**0.385**	0.384	0.389

Limitations: Poor Spatial and Temporal Motion Consistency

- Fast motions often lead to “torn-up” effect
- Gaussian tracking from monocular input severely **underconstrained**
- Potential solution: **Low-parametric** representations

Limitations: Poor Geometry

- Geometry often **overfit** to recent training views
  → Even with perfect motion consistency, rendering quality can still suffer from viewpoint sensitivity
- Potential solution: **Surface normal** priors to better constrain geometry

GT

MonoDy-GS novel-view rendering

Future Work

- Explore **low-parametric** representations of Gaussians to better organize & constrain Gaussian motions
- Adapt to **pure monocular** setup using off-the-shelf depth estimators
- Make use of **surface normals** for better geometry
- Try robust **visual features** such as DINOv2 [Oquab 2023]

References

- Bernhard Kerbl, Georgios Kopanas, Thomas Leimk¨uhler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics, 42(4), July 2023.
- Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In 3DV, 2024.
- Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. arXiv preprint arXiv:2405.17421, 2024.
- Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM.2024.
- Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 20331–20341, 2024.
- Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d gaussian splatting: Towards efficient novel view synthesis for dynamic scenes. arXiv preprint arXiv:2402.03307, 2024.
- Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. arXiv preprint arXiv:2312.00112, 2023.
- Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020.
- Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Dynamic stereo: Consistent dynamic depth from stereo videos. CVPR, 2023.
- J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In Proc. of the International Conference on Intelligent Robot Systems (IROS), Oct.2012.
- Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In NeurIPS, 2022.
- J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In Proc. of the International Conference on Intelligent Robot Systems (IROS), Oct.2012.
- Maxime Oquab, Timoth´ee Darcet, Th´eo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193, 2023.

ETH zürich

Thanks for listening!
Any questions?