58th IEEE/ACM International Symposium on Microarchitecture (MICRO 2025)

ACM SIGARCH

RTGS: Real-Time 3DGS Gaussian Splatting SLAM via Multi-Level Redundancy Reduction

Leshu Li*, Jiayin Qin*, Jie Peng, Zishen Wan, Huaizhi Qu, Ye Han, Pingqing Zheng, Hongsen Zhang, Yu Cao, Tianlong Chen, and Yang (Katie) Zhao

University of Minnesota, Twin Cities
University of North Carolina at Chapel Hill
Georgia Institute of Technology
Vanderbilt University
* means equal contribution

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Georgia Tech

VANDERBILT

An Edge GPU Plug-in for Real-Time 3D Gaussian Splatting-based SLAM

Leshu Li
Advisor: Yang (Katie) Zhao

6/06/2026

UNIVERSITY OF MINNESOTA
Driven to Discover℠

What is SLAM?

SLAM → **S**imultaneous **L**ocalization **A**nd **M**apping

SLAM System [1]

- **Input:** Streaming camera-captured input images
- **Output:** Camera pose trajectory + 3D map

[1] Hidenobu Matsuki, et. al., CVPR'24

SLAM Systems Are Widely Applied

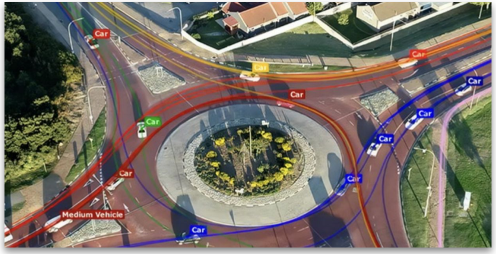
SLAM Systems

Enable

"SLAM enables robots or devices to know where they are and build a map of their surroundings."
— SLAM: Bringing Art to Life through Technology, Meta Engineering Blog (2017)

Applications

Mobile AR Localization [2]

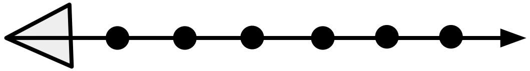


Self-driving City [3]

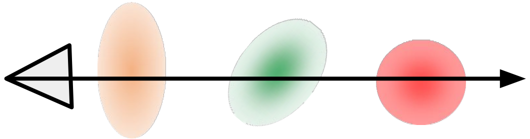
[2] Source:youtube/vG0Q5uGv10o&t; [3] Source:youtube/v=f5eTvbYsLIU

3DGS: Novel 3D Scene Representation

- **3DGS** is the **Novel 3D scene representation**:



NeRF-based

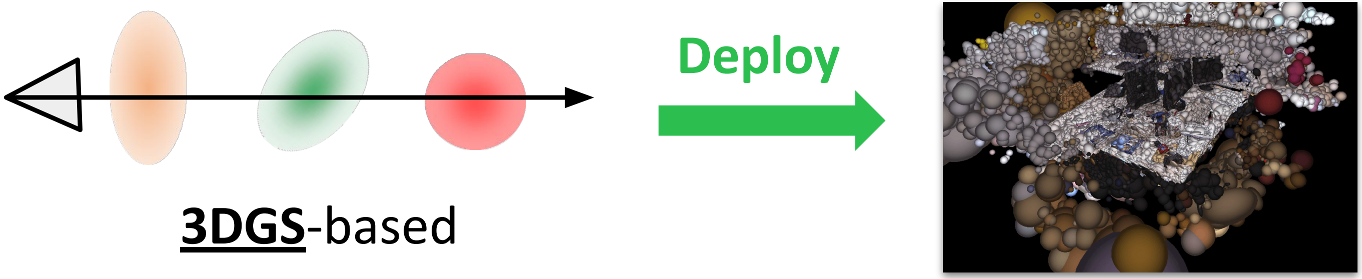


3DGS-based

NeRF-based	3DGS-based
✗ **Low** Rendering Speed	✓ **Fast** Rendering Speed
✗ **Hard** to edit	✓ **Easy** to edit
✓ **High** Fidelity	✓ **High** Fidelity

Apply 3DGS into SLAM: 3DGS-SLAM

- Apply **3DGS** into **SLAM**: **3DGS-SLAM**



[1] Hidenobu Matsuki, et. al., CVPR'24

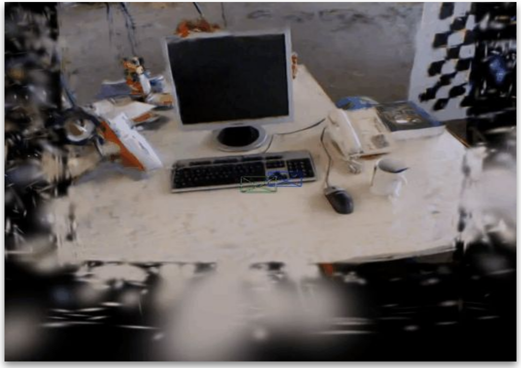
Real-Time 3DGS-SLAM on Edge: Not Yet Possible

Real-World Expectations

Real-Time performance (**> 30 FPS**)

On-Device memory (**< 10 GB**)

Current Limitations



Slow performance (**< 10 FPS**)

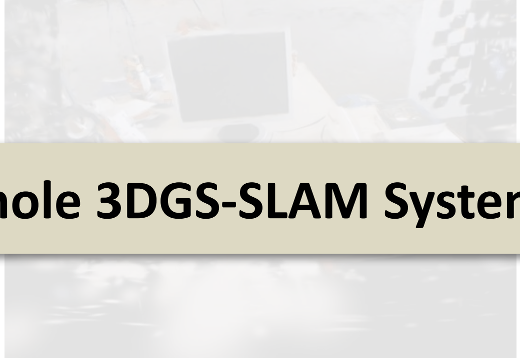
Inefficient storage efficiency (**> 15 GB**)

(Profiling on an edge GPU, Jetson ONX)

Key Insights: Multi-Level Redundancies in 3DGS-SLAM

Real-World Expectations

Current Limitations



Multi-Level Redundancies in the Whole 3DGS-SLAM System

Real-time performance (**> 30 FPS**)

Slow performance

On-device memory (**< 10 GB**)

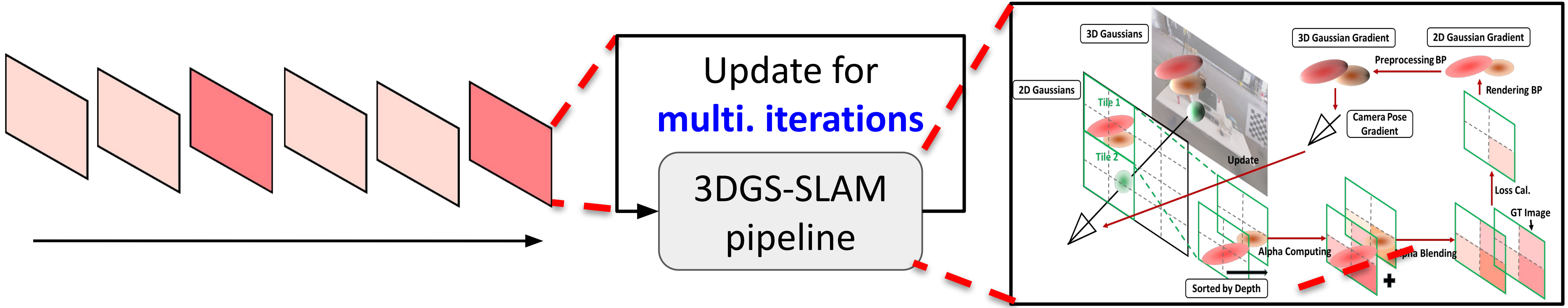
Inefficient storage efficiency

Key Insights: Multi-Level Redundancies in 3DGS-SLAM

Key Insights

Multi-level redundancies in the whole 3DGS-SLAM system

Frame-level Redundancies → Iteration-level Redundancies → Step-level Redundancies



Our Proposed Techniques

Key Insights

Multi-level redundancies in the whole 3DGS-SLAM system

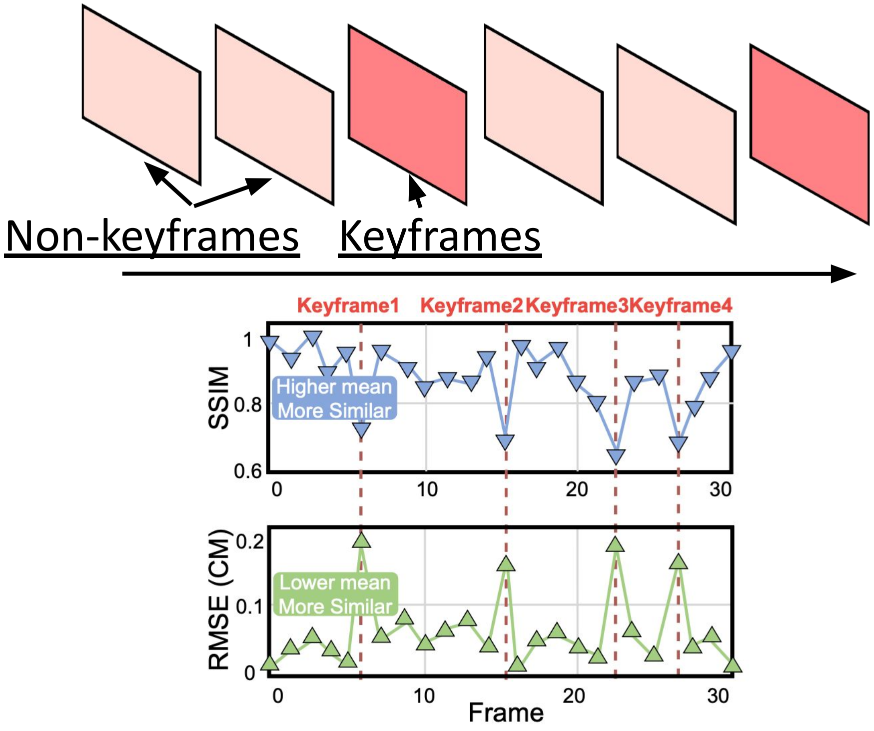
Frame-level Redundancies → Iteration-level Redundancies → Step-level Redundancies

Our Techniques

Leveraging the existing 3DGS-SLAM pipeline to manage the overhead of identifying and reducing the redundancies

Multi-Level Redundancies: Frame-Level Profiling

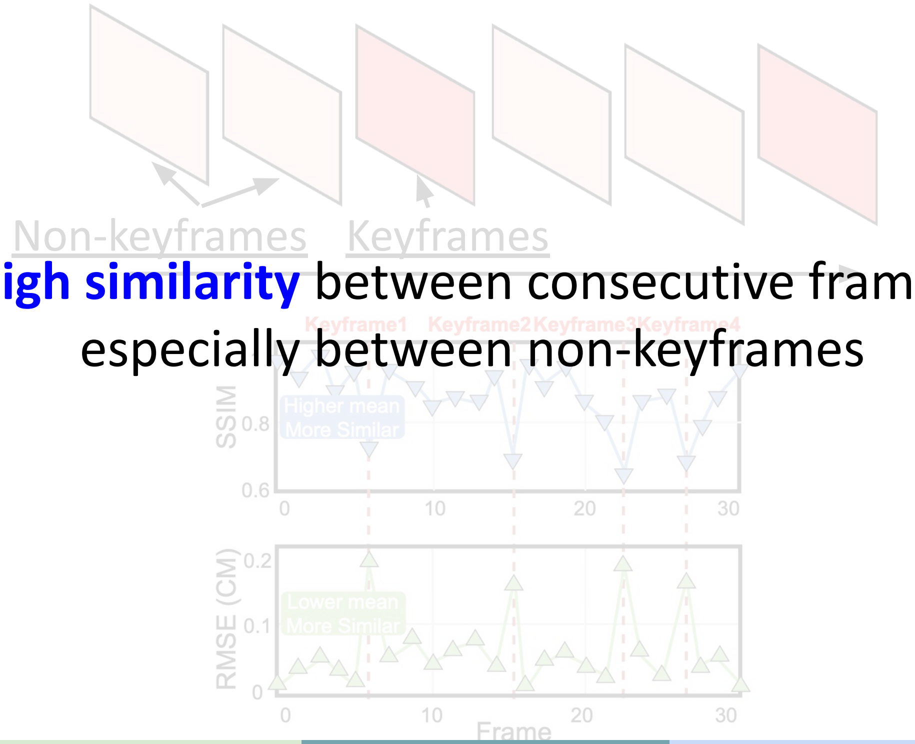
- Frame-level profiling of 3DGS-SLAM to locate bottlenecks:



High similarity between consecutive frames, especially between non-keyframes

Multi-Level Redundancies: Frame-Level Profiling

- Frame-level profiling of 3DGS-SLAM to locate bottlenecks:



High similarity between consecutive frames, especially between non-keyframes

Frame-level Redundancies

☹ Significant amount of **redundant pixel processing**

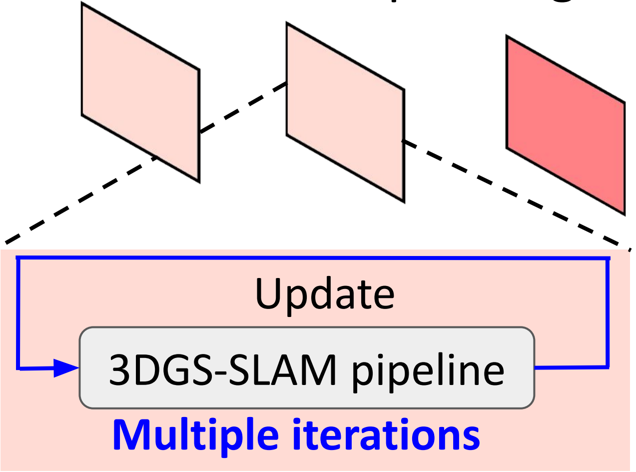
Multi-Level Redundancies: Step-Level Profiling

Frame-level Redundancies

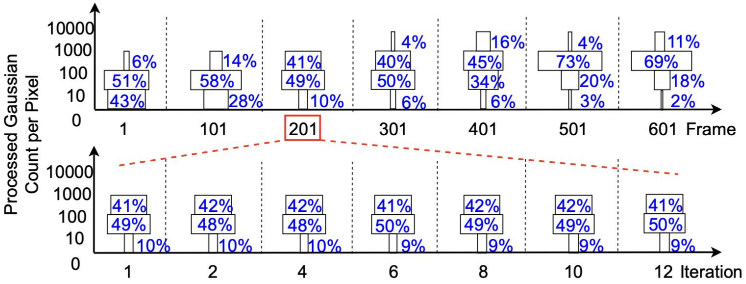
Significant amount of **redundant pixel processing**.

Multi-Level Redundancies: Iteration-Level Profiling

- Iteration-level profiling of 3DGS-SLAM to locate bottlenecks:



Inter-iteration **Similar workload distribution** &
Intra-iteration **workload imbalance**



Multi-Level Redundancies: Iteration-Level Profiling

- Iteration-level profiling of 3DGS-SLAM to locate bottlenecks:

Inter-iteration **Similar workload distribution** & Intra-iteration **workload imbalance**

→ ☹

Iteration-level Redundancies

Intra-iteration **Redundant Workload Assignment** across pixels

Multi-Level Redundancies: Step-Level Profiling

Frame-level Redundancies

Significant amount of **redundant pixel processing**.

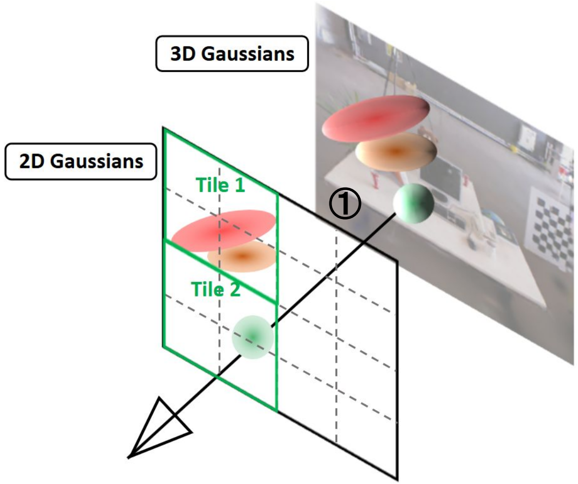
Iteration-level Redundancies

Intra-iteration **Redundant workload assignment** across pixels.

Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

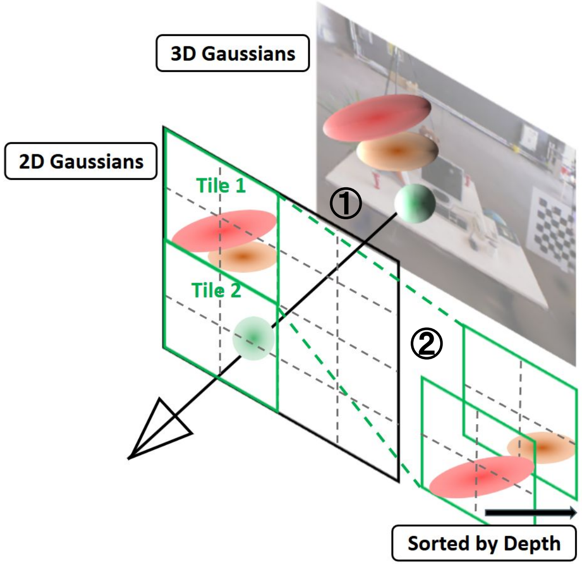
① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP



Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

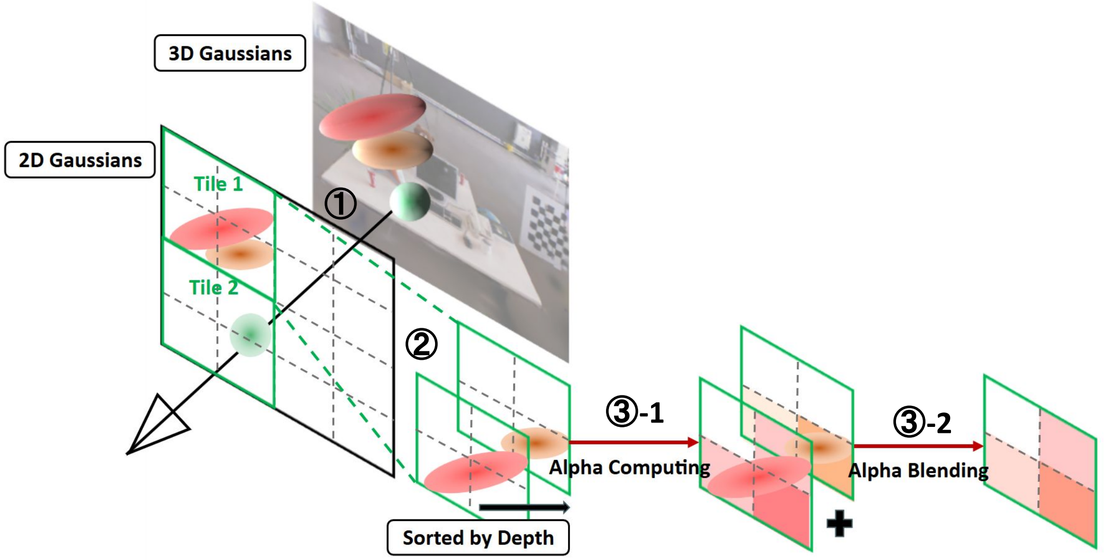
① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP



Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

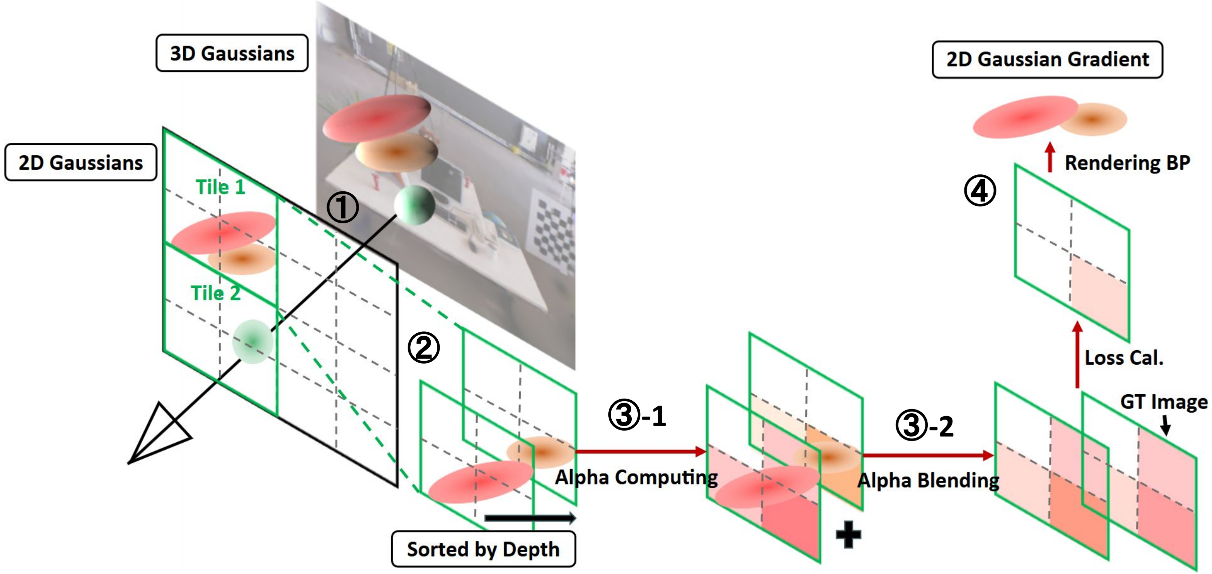
① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP



Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

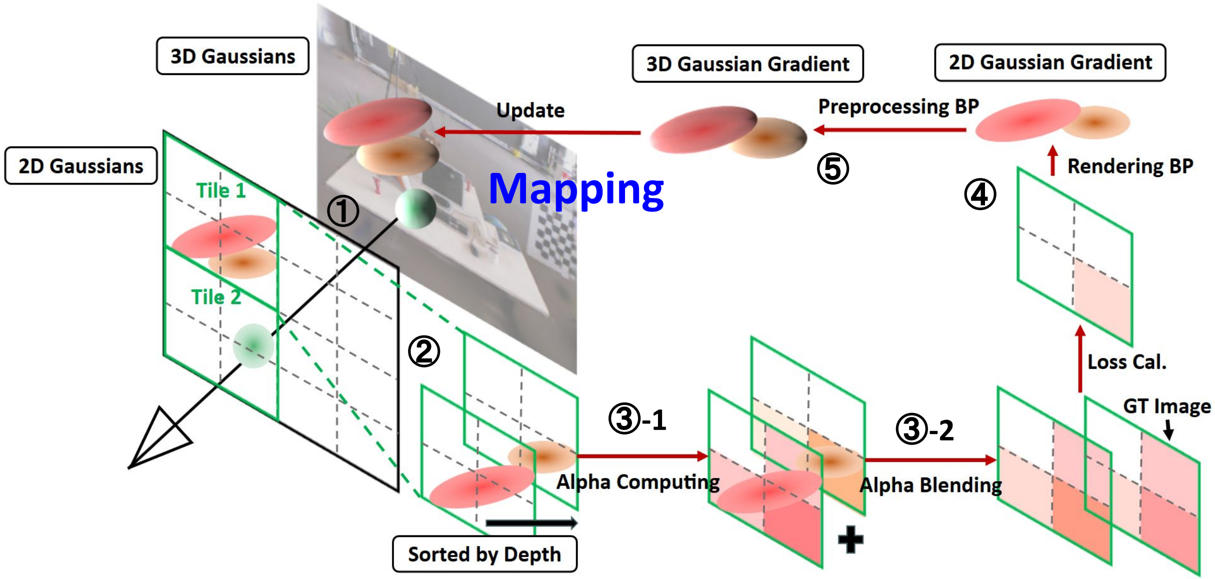
① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP



Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

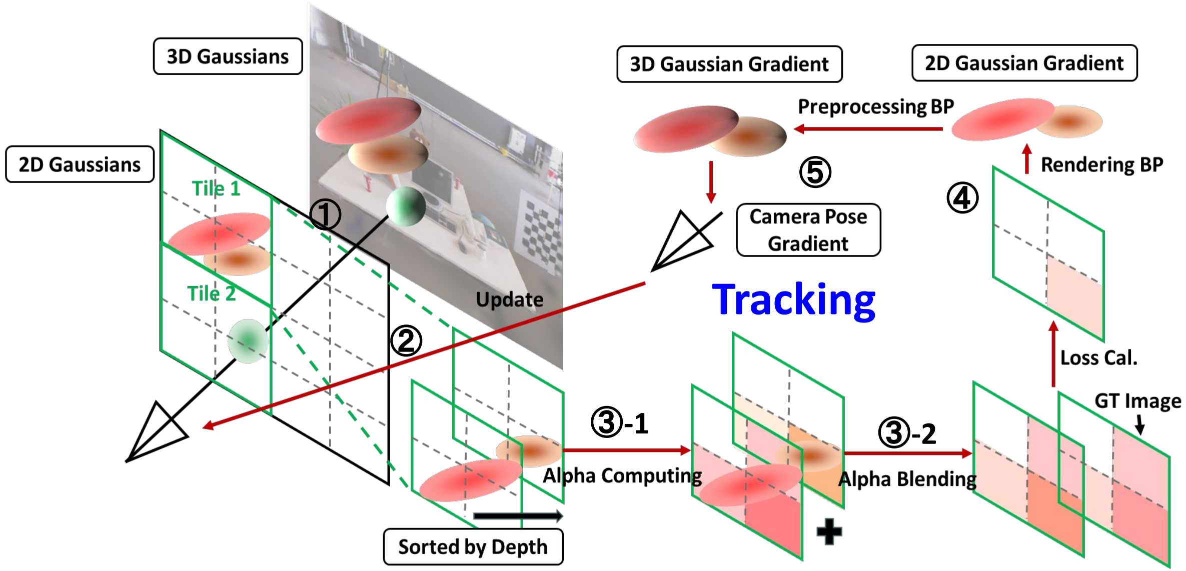
① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP



Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

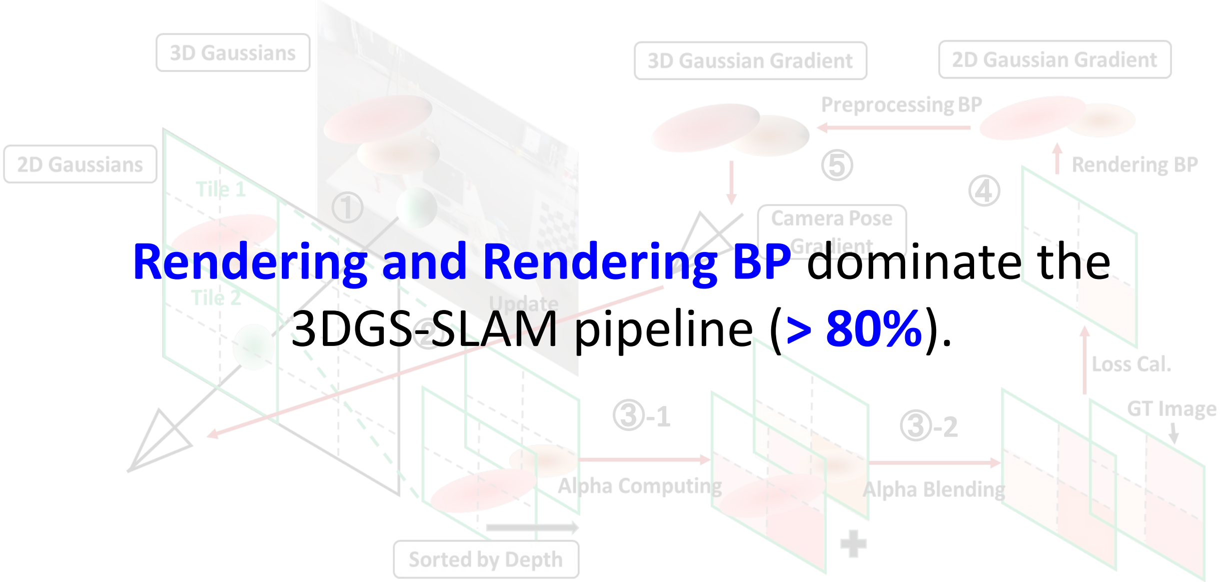
① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP



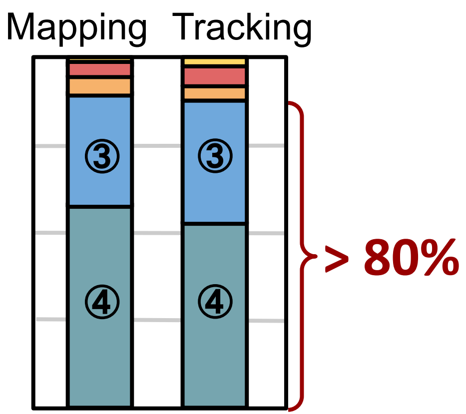
Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP



Rendering and Rendering BP dominate the 3DGS-SLAM pipeline (**> 80%**).



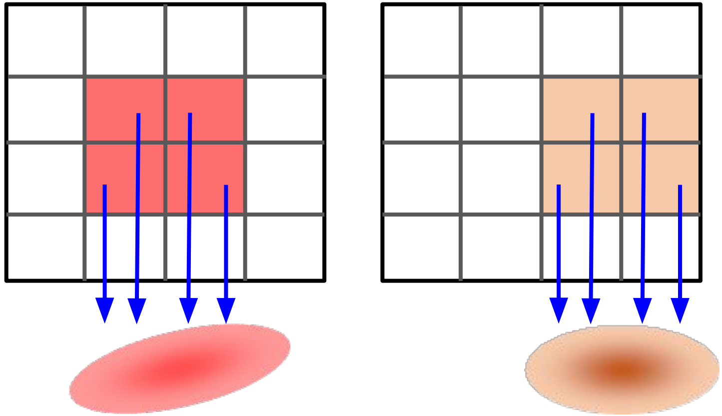
SOTA ONX edge GPU
On GSSLAM

Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP

  - **Rendering and Rendering BP** dominate the 3DGS-SLAM pipeline.



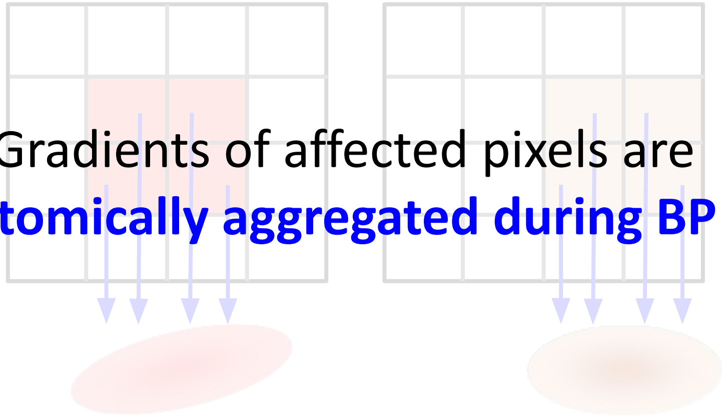
Gradients of affected pixels are **atomically aggregated during BP**

Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP

  - **Rendering and Rendering BP** dominate the 3DGS-SLAM pipeline.



Gradients of affected pixels are **atomically aggregated during BP** → ☹ Massive **time-consuming** atomic add operations

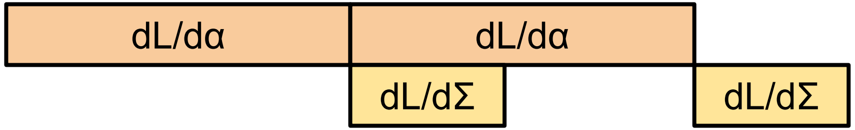
Step-level Redundancies

Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP

  - **Rendering and Rendering BP** dominate the 3DGS-SLAM pipeline.



e.g. Alpha Grad. Comp. (**20 cycles**) v.s. others (**8 cycles**) during Rendering BP

Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP

  - **Rendering and Rendering BP** dominate the 3DGS-SLAM pipeline.

Step-level Redundancies

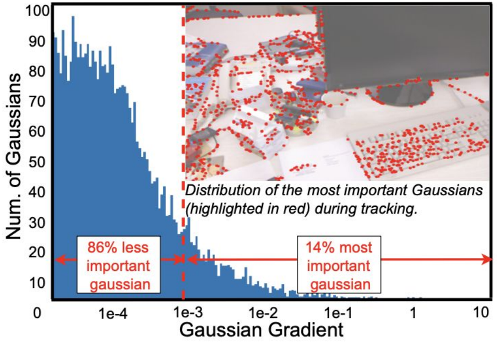
e.g. Alpha Grad. Comp. (**20 cycles**) v.s. others (**8 cycles**) during Rendering BP → **Imbalanced pipelines** during Rendering and Rendering BP

Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP

  - **Rendering and Rendering BP** dominate the 3DGS-SLAM pipeline.



~ 86% **less important Gaussians** in Rendering and Rendering BP

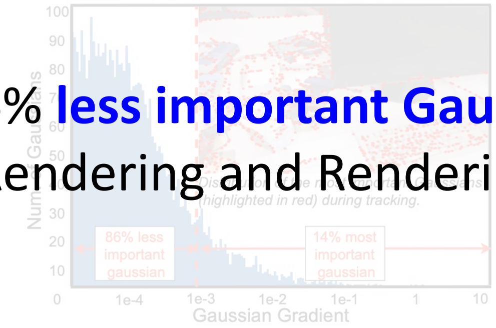
Multi-Level Redundancies: Step-Level Profiling

- Step-level profiling of 3DGS-SLAM to locate bottlenecks:

① Preprocessing → ② Sorting → ③ Rendering → ④ Rendering BP → ⑤ Preprocessing BP

  - **Rendering and Rendering BP** dominate the 3DGS-SLAM pipeline.

Step-level Redundancies



~ 86% **less important Gaussians** in Rendering and Rendering BP → **Massive redundant Gaussians** involved in computing Rendering and Rendering BP.

Multi-Level Redundancies: Step-Level Profiling

Frame-level Redundancies

Significant amount of **redundant pixel processing**.

Iteration-level Redundancies

Intra-iteration **Redundant workload assignment** across pixels.

Step-level Redundancies

Massive **time-consuming** atomic add operations.

Imbalanced pipelines during Rendering and Rendering BP.

Massive redundant Gaussians involved in computing Rendering and Rendering BP.

Our Solution for Multi-Level Redundancies

Frame-level Redundancies — Significant amount of redundant pixel processing.

Iteration-level Redundancies — Intra-iteration Redundant workload assignment across pixels.

Massive time-consuming atomic add operations.

Step-level Redundancies — Imbalanced pipelines during Rendering and Rendering BP.

Massive redundant Gaussians involved in computing Rendering and Rendering BP.

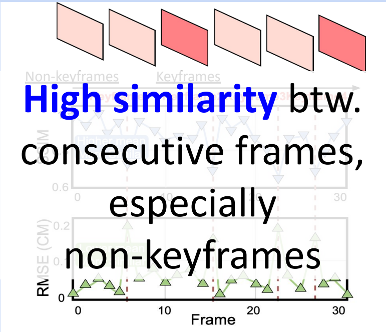
Multi-level redundancies need to be addressed!

Potential Solution: Leveraging the **existing 3DGS-SLAM pipeline** to manage the **overhead** of identifying and reducing the redundancies

RTGS Algorithm: Two Solutions

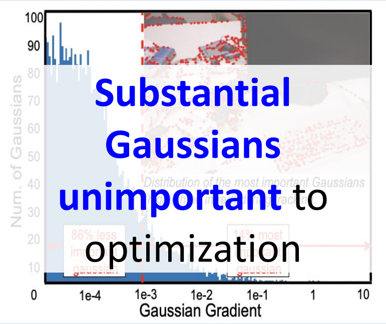
Challenges

Frame-level redundant pixel processing



High similarity btw. consecutive frames, especially non-keyframes

Step-level redundant Gaussian computing



Substantial Gaussians unimportant to optimization

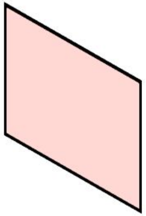
Proposed Techniques

Dynamic Downsampling: dynamic resolution reduction for non-keyframes

Adaptive Pruning: gradient-based importance eval. +mask-prune strategy

RTGS Algorithm

Algorithm: Dynamic Downsampling



Dynamic Downsampling: **reduce resolution** for non-keyframes

- Observation 1: Keyframes are **more important** than non-keyframes

Keyframes: $R_n = R_0$ → **Restoring**

- Observation 2: Consecutive non-keyframes exhibit **high similarity**

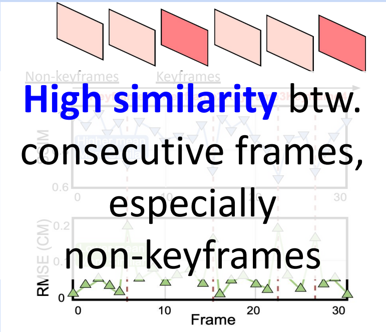
Non-keyframe: $R_n = \min\left((1/16)R_0 \times m^{(n-k-1)},\ (1/4)R_0\right)$

→ **Downsampling**

RTGS Algorithm: Two Solutions

Challenges

Frame-level redundant pixel processing



High similarity btw. consecutive frames, especially non-keyframes

Step-level redundant Gaussian computing

Substantial Gaussians unimportant to optimization

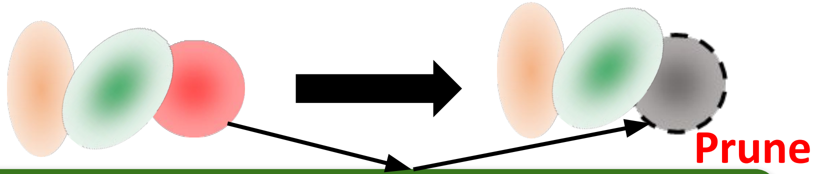
Proposed Techniques

Dynamic Downsampling: dynamic resolution reduction for non-keyframes

Adaptive Pruning: gradient-based importance eval. +mask-prune strategy

RTGS Algorithm

Algorithm: Adaptive Pruning



$$\text{Score}_{\text{gaussian}} = \left\| \frac{d\mathcal{L}}{d\boldsymbol{\mu}} \right\| + \lambda \times \left\| \frac{d\mathcal{L}}{d\Sigma} \right\| < \textbf{Threshold}$$

Adaptive Pruning: reduce redundant Gaussian computations

- To avoid **extra computational overhead**

  pruning based on existing gradients ➡ **Adaptive Pruning**

- To enable **skipping sorting and tile intersection**

  mask every iteration, **prune** K iterations ➡ **Mask-Prune Strategy**

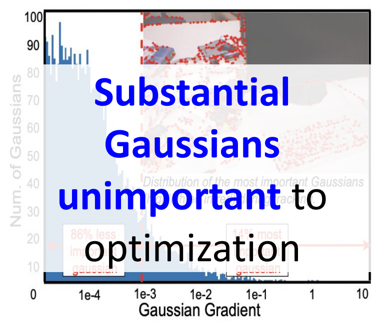
RTGS Algorithm: Two Solutions

Challenges

Frame-level redundant pixel processing

High similarity btw. consecutive frames, especially non-keyframes

Step-level redundant Gaussian computing



Substantial Gaussians unimportant to optimization

Proposed Techniques

Dynamic Downsampling: dynamic resolution reduction for non-keyframes

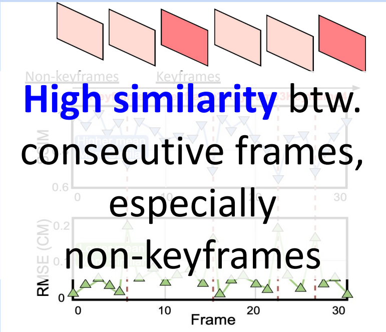
Adaptive Pruning: gradient-based importance eval. +mask-prune strategy

RTGS Algorithm

RTGS Algorithm: Two Solutions

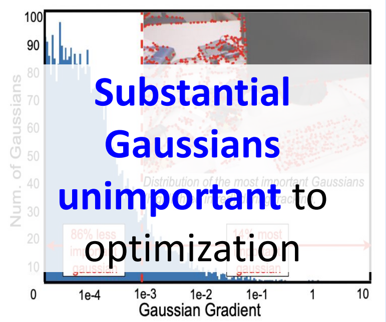
Challenges

Frame-level redundant pixel processing



High similarity btw. consecutive frames, especially non-keyframes

Step-level redundant Gaussian computing



Substantial Gaussians unimportant to optimization

Proposed Techniques

Dynamic Downsampling: dynamic resolution reduction for non-keyframes

Adaptive Pruning: gradient-based importance eval. +mask-prune strategy

RTGS Algorithm

RTGS Hardware: Three Solutions

Challenges

Challenge	Description
Iteration-level redundant workload assignment	Inter-iteration **Similar workload distribution** & Intra-iteration **imbalance**
Step-level imbalance pipeline	Several **imbalanced pipelines**
Step-level time-consuming atomic adds	Massive **memory contentions** during BP

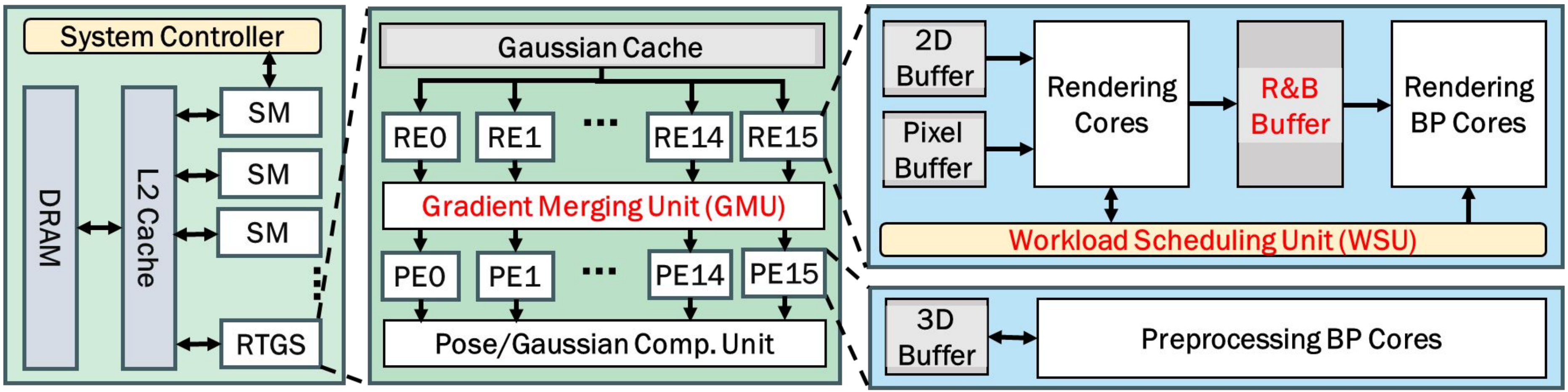
Proposed Techniques

- **Workload Scheduling Unit (WSU):** pixel-level pairwise scheduling
- **Rendering & Backpropagation (R&B) Buffer:** parameter reuse
- **Gradient Merging Unit (GMU):** tree-like merging

RTGS Hardware

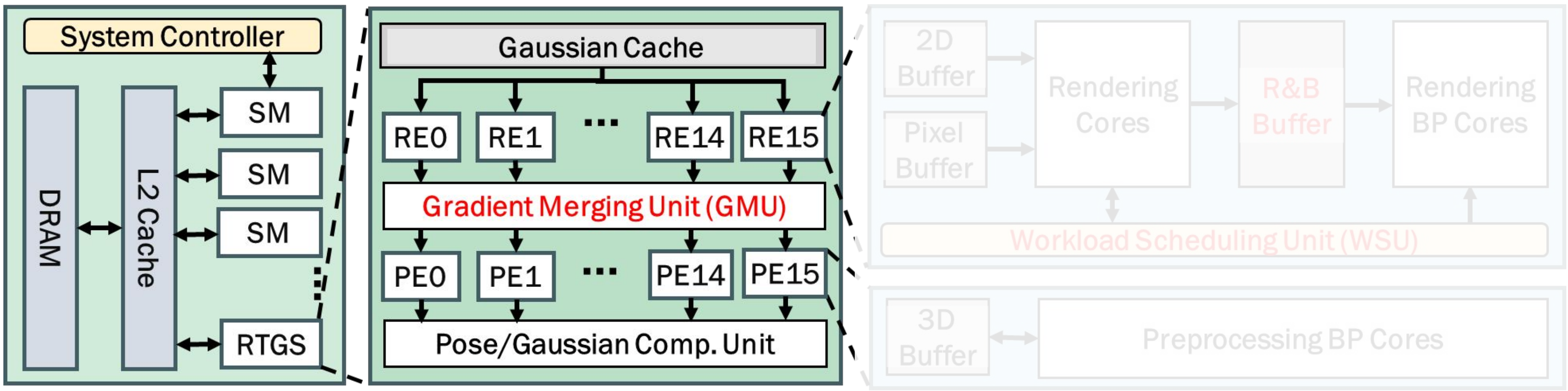
RTGS Hardware: A plug-in Accelerator

- RTGS: A **plug-in accelerator** for Rendering, Rendering BP and Preprocessing BP in 3DGS-SLAM



RTGS Hardware: A Plug-in Accelerator

- RTGS: A **plug-in accelerator**
  - **Compute:** RE for Rendering & Rendering BP; PE for Preprocessing BP
  - **Memory:** Global Gaussian Cache (80 KB)
  - **Merging Unit:** GMU with tree-based merging function

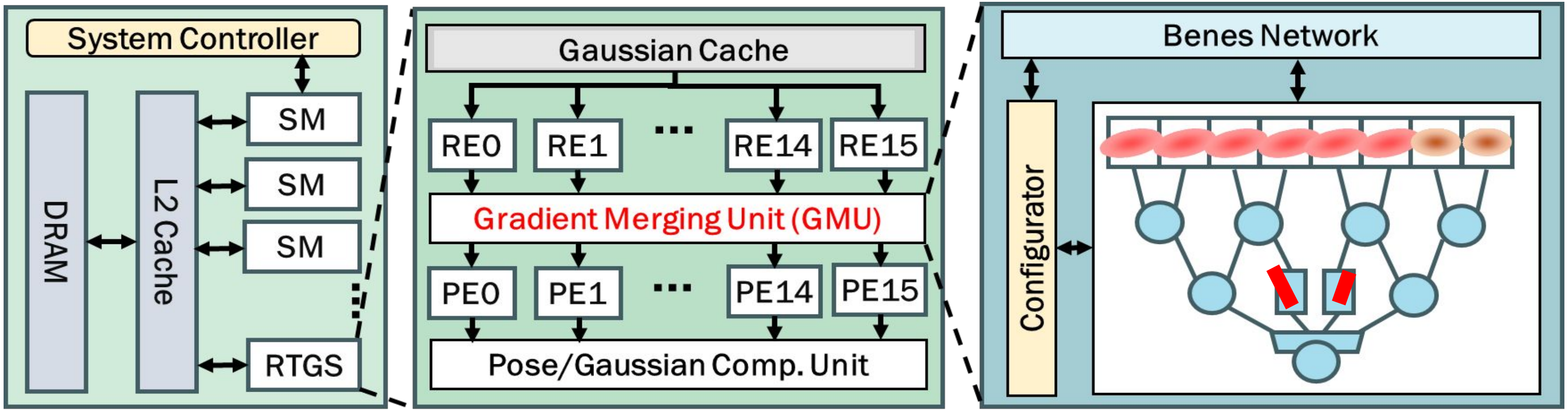


RTGS Hardware: GMU

Merging Unit: GMU with tree-based merging function

Challenge: Traditional adder tree for dense aggregation v.s. sparse aggregation in 3DGS-SLAM

Solution: Use dedicated GMU with bypass links



RTGS Hardware: Three Solutions

Challenges

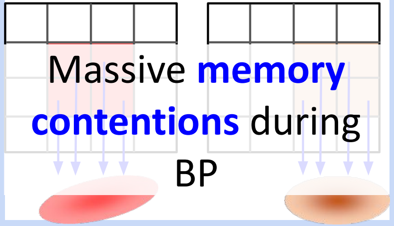
Iteration-level redundant workload assignment

Inter-iteration **Similar workload distribution** & Intra-iteration **imbalance**

Step-level imbalance pipeline

Several **imbalanced pipelines**

Step-level time-consuming atomic adds



Massive **memory contentions** during BP

Proposed Techniques

Workload Scheduling Unit (WSU): pixel-level pairwise scheduling

Rendering & Backpropagation (R&B) Buffer: parameter reuse

Gradient Merging Unit (GMU): tree-like merging

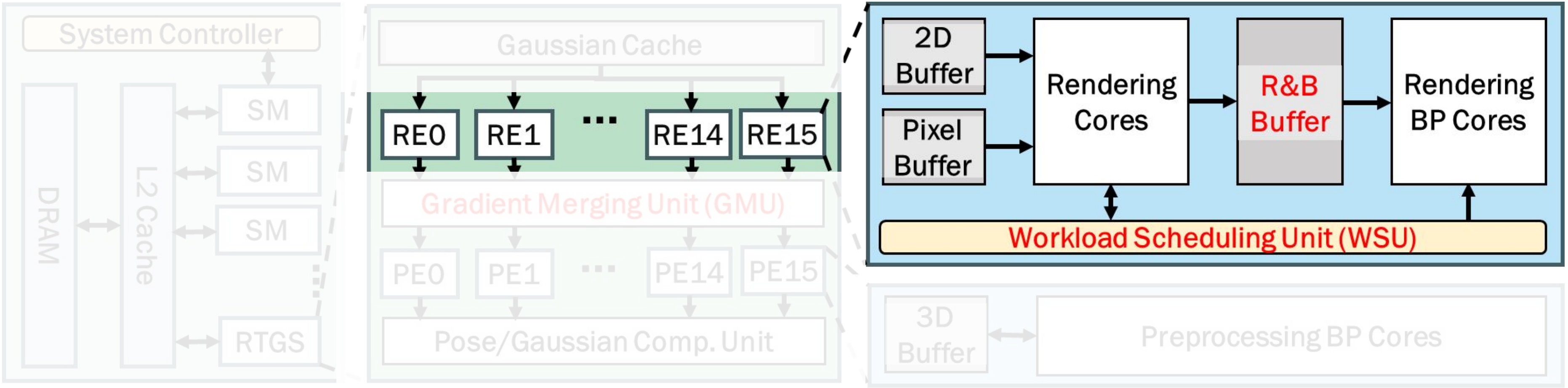
RTGS Hardware

RTGS Hardware: RE in RTGS Accelerator

Rendering and Rendering BP dominate the 3DGS-SLAM pipeline

➡ Contribution mainly in REs

- **Scheduler:** WSU for pairwise workload scheduling
- **Memory:** R&B Buffer for parameter reuse

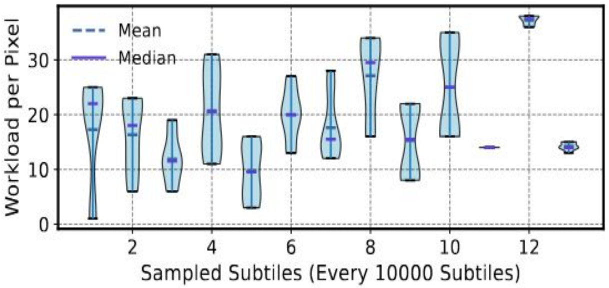


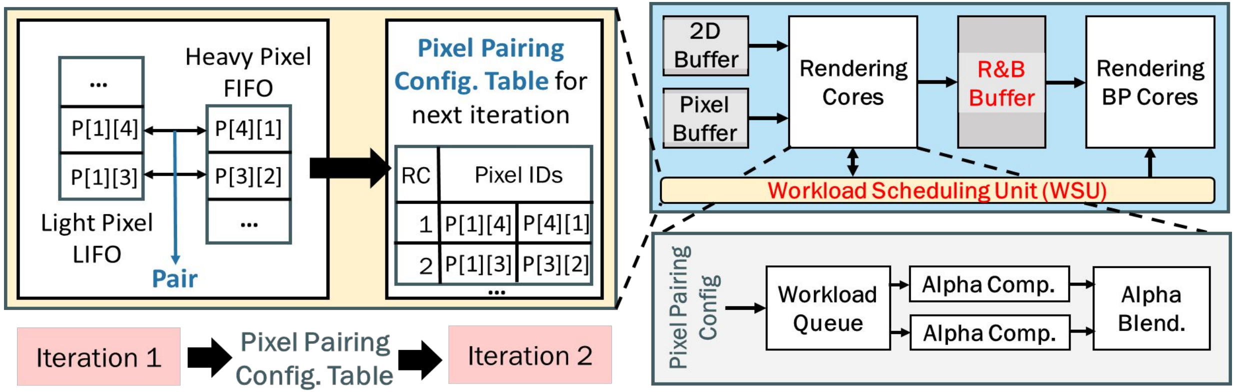
RTGS Hardware: WSU

Scheduler: WSU for pairwise workload scheduling btw. light & heavy pixels (highly symmetrically)

Challenge: **Rescheduling** during each iteration ➡ **Scheduling overhead**

Solution: Pixel Pairing Config. Table in WSU for **inter-iteration scheduling information reuse**





RTGS Hardware: Three Solutions

Challenges

Iteration-level redundant workload assignment

Inter-iteration **Similar workload distribution** & Intra-iteration **imbalance**

Step-level imbalance pipeline

Several **imbalanced pipelines**

Step-level time-consuming atomic adds

Massive **memory contentions** during BP

Proposed Techniques

Workload Scheduling Unit (WSU): pixel-level pairwise scheduling

Rendering & Backpropagation (R&B) Buffer: parameter reuse

Gradient Merging Unit (GMU): tree-like merging

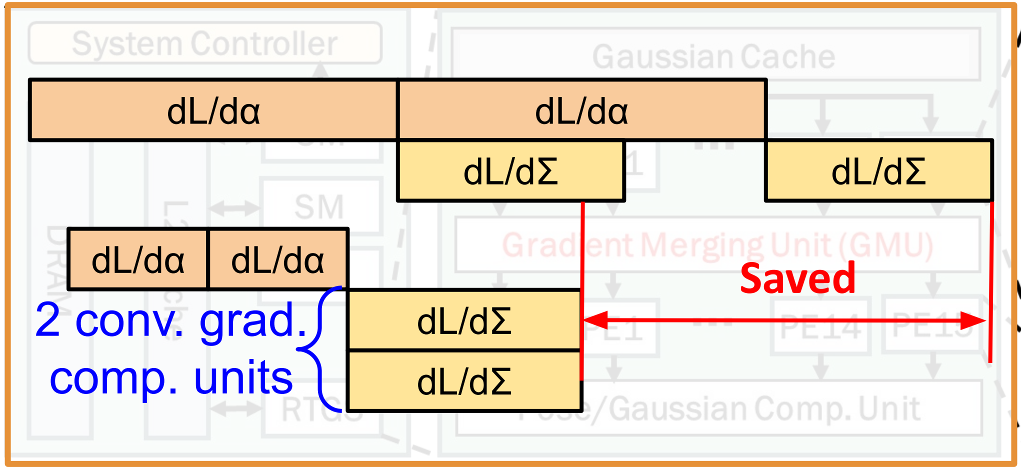
RTGS Hardware

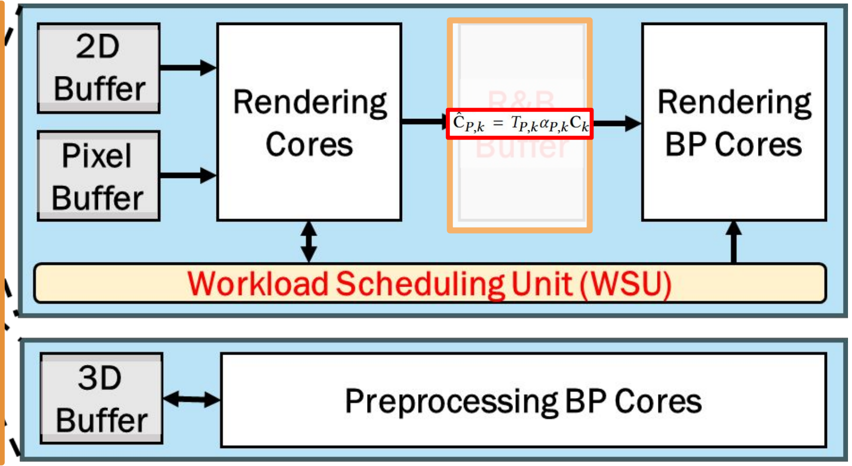
RTGS Hardware: R&B Buffer

- **Memory:** R&B Buffer for parameter reuse

Challenge: **Simply reuse** $\hat{C}_{P,k} = T_{P,k}\alpha_{P,k}C_k$ from R&B Buffer during Rendering BP still cause **pipeline imbalance** (20 : 8 cycles -> 4 : 8 cycles).

Solution: **Reuse + reallocate resources** to achieve a balanced pipeline.





RTGS Hardware: Three Solutions

Challenges

Iteration-level redundant workload assignment

Inter-iteration **Similar workload distribution** & Intra-iteration **imbalance**

Step-level imbalance pipeline

Several **imbalanced pipelines**

Step-level time-consuming atomic adds

Massive **memory contentions** during BP

Proposed Techniques

Workload Scheduling Unit (WSU): pixel-level pairwise scheduling

Rendering & Backpropagation (R&B) Buffer: parameter reuse

Gradient Merging Unit (GMU): tree-like merging

RTGS Hardware

RTGS Hardware: Three Solutions

Challenges

Iteration-level redundant workload assignment

Inter-iteration **Similar workload distribution** & Intra-iteration **imbalance**

Step-level imbalance pipeline

Several **imbalanced pipelines**

Step-level time-consuming atomic adds

Massive **memory contentions** during BP

Proposed Techniques

Workload Scheduling Unit (WSU): pixel-level pairwise scheduling

Rendering & Backpropagation (R&B) Buffer: parameter reuse

Gradient Merging Unit (GMU): tree-like merging

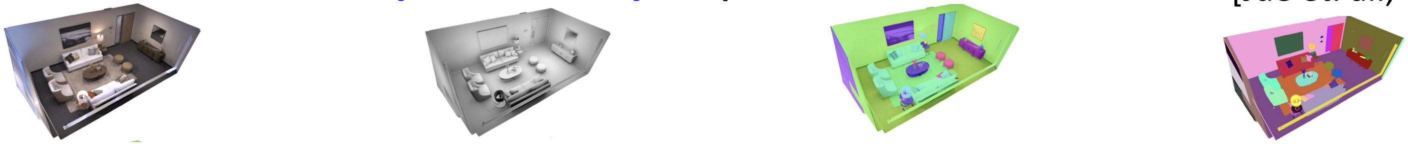
RTGS Hardware

Evaluation Setup

- Consider **18 scenes** from 4 datasets with different resolution

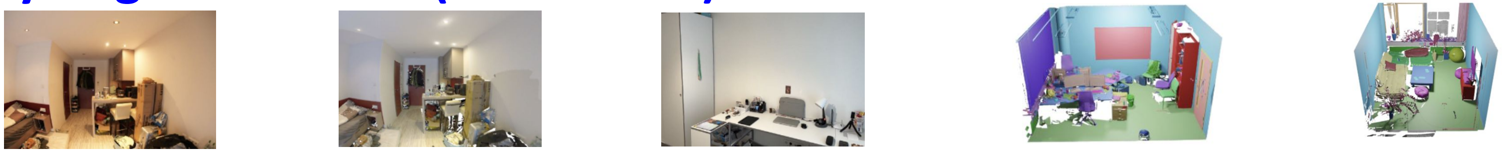
- **Small resolution (480x640)** TUM dataset [Jurgen et. al., IROS'12]

- **Medium resolution (680x1200)** Replica dataset [Jue et. al., CVPR'19]



- **Large resolution (968x1296)** ScanNet dataset [Dai et. al., CVPR'17]

- **Very large resolution (1160x1752)** ScanNet++ dataset [Yesh et. al., ICCV'23]



Evaluation Setup

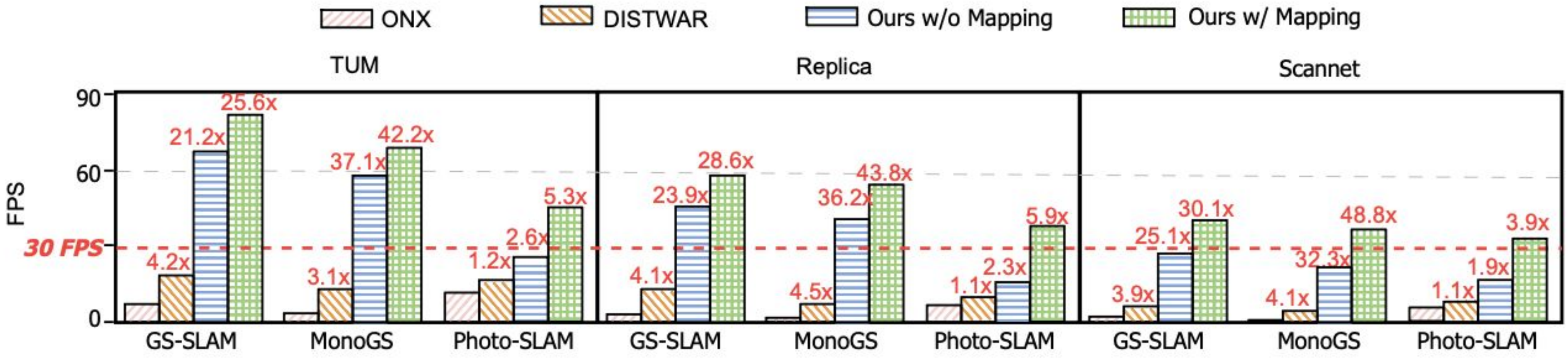
- Benchmark w/ **4 baselines** with various computing resources
  - Jetson ONX GPU
  - RTX 3090 GPU
  - GauSPU
  - RTGS

Device	Jetson ONX	RTX 3090	GauSPU*	**RTGS (Ours)**
SRAM	4 MB	80.25 MB	560 KB	197 KB
Area	450 mm^2	628	30 mm^2	28.41 mm^2
Power	15 W	352 W	9.4 W	8.11 W
Technology	8 nm	8 nm	12 nm	28 nm

*GauSPU: A GPU plug-in for accelerating 3DGS-SLAM through sparse tile sampling [Wu et. al., MICRO'24]

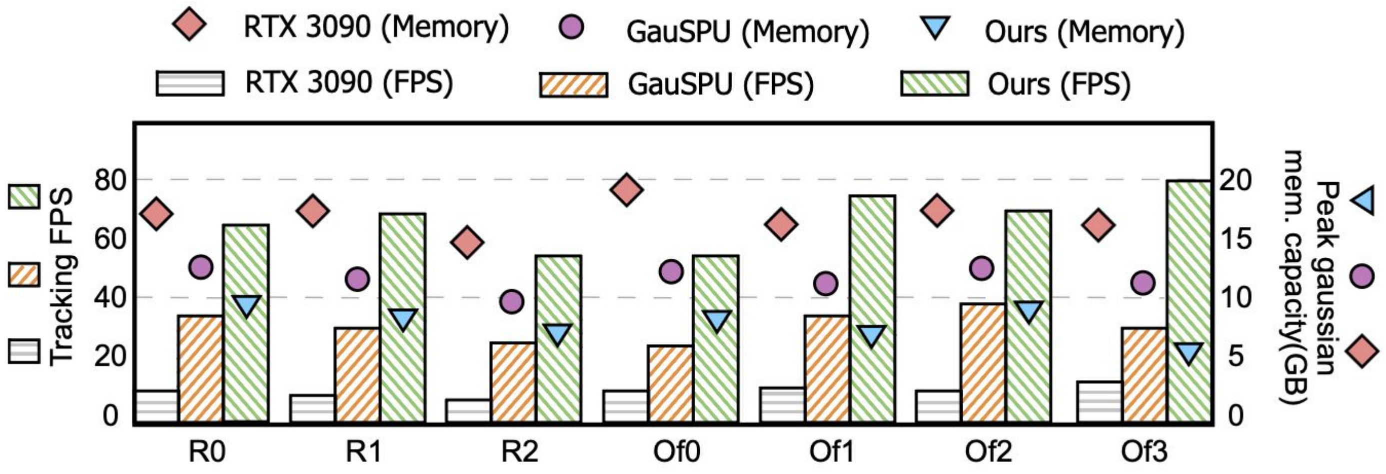
Evaluation: Real-Time 3DGS-SLAM Across Various Scenes

- The FPS is improved by up to **48.8×** compared to the baseline.
- **Real-time performance** (> 30 FPS) is achieved across all datasets.



Evaluation: On-Device Memory Requirement

- The peak memory usage was reduced by **2.3×**, bringing it **below 10 GB** while achieving state-of-the-art performance.



58th IEEE/ACM International Symposium on Microarchitecture (MICRO 2025)

RTGS: Real-Time 3DGS Gaussian Splatting SLAM via Multi-Level Redundancy Reduction

Leshu Li*, Jiayin Qin*, Jie Peng, Zishen Wan, Huaizhi Qu, Ye Han, Pingqing Zheng, Hongsen Zhang, Yu Cao, Tianlong Chen, and Yang (Katie) Zhao

University of Minnesota, Twin Cities
University of North Carolina at Chapel Hill
Georgia Institute of Technology
Vanderbilt University