Compute Where It Matters

TurboGS : Accelerating 3D Gaussian Splatting via Error-Guided Sparse Pixel Sampling and Optimization

Zheng Dong†[1], Daifei Qiu[2], Pinxuan Dai[3], Ke Xu[4], Jiamin Xu[5], Lili He[1], Rynson W. H. Lau[4], Weiwei Xu†[3]

1 ZHEJIANG SCI-TECH UNIVERSITY 浙江理工大学 1897

2 MALANSHAN AUDIO&VIDEO LABORATORY 马栏山音视频实验室

3 浙江大学 ZHEJIANG UNIVERSITY 1897

4 CityU

5 HANGZHOU DIANZI UNIVERSITY 杭州电子科技大学 1956

ICML International Conference On Machine Learning

Background : Why Do We Need Fast 3DGS Optimization ?

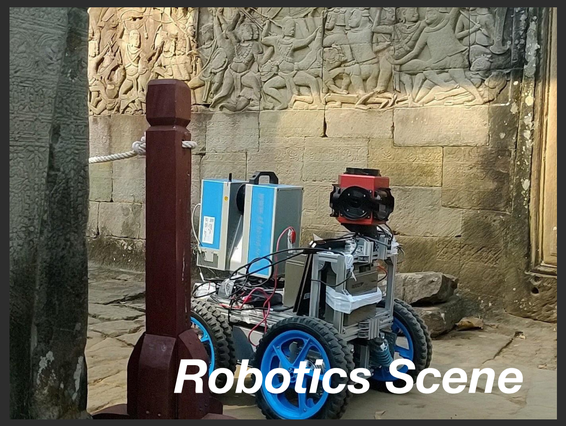
Rome Images

3DGS Optimization

3D Gaussian Scene

Ours : ~ 92s

> 12x Faster Training than Vanilla 3DGS



For Frequently Updated Scenes, Optimization Latency Becomes the Bottleneck

Baseline 3DGS : Inefficient Computation Allocation

Gradient-Dominated (No efficient density control)

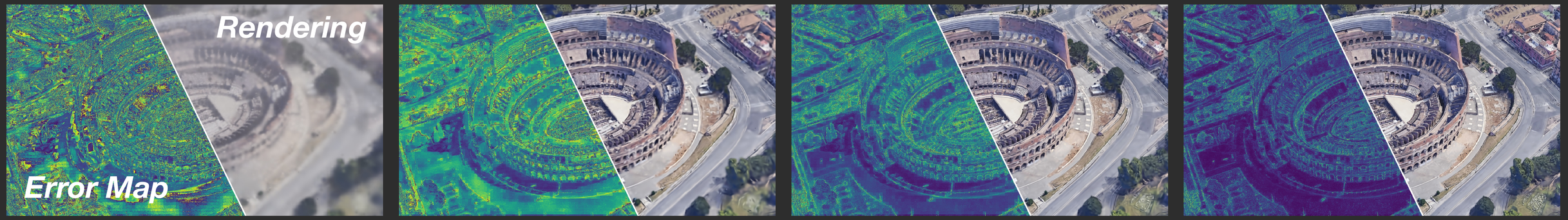
3DGS : All Pixels Treated Equally

1.0 seond

10.0 seonds

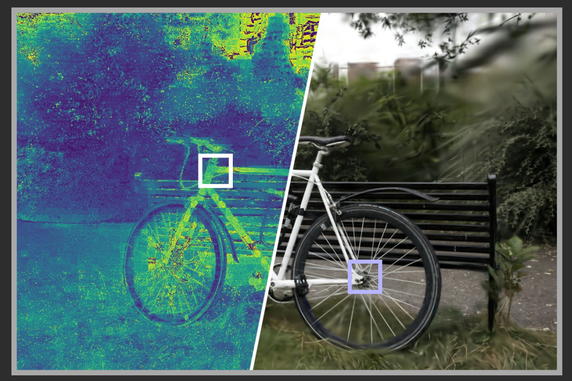
20.0 seonds

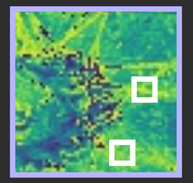
40.0 seonds ...



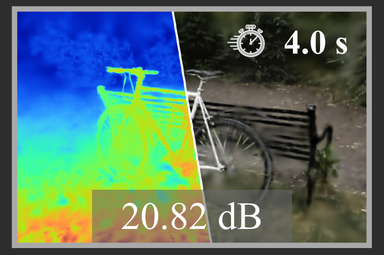
Most Pixels are Rapidly Well Reconstructed, yet They Still Consume Large Computation

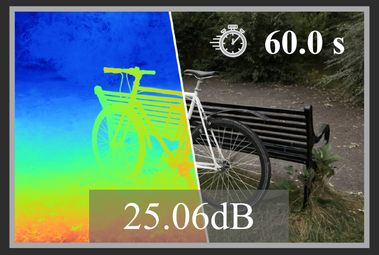
Motivation : Compute Where It Matters

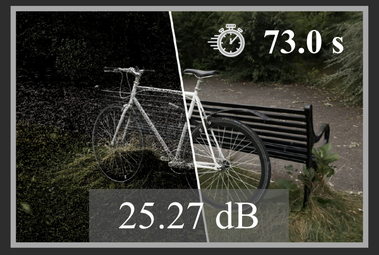




Idx. 0
Idx. 1
Idx. 2
⋮
Idx. N

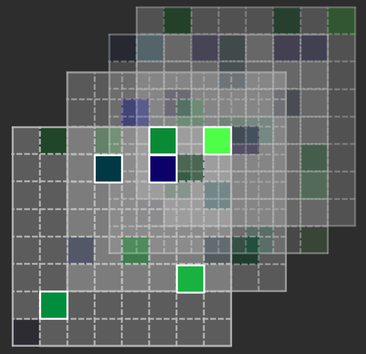




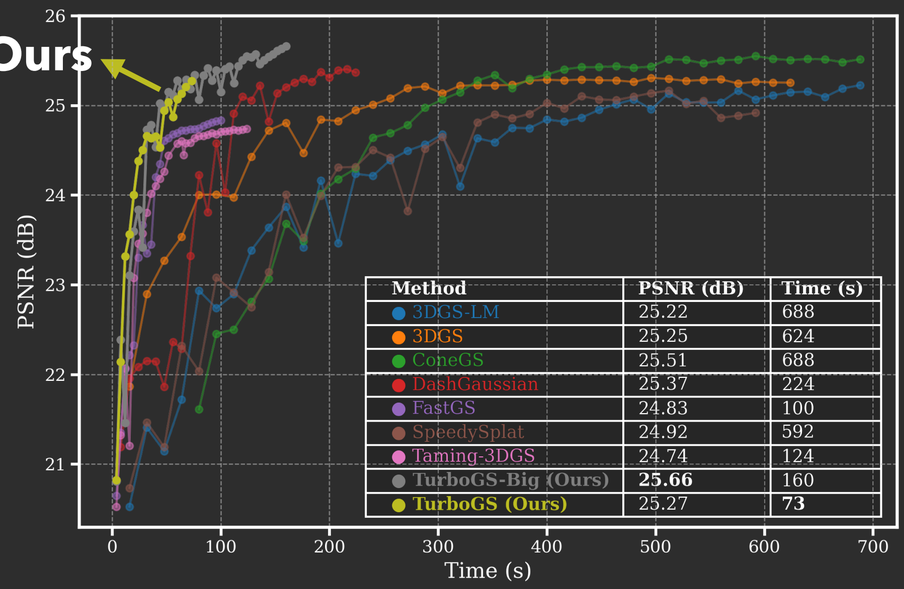


Error-Driven
Density Control

Error-Guided
Pixel Sampling



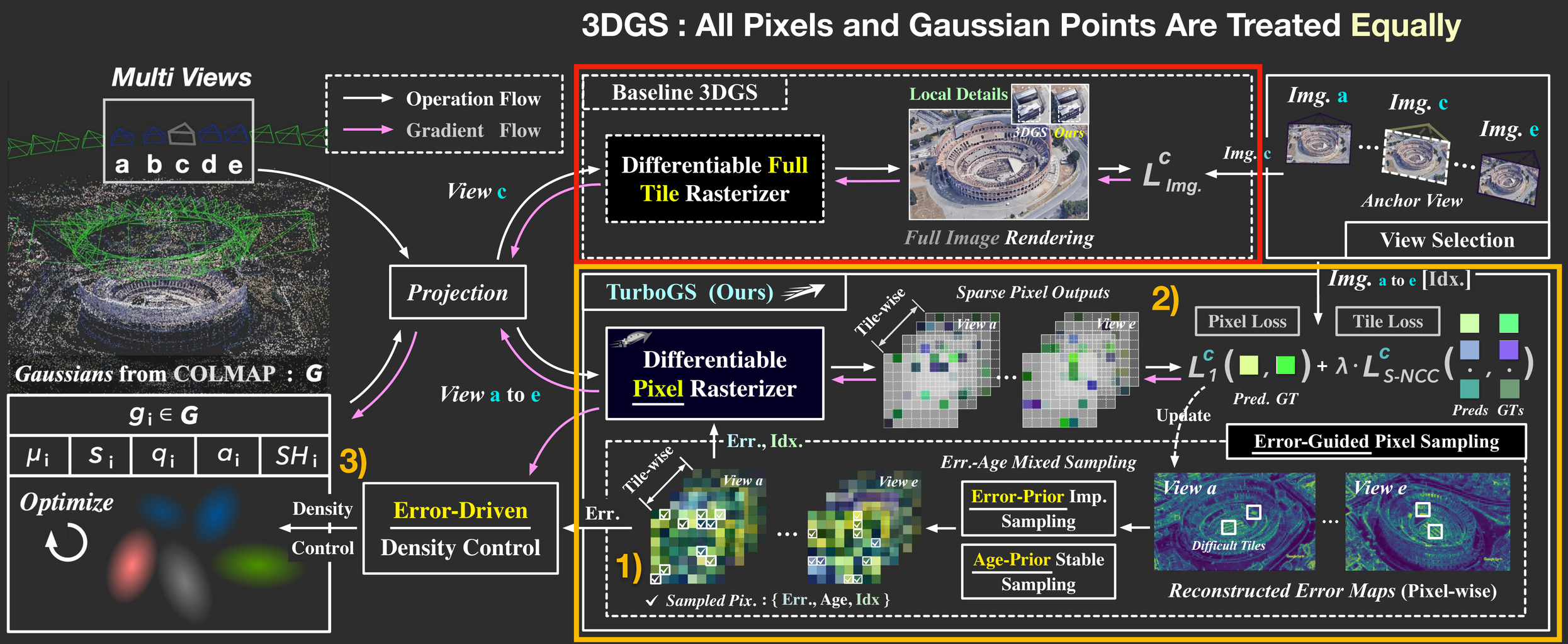
GT Pixels



Method	PSNR (dB)	Time (s)
3DGS-LM	25.22	688
3DGS	25.25	624
ConeGS	25.51	688
DashGaussian	25.37	224
FastGS	24.83	100
SpeedySplat	24.92	592
Taming-3DGS	24.74	124
TurboGS-Big (Ours)	**25.66**	160
TurboGS (Ours)	25.27	**73**

PSNR - TIME **on Bicycle Data**

TurboGS : Framework Overview

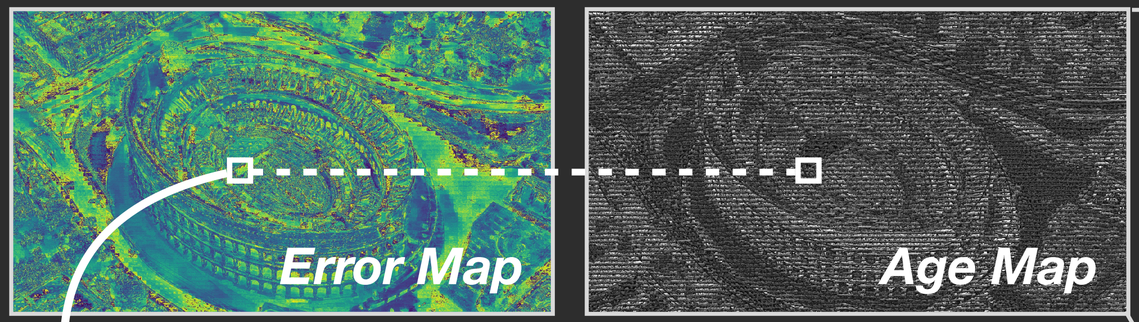


TurboGS Focuses on 1) *Where to Optimize ?* 2) *How to Optimize ?* 3) *What to Optimize ?*

Adaptive Error-Guided Sparse Pixel Sampling

Persistent Informative Maps

5.0 s



Determine Where to Optimize ?

Pixel Loss as Error Signal : $e_{t,v}(\mathbf{p}) = \|\hat{\mathbf{I}}_{t,v}(\mathbf{p}) - \mathbf{I}_v(\mathbf{p})\|_1$

EMA Update of Pixel Errors :

$$\mathbf{E}_v(\mathbf{p}) \leftarrow (1-\beta)\,\mathbf{E}_v(\mathbf{p}) + \beta\, e_{t,v}(\mathbf{p})$$

Pixel Age Update : $\mathbf{A}_v(\mathbf{p}) \leftarrow \begin{cases} 0, & \mathbf{p} \text{ sampled,} \\ \mathbf{A}_v(\mathbf{p}) + 1, & \text{otherwise.} \end{cases}$

Error-Prior Importance Sampling :

$$\pi_E(\mathbf{p} \mid \tau, v) = \frac{\mathbf{E}_v(\mathbf{p})}{\sum_{\mathbf{q}\in\tau} \mathbf{E}_v(\mathbf{q}) + \epsilon}, \quad \mathbf{p} \in \tau,$$

Age-Prior Stable Sampling :

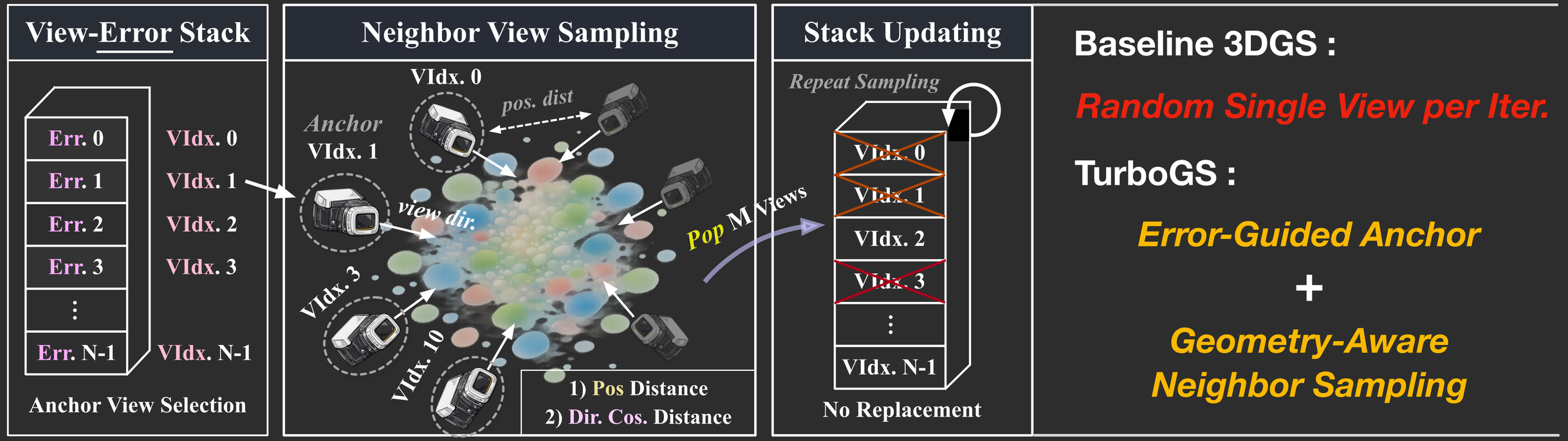
$$\mathcal{S}_{t,v}(\tau) = \mathrm{TopK}_{\mathbf{p}\in\tau\setminus\mathcal{H}_{t,v}(\tau)}\big(\mathbf{A}_v(\mathbf{p}), N_{\text{stable}}\big),$$

Full Sampled Set : $\mathcal{P}_{t,v}(\tau) = \mathcal{H}_{t,v}(\tau) \cup \mathcal{S}_{t,v}(\tau)$

Focus on Informative Pixels with Higher Recon. Error

Adaptive Geometry-Aware View Sampling

Where to Optimize Globally ? **Multi-View Supervision with Local Geometric Perception**



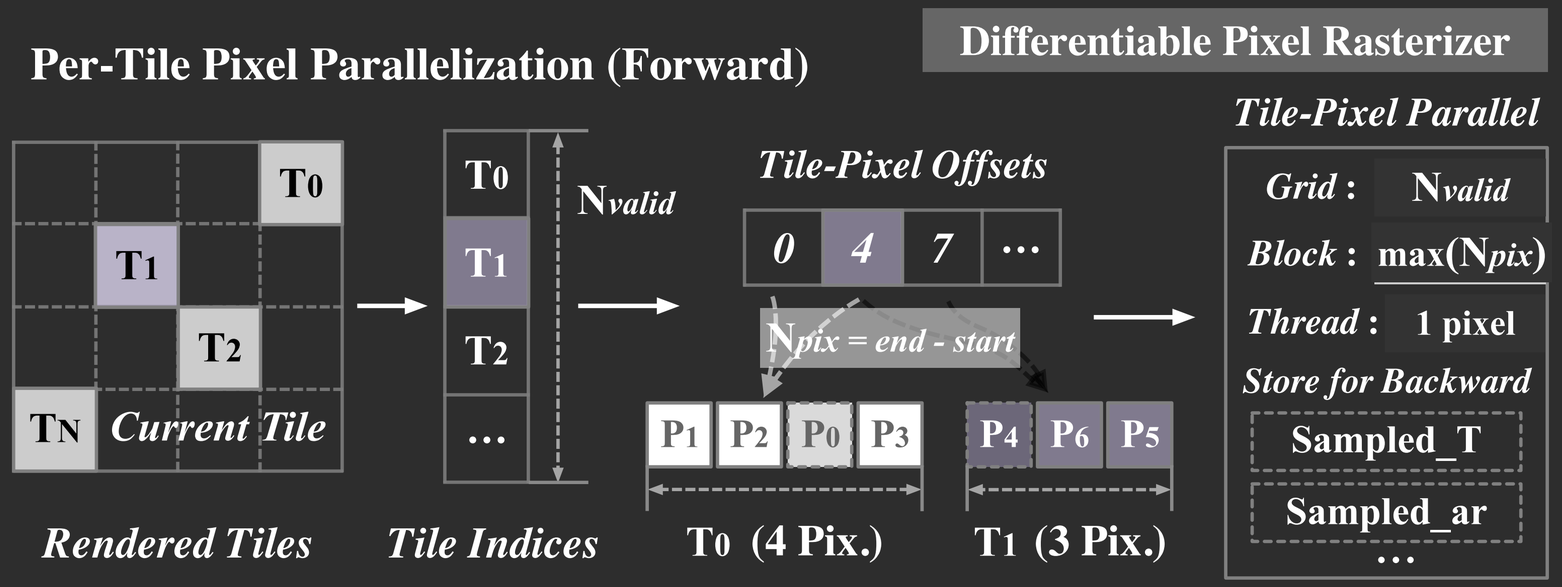
Error Map Information Statistics :

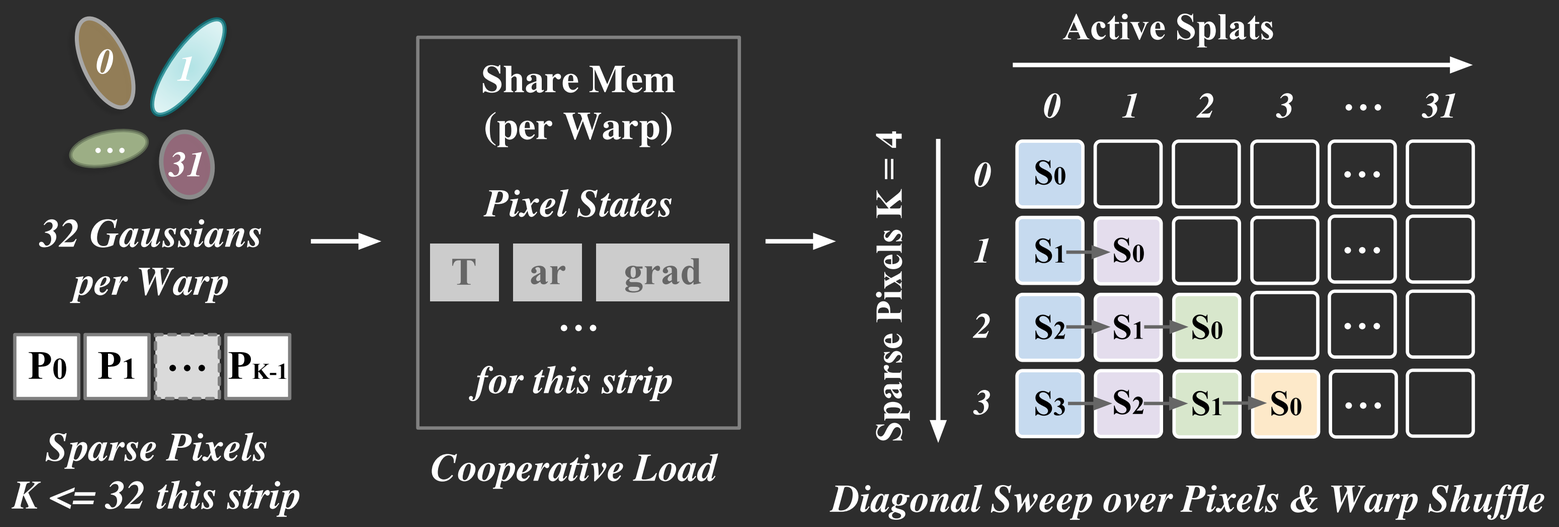
$$\bar{E}_v = \frac{1}{|\Omega|} \sum_{\mathbf{p}\in\Omega} \mathbf{E}_v(\mathbf{p}), \qquad v^\star = \arg\max_v \bar{E}_v, \quad \text{(Anchor View)}$$

Neighbor-View Searching :

$$\pi(u \mid v^\star) \propto \exp\Big(-\lambda_d \|\mathbf{c}_u - \mathbf{c}_{v^\star}\|_2 - \lambda_\theta \big(1 - \langle \mathbf{d}_u, \mathbf{d}_{v^\star} \rangle\big) \Big), \quad \text{(K Sampled Views)}$$

Efficient Sparse Pixel Rasterization





Forward Pixel Rasterization :

1) Maintain **Valid Tile Indices**

2) Calc. **Tile-Pixel Offsets**

3) Query Pixel Indices in **Each Block**

Backward :

1) Sampled Sparse Pixels

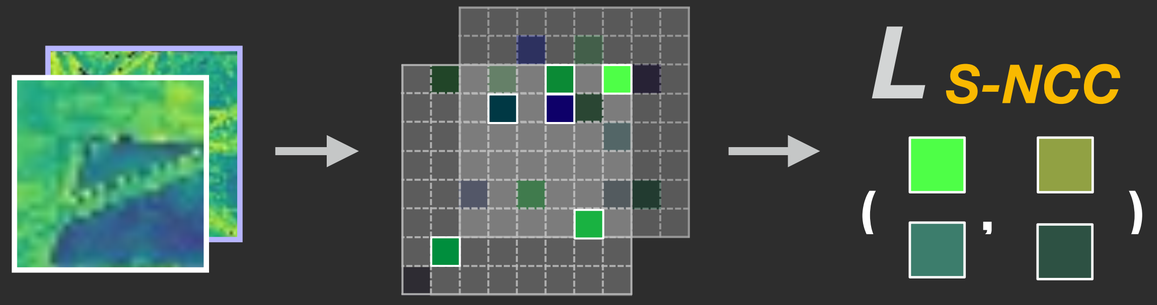
2) **Per-Gaussian** Warp Assignment

3) **Warp-Shuffle** Accumulation

4) Gradient **Atomic Reduction**

Making Sparse Optimization Effective

How to Optimize ?



Tile-Wise Structure Loss :

$$\mathscr{L}^{\tau}_{S-NCC} = 1 - \mathrm{NCC}\big(\hat{\mathbf{I}}_{t,v}(\mathscr{P}_{t,v}(\tau)), \mathbf{I}_{v}(\mathscr{P}_{t,v}(\tau))\big)\,.$$

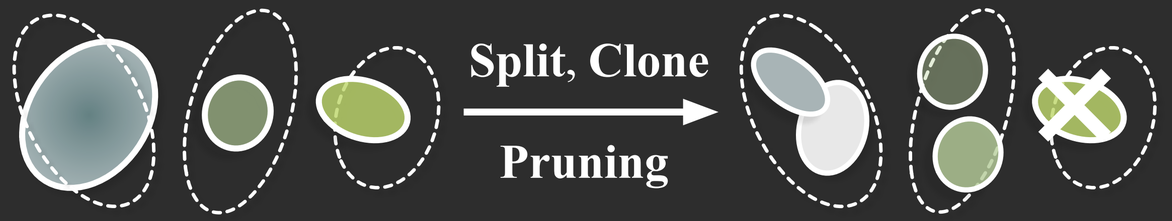
Gradient-Balanced Weighting :

$$\lambda^{\tau}_{ncc} = \mathrm{clip}\left(\frac{\bar{g}^{\tau}_{\ell_1}}{\bar{g}^{\tau}_{S-NCC}+\epsilon}, 0, 1\right),$$

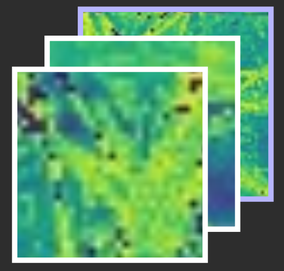
$$\bar{g}^{\tau}_{k} = \mathbb{E}_{\mathbf{p}\in\mathscr{P}_{t,v}(\tau)}\Big[\|\nabla_{\hat{\mathbf{I}}(\mathbf{p})}\mathscr{L}^{\tau}_{k}(\mathbf{p})\|_1\Big], \quad k \in \{\ell_1, \text{S-NCC}\}\,.$$

Moment-LM Solver : $\big(\mathbf{J}_g^{\top}\mathbf{J}_g + \mathrm{diag}(\sqrt{\mathbf{v}_g}+\epsilon)\big)\,\Delta_g = -\,\mathbf{m}_g$

What to Optimize ? Compact Gaussians



$Grad_{2d} > \tau_g$ & $Err_{gau} > \varepsilon_g$ & $Dist_{gau} > d_g$



Rasterize (Warp Reduce)

Error Maps

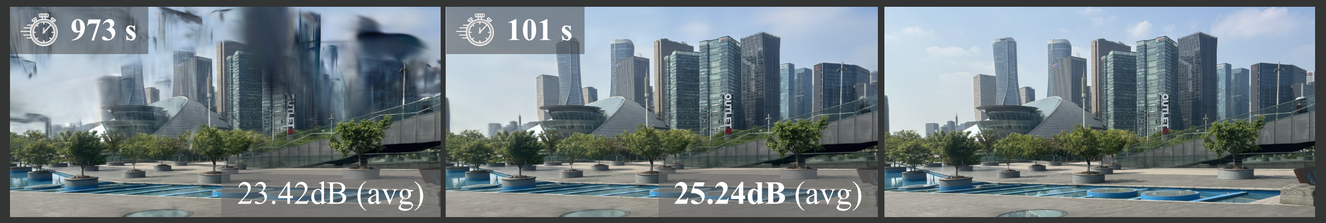
Per-Gaussian Error & Weight Accumulation :

$$\mathscr{E}^{(t)}_{i} = \sum_{v,\,\mathbf{p}} e_{t,v}(\mathbf{p})\, w_{t,i}(\mathbf{p}), \quad \mathscr{W}^{(t)}_{i} = \sum_{v,\,\mathbf{p}} w_{t,i}(\mathbf{p})\,.$$

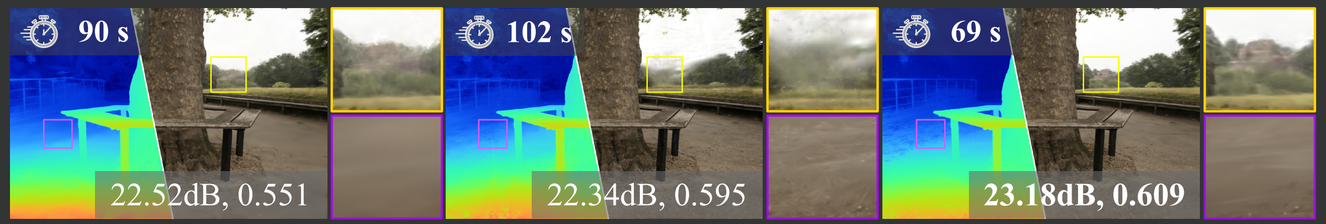
Normalization by Visibility :

$$e^{(t)}_{i} = \mathscr{E}^{(t)}_{i}/(\mathscr{W}^{(t)}_{i}+\epsilon) \qquad d^{(t)}_{i} = \mathscr{D}^{(t)}_{i}/(\mathscr{W}^{(t)}_{i}+\epsilon)$$

Ablation Study : WHY It Works

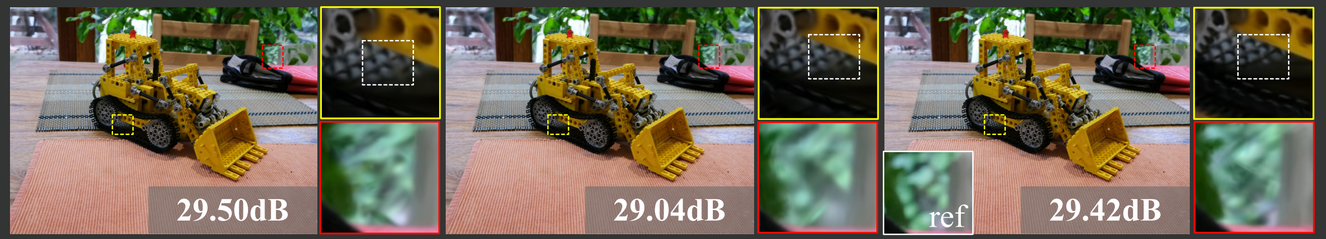


3DGS (baseline) + Err-SPS. ⋯ Err-DP. **(Full)** **Ground Truth**



+ Err-SPS. & Geo-VS. + Sparse-NCC. + Err-DP. **(Full)**

Err-SPS. → RPS. Geo-VS. → RVS. **Ours (Full)**



S-NCC. → S-SSIM. Sparse Adam (only) **Ours (Full)**

Ablation on *MipNeRF-360* Dataset

Ablated Method	Time↓	PSNR↑	SSIM↑	LPIPS↓	N_{GS}↓
3DGS	891	27.55	0.816	**0.215**	2.73M
+ Err-SPS.	85	26.58	0.737	0.331	0.76M
+ Geo-VS.	89	26.73	0.743	0.321	1.05M
+ Sparse-NCC.	95	27.14	0.773	0.279	0.98M
+ Err-DP. **(Full)**	77	**27.57**	0.794	0.256	0.64M
Err-SPS. → RPS.	**75**	27.10	0.780	0.276	0.56M
Geo-VS. → RVS.	76	27.26	0.780	0.274	**0.47M**
Err-DP. → FastGS	91	27.55	0.801	0.258	0.60M
ℓ_1 loss (only)	92	27.09	0.761	0.302	0.76M
S-NCC. → S-SSIM.	86	27.21	0.773	0.287	0.74M

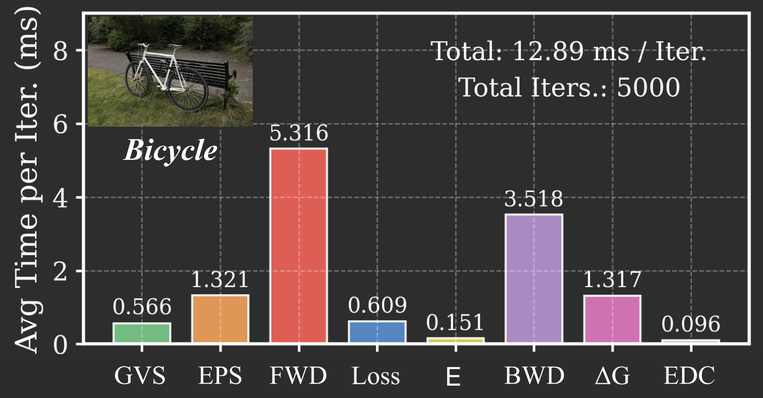
Err-SPS. ***Dominates Major Acceleration***

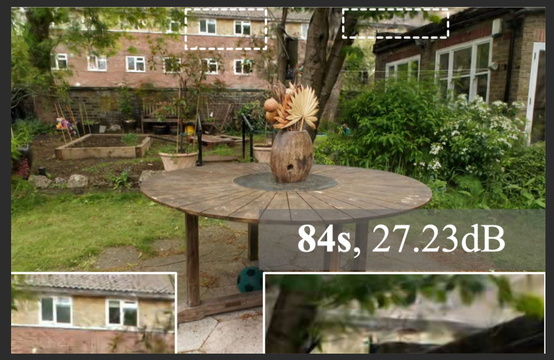
Geo-VS. ***Improved Multi-View Consistency***

Sparse-NCC. ***Recovers Fine Structures***

Err-DP. ***Adaptive Capacity Allocation***

Analysis of TurboGS

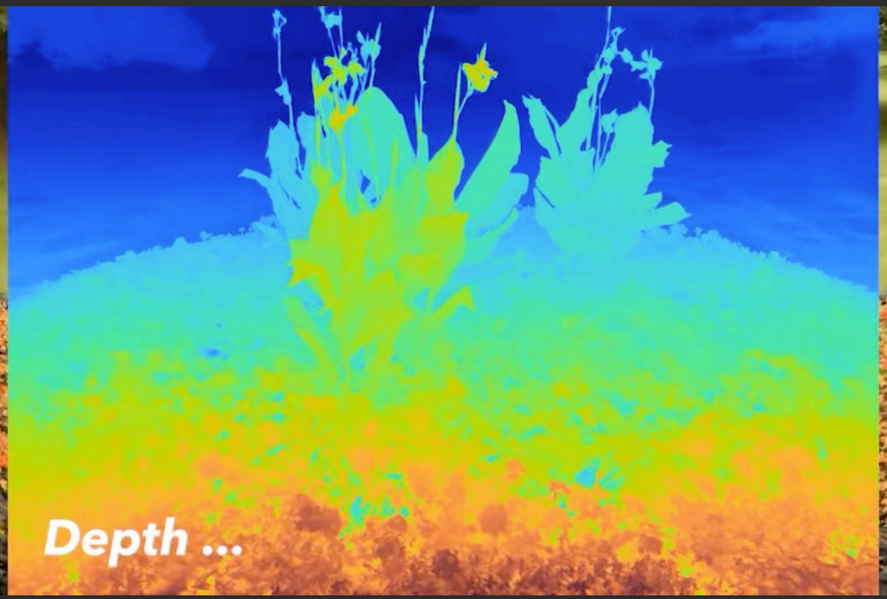




Completes training within ~80s in three representative datasets, while achieving competitive quality.

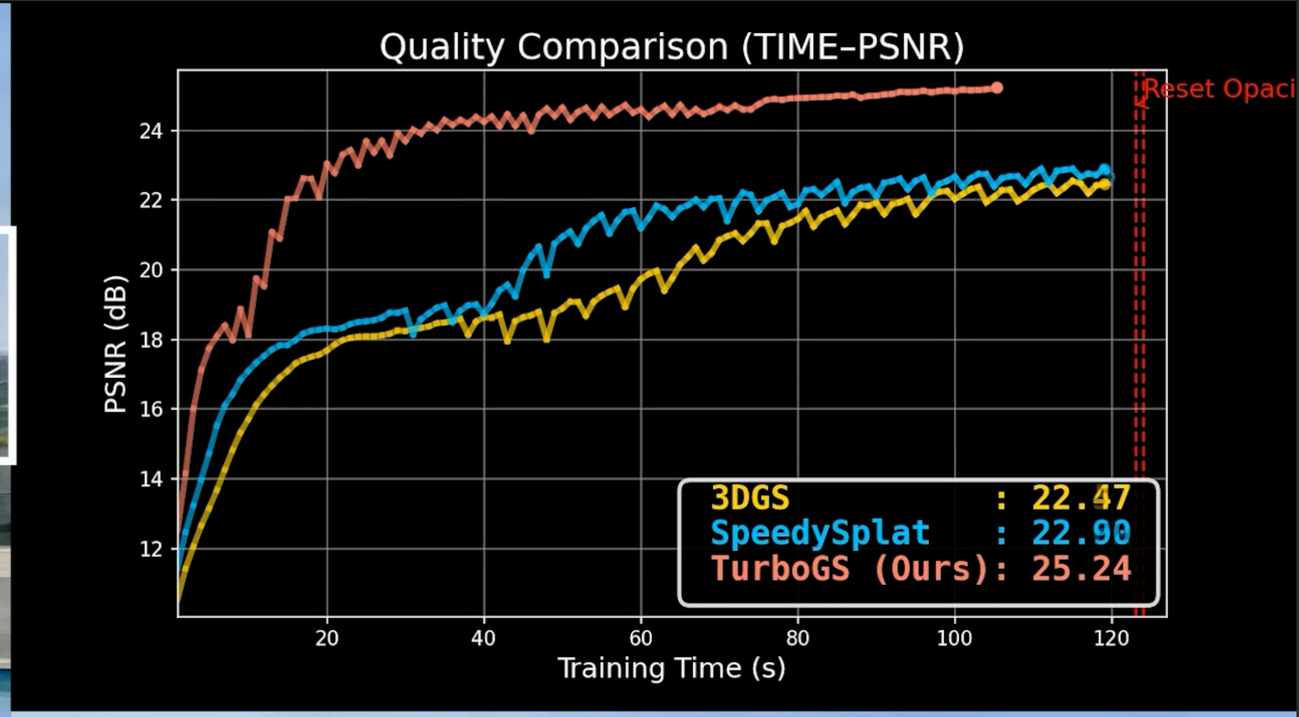
Method	Mip-NeRF 360 (*Barron et al., 2022*)						Tanks & Temples (*Knapitsch et al., 2017*)						Deep Blending (*Hedman et al., 2018*)					
	Time↓	PSNR↑	SSIM↑	LPIPS↓	N_{GS}↓	FPS↑	Time↓	PSNR↑	SSIM↑	LPIPS↓	N_{GS}↓	FPS↑	Time↓	PSNR↑	SSIM↑	LPIPS↓	N_{GS}↓	FPS↑
3DGS	891	27.55	0.816	**0.215**	2.73M	199	583	23.84	**0.853**	**0.169**	1.58M	252	921	29.85	**0.907**	**0.238**	2.48M	208
Taming-3DGS	**148**	27.26	0.795	0.659	0.67M	**413**	**97**	23.77	0.836	0.211	0.32M	**612**	**100**	29.92	0.903	0.271	0.29M	601
Mini-Splatting	674	27.32	**0.822**	**0.217**	**0.49M**	279	479	23.24	0.836	0.202	**0.20M**	405	586	29.95	**0.907**	0.254	0.35M	364
Speedy-Splat	533	26.95	0.786	0.288	**0.32M**	**983**	292	23.41	0.820	0.239	**0.19M**	**1138**	497	29.55	0.903	0.268	**0.25M**	**1121**
3DGS-LM	622	27.41	0.814	0.220	3.35M	218	369	23.57	**0.844**	0.185	1.80M	285	572	29.58	**0.906**	**0.246**	2.89M	218
DashGaussian	169	**27.69**	**0.817**	0.220	2.24M	259	141	**24.04**	**0.851**	**0.181**	1.20M	334	112	**30.12**	**0.907**	**0.248**	1.95M	289
FastGS	**111**	27.51	0.805	0.261	**0.38M**	**1002**	**91**	**24.06**	0.842	0.209	**0.24M**	**1087**	**81**	30.06	0.905	0.267	**0.22M**	**1144**
ConeGS	778	**27.74**	**0.823**	**0.202**	0.87M	**427**	749	23.88	**0.853**	**0.160**	0.55M	564	805	**30.28**	**0.909**	**0.238**	0.52M	**655**
TurboGS (Ours)	**77**	27.57	0.794	0.256	0.64M	153	**73**	23.79	0.832	0.200	0.58M	170	**62**	30.00	0.900	0.277	0.49M	231
TurboGS-Big (Ours)	168	**27.99**	0.815	0.224	1.34M	134	146	**24.14**	0.841	**0.181**	0.91M	151	132	**30.26**	**0.909**	0.259	0.69M	216

Optimization Progress Visualization



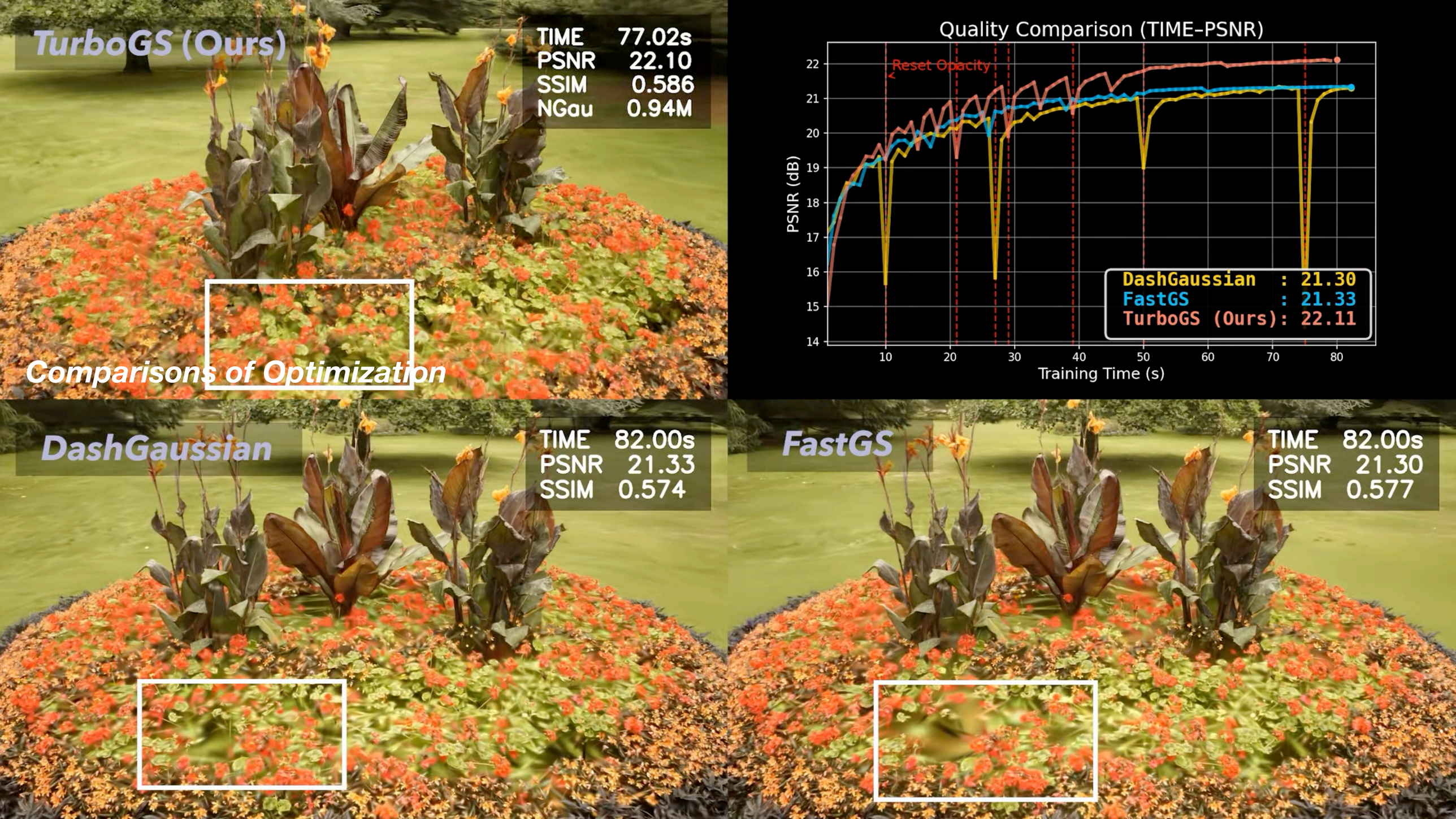

OMMO-10 Part1
TIME : 147s
PSNR : 28.99dB
SSIM : 0.845
OMMO-10 Part2
TIME : 147s
PSNR : 28.99dB
SSIM : 0.845
Large-Scale Scene Results
1146 x 856
Rubble Full Data
Zoom in
Zoom in
4580 x 3430
TIME 367.00s
PSNR 25.88
SSIM 0.775
NGau 2.78M
Rubble Sub Region (Re-Run COLMAP)


TurboGS (Ours)
TIME 100.00s
PSNR 25.13
SSIM 0.772
NGau 0.66M
Faster with Fine Details
Comparisons of Optimization


Quality Comparison (TIME–PSNR)
Reset Opaci
PSNR (dB)
Training Time (s)
3DGS : 22.47
SpeedySplat : 22.90
TurboGS (Ours): 25.24


3DGS
TIME 119.00s
PSNR 22.47
SSIM 0.698


Speedy-Splat
TIME 119.00s
PSNR 22.90
SSIM 0.715


TurboGS (Ours)
TIME 77.02s
PSNR 22.10
SSIM 0.586
NGau 0.94M
Comparisons of Optimization
Quality Comparison (TIME-PSNR)
Reset Opacity
PSNR (dB)
Training Time (s)
DashGaussian : 21.30
FastGS : 21.33
TurboGS (Ours): 22.11
DashGaussian
TIME 82.00s
PSNR 21.33
SSIM 0.574
FastGS
TIME 82.00s
PSNR 21.30
SSIM 0.577

Key Takeaways

Main Insight : Not All Pixels deserve equal computation.

Allocate computation adaptively across views, pixels and Gaussians.

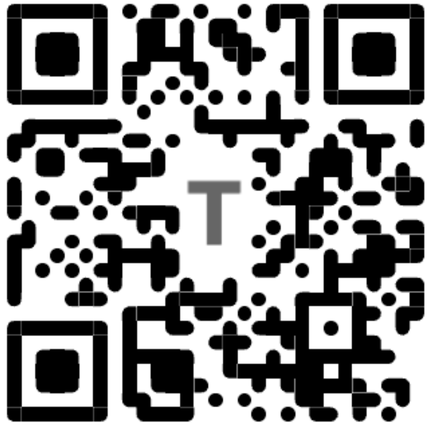
Conclusion :

- ***Error-Guided Sparse Optimization** achieves **> 10x Speedup** than Baseline 3DGS*
- ***Compact Gaussian Representation** with **Error-Guided Density Control***

Limitation :

- *Low-Texture Regions : May Need Adaptive Dense Supervision;*
- *Extend to Dynamic Scene*

THANKS YOU



Project Page

ICML
International Conference
On Machine Learning

https://zhengdong.site/projects/TurboGS/