NEURAL INFORMATION PROCESSING SYSTEMS

ODGS: 3D Scene Reconstruction from Omnidirectional Images with 3D Gaussian Splatting

Suyoung Lee[1]*, Jaeyoung Chung[1]*, Jaeyoo Huh[2], Kyoung Mu Lee[12]

[1]Dept. of ECE & ASRI, [2]IPAI
Seoul National University, Seoul, Korea

Computer Vision Lab
Seoul National University

Overview

- We introduce ODGS, a **3D reconstruction framework for omnidirectional images** based on 3D Gaussian splatting, achieving both 100 times faster rendering speed and higher reconstruction accuracy than NeRF-based methods

- We present a detailed **geometric interpretation** of the rasterization for omnidirectional images, along with **mathematical verification**, and propose a CUDA rasterizer based on the interpretation

Motivation

- There are increasing demands in 3D scene reconstruction by:
  - Development and popularization of VR/MR devices
  - Growth in many applications such as robotics or autonomous driving

- Perspective V.S. Omnidirectional images for surrounding scenes
  - Perspective images are popular **but** requires computations to stitch and compose multiple images
  - Omnidirectional images contain the entire scene in a single image, thereby reducing the additional computation cost of composing the images

- Omnidirectional images are gaining more popularity with the spread of 360-degree cameras.

Motivation

- There are significant development in representing 3D scenes:
  - Neural Radiance Field (NeRF): Implicit representation
  - 3D Gaussian splatting (3DGS): Explicit representation

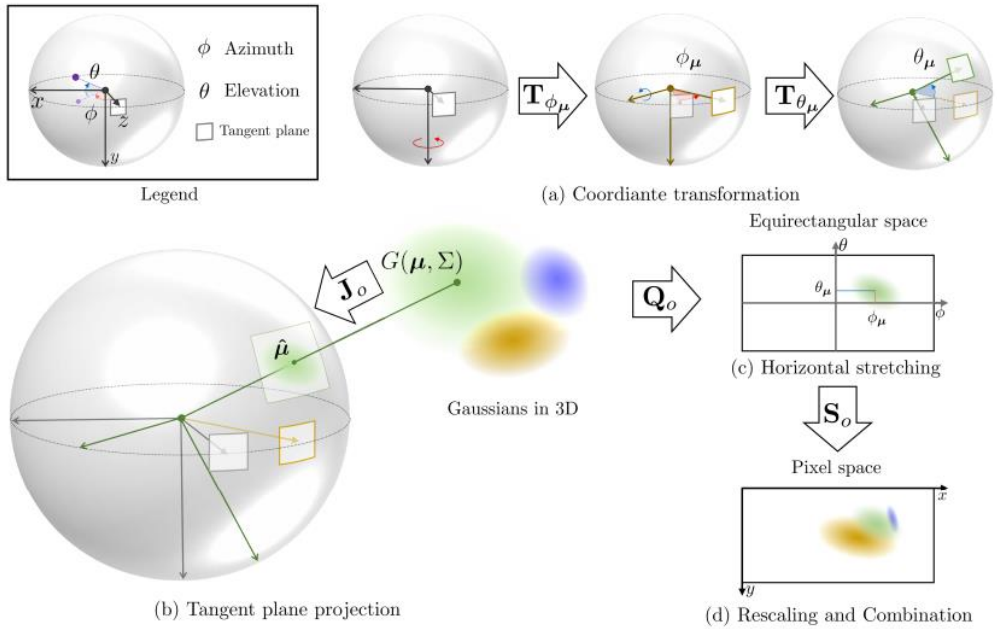
- 3DGS is getting more attention for fast optimization and rendering

- However, applying typical 3DGS to omnidirectional images would fail due to the different optical characteristics between omnidirectional a nd perspective images for rasterization

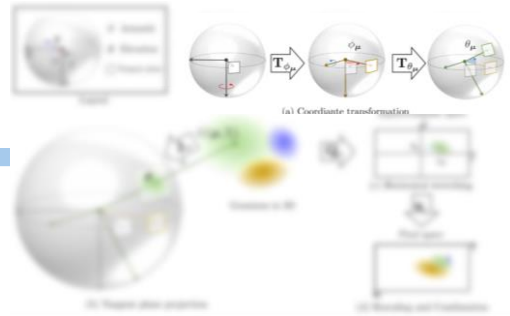
Method

- ODGS(OmniDirectional Gaussian Splatting)
  - Fast CUDA rasterizer designed for omnidirectional images along with geometric interpretation
  - **Gist**: project each Gaussian to the corresponding tangent plane of the sphere and combine all projected Gaussian in the omnidirectional image plane.

- 4 processes in ODGS rasterizer
  - Coordinate transformation
  - Perspective projection
  - Equirectangular transformation
  - Pixel space transformation



Method

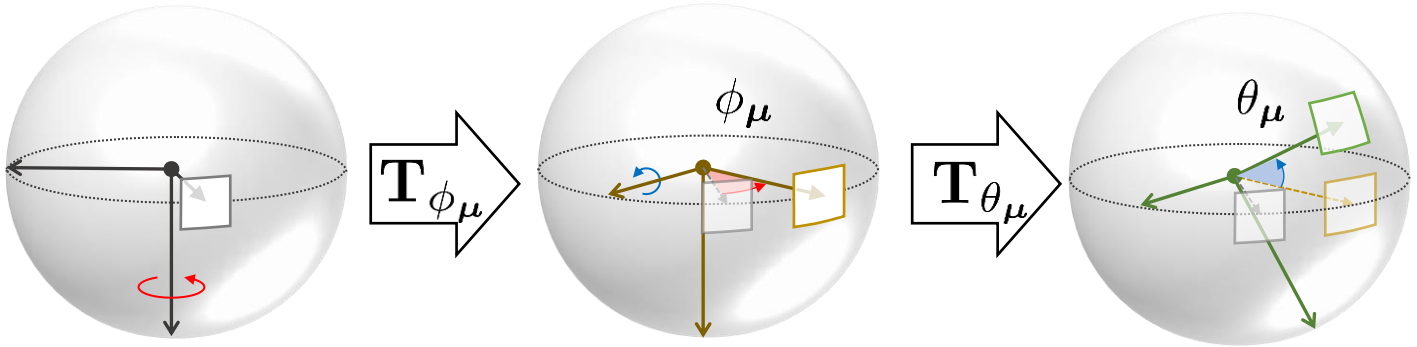


1. Coordinate transformation ($\mathbf{T}_{\mu}$)

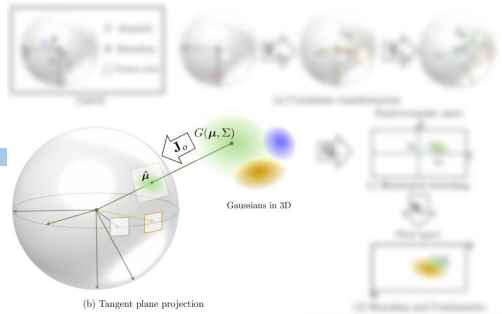
- We consider a perspective camera with unit focal length (*called* “unit camera”)
  → The image plane would be placed on the surface of the unit sphere
- We transform the coordinate from the camera to make the z axis heads toward the center of the Gaussian

$$\mathbf{T}_{\boldsymbol{\mu}} = \mathbf{T}_{\theta_\mu} \times \mathbf{T}_{\phi_\mu} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_\mu & \sin\theta_\mu \\ 0 & -\sin\theta_\mu & \cos\theta_\mu \end{bmatrix} \times \begin{bmatrix} \cos\phi_\mu & 0 & -\sin\phi_\mu \\ 0 & 1 & 0 \\ \sin\phi_\mu & 0 & \cos\phi_\mu \end{bmatrix}$$

$$= \begin{bmatrix} \cos\phi_\mu & 0 & -\sin\phi_\mu \\ \sin\theta_\mu \sin\phi_\mu & \cos\theta_\mu & \sin\theta_\mu \cos\phi_\mu \\ \cos\theta_\mu \sin\phi_\mu & -\sin\theta_\mu & \cos\theta_\mu \cos\phi_\mu \end{bmatrix}.$$

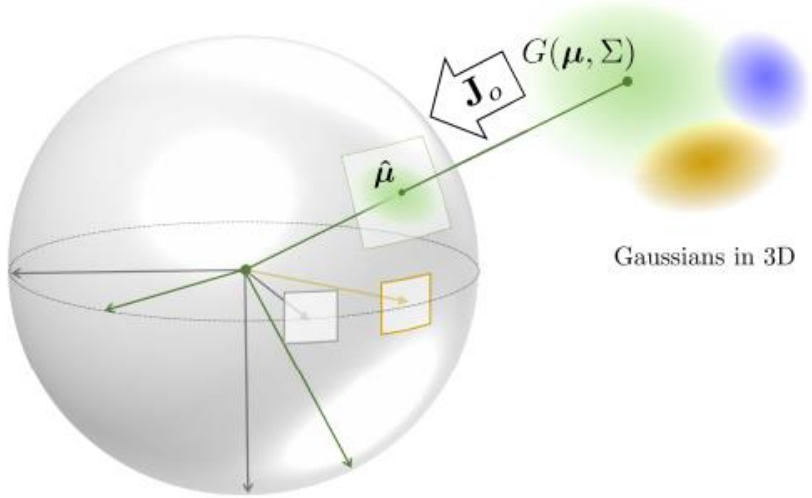


Method



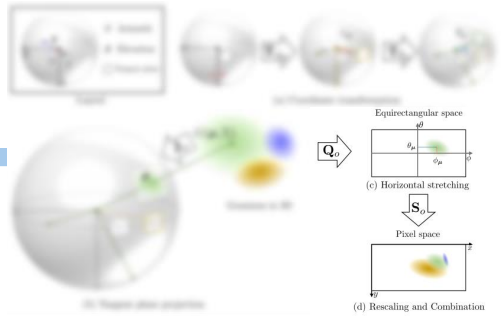
2. Perspective projection on the unit camera ($\mathbf{J}_o$)
   - Each Gaussian is projected to its corresponding tangent plane
   - This projection assumes a local affine approximation, similar to 3DGS
   - Since the focal length of the image is 1, the Jacobian matrix is computed as:

$$\mathbf{J}_o = \begin{bmatrix} 1/||\boldsymbol{\mu}|| & 0 & 0 \\ 0 & 1/||\boldsymbol{\mu}|| & 0 \end{bmatrix}$$



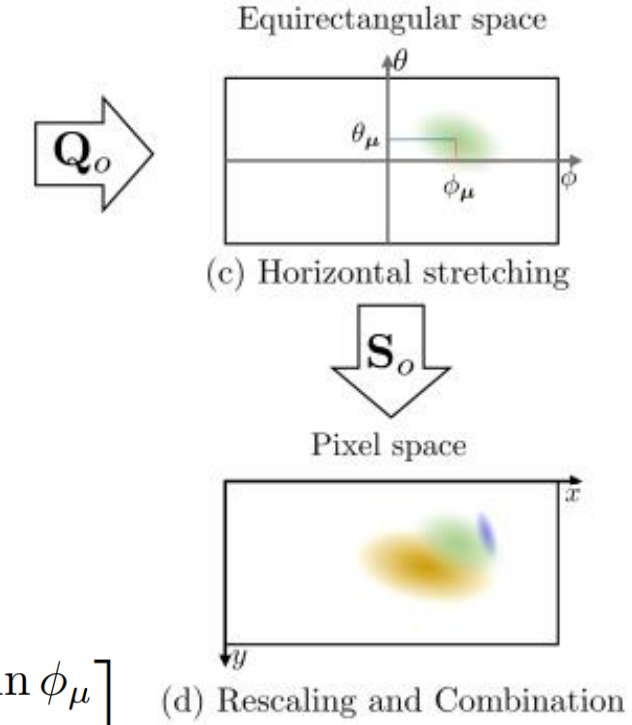
(b) Tangent plane projection

Method



3. Transform to equirectangular space ($\mathbf{Q}_o$)
   - By equirectangular transformation from the spherical surface to the cylindrical map, the horizontal size of Gaussian is stretched $\sec(\theta)$ times
4. Transform to pixel space ($\mathbf{S}_o$)
   - Finally, we rescale the covariance to the pixel space

$$\mathbf{Q}_o = \begin{bmatrix} \sec\theta_\mu & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{S}_o = \begin{bmatrix} W/2\pi & 0 \\ 0 & H/\pi \end{bmatrix}$$



- The final Jacobian matrix for ODGS:

$$\mathbf{J}_{omni} = \mathbf{S}_o\mathbf{Q}_o\mathbf{J}_o\mathbf{T}_{\boldsymbol{\mu}} = \begin{bmatrix} \frac{W}{2\pi||\boldsymbol{\mu}||}\sec\theta_\mu\cos\phi_\mu & 0 & -\frac{W}{2\pi||\boldsymbol{\mu}||}\sec\theta_\mu\sin\phi_\mu \\ \frac{H}{\pi||\boldsymbol{\mu}||}\sin\theta_\mu\sin\phi_\mu & \frac{H}{\pi||\boldsymbol{\mu}||}\cos\theta_\mu & \frac{H}{\pi||\boldsymbol{\mu}||}\sin\theta_\mu\cos\phi_\mu \end{bmatrix}$$

Proof of ODGS

- The mathematical verification of ODGS can be found in the appendix

$$\frac{\partial \pi_o(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \begin{bmatrix} \frac{W}{2\pi}\frac{\boldsymbol{\mu}_z}{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2} & 0 & -\frac{W}{2\pi}\frac{\boldsymbol{\mu}_x}{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2} \\ -\frac{H}{\pi||\boldsymbol{\mu}||^2}\frac{\boldsymbol{\mu}_x\boldsymbol{\mu}_y}{\sqrt{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2}} & \frac{H}{\pi||\boldsymbol{\mu}||^2}\sqrt{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2} & -\frac{H}{\pi||\boldsymbol{\mu}||^2}\frac{\boldsymbol{\mu}_y\boldsymbol{\mu}_z}{\sqrt{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{W}{2\pi||\boldsymbol{\mu}||}\frac{||\boldsymbol{\mu}||}{\sqrt{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2}}\frac{\boldsymbol{\mu}_z}{\sqrt{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2}} & 0 & -\frac{W}{2\pi||\boldsymbol{\mu}||}\frac{||\boldsymbol{\mu}||}{\sqrt{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2}}\frac{\boldsymbol{\mu}_x}{\sqrt{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2}} \\ \frac{H}{\pi||\boldsymbol{\mu}||}\frac{-\boldsymbol{\mu}_y}{||\boldsymbol{\mu}||}\frac{\boldsymbol{\mu}_x}{\sqrt{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2}} & \frac{H}{\pi||\boldsymbol{\mu}||}\frac{\sqrt{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2}}{||\boldsymbol{\mu}||} & \frac{H}{\pi||\boldsymbol{\mu}||}\frac{-\boldsymbol{\mu}_y}{||\boldsymbol{\mu}||}\frac{\boldsymbol{\mu}_z}{\sqrt{\boldsymbol{\mu}_x^2+\boldsymbol{\mu}_z^2}} \end{bmatrix}$$

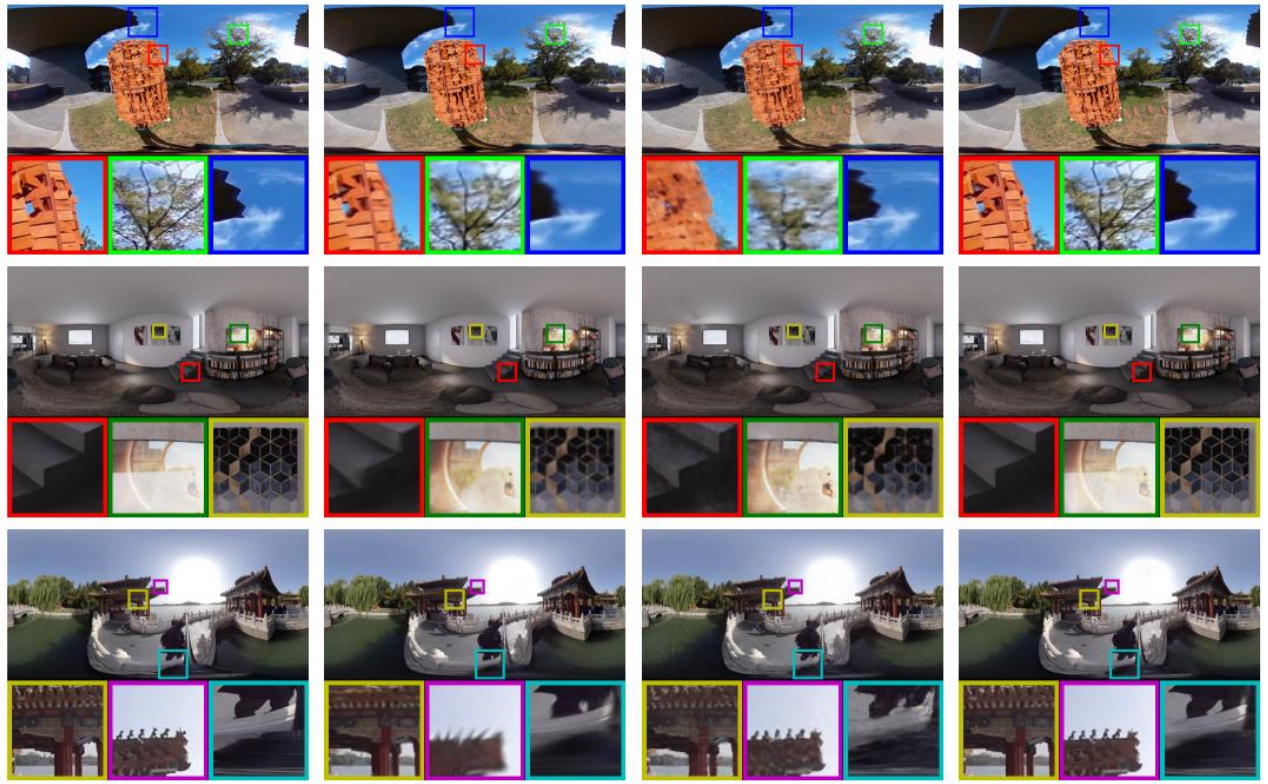
$$= \begin{bmatrix} \frac{W}{2\pi||\boldsymbol{\mu}||}\sec\theta_\mu\cos\phi_\mu & 0 & -\frac{W}{2\pi||\boldsymbol{\mu}||}\sec\theta_\mu\sin\phi_\mu \\ \frac{H}{\pi||\boldsymbol{\mu}||}\sin\theta_\mu\sin\phi_\mu & \frac{H}{\pi||\boldsymbol{\mu}||}\cos\theta_\mu & \frac{H}{\pi||\boldsymbol{\mu}||}\sin\theta_\mu\cos\phi_\mu \end{bmatrix}$$

$$= \mathbf{J}_{omni}$$

Experiment

- Results on Egocentric videos
  - ODGS produces prominent results with only 10 minutes of optimization
  - ODGS shows 100 times faster rendering speed than EgoNeRF

Dataset	Methods	10 min PSNR↑	10 min SSIM↑	10 min LPIPS↓	100 min PSNR↑	100 min SSIM↑	100 min LPIPS↓	Time↓ (sec.)
OmniBlender	NeRF(P)	19.20	0.6124	0.5359	20.04	0.6092	0.4949	62.71
	3DGS(P)	29.36	0.8770	0.1400	21.19	0.7528	0.3021	0.112
	TensoRF	25.36	0.7249	0.3855	26.08	0.7416	0.3170	10.77
	EgoNeRF	28.29	0.8309	0.2194	30.89	0.8934	0.1269	23.78
	ODGS	**32.76**	**0.9234**	**0.0469**	**33.05**	**0.9229**	**0.0343**	**0.028**
Ricoh360	NeRF(P)	14.33	0.5616	0.5794	16.16	0.5617	0.5716	62.46
	3DGS(P)	**25.12**	0.7932	0.2397	22.07	0.7228	0.3218	0.152
	TensoRF	23.35	0.6812	0.5200	23.97	0.6936	0.4653	10.30
	EgoNeRF	24.74	0.7467	0.3243	25.49	0.7737	0.2825	23.89
	ODGS	24.94	**0.8135**	**0.1489**	**26.27**	**0.8462**	**0.1051**	**0.026**
OmniPhotos	NeRF(P)	18.14	0.6158	0.5514	20.80	0.6388	0.4772	62.08
	3DGS(P)	25.61	0.8310	0.2100	23.30	0.7859	0.2670	0.110
	TensoRF	22.78	0.6841	0.5089	23.73	0.7038	0.4467	9.707
	EgoNeRF	25.20	0.7722	0.2662	26.90	0.8349	0.1766	23.88
	ODGS	**26.24**	**0.8704**	**0.1108**	**27.04**	**0.8878**	**0.0875**	**0.028**



(a) Ground truth (b) 3DGS(P) [27] (c) EgoNeRF [10] (d) ODGS

Experiment

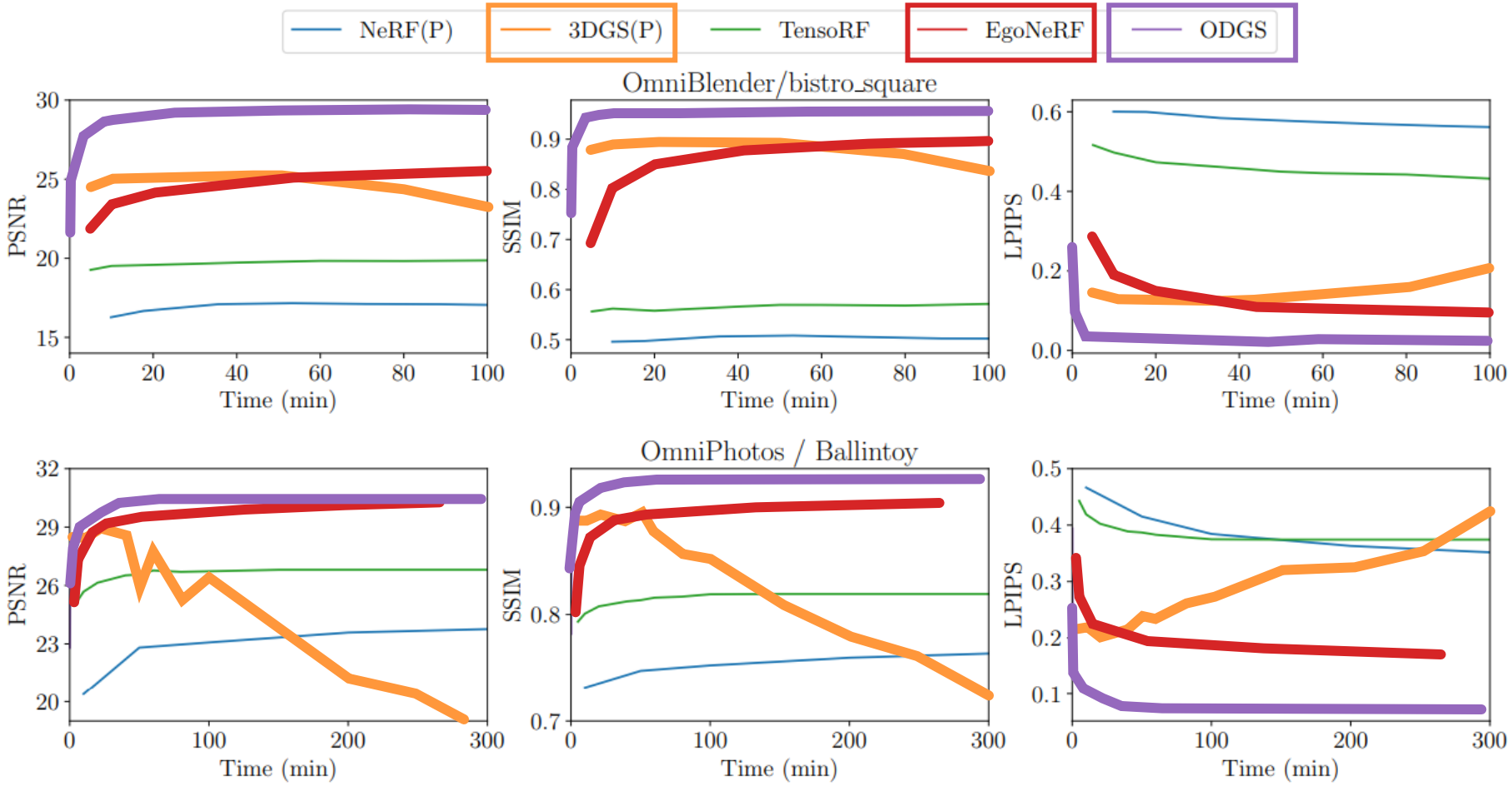
- Results on Roaming videos
  - ODGS produces prominent results with only 10 minutes of optimization.
  - Methods designed only for egocentric motion cannot reconstruct large scenes created by roaming.

Dataset	Methods	10 min PSNR↑	10 min SSIM↑	10 min LPIPS↓	100 min PSNR↑	100 min SSIM↑	100 min LPIPS↓	Time↓ (sec.)
360Roam	NeRF(P)	15.07	0.6848	0.4839	15.26	0.6813	0.5025	62.98
	3DGS(P)	20.17	0.7001	0.3536	19.34	0.6576	0.3837	0.104
	TensoRF	18.00	0.5988	0.7488	18.12	0.5895	0.7133	9.052
	EgoNeRF	20.45	0.6358	0.5334	**21.18**	0.6718	0.4444	24.03
	ODGS	**21.08**	**0.7066**	**0.3003**	20.85	**0.7111**	**0.2254**	**0.029**
OmniScenes	NeRF(P)	15.69	0.7218	0.4546	15.98	0.6890	0.4914	62.90
	3DGS(P)	23.61	0.8444	0.2835	17.14	0.7119	0.3906	0.194
	TensoRF	23.58	0.8118	0.3534	24.21	0.8208	0.3091	8.100
	EgoNeRF	22.78	0.7997	0.3463	**24.76**	0.8313	0.2623	23.66
	ODGS	**24.42**	**0.8526**	**0.1391**	24.51	**0.8505**	**0.1282**	**0.032**
360VO	NeRF(P)	15.71	0.6186	0.4949	17.78	0.6373	0.5064	61.97
	3DGS(P)	22.87	0.7861	0.2970	22.73	0.7822	0.3061	0.091
	TensoRF	19.74	0.6543	0.5876	20.31	0.6721	0.5640	7.815
	EgoNeRF	22.47	0.7325	0.4342	23.78	0.7677	0.3680	23.96
	ODGS	**24.63**	**0.8245**	**0.2175**	**26.68**	**0.8694**	**0.1264**	**0.026**

(a) Ground truth (b) 3DGS(P) [27] (c) EgoNeRF [10] (d) ODGS

Experiment

- Changes of performance according to optimization time
  - ODGS shows both faster convergence time and high performance



Conclusion

- We introduce ODGS, a **3D reconstruction framework for omnidirectional images** based on 3D Gaussian splatting, achieving both 100 times faster rendering speed and higher reconstruction accuracy than NeRF-based methods

- We present a detailed **geometric interpretation** of the rasterization for omnidirectional images, along with **mathematical verification**, and propose a CUDA rasterizer based on the interpretation

- Check our code at our Github repo!

arXiv

GitHub

NEURAL INFORMATION PROCESSING SYSTEMS

Thank You

Computer Vision Lab
Seoul National University