Deformable Gaussian Splatting for Dynamic View Synthesis

2024 summer seminar

Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

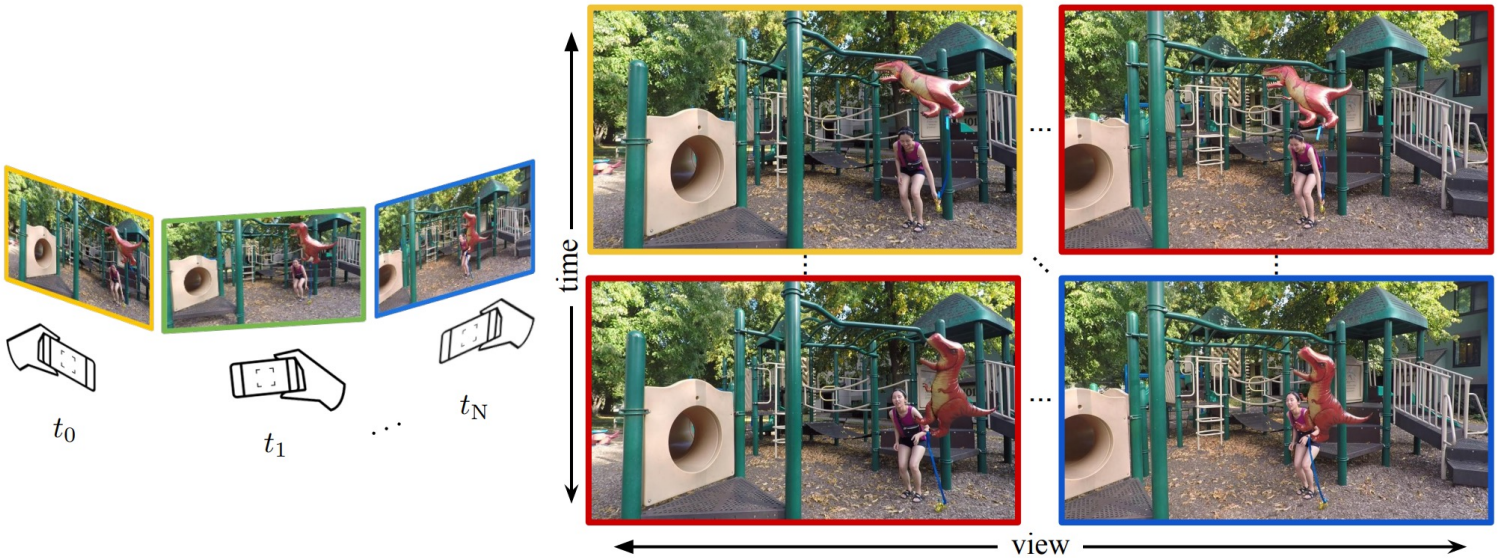
Presented By
Jimin Roh

Background

- Dynamic View Synthesis란?

  목적

  - Dynamic scene에 대한 monocular video가 주어지면, 특정 view와 특정 time에 대해 rendering하여 새로운 장면을 예측하는 것
  - Statice scene과 달리 time t에 따라 dynamic object의 deformation을 학습하는 것이 핵심



(a) Input: monocular video (b) Output: free-viewpoint rendering

< Dynamic view synthesis 설명 >

4D Gaussian Splatting for Real-Time Dynamic Scene Rendering
CVPR 2024

Abstract

- 이전 연구 문제점

  문제 1.

  - Dynamic scene에서는 시간에 따라 동적으로 객체들의 변화 (motion, shape)가 발생하기 때문에 하나의 Gaussian 집합으로 rendering이 어려움

  문제 2.

  - Dynamic view synthesis에서 기존 NeRF 방식은 training속도가 매우 느리며, rendering cost가 큼
  - 또한 deformation 학습과정에서 implicit하게 학습하여 3D scene geometry를 학습하기 어려움

Introduction

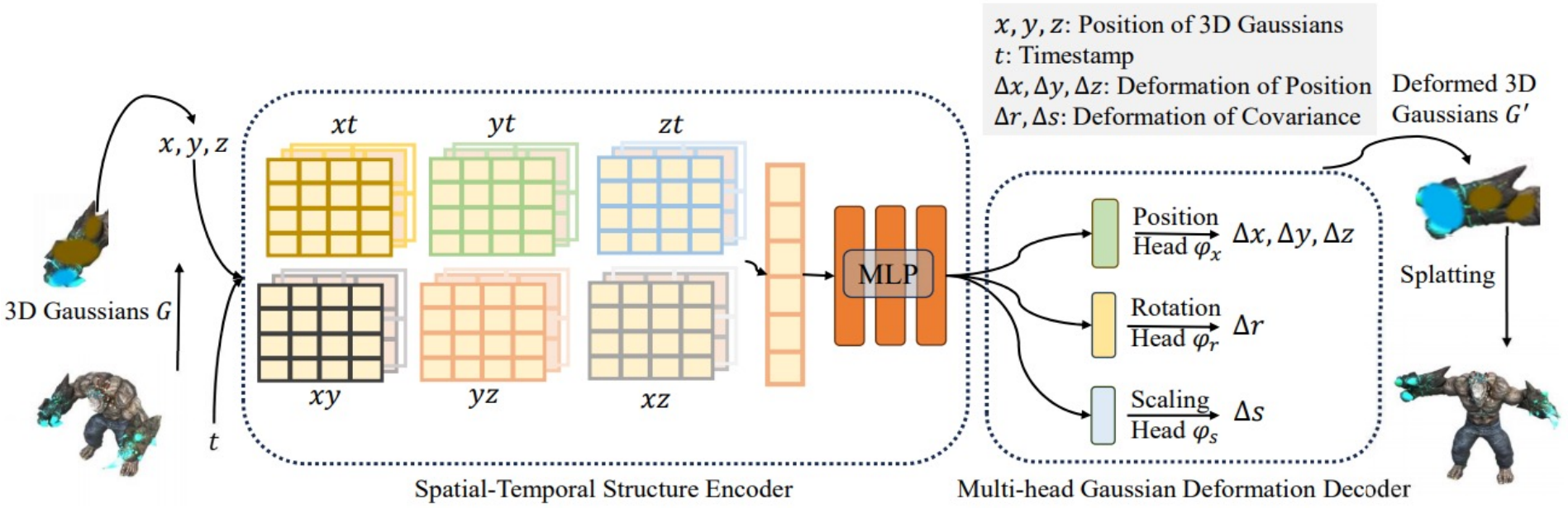
- 4D Gaussian splatting

  Spatial-Temporal Structure Encoder

  - Hexplane을 사용하여 spatial-temporal feature를 생성

  Multi-head Gaussian Deformation Decoder

  - 3개의 branch를 생성하여 Gaussian의 각 특성의 deformation을 예측



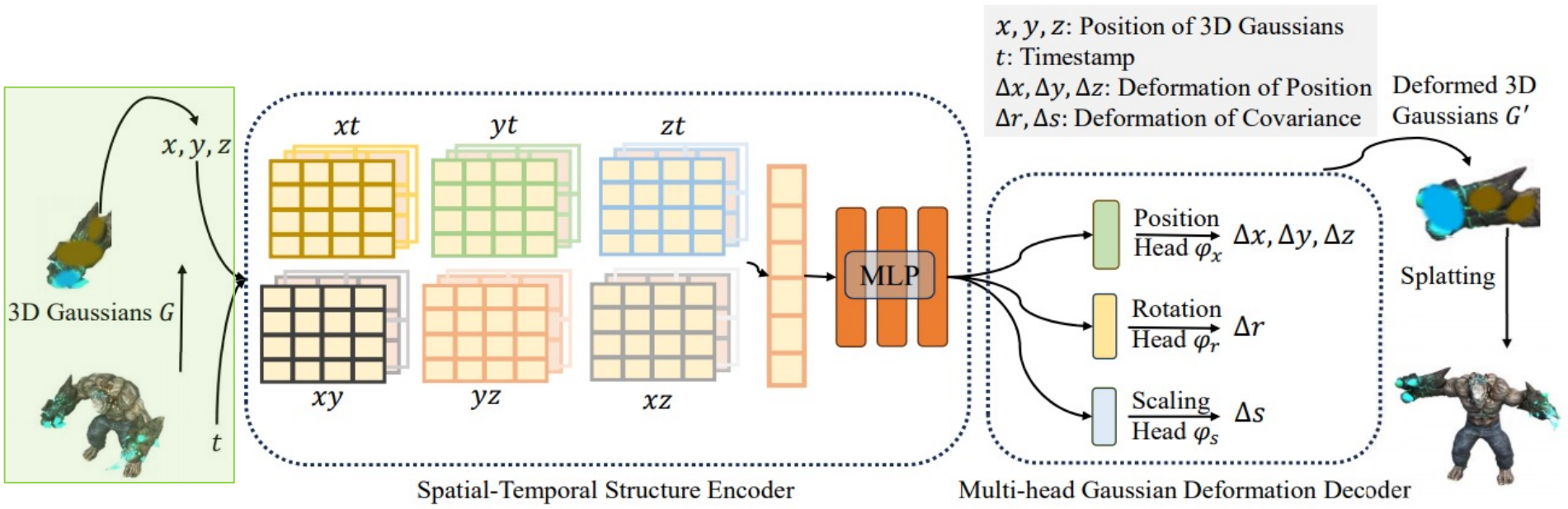
Introduction

- Contributions

  Initialize Gaussian set

  - Camera view와 timestamp에 정보를 담은 하나의 canonical Gaussian set을 생성

    ⁘ 각 time마다 Gaussian set들을 생성할 필요가 없어서 storage/memory cost를 줄임

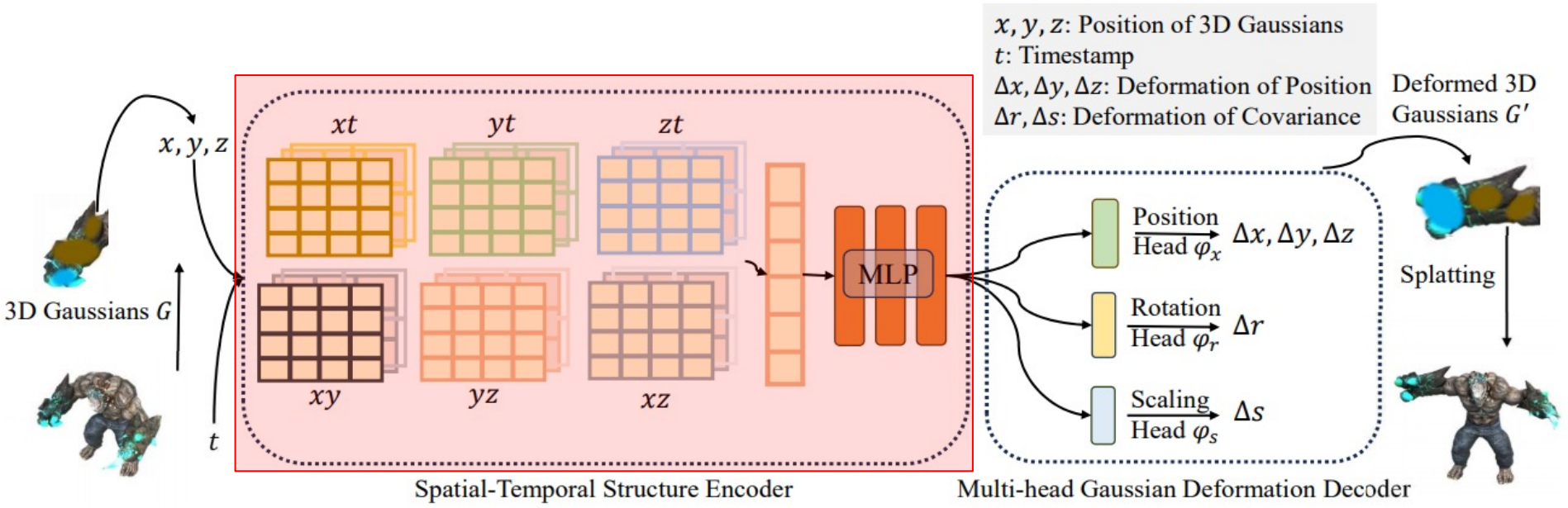


Introduction

- Contributions

Spatial-Temporal Structure Encoder

- 각 Gaussian을 HexPlane방식으로 decomposition하여 grid feature로 변경
  - Temporal information과 Gaussian point를 결합하여 spatial-temporal feature 생성
  - Multi-scale 방식을 사용하여 주변 point간의 연관성을 갖는 feature 생성

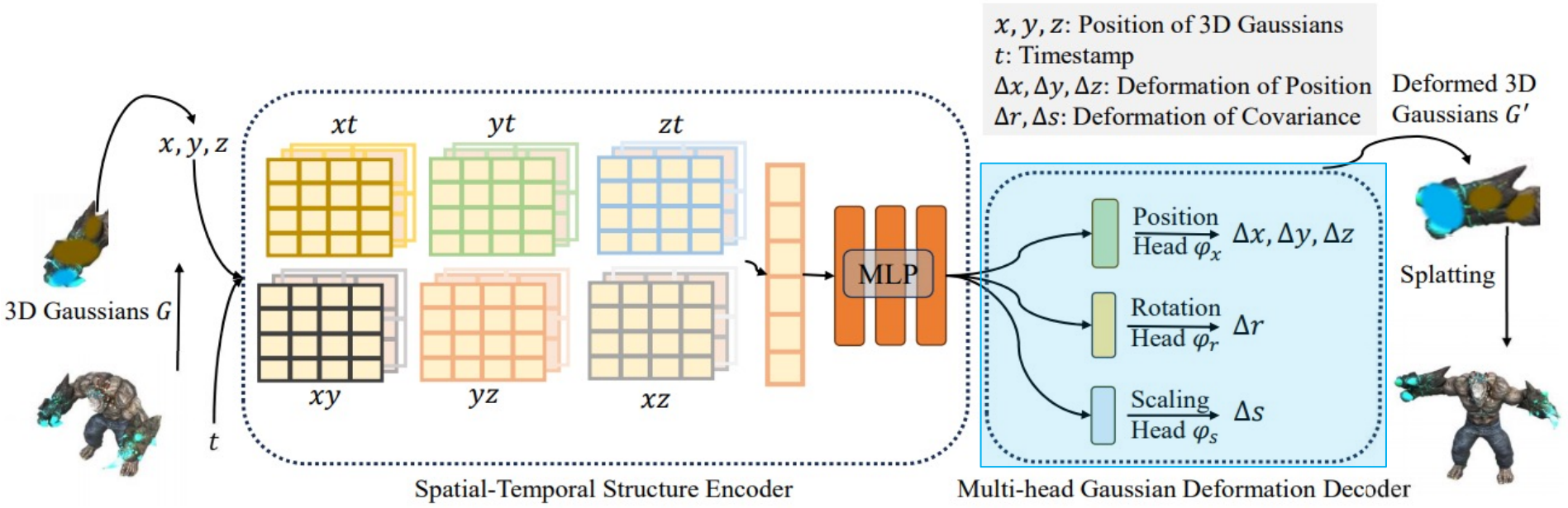


Introduction

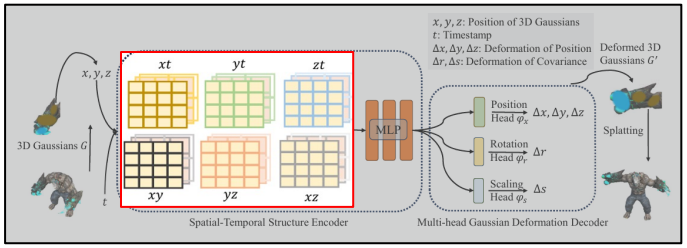
- Contributions

Multi-head Gaussian Deformation Decoder

- Encoder에서 생성한 grid feature를 multi-head layer의 입력으로 사용해 각 Gaussian의 deformation을 추출
  - 각 timestamp에 대한 Gaussian의 deformation을 추출하여, 각 canonical Gaussian의 motion, shape에 대한 변화를 예측



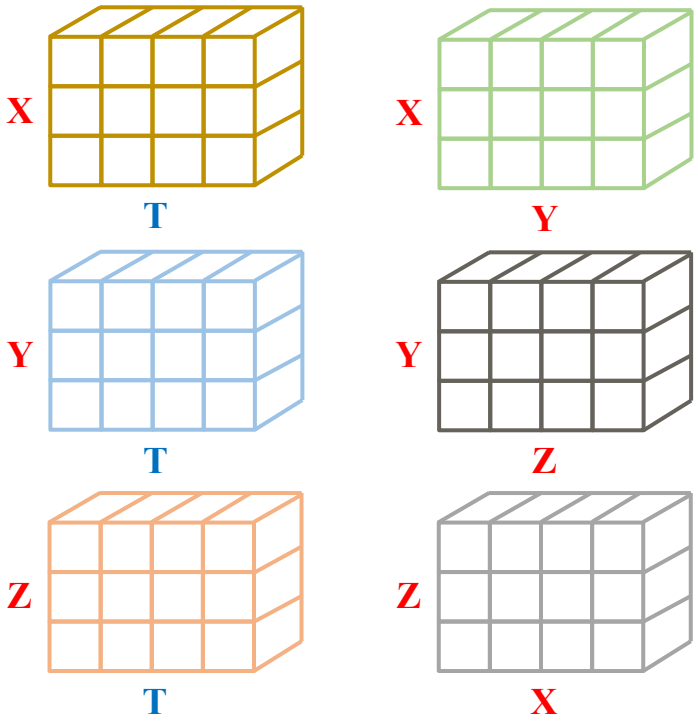
Method



- Spatial-temporal structure encoder

  Multi-resolution Hexplane 생성

  - 각 spatial(X,Y,Z)한 정보와 temporal(T)의 연관성 있는 feature 생성
  - Multi-scale의 정보를 이용해 주변 point의 관계도 알 수 있음



Scale

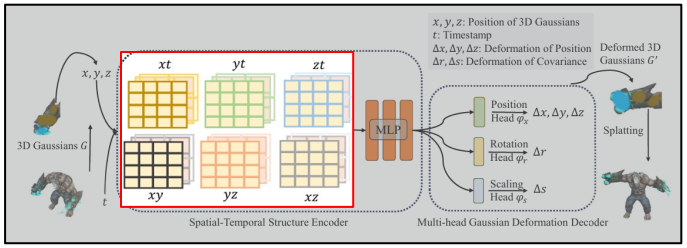
	X	Y	Z	T
• Rank 1 :	64,	64,	64,	150
• Rank 2 :	128,	128,	128,	150
• Rank 3 :	256,	256,	256,	150

Grid

- **Rank 1**
  - (X, Y) : 1, 16, 64, 64
  - (X, T) : 1, 16, 150, 64
- **Rank 2**
  - (X, Y) : 1, 16, 128, 128
  - (X, T) : 1, 16, 150, 128
- **Rank 3**
  - (X, Y) : 1, 16, 256, 256
  - (X, T) : 1, 16, 150, 256

Method

- Forward process



Normalization & Interpolation

Canonical Gaussian

Axis Aligned Bounding Box

Normalized Gaussian

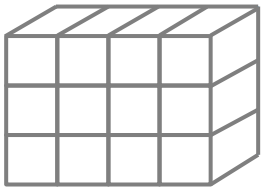
Point : X, Y, Z
(N, 3)

Time : T
(N, 1)

Concatenation

Input Gaussian
(N, 4)

(N x Rank)

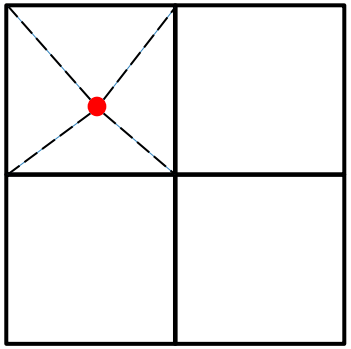


X,Y grid
(1, 16, 64, 64)

+

X,Y position
(N, 2)

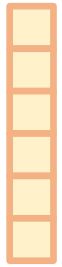
Interpolation



Grid feature
(N, 16)

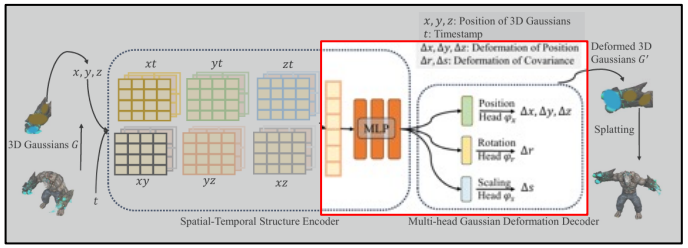
Concatenation

(N, 16) x Rank



Grid features
(N, 48)

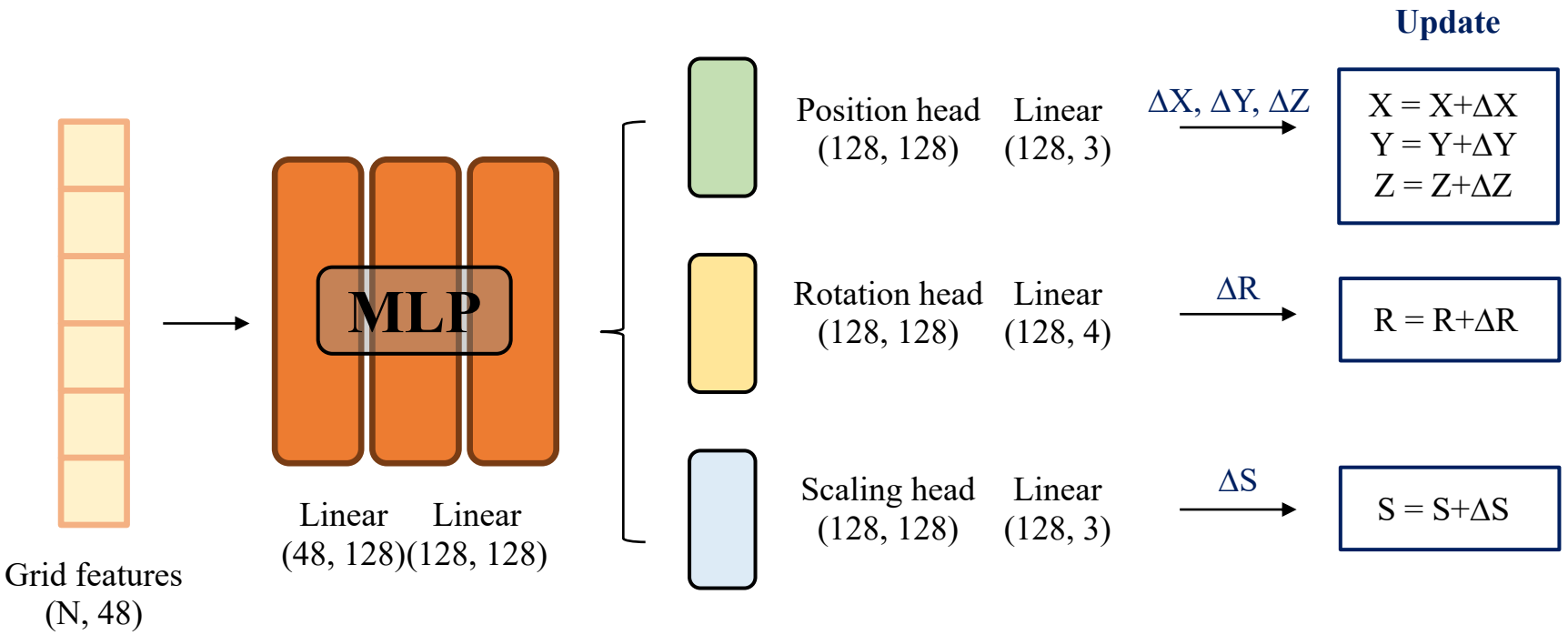
Method



- Forward process

Multi-head Gaussian deformation decoder

- Encoder를 통해 만들어진 grid feature로 Gaussian에 대한 deformable을 학습
- 이후 각 branch별로 세부적인 deformable을 추출

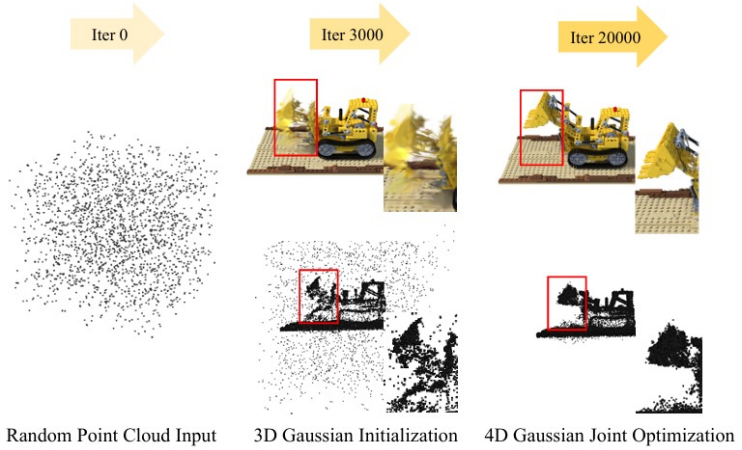


Experiment

- Optimization

Initialization

- SfM으로 추출한 point cloud를 초기 point로 사용
- Course단계에서 3DGS로 3000 iter까지 학습함
  - 이때 deformation network를 사용하지 않음
  - 즉, dynamic Gaussian을 추정하지 않음



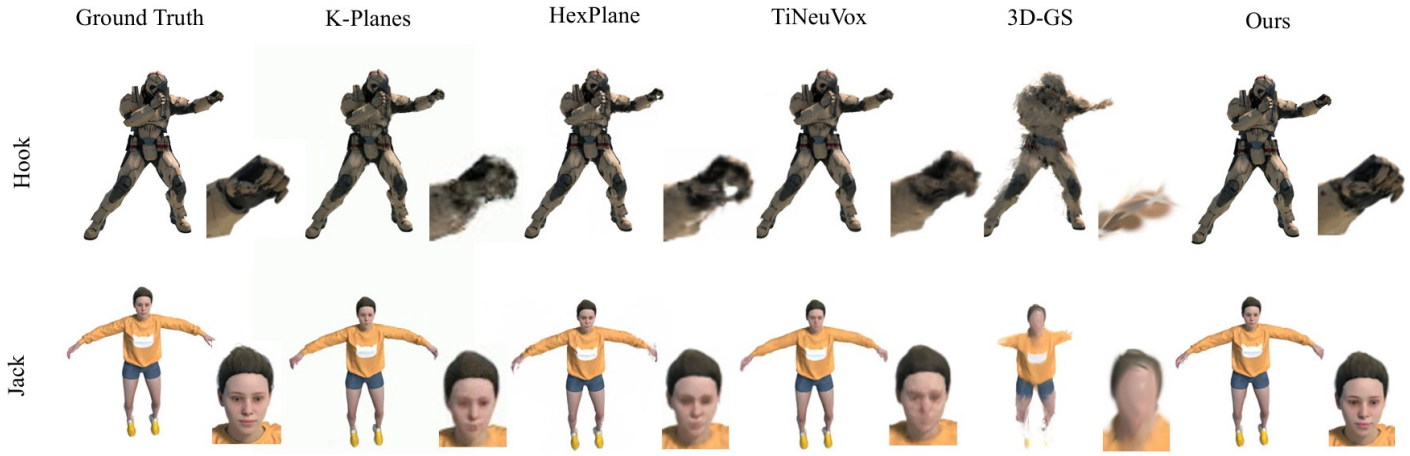
Loss function

- L1 loss
  - 원본 이미지와 rendering된 이미지 사이의 차이가 비슷하게 업데이트
- Total variational loss
  - Rendering된 이미지의 수직, 수평 옆 pixel간의 차이를 구하고 합침
  - Rendering된 이미지가 smooth하고 자연스럽게 변하도록 업데이트

Experiment

- Synthesis dataset

D-NeRF dataset



Method	Bouncing Balls			Hellwarrior			Hook			Jumpingjacks		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
3D-GS [14]	23.20	0.9591	0.0600	24.53	0.9336	0.0580	21.71	0.8876	0.1034	23.20	0.9591	0.0600
K-Planes[8]	40.05	0.9934	0.0322	24.58	0.9520	0.0824	28.12	0.9489	0.0662	31.11	0.9708	0.0468
HexPlane[4]	39.86	0.9915	0.0323	24.55	0.9443	0.0732	28.63	0.9572	0.0505	31.31	0.9729	0.0398
TiNeuVox[6]	40.23	0.9926	0.0416	27.10	0.9638	0.0768	28.63	0.9433	0.0636	33.49	0.9771	0.0408
Ours	40.62	0.9942	0.0155	28.71	0.9733	0.0369	32.73	0.9760	0.0272	35.42	0.9857	0.0128

Method	Lego			Mutant			Standup			Trex		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
3D-GS [14]	23.06	0.9290	0.0642	20.64	0.9297	0.0828	21.91	0.9301	0.0785	21.93	0.9539	0.0487
K-Planes [8]	25.49	0.9483	0.0331	32.50	0.9713	0.0362	33.10	0.9793	0.0310	30.43	0.9737	0.0343
HexPlane [4]	25.10	0.9388	0.0437	33.67	0.9802	0.0261	34.40	0.9839	0.0204	30.67	0.9749	0.0273
TiNeuVox [6]	24.65	0.9063	0.0648	30.87	0.9607	0.0474	34.61	0.9797	0.0326	31.25	0.9666	0.0478
Ours	25.03	0.9376	0.0382	37.59	0.9880	0.0167	38.11	0.9898	0.0074	34.23	0.9850	0.0131

Experiment

- Real-world dataset

HyperNeRF dataset

Method	3D Printer		Chicken		Broom		Banana	
	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM	PSNR	MS-SSIM
Nerfies [24]	20.6	0.83	26.7	0.94	19.2	0.56	22.4	0.87
HyperNeRF [25]	20.0	0.59	26.9	0.94	19.3	0.59	23.3	0.90
TiNeuVox-B [6]	22.8	0.84	28.3	0.95	21.5	0.69	24.4	0.87
FFDNeRF [12]	22.8	0.84	28.0	0.94	21.9	0.71	24.3	0.86
3D-GS [14]	18.3	0.60	19.7	0.70	20.6	0.63	20.4	0.80
Ours	22.1	0.81	28.7	0.93	22.0	0.70	28.0	0.94

DyNeRF dataset

Method	Cut Beef		Cook Spinach		Sear Steak		Flame Steak		Flame Salmon		Coffee Martini	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
NeRFPlayer [35]	31.83	0.928	32.06	0.930	32.31	0.940	27.36	0.867	26.14	0.849	32.05	0.938
HexPlane [4]	32.71	0.985	31.86	0.983	32.09	0.986	31.92	0.988	29.26	0.980	-	-
KPlanes [8]	31.82	0.966	32.60	0.966	32.52	0.974	32.39	0.970	30.44	0.953	29.99	0.953
MixVoxels [38]	31.30	0.965	31.65	0.965	31.43	0.971	31.21	0.970	29.92	0.945	29.36	0.946
Ours	32.90	0.957	32.46	0.949	32.49	0.957	32.51	0.954	29.20	0.917	27.34	0.905

Experiment

- Real-world dataset

  HyperNeRF dataset

Experiment

- Ablation

(a) Ours (b) Ours w/o ds (c) Ours w/o dr

(a) Ours w/o dx (b) Ours

(a) Ours (b) Ours w φ_{shs}, φ_o (c) TiNeuVox

Model	PSNR(dB)↑	SSIM↑	LPIPS↓	Time↓	FPS↑	Storage (MB)↓
Ours w/o HexPlane $R_l(i,j)$	27.05	0.95	0.05	10 mins	140	12
Ours w/o initialization	31.91	0.97	0.03	19 mins	79	18
Ours w/o ϕ_x	26.67	0.95	0.07	20 mins	82	17
Ours w/o ϕ_r	33.08	0.98	0.03	20 mins	83	17
Ours w/o ϕ_s	33.02	0.98	0.03	20 mins	82	17
Ours	34.05	0.98	0.02	20 mins	82	18

Limitations

- Novel view reconstruction

  Monocular setting은 input data의 camera pose, timestep에 대해 sparse하기 때문에 training view에 overfitting을 유발함

- Large motions or dramatic scene changes

  4DGS의 deformation network는 3D Gaussian들의 motion을 예측하지만 large motion의 경우 예측하지 못함

(a) Training View (b) Novel View 1 (c) Novel View 2

< IPhone dataset에 대한 novel view rendering 결과>

(a) Ours (b) Ground Truth

< Sports dataset에 대한 rendering 결과>

3D Geometry-aware Deformation Gaussian Splatting for Dynamic View Synthesis
CVPR 2024

Abstract

- Contributes

  Geometry-aware feature extraction network

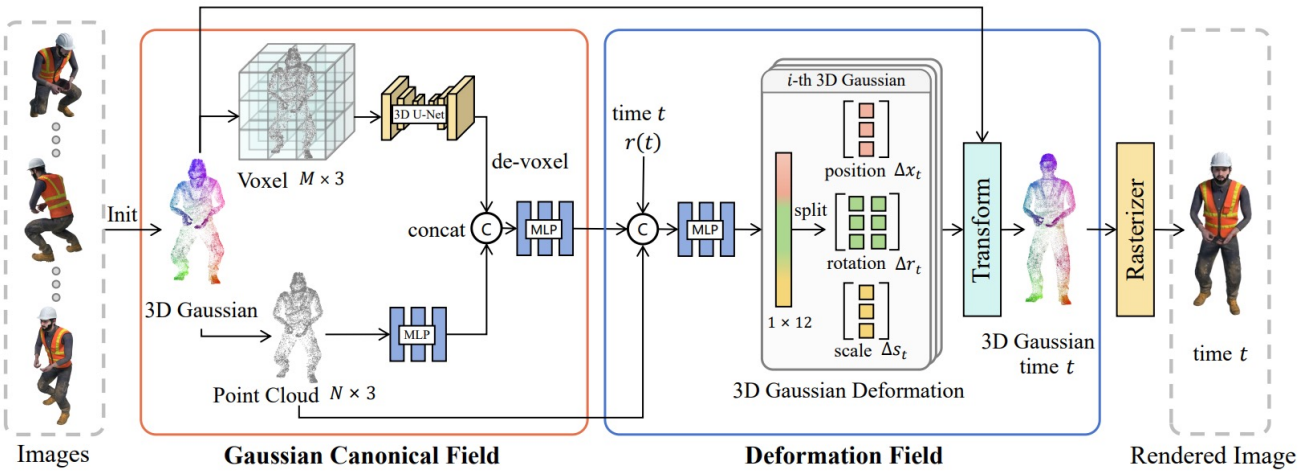
  - Voxelization과 3D-Unet을 통해 explicit하게 geometry한 feature를 추출

  Continuous 6D rotation representation

  - Smooth한 rotation의 variation을 학습하는데 도움을 줌

  Density control strategy to adapt Gaussian splatting to dynamic scenes

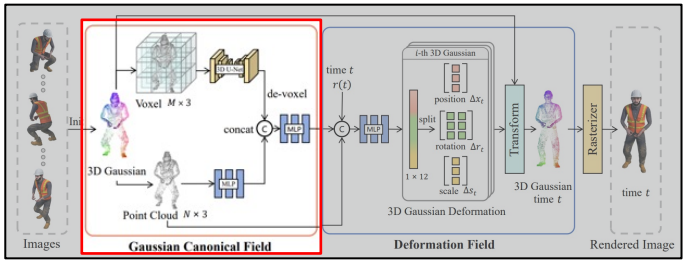
  - 현재 time의 Gaussian에 대해서 densification을 수행하여 canonical field에 Gaussian들을 효율적으로 학습함



< GAGS 전체 파이프라인 >

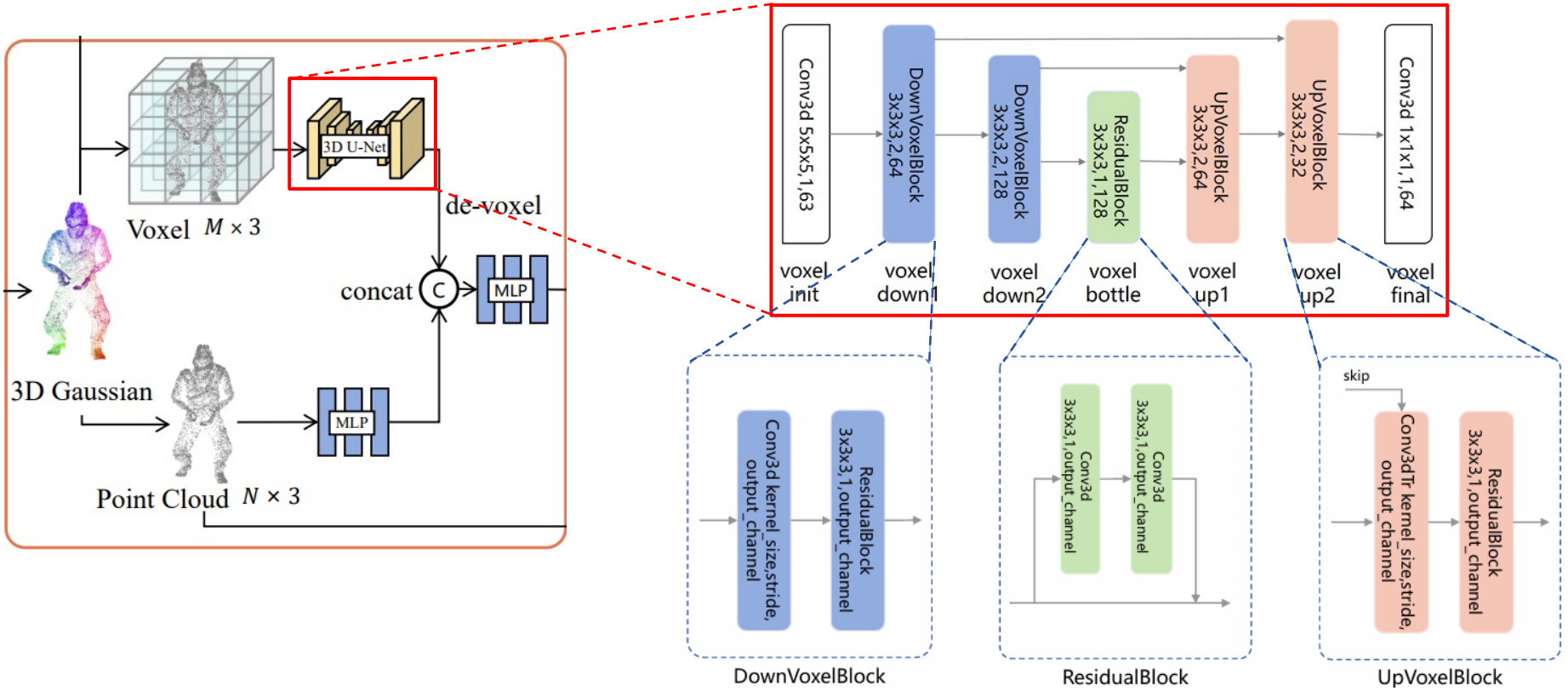
Method

- Geometry-aware feature extraction network



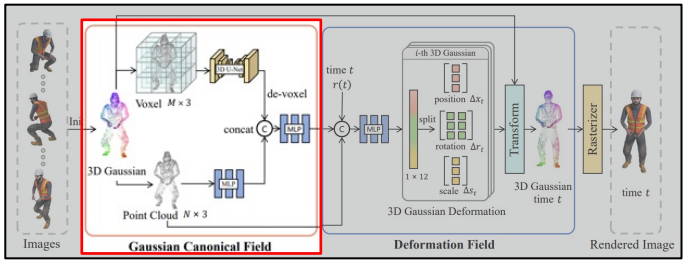
Geometry branch

- Point cloud들을 voxelization을 수행하여 3D-Unet으로 geometric한 feature를 생성



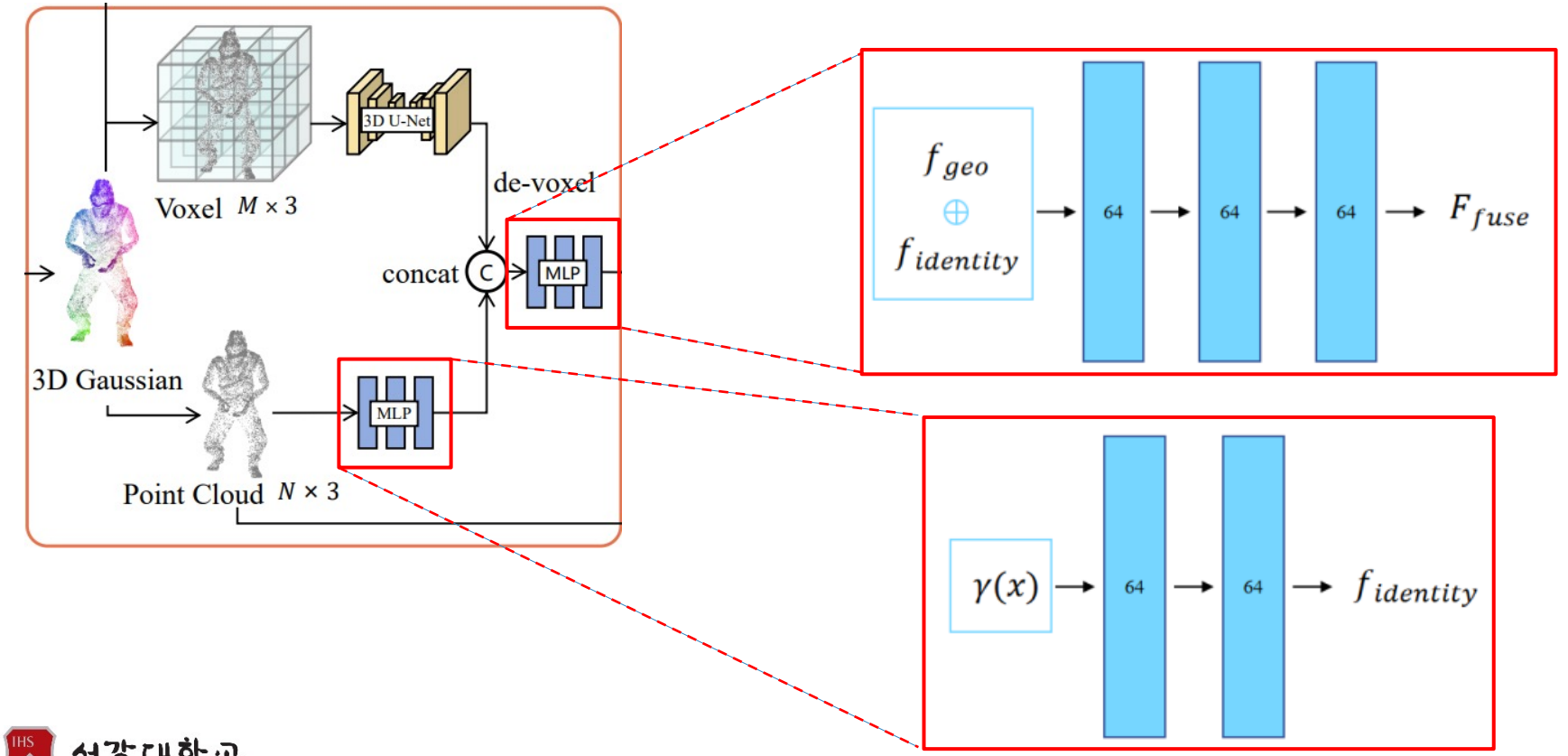
Method

- Geometry-aware feature extraction network



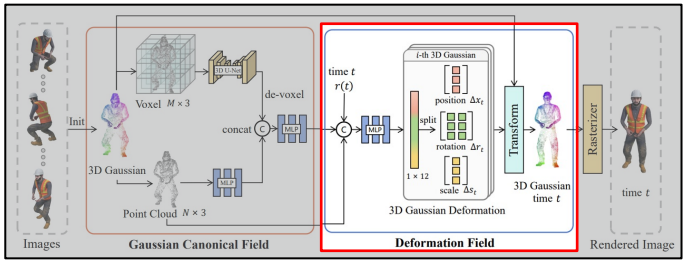
Identity branch & Fuse branch

- 기존 방식과 동일하게 MLP network로 point cloud에 대한 feature를 생성 후, geometry branch의 feature와 concatenation하여 다시 한 번 MLP network로 fuse feature를 생성



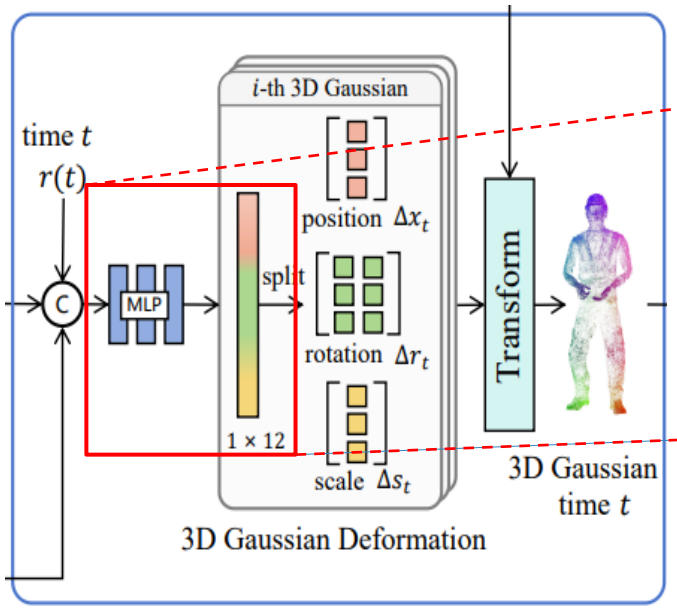
Method

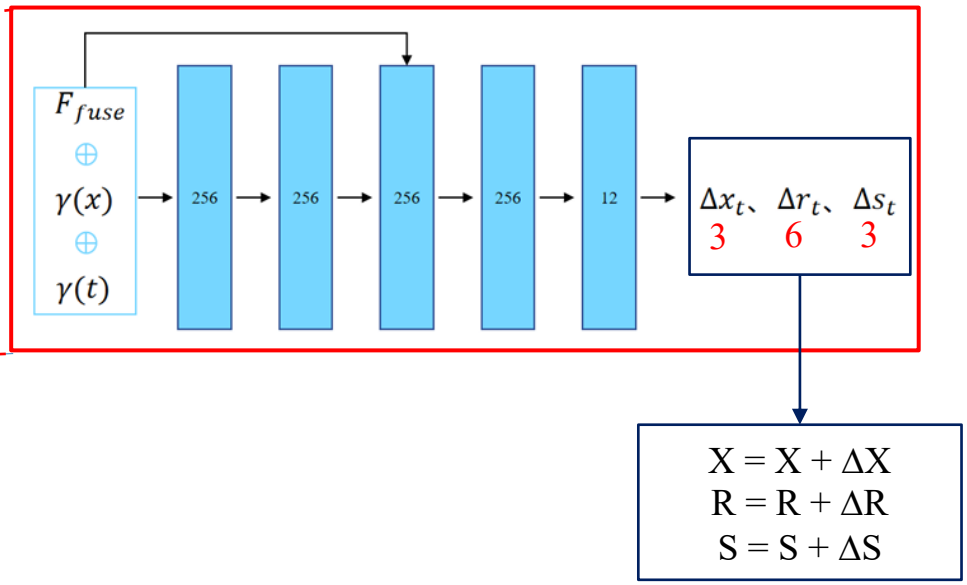
- Geometry-aware feature extraction network



Deformation field

- Canonical field에서 생성한 fuse feature, time t, point cloud를 concatenation하고 MLP network를 거쳐 canonical Gaussian의 deformation을 추정하여 update를 진행





Method

- Continuous 6D rotation representation

3D Gaussian splatting

- Quaternion을 사용하여 rotation을 representation함

Geometry-aware Gaussian splatting

- 6D rotation representation을 사용하여 deformation network가 canonical field의 Gaussian들의 smooth rotation variation을 학습하는데 도움을 줌

$$f_{\text{V2M}}\left(\begin{bmatrix} | & | \\ a_1 & a_2 \\ | & | \end{bmatrix}\right) = \begin{bmatrix} | & | & | \\ b_1 & b_2 & b_3 \\ | & | & | \end{bmatrix},$$

$$b_i = \left[\begin{cases} \mathcal{N}(a_1) & \text{if } i = 1 \\ \mathcal{N}(a_2 - (b_1 \cdot a_2)b_1) & \text{if } i = 2 \\ b_1 \times b_2 & \text{if } i = 3 \end{cases} \right]^\top,$$

Rotation matrix
6D vector
Normalization function
Transform from 6D to rotation

< 6D rotation representation 수식 >

Method

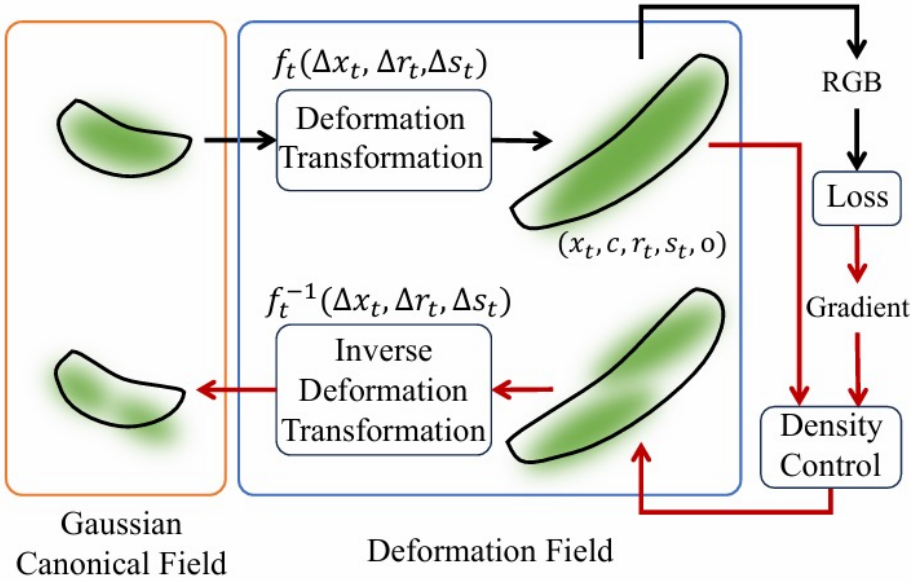
- Density control

3D Gaussian splatting

- 기존 3DGS는 gaussian의 densification과정을 time을 고려하지 않고 진행함

Geometry-aware Gaussian splatting

- Dynamic scene에서는 time을 고려하여 canonical field의 Gaussian들을 deformation field를 거쳐 현재 time에 Gaussian에 대해서 densification을 진행



< Density control 파이프라인 >

Method

- Optimization

Intialization

- 초기 3000 iterations에는 static scene에 대해서만 학습
- 이후 deformation network를 학습하여 dynamic에 대해서 학습

Loss function

- Photometric loss
  - Rendering된 image와 GT image 사이의 L_1 distance와 $L_{D\text{-}SSIM}$를 구함
- Regularization
  - Scene에서 dynamic point는 static point보다 적고 motion의 크기는 크지 않기 때문에 가능한 scene에서 point는 static이어야 함.

$$L_{photo} = (1-\lambda)L_1 + \lambda L_{D-SSIM}.$$

$$L_{motion} = \|\Delta \mathbf{x_t}\|_1 .$$

$$L = L_{photo} + \omega L_{motion},$$

Experiment

- Synthesis dataset

  D-NeRF dataset

Method	Hell Warrior			Mutant			Hook			Bouncing Balls		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM ↑	LPIPS↓
3D-GS [22]	15.3924	0.8776	0.1300	21.7554	0.9359	0.0575	18.6933	0.8733	0.1144	22.5575	0.9485	0.0647
D-NeRF [40]	25.0293	0.9506	0.0691	31.2900	0.9739	0.0268	29.2567	0.9650	0.1174	38.9300	0.9900	0.1031
TiNeuVox-B[12]	28.2058	0.9661	0.0631	33.9029	0.9771	0.0301	31.7929	0.9718	0.0436	40.8536	0.9913	0.0401
NDVG [17]	26.4933	0.9600	0.0670	34.4131	0.9801	0.0270	30.0009	0.9626	0.0463	37.5157	0.9874	0.0751
FDNeRF [18]	27.7120	0.9665	0.0508	34.9727	0.9810	0.0312	32.2867	0.9756	0.0388	40.0191	0.9912	0.0395
4D-GS [61]	28.1196	0.9730	0.0276	38.3411	0.9936	0.0062	33.1560	0.9810	0.0168	40.7418	0.9941	0.0105
Ours	**32.2712**	**0.9835**	**0.0164**	**41.4284**	**0.9969**	**0.0029**	**36.9647**	**0.9916**	**0.0076**	**43.5929**	**0.9960**	**0.0061**

Method	Lego			T-Rex			Stand Up			Jumping Jacks		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM ↑	LPIPS↓
3D-GS [22]	23.0991	0.9329	0.0567	25.7496	0.9567	0.0474	19.3779	0.9200	0.0909	20.7163	0.9227	0.0980
D-NeRF [40]	21.6427	0.8394	0.1654	31.7568	0.9767	0.0396	32.7992	0.9818	0.0215	32.8031	0.9810	0.0373
TiNeuVox-B[12]	25.1748	0.9217	0.0689	32.7750	0.9783	0.0307	36.2031	0.9859	0.0199	34.7390	0.9823	0.0328
NDVG [17]	25.0416	0.9395	0.0534	32.6229	0.9781	0.0330	33.2158	0.9793	0.0302	31.2530	0.9737	0.0398
FDNeRF [18]	25.2700	0.9390	0.0460	30.7068	0.9731	0.0368	36.9107	0.9878	0.0188	33.5521	0.9812	0.0329
4D-GS [61]	25.4024	0.9434	0.0377	33.3912	0.9869	0.0130	38.2610	0.9923	0.0071	35.6656	0.9882	0.0159
Ours	**25.4411**	**0.9474**	**0.0329**	**39.0285**	**0.9952**	**0.0052**	**42.2101**	**0.9966**	**0.0028**	**37.9604**	**0.9928**	**0.0088**

Experiment

- Real-world dataset

HyperNeRF dataset

Method	Chicken		3D Printer		Broom		Peel Banana	
	PSNR↑	MS-SSIM↑	PSNR↑	MS-SSIM↑	PSNR↑	MS-SSIM↑	PSNR↑	MS-SSIM↑
TiNeuVox[12]	28.2861	**0.9474**	22.7514	0.8392	21.2682	0.6832	24.5136	0.8743
NDVG [17]	27.0536	0.9390	22.4196	0.8389	21.4658	0.7028	22.8204	0.8279
FDNeRF [18]	27.9627	0.9438	**22.8027**	**0.8453**	**21.9091**	**0.7154**	24.2515	0.8645
3D-GS [22]	20.8915	0.7426	18.3991	0.6114	20.3953	0.6598	20.5654	0.8094
Ours	**28.5342**	0.9331	22.0403	0.8098	20.8994	0.5241	**25.5785**	**0.9067**

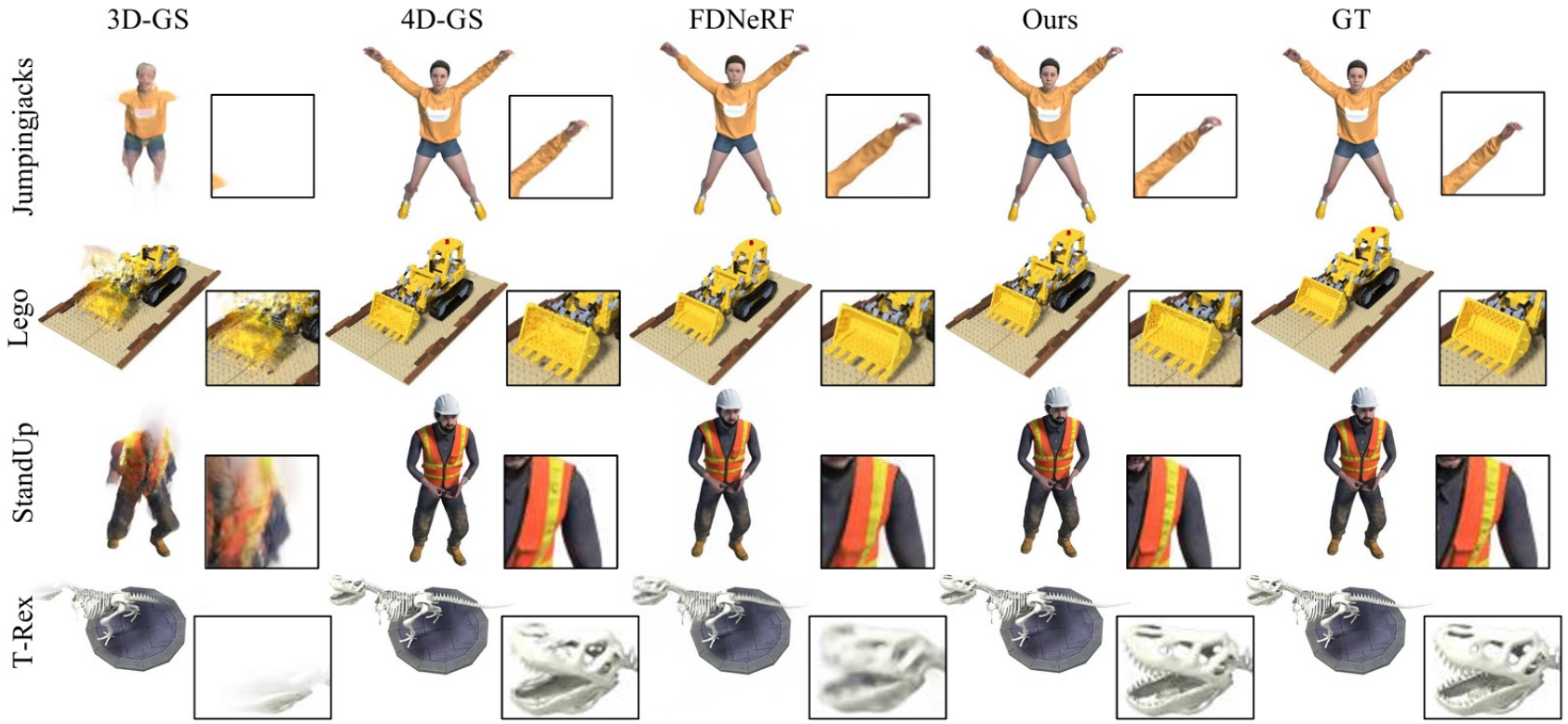
Neu3DV dataset

Scene	Cook Spinach			Cut Roast Beef			Flame Steak			Sear Steak		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
MixVoxels [59]	31.39	0.931	0.113	31.38	0.928	0.111	30.15	0.938	0.108	30.85	0.940	0.103
K-Planes [14]	31.23	0.926	0.114	31.87	0.928	0.114	31.49	0.940	0.102	30.28	0.937	0.104
Hexplanes‡ [7]	31.05	0.928	0.114	30.83	0.927	0.115	30.42	0.939	0.104	30.00	0.939	0.105
Hyperreel [2]	31.77	0.932	**0.090**	**32.25**	0.936	**0.086**	31.48	0.939	**0.083**	31.88	0.942	**0.080**
NeRFPlayer† [48]	30.58	0.929	0.113	29.35	0.908	0.144	31.93	0.950	0.088	29.13	0.908	0.138
StreamRF [24]	30.89	0.914	0.162	30.75	0.917	0.154	31.37	0.923	0.152	31.60	0.925	0.147
SWAGS [46]	**31.96**	0.946	0.094	31.84	**0.945**	0.099	**32.18**	0.953	0.087	32.21	0.950	0.092
Ours	31.39	**0.947**	0.144	29.87	0.944	0.156	31.35	**0.954**	0.129	**32.62**	**0.955**	0.130

Experiment

- Synthesis dataset

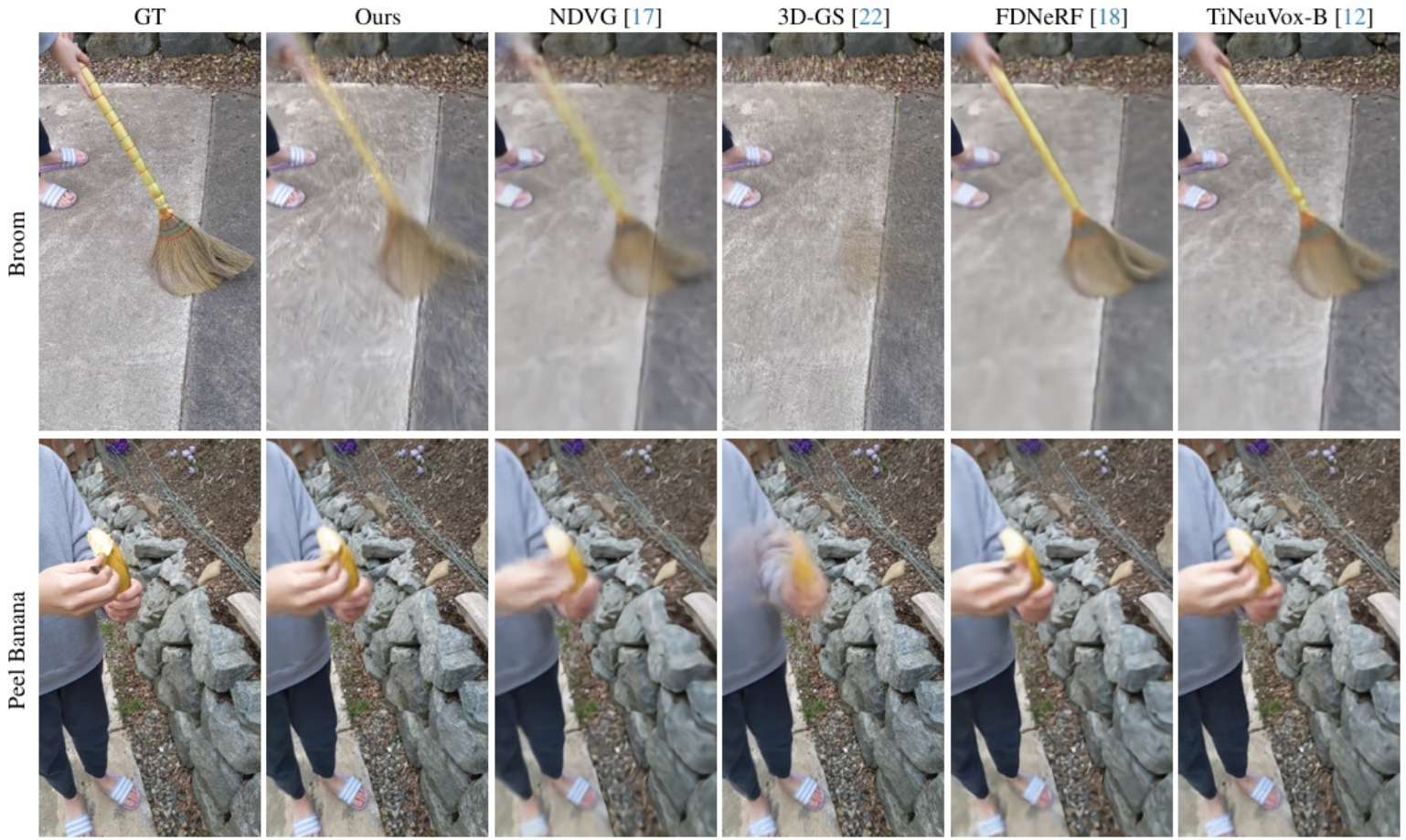
  D-NeRF dataset



Experiment

- Real-world dataset

  HyperNeRF dataset



Experiment

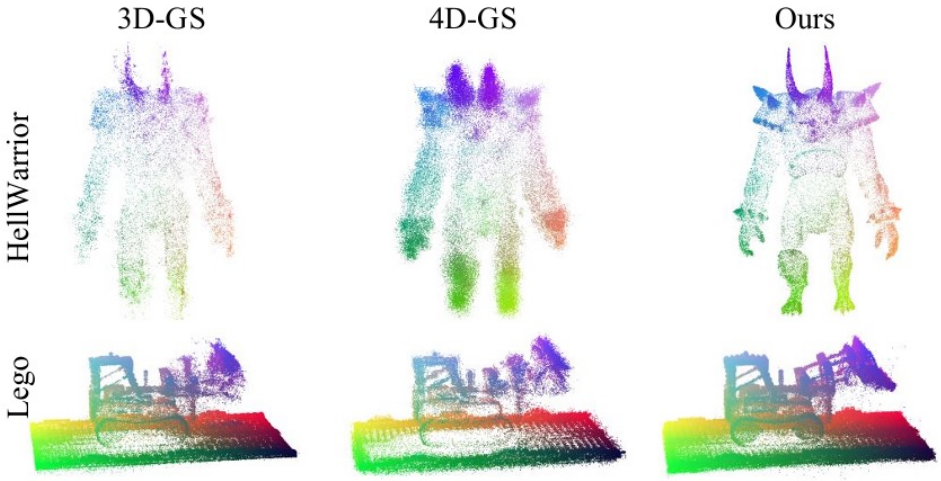
- Visualization experiment

  실험 1. 학습된 Gaussian을 visualization하여 비교

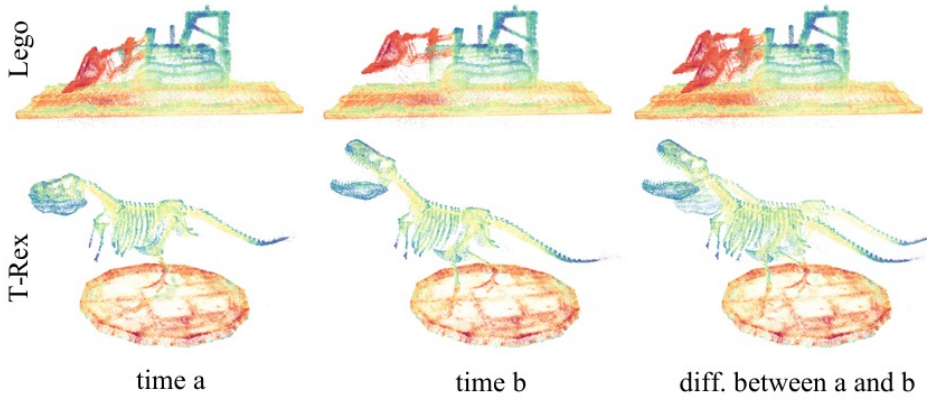
  - 3DGS는 dynamic한 부분에 대해 학습하지 못함
  - 4DGS보다 GAGS의 point들이 더욱 정교함, 이는 geometry한 정보를 학습했기 때문

  실험 2. 서로 다른 time t에 대해 geometry-aware feature를 시각화

  - Static한 부분과 dynamic한 부분의 point들이 잘 구분됨



< Gaussian 시각화 >



< Geometry-aware feature 시각화 >

감사합니다