『2025 겨울 세미나』

Dynamic 3D Gaussians

Pusan National University
Dept. of Electronics Engineering
Computer Vision & Signal Processing Lab.

Presented By
문형주

Outline

- Background
- Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis [CVPR 24]
- E-D3DGS : Embedding –Based Deformable 3D Gaussian Splatting [ECCV 24]

Background

- 3D Gaussian Splatting
  - 여러 시점에서 촬영된 이미지와 알려진 camera pose가 주여졌을 때 differentiable rasterization을 통해 3D Gaussian을 최적화하여 static 3D scene을 표현하는 기법
  - 3D Gaussian의 parameter
    - $\alpha_i = \sigma_i \exp(-\frac{1}{2}(\mathrm{x}-\mu_i)^T \sum_i^{-1}(\mathrm{x}-\mu_i))$: 임의의 공간적 지점 x에서 최종 opacity(불투명도)
    - σ_i : opacity (불투명도)
    - μ_i : 3D Gaussian i의 위치
    - Σ_i : covariance matrix
      - $\Sigma_i = R_i S_i S_i^T R_i^T$
        - ✓S_i : scaling matrix, diagonal matrix
        - ✓R_i : Rotation matrix, quaternion으로 parameterized된 matrix

Background

- 3D Gaussian Splatting
  - Projection to 2D Image Space
    - 이미지를 렌더링하기 위해 3D Gaussian들은 2D image space에 projection 됨
      - 투영변환의 approximation을 사용하여 수행 됨
  - 3D Gaussian의 projection
    - Depth 순서로 Gaussian들을 정렬한 후 , pixel에서의 color는 volume rendering을 통해 계산 됨
      - $I = \sum_{i \in \mathcal{N}} c_i \alpha_i^{2D} \prod_{j=1}^{i-1} (1 - \alpha_j^{2D})$
        - ✓α_i^{2D} : 투영된 opacity
        - ✓c_i : Spherical Harmonics(SH) 계수 h_i와 시점을 이용하여 계산된 최종 RGB color

Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis [CVPR 24]

Introduction

- Task
  - Dynamic 3D Gaussians
    - Dynamic scene에 대한 3DGS의 novel view synthesis



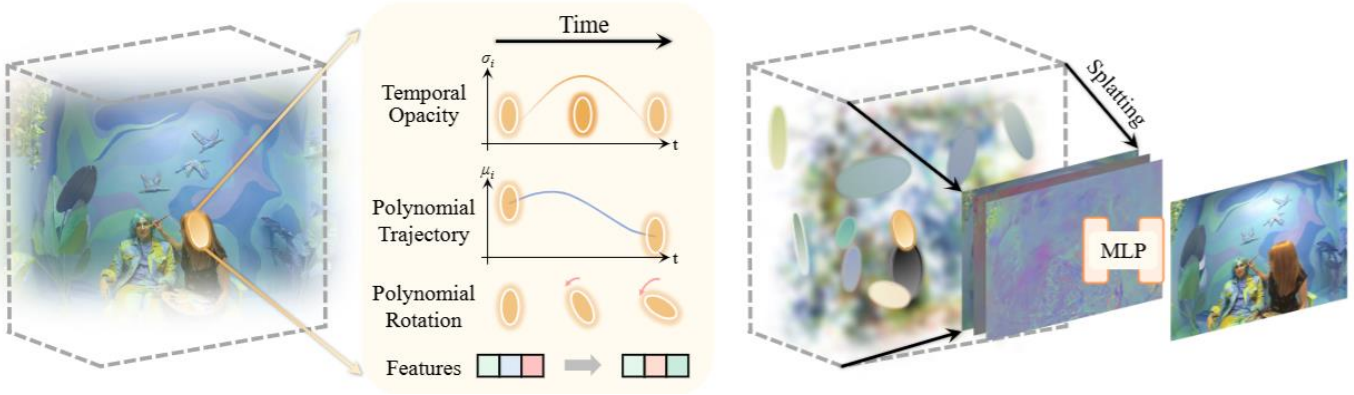
< Qualitative result of Neural 3D Video Dataset >

Introduction

- Problem & Solution
  - Existing 3DGS models
    - Dynamic scene에 대해 프레임 별로 적용하는 것은 model size와 training time 측면에서 부담이 큼
    - Grid 기반 표현 방식은 dynamic scene의 복잡한 deformation을 완전히 반영하지 못하여, 디테일한 표현력을 저해하는 한계점이 존재함
  - Proposed Method
    - Dynamic view synthesis를 위한 새로운 표현 방식을 제안 함
      - 3D Gaussian을 4D Spacetime domain으로 확장하는 방식

Method

- Overview
  - Spacetime Gaussians(STG)
    - 3D Gaussians에 시간적 요소를 추가하여 STG라는 Gaussians 개념을 제안
      - 4D dynamic scene에서 나타나거나 사라지는 content, motion 및 deformation까지 모델링할 수 있음
    - Temporal opacity를 인코딩하기 위해 temporal radial basis function을 도입
      - 비디오의 지속 시간 동안 나타나거나 사라지는 scene content를 효과적으로 모델링할 수 있음
    - 3D Gaussian의 position 및 rotation을 모델링하기 위해 time-conditioned parametric function을 사용
      - Scene 내의 motion과 deformation을 표현

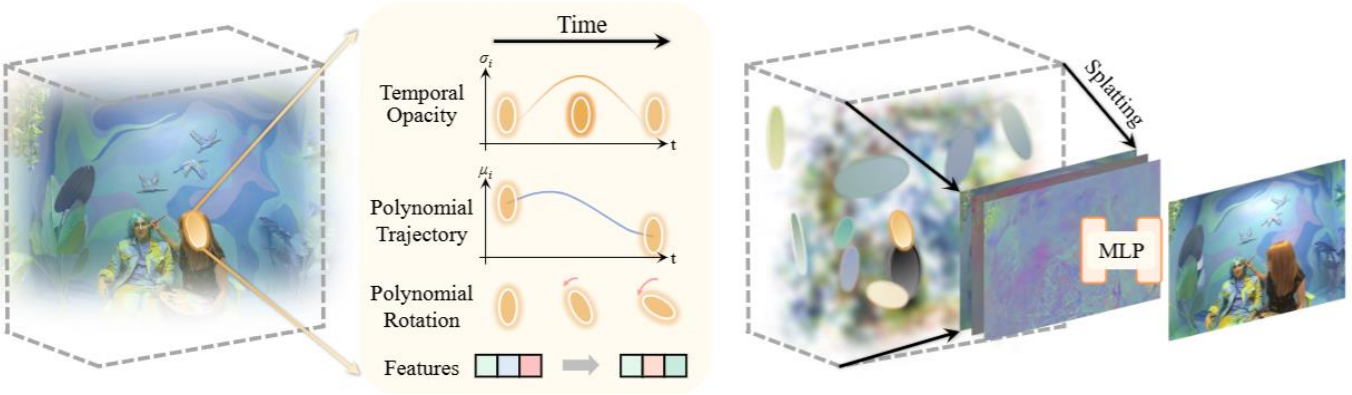


< Spacetime Gaussians> < Feature Splatting and Rendering>

< Overview of Spacetime Gaussians and splatted feature rendering>

Method

- Overview
  - Splatted Feature Rendering
    - View와 time에 dependen한 appearance modeling의 facilitate를 위해 feature-based rendering 도입
    - Model의 small size를 유지함



< Spacetime Gaussians>

< Feature Splatting and Rendering>

< Overview of Spacetime Gaussians and splatted feature rendering>

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Method

- Spacetime Gaussians
  - Spacetime Gaussians(STG)
    - Spacetime point (x, t)에서의 STG의 opacity

$$\alpha_i(t) = \sigma_i(t) \exp(-\frac{1}{2}(x - \mu_i(t))^T \sum_i(t)^{-1} (x - \mu_i(t)))$$

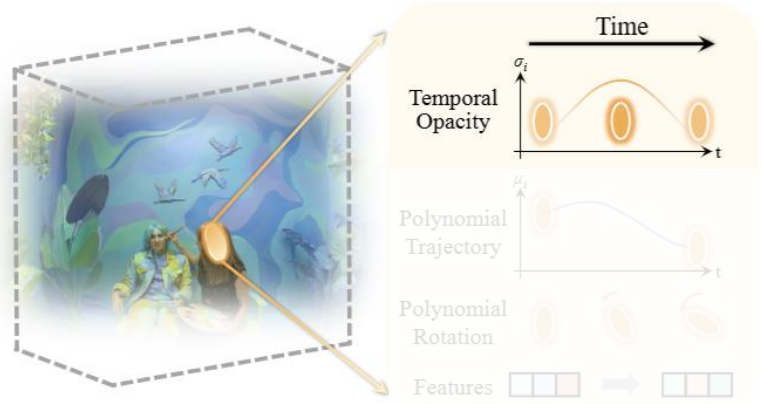
✓ $\alpha_i(t)$: temporal opacity

✓ $\mu_i(t)$: time-dependent position

✓ $\sum_i(t)$: time-dependent covariance

✓ i : i번째 STG

    - 기존 opacity 식에서 time (t) 만 추가됨



< Overview of Spacetime Gaussians>

Method

- Spacetime Gaussians
  - Temporal Radial Basis Function
    - Temporal opacity를 표현하기 위해 temporal radial basis function $\sigma_i(t)$ 사용

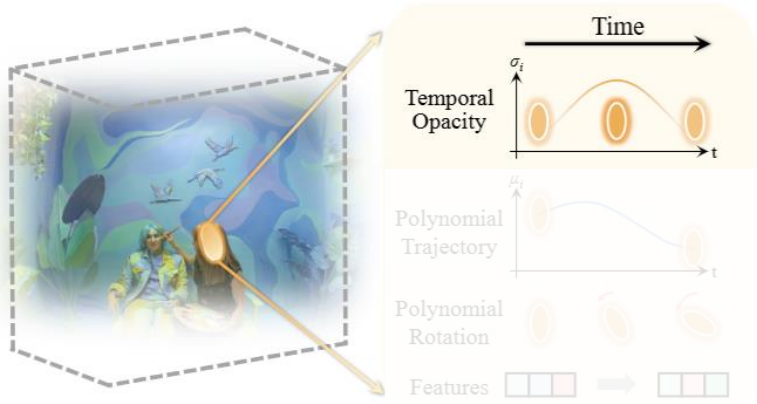
$$\sigma_i(t) = \sigma_i^s \exp(-s_i^\tau |t - \mu_i^\tau|^2)$$

✓ $\alpha_i(t) = \sigma_i(t) \exp(-\frac{1}{2}(\mathrm{x} - \mu_i(t))^T \sum_i(t)^{-1} (\mathrm{x} - \mu_i(t)))$

✓ μ_i^τ : temporal center (STG가 가장 잘 보이는 time point)

✓ s_i^τ : temporal scaling factor (STG의 effective duration = opacity가 높은 duration)

✓ σ_i^s : time-independent spatial opacity (STG 간의 spatial opacity variation을 허용하기 위해 포함)



< Overview of Spacetime Gaussians>

Method

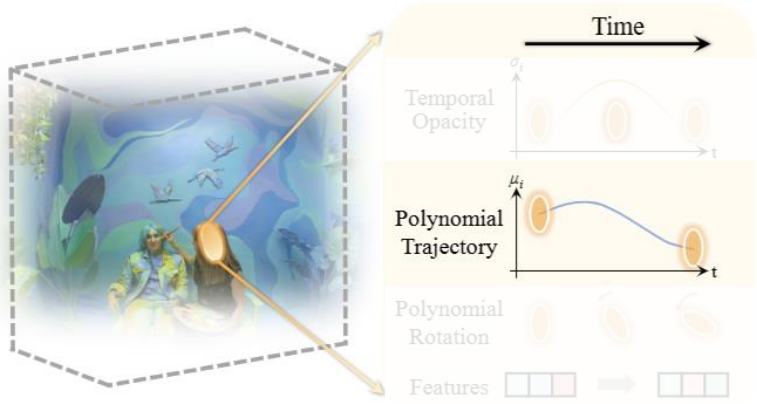
- Spacetime Gaussians
  - Polynomial Motion Trajectory
    - Time-conditioned function을 활용하여 motion 모델링

$$\mu_i(t) = \sum_{k=0}^{n_p} b_{i,k}(t-\mu_i^\tau)^k$$

✓Function으로 polynomial(다항식) 사용

✓$\mu_i(t)$: time t에서의 STG의 spatial position

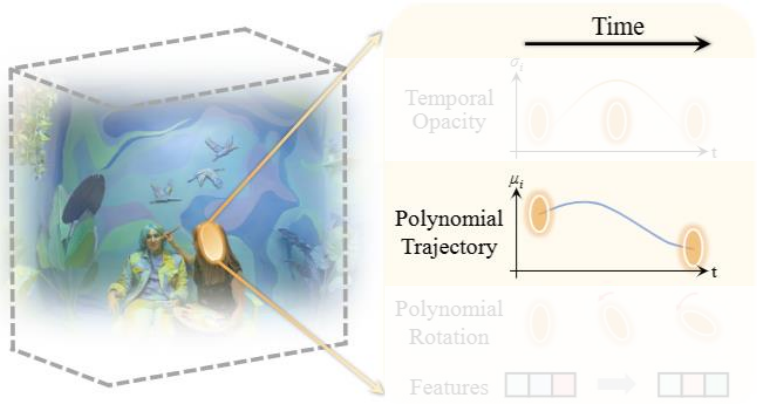
✓$\{b_{i,k}\}_{k=0}^{n_p}, b_{i,k} \in \mathbb{R}$: polynomial(다항식) 항의 계수



< Overview of Spacetime Gaussians>

Method

- Spacetime Gaussians
  - Polynomial Motion Trajectory
    - Time-conditioned function을 활용하여 motion 모델링
      - $\mu_i(t) = \sum_{k=0}^{n_p} b_{i,k}(t-\mu_i^\tau)^k$ -------- time t에서의 STG의 spatial position
      - $\sigma_i(t) = \sigma_i^s \exp(-s_i^T|t-\mu_i^\tau|^2)$ -------- temporal radial basis function
      - 두 식을 조합하여 복잡하고 긴 움직임을 여러 개의 짧고 단순한 움직임 segments로 표현 가능



< Overview of Spacetime Gaussians>

Method

- Spacetime Gaussians
  - Polynomial Rotation
    - Rotation matrix를 매개변수화하는 quaternion을 polynomial(다항식)으로 표현함

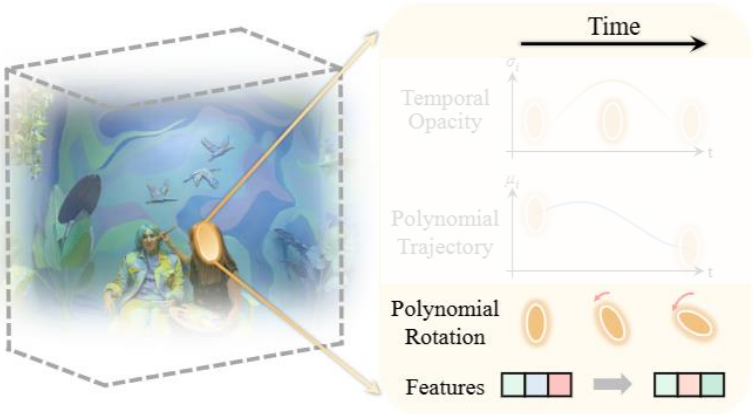
$q_i(t) = \sum_{k=0}^{n_q} c_{i,k}(t - \mu_i^\tau)^k$

✓ $q_i(t)$: time t에서 STG의 rotation (in quaternion)

✓ $\{c_{i,k}\}_{k=0}^{n_q}, c_{i,k} \in \mathbb{R}$: polynomial 계수

Rotation matrix를 표현하는 quaternion을 time t에 대해 표현함으로써 time t에 대한 covariance matrix를 얻을 수 있음

✓ $\sum_i(t) = R_i(t)S_iS_i^TR_i^T(t)$



< Overview of Spacetime Gaussians>

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Method

- Splatted Feature Rendering
  - Feature
    - View 및 time에 따라 변화하는 radiance를 정확하게 인코딩하기 위해 Spherical Harmonics(SH)대신 feature 저장

$$f_i(t) = \left[f_i^{base}, f_i^{dir}, \left(t - \mu_i^T\right) f_i^{time}\right]^T$$

✓ $f_i(t)$: 각 STG의 feature

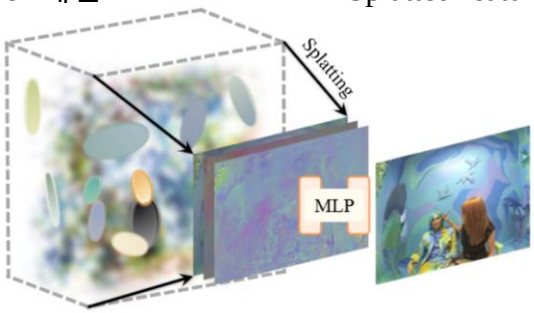
✓ $f_i^{base} \in \mathbb{R}^3$: 기본 RGB 색상을 포함하는 feature

✓ $f_i^{dir}, f_i^{time} \in \mathbb{R}^3$: view direction(시점) 및 time과 관련된 정보 인코딩

$$\mathrm{I} = \sum_{i\in} c_i \alpha_i^{2D} \prod_{j=1}^{i-1} (1 - \alpha_j^{2D}) \quad \longrightarrow \quad \mathrm{I} = \sum_{i\in} f_i(t) \alpha_i^{2D} \prod_{j=1}^{i-1} (1 - \alpha_j^{2D})$$

< 기존 image pixel 별 color 계산>

< Splatted feature를 사용한 image pixel 별 color 계산>



< Overview of splatted feature rendering>

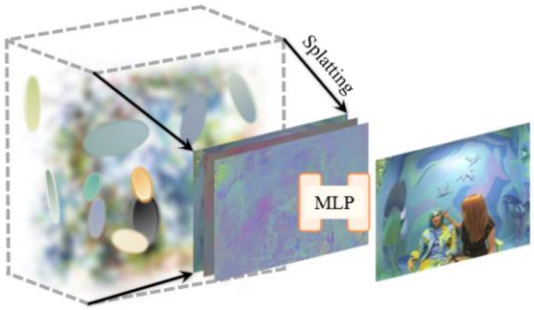
부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Method

- Splatted Feature Rendering
  - Feature rendering
    - Image space로 splatting한 후 각 픽셀의 splatting된 특징을 $F^{base}, F^{dir}, F^{time}$으로 분리
    - 최종 RGB색상은 2-layer MLP Φ 를 거쳐 얻어짐

      $I = F^{base} + \Phi(F^{dir}, F^{time}, r)$

      ✓r : 픽셀의 *view direction, 입력으로 추가됨*
    - 기존 SH 인코딩에 비해 feature-based 방식이 parameter 개수가 훨씬 적음
    - MLP가 shallow하고 narrow해서, 빠른 렌더링 속도 달성



< Overview of splatted feature rendering>

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Experiments

- Real-world benchmarks dataset
  - Neural 3D Video Dataset
    - 6 indoor multi-view sequences captured by 18 to 21 cameras at 2704 × 2028 resolution
    - Training and Evaluation with half resolution
    - 300 frame
  - Google Immersive Dataset
    - Indoor and outdoor scenes captured with a 46-camera
    - 7 selected scenes (Welder, Flames, Truck, Exhibit, Face Paint 1, Face Paint 2,Cave)
    - Center camera as test view
  - Technicolor Dataset
    - 4 × 4 camera array (time-synchronized)
    - Spatial resolution 2048 × 1088
    - 5 selected scenes (Birthday, Fabien,Painter, Theater, Trains)

< Holynski et al 데이터셋 구성>

Experiments

- Quantiative Results
  - Neural 3D Video Dataset
    - Metric
      - PSNR, $DSSIM_1$, $DSSIM_2$, LPIPS, FPS, Size

Method	PSNR↑	$DSSIM_1$↓	$DSSIM_2$↓	LPIPS↓	FPS↑	Size↓
Neural Volumes [62] [1]	22.80	-	0.062	0.295	-	-
LLFF [65] [1]	23.24	-	0.076	0.235	-	-
DyNeRF [48] [1]	**29.58**	-	**0.020**	0.083	0.015	**28 MB**
Ours [1]	29.48	0.038	0.022	**0.063**	**103**	300 MB
HexPlane [12] [2]	31.71	-	-	0.075	-	200 MB
Ours [2]	**32.74**	0.027	0.012	**0.039**	**140**	**190 MB**
StreamRF [47]	28.26	-	-	-	10.9	5310 MB
NeRFPlayer [80]	30.69	0.034	-	0.111	0.05	5130 MB
HyperReel [4]	31.10	0.036	-	0.096	2	360 MB
K-Planes [28]	31.63	-	0.018	-	0.3	311 MB
MixVoxels-L [87]	31.34	-	0.017	0.096	37.7	500 MB
MixVoxels-X [87]	31.73	-	0.015	0.064	4.6	500 MB
Ours	**32.05**	**0.026**	**0.014**	**0.044**	**140**	**200 MB**

< Quantitative comparisons on the Neural 3D Video Dataset>

  - Google Immersive Dataset

Method	PSNR↑	$DSSIM_1$↓	LPIPS↓	FPS↑	Size/Fr↓
NeRFPlayer [80]	25.8	0.076	0.196	0.12	17.1 MB
HyperReel [4]	28.8	0.063	0.193	4	**1.2 MB**
Ours	**29.2**	**0.042**	**0.081**	**99**	**1.2 MB**

< Quantitative comparisons on the Google Immersive Dataset>

Experiments

- Quantiative Results
  - Technicolor Dataset
    - Metric
      - PSNR, $DSSIM_1$, $DSSIM_2$, LPIPS, FPS, Size/Fr

Method	PSNR↑	$DSSIM_1$↓	$DSSIM_2$↓	LPIPS↓	FPS↑	Size/Fr↓
DyNeRF [48]	31.8	-	0.021	0.140	0.02	**0.6 MB**
HyperReel [4]	32.7	0.047	-	0.109	4.00	1.2 MB
Ours	**33.6**	**0.040**	**0.019**	**0.084**	**86.7**	1.1 MB

< Quantitative comparisons on the Technicolor Dataset>

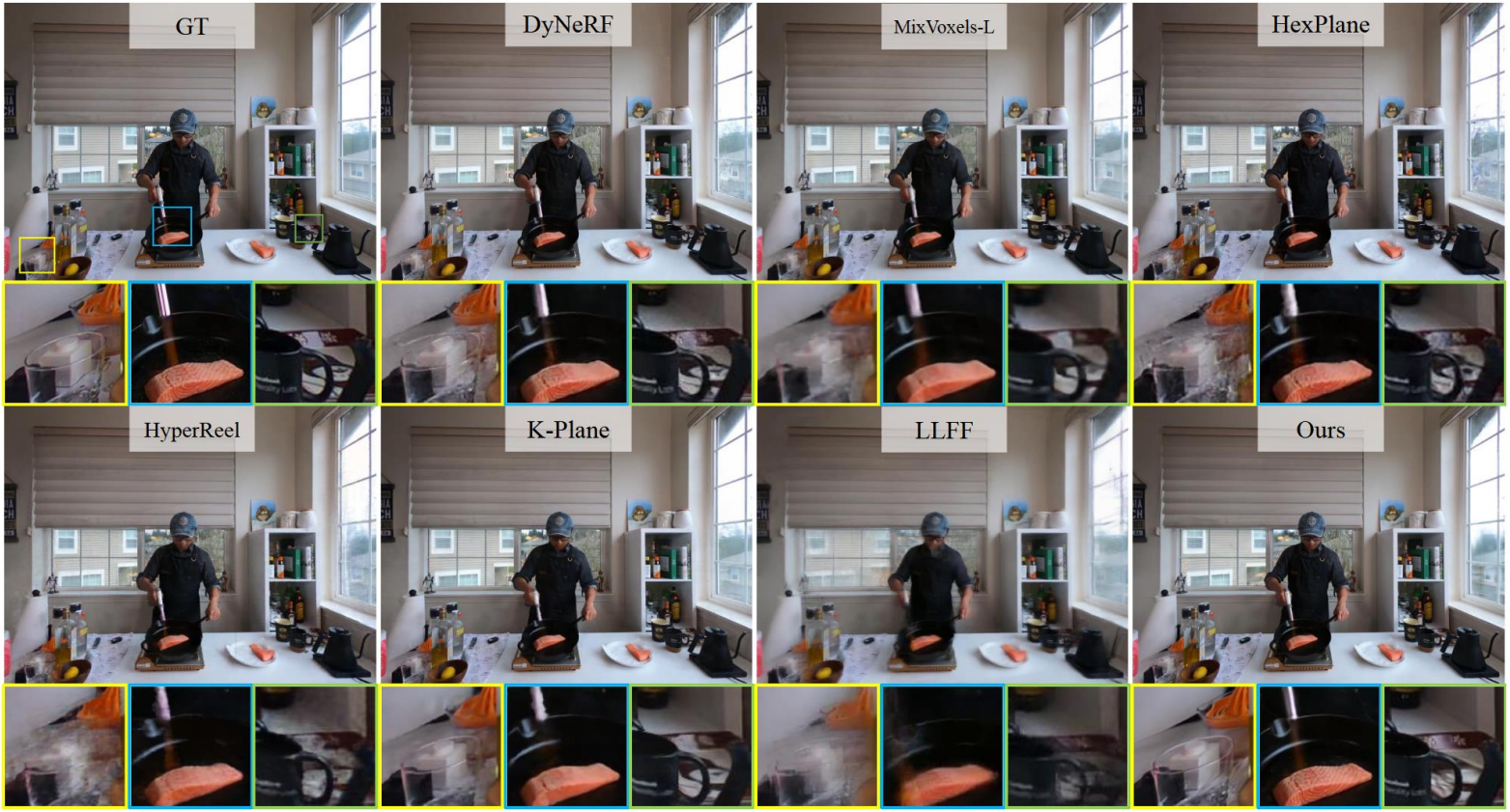
  - Ablation study of proposed components (Technicolor Dataset)

Method	PSNR↑	$DSSIM_1$↓	LPIPS↓
w/o Temporal Opacity	31.0	0.063	0.153
w/o Polynomial Motion	32.6	0.045	0.099
w/o Polynomial Rotation	33.4	0.042	0.085
w/o Feature Splatting	33.0	0.044	0.097
w/o Guided Sampling of Gaussians	33.3	0.041	0.085
Ours Full	**33.6**	**0.040**	**0.084**

< Ablation study of proposed components>

Experiments

- Qualtative Results (Neural 3D Video Dataset)



< Qualitative comparisons on the Neural 3D Video Dataset >

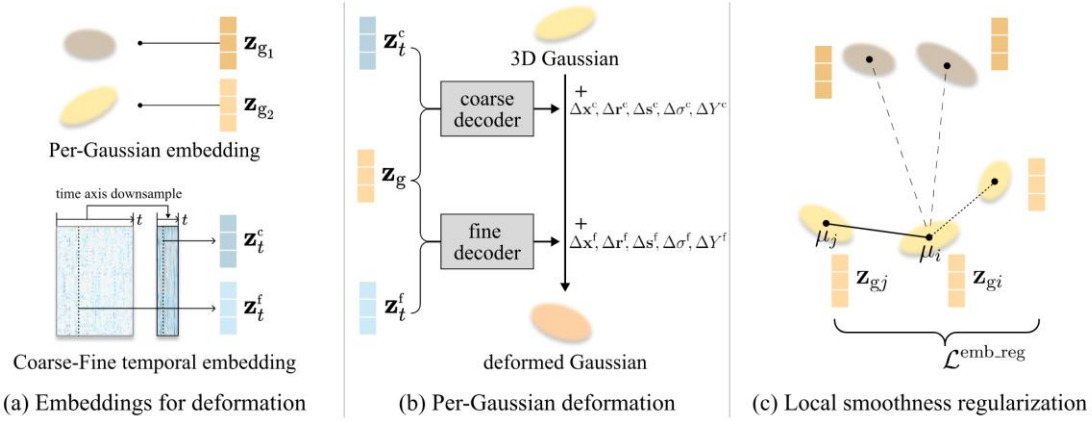
E-D3DGS : Embedding-Based Deformable 3D Gaussian Splatting [ECCV 24]

Introduction

- Problem & Solution
  - Existing 3DGS models
    - 기존 3DGS mixture of multiple volumetric fields이므로, single coordinate 기반 network를 사용하여 Gaussian parameters의 deformation을 모델링하는 것은 적절하지 않음
      - 복잡한 dynamic scene을 제대로 표현하지 못 함
  - Proposed method
    - Per-Gaussian embeddings을 통해 Gaussian마다 서로 다른 deformation을 정밀하게 모델링할 수 있도록 함

Method

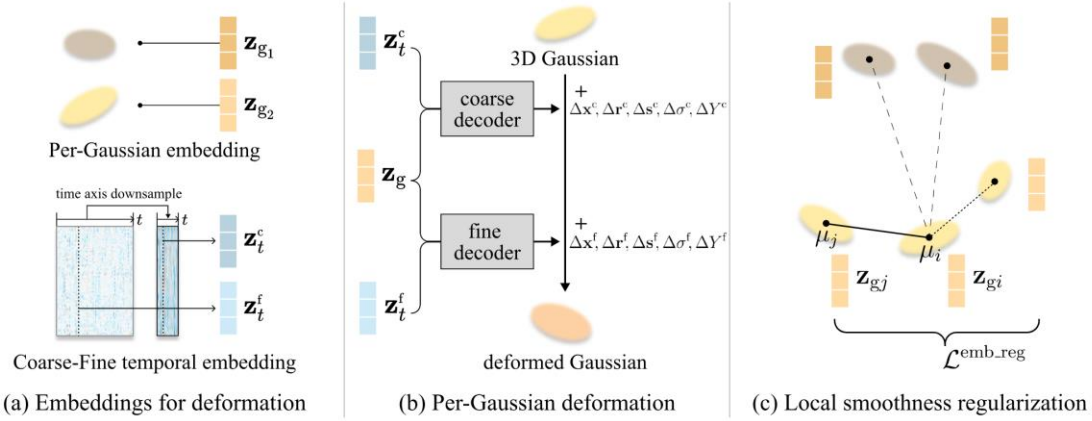
- Overview
  - Embedding-Based Deformation for Gaussians
    - Gaussian 별로 deformation하기 위해 individual Gaussians에 대한 embedding 사용
  - Coarse-Fine Deformation
    - Scene에는 복잡하고 빠른 motion과 단순하고 느린 motion이 모두 존재함
    - 두 가지 각각을 다르게 deformation하여 여러 motion들에 대해 잘 표현하도록 함



< E-D3DGS Framework>

Method

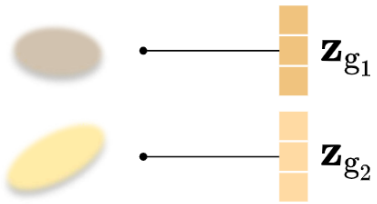
- Overview
  - Local Smoothness Regularization
    - Dynamic objects를 구성하는 Gaussian들은 유사한 deformation을 가지는 경향이 있음
    - Per-Gaussian embedding을 통한 deformation에 가까운 Gaussian들이 유사한 deformation을 가지도록 유도하는 loss term을 추가함



< E-D3DGS Framework>

Method

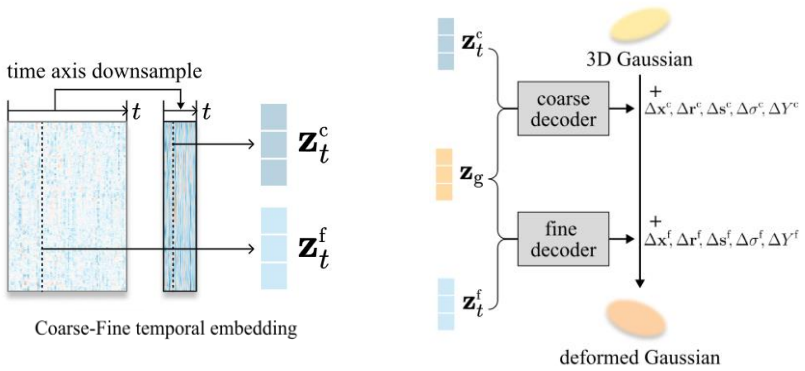
- Embedding-Based Deformation for Gaussians
  - Per-Gaussian embedding
    - Gaussian별로 deformation하기 위해 embedding 진행
      - $z_g \in \mathbb{R}^{32}$: individual Gaussians에 대한 learnable한 embeddings
      - $z_t \in \mathbb{R}^{256}$: typical temporl embddings for different frames.
    - Embedding deformation을 구하기 위해 function $\mathcal{F}_\theta$ 사용
      - $\mathcal{F}_\theta : (z_g, z_t) \rightarrow (\Delta\mathbf{x}, \Delta\mathbf{r}, \Delta\boldsymbol{s}, \Delta\sigma, \Delta Y)$
        - ✓$\mathbf{r}$: rotation quaternion
        - ✓$\boldsymbol{s}$: vector for scaling
        - ✓Y : SH 계수 for view-dependent color
        - ✓$\mathcal{F}_\theta$는 shallow MLP로 각각의 parameter에 대한 head(출력)를 다르게 둠
    - Frame t에서 Gaussian parameters는 canonical gaussian parameters + $\mathcal{F}_\theta(z_g, z_t)$로 결정됨



< Per Gaussian Embedding>

Method

- Coarse-Fine Deformation
  - Coarse-Fine Deformation
    - Coarse-fine deformation은 같은 architecture를 가지는 2 가지 function 사용
      - 각각이 coarse와 fine deformation을 담당함
    - Fine deformation
      - Temporal embedding $\boldsymbol{z}_t^f = interp(Z, t)$
        - ✓$Z \in \mathbb{R}^{N\times 256}$: N frame에 대한 1D feature grid
    - Coarse deformation
      - 빠르고 디테일한 deformation을 유발하는 High-frequencies를 제거하기 위해 Z를 5배 downsampling한 후 interpolation을 통해 $\boldsymbol{z}_t^c$ 계산

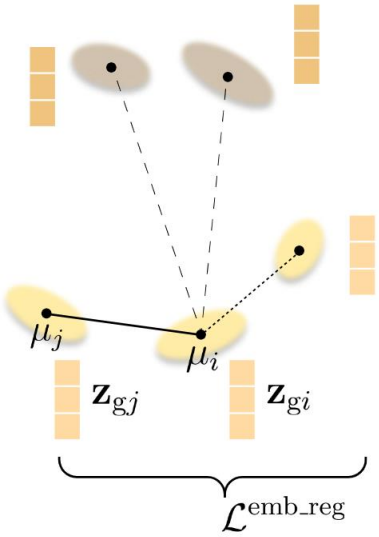


< LucidDreamer의 Point cloud construction >

부산대학교 PUSAN NATIONAL UNIVERSITY
CVSP Lab Computer Vision & Signal Processing

Method

- Local Smoothness Regularization
  - Local Smoothness Regularization for per gaussian embedding
    - 인접한 Gaussian i와 j 사이의 deformation 차이를 최소화하는 정규화 loss
    - $\mathcal{L}_{emb_reg} = \frac{1}{k|S|}\sum_{i\in S}\sum_{j\in \mathrm{KNN}_{i,k}}(w_{i,j}\left\|\boldsymbol{z}_{g_i} - \boldsymbol{z}_{g_j}\right\|_2)$
      - $w_{i,j} = \exp(-\lambda_w\left\|\mu_j - \mu_i\right\|_2^2)$: Gaussian 중심간의 거리를 기반으로 하는 weighting factor



< Local smoothness regularization>

Experiment

- Dataset
  - Neural 3D Video Dataset
    - 6 indoor multi-view sequences captured by 18 to 21 cameras at 2704 × 2028 resolution
    - Training and Evaluation with half resolution
    - 300 frame
  - Technicolor Dataset
    - 4 × 4 camera array (time-synchronized)
    - Spatial resolution 2048 × 1088
    - 5 selected scenes (Birthday, Fabien,Painter, Theater, Trains)

Experiment

- Quantiative Results
  - Neural 3D Video Dataset
    - Metric
      - PSNR, SSIM, LPIPS, Training time, FPS, Model size

model	metric			computational cost		
	PSNR↑	SSIM↑	LPIPS↓	Training time↓	FPS↑	Model size↓
DyNeRF[12] [12]	29.58	-	0.083	1344 hours	0.01	56 MB
NeRFPlayer[13] [24]	30.69	-	0.111	6 hours	0.05	1654 MB
MixVoxels [27]	30.30	0.918	0.127	1 hours 40 mins	0.93	512 MB
K-Planes [6]	30.86	0.939	0.096	1 hours 40 mins	0.13	309 MB
HyperReel[4] [1]	30.37	0.921	0.106	9 hours 20 mins	1.04	1362 MB
4DGS [30]	31.19	0.940	0.051	9 hours 30 mins	33.7	8700 MB
4DGaussians [29]	30.71	0.935	0.056	50 mins	51.9	59 MB
Ours	**31.31**	**0.945**	**0.037**	1 hours 52 mins	74.5	35 MB

< Quantitative comparisons on the Neural 3D Video Dataset>

- Technicolor Dataset

model	metric			computational cost		
	PSNR↑	SSIM↑	LPIPS↓	Training time↓	FPS↑	Model size↓
DyNeRF	31.80	-	0.140	-	0.02	0.6 MB
HyperReel	32.32	0.899	0.118	2 hours 45 mins	0.91	289 MB
4DGaussians	29.62	0.844	0.176	25 mins	34.8	51 MB
Ours	**33.24**	**0.907**	**0.100**	2 hours 55 mins	60.8	77 MB

< Quantitative comparisons on the Technicolor Dataset>

Experiment

- Quantiative Results
  - Ablation study
    - Metric
      - PSNR, SSIM, LPIPS

Method	PSNR↑	SSIM↑	LPIPS↓
Ours	**29.70**	**0.933**	**0.041**
coarse deformation only	29.48	0.931	0.044
fine deformation only	29.23	0.932	0.043
Ours + **x** injection	29.60	0.931	0.045

< Quantitative ablation results on coarse-fine deformation>

Method	PSNR↑	SSIM↑	LPIPS↓
Ours w/o embedding reg	32.26	0.951	0.037
+ our embedding reg	**32.34**	**0.952**	**0.036**
+ physically-based reg	32.08	0.950	0.036

< Quantitative ablation results on local smoothness regularization>

supplementary

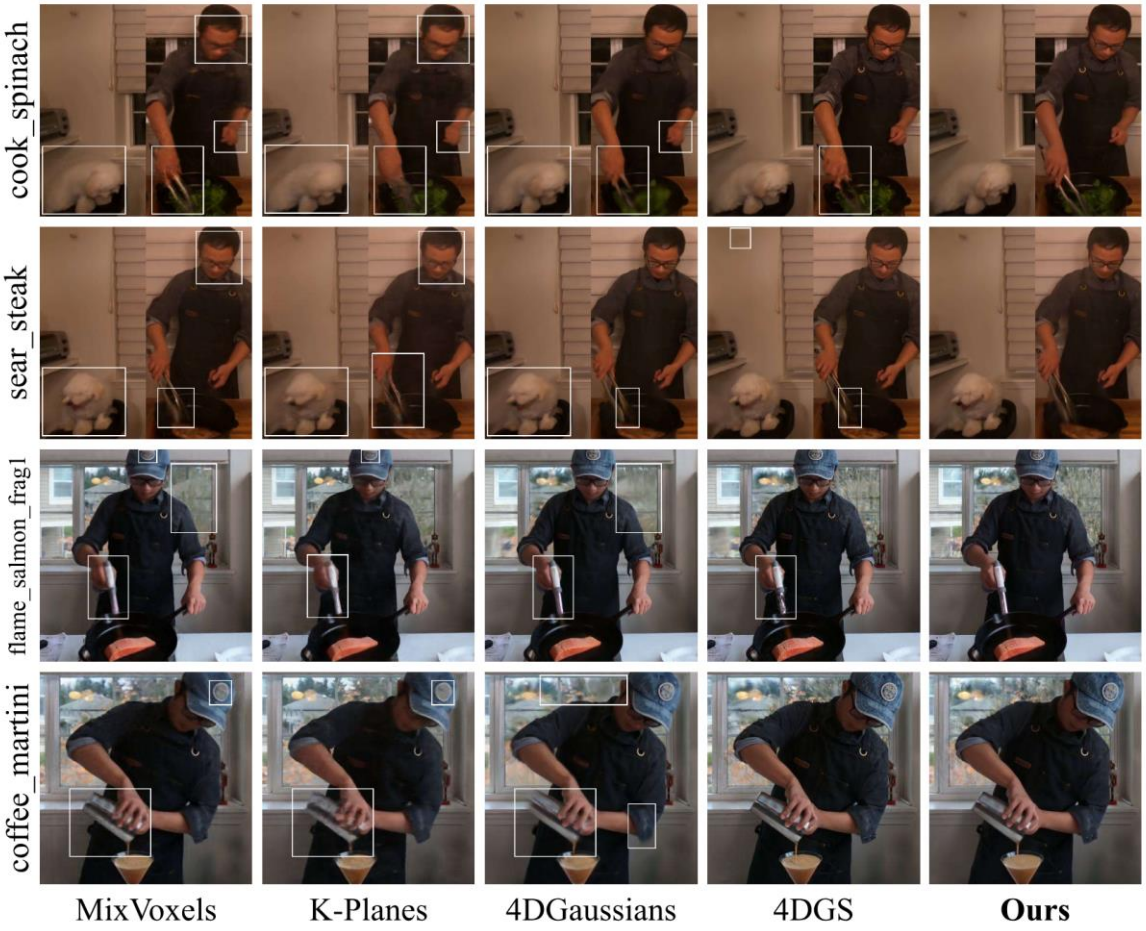
- Q1 Spacetime 논문 metric DSSIM_1과 DSSIM_2의 차이(Spacetime)
  - DSSIM 측정시 structural_similarity function from scikit-image library를 사용하였는데 DSSIM_1 은 data_range 1.0으로 설정/ DSSIM_2 는 data_range 2.0으로 설정하여 두 카테고리로 측정
- Q2 $\mathrm{I} = \mathrm{F}^{\mathrm{base}} + \Phi(\mathrm{F}^{\mathrm{dir}}, \mathrm{F}^{\mathrm{time}}, \mathrm{r})$에 입력되는 r (Spacetime)
  - Image × 6 channel의 ray H × W 값이 입력됨
- Q3 $\mathrm{I} = \mathrm{F}^{\mathrm{base}} + \Phi(\mathrm{F}^{\mathrm{dir}}, \mathrm{F}^{\mathrm{time}}, \mathrm{r})$의 Φ (Spacetime)
  - MLP Φ는 featur를 입력으로 받아 Image로 변환하는 MLP
- Q4 $f_i(t) = \left[f_i^{base}, f_i^{dir}, \left(t - \mu_i^T\right) f_i^{time}\right]^T$ (Spacetime)
  - f_i featur들은 learnable한 paramete로 네트워크를 통해 추출

supplementary

- Q5 quaternion $q_i(t) = \sum_{k=0}^{n_q} c_{i,k}(t - \mu_i^\tau)^k$ (Spacetime)
  - μ_i^τ가 point cloud의 좌표값이 아닌 STG가 가장 잘 보이는 time point로 time값을 가짐
- Q6 Polynomial trajector를 구할 때 따로 optical flow나 scene flow같은 prior 정보를 이용한 추가적인 regularization term이 있지 않음

Experiment

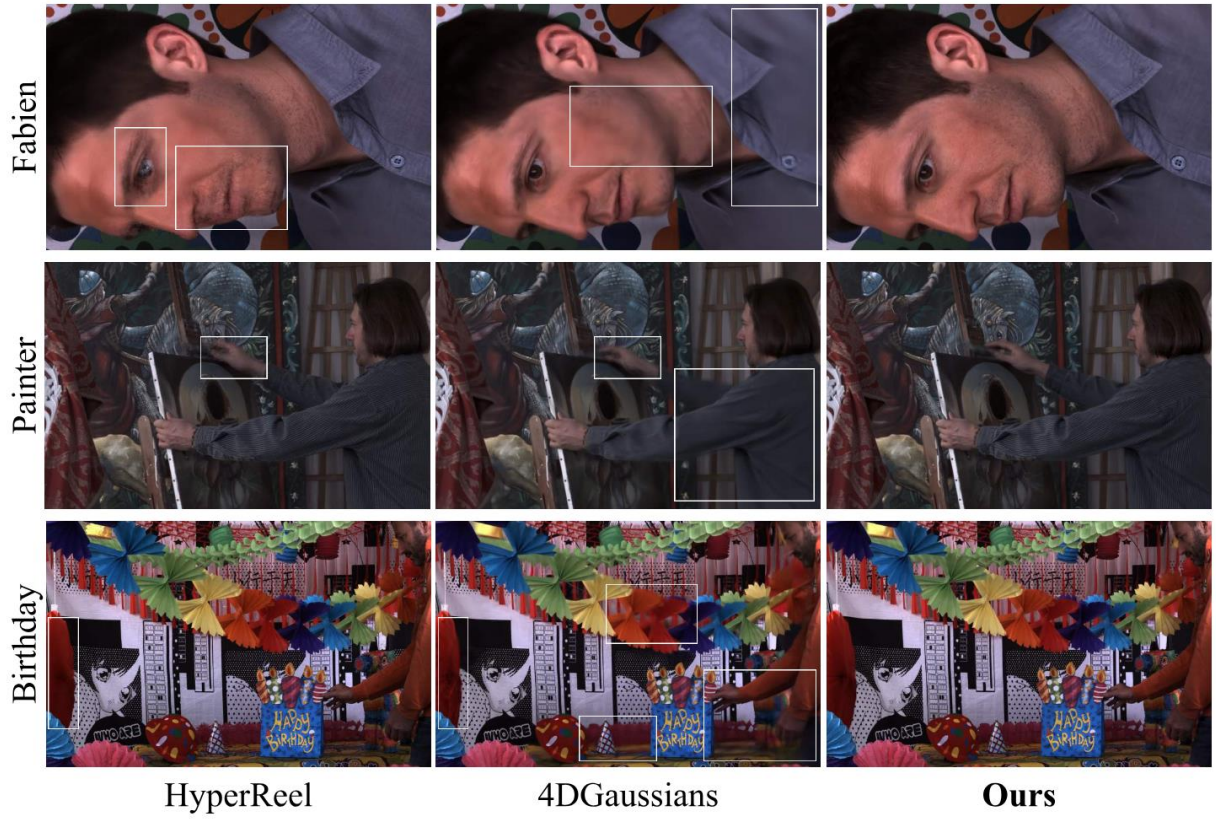
- Qualtative Results (Neural 3D Video Dataset)



< Qualitative comparisons on the Neural 3D Video dataset>

Experiment

- Qualtative Results (Technicolor Dataset)



< Qualitative comparisons on the Technicolor dataset>

감사합니다

supplementary

- Q1 Spacetime 논문 metric DSSIM_1과 DSSIM_2의 차이(Spacetime)
  - DSSIM 측정시 structural_similarity function from scikit-image library를 사용하였는데 DSSIM_1 은 data_range 1.0으로 설정/ DSSIM_2 는 data_range 2.0으로 설정하여 두 카테고리로 측정
- Q2 $\mathrm{I} = \mathrm{F}^{\mathrm{base}} + \Phi(\mathrm{F}^{\mathrm{dir}}, \mathrm{F}^{\mathrm{time}}, \mathrm{r})$에 입력되는 r (Spacetime)
  - pixel × 6 channel의 ray H × W 값(camera parameter)이 입력됨
- Q3 $\mathrm{I} = \mathrm{F}^{\mathrm{base}} + \Phi(\mathrm{F}^{\mathrm{dir}}, \mathrm{F}^{\mathrm{time}}, \mathrm{r})$의 Φ (Spacetime)
  - MLP Φ는 featur를 입력으로 받아 Image로 변환하는 MLP
- Q4 $f_i(t) = \left[f_i^{base}, f_i^{dir}, \left(t - \mu_i^T\right) f_i^{time}\right]^T$ (Spacetime)
  - f_i featur들은 embedding된 learnable한 parameter

supplementary

- Q5 quaternion $q_i(t) = \sum_{k=0}^{n_q} c_{i,k}(t - \mu_i^\tau)^k$ (Spacetime)
  - μ_i^τ가 point cloud의 좌표값이 아닌 STG가 가장 잘 보이는 time point로 time값을 가짐
- Q6 Polynomial trajector를 구할 때 따로 optical flow나 scene flow같은 prior 정보를 이용한 추가적인 regularization term이 있지 않음
- Q7 Spacetime Canonical space 관련 질문
  - 일반적으로 사용하는 means3D=pc.get_xyz 값을 불러와 해당 값을 polynomial 식에 사용하여 최종 means3D값 계산하여 매 space마다 gaussian의 흐름을 예측함