Novel Rendering Method Without NN

3D Gaussian Splatting for Real-Time Radiance Field Rendering(SIGGRAPH 2023)

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

Presented By
염수웅

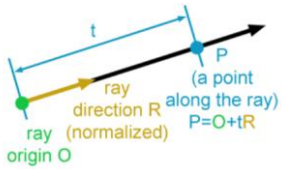
Outline

- Introduction
  - Neural rendering methods
  - Spherical Harmonics
  - 3D Gaussian
  - Dataset
- Main Methods
  - Differentiable 3D Gaussian Splatting
  - Optimization With Adaptive Density Control Of 3D Gaussians
  - Fast Differentiable Rasterizer For Gaussians
- Result
- Conclusion

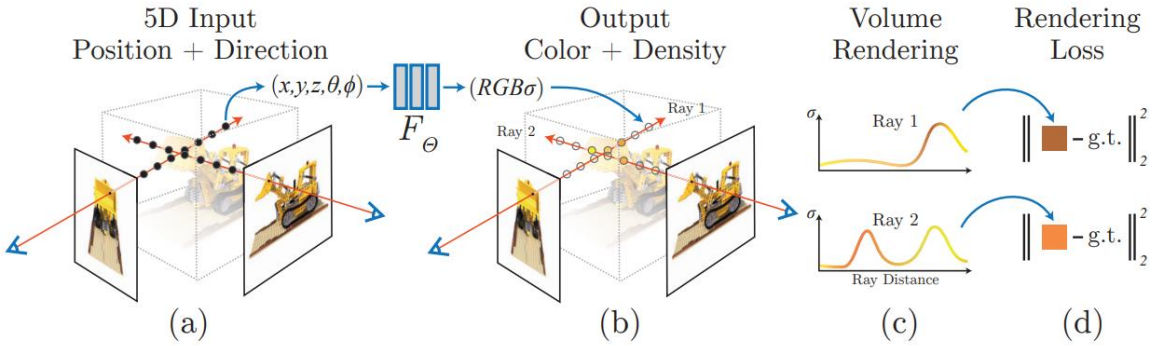
Introduction

- Neural rendering methods(using implicit MLP representation)
  - 대표 모델
    - NeRF[ECCV 20]

Ray $r(t) = o + td$



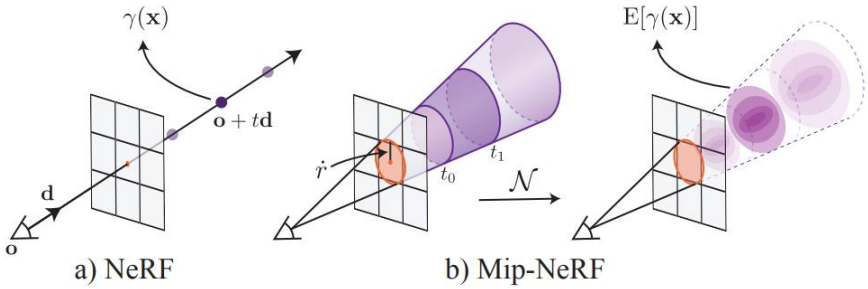
      - 카메라 파라미터를 이용하여 3차원 공간으로 방사되는 ray 계산
      - Stratified sampling 을 통해 ray 위의 coordinate를 추출 후 coarse MLP 에 임베딩
      - Coarse MLP 의 density 가 큰 영역에서 추가 sampling 진행
      - 추가로 sampling 된 coordinate 를 fine MLP에 임베딩하여 density 와 color 출력
      - Volume rendering 을 통해 최종 픽셀 값 렌더링



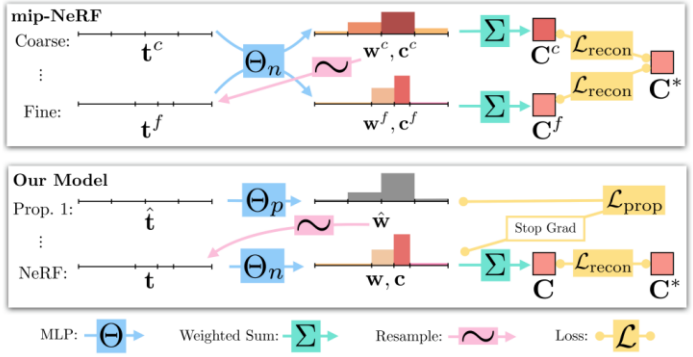
NeRF model architecture

Introduction

- Neural rendering methods(using implicit MLP representation)
  - 대표 모델
    - Mip-NeRF[ICCV 21] & Mip-NeRF360[CVPR 22]
      - NeRF 에서 발생하는 aliasing 문제를 해결한 Mip-NeRF
        - ✓Cone tracing 기법과 Integrated Positional Encoding(IPE) 을 적용하여 문제 해결
      - Unbounded Scene 및 360도 scene 에서 렌더링 품질을 향상한 Mip-NeRF360
        - ✓Unbounded Scene에 normalized 기법을 통해 Mip-NeRF 성능 개선
        - ✓Coarse network 대신 density만 출력하는 proposal network 사용



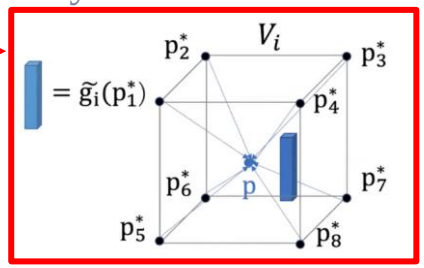
NeRF vs Mip-NeRF



Mip-NeRF vs Mip-NeRF360

Introduction

- Neural rendering methods(using explicit voxel representation)
  - 대표 모델
    - NSVF[NeurIPS 20]
      - 카메라 파라미터를 이용하여 각 픽셀에서 방사되는 ray 계산
      - 3차원 공간을 voxel grid 로 표현후 ray 와 교차되는 voxel 탐색
      - Ray marching 을 통해 voxel 내부의 point sampling 후 color 와 density 계산
        - ✓$g_i(p) = \zeta(\chi(\tilde{g}_i(p_1^*), \dots, \tilde{g}_i(p_8^*))$



Transparency

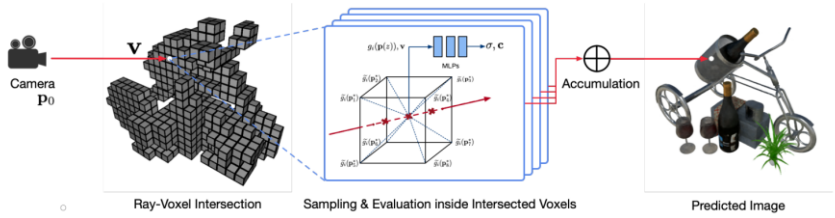
$A(p_0, v) = \prod_{i=1}^{N} \alpha(z_i, \Delta_i)$

$\alpha(z_i, \Delta_i) = \exp(-\sigma(p(z_i)) \cdot \Delta_i)$

$\Delta_i = z_{i+1} - z_i$

- $p_1^* \sim p_8^* \in \mathbb{R}^3$: eight vertices
- $\tilde{g}_i(p_1^*), \dots, \tilde{g}_i(p_8^*) \in \mathbb{R}^d$: feature vector stored at each vertex
- $\chi(\cdot)$: trilinear interpolation
- $\zeta(\cdot)$: positional encoding

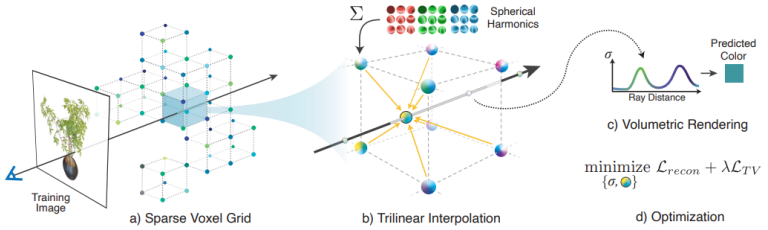
      - Transparency 값 확인 후 thresholding
        - ✓Ray marching 종료



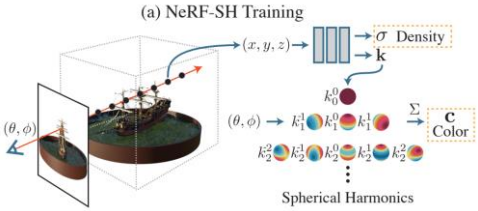
NSVF model architecture

Introduction

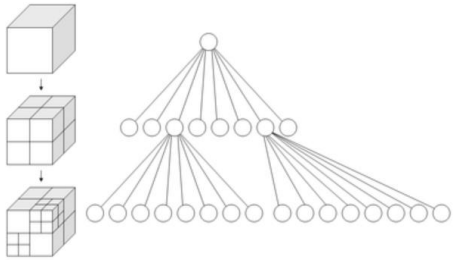
- Neural rendering methods(using explicit voxel representation)
  - 대표 모델
    - PlenOctree[ICCV 21]
      - NSVF 와 동일하게 voxel grid로 공간을 표현하며 spherical harmonics 사용
        - ✓Spherical harmonics를 통해 color 값 연산량 축소
        - ✓Octree 구조를 통해 샘플링 연산량 축소
    - Plenoxels[CVPR 22]
      - NN 사용하지 않고 voxel grid 표현만을 이용하여 렌더링
        - ✓Voxel 의 vertex 에 spherical harmonics coefficient 저장
        - ✓Voxel 의 vertex 에 density 에 대한 feature 저장



Plenoxels model architecture

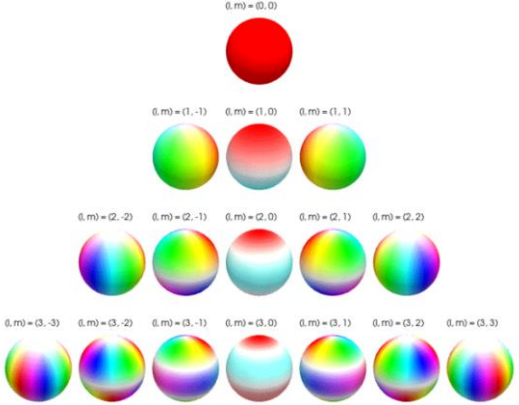


Spherical harmonics 활용

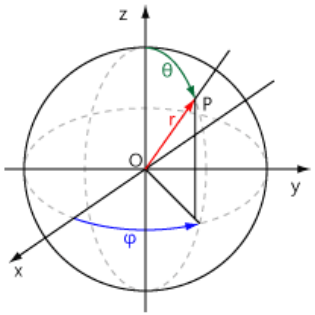


Octree 구조 활용

Introduction



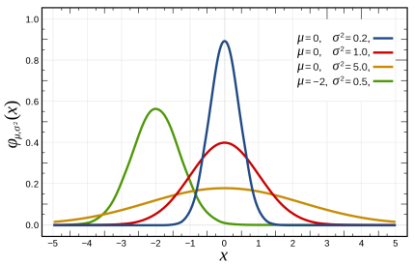
- Spherical Harmonics
  - Spherical harmonics 란?
    - 구면 좌표계에서 각도를 입력 받고 해당하는 구면의 위치에서 값을 출력하는 함수
      - 구면좌표계에서 정의된 함수로, 공간의 복잡한 형태나 패턴을 표현할 수 있는 방법을 제공
        - ✓특정한 수학적 성질을 가지고 있어, 빛의 분포나 반사와 같은 물리적 현상을 모델링하는데 효과적
    - 라플라스 방정식을 이용하여 spherical harmonics의 일반적인 형태 표현 가능
      - $Y_l^m(\theta,\phi) = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}} P_l^{|m|}(cos\theta)e^{im\phi}$



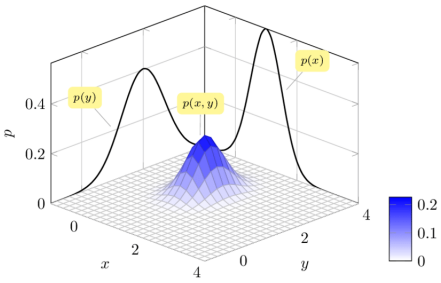
        - ✓라플라스 방정식의 해를 구면 좌표계에서 나타낸 형태
        - ✓l:차수(order), m: 모드(degree)
        - ✓l이 같은 함수들을 묶어서 band라고 부름
      - $C = \sum_{l=0}^{l_{max}} \sum_{m=-l}^{l} k_l^m Y_l^m(\theta,\phi)$
        - ✓뉴럴 렌더링 모델에서 색상을 표현하기 위한 표현

Introduction

- 3D Gaussian
  - 1D Gaussian Distribution
    - $N(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}}\exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\}$

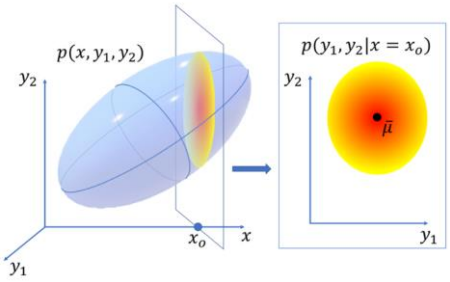


1D Gaussian



2D Gaussian

  - Multivariate Gaussian Distribution
    - $N(X|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma|^{1/2}}\exp\{-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\}$
      - Σ: covariance matrix
  - Ellipsoid 형체를 유지하는 Gaussian
    - Gaussian distribution 에 대한 PDF의 contour 는 ellipsoid 를 보임



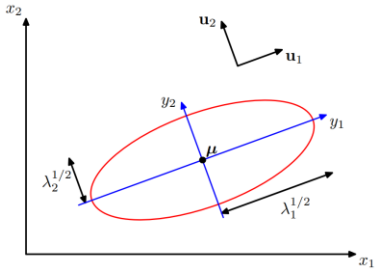
3D Gaussian

$$\Delta^2 = (X-\mu)^T\Sigma^{-1}(X-\mu)$$

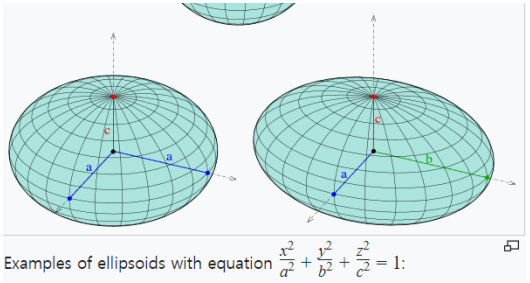
$$\Sigma\cdot u_i = \lambda_i u_i \qquad u_i^T u_j = I_{ij}$$

$$\Sigma = \sum_{i=1}^{D}\lambda_i u_i u_i^T \qquad \Sigma^{-1} = \sum_{i=1}^{D}\frac{1}{\lambda_i}u_i u_i^T$$

$$\Delta^2 = \sum_{i=1}^{D}\frac{y_i^2}{\lambda_i},\ y_i = u_i^T(X-\mu)$$



Gaussian Contour



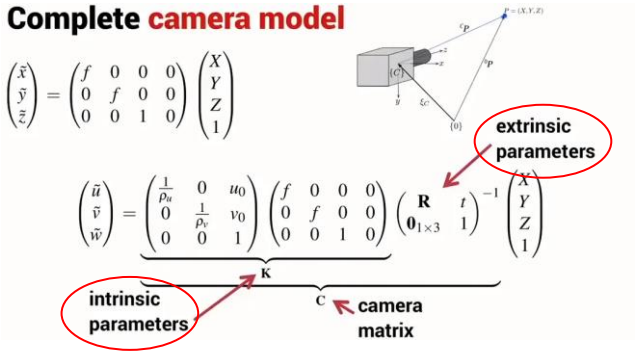
Ellipsoid formulation

Introduction

- Dataset
  - 3D Gaussian Splatting 모델 학습을 위한 데이터 구성
    - Multi view images, 3D point cloud, calibrated camera parameter

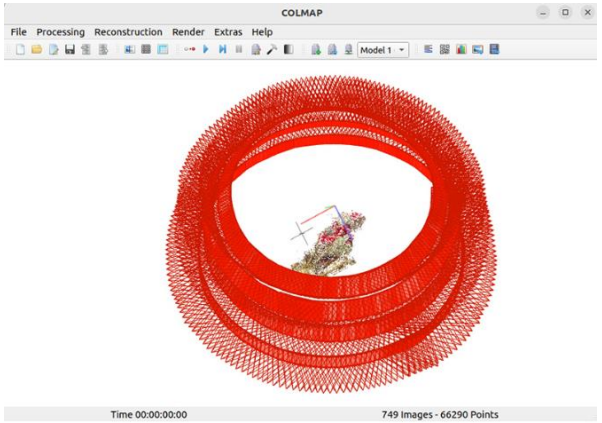
Multi view images

3D point cloud



Calibrated camera parameter

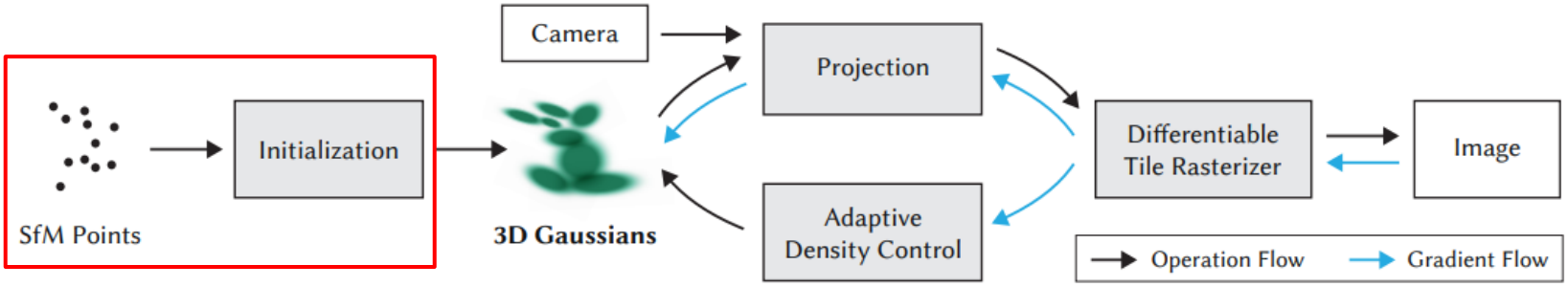
    - 3D point cloud 와 calibrated camera parameter
      - SfM 기반 프로그램을 통해 획득
        - ✓대표적으로 COLMAP 존재



COLMAP Result

Main Method

- Differentiable 3D Gaussian Splatting
  - SfM 알고리즘을 통해 획득한 sparse point cloud 와 카메라 파라미터를 활용
    - 뉴럴 렌더링 모델과 동일하게 카메라 파라미터를 활용
      - 3D Gaussian을 2D 이미지 평면으로 투영하기 위함
    - 뉴럴 렌더링 모델과 다르게 sparse point cloud 를 학습에 활용
      - NeRF, NSVF 와 같은 뉴럴 렌더링 모델에서는 3D point cloud를 사용하지 않음
        - ✓뉴럴 렌더링 모델은 3차원 공간상에 수많은 coordinate 및 voxel을 사용하기 때문
          - 3D point cloud를 학습되는 매개변수로 사용할 경우 novel view synthesis 의미가 사라짐
        - ✓활용하는 경우도 있지만 위치 및 기하학적 정보를 활용하는 보조 수단에 불과



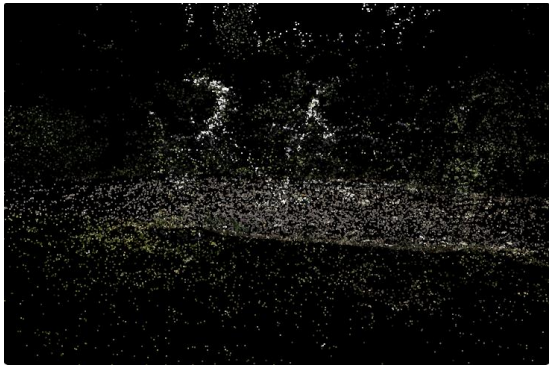
3D Gaussian model architecture

Main Method

- Differentiable 3D Gaussian Splatting
  - 3D Gaussian modeling
    - 3D Gaussian 을 이용하여 3차원 체적을 모델링
      - Point cloud 의 coordinate 를 평균μ 으로 하며 world space 에 정의된 3D covariance
        - ✓$G(x) = e^{-\frac{1}{2}(x)^T \Sigma^{-1}(x)}$
    - 2D 이미지로 투영하는 3D Gaussian
      - 투영된 2D Gaussian의 covariance matrix
        - ✓$\Sigma' = JW\Sigma W^T J^T$
          - J : projective transformation 에 대한 affine 근사 값의 jacobian

World 좌표계에서 카메라 좌표계로의 변환 matrix

카메라 좌표계에서 이미지 좌표계로의 변환 matrix


Main Method

- Differentiable 3D Gaussian Splatting
  - Covariance matrix modeling
    - 3D Gaussian 을 명확하게 얻기 위해 covariance matrix 를 직접 최적화하는것이 바람직함
      - Covariance matrix 는 positive semi-definite 일 때만 유의미한 물리적 의미를 가짐
      - Gradient descent 기반의 방법을 사용할 경우, covariance 가 유의미한 의미를 가지게 하도록 제약을 주기 어려움
        - ✓Eigenvalue 가 음수가 되지 않도록 제약하는 것은 수학적으로 복잡한 과정 필요
          - Covariance matrix 가 positive semi-definite 가 되도록 기울기 업데이트가 쉽지 않음

      ➡ covariance matrix 를 직접 최적화 하는데는 한계점 존재

    - 3D Gaussian이 ellipsoid 형태를 가지도록 covariance matrix를 재구성
      - Scaling 과 Rotation 행렬 성분으로 분해 ⟶ Anisotropic covariance
      - Covariance matrix 는 positive semi definite 일 때 ellipsoid 형체를 가짐
      - Covariance matrix 를 ellipsoid 형태가 되도록 모델링함으로써 positive semi definite 하게 최적화가 가능
        - ✓covariance matrix 가 물리적으로 유의미한 의미를 가지게 되어짐

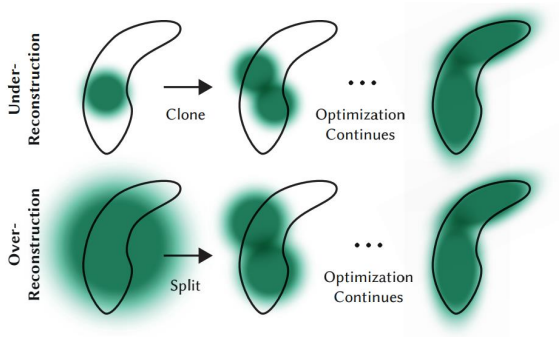
분산을 의미하는 대각성분이 음수가 되지 않기 때문!

Main Methods

- Optimization With Adaptive Density Control Of 3D Gaussians
  - Optimization
    - 최적화하고자하는 3D Gaussian 파라미터
      - Point cloud 위치 p, anisotropic covariance Σ, Opacity α, spherical harmonic coefficients
    - 최적화 기법
      - Stochastic Gradient Descent 기반의 Adam optimizer 사용
      - 일부 작업에대해 CUDA 커널을 추가하여 GPU 작동을 가속
        - ✓CUDA 프로그래밍을 통하여 GPU에서 실행되는 연산 가속화 및 메모리 효율화
    - 손실함수
      - $\mathcal{L} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{D-SSIM}$
        - ✓$SSIM(x,y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)}$
        - ✓$DSSIM(x,y) = \frac{1-SSIM(x,y)}{2}$

Main Methods

- Optimization With Adaptive Density Control Of 3D Gaussians
  - Adaptive control of Gaussians
    - Scene 에 맞게 3D Gaussian 들을 adaptive 하게 변형
      - 매개변수들은 iteration 마다 update 하고 3D Gaussian 은 100 iteration 마다 update
        - ✓3D Gaussian 을 100iteration 마다 제거, 분할, 복제 수행
        - ✓초기의 sparse한 Gaussian 분포 세트에서 장면을 더 잘 표현하는 밀집된 세트로 구성이 가능함과 동시에 더 정확한 매개변수 획득 가능
      - Opacity α가 threshold ϵ_α 보다 낮은 3D Gaussian 은 제거
        - ✓제거 후 geometric feature 가 누락된 영역은 under-reconstruction 수행
        - ✓넓은 영역을 차지하는 Gaussian 분포가 존재하는 영역은 over-reconstruction 수행
        - ✓두 상황 모두 view-space position gradient가 큼
          - 장면의 3차원 공간상에서 위치 변화가 큼을 의미

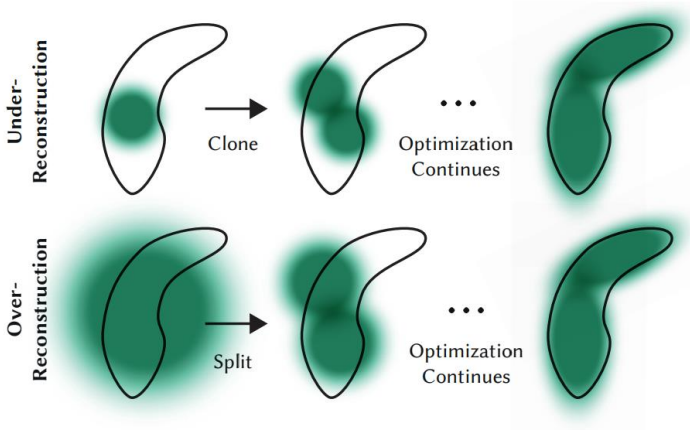


Reconstruction method

Main Methods

- Optimization With Adaptive Density Control Of 3D Gaussians
  - Adaptive control of Gaussians
    - Reconstruction → 두 경우 모두 position gradient 가 크기 때문에, position gradient 에 threshold를 정해서 reconstruction 수행
      - Under reconstruction
        - ✓작은 크기의 3D Gaussian 들을 같은 크기로 복제
          - 복제된 3D Gaussian들을 position gradient 방향으로 배치
      - Over reconstruction
        - ✓큰 크기의 3D Gaussian 들을 작은 크기로 분해
          - 분리된 3D Gaussian의 위치는 초기 Gaussian 의 확률밀도값에 따라 배치

Scale 을 thresholding 하여 reconstruction 종류 결정



Reconstruction method

Main Methods

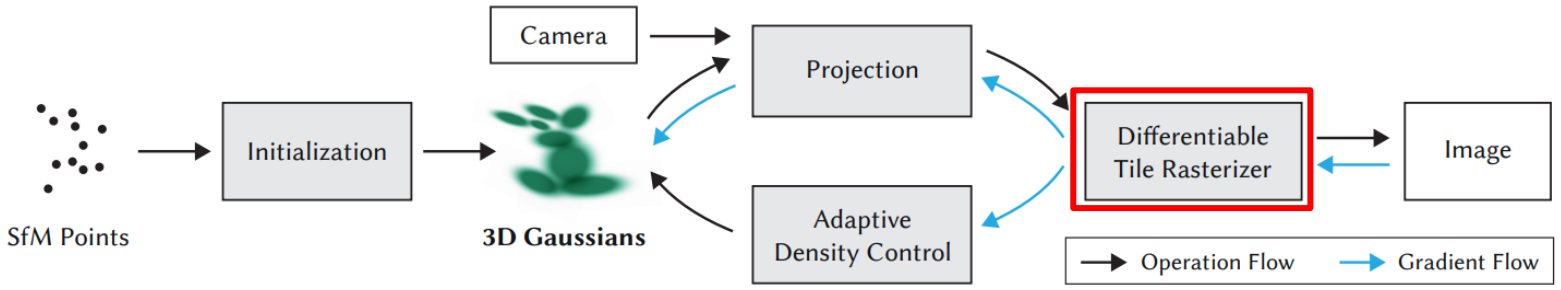
- Fast Differentiable Rasterizer For Gaussians
  - NeRF 의 volume rendering 과 동일한 역할 수행
    - Rasterization 은 rendering 에 포함되는 개념
      - Rendering 과정 중 하나의 단계로 볼 수 있는 rasterization
      - Rendering 과 rasterization 모두 3D 공간을 통해 2D 이미지를 생성하는 기법
  - Rendering 이 아닌 rasterization 을 사용한 이유
    - Point 를 샘플링할 필요 없이 3D Gaussian 이 어떤 픽셀로 떨어지는지만 알면 되기 때문
  - Rasterization
    - 3D Gaussian 을 2D Gaussian 으로 투영
    - 이미지를 16x16 pixel 의 tile들로 분할
      - 이미지의 전체 픽셀을 기준으로 할 경우 너무 많은 Gaussian 을 담게 되어짐
    - 각각의 tile 에 해당하는 Gaussian 을 분류
      - Gaussian 이 여러 tile 에 overlap 되어지기 때문
    - 각 tile 에 있는 Gaussian 들을 GPU Radix sort 를 통해 카메라 direction 방향으로 sorting

Gaussian 이 속해있는 tile 과 depth 정보

GPU를 활용하여 빠르게 sorting 하는 방법

Main Methods

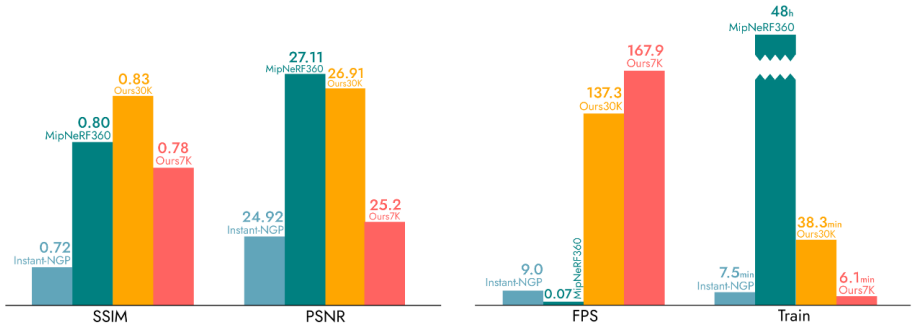
- Fast Differentiable Rasterizer For Gaussians
  - Rasterization
    - 타일 내부의 픽셀에 대해 sorting 된 gaussian의 depth 와 tile 정보를 활용
      - Depth 를 따라 순차적으로 opacity(α) 와 color 값을 accumulate
        - ✓$C = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j)$ ⟶ Alpha-blending
        - ✓Color 값(c_i)은 추정된 매개변수인 coefficient 와 spherical harmonic 을 곱하여 획득



3D Gaussian model architecture

Result

- 이미지 품질 및 추론/학습 시간



  - 학습 및 추론 속도가 압도적으로 빠름을 확인 가능
    - Instant-NGP → 기존 추론/학습 시간 SOTA 모델
  - 성능은 SOTA 가 아니지만 빠른 추론/학습 시간에서 SOTA에 가까운 성능 달성
    - MipNeRF360 → 기존 렌더링 성능 SOTA 모델

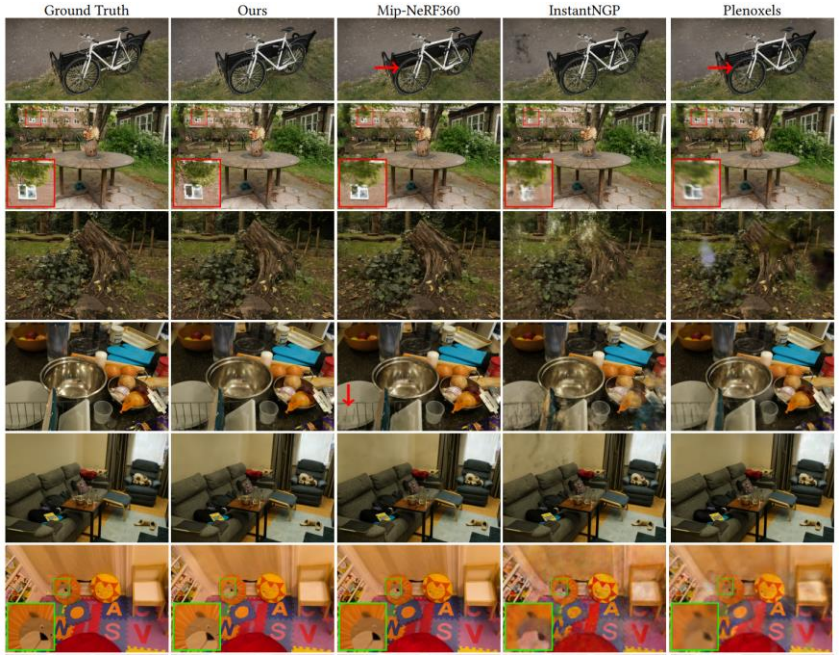
- Point cloud 초기값 없이 학습 할 경우

	Mic	Chair	Ship	Materials	Lego	Drums	Ficus	Hotdog	Avg.
Plenoxels	33.26	33.98	29.62	29.14	34.10	25.35	31.83	36.81	31.76
INGP-Base	36.22	35.00	31.10	29.78	36.39	26.02	33.51	37.40	33.18
Mip-NeRF	36.51	35.14	30.41	30.71	35.70	25.48	33.29	37.48	33.09
Point-NeRF	35.95	35.40	30.97	29.61	35.04	26.06	36.13	37.30	33.30
Ours-30K	35.36	35.83	30.80	30.00	35.78	26.15	34.87	37.72	33.32

→ Point cloud 없이도 준수한 성능을 보임

Result

- 정성 & 정량 결과
  - 성능 측면과 학습/추론 속도 관점에서 대부분의 데이터에서 SOTA 달성



Dataset	Mip-NeRF360						Tanks&Temples						Deep Blending					
Method|Metric	*SSIM*↑	*PSNR*↑	*LPIPS*↓	Train	FPS	Mem	*SSIM*↑	*PSNR*↑	*LPIPS*↓	Train	FPS	Mem	*SSIM*↑	*PSNR*↑	*LPIPS*↓	Train	FPS	Mem
Plenoxels	0.626	23.08	0.463	25m49s	6.79	2.1GB	0.719	21.08	0.379	25m5s	13.0	2.3GB	0.795	23.06	0.510	27m49s	11.2	2.7GB
INGP-Base	0.671	25.30	0.371	5m37s	11.7	13MB	0.723	21.72	0.330	5m26s	17.1	13MB	0.797	23.62	0.423	6m31s	3.26	13MB
INGP-Big	0.699	25.59	0.331	7m30s	9.43	48MB	0.745	21.92	0.305	6m59s	14.4	48MB	0.817	24.96	0.390	8m	2.79	48MB
M-NeRF360	0.792†	27.69†	0.237†	48h	0.06	8.6MB	0.759	22.22	0.257	48h	0.14	8.6MB	0.901	29.40	0.245	48h	0.09	8.6MB
Ours-7K	0.770	25.60	0.279	6m25s	160	523MB	0.767	21.20	0.280	6m55s	197	270MB	0.875	27.78	0.317	4m35s	172	386MB
Ours-30K	0.815	27.21	0.214	41m33s	134	734MB	0.841	23.14	0.183	26m54s	154	411MB	0.903	29.41	0.243	36m2s	137	676MB

Conclusion

- Conclusion
  - 3D Gaussian 을 통한 3차원 radiance fields 모델링
    - Point cloud 활용한 3D Gaussian 을 통해 기존 샘플링 기반 뉴럴 렌더링과 다르게 모델링
    - 뉴럴 네트워크를 사용하지 않고 모델 구성
  - Adaptive control 을 통해 3차원 공간을 효과적으로 표현
    - Under-Reconstruction 을 통해 3차원 상의 빈 공간이 없도록 학습 진행
    - Over-Reconstruction 을 통해 불필요하게 퍼져있는 공간을 보정
  - 뉴럴 렌더링에서 사용하는 volume rendering 대신 raterization 사용
    - 빠른 연산을 가능하게 하여 신시간 렌더링 가능
- Limitation
  - Point cloud 가 준비되어 있지 않으면 기존 뉴럴 렌더링 방법보다 성능이 낮음
    - SfM을 통해 sparse point cloud를 생성하는 시간까지 고려하면 매우 긴 시간이 소요됨
  - 잘 계산되어있는 카메라 파라미터가 필수적
  - 동적 객체가 포함된 장면은 point cloud 생성이 어렵기 때문에 적용하기 어려움

감사합니다