3D Gaussian Splatting

Presented By
정성훈

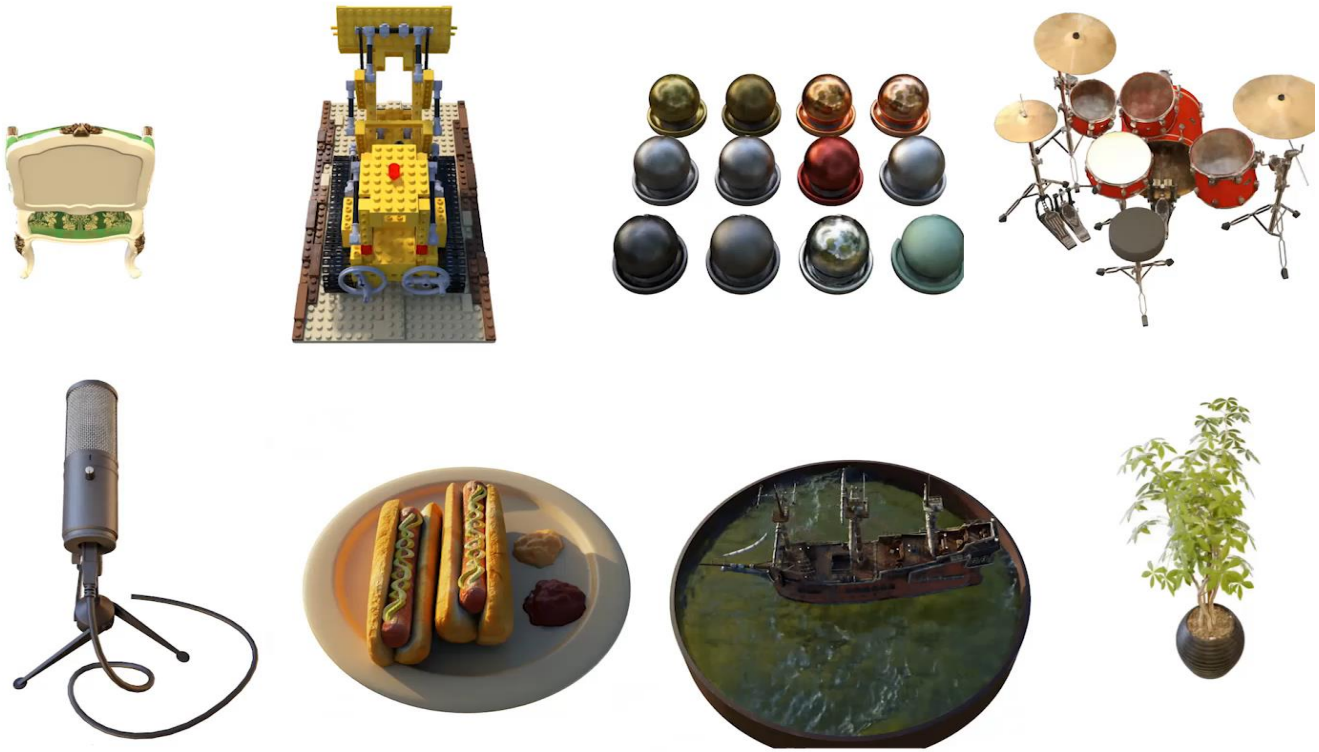
Outline

- 개요
- 3D Gaussian splatting

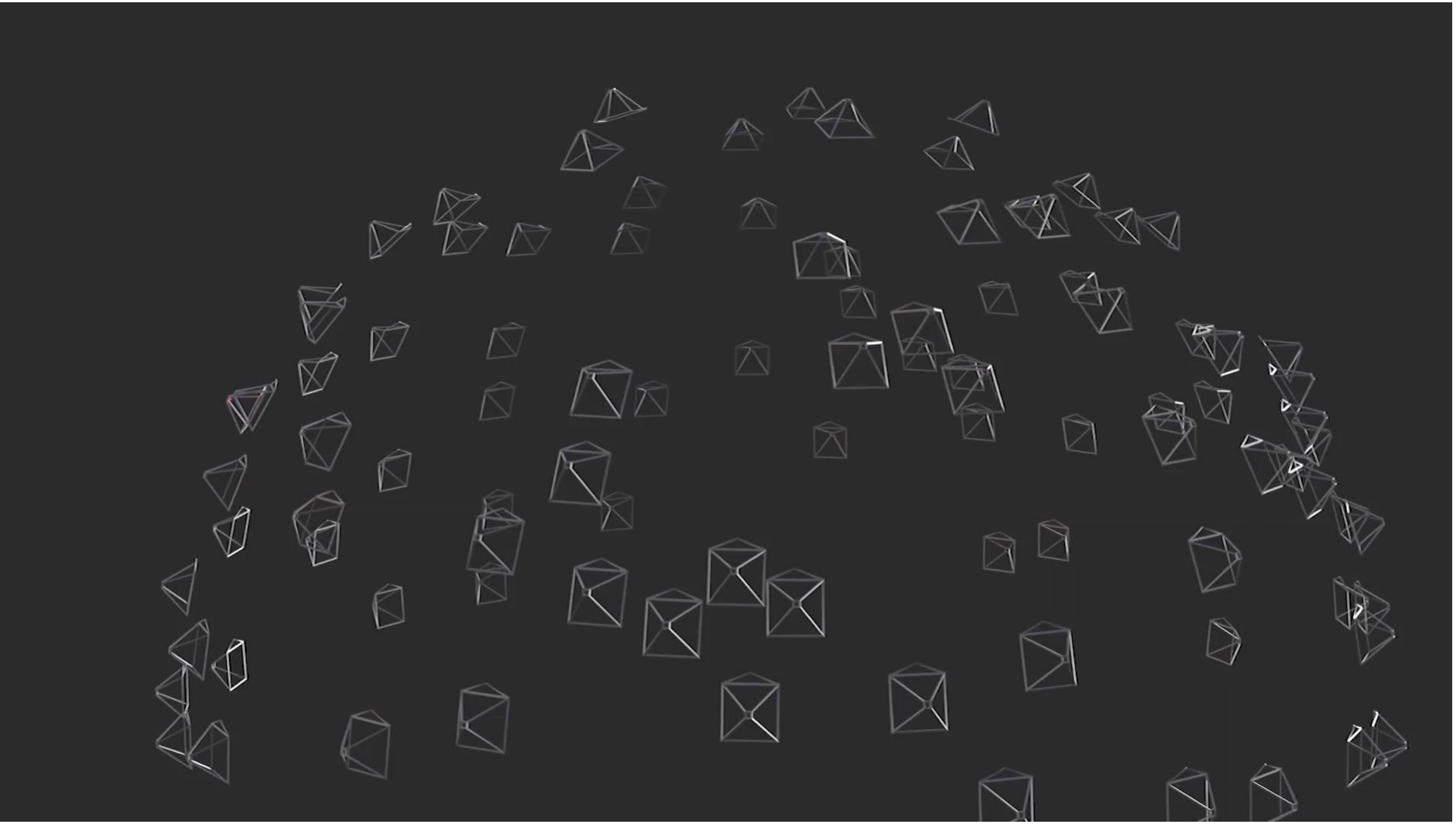
개요

개요

- Novel View Synthesis
  - 의미
    - 주어진 이미지에 대해 새로운 카메라 각도에서 바라본 이미지를 추정하는 task
    - 대표적으로 NeRF[1)]가 있음



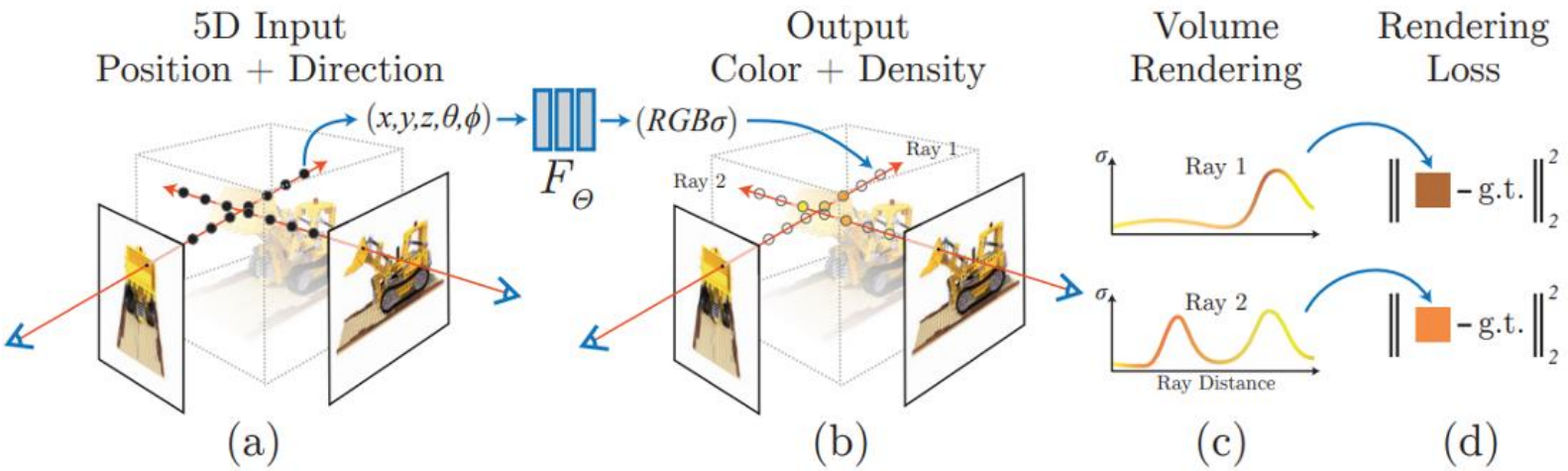
개요



개요

1) Mildenhall, Ben, et al. "Nerf: Representing scenes as neural radiance fields for view synthesis." *Communications of the ACM* 65.1 (2021): 99-106.

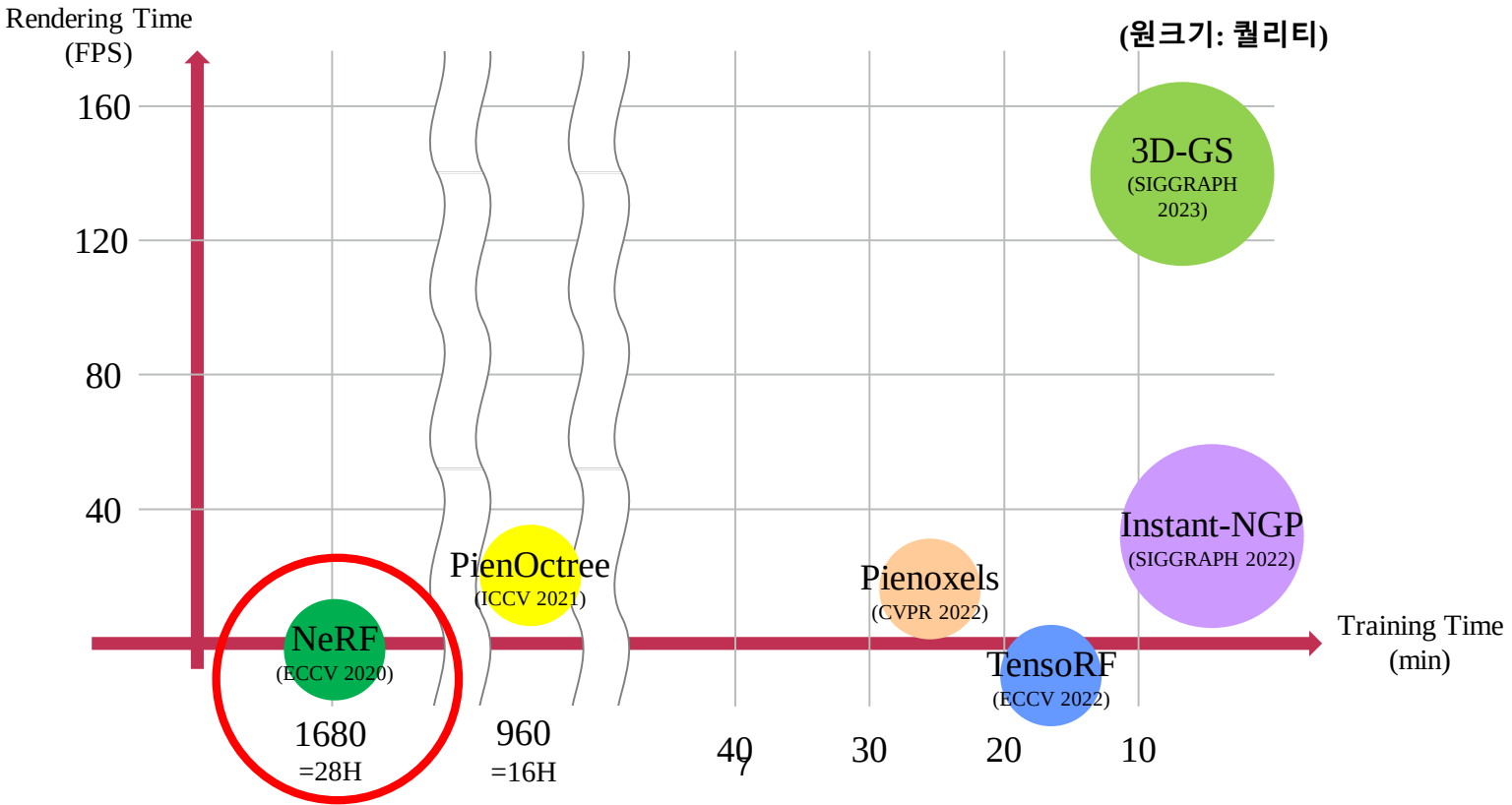
- NeRF[1)]
  - Ray-based sampling
    - 이미지 각 pixel마다 ray를 그려서 여러 점을 샘플링
    - 샘플링한 각 점의 color와 volume density를 계산
    - ray위의 이 값들을 summation 하여 이미지를 Rendering



개요

1) Mildenhall, Ben, et al. "Nerf: Representing scenes as neural radiance fields for view synthesis." *Communications of the ACM* 65.1 (2021): 99-106.

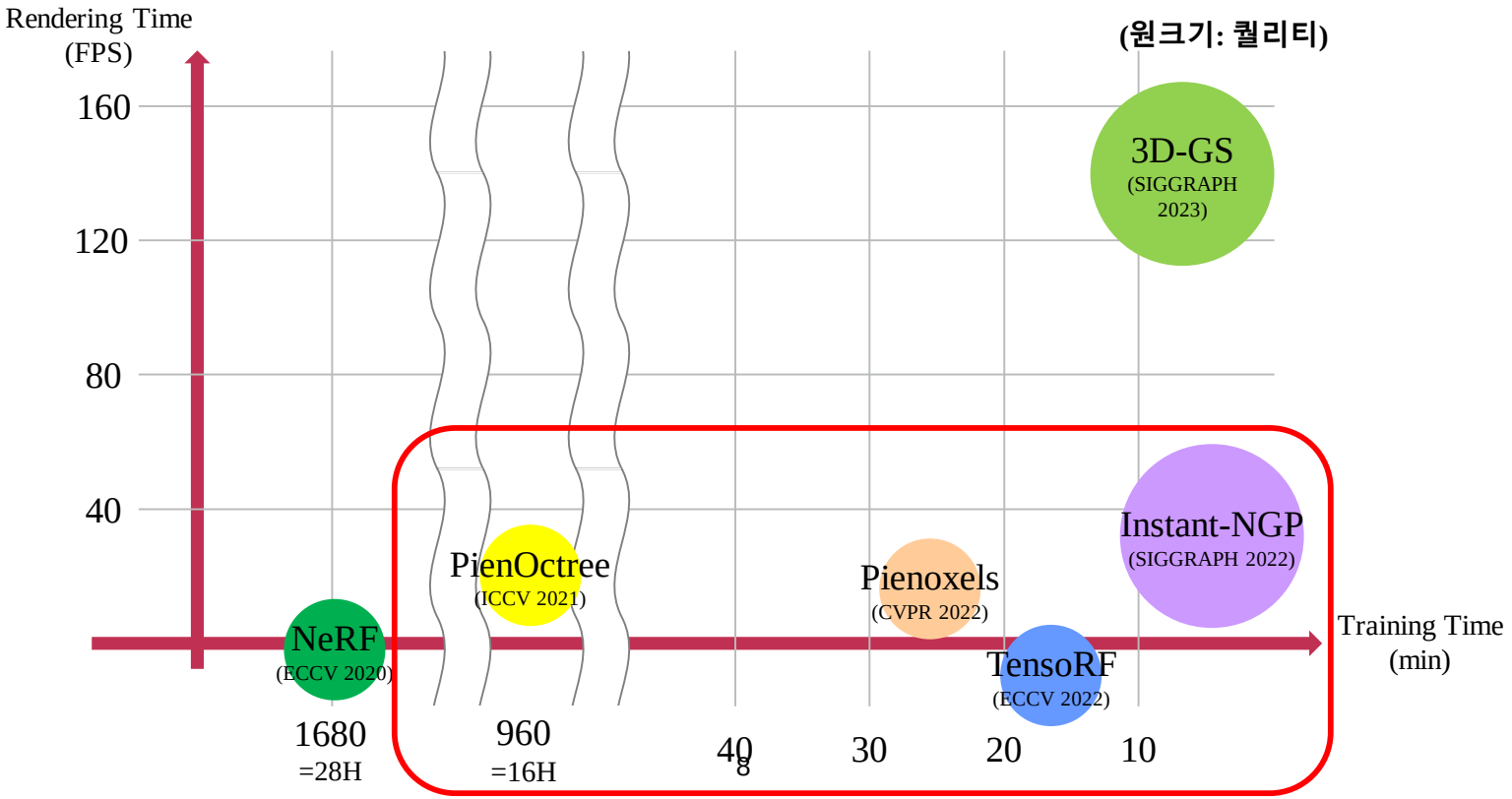
- NeRF[1)]
  - 단점
    - 하나의 장면을 학습하는데 1일 이상의 시간 소요
    - Rendering: 새로운 이미지 1장에 대해 30초 소요
    - 실시간 불가능



개요

1) Mildenhall, Ben, et al. "Nerf: Representing scenes as neural radiance fields for view synthesis." *Communications of the ACM* 65.1 (2021): 99-106.

- NeRF[1)]
  - 후속 연구
    - 마찬가지로 Ray-based sampling
    - 연산 시간을 줄이는 방향으로 연구가 진행됨
    - 하지만 렌더링 속도가 느려 실시간으로 적용하기 어려움
    - 퀄리티와 속도의 trade-off

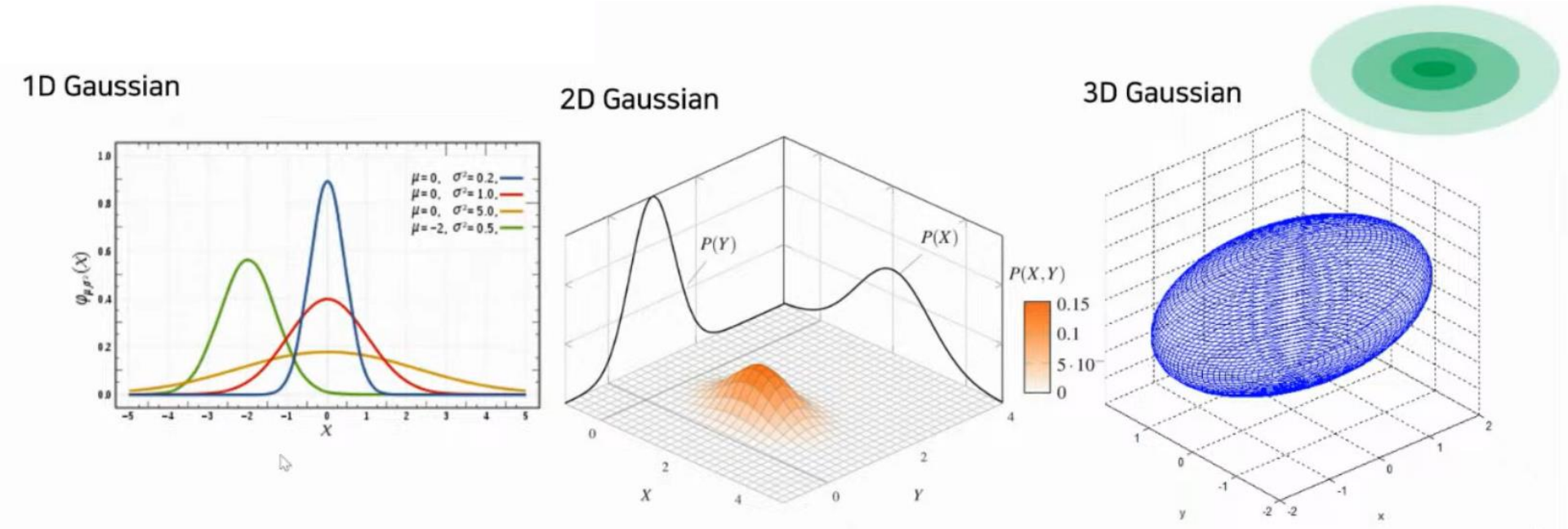


3D Gaussian Splatting

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- 3D Gaussian
  - 의미
    - 평균과 분산으로 이루어진 정규 분포를 3차원으로 표현한 것

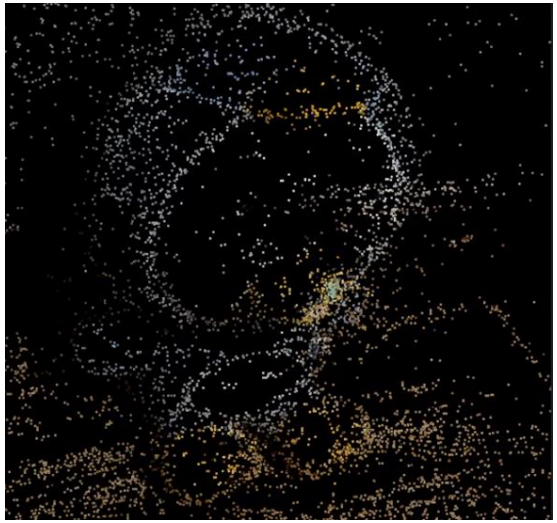


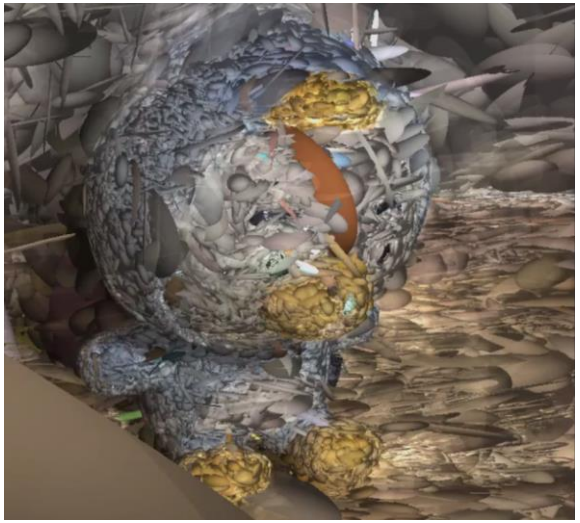
3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Splatting
  - 의미
    - 사전적 의미: 물기를 머금은 사물 따위가 무겁게 내려 앉거나 부딪치는 모습을 의성어로 표현한 단어
    - 3차원 공간으로 흩뿌려진 3D Gaussian을 최적화 함으로써 reconstruction 수행







3D Gaussian Splatting

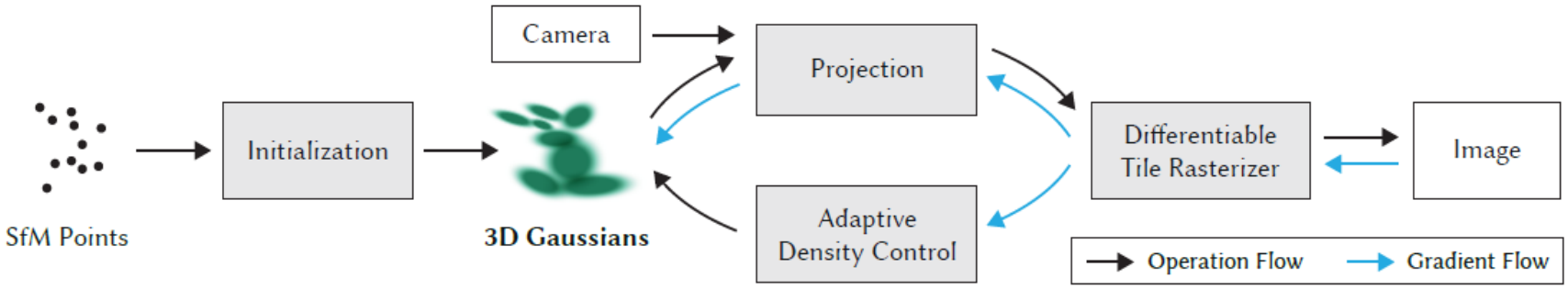
Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Why Gaussian?
  - 기존 3D representation
    - Mesh와 point cloud: GPU/CUDA 기반의 resterization(3D를 2D 이미지화)에 최적화되어 있음
      - 일반적으로 3D modeling 또는 reconstruction에 사용되는 표현 방법
    - 최근 NeRF 연구들은 다양한 방법을 통해 연산시간을 줄였지만 아직 Ray-based sampling을 통해 렌더링하기 때문에 연산량이 많고 noise가 형성될 수 있음
  - 3D Gaussian
    - Differentiable volumetric representation (미분가능한 볼륨 표현)
    - Explicit하게 표현 가능
      - Neural Network는 구조가 감춰져 있는 implicit한 표현
    - 3D 모델의 2D projection과 α-blending을 효율적으로 수행 가능
      - Projection, α-blending: resterization을 구성하는 일부 단계
    - 위 특성으로 인해 빠른 rendering 가능

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

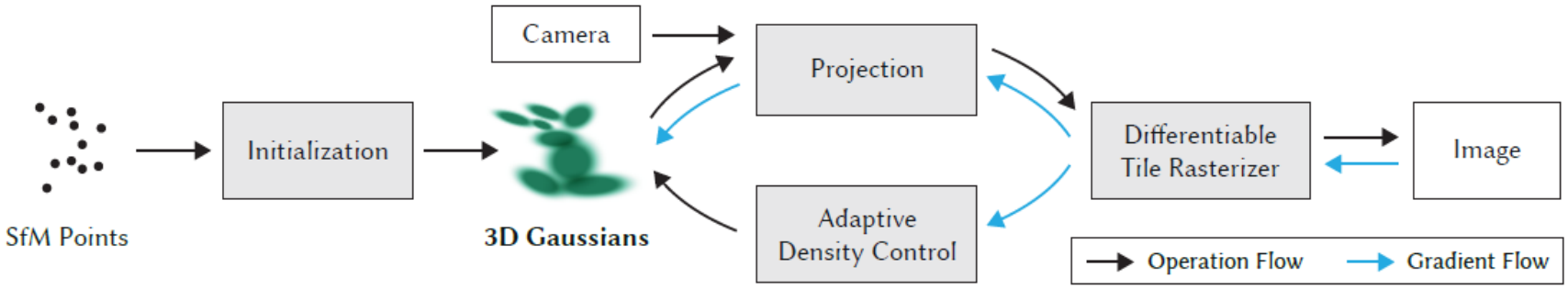
- Overview
  - Initialization
    - SfM 알고리즘을 통해 생성된 sparse한 point cloud 획득
      - 이를 3D Gaussian의 초기값으로 사용
  - Projection
    - 3D Gaussian이 plane 이미지로 projection되어 2D Gaussian형태로 변환
  - Differentiable Tile Rasterizer
    - 미분 가능한 형태의 Tile Rasterization을 통해 2D Gaussian을 하나의 이미지로 생성



3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Overview
  - Gradient Flow
    - 생성된 이미지와 GT 이미지의 Loss를 계산하고 parameter update
  - Adaptive Density Control
    - Gradient를 기반으로 Gaussian의 형태를 변화시킴



3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- 알고리즘 구성
  - Initialization
  - Parameter Update
  - Adaptive Density Control

Algorithm 1 Optimization and Densification

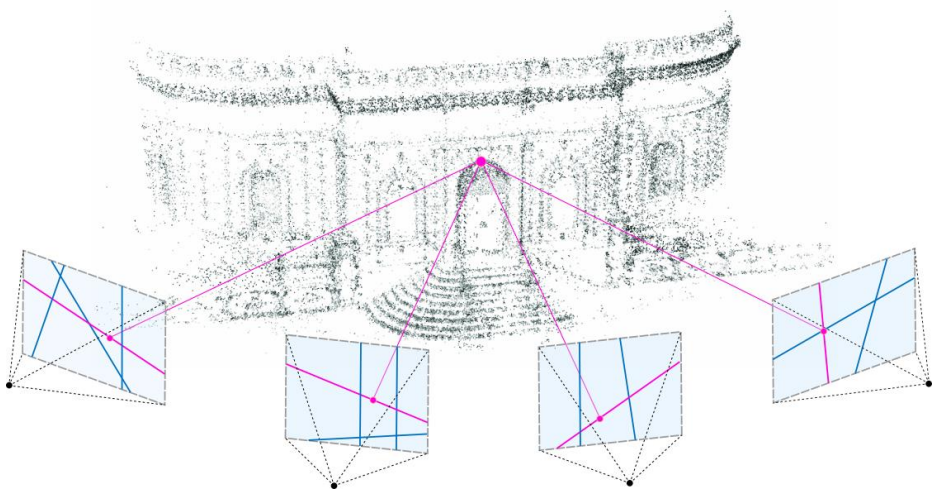
w, h: width and height of the training images

$M \leftarrow$ SfM Points ▷ Positions
$S, C, A \leftarrow$ InitAttributes() ▷ Covariances, Colors, Opacities
$i \leftarrow 0$ ▷ Iteration Count
while not converged **do**
  $V, \hat{I} \leftarrow$ SampleTrainingView() ▷ Camera V and Image
  $I \leftarrow$ Rasterize(M, S, C, A, V) ▷ Alg. 2
  $L \leftarrow Loss(I, \hat{I})$ ▷ Loss
  $M, S, C, A \leftarrow$ Adam(∇L) ▷ Backprop & Step
  if IsRefinementIteration(i) **then**
    for all Gaussians (μ, Σ, c, α) **in** (M, S, C, A) **do**
      if $\alpha < \epsilon$ or IsTooLarge(μ, Σ) **then** ▷ Pruning
        RemoveGaussian()
      end if
      if $\nabla_p L > \tau_p$ **then** ▷ Densification
        if $\|S\| > \tau_S$ **then** ▷ Over-reconstruction
          SplitGaussian(μ, Σ, c, α)
        else ▷ Under-reconstruction
          CloneGaussian(μ, Σ, c, α)
        end if
      end if
    end for
  end if
  $i \leftarrow i + 1$
end while

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Initialization
  - 변수 초기화
    - SfM 알고리즘을 통해 생성된 sparse한 point cloud 획득
  - SfM
    - Structure from Motion
    - 2D 이미지로 3D 정보를 추출
    - 동일한 객체를 다른 시점에서 중첩되도록 찍은 multi-view 이미지로부터 3D structure와 camera pose를 복원하는 프로세스
    - 이를 통해 point cloud 생성



3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Initialization
  - M
    - SfM으로 획득한 point cloud
    - 초기 3D Gaussian들의 평균 값으로 사용
      - 3D gaussian의 중심이 되는 지점
  - S
    - 3D Gaussian의 Covariance Matrix (공분산 행렬)

$$\Sigma = RSS^T R^T$$

    - 수식에서의 S: Scale Matrix
      - x,y,z 축에 대한 배율 정보
    - 수식에서의 R: Rotation Matrix
      - Gaussian의 회전 정도

$M \leftarrow$ SfM Points ▷ Positions

$S, C, A \leftarrow$ InitAttributes() ▷ Covariances, Colors, Opacities

$i \leftarrow 0$ ▷ Iteration Count

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Initialization
  - S
    - 3D Gaussian의 Covariance Matrix (공분산 행렬)

$$\Sigma = RSS^T R^T$$

    - RS : scaled rotation matrix
    - $S^T R^T$: Covariance Matrix는 Symmetric한 성질을 갖기 때문에, 이러한 특성을 만들기 위해

$M \leftarrow$ SfM Points ▷ Positions
$S, C, A \leftarrow$ InitAttributes() ▷ Covariances, Colors, Opacities
$i \leftarrow 0$ ▷ Iteration Count

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Initialization
  - C
    - 3D Gaussian의 color값
    - Spherical Harmonics(SH)로 정의됨
    - 이전의 NeRF에서는 pixel의 색만 정의하면 되는 문제였지만, 3D Gaussian은 pixel보다 넓은 범위를 채우기 때문에 표면의 색을 정의해주는 함수를 이용
  - A
    - 3D Gaussian의 투명도 값

$M \leftarrow$ SfM Points ▷ Positions

$S, C, A \leftarrow$ InitAttributes() ▷ Covariances, Colors, Opacities

$i \leftarrow 0$ ▷ Iteration Count

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Optimization
  - 1번 – load data
    - 정답이미지 $\hat{I}$와 해당 이미지의 camera pose 정보 V 읽어옴
  - 2번 – Rasterizer
    - M(mean = x, y, x), S(covariance), C(color), A(투명도), V(camera pose)를 입력으로 받아 Rasterize 수행 후 prediction된 이미지를 생성
  - 3번 – Loss
    - Predicted image와 GT image 비교하여 loss 계산
  - 4번 – Update
    - Adam opitmizer로 M, S, C, A값 update

while not converged **do**

1. $V, \hat{I} \leftarrow$ SampleTrainingView() ▷ Camera V and Image
2. $I \leftarrow$ Rasterize(M, S, C, A, V) ▷ Alg. 2
3. $L \leftarrow Loss(I, \hat{I})$ ▷ Loss
4. $M, S, C, A \leftarrow$ Adam(∇L) ▷ Backprop & Step

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- 2번 – Rasterizer
  - 알고리즘
    - M(mean = x, y, x), S(covariance), C(color), A(투명도), V(camera pose)를 입력으로 받아 Rasterize 수행 후 prediction된 이미지를 생성
    - 3D Gaussian 들로 구성된 3D model을 특정 camera pose에서 그려질 2D 이미지에 대한 렌더링 방법

Algorithm 2 GPU software rasterization of 3D Gaussians

w, h: width and height of the image to rasterize
M, S: Gaussian means and covariances in world space
C, A: Gaussian colors and opacities
V: view configuration of current camera

```
function RASTERIZE(w, h, M, S, C, A, V)
    CullGaussian(p, V)                          ▷ Frustum Culling
    M', S' ← ScreenspaceGaussians(M, S, V)      ▷ Transform
    T ← CreateTiles(w, h)
    L, K ← DuplicateWithKeys(M', T)             ▷ Indices and Keys
    SortByKeys(K, L)                            ▷ Globally Sort
    R ← IdentifyTileRanges(T, K)
    I ← 0                                       ▷ Init Canvas
    for all Tiles t in I do
        for all Pixels i in t do
            r ← GetTileRange(R, t)
            I[i] ← BlendInOrder(i, L, r, K, M', S', C, A)
        end for
    end for
     return I
end function
```

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- 2번 – Rasterizer
  - Input
    - w, h : image width, height
    - M, S : World 좌표계에서 3D Gaussian 평균, 공분산
    - C, A : 3D Gaussian의 color와 opacity(투명도)
    - V : 이미지에 대한 camera pose

Algorithm 2 GPU software rasterization of 3D Gaussians

w, h: width and height of the image to rasterize
M, S: Gaussian means and covariances in world space
C, A: Gaussian colors and opacities
V: view configuration of current camera

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- 2번 – Rasterizer
  - Cull Gaussian
    - 주어진 camera pose V 에서 관측 될 수 있는 3D Gaussian들만 선택
  - Screen space Gaussian
    - 3D Gaussian을 image plane으로 projection하여 2D Gaussian 생성

$$\Sigma' = JW\Sigma W^T J^T \quad \rightarrow \quad 2D\ projection\ equation$$

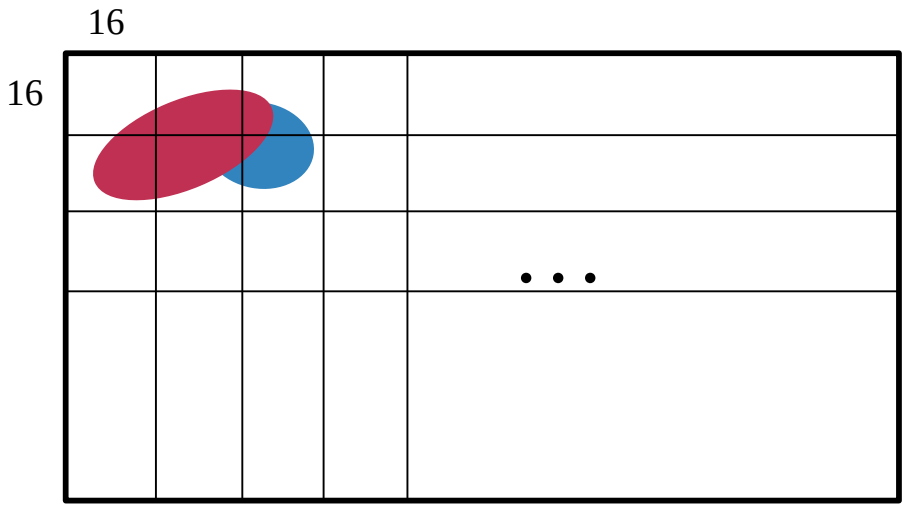
    - J: Projective Transformation의 Jacobian(각 축으로 편미분한 Matrix)
      - Projective Transformation : Camera좌표계에서 Image 좌표계로 변환하는 matrix
    - W: Viewing Transformation
      - World 좌표계에서 Camera 좌표계로 변환하는 matrix
    - Σ : World좌표계에서의 Covariance Matrix
    - Σ' : Image좌표계에서의 Covariance Matrix

```
function RASTERIZE(w, h, M, S, C, A, V)
    CullGaussian(p, V)                          ▷ Frustum Culling
    M', S' ← ScreenspaceGaussians(M, S, V)      ▷ Transform
    T ← CreateTiles(w, h)
```

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- 2번 – Rasterizer
  - Create Tiles
    - 입력으로 받은 w, h 로 구성된 image를 16x16 pixel의 Tile로 나눔
      - CUDA에서 병렬로 처리하기 위함



function RASTERIZE(w, h, M, S, C, A, V)
  CullGaussian(p, V) ▷ Frustum Culling
  $M', S' \leftarrow$ ScreenspaceGaussians(M, S, V) ▷ Transform
  $T \leftarrow$ CreateTiles(w, h)

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- 2번 – Rasterizer
  - Duplicate with Keys
    - 각 2D Gaussian이 Overlap되는 tile 수 만큼 instance로 변환
    - CUDA에서 병렬적으로 Threading하기 때문에 1개의 Guassian이 여러 Tile로 복제됨
    - 각 instance들은 view space depth와 tile ID로 조합하여 (dictionary) Key K 생성
    - K: 총 64 bit 길이
      - 앞의 32 bit: depth 정보 encoding
      - 뒤의 32 bit: overlap 되는 tile의 index가 encoding

```
L, K ← DuplicateWithKeys(M′, T)          ▷ Indices and Keys
SortByKeys(K, L)                          ▷ Globally Sort
R ← IdentifyTileRanges(T, K)
I ← 0                                     ▷ Init Canvas
for all Tiles t in I do
    for all Pixels i in t do
        r ← GetTileRange(R, t)
        I[i] ← BlendInOrder(i, L, r, K, M′, S′, C, A)
    end for
end for
 return I
end function
```

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- 2번 – Rasterizer
  - Sort by Keys
    - Key로 sorting
    - Tile 마다 모든 splat들에 대해 depth ordering(정렬) 수행
      - 이를 통해 카메라와 가까운 Gaussian 순서로 정렬됨
    - 기존 방법들은 pixel마다 정렬이 필요하여 연산량이 높았으나 이러한 방법으로 효율적으로 정렬 수행

```
    L, K ← DuplicateWithKeys(M′, T)          ▷ Indices and Keys
    SortByKeys(K, L)                          ▷ Globally Sort
    R ← IdentifyTileRanges(T, K)
    I ← 0                                     ▷ Init Canvas
    for all Tiles t in I do
        for all Pixels i in t do
            r ← GetTileRange(R, t)
            I[i] ← BlendInOrder(i, L, r, K, M′, S′, C, A)
        end for
    end for
    return I
end function
```

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

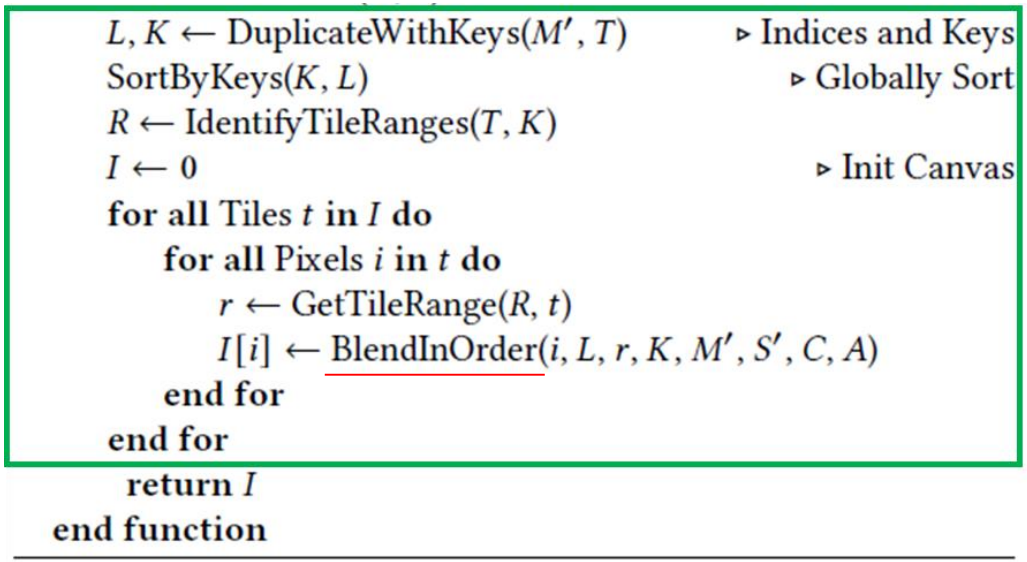
- 2번 – Rasterizer
  - Identify Tile Ranges
    - 같은 Tile ID를 가진 시작과 끝 Gaussian을 식별(identify)하여 Gaussian list를 효율적으로 관리
  - Get Tile Range
    - 이미지의 모든 tile 각각에 대해 Range r을 읽어옴

```
    L, K ← DuplicateWithKeys(M′, T)          ▷ Indices and Keys
    SortByKeys(K, L)                          ▷ Globally Sort
    R ← IdentifyTileRanges(T, K)
    I ← 0                                     ▷ Init Canvas
    for all Tiles t in I do
        for all Pixels i in t do
            r ← GetTileRange(R, t)
            I[i] ← BlendInOrder(i, L, r, K, M′, S′, C, A)
        end for
    end for
    return I
end function
```

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- 2번 – Rasterizer
  - Blend In Order
    - Tile 마다 주어진 하나의 pixel에 대해 앞에서 뒤로 순차적으로 color와 alpha(투명도) 값을 accumulate



$L, K \leftarrow$ DuplicateWithKeys(M', T) ▷ Indices and Keys
SortByKeys(K, L) ▷ Globally Sort
$R \leftarrow$ IdentifyTileRanges(T, K)
$I \leftarrow 0$ ▷ Init Canvas
for all Tiles t **in** I **do**
  for all Pixels i **in** t **do**
    $r \leftarrow$ GetTileRange(R, t)
    $I[i] \leftarrow$ BlendInOrder$(i, L, r, K, M', S', C, A)$
  end for
end for
return I
end function

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- 3번 – Loss
  - Function
    - Predicted image와 GT image 비교하여 loss 계산

$$\mathcal{L} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{D-SSIM} \rightarrow \lambda = 0.2$$

  - SSIM
    - Structural Similarity Index, Image Quality Assignment 방법 중 하나
    - 원본 이미지와 압축된 이미지를 비교할 때, 픽셀간 비교하는 것이 아닌, Luminance, Contrastm Structure 세 가지 측면에서 품질 비교
  - D-SSIM
    - Structural dissimilarity
    - $\mathcal{L}_{D-SSIM} = 1 - SSIM$

```
# Loss
gt_image = viewpoint_cam.original_image.cuda()
Ll1 = l1_loss(image, gt_image)
loss = (1.0 - opt.lambda_dssim) * Ll1 + opt.lambda_dssim * (1.0 - ssim(image, gt_image))
loss.backward()
```

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

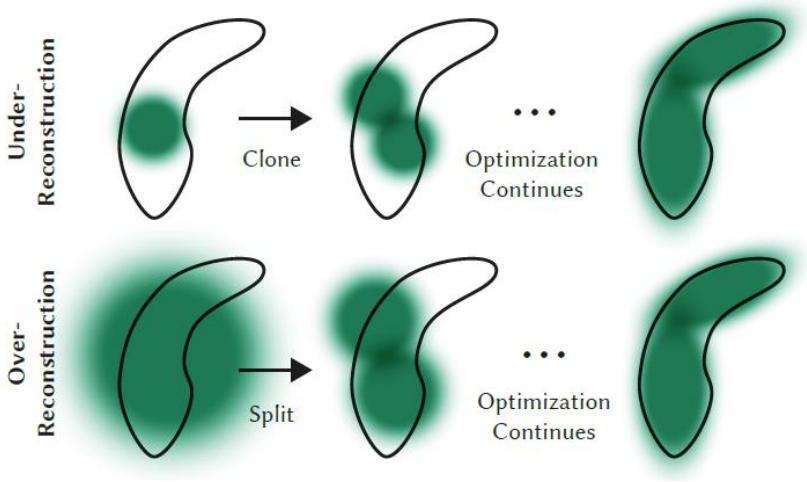
- Adapive Control of Gaussians
  - 3D Gaussian Optimization
    - Scene에 맞게 3D Gaussian들을 adaptive하게 변형시키는 단계
    - M(mean = x, y, x), S(covariance), C(color), A(투명도) parameter들은 매 iteration마다 update
    - 3D Gaussian은 100 iteration 마다 update
  - Remove Gaussian
    - 특정 Threshold (=ϵ, 0.005)보다 낮은 alpha 값 (α, 투명도)을 가진 Gaussian은 제거 됨

```
if IsRefinementIteration(i) then
    for all Gaussians (μ, Σ, c, α) in (M, S, C, A) do
        if α < ε or IsTooLarge(μ, Σ) then                ▷ Pruning
            RemoveGaussian()
        end if
        if ∇_p L > τ_p then                              ▷ Densification
            if ||S|| > τ_S then                          ▷ Over-reconstruction
                SplitGaussian(μ, Σ, c, α)
            else                                         ▷ Under-reconstruction
                CloneGaussian(μ, Σ, c, α)
            end if
        end if
    end for
end if
```

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

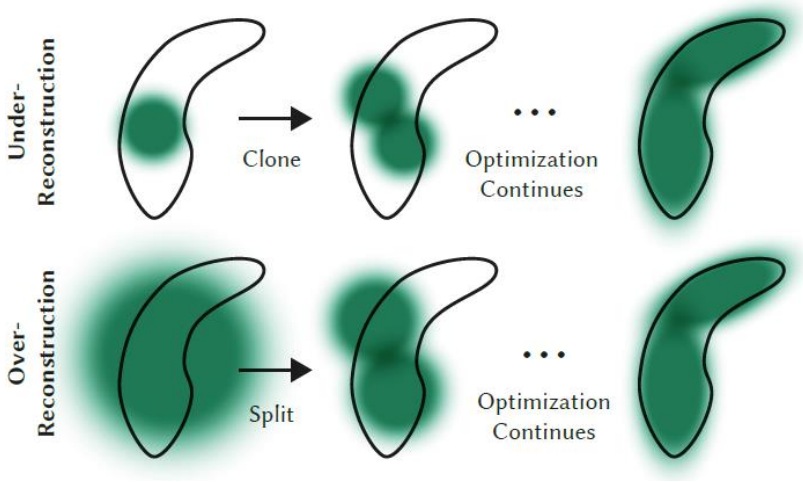
- Adapive Control of Gaussians
  - Under/Over-reconstruction region
    - 추정된 3D Gaussian의 크기가 크거나 작아 영역을 완전하게 채우지 못한 경우
      - 공통적으로 view-space positional gradient가 큰 값을 가짐
      - 직관적으로, 이는 아직 완전히 복원되지 않은 region에 해당하며 최적화가 적용되어야 하는 부분이기 때문
    - View-space position gradient의 평균 크기(average magnitude)가 특정 Threshold(0.0002) 이상일 때, Gaussian들을 split하거나 clone



3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Adapive Control of Gaussians
  - Clone Gaussian
    - Under-Reconstruction region
    - 작은 크기의 (covariance가 작은) 3D Gaussian들은 같은 크기로 copy
    - 이후 positional gradient의 방향에 배치
  - Split Gaussian
    - Over-Reconstruction region
    - 큰 크기의 (covariance가 큰) 3D Gaussian들은 작은 Gaussian으로 split
    - 분할되는 크기 x 1.6 = 현재의 큰 3D Gaussian의 크기
    - 분할되는 크기 = 현재의 큰 3D Gaussian의 크기 / 1.6(실험적으로 결정한 값)



3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Adapive Control of Gaussians
  - 3D Gaussian 수 조절
    - 다른 volumetric 기법들과 동일하게, 카메라의 가까운 영역에 floater들이 생성될 수 있음
    - 이러한 floater들은 3D Gaussian의 수를 무작위로 증가시킴
    - 이를 제어하기 위해 3000 iteration 마다 α값을 0에 가깝게 설정
    - 또한 매우 큰 Gaussian들은 주기적으로 제거

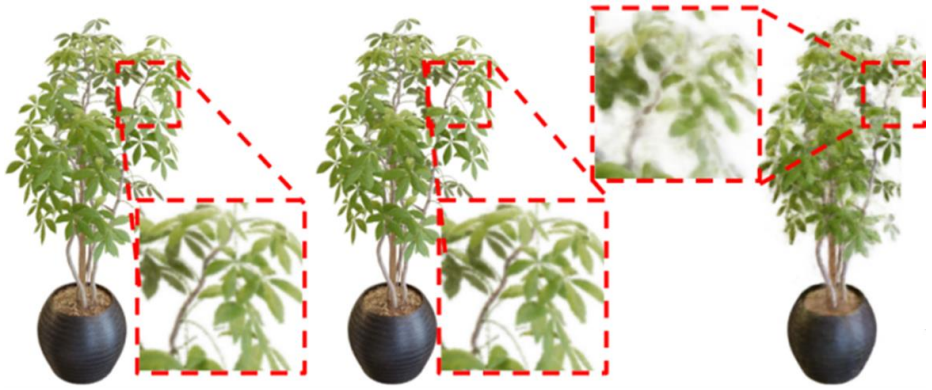


3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Quantitative Results
  - PSNR (Peak Signal-to-noise ratio)
    - 원본 이미지에 비해 화질이 얼마나 손상되었는지 평가
    - MSE는 원본 이미지와 비교 이미지 간의 각 pixel 오차 계산
    - PSNR에서 MSE는 분모로 사용되며, MSE 값이 작을 수록 PSNR값이 커짐
    - MAX: 1개 pixel 당 갖고 있는 정보량 (gray: 255, RGB: 3x255, RGBA: 4x255)
    - MSE의 scale을 변경하여 표현하는 것이 PSNR

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \qquad MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M * N}$$

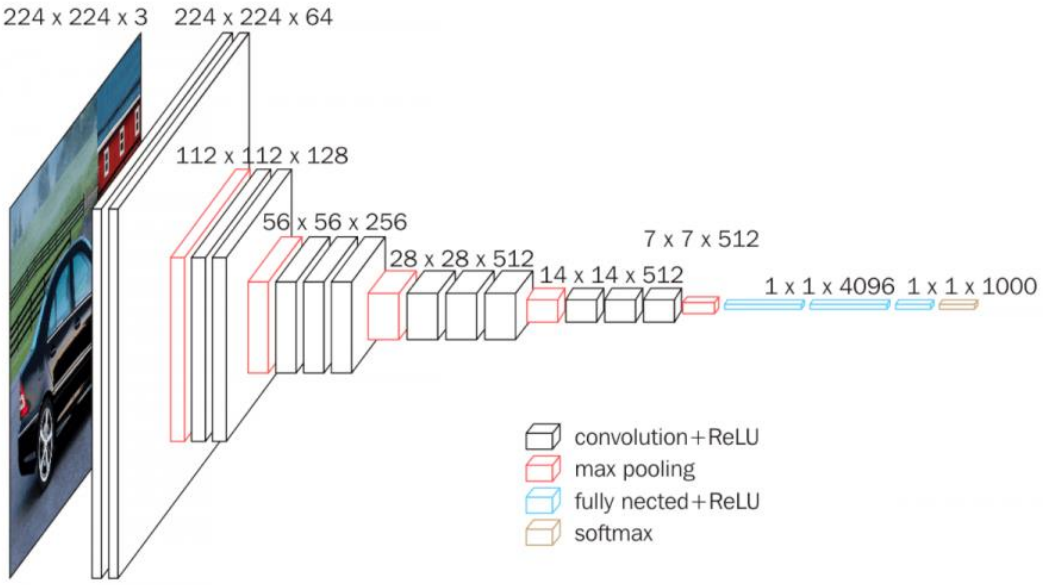


PSNR 31.05 PSNR 36.13 PSNR 30.13

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Quantitative Results
  - LPIPS (Learned Perceptual Image Patch Similarity)
    - 두 이미지의 유사도 평가
    - Classification Task를 Supervised, Self-supervised Unsupervised 딥러닝 모델(VGG)로 학습
    - 비교할 두 이미지를 학습된 Network를 통해 feature를 추출
    - Feature를 비교하여 유사도 평가
    - VGG: Visual Geometry Group. 다중 레이어가 있는 CNN 아키텍쳐



3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Quantitative Results
  - SOTA
    - 거의 대부분의 지표에서 최고 품질 달성
    - 압도적인 FPS
  - 단점
    - 많은 GPU memory 사용
      - 밑의 표에는 나와있지 않지만 최고 20GB 사용

Dataset	Mip-NeRF360						Tanks&Temples					
Method|Metric	*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
INGP-Base	0.671	25.30	0.371	5m37s	11.7	13MB	0.723	21.72	0.330	5m26s	17.1	13MB
INGP-Big	0.699	25.59	0.331	7m30s	9.43	48MB	0.745	21.92	0.305	6m59s	14.4	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
Ours-7K	0.770	25.60	0.279	6m25s	160	523MB	0.767	21.20	0.280	6m55s	197	270MB
Ours-30K	0.815	27.21	0.214	41m33s	134	734MB	0.841	23.14	0.183	26m54s	154	411MB

3D Gaussian Splatting

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." ACM Transactions on Graphics 42.4 (2023).

- Qualitative Results







감사합니다.