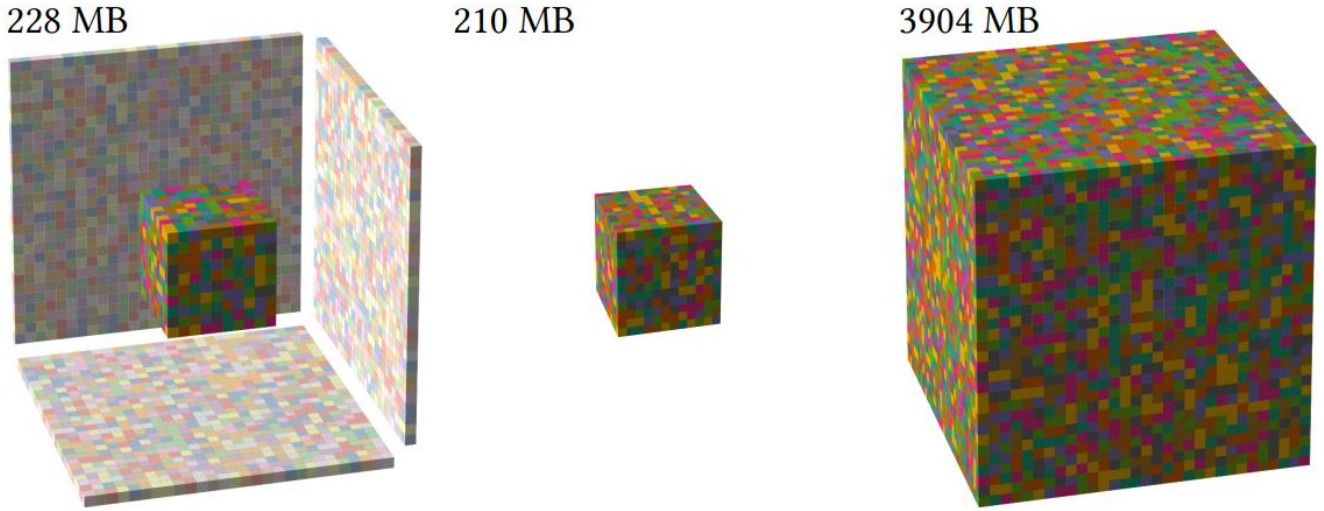
MERF: Memory-Efficient Radiance Fields for Real-time View Synthesis in Unbounded Scenes
(SIGRAPH 2023)



한국과학기술연구원(KIST)
CVIPL 학생연구원 **김연욱**

한국과학기술연구원
Korea Institute of Science and Technology

MERF: Memory-Efficient Radiance Fields for Real-time View Synthesis in Unbounded Scenes

CHRISTIAN REISER*, University of Tübingen, Tübingen AI Center, Germany and Google Research, United Kingdom
RICHARD SZELISKI, Google Research, United States of America
DOR VERBIN, Google Research, United States of America
PRATUL P. SRINIVASAN, Google Research, United States of America
BEN MILDENHALL, Google Research, United States of America
ANDREAS GEIGER, University of Tübingen, Tübingen AI Center, Germany
JONATHAN T. BARRON, Google Research, United States of America
PETER HEDMAN, Google Research, United Kingdom

Dor Verbin

팔로우

Research Scientist at Google Research
google.com의 이메일 확인됨 - 홈페이지
Computer Vision

제목	인용	연도
Mip-nerf 360: Unbounded anti-aliased neural radiance fields JT Barron, B Mildenhall, D Verbin, PP Srinivasan, P Hedman Proceedings of the IEEE/CVF conference on computer vision and pattern …	2636	2022

Ben Mildenhall

팔로우

Cofounder, World Labs
worldlabs.ai의 이메일 확인됨 - 홈페이지
Computer vision computer graphics computational imaging

제목	인용	연도
NeRF: Representing scenes as neural radiance fields for view synthesis B Mildenhall, PP Srinivasan, M Tancik, JT Barron, R Ramamoorthi, R Ng arXiv preprint arXiv:2003.08934	15625	2020

Pratul Srinivasan

팔로우

Research Scientist at Google DeepMind
google.com의 이메일 확인됨 - 홈페이지
Computer Vision Computer Graphics Machine Learning

제목	인용	연도
Nerf: Representing scenes as neural radiance fields for view synthesis B Mildenhall, PP Srinivasan, M Tancik, JT Barron, R Ramamoorthi, R Ng Communications of the ACM 65 (1), 99-106	15625	2021
Fourier features let networks learn high frequency functions in low dimensional domains M Tancik, P Srinivasan, B Mildenhall, S Fridovich-Keil, N Raghavan, … Advances in neural information processing systems 33, 7537-7547	3505	2020

Richard Szeliski

팔로우

Google DeepMind + CSE, The University of Washington
cs.washington.edu의 이메일 확인됨 - 홈페이지
Computer Vision Computer Graphics

제목	인용	연도
Computer vision: algorithms and applications R Szeliski Springer Nature	11462	2022
A taxonomy and evaluation of dense two-frame stereo correspondence algorithms D Scharstein, R Szeliski International journal of computer vision 47 (1), 7-42	10780	2002
Photo tourism: exploring photo collections in 3D N Snavely, SM Seitz, R Szeliski ACM siggraph 2006 papers, 835-846	5095	2006

Contents

OverView

OverView

Phone, labtop 과 같은 고사양 GPU가 없는 기기에서도 실시간 렌더링이 가능하도록 해보자

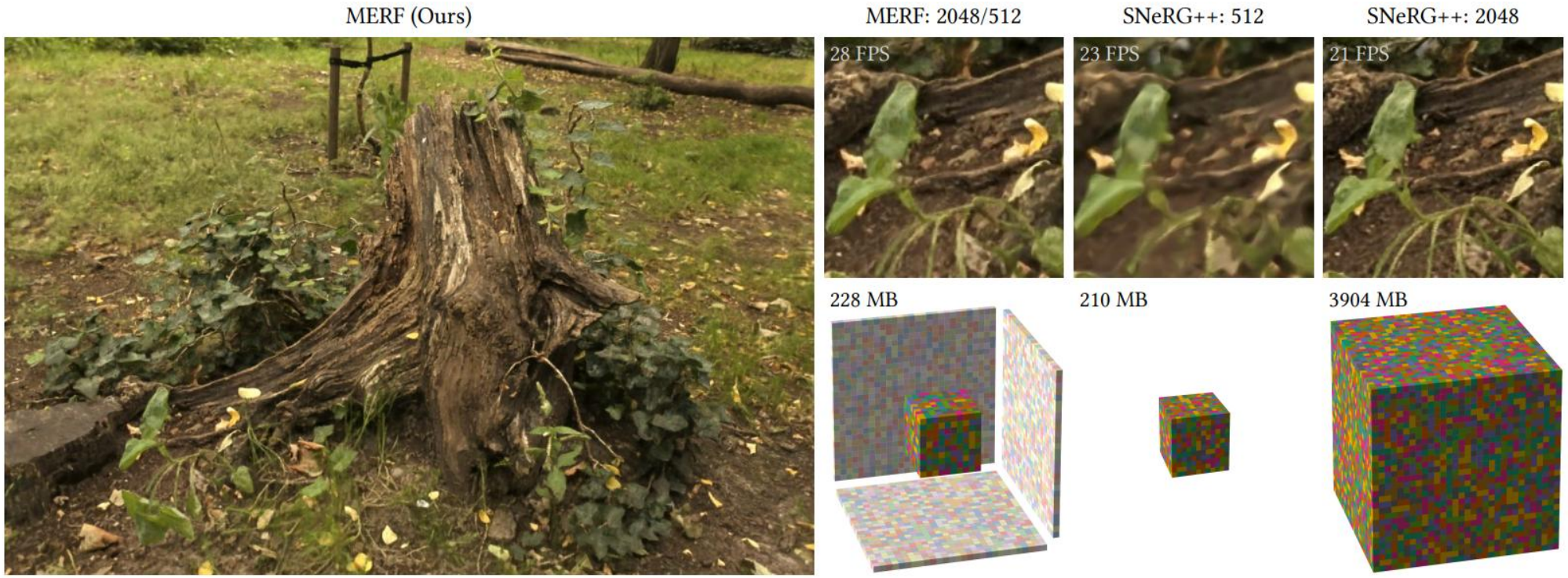
Smartphone

Laptop with
weak GPU

Laptop with
strong GPU

OverView

Memory efficience



Introduction

Introduction

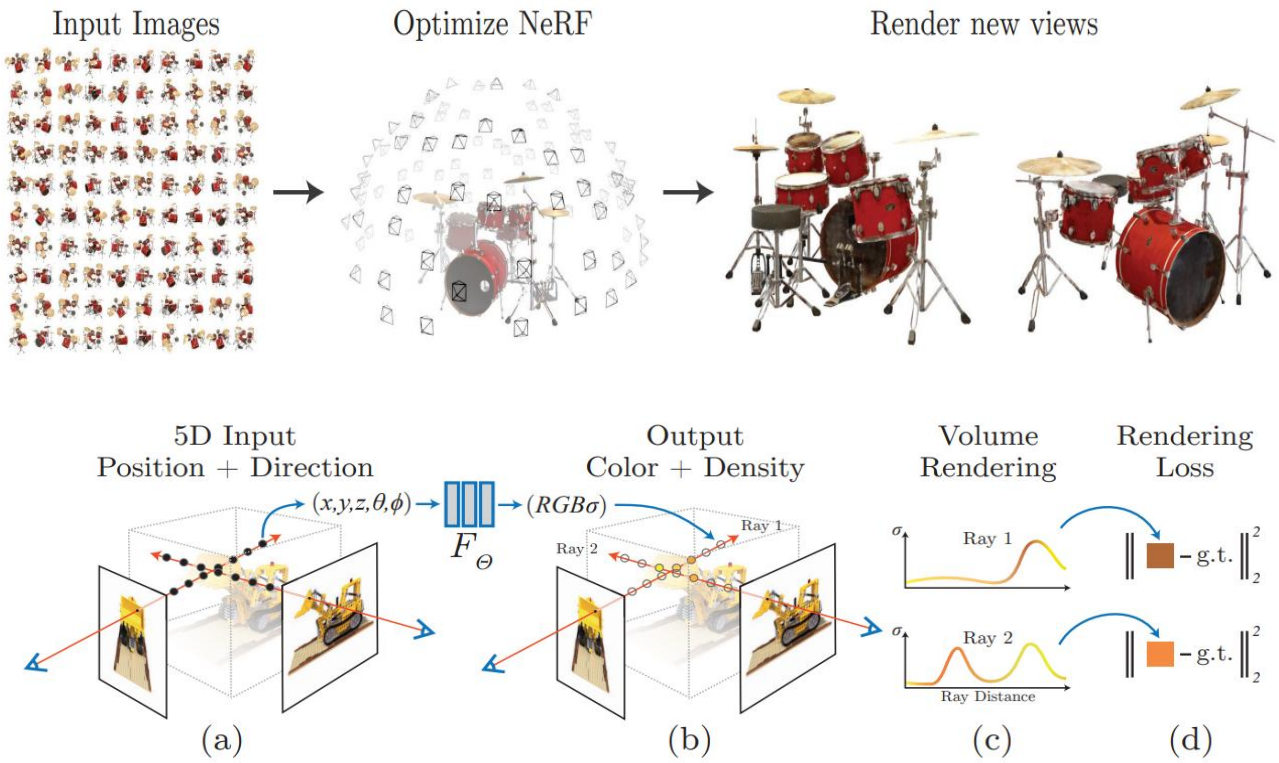
NeRF(Volumetric methods)'s trade off:

1. Volume *vs.* Surface

- Volume: 뛰어난 품질, 느린 속도와 큰 연산 비용
- Surface: 빠른 렌더링 속도, 이미지 품질 저하

2. Memory bound *vs.* *Compute bound*

- Memory bound: 빠른 렌더링 속도, 방대한 그래픽 메모리
- Compute bound: 적은 메모리 사용, query 시 복잡한 MLP 연산(FLOPS)



Introduction

- 품질, 메모리, 속도 세가지를 모두 만족하시키는 것은 힘들다.

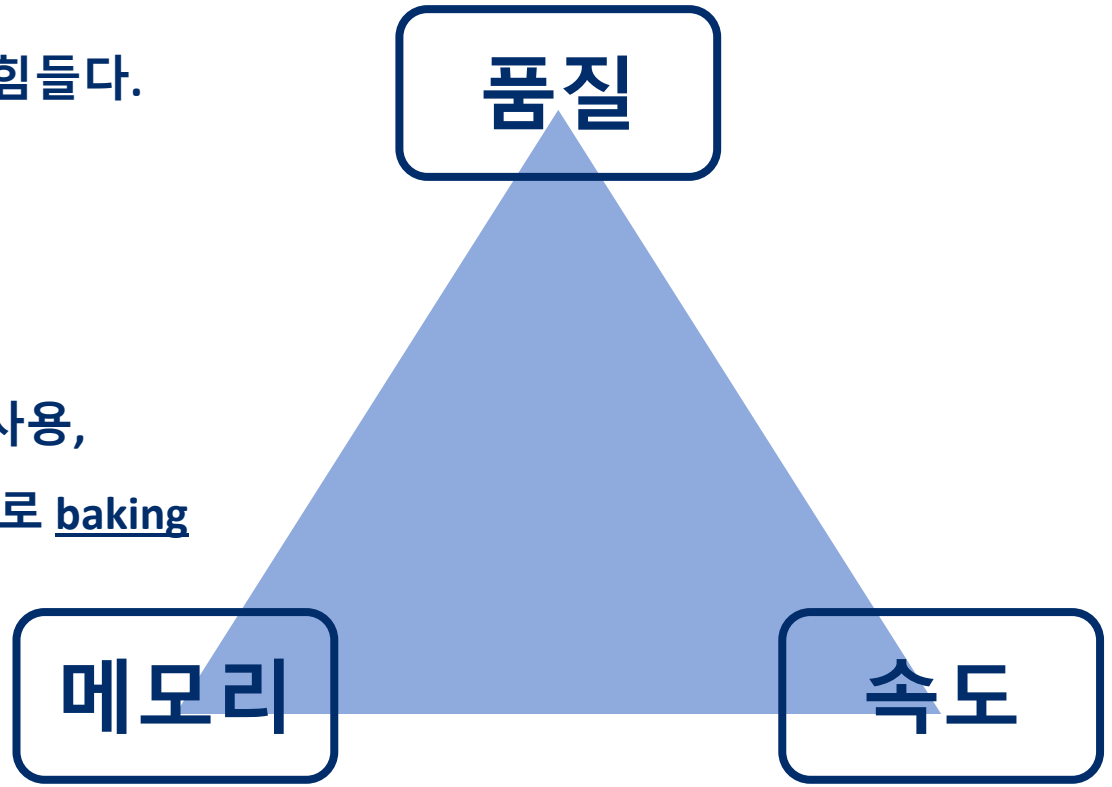
→ 일반적으로 학습단계와 렌더링 단계를 분리.

학습은 속도는 느리지만 메모리를 적게 차지하도록 MLP 사용,

렌더링은 용량은 더 크지만 속도가 훨씬 빠른 표현 방식으로 baking

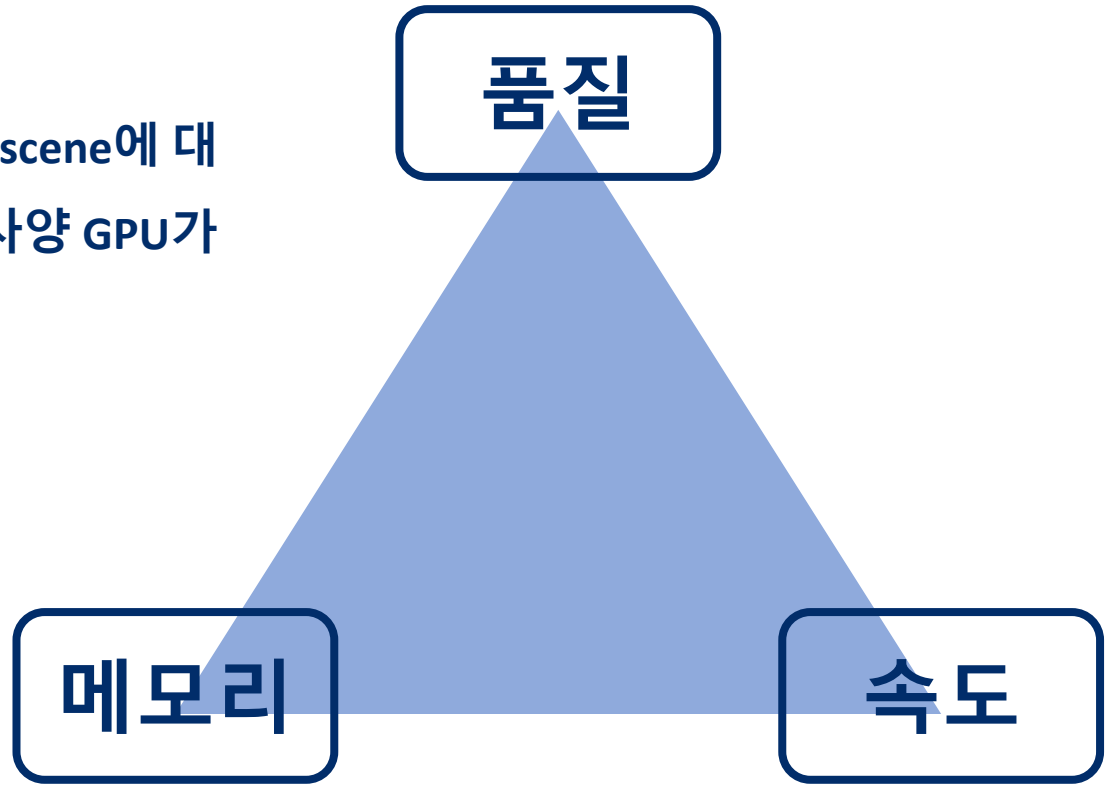
*baking: precomputing and storing

= 값들을 미리 다 계산해서 바로 읽어 쓸 수 있는 형태로 저장하는 기술



Introduction

- MeRF에서는 3D grid와 2D Planes를 결합하여 unbounded scene에 대해서도 품질을 방어하면서 빠른 속도, 적은 메모리로 고사양 GPU가 없는 기기에서 실시간 렌더링을 해보겠다.

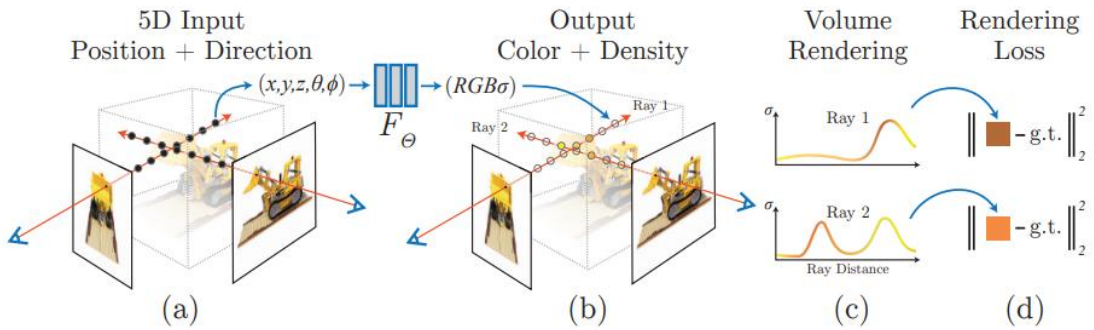


Preliminaries

Preliminaries

NeRF

- T_i : transmittance
- α_i : alpha values
- $\delta_i = t_{i+1} - t_i$: distance between adjacent samples
- τ : density



== NeRF

$$\mathbf{C} = \sum_i w_i \mathbf{c}_i\,, \quad w_i = \alpha_i T_i\,, \quad T_i = \prod_{j=1}^{i-1}(1-\alpha_j)\,, \quad \alpha_i = 1 - e^{-\tau_i \delta_i}, \quad (1)$$

투과율

불투명도

밀도와 구간이 커질수록 0에 가까워진다
→ 불투명도는 1에 가까워진다

Preliminaries

SNeRG

SNeRG uses a **deferred shading model** in which the radiance field is decomposed into

a 3D field of **densities τ**, **diffuse RGB colors Cd** , and **feature vectors f**

*deferred shading model: it can reduce the number of MLP evaluations to one per ray

Deferred shading model

- **보는 각도(Direction)에 따른 색상 계산을 언제, 몇 번 하느냐의 차이**
- NeRF :
  - 광선을 따라 점을 샘플링할 때마다 → 광선 하나당 무수히 많이
- SNeRG :
  - 광선 위의 모든 점들의 고유 색상과 특징만 합친 후, 마지막에 각도에 따른 반사광 연산

Preliminaries

SNeRG

수식 (2) : Accumulation

$w_i = \alpha_i T_i$: 가중치

$$\mathbf{C}_d = \sum_i w_i \mathbf{c}_{d,i}, \quad \mathbf{F} = \sum_i w_i \mathbf{f}_i, \tag{2}$$

수식 (3) : Deferred Shading (반사광)

$$\mathbf{C} = \mathbf{C}_d + h(\mathbf{C}_d, \mathbf{F}, \mathbf{d}). \tag{3}$$

픽셀의 색

MLP

Preliminaries

Mip-NeRF 360

In order for radiance fields to **render** high quality **unbounded scenes**

containing nearby objects as well as objects far from the camera

Contraction Function

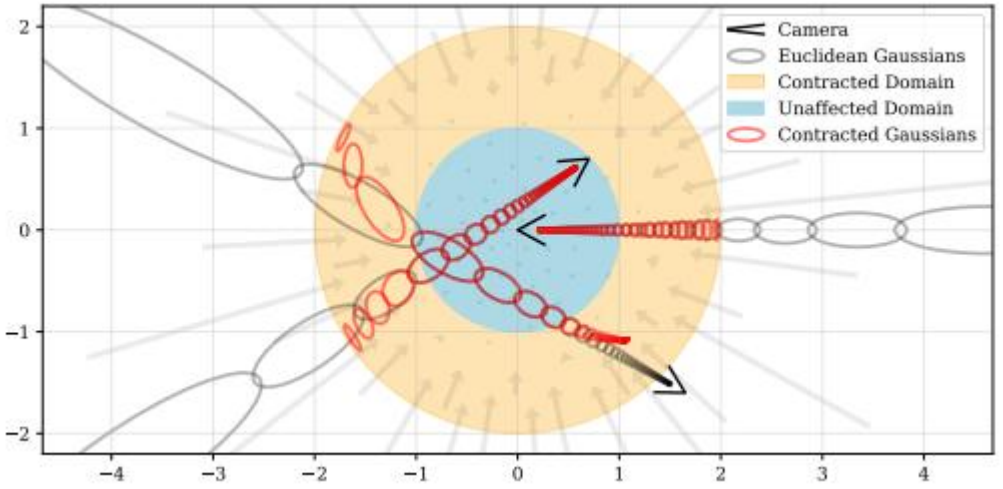
- To wrap the unbounded scene domain into a finite sphere

카메라 기준 거리 1 이내는 그대로

$$\text{contract}(\mathbf{x}) = \begin{cases} \mathbf{x} & \text{if } \|\mathbf{x}\|_2 \leq 1 \\ \left(2 - \frac{1}{\|\mathbf{x}\|_2}\right) \frac{\mathbf{x}}{\|\mathbf{x}\|_2} & \text{if } \|\mathbf{x}\|_2 > 1 \end{cases} \quad (4)$$

1~2 사이 값 방향 벡터

거리 1보다 먼 곳은 반지름이 2인 구 안쪽으로 압축

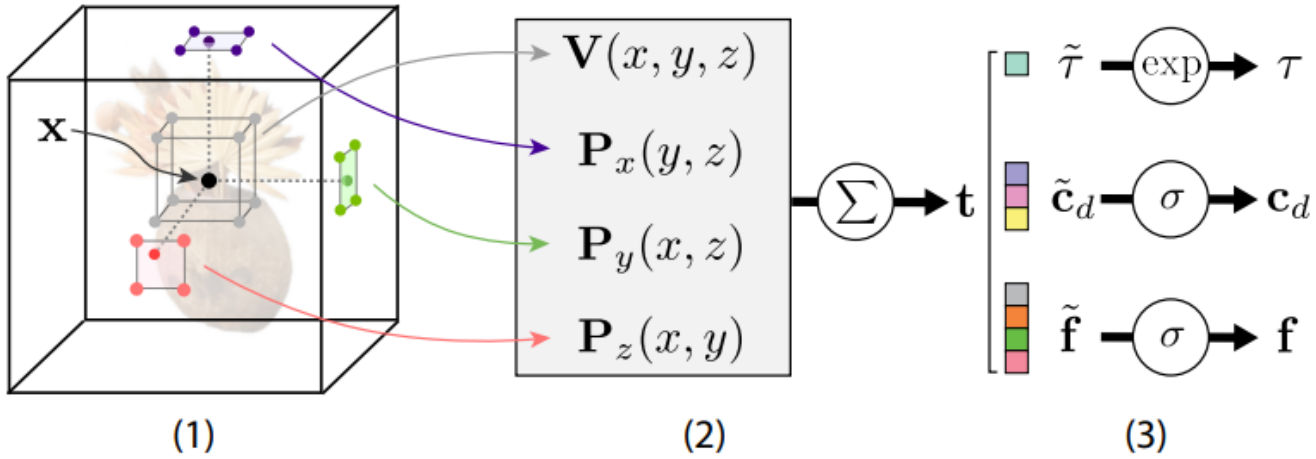


Method

Method – Volume Parameterization

- represents a scene using a 3D field of **volume densities** $\tau \in \mathbb{R}_+$, **diffuse RGB colors** $\mathbf{c_d} \in \mathbb{R}^3$, and **feature vectors** $\mathbf{f} \in \mathbb{R}^K$ **← deferred shading**
- parameterize this field with **a low-resolution 3D** $L \times L \times L$ **voxel grid** V and **three high-resolution 2D** $R \times R$ **grids** Px , Py, **and** Pz

➔V, Px , Py, **and** Pz **stores a vector with C = 4 + K channels. (C=8, L = 512, R = 2048)**

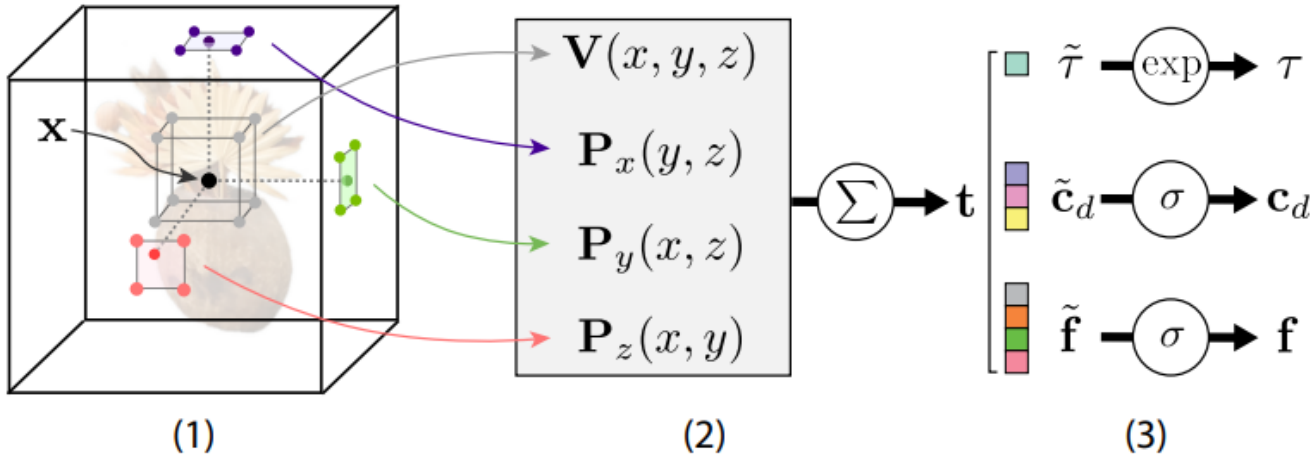


Method – Volume Parameterization

- represents a scene using a 3D field of **volume densities** $\tau \in \mathbb{R}_+$, **diffuse RGB colors** $\mathbf{c_d} \in \mathbb{R}^3$, and **feature vectors** $\mathbf{f} \in \mathbb{R}^K$ ← **deferred shading**
- parameterize this field with **a low-resolution 3D** $L \times L \times L$ **voxel grid** V and **three high-resolution 2D** $R \times R$ **grids** Px , Py, **and** Pz

밀도 1, RGB 3 ← → 특질벡터 f의 차원 K

➔V, Px , Py, **and** Pz **stores a vector with C = 4 + K channels. (C=8, L = 512, R = 2048)**



Method – Volume Parameterization

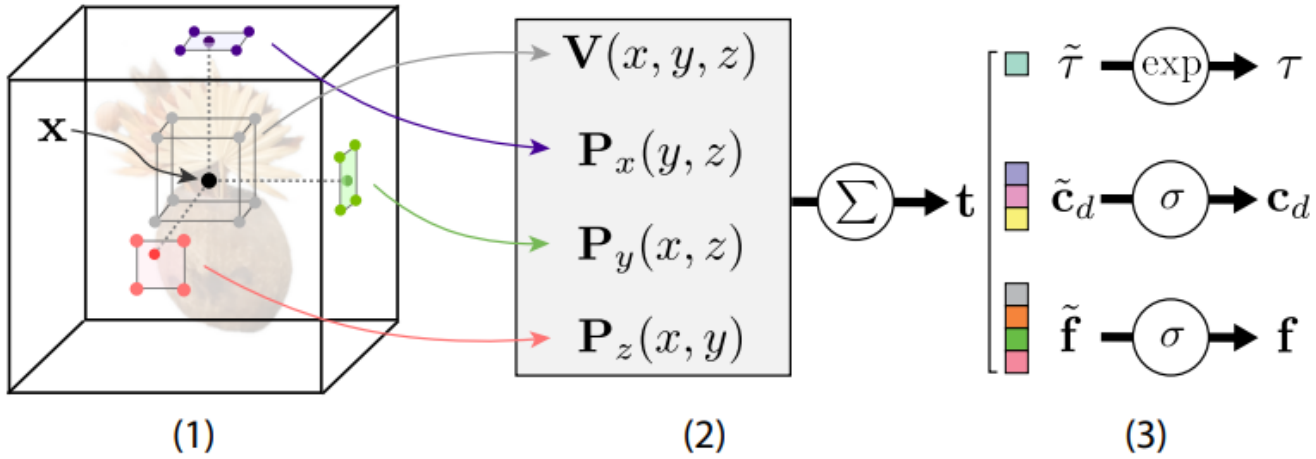
- parameterize this field with **a low-resolution 3D $L \times L \times L$ voxel grid V** and **three high-resolution 2D $R \times R$ grids Px , Py, and Pz : tri-plane**

  *Tri-plane : 3차원 공간 정보를 2차원 평면 3장으로 분해해서 표현하는 방식

➔3D voxel grid와 3개의 2D grid로 나누는 이유:

- **3D voxel grid 모든 정보를 담으면 크기가 매우 커진다.**

저해상도 3D voxel grid에는 대략적인 형태나 구조, 고해상도 2D grid에는 디테일을 다룬다.



Method – Volume Parameterization

- define the continuous field of C-vectors as the **sum** of **trilinearly** interpolated vectors from the **3D grid** and **bilinearly** interpolated vectors from the three **2D grids**

1. Bilinear Interpolation (이중선형 보간)

- **어디서 쓰나: 2D 평면** (MeRF의 고해상도 2D 그리드)
- **원리:** 어떤 점 (x, y)의 값을 알기 위해, 그 점을 감싸고 있는 **네 개의 모서리(픽셀)** 값을 사용합니다.
  1. 가로 방향(x)으로 두 번 선형 보간을 하고,
  2. 그 결과를 가지고 세로 방향(y)으로 한 번 더 선형 보간을 합니다.
  - **"Bi(2) + Linear"**: 선형 보간을 두 방향으로 수행한다고 해서 '이중'입니다.
- **쉽게 말해:** 4개의 기둥 사이에 해먹을 걸어놓았을 때, 해먹 위 특정 지점의 높이를 4개 기둥 높이를 적절히 섞어서 구하는 것과 같습니다.

2. Trilinear Interpolation (삼중선형 보간)

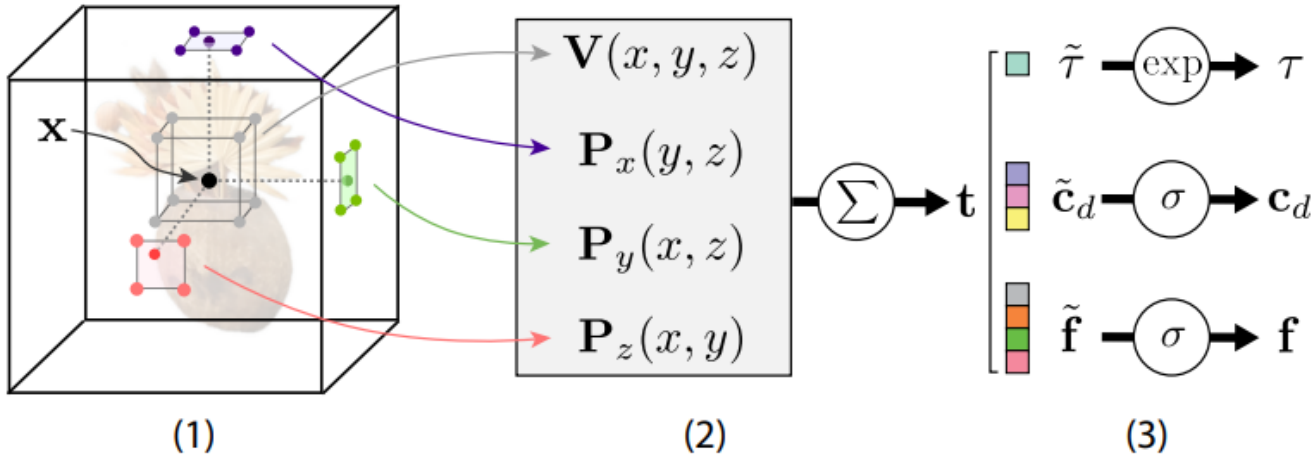
- **어디서 쓰나: 3D 공간** (MeRF의 저해상도 3D 그리드)
- **원리:** 어떤 점 (x, y, z)의 값을 알기 위해, 그 점을 감싸고 있는 **여덟 개의 모서리(복셀 꼭짓점)** 값을 사용합니다.
  1. x축 방향으로 보간 (4회)
  2. y축 방향으로 보간 (2회)
  3. z축 방향으로 보간 (1회)
  - **"Tri(3) + Linear"**: 선형 보간을 세 방향(x, y, z)으로 수행한다고 해서 '삼중'입니다.
- **쉽게 말해:** 정육면체 방 안의 공기 온도를 잴 때, 방의 8개 구석에 있는 온도계 값을 거리에 비례해서 섞는 것입니다.

선형 보간

Method – Volume Parameterization

- define the continuous field of C-vectors as the **sum** of **trilinearly** interpolated vectors from the **3D grid** and **bilinearly** interpolated vectors from the three **2D grids**

$$\mathbf{t}(x, y, z) = \mathbf{V}(x, y, z) + \mathbf{P}_x(y, z) + \mathbf{P}_y(x, z) + \mathbf{P}_z(x, y), \qquad (5)$$

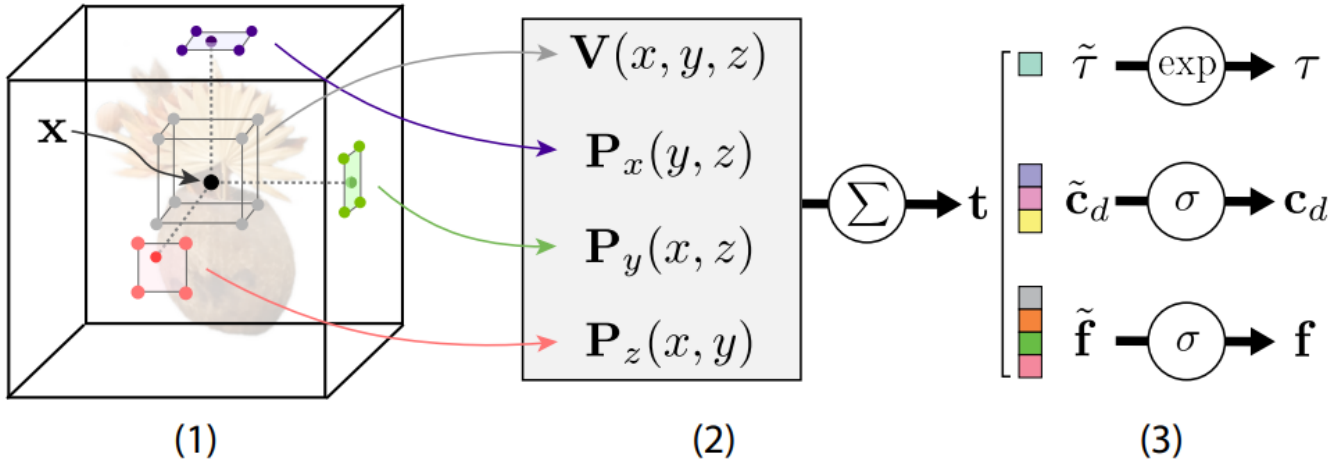


$$\mathbf{P}_i : \mathbb{R}^2 \rightarrow \mathbb{R}^C$$
$$\mathbf{V} : \mathbb{R}^3 \rightarrow \mathbb{R}^C$$

Method – Volume Parameterization

- split the C-vector at any 3D location into three components corresponding to **density** $\tilde{\tau} \in \mathbb{R}$
- , **diffuse color** $\tilde{\mathbf{c}}_d \in \mathbb{R}^3$, and **view-dependence feature** $\tilde{\mathbf{f}} \in \mathbb{R}^K$

$$\mathbf{t}(x, y, z) = \mathbf{V}(x, y, z) + \mathbf{P}_x(y, z) + \mathbf{P}_y(x, z) + \mathbf{P}_z(x, y), \qquad (5)$$

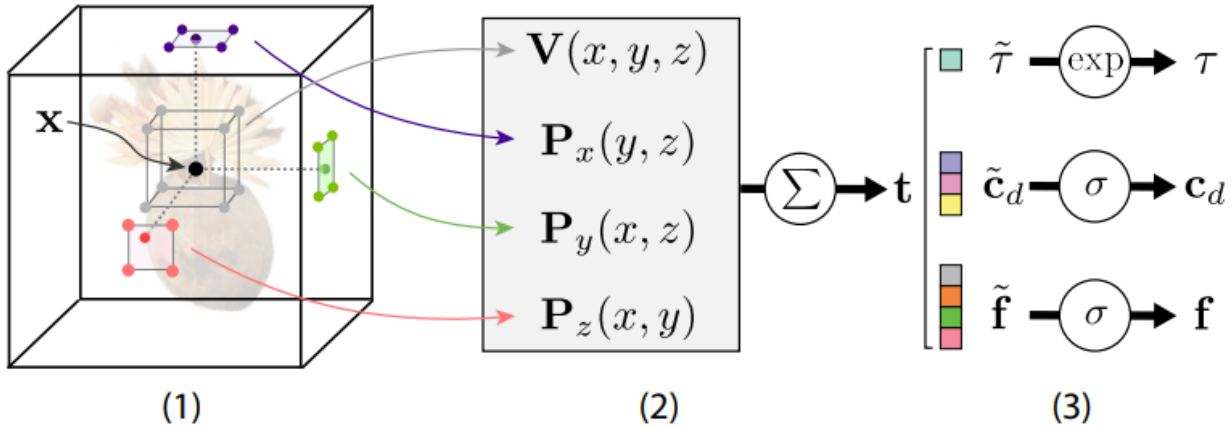


Method – Volume Parameterization

- split the C-vector at any 3D location into three components corresponding to **density** $\tilde{\tau} \in \mathbb{R}$
- , **diffuse color** $\tilde{\mathbf{c}}_d \in \mathbb{R}^3$, and **view-dependence feature** $\tilde{\mathbf{f}} \in \mathbb{R}^K$
- Then, apply **nonlinear functions** to obtain the three values

$$\tau = \exp(\tilde{\tau}), \quad \mathbf{c_d} = \sigma(\tilde{\mathbf{c}}_d), \quad \mathbf{f} = \sigma(\tilde{\mathbf{f}}), \tag{6}$$

Sigmoid fucntion

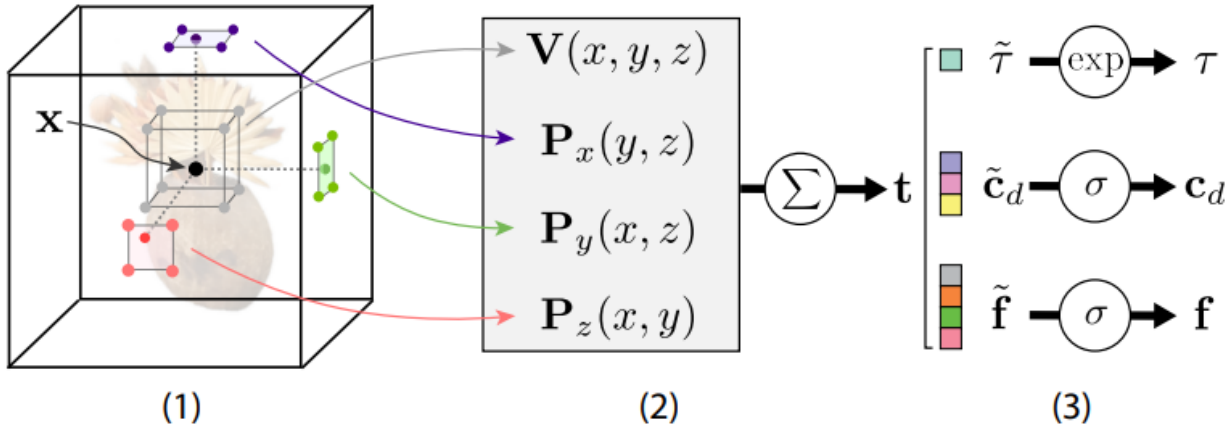


Method – Volume Parameterization

- split the C-vector at any 3D location into three components corresponding to **density** $\tilde{\tau} \in \mathbb{R}$
- , **diffuse color** $\tilde{\mathbf{c}}_d \in \mathbb{R}^3$, and **view-dependence feature** $\tilde{\mathbf{f}} \in \mathbb{R}^K$
- Then, apply **nonlinear functions** to obtain the three values

$$\tau = \exp(\tilde{\tau}), \quad \mathbf{c_d} = \sigma(\tilde{\mathbf{c}}_d), \quad \mathbf{f} = \sigma(\tilde{\mathbf{f}}), \tag{6}$$

Sigmoid fucntion



$$\mathbf{C}_d = \sum_i w_i \mathbf{c}_{d,i}, \quad \mathbf{F} = \sum_i w_i \mathbf{f}_i, \tag{2}$$

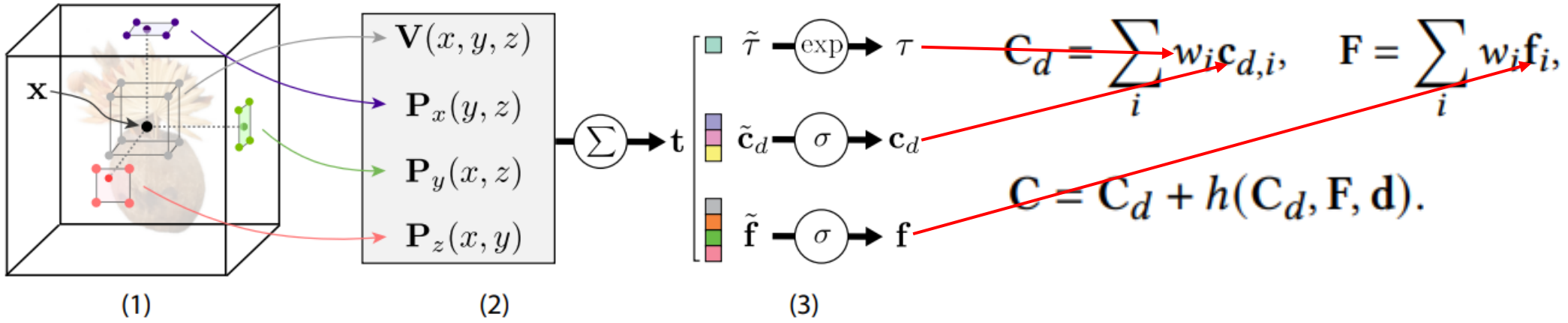
$$\mathbf{C} = \mathbf{C}_d + h(\mathbf{C}_d, \mathbf{F}, \mathbf{d}). \tag{3}$$

Method – Volume Parameterization

- split the C-vector at any 3D location into three components corresponding to **density** $\tilde{\tau} \in \mathbb{R}$
- , **diffuse color** $\tilde{\mathbf{c}}_d \in \mathbb{R}^3$, and **view-dependence feature** $\tilde{\mathbf{f}} \in \mathbb{R}^K$
- Then, apply **nonlinear functions** to obtain the three values

$$\tau = \exp(\tilde{\tau}), \quad \mathbf{c_d} = \sigma(\tilde{\mathbf{c}}_d), \quad \mathbf{f} = \sigma(\tilde{\mathbf{f}}), \tag{6}$$

Sigmoid fucntion



$$\mathbf{C}_d = \sum_i w_i \mathbf{c}_{d,i}, \quad \mathbf{F} = \sum_i w_i \mathbf{f}_i, \tag{2}$$

$$\mathbf{C} = \mathbf{C}_d + h(\mathbf{C}_d, \mathbf{F}, \mathbf{d}). \tag{3}$$

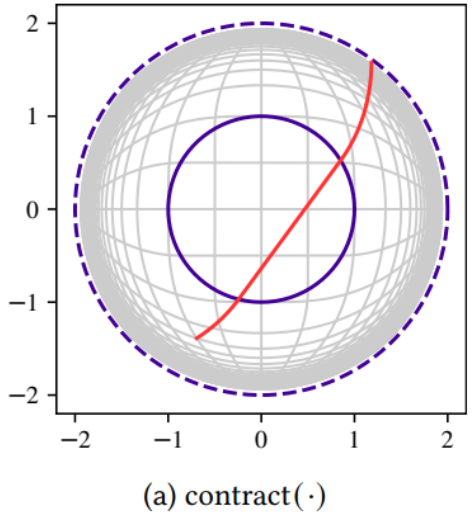
Method – Piecewise-projective Contraction

Mip-NeRF 360's contraction function

*contraction function : maps large far-away regions of space into small regions in contracted space

- The contraction function of Mip-NeRF **non-linearly** maps any point in space $\mathbf{x} \in \mathbb{R}^3$ into a radius-2 ball, and represents the scene within this contracted space.
- Mip-NeRF 360's contraction function **does not preserve straight lines.**

→makes ray-AABB(axis-aligned bounding box) intersections **challenging** to compute.(실시간 렌더링을 어렵게)

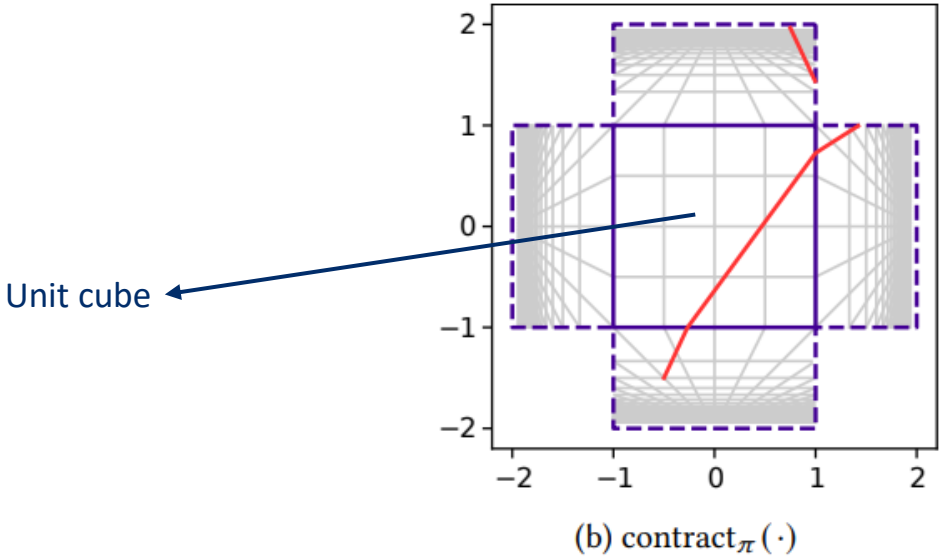


(a) contract(·)

Method – Piecewise-projective Contraction

MeRF's contraction function

- propose a novel contraction function for which **ray-AABB intersections** can be computed trivially.
- it contains **seven regions**, and within each region, it is a projective transformation.
- The **unit cube** $\|x\|_\infty$ is preserved, and the other six regions defined by the coordinate maximizing $|x_j|$ and its sign each get mapped by a different projective transformation.



(b) $\text{contract}_\pi(\cdot)$

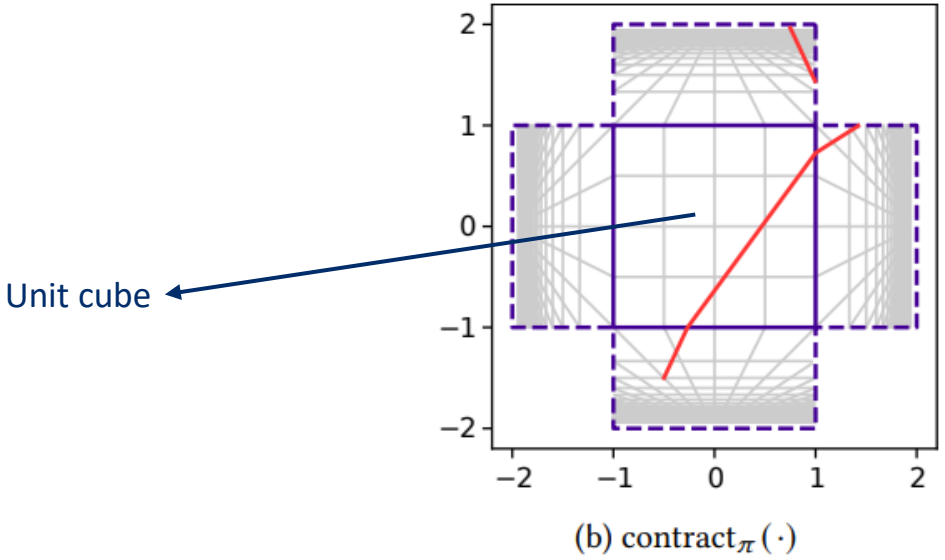
$$\text{contract}_\pi(\mathbf{x})_j = \begin{cases} x_j & \text{if } \|\mathbf{x}\|_\infty \le 1 \\ \frac{x_j}{\|\mathbf{x}\|_\infty} & \text{if } x_j \ne \|\mathbf{x}\|_\infty > 1 \\ \left(2 - \frac{1}{|x_j|}\right)\frac{x_j}{|x_j|} & \text{if } x_j = \|\mathbf{x}\|_\infty > 1 \end{cases}, \quad (7)$$

$\|\cdot\|_\infty$ is the L_∞ norm ($\|\mathbf{x}\|_\infty = \max_j |x_j|$)

Method – Piecewise-projective Contraction

MeRF's contraction function

- propose a novel contraction function for which **ray-AABB intersections** can be computed trivially.
- it contains **seven regions**, and within each region, it is a projective transformation.
- The unit cube $\|x\|_\infty$ is preserved, and the other six regions defined by the coordinate maximizing $|x_j|$ and its sign each get mapped by a different projective transformation.



(b) $\text{contract}_\pi(\cdot)$

최대 거리x, 길이 >= 1 : 원근법 적용

길이 <= 1 : unit cube

$$\text{contract}_\pi(\mathbf{x})_j = \begin{cases} x_j & \text{if } \|\mathbf{x}\|_\infty \le 1 \\ \frac{x_j}{\|\mathbf{x}\|_\infty} & \text{if } x_j \neq \|\mathbf{x}\|_\infty > 1 \\ \left(2 - \frac{1}{|x_j|}\right)\frac{x_j}{|x_j|} & \text{if } x_j = \|\mathbf{x}\|_\infty > 1 \end{cases}, \quad (7)$$

(1~2) 부호화

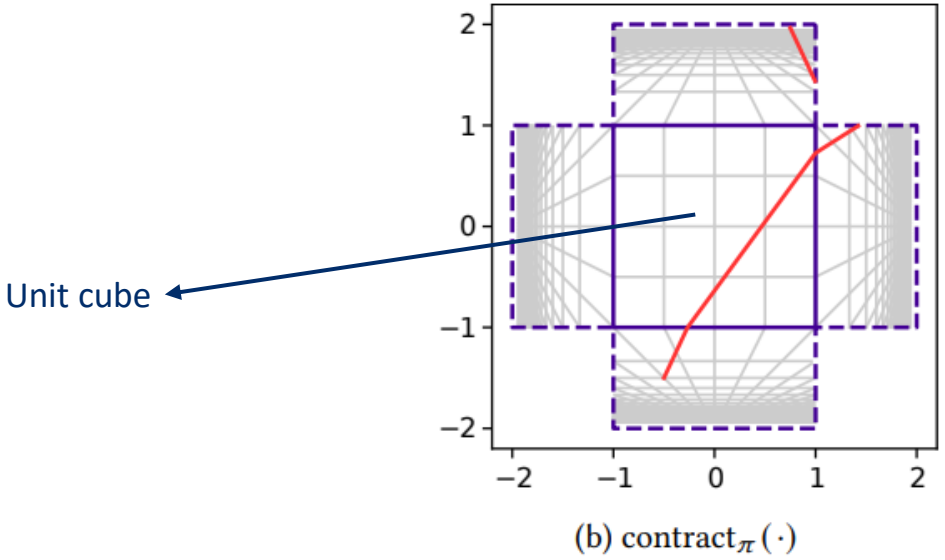
$\|\cdot\|_\infty$ is the L_∞ norm ($\|\mathbf{x}\|_\infty = \max_j |x_j|$)

Method – Piecewise-projective Contraction

MeRF's contraction function

- projective transformations preserve **straight lines**, and the only **discontinuities** in its direction are on the **boundaries** between the seven regions.
- This allows us to use standard **ray-AABB intersection**.

  → **empty space skipping**로 **렌더링 속도 증가**



(b) $\text{contract}_\pi(\cdot)$

최대 거리x, 길이 >= 1 : 원근법 적용

길이 <= 1 : unit cube

$$\text{contract}_\pi(\mathbf{x})_j = \begin{cases} x_j & \text{if } \|\mathbf{x}\|_\infty \le 1 \\ \frac{x_j}{\|\mathbf{x}\|_\infty} & \text{if } x_j \neq \|\mathbf{x}\|_\infty > 1 \\ \left(2 - \frac{1}{|x_j|}\right)\frac{x_j}{|x_j|} & \text{if } x_j = \|\mathbf{x}\|_\infty > 1 \end{cases}, \quad (7)$$

(1~2) 부호화

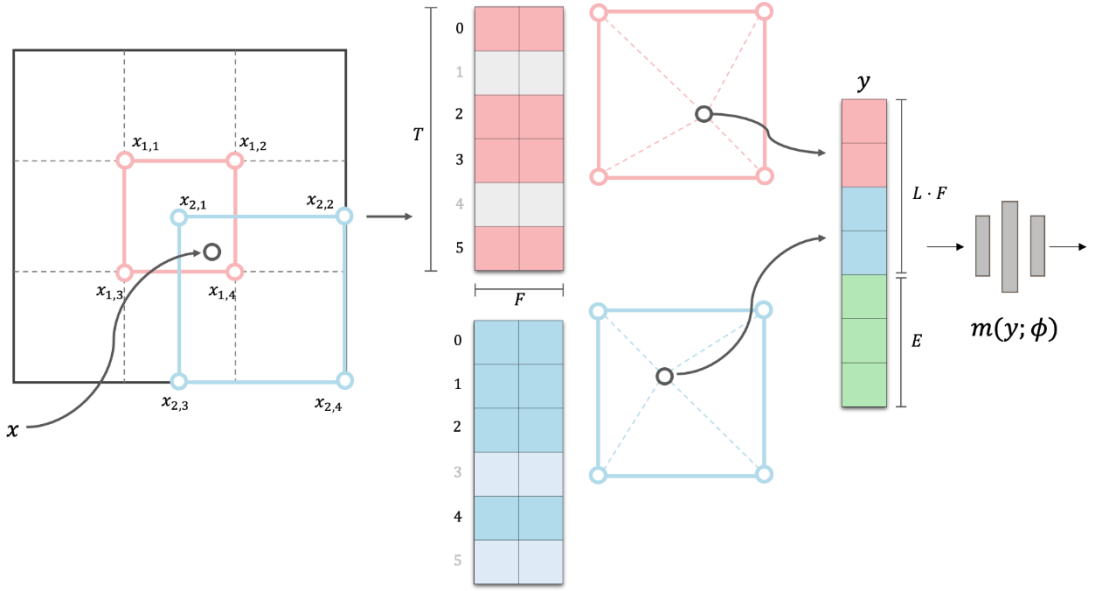
$\|\cdot\|_\infty$ is the L_∞ norm ($\|\mathbf{x}\|_\infty = \max_j |x_j|$)

Method – Training

MeRF Training

- Significantly **more memory** is consumed during **training than** during **rendering** because training requires **intermediate activations** for the **backpropagation** and the **Adam optimizer**.
- In MeRF, training requires more than **twelve times** as much video memory as rendering.

  → it should be **optimized** with a **multi-resolution hash encoding**.



Method – Training

MeRF Training

- baking the MLPs' outputs onto discrete grids for rendering introduces a mismatch between the representations used for training and for rendering.

→ 학습은 MLP (연속 함수), 렌더링은 baked grid(이산화된 함수)

→ MLP가 학습한 곡선을 그리드의 직선으로 바꾸면 오차 발생

방법: MLP가 grid처럼 학습하도록.

Method – Training

NeRF Training

- MLP에 좌표 x를 직접 입력

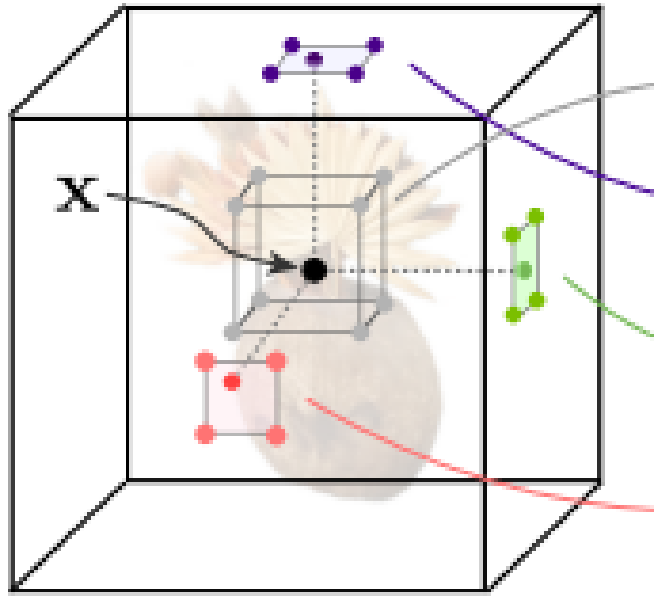
  → f(x) 출력

MeRF Training

- 3D는 8개, 2D는 4개의 Virtual Grid Corners를 찾아서, MLP에 입력

  → 2D: $f(x_1, x_2, x_3, x_4,)$, 3D: $f(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$ 출력

방법: Linear interpolation을 MLP의 학습 단계에서 주입.



Method – Training

NeRF Training

- MLP에 좌표 x를 직접 입력

  → f(x) 출력

MeRF Training

- 3D는 8개, 2D는 4개의 **Virtual Grid Corners**를 찾아

  → 2D: $f(x_1, x_2, x_3, x_4)$, 3D: $f(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$

방법: Linear interpolation을 MLP의 학습 단계에서 주

2. 코너를 찾는 수학적 과정 (Snap to Grid)

이제 어떤 실수 좌표 x가 들어왔을 때, 가상의 코너를 어떻게 계산하는지 **1차원 예시**로 보여드릴게요.

- **설정:** 그리드 해상도 $L = 100$ (전체 공간 0~1 사이를 100칸으로 쪼갬)
- **입력:** 현재 샘플링된 점 $x = 0.555$ (실수)

이 점을 감싸는 **왼쪽 코너(x_{left})**와 **오른쪽 코너(x_{right})**를 어떻게 찾을까요?

1. **뻥튀기 (Scale):** 입력값에 해상도를 곱합니다.

$$0.555 \times 100 = 55.5$$

2. **내림 (Floor - 정수 인덱스 찾기):** 소수점을 버려서 **왼쪽 칸 번호**를 찾습니다.

$$\lfloor 55.5 \rfloor = 55$$

(이게 왼쪽 코너의 인덱스입니다.)

$$55 + 1 = 56$$

(이게 오른쪽 코너의 인덱스입니다.)

3. **원상복구 (Normalize - 좌표로 변환):** 다시 해상도로 나눠서 실제 좌표로 만듭니다.

$$x_{left} = 55/100 = \mathbf{0.55}$$

$$x_{right} = 56/100 = \mathbf{0.56}$$

결과: 우리는 0.555라는 점이 0.55**와** 0.56**이라는 두 가상의 그리드 코너** 사이에 있다는 것을 수학적으로 계산해냈습니다. (데이터를 조회한 게 아니라 계산한 것입니다.)

Method – Training

Proposal MLP

- 빈 공간이 많은 unbounded scene에서는 모든 곳을 샘플링하면 시간 ↑
- Mip-NeRF처럼 Proposal MLP로 Hierarchical Sampling하여 장면의 Density 분포를 파악하여 물체가 있을 확률이 높은 곳에만 샘플을 집중

- Proposal MLP

Training

- 샘플링할 위치를 학습한다.

Baking

- Proposal MLP가 물체가 있다고 한 voxel만 선별하여 3D Grid로 bake

Rendering

- Rendering 시에는 사용 X

Method – Training

Quantization: Voxel의 메모리를 줄이기 위한 작업, 실수 → 1Byte 정수

- **quantize** each of the C dimensions at **every location** in the grid to **a single byte** to reduce memory consumption **at render time**
- **simply quantizing** the optimized grid values after training leads to a **drop** in **rendering quality** because of **mismatches** between the **optimized model** and the **one used for rendering**

  → optimization 중에 모든 위치의 C values를 quantize하고 학습하자.

  == 학습완료된 값을 **Quantization** 하지말고, **Quantization**된 값으로 학습하자**.**

Method – Training

Quantization

- **quantization function q**

반올림된 값

$$q(x) = x + \not\nabla\left(\frac{\lfloor (2^8 - 1)x + {}^1\!/\!_2 \rfloor}{2^8 - 1} - x\right), \qquad (9)$$

1Byte이므로, 8bit → 255 까지의 값

Stop-gradient:
Forward는 반올림된 값 → 오차가 있는 것으로 계산하여 Loss 계산
Backward는 q(x) = x로 취급하여 기울기 1로 취급하여 학습진행
이유: 반올림 함수는 계단식 모양의 함수여서 기울기가 무한대, 0 반복이기 때문에 학습이 불가하다.

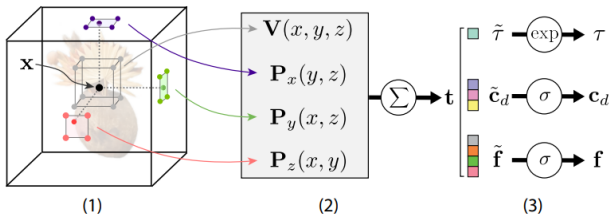
Method – Training

Quantization

- **nonlinearly** map them to lie in [0, 1] using a **sigmoid** σ, then **quantize** them to **a single byte** using a quantization function q, and finally **affinely map** the result to the range $[-m,m]$

$$q(x) = x + \cancel{\nabla}\left(\frac{\lfloor (2^8-1)x + 1/2 \rfloor}{2^8-1} - x\right), \tag{9}$$

$$\tilde{\mathbf{t}}' = 2m \cdot q\left(\sigma\left(\tilde{\mathbf{t}}\right)\right) - m, \tag{8}$$



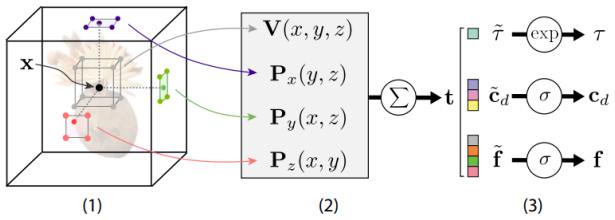
Method – Training

Quantization

- **nonlinearly** map them to lie in [0, 1] using a **sigmoid** σ, then **quantize** them to **a single byte** using a quantization function q, and finally **affinely map** the result to the range $[-m,m]$

$$q(x) = x + \nabla\left(\frac{\lfloor (2^8-1)x + 1/2 \rfloor}{2^8-1} - x\right), \tag{9}$$

$$\tilde{\mathbf{t}}' = 2m \cdot q\left(\sigma\left(\tilde{\mathbf{t}}\right)\right) - m, \tag{8}$$



Sigmoid함수로 0~1 사이로 값을 바꿨으니 다시 늘리는 과정

Method – Baking

Baking

- **After training**, evaluate and **store the MLP's outputs** on discrete grids for real-time rendering.
- Binary Occupancy Grid **A**
  - Proposal MLP가 제안한 샘플 위치(x_i)가 아래의 둘 다 만족하면 주변 **8개**(2x2x2) **Voxel**을 occupie로 저장
  - 1. w_i > 0.005 (앞에 가려지지 않고 기여도가 있는지)
  - 2. α_i > 0.005 (충분히 진한지)

→ Baking **3D** Voxel Grid in **Block-sparse format** and three high-resolution **2D Planes** using **binary grid A** + Occupancy Grid(grid **A**)를 Max-pooling하여 Multi-resolution Hierarchy를 생성하여 렌더링 때 empty space Skipping 사용.

Method – Rendering

Real-Time Rendering

- employ a **multi-resolution hierarchy** of **occupancy grids**.
- The set of occupancy grids is created by **max-pooling** the full-resolution binary grid **A.**
- leverage this **multi-resolution hierarchy** of occupancy grids for **faster space skipping**.

- query the occupancy grids in a **coarse-to-fine** manner.

→ If any level indicates the voxel as **empty,** then **skip** the volume

→ only access when occupancy grid levels are **occupied**.

→ **terminate** ray marching when $\boldsymbol{Ti}$(transmittance value) falls below 2 × 10^4.

Method – Rendering

Real-Time Rendering

- employ a **multi-resolution hierarchy** of **occupancy grids**.
- The set of occupancy grids is created by **max-pooling** the full-resolution binary grid **A.**
- leverage this **multi-resolution hierarchy** of occupancy grids for **faster space skipping**.

- query the occupancy grids in a **coarse-to-fine** manner.

→ If any level indicates the voxel as **empty,** then **skip** the volume

→ only access when occupancy grid levels are **occupied**.

→ **terminate** ray marching when $\boldsymbol{Ti}$(transmittance value) falls below 2 × 10^4.

Experiment

Experiment

	Outdoor			Indoor		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [Mildenhall et al. 2020]	21.46	0.458	0.515	26.84	0.790	0.370
NeRF++ [Zhang et al. 2020]	22.76	0.548	0.427	28.05	0.836	0.309
SVS [Riegler and Koltun 2021]	23.01	0.662	**0.253**	28.22	0.907	**0.160**
Mip-NeRF 360 [Barron et al. 2022]	**24.47**	**0.691**	0.283	**31.72**	**0.917**	0.180
Instant-NGP [Müller et al. 2022]	22.90	0.566	0.371	29.15	0.880	0.216
Deep Blending [Hedman et al. 2018]	21.54	0.524	0.364	26.40	0.844	0.261
Mobile-NeRF [Chen et al. 2022a]	21.95	0.470	0.470	–	–	–
Ours	**23.19**	**0.616**	**0.343**	**27.80**	**0.855**	**0.271**

Table 1. Quantitative results of our model on all scenes from Mip-NeRF 360 [Barron et al. 2022]. Models that render in real-time are highlighted. Mobile-NeRF did not evaluate on the "indoor" scenes, so those metrics are absent.

Experiment

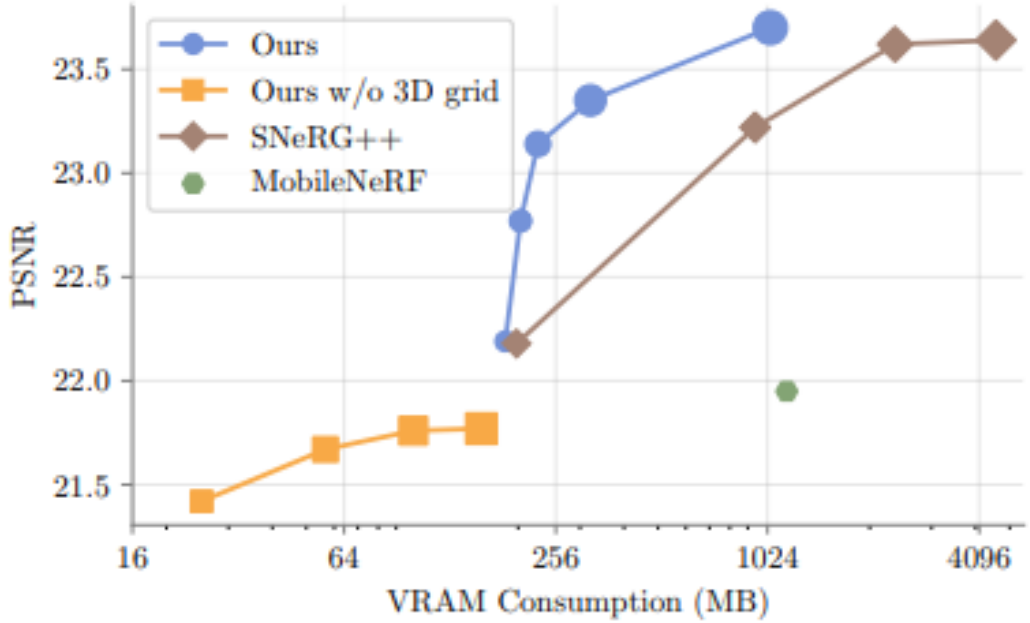
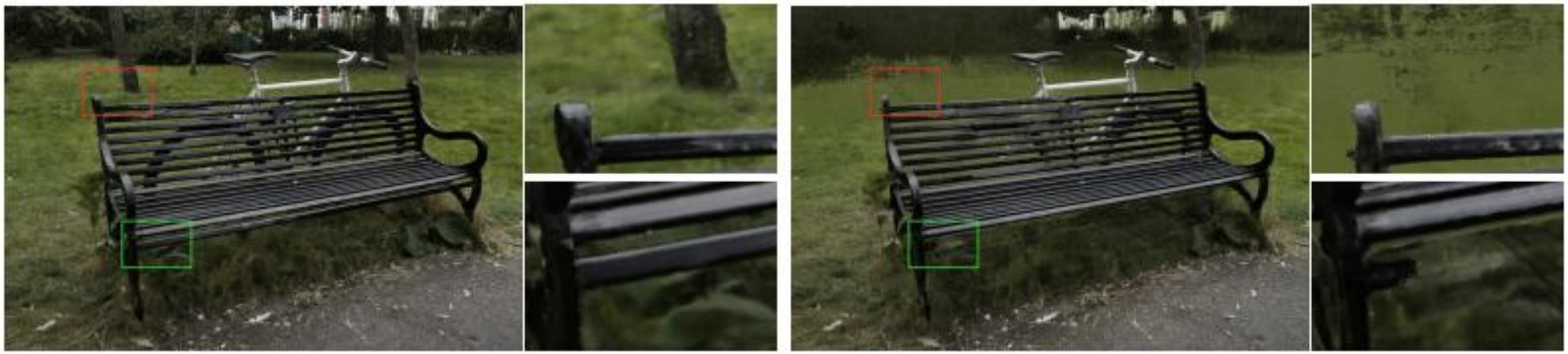


Fig. 4. PSNR (higher is better) vs VRAM consumption (lower is better) for our model, our improved SNeRG baseline, MobileNeRF, and an ablation of our model without 3D grids. Each line besides MobileNeRF represents the same model with a varying resolution of its underlying spatial grid, indicated by marker size.

Experiment

Ablation Study



With low-resolution 3D grid

Without low-resolution 3D grid

Fig. 7. Visual comparison between MERF with and without a low-resolution 3D voxel grid (both models use three high-resolution 2D grids). Omitting the 3D grid often results in parts of the scene being poorly reconstructed; note the missing background in this example.

Experiment

Ablation Study

	PSNR ↑	SSIM ↑	LPIPS ↓
(a) Pre-baking	23.20	0.620	0.336
(b) w/o quant.-aware training	22.64	0.603	0.347
(c) Spherical contraction	23.22	0.619	0.341
Ours (Post-baking)	23.19	0.616	0.343

Table 2. We compare our final model after baking to the model before baking (a) demonstrating that our Proposal-MLP-aware baking pipeline is almost lossless. Omitting quantization-aware training (b) leads to a drop in rendering quality. Our proposed contraction function performs on par with the original spherical contraction function (c), while enabling efficient ray-AABB intersection tests. Results are averaged over all outdoor scenes.

렌더링에서 baking 하기 전. 즉, mlp를 렌더링으로 사용했을 때와 비교해도 손실이 별로 없다

실시간 렌더링을 위해 AABB를 하기 위한 piecewise-projection을 해도 손실이 별로 없다

Experiment

	PSNR ↑	SSIM ↑	LPIPS ↓	VRAM ↓	DISK ↓	FPS ↑
MacBook M1 Pro, 1280×720						
Mobile-NeRF	21.95	0.470	0.470	1162	345	**65.7**
SNeRG++	**23.64**	**0.672**	**0.285**	4571	3785	18.7
Ours	23.19	0.616	0.343	**524**	**188**	28.3
NVIDIA RTX 3090, 1920×1080						
Instant-NGP	22.90	0.566	0.371	–	**107**	4
Ours	**23.19**	**0.616**	**0.343**	**524**	188	**119**

Table 3. Performance comparison on the "outdoor" scenes.

Experiment

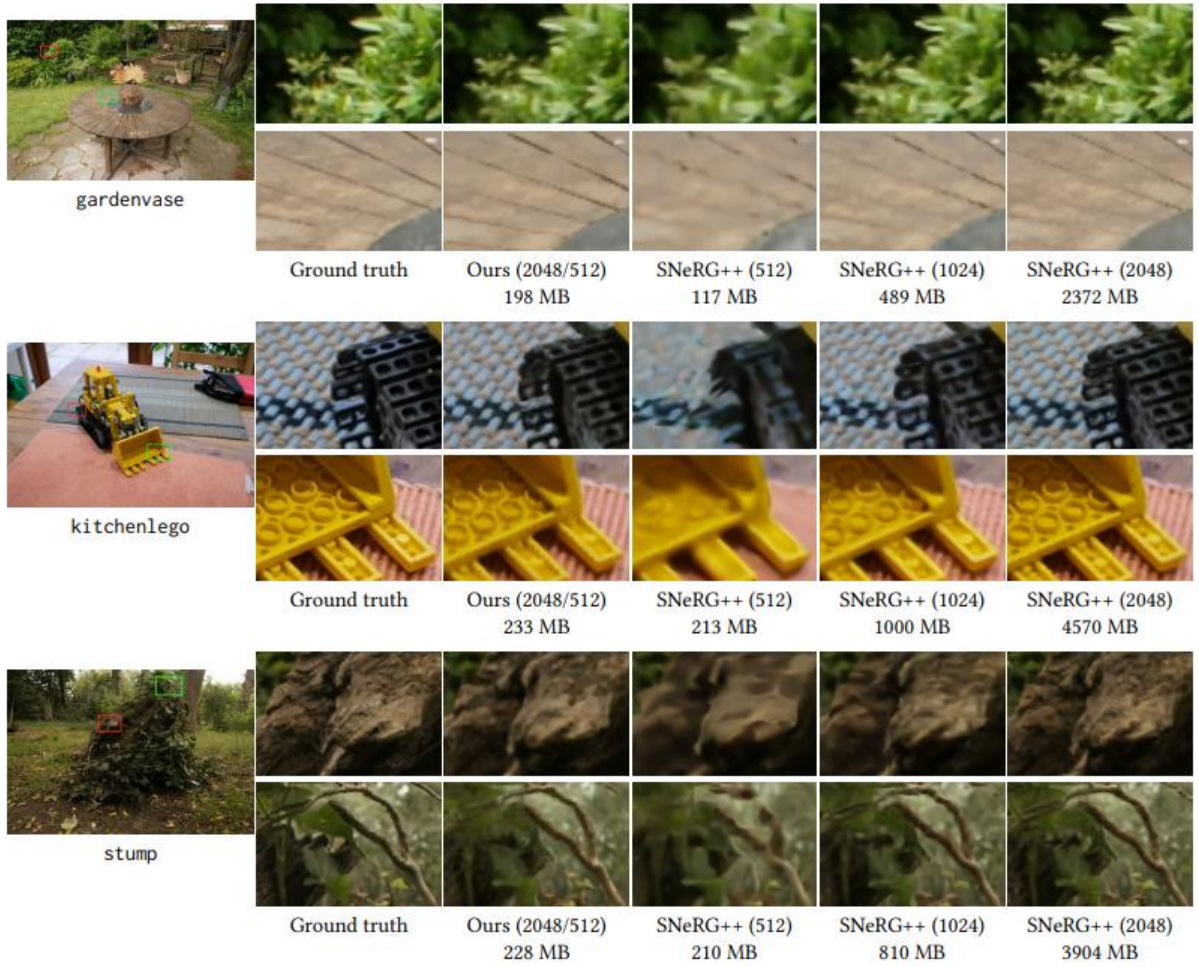


Fig. 5. Visual comparison between MERF and various resolution SNeRG++ [Hedman et al. 2021] models. Total VRAM (GPU memory) usage during rendering is listed beneath each method name. Only SNeRG++ (512) has comparable size to our model, whereas SNeRG++ (1024) and SNeRG++ (2048) are significantly larger.

Experiment

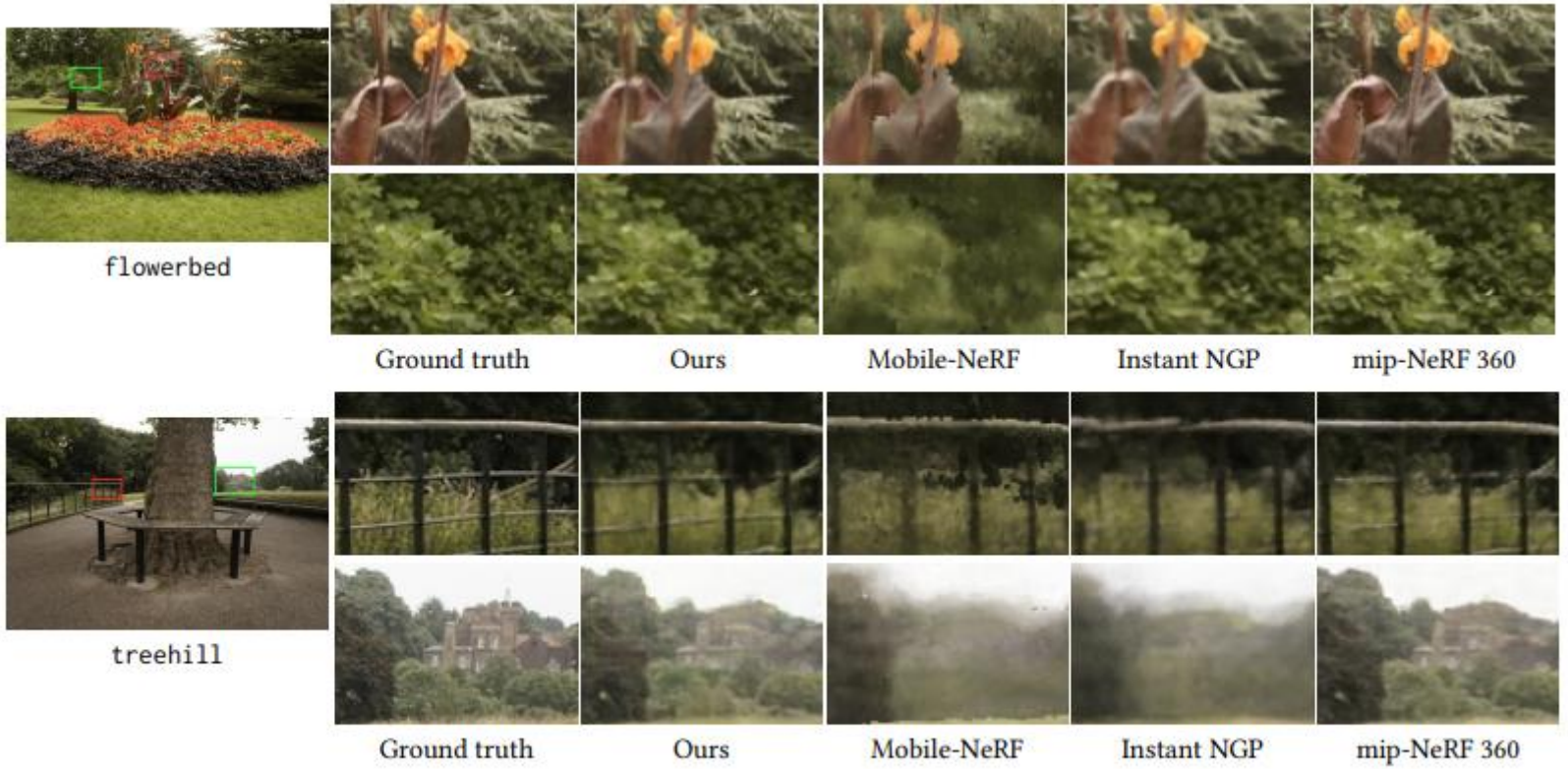


Fig. 6. Visual comparison between MERF and other view synthesis methods. Mobile-NeRF [Chen et al. 2022a] is the only other real-time method (30fps or better). Instant NGP [Müller et al. 2022] runs at interactive rates (around 5fps) and mip-NeRF 360 [Barron et al. 2022] is an extremely heavyweight offline method (around 30 seconds to render a single frame), representing the current state-of-the-art view synthesis quality.

Conclusion

Conclusion

Limitation

- Limited rendering of semi-transparent objects due to the use of the deferred shading model, which evaluates view-dependent color only once per ray

- Because a tiny MLP for real-time processing, it is difficult to scale to much larger scenes

$$\mathbf{C} = \mathbf{C}_d + h(\mathbf{C}_d, \mathbf{F}, \mathbf{d}). \quad (3)$$

- Limited or lower-performance devices like **mobile phones or headsets**

Conclusion

Contribution

- It presents a method that enables real-time rendering of large-scale unbounded scenes directly in a web browser.
- It proposes a hybrid representation combining a 3D grid and 2D tri-planes, reducing memory consumption
- It introduces a piecewise-projective contraction function enabling efficient Ray-AABB intersection tests