Medical Imaging & Intelligent Reality Lab.
Convergence Medicine/Radiology

서울아산병원 의공학연구소

M.E 박주영

Contents

Paper - Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

THOMAS MÜLLER, NVIDIA, Switzerland
ALEX EVANS, NVIDIA, United Kingdom
CHRISTOPH SCHIED, NVIDIA, USA
ALEXANDER KELLER, NVIDIA, Germany

`https://nvlabs.github.io/instant-ngp`

Main Subject: Input Encoding in Neural graphics primitives

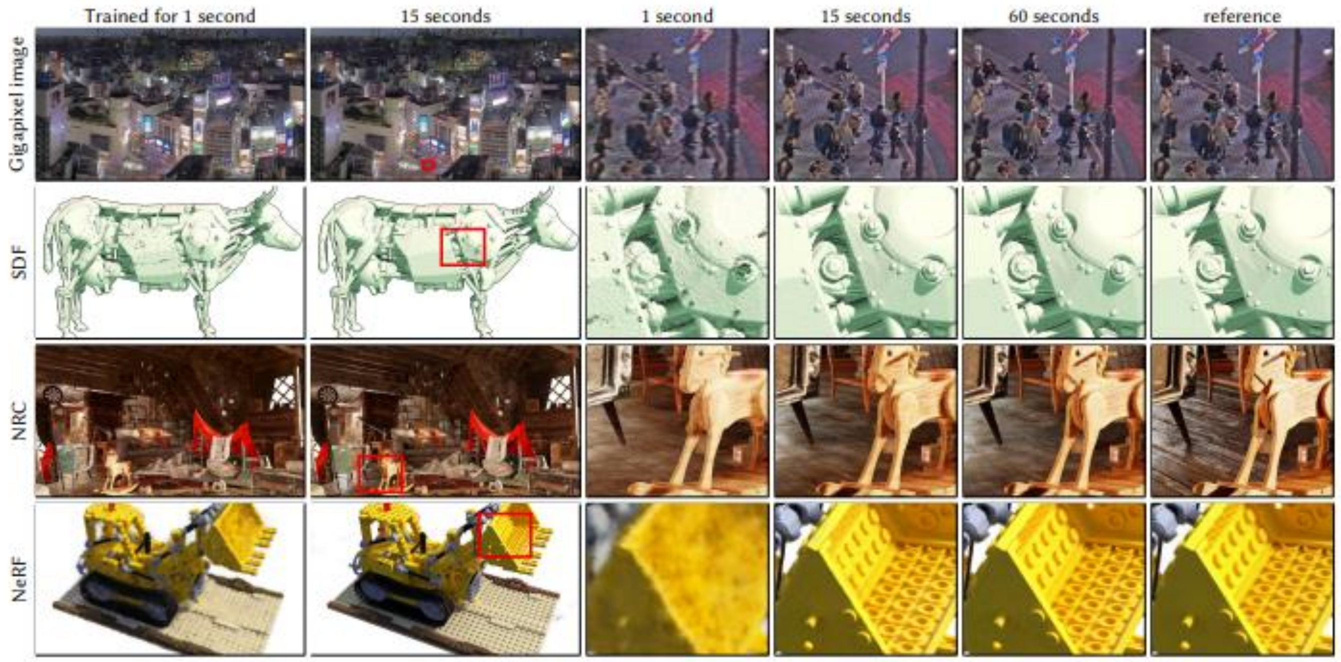
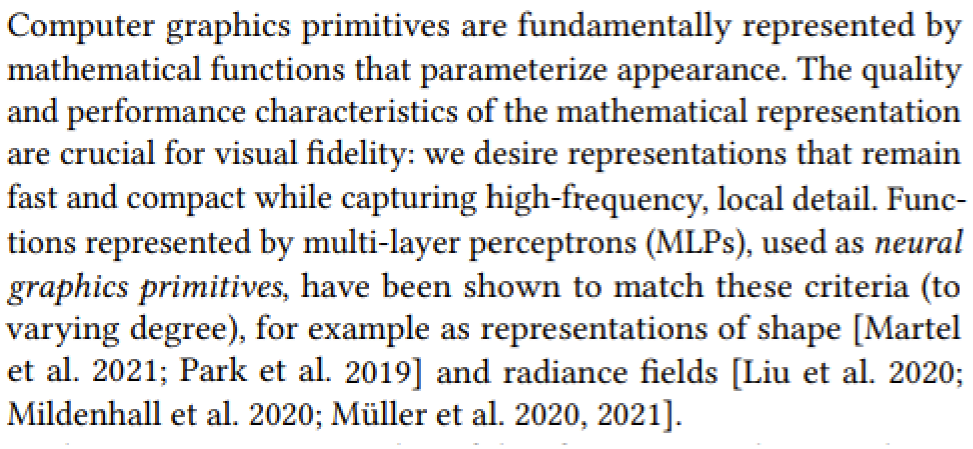


Fig. 1. We demonstrate instant training of neural graphics primitives on a single GPU for multiple tasks. In *Gigapixel image* we represent a gigapixel image by a neural network. *SDF* learns a signed distance function in 3D space whose zero level-set represents a 2D surface. Neural radiance caching (*NRC*) [Müller et al. 2021] employs a neural network that is trained in real-time to cache costly lighting calculations. Lastly, *NeRF* [Mildenhall et al. 2020] uses 2D images and their camera poses to reconstruct a volumetric radiance-and-density field that is visualized using ray marching. In all tasks, our encoding and its efficient implementation provide clear benefits: rapid training, high quality, and simplicity. Our encoding is task-agnostic: we use the same implementation and hyperparameters across all tasks and only vary the hash table size which trades off quality and performance. Photograph ©Trevor Dobson (CC BY-NC-ND 2.0)

Input Encoding

Main Subject: Input Encoding in Neural graphics primitives

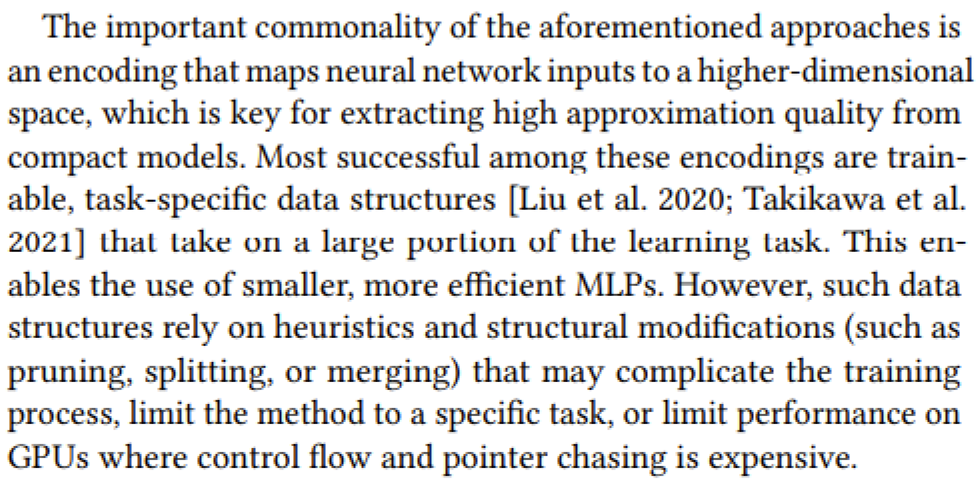


Computer graphics primitives are fundamentally represented by mathematical functions that parameterize appearance. The quality and performance characteristics of the mathematical representation are crucial for visual fidelity: we desire representations that remain fast and compact while capturing high-frequency, local detail. Functions represented by multi-layer perceptrons (MLPs), used as *neural graphics primitives*, have been shown to match these criteria (to varying degree), for example as representations of shape [Martel et al. 2021; Park et al. 2019] and radiance fields [Liu et al. 2020; Mildenhall et al. 2020; Müller et al. 2020, 2021].

Graphics 분야에서 'appearance' 는 수학적 함수로 표현하곤 하는데, 이 수학적 함수를 MLP로 대체하는 움직임들이 많이 보이고 있다.

Input Encoding

Main Subject: Input Encoding in Neural graphics primitives


The important commonality of the aforementioned approaches is an encoding that maps neural network inputs to a higher-dimensional space, which is key for extracting high approximation quality from compact models. Most successful among these encodings are trainable, task-specific data structures [Liu et al. 2020; Takikawa et al. 2021] that take on a large portion of the learning task. This enables the use of smaller, more efficient MLPs. However, such data structures rely on heuristics and structural modifications (such as pruning, splitting, or merging) that may complicate the training process, limit the method to a specific task, or limit performance on GPUs where control flow and pointer chasing is expensive.

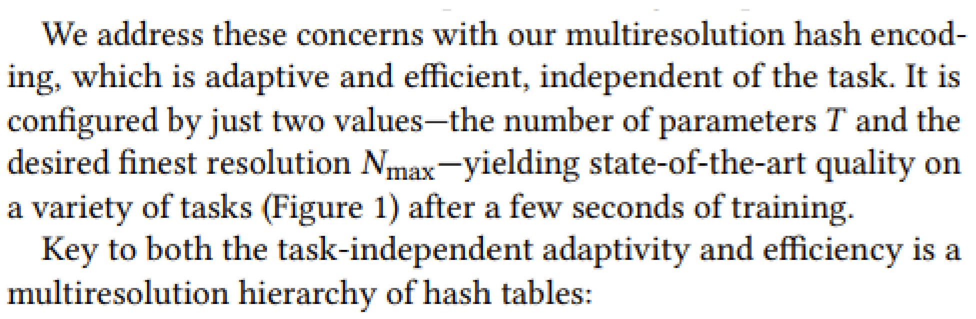
위에서 언급한 MLP 를 사용하는 연구들의 공통점은 MLP에 공급할 input을 higher dimension으로 'encoding' 한다는 점에 있다.

이런 'encoding' 방법론의 성공 사례들은 대개 trainable 한 parameter 들을 각자의 'task에 맞는 자료구조' 에 저장하는데, 이는 MLP의 크기를 더 작고 효율적이게 만들어주었다.

그러나, 'task 에 맞는 자료구조'의 선정은 heuristics에 의존적이며, 해당 자료구조를 보강하는 알고리즘들은 학습을 방해한다.

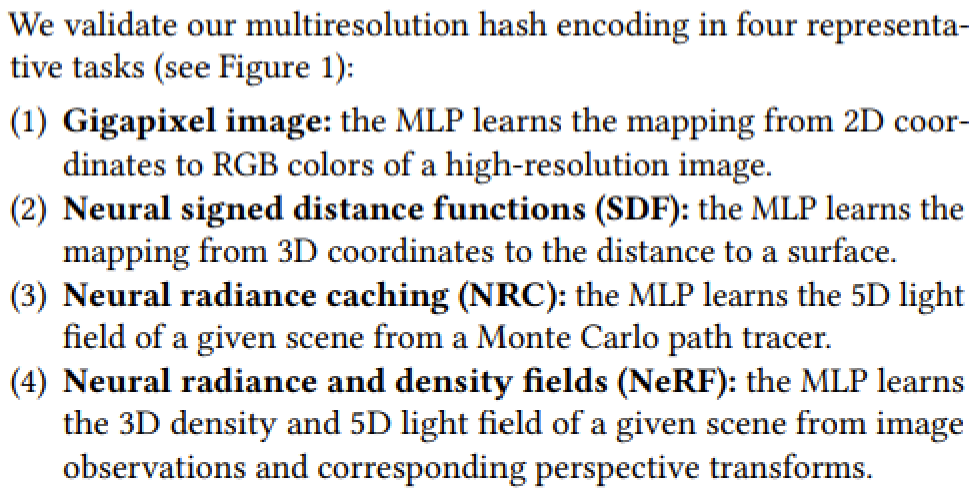
Input Encoding

Main Subject: Input Encoding in Neural graphics primitives


We address these concerns with our multiresolution hash encoding, which is adaptive and efficient, independent of the task. It is configured by just two values—the number of parameters T and the desired finest resolution $N_{\max}$—yielding state-of-the-art quality on a variety of tasks (Figure 1) after a few seconds of training.

Key to both the task-independent adaptivity and efficiency is a multiresolution hierarchy of hash tables:

이런 문제를 해결하기 위해 우리는 'multiresolution hash encoding' 을 제안한다.
이 방식은 효율적이고, 다른 task에도 호환이 가능하다.


We validate our multiresolution hash encoding in four representative tasks (see Figure 1):

(1) **Gigapixel image:** the MLP learns the mapping from 2D coordinates to RGB colors of a high-resolution image.
(2) **Neural signed distance functions (SDF):** the MLP learns the mapping from 3D coordinates to the distance to a surface.
(3) **Neural radiance caching (NRC):** the MLP learns the 5D light field of a given scene from a Monte Carlo path tracer.
(4) **Neural radiance and density fields (NeRF):** the MLP learns the 3D density and 5D light field of a given scene from image observations and corresponding perspective transforms.

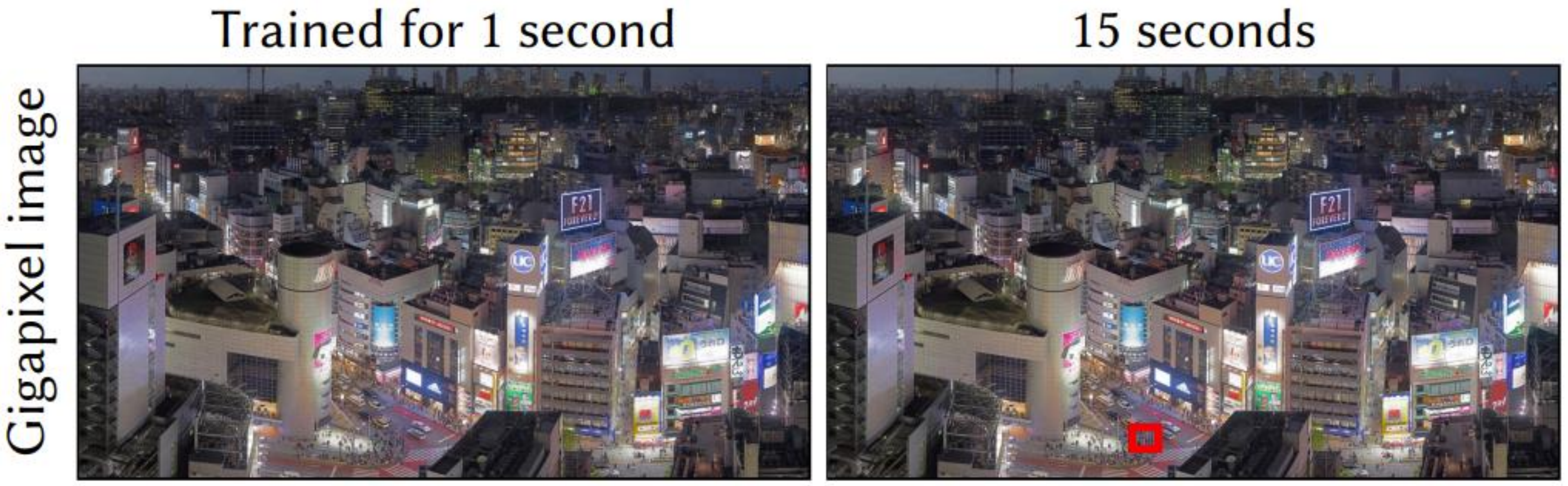
다른 task에도 호환될 수 있음을 증명하기 위해 다음과 같이 4종류의 서로 다른 neural graphics primitives task에 대해 실험하였다.

Paper - Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Main Subject: Input Encoding in Neural graphics primitives

What is Neural graphics primitives?

1. Gigapixel image:
2d coordinate와 RGB color 간의 mapping 관계를 학습하는 태스크

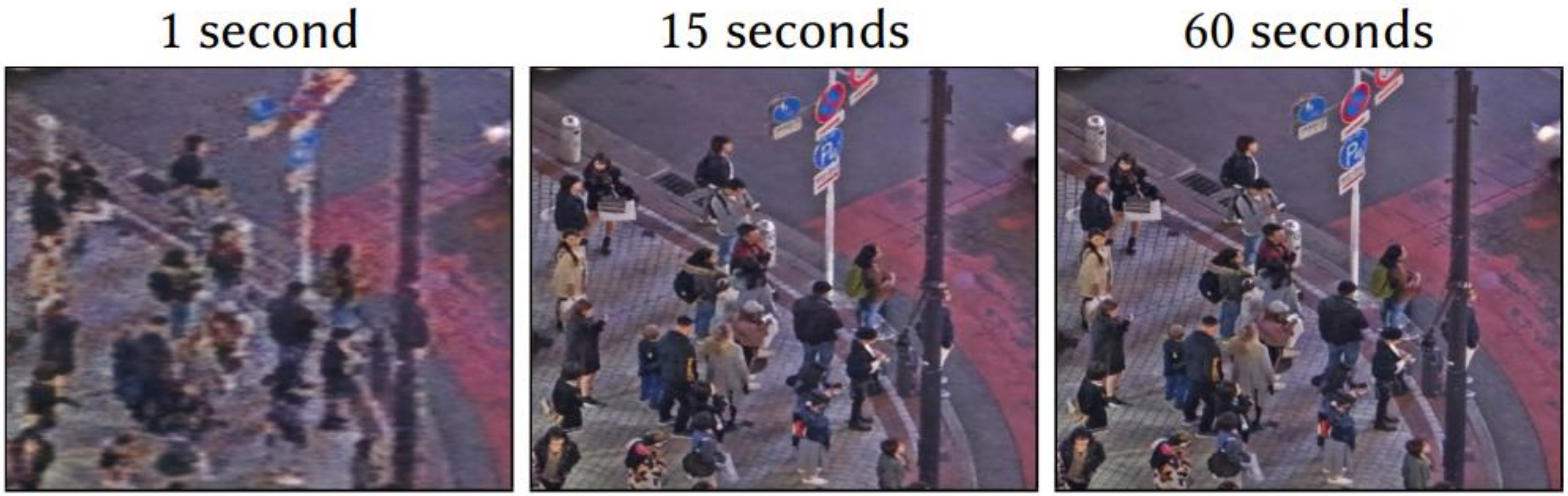


Paper - Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Main Subject: Input Encoding in Neural graphics primitives

What is Neural graphics primitives?

1. Gigapixel image:
2d coordinate와 RGB color 간의 mapping 관계를 학습하는 태스크



Paper - Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Main Subject: Input Encoding in Neural graphics primitives

What is Neural graphics primitives?

2. Neural signed distance functions (SDF):

기존의 SDF: 임의의 Geometry 가 주어질 때, 내부는 +, 외부는 -, Edge에 근접할 수록 0이 되도록 mapping해주는 function

Input Geometry

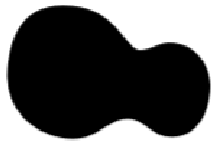


Figure 1.1: A two-dimensional shape.

Output of SDF

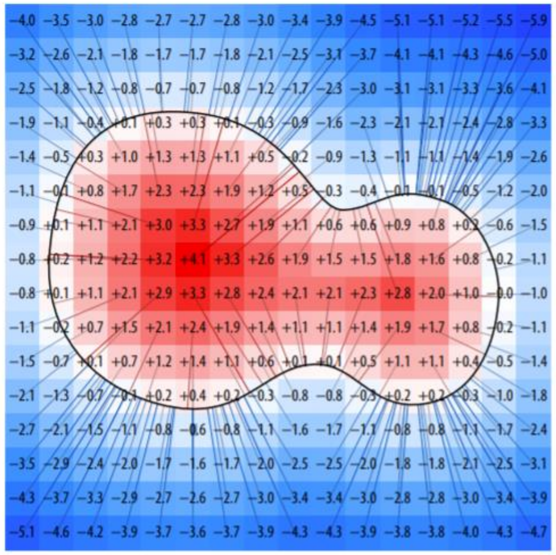


Figure 1.2: The shape's signed distance field.

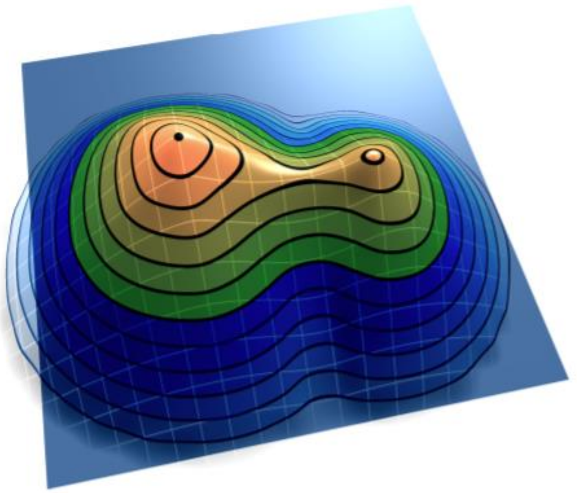


Figure 1.5: A 3D representation of the signed distance field.

https://www.youtube.com/watch?v=-O0-HEZAwg8

Paper - Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

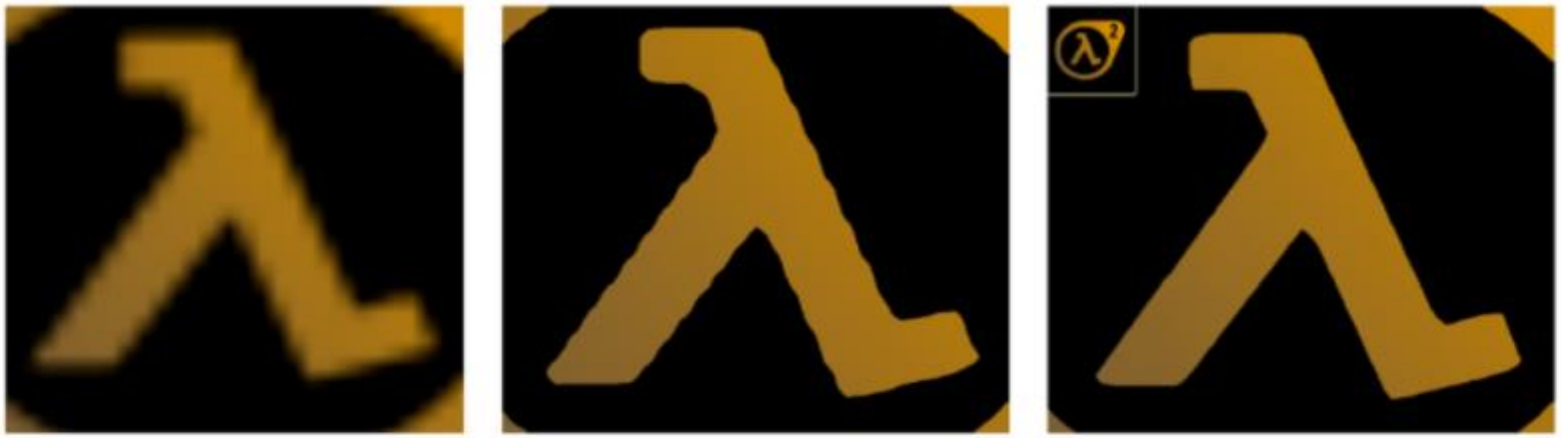
Main Subject: Input Encoding in Neural graphics primitives

What is Neural graphics primitives?

2. Neural signed distance functions (SDF):

기존의 SDF: 임의의 Geometry 가 주어질 때, 내부는 +, 외부는 -, Edge에 근접할 수록 0이 되도록 mapping해주는 function

왜 필요한가?


(a) 64x64 texture, alpha-blended (b) 64x64 texture, alpha tested (c) 64x64 texture using our technique

https://steamcdn-a.akamaihd.net/apps/valve/2007/SIGGRAPH2007_AlphaTestedMagnification.pdf

Paper - Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

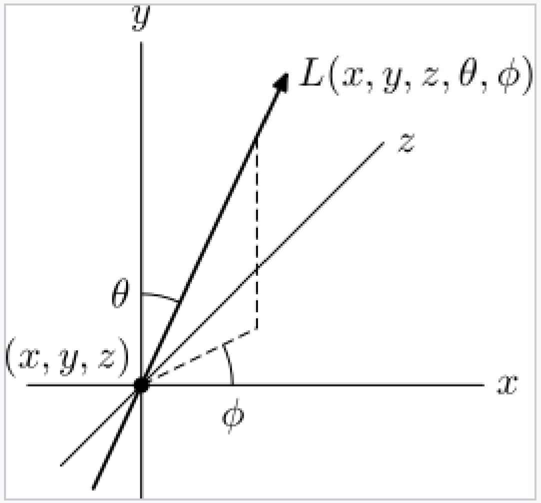
Main Subject: Input Encoding in Neural graphics primitives

What is Neural graphics primitives?

3. Neural radiance caching (NRC):

MLP network 가 이미지안의 pixel들이 주어질 때 radiance field를 approximation 하는 태스크

radiance field? -> 한 voxel의 3d coordinate(x, y, z)와
그 voxel과 view point를 잇는 ray 의 방향(θ,φ) 을 합쳐 5D coordinate로 pixel을 표현한 것.



Parameterizing a ray in 3D space by position (*x*, *y*, *z*) and direction (*θ*, *ϕ*).

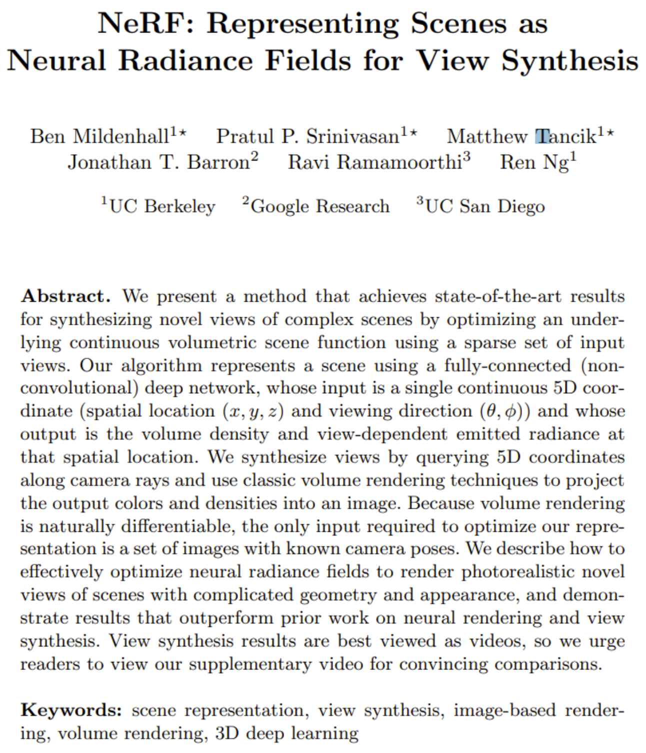
https://en.wikipedia.org/wiki/Light_field

Paper - Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Main Subject: Input Encoding in Neural graphics primitives

What is Neural graphics primitives?

4. Neural radiance and density fields (NeRF):



NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ben Mildenhall[1⋆] Pratul P. Srinivasan[1⋆] Matthew Tancik[1⋆]
Jonathan T. Barron[2] Ravi Ramamoorthi[3] Ren Ng[1]

[1]UC Berkeley [2]Google Research [3]UC San Diego

Abstract. We present a method that achieves state-of-the-art results for synthesizing novel views of complex scenes by optimizing an underlying continuous volumetric scene function using a sparse set of input views. Our algorithm represents a scene using a fully-connected (non-convolutional) deep network, whose input is a single continuous 5D coordinate (spatial location (x, y, z) and viewing direction (θ, ϕ)) and whose output is the volume density and view-dependent emitted radiance at that spatial location. We synthesize views by querying 5D coordinates along camera rays and use classic volume rendering techniques to project the output colors and densities into an image. Because volume rendering is naturally differentiable, the only input required to optimize our representation is a set of images with known camera poses. We describe how to effectively optimize neural radiance fields to render photorealistic novel views of scenes with complicated geometry and appearance, and demonstrate results that outperform prior work on neural rendering and view synthesis. View synthesis results are best viewed as videos, so we urge readers to view our supplementary video for convincing comparisons.

Keywords: scene representation, view synthesis, image-based rendering, volume rendering, 3D deep learning

Main Task: View Synthesis

Paper - Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Main Subject: Input Encoding in Neural graphics primitives

What is Neural graphics primitives?

4. Neural radiance and density fields (NeRF):

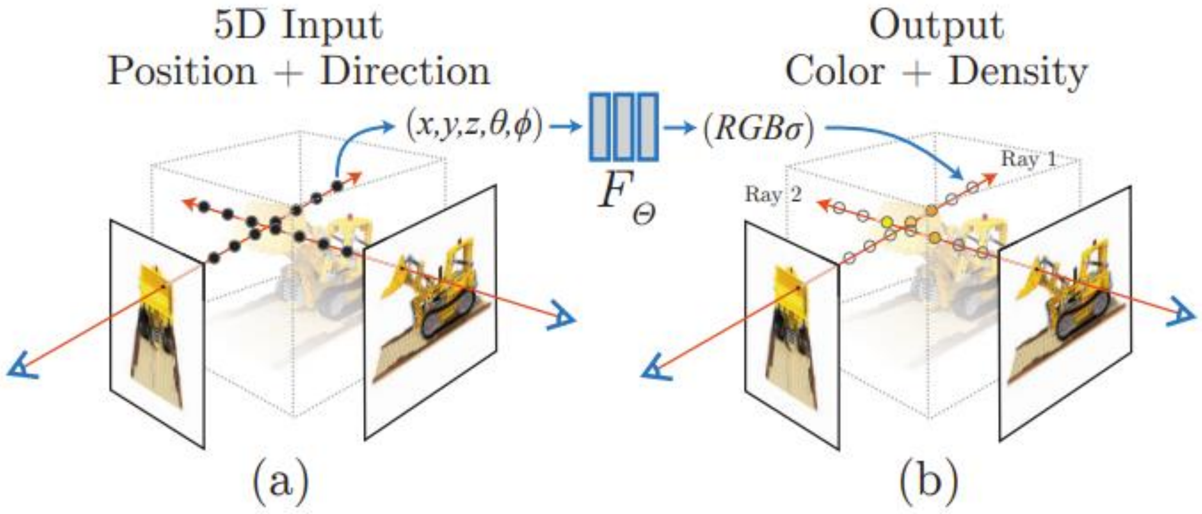
MLP 네트워크에게 3d coordinate와 ray의 direction을 주고 view synthesis 를 배우게 하는 것

1. Base Color 값의 예측
2. P와 P와 같은 ray에 있는 물질들의 Opacity 예측
3. Opacity에 따른 Color값 보정.
    1. opacity가 큰 곳은 pixel값을 크게, 작은 곳은 pixel값을 작게 보정해야 함
    2. P를 가로막는 점들의 opacity 값들 합이 작을수록 pixel 값을 크게, 클수록 pixel 값을 작게 보정해야 함

$$\text{MLP network } F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$$

where

$$x = (x, y, z)$$
$$d = (\theta, \phi)$$
$$c = (R, G, B)$$
$$\sigma = Density$$



Paper - Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Main Subject: Input Encoding in Neural graphics primitives

What is Neural graphics primitives?

4. Neural radiance and density fields (NeRF):

MLP 네트워크에게 3d coordinate와 ray의 direction을 주고 view synthesis 를 배우게 하는 것

이를 모델링 하려면 P의

1. Base Color 값의 예측
2. P와 P와 같은 ray에 있는 물질들의 Opacity 예측
3. Opacity에 따른 Color값 보정.
   1. opacity가 큰 곳은 pixel값을 크게, 작은 곳은 pixel값을 작게 보정해야 함
   2. P를 가로막는 점들의 opacity 값들 합이 작을수록 pixel 값을 크게, 클수록 pixel 값을 작게 보정해야 함

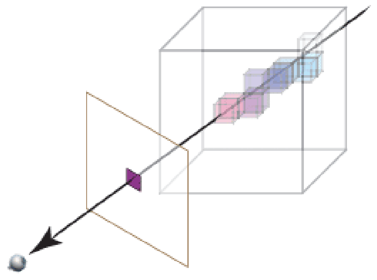
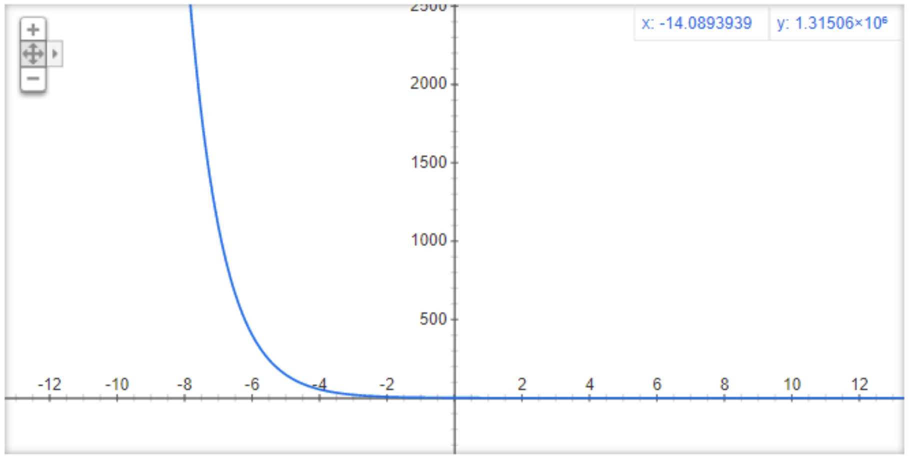


Figure 2: *The ray casting integral sums the color and opacity properties of each data voxel that intersects the ray.*



$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t),\mathbf{d})dt\,,\ \text{where}\ T(t) = \exp\left(-\int_{t_n}^{t}\sigma(\mathbf{r}(s))ds\right)$$

Paper - Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Main Subject: Input Encoding in Neural graphics primitives

What is Neural graphics primitives?

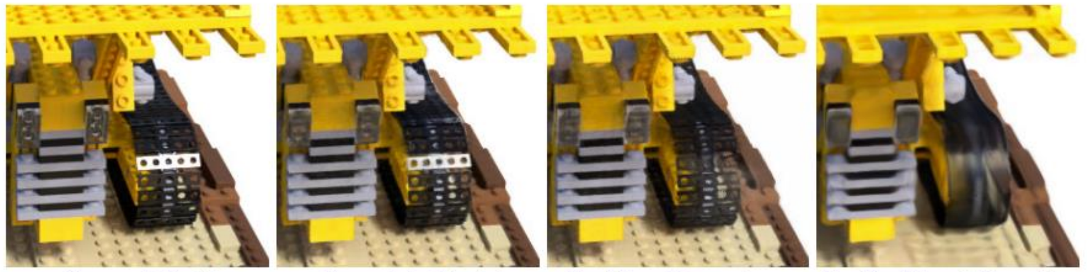
4. Neural radiance and density fields (NeRF):

MLP 네트워크에게 3d coordinate와 ray의 direction을 주고 view synthesis 를 배우게 하는 것

positional Encoding

$$\gamma(p) = \left(\sin(2^0\pi p), \cos(2^0\pi p), \cdots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)\right).$$

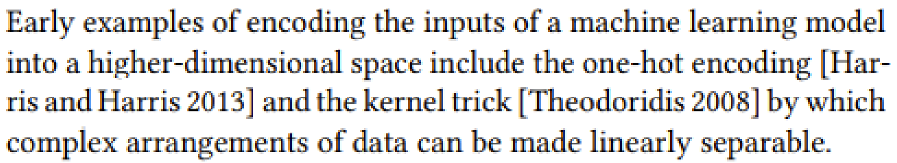
we set $L = 10$ for $\gamma(\mathbf{x})$ and $L = 4$ for $\gamma(\mathbf{d})$.


Ground Truth Complete Model No View Dependence No Positional Encoding

Despite the fact that neural networks are universal function approximators [14], we found that having the network FΘ directly operate on xyzθφ input coordinates results in renderings that perform poorly at representing high-frequency variation in color and geometry. This is consistent with recent work by Rahaman et al. [35], which shows that deep networks are biased towards learning lower frequency functions. They additionally show that mapping the inputs to a higher dimensional space using high frequency functions before passing them to the network enables better fitting of data that contains high frequency variation.

Background & Related Works

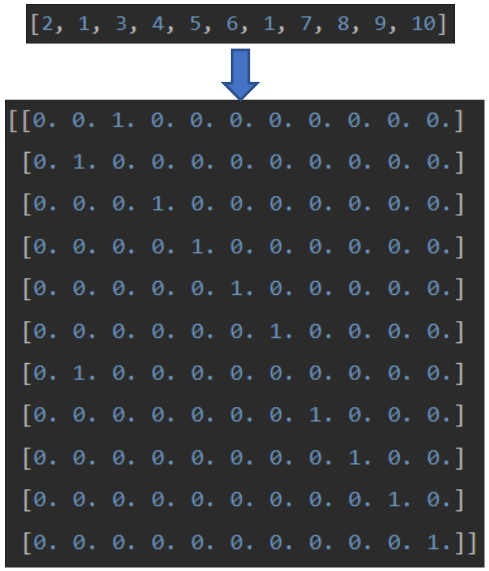
Background


Early examples of encoding the inputs of a machine learning model into a higher-dimensional space include the one-hot encoding [Harris and Harris 2013] and the kernel trick [Theodoridis 2008] by which complex arrangements of data can be made linearly separable.

처음으로 Input Encoding이 시도된 것은 Machine Learning 모델에서 이며, 그 예시로 one-hot encoding, kernel trick 이 있다.

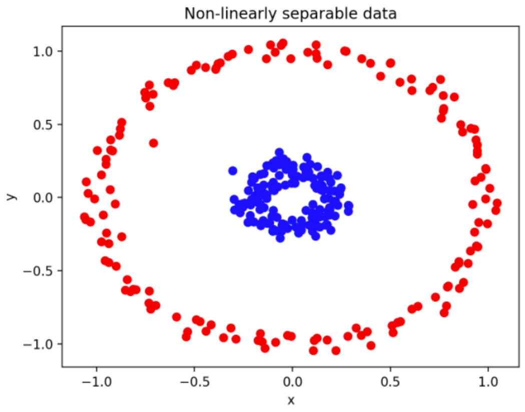
**즉, input encoding은 input data를 더 높은 차원으로 임베딩 해주는 모듈로 이해 할 수 있다.

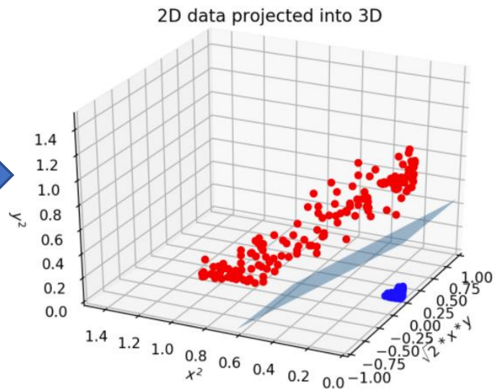
One-hot encoding


```
[2, 1, 3, 4, 5, 6, 1, 7, 8, 9, 10]
```

```
[[0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

Kernel trick



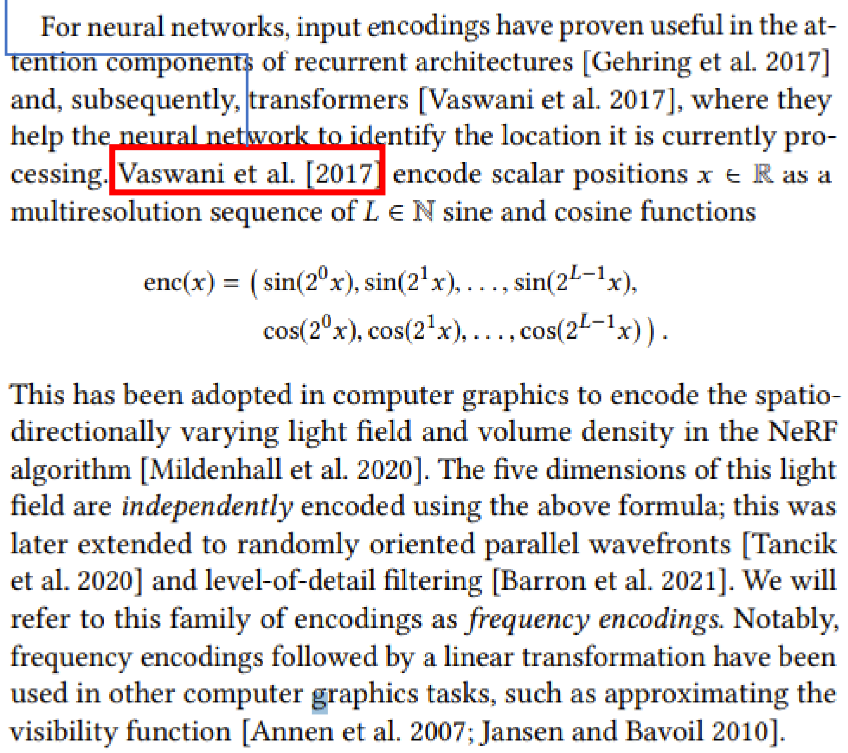


https://hleecaster.com/ml-svm-concept/

Background & Related Works

Related work 1. Frequency encodings

Attention is all you need 논문입니다.



For neural networks, input encodings have proven useful in the attention components of recurrent architectures [Gehring et al. 2017] and, subsequently, transformers [Vaswani et al. 2017], where they help the neural network to identify the location it is currently processing. Vaswani et al. [2017] encode scalar positions $x \in \mathbb{R}$ as a multiresolution sequence of $L \in \mathbb{N}$ sine and cosine functions

$$\mathrm{enc}(x) = \big(\sin(2^0 x), \sin(2^1 x), \ldots, \sin(2^{L-1} x), \cos(2^0 x), \cos(2^1 x), \ldots, \cos(2^{L-1} x) \big) .$$

This has been adopted in computer graphics to encode the spatio-directionally varying light field and volume density in the NeRF algorithm [Mildenhall et al. 2020]. The five dimensions of this light field are *independently* encoded using the above formula; this was later extended to randomly oriented parallel wavefronts [Tancik et al. 2020] and level-of-detail filtering [Barron et al. 2021]. We will refer to this family of encodings as *frequency encodings*. Notably, frequency encodings followed by a linear transformation have been used in other computer graphics tasks, such as approximating the visibility function [Annen et al. 2007; Jansen and Bavoil 2010].

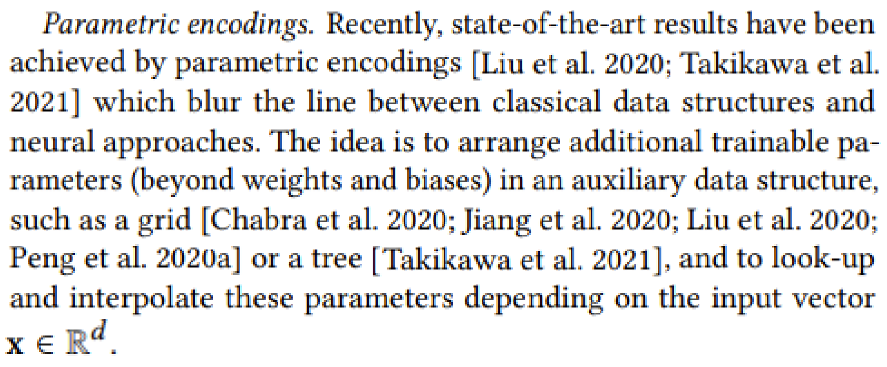
Neural Network에서도 input encoding이 효과적인 사례가 있었음
Ex. Attention에서 사용되었던 positional encoding 방법.

NeRF에서도 Positional Encoding이 사용 되었으며, 이것의 연장선상으로 randomly oriented parallel wavefronts, level-of-detail filtering 과 같은 방법들이 개발되었다.

본문에서는 이들을 통틀어 frequency encoding으로 통칭함.

Background & Related Works

Related work 2. Parametric encodings



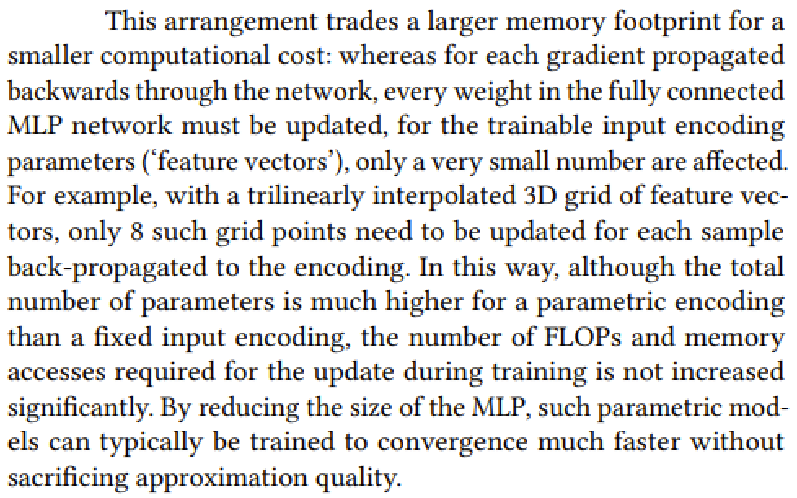
Parametric encodings. Recently, state-of-the-art results have been achieved by parametric encodings [Liu et al. 2020; Takikawa et al. 2021] which blur the line between classical data structures and neural approaches. The idea is to arrange additional trainable parameters (beyond weights and biases) in an auxiliary data structure, such as a grid [Chabra et al. 2020; Jiang et al. 2020; Liu et al. 2020; Peng et al. 2020a] or a tree [Takikawa et al. 2021], and to look-up and interpolate these parameters depending on the input vector $\mathbf{x} \in \mathbb{R}^d$.

Frequency encoding 방법 이후 parametric encoding 방식이 제시되었다.

Grid, Tree와 같은 자료구조 안에 trainable parameter를 추가로 할당하고

학습 중에 input에 대해 look-up하고 interpolate 하는 방식

** 더 높은 차원으로 임베딩 한다는 input encoding의 개념 상 parametric encoding에 사용되는 자료구조들은 꽤 큰 메모리가 필요함을 유추할 수 있다.



This arrangement trades a larger memory footprint for a smaller computational cost: whereas for each gradient propagated backwards through the network, every weight in the fully connected MLP network must be updated, for the trainable input encoding parameters ('feature vectors'), only a very small number are affected. For example, with a trilinearly interpolated 3D grid of feature vectors, only 8 such grid points need to be updated for each sample back-propagated to the encoding. In this way, although the total number of parameters is much higher for a parametric encoding than a fixed input encoding, the number of FLOPs and memory accesses required for the update during training is not increased significantly. By reducing the size of the MLP, such parametric models can typically be trained to convergence much faster without sacrificing approximation quality.

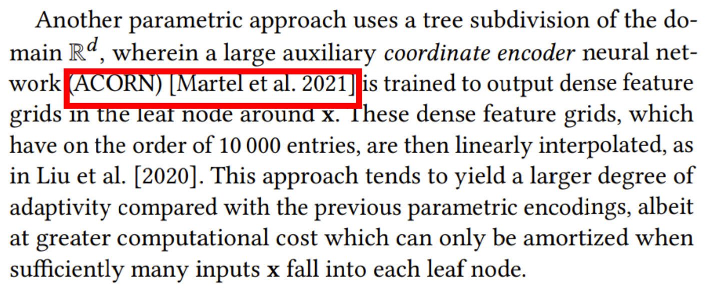
이 방식은 더 작은 연산을 위해 더 큰 메모리 공간을 맞교환한 형태이다.

비록 추가적인 파라미터가 필요하지만, 실제로 한 번의 back-propagation에 영향을 받는 'feature vector'의 수는 적다.

따라서 다른 input encoding 방법론들과 비교했을 때 parameter 수가 훨씬 클 지라도, FLOPs 나 memory access 량은 크게 증가하지 않는다.

Background & Related Works

Related work 2. Parametric encodings

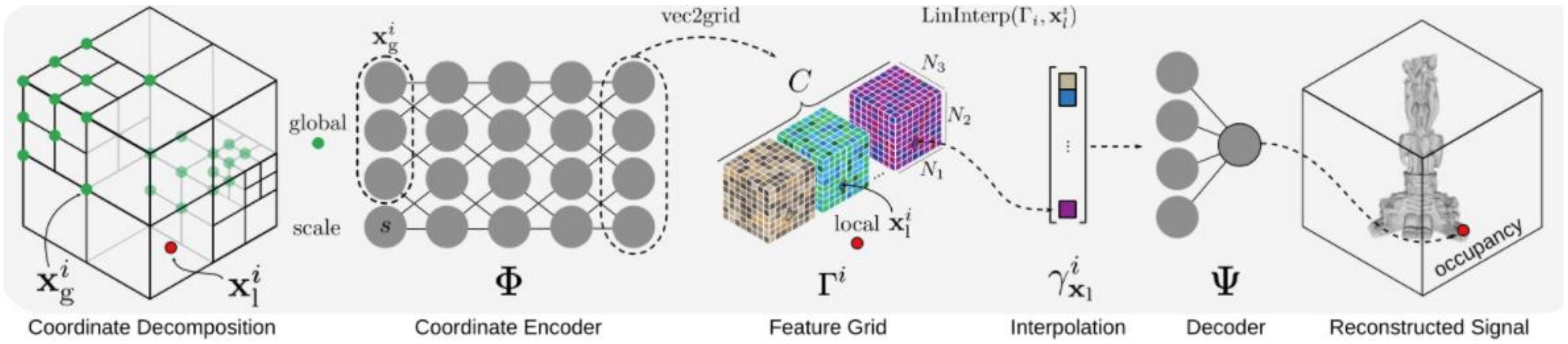


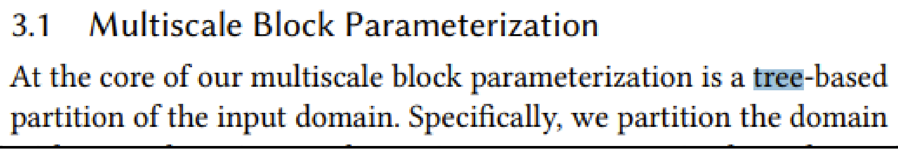
Another parametric approach uses a tree subdivision of the domain $\mathbb{R}^d$, wherein a large auxiliary *coordinate encoder* neural network (ACORN) [Martel et al. 2021] is trained to output dense feature grids in the leaf node around **x**. These dense feature grids, which have on the order of 10 000 entries, are then linearly interpolated, as in Liu et al. [2020]. This approach tends to yield a larger degree of adaptivity compared with the previous parametric encodings, albeit at greater computational cost which can only be amortized when sufficiently many inputs **x** fall into each leaf node.

parametric encoding 방법론 중엔 tree subdivision 을 응용한 사례도 있다.

** Gigapixel Image Task에 도전한 논문으로,
좀 더 세밀한 표현을 얻기 위해 encoder-decoder 구조를 차용하였다.

Input: d차원의 좌표
Output of multiscale block-coordinate decomposition: tree based partition of input
Output of encoder: dense feature Grid





3.1 Multiscale Block Parameterization

At the core of our multiscale block parameterization is a tree-based partition of the input domain. Specifically, we partition the domain

Background & Related Works

Related work 2. Sparse Parametric encodings

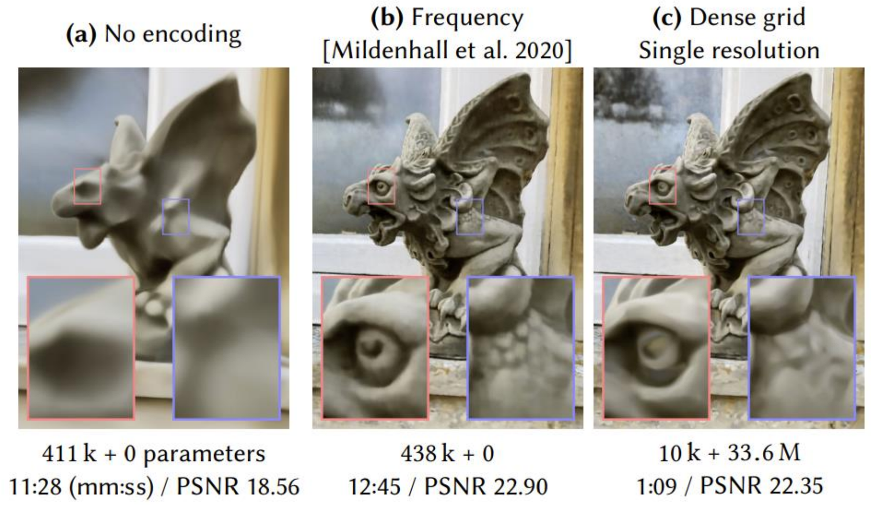
Sparse parametric encodings. While existing parametric encodings tend to yield much greater accuracy than their non-parametric predecessors, they also come with downsides in efficiency and versatility. Dense grids of trainable features consume much more memory than the neural network weights. To illustrate the trade-offs and to motivate our method, Figure 2 shows the effect on reconstruction quality of a neural radiance field for several different encodings. Without any input encoding at all **(a)**, the network is only able to learn a fairly smooth function of position, resulting in a poor approximation of the light field. The frequency encoding **(b)** allows the same moderately sized network (8 hidden layers, each 256 wide) to represent the scene much more accurately. The middle image **(c)** pairs a smaller network with a dense grid of 128^3 trilinearly interpolated feature vectors in $\mathbb{R}^{16}$, for a total of 33.6 million trainable parameters. The large number of trainable parameters can be efficiently updated, as each sample only affects 8 grid points.

앞서 제시된 parametric encoding 방법론은 다른 방법론들과 비교했을 때 더 좋은 성능을 보여주지만 문제는 효율성 과 같은 측면은 포기해야 된다는 점이다.

효율성과 성능 간의 trade-off 관계를 보이기 위해 아래의 figure 를 제시.
(NeRF에 각각의 Encoding 방법론들을 적용 후 비교한 실험.)

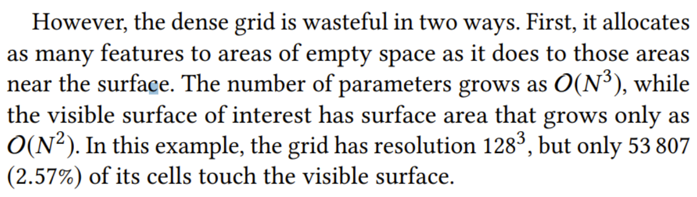
A는 encoding 방법론의 필요성을 증명하고
B는 frequency encoding 방법론의 효과를 보이며
C는 128^3 크기의 Dense grid로 parametric encoding 방법을 실험한 결과이다.

해석해보자면
1. encoding 방법론은 필요하며,
2. Parametric 방법론이 frequency 방법론에 비해 메모리는 크게 쓰지만, 네트워크 자체의 크기는 작고, inference time도 훨씬 더 빠르다.


(a) No encoding
(b) Frequency [Mildenhall et al. 2020]
(c) Dense grid Single resolution
411 k + 0 parameters
11:28 (mm:ss) / PSNR 18.56
438 k + 0
12:45 / PSNR 22.90
10 k + 33.6 M
1:09 / PSNR 22.35

Background & Related Works

Related work 2. Sparse Parametric encodings



However, the dense grid is wasteful in two ways. First, it allocates as many features to areas of empty space as it does to those areas near the surface. The number of parameters grows as $O(N^3)$, while the visible surface of interest has surface area that grows only as $O(N^2)$. In this example, the grid has resolution 128^3, but only 53 807 (2.57%) of its cells touch the visible surface.

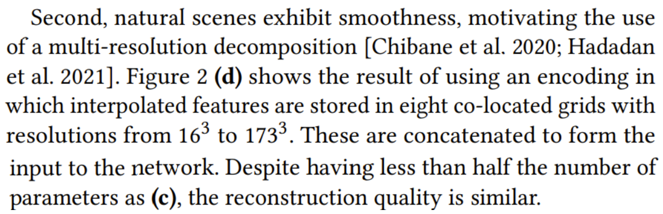
즉 Dense Grid (parametric) 방법론은 기존의 방법론들에 비해 더 좋은 장점을 가진다.

그러나 2가지 측면에서 문제가 있다.

1. Grid의 크기에 따라 파라미터 수가 O(N^3)으로 증가한다.

Background & Related Works

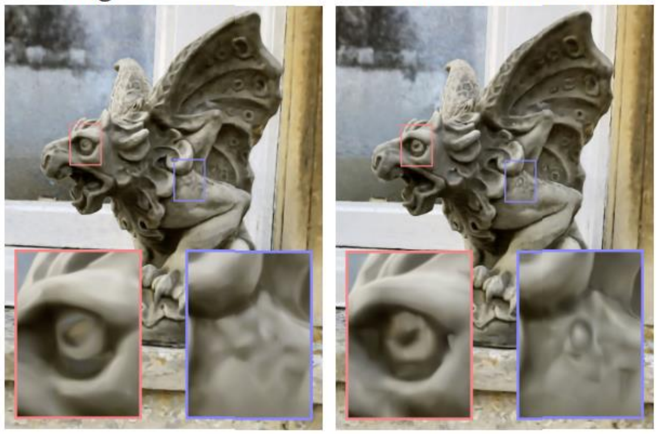
Related work 2. Sparse Parametric encodings



Second, natural scenes exhibit smoothness, motivating the use of a multi-resolution decomposition [Chibane et al. 2020; Hadadan et al. 2021]. Figure 2 **(d)** shows the result of using an encoding in which interpolated features are stored in eight co-located grids with resolutions from 16^3 to 173^3. These are concatenated to form the input to the network. Despite having less than half the number of parameters as **(c)**, the reconstruction quality is similar.

(c) Dense grid
Single resolution

(d) Dense grid
Multi resolution



10 k + 33.6 M
1:09 / PSNR 22.35

10 k + 16.3 M
1:26 / PSNR 23.62

즉 Dense Grid (parametric) 방법론은 기존의 방법론들에 비해 더 좋은 장점을 가진다.

그러나 2가지 측면에서 문제가 있다.

2. Natural scene 들은 digitized 되어 있지 않기 때문에 유려한 선이 필요하다.
이를 구현하기 위해 multi-resolution decomposition들이 많이 사용된다.
Figure2. d는 Dense Grid를 multi-resolution 방법으로 실험한 결과이다.
** 해석
Dense Grid는 multi-resolution 방법으로 사용되면 현재의 파라미터 수 이상으로 쓸 수 없었다.
그럼에도 불구하고 Single resolution 으로 사용되었을 때와의 성능차가 크게 나지 않았다.

Background & Related Works

Related work 2. Sparse Parametric encodings

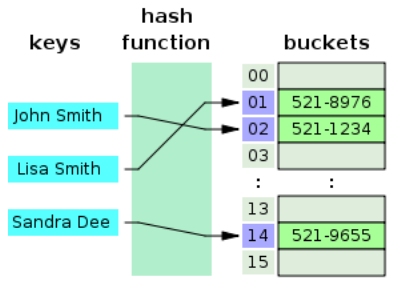
If the surface of interest is known a-priori, a data structure such as an octree [Takikawa et al. 2021] or sparse grid [Chabra et al. 2020; Chibane et al. 2020; Hadadan et al. 2021; Jiang et al. 2020; Liu et al. 2020; Peng et al. 2020a] can be used to cull away the unused features in the dense grid. However, in the NeRF setting, surfaces only emerge during training. NSVF [Liu et al. 2020] and several concurrent works [Sun et al. 2021; Yu et al. 2021a] adopt a multi-stage, coarse to fine strategy in which regions of the feature grid are progressively refined and culled away as necessary. While effective, this leads to a more complex training process in which the sparse data structure must be periodically updated.

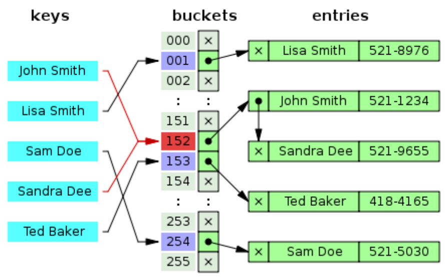
만약 surface에 대한 정보를 미리 알고 있다면, 공백이 들어있는 부분의 parameter들을 grid 안에서 지우면서 학습이 가능하다. (octree, sparse grid)
그러나 NeRF는 학습시에 surface 정보를 활용하는 컴포넌트가 없다.
그래서 NSVF와 같이 multi-stage를 구성, coarse-to-fine 전략을 사용하여 feature grid를 progressive하게 refine 하는 방법론이 제시되었다.

Our method—Figure 2 **(e,f)**—combines both ideas to reduce waste. We store the trainable feature vectors in a compact hash table, whose size is a hyper-parameter T which can be tuned to trade the number of parameters for reconstruction quality. It does not rely on any spatial data structure, progressive pruning during training, or a-priori knowledge of the geometry of the encoded scene. Analogous to the multi-resolution grid example in **(d)**, we use multiple separate hash tables indexed at different resolutions, whose interpolated outputs are concatenated before being passed through the MLP. The reconstruction quality is comparable to the dense grid encoding, despite having 20× fewer parameters.

본 논문에서는 앞에서 언급한 두 가지의 insight를 활용하여 낭비를 줄일 수 있는 parametric encoding 방법을 제시한다.

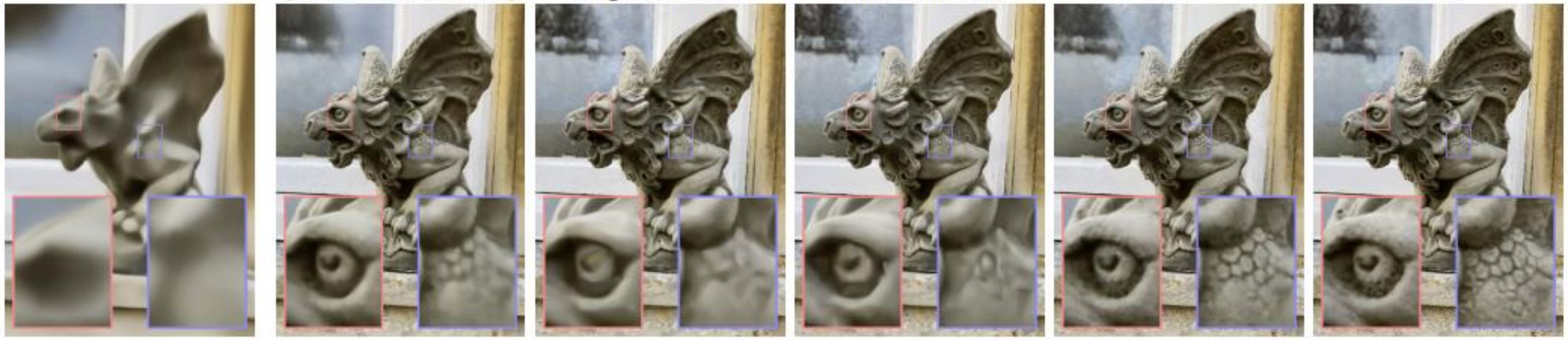
Trainable parameter 들을 고정된 T 크기의 hash table들 안에 저장하는 방법으로
이 방법은 surface에 대한 사전 정보를 필요로 하지 않으며,
pruning과 같은 자료구조의 보강을 위한 추가 알고리즘도 필요로 하지 않는다.





Methods

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding • 36:3


(a) No encoding
(b) Frequency [Mildenhall et al. 2020]
(c) Dense grid Single resolution
(d) Dense grid Multi resolution
(e) Hash table (ours) $T = 2^{14}$
(f) Hash table (ours) $T = 2^{19}$
411 k + 0 parameters
11:28 (mm:ss) / PSNR 18.56
438 k + 0
12:45 / PSNR 22.90
10 k + 33.6 M
1:09 / PSNR 22.35
10 k + 16.3 M
1:26 / PSNR 23.62
10 k + 494 k
1:48 / PSNR 22.61
10 k + 12.6 M
1:47 / PSNR 24.58

Methods – Multi-resolution Hash Encoding

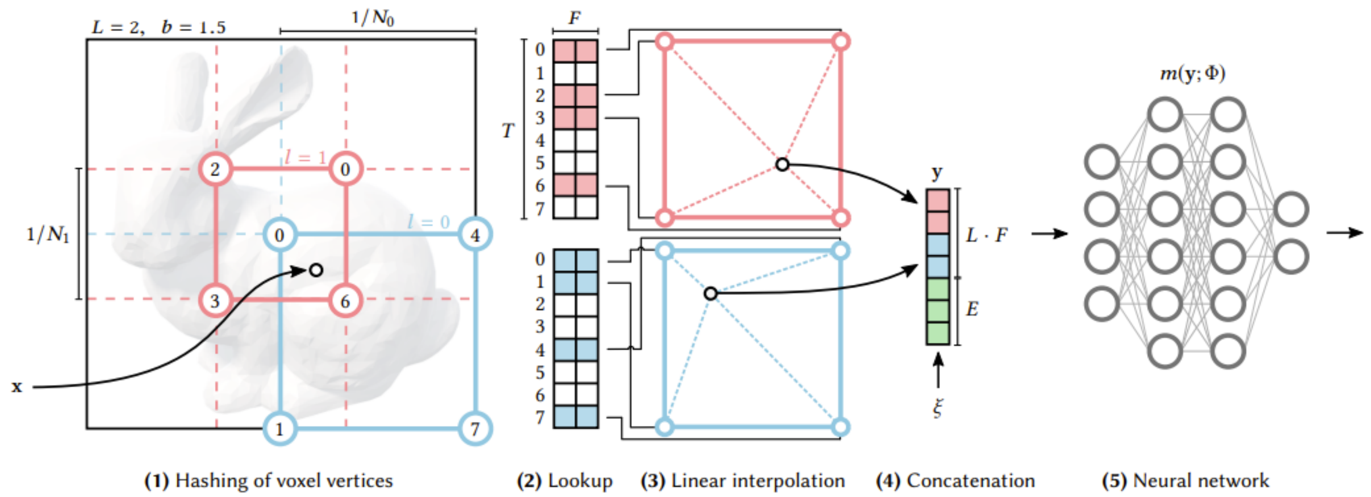
Table 1. Hash encoding parameters and their typical values. Only the hash table size T and max. resolution $N_{\max}$ need to be tuned to the use case.

Parameter	Symbol	Value
Number of levels	L	16
Max. entries per level (hash table size)	T	$[2^{14}, 2^{24}]$
Number of feature dimensions per entry	F	2
Coarsest resolution	$N_{\min}$	16
Finest resolution	$N_{\max}$	[512, 524288]

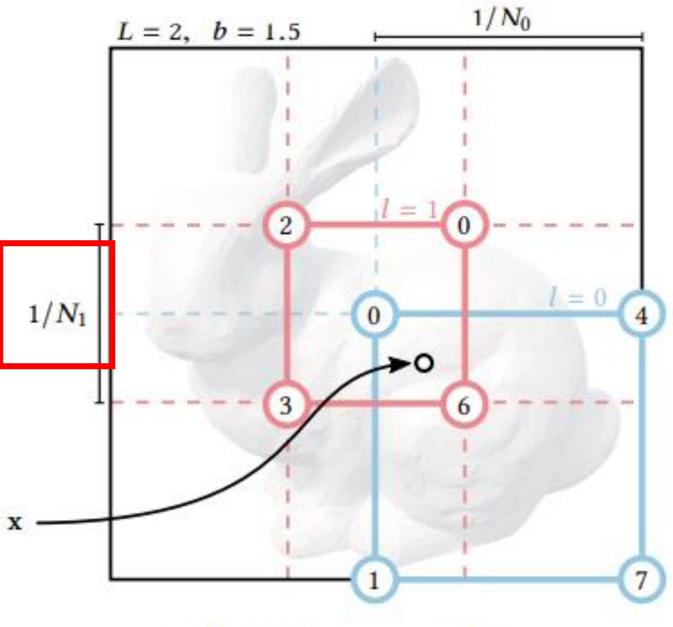
$m(\mathbf{y};\Phi)$:NeRF

$\mathbf{y} = \mathrm{enc}(\mathbf{x};\theta)$:NeRF에 넘겨줄 정보. Enc는 multi-resolution hash encoding

Θ = L(하나의 점에 대해 관찰하는 level 갯수)*T(hash table 길이)*F(hash table의 레코드 길이)



Methods – Multi-resolution Hash Encoding



(1) Hashing of voxel vertices

Table 1. Hash encoding parameters and their typical values. Only the hash table size T and max. resolution $N_{\max}$ need to be tuned to the use case.

Parameter	Symbol	Value
Number of levels	L	16
Max. entries per level (hash table size)	T	$[2^{14}, 2^{24}]$
Number of feature dimensions per entry	F	2
Coarsest resolution	$N_{\min}$	16
Finest resolution	$N_{\max}$	$[512, 524288]$

$$N_l := \left\lfloor N_{\min} \cdot b^l \right\rfloor,$$

$$b := \exp\left(\frac{\ln N_{\max} - \ln N_{\min}}{L-1}\right)$$

use cases have $b \in [1.38, 2]$.

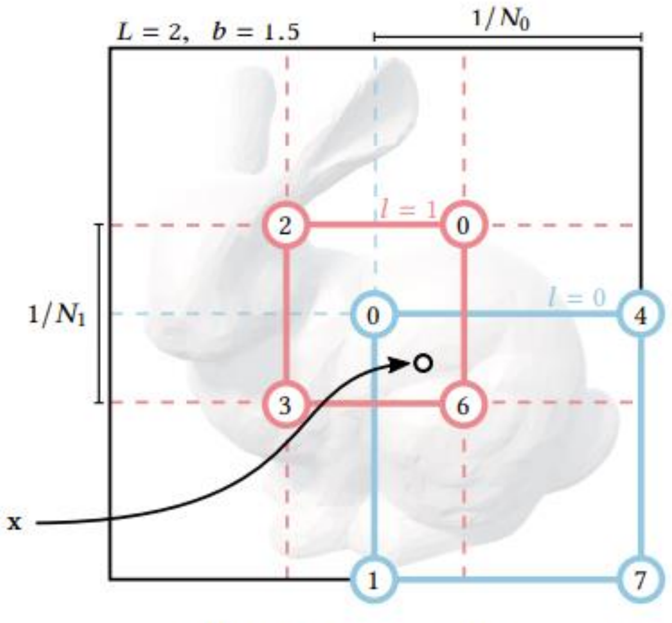
$$\lceil \mathbf{x}_l \rceil := \lceil \mathbf{x} \cdot N_l \rceil.$$

$$\lfloor \mathbf{x}_l \rfloor := \lfloor \mathbf{x} \cdot N_l \rfloor,$$

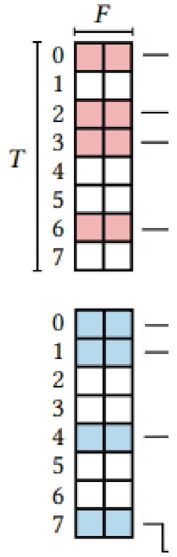
x: Input coordinate,
Level: N과 b로 계산된 특정 크기의 voxel
l: index of level
L: max index of level + 1 (0-indexed)

1. X를 둘러싸는 voxel의 모서리 위에 있는 point들 추출. (주어진 level 크기 N_l 에 대해 반올림/반내림을 하여 얻은 두 값의 사이에 있는 point를 추출)
   즉, COARSE 한 level (N이 작은) 에서는 샘플링 되는 point들이 적을 것이고,
   FINE 한 level (N이 큰) 에서는 샘플링 되는 point들이 많을 것이다.

Methods – Multi-resolution Hash Encoding



(1) Hashing of voxel vertices



(2) Lookup

$$h(\mathbf{x}) = \left(\bigoplus_{i=1}^{d} x_i \pi_i \right) \mod T$$

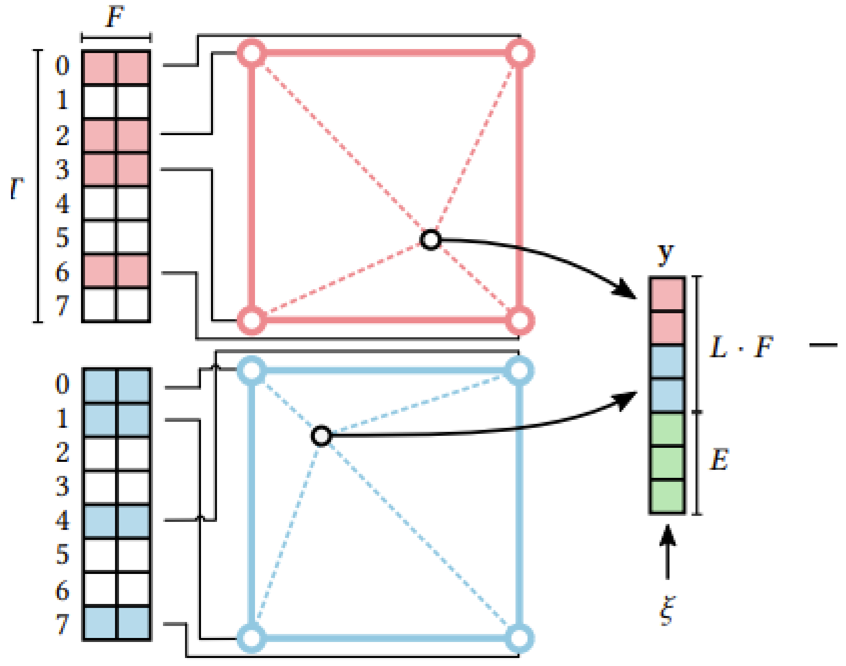
$\oplus$ denotes the bit-wise XOR operation
π_i are unique

x: Input coordinate,
Level: N과 b로 계산된 특정 크기의 voxel
l: index of level
L: max index of level + 1 (0-indexed)

2. 이때, coarse 한 level에서는 샘플된 point들이 T개보다 적거나 같기에 바로 hash table의 인덱스들과 1:1 매칭을 해주고,
Fine 한 부분에서는 상대적으로 샘플링 되는 corner 들이 많을 것이기 때문에 Hash function h(x) 로부터 샘플링한 point 들의 index 들과 매칭을 얻는다.

T 보다 크면 collision 이 나지 않나요???
-> Gradient based optimization 을 통해, 쓸데없는 feature parameter를 제거할 수 있다! (MLP가 학습되면서 필요 없는 부분을 날려줄 것이다.)

Methods – Multi-resolution Hash Encoding



(2) Lookup (3) Linear interpolation (4) Concatenation

3. Lastly, the feature vectors at each corner are d-linearly interpolated according to the relative position of $\mathbf{x}$ within its hypercube, i.e. the interpolation weight is $\mathbf{w}_l := \mathbf{x}_l - \lfloor \mathbf{x}_l \rfloor$.

i.e. the interpolation weight is $\mathbf{w}_l := \mathbf{x}_l - \lfloor \mathbf{x}_l \rfloor$.

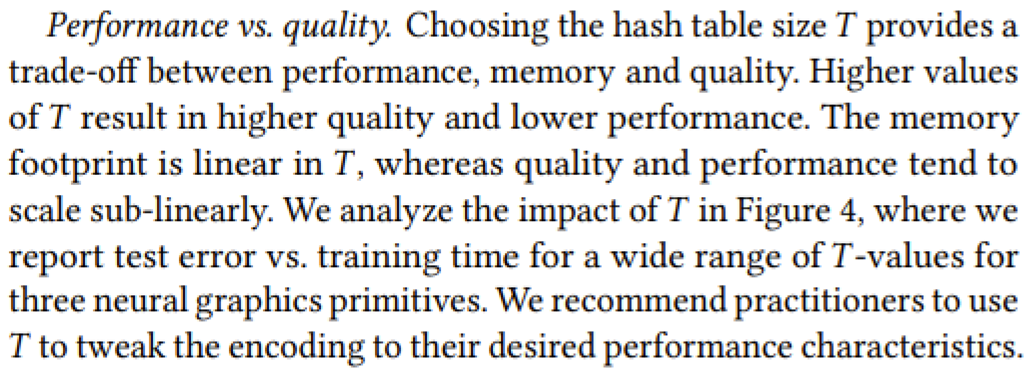
4. Recall that this process takes place independently for each of the L levels. The interpolated feature vectors of each level, as well as auxiliary inputs $\xi \in \mathbb{R}^E$ (such as the encoded view direction and textures in neural radiance caching), are concatenated to produce $\mathbf{y} \in \mathbb{R}^{LF+E}$, which is the encoded input enc($\mathbf{x}; \theta$) to the MLP $m(\mathbf{y}; \Phi)$.

x: Input coordinate,
Level: N과 b로 계산된 특정 크기의 voxel
l: index of level
L: max index of level + 1 (0-indexed)

3. point와 매칭된 index 안의 값을 참조하여, 각 point에서 x의 위치로의 방향으로 linear interpolation 시켜준다.
4. 각 레벨에서 interpolation 하여 얻은 feature vector들을 concat하되, viewing direction 과 같은 추가 정보들을 같이 concat 해준다.

Methods – Multi-resolution Hash Encoding

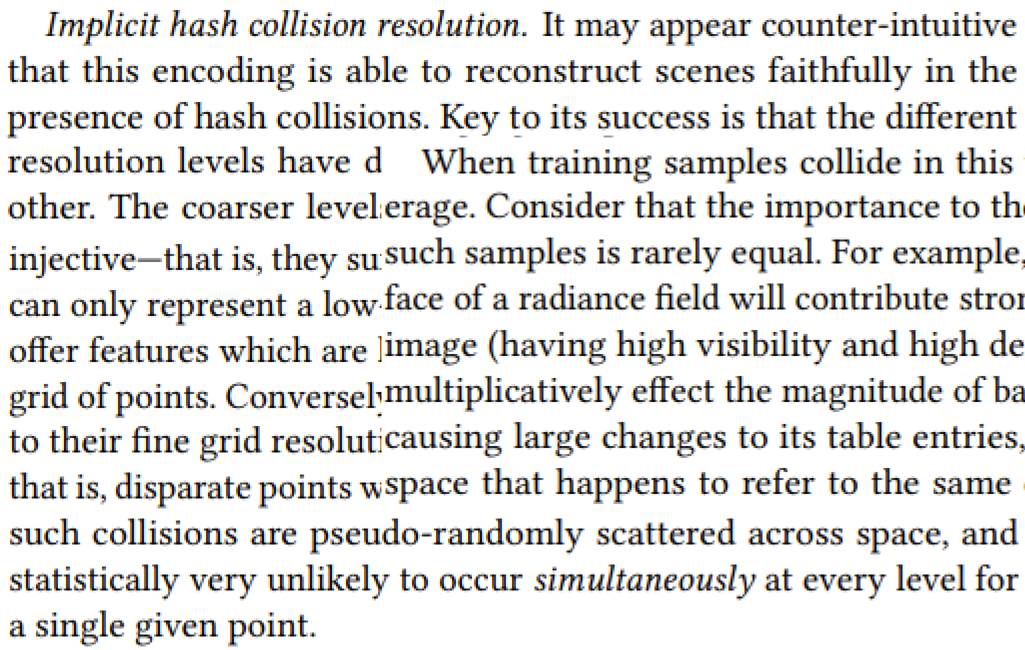
1. performance v.s. quality



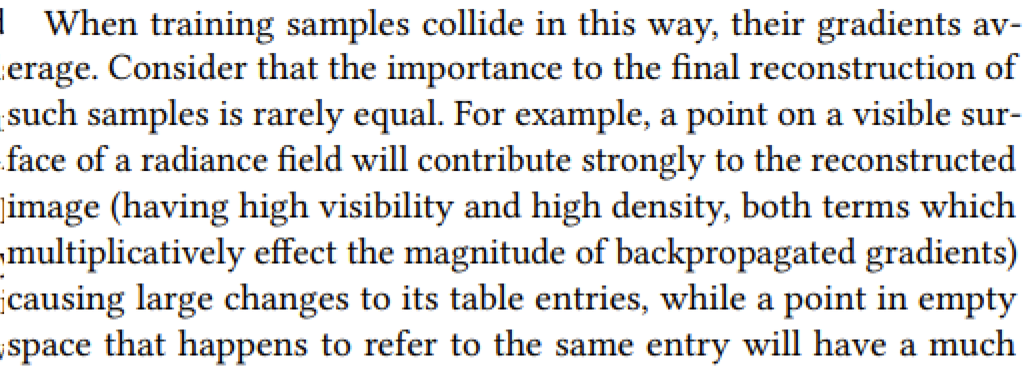
Performance vs. quality. Choosing the hash table size T provides a trade-off between performance, memory and quality. Higher values of T result in higher quality and lower performance. The memory footprint is linear in T, whereas quality and performance tend to scale sub-linearly. We analyze the impact of T in Figure 4, where we report test error vs. training time for a wide range of T-values for three neural graphics primitives. We recommend practitioners to use T to tweak the encoding to their desired performance characteristics.

Methods – Multi-resolution Hash Encoding

2. Implicit hash collision resolution



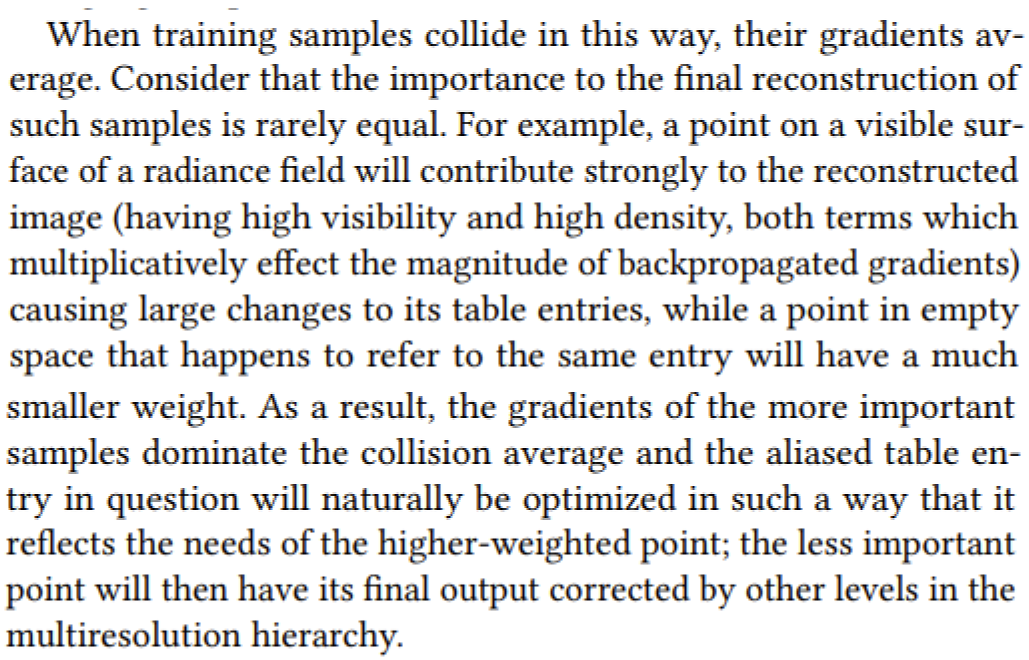
Implicit hash collision resolution. It may appear counter-intuitive
that this encoding is able to reconstruct scenes faithfully in the
presence of hash collisions. Key to its success is that the different
resolution levels have d
other. The coarser level
injective—that is, they su
can only represent a low
offer features which are l
grid of points. Conversel
to their fine grid resoluti
that is, disparate points w
such collisions are pseudo-randomly scattered across space, and
statistically very unlikely to occur *simultaneously* at every level for
a single given point.



When training samples collide in this way, their gradients average. Consider that the importance to the final reconstruction of such samples is rarely equal. For example, a point on a visible surface of a radiance field will contribute strongly to the reconstructed image (having high visibility and high density, both terms which multiplicatively effect the magnitude of backpropagated gradients) causing large changes to its table entries, while a point in empty space that happens to refer to the same entry will have a much

Methods – Multi-resolution Hash Encoding

2. Implicit hash collision resolution



When training samples collide in this way, their gradients average. Consider that the importance to the final reconstruction of such samples is rarely equal. For example, a point on a visible surface of a radiance field will contribute strongly to the reconstructed image (having high visibility and high density, both terms which multiplicatively effect the magnitude of backpropagated gradients) causing large changes to its table entries, while a point in empty space that happens to refer to the same entry will have a much smaller weight. As a result, the gradients of the more important samples dominate the collision average and the aliased table entry in question will naturally be optimized in such a way that it reflects the needs of the higher-weighted point; the less important point will then have its final output corrected by other levels in the multiresolution hierarchy.

Results

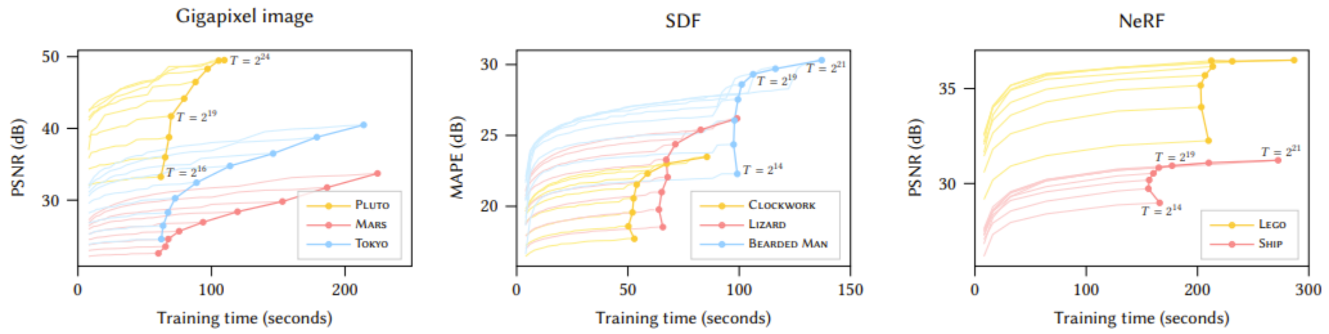


Fig. 4. The main curves plot test error over training time for varying hash table size T which determines the number of trainable encoding parameters. Increasing T improves reconstruction, at the cost of higher memory usage and slower training and inference. A performance cliff is visible at $T > 2^{19}$ where the cache of our RTX 3090 GPU becomes oversubscribed (particularly visible for SDF and NeRF). The plot also shows model convergence over time leading up to the final state. This highlights how high quality results are already obtained after only a few seconds. Jumps in the convergence (most visible towards the end of SDF training) are caused by learning rate decay. For NeRF and Gigapixel image, training finishes after 31 000 steps and for SDF after 11 000 steps.

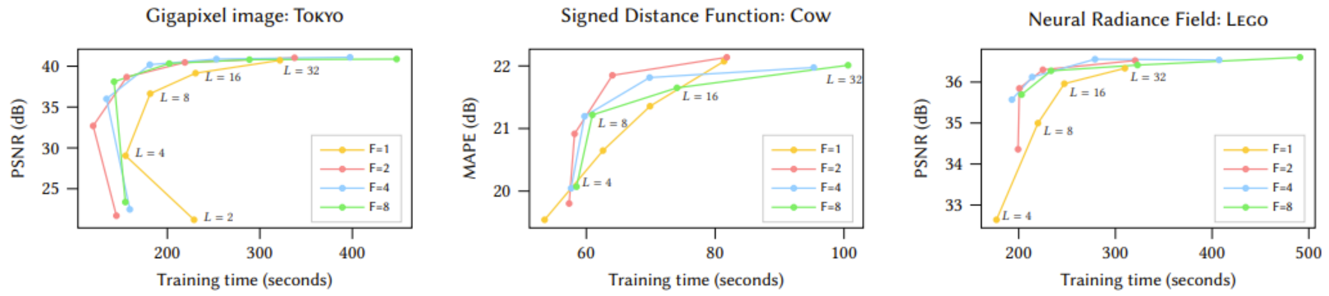


Fig. 5. Test error over training time for fixed values of feature dimensionality F as the number of hash table levels L is varied. To maintain a roughly equal trainable parameter count, the hash table size T is set according to $F \cdot T \cdot L = 2^{24}$ for SDF and NeRF, whereas gigapixel image uses 2^{28}. Since $(F = 2, L = 16)$ is near the best-case performance and quality (top-left) for all applications, we use this configuration in all results. $F = 1$ is slow on our RTX 3090 GPU since atomic half-precision accumulation is only efficient for 2D vectors but not for scalars. For NeRF and Gigapixel image, training finishes after 31 000 steps whereas SDF completes at 11 000 steps.

Results

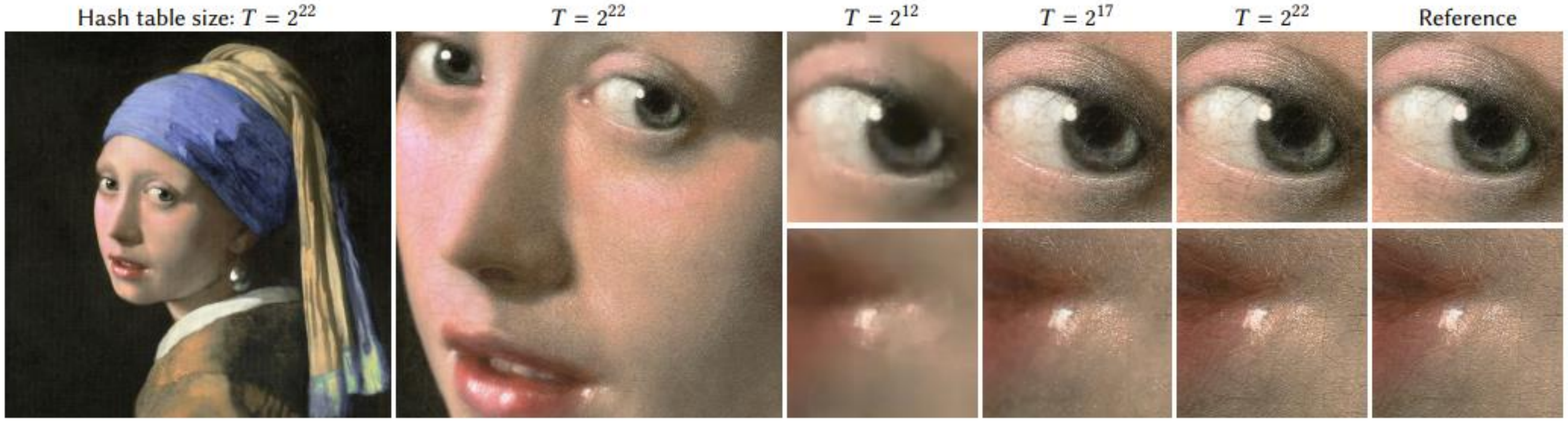
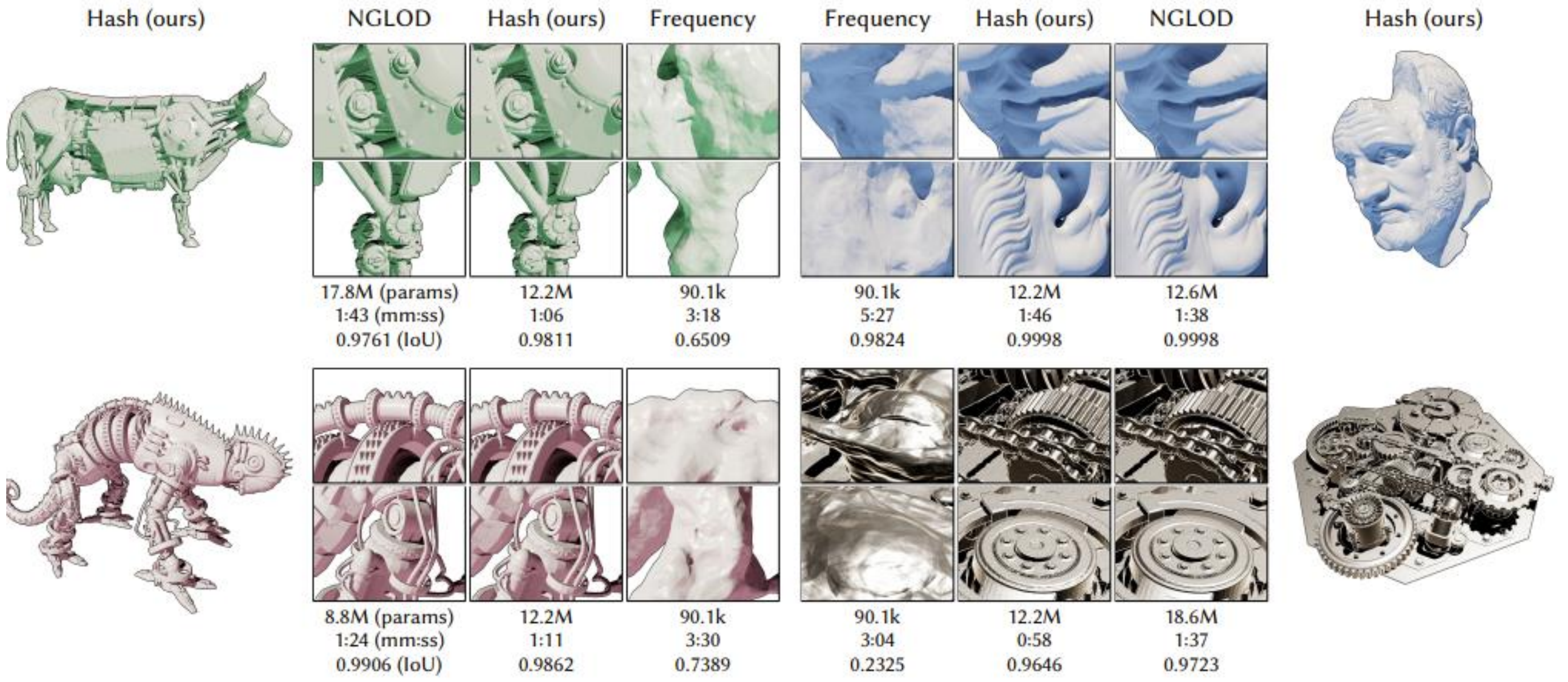


Fig. 6. Approximating an RGB image of resolution 20,000 × 23,466 (469M RGB pixels) with our multiresolution hash encoding. With hash table sizes T of 2^{12}, 2^{17}, and 2^{22} the models shown have 117 k, 2.7 M, and 47.5 M trainable parameters respectively. With only 3.4% of the degrees of freedom of the input, the last model achieves a reconstruction PSNR of 29.8 dB. "Girl With a Pearl Earing" renovation ©Koorosh Orooj (CC BY-SA 4.0)

Results



https://nvlabs.github.io/instant-ngp/

Thank You