Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Schneidt, Lukas Michael

Department of Informatics - Technische Universität München

Abstract

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller presented their paper "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding" [4] at SIGGRAPH 2022, where it won the best paper award. It establishes a hash encoding on multiple resolutions that offers near-instant training of several neural graphics primitives. They evaluate it on 3D signed distance functions (SDF), gigapixel images, neural radiance caching, and neural radiance and density fields (NeRF). Its efficient implementation provides almost instant training and high quality on all evaluated primitives. The exact implementation and hyperparameters can be used for all experiments, making it task-agnostic, while the hash table size is used for trading off quality and training time.

1 Introduction

Simple primitives in computer graphics describe elements like an arc, a square, or a cone, from which more complicated objects or images can be built. They are defined by a mathematical function that fully describes their appearance, given a set of parameters. This concept can be extended to more complex primitives.
The authors of the presented paper came up with an encoding of the input of so-called neural graphics primitives, which are graphics primitives that are described by a neural network.
This parameterization is evaluated on the following four neural graphics primitives:

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

Lately, parametric encodings of inputs for neural networks have achieved state-of-the-art results. These parametric encodings arrange additional parameters (e.g., positional information, etc.) in a data structure like a grid or a tree in order to

be able to look up and interpolate them. Although these encodings consume more memory, the computational cost can be reduced, and they tend to yield better accuracy than non-parametric encodings.
Figure 1 illustrates the trade-offs of these parametric encodings and motivates the author's approach. The dense structures in parametric encodings use the same number of features for empty areas (less important) as it uses for areas around the surface (more important). Furthermore, as natural scenes present smoothness, a multi-resolution decomposition seems rational. The multiresolution hash encoding presented in the paper tackles both problems.

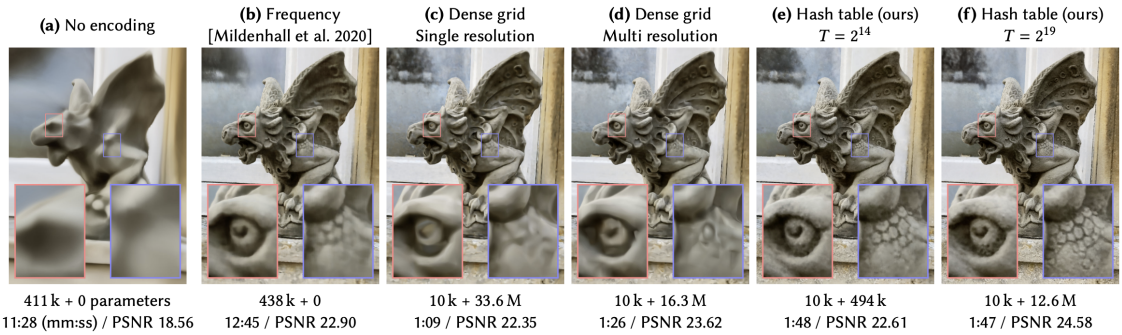

Figure 1: Reconstruction quality using different encodings and parametric data structures for storing trainable feature embeddings

2 Method description

Additional to the ordinary trainable parameters ϕ of a neural network, this approach comes with trainable encoding parameters θ, which are arranged into L resolution levels. Each level L contains up to T feature vectors, with each vector having dimension F.
The resolution of each level L is chosen to be a geometric progression between the coarsest and finest resolutions $[N_{min}, N_{max}]$, with N_{max} matching the finest detail in the training data:

$$b := exp(\frac{lnN_{max} - lnN_{min}}{L - 1}) \tag{1}$$

$$N_l := N_{min} * b_l \tag{2}$$

Let's assume that we want to apply the multiresolution hash encoding in 2D. As a first step, we divide the input into voxels of size N_l as described in Formula 1, with $L = 2$. Once that's done, we look for the voxel that contains input x. Each corner of the voxel corresponds to a feature vector in the hash table.
We look up the corresponding F-dimensional feature vector of each corner and linearly interpolate them corresponding to the relative position of input x.
These steps are continued for all resolution levels L, and each of the interpolated vectors is concatenated to a vector of size $L * F$. Additionally, an auxiliary input $\xi \in \mathbb{R}^E$ is concatenated at the end to yield the encoded vector $y \in \mathbb{R}^{LF+E}$. This auxiliary input can be something like the view direction and material parameters when learning a light field.

This vector is used as input for a small neural network. In order to train the encoding, the loss gradients are backpropagated through the neural network, the concatenation, the interpolation, and in the end, accumulated in the looked-up feature vectors. The whole encoding process is illustrated in Figure 2.

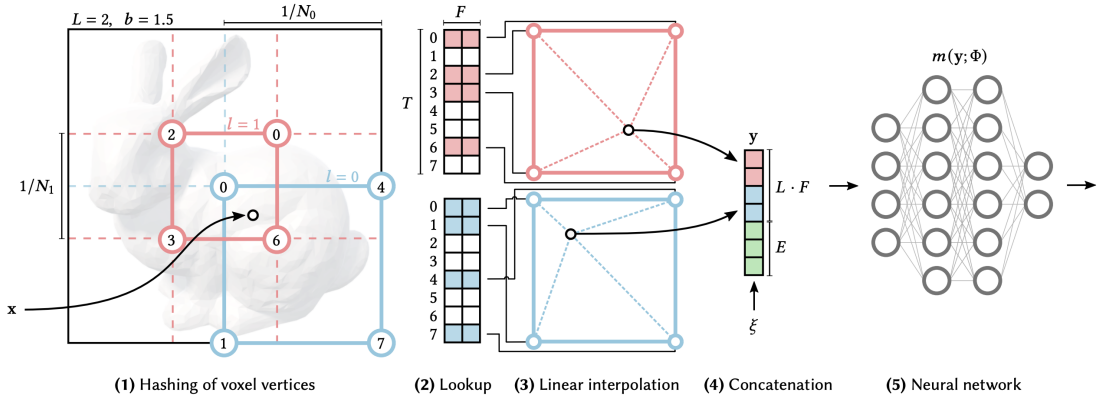

Figure 2: Multiresolution Hash Encoding

2.1 Hash Collision

In lower resolution levels, the hash table is large enough to hold the feature vectors for each corner of the voxels in its unique spot. However, with higher resolutions, it's probable that some hash table elements have to house more than one feature vector. This is called a hash collision.
If a hash collision occurs, the feature vectors are averaged, meaning that the largest gradients, which are most relevant to the loss function, will dominate. Although one could expect inaccurate results because of these collisions, the encoding is able to reconstruct scenes accurately. This is due to the different resolutions that are complementing each other with their own strengths.

2.2 Hyperparameters

2.2.1 Hashtable size T

Figure 3 visualizes the impact of the hash table size T with respect to training time and quality. While a larger hash table size T yields a more accurate result, it takes longer to train the encoding. In all shown examples, the performance is growing slower after a hash table size of 2^{19}, which can be explained by the cash of their hardware (RTX 3090 GPU) getting oversubscribed.

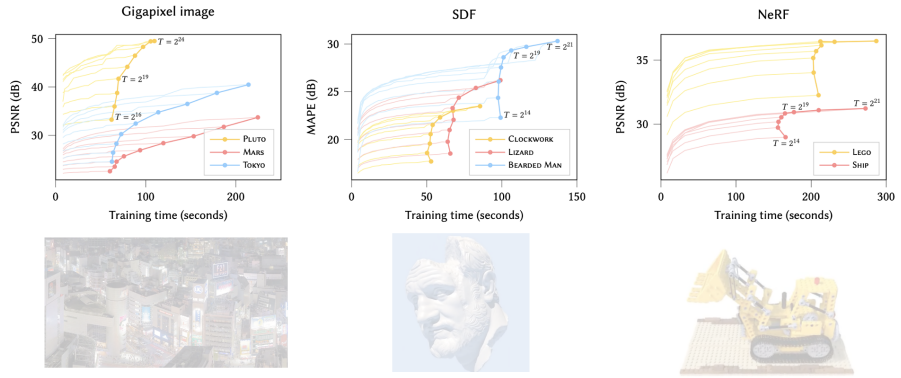

Figure 3: Hyperparameter T: Hash table size

2.2.2 Resolution Level L

The choice of numbers of resolution levels L and the feature vector's dimensionality F trade off quality and performance as well. Figure 4 illustrates this trade-off while keeping the number of trainable parameters $F * T * L$ at 2^{24} for SDF and NeRF and 2^{28} for Gigapixel images.
The authors found $L = 16$ and $F = 2$ to be the optimal choice for their tests.

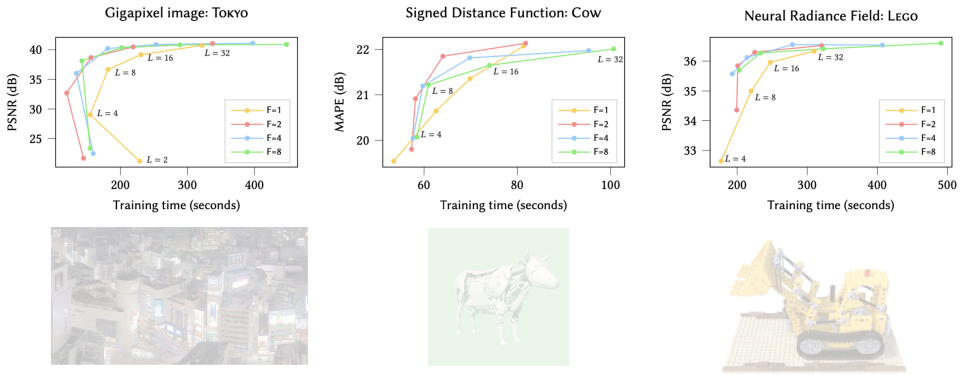

Figure 4: Hyperparameter L and F: Resolution level L and feature vector dimensionality F

3 Experiments and results

As discussed earlier, the authors evaluated the encoding on four different neural graphics primitives and compare it to previous encodings.

3.1 Gigapixel Image

With the state of the art method of adaptive coordinate networks (ACORN) [2], a PSNR of 38.59 dB is achieved on the Tokyo panorama from figure 5 after 36.9h of training. The authors achieved the same PSNR with their encoding after a training

time of only 2.5 minutes. This showcases the drastic speedup of the multiresolution hash encoding.

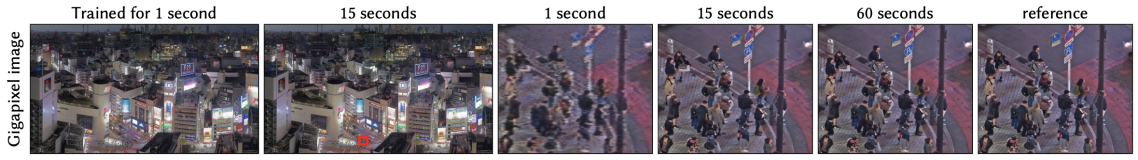

Figure 5: Gigapixel training stages and ACORN as reference. Tokyo gigapixel photograph © Trevor Dobson (CC BY-NC-ND 2.0)

3.2 Signed Distance Function

In Figure 6, the authors compare Neural Geometric Level of Detail (NGLOD) [5] and the frequency encoding [3] with their encoding. While NGLOD achieves the highest reconstruction quality, it is easy to see, that the frequency encoding struggles to learn sharp details of the models.
The new encoding produces roughly equal quality in terms of intersection over union, but it produces rough surfaces, rather than the smooth surfaces produced by NGLOD.

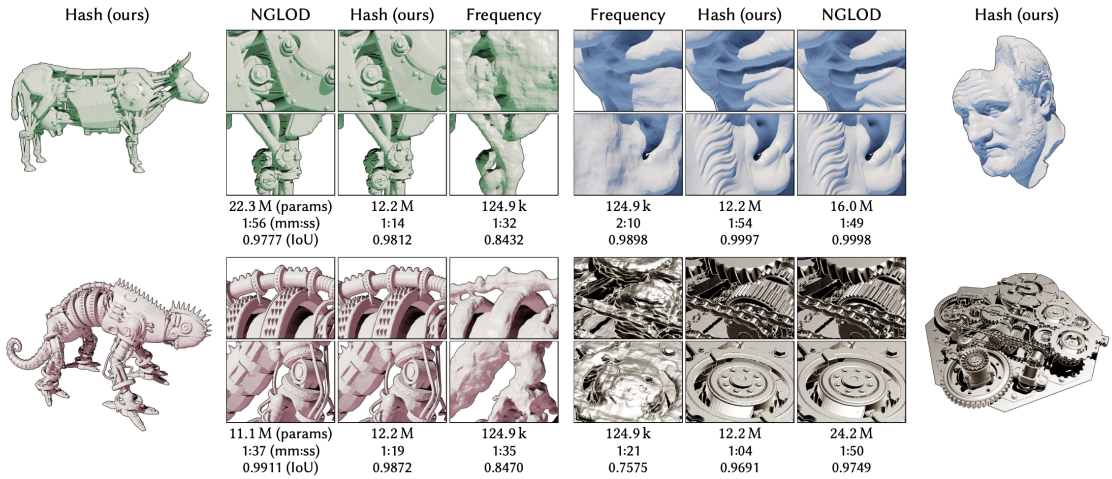

Figure 6: Signed distance function (SDF)

3.3 Neural Radiance Caching

The multiresolution hash encoding is compared against the triangle wave encoding [6] in figure 7. It yields much sharper results while having slightly worse performance. Interestingly, while the shadows are sharper with the multiresolution hash encoding, they look more realistic with the triangle wave encoding.

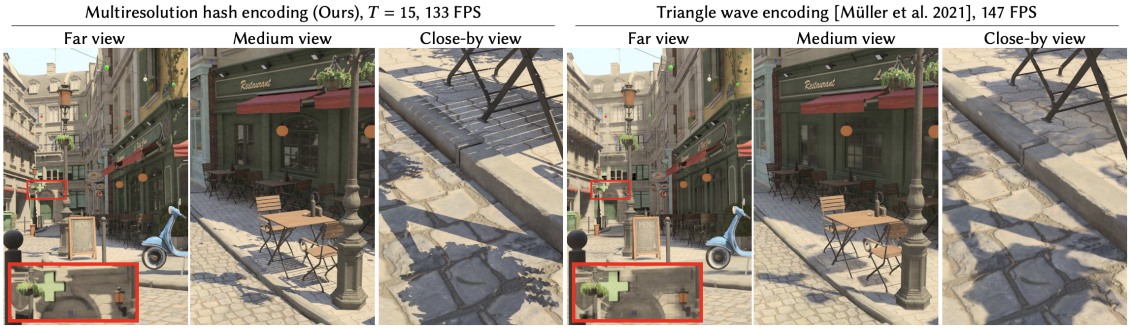

Figure 7: Neural radiance caching (NRC)

3.4 Neural Radiance and Density Fields (NeRF)

The table in Figure 8 compares the quality of neural radiance and density fields with different encodings after a specific training time. The gold, silver and bronze circles highlight the best encoding for each reconstructed 3D model. Very good results are achieved by mip-NeRF [1] on all models after a training time of hours. However, we can see that the new multiresolution hash encoding achieves similar results after a training time of only 5 minutes. Once again, this highlights the massive speedup and the power of the encoding created by the authors.

	Mic	Ficus	Chair	Hotdog	Materials	Drums	Ship	Lego	avg.
Ours: Hash (1 s)	26.09	21.30	21.55	21.63	22.07	17.76	20.38	18.83	21.202
Ours: Hash (5 s)	32.60	30.35	30.77	33.42	26.60	23.84	26.38	30.13	29.261
Ours: Hash (15 s)	34.76	32.26	32.95	35.56	28.25	25.23	28.56	33.68	31.407
Ours: Hash (1 min)	35.92	33.05	34.34	36.78	29.33	25.82	30.20	35.63	32.635
Ours: Hash (5 min)	36.22	33.51	35.00	37.40	29.78	26.02	31.10	36.39	33.176
mip-NeRF (~hours)	36.51	33.29	35.14	37.48	30.71	25.48	30.41	35.70	33.090
NSVF (~hours)	34.27	31.23	33.19	37.14	32.68	25.18	27.93	32.29	31.739
NeRF (~hours)	32.91	30.13	33.00	36.18	29.62	25.01	28.65	32.54	31.005
Ours: Frequency (5 min)	31.89	28.74	31.02	34.86	28.93	24.18	28.06	32.77	30.056
Ours: Frequency (1 min)	26.62	24.72	28.51	32.61	26.36	21.33	24.32	28.88	26.669

Figure 8: Comparison of encodings for Neural radiance and density fields (NeRF)

4 Conclusion

The proposed multiresolution hash encoding gives a learning based encoding, with many benefits. Due to multiple resolutions, it automatically focuses on relevant details. It is furthermore independent of the task and can be used for several applications such as gigapixel images, signed distance functions, neural radiance caching and neural radiance and density fields. Training a NeRF is speed up by several orders of magnitude and all other evaluated neural graphics primitives yield promising performance and quality of reconstruction. The hash collisions produce very small errors, which are especially noticeable in the example of signed distance functions. The treatment of such hash collisions could be explored in a future work.

References

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields, 2021.

[2] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation, 2021.

[3] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[4] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding, 2022.

[5] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes, 2021.

[6] Jan Novák Thomas Müller, Fabrice Rousselle and Alexander Keller. Real-time neural radiance caching for path tracing, 2021.