Article overview: Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Roman Rodionov, 522 group

Introduction. The way surfaces are represented in three-dimensional space is one of the key problems in computer graphics. Ideal representation should combine computational efficiency, compactness and the ability to accurately store high-frequency details.

To some extent, these conditions are satisfied by a recent approach based on the use of multi-layer perceptrons (MLPs). These methods can remember the shape of a surface (neural signed distance fields or calculation of primary and secondary rays), store information about the propagation of light through a scene (neural radiance fields and neural radiance caching), or map 2D space to an RGB image.

The key issue when applying MLP for implicit surface representation is the choice of encodings that transform the input parameter space into a higher-dimensional space, since in practice this has the greatest impact on the appearance and quality of final image, as well as on the computational efficiency of the method. Problems may be related to, for example, high computational costs, cache misses or too large number of parameters.

In this work I want to consider one of the latest and most promising encoding methods, which is actively used in state-of-the-art approaches.

Simple encoders. Early approaches to transforming input data into a multidimensional space include methods such as one hot encoding and the kernel trick. One of the first approaches applied to computer graphics problems was the encoding based on sine and cosine functions [2]:

$$enc(x) = (sin(2^0 x), sin(2^1 x), ..., sin(2^{L-1} x),$$

$$cos(2^0 x), cos(2^1 x), ..., cos(2^{L-1} x))$$

This type of encoding was applied for use in NeRF.

Müller et al. suggested a continuous variant of the one-hot encoding based on rasterizing a kernel, the one-blob encoding, which achieves more accurate results than frequency encoding in limited areas at the cost of being single-scale [3].

Dense parametric encoders. Another family of approaches is a combination of classical data structures such as a regular grid, voxel trees, etc., and learnable parameters. The input vector defines the coefficients with which these parameters are interpolated.

Although these methods require more memory, their use allows avoiding computational costs. Most of the parameters are contained in the encoding. When training MLP, we must update absolutely every parameter, in a network with parametric encoding, only a small part of parameters is used each time.

Since most of the parameters are in the encoding itself, the MLP that interprets the values obtained from the encoder can have few parameters. In addition, this approach allows the network to be trained much faster than a conventional large MLP.

The large memory consumption is a serious problem for such methods. If it is a three-dimensional regular dense grid with trainable parameters at the nodes, then the amount of memory consumed grows as $O(N^3)$, proportional to the grid resolution. However, the number of parameters that are actually used to define a surface in three-dimensional space is growing at a rate of $O(N^2)$. So we have a lot of unnecessary parameters.

The quality of methods with regular grid can be improved by using a multi-level grid with different resolutions. Feature vectors are interpolated at each level and concatenated into one. Fewer parameters can be stored in the nodes of such a grid while giving the same or higher quality, which can help to reduce memory consumption.

Sparse parametric encoders. If the surface is specified in advance, the amount of memory used can be reduced by using an octree or a sparse voxel grid. These structures let us only contain parameters that are close to surface. However, this approach is not applicable to a number of methods in which this condition is not met (NeRF-like approaches). For such cases, there are approaches that refine the structure during training.

Many of the above-mentioned shortcomings were solved using the method described in the article under consideration. The approach is based on a combination of a multiresolutional regular grid and a hash table.

Multiresolutional hash grid encoder [1]. Consider a fully connected neural network $m(y; \Phi)$ with encoding $y = enc(x; \theta)$. Now we have two types of trainable parameters: weight parameters Φ and encoding parameters θ. Encoding parameters are arranged into L levels containing up to T feature vectors with dimensionality F.

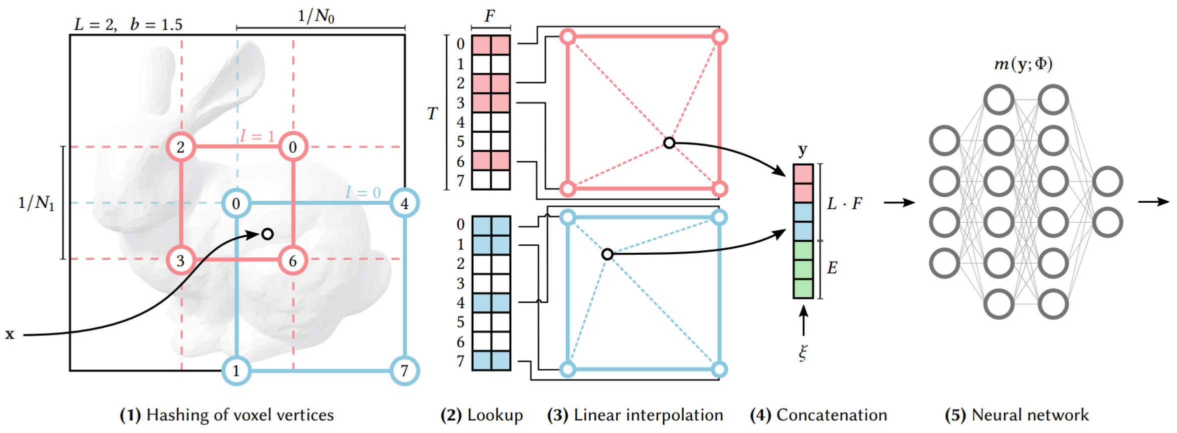

Figure 1: Scheme of neural network with multiresolutional hash grid encoder [1].

The main idea of hash grid is to store indices, that direct to feature vectors in hash table, at grid nodes (Figure 1). Resolution of grids in layers is chosen to be a geometric progression between the coarsest and finest resolution. If the number of nodes at coarse levels is less than T, the nodes are matched with the feature table as 1:1. Otherwise indices at nodes are chosen using hash function:

$$h(x) = \bigoplus_{i=1}^{d} x_i \pi_i \pmod{T},$$

where $\oplus$ denotes the bit-wise XOR and π_i are large prime numbers. The problem of hash collisions is supposed to be solved by MLP decoder.

Thus, the number of parameters in multiresolutional hash grid doesn't depend on grid resolution, but on chosen hash table size (T), number of levels (L) and size of feature vector (F), and bounded by $T \cdot L \cdot F$.

The method under consideration has a serious advantage over conventional parametric structures - flexibility. The choice of parameters, such as the size of the hash table, the number of grid levels and the length of the feature vector, allows finding a balance between the degree of compression of a three-dimensional scene, quality of the resulting image and training time (Figure 2).

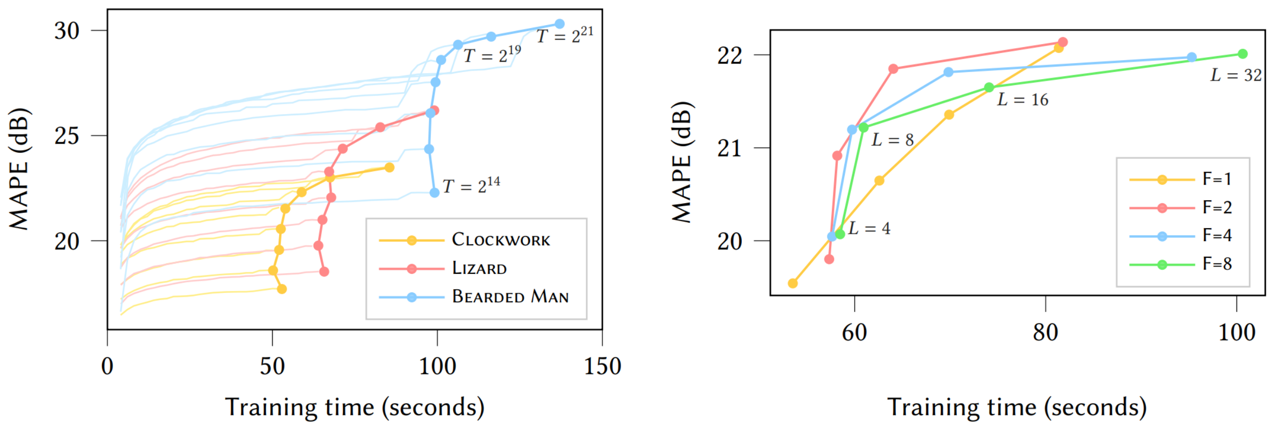

Figure 2: Dependence of quality of SDF with multiresolutional hash encoding on hyperparameters and training time [1].

Signed distance functions. There are a lot of types of implicit neural representations. In this overview I want to consider neural signed distance functions.

Signed distance functions (SDF) represent 3D shapes as the zero level-set of some function of position. First approaches for neural SDF representation used large MLP for one or more shapes at a time [4]. Another approach used small mlp with octree of trainable feature vectors and reached a great balance between speed and quality [5]. The problem with this method is that normal SDFs are defined in the entire space, while NGLOD is defined only in voxels lying in the octree, which does not allow using some techniques, such as soft shadows.

The article under consideration compares neural SDFs using different encodings: frequency, NGLOD, and hash grid. In my implementation of neural rendering [6], I implemented the first and the last (Figure 3). It is worth noting that real-time implementations of neural rendering are often written using CUDA and

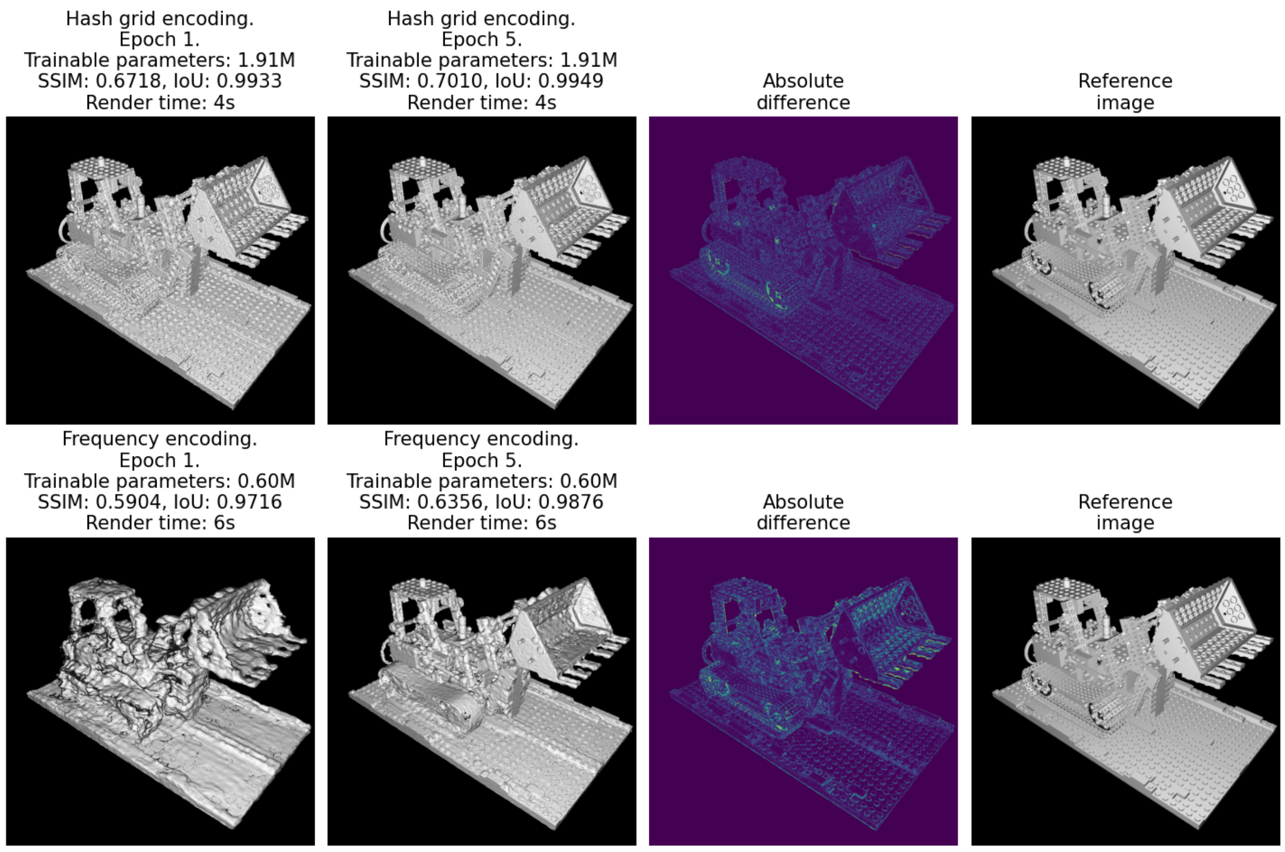
Hash grid encoding. Epoch 1. Trainable parameters: 1.91M SSIM: 0.6718, IoU: 0.9933 Render time: 4s
Hash grid encoding. Epoch 5. Trainable parameters: 1.91M SSIM: 0.7010, IoU: 0.9949 Render time: 4s
Absolute difference
Reference image
Frequency encoding. Epoch 1. Trainable parameters: 0.60M SSIM: 0.5904, IoU: 0.9716 Render time: 6s
Frequency encoding. Epoch 5. Trainable parameters: 0.60M SSIM: 0.6356, IoU: 0.9876 Render time: 6s
Absolute difference
Reference image

Figure 3: Comparison of two different encoding methods in SDF rendering task. First network has 2 hidden layers in decoder, while second one has 8 hidden layers. Both have 256 neurons per hidden layer. Render method - sphere tracing with 150 steps. Resolution: 800 x 800.

C++, so my implementation, using PyTorch and Python, does not claim to be fast enough for real-time applications.

As expected, the network with a hash grid wins over frequency encoding in almost all parameters: quality, speed, convergence. If you pay attention to the comparison of the hash grid and NGLOD at the article, the first one turns out to be comparable to the second one in terms of quality and speed of work, and memory costs. One of the notable issues is noticeable noise on the surface, which the authors attribute to hash collisions. It is worth noting that both approaches are quite similar, except for the hash collisions and the volume in which SDF is defined.

Conclusion. The structure described in the paper turned out to be quite an innovative and thoughtful approach. It has a wide range of applications in completely different tasks, including neural rendering and data compression. The authors note the shortcomings of their work, such as hash collisions, however the approach has many more advantages, including speed of work and convergence speed, quality and compactness.

References

[1] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. 41, 4, Article 102 (July 2022), 15 pages. https://doi.org/10.1145/3528223.3530127

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. (June 2017). DOI:https://doi.org/10.48550/arXiv.1706.03762

[3] Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural Importance Sampling. ACM Trans. Graph. 38, 5, Article 145 (October 2019), 19 pages. https://doi.org/10.1145/3341156

[4] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. Retrieved from https://arxiv.org/abs/1901.05103

[5] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. Retrieved from https://arxiv.org/abs/2101.10994

[6] Implementation of neural SDF using multiresolution hash grid encoding [Web resource] // URL: https://github.com/RomanRodionov/NeuralSDF_Demo