Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

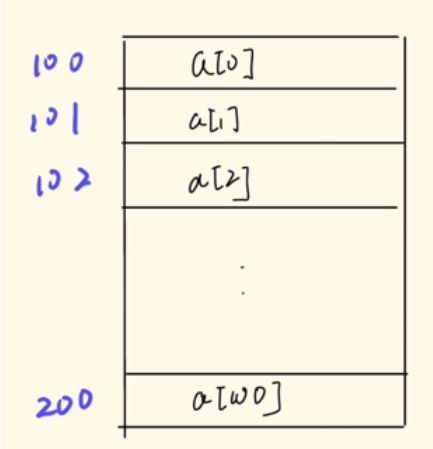
Year: 2022

前置知识

哈希（散列）存储

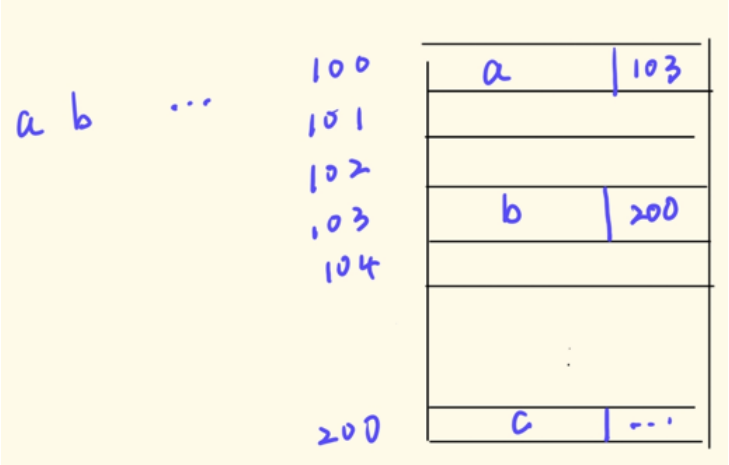
顺序存储

在计算机中用一组地址连续的存储单元依次存储线性表的各个数据元素,称作线性表的顺序存储结构。



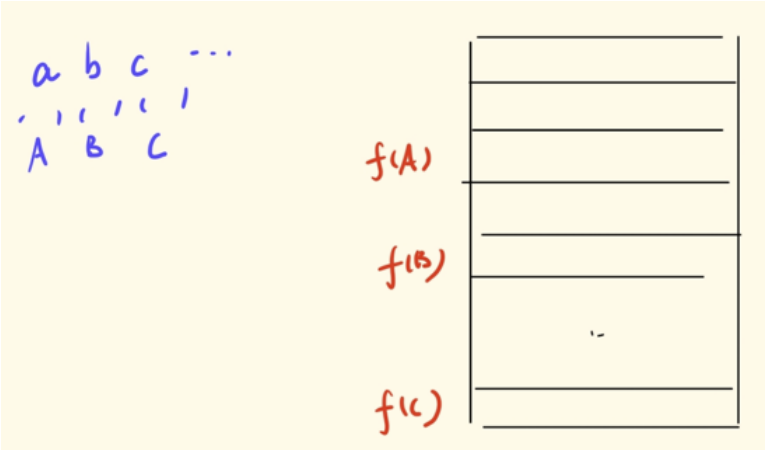
链式存储

在计算机中用一组任意的存储单元存储线性表的数据元素(这组存储单元可以是连续的,也可以是不连续的)。它不要求逻辑上相邻的元素在物理位置上也相邻.因此它没有顺序存储结构所具有的弱点,但也同时失去了顺序表可随机存取的优点。



哈希（散列）存储

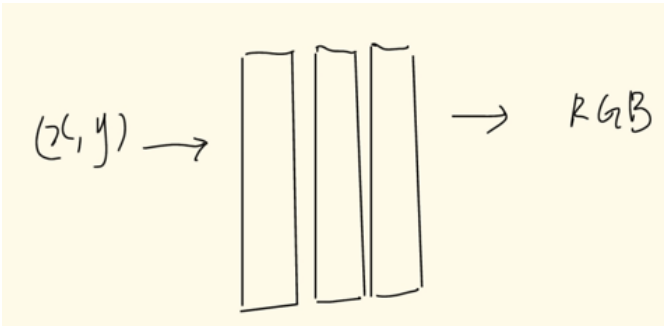
由节点的关键码值决定节点的存储地址。



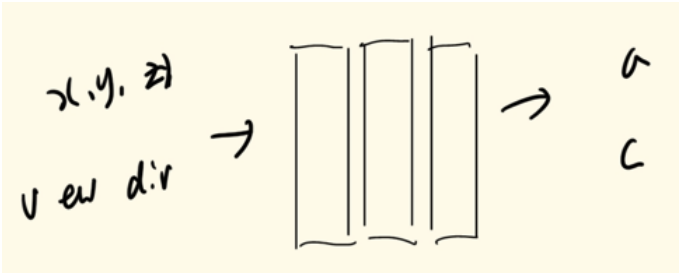
Neural graphics primitives

我们对计算机视觉或图形学中二维或三维数据的存储，有许多的方法，一种你可以用标准化的数据结构进行存储，比如你想存一幅100x100三通道的图像的话， 你可以用HXWX3B来存，大概30KB的大小，在深度学习的领域，你也可以用神经网络来存二维或三维的数据，这种方式叫做Neural graphics primitives，有两个比较典型的例子：

1. 在超大图像压缩中，可以训练一个神经网络，输入一个图像的坐标位置，得到相应位置的RGB值，当神经网络的参数与原来的（HxWx3B)要小时，就完成了一个图像的压缩任务。



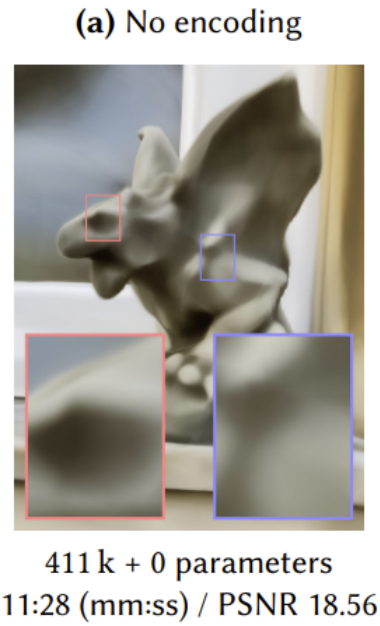
2.神经辐射场，想要得到空间中某一个粒子的颜色和密度，也是将粒子对应的位置和观察角度输入神经网络，得到粒子对应的密度和颜色值，其实也是将三维空间里的外观信息保存在一个神经网络中。



位置编码

使用神经网络对二维或三维数据的外观信息进行保存的话，其输入的位置编码是很重要的。

这个图片显示了，使用神经网络进行图像压缩任务中，输入不进行编码的结果。



这是这篇论文里有贴过的没有位置编码的重建效果。

固定编码

使用某种映射方式将输入从低维映射到高维空间，如在Nerf中使用到的频率编码方式：

$$(x,y,z)->(x,y,z,sin2^0x,sin2^0y,sin2^0z,cos2^0x,cos2^0y,cos2^0z,...,,sin2^Lx,sin2^Ly,sin2^Lz,cos2^Lx,cos2^Ly,cos2^Lz)$$

从原来的 3 维变为了 $3+6*(L+1)$ 维。

可学习编码

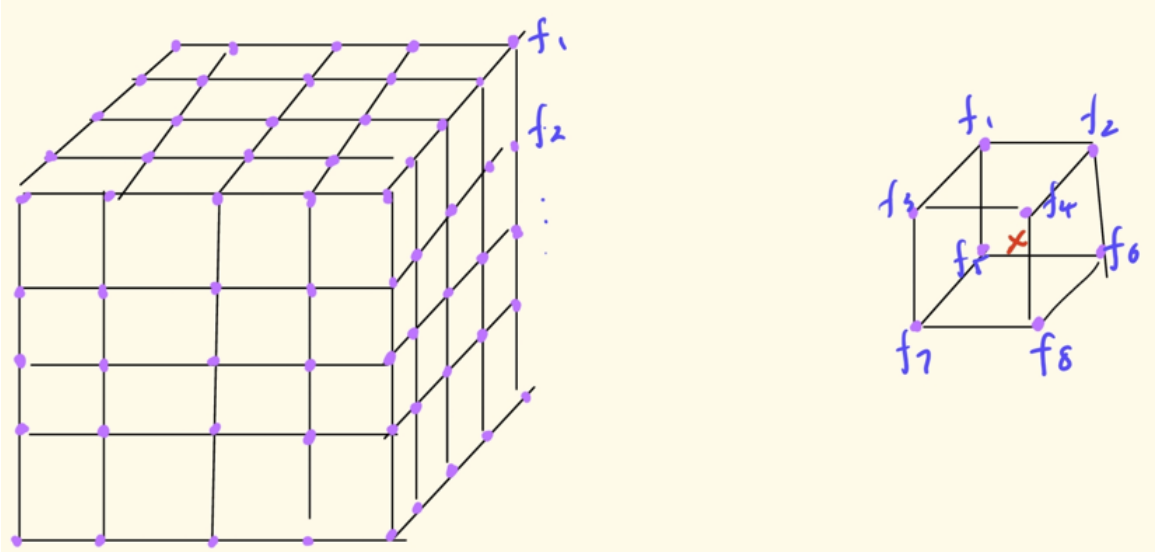
可学习的编码方式，是指某一个位置的编码是由可学习的，不由某种固定的映射决定，因为三维空间的坐标是连续的，不可能对连续空间的所有位置进行学习，所以通常采用Dense Grid的方式，对Dense Grid上顶点坐标的编码进行学习，这里又分为Single resolution和Multi resolution两种。

(b) Frequency
[Mildenhall et al. 2020]

438 k + 0
12:45 / PSNR 22.90

这是使用频率编码进行训练的结果，网络大小与无编码方式差不多，但是效果好非常多。

Single Resolution

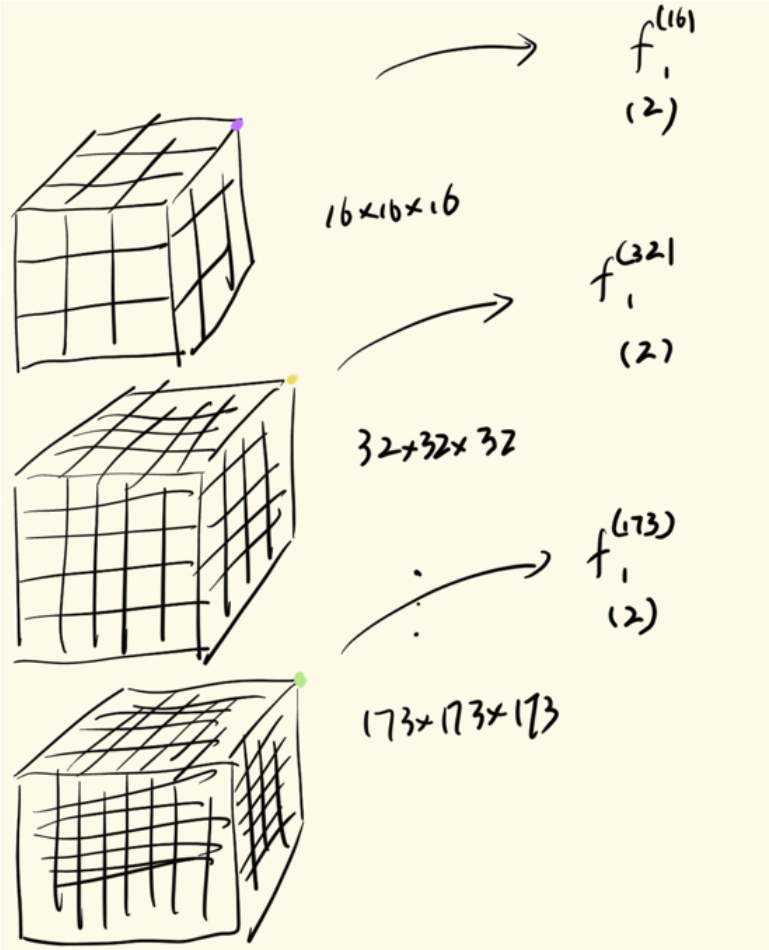


将三维空间缩放到 0 − 1 的空间内，对三维空间进行切分，分为 $n \times n \times n$ 的网格，对网格上每个点的位置编码进行学习，论文里将分为了 $128 \times 128 \times 128$ 这么多个网格，每个网络上位置编码的维度为16。输入的时候如果输入位置并不在某个顶点上，就使用三线性插值的方式对位置编码进行插值。

这是使用single resolution的结果，网络大小比Frequency要小非常多，但是效果更好，这里编码方式的计算公式：

$$128 * 128 * 128 * 16 = 33.6M$$

Multiple Resolution



使用多个不大小的网络对空间进行切分，对不同网络大小的点进行位置编码，在论文里共有8个不同大小的网格，每个网格上对应点的编码维度为2。

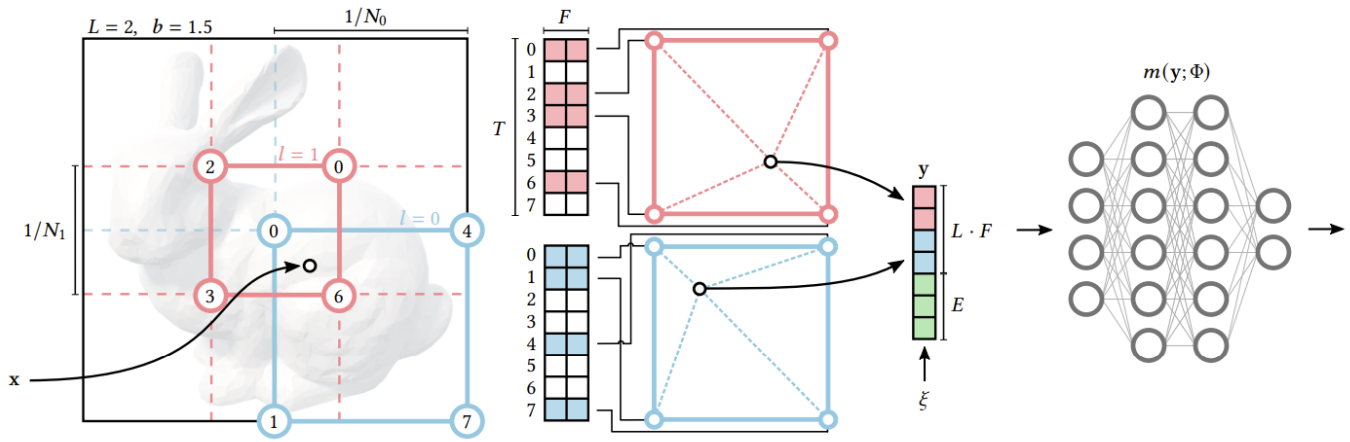
在Multiple Resolution下，想要知道某个点的编码是多少呢，就在不同尺度下的网格上进行三线性插值，将插值得到的8个2维的位置编码concat一起就可以(16维)。

这是Multiple Resolution的效果，编码参数比Single Resolution小，但效果比Single Resolution还要好，参数计算公式应为：

$$16 * 16 * 16 * 2 + ... + 173 * 173 * 173 * 2$$

虽然Single Resolution和Multiple Resolution的效果都不错，但是其编码学习参数量还是很大，并且想要最后结果的效果好一些，当然是希望网格尽可能切小一点，但是网格切小了，参数量成三次方增加，是否需要这些多参数量呢？其实不尽然，因为在Nerf训练里面，其实我们只希望在有物体的地方进行合理的位置编码，并不需要太care没有物体的地方。

Multiresolution Hash Encoding



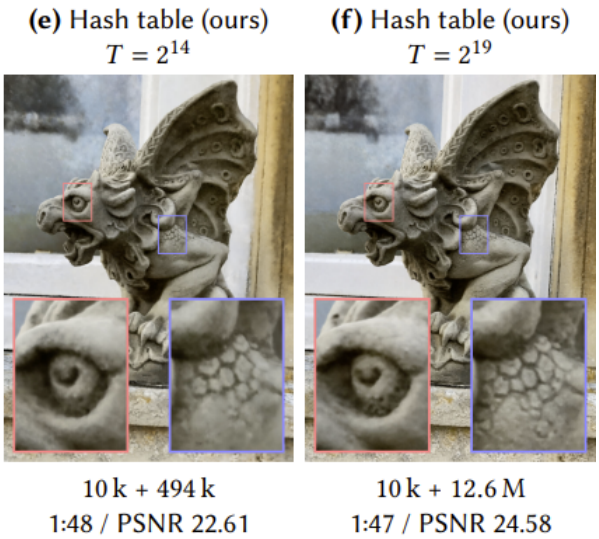
还是按Multiple Resolution的方式，把三维空间切成不同的大小的网格，每个网格上的点通过哈希映射，映射到哈希表的对应位置，为每个分辨率下的网格，提供一个哈希表，其中低分辨率的哈希表比较短，高分辨率的网格哈希表比较长一些，在哈希表中对每项进行一个位置编码，网格点的编码对应相应哈希表项中的位置编码，哈希表项中的位置编码是可学习的。

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

由于哈希表的长度比原来网格上点的长度要小很多，所以可学习的编码参数会变少！以分辨率最小的网格为例，本来需要学习的参数至少是：

$$512 \times 512 \times 512 = 2^{27}$$

哈希表为长度为 2^24 。



这是使用Hash encoding的结果，参数比Single还有multi resolution要少很多，但是效果也不比他们差。

How it works?

为什么Hash encoding的方式有效，这个结果看起来是比较反直觉的，因为论文里并没有处理hash冲突，也就是不同的坐标点可能会映射到相同的哈希表项，也就是拥有同样的位置编码，这样真的ｗｏｒｋ吗?
我个人觉得这个编码方式可以成功的原因有两个，一个是其实很多网格点的编码我们是不CARE的，比如这个地方没有东西，对最后成像的颜色值没有贡献，即使编码不精确也没有关系，第二点是论文里有提到的，multiresolution在粗网格上面，其实ｈａｓｈ是一一对应的，也就是没有哈希冲突，因此可以保证粗网格上编码是正确的，这样可以有效缓解细粒度网格上映射不准确的问题。

Hash函数

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

$$\pi 1 = 1$$
$$\pi 2 = 2654435761$$
$$\pi 3 = 805459861$$

用一个实例说明，比如你的输入是点在整个网格上的位置是(x,y,z)(整型数据)。

$$(x * \pi 1) \oplus (y * \pi 2) \oplus (y * \pi 2) mod T$$

最后得到就是hash表中的位置，关于为什么选择这个Hash函数，我并没有在论文里找到很确切的解释，大概解释可能是简单，cuda处理速度快，不同坐标经过不同的π_i之后，原本的位置会被打乱，鲁棒性更强。