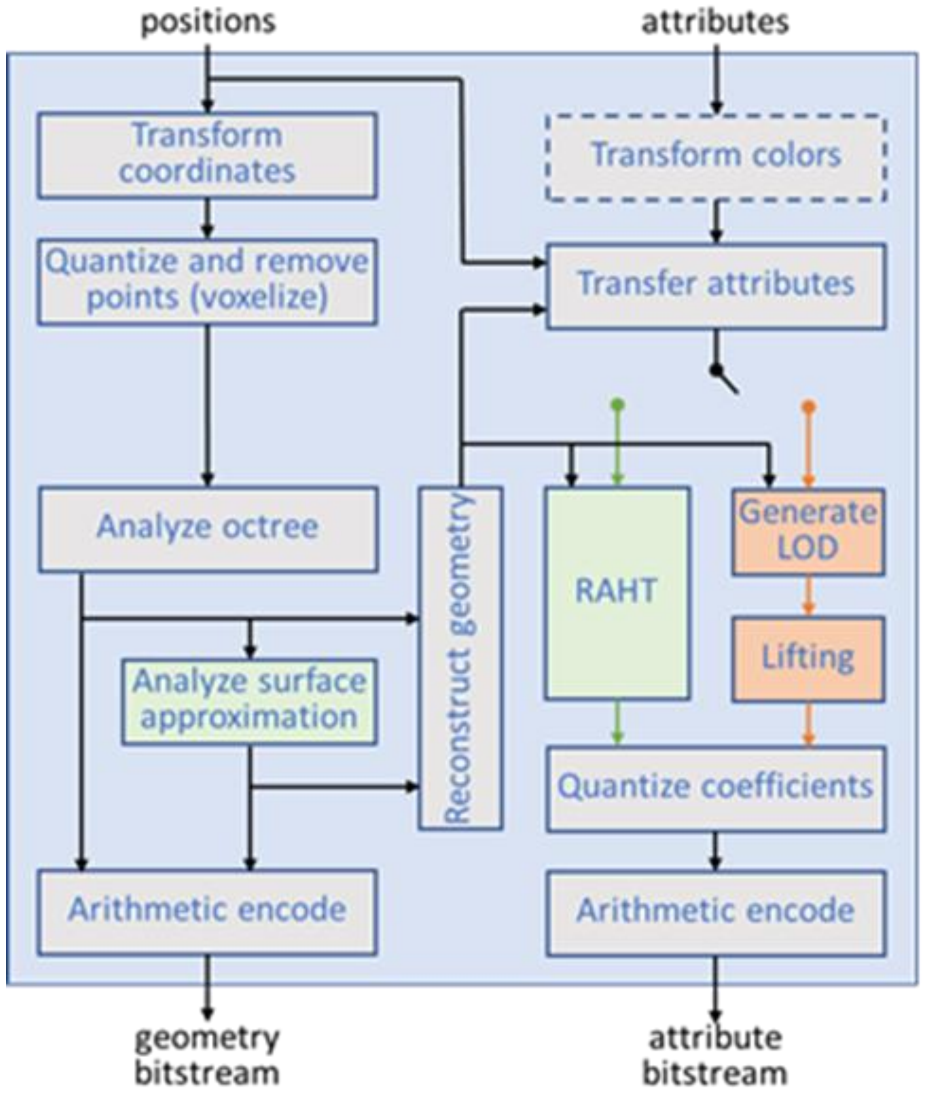
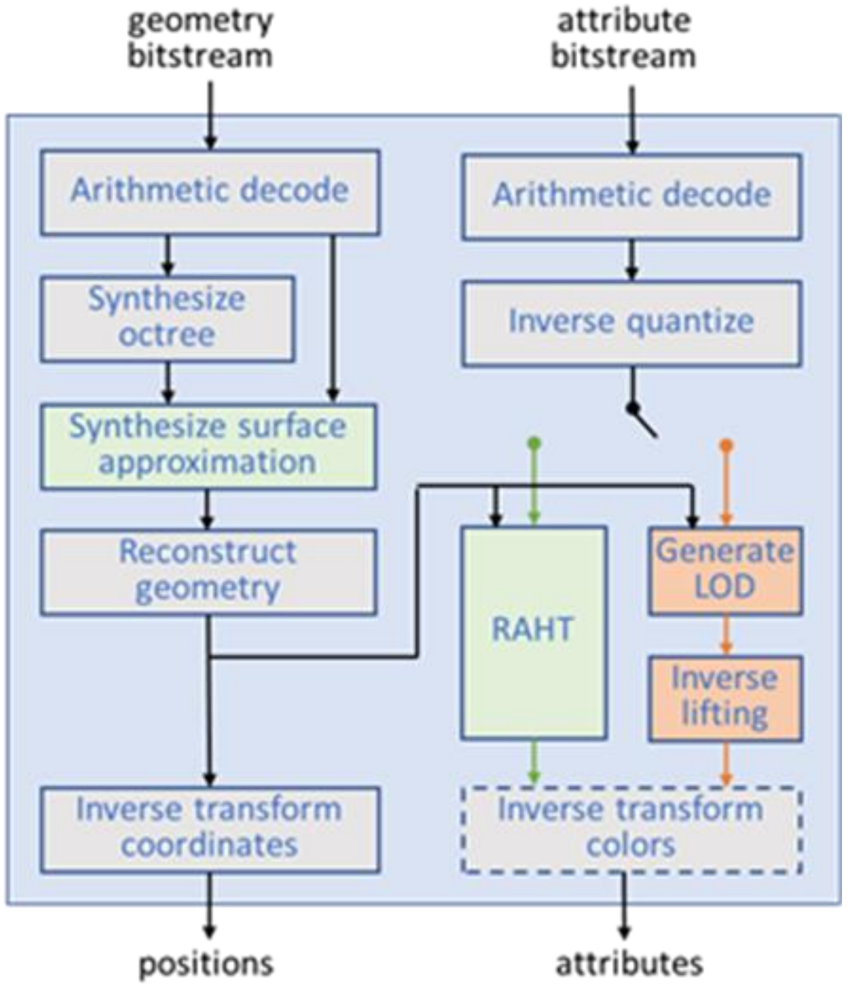
Introduction to GPCC

Overview

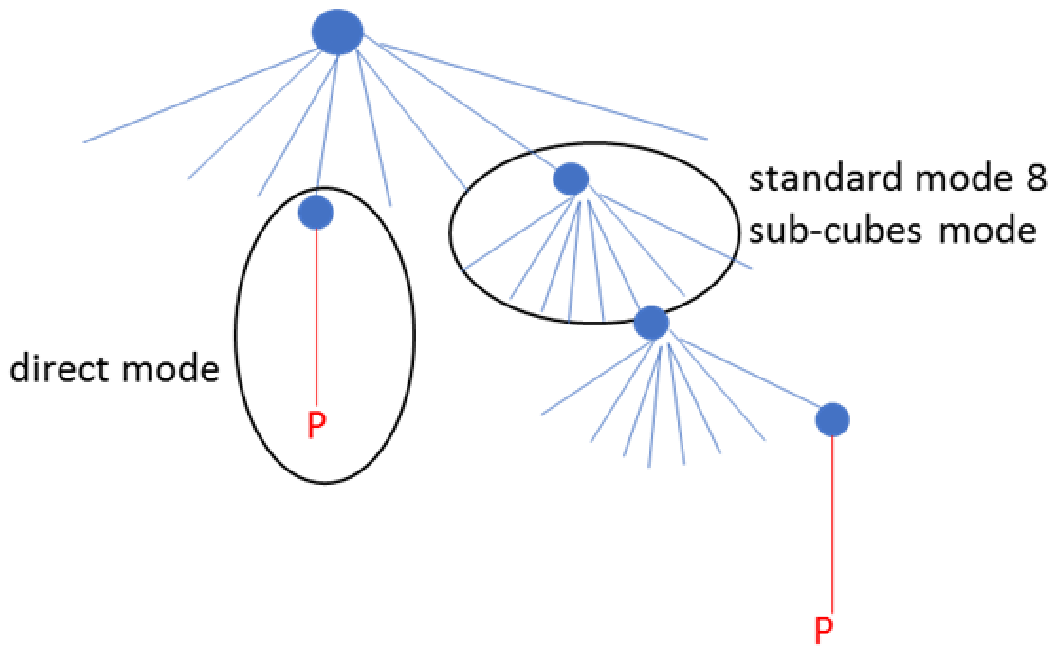

positions
attributes
Transform coordinates
Transform colors
Quantize and remove points (voxelize)
Transfer attributes
Analyze octree
Reconstruct geometry
RAHT
Generate LOD
Lifting
Analyze surface approximation
Quantize coefficients
Arithmetic encode
Arithmetic encode
geometry bitstream
attribute bitstream


geometry bitstream
attribute bitstream
Arithmetic decode
Arithmetic decode
Synthesize octree
Inverse quantize
Synthesize surface approximation
Reconstruct geometry
RAHT
Generate LOD
Inverse lifting
Inverse transform coordinates
Inverse transform colors
positions
attributes

- Octree geometry encoding
  - DCM
  - Neighbour-Dependent Entropy Context
- Attribute coding
  - LOD
  - Lifting
  - RAHT
- Functionality
  - Tile and Slice

- Octree geometry encoding
    - DCM
    - Neighbour-Dependent Entropy Context
- Attribute coding
    - LOD
    - Lifting
    - RAHT
- Functionality
    - Tile and Slice

Direct coding mode(DCM)



有孤立点的情况直接编码其在父节点xyz
轴相对位置，省5bits

Direct coding mode(DCM)

给每个叶子节点加一个flag表示其是否是孤立点，这种方式消耗太多bits→IDCMD(Inferred Direct Coding Mode)

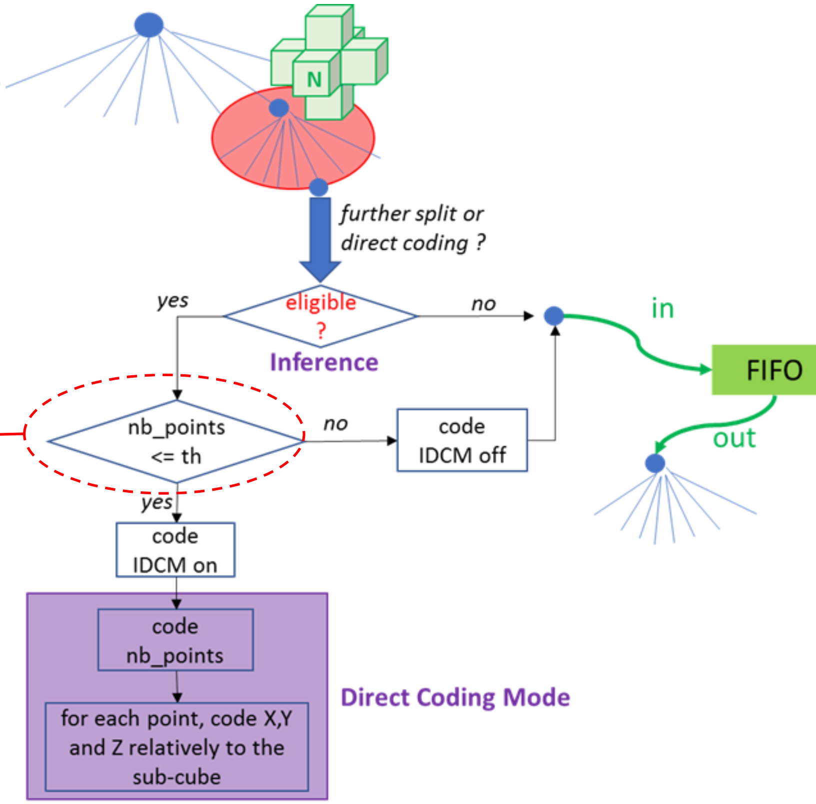
根据父节点和邻居节点的占用情况决定当前点是否使用DCM

判断条件包括:

- 父节点的子节点只有一个非空(即当前点)且父节点所在的层至多两个非空
- 父节点的子节点只有一个非空(即当前点)且当前点没有共面非空邻居

可以通过cfg调节满足哪个或哪些判断条件才使用DCM。

DCM可以编码多个点，通过cfg可调。启用DCM后先编码孤立点的个数再编码每个点的相对位置。

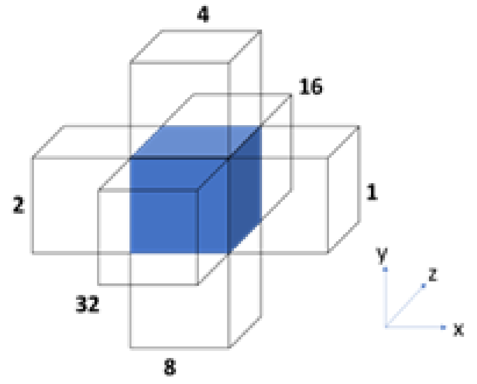


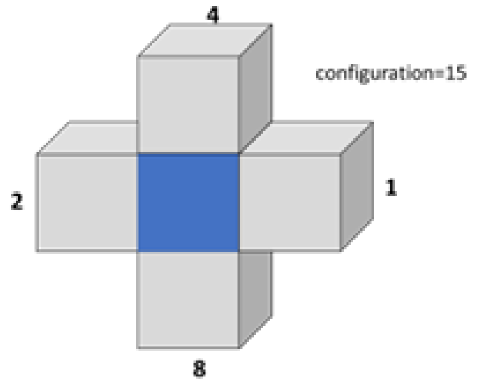
- Octree geometry encoding
  - DCM
  - Neighbour-Dependent Entropy Context
- Attribute coding
  - LOD
  - Lifting
  - RAHT
- Functionality
  - Tile and Slice

Neighbour-Dependent Entropy Context

根据邻居节点的占用情况获得当前节点占用情况的先验概率用以算术编码

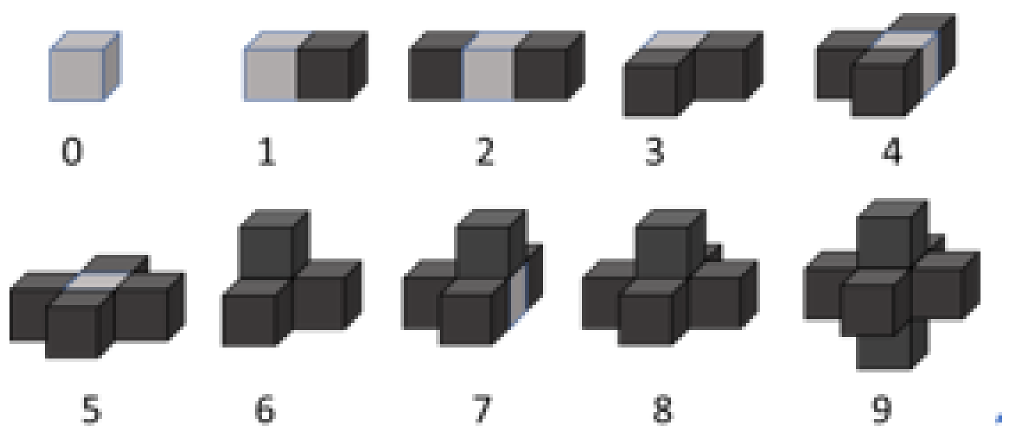
当前节点的父节点的共面邻居共有64种占用情况（每个节点有空和非空两种情况，2^6），在硬件实现时就需要64块内存单元存每种占用情况的分布，如果再考虑节点内的预测，至多有128*64= 8192 states



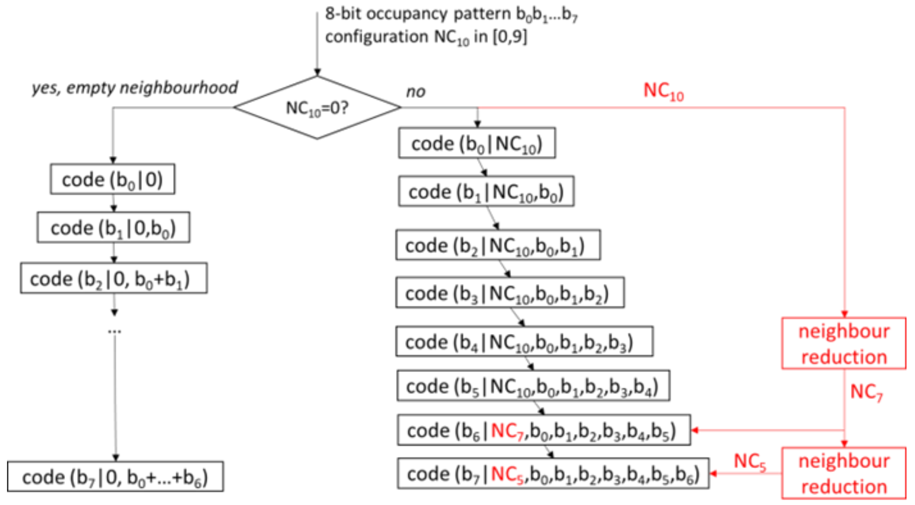


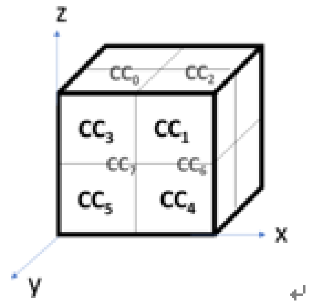
Neighbour-Dependent Entropy Context

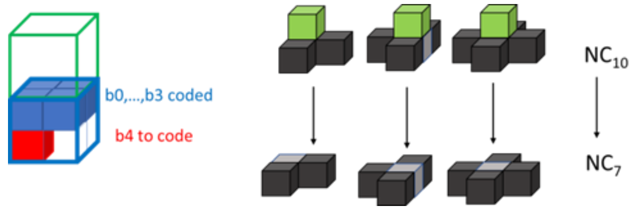
利用旋转不变性和对称不变性进一步减少到10个

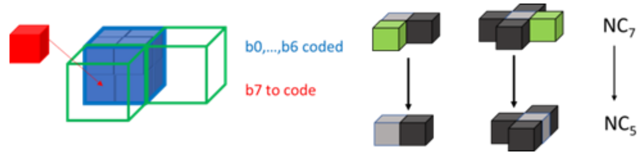


Further Reduction


8-bit occupancy pattern $b_0b_1…b_7$
configuration NC_{10} in [0,9]
yes, empty neighbourhood
NC_{10}=0?
no
NC_{10}
code (b_0|0)
code (b_1|0,b_0)
code (b_2|0, b_0+b_1)
…
code (b_7|0, b_0+…+b_6)
code (b_0|NC_{10})
code (b_1|NC_{10},b_0)
code (b_2|NC_{10},b_0,b_1)
code (b_3|NC_{10},b_0,b_1,b_2)
code (b_4|NC_{10},b_0,b_1,b_2,b_3)
code (b_5|NC_{10},b_0,b_1,b_2,b_3,b_4)
code (b_6|NC_7,b_0,b_1,b_2,b_3,b_4,b_5)
code (b_7|NC_5,b_0,b_1,b_2,b_3,b_4,b_5,b_6)
neighbour reduction
NC_7
NC_5
neighbour reduction



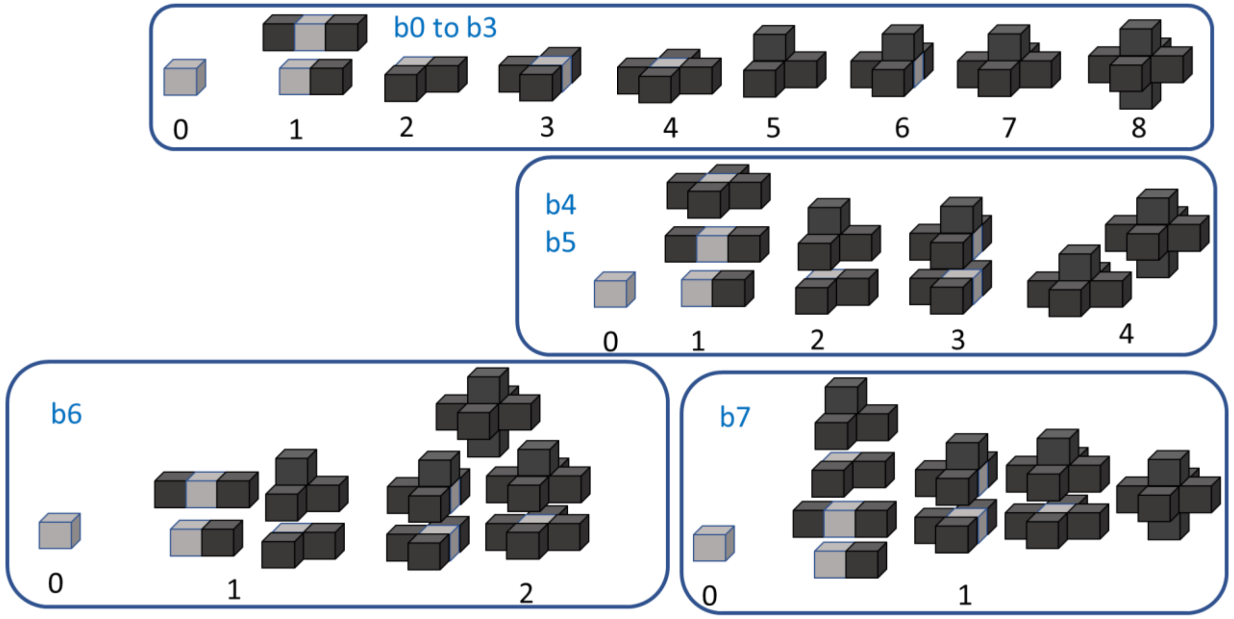




至多128*4+8 = 520

Neighbour-Dependent Entropy Context

Further Reduction



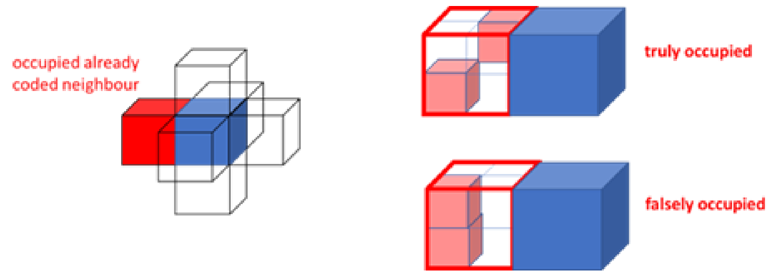
Occupancy bit	Number of neighbour config	States from NC=0	States from NC>0	Total states	Old total using NC_{10}
b_0	9	1	8*1=8	**9**	*10*
b_1	9	2	8*2=16	**18**	*20*
b_2	9	3	8*4=32	**35**	*39*
b_3	9	4	8*8=64	**68**	*76*
b_4	5	5	4*16=64	**69**	*149*
b_5	5	6	4*32=128	**134**	*294*
b_6	3	7	2*64=128	**135**	*391*
b_7	2	8	1*128=128	**136**	*520*

reduction一定程度上会带来loss，合并相似度高的loss较小，是性能和硬件实现的trade-off

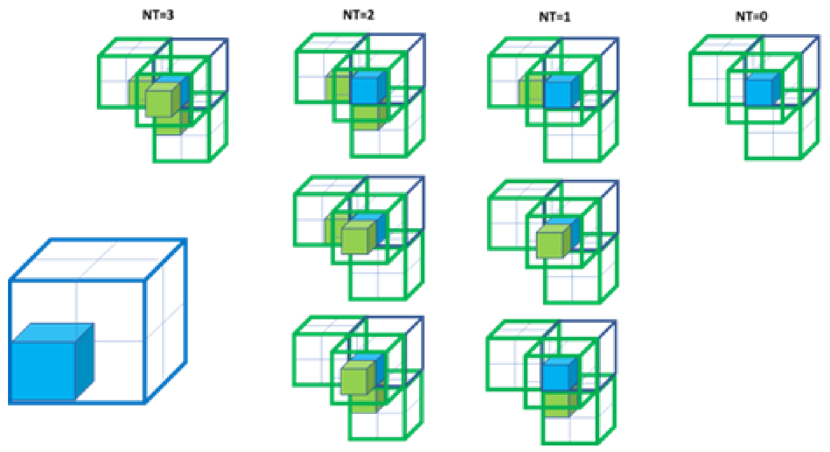
Neighbour-Dependent Entropy Context

按照广度优先搜索的顺序，当前节点父节点的6个邻居中有3个已经编码，可以进一步利用这3个已经编码的邻居的子节点占用情况进一步预测当前点的占用情况概率

Falsely occupied neighbour



Occupied child nodes adjacent to a current sub-node



NT--Number Touching。可进一步离散成两种情况，原上下文数量翻倍

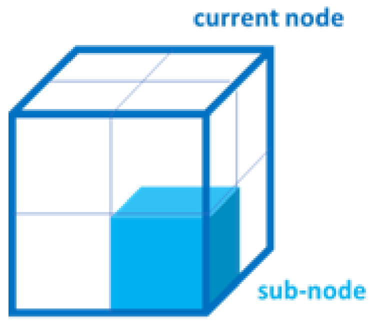
Neighbour-Dependent Entropy Context

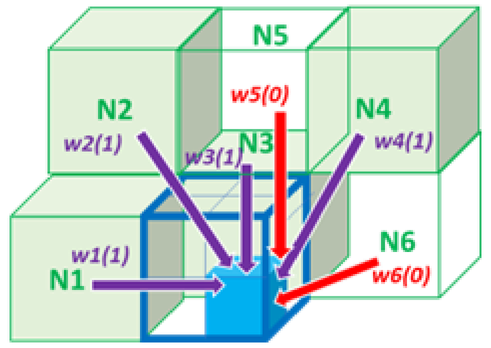
仅采用共面邻居提供的信息可能不足，但使用共面共线共点的26个邻居会使上下文数量巨大→occupancy score

$$\text{score}_m = \frac{1}{26}\sum_{k=1}^{26} w_{k,m}(\delta_k)$$

$$\boldsymbol{w_{k,m}}(\boldsymbol{\delta_k}) = \boldsymbol{W}(\boldsymbol{d_{k,m}}, \boldsymbol{\delta_k})$$

$$\boldsymbol{W}(\boldsymbol{d_{k,m}}, \boldsymbol{\delta_k}) = \begin{cases} W0(\boldsymbol{d_{k,m}}) & if\ \ \boldsymbol{\delta_k} = 0 \\ W1(\boldsymbol{d_{k,m}}) & if\ \ \boldsymbol{\delta_k} = 1 \end{cases}$$





W0 = {-1, -6, 12, 20, 14, 28, 22, 12},
W1 = {27, 39, 20, 8, 18, 4, 11, 18}.

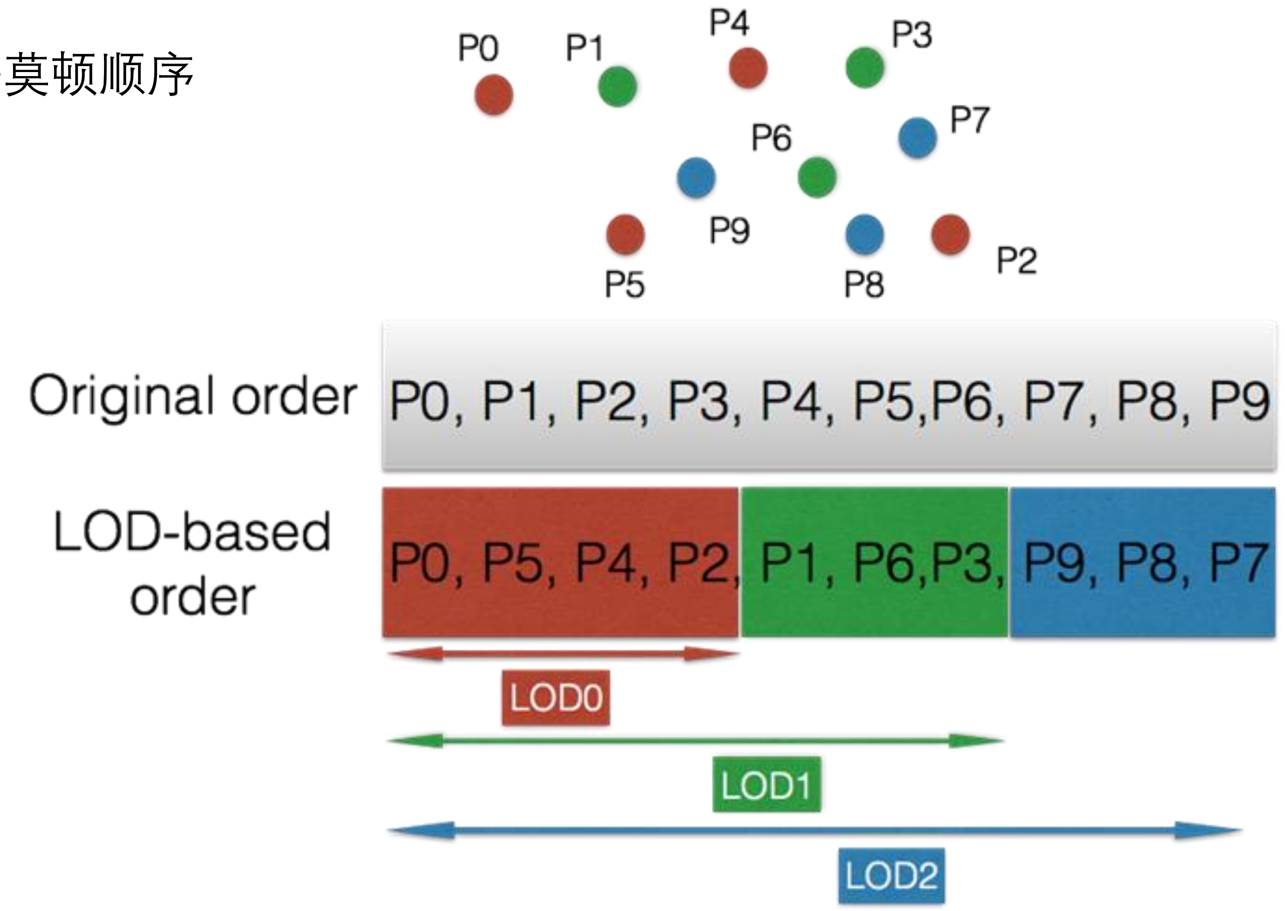
通过score预测当前子节点占用情况三种情况：占用，非占用，不可预测。上下文数量增加三倍
最终的最大上下文数量为$136 \times 2 \times 3 = 816$

- Octree geometry encoding
  - DCM
  - Neighbour-Dependent Entropy Context
- Attribute coding
  - LOD
  - Lifting
  - RAHT
- Functionality
  - Tile and Slice

Level of detail generation

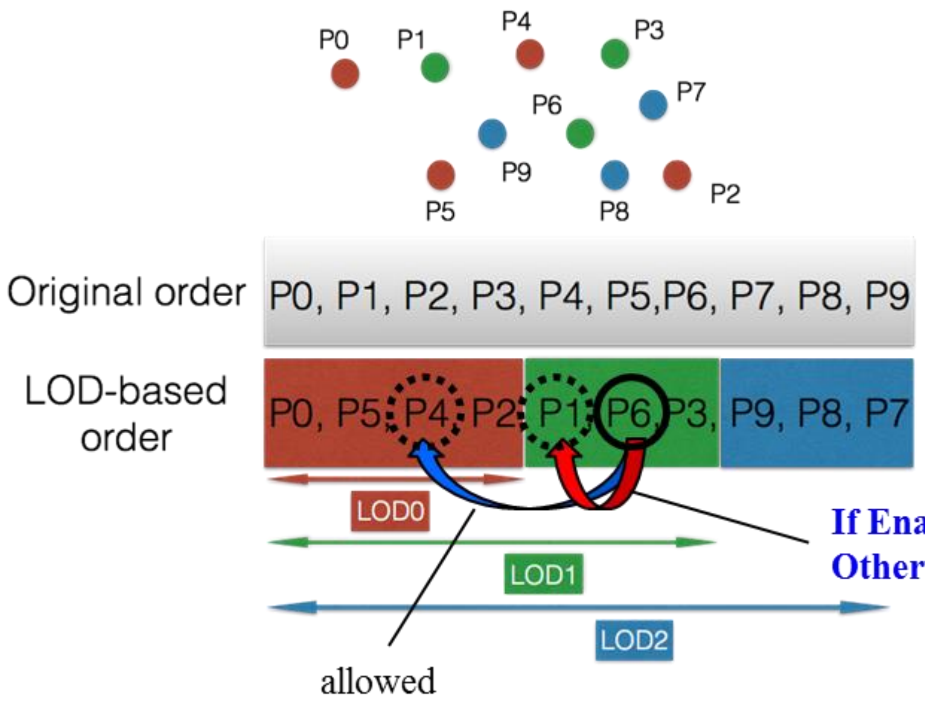
给定一系列距离$(d_l)_{l=0\ldots L-1}$，就点云分成多个精细程度递增的level
generation流程同FPS

low-complexity实现：给定一系列$(k_l)_{l=0\ldots L-1}$，将点按莫顿顺序排列，每隔k_l个点选一个点形成第l层

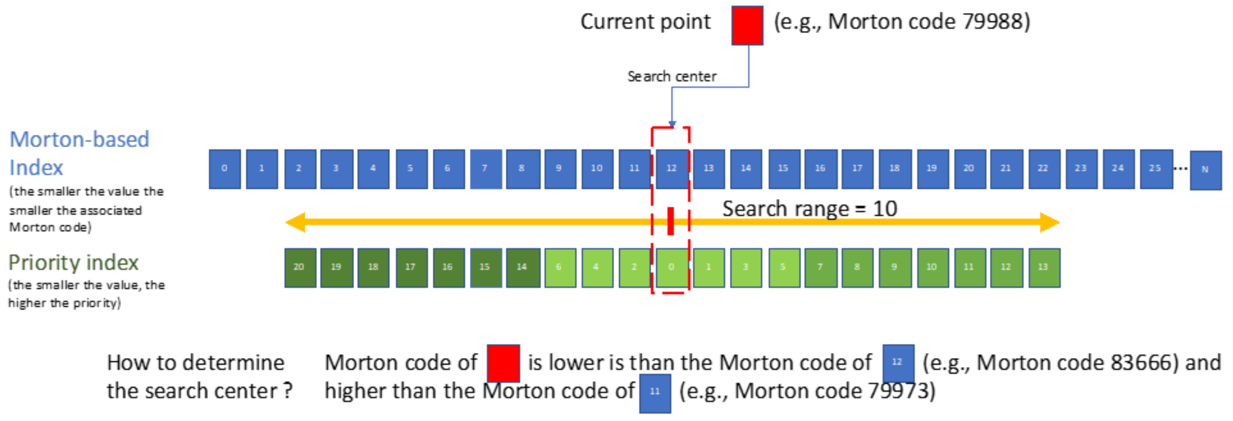

P0
P1
P4
P3
P7
P6
P9
P5
P8
P2
Original order
P0, P1, P2, P3, P4, P5,P6, P7, P8, P9
LOD-based order
P0, P5, P4, P2, P1, P6,P3, P9, P8, P7
LOD0
LOD1
LOD2

Level of detail generation

Intra LOD and Inter LOD prediction


P0
P1
P4
P3
P6
P7
P9
P5
P8
P2
Original order
P0, P1, P2, P3, P4, P5,P6, P7, P8, P9
LOD-based order
P0, P5, P4, P2, P1, P6, P3, P9, P8, P7
LOD0
LOD1
LOD2
allowed

If EnableRefferingSameLoD = 1 then allowed
Otherwise not allowed


Current point
(e.g., Morton code 79988)
Search center
Morton-based Index
(the smaller the value the smaller the associated Morton code)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 … N
Search range = 10
Priority index
(the smaller the value, the higher the priority)
20 19 18 17 16 15 14 6 4 2 0 1 3 5 7 8 9 10 11 12 13

How to determine the search center ? Morton code of 🟥 is lower is than the Morton code of 12 (e.g., Morton code 83666) and higher than the Morton code of 11 (e.g., Morton code 79973)

基于莫顿顺序查找最近邻

Adaptive predictor selection

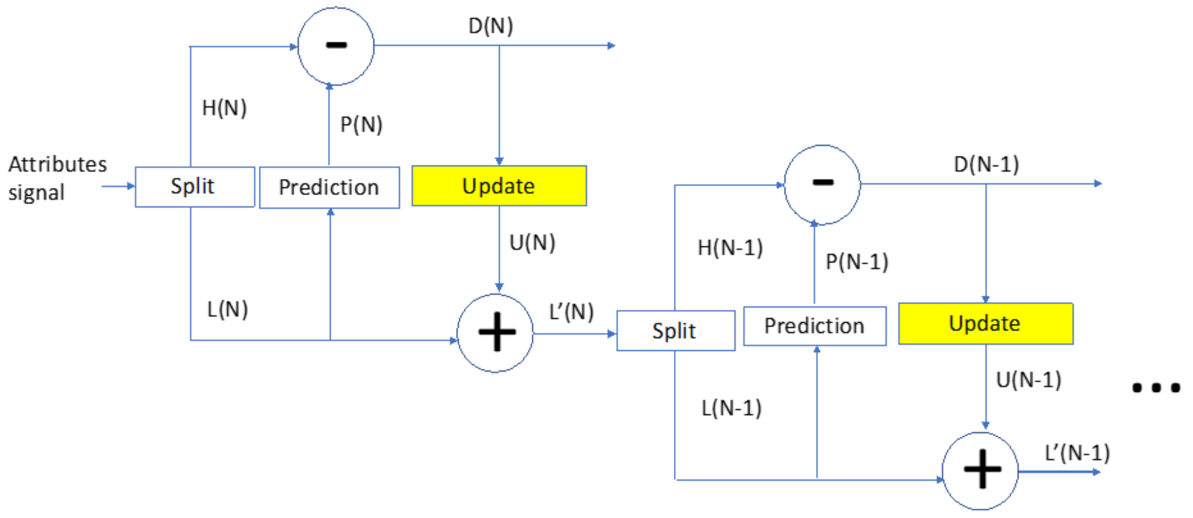
Predictor index	Predicted value
0	average
1	P4 (1st nearest point)
2	P5 (2nd nearest point)
3	P0 (3rd nearest point)

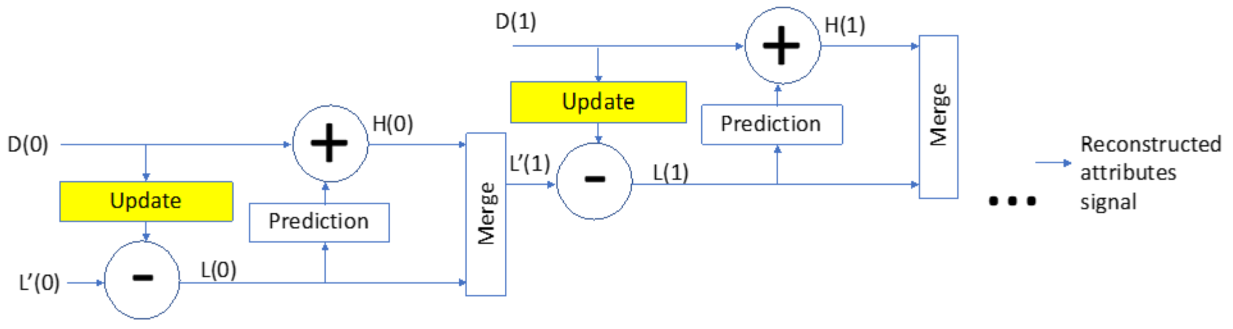
以编码P2为例，创建多个predictor candidate，根据RDO选择最优predictor，编码index

- Octree geometry encoding
  - DCM
  - Neighbour-Dependent Entropy Context
- Attribute coding
  - LOD
  - Lifting
  - RAHT
- Functionality
  - Tile and Slice

Lifting

只用来做有损属性编码，为了让低层LOD中的点更具影响力





$$w(\psi) = w(\psi) + \sum_{i}^{K} \zeta_i \times w(\phi_i).$$

ψ为低层LOD的点，ϕ_i为高层LOD中使用ψ作为预测点的点，ζ_i为两点距离

$$Att(\psi') = Att(\psi) + \frac{\sum_i^K \zeta_i \times w(\phi_i) \times Resi(i)}{\sum_i^K \zeta_i \times w(\phi_i)},$$

被用作预测点的次数越多w越大，计算预测残差时影响力越大

- Octree geometry encoding
  - DCM
  - Neighbour-Dependent Entropy Context
- Attribute coding
  - LOD
  - Lifting
  - RAHT
- Functionality
  - Tile and Slice

RAHT[1]

有损无损编码均可实现。
本质上是对属性值做空间变换。
令A_{n}表示属性值，T_n表示变换系数，初始化令$T_n = A_n$
按照由底向上的顺序，对于两个相邻的block, $(A_{01}, A_{02}, \dots, A_{0w_0})$和$(A_{11}, A_{12}, \dots, A_{1w_1})$分别是他们的点的属性值，$(T_{01}, T_{02}, \dots, T_{0w_0})$ 和 $(T_{11}, T_{12}, \dots, T_{1w_1})$ 分别表示变换系数（按莫顿顺序排列），对于parent block的变换系数$(T_1, T_2, \dots, T_{w_0+w_1})$，有：

$$\left(T_1, T_2, \dots, T_{w_0}, T_{w_0+1}, T_{w_0+2}, \dots, T_{w_0+w_1}\right) = \left(aT_{01} + bT_{11}, T_{02}, \dots, T_{0w_0}, -bT_{01} + aT_{11}, T_{12}, \dots, T_{1w_1}\right)$$

$a = \sqrt{\frac{w_0}{w_0+w_1}}$，$b = \sqrt{\frac{w_1}{w_0+w_1}}$ （w_0, w_1是点数）
每个block的第一个值相当于是DC系数，只有它经过了变换，其余保持不变

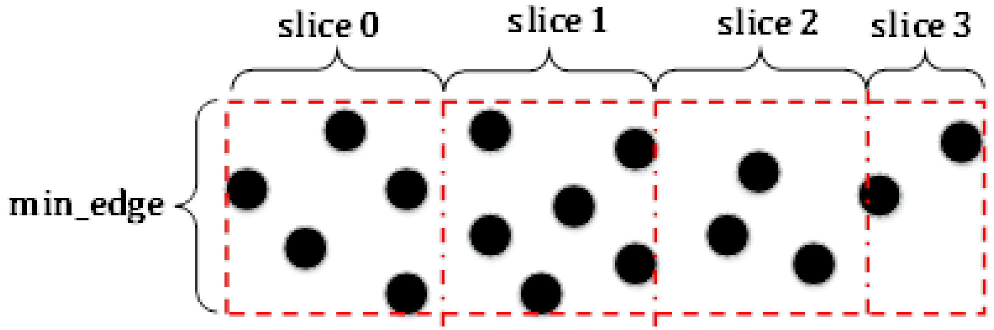
[1] R. L. de Queiroz and P. A. Chou, "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform," in IEEE Transactions on Image Processing, vol. 25, no. 8, pp. 3947-3956, Aug. 2016.

- Octree geometry encoding
  - DCM
  - Neighbour-Dependent Entropy Context
- Attribute coding
  - LOD
  - Lifting
  - RAHT
- Functionality
  - Tile and Slice

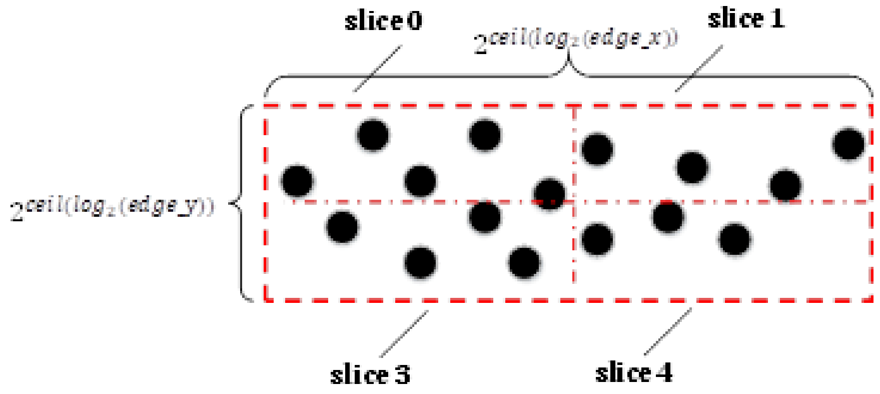
Tile and Slice

Slice是包含多个点的集合，Tile是包含一个或多个slice的bounding box。
两种slice划分方法:

- Uniform-geometry partition along the longest edge:
  划分maxEdge/minEdge个slice，根据点数进一步进行merge或split



- Uniform-Geometry partition using Octree
  将原始点云划分为$8^{depOctree}$个slice，根据点数进一步进行merge或split


slice 0
$2^{ceil(log_2(edge\_x))}$
slice 1
$2^{ceil(log_2(edge\_y))}$
slice 3
slice 4