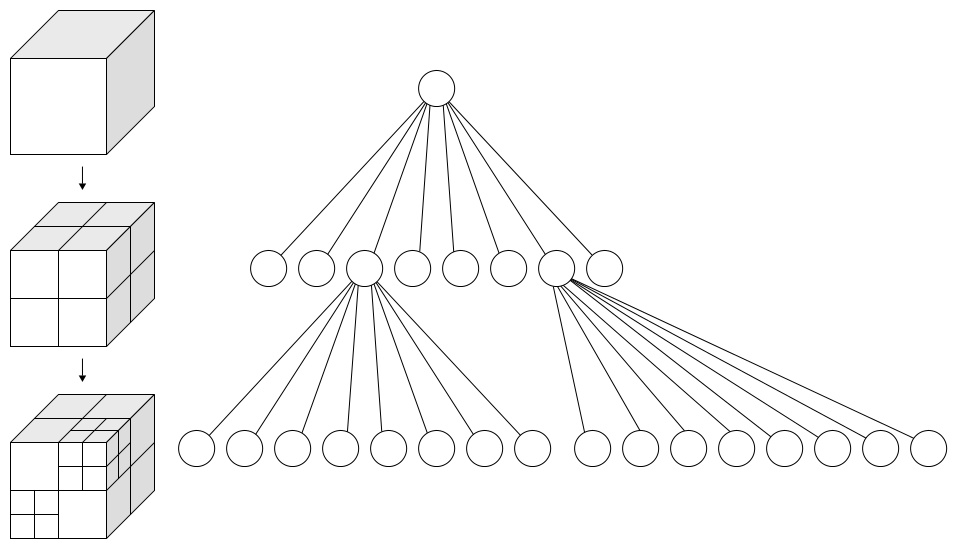
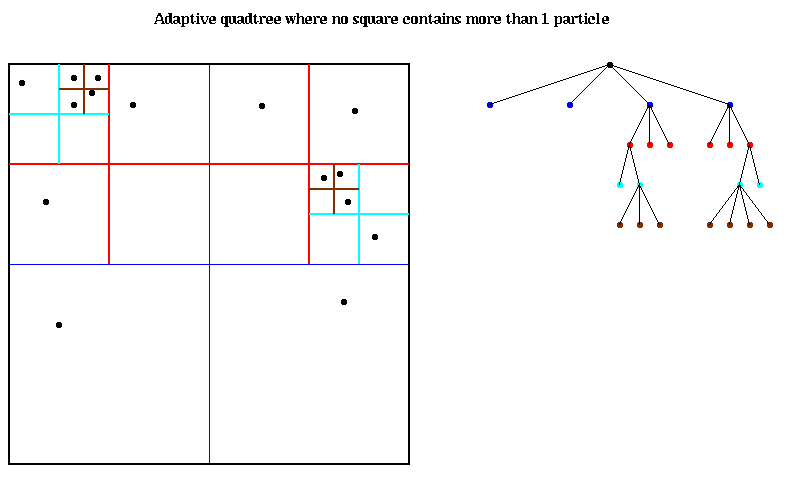
**八叉树点云压缩方法**

**一、八叉树结构**

八叉树（Octree）是一种用于描述三维空间的树状数据结构。八叉树的每个节点表示一个正方体的体积元素，每个节点有八个子节点，将八个子节点所表示的体积元素加在一起就等于父节点的体积。八叉树是四叉树在三维空间上的扩展，二维上我们有四个象限，而三维上，我们有8个卦限。简单来说，八叉树的空间划分方式是，把一个立方体分割为八个小立法体，然后递归地分割小立方体。八叉树主要用于空间划分和最近邻搜索，同样在点压缩方面也可以发挥重要的作用。



由于三维空间较难理解，对于树的生成先以四叉树作图示解释，把每点放到正方形空间里，若该正方形含有超过一个点，就把该正方形分割，直至每个小正方形（叶节点）仅含有一个点，就可以得出以下的分割结果：

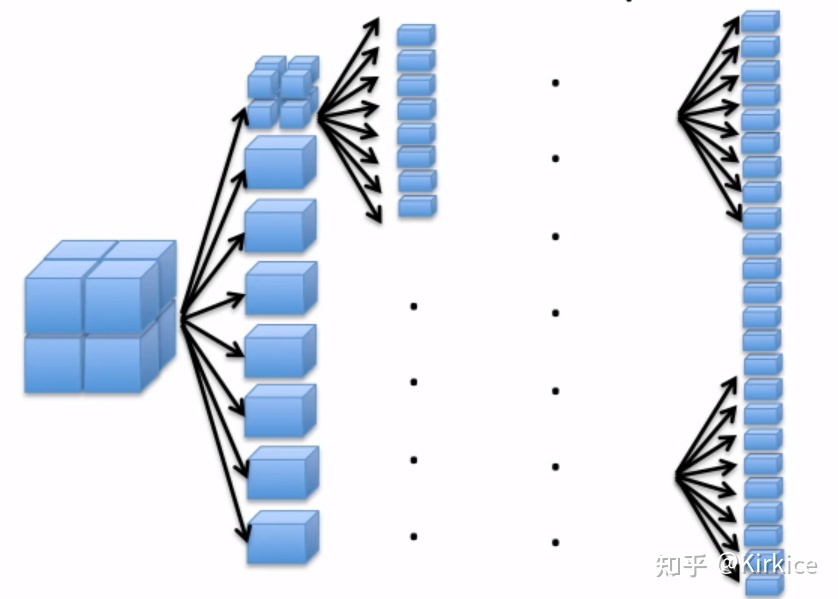


实现Octree的原理包括以下几个步骤：

## 将当前的立方体细分为八个子立方体。

## 如果任何一个子立方体内包含多个点，则将其进一步细分为八个子立方体。

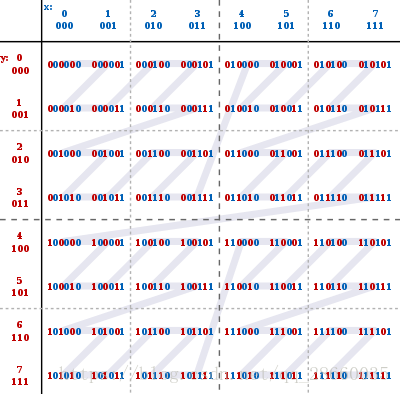
## 重复以上操作使得每个子立方体内包含最多一个点。



1. **利用莫顿码实现点云八叉树编码**
2. **莫顿码**

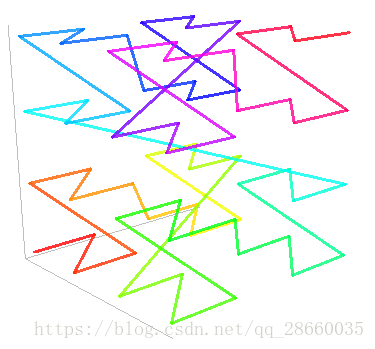
莫顿码可以将多维数据转化为一维数据编码，是一种比较常用的压缩编码方法。莫顿编码也叫z-order code，因为其编码顺序按照空间z序。

从二维莫顿编码的计算开始：



上图展示了8\*8图像每个像素的空间编码，从000000到111111，用一维二进制数，编码了x，y值在0-7的位置坐标。交错二进制坐标值，获得二进制z值图。沿着数值方向连接z型，产生递归的z形曲线。图中每个位置上就按连接顺序放上了z值。

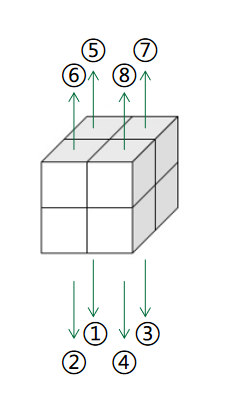
上升到3维情况，其递归过程为：



1. **点云八叉树位置编码**
2. 根据点云的三维坐标最大值计算要生成的八叉树的层级以及莫顿码的位数：

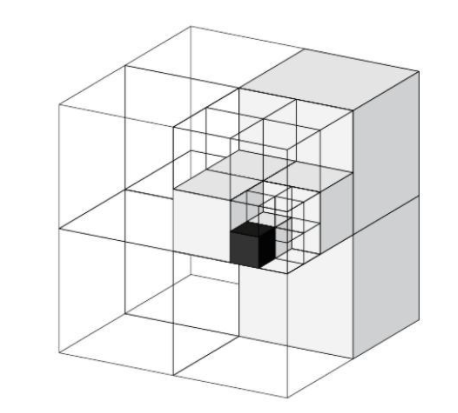
其中n表示保证在叶子节点里只包含一个点的情况下生成的八叉树的层级，N表示计算得到的点云三维坐标里的最大值。

1. 在每一个八叉树子块里用三位二进制码（zyx）来表示下一次划分形成的子块的空间位置，递归下去得到用莫顿码实现的三维数据的一维表示，以此来对点云数据进行压缩。



其中①子块的位置对应于二进制000，②子块的位置对应于二进制001，以此类推表达八叉树子块的空间位置。

1. 算术编码：根据上一步生成的二进制码流中不同字符出现的概率，用不同长度的编码来表示不同字符。进一步对二进制码流进行编码压缩。
2. **点云八叉树位置解码**
3. 算术解码：实现算术编码的逆过程得到原始的莫顿码。
4. 莫顿码解码：解码过程将叶子节点所在的八叉树子块的顶点近似作为实际点云中点的位置，如图黑点。



莫顿码中每三位表示八叉树的一层，每一位代表点的zyx坐标，解码时，对每一个叶子节点逐层逐坐标乘以得到近似的点云的实际坐标，其中n表示保证在叶子节点里只包含一个点的情况下生成的八叉树的层级。

1. **Matlab例程**
2. 八叉树生成与编码

function [Codes,Octree] = GenOctree(points)

tic

mcode = Morton(points);

Lmax = ceil((size(mcode,2)/3));

pointNum = size(mcode,1);

pointID = 1:pointNum;

nodeid = 0;

proot.nodeid = nodeid;

proot.childPoint={pointID};

proot.occupancyCode=[];

proot.parent=0;

Octree(1:Lmax+1) =struct('node',[],'level',0);

Octree(1).node=proot;

% Octree(1).nodeNum = 1;

for L=1:Lmax

Octree(L+1).level = L;

NodeTemp(1:min([pointNum,8^(L-1)])) = struct('nodeid',nan,'childPoint',{[]},'parent',0,'occupancyCode',[]);

nodeNum = 0;

for node = Octree(L).node

for ptid = node.childPoint

PId = ptid{:};

if isempty(PId)

continue

end

PId = pointID(PId);

nodeid=nodeid+1;

Node.nodeid = nodeid;

Node.childPoint=cell(1,8);

Node.parent=node.nodeid;

n = L-1;

mn = mcode(PId,1+n\*3:3+n\*3);

idn = bin2dec(mn)+1;

for i = 1:8

Node.childPoint(i)= {PId(idn==i)};

end

Node.occupancyCode = ismember(8:-1:1,idn);

nodeNum = nodeNum+1;

NodeTemp(nodeNum)=Node;

end

end

Octree(L+1).node= NodeTemp(1:nodeNum);

end

Octree(1)=[];

toc

Nodes = [Octree.node]';

Codes = bin2dec(num2str(cell2mat({Nodes.occupancyCode}')));

end

1. 八叉树解码

function ptRec = DeOctree(dtext)

% DeOctree

ptRec = DeOctree(dtext);

function points = DeOctree(Codes)

occupancyCode = flip(dec2bin(Codes)-'0',2);

codeL = size(occupancyCode,1);

N = ones(1,30);

codcal = 1;

L = 1;

while codcal+N(L)-1<=codeL

L = L+1;

N(L+1)= sum(occupancyCode(codcal:codcal+N(L)-1,:),'all');

codcal = codcal+N(L);

end

Lmax = L;

Octree(1:Lmax)=struct('node',[],'nodeNum',1);

proot = struct('pos',[0,0,0]);

Octree(1).node = proot;

codei = 1;

for L = 2:Lmax

Octree(L).nodeNum = N(L+1);

childId = 0;

childNode(1:Octree(L).nodeNum) = struct('pos',[0,0,0]);

for currentNode = Octree(L-1).node

code =occupancyCode(codei,:);

for bit = find(code)

childId = childId+1;

childNode(childId).pos = currentNode.pos+ bitshift((dec2bin(bit-1,3)-'0'),Lmax-L);

end

codei = codei+1;

end

Octree(L).node = childNode(1:Octree(L).nodeNum);

end

points = cell2mat({Octree(Lmax).node.pos}');

end

end