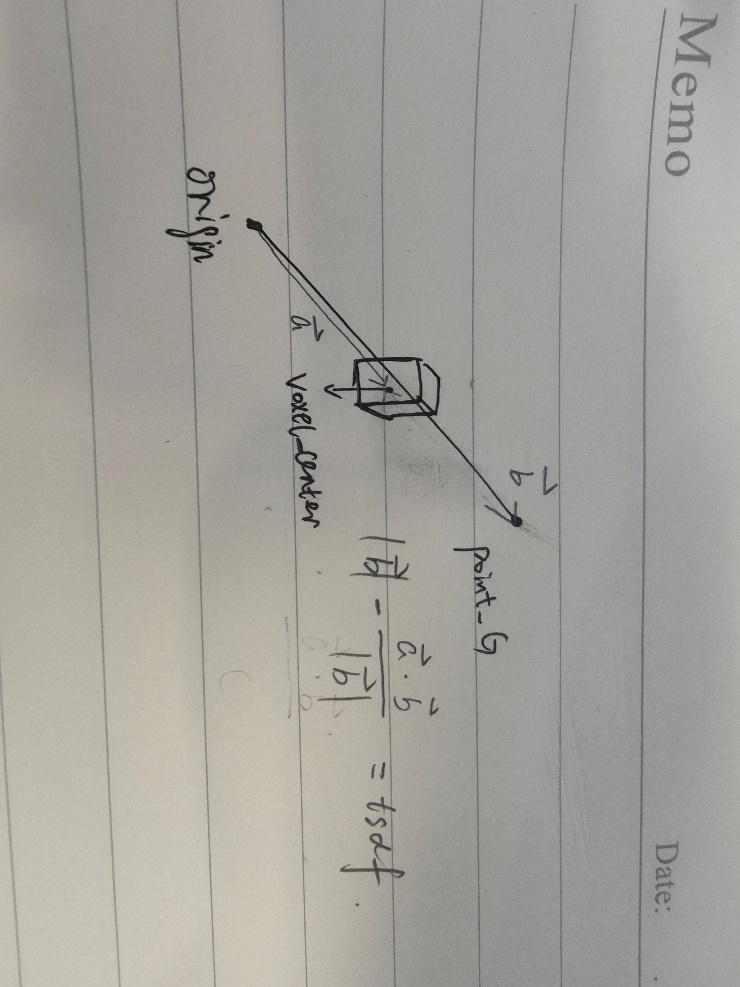
# Voxblox算法实现文档

参考博客：<https://www.guyuehome.com/15664>

## 一、computeDistance



origin是车辆的位置，point\_G是传感器的点在世界坐标系的坐标。计算得到的距离越小则表示距离障碍物越远

## 二、marching cube

参考博客：<https://www.bilibili.com/video/BV1Ev411r7vx/?spm_id_from=333.337.search-card.all.click&vd_source=984a70561ab2b42249f5eeff92b8d9f1>

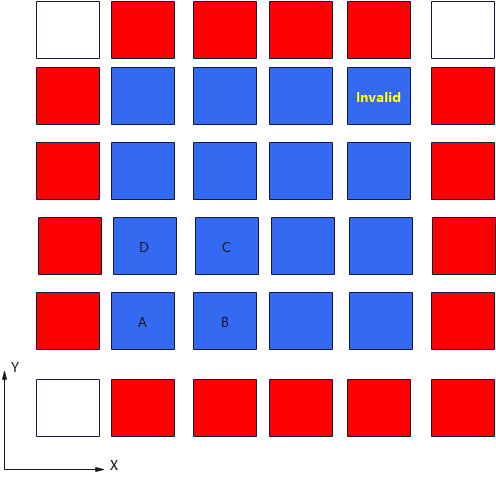
<https://paulbourke.net/geometry/polygonise/>

<https://www.youtube.com/watch?v=M3iI2l0ltbE&t=58s&ab_channel=SebastianLague>

## 三、extractBlockMesh

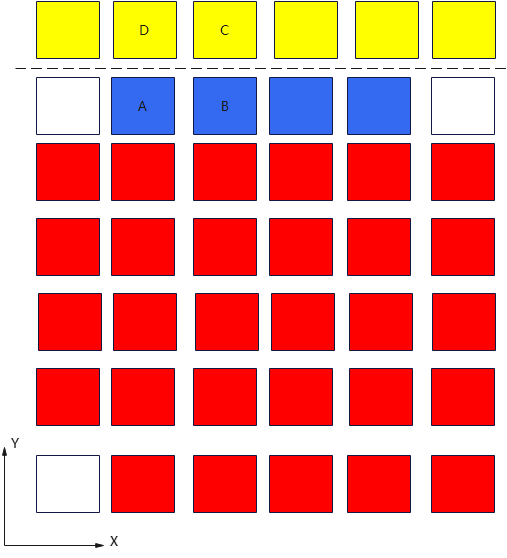
为了简化讲解，我们以平面的形式对extractBlockMesh函数进行讲解，下图绘制的是在一个block中的所有voxel，其中白色的voxel为没有被光线击中的voxel，其他颜色的voxel为都有有效的测量值。

1.首先遍历内层的voxel，如图中蓝色方框。遍历蓝色方框的voxel判断周围的所有voxel是否为有效的，发现右上角标注invalid的voxel是无效的，不参与marching cube的计算。

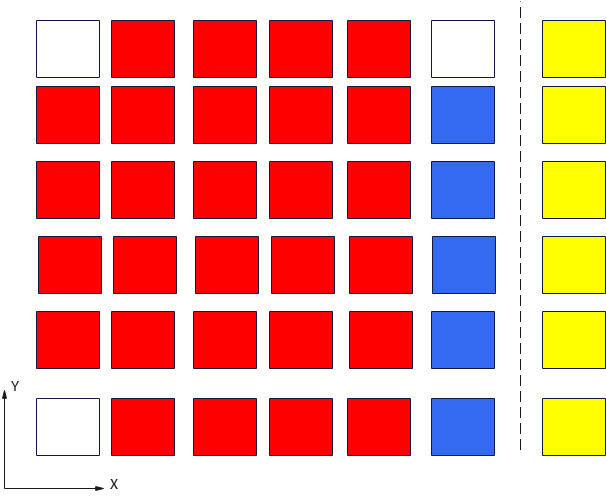


使用marching cube算法，计算每个voxel的切点。cube是按照xy方向进行构建的，如上图。如果遍历到A voxel后，则使用ABCD四个voxel构建cube。

2.然后遍历外层的voxel，先固定y的最大值，然后遍历x，如下图蓝色方框标注的voxel会参与。在遍历block的边缘voxel时会依赖相邻block里面的voxel，如下图中黄色voxel。当计算A voxel的marching cube结果时，会用到ABCD这四个voxel，首先会判断DC的block是否存在，如果存在再计算DC在对应block中的voxel坐标，最后再判断DC两个voxel是否被击中。只有ABCD四个voxel都被击中才会进入marching cube算法计算顶点坐标。如下图所示：



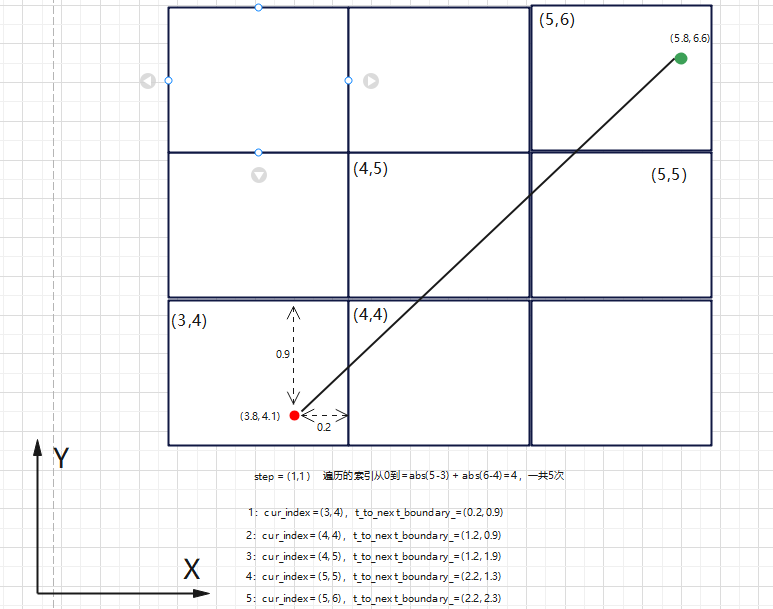
再固定x的最大值，然后遍历y，如下图蓝色方框标注的voxel会参与：



如果是三维的话，会固定z的最大值，然后遍历x和y。

## 四、构建ray\_caster

已知光线起点坐标和光线结束终点坐标。需要计算光线通过的所有voxel



光线起点为红色点(坐标为3.8, 4.1)，结束点为绿色点(坐标为5.8, 6.6)。第一次遍历时当前voxel索引为(3,4)，t\_to\_next\_boundary初值等于(0.2,0.9)，如图所示。

第二次遍历时，因为(0.2,0.9)中X坐标值小于Y轴坐标，因此下一个voxel index X轴就要加1，从(3,4)变为了(4,4)，依次类推。那么遍历的总次数如上图所示，总共可以遍历5次。