基于体素化与降采样的三角形网格 LOD 模型生成算法

华南理工大学 练孙鸿

摘要 在计算机图形学中，三维物体常常以三角形网格的形式表示。在不同的情景下渲染不同精细度的模型可以提高渲染效率。细节层次（LOD，Level of Detail）模型就是一系列不同精细度的模型。本文提出一种闭合网格模型的 LOD 模型生成算法，该算法先使用本文提出的方法对给定网格模型进行体素化（Voxelization），得到指定分辨率的体素（Voxel）模型中间表示。然后对体素模型进行一定分辨率的降采样与滤波之后用 Marching Cubes 算法来重构出不同细节程度的三角形模型。本文算法有如下特点：可以通过调节体素化分辨率与降采样分辨率，在 LOD 模型生成质量与生成速度之间进行权衡；体素化只需要在生成一系列 LOD 模型之前执行一次，之后生成 n 级 LOD 模型只需要耗费 n 次重构网格的时间；生成的三角形网格比较均匀；能在其他场景应用，例如 CT 切层扫描数据、3D 扫描数据的网格 LOD 模型生成、近似的网格布尔运算。

关键词：LOD 体素化 降采样 网格重构 MarchingCube 网格简化

Abstract In computer graphics, 3-dimensional objects are often represented as triangular mesh. Multiple versions of models with different fineness are required under different circumstances, in order to accelerate rendering. Level of Detail (LoD) models are a series of approximated models with different amount of details retained. In this paper, a scheme to

generate a series of LOD models for a single enclosed mesh model is proposed. This algorithm proposes a sub-algorithm for mesh voxelization first, with which we could yield voxel model, a kind of intermediate representation. Then down-sampling and Marching Cubes based mesh reconstruction are applied to the voxel representation to generate a final triangulated mesh. The LOD-generation scheme in this paper has the follow features: a trade off can be made between the quality of simplified mesh and the speed of computation by adjusting the resolution of voxelization and down-sampling; Each approximated mesh in a LOD series are acquired at a price of time for mesh reconstruction, while voxelization only needs to be executed once before the LoD series generation; the size of the triangles are uniformly distributed; modified version of this algorithm can be applied to other situations like surface extraction of CT/MRI scan data and 3D scan point cloud data, or approximated mesh boolean operations.

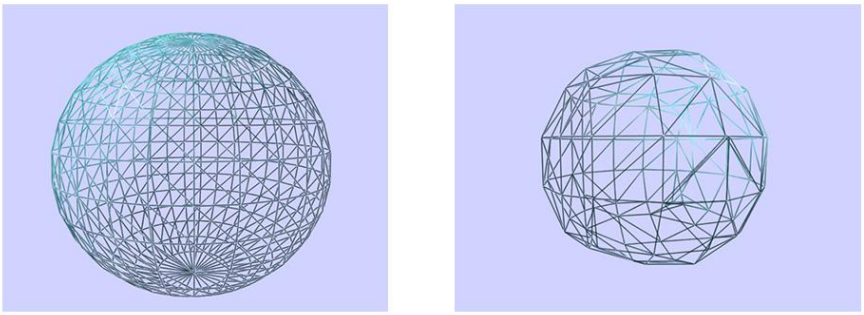
Keywords LOD, Voxelization, down-sampling, mesh reconstruction, Marching Cubes algorithm, mesh simplification

1 引言

在游戏、科学可视化等计算机图形学的应用中，渲染效率与渲染质量是都是要兼顾的。当前大部分的计算机性能还不不能以较快的速度渲染海量三角形。经过精心制作的模型，或者 3d 扫描仪扫描并重构得到的模型一般都有几十万、几百万甚至更多的三角形。

但不是所有时候都需要用最高精度的模型。例如，同样的模型处在远处时成像会比较小，从而很多细节就被忽略了，这时候高精度的模型就会浪费计算性能。这时候如果在远处的能用更低精度的模型渲染，

那么在不太影响渲染质量的情况下可以提高效率。这就需要用网格简化算法来生成更低细节层次（Level Of Detail，LOD）的模型，于是可以在不同情况下选择不同细节层次的模型来渲染。所以网格简化算法与 LOD 模型在科学研究与工业界中均有很大用处。



(a)896个面的球体模型　　(b)经过简化的120面球体

图 1：同一个模型不同细节层次的网格，(b)由本文算法生成

网格简化算法在国内外已有不少研究成果。网格简化有很多种思路与方法。Rossignac[1]提出用顶点聚类消除多余顶点，这是一种比较简洁快速的算法，但效果不一定理想。周昆[2]用八叉树自适应划分对此算法进行改进，Kanaya[5]也给出了保持拓补(topology-preserving)的改进。Maria-Elena[3]提出在一种使用边坍缩来简化网格的方法。当然还有不少网格简化的改进与变体，但是大部分算法的思路在[4]中有了精炼的概括：顶点聚类；合并共面三角形；受控的顶点、边、三角形删除(decimation)；基于能量函数的优化。

本文提出一种新的思路：对网格进行**体素化(Voxelization)**，然后进行重采样（一般是降采样），然后用**等值面提取算法（特指 Marching Cubes[8]算法）重构出三角形网格**。考虑到 Marching Cube 算法的特点，该算法需要实心的、经过填充的体素模型，所以这种思路会更适合闭合的三维网格模型，因为本文提出的体素化算法只能用在**闭合网格模型**上。

本文先提出一种基于切层与扫描线填充的闭合模型的**表面与内部体素化**方法，由这个体素化方法得到指定分辨率的、经过内部填充的体素模型。关于体素化算法的工作，[17]是比较早的工作，从“像素化”直

线开始推广到“体素化”面。[7]中给出了基于八叉树体素化（相当于光栅化的三维推广）每个三角形方法，进而实现体素化整个网格模型的功能。[6]先从体素化三角形开始，再用种子填充算法填充模型内部体素，从而得到实心的体素模型。[10]中提出与本文体素化算法类似的思路，先用一系列切片去切割模型，而且用了 GPU 加速与硬件自带的光栅化功能去实现体素化，但是这样的原理并不能直接填充模型内部。

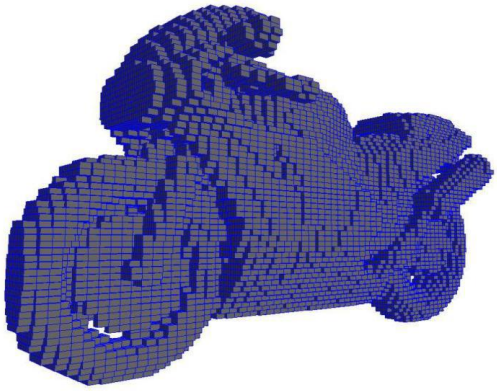


图 2：体素模型，用大量的体素(voxel)去逼近物体

2 算法描述

2.1 本文算法的一些约定

规定 0 本文的所有几何描述都在三维笛卡尔坐标系（左手系）下，其中 y 轴垂直于水平面，xz 平面水平。

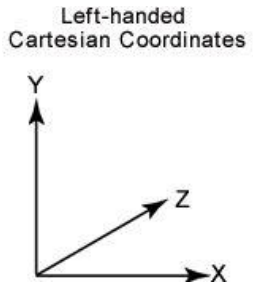


图 3：本文约定使用的坐标系

规定 1 顶点表示为实数的三元组 $\mathrm{V}(x,y,z),\text{其中}x,y,z\in R$

规定 2 线段表示为实数的三元组 $\mathrm{L}(V_1,V_2)$

规定 3 三角形表示为顶点的三元组 $T(V_1, V_2, V_3)$

规定 4 三角形表示为顶点集与三角形集的二元组 $Mesh(VSet, TSet)$

$VSet = V_i, i = 0,1...VertexCount - 1$ ，

$TSet = Ti(V_a, V_b, V_c), i = 0,1...TriangleCo\ unt - 1, V_a, V_b, V_c \in VSet$

其中 VertexCount 为网格的顶点个数，TriangleCount 为网格的三角形个数。网格的所有顶点坐标均为**模型空间坐标。**

规定 5 $MinX(obj)$ 为几何对象 obj（线段、三角形、三角形网格）可以取到的最小 x 坐标。

规定 6 $MaxX(obj)$ 为几何对象 obj（线段、三角形、三角形网格）可以取到的最大 x 坐标。

规定 7 与规定 5，6 类似，同理定义 $MinY(obj)$ ， $MaxY(obj)$ ， $MinZ(obj)$ ， $MaxZ(obj)$

规定 8 在规定 5，6，7 的基础上，定义:

$RangeX(obj) = MaxX(obj) - MinX(obj)$

$RangeY(obj) = MaxY(obj) - MinY(obj)$

$RangeZ(obj) = MaxZ(obj) - MinZ(obj)$

规定 9 设经过网格体素化得到的体素模型的在体素空间的 x,y,z 方向上分辨率分别为 $\Re_x, \Re_y, \Re_z$ ，即生成的体素模型会有 $\Re_x \times \Re_y \times \Re_z$ 个体素。

规定 10 线性插值函数： $Lerp(a,b,t) = a + (b-a) \times t$ 。一般情况下 a,b 指的是实数。若 a,b 指的是同样维度的向量时，两个向量的按分量逐个进行线性插值。

2.2 算法流程与应用场景

本文算法的大致步骤如图 4 所示，大体上有三个阶段：

- 原始的闭合网格模型
- 经体素化后的体素化模型
- 经过体素模型重采样、网格重构后的简化/细分网格模型（LOD）

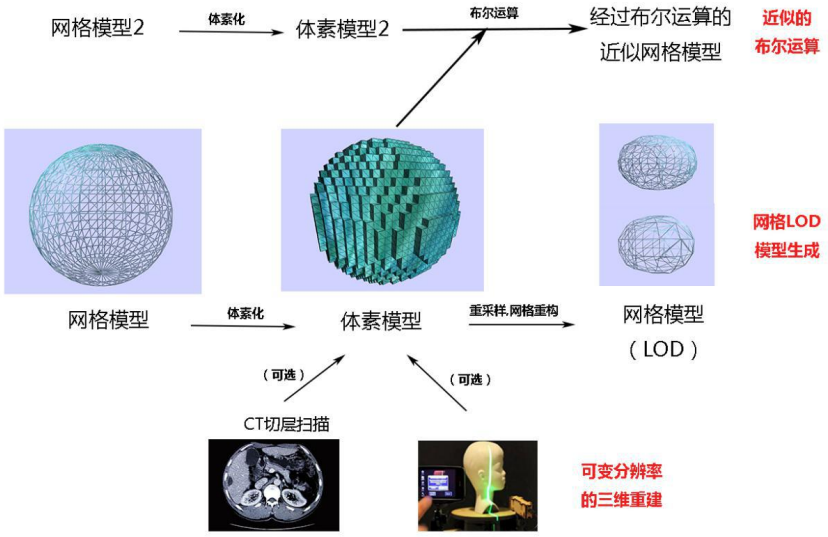


图 4：本算法的大致流程以及其他应用场景

有一点是值得注意的，本文算法的应用场景不只是能用于闭合网格模型的简化，还可以有其他方面的应用，例如：

- CT/MRI 切层扫描数据的曲面重建：CT/MRI 等医学扫描数据可以很直观地转化为体素模型（处在扫描切层上的像素可以直接转换为体素），继而可以用本算法的“重采样、网格重构”直接生成网格 LOD 模型。
- 3D 扫描数据：一般的 3D 扫描得到的原始数据就是点云。点云经过处理（例如用体素化的三维栅格来滤波）以后，也是比较容易转化成体素模型。因为扫描数据一般比较密集，要可视化扫描到的物体一般都要求进行简化。所以用本算法的“重采样-网格重构”可以一步到位生成简化的网格，而不是先重构再简化。
- 曲面细分：虽然体素模型重采样阶段一般做得都是降采样，但是重构之后的三角形却有可能比原

来的多，所以此算法有潜力实现真正的曲面细分。但是问题在于，原网格模型是原始数据中最精确的表示了，不断的降采样只会减少信息。本算法生成的新网格即使面数更多，其表面曲率信息也是不经过插值的，不比原模型的表面信息多。所以算法虽然可以生成更密集更均匀的网格模型，但还不能做到平滑的曲面细分。

- 近似的网格布尔运算：三维网格模型的布尔运算也是一个很复杂的问题，网格模型直接进行并、交、差等问题必定涉及大量的几何分析与特殊情况的处理。文献[11][18]中就直接用精确的几何求交然后进行三角化相交边缘来得到进行了布尔运算的模型。而本文算法经稍加修改，即可实现一个网格的近似布尔运算的方法。体素模型的布尔运算（并、交、差）非常的直观，而且这些运算都可以用 c++的计算位运算符实现，快速且方便。但是这种方法原理上比直接用几何求交的算法更粗糙一点，只能得到近似的结果，但是体素模型布尔运算再重构网格直观上讲就比较的健壮。基于体素化表示进行网格布尔运算的重点在于如何抽取模型表面，这一点是有相关研究的[8][19][20]。

限于主题，本文接下来的部分只介绍基于体素化与重采样的**网格模型简化**算法。

2.3 体素化

2.3.1 体素化算法与相关术语规定

体素（Voxel）[9]，体积元素（Volume Pixel）,是数字数据在三维空间上分割最小单位。在本文里体素含义类似，特指逻辑上不可以分割的正方体单元，但其尺寸是我们不关心的问题。如何把一个三角形网格模型转化为由体素逼近表示的模型，就是体素化（Voxelization）要讨论的问题。为了节省内存，**本文规定体素是二值化的，即一个体素只能取 0 或者 1 两个值。**

本文的体素化算法分成两个步骤：

- 切层，即用平行于 XZ 平面的平面移动切割原模型
- 光栅化，基于上一步得到的结果进行光栅化与内部填充。

把一系列都经过光栅化与填充之后的切层组合起来就得到了体素模型，每个体素都可以用体素空间的坐标 (i, j, k) 唯一地标识。

2.3.2 切层

本文提出的体素化算法借鉴了“3D 打印”的思想。3D 打印利用很多层有一定厚度的材料来逼近目标模型。在这里，有一定厚度的 3D 打印的材料线就可以抽象为一系列体素的集合。

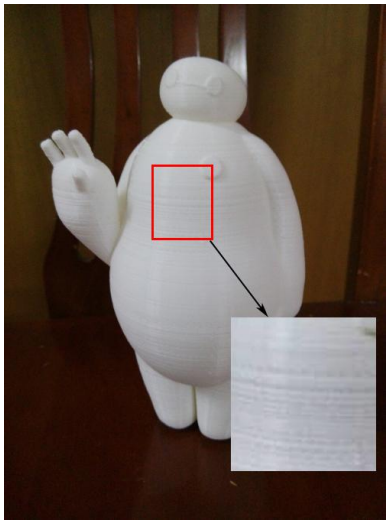


图 5：3D 打印模型，局部放大可以明显看到其分层叠加的树脂

所以体素化算法的第一步可以描述为：

问题 1 给定闭合曲面 $\mathrm{F}(x, y, z) = 0$ ，其 y 坐标取值范围为 $[y_{\min}, y_{\max}]$。用 $\Re_y$ 个平面 $y = y_j$，（其中 $y \in (y_{\min}, y_{\max})$，$j = 0,1 \ldots \Re_y - 1$）来与曲面 $\mathrm{F}(x, y, z) = 0$ 相交，求这 j 次相交得到的隐式曲线方程集合 $\mathrm{F}_j(x, y_j, z) = 0$。

换句话讲，我们分别用 $\Re_y$ 个水平面去切割模型，每一次切割就是一次平面与闭合曲面求交的过程。显而易见的是（但证明起来有难度），第 j 次切割得到的结果会是**一条或多条封闭曲线**（由隐式曲线方程 $\mathrm{F}_j(x, y_j, z) = 0$ 来表示）。

闭合的三角形网格是分片光滑的曲面，也可以用 $\mathrm{F}(x, y, z) = 0$ 来表示，而且每个分片都是三角形，是一种用计算机去逼近描述曲面的不错的形式。所以问题 1 的求解可以采用分而治之的思路解决，即只需分别考虑每个三角形与平面的相交情况，然后再把结果综合起来。

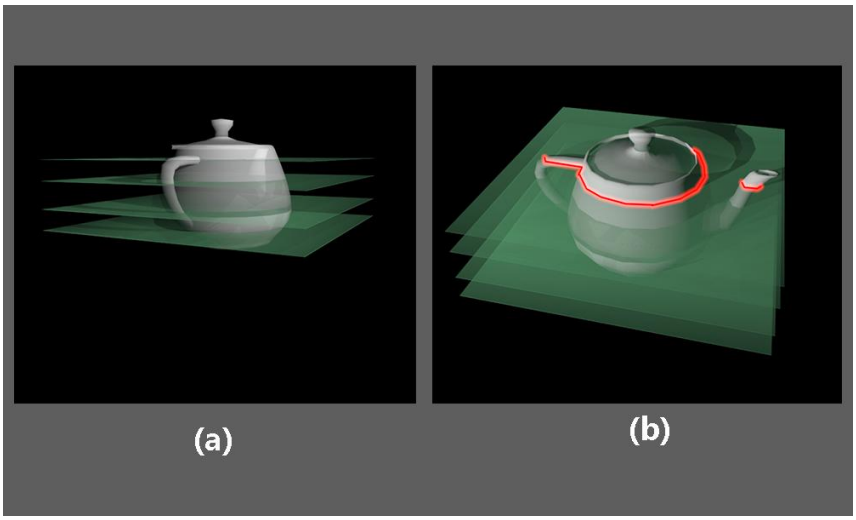


图 6：用多个平面去切割三角形网格，(b)中红色标记的线为切割得到的线段集合

设单个三角形 T 与平面 $y = y_j$ 的一次求交函数为：

$$Segment(T, y) = \begin{cases} \text{相交线段的集合}, \mathrm{y} \in [MinY(\mathrm{T}), \mathrm{Max}Y(T)] \\ \phi, y \notin (MinY(T), MaxY(T)) \end{cases} \tag{2-1}$$

显然函数 Segment(T,y)的返回值要么是只有一个元素的线段集合（水平面与三角形 T 相交线段），要么就是空集（水平面与三角形 T 不相交）。那么问题 1 中描述的曲面可以限制为三角形网格 $Mesh$ 以后，问题可以转换为如下描述：

问题 2 给定三角形网格 $Mesh(VSet, TSet)$，用多个相等间距的水平面

$Pi: y = MinY(mesh) + RangeY(mesh) \times \frac{i}{\Re_x}$ 与 $Mesh(VSet, TSet)$ 求交，求所有相交得到线段的集合 $LSet$，其中

$$LSet = \bigcup_{i=0}^{(\Re_y - 1)} \bigcup_{j=0}^{(TriangleCount-1)} Segment(T_j, y_i) \tag{2-2}$$

问题 2 求解出来理论上会得到多层的线段集合，每层的线段集合在可视化出来也应该是一个或多个封闭图形的轮廓。这样的问题描述就意味着求解 $LSet$ 时写的程序必须有两重循环，而且要把所有三角形与切割面都配对求交，时间复杂度为 $O(\Re_y \times TriangleCount)$。考虑到一般情况下大部分三角形 T 的 y 坐标范围 $RangeY(T)$都远小于网格模型的 y 坐标范围 $RangeY(Mesh)$，所以 $LSet$ 可以减少遍历的切割面 y 坐标范围，交换循环的嵌套顺序，把不可能与三角形 T 相交的切割面预先排除掉。设比 y 坐标 y_0 小的最近切割面标号（按 y 坐标从小到大给切割面标号）为 $P_{near}(y)$，有

$$P_{near}(y) = \left\lfloor \Re_y \times \frac{y - MinY(mesh)}{RangeY(mesh)} \right\rfloor \tag{2-3}$$

则 $LSet$ 可以优化表示为：

$$LSet = \bigcup_{i=0}^{(TriangleCount-1)} \bigcup_{j=P_{near}(MinY(T_i))}^{P_{near}(MaxY(T_i))} Segment(T_i, y_j) \tag{2-4}$$

实验证明改变循环嵌套顺序并减少相交次数的优化之后，算法运行时间明显减少。现在只需要来讨论 $Segment(T, y)$ 的具体实现，即求三角形与平面相交所得线段。

其实平面与平面相交实现起来还是一件比较繁琐的事情。幸运的是，这里的相交算法规定了无限大的切割面是平行于 XZ 平面的，多了一个这样的限制条件将会使得推导更加容易。下面我们进行分类讨论 $Segment(T, y)$：

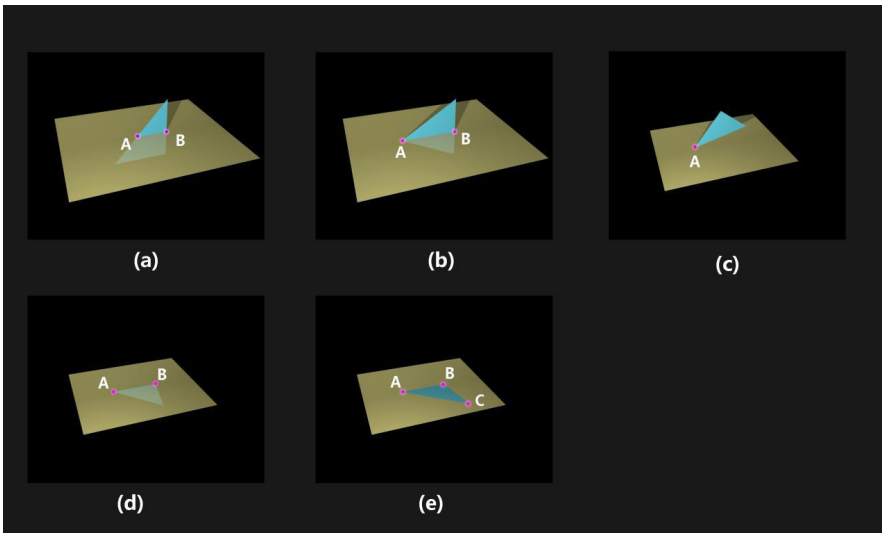


图 7：单个三角形与水平面相交的几种情况，按三角形恰好在平面上的顶点数分类

看到图 7 分了五种相交情况，按照“恰好落在切层上的三角形顶点数”来分类，先排除了平凡的不相交情况。设在线段 $L(V_1,V_2)$ 上的、给定 y 坐标的点的坐标可以由函数求得：

$$PlaneXSegment(L(V_1(x_1,y_1,z_1),V_2(x_2,y_2,z_2)),y)=Lerp(V_1,V_2,\frac{y-y_1}{y_2-y_1}) \tag{2-5}$$

其中，$y\in(y_1,y_2)$，注意这是开区间，特意把线段的两个端点排除。这是给了下一小节扫描线光栅化的方便，统一三角形与切层相交得到的线段端点个数的奇偶性。

设切层的 y 坐标为 y_{slice}，下面给出三角形与切层相交线段的求解方法：

(a)没有顶点落在切层上。设三角形与切割面相交的两条边为 $L_1(V_1,V_2)$ 和 $L_2(V_1,V_3)$，则相交线段为 $AB:L(PlaneXSegment(L_1,y_{slice}),PlaneXSegment(L_2,y_{slice}))$

(b)恰有一个顶点落在切层上，顶点所对的三角形边与切层相交。设三角形与切割面相交的那一条边为 $L_1(V_2,V_3)$，落在切层上的三角形顶点为 V_1 则相交线段为 $AB:L(V_1,PlaneXSegment(L_1,y_{slice}))$

(c)恰有一个顶点落在切层上，顶点所对的三角形边与切层不相交。则这种情况不生成相交线段。

(d)恰有两个顶点落在切层上。设落在切层上的两个顶点为V_1, V_2，则相交线段为 $AB: L(V_1, V_2)$

(e)所有顶点都落在切层上。这种情况其实可以不产生任何线段，因为体素化算法的基础要求就是曲面闭合，这就意味着必定有其他三角形的某些边与这个三角形的三条边共边。

综上，三角形与切割平面 $y = y_{slice}$ 的相交线段的求解函数 $Segment(T(V_1, V_2, V_3), y)$ 由下表给出：

线段集合	条件
$L(PlaneXSegment(L_1, y), PlaneXSegment(L_2, y))$	情况(a)
$L(V_1, PlaneXSegment(L_1, y))$	情况(b)
ϕ	情况(c)
$L(V_1, V_2)$	情况(d)
ϕ	情况(e)
ϕ	$y \notin (MinY(T), MaxY(T))$

表 2-1：三角形与切面求交函数 Segment(T,y)的实现细节

至此，“切层”得以实现，我们现在就有了由线段组成的模型切片外轮廓了。

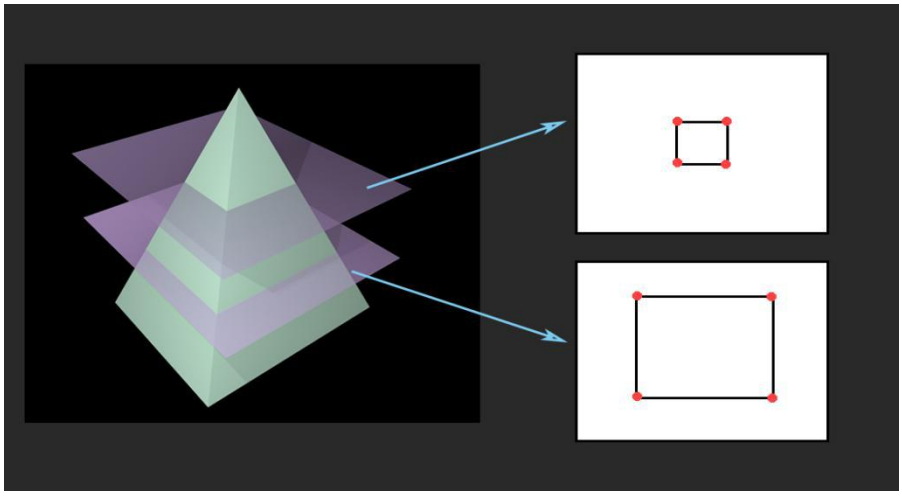


图 8：切层算法的输出，每个切片都有一个切割轮廓，由一系列线段组成

2.3.3 扫描线光栅化

经过切层的处理之后，原始网格模型的体素化问题就被转化成了“二维矢量多边形的光栅化及二值化填充”问题。本阶段的目标是把 $\Re_y$ 个切层转换为 $\Re_y$ 张分辨率为 $\Re_x \times \Re_z$ 二值化位图。

[12]光栅化(Rasterization) ,其实就是把矢量图形转换为像素点的过程。并且如果要填充多边形的内部，那就要求使用多边形填充算法。一般的多边形填充算法有基于扫描线[13]的算法，有基于种子填充的算法[14]，还有两者结合的算法[15]。很多文献都介绍了基于这些基本填充方法的改进算法。本文的扫描线光栅化填充算法在原理上没有大的突破，但是**做了一点修改用于解决扫描线填充的一种二义性情况。**

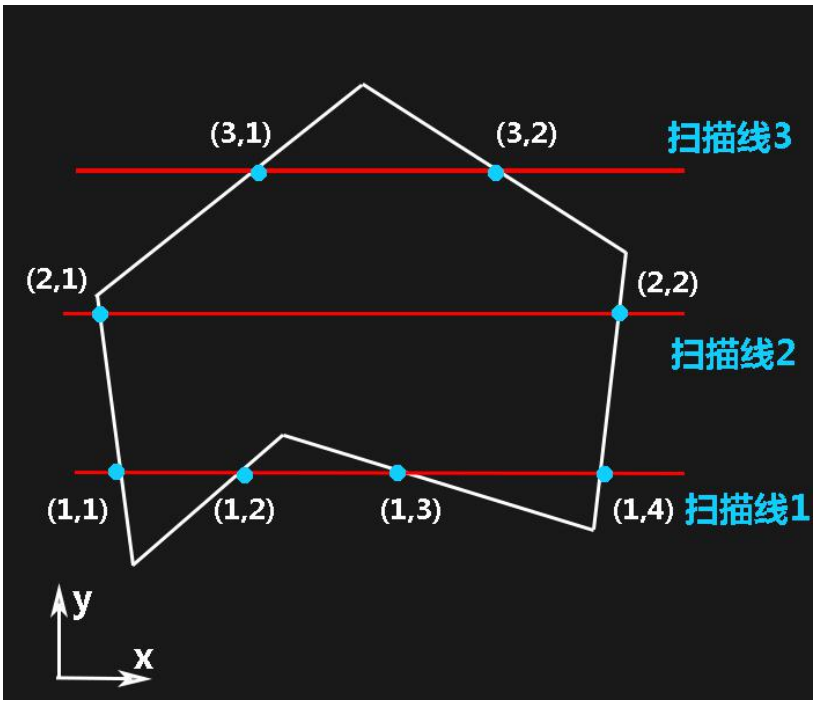


图 9：扫描线填充的基本思路，二元组 (i, j) 表示第 i 条扫描线上的第 j 的交点

在这里我们把目光就关注在单个二维切面上，坐标系在光栅化阶段也局部地切换为 x-y 笛卡尔直角坐标系，如图 9 所示。切面上的 x 轴对面三维坐标系的 x 轴，y 轴对应的就是三维坐标系的 z 轴。

作为一种常用的光栅化以及多边形填充的方法，扫描线填充法思路还是比较的直观。下面是几点基本设定：

- 每条扫描线就对应着光栅化图像的一行像素。
- 因为此处光栅化与填充是为了给定分辨率的体素化而服务，所以本阶段一共会有 $\Re_z$ 条平行于 x 轴的扫描线。
- 扫描线可能会与多边形有会多个交点，同一条扫描线与多边形相交得到的交点按 x 坐标从小到大编号 1~n。

一般情况（扫描线没有恰与多边形的端点相交时）下，扫描线与多边形的交点会有偶数个，因为扫描线穿过闭合多边形必定有“一进一出”。按照图 9 的标号方法，设 y 坐标从小到大，等距步进的第 i 条扫描线与多边形的第 j 个交点的 x 坐标为为 $X_{scan}(i,j)$，则当前行的填充区间应为：

$$[X_{scan}(i,2k-1), X_{scan}(i,2k)], k \in Z^+ \tag{2-6}$$

也就是每条扫描线上交点标号为[奇数，偶数]的 x 坐标区间表示多边形内部，是要被填充的区域。要注意的一点是，三维模型的切层阶段得到的是一系列的**无序的线段**，多边形的拓补关系并没有被计算。实际上也不需要计算，因为我们只需要求得 $\Re_z$ 条扫描线与多边形的边的交点的集合 $XPSet$，然后**每条扫描线上的交点按 x 坐标进行排序。**设在二维切面上，y 坐标比 y_{scan} 小的最近一条扫描线的 y 坐标为：

$$SL_{near}(y_{scan}) = \left\lfloor \Re_z \times \frac{y_{scan} - MinZ(mesh)}{RangeZ(mesh)} \right\rfloor \tag{2-7}$$

则一个切面上，扫描线与线段集的所有交点的集合为：

$$XPSet = \bigcup_{i=0}^{(\Re_z-1)} \bigcup_{j=SL_{near}(MinY(L_i))}^{SL_{near}(MaxY(L_i))} ScanlineXP(L_i, y_j) \tag{2-8}$$

其中函数 $ScanlineXP(L, y_j)$ 返回值为线段 L 与第 j 条扫描线 $y = y_j$ 的交点集合，用。而平行于 x 轴的扫描线与线段的交点坐标还是很好求的，所以有：

$$ScanlineXP(L(V_1(x_1, y_1), V_2(x_2, y_2)), y) = \begin{cases} (\frac{y - y_1}{y_2 - y_1} \times (x_2 - x_1), y), & 当 y_2 \neq y_1 且 \frac{y - y_1}{y_2 - y_1} \in (0,1) \\ \phi, 其他情况 \end{cases} \quad (2\text{-}9)$$

注意到函数 $ScanlineXP(L, y)$ 的第一个成立条件里的刻意用了开区间来排除线段 L 的端点，这是因为要如果线段端点在扫描线上，填充会产生二义性。

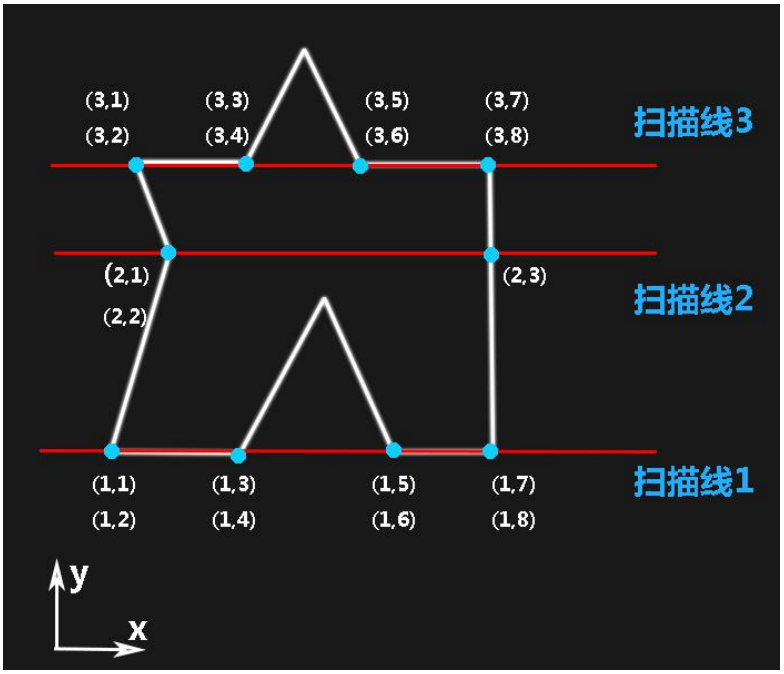


图 10：扫描线填充的二义性情况。二元组 (i, j) 表示第 i 条扫描线上的第 j 个的交点

图 10 说明了这一种二义性的根源。扫描线沿着 y 方向移动并与多边形求交的过程中，有可能会刚好落在多边形的一个或多个端点上。这会使得同一个位置上有不止一个交点。问题就在于同一位置不止一个交点的话，式子(2-5)的填充区间就不对了，奇偶性会有改变。图 10 的扫描线 2 上(2,1)与(2,2)就是从两条相邻线段上产生的位置重合的交点，这会导致(2,2)到(2,3)的区域不被填充，直接空了一行

像素。图 10 的扫描线 1、3 更是表示一种难以解决二义性的情况，就是线段的两个端点都在扫描线上。(1,4)与(1,5)之间，(3,4)与(3,5)之间都不会被填充，但是只有(1,4)与(1,5)之间不被填充才是正确的。而这种填充区域的二义性却很难通过邻接线段信息来解决，因为很难判断多边形的局部凹凸性。与其大费周折去判断这些特殊情况该怎么处理，不如想办法避免二义性的情况，然后得到填充区域的近似解。本文算法用一种近似的解决方案，使得算法避免处理二义性情况的同时，光栅化的效果几乎没有差别。思路很简单，就是给遇到填充二义性那条扫描线的 y 坐标加上一点偏移量之后重新求交，并用新的求交结果近似代替原二义性结果。即如果扫描线扫到多边形的顶点，就给这条扫描线的 y 坐标加一个足够小的偏移量，避免与多边形端点相交，然后再进行计算。如果加完坐标偏移量的扫描线还产生二义性，那么就不断就更多偏移量，或者想其他办法调整扫描线的 y 坐标，直到没有二义性为止。

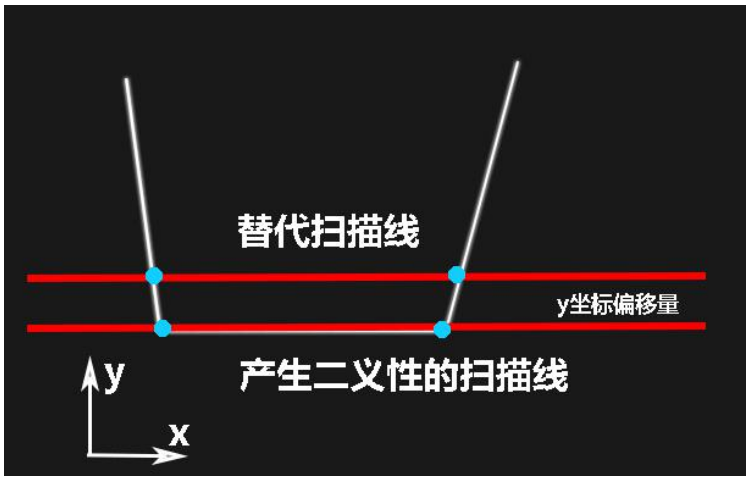


图 11：填充区域二义性的近似解决

2.3.4 体素化算法小结

本文提出的体素化方法分为三个阶段：

- 切层阶段：用一系列等距步进、平行于 XZ 平面的“扫描面”（亦称为“切层”）去切割闭合三角形网格模型，得到 $\Re_y$ 层切割轮廓，每一层应有一个或多个闭合多边形。轮廓用无序的矢量线段表示，即此阶段未生成多边形的拓补信息。
- 光栅化与扫描线填充阶段：对于切层阶段得到的每一个包含一系列线段的切层，用一系列等距步

进、平行于三维坐标系 Z 轴的“扫描线”（亦称为“切层”）去与闭合多边形相交。每条扫描线与多边形应相交得一个或多个交点，然后根据相邻交点序号的奇偶性来决定是否填充两个交点之间的区域。但是这种方法在扫描线落在多边形端点时可能会有二义性，本文提出了一种近似的解决方案。

- 合并光栅化结果：经过扫描线填充阶段之后，我们有了 $\Re_y$ 张分辨率为 $\Re_x \times \Re_z$ 二值化位图，其中第 j 层的位图里面的像素坐标 (i,k) 都对应着**体素空间坐标** (i,j,k)。体素模型就可以用下列**体素描述函数**来表示：

$$Vox(i,j,k)=\begin{cases}1,(i,j,k)\text{处有体素且}x\in[0,\Re_x),y\in[0,\Re_y),z\in[0,\Re_z),i,j,k\in Z\\ 0，\text{其他情况，}\ i,j,k\in Z\end{cases} \tag{2-10}$$

所以本文体素模型的有效空间分辨率为 $\Re_x \times \Re_y \times \Re_z$，量化分辨率为 2。要注意 $Vox(i,j,k)$ 的定义域是离散的，因此每个体素也可以被称作“体素点”（与像素点类似）

2.3.5 体素化算法实验结果

本文的体素化算法只是生成网格 LOD 模型的重要一环，但不是全部，所以本小节只展示体素化模型在各个分辨率下的可视化结果，此部分算法效率将在 LOD 模型生成算法介绍完之后的实验数据中展现。

为了精确地可视化体素模型，本文使用一个简单的办法网格化体素模型（仅仅把一个体素转换为一个正方体），从而使体素模型可以可视化。

考虑体素模型 $Vox(i,j,k)$，如果给定体素坐标 (i_0,j_0,k_0) 使得 $Vox(i_0,j_0,k_0)=1$，考虑此体素的 6-邻域，即体素坐标 (i_0+1,j_0,k_0)、(i_0-1,j_0,k_0)、(i_0,j_0+1,k_0)、(i_0,j_0-1,k_0)、(i_0,j_0,k_0+1)、(i_0,j_0,k_0-1)。如果 (i_0,j_0,k_0) 的某个 6-邻接体素值为 0，那么就生成两个体素的重合面（即生成两个邻接正方体重合的正方形面的网格）。

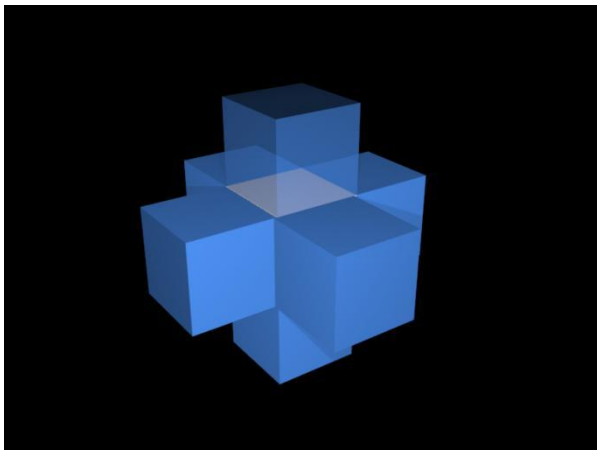


图 12：体素的 6-邻域

下面给出各个模型在不同分辨率体素化模型的可视化结果。因为体素模型直接网格化会产生海量的三角形，渲染效率较低，所以本小节模型体素化分辨率不设置太高。注意下面可视化的结果可能有少量拉伸。

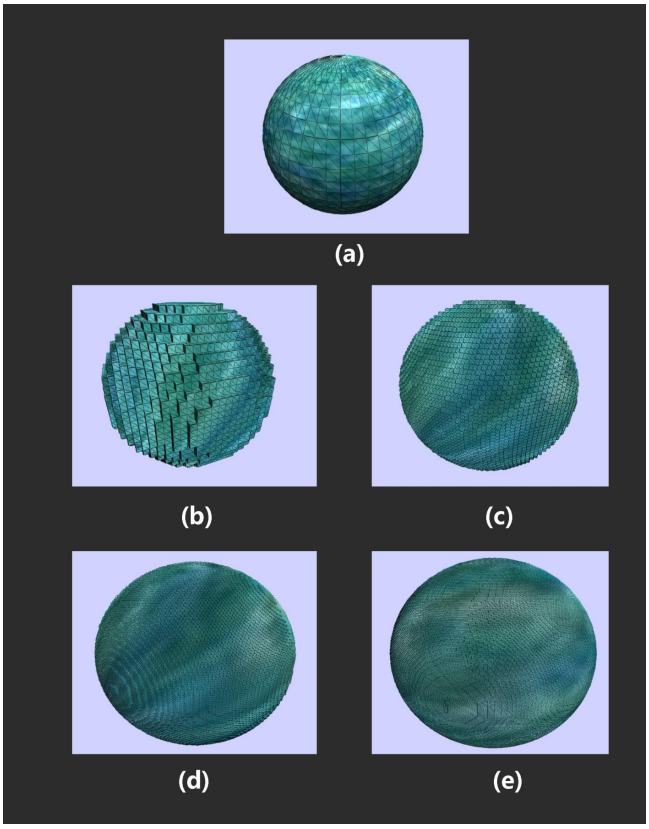


图 13：球体体素化模型的可视化结果。(a)为原网格，(b)(c)(d)(e)分别为分辨率 32x32x32、64x64x64、128x128x128、200x200x200 的体素模型的可视化结果

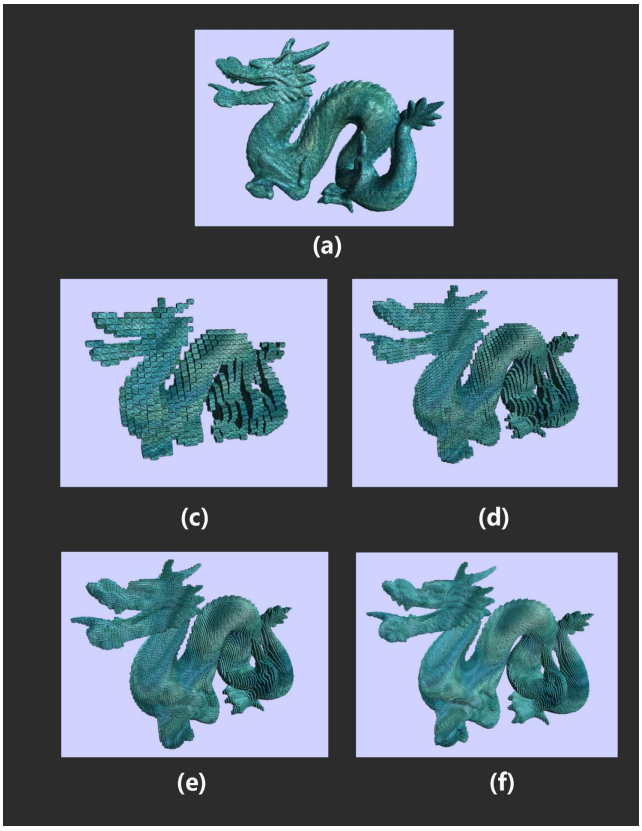


图 14 :龙模型体素化模型的可视化结果。(a)为原网格，(b)(c)(d)(e)分别为分辨率 32x32x32、64x64x64、128x128x128、200x200x200 的体素模型的可视化结果

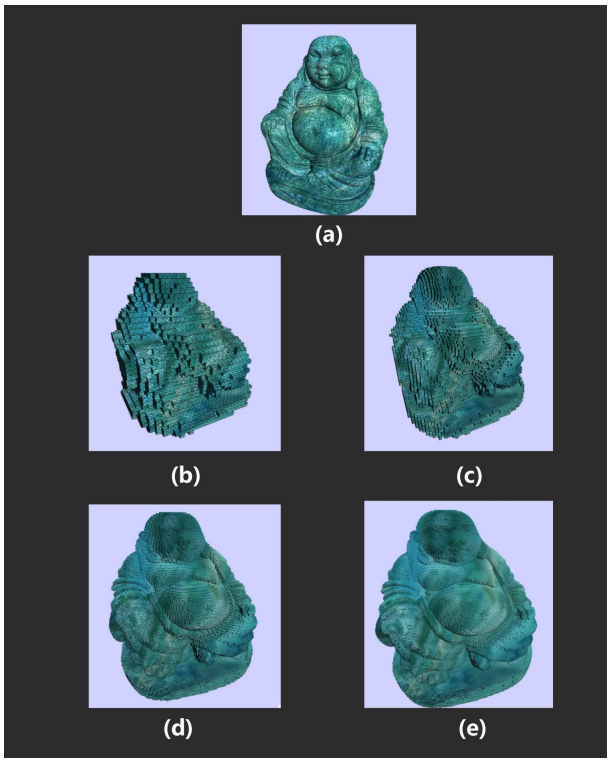


图 15 :佛模型体素化模型的可视化结果。(a)为原网格，(b)(c)(d)(e)分别为分辨率 32x32x32、64x64x64、128x128x128、200x200x200 的体素模型的可视化结果

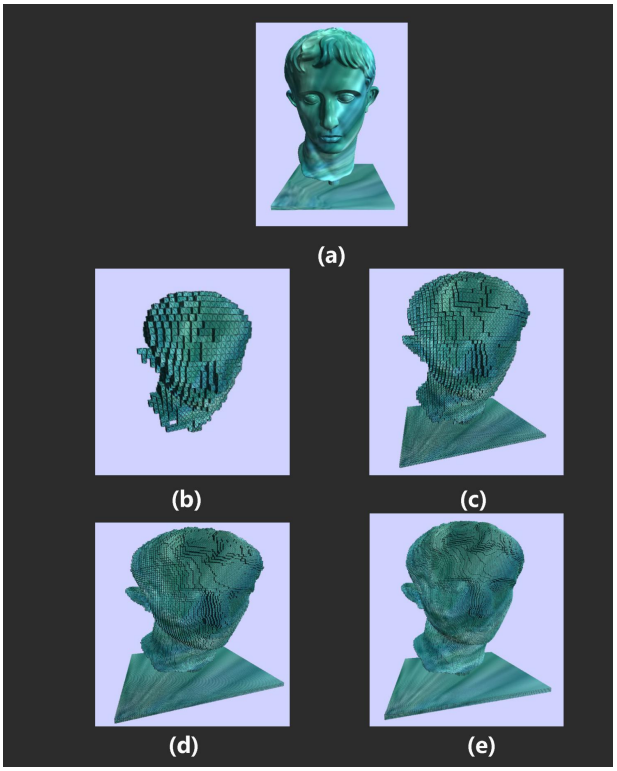


图 15：头雕像模型体素化模型的可视化结果。(a)为原网格，(b)(c)(d)(e)分别为分辨率 32x32x32、64x64x64、128x128x128、200x200x200 的体素模型的可视化结果

2.4 基于 Marching Cube 算法的网格重构

2.4.1 重采样

[16]在二维图像处理领域中，想要改变一张位图的分辨率，就肯定要涉及到用原图像中一定区域内的像素点去估计新图像的像素点的过程。而这个过程就是重采样的过程。重采样分为**升采样(Up-Sampling)**和**降采样(Down-Sampling)**。因为重采样这个过程是用原图像的某些像素综合起来去估计一个新图像的像素，所以这里就涉及到了“加权积分”与插值(Interpolation)。[16]把大部分的重采样方法广义地解释为卷积，因为卷积是一种比较通用的插值表示形式，所以不同的重采样算法一般只是有着不同的卷积核(Convolution Kernel)。

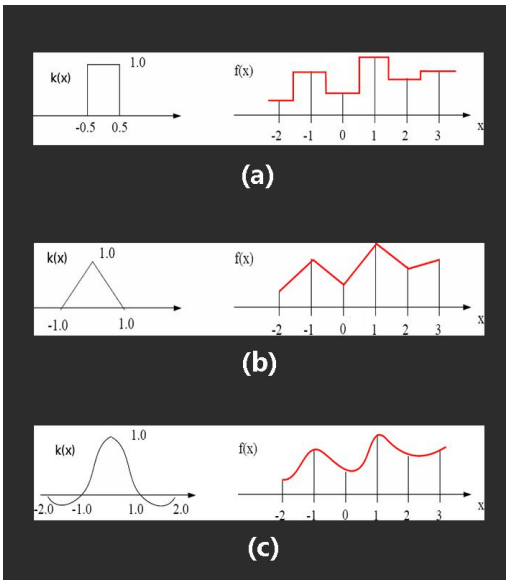


图 16：三种一维插值方法，分别用了不同的卷积核 k(x)。(a)为最邻近插值，(b)为线性插值，(c)为双三次插值。

从图 16 就可以看出在一维信号的插值中，不同的卷积核就会对产生不同的插值效果。这种基于卷积的重采样思路其实可以推广到更高维的情形。

2.4.2 三线性插值与扩展体素描述函数

与音频、图像类似，体素模型也可以看作是一组离散的数字信号，那么重采样与插值算法自然也可以运用在体素模型上。那么把双线性插值推广成三维情况，就可以使得体素值描述函数 $Vox(i,j,k)$ 定义域从 Z^3 延拓为 R^3。

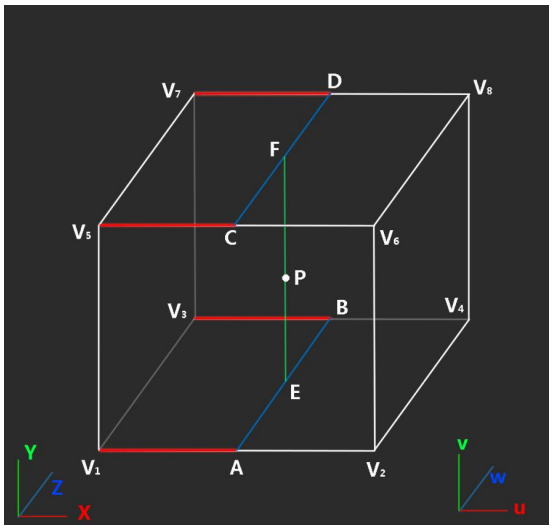


图 17：三线性插值示意图

设经过定义域延拓的扩展体素描述函数为 $VoxExt(i,j,k),i,j,k \in R$ ，此描述函数将建立在 $Vox(i,j,k)$ 的基础上。设要考察的体素空间中的一点 $P(i_p,j_p,k_p)$ ， $V_0,V_1,V_2...V_8$ 为体素空间中最靠近 P 的八个分量均为整数的点。则体素点 $V_0,V_1,V_2...V_8$ 的扩展体素空间坐标分别为

$$\begin{aligned}
V_1 &= (\lfloor i_p \rfloor + 0, \lfloor j_p \rfloor + 0, \lfloor k_p \rfloor + 0) \\
V_2 &= (\lfloor i_p \rfloor + 1, \lfloor j_p \rfloor + 0, \lfloor k_p \rfloor + 0) \\
V_3 &= (\lfloor i_p \rfloor + 0, \lfloor j_p \rfloor + 0, \lfloor k_p \rfloor + 1) \\
V_4 &= (\lfloor i_p \rfloor + 1, \lfloor j_p \rfloor + 0, \lfloor k_p \rfloor + 1) \\
V_5 &= (\lfloor i_p \rfloor + 0, \lfloor j_p \rfloor + 1, \lfloor k_p \rfloor + 0) \\
V_6 &= (\lfloor i_p \rfloor + 1, \lfloor j_p \rfloor + 1, \lfloor k_p \rfloor + 0) \\
V_7 &= (\lfloor i_p \rfloor + 0, \lfloor j_p \rfloor + 1, \lfloor k_p \rfloor + 1) \\
V_8 &= (\lfloor i_p \rfloor + 1, \lfloor j_p \rfloor + 1, \lfloor k_p \rfloor + 1)
\end{aligned} \tag{2-11}$$

也就是对于扩展体素空间中的任一点 P，我们要考虑其附近的 8 个体素点然后进行插值来来决定 P 处的值。而插值方法就是刚才提到的三线性插值。在三线性插值中，三个正交方向上插值比例系数为 u,v,w ，且有

$$\overrightarrow{V_1P} = u\overrightarrow{V_1V_2} + v\overrightarrow{V_1V_5} + w\overrightarrow{V_1V_3} \tag{2-12}$$

$$\begin{aligned}
u &= \frac{i_p - i_1}{\left|\overrightarrow{V_1V_2}\right|} = i_p - i_1 \\
v &= \frac{j_p - j_1}{\left|\overrightarrow{V_1V_5}\right|} = j_p - j_1 \\
w &= \frac{k_p - k_1}{\left|\overrightarrow{V_1V_3}\right|} = k_p - k_1
\end{aligned} \tag{2-13}$$

考察点 P 的值经过 8 个体素点值的三线性插值可以求得。设 $A_i = Vox(V_i), i = 0,1..8$ ，线性插值函数 $Lerp(a,b,t) = \mathrm{a} + (b-a) \times t = (1-t)a + bt$ ，则具体推导如下：

$$
\begin{aligned}
&VoxExt(x_p, y_p, z_p) \\
&= Lerp(Vox(E), Vox(F), v) \\
&= Lerp(Lerp(Vox(A), Vox(B), w), Lerp(Vox(C), Vox(D), w), v) \\
&= Lerp(Lerp(Lerp(A_1, A_2, u), Lerp(A_3, A_4, u), w), Lerp(Lerp(A_5, A_6, u), Lerp(A_7, A_8, u), w), v) \\
&= Lerp(Lerp((1-u)A_1 + uA_2, (1-u)A_3 + uA_4, w), Lerp((1-u)A_5 + uA_6, (1-u)A_7 + uA_8, w), v) \\
&= Lerp((1-u)(1-w)A_1 + u(1-w)A_2 + (1-u)wA_3 + uwA_4, (1-u)(1-w)A_5 + u(1-w)A_6 + (1-u)wA_7 + uwA_8), v) \\
&= (1-u)(1-v)(1-w)A_1 + u(1-v)(1-w)A_2 + (1-u)(1-v)wA_3 + u(1-v)wA_4 \\
&+ (1-u)v(1-w)A_5 + uv(1-w)A_6 + (1-u)vwA_7 + uvwA_8
\end{aligned}
\tag{2-14}
$$

综合(2-11)(2-12)(2-13)(2-14)，扩展体素描述函数 $VoxExt(i, j, k), i, j, k \in R$ 在体素空间中任意点的函数值可以被确定。因为 $VoxExt(i, j, k)$ 是由二值化的体素描述函数 $Vox(i, j, k)$ 在多个点取的值的线性内插组合得到，所以 $VoxExt(i, j, k) \in [0,1]$。扩展体素描述函数将在之后网格重构阶段作为体素空间任意点的采样工具出现。

2.4.3Marching Cubes 算法

[8]Marching Cubes 算法是 Lorensen 等人在 1987 年提出的一个等值面提取算法，至今(2017.9)引用数已经破万，下文将简称 MC 算法。[8]介绍 MC 算法可以应用于 CT/MRI 扫描等医学图像的三维可视化，由一系列的二维图像切片重构出一个三角形网格。因为本质上 CT/MRI 等医学扫描图像单个切片是位图，所以其实 CT/MRI 扫描图可以轻易地转化为体素模型，而这也暗示了 MC 算法可以用在本文的网格重构阶段。

MC 是一种是分治(divide and conquer)算法。算法名字里面的“Cubes”，也就是小立方体（或者小长方体），就是 MC 算法分治处理的基本单位。MC 算法本质上是把模型包围盒的空间分成 $C_x \times C_y \times C_z$ 个相邻小长方体空间，对于每一个小长方体，分别判断其八个顶点是否在物体的内部。如果某个顶点在内部，那么就给这顶点标号为 1，否则标号为 0，那么一个小长方体的 8 个顶点就有 $2^8 = 256$ 种标号组合。而判断某个顶点是否在物体内部就要用一个指示函数（indicator function）$F(x, y, z)$ 来判断，如果设定阈值为 $Threshold$ 的话，可以根据 $F(x_0, y_0, z_0)$ 与 $Threshold$ 的大小关系来确定点 (x_0, y_0, z_0) 是否在物体内部。

这 256 种情况都有自己的等值面三角化情形，256 种情形又可以概括为 15 种基本情况，具体可参考[8]。只要把所有小长方体都处理一次，那么就可以生成表示模型轮廓的三角形网格了。

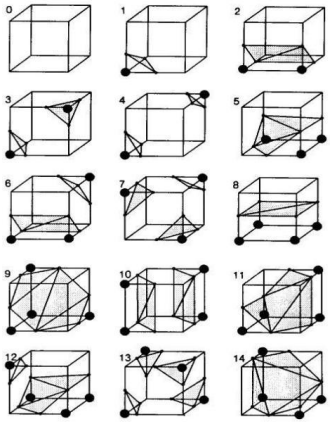


图 18：[8]中对 15 种立方体基本状态的三角化方法

对于本文算法而言，判断一个顶点是否在物体内要在体素空间完成。指示函数可以使用 2.4.2 中的扩展体素描述函数 $VoxExt(i,j,k)$ 。

2.4.4 降采样与网格重构

MC 算法里面对于每个小长方体，新生成的三角形的顶点都会在长方体的棱上。而如果我们直接用二值化的体素描述函数作为 MC 的指示函数进行网格重构，那么新三角形的顶点在长方体棱上的相对位置就只能是固定的。这样子很容易造成生成网格有较明显的“阶梯锯齿”，且生成网格的面数也会非常多。

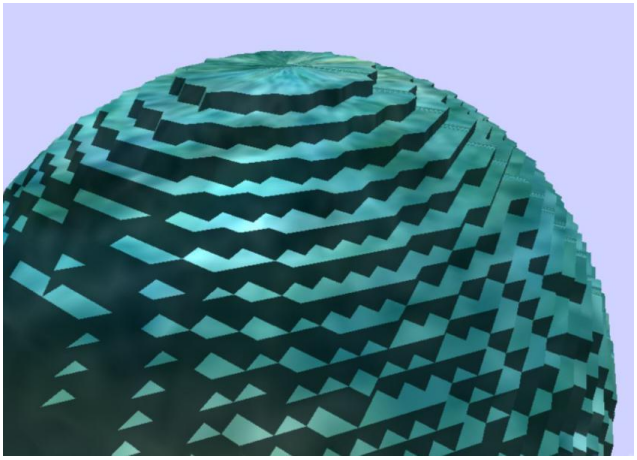


图 19：直接对二值化体素模型用 MC 算法重构网格，会产生大量“阶梯状”锯齿

为了减少重构的网格锯齿以及让其更加平滑，本文算法的体素化阶段可以看作是一次**超采样(super-sampling)**。在本阶段，体素模型进行降采样之后，用 MC 算法重构出近似的网格。这里**降采样的意思是 MC 算法的基本单元——MC 立方体的个数会比体素模型的体素数要少**。为了取得 MC 算法里的长方体每一个顶点处的描述函数值，就要对体素模型进行一次采样。只要采样的分辨率低于体素模型 $Vox(i,j,k)$ 的分辨率，那么这个过程就可以称作是降采样(down-sampling)。设本阶段的 MC 算法中，使用了 $C_x \times C_y \times C_z$ 个小长方体去重构网格，则此次降采样在 x，y，z 轴的采样分辨率记为 $\Lambda_x \times \Lambda_y \times \Lambda_z$ ，则显然

$$\begin{aligned} \Lambda_x &= C_x + 1 \\ \Lambda_y &= C_y + 1 \\ \Lambda_z &= C_z + 1 \end{aligned} \tag{2-15}$$

其中 $\Lambda_x, \Lambda_y, \Lambda_z \in Z^*$ 。要注意一件事，虽然扩展体素描述函数 $VoxExt(i,j,k)$ 的定义域为 R^3 ，但由于此函数仅仅是由分辨率为 $\Re_x \times \Re_y \times \Re_z$ 的体素模型插值得到的，所以可以认为 $VoxExt(i,j,k)$ 的有效分辨率依然为 $\Re_x \times \Re_y \times \Re_z$ 。这里要定义一个**降采样比例**的概念：在与坐标轴平行的方向上，原分辨率与降采样后分辨率之比。所以在三个坐标轴方向上的降采样比例分别为：

$$\varphi_x = \frac{\Re_x}{\Lambda_x}, \quad \varphi_y = \frac{\Re_y}{\Lambda_y}, \quad \varphi_z = \frac{\Re_z}{\Lambda_z} \tag{2-16}$$

因为本文算法最初的输入是一个网格，为了要维持网格的空间尺寸基本不变，需要用降采样比例作为放缩系数去放缩由 MC 生成的网格。所以单个 MC 小长方体在模型空间中的沿 x,y,z 轴的长度分别为：

$$\begin{aligned} CubeWidth_{scaled} &= \frac{RangeX(mesh)}{\Re_x} \times \varphi_x \\ CubeHeight_{scaled} &= \frac{RangeY(mesh)}{\Re_y} \times \varphi_y \\ CubeDepth_{scaled} &= \frac{RangeZ(mesh)}{\Re_z} \times \varphi_z \end{aligned} \tag{2-17}$$

经过一定比例的降采样之后，在模型空间中，一个 MC 立方体平均能容纳 $\varphi_x \times \varphi_y \times \varphi_z$ 个体素点。所以用 MC 算法重构网格还有一个重要问题要解决 ,那就是每个 MC 立方体棱上顶点的线性插值比例怎么确定。

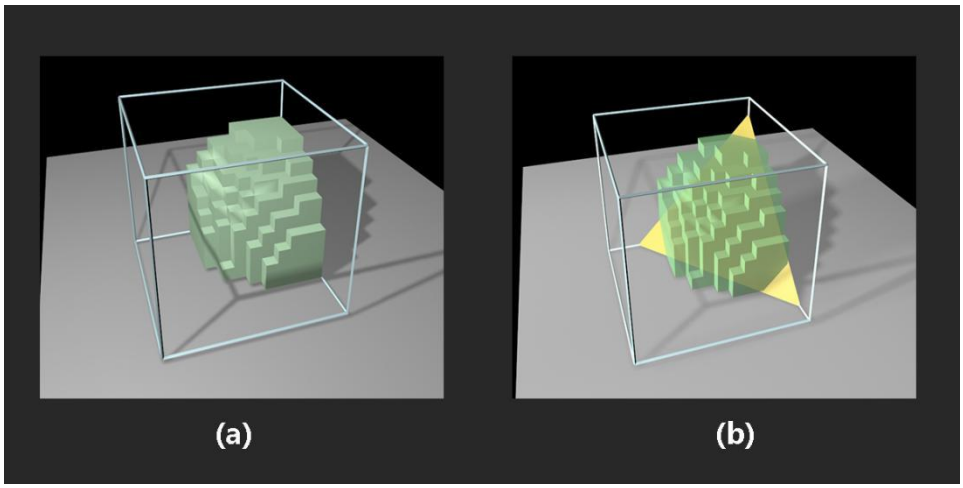


图 20：经过放缩的 MC 小立方体能容纳许多体素，要想办法用合适的三角形网格去近似描述局部曲面，如(b)的黄色三角形。这些生成的三角形的顶点都会在 MC 小立方体的棱上产生，所以要用一些合适的方式去计算三角形顶点在立方体棱上的相对位置

由于 MC 是个分治算法，所以每个小立方体之间的关系相对比较独立，相对联系较浅。但为了使重构出来的三角形网格连续而没有空洞、裂缝，即相邻 MC 立方体产生的三角形是连着的，那么在同一条立方体棱边（一条棱会被 4 个立方体共享）上产生的新三角形顶点位置要一致，不能让共享某条棱的 4 个 MC 立方体在共享棱上产生不同位置的顶点。在这里假设某条棱（线段）$L(V_1, V_2)$ 上会产生新顶点 V_{new} ，且 $V_{new} = Lerp(V_1, V_2, t)$ ，那么设用于估计某条棱上新顶点线性插值坐标 t 的函数为：

$$EdgeRatio(V_1, V_2)$$

其中 V_1, V_2 均为体素空间下的点。函数通过在这条棱上从 V_1 到 V_2 均匀地采样，设采样点序列为 $S_0, S_1 \ldots S_n$ ，若 S_0 在物体内部而 S_i 在外部、或者 S_0 在物体外部而 S_i 在内部，那么就返回 S_i 对应的线性插值坐标。而采样点 S 在物体的内外部情况，就由 $VoxExt(S)$ 与一个自定义阈值的大小关系决定。下面给出此函数的伪代码：

```
'函数：EdgeRatio，用扩展体素描述函数求棱(V1,V2)上新顶点的线性插值系数
EdgeRatio(V1,V2)
begin
  sampleCount=0                                    '棱上的步进采样次数
  binarizedThreshold =区间(0,1)中的一个自定义值      'VoxExt(V)大于此阈值时，V 在物体内部，否则在外部
  if L(V1,V2)平行于 X 轴 Then sampleCount=φ_x       'φ_x 为 x 方向降采样比例
  else if L(V1,V2)平行于 Y 轴 Then sampleCount=φ_y  'φ_y 为 y 方向降采样比例
  else if L(V1,V2)平行于 Z 轴 Then sampleCount=φ_z  'φ_z 为 z 方向降采样比例

  edgeStartValue = VoxExt(V1)                      '取 V1 处的体素值
    for i = 0 to sampleCount-1                     '要在棱上进行 sampleCount 次采样，从 V1 步进到 V2
      stepLength = (L(V1,V2)的长度)/SampleCount
      t = i * stepLength                           'V1 到 V2 的线性插值比例
      V_Sample=Lerp(V1,V2,t)                       '在棱 L(V1,V2)上的采样点
      sampleValue = VoxExt(V_Sample)

      b1=edgeStartValue 是否大于 binarizedThreshold 'b1 是个布尔值
      b2=sampleValue 是否大于 binarizedThreshold   'b2 也是个布尔值
      if b1 不等于 b2 Then                         '若采样值与阈值的大小关系与一开始相比有所变化
          return t                                 '返回线性插值坐标
      end if
    end for
    return 1.0                                     '采样完整条棱 b1 依旧等于 b2
end
```

2.4.5 降采样网格重构阶段小结

本阶段主要用了 Marching Cubes 算法的框架去重构网格。本阶段输入是体素化阶段输出的的二值化体素模型。先把模型空间的模型包围盒分成 $C_x \times C_y \times C_z$ 个小型 MC 立方体，每一个 MC 立方体是 MC 算法的分治处理单元。MC 立方体的每个顶点都对应着一次体素空间的采样。要注意 $C_x \times C_y \times C_z$ 应远小于体素模型分辨率（这个特点下一节会提到），也就是降采样比例要比较大，重构结果才会比较好。对于每一个 MC 立方体，先判断 8 个顶点是否在体素模型的内部，判断方法是那顶点在体素空间的采样值——指示函数：扩展体素函数 $VoxExt(i,j,k)$ 值是否大于某个处于 $(0,1)$ 的自定义阈值，大于就是内部，小于就是外部。根据 8 个顶点的在模型内外部的情况就可以确定此 MC 立方体的三角化情形（最多 256 种情形）。其中生

成的新三角形的顶点都在 MC 立方体的棱上，其在棱上的相对位置用函数 $EdgeRatio(V_1,V_2)$ 来确定。用上述过程处理所有 MC 立方体便可以得到重构的三角形网格。

3 实验结果

3.1 实验设计

本文的网格 LOD 模型生成算法分了两个阶段：体素化阶段、网格重构阶段。所以设计实验要监测的变量就非常的明显了：

- 初始网格的三角形数（体素化阶段）
- 体素化分辨率（体素化阶段）
- 降采样分辨率（网格重构阶段）

本文讨论的衡量算法效果的指标有：

- 算法运行时间（定量）
- 网格生成质量（定性）
- 简化率（定量，简化网格三角形数比上原网格三角形数）

本文的算法的实验代码用 c++实现，平台 Windows 8.1，采样单线程的 cpu 实现，简化算法无硬件加速。

实验硬件：

CPU：Intel(R) Core(TM) i7-4720HQ CPU@2.60GHz，

内存：8GB

3.2 不同初始网格面数的对比

模型名称	体素化分辨率	降采样分辨率	模型面数	重构网格面数	耗时-体素化(ms)	耗时-网格重构(ms)	总耗时(ms)	简化率
球体	200x200x200	20x20x20	2401	3740	16.11	14.67	30.78	1.5577
奶牛			8627	3344	19.91	13.08	32.99	0.3876
佛			61892	3976	37.89	14.84	52.73	0.0642
龙			100001	3988	56.71	15.27	71.98	0.0399
头部雕像 1			499059	2632	91.62	10.91	102.53	0.0053

表 3-1：输入模型面数与算法执行效率的关系

因为输入模型的三角形面数与最终生成网格的外观并无太大关系，所以这里值给出效率的分析。在固定了体素化分辨率与降采样分辨率后，此节采用了不同的模型进行实验（这也意味着有不同的面数）。可以发现在一定的体素化分辨率下，**体素化阶段耗时与模型面数成正比，网格重构阶段耗时与模型面数基本无关**。具体的运行效率还是取决于具体的模型情况，所以下列图线没有表现出很好的线性性。

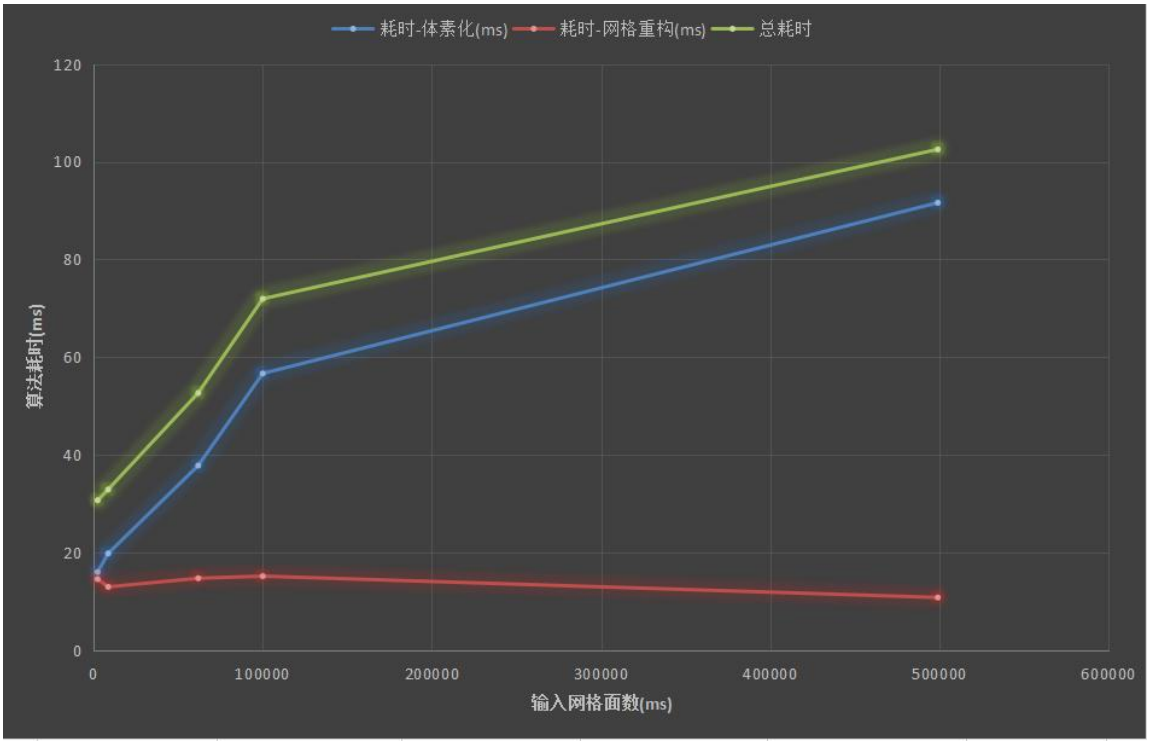


图 21：表 3-1 的 XY 散点图。体素化阶段耗时与输入模型面数成正比

3.3 不同体素化分辨率的对比

模型名称	体素化分辨率	降采样分辨率	模型面数	重构网格面数	耗时-体素化(ms)	耗时-网格重构(ms)	总耗时(ms)	简化率	降采样比例
佛	50x50x50	50x50x50	61892	27080	10.1721	68.7677	78.9398	0.4375	1x1x1
	100x100x100			27276	18.9258	76.2594	95.1852	0.4407	2x2x2
	150x150x150			27252	29.0245	84.4376	113.4621	0.4403	3x3x3
	200x200x200			27228	42.878	92.3311	135.2091	0.4399	4x4x4
	300x300x300			27160	73.0544	104.834	177.8884	0.4388	6x6x6
	500x500x500			27068	151.841	132.292	284.133	0.4373	10x10x10

表 3-2：佛祖模型用不同体素化分辨率时算法执行效率

模型名称	体素化分辨率	降采样分辨率	模型面数	重构网格面数	耗时-体素化(ms)	耗时-网格重构(ms)	总耗时(ms)	简化率	降采样比例
牛	50x50x50	20x20x20	8627	3548	23.7292	6.7152	30.4444	0.4113	2.5x2.5x2.5
	100x100x100			3384	27.6235	9.0261	36.6496	0.3923	5x5x5
	150x150x150			3408	35.1337	11.8841	47.0178	0.3950	7.5x7.5x7.5
	200x200x200			3344	40.0362	13.0735	53.1097	0.3876	10x10x10
	300x300x300			3344	60.6014	17.5745	78.1759	0.3876	15x15x15
	500x500x500			3340	108.269	28.3968	136.6658	0.3872	25x25x25

表 3-3：牛模型用不同体素化分辨率时算法执行效率

图表 3-3 与 3-4 分别是两个模型的实验结果，均以体素化分辨率为自变量。观察表中数据可以留意到

- **重构网格面数、简化率均与体素化分辨率基本无关。**
- **体素化阶段耗时与体素化分辨率成正比，网格重构阶段耗时也与体素化分辨率成正比。**

之所以体素化分辨率看起来会影响到网格重构阶段的效率，是因为在此实验中降采样分辨率 $\Lambda_x \times \Lambda_y \times \Lambda_z$ 不变，体素化分辨率 $\Re_x \times \Re_y \times \Re_z$ 的改变直接影响了降采样比例。而降采样比例的改变会影响 $EdgeRatio(V_1, V_2)$ 函数在求解时使用 $VoxExt(i, j, k)$ 采样的次数，即降采样比例越高，$EdgeRatio(V_1, V_2)$ 函数求解时间越长，也就导致了网格重构阶段耗时加长。

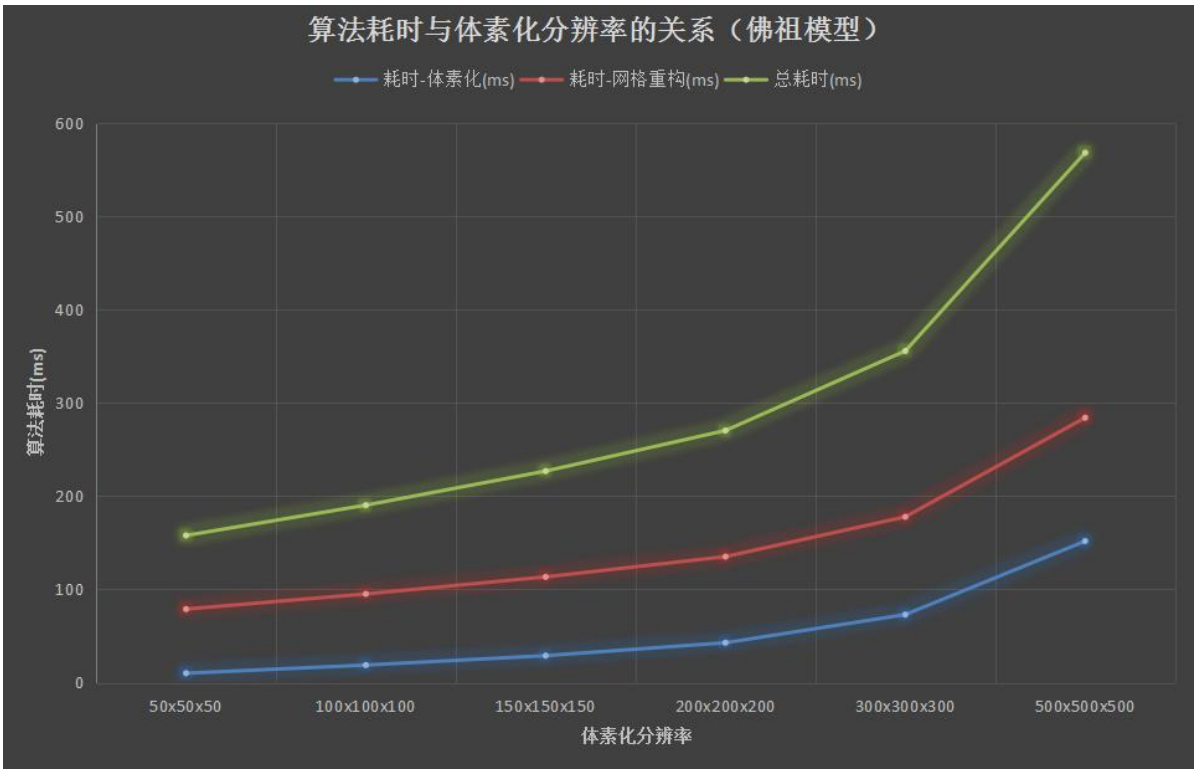


图 22：表 3-2 的 XY 散点图，三条折线的趋势说明算法的两个阶段的耗时均与体素化分辨率成正比

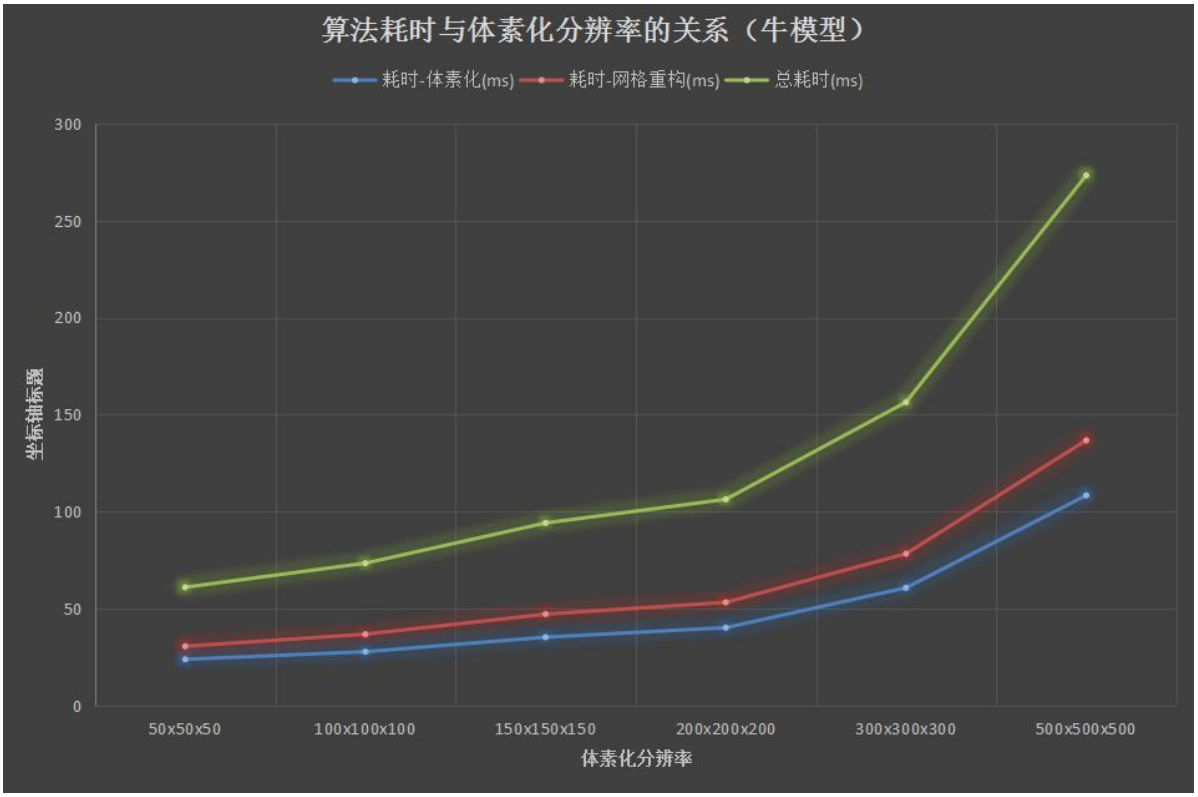


图 23：表 3-3 的 XY 散点图，与表 3-2 趋势一致但有所不同，说明算法的效率还取决于具体的模型

下面是各组实验数据对应的重构网格效果图：

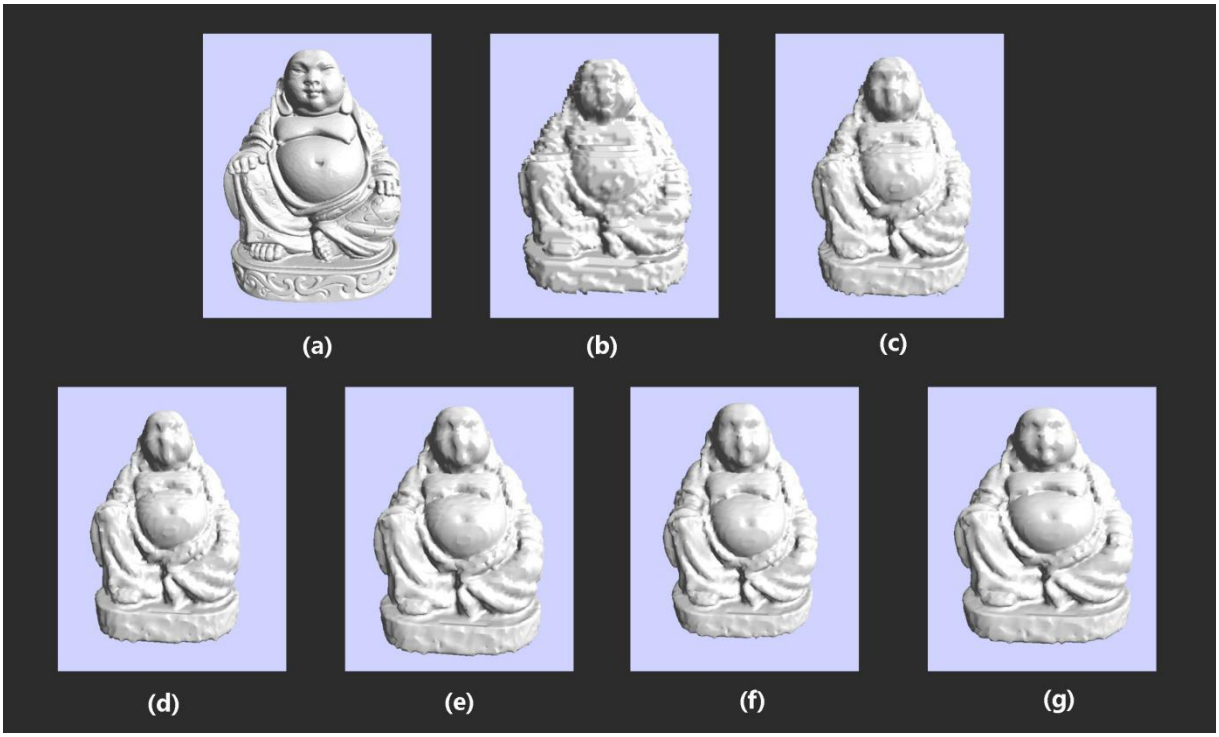


图 24：表 3-2 的实验截图，(a)为原网格，其他均为不同参数设置的输出网格。
遗憾的是，部分区域切线难以恰当计算造成光影效果不够光滑

图 25：佛祖模型的重构网格（体素化分辨率 500x500x500,降采样分辨率 50x50x50）

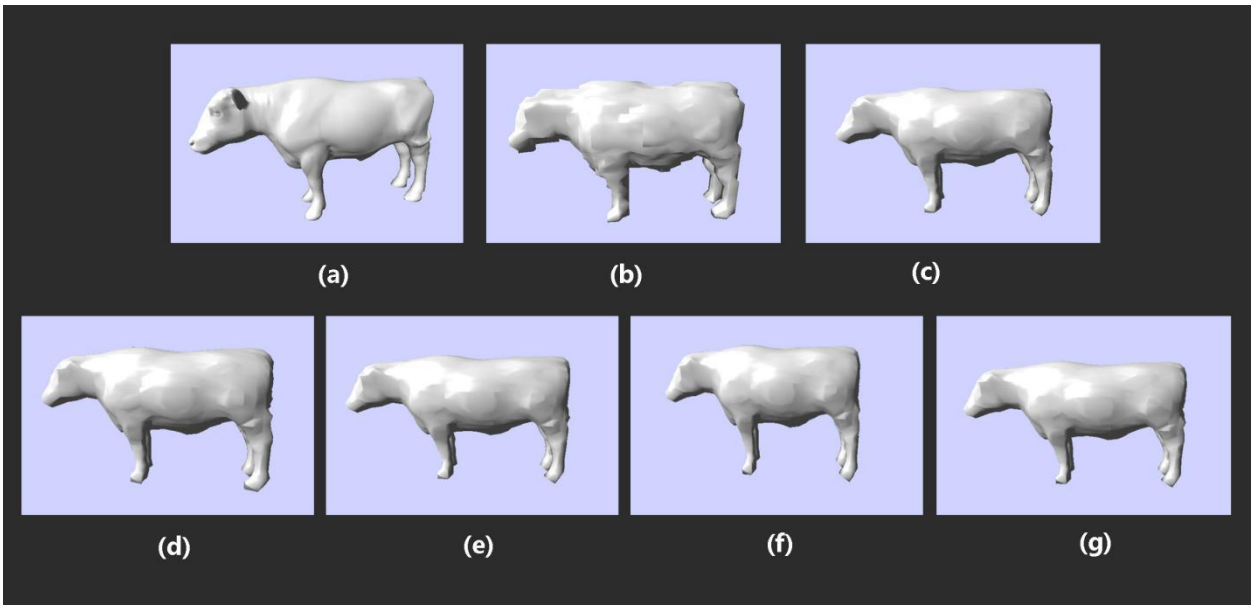


图 26：表 3-3 的实验截图，(a)为原网格。可以观察到(d)(e)(f)(g)的网格外观变化几乎没有变化。可见降采样比例太大的时候，精确度的增加人眼已经有点难以辨别了

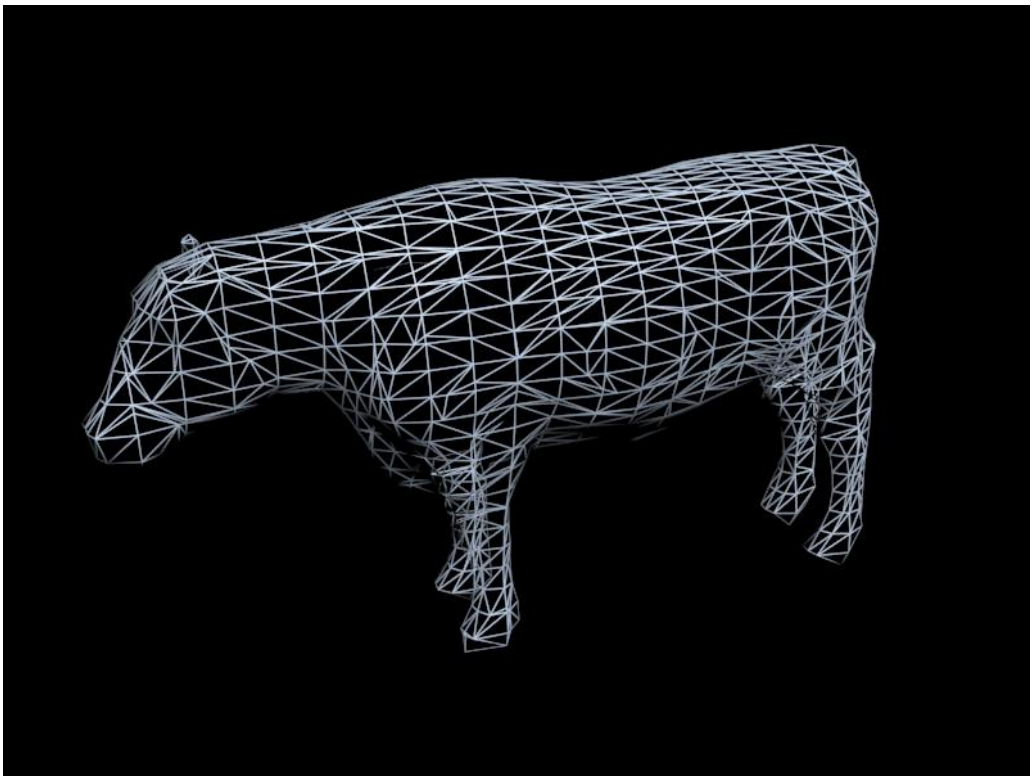


图 27：牛的重构网格（体素化分辨率 500x500x500，降采样分辨率 20x20x20）

3.4 不同的降采样分辨率的对比

模型名称	体素化分辨率	降采样分辨率	降采样体素数	模型面数	重构网格面数	耗时-体素化(ms)	耗时-网格重构(ms)	总耗时(ms)	简化率	降采样比例
Aphrodite 雕像	100x100x100	10x10x10	1000	579905	832	72.6059	2.7088	75.3147	0.0014	10x10x10
		20x20x20	8000		3468	73.7687	8.9589	82.7276	0.0060	5x5x5
		30x30x30	27000		8204	71.6364	20.645	92.2814	0.0141	3.3x3.3x3.3
		50x50x50	125000		23112	73.9485	70.655	144.6035	0.0399	2x2x2
		100x100x100	1000000		95036	74.6696	430.612	505.2816	0.1639	1x1x1

表 3-4：Aphrodite 雕像模型（体素化分辨率 100x100x100）在不同降采样分辨率下的数据

模型名称	体素化分辨率	降采样分辨率	降采样体素数	模型面数	重构网格面数	耗时-体素化(ms)	耗时-网格重构(ms)	总耗时(ms)	简化率	降采样比例
Aphrodite 雕像	500x500x500	10x10x10	1,000	579905	832	278.678	14.4915	293.1695	0.0014	50x50x50
		20x20x20	8,000		3496	278.245	30.1771	308.4221	0.0060	25x25x25
		30x30x30	27,000		8096	279.213	51.0057	330.2187	0.0140	16.6x16.6x16.6
		50x50x50	125,000		23232	275.612	120.693	396.305	0.0401	10x10x10
		100x100x100	1,000,000		95368	276.998	551.183	828.181	0.1645	5x5x5

表 3-5：Aphrodite 雕像模型（体素化分辨率 500x500x500）在不同降采样分辨率下的数据

模型名称	体素化分辨率	降采样分辨率	降采样体素数	模型面数	重构网格面数	耗时-体素化(ms)	耗时-网格重构(ms)	总耗时(ms)	简化率	降采样比例
龙	300x300x300	10x10x10	1,000	100001	788	90.125	7.00376	97.12876	0.0079	30x30x30
		20x20x20	8,000		3932	92.625	25.4709	118.0959	0.0393	15x15x15
		30x30x30	27,000		9344	91.3612	41.0049	132.3661	0.0934	10x10x10
		50x50x50	125,000		27420	89.1138	114.834	203.9478	0.2742	6x6x6
		100x100x100	1,000,000		111840	89.873	532.142	622.015	1.1184 *	3x3x3

表 3-6：龙模型（体素化分辨率 300x300x300）在不同降采样分辨率下的数据

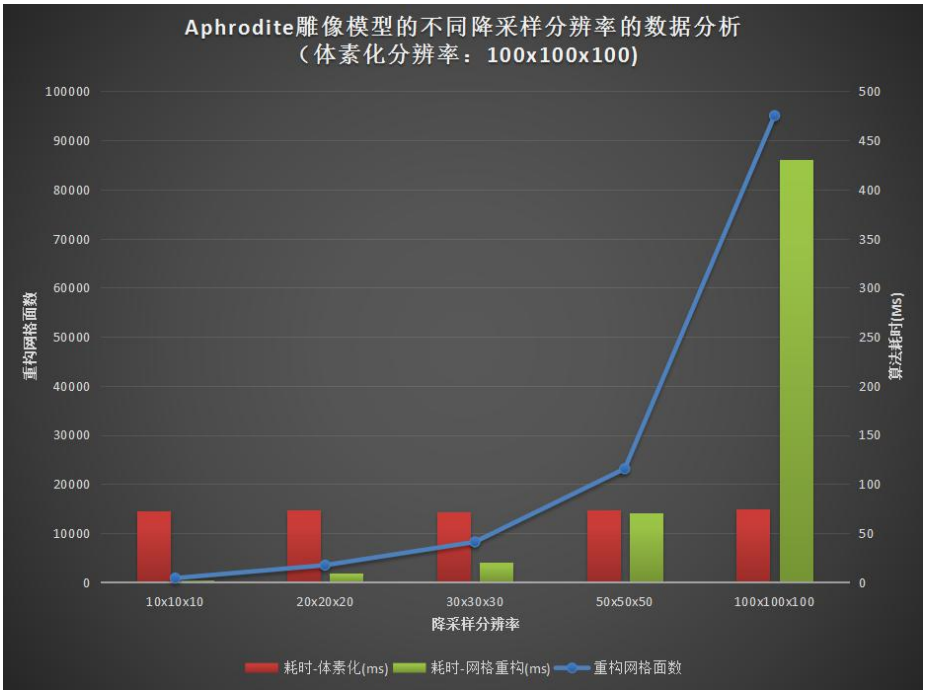


图 28：表 3-4 的数据可视化结果，表明网格重构耗时、重构网格面数均与降采样分辨率成正比

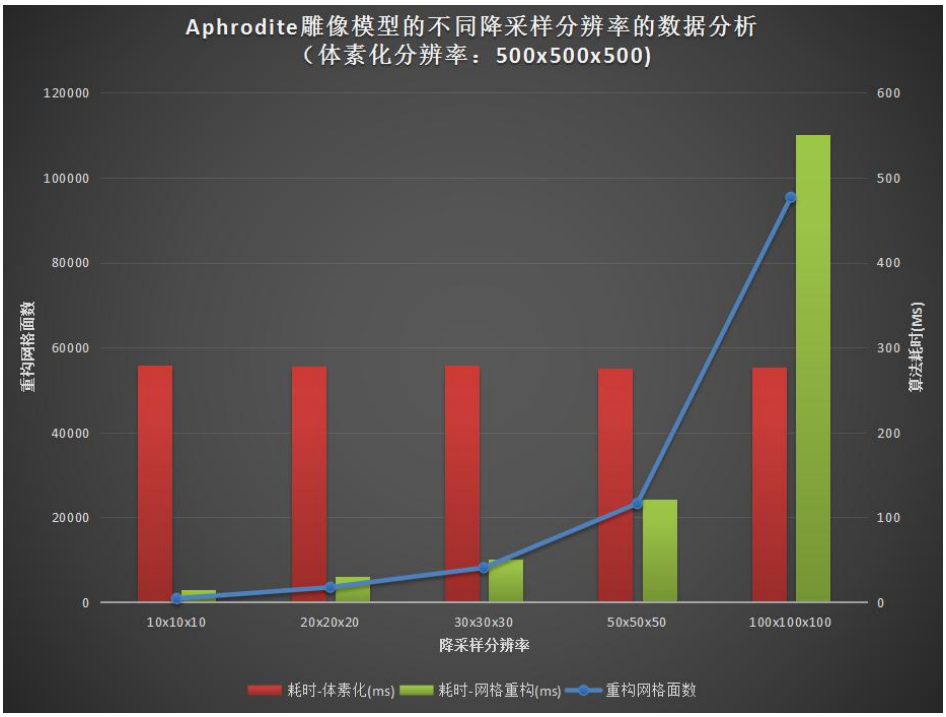


图 29：表 3-5 的数据可视化结果，趋势与图 28 展现的比较类似，体素化阶段耗时比表 3-4 中的结果长也在预料之中。但是要结合下文的模型可视化结果才能综合分析

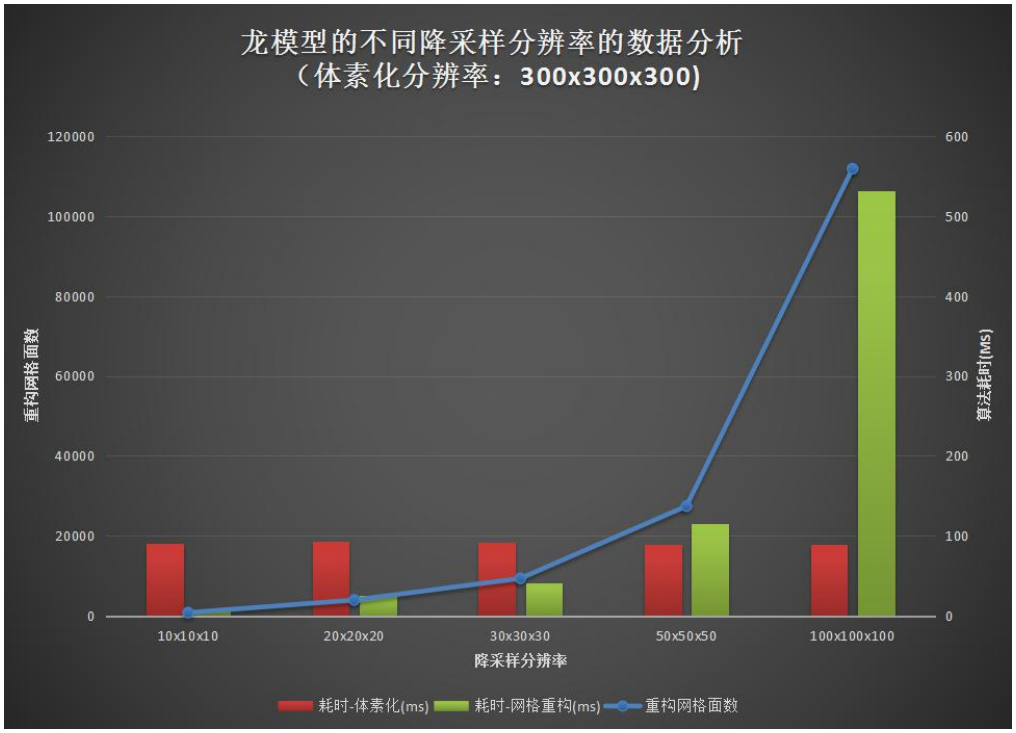


图 30：表 3-6 的数据可视化结果，趋势与图 28、29 均类似

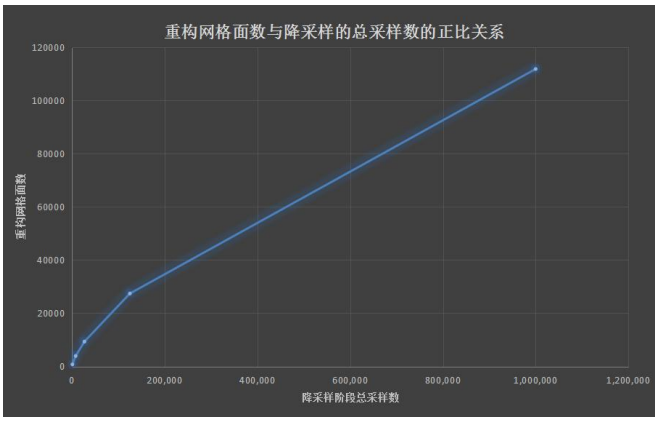


图 31：表 3-6 的部分数据可视化结果。可以看到重构网格面数与降采样总采样数成正比，曲线略微比线性函数低阶。

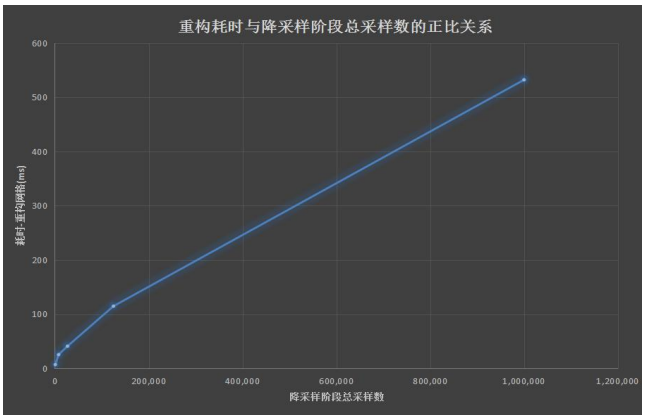


图 32：表 3-6 的部分数据可视化结果。可以看到网格重构耗时与重构网格面数成正比，曲线略微比线性函数低阶。

上面三组实验的数据可以分析出一些结论与算法特性：

- 重构网格面数与降采样的总采样数成正比：降采样分辨率是分成三个方向上的分辨率来表示的，分别为 $\Lambda_x, \Lambda_y, \Lambda_z$ ，所以整个模型在降采样阶段的总采样数为 $\Lambda_x\Lambda_y\Lambda_z$ 。由表 3-4 与 3-5 中的数据变化趋势可以发现，重构网格面数与降采样的总采样数成正比，也意味着用于生成新三角形的 MC

小正方体越多，最终网格面数越多。这从原理上也很容易分析得出来，实验结果也基本符合。其中图 31 更显示出在其他变量不变时这个正比关系可以近似为线性关系。

- 重构网格耗时与降采样的总采样数成正比：这个其实也比较的显然，因为 MC 小立方体越多，求解 8 个顶点采样值与求解 $EdgeRatio(V_1,V_2)$ 的次数也就越多，从而导致算法耗时变长。所以从原理和实验上都说明了重构网格耗时与降采样的总采样数成正比。图 32 显示了这两个量在其他变量不变时具有一定的线性性。
- 简化率可以超过 1：表 3-6 中打星号的项表明简化率可以超过 1。网格简化率超过 1 的意思是重构后的网格模型面数比输入要多。这个特性是一把双刃剑，如果在单纯只需要简化网格的情境下这个特性就可以算是一个缺点了。如果某个情景需要控制简化率小于 1，那么整个 LOD 模型序列需要从低降采样分辨率开始，逐步提高降采样分辨率，然后用前几次重构生成的低精度模型的面数使用线性回归近似求得结果面数的上界，并根据这个上界判断更高的降采样分辨率是否会生成简化率大于 1 的网格模型。
- 降采样分辨率可以控制输出网格面数：因为本文算法可以从外部传入的参数中包含“降采样分辨率”，而理论分析与实验数据都表明重构网格面数与降采样分辨率成正比，所以可以知道，降采样分辨率可以控制输出网格面数。
- 模型简化率越高，重构速度越快：这个特性与基于“几何对象消除”原理的网格简化算法刚好相反。边坍缩、三角形收缩等方法随着每步迭代减少简化率，所以简化率越高，重构速度越慢。

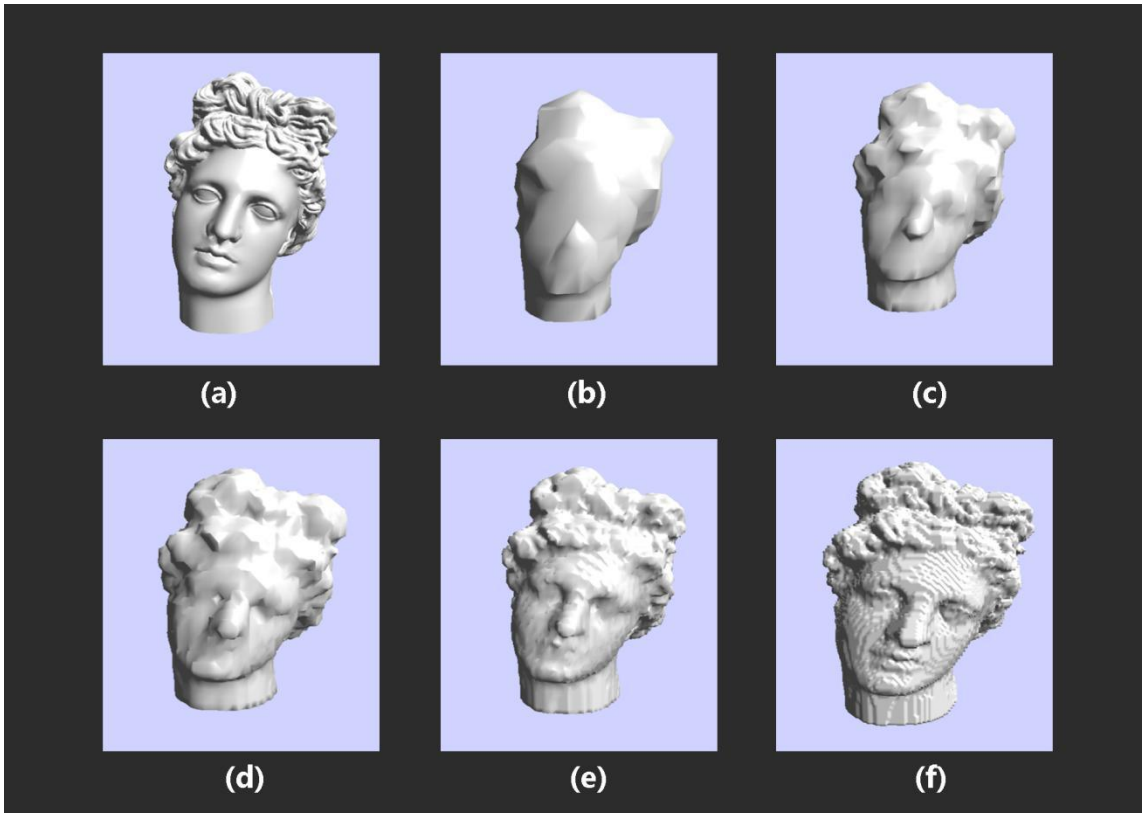


图 33：表 3-4 的模型截图，(a)为原网格。可以看到如果降采样比例不够大，可能会生成(f)那样有明显锯齿的网格

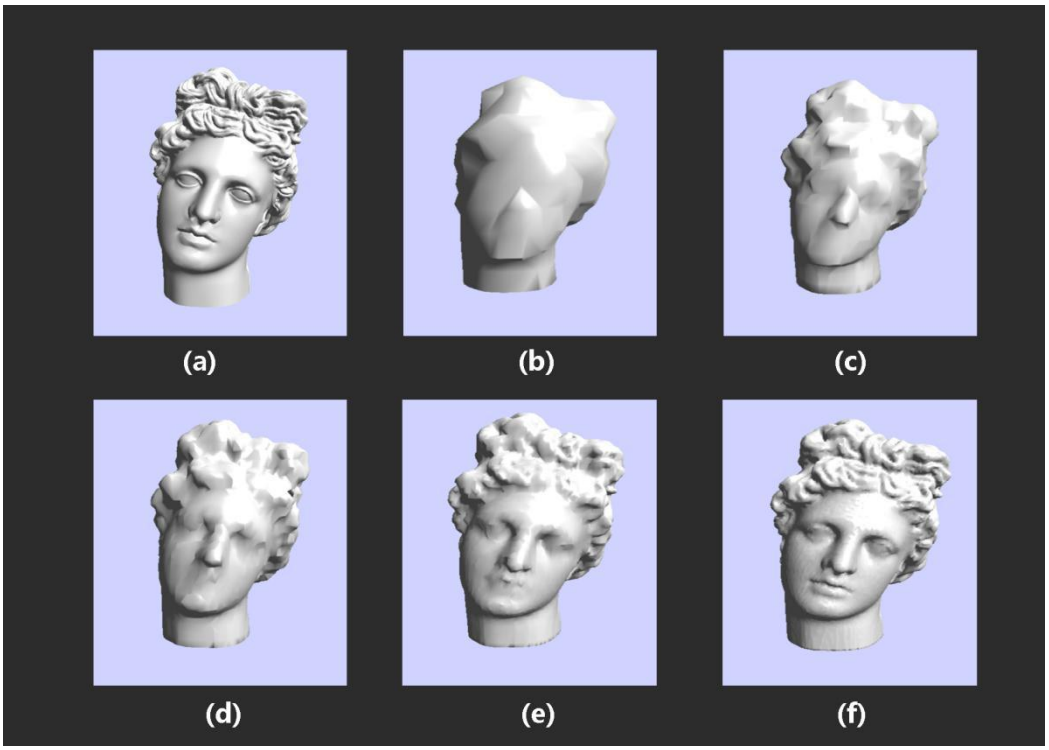


图 34：表 3-5 的模型截图，(a)为原网格。相比起图 33 的结果，这里的(d)(e)(f)明显更加光滑与精细。

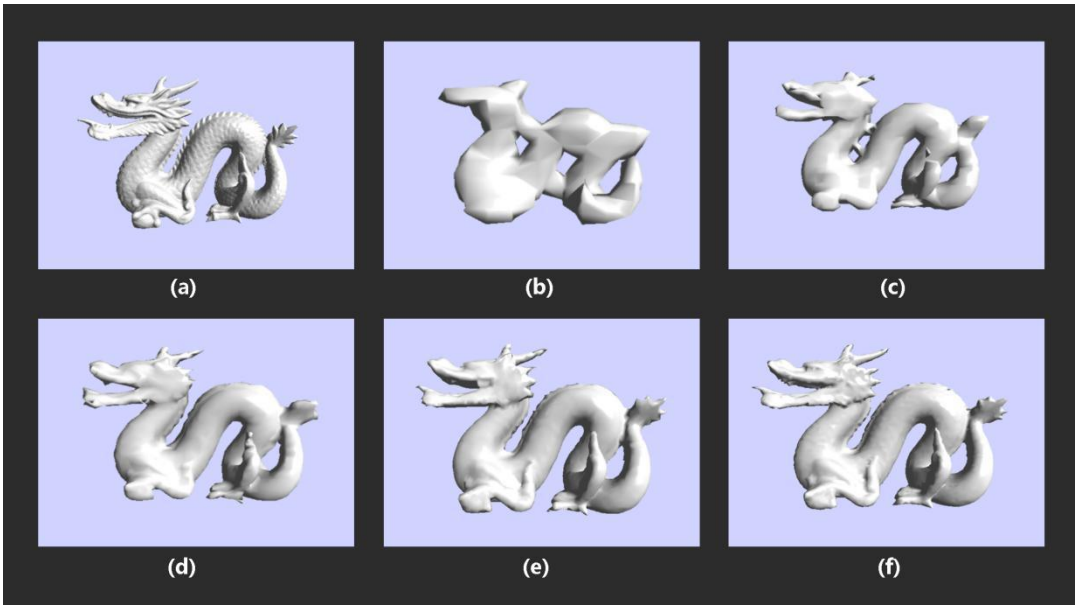


图 35：表 3-6 的模型截图，(a)为原网格，(b)-(f)都焊接了顶点。其中(f)的面数比原模型多。

图 33 与图 34 的体素化分辨率差距较大，所以两组图的(f)对比非常的明显。虽然两图的(f)都是由 100x100x100 的降采样分辨率得到的，但图 33 的(f)锯齿状非常的明显，而图 34 的(f)却明显平滑许多。这是因为图 33(f)降采样比例为 1x1x1，图 34(f)为 5x5x5。但如果图 34 的(f)的降采样分辨率为 500x500x500，也即降采样比例为 1x1x1，那么重构的网格也必定会有微小的锯齿。所以可以得出结论：某方向上的降采样比例越高，得到网格的锯齿程度越低。一般来说所有方向上的降采样比例在 5~15 附近为佳，比 5 少的话网格有生成锯齿的倾向，大于 15 的重构效果的改进肉眼难以分辨。

而且还有一点留意到，本文降采样阶段的“降采样”指的是体素的降采样与滤波，所以重构网格的面数理论上确实可以比原模型多。图 35 中的(f)面数比原模型(a)要多三角形数多，网格密度更大，经过顶点焊接以后的渲染效果不错。但可以看到(f)虽然面数更多，但是细节其实是减少了。理论上本文算法不会让新生成的网格比原网格更加平滑与光滑，原网格本身就是模型描述精细度的上限。所以即使面数更多与不能做到曲面细分的效果。

3.5 算法优缺点

3.5.1 优点

1. 本文的 LOD 模型生成算法分成两个阶段进行。一系列 LOD 模型一般有多个，而本文算法的体素化阶段只需要进行一次，后面的网格重构阶段可以重复使用第一阶段的体素化中间表示。这样一来随着生成的 LOD 模型的级数增加，算法的平均时间成本下降。从第二个网格开始，运行耗时的边际成本仅为重构阶段的耗时。所以本文算法生成 n 级精细度模型的总耗时为

$$T = time(\text{体素化}) + \sum_{i=1}^{n} time(\text{三角化}(i))$$

2. 可以在算法速度与网格生成质量之间权衡。想要提高速度，可以降低体素化分辨率、降采样分辨率与 EdgeRatio()在每条 MC 立方体边上的采样率 ,但网格质量会有一定下降 ,可能会出现一定程度的锯齿。同样的，可以多花费一点运行时间与内存，提高采样率，然后就可以提高网格重构质量。

3. 生成网格比较均匀，每个三角形面积不会超过 $(\frac{\sqrt{3}}{2} \times MC\text{立方体最长边长})$

4. 目标简化程度高时算法效率较快。

5. 降采样分辨率各方向分量不需要是 2 的幂。

3.5.2 不足之处

1. 生成网格的面数可能比原网格多，但又做不到曲面细分的平滑效果。

2. 本文算法降采样阶段的采样点在空间中的分布均匀，导致在所有地方的采样密度都一样。这就有可能使得某些低曲率部位面数增加，而高曲率部位的面数不足。

3. 由于体素化阶段的原理的局限，本文算法输入的网格模型要求闭合。

4 结语

本文提出一种闭合网格模型的 LOD 模型生成算法，该算法先使用本文提出的方法对给定网格模型进行体素化，得到指定分辨率的体素模型中间表示，然后对体素模型进行一定分辨率的降采样与滤波之后用 Marching Cubes 算法来重构出不同细节程度的三角形模型。体素化算法借鉴了 3D 打印的原理，用一系列平行的面切割模型，用几何求交的方法得到求交轮廓。然后对切割结果（一系列闭合多边形）进行光栅化与扫描线填充，得到体素模型（中间表示）。用三线性插值对二值化体素模型进行定义域延拓之后，得到一个可以在体素空间任意点采样的扩展体素描述函数 $VoxExt(i,j,k)$。基于 $VoxExt(i,j,k)$ 可以进行指定分辨率的降采样，然后用 MC 算法重构出网格。

本文的网格 LOD 模型生成算法有如下特点：可以通过调节体素化分辨率与重采样分辨率，在 LOD 模型生成质量与生成速度之间进行权衡；降采样密度不需要是 2 的幂；生成一系列 LOD 网格模型时，随着 LOD 级别数的增加平均耗时会逐渐下降（体素化只需要进行一次）；生成的三角形网格比较均匀；能在其他场景应用，例如 CT 切层扫描数据、3D 扫描数据的网格 LOD 模型生成、近似的网格布尔运算。

但仍有更多工作需要去做。例如网格重构阶段的效率仍可优化，很可能经常对同一条 MC 立方体的边用过 4 次 $EdgeRatio(V_1,V_2)$，因为一条立方体边会被 4 个相邻立方体共享。可以把考虑计算过的结果存在一个哈希表里面，需要用到时便去查表。还有就是降采样阶段的采样点太过均匀，使得采样点不总是能合理地分布。以后的工作可以考虑给每个采样点加一点位置的偏移量，使得在高曲率区域采样点更多，使得局部重构效果更细致。这就需要一种空间外观重要性的评估标准，以及把 Marching Cubes 移动立方体算法推广成移动六面体算法。

参考文献

[1]Rossignac J, Borrel P. Multi-resolution 3D approximations for rendering complex scenes[J]. Journal of Trauma & Dissociation the Official Journal of the International Society for the Study of Dissociation, 1993, 7(1):5-18.

[2]周昆, 潘志庚, 石教英. 一种新的基于顶点聚类的网格简化算法[J]. 自动化学报, 1999, 25(1): 1-8

[3]Maria-Elena A, Francis S. Mesh Simplification[C]. Computer Graphics Forum. Blackwell Science Ltd, 1996:77-86.

[4]Cignoni P, Montani C, Scopigno R. A comparison of mesh simplification algorithm[J]. Computers & Graphics, 1997, 22(1):37-54.

[5]Kanaya T, Teshima Y, Kobori K I, et al. A topology-preserving polygonal simplification using vertex clustering[C]// International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia 2005, Dunedin, New Zealand, November 29 - December. DBLP, 2005:117-120.

[6]吴耕宇, 潘懋, 郭艳军. 利用几何求交实现三角网格模型快速体素化[J]. 计算机辅助设计与图形学学报, 2015, 27(11):2133-2141.

[7]吴晓军, 刘伟军, 王天然. 基于八叉树的三维网格模型体素化方法[J]. 图学学报, 2005, 26(4):1-7.

[8]Lorensen W E, Cline H E. Marching cubes: A high resolution 3D surface construction algorithm[J]. Acm Siggraph Computer Graphics, 1987, 21(4):163-169.

[9]百度百科-体素. https://baike.baidu.com/item/%E4%BD%93%E7%B4%A0/8945761?fr=aladdin.2017.8.27

[10]谭皓, 杨忠, 李玉峰,等. 基于硬件加速的真三维显示体素化算法[J]. 南京航空航天大学学报(英文版), 2006, 23(1):59-64.

[11]毕林, 王李管, 陈建宏,等. 三维网格模型的空间布尔运算[J]. 华中科技大学学报(自然科学版), 2008, 36(5):82-85.

[12]wikipedia-Rasterization.[EB/OL] https://en.wikipedia.org/wiki/Rasterisation ,2017-8-30

[13]甘泉. 通用扫描线多边形填充算法[J]. 计算机工程与应用, 2000, 36(2):57-59.

[14]陈元琰, 陈洪波. 一种基于链队列的种子填充法[J]. 广西师范大学学报(自然科学版), 2003, 21(3):30-33.

[15]王培珍, 许睿. 任意多边形填充新算法[J]. 安徽工业大学学报(自科版), 2009, 26(4):405-408.

[16]TUTORIAL:IMAGE RESCALING.[EB/OL] https://clouard.users.greyc.fr/Pantheon/experiments/rescaling/index-en.html ,2017-9-4

[17]Huang J, Yagel R, Filippov V, et al. An accurate method for voxelizing polygon meshes[C]// IEEE, 1998:119-126.

[18]Bi L, Wang L, Chen. Spacial Boolean operations of 3D mesh model[J]. Journal of Huazhong University of Science & Technology, 2008.

[19]Kobbelt L P, Botsch M, Schwanecke U, et al. Feature sensitive surface extraction from volume data[C]// Conference on Computer Graphics and Interactive Techniques. ACM, 2001:57-66.

[20]Ju T, Losasso F, Schaefer S, et al. Dual contouring of hermite data[J]. Acm Transactions on Graphics, 2002, 21(3):339-346.