深蓝学院
shenlanxueyuan.com

基于图像的三维模型重建

——点云到网格的重建

主讲人 隋博

课程内容

- ✓ **三维模型的表述方式**
  - ✓ 边界表述法
  - ✓ 空间划分法
- ✓ **德劳内三角剖分(Delaunay Triangulation)**
  - ✓ 空圆特性
  - ✓ 德劳内重建算法流程
- ✓ **基于隐函数的三维模型重建**
  - ✓ 符号距离场(Signed Distance Field)与隐函数(Implicit Function)
  - ✓ 均匀划分(Grid)与八叉树(Octree)
  - ✓ Marching Cube算法生成表面网格

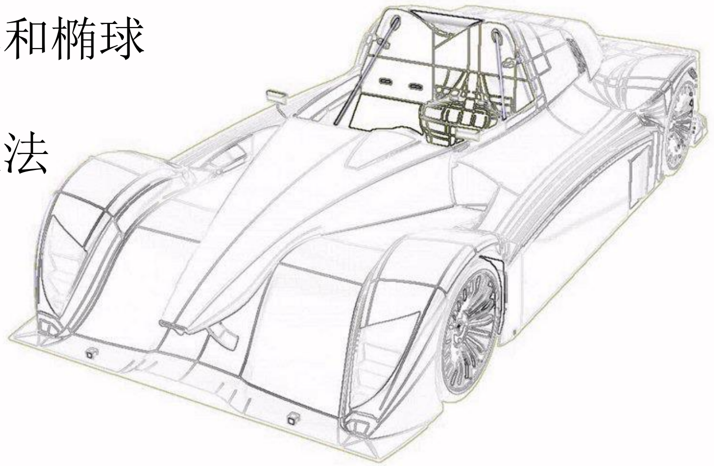
三维模型的表述方式

- 三维模型
  - 边界表述法 Boundary Representation, B-reps
    - 多面体表述法
    - 曲面表述法
      - 参数平面表述法 Parametric Surface
      - 隐式平面表述法 Implicit Surface
  - 空间划分法 Space-partitioning Representations
    - 构造体素法 Constructive Solid Geometry
    - 八叉树 (Octree)
    - 二分空间法Binary Space-partitioning)

三维模型的表述方式

边界表述法(Boundary Representation)

- 将三维物体描述成一组表面，该表面将物体的内部和外部分离开
- 有较多的关于面、边、点及其相互关系的信息，便于对模型进行几何运算和操作
- 可以精确地表示简单规则的物体，例如多面体和椭球
- 可分为基于多面体的表述法和基于曲面的表述法

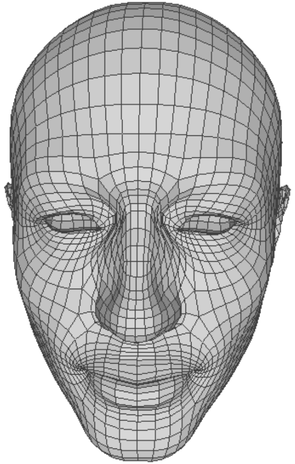


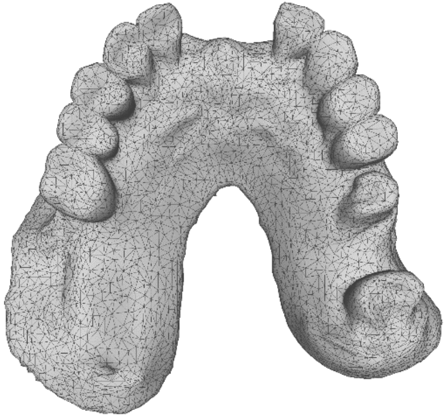
三维模型的表述方式

边界表述法-多面体表述法

将物体表面表述成一组封闭物体空间的多边形，其中最常用的是三角形和四边形，其中三角形表示物体的表面也叫做三角剖分。三角剖分具有以下特性

- 稳定性强
- 能够表示各种形状的三维模型
- 有助于恢复模型的表面细节
- 要求点云稠密且分布均匀

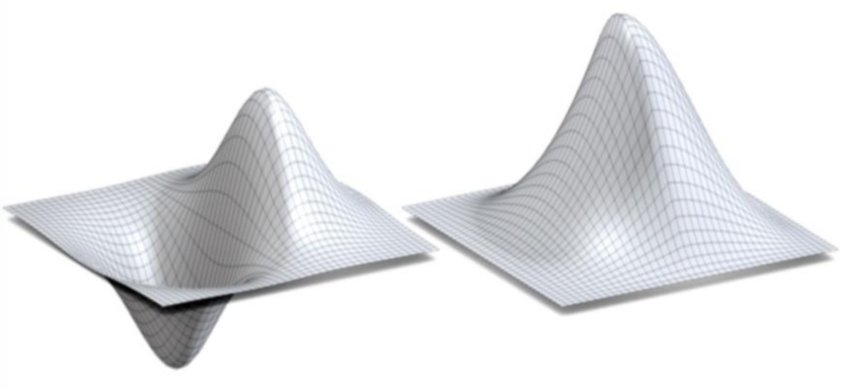




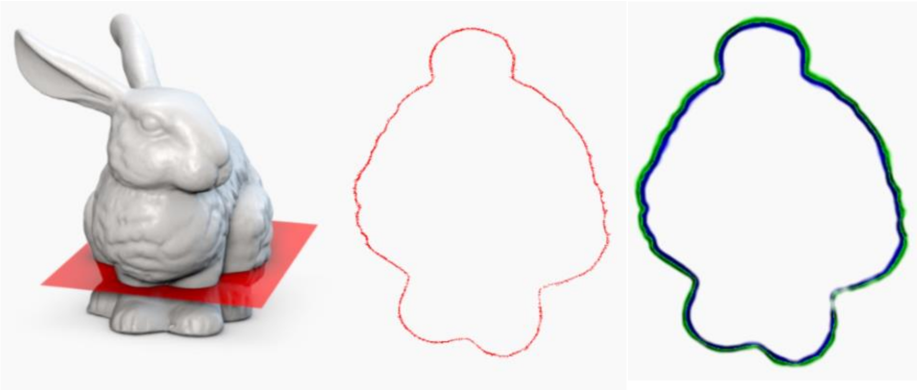
三维模型的表述方式

边界表述法-曲面表述法

将物体表面表述成一组参数或者非参数化的曲面。曲面函数能精确地表示重建的物体模型



参数曲面

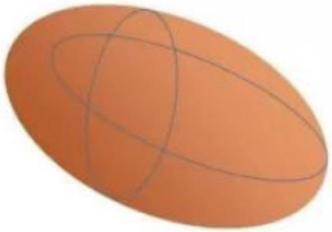


隐式曲面

三维模型的表述方式

边界表述法-参数曲面

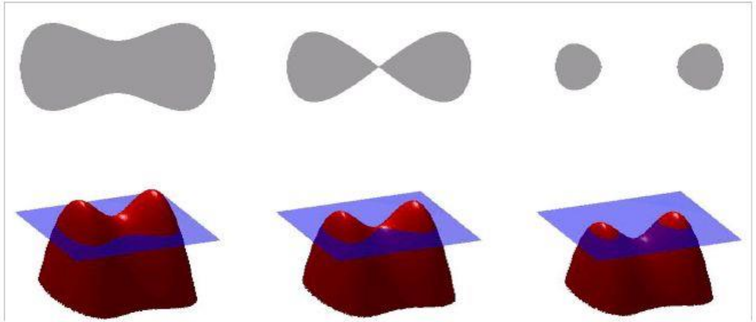
参数曲面形式为$z = f(x, y)$，常用来表述场景中规则的物体，如球、椭球、圆环等。常用的参数曲面有二次曲面、多项式曲面和样条曲面等。




三维模型的表述方式

边界表述法-隐式曲面

隐式曲面通过函数$f(x,y,z)$的零水平集，即$\{x,y,z|f(x,y,z)=0\}$所代表的曲面来确定物体表面的空间位置。相比较于参数平面，隐式平面更加灵活，适合描述结构比较复杂的物体。



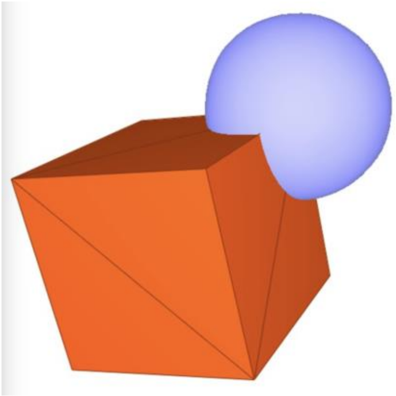
$$\frac{\partial \varphi}{\partial t} = \nu |\nabla \varphi|$$

$$\Gamma = \{(x,y) \mid \varphi(x,y) = 0\}$$

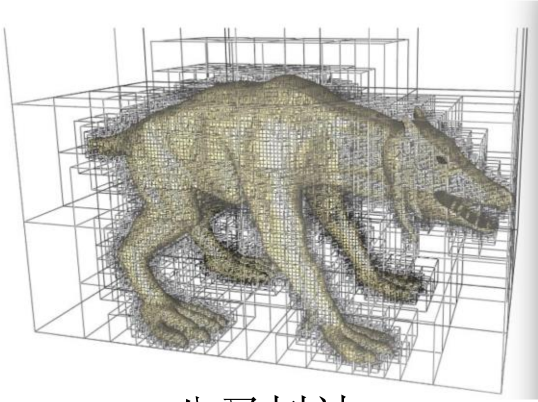
三维模型的表述方式

空间划分法

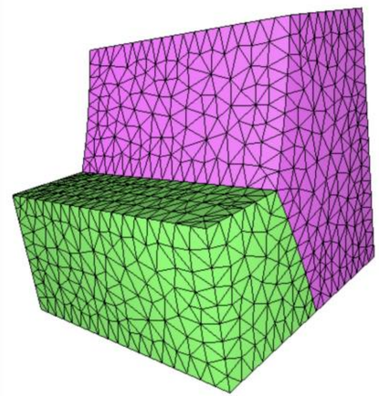
将物体内部的空间划分成细小、不重叠的连续实体来描述物体的形状，常用方法有构造体素法,八叉树法和二分空间法



构造体素法



八叉树法

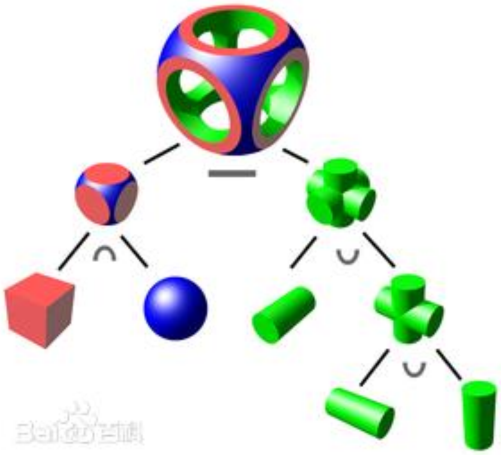


二分空间法

三维模型的表述方式

空间划分法-构造体素法

通过对一些基本元素(如四面体、圆柱体、圆锥、球体或者带有样条曲面的刚体)进行加、减、并集和交集等组合运算生成新的物体。

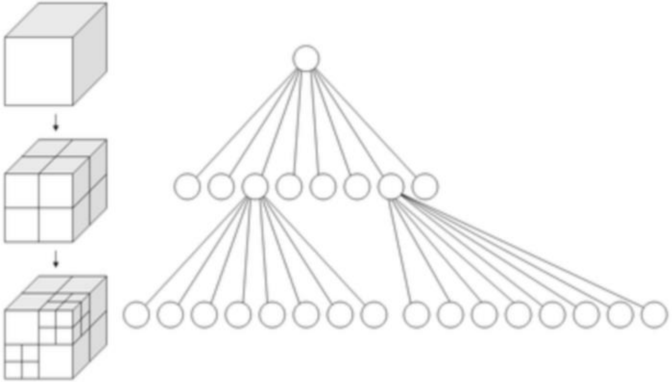


- 操作简单，便于实现
- 只能用来表述结构较为简单的实体，无法用于形状复杂或者表面细节丰富的物体
- 由于信息简单，这种数据结构无法存贮物体最终的详细信息，例如边界、顶点的信息

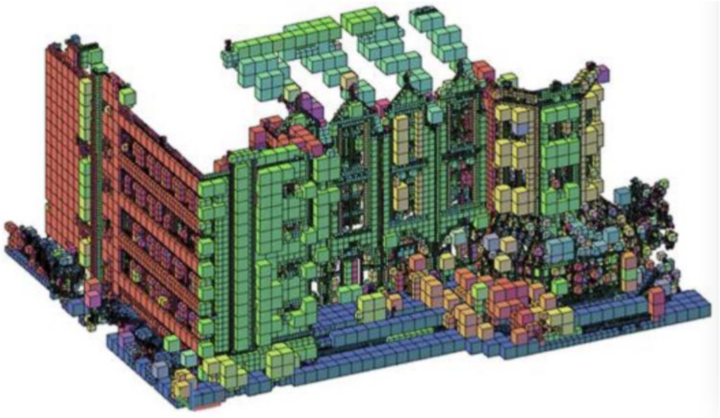
三维模型的表述方式

空间划分法-八叉树法

利用分层的树结构将要表述的物体建造一个树结构，树节点对用空间中一块特定的区域(根节点是包含整个物体的正方体边界区域)。



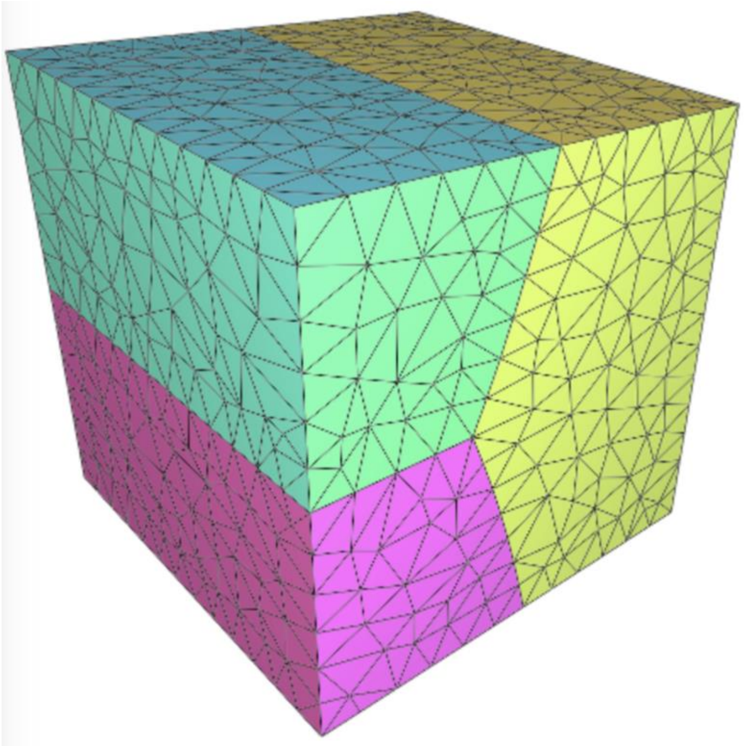
从根节点开始，包含物体的节点将被均匀地划分成八个字节点。这种迭代进行直到满足终止条件为止



三维模型的表述方式

空间划分法-二分空间法

与八叉树结构表述法类似，都是对空间进行逐步划分。不同之处在于二分空间法每一步都将空间划分成两部分，且划分平面的位置和方向根据物体的空间分布随时调整，因此增加了表述的灵活性。



三维模型的表述方式

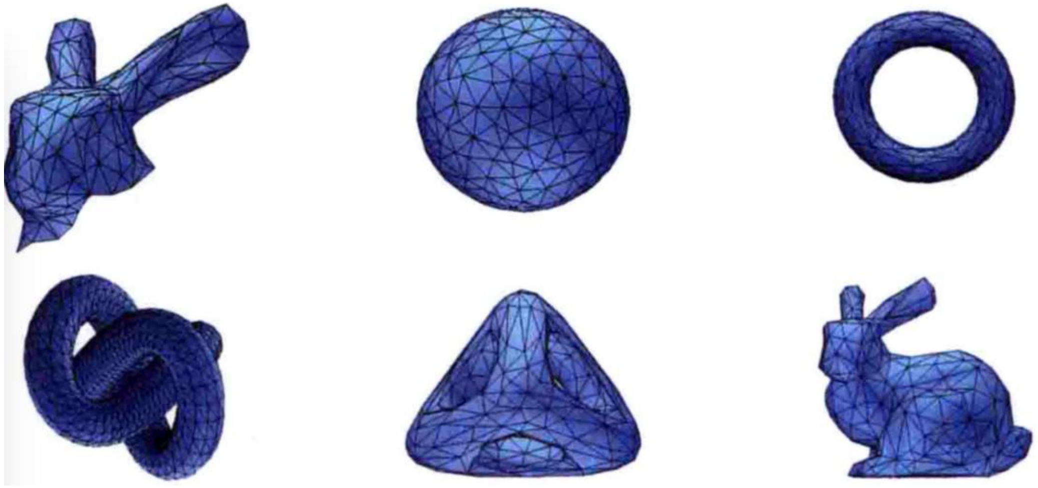
最常用的三维模型表述方式-三角网格

计算机图形学中的三维网格处理
绝大部分都是基于三角网格

三角网格的基本结构：

- 顶点 (Vertex)
- 面片 (Facet)
- 边　(Edge)

点线和面上的各种属性：
颜色(Color), 法向量(Normal),
纹理坐标(Texture Coordinate)



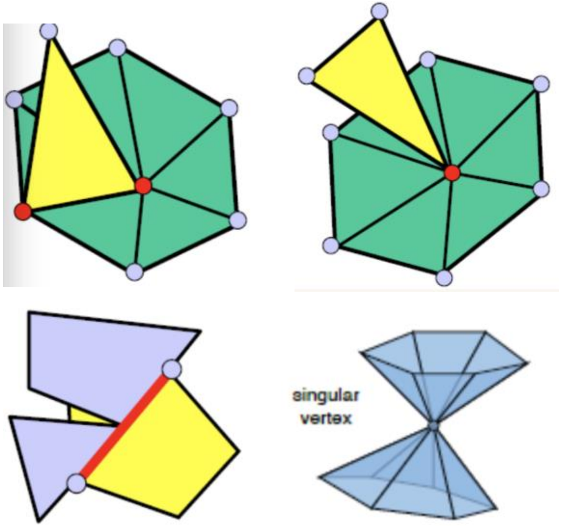
三维模型的表述方式

流形(Manifold Mesh)与非流形(Non-manifold Mesh)

流形的定义：

三角网格曲面中大多数算法是基于流形网格的，其定义如下：

1）一条边由一个或两个面片共享
2）一个网格顶点的一环邻域三角片构成一个闭合或者开放大扇面



非流形的几种情况

三维模型的表述方式

深蓝学院
shenlanxueyuan.com

三角网格常用的数据结构—共享顶点(Shared Vertex)

共享顶点的数据结构主要包含2个部分：

1）顶点的坐标数组

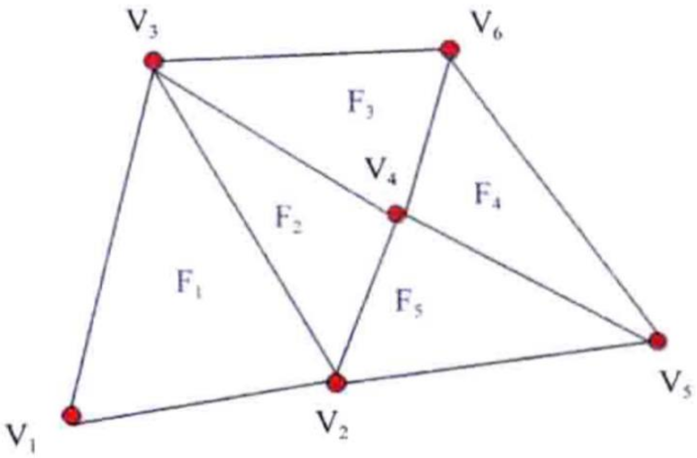
2）三角形面的数组（每个面片只存储相应的顶点数组的序号）

顶点
顶点坐标
[40 5 20]
[10 20 30]
[10 4 3]
...

三角形		
顶点索引	顶点索引	顶点索引
2	1	3
...		

三维模型的表述方式

三角网格常用的数据结构—共享顶点(Shared Vertex)



三角网格模型

顶点	
V_1	[40 5 20]
V_2	[10 20 30]
V_3	[10 4 3]
…	

面	三角形		
F_1	2	1	3
…			

三维模型的表述方式

三角网格常用的数据结构—共享顶点(Shared Vertex)

缺点:

可以表示点之间的连接关系(Connectivity)，但是没有局部之间的邻接关系(Neighborhood),例如从一个顶点到与其相邻的面片，因此在很多局部操作上速度和效率低

三维模型的表述方式

半边数据结构 (Half-Edge Data Structure)

- 一边为中心的数据结构
- 存储三维模型所有顶点、边和面的数据以及相邻接关系的信息
- 利用半边表示边的方向

- 局部操作速度快
- 只能处理流形(Manifold)模型

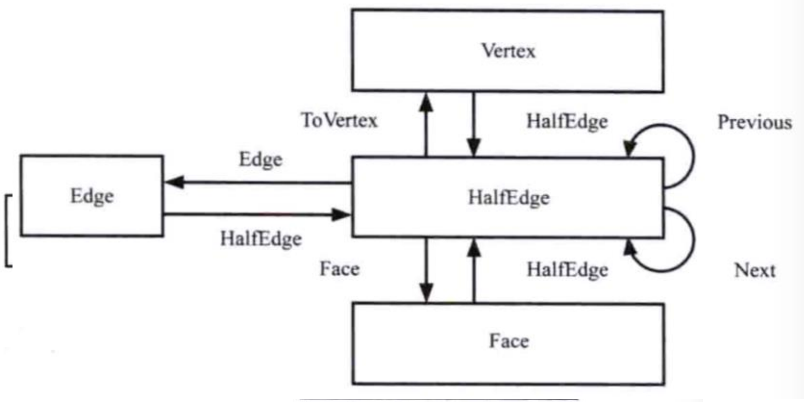
三维模型的表述方式

半边数据结构 (Half-Edge Data Structure)

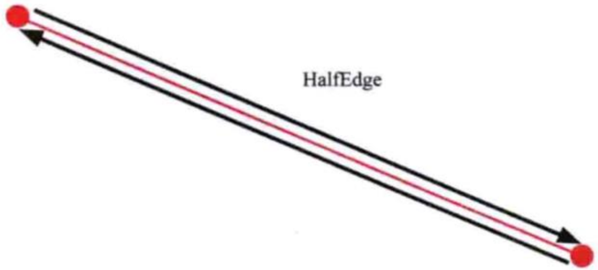
半边结构的5个组成部分：

- 顶点(Vertex)
- 半边(HalfEdge)
- 边(Edge)
- 面(Face)
- 模型(Mesh)

半边是有方向(Oriented)的边,而边是没有方向(Non-Oriented)的边，一个没有方向的边可以视为两个方向相反的半边



半边数据结构

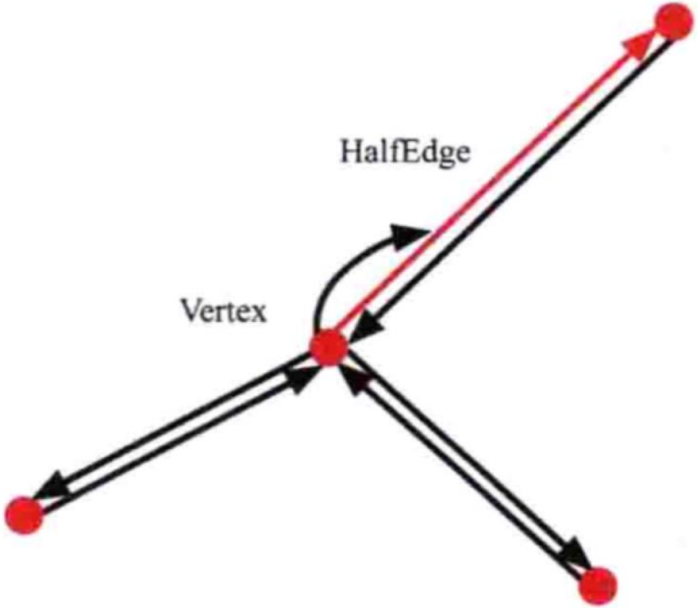


三维模型的表述方式

半边数据结构 (Half-Edge Data Structure)

顶点需要保存的信息：

- 以此顶点为源点的半边（随机选取一条）

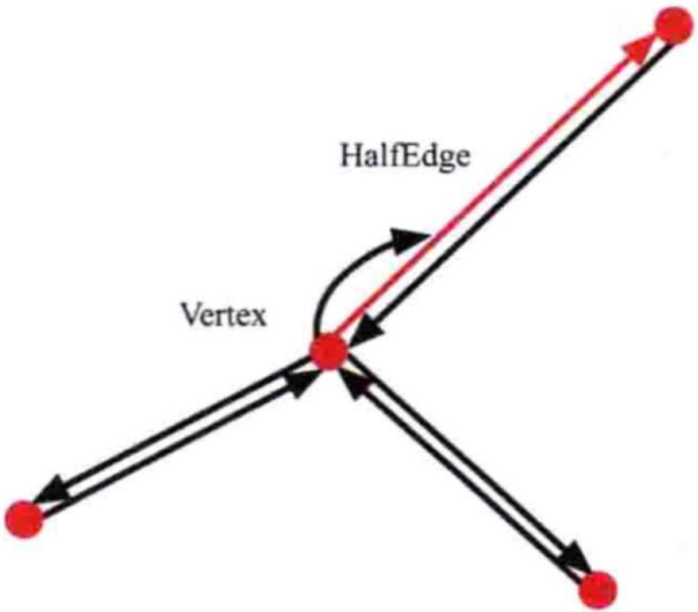


三维模型的表述方式

半边数据结构 (Half-Edge Data Structure)

顶点需要保存的信息：

- 以此顶点为源点的半边（随机选取一条）

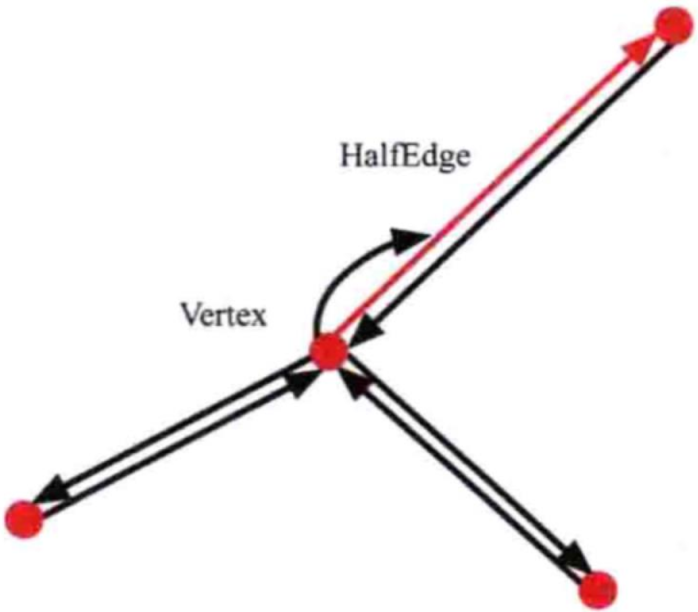


三维模型的表述方式

半边数据结构 (Half-Edge Data Structure)

半边需要保存的信息：

- 目的顶点（Target Vertex）
- 半边左侧面
- 前一个（Prev）半边
- 下一个（Next）半边
- 孪生（Twin）半边

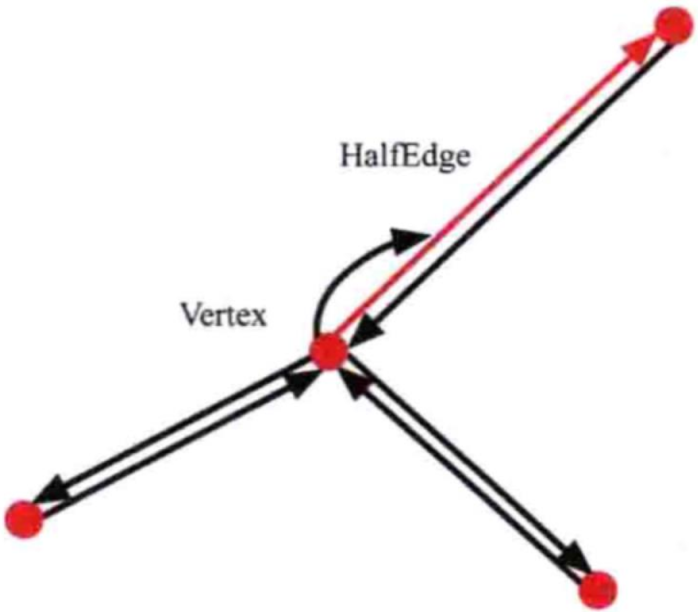


三维模型的表述方式

半边数据结构 (Half-Edge Data Structure)

边存储的信息：

- 任意一条半边

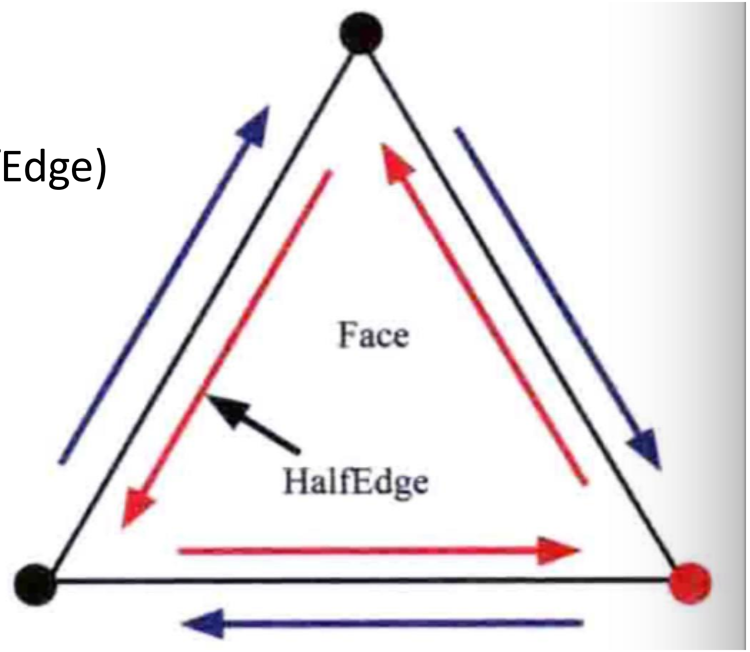


三维模型的表述方式

半边数据结构 (Half-Edge Data Structure)

面存储的信息：

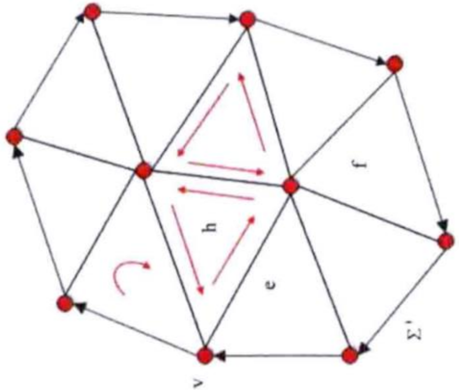
- 一条和此面相邻的半边(Adjacent HalfEdge)



三维模型的表述方式

半边数据结构 (Half-Edge Data Structure)

Mesh存储的信息：



顶点列表

边列表

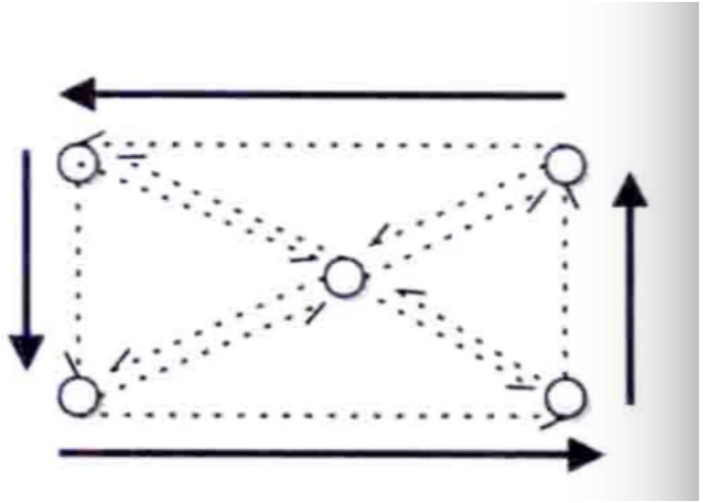
半边列表

面列表

三维模型的表述方式

半边数据结构 (Half-Edge Data Structure)

示例：找顶点的邻域顶点



三维模型的表述方式

半边数据结构 (Half-Edge Data Structure)

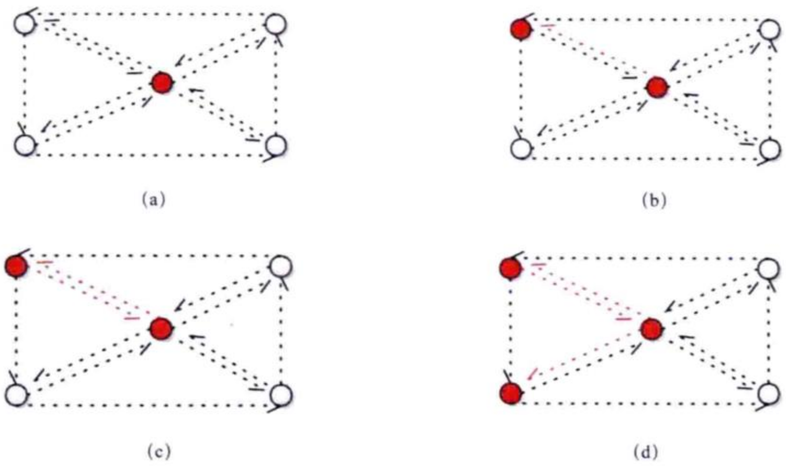
示例：找顶点的邻域顶点

a）确定给顶点

b）通过该点可以直接得到此点的一个相邻的半边

c）通过该半边便可以确定一个相邻的顶点,以及此半边下一个相邻的半边

d）重复上述过程可以得到所有的相邻顶点



(a) (b) (c) (d)

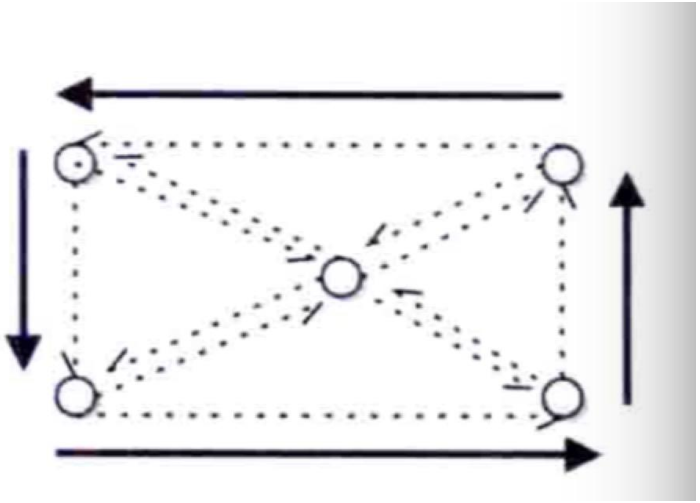
三维模型的表述方式

半边数据结构 (Half-Edge Data Structure)

思考：

1）如何通过半边结构遍历顶点所有的相邻的面？

2） 如果通过半边结构遍历面片的所有顶点？



课程内容

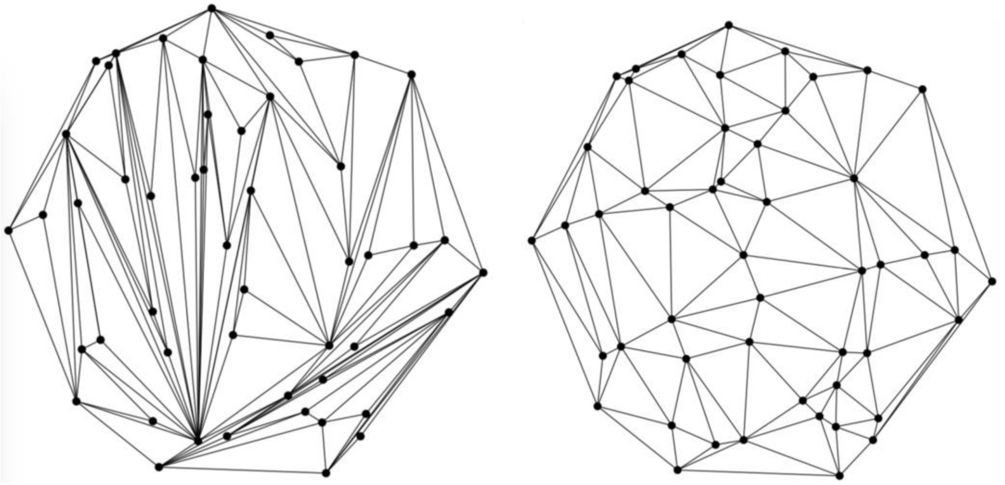
- ✓ 三维模型的表述方式
  - ✓ 边界表述法
  - ✓ 空间划分法
- ✓ **德劳内三角剖分(Delaunay Triangulation)**
  - ✓ 空圆特性
  - ✓ 德劳内重建算法流程
- ✓ 基于隐函数的三维模型重建
  - ✓ 符号距离场(Signed Distance Field)与隐函数(Implicit Function)
  - ✓ 均匀划分(Grid)与八叉树(Octree)
  - ✓ Marching Cube算法生成表面网格

德劳内三角剖分

德劳内三角剖分 Delaunay Triangulation

点集P的德劳内三角剖分满足满足任意P内任意一个点都不在P 内任意一个三角面片的外接圆内。

德劳内三角剖分最大化三角面片内三角形的最小角

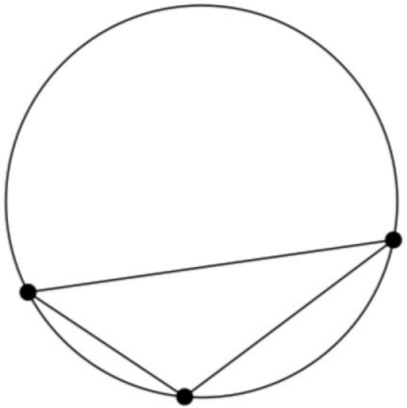


非德劳内三角剖分与非德劳内三角剖分的对比

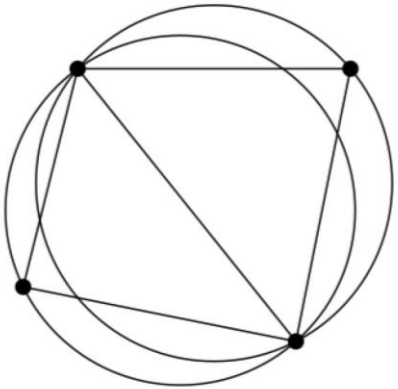
德劳内三角剖分

空圆特性

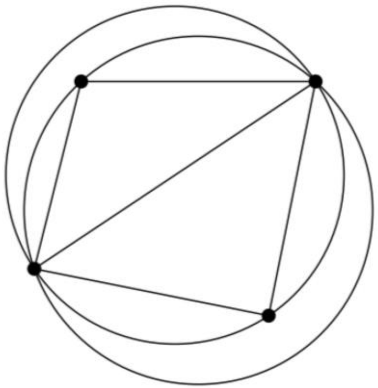
任意3个点的外接圆不包含第4个点，即任意3个点的外接圆是空的



三角形的外接圆



满足空圆特性的三角剖分

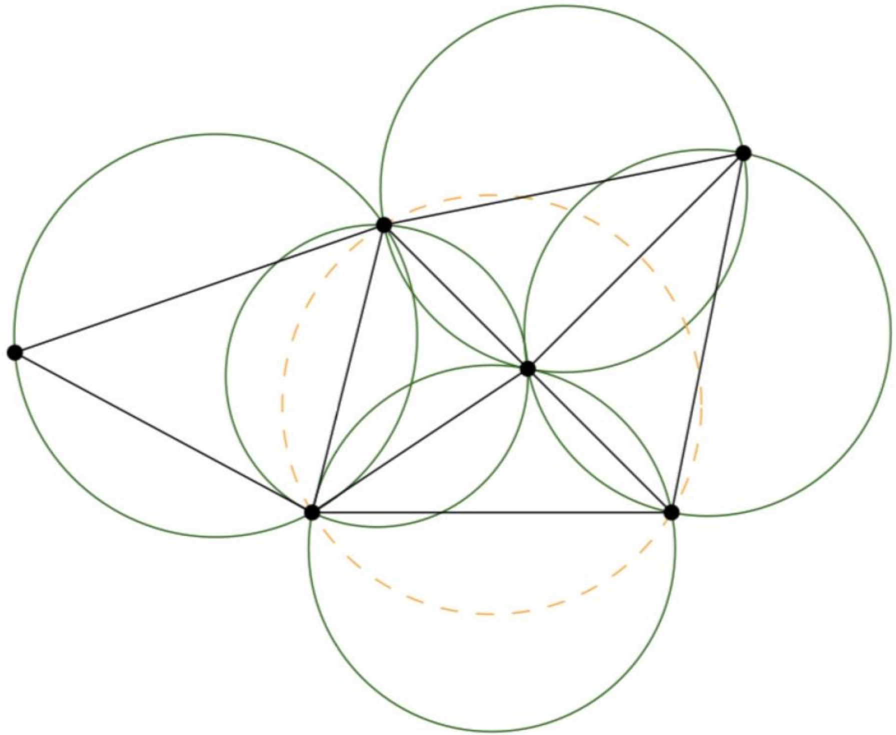


不满足空圆特性的三角剖分

德劳内三角剖分

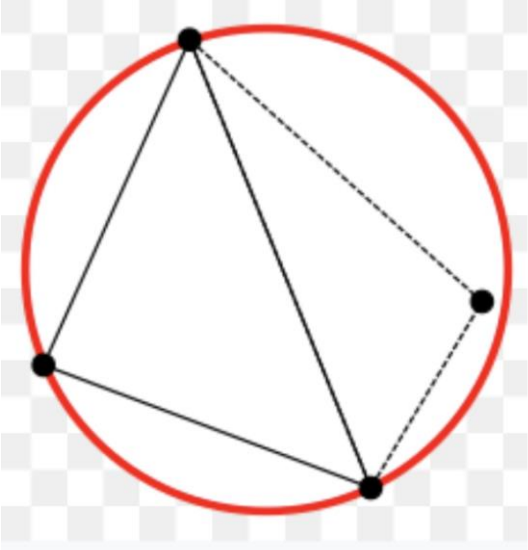
空圆特性

德劳内三角剖分中所有的
三角形都满足空圆特性



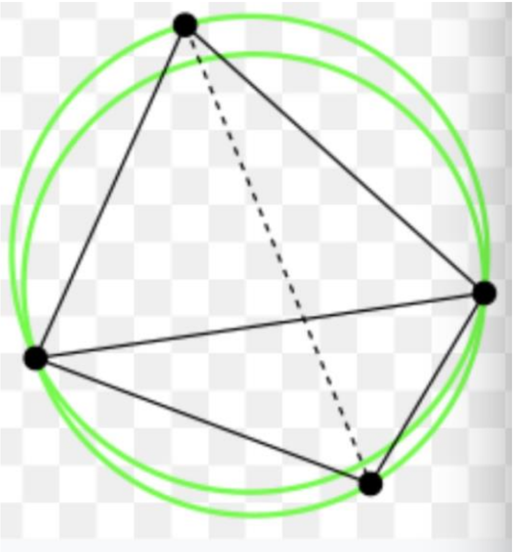
德劳内三角剖分

Lawson Flip Algorithm



不满足空圆特性

翻转（Flip）



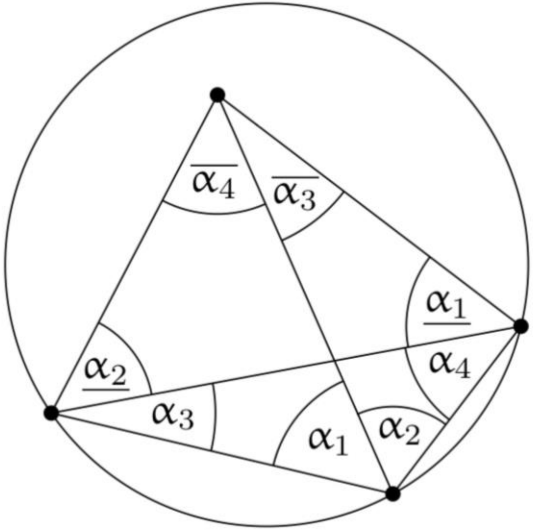
满足空圆特性

德劳内三角剖分

最大化最小角

翻转前的内角：$\alpha_1 + \alpha_2, \quad \alpha_3, \quad \alpha_4, \quad \underline{\alpha_1}, \quad \underline{\alpha_2}, \quad \overline{\alpha_3} + \overline{\alpha_4}$

翻转后的内角：$\alpha_1, \quad \alpha_2, \quad \overline{\alpha_3}, \quad \overline{\alpha_4}, \quad \underline{\alpha_1} + \alpha_4, \quad \underline{\alpha_2} + \alpha_3$



翻转前后的三角形内角

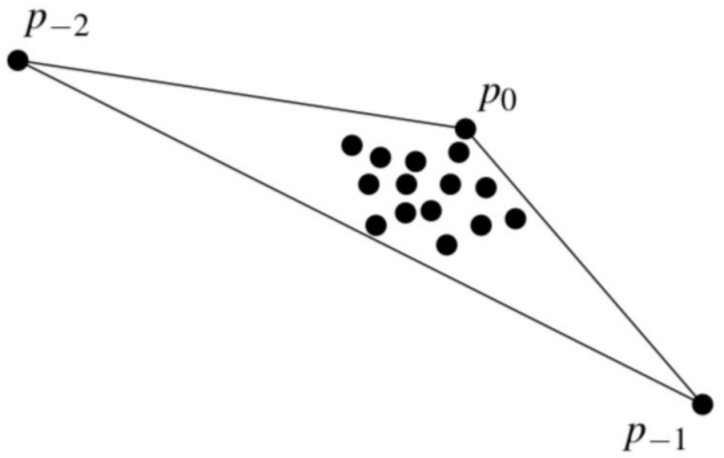
$$
\begin{aligned}
\alpha_1 &> \underline{\alpha_1}, \\
\alpha_2 &> \underline{\alpha_2}, \\
\overline{\alpha_3} &> \alpha_3, \\
\overline{\alpha_4} &> \alpha_4, \\
\underline{\alpha_1} + \alpha_4 &> \alpha_4, \\
\underline{\alpha_2} + \alpha_3 &> \alpha_3.
\end{aligned}
$$

每一次翻转都是在增大三角剖分的最小角

德劳内三角剖分

一种增量的德劳内三角剖分算法

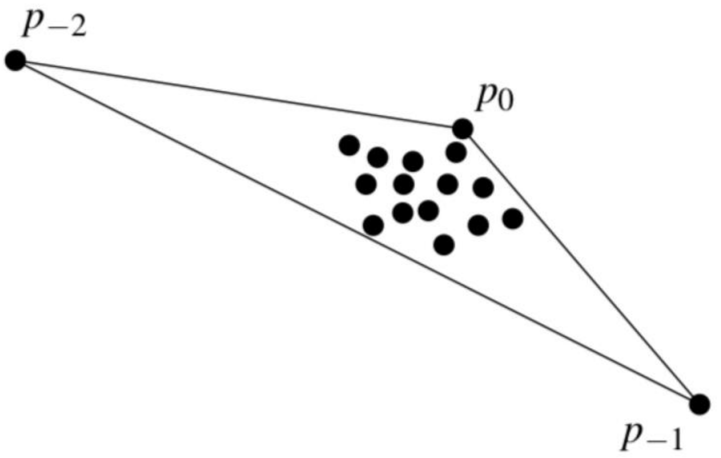
Step 1 添加两个足够远的点p_{-1}, p_{-2}
以保证p_{-1}, p_{-2} 位于所有三角形
的外接圆外



德劳内三角剖分

一种增量的德劳内三角剖分算法

Step 1 添加两个足够远的点p_{-1}, p_{-2} 以保证p_{-1}, p_{-2} 位于所有三角形的外接圆外。三角形$p_0 p_{-1} p_{-2}$ 包含所有的点。初始的时候仅有一个空的三角形。

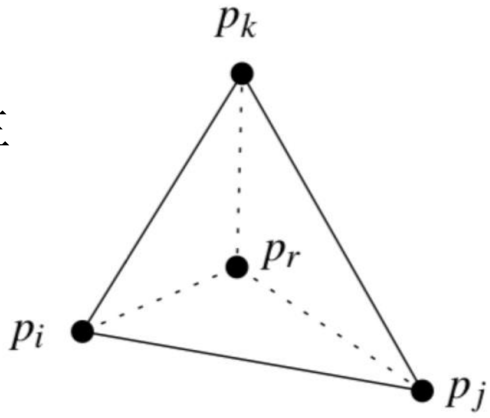


德劳内三角剖分

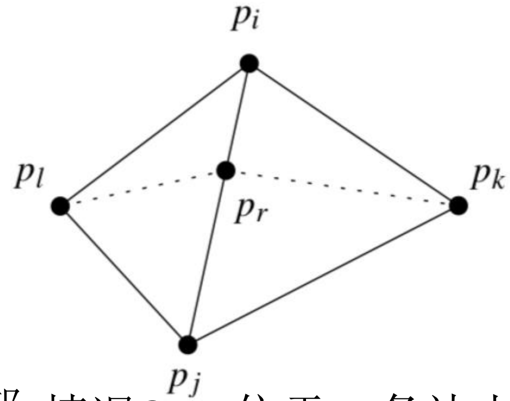
一种增量的德劳内三角剖分算法

Step 2 依次添加新的点p_r

第一种情况： 分别连接p_rp_i, p_rp_j, p_rp_k, 分别检查与三角形$p_rp_ip_k$，$p_rp_jp_k, p_rp_ip_k$相邻的三角形的空圆特性



情况1：p_r位于三角形$p_ip_j\ p_k$内部



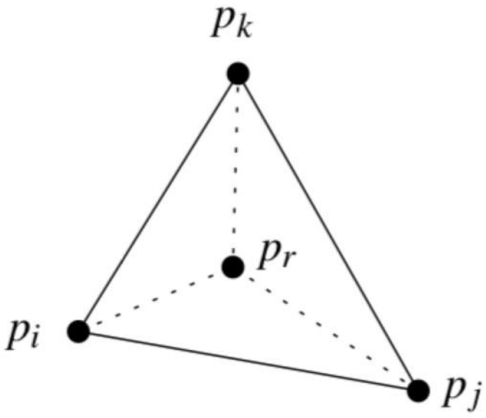
情况2: p_r位于一条边上

德劳内三角剖分

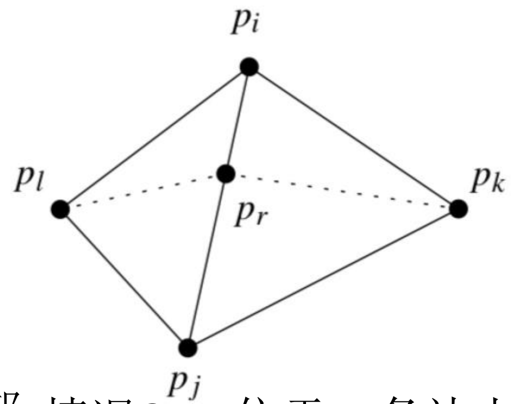
一种增量的德劳内三角剖分算法

Step 2 依次添加新的点p_r

第二种情况： 分别连接$p_r p_l, p_r p_k$, 分别检查与三角形$p_r p_j p_k$, $p_r p_j p_l$, $p_r p_l p_i$, $p_r p_k p_i$相邻的三角形的空圆特性



情况1：p_r位于三角形$p_i p_j\ p_k$内部



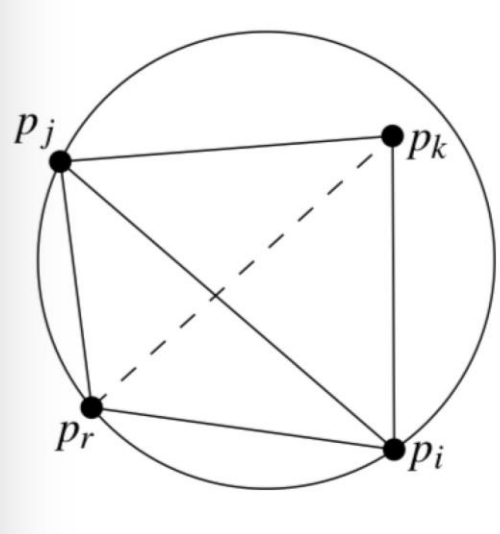
情况2: p_r位于一条边上

德劳内三角剖分

一种增量的德劳内三角剖分算法

Step 2 依次添加新的点p_r

检测相关三角形的空圆特性



德劳内三角剖分

一种增量的德劳内三角剖分算法

Algorithm DELAUNAYTRIANGULATION(P)
Input. A set P of $n+1$ points in the plane.
Output. A Delaunay triangulation of P.

1. Let p_0 be the lexicographically highest point of P, that is, the rightmost among the points with largest y-coordinate.
2. Let p_{-1} and p_{-2} be two points in $\mathbb{R}^2$ sufficiently far away and such that P is contained in the triangle $p_0 p_{-1} p_{-2}$.
3. Initialize $\mathcal{T}$ as the triangulation consisting of the single triangle $p_0 p_{-1} p_{-2}$.
4. Compute a random permutation $p_1, p_2, \ldots, p_n$ of $P \setminus \{p_0\}$.
5. **for** $r \leftarrow 1$ **to** n
6.   **do** (∗ Insert p_r into $\mathcal{T}$: ∗)
7.   Find a triangle $p_i p_j p_k \in \mathcal{T}$ containing p_r.
8.   **if** p_r lies in the interior of the triangle $p_i p_j p_k$
9.   **then** Add edges from p_r to the three vertices of $p_i p_j p_k$, thereby splitting $p_i p_j p_k$ into three triangles.
10.   LEGALIZEEDGE($p_r, \overline{p_i p_j}, \mathcal{T}$)
11.   LEGALIZEEDGE($p_r, \overline{p_j p_k}, \mathcal{T}$)
12.   LEGALIZEEDGE($p_r, \overline{p_k p_i}, \mathcal{T}$)
13.   **else** (∗ p_r lies on an edge of $p_i p_j p_k$, say the edge $\overline{p_i p_j}$ ∗)
14.   Add edges from p_r to p_k and to the third vertex p_l of the other triangle that is incident to $\overline{p_i p_j}$, thereby splitting the two triangles incident to $\overline{p_i p_j}$ into four triangles.
15.   LEGALIZEEDGE($p_r, \overline{p_i p_l}, \mathcal{T}$)
16.   LEGALIZEEDGE($p_r, \overline{p_l p_j}, \mathcal{T}$)
17.   LEGALIZEEDGE($p_r, \overline{p_j p_k}, \mathcal{T}$)
18.   LEGALIZEEDGE($p_r, \overline{p_k p_i}, \mathcal{T}$)
19. Discard p_{-1} and p_{-2} with all their incident edges from $\mathcal{T}$.
20. **return** $\mathcal{T}$

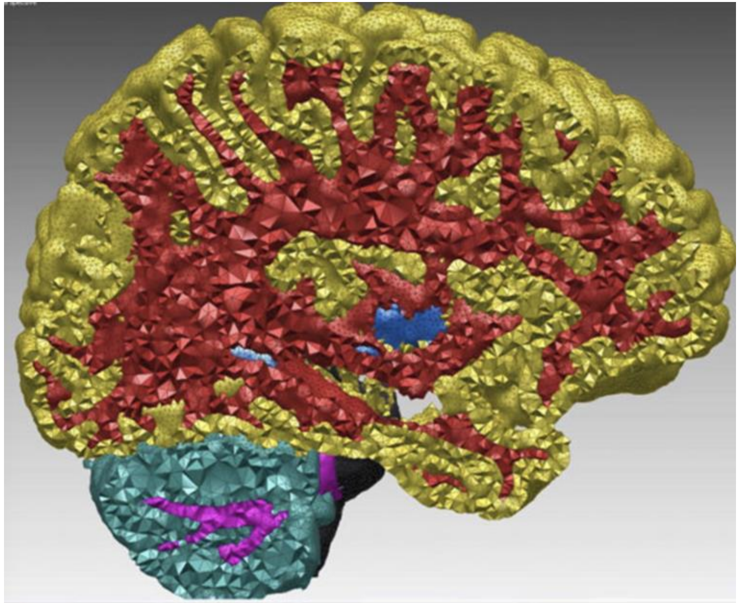
LEGALIZEEDGE($p_r, \overline{p_i p_j}, \mathcal{T}$)

1. (∗ The point being inserted is p_r, and $\overline{p_i p_j}$ is the edge of $\mathcal{T}$ that may need to be flipped. ∗)
2. **if** $\overline{p_i p_j}$ is illegal
3.   **then** Let $p_i p_j p_k$ be the triangle adjacent to $p_r p_i p_j$ along $\overline{p_i p_j}$.
4.   (∗ Flip $\overline{p_i p_j}$: ∗) Replace $\overline{p_i p_j}$ with $\overline{p_r p_k}$.
5.   LEGALIZEEDGE($p_r, \overline{p_i p_k}, \mathcal{T}$)
6.   LEGALIZEEDGE($p_r, \overline{p_k p_j}, \mathcal{T}$)

德劳内三角剖分

三维上的德劳内三角剖分（四面体剖分）

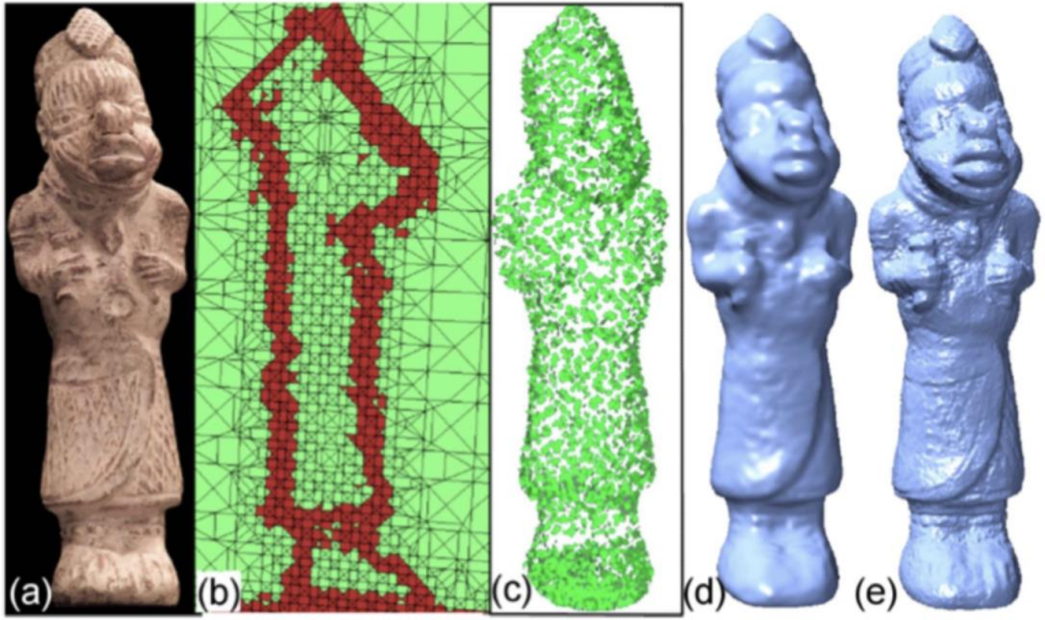
3D 对应的是德劳内四面体剖分，得到的是体数据，无法直接得到面片



德劳内三角剖分

三维上的德劳内三角剖分（四面体剖分）

3D 上德劳内四面体剖分+MRF
优化得到三维网格



课程内容

深蓝学院
shenlanxueyuan.com

- ✓ 三维模型的表述方式
  - ✓ 边界表述法
  - ✓ 空间划分法
- ✓ 德劳内三角剖分(Delaunay Triangulation)
  - ✓ 空圆特性
  - ✓ 德劳内重建算法流程
- ✓ **基于隐函数的三维模型重建**
  - ✓ 符号距离场(Signed Distance Field)与隐函数(Implicit Function)
  - ✓ 均匀划分(Grid)与八叉树(Octree)
  - ✓ Marching Cube算法生成表面网格

基于隐函数的三维模型重建

重建流程

空间划分

−均匀(grid)

−非均匀(octree)

构建符号距离场

−全局

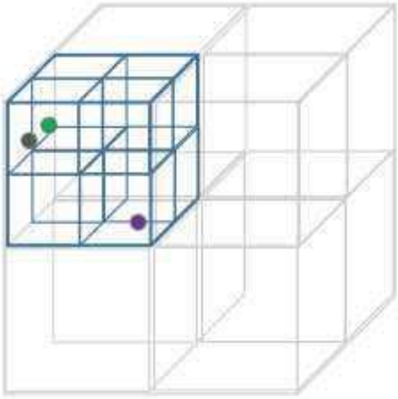
−局部

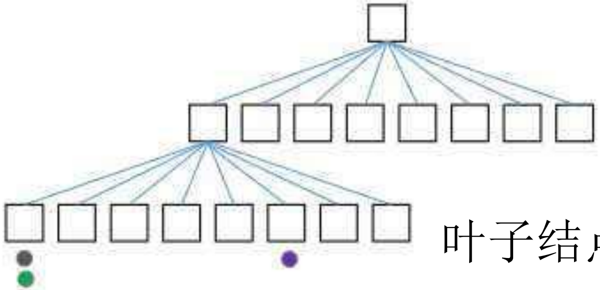
Marching Cube
生成表面

基于隐函数的三维模型重建

空间划分

Grid 空间均匀划分

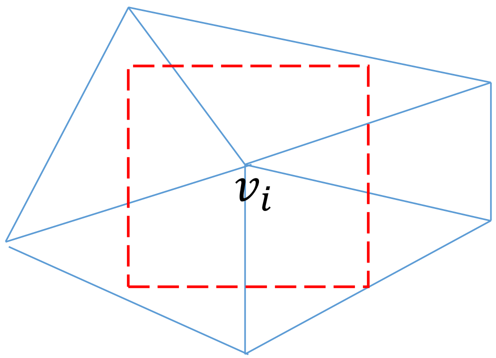




Octree 非均匀划分

基于隐函数的三维模型重建

八叉树深度的确定—点的尺度



点的尺度scale可以定义为顶点到最近邻的平均距离，它反映的点的分辨率

用点的尺度决定八叉树Node的宽度，从而确定深度

$$S_\ell \leq s_i < S_{\ell-1} \quad \Leftrightarrow \quad S_\ell \leq s_i < 2S_\ell$$

⬇

$$1/2\ s_i < S_\ell \leq s_i$$

s_i表示点的尺度 S_l和S_{l-1}分别表示第l和第$l-1$层节点的宽度

且$S_l = 2S_{l-1}$

基于隐函数的三维模型重建

符号距离函数(Signed Distance Function)

$$f(x)=\begin{cases} d(x,\partial\Omega) & if\ x\in\Omega_i \\ -d(x,\partial\Omega) & if\ x\in\Omega_o \end{cases}$$

$f(x)=0$ 表示零势面

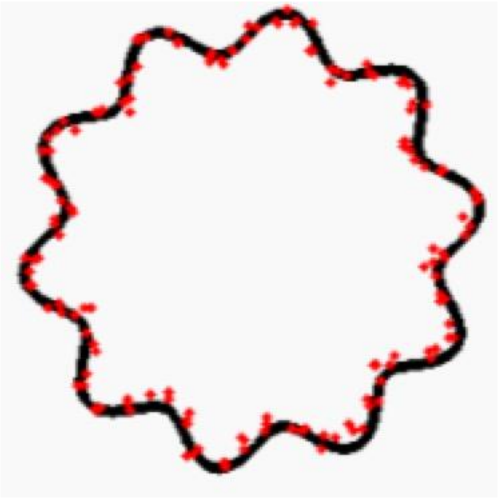
符号距离函数Signed Distance Function是某度量空间X中的一个集合Ω的函数，决定X中任一点到 Ω边界∂Ω的距离，并且由x是在Ω 内还是Ω外确定其SDF的正负号：当x在Ω内时，SDF为正；当x在Ω外时，SDF为负。

基于隐函数的三维模型重建

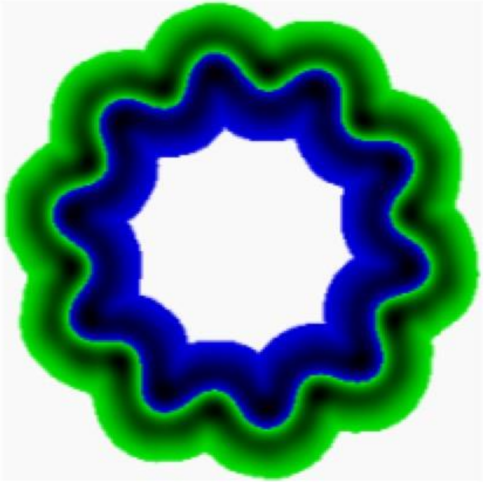
	4	3	3	2	1	1	0	-1	-2
	4	3	2	1	1	0	-1	-2	-3
4	3	2	1	0	0	-1	-2	-3	4
4	3	2	1	0	-1	-2	-3	-4	
4	3	2	1	0	-1	-2	-3	-4	
4	3	2	2	1	0	-1	-2	-3	-4
3	2	1	1	0	-1	-2	-3	-4	
3	2	1	0	-1	-2	-3	-4		
2	1	0	-1	-2	-3	-4			

基于隐函数的三维模型重建

符号距离函数(Signed Distance Function)



输入点云



符号距离场

基于隐函数的三维模型重建

符号距离场的构建

- 对于grid计算每个node的符号距离
- 对于octree计算每个叶子结点的符号距离值

基于隐函数的三维模型重建

符号距离场的构建-全局的方法

把节点处的符号距离值看作是对空间隐函数的采样，采用拟合的方式计算符号距离函数，最常用的方法是在每个节点上建立一个基函数$\emptyset(x)$

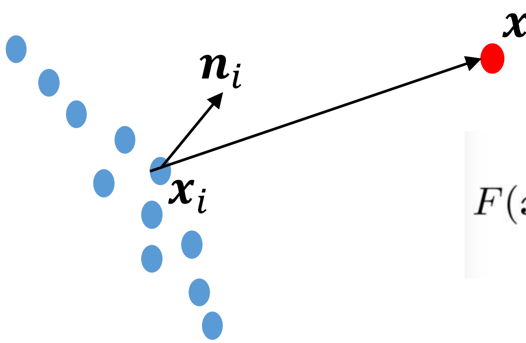
$$s(x) = p(x) + \sum_{i=1}^{N} \lambda_i \phi(|x - x_i|)$$

Carr J C, Beatson R K, Cherrie J B, et al. Reconstruction and representation of 3D objects with radial basis functions[C] 2001:67-76.

基于隐函数的三维模型重建

符号距离场的构建-局部的方法

直接计算每个节点处分符号距离值



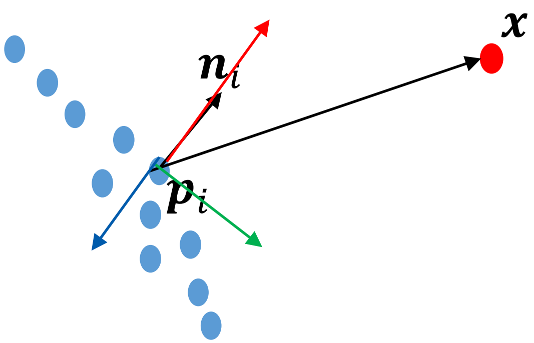
$$F(\boldsymbol{x}) = \frac{\sum_i c_i w_i(\boldsymbol{x}) f_i(\boldsymbol{x})}{\sum_i c_i w_i(\boldsymbol{x})} \qquad W(\boldsymbol{x}) = \sum_i c_i w_i(\boldsymbol{x})$$

最简单的情况

$$f_i(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{p}_i) \cdot \boldsymbol{n}_i \quad w_i(x) = \exp(-||\boldsymbol{x} - \boldsymbol{p}_i||^2)$$

基于隐函数的三维模型重建

符号距离场的构建-局部的方法



$$\boldsymbol{x}_i = R_i \cdot (\boldsymbol{x} - \boldsymbol{p}_i)$$

$$f(\boldsymbol{x}_i) = f_x(x) f_y(y) f_z(z) = \frac{x}{\sigma^4 2\pi} \cdot e^{\frac{-1}{2\sigma^2}(x^2+y^2+z^2)}$$

$$w(\boldsymbol{x}_i) = w_x(x) \cdot w_{yz}(\sqrt{y^2+z^2})$$

$$w_x(x) = \begin{cases} \frac{1}{9}\frac{x^2}{\sigma^2} + \frac{2}{3}\frac{x}{\sigma} + 1 & x \in [-3\sigma, 0) \\ \frac{2}{27}\frac{x^3}{\sigma^3} - \frac{1}{3}\frac{x^2}{\sigma^2} + 1 & x \in [0, 3\sigma) \\ 0 & \text{otherwise} \end{cases}$$

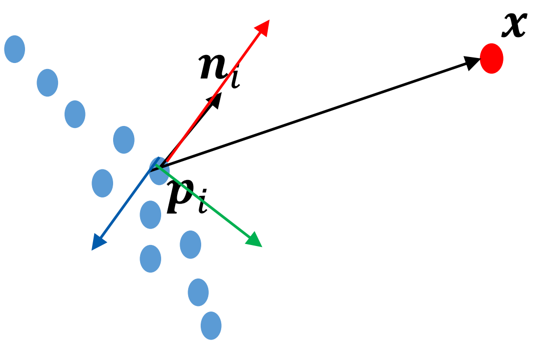
$$w_{yz}(r) = \begin{cases} \frac{2}{27}\frac{r^3}{\sigma^3} - \frac{1}{3}\frac{r^2}{\sigma^2} + 1 & r < 3\sigma \\ 0 & \text{otherwise} \end{cases}$$

$$r = \sqrt{y^2 + z^2}.$$

Simon Fuhrmann, Michael Goesele. Floating scale surface reconstruction[J]. ACM Transactions on Graphics (TOG), 2014, 33(4):1-11.

基于隐函数的三维模型重建

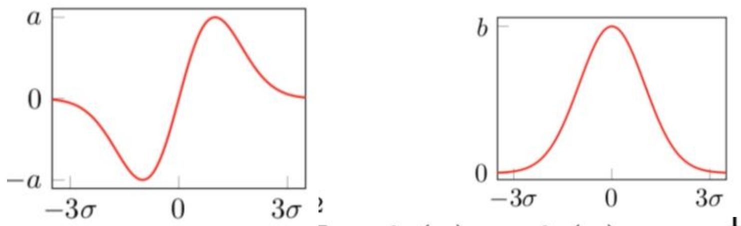
符号距离场的构建-局部的方法



$$\boldsymbol{x}_i = R_i \cdot (\boldsymbol{x} - \boldsymbol{p}_i)$$

符号距离函数

$$f(\boldsymbol{x}_i) = f_x(x) f_y(y) f_z(z) = \frac{x}{\sigma^4 2\pi} \cdot e^{\frac{-1}{2\sigma^2}(x^2+y^2+z^2)}$$

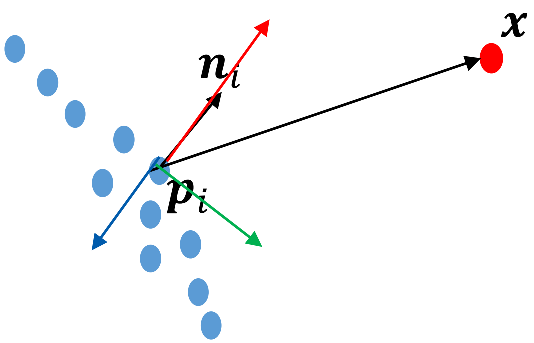


$$f_x(x) = \frac{x}{\sigma^2} e^{\frac{-x^2}{2\sigma^2}} \qquad f_y(x) = f_z(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}}$$

Simon Fuhrmann, Michael Goesele. Floating scale surface reconstruction[J]. ACM Transactions on Graphics (TOG), 2014, 33(4):1-11.

基于隐函数的三维模型重建

符号距离场的构建-局部的方法

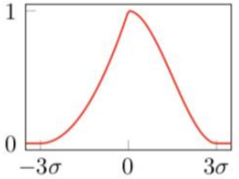


$$\boldsymbol{x}_i = R_i \cdot (\boldsymbol{x} - \boldsymbol{p}_i)$$

权重函数

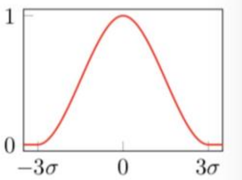
$$w(\boldsymbol{x}_i) = w_x(x) \cdot w_{yz}\left(\sqrt{y^2 + z^2}\right)$$

$$w_x(x) = \begin{cases} \frac{1}{9}\frac{x^2}{\sigma^2} + \frac{2}{3}\frac{x}{\sigma} + 1 & x \in [-3\sigma, 0) \\ \frac{2}{27}\frac{x^3}{\sigma^3} - \frac{1}{3}\frac{x^2}{\sigma^2} + 1 & x \in [0, 3\sigma) \\ 0 & \text{otherwise} \end{cases}$$



$$w_{yz}(r) = \begin{cases} \frac{2}{27}\frac{r^3}{\sigma^3} - \frac{1}{3}\frac{r^2}{\sigma^2} + 1 & r < 3\sigma \\ 0 & \text{otherwise} \end{cases}$$

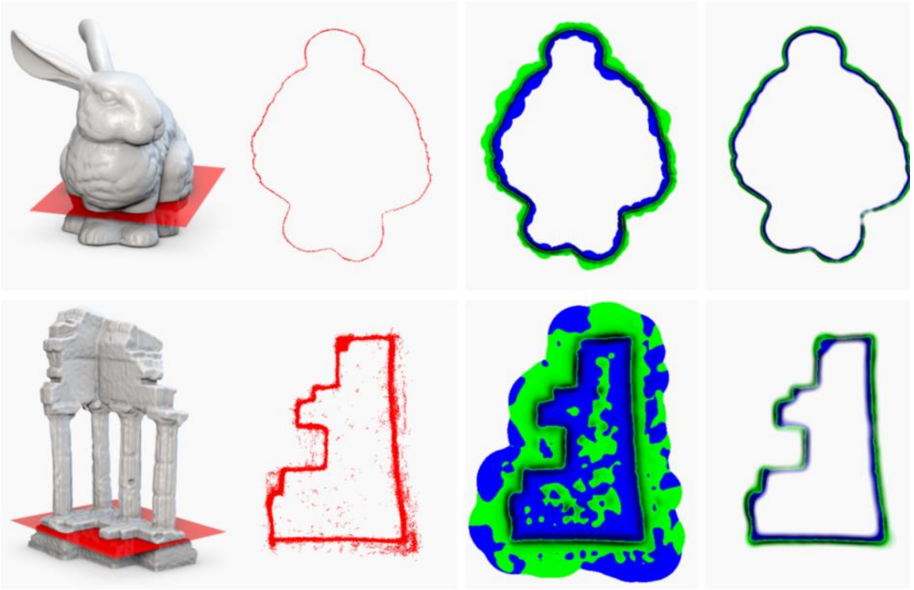
$$r = \sqrt{y^2 + z^2}.$$



Simon Fuhrmann, Michael Goesele. Floating scale surface reconstruction[J]. ACM Transactions on Graphics (TOG), 2014, 33(4):1-11.

基于隐函数的三维模型重建

符号距离场的构建-局部的方法

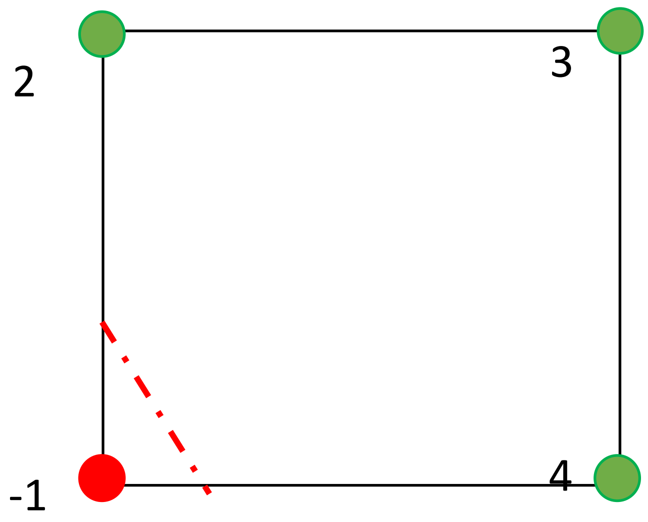


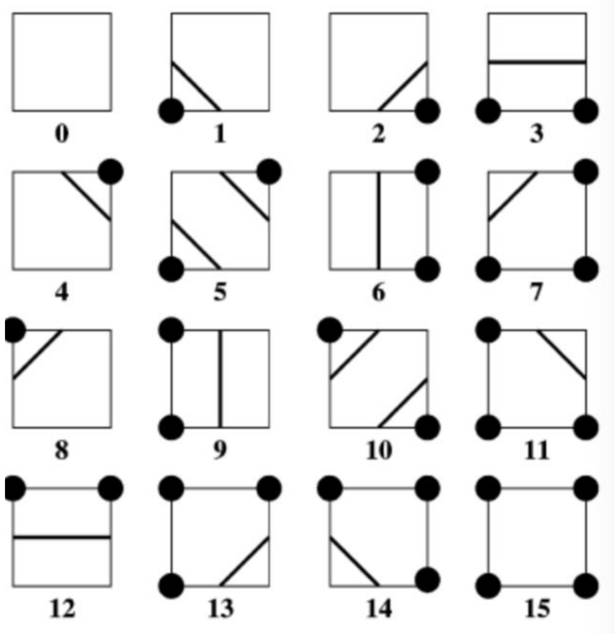
Simon Fuhrmann, Michael Goesele. Floating scale surface reconstruction[J]. ACM Transactions on Graphics (TOG), 2014, 33(4):1-11.

基于隐函数的三维模型重建

Marching Cube生成表面(二维)

根据四边形的4个顶点的符号距离值，通过插值的方法生成直线段



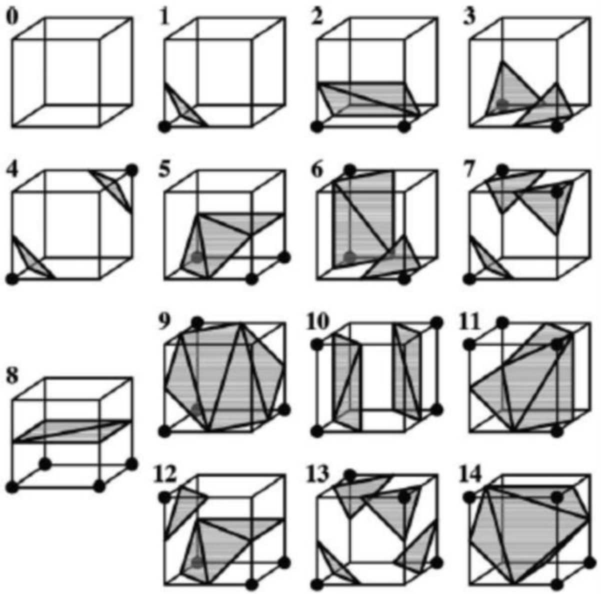


二维一共有16种情况

基于隐函数的三维模型重建

Marching Cube生成表面

根据体素的8个顶点的符号距离值，通过插值的方法生成面片

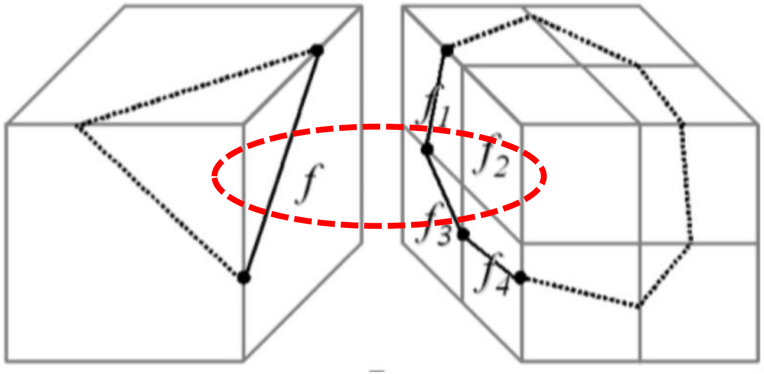


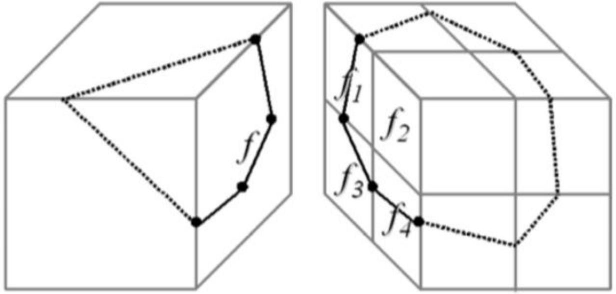
三维一共有256种情况

基于隐函数的三维模型重建

针对Octree的Marching Cube

相邻体素分辨率不同，导致出现空洞





Kazhdan M, Klein A, Dalal K, et al. Unconstrained isosurface extraction on arbitrary octrees[J]. 2007.

基于隐函数的三维模型重建

常用的表面重建算法

泊松表面重建算法	http://hhoppe.com/proj/poissonrecon/
Fssr重建算法	http://mesh.brown.edu/ssd/paper.html
Ssd 重建算法	http://www.regard3d.org/index.php/documentation/details/surface

深蓝学院
shenlanxueyuan.com

感谢各位聆听!

Thanks for Listening