

FDU 数字图像处理 8. 可视化

期末考试:

- 选择题
- 填空题
- 问答题 (伪代码)
- 证明题
- 计算题

8.1 二维数据可视化

二维数据 (x, y)

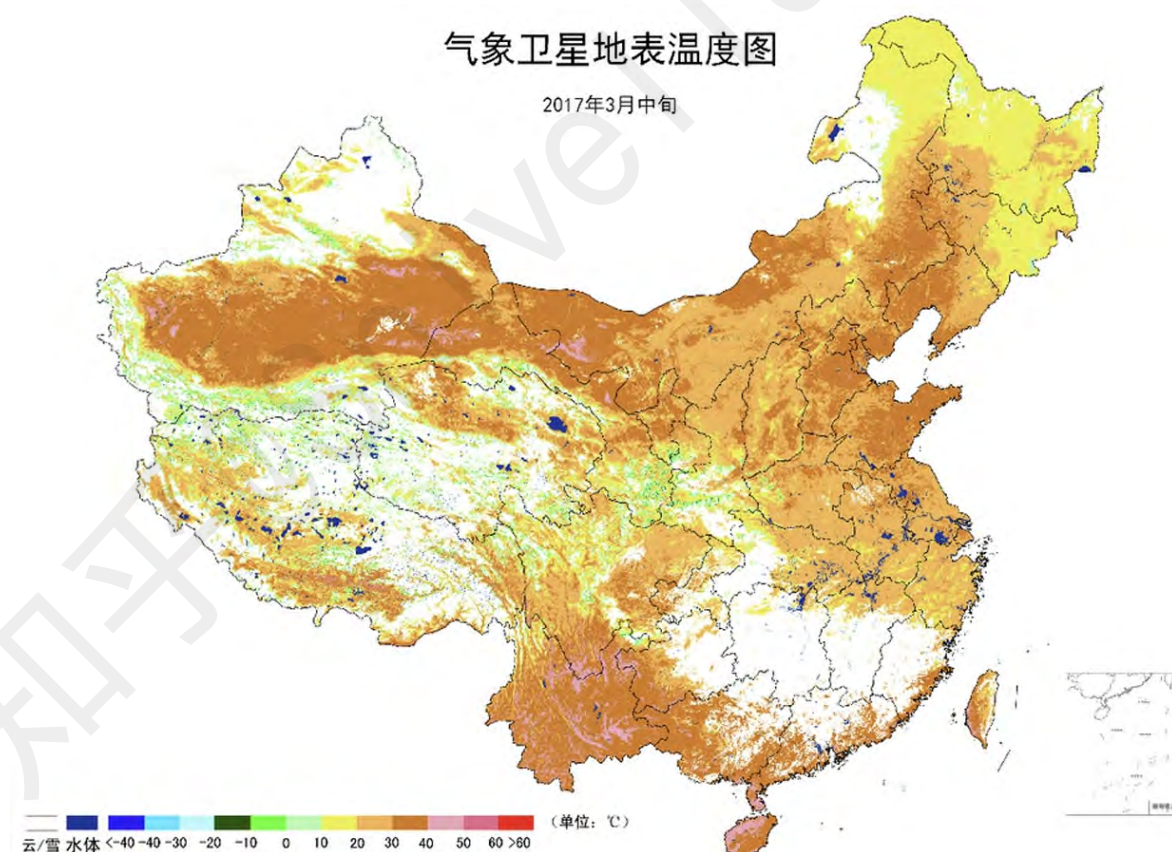
二维数据加时间 (x, y, t)

三维数据加时间 (x, y, z, t)

8.1.1 颜色映射

颜色映射 (color map)——传输函数设计 (将物理数值与颜色相联系)

- ① 建立颜色映射表
- ② 将标量数据转换为颜色表的索引值



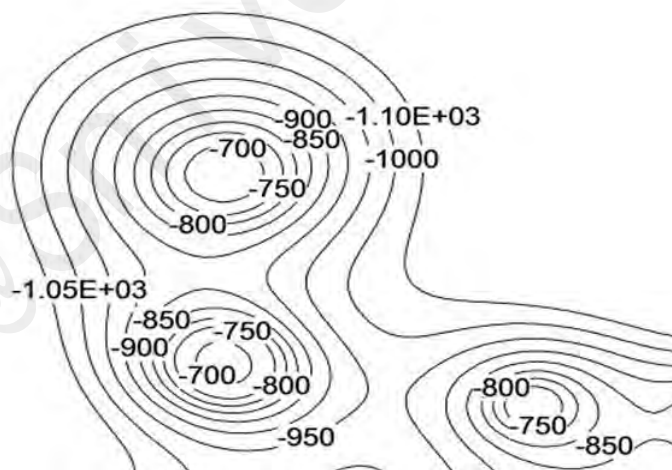
8.1.2 高度映射

高度通常用于编码测量到的数据.
(在三维数据可视化无法使用)



8.1.3 等值线映射

等值线 (iso-contour) 上的每一点代表的数值相同.
等值线稀疏处代表地形平缓.
在三维数据场下为等值面计算 (Marching Cube 算法)



8.2 三维数据可视化

三维标量场指分布在三维空间中, 记录空间场分布的物理化学等属性及其演化规律的数据场.
例如医学影像数据 (CT, 磁共振成像 (MRI)) 或地理气候信息 (大气数值模拟数据)

要求:

- ① 展示清晰
- ② 效率 (观察者看得清晰、计算机运算快)

8.2.1 截面可视化

截面可视化: 常见的如三个正交平面或任意角度切片
XY 平面 (水平切面)、XZ 平面 (纵向切面) 和 YZ 平面 (侧向切面)

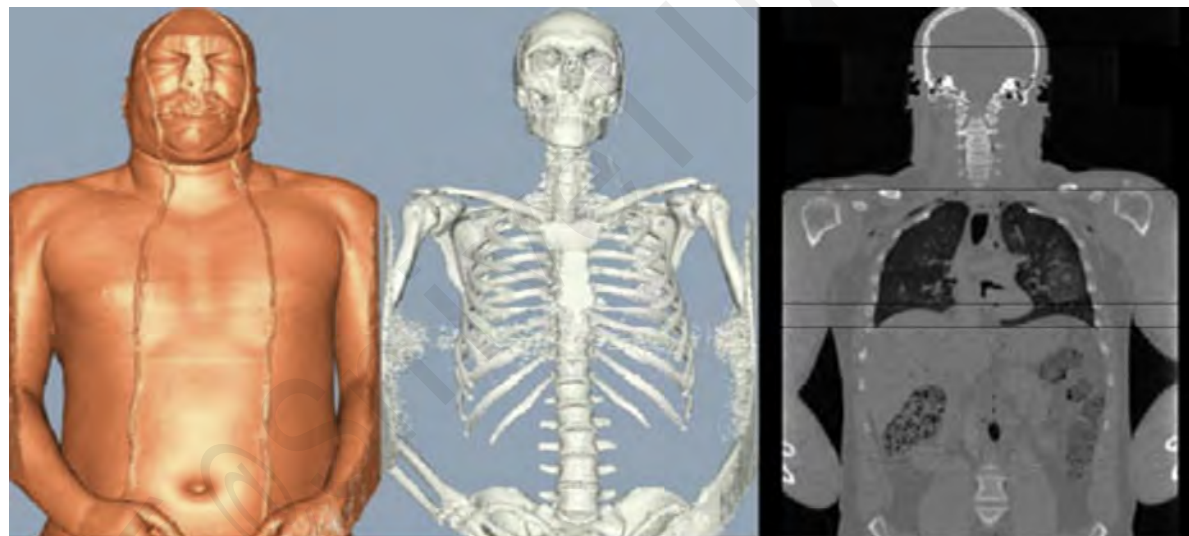
8.2.2 直接体渲染



3D 渲染



8.2.3 间接体渲染 (等值面渲染)



等值面1 (皮肤CT值)

等值面2 (骨骼CT值)

冠状截面图

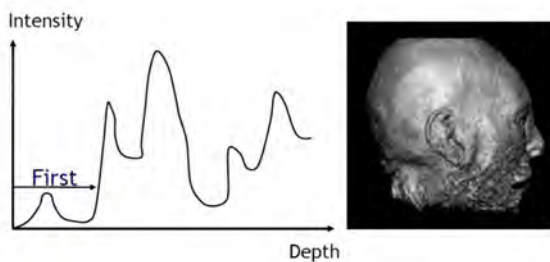
8.2.4 基本框架

体绘制 (Volume Rendering) 是处理和显示三维体数据的一种技术。
在体绘制中, 基于合成顺序的不同, 渲染方法可以分为两大类:

- ① 图像空间扫描的体绘制方法 (从屏幕发出射线)
- ② 物体空间扫描的体绘制方法 (像素点往屏幕扔雪球, 叠加所有投影)

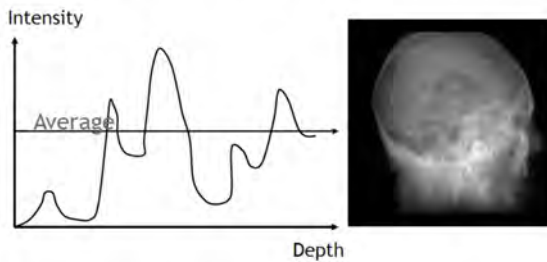
光线遍历 (Ray Traversal)

Ray traversal - First



First: extracts iso-surfaces

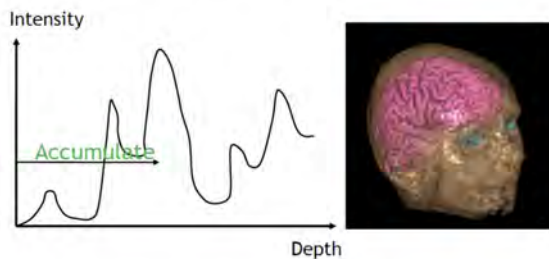
Ray traversal - Average



Average:
produces basically an X-ray picture,
an integral projection

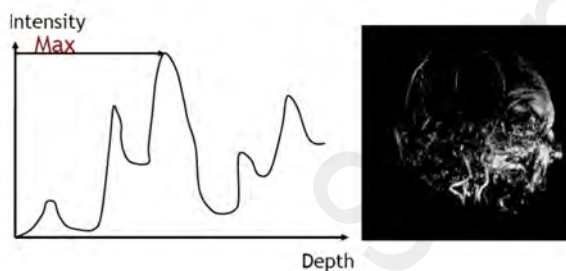
Distance along ray
can also be taken into
account **and used**

Ray traversal - Accumulate



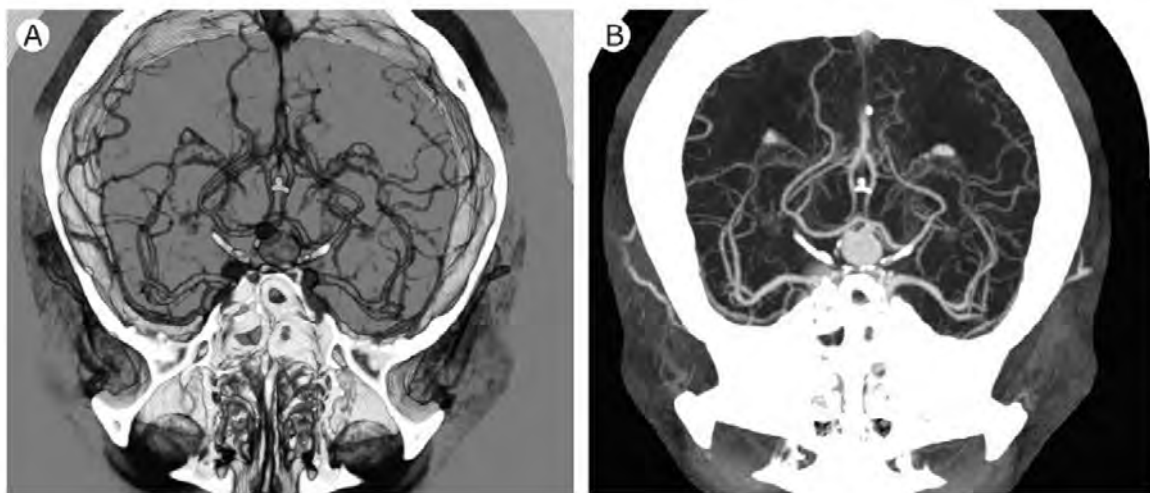
Accumulate opacity while compositing colors:
make transparent layers visible

Ray traversal - MIP



Max: Maximum Intensity Projection
used commonly for MRA images

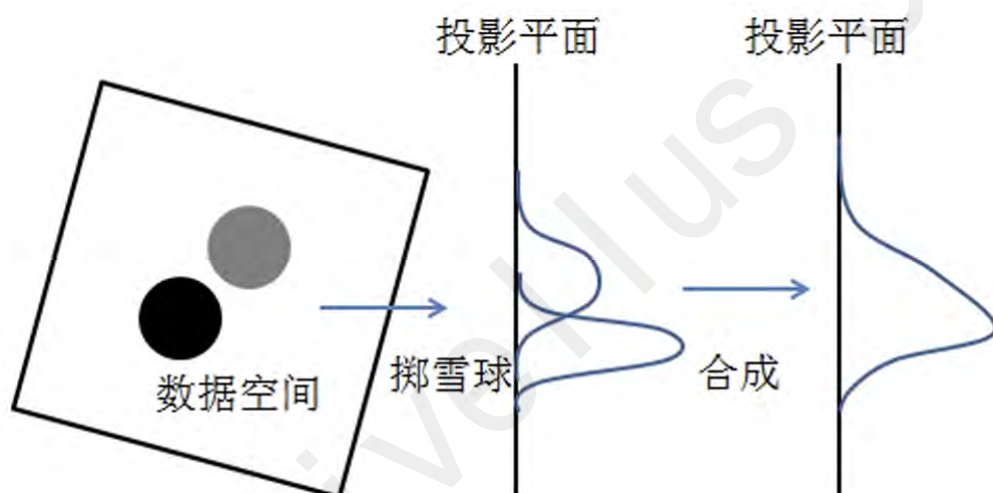
- ① 第一个交点
例如设置阈值 $\tau = 150$ ，记录射线碰到 $\tau = 150$ 的位置 (因为不透明，所以只记录第一个位置)
若没碰到阈值 $\tau = 150$ ，则不赋予颜色；
若碰到了阈值 $\tau = 150$ ，则根据离 "光源" 位置的远近赋予不同的颜色 (越近越鲜艳，越远越暗淡)
相当于提取等值面 (iso-surfaces)，适用于展示数据中的显著边界或结构
- ② 平均值投影
对光线在整个路径上穿过的数据强度进行平均计算，得到一个整体的强度值
这种方法相当于将体数据压缩为二维图像，类似于X射线成像 (积分投影)
适用于快速查看大致结构 (如骨骼、组织分布)，但立体感不高
- ③ 累积合成 (accumulate)
沿光线方向累积体素的不透明度和颜色值，逐步生成可视化结果。
累积的不透明度使得内部的透明层也可以部分可见，从而形成更丰富的视觉效果。
适用于可视化多层组织。
- ④ **最大强度投影 (MIP)** (直接体绘制的一种变体)
对光线在整个路径上的强度值取最大值，用以突出强度最高的区域
这种方法常用于显示局部高强度区域，例如血管中的造影剂。
常用于磁共振血管成像 (MRA)，以突出显示血管或高密度结构。



Christof Rezk-Salama

掷雪球法:

以三维空间数据场为处理对象，从数据空间出发向图像平面传递数据信息，累积光亮度贡献。
(权重依赖于数据点到屏幕的距离，以及投影点到屏幕上发光元件的距离)



绘制方法	绘制流程	特点	属于
光线投射法	从视点出发对每一像素投射光线，与三维标量场相交采样，合成沿光线上采样点的光学属性	体绘制积分的直接实现，简单，高效，灵活，绘制质量较高，是当前主流的体绘制算法	图像空间扫描法
滚雪球法	将体素按深度顺序投影到屏幕空间，计算每个体素的投影足迹，依次合成每个体素的光学属性	体绘制积分的近似实现，简单，高效，绘制质量一般，适合结构较为稀疏的三维标量场	物体空间投影法

8.3 面渲染

8.3.1 等值线

等值线 (2D Iso-contour)

基于双线性插值的思想计算等值线:

- ① 基本思想

等值线是一条函数值等于某特定值 v_{iso} 的曲线

对于给定的二维网格数据 (例如图像像素值或网格点上的标量值), 等值线通常在相邻网格点之间的边界上穿过。

双线性插值利用网格点的标量值, 计算等值线穿过网格边界的确切位置。

通过逐个处理网格单元, 累积等值线的交点并连接它们, 形成完整的等值线。

- ② 网格单元插值计算交点

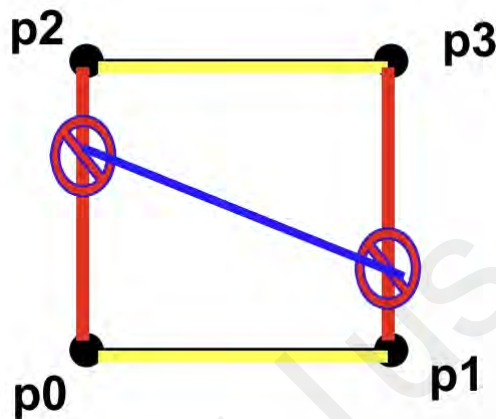
考虑顶点为 p_0, p_1, p_2, p_3 的网格, 顶点的像素值分别为 v_0, v_1, v_2, v_3

假设 $v_0, v_2 > v_{iso}$ 且 $v_1, v_3 < v_{iso}$, 则等值线只能与 p_0p_1 和 p_2p_3 相交。

以 p_0p_1 为例, 交点 $p = p_0 + \frac{v_{iso}-v_0}{v_1-v_0}(p_1 - p_0)$

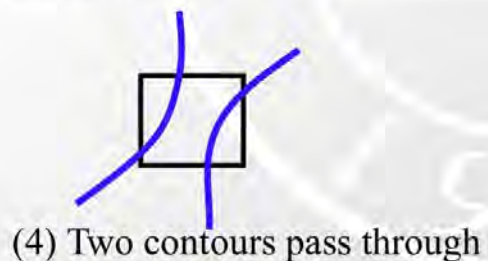
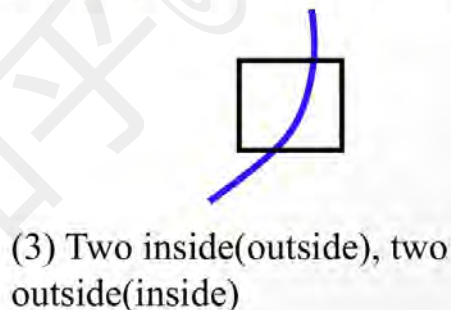
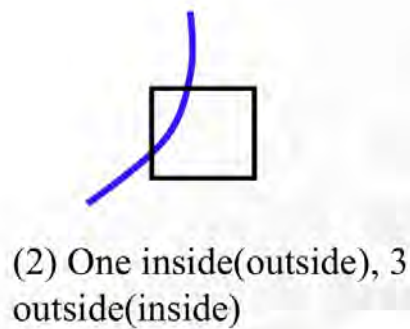
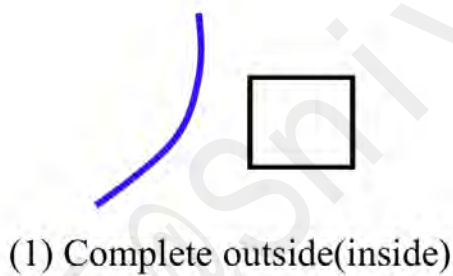
以 p_2p_3 为例, 交点 $p = p_2 + \frac{v_{iso}-v_2}{v_3-v_2}(p_3 - p_2)$

最终我们将两个交点连接起来

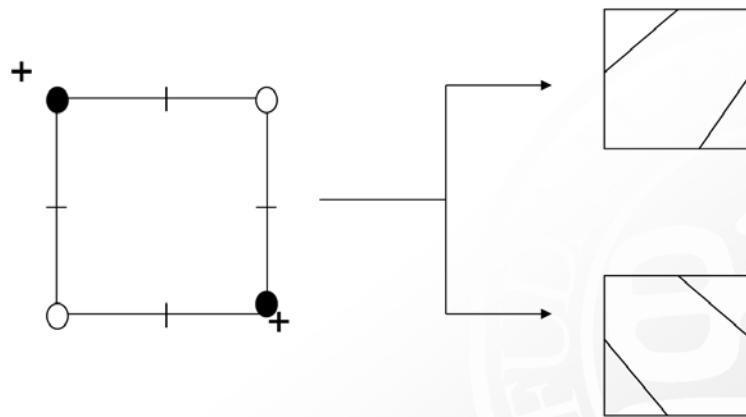


拓扑上一共有 4 种情况, 其中一种不需要写代码解决:

(图中绘制的是曲线, 但实际编程中, 单元格中的连线是直线)



第四种情况可能会出现歧义, 需要借助其余单元格的连接情况进行判断:



步骤总结如下:

- ① Look at one cell at a time
- ② Compare the values at 4 vertices with the iso-value v_{iso}
- ③ Linear interpolate along the edges
- ④ Connects the interpolated points together

使用三角单元格可以消除歧义 (而且要处理的只有一种情况)

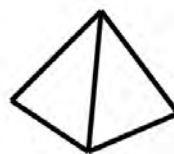
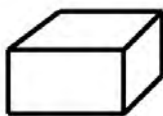
但会使得计算不准确、计算速度慢、图像碎片多 (这涉及渲染结果的碎片去除的任务)

8.3.2 等值面

等值面 (3D iso-surface)

Extend the same divide-and-conquer algorithm to three dimension

- 3D cells

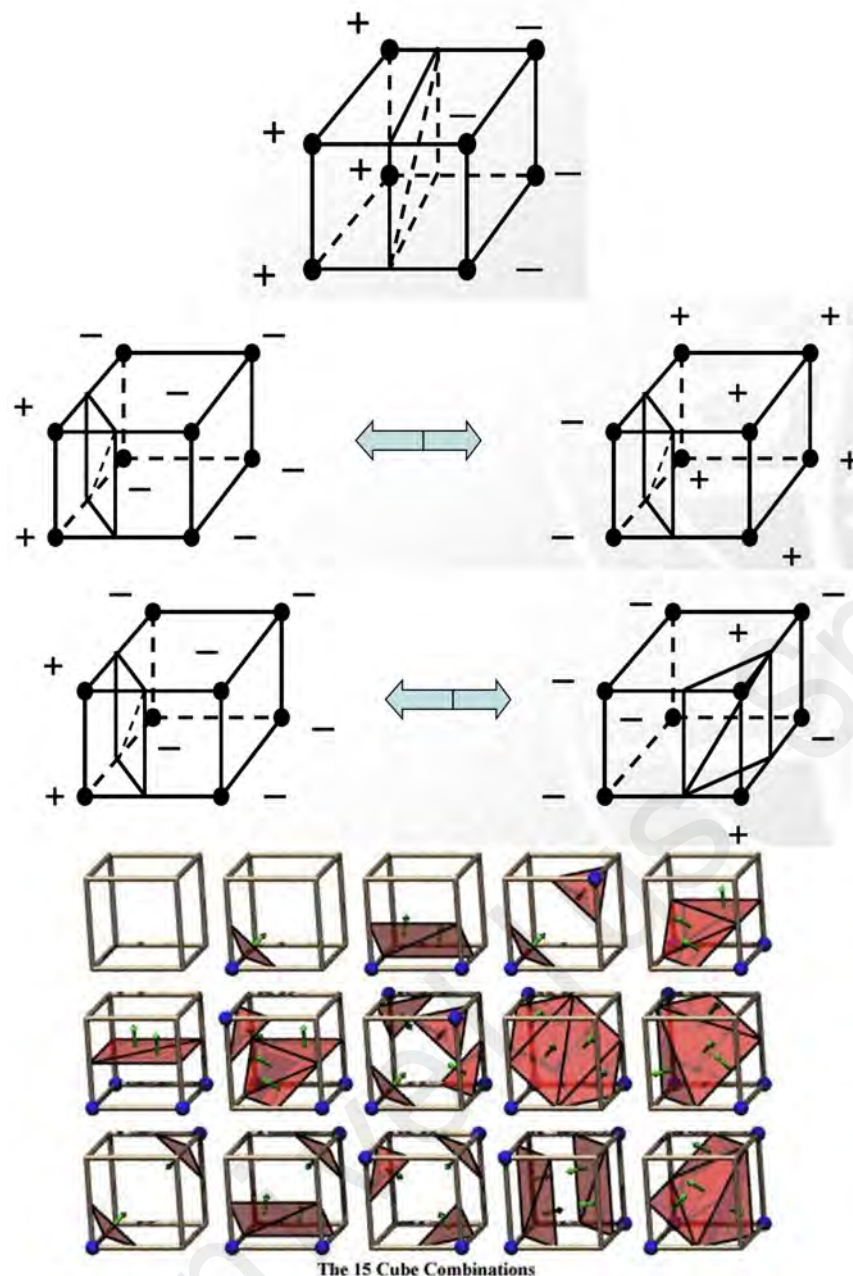


- Look at one cell at a time
- Let's only focus on voxel

考虑立方单元格 (记值大于 v_{iso} 顶点的 +, 记值小于 v_{iso} 顶点的 -)

8 个顶点一共有 $2^8 = 256$ 种情况,

但去除镜像对称和旋转对称的情况后, 拓扑上我们有 15 种不同情况



需要处理的仅有 14 种，但其中有好几种会有歧义性 (需要结合其他单元格进行判断)

8.3.3 Marching Cube 算法

A Divide-and-Conquer Algorithm:

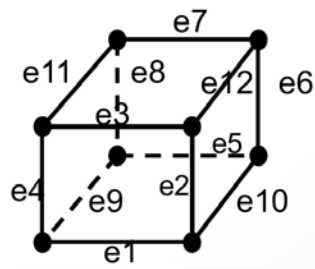
- ① Isosurface cells: cells that contain isosurface

$$\min < \text{isovalue} < \max$$

Isosurface cell search:

For a given isovalue, only a small portion of cells are isosurface cell.
Connect these isosurface cells.

- ② Perform linear interpolations at the edges to calculate the intersection points.
Used a lookup table to:
 - identify the edges that has intersections with the iso-surface
 - methods to form the triangular patches



Index intersection edges methods

0	...
1	...
2	
3	
	...
14	

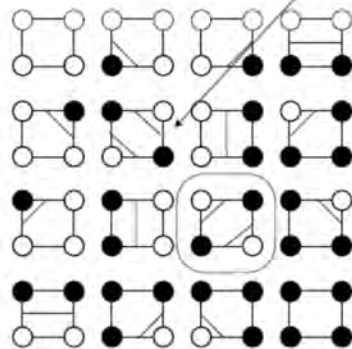
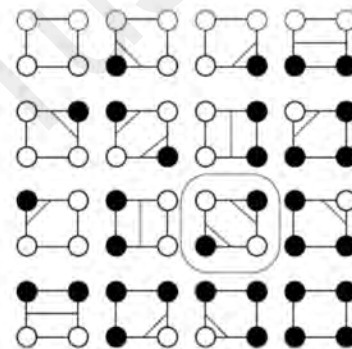
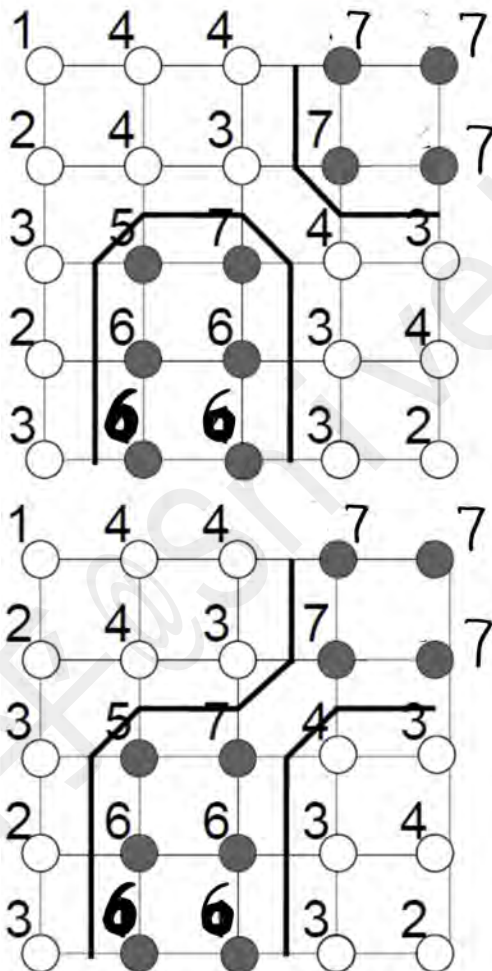
perform linear interpolations at the edges to calculate the intersection points.

- ③ Connect the triangular patches in isosurface cells.
(Color, Light...)

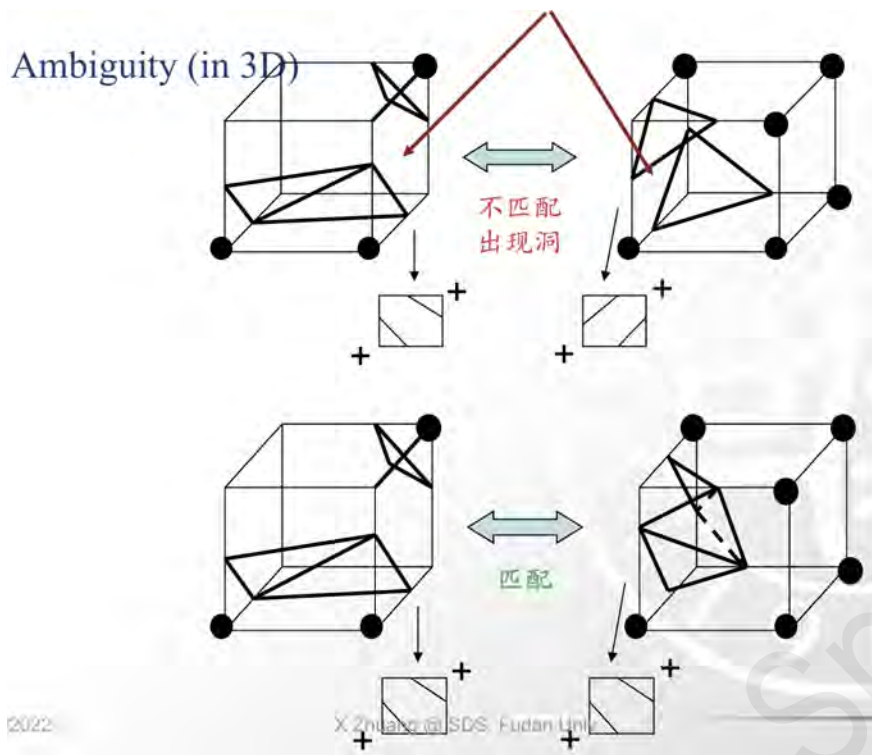
2D 等值线的歧义问题: (看上去没有那么严重)

Ambiguity (in 2D)

- $C=5$ (foreground ≥ 5)



3D 等值线的歧义问题:



[Marching Cube](#) 基本算法的 15 模式中:

- 无二义性表面: 0, 1, 2, 4, 5, 8, 9, 11, 14
- 各有一个二义性表面: 3, 6 (两种连接方式)
- 各有二个二义性表面: 10, 12 (四种连接方式)
- 有三个二义性表面: 7 (八种连接方式)
- 有六个二义性表面: 13 (64 种连接方式)

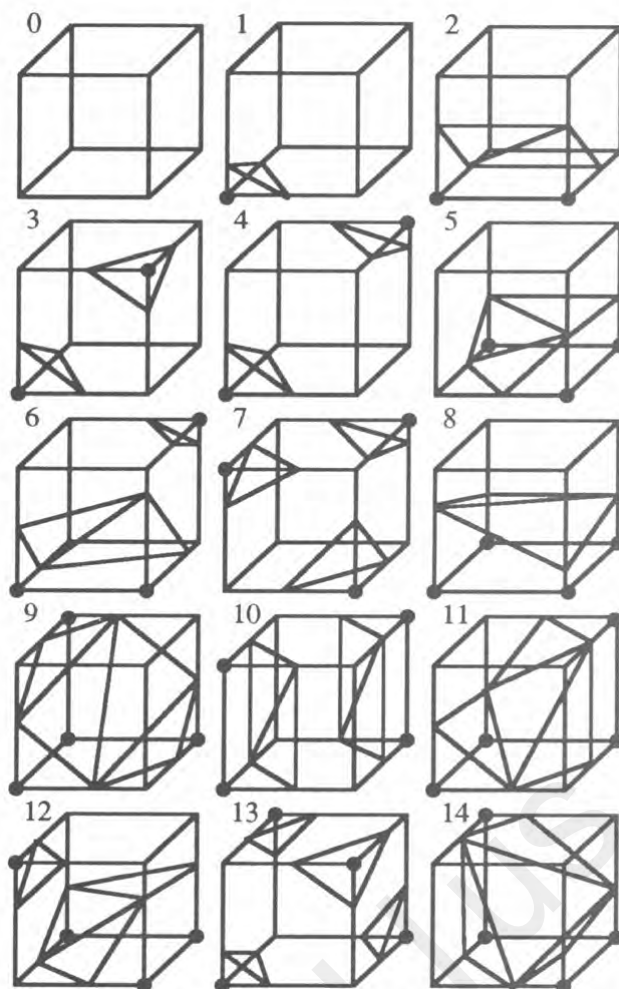
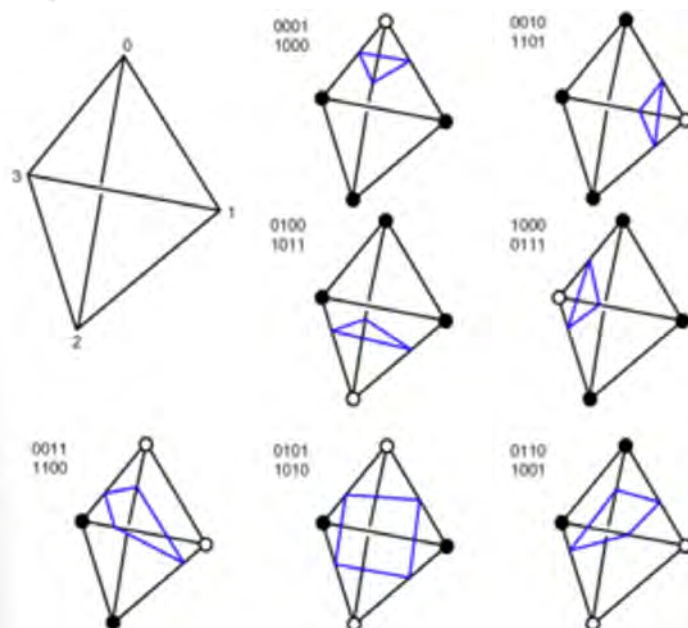
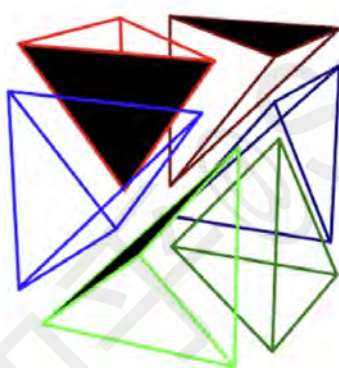


Figure 2. Configurations.

移动四面体法:

可以减少 (甚至避免) 歧义性

但容易造成较多碎片 (即 Homework 08 要解决的事情) 而且由于单元格变多, 计算速度也会变慢.



三维等值面显示时法向量的重要性:

- 法向量是光照模型 (如 Phong 模型或 Blinn-Phong 模型) 的重要输入。
在渲染中, 通过法向量计算表面反射光、漫反射光和环境光, 从而生成逼真的光影效果。
如果法向量计算不准确, 渲染出的等值面会显得平坦或失真。

- 最简单的计算方式就是使用归一化的图像梯度
(在离散网格数据中, 使用有限差分近似梯度)
(使用中心差分可以提高梯度计算精度, 减少数值误差)
(在标量场数据上应用 Sobel 滤波器, 直接计算梯度)

$$\nabla I = \begin{bmatrix} G_x \\ G_y \\ G_z \end{bmatrix}$$

$$\vec{n} = \frac{1}{\sqrt{G_x^2 + G_y^2 + G_z^2 + \epsilon}}$$

单精度大约 7 位十进制有效数字, 双精度大约 16 位十进制有效数字.

8.3.4 碎片消除

体数据获取过程中引入的椒盐噪声可能导致三角面碎片的产生.

我们可以在进行面渲染之前, 使用平滑核 (例如 Gauss 低通滤波器或中值滤波器) 进行平滑操作. 也可以直接对面渲染的结果进行平滑操作 (例如 VTK 库实现了 Laplace 平滑), 消除三角面碎片.

8.4 体绘制

等值面渲染的不足:

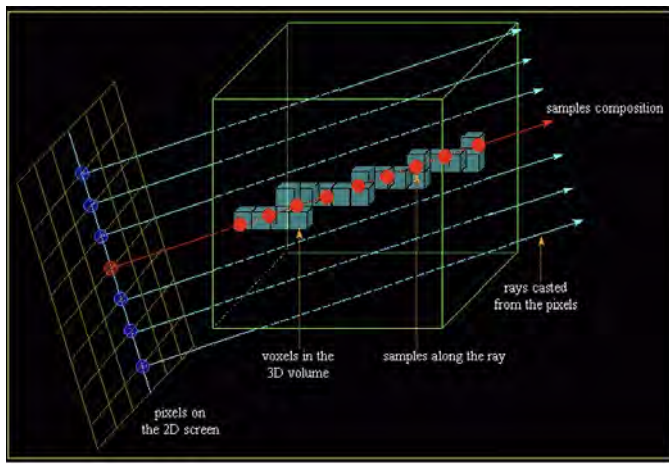
- 必须通过某种方法构造出中间曲面
- 细节丢失, 分割面被扩大
- 更高的图形质量的需求

体绘制 (volume render):

- 直接计算最终可视化里的每一个像素
- 所有体素对最后的图像亮度都有贡献

我们考虑基于**光线投射** (ray casting) 的直接体绘制方法.

- ① 采样重建 (体采样)
- ② 光照计算 (体光照模型)
- ③ 数据分类 (体分类 & 传输函数设计)
- ④ 光学积分 (体积分)



8.4.1 体采样

采样:

- **采样率:** 每个体素至少需要 2 个采样点 (Nyquist 采样定理)
- **等距采样:**
采样间隔 (步长) 太小, 采样效率低; 采样间隔太大, 会遗漏特征 (细节)
(采样间隔至少要小于体素间距 (spacing) 的一半, 这样不会丢失高频信息)
- **自适应采样:**
平缓均匀区域增大采样间隔, 而在复杂区域 (如梯度变化大或高频细节区域) 缩小步长以捕捉细节.
设光线当前采样点为 $p = f(x, y, z)$
线性步长调整公式为:

$$s(x, y, z) = \max(s_{\min}, \frac{s_{\max}}{1 + k \|\nabla f(x, y, z)\|})$$

- s_{\max} : 最大步长 (通常接近体素间距)
- s_{\min} : 最小步长 (确保高频区域的采样间隔不过小)
- k : 调节因子, 用于控制梯度对步长的影响

插值方法: 最邻近内插和三线性内插

- **① 最邻近内插 (nearest neighbor interpolation):**
将体数据中最近邻的体素的标量值作为采样点的采样值.

$$f_{\text{nearest}}(x, y, z) := f(\text{round}(x), \text{round}(y), \text{round}(z))$$

- **② 三线性插值 (trilinear interpolation):**
使用采样点周围 $2 \times 2 \times 2 = 8$ 个最邻近体素来计算采样点的值.
设采样点的坐标为 (x, y, z) , 定义:

$$x_0 = \lfloor x \rfloor$$

$$y_0 = \lfloor y \rfloor$$

$$z_0 = \lfloor z \rfloor$$

$$dx = x - x_0$$

$$dy = y - y_0$$

$$dz = z - z_0$$

$$f_{000} = f(x_0, y_0, z_0)$$

$$f_{100} = f(x_0 + 1, y_0, z_0)$$

$$f_{010} = f(x_0, y_0 + 1, z_0)$$

$$f_{001} = f(x_0, y_0, z_0 + 1)$$

$$f_{110} = f(x_0 + 1, y_0 + 1, z_0)$$

$$f_{101} = f(x_0 + 1, y_0, z_0 + 1)$$

$$f_{011} = f(x_0, y_0 + 1, z_0 + 1)$$

$$f_{111} = f(x_0 + 1, y_0 + 1, z_0 + 1)$$

三线性插值分三步完成:

- 在 x 方向插值: (投影到 (y, z) 平面)

$$f_{_00} := f_{000}(1 - dx) + f_{100}dx$$

$$f_{_01} := f_{001}(1 - dx) + f_{101}dx$$

$$f_{_10} := f_{010}(1 - dx) + f_{110}dx$$

$$f_{_11} := f_{011}(1 - dx) + f_{111}dx$$

- 在 (y, z) 平面插值:

$$f_{\text{trilinear}}(x, y, z) := f_{_00}(1 - dy)(1 - dz) + f_{_10}dy(1 - dz) + f_{_01}(1 - dz)dy + f_{_11}dydz$$

8.4.2 体光照模型

典型的几种体绘制光照模型:

- 只发射
- 只吸收
- **发射 + 吸收 (最常用)**
- 散射 + 阴影
- 多重散射

体绘制中最常用 "发射 + 吸收" 模型:

- **发射:** 颜色 (RGB)
- **吸收:** 不透明度 (α)

外部光源:

数据集 \rightarrow 三维几何 (加强深度感觉, 增强面结构信息) \rightarrow 图像 (光照)

Blinn-Phong 光照计算模型:

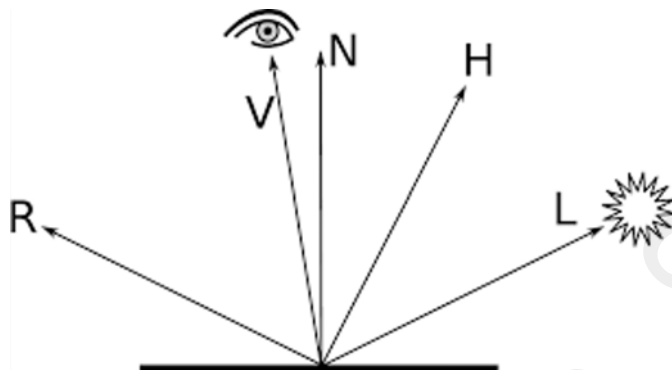
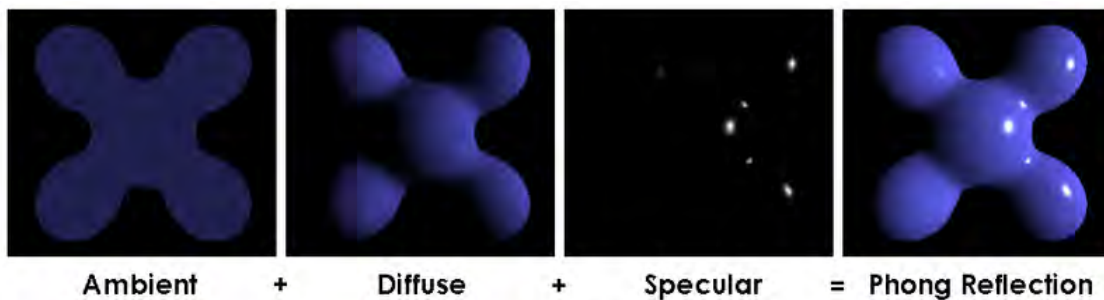
用于模拟物体表面与光线交互时的明暗效果.

它是一种经验性模型, 结合了三种基本光照组件:

- **环境光** (Ambient Light)
- **漫反射** (Diffuse Reflection)

- **高光 (Specular Highlight)**

(与物体颜色 (即从传输函数上采样得到的颜色 C_{transfer}) 无关, 只与外部光源的颜色有关)



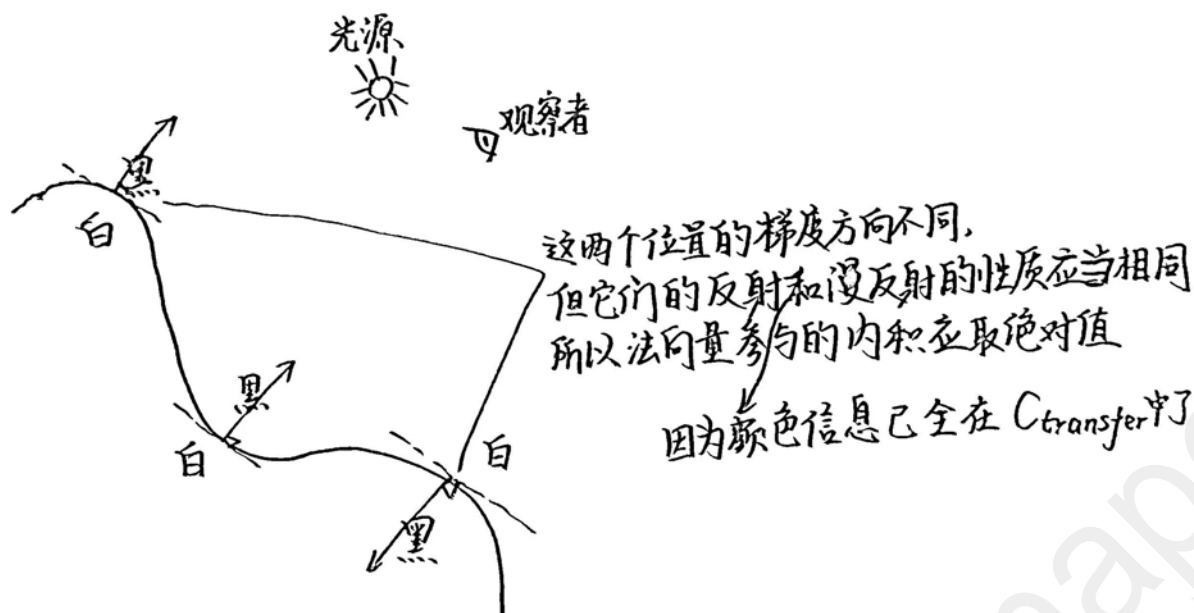
$$C := (\kappa_{\text{ambient}} + \kappa_{\text{diffusion}} \langle \vec{N}, \vec{L} \rangle) \cdot C_{\text{transfer}} + \kappa_{\text{specular}} (\langle \vec{N}, \vec{H} \rangle)^p$$

- \vec{N} 是法线方向, $\langle \cdot, \cdot \rangle$ 代表实 Euclid 空间的内积
(使用梯度近似法向量可能出现方向上的问题, 这可以通过对内积取绝对值来解决)
- \vec{L} 是光线方向, \vec{R} 是反射方向
- \vec{V} 是视线方向, \vec{H} 是 \vec{V} 和 \vec{L} 的平均方向
- C_{transfer} 是传输函数 $T(\cdot)$ 根据采样值 f 得到的颜色.
- κ_{ambient} 是环境光照系数, $\kappa_{\text{diffusion}}$ 是漫反射光照系数
- p 是高光系数
- κ_{specular} 是镜面反射光照系数.

体绘制中使用体数据的**梯度 (gradient)** 作为**体素 (voxel)** 的法向量.

对于正交网格, 我们通常采用中心差分的方式计算梯度:

$$\nabla F_{i,j,k} = \begin{bmatrix} \frac{F_{i+1,j,k} - F_{i-1,j,k}}{2\Delta x} \\ \frac{F_{i,j+1,k} - F_{i,j-1,k}}{2\Delta y} \\ \frac{F_{i,j,k+1} - F_{i,j,k-1}}{2\Delta z} \end{bmatrix}$$



法向量 $\vec{n}_{i,j,k}$ 是单位向量，因此需要对 $\nabla F_{i,j,k}$ 标准化。
标准化时需要规避零除问题：

- 若 $\|\nabla F_{i,j,k}\| < \text{eps}$ ，则取 $\vec{n}_{i,j,k} = [0, 0, 0]^T$ 等默认值。
- 若 $\|\nabla F_{i,j,k}\| > \text{eps}$ ，则取 $\vec{n}_{i,j,k} = \frac{\nabla F_{i,j,k}}{\|\nabla F_{i,j,k}\|}$

面绘制中的 Phong 光照模型: (以 VTK 库为例)

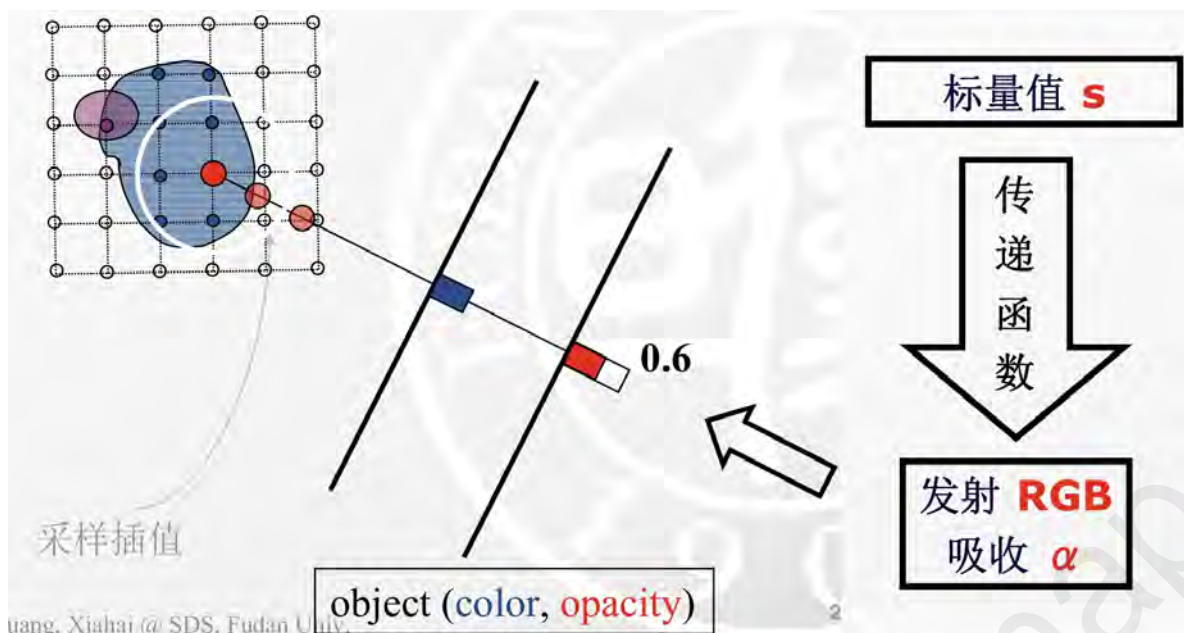
```
actor.GetProperty().SetColor(1,1,0)
actor.GetProperty().SetAmbient(0.25)
actor.GetProperty().SetDiffuse(0.6)
actor.GetProperty().SetSpecular(1.0)
actor.GetProperty().SetOpacity(0.6)
```

8.4.3 体分类和传输函数

传输函数 (transfer function):

一组定义了体数据的标量值 (如密度、温度或其他物理属性) 与不透明度和 RGB 颜色值等视觉元素之间的映射关系的函数。

- 不透明度 α 决定吸收的值 (阻挡住后方的采样点)
- RGB 颜色值决定发射的值

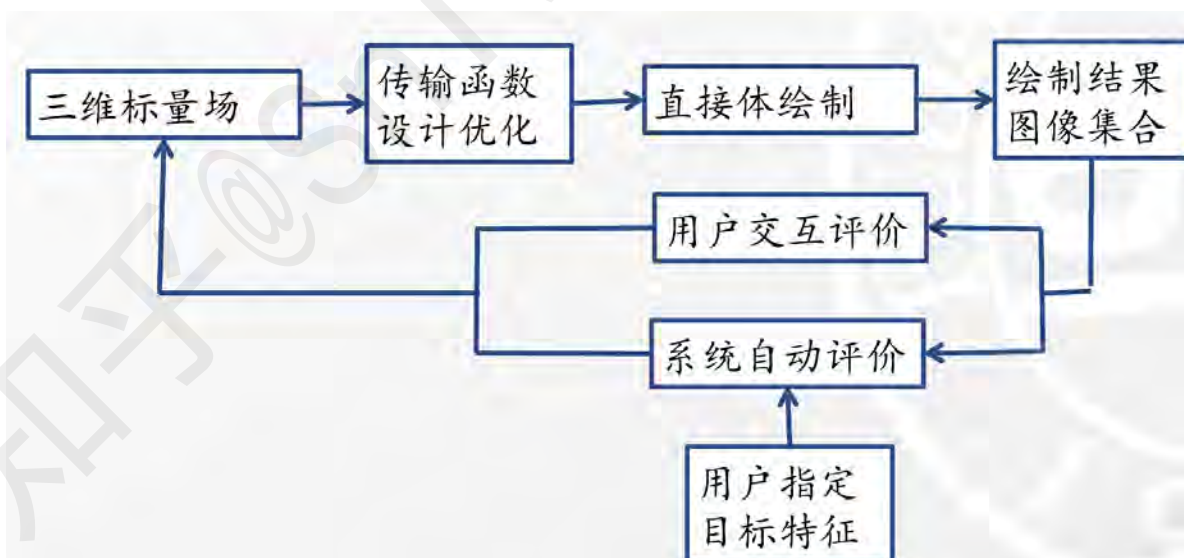


设计传输函数时的主要目标包括:

- ① **突出感兴趣的区域, 降低视觉混乱:**
例如在医学成像中, 可以突出血管、骨骼或肿瘤区域;
在科学计算中, 可以强调高温、高密度等重要区域。
对次要或不重要的特征赋予较低的不透明度, 避免干扰。
- ② **表现连续性和结构细节:**
保持相邻数据之间颜色和透明度的平滑过渡, 防止产生突兀的可视化效果。
- ③ **支持多属性映射:**
当数据包含多个属性时 (例如密度和梯度强度), 需要通过多维传输函数综合表示。

用户交互的传输函数设计通常包含两个方面:

- 映射规则设计
- 光学属性设计



(1) 一维传输函数

一维传输函数通常使用**查找表** (Lookup Table, LUT) 实现:

- 将标量值量化为离散范围
- 对每个离散值预计算颜色和不透明度.
- 使用线性插值处理连续值.

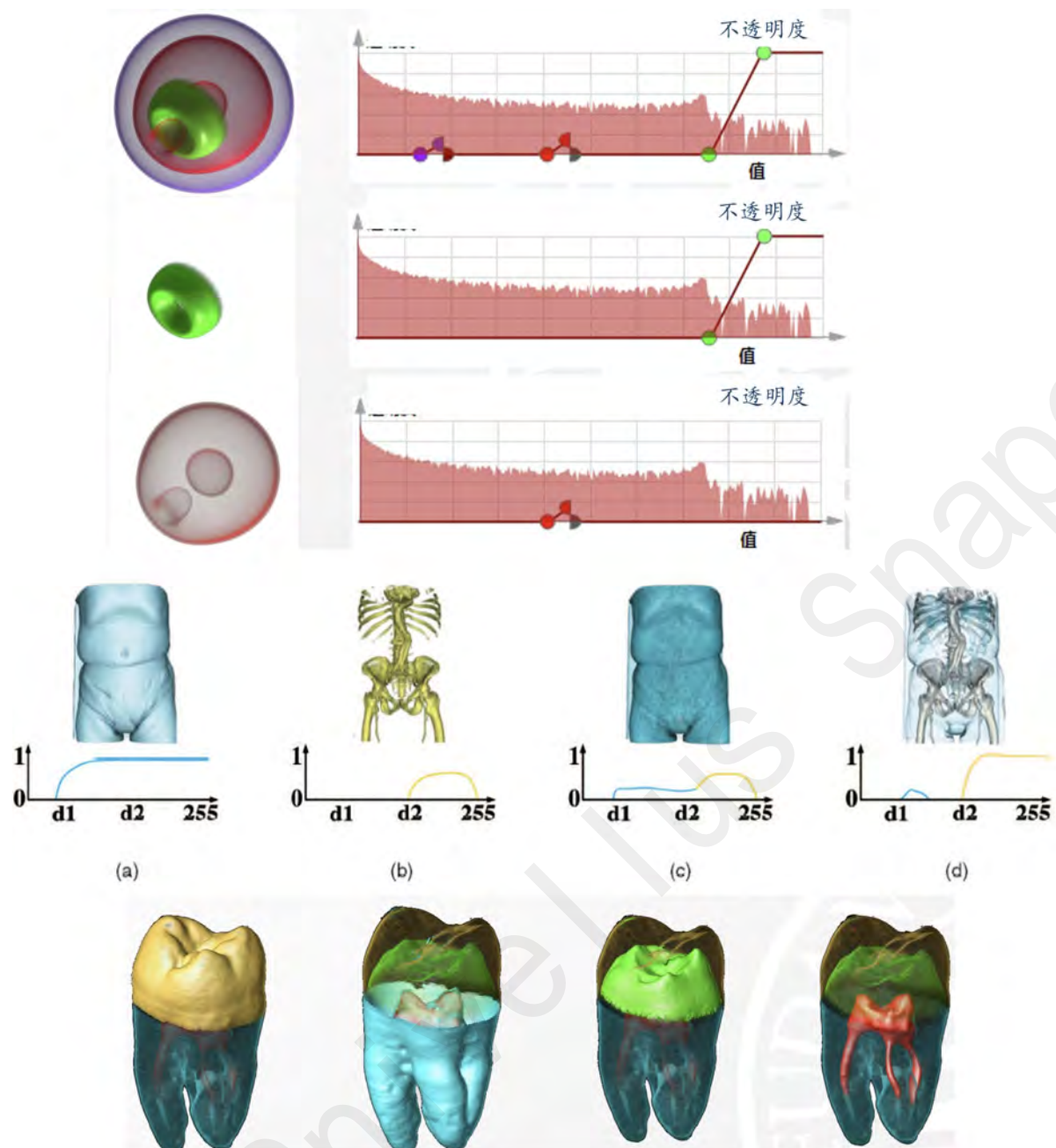
$$T(f) := T(f_1) + \frac{f - f_1}{f_2 - f_1} (T(f_2) - T(f_1)) \quad (f_1 \leq f \leq f_2)$$

在 CT 或 MRI 数据中, 传输函数可以用于分离不同的组织:

- **骨骼:** 赋予高不透明度和白色, 以突出
- **软组织:** 设置适中的透明度和颜色 (如红色或灰色)
- **背景:** 设置低不透明度, 使其接近透明.

例如基于 Hounsfield 单位 (CT 值) 的传输函数可能如下:

CT 值范围	颜色	不透明度
-1000~-500	蓝色	0.1
-500~0	红色	0.5
0~1000	白色	0.9



一维传输函数使用简单方便，允许用户反复尝试调整
但不满足特定分类需求（区分边界等），传输函数设计与绘制结果缺少直观的联系。

(2) 二维传输函数

二维传输函数: 以数据为中心的方法

标量值 + 梯度模

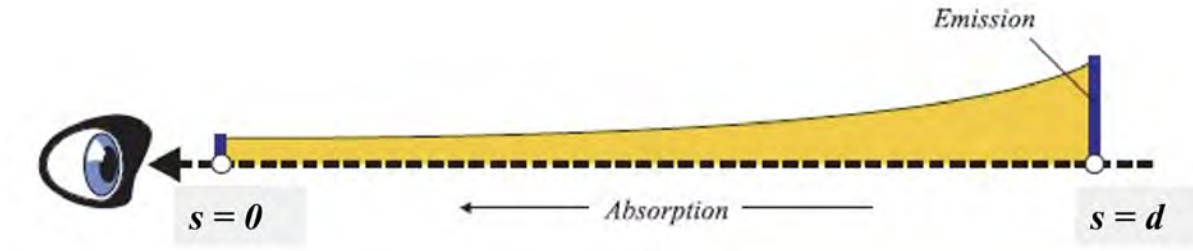
(VTK 库 SetScalarOpacity, SetGradientOpacity)

梯度模较大的采样点对应于边界。

扩展了函数的定义域，可以引入以下信息:

- 梯度模 (一阶中心差分)
- 曲率 (二阶中心差分)
- 特征形状 (纹理、尺度、形状和可见性等信息)
- 标量场的统计属性

8.4.4 体积分



Emission: 发射; Absorption: 吸收

记 $C(\cdot)$ 为颜色函数 (基于 Blinn-Phong 光照计算模型), $\alpha(\cdot)$ 为不透明度函数

则光线 r 的体积分 $I(r)$ 为:

$$I(r) := \int_0^d C(t) \exp \left\{ - \int_0^t \alpha(s) ds \right\} dt$$

$$\text{where } C := (\kappa_{\text{ambient}} + \kappa_{\text{diffusion}} \langle \vec{N}, \vec{L} \rangle) \cdot C_{\text{transfer}} + \kappa_{\text{specular}} (\langle \vec{N}, \vec{H} \rangle)^p$$

- \vec{N} 是法线方向, $\langle \cdot, \cdot \rangle$ 代表实 Euclid 空间的内积
- \vec{L} 是光线方向, \vec{R} 是反射方向
- \vec{V} 是视线方向, \vec{H} 是 \vec{V} 和 \vec{L} 的平均方向
- C_{transfer} 是传输函数根据采样值得到的颜色.
- κ_{ambient} 是环境光照系数, $\kappa_{\text{diffusion}}$ 是漫反射光照系数
- p 是高光系数
- κ_{specular} 是镜面反射光照系数.

实际计算中我们需要将上述积分离散化 (也就是采样)

设沿着光线 r 在 $[0, d]$ 区间上进行 n 点等距采样.

记第 i 个采样点为 t_i , 所发射的颜色值为 $C_i := C(t_i)$, 不透明度为 $\alpha_i := \alpha(t_i) \in [0, 1]$

我们可用光线 r 的体积分 $I(r)$ 的 Riemann 和来近似它:

$$\begin{aligned} I(r) &\approx \sum_{i=1}^n \left\{ C_i \exp \left(- \sum_{j=1}^{i-1} \alpha_j \cdot \frac{d}{n} \right) \cdot \frac{d}{n} \right\} \\ &= \frac{d}{n} \sum_{i=1}^n \left\{ C_i \exp \left(- \frac{d}{n} \sum_{j=1}^{i-1} \alpha_j \right) \right\} \\ &= \frac{d}{n} \sum_{i=1}^n \left\{ C_i \prod_{j=1}^{i-1} \exp \left(- \frac{d}{n} \alpha_j \right) \right\} \quad (\text{note that } \exp(-x) = 1 - x + O(x^2)) \\ &\approx \frac{d}{n} \sum_{i=1}^n \left\{ C_i \prod_{j=1}^{i-1} \left(1 - \frac{d}{n} \alpha_j \right) \right\} \quad (\text{we assume that } n \text{ is large enough, so } \frac{d}{n} \alpha_j \text{ is small}) \end{aligned}$$

简单起见, 我们假设不同光线的路径长度 d 是一致的, 采样点个数 n 也是一致的.

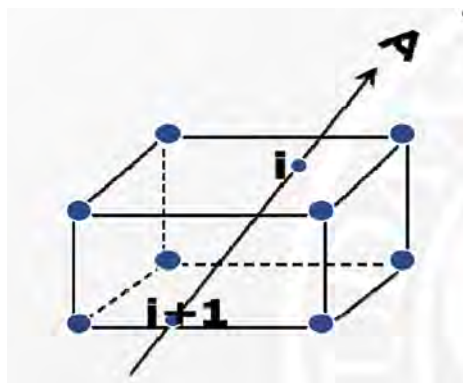
那么我们可以在设计传输函数时将常数项 $\frac{d}{n}$ 考虑进去, 从而简化公式.

在上述假设下, 体积分 $I(r)$ 的近似公式为:

$$\sum_{i=1}^n \left\{ C_i \prod_{j=1}^{i-1} (1 - \alpha_j) \right\}$$

(高效实现) 两种合成顺序:

- ① 从后向前的积分:
从 $i = n - 1$ 到 0 执行 $I_i = C_i + I_{i+1}(1 - \alpha_i)$

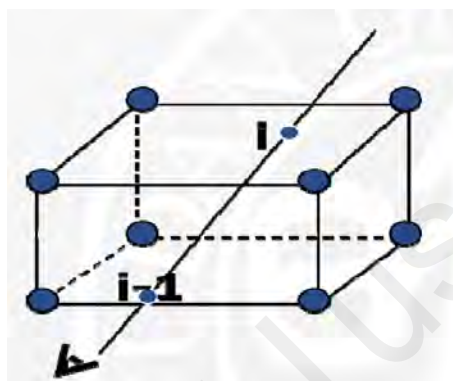


• ② 从前往后的积分:

从 $i = 1$ 到 n 执行:

$$I_i = I_{i-1} + C_i(1 - A_{i-1})$$

$$A_i = (1 - A_{i-1})\alpha_i + A_{i-1} = \alpha_i - A_{i-1}(1 - \alpha_i)$$



其中 A_i 是累积下来的对当前像素发光的总吸收率。

它等于前面的总透过率 $(1 - A_{i-1})$ 乘以当前位置的吸收率 (α_i) ，再加上前面的总吸收率 (A_{i-1})

当 $A_i \approx 100\%$ 时，我们就不需要继续往后走了，因为后面的采样点对体积分的贡献已经很小了。

光线投射算法伪代码

Procedure TraceRay(R)

Begin

$C(R)=0$; $A(R)=0$;

$x1=First(R)$; //在对象空间中光线进入的位置

$x2=Last(R)$; //在对象空间中光线离开的位置

$u1=Image(x1)$; //转换到图像空间的位置

$u2=Image(x2)$;

For $S=u1$ to $u2$ Do

$Sx=Object(S)$ // 转换到对象空间的位置

if($A(R) < 1$)

获取 Sx 点处的颜色 $C(Sx)$ 和透明参数 $a(Sx)$

$C(R)=C(R) + (1-A(R))*C(Sx)$

$A(R)=A(R) + (1-A(R))*a(Sx)$

endif

End for

End TraceRay

The End

知乎@Snivellus Snape