# 基于区域结构的多尺度体数据表达及其应用

(申请清华大学工学博士学位论文)

培 养 单 位: 高等研究院
学　　科: 计算机科学与技术
研　究　生: 王 律 迪
指 导 教 师: 郭 百 宁 教 授

二〇一一年六月

# A Region-Based Multiscale Volume Representation and Its Applications

Dissertation Submitted to

**Tsinghua University**

in partial fulfillment of the requirement

for the degree of

**Doctor of Engineering**

by

**Wang Lvdi**

**( Computer Science and Technology )**

Dissertation Supervisor : Professor Guo Baining

**June, 2011**

# 关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）根据《中华人民共和国学位条例暂行实施办法》，向国家图书馆报送可以公开的学位论文。

本人保证遵守上述规定。

**（保密的论文在解密后应遵守此规定）**

作者签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

# 摘 要

面模型与体模型是计算机图形学中最主要的两种物体表示方式。与面模型相比，体模型不仅描述物体表面，更刻画物体的内部属性，因而理论上具有更强通用性。然而，现有的体模型表示表达能力有限，不能很好地表达物体内部不同尺度的复杂细节，并存在建模困难、存储开销大以及绘制效率低等问题，极大限制了体模型在实际问题中的应用。

针对这一问题，本论文对真实世界物体进行了分析，发现物体在一定尺度上总可以看作由一些不同材质的区域组成：区域内部材质属性变化相对平缓，区域之间的边界处属性不连续跳变，这种区域划分构成物体独有的结构特征。基于这一基本思想，论文研究并提出了一种新的多尺度体模型表达和其相应的建模，绘制，存储算法，并将其扩展到头发的建模与表达中。论文的主要工作和贡献为：

- 提出了一种全新的基于区域结构的多尺度体数据表达。该表达能够以分辨率无关的方式准确描述物体内部跨越多个尺度层次的区域结构特征。使用有限存储，该表达可有效保持物体内部不同尺度的丰富细节，达到了表达能力与存储开销之间很好的平衡。
- 提出了完整的基于区域结构的体建模流程，其中包括一种基于 XML 的体对象标记语言及多种交互式建模工具。提出了体模型的转换与拟合算法，可以精确有效地从传统体数据表达中生成新的基于区域表达的体模型。从而大大扩展了体模型的应用范围和用户的建模能力。
- 提出了基于图形处理器的针对区域结构表达的实时绘制与反走样算法。算法通过对随机访问性能和底层数据结构的优化，支持具有相当复杂程度的区域结构体模型在主流图形处理器上的实时绘制，并且仅占用很少内存。由于算法不需要预计算，物体的边界可以动态变化。所提出的反走样算法具有与高倍多重采样类似的质量，但对绘制效率影响极小。
- 将区域结构表示扩展到头发几何，提出了第一个针对头发的几何特征分析与合成方法。通过对区域特征的自动抽取，允许用户以现有头发模型作为输入快速创建具有相似特征的高质量新模型，提高了头发几何建模的效率及已有数据的可重用性，也显示出基于区域结构的表示作为一种更通用数据结构的潜力。

**关键词：**体模型；多尺度建模；分辨率无关表达；空间剖分；隐式表面

# Abstract

Most 3D object representations in computer graphics can be categorized into surface models and volume models. In theory a volume model is more general because it describes the properties both on the boundary and at the interior of an object. In practice, however, the application of volume models is largely limited due to the difficulties in expressiveness, modeling, storage and rendering. In this dissertation we introduce a new representation for volume objects that utilizes the multiscale regional structures commonly found in real objects. It decomposes a volume model into multiple regions, each of which can define its own material properties. The boundaries between different regions can have arbitrarily complex shapes and are represented in a resolution-independent way. Our new representation has the following advantages over existing approaches:

*Expressive*: It can represent rich volumetric details spanning a wide range of scales in a precise and resolution-independent way, which means the sharp features of an object will remain crisp no matter how much it is magnified.

*Easy-modeling*: It enables an iterative modeling workflow that differs from traditional ones for volume objects. With the definition of region structures, a complex object can be decomposed into several components and modeled in a divide-and-conquer fashion. Using an XML-based Volumetric Object Markup Language and a handful of interactive tools we have provided, highly complex volume models can be created from scratch.

*Compact storage* and *efficient rendering*: With our carefully designed data structures and rendering algorithms, arbitrary cross sections of a complex region-based volume model can be rendered in real time on commodity graphics hardware, while consuming a very small amount of video memory.

Furthermore, we have extended our representation to hair, an object that exhibits prominent volumetric features. Based on an automatic region analysis algorithm, we propose an example-based pipeline that generates high quality novel hairstyles from existing ones, and thus significantly improves the efficiency of hair modeling. This also implies that the region-based representation could potentially be a more general data structure.

**Key words:** volume models; multiscale modeling; resolution-independent representations; space partitioning; implicit surfaces

# 目 录

# 主要符号对照表

| | |
|---|---|
| BSP | 二叉空间分割（Binary Space Partitioning） |
| CSG | 构造实体几何（Constructive Solid Geometry） |
| OBB | 定向包围盒（Oriented Bounding-Box） |
| RBF | 径向基函数（Radial Basis Functions） |
| SDF | 符号距离函数（Signed Distance Function） |
| VOML | 体对象标记语言（Volumetric Object Markup Language） |
| $S$ | 三维几何实体 |
| $\partial S$ | 三维几何实体的边界 |
| $\mathcal{D}$ | 符号距离函数 |
| $C$ | 材质属性函数 |
| $\mathcal{P}$ | 对三维物体的区域划分 |
| $\kappa$ | 区域数目 |
| $\ell$ | 区域标记 |
| $\mathcal{G}$ | 三维规则网格 |

# 第 1 章　计算机图形学中的体数据表达

三维模型是对三维空间中物体的抽象描述。随着计算机图形学与硬件水平的飞速发展，三维模型的应用早已深入各个领域，从工业设计到建筑效果，从电影特效到视频游戏，从文物古迹的重建到基础科学的研究。三维建模的目的不仅是真实物体的数字克隆，更可以帮助人们更好地认识真实世界中难以直接观察的事物，例如人体的内部构造、物质的微观结构等等。

在几十年的历史中人们提出了很多种三维模型的表达形式，这些不同的表达可以分为表面模型与体模型两大类。表面模型（surface models）又称边界模型，顾名思义，描述物体的表面而忽略物体的内部。表面模型通过定义物体表面上点的空间位置及属性描述物体的几何形状以及材质信息。由于三维物体的表面是二维曲面，其几何形状在常见的表面模型中通常表示为多边形网格、样条曲面、细分曲面、点云等形式。材质属性的表示则主要通过二维纹理映射的方式。与之相对，体模型（volume models）通过物体所占的空间，亦即空间中任意一点是否属于该物体定义其几何形状，并且对物体表面及内部各点的材质属性均有定义。

表面模型对物体内部信息的忽略通常是建立在两点假设基础上。第一，对于真实世界中的绝大多数不透明物体，光线与物体的交互仅发生在表面或是表面以下很浅的次表面，这即是说物体的外观很大程度由其表面附近的属性决定，如图1.1所示；第二，物体的边界是静态的，亦即虽然物体作为一个整体可以发生变形，但物体内部不会动态地暴露出来形成新的边界表面。

然而，许多真实物体的外观并不仅仅由表面属性决定，还受光线与其内部复杂交互过程的影响，例如许多非均匀半透明材质构成的固体，以及火、烟雾、液体等流体。另一方面，除了原始的边界表面，很多时候还需要动态地呈现物体内部的信息，例如在交互式应用中模拟物体的破碎、腐蚀、切割等效果，在生物研究与医学诊疗中显示研究对象内部任意切面上的情形等。以上这些场景（如图1.2）使用传统的表面模型均难以描述，而体模型却可以很好地胜任。

尽管体模型理论上是一种更通用的三维物体表示，但它目前在图形学中，尤其是表达一般三维物体的应用却远不及表面模型广泛。究其原因，我们认为主要是传统体模型的以下几点问题造成的：

- **表示能力有限：** 真实世界中的绝大多数物体都是由非均匀的材质构成的。即使像水果这样的寻常物体内部往往也有着复杂的结构和丰富的细节。如果

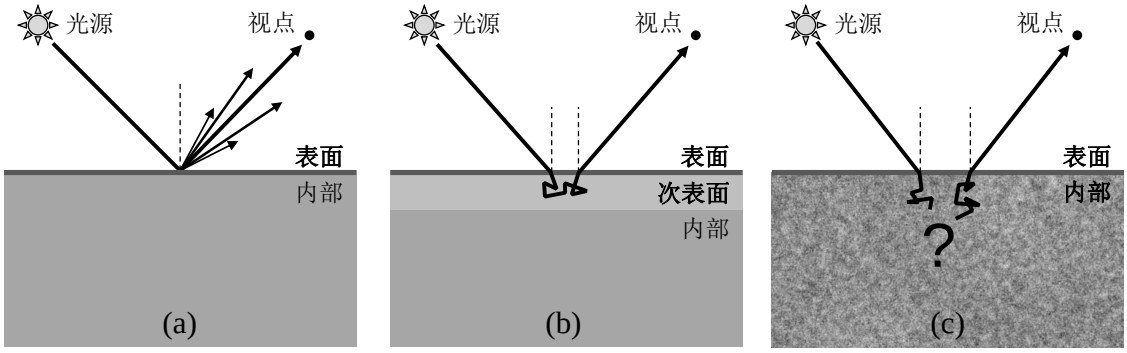



图 1.1　光与物体的不同交互方式。从左至右，光与物体的交互依次发生在 (a) 表面，(b) 次表面，以及 (c) 物体内部。

我们把物体内部材质属性的变化看作一个三维信号，这个信号通常不是带限（band-limited）的。传统的基于体素（voxel）的体数据表示由于其离散采样的本质，对于这样的材质属性变化仅能做到近似表示，在放大物体局部进行绘制时难以避免走样、模糊等问题。

- **建模困难：** 由于体数据本身的复杂性以及体模型表达能力的限制，目前体建模的手段十分有限。现有的体模型主要是通过体扫描技术（如 CT、MRI 等）重建真实样本，或是使用基于过程的方法（如过程噪声、纹理合成等）产生。值得注意的是，相对于体数据，许多二维图像都是通过手工创建或以手工方式编辑已有内容，因为手工建模具有最大的可控性和灵活性。然而手工创建或编辑体数据却相当困难，最根本的原因在于，人类的视觉来自周围景象在视网膜上的二维投影，这一机制决定了人无法在同一时刻没有歧义地看到一个体数据中的全部信息①，因此无法像在平面上作画一样能够立即看到自己操作产生的效果。此外，目前主流的人机交互界面仍然是为二维输入输出设计的[1]，这使得设计交互式的体数据建模界面更加困难。

- **存储困难：** 三维物体内部的信息量远大于其表面的信息量，这使得体模型往往需要更多的存储空间。对于常见的基于离散采样的体模型，对存储空间的需求随着细节程度的提高呈立方增长（在三个方向上分别提高一倍分辨率将使存储开销增至原先的 8 倍）。正因为此，基于离散采样的体模型所能表现的细节程度与其存储开销之间存在难以调和的矛盾。例如一个 $1024^3$ 分辨率、24 位颜色格式的离散采样体数据大约需要 3.2 GB 的存储空间，这不仅远超当前主流的显存容量[2]，甚至超过一些个人计算机的内存容量。存储空

① 特定形式（例如离散采样）的体数据可以以一定方式展开到二维平面，但在展开过程中损失的空间邻接关系等信息会使人难以直观地在大脑中重构原始的体数据。


图 1.2　一些适合用体模型表达的真实物体。(图片来源：水母、蛋糕来自 DeviantArt.com；水晶来自 WonderWorlds.org；水果来自 Discovery.com；水磨石来自 Flickr.com；人体解剖图和地层结构来自 University of Utah)

间的问题也进一步限制了体模型的表达能力。

- **绘制困难：** 体模型绘制上的困难主要反映在绘制质量与绘制效率两方面。基于离散采样的体模型由于分辨率在实际应用中受存储空间严重限制，在绘制时表现为因插值产生模糊、缺少高频图像细节。一些非离散采样的体数据模型[3–6] 虽然能够支持分辨率无关的特征，但由于本身的设计并不能支持高效的随机访问，因此不能很好地在图形处理器上进行实时绘制，或是每次体模型发生变化均需要一定时间的预计算。

以上这几方面困难并非相互独立，而是彼此关联和影响着。在现有的体数据表达基础上很难完全解决所有问题，因此我们希望寻找一种更好的体数据表达。

这里我们首先明确定义本文中出现的一些关键概念：

本文中的**体数据**泛指分布或定义于三维空间的数据集合。**体数据表达**指一种特定的体数据格式，例如基于离散采样的体数据表达。**体模型**则指以某种体数据表达描述的物体实例。

本文中的体模型既包含物体的几何形状，又包含物体的材质属性，因此区别于作为一种纯几何表示的实体几何（solid geometry)。物体的**材质属性**泛指物体除几何形状以外的随空间变化的内在属性，可以是颜色、透光率、折射率等外观属性，也可以是密度、硬度等物理属性。体模型的材质属性类似体纹理（solid texture)，但我们期望描述一般的材质属性变化，而体纹理通常指以一定重复性或随机性模式变化的材质属性，因而可以看作体数据的一个特例。

根据以上定义，我们可以将任意体模型表示为两个定义在 $\mathbb{R}^3$ 上的函数 $\{\mathcal{D}(x),\mathcal{C}(x)\}$。对于三维空间中任意一点 $x$，$\mathcal{D}(x)$ 定义 $x$ 点是否属于该物体，我们称 $\mathcal{D}$ 为几何属性函数；对所有属于该物体的点 $x$，$\mathcal{C}(x)$ 定义物体在 $x$ 点的材质属性，因此称 $\mathcal{C}$ 为材质属性函数。

本论文的主要研究目标是寻找一种表达能力强、建模方便、存储紧凑、绘制高效的新的体数据表达。具体来说，我们希望它能够以分辨率无关的方式准确表达真实物体内部遍布多个不同尺度的丰富细节，能够支持直观高效、灵活多样的建模方式，同时在存储开销以及绘制效率上能够允许复杂的体模型在目前主流的图形处理器上实时绘制。在本章的余下部分我们首先一起回顾相关领域的研究工作，接着提出本文的工作及主要贡献。

## 1.1　相关工作

本文的工作很大程度上建立在体数据的表达、建模、存储与绘制领域。我们将传统的体数据表达划分为三类，分别是基于离散采样的体数据表达、基于过程噪声的体数据表达与基于结构信息的体数据表达，在本节中我们将依次对这三个类别以及各自具有代表性的研究工作加以介绍。在分析一种体数据表达时，我们将特别关注四个方面：表达能力，建模方便程度，存储空间占用，以及绘制效率。此外，由于本文提出的基于区域结构的体数据表达具有分辨率无关的特性，我们也将简要介绍二维图像或表面上分辨率无关表示的相关工作，特别是近年来利用图形处理器实时在任意物体表面上绘制矢量图的研究。

### 1.1.1　基于离散采样的体数据表达

在基于离散采样的体数据表达中，物体的几何属性函数 $\mathcal{D}$ 及材质属性函数 $\mathcal{C}$ 均以离散采样的方式表达。离散采样表达中最常见的是在一个三维均匀网格上采样。这种表达可以看作是二维位图的一个直接三维扩展，位图中像素的概念在扩展到三维后通常称为体素（voxel）。

与位图一样，基于体素的表达支持高效的随机访问，因而非常适于目前的图形处理器使用。对于图形处理器来说，高效的随机访问意味着内存访问的局部性。对于基于体素的体数据，空间中任意一点的值可以由定义在该点附近网格节点上的值插值得到。常用的插值方法有最近邻插值、三线性插值、三立方插值等，插值所需要访问的临近网格节点数依次为 1、8、64，因而插值的质量与时间复杂度均逐级提高。由于目前的图形处理器均支持硬件三线性插值，使得线性插值与

最近邻插值相比几乎不需要额外的计算代价[7]，根据这一点，Sigg 和 Hadwiger[8] 提出将三立方插值所需的 64 次内存访问转化为 8 次利用硬件三线性插值的内存访问，使得图形处理器上对基于体素表示体数据的三立方插值效率明显提高。Ruijters 和 Thévenaz[9] 在其基础上进一步提高了插值的准确性。

离散采样体模型的一个主要问题是存储空间与表达能力的矛盾。一个离散采样体模型所能表达的细节与采样点的数量直接相关，而后者又决定了存储空间的占用。例如定义在一个分辨率为 $d \times d \times d$ 的三维网格上的体素模型，若在三个方向上的分辨率分别提高一倍，则内存占用呈立方增长至原来的 8 倍。过大的存储空间占用有可能使得体模型不能被完全放入显存甚至内存，而这会严重地影响绘制效率。因此实际当中，用户必须在体模型所能表达的细节程度与有限的内存空间之间做出取舍。值得注意的是，在个人桌面计算机的屏幕分辨率已达到 $1920 \times 1200$ 的今天，多数交互式应用程序中用到的基于体素的体模型分辨率却很少超过 $512^3$。

即使忽略存储空间的因素，离散采样体模型的表达能力仍然受其自身特性的限制。复杂的真实物体通常由多种不同的材质组成，物体的材质属性函数 $C$ 在不同材质之间的介面处会发生不连续的跳变。从信号处理与频率分析的角度看，这种跳变对应于无限高的频率，因此从本质上无法用有限频率的离散采样完全描述。

此外，离散采样体模型的特点使得创建和编辑这一类体模型的方式十分有限。图形学中针对不同类型内容的建模方法几乎都可以划分为以下三类：手工方法，基于过程的方法，以及采集实物。其中手工方法具有最大的灵活性，基于过程的方法通常仅适用于特定类别的对象，而采集实物仅能产生一部分真实存在的物体。如前文所述，由于离散采样体模型是对三维物体的属性函数进行均匀离散采样而并不分析和利用物体的内在结构，使得手工创建及编辑的方法效率低下。因此，现有的离散采样体模型几乎都是通过后两类方法生成的。

#### 1.1.1.1　直接采集

在医疗领域通常利用计算机断层成像（CT）或核磁共振（MRI）等技术获取生物体内部的三维信息[10]。此外，计算机断层成像也被广泛应用于工业中，例如缺陷检测、装配分析、逆向工程[11,12] 等。这类方法使得人们可以在不破坏外部构造的前提下获知物体的内部信息，从而建立真实物体的数字化副本并进行各种分析研究，例如提取感兴趣的等值面[13]。

与所有采集自实物的建模方法一样，这类方法的主要问题是能够采集的对象

有限。受采集设备的尺寸、采集方法、分辨率限制，并非任何物体均可以通过这种方法得到有意义的内部信息。另外，这类方法采集得到的物体材质属性通常仅反映对特定射线（如 X 射线）的放射密度（radiodensity）变化，而难以直接获知其它类型的材质属性。最后，这类方法仅能够重建真实存在的物体。

#### 1.1.1.2　基于样本的纹理合成

尽管手工创建一个大而复杂的体模型需要花费难以忍受的时间和精力，但创建一个很小的体模型却是可能的。如果一个大的体模型中任意一点的局部邻域都与一个小的体模型相似，一个直接的想法是能否先手工创建这个小的体模型，再以此为样本设计某种算法自动产生一个大的体模型。另一方面，有的体模型中不同的二维截面均相似，一个自然的想法是能否首先创建任意一个二维截面上的信息，再以此为样本生成完整的体数据。根据这些想法，一些研究工作尝试通过基于样本的纹理合成进行体数据建模。

纹理（texture）在图形学中的定义有两种。第一种定义泛指可以包含任意内容的一般图像；而第二种定义，亦即在计算机视觉、图像处理、以及本文所讨论的范畴中的定义，指具有一定重复性图案的一类特殊图像。基于样本的纹理合成的基本思想可以归纳为根据一个输入的、通常较小的纹理样本生成一个任意尺寸的、与输入样本具有类似统计学特征的输出纹理。

近年来几乎所有成功的基于样本的纹理合成算法都是基于马尔可夫随机场（Markov Random Fields，MRF）的[14]。在马尔可夫随机场假设下，纹理与一般图像最重要的两个区别是静态性（stationarity）与局部性（locality）。其中静态性是指，对于纹理上的任意两个不同位置 $x_1$ 与 $x_2$，它们各自周围一定大小的邻域应该在视觉上具有相似性；局部性是指纹理在 $x_1$ 或 $x_2$ 点的值仅与它们各自周围一定大小的邻域内的值有关。而一般图像并不一定满足这两点假设。

##### 二维纹理合成

Efros 与 Leung[15] 以及 Wei 与 Levoy[16] 分别提出了基于非参数化采样、以像素为单位进行最近邻匹配的纹理合成算法，他们的工作为近些年纹理合成算法的发展奠定了基础。Efros 和 Freeman[17] 以及 Liang[18] 随后将以像素为单位（pixel-based）的方式改为以块为单位（patch-based），使得合成的速度有了较大提高。Kwatra 等人[19] 提出了纹理优化的概念，通过将纹理合成的过程看作能量最小化问题他们的方法取得了质量很高的结果。从纹理合成的方式看，以像素为单位、以块为单位以及基于优化的方法是目前最主要的三种合成方式。

最基本的以像素为单位的纹理合成算法[15,16] 通常速度较慢。目前最有效的加速方法是利用纹理中的一致性（coherence)。Ashikhmin 等人[20] 最早研究了一致性在纹理合成中的应用，其基本思想是：在输出纹理中相邻的像素，有很大可能在输入纹理样本中也相邻，而不是来自随机的位置。Tong 等人[21] 提出了 *k*-coherence 的概念，他们的算法分为分析与合成两个阶段。在分析阶段，为输入纹理样本中的每个像素构造一个 “相似集”，即纹理样本中其它与该像素具有相似邻域的像素集合。在合成阶段与之前算法的区别是，首先，在将输入纹理的像素复制到输出纹理时，除了颜色信息还复制像素在输入纹理中的位置，这样相当于存储了每个输出纹理像素与输入纹理像素之间的匹配关系；其次，对于每个输出纹理像素，不需要在所有的输入纹理像素中进行最近邻匹配，而是在输出像素的相邻像素所对应的输入纹理像素的相似集中进行查找。*k*-coherence 对纹理合成的速度和质量均有很大提升。

与以像素为单位的纹理合成不同，以块为单位的方法每次将输入纹理中的一整块像素复制到输出纹理。由于这些像素块通常在输出纹理中会相互重叠，这类方法的主要难点是如何将这些像素块平滑地衔接以避免明显的缝隙出现。Liang 等人[18] 将像素块的重叠部分进行 alpha 混合以消除缝隙，但这种方法有可能会导致纹理细节变模糊。Efros 和 Freeman[17] 使用动态规划的方法寻找重叠像素块之间的一条最优衔接路径。Kwatra 等人[22] 通过 graph cut 改进了这一方法。

基于优化的方法一定程度上集合了另外两类方法的特性。这类方法以块为单位将输入纹理样本的像素复制到输出纹理中，但在混合这些像素块时却是以像素为单位通过优化一个二次能量函数进行。第一个纹理优化的算法[19] 取得了很好的质量，但是对一部分纹理样本来说仍然可能使细节变模糊。Han 等人[23] 将 *k*-coherence 与纹理优化相结合，不仅提高了纹理合成的速度，也对细节模糊的问题有所改善。

早期的纹理合成方法主要是顺序相关的（order-dependent)，亦即输出纹理中的像素必须以特定的顺序产生。这种合成方式本质上并不利于运用图形处理器的并行运算能力。Wei 和 Levoy[24] 提出一种迭代式的顺序无关纹理合成算法。在此基础上，Lefebvre 和 Hoppe 提出了一个图形处理器上的并行纹理合成算法[25]，使得很大尺寸的输出纹理可以被实时地生成。这一工作随后被进一步扩展到外观空间（appearance space)[26]。通常来说，基于样本的纹理合成算法难以处理图像中大尺度的结构化特征，因为这类特征并不符合马尔可夫随机场假设。Risser 等人[27] 提出一种多尺度的描述符（multiscale descriptor）使得纹理合成算法可以处理具有一定结构化信息的输入。

基于纹理优化的思想，Han 等人[28] 提出了一种多尺度纹理合成算法。在这种方法中，用户可以指定不同尺度上的多个输入纹理样本，算法的输出是一个具有多尺度特征的纹理。当用户在不同尺度下观察时，算法可以即时生成对应尺度的细节特征。对于多尺度上自相似的纹理，通过这种即时合成的机制，用户理论上可以无限放大纹理窗口，类似于分形图案。这种多尺度纹理的表示一定程度上解决了传统纹理表示的分辨率相关性，因而给予我们不少启发。但这种方法所能产生的结果仍然受马尔可夫随机场假设限制，局限于随机、重复性的纹理特征。

### 体纹理合成

体纹理（solid texture）是具有纹理特征的一类特殊体数据。虽然体纹理的属性呈三维分布，但其任意位置与朝向的二维截面仍然具有相似性，因此以一个二维纹理样本而不是三维样本作为输入生成一个三维体纹理是可能的。

早期的体纹理合成算法主要是基于参数化采样的。例如 Ghazanfarpour 和 Dischler[29,30] 试图匹配输入二维纹理样本与输出体纹理的频域特征。Heeger 和 Bergen[31] 则尝试匹配输入与输出多尺度上的颜色直方图。这些方法与早期的基于参数化采样的二维纹理合成算法具有类似的局限性，即并非任意纹理都可以用全局的参数（例如频谱、直方图等）完全描述，因此这类方法难以生成具有局部结构特征的体纹理。

Wei 将非参数化采样的二维纹理合成算法直接应用于体纹理[16,32,33]，尽管得到的结果质量并不令人满意，但其中 [33] 提出的用三个相互垂直的平面构成最近邻匹配的邻域定义为随后的体纹理合成算法普遍采用。Qin 和 Yang[34] 提出一种利用 Aura 矩阵的体纹理合成算法，其中 Aura 矩阵描述纹理中灰度值的概率分布。这种方法对于一部分纹理能够得到高质量的结果，但其主要问题是本质上该方法仅适用于单通道的灰度图像。为了合成多通道的彩色纹理，不同的颜色通道必须被去相关（decorrelate）并分别合成。然而彩色纹理中的颜色通道通常具有很强的相关性，分别处理独立通道后加以合并的方法会导致严重的色偏问题。

真实世界中的一类材质是由许多小的粒子（particle）镶嵌在某种背景介质中构成的，最常见的如人工石英石、水磨石等。Jagnow 等人[35] 提出了一种针对这种 “粒子镶嵌” 类型体纹理的合成算法。该方法利用体视学（stereology）的技术对二维图像样本分析，得到不同种类粒子的形状、颜色及空间分布，再在三维空间中产生新的粒子使其特征与输入样本的分析结果匹配。这种方法能够得到非常高质量的结果（如图1.3所示），遗憾的是，体视学的应用仅适用于少数几种材质，对于更一般的体纹理并不适用。

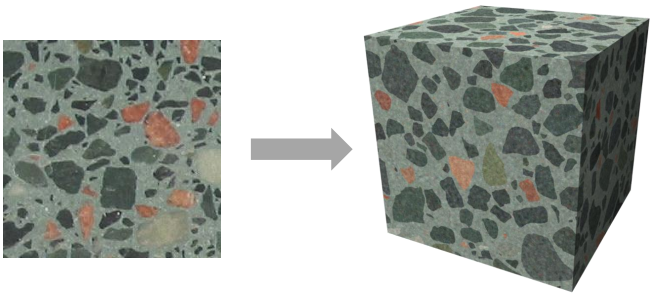


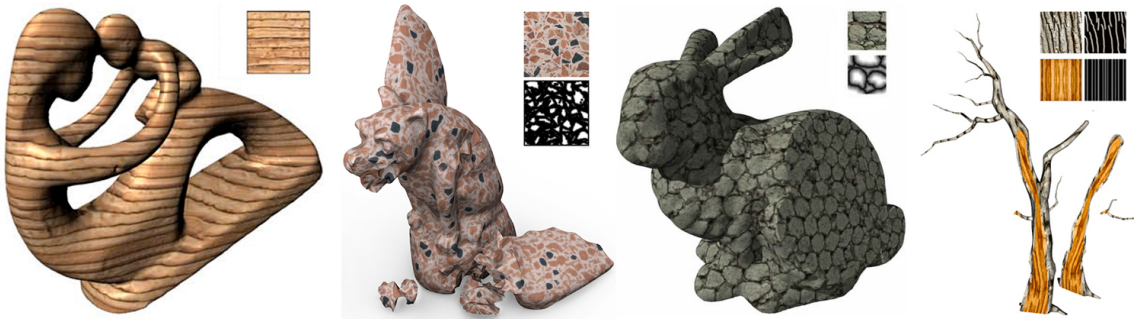
图 1.3　利用体视学技术由样本生成的体模型。(图片引自 [35])

Kopf 等人[36] 将二维纹理优化[19] 扩展到三维体纹理，并且将早期基于参数化采样方法中的颜色直方图匹配（histogram matching）引入非参数化采样的框架，有效地降低了后者生成无效纹理内容（garbage region）的几率。他们的方法对于很大一类体纹理都能取得较好的结果。Chen 和 Wang[37] 将颜色直方图匹配扩展到位置直方图与索引直方图的匹配，进一步提升了生成体纹理的质量。

以上方法生成的结果都是完整的体纹理。生成一个完整的高分辨率体纹理不仅需要花费很长时间，也需要占用大量的存储空间。Dong 等人[38] 注意到，对一个应用体纹理的物体来说，事实上仅仅物体表面附近的体素需要被合成，如果可以在保持空间一致性的前提下仅合成体纹理的一个局部区域，并且合成的速度足够快的话，就不需要预先合成整个体纹理。为此，在 [36] 的基础上他们提出一种 “按需合成” 的方法，即每次仅合成体纹理中靠近物体表面的体素。通过引入 *k*-coherence，该算法可以在图形处理器上高效实现。由于合成的速度足够快，即使物体表面发生变化（如物体被切开或破碎），新暴露出的表面附近的纹理也可以被实时合成。

单纯的纹理合成结果通常难以表现全局的结构特征，用户也不容易对结果加以控制。Xu 等人[39] 注意到，以三维模型上的一些显著几何特征作为导向进行纹理合成可以产生更好的视觉效果，并且突出物体本身的几何形状，他们提出了一种方法可以自动检测三维模型中明显的几何特征曲线（salient curves）并以此生成一个表面向量场，在随后的纹理合成时使纹理局部朝向与向量场相符，使得最终的纹理结果能够体现三维模型整体的结构特征。基于相似原因，Zhang 等人[40] 提出了一种以笔画作为导向的交互式体纹理合成算法。在他们的方法中，用户在三维模型表面通过一些简单笔划标记出局部的纹理朝向，算法自动根据这些笔划计算一个三维张量场，并生成局部朝向与张量场相符的体纹理。通过这种方法，用户可以直观控制体纹理的全局特征。与 [38] 类似，该方法也不需要预先生成完整体纹理，而是根据需要在物体表面即时合成。

图1.4展示了一些不同体纹理合成方法产生的结果。关于基于样本的纹理合成



Kopf et al.[36]　　Dong et al.[38]　　Chen and Wang[37]　　Zhang et al.[40]

图 1.4　不同体纹理合成方法生成的结果。图片均引自各自论文。

领域相关工作更详尽的综述可以参见 [14] 或 [41]。

#### 1.1.1.3　其它建模方式

Owada 等人[42] 设计了一套交互式的体建模系统，在该系统中，用户可以在三维物体的不同截面上指定二维参考图像，该系统利用二维纹理合成的方法生成截面上的颜色信息。但由于这种方法没有考虑到体数据在空间各个方向上的一致性，因此只能产生一些比较简单的结果。

Takayama 等人[43] 将 Praun 等人提出的重叠纹理（lapped textures）[44] 推广到三维，提出一种重叠体纹理。在这种表示中三维物体由四面体网格（tetrahedron mesh）构成，而在网格中的每个四面体上定义一个或多个小型体纹理。这种方法有两个好处：首先，由于每个四面体上都可以定义独立的体纹理且同一个体纹理可以反复出现，因此整个四面体网格的分辨率通常远远小于传统体纹理的规则三维网格，从而可以节省存储空间。另外，通过在物体内部的不同部分定义不同的体纹理，可以描述物体全局的材质属性变化，从而生成传统体纹理难以表达的具有一定结构信息的体模型。但是由于这种方法本质上仍然是基于离散采样的，为了表达丰富的内部细节和结构特征仍然需要高分辨率的四面体网格及体纹理，因此同样面临质量与存储空间的矛盾。

许多体模型内部信息的分布并不均匀，例如一个物体占据的空间与它的包围盒相比可能是稀疏的，在这种情况下传统的基于离散采样的体模型将整个立方体区域划分为均匀的三维网格会浪费很多存储空间。Benson 等人提出的八叉树纹理[45]（octree textures）利用八叉树数据结构对均匀采样的体数据进行压缩，可以有效地减少信息分布稀疏的体数据所需的存储空间，同时可以支持图形处理器高效的随机访问[46]。但由于八叉树结构本身需要的存储空间随着树的深度增加快速升高，对于有效信息稠密分布的一般体数据，存储八叉树结构的开销甚至有可能

超过原始体数据。因此，八叉树纹理主要被用于存储物体表面上的纹理而不适于表示内部细节丰富的体模型。

### 1.1.2　基于过程噪声的体数据表达

基于过程噪声的方法是图形学中一类重要的建模方法，目前仍然广泛应用于真实感绘制等许多领域[47–50]。根据 [51] 中的定义，这里的 “过程” 指 “算法或程序代码”，“噪声” 则指 “随机的、无结构的模式”，从而将过程噪声区别于数学意义上的随机函数。基于过程的体数据表达通过一定的噪声函数表示三维物体的几何属性 $\mathcal{D}$ 及材质属性 $C$，不需要像离散采样的体数据一样存储三维网格，因此在存储空间占用方面具有巨大优势。过程噪声是分辨率无关的，可以很容易的生成多尺度、甚至无限递归的细节特征。此外，噪声函数通常支持高效的随机访问，亦即空间中任意一点噪声函数的值都可以在固定的时间内求出，这在图形处理器无处不在的今天具有很重要的意义。

基于过程噪声的方法主要存在以下问题：首先，通常一种特定的噪声函数仅能够产生特定的一类体数据，很难找到一种噪声函数普遍适用于一般的体数据；其次，一种噪声函数通常包含多个参数，参数的不同取值会对生成的结果有很大影响，然而这些参数与最终结果往往并没有直观的对应关系，对一个没有经验的用户来说，为了产生一种特定的结果而调整这些参数通常是非常困难的；最后，尽管人们提出了很多不同的噪声函数，基于过程噪声的方法能够产生的结果主要还是局限于具有一定重复性和随机性的体纹理。因为这些原因，虽然这种方法可以很好地描述真实世界中随处可见的各种随机变化，却难以表现全局尺度的结构特征。在实际的内容创建过程中，专业人员通常需要将基于过程噪声的方法与其它方法共同使用。

另外，由于过程噪声函数是一个实值函数 $\mathcal{N}:\mathbb{R}^n \to \mathbb{R}$，为了表示有意义的材质属性，通常的做法是定义一个一维函数（有时称 color ramp）$\varphi:\mathbb{R} \to \mathbb{R}^p$ 将噪声函数的值映射为某材质属性：

$$C(x) = \varphi(\mathcal{N}(x)) \tag{1-1}$$

Perlin 在 1985 年提出了图形学领域第一个广为人知的过程噪声函数[47,52]，通过将不同倍频程的 Perlin 噪声叠加在一起可以产生多种不同的纹理效果，如云、火焰、大理石等。除了表现材质属性，过程噪声也被广泛用于定义复杂的物体几何[53]。之后的几十年里人们提出了大量不同的过程噪声函数，例如稀疏卷积噪声[54]、小波噪声[55]、各向异性噪声[56]。

Perlin 噪声[52]　　各向异性噪声[56]　　稀疏 Gabor 卷积[4]

图 1.5　不同过程噪声产生的体纹理。图片均来自各自论文。

为了解决传统过程噪声方法参数不直观、难以控制结果的缺点，Lagae 等人提出一种基于稀疏 Gabor 卷积的噪声函数[4,5]，允许用户以相对直观的方式调节噪声函数的频率分布以及各向异性程度。Lagae 等人在另外一项工作[57] 中尝试将基于样本的纹理合成思想与过程噪声结合，提出一种方法能够分析一个输入的图像并生成一个具有与输入图像类似视觉效果的过程噪声函数。不过目前这种方法仍然只能处理随机、各向同性的简单纹理。

图1.5例举了一些基于过程噪声方法生成的体模型。对过程噪声函数领域更详尽的文献综述可以参考 [51] 或 [58]。

### 1.1.3　基于结构信息的体数据表达

真实物体内部通常由多种不同类型的材质构成，材质与材质之间的界面上物体的材质属性函数 $C$ 会发生跳变。即使是同一种（通常意义下的）材质，其材质属性函数内部也可能有大量不连续的介面，例如木头中的纹路。我们称这种材质属性的不连续为物体的内部结构信息。经过插值的离散采样体数据和基于过程噪声的体数据均将物体的材质属性表示为空间中的连续函数，因而难以准确表达这样的结构信息。

与前面介绍的两类体数据表达不同，对基于结构信息体数据表达与建模的研究直到近些年才开始出现。

Cutler 等人在 2002 年[3] 最早提出一种基于层结构（layered structure）的体数据建模方法。这种方法根据给定物体的边界计算一个符号距离函数（signed distance function），距离函数中不同的等值面将空间分割为多个层，继而可以为每个层分别定义不同的材质。Cutler 等人还设计了一种脚本语言用于定义层结构以及各层的属性。通过实现类似构造实体几何（CSG）的操作，他们的方法可以将多个相对简单的物体边界经布尔操作组合成更复杂的物体边界。在这种体数据表示的基础上，他们实现了一些基于过程的操作以模拟真实物体的风化、腐蚀、破

图 1.6　Cutler 等人的方法生成的具有层结构的体模型。图片引自 [3]

碎、变形等效果。总的来说，Cutler 等人的方法允许用户直接定义物体中的层结构，从而可以快速创建出一些用传统方法难以产生的体模型，尽管这种方法能够描述的对象主要局限于具有层结构的物体（如图1.6所示）。此外，在具体实现时，基于层结构的体模型需要被预先转化为四面体网格的形式以便绘制。而对于一个比较复杂的模型，高分辨率的四面体网格带来的存储开销可能大幅提高，并且使得对模型的交互式切割等操作更加费时。

Takayama 等人在 2010 年[6] 提出一种名为 Diffusion Surfaces 的体数据表达。这种方法可以看作是一种名为 Diffusion Curves[59] 的二维矢量图表示在三维的扩展。其基本思想是允许用户直接在物体内部创建一些稀疏分布的曲面并且在曲面上定义颜色，而物体内部任意一点的颜色由定义在附近曲面上的颜色插值得到。这种表达利用了物体内部材质属性的不连续形成曲面的特点，使用户可以直接用曲面描述这种结构，因此能够表达很大一类真实物体的结构信息。为了方便用户创建具有复杂内部构造的物体，特别是具有对称性的物体，Takayama 等人设计了一套直观的基于笔划的交互式系统，进一步降低了体建模的困难程度。结果显示，根据参考图像和一些简单的交互，用户可以快速创建多种具有高度结构化特征的体模型（如图1.7所示）。这种方法最主要的问题在于，由于物体内部任意一点的颜色是由附近稀疏分布的曲面上定义的颜色插值得到的，因此在远离曲面的地方颜色仅能平缓变化，缺少丰富的细节特征。其次，由于定义颜色的曲面在目前的实现中是三角形网格，因此体模型中的结构特征本质上说并不是分辨率无关的。最后，这种数据表达并不支持高效的随机访问：每当物体表面发生变化时，对新暴露出的表面上的每一个顶点，都需要在图形处理器上进行一次单独的绘制过程以计算该顶点的颜色。

我们在表1.1中列出了上文介绍的几类体模型表示，并从表达能力，建模方式，存储空间占用，以及绘制效率这四方面对它们进行了分析比较。

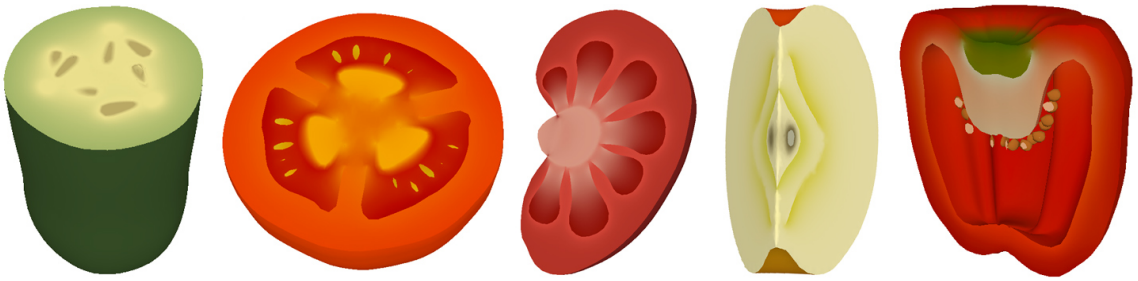


图 1.7　Takayama 等人的方法生成的体模型。图片引自 [6]

表 1.1　不同体数据表达的对比。

| | 离散采样 | 过程噪声 | Cutler et al.[3] | Takayama et al.[6] |
|---|---|---|---|---|
| 表达能力 | 能表达复杂的材质属性变化，受分辨率限制，难以描述结构信息 | 分辨率无关，支持多尺度细节，仅能表示特定种类的纹理，不能描述全局结构 | 限于层结构，每层可独立定义材质属性，分辨率无关 | 直接表达材质属性不连续介面，难以描述细节颜色变化 |
| 建模方式 | 手工建模困难，以直接采集或纹理合成为主 | 调节参数，不够直观 | 脚本语言，基于过程方法，物理模拟 | 基于笔划的交互式界面，参考图像，自动对称性分析 |
| 存储空间 | 随分辨率提高呈立方增长，是限制表达能力的主要因素 | 内存开销极小 | 需要存储四面体网格，潜在内存开销较大 | 仅需存储稀疏的颜色曲面，内存开销较小 |
| 绘制效率 | 图形处理器硬件支持高效绘制 | 支持随机访问，比较高效 | 表面发生变化后需要预计算 | 表面发生变化后需要预计算 |

### 1.1.4　分辨率无关的表面数据表达

位图（bitmap）与矢量图（vector graphics）是二维图像最主要的两种表示形式。矢量图主要通过基于数学公式的几何形状，例如曲线、多边形、颜色渐变等表现图像内容。与位图相比，矢量图的一个重要优点是图像中的形状特征不会因为放大而变模糊。这种分辨率无关性也是我们在设计新的体数据表达时期望的目标之一，因此在这里我们对相关领域的工作做一简要回顾。

早期的矢量图主要用于创建相对简单的几何形状，其中填充均匀的、线性的或是径向的颜色渐变，典型的例子包括字体、示意图、地图、卡通等。除了手工创建的方式，许多研究者提出了将位图图像自动矢量化的方法[60–62]，这些方法的基本思路是将输入图像用图像分割的方法划分为多个内部颜色变化平缓的区域，

再用多边形拟合等方法生成描述区域形状的几何元素。但是由于这些方法使用的矢量图形式对颜色变化的表达能力有限，因此为了矢量化具有复杂颜色变化的图像通常需要产生大量细小的区域以及几何形状，使得矢量图存储紧凑以及易于编辑的优势不复存在。

近年来许多研究工作致力于真实照片的矢量化[59,63–67]。为了有效地表示真实图像中复杂的颜色变化，人们提出了一些新的矢量图形式。Adobe Illustrator 以及 Corel CorelDraw 中提出了梯度网格（Gradient Mesh）的矢量表达，允许用户在一个平面四边形网格的顶点上指定颜色，网格面片上的颜色则由顶点上的颜色平滑插值得到。基于梯度网格的表达，Sun 等人[65] 以及 Lai 等人[66] 分别提出了高质量的图像矢量化方法。Xia 等人[67] 提出一种支持曲线边的三角形网格取代梯度网格中使用的四边形网格，使得网格边能更好地与输入图像中的尖锐特征对齐。Orzan 等人[59] 提出一种名为 Diffusion Curves 的矢量表达，与基于网格的方法不同，他们的方法允许在平面中一些曲线上定义颜色，这些颜色通过 “扩散”（diffusion）平滑地填满整个图像。

许多商业软件都可用于直接创建矢量图内容或自动矢量化位图图像，例如 Adobe Live Trace、Adobe Flash、VectorEye、Inkscape 等。W3C 提出的 SVG[68]（Scalable Vector Graphics）是目前广泛使用的矢量图文件格式标准。

和位图相比，现有的矢量图表示通常不支持高效的随机访问，这使得在很长一段时间内矢量图的绘制主要都是在 CPU 上进行的。Loop 和 Blinn[69] 在 2005 年提出一种在图形处理器上高效绘制分辨率无关的曲线及封闭形状的方法。Nehab 和 Hoppe[70] 以及 Qin 等人[71] 进一步研究了一般 SVG 格式矢量图在图形处理器上的绘制，这使得矢量图可以像位图一样作为纹理映射在任意三维物体的表面。Jeschke 等人[72,73] 则尝试利用图形处理器绘制 Diffusion Curves 形式的矢量图。

除此以外，还有一些图像表示介于纯粹的矢量表示与位图表示之间[74–78]。本质上这些表示仍然存储一个位图图像，但通过使用某种形式的矢量形状（如曲线）标记出图像中的显著特征，并且在放大图像时使用特殊的插值方法利用这些额外的矢量信息，使得放大后的位图图像中仍然能保持相应的特征清晰。

## 1.2　本文工作

我们认为，一种理想的体数据表达应该具有**表达能力强、建模方便、存储开销小、绘制效率高**的特点。这样一种表达将使得人们可以容易地创建高度复杂、具有丰富细节的体模型，改变目前图形学领域高质量体模型短缺，体数据表示应

用范围狭窄的现状。

通过对相关工作的分析，我们认为体建模面临的一个关键问题是对真实物体中结构信息的描述。传统的离散采样表达或过程噪声表达均关注于底层的材质属性变化，受制于体数据庞大的信息量以及交互的困难，用户很难直观、准确和高效地创建新的内容或是编辑已有的内容。而基于结构信息的方法通过抽离结构信息使用户可以站在更高的层次把握一个复杂的对象，由此可以衍生出相比传统更加方便和灵活的建模方式。

真实世界中的物体千变万化，是否能够为不同类型的体结构给出统一形式的描述直接决定了一种体数据表达的通用性。而这种表达的存储空间是否紧凑，能否在确保绘制效率的前提下最大限度体现体模型的优势（例如实时在物体上切割出任意截面），这些与实现密切相关的问题也从另一个方面决定了这种数据表达在实际当中所能描述的物体复杂程度。

通过观察发现，真实物体内部的材质属性变化可以划分为连续变化与不连续跳变两类。其中不连续的跳变通常对应于两种不同材质之间的介面，这些二维曲面将物体所在的空间划分为许多独立的区域，形成物体独有的特征结构。区域内部的材质属性变化平缓，而为了准确描述区域边界处的不连续变化则需要一种分辨率无关的表达方式。

根据以上分析，本文提出了一种基于区域结构的体数据表达，并且深入研究了由该表达衍生出的新的体数据建模方法，以及运用该表达的体模型在紧凑存储与高效绘制等方面的问题。使用这种表达，我们成功地创建出大量具有传统方法难以获得的复杂度与质量的体模型。进一步研究发现，基于区域结构的表达不仅适用于传统意义上的体数据，也可以应用到更一般意义的具有体特征的对象。作为一项专题研究，我们尝试将区域结构的思想用在头发的几何分析与建模上，取得了良好的效果。

本文的主要贡献如下：

- **第2章：** 提出了一种基于区域结构的体数据表示。该表示能够准确描述物体内部跨越多个尺度的区域结构特征，每个区域可以独立地定义内部材质属性，区域边界可以具有任意复杂的几何形状，并且材质属性在区域边界的不连续跳变以分辨率无关的方式表达，因而体模型在绘制时可以任意放大而不会出现模糊等问题。
- **第3章：** 新的体数据表示改变了传统的体建模方式。通过深入研究，我们提出了一套完整的基于区域结构的体建模流程，其中包括一种基于 XML 的体对象标记语言以及多种交互式建模工具。结果显示，通过我们方法创建的体

模型，在结构复杂程度与细节丰富程度等方面均超越了传统的方法，这使得手工创建具有高度真实感的复杂体模型成为可能。除手工建模方式外，我们也提出了不同的算法可以将其它形式的数据转换为基于区域结构的表达，以便进一步编辑。

- **第4章：** 我们提出的实时绘制算法在任意放大的情况下仍然可以保持体模型中材质属性不连续的区域边界清晰，同时支持高质量的反走样以及允许用户根据需要控制边界的模糊程度。此外，与传统方法相比，区域结构体模型在绘制时不需要预计算，因此物体边界可以任意动态变化。通过对数据结构的优化，区域结构体模型具有低廉的存储开销。一个包含多个具有丰富细节体模型的完整场景仅需占用主流图形处理器内存的很小一部分，并且可以实时地绘制。
- **第5章：** 将基于区域结构的表达应用于头发的几何分析与建模。通过对现有头发几何模型中区域结构的分析和抽取，我们提出了一种基于样本的头发几何建模方法。这使得用户首次可以利用现有的头发几何模型快速创建具有相似特征的新模型，从而解决了已有头发几何模型难以重用和编辑的问题。这也显示出基于区域结构的表达作为一种更通用数据结构的潜力。

# 第 2 章　基于区域结构的体数据表达

本章将定义基于区域结构的体数据表达（以下简称区域体数据表达)。我们首先给出区域结构的定义并分析真实世界物体区域结构的特性（2.1节)，之后概述区域体数据表达的核心思想与形式（2.2节)，最后我们给出区域体数据表达中主要元素的具体定义（2.3至2.5节)。

## 2.1　真实物体的区域结构

真实世界中的一部分物体是由单一均匀材质构成的，例如金属、玻璃、塑料制品等。然而更多的复杂物体由非均匀的材质甚至多种不同的材质构成，这里的“材质”一词既可以指构成物体的具有某种**特定属性**的物质，例如构成地壳的不同岩层，也可以指物体内部具有一定**逻辑语义**的组成部分，例如构成人体的皮肤、肌肉、骨骼等。物体内部的不同材质形成一个个独立的区域，在相邻区域之间的介面处物体的属性往往发生跳变，从而形成清晰而明确的外观特征。通过观察发现，这样的区域结构在真实物体中是广泛存在的。

从几何的角度看，真实物体中的区域结构具有以下特性:

**分辨率无关：** 区域的边界，亦即不同材质之间的介面，通常表现为物体外观属性的跳变，不论观察者距离物体多近仍然能看到不同材质之间清晰明确的分界。这即是说区域边界是分辨率无关的（resolution-independent)。从频率的角度，区域边界处物体属性的变化对应于无限高的频率，因此传统的基于离散采样的体数据表示很难准确描述物体的这类特征——即使采用相当高的采样频率（同时意味着巨大的存储空间消耗)，在绘制放大后的局部时仍然无法避免走样或模糊等问题。

**形状复杂：** 物体内部的每个独立区域都可以看作为一个三维实体，其边界曲面可能拥有任意复杂的形状，包括不可导的几何特征，如折痕（crease)，角点（corner）等。在一个复杂的区域结构中，多个不同区域可能彼此相邻，使得所有区域的边界在空间中形成一个曲面网络（surface network)，其中含有大量非流形的结构，例如 T 形交叉。

**多尺度：** 对一个物体内部组成的描述通常要在特定的尺度层次下才有意义。例如从宏观尺度看，人体可以分成皮肤、肌肉、骨骼等区域；而从微观尺度看，人体是由一个个独立的细胞组成的，每个细胞内部可以进一步分解为更细小的区

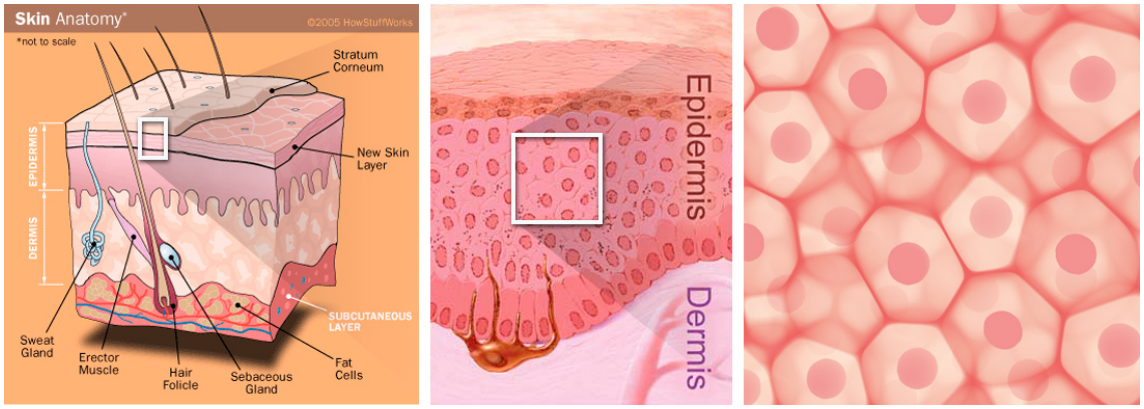



图 2.1　皮肤中不同尺度下的区域结构。左图由 HowStuffWorks 拥有版权，中图由 National Cancer Institute 拥有版权，右图为本文方法生成的结果。

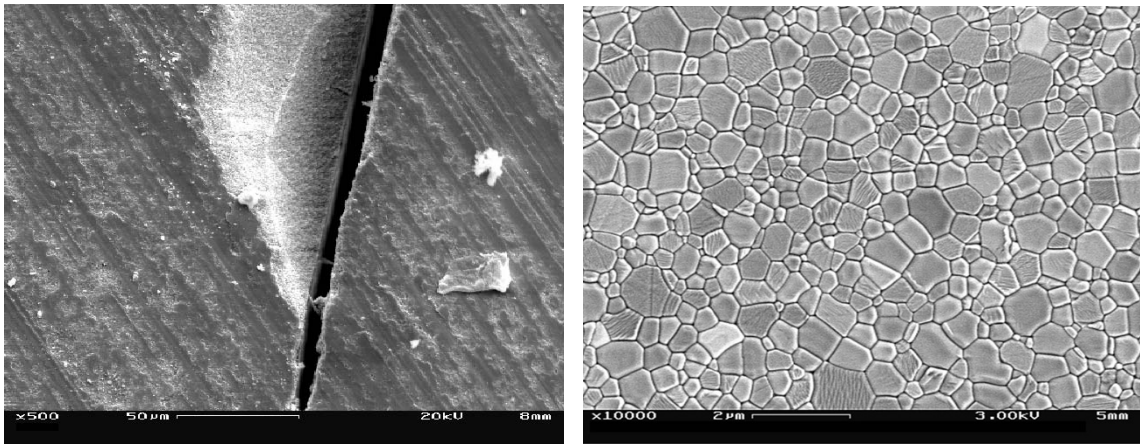



图 2.2　扫描电子显微镜下的三氧化二铝陶瓷表面。在 500 倍（左）与 10000 倍（右）放大倍率下呈现出不同区域结构。图片引自 [79]

域结构如细胞膜、细胞核等。物体不同尺度的特征相互嵌套，形成多层次的区域结构。图2.1和2.2展示了一些真实物体内部的区域结构。

## 2.2　区域结构体数据表达概述

真实物体的复杂区域结构很难使用传统的体数据表达描述，为此我们提出一种新的基于区域结构的体数据表达，其核心思想是将三维物体 $O$ 所在的空间划分为 $\kappa$ 个互不相交的区域。我们将这样一个区域划分记为 $\mathcal{P}$，并用 $\mathcal{P}_i$ 表示第 $i$ 个区域。由于这些区域互不相交且完全覆盖 $O$，可知对任意 $O$ 内部的点 $x$，存在唯一的 $i \in \{1,\ldots,\kappa\}$ 使得 $x \in \mathcal{P}_i$。同时我们定义 $\kappa$ 个三维函数 $C_i$ 用于分别描述每个区域内部的材质属性变化。物体中任意一点 $x$ 处的材质属性则可以表示为：

$$C(x) = C_i(x), \quad i = \{i \,|\, x \in \mathcal{P}_i\} \tag{2-1}$$

为描述构成区域边界的曲面，一种直接的想法是使用显式的曲面表示形式，如参数曲面、多边形网格等。然而，尽管显式曲面可以灵活地表达复杂的形状，判断空间中任意一点位于曲面的哪一侧却相对困难甚至是不可能的（对于非闭合曲面）。另外，显式表示难以处理曲面拓扑结构的变化，这将对体模型的编辑甚至动态体模型的实现造成障碍。基于这样的考虑，我们使用符号距离函数表示的隐式曲面描述区域边界（2.3节）。

空间中的区域分割在计算几何与图形学领域均有广泛而深入的研究。然而常见的空间划分方法如八叉树[80] 和 $N^3$ 树[25] 只能将每个立方形的空间节点均匀剖分为 $n^3$ 个立方形子节点，BSP 树[81] 或 kd 树[82] 通过迭代地用平面将空间分为两个半空间实现剖分。这些方法均无法实现任意形状的区域划分。另一方面，离散采样的体数据可以在每个体素上存储区域标记，这相当于将图像分割应用于三维。然而体素模型的离散采样本质决定了用这种方法得到的区域边界是分辨率相关（resolution-dependent）的。我们为此提出一种以符号距离函数作为基础构件的区域划分数据结构（2.4节）。

真实物体内部除了对应于区域边界的材质属性跳变，也可能出现平缓过渡的材质属性变化。这类变化我们通过每个区域内部独立定义的三维函数描述（2.5节）。

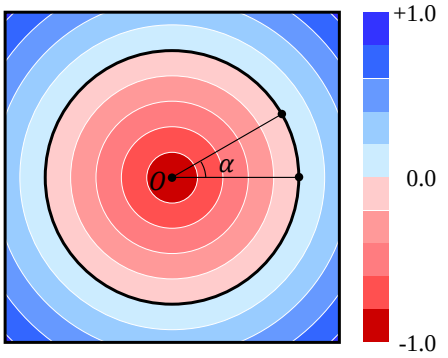
## 2.3　符号距离函数的定义

令 $S$ 为三维空间 $\mathbb{R}^3$ 的一个子集，则 $S$ 的边界 $\partial S$ 可以通过定义在三维空间上的实值函数 $\mathcal{D}:\mathbb{R}^3\to\mathbb{R}$ 隐式定义：

$$\begin{aligned}\mathcal{D}(x)<0 &\iff x\in\mathring{S}\\ \mathcal{D}(x)=0 &\iff x\in\partial S\\ \mathcal{D}(x)>0 &\iff x\in\mathbb{R}^3-S\end{aligned}\tag{2-2}$$

其中 $\mathring{S}$ 表示 $S$ 的内部（开核）。若 $|\mathcal{D}(x)|$ 是在某种度量下 $x$ 点与 $\partial S$ 的最短距离，则称 $\mathcal{D}$ 为 $S$ 的一个符号距离函数（Signed Distance Function，SDF），且 $\mathcal{D}$ 的零值面（值为 0 的点集）即为 $\partial S$。图2.3展示了一个圆形及其对应的二维符号距离函数。

注意这里我们规定 $\mathcal{D}$ 的符号在 $S$ 内部为负、$S$ 外部为正仅是为了在本文中表述一致。在隐式表面的相关工作中为表达方便也存在其它形式的人为约定，例如 [83] 中规定 $\mathcal{D}$ 的符号总为正，并且用 $\mathcal{D}(x)$ 的值大于、等于或小于 1 来区分 $x$ 处




显式表示：

$$x = (\cos\alpha, \sin\alpha), \alpha \in [0, 2\pi]$$

隐式表示：

$$\mathcal{D}(x) = \|x\| - 1, x \in \mathbb{R}^2$$

图 2.3　单位圆形的显式以及隐式表示。左图中不同的颜色表示符号距离函数中不同的值，白色线条代表距离函数中的等值线。

于外部、边界上或是内部。符号距离函数表示的隐式曲面在图形学中有广泛的应用，包括几何造型[83–89]、碰撞检测[90,91]、动态模拟[92] 等。

隐式表面与显式表面均可以描述复杂的表面形状，两者在多数情况下亦可以相互转化。但是与显式表面相比，以符号距离函数表示的隐式表面具有一个重要的优点，即对空间中任意一点，凭借距离函数在该点的符号即可立即判断它处于实体内部或是外部。从另一角度看，一个符号距离函数的零值面将三维空间划分为两个区域：距离符号非负的区域 $\mathcal{P}_{\oplus}$ 以及距离符号为负的区域 $\mathcal{P}_{\ominus}$。因此，在基于区域结构的体数据表达中，我们选择符号距离函数作为基础构件（building-block），与其它数据结构相结合构造任意复杂的区域划分。

针对不同的应用场景，符号距离函数可以有多种不同的形式，以下我们将给出基于区域结构的体数据表达中的符号距离函数定义。

### 2.3.1　简单符号距离函数

符号距离函数通常可以通过三类方法定义：第一类方法是离散采样，即在空间中一个均匀的三维网格上存储符号距离值。除了均匀网格，Frisken 等人提出的自适应采样的距离函数[93]（Adaptively-sampled Distance Function，ADF）使用八叉树取代均匀网格对空间进行自适应的划分，并在八叉树的每个单元格中存储距离函数的采样值，从而可以将有限的存储空间更多地分配给细节丰富的区域；第二类方法是数学函数，亦即 $\mathcal{D}(x)$ 具有解析的函数表达，如图2.3中的符号距离函数可以表达为 $\mathcal{D}(x) = \|x\| - 1$ 的形式；第三类是过程方法，即通过特定的算法过程计算 $\mathcal{D}(x)$ 的值。后两类方法虽然对存储空间的要求非常小，但能够隐式表达的曲面类型十分有限，并且求值的时间复杂度通常更高。因此在区域体数据表达中，我们采用离散采样表达，并通过多种方法尽可能降低存储空间消耗。

### 三维网格定义

定义 $\mathcal{G}$ 为三维空间 $\mathbb{R}^3$ 中一与坐标轴对齐的规则网格。$\mathcal{G}$ 有 $d_W \times d_H \times d_D$ 个节点，用 $x_{ijk}$ 表示第 $(i, j, k)$ 个节点的空间位置，且有：

$$x_{ijk} = \left(\frac{i-1}{d_W-1}, \frac{j-1}{d_H-1}, \frac{k-1}{d_D-1}\right),\ \ i \in \{1, \ldots, d_W\}, j \in \{1, \ldots, d_H\}, k \in \{1, \ldots, d_D\} \quad (2\text{-}3)$$

亦即 $x \in [0,1]^3$ 的单位立方形区域被网格 $\mathcal{G}$ 均匀划分为 $(d_W-1)\times(d_H-1)\times(d_D-1)$ 个单元格（cell），如图2.4所示。

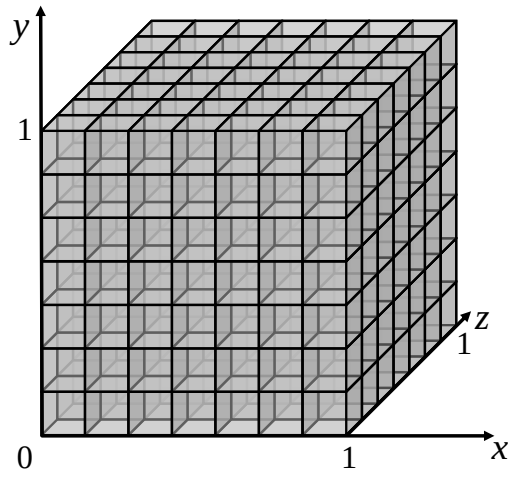



图 2.4　含有 $8\times8\times8$ 个节点的规则网格将 $[0,1]^3$ 区域划分为 $7\times7\times7$ 个单元格。

### 距离函数的插值

我们将符号距离函数 $\mathcal{D}$ 定义在网格 $\mathcal{G}$ 的节点上，网格内部任意点（非节点）的符号距离值由附近节点上的值插值得到。我们使用三立方（tricubic）插值以最大程度地保证区域边界的平滑与连续，计算任意点的值需要用到该点所在网格单元格及周围 26 个单元格的总共 64 个节点（$4\times4\times4$）上定义的距离值。

需要注意的是，尽管我们通过离散采样的方式定义符号距离函数，但这与离散采样的体数据是不同的概念。在离散采样的体数据中三维网格上存储的是某种材质属性（例如颜色），这些属性在物体内部不同组成材质之间的介面处会发生不连续的跳变，而这种跳变在插值时会无法避免地造成走样（aliasing）或模糊等问题。而离散采样的符号距离函数在零值面处距离值是连续变化的，因而即使网格分辨率（采样频率）相对较低也不会造成走样及模糊，如图2.5所示。

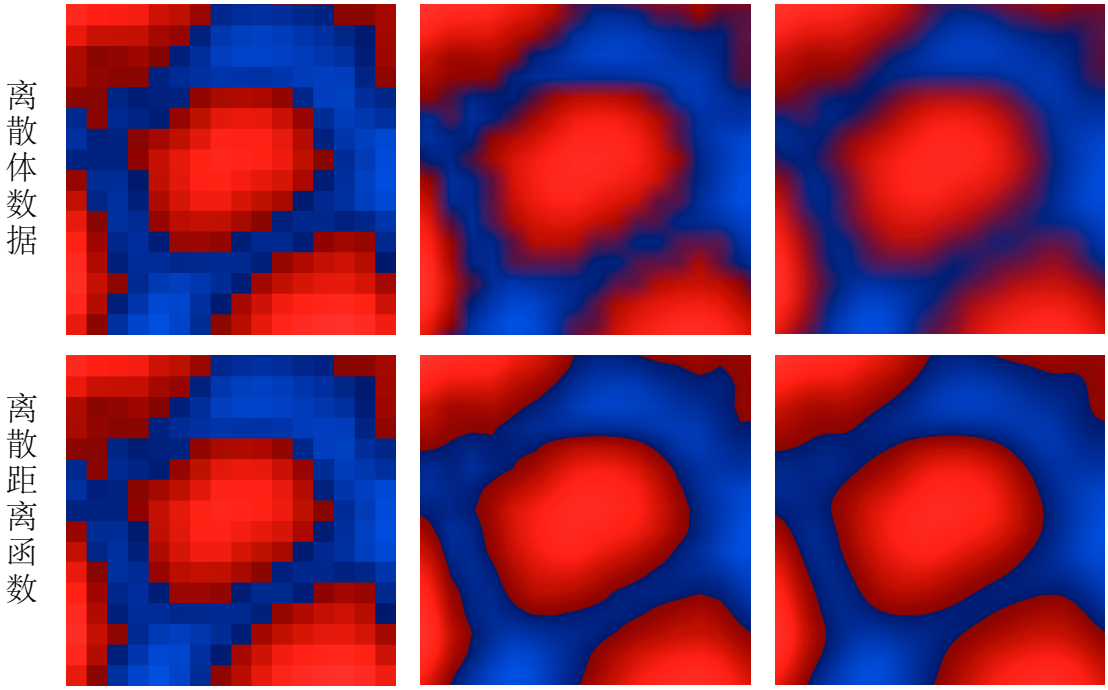


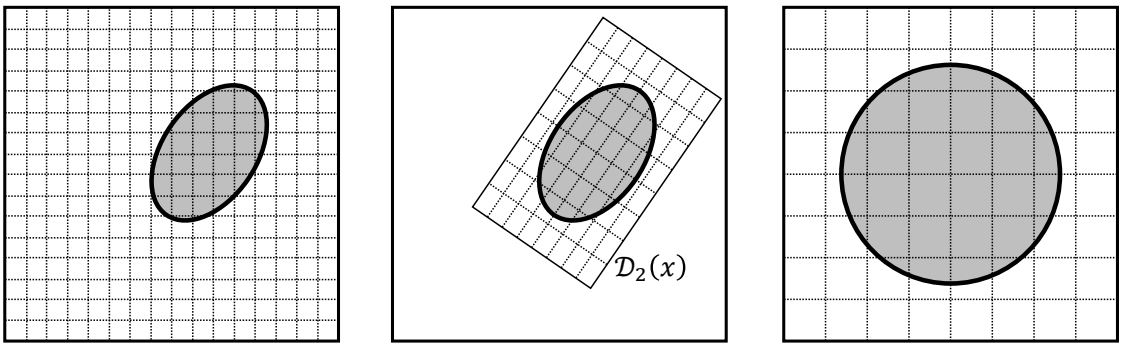

图 2.5　离散采样的符号距离函数与离散采样的体数据对比。

**距离函数的线性变换**

由于定义在三维网格上的符号距离函数对于定义域 $[0,1]^3$ 内的各处采用相同的采样频率，而采样频率与距离函数能够表达的隐式表面复杂程度与细节尺度直接相关。过高的采样频率虽然能够表达更多的表面细节，却会同时显著提高存储空间占用。特别是在区域结构体数据表达中，由于区域边界的复杂性，单个的符号距离函数通常仅被用于表达一小部分边界表面，而这一部分可能仅占空间中一个很小的区域，使用定义在整个体空间的符号距离函数在这种情况下难免造成存储空间的浪费或是被迫降低采样频率导致的表面细节丢失。

为了解决该问题，我们为每个符号距离函数额外定义一线性变换 $\mathbf{M}$。直观上看，符号距离函数 $\mathcal{D}$ 被嵌入一个原始位置与立方区域 $[0,1]^3$ 重合的定向包围盒（Oriented Bounding-Box）中，而 $\mathbf{M}$ 定义了该包围盒在三维空间 $\mathbb{R}^3$ 中的几何变换。距离函数的线性变换允许用户仅对需要表达的局部区域边界所在的定向包围盒进行采样，从而可以更有效地利用有限的存储空间，这在一些扁平或狭长的区域边界表示中尤其明显，如图2.6所示。

对于处在符号距离函数的包围盒以外的点，我们定义其符号距离为一常数 $\epsilon_D > 0$。由此，对于一个定义在 $[0,1]^3$ 的符号距离函数 $\mathcal{D}$ 及一线性变换矩阵 $\mathbf{M}$，




(a) $\mathcal{D}_1(x)$, $17 \times 17$　(b) $\widetilde{\mathcal{D}}_2(x)$, $9 \times 9$　(c) $\mathcal{D}_2(x)$, $9 \times 9$

图 2.6　对于只占很小区域的符号距离函数 (a)，我们可以在其周围定义定向包围盒 (b) 并在包围盒内均匀采样 (c) 以避免存储空间浪费。图中灰色部分为距离函数中符号为负的区域，粗黑实线为零值面，以下同。

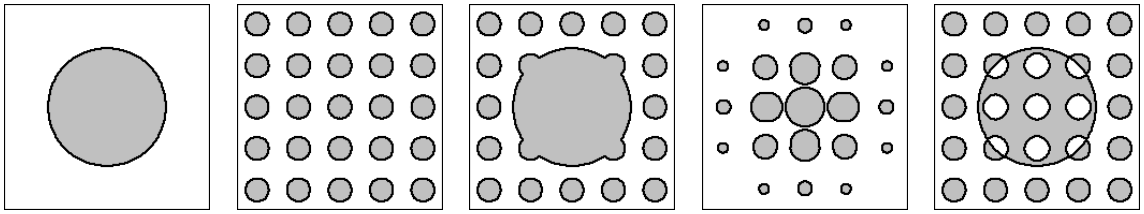
我们可以通过下式计算空间中任意一点 $x$ 的符号距离 $\tilde{\mathcal{D}}(x)$:

$$\tilde{\mathcal{D}}(x) = \begin{cases} \mathcal{D}(x \cdot \mathbf{M}), & (x \cdot \mathbf{M}) \in [0,1]^3 \\ \epsilon_D, & (x \cdot \mathbf{M}) \notin [0,1]^3 \end{cases} \tag{2-4}$$

### 2.3.2　复合符号距离函数

单个离散采样的符号距离函数理论上仅能表达封闭的、可导的隐式表面。然而实际物体内部的区域边界通常并非处处光滑可导，可能会出现曲折（crease）、角点（corner）等尖锐几何特征。其次，区域边界上几何特征的尺度通常并非均匀分布，同一个区域边界表面的不同位置可能几何特征尺度相差悬殊。如果用一个简单的符号距离函数表达这样的边界表面通常会面临存储空间与采样频率之间难以调和的矛盾。另外，一些复杂的区域边界通常包含许多相似的特征，或是具有类似分形几何的多尺度自相似特征，使用单一的符号距离函数难以利用这种重复性特征带来的信息冗余，甚至无法描述分形中无限递归定义的边界形状。为了表达以上这些复杂的区域边界，我们在本节中提出复合符号距离函数的定义。复合符号距离函数在逻辑上仍然是一个距离函数，但可能是由多个不同的符号距离函数共同定义的。

在几何造型领域，隐式表示的一个独特优势就是可以将多个相对简单的几何形体通过代数运算或布尔运算叠加以生成一个更复杂的几何形体（例如计算机辅助设计中广泛使用的构造实体几何方法[83]）。与之类似，我们可以通过多个符号距离函数的运算得到一个新的距离函数，从而定义更加复杂多变的区域边界。给定 $k$ 个符号距离函数 $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$，以及运算 $\mathcal{F}$，我们可以将复合符号距离函数



$\mathcal{D}_a$　　$\mathcal{D}_b$　　$\min(\mathcal{D}_a, \mathcal{D}_b)$　　$\mathcal{D}_a + \mathcal{D}_b$　　$\mathcal{D}_a\mathcal{D}_b$

图 2.7　距离函数的叠加

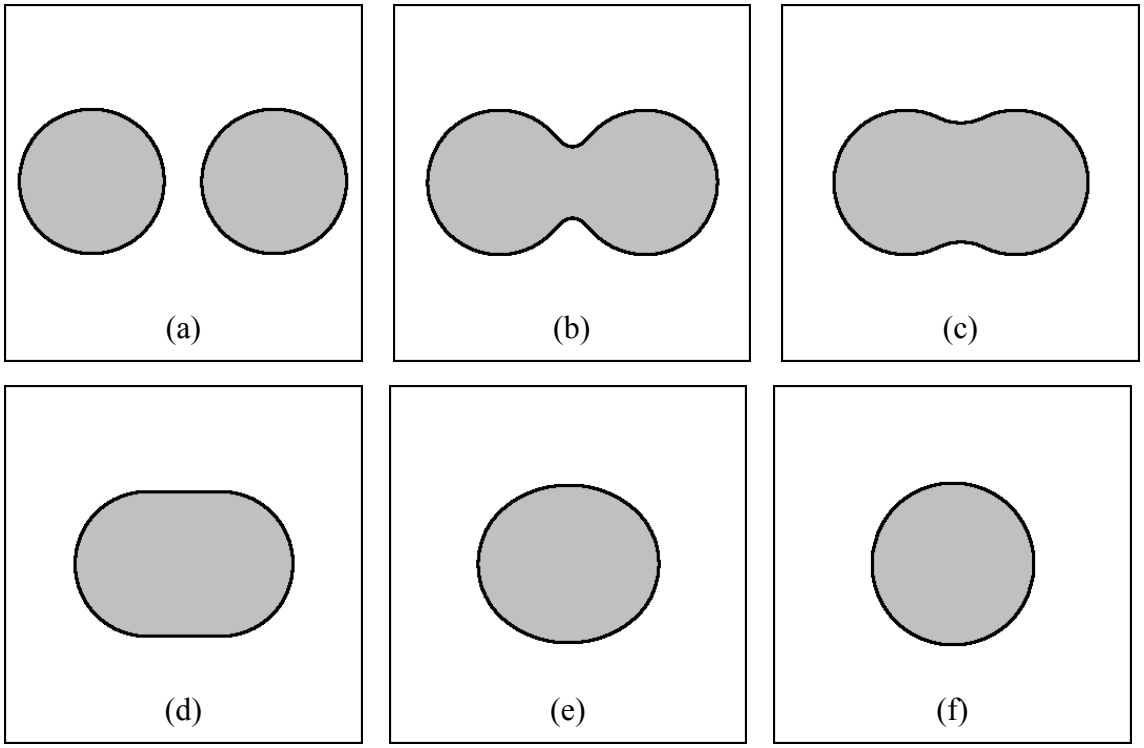



图 2.8　两个基元距离函数使用 Wyvill 函数软混合。由 (a) 至 (f) 两个距离函数的包围盒逐渐靠近直至完全重合。

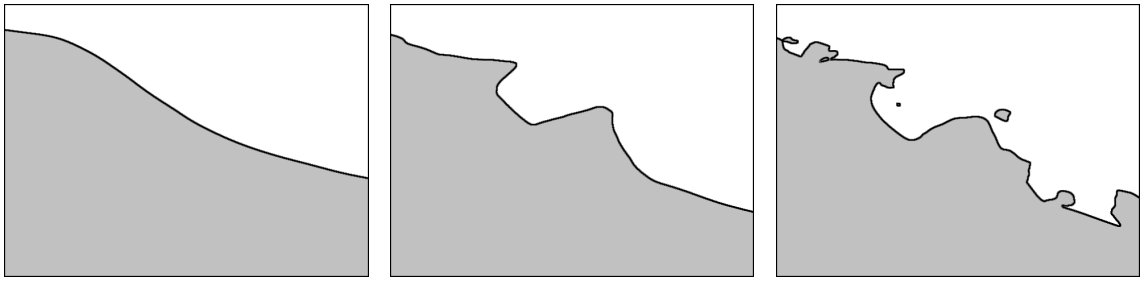
表示为:

$$\mathcal{D}(x) = \mathcal{F}(\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k). \tag{2-5}$$

这里我们称 $\{\mathcal{D}_i\}_{i=1}^{k}$ 为 “基元符号距离函数”（primitive signed distance function）。

对于同样的 $k$ 个基元距离函数，不同的运算 $\mathcal{F}$ 将产生不同的的结果。图2.7例举了两个基元符号距离函数不同的代数或布尔运算结果。除了代数或布尔运算，$\mathcal{F}$ 也可以是软混合（soft blending）函数，例如 Blinn[84] 或 Wyvill[88] 等人提出的混合函数。图2.8展示了两个符号距离函数利用 Wyvill 函数[88] 进行混合的结果。

除了对不同的基元距离函数进行运算，我们也可以利用同一个基元距离函数作为输入产生类似分形（fractal）的自相似区域边界。首先我们定义一个连续的



$k=1$　　$k=2$　　$k=3$

图 2.9　利用距离函数叠加得到的类似分形的自相似区域边界，其中 $a_1=0.5, a_2=0.3, b_1=4, b_2=16$。

且平移不变（translation invariant）的符号距离函数 $\dot{\mathcal{D}}$:

$$\dot{\mathcal{D}}(x+(n,0,0))=\dot{\mathcal{D}}(x+(0,n,0))=\dot{\mathcal{D}}(x+(0,0,n))=\dot{\mathcal{D}}(x),\ \ n\in\mathbb{Z} \tag{2-6}$$

这样的符号距离函数直观上类似于可无缝拼接（tileable）的二维纹理。接着我们可以以 $\dot{\mathcal{D}}$ 作为基元距离函数定义如下复合距离函数:

$$\mathcal{D}(x)=\mathcal{F}(a_1\dot{\mathcal{D}}(b_1x)+a_2\dot{\mathcal{D}}(b_2x)+\cdots+a_k\dot{\mathcal{D}}(b_kx)). \tag{2-7}$$

图2.9展示了 $\mathcal{F}$ 为求和运算时2-7式定义的复合符号距离函数。

2-7式可以看作是一个基元距离函数的多个实例（instance）进行不同缩放后叠加在一起得到一个新的距离函数。通过这种方法，我们可以由一个简单的距离函数产生非常复杂的区域边界而并不需要额外的存储负担。类似于实时绘制领域广泛使用的几何实例化（geometry instancing），我们可以进一步推广2-7式，使得一个基元距离函数的每个实例可以拥有自己独立的线性变换。具体来说，对于基元距离函数 $\dot{\mathcal{D}}$，我们定义如下复合距离函数:

$$\mathcal{D}(x)=\mathcal{F}(\dot{\mathcal{D}}(x\cdot\mathbf{M}_1)+\dot{\mathcal{D}}(x\cdot\mathbf{M}_2)+\cdots+\dot{\mathcal{D}}(x\cdot\mathbf{M}_k)) \tag{2-8}$$

图2.10展示了 $\mathcal{F}$ 为最小值运算（对应布尔运算的并）时2-8式定义的复合符号距离函数。

值得注意的是，通过这些方法定义的复合距离函数并不一定仍然是欧几里得度量空间中严格意义的距离函数，即空间中任意一点的距离值不一定等于该点到最近零值面上点的欧式距离。但由于我们关注的是零值面及其附近的距离值，这种不严格的距离函数并不会影响我们定义区域边界形状的用途，因此我们仍然不加区分地称其为距离函数。另外，尽管复合距离函数由多个不同的基元距离函数、或是一个基元距离函数的多个不同实例共同定义，但复合距离函数本身在逻辑上仍然是一个距离函数。

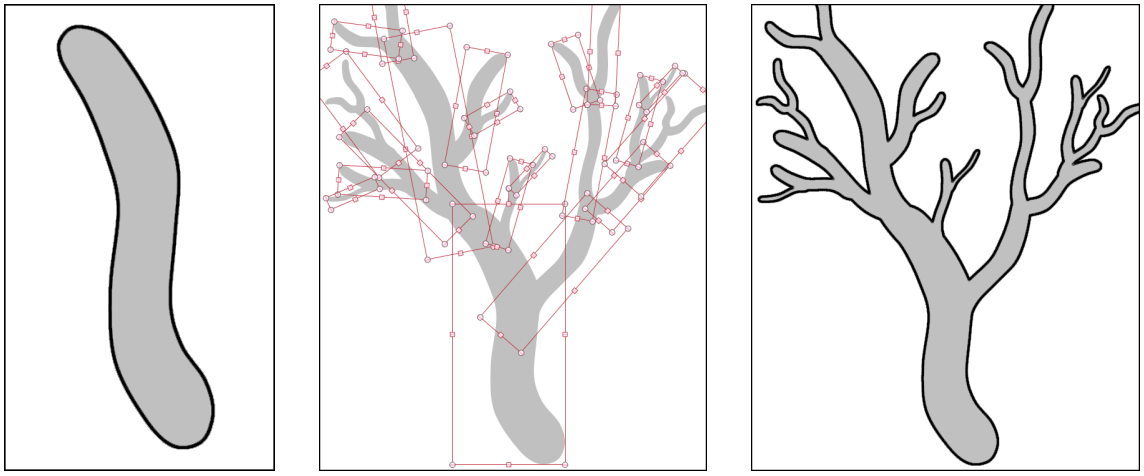
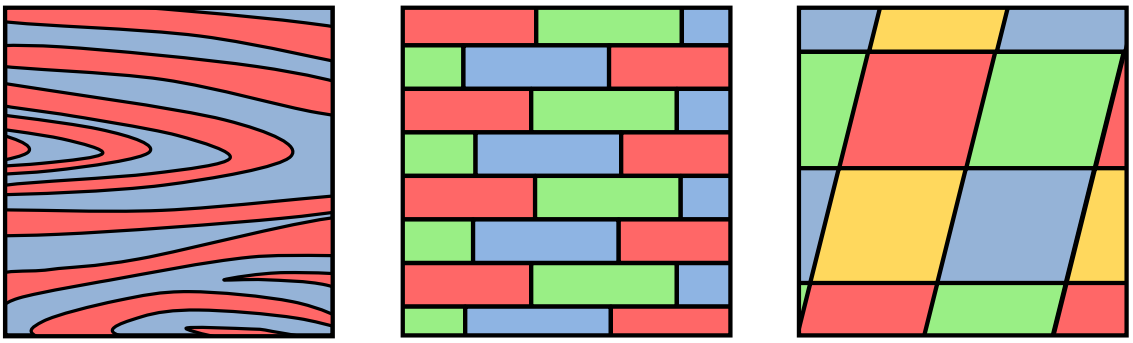


图 2.10　左图中的基元距离函数通过实例化生成右图中的复合符号距离函数。中图红色框为每个实例的定向包围盒。

## 2.4　区域划分的定义

一个符号距离函数 $\mathcal{D}$ 事实上已经将空间划分为两个区域，即距离值为正的区域与距离值为负的区域，且这两个区域之间的边界是符号距离函数的零值面。真实物体内部通常包含大量的区域，这些区域之间可能有复杂的邻接关系。如果我们将物体内部的区域划分 $\mathcal{P}$ 看作一个图 $G(\mathcal{P})$，$G$ 中每个顶点 $v_i \in G$ 对应于 $\mathcal{P}$ 中的一个区域 $\mathcal{P}_i$，边 $(v_i, v_j)$ 表示区域 $\mathcal{P}_i$ 与 $\mathcal{P}_j$ 相邻，则我们可以用图 $G$ 的着色数（chromatic number）将区域划分 $\mathcal{P}$ 分为**可二着色的**与**不可二着色的**两类。需要注意的是，通常在地图（二维区域划分）的着色问题中两个区域相邻当且仅当它们共享边，而本文这里定义的相邻关系包括两个区域共享面、边和点这三种情况。图2.11例举了一些不同着色数的区域划分。

在可二着色的区域划分中，根据图着色的定义，在边界表面上任意一点总可以找到一个足够小的邻域使得其中只包含两个区域，因此所有的边界表面事实上可以用一个逻辑上的符号距离函数隐式定义。我们针对这种情况提出一种单距离函数的区域划分方法（2.4.1节）。在着色数大于二的区域划分中，多个相邻区域之间的边界可能形成 T 形或更复杂的交叉。为了描述这种非流形的边界特征，我们提出一种基于多个符号距离函数的 SDF 树数据结构（2.4.2节）。最后，为了描述真实物体中跨越多个尺度层次的区域划分以及利用物体中重复性的特征，我们定义体模型的嵌套与实例化（2.4.3节）。



$\chi(G)=2$　　$\chi(G)=3$　　$\chi(G)=4$

图 2.11　不同着色数的区域划分。为了表示方便这里以二维情况举例，在三维体模型中区域之间的邻接关系更加复杂。

### 2.4.1　单距离函数区域划分

符号距离函数 $\mathcal{D}$ 的零值面将空间分割为多个连通部分（connected components），其中每个连通部分内部所有的点都具有相同的距离符号，并且对应于物体内部的一个区域。为了区分具有相同符号的不同连通部分，我们对空间中任意一点 $x$ 定义一个**区域标记**，记为 $\ell(x)$。与距离函数类似，区域标记也定义在和符号距离函数相同的离散网格 $\mathcal{G}$ 上。为判断任意点 $x$ 所属的区域标记 $\ell(x)$，我们首先计算 $\mathcal{D}(x)$ 的符号。若区域的连通性是根据每个网格结点的 26 个相邻结点[①] 确定的，则易知网格中每个单元格中的 8 个节点上至多出现两种不同的区域标记，且对于单元格中的任意两个节点 $x_a$，$x_b$，必然有：

$$\mathcal{D}(x_a)\cdot\mathcal{D}(x_b)\le 0 \iff \ell(x_a)\neq\ell(x_b). \tag{2-9}$$

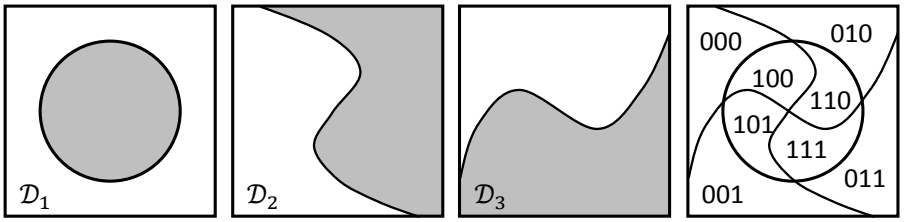
由此，根据 $\mathcal{D}(x)$ 的符号以及 $x$ 点所在 $\mathcal{G}$ 中单元格的 8 个节点的区域标记，我们可以唯一地确定 $x$ 点的所属区域 $\ell(x)$。

如前所述，单距离函数区域划分要求 $\mathcal{P}$ 是可二着色的。这是一个相对严格的限制：尽管我们观察到真实世界中的一大类体模型的区域划分都是可二着色的，不少更复杂的、高度结构化的物体却违反这一限制。因此，我们将在下一节中提出一种更通用的基于多个距离函数的区域划分方法。

### 2.4.2　多距离函数区域划分（SDF 树）

在不可二着色的区域划分中，超过两个区域可能彼此邻接，使得它们之间的边界曲面在空间中形成含有非流形交叉结构的复杂曲面网络，因此不可能用单一

① 对应于立方体 6 个面、12 条边、8 个顶点位置的相邻网格结点。




| $\mathcal{D}_1$ | $\mathcal{D}_2$ | $\mathcal{D}_3$ | $s$ | 区域 |
|---|---|---|---|---|
| − | − | − | 000 | [■] |
| − | − | + | 001 | [■] |
| − | + | − | 010 | [■] |
| − | + | + | 011 | [■] |
| + | − | − | 100 | [■] |
| + | − | + | 101 | [■] |
| + | + | − | 110 | [■] |
| + | + | + | 111 | [■] |

图 2.12　三个重叠的符号距离函数（$\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$）将空间划分为 8 个初始区域，通过符号标记到实际区域的映射（上表）最终定义了 5 个区域（以不同颜色区分）。

符号距离函数加区域标记的方法描述。但注意到曲面网络中非流形的部分是由多个局部的流形部分组成的，其中每个部分都可以由单独的符号距离函数隐式描述，并且空间中任意点的所属区域可以由所有距离函数的符号共同决定。

假设空间中有 $m$ 个重叠的符号距离函数，亦即在空间中任意一点 $x$ 均定义有 $m$ 个符号距离值 $(\mathcal{D}_1(x), \mathcal{D}_2(x), \ldots, \mathcal{D}_m(x))$，如图2.12所示，这些符号距离函数的零值面共同构成一个初始的空间划分 $\bar{\mathcal{P}}$。对于空间中的任意一点 $x$，考虑 $m$ 个距离函数在该点的符号，我们可以定义一个二进制整数 $s(x)$，其第 $i$ 位的值代表第 $i$ 个距离函数的符号，亦即：

$$s(x) = a_m a_{m-1} \ldots a_1, \quad \text{其中} a_i = \begin{cases} 1, & \mathcal{D}_i(x) \geq 0 \\ 0, & \mathcal{D}_i(x) < 0 \end{cases} \tag{2-10}$$

称 $s(x)$ 为 $x$ 点的**符号标记**。根据其定义，初始的空间划分 $\bar{\mathcal{P}}$ 中每一个区域内部所有的点都具有相同的符号标记，并且任意两个相邻区域内的点均具有不同的符号标记。当然，并不是 $\bar{\mathcal{P}}$ 中的每一个区域都对应于实际体模型中的一个区域，因此 $\bar{\mathcal{P}}$ 可以看作是一个 “过分割”（over-segmentation），一个实际区域划分 $\mathcal{P}$ 中的区域可能由多个初始区域划分 $\bar{\mathcal{P}}$ 中的区域共同构成。这相当于在符号标记与实际的区域标记之间建立一个满射 $f : s \to \ell$。

### SDF 树数据结构

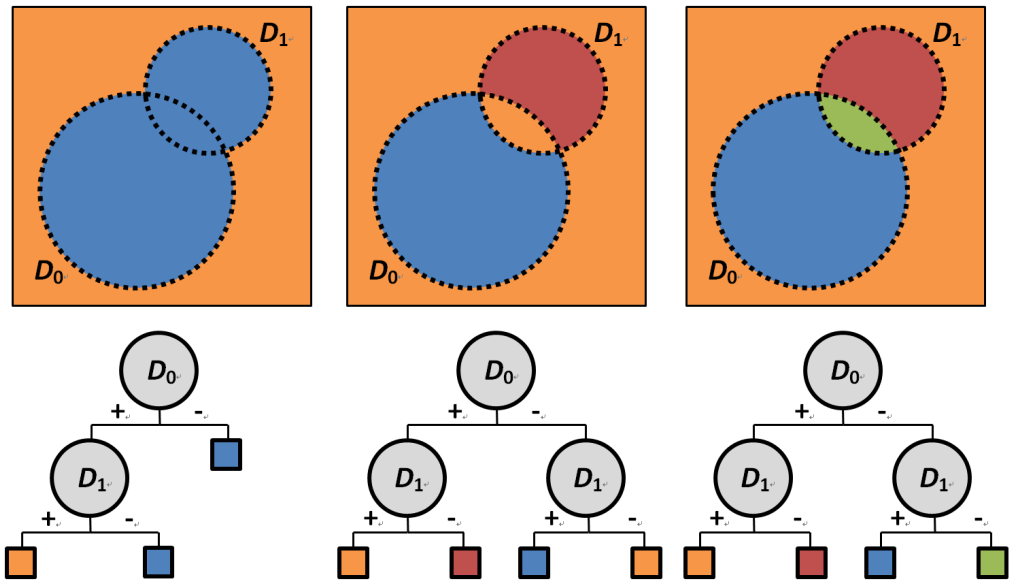
在一个由 $m$ 个距离函数 $\{\mathcal{D}_i\}_{i=1}^m$ 以及映射 $f: s \to \ell$ 定义的体模型区域划分中，为了判断 $x$ 点的所属区域，一种最直接的办法是根据 $x$ 点处 $m$ 个距离函数的符号计算 $s(x)$，再由 $f$ 得到最终的区域标记。由于初始区域划分 $\bar{\mathcal{P}}$ 和实际区域划分 $\mathcal{P}$ 中的区域数都是有限的，$f$ 在实现中可以存储为查找表的形式。然而为了获得符号标记 $s(x)$ 需要计算 $m$ 个距离函数的值，这在一个由大量距离函数（几十至上百个）定义的复杂区域划分中计算代价将是很高的。特别是，随着距离函数数量的增加，很大一部分距离函数的符号很可能并不影响 $x$ 的区域归属，对每一个感兴趣的点均计算一次所有距离函数的值无疑是一种浪费。

为此，我们提出一种称为 SDF **树**的树状数据结构。SDF 树中的每个非叶节点对应一个符号距离函数，因而也称 SDF **节点**；每个叶节点则对应一个区域标记，因此也称**区域节点**。由于每个符号距离函数的零值面都将空间划分为两个半空间（half space），相应地每个 SDF 节点也有两个子节点，我们人为定义左子节点对应距离值为正的半空间，右子节点对应距离值为负的半空间，因此可知 SDF 树是一个二叉树。若一个 SDF 节点 $v_a$ 的子节点 $v_b$ 也是一个 SDF 节点，则意味着 $v_a$ 对应的距离函数中相应的半空间被 $v_b$ 对应的距离函数进一步划分为两个半空间。

这里有几点需要注意：首先，同一个符号距离函数可以出现在不同的 SDF 节点，前提是这些节点不处在同一条从根节点至叶节点的路径上。其次，多个区域节点可以对应相同的区域标记。根据这样的定义，同样的 $m$ 个符号距离函数经过不同的组合，可以产生完全不同的区域划分结果，如图2.13所示。另外，区域节点对应的区域标记可以为“空”，这表示该区域内部的点不属于体模型内部。通过这种方式，SDF 树不仅可以表示物体的内部区域结构，也可以统一地表示物体的外部边界表面。

对于一个由 SDF 树定义的区域划分，判断空间中任意一点 $x$ 所属区域的过程即是对 SDF 树的遍历：由根节点开始，对于每一个 SDF 节点 $v$，根据该节点对应的距离函数在 $x$ 点的符号正负决定下一个遍历的是 $v$ 的两个子节点中的哪一个，这一过程依此进行直至遍历到达一个区域节点，该节点对应的区域标记即指出 $x$ 点的所属区域。相比于直接计算符号标记，通过 SDF 树遍历决定区域归属的时间复杂度由 $O(m)$ 降低为 $O(\log m)$。

值得注意的是，SDF 树与符号标记这两种区域划分的表示方法是等价的：遍历 SDF 树的路径即表明了路径上的 SDF 节点所对应的距离函数的符号，但 SDF 树使得只有对 $x$ 点区域归属有影响的距离函数的值需要被计算，从而有效降低了计算开销。某种意义上 SDF 树可以看作是图形学中广泛应用的 BSP（binary space




(a)　(b)　(c)

图 2.13　SDF 树结构。通过将同样两个符号距离函数 $\mathcal{D}_1$ 和 $\mathcal{D}_2$ 组织成不同结构的 SDF 树可以得到不同的区域划分结果。

partitioning）树[81] 的一种推广：BSP 树使用平面进行空间剖分，而 SDF 树允许使用任意复杂形状的隐式曲面。

### 与 CSG 树的比较

尽管 SDF 树与基于隐式表面的 CSG 树[83] 均是由多个隐式表面构成的树状结构，但两者有着本质区别。在一个 CSG 树中，所有的隐式表面都处于叶节点，而每个非叶节点对应于某种布尔运算。这些隐式表面以**从底向上**的方式依次经布尔运算组合最终生成位于根节点的**一个实体几何**。而在一个 SDF 树中，每一个非叶节点均对应一个隐式表面（符号距离函数），这些隐式表面以**自顶向下**的方式递归地将位于根节点的体空间剖分为**多个更小的区域**。

CSG 树定义的实体几何的边界表面是一个二维流形，通过多个隐式表面的运算，这个二维流形可以含有导数不连续的几何特征，如折面、角点等；而 SDF 树定义的区域边界可以是包含大量非流形结构的复杂曲面网络，这些曲面将空间划分为多个区域，其中每个区域都可以看作是一个具有任意复杂边界形状（包括导数不连续的几何特征）的实体几何。更进一步的分析指出：**对由一个 SDF 树定义的区域划分中任意一个区域 $\mathcal{P}_i$，都可以用该 SDF 树中出现过的符号距离函数构造一个 CSG 树，使得由这个 CSG 数定义的实体几何形状恰为 $\mathcal{P}_i$**。

#### 与单距离函数区域划分的比较

使用多个距离函数除了可以灵活地表示不可二着色的区域划分中复杂的非流形区域边界外，一个额外的好处是不再需要存储单距离函数区域划分中使用的区域标记结构。由于区域标记是存储在三维规则网格上的，因此需要占据相当的存储空间（尽管4.1.3节中将介绍一种有效的压缩方法），并且网格的分辨率会限制物体中区域的最小尺度。事实上，在多个符号距离函数构成的 SDF 树结构中，区域标记已经被隐式地表现在不同的树路径中，因此不再需要显式地记录这一信息，这不仅降低了存储开销，同时进一步提高了多距离函数区域划分的表达能力。

另外，多距离函数区域划分中的每一个符号距离函数都是**逻辑上**的符号距离函数。尽管单距离函数区域划分中的距离函数可以是由多个基元距离函数共同定义的复合距离函数，但在逻辑上仅有一个距离函数参与了区域划分。

### 2.4.3　嵌套与实例化

真实物体的区域划分通常是多尺度层次的。对一个体模型来说，在一个尺度上进行区域划分得到的每个区域都可能是一个包含更多更小尺度结构特征的独立体模型。为了描述这样的多尺度结构特征，一种方法是利用 SDF 树的特性进行递归地区域划分。这里我们介绍另一种方法，可以将一个已有的体模型嵌入另一个体模型中的某个区域。

体模型嵌套最大的好处是提高了已有体模型的可重用性，同时也使得基于区域结构的体建模方式更加灵活：用户既可以采用自顶向下的方式，首先定义体模型内部大尺度的区域结构，再逐级向下地将每个区域根据需要细化为更多小尺度的区域，最后定义每个区域内部的材质属性；也可以采用自底向上的方式，首先创建每个尺度层次的代表性体模型，再通过嵌套与实例化等方式将多个体模型组织起来构成一个完整的复杂体模型。

#### 体模型的嵌套

不失一般性，我们将每个体模型都定义在 $[0,1]^3$ 的单位立方区域，即物体的局部坐标系内，并通过线性变换 $\mathbf{M}$ 定义体模型在三维空间，即世界坐标系中的位置、尺寸和朝向。对于两个体模型 $A$ 和 $B$，我们可以定义由 $A$ 至 $B$ 的嵌入关系，即 $A$ 内部的某个区域 $i$ 中包含 $B$，且用线性变换 $\mathbf{M}_{A,B}$ 描述 $B$ 在 $A$ 的局部坐标系中的位置、尺寸与朝向。在判断空间中任意一点 $x$ 的所属区域 $\ell(x)$ 时，我们首先根据 $A$ 的区域划分判断，若 $x$ 在 $A$ 中的所属区域 $\ell_A(x) = i$，则进一步判

断 $x$ 在 $B$ 中的所属区域 $\ell_B(x \cdot \mathbf{M}_{A,B})$，注意这里 $x$ 需要被变换到 $B$ 的局部坐标系中。若 $x$ 不属于 $B$ 中的任何区域（$\ell_B(x \cdot \mathbf{M}_{A,B})$ 为空），则 $\ell(x) = \ell_A(x)$，否则，有 $\ell(x) = \ell_B(x \cdot \mathbf{M}_{A,B})$。

物体的同一个区域中可以嵌入多个体模型，而若这些体模型之间有重叠，则上述判断区域归属的方法会产生歧义。对于这种情况，我们参考二维矢量图或图像处理软件中经常使用的 $z$ 顺序（$z$-order）及图层概念，人为地定义嵌入同一区域中的多个体模型的先后顺序。

体模型的嵌套可以是多层的，需要注意的是每次嵌套时局部坐标都需要进行相应的变换。理论上，嵌套可以是任意多层的，甚至一个体模型可以嵌套自身，形成无限的递归嵌套。但在实际中，嵌套深度的增加会导致判断区域归属的时间开销变大，因而实际能够达到的嵌套深度受系统性能及模型复杂程度的限制。

### 体模型的实例化

真实世界中的许多物体中都包含大量相似的元素，例如组成生物体的众多细胞。与之相对应，一个体模型内部也可能含有大量相似的、微小的子模型。单独地为每一个子模型建模所需要花费的时间和精力通常远远超过实际中可以接受的程度，更不用说单独存储每一个子模型需要的存储开销可能呈指数增长。在实时绘制领域，几何实例化[94]（geometry instancing）是一种广泛使用的绘制具有大量重复元素场景的方法，通过使所有重复的实例共享通用数据降低存储开销，提高绘制效率。例如对于一个包含大量人物模型的虚拟场景，尽管不同的人物模型可以有不同的位置、速度、动作，但它们可以在内存中共享同一份网格模型及贴图数据。Wei 等人[95] 和 Wang 等人[96] 则通过分析二维纹理或一般图像中具有重复性的元素达到图像压缩等目的。

在基于区域结构的体模型中也可以很容易地实现体模型的嵌套。对嵌入到体模型 $A$ 一个区域中的体模型 $B$，我们可以定义多个线性变换 $\{\mathbf{M}_{A,B,i}\}_{i=1}^{n}$，每个线性变换与 $B$ 一起构成 $B$ 的一个实例。在逻辑上，$B$ 的 $n$ 个实例相当于 $n$ 个独立的体模型，因此可以按照一般的多个体模型嵌套的情况判断空间中点的所属区域，但是由于 $B$ 在自身局部坐标系中的定义被所有实例共享，因而可以极大降低内存开销，并且用户仅需要为每个实例指定一线性变换而不是分别建模。设想一个表示人体组织局部的体模型中嵌套有 1000 个表示细胞的体模型，每个细胞体模型内部又嵌套有 100 个表示核糖体的体模型，利用实例化，我们仅需要建模和存储三个独立的体模型以及 1100 个线性变换矩阵，而不是 $10^5$ 个独立的体模型。

## 2.5　区域内部的定义

本节中我们讨论基于区域的体模型中材质属性函数 $C$ 的定义。在区域结构体模型中，每个区域都是相对独立的，具有明确的边界。这种区域划分所带来的一个最直接的好处就是每个区域内部可以独立地定义材质属性而不用担心影响周围的区域。由于体模型的材质属性函数 $C$ 中的不连续介面已由区域边界表示，每个区域内部的材质属性特征主要可以分为低频的平缓变化或是高频的随机噪声两种。

针对低频的材质属性变化，我们采用径向基函数（radial basis functions，RBFs）拟合的方式表示。Zhou 等人[97] 在模拟烟雾效果中曾使用径向基函数表示烟雾中的低频部分，取得了很好的效果。在 [97] 中用于表示烟雾的体数据是一个标量的密度场，我们将其扩展到用径向基函数拟合表示任意材质属性的三维向量场。具体来说，我们将材质属性函数 $C$ 定义如下：

$$C(x) = \sum_{q=1}^{m} w_q B_q(x) \tag{2-11}$$

其中基函数 $B_q(x)$ 的值取决于 $x$ 点到 $B_q$ 的中心 $c_q$ 以及半径 $r_q$：

$$B_q(x) = \phi\left(\frac{\|x - c_q\|}{r_q}\right) \tag{2-12}$$

使用径向基函数可以表达材质属性复杂的连续变化，并且存储开销通常远远小于离散采样体数据的形式。我们将在3.3.2节中更细致地介绍使用径向基函数表达区域内部材质属性的方法，并给出一种利用径向基函数拟合一个给定离散采样体数据的算法。

针对材质属性中的高频噪声部分，我们采用 Perlin 噪声[47] 表示。类似其它过程噪声的工作（如 [4,47]），我们将 Perlin 噪声的标量值通过一个一维函数映射到材质属性向量（例如颜色）。

目前我们仅实现了径向基函数与 Perlin 噪声这两种区域内部材质属性的定义方式，事实上，材质属性的定义可以扩展到任意形式的三维函数，例如以解析形式描述的颜色渐变等。此外，由于区域结构体模型中各区域的独立性，一个体模型可以同时使用多种不同的方式定义不同区域内部的材质属性。

## 2.6　本章小结

在这一章我们给出了基于区域结构体数据表达的定义。在设计这种数据表达时，我们特别考虑了表达能力、建模方式、存储空间、绘制效率这四方面因素。

区域结构在真实物体中广泛存在，这些区域具有任意复杂的边界形状，不同的区域或边界特征可能具有相差悬殊的尺度，区域之间可能构成多层嵌套关系。我们提出的 SDF 树结构通过多个符号距离函数进行空间划分，可以灵活表达以上这些不同的区域结构特征。区域结构的定义使用户可以从语义的角度把握体模型内部的材质属性变化，因而使得直接创建或编辑高度复杂的体模型成为可能。通过复合符号距离函数，我们可以用较小的存储开销表达复杂的边界几何形状。此外，符号距离函数的特性使得区域结构体数据表达支持高效的随机访问。以区域划分为基础，我们可以分别定义每个区域内部的材质属性，以分辨率无关方式表达的区域边界使得体模型在绘制时材质属性不连续的跳变能准确呈现，不会出现走样或模糊的问题。

# 第 3 章　区域结构体模型的创建与编辑

本章将讨论如何创建或编辑基于区域结构的体模型。创建一个区域结构体模型主要可以通过三种方式：直接创建，基于样本创建，以及由其它表达形式的体模型转化得到。对于传统的基于离散采样的体模型，直接创建通常是困难的。由于没有对结构信息的描述，用户需要直接关注 “空间中某点的颜色是什么” 这样的底层问题，类似于用笔在白纸上作画。然而由于体数据的信息量庞大，且其三维本质决定了人无法在同一时刻没有歧义地看到一个体数据中的全部信息，这种手工指定空间中颜色的方法是非常低效和不准确的。Takayama 等人[43] 曾提及一种 “体素编辑器”，利用该程序他们成功地创建了一些规模非常小（分辨率 $32^3$ 左右）的简单体模型作为进一步算法的输入样本，但创建更大、更复杂的完整体模型所需的时间和精力使得这种方法变得不再实际。

基于区域结构的体数据表达使用户可以从一种新的角度考虑体模型的创建过程：真实世界中的任何具有复杂内部构造的物体在一定尺度上都可以分解为不同均匀 “材质” 构成的区域，这里的材质既可以是物理意义的，例如墙体内的钢筋和混凝土，也可以是逻辑意义的，例如皮肤中的表皮、真皮和皮下组织。另外，这种区域结构的定义可以是递归的：例如，一个体模型中表示 “皮下组织” 的区域可以进一步划分为 “微动脉”、“微静脉”、“神经”、“脂肪细胞” 等区域，而 “脂肪细胞” 又可以划分为 “细胞膜”、“细胞核” 等，以此类推，复杂的体模型可以表示为多尺度层级的区域划分。这种层次结构可以使用户更加直观、清晰地把握复杂体模型的内部结构。

从设计方法学的角度，传统的体建模通常是线性的过程，而基于区域结构的体数据表达允许用户以迭代式的过程[98,99] 进行建模。具体来说，创建一个区域结构体模型的典型过程可以分为**规划**和**实施**两个阶段。在规划阶段，用户根据准备创建的对象不同考虑如下问题：

- 体模型中需要有几个尺度层级？
- 每个层级中有哪些区域？
- 每个区域的内部属性如何？

在实施阶段，用户可以迭代地进行以下三个步骤：创建定义区域边界所需的符号距离函数，定义区域划分，以及设定各区域的内部属性。其中每一个步骤都可以有多种实现方式，用户可以根据具体需求灵活选择（例如手工直接创建或是

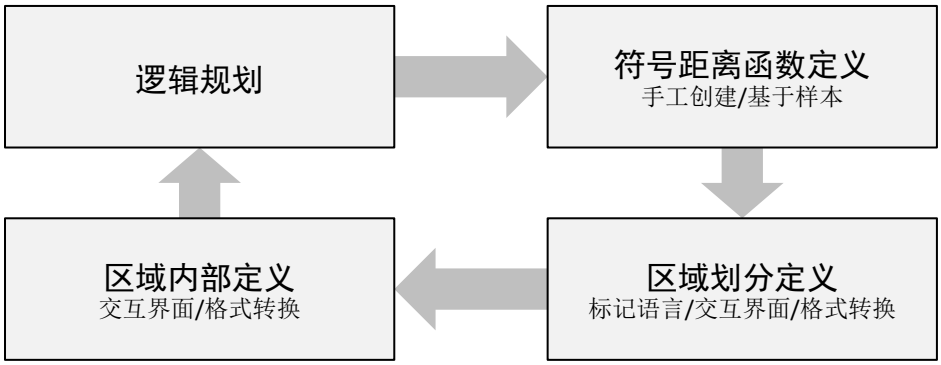



图 3.1　基于区域结构的体建模流程。

基于输入样本)。这三个步骤并没有严格的先后顺序，例如，用户可以在任何时候创建新的距离函数并将一个现有的区域划分为更多子区域，或是在定义好区域划分后再修改某个区域的边界形状。由于区域结构的存在，对一个区域的改动不会影响到体模型的其它部分。整个基于区域结构的体建模流程如图3.1所示。

## 3.1　创建与编辑距离函数

区域结构体模型中的区域边界可以是任意形状的曲面，这些曲面由一个或多个符号距离函数隐式定义。本节我们将讨论创建这些符号距离函数的不同方法。

显式曲面（explicit surfaces）与隐式曲面（implicit surfaces），以及两者间的联系与相互转化在数学中的分析几何、图形学中的几何建模等领域均有长期深入研究[100]。显式表示的曲面（如参数曲面、多边形网格）因其易于控制、渲染、存储等优点被当今绝大多数主流建模软件支持[101–104]，这些软件提供了大量功能强大且为广大专业人员熟练掌握的交互式建模工具。而隐式曲面虽然在常见的几何建模中应用不如显式曲面广泛，但却具有一些独特的优点，例如高效的布尔操作、拓扑结构变化等，这些因素使得隐式曲面建模能更好地支持基于笔画的建模方式[105] 以及表现类似生物组织（organic）的几何形状。

在我们的建模系统中实现了四种方法创建或编辑符号距离函数。首先，为了利用现有的显式曲面建模工具，3.1.1节介绍一种将显式表示的多边形网格转化为符号距离函数的方法。另外三种建模方法均利用了隐式曲面表示的独特优势，3.1.2节介绍一种基于笔画的交互式界面，3.1.3节和3.1.4节分别介绍两种根据输入样本生成符号距离函数的算法。

### 3.1.1　通过多边形网格

将多边形网格转化为隐式表示即是计算以网格表面为零值面的符号距离函数，这通常要求该多边形网格是一个封闭的流形，从而可以明确定义表面的内部与外部。这个问题一种最直接的解决办法是：对空间中每一点，计算从该点到多边形网格的最短距离，并根据该点属于曲面内部或外部决定距离的符号。这种方法虽然简单直观，但效率低下，使用 CPU 计算一个高分辨率的符号距离函数可能需要花费数小时[106]。为此人们提出了多种加速算法，如 prism scan[107]、特征扫描转换[108]、基于图形处理器的四面体扫描转换[106] 等。

这一问题可以形式地描述如下：令 $\partial S$ 为三维实体 $S$ 表面的多边形网格，且 $\partial S$ 是二维流形，$\mathring{S}$ 表示 $S$ 内部点的集合。$\mathcal{G}$ 为定义在三维空间，且与坐标轴对齐的均匀网格，含有 $d_W \times d_H \times d_D$ 个节点。求定义在 $\mathcal{G}$ 上的符号距离函数 $\mathcal{D}$，使得对任意 $x \in \mathcal{G}$:

$$\mathcal{D}(x) = \begin{cases} -\inf_{y \notin S} \|x - y\|, & x \in \mathring{S} \\ 0, & x \in \partial S \\ \inf_{y \in S} \|x - y\|, & x \in \mathbb{R}^3 - S \end{cases} \tag{3-1}$$

我们使用的算法基于 Jones 提出的计算点与多边形网格最短距离的算法[109]，但利用了图形处理器加速距离符号的判断过程。算法分为两个步骤：

#### 确定符号

在这一步中我们需要判断网格 $\mathcal{G}$ 的每个节点位于多边形网格的内部抑或外部。在 [107,108] 等方法中通过查找距离空间中一点 $x$ 最近的多边形并根据该多边形的法向量决定 $x$ 在其哪一侧，进而判断 $x$ 点处的距离符号，这种做法在有的情况下会得到错误的符号[106]。我们方法的基本思想是将三维网格 $\mathcal{G}$ 看作沿空间中 $z$ 轴紧密排列的 $d_D$ 个分辨率为 $d_W \times d_H$ 的二维 “切片”。对每一个切片 $\mathcal{G}_z$，我们将多边形网格正交投影到 $\mathcal{G}_z$ 所在平面，并利用图形处理器将其扫描转换为一个对应的 $d_W \times d_H$ 分辨率的图像。通过将 $\mathcal{G}_z$ 设定为剪裁平面，在扫描转换多边形网格时仅有在 $\mathcal{G}_z$ 一侧的部分被绘制，且我们在像素着色器（pixel shader）中令所有背向投影方向的多边形扫描转换后的值为 $-1$，其它多边形及背景扫描转换的值为 $+1$。在所有切片上进行这样的扫描转换后即得到三维网格上每个节点的对应符号。

直观上，这一方法相当于从 $\mathcal{G}$ 的每个节点沿 $z$ 轴方向发射一条射线并判断与多边形网格的交点数的奇偶，因而对于封闭的多边形网格具有很高准确性。通过

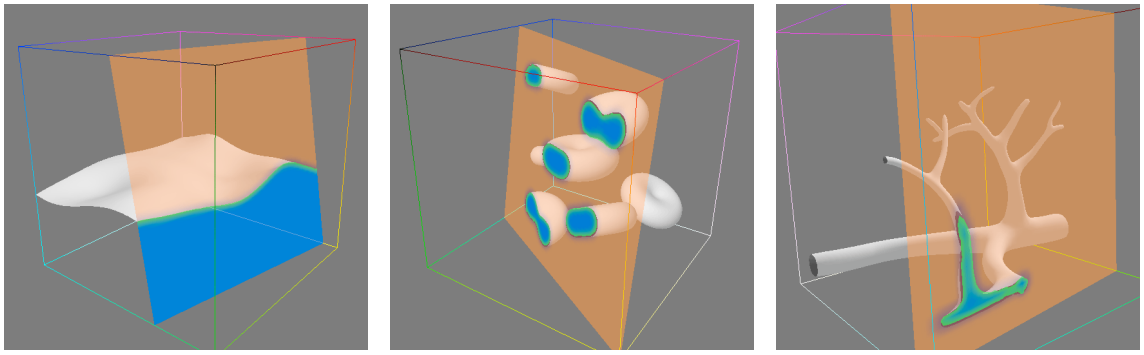


图 3.2　根据多边形网格生成的带符号的距离函数。

图形处理器上多边形扫描转换的实现，我们可以并行判断多个节点，因而大大减小了判断距离符号的时间开销。在我们的实验中，将顶点数约 $10^5$ 的多边形网格转化为分辨率 $256^3$ 的符号距离函数通常可在不到一秒的时间内完成。

#### 计算最短距离

在这一步中我们计算 $\mathcal{G}$ 中每个节点到多边形网格的**无符号**最短距离。这里我们采用了 Jones 提出的算法[109]。值得注意的是，由于距离函数在区域结构体模型中仅用来表达区域边界曲面，在实际中我们并不需要计算 $\mathcal{G}$ 中全部节点的最短距离而仅需考虑网格附近的距离值。这可以极大减少计算所需的时间。图3.2展示了从多边形网格生成距离函数的过程。

为了让用户的建模过程更加连贯，我们将计算符号距离函数的功能模块集成进开源建模软件 Blender 中，用户可以直接在 Blender 中将创建、编辑好的多边形网格导出为符号距离函数。

### 3.1.2　通过交互式工具

和显式表面相比，隐式表面的一个好处是允许用户通过直接修改距离函数的值定义表面形状，这尤其适合以基于笔画的方式创建自由形状[105,110]。在我们的建模系统中也实现了一组基于笔画的交互式工具，允许用户直接创建距离函数或是修改一个现有的距离函数。这些工具在设计时着重考虑了距离函数的隐式表达特点，这里简要介绍其中几种主要工具的功能与实现思路。

#### 曲面画刷

曲面画刷允许用户直接在空间中画出一个区域。画刷的形状由用户指定的一个符号距离函数 $\mathcal{D}_B$ 定义，这个距离函数沿着用户画出的笔迹与原始的距离函数

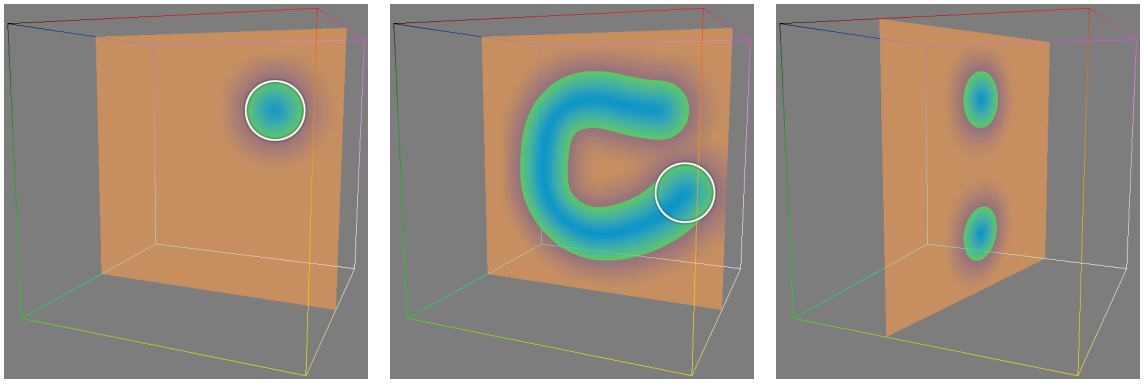


图 3.3　通过曲面画刷直接创建的距离函数及其定义的区域边界曲面。右图显示了中图距离函数的另外一个截面。

$\mathcal{D}$ 按 2.3 节中定义的方式叠加，生成新的距离函数 $\mathcal{D}'$：

$$\mathcal{D}'(x) = \min(\mathcal{D}(x),\ \mathcal{D}_B(x \cdot \mathbf{M}_B)) \tag{3-2}$$

其中 $\mathbf{M}_B$ 定义画刷距离函数的线性变换，由用户指定的画刷大小、方向以及位置决定。

在实际操作中，用户首先选择空间中一个任意位置、朝向的的二维“工作平面”，随后画刷的移动则局限在该平面上。图3.3示意了使用曲面画刷创建区域边界表面的过程。

### 曲面变形画刷

曲面变形画刷允许用户修改现有距离函数中零值面的局部形状。它在现有距离函数中用户指定的位置叠加一个随时间变化的高斯函数：

$$\mathcal{D}'(x) = \mathcal{D}(x) + a_B t e^{-\frac{(x-b_B)^2}{2c_B^2}} \tag{3-3}$$

其中 $a_B$、$c_B$ 是用户可以调节的参数，决定变形的强度和范围，$b_B$ 是画刷的位置，$t$ 是用户应用画刷的时长。曲面变形画刷的效果如图3.4所示。

### 随机噪声工具

Perlin 和 Hoffert[53] 提出在一个现有符号距离函数中叠加一个过程噪声函数来增强隐式曲面的表面细节与真实感。与之类似，我们允许用户向一个现有的距离函数加入简单的噪声来为过于平滑的区域表面增加细微变化。用户可以通过参数控制噪声的强度、频率以及各向异性，如图3.5所示。

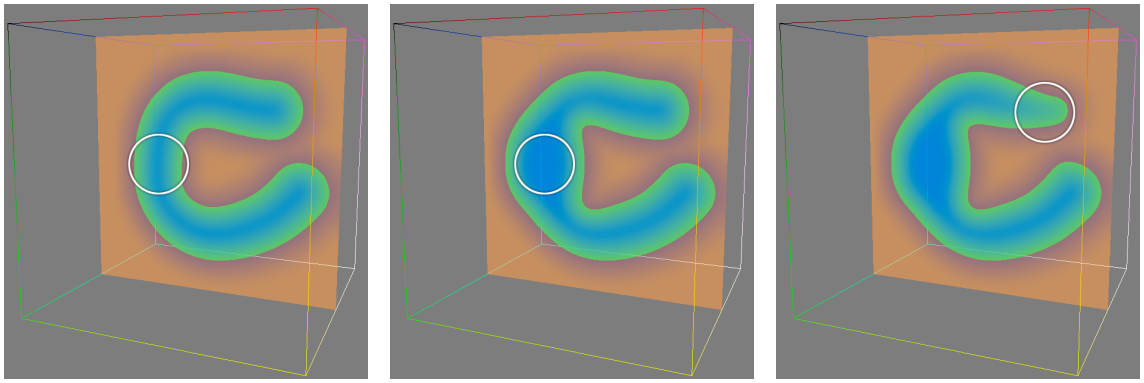


图 3.4　通过曲面变形画刷修改现有距离函数的过零等值面形状。

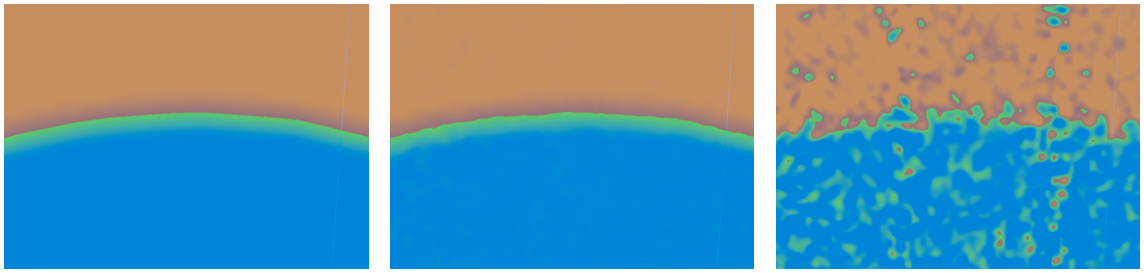


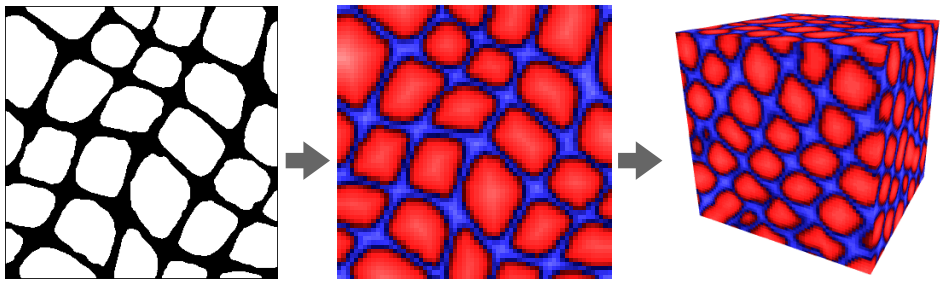
图 3.5　通过引入随机噪声向隐式曲面增加表面细节。

### 3.1.3　通过基于样本的纹理合成

通过上文介绍的交互式工具或通过多边形网格转换的方式创建符号距离函数赋予用户最大的自由度与灵活性，但对于真实世界中常见的一些随机、重复性的结构，诸如碎石、木纹等，这类方法却往往需要花费较大时间和精力。由于这类结构的真实照片通常容易得到，并且其结构特征往往很好地符合马尔可夫随机场对纹理局部性和静态性的假设，这意味着我们可以使用基于样本纹理合成的技术来自动生成。实际上，符号距离函数早已应用在纹理合成中，在一些二维纹理合成[26,111] 及体纹理合成[36,38] 的研究中发现，在输入纹理颜色的基础上额外增加一个 “特征图” 通道（feature map）一起进行合成能够明显提高一部分具有较强结构特征纹理的合成质量。这里的特征图实际就是对应纹理中明显区域结构的符号距离函数。

利用纹理合成生成符号距离函数的过程可以描述为：对于一个给定的二维符号距离函数 $\mathcal{D}_{2D}$，生成一个三维符号距离函数 $\mathcal{D}_{3D}$，使其零值面从统计角度看具有与输入 $\mathcal{D}_{2D}$ 类似的几何特征。与 [26,111] 相似，为了生成二维符号距离函数，用户需要提供一个二值图 $\mathcal{M}_{2D}$ 标示出二维的区域结构，我们的算法首先提取二值图中区域边界像素的位置并建立 kd 树结构的空间索引[82]，再在一个二维规则网格的每个节点上查找到区域边界的最近距离并根据二值图的值决定距离符号，从

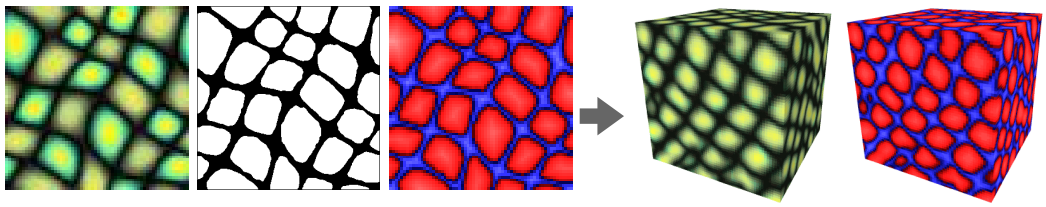
而生成二维符号距离函数 $\mathcal{D}_{2D}$。为了避免出现走样，二值图的分辨率通常是二维符号距离函数的 4 至 8 倍。接下来，我们使用 Kopf 等人提出的基于体纹理优化的方法[36] 生成三维符号距离函数 $\mathcal{D}_{3D}$，这一过程如图3.6所示。



$\mathcal{M}_{2D}$　　$\mathcal{D}_{2D}$　　$\mathcal{D}_{3D}$

图 3.6　利用纹理合成生成三维符号距离函数。

除了单独合成符号距离函数，我们也可以用同样的方法生成一个同时包含颜色信息与符号距离函数的体纹理，如图3.7所示。这个体纹理中离散采样的颜色信息可以被进一步转化为径向基函数的简洁表示，3.3.2节将深入讨论这一问题。



$\mathcal{C}_{2D}$　$\mathcal{M}_{2D}$　$\mathcal{D}_{2D}$　$\mathcal{C}_{3D}$　$\mathcal{D}_{3D}$

图 3.7　根据二维样本生成三维距离函数。根据输入的二维颜色图像 $\mathcal{C}_{2D}$、二值区域标记 $\mathcal{M}_{2D}$ 和二维距离函数 $\mathcal{D}_{2D}$，我们使用体纹理合成的方法生成一个三维体颜色纹理 $\mathcal{C}_{3D}$ 以及三维距离函数 $\mathcal{D}_{3D}$。

需要注意的是，通过纹理合成得到的符号距离函数并非真正意义上的距离函数，因为其中任意点的绝对值并不一定准确等于从该点到零值面的最短距离。但由于距离函数在我们的区域结构体模型中仅仅作为区域边界曲面的隐式表示，这样的距离函数已经足够满足我们的需求了。

### 3.1.4　通过图像类比

近年来对体纹理合成的研究工作[36–38,40] 产生了大量高质量的体纹理素材，尽管使用这些方法可以从一个二维图像纹理生成含有符号距离函数的体纹理，有时用户可能会希望直接将一个现有的**不含有符号距离信息的**体纹理转化为基于区域结构的体模型，这就需要根据纹理的颜色信息定义符号距离函数。


图 3.8　图像类比。(图片来自 [112])

对于二维纹理来说，一种简单的做法是手工创建一个二值图作为区域标记并根据该二值图计算符号距离函数。但是对于体纹理，由于体数据性质造成的困难，手工创建一个三维的二值图在多数情况下是不现实的。因此我们提出一种算法能自动根据用户提供的二维二值图生成一个三维二值图。

我们的算法受 Hertzmann 等人提出的图像类比[112]（Image Analogies）启发。图像类比所解决的核心问题如图3.8所示，对于给定的三幅图像 $A$、$A'$ 和 $B$，其中 $A$ 与 $A'$ 具有一定的**类比关系**，生成一个新的图像 $B'$，使得 $B'$ 相对于 $B$ 具有和 $A'$ 相对于 $A$ 类似的类比关系。Hertzmann 等人将这种类比记为：

$$A : A' :: B : B' \tag{3-4}$$

在我们的应用场景中，我们注意到体纹理中的颜色信息和符号距离函数并不是独立的而是具有很强的相关性：颜色的不连续跳变通常对应于符号距离函数中的零值面。并且若体纹理整体中存在这种相关性，由纹理的马尔可夫场假设可以推断，它的任意一个二维截面应该也存在这种相关性。令 $C_{3D}$ 和 $\mathcal{D}_{3D}$ 分别表示一个体纹理的颜色信息与符号距离函数，$C_{2D}$ 和 $\mathcal{D}_{2D}$ 则分别表示体纹理的一个截面或是生成体纹理的原始二维纹理样本的颜色信息与符号距离函数，借用 Hertzmann 等人的标记方法，我们可以将这一类比关系表述为：

$$C_{2D} : \mathcal{D}_{2D} :: C_{3D} : \mathcal{D}_{3D} \tag{3-5}$$

这里的体纹理 $C_{3D}$ 是已知的，$C_{2D}$ 可以是用于生成 $C_{3D}$ 的原始样本，或是从 $C_{3D}$ 中抽取的任意截面。用户仅需要根据 $C_{2D}$ 提供一个二维的二值图 $\mathcal{M}_{2D}$，使用3.1.3节中的方法可以由二值图 $\mathcal{M}_{2D}$ 生成二维符号距离函数 $\mathcal{D}_{2D}$，接下来，我们可以根据以上类比关系自动计算三维符号距离函数 $\mathcal{D}_{3D}$。

计算 $\mathcal{D}_{3D}$ 的过程是基于最近邻匹配的：对于 $\mathcal{D}_{3D}$ 中每一点 $x_{3D}$，在 $C_{3D}$ 中同一位置可以定义三个相互正交的二维邻域 $E_x, E_y, E_z$，通过最近邻查找，我们可以在二维纹理 $C_{2D}$ 中找到与这三个邻域最匹配的像素位置 $x_{2D,x}$，$x_{2D,y}$，$x_{2D,z}$，此时，

我们将 $\mathcal{D}_{3D}$ 在 $x_{3D}$ 点的位置定义为:

$$\mathcal{D}_{3D}(x_{3D}) = \frac{1}{3}(\mathcal{D}_{2D}(x_{2D,x}) + \mathcal{D}_{2D}(x_{2D,y}) + \mathcal{D}_{2D}(x_{2D,z})) \tag{3-6}$$

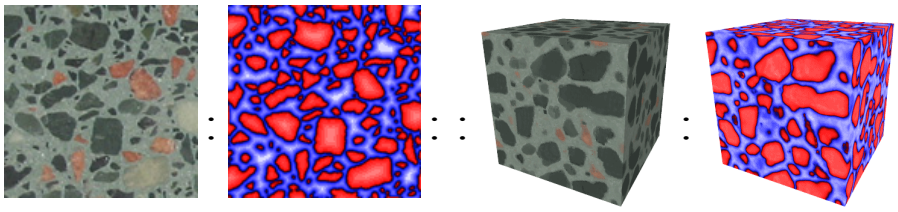



图 3.9　使用图像类比（image analogies）[112] 中的标记方法描述从现有体颜色纹理中提取三维距离函数的问题。

## 3.2　创建与编辑区域划分

符号距离函数中的零值面将体模型划分为多个初始的连通区域，但是仅凭符号距离函数不足以判定空间中任意点的区域归属。对于使用单个符号距离函数的区域划分，可以直接根据距离函数中正负区域的连通性自动为每个连通区域分配一个区域标记（见2.4.1节）。而对于由多个距离函数定义的区域划分，需要将这些距离函数组织成一个 SDF 树结构，对于更复杂的具有多尺度结构的体模型，还需要定义模型的嵌套与实例化。

我们的建模系统中共有三种方法可以创建 SDF 树及定义嵌套与实例化关系：3.2.1节介绍一种为描述体模型中的区域结构设计的基于 XML 的标记语言；3.2.2节介绍一套以拖拽（drag-and-drop）为主要操作方式的交互界面用于构造复杂的 SDF 树；最后，3.2.3节将介绍如何将医学扫描得到的体数据中的离散分区信息转化成 SDF 树的表示。

### 3.2.1　通过体对象标记语言（VOML）

在基于区域结构的体模型中，多尺度的区域结构由 SDF 树描述。在一个复杂的体模型中，SDF 树可能有几十甚至上百个节点。为了使用户能够直观且明确地定义 SDF 树以及区域结构体模型的其它信息，我们设计了一种基于 XML[113] 的**体对象标记语言**（Volumetric Object Markup Language，以下简称 VOML 语言）。

作为一种基于 XML 的语言，一个 VOML 文档中最主要的组成元素是**标签**（tags）。不同类型标签内部可以定义不同的**属性**（attributes），并且标签可以相互嵌套，因此很自然地，我们可以将 SDF 树的层次结构表示为标签的层次结构。

SDF 树是一个二叉树，其中每个非叶节点对应一个符号距离函数，并且有两个子节点，分别对应零值面划分出的两个半空间，因此我们在 VOML 中通过<SDF>标签定义一个 SDF 节点：

```
<SDF name="D1" file="1.sdf">
  <POSITIVE region="Region1"/>
  <NEGATIVE region="Region2"/>
</SDF>
```

在以上 VOML 片段中一个符号距离函数D1 将体模型划分为Region1 和Region2 两个区域。这些区域的内部属性可以由<REGION> 标签单独定义，例如：

```
<REGION name="Region1" rgb="1 0 0"/>
<REGION name="Region2" rgb="0 0 1"/>
```

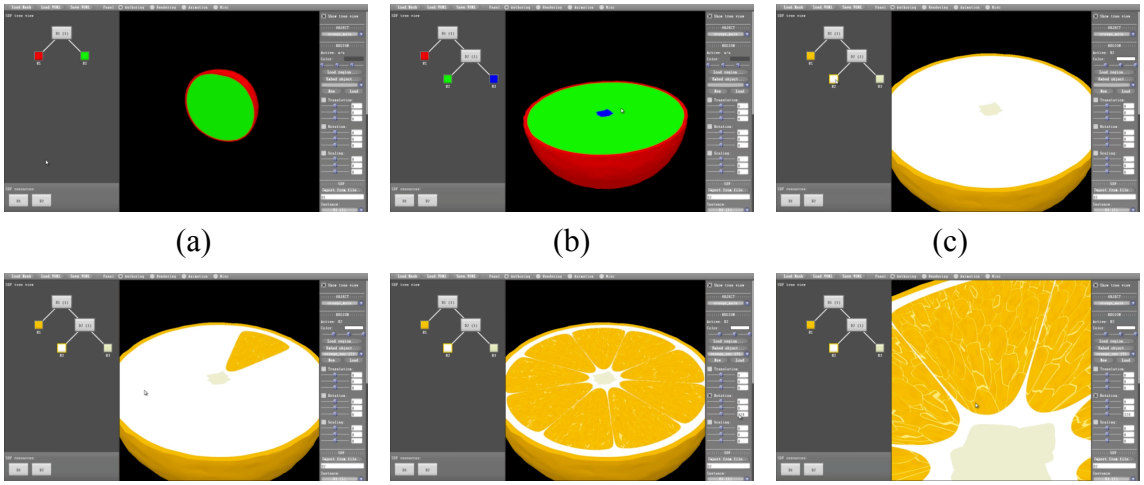
多层的 SDF 树可以通过标签的嵌套很容易地构造，例如以下是一段稍微复杂的 VOML 片段：

```
<OBJECT name="skin">
  <SDF name="D0" file="0.sdf">
    <POSITIVE region="hypodermis">
    <NEGATIVE>
      <SDF name="D1" file="1.sdf">
        <POSITIVE>
          <SDF name="D2" file="2.sdf">
            <POSITIVE region="dermis"/>
            <NEGATIVE region="epidermis"/>
          </SDF>
        </POSITIVE>
        <NEGATIVE region="epidermis"/>
      </SDF>
    </NEGATIVE>
  </SDF>
</OBJECT>
```

VOML 为区域结构体模型中的 SDF 树提供了一种直观易读的形式化描述。通过定义新的标签，VOML 可以很容易地扩展对体模型中更多特性的描述。与 X3D[114] 和 SVG[68] 等标记语言类似，VOML 还可以用于描述一个包含多个体模型的复杂三维场景。附录A中将包含更详尽的 VOML 语法规则及示例。

### 3.2.2　通过交互式工具

对于简单的区域结构体模型，用户可以直接通过 VOML 语言定义区域划分。为了方便用户创建结构更复杂的体模型，我们实现了一套交互式的图形用户界面。在这个原型系统中，用户可以首先导入先前创建的符号距离函数，并定义体



(a)　(b)　(c)

(d)　(e)　(f)

图 3.10　交互式建模界面：(a)-(b) 用户通过拖拽的方式直观地将 SDF 节点组织成 SDF 树；(c) 用户可以选中并编辑每个区域的内部属性；(d)-(f) 向区域中嵌入其它体对象并编辑每个实例的仿射变换。

模型中的区域，这些符号距离函数和区域定义在图形界面中表示为节点，用户可以通过拖拽的方式将这些节点自由组织成 SDF 树结构，或是定义多个体模型之间的嵌套关系。在用户进行操作时对应的体模型会即时更新。图3.10展示了我们的交互式建模界面。

### 生成体模型的实例

为了生成体模型实例化中的每个实例，用户可以选择两类方法。第一类是手工指定，用户可以直接通过 VOML 语言的<EMBEDDED> 标签及相应属性定义被嵌入体模型实例的线性变换，例如：

```
<REGION name="Region1" rgb="1 0 0">
  <EMBEDDED object="SubObj" scale="0.5 0.5 0.5"
                            rotate="0 0 45"/>
</REGION>
```

用户也可以在 Blender 中复制一个 “代理”（proxy）体模型，并调整代理模型每个副本的位置、尺寸、朝向等。代理模型是一个与原始体模型相似的多边形网格模型。我们对 Blender 的扩展可以将每一个副本的线性变换导出以便在 VOML 文档中引用。这类方法适合于少数需要精确控制变换的实例。

第二类方法是通过过程的方法自动或半自动地生成。图形学中对于大量聚集物体的生成有广泛研究，选择哪种方法取决于具体的应用场景和用户希望取得的效果。我们的交互式系统中实现了以下几种生成实例的方法：对于在空间中均

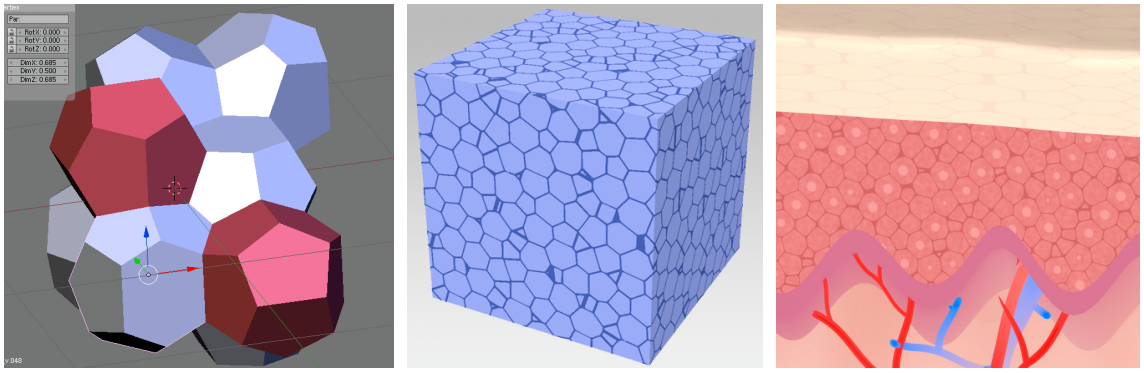


图 3.11　Weaire-Phelan 结构及由其得到的细胞排列结果。从左至右依次是：构成 Weaire-Phelan 一个平铺单元的 8 个多面体，平铺后经过旋转和缩放变换的区域划分，在实际体模型中的应用。

匀分布且彼此之间相距足够远以致不会发生相交的实例，我们采用泊松圆盘采样[115] 的方法产生实例的中心位置。这种方法的好处是可以在图形处理器上实现，迅速地生成大量实例，且其密度分布可以由用户自由控制。对于紧密排列的实例，则必须考虑实例的形状以确保紧密相邻的实例之间不会出现不真实的相交。为此我们采用物理模拟的方法生成[116]。对于在一个曲面上均匀分布的实例（例如细胞膜上的蛋白分子），我们首先在曲面上随机选取采样点，再通过 relaxation[117] 使其分布均匀（一种更高效的方法是直接在曲面上进行均匀采样[118]）。实例线性变换中的旋转项（即实例的朝向）可根据用户给定的方向场、曲面的法向、或随机决定。这类方法适合产生数量庞大（通常在几千至上万）的实例。

除了上面介绍的这些方法，对一些特殊的实例分布，有时采用专门的方法生成更为有效。比如我们利用 Weaire-Phelan 结构[119] 来生成用于表现紧密排列细胞的实例分布①。通过这种方法，我们仅需定义 8 个实例即可产生能无限平铺的细胞排列效果（如图3.11所示）。

### 3.2.3　通过离散区域分割

在3.1.3节和3.1.4节中曾介绍如何从现有的体纹理中提取符号距离函数。但这些方法要求体纹理中的区域划分是可二着色的，也因此只能提取单个距离函数。这里介绍一种简单的方法可以处理具有不可二着色区域划分的离散采样数据，并自动生成由多个符号距离函数构成的 SDF 树。

给定一个包含分区信息的离散采样体数据，在三维网格 $\mathcal{G}$ 上同时定义有材质

① Weaire-Phelan 结构是一种蜂巢结构。几何学中的蜂巢问题研究如何用一系列的多面体紧密无缝、互不重叠地填满整个空间。Weaire-Phelan 结构使用了两种不同形状但相同体积的多面体，其最小重复单元包含 2 个十二面体和 6 个十四面体。

属性函数 $C$ 和区域标记 $\ell$，且 $\ell$ 是不可二着色的，亦即对于网格 $\mathcal{G}$ 上任意一个单元格的 8 个顶点中可能有超过 2 种不同的区域标记。由于此时的区域边界表面是含有非流形特征的曲面网络，无法用单一的符号距离函数表示，因此为了准确表达这种区域结构，我们需要从输入体数据中提取多个符号距离函数，并将它们组织成 SDF 树。在这一节中我们暂时忽略材质属性 $C$ 而仅靠虑区域边界以及 SDF 树的创建，在下一节中（3.3.2节）将讨论材质属性的转化。

我们算法的基本流程如下：

1. 根据输入离散体数据中属于每个区域的体素数对所有区域按降序排序，记排序后的区域为 $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_\kappa\}$。
2. 创建 $\kappa - 1$ 个标记体数据 $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{\kappa-1}\}$，且这些标记满足下式：

$$\mathcal{M}_i(x) = \begin{cases} +1, & \ell(x) = i \\ 0, & \ell(x) < i \\ -1, & \ell(x) > i \end{cases} \tag{3-7}$$

3. 我们将 $\mathcal{M}_i$ 中标记为 0 的网格顶点看作是未定义的，并用 fast marching 算法[120] 将其它顶点上有定义的标记 “扩散” 至所有顶点，最终得到 $\kappa - 1$ 个二值体数据，如图3.12所示。使用3.1中的方法可以进一步由这些二值体数据分别计算得到 $\kappa - 1$ 个符号距离函数 $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{\kappa-1}\}$。
4. 由3-7式可知，每个符号距离函数 $\mathcal{D}_i$ 均将前一个符号距离函数 $\mathcal{D}_{i-1}$ 符号为负的半空间继续一分为二，因此我们可以简单地构造一个 SDF 树结构，其中距离函数 $\mathcal{D}_i$ 对应的 SDF 节点是 $\mathcal{D}_{i-1}$ 对应的 SDF 节点的右子节点，从而构成一个不可二着色的区域划分。

## 3.3　创建与编辑区域属性

由于区域结构体模型中各区域的独立性，我们可以分别定义每个区域内部的材质属性而不用担心影响周围区域。在目前的实现中，我们使用两种形式定义材质属性：径向基函数与过程噪声。其中前者适合描述材质属性低频的平缓变化，后者适合描述高频的随机变化，两者均具有存储开销低、分辨率无关的优点。在我们的建模系统中，用户可以通过交互式工具直接定义这两种形式的区域属性(3.3.1节)，也可以通过一种自动算法将离散采样表示的材质属性数据转化为径向基函数表示。

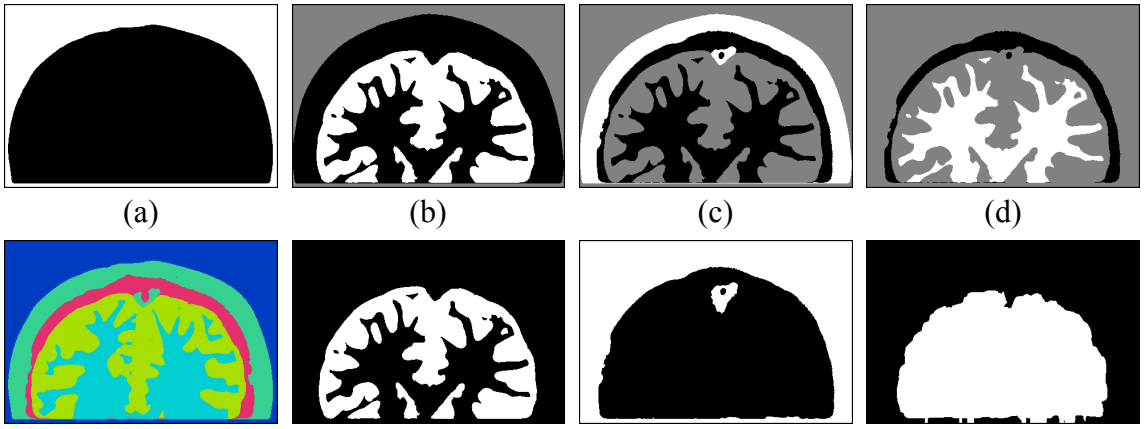



图 3.12　根据输入的离散区域标记信息 (e)，我们首先根据3-7式创建 $\kappa - 1$ 个标记体数据 (a-d)，再使用 fast marching 计算 (b-d) 中未定义的区域（灰色），得到完整的二值标记 (f-h) 并进一步将其转化为符号距离函数。

### 3.3.1　通过交互式工具

在我们的交互式建模系统中，用户首先通过点击体模型或 SDF 树节点选择一个区域，再进一步设定该区域的材质属性。

#### 径向基函数画刷

与3.1.2节介绍的几种符号距离函数建模工具类似，我们也实现了一种基于笔画的径向基函数画刷。在用户笔画的轨迹上会生成一系列径向基函数，这些基函数共同定义材质属性函数 $C$:

$$C(x) = \sum_{q=1}^{m} w_q B_q(x) \tag{3-8}$$

径向基函数画刷的半径和材质属性为用户可以调整的参数。

#### 各向异性的过程噪声

通过建模系统的图形用户界面，用户可以直观地设定过程噪声函数 $\mathcal{N}$ 的强度、频率、各向异性，并创建一个一维函数 $\varphi : \mathbb{R} \to \mathbb{R}^p$ 将噪声函数的标量值映射为材质属性向量:

$$C(x) = \varphi(\mathcal{N}(x)) \tag{3-9}$$

### 3.3.2　径向基函数的非线性拟合

在3.1节和3.2节中我们讨论了如何从一个离散采样表示的体模型中提取符号距离函数以及定义区域划分，本节中我们将讨论如何根据输入中以离散采样表示的数据 $C$ 决定区域结构体模型中每个区域内部的材质属性。这里有两种最直接的办法：一是直接计算 $C$ 中属于每个区域全部体素的平均颜色作为该区域的颜色；二是直接为每个区域单独保存 $C$ 中属于该区域的体素。由于我们已假设 $C$ 中的高频特征可以由区域边界刻画，每个区域内部的颜色变化相对平缓，因而第一种方法在很多情况下可以作为很好的近似，且仅占用几乎可以忽略的存储空间。第二种方法保留原始输入 $C$ 中的所有细微颜色变化，但需要占用大量存储空间，特别是对于原本仅剩低频颜色变化的区域会造成无意义的空间浪费。更麻烦的是，为了防止相邻区域之间的颜色相互影响（color bleeding），我们需要将属于每一个区域的体素单独保存，这样一个存在大量不规则形状区域的体数据所需要的存储空间甚至可能大大超过原始的离散采样体模型本身。第2.5节曾介绍用径向基函数来表示区域内部材质属性变化具有表达能力强、节约存储空间等优点，这里很自然地希望可以将离散采样的材质属性转化为径向基函数的表示方式。

为便于描述，首先不考虑多个区域的情况，假设整个体数据都属于同一个区域，这个问题可以重新表述如下：不失一般性，假设 $\mathcal{G}$ 为一定义在 $[0,1]^3$ 由 $d_W \times d_H \times d_D$ 个节点构成的三维网格；$x_{ijk}$ 表示网格节点 $(i,j,k)$ 在三维空间中的位置，且有

$$x_{ijk} = (\frac{i-1}{d_W-1}, \frac{j-1}{d_H-1}, \frac{k-1}{d_D-1}),\ \ i \in \{1,\ldots,d_W\}, j \in \{1,\ldots,d_H\}, k \in \{1,\ldots,d_D\} \tag{3-10}$$

$C$ 为定义在 $\mathcal{G}$ 上的材质属性函数，用 $C(x)$ 表示空间点 $x$ 所在位置的材质属性值（通常为一向量，包含颜色等信息）；再令 $\tilde{C}$ 表示以径向基函数拟合 $C$ 得到的近似材质属性函数，则现在可将径向基函数拟合表述为如下优化问题：

$$\min \sum_{x_{ijk} \in \mathcal{G}} \|\tilde{C}(x_{ijk}) - C(x_{ijk})\|^2. \tag{3-11}$$

#### 径向基函数的定义

$\mathbb{R}^n$ 空间中的径向基函数 $B$ 指任意一点 $x$ 的函数值 $B(x)$ 仅取决于 $x$ 到空间中某点距离的一类函数，表示如下：

$$B(x) = \phi\left(\frac{\|x-c\|}{r}\right), \tag{3-12}$$

其中 $c$ 称为径向基函数 $B$ 的 “中心”，$r$ 称作 $B$ 的 “半径”。多个径向基函数的加权和被广泛用于函数拟合、离散数据插值等领域[97,121,122]。其基本思想是以 $m$ 个径向基函数的加权和 $\tilde{C}$ 近似某已知函数 $C$:

$$C(x) \approx \tilde{C}(x) = \sum_{q=1}^{m} w_q B_q(x). \tag{3-13}$$

其中 $w_q$ 称作基函数 $B_q$ 的 “权重”。

在将径向基函数用于材质属性拟合时有几点需要注意：第一，每个基函数的权重是一个高维向量而非标量；第二，考虑到作为拟合对象的材质属性多数情况下变化平缓，我们在拟合前预先计算每个区域的平均值，记为 $\bar{C}$，实际用径向基函数拟合平均值与原始值的**残差**，即:

$$\tilde{C}(x) = \bar{C} + \sum_{q=1}^{m} w_q B_q(x). \tag{3-14}$$

因为平均值 $\bar{C}$ 可以看作是原始材质属性变化一个较好的零阶近似，这样可以极大提高对于变化平缓区域的拟合效率。第三，对于径向基函数 $B$ 中 $\phi$ 的选择通常有很多种，如高斯函数、多重调和基样条、薄板样条等，这些函数都具有无限的支撑集，这即是说空间中任意一点 $x$ 处拟合函数 $\tilde{C}$ 的值都受到所有径向基函数的影响，这有悖于我们在设计基于区域结构的体模型时追求高效随机访问性能的目标，尤其是当径向基函数的数目很大、平均半径却较小时，在绘制体纹理的过程中为了决定屏幕上每个像素的颜色需要考虑区域中全部的径向基函数，即使大部分基函数的中心与该像素所对应的空间位置相距很远且影响很小。因此，为了提高随机访问效率，我们选择 Wyvill 函数[88] 作为径向基函数的实现:

$$\phi(r) = \begin{cases} 1 - \frac{4}{9}r^6 + \frac{17}{9}r^4 - \frac{22}{9}r^2, & r \le 1 \\ 0, & r > 1 \end{cases} \tag{3-15}$$

Wyvill 函数的最大优点在于支撑集是有限的，此外求值的计算量相对较小。

### 拟合算法

公式 3-11是一个典型的非线性优化问题，变量为 $c_q, r_q, w_q, (1 \le q \le m)$。这里采用迭代法求解：首先在体数据所在空间中随机选取 $m$ 个点作为径向基函数中心 $c_q$ 的初始值，可以额外应用 relaxation 方法[117] 以使位置的空间分布更加均匀（另一种等价的做法是使用泊松圆盘采样算法[115] 直接生成均匀分布的初始位置)。基函数半径 $r_q$ 的初始值默认均为 $m^{-1/3}$，权重 $w_q$ 的初始值设为 $C(c_q) - \bar{C}$。随后，我

们使用 L-BFGS-B 算法[123] 迭代求解。L-BFGS-B 是一种基于梯度的算法，并且允许限定变量的取值范围。为此我们将变量限定在如下范围：

$$\begin{aligned} 0 &< c_q < 1, \\ 0.5m^{-1/3} &\le r_q \le 2m^{-1/3}, \\ w_{\min} - \frac{w_{\max} - w_{\min}}{2} &\le w_q \le w_{\max} + \frac{w_{\max} - w_{\min}}{2}, \end{aligned}$$

其中

$$\begin{aligned} w_{\min} &= \min_{x_{ijk} \in \mathcal{G}} [C(x_{ijk} - \bar{C}(x_{ijk}] \\ w_{\max} &= \max_{x_{ijk} \in \mathcal{G}} [C(x_{ijk} - \bar{C}(x_{ijk}] \end{aligned} \tag{3-16}$$

对于径向基函数数量 $m$ 的选取有两种策略。第一种是根据区域的大小以及颜色变化的丰富程度手工确定；另一种是指定一个较小的初始数量及一个拟合错误阈值，每次拟合结束后考察最大拟合错误：

$$\epsilon_{max} = \max_{x_{ijk} \in \mathcal{G}} \|\tilde{C}(x_{ijk}) - C(x_{ijk})\|^2. \tag{3-17}$$

若 $\epsilon_{max}$ 超过指定阈值，则加入新的基函数后继续进行拟合直到 $\epsilon_{max}$ 小于阈值。对于本文中出现的结构化体数据，单个区域中包含的径向基函数通常小于 5000 个。

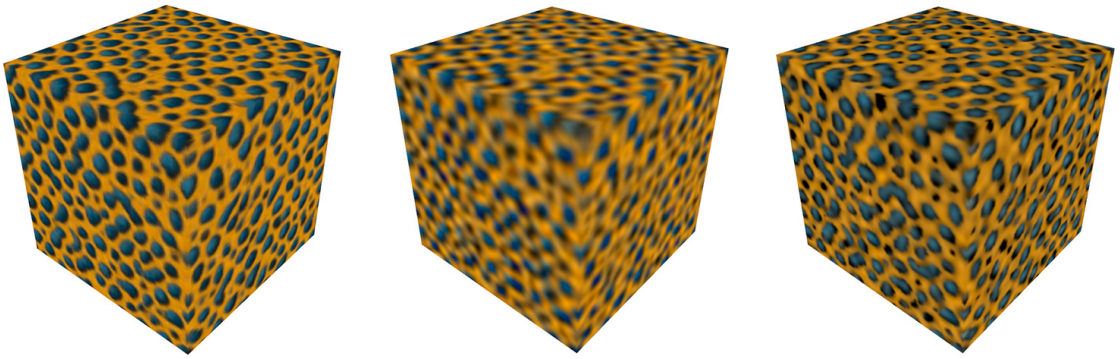
由于变量数量较大（使用 5000 个径向基函数对 RGB 颜色进行拟合共有 $3.5 \times 10^4$ 个变量），在优化3-11式时很容易陷入局部极小值。为此我们采用 teleportation 策略[97,124] 改善这一状况。具体做法是在每次拟合收敛后，尝试将“影响最小” 的径向基函数移动到拟合错误最大的位置继续拟合并检查能否减小总体拟合错误，若拟合错误无法进一步减小则 teleportation 结束。一个径向基函数 $B$ 的 “影响” 定义为移除 $B$ 导致目标函数（3-11式）值的变化，亦即：

$$\sum_{x_{ijk} \in \mathcal{G}} \|\tilde{C}(x_{ijk}) - C(x_{ijk}) - wB(x_{ijk})\|^2 - \sum_{x_{ijk} \in \mathcal{G}} \|\tilde{C}(x_{ijk}) - C(x_{ijk})\|^2. \tag{3-18}$$

在我们的实验中，应用 teleportation 策略通常可以将总体拟合错误进一步降低约 40%。

### 多区域的情况

以上讨论针对整个体数据中仅有一个区域的情况。区域结构体数据表示的一个重要优点就是通过区域划分，每个区域内部定义完全独立的材质属性，相邻区域之间的边界可以准确表现传统方法难以表现的材质属性的不连续跳变，并且对



(a) 离散体数据　　(b) 单区域拟合 $\epsilon = 3.34 \times 10^{-2}$　　(c) 多区域拟合 $\epsilon = 2.21 \times 10^{-2}$

图 3.13　(a) 原始输入；(b) 单区域 RBF 拟合；(c) 多区域 RBF 拟合。

一个区域的编辑不会影响的周围的区域。同样的，在对离散采样数据进行径向基函数拟合时，我们也必须考虑区域结构，以免原本应属于不同区域（材质）的属性相互干扰，造成颜色泄漏（color bleeding）的问题，如图3.13所示。为此，在拟合时，我们为体模型中的每个区域分配一组独立的径向基函数，并且对输入的离散采样数据中属于每个区域的体素分别进行拟合。

假设离散采样体数据中共有 $\kappa$ 个区域，为所有区域分配的径向基函数总数为 $n$，则优化目标3-11式可以被重写为：

$$\min \sum_{p=1}^{\kappa} \sum_{x_{ijk} \in \mathcal{G}_p} \|\tilde{C}(x_{ijk}) - C(x_{ijk})\|^2, \tag{3-19}$$

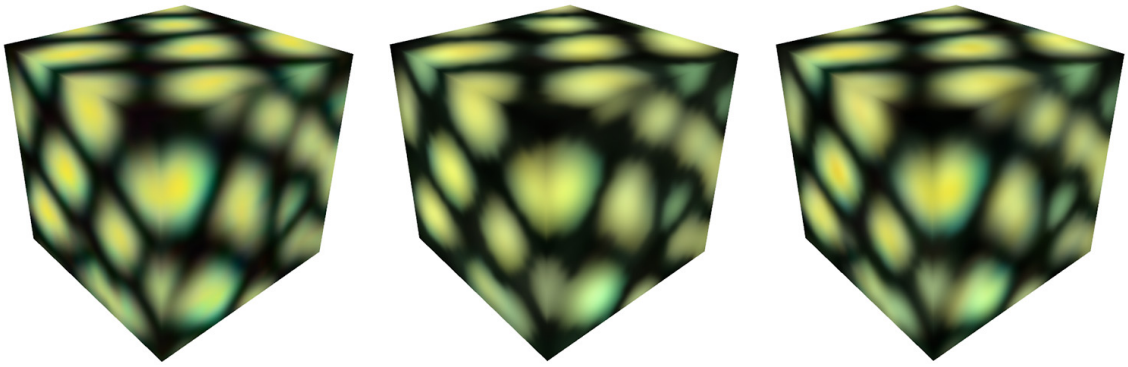
相应地3-14式改写为：

$$\tilde{C}(x) = \bar{C}_p + \sum_{q=1}^{m_p} w_{pq} B_{pq}(x), \quad p = \{p \,|\, x \in \mathcal{G}_p\}, \tag{3-20}$$

其中 $\mathcal{G}_p$ 表示属于区域 $p$ 的网格节点集合；$m_p$ 表示区域 $p$ 中径向基函数的数目且有 $\sum_p m_p = n$；$B_{pq}$ 及 $w_{pq}$ 分别表示区域 $p$ 的第 $q$ 个径向基函数及其相应权重。

在拟合时我们首先选取 $n$ 个点作为径向基函数中心的初始位置并将各基函数分配给其中心所属的区域。当 teleportation 过程中一个基函数被移动到新的位置时，若该位置属于不同的区域则该基函数的所属区域也需要即时更新。

对于任意一个输入的离散采样体数据，拟合所需的径向基函数总数由输入中材质属性的变化复杂程度与用户对拟合错误的容忍度决定。图3.14列举了使用不同数目径向基函数进行拟合一个离散采样体数据的结果。



(a) 离散体数据　(b) 400 RBF $\epsilon = 6.61 \times 10^{-3}$　(c) 1000 RBF $\epsilon = 3.13 \times 10^{-3}$

图 3.14　不同数量径向基函数的拟合结果。

## 3.4　建模结果与讨论

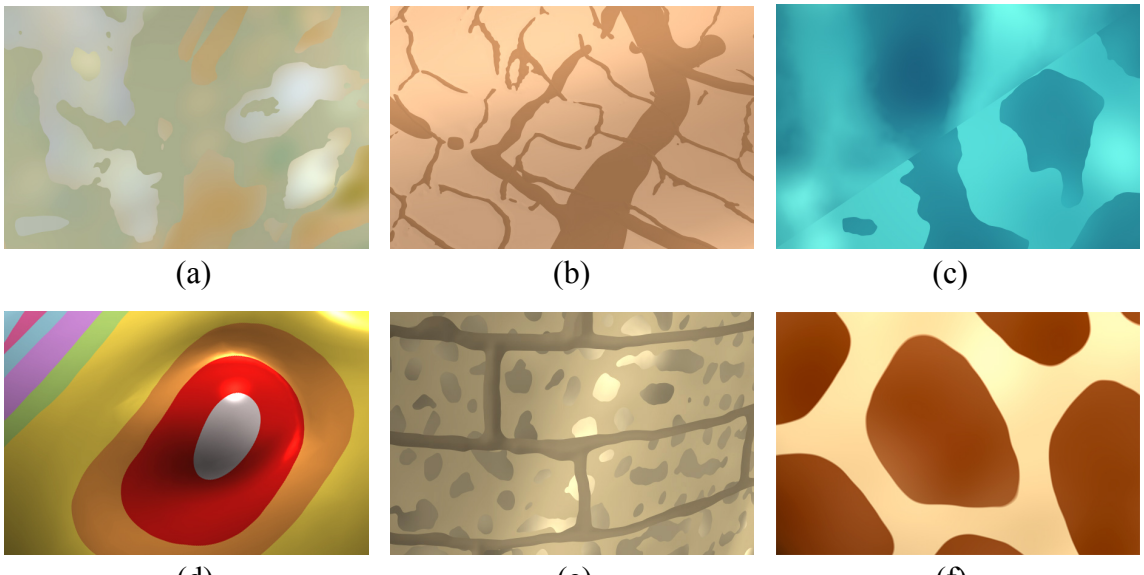
通过本章介绍的方法，我们尝试创建了大量不同类型的区域结构体模型，其中包括根据二维图像样本创建的（图3.15），由离散采样的体模型转化得到的（图3.16与3.17），以及完全手工创建的（图3.18至3.22）。这些体模型与使用传统方法创建的体模型最主要的区别有以下几方面：

- 体模型中材质属性不连续的跳变能够通过区域划分准确表达。由于区域之间的边界是分辨率无关的光滑隐式表面，即使任意放大仍然能保持细节特征的清晰，不会出现模糊或锯齿状瑕疵。
- 体模型中平缓连续的材质属性变化通过径向基函数或过程噪声描述，在确保对非平凡属性变化表达能力的同时大大降低存储开销。本文中出现的体模型存储开销在几十 KB 至十几 MB 之间。对于由传统离散采样体模型转化得到的区域结构体模型，转化后的存储空间占用大约是输入的 20%。
- 通过多尺度区域划分以及体模型的嵌套和实例化，同一个体模型中可以包含尺度相差悬殊的丰富细节特征，同时仍然保持紧凑的存储。

完全手工创建区域结构体模型的过程，包括创建多边形网格，视模型的复杂程度不同所需的时间在几分钟至几个小时之间。将现有的离散采样的体模型转换为区域结构表示所需的时间主要集中在径向基函数的非线性拟合阶段，总的转换时间取决于输入体模型的分辨率、径向基函数的数量等因素，大约在 1 至 20 分钟之间。

此外，本文中出现的所有区域结构体模型均可以在主流的硬件平台上实时绘制。下一章中我们将对与绘制相关的问题以及具体的绘制效率做进一步讨论。


图 3.15　根据二维图像样本创建的一些区域结构体模型。每一列中第二行与第三行比较了区域结构体模型与传统离散采样体模型在放大局部时的绘制效果。区域结构体模型由于分辨率无关性，重要的特征不会因放大而模糊。中间列的体模型材质属性除颜色外还包含镜面反射系数，最右列的体模型中使用了由多个不同尺度基元距离函数叠加而成的复合距离函数，以产生自相似的边界形状。



(a)　(b)　(c)

(d)　(e)　(f)

图 3.16　根据离散采样体模型生成的区域结构体模型，下方小图为各体模型的局部。(b) 中的符号距离函数为多个不同尺度距离函数叠加而成的复合距离函数；(c) 应用了边界模糊特效；(e) 由两个尺度的体模型嵌套而成。

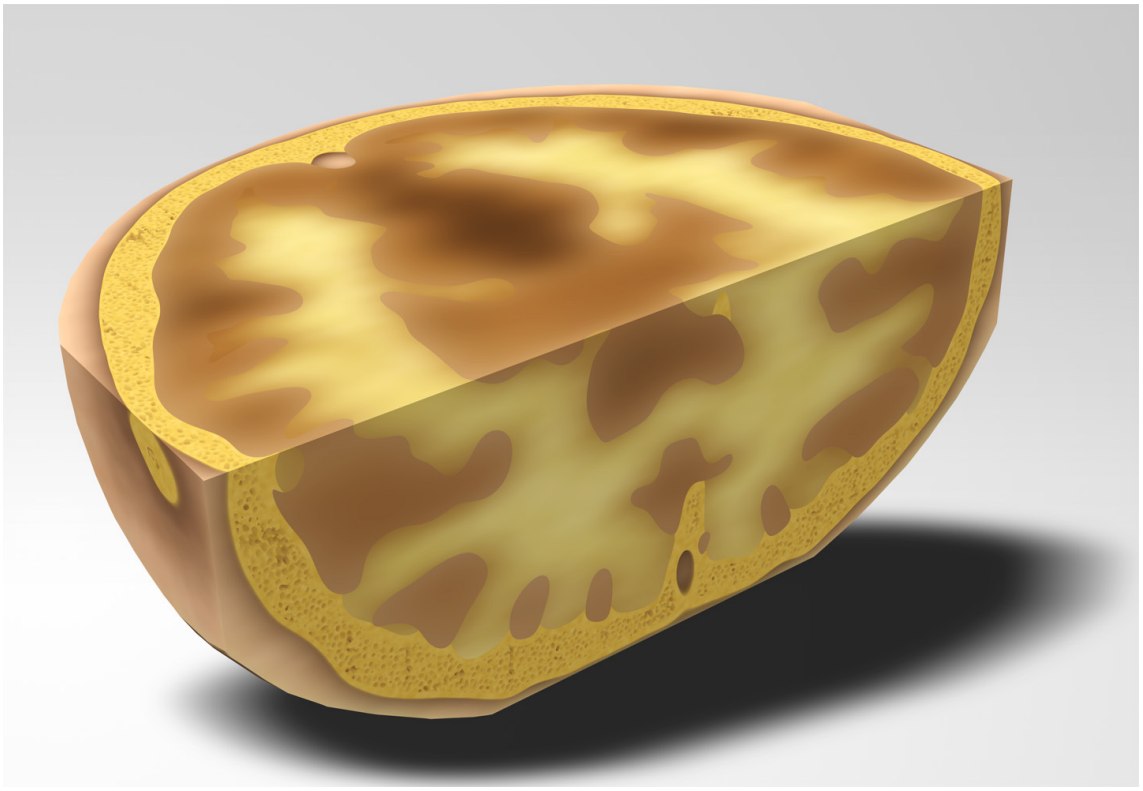


图 3.17　根据离散分区数据生成的大脑：11.6 MB

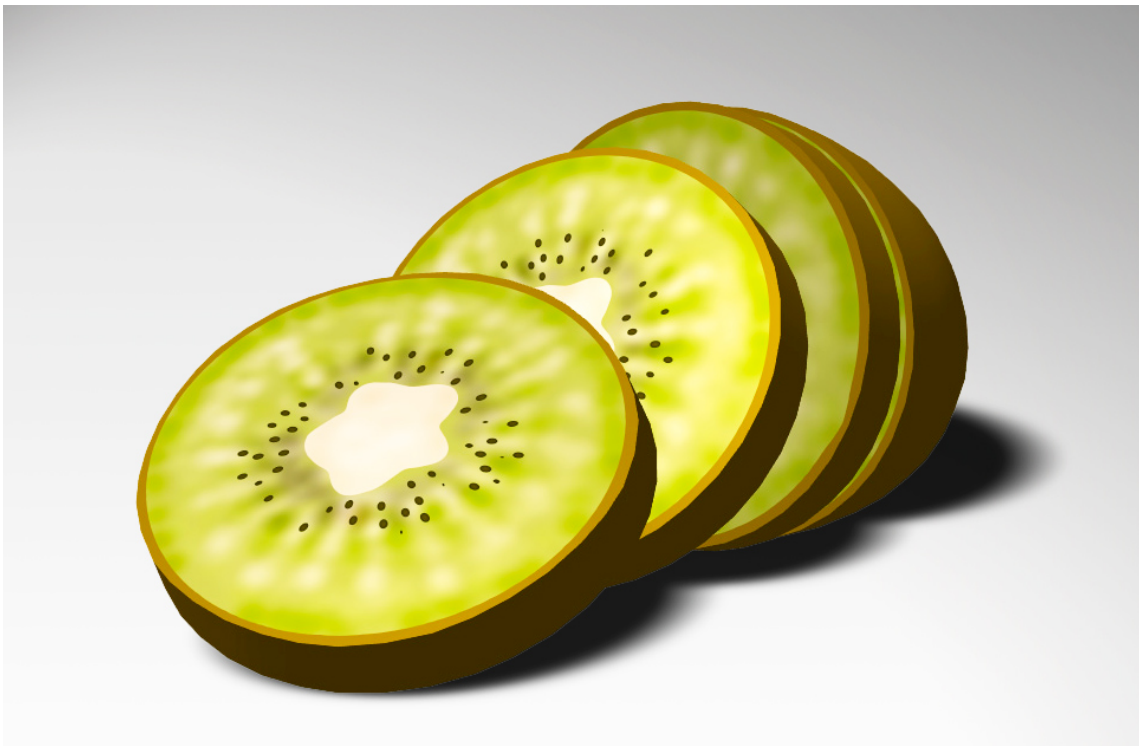


图 3.18　猕猴桃：1.28 MB

图 3.19　人的皮肤：6.80 MB

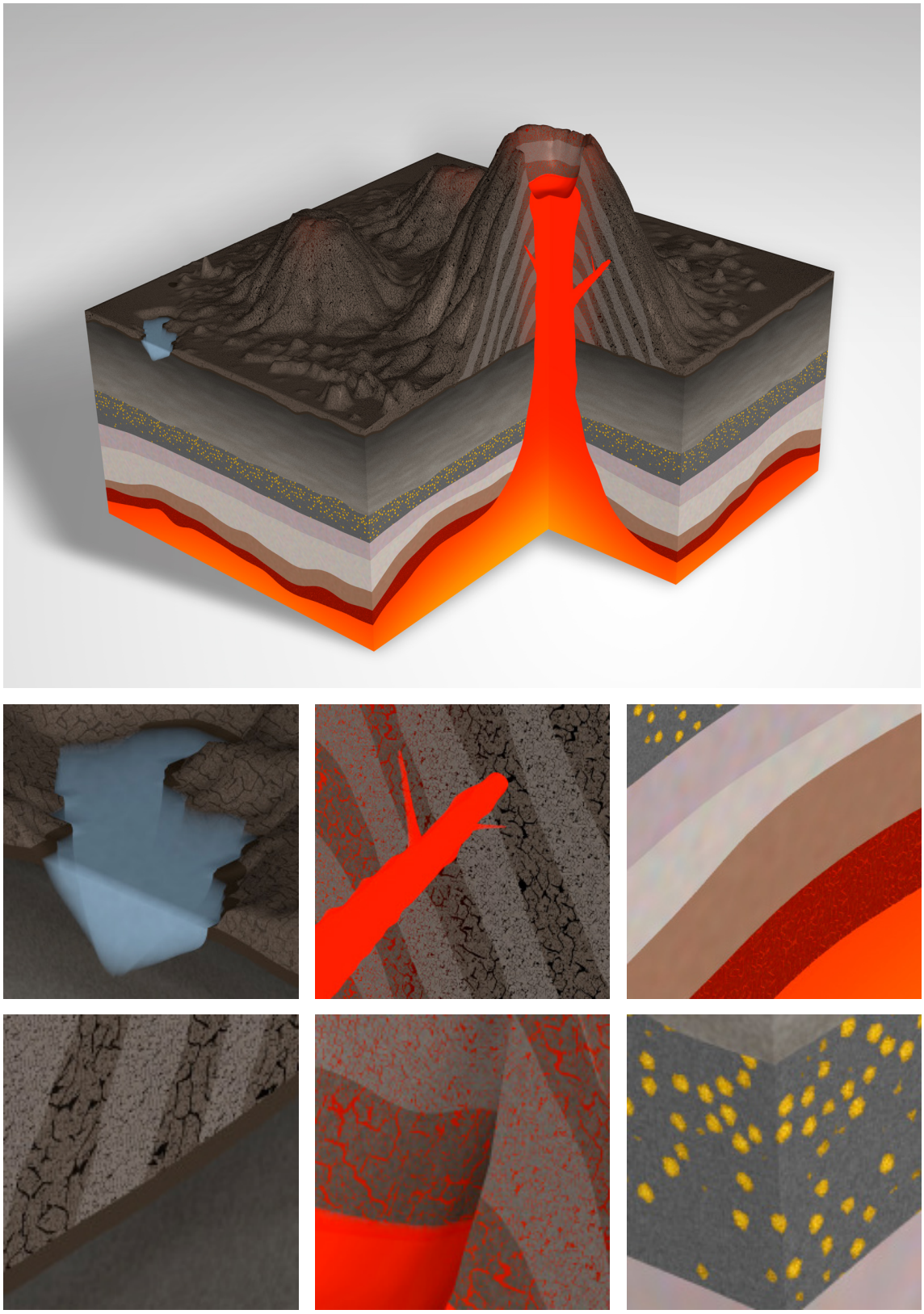


图 3.20　火山：6.50 MB

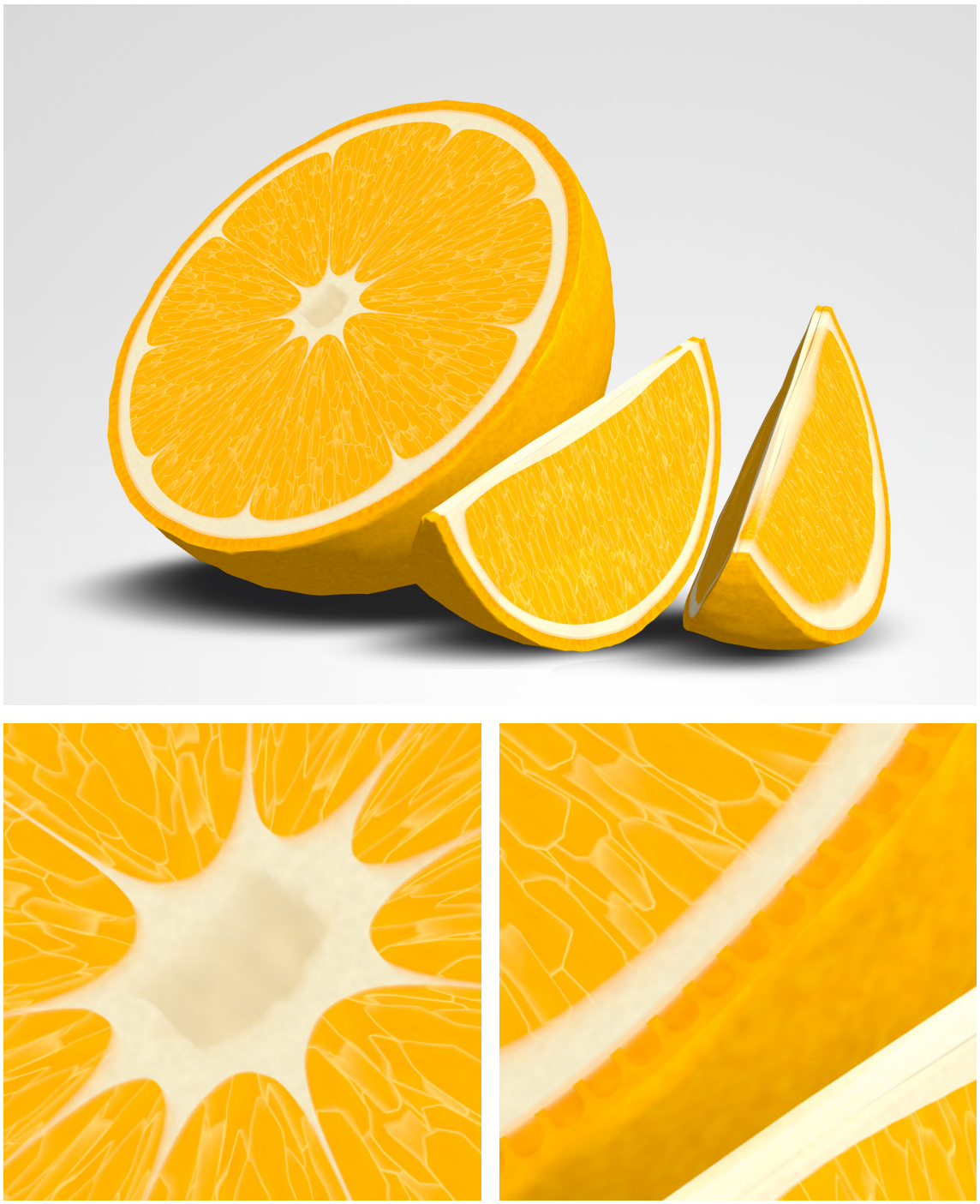


图 3.21　橙子：4.20 MB

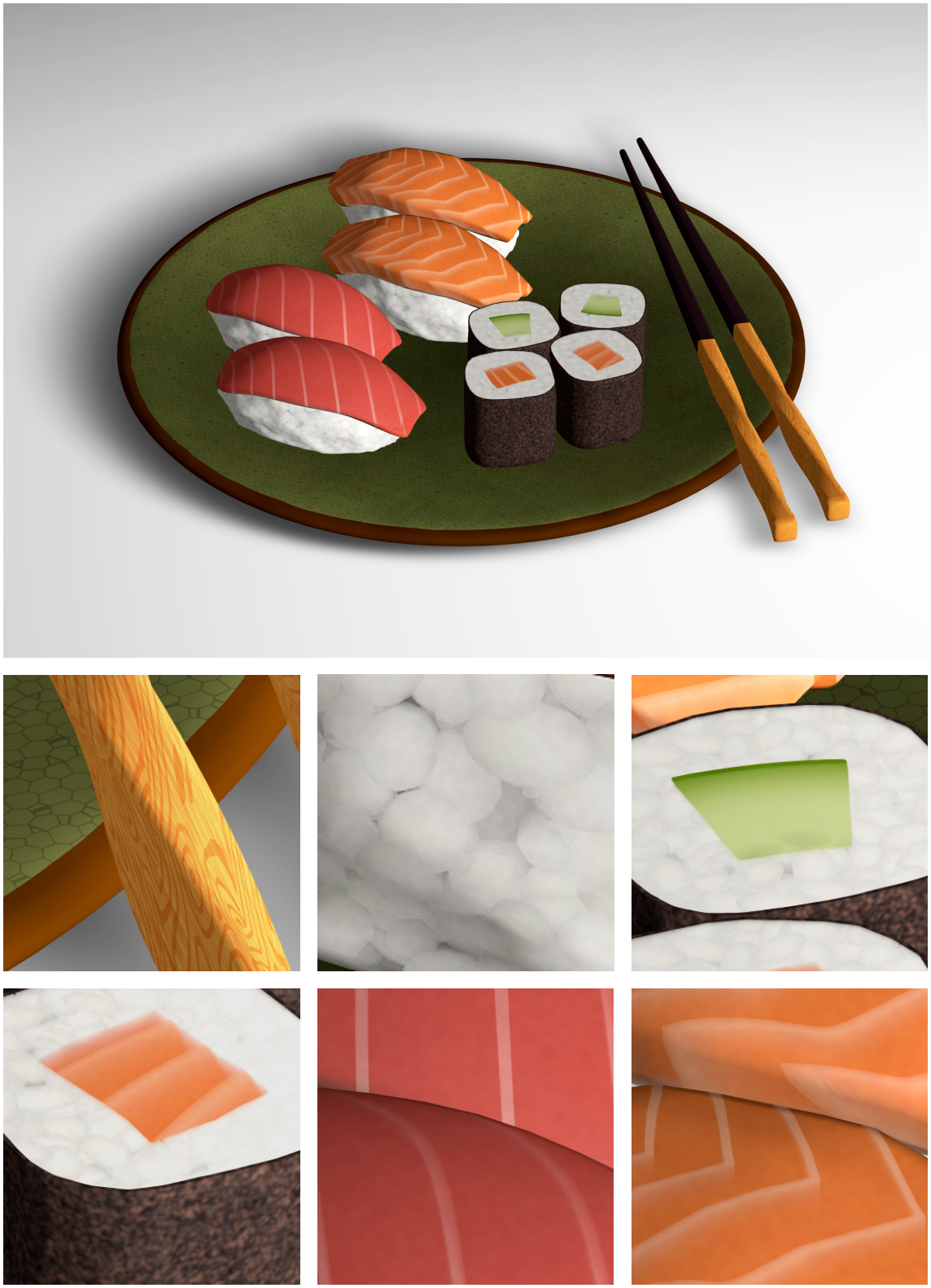


图 3.22　寿司：4.10 MB

## 3.5　本章小结

与传统体数据表达的线性建模方式相比，基于区域结构的体数据表达允许迭代式的建模，因而对用户更加友好。通过多尺度区域结构的定义，复杂对象的建模被分解为多个简单对象的建模问题，通过实例化等方法，大量重复性的子对象仅需一次建模，这使得普通用户可以直接手工创建高度复杂的体模型。除了直接建模，区域结构体模型的创建还可以充分利用现有的资源。例如，由于区域边界采用符号距离函数表示，我们可以利用体纹理合成技术由输入样本生成复杂的边界形状；我们提出的颜色径向基函数优化方法可以将传统的离散采样体数据转化为紧凑的径向基函数表示，在表达丰富颜色变化的同时极大地减少内存开销。结果显示，使用我们的方法可以生成用传统方法难以生成的具有复杂内部结构和丰富细节的体模型。

# 第 4 章　区域结构体模型的存储与绘制

基于区域结构的体数据表达在设计时的两个主要目标就是紧凑的存储与高效的绘制。以目前计算机的发展趋势看，处理器的运算能力增长得比内存带宽更快[51]，在图形处理器上的实时绘制中，绘制效率的瓶颈也往往发生在数据传输部分。因此，更小的内存开销不仅节省有限的空间，也意味着更高效的绘制。在本章中，我们将集中讨论针对区域结构体模型的紧凑存储与高效绘制算法。

在2.4节中我们曾介绍过两种不同的区域划分方式。其中使用单距离函数的区域划分与使用多距离函数（SDF 树）的区域划分相比虽然表达能力受到二着色性限制，但在绘制时具有一些额外的优势，例如支持模糊区域边界的效果。因此本章中介绍的一些算法是针对其中一种特定区域划分方式的。在实际中选择哪种区域划分方式可以根据体模型的特点决定。

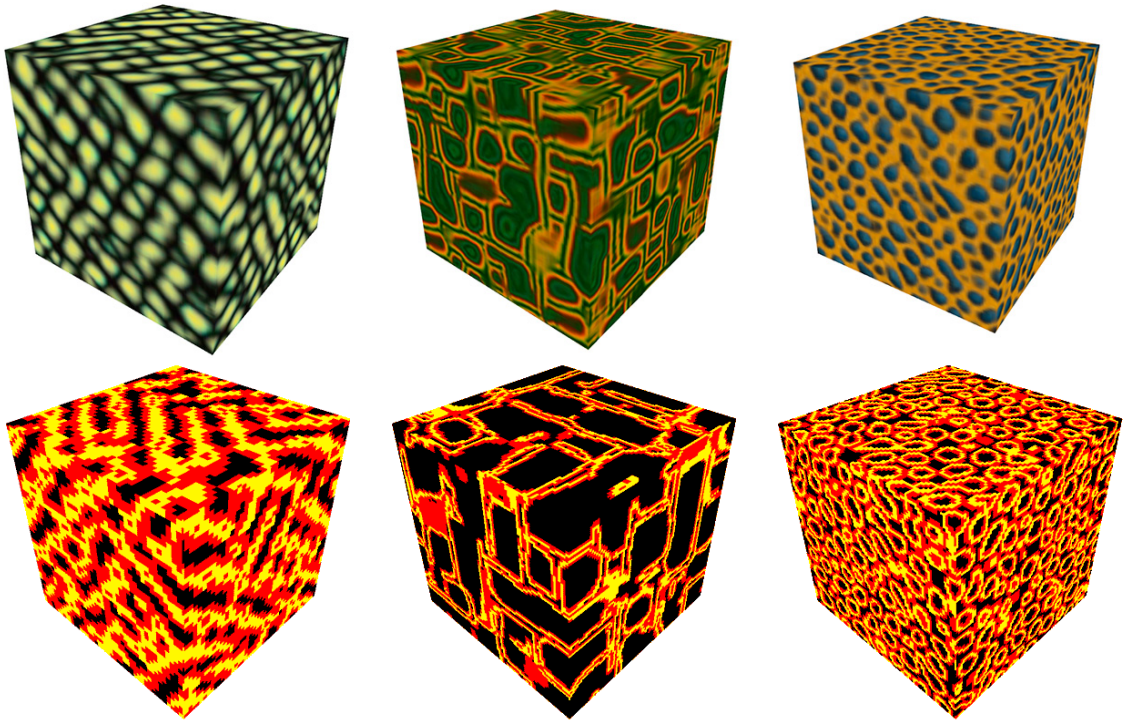
本章余下的部分中，4.1节将介绍区域结构体模型在内存中的布局以及降低存储开销的一些方法；4.2节将介绍区域体模型的绘制算法，包括如何提高绘制的质量与效率两方面，并对本文中出现的区域体模型的绘制效率做出分析。

## 4.1　区域结构体模型的紧凑存储

基于区域结构的体数据表达包括三个主要部分：符号距离函数，区域划分结构，以及各区域内部的定义。这些数据在绘制时均存放于图形处理器的显存中。我们的绘制程序使用 Direct3D 11 实现。其中符号距离函数和单距离函数区域划分使用的区域标记均存储为三维纹理（Texture3D）格式；多区域划分使用的 SDF 树结构、每个区域的内部属性、径向基函数等数据存储为结构缓存（StructuredBuffer）格式；模型中用到的所有线性变换（包括符号距离函数、体模型嵌套与实例化等）统一存储为一个纹理缓存（tbuffer）格式的矩阵数组。在这个基本的内存布局基础上，我们通过不同的方法进一步减小存储空间的占用。

### 4.1.1　符号距离函数的压缩

由于符号距离函数是以离散采样的形式存储的三维纹理，并且一个复杂的区域结构体模型中可能包含大量距离函数，因此距离函数的存储占用对于体模型整体的存储占用影响很大。Frisken 等人[93] 提出了一种自适应采样的距离函数表示（Adaptively-sampled Distance Function，以下简称 ADF）。ADF 的基本思路是用八

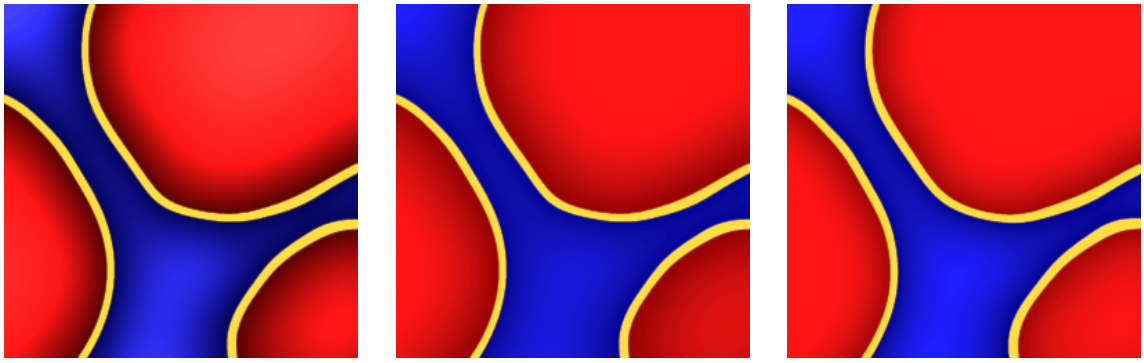


| | | |
|---|---|---|
| $\eta_{\text{linear}} = 21.7\%$ | $\eta_{\text{linear}} = 20.7\%$ | $\eta_{\text{linear}} = 30.1\%$ |
| $\eta_{\text{cubic}} = 54.4\%$ | $\eta_{\text{cubic}} = 48.2\%$ | $\eta_{\text{cubic}} = 75.5\%$ |

图 4.1　一些体模型的符号距离函数中影响零值面形状的体素及其所占比例。其中黄色体素（$\eta_{\text{linear}}$）和红色体素（$\eta_{\text{cubic}}$）分别为三线性插值和三立方插值下影响零值面的体素。

叉树替换规则网格，并根据距离函数中零值面的几何复杂程度决定八叉树的细分深度已达到自适应存储的目的。尽管这种方法对于表示实体几何的符号距离函数很有效，却不容易应用在区域结构体模型中，因为单个实体几何的边界是三维空间中的一个二维流形，决定边界表面形状的符号距离采样点在空间中的分布比较稀疏，因此适合通过八叉树压缩；而区域体模型中的区域边界通常构成复杂的曲面网络，对这些曲面形状起决定作用的采样点空间分布非常稠密（如图4.1所示），这就意味着相应的八叉树为了在表示时不遗漏细节，需要细分至很深的层次，以致存储八叉树结构本身造成的开销大到难以接受。

区域结构体模型中符号距离函数在空间域上的稠密性促使我们转而从值域的角度考虑。初始的距离符号值是 32 位浮点数的格式，然而由于符号距离函数中过零面的形状由其附近采样点的值决定，更具体的说，使用三立方插值时由周围 $4 \times 4 \times 4$ 个网格节点上的距离值决定。对于一个使用欧式距离度量的距离函数，假设三维网格中单元格是边长为 1 的立方形，则通过计算易知影响过零面形状的采样点的距离值均处于 $\left|\mathcal{D}(x)\right| \leq 3\sqrt{3}$ 的范围内。因此我们首先将符号距离函数中的距离值截断在 $\pm 3\sqrt{3}$ 的范围内，再对距离值进行量化（quantization）。我们发现



(a) 32-bit　　(b) 8-bit　　(a) 4-bit

图 4.2　符号距离函数的量化。

对距离值的量化压缩与空间域压缩相比，在同样的压缩比率下能够更好地保持原有的过零面形状，如图4.2。事实上，对于绝大部分区域结构体模型，我们仅需 4 位整型数即可准确描述过零面的几何特征。

### 4.1.2　区域的重标记

2.4.1节介绍的使用单距离函数的区域划分需要额外存储与距离函数定义在相同三维规则网格上的区域标记数据，存储每一个区域标记的数据类型取决于体模型中总的区域数。手工创建的体模型中通常区域数不会太多，但是对于一个由离散采样体模型转换得到的区域结构体模型，其中可能有上千个区域，意味着三维网格中的每个区域标记至少需要一个 10 位的整型数存储。这里我们针对**单距离函数的区域划分**，特别是由离散采样体模型转换得到的区域结构体模型提出一种降低区域标记存储开销的方法。

我们注意到两个事实：第一，对区域进行标记最主要的目的是防止相邻区域的径向基函数相互影响；第二，所有的径向基函数都是有限支撑的且半径有上限。这即是说，如果两个区域内部的径向基函数彼此不影响对方，那么事实上它们可以共享同一个区域标记。

为了自动找出一个给定区域结构体模型中满足上述条件的区域，我们可以将其看作一个经典的图着色问题：给定一现有区域结构体模型的区域标记及每个区域中的径向基函数，我们可以构造一个无向图 $G$，其中 $G$ 的每个顶点 $v_i$ 对应于体模型的一个区域 $\mathcal{P}_i$，$G$ 中的边 $(v_i, v_j)$ 表示区域 $\mathcal{P}_i$ 或 $\mathcal{P}_j$ 中至少有一个径向基函数的影响范围覆盖对方。我们的目标是为这些区域重新分配标记，在相互影响的区域不分配相同标记的条件下尽可能减少使用的标记数，这相当于寻找图 $G$ 的最小着色数（chromatic number）。寻找任意图的最小着色数已被证明是 NP 完全的，因

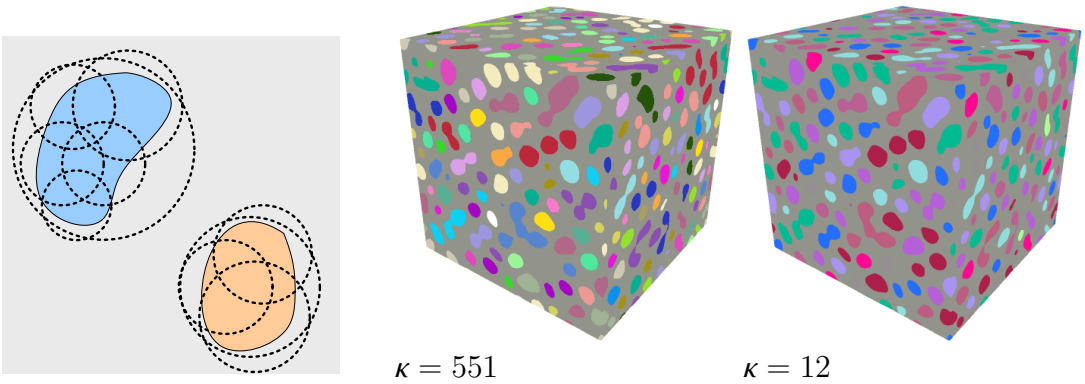



图 4.3　区域重标记。左图中的蓝色区域和橙色区域由于径向基函数不互相影响因此可以共用同一标记。中图一个有 551 个区域的体模型在重标记后减为 12 个区域。

此我们采用了由 Welsh 和 Powell 提出的一种启发式算法[125]。通过实验发现，对于我们测试过的所有体模型，这种算法已足以取得令人满意的结果，将所需的区域标记数降为 19 以下，从而使其可以存储为 5 位的整型数。

图4.3展示了一个区域重标记的结果。

### 4.1.3　区域标记对及其八叉树压缩

以上两种方法均从值域的角度对符号距离函数和区域标记进行压缩。但是与符号距离函数不同，区域标记在每个区域内部是不变的常量，这使我们想到应该有比规则网格更紧凑的存储方式。然而由于区域结构体模型中区域划分的复杂性，直接对区域标记应用八叉树或游程编码（Run-Length Encoding，RLE）等常用压缩方式依然会造成昂贵的存储负担，并且间接影响绘制效率。为此我们提出一种新的针对**单距离函数区域划分**中区域标记的空间压缩方法。

根据2.4.1节的定义，在三维网格 $\mathcal{G}$ 中每个单元格的 $2\times2\times2$ 个节点上最多仅可能出现两种不同的区域标记。两种区域标记的出现意味着必然有符号距离函数的一部分零值面从该单元格中穿过，且这两种标记所对应的网格顶点上的距离符号相反。基于这一事实，我们不再在每个网格节点上存储原始的区域标记，而是在每个**网格单元格中存储两个区域标记**，分别称为“正区域标记”（记为 $\ell_\oplus$）和“负区域标记”（记为 $\ell_\ominus$），且分别对应单元格中距离值为正的网格节点与距离值为负的网格节点的区域标记，我们称这样一对区域标记为“区域标记对”。

显然，在许多没有过零面穿过的单元格内，所有的网格节点均拥有同样的距离符号，因此正区域标记 $\ell_\oplus$ 或负区域标记 $\ell_\ominus$ 之一是没有定义的。我们将所有的正区域标记和所有的负区域标记分别看作独立的三维数组，其中各自存在未定义的元素，我们使用 fast marching 算法[120] 分别将这两个三维数组中有定义的元素

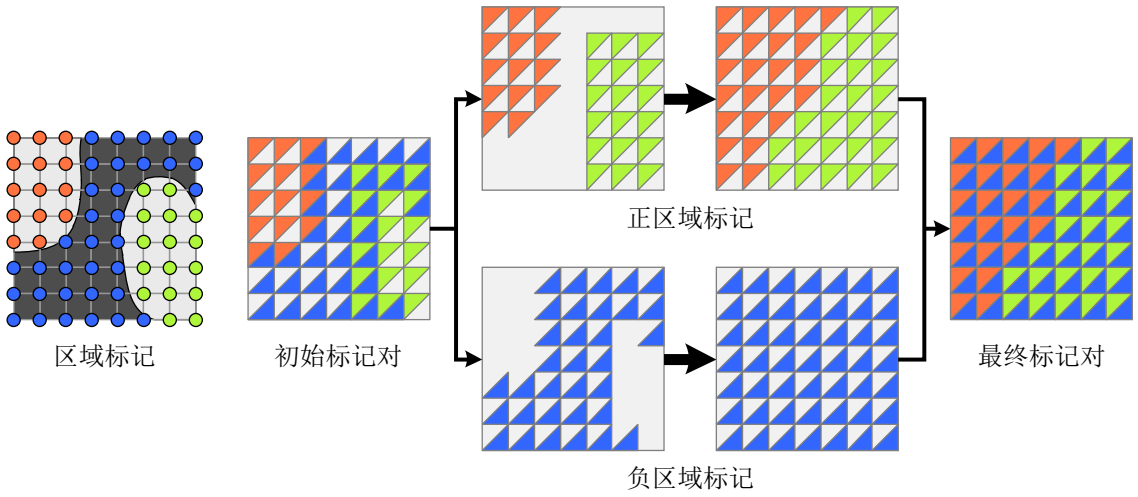



图 4.4　区域标记对的生成过程。

扩散至填满所有未定义的元素，从而使得三维网格的任意单元格内 $\ell_{\oplus}$ 和 $\ell_{\ominus}$ 均有定义。对于一个初始时仅定义了一种区域标记的单元格，由 fast marching 计算出的另一种区域标记的直观含义是距离此单元格最近的相邻区域，因此通过区域标记对，我们不仅存储了体模型中每一点的所属区域，同时还存储了距离该点最近的相邻区域信息。这一过程如图4.4所示。

我们的最终目的是减小存储开销，而是直觉上看，存储区域标记对需要的存储空间翻了一番。但是实际上，由于区域标记对结构在空间中各个方向上的跳变远远少于原始的区域标记结构，因此十分适合使用常用的空间压缩方法。我们将区域标记对存储为一个八叉树结构并以类似 Lefebvre 等人[25] 的方式在图形处理器上实现。通过我们的实验，以八叉树的方式存储区域标记对能够比在规则网格上存储原始的区域标记节省 70% 至 99.998% 的存储空间。由于对内存带宽需求的相应降低以及对随机访问的优化，使用区域标记对甚至能为绘制效率带来 5% 左右的提升。

## 4.2　区域结构体模型的绘制

区域结构体模型的绘制总的来说可以概括为**判断区域归属**$\ell(x)$ 以及**计算相应区域内的材质属性函数**$C_{\ell(x)}(x)$ 两步。

对于单距离函数的区域划分，我们通过区域标记对结构判断区域归属，表4.1列出了基本算法。对于多距离函数的区域划分，我们通过 SDF 树遍历判断区域归属，表4.2列出了相应流程。对于一个具有多尺度区域结构、嵌套以及实例化的复杂体模型，在判断空间中点 $x$ 区域归属的过程中 $x$ 的坐标需要由最初的世

表 4.1　单距离函数区域划分的体模型绘制

**procedure** RenderWithSingleSDF($x$)　　▷ $x \in [0,1]^3$
　$d \leftarrow \mathcal{D}(x)$　　▷ 三立方插值
　$(\ell_{\oplus}, \ell_{\ominus}) \leftarrow P(x)$　　▷ $x$ 所在单元格的区域标记对
　**if** $d > 0$ **then** $\ell \leftarrow \ell_{\oplus}$
　**else** $\ell \leftarrow \ell_{\ominus}$
　$c \leftarrow C_{\ell}(x)$
　**return** $c$

表 4.2　SDF 树区域划分的体模型绘制

**procedure** RenderWithSDFTree($x$)
　$T \leftarrow T_1$　　▷ 体模型的 SDF 树
　$node \leftarrow T.root$
　**while** $node.type = SDFNODE$ **do**
　　$d \leftarrow node.\mathcal{D}(x)$
　　**if** $d > 0$ **then**
　　　$node \leftarrow node.posChild$
　　**else**
　　　$node \leftarrow node.negChild$
　$\ell \leftarrow node.regionLabel$
　$c \leftarrow C_{\ell}(x)$
　**return** $c$

界坐标被逐层变换到各局部坐标系，直到达到所属区域后，根据最终变换过的局部坐标计算材质属性函数。图4.5展示了一个复杂体模型各点的最终局部坐标。

符号距离函数的三立方插值使用了 Sigg 和 Hadwiger 提出的加速算法[8]。该方法利用图形处理器对三线性插值的硬件支持，将直接实现三立方插值所需的 64 次纹理访问操作转化成 8 次经三线性插值的纹理访问，从而大幅减小了绘制开销。

此外，为了进一步提高区域结构体模型的随机访问性能，我们为实例化的符号距离函数、体模型以及径向基函数构建了空间索引。这些数据的共同点是数量大但是单个对象的影响范围有限，为此我们构造一个稀疏的三维网格，并在网格

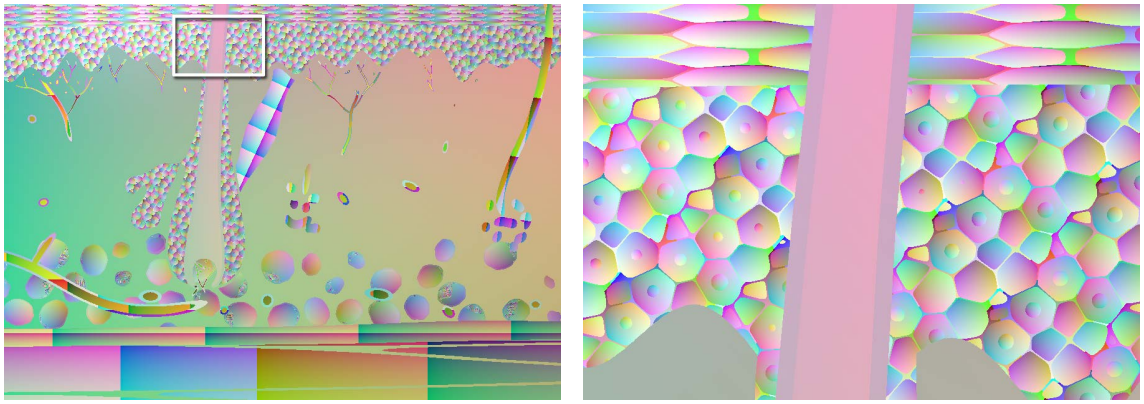


图 4.5　经过 SDF 树遍历后各点的最终局部坐标。

每个单元格中记录对该单元格产生影响的对象。这相当于将 Nehab 和 Hoppe[70] 用于加速图形处理器上二维矢量图绘制所用的 indirection table 扩展到三维。值得注意的是，对于多尺度嵌套的区域结构体模型，每个被嵌套的子模型均可以有自己的空间索引，因此尽管单个的空间索引结构是规则网格划分，但从整体上看体模型的空间索引是根据细节丰富程度自适应划分的。

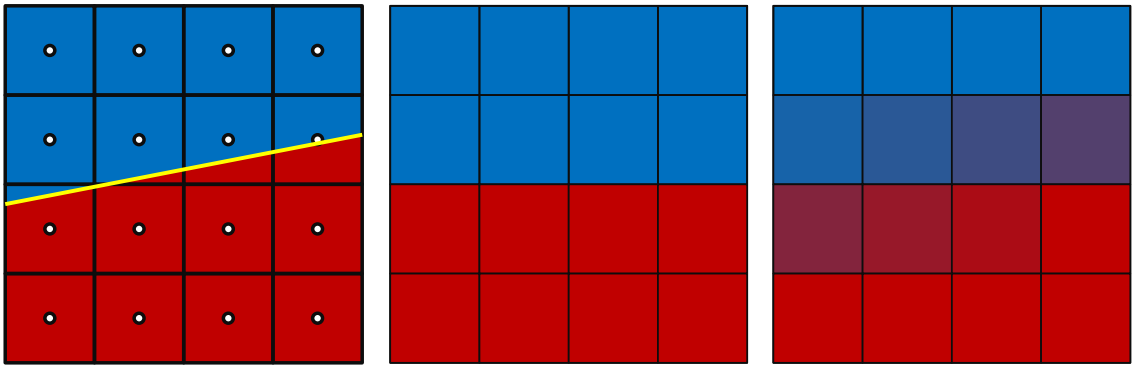
### 4.2.1　三维纹理的拼装与索引

如前所述，一个区域结构体模型会使用大量三维纹理，但是目前的图形硬件及 API 并不支持动态地索引三维纹理，亦即绘制每个像素时需要访问哪些三维纹理无法根据该像素的位置动态决定。为此，我们不得不将同一类型（例如符号距离函数）的所有三维纹理拼装成一个大的三维纹理，并通过三维纹理坐标索引某个小纹理在大纹理中的位置，相当于一个三维的 texture atlas。

自动拼装给定的一组三维纹理并尽可能减少空间浪费可以看作是一个三维装箱问题（bin-packing problem），虽然寻找一个非平凡输入的最优解是 NP 困难的，但实际应用中存在许多启发式算法。我们选用 Corcoran 和 Wainwright 提出的启发式算法[126] 在效率与质量之间取得了较好的平衡。我们希望将来的图形硬件及 API 能够支持三维纹理的动态索引，从而完全消除纹理拼装带来的空间浪费与索引的额外时间开销。

### 4.2.2　反走样

与二维矢量图的绘制类似，由于区域结构体模型中对区域边界的明确定义以及各区域内部独立定义的材质属性，在绘制时屏幕上两个相邻的像素有可能分别属于两个不同区域，从而造成走样（aliasing）。尽管随着显示设备的发展，具有更高像素密度（$> 150$ dpi）的屏幕会使得这种走样不容易被人眼察觉，但对于目



(a)　　(b)　　(c)

图 4.6　走样与反走样。(a) 将以黄线为边界的两个区域光栅化为 $4\times4$ 的像素区域；(b) 若直接根据像素中心所在区域决定颜色会导致走样现象；(c) 反走样的基本思想是根据像素内部不同区域面积比例计算像素最终颜色。

前大多数低像素密度（$\approx 72$ dpi）的桌面显示设备，反走样（anti-aliasing）仍然是提高绘制质量的重要手段。

在区域结构体模型的绘制中，我们分别针对单距离函数区域划分和多距离函数区域划分采用了两种不同的反走样方法，下面将逐一进行介绍。

#### 4.2.2.1　单距离函数

走样出现的根本原因是由于在绘制时根据每个像素**中心所在位置**决定整个像素所属的区域以及最终颜色，但实际上每个像素并非数学意义上的点，而是具有一定面积的形状（为方便讨论通常将其看作正方形），因此像素的颜色应该由像素内部各区域颜色共同决定，理想状况下，各区域对最终颜色的影响应该和它们在像素内部所占面积的大小成正比，如图4.6所示。

目前应用最广的反走样算法均是通过提高每个像素内的采样点数，再对所有的采样值进行（加权）平均以决定最终的像素颜色。理论上，随着采样数目的提高这种方法能够尽可能地逼近理想结果，然而在实际中这意味着更高的绘制开销。在电影等对画质要求很高的离线绘制应用中每个像素内的采样数目可能高达 $13\times13$，而在游戏等实时绘制应用中，为了保证绘制速度采样数目通常在 $8\times8$ 之内，即使如此与未使用反走样相比仍然会显著降低绘制速度。

在4.1.3节中介绍过一种针对单符号距离函数区域划分的区域标记对结构。通过这一结构，对于任意一点 $x$ 我们不仅知道它当前所处的区域 $\ell_\oplus$（或 $\ell_\ominus$），还知道距离 $x$ 最近的相邻区域 $\ell_\ominus$（或 $\ell_\oplus$），并且符号距离函数的绝对值 $\left|\mathcal{D}(x)\right|$ 明确给出了 $x$ 点至区域 $\ell_\oplus$ 与区域 $\ell_\ominus$ 之间边界的最短距离。利用这一特点，在任意一点 $x$，我们可以计算两个颜色值：$C_\oplus$ 和 $C_\ominus$，分别对应区域 $\ell_\oplus$ 与区域 $\ell_\ominus$ 各自的材质

属性函数在 $x$ 点的值：

$$\begin{aligned} C_{\oplus}(x) &= C_{\ell_{\oplus}(x)}(x) \\ C_{\ominus}(x) &= C_{\ell_{\ominus}(x)}(x) \end{aligned} \tag{4-1}$$

$x$ 点的最终颜色 $C(x)$ 则由这两个颜色值的线性插值计算得到：

$$C(x) = (1-\omega)C_{\ominus}(x) + \omega C_{\oplus}(x) \tag{4-2}$$

其中：

$$\omega = \frac{\mathcal{D}(x)+\delta}{2\delta}, \quad \left|\mathcal{D}(x)\right| < \delta \tag{4-3}$$

上式中的 $\delta$ 为一距离阈值，限定了体模型中需要根据4-2式计算颜色值的范围。直观上看，这相当于在体模型中所有区域边界附近定义了一个宽度为 $2\delta$ 的空间，在这一空间内的颜色是两个临近区域颜色的混合，从而形成模糊的区域边界。

为了实现准确的屏幕反走样，我们需要仔细选择 $\delta$ 以使得混合区域颜色的空间宽度恰好等于体模型被光栅化后对应屏幕上一个像素宽度的二倍。也即是说，当且仅当一个像素内部存在某两个区域之间的边界时，像素的颜色由这两个区域的颜色根据4-2式混合而成。为此我们将点 $x$ 处的距离阈值 $\delta$ 定义为：

$$\delta(x) = 0.5\|\nabla\mathcal{D}(x)\| \tag{4-4}$$

其中 $\mathcal{D}(x)$ 是 $x$ 点的符号距离值，$\nabla\mathcal{D}(x)$ 是符号距离函数在 $x$ 点处的**屏幕空间**梯度。利用目前图形处理器对屏幕空间偏导数计算的硬件支持（如 HLSL 语言中内置的 ddx 和 ddy 函数），我们可以在像素着色器中快速计算 $\nabla\mathcal{D}(x)$。由于这种反走样算法不需要在像素内部进行多次采样，因此对绘制效率的影响很小，并且由于利用符号距离函数的性质更准确地估计像素内部不同区域的面积比例，取得了比采样数较低时的多重采样反走样算法更高的画面质量，如图4.7所示。

这种方法的另一个好处是通过增大距离阈值 $\delta$，我们可以得到颜色平滑过渡的区域边界，类似于二维矢量表达 Diffusion Curves[59] 中模糊属性（blur attribute）的效果。这里 $\delta$ 可以定义为区域结构体模型中随空间位置而变化的一个内在属性，从而实现区域边界一部分清晰一部分模糊的特殊效果。另外，我们也可以将 $\delta$ 定义为随屏幕像素位置而变化的属性，例如在图4.8中，我们根据体模型表面各点在视空间的深度决定距离阈值 $\delta$，从而产生类似浅景深的效果。

值得注意的是，这种模糊区域边界的算法是常数时间复杂度的，对绘制效率的影响不随模糊半径（距离阈值 $\delta$）变化而改变。但同时这种方法有一个局限：

$\delta = 0$　　$\delta = 0.5\|\nabla \mathcal{D}(x)\|$　　$\delta = 0.3$

图 4.7　利用距离函数特性的反走样及边界模糊。从左至右依次是：未使用反走样；根据屏幕空间梯度设定距离阈值实现反走样；手工设定距离阈值实现边界模糊。

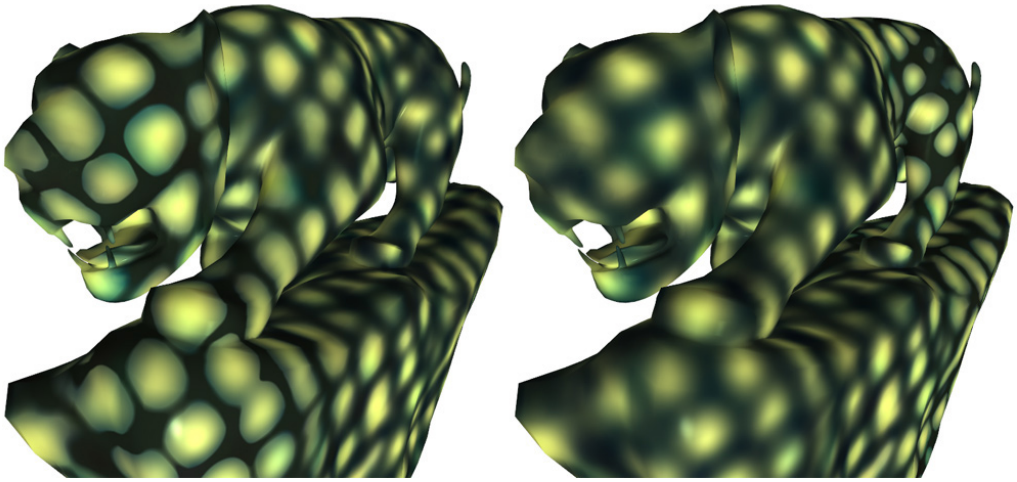


图 4.8　模拟景深效果。左图中焦点在雕像头部，右图中焦点在雕像尾部。

由于4-2式仅对两个临近区域的颜色进行混合，当模糊半径覆盖超过两个区域时有可能会产生不正确的结果，因此半径不能无限制地增大，如图4.9所示。在实际绘制时，为屏幕反走样而定义的距离阈值（记为 $\delta_{2D}$）与为边界模糊而定义的距离阈值（记为 $\delta_{3D}$）可以统一定义为 $\delta = \max(\delta_{2D}, \delta_{3D})$。

#### 4.2.2.2　多距离函数

4-2式的方法可以取得高质量、高效率的反走样结果，但是只适用于单符号距

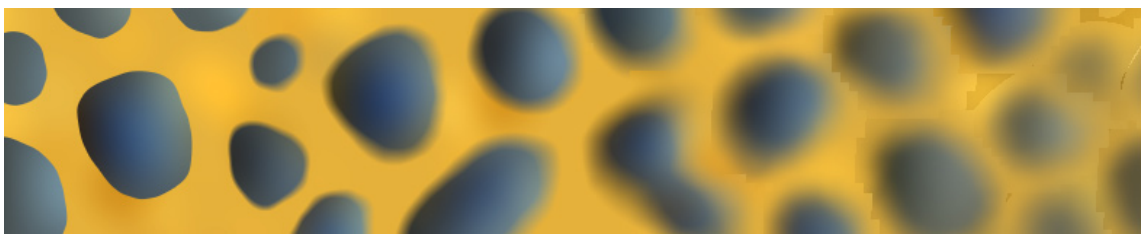


图 4.9　边界模糊的限制。随着模糊半径的增大，颜色混合可能会产生不正确的结果。

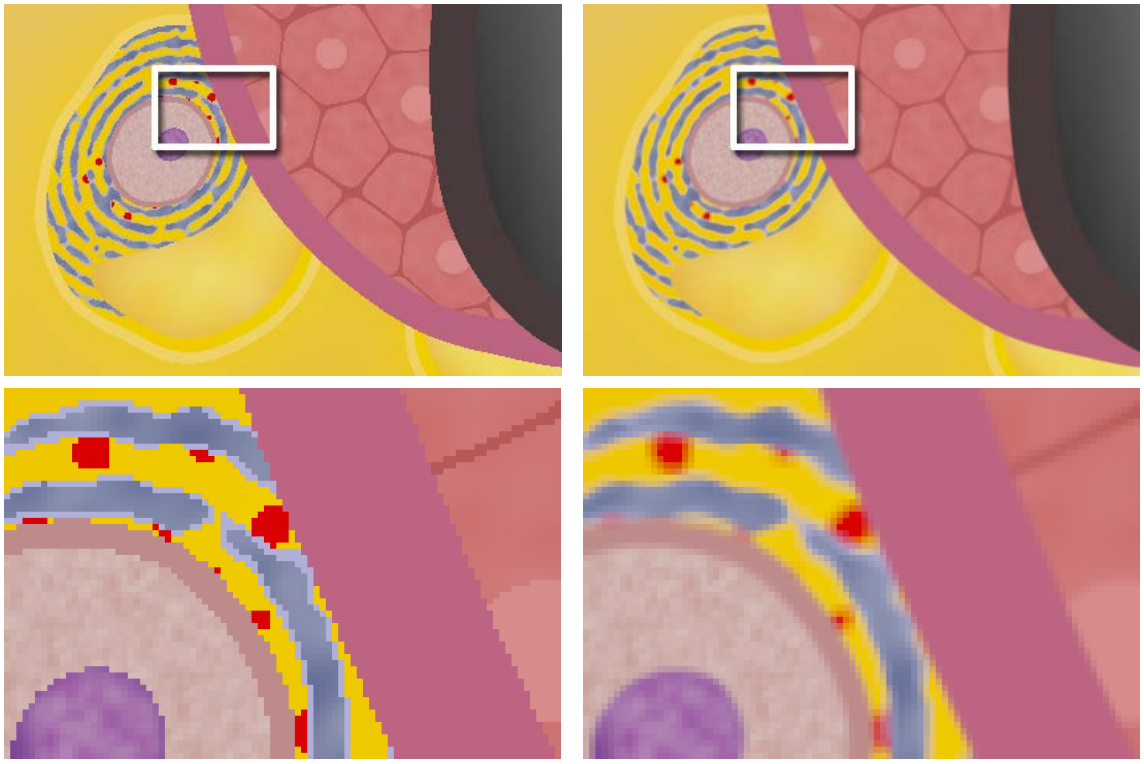


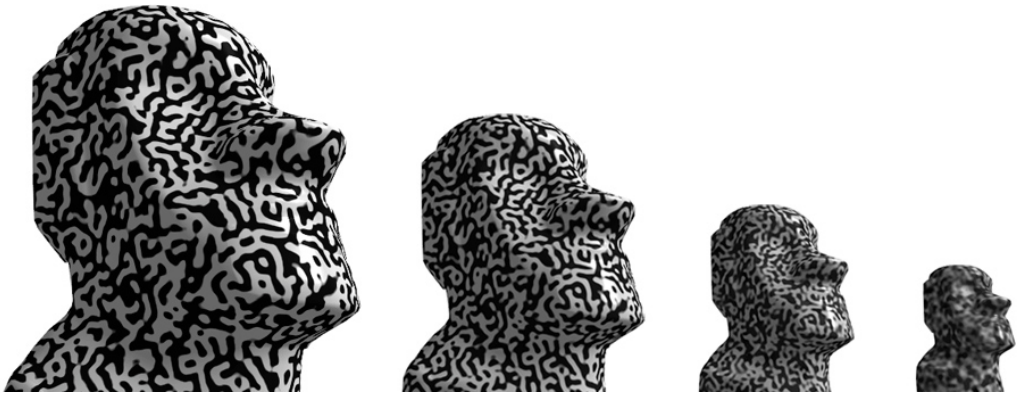
图 4.10　左图为未使用反走样的绘制效果，右图为使用屏幕空间反走样的绘制效果。

离函数区域划分的体模型。对于通过 SDF 树定义区域划分的体模型，我们采用一种不同的反走样策略。

为了避免多重采样造成的性能负担，我们的算法基于延迟渲染（deferred shading）中经常采用的后处理（post-processing）式反走样[127]。算法分为两次绘制过程，在第一次绘制时，像素着色器除了颜色值以外，还将每个像素的**区域标记**以及**最小符号距离值**一起写入屏幕缓存；在第二次绘制过程中，像素着色器将每个像素的区域标记与其周围像素的区域标记对比，以决定该像素是否处在区域边界处，若是，则将周围 $3 \times 3$ 范围内像素的颜色进行加权平均。与通常延迟绘制中使用的反走样算法不同，由于体模型中区域边界是由符号距离函数隐式表达的，我们可以根据每个像素的符号距离值决定计算平均颜色时的权重，因而得到更好的反走样质量，如图4.10所示。

### 4.2.3　Mipmapping

除了4.2.2节中讨论的因相邻像素所属区域不同造成的走样以外，绘制中另外一类常见的走样现象发生在光栅化的分辨率小于物体材质属性本身的分辨率时，例如绘制一个距离视点很远的物体。这种走样问题在传统的体模型表达，如离散采样或过程噪声表达中同样存在。其中离散采样数据最常见的方法是对原始数据



$128^3$　　$64^3$　　$32^3$　　$16^3$

图 4.11　Mipmap 层次

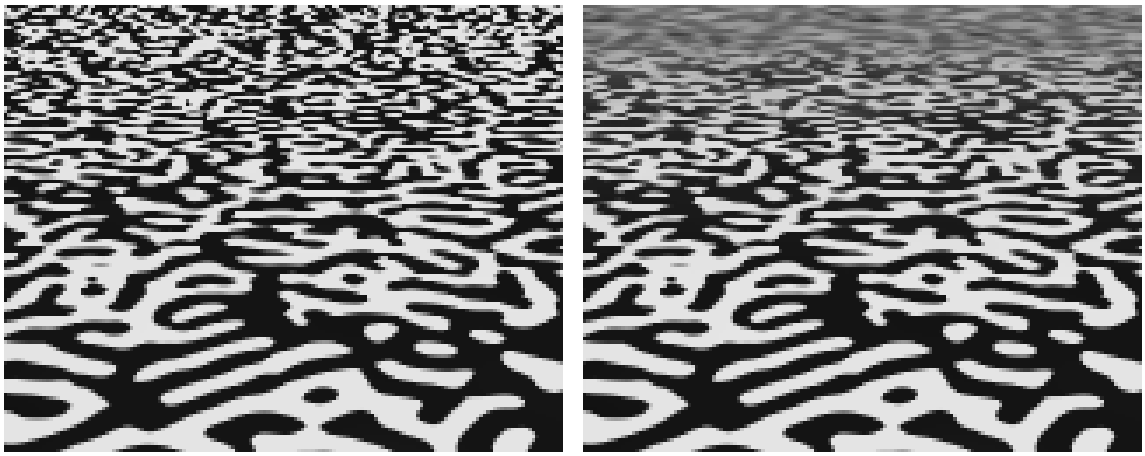


图 4.12　Mipmap 效果对比。左图为未使用 Mipmap 的绘制效果，右图为使用 4 层 Mipmap 的绘制效果。

预先进行多级的低通过滤和降采样，构成一个 mipmap。在绘制时根据物体与视点的远近选择 mipmap 中合适层中经过预过滤（pre-filtered）的数据，从而避免走样现象。

在区域结构体模型中我们可以采用类似的方法构造 mipmap。在 mipmap 中的每层，符号距离函数的分辨率都是上一层的 $1/2$，每个区域内的径向基函数数目为上一层的 $1/8$，如图4.11所示。

在绘制时，我们根据体模型表面各点与视点之间的距离以及体模型中符号距离函数的分辨率选择 mipmap 中最合适的两层，并对两层的绘制结果进行线性插值[128]。图4.12比较了未使用 mipmap 与使用后的绘制结果。

表 4.3　本文中体模型的绘制效率统计。总内存开销包括体模型本身以及空间索引结构。绘制速度数据是在 NVidia GeForce 8800 GTX 图形处理器上绘制分辨率为 $800 \times 800$ 的区域时测量所得。

| 体模型 | 网格 | RBF 数目 | 总内存开销 (MB) | | 绘制速度 (FPS) |
|---|---|---|---|---|---|
| | | | 体模型自身 | 空间索引 | |
| Fig. 3.15 (左) | $128^3$ | 5183 | 1.117 | 1.173 | 148.6 |
| Fig. 3.15 (中) | $128^3$ | 202 | 1.002 | 1.016 | 549.7 |
| Fig. 3.15 (右) | $128^3$ | 2015 | 1.050 | 1.087 | 247.1 |
| Fig. 3.16 (a) | $128^3$ | 1387 | 1.120 | 1.152 | 252.4 |
| Fig. 3.16 (b) | $128^3$ | 1060 | 1.074 | 1.100 | 335.5 |
| Fig. 3.16 (c) | $128^3$ | 352 | 1.004 | 1.026 | 363.2 |
| Fig. 3.16 (d) | $128^3$ | 69 | 1.497 | 1.548 | 146.4 |
| Fig. 3.16 (e) | $128^3$+$64^3$ | 3893 | 1.290 | 1.369 | 150.5 |
| Fig. 3.16 (f) | $128^3$ | 4735 | 1.141 | 1.230 | 144.0 |
| Fig. 3.18 | $128^3$ | 7137 | 1.199 | 1.275 | 140.8 |
| Fig. 4.8 | $96^3$ | 5002 | 0.479 | 0.569 | 96.5 |

## 4.3　本章小结

表4.3列出了本文中出现的一些区域结构体模型的存储空间占用以及绘制效率。

基于区域结构的体数据表示本身的多尺度和分辨率无关特性使得我们不需要通过盲目提高采样频率的方式表达更多的细节，因而可以做到紧凑的存储。在本章中，我们提出了一系列算法进一步减小了区域结构体模型的存储开销。本文中出现的区域结构体模型的存储空间占用在数百 KB 至十几 MB 之间，而为了取得类似的视觉质量，传统的离散采样体模型需要几 GB 甚至更多的存储空间——取决于绘制时采用的分辨率。

利用表示区域边界的符号距离函数特性，我们提出了一种高质量的屏幕反走样算法，这种算法在质量上优于实时绘制中常用的多重采样反走样算法，并且对绘制效率的影响很小。对于使用单距离函数进行区域划分的体模型，屏幕反走样的算法可以进一步扩展以实现模糊区域边界的效果。在空间索引结构的帮助下，高度复杂的体模型可以在目前主流的图形处理器上实时绘制，并且不需要在物体表面发生变化时进行任何预计算。

# 第 5 章　基于区域结构的头发几何建模

头发或毛发是人和动物的一种非常重要的外观特征，也是被影视作品广泛用于塑造人物角色的一种手段。准确的头发建模往往能极大提高计算机生成虚拟人物的真实感，因而在计算机图形学，特别是真实感绘制与动画领域中具有重大意义。本章将讨论如何利用基于区域结构的体数据分析、表示与生成具有高度真实感和复杂性的头发几何模型。

头发的建摸长久以来一直困难重重。这里的困难主要源自真实头发本身的复杂性：人类通常有 10 万根左右发丝；每根发丝都是三维空间中的一条曲线，而其直径却仅有 40-120 微米；这些发丝的形态受到多种因素的影响，且彼此之间有复杂的相互作用。从绘制的角度，头发这种发丝数量巨大而每根发丝十分细微的对偶性已然为高效率、高质量的绘制带来一系列挑战，而发丝对光线的复杂散射方式更进一步加深了绘制困难[129]。不幸的是，目前仍然没有一个普遍接受的数学或物理模型能够准确全面地描述真实头发各方面的复杂性。

头发的复杂性使得一个复杂发型的建模过程——不论手工建模还是基于图像的采集方法都相当费时费力。而由于缺少结构信息，建模结果通常难以被重用或进一步编辑[130]，重复地创建相似的发型更进一步增加了建模代价。因此，有必要寻找一种基于样本的建模方法，能够根据一个已有的头发几何模型（样本）高效地产生具有相似特征的新模型。基于样本的方法可以概括为对输入样本的分析与合成两个步骤，其中分析阶段从原始输入样本中抽取所需的特征，合成阶段则将抽取出的样本特征通过特定的随机过程产生新的输出。如何准确、合理地抽取和描述样本特征是基于样本方法需要解决的一个重要问题。

头发虽然不属于一般意义的体数据，但是头发作为一个整体表现出明显的“**体特征**”，例如：具有一定长度的发型中，靠近头皮的头发通常完全或部分地被靠近外侧的头发遮挡，这些处于整体内部的头发虽然未必直接可见，但对整个发型的几何形状、动态、光影效果都会产生不可忽略的影响。许多研究者采用离散采样的三维向量场（3D vector field）描述发型内部各点的局部头发朝向并以此作为辅助头发建模的工具[130–133]。另一方面，许多复杂的真实发型亦表现出显著的“**结构特征**”，例如发簇、辫子等。一些手工头发建摸的方法利用这种结构特征，将头发整体看作一个多层的发簇结构，允许用户以自顶向下的方式创建或编辑头发几何模型[134]。

头发的体特征和结构特征使得我们可以通过基于区域结构的体数据表达加以描述。具体来说，我们以 Kim 和 Neumann 提出的多层发簇结构[134] 为基础，但将每一个发簇（具有一致几何特征的一组相邻发丝）视作一个独立区域，提出一种基于区域结构的头发几何表示。根据这种表示，我们提出一种基于样本的头发几何建模方法。该方法包括分析（analysis）与合成（synthesis）两个步骤：在分析步骤（5.2节）中，我们提出一种基于变分思想的层次区域划分算法，能够自动从一个完整的输入发型中提取发簇区域结构，并将每个发簇区域内部发丝的几何特征以一组特征向量表述；在合成步骤（5.3节）中，利用分析得到的头发特征描述，我们尝试了多种基于样本的头发几何合成应用，包括快速生成与输入发型具有类似特征的新发型、混合多个不同发型的特征、保持局部特征的交互式发型编辑等。

## 5.1　头发几何建模与相关工作

计算机图形学中对头发的研究主要分为三方面：几何建模、动态模拟和绘制。这里我们仅关注几何建模领域。Watanabe 和 Suenaga[135] 提出了一种基于发簇（wisp）的头发几何表达，Chen 等人[136] 进一步扩展了他们的方法。Daldegan 等人[137] 通过特征发丝定义发束的边界，并实现了一个头发建模系统。Hadap 和 Thalmann[138] 以及 Yu[131] 发现特定的发型可以表示为一个三维向量场，根据这一联系，他们提出了不同的交互式头发建模方法。Choe 和 Ko[139] 基于统计学和物理学的方法设计了一套建模系统，能够一定程度减轻用户的交互量。Kim 和 Neumann[134] 提出了一种多分辨率的头发几何建模方法，利用了头发的几何特征往往是多尺度的这一特点。在他们的方法中，用户首先创建少量的 “高层” 发丝，再自顶向下地将每根高层发丝细分为多根 “低层” 发丝。我们的方法可以看作是这一过程的逆过程，即从仅包含大量底层发丝的头发几何模型中经过区域分析自动提取高层发丝并构造一个多层的发簇结构。

除了手工创建以外，近年来对基于图像的头发几何重建的研究有了很大发展[130,132,133]。Paris 等人在 2004 年提出了一种方法，在固定相机视角与变化光照角度的条件下获得多张真实头发的图像，通过分析图像中每个像素对不同朝向滤波核的响应以及光照信息重建出描述头发在空间中各点切向量的三维向量场，并从这个向量场中产生几何发丝。Wei 等人[133] 以及 Paris 等人[130] 对这一方法做了进一步改进，其中 Wei 等人的方法允许使用普通摄像机在自然光照条件下从不同角度对真实头发拍摄多张图像并恢复头发的几何与颜色信息，Paris 等人的方

法使用更加复杂的包含多个相机与光源的采集设备，取得了质量非常高的结果。Ishikawa 等人[140] 以及 Yamaguchi 等人[141] 分别尝试采集动态的头发，但目前这些方法仅能支持相当简单的发型与有限的头发运动。

另外一个与我们工作相关的领域是基于样本的纹理合成及其推广，特别是将纹理合成应用于曲线的生成[142] 或是其它几何模型的生成[143–147]。但是这些方法均无法应用于基于样本的头发几何建模。

对头发建模相关工作更全面的介绍可以参考 Ward 等人的文献综述[148]。

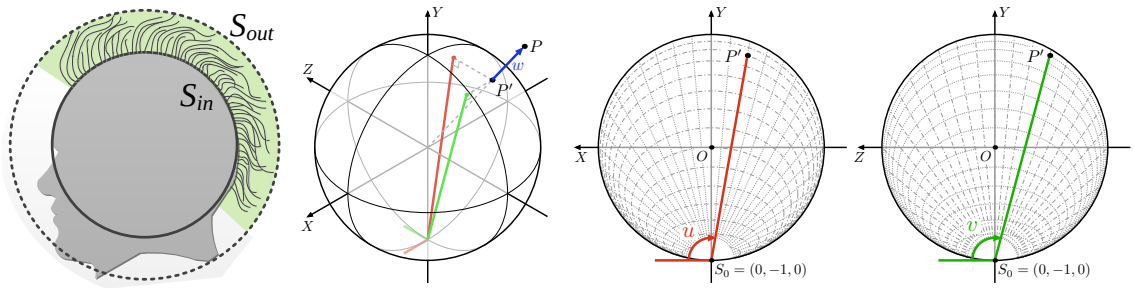
## 5.2　基于区域结构的头发几何分析

我们采用的输入头发几何模型中，每根发丝用一条折线（polyline）表示。一个完整头发模型（例如 Paris 等人[130] 的采集结果）中的折线数量通常为几千至十几万不等，每根折线包含几十至上百个顶点。真实头发中的发根均分布于一个二维曲面（头皮）上，发根相邻的发丝倾向于具有相似的几何形状并且构成发簇（wisp）区域。为了分析一个发型中的发簇区域结构，首先我们将头发占据的三维空间进行参数化（5.2.1节），这样的参数化为每一根发丝定义了局部坐标系以及其发根在头皮表面的二维参数坐标。在参数空间中，我们可以定义任意两根发丝的几何相似性度量，我们进而提出一种聚类算法自动根据相邻发丝的相似性将输入的头发几何模型划分为多个发簇区域（5.2.2节）。对于每个发簇区域内部的发丝，我们计算出一根具有代表性的特征发丝（feature strand）并将其几何特征转化为高维向量，称为特征向量（feature vector）。利用头皮表面的参数化，我们可以将一个完整发型的几何特征描述转化为一副二维的特征图（feature map），从而可以利用已有的图像纹理合成算法进行头发的几何合成（5.2.3）。

### 5.2.1　头发参数化空间的定义

人的头皮是近似于球面的弯曲表面，而头发则分布于头皮表面上方的三维空间。不失一般性，我们可以假设头皮表面是一个单位球面 $S_{\text{in}}$，其球心位于世界坐标系的原点，以人头为参照，世界坐标系的 $y$ 轴指向上方，$z$ 轴指向前方，且头发分布于 $S_{\text{in}}$ 与另一同心球面 $S_{\text{out}}$ 之间的包围区域中，如图5.1(a) 所示。对于该空间中的任意一点 $P=(x,y,z)$，我们定义其参数化坐标 $(u,v,w)$ 如下：

$$
\begin{aligned}
u &= \arccos \frac{\hat{x}}{\sqrt{\hat{x}^2+(\hat{y}+1)^2}} \\
v &= \arccos \frac{\hat{z}}{\sqrt{\hat{z}^2+(\hat{y}+1)^2}}
\end{aligned}
$$




(a)　(b)　(c)　(d)

图 5.1　头发所在空间的三维参数化。给定世界坐标系中的任意点 $P(x,y,z)$ 及其在单位球面上的球面投影 $P'$，$P$ 点的参数化坐标 $(u,v,w)$ 具有直观含义：$u$ 和 $v$ 为 (c)(d) 中的角度，其值域为 $(0,\pi)$，而 $w$ 为 (b) 中所示 $P$ 与其投影的距离。

$$w = \sqrt{x^2+y^2+z^2} - d(\hat{x},\hat{y},\hat{z}) \tag{5-1}$$

其中 $(\hat{x},\hat{y},\hat{z})$ 是 $(x,y,z)$ 在 $S_{\text{in}}$ 上的球面投影，$d(\hat{x},\hat{y},\hat{z})$ 是沿 $(\hat{x},\hat{y},\hat{z})$ 方向从球心到头皮表面的距离。

如图5.1所示，参数 $u$ 可以直观地理解为 $x$ 轴与 $S_0P'$ 在 $XY$ 平面上投影的夹角，其中 $S_0=(0,-1,0)$；类似的，参数 $v$ 可以理解为 $z$ 轴与 $S_0P'$ 在 $YZ$ 平面上投影的夹角；参数 $w$ 则为 $P$ 点相对头皮表面的高度。

根据5-1式可以得到由参数坐标到世界坐标的变换：

$$\begin{aligned} x &= \rho h \cot u \\ y &= \rho(h-1) \\ z &= \rho h \cot v \end{aligned} \tag{5-2}$$

其中：

$$\begin{aligned} h &= \frac{2}{\cot^2 u + \cot^2 v + 1} \\ \rho &= w + d(h\cot u, h-1, h\cot v) \end{aligned} \tag{5-3}$$

与许多头发建模方法（如 [131]）采用的经纬度坐标参数化方法相比，我们的参数化仅在 $(0,-1,0)$ 有一个奇点，从而确保头发分布集中的区域参数连续且几何形变较小，如图5.2所示。

基于这样的参数化空间，我们可以定义头皮表面任意一点上由法向量（normal）、切向量（tangent）和副法向量（binormal）构成的局部坐标系。为了计算一点 $P$ 的切向量和副法向量，我们首先计算由 $y$ 轴正方向至 $P$ 点法向量的旋转变换矩阵 $\mathbf{R}$。接下来，我们将旋转变换 $\mathbf{R}$ 分别应用于 $x$ 轴的正方向与 $z$ 轴的正方

图 5.2　不同参数化方法比较。从左至右依次是：未参数化的头皮表面，传统的基于经纬度的参数化结果，我们的参数化结果。注意经纬度参数化中明显的奇点和不均匀性。

向即得到 $P$ 点的切向量和副法向量。头皮表面的局部坐标系定义可以被进一步扩展至空间中任意点，仅需规定空间中任意点 $P$ 的局部坐标系的坐标轴与 $P$ 在头皮表面上的投影 $P'$ 点局部坐标系的坐标轴平行即可。

### 5.2.2　发簇区域结构的自动划分

受 Cohen-Steiner 等人对三角形网格的聚类研究工作[124] 的启发，我们提出一种基于变分思想的发簇区域划分算法。对于输入的含有 $n$ 根发丝的头发几何模型，该算法自动根据几何相似性将这 $n$ 根发丝划分为 $k$ 个发簇，并且为每个发簇计算一根 “特征发丝”（feature strand）。我们称这 $k$ 个发簇构成的集合为一个 “k 分区”。通过将每一次分区得到的特征发丝作为下一层区域划分的输入发丝，我们可以很容易地构建多层的发簇结构。

我们的发簇区域划分思想与 Lloyd 算法[149]（亦即 k-means 聚类）相似，可以确保收敛性。算法交替地重复 “分区” 与 “拟合” 两个步骤。在第一个步骤中我们设计了一种区域增长算法，使得每个发簇区域由最开始的初始种子发丝开始同步地向外增长，直到任意发丝均被某区域覆盖。这种方法同时确保同一区域中的发丝是彼此邻接的。在第二个步骤中，我们根据每个区域中的发丝计算一个最优的特征发丝。

#### 几何相似性度量及能量函数

发簇区域划分的一个重要前提是合理地定义发丝之间的相似性度量。给定任意两根发丝 $\gamma_a$ 和 $\gamma_b$，我们首先分别将其重新均匀采样为 $n_s$ 个顶点。用 $\gamma_a(l)$ 和 $\gamma_b(l)$ 分别表示 $\gamma_a$ 和 $\gamma_b$ 从发根至发梢方向的第 $l$ 个顶点在各自局部坐标系中的坐

标。接下来我们可以定义两根发丝之间的 $\mathcal{L}^2$ 距离：

$$\mathcal{L}^2(\gamma_a, \gamma_b) = \sum_{l=1}^{n_s} \|\gamma_a(l) - \gamma_b(l)\|^2 \tag{5-4}$$

对发丝之间 $\mathcal{L}^2$ 距离的直观解释是将两根发丝进行均匀采样并将局部坐标系对齐后比较各对采样点之间的距离。由于根据5.2.1节中介绍的参数化空间定义的局部坐标系在头皮表面的各方向上变化缓慢、连续，并且在头发分布的区域内没有奇点，因此5-4式可以很大程度避免头皮表面弯曲造成的全局影响，很好地度量相邻发丝局部几何形状的相似性。

记输入的 $n$ 根发丝的集合为 $\mathcal{S}$，区域划分所得的第 $k$ 个发簇区域为 $\mathcal{S}_i$，该区域对应的特征发丝为 $\bar{\gamma}_i$，则我们可以定义一个 k 分区的总体能量函数如下：

$$\mathcal{E}(\mathcal{S}) = \sum_{i=1}^{k} \sum_{j=1}^{|\mathcal{S}_i|} \mathcal{L}^2(\gamma_j^i, \bar{\gamma}_i), \tag{5-5}$$

其中 $\gamma_j^i$ 表示第 $i$ 个发簇区域中的第 $j$ 根发丝。

**分区阶段**

这一阶段的目的是对于给定的 $k$ 根特征发丝，找到一个 k 分区，使得5-5式中的能量函数最小化。在区域划分开始之前，我们首先将输入的全部发丝的发根顶点进行 Delaunay 三角化以建立发丝的邻接关系。在第一次迭代时，我们从输入的发丝中随机选取 $k$ 根作为初始的特征发丝，并且这 $k$ 根发丝分别作为初始状态下 $k$ 个发簇区域中唯一的发丝。所有其它发丝的区域归属则初始化为空。

具体的分区过程如下：对于第 $i$ 个发簇区域，我们首先从区域内部找到与特征发丝最相似的发丝 $\gamma_1^i$ 作为“种子”（seed strand）：

$$\gamma_1^i = \arg\min_{\gamma_j \in \mathcal{S}_i} \mathcal{L}^2(\gamma_j, \bar{\gamma}_i) \tag{5-6}$$

为了确保我们仅将与特征发丝相似的发丝加入发簇区域，对于每个种子发丝 $\gamma_1^i$，我们将所有与其直接相邻的发丝 $\gamma_j$ 加入一个全局的优先队列（priority queue），队列中元素的优先级反比于 $\mathcal{L}^2(\gamma_j, \bar{\gamma}_i)$，并且每个元素附加有一个标签用于记录 $\gamma_j$ 尝试加入的发簇区域标记 $i$。

区域增长的过程随即不断地从优先队列中弹出具有最高优先级的发丝。对每一个弹出的发丝 $\gamma$，我们首先检查其区域归属：若它已经被划分入某个区域，则跳过该发丝，转而弹出队列中的下一个发丝；否则，我们将 $\gamma$ 加入其附属标签指明的发簇区域，并将所有与 $\gamma$ 直接相邻且尚未被划分区域的发丝加入全局优先队

列。当优先队列为空时，每一根发丝都被划分入某个区域，该次迭代的分区阶段结束。

这一过程确保每根发丝都属于且仅属于一个发簇区域，且每个区域内部的发丝是邻接的。对于数量为 $n$ 的输入发丝，这一过程的时间复杂度为 $O(n\log n)$。

#### 拟合阶段

这一阶段的目的是对于一个给定的 k 分区，为其中每个发簇区域计算一个最优的特征发丝作为下一次迭代中分区阶段的输入。根据5-4式的 $\mathcal{L}^2$ 距离定义可知，最小化5-5式中能量函数的特征发丝即为各区域中发丝的平均，亦即：

$$\bar{\gamma}_i(l) = \frac{1}{|\mathcal{S}_i|}\sum_{j=1}^{|\mathcal{S}_i|}\gamma_j^i(l), \quad l \in \{1,2,\ldots,n_s\} \tag{5-7}$$

#### 自适应的区域划分

基于 Lloyd 算法的发簇区域划分要求指定最终的区域数 $k$，然而对于任意一个输入头发几何模型，用户通常并不容易决定合适的 $k$。为此我们对上文介绍的算法进行了扩展，允许用户指定一个错误阈值 $\epsilon$，算法将自动根据 $\epsilon$ 决定合适的区域数。自适应区域划分算法的基本思想是迭代地计算一个 $k$ 分区，从而使得对于输入 $\mathcal{S}$ 中的任意发丝 $\gamma_j$ 及其所属区域的特征发丝 $\bar{\gamma}_i$，均满足如下关系：

$$\mathcal{L}^2(\gamma_j, \bar{\gamma}_i) < \epsilon \tag{5-8}$$

算法首先根据输入头发几何模型 $\mathcal{S}$ 的发丝数选定一较小的发簇区域数 $k_0$，并对 $\mathcal{S}$ 进行固定 $k$ 的区域划分。在区域划分结束后，对于每个发簇区域，若区域内部存在某发丝与对应特征发丝的 $\mathcal{L}^2$ 距离大于 $\epsilon$，则选出该区域中与特征发丝 $\mathcal{L}^2$ 距离最大的发丝作为一个新发簇区域的种子发丝，亦即该区域分裂为两个区域。之后，所有新生成的种子发丝与原始的种子发丝一起作为输入对 $\mathcal{S}$ 重新进行区域划分。

当创建一个完整的多层发簇区域结构时，我们仅通过自适应的区域划分决定最底层的区域数。上层的区域数设定为其下层区域数的四分之一。

图5.3展示了能量 $\mathcal{E}(\mathcal{S})$ 随发簇区域划分算法的迭代次数变化的趋势。

### 5.2.3　特征向量与特征图

通过发簇区域划分我们由输入的头发几何模型构建出多层的区域结构，其中不同的层次描述了头发在不同尺度上的几何特征。在一个由 100000 根发丝的输

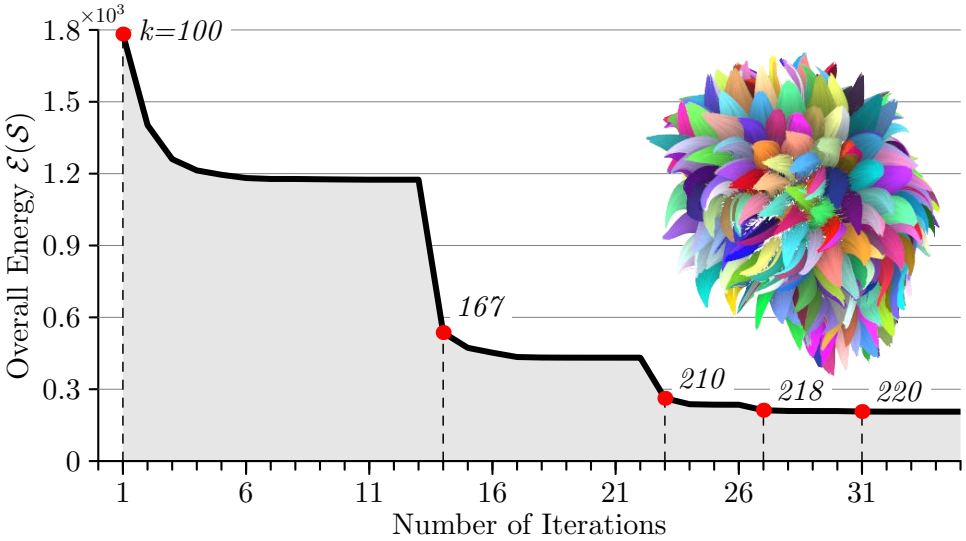



图 5.3　自适应区域划分算法将一个包含 40000 根发丝的头发几何模型自动划分为 220 个发簇区域。图中曲线显示了区域划分过程的不同迭代阶段分区能量 $\mathcal{E}(\mathcal{S})$ 的变化趋势；红色圆点表示区域分裂及新区域的产生；红点旁的斜体数字表示对应的区域数。

入发型构造的 3 层发簇区域结构中，最上层的特征发丝大约在 100 至 500 根，通常仅包含输入发型全局尺度上的特征。由于每根发丝均有 $n_s$ 个顶点，通过将这些顶点在发根局部坐标系中的坐标连接起来可以得到一个 $3n_s$ 维向量。我们进一步对所有特征发丝的向量做主元分析（Principal Components Analysis）并将其投影至 $n_f$ 维子空间，则每根特征发丝可以表示为一个 $n_f$ 维向量，我们称之为该发丝的 “特征向量”（feature vector）。理论上若 $n_f = 3n_s$，特征向量可以完全描述发丝的几何形状。实际当中，我们令 $n_f = 16$，在我们测试过的头发几何模型中，此时因降维产生的平均误差小于 5%。

考虑到发根分布在头皮表面上，我们可以利用5.2.1节介绍的参数化空间由所有发丝特征向量生成一幅二维 “特征图”（feature map）。由于发根的分布是不均匀的，我们在头皮表面参数化空间中定义一规则网格，并在网格的各节点上对附近的特征向量进行插值，如图5.4所示。

## 5.3　头发的几何建模

在基于区域结构的头发几何分析阶段，输入的原始头发几何模型被表示为多层发簇区域结构与特征图，利用这些结构我们可以合成具有与输入相似几何特征的新的头发几何模型。这种基于样本的建模方式有许多潜在应用，例如将已有的高质量头发模型（如根据真实人物的图像重建得到的）应用于一个甚至多个不同的虚拟人物。

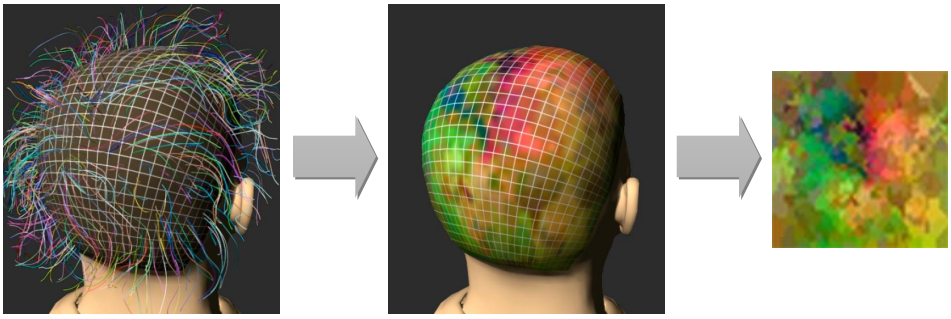


图 5.4　特征向量与特征图。由左至右依次是：特征发丝，特征发丝对应的特征向量，经头皮参数化空间重新采样得到的二维特征图。特征图中的颜色为特征向量的前三维经正规化后映射到 RGB 通道。

本节中我们首先提出一种基于发簇结构的合成算法（5.3.1节）。给定一个多层发簇结构及其特征图作为输入，算法首先利用二维纹理合成技术产生一个新的特征图，再自顶向下地生成一个新的多层发簇结构，最终合成新的头发几何模型。

头发的几何合成必须考虑发丝的空间一致性，这里“空间一致性”具有双重意义：首先，发根相邻的发丝倾向于具有相似的几何形状并形成发簇结构。其次，每根发丝都可以看作三维空间中的一条弯曲的轨迹，经过空间中同一点附近的轨迹倾向于具有相似的切向量。前文提到的基于发簇结构的合成算法仅考虑第一种空间一致性。为了确保第二种空间一致性，我们引入一种基于三维向量场的头发切向量优化算法（5.3.2节）。

### 5.3.1　基于发簇结构的头发合成

在对输入头发几何模型的分析过程中，我们自底向上地构建了多层发簇结构，并且将最顶层的特征发丝表示为特征图的形式。在合成新的输出头发几何模型过程中，我们则以自顶向下的方式进行，首先生成输出头发几何模型的特征图，进而逐层向下生成完整的多层发簇结构，整个过程如图5.5所示。

由于特征向量与特征图的定义可以将头发的三维几何转化为二维描述，并且特征向量的分布一定程度上符合马尔可夫随机场（Markov Random Field）的特征，我们可以使用基于样本的图像纹理合成方法[14] 由输入头发几何模型的特征图生成一个新的特征图。在目前的实现中我们采用 Kwatra 等人提出的纹理优化[19] 算法，该算法在合成质量与计算时间之间可以取得较好平衡。

在生成新的特征图后，我们可以从其中的特征向量重建新发型中最顶层的特征发丝：首先我们在用户指定的头皮表面区域内随机产生采样点作为特征发丝的初始发根位置，采样点的数目根据对应输入发型发根的面密度自动决定。这些随

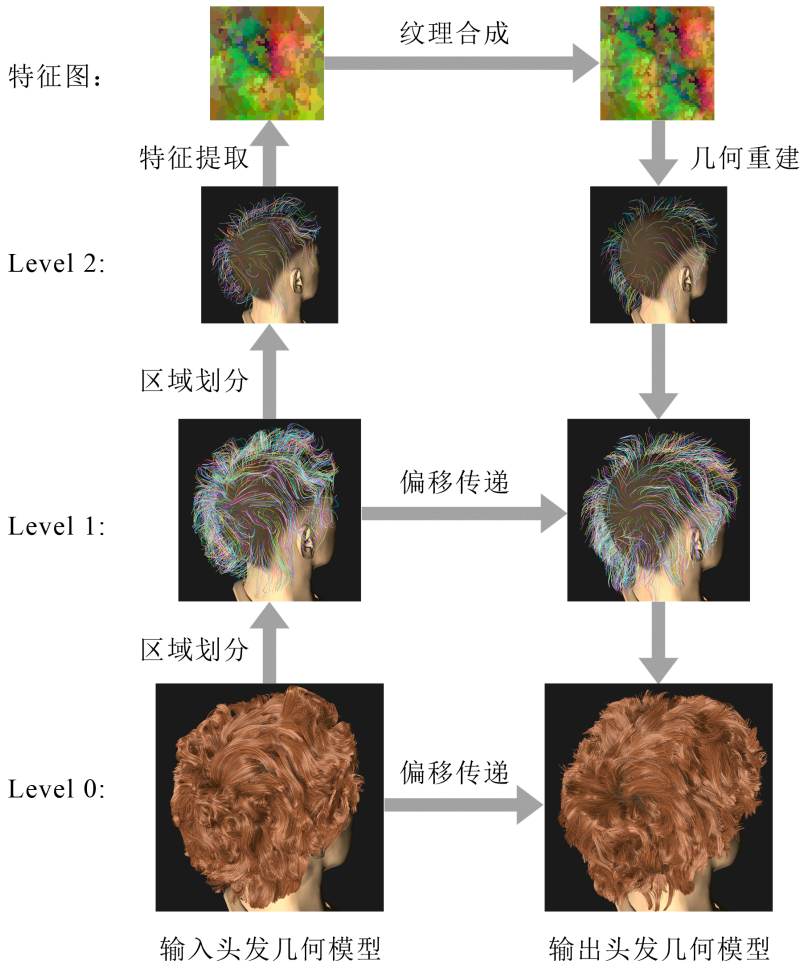



图 5.5　基于样本的头发几何分析与合成过程。

机点可以进一步通过 relaxation 方法[117] 使其分布更加均匀。接下来，我们根据每个发根在头皮表面的参数坐标 $(u,v)$ 从特征图中找到对应的特征向量，由于特征向量中的元素事实上是对发丝采样点的局部坐标进行主元分析的系数，我们将其逆映射至发根局部坐标系空间中即可得到发丝 $n_s$ 个顶点的几何坐标。

至此我们得到了输出头发几何模型的最顶层特征发丝，下一步我们希望逐层向下地构建完整的多层发簇区域结构。发簇的形状以及内部发丝的细微几何变化是反映一个发型独特外观的重要因素，我们希望输出的头发几何模型能够继承输入发型中这种独特的发簇特征。为此，在生成更细一层发簇结构中的发丝时，我们采用一种 “逐发簇” 的策略：假设已知第 $(m+1)$ 层的特征发丝，现欲生成第 $m$ 层的特征发丝。首先我们采取和生成顶层发丝相同的方法产生第 $m$ 层发丝的发根位置，对于每个发根 $\gamma_m^p$，找到第 $(m+1)$ 层中距离最近的特征发丝 $\gamma_{m+1}^q$，将 $\gamma_m^p$ 加入以 $\gamma_{m+1}^q$ 为特征发丝的发簇区域。接下来，对于第 $m$ 层的每个发簇区域，我们从输入发簇结构的第 $m$ 层找到一个相应的发簇区域，使得这两个发簇区域的特征

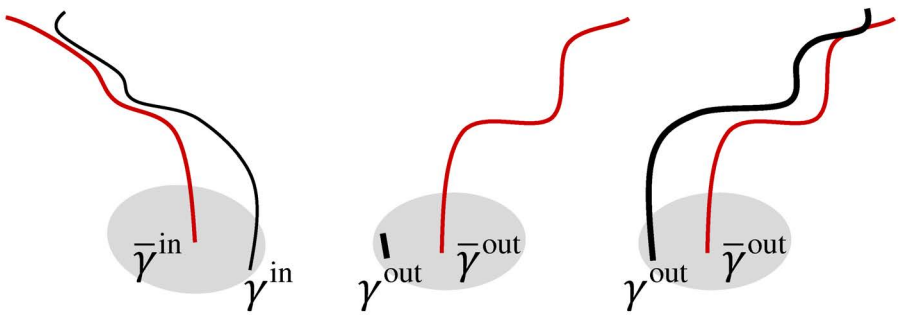



图 5.6　发丝的相对偏移传递。对于一个输出发簇及一个相匹配的输入发簇，我们希望输出发簇中的发丝 $\gamma^{\text{out}}$ 与特征发丝 $\bar{\gamma}^{\text{out}}$ 的 “相对偏移” 和输入发簇中的发丝 $\gamma^{\text{in}}$ 与特征发丝 $\bar{\gamma}^{\text{in}}$ 的 “相对偏移” 类似。

发丝在5-4式定义的 $\mathcal{L}^2$ 度量下最相似。

对于最相似特征发丝的搜索可以通过经主元分析降维的特征向量加速。这里有两点需要注意：第一，我们为每个特征发丝定义一个新的局部坐标系。新坐标系的计算过程为将5.2.1节中定义的局部坐标系以法向量为轴旋转，直到切向量方向与特征发丝的**平均切向量方向在头皮切平面上的投影**对齐。对任意一根发丝 $\gamma$，我们可以定义其平均切向量如下：

$$\mathbf{v}_g(\gamma) = \frac{\sum_{i=2}^{n_s}(\gamma(i) - \gamma(1))}{\| \sum_{i=2}^{n_s}(\gamma(i) - \gamma(1))\|} \tag{5-9}$$

第二，在计算最近邻搜索使用的特征向量时，我们同时对输入与输出发簇结构中的特征发丝做主元分析。以上两点可以确保在进行特征发丝相似性度量时仅关注发簇的局部形状而不会受到发簇整体朝向不同的影响。

通过相似特征发丝搜索，我们为输出头发几何模型中的每一个发簇区域 $\mathcal{S}^{\text{out}}$ 找到了一个相匹配的输入发簇 $\mathcal{S}^{\text{in}}$。注意此时输出发簇 $\mathcal{S}^{\text{out}}$ 仅包含一组发根及一根 $(m+1)$ 层的特征发丝。为了保持输入发簇内部的细微几何特征，我们希望生成 $\mathcal{S}^{\text{out}}$ 中的发丝并且使得这些输出发丝相对输出特征发丝的偏移与输入发丝相对输入特征发丝的偏移类似，如图5.6所示。下面我们首先定义两根发丝之间的相对偏移，再提出一种偏移传递的方法。

### 发丝偏移（Displacement）的定义

对于任意两根发丝 $\gamma_a$ 和 $\gamma_b$，用 $\gamma_a(i)$ 和 $\gamma_b(i)$ 分别表示它们从发根至发梢的第 $i$ 个顶点的世界坐标，我们可以定义从 $\gamma_a$ 到 $\gamma_b$ 的逐点偏移。为了便于描述，我们将从 $\gamma_a$ 到 $\gamma_b$ 的偏移分解为 $\mathcal{D}_\alpha(\gamma_a, \gamma_b)$ 和 $\mathcal{D}_\beta(\gamma_a, \gamma_b)$ 两项并分别定义。$\mathcal{D}(\gamma_a, \gamma_b, i)$ 表示 $\gamma_a(i)$ 与 $\gamma_b(i)$ 在第 $i$ 个顶点上的偏移。

第一个偏移项 $\mathcal{D}_\alpha(\gamma_a,\gamma_b)$ 描述两根发丝在**发根局部坐标系中的相对偏移**：

$$\mathcal{D}_\alpha(\gamma_a,\gamma_b,i)=\mathbf{M}_{\text{nbt}}^T(\gamma_b)\left[(\gamma_a(i)-\gamma_b(i))-(\gamma_a(1)-\gamma_b(1))\right] \tag{5-10}$$

其中 $\mathbf{M}_{\text{nbt}}^T(\gamma_b)$ 为将世界坐标转换为 $\gamma_b$ 发根处局部坐标的变换矩阵。计算 $\mathcal{D}_\alpha$ 的过程可以直观想象为首先平移 $\gamma_a$ 使其发根与 $\gamma_b$ 的发根重合，再计算 $\gamma_b$ 发根局部坐标系中两根发丝的逐顶点偏移。

第二个偏移项 $\mathcal{D}_\beta(\gamma_a,\gamma_b)$ 描述两根发丝在**各顶点局部坐标系中的绝对偏移**。为此首先需要定义发丝各顶点处的局部坐标系，亦即发丝曲线在该点的切向量、法向量及副法向量。我们使用 Bloomenthal[150] 提出的算法计算这样的局部坐标系。$\mathcal{D}_\beta(\gamma_a,\gamma_b)$ 的定义如下：

$$\mathcal{D}_\beta(\gamma_a,\gamma_b,i)=\mathbf{M}_{\text{nbt}}^T(\gamma_b(i))(\gamma_a(i)-\gamma_b(i)) \tag{5-11}$$

其中 $\mathbf{M}_{\text{nbt}}^T(\gamma_b(i))$ 为将世界坐标转换为 $\gamma_b$ 的第 $i$ 个顶点处局部坐标的变换矩阵。

#### 偏移传递（Displacement Transfer）

偏移传递的过程如下：对于输出发簇 $\mathcal{S}^{\text{out}}$ 中的每根发丝 $\gamma^{\text{out}}$，我们首先计算 $\gamma^{\text{out}}$ 的发根顶点与 $\mathcal{S}^{\text{out}}$ 特征发丝 $\bar{\gamma}^{\text{out}}$ 发根顶点的偏移；接下来从与 $\mathcal{S}^{\text{out}}$ 相匹配的输入发簇 $\mathcal{S}^{\text{in}}$ 中找到一根发丝 $\gamma^{\text{in}}$，使得 $\gamma^{\text{in}}$ 与特征发丝 $\bar{\gamma}^{\text{in}}$ 的发根顶点偏移与 $\gamma^{\text{in}}$ 与 $\bar{\gamma}^{\text{in}}$ 的发根顶点偏移最接近；最后我们可以根据下式计算输出发丝 $\gamma^{\text{out}}$：

$$\gamma^{\text{out}}(i)=\frac{n_s-i}{n_s-1}\gamma_\alpha^{\text{out}}(i)+\frac{i-1}{n_s-1}\gamma_\beta^{\text{out}}(i),\quad i=1,\ldots,n_s \tag{5-12}$$

其中

$$\gamma_\alpha^{\text{out}}(i) = \bar{\gamma}^{\text{out}}(i)+\mathbf{M}_{\text{nbt}}(\bar{\gamma}^{\text{out}})\mathcal{D}_\alpha(\gamma^{\text{in}},\bar{\gamma}^{\text{in}},i)+\gamma^{\text{out}}(1)-\bar{\gamma}^{\text{out}}(1) \tag{5-13}$$

$$\gamma_\beta^{\text{out}}(i) = \bar{\gamma}^{\text{out}}(i)+\mathbf{M}_{\text{nbt}}(\bar{\gamma}^{\text{out}}(i))\mathcal{D}_\beta(\gamma^{\text{in}},\bar{\gamma}^{\text{in}},i) \tag{5-14}$$

这里 $\gamma_\alpha^{\text{out}}(1)$ 确保生成发丝 $\gamma^{\text{out}}$ 的发根顶点在头皮表面上的初始位置，$\gamma_\beta^{\text{out}}(n_s)$ 则确保 $\gamma^{\text{in}}$ 与 $\bar{\gamma}^{\text{in}}$ 发尖顶点的相对偏移在 $\gamma^{\text{out}}$ 与 $\bar{\gamma}^{\text{out}}$ 之间被正确保持。

### 5.3.2　头发切向量场优化

用于描述发丝几何形状的特征向量在空间上被记录在发根所在的头皮参数表面上，然而发丝可以是三维空间中一根长且复杂的弯曲轨迹，事实上发丝对于发型整体外观的影响并非局限于发根所在位置，而是遍布整条空间轨迹。对真实头发的更进一步观察发现，一些发根距离很远的发丝在空间中呈现出一定的一致

性。例如在图5.7所示的情况中，两根发根并不临近的发丝的空间轨迹却呈现连续性。对于一个观察者来说，这两根发丝在视觉上更像是一根更长的发丝。而这种空间一致性很难在二维的特征图中体现出来，造成的后果就是合成的输出头发几何模型在视觉上比输入发型更加杂乱。为了改善这种情况，增强输出发型的空间一致性，我们提出一种优化方法首先根据原始输出头发几何模型计算一个三维的发丝切向量场，再根据这个向量场重新生成发丝并替换原始发丝。


图 5.7　两根发根相距较远但在空间中具有一致性的发丝在视觉上像是一根连续的发丝。

我们在5.2.1节定义的参数化空间中定义一个三维规则网格。令 $(i,j,k)$ 表示网格中的一个位于参数坐标 $(u_i,v_j,w_k)$ 的单元格，$d\mathbf{s}$ 表示任意发丝上任意小的一段，$\mathbf{T}(d\mathbf{s})$ 表示 $d\mathbf{s}$ 的切向量，$C(i,j,k)$ 表示所有穿过单元格 $(i,j,k)$ 的发丝片段的集合。在上述三维规则网格上，我们定义三维发丝切向量场如下：

$$\bar{\mathbf{v}}^t(i,j,k)=\frac{\sum_{d\mathbf{s}\in C(i,j,k)}\omega^t(\mathbf{T}(d\mathbf{s}))\mathbf{T}(d\mathbf{s})}{\|\sum_{d\mathbf{s}\in C(i,j,k)}\omega^t(\mathbf{T}(d\mathbf{s}))\mathbf{T}(d\mathbf{s})\|} \tag{5-15}$$

其中 $\omega^t(\cdot)$ 具有如下定义：

$$\omega^t(\mathbf{v})=0.5(\mathbf{v}\cdot\bar{\mathbf{v}}^{t-1}(i,j,k)+1) \tag{5-16}$$

且有 $\bar{\mathbf{v}}^0(i,j,k)=\mathbf{0}$。没有任何发丝穿过的单元格中的切向量也设为 $\mathbf{0}$。

注意依照5-15式计算切向量场的过程是迭代的，每次迭代权重项 $\omega^t(\cdot)$ 都会更新。这种策略的目的是确保在每个单元格中，一个发丝片段的切向量与上一次迭代中单元格的平均切向量相差越大，它对本次迭代中的平均切向量计算结果影响越小。我们知道，给定任一单元格中的一组不一致的向量估计，其最小二乘意义下的解即是这组向量的平均值。然而最小二乘解很容易受到异常值的干扰。Hampel 等人[151] 指出，迭代重加权最小二乘（iteratively re-weighted least squares）的解可以渐进最优解。在我们的实现中，对5-15式的迭代仅进行两次，我们发现更多迭代次数对最终视觉效果的影响微乎其微。

在得到三维切向量场后，我们可以从中重新生成发丝。这一过程类似基于图像的头发建模[130,132,133] 中广泛采用的发丝生长方法，亦即从发根开始沿着向量场的方向做离散的曲线积分。由于向量场是定义在规则网格上的，对于非网格节点

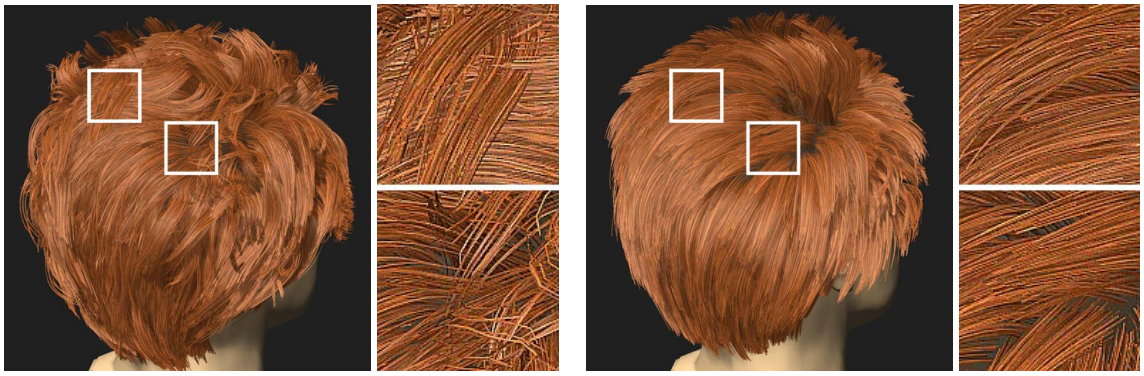


图 5.8　头发切向量场优化。左侧的头发几何模型没有经过优化而右侧的模型经过优化。可以看到经过优化的发丝具有更好的空间一致性与连续性。

的空间位置，我们根据其所处单元格的 8 个网格节点上的向量值使用三线性插值计算切向量。当满足下列两个条件中的任一个时发丝生长停止：发丝达到预定的长度，或是发丝生长到达一个 $\bar{\mathbf{v}} = 0$ 的单元格。

总的来说，头发切向量场的优化可以增强头发的空间一致性和连续性，如图5.8所示。

## 5.4　结果与讨论

为了验证基于区域结构的几何分析与合成的适用性，我们实验了不同种类的输入头发几何模型，包括平直的、卷曲的、杂乱的等，并且这些模型是不同建模方法的结果，包括手工建模与基于图像的重建。以对这些输入模型的分析结果为基础，我们尝试了不同的合成应用，包括生成具有与输入发型类似发簇特征的新发型（如图5.5和图5.12所示），混合多个输入发型的外观特征（图5.13(a)(b)(c)），保持发簇内部结构的交互式编辑（图5.13(d)）等。

在混合多个输入发型的实验中，我们将发型 *A* 的多层发簇结构的最顶层特征发丝替换为另一发型 *B* 最顶层的特征发丝，从而生成具有 *B* 的全局几何特征与 *A* 的局部几何特征的新发型。在交互式编辑的实验中，我们实现了一个基于笔画的“梳子” 工具，允许用户通过在屏幕上画一些曲线直观地改变头发的整体朝向。尽管一些成熟的三维建模软件（如 3D Studio Max、Blender 等）也提供类似的梳子工具，但这些工具在改变头发整体形状的同时也会影响原始发型中独特的局部特征，例如使卷曲的头发变得过于平滑。而我们的梳子工具直接作用于多层发簇结构的最顶层，在最顶层的特征发丝形状发生变化后，底层的发丝会根据原先的相对偏移重新生成，从而保持发簇内部的相对结构。

表 5.1　可调整的参数及其典型值。

| 符号 | 描述 | 典型值 |
|---|---|---|
| $n_h$ | 多层发簇区域结构的层数 | 2 to 3 |
| $k_0$ | 发簇区域划分的初始区域数 | 200 |
| $\epsilon$ | 自适应区域划分的错误阈值 | 0.1 to 0.3 |
| $n_s$ | 发丝的采样顶点数 | 64 |
| $n_f$ | 特征向量（主元分析的子空间）维数 | 16 |
| $\Delta_{2D}$ | 特征图的网格间隔（参数化空间中） | 0.02 |
| $\Delta_{3D}$ | 切向量场的网格间隔（参数化空间中） | 0.02 |
| $\delta$ | 发丝生长的积分步长 | 0.02 |

### 5.4.1　系统实现

表5.1列出了基于发簇区域结构的头发几何分析与合成过程中所用到参数的默认值。其中两个参数需要特别注意。首先，多层区域结构的层数 $n_h$ 决定了对输入头发几何模型中不同尺度特征的区分能力。划分更多的层数倾向于提取输入发型中的全局特征，如发簇的整体分布、朝向。对于比较简单的输入发型，例如图5.12(d) 中的发型，令 $n_h = 2$ 即可取得较好的结果。对于本章中出现的其它结果，我们均令 $n_h = 3$。在我们的实验中，超过三层的区域结构对最终合成结果的影响则十分有限。其次，对于定义三维切向量场网格分辨率 $\Delta_{3D}$ 的选取应与头发几何模型的复杂度以及几何特征的尺度相匹配。对于特征图合成所用的纹理优化算法，我们使用与 [19] 中相同的参数。

头发几何模型的绘制方面，每根发丝对应 Direct3D 中的一条反走样折线。我们使用 Marschner 模型[129] 计算高光项，使用 Kajiya-Kay 模型[152] 模型计算环境光与漫反射项，使用 Yuksel 和 Keyser 提出的 Deep Opacity Maps（DOM）算法[153] 生成头发的阴影效果。我们实验的头发几何模型均可以实时绘制。

### 5.4.2　质量与性能分析

结果显示，利用我们的方法合成的头发几何模型能够继承输入模型中代表性的发簇分布与发簇内部细微变化，同时从整体看又与输入有足够的区分性。对于一个 100000 根发丝的输入模型，分析与合成的总时间消耗通常小于一分钟，其中发簇区域结构划分花费的时间约占 50%。此外，由于构建了多层的发簇区域结构，我们只需要对其中最上层的特征发丝生成特征图（其分辨率通常低于

表 5.2　不同头发几何模型的发丝数量及分析与合成总用时。

| 头发几何模型 | 输入发丝数 | 总用时（秒） |
|---|---|---|
| Figure 5.5 | 72859 | 43 |
| Figure 5.12(a) | 84587 | 51 |
| Figure 5.12(b) | 69661 | 46 |
| Figure 5.12(c) | 10000 | 11 |
| Figure 5.12(d) | 39695 | 17 |

图 5.9　直接应用图像纹理合成技术生成的头发几何模型（左）与用我们的方法生成的模型（右）的对比。

$100 \times 100$）并应用纹理合成，因此可以极大地减少总体的时间花费。本章中出现结果的分析与合成时间参见表5.2。

### 5.4.3　与其它方法的比较

尽管在本文写作之际据我们了解并没有其它与本章提出方法类似的基于样本的头发建模方法，作为比较，我们尝试了另外两种可能的策略，分别是基于二维与三维的纹理合成算法。

第一种可能的策略是直接根据输入的头发几何模型生成特征图并利用图像纹理合成技术产生输出发型的特征图，再由该特征图重建输出头发几何。这种方法有两个主要问题：首先，由于没有利用头发的发簇区域结构以及空间一致性，这种方法难以保持输入发型中具有代表性的发簇特征；其次，由于原始输入发型中的发丝数量较大，为了准确描述对应的特征向量，需要高分辨率的特征图，从而使得纹理合成所需要的时间大幅提升。图5.9比较了使用这种方法生成的头发几何模型与我们的方法生成的模型。

第二种可能的策略是将输入的头发几何模型直接表示为一个三维切向量场。离散采样的向量场可以看作为三维纹理，因此可以考虑利用三维纹理合成。然而

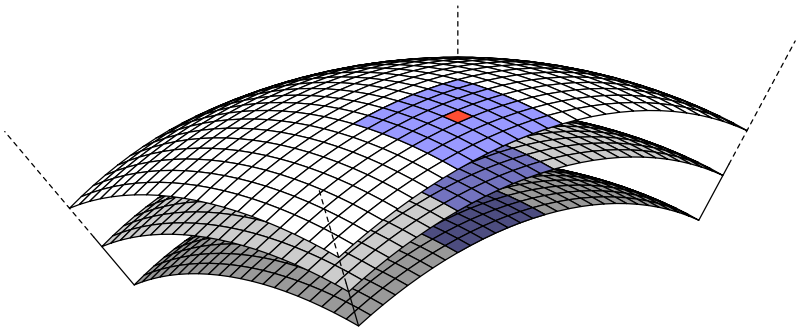


图 5.10　利用纹理合成方法直接合成三维切向量场的过程。合成从靠近头皮的层向远离头皮的层顺序进行，每一体素的邻域是由当前层及之前已合成的层中附近体素构成的三维邻域。

图 5.11　直接应用三维纹理合成技术生成的头发几何模型（左）与用我们的方法生成的模型（右）的对比。

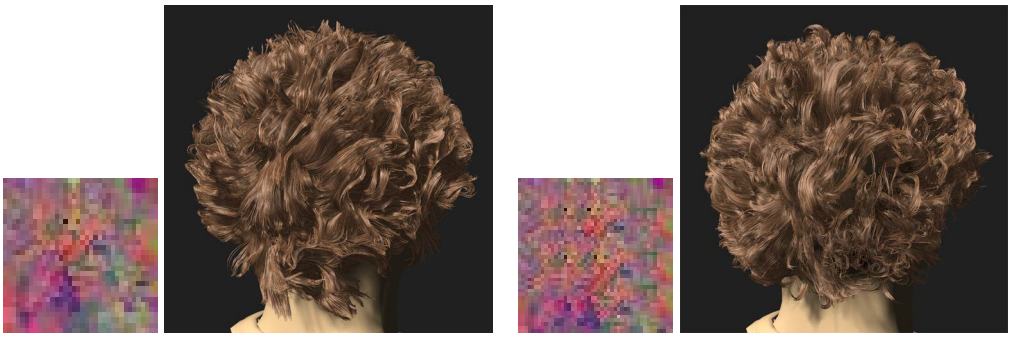
由于在一个头发几何模型中必须确保所有的发根位于头皮表面，因此通常的三维纹理合成技术[36,38] 无法直接适用。我们为此对二维的纹理优化算法[19] 进行扩展，将三维向量场看作一系列叠在一起的曲面，从头皮曲面向远离头皮的曲面逐层合成，并且在进行最近邻查找时使用三维邻域，如图5.10所示。然而，这种方法生成的头发几何模型同样无法保持输入中的发簇结构特征，如图5.11所示，并且整个合成过程相当费时（几十至上百分钟）。

## 5.5　本章小结

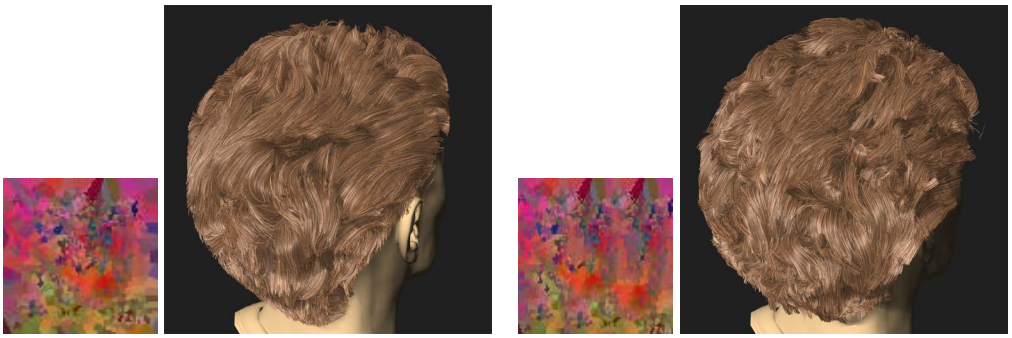
我们提出了一种基于区域结构的头发几何建模方法，通过将一个由大量空间曲线构成的头发几何模型划分为多层的发簇区域，该方法得以提取原始发型中不同尺度的结构特征并生成特征图；对已有发型结构特征的分析使得我们可以用基于样本的方式创建新的头发几何模型，例如生成与输入发型具有统计学意义相似性的输出发型，混合多个输入发型，保持细节的交互式头发编辑等。这是第一个

对已有头发几何模型进行结构分析与合成的工作。实验结果表明，我们的方法可以根据多种不同发型输入高效地生成高质量的新头发几何模型。

在头发几何建模领域的成功应用表明基于区域结构的体数据表达不仅适用于传统意义的体数据，通过一定的扩展更可以应用于具有体数据特征的更一般物体表示，显示出基于区域结构的表达作为一种更通用数据结构与建模方式的潜力。



(a) Puffy



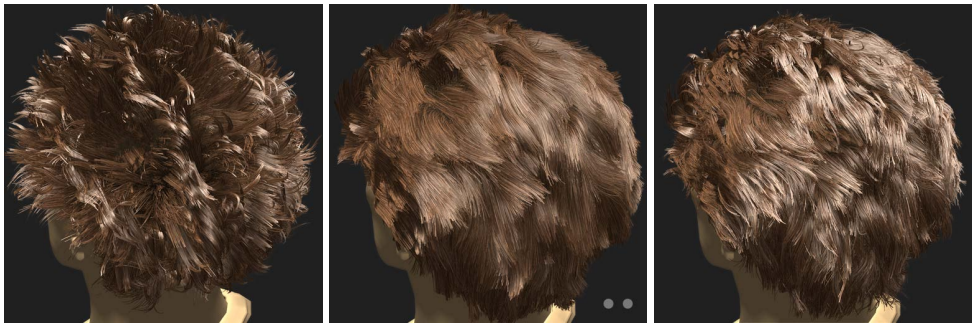
(b) Wavy

(c) Curly

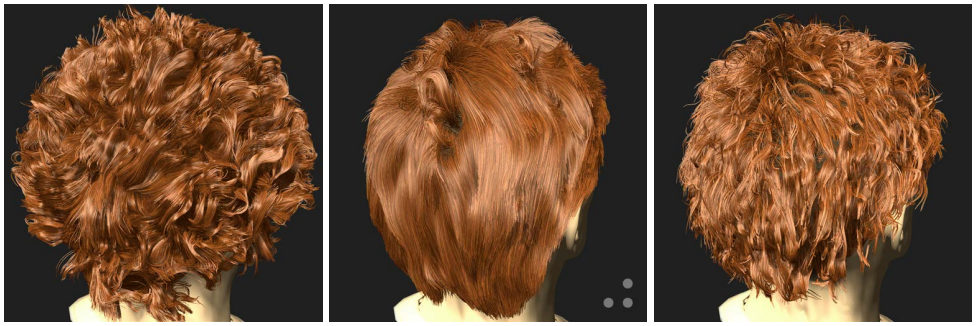
(d) Spiky

图 5.12　使用基于样本方法创建的不同类型发型结果。每一行中左侧为输入的原始头发几何模型及经发簇区域分析提取的特征图，右侧为生成的特征图以及新的头发几何模型。其中 (a) 和 (b) 的输入发型是由基于图像的方法创建的，(c) 和 (d) 的输入发型是手工方法创建的。

(a) curly + long straight



(b) short spikes + wavy



(c) puffy + straight

(d) combing

图 5.13　(a-c) 通过偏移传递生成同时具有多个输入发型特征的新发型，其中左图和中图为输入，右图为输出头发几何模型。(d) 利用我们的交互式梳子工具，在修改 (a-c) 中有圆点标记的头发几何模型的整体形状同时保持了局部的发簇特征。

# 第 6 章　总结与展望

## 6.1　本文工作总结

体模型作为图形学中一类重要的三维物体形式，尽管理论上具有比表面模型更强大的表达能力，在实际中的应用面却相当有限，原因很大程度上可以归结于传统体数据表达在表达能力、建模方式、存储空间以及绘制效率等方面的不足。特别是以离散采样或过程噪声为主的体数据表达并不能描述真实物体内部多尺度的复杂结构特征，这种 “无结构” 的本质不仅限制了它们的表达能力，更使得复杂对象的建模难以下手以及存储低效，最终导致常见的体模型多限于小规模、单调的、纹理性质的内容，与二维内容（如图像、表面模型）的丰富多样形成鲜明对比。

针对这些不足，本文提出了全新的基于区域结构的体数据表达。这一表达很好地利用了物体内部的区域特征，精确高效地描述了物体内部不同尺度的丰富特征与细节，极大地扩展了体模型的表达能力，并使得描述、创建、紧凑存储以及实时绘制具有高度复杂内部结构与丰富细节特征的体数据成为可能。

- 我们首先提出了基于符号距离函数的空间剖分数据结构，将体模型划分为多尺度、任意形状的多个区域。该表达有效将物体内部的特征分为区域之间的不连续跳变以及区域内部的连续变化。前者通过距离函数表示的隐式表面以分辨率无关的方式描述，后者则通过径向基函数和过程噪声共同紧凑表达。通过定义嵌套和实例化，一个体模型中可以同时包含跨越多个尺度层次的丰富细节，并实现了表达能力与存储开销的平衡。
- 论文在第3章讨论了区域结构表达支持的多样建模方式。我们所提出的建模语言与拟合算法改变了传统体模型创建方式单一、难以编辑的局面。通过专门设计的体对象标记语言以及交互式工具，用户可以快速创建高质量体模型。得益于尺度层次和区域的划分，用我们方法创建的体模型其规模、结构复杂性以及细节均远远超过以前方法所能达到的程度。而拟合算法通过符号距离函数提取和径向基函数拟合，可以自动将其它形式的体模型转换为区域结构的表达以便进一步编辑。
- 多尺度以及分辨率无关的特点使得区域结构体模型可以在绘制时任意放大并仍然保持细节特征的清晰。根据符号距离函数的性质，我们在第4章提出

并讨论了高质量、高效率的绘制和反走样算法。在空间索引结构的辅助下，区域结构体模型支持高效的随机访问，并且由于不需预计算，可以在保证实时绘制的前提下允许体模型表面动态变化，例如切割、偏移等。同时通过距离函数量化、区域重标记、区域符号对等数据结构的优化，区域结构体模型具有低廉的存储开销，具有相当复杂程度的体模型仅占用 10 MB 左右显存空间。

- 此外，我们尝试将区域结构的思想应用在头发的几何分析与建模上，提出了第一个针对头发模型的几何特征分析与合成方法。实验结果证明，对头发多尺度区域结构的分析可以用于高效地生成具有与输入样本相似几何特征的新发型，为头发的几何建模提供了一种新的选择。

## 6.2　未来工作展望

我们相信，以物体内部的结构信息为基础将是未来体数据表示与建模的发展趋势，最近的一些相关工作[6] 进一步佐证了这一点。如何更好描述不同类型体结构，同时保持建模方便、存储紧凑、绘制高效等特性值得长期深入研究。

体建模的一个重要难点是三维体数据难以完整地被人眼观察（同一时刻仅能看到体模型的一个截面)。基于结构的方法一方面帮助人们创造出更加复杂的体模型，同时更高的复杂度也可能使人更难以把握体模型中的丰富信息。解决这一问题需要从多个方面进行努力，例如寻找更加清晰的体数据呈现方式，设计更加直观的人机交互方式等。

本文提出的体数据表达假设物体内部的高频特征形成闭合的区域边界，且区域内部材质属性的变化可以分解为低频部分与高频噪声两部分，两者分别用径向基函数和过程噪声描述。虽然绝大多数真实物体符合这样的假设，实际建模中仍可能存在例外，如区域内部材质属性无法容易地分解为径向基函数和过程噪声的和。尝试支持表达力更强且更易于用户控制的区域内部材质属性函数实现（例如稀疏 Gabor 卷积噪声[4]）将是一个值得继续研究的方向。

基于区域结构的表达在头发几何分析与建模上的成功应用使我们意识到，这种表达的能力应该并不局限于传统意义上的体数据，而可以成为一种更通用的数据结构。在将来我们准备在这一方向上做进一步的研究。

最后，我们认为体模型作为一种三维物体的表达形式在今天仍然具有巨大潜力。随着新的数据表达与建模方式的提出，更加复杂多变的体模型会被更容易地创建出来，相信这将使得体模型在图形学中得到更加广泛的应用。

# 参考文献


[1] Dix A, Finlay J E, Abowd G D, et al. Human-Computer Interaction. 3rd ed., Prentice Hall, December, 2003.

[2] NVIDIA GeForce GTX 560 Ti Specifications. http://www.nvidia.com/object/product-geforce-gtx-560ti-us.html.

[3] Cutler B, Dorsey J, McMillan L, et al. A procedural approach to authoring solid models. ACM Transactions on Graphics, 2002, 21(3):302–311.

[4] Lagae A, Lefebvre S, Drettakis G, et al. Procedural noise using sparse Gabor convolution. ACM Trans. Graph., 2009, 28(3):Article 54.

[5] Lagae A, Lefebvre S, Dutré P. Improving Gabor noise. IEEE Transactions on Visualization and Computre Graphics, 2010..

[6] Takayama K, Sorkine O, Nealen A, et al. Volumetric modeling with diffusion surfaces. ACM Transactions on Graphics, 2010, 29(5).

[7] NVIDIA. NVIDIA CUDA C Programming Guide, 3.1.1 ed., July, 2010.

[8] Sigg C, Hadwiger M. Fast third-order texture filtering. Proceedings of GPU Gems 2, chapter 20. 2005:.

[9] Ruijters D, Thévenaz P. GPU prefilter for accurate cubic B-spline interpolation. The Computer Journal, 2010..

[10] Arridge S R, Schotland J C. Optical tomography: forward and inverse problems. Inverse Problems, 2009, 25(12):123010.

[11] Flisch A. Industrail computed tomography in reverse engineering applications. Proceedings of International Symposium on Computerized Tomography for Industrial Applications and Image Processing in Radiology, 1999.

[12] Industrial CT scanning. http://en.wikipedia.org/wiki/Industrial_CT_Scanning.

[13] Lorensen W E, Cline H E. Marching cubes: a high resolution 3D surface construction algorithm. Computer Graphics, 1987, 21(4).

[14] Wei L Y, Lefebvre S, Kwatra V, et al. State of the art in example-based texture synthesis. Proceedings of Eurographics, State of the Art Report, EG-STAR. Eurographics Association, 2009.

[15] Efros A, Leung T. Texture synthesis by non-parametric sampling. Proceedings of ICCV, 1999. 1033–1038.

[16] Wei L Y, Levoy M. Fast texture synthesis using tree-structured vector quantization. Proceedings of ACM SIGGRAPH, New York, NY, USA: ACM Press/Addison-Wesley Publishing, 2000. 479–488.

[17] Efros A, Freeman W. Image quilting for texture synthesis and transfer. Proceedings of ACM SIGGRAPH, 2001. 341–346.

[18] Liang L, Liu C, Yu Y, et al. Real-time texture synthesis using patch-based sampling. ACM Transactions on Graphics, 2001, 20(3):127–150.

[19] Kwatra V, Essa I, Bobick A, et al. Texture optimization for example-based synthesis. ACM Transactions on Graphics, 2005, 24(3):795–802.

[20] Ashikhmin M. Synthesizing natural textures. Proceedings of ACM Symposium on Interactive 3D Graphics, 2001. 217–226.

[21] Tong X, Zhang J, Liu L, et al. Synthesis of bidirectional texture functions on arbitrary surfaces. Proceedings of ACM SIGGRAPH, 2002. 665–672.

[22] Kwatra V, Schodl A, Essa I, et al. Graphcut textures: image and video synthesis using graph cuts. ACM Transactions on Graphics, 2003, 22(3):277–286.

[23] Han J, Zhou K, Wei L Y, et al. Fast example-based surface texture synthesis via discrete optimization. The Visual Computer (Pacific Graphics 2006), 2006..

[24] Wei L Y, Levoy M. Order-independent texture synthesis. Technical report, Stanford University, 2003.

[25] Lefebvre S, Hoppe H. Parallel controllable texture synthesis. ACM Transactions on Graphics, 2005, 24(3):777–786.

[26] Lefebvre S, Hoppe H. Appearance-space texture synthesis. ACM Transactions on Graphics, 2006, 25(3):541–548.

[27] Risser E, Han C, Dahyot R, et al. Synthesizing Structured Image Hybrids. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010), 2010, 29(4):85:1–85:6.

[28] Han C, Risser E, Ramamoorthi R, et al. Multiscale texture synthesis. ACM Transactions on Graphics, 2008, 27(3):Article 51.

[29] Ghazanfarpour D, Dischler J M. Spectral analysis for automatic 3-D texture generation. Computers and Graphics, 1995, 19(3):413–422.

[30] Ghazanfarpour D, Dischler J M. Generation of 3D texture using multiple 2D model analysis. Computer Graphics Forum, 1996, 15(3):311–323.

[31] Heeger D, Bergen J. Pyramid-based texture analysis/synthesis. Proceedings of ACM SIGGRAPH, 1995. 229–238.

[32] Wei L Y, Levoy M. Texture synthesis over arbitrary manifold surfaces. Proceedings of ACM SIGGRAPH, 2001. 355–360.

[33] Wei L Y. Texture Synthesis by Fixed Neighborhood Searching[D]. Stanford University, 2002.

[34] Qin X, Yang Y H. Auro 3D textures. IEEE Transactions on Visualization and Computer Graphics, 2007, 13(2):379–389.

[35] Jagnow R, Dorsey J, Rushmeier H. Stereological techniques for solid textures. ACM Transactions on Graphics, 2004, 23(3):329–335.

[36] Kopf J, Fu C W, Cohen-Or D, et al. Solid texture synthesis from 2D exemplars. ACM Trans. Graph., 2007, 26(3):Article 2.

[37] Chen J, Wang B. High quality solid texture synthesis using position and index histogram matching. The Visual Computer, 2010, 26(4).

[38] Dong Y, Lefebvre S, Tong X, et al. Lazy solid texture synthesis. Computer Graphics Forum, 2008, 27(4):1165–1174.

[39] Xu K, Cohen-Or D, Ju T, et al. Feature-aligned shape texturing. Proceedings of ACM SIGGRAPH Asia, 2009. 1–7.

[40] Zhang G X, Du S P, Lai Y K, et al. Sketch guided solid texturing. Graphical Models, 2011..

[41] Pietroni N, Cignoni P, Otaduy M A, et al. Solid-texture synthesis: a survey. IEEE Computer Graphics and Applications, 2010, 30(4):74–89.

[42] Owada S, Harada T, Holzer P, et al. Volume painter: geometry-guided volume modeling by sketching on the cross-section. Proceedings of EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling, 2008. 1–8.

[43] Takayama K, Okabe M, Ijiri T, et al. Lapped solid textures: filling a model with anisotropic textures. ACM Transactions on Graphics, 2008, 27(3):Article 53.

[44] Praun E, Finkelstein A, Hoppe H. Lapped textures. Proceedings of ACM SIGGRAPH, 2000. 465–470.

[45] Benson D, Davis J. Octree textures. ACM Transactions on Graphics, 2002, 21(3).

[46] Lefebvre S, Hornus S, Neyret F. Octree textures on the GPU. Proceedings of GPU Gems 2, chapter 37. 2005:.

[47] Perlin K. An image synthesizer. Proceedings of ACM SIGGRAPH, 1985. 287–296.

[48] Peachey D R. Solid texturing of complex surfaces. Proceedings of ACM SIGGRAPH, 1985. 279–286.

[49] Ebert D S, Musgrave F K, Peachey D, et al. Texturing and Modeling: A Procedural Approach. Academic Press, 1994.

[50] Worley S. A cellular texture basis function. Proceedings of ACM SIGGRAPH, 1996. 291–294.

[51] Lagae A, Lefebvre S, Cook R, et al. A survey of procedural noise functions. Computer Graphics Forum, 2010, 29(8):2579–2600.

[52] Perlin K. Improving noise. Proceedings of ACM SIGGRAPH, 2002. 681–682.

[53] Perlin K, Hoffert E M. Hypertexture. Proceedings of ACM SIGGRAPH, 1989.

[54] Lewis J P. Algorithms for solid noise synthesis. Proceedings of ACM SIGGRAPH, volume 23, 1989. 263–270.

[55] Cook R L, DeRose T. Wavelet noise. ACM Transactions on Graphics, 2005, 24(3):803–811.

[56] Goldberg A, Zwicker M, Durand F. Anisotropic noise. ACM Transactions on Graphics, 2008, 27(3):54:1–54:8.

[57] Lagae A, Vangorp P, Lenaerts T, et al. Procedural isotropic stochastic textures by example. Computer & Graphics, 2010, 34(4):312–321.

[58] Ebert D S, Musgrave F K, Peachey D, et al. Texturing and Modeling: A Procedural Approach. 3 ed., Academic Press, 1994.

[59] Orzan A, Bousseau A, Winnemoller H, et al. Diffusion curves: a vector representation for smooth-shaded images. ACM Transactions on Graphics, 2008, 27(3):Article 92.

[60] Chang H H, Hong Y. Vectorization of hand-drawn image using piecewise cubic Bezier curve fitting. Pattern Recognition, 1998, 31(11):1747–1755.

[61] Zou J J, Yan H. Cartoon image vectorization based on shape subdivision. Proceedings of Computer Graphics International, 2001. 225–231.

[62] Hilaire X, Tombre K. Robust and accurate vectorization of line drawings. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006, 28(6):890–904.

[63] Lecot G, Levy B. Ardeco: automatic region detection and conversion. Proceedings of Eurographics Symposium on Rendering, 2006. 349–360.

[64] Price B, Barrett W. Object-based vectorization for interactive image editing. The Visual Computer, 2006, 22(9):661–670.

[65] Sun J, Liang L, Wen F, et al. Image vectorization using optimized gradient meshes. ACM Transactions on Graphics, 2007, 26(3):Article 11.

[66] Lai Y K, Hu S M, Martin R. Automatic and topology-preserving gradient mesh generation for image vectorization. ACM Transactions on Graphics, 2009, 28(3):Article 85.

[67] Xia T, Liao B, Yu Y. Patch-based image vectorization with automatic curvilinear feature alignment. ACM Transactions on Graphics, 2009, 28(5):Article 115.

[68] Scalable Vector Graphics (W3C SVG Working Group). http://www.w3.org/Graphics/SVG/.

[69] Loop C, Blinn J. Resolution Independent Curve Rendering using Programmable Graphics Hardware. Proceedings of ACM SIGGRAPH 2005, 2005. 1000–1010.

[70] Nehab D, Hoppe H. Random-access rendering of general vector graphics. ACM Transactions on Graphics, 2008, 27(5):135:1–10.

[71] Qin Z, McCool M D, Kaplan C. Precise vector textures for real-time 3D rendering. Proceedings of SIGRAPH Symposium on Interactive 3D Graphics and Games 2008, Redwood City, California, USA, 2008. 199–206.

[72] Jeschke S, Cline D, Wonka P. A GPU Laplacian solver for diffusion curves and Poisson image editing. ACM Transactions on Graphics, 2009, 28(5).

[73] Jeschke S, Cline D, Wonka P. Rendering surface details with diffusion curves. ACM Transactions on Graphics, 2009, 28(5).

[74] Ramanarayanan G, Bala K, Walter B. Feature-based textures. Proceedings of Eurographics Symposium on Rendering, 2004. 65–73.

[75] Sen P. Silhouette maps for improved texture magnification. Proceedings of Graphics Hardware, 2004. 65–73.

[76] Tumblin J, Choudhury P. Bixels: picture samples with sharp embedded boundaries. Proceedings of Eurographics Symposium on Rendering, 2004. 186–196.

[77] Tarini M, Cignoni P. Pinchmaps: textures with customizable discontinuities. Computer Graphics Forum, 2005, 24(3):557–568.

[78] Parilov E, Zorin D. Real-time rendering of textures with feature curves. ACM Transactions on Graphics, 2008, 27(1):Article 3.

[79] Steinhauser M O. Computational Multiscale Modeling of Fluids and Solids. Springer, 2008.

[80] Finkel R, Bentley J L. Quad trees: a data structure for retrieval on composite keys. Acta Information, 1974, 4(1):1–9.

[81] Fuchs H, Kedem Z M, Naylor B. On visible surface generation by A priori tree structures. ACM Computer Graphics, 1980, 14(3):124–133.

[82] Bentley J L. Multidimensional binary search trees used for associative searching. Communications of the ACM, 1975, 18(9).

[83] Ricci A. A constructive geometry for computer graphics. Computer Jornal, 1974, 16(2):157–160.

[84] Blinn J F. A generalization of algebraic surface drawing. ACM Transactions on Graphics, 1982, 1(3):235–256.

[85] Barr A H. Global and local deformations of solid primitives. Computer Graphics, 1984, 18(3):21–30.

[86] Nishimura H, Hirai M, Kawai T, et al. Object modeling by distribution function and a method of image generation. Proceedings of Electronics Communication Conference ’85, 1985. 718–725.

[87] Hoffman C, Hopcroft J. Automatic surface generation in computer aided design. Visual Computer, 1985, 1:92–100.

[88] Wyvill G, McPheeters C, Wyvill B. Data structure for soft objects. The Visual Computer, 1986, 2(4):227–234.

[89] Bloomenthal J. Polygonization of implicit surfaces. Computer Aided Geometric Design, 1988, 5(4):341–355.

[90] Bridson R, Marino S, Fedkiw R. Simulation of clothing with folds and wrinkles. Proceedings of ACM/Eurographics Symposium on Computer Animation, 2003. 28–36.

[91] Hsu S W, Keyser J. Piles of objects. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2010, 2010, 29(6).

[92] Hoff K E, Keyser J, Lin M, et al. Fast computation of generalized Voronoi diagrams using graphics hardware. SIGGRAPH ’99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques, 1999..

[93] Frisken S F, Perry R N, Rockwood A P, et al. Adaptively sampled distance fields: a general representation of shape for computer graphics. Proceedings of ACM SIGGRAPH, 2000. 249–254.

[94] Carucci F. Inside geometry instancing. Proceedings of GPU Gems 2, chapter 3. 2005:.

[95] Wei L Y, Han J, Zhou K, et al. Inverse texture synthesis. ACM Transactions on Graphics, 2008, 27(3).

[96] Wang H, Wexler Y, Ofek E, et al. Factoring repeated content within and among images. ACM Transactions on Graphics, 2008, 27(3).

[97] Zhou K, Ren Z, Lin S, et al. Real-time smoke rendering using compensated ray marching. ACM Trans. Graph., 2008, 27(3):Article 36.

[98] Cockburn A. Using both incremental and iterative development. STSC CrossTalk, 2008, 21(5):27–30.

[99] Larman C, Basili V R. Iterative and incremental development: a brief history. IEEE Computer, 2003, 36(6):47–56.

[100] Bloomenthal J. Implicit Surfaces. Encyclopedia of Computer Science and Technology, 2001..

[101] Autodesk AutoCAD. http://usa.autodesk.com/autocad/.

[102] Autodesk 3ds Max. http://usa.autodesk.com/3ds-max/.

[103] Autodesk Maya. http://usa.autodesk.com/maya/.

[104] Blender. http://www.blender.org/.

[105] Schmidt R, Wyvill B, Sousa M, et al. ShapeShop: sketch-based solid modeling with BlobTrees. Proceedings of 2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling, 2005. 53–62.

[106] Erleben K, Dohlmann H. Signed Distance Fields Using Single-Pass GPU Scan Conversion of Tetrahedra. Proceedings of GPU Gems 3, chapter 34. 2007:.

[107] Sigg C, Peikert R, Gross M. Signed distance transform using graphics hardware. Proceedings of IEEE Visualization, 2003. 83–90.

[108] Mauch S. Efficient Algorithms for Solving Static Hamilton-Jacobi Equations[D]. California Institute of Technology, 2003.

[109] Jones M W. 3D distance from a point to a triangle. Technical report, University of Wales Swansea, 1995.

[110] Galyean T, Hughes J. Sculpting: an interactive volumetric modeling technique. Computer Graphics, 1991, 25(4):267–274.

[111] Wu Q, Yu Y. Feature matching and deformation for texture synthesis. ACM Transactions on Graphics, 2004, 23(3):364–367.

[112] Hertzmann A, Jacobs C E, Oliver N, et al. Image analogies. Proceedings of ACM SIGGRAPH, 2001. 327–340.

[113] Extensible Markup Language (XML). http://www.w3.org/XML/.

[114] X3D. http://www.web3d.org/x3d/.

[115] Wei L Y. Parallel Poisson disk sampling. ACM Trans. Graph., 2008, 27(3).

[116] Bullet Physics Library. http://bulletphysics.org/.

[117] Turk G. Generating textures on arbitrary surfaces using reaction-diffusion. Computer Graphics, 1991, 25(4):289–298.

[118] Bowers J, Wang R, Wei L Y, et al. Parallel Poisson disk sampling with spectrum analysis on surfaces. ACM Trans. Graph., 2010, 29(5).

[119] Weaire D, Phelan R. A counter-example to Kelvin's conjecture on minimal surfaces. Philosophical Magazine Letters, 1994, 69(2):107–110.

[120] Sethian J. Level Set Methods and Fast Marching Methods. Cambridge University Press, 1999.

[121] Powell M J D. Approximation Theory and Methods. Cambridge University Press, 1981.

[122] Buhmann M. Radial Basis Functions: Theory and Implementations. Cambridge University Press, 2003.

[123] Zhu C, Byrd R, Lu P, et al. L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. ACM Trans. Math. Softw., 1997, 23(4):550–560.

[124] Cohen-Steiner D, Alliez P, Desbrum M. Variational shape approximation. ACM Trans. Graph., 2004, 23(3):905–914.

[125] Welsh D, Powell M. An upper bound for the chromatic number of a graph and its application to timetabling problems. The Computer Journal, 1967, 10(1):85–86.

[126] Corcoran A L, Wainwright R L. A genetic algorithm for packing in three dimensions. Proceedings of ACM/SIGAPP Symposium on Applied Computing, 1992.

[127] Shishkovtsov O. Deferred shading in S.T.A.L.K.E.R. Proceedings of GPU Gems 2, chapter 9. 2005:.

[128] Cantlay I. Mipmap-level measurement. Proceedings of GPU Gems 2, chapter 28. 2005:.

[129] Marschner S, Jensen H W, Cammarano M, et al. Light scattering from human hair fibers. ACM Transactions on Graphics, 2003, 22(3):780–791.

[130] Paris S, Chang W, Kozhushnyan O I, et al. Hair Photobooth: geometric and photometric acquisition of real hairstyles. Proceedings of ACM SIGGRAPH, 2008.

[131] Yu Y. Modeling realistic virtual hairstyles. Proceedings of Pacific Graphics, 2001. 295–304.

[132] Paris S, Briceño H, Sillion F. Capture of Hair Geometry from Multiple Images. Proceedings of ACM SIGGRAPH, 2004. 712–719.

[133] Wei Y, Ofek E, Quan L, et al. Modeling hair from multiple views. ACM Transactions on Graphics, 2005, 24(3):816–820.

[134] Kim T Y, Neumann U. Interactive multiresolution hair modeling and editing. ACM Transcations on Graphics, 2002, 21(3).

[135] Watanabe Y, Suenaga Y. A trigonal prism-based method for hair image generation. IEEE Computer Graphics and Applications, 1992, 12(1):47–53.

[136] Chen L H, Saeyor S, Dohi H, et al. A system of 3D hair style synthesis based on the wisp model. The Visual Computer, 1999, 15(4):159–170.

[137] Daldegan A, Thalmann N, Kurihara T, et al. An integrated system for modeling, animating and rendering hair. Computer Graphics Forum (Eurographics '93), 1993, 12(3):211–221.

[138] Hadap S, Thalmann N. Interactive hair styler based on fluid flow. Proceedings of Eurographics Workshop on Computer Animation and Simulation, 2000.

[139] Choe B, Ko H S. A statistical wisp model and pseudophysical approaches for interactive hairstyle generation. IEEE Transactions on Visualization and Computer Graphics, 2005, 11(2):160–170.

[140] Ishikawa T, Kazama Y, Sugisaki E, et al. Hair motion reconstruction using motion capture system. ACM SIGGRAPH 2007 posters, 2007..

[141] Yamaguchi T, Wilburn B, Ofek E. Video-based modeling of dynamic hair. Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology, Berlin, Heidelberg, 2009. 585–596.

[142] Hertzmann A, Oliver N, Curless B, et al. Curve analogies. Proceedings of Eurographics workshop on Rendering, 2002. 233–246.

[143] Bhat P, Ingram S, Turk G. Geometric texture synthesis by example. Proceedings of Eurographics/ACM SIGGRAPH symposium on Geometry processing, 2004. 41–44.

[144] Lai Y K, Hu S M, Gu X, et al. Geometric texture synthesis and transfer via geometry images. Proceedings of ACM Symposium on Solid and Physical Modeling, 2005. 15–26.

[145] Lagae A, Dumont O, Dutre P. Geometry Synthesis by Example. Proceedings of International Conference on Shape Modeling and Applications, 2005. 176–185.

[146] Zhou K, Huang X, Wang X, et al. Mesh quilting for geometric texture synthesis. ACM Transactions on Graphics, 2006, 25(3).

[147] Zhou H, Sun J, Turk G, et al. Terrain synthesis from digital elevation models. IEEE Transactions on Visualization and Computer Graphics, 2007, 13(4):834–848.

[148] Ward K, Bertails F, Kim T Y, et al. A survey on hair modeling: styling, simulation, and rendering. IEEE Transactions on Visualization and Computer Graphics, 2007, 13(2):213–234.

[149] Lloyd S P. Least squares quantization in PCM. IEEE Transactions on Information Theory, 1982, 28(2):129–137.

[150] Bloomenthal J. Calculation of reference frames along a space curve. Graphics Gems, 1990, 1:567–571.

[151] Hampel F R, Ronchetti E M, Rousseeuw P J, et al. Robust Statistics: The Approach Based on Influence Functions. New York: Wiley, 1986.

[152] Kajiya J T, Kay T L. Rendering fur with three dimensional textures. Computer Graphics, 1989, 23(3):271–280.

[153] Yuksel C, Keyser J. Deep opacity maps. Computer Graphics Forum (Proceedings of EUROGRAPHICS 2008), 2008, 27(2).

# 致　谢

衷心感谢导师郭百宁教授五年来对我的悉心指导。郭老师像师长也像朋友，他的信任使我能够自由选择自己感兴趣的研究课题，他的言传身教总能令我更清楚地认识到自己的成绩与不足，他对图形学的热情、对研究方向的把握以及对工作的投入更是深深影响了我。

感谢童欣研究员、魏立一研究员、周昆教授、俞益洲教授，与他们的每一次合作对我而言都是极其宝贵的学习机会。他们严谨的治学态度、严密的思维方法以及严格的自我要求潜移默化的影响将令我受益终生。更令我感动的是，除了研究工作，他们对我生活中遇到的问题也给予了真诚的关心和帮助，使我更快地走出各种困境。

感谢吴念乐教授、徐湛教授、李丽老师，他们为了给我创造一个更好的学术研究环境默默地做出了大量努力。

感谢董悦、侯启明、龚敏敏、孙鑫、马重阳、沈方阳、Steve Lin、王希、邓可，与他们的交流与讨论令我获益匪浅。

感谢雍俊海教授和张慧教授，本科时正是他们的课程引领我走进图形学的大门并最终选择了这条研究之路。

感谢参与论文评审与答辩委员会的各位老师，他们提出的大量宝贵意见和建议不仅帮助我更好地完善了这篇论文，也对我今后的研究工作给予许多启发。

感谢我的家人，一路走来有他们的爱与支持是我最大的幸运。

# 声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

# 附录 A 体对象标记语言（VOML）

## A.1 语法规则

体对象标记语言（Volumetric Object Markup Language，VOML）的设计基于可扩展标记语言（XML）。一个 VOML 文档的基本结构是一组相互嵌套的标签（tag），不同类型的标签内部可以定义不同的属性（attribute），如下列形式：

```
<TAG1 attribute1="value1" attribute2="value2">
  <TAG2 attribute3="value3">
    ...
  </TAG3>
</TAG1>
```

以下列出的是一个简单的 VOML 文档，定义了一个包含两个区域的体数据。

```
<OBJECT name="minimum">
  <SDF name="boundary" file="bound.sdf">
    <POSITIVE region="OUT" rgb="1 0 0"/>
    <NEGATIVE region="IN" rgb="0 0 1"/>
  </SDF>
</OBJECT>
```

### A.1.1 OBJECT 标签

OBJECT 标签是 VOML 文档中最基本的标签，用于声明一个体对象。该标签可以嵌套 SDF、REGION 等其它标签定义对象内部的区域划分。

**属性：**

- *name*: 体对象的名称/唯一标识符。

### A.1.2 SDF 标签

SDF 标签用于声明一个符号距离函数（Signed Distance Function）。一对 SDF 标签内部可以嵌套 POSITIVE 和 NEGATIVE 两个子标签，分别对应 SDF 树中 SDF 节点的两个子节点指针。通过进一步在 POSITIVE 或 NEGATIVE 标签内部嵌套另外的 SDF 标签可以声明任意复杂的 SDF 树结构。

SDF 标签本身并不直接定义符号距离函数的内容，而是通过 *file*属性指向包含距离函数定义的外部文件。SDF 标签可以直接在 OBJECT 标签内部定义，亦可以在 OBJECT 标签外提前定义。

具有不同 *name*属性的 SDF 标签可以具有相同的 *file*属性，虽然这些 SDF 标签在逻辑上声明了不同的距离函数，并且可以有不同的自身属性，但事实上它们在内存中指向相同的距离函数定义，可以看作是同一个距离函数的多个实例。

**属性：**

- *name*: 符号距离函数的名称/唯一标识符。
- *file*: 包含符号距离函数定义的外部文件路径。
- *translate / scale / rotate*: 此三个属性定义应用于该符号距离函数的仿射变换。其中每个属性的值均为一浮点数的三元组，rotate 属性的值为以角度为单位的欧拉角。
- *offset*: 通过为符号距离函数增加一个常量（令 $D(x)$ 为原始距离函数，则新的距离函数为 $D'(x) = D(x) + c$）使过零等值面发生偏移。利用该属性可以产生边界曲面膨胀/收缩，或是多层结构等效果。
- *instances*: 定义距离函数实例化。该属性指向一个包含一系列仿射变换的外部文件。与多个具有相同 *file*属性、不同 *name*属性的 SDF 标签不同，由 *instances*属性定义的所有距离函数实例作为一个整体用于定义一个单独的距离函数。

**示例：**

同一符号距离函数的多个实例：

```
<SDF name="shape1" file="shape.sdf" />

<SDF name="shape2" file="shape.sdf" scale="1.5 0.8 0.8"
                                    rotate="0 90 0" />

<SDF name="shape3" file="shape.sdf" offset="0.1" />
```

通过实例化定义的一个距离函数：

```
<SDF name="instanced" file="shape.sdf"
                      instances="seeds.txt"/>
```

### A.1.3 POSITIVE 与 NEGATIVE 标签

这两种标签仅出现在一对 SDF 标签内部，用于分别定义一符号距离函数的“正区域”与“负区域”的内容。若其中一个区域还将被其它距离函数继续剖分，则对应的 POSITIVE 或 NEGATIVE 标签中将嵌入其它的 SDF 标签。否则，该标签对应于 SDF 树中的一个叶节点，通过 *region*属性指向一个体数据区域的定义。

**属性：**

- *region*: （当标签对应于 SDF 树中的叶节点时）所指向的体数据区域的名称/唯一标识符。

### A.1.4 REGION 标签

REGION 标签定义体数据中一个区域内部的颜色函数。与 SDF 标签类似，在一个复杂的 VOML 文档中 REGION 标签可以在被引用前声明。不同的 POSITIVE 或 NEGATIVE 标签可以指向同一个 REGION 标签。

**属性：**

- *name*: 区域的名称/唯一标识符。
- *empty*: 值可以为 “true” 或 “false”，标明该区域是否为空区域（在绘制时不可见）。
- *rgb*: 定义一均色颜色函数。
- *opacity*: 区域的不透明度，在基于 ray-tracing 的绘制方法中有效。
- *bitmap*: 指向一离散采样的体纹理作为颜色函数。
- *texture*: 指向一基于过程噪声的颜色函数。该函数的定义见下文 TEXTURE 标签。
- *translate / scale / rotate*: 与 SDF 标签类似，对于颜色函数可以定义坐标的仿射变换。

**示例：**

不同类型的区域定义：

```
<REGION name="region1" rgb="1 0 0" />
<REGION name="region2" bitmap="wood.dat" scale="2 2 2" />
<REGION name="region3" texture="noise1" opacity="0.1" />
<REGION name="region4" empty="true" />
```

### A.1.5 EMBEDDED 标签

EMBEDDED 标签仅出现在一对 REGION 标签内部，定义了嵌入该区域内部的另外一个体模型。一对 REGION 标签内部可以并列出现多组 EMBEDDED 标签，以表示同一区域内嵌入的多个不同体模型。EMBEDDED 本身也可以定义体模型的实例化，与 SDF 标签类似，也是通过引用定义在外部文件中的仿射变换实现。

**属性：**

- *object*: 被嵌入子模型的名称/唯一标识符。
- *translate / scale / rotate*: 被嵌入子模型的仿射变换。
- *instances*: 定义被嵌入子模型的实例化。该属性指向一个包含一系列仿射变换的外部文件。

**示例：**

不同类型的区域定义：

```
<REGION name="region1" rgb="1 0 0">
  <EMBEDDED object="sub_obj_1" />
  <EMBEDDED object="sub_obj_2" scale="0.7 0.7 0.7"
                               rotate="45 0 90"/>
  <EMBEDDED object="sub_obj_3" instances="inst.txt"/>
</REGION>
```

### A.1.6 TEXTURE 标签

TEXTURE 标签定义一个基于过程噪声的颜色纹理函数。在目前的实现中，我们采用预计算的白噪声函数作为基础，用户可以将噪声函数的值（范围在 0 至 1 之间）映射到一个一维颜色函数，也可以通过指定仿射变换一定程度上控制噪声函数的各向异性。

**属性：**

- *name*: 纹理函数的名称/唯一标识符。
- *ramp*: 一维颜色函数的文件路径。
- *translate / scale / rotate*: 针对纹理函数的仿射变换。

### A.1.7 IMPORT 标签

IMPORT 标签用于在 VOML 文档中包含另一个 VOML 文档的内容。这使得一个复杂体对象的定义可以分解为多个独立的 VOML 文档从而便于管理和维护。

**属性：**

- *file*: 包含的文件路径。

## A.2 示例文档

以下为图3.21中体数据的完整 VOML 文档。

```
<Import file="orange_textures.voml"/>

<!--Defines a single particle for the pith structure-->
<OBJECT name="pith_particle">
  <SDF name="pith"    file="pith.sdf"/>
  <SDF name="in_pith" file="pith.sdf" offset="0.02"/>

  <REGION name="EMPTY" empty="true"/>
  <REGION name="PITH" rgb="1 0.8 0.2" opacity="0.3"/>
  <REGION name="IN_PITH" texture="pith_bg"
          opacity="0.2"/>

  <!-- tree definition -->
  <SDF name="pith">
    <POSITIVE region="EMPTY"/>
    <NEGATIVE>
      <SDF name="in_pith">
        <POSITIVE region="PITH"/>
        <NEGATIVE region="IN_PITH"/>
      </SDF>
    </NEGATIVE>
  </SDF>
</OBJECT>

<!--Defines a single section in an orange-->
<OBJECT name="section">
  <SDF name="section" file="section.sdf"/>

  <REGION name="IN_SECTION" texture="pith_bg"
        scale="0.1 0.1 0.1" opacity="0.3">
    <EMBEDDED object="pith_particle"
              instances="inst_pith.txt"/>
  </REGION>
```

```
  <SDF name="section">
    <POSITIVE region="EMPTY"/>
    <NEGATIVE region="IN_SECTION"/>
  </SDF>
</OBJECT>

<!--The main orange object-->
<OBJECT name="orange">
  <SDF name="peel" file="peel.sdf"/>
  <SDF name="inner_peel" file="peel.sdf"
       offset="0.03"/>
  <SDF name="peel_cells" file="cell.sdf"
       instances="inst_cell.txt"/>
  <SDF name="hollow" file="hollow.sdf"
       instances="inst_hollow.txt"/>

  <REGION name="PEEL" texture="skin_bg2"
          scale="0.2 0.2 0.2"/>
  <REGION name="PEEL_CELL" texture="skin_bg3"
          scale="0.2 0.2 0.2" opacity="0.2"/>
  <REGION name="INNER_SKIN" rgb="1 0.7 0.3"
          opacity="0.1"/>
  <REGION name="HOLLOW" texture="hollow_bg"
          scale="0.1 1 0.1" opacity="0.01"/>
  <REGION name="INTERIOR" texture="inter_section"
          scale="0.06 0.06 0.06" opacity="0.02">
    <EMBEDDED object="section" />
    <EMBEDDED object="section" rotate="0 0 40"/>
    <EMBEDDED object="section" rotate="0 0 80"/>
    <EMBEDDED object="section" rotate="0 0 120"/>
    <EMBEDDED object="section" rotate="0 0 160"/>
    <EMBEDDED object="section" rotate="0 0 200"/>
    <EMBEDDED object="section" rotate="0 0 240"/>
    <EMBEDDED object="section" rotate="0 0 280"/>
    <EMBEDDED object="section" rotate="0 0 320"/>
  </REGION>

  <SDF name="peel">
    <POSITIVE>
      <SDF name="peel_cells">
        <POSITIVE region="PEEL"/>
        <NEGATIVE region="PEEL_CELL"/>
      </SDF>
    </POSITIVE>
    <NEGATIVE>
      <SDF name="inner_peel">
        <POSITIVE region="INNER_SKIN"/>
        <NEGATIVE>
```

```
          <SDF name="hollow">
            <POSITIVE region="INTERIOR"/>
            <NEGATIVE region="HOLLOW"/>
          </SDF>
        </NEGATIVE>
      </SDF>
    </NEGATIVE>
  </SDF>
</OBJECT>
```

# 个人简历、在学期间发表的学术论文与研究成果

## 个人简历

1984 年 4 月 28 日出生于四川省成都市。

2002 年 9 月考入清华大学软件学院计算机软件专业，2006 年 7 月本科毕业并获得工学学士学位。

2006 年 9 月免试进入清华大学高等研究院攻读工学博士学位至今。

## 发表的学术论文

[1] Wang L, Zhou K, Yu Y, et al. Vector solid textures. ACM Transactions on Graphics (SIGGRAPH 2010), 2010, 29(4):1–8. (SCI 收录, 检索号: 624GZ)

[2] Wang L, Yu Y, Zhou K, et al. Example-based hair geometry synthesis. ACM Transactions on Graphics (SIGGRAPH 2009), 2009, 28(3):1–9. (SCI 收录, 检索号: 487MF)

[3] Wang L, Wei L Y, Zhou K, et al. High dynamic range image hallucination. EUROGRAPHICS Symposium on Rendering, 2007, 321–326.

[4] Wang L, Wang X, Sloan P P, et al. Rendering from compressed high dynamic range textures on graphics hardware. SIGGRAPH Symposium on Interactive Graphics and Games, 2007, 17–24

## 研究成果

[1] Shum H Y, Guo B, Zhou K, et al. High dynamic range image hallucination: USA, No.11/781,227. (美国发明专利申请号.)