2.1概述及相关工作

在以SPH，MPM等为代表的各类无网格法仿真计算的结果表示中，多以粒子的形式存在。各粒子携带有物质的质量、速度、温度等各种物理量，而其最重要的一个特点就是，粒子的位置分布是非规则的，所以这也造成了其所代表的模型很难从这样的非规则分布粒子中提取表面信息。

针对这类问题，隐式表面方法是一种被广泛采用的重构表面信息的方法。该方法主要有两种形式，其一为Hoppe等人【1】提出的符号距离函数法，该方法分两个步骤：首先通过一点附近邻域内的点集进行线性拟合，得到一个局部的正切平面；然后对于空间中任意一点，计算到的距离函数，若位于表面外则记为正，反之为负。则该重构的表面便可以用隐式地表达。其二为M¨uller等人【2】使用的用标量场的等值面定义表面的算法。该算法在每一个粒子的位置定义一个“核”，空间一点处的值为该点到核心的半径的函数，对空间中任意一点，该处的标量场值为所有“核”在该点处贡献的叠加。

第一类方法适用于粒子数量较大、密度分布较为均匀的情况，而当模型局部的粒子比较少的时候，通过局部拟合来构造表面的方法明显不合适，而第二类方法却不受这个限制，即使只有一个粒子，也可以完成表面构造。而本文需要处理碎片等情况，会有局部粒子较少的情况，所以考虑使用第二类方法。

本文中此部分的功能主要基于Yu和Turk在2013年的工作【3】，这篇文章主要基于【2】的方法，但是通过引入各向异性核，并加入表面粒子位置调整功能从而实现了更加光滑的表面构造，并且能够保持边角棱等模型的局部细节。本章针对【3】中的一些不足之处对其中的算法提出一定改进，以及结合本文后处理模块的特点和需求，对其算法进行了适用性扩展，主要工作如下：

1.【3】中的主要效率瓶颈为标量场的计算，其算法要为模型中的每个粒子计算标量场值，这消耗了大量时间，同时存储所有网格结点也会耗费较大内存。本文通过引入一种增长型八叉树结构，可以实现只计算模型表面附近的标量场从而达到大大降低计算时间和减小内存消耗的目的；同时实现了空间划分，加速光线跟踪算法中光线和重构出的三角形表面求交运算的速度。

2.由于【3】中算法仅仅是针对算例，需要根据模型数据的不同手工调试参数，这对于后处理系统来说明显不现实，于是本文设计了一套对数据文件的自动分析模块，具体实现参考第四章。该模块的分析结果在表面重构中可以动态自动设置标量场值范围、粒子半径及邻近粒子搜索半径等参数。

2.2各向异性核重构算法描述

2.2.1表面的隐式表示

对各向同性核，标量场可以表示为



其中j为所有影响到位置 的粒子的编号，W即为每个粒子处的各向同性核，形式如下：



其中是一个缩放值，d为仿真的维数，h为核心的影响半径，一般由仿真计算阶段提供，为某点到核心的位移矢量 ，P为一个对称的衰减函数，用来描述场变量值在核内的分布。分母中的保证了场值在一个核内的积分为固定值，这样可以保证不同粒子对整体表面的贡献度只和该粒子所实际代表的物理量相关，不会受粒子间距的影响。将所有影响到任意空间点的核在处的场值累加后便得到了该点最终的场值，在设定阈值t后，该标量场的t等值面便为该表面的隐式表示。

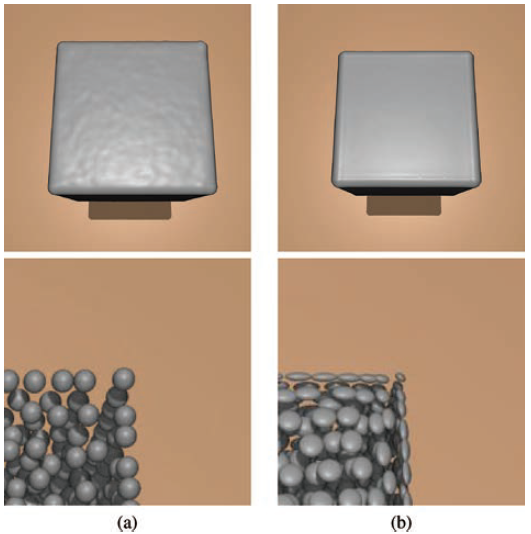
对各向异性核，【3】中使用了一个的正实数矩阵来代替，这使得核心从球体变成了椭球体（椭圆），的各阶特征值的倒数代表该椭球体的各主轴长度。此时核心表示为



于是场变量表示为



其中是影响域内第j个粒子为了平滑表面而被调整后的位置。其中矩阵的构造需要用到邻近粒子的信息，即要进行邻近点搜索，并有构造过程中还需要计算矩阵特征值和特征向量的操作，于是这是该算法最耗时的地方。不同核心的对比效果如图【】所示



其中（a）是各向同性核的表示及其效果，可见容易产生表面的凹凸不平，(b)是各向异性核的效果，可见表面明显比各向同性核要平滑很多。

2.2.2 隐式表面的显式表示方法

由于使用以上算法重构出的表面时隐式表示的，不能够直接用来渲染生成最终图形，所以需要对该隐式表面进行采样表示。

对于光线跟踪渲染框架，一种比较直观的办法是在追踪光线的运行路径上进行二分查找，即以一个初始步长向前移动光线前端，计算每一步光线前端处的场变量值，当该值大于等值面阈值后，将步长二分，并将光线前端退回新步长距离，如此二分下去，直到步长值小于给定精度要求。过程如图【】所示。在找到交点后，根据标量场的梯度计算该点法向量即可实现光照计算完成渲染。

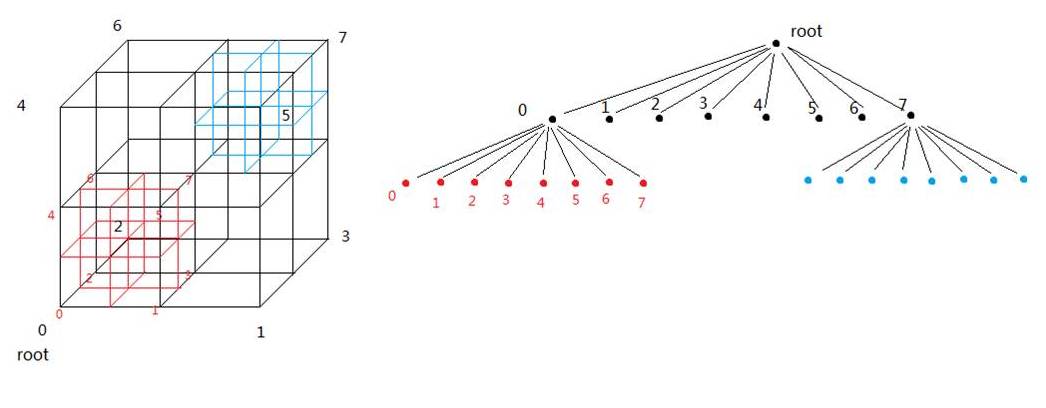
该方法的优点是精度非常高，因为它可以精确计算光线和物体表面的交点，但其缺点是每次需要用到物体表面的某一点信息时，都需要重新寻找该点，造成巨大的浪费，在进行光子追踪，阴影计算和表面光照计算时很可能若干次用到该点信息，这样将造成巨大的浪费，导致效率极端低下，于是现在比较流行的算法是Lorensen 和 Cline 在1987年提出的Marching Cubes算法【4】。

该算法通过在数据区内构建空间规则网格，并在网格点处对标量场进行采样，然后针对每个立方体单元，根据角点和重构表面的相对位置来在单元内构造三角形面片，找到全部表面的面片后再进行传统渲染。该算法的优点是对表面进行离散采样，速度较快，且生成的三角面片方便各种渲染体系进行渲染，适用性比较广；但是其缺点是需要构建空间规则网格，并计算所有网格点的场变量值，当模型比较巨大的时候，其运算量和内存消耗量都是非常可观的。

2.3基于扩展型八叉树的算法改进

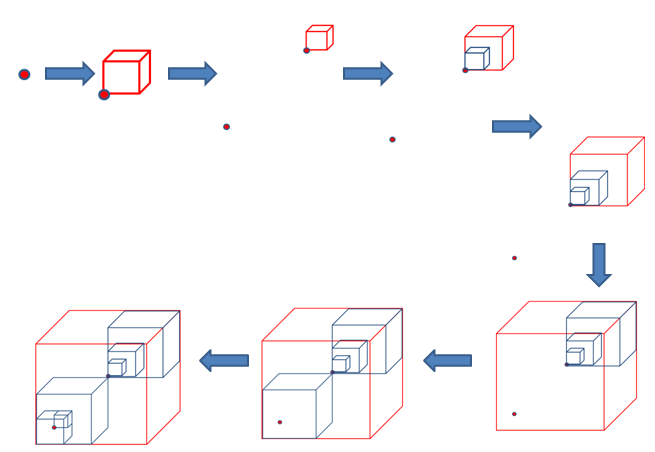
2.3.1数据结构

传统的八叉树结构是一种区域划分结构，它将容纳物体的最大区域按照每层八个卦限划分下去，直到某一层的一个区域内不再包含物体或已经划分到底层体素，如图【】所示



区域划分出的八个卦限按照x方向相差1，y方向相差2，z方向相差4的方式编号，这样对于空间任一点，可以由坐标直接计算出其所在卦限的子节点索引，无需将点坐标和区域边界进行比较。该结构插入和查找的时间复杂度均为，可以大大提高效率。

以上这种传统八叉树是在确定了物体最大范围情况下进行逐级细分，而有些情况下确定物体最大范围并不是一件容易的事，或者这个范围是在动态变化的，这时候这种数据结构实用起来就显得较为不方便。于是我对八叉树的生成方式进行了改进，使它能够从最底层的一个体素开始“增长”。图【】展示了增长式生成的过程



在插入每一个结点时，都按最小体素长度对结点位置进行对其，然后再进行插入操作；

Step1. 插入第一个结点时，以它为零点构造第一个最小的体素，并使之成为根结点，记高度为0；

Step2. 插入第二个结点时，如果它在根结点区域外，则构造大一层的立方体区域，使之成为新的根节点，记高度为1，原有根节点成为它的子节点；

Step3. 如果新插入结点和跟结点的距离大于根节点区域的边长，那么继续构造大一层的区域，更新根结点，高度增加1，重复step2直到该距离小于根节点区域边长

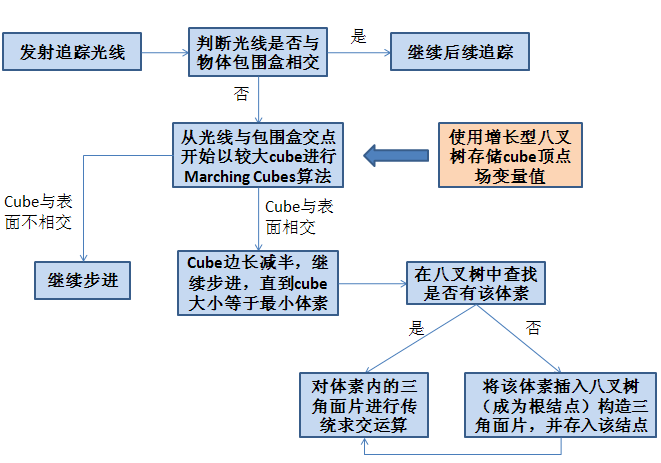
Step4. 判断新插入结点所在的卦限，由此调整原有根节点在新根节点中的卦限

Step5. 按照传统插入算法完成新结点的插入过程

2.3.2算法原理和框架

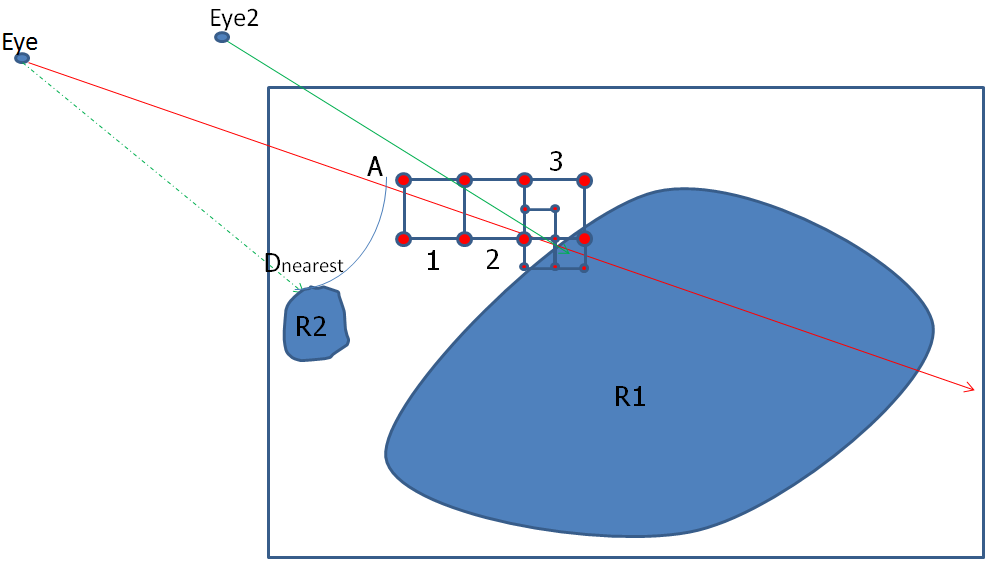
本文的重构算法和已有工作最大的区别在于，已有工作是将所有表面重构完成后再统一渲染，然而对物体的内部区域和不可见的表面所做的运算是一种巨大的浪费；而由于本文采用光线跟踪的渲染框架，追踪光线并不会到达物体内部（不透明物体）和不可见表面，所以可以只对光线跟踪点进行表面重构，即实现边渲染边重构的结构，省去了不必要的计算量，同时也节省了内存消耗。而且，当变换渲染视角后，该算法也无需重新构造全部表面，只需要查询之前构造好的表面，并继续重构变换视角后新看到的一部分表面即可。

算法流程如图【】所示



2.3.3算法实现

由于重构算法中最耗时的部分为标量场值的计算（包括计算每个粒子的G矩阵，累加计算值等），所以改进的一切出发点就是尽可能地减少求值的次数。一般来说，在载入模型时，都要对模型求一次AABB包围盒，这样光线如果不与包围盒相交，就无需进行表面重构操作；进一步的，我们还可以用k-d树结构实现的最近邻搜索算法进行一次最近的粒子搜索，若该距离大于视点到光线与包围盒交点距离，则还可以将光线前端进一步前移，进一步减少值计算次数，如图【】所示



当从视点（Eye）发出追踪光线后（红色光线），首先搜索最近粒子，并找到了R2区域，得到了Dnearest，于是可以直接从A点开始执行MarchingCubes算法，并一开始以一个实现设定好的较大边长的cube步进，如果cube不与物体表面相交，则将光线穿出该cube，并向前移动一个小量，由此计算坐标可以找到下一个cube，如图中的1和2；当cube与物体表面相交时，如3，则二分cube的边长按上面的步骤继续步进，直到cube已经减小到最小体素，这时当找到第一个与物体相交的最小体素时，按照MarchingCubes算法构造三角面片，并将其存储在体素（八叉树根节点）中，然后计算光线与三角形面片的交点；如果不存在交点，则以最小cube继续步进，直到找到交点为止。图中色圆点是需要计算值的位置，计算一点的值时，为每个被第一次搜索到的粒子计算其G矩阵。这种策略导致当模型物体体积较大时，完全无需计算物体内部大量网格结点的值，且当另外一条光线再次经过已经构造好的表面时，只需要去计算光线和三角形面片交点即可，如图中Eye2的光线。以上情况针对于反射表面，如果是投射，当光线找到入射表面的交点射入物体内部时，可以再次增大步进cube的边长，这样物体内部也只需计算少量值。

2.4重构算法中各参数的自适应调整

考虑到本章算法针对的是一套后处理系统，不同于具体算例，它要求有较高的鲁棒性，应该尽可能地避免对不同模型和渲染情形下对各参数的手动设置。这其中最需要考虑的一个问题是，当不同模型的粒子密度差异很大，或者当一个模型被以不同的放大系数载入时。这两个问题分别可能产生如下效果：重构表面的质量，包括平滑度和对细节的捕捉；不同放大系数下重构表面的相似性，即不应该因放大系数的不同而使重构的表面产生结构上的不相似，当把表面按照某一缩放系数进行缩放后，应该和该缩放系数下加载的模型重构出的表面重合。为了达到以上要求，在重构算法中需要考虑的参数有如下几个：

1. 步进Cubes时最大cube的边长
2. 粒子半径
3. 邻近粒子搜索半径
4. 公式2-3中G矩阵的特征值大小
5. 标量场等值面的阈值

在以上5类参数中，除了第5个参数阈值，其余三个都可和某个长度量纲相关联，而参数5由于其和表面重构效果相关，且其效果和值并没有一个简单的线性关系，所以本文采用通过试验找到一个使效果最佳的阈值固定下来，通过调整参数1,2,3,4来实现前面的要求。

通过总结，不同模型粒子密度的差异和不同缩放系数导致的差异就是粒子的间距，所以本文采用粒子的平均距离这一长度量纲来调整参数1,2,3,4。

【3】中计算每个粒子的G矩阵方法如下：

首先文中对每个粒子，利用其邻近粒子信息构造了一个协方差矩阵C，



其中为一距离相关的权重函数，对C进行正交分解，



于是构造G矩阵如下，

，其中，为的各阶特征值

其中为每个粒子的半径，此参数可以通过缩放系数和全局平均半径动态调整，而的影响半径需要和粒子半径线性相关，且是单位旋转矩阵，于是对角阵需要为常数量纲；再由公式2-5分析C矩阵量纲为，于是只需要定义，其中为粒子平均半径，其具体计算方法详见第4章。为一个常量，根据表面重构的效果实验确定。

【3】中对各向异性核的形状做了一定限制，长轴和短轴最大比不得超过一个定值，根据前面确定的，取G的最大长轴即为各向异性核的最大影响半径，以此作为邻近粒子搜索半径，即可保证所有对搜索点有影响的各向异性核都被搜索到，保证结果的准确性。

另外，在计算值时，由公式2-3可知，其量纲为，为了使用固定的等值面阈值，需要对每一点计算后的值乘上使其变为常量纲。

对于最大步进cube的边长，可以使用粒子搜索半径的一个固定的小于1的倍数，这样可以保证每一次步进时，不会因为距离过大而漏掉部分粒子。

2.5结果与讨论

本文中渲染结果的分辨率为600\*500像素

值得注意的是，该算法在某些特殊情况下会发生个别区域的表面没有被重构的错误，经检查，是因为在MarchingCubes算法中构造三角形时需要利用标量场计算三角形每个顶点的法向量以便进行Gouround着色，而在计算三角形顶点位置时是按照阈值在cube各顶点之间线性插值得到的；然而实际标量场并非线性分布，这就会造成某些顶点处的场值为0，无法构造法向量，形成错误。目前为止这个问题没有得到非常好的解决，只是通过限制最小cube的边长不能大于粒子半径来避免，但是这样明显限制了算法的灵活性。

另外，由于本文的改进算法使用了较为粗大的步进cube作为初始cube，这导致了尺寸过小的细节，如固体的飞溅小碎片，液体的小液滴等，在有些情况下是无法被捕捉到的。此问题可以通过模型预处理阶段（详见第4章）去找到这些局部小细节，并预先使用已有方法重构出表面，最后和动态重构阶段的表面和在一起进行渲染。但是由于时间所限，本文还没有去实现该功能，一种可行的方法是在一个指定范围内搜索邻近粒子，根据搜索到的粒子数判断该处模型区域是否足够小。

2.6总结与展望

本章提出了基于各向异性核的表面重构算法的改进算法。该算法基于扩展型八叉树动态地构造表面，省去了不可见区域的不必要计算，并节省了内存消耗。同时，本章提出了重构算法中自适应参数的生成方法，使得后处理系统在渲染不同模型，在不同观察条件下都可以无需复杂的参数设置而实现良好效果成为了可能。

但是本文实现的该算法目前还是在CPU上进行运算，且没有使用任何并行。于是，即便在算法层面上对重构进行了一定的效率提升，但是其速度依然是比较慢的。鉴于光线跟踪系统和表面重构时邻近粒子搜索的高度并行特性，后续工作的重点应该是使用GPU对其进行并行化。

另外，对于不同的材料，其表面特征应是不同的，如固体材料表面棱角应更尖锐一些，液体材料表面棱角应更圆润一些，这一区别并没有在本文的算法中体现出来，这也是一个后续工作应当重点关注的问题。

[1]Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, Werner Stuetzle. Surface reconstruction from unorganized points. Computer Graphics (Proceeding of ACM SIGGRAPH 92),26(2):71-78,1992

【2】M¨ULLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications.

In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Eurographics Association, 154–159.

【3】Yu, J. and Turk, G. 2013. Reconstructing surfaces of particle-based fluids using anisotropic kernels. ACM Trans. Graph. 32, 1, Article 5 (January 2013)

【4】LORENSEN, W. E. AND CLINE, H. E. 1987.

Marching cubes: A high resolution 3d surface construction algorithm.

In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques. ACM, New York, 163–169.