基于 Surfel 表示的快速物体 Boolean 运算

严冬明 024942 **清华大学计算机系** yandm@cg.cs.tsinghua.edu.cn

武勃 026017 **清华大学计算机系** wb98@mails.tsinghua.edu.cn

摘要

本次课程试验我们力图实现 Bart Adams 和 Philip Dutré 在文献[1]中提出的基于 Surfel 表示的快速物体 Boolean 运算。该算法可对任意形状的基于 Surfel 表示的物体进行“交”,“并”和“差”运算。其核心是一个判断两个 Surfel 之间位置关系的快速内外检测算法。Bart Adams 和 Philip Dutré 称这个算法可以达到实时交互的效果（在一台 Athlon 1.6GHz PC 上运行速度最快可达 7.7FPS）。Boolean 运算后物体相接处会产生所谓的“参差边缘”，严重影响真实感。为了解决这一问题 Bart Adams 和 Philip Dutré 在[1]中提出了一个重采样算子，用于去除参差边缘。此外，他们还针对“并”运算提出了一个平滑算子，可以使得合并结果的相接部分具有自然的过渡，这一算子在他们的技术报告[2]中有详细描述。本次试验中我们也实现了这两个补充算法。实验结果表明，Surfel 表示下的物体确实有很强的真实感效果，但其 Boolean 运算并未能达到[1]中所说的运算速度，在我们的实验机器（PIV 2.0G, 512M RAM）上最快速度约为 2FPS。

1．介绍

在计算机图形学领域，CSG（Constructive Solid Geometry）早已成为一种十分有用的工具。通常 CSG 作用于简单形状的物体，如球，圆柱等，进而生成更复杂的几何形体。然而，对任意形状物体来说，Boolean 运算同样十分重要。[1]中提出的基于 Surfel 表示的 Boolean 运算可以对任意形状的物体进行“交”,“并”和“差”运算。Surfel 表示[3]是一种基于点的物体表示方法，它将物体表示为许多三维空间中的有向点的集合。形象地说，每个 Surfel 是一个三维空间中的二维小圆盘，圆盘中心是物体表面上的某点，圆盘法向是该点处物体的法向，这样一个 Surfel 就近似表示了物体在其圆心处一个小邻域内的表面形状。一个任意形状物体的表面就是由许多这样的 Surfel 表示的。对于基于 Surfel 表示的物体，Boolean 运算就归结为 Surfel 与 Surfel 之间的 Boolean 运算。而对于两个物体 A 和 B 而言，A 的 Surfel 中只有那些与 B 表面相交的 Surfel 才需要重新计算，这些 Surfel 只占很少一部分，而其余 A 的 Surfel 要么在 B 内部，要么在 B 外部，并不需要改变。基于这一想法，[1]中给出了一个快速 Surfel 内外检测算法，用来判断一个物体的 Surfel 相对于另一个物体的位置关系。整个 Boolean 运算包含如下两个主要步骤：1）判断两个待运算物体中的所有 Surfel 的相对位置，即内部、外部或相交；2）根据具体运算，去掉不需要的 Surfel，并对相交部分进行重采样运算。

实验报告的余下部分安排如下：第 2 节介绍如何用 Surfel 表示任意形状物体；第 3 节介绍内外检测算法；第 4 节介绍重采样算子；第 5 节介绍平滑算子；第 6 节对前算法有待改进的地方进行了讨论；第 7 节给出我们的结论和一些评价。

2．Surfel 物体表示

用 Surfel 表示物体，确切的说就是用 Surfel 表示物体的表面。每个 Surfel 主要由三个参数确定：空间中位置 $\mathbf{x}_s$，半径 r_s 和法向 $\mathbf{n}_s$。因此一个 Surfel 可以看作是垂直于 $\mathbf{n}_s$ 的圆心为 $\mathbf{x}_s$ 半径为 r_s 的圆盘。选取半径 r_s 时，应该使得圆盘在成像平面上的投影互相重叠，不产生空隙。具体的点采样算法在[3]中有详细描述，此处不再赘述。设各维的采样间隔为 h，则取 Surfel 半径为 $r_s = \sqrt{h}$。

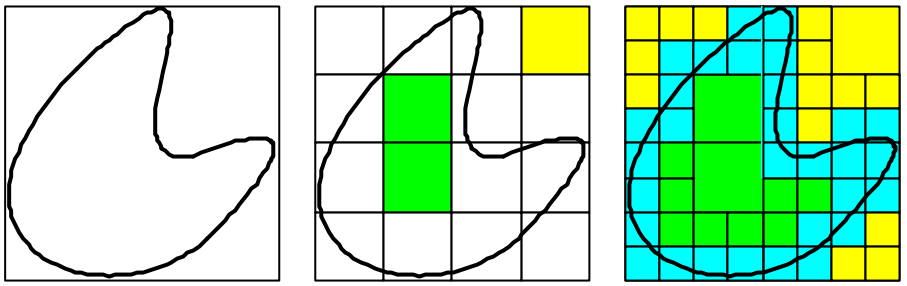


图 1．构造八叉树。（绿色节点为内部节点，黄色节点为外部节点，蓝色节点为边界节点）

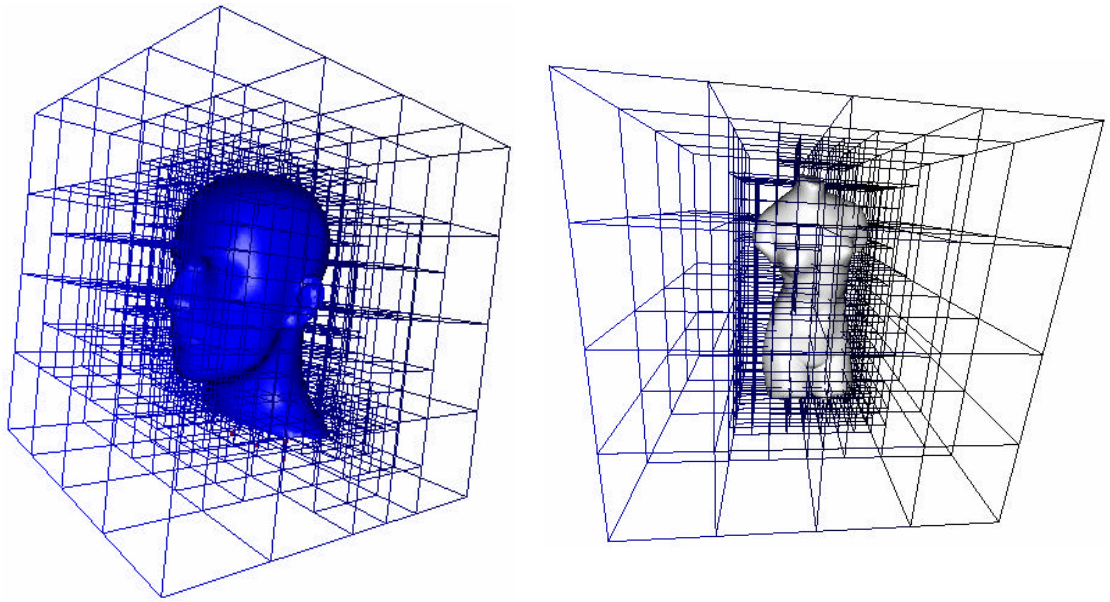
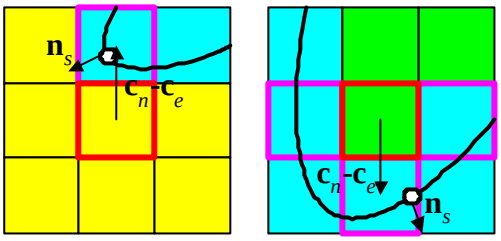


图 2．人头和维纳斯的八叉树剖分

3．内外检测

对于两个物体 A 和 B，Boolean 运算首先要判断它们的 Surfel 的相对位置关系，这就是内外检测算法所要做的。[1]中提出的快速内外检测算法基于三色八叉树结构，树的叶节点被分类为外侧、内侧或边界。边界节点中的 Surfel 是进一步处理的对象。



(a) (b)

图 3．判断空节点位置。(a)空节点只有一个非空邻居；(b)空节点有多个非空邻居。（红色框表示当前考虑节点，紫色框表示当前框的非空邻居。）

3.1 构造八叉树

简单起见，我们在二维空间中进行讨论。八叉树的构造是一个递归过程。其具体步骤如下首先构造一个节点包含物体的所有 Surfel，即八叉树的根。将这一节点等分为八个节点，作为它的孩子。对八叉树中已有的节点不断进行这一剖分过程，直到节点不含有 Surfel 或已达到规定的八叉树最大深度。图 1 演示了一个二维三层八叉树的剖分过程，图 2 给出了两个模型的八叉树。在构造八叉树的同时就可以将节点分为外侧、内侧或边界。对每一个空节点，即不含 Surfel 的节点，根据距其最近的 Surfel 法向就可以判断其相对位置。若其最近 Surfel 的法向指向该节点，则为外侧节点，反之，则为内侧节点。形式化的表示，记某空节点位置为 $\mathbf{c}_e$，其在某一维上的非空邻居节点的位置为 $\mathbf{c}_n$，最近 Surfel 的法向为 $\mathbf{n}_s$，则若$(\mathbf{c}_n-\mathbf{c}_e)\mathbf{n}_s>0$，节点被判定为内侧节点，否则为外侧节点。

生成某一空节点时可能的情况有三种：a)空节点只有一个非空邻居；b)空节点有两个或两个以上非空邻居；c)空节点没有非空邻居，见图 3。第一种情况只需要考虑唯一的邻居就可以了。第二种情况，只考虑多个邻居中的一个。第三种情况，先对其空邻居进行判断，再将同样的值赋给该节点即可。由于八叉树中至少有一个非空节点，因此所有空节点最终都可以被划分。所有非空节点都是边界节点。

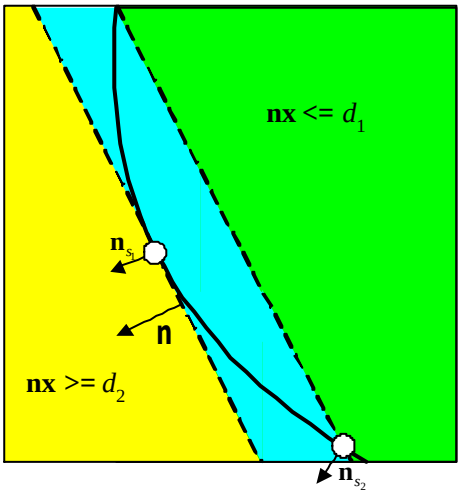


图 4．细分边界节点

3.2 细分边界节点

对于边界节点还要进一步进行划分。在一个边界节点中构造两个平面，将该节点分为三

个更小的子部分，分别对应外部、边界和内部。具体做法如下：令 $\mathbf{n} = \sum \mathbf{n}_s / \left\| \sum \mathbf{n}_s \right\|$ 为该节点内所有 Surfel 的平均法向量。定义两个平面 P_1: $\mathbf{n} \bullet \mathbf{x} - d_1 = 0$ 和 P_2: $\mathbf{n} \bullet \mathbf{x} - d_2 = 0$，其中 $d_1 = \min(\mathbf{n} \bullet \mathbf{x}_s)$，$d_2 = \max(\mathbf{n} \bullet \mathbf{x}_s)$，max 和 min 都是对接点中所有 Surfel 而言。记取到 d_1 和 d_2 的两个 Surfel 分别为 s_1, s_2。易见节点中的所有 Surfel 都位于平面 $\mathbf{n} \bullet \mathbf{x} - d_1 = 0$ 和 $\mathbf{n} \bullet \mathbf{x} - d_2 = 0$ 所夹的区域内，见图 4。而平面的另一侧为空，可根据 $\mathbf{n} \cdot \mathbf{n}_{s_1}$ 和 $\mathbf{n} \cdot \mathbf{n}_{s_2}$ 的符号被分为内侧或外侧。

当八叉树的深度足够时这种细分是合理的，因为同一节点内的 Surfel 的法向都大致相同，且近似分布在一个小平面上。

表 1．Boolean 运算保留部分

运算	*A* 保留的部分	*B* 保留的部分
$A \cap B$	在 *B* 外部	在 *A* 外部
$A \cup B$	在 *B* 内部	在 *A* 内部
A-B	在 *B* 外部	在 *A* 内部
B-A	在 *B* 内部	在 *A* 外部

3.3 内外检测算法

内外检测算法是[1]中最重要的创新点。对于给定的两个经过八叉树剖分的模型 A 和 B 以及运算符（交，并，差），分别判断两个模型的非空叶节点和另外一个模型的关系，最后根据 Surfel 的位置和所作的 Boolean 运算就可以决定要保留那些 Surfel，丢掉那些 Surfel，见表 1。

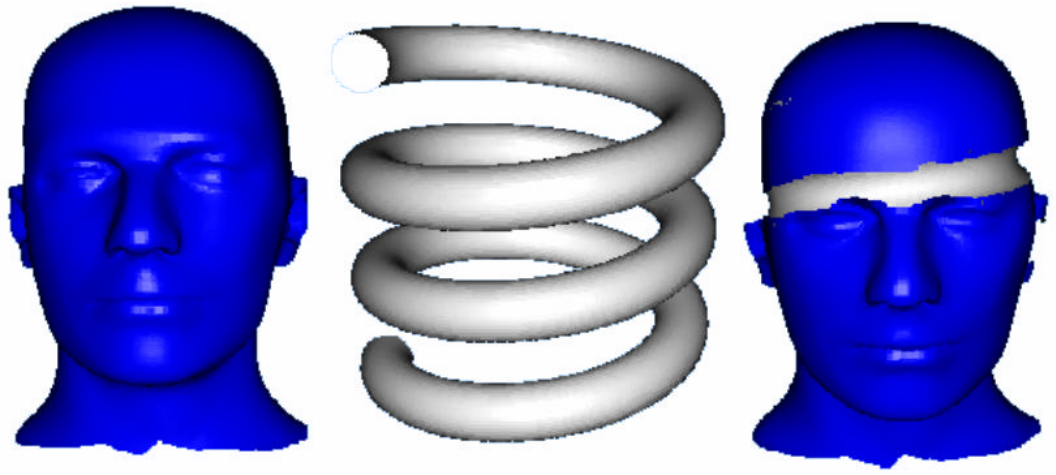


图 5．差运算例子

首先从两个八叉树的根节点开始检测是否相交。在测试相交的时候，我们将每个包围盒都向外扩张一个长度 $r_{\max}$（最大的 surfel 半径），以便不漏掉边界处的 surfel。在试验中，我们实现了 AABB[4]算法进行包围盒求交测试（如果采用 OBB[4]算法，可能会进一步提高速度）。求交测试中，可能会出现下面四种情况：

1. 包围盒不相交
2. 包围盒仅和另一个模型的外部节点相交
3. 包围盒仅和另一个模型的内部节点相交
4. 包围盒至少和另一个模型的一个边界节点相交

一个包围盒不可能同时和另一个模型的内部，外部节点相交，所以，情况 1 和 2 可以直接判断包围盒和另一个体不相交。他包含的所有 surfel 分为在外部。情况 3 也不发生相交，包围盒包含的所有 surfel 分为在内部。对于第 4 种情况，不能直接得出结论，需要递归的判断该节点的所有非空子节点和另一个模型的关系。如果到达了叶节点还是出现第 4 种情况，我们对该叶节点的每个 surfel 相对于另一个体做内外测试。

很显然，如果一个 surfel s 落入了另一个模型的内部节点，则分为在内部；如果落入了外部节点，则分为在外部；如果落入了边界节点（叶节点），则进一步和它的两个切分平面作判断。如果 s 落入平面 P_1 的负侧并且 $\mathbf{n}\cdot\mathbf{n}_{s_1} > 0$，则分为在内部，否则分为在外部；采用同样的方法对落入平面 P_2 正侧的 s 进行分类。最后一种情况就是 s 落在平面 P_1 和 P_2 之间，我们找到 s 的最近邻 surfel t，这里我们实现了 ANN[5]算法（论文中采用 TINN[6]）。如果 s 和 t 相交，则 s 分为相交；如果 s 与 t 不交，且 s 在过 t 的平面负侧，$\mathbf{n}_t\bullet(\mathbf{x}_s-\mathbf{x}_t)<0$，则分为在内部；否则分为在外部。整个算法的伪代码如下：

```
InsideOutsideTest(node A, node B){
    if (B does not intersect with A)
        classify all surfels of B as OUTSIDE
    else if (B only intersects with exterior cells of A)
        classify all surfels of B as OUTSIDE
    else if (B only intersects with interior cells of A)
        classify all surfels of B as INSIDE
    else
        if (B not a leaf node)
            for each non-empty child C of B do:
                InsideOutsideTest(A, C)
        else
            for each surfel s of B do:
                InsideOutsideForSurfel(A, s)
}
InsideOutsideForSurfel(node A, surfel s){
    if (s outside A) classify s as OUTSIDE
    else if (s in exterior cell of A) classify s as INSIDE
    else if (s in interior cell of A) classify s as INSIDE
    else
        if ( x_s · n − d_1 + (r_max + r_s) ≤ 0 )
            if ( n · n_{s_1} > 0 ) classify as INSIDE
            else classify as OUTSIDE
        else if ( x_s · n − d_2 − (r_max + r_s) ≥ 0 )
            if ( n · n_{s_2} > 0 ) classify as OUTSIDE
            else classify as INSIDE
        else
            surfel t = nearest neighbor of s in A
            if (s and t do not intersect)
```

if ($\mathbf{n}_t \cdot (\mathbf{x}_s - \mathbf{x}_t) > 0$) classify *s* as OUTSIDE

else classify *s* as INSIDE

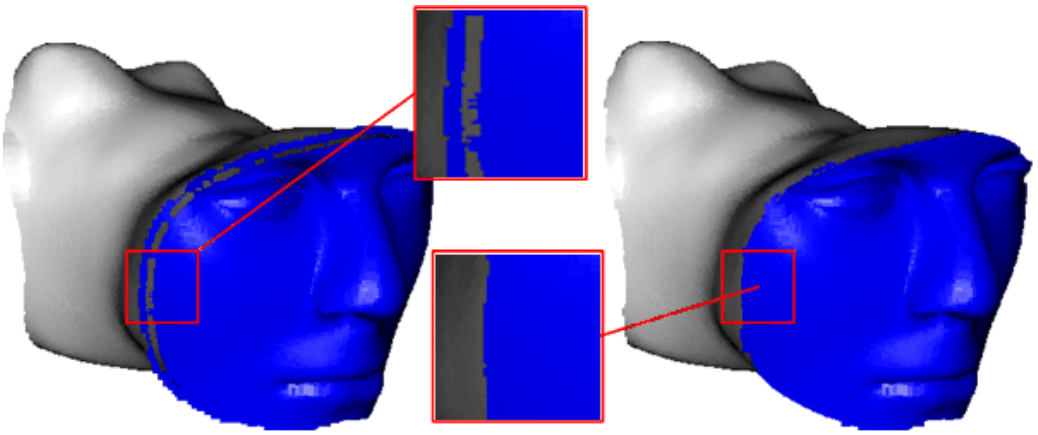
else classify *s* as INTERSECTIN

}

上述过程需要对模型 A 和 B 分别相对对方分类，最后根据表 1 进行取舍，得到结果模型。如图 5 给出了一个人头模型与螺丝模型做差的结果。

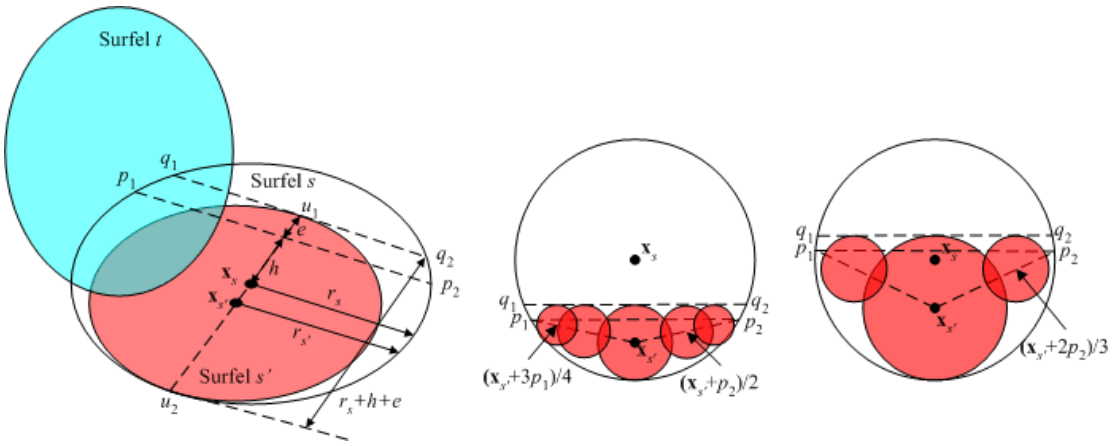
4．重采样算子

在内外检测的结果中，两个被处理物体 A 和 B 都有一些 Surfel 被分类为“相交”。这些 Surfel 处于两个物体交汇的边界上，如果简单的将它们抛弃掉，则会产生很多空洞。如果原封不动的保留下来，就会产生所谓的“参差边缘”效果，见图 6.a。这是由于这些 Surfel 原有的半径过大，使得圆盘有些部分处于物体内侧，有些部分处于物体外侧。直观的解决方法是，将边界上的 Surfel 划分成更小的 Surfel，重新进行内外检测，去掉那些不需要的。这一过程就是“重采样”。具体过程如图 7.a 所示。设 A 物体某一边界上 Surfel 为 *s*，与其有交且距离最近的 B 物体上的 Surfel 为 *t*，*s* 和 *t* 所在平面的交线被 *s* 圆盘所截的线段为 p_1p_2。根据所作的 Boolean 运算和 Surfel 的法向可以确定要保留 *s* 的那半部分，不失一般性假设保留较大的部分。也就是说要抛弃较小的部分，并且用若干个新的 Surfel 代替较大的部分。定义覆盖因子 *e*，将 p_1p_2 向保留部分的异侧平移 *e*，生成一条新的弦 q_1q_2。设 *s* 垂直于 q_1q_2 的直径与 q_1q_2 交于点 u_1，与要保留部分对应的圆弧交于点 u_2。以 u_1 u_2 为直径做一个新的 Surfel *s'*，代替 *s*。设 *s* 的圆心 $\mathbf{x}_s$ 到 p_1p_2 的距离为 *h*，则 *s'* 的圆心 $\mathbf{x}_{s'}$ 为 $(u_1+u_2)/2$，半径 $r_{s'}$ 为 $(r_s+h+e)/2$，其中 r_s 为 *s* 的半径。当 $r_{s'}$ 较大时用 *s'* 代替 *s* 就足够了，但 $r_{s'}$ 较小时还需要在 *s* 两侧再追加若干更小的 Surfel。试验中，若 $r_{s'}<r_s/2$ 时，我们补上 4 个更小的 Surfel，若 $3\,r_s/4<r_{s'}<r_s/2$ 时，补上 2 个 Surfel，见图 7.b 和图 7.c。重采样算子可以很好的消除“参差边缘”，图 6.b 给出了一个重采样后的例子。



(a) (b)

图 6．重采样前后



(a) (b) (c)

图 7．重采样算子

5．平滑算子

当对物体 *A* 和 *B* 求并时，两物体相交部分可能产生非常突然的变化。在某些情况下我们不希望有这种突变，而是 *A* 的部分到 *B* 的部分之间有比较自然的过渡，这就是平滑算子的作用。平滑算子主要有两个步骤：a)**将相交的** Surfel **旋转平移到新的位置；**b)**将相交** Surfel **周围的非相交** Surfel **旋转平移到新的位置，见图 8。**

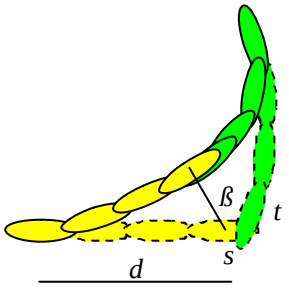


图 8．平滑算子

假设某相交 Surfel **为 *s*，其在另一物体上的最近邻为 *t*。第一步中，首先对 *s* 进行旋转：**

$$\mathbf{n}_s \leftarrow \frac{\mathbf{n}_s + \mathbf{n}_t}{\|\mathbf{n}_s + \mathbf{n}_t\|},$$

再在新的法向上平移一段距离 *ß*：

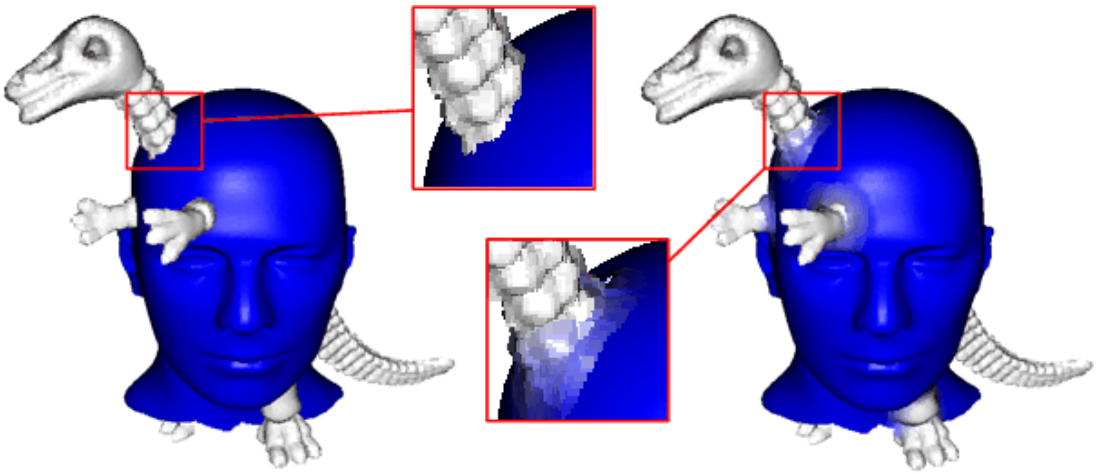
$$\mathbf{x}_s \leftarrow \mathbf{x}_s + b \cdot \mathbf{n}_s \text{。}$$

第二步，对某一非相交 Surfel *s'*，**假设离其最近的相交** Surfel **为 *s*，且距离为 d_s。若 d_s 小于某个给定的阈值 *d*，则对 *s'* 进行如下旋转平移操作：**

$$\mathbf{n}_{s'} \leftarrow \frac{a_{s'} \cdot \mathbf{n}_{s'} + (1 - a_{s'}) \cdot \mathbf{n}_s}{\|a_{s'} \cdot \mathbf{n}_{s'} + (1 - a_{s'}) \cdot \mathbf{n}_s\|}$$

$$\mathbf{x}_{s'} \leftarrow \mathbf{x}_{s'} + (1 - a_{s'}) \cdot b \cdot \mathbf{n}_{s'}$$

其中 $a_{s'} = d_s/d$。参数 *d* 和 *ß* 是由用户自己定义的，决定了平滑算子的最终效果。平滑一般只在并运算时才使用，并且不和重采样算子同时使用。图 9 给出了一个平滑算子的例子。其中颜色的过渡也是采用类似的方法实现的。



(a)平滑运算前　　(b)平滑运算后

图 9．平滑前后对比

6. 算法局限

[1]中的算法存在若干限制，使其不能适用于所有情况。第一个限制是八叉树剖分过程只有在物体是完全闭合时才得出正确的结果。对于下部开口的人头模型，在判断八叉树页节点内外位置关系时会出现错误，如图 10 所示。开口下放几个由红点标出节点本应分为外部节点，但由于其上方的邻居是内部节点，而使得它被分为内部节点。这种错误可以通过事先对模型进行处理，保证其封闭有界就可以避免。

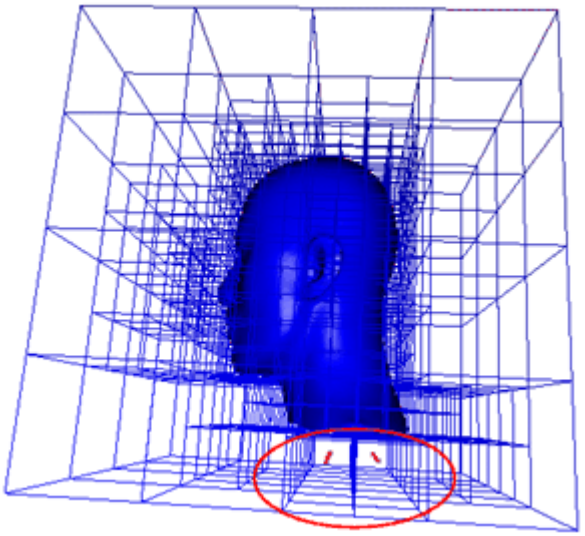


图 10．八叉树剖分的错误例子

第二个限制要更致命一些。在对边界八叉树节点进行进一步细分时，[1]中假设在节点内部物体表面可以用一个小平面近似，这一假设实际上并不容易满足，见图 11。人头耳朵的一部分被包含在一个节点内，显然这部分表面无法用一个平面很好的近似。这时再用 3.2 节中的算法进行内外检测就会出现错误。解决这一问题的一个方法是加深八叉树的深度，使得每个节点更小，进而满足局部线性的要求；另一个解决方法是用一个曲面近似节点内部的表面。但无论哪一种解决方案都要引入更多的计算。

7．结论

本次试验我们基本上实现了文献[1]中提出的给予 Surfel 表示的 Boolean 运算方法。但在运行速度上并没有达到文献[1]中提到的实时交互效果。就其原因可能有两点：1）由于时间精力有限，我们对算法本身所作的优化还很不够。计算几何和图形学领域的许多问题都存在优化算法，我们所作的系统中设计到了许多这样的问题，如求最近邻。这些子问题如何解决在文献[1]中并没有详细给出，因此在具体实现上我们肯定与[1]中的工作有很大差距。2）[1]中所用的试验机器配置了 GeForce4 MX 图形卡，这也可能是我们的性能较差的一个因素。总之如果想在这篇文章的基础上有所发展或改进，还需要更进一步的更细致的工作。

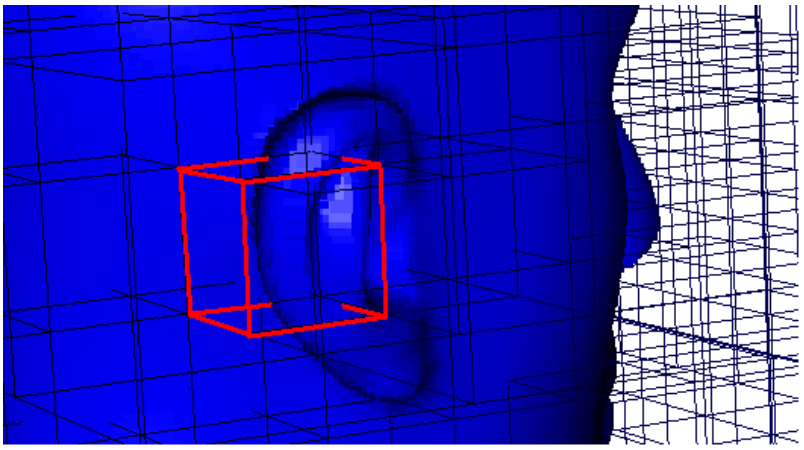


图 11．节点内无法用平面近似的例子

参考文献

[1] Bart Adams and Philip Dutré, “Interactive Boolean Operations on Surfel-Bounded Solids”, SIGGRAPH 2003, Annual Conference, San Diego, USA, 26-31, Jul 2003.

[2] Bart Adams and Philip Dutré, “A Smoothing Operator for Boolean Operations on Surfel-Bounded Solids”, Technical Report CW 359, Apr 2003.

[3] Hans Peter Pfister, Matthias Zwicker, Jeoren Van Baar, and Markus Gross, “Surfels: Surface Elements as Rendering Primitives”, In Proceedings of SIGGRAPH 2000, 2000.

[4] Tomas Moller, Eric Haines. “Real-Time Rendering”. Printed in the USA.

[5] http://www.cs.umd.edu/~mount/ANN/

[6] GREENSPAN, M., GODIN, G., AND TALBOT, J. 2000. “Acceleration of binning nearest neighbor methods”. In Proceedings of Vision Interface2000, 337-344.