$\det(A)=1$。

2.二次变形

线性变换只能使对象表现出剪切和拉伸变形，为了使对象能进行扭曲和弯曲变形，构造二次变换矩阵。通过下面的式子定义二次变换矩阵并求粒子的目标位置

$$g_i=\begin{bmatrix}A & Q & M\end{bmatrix}\tilde{q}_i \tag{3-43}$$

其中 $g_i\in R^3,\tilde{q}=\left[q_x,q_y,q_z,q_x^2,q_y^2,q_z^2,q_xq_y,q_yq_z,q_zq_x\right]^T\in R^9,A\in R^{3x3}$ 中是线性项的系数，$Q\in R^{3x3}$ 是完全二次项的系数，$M\in R^{3x3}$ 是复合项的系数。现在要使式（3-44）的值最小，

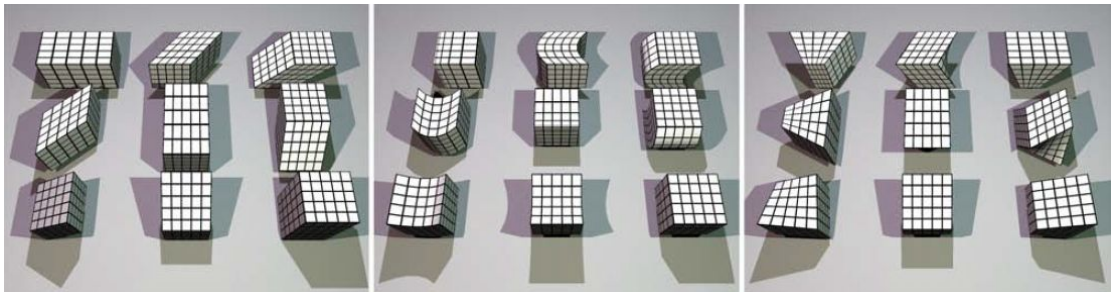
$$\sum_i m_i\left(\tilde{A}\tilde{q}_i-p_i\right)^2 \tag{3-44}$$

经计算得最优二次变换矩阵为

$$\tilde{A}=\left(\sum_i m_ip_i\tilde{q}_i^T\right)\left(\sum_i m_i\tilde{q}_i\tilde{q}_i^T\right)^{-1}=\tilde{A}_{pq}\tilde{A}_{qq} \tag{3-45}$$

类似于前述的线性变形情况，利用 $\beta\tilde{A}+(1-\beta)\tilde{R}$ 代换原来的 R，其中 $\tilde{R}\in R^{3x9}=\begin{bmatrix}R & 0 & 0\end{bmatrix}$。

利用改进后的积分式可以得到的线性和二次变形效果如图 3.8。



（a）线性变形　　（b）弯曲变形　　（c）综合变形

图 3.8 变换矩阵变形后的应用效果

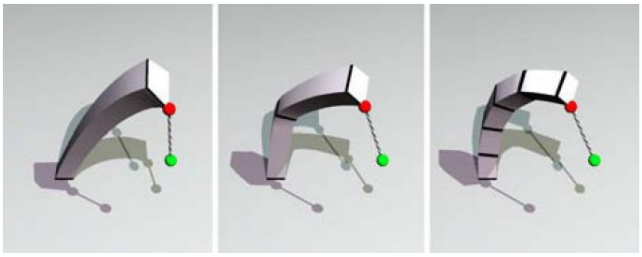
3.基于簇的变形

为了进一步扩展变形的幅度，可以将粒子划分为簇，彼此之间存在交叠部分。对于体模型对象，将模型中的各个顶点作为粒子，将与同一个单元（如四面体）相邻接的粒子作为一个簇。如果我们使用面模型，我们将面模型内部划分为彼此有重叠部分的立方体，将每个立方体包含的所有顶点作为一个簇。

在每个积分步，每个簇独立的用其自身的初始形状匹配其实际形状，然后将求出的变换矩阵和平移向量应用于这个簇中的所有粒子，单独计算每个簇的变形，整合在一起形成总体的变形。

图 3.12 显示了三个拥有 60 个质点但划分为不同数量簇的柱形体二次变形情况，三个图中划分的簇数依次为 1、2 和 5，施加的力的位置、方向相同，从图中可看出划分的

簇数越多，变形细节体现的越好，变形幅度越大。



（a）1 个簇　　（b）2 个簇　　（c）5 个簇

图 3.9 不同簇数对对象变形灵活性的影响

3.4.5 实验

文中方法的时间性能主要取决于形状匹配的粒子的数量和划分的簇数。为了定性说明进行如下实验，对一定数量的粒子随机安排其初始位置及目标位置，然后选取不同的粒子数，不同的簇数，进行线性变形和二次变形实验。图 3.13 显示了实验结果，从图中可以看出不论采用哪种方案，计算复杂度与参与形状匹配的质点数都是呈线性关系。同时划分簇数越多，因为需要的极分解越多，所以计算代价也越大。此外，二次变形比线性变形复杂，计算代价也相应增加。在实验中，100 个包含 100 个参与形状匹配粒子，划分成 8 个簇的对象进行二次变形 fps 值可以达到 50。

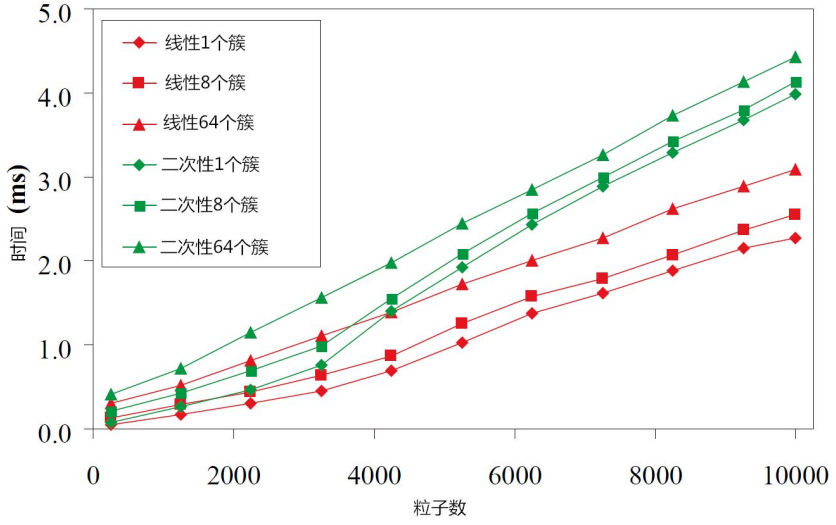


图 3.10 不同实验方案下参与形状匹配粒子数与每一帧计算耗费时间关系

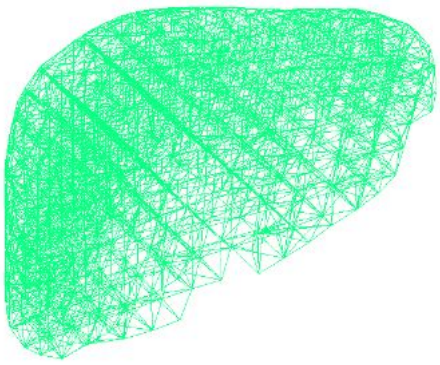
3.5 物理建模方法对比实验

对物理建模方法效果的评价主要集中在实时性、逼真度、稳定性方面。我们利用第二章建立的肝脏体模型分别针对质点-弹簧、有限元和形状匹配无网格法进行变形实验。实验硬件环境为个人电脑，配置为 CPU：Intel Core i3-2100 @ 3.1GHz、GPU：NVIDIA

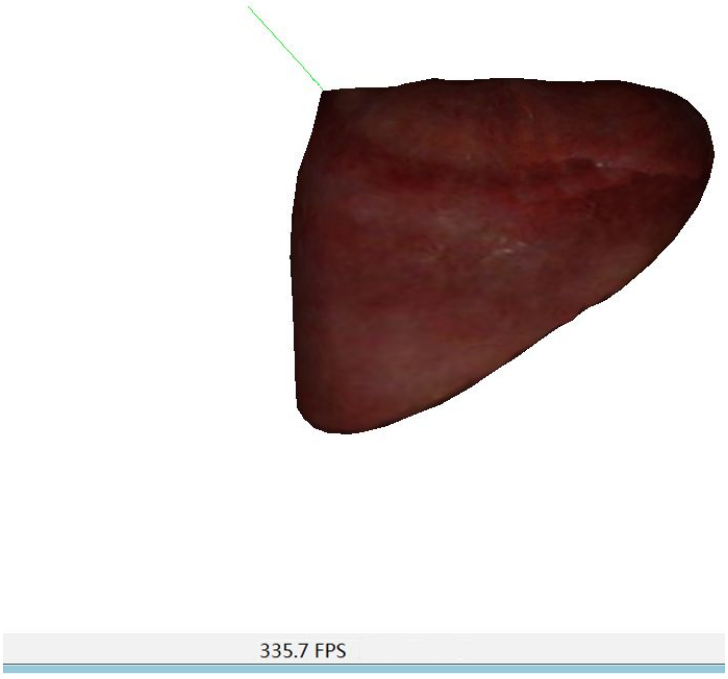
GeForce GTS450、内存：6G。软件环境：VS2010 集成开发环境，C++和 OpenGL 编程语言。

1.实时性对比

质点弹簧刚度系数设为 180，阻尼系数设为 5，弹簧使用四面体形式的连接方式；有限元方法的弹性模量设为 3000，泊松比设为 0.3，单元使用四面体单元；基于形状匹配的无网格方法的簇数设为 50，区域半径设为 1，参与形状匹配粒子数为 96，α 设为 0.72，β 为 0.63，使用二次变形方法。参数进行如上设置后可以比较好地模拟肝脏的变形，实验截图如下：

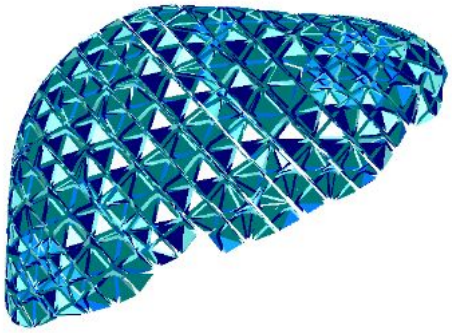


（a）弹簧连接网格

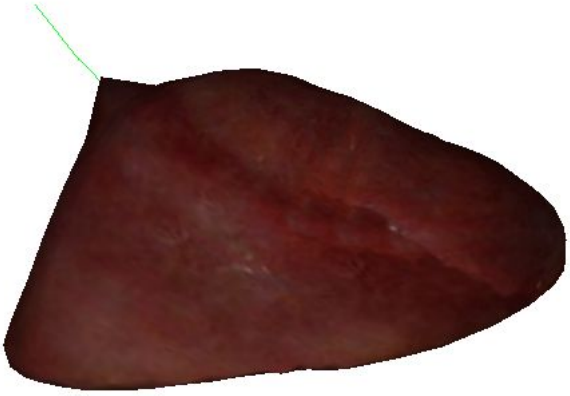


（b）拉伸变形效果及 FPS 值

图 3.11 质点--弹簧模型方法效果

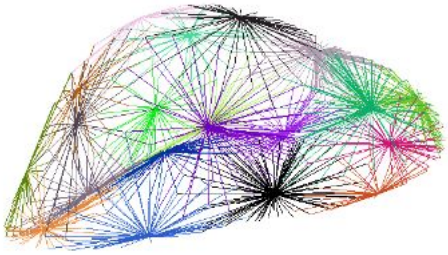


（a）有限元四面体单元剖分

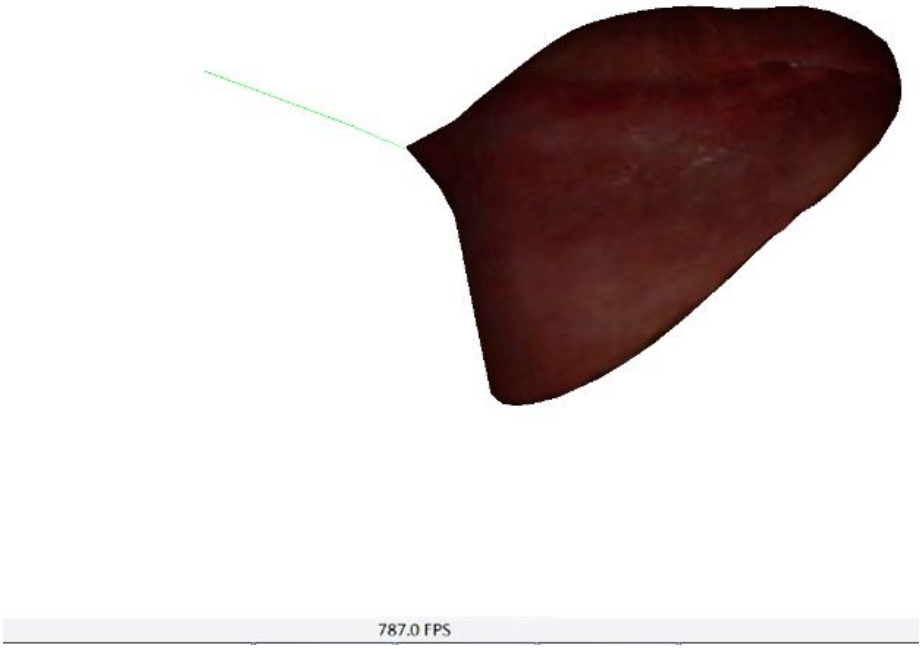


（b）拉伸变形效果及 FPS 值

图 3.12 有限元方法效果



（a）簇划分



（b）拉伸变形效果及 FPS 值

图 3.13 基于形状匹配无网格方法效果

从实验结果可看出，在具有相似的变形逼真度的情况下，有限元法的 FPS 值为 36.5，质点-弹簧法的为 335.7，而形状匹配无网格法的为 787，本文算法的实时性占很大优势。

2.稳定性、大变形对比

在实验过程中使用本文形状匹配无网格法对肝脏无论如何变形，变形效果在视觉上都不会出现畸变，并且都能恢复原形，在上面的原理介绍中也在理论上说明了其无条件稳定性。

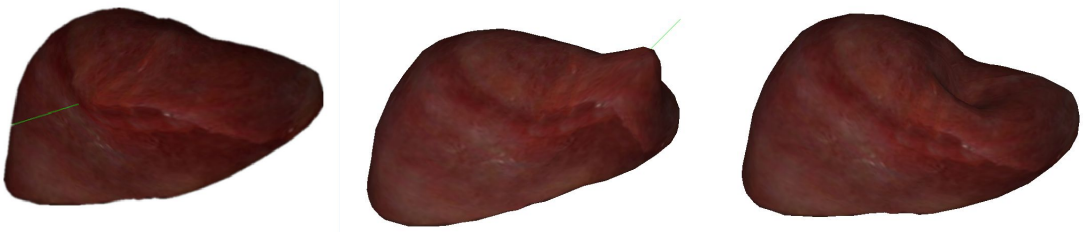


图 3.14 形状匹配无网格任意变形效果

而质点弹簧模型在大变形情况下出现变形效果不真实，并且无法恢复，如图

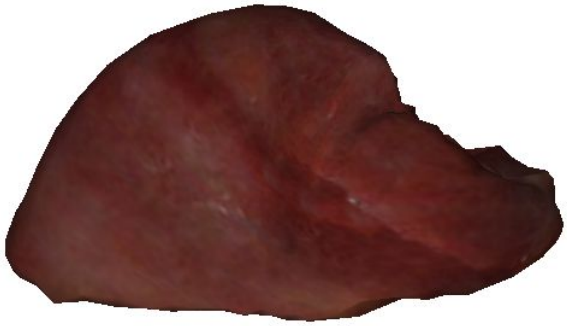


图 3.15 质点-弹簧方法变形失真并且无法恢复

由于有限元模型是基于小变形假定下的方法，不适用于大变形情况。所以形状匹配无网格法在大变形、稳定性方面有优势。

综上所述，本文基于形状匹配的无网格方法在实时性、稳定性和大变形适用性方面表现很好。

3.6 本章小结

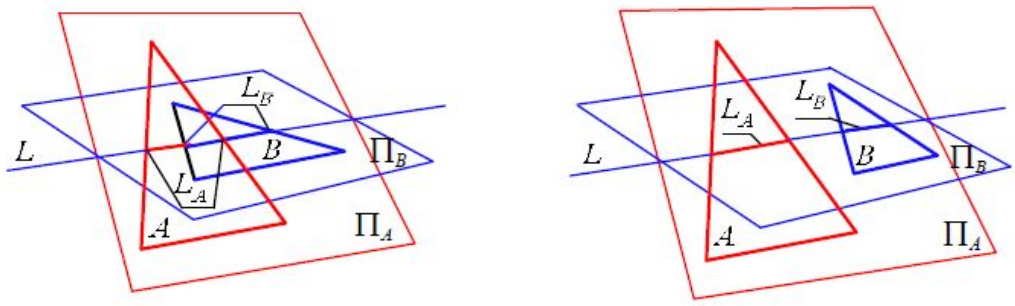
本章主要对软体物理建模方法做了研究。先对基于网格的质点-弹簧模型和有限元模型做了介绍，然后重点对本文选用的基于形状匹配的无网格方法做了阐述。虽然此模型是基于几何驱动而不是基于物理属性驱动的，但其在表现变形的逼真度方面的能力与质点-弹簧方法还是相当的。相对于质点-弹簧模型和有限元模型方法，其原理简单、易实现，不需要网格划分，不需要复杂的数据结构，通过对比实验说明，本文方法的仿真效率很高；同时算法针对任何的变形操作都具有无条件稳定性，适用于大变形。

也是碰撞检测中很关键的环节，其影响着碰撞检测的精度与实时性。

4.4.1 常用三角形间相交测试算法

三角形间相交测试算法发展到现在已经有很多种，大体上可以归为两类，一是标量判别类型，另一类是矢量判别类型。

标量判别类型，由 Möller, Held 和 Tropp 等人提出的算法是其中的典型代表。Möller 算法[55]：如图 4.12 所示，判断三角形 A 和 B 是否相交。首先，计算三角形各自所在的平面 Π_A 和 Π_B；然后，进行相离排除，计算三角形 A 的三个顶点是否在 Π_B 的同一侧，如果在同一侧则三角形 A 和 B 肯定相离，同理判断三角形 B 的三个顶点是否在 Π_A 的同一侧（判断方法是求取点到平面的距离，根据距离的正负号判断点在平面的哪一侧）；如果不能排除二者相离的可能性，则求取 A 与 Π_B 的交线 L_A，B 与 Π_A 的交线 L_B，L_A 与 L_B 肯定共线，最后，判断 L_A 与 L_B 是否有重叠部分即可判断，存在重叠，三角形相交，（图 4.12(a)）,否则分离，（图 4.12(b)）。



（a）相交　　（b）相离

图 4.12 Möller 三角形相交测试算法

Held 算法与 Möller 算法相似，都是从几何观点出发，把三维问题降维为二维问题，解决三角形相交测试问题。而 Tropp 算法另辟蹊径，从代数观点出发，利用求解线性方程组解决此问题。

矢量判别类型，由 Devillers&Guigue、Shen 等提出的算法是其中的典型代表。Devillers&Guigue 算法基于三角形顶点组成的行列式的几何意义，Shen 提出的算法基于分离平面。

一般而言，标量判别法可以直接获取相交点的坐标，但是由于计算累积误差的存在，其精度受到一些影响；矢量判别法相对来说在检测结果的判定上更健壮，因为其判定不依赖于计算精度，但是如果想要获取碰撞的详细信息需要做额外的处理。然而，以上方法都存在这样的缺陷，在算法的稳定性方面都只是在代数角度上进行了说明，没有在理论上阐述针对奇异情况的处理。同时，在算法的评估上偏重于速度，没有对算法的稳定性进行分析测试，而在实际应用中针对奇异情况的处理失误有可能使系统崩溃。

针对上述算法稳定性方面的问题，于海燕等人[56]提出了基于投影理论的三角形间相

交测试方法。算法关注了两三角形间相对位置的奇异情形，根据投影降维思想转化为平面上的几种简单情形，在理论上对几何奇异情形（如点碰撞、线碰撞等）对算法稳定性的影响进行了说明。所以本文选用此方法作为三角形间的相交测试方法，然而该算法还不尽完善，本文在投影分类，奇异情况处理、分离情况排除和三角形线裁剪算法方面对算法进行改进以提升其实时性、稳定性和准确性，然后应用于本文系统。

4.4.2 基于投影降维的三角形间相交测试算法

如图 4.13，两三角形 A 、 B 。

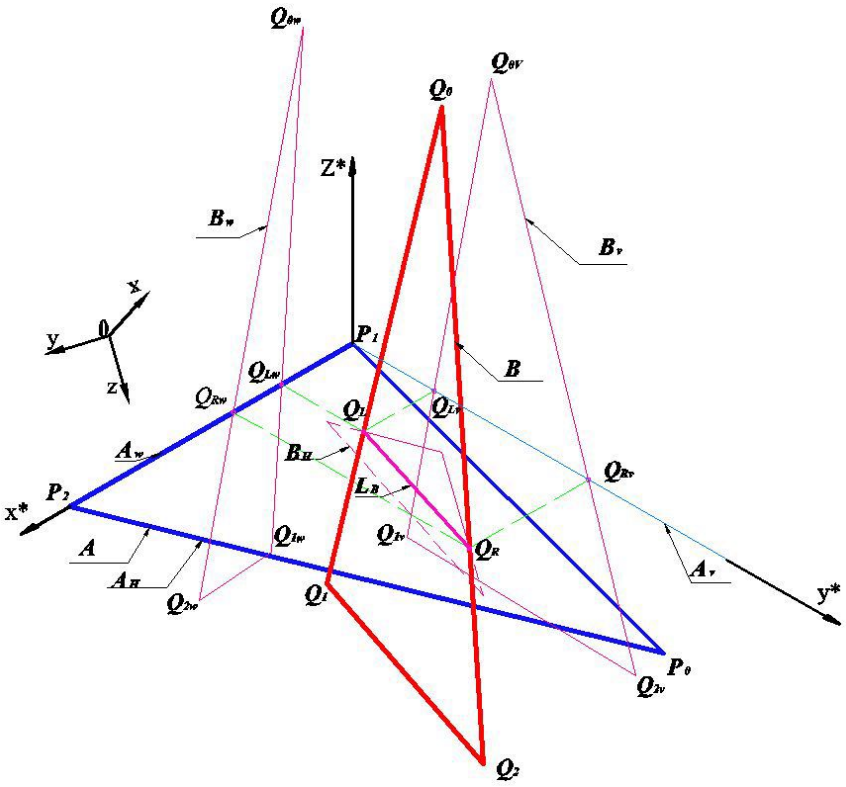


图 4.13 计算坐标系下的三角形

图中标记符号的含义如下：xyz 和 $x^*y^*z^*$ 分别代表世界坐标系和计算坐标系，我们将 $x*-y^*$ 平面记为 H 平面，$x*-z^*$ 平面记为 W ，$y*-z^*$ 记为 V 。P_0 、P_1 、P_2 、Q_0 、Q_1 、Q_2 分别是 A 、B 的顶点；Q_{0v} 、Q_{1v} 、Q_{2v} 、Q_{0w} 、Q_{1w} 、Q_{2w} 分别是三角形 B 的顶点在 V 、W 平面上的投影；A_V 、A_H 、A_W 、B_V 、B_H 、B_W 分别是三角形 A 、B 在 V 、H 、W 平面上的投影；L_B 是三角形 B 与 H 平面的交线；Q_L 、Q_R 是 L_B 的端点；Q_{LV} 、Q_{RV} 、Q_{LW} 、Q_{RW} 分别是 Q_L 、Q_R 在 V 、W 平面上的投影。

我们知道，如果两三角形有相交点，那它们在三个坐标平面的投影肯定都会存在交点。若两三角形在三个坐标平面的投影存在没有交点的情况，则两空间三角形必定不相交 ，基于投影降维三角形相交检测改进算法基本原理就在于此。

1.坐标系转换

一个合适的坐标系可以简化几何表述与代数运算，首先建立一个计算坐标系，方法与原算法类似。

建立计算坐标系：

如图 4.13，以三角形 A 所在的平面作为 $x*-y^*$ 平面，此平面的法向作为 z^* 轴，并以 A 的一条边（如 P_1P_2 边）作为 x^* 轴。

（1）取 P_1P_2 向量的单位向量作为 $n_1(a_1,a_2,a_3)$。

（2） 对向量 P_1P_2， P_1P_0 求叉乘可以得到一个垂直于三角形 A 的向量，取单位向量为 $n_3(c_1,c_2,c_3)$。

（3） 利用 n_3 x n_1 然后取单位向量可得 $n_2(b_1,b_2,b_3)$。

以 P_1 为原点，以 n_1、n_2、n_3 为轴向建立计算坐标系 $x^*y^*z^*$。

$$\begin{pmatrix}\boldsymbol{n_1} & \boldsymbol{n_2} & \boldsymbol{n_3}\end{pmatrix}=\begin{pmatrix}a_1 & a_2 & a_3\\ b_1 & b_2 & b_3\\ c_1 & c_2 & c_3\end{pmatrix} \tag{4-3}$$

坐标变换：

因为 P_1 为计算坐标系 $x^*y^*z^*$ 的原点，我们可以得到 xyz 坐标系到 $x^*y^*z^*$ 的平移矩阵：

$$\boldsymbol{T}=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ -P_{1x} & -P_{1y} & -P_{1z} & 1\end{pmatrix} \tag{4-4}$$

同时可以得到旋转矩阵：

$$\boldsymbol{R}=\begin{pmatrix}a_1 & a_2 & a_3 & 0\\ b_1 & b_2 & b_3 & 0\\ c_1 & c_2 & c_3 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{4-5}$$

从而坐标变换公式为：

$$(x^*,y^*,z^*,1)=(x,y,z,1)\cdot\boldsymbol{T}\cdot\boldsymbol{R} \tag{4-6}$$

$$(x,y,z,1)=(x^*,y^*,z^*,1)\cdot(\boldsymbol{T}\cdot\boldsymbol{R})^{-1} \tag{4-7}$$

将两三角形的坐标都转换到计算坐标系下。

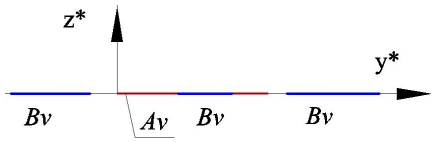
2.基于投影理论对三角形空间位置关系的分析

这一部分是算法的关键之处，也是我们对原算法改进之处。原算法中只用到了三角形在 V 面和 H 面投影，当三角形 A 与 B 不相交时，如果 B_V 与 A_V 没有交点，可以排除这种相离的情况，但如果 B_V 与 A_V 存在交点，而 B_W 与 A_W 没有交点，这种情况下显然 A 与 B 是相离的，而原算法针对此种情况无法进行前期排除，只能经过后续的一些计算后才判断出相离，显然会花费比较多的代价。针对这个问题，本文利用 V、W 两个平面的投影

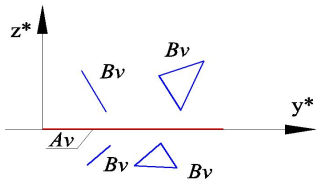
进行分离情况的排除来解决。当两三角形在 V 面上的投影情形跟图 4.14(a)(b)中所示不同时，原算法需要计算 B_V（B 的投影三角形）的三条边与 A_V 所在的坐标轴的交点，根据交点的个数不同做不同的处理，这增加了很多计算。针对这个问题，本文通过顶点交换实现 B_V 的特定边与 A_V 所在的坐标轴相交，使奇异情况处理更简单，减少计算。同时本文对三角形投影分类更加细化，使相交测试更条理。

因为计算坐标系基于三角形 A 所建，下面的结论是显而易见的：$A_H = A$；A_V 和 A_W 分别为位于 y^* 和 x^* 轴上的线段；尽管三角形 B 在三个坐标平面上的投影形状不确定，但也只有线段和三角形的可能。投影为线段的情形实际是投影为三角形的特例，针对这种情形相关参数的求解处理都按三角形处理。

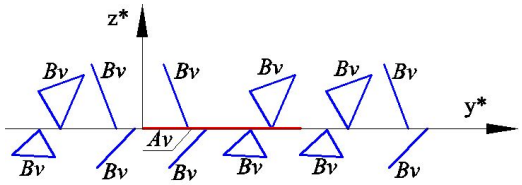
以三角形 A 和 B 在 V 平面上的投影为例（W 平面上的类似），所有可能的投影形状和相对位置关系如图 4.14 所示。



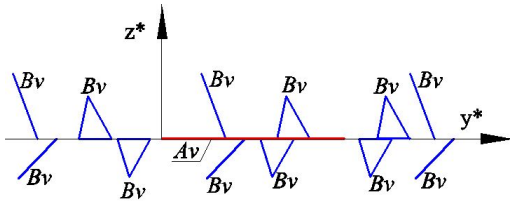
(a)



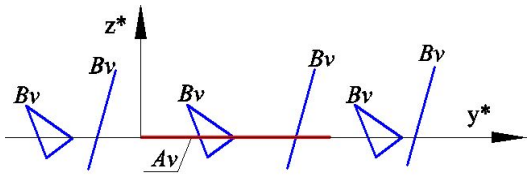
(b)



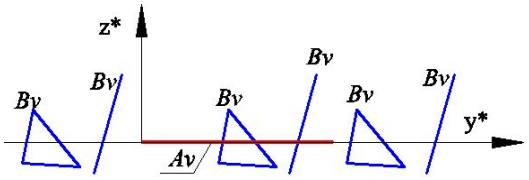
(c)



(d)



(e)



(f)

图 4.14 三角形 A 和 B 在 V 平面上的投影

（红线代表三角形 A 的投影，蓝线或蓝色三角形代表三角形 B 的投影）

投影分类及对应相交测试：

（1）如图 4.14（a）所示，B_V 是 y^* 轴上的线段，当三角形 B 所有顶点的 z^* 坐标均为 0 时出现这种情形。如果 B_V 与 A_V 没有重叠部分，则三角形 A 与 B 相离；如果有重叠部分，则继续判断 B_W 与 A_W 是否有重叠部分，没有则三角形间相离；如果有则需要判断 H 平面内二维三角形 A_H 与 B_H 是否相交，不相交则 A 与 B 相离，否则 A 与 B 相交。如何判断两共线线段是否存在重叠部分和如何判断两共面三角形是否相交的方法将在后面详述。

（2）如图 4.14（b）所示，B_V 为位于 A_V 之上或之下的线段或三角形，当三角形 B 所有顶点的 z^* 坐标均大于或小于 0 时出现这种情况。显然此种情况下三角形 A 与 B 相离。

（3）如图 4.14（c）所示，B_V 与 y^* 轴有一个交点，当三角形 B 的一个顶点的 z^* 坐标为 0 而另两个坐标同时大于或小于 0 时出现这种情况。我们把 z^* 坐标为 0 的顶点标记为 Q_0（利用坐标轮换）。显然三角形 B 与 H 平面间只有一个交点--Q_0，即 $Q_L = Q_R = Q_0$。Q_L 在 V 平面上的投影为 $Q_{LV}(0, y^*_{Q_0}, 0)$，如果 Q_{LV} 没在 A_V 上（这很容易判断），三角形 A 与 B 相离；如果在，则判断 $Q_{LW}(x^*_{Q_0}, 0, 0)$ 是否在 A_W 上，如果不在，三角形 A 与 B 相离；如果在，则需要判断在 H 平面内，Q_L 点是否在三角形 A_H 内，如果在，在三角形 A 与 B 相交于点 Q_L，是点碰撞，否则二者相离。如何判断在同一个平面内点与三角形的关系的方法在后面详述。

（4）如图 4.14（d）所示，B_V 与 y^* 轴有两个交点，当三角形 B 的两个顶点的 z^* 坐标为 0，剩余的顶点 z^* 坐标不为 0 时出现这种情况。我们把 z^* 坐标为 0 的顶点标记为 Q_0、Q_1（利用坐标轮换）。显然三角形 B 与 H 平面间有两个交点--Q_0、Q_1，即 $Q_L = Q_0$、$Q_R = Q_1$。Q_L、Q_R 在 V 平面上的投影分别为 $Q_{LV}(0, y^*_{Q_0}, 0)$、$Q_{RV}(0, y^*_{Q_1}, 0)$。如果以 Q_{LV}、Q_{RV} 为端点的线段 L_{B_V}（L_B 在 V 平面上的投影）与 A_V 没有重叠部分，三角形 A 与 B 相离；如果有，则判断 L_{B_W}（L_B 在 W 平面上的投影）与 A_W 是否有重叠部分，如果没有，三角形 A 与 B 相离；如果有，则需要判断在 H 平面内，线段 L_B 与三角形 A_H 的相交情况，如果相交，三角形 A 与 B 相交，属于线碰撞，交线为 $L_B \cap A_H$，否则二者相离。如何判断在同一个平面内线段与三角形的关系的方法在后面详述。

（5）如图 4.14（e）所示，B_V 与 y^* 轴有两个交点，当三角形 B 的一个顶点的 z^* 坐

标为 0，一个顶点 z^* 坐标大于 0，一个顶点 z^* 坐标小于 0 时出现这种情况。我们把 z^* 坐标为 0 的顶点标记为 Q_0（利用坐标轮换）。显然三角形 B 与 H 平面间有两个交点，一个是 Q_0，记 $Q_L = Q_0$，另一个是三角形 B 的边 Q_1Q_2 与 H 平面的交点，记为 Q_R。后续步骤跟 4）中相关步骤相同。关于 Q_R 的求取在后面介绍。

（6）如图 4.14（f）所示，B_V 与 y^* 轴有两个交点，当三角形 B 的一个顶点的 z^* 坐标大于 0（小于 0），另两个顶点的 z^* 坐标均小于 0（大于 0）时出现这种情况。我们把 z^* 坐标大于 0（小于 0）的那个顶点标记为 Q_0（利用坐标轮换）。显然三角形 B 与 H 平面间的两个交点一个是三角形 B 的边 Q_0Q_1 与 H 平面的交点，记为 Q_L，另一个是三角形 B 的边 Q_0Q_2 与 H 平面的交点，记为 Q_R。后续步骤参考 5）。

3.相关详细算法

（1）两共线线段相交检测

如图 4.15 所示，两共线线段 L_1（端点为 P、Q）、L_2（端点为 M、N），根据端点坐标判断，如果端点 Q 位于端点 M 左侧或端点 P 位于端点 N 右侧，两线段无重叠部分，否则，有重叠部分。

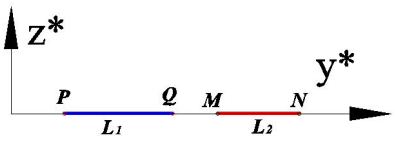


图 4.15 两共线线段

（2）同一平面内线段与三角形的相交检测

如图 4.16 所示，三角形 A（顶点 P_0、P_1、P_2），线段 L_1。二者的相交测试利用孙燮华提出的凸多边形线裁剪算法[57]解决，这也是本文与原算法不同之处，本文采用的方法更简易，同时能处理所有出现的裁剪情况。首先，求取线段 L_1 的一般方程，如式（4-8）所示；其次，将三角形顶点 $P_i(i=0,1,2,3;P_3=P_1)$ 的坐标代入式（4-8）求取特征值 e_i；再次，计算线段与三角形的交点及交点数量，下面所示的伪代码表述了这一求解过程；最后，如果得到的交点数为 0 或者是 1 个但交点不在 L_1 线段上或者 2 个但交点组成的线段 $L_{Q_1Q_2}$ 与线段 L_1 没有重叠部分，线段 L_1 与三角形 A 相离，否则相交。利用上述方法针对图 4.16 中所示的线与三角形的相交情况都能判别出。

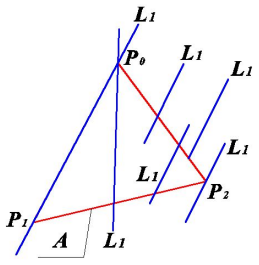


图 4.16 平面内线段与三角形位置关系

$$ax + by + c = 0 \tag{4-8}$$

线段与三角形的交点及交点数量求取伪代码：

```
int n = 0;//counts of intersection points
Point Q1, Q2, Q;//define two intersection points
float e[4];
if (e[0] == 0)
{
    n++;
    Q1 = P[0]; //P[i]is the vertex of triangle
}
for (i = 0; i <= 3; i++)
{
    if (e[i + 1] == 0)
    {
        n++;
        if (n == 1)
            Q1 = P[i + 1];
        if (n == 2)
        {
            Q2 = P[i + 1];
            break;
        }
    }
    else if (e[i] * e[i + 1]<0)
    {
        n++;
        Q = CalInterPoint();//calculate intersect point
        if (n == 1)
            Q1 = Q;
        if (n == 2)
        {
            Q2 = Q;
            break;
```

```
        }
    }
}
```

（3）同一平面内两三角形的相交检测

假设有两三角形 A、B 在同一平面内，首先，利用上述 2）中的方法判断 B 的边中是否存在与三角形 A 相交的，如果存在，则两三角形相交；如果不存在，不能得出二者相离的结论[58]，如图 4.17 所示的情形，三角形 A 位于 B 之内，显然二者是相交情况，这需要判断 A 的顶点是否存在位于 B 之内的，如果有则三角形 A 位于 B 之内，否则，A 与 B 相离。

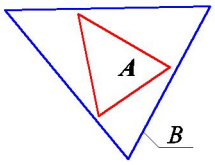


图 4.17 同一平面内三角形 A 在三角形 B 内部

（4）同一平面点与三角形的位置关系

如图 4.18 所示，有一点 P 和三角形 A 在同一平面内。为了计算方便，对点 P 和 A 的顶点进行平移变换，使点 P 位于原点。如果 P 位于 A 之内，则 P 一定位于 A 各边的同一侧（左侧或右侧）否则，P 位于 A 之外。详细计算参考文献[52]。

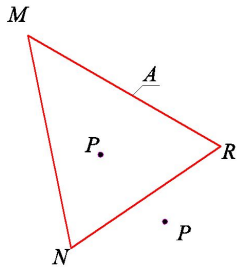


图 4.18 同一平面点与三角形的位置关系

（5）线与面交点计算

如图 4.13 所示，三角形 B 的 Q_0Q_1 边与 H 平面相交于点 Q_L，Q_0Q_1 在 V 平面的投影 $Q_{0V}Q_{1V}$ 与 y^* 轴的相交于点 Q_{LV}。我们可以容易的得到点 Q_L 和 Q_{LV} 的参数方程如式（4-9）所示，显然 t 与 t_V 相等。

$$\begin{aligned} Q_L &= Q_1 + t(Q_0 - Q_1) \\ Q_{LV} &= Q_{1V} + t_V(Q_{0V} - Q_{1V}) \end{aligned} \tag{4-9}$$

由图 4.19 可得：

$$t_V = \frac{Q_{LV} - Q_{1V}}{Q_{0V} - Q_{1V}} = \frac{\left|Z^*_{Q1}\right|}{\left|Z^*_{Q1}\right| + \left|Z^*_{Q0}\right|} \tag{4-10}$$

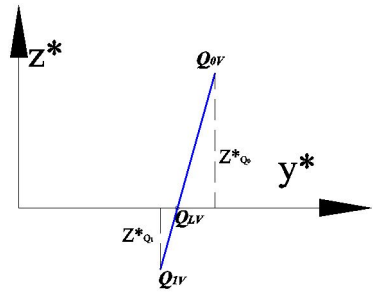


图 4.19 Q_0Q_1 在 V 平面的投影 $Q_{0V}Q_{1V}$

4.本文改进算法与原算法对比实验

实验条件同第三章，为了使算法的每一步都至少执行一次并测试算法的稳定性，本文专门设计了 40 对三角形的位置关系，包含了所有可能的三角形相交奇异情况，部分示例如图 4.20，利用原算法和本文改进后的算法对这 40 对三角形进行测试并且执行不同的次数，测试结果如表 4.1 所示。

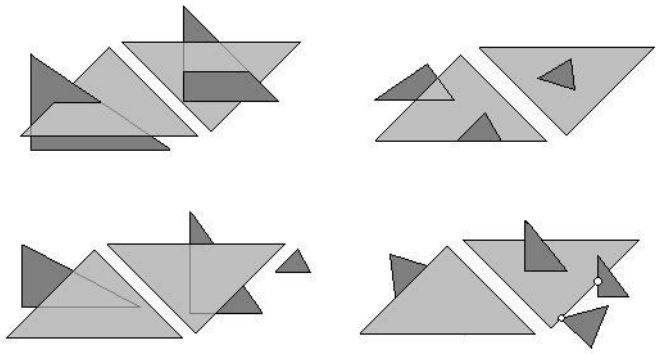


图 4.20 两三角形空间位置关系举例

表 4.1 改进算法与原算法性能对比

参加测试的三角形对数量	测试时间	
百万对	ms	
	原算法	本文改进算法
0.1	70	42
0.5	392	231
1	654	384
1.5	947	557
2	1256	739

改进后算法的时间效率提升约 41%，同时设计的三角形对的位置关系碰撞检测结果跟预设情况对比无误，并且针对奇异情况处理程序运行稳定。说明了本文改进算法在效率、准确性、稳定性方面有提升。

4.5 碰撞检测算法对比实验

实验平台与上一章相同，碰撞对象分别采用第 2 章建立的肝脏体模型和用 3ds Max 建立的手术器械，手术器械有 628 个顶点，1248 个三角形。实验方案为固定肝脏面模型，控制手术器械运动，仿真二者的碰撞过程；进行三组实验，第一组采用本文的复合层次包围盒法和基于投影降维的三角形间相交测试法方案，因为肝脏体模型表面共有 9313 个三角形，让每个叶子结点包围盒最多包围 10 个三角形单元且最多划分 11 层包围盒即可，其中上面三层用 Sphere，中间三层用 Sphere-AABB，下面五层用 AABB，手术器械三角形较少，划分为 9 层即可，上面两层用 Sphere，中间两层用 Sphere-AABB，下面五层用 AABB（这里之所以对手术器械也同样采用这样的包围盒而不是传统的只对手术器械尖端建立一个 Sphere 或 AABB 包围盒，是为了能够精确检测出手术器械与软体对象的精确接触点从而可以对软体精确施加不同方向的力，后续还可以考虑手术器械相对软体不同角度施加力对软体变形效果的影响）。第二组采用 Sphere 层次包围盒法和 Möller 三角形相交测试法方案，第三组采用 AABB 层次包围盒法和 Möller 三角形相交测试法方案。

碰撞实验截图如图 4.21。

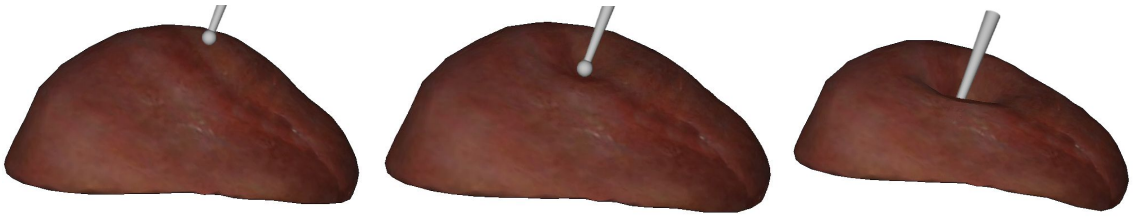


图 4.21 肝脏碰撞过程

每组实验中两对象的相对位置和手术器械的运动速度是相同的，记录每组实验中相交三角形数目相同时的 FPS 值，具体实验测试数据采样值如下。

表 4.2 不同碰撞检测方案下 FPS 值与相交三角形数量实验数据

相交三角形对数	帧速率（FPS）		
	复合层次包围盒法（Sphere-AABB）	球形包围盒（Sphere）	轴向包围盒（AABB）
0	263.2	263.2	254.7
32	260.3	259.7	251.6
97	253.6	250.1	244.8

279	217.9	211.2	214.6
534	163.2	148.4	158.9
733	129.4	95.6	119.4
925	89.6	47.3	73.1
1106	53.2	17.2	42.7

将上述表格内数据绘制成曲线图，如图 4.22.

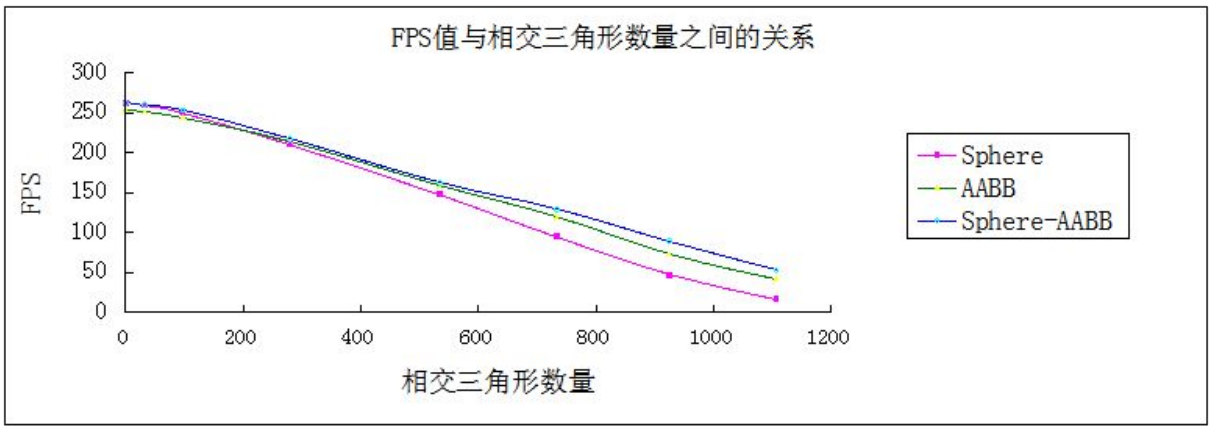


图 4.22 不同碰撞检测方法下 FPS 值与相交三角形数量的关系

从上述表格和折线图中可以看出，当对象没有相交或相交的三角形数量较少时，Sphere 层次包围盒法速度快，这是因为大部分三角形对不相交，通过 Sphere 相离排除简单，当相交数量增多时，其包围紧密性差的缺点暴露出来，速度迅速下降，当相交数量超过 1000 时，FPS 降到 30 以下。AABB 层次包围盒法在相交的三角形数量较少时速度小于 Sphere 方法，因为其相交测试比 Sphere 慢，分离排除就会慢一些，当相交数量增多时，其速度下降较慢，因为其包围紧密性相对较好。Sphere-AABB 复合方法结合了二者的优点，随相交数量的增加，速度变化平稳，性能均在单独使用 Sphere 和 AABB 方法之上，当三角形相交数量超过 1106 时，FPS 值仍为 53.2，满足视觉反馈实时性要求。

4.6 本章小结

本章主要针对虚拟手术系统中的碰撞检测方法进行了研究。根据不同包围盒的优缺点及适用场景，提出 Sphere-AABB 复合层次包围盒方法作为本文的碰撞检测方法，对该方法涉及到的基本包围盒的构建，相交测试，计算与更新，树的构建及更新做了详细介绍。对于海燕等人提出的基于投影降维的三角形间的碰撞检测算法做了改进并应用在基本图元间的相交测试中。最后对不同碰撞检测方法做了实验对比，说明了复合层次包围盒结合基于投影降维三角形碰撞检测方法在实时性方面的优势。

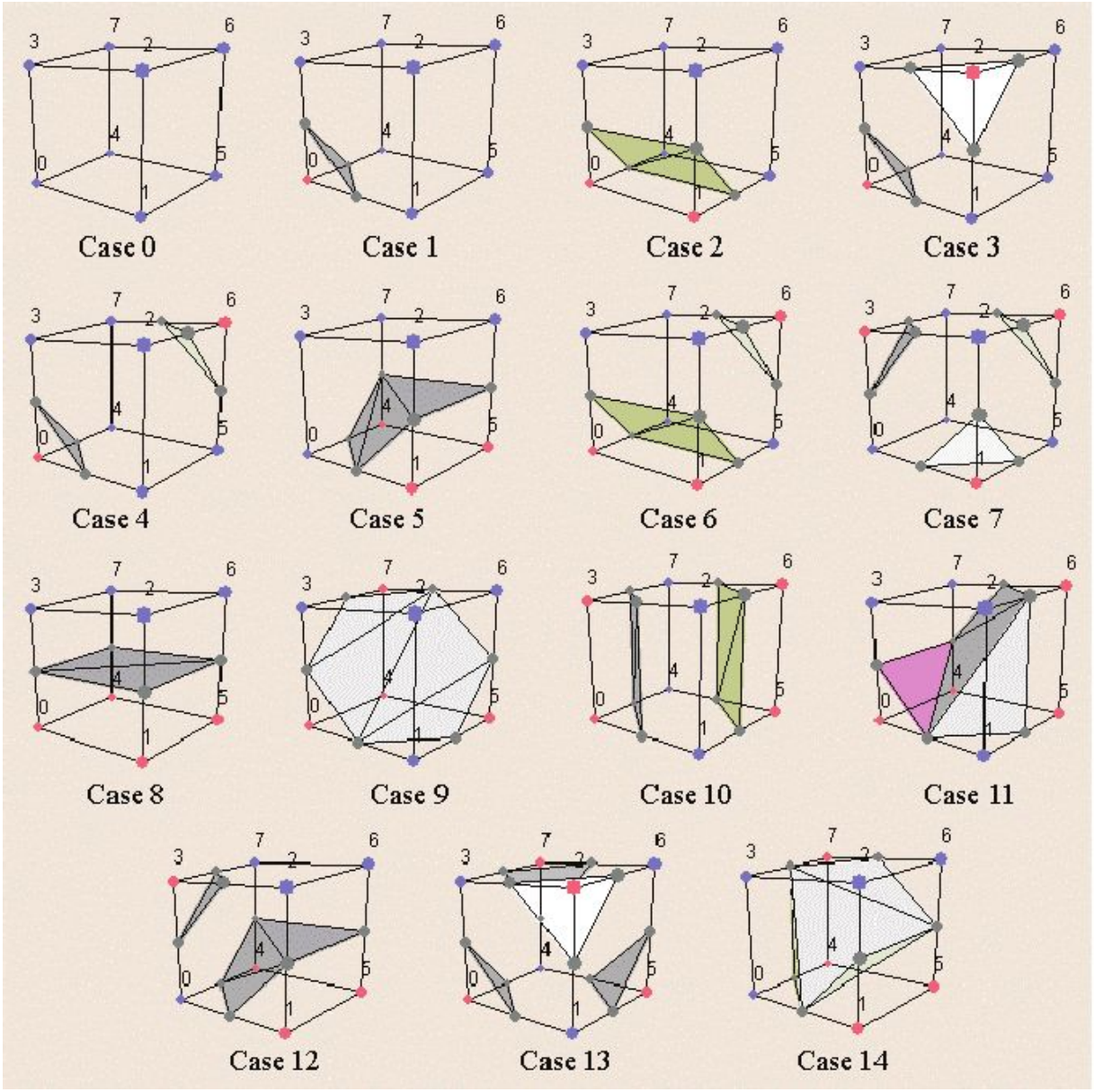


图 2.4 体素内等值面的分布情况

$$g(x,y,z)=\nabla f(x,y,z)$$

$$\begin{cases} g_x=\dfrac{f(x_{i+1},y_j,z_k)-f(x_{i-1},y_j,z_k)}{2\Delta x} \\ g_y=\dfrac{f(x_i,y_{j+1},z_k)-f(x_i,y_{j-1},z_k)}{2\Delta y} \\ g_z=\dfrac{f(x_i,y_j,z_{k+1})-f(x_i,y_j,z_{k-1})}{2\Delta z} \end{cases} \tag{2-2}$$

等值点的相应坐标和法向量可通过式（2-3）求得。

$$P=P_1+(P_2-P_1)(isovalue-V_1)/(V_2-V_1) \tag{2-3}$$

其中 P 代表等值点的相应量（坐标或法向量），P_1, P_2 代表端点的相应量，V_1, V_2 代表端点的灰度值，*isovalue* 代表等值点的灰度值。这里假设在边上灰度值是线性变化的。

2.4 利用三维建模软件绘制法

这种方法是指利用三维建模软件（如 3ds Max，Maya，Blender 等）绘制出具有三维数据的模型。

以 3dsMax 建模软件建立一个心脏模型为例简单介绍一下建模过程[43]。

（1）首先参照相关图片或实体绘制出心脏的 6 个方向的位图，位图大小为 900x900 像素，如图 2.5 所示。

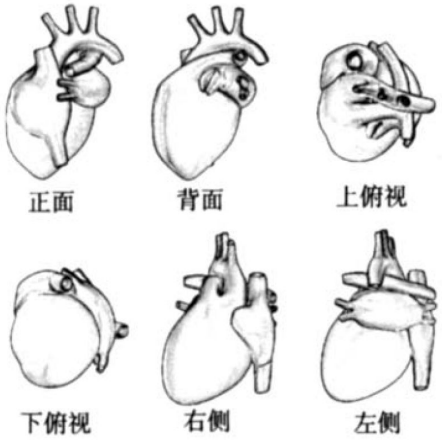


图 2.5 心脏 6 个方向的位图

（2）将上述位图导入到建模软件中，如图 2.6

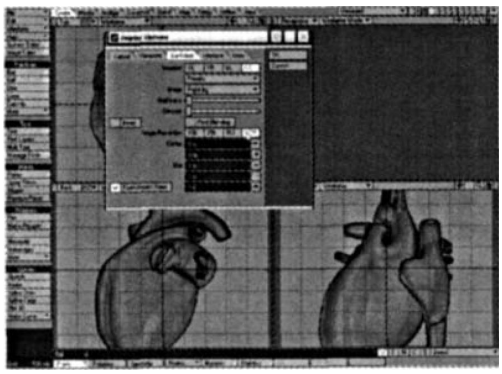
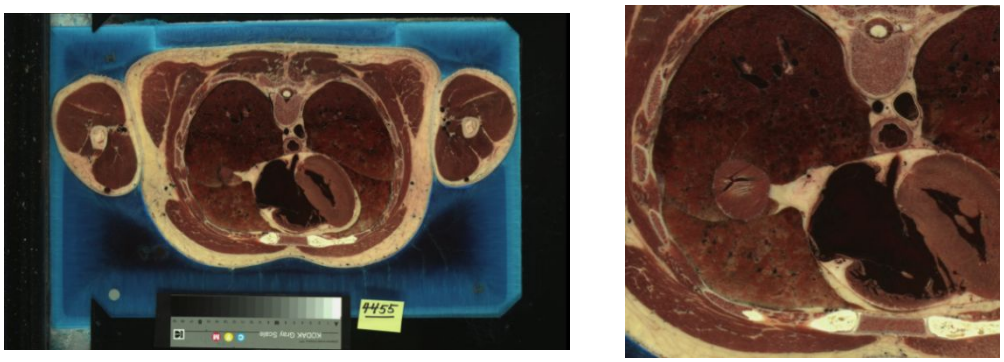


图 2.6 将位图导入软件

（3）将模型分解成多个部分分别来建立，下面只介绍建立模型中的一部分。

首先建立一个最简单的 Box 对象，利用 polygones 选项中的 metafrom 结合手动操作不断修正 Box 对象使其与导入的三个方向上的位图主体相吻合。如图 2.7 所示。

本文拟建立人体肝脏模型，CT 图像选用美国国立医学图书馆“可视人计划”中的腹部部分切片图像，因为切片间距只有 1mm，切片图像太多，为减少数据量，我们等间隔选用其中的包含肝脏部分的 200 张。总的数据量大小为2048×1216×200 体素，其中只有肝脏部分是我们所需要的，为了减小数据量，我们对图像进行裁剪只保留需要的部分。裁剪前后对比如图 2.13。



(a) 剪裁前 2048×1216×200 体素　　(b) 剪裁后 619×635×200 体素

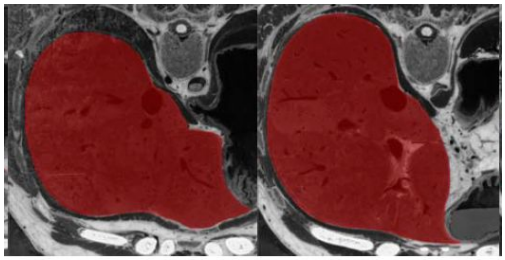
图 2.13 剪裁前后数据量对比

2.图像导入

通过 File/Load 菜单将图像全部导入并调整参数。导入后对象池中会显示其图标。

3.图像分割及平滑

这一步是三维重建的基础。分割的质量和速度都很重要，现在分割的方法有手动分割和自动分割，在图像中我们可以看到肝脏组织与周边组织对比度不高，边界模糊，其他微细结构又对其有影响，使用自动分割很困难，而如果手动分割工作量又太大[46]。所以我们采用二者结合的方法，取一个合适的间隔值将图片分组，间隔处采用手动分割，然后利用 Amira 的自动插值方法分割间隔之间的图片。分割后检查一遍，分割不好的地方手动修正，同时为每张图像作上标签。利用“smooth label”对边界做光滑处理，“remove islands”去除孤岛，增强分割效果。分割后并进行填充后的效果如图 2.14。



52张/200张　　80张/200张

图 2.14 肝脏分割效果

4.三维重构

使用 Amira 提供的移动立方体法构建肝脏面模型，并对其进行表面光滑处理。

5.模型导出

因为 Amira 导出的模型格式不能直接用于我们的系统，所以我们利用 3ds Max 软件先进行一下全局坐标调整，然后将其导出为.obj 格式文件，重建并加入纹理的肝脏面模型如图 2.15，共有 725 个顶点，1448 个三角形组成。

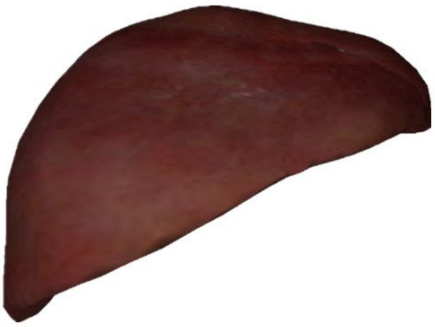
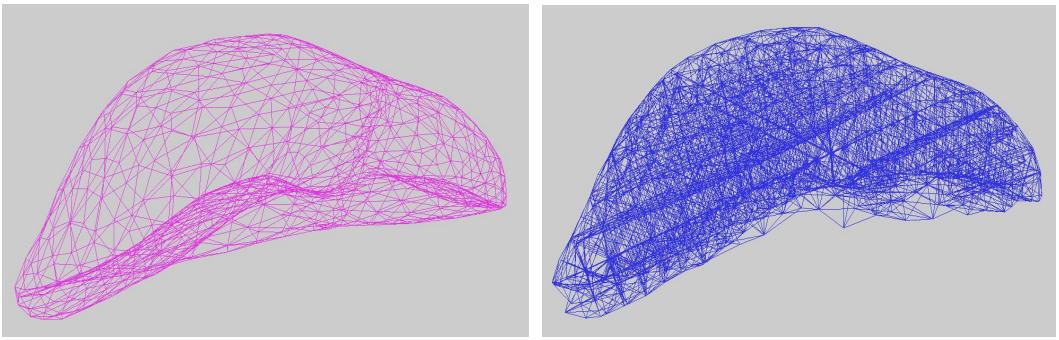


图 2.15 三维重建得到的肝脏面模型

因为肝脏为实体，应当建立其体模型，建立体模型的常用方法就是将面模型进行四面体剖分，本文利用 QTetraMesher（http://code.google.com/p/qtetramesher/）软件来完成此工作，建立后的体模型如图 2.16（b），共含有 1206 个顶点，4230 个四面体，9313 个三角形和 6288 条边。图 2.16（a）为四面体剖分前的网格表示的肝脏面模型。



（a）网格表示形式的面模型　　（b）通过四面体剖分获得的体模型

图 2.16 肝脏几何模型

2.6 本章小结

本章主要对软体模型的几何建模方法做了总结，是本文的基础章节。首先介绍了医学图像三维重建中的面绘制法原理，然后以心脏模型为例介绍了利用 3ds Max 三维建模软件建立几何模型的过程。因为软体要用在虚拟手术训练系统中，对模型逼真度要求较高，所以，选择利用医学图像三维重建方法借助 Amira 软件建立了本文要用到的肝脏几何模型。

5.3.2 系统中重要的数据结构

在系统的实现中，数据结构是非常重要的，因为，程序就是算法与数据结构的融合合，数据结构相当于骨架，是算法实现的基础，良好的几何模型数据结构不仅可以提高系统的整体性能还有助于计算的简化。设计数据结构就是选择存储方式，在这里介绍一下系统中用到的重要的数据结构的思想。

如图 5.4 所示为简化的的数据结构，它能够清晰地表示点、面、对象之间的相互关系，从而使数据的访问和更新更快捷，方便。当然图中只是用简化的形式表示了系统中相关数据结构的思想，实际的数据结构比这复杂的多。

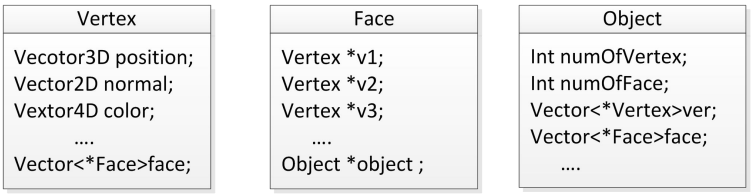


图 5.4 系统中重要的数据结构

5.4 实验结果

5.4.1 实验结果截图

1、肝脏在不同角度观察下的变形前状态，如图 5.5 所示。

(a) 左前方视图　　(b) 左后方视图

图 5.5 不同角度观察下的变形前状态

2、按压变形效果，如图 5.6 所示。

(a)左前方视图　　(b)主视图

图 5.6 肝脏按压形变效果

3、提拉变形效果，如图 5.7 所示。

(a)左前方视图　　(b)左后方视图

图 5.8 肝脏提拉形变效果

5.4.2 结果分析

在整个实验中平均 fps 值均在 50 以上，在虚拟手术系统中视觉反馈 fps 至少要达到 24 以上[60]，实验中的 fps 值远高于此值，视觉反馈流畅，满足了实时性的要求。从图 5.7 和 5.8 中可看出肝脏的按压、提拉变形效果接近于真实情况，没有出现塌陷和超弹性现象，同时手术器械没有穿越和远离软体组织，满足逼真性的要求。从而验证了本文基于形状匹配的无网格物理建模方法和复合层次包围盒树碰撞检测方法的有效性。

5.5 本章小结

本章在 VS2010 集成开发环境利用 C++语言、OpenGL 三维图形库和文中的算法搭建了一个简易虚拟手术系统，进行了肝脏模型的按压、提拉变形实验。分别对 OpenGL 的核心原理、虚拟手术系统的框架和重要的数据结构做了简要介绍，同时展示了实验的结果并做了分析。实验结果说明系统满足实时性和逼真性两个方面的要求，验证了本文算法的有效性。