

基于Delaunay方法的二维三角网格局部调整技术 及其应用

高天成

School of Mathematics
and Systems Science
Beihang University
Beijing, China 100091
Email: gtczz@sina.com

Homer Simpson

Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

James Kirk

and Montgomery Scott
Starfleet Academy
San Francisco, California 96678-2391
Telephone: (800) 555-1212
Fax: (888) 555-1212

Abstract—在本文中，我们提出了一种基于动边界局部调整的二维网格生成方法。通过引入局部调整而非全局重生成，在提升网格生成效率的同时保证的网格的质量。更进一步，通过与高维嵌入技术结合，可以针对移动边界生成自适应的动态网格。并通过具体例子展示该方法的有效性，效率，以及稳定性。

Index Terms—网格生成, Delaunay方法, 动态网格生成, 高维嵌入

I. 引言

网格生成通常是科学计算的第一步，同时也是消耗最多的第一步。文献[1]指出，其时间花费通常占到整个科学计算任务的60%。通过移动物体的边界，可以描述物体的变形或移动。所以动态网格可以理解为一种描述边界移动对网格影响的方法。简单来说，动态网格分成两种：保拓扑方法和不保拓扑方法。保拓扑方法指的是在变动前后网格的拓扑性质不变，即只有网格中点的位置移动，不会有点的消失或添加或边的变动。由此，为了防止网格穿透，每个点的移动不能过大以避免越过某一条边。尽管有这些缺点，保拓扑方法天然保证解算区域的守恒性，这点对于最终解的收敛性是至关重要的。其中典型的方法有弹簧拉伸法和背景网格映射法。然而现实中显然存在一些实例使得网格无法保拓扑，例如物体分成两半，新的边和点必须在分裂之后的两个新物体之间生成。故而，因为这些本质不保拓扑的，或者强行保持网格拓扑性质会导致网格质量极度变差的问题，例如相对运动，不保拓扑的方法逐渐发展起来。在本文中，我们提出了一种简单但有效的基于Delaunay准则的动态网格生成方法。因为Delaunay准则的要求，网格的拓扑性质是无法保持的，故而属于不保拓扑方法。该方法的优点是对物体大位移的网格灵活性以

及Delaunay方法带来的算法鲁棒性。同时为了提升算法效率，采用局部调整而非全局重生成。最后，将该方法与高维嵌入技术结合，从而生成自适应的动态网格。

II. DELAUNAY三角化和高维嵌入

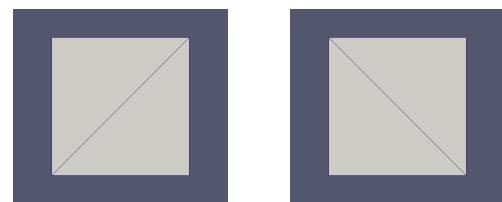
在本节中，首先会为了后文的深入讨论给出Delaunay三角化的简要介绍。其中，作为动态网格生成算法的基础，Lawson翻转算法和Delaunay再优化将会被着重介绍。

A. Delaunay三角化

我们首先给出二维三角化的定义[3]:

Definition 1: 一个平面点集S的三角化是这个平面的一个划分，并且极大化，两个顶点均是S中的点的，不相交的边的数量。

通常，一个固定点集有许多不同的三角化。然而，通过一系列边翻转，即通过将两个相邻三角形组成四边形，将原来两个三角形的公共边替换成四边形的另一条对角线，可以使得这些三角化相互转化[4]。例如图1中展示的，从1(a)到1(b)的变换就是一次边翻转。



(a) 翻转前

(b) 翻转后

在一个确定点集的所有三角化中，有些三角化相对于其他的三角化在大多数情况下是更有意义的。Delaunay三角化正是这样一个特殊的三角化。它有着许多等价的定义，其中一种为[3]：

Definition 2: 对于一个三角化中所有的三角形，其外接圆内不包含除去该三角形三个顶点以外的任何点，则这个三角化被称为Delaunay三角化。

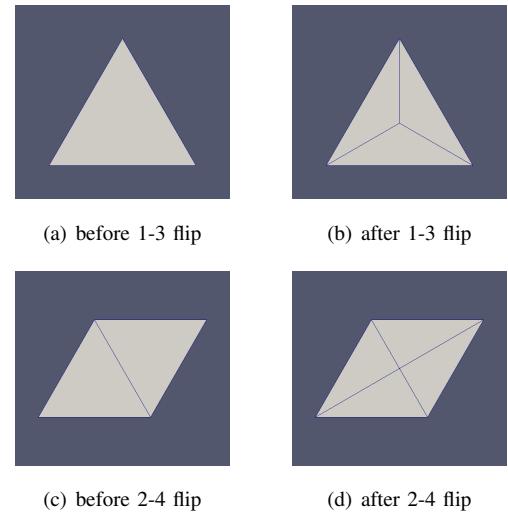
这个定义表明了Delaunay三角化包含的规则的三角形相对于其它的三角化更多，而这点正好保证了网格的质量。因此，基于Delaunay的网格生成方法成为了一种主流的三角网格生成方法。而Delaunay三角化生成算法有很多种，例如分治算法，增量算法，通过Voronoi划分的对偶图获得等等。在本文的动态网格生成算法中，使用了基于Lawson边翻转的增量算法[4]（将任一三角化通过一系列边翻转转化成Delaunay三角化的算法）。之所以使用该算法是因为边翻转算法的灵活性有助于后续的动态网格生成，同时其鲁棒性保证回归到Delaunay三角化，尽管其计算效率并不是最优的。

Algorithm 1: 增量Delaunay三角化生成算法

```

Input:
一个点集 $S$ ;
Output:
一个Delaunay三角化 $T$ ;
1 随机选取三个点生成第一个三角形，从 $S$ 中移除这三个点;
2 while  $S$ 非空 do
3   取出一个点，记为 $p$ 在当前三角化中定位该点;
4   if  $p$  在当前三角化的闭包中 then
5     | 通过1-3和2-4边翻转将新点插入当前三角化中;
6   else
7     | 将新点与对应合适的闭包上的边生成一个新的三角形;
8   end
9   通过Lawson边翻转将新的三角化变成Delaunay三角化;
10 end
```

这里1-3翻转和2-4翻转指的是插入点的方法。如果待插入的点在某一个三角形内部，则进行一次1-3翻转，将原三角形分成三个三角形。如果待插入的点在某两个三角形的公用边上，则进行一次2-4翻转将这两个相邻三角形分成4个三角形。形如图2中所展示的。



B. 强制Delaunay三角化和Delaunay再优化

Delaunay三角化为生成高质量的三角网格体重了一种有效的方法。并且，对任一确定的点集，如果该点集处于通常状态，即没有四点共圆，Delaunay三角化是唯一的。然而，在具体应用中，有时不能只对简单点集进行三角化。比如有的物体的边界在网格生成时必须保留，从而描述这个物体的形状变化或运动。但是这些物体的边界与Delaunay准则相违背，这时需要强制这些边在最后的三角化中存在。因此，强制Delaunay三角化应运而生。强制Delaunay三角化指的是是一种全局Delaunay而局部非Delaunay的三角化，强制Delaunay三角化可以通过普通Delaunay三角化通过一系列边翻转，恢复强制边来生成，其具体算法见文献[5]。而Delaunay再优化方法，则是一种通过添加原点集以外的额外点来提升网格质量的方法。因为原点集在平面的分布不一定是均匀的，故而即使Delaunay三角化可能仍然会产生质量很差的三角化。对于这种情况，可以使用Delaunay再优化方法来添加额外点，分割大的三角形，从而导致不满足Delaunay准则，进而产生边翻转，最终消除过小或过大的角。具体来说，这个过程先检查所有三角形，然后再其中过大的三角形中心，或者过小角的强制边上插入点[7]，然后进行Lawson边翻转恢复强制Delaunay性质。通过迭代进行这一步骤，原始的Delaunay三角化得以获得均一化和优化。

C. 高维嵌入技术

在文献[2]中Dassi、Si和Perotto提出了一种新颖的通过高维嵌入的方法生成各向异性网格的方法。其主要思想是通过将平面三角化的节点通过嵌入函数提升到高维空间，称为嵌入空间。然后在嵌入空间中生成一个拟Delaunay的

三角化，这一步通过在嵌入距离下分割，拼接和翻转三角化的边实现。最终，将这个拟Delaunay 三角化投影回原平面。其中各向异性网格的生成是基于特殊的嵌入函数的性质获得的。换言之，可以通过选择合适的嵌入函数将该方法变成一种自然的自适应网格优化方法。

III. 动态网格生成

本文的动态网格生成方法是基于前一节介绍的强制Delaunay三角化方法的。通过一系列首尾相连的强制边组成一个物体边界的表示。所以动态网格生成的主要任务是移动这些强制边来模拟物体的移动和变形。

A. 动态网格生成算法总览

简单来说，本问的动态网格生成方法的核心思路是在移动物体周围“挖一个洞”，从而使得在移动表示物体边界的强制边的时候，不会与其它点和边产生交叉。在将边界强制边移动完后，通过Lawson边翻转和Delaunay再优化来恢复网格的Delaunay性质并进行网格质量的局部优化。算法流程如下2：

Algorithm 2: 动态网格生成算法

Input:

一个点集 S

一组描述物体边界运动的向量 $\{v_s^t\}$, t 代表当前步数,
 s 代表对应点的序号

总步数 $Maxstep$

Output:

一系列网格 $\{M^t\}$

1 生成初始状态的网格;

2 $step = 0$;

3 输出该初始网格 M^{step} ;

4 **while** $step < Maxstep$ **do**

5 将再优化过程中添加的自由边界点移除，恢复强制边;

6 移除边界运动过程中的冲突点和冲突边;

7 通过移动强制点来完成对物体边界的移动:

$$S_i^{step+1} = S_i^{step} + v_{S_i}^{step};$$

8 通过局部Lawson边翻转和Delaunay再优化提升局部网格质量;

9 输出当前时间层的网格: M^{step} ;

10 $step = step + 1$;

11 **end**

这个算法是基于人们的直观感觉，即清除移动路径上的所有阻碍边和点，这样边界强制边就可以没有冲突的直

接移动。因为算法调用了删点和加点的操作，故而这个算法属于不保持拓扑的网格生成算法。并且该算法对大变形，相对运动，大位移的网格变形天然支持，在时间效率上，因为在每一个时间步中没有任何的迭代操作，所以计算量极大地减小了。

B. 算法细节讨论

在本小节中将对算法的细节进行一些讨论。因为对每一个时间步物体边界的描述是通过在物体强制边界点上的一个位移向量表示的，所以那些因为前步Delaunay再优化添加的自由边界点需要先被移除，因为其上没有位移量的描述。这样可以在恢复物体边界一致性的同时，减少物体周围三角形的数量，一定程度上降低计算量。为了移除可能的冲突点，需要首先在所有点的列表中找出所有的冲突点。针对所有运动的边界边，区域内所有在该边运动路径上的点都是这个边界边的冲突点。因此，可以将所有边界边的冲突点全部分别移除。但是，这样对每一个边遍历一次全部点列表会导致计算效率十分低下。所以，可以通过放宽对冲突点的判断标准，从而获得查找冲突点计算量的降低。在本文的实例中，使用了一个平行于两个坐标轴的矩形，包围住物体移动前后的位置。同时，为了防止移动之后产生质量过差的三角形，可以将这个矩形4个方向都扩大一些。然后只需要对全局点列表循环一次，移除矩形内部所有的非边界点，就可以去除所有的冲突点。然而，仅仅移除所有冲突点并不能让边界无阻碍地移动打终止位置，因为有可能还有冲突边留存，如图??所示。故而还需要处理冲突边，对此，可以将所有的冲突边按照距离待移动点从近到远的顺序进行边翻转，这样将这些边的一端成为待移动点，从而不再成为边界点移动的阻碍。在所有强制边和边界点移动之后，再局部调用Lawson边翻转算法和Delaunay再优化算法。之所以局部调用的目的是减少计算代价，具体通过只将被移动的边界点相邻的三角形推入待处理队列，然后使用上述算法对这个队列中的三角形进行处理。这样可以避免对大量无关的三角形进行无意义的判断。

C. 动态高维嵌入

为了将高维嵌入算法与本文的动态网格生成方法结合，一个很直观的想法是通过让嵌入函数跟随移动物体一同变化，每一步对嵌入函数进行修改。这样，可以将原本的动态网格生成方法最后的部分变成局部Lawson边翻转和高维嵌入，通过将Delaunay再优化替换为高维嵌入方法，并相应的修改嵌入函数，从而做到使得高维嵌入的自适应效果跟随物体移动进行。这个算法能够生成自适应

Algorithm 3: 动态高维嵌入网格生成算法

Input:

 一个点集 S

 一组描述物体边界运动的向量 $\{v_s^t\}$ t 代表当前步数, s 代表对应点的序号

 总步数 $Maxstep$

 动态嵌入函数 $F(x, y, t)$, $t = 1, 2, \dots, Maxstep$
Output:

 一系列网格 $\{M^t\}$

1 生成初始状态的网格;

 2 $step = 0;$

 3 输出该初始网格 M^{step} ;

 4 **while** $step < Maxstep$ **do**

5 将高维嵌入过程中添加的自由边界点移除, 恢复强制边;

6 移除边界运动过程中的冲突点和冲突边;

7 通过移动强制点来完成对物体边界的移动:

$$S_i^{step+1} = S_i^{step} + v_{S_i}^{step};$$

 8 基于当前的动态嵌入函数 $F(x, y, step)$ 进行高维嵌入;

 9 输出当前时间层的网格: M^{step} ;

 10 $step = step + 1$

 11 **end**

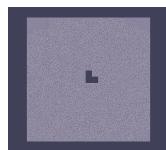
再优化的动态网格, 对具体情况而言, 可以对变形物体边界处生成各向异性网格, 从而模拟边界层等实际情况所需要的网格特性。

IV. 网格生成实例

本节主要展示一些具体的动态网格生成实例, 以及对动态网格生成方法与每个时间步网格重新生成方法进行计算时间的统计比较。

A. 物体简单平移, 相对运动, 变形的动态网格实例及时间统计

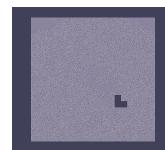
1) 物体简单平移动态网格实例:



(a) T=1



(b) T=40



(c) T=80

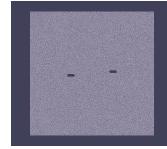
3 中展示了L型物体简单平移对应的动态网格。整个平移进行了79步, 包括初始状态共生成80个平面网格。整个平面尺寸约为物体尺寸的10倍, 移动距离约为平面尺寸的1/3。图中所展示的是初始网格, 第经过40步的网格, 和最终状态的网格。其时间统计如表I所示。

TABLE I
L型物体平移时间统计表

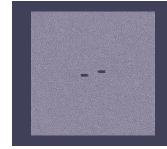
网格序号	10	20	30	40	50	60	7
动态生成累计用时 (ms)	690	1001	1333	1674	2034	2406	28
重生成累计用时 (ms)	3452	6890	10347	13801	17233	20671	241

可以看出在处理简单图形的简单运动时, 动态网格的总用时大约是每个时间步重新生成所用时间的1/9左右, 而在除去第一张基础网格的生成所用时间之外, 每一步动态网格的生成约是重新生成的1/10左右。

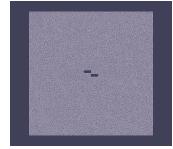
2) 五边形物体相对运动动态网格实例:



(a) T=1



(b) T=40



(c) T=80

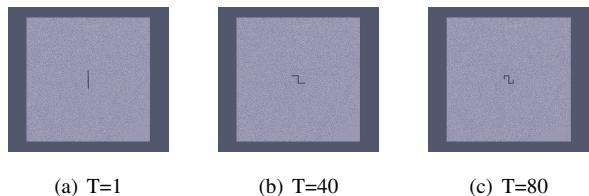
4中展示了两个五边形物体相向运动, 然后交错而过的动态网格, 整个运动过程进行了79步, 包括初始状态共生成80个平面网格。整个平面尺寸约为物体尺寸的10倍, 移动距离约为平面尺寸的1/2。图中所展示的是初始网格, 第经过40步的网格, 和最终状态的网格。其时间统计如II所示。

TABLE II
L型物体平移时间统计表

网格序号	10	20	30	40	50	60	7
动态生成累计时间 (ms)	646	954	1270	1595	1924	2244	25
重生成累计时间 (ms)	3591	7168	10755	14384	17966	21532	250

可以看到与简单物体平移相仿的结果, 动态网格生成的总耗时大约是重生成总耗时的1/10, 除去初始网格以外, 动态网格生成每一张网格的耗时大约是重生成生成一张网格的1/11。

3) 棍状物体弯折大变形动态网格实例:



5中展示了一个棍状物体连续弯折，最终形成一个S型的物体。整个运动过程进行了79步，包括初始状态共生成80个平面网格。整个平面尺寸约为物体尺寸的10倍，变形位移与物体尺寸相当。图中所展示的是初始网格，第40步的网格，和最终状态的网格。其时间统计如III所示。

TABLE III
棍状物体大变形时间统计表

网格序号	10	20	30	40	50	60	70	80
动态生成累计时间 (ms)	798	1194	1642	2116	2593	3066	3530	3985
重生成累计时间 (ms)	5955	11806	17641	23478	29324	35185	41039	47108

V. CONCLUSION

The conclusion goes here.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

- [1] Zhe Jiang. Research of the Generation of Dynamic Unstructured Meshes[D]. Nanjing University of Science & Technology, 2006.
- [2] Dassi F, Si H, Perotto S, et al. Anisotropic Finite Element Mesh Adaptation via Higher Dimensional Embedding[J]. Procedia Engineering, 2015, 124:265-277.
- [3] Devadoss S L, O'Rourke J. Discrete and computational geometry[M]. Springer, 2001.
- [4] Charles Lawson. Transforming triangulations. Discrete Mathematics, Volume 3, pages 365 – 372, 1972.
- [5] Chew L P. Constrained Delaunay triangulations[J]. Algorithmica, 1989, 4(1-4):97-108.
- [6] Shewchuk, Jonathan R. (2008). "General-Dimensional Constrained Delaunay and Constrained Regular Triangulations, I: Combinatorial Properties". 39 (1-3): 580 – 637.
- [7] Ruppert, Jim (1995). "A Delaunay refinement algorithm for quality 2-dimensional mesh generation". Journal of Algorithms. 18 (3): 548 – 585. doi:10.1006/jagm.1995.1021