文章编号:1007-5321(2019)05-0083-08 **DOI**:10.13190/j.jbupt.2018-266

基于重心映射的三角形网格参数化方法研究与实现

管焱然[1], 奥利弗·范凯克[1], 管有庆[2]
(1. 卡尔顿大学 计算机科学学院, 渥太华 K1S 5B6; 2. 南京邮电大学 物联网学院, 南京 210003)

摘要: 针对在几何处理领域有着广泛应用的三角形网格参数化问题,研究了基于重心映射的三角形网格参数化方法. 利用 geometry-processing-js 类库中的半边数据结构,采用均匀拉普拉斯权重、拉普拉斯-贝尔特拉米权重和中值权重 3 种加权方案,实现了重心映射法,并根据三角形的形变量分析了参数化结果. 结果表明,中值权重为重心映射法的最优加权方案.

关 键 词: 参数化; 重心映射; 三角形网格; 几何处理
中图分类号: TN911.22 **文献标志码**: A

Research and Implementation of Triangle Mesh Parameterization Method Based on Barycentric Mapping

GUAN Yan-ran[1], Oliver van Kaick[1], GUAN You-qing[2]
(1. School of Computer Science, Carleton University, Ottawa K1S 5B6, Canada;
2. School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: The parameterization of triangle meshes has numerous applications in the field of geometry processing. Using the halfedge-based data structure provided by geometry-processing-js, this paper researches on the mesh parameterization and its implementation based on barycentric mapping, as well as three weight sets applied in barycentric mapping, namely, the uniform Laplacian weights, the Laplace-Beltrami weights, and the mean value weights. A comparison between the results of parameterization is given based on the distortion caused to the triangles, and a conclusion is given that the mean value weights provide the best weighting scheme for barycentric mapping.

Key words: parameterization; barycentric mapping; triangle mesh; geometry processing

三角形网格由一组通过公共边或公共角相连的三角形构成. 由于三角形本身具有的简单几何特性,在几何处理领域,三角形网格是一种十分高效而且常用的对三维物体的二维流型表面进行离散化表达的方式. 参数化通常指平面参数化. 对于任意用三角形网格表示的物体表面,存在一个从网格上的点 P_i 到二维平面上点 P_i^* 的一一映射,使得二维平面上的网格与原网格的拓扑同构,这样的一一映射被称为三角形网格的参数化[1]. 三角形网格映射到的二维平面则被称为参数域.

三角形网格参数化在几何处理和计算机图形学中有着广泛的应用. 通过参数化,可以找到二维图像和三维物体表面之间的对应点,从而实现纹理贴图[2-3]. 类似地,三维物体表面的其他细节信息,例

收稿日期: 2018-11-16
基金项目: 江苏省高校自然科学研究计划项目(05KJD520146)
作者简介: 管焱然(1994—), 男, 硕士生.
通信作者: 管有庆(1963—), 男, 副研究员, E-mail: guanyouq@njupt.edu.cn.

如法线信息,也可以通过参数化映射到二维平面上,从而更好地实现光照对物体表面的渲染[4]. 此外,参数化能够保存三维物体表面细节的性质还可以应用于细节转换[5-6]、网格编辑[7]、曲面变形[8]等.

目前,实现将三角形网格映射到二维平面上的参数化方法有许多种,比较常用的有:重心映射法(barycentric mapping)或称塔特嵌入法(Tutte embedding)[9]、基于角度的拍平化法(ABF, angle based flattening)[10]及其改进方法ABF++[11]、基于最小二乘的保角映射法(LSCM, least squares conformal maps)[12]、最等距参数化法(MIPS, most isometric parameterization of surfaces)[13]及其改进方法(AMIPS, advanced most isometric parameterization of surfaces)[14]、有界失真映射法(BDM, bounded distortion mapping)[15]、局部内射映射法(LIM, locally injective mapping)[16]、边界优先拍平化法(BFF, boundary first flattening)[17]等. 基于geometry-processing-js几何处理类库[18]中三角形网格的半边数据结构(halfedge-based data structure)[19-21],实现了重心映射参数化方法,并采用了3种加权方案:均匀拉普拉斯权重(uniform Laplacian weights)[22]、拉普拉斯-贝尔特拉米权重(Laplace-Beltrami weights)[23]和中值权重(mean value weights)[24],分析了3种加权方案对参数化结果产生的影响.

1 三角形网格的半边数据结构

三角形网格是由一系列离散且无序的三角形组成的. 一个设计良好的三角形网格数据结构不仅可以实现对网格的快速构建,还能够对网格中点、边、面等信息进行高效地遍历与查找,并减少存储和处理网格需要的内存消耗和冗余. 基于半边的数据结构就是一种优秀的构建和存储三角形网格的方式.

半边指将三角形的一条边分解为一对拥有相反方向的有向边. 在基于半边的数据结构中,网格的每个面和每条边界由半边逆时针环绕而成. 如图1所示,半边$\langle A,B\rangle$、$\langle B,C\rangle$和$\langle C,A\rangle$构成面ABC;半边$\langle A,C\rangle$、$\langle C,D\rangle$和$\langle D,A\rangle$构成面ACD;半边$\langle B,A\rangle$、$\langle A,D\rangle$、$\langle D,C\rangle$和$\langle C,B\rangle$构成了网格边界(这样的半边称为边界半边,这些半边间的顶点称为边界顶点). 网格边界也可以因此看作由多条半边逆时针环绕而成的虚拟面. 此外,三角形中的每一条半边也能定义其唯一的对应角(即两个相邻面中的非共享角). 图1中,$\angle ABC$为半边$\langle C,A\rangle$的对应角,$\angle ADC$为半边$\langle A,C\rangle$的对应角. 同理,三角形中的每一个角也能找到其对应半边.

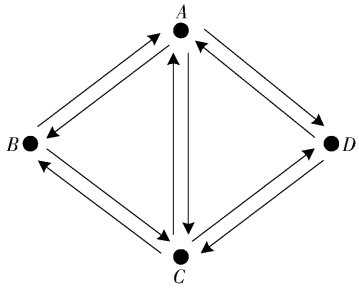


图1 由半边表示的三角形网格

在三角形网格的半边数据结构中,每一条半边中存储的信息如下:

1) 半边的出射顶点,如图1中,顶点C为半边$\langle C,A\rangle$的出射顶点;

2) 半边的对应角(边界半边则没有对应角),如图1中,$\angle ABC$为半边$\langle C,A\rangle$的对应角;

3) 半边的所在面(边界半边的所在面为表示边界的虚拟面),如图1中,面ABC为半边$\langle C,A\rangle$的所在面;

4) 半边所在面或所在边界中的前驱半边,如图1中,$\langle B,C\rangle$为$\langle C,A\rangle$的前驱半边,$\langle B,A\rangle$为$\langle A,D\rangle$的前驱半边;

5) 半边所在面或所在边界中的后继半边,如图1中,$\langle A,B\rangle$为$\langle C,A\rangle$的后继半边,$\langle D,C\rangle$为$\langle A,D\rangle$的后继半边;

6) 半边的反向半边,如图1中,$\langle A,C\rangle$为$\langle C,A\rangle$的反向半边.

这样,可以将网格的几何信息存入三角形的顶点、角和面中,将网格的拓扑信息存入半边中,从而构建出三角形网格的半边数据结构,如表1所示.

由表1可知,三角形网格的每一个顶点只与其出射半边中的一条相关联,每一个角与其对应半边关联,每一个面只与该面中的一条半边关联,而每一条半边中则存储了大量连接信息,包括出射顶点、对应角和所在面的引用以及其前驱、后继和反向半边的引用.

在geometry-processing-js几何处理类库中,网格的顶点、角、面和半边均以顺序表的形式存储,对这些信息的引用则通过索引的方式实现.

初始化三角形网格半边数据结构的过程可以概括如下步骤:

步骤1 读取全部的顶点实现对顶点对象的构

表 1　基于半边的数据结构

变量类型	变量名	注释
Vertex(顶点)		
Point	position	该顶点的坐标
HalfedgeRef	halfedge	以该点为起点的一条出射半边
Corner(角)		
HalfedgeRef	halfedge	该角的对应半边
Face(面)		
HalfedgeRef	halfedge	该面中的一条半边
Halfedge(半边)		
VertexRef	vertex	该半边的出射顶点
CornerRef	corner	该半边的对应角
FaceRef	face	该半边的所在面
HalfedgeRef	prev	前驱半边
HalfedgeRef	next	后继半边
HalfedgeRef	twin	反向半边
Boolean	boundary	边界标记:“真”/“假”

造,并存入一个顶点表中;

步骤 2　对构成三角形面的顶点组按照逆时针的环绕顺序每 3 个一组批量读取,实现对面和内部半边(非边界半边,构造时将边界标记 boundary 置为“假”)的构造;

步骤 3　遍历所有步骤 2 中创建的内部半边,将没有反向半边的内部半边找出,构造它们的反向半边,并标记为边界半边(构造时将边界标记 boundary 置为“真”),构造表示边界的虚拟面;

步骤 4　遍历步骤 2、步骤 3 中创建的半边,为内部半边构造对应角,并将对应角和其对应半边相互关联.

在每 3 个一组批量读取顶点、构造面和内部半边时,先将所有顶点的状态标为“未读”,每次读取完成后将本次读取顶点的状态标为“已读”. 对于一组顶点的读取,需要执行的步骤如下:

步骤 1　构造一个新的面对象,并将其引用存入面表中;

步骤 2　按照读取顶点的顺序,对每个顶点构造出射半边对象,并将边界标记 boundary 置为“假”,将出射顶点的引用存入其中,将其前驱和后继的引用存入其中,将步骤 1 中构造的面的引用存入其中,将该半边的引用存入出射顶点和半边表中.

步骤 3　默认使用步骤 2 中构造的最后一条半边作为步骤 1 中创建的面的关联半边,将其引用存入步骤 1 中创建的面中.

步骤 4　检查本组顶点的状态,若存在 2 个顶点状态同时为“已读”,这意味着它们之间存在之前读取顶点组时创建的半边,此时遍历半边表,找到它们之间先前创建的半边,将其与步骤 2 中创建的两点之间的新半边相互关联为反向半边.

步骤 5　将本组读取顶点的状态置为“已读”.

通过上述步骤,可以确保顶点表中的每个顶点与其一条出射半边相关联.

这样,基于表 1 给出的半边数据结构,从网格中任意一个顶点出发,都可以在无需条件判断的情况下,实现对其单环邻域(one-ring neighborhood)内所有元素的遍历. 算法 1 描述了对顶点的所有出射半边的逆时针遍历.

算法 1　遍历出射半边

```
Input: center //起始顶点
Output: outgoings //出射半边集合
function OutgoingHalfedges(center)
1 h←center.halfedge
2 h_stop←h
3 outgoings ← {h}
4 do
5   h ← h.twin.next
6   outgoings ← outgoings ∪ {h}
7 while h≠h_stop
8 return outgoings
```

对于三角形网格中的任意一个顶点,算法 1 从与其关联的半边出发,依次访问反向半边的后继半边,直到返回出发的半边为止,即可实现对顶点所有出射半边的遍历.

同理,对于顶点的单环邻域内的其他元素,如邻接顶点和邻接三角形,也可以写出类似算法 1 的遍历器实现对其的遍历.

2　重心映射

重心映射法是目前使用最为广泛的三角形网格参数化方法之一. 该方法基于图论中的塔特重心映射定理(Tutte's barycentric mapping theorem)[9],描述如下.

设一个三角形网格与圆盘同胚,若一个二维域能满足下列 2 个条件:

1) 使得网格边界上顶点的坐标(u,v)分布在一

个凸多边形上;

2) 使网格每个内部顶点(即非边界顶点)的坐标为其邻接顶点坐标的凸组合(convex combination).

则这样的二维域是一个有效参数域,这样的(u,v)坐标构成一个无自交的有效参数化.

现假设一个三角形网格M中有n个顶点,分别为$\{V_1,V_2,\cdots,V_n\}$,它们在参数域上的坐标记为$\{(u_1,v_1),(u_2,v_2),\cdots,(u_n,v_n)\}$,这些顶点按照一定顺序排列,其中内部顶点的索引为$\{1,2,\cdots,n_{\text{int}}\}$,边界顶点的索引为$\{n_{\text{int}}+1,n_{\text{int}}+2,\cdots,n\}$. 根据凸组合的定义,塔特重心映射定理中的条件2)可以写为

$$\sum_{j\neq i} a_{ij}\begin{pmatrix}u_j\\v_j\end{pmatrix}-a_{ii}\begin{pmatrix}u_i\\v_i\end{pmatrix}=0 \tag{1}$$

其中:i为M中任意一个内部顶点V_i的索引,即$i\in\{1,2,\cdots,n_{\text{int}}\}$,$j$为$M$中除$V_i$外任意一个内部顶点的索引,即$j\in\{1,2,\cdots,n_{\text{int}}\}$且$j\neq i$,系数$a_{ij}$和$a_{ii}$则由式(2)给出.

$$\left.\begin{aligned}a_{ij}&>0,\quad \text{若 } V_i \text{ 和 } V_j \text{ 是邻接顶点}\\a_{ij}&=0,\quad \text{若 } V_i \text{ 和 } V_j \text{ 不是邻接顶点}\\a_{ii}&=-\sum_{j\neq i}a_{ij}\end{aligned}\right\} \tag{2}$$

而对于M边界上的顶点,只需将其在参数域上的坐标固定在一个凸多边形上即可. 假设参数域上有一个预先定义好的凸多边形,其顶点坐标为$\{(\bar{u}_{n_{\text{int}}+1},\bar{v}_{n_{\text{int}}+1}),(\bar{u}_{n_{\text{int}}+2},\bar{v}_{n_{\text{int}}+2}),\cdots,(\bar{u}_n,\bar{v}_n)\}$,则塔特重心映射定理中的条件1)可以写为

$$a_{ii}\begin{pmatrix}u_i\\v_i\end{pmatrix}=\begin{pmatrix}\bar{u}_i\\\bar{v}_i\end{pmatrix} \tag{3}$$

其中:i为M中任意一个边界顶点V_i的索引,即$i\in\{n_{\text{int}}+1,n_{\text{int}}+2,\cdots,n\}$,系数$a_{ii}=1$.

根据式(1)和式(3),可构建如下的线性方程组,从而同时满足塔特重心映射定理中的条件:

$$\left.\begin{aligned}\boldsymbol{Au}&=\bar{\boldsymbol{u}}\\\boldsymbol{Av}&=\bar{\boldsymbol{v}}\end{aligned}\right\} \tag{4}$$

其中:$\boldsymbol{A}$为以系数a_{ij}为元素的n阶稀疏矩阵,$\boldsymbol{A}=(a_{ij})_{n\times n}$,其中$i,j\in\{1,2,\cdots,n\}$. 由式(2)和式(3)可知,$\boldsymbol{A}$中各元素的取值描述如下.

非对角线元素,即a_{ij},且$j\neq i$,如式(5)所示.

$$\left.\begin{aligned}a_{ij}&>0,\quad V_i \text{ 是内部顶点,且 } V_i \text{ 与 } V_j \text{ 邻接}\\a_{ij}&=0,\quad \text{其他情况}\end{aligned}\right\} \tag{5}$$

对角线元素,即a_{ii},如式(6)所示.

$$\left.\begin{aligned}a_{ii}&=-\sum_{j\neq i}a_{ij},\quad V_i \text{ 是内部顶点}\\a_{ii}&=1,\quad V_i \text{ 是边界顶点}\end{aligned}\right\} \tag{6}$$

式(4)中,$\bar{\boldsymbol{u}}$和$\bar{\boldsymbol{v}}$为2个等长的n元向量,用来存放M边界上的顶点在参数域凸多边形上的对应坐标,$\bar{\boldsymbol{u}}=(\bar{u}_1,\bar{u}_2,\cdots,\bar{u}_n)$,$\bar{\boldsymbol{v}}=(\bar{v}_1,\bar{v}_2,\cdots,\bar{v}_n)$,它们的分量分别记为$\bar{u}_i$和$\bar{v}_i$,其中$i\in\{1,2,\cdots,n\}$. 当$i\leqslant n_{\text{int}}$时,$\bar{u}_i$和$\bar{v}_i$均取0;当$i>n_{\text{int}}$时,$(\bar{u}_i,\bar{v}_i)$为参数域上凸多边形的对应坐标.

式(4)中,$\boldsymbol{u}$和$\boldsymbol{v}$为2个等长的n元向量,用来存放M中顶点参数化后的对应坐标,$\boldsymbol{u}=(u_1,u_2,\cdots,u_n)$,$\boldsymbol{v}=(v_1,v_2,\cdots,v_n)$,它们的分量分别记为$u_i$和$v_i$,其中$i\in\{1,2,\cdots,n\}$. (u_i,v_i)即为M中的顶点V_i参数化后的坐标.

这样,只需求解式(4)给出的线性方程组,分别求出向量$\boldsymbol{u}$和$\boldsymbol{v}$,即可得到M的参数化结果.

2.1 均匀拉普拉斯权重

M中由内部顶点组成的网格可以被视作一个无向权重图G,其顶点集为V,边集为E,即$G=(V,E)$,且$V=\{V_1,V_2,\cdots,V_{n_{\text{int}}}\}$. 式(4)中矩阵$\boldsymbol{A}$中描述内部顶点的子矩阵,即可视作$G$的拉普拉斯矩阵,记为$\boldsymbol{A}'$,$\boldsymbol{A}'=(a_{ij})_{n_{\text{int}}\times n_{\text{int}}}$,其中$i,j\in\{1,2,\cdots,n_{\text{int}}\}$. $\boldsymbol{A}'$中非零的非对角线元素a_{ij}(其中$j\neq i$且$(V_i,V_j)\in E$,下同)可以视作M中两邻接点V_i和V_j之间边(V_i,V_j)的权重. 因此,可以通过选择对边集E不同的加权方案,构造出不同的矩阵$\boldsymbol{A}'$,从而对M实现不同的参数化结果.

均匀拉普拉斯权重方案,即指对G中的边集E进行平均地加权,通常将权重取值为1即可. 这样,$\boldsymbol{A}'$中的非零非对角线元素a_{ij}和对角线元素a_{ii}的取值为

$$\left.\begin{aligned}a_{ij}&=1\\a_{ii}&=-D_i\end{aligned}\right\} \tag{7}$$

其中:D_i为顶点V_i的度数,即V_i邻接顶点的个数.

均匀拉普拉斯权重简单易实现,经典的重心映射法就是基于此实现的. 然而对边集E的均匀加权却造成了对三角形网格几何属性的忽视,例如边长、三角形内角度数等,从而造成参数化结果的失真. 因此,接下来将介绍2种其他的加权方案,以减少此类失真.

2.2 拉普拉斯-贝尔特拉米权重

拉普拉斯算子(Laplace operator)是一个多元函数的二阶微分算子,其定义为梯度的散度,可以用来

衡量函数正则性(即函数的平滑程度),其值越小,函数越平滑. 拉普拉斯-贝尔特拉米算子(Laplace-Beltrami operator)则是拉普拉斯算子在流形表面上的推广形式.

因此,可以利用拉普拉斯-贝尔特拉米算子对边集 E 进行加权,从而得到较为平滑的参数坐标,来减少参数化结果的失真. 基于拉普拉斯-贝尔特拉米算子的离散化表达式[23],对 $\boldsymbol{A}'$ 的非零非对角线元素 a_{ij} 和对角线元素 a_{ii} 取值如下:

$$\left.\begin{aligned} a_{ij} &= \frac{1}{2S_i}(\cot\alpha_{ij} + \cot\beta_{ij}) \\ a_{ii} &= -\sum_j a_{ij} \end{aligned}\right\} \tag{8}$$

其中:α_{ij} 和 β_{ij} 的位置如图 2 所示,S_i 为顶点 V_i 维诺域(Voronoi region)的面积,指的是 V_i 与其邻接点连线的垂直平分线所围成区域的面积,即图 2 中虚线围成的区域.

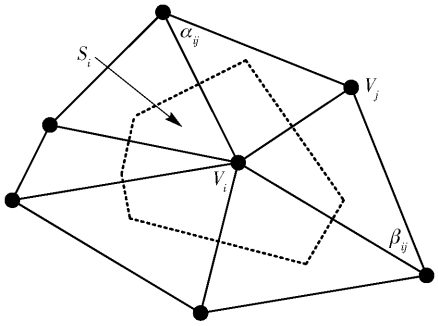


图 2　计算拉普拉斯-贝尔特拉米权重所需的角和面积

式(8)中维诺域 S_i 的面积计算如下[23]:

$$S_i = \frac{1}{8}\sum_j (\cot\alpha_{ij} + \cot\beta_{ij}) l_{ij} \tag{9}$$

其中 l_{ij} 为边 (V_i, V_j) 的长度.

为简化计算难度,可以分别对 V_i 的每个邻接三角形计算其维诺域面积. V_i 的一个邻接三角形 $\triangle V_iV_jV_k$ 中的维诺域 S_i' 如图 3 所示.

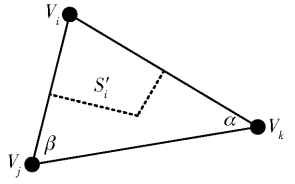


图 3　单个邻接三角形内的维诺域

在式(9)的基础上,易得 S_i' 的面积计算式为

$$S_i' = \frac{1}{8}(l_{ij}^2\cot\alpha + l_{ik}^2\cot\beta) \tag{10}$$

其中:α 和 β 的位置如图 3 中所示,分别为半边 $\langle V_i, V_j\rangle$ 和 $\langle V_k, V_i\rangle$ 的对应角,l_{ij} 为边 (V_i, V_j) 的长度,l_{ik} 为边 (V_i, V_k) 的长度.

这样,基于表 1 给出的半边数据结构和算法 1 中实现的对顶点出射半边进行逆时针遍历的遍历器 OutgoingHalfedges( ),可以实现对任意顶点的维诺域面积的计算,如算法 2 所示.

算法 2　计算维诺域面积

```
Input: center //起始顶点
Output: area //维诺域面积
function VoronoiArea(center)
1 area ← 0
2 foreach h in OutgoingHalfedges(center) do
3    l1 ← Length(h.prev)
4    l2 ← Length(h)
5    cotA ← Cotan(h.prev.corner)
6    cotB ← Cotan(h.corner)
7    area ← area + (l1l1cotA + l2l2cotB)/8
8 return area
```

算法 2 中,Length( ) 为计算半边长度的函数,Cotan( ) 为计算角的余切值的函数,在表 1 的半边数据结构基础上,这些函数的算法都容易实现. 对于顶点的每一条出射半边,求出它和它前驱半边的长度以及它和它前驱半边对应角的余切值,代入式(10),即可求出该半边所在三角形(顶点的一个邻接面)内的维诺域面积. 这样,依次遍历顶点的每一条出射半边,求出该顶点所有邻接面内的维诺域面积,再对每个三角面的维诺域面积进行累加,即可求得顶点的维诺域面积.

利用三角形网格的半边数据结构,式(8)中的 α_{ij} 和 β_{ij} 十分易得,其中 α_{ij} 是半边 $\langle V_i, V_j\rangle$ 的对应角,β_{ij} 是 $\langle V_i, V_j\rangle$ 的反向半边 $\langle V_j, V_i\rangle$ 的对应角. 将上述两角以及根据算法 2 计算出的维诺域面积代入式(8),即可计算出 $\boldsymbol{A}'$ 中各元素的值.

2.3　中值权重

在三角形网格中出现钝角的情况下,基于式(8)计算出的权重会出现负值,这会产生反转三角形,导致参数化的结果发生错误. 当然,对网格进行再分即可消除其中的钝角[25],从而避免负权重的出现. 此外,基于拉格朗日中值定理(Lagrange's mean value theorem),Floater[24] 提出了一种加权方案,确保权重一直为正,该方案称为中值权重.

中值权重方案对 $\boldsymbol{A}'$ 的非零非对角线元素 a_{ij} 和

对角线元素 a_{ii} 取值如下：

$$\left.\begin{aligned} a_{ij} &= \frac{1}{l_{ij}}\left(\tan\frac{\delta_{ij}}{2}+\tan\frac{\gamma_{ij}}{2}\right)\\ a_{ii} &= -\sum_{j} a_{ij} \end{aligned}\right\} \tag{11}$$

其中：l_{ij} 为边 (V_i, V_j) 的长度，δ_{ij} 和 γ_{ij} 为位于边 (V_i, V_j) 两侧 V_i 的邻接角，如图4所示.

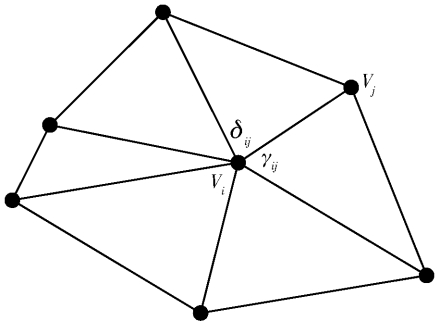


图4 计算中值权重所需的角

在具体实现中，利用三角形网格的半边数据结构，δ_{ij} 和 γ_{ij} 十分易得. δ_{ij} 是半边 $\langle V_i, V_j\rangle$ 后继半边的对应角，γ_{ij} 是半边 $\langle V_j, V_i\rangle$ 前驱半边的对应角. 计算出 δ_{ij} 和 γ_{ij} 的角度后，将其代入式(11)，即可实现对 $\boldsymbol{A}'$ 中各元素的赋值.

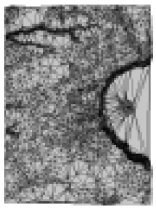
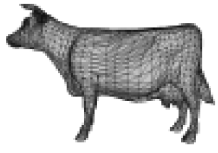
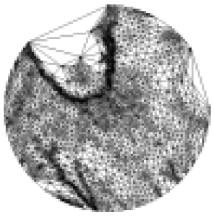
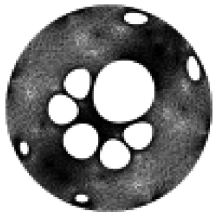
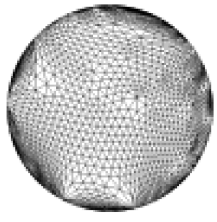
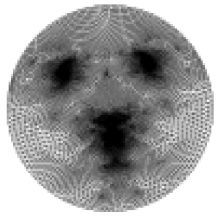
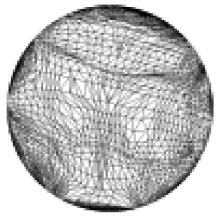
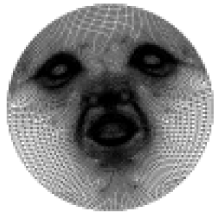
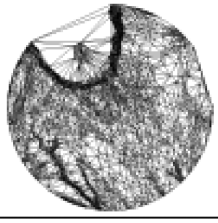
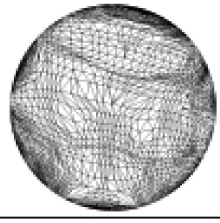
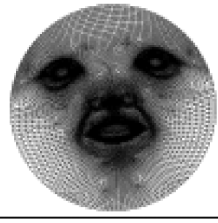
3 参数化结果分析与比较

在 Google Chrome(版本号：68.0.3440.106)的运行环境下，基于 geometry-processing-js 几何处理类库，运用 JavaScript 编程语言，实现了使用均匀拉普拉斯权重、拉普拉斯-贝尔特拉米权重和中值权重的重心映射参数化方法，并选用4种物体的三角形网格进行实验，分别为环形山、甲壳虫汽车、奶牛和人脸.

3.1 可视化分析

参数化的可视化结果如表2所示，分别给出了物体的原三角形网格以及使用均匀拉普拉斯权重、拉普拉斯-贝尔特拉米权重和中值权重进行参数化的结果. 观察表2可以发现，相较于均匀拉普拉斯权重，使用拉普拉斯-贝尔特拉米权重和中值权重的参数化能更好地维持原网格上三角形的形状. 例如，对比环形山和奶牛的参数化结果可知，均匀拉普

表2 参数化结果比较

加权方案	环形山	甲壳虫汽车	奶牛	人脸
原三角形网格				
均匀拉普拉斯权重				
拉普拉斯-贝尔特拉米权重				
中值权重				

拉斯权重使得原网格上参差不齐的三角形经参数化后形状大小更趋于一致，更接近等边三角形. 均匀拉普拉斯权重的这一特性也使得网格内部的空洞在参数化后变得更小，如表 2 中甲壳虫汽车的参数化结果. 此外，使用拉普拉斯–贝尔特拉米权重和中值权重的参数化结果更好地反映出了原网格上的一些特征. 如表 2 中，拉普拉斯–贝尔特拉米权重和中值权重使得人脸上的眼睛、鼻子与嘴巴的形状被完好地保存在了参数化结果中.

3.2　形变量分析

基于从参数域上的三角形到原网格上对应三角形的仿射变换[26]，参数化对三角形造成的形变可由该仿射变换雅可比矩阵(Jacobian matrix)的奇异值进行衡量[27]. 形变量的计算过程描述如下.

设参数域上的一个三角形为 $\triangle \boldsymbol{p}_1\boldsymbol{p}_2\boldsymbol{p}_3$，其中 $\boldsymbol{p}_i=(u_i,v_i)$ $(i\in\{1,2,3\}$，下同)，三角形 $\triangle \boldsymbol{q}_1\boldsymbol{q}_2\boldsymbol{q}_3$ 为其在原网格上的对应三角形. 令 $f:\mathbb{R}^2\rightarrow\mathbb{R}^3$ 为从 $\triangle \boldsymbol{p}_1\boldsymbol{p}_2\boldsymbol{p}_3$ 到 $\triangle \boldsymbol{q}_1\boldsymbol{q}_2\boldsymbol{q}_3$ 的仿射变换，即 $f(\boldsymbol{p}_i)=\boldsymbol{q}_i$. 令 $S_{\triangle p_1p_2p_3}=\frac{1}{2}((u_2-u_1)(v_3-v_1)-(u_3-u_1)(v_2-v_1))$ 为 $\triangle \boldsymbol{p}_1\boldsymbol{p}_2\boldsymbol{p}_3$ 的面积. 则对于 $\triangle \boldsymbol{p}_1\boldsymbol{p}_2\boldsymbol{p}_3$ 中任意一点 $\boldsymbol{p}$，仿射变换 f 由式(12)给出.

$$f(\boldsymbol{p})=\frac{S_{\triangle pp_2p_3}\boldsymbol{q}_1+S_{\triangle pp_3p_1}\boldsymbol{q}_2+S_{\triangle pp_1p_2}\boldsymbol{q}_3}{S_{\triangle p_1p_2p_3}} \tag{12}$$

f 在 u 和 v 两个方向上的偏导数为

$$\left.\begin{aligned}\frac{\partial f}{\partial u}&=\frac{(v_2-v_3)\boldsymbol{q}_1+(v_3-v_1)\boldsymbol{q}_2+(v_1-v_2)\boldsymbol{q}_3}{2S_{\triangle p_1p_2p_3}}\\ \frac{\partial f}{\partial v}&=\frac{(u_3-u_2)\boldsymbol{q}_1+(u_1-u_3)\boldsymbol{q}_2+(u_2-u_1)\boldsymbol{q}_3}{2S_{\triangle p_1p_2p_3}}\end{aligned}\right\} \tag{13}$$

f 的 3×2 雅可比矩阵 $\left[\frac{\partial f}{\partial u}\ \frac{\partial f}{\partial v}\right]$ 的最大和最小奇异值为

$$\left.\begin{aligned}\sigma_{\max}&=\sqrt{\frac{1}{2}((a+c)+\sqrt{(a-c)^2+4b^2})}\\ \sigma_{\min}&=\sqrt{\frac{1}{2}((a+c)-\sqrt{(a-c)^2+4b^2})}\end{aligned}\right\} \tag{14}$$

其中：$a=\frac{\partial f}{\partial u}\cdot\frac{\partial f}{\partial u}$，$b=\frac{\partial f}{\partial u}\cdot\frac{\partial f}{\partial v}$，$c=\frac{\partial f}{\partial v}\cdot\frac{\partial f}{\partial v}$.

$\sigma_{\max}$ 和 $\sigma_{\min}$ 分别表示仿射变换 f 对于平面上单位长度产生的最大和最小缩放倍数. 对于三角形的几何形变，拉伸和收缩的测定标准需要保持一致. 因此，对三角形的形变量 d 定义为[27]

$$d=\max\left(\sigma_{\max},\frac{1}{\sigma_{\min}}\right) \tag{15}$$

这样就确保了 $d\geqslant1$，当且仅当 f 为全等变换时等号成立.

对于三角形网格的参数化，可以计算出参数域上每一个三角形与原网格上对应三角形之间的形变量，从而得到参数化对网格上所有三角形造成的平均形变量 d_{mean}、最大形变量 $d_{\max}$(形变的最坏情况)以及形变量的标准差 d_{std}(形变的波动情况). 具体计算结果如表 3 所示.

表 3　参数化对三角形网格造成的形变量比较

加权方案	形变量	环形山	甲壳虫汽车	奶牛	人脸	均值
均匀拉普拉斯权重	d_{mean}	2.03	3.11	6.08	7.45	4.67
	$d_{\max}$	6.82	18.16	52.10	96.17	43.31
	d_{std}	0.84	2.01	5.28	12.63	5.19
拉普拉斯–贝尔特拉米权重	d_{mean}	1.40	4.70	5.52	2.77	3.60
	$d_{\max}$	2.94	170.35	65.96	37.47	69.18
	d_{std}	0.17	12.00	4.98	5.19	5.59
中值权重	d_{mean}	1.39	2.49	5.22	3.09	3.05
	$d_{\max}$	2.92	29.82	56.06	47.21	34.00
	d_{std}	0.18	2.87	4.55	6.06	3.42

表 3 的结果表明，以均匀拉普拉斯权重作为比较基准，拉普拉斯–贝尔特拉米权重和中值权重均能减少参数化对三角形产生的平均形变，从而减少参数化的失真. 拉普拉斯–贝尔特拉米权重虽然能够整体上使参数化保真，但产生的最大形变高. 与之相比，中值权重更好地控制了形变的最坏情况和形变的波动情况.

4　结束语

研究了基于重心映射的三角形网格参数化的理论来源和实现方法，描述了实现重心映射法所采用的三角形网格半边数据结构，分析并比较了实现重心映射法所采用的 3 种加权方案：均匀拉普拉斯权重、拉普拉斯–贝尔特拉米权重和中值权重，给出了使用这 3 种权重的重心映射参数化在一些具体物体的三角形网格上的实验结果，并计算了参数化对网格上三角形产生的形变量. 通过对比发现，拉普拉斯–贝尔特拉米权重和中值权重能产生较低的三角形平均形变量，从而更好地保持原三角形网格的形状与特征，减少参数化失真. 而中值权重作为最优

加权方案,能更进一步地减少三角形的最大形变以及形变量的波动. 基于重心映射的参数化的代码与实现可参见:https://isaacguan.github.io/projects/tutte-embedding/.

参考文献:

[1] 郭风华, 张彩明, 焦文江. 网格参数化研究进展[J]. 软件学报, 2016, 27(1): 112-135.
Guo Fenghua, Zhang Caiming, Jiao Wenjiang. Research progress on mesh parameterization[J]. Journal of Software, 2016, 27(1): 112-135.

[2] Lévy B. Constrained texture mapping for polygonal meshes[C]//Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press, 2001: 417-424.

[3] Cheng Peng, Wang Jiaye, Miao Chunyan, et al. Constrained texture mapping via approximate Voronoi base domain[C]//Proceedings of Computer Graphics International 2018. New York: ACM Press, 2018: 19-22.

[4] Qian Kun, Li Yinghua, Su Kehua, et al. A measure-driven method for normal mapping and normal map design of 3D models[J]. Multimedia Tools and Applications, 2018, 77(24): 31969-31989.

[5] Biermann H, Martin I, Bernardini F, et al. Cut-and-paste editing of multiresolution surfaces[J]. ACM Transactions on Graphics, 2002, 21(3): 312-321.

[6] Eyiyurekli M, Breen D E. Detail-preserving level set surface editing and geometric texture transfer[J]. Graphical Models, 2017, 93: 39-52.

[7] Lévy B. Dual domain extrapolation[J]. ACM Transactions on Graphics, 2003, 22(3): 364-369.

[8] Kwok T H, Zhang Yunbo, Wang C C L. Efficient optimization of common base domains for cross parameterization [J]. IEEE Transactions on Visualization and Computer Graphics, 2012, 18(10): 1678-1692.

[9] Tutte W T. Convex representations of graphs[J]. Proceedings of the London Mathematical Society, 1960, 10 (3): 304-320.

[10] Sheffer A, de Sturler E. Parameterization of faceted surfaces for meshing using angle-based flattening[J]. Engineering with Computers, 2001, 17(3): 326-337.

[11] Sheffer A, Lévy B, Mogilnitsky M, et al. ABF + +: fast and robust angle based flattening[J]. ACM Transactions on Graphics, 2005, 24(2): 311-330.

[12] Lévy B, Petitjean S, Ray N, et al. Least squares conformal maps for automatic texture atlas generation[J]. ACM Transactions on Graphics, 2002, 21(3): 362-371.

[13] Hormann K, Greiner G. MIPS: an efficient global parametrization method[M]//Laurent P J, Schumaker L L, Sablonniere P. Curve and Surface Design: Saint-Malo 1999. Nashville: Vanderbilt University Press, 2000: 153-162.

[14] Fu Xiaoming, Liu Yang, Guo Baining. Computing locally injective mappings by advanced MIPS[J]. ACM Transactions on Graphics, 2015, 34(4): 71.

[15] Lipman Y. Bounded distortion mapping spaces for triangular meshes[J]. ACM Transactions on Graphics, 2012, 31(4): 108.

[16] Schüller C, Kavan L, Panozzo D, et al. Locally injective mappings[J]. Computer Graphics Forum, 2013, 32(5): 125-135.

[17] Sawhney R, Crane K. Boundary first flattening[J]. ACM Transactions on Graphics, 2017, 37(1): 5.

[18] Sawhney R. geometry-processing-js[EB/OL]. (2018-03-11)[2018-08-30]. https://geometrycollective.github.io/geometry-processing-js/.

[19] Mäntylä M. An introduction to solid modeling[M]. New York: Computer Science Press, 1988.

[20] Kettner L. Using generic programming for designing a data structure for polyhedral surfaces[J]. Computational Geometry, 1999, 13(1): 65-90.

[21] 张应中, 谢馥香, 罗晓芳, 等. 采用半边编码的三角网格拓扑数据结构[J]. 计算机辅助设计与图形学学报, 2016, 28(2): 328-334.
Zhang Yingzhong, Xie Fuxiang, Luo Xiaofang, et al. A topological data structure using coding of half-edges for triangle meshes[J]. Journal of Computer-Aided Design & Computer Graphics, 2016, 28(2): 328-334.

[22] Taubin G. A signal processing approach to fair surface design[C]//Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press, 1995: 351-358.

[23] Meyer M, Desbrun M, Schröder P, et al. Discrete differential-geometry operators for triangulated 2-manifolds [M]//Hege H C, Polthier K. Visualization and Mathematics III. Heidelberg: Springer, 2003: 35-57.

[24] Floater M S. Mean value coordinates[J]. Computer Aided Geometric Design, 2003, 20(1): 19-27.

[25] Rivara M C. Mesh refinement processes based on the generalized bisection of simplices[J]. SIAM Journal on Numerical Analysis, 1984, 21(3): 604-613.

[26] 管焱然, 管有庆. 基于OpenCV的仿射变换研究与应用[J]. 计算机技术与发展, 2016, 26(12): 58-63.
Guan Yanran, Guan Youqing. Research and application of affine transformation based on OpenCV[J]. Computer Technology and Development, 2016, 26(12): 58-63.

[27] Sorkine O, Cohen-Or D, Goldenthal R, et al. Bounded-distortion piecewise mesh parameterization[C]//Proceedings of the Conference on Visualization'02. Washington, D. C.: IEEE Computer Society, 2002: 355-362.