圆组填充算法驱动的平面马赛克模拟

张 凯[1,3], 柯 颖[2], 曹 娟[2)*], 陈中贵[1,3]

[1] (福建省智慧城市感知与计算重点实验室(厦门大学) 厦门 361005)
[2] (厦门大学数学科学学院 厦门 361005)
[3] (厦门大学信息科学与技术学院 厦门 361005)
(juancao@xmu.edu.cn)

摘 要: 为了生成不规则嵌片排列紧凑的马赛克图案, 提出一种基于圆组排列的平面马赛克模拟方法. 首先借助嵌片多边形的直骨架得到一组逼近嵌片轮廓的圆; 然后以圆半径的平方为权值, 在平面上生成关于圆组的 Power 图, 使每组圆各自对应一个 Power 区域; 最后采用松弛法, 将圆组在其对应 Power 区域内尽可能增长到最大. 通过不断迭代生成 Power 图和放大圆组, 最后得到嵌片紧凑排列的结果. 实验结果表明, 该方法得到的马赛克图案有较高的覆盖率, 能适应不同嵌片, 具有较强的鲁棒性和灵活性.

关键词: 马赛克模拟; 圆组填充; Power 图
中图法分类号: TP391.41 **DOI:** 10.3724/SP.J.1089.2018. 16767

Circle Group Packing Algorithm for Mosaic Synthesis

Zhang Kai[1,3], Ke Ying[2], Cao Juan[2)*], and Chen Zhonggui[1,3]

[1] (*Fujian Key Laboratory of Sensing and Computing for Smart City, School of Information Science and Engineering, Xiamen University, Xiamen* 361005)
[2] (*School of Mathematical Sciences, Xiamen University, Xiamen* 361005)
[3] (*School of Information Science and Engineering, Xiamen University, Xiamen* 361005)

Abstract: To produce the mosaic patterns with compact arrangements of irregular tiles, this paper proposes a flexible method for mosaic synthesis, which is based on a circle group packing algorithm. First, we create a group of circles to approximate the contour of each tiles, resorting to its straight skeleton. Next, the power diagram is generated by setting the weight of each site as each circle's squared radius, and each circle group has its corresponding power region. Finally, all circle groups are magnified in their power regions by a relaxation method. By generating the power diagram and magnifying the circle groups iteratively, we get a compact packing of the tiles. The experiment results show that, our method is able to generate mosaic patterns with high coverages and arbitrary tile shapes, and it has good robustness and flexibility.

Key words: mosaic synthesis; circle group packing; power diagram

马赛克是一种广为人知的艺术形式, 它将许多片状材料拼在一起, 创造出令人惊奇的作品. Opus Palladianum[1]是用于生成马赛克的一种常见技术, 这种技术将不规则的嵌片拼接在一起, 形成碎石状的效果. 不同于其他技术产生的规则马赛克风格, Opus Palladianum[1]形成的不规则风格

收稿日期: 2017-07-07; 修回日期: 2017-09-30. 基金项目: 国家自然科学基金(61472332); 福建省自然科学基金(2018J01104); 中央高校基本科研业务费专项基金(20720150002). 张 凯(1994—), 男, 硕士研究生, 主要研究方向为计算机图形学; 柯 颖(1992—), 女, 硕士研究生, 主要研究方向为计算机图形学; 曹 娟(1983—), 女, 博士, 副教授, 硕士生导师, CCF 会员, 论文通讯作者, 主要研究方向为 CAGD&CG; 陈中贵(1982—), 男, 博士, 副教授, 硕士生导师, CCF 会员, 主要研究方向为计算机图形学、数字几何处理.

更具有现代感；但因其不规则的特性，想得到好的结果往往需要耗费极大精力，尤其是当嵌片数量较多时还要保证它们之间的相对位置合理.

进行平面马赛克模拟时，首先应确保嵌片之间不能有重叠部分，其次要考虑嵌片外部轮廓之间是否适配以及模拟效果的美观性. Hu 等[2]用不规则多边形拟合嵌片的外部轮廓，在给定平面生成关于这些多边形的 Voronoi 图；原目标是提高全体嵌片在平面上的覆盖率，现只需提高每块嵌片在其对应 Voronoi 区域的覆盖率，即让每个多边形在其 Voronoi 区域内独立增长，由此将复杂问题简单化.

为了进一步简化程序，提高算法效率，本文不直接对多边形进行操作，而是用一组圆近似代替多边形. 圆逼近多边形大体上有 2 种思路：一是在多边形内部填充多个圆，规定圆不能超出多边形边界，且圆与圆之间不能发生重叠；二是用多个圆覆盖住多边形，允许圆与圆之间发生重叠. 本文采用了第 2 种思路.

基于 Hu 等[2]的算法中的连续优化部分，本文以圆的半径平方为权值形成平面的 Power 图，近似代替多边形的每组圆各自独立地在对应的 Power 区域里增长. 相对于直接使用多边形，使用圆组能够更好地优化嵌片之间的相对位置，提高平面的覆盖率. 由于形成 Power 图使用的权值与圆的大小相关，本文算法在优化多边形的位置时更加灵活. 相对于直接使用多边形，该算法能够更好地优化嵌片的相对位置，提高嵌片在平面上的覆盖率.

1　相关工作

1.1　不规则多边形填充

不规则多边形填充问题是指在一个给定区域内放置若干不规则多边形的问题，其目的是寻找最紧凑的排列结构，使得空间利用率最大. 不规则多边形填充问题的研究历史很长，一些方法在工业界也得到了广泛应用. 填充问题是 NP-hard 问题[3]. Dowsland 等[4]和 Bennell 等[5]在不同时期分别论述了二维不规则多边形填充问题的进展. 已有的算法大多采用启发式算法; Lu 等[6]将 Power 图应用到圆填充多边形问题；Hu 等[2]将填充区域划分为与不规则多边形有关的 Voronoi 区域，使得不规则多边形填充问题成为不规则多边形包含问题；Reinert 等[7]使用重心 Voronoi 镶嵌和 GPU 计算，生成能实时交互的艺术化填充排列布局.

1.2　圆覆盖多边形

圆覆盖多边形问题是一个经典的数学问题. Macri 等[8]提出使用八叉树结构生成圆的算法，虽然这种算法可以适用于不规则的多边形，但生成圆的个数也较多. Zhang 等[9]利用多边形的外部轮廓生成圆，能很好地覆盖多边形的外部轮廓，也能显著减少生成圆的个数. Rocha 等[10]利用多边形的中轴线和外部轮廓生成能够满足特定要求的圆，并将生成结果应用到嵌套问题中. 用圆覆盖多边形本质上是为了利用圆的某些特性或避开多边形的某些缺陷，因此，在实际应用中选择的算法应该是为了满足问题的特定需求.

1.3　计算模拟马赛克

马赛克作为一种常见的艺术表现形式，有悠久的历史，人们一直试图用计算机自动模拟出马赛克效果. Liu 等[11]提出一种基于图片分割的自动生成马赛克的算法. Kim 等[12]通过区域划分和颜色匹配，将图片填充到指定区域内以产生马赛克效果. 在最小化能量函数以得到更紧致的排列期间，图片会发生形变. 陈中贵等[13]基于测地距离下的 Voronoi 图结构，提出一种保特征的自适应马赛克图像生成方法. Zhang 等[14]提出一种基于多边形镶嵌和颜色匹配的算法，依赖预设的数据集生成与输入图片相适应的马赛克效果. Lai 等[15]使用曼哈顿度量的粒子优化算法，在三维网格表面生成由等尺寸的矩形嵌片构成的马赛克效果. 律睿憋等[16]通过对图像局部细节层次进行三角剖分，实现了一种基于模糊色彩集分类的马赛克图像渲染技术. Battiato 等[17]基于方向引导线和图片切割，生成能够紧密贴合图片颜色变化明显部分的边界.本文实验中不允许图片切割和扭曲，只允许图片大小发生变化.

2　问题描述

马赛克模拟问题的输入包括平面 M 和一组不规则源物件 $S=\{S_i\}_{i=1}^{m}$，其中，M 通常为平面待填充区域. 对于每一个源物件 S_i，用多个可重叠的圆构成的圆组 C_i 从内部逼近物件的外部轮廓. 由此，本文问题的输入变为平面待填充区域 M、初始位置点集 $O=\{O_i\}_{i=1}^{n}$ 和近似代替不规则源物件的圆组 $C=\{C_i\}_{i=1}^{n}$，其中，初始位置点集 $O=\{O_i\}_{i=1}^{n}$ 为 M 上 n 个位置点，位置由用户给定，与源物件个数不一定相同；每个圆组的重心与初始点 O_i 重合.

本文的目标是寻找使得平面覆盖率尽可能高的多边形分布 $\{(P_i,\Theta_j)\}_{i=1}^{m}$，其中，$P_i$ 是源物件 $S=\{S_i\}_{i=1}^{m}$ 的外部轮廓多边形；Θ_i 是描述 P_i 的一组结构参数，即它在平面上的位置坐标和方向角. 嵌片在迭代变换状态的过程中与其他嵌片不重叠，因此用圆组代替嵌片进行优化时也应该确保同一组圆的相对位置不变，不同组圆之间不能发生重叠.

假设给定的嵌片数目为 n，即位置点的个数. 为了能与其他算法的结果作比较，平面的覆盖率定义为多边形的面积之和与平面面积的比值. 本文算法中，初始嵌片参数会根据嵌片的数量和大小进行调整，防止在初始情况下发生重叠. 在优化过程中，本文不允许嵌片在迭代时缩小，因此增长因子最小为 1.0.

3 算法概述

本文采用松弛迭代算法调整嵌片的排列，包括位置、方向角和缩放比例，使平面达到尽可能高的覆盖率；同时，使用 Power 图划分平面，由于权值与圆组的半径相关，能够比 Voronoi 图更灵活地匹配大小发生变化的嵌片. 在优化过程中，Power 图会依据嵌片的相关参数更新 Power 区域.

算法 **1**. 马赛克模拟算法

输入. 平面待填充区域 M，决定算法停止的临界值 α，一组源物件 $S=\{S_1,S_2,\cdots,S_m\}$

输出. M 上排列紧凑的嵌片.

Step1. 执行圆组填充算法，生成初始圆组 $C=\{C_{1,j},C_{2,j},\cdots,C_{m,j}\}$.

Step2. 初始化圆，使其不重叠地放置在 M 上.

Step3. 生成 Power 图.

Step4. 初始化嵌片.

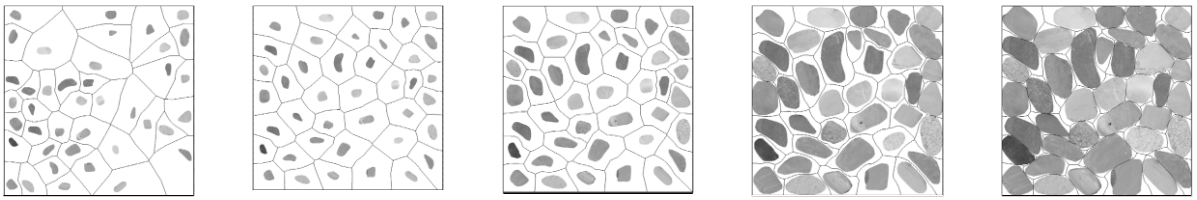
Step5. do

更新 Power 图.

执行嵌片排列算法.

while 增长因子平均值 $s_{\text{average}}<\alpha$

图 1 所示为算法 1 流程及最终结果.



a. 初始化 b. 初始松弛迭代 c. 算法迭代 5 次 d. 算法迭代 10 次 e. 迭代最终结果

图 1 本文算法结果

4 嵌片排列优化算法

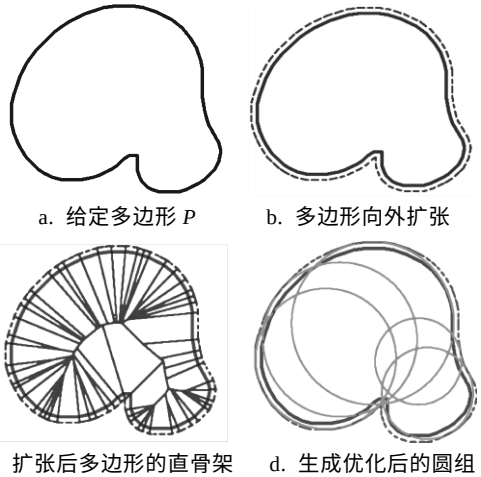
4.1 圆组填充算法

已有的马赛克模拟算法用不规则多边形表示嵌片外部轮廓，或者用单个圆粗略地拟合嵌片. 相对于用多边形近似物体的边界，以数目较少的圆近似代替多边形，能够更灵活地调整逼近的精度.

本文提出的圆组填充算法的流程如下：首先对于给定的多边形 P，将其边界向外扩张 τ 个单位，记扩张边界后的多边形为 P'. 然后求出 P' 的直骨架[18]，并在每条直骨架上生成 k 个与 P' 对应边界相切的圆. 最后判断所有圆两两之间的相互关系：若二者相离或外切，不做处理；若二者是包含或内切关系，删除半径较小的圆；若二者相交，假设 2 个圆半径分别为 r_1 和 r_2，圆心距为 d，当

$$\frac{d}{\max(r_1,r_2)}<U_1 \text{ 或 } \frac{d+\min(r_1,r_2)}{\max(r_1,r_2)}<U_2$$

时，删除半径较小的圆. 其中，τ 是圆半径减去圆心到多边形 P 上相应边界距离所得值，用于控制圆逼近多边形的精度；k 与每条直骨架的长度有关，即每条直骨架对应的 k 值各不相同；U_1 和 U_2 为常数，控制最终生成圆的个数. 这几个参数的值均由用户来设定.



a. 给定多边形 P b. 多边形向外扩张 c. 扩张后多边形的直骨架 d. 生成优化后的圆组

图 2 圆组填充算法流程图

4.2　初始化嵌片位置

为防止嵌片在初始情况下发生重叠，同时保证嵌片在平面分布的多样性，本文处理方式如下：

Step1. 在基础表面 M 上随机生成 n 个位置点，通过文献[19]算法优化位置点的分布，使得位置点分布适度均匀，保证位置点之间的距离适中，防止 2 个位置点距离过近，且位置点的分布具有随机性.

Step2. 如果源物件个数多于 n，选取前 n 个源物件；如果源物件个数少于 n，循环读取源物件，直到读取的源物件个数达到 n.

Step3. 根据输入的源物件大小和位置点个数 n 调整嵌片在平面上的初始大小.

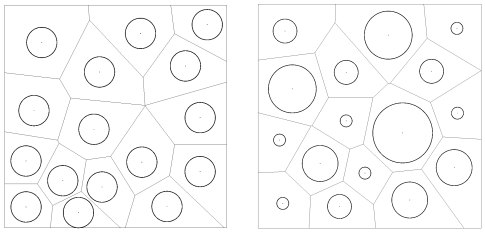
假设嵌片输入时的面积为 $\{A_i\}_{i=1}^n$，平面面积为 A_{M}，按照 $A_i^0=\mu\times A_i\times\sqrt{\left(n\cdot A_{\mathrm{M}}\right)\Big/\left(n\cdot\sum_{i=1}^n A_i\right)}$ 调整嵌片在平面的初始大小. 其中，A_i^0 是第 i 个嵌片在平面上的初始大小；μ 是系数常量，根据经验设置为 0.3. 需要注意的是：接下来的松弛步骤可能会使最终的嵌片大小比输入时的元物件大，即本文允许最终的嵌片与实际输入的大小不同.

4.3　Power 图

Power 图是 Voronoi 图的推广[20]，是顶点带权的 Voronoi 图. 在 D 维空间 $\mathbb{R}^D$ 中有一系列相互独立的点 $V=\{v_1,\cdots,v_m\}$，对每个点 $v_i\in\mathbb{R}^D$ 赋予权值 $w_i\geqslant 0$. $\mathbb{R}^D$ 上任意 2 点 v 和 v_i 的 Power 距离定义为 $d_{\mathrm{w}}(v,v_i)=\|v-v_i\|^2-w_i$，其中，$\|\cdot\|$ 表示欧几里得距离. 对于 $\mathbb{R}^D$ 上满足到点 v_i 的 Power 距离小于等于到 $v_j\in V-\{v_i\}$ 的点 v 的集合 v_i^{w}，即 $v_i^{\mathrm{w}}=\{v\in\mathbb{R}^D\mid d_{\mathrm{w}}(v,v_i)\leqslant d_{\mathrm{w}}(v,v_j),\forall v_j\in V\}$，可将 $\mathbb{R}^D$ 划分为若干区域，这些区域的集合即为 V 在 $\mathbb{R}^D$ 上的 Power 图.

在二维空间 $\mathbb{R}^2$ 中，将权值为 w_i 的点 v_i 看做圆心为 v_i、半径为 $\sqrt{w_i}$ 的圆. 对于 $\mathbb{R}^2$ 上的有界区域 Ω，Ω 上有限点集 $V=\{v_1,\cdots,v_m\}$ 的权值分别为 $W=\{w_1,\cdots,w_m\}$，其 Power 图表示为 $\Omega_{\mathrm{Power}}=v_{\mathrm{Power}}^{\mathrm{w}}\cap\Omega$. 可以看出，Voronoi 图是权值相同的点集的 Power 图. 图 3 所示为 Voronoi 图和 Power 图的对比示意图.

在 Voronoi 图中，不同顶点的 Voronoi 区域边界是 2 点之间的垂直平分线. 而在 Power 图中，顶点带有权值，Power 区域的边界为 2 点之间的垂直分割线，权重值大的点离垂直分割线远，权重值小的点离分割线近，点的权重值越大，其 Power 区域越大. 可以看出，相对于 Voronoi 图，Power 图能够适应不同权值的圆，区域划分更灵活. 本文不允许圆不在其 Power 区域里，即不允许 2 个圆包含. 因此能在平面 M 上得到合适的 Power 图.

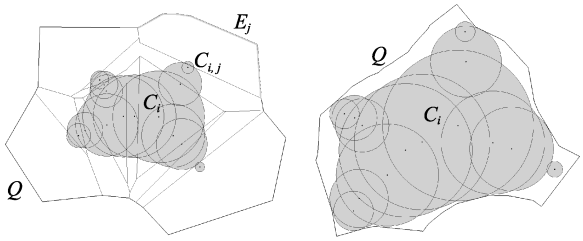


a. Voronoi 图　　b. Power 图

图 3　Voronoi 图和 Power 图对比示意图

4.4　嵌片填充最优化

要使嵌片在平面 M 上的覆盖率最大，需要得到达到最大覆盖率时每一块嵌片的位置和方向角. 由于本文允许嵌片和实际输入时的大小不同，因此还需要求出嵌片相对于实际输入的增长率. 当覆盖率最大时，每一块嵌片在其对应 Power 区域的覆盖率一定是最大的，因此，解决所有嵌片在 M 上覆盖率最大的问题可以转化为圆组在其对应 Power 区域里的极限填充问题. 令 P_i 表示第 i 块嵌片外部轮廓的多边形，Q_i 为对应的待填充的 Power 区域. 用圆组 $C_i=\{C_{i,1},\cdots,C_{i,j},1<j<J\}$ 逼近多边形 P_i，其中 J 是 C_i 中圆的个数. 计算圆组 C_i 在 Q_i 里的 Power 图，令 $\{e_1,\cdots,e_l\}$ 为 Q_i 的边. 令 E_j 为 Q_i 上与圆 $C_{i,j}$ 的 Power 图相交的边的集合，如图 4 所示. $C_{i,j}$ 的位置、方向和大小会在迭代过程中发生变化，因此 $C_{i,j}$ 的 Power 图和 E_j 也会随之更新.



a. C_i的初始状态　　b. 本文算法的最终结果

图 4　带约束的极限填充优化示意图

本文方程依赖于 C_i 的局部坐标系. C_i 的变化情况可以表示为位移 $\boldsymbol{t}=(t_1,t_2)$、相对于原始方向的旋转角 θ 和增长因子 s. 当 C_i 发生移动时，C_i 的变化后的情况 C_i' 可由与 (s,θ,t_1,t_2) 有关的函数表示为

$$C_i'=C_i(s,\theta,t_1,t_2),1\leqslant i\leqslant n.$$

根据以上内容，可以将不规则多边形的极限填充问题转化为一个带约束的非线性优化问题

$$\begin{aligned}&\text{maximize} \quad f(s,\theta,t_1,t_2)=s\\&\text{subject to} \quad \frac{\left(o'_{i,j}-\boldsymbol{e}_k^0\right)}{|\boldsymbol{e}_k|}-r'_{i,j}\geqslant 0\\&1\leqslant i\leqslant n,1\leqslant j\leqslant J,\boldsymbol{e}_k\in E_j,-\frac{\pi}{6}\leqslant\theta\leqslant\frac{\pi}{6}.\end{aligned}$$

其中，$o'_{i,j}$ 是变化后 $C'_{i,j}$ 的圆心；$\boldsymbol{e}_k^0$ 是有向边 $\boldsymbol{e}_k$ 的起始点；$r'_{i,j}$ 是变化后的 $C_{i,j}$ 的半径. 可以看出，对目标函数进行优化就是要寻找合适的增长因子 s，使得 C_i 的面积足够大；同时，只能在 C_i 的 Power 图内与 Power 图内切. 为了防止出现较大程度的旋转，本文限制旋转角的变化范围，根据经验设定为 $[-\pi/6,\pi/6]$，使得结果保持稳定.

本文使用 Knitro 软件库[21]提供的内部点算法解决上述的非线性优化问题. (s,θ,t_1,t_2) 的初始值设定为 $(1,0,0,0)$.

算法 **2**. 嵌片排列算法

输入. 平面待填充区域 M，M 上初始圆组 $\{C_1,C_2,\cdots,C_n\}$，最大迭代次数 $h_{\max}$.

输出. M 上的嵌片排列.

Step1. $h_{\text{curr}}\leftarrow 0$.

Step2. do

计算 Power 区域 $(Q_1,\cdots,Q_n)$.

for 每个圆组 C_i do

找到 C_i 在 s 里的最大副本，得到对应 $(s_i,\theta_i,t_{1,i},t_{2,i})$

end for

while $h_{\text{curr}}<h_{\max}$

Step3. 得到比初始更紧凑的嵌片排列.

5 实验与结果分析

本文使用 C++作为程序的编程语言，所有的结果都是在 Intel® Core™ 2.9 GHz CPU 和 12 GB 内存台式机上得到的. 在嵌片填充优化过程中使用 OpenMP 自动并行运行代码. 程序运行时间主要与嵌片的数量有关. 本文算法对于 100 个嵌片生成最终的马赛克拟合结果只需 5 s.

为防止嵌片增长过快导致无法充分优化嵌片结构，本文为增长因子 s 设置了一个限定值 s_{b}，当 $s>s_{\text{b}}$ 时，令 $s=s_{\text{b}}$. 通过减缓嵌片的增长速度，使得算法的执行更有效，嵌片最终的分布更加合理. 图 5 所示为设置限定值能够使增长因子增长过程更稳定，同时能显著提高嵌片在平面的覆盖率.

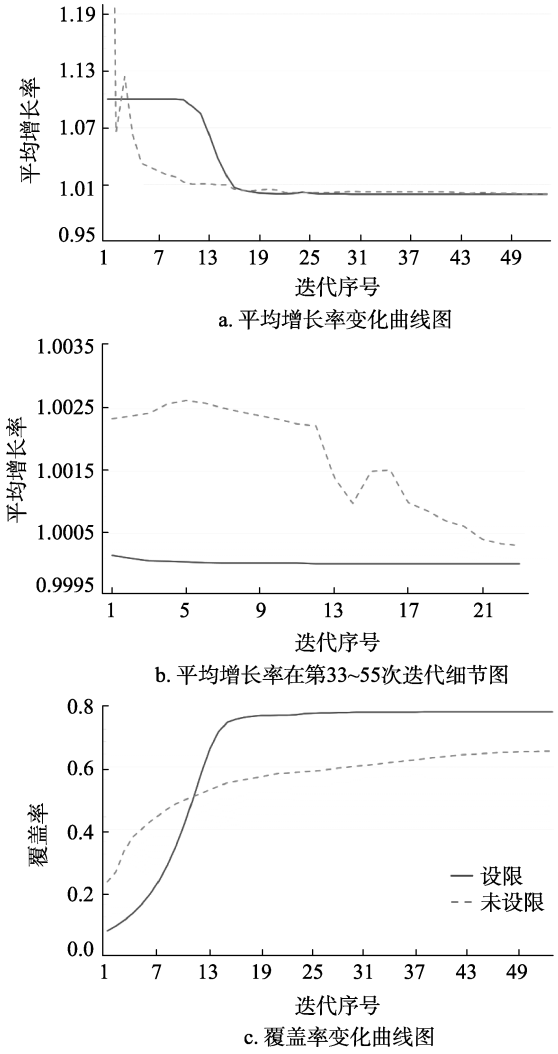
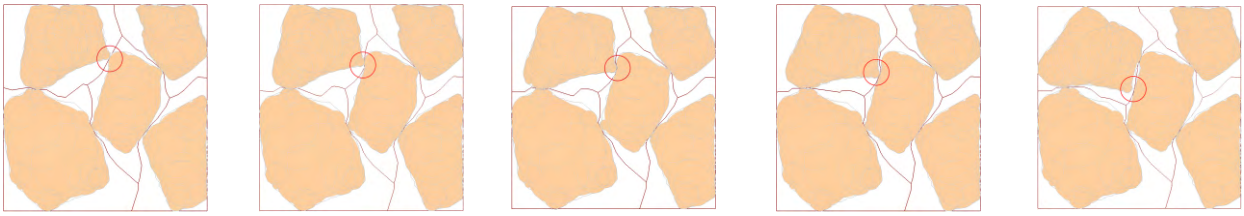


图 5 设置限定值与不设置限定值的比较

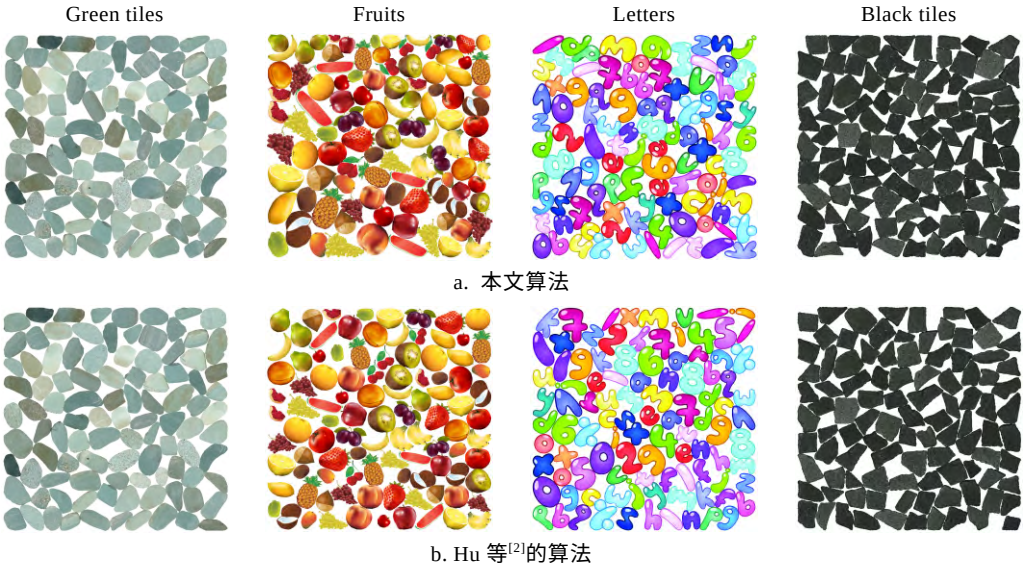
相对于通常算法在多边形外部轮廓上设置样本点，本文使用圆组近似代替多边形，能够避免某些情况下多边形之间相互阻碍增长. 从图 6 可以看出，在优化过程中，使用圆逼近多边形能够避免使用不规则多边形时出现的边缘锯齿，导致 2 个多边形过早贴近，未能充分优化嵌片分布结构的情况. 图 7 所示为本文算法和 Hu 等[2]算法中连续优化部分的结果. 从 Fruits 图像可以看出，本文算法能够让嵌片之间贴合得更紧密. 如图 8 所示，本文算法与 Hu 等[2]的算法中连续优化部分对比发现，本文算法在达到相同的覆盖率的情况下，所需时间比 Hu 等[2]的算法中连续优化部分更少. 需要注意的是：为了保证公平性，本文为 Hu 等[2]的算法中连续优化部分也设置了限定值 s_{b}，并未使用其算法中的组合优化部分. 本文算法不仅所需时间更短，而且每次迭代的时间波动较少，迭代更加稳定. 在嵌片相互贴近时，Hu 等[2]的算法需要增加样本点

保证 Voronoi 图划分稳定，因此迭代后期的时间会增加；而本文算法采用圆组逼近多边形，2 个圆圆心不可能互相贴近，能够很好地避免样本点靠得太近导致 Voronoi 划分不稳定．本文算法保留了灵活性，使用者可以选定不同的覆盖率，以满足自己的要求，不同覆盖率下的结果如图 9 所示．



a．圆组在迭代过程中卡住　b．迭代后脱离卡住状态　c．再次卡住　d．继续迭代，脱离卡住状态　e．达到稳定状态

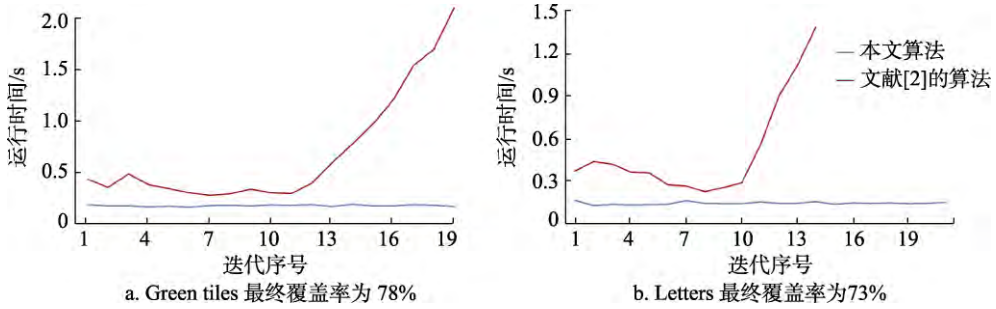
图 6　圆组避免卡住的过程图



a．本文算法

b. Hu 等[2]的算法

图 7　2 种算法的结果比较



a. Green tiles 最终覆盖率为 78%　　b. Letters 最终覆盖率为73%

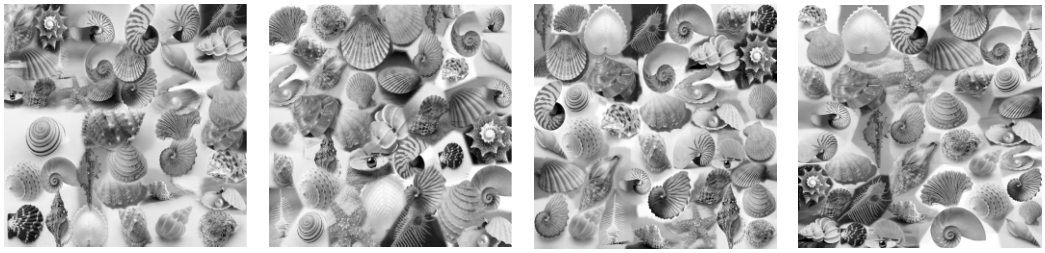
图 8　2 种算法在时间上的比较

a．覆盖率为 54.3%　b．覆盖率为 63.3%　c．覆盖率为 76.2%　d．覆盖率为 80.0%

图 9　不同覆盖率的结果

图片是非常重要的展现信息的方式,如何将图片合理地拼贴在一起已成为目前研究的热点. AutoCollage[22]将能量最小化算法应用到拼贴问题上,以突出图片的显著性区域. Yu 等[23]把拼贴问题视为区域划分问题,通过启发式搜索算法测度图片的显著性信息,将图片的显著性信息程度与圆的半径相对应,利用基于Power图的圆填充算法将区域划分为若干部分,每部分的大小与其对应的显著性信息相关. Jing[24]在 Yu 等[23]的基础上直接利用显著性区域的轮廓作为输入,结合 Hu 等[2]的算法,将区域划分为更贴合显著性区域的若干部分.

不同于 Jing[24]的算法,本文在 Yu 等[23]算法的基础上,通过获得包含显著性区域的最小凸多边形,用圆组逼近凸多边形,利用嵌片排列算法优化圆组,将区域划分为若干区域. 相对于 Yu 等[23]的算法,本文算法能更好地突出图片的显著性信息. 相对于 Jing[24]的算法,本文算法对图片的显著性信息提取要求不高,同时由于圆组填充算法能很好地覆盖多边形区域,图片的显著性信息能更好地保留. 图 10 所示为本文算法和上述算法的比较结果.



a. Microsoft AutoCollage b. Yu 等[23]的算法 c. Jing[24]的算法 d. 本文算法

图 10 不同算法处理 Photo Collage 的结果

6 结 语

本文采用 Hu 等[2]的算法中连续优化部分,并对其进行了改进,将圆组填充算法应用于马赛克模拟问题,提出一种马赛克模拟算法. 本文利用圆组代替不规则多边形,由原来样本点之间的连接关系转变为圆之间的覆盖关系. 为有效地减少逼近多边形所用圆的个数以及提高逼近精度,本文算法允许圆部分重叠,但不允许圆发生包含;同时,采用Power图划分平面,使得划分的区域能很好地适应圆的大小. 相对于 Hu 等[2]的算法中连续优化部分,本文算法有效地优化了嵌片分布结构,进一步提高覆盖率;同时,也能灵活地调整覆盖率,满足使用者的不同要求.

当然,本文算法也存在不足之处:(1) 输入数据的质量好坏对覆盖率有较大影响,且圆逼近多边形的精度对覆盖率有一定影响;(2) 为获得更高的覆盖率,本文结果的嵌片大小与输入的大小可以不一致,也就是说,本文的结果同实际存在一定差异,但这并不影响该算法在实际中的应用. 另外,本文算法目前只考虑到二维平面的情况,希望未来在三维情况下有更多的应用. 对于马赛克拟合问题,本文目前仅探索如何提高覆盖率,而马赛克的其他特性,如颜色分布,也是值得探索的问题.

参考文献(References):

[1] Jacobsen R A. Mosaics for the first time[M]. New York: Sterling Publishing Company, Inc., 2005: 17-18

[2] Hu W C, Chen Z G, Pan H, *et al*. Surface mosaic synthesis with irregular tiles[J]. IEEE Transactions on Visualization & Computer Graphics, 2016, 22(3): 1302-1313

[3] Fowler R J, Paterson M S, Tanimoto S L. Optimal packing and covering in the plane are NP-complete[J]. Information Processing Letters, 1981, 12(3): 133-137

[4] Dowsland K A, Dowsland W B. Solution approaches to irregular nesting problems[J]. European Journal of Operational Research, 1995, 84(3): 506-521

[5] Bennell J A, Oliveira J F. A tutorial in irregular shape packing problems[J]. The Journal of the Operational Research Society, 2009, 60(Sup1): S93-S105

[6] Lu L, Choi Y K, Sun F, *et al*. Variational circle packing based on power diagram[R]. Hong Kong: The University of Hong Kong. Department of Computer Science, 2011

[7] Reinert B, Ritschel T, Seidel H P. Interactive by-example design of artistic packing layouts[J]. ACM Transactions on Graphics, 2013, 32(6): Article No.218

[8] Macri M, De S, Shephard M S. Hierarchical tree-based discretization for the method of finite spheres[J]. Computers & Structures, 2003, 81(8-11): 789-803

[9] Zhang W H, Zhang Q. Finite-circle method for component approximation and packing design optimization[J]. Engineering Optimization, 2009, 41(10): 971-987

[10] Rocha P, Rodrigues R, Miguel Gomes A, *et al*. Two-phase approach to the nesting problem with continuous rotations[J]. IFAC-PapersOnline, 2015, 48(3): 501-506

[11] Liu Y, Veksler O, Juan O. Simulating classic mosaics with graph cuts[M] //Lecture Notes in Computer Science. Heidelberg: Springer, 2007, 4679: 55-70

[12] Kim J, Pellacini F. Jigsaw image mosaics[J]. ACM Transactions on Graphics, 2002, 21(3): 657-664

[13] Chen Zhonggui, Ouyang Yongsheng, Cao Juan. Feature-preserving method for mosaic image generation[J]. Journal of Computer-Aided Design & Computer Graphics, 2014, 26(4): 502-527(in Chinese)
(陈中贵，欧阳永昇，曹　娟. 特征保持的马赛克图像生成方法[J]. 计算机辅助设计与图形学学报, 2014, 26(4): 520-527)

[14] Zhang L, Yu J H. Image mosaics with irregular tiling[C] //Proceedings of the 12th International Conference on Computer-Aided Design and Computer Graphics. Los Alamitos: IEEE Computer Society Press, 2011: 155-162

[15] Lai Y K, Hu S M, Martin R R. Surface mosaics[J]. The Visual Computer, 2006, 22(9-11): 604-611

[16] Lyu Ruimin, Xu Mandi, Liu Yuan, *et al*. Artistic mosaic rendering using fuzzy color modeling[J]. Journal of Computer-Aided Design & Computer Graphics, 2016, 28(10): 1688-1698(in Chinese)
(律睿憼，徐曼頔，刘　渊，等. 融合模糊色彩思维建模的马赛克风格渲染技术[J]. 计算机辅助设计与图形学学报, 2016, 28(10): 1688-1698)

[17] Battiato S, Di Blasi G, Farinella G M, *et al*. A novel technique for Opus Vermiculatum mosaic rendering[C] //Proceedings of the 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision. Plzen: UNION Agency, 2006: 133-140

[18] Cacciola F. 2D straight skeleton and polygon offsetting[OL]. [2017-07-07]. http://doc.cgal.org/4.10/Manual/packages.html#PkgStraightSkeleton2Summary

[19] Chen Z G, Yuan Z, Choi Y K, *et al*. Variational blue noise sampling[J]. IEEE Transactions on Visualization and Computer Graphics, 2012, 18(10): 1784-1796

[20] Aurenhammer F. Power diagrams: properties, algorithms and applications[J]. SIAM Journal on Computing, 1987, 16(1): 78-96

[21] Byrd R H, Nocedal J, Waltz R A. Knitro: an integrated package for nonlinear optimization[M] //Pillo D G, Roma M. Nonconvex Optimization and Its Applications, vol 8. 1st ed. Heidelberg: Springer, 2006: 35-59

[22] Rother C, Bordeaux L, Hamadi Y, *et al*. AutoCollage[J]. ACM Transactions on Graphics, 2006, 25(3): 847-852

[23] Yu Z Q, Lu L, Guo Y W, *et al*. Content-aware photo collage using circle packing[J]. IEEE Transactions on Visualization and Computer Graphics, 2014, 20(2): 182-195

[24] Jing G M. Image interpolation and image-based content summary[D]. Hong Kong: The University of Hong Kong, 2015