基于GraphCut的纹理结构生成
计76 沈诣博

1. 代码结构和思路：

本人的代码使用python实现，并且集中在了同一个文件`mincut.py`内

通过修改其中的如下三行：

```
a = np.array(imageio.imread("$1"),dtype=np.int)[:,:,0:3]
imageio.imwrite("$2",rst)
```

来修改读写图片内容，且通过修改Mincut类中的参数（大小，贴片距离，refine轮数）来定义最终纹理生成形式。

主要功能集中在Mincut类里，该类由不同的matching函数，最小割(mincut)函数以及各种辅助函数组成。

首先，在生成一种纹理的时候，先使用random matching的方式并控制随机的方式，先把整个画布铺满；在此之后，再使用entire matching方式，求出每一个贴片位置的SSD值，从而进一步结合收敛因子k算出被选中的概率。这些计算我使用了numpy包完成。

在每一轮贴图时，首先通过各种辅助函数来计算重合范围，从而辅助下一步建图。同时，计算出重合范围中的边界点和内部点，对于前一种点来进行和源汇之间的连接。然后，在mincut函数内实现建图和求图。该部分使用了networkx包辅助建图和求最小割。本人也使用了dinic算法手动实现了另一个最小割，但是因为效率较低最终并未采用，该算法和包装在`graph.py`内。

在建图的时候，因为重合范围本身的规则性，重合区域内部点之间的关系可以沿着左上——右下的顺序去扫描，每个点和其右下方合法的点进行权值计算和连边。对于边界点，因为辅助函数已经给出，只需要判断其适合源还是汇连接即可。

对于进阶考虑的Old cuts的情况，可以按照如下的方式来实现寻找和建图：首先，可以通过一个公用的seam来记忆每一轮迭代的接缝区域，在一对邻居来自于不同的patch时，记下seam不为0，即为老接缝处。之后，在按照文中的方式建图，用新节点连接老接缝和新的块。

2. 采分点：

1. 基本算法：第三章中的最小割对应了Mincut类中的mincut函数，实现建图和最小割的求解第四章的entire patch matching算法对应了Mincut类函数中的entire_matching类，实现贴片功能。
2. Old cuts：在函数Mincut::mincut中，建图考虑了Seam nodes。

3. 实验结果和分析

下面分别是提供的4张图片的结果：

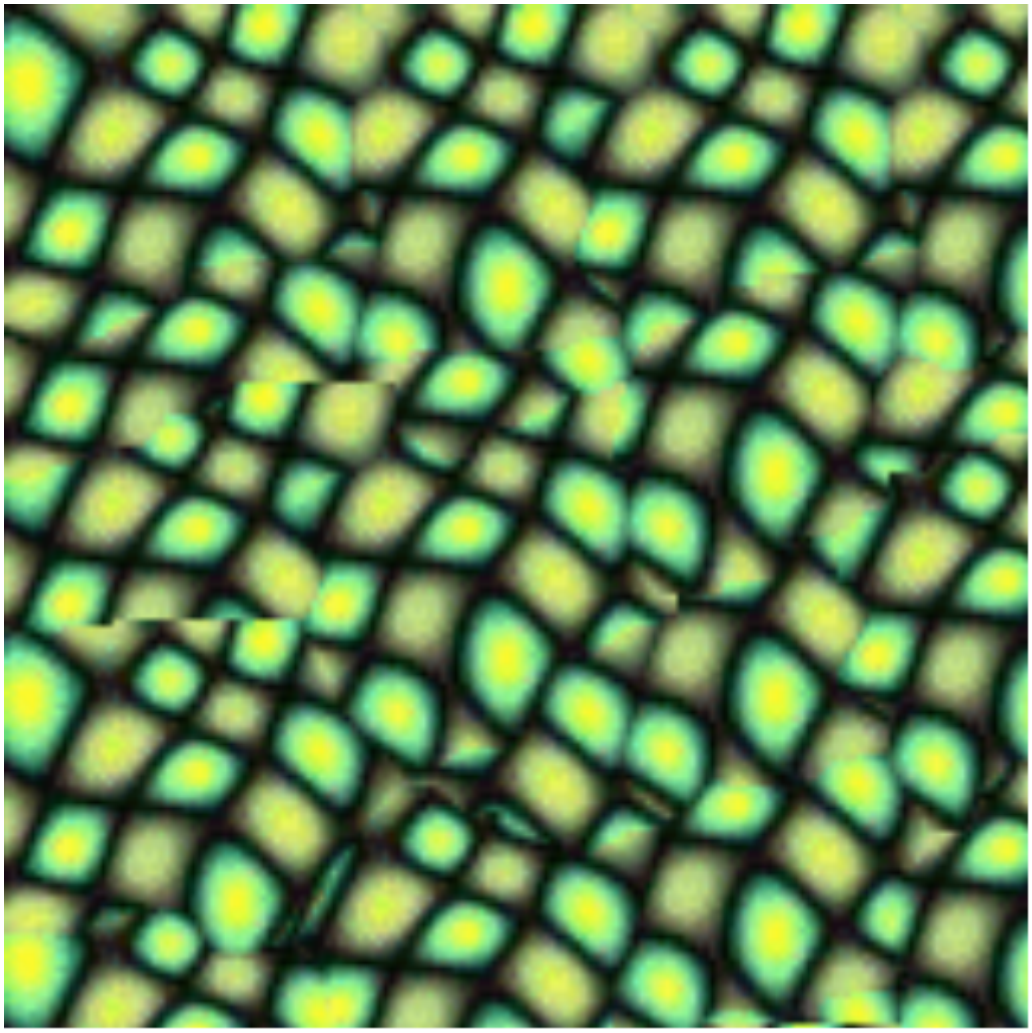

图1 绿色点

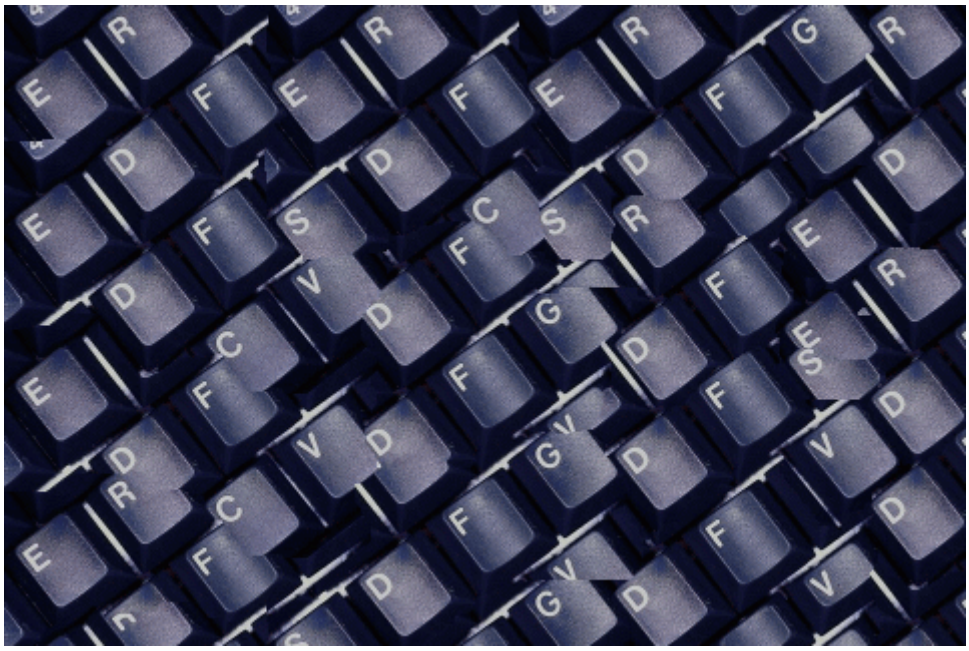

图2 键盘

图3 草莓

图4 鹰嘴豆

因为在生成贴片时只进行了一轮，所以这些效果还并不是最好，但是可以看出，鹰嘴豆和草莓的效果好于绿色点，而绿色点又好于键盘；其中鹰嘴豆效果最好，在不考虑明暗的时候看起来已经很逼真。分析这个效果差异，可能是来源于原有纹理的不规则性，从而一方面在主观容错能力上较大，一方面因为曲线较多的原因较好连接；而绿点和键盘为代表的纹理因为线型平直，规则性强，很难完全消除接缝的影响。

4. 致谢

感谢徐老师的精彩讲解，助教的详细说明和认真批改。

感谢余天枢同学和赵成钢同学在我完成作业时对我的提醒和帮助。