实验报告——Seam Carving图像缩放

马喆轩 2021080078

（清华大学 软件工程)

**摘要**：本次实验是实现一个将图像进行seam carving压缩的程序，将的图像压缩为的图像。本次实验环境为Visual Studio 2019与使用opencv库。

**1 Seam Carving算法**

**1.1 概述**

Seam carving是一种图像处理技术，旨在将图像压缩或裁剪为所需尺寸，同时最大程度地保留图像的重要信息。该技术基于图像中能量值较低的像素被认为是可删除的假设，通过在图像中找到能量最低的路径（称为“缝”），并将其删除或重复以改变图像尺寸。

相比传统的图像压缩和裁剪方法，seam carving在保留图像细节方面具有更高的效果，因为它考虑了每个像素的重要性。但是，它也有一些局限性，例如在图像中存在大量水平或垂直直线时，会导致压缩或裁剪效果不佳。

**1.2 能量函数**

假设一幅大小为的图像，我们可以利用能量函数来确定哪个区域是图像中比较重要的区域，即在裁剪后需要保留的部分。①将图像转化为灰色，再利用opencv库中的算子与相加运算就能求得初始的能量E。

**1.3 竖直与水平seam**

竖直seam指的是在图像中从上到下沿着能量值最小的路径（像素列）的集合，它们被删除后可以将图像的宽度减小。竖直seam的定义公式如下：

水平seam则是指在图像中从左到右沿着能量值最小的路径（像素行）的集合，它们被删除后可以将图像的高度减小。水平seam的定义公式如下：

简单来说，seam即是列(行)编号最多相隔一个像素的最小路径的集合。

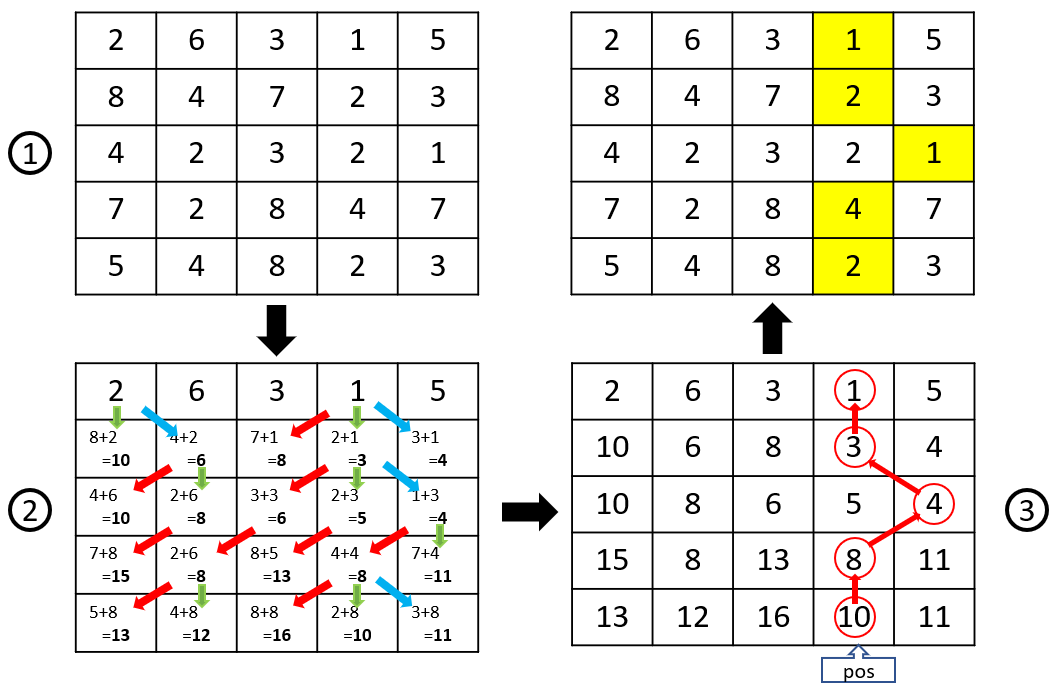
**1.4 竖直与水平seam动态规划求解法**

通过能量函数得出初始能量“矩阵”E后，就可以通过动态规划来求得竖直/水平seam。先讨论竖直seam，根据seam的定义，seam上两个连续的像素最多相隔一个像素，即只能从三个方向来选择，即左上方，正上方与右上方，而左右两端的边缘只能从正上方与左或右上方其中一方来选择。由于我们希望得出最短路径，因此我们每一步都要尽可能要从量较小的一方作为出发点，因此可以得出状态转移方程，

② M矩阵的第零行(下标由0开始) 就是初始的能量，通过上述状态转移方程即可求出整个M矩阵。

③在M矩阵的最后一行中找出最小值所在的列，并用回溯的方式找到它对应的能量最小的竖直的seam（缝）或路径。

水平seam求解法与竖直的是大同小异的，只是在进行了①后对E进行转置(transpose)与反转(flip)即可求出“竖直形式的水平seam”。

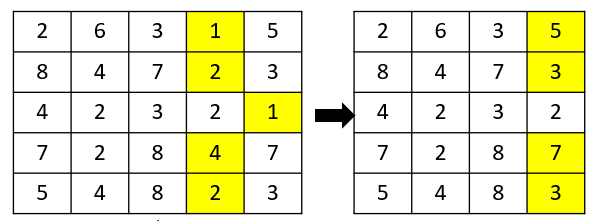


**图1** seam的动划求解法

**1.4.1 回溯路径求解技巧**

定义结构体Struct Point{float val, int path};若是从左上方延申的路径，则记录path=-1，正上方path=0，右上方path=1。当我们在M矩阵的最后一行中找出最小值所在的列(记为pos)之后，pos+M[i][pos].path即是第i-1行(上一步)的列的信息，通过不断的更新pos与i，即可得出一条完整的seam。

**1.5 删除seam**

④在求出seam后，需要删除seam才可以实现图片在真正意义上的缩小。如图2所示，在每一行的seam所在之列前面的元素，都保持不变，而在seam与seam之后的元素都用下一列的元素来进行覆盖，最后一列删除，因此图片的宽度就减小1了，同理也能够对高度进行缩小。

**图2** 删除seam演示图

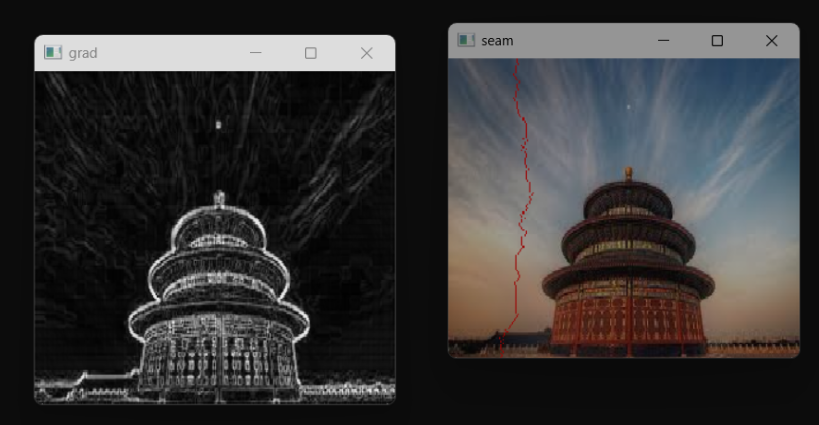
**1.6 算法细节**

我们要删除m/2行与n/2列的像素，设count\_row与count\_col是已删除的行与列的计数。当count\_row<m/2并且count\_w<n/2时，就会分别求出竖直与水平方向的seam，并且比较两条seam的能量，且取较小能量代价的seam作为待删除的行(列)。直到宽度或高度满足指定长度后，才进行一个方向上的直接求出seam与删除操作。

不断地重复①②③④，直到的图片压缩至，算法结束。

**2 运行实例**

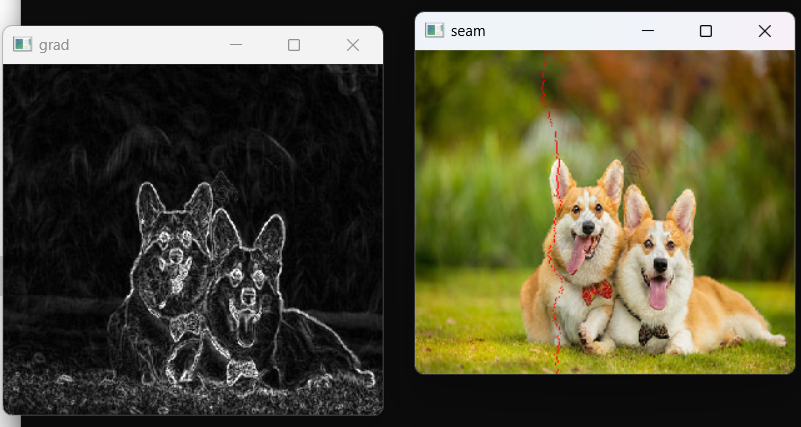
**2.1 例1**

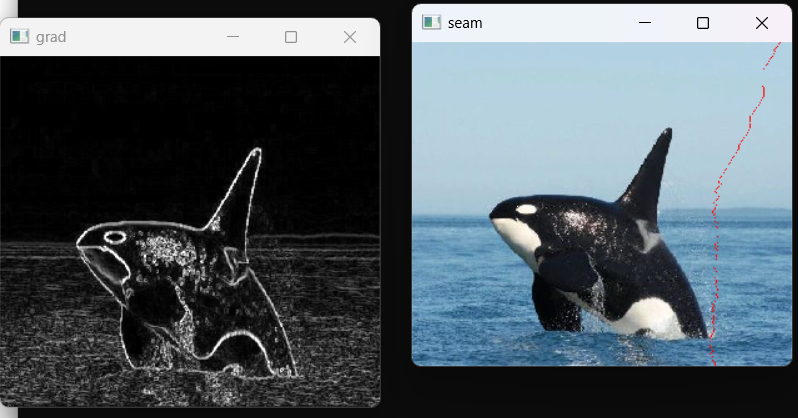
**图3** 原图

**图4** 灰度图（左），垂直seam（右）



**图5** 输出图

**2.2 例2**

**图6** 原图

**图7** 灰度图（左），垂直seam（右）

**图8** 输出图

**2.3 例3（变形）**

**图9** 原图

**图10** 灰度图（左），垂直seam（右）

****

**图10** 输出图

从例3的输出图中可以看出小狗的头部已经严重变形，造成本次压缩的结果不理想。

**3 总结**

在seam carving中，算法会寻找最小能量路径，即跨越图像的一条像素路径，其中每个像素都有一个能量值，代表其重要性。然后，该路径上的像素将被删除或复制，从而缩小或扩大图像。该算法可以在保留重要内容的同时调整图像的宽度和高度。

总体而言，seam carving是一种非常有用的图像处理技术，尤其适用于需要对图像进行精确调整的情况，例如将一个图片缩小以适应不同尺寸的屏幕。但是，它也存在一些局限性，例如在处理一些复杂的图像时，可能会出现一些不自然的变形，因此使用时需要谨慎。

**参考文献 (References)**

1. 翟胜楠,孟凡佳,刘金华.一种改进的Seam Carving图像重定向算法[J].湖北民族学院学报(自然科学版),2018,36(02):194-198.DOI:10.13501/j.cnki.42-1569/n.2018.06.018.
2. 杨剑炉.结合边缘检测的Seam Carving图像缩放算法[J].宜春学院学报,2019,41(03):35-38.
3. 王海昱,范仲然.一种基于Seam Carving的图像缩放方法[J].电脑编程技巧与维护,2017,No.385(19):5-8.DOI:10.16184/j.cnki.comprg.2017.19.001.
4. SyGoing. Seam carving 图像缩放(2019-07-2)[2023-04-8]. <https://blog.csdn.net/ouyangfushu/article/details/94470231> https://github.com/SyGoing/seam-carving
5. 【C++ OpenCV】Sobel算子边缘检测(2019-09-20)[2023-04-8]. <https://blog.csdn.net/Feeryman_Lee/article/details/101074986>

1. **时间复杂度**：基数排序的时间复杂度为，其中是数字位数，是输入数组的元素个数，是数字的基数。因为基数排序要对每一位进行排序，所以时间复杂度与数字位数有关。当较小且不是很大时，基数排序的时间复杂度可以达到线性级别。

2. **空间复杂度**：基数排序的空间复杂度取决于桶的数量和每个桶中的元素个数。如果每个桶中的元素个数是常数级别的，那么空间复杂度为，其中是数字的基数。如果每个桶中的元素个数不是常数级别的，那么空间复杂度可能会更高。

3. **稳定性**：基数排序是一种稳定的排序算法，它不会改变相等元素的顺序。这是因为基数排序是根据每个数字的位数依次排序，每次排序都只考虑一位，所以不会改变相等元素的顺序。

4. **最佳情况**：在最佳情况下，输入数组中的每个数字的位数相同且桶的数量等于基数，此时基数排序的时间复杂度可以达到。

5. **最坏情况**：在最坏情况下，输入数组中的每个数字的位数都不同，且基数很大，此时基数排序的时间复杂度可能会达到，因为需要对每一位进行排序，而每一位上的数字又需要用比较排序进行排序。

6. **平均情况**：在平均情况下，基数排序的时间复杂度取决于数字的分布情况和位数。如果数字的分布均匀且位数不太大，那么基数排序的时间复杂度可以达到线性级别。如果数字的分布不均匀或位数很大，那么时间复杂度可能会更高。

2.6 排序对比表

3 实例测试

，而希尔排序的曲线大于归并排序，基数排序与快速排序的曲线，而快速排序是最小的函数曲线。

除了基数排序，其他四种排序的结果都是与理论预期的差不多，虽然在理论上希尔排序的时间复杂度在到之间，但从实际数据上来看，它运行时间比插入排序快不少个数量级，且没有到的级别，10次运行结果都是平均在52000ms左右，在运行时间上是“稳定”的，这里仅讨论等概率下随机生成的数据，不考虑极端情况。

）