

计算机图形学Seam Carving图像缩放实验报告

PB19051035周佳豪

问题描述

使用Seam Carving实现图像的缩放

input:输入一个图像和该图像的能量图, 以及缩放后的长宽

output:输出缩放后的图像, 要保持图像的主体部分不扭曲

算法说明

图像的缩小

由于图像的横向缩小与纵向缩小思路一样, 故在此只说明横向缩小图像的思路。

1. 根据目标图像与源图像宽度的差距, 可得到要删减的总列数, 记为 m

2. 遍历 `saliency.png` 图, 得到总能量值最少的一列, 设为 `column`。

`column` 为一个数组, 依次记录着每一行要删除的一个像素所在的列数。

遍历方法即为暴力搜索。设第 i 行选择了第 j 个元素, 则第 $i+1$ 行选择的元素是 $\min(saliency_{i+1,j-1}, saliency_{i+1,j}, saliency_{i+1,j+1})$ 。根据此规则即可得到从某个起点开始的最小能量的列。再遍历所有起点 (第一行的所有元素), 即可得到总能量值最小的一列, 操作过程中需要创建一个中间数组来储存能量最小的一组元素的位置。得到能量值最小的一列的时间复杂度为 $O(wh)$, 空间复杂度为 $O(h)$, 其中 w 为图像的宽度, h 为图像的高度。

3. 若图像的宽度缩小 m 列, 每次缩小一列都要找到能量值最小的一组元素, 然后依次删除能量图和原图像中的相应像素。删除能量图中的相应像素是为了在接下来的删除过程中, 新得到的最小能量的元素能和已经删除过像素的图像一一对应。

其中删除图像的一组元素的具体操作为: 将原图像的元素拷贝到新图像里, 在拷贝的过程中跳过要删除的像素, 操作过程中需要创建一个中间数组来储存新的图像, 最后再把该图像赋给 `pixels` 即可。可得时间复杂度为 $O(wh)$, 空间复杂度为 $O(wh)$ 。

故最终可得缩小图像的时间复杂度为 $O(mwh)$, 空间复杂度为 $O(wh)$, 比较合理。

图像的扩充

由于图像的横向扩大与纵向扩大思路一样, 故在此只说明横向扩大图像的思路。

1. 寻找能量最小的一列元素思路和缩小图像思路一样, 这里不再赘述。

2. 对于已经找到的能量最小的元素, 首先需要对能量图进行操作, 将相应位置的元素能量值增加到最高, 以防止图像在扩充的过程中多次选择复制同一列的像素。

同时, 为了使能量图和已经扩充后的图像像素一一对应, 也需要对能量图进行扩充。扩充的思路是在已经得到的能量最小的元素与其右邻居之间插入一个能量值最高的像素(这操作与图像缩小有点区别)。

3. 对于扩充图像的一组元素, 其具体操作为: 将原图像的元素拷贝到新图像里, 在拷贝的过程中若遇到需要扩充的像素, 对该像素拷贝两次即可。操作过程中需要创建一个中间数组来储存新的图像, 最后再把该图像赋给 `pixels` 即可。

4. 扩充图像的时间与空间复杂度与缩小图像一样, 时间复杂度为 $O(mwh)$, 空间复杂度为 $O(wh)$

实验结果

缩小图片



其中左上角图片是原图, 右上角为缩小了宽度, 左下角为缩小了高度, 右下角既缩小了宽度也缩小了高度。

扩充图片





其中左上角为原图，右上角为扩大了宽度，左下角为扩大了高度，右下角既扩大了宽度也扩大了高度。

结果分析

- 可以看出，图像效果还不错，最大限度保护了图像的主体部分。
- 实验的执行时间合理，执行一次扩充或缩放，大概需要1s时间。日后可能可以优化找寻最小列的算法来进一步缩短算法执行时间。