

# SeamCarving 图像压缩实验报告

姓名：覃果 学号：2020012379 班级：软件02

2022 年 4 月 14 日

## 摘要

实现了 SeamCarving 算法，并测试了该算法在不同大小图片上的运行时间和运行效果。

## 1 实验环境

CPU: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz

OS: ubuntu

Python interpreter: Python 3.8.0

## 2 算法分析

### 2.0.1 算法过程

首先计算出每个点的破坏度  $d[i, j]$ ，此处破坏度定义为两个轴方向的梯度的绝对值的和。

记  $C[i, j]$  为从第一行到点  $(i, j)$  的最低破坏度。

则  $C[i, j] = d[i, j] + \min(C[i - 1, j - 1], C[i - 1, j], C[i - 1, j + 1])$ ，同时用  $path[i, j]$  记录  $C[i, j]$  的来源

计算出所有的  $C[i, j]$  后找出  $C[n, j]$  中的最小者，并根据  $path[i, j]$  找出对应的狭缝并删除狭缝对应的像素

如果要删除的是行，则先将图片转置然后调用上述步骤，最后将删除了狭缝的图片再转置一次即可

多次执行以上步骤直到图片达到要求的尺寸

### 2.0.2 算法分析

时间复杂度:  $O(mn(n - n' + m - m'))$  m, n为原始图片的宽和高, m', n'为最终图片的宽和高

空间复杂度:  $O(mn)$  m, n的含义如上, 用于储存图片数据以及破坏度等中间数据

### 2.0.3 算法评价

这种缩放的方法是根据内容的缩放, 每次删除相对不重要的像素, 能够在缩小图片尺寸的同时较好地保持图片中比较重要的信息。

不过此方法的时间复杂度较高, 在处理较大的图片时所需要的时间比较长。

## 3 实验设计思路

使用该算法, 将大小分别为 1080\*1920, 1080\*960, 540\*1920, 540\*960, 540\*480, 270\*960。270\*480 的图片的宽和高都缩小至原来的一半。测试其花费的时间以及得到最后的图像结果。

## 4 结果分析

实验结果见下表:

h*w	1080*1920	1080*960	540*1920	540*960	540*480	270*960	270*480
time/s	17408.535	5595.577	6481.986	2085.744	741.9016	776.6455	246.2957

根据实验结果可见, 当宽或者高变为原来的一半时, 时间大概变为原来的一半, 这符合对应算法的复杂度

图片结果如下:



图 1: 1080\*1920压缩前



图 2: 1080\*1920压缩后



图 3: 1080\*960压缩前



图 4: 1080\*960压缩后



图 5: 540\*1920压缩前



图 6: 540\*1920压缩后



图 7: 540\*960压缩前



图 8: 540\*960压缩后



图 9: 540\*480压缩前



图 10: 540\*480压缩后



图 11: 270\*960压缩前



图 12: 270\*960压缩后



图 13: 270\*480压缩前



图 14: 270\*480压缩后