
Seam Carving Tutorial
李思哲
2022秋季《计算概论(B)》

Seam Carving算法简介

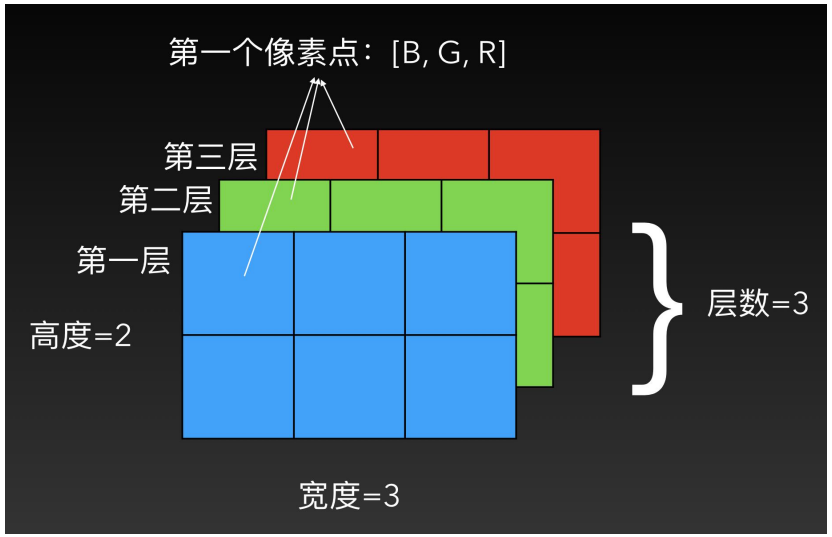
- Seam Carving算法通过计算图像中的“关键区域”，使得改变图像长宽比时，不会丢失图像信息，且不会失真

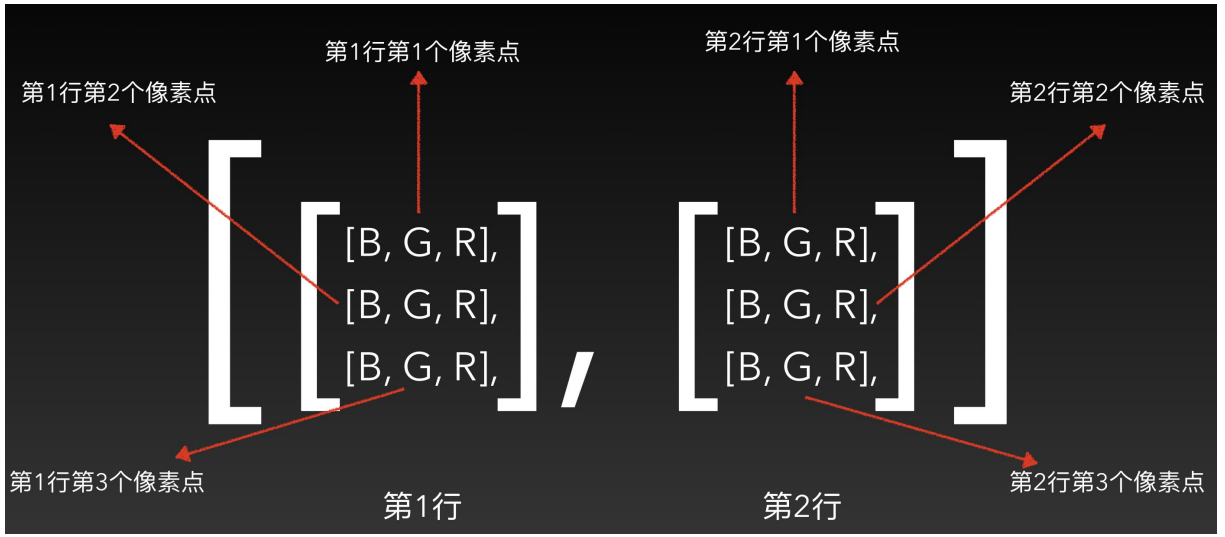
- 算法步骤
  ①. 读入图像，计算图像的能量图
  ②. 利用动态规划算法寻找能量图中能量最低的路线（“接缝”）
  ③. 在原图中删除该接缝
  ④. 重复②③，直至输出图片满足长宽比要求

- 下面我们会依次讲解每一步

读入图片（代码给出）

- 图像由多个排列整齐的像素点构成，而像素点的颜色可以通过RGB三个通道表示。因此，对于一张w*h的图像，可以使用w*h*3大小的数组进行存储。
- python的图像处理工具有很多，例如Open CV，PIL，matplotlib等。值得注意的是，在Open CV中图像是以BGR颜色模型来存放的。
- 在这里，我们给出了读入图片的代码。我们利用matplotlib.image库将图片保存为numpy.ndarray格式，以便后续处理。





计算能量图（需完成）-1

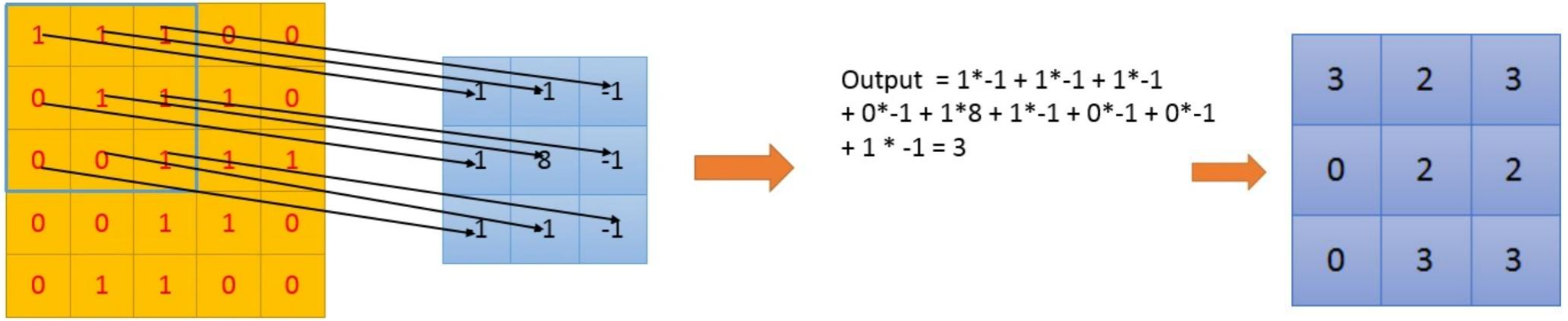
- 为了表示每一个像素点的“重要程度”，我们引入能量图的概念。一种常用的能量图使用梯度幅度来表示，这种表示的合理性在于，梯度较大的点往往更可能是图中物体的边缘。此外，也可以利用熵图或显著图来表示。
- 我们以最基本的梯度图为例

$$e_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|$$

- 这个公式告诉我们，图中的每个像素，其能量值为x,y两个方向的偏导数绝对值之和。
- 而像素点的分布可以看作离散的，因此，在这里求偏导数只需要与附近的点作差。

计算能量图（需完成）-2

- 在实现偏导数的计算前，我们先引入卷积的概念。



图像　　卷积核

- 直白地说，卷积操作即对于每一个像素点，将特定的卷积核“盖”在该点上，对应点的值相乘后作和，即为该点经过卷积操作后的值。

计算能量图（需完成）-3

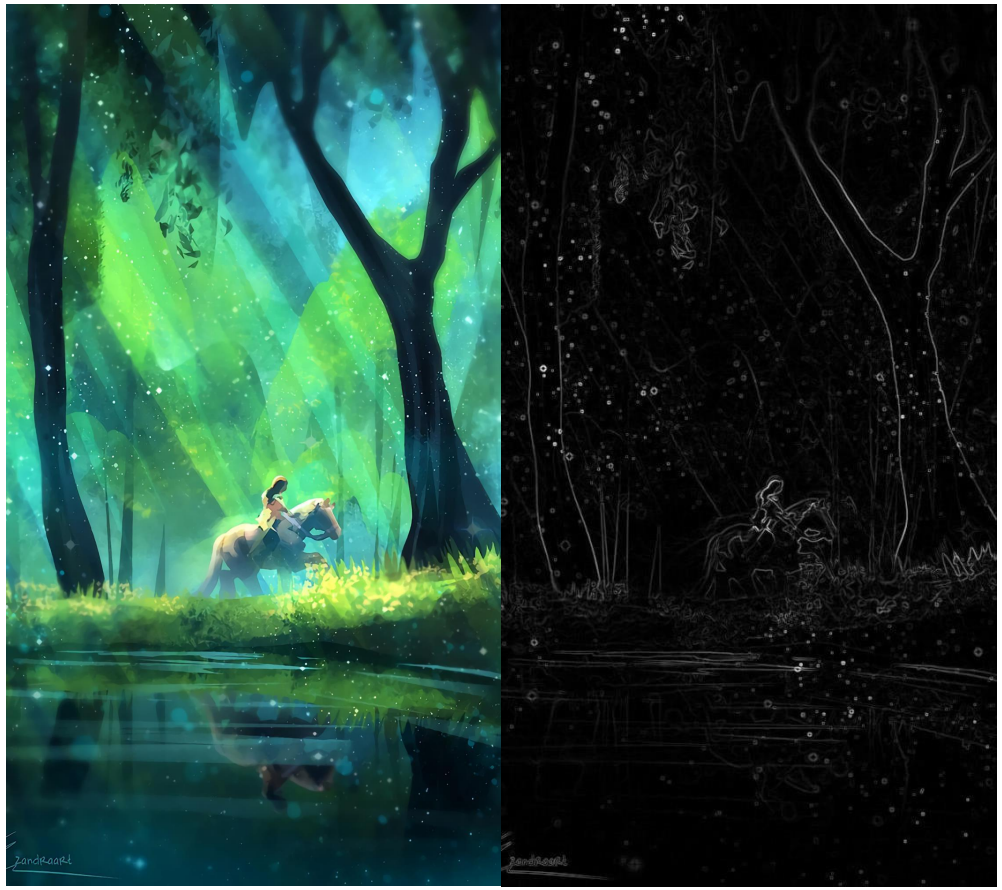
- 一些特定的卷积核在计算偏导数时有着重要的作用，例如在这里，我们考虑使用sobel滤波器的卷积核作用在图像G上，即

$$p'_u = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{G} \quad \text{and} \quad p'_v = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{G}$$

- 其中p_u和p_v分别代表横向和纵向的偏导数。
- 从直觉上可以认为，第一个滤波器会用其顶部值的差将每个像素替换为其在底部的值。第二个滤波器会用其左边值和右边值的差替换每个像素。这样就能捕捉 3X3 区域像素的整体趋势。实际上，这种方法和边缘检测算法高度相关。

计算能量图（需完成）-4

- 利用上述思想可计算能量图
- 可能用到的操作：
  - 将list变量转化为numpy.array类型: a = np.array(b)
  - 堆叠多个numpy.array数组: np.stack((a, b), axis=0)
  - 计算卷积 (scipy库中的convolve函数): convolve(img, kernel)
  - 将numpy.array数组中每个元素取绝对值: np.absolute(a)
  - ……



寻找接缝（需完成）

- 在能量图中寻找能量最低的接缝，可看作经典动态规划问题

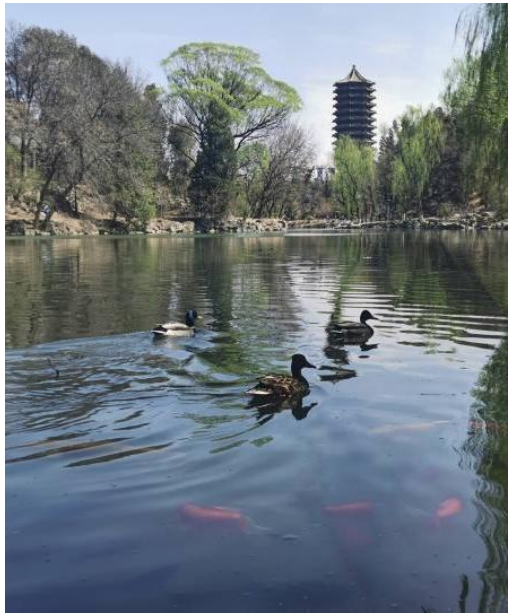
- 由于我们要求接缝中各像素点连续，因此通过 (i, j) 像素点的接缝一定通过 (i-1, j-1) 或 (i-1, j) 或 (i-1, j+1) 像素点，满足最优子结构。
- 设 dp[i][j] 表示从 i=0 处出发，到达 (i, j) 像素点的能量最低接缝的能量，则动态规划的初始条件为 dp[0][j] = energy[0][j]，状态转移方程为 dp[i][j] = energy[i][j] + min(dp[i-1][j-1], dp[i-1][j], dp[i-1][j+1])，目标为寻找最小的 dp[h-1][j]

删除接缝（需完成）

- 找到能量最低的接缝后，利用numpy功能删除接缝，从而改变图片比例。
- 在这里，你需要设计合适的数据结构，存储找到的接缝的具体信息。

基本任务（必做）- Seam Carving算法

- 完成上述 Seam Carving 算法 (补全sc.py)，即实现 energy_function 函数、 dp 函数、delete_one_road 函数
- 代码中给出的函数及接口可直接使用，也可自行定义数据结构及接口



拓展任务（选做）- 横向或纵向缩小

- 我们目前默认为删除单个方向的接缝（从上至下或从左至右），config.yml 中 direction 参数可以控制接缝的方向，进而达到更灵活地改变图片比例的目的

- 拓展任务1：实现可指定方向的比例改变

- 难度：★
- 提示：虽然改变接缝方向实现起来有些繁琐，但或许可以考虑旋转图片？

拓展任务（选做）- 图像放大

- 缩小图片可以通过删除接缝来实现，同理，放大图片则可以根据添加接缝来实现。
- 拓展任务2：通过添加接缝实现图像放大
- 难度：★

拓展任务（选做）- 图像放大plus

- 如果依旧采用前述方法实现图像放大，是否会发现存在问题？如何解决？
- 考虑能量最低的接缝被复制后依旧为能量最低，因此会导致图片有明显的拉伸感。那么我们可以记录n条能量最低的接缝，一并进行像素点的复制，从而解决问题。
- 拓展任务2+：通过记录多条接缝实现图像放大
- 难度：★★★

拓展任务（选做）- 双向压缩

- 我们前述的任务中只能通过一个方向的接缝的增减来改变图片整体比例，但有时我们只想要保留图片中最重要的区域，这时则需要我们在两个方向上均有接缝的增减操作。以两个方向均删除接缝为例，值得注意的是，是否应该某一方向的接缝删除完全后再删除另一方向接缝？还是两个方向交错操作呢？

- 拓展任务3：实现图像重要区域保留（即双方向删除接缝）

- 难度：★★

拓展任务（选做）– 实时处理

- 注意到，前述程序运行的时间与增删接缝的数量成正比，因此当删除接缝数量较大时，程序运行无法做到实时。考虑到读入图像后，我们可以先对图像做预处理（可能用时较长），依次记录能量最低、第二低...的接缝，那么当有多种操作需求时，这些操作都可以在较短时间内完成，从而实现实时操作。

- 拓展任务3：实现图像实时处理

- 难度：★★★★★

作业要求

- 完成**基本任务**即可得**满分**

- 拓展任务供同学们思考，鼓励大家进行探索和实现，但不作为硬性要求，也欢迎和助教进行讨论。完成拓展任务可获得额外加分，但总分不得溢出。

- 实验报告要求写清楚**实现过程**（用到了哪些包、遇到了哪些困难、如何解决、有哪些创新点等），并对效果不佳的实例**作出分析**。鼓励在报告中写出算法的其他应用以及自己的思考，鼓励报告中加入图片，报告清楚简明即可，不要求字数（不要卷字数！写清楚就好！）

- 请将代码和实验报告放入同一文件夹后，打包上传到教学网。

写在最后

- Seam Carving是传统计算机视觉中最优美的算法之一，其应用场景远不止我们前面提及的部分。论文原文中也提到了更多的应用，感兴趣的同学可阅读论文原文（https://perso.crans.org/frenoy/matlab2012/seamcarving.pdf），对于视觉领域感兴趣的同学也欢迎和助教交流。

- Seam Carving算法实现用到了动态规划，也许对于一些同学来说有一定困难。算法在网络上很容易找到代码实现，但是不鼓励大家照搬他人的代码，而是更希望大家能够经历钻研的过程。**如有借鉴他人的代码，请务必标明出处。**

- 祝大家大作业顺利、愉快完成，and enjoy it!