图像接缝裁剪 Seam Carving

吴欣怡　课程 ID 112　学号 PB21051111

2024 年 3 月 16 日

摘要

本题为对经典的图像缩放算法 Seam Carving 算法的复现。在缩小图像时，算法的主体思路是根据图片中的像素特征结合特定卷积核计算得到各个点位的能量值，采用动态规划和回溯法找到横向或纵向的一条能量值最低的路径并删除此路径上的所有像素点，其余点位进行平移补位，多次删除后操作得到指定尺寸的图片。

一、前言（问题的提出）

1.1 要研究什么问题

朴素的图像裁剪拼接方法常常会因为对图像的直接拉伸和剪裁导致图片中关键元素的失真或丧失。我们需要找到能最大限度保持图像中的关键元素及真实度的方法。Seam Carving 算法可以尽可能保持图像中“重要区域”的比例，避免由于直接缩放造成的“失真”。

1.2 Seam Carving 算法的优点

Seam Carving 算法通过对原图增加（删除）像素来实现图像缩放，可以尽可能保持图像中“重要区域”的比例，避免由于直接缩放造成的“失真”。

二、相关工作

Seam Carving 算法最初由 Shai Avidan 和 Ariel Shamir 在他们的 2007 年论文中提出他们介绍了 Seam Carving 作为一种图像大小调整技术，以在保持图像内容结构的同时改变图像的尺寸。自 Seam Carving 的最初提出以来，许多研究人员已经对该算法进行了改进和扩展。例如，有些工作集中于改进 Seam Carving 的能量计算方法，而其他工作则探索了用于对象保护的变体。除了图像大小调整之外，Seam Carving 还被应用于许多其他领域。例如，它已经被用于视频剪辑、图像增强和计算机辅助设计。

三、问题分析

3.3 性能评估

Seam Carving 算法的性能评估是本实验的关键。我们将评估以下几个方面：

- 图像质量：裁剪后的图像是否保留了重要内容，并且没有出现明显的失真。
- 裁剪速度：算法执行的时间与图像尺寸的关系。

3.4 算法优化

Seam Carving 算法的实现可以进行一些优化，以提高性能或减少计算成本。例如：

- 并行化：利用 Matlab 的并行计算功能加速算法执行。
- 优化能量计算：选择更高效的能量计算方法，减少计算复杂度。

四、建模的假设

我们在实现 Seam Carving 算法时做出以下假设：

3.5 图像内容连续性

我们假设图像中重要内容通常是连续分布的，因此沿着能量路径裁剪图像时，不会出现明显的断裂或失真。

3.6 能量函数有效性

我们假设所选用的能量函数能够准确地反映图像中像素的重要性，以便正确地识别最小能量路径。

五、符号说明

能量转移公式描述了如何计算到达每个像素点的最优 Seam 的能量总和。对于竖直 Seam，我们可以将问题视为图论中的最短路径问题。每个像素点都可以看作是一个节点，相邻像素点之间存在连接边。连接的规则是当前行的像素点只能连接到下一行的左、中、右三个像素点，因此我们需要找到从图像的第一行像素点出发，到达最后一行像素点的最短路径。

经典的最短路径算法可以解决这个问题，但是在这里我们可以使用动态规划来更高效地求解。我们定义一个二维数组 $M[i][j]$，表示到达像素点 (i, j) 的最优 Seam 的能量总和。(在求解竖直方向缝隙时) 只需要找到最后一行中能量最小的像素点，即 $min(M[H-1][j])$，然后通过回溯即可找到最优 Seam。

假设像素点 (i, j) 的能量为 $e[i][j]$，则能量转移公式为：

$$M[i][j] = e[i][j] + \min(M[i-1][j-1], M[i-1][j], M[i-1][j+1]) \tag{1}$$

其中，$M[i][j]$ 表示到达像素点 (i, j) 的最优 Seam 的能量总和，$e[i][j]$ 表示像素点 (i, j) 的能量，H 表示图像的总高度。水平方向同理。

表 1: **符号说明**

符号	说明
$M[i][j]$	到达像素点 (i, j) 的最优 Seam 的能量总和
$e[i][j]$	像素点 (i, j) 的能量
H	图像的高度

六、数学模型建立

1. 能量函数：
   - Seam Carving 算法的第一步是定义一个能量函数，用于衡量图像中每个像素的重要性。该函数量化了每个像素与其相邻像素的对比度。较高的能量值表示像素更为重要。
   - 在 Seam Carving 中常用的一个能量函数是图像的梯度幅度。它是根据图像强度的梯度计算得出的。梯度幅度被计算为每个像素位置的水平和垂直梯度的平方和的平方根。数学上，可以表示为：

$$E(x, y) = |\nabla I(x, y)| = \sqrt{(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2}$$

   - 在代码中，使用了老师提供的框架，能量值的计算使用 Sobel 滤波器来近似计算梯度幅度以得到每个像素的能量值。

2. 寻找缝隙：
   - 在图像中找到具有最小能量的缝隙。缝隙是图像中从顶部到底部（或从左到右）的连接像素路径，该路径使得路径的最后一个位置的像素的能量最小。
   - 寻找缝隙使用动态规划：计算一个成本矩阵，其中每个元素表示从图像顶部到达该像素的最小成本。到达位置 (x, y) 的最小成本是基于当前像素的能量和在前一行中到达其相邻像素的最小成本计算的。然后，从底部行向顶部行回溯，每一步选择累积成本最小的像素。
   - 在代码中，‘find_min_seam‘ 函数实现了动态规划来在垂直或水平方向上找到最小能量的缝隙。

3. 删除缝隙：
   - 一旦找到具有最小能量的缝隙，就从图像中删除它。删除缝隙涉及沿着缝隙路径从图像的每一行（或列）中删除一个像素。这会将图像的宽度（或高度）减少一个像素。
   - 在代码中，‘remove_VERTICAL‘ 和 ‘remove_HORIZONTAL‘ 函数通过相应地移动像素来从图像中删除缝隙。分别对应竖直方向裁剪和水平方向裁剪。

总的来说，Seam Carving 算法通过迭代地找到并删除图像中的缝隙，直到达到所需的宽度或高度。它在调整图像大小时保留了最重要的内容。

七、结果（与对比）

你的算法产生的结果，如果有多种算法产生的结果，需要做对比。

7.7 结果展示

图片显著缩小并且保留了主体内容。

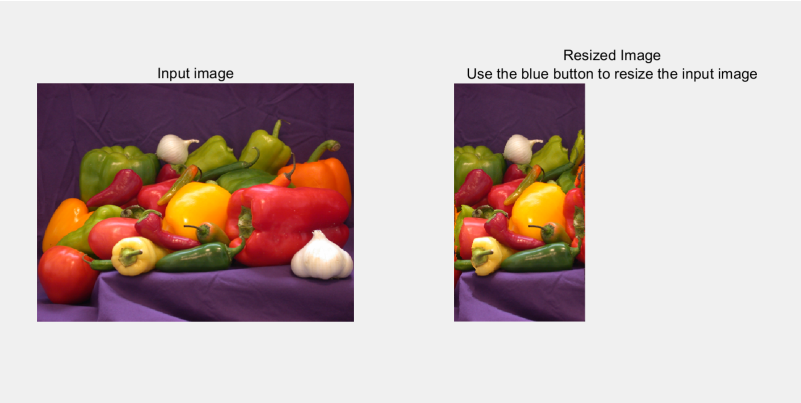

图 1: 原图和 seam carving 处理后的图片对比

图 2: 原图和 seam carving 处理后的图片对比

八、结论

从实验结果可以看出，使用缝刻算法进行图像缩放时，能够不改变图像的主体内容，虽然图像中内容主体间的相对位置会发生一些变化，但并不影响图片的观感。该算法和传统的对图像直接进行缩放不同，此算法是基于图像内容，对图片中不重要的内容（梯度）进行删减，从而达到图片缩放的目的。但是，另一方面，由于算法仅仅考虑梯度，在某些情况下，缩放的效果并不好，Seam carving 能够有效对图像进行缩放，但是具有一些缺点，如上，另外在缩放比例较大时，该算法处理时间大幅度增加，效率变低。对于示例中的 peppers.png 图片，只删除 300 个单位长度情况下，处理速度大概为 10s，还是相对迅速的。后面使用的 2276*1280 大小的图片在删除水平方向 800 个单位时大概需要 5min 及以上的时间。

九、问题

使用其它的卷积核可能效果更好

参考文献

[1] Shai Avidan, Ariel Shamir,Info Claims, Seam Carving for Content-Aware Image Resizing, ACM Transactions on Graphics,Volume 26,Issue 3, 2007