基于接缝裁剪的图像压缩 实验报告

吴佳龙 2018013418

摘要

本次实验结合理论分析和程序设计，实现了基于接缝裁剪的图像压缩算法，在求解最优接缝的过程中运用了动态规划算法；在不同的内容、不同的尺寸的图像上运行该算法，发现该算法能够在对图像的宽度进行裁剪时 content-aware 地保留重要的内容，保留较高的自然感，但是在对图像的高度进行裁剪、对非风景图进行裁剪时，结果较差。

1 问题

实现一个将图像进行 seam carving 压缩的程序，程序的基本功能是可以将 $m \times n$ 的图像压缩为 $m/2 \times n/2$，并有方便的输入输出功能。

2 实验环境

操作系统：macOS 10.15.3

Python 版本：3.7.6

环 境：numpy=1.18.1，opencv=4.2.0，tqdm=4.43.0

3 算法分析

3.1 基于接缝裁剪的图像压缩

定义图像 $A[1..m, 1..n]$ 的一条（竖直的）接缝为：从每行删除一个像素，使得图像变窄一个像素；为了避免影响视觉效果，要求相邻两行中删除的像素必须位于同一列或相邻列；删除的像素从顶端行到底端行构成一条“接缝”(seam)。

假定每个像素都已经计算了一个破坏度 $d[i, j]$，表示删除像素对图像视觉效果的破坏。每次压缩一个宽度，我们希望删除破坏度之和最低的接缝。

3.2 破坏度的定义

直觉上，一个像素的破坏度越低，它与相邻像素的相似度越高。我们使用 Sobel 算子来计算一个像素与相邻的像素之间的差异，具体地，对图像 I 进行卷积：

$$\mathrm{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathrm{I}$$

$$\mathrm{G_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathrm{I}$$

$$\mathrm{G} = |\mathrm{G_x}| + |\mathrm{G_y}|$$

我们可以选择 G 可以作为逐像素的破坏度。

3.3 求解接缝的动态规划算法

给定破坏度，我们使用动态规划求解最优接缝。

定义状态 $f[i, j]$ 表示：**从第一行逐行选择到第 i 行，且第 i 行选择了像素 $A[i, j]$ 时当前选择的所有像素的破坏度之和的最小值**，且 $f[i, 0] = f[i, n + 1] = \infty$ ，则 f 有如下最优子结构的性质

$$f[i, j] = \min\{f[i-1, j-1], f[i-1, j], f[i-1, j+1]\} + d[i, j]$$

分别表示上一行选取的是左上角、正上方、右上角的像素取最小值。最终的结果，也就是破坏度最小的接缝的破坏度是

$$\min_{1 \le j \le n} f[m, j]$$

该算法的时间复杂度是 $\Theta(nm)$。

算法的伪代码如下：

```
1  FIND−SEAM(A,m,n)
2      initialize f[1..m, 0..n+1] with f[1,1..n]=
           d[1,1..n], f[1..m,0]=f[1..m,n+1]=
           inf
3      initialize s[1..m, 1..n]
4      for i = 2 to m
5          for j = 1 to n
6              f[i,j] = min( f[i−1,j−1], f[i−1,j
                   ], f[i−1,j+1] )+d[i,j]
7              s[i,j] = argmin( f[i−1,j−1], f[i
                   −1,j], f[i−1,j+1] )
8
9      initialize result[1..m] to restore the
           column selected on every line
10     result[m] = argmin(f[m,1..n])
11     for i = m−1 to 1
12         result[i] = s[i+1,result[i+1]]
13     return result
```

4 实验

4.1 实验细节

本次实验使用 Python 语言实现上述算法，且使用了 OpenCV 处理图像。

对于 RGB 三通道的图像，我们先对每一通道分别求出 G，然后三个通道的 G 相加得到最终的破坏度。

图像压缩的具体步骤为：首先我们不断删除竖直方向的最优接缝，直到宽度满足裁剪的要求；然后将图像转置，依然不断删除竖直方向的最优接缝，直到高度满足要求；最后将图像再次转置成正常的样子。总体的时间复杂度是 $O(\max(n,m)\cdot mn)$。另外还尝试过，交替删除行列直到达到要求的尺寸，最终结果与先删列再删行视觉上没有显著的差异。

4.2 实验结果

对不同的尺寸为 320×240 或 320×214 的图片应用该算法的结果见图 1 。

从结果可以看出，该算法在对图片进行 resize 时，可以 content-aware 地裁剪掉不重要的内容，如重复的天空、重复的水域，尤其是图 1 第 1 行第 2 列展现的结果：去除了大片的天空和草地，保留了人和塔的尺寸基本不变。在第 4 行和第 5 行中，也都将重要的内容如山、岩石等通过缩小他们的距离来保留。

但是在第 3 列结果中，除了第 2 行和第 3 行这种无主次内容的风景图，其余图片均出现了图中物体尺度、比例不自然的情况，且物体间大量空隙被裁剪，过于集中拥挤，构图不自然。经过多次实验，总结出，该算法在对于图片高度进行缩放时表现不佳。

另外，从第 6 行和第 7 行可以看出，该算法不能很好地理解图像中的语义信息，将背影、猫等部分裁剪，得出了十分奇怪的结果。

本次实验还对尺寸更大的图片（如 800×533）运行过该算法，得到的结论与以上一致。

4.3 运行时间

在本次实验环境下，将一张 240×320 的图片压缩为 120×160，大约需要运行 2 分钟。若采用 C++ 等其他语言实现，将得到更短的计算时间。

5 总结

本次实验实现了基于接缝裁剪的图像压缩算法，该算法在对图像的宽度进行裁剪时结果较自然，artifacts 不显著，但是该算法在对图像的高度进行裁剪或者对非风景图进行裁剪时，得到的结果较差。

参考文献 [1] 中还提及了更多的破坏度的定义方法，以及该算法的更多应用，如增加宽高、去除指定物体等，有待进一步探索。

References

[1] Avidan, S., Shamir, A.: Seam carving for content-aware image resizing. ACM Transactions on Graphics (TOG) 26, 10 (2007)

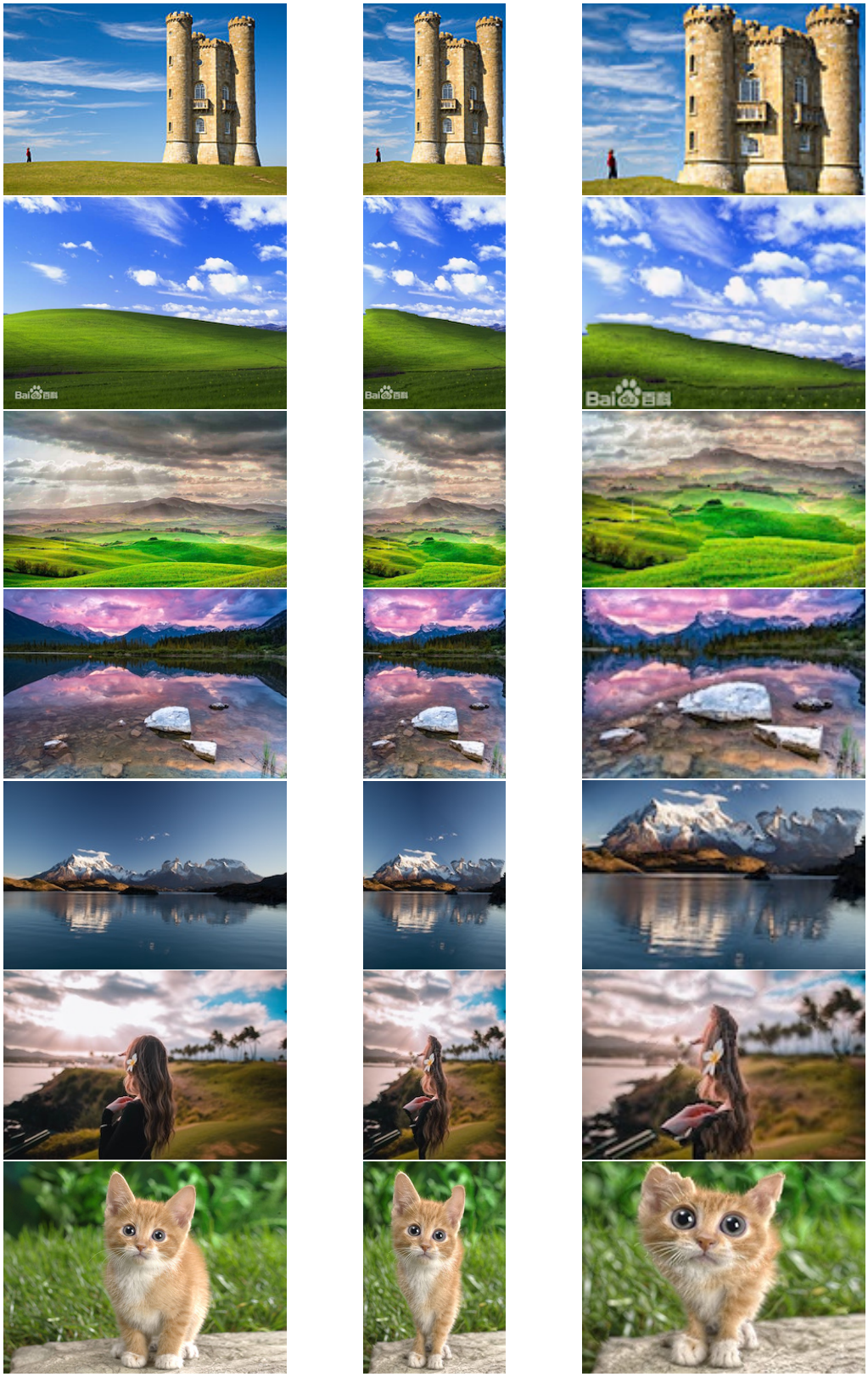

Figure 1: 基于 seam carving 的图片压缩算法的部分结果。其中第一列表示原图，第二列表示经过裁剪掉一半的列数的竖直接缝后的结果，第三列表示再裁剪掉一半的行数的水平接缝的结果。