Content-Aware Image Resizing[*]

石大川 <sdc17@mails.tsinghua.edu.cn>

2020 年 4 月 5 日

1 Aspect Ration Change

1.1 Procedure

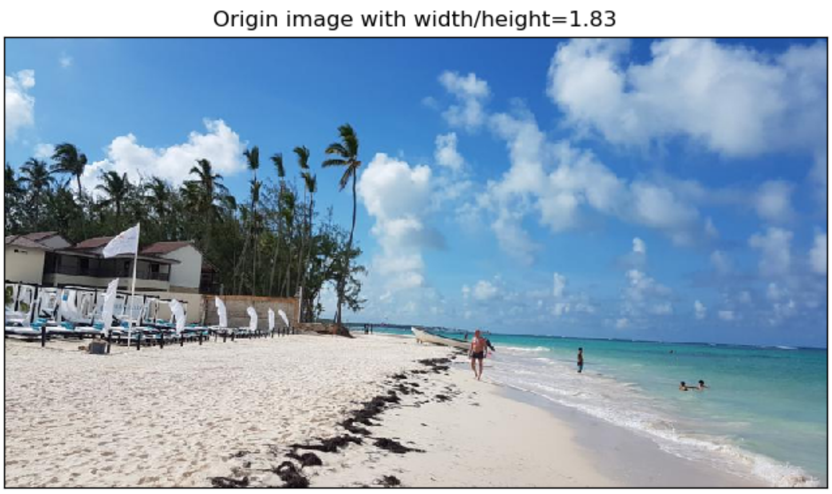
Algorithm 1 Aspect Ration Change

```
1:  Compute number of seam to remove:
        $$height \times (old\_ratio - new\_ration)$$
2:  for all Seam to remove do
3:      Compute Energy, $I$ denotes input pixels space, $\Omega$ denotes its boundary,
4:      for all $p \in I$ do
5:          if $p \in \Omega$ then
6:              Compute energy given by three existing surrounding pixels
7:          else if $p \in I \backslash \Omega$ then
8:              Compute energy given by:
                    $$E(p) = |\frac{\partial p}{\partial x}| + |\frac{\partial p}{\partial y}|$$
                    $$\frac{\partial p}{\partial x} = \frac{value(p(i, j+1)) - value(p(i, j-1))}{2}$$
                    $$\frac{\partial p}{\partial y} = \frac{value(p(i+1, j)) - value(p(i-1, j))}{2}$$
9:          end if
10:     end for
11:     Compute dp
12:     for all $p \in E$ do
13:         if $p \in bottom$ then
14:             Compute dp as E(i, j)
15:         else if $p \in E \backslash bottom$ then
16:             Compute dp given by:
                    $$dp(i, j) = E(i, j) + min(dp(i-1, j-1), dp(i-1, j), dp(i-1, j+1))$$
17:         end if
18:     end for
```

[*]https://github.com/sdc17/NaiveIR

```
19:     Backtracing the seam
20:     for all $p \in dp$ do
21:         if $p \in bottom$ then
22:             Compute trace as argmin(dp(i, :))
23:         else if $p \in dp \backslash bottom$ then
24:             Compute dp given by:
                $$trace = last + argmin(dp(i, last-1), dp(i, last), dp(i, last+1)) - 1$$
25:         end if
26:     end for
27:     Remove seam from $I$ according to trace
28: end for
```

1.2 Result



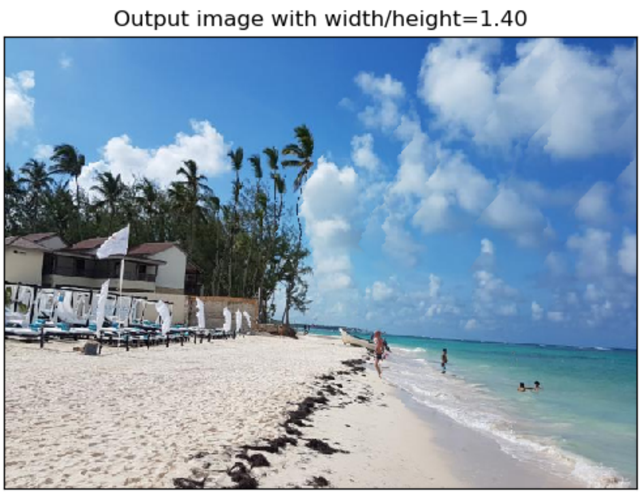


Figure 1: Aspect Ratio Change

Figure 1 中上图为原图，宽长比为 1.83，下图将其改为了 1.40。可见 Aspect Ratio Change 后的图较好地保留了原图中能量较高的信息，同时压缩了图片的宽度。

2 Object Removal and Image Enlarging

核心代码和 Aspect Ration Change 有很多重合，这部分不再详写。

此外，在实验中发现使用 seam 的 k 个全局最小会使得低能量 seam 密集的区域产生拉伸感的 artifact，所以在取 seam 时限定了第 i 个 seam 的终点和第 $i+1$ 个 seam 的终点之间的间距要大于 $2i$，否则第 i 个 seam 往后顺延直到第 i' 个 seam 的终点和第 $i'+1$ 个 seam 的终点之间的间距大于 $2i$。

2.1 Procedure

Algorithm 2 Aspect Ration Change

```
1:  Object remove
2:  while ∃ labeled piexl ∈ I do
3:      Get next labeled piexl to remove from up to bottom, left to right
4:      Compute energy the same as Algorithm1
5:      Compute dp the same as Algorithm1
6:      Compute trace above p(i, :) and below p(i, :) respectively to get a seam through the pixel
7:      Remove seam form $I$ accroding to trace
8:  end while
9:  Image Enlarging
10: Compute energy the same as Algorithm1
11: Compute dp the same as Algorithm1
12: Compute first $k$ seam, in which $k$ euqals to the difference between width of input image and
    image after removal
13: Duplicate all of seams to the right side of origin seams given by last step
```

2.2 Result

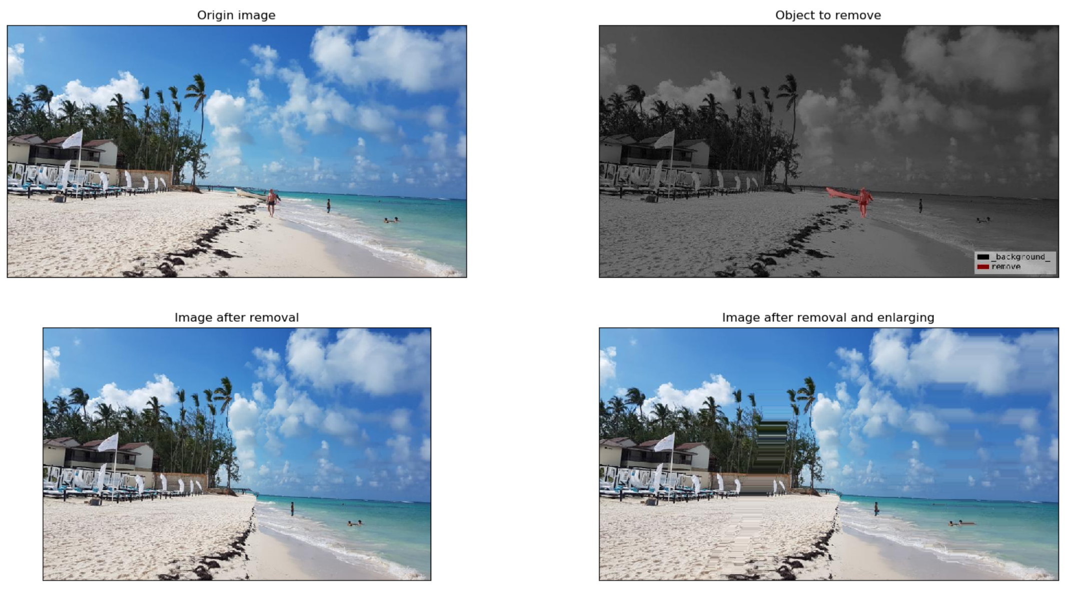


Figure 2: Object Removal and Image Enlarging

Figure 2中左上为原图，右上拿 labelme 标出了需要被移除的物体——人和他后面的

船，左下是移除指定物体后的结果，右下是在左下的基础上做 Image Enlarging 恢复到原图尺寸的结果。从左下图可见 Object Removal 后的图完整地移除了指定的物体且较好地保留了原图中其他部分的空间结构，从右下图可见 Image Enlarging 后的图恢复到了原图尺寸且对能量较小的 seam 做了复制，但是左中部还是有较为明显的 artifacts。

其原因在于图左中部树林部分像素值很相近，图像梯度小，能量小。从图像底部开始做 Backtracing 时，很大范围内 seam 的路径走到图像中部都会经过这块绿色的低能量区域，造成这个区域被重复复制，进而产生拉伸感的 artifacts。解决方案可以尝试从图像 y 轴的中部作为 Backtracing 的起点，然后向上下两侧分别做 Backtracing 和 Forwardtracing 找到 seam.

3 Conclusion

这次作业加深了我对于传统 Image Resize 和 Content-Aware Image Resizing 的理解，动手实现 Aspect Ratio Change, Object Remove 和 Image Enlarging 的过程则让我对于 numpy 的使用变得更加熟练。