

SeamCarving实现图像智能缩能缩放

2017013632 石大川

实现功能

- 所有基本要求
- 分阶段的图像扩展
- 目标移除
- 改进的能量公式

实现说明

- 基本算法

```
for all Seam to remove do
    Compute Energy,  $I$  denotes input pixels space,  $\Omega$  denotes its boundary,
    for all  $p \in I$  do
        if  $p \in \Omega$  then
            Compute energy given by three existing surrounding pixels
        else if  $p \in I \setminus \Omega$  then
            Compute energy given by:
            
$$E(p) = \left| \frac{\partial p}{\partial x} \right| + \left| \frac{\partial p}{\partial y} \right|$$

            
$$\frac{\partial p}{\partial x} = \frac{value(p(i, j+1)) - value(p(i, j-1))}{2}$$

            
$$\frac{\partial p}{\partial y} = \frac{value(p(i+1, j)) - value(p(i-1, j))}{2}$$

        end if
    end for
    Compute dp
    for all  $p \in E$  do
        if  $p \in bottom$  then
            Compute dp as  $E(i, j)$ 
        else if  $p \in E \setminus bottom$  then
            Compute dp given by:
            
$$dp(i, j) = E(i, j) + \min(dp(i-1, j-1), dp(i-1, j), dp(i-1, j+1))$$

        end if
    end for
    Backtracing the seam
    for all  $p \in dp$  do
        if  $p \in bottom$  then
            Compute trace as  $\text{argmin}(dp(i, :))$ 
        else if  $p \in dp \setminus bottom$  then
            Compute dp given by:
            
$$trace = last + \text{argmin}(dp(i, last-1), dp(i, last), dp(i, last+1)) - 1$$

        end if
    end for
    Remove seam from  $I$  according to trace
end for
```

@sdc17

- 水平缩小:

```
python -W ignore CAResize.py -i pic1.png -tw 400
```



左图为原图，右图为实验结果。可见在减小图片宽度的同时较好地保留了重要的区域。

- 坚直缩小：

```
python -W ignore CAResize.py -i pic2.jpg -th 250
```



左图为原图，右图为实验结果。可见在减小图片高度的同时较好地保留了重要的区域。

- 能量函数：

默认为Gradient，修改generate_energy函数中的参数mode为HoG后：

```
python -W ignore CAResize.py -i pic1.png -tw 400
```



左图为Gradient，右图为HoG。从红框标出的区域可见对于这张图片Gradient的效果更好。

- 分阶段的图像扩展

我实现了分阶段的图像扩展，限制了每个阶段扩展的细缝数为总细缝数目的 $\frac{1}{3}$ ，这个比例可以通过修改insert_vertical_seam函数中的scale参数来改变。

除了将最后的按回溯的位置信息删除seam改为添加seam外，单阶段扩展的核心算法和缩小所用的算法是高度重合的。分阶段的扩展就是在此基础上将每次添加的seam数目控制在一定的范围内。

```
python -W ignore CAResize.py -i pic3.jpg -tw 700
```



左图为原图，右图为扩展后的图，可见被扩展的区域多集中于能量较低的浪花区域，这是符合预期的。

- 目标移除

我用labelme标出了需要被移除的目标，遍历返回的label文件，直到里面所有被标记的点都被删除：



```
python -W ignore CAResize.py -i beach.jpg -m beach.npy
```



左图为原图，右图为移除指定目标后的结果，可见算法较好地实现了对象移除的功能。

- 改进的能量公式

按照PPT给的公式修改原来的能量公式即可。

修改remove_vertical_seam函数中的参数mode为forward后：

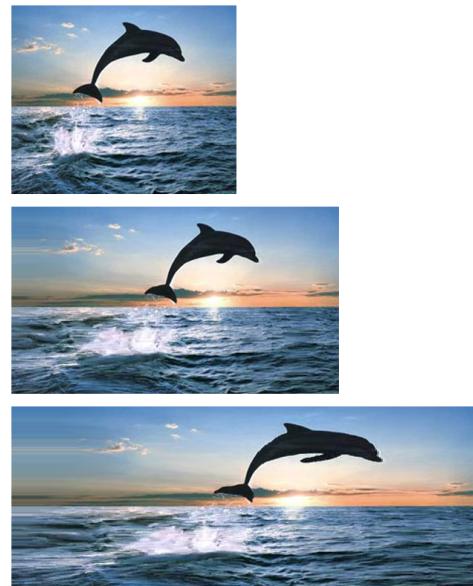
```
python -W ignore CAResize.py -i pic1.png -tw 400
```



左图为原能量公式的结果，右图为前向能量公式的结果，观察图像中间的两根白色柱子和图像下部的斜台阶可见，前向能量公式生成的结果顺滑了很多。

- 效果不佳的图片

以分阶段的图像扩展为例：



最上方为原图，中间和下方为扩展图。观察最下方图片的鲸鱼和水花可见Artifacts变得比较明显了。这是因为当复制的seam过多时算法的效果就类似于直接做resize了，所以会产生明显的拉伸感。