Seam Carving 实验报告

计62 金镇书 2016080036

1. 算法流程

1.1 Carving

能量函数：

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

首先，按照论文中定义的能量函数，获取energy map，其中每个元素代表一个个pixel的能量

```
def get_energy(way, img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    dx = x_gradient(gray)
    dy = y_gradient(gray)
    energy = cv2.add(np.absolute(dx), np.absolute(dy))
```

$$M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

然后通过论文中提到的动态规划的方法，计算得到M矩阵，我们可以再M的最后一行得到最小能量，之后开始向上回溯，寻找最小的八连通路径。需要注意的是找到的路径一定要是联通的

```
for i in range(1, height):
    # 为了不计算i=0，从1开始
    for j in range(width):
        if j == 0:
            M[i, j] = energy[i, j] + min(M[i-1, j], M[i-1, j+1], M[i-1,
j+2])
        elif j == width-1:
            M[i, j] = energy[i, j] + min(M[i-1, j-2], M[i-1, j-1], M[i-1,
j])
        else:
            M[i, j] = energy[i, j] + min(M[i-1, j-1], M[i-1, j], M[i-1,
j+1])
```

```
last_min = np.argmin(M[height-1, :])
seam.append([height-1, last_min])
for i in reversed(range(height-1)):
    if last_min == 0:
        last_min = last_min + np.argmin([M[i, last_min], M[i,
last_min+1]])
    elif last_min == width-1:
        last_min = last_min + np.argmin([M[i, last_min-1], M[i,
last_min]]) - 1
    else:
        # print(M[i, last_min-1],  M[i, last_min], M[i, last_min+1])
        last_min = last_min + np.argmin([M[i, last_min-1], M[i, last_min],
M[i, last_min+1]]) - 1
    # print(last_min)
    seam.append([i, last_min])
```

找到路径，即seam之后，对原图进行删除（跳过seam对应的像素，从原图转移像素）

```
def carve(way, M, img, seam):
    # print("carve_img:", img.shape)
    img_height, img_width, img_channel = img.shape
        # need to initialize first value at the bottom
        # remove seam that we dont need (vertical)
    new_img = np.zeros((img_height, img_width-1, img_channel), np.uint8)
    for i, j in seam:
        for k in range(j):
            new_img[i, k] = img[i, k]
        for k in range(j, img_width-1):
            new_img[i, k] = img[i, k+1]
```

继续迭代上述的各个步骤，直到长度缩小到新长度即可。

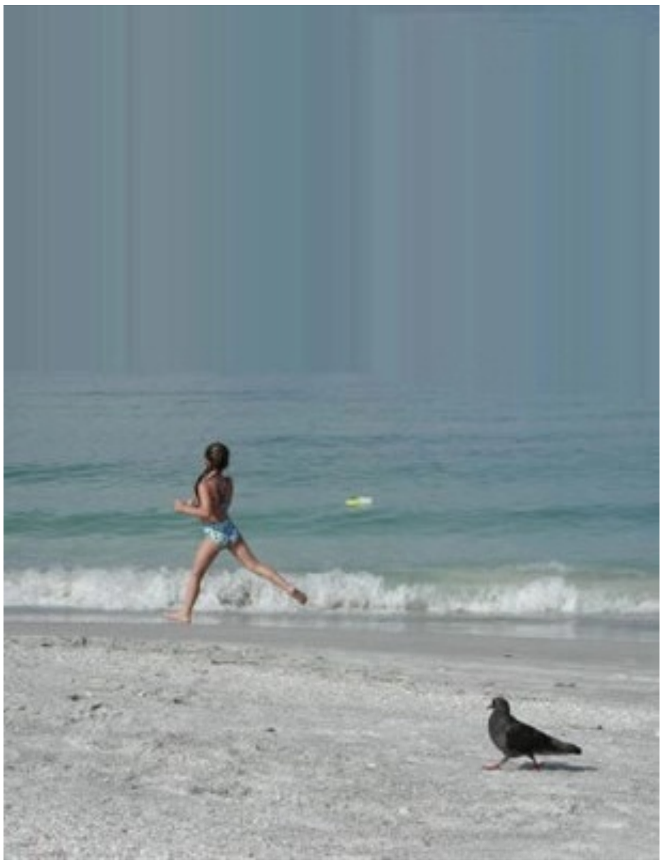
总结：

1. 计算能量值
2. 动态规划求值
3. 回溯求seam
4. 删除seam
5. 重复1，2，3，4

上面针对于竖直方向的seam，对于水平方向只需要换个坐标，详情见源码。

1.2 Enlarge

一开始我以为只要像缩小一样，只需要每次对单个seam进行复制，增加一个像素，却发现效果很奇怪，感觉就复制了同样的seam：



之后发现我没有好好的看论文，修改后的算法为：

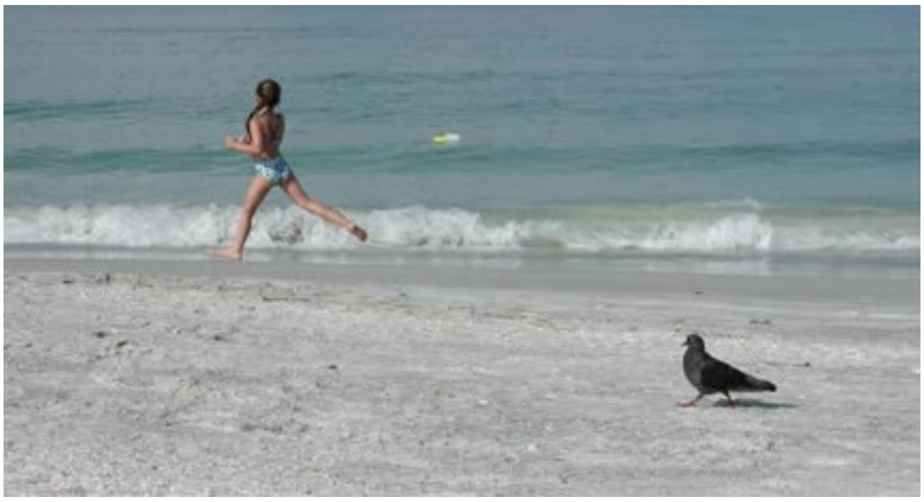
1. 找出多条seam
2. 最后进行复制

对于算法的第1条：

首先像之前的carving一样，先计算energy map，动态规划算出M之后，得到了seam，这时我们可以维护一个布尔矩阵，在seam对应的位置标志True，然后对图片进行缩放。

因为之前直接对单个seam复制的话，每次都只会找到同一个seam，所以通过carving的方法，找到所有的seam，最一起期进行放大：（水平的效果好些所以放了水平的效果图）

原图



放大效果图

放大部分代码：

```
def save_seam(way, seam_mask, seam, seam_count):
    height, width = seam_mask.shape
    if seam_count == 0:
        for i, j in seam:
            seam_mask[i, j] = True
    else:
        # 除了第一次直接对mask赋值以外，因为每次缩小了图得到的seam的坐标，与原图已经对
不上了
        # 所以需要用计数器，确定缩小后的坐标到原图坐标的映射
        for i, j in seam:
            # print("i and last_min:", i, j)
            k = 0
            counter = -1
            for k in range(width):
                # print("iteration :" , k, seam_mask[i, k])
                if seam_mask[i, k] == False:
                    counter += 1
                if counter == j:
                    seam_mask[i, k] = True
                    break
    return seam_mask


def get_seams(way, img, scale):
    seam_count = int(width * scale)
    for i in tqdm.trange(seam_count):
        M = get_energy(way, carve_img)
        seam = get_seam(way, M, carve_img)
        seam_mask = save_seam(way, seam_mask, seam, i)
        carve_img = carve(way, M, carve_img, seam)

    return seam_mask, seam_count

def enlarge_img(way, img, width, height, scale):
    # 放大比例
    seams_percentage = scale
    img_height, img_width, img_channel = img.shape
    seam_mask, seam_count = get_seams(0, img, seams_percentage)
    new_image = np.zeros((img_height, img_width + seam_count ,
img_channel), np.uint8)

    for i in tqdm.trange(img_height):
        p = 0 # 维护指针进行赋值
        for j in range(img_width):
            if seam_mask[i, j] == True:
                new_image[i, p] = img[i, j]
                p += 1
                new_image[i, p] = img[i, j]
            else:
                new_image[i, p] = img[i, j]
            p += 1
```