Seamcarving 作业报告

一、基本算法

为了使内容敏感的部分尽量保持完整,我采用 seamcarving 算法。

作为预备,先创建几个基础类: Color 储存颜色信息,有 rgb 三个 unsigned char。Node 储存像素点的信息,Pic 储存图片信息。

在 Node 中,有以下信息:

```
Color c;
double e; //energy
int x; //location on oringin picture
int y; //location on oringin picture
int prev; //previous index, used to backtrack
Type type;
```

其中 type 有七种,1/2/3/4 用于输出 seam 示意图。 Pic 中有三个像素矩阵,分别储存原本图片,当前图片和放大后图片,使用 Node*方便储存引用。用 direct 记录当前操作方向。使用 opencv 的 mat 类进行输入输出。

Jenum Type {
 ori = 0, //oringin
 remX = 1, //removeX
 remY = 2, //removeY
 del = 3, //delete
 pro = 4, //protect
 insX = 5, //insertX
 insY = 6 //insertY

1、缩小

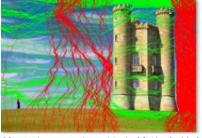
缩小时,进行循环,次数为缩小的 seam 条数。

a. 先使用 seam()函数得到一条 seam 的起始位置。在 seam()中,先使用 sobel()函数遍历图片,更新每个像素的能量。Sobel()函数计算每个 点 周 围 的 颜 色 , 采 用 卷 积 矩

$$G_x = egin{bmatrix} -1 & 0 & 1 \ -2 & 0 & 2 \ -1 & 0 & 1 \end{bmatrix} G_y = egin{bmatrix} 1 & 2 & 1 \ 0 & 0 & 0 \ -1 & -2 & -1 \end{bmatrix}_{ ext{#30}}$$
 $ext{#30}$

则取 0。采用多种返回值,实际测试发现前三种结果类似,第四种效果较差,不能兼顾图片双向的信息。

```
double x = rx * rx + gx * gx + bx * bx;
double y = ry * ry + gy * gy + by * by;
double s = x + y;
//return sqrt(s);
//return s;
return abs(rx) + abs(ry) + abs(bx) + abs(by) + abs(gx) + abs(gy);
//return (direct == X)? sqrt(x): sqrt(y);
```





- b. 接着遍历图片,从当前像素的前序三个参数中选择能量最小的像素作为前序像素,并将其能量加到自身。遍历结束后从末尾找到能量最小的像素,即为能量最小的 seam 的起点。返回起点 index。
- c. 从 index 开始回溯, 将遇到的像素从当前图片中删去, 并改变原始 图片中对应像素点的类型。
- d. 输出时, 先输出当前图片为结果, 再将原始图片中改变类型的点换

成红或者绿色,输出 seam 标记图。

换个方向, 重复以上循环。

2、删除与保护

在开始循环前,将需要删除的部分的能量先设置成一个极小值并不再改变,将需要保护的部分的能量先设置成一个极大值并不再改变,这样在第 2 步的选择前序像素中,必定选择删除的点,不能选择保护的点。这样达到了删除与保护的作业。其他与缩小的步骤一致。

3、扩大

先在一个方向完成缩小的循环,接着使用经过缩小步骤中更改类型的原始 图片,遍历,将图中标记为删除的点,根据其周围的颜色插值复制到下一 位像素。最后删去了多少,则复制(多)了多少个像素,并将新增的像素 类型改变。

使用 recover()函数,将单向扩增的图片作为原始图片,更改当前图片,并在另一个方向重复以上操作。