第四次作业报告

PB21010362 汪兆辰

2023 年 4 月 2 日

1 算法介绍

图像的缩放过程可以通过在视觉效果显著性低的区域删掉缝隙达到保留主要内容信息的效果，为了实现这一效果，我们需要对图像定义能量，并生成能量图，在能量较低的区域进行操作. 方便起见，这里只讨论调整宽度的情况，即缝隙全部为从上到下的，因为对于调整高度的情况，我们只需要将图像转置再调整宽度即可.

为了找到能量最低的缝隙，我们用与原图像尺寸相同的矩阵 Mat 来存储每一点的累计能量，其中 Mat 的第一行初始化为能量图的每点能量，从第二行开始，利用递推关系式：

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

Mat 的每点元素即为能量图上该点能量与上方三个点能量最小值的和，也即该点的累计能量. 从而在 Mat 的最后一行可以找到最小的累计能量，由这一点向上回溯即可找到能量最小的一条缝隙.

2 算法实现

在上述算法的实现中，用二维的 vector 来存储矩阵 Mat，令缝隙当前位置的横坐标为 loc，则下一个位置在上一行的相邻三格中，回溯的代码如下：

```
if(Loc[i]==0) {
    Loc[i]=Mat[r-1][0]>Mat[r-1][1]?1:0;
}
else  if(Loc[i]==wd-1){
    Loc[i]=Mat[r-1][wd-2]>Mat[r-1][wd-1]?wd-1:wd-2;
}
else {
    if(Mat[r-1][Loc[i]-1]>Mat[r-1][Loc[i]+1]){
        if(Mat[r-1][Loc[i]]>Mat[r-1][Loc[i]+1])Loc[i]=Loc[i]+1;
```

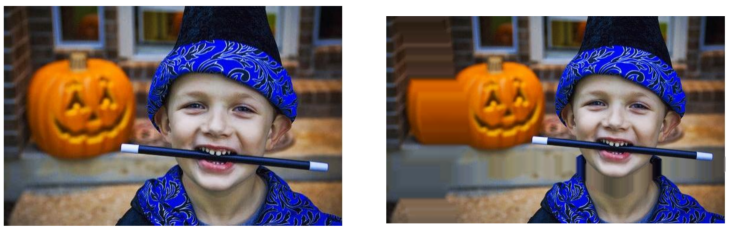
```
        }
        else  if(Mat[r-1][Loc[i]]>Mat[r-1][Loc[i]-1])Loc[i]=Loc[i]-1;
    }
```

在找到缝隙在每一行的位置后，只需将缝隙以外的点复制，而缝隙点按照算法要求被删除或插值. 由于最后生成新图像需要参数的类型为 *vector<BYTE>, 考虑到对原图像的复制顺序用队列存储图像的每一行，用栈存储由每一行生成的队列，在原图被复制完后，只需要按顺序退栈与出队列即可得到按顺序排列的像素点列 vector，从而生成处理过的新图像.

3 处理效果

(a) Row (b) Shrink

图 1: 缩小效果



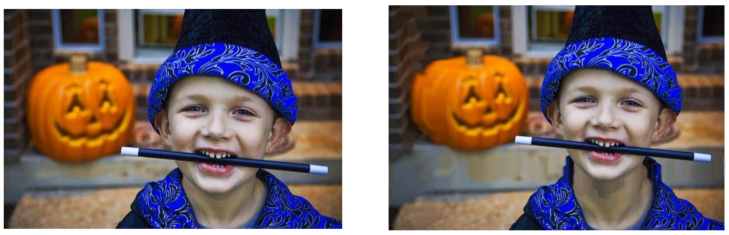
(a) Row (b) Expand

图 2: 放大效果

注意到，在删除的过程中人脸被完全保留下来，在放大过程中人脸也没有变化. 但是增加是在能量最小的地方增加一条缝隙，为了保持图像的一致性，增加的缝隙点是从点插值得到的，这也导致了几乎所有缝隙都加在了同一个地方.

一种改进方法是一次增添数条缝隙，但由于上述算法生成缝隙是从最后一行向上回溯，对于两个不同点生成的缝隙，无法保证两条缝隙不相交，而相交后两条缝隙将变为一条. 若将所有不相交的缝隙全部计算出来再选取其中能量较小的几条，无论从时间还是空间复杂度上看都难以实现. 因此考虑当两条缝隙相交后，保留其中的一条，并在相交的这一行中找到一个能量较小的点作为起点，再生成

一条缝隙，结果如下，其中参数设置为 step=2：处理的结果表明，当 step 设置相对较小 (如 2,3) 时，



(a) Row (b) Expand

图 3: 改进结果

能够保留图像的主要信息，并一定程度改善同一条缝隙反复复制的情况，但是当 step 较大后，图像信息会被破坏.

另一种改进方法是在添加缝隙的时候考虑对同样的一张副本图进行删除缝隙操作，而将副本删除掉的缝隙复制到目标图的对应位置，这样就保证了每条缝都不会重合，同时也保持了操作后图像的一致性，但是鉴于时间原因代码没有调通，只在 main.cpp 中 widthAdding 函数展示，并未调用.

参考文献

[1] Shai Avidan and Ariel Shamir. Seam Carving for Content-Aware Image Resizing.SIGGRAPH2007.