**Readme**

1. **代码说明**

Project当中使用的源代码，大致可以分为如下几个部分：

第一，读入图片，并创建图片的能量矩阵，也就是代码中的createEnergyImage函数，这一部分源代码是采用梯度绝对值的能量函数，主要是通过调用函数来实现的。

第二，是在能量函数的基础上，根据要求选择需要删去的seam的位置，即代码中的createCumulativeEnergyMap和findOptimalSeam两个函数。源代码中的实现方法是通过首先在一个新的矩阵cumulative\_energy\_map中计算出能量值的和函数，这样利用和函数，就可以直接比较最后一行(列)的函数值，找到能量最小的位置。

第三，在已知seam位置后，就可以在原图上进行删去接缝，显示，输出运行时间和结果图等操作，对应于代码的其他部分。

这一部分project的主要目标是研究不同seam carving的实现方式，因此主要修改的是第二部分的代码，没有涉及到能量函数的改动，同时在实现过程中没有用到createCumulativeEnergyMap这一函数(因此在代码中已经删去这一部分，但其他地方的名称并没有改动)。主要是用不同方式实现findOptimalSeam这一函数，比较输出的运行时间。

代码在实现过程中每删除一条seam，就会进行一次能量矩阵的更新。

代码运行所需要的**工具**：visual studio 2017，opencv

**环境配置**：参考链接<https://www.cnblogs.com/YiYA-blog/p/10296224.html>

1. **主要工作**

通过参考论文和网上搜索，了解到图片的seam carving采用的主要实现方法包括图算法中的dijskra，拓扑排序以及动态规划。因此这一部分采用两种图算法以及动态规划和源代码4种代码实现分别运行，比较运行时间，运行结果在excel表中给出，运行效果也可以在相应的文件夹中查看(这里只保留了两种效果，因为其它两种得到的path路径与这两种相同，这里保留的2种是有所区别的)。

4种代码对应部分保留在实验代码文件中，另附有实验版本1文件，包括两种图算法的最初版本。(最初版本中的代码运行速度更慢一些，但其实现方式更接近于图算法本身的表述)。

(测试图片为test1和test2)

1. **使用方法**

在project1文件中，直接在findOptimalSeam函数中把对应的if和else if部分替换，可以换成实验代码中对应的任意部分，直接运行即可。

1. **理论分析**

**Dijskra**算法选择矩阵中据源点距离最近的点加入集合中，并更新距离值。在每层循环中，最多要更新3个距离值，最多将3个点加入到优先队列中。根据堆结构的特点入队复杂度为O(logn)，因此总的计算复杂度大概为3+3logV。而这一循环最坏情况下对每个点都要执行一次，如果以比较为基本运算的话，设总的像素点数为V，总复杂度为O(3V+3VlogV)。

**优点：**只需要有一个最后一行的点加入集合，就可以结束算法，因此对于一些能量线比较明显，能量函数差异较大的图，可能可以减少遍历的顶点数。

**缺点：**需要先构造图结构，为了挑选距离最近的顶点，需要每个循环中进行比较，使用堆结构优化后，仍然会使复杂度增加。

**拓扑排序算法**从源点开始对整个图结构进行拓扑排序，依据拓扑排序的序列，对每个顶点进行放松操作，即如果有dist(u,v)+d(u)<d(v)，则d(v)=dist(u,v)+d(u)。没了达到目的点t，需要对图中每个顶点做放松，即O(V)次操作。而每次放松，需要进行至多3次比较操作，因此总复杂度为O(3V)，在参考论文中有给出理论分析，结论是一个时间下界O(5V)。（不包括构图时间）。

**优点：**相对dijskra算法，复杂度有了降低，避免对距离进行排序和统一顶点多次入队。

**缺点：**需要进行拓扑排序和构造图结构，必需遍历每个像素点。

**动态规划算法**直接计算到达这一像素点，且包含这一像素点的能量线的能量值总和，从上到下，从左到右依次计算，总共需计算O(V)次，在每个循环中至多需要进行3次比较，因此总的复杂度为O(3V)。

**优点：**直接在矩阵上进行操作，复杂度最低。

**实验代码对源代码的优化：**减少了能量矩阵的复制和函数调用，在求最低能量线的过程中，减少了一次比较。

**实验结论：**相比于另两种图算法，在大多数情况下，动态规划算法的时间效率更高，它可以直接在能量矩阵上进行操作，而不需要多余的构造。理论上来说要求得最低能量线，必须遍历每一个像素点的值，否则都可能无法取得最值，因此时间复杂度最低为O(V)，因此目前来说，动态规划算法是实现这一功能的最佳算法。

**可能的改进：**进一步降低时间复杂度，可能需要不每次更新能量函数，这样在每次求能量线时，也可以只对一部分做操作。当然，不及时更新能量函数可能会对图片效果造成一定的影响。