问题1

算法描述：

读入两张图片

混合图片=图片1

For 圆形半径 小于 对角线长度

Foreach 像素 in 混合图片

If 该像素在圆形区域内

混合图片圆形半径内某位置的像素=图片2该位置上的像素

半径扩大

关闭图片窗口

辅助函数函数：输入矩阵上某个坐标，判断该坐标是否位于圆形区域内

优化1 按照以上算法，对于已经在圈内的像素点，会被重复判断是否在圆形区域内，这样会带来无用的性能消耗。 作为优化的算法，只用判断由于半径扩大所带来的“新”的圆环部分的像素点即可

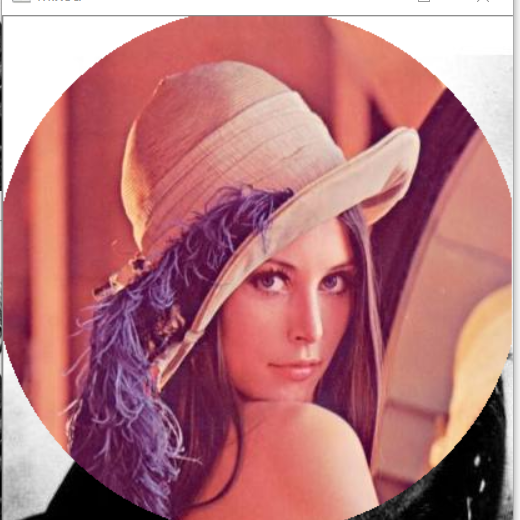
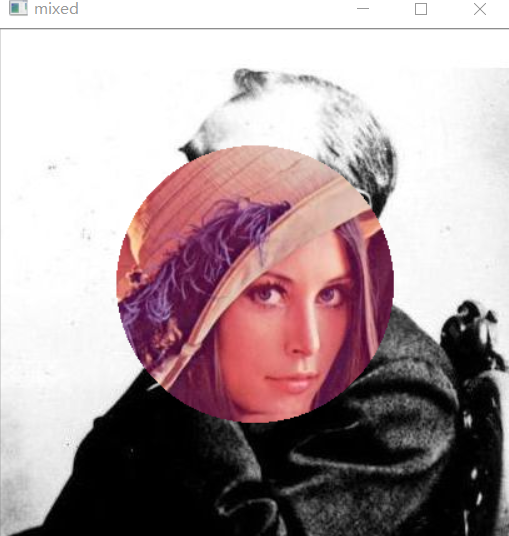
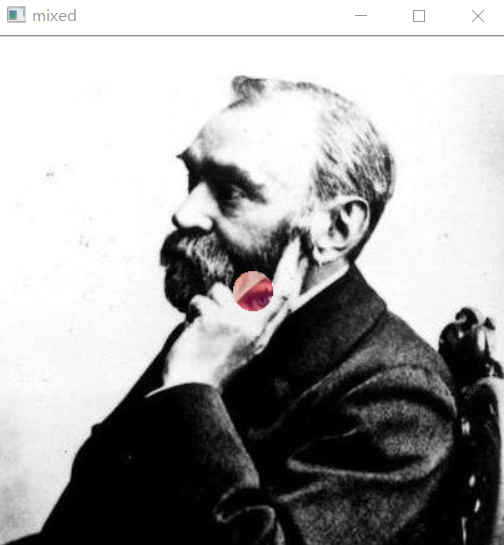
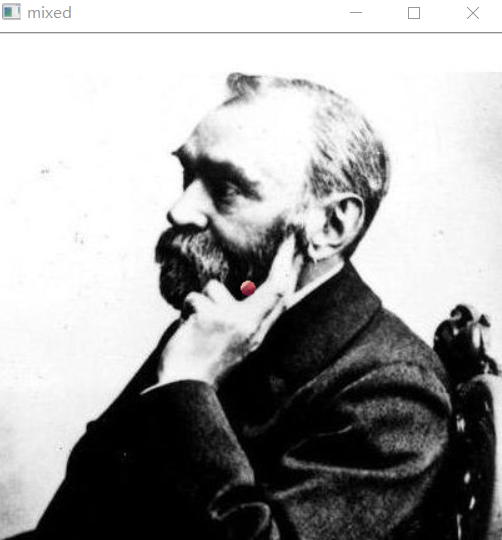
程序实现：

语言及库：python3.7 cv2库

（未优化版本）



实现效果



问题2

算法描述：

这是一个LUT算法

本次作业采用二叉递归树的方法

步骤为：

1. 按照红绿蓝的次序划分八次，得到256个块

2. 每个块计算其中心颜色

3. 把原颜色替换成中心颜色

4. 得到新位码（8bit）和颜色的对应表

优化1. 如果随机取中心颜色会造成黑点，所以采用平均值的方法

优化2. 只用优化1导致一些方块边缘的颜色出现突变，可以采用计算欧拉距离取最小的方法减少突变

计算优化：

递归部分伪代码

#传入块 和 上一层次赋予的值

#if depth==8 计算结果保存并输出

# else{ 先序遍历 进行下一层递归

# 先给块内所有的颜色附上位值

#求中值

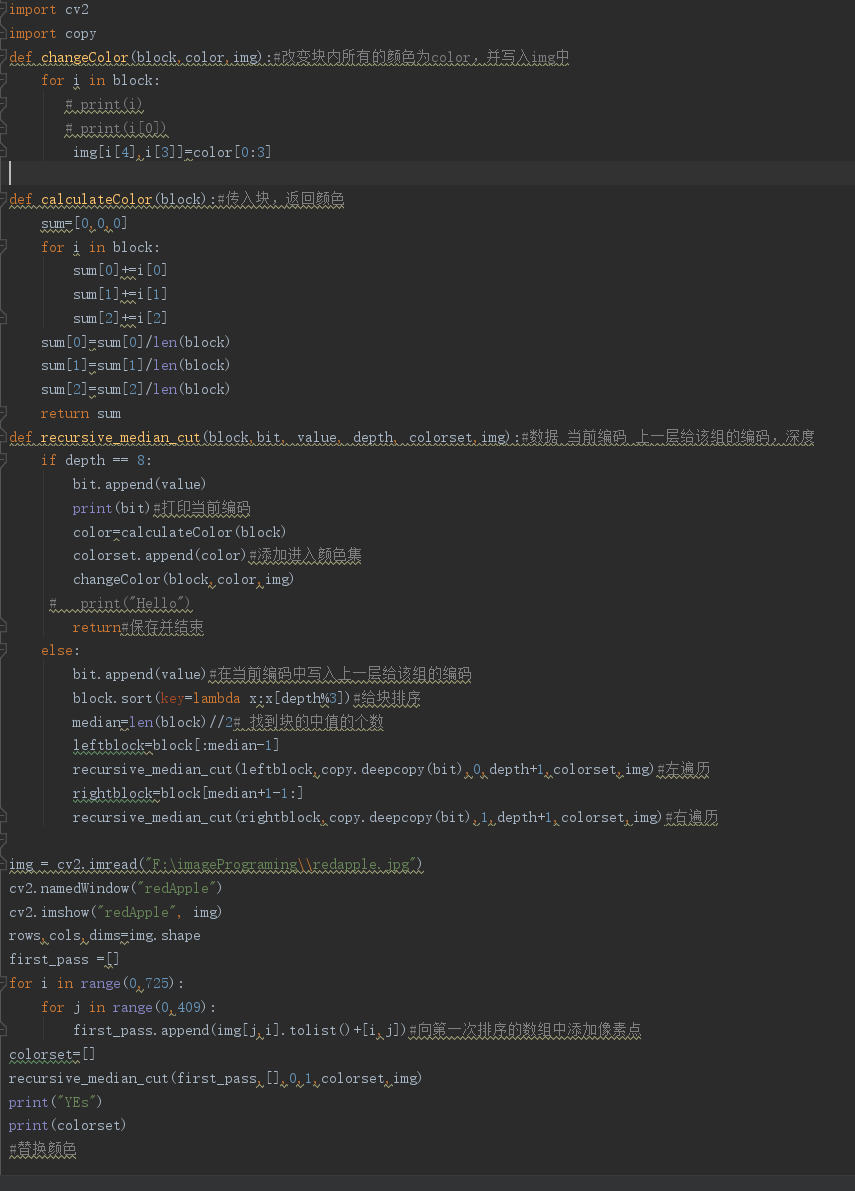
#左子树（小块和0,depth++）

#右子树（大块和1,depth++）

Briefly explain: 因为比起蓝色，人类对红色更加敏感

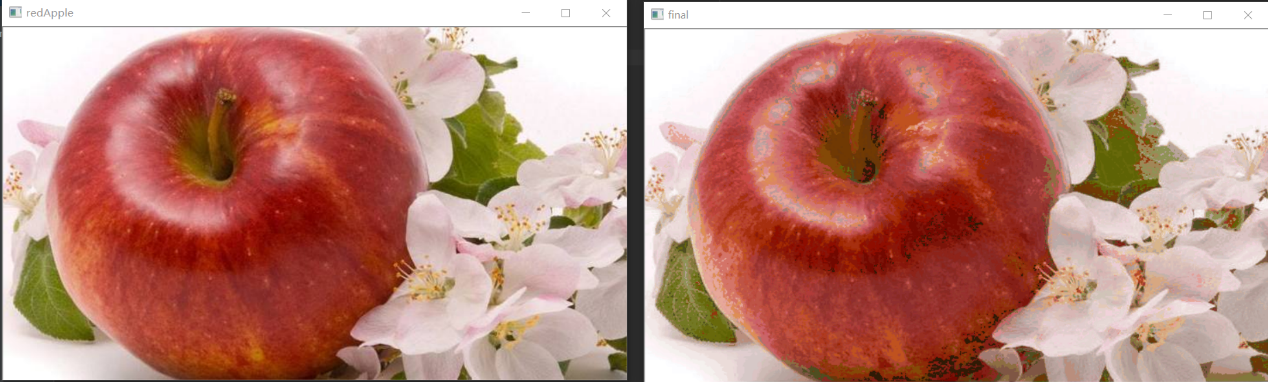
程序实现：

使用库cv2，python3.7 JetBrainspycharm

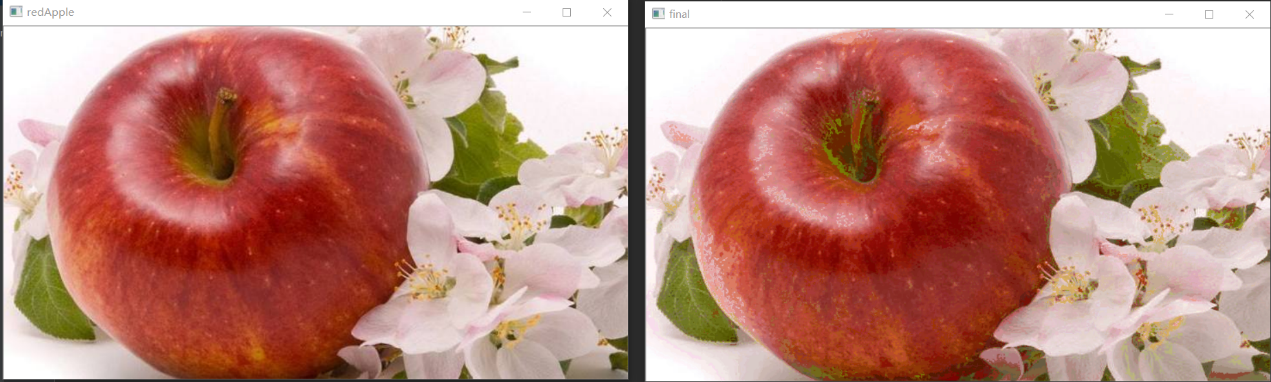


实现效果：

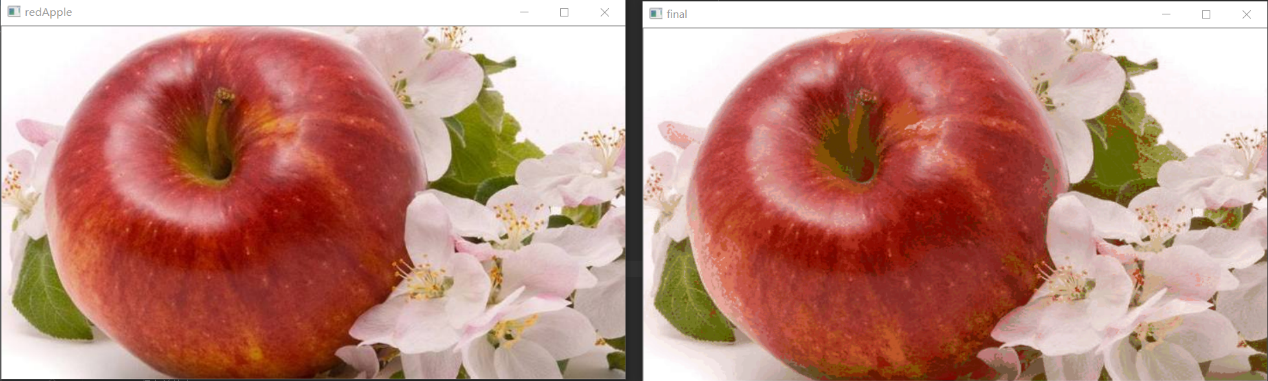
7bit调色板（未优化）



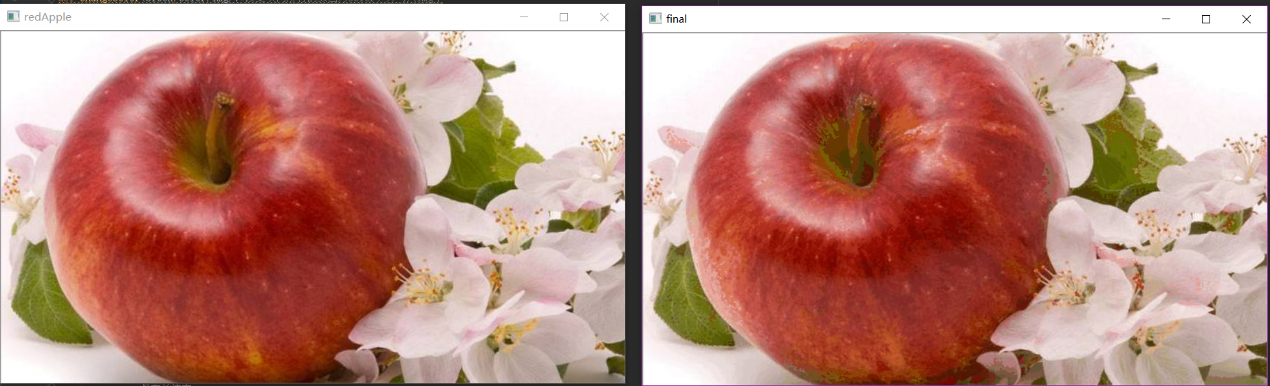
8bit调色板（未优化）



7bit调色板（优化）



8bit调色板（优化）



色码（8bit：rgbrgbrg）对应RGB颜色

说明：第一位无意义，必定是0

