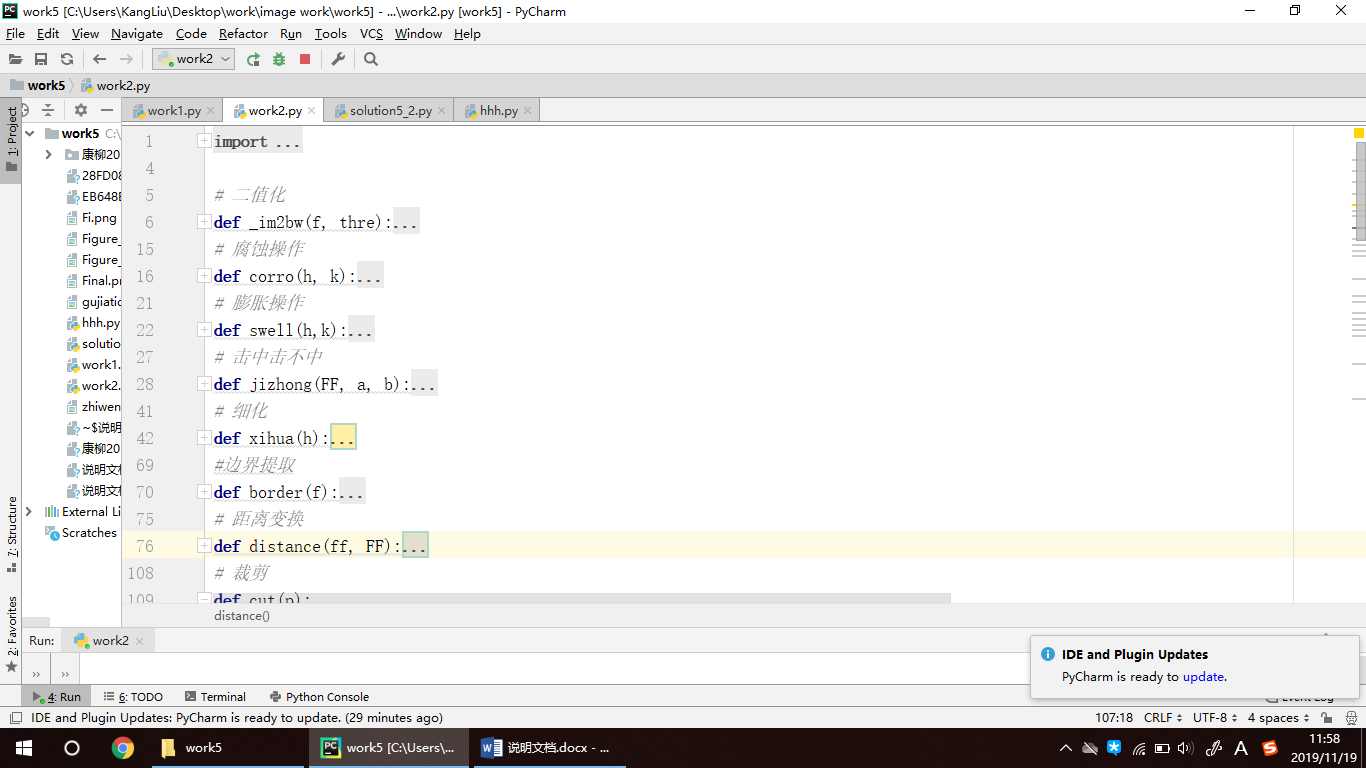
注：该代码在本人电脑上跑了大概10min左右，比较长。

下图为本次作业程序中的所有函数。

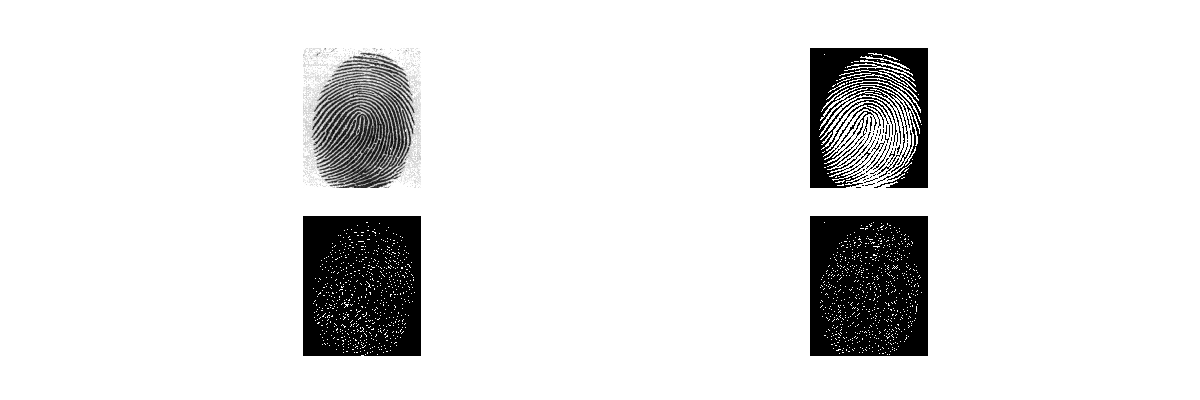
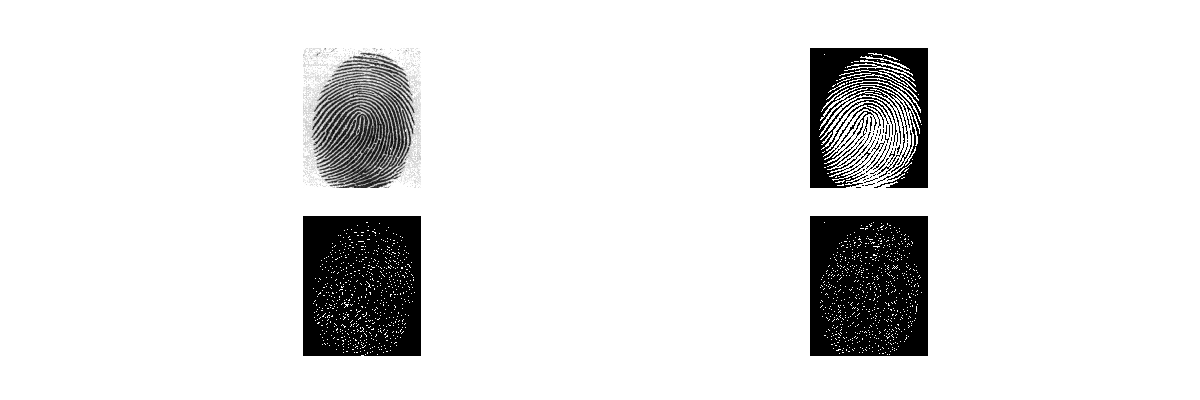


**问题描述：对图像二值化（阈值法）**

**代码思路：**设定一个初始阈值127，大于127的置0，小于127的置1。

F = \_im2bw(f, 127)

**结果：**左面为读入的原指纹图，右面为二值图。



**问题描述：形态学骨架提取**

**代码思路：**细化图像。f\_xihua = xihua(F)

用结构元去与二值图做击中击不中，击中击不中：用B1去腐蚀X，然后用B2去腐蚀X的补集，得到的结果取交集就是击中击不中变换。再用原图与之结果做差，如此反复，用八个结构元一直轮流迭代，直到得到的结果不再发生变化。

【腐蚀操作：卷积cv2.filter2D(h, -1, k)

保留全为1的img[np.where(\_img == np.sum(k))] = 1】

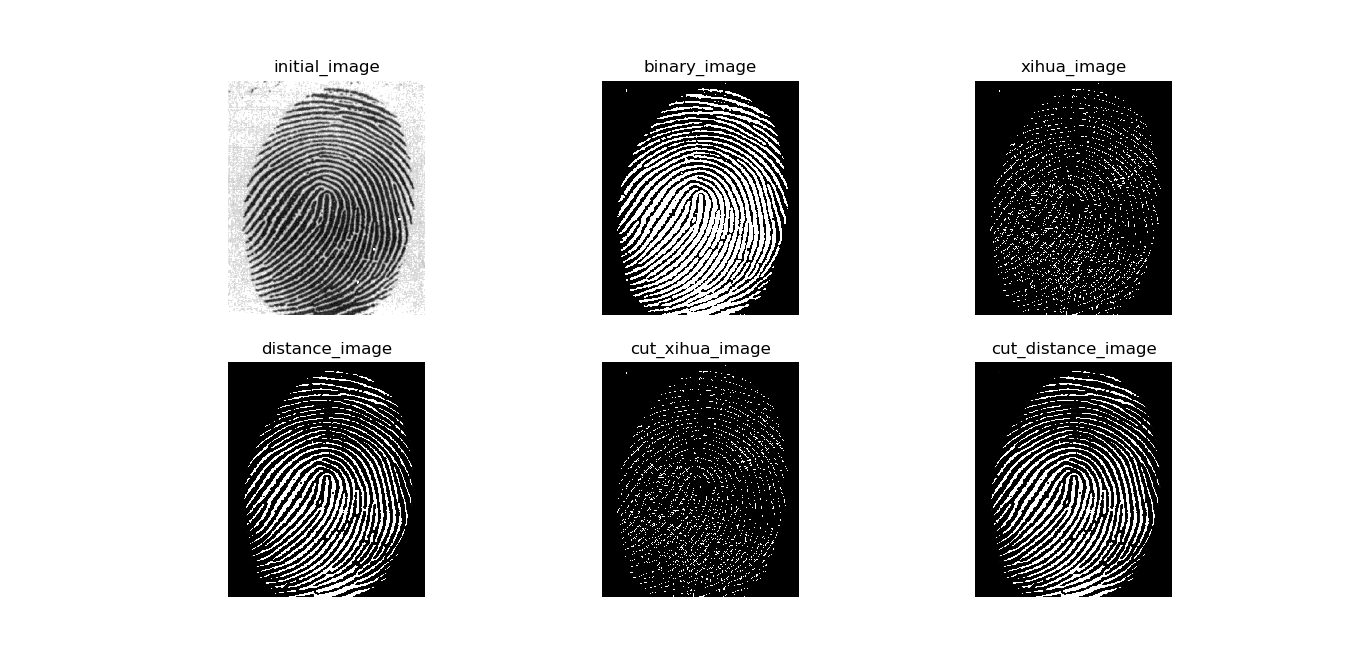
**结果：**左图为二值图，右图为细化图像结果图。



**问题描述：距离变换骨架提取**

**代码思路：**把背景0置无穷大【if0则置为255】，提取边界【二值图减腐蚀结果h = hh-corro(hh,b)】，用串行距离变换算法，即两个模板，先用q1 = np.array([[4,3,4],[3,0]])从上到下，从左往右与F图相加，取最小值，再通q2 = np.array([[3,0],[4,3,4]]) 从下到上，从右往左与刚刚得到的图相加，依然取最小值，最后自己定义一个5\*5的邻域找局部极大值，判断这个值是否大于它的所有邻域值，是的话保留，否的话置0，还原，0置255，255置0。

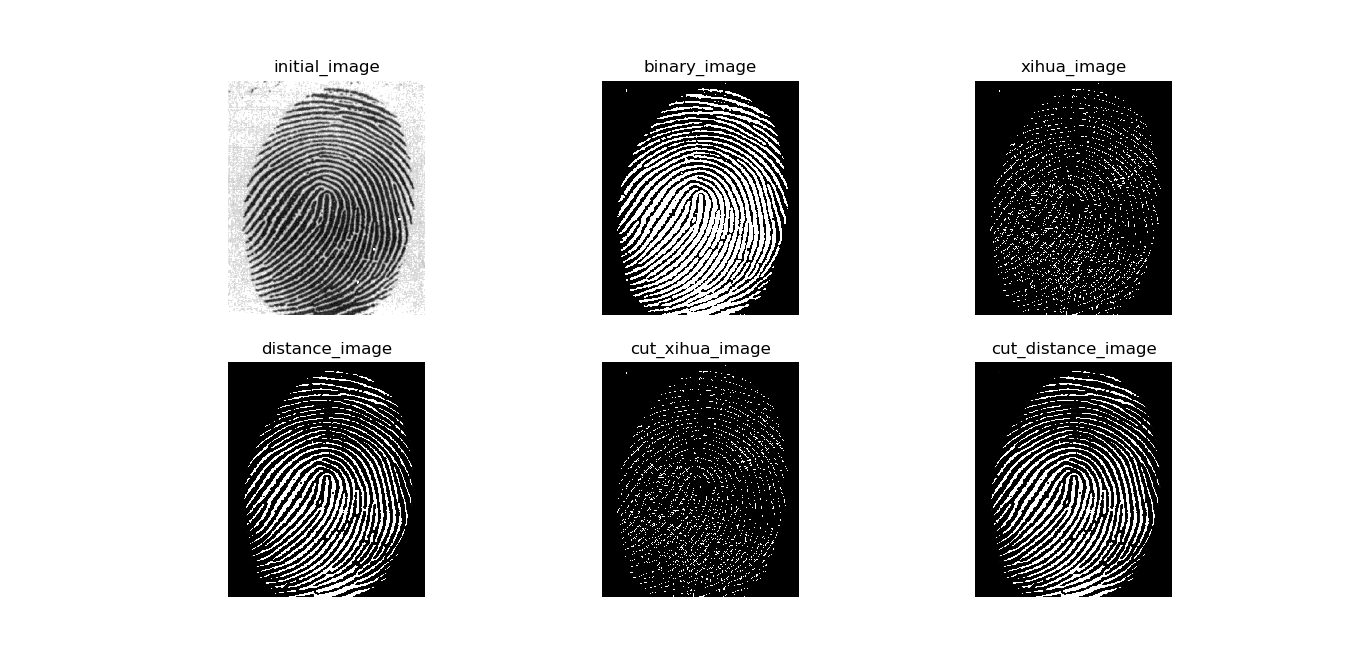
**结果：**



**问题描述：裁剪算法**

**代码思路：**1. 使用一系列检测端点的模式B对A做细化（这里重复做3次），【调用上面问题中的细化函数】得到细化结果X1：其中{B}一般取专门的结构元序列。2. 细化结果X1可能会丢失一些必要的端点，需要做一些补偿，首先用{B}分别对X1做击中，结果取或：然后以A为限定，对X2做条件膨胀，这里H取3\*3。3. 取X1和X3的并集就是最终结果

**结果：**

****

**总结：细化方法断点很明显，相比之下距离变换方法更好。**