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。

```
import ...

# 二值化
def _im2bw(f, thre):...
# 腐蚀操作
def corro(h, k):...
# 膨胀操作
def swell(h,k):...
# 击中击不中
def jizhong(FF, a, b):...
# 细化
def xihua(h):...
#边界提取
def border(f):...
# 距离变换
def distance(ff, FF):...
# 裁剪
```

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

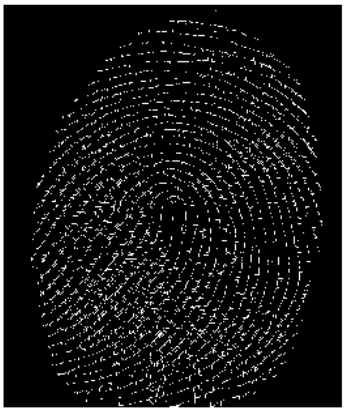
代码思路：细化图像。f_xihua = xihua(F)

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。

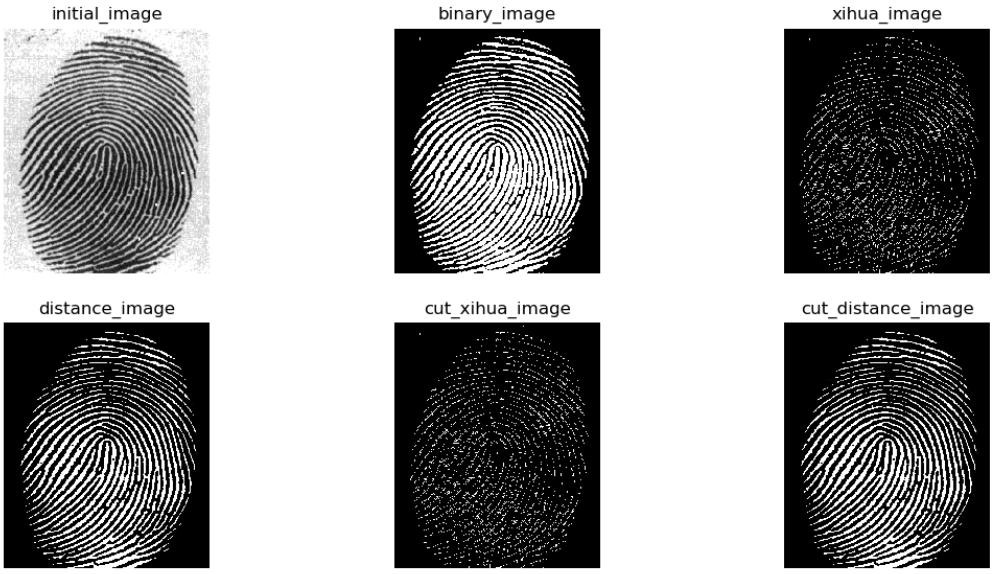


问题描述：距离变换骨架提取

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。