综合作业 - 指纹脊线增强

blahblah

2018.11.15

摘要

本次作业实现了指纹脊线增强算法，并在三张情况迥异的指纹图片上进行了测试。算法的大体思路和论文大体相近，其中指纹区域的分割方式及分块频率的计算方式按作业要求，和论文有所不同。此外，代码中对两张噪声相对较大的图片进行了一定的预处理，显著提高了最终的增强效果。

目录

1 运行说明

1.1 文件内容

`FTIR.m, latent.m, phone.m`：三张图片的处理程序

函数：

- localOrientation.m：获取分块方向图
- localFrequency.m：获取分块频率图

- extractFingerprint.m：分割指纹区域
- plotOrientation.m：在原图上绘制脊线方向和 Mask
- spatialGabor：Gabor 滤波
- Bfilter.m: 巴特沃斯低通滤波器[1]
- normalization.m：图像归一化

1.2 运行方式

运行 `FTIR.m, latent.m, phone.m` 即可绘制相应图片的增强结果、方向图以及指纹区域分割结果。

其中每个程序中的参数 vthresh1，vthresh2，fthresh 可调，作用是调整指纹分割区域的大小。

2 算法思路

分步对程序进行说明。

2.1 图像分块

将原始图像分成许多 8*8 的图像块，对每个图像块计算 DFT。实际计算方式是以 8*8 的方块为中心，计算 32 * 32 图像块的 DFT，并将由此得到的方向和频率作为 8*8 块的方向和频率。指纹图像的像素通常变化不大（论文中提到常见尺寸为 500*500），因此以选择 32 作为尺寸是普适的。

2.2 根据幅度谱提取指纹区域

2.2.1 计算局部方向和频率

计算出每个图像块的 DFT，并用 fftshift 搬移到中心后，方向和频率也可以计算得到。方向的计算原理是：指纹图像的局部近似正弦波，在理想情况下频谱会有两个中心对称的峰，其法线方向对应空域中指纹脊线方向。

于是在 `localOrientation.m` 中，计算方向的具体步骤是先判断频谱上最大值的个数是否恰为 2 个；如果是，则计算其连线关于水平方向的夹角，否则取唯一的最大值，计算最大值点与图像块中点和水平方向的夹角。这样做相比直接取一个最大值进行计算会更加准确。以第一张图和第二张图的指纹区域为例，其中绝大多数点的频谱最大值个数都恰好为 2，由此可以得到十分精确的方向。

频率的计算和方向十分类似，大致思路是计算峰值点和中心的欧氏距离。

[1] https://blog.csdn.net/cjsh_123456/article/details/79342300

2.2.2 估计指纹区域

对应代码 `extractFingerprint.m`。

估计指纹区域的算法主要基于宽度优先搜索。具体使用方式是从图片中心区域开始（这里假定中心区域属于指纹区域，实际上三张图的中心都是典型的指纹），设置一定的条件对图片进行 BFS，最终得到一个连通的指纹区域。

宽度优先搜索在本次作业的场景下主要有以下优势：

1. **较好处理无关噪声。以第二张图为例，除了中心的主要指纹之外，图片中周边区域还有很多无关的纹理。如果简单的对一些参数设置阈值在全图遍历，难免会把这些远离指纹、和指纹很相似的噪声也囊括在指纹区域内。而 BFS 搜索时无法跨越横亘在指纹和噪声之间的空白区域。**
2. **区域灵活。按照前面的分块进行 blockwise 搜索，获得的区域和指纹本身的轮廓十分接近，避免各种手工剪裁。**
3. 速度快。整个搜索过程只会在指纹区域内进行，而全图总共块数只有几十乘几十，因此算法复杂度很小。

阈值参数的设计（后续操作都是针对块而不是像素）：

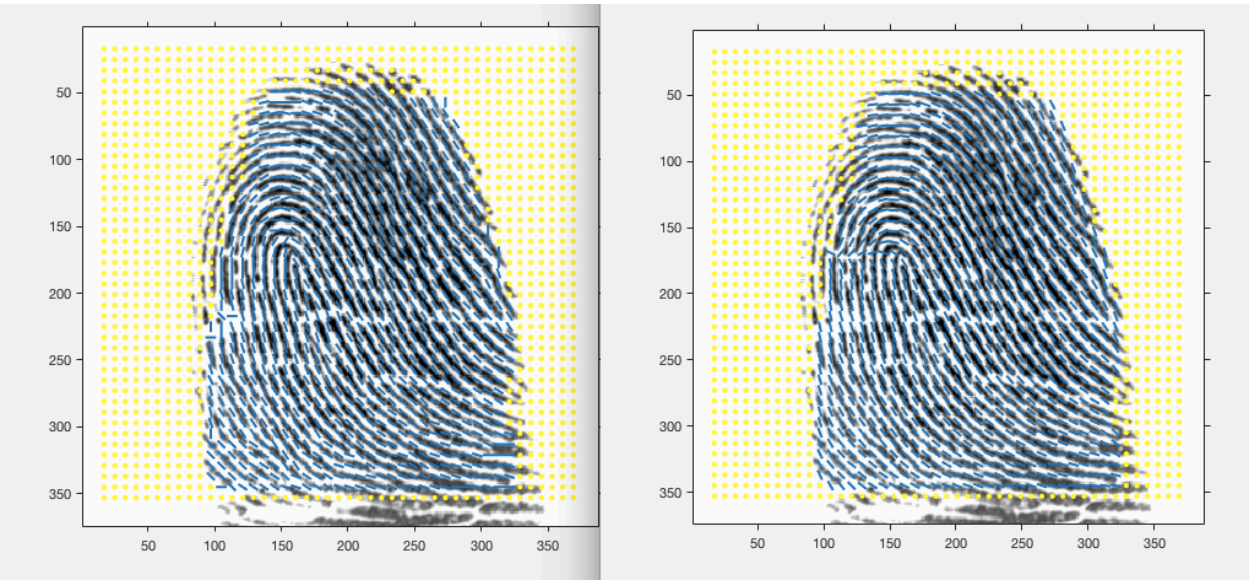
- 邻域（如以当前块为中心的 25 个块）中频率的方差（最关键的参数）
- 当前块频率与典型指纹块（图片中心）频率的相对误差
  _ $\frac{|regionFreq - sampleFreq|}{sampleFreq}$
- 邻域中“方向”的方差

选择第一个参数是因为指纹区域频率较一致，且和周围相差较大，用方差可以识别边缘。选择第二个参数是为了避免搜索到空白区域，因为频率方差同样很小而误判为指纹的情形。第二个参数通常设置为 0.4 - 0.6 即可，第三个参数只用于微调，因此需要调节的参数基本上只有第一个。

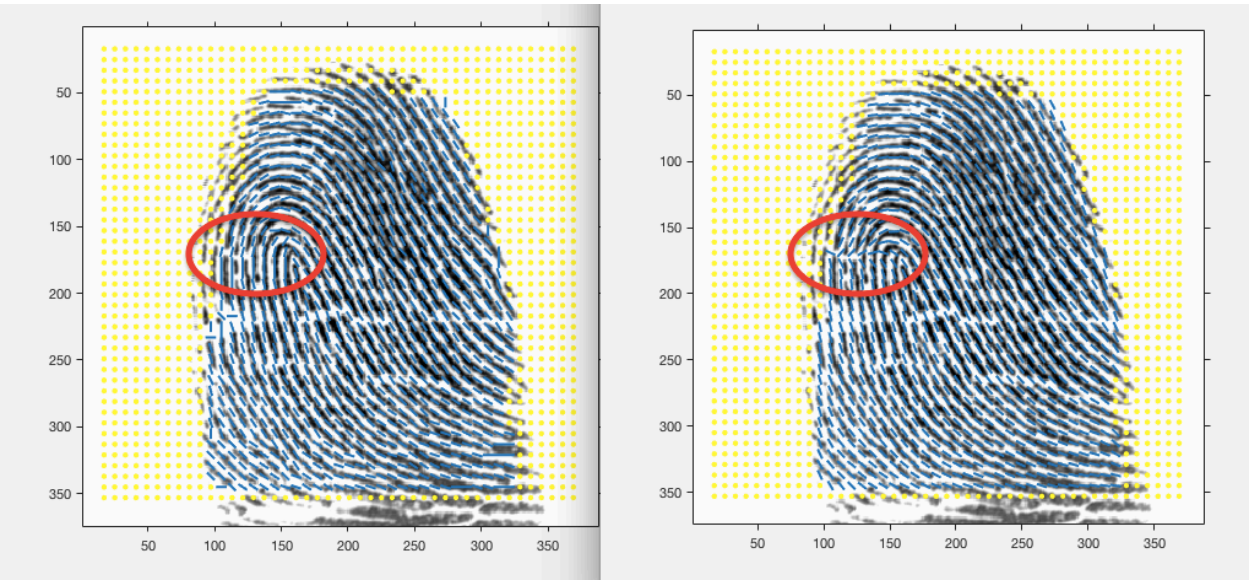
2.3 方向图和频率图的平滑

因为噪声干扰，需要用空域滤波对方向图、频率图分别进行平滑。按照参考文献的方法，都采用高斯低通滤波器进行滤波。

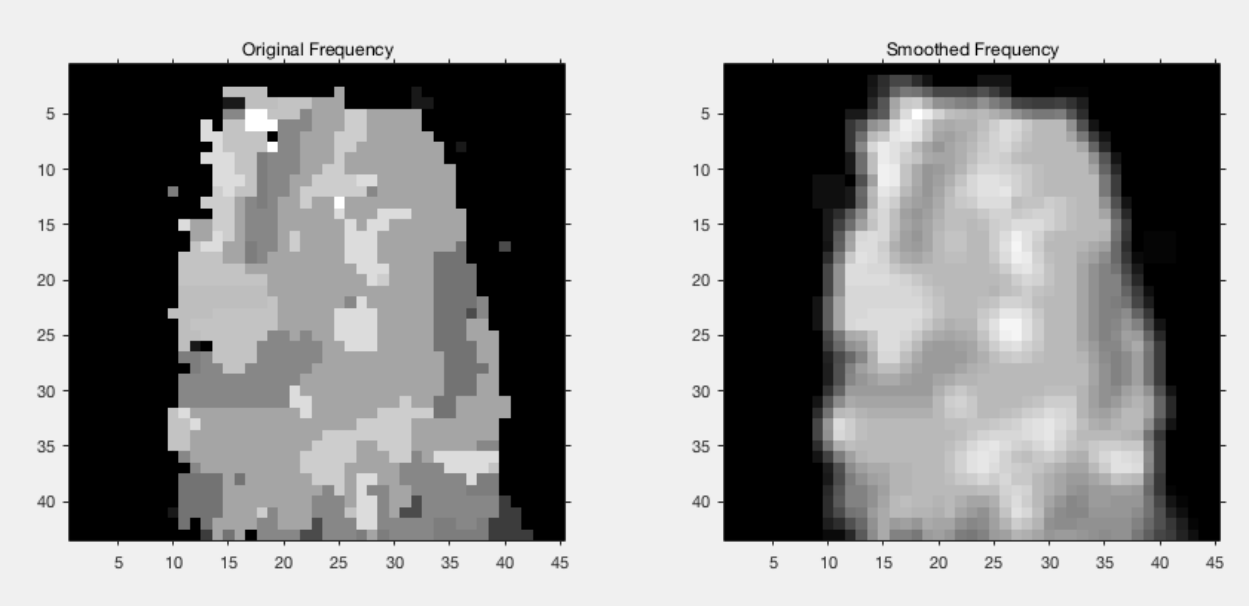
方向图平滑结果：



左右分别为原始和平滑后的方向图。可以看出，滤波后左下和右上区域方向的线条变得更加连贯。这是由于在这些区域纹理本身比较平行，使用滤波后引入邻域的信息，容易提升效果。然而，在图中这枚指纹中心偏左的区域，因为方向变化较快，在低通滤波后反而产生了不应有的脊。可见滤波也有一定代价。

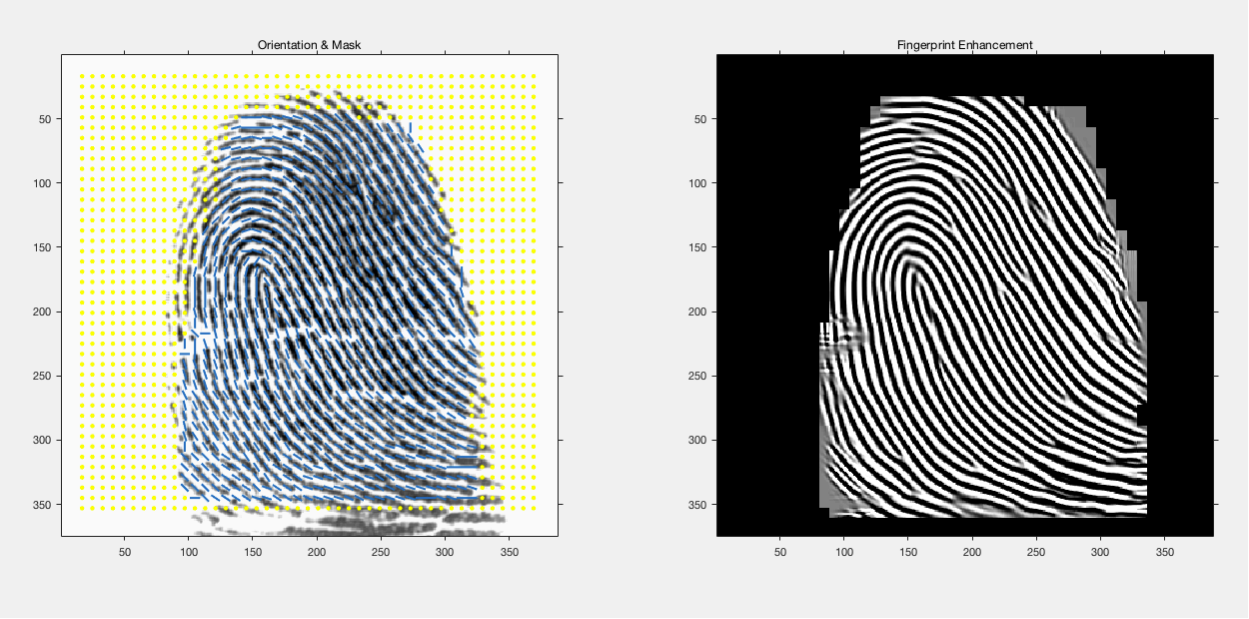


频率图平滑结果：



2.4 Gabor 滤波

基于方向和频率信息对原图进行滤波。使用工具箱中的 `imgaborfilt` 实现，主要参数为波长（频率倒数）、波方向（和脊线垂直）。



值得一提的是结果的线条看起来比较光滑。其原因是程序中滤波是对每个 32*32 的大图像块进行，这样一来在图像的绝大多数区域都会处于大图像块的交集。也就是说，这些交集的区域有着多个大图像块滤波得到的不同结果，那么对交叉的次数取其均值就有了最终纵享丝滑的结果。

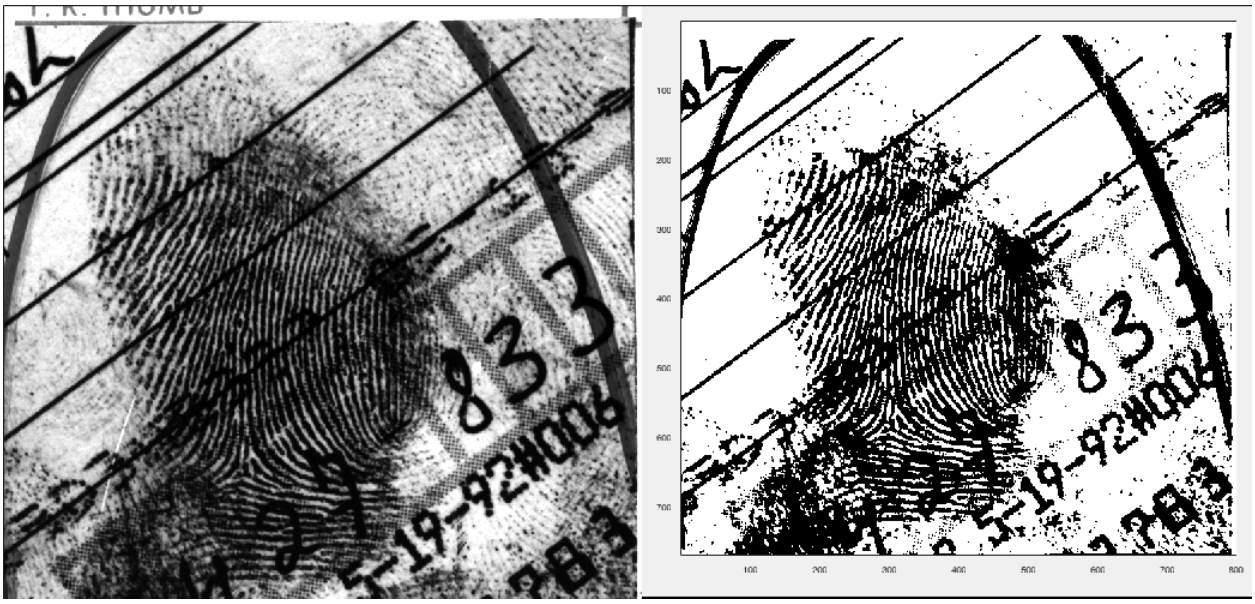
3 预处理的几种方法

3.1 latent.bmp：局部阈值

这张图虽然噪声无处不在，但指纹区域的平均灰度还是肉眼可见的高于其他地方。因此进行如下预处理：

- 直方图均衡
- 将原图分成 16*16 块（不交叉）
- 求每块灰度中位数
- 用中位数作为阈值进行二值化

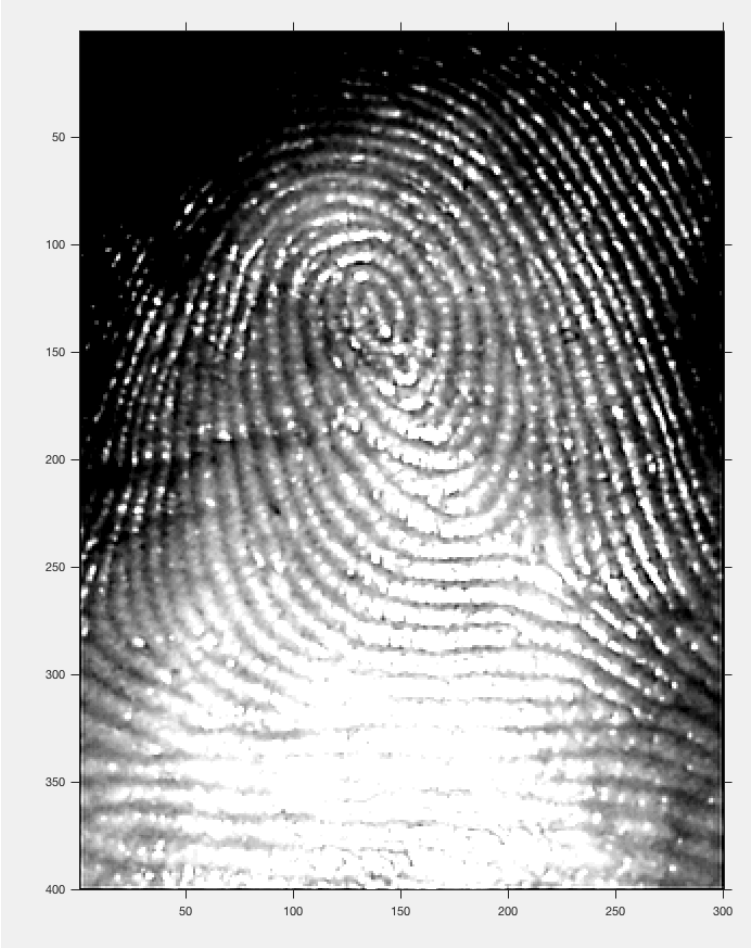
得到以下结果：



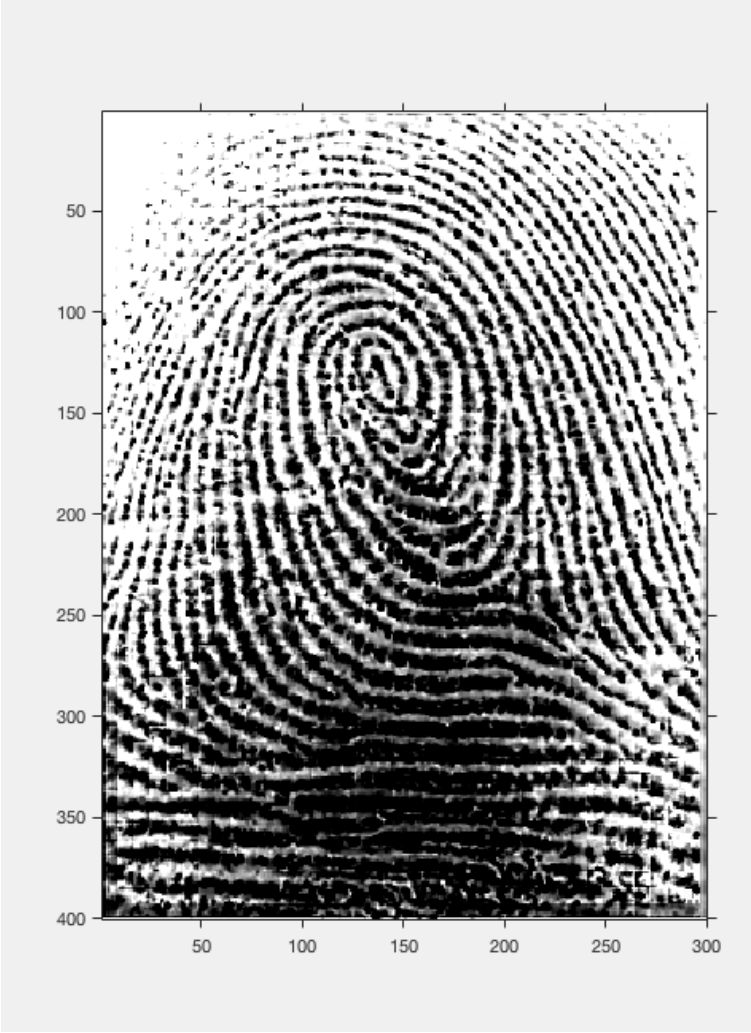
接下来用这个结果来求取方向图，最后再应用到原图上，相比比直接在原图上求方向减少了很多干扰。

3.2 phone.bmp

首先 normalize：



下一步，类似上一张图的预处理，将对局部进行 normalize，以局部的中位数作为均值：

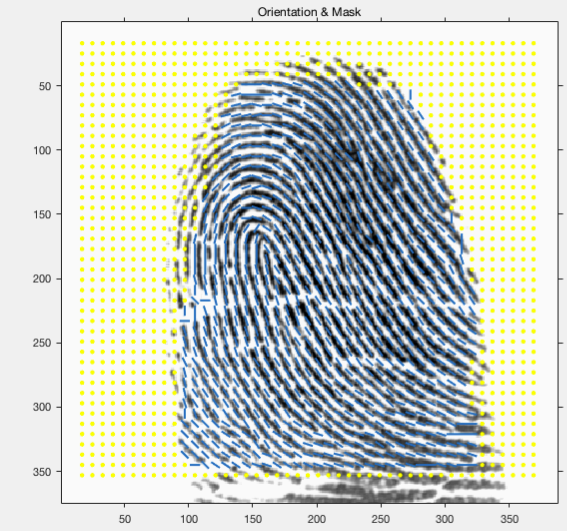


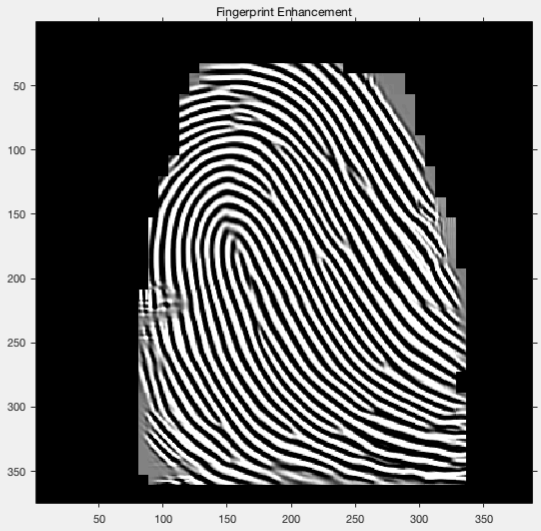
这时的结果除了不少断断续续处，实际上指纹已经清晰可辨了。

4 效果截图

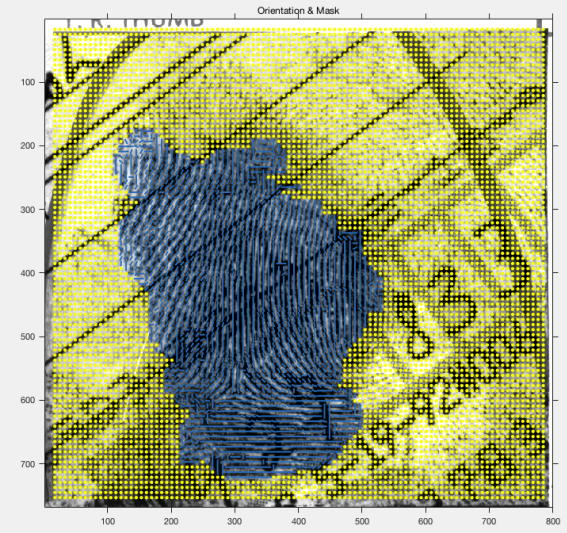
结果图附在上传的压缩包中。

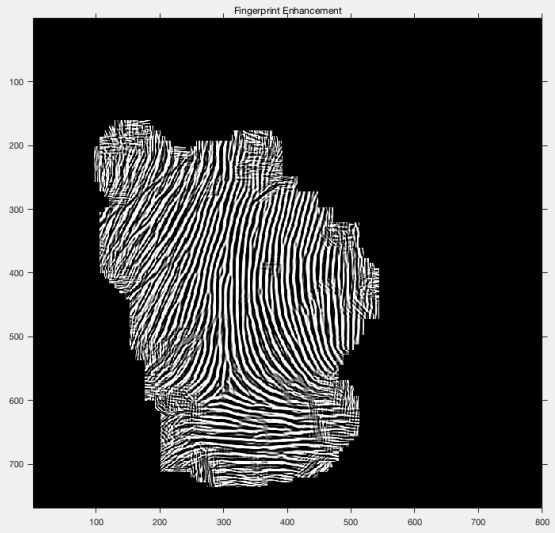
- FTIR.bmp



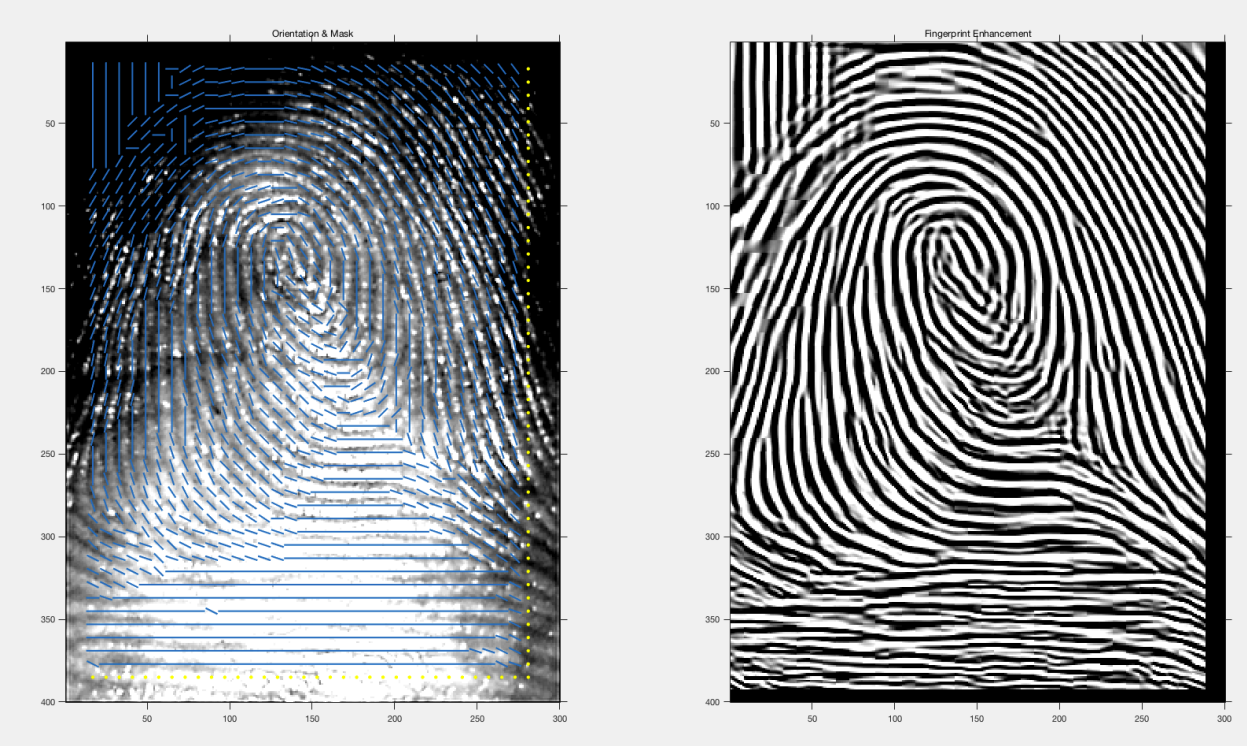


- latent.bmp





- phone.bmp



5 参考文献

Butterworth filter in matlab: https://blog.csdn.net/cjsh_123456/article/details/79342300.

Lin Hong, Yifei Wan, and Anil K. Jain. “Fingerprint image enhancement: Algorithm and performance evaluation.” IEEE Transactions on Pattern Analysis and Machine Intelligence 20.8 (1998): 777-789.