现场指纹识别项目报告

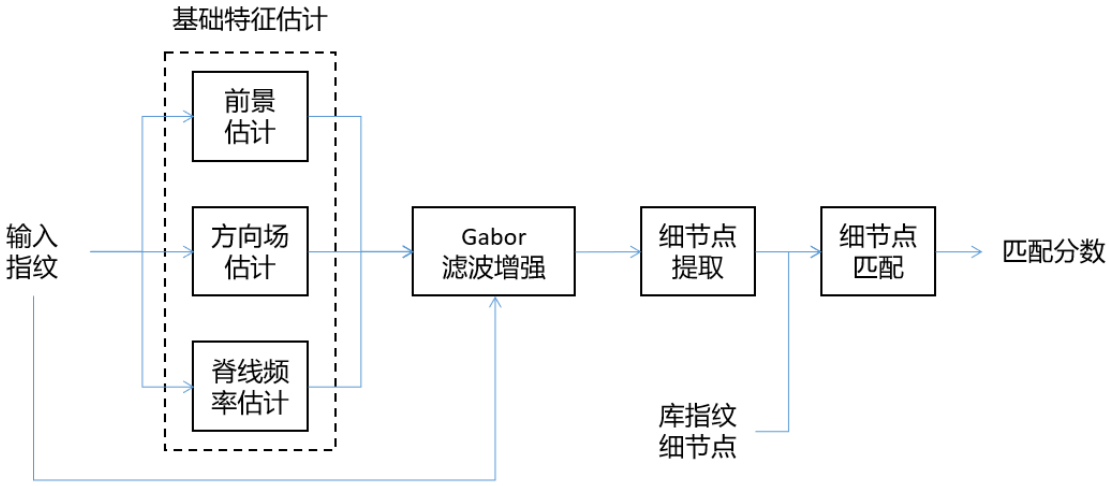
刘赫 自 16 2021010417

一、项目介绍

本项目主要借助前面作业所实现的各种功能，例如指纹图像增强，特征点提取，细节点比对等功能，来实现对于现场指纹与库指纹之间的比对任务。项目对于之前作业中的增强算法进行了重建，基于 Anil K. Jain 的文章对于现场指纹以及库指纹进行了增强，此外，针对此次项目所处理指纹图片复杂性对于特征点提取算法进行了优化。最终建立了一个 Matlab App 来实现对于现场指纹图片处理，以及查找匹配指纹的 GUI。项目在对于库指纹的特征点提取上取得了很好效果，对于现场指纹的处理效果较差。

二、算法介绍

本次算法的整体思路基于下图所示流程图：



下面分指纹增强、特征点提取和细节点匹配三部分介绍具体算法：

1.指纹增强

指纹增强主要参考了 Anil K. Jain 的文章 "Fingerprint image enhancement: Algorithm and performance evaluation." 整体流程为输入图像，归一化，生成方向场，生成频率场，计算指纹前景掩膜，对于指纹原图像进行滤波，输出增强后的指纹图像。部分代码由于在参考论文进行复现的过程中实现效果不好，使用了西澳大学 Peter Kovesi 的代码，或在其函数进行了修改。

1.归一化

将输入的指纹图像的灰度图像的灰度分布，调整为均值为 0，方差为 1 的分布，该调整过程中没有损失原有图像灰度分布的信息，但调整到理想的分布，可以更好地对于图像的处理，例如通过局部小块内标准差计算是否为指纹前景。

2.生成方向场

所用到的参数有归一化图像，用来计算图像梯度的高斯微分函数的标准值，用于 sum 梯度量的高斯权重的标准值，最终对方向场进行高斯滤波的标准值。

生成方向场的过程，首先计算图像的梯度，然后计算图像在 x，y 两个方向的梯度，通过找到图像梯度变化的主轴来判断局部的脊线方向，然后计算图像的协方差数据，然后对协方差数据进行平滑来对于数据进行一个带权的加和，然后进而求出每个像素的方向数据，然后用高斯滤波对于正弦值和余弦值进行平滑，最终得出平滑后的方向场。

3.生成频率场

利用到的参数有归一化的指纹图，通过对局部标准差进行分析得到的初步的掩膜，以及前面计算得到的方向场，以及分块的大小等。

生成频率场的过程中，首先对于指纹图像进行分块处理，然后选取以该小块为中心的大块，进行了对频率的计算。计算思路是，首先计算该大块中的平均方向，然后以该方向为行方向建立一个在大块内的窗口，对列的像素进行加和，即沿着脊线方向加和，通过每列像素和来判断是否存在脊线结构，如果存在计算频率值赋予小块，否则，置零。然后频率场和初步掩膜运算，得到掩膜后的频率场，然后计算频率场的中位值并输出。（中位值在尝试下用于 gabor 滤波效果更好）

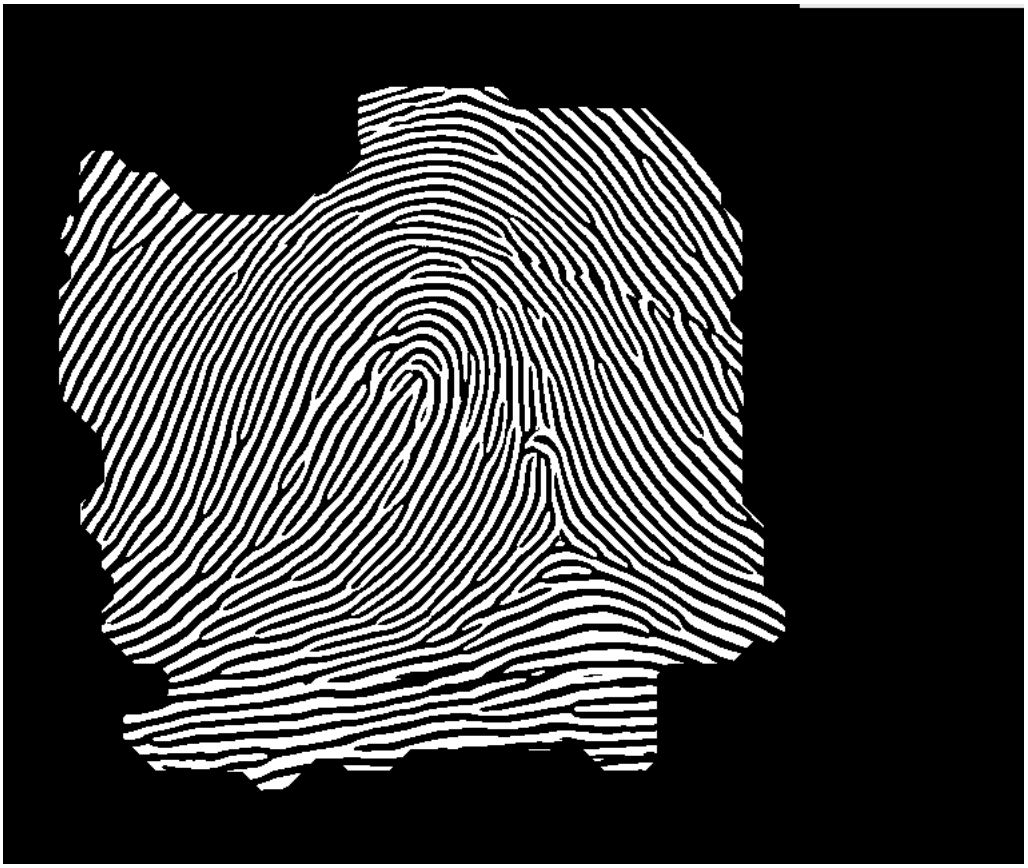
4．计算前景掩膜

前面有提到，初步掩膜基于归一化后局部的标准差值，标准差较大说明存在像素变化，即存在脊线结构，然后基于频率场计算结果，再频率计算过程中，窗口中波峰波谷像素差距过小，以及周期过大的局部进行筛除，得到二次掩膜，二次掩膜进行平滑以及去除小块得到最终掩膜。

5.进行 gabor 滤波

Gabor 滤波利用前面计算得到的方向场和掩膜后的频率场进行滤波处理，得到 gabor 滤波后的结果，由于图像为归一化处理过的图像，因此以 0 为阈值进行二值化后得到脊线图。

指纹强化结果如图（以库指纹 4 为例）:



可以看出呈现出很好的 gabor 滤波效果。

1.特征点提取

特征点提取的算法，利用前面输出的滤波后的二值脊线图，进行处理，提取特征点的坐标和方向等信息，并按照一定的格式输出为'.txt'的文件，用于后续的细节点匹配。

提取特征点主要包括以下过程：

1. 对于脊线图像，用 2x2 的方块进行开闭运算，对于脊线进行平滑处理，同时注意不要造成相邻脊线间的干扰。
2. 利用 bwmorph 函数进行细化，细化后用 3x3 的 square 结构进行膨胀，来去除小的断裂，后再细化。
3. 利用 bwareaopen 函数去除小的短线，利用编写的 Pruning 剪枝算法，去除小的毛刺，对于指纹图像的复杂带来的复杂的桥接，目前没有想到较好的处理方法，选择利用细节点的筛除来处理因为处理不当，带来的无效细节点，由于无效细节点多出现堆积的情况，故筛除距离较近的重复细节点。
4. 上述处理后的图像进行检测，然后对于细节点进行筛选，然后利用 mask 取反膨胀，来筛除边缘细节点，保留的细节点进行细节点方向的计算，然后将最终保留的细节点的坐标、方向等以“.txt”文件保持在相关 Icn 文件夹中。

提取除的特征点在原指纹图的可视化后结果如图（以库中指纹 4 为例）：

可以看出有效特征点率很高，对于库指纹的特征提取效果很好。

3.细节点匹配

细节点匹配算法，利用两对图片的特征点 txt 文件进行配对，通过令其中一对特征点遍历刚体变换，计算每次遍历匹配点对数，通过计算最大达到的匹配点数，从而得到最佳的特征点匹配结果。

其中每次遍历下的匹配情况的查看，设置在坐标和角度的误差允许范围，若某两个图的两个点的坐标差以及角度差在误差允许的范围内，则表明这两个点在误差允许下是匹配的。

有关于匹配分数的设定，对于现场指纹和库中指纹的匹配中，将匹配分数设置为最佳匹配状态下的匹配点对数与现场指纹的特征点值的比值乘以 100。

三、现场指纹识别功能 GUI 介绍

（下按使用顺序介绍）

1. 载入现场指纹按钮：点击可以从用户电脑读取指纹文件（有一定的文件命名要求和相关 mask 文件要求），同时在按钮上的坐标区中显示指纹图像
2. 特征点提取按钮：点击后会提取在 1 中载入指纹的特征点，会有 msgbox 提示提取完成，并且自动生成相关 txt 文件置于 IcnSearch 文件夹中；
3. 显示含特征点指纹图片按钮：会生成 figure，显示基于上述载入的指纹和提取的特征点的可视化的结果；
4. 匹配库指纹按钮：需在 1.2 按钮实现后点击，后将特征点与库指纹的指纹的特征点进行匹配，并按照匹配分数从高到低，将可能的匹配结果显示在右上角的排行列表中，匹配的过程中会有 msgbox 显示匹配的进度。
5. 查看匹配图按钮：点击前需要在查看排名后的文本框中输入希望查询的排名数，会生成 figure，显示具体的匹配情况的可视化结果，可以用于对于排名较高的结果是否真的合理匹配进行查看。
6. 查看真匹配情况按钮，可以查看该现场指纹对于的真实库指纹，以及具体的匹配情况的可视化，同时会有 msgbox 告知真匹配情况的匹配分数和匹配排行。
7. 退出按钮：从 GUI 中退出；

四、项目总结

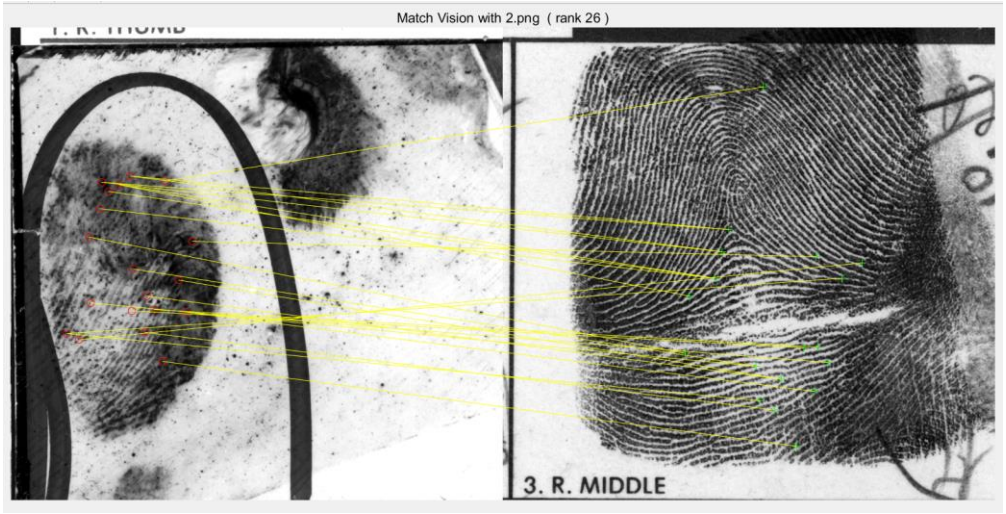
本人在项目中觉得做的不错的是，实现效果很好的 gabor 滤波结果，以及对于库指纹的特征点提取的正确率很高，但是这些处理代码，对于现场的指纹的处理效果很差，从而导致了现场的指纹的特征点错误率很高，从而带来了最终匹配结果的效果的不理想。

此外，本人为了提高的在与大量库指纹进行匹配的速度，降低了匹配算法的精度，但是最终的呈现是，一组现场指纹的特征点在与库匹配的时间也在 3 分钟左右。我觉得我的算法在时间的复杂度方面还有很大的优化的空间。

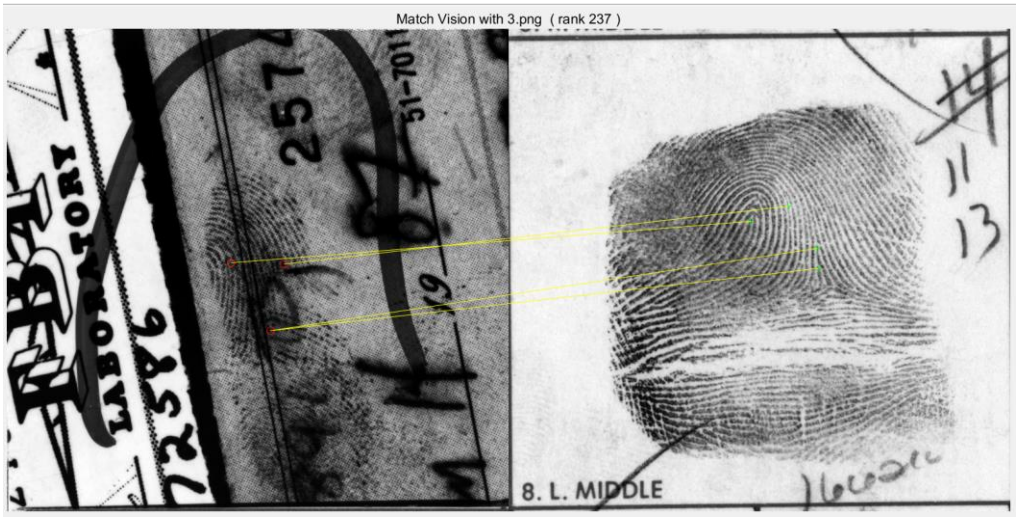
本次项目综合从指纹增强到细节点匹配的各个阶段，对于所学的课程的内容有了很好的总结，此外对于各种函数及代码的衔接，以及 GUI 的设计中，我都掌握了很多 Matlab 编程的技巧，最终也实现了一个具有较完备的功能的可以方便使用的指纹识别 GUI。

可视化结果及排名

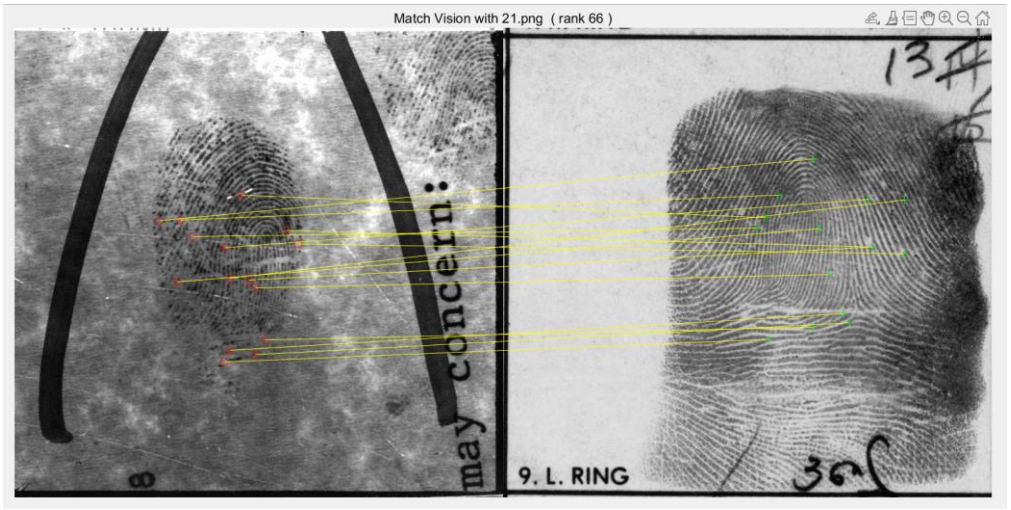
现场指纹 2 ， rank26



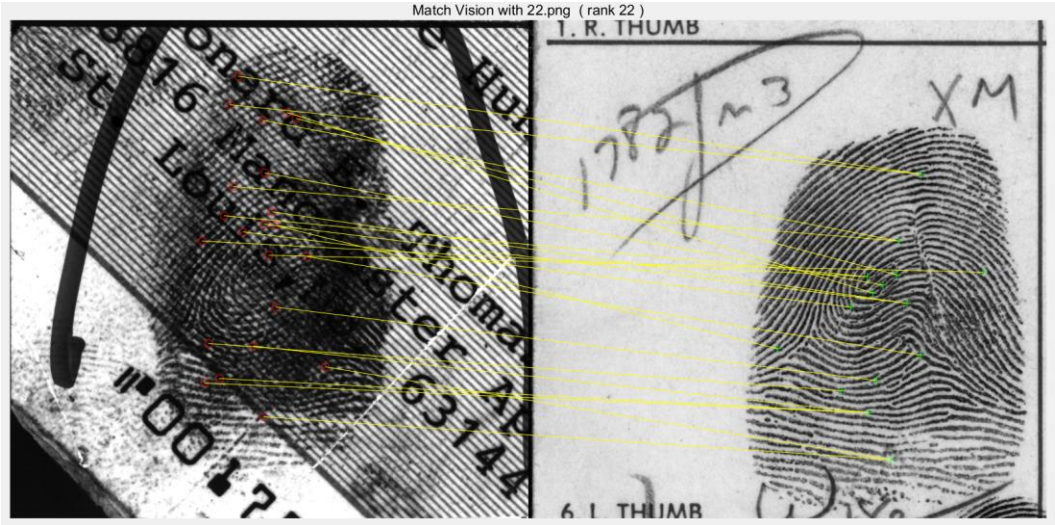
现场指纹 3 ， rank 237



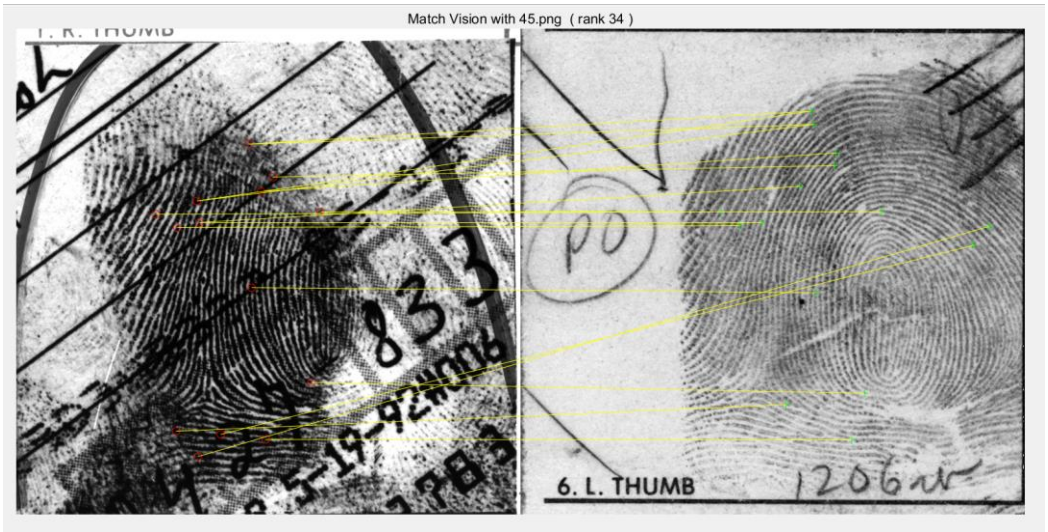
现场指纹 21，rank66



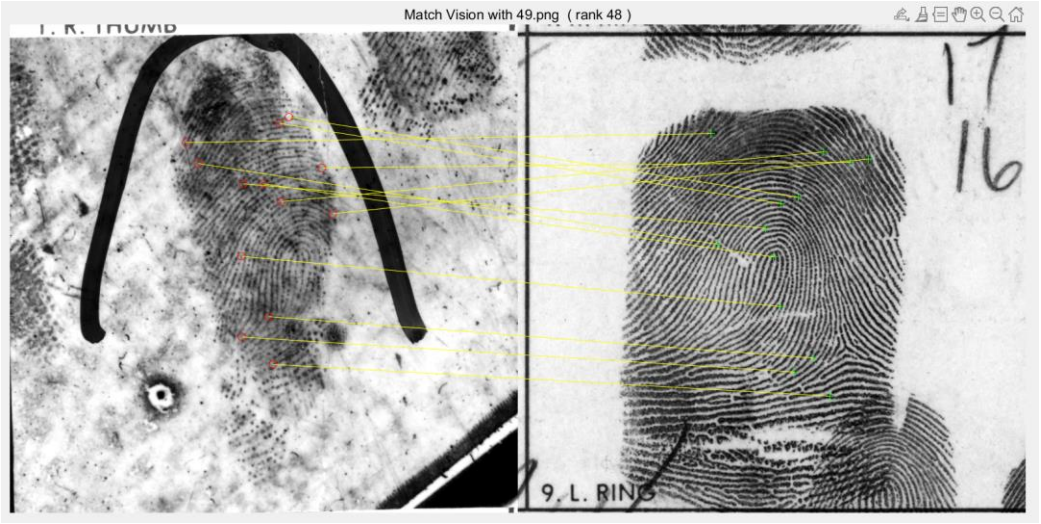
现场指纹 22，rank22



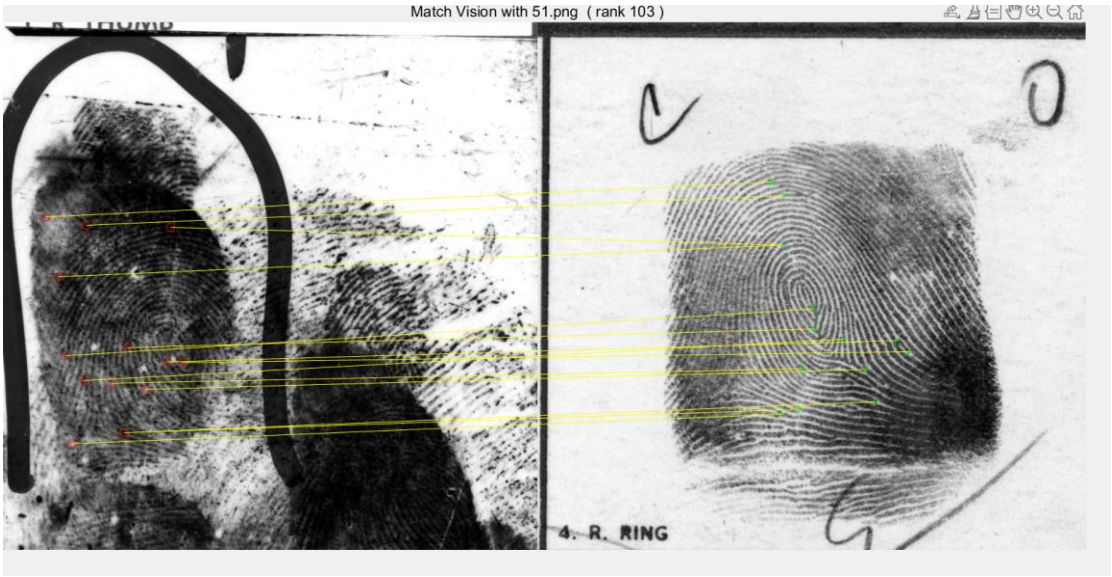
现场指纹 45，rank34



现场指纹 49，rank48



现场指纹 51，rank103



现场指纹 60，rank211