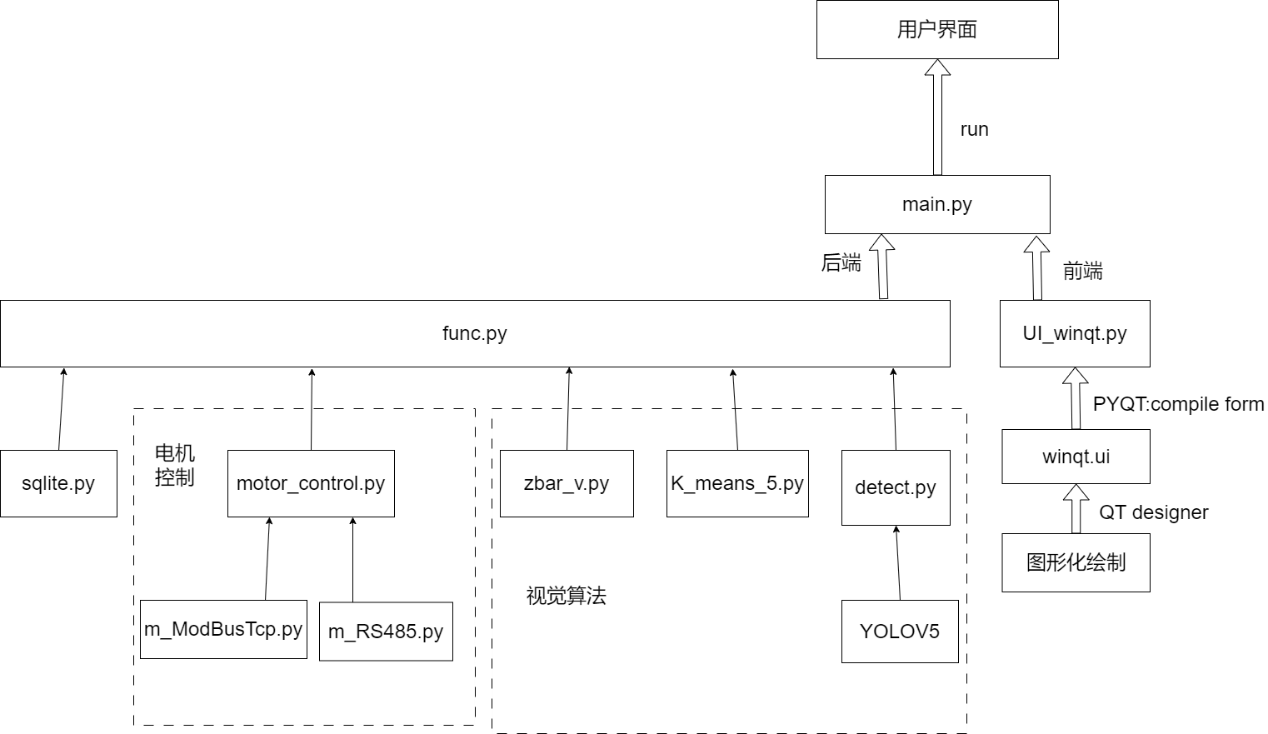
嵌入式技术文档

本项目嵌入式方面由电机控制、视觉算法、上位机三个模块组成。电机控制方面搭建以Jetson Nano为主控的Modbus通信主站，使用Modbus RTU和Modbus TCP两种通信协议实现通信，电机驱动器配置使用厂家提供的上位机设置。视觉算法由YOLOV5网络的采血管种类目标检测算法、基于pyzbar的条形码识别算法以及基于KNN的采血管血液分层识别算法组成。上位机使用PYQT5开发，内置数据库、多线程搭建框架。

整体的软件架构如下图，sqlite.py实现数据库，m\_ModBusTcp.py和m\_RS485.py实现电机控制的通信层，motor\_control.py实现电机控制的应用层。zbar\_v.py实现条形码识别算法，K\_means\_5.py实现采血管血液分层识别算法，YOLOV5实现采血管种类目标检测算法。上述集合成后端func.py。前端则由QT designer绘制的winqt.ui转化生成的UI\_winqt.py实现。最后运行main.py开启用户界面。每个文件的API接口都在对应文件中有介绍。



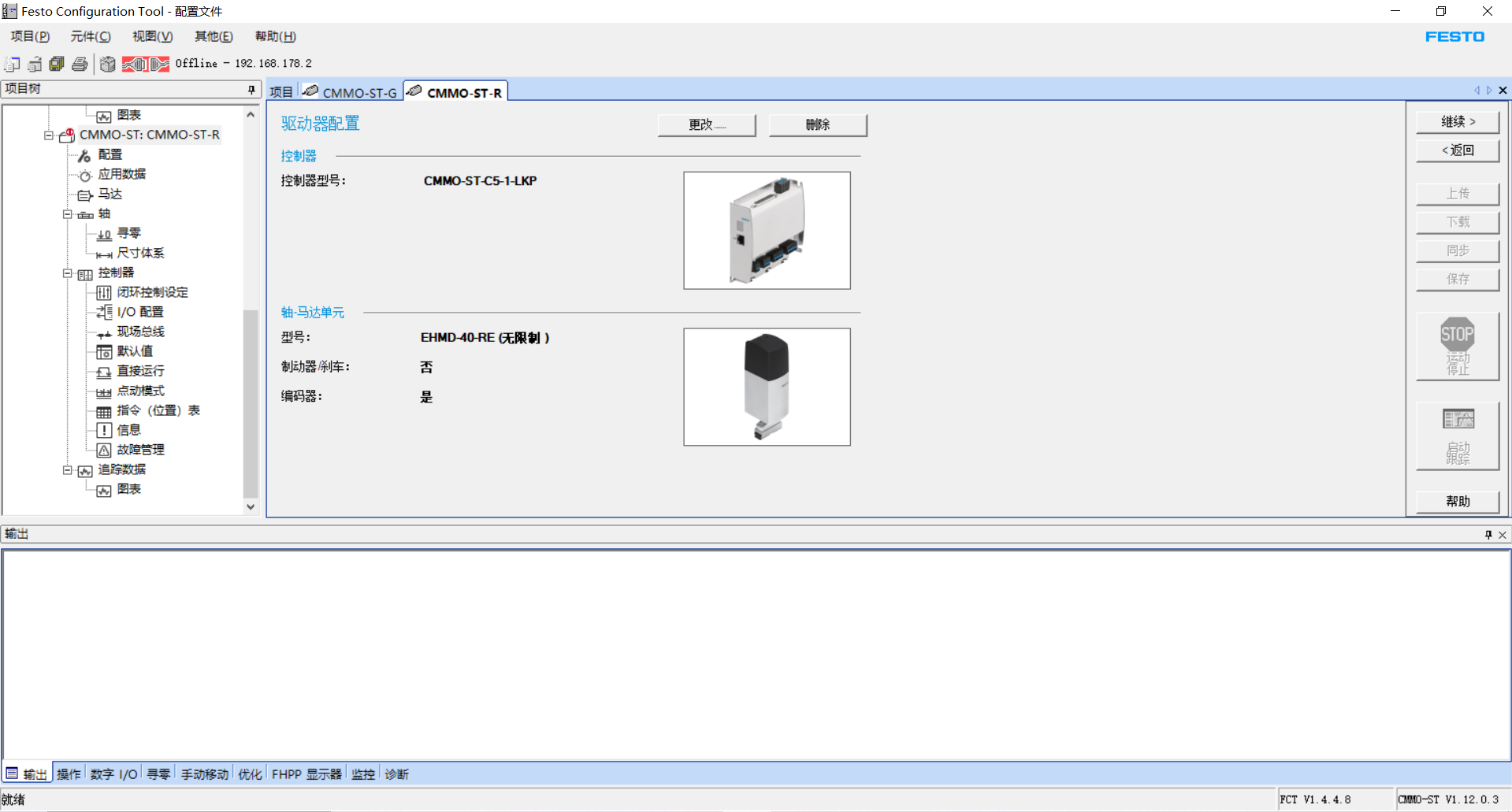
目前jetson nano上插的卡用户名是hxq，密码是123456.

**电机控制**

1. **电机驱动器配置**

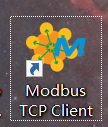
因为使用的电机都是驱控一体化的，所以有对应的上位机配置相应的参数，现在的电机是按照现有需要的设计调整的参数，假如需要更改就要使用上位机进行烧录文件，因此还是介绍一下。假如不打算修改参数或者指令集可以跳过这一小节。

1. 旋转夹持电机使用的是Modbus TCP协议，用以太网口连接交换机进行通信。打开软件，先安装软件包，然后打开项目，出现界面如下。

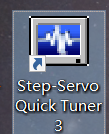
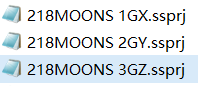


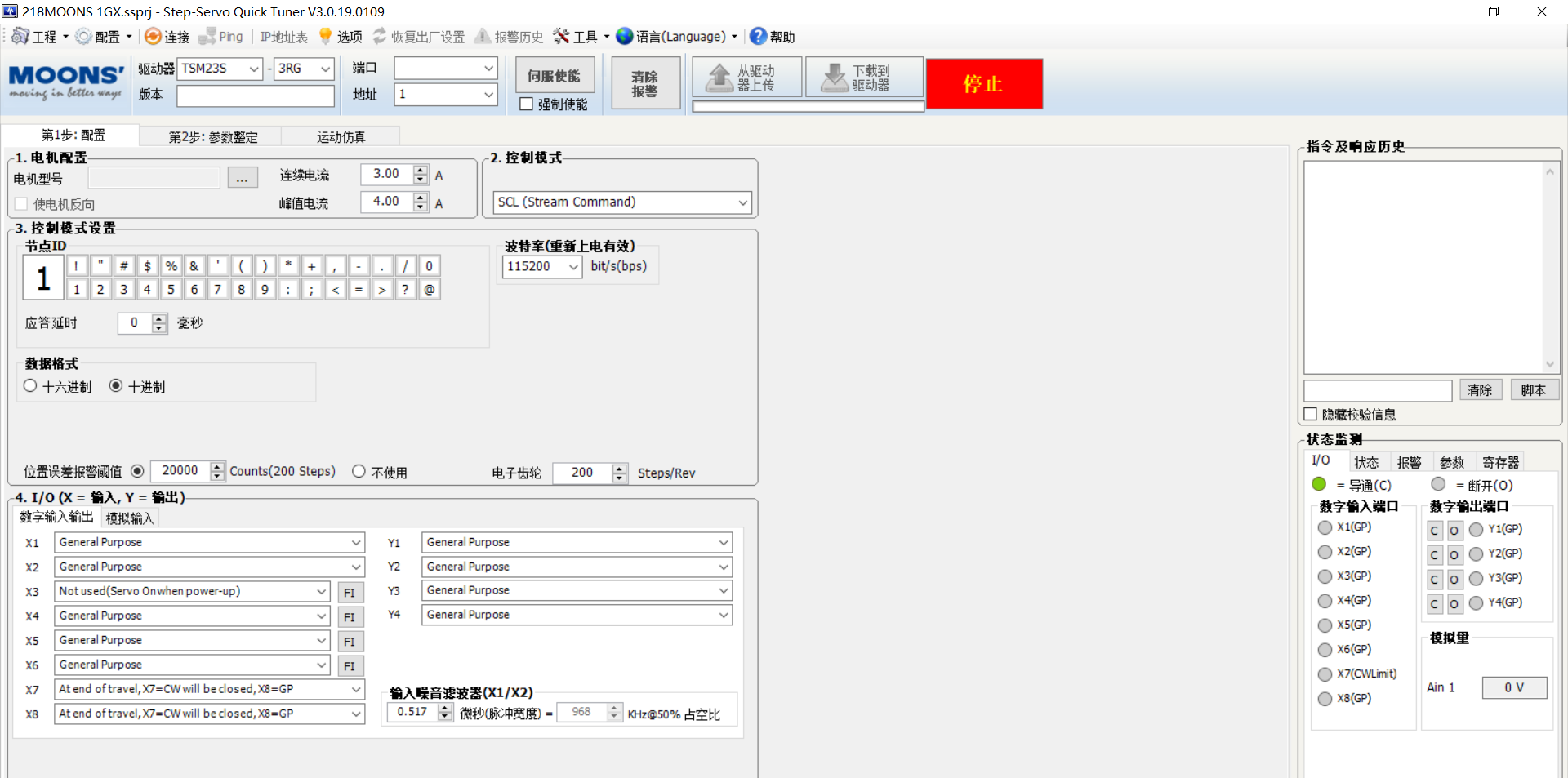
点击可以连接对应的驱动器烧录或者测试，其中G是夹持电机，R是旋转电机。对应的说明书是，里面有操作说明，有些看起来很奇怪的东西推荐看英文原文说明书

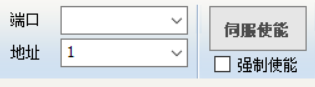
。

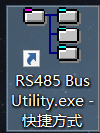
假如要一条一条测试指令是否正确，可以使用软件进行调节。按照说明书里面5.2小节操作就行。同时搭配软件抓包读取通信，分析发送和返回的指令。目前Modbus TCP我是使用指令选择模式控制的。

1. XYZ轴电机使用的是Modbus RTU协议，物理层是RS-422（RS-485）通信。

配置参数或者测试时用软件，打开项目就是对应的配置文件。软件界面如下，



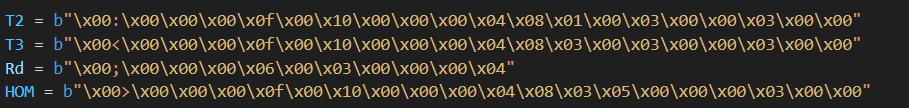
选择电脑对应端口可以使能电机驱动器，然后烧录或者下载配置。

是单独的驱动测试脚本，脚本语言为，按照说明书一条一条看即可，想了解通信过程的话可以看看，里面有较为详细的通信过程。或者直接借助m\_ModBusTcp.py和m\_RS485.py里面的控制逻辑分析哪些是直接用到的。然后假如想要同时测试XYZ三轴电机运行，使用软件可以做到。

1. **电机控制脚本**

目前在后端调用的电机控制的应用层是在motor\_control.py脚本里面实现的，从单个电机的控制函数如G\_inward\_4mm集合成detect\_once再到detect\_all，具体得到控制流程是结合多线程通过事件实现与拍照的通信。

在detect\_all中设置Y\_Number和X\_Number可以调整采集的排数，方便测试。

而motor\_control.py是通过m\_ModBusTcp.py和m\_RS485.py两个脚本完成5个电机的通信。目前一些指令我已经通过宏定义实现，如下所示。假如不需要更改底层通信的话，直接使用现有代码就行，旋转夹持电机的我已经分别写入指令1、2的通信函数，对应功能在motor\_control.py看解释就行。XYZ轴电机的话底层通信对应的MOVE函数就OK，结合说明书看看注释大概就能了解。需要调整参数的话最好在motor\_control.py里面修改对应函数的参数，方便理解和操作。

**视觉算法**

1. **条形码识别**

因为对每个样本录入数据库需要有唯一识别信息，目前选用的是条形码作为识别手段。在zbar\_v.py中目前使用的条形码识别算法是基于pyzbar库的函数，我们队输入照片进行灰度化和二值处理再解码基本就OK了，可以看看脚本里面的注释。

额外还可以进行多图的测试实验，包括二值化阈值得到的方法或者二值化的阈值等，统计后输出到Excel文件中或者直接统计，脚本里面都有相应的代码。

1. **采血管种类目标检测算法**

如图所示，对于样本管是否有盖和样品管类型需要识别，分类任务还挺简单的，目前选用yolov5网络。



由于jetson nano b01的系统是Ubuntu18.04，内置Python版本为3.6.9，因此选用yolov5-ultralytics-HUB这个比较老的版本做适配。YOLOV5网络相关介绍可以直接查看资料，网上很多，这里就不展开说了。

需要说说的是目前主要就是做了一下采血管本身、盖、有无凝胶块的标签，通过对数据集标注训练。

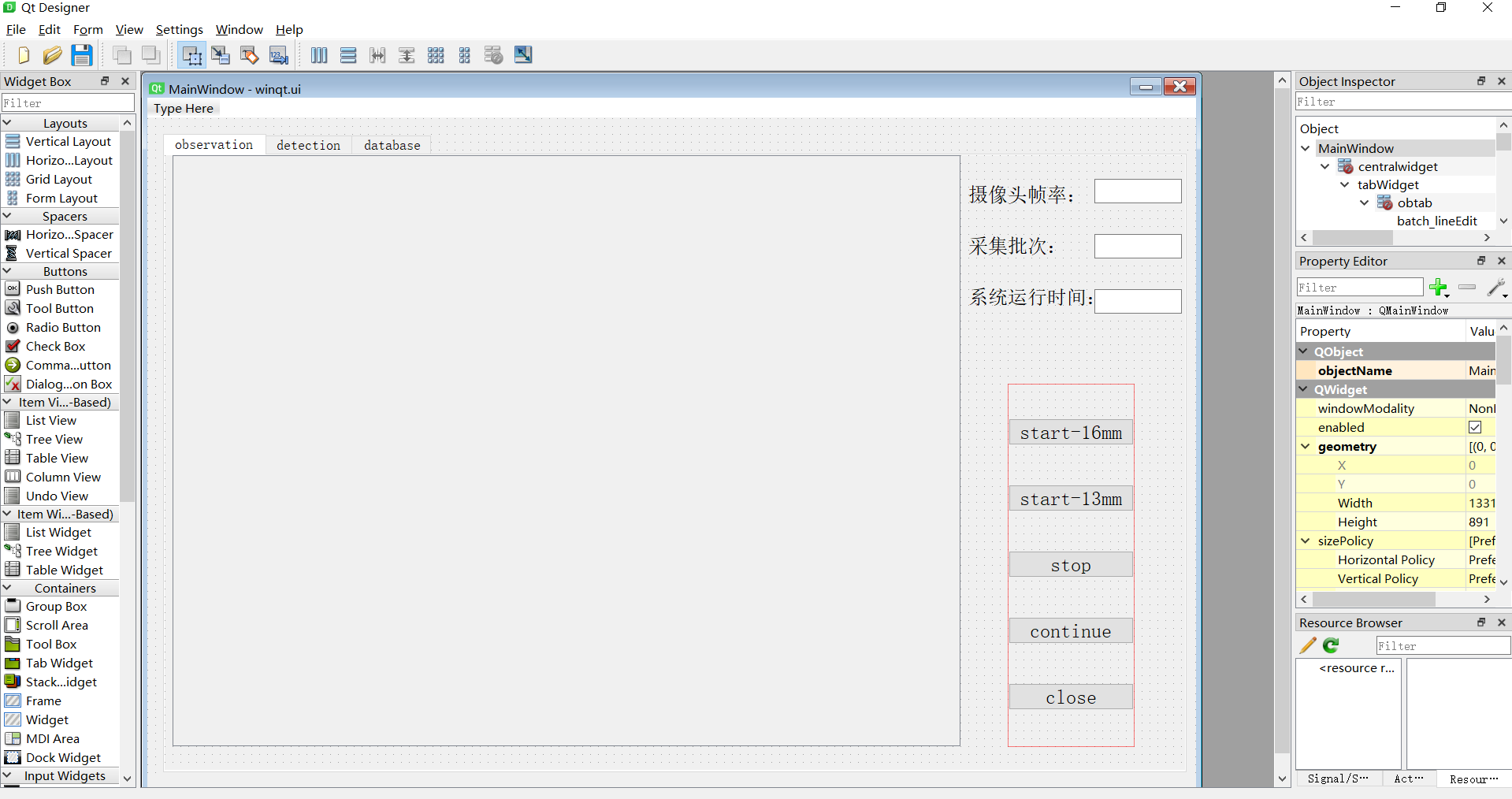
1. **采血管血液分层识别算法**

采血管血液分层识别算法是项目算法的核心内容，算法方面是另一个同学负责的，我这边主要是根据他的脚本封装出了一个test函数使用，直接对图片调用K\_means\_5.py里面的test函数就可以得到结果，详细信息查看脚本注释。

**上位机**

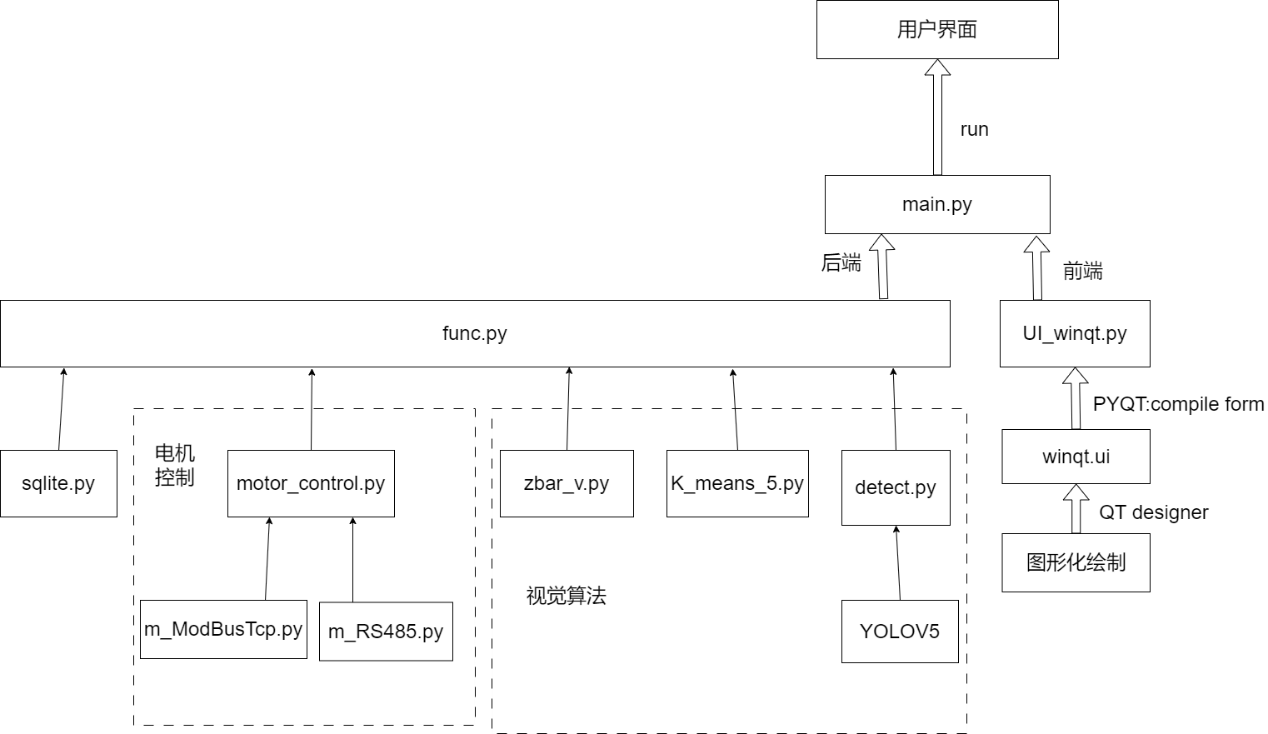
1. **前端界面绘制**

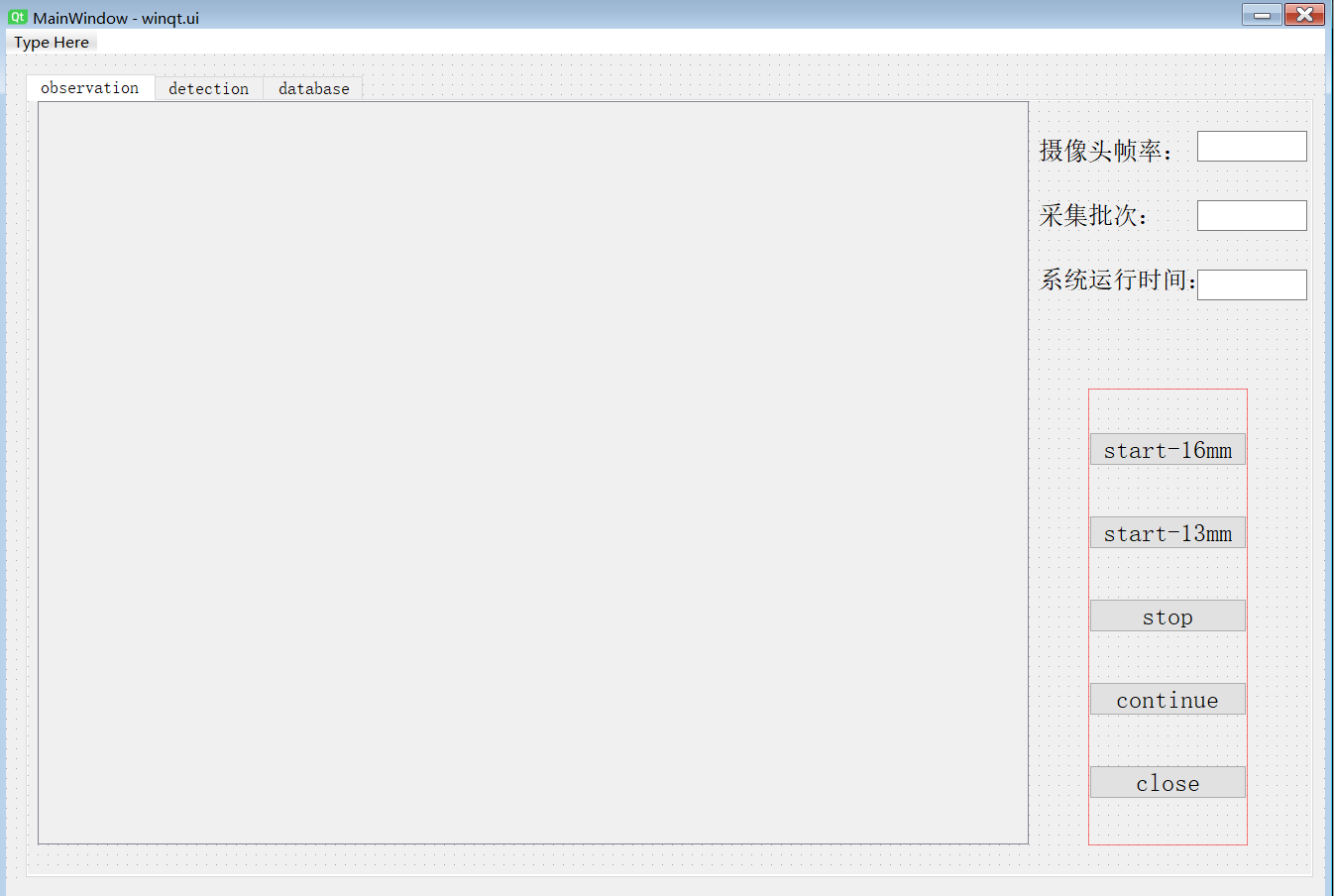
两种方案，一种是使用PYQT5中的QT designer进行图形化的绘制，如下图所示，绘制过程可以上网搜教程。绘制完得到winqt.ui然后下载插件PYQT的插件使用对应功能生成UI\_winqt.py得到编程化文本，也就是前端界面，后端程序就在此之上编写。另一种是全程使用编程语言设计界面，不使用图形化编程，哪种顺手用哪种即可，我个人是使用第一种方案的。



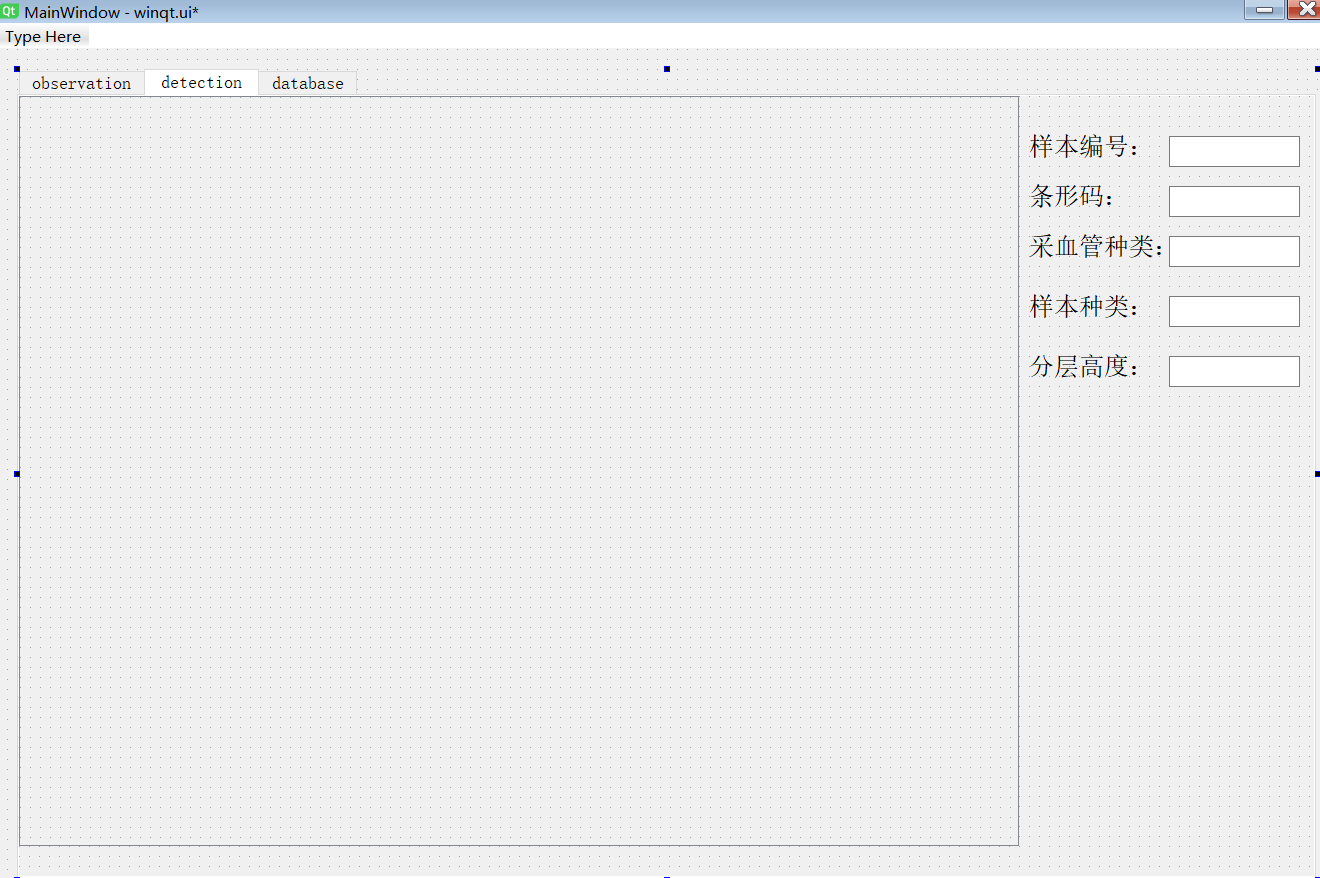
1. **后端系统设计**

后端文件是func.py，在此内集成了所有子模块的文件，也就是实现前端按钮对应功能的文件。下面我先介绍一下操作流程。

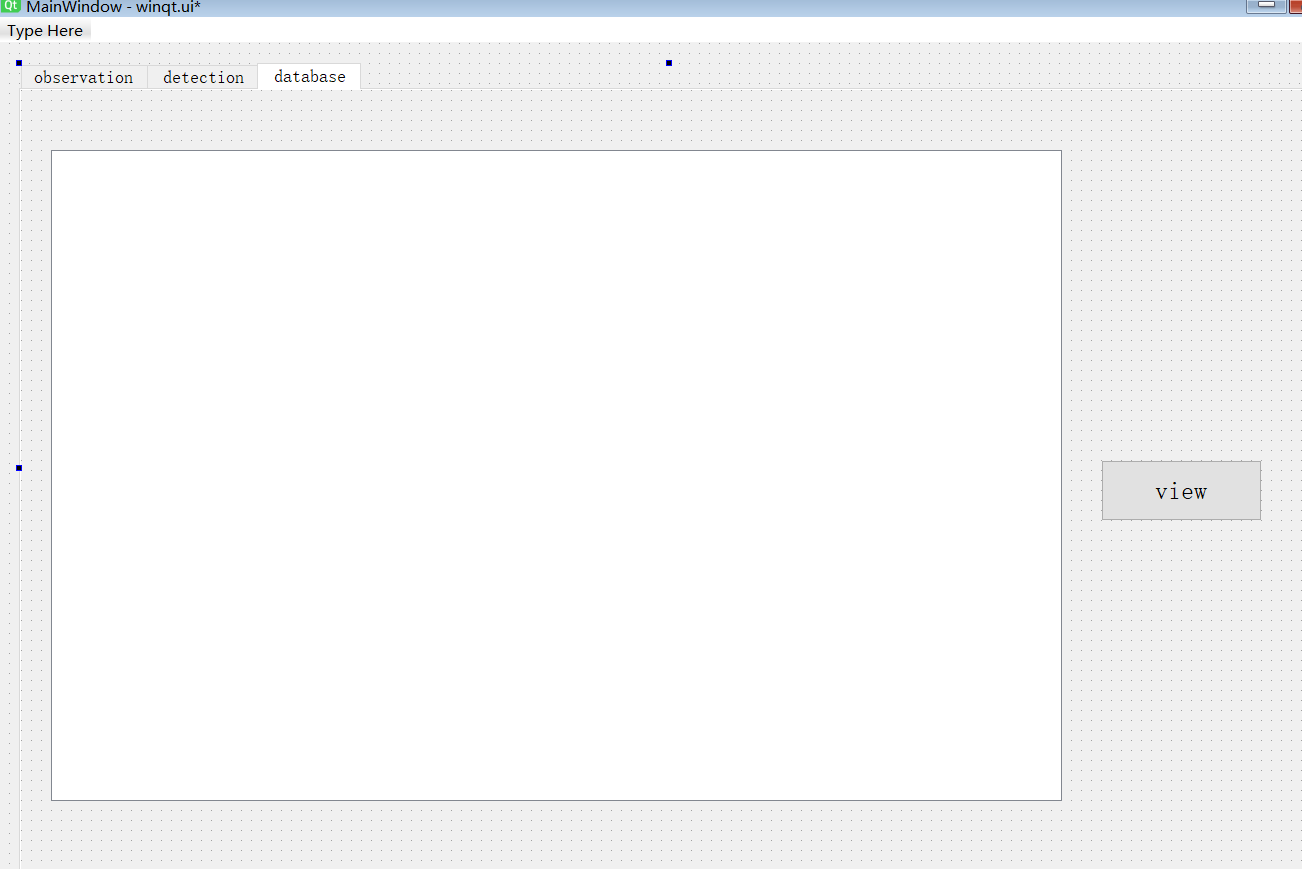




运行main.py后打开系统界面，开启后自启动摄像头初始化和电机初始化线程，进行系统的初始化工作。上面是系统主界面，点击start后开始一个检测流程。中间可以点击stop暂停，continue继续。一个检测流程做完后，可以放上新的采血管架点击continue继续下一个检测流程，最后可以点击close关闭系统。



Detection界面主要是观察每一个样本检测过程的信息，结果在右上角。



Database界面目前是完成了一个数据库信息展示功能，点击view按钮即可。

所有的后端逻辑通过QT designer中的槽函数可以得到函数映射逻辑，相关操作都是标准的pyqt操作。

**优化点**

1. **采血管种类目标检测算法优化**

目前只做了采血管本身、盖、有无凝胶块的标签，在需求的2.6小节中，还需要很多其他样本管的识别，而在医院中，不同样本管对应的盖是不一样的，因此可以通过识别盖的种类来识别样本管。

因此后续可以再采集其他的数据集加入训练集中，通过标注训练即可，我使用的标注网站是[Make Sense](https://www.makesense.ai/)，使用说明看网站说明即可。

1. **条形码识别算法优化**

目前的条形码识别需要提前知晓这次的样本集中的条形码是什么类型的，或者说需要知道是多少位数的，然后根据位数检测识别结果是否正确，不正确就增大二值化阈值直到正确。因为这块的识别结果肯定要做到接近100%正确率的，还有要缩短识别时间，因此需要进行优化。

后续有三条优化路径：一是自己根据每种条形码的编码规则做正确性校验的模块，在基于pyzbar库识别出条形码后，检验结果的正确性。

二是阅读一下zbar的源码，看看内部识别不同条形码模式的方式，仿照其识别模式校验一下结果的正确性。

三是从硬件上优化，调整光照条件或者图片质量，增大识别结果的正确率。

1. **图片拼接**

目前拍摄采血管是前后分别拍摄两次并对四张照片都进行视觉识别，有两个不足。一是旋转180°导致侧边的有效信息很少，识别不完全，当识别窗口只在侧面时，识别结果不好。二是识别效率差，目前就要识别4次，如果为了保证侧面的有效性，很可能得识别6次及以上。因此可以通过拍摄多张不同角度的图片然后把图片拼接的方式优化掉上述问题。

图片拼接有两种方法。一是由于拍摄时采血管位置是电机确定的，我们可以把位置固定，因此采血管在图片中的位置也就固定了，然后只裁剪需要的部分进行多张的拼接。

二是利用全景图片的算法，将多张图片融合在一起。上述方案各有利弊，推荐都做一下实验看看效果。

1. **电机加速**

目前采集流程中一支采血管平均需要27秒，时间有点长，而一支采血管的主要用时分为电机移动和视觉识别两块，因此可以通过电机提速来提高检测效率。

但是由于目前的采血管架设计不是很稳定，导致电机提速后导致平台晃动，然后采血管会倾倒，常常会发生干涉现象。

有两个优化方案，一是直接减少晃动的可能性，通过更加采血管固定的设计方案使得晃动现象减少，需要更新机械设计。

二是通过传感器检测平台振动幅度，设计相关实验寻找合理的电机速度。

1. **视觉识别加速**

第四点说过视觉识别也是占用时间大块，其中有些时间花在模型加载上面。将视觉识别划分为加载模型-处理图像-图像识别-结果输出等过程，通过利用多线程中电机运行的空闲时间可以分别将上述过程在不同的空闲时间中进行，从而提高识别效率。

1. **升级硬件**

因为后续很有可能会使用多个YOLO进行图像识别，而目前使用的jetson nano b01在内存上不太够，最好是选用更高级的jetson nano。

下面是我做做项目过程中参考的一些博客，可以稍微看看。

全景拼接：

1. [Python图像拼接：创建全景图\_python拼接全景图-CSDN博客](https://blog.csdn.net/qq_43605229/article/details/115981870)
2. [全景图拼接算法实现与改进-CSDN博客](https://blog.csdn.net/yong_qi2015/article/details/105236719)
3. [一种针对圆柱形物体的拍摄点位生成算法 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/678750216)
4. [opencv-python圆柱图像展开 - CSDN文库](https://wenku.csdn.net/answer/11365c5dfe334803a7e114d3be51de64)
5. [opencv-python柱面展开 - CSDN文库](https://wenku.csdn.net/answer/b5c6c6af2ba045c1804397972728f451)
6. [python opencv 圆柱面展开\_mob649e816a77bf的技术博客\_51CTO博客](https://blog.51cto.com/u_16175524/9126617)
7. [柱面展开为平面python - CSDN文库](https://wenku.csdn.net/answer/df842f40a8be45f09d8742149c31a0d1)
8. [曲面（弧面、柱面）展平（拉直）瓶子标签识别ocr\_书页曲面展平算法-CSDN博客](https://blog.csdn.net/sunnyrainflower/article/details/132353931)

目标检测：

1. [快速上手yolov8进行目标检测，顺便读一读yolo的源码 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/673924499)
2. [目标检测与YOLO算法（用Python实现目标检测）\_yolo算法实现图片标框-CSDN博客](https://blog.csdn.net/NSSWTT/article/details/107612485)
3. [Jetson AGX Xavier python3.6环境运行YOLOV8\_python3.6 安装yolov8-CSDN博客](https://blog.csdn.net/xc18113397842/article/details/135775301)
4. [最终章-让我们的自制推理框架实现Yolov5的推理 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/618043840)
5. [不同颜色真空采血管用途 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/403431779)

模糊图像识别：

1. [OpenCV快速傅里叶变换(FFT)用于图像和视频流的模糊检测 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/245527379)
2. [模糊图像检测-无参考图像的清晰度评价 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/97024018)
3. [Python+Opencv检测模糊图片-CSDN博客](https://blog.csdn.net/WZZ18191171661/article/details/96602043)