

传统视觉部署避坑手册-RoboMaster-QIQI

配置工控环境

- 基本配置

所需工具：

1. 空U盘(最好64G起步，烧镜像别买惠普u盘，推荐金士顿)

1. ubuntu官网下载所需版本的镜像，用镜像烧录工具，如：[SD Card Formatter](#)、[Fedora Media Writer](#)等，把下载的(.iso)文件烧录到空u盘中，制作系统盘

烧录镜像前最好先把u盘格式化，推荐使用[Fedora Media Writer](#)中的格式化功能

2. 买来的主板装好内存条、固态硬盘，最好直接把主板装到保护壳内再进行配置，以防损坏

工控电源一定要看好是否适配主板的电压电流，低了开不了机，高了会烧主板

3. 开机前先插系统盘再开机，然后看出不出BIOS选择系统选项进入的GUI界面，若没有就开机前按F2进入BIOS，在boot选项中把系统盘启动项换到首选项，然后保存并退出

4. 装ubuntu 22.04 LTS则在BIOS系统选项GUI界面中选择Try to install ubuntu，若是装ubuntu 20.04 LTS则选择ubuntu，这两项都能进入ubuntu安装界面

5. 安装时，语言选中文，正常安装且附带媒体播放器等第三方功能，正常安装ubuntu 22.04 LTS(若是重装则选择删除当前的ubuntu 22.04 LTS并重新安装)

6. 安装完毕，检查是否有wifi，没有就买个带天线免驱动安装的wifi接收器，附可购链接<https://m.tb.cn/h.5E0tk1j5Z840KMD?tk=ENAKWm1A8We> 配好wifi，在开始界面中打开软件更新器进行更新，更新完毕后重启，新建终端，输入

```
sudo apt-get update
sudo apt-get upgrade
```

来检查更新

7. 依次完成以下基本配置：

- 谷歌输入法

```
sudo apt install fcitx-googlepinyin
```

- Git

```
sudo apt install git
```

- QQ 官网下载
- Clash -> 加速访问github以及上传下载代码 队内资源共享或github下载
- steam++ (瓦特加速器) 可加速github, 涵盖windows和linux平台
- vscode -> 查看和编辑代码 官网下载
- 小鱼一键安装ROS2

```
wget http://fishros.com/install -O fishros && sudo bash fishros
```

- 工业相机对应的SDK和驱动工具包 去品牌官网下载页下载
 - edge浏览器 推荐使用, 在ubuntu上能用的浏览器里算是比较好用的, 账号登陆后自动同步收藏夹很方便, 教程随手收藏, 系统崩了重装不丢失
- 在主目录下新建文件夹HOME, 用于存放安装的其他软件或功能包等, 自瞄代码工作空间文件夹直接放到主目录下即可
 - 在主目录创建自瞄代码工作空间文件夹RM-Vision-Main, 然后用github把自瞄代码源代码文件夹src放进去, 整体呈现编译架构如下:

```
RM-Vision-Main(workspace)
|
|—src
|—log
|—build
|—install
```

其中log,install,build是由src源码包编译得到

下载安装包前需要终端输入uname -m或sudo dpkg --print-architecture查看系统架构, 选择对应版本的安装包, 后续如相机SDK工具包也要看好架构安装

- 到自瞄代码工作空间.../RM-Vision-Main下, 终端输入

```
colcon build --symlink-install
```

进行编译, 把编译过程中的错误一一解决即可

- 缺少camera_info_manager.hpp需要终端输入

```
sudo apt-get install ros-humble-camera-info-manager
```

- 运行陈君的自瞄代码需要看源码中每个功能包里的README.md装好相关依赖, 如serial_driver等

```
sudo apt install ros-humble-serial-driver
```

11. 直到自瞄代码的src能成功编译就算整个环境配置完毕了，建议再设置以下仿真工具rqt调出其图像显示和debug列表
12. 编译后运行自瞄代码时可能会提示缺少xacro，需要安装下载对应版本的xacro

```
sudo apt install ros-humble-xacro
```

13. 安装ROS仿真可视化工具foxglove 网站搜索fosglove在官网安装，如下链接：
<https://foxglove.dev/download> 还需要安装ros-foxglove进行桥接通信的包，才能在foxglove端收到ros2从相应端口发送的各节点的数据

```
sudo apt install ros-humble-foxglove-bridge
```

- VNC远程控制环境配置

准备一根网线，2m左右最优

1. 链接双方设置静态ip
2. ubuntu端在设置中的信息查看窗口系统是否为X11，若为X11则可以直接进行x11vnc的安装与配置；若为Wayland则需要先在系统中关掉WaylandEnable，然后重启再次查看，变为X11后再进行后续操作 具体操作如下：

1. 打开/etc/gdm3/custom.conf文件编辑，可以用nano或vim等，

```
sudo nano /etc/gdm3/custom.conf
```

2. 将#WaylandEnable=false这一行前面的#去掉，取消该行注释，最后保存并关闭文件
3. 重启GDM3

```
sudo systemctl restart gdm3
```

在Wayland改X11这一步中，部分工控完成该操作后重启可能会出现系统无法正常开机的现象，这就表明该工控无法修改为X11窗口系统，也就不能配置x11vnc远程操作环境，需要放弃该远程操控方案

3. 安装x11vnc及网络工具包

```
sudo apt install x11vnc net-tools -y
```

4. 设置远程连接密码

```
x11vnc -storepasswd
```

输入上述命令后会提示自行设置密码，密码输入时是不可见的，需要盲打，建议设为123 确认密码后，会打印提示信息

```
Write password to /home/qiqi/.vnc/passwd? [y]/n
```

同意后就会将密码文件存储在所示目录下的文件里，后续启动x11vnc服务时直接使用该密码文件即可

5. 常用x11vnc启动指令

```
x11vnc -auth guess -forever -loop -noxdamage -repeat -rfbauth  
/home/qiqi/.vnc/passwd -rfbport 5900 -shared
```

-rfbauth <密码文件路径> 设置vnc服务远程连接密码 -rfbport <启动端口号> 设置vnc服务远程连接所用端口

6. 设置vnc服务开机启动 前提为取消开机输入密码这一操作，改为直接进入桌面

1. 创建vnc配置文件

```
cd /etc/init  
sudo gedit x11vnc.conf
```

文件内容如下

```
exec /usr/bin/x11vnc -auth guess -forever -loop -noxdamage -repeat -  
rfbauth /home/qiqi/.vnc/passwd -rfbport 5900 -shared
```

2. 尝试启动vnc服务进行测试

```
source /etc/init/x11vnc.conf
```

3. 设置开机自启动 在某一路径下(最好是自瞄工作空间里)

```
gedit x11vnc.sh
```

内容如下：

```
#!/bin/bash  
source /etc/init/x11vnc.conf
```

移动文件并添加权限

```
sudo mv x11vnc.sh /etc/init.d/  
sudo chmod 777 /etc/init.d/x11vnc.sh
```

然后将其添加到启动项中，即ubuntu开始菜单栏自带的**启动应用程序**，重点在**命令栏**，需要输入

```
bash /etc/init.d/x11vnc.sh
```

4. 重启测试

```
sudo reboot
```

!!!到这步后一定先重启一次拔掉minipc的HDMI线，只用网线连接笔记本和minipc进行远程连接尝试，确认是否可以连接后再选择是否进行下面的可选操作

7. (可选操作)若配置完成后，通过网线进行vnc连接出现黑屏只有鼠标的情况，则需要在ubuntu端额外配置虚拟显示屏

完成虚拟显示屏配置后，minipc只能通过网线连接其他设备通过vnc打开其图形化显示界面，HDMI接显示屏的方式会失效，除非将配置文件中的内容全部注释掉或直接删除该文件

1. 安装虚拟显示器所需的软件包

```
sudo apt-get install xserver-xorg-core-hwe-18.04  
sudo apt-get install xserver-xorg-video-dummy-hwe-18.04 --fix-missing
```

2. 创建配置文件

```
sudo vi /usr/share/X11/xorg.conf.d/xorg.conf
```

粘贴以下内容

```
Section "Monitor"  
    Identifier "Monitor0"  
    HorizSync 28.0-80.0  
    VertRefresh 48.0-75.0  
    # https://arachnoid.com/modelines/  
    # 1920x1080 @ 60.00 Hz (GTF) hsync: 67.08 kHz; pclk: 172.80 MHz  
    Modeline "1920x1080_60.00" 172.80 1920 2040 2248 2576 1080 1081 1084 1118
```

```
-HSync +Vsync
EndSection
Section "Device"
    Identifier "Card0"
    Driver "dummy"
    VideoRam 256000
EndSection
Section "Screen"
    DefaultDepth 24
    Identifier "Screen0"
    Device "Card0"
    Monitor "Monitor0"
    SubSection "Display"
        Depth 24
        Modes "1920x1080_60.00"
    EndSubSection
EndSection
```

保存后退出 然后重启系统进行远程连接测试

编译/包

- 每次修改src中的源代码后一定要在/home/dev_ws中重新编译一遍，替换掉上次编译的代码，如此才能实现代码的更新 若只是修改src中某文件或代码中的参数，则直接在工作空间重新编译即可实现自动覆盖，若是更改幅度较大或上次编译中出现error，则需要将上次编译后的包全部删除重新编译

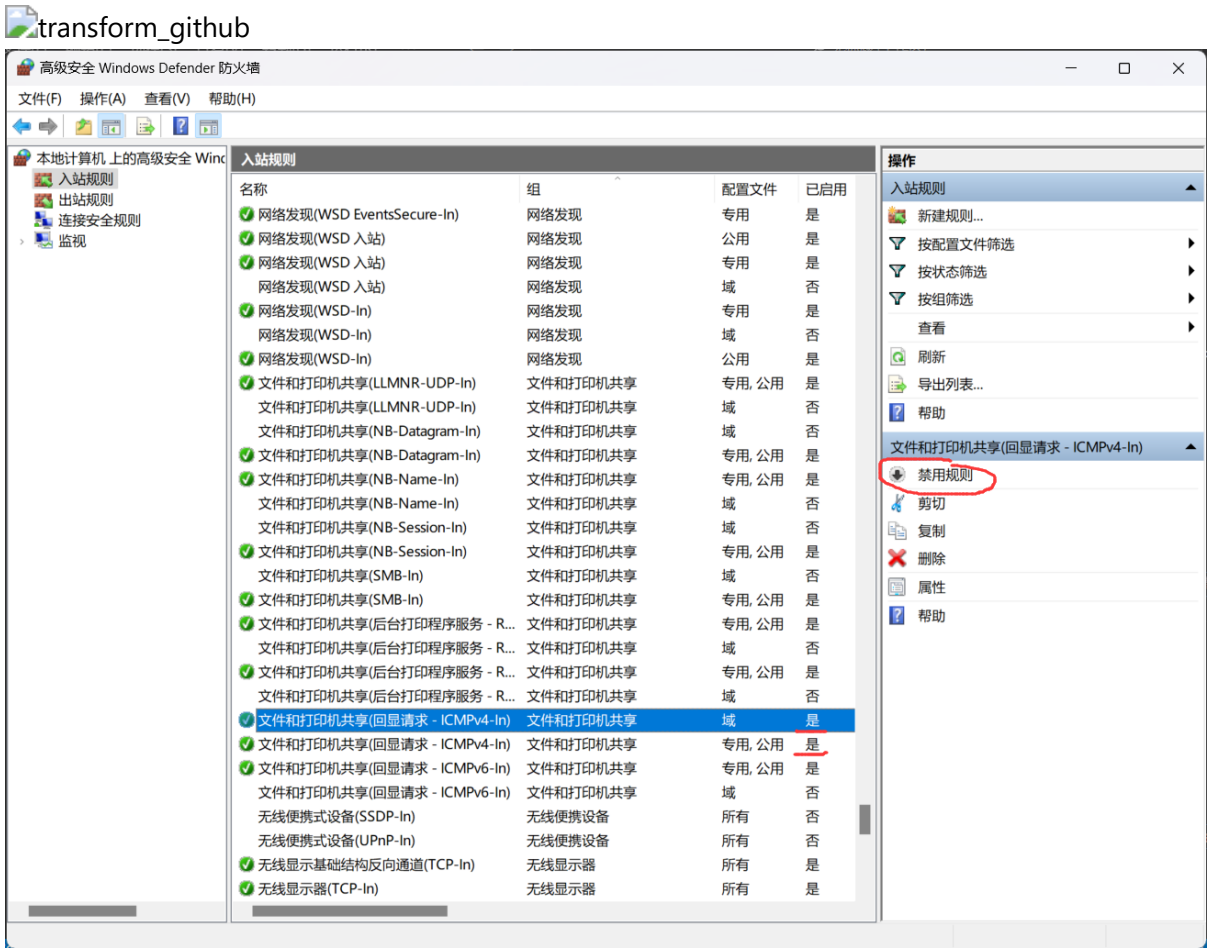
```
colcon build --symlink-install //编译的指令
source ./install/setup.bash //在工作空间中完成该命令
ros2 run //该命令执行的是install文件夹中的文件，故需要更新其中文件才能在终端中运行新版代码
```

- 每个具体到节点的代码文件，在修改后都需要在对应父类文件夹的setup.py文件中进行相应的配置：找到entry_points={...} 修改大括号内的内容即可（初次配置好后默认就已经是完整的，无需再配置）
- 在ubuntu上用命令行下载安装某些包时最好保证较好的网速，而且部分包在安装时会定位到外网，也就是需要挂梯子，如在conda虚拟环境中用pip安装torch相关工具包

ubuntu系统的梯子可以用Clash，免费注册一个，有免费的加速节点，第一次用之前还要手动配置一下网络代理IP，之后上外网就先切换为手动网络代理+开启Clash，如果是访问国内网站如csdn等就再切换回自动代理即可

通信

- Windows ping Ubuntu
 1. 先设置好windows和ubuntu(主板)的静态IPv4地址，然后网线连接
 2. windows打开Windows Defender 防火墙，在入站规则中的名称一栏下找文件和打印机共享(回显请求 - ICMPv4-In)，组为文件和打印机共享，共有两个，分别将这两个规则启用即可



相机

- 调用hik_camera前一定先source对应工作空间里的.bash/.sh文件(一般是用.bash文件)
- 每次进行完相机标定后需要将opt.yaml改名为camera_info.yaml(与包中的相机参数文件名保持一致), 然后在以下两个路径中替换掉原参数文件, 分别是: ./RM-Vision-main/ws/src/ros2_hik_camera-main/config/RM-Vision-main/ws/src/rm_vision-main/rm_vision-bringup/config

安装camera_calibration包时不要盲目跟教程直接调用!!!

```
sudo apt install ros-galactic-camera-calibration-parsers
```

上面这条命令是从csdn教程上抠的, 它会在/opt/ros/humble/中下载并安装camera-calibration-parsers, 而不是多数教程会用到的camera-calibration 所以一般会调用下面这条下载安装命令

```
sudo apt install ros-humble-camera-calibration
```

- 工业相机标定流程
 - 1. 安装相机标定工具包

```
sudo apt install ros-humble-camera-calibration-parsers  
sudo apt install ros-humble-camera-info-manager  
sudo apt install ros-humble-launch-testing-ament-cmake
```

2. 官网下载对应品牌工业相机的驱动和SDK 如海康机器人官网:

<https://www.hikrobotics.com/cn/machinevision/service/download?module=0>

3. 启动相机节点

1. 先进入包含有相机启动文件的工作空间(事先编译过的)
2. 手动包含ROS相关工具指令

```
./install/setup.bash
```

3. 启动相机节点(这里以海康相机为例)

```
ros2 launch hik_camera hik_camera.launch.py
```

4. 相机标定

```
ros2 run camera_calibration cameracalibrator --size 11x8 --square 0.50  
image:=/camera/image_raw camera:=/camera
```

--size后的参数参考实际使用的标定板的规格，同样--square后的参数也需根据实际进行调整，
image:=/表示图像发布话题， camera:=/表示相机名称


📁 北极熊算法-部署华师视觉项目-群精华_github

<

2.7.3 经验总结
常用数字。

RM273

...



辽科-COD-王柏程

楼主

06-30 00:12 发表帖子

分享一个标定手段，首先测出pitch转轴离地高度，然后将装甲板放置于一定距离远，离地高度与测量值一致（即期望 $z=0$ ）。验证测距误差足够小后，调整urdf坐标变换中camera与gimbal的pitch旋转直至 z 贴近0。最后可以变换距离再测一组 z 的数据作为验证。——来自cj

👁

阅读 348

仿真

- 调出`rqt`或`rviz2`后可以查看经过cv处理后的图像`result_img`，在陈君自瞄代码的实际运行情况中，仿真中查看图像的中心处有绘制一个红色小圆圈作为图像中心点，与中心点绘制相关的代码为`rm_auto_aim-main/armor_detector/src/detectoe.cpp`中的`cam_center`参数部分 该中心点仅供参考图像中心位置，作为实际击打路径的比较点，也可以观察中心点是否在图像中心来判断源码中相机内参文件是否正确对应
- 推荐使用`rqt`进行调参，可在菜单栏中打开debug列表、话题实时发布列表、图像显示框等实用窗口

君佬部署自瞄代码-流程剖析

全过程基于foxglove实现

- 君佬部署自瞄直播回放 <https://flowus.cn/lihanchen/share/8e866082-3758-447d-a034-aa1ab813a417>
- 查看设备接口及其总线id等信息

```
lsusb
```

- 安装温度传感器查看主板及其连接设备的温度

```
sudo apt install lm-sensors
sensors
```

- 计算src/rm_vision_main/rm_vision_bringup/config/laungh_params.yaml中的r_xyz_factor: 观察/tracker/measurement.x的Plot图, 得出x的范围值, 如 $[-1.88, -1.96]$, 取差值 0.08 , $0.08/4=0.02$, 取x轴差值除以4 $0.02^2=0.0004=4e-4$, 将除后结果取平方 $0.0004/1.88=2e-4$ 量级\$ 平方结果再去比x轴范围值中的最小值, 最后取结果的量级 $2e-4$ 即 $2 * 10^{-4}$

ps: 若使用计算值后效果并不理想, 则只能手动微调, 一般只需要改指数即可; 或者直接把科学计数法形式换成小数形式, 改好万分位再微调十万分位

- 海康相机限帧率? 海康的数字软件可以设置帧率, 默认70帧, 可以设置160帧
- 相机标定稳定性评价标准 相机标定后, foxglove中观测/detector/armors中的数据, /armors[]/pose[]/position中的norm数据与相机到装甲板识别中心距离理应基本相等, 单位(m) "5m内相差0.2m稳态误差大概为 $0.2/5=0.04$, 略偏大, 尽量控制在 2% 左右, 相机FOV小(镜头焦距大)的情况下, 稳态误差越大影响越大"
- /tracker/info -> position的norm和x在调好参数后并不相等
- ubuntu系统之间通过wifi直接终端传输文件,

```
scp <document_name> <ubuntu_hostname>@<ubuntu_ip>:~/
```

document_name为文件名, 带后缀 ubuntu_hostname为ubuntu系统用户名 ubuntu_ip可在ubuntu系统联网后, 在终端输入

```
ifconfig
```

来进行查看 ~/表示传输到目标ubuntu系统的主目录下

- rm_vision/rm_vision_bringup/launch/vision_bringup.launch.py 君开始调参时更改了下方这两个参数的period, 调的英雄, 君把period都增加了0.5

```
delay_serial_launch = TimerAction(
    period = 1.0,
    action = [rm_serial_launch],
)
```

```
delay_tracker_node = TimerAction(
    period = 1.5,
    action = [tracker_node],
)
```

- `src/rm_gimbal_description/urdf/rm_gimbal.urdf.xacro` `<origin xyz="" rpy="" />`中, `rpy`均为弧度制, 需要量出角度后转弧度制, `xyz`单位为(m) 以yaw轴为例, 其取值范围为[-3.14, 3.14], 有正负之分是看yaw轴从0轴线以顺时针还是逆时针转到目标角度 角度转换可以直接搜`degree to rad`找线上转换器用

tip: 机械/电控量出真实值前, 可以先放入期望值进行初步调试, 看发弹、识别、坐标解算和跟随等基本功能是否呈现正常趋势

- 参数剖析: 坐标系转换、相机标定没问题后, 识别跟随小陀螺时, `/tracker/measurements`和`/tracker/target -> position/xyz`应该是目标装甲板在世界坐标系中的数据 `radius_2`为固定时间段内取某一时刻记录下`radius_1`的值 `/tracker/target -> velocity`
- 电控协议?
- foxglove的3D图中显示出的`aim_point`击打位置(一个白球)大概率是与电控的控制部分代码有关, 看这部分处理在视觉上位机还是电控下位机
- 调车时相机画面左上角的`Latency`高是因为开了debug调试画面, 把图像回传去掉后帧率就正常
- 要养成看[LOG]的习惯, 在foxglove里面可以直接调出来
- 每个相机最好放大识别画面看一下灯条的识别效果, 看灯条的角点标的准不准, 不准的话改`node_params.yaml`里的`binary_thres`, 根据实际效果进行调整 角点可以开`/detector/binary_img`看灯条二值化后的效果图, 根据这个来改`binary_thres.yaml`
- 步兵自瞄调参流程

1. 验证距离

更换相机参数文件, 每个相机的参数都需要通过标定获得, 具体评价标准往上找 期间最好保证相机帧率正常(指相机直出的帧率)

```
ros2 topic hz /camera_info
```

2. 根据相机的实际安装位置, 更

改`src/rm_vision/rm_vision_bringup/config/launch_params.yaml`中的`odom2camera`各项参数

需要注意, 即便机械组能够从图纸上直接获取精准参数, 但在实际安装时还是会存在一些难以避免的偏差, 需要手动在车上测量出实际数据, 或者在文件里一点点调出较为理想的参数

- `exposure_time` 曝光时间
- `r_xyz_factor` 实际识别的装甲板xyz到预测装甲板xyz的转换系数

3. 更改/src/rm_vision/rm_vision_bringup/config/node_params.yaml:

- odom2camera -> xyz rpy

4. foxglove里拉出Parameters列表和/tracker/info.position.x的Plot图，小幅度调整serial_driver.timestamp_offset使得plot图折线在一段时间内上下限基本可以用水平平行线限定，即保证波动幅度在很小范围内，整体分布大概在一个可限定的区间内 -> 简单来说，就是通过调这个参数让plot折线的极值收敛，更接近平均值

5. ----- 阶段测试 -----

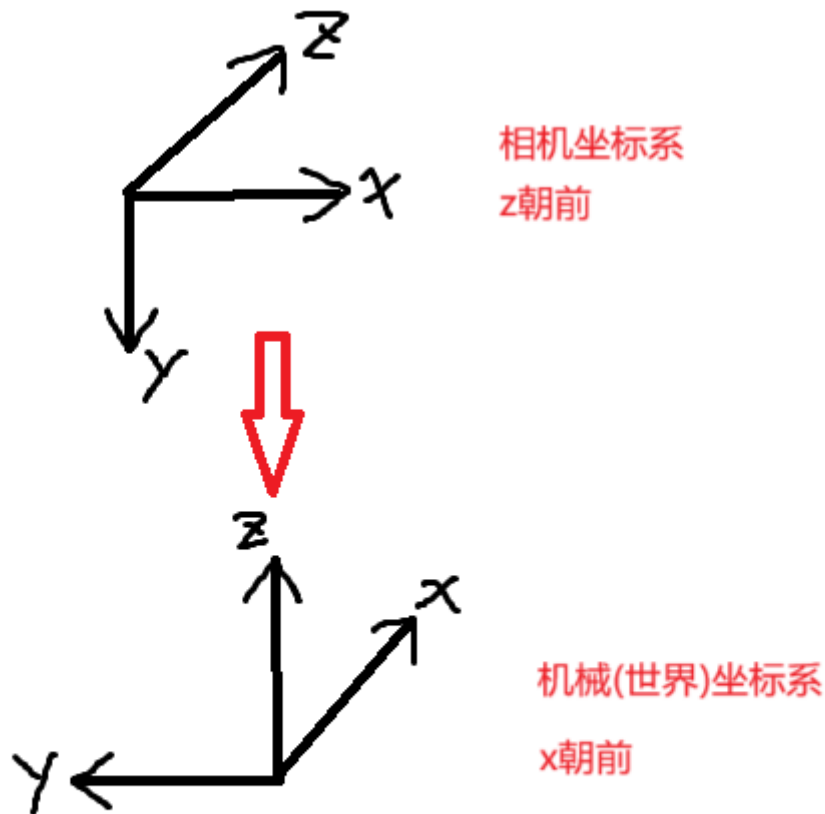
标准如下：靶车小陀螺状态下，对于相机识别装甲板画的线和识别率，装甲板固定位置不动，无论相机怎么晃，只要装甲板在视野范围内就不会丢失识别，不会出现相机晃到某个角度完全不识别的情况 foxglove的3D图中，红色装甲板为计算出的位置，蓝色装甲板为识别到并进行预测后的装甲板，调试好后的理想效果应该是蓝色装甲板略紧贴红色装甲板，有偏角无所谓，位置要求重叠在一起，并且转小陀螺后转过去的被识别到的蓝色装甲板不会乱飞，而是几乎紧贴着对应红色装甲板消失(主要评测标准)；靶车平移，红色装甲板跟着车动，蓝色装甲板理应先于红色装甲板进行移动，可描述为考虑了飞行时间的实际击打板

若foxglove的3D图刷新帧率低，可以关闭一些功能面板，只留需要观测的面板，减少软件负荷，提高刷新率

/tracker/target.yaw的plot折线图在靶车小陀螺时呈现规律变化

• 君佬谈自瞄

- 对于灯条识别，场上只需要调成曝光和二值化阈值就能达到较高的识别率
 - 二值化阈值 若该参数不合理，会导致灯条角点不在灯条边缘，使得pnp解算得到的识别距离错误，进而影响后续处理 因此每次在不同的环境下部署后，一定要检查灯条角点的识别是否精准
 - norm表示识别目标距离相机光心的距离，可以用来验证测距和相机标定是否正确
- 君佬对于灯条配对这方面并未做太多处理和限定，而是主要依靠数字识别再加一个分类器来完成装甲板目标的识别，且实际效果很好
- 数字识别其实还有小问题，它对于前哨站、哨兵图案等复杂图案的识别鲁棒性较差，若未将这类图案送入数据集进行训练，实际效果中很容易出现误识别的情况 若出现误识别的情况，可以将误识别的图像拍下来放入数据集的negative中进行训练
- 君佬代码中对于装甲板的识别和处理是分开写的，两者没有任何耦合。因此，在电控没有发送姿态数据的情况下，视觉完全可以对识别算法的正确性进行验证
 - 识别部分应用opencv，到处理部分后只需要接收识别部分计算出的x,y,z等参数，然后全部都在odom惯性系下进行处理，不需要考虑前面部分是如何运行的
- 坐标系转换

 transform_github


- 机器人状态从 `gimbal_description.urdf.xacro` 中读取坐标系转换数据
- tf时间戳
 - Latency是识别到图像后到处理话题发布间的延迟
 - 完成识别和处理部分的时间戳对齐后，动云台不动目标，在foxglove的3D图中看识别目标会不会抖，理应目标在惯性系下一动不动
- 预测
 - 对于平移目标，因为整车没有进行旋转，整车中心到识别装甲板中心的半径 r 一直无法迭代。因此，君佬在处理平移目标时会对 r 做抑制使其不会缩减到0
 - $Q \rightarrow \text{process}$ $R \rightarrow \text{measurement}$ Q, R 均为矩阵 Q 相当于概率论中的协方差，可以理解为运动噪声。例如：A \rightarrow B，通过A运动方程预测B的运动方程， Q 可以表示为预测结果与实际结果的匹配程度。状态量有 n 个， Q 就是 $n * n$ 矩阵 R 为观测量的噪声，卡尔曼滤波器更新时需要通过预测量和观测量来完成。观测量有 m 个， R 就是 $m * m$ 矩阵
 - `tracker/target`即发给电控的数据包
 - `processor/target.radius_1`为当前识别到的装甲板距离整车中心的距离，也可以理解为半径，来回跳变是因为该数据不断地从滤波器中存入拿出，可能会有两个稳定值是因为四块装甲板可能会有两个半径值
 -

- 更多内容请转陈君视频讲解回放(bilibili) https://www.bilibili.com/video/BV1oX4y167RP/?spm_id_from=333.1350.jump_directly&vd_source=619d2c4356ffd5e4f3ecbf199584c1a3
 - 1:50:00起 -> 讲解装甲板绘制
 - 1:56:00起 -> 讲解电控方面解析

24赛季视觉电控自瞄联调-个人总结版

整车进行实战训练前的一次完整自瞄调试流程

视觉可进行改动的参数:

1. 基础参数(给真实值):
 - 相机位姿(xyz/rpy)
2. 后续可调参数:
 - exposure_time //曝光时间

电控可进行改动的参数:

1. 基础参数(给真实值):
 - 枪口前推距离
 - 枪口垂直距离
 - 重力加速度系数
2. 后续可调参数:
 - 机器人固有时间偏差
 - 视觉计算时间
 - 空气阻力系数

1. 确认弹速

弹速是后续调车准确性的保障!!!

正式开始调参前先以同一弹速进行多次发弹, 在主控上查看每次发弹的实际弹速, 使其稳定在一个较小范围内, 然后将稳定后实际平均弹速填入预设的弹速(电控代码中防止读不到测速模块的弹速而设置的宏定义常数)

2. 安装相机并修改其相对位姿参数(xyz/rpy)

自瞄调参前一定要给相机加装保护壳, 规范安装到每车图纸的对应位置, 这里的规范安装指用螺丝稳定固定 该阶段需要根据机械组在图纸上测量imu到镜头表面的距离来修改位姿参数xyz

imu指c板上相对较大的芯片, xyz坐标系是以imu为原点, 右手食指朝前方作x轴, 大拇指朝上作z轴, 中指垂直于大拇指与食指构成的平面作y轴进行坐标系建立 rpy(roll轴 pitch轴 yaw轴)分别是绕x轴、y轴和z轴进行转动的, 逆时针转动视为正向 imu到相机的相对位姿指从imu到相机镜头表面中心(光心)

3. 修改枪口前推距离和枪口垂直距离

基于23赛季马哥版下位机自瞄控制代码: 电控组在vision_task.h的宏定义常量中修改, 单位为m 弹丸推出点为两个摩擦轮中心(包括z轴厚度)连线的中点 枪口前推距离指的是c板上imu到弹丸推出点的x轴距离 枪口垂直距离为imu到到弹丸推出点的z轴距离

4. 打靶测试

1. 静止靶:

1. 在2m左右距离处, 保证相机位姿xyz数值正确的前提下, 调整rpy使弹丸落点在装甲板中心
2. 分别移动靶车至3m、5m、1m处, 调整空气阻力系数和相机位姿pitch轴参数使弹丸均能上板, 尽量使上下偏移量维持在较小的范围内

2. 移动靶:

1. 在2-3m距离处让靶车进行小陀螺, 先手动小陀螺(低转速)在foxglove中看自瞄仿真的预测效果, 若转了十几圈后未出现丢失识别的情况(仿真出来的板子一直是4块没有丢失), 则可以关掉foxglove进行自瞄击打看效果(开foxglove看仿真也会占用ros2数据传输速率, 降低程序处理速度, 因此要模拟上场时只开自瞄程序的情况)
2. 先保证慢速小陀螺命中率80%左右(有明显的预测行为, 每个装甲板转过来规律地打一梭子)后, 再尝试80W功率的快速小陀螺, 同样先看仿真效果再进行击打测试, 最后再数弹丸测命中率

5. 边打靶边调参(联调)

1. 首先保证自瞄的识别部分稳定 启动自瞄后, 在foxglove中调出识别图像result_img和/tracker/info的实时参数

1. 先让靶车开到1.5m远处, 装甲板正对相机, foxglove中看图像中装甲板识别率, 调整 `exposure_time, gain, threshold` 等识别部分参数, 保证装甲板正对相机的识别率稳定在95%以上, 测对识别率90%以上(理论上调好后识别率都在95%以上且很稳) 需要注意的是不同环境下光照亮度不同, 可能需要经常对曝光时间进行调整, 相机增益一般设置为16即可 最优识别场景为室内高光环境, 此时曝光时间只需调整为1000甚至更低一点, 其识别效果就很好; 若环境很暗, 无论曝光时间拉到多高识别效果大概率都不行, 因为低亮度低曝光条件下装甲板数字很难被识别并正确分类, 若调整为高曝光则灯条光线会出现严重膨胀导致无法识别
2. 靶车在1.5m左右的距离进行慢速小陀螺, 多转一段时间, 在仿真上看预测出的四块装甲板模型是否按照预期轨迹进行运动, 有无装甲板丢失情况 若四块装甲板几乎无运动轨迹和运动趋势, 大概率 `node_params.yaml` 文件中的相机位姿参数错误, 填的数据不是车上的实际参数; 若出现小陀螺时装甲板丢失情况, 大概率是识别部分不稳定, 需要调整识别部分的参数(当然, 也可能是环境亮度过暗导致数字识别不稳)