

实验六. 数据标注、模型训练及转换

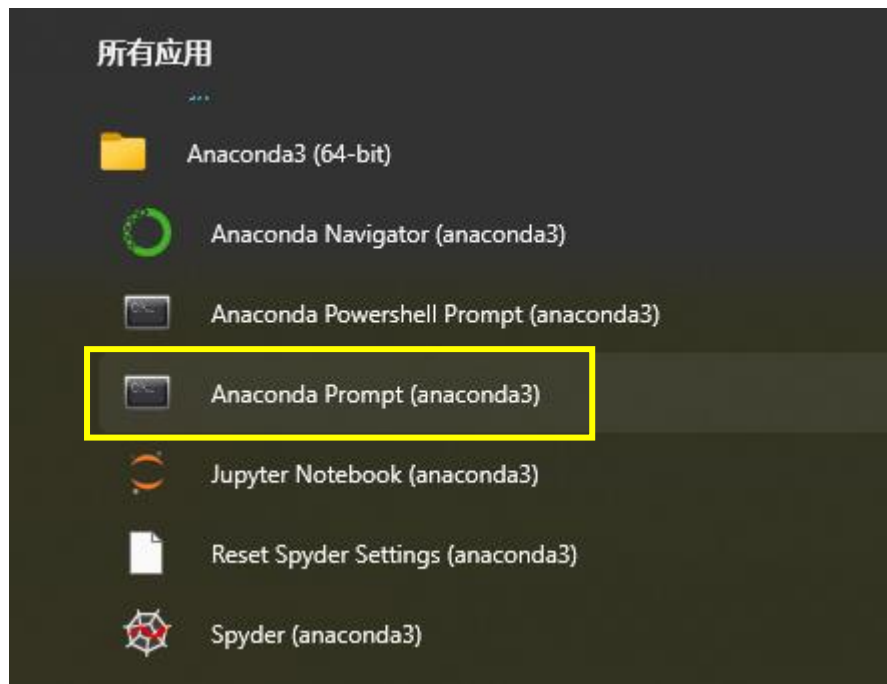
实验目的

了解数据标注、模型训练和转换的过程

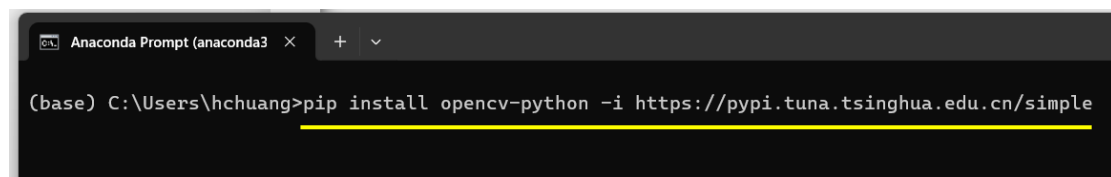
实验过程

1) 安装相应的库

点击 Anaconda Prompt



分别输入以下命令，安装相应库。



①OpenCV

`pip install opencv-python -i https://pypi.tuna.tsinghua.edu.cn/simple`

②Pytorch

`pip install torch -i https://pypi.tuna.tsinghua.edu.cn/simple`
`pip install torchvision -i https://pypi.tuna.tsinghua.edu.cn/simple`

③onnx

```
pip install onnx -i https://pypi.tuna.tsinghua.edu.cn/simple
pip install onnxruntime -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2) 上传文件

在计算机，新建一个工作文件夹（文件夹名不要用中文），如 D:/originbot。

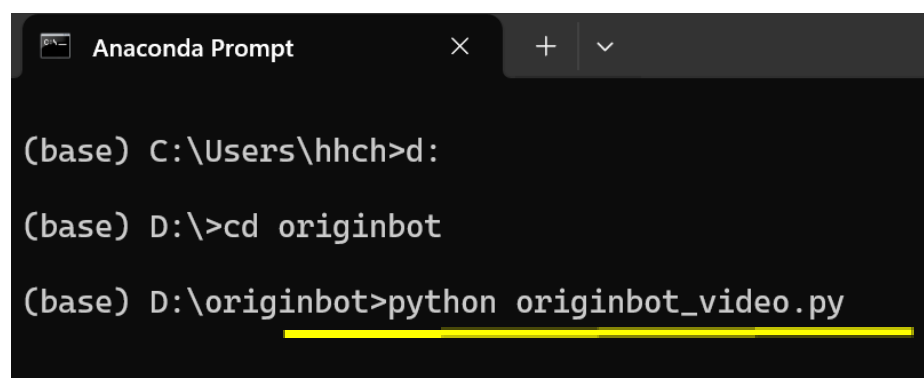
上传车道线视频文件（把视频文件名改为 record.avi）、originbot_video.py、originbot_data.py、originbot_train.py、originbot_onnx.py、originbot_test.py 到该文件夹内。

并新建文件夹 image_dataset。



3) 预观察车道线数据

转到工作文件夹，运行 `python originbot_video.py`

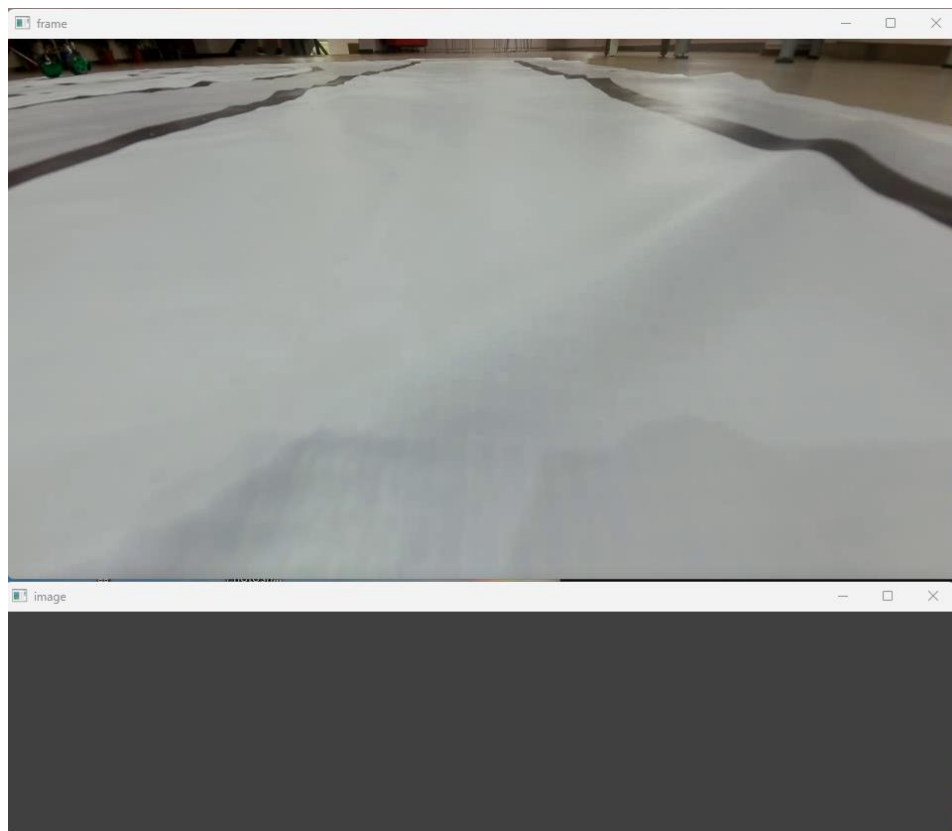


如车道线区域不清晰，可调节 originbot_video.py 文件中 h 的值。

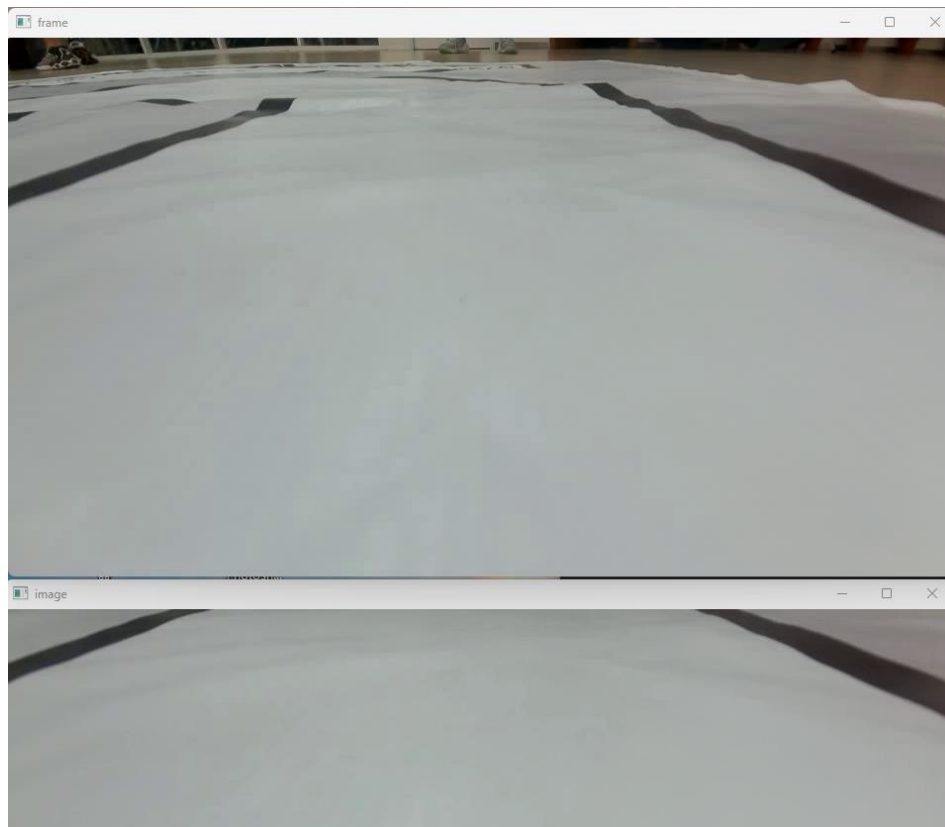
4) 车道线数据标准

运行 `python originbot_data.py`

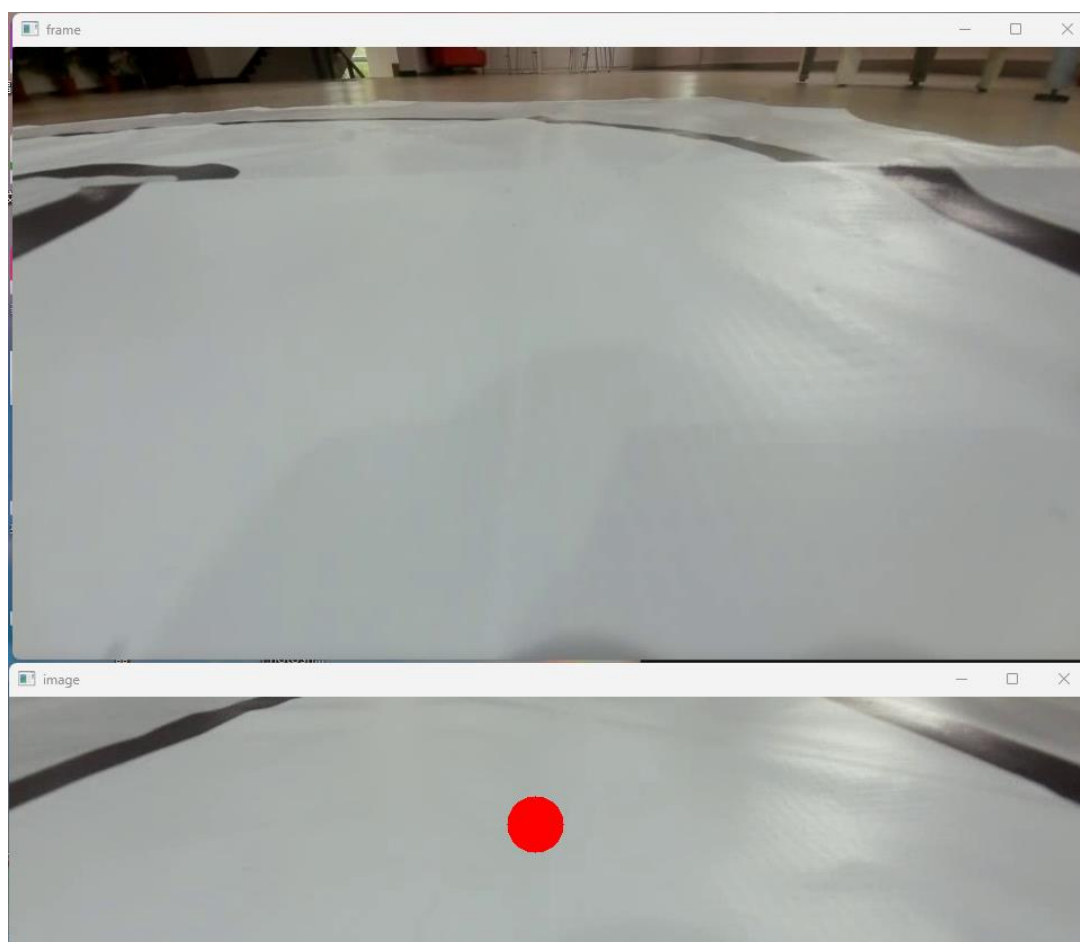
出现一大一小两个图像窗口。



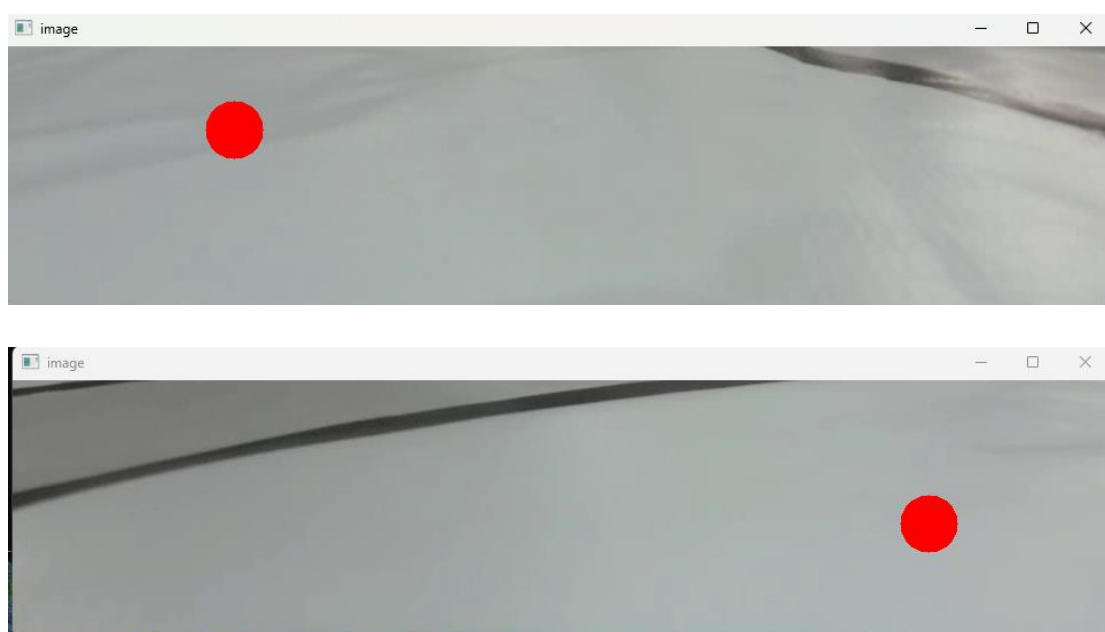
大窗口在播放车道线视频，准备采集哪段车道线，按下回车键。小窗口自动截取该段车道线。



在小窗口，用鼠标左键点取车道线中心点。

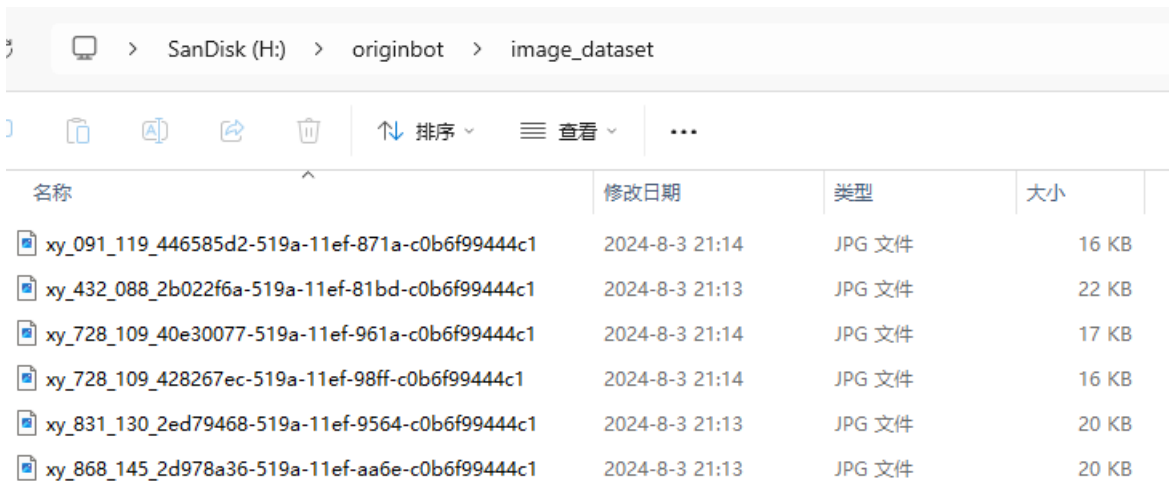


如出现单车道线图片，尽量把中心点标注在另外一侧，不要标注在图中间。



反复操作，选取一些典型的车道线，对数据进行标注，标注文件在

image_dataset 中，建议采集图片 100 张以上，可按“q”键+回车键退出标注。



名称	修改日期	类型	大小
xy_091_119_446585d2-519a-11ef-871a-c0b6f99444c1	2024-8-3 21:14	JPG 文件	16 KB
xy_432_088_2b022f6a-519a-11ef-81bd-c0b6f99444c1	2024-8-3 21:13	JPG 文件	22 KB
xy_728_109_40e30077-519a-11ef-961a-c0b6f99444c1	2024-8-3 21:14	JPG 文件	17 KB
xy_728_109_428267ec-519a-11ef-98ff-c0b6f99444c1	2024-8-3 21:14	JPG 文件	16 KB
xy_831_130_2ed79468-519a-11ef-9564-c0b6f99444c1	2024-8-3 21:13	JPG 文件	20 KB
xy_868_145_2d978a36-519a-11ef-aa6e-c0b6f99444c1	2024-8-3 21:13	JPG 文件	20 KB

5) 模型训练

运行训练模型，输入：

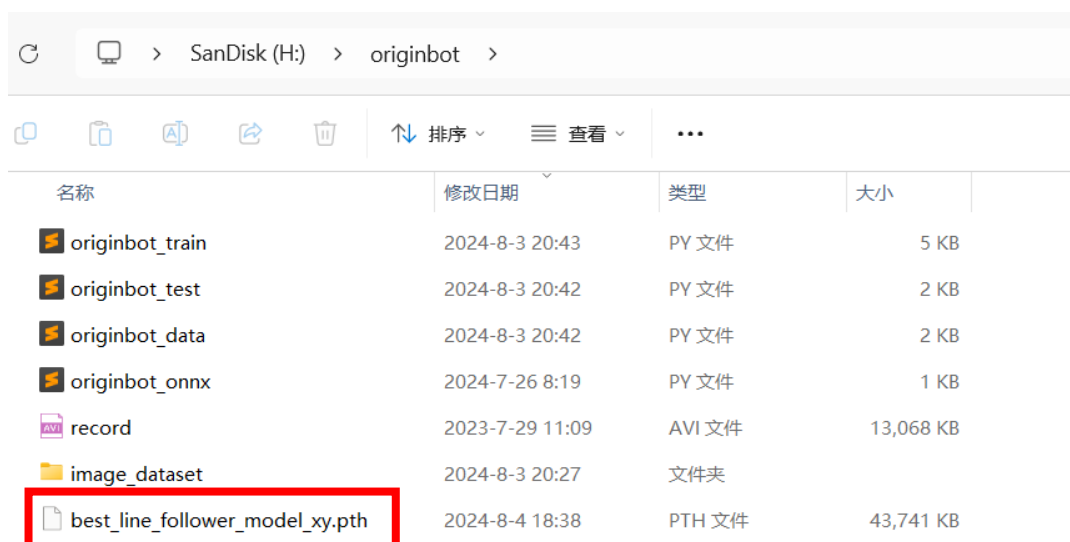
```
python originbot_train.py
```

```
e most up-to-date weights.
warnings.warn(msg)
0, 2.048392, 1.455791
save
1, 0.563686, 13.019084
2, 0.183825, 6.373345
3, 0.116847, 2.481668
4, 0.079362, 0.302848
save
5, 0.074989, 0.119999
save
6, 0.050870, 0.188420
```

从 0 (epoch)开始，到 49 (epoch)结束。记录最后一个 save 对于的损失函数数值。

```
30,0.052832, 0.016933
31,0.048765, 0.004349
save
32,0.035910, 0.006562
33,0.062664, 0.014692
34,0.033300, 0.030140
35,0.081289, 0.018947
36,0.090032, 0.033320
37,0.055062, 0.018136
38,0.057285, 0.016602
39,0.074300, 0.019293
40,0.048584, 0.034081
41,0.054161, 0.029918
42,0.058362, 0.023742
43,0.041876, 0.041305
44,0.035947, 0.023543
45,0.048690, 0.015443
46,0.033625, 0.016000
47,0.048572, 0.016300
48,0.051628, 0.031545
49,0.033570, 0.043242
```

生成文件 `best_line_follower_model_xy.pth`。



名称	修改日期	类型	大小
originbot_train	2024-8-3 20:43	PY 文件	5 KB
originbot_test	2024-8-3 20:42	PY 文件	2 KB
originbot_data	2024-8-3 20:42	PY 文件	2 KB
originbot_onnx	2024-7-26 8:19	PY 文件	1 KB
record	2023-7-29 11:09	AVI 文件	13,068 KB
image_dataset	2024-8-3 20:27	文件夹	
best_line_follower_model_xy.pth	2024-8-4 18:38	PTH 文件	43,741 KB

6) 模型转换成 onnx 格式

输入

```
python originbot_onnx.py
```

生成文件 `best_line_follower_model_xy.onnx`。

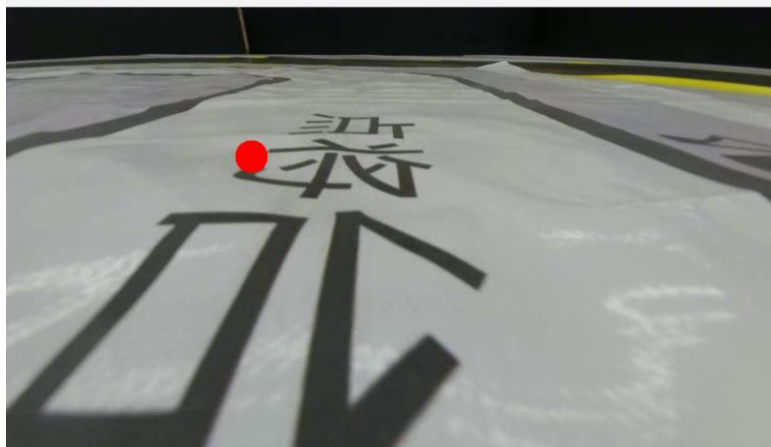
SanDisk (H:) > originbot >				
名称	修改日期	类型	大小	
originbot_train	2024-8-3 20:43	PY 文件	5 KB	
originbot_test	2024-8-3 20:42	PY 文件	2 KB	
originbot_data	2024-8-3 20:42	PY 文件	2 KB	
originbot_onnx	2024-7-26 8:19	PY 文件	1 KB	
record	2023-7-29 11:09	AVI 文件	13,068 KB	
image_dataset	2024-8-3 20:27	文件夹		
best_line_follower_model_xy.pth	2024-8-4 18:38	PTH 文件	43,741 KB	
best_line_follower_model_xy.onnx	2024-8-4 18:48	ONNX 文件	43,653 KB	

7) 验证 onnx 模型的准确性

输入

```
python originbot_test.py
```

观察红点（推理出来的坐标值），是否始终落在车道线中



8) 模型转换成 bin 格式

(1) 安装 Docker

打开计算机上的虚拟机，ctr+alt+t 打开一个终端

输入: `wget http://fishros.com/install -O fishros && . Fishros`

选择 8, 安装 Docker


```
wget -c
```

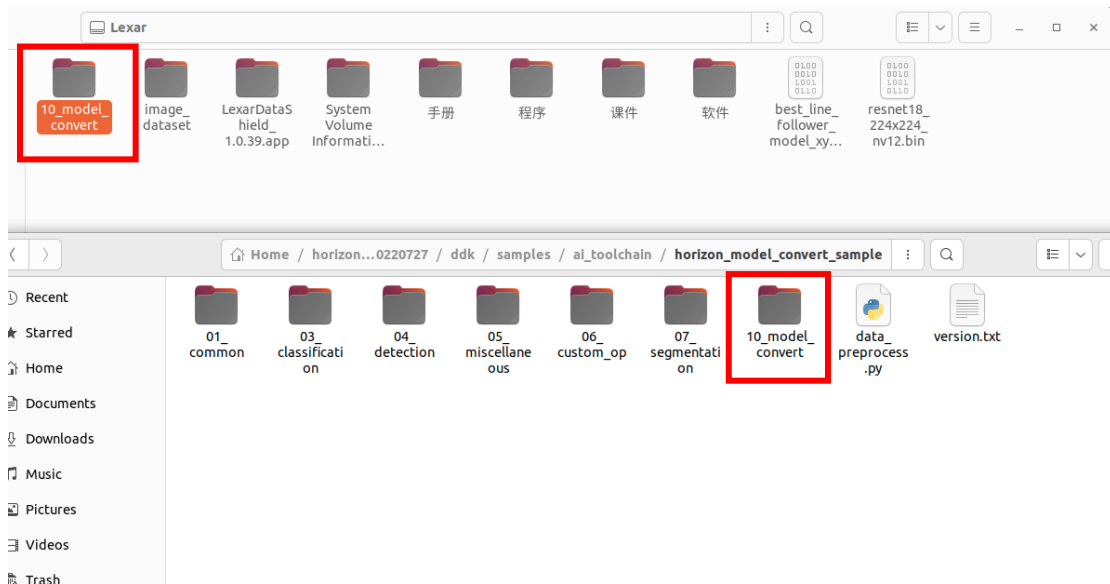
```
ftp://vrftp.horizon.ai/Open_Explorer_gcc_9.3.0/2.3.3/x3pi_toolchain/ai_toolchain_2.3.3.tar.gz
```

装载该镜像文件

```
docker load -i ai_toolchain_2.3.3.tar.gz
```

将文件夹 10_model_convert 拷贝到

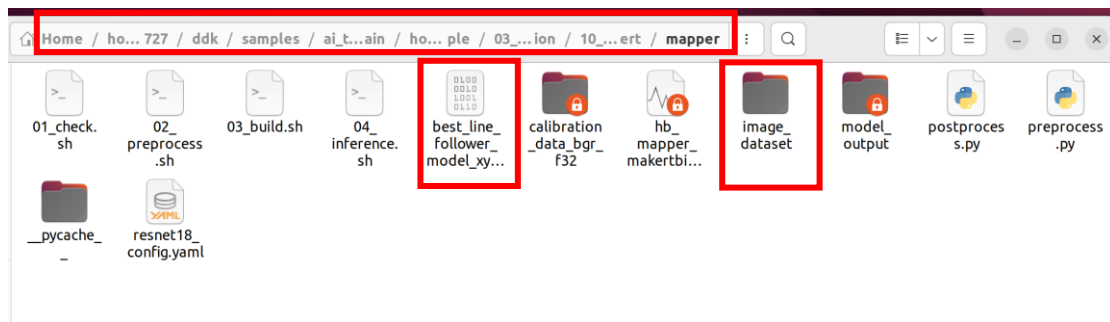
horizon_xj3_open_explorer_v2.3.3_20220727/ddk/samples/ai_toolchain/horizon_model_convert_sample/这个目录下。



(3) 转换

①将标注好的数据集文件夹 image_dataset 和生成的 onnx 模型拷贝到以下

/home/gyh/horizon_xj3_open_explorer_v2.3.3_20220727/ddk/samples/ai_toolchain/horizon_model_convert_sample/03_classification/10_model_convert/mapper 目录中



②cd horizon_xj3_open_explorer_v2.3.3_20220727/

③sudo bash run_docker.sh /data cpu

```

root@228b7eafa324: /open_explorer
gyh@gyh-vpc:~/horizon_xj3_open_explorer_v2.3.3_20220727$ sudo bash run_docker.sh /data cpu
[sudo] password for gyh:
Docker version is 2.3.3
Dataset path is /data
OpenExplorer package path is /home/gyh/horizon_xj3_open_explorer_v2.3.3_20220727
root@228b7eafa324:/open_explorer#

```

④cd

ddk/samples/ai_toolchain/horizon_model_convert_sample/03_classification/10_model_convert/mapper/

⑤sh 02_preprocess.sh

```

root@228b7eafa324:/open_explorer# cd ddk/samples/ai_toolchain/horizon_model_convert_sample/03_classification/10_model_convert/mapper/
root@228b7eafa324:/open_explorer/ddk/samples/ai_toolchain/horizon_model_convert_sample/03_classification/10_model_convert/mapper# sh 02_preprocess.sh

```

⑥sh 03_build.sh

```

2024-10-21 17:24:52,444 INFO scale_value : 0.0171248,0.017507,0.0174292,
2024-10-21 17:24:52,444 INFO cal_data_dir : /open_explorer/ddk/samples/ai_toolchain/horizon_model_convert_sample/03_classification/10_model_convert/mapper/calibration_data_bgr_f32
2024-10-21 17:24:52,444 INFO -----input info : input end -----
2024-10-21 17:24:52,444 INFO -----
2024-10-21 17:24:52,444 INFO ##### calibration_parameters info #####
2024-10-21 17:24:52,444 INFO preprocess_on : False
2024-10-21 17:24:52,444 INFO calibration_type: : kl
2024-10-21 17:24:52,444 INFO cal_data_type : N/A
2024-10-21 17:24:52,444 INFO ##### compiler_parameters info #####
2024-10-21 17:24:52,444 INFO hbdk_pass_through_params: --fast --03
2024-10-21 17:24:52,444 INFO input-source : {'input': 'pyramid', '_default_value': 'ddr'}
2024-10-21 17:24:52,458 INFO Convert to runtime bin file successfully!
2024-10-21 17:24:52,458 INFO End Model Convert
root@228b7eafa324:/open_explorer/ddk/samples/ai_toolchain/horizon_model_convert_sample/03_classification/10_model_convert/mapper#

```

⑦copy model_output/resnet18_224X224_nv12.bin 到当地计算上

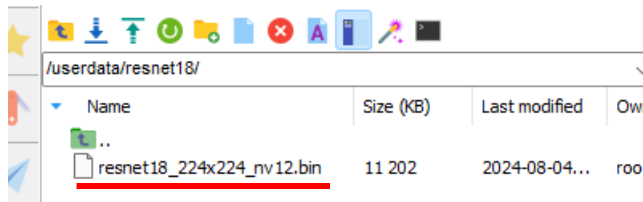


实验七. 在小车上模型部署

实验过程

1) ssh 连接到小车

2) 将文件 resnet18_224x224_nv12.bin 上传到/userdata/resnet18



3) 启动小车底盘通讯和摄像头，输入

```
ros2 launch originbot_bringup originbot.launch.py use_camera:=true
```

```
root@ubuntu:/userdata/dev_ws# ros2 launch originbot_bringup originbot.launch.py use_camera:=true
[INFO] [launch]: All log files can be found below /root/.ros/log/2024-08-04-20-07-04-474491-ubuntu-3938
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [originbot_base-1]: process started with pid [3940]
[INFO] [static_transform_publisher-2]: process started with pid [3942]
[INFO] [static_transform_publisher-3]: process started with pid [3944]
[INFO] [mipi_cam-4]: process started with pid [3946]
[INFO] [transport_img-5]: process started with pid [3948]
[mipi_cam-4] [WARN] [1722773225.012976925] [mipi_cam]: This is version for optimizing camera timestamp!
[originbot_base-1] Loading parameters:
[originbot_base-1] - port name: ttyS3
[originbot_base-1] - correct factor vx: 0.8980
[originbot_base-1] - correct factor vth: 0.8740
[originbot_base-1] - auto stop on: 0
[originbot_base-1] - use imu: 0
[static_transform_publisher-3] [INFO] [1722773225.448238349] [static_transform_publisher_Y3MmezdnIBLZZBL]:
[static_transform_publisher-2] [INFO] [1722773225.458544386] [static_transform_publisher_QAXjV33HTk2sngxvH]:
link'
[originbot_base-1] [INFO] [1722773225.491664387] [originbot_base]: originbot serial port opened
[originbot_base-1] [INFO] [1722773225.993605665] [originbot_base]: OriginBot Start, enjoy it.
```

4) 启动推理程序

SSH 连接小车，再开第 2 个窗口，输入

```
ros2 run racing_track_detection_resnet racing_track_detection_resnet
```

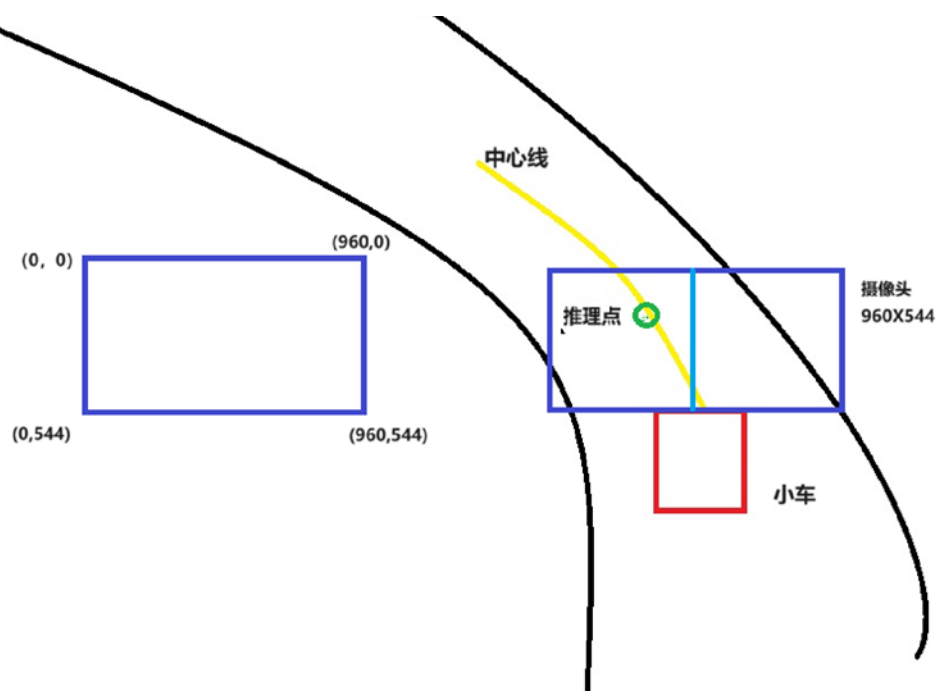
出现滚动，说明推理进行中。

```
[INFO] [1722773527.268269531] [TrackDetectionNode]: post coor x: 630 y:272
[INFO] [1722773527.268524909] [dnn]: task id: 0 set bpu core: 2
[INFO] [1722773527.277575808] [TrackDetectionNode]: coor rawx: 0.313268, rawy:-0.006707, x: 630.368713 y:272.751160
[INFO] [1722773527.277835320] [TrackDetectionNode]: post coor x: 630 y:272
[INFO] [1722773527.302021454] [TrackDetectionNode]: input size:1 roi size:1
[INFO] [1722773527.302276464] [dnn]: task id: 0 set bpu core: 2
[INFO] [1722773527.311351073] [TrackDetectionNode]: coor rawx: 0.310683, rawy:-0.010697, x: 629.128052 y:273.198090
[INFO] [1722773527.311616629] [TrackDetectionNode]: post coor x: 629 y:273
[INFO] [1722773527.335543460] [TrackDetectionNode]: input size:1 roi size:1
[INFO] [1722773527.335796261] [dnn]: task id: 0 set bpu core: 2
[INFO] [1722773527.344821435] [TrackDetectionNode]: coor rawx: 0.312677, rawy:-0.009462, x: 630.084900 y:273.059692
[INFO] [1722773527.345082197] [TrackDetectionNode]: post coor x: 630 y:273
[INFO] [1722773527.369337732] [TrackDetectionNode]: input size:1 roi size:1
[INFO] [1722773527.369594742] [dnn]: task id: 0 set bpu core: 2
[INFO] [1722773527.378652137] [TrackDetectionNode]: coor rawx: 0.314524, rawy:-0.011875, x: 630.971497 y:273.330017
[INFO] [1722773527.378914441] [TrackDetectionNode]: post coor x: 630 y:273
[INFO] [1722773527.402886331] [TrackDetectionNode]: input size:1 roi size:1
[INFO] [1722773527.403145592] [dnn]: task id: 0 set bpu core: 2
[INFO] [1722773527.412190399] [TrackDetectionNode]: coor rawx: 0.316213, rawy:-0.008844, x: 631.781982 y:272.990479
[INFO] [1722773527.412450119] [TrackDetectionNode]: post coor x: 631 y:272
[INFO] [1722773527.436512749] [TrackDetectionNode]: input size:1 roi size:1
[INFO] [1722773527.436817111] [dnn]: task id: 0 set bpu core: 2
[INFO] [1722773527.445880174] [TrackDetectionNode]: coor rawx: 0.314823, rawy:-0.009352, x: 631.114868 y:273.047424
```

实验八. 基于 AI 车道巡线

实验过程

1) 计算小车转角



参考上图，计算出小车的转角（弧度），填写 lanedetection.py 文件中小车速度和转角

```
def point_callback(self, msg):  
    self.point=msg  
  
    #self.point.point.x    识别出来的x坐标  
    #self.point.point.y    识别出来的y坐标  
  
    self.twist.linear.x =      #小车速度  
    self.twist.angular.z =     #小车转角
```

2) 创建 lanedetection 节点（①上传文件、②修改 setup.py、③编辑 build）

3) 开三个窗口，运行小车车道巡线

```
ros2 launch originbot_bringup originbot.launch.py use_camera:=true  
ros2 run racing_track_detection_resnet racing_track_detection_resnet  
ros2 run lanedetection lanedetection
```