Model prediction

Note: The environment has been set up. Simply follow the instructions to run the corresponding function commands to use it.

1. Program Description

Use Python to call a USB camera or depth camera using a pre-trained module. Here, we use the model we trained to recognize oranges as an example. If you need to train your own model, please refer to the two links below. However, training on a RDK X5 is not recommended.

Annotation: GitHub - HumanSignal/label-studio: Label Studio is a multi-type data labeling and annotation tool with standardized output format

Training: Model Training with Ultralytics YOLO - Ultralytics YOLO Docs

2. Program Startup

3.1. Camera Startup

Start the following program based on your camera model. In the terminal, enter:

```
#usb camera
ros2 launch usb_cam camera.launch.py
#nuwa camera
ros2 launch ascamera hp60c.launch.py
```

Open another terminal and navigate to the code folder:

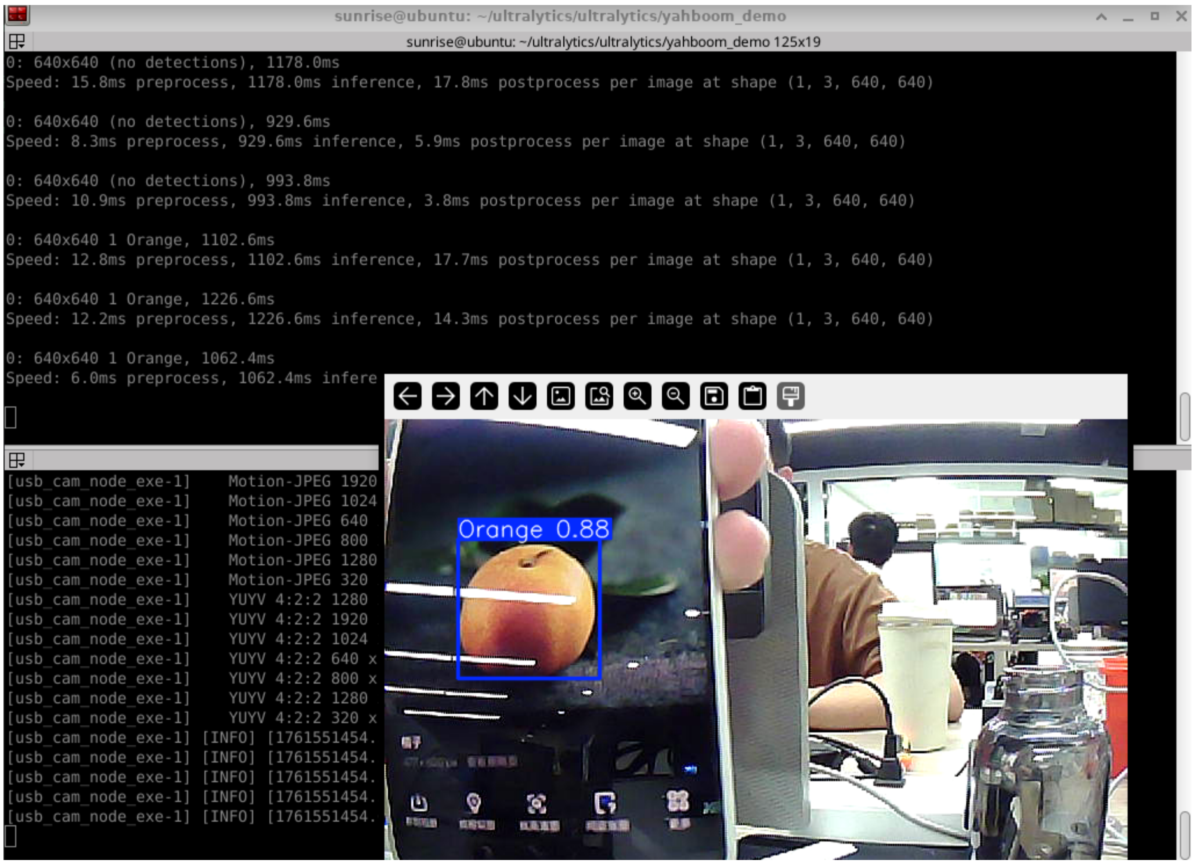
```
cd /home/sunrise/ultralytics/ultralytics/yahboom_demo
```

Run the code: Click the preview screen and press q to terminate the program!

```
python3 06.orange_camera_usb.py
```

Preview

Yolo recognition output video location: /home/sunrise/ultralytics/ultralytics/output/


sunrise@ubuntu: ~/ultralytics/ultralytics/yahboom_demo
sunrise@ubuntu: ~/ultralytics/ultralytics/yahboom_demo 125x19
Orange 0.88

```
0: 640x640 (no detections), 1178.0ms
Speed: 15.8ms preprocess, 1178.0ms inference, 17.8ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 (no detections), 929.6ms
Speed: 8.3ms preprocess, 929.6ms inference, 5.9ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 (no detections), 993.8ms
Speed: 10.9ms preprocess, 993.8ms inference, 3.8ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 1 Orange, 1102.6ms
Speed: 12.8ms preprocess, 1102.6ms inference, 17.7ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 1 Orange, 1226.6ms
Speed: 12.2ms preprocess, 1226.6ms inference, 14.3ms postprocess per image at shape (1, 3, 640, 640)

0: 640x640 1 Orange, 1062.4ms
Speed: 6.0ms preprocess, 1062.4ms infere
```

```
[usb_cam_node_exe-1]    Motion-JPEG 1920
[usb_cam_node_exe-1]    Motion-JPEG 1024
[usb_cam_node_exe-1]    Motion-JPEG 640
[usb_cam_node_exe-1]    Motion-JPEG 800
[usb_cam_node_exe-1]    Motion-JPEG 1280
[usb_cam_node_exe-1]    Motion-JPEG 320
[usb_cam_node_exe-1]    YUYV 4:2:2 1280
[usb_cam_node_exe-1]    YUYV 4:2:2 1920
[usb_cam_node_exe-1]    YUYV 4:2:2 1024
[usb_cam_node_exe-1]    YUYV 4:2:2 640 x
[usb_cam_node_exe-1]    YUYV 4:2:2 800 x
[usb_cam_node_exe-1]    YUYV 4:2:2 1280
[usb_cam_node_exe-1]    YUYV 4:2:2 320 x
[usb_cam_node_exe-1] [INFO] [1761551454.
[usb_cam_node_exe-1] [INFO] [1761551454.
[usb_cam_node_exe-1] [INFO] [1761551454.
[usb_cam_node_exe-1] [INFO] [1761551454.
[usb_cam_node_exe-1] [INFO] [1761551454.
```

Sample code:

```python
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Image, CompressedImage
from cv_bridge import CvBridge
import cv2
from ultralytics import YOLO
import os

class Image_detection(Node):
    def __init__(self):
        super().__init__('Image_detection')
        self.model = 
YOLO("/home/sunrise/ultralytics/ultralytics/data/yahboom_data/orange_data/runs/d
etect/train/weights/best.onnx") #这里的路径是个人训练模型的路径 The path here is the
path of the personal training model
        self.camera_type = os.getenv('CAMERA_TYPE', 'usb')
        self.bridge = CvBridge()
        if self.camera_type == 'usb':
            topic_name = '/usb_cam/image_raw'
        else:
            topic_name = '/ascamera_hp60c/camera_publisher/rgb0/image'

        self.subscription = self.create_subscription(Image,topic_name,
self.image_callback,10)

        # Get the video frame size and frame rate
        frame_width = 640
        frame_height = 480
        fps = 15
        output_path =
"/home/sunrise/ultralytics/ultralytics/output/06.orange_camera_usb.mp4"
        fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # You can use 'XVID' or 'mp4v'
depending on your platform
        self.out = cv2.VideoWriter(output_path, fourcc, fps, (frame_width,
frame_height))

    def image_callback(self, msg):
        cv_image = self.bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')

        self.proecc(cv_image)

# Loop through the video frames
    def proecc(self,frame):
        # Run YOLO inference on the frame
        results = self.model(frame)

        # Visualize the results on the frame
        annotated_frame = results[0].plot()

        # Write the annotated frame to the output video file
        self.out.write(annotated_frame)


        # Display the annotated frame
        cv2.imshow("YOLO Inference", cv2.resize(annotated_frame, (640, 480)))

        # Break the loop if 'q' is pressed
        cv2.waitKey(1) & 0xFF == ord("q")

    def cancel(self):
        cv2.destroyAllWindows()
        self.out.release()

def main(args=None):
    rclpy.init(args=args)
    node = Image_detection()
    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        pass
    finally:
        node.cancel()
        node.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```