Oriented object detection.md

Note: Using the Docker container in the factory image does not require re-setting up the environment. The environment is already set up. Simply enter Docker and run the corresponding function commands according to the previous tutorial.

1. Orienting Objects: Images

Use yolo11n-obb.pt to predict images provided with the Ultralytics project.

Go to the code folder:

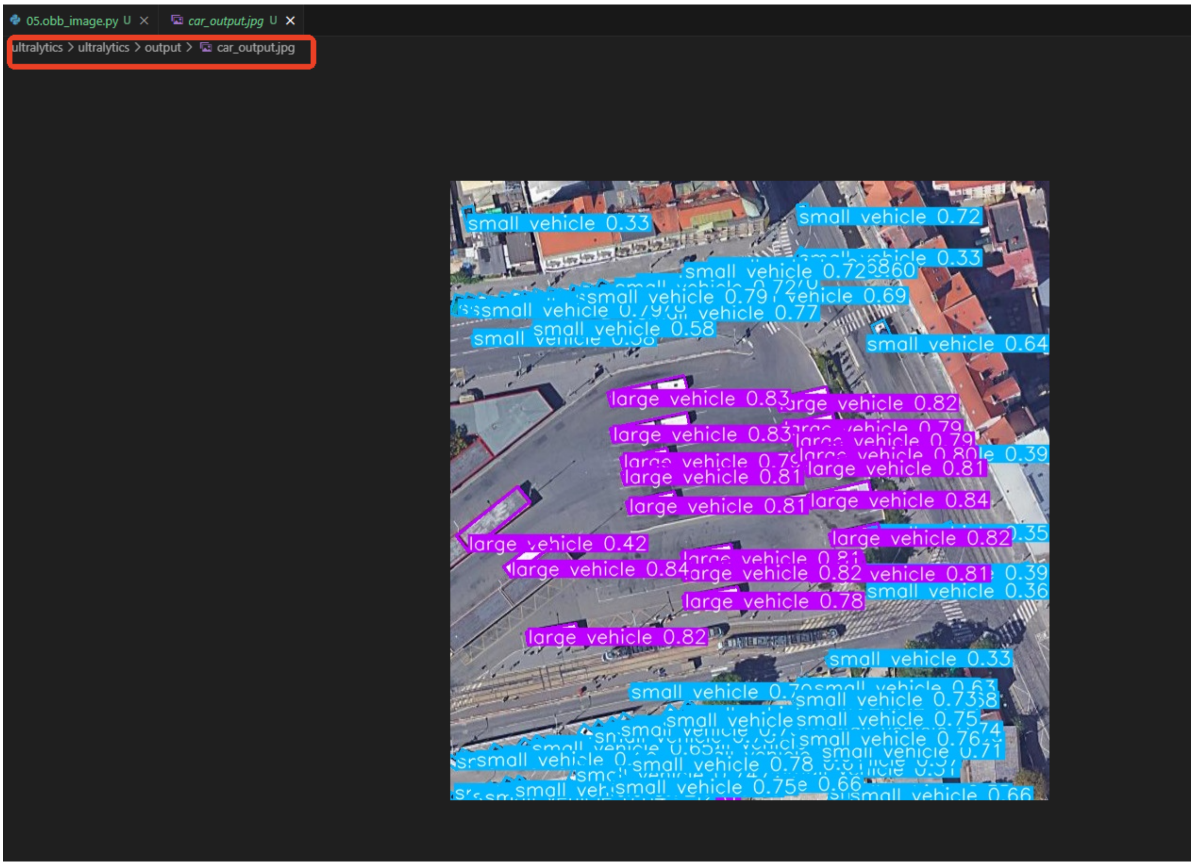
```
cd /root/ultralytics/ultralytics/yahboom_demo
```

Run the code:

```
python3 05.obb_image.py
```

Preview

Yolo recognition output image location: /root/ultralytics/ultralytics/output/



Sample code:

```
from ultralytics import YOLO

# Load a model
model = YOLO("/root/ultralytics/ultralytics/yolo11n-obb.pt")
# model = YOLO("/root/ultralytics/ultralytics/yolo11n-obb.onnx")
```

```
# model = YOLO("/root/ultralytics/ultralytics/yolo11n-obb_ncnn_model")

# Run batched inference on a list of images
results = model("/root/ultralytics/ultralytics/assets/car.jpg")  # return a list
of Results objects

# Process results list
for result in results:
    boxes = result.boxes  # Boxes object for bounding box outputs
    # masks = result.masks  # Masks object for segmentation masks outputs
    # keypoints = result.keypoints  # Keypoints object for pose outputs
    # probs = result.probs  # Probs object for classification outputs
    obb = result.obb  # Oriented boxes object for OBB outputs
    result.show()  # display to screen
    result.save(filename="/root/ultralytics/ultralytics/output/car_output.jpg")
 # save to disk
```

2. Oriented Object Detection: Video

Use yolo11n-obb.pt to predict videos in the Ultralytics project (not included with Ultralytics).

Go to the code folder:

```
cd /root/ultralytics/ultralytics/yahboom_demo
```

Run the code:

```
python3 02.segmentation_video.py
```

Preview

Yolo recognition output video location: /root/ultralytics/ultralytics/output/

Sample code:

```python
import cv2
from ultralytics import YOLO

# Load the YOLO model
model = YOLO("/root/ultralytics/ultralytics/yolo11n-obb.pt")
# model = YOLO("/root/ultralytics/ultralytics/yolo11n-obb.onnx")
# model = YOLO("/root/ultralytics/ultralytics/yolo11n-obb_ncnn_model")

# Open the video file
video_path = "/root/ultralytics/ultralytics/videos/street.mp4"
cap = cv2.VideoCapture(video_path)

# Get the video frame size and frame rate
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(cap.get(cv2.CAP_PROP_FPS))

# Define the codec and create a VideoWriter object to output the processed video
output_path = "/root/ultralytics/ultralytics/output/05.street_output.mp4"
fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # You can use 'XVID' or 'mp4v'
depending on your platform
out = cv2.VideoWriter(output_path, fourcc, fps, (frame_width, frame_height))

# Loop through the video frames
while cap.isOpened():
    # Read a frame from the video
    success, frame = cap.read()

    if success:
        # Run YOLO inference on the frame
        results = model(frame)

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

        # Write the annotated frame to the output video file
        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
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    else:
        # Break the loop if the end of the video is reached
        break
# Release the video capture and writer objects, and close the display window
cap.release()
out.release()
cv2.destroyAllWindows()
```

3. Directed Object Detection: Real-Time Detection

3.1. Launching the Camera

Launch 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:

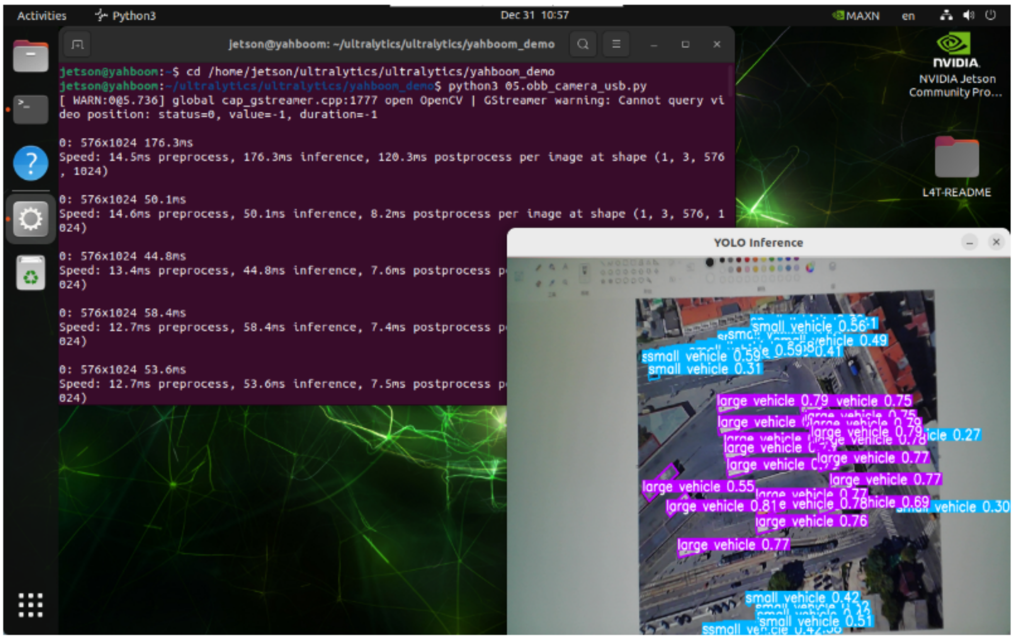
```
cd /root/ultralytics/ultralytics/yahboom_demo
```

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

```
python3 05.obb_camera_usb.py
```

Preview

Yolo recognition output video location: /root/ultralytics/ultralytics/output/



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')
```

```python
        self.model = model = YOLO("/root/ultralytics/ultralytics/yolo11n-obb.pt")
        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 =
"/root/ultralytics/ultralytics/output/05.obb_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:
```

```python
        node.cancel()
        node.destroy_node()
        rclpy.shutdown()

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