Oriented object detection

Use Python to demonstrate the effects of **Ultralytics**: Oriented Bounding Boxes Object Detection in images, videos, and real-time detection.

1. Enable optimal performance of the motherboard

1.1. Enable MAX power mode

Enabling MAX Power Mode on Jetson will ensure that all CPU and GPU cores are turned on:

```
sudo nvpmodel -m 0
```

1.2. Enable Jetson clocks

Enabling Jetson Clocks will ensure that all CPU and GPU cores run at maximum frequency:

```
sudo jetson_clocks
```

2. Oriented object detection: image

Use yolo11n-obb.pt to predict images under the ultralytics project (not ultralytics built-in images).

Enter the code folder:

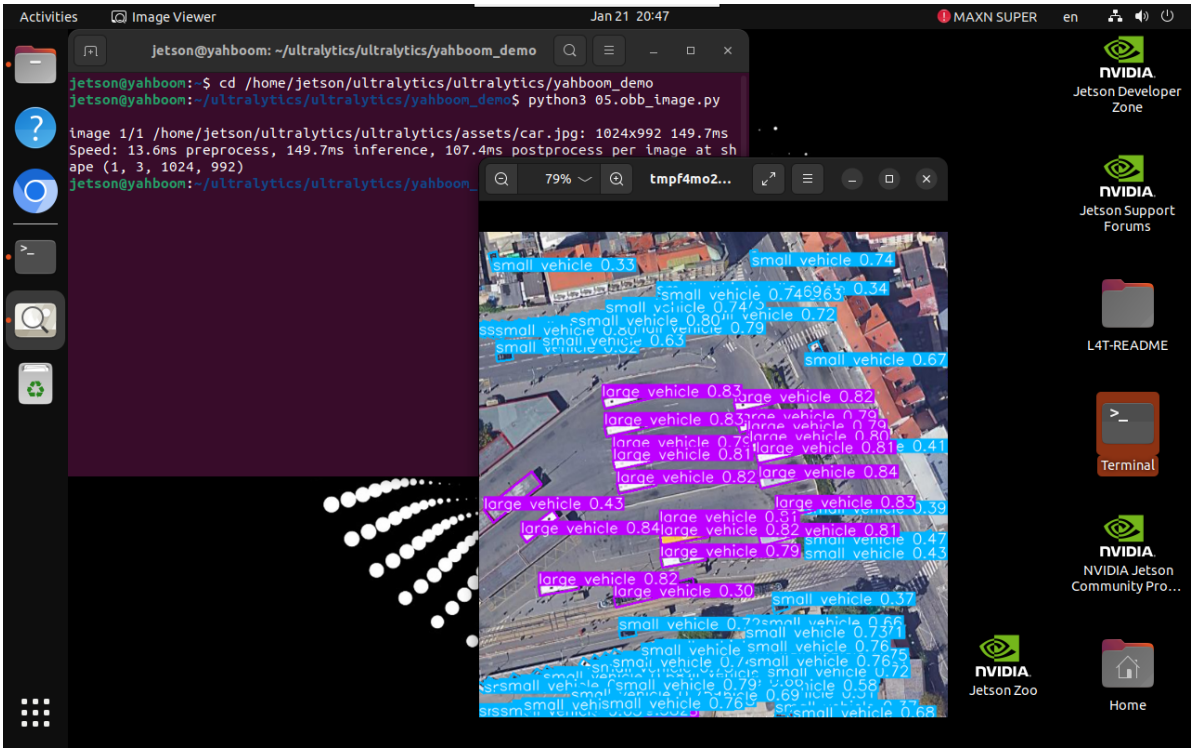
```
cd /home/jetson/ultralytics/ultralytics/yahboom_demo
```

Run the code:

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

Effect preview

Yolo recognition output image location：/home/jetson/ultralytics/ultralytics/output/



Sample code:

```
from ultralytics import YOLO

# Load a model
model = YOLO("/home/jetson/ultralytics/ultralytics/yolo11n-obb.pt")

# Run batched inference on a list of images
results = model("/home/jetson/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="/home/jetson/ultralytics/ultralytics/output/car_output.jp
g")  # save to disk
```

3. Directed object detection: video

Use yolo11n-obb.pt to predict the video under the ultralytics project (not the video that comes with ultralytics).

Enter the code folder:

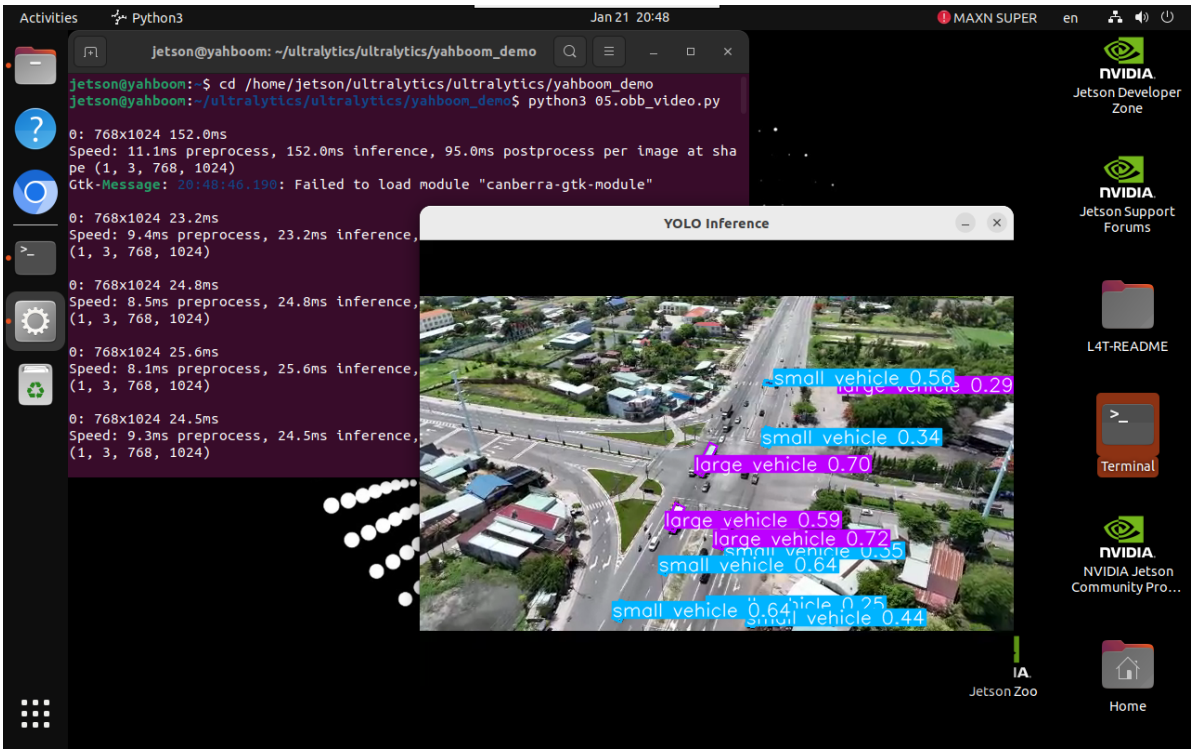
```
cd /home/jetson/ultralytics/ultralytics/yahboom_demo
```

Run the code:

```
python3 05.obb_video.py
```

Effect preview

Yolo identifies the output video position：/home/jetson/ultralytics/ultralytics/output/



Sample code:

```
import cv2
from ultralytics import YOLO

# Load the YOLO model
model = YOLO("/home/jetson/ultralytics/ultralytics/yolo11n-obb.pt")

# Open the video file
video_path = "/home/jetson/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 = "/home/jetson/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()
```

4. Directed object detection: real-time detection

4.1. USB camera

Use yolo11n-obb.pt to predict the USB camera screen.

Enter the code folder:

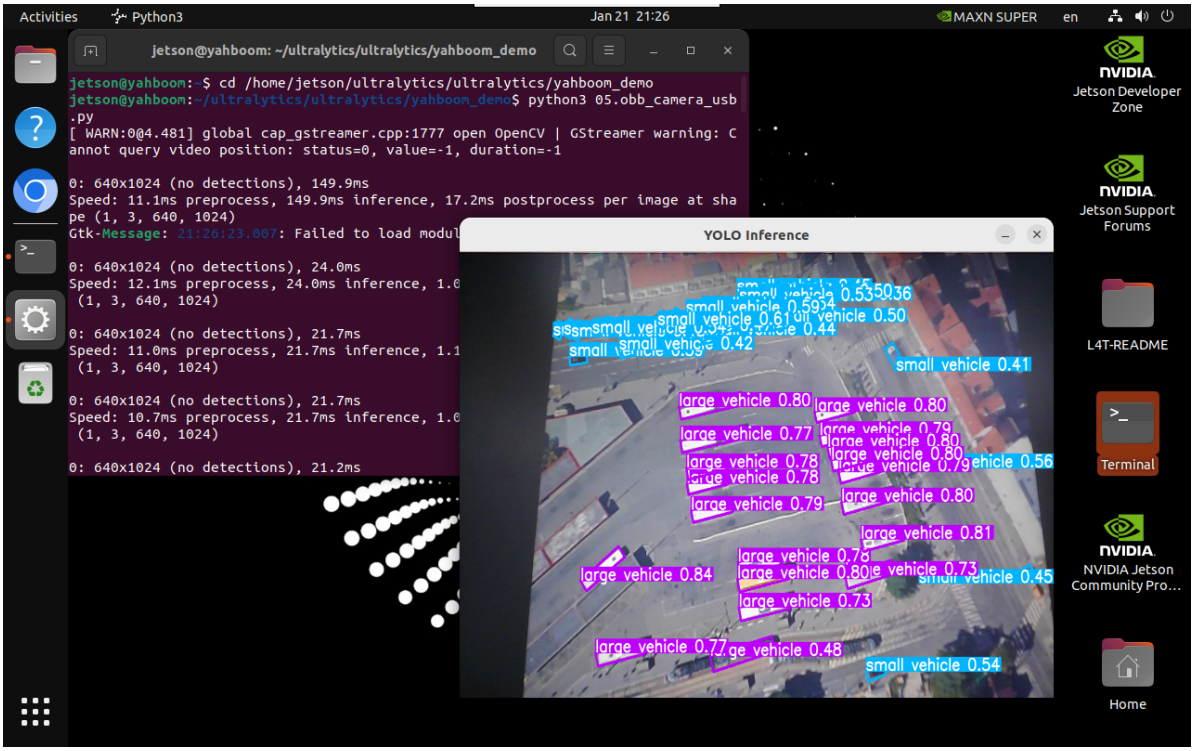
```
cd /home/jetson/ultralytics/ultralytics/yahboom_demo
```

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

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

Effect preview

Yolo identifies the output video position：/home/jetson/ultralytics/ultralytics/output/



Sample code:

```
import cv2
from ultralytics import YOLO

# Load the YOLO model
model = YOLO("/home/jetson/ultralytics/ultralytics/yolo11n-obb.pt")

# Open the cammera
cap = cv2.VideoCapture(0)

# 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 =
"/home/jetson/ultralytics/ultralytics/output/05.obb_camera_usb.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()
```

4.2, CSI camera

Use yolo11n-obb.pt to predict the CSI camera image.

Enter the code folder:

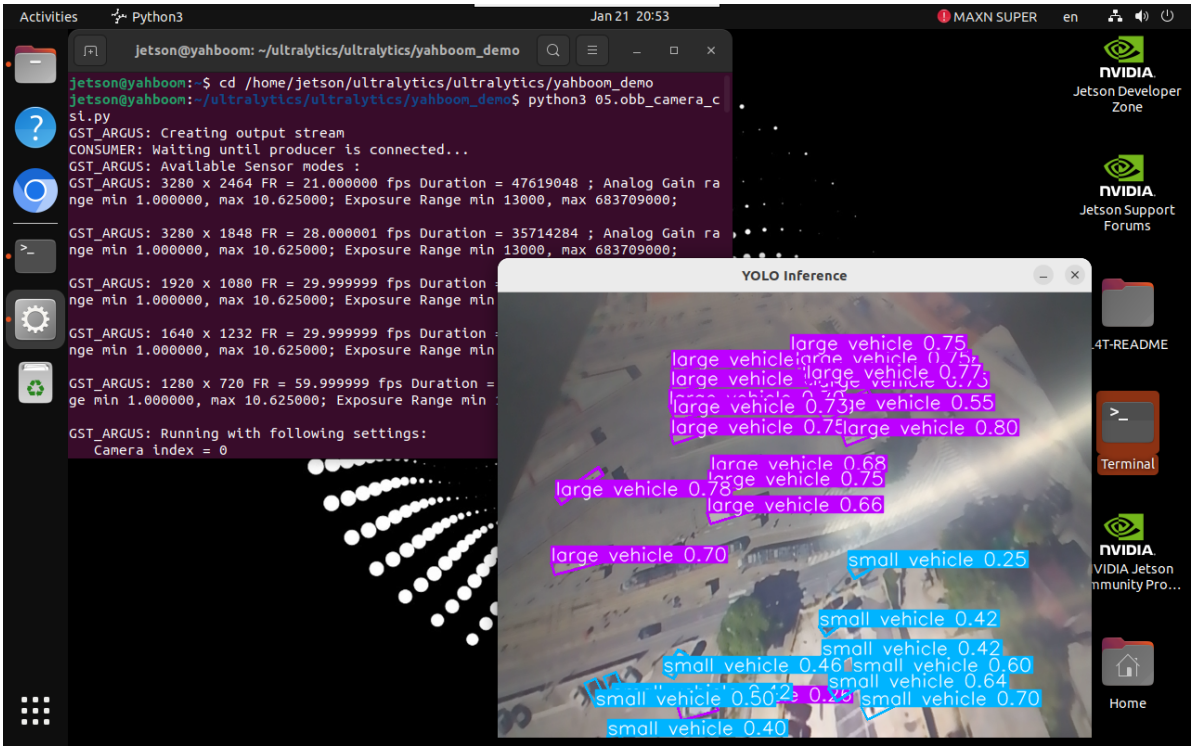
```
cd /home/jetson/ultralytics/ultralytics/yahboom_demo
```

Run the code: Click the preview image, press the q key to terminate the program!

```
python3 05.obb_camera_csi.py
```

Effect preview

Yolo identifies the output video position：/home/jetson/ultralytics/ultralytics/output/



Sample code:

```
import cv2
```

```
from ultralytics import YOLO
from jetcam.csi_camera import CSICamera

# Load the YOLO model
model = YOLO("/home/jetson/ultralytics/ultralytics/yolo11n-obb.pt")

# Open the camera (CSI Camera)
cap = CSICamera(capture_device=0, width=640, height=480)

# Get the video frame size and frame rate
frame_width = 640
frame_height = 480
fps = 30

# Define the codec and create a VideoWriter object to output the processed video
output_path =
"/home/jetson/ultralytics/ultralytics/output/05.obb_camera_csi.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 True:
    # Read a frame from the camera
    frame = cap.read()

    if frame is not None:
        # 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 no frame is received (camera error or end of stream)
        print("No frame received, breaking the loop.")
        break

# Release the video capture and writer objects, and close the display window
cap.release()
out.release()
cv2.destroyAllWindows()
```

References

https://docs.ultralytics.com/tasks/obb/