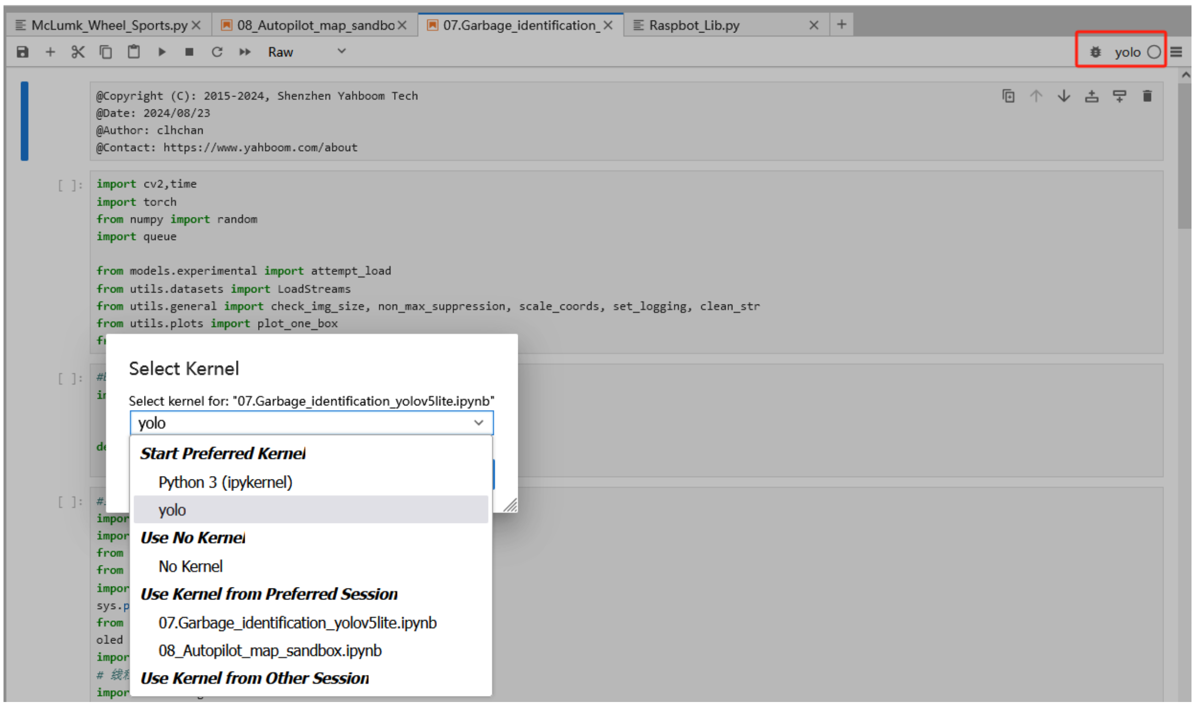
7. Garbage Identification

7. Garbage Identification
- 7.2. Experimental code
- 7.3. Experimental Phenomenon

- ## 7.1. Experimental objectives
  - In this chapter, we use the yolov5-lite model to realize the recognition of garbage images on building blocks

  Before use, switch the kernel environment of jupyter lab to **yolo environment**



McLumk_Wheel_Sports.py
08_Autopilot_map_sandbo
07.Garbage_identification_
Raspbot_Lib.py
Raw
yolo
Start Preferred Kernel
Python 3 (ipykernel)
yolo
Use No Kernel
No Kernel
Use Kernel from Preferred Session
07.Garbage_identification_yolov5lite.ipynb
08_Autopilot_map_sandbox.ipynb
Use Kernel from Other Session

```
@Copyright (C): 2015-2024, Shenzhen Yahboom Tech
@Date: 2024/08/23
@Author: clhchan
@Contact: https://www.yahboom.com/about
```

```
import cv2,time
import torch
from numpy import random
import queue

from models.experimental import attempt_load
from utils.datasets import LoadStreams
from utils.general import check_img_size, non_max_suppression, scale_coords, set_logging, clean_str
from utils.plots import plot_one_box
```

Select Kernel

Select kernel for: "07.Garbage_identification_yolov5lite.ipynb"

7.2. Experimental code

Source code path:

/home/pi/project_demo/08.AI_Visual_Interaction_Course/07.Garbage_identification_yolov5lite/07.Garbage_identification_yolov5lite.ipynb

```
import cv2,time
import torch
from numpy import random
import queue

from models.experimental import attempt_load
from utils.datasets import LoadStreams
from utils.general import check_img_size, non_max_suppression, scale_coords,
set_logging, clean_str
from utils.plots import plot_one_box
from utils.torch_utils import select_device, time_synchronized
```

```python
#bgr8转jpeg格式 bgr8 to jpeg format
import enum


def bgr8_to_jpeg(value, quality=75):
    return bytes(cv2.imencode('.jpg', value)[1])
```

```python
#显示摄像头组件 Display camera components
import traitlets
import ipywidgets.widgets as widgets
from IPython.display import display
from Raspbot_Lib import Raspbot
import sys
sys.path.append('/home/pi/software/oled_yahboom/')
from yahboom_oled import *
oled = Yahboom_OLED(debug=False)
import time
# 线程功能操作库 Thread function operation library
import threading
import inspect
import ctypes

image_widget = widgets.Image(format='jpeg', width=640, height=480)
```

```python
#初始化oled进程 Initialize oled process
oled.init_oled_process()
oled.clear()
oled.add_line("garbage_type:", 1)
oled.add_line("None", 3)
oled.refresh()
#复位舵机 reset servo
bot = Raspbot()
def servo_reset():
    bot.Ctrl_Servo(1,90)
    bot.Ctrl_Servo(2,25)
servo_reset()
```

```python
def _async_raise(tid, exctype):
    """raises the exception, performs cleanup if needed"""
    tid = ctypes.c_long(tid)
    if not inspect.isclass(exctype):
        exctype = type(exctype)
    res = ctypes.pythonapi.PyThreadState_SetAsyncExc(tid,
ctypes.py_object(exctype))
    if res == 0:
        raise ValueError("invalid thread id")
    elif res != 1:
        # """if it returns a number greater than one, you're in trouble,
        # and you should call it again with exc=NULL to revert the effect"""
        ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)

def stop_thread(thread):
    _async_raise(thread.ident, SystemExit)
```

```python
def detect(weights='weights/gabarge.pt', source='0', img_size=320,
conf_thres=0.45, iou_thres=0.35, device=''):
    global classes
    label=None
    # Initialize
    set_logging()
    device = select_device(device)
    half = device.type != 'cpu'  # half precision only supported on CUDA

    # Load model
    model = attempt_load(weights, map_location=device)  # load FP32 model
    stride = int(model.stride.max())  # model stride
    imgsz = check_img_size(img_size, s=stride)  # check img_size
    if half:
        model.half()  # to FP16

    # Set Dataloader
    dataset = LoadStreams(source, img_size=imgsz, stride=stride)

    # Get names and colors
    names = model.module.names if hasattr(model, 'module') else model.names
    colors = [[random.randint(0, 255) for _ in range(3)] for _ in names]

    frame_count = 0
    start_time = time.time()

    # Run inference
    if device.type != 'cpu':
        model(torch.zeros(1, 3, imgsz,
imgsz).to(device).type_as(next(model.parameters())))  # run once
    for path, img, im0s, vid_cap in dataset:
        img = torch.from_numpy(img).to(device)
        img = img.half() if half else img.float()  # uint8 to fp16/32
        img /= 255.0  # 0 - 255 to 0.0 - 1.0
        if img.ndimension() == 3:
            img = img.unsqueeze(0)

        # Calculate FPS before processing the frame
        frame_count += 1
        elapsed_time = time.time() - start_time
        fps = frame_count / elapsed_time

        # Display FPS on the frame
        cv2.putText(im0s[0], f"FPS: {fps:.2f}", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

        # Inference
        pred = model(img, augment=False)[0]

        # Apply NMS
        pred = non_max_suppression(pred, conf_thres, iou_thres)

        # Process detections
        for i, det in enumerate(pred):  # detections per image
            if det is not None and len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_coords(img.shape[2:], det[:, :4],
im0s[i].shape).round()

                # Print results
                for *xyxy, conf, cls in reversed(det):
                    label = f'{names[int(cls)]} {conf:.2f}'
                    classes = f'{names[int(cls)]}'
                    #classes = f'{names[int(cls)]}'
                    #print(classes)  # 打印识别的标签和置信度 Print the recognized
labels and confidence
                    plot_one_box(xyxy, im0s[i], label=label,
color=colors[int(cls)], line_thickness=3)
        oled.clear()
        oled.add_line("garbage_type:", 1)
        if label!=None:
            oled.add_line(label, 3)
            label=None
        else:
            oled.add_line("None", 3)
        oled.refresh()
        # Stream results
        #cv2.imshow('Video0', im0s[0])
        global image_widget
        image_widget.value = bgr8_to_jpeg(im0s[0])

        # if cv2.waitKey(1) == ord('q'):  # Press 'q' to quit
        #     break

    #cv2.destroyAllWindows()
```

```
display(image_widget)

thread1 = threading.Thread(target=detect)

thread1.daemon=True
thread1.start()
```

```
# Please place the block in the center
# 请将积木块正放在中心
```

7.3. Experimental Phenomenon

After the code block is run, the servo will face down, put the garbage building block in the camera screen, and then it will start to identify the garbage. At the same time, the OLED will also display the name of the identified garbage.

Note: Because the camera is greatly affected by the ambient light, it may not be able to recognize. When this phenomenon occurs, you can change to a more brightly lit environment, illuminate the building block, or set up a car and let the car look down at the building block.