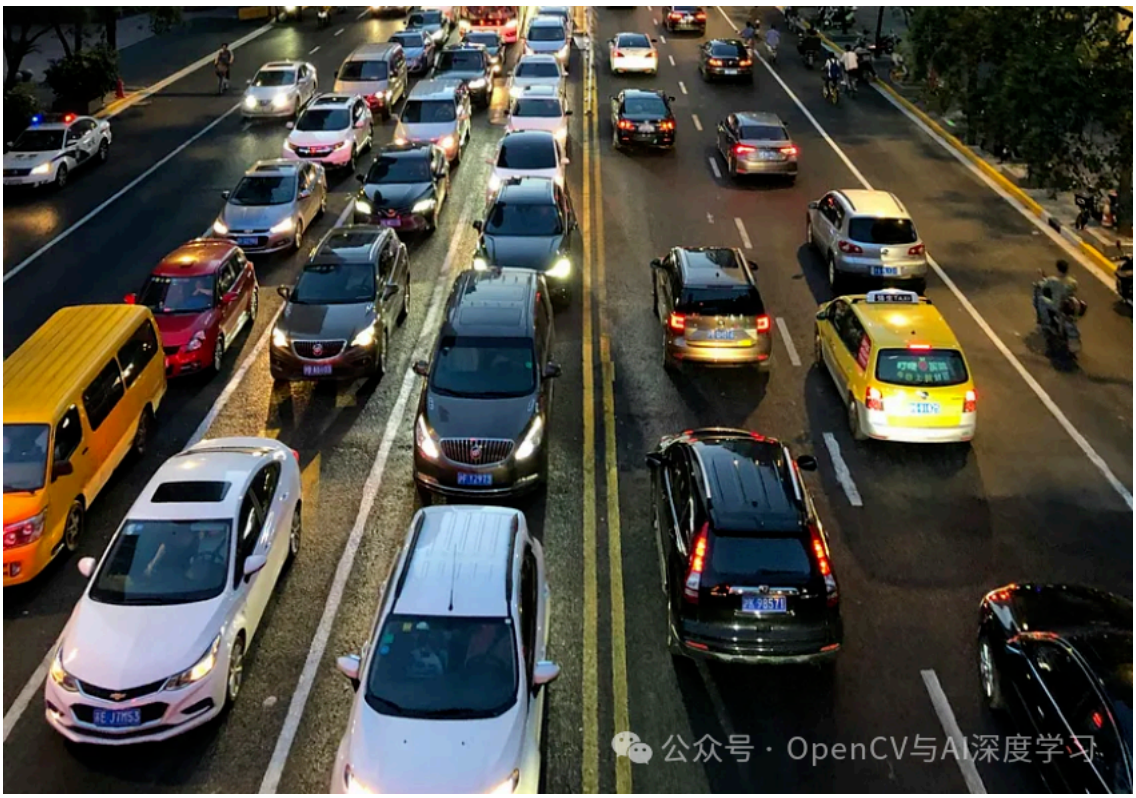
实战 | 基于YOLOv9和OpenCV实现车辆跟踪计数（步骤 + 源码）

原创 Color Space OpenCV与AI深度学习 2024-03-13 07:45 重庆

导　读

本文主要介绍使用YOLOv9和OpenCV实现车辆跟踪计数（步骤 + 源码）。

实现步骤



监控摄像头可以有效地用于各种场景下的车辆计数和交通流量统计。先进的计算机视觉技术（例如对象检测和跟踪）可应用于监控录像，以识别和跟踪车辆在摄像机视野中移动。

【1】安装ultralytics，因为它拥有直接使用 YoloV9 预训练模型的方法。

```
pip install ultralytics
```

【2】完成后，就可以创建跟踪器函数来跟踪对象了。我们只是为此创建了一个名为tracker.py的python文件。

```
import math

class CustomTracker:
    def __init__(self):
        # Store the center positions of the objects
        self.custom_center_points = {}
        # Keep the count of the IDs
        # each time a new object id detected, the count will increase by c
        self.custom_id_count = 0

    def custom_update(self, custom_objects_rect):
        # Objects boxes and ids
        custom_objects_bbs_ids = []

        # Get center point of new object
        for custom_rect in custom_objects_rect:
            x, y, w, h = custom_rect
            cx = (x + x + w) // 2
            cy = (y + y + h) // 2
```

```
            # Find out if that object was detected already
            same_object_detected = False
            for custom_id, pt in self.custom_center_points.items():
                dist = math.hypot(cx - pt[0], cy - pt[1])

                if dist < 35:
                    self.custom_center_points[custom_id] = (cx, cy)
                    custom_objects_bbs_ids.append([x, y, w, h, custom_id])
                    same_object_detected = True
                    break

            # New object is detected we assign the ID to that object
            if same_object_detected is False:
                self.custom_center_points[self.custom_id_count] = (cx, cy)
                custom_objects_bbs_ids.append([x, y, w, h, self.custom_id_
                self.custom_id_count += 1

        # Clean the dictionary by center points to remove IDS not used any
        new_custom_center_points = {}
        for custom_obj_bb_id in custom_objects_bbs_ids:
            _, _, _, _, custom_object_id = custom_obj_bb_id
            center = self.custom_center_points[custom_object_id]
            new_custom_center_points[custom_object_id] = center

        # Update dictionary with IDs not used removed
        self.custom_center_points = new_custom_center_points.copy()
        return custom_objects_bbs_ids
```

【3】编写车辆计数的主要代码。

```
# Import the Libraries
import cv2
import pandas as pd
from ultralytics import YOLO
from tracker import *
```

导入所有必要的库后，就可以导入模型了。我们不必从任何存储库下载模型。Ultralytics 做得非常出色，让我们可以更轻松地直接下载它们。

```
model=YOLO('yolov9c.pt')
```

这会将 yolov9c.pt 模型下载到当前目录中。该模型已经在由 80 个不同类别组成的 COCO 数据集上进行了训练。现在让我们指定类：

```
class_list = ['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus',
              'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 's
```

```
'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 's
'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elep
'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbe
'baseball bat', 'baseball glove', 'skateboard', 'surfboard',
'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwic
'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant', '
'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'ov
'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toc
```

现在，下一步是加载您要使用的视频。

```
tracker=CustomTracker()
count=0

cap = cv2.VideoCapture('traffictrim.mp4')

# Get video properties
fps = int(cap.get(cv2.CAP_PROP_FPS))
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

# Create VideoWriter object to save the modified frames
output_video_path = 'output_video.mp4'
fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # You can use other codecs like
out = cv2.VideoWriter(output_video_path, fourcc, fps, (width, height))
```

在这里，我们在加载视频后获取视频属性，因为它们对于使用计数器重新创建视频并最终将其存储在本地非常有用。

```
# Looping over each frame and Performing the Detection

down = {}
counter_down = set()
while True:
    ret, frame = cap.read()
    if not ret:
        break
    count += 1

    results = model.predict(frame)

    a = results[0].boxes.data
    a = a.detach().cpu().numpy()
    px = pd.DataFrame(a).astype("float")
    # print(px)

```

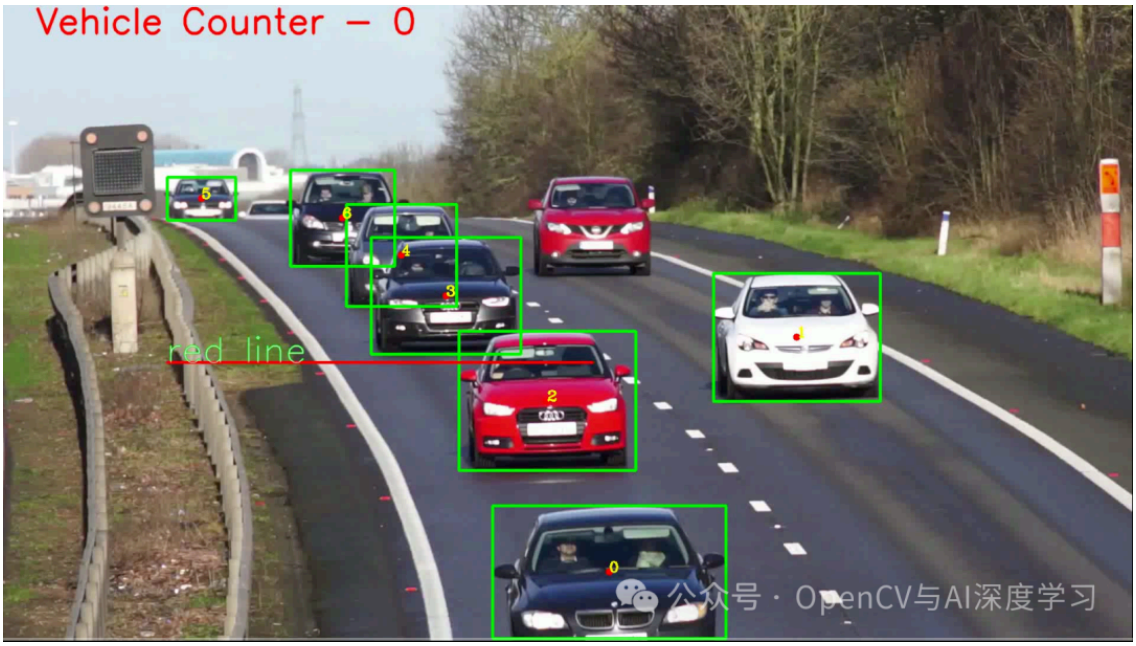
```
    list = []

    for index, row in px.iterrows():
        #        print(row)
        x1 = int(row[0])
        y1 = int(row[1])
        x2 = int(row[2])
        y2 = int(row[3])
        d = int(row[5])
        c = class_list[d]
        if 'car' in c:
            list.append([x1, y1, x2, y2])

    bbox_id = tracker.custom_update(list)
    # print(bbox_id)
    for bbox in bbox_id:
        x3, y3, x4, y4, id = bbox
        cx = int(x3 + x4) // 2
        cy = int(y3 + y4) // 2
        # cv2.circle(frame,(cx,cy),4,(0,0,255),-1) #draw ceter points of b
        # cv2.rectangle(frame, (x3, y3), (x4, y4), (0, 255, 0), 2)  # Draw
        # cv2.putText(frame,str(id),(cx,cy),cv2.FONT_HERSHEY_COMPLEX,0.8,(

        y = 308
        offset = 7

        ''' condition for red line '''
        if y < (cy + offset) and y > (cy - offset):
            ''' this if condition is putting the id and the circle on the

            down[id] = cy  # cy is current position. saving the ids of the
            # This will tell us the travelling direction of the car.
            if id in down:
                cv2.circle(frame, (cx, cy), 4, (0, 0, 255), -1)
                #cv2.putText(frame, str(id), (cx, cy), cv2.FONT_HERSHEY_CO
                counter_down.add(id)

                # # line
    text_color = (255, 255, 255)  # white color for text
    red_color = (0, 0, 255)  # (B, G, R)

    # print(down)
    cv2.line(frame, (282, 308), (1004, 308), red_color, 3)  # starting cor
    cv2.putText(frame, ('red line'), (280, 308), cv2.FONT_HERSHEY_SIMPLEX,


    downwards = (len(counter_down))
```

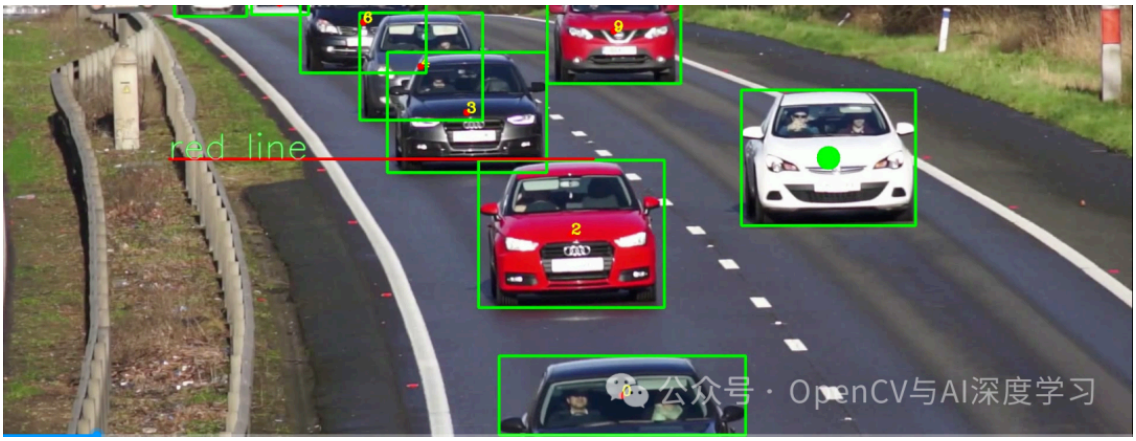
```
    downwards = (len(counter_down))
    cv2.putText(frame, ('Vehicle Counter - ') + str(downwards), (60, 40),
                cv2.LINE_AA)

    cv2.line(frame,(282,308),(1004,308),red_color,3)  #  starting cordinat
    cv2.putText(frame,('red line'),(280,308),cv2.FONT_HERSHEY_SIMPLEX, 0.5

    # This will write the Output Video to the location specified above
    out.write(frame)
```

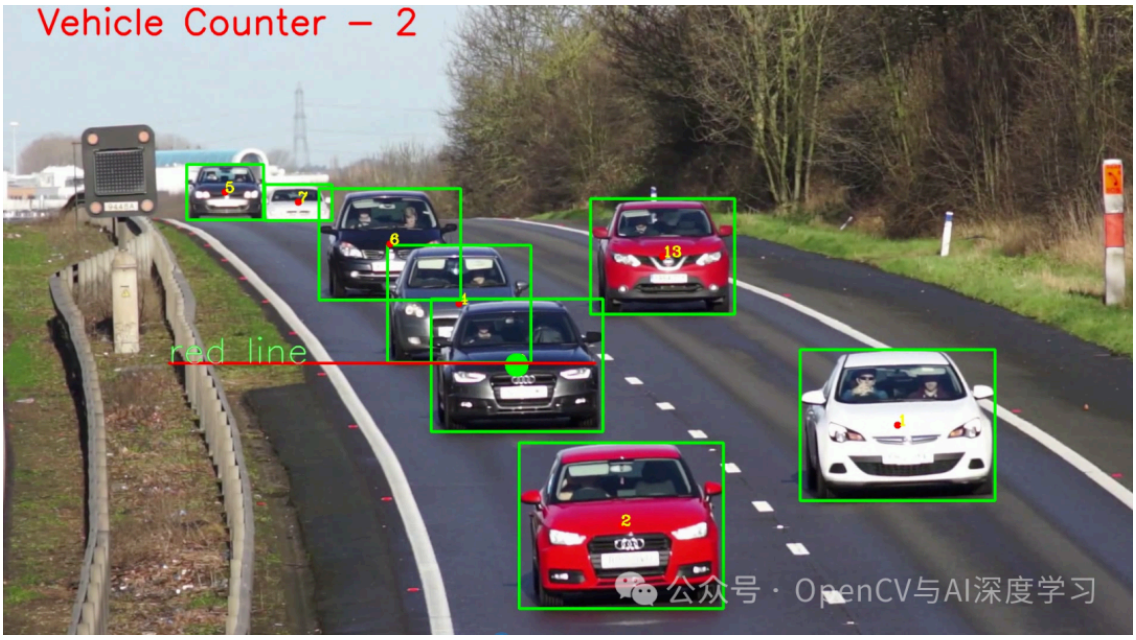
在上面的代码中，我们循环遍历视频中的每个帧，然后进行检测。然后，由于我们仅对车辆进行计数，因此仅过滤掉汽车的检测结果。

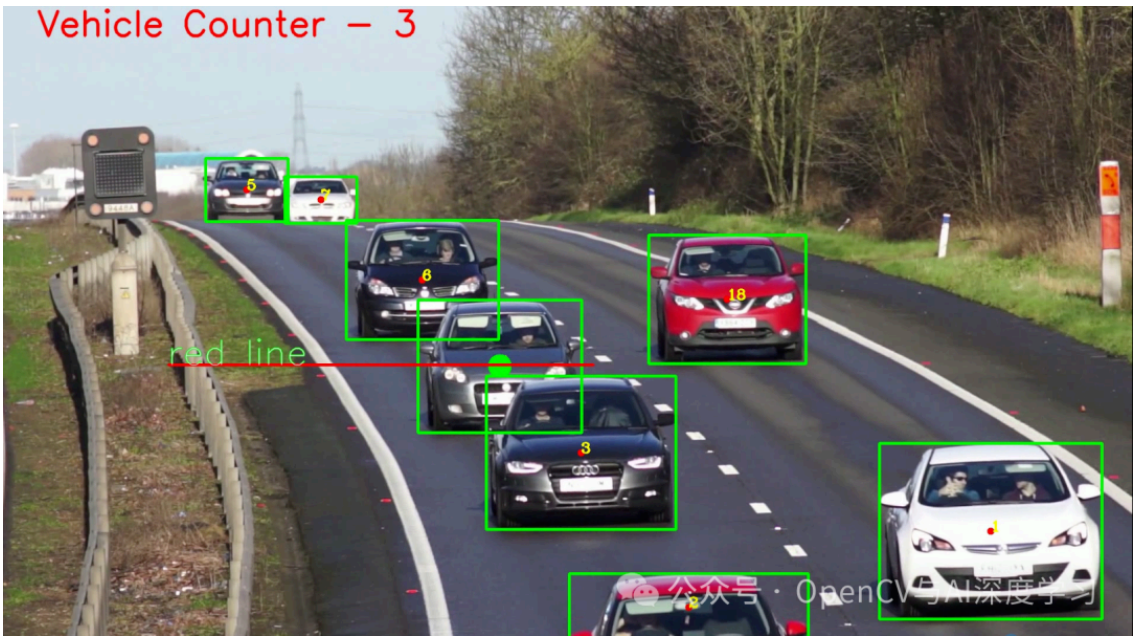
之后，我们找到检测到的车辆的中心，然后在它们穿过人工创建的红线时对它们进行计数。我们可以在下面的视频快照中清楚地看到它们。

```
0: 384x640 6 persons, 8 cars, 1 traffic light, 29.5ms
Speed: 1.0ms preprocess, 29.5ms inference, 2.1ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 persons, 8 cars, 1 traffic light, 29.5ms
Speed: 2.5ms preprocess, 29.5ms inference, 1.3ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 3 persons, 8 cars, 1 truck, 1 traffic light, 27.7ms
Speed: 1.2ms preprocess, 27.7ms inference, 2.1ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 persons, 8 cars, 1 traffic light, 32.6ms
Speed: 2.0ms preprocess, 32.6ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 persons, 8 cars, 1 traffic light, 29.9ms
Speed: 2.9ms preprocess, 29.9ms inference, 2.3ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 persons, 8 cars, 1 traffic light, 29.1ms
Speed: 2.0ms preprocess, 29.1ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 persons, 7 cars, 1 traffic light, 27.3ms
Speed: 2.2ms preprocess, 27.3ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 5 persons, 7 cars, 1 traffic light, 29.9ms
Speed: 2.0ms preprocess, 29.9ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)
```


red_line
公众号 · OpenCV与AI深度学习


Vehicle Counter – 2
red_line
公众号 · OpenCV与AI深度学习


Vehicle Counter – 3
red_line
公众号 · OpenCV与AI深度学习

00:02

我们可以看到，当车辆越过红线时，视频左上角的计数器不断增加。

—THE END—

下载1：Pytorch常用函数手册

在「**OpenCV与AI深度学习**」公众号后台回复：**Pytorch函数手册**，即可下载学习全网第一份Pytorch函数常用手册，包括Tensors**介绍、基础函数介绍、数据处理函数、优化函数、CUDA编程、多处理**等十四章内容。

下载2：145个OpenCV实例应用代码

在「**OpenCV与AI深度学习**」公众号后台回复：**OpenCV145**，即可下载学习145个OpenCV实例应用代码（**Python和C++双语言实现**）。

欢迎加入**CV**学习交流微信群！

觉得有用，记得点个赞和在看

喜欢此内容的人还喜欢

实战｜OpenCV实时弯道检测(详细步骤+源码)

OpenCV与AI深度学习

基于OpenCV实现模糊检测 / 自动对焦

OpenCV与AI深度学习

实战 | 用Python和OpenCV搭建一个老人跌倒智能监测系统 (步骤 + 源码)

OpenCV与AI深度学习