使用OpenCV搭建違章停車檢測系統

小白 AI算法與圖像處理 2022-02-24 17:21

各位小伙伴大家好，今天將會帶領大家一起學習如何搭建一個違章停車檢測系統。需要重點說明的是，今天使用的邏輯和判定條件比較難，尤其是他的編程實現。不過小伙伴不要怕，我們提供了項目的開源代碼，具體鏈接如下：

> https://github.com/hasantha-nirmal/Traffic_Violation_Detection_Yolov4_Deep-Sort

接下來我們將詳細介紹如何實現這個系統

首先，我們需要簡要了解一下這個項目中的停車違章檢測是什麼意思。為了介紹方便，我們將穿插使用學術術語和生活用語。想像一下，我們有一個感興趣的區域也可以稱為某個地方，車輛不應該停放在那裡。但是，如果車輛只是停在那裡一會兒，我們也不能將其是為為違章，例如馬路邊允許臨時停車的區域。如果我們正在談論的這個區域離道路太近，那麼經過的車輛可能會與這個感興趣的區域相交，並且會立即被視為違章。所以首先，我們必須消除這種情況。也就是首先解決臨時停車不會判定為違章的情況。最簡單的解決方案是為每輛單獨的車輛引入計時器。由此，我們可以確保車輛已經在該限制區域內停留了多長時間。

採用感興趣區域作為停車位，我們只需使用一個攝像頭即可，同時這個攝像頭不僅可以同時監控和檢測特定車輛佔用該停車位的時間，還可以計算出每輛車必須支付的費用停車場。當然了，這需要額外的開發，這裡小白做過多的介紹

我們前面說過，我們需要建立一個計時器。這個任務的難點是需要考慮許多車輛肯定會同時出現，這使得任務變得比較複雜。不過也沒有關係，我們只需要為不同的車輛設置不同的計時器即可。

接下來我們將詳細介紹如何通過代碼實現上述的功能。

```
# Parking space coordinates; #line 113
parking_co = []

#blanked = np.zeros((658,1024), dtype=np.uint8)
blanked = np.zeros((2048, 1024), dtype=np.uint8)
#pts = np.array(([156, 704], [2, 893], [476, 932], [270, 708]))
pts = np.array(([513, 716], [321, 943], [884, 979], [630, 701]))
#blanked = np.zeros((720,1280), dtype=np.uint8)
#pts = np.array(([38, 433], [95, 322], [1246, 570], [1065, 709]))
cv2.fillPoly(blanked, np.int32([pts]), 255)
```

上面的代碼給出了我們如何選擇停車位作為我們的感興趣區域。我們首先定義了一個名為park_co的空白數組，之後創建了一個於圖像分辨率具有相同高度或者相同寬度的一個權威零的數組。之後選擇感興趣區域的頂點坐標。在pts變量中存放我們選擇的感興趣區域的頂點坐標。之後我們使用OpenCV中的fillPoly函數將感興趣區域填充上，以便於我們判斷車輛是否與感興趣區域相交。

感興趣區域的選擇如下圖所示

现在，我们有了感兴趣的区域或禁止车辆停放的地方的像素的所有坐标点。然后我们选取车辆的边界框坐标（如何识别车辆呢，可以参考小白之前的文章）。但是，这又带来了一个问题。如果相机离这个感兴趣区域太近，当有车辆接近该区域时，它的边界框会占据非常多的坐标点，当同时有车辆时，必须对视频的每一帧重复这个过程，导致帧率急剧下降。所以，我对这个案例提出了一个假设：如果一个车辆/边界框与这个 ROI 相交，它肯定也与边界框的底线相交。所以就像在车道线违例中一样，而不是取车辆的所有边界框坐标

```
bbox_bottom_line_co = list(zip(*line(*(int(bbox[0])+50,int(bbox[3])), *(int(bbo
```

上面是提取底线坐标的代码。我们通过从线条的每一侧移除 50 个像素坐标来减少线条长度，以便更好地表示车辆。因此，该线始终停留在车辆区域内，并且不占用其周围的任何空闲空间。

```
if len(intersection(parking_co, bbox_bottom_line_co)) > 0:
frame_matrix.append((str(frame_num) + class_name + str(t),
(
str(int(bbox[0])).zfill(4), str(int(bbox[1])).zfill(4), str(int(bbox[2])).zfill
str(int(bbox[3])).zfill(4))))
```

想象一辆车第一次与该地区相交。当它发生在特定帧内的那一刻，我们立即附加该事件的帧号frame_num车辆类型（class_name），车辆跟踪 ID（str(t)）和该车辆在该帧的边界框坐标

```
chk_index = str(frame_matrix).find(str(frame_num - 1) + class_name + str(t))
```

然后立即检查前一帧是否为同一车辆发生了相同类型的相交点。如果这个相交点是第一次发生，则不满足这个条件，程序进入下一帧，没有任何进一步的交互。但是想象一下，如果这是车辆与感兴趣区域相交后的第二帧，那么会为这个chk_index变量赋一个值。

```
if bool(chk_index + 1) == True:
previous_bbox_co_str = str(frame_matrix)[
(chk_index - 1) + len(str(frame_num - 1)) + len(class_name + str(t)) + 5:(chk_i
```

我们首先对(chk_index+1)的布尔值给出一个正值。设置为1的原因是，对于特定车辆，它的日志详细信息可能从数组的最开头开始，因此将其在数组中的放置值设置为 0。如果发生这种情况，bool(0)会使条件为 false即使它为真。如果没有这样的日志条目，则 chk_index仅返回 -1，当设为 +1 时，它会给出所需的 False 输出。

此外，当该条件为真时，将会有关于边界框的前一帧日志详细信息获取到另一个名为previous_bbox_co_str的变量中。

现在我们知道了车辆在当前帧和前一帧的边界框坐标。由于我们一直都知道车辆一直在 ROI 中，因此我们需要确定车辆是静止（不动）还是移动的。这里我们声明了一个名为immobile的新函数，用于实现这个功能。

```
# Check the immobility of vehicle #line 99
def immobile(bbox, previous_bbox_str):
previous_bbox0 = int(previous_bbox_str[1:5])
previous_bbox1 = int(previous_bbox_str[9:13])
previous_bbox2 = int(previous_bbox_str[17:21])
previous_bbox3 = int(previous_bbox_str[25:29])

total = abs(bbox[0] - previous_bbox0) + abs(bbox[1] - previous_bbox1) + abs(bb
bbox[3] - previous_bbox3)
if total <= 4:
return True
else:
return False
```

这里我们使用了previous_bbox_co_str作为函数的属性，以及当前的边界框坐标。我们分别计算xmin、ymin、xmax 和 ymax 值的差值，如果绝对差值小于 4，输出 True，表示车辆是不动的，否则输出 False。
需要注意，即使车辆或任何物体完全停止，YOLO 也会给出波动的边界框坐标。为避免这个现象并使此过程稳健，我们在此处将判定变量设置为比较高的值。该值越高，程序对边界框的随机波动就越鲁棒。
因此，如果作为车辆的函数输出是不动的（静止的），那么我们需要立即检查 cache_matrix 中是否有任何先前记录的条目，如果没有，我们需要加上当前时间（t_start）和车辆类型（class_name）并跟踪标识（str(t)）

```
if str((class_name + str(t))) not in str(cache_matrix): #line 310 t_start
= datetime.now()
cache_matrix.append((str(t_start), class_name + str(t)))
print(cache_matrix)
```

之后，我们检查车辆是否在cache_matrix中。同时还要检查同一辆车是否也在viol_matrix中。（因为我已经将违规车辆记录，这些车辆已经静止并超过了时间限制）。如果不是，我们应立即检索该特定车辆的t_start值并使用当前时间检查不同的时间。

```
index = (str(cache_matrix).find(str((class_name + str(t))))) #line 318
t_start_cm = str(cache_matrix)[index – 28:index – 11]
t_spending = (datetime.now() – datetime.strptime(t_start_cm,
‘%y-%m-%d %H:%M:%S’)).total_seconds()
```

如果时间差 (t_spending) 超过时间限制，那么我们将详细信息记录到电子表格中，其中包括车辆第一次进入时间 (t_start_cm)、车辆类型 (class_name) 和车辆跟踪 ID (str(t)) . 在这种情况下，出于演示目的，我们将计时器设置为 10 秒。示例如下

```
sheet.write(row_num, 0, str(t_start_cm), style) #line 328
# sheet.write(row_num, 1, str(round(t_spending, 2)), style)
sheet.write(row_num, 1, str(class_name ) + str(t), style)
row_num += 1
workbook.save('outputs/xlsx/parking/details.xls')
```

然后我们必须将这辆特定车辆作为日志条目放入viol_matrix数组中，因为该车辆现在已经违规了。这避免了进一步的重复和错误的日志条目。然后，我们可以拍摄该违规车辆的图像并将其保存为车辆类型，并将跟踪 ID 作为其名称。

```
viol_matrix.append((str((class_name + str(t))))) #line 334
# print(t_start_cm, t_spending, datetime.now())
cropped = image.crop((int(bbox[0]), int (bbox[1]), int(bbox[2]), int(bbox[3])))
cropped.save(
'outputs/caps_of_detections/parking/' + str(
class_name) + str(t) + str(' .jpg'))
```

以上基本上是我们设计的违章停车检测系统的想法。为避免frame_matrix的内存过载，每超过107 个条目使用以下函数刷新该数组。

```
#Avoid buffer overflow #line 353
if len(frame_matrix)>10⁷:
frame_matrix=[]
```

以上就是本文的全部内容啦，感兴趣的小伙伴们可以操练其来了！

努力分享优质的计算机视觉相关内容，欢迎关注：

AI算法与图像处理

考研逆袭985，非科班跨行AI，目前从事计算机视觉的工业和商业相关应用的工作。分...

251篇原创内容

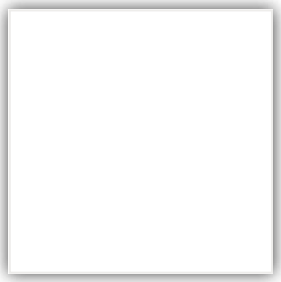
公众号

交流群

欢迎加入公众号读者群一起和同行交流，目前有**美颜、三维视觉、计算摄影**、检测、分割、识别、**NeRF、GAN、算法竞赛**等微信群

个人微信（如果没有备注不拉群！）

请注明：地区+学校/企业+研究方向+昵称



下载1：何恺明顶会分享

在「AI算法与图像处理」公众号后台回复：何恺明，即可下载。总共有6份PDF，涉及ResNet、Mask RCNN等经典工作的总结分析

下载2：终身受益的编程指南：Google编程风格指南

在「AI算法與圖像處理」公眾號後台回复：c++ ，即可下載。歷經十年考驗，最權威的編程規範！

下載3 CVPR2021

在「AI算法與圖像處理」公眾號後台回复：CVPR，即可下載1467篇CVPR 2020論文和CVPR 2021最新論文