工作记录

昨天课太多杂事太多，没有工作量。

今天工作安排：

1.图像标记加强

效果：可以分出车辆车位背景环境和停车线，当车辆在停车位上时，露出部分均标记为停车位，车位外为背景色，完善停车线车辆和背景，将噪声点和不连续段补全

实现：

针对车辆噪点：根据像素和位置之间的关系，比较周围的像素点，补全噪点

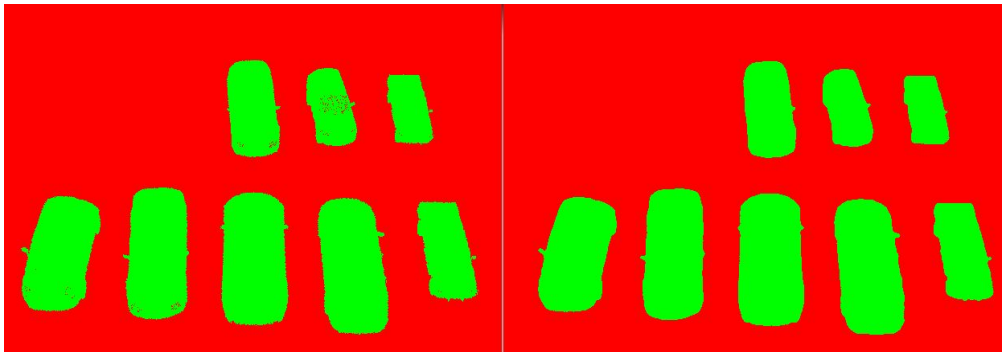
针对停车线不连续部分：根据停车线相对位置关系进行补全

针对停车位：先绘制出停车位部分，再绘制车辆部分

1.针对车辆噪点，使用像素对比方式，代码如下：

```
for i in range(10,img_allblack.shape[0]-10):
    for j in range(10,img_allblack.shape[1]-10):
        #print img_allblack[i,j]
        if str(img_allblack[i-1,j])==str([0,0,0]) and str(img_allblack[i+1,j])==str(
            img_allblack[i,j]=[0,0,0]
```

处理结果和原始结果对比：



分析：实现方法简单，但是每张图片计算点数太多，速度较慢，应该采用其他方式进行处理，如滤波，尝试各类滤波算法，代码如下：

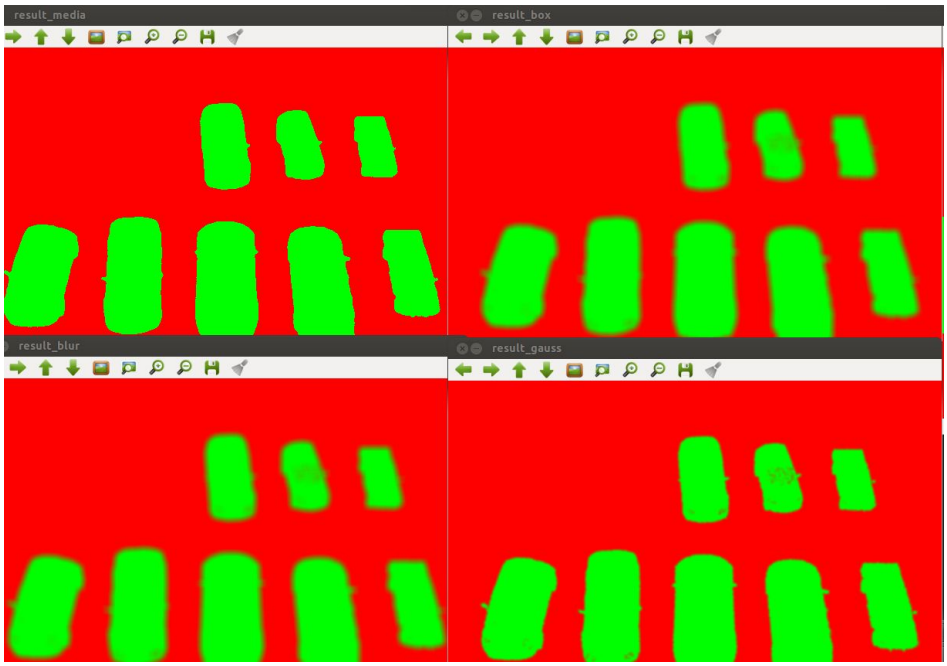
```
result_blur= cv2.blur(img_allblack, (10,10))


result_box = cv2.boxFilter(img_allblack, -1, (10, 10))
gaussianResult = cv2.GaussianBlur(img_allblack,(5,5),1.5)

result_media = cv2.medianBlur(img_allblack,5)

cv2.imshow('result_media',result_media)
cv2.imshow('result_box',result_box)
cv2.imshow('result_blur',result_blur)
cv2.imshow('result_gauss',gaussianResult)
cv2.imshow('imgsource',img_allblack)
cv2.waitKey(0)
cv2.destoryAllWindows()
```

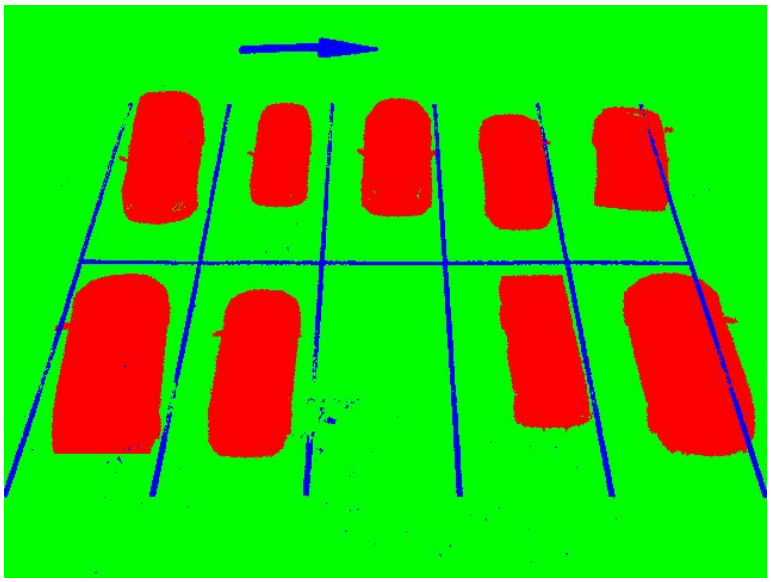
结果如下：



PS：效果最好的就是中值滤波，和之前编写的代码效果相同，但是处理速度加快

2.针对停车线不连续部分

原始检测图如下：



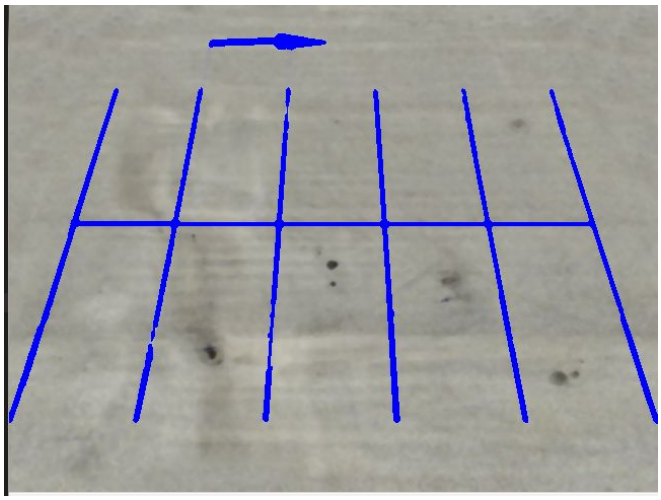
注：可见停车线不够连续，新的方法代码如下：

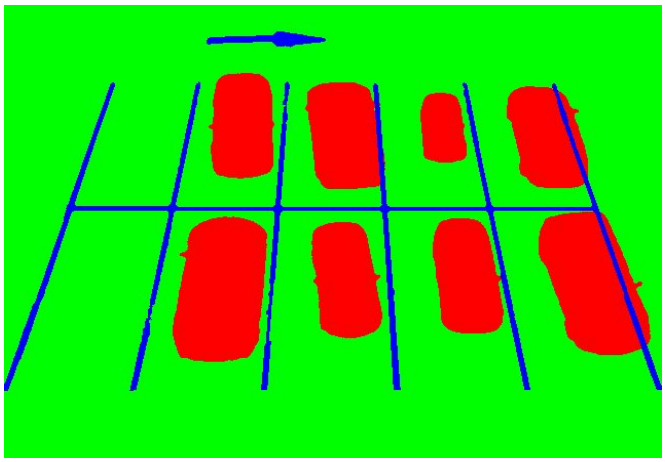
```
img = cv2.imread('RGB.png')

gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray,50,200)
plt.subplot(121),plt.imshow(edges,'gray')
plt.xticks([]),plt.yticks([])
#hough transform
lines = cv2.HoughLinesP(edges,1,np.pi/180,28,minLineLength=28,maxLineGap=60)
lines1 = lines[:,0,:]
for x1,y1,x2,y2 in lines1[:]:
    cv2.line(img,(x1,y1),(x2,y2),(255,0,0),1)

img = cv2.medianBlur(img,5)
img[np.where((img>[182,0,0]).all(axis=2))]=[255,0,0]
cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destoryAllWindows()
```

最终效果：





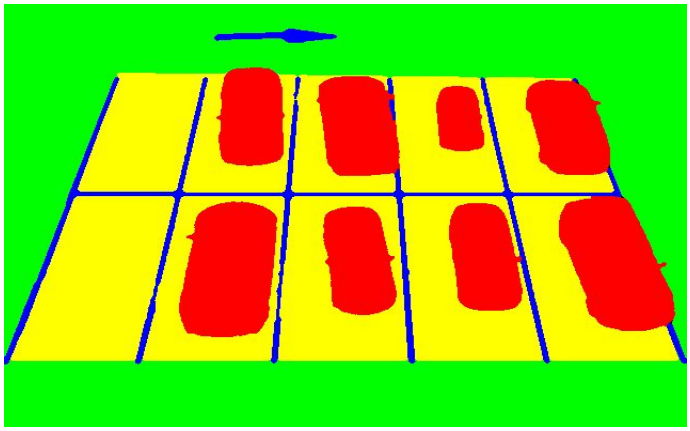
3.针对停车位

需要实现停车位颜色和背景颜色不想同，当车位上有车时，露出来的车位显示的是车位的颜色

代码实现如下：

```
for img_list in imgs_list:
    img_list_read=cv2.imread(img_list)
    img_list_read_copy=img_list_read.copy()
    #prfloat img_posis [img_index]
    pts = np.array([[106,78],[1,404],[632,404], [532,84]],np.int32)
    pts = pts.reshape((-1,1,2))
    cv2.fillPoly(img_list_read_copy,[pts],(0,255,255))
    for parkingline_Posi in parkingline_Posis:
        img_list_read_copy[parkingline_Posi[0],parkingline_Posi[1]]=[255,0,0]
    img_list_read_copy[np.where((img_list_read_copy==[255,255,255]).all(axis=2))]=[0,255,0]
    img_list_read_copy[np.where((img_list_read<[255,255,255]).all(axis=2))]=[0,0,255]
    img_list_read_copy = cv2.medianBlur(img_list_read_copy,5)
```

最终实现效果如下：



2.无人机图片标记

效果：通过K聚类方法实现自动标记，达到较高的准确率

实现：

```
'''
kMeans
'''
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('woman.jpg',0)#image read be 'gray'
plt.subplot(121),plt.imshow(img,'gray'),plt.title('original')
plt.xticks([]),plt.yticks([])

#change img(2D) to 1D
img1 = img.reshape((img.shape[0]*img.shape[1],1))
img1 = np.float32(img1)

#define criteria = (type,max_iter,epsilon)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,10,1.0)

#set flags: hou to choose the initial center
#---cv2.KMEANS_PP_CENTERS ; cv2.KMEANS_RANDOM_CENTERS
flags = cv2.KMEANS_RANDOM_CENTERS
# apply kmenas
compactness,labels,centers = cv2.kmeans(img1,4,None,criteria,10,flags)

img2 = labels.reshape((img.shape[0],img.shape[1]))
plt.subplot(122),plt.imshow(img2,'gray'),plt.title('kmeans')
plt.xticks([]),plt.yticks([])
```

最终效果比较差，无法正常使用

PS:现在代码越来越多，不好管理了，该是要找个好办法管理代码了，github 可以，不过得想点办法，让自己比较方便管理