人工智能基础与进阶

基于OpenCV的视觉信息处理

上海交通大学

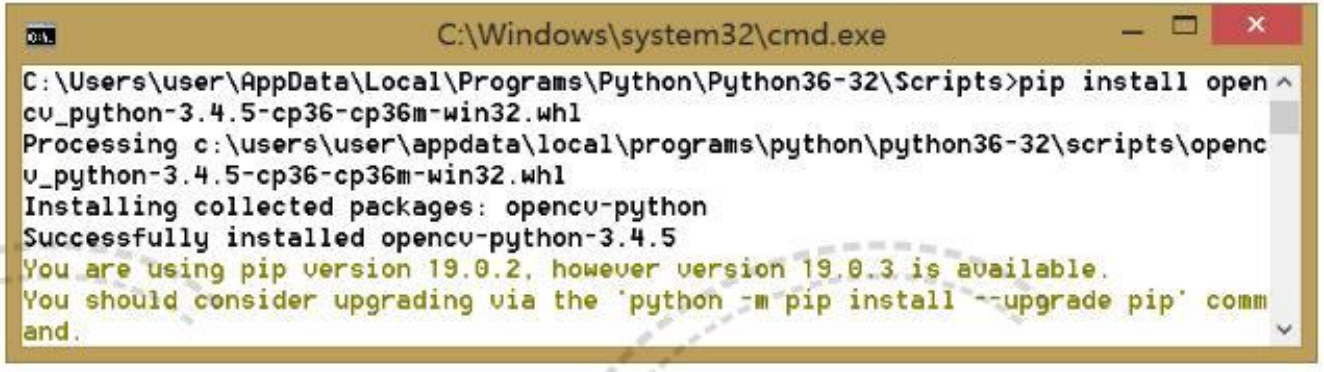
图像的读取和简单处理

Python应用

这小节将说明**OpenCV计算机视觉库的应用**。OpenCV是一个开源跨平台的计算机视觉库，可以运行在Windows、Linux、Mac OS、和Android操作系统上，由一系列C函数和少量C++类构成，同时提供了Python语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

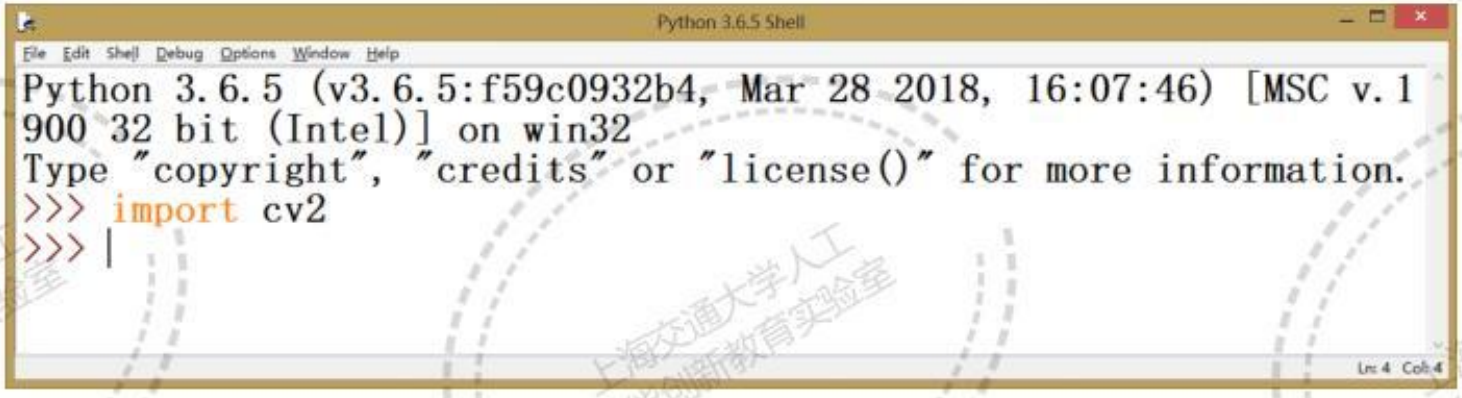
Python应用-安装OpenCV

pip install opencv_python



Python应用-安装OpenCV

测试是否安装成功，在Python Shell窗口输入import cv2没有错误信息表示已经成功安装opencv了，注意导入OpenCV是import cv2而不是import opencv。



OpenCV读取和显示图像

1. 建立OpenCV图像窗口
2. OpenCV读取图像
3. OpenCV显示图像
4. 关闭OpenCV图像窗口
5. 时间等待
6. 储存图像
7. ROI的截取

OpenCV读取和显示图像

(1) 建立OpenCV图像窗口

OpenCV使用namedWindow( )建立显示图像的窗口，表达式如下：

cv2.namedWindow(窗口名称,窗口标识)

窗口标识一般默认为WINDOW_AUTOSIZE，可能值如下：

WINDOW_AUTOSIZE：窗口大小自动适应图像大小，并且不可手动更改。

WINDOW_NORMAL：窗口大小可以改变。

WINDOW_OPENGL：窗口创建的时候会支持OpenGL。

OpenCV读取和显示图像

(2) OpenCV读取图像

OpenCV使用imread()读取图像，回传给图像对象，如同pillow图像处理时使用open()。表达式如下：

Image_object = cv2.imread(图像路径，图像标识)

Image_object为图像对象，名称可以自行命名，图像标识可能值如下：

cv2.IMREAD_COLOR：预设值表示读取彩色图像，也可用值为1表示。

cv2.IMREAD_GRAYSCALE：表示读取灰度图像，也可用值为0表示。

cv2.IMREAD_UNCHANGED：表示读取包含alpha通道的彩色图像。

OpenCV读取和显示图像

(3) OpenCV显示图像

OpenCV使用imshow( )显示读取的图像对象显示在指定窗口内，表达式如下：

cv2.imshow(窗口名称，图像对象)

OpenCV读取和显示图像

(4) 关闭OpenCV图像窗口

图像显示在窗口后，在大型的程序中，会设计关闭窗口来清除内存，是非常重要的，可以是暂停几秒或是按下某键后关闭窗口。表达式如下：

```
cv2.destroyWindow(窗口名称)    #删除单一指定窗口
cv2.destroyAllWindows( )    #删除所有窗口
```

OpenCV读取和显示图像

(5) 时间等待

OpenCV使用cv2.waitKey(s)等待时间，s单位为毫秒，若s=0代表永久等待

OpenCV读取和显示图像

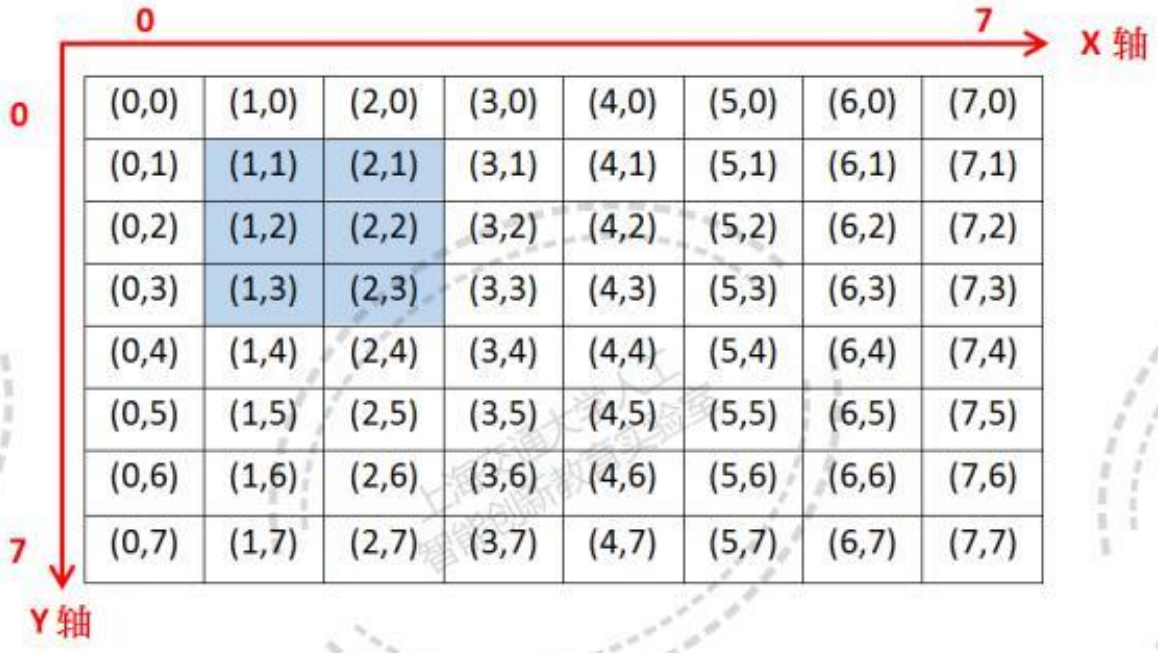
(6) 储存图像

OpenCV使用imwrite( )储存图像，表达式如下：

cv2.imwrite(图像路径,图像对象)

图像坐标

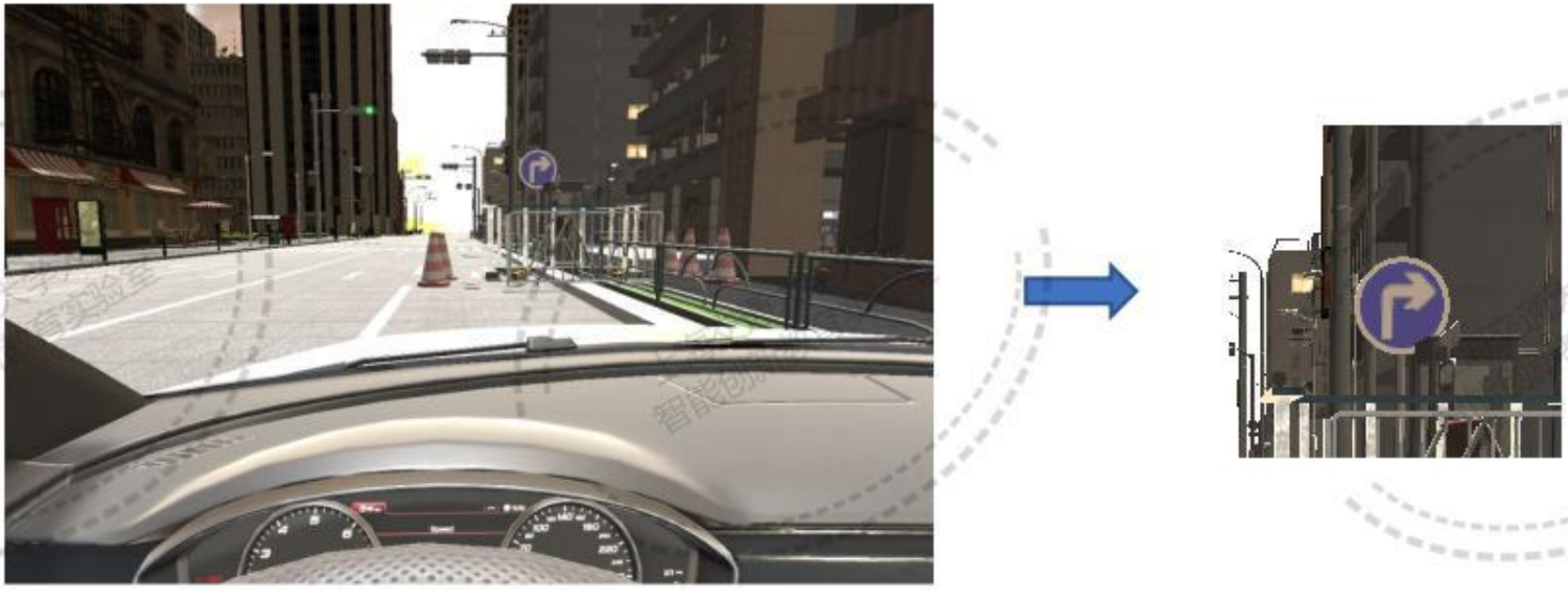
有些图像处理的方法是针对图像的像素值进行更改或是取值，因此图像坐标非常重要，才能准确找到像素点的位置，甚至是一个范围进行裁切等。



(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(7,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)	(7,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	(6,3)	(7,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)	(7,4)
(0,5)	(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)	(7,5)
(0,6)	(1,6)	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)	(7,6)
(0,7)	(1,7)	(2,7)	(3,7)	(4,7)	(5,7)	(6,7)	(7,7)

OpenCV ROI （region of interest）

简单的说就是对图像感兴趣的区域，机器视觉、图像处理中，从被处理的图像以方框、圆、椭圆、不规则多边形等方式勾勒出需要处理的区域，称为感兴趣区域，ROI。举个例子来说：有一副图片，图片上有各种动物，但是你只喜欢图片里的狗，那么这个狗所在的区域就是感兴趣的区域（ROI）。



OpenCV ROI （region of interest）

```
RoI = image[130:300,730:900,:]
```

#将变量image中纵向130~300像素，横向730~900像素，RGB复制给RoI

OpenCV读取和显示图像

任务：

1. 在sign_exp文件夹下的opencv文件夹中，打开imshow.py

2. 阅读程序并运行程序，使得程序依次显示原始图像和RoI区域的图像，并且能保存RoI图像到这个文件夹中。

OpenCV读取和显示图像

任务：

1. 在sign_exp文件夹下的opencv文件夹中，打开imshow.py
2. 阅读程序并运行程序，使得程序依次显示原始图像和RoI区域的图像，并且能保存RoI图像到这个文件夹中。

```
import cv2
# image= cv2.imread("./images/00194.png")
image = cv2.imread("../sign_svm/picture/right.png")

cv2.namedWindow("right")
cv2.imshow("right",image)
cv2.waitKey(0)
cv2.destroyWindow("right")

ROI=image[130:300,730:900,:]
cv2.namedWindow("ROI show")
cv2.imshow("ROI show",ROI)
key=cv2.waitKey(0)
cv2.destroyWindow("ROI show")

if key&0xff ==ord('s'):
    cv2.imwrite("ROI.png",ROI)
    print('This image is saved.')
else:
    print("This image is not saved")
```

OpenCV ROI (region of interest)

#将取出的区域改变为灰度图像

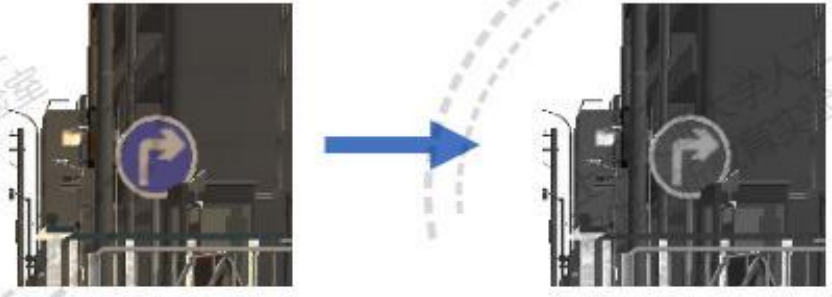
gray=cv2.cvtColor(roisrc,cv2.COLOR_BGR2GRAY)

任务：

1. 在sign_exp文件夹下的opencv文件夹中，打开grayRoI.py
2. 补全程序中缺失的部分，使其显示灰色的灰度图像。

```
import cv2

#补全缺失的程序，使其显示灰度图像
roisrc = cv2.imread("RoI.png")
```



OpenCV ROI (region of interest)

#将取出的区域改变为灰度图像

gray=cv2.cvtColor(roisrc,cv2.COLOR_BGR2GRAY)

任务：

1. 在sign_exp文件夹下opencv文件夹中，打开grayRoI.py
2. 补全程序中缺失的部分，使其显示灰色的灰度图像。

参考程序：

```
import cv2

#补全缺失的程序，使其显示灰度图像
roisrc = cv2.imread("RoI.png")

gray=cv2.cvtColor(roisrc,cv2.COLOR_BGR2GRAY)
cv2.namedWindow("gray")
cv2.imshow("gray",gray)
cv2.waitKey(0)
cv2.destroyWindow("gray")
```

图像处理实验

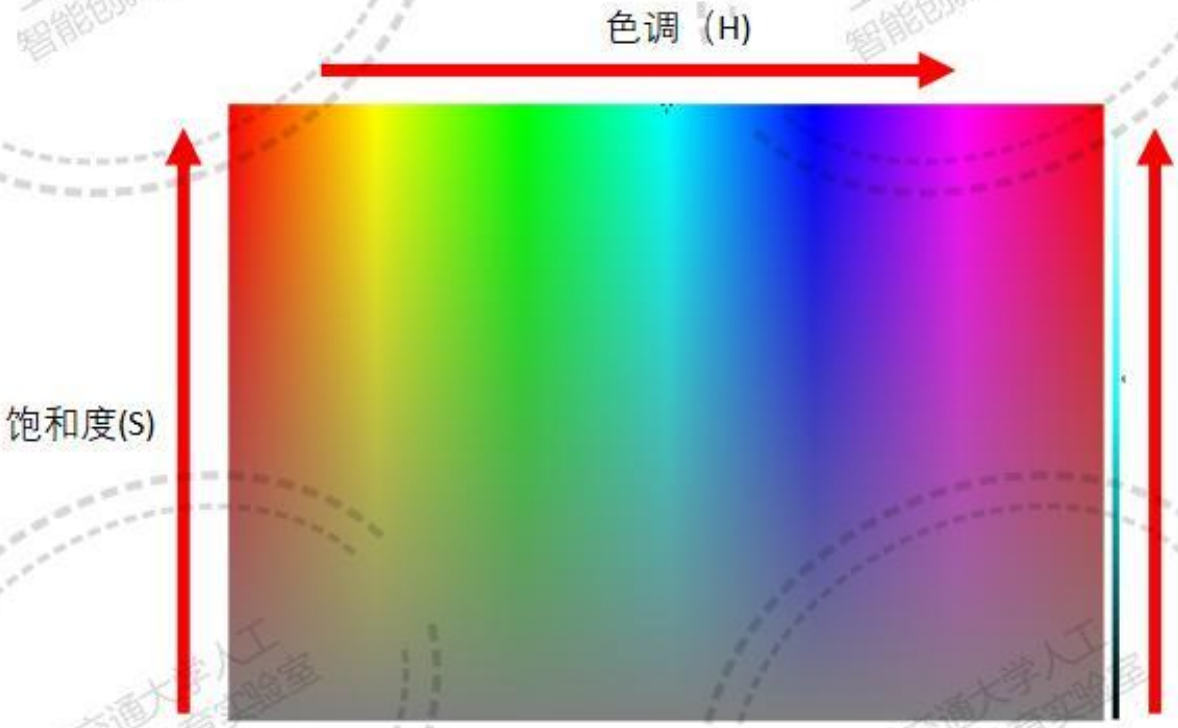
第一节 红绿灯标志牌检测

第一节　红绿灯，标志牌以及车道线检测

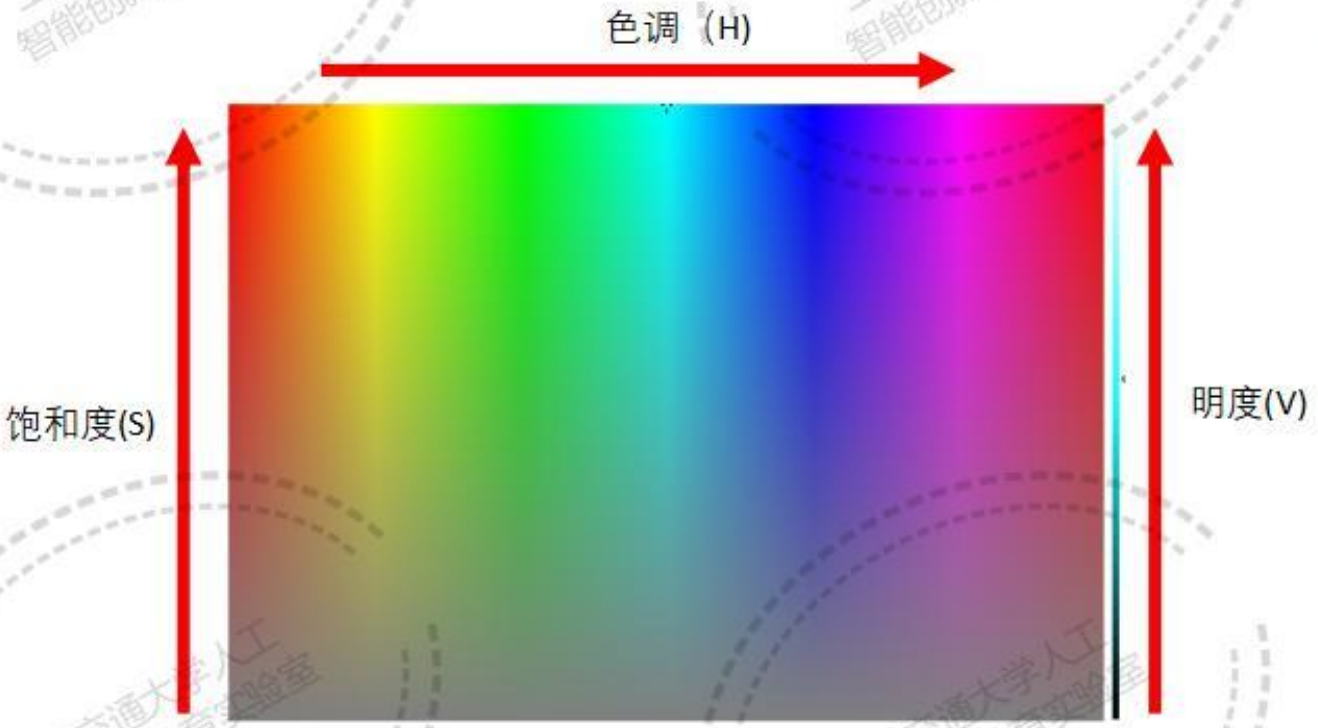
- 图像的HSV
- 颜色分割检测标志牌
- 霍夫圆检测标志牌
- 霍夫直线检测车道线

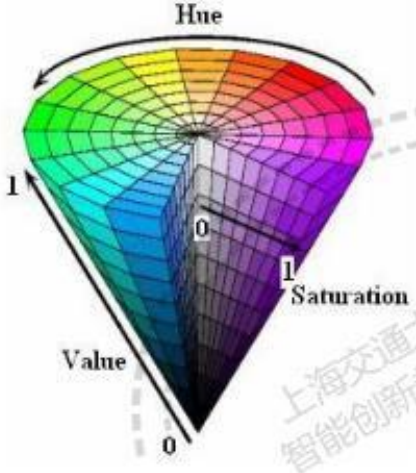
检测

查看图像HSV值



查看图像HSV值





使用色调筛选出黄色且滤除其他颜色，框出筛选后的黄色区域

1. 在extend_exp目录下运行get_pixel.py

2. 在弹出的窗口中点击图片，观察不同颜色对应的HSV值 H（输出的三个数字中的第一个）

3. 思考黄色灯的色调H的取值范围，以滤除其他颜色

4. 在detection.py所在文件夹下打开test1.py，修改右侧代码中的h_low，h_high 取值，查看检测效果

```
import cv2

from detection import detection

detector = detection()

im = cv2.imread("./images/extend.png")

detector.change_threshold(h_low=,h_high=)

rects = detector.seg_color(im)

detector.show_rects(im,rects)
```

◆打开detection.py，阅读detection类下除__init__(self)外的6个函数

判断每个函数的输入是什么，输出是什么？

如果没有输出，作用是什么？

detection 的6个函数:

- show_circles(im, circles)
- show_rects(im, rects)
- change_threshold(h_low, h_high)
- seg_color(im)
- det_circle(img)
- ensemble(img)

用颜色分割检测标志牌

使用色调筛选出蓝色且滤除其他颜色，框出筛选后的蓝色区域

1. 在detection.py所在文件夹下打开test2.py

2. 把红色部分改成数值，使得程序能检测图片right.png

3. 运行程序，查看检测效果

4. 试试把图像换成left.png？

```
import cv2

from detection import detection

detector = detection()

im = cv2.imread("./images/00194.png")

roi =im[ymin:ymax,xmin:xmax,:]

rects = detector.seg_color(roi)

detector.show_rects(roi,rects)
```

使用色调筛选出蓝色且滤除其他颜色，框出筛选后的蓝色区域

1. 在detection.py所在文件夹下打开test2.py

2. 把红色部分改成数值，使得程序能检测图片right.png

3. 运行程序，查看检测效果

4. 试试把图像换成left.png？

```
import cv2

from detection import detection

detector = detection()

im = cv2.imread("./images/right.png")

roi =im[130:300,730:900,:]

rects = detector.seg_color(roi)

detector.show_rects(roi,rects)
```

霍夫圆检测标志牌

使用霍夫圆检测框出图像中的标志牌

1. 在detection.py所在文件夹下新建test3.py

2. 修改红色部分路径和数值，使得程序能检测图片right.png

```
import cv2

from detection import detection

detector = detection()

im =cv2.imread ("./images/right.png")

roi =im[ymin:ymax,xmin:xmax,:]

circles = detector.det_circle(roi)

detector.show_circles(roi,circles)
```

使用霍夫圆检测框出图像中的标志牌

1. 在detection.py所在文件夹下新建test3.py

2. 修改红色部分路径和数值，使得程序能检测图片right.png

3. 程序是否能正常运行？

4. 查看报错，尝试自己解决这个bug

5. 试试把图像换成left.png？

```
import cv2

from detection import detection

detector = detection()

im = cv2.imread("./images/right.png")

roi =im[130:300,730:900,:]

circles = detector.det_circle(roi)

detector.show_circles(roi,circles)
```

霍夫直线检测车道线

使用霍夫直线检测标出图像中的车道线

1. 在detection.py所在文件夹下新建，houghline.py

2. 修改红色部分路径和数值，使得程序能检测图片line.png

```
import cv2

src=cv2.imread("sign_svm/picture/line3.png")

gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)

label = (h >= H_low_thresh) * (h < H_high_thresh)
* (s >= S_low_thresh) * (s < S_high_thresh) * (v >
254)

gray = gray * label
gray[label] = 255

lines = cv2.HoughLines(gray, 1, np.pi/180,xxx)
```

使用霍夫直线检测标出图像中的车道线

1. 在detection.py所在文件夹下新建，houghline.py

2. 修改红色部分路径和数值，使得程序能检测图片line.png

3. 程序是否能正常运行？

4. 查看报错，尝试自己解决这个bug

5. 试试把图像换成line2.png？

```
import cv2

src=cv2.imread("sign_svm/picture/line3.png")

gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)

label = (h >= H_low_thresh) * (h < H_high_thresh)
* (s >= S_low_thresh) * (s < S_high_thresh) * (v >
254)

gray = gray * label
gray[label] = 255

lines = cv2.HoughLines(gray, 1, np.pi/180,500)
```

第二节 使用SVM检测标志牌

实验目的

学习使用SVM算法对三种交通标志牌（左转、右转和直行）的图像进行分类。



实验步骤：图像特征提取、训练SVM分类器、测试分类

代码分析

prepare.py——提取特征

svm.py——封装的SVM类

实验方法

1. 特征提取

调用prepare.py文件的prepare_data函数对data/train_data文件夹下的训练图片进行特征提取。

实验步骤：

```
from prepare import prepare_data

data = prepare_data('hog')
```

结果：

```
Number of training images :  2309
Number of classes         :     3
```

实验方法

2. 训练SVM分类器

初始化SVM对象，运行svm.train(data)训练SVM分类模型，并将结果保存到model文件夹下

扩展上一步代码为：

```
from svm import SVM
from prepare import prepare_data

svm = SVM()
data = prepare_data('hog')
svm.train(data)
```

结果：

```
Test accuracy of SVC:  1.0
```

实验方法

3. 测试分类

初始化svm对象，运行svm.predict(image, feature_type)对测试标志牌进行分类。

实验代码：

```
import cv2
from svm import SVM
img = cv2.imread('./data/test_data/left.png')
svm = SVM()
ID_num = svm.predict(img, 'hog')
print('Sign prediction calss ID: ', ID_num)
```

结果：

```
Sign prediction calss ID:  34
```

实验方法

4. 调用训练好的SVM模型对./data/test_data文件夹中的不同标志牌图像进行分类，并将分类结果打印到图片上显示。