北京郵電大學

《多媒体通信》

课程设计

姓　　名　王卓

学　　院　信息与通信工程院

专　　业　通信工程

班　　级　2015211112

学　　号　2015210335

班内序号　11

2018 年 7 月

一、实验目标

在视频中可以检测到行人位置并标出，进而达到统计行人个数的目的。但由于实验结果对于行人个数统计的准确度较低，因此本实验重点仍放在行人检测上，即框出行人所在位置，并允许一定的误差。本次实验所用的数据集为 INRIA 数据库，网址为：http://pascal.inrialpes.fr/data/human/

二、技术概述

行人检测(Pedestrian Detection)是利用计算机视觉技术判断图像或者视频序列中是否存在行人并给予精确定位。该技术可与行人跟踪，行人重识别等技术结合，应用于人工智能系统、车辆辅助驾驶系统、智能机器人、智能视频监控、人体行为分析、智能交通等领域。

由于行人兼具刚性和柔性物体的特性，外观易受穿着、尺度、遮挡、姿态和视角等影响，使得行人检测成为计算机视觉领域中一个既具有研究价值同时又极具挑战性的热门课题。

现有行人检测方法有三种，分别为基于全局特征的方法、基于人体部位的方法、基于立体视觉的方法。

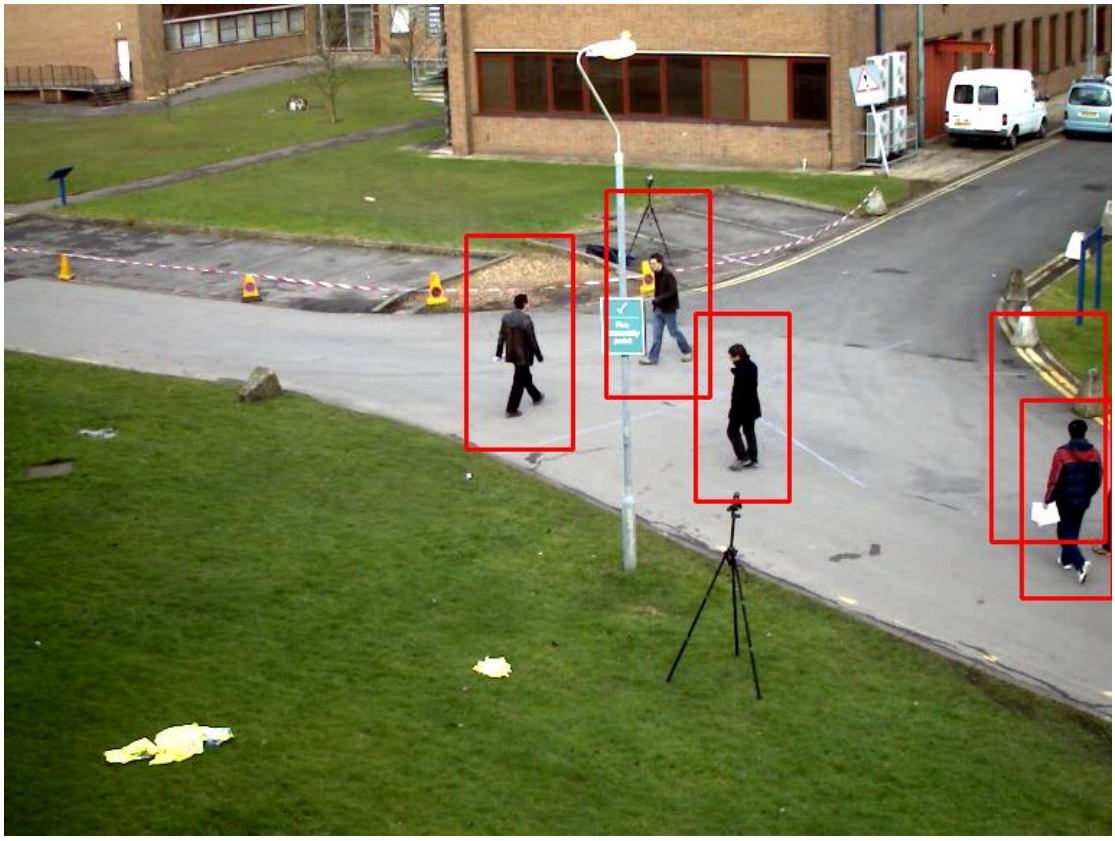
其中基于全局特征的方法是目前较为主流的行人检测方法，主要采用边缘特征、形状特征、统计特征或者变换特征等图像的各类静态特征来描述行人，主要由如下几种方法构成：

1、基于 Haar 小波特征的方法；
2、基于 HOG 特征的方法；
3、基于 edgelet 特征的方法；
4、基于 Shapelet 特征的方法；
5、基于轮廓模板的方法；
6、基于运动特征的方法；

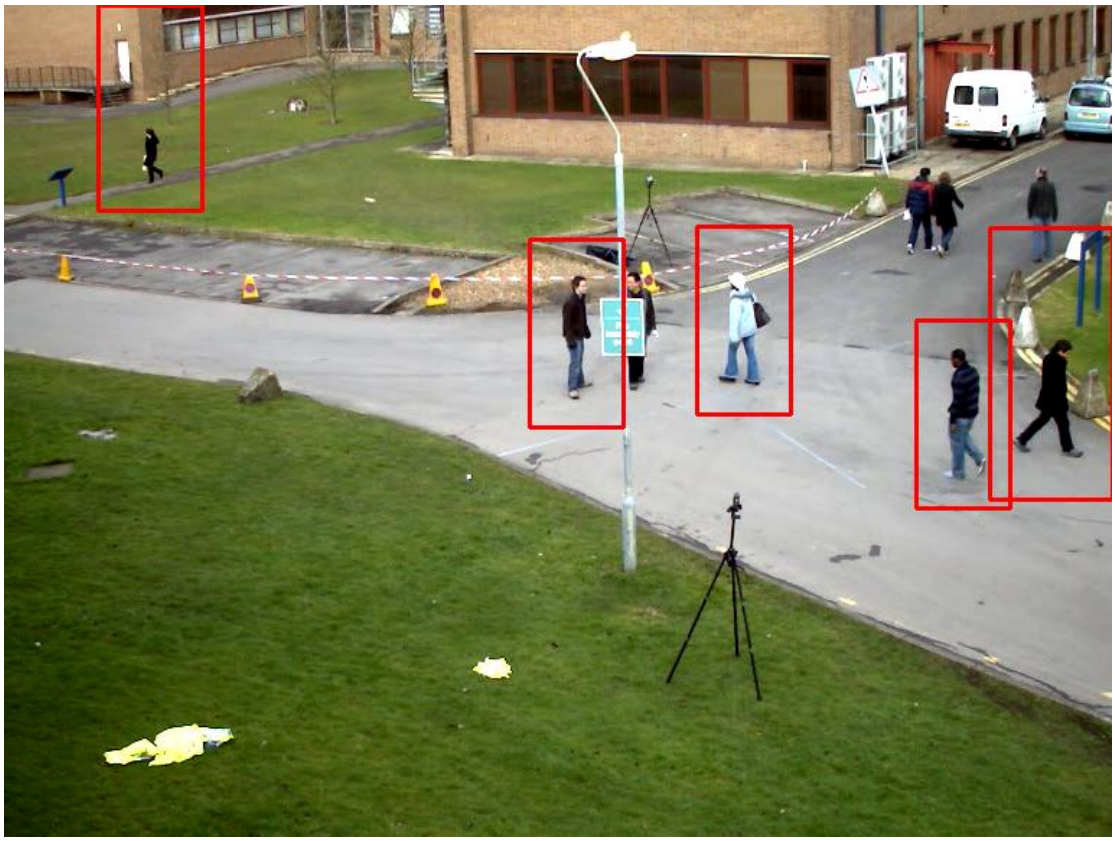
本次实验所采用的方法是方法二，基于 HOG 特征的方法；

三、功能展示

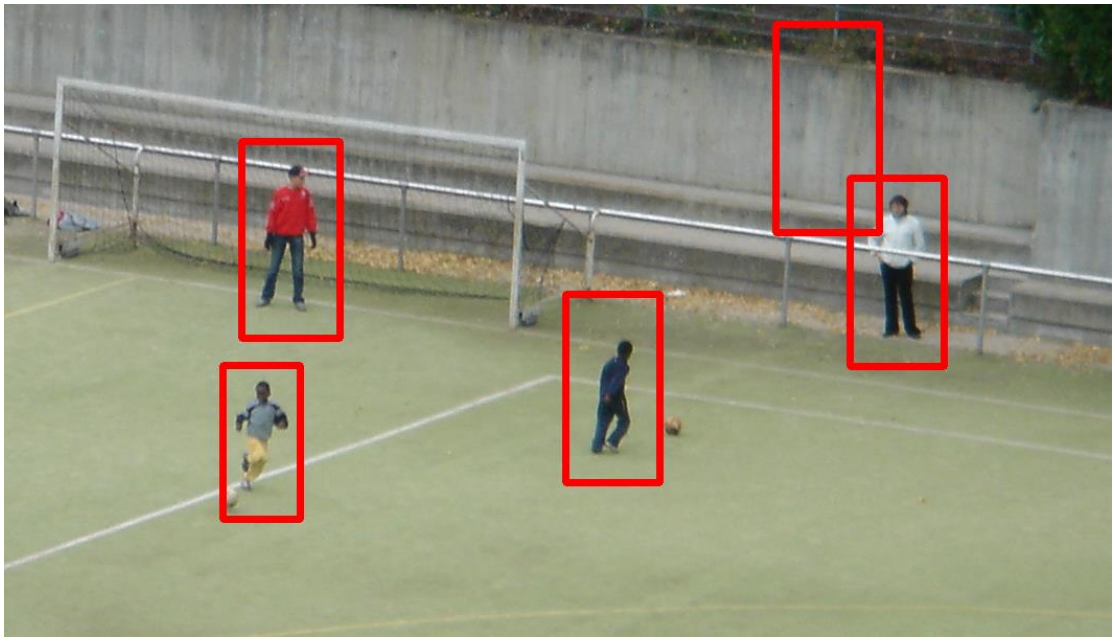
1、**软件功能：**能够实现视屏中行人的标注，以及静态图片中行人的标注。本实验中所使用的 test 资源，视屏使用的是 opencv 自带的 data 文件中的 vtest.avi 视频。静态图片选用的是 INRIA 行人检测数据集中的 test 图片。为了更好理解附几张运行截图如下：



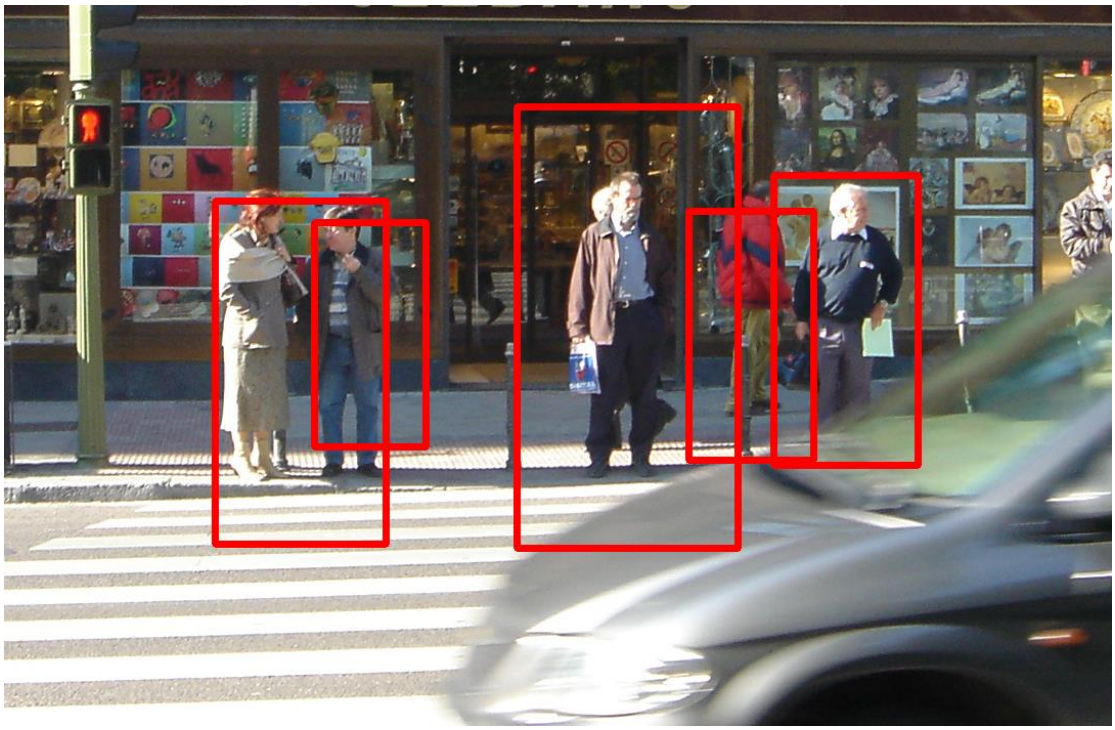
视频 test（1）



视频 test（2）



静态图 test（1）

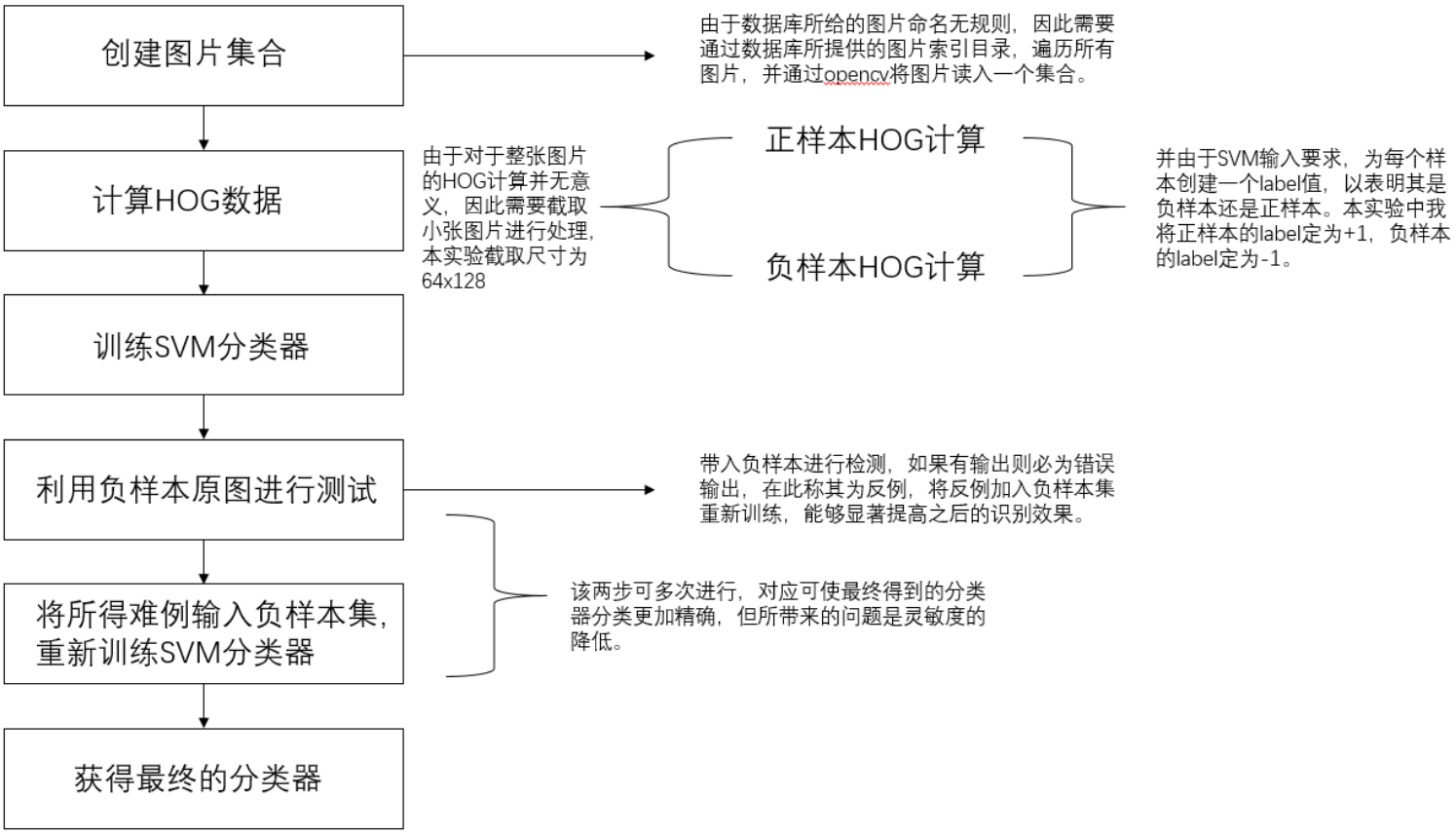


静态图 test（2）

四、总体流程

1、**概述**：本实验采用基于 HOG 特征的方法。HOG（Histogram of Oriented Gradients）是指方向梯度直方图，HOG 方法即在对一幅图像进行分析时，局部目标的表象和形状可以被梯度分布很好的好的描述. 我们对图像的各个像素点采集方向梯度直方图，根据直方图的信息就可以描述图片的特征。再将梯度信息输入 SVM 分类器中进行分类，便可实现图像识别。在行人检测问题上分类类别只有两类即行人与非行人。另外 HOG-SVM 计算机视觉的方法在 opencv 中已经得到封装，这对我完成程序大有帮助。

2、**流程图示如下**：



五、关键模块

1、工具类：

（1）负样本的获得

由于下载的数据库不自带正样本图片，只有负样本原图，因此需要自行从负样本原图中截取大小为 64x128 的图片作为形成自己的负样本集合。数据库所给的负样本原图比较少，大约为 1300 张，因此我从每张负样本中随机截取 10 张大小为 64x128 的图片作为负样本，最终得到了 13397 张负样本。与其数据库中自带的已经截好的 2415 张正样本一起构成我的样本库。在构成负样本时为了使函数拥有统一接口，因此也输出了一个 txt 文件，用以作为

对负样本集合进行索引的作用。

截取负样本的代码如下：

```
        img = cv.imread(string)
h, w, c = img.shape
if (w >= W) & (h >= H):
    while (map_name00 <= sample_time):
        x = random.randint(0, 1000) % (w - W)
        y = random.randint(0, 1000) % (h - H)
        imgROI = img[y:y + H, x:x + W]
        str_son_final = str_son + r'Train_64x128/neg/' +
str(name_num) + r'.png'
        cv.imwrite(str_son_final, imgROI)
        fw.write(str(name_num) + ".png" + "\n")
```

（2）正样本 matlab 读取

数据库所提供的正样本虽然已经截好，且大小统一为 64x128，且能够双击打开。但是却不能用 opencv 读入，很是奇怪，通过查阅网上资料得知，需要使用 matlab 读入后再输出才能被 opencv 使用。按照这一说法果然解决了问题，虽然仍然不知道原因。

（3）计算 HOG 数据

由于 opencv 已将 HOG 的计算以函数形式封装了，因此只需按照 opencv library 上的指示使用函数即可。首先将待测图像由 RGB 格式转换 GRAY 灰度图格式，调用函数计算出 HOG 即可完成。

（4）利用所得到的 HOG 数据集和 label 标识集来训练 SVM 分类器

本模块难点在于对于 SVM 的初始化，我主要是根据 opencv 官网上的范例（https://docs.opencv.org/3.4.0/d5/d77/train_HOG_8cpp-example.html），将其改造成 python 描述，从而达到目的。对于函数的参数解释请参见结尾代码附录部分。

（5）用反例原图进行测试获得难例

难点主要在于将捕捉到的图片范围缩放成 64x128 大小的图片，以放入反例集中进行再次训练。这里主要运用了 opencv 中自带的 resize 函数，如下：

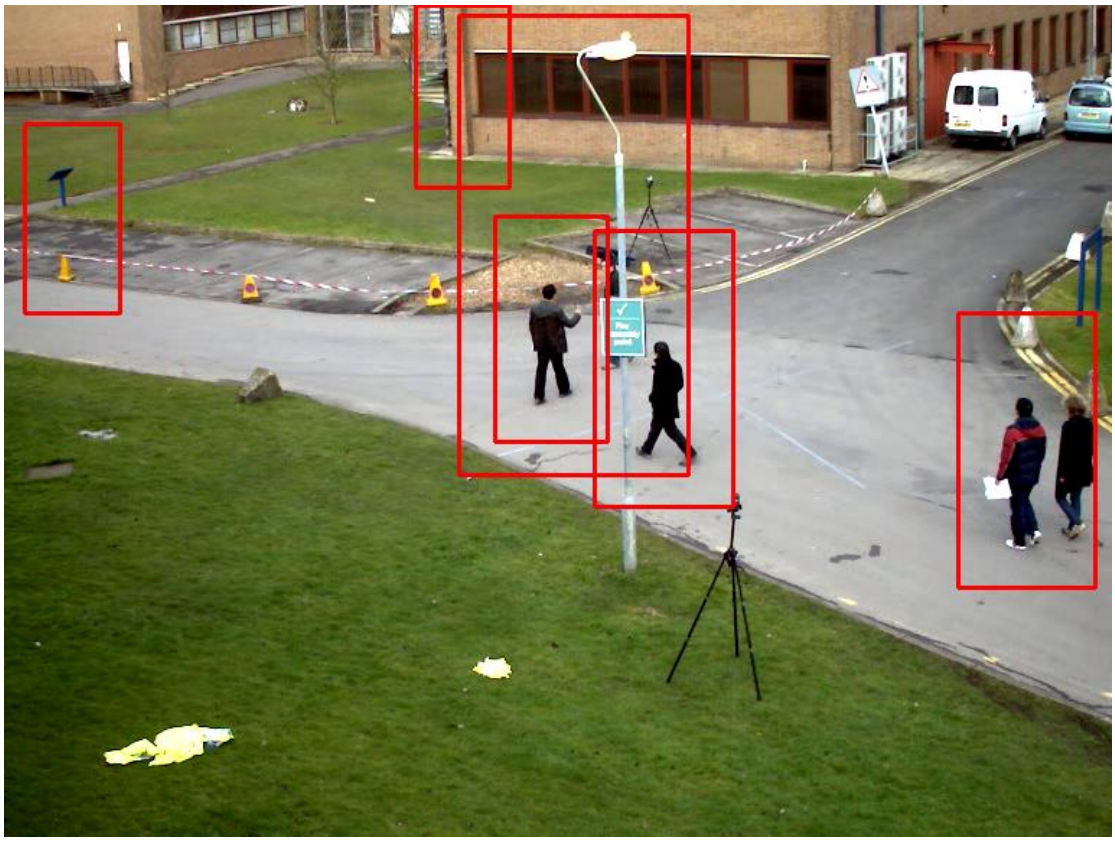
```
imgROI = cv2.resize(imgROI, (64, 128),
interpolation=cv2.INTER_CUBIC)  # 将检测错误的地方变成 64x128 存储
```

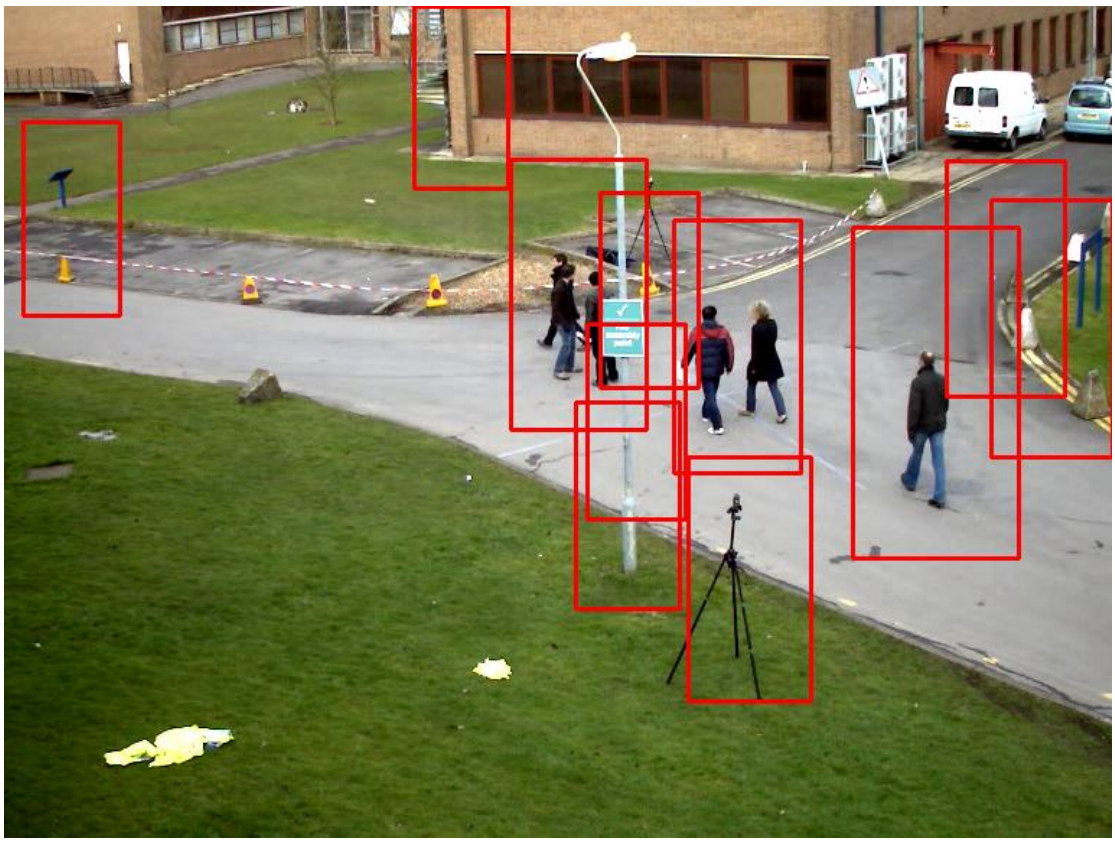
另外具体解释这种重复训练的思想。首先解释什么是难例，难例是指利用第一次训练的分类器在负样本原图(肯定没有人体)上进行行人检测时所有检测到的矩形框，这些矩形框区域很明显都是误报，把这些误报的矩形框保存为图片，加入到初始的负样本集合中，重新进行 SVM 的训练，可显著减少误报。这种方法叫做自举法，自举法首先使用初始负样本集来训练一个模型，然后收集被这个初始模型错误分类的负样本来形成一个负样本难例集。用此负样本难例集训练新的模型，此过程可以重复多次。下面几张是我的难例图片：

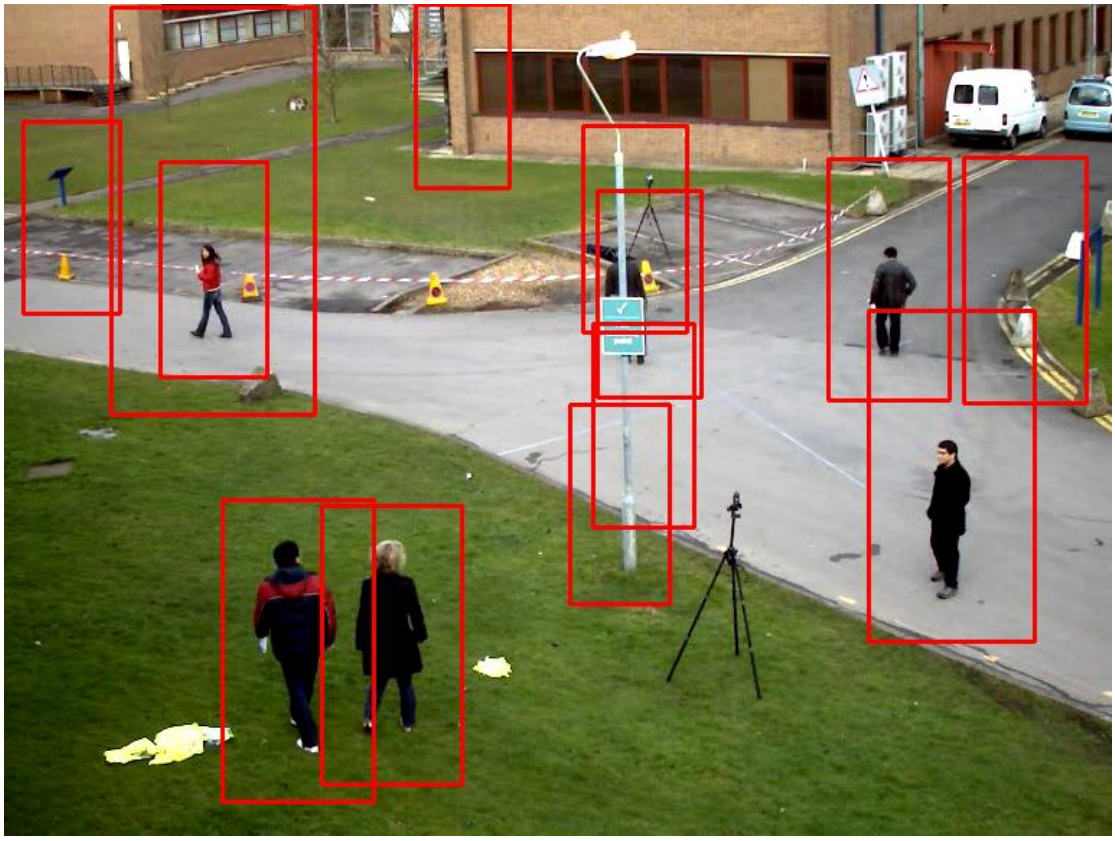
六、测试结果和分析

1、结果综述

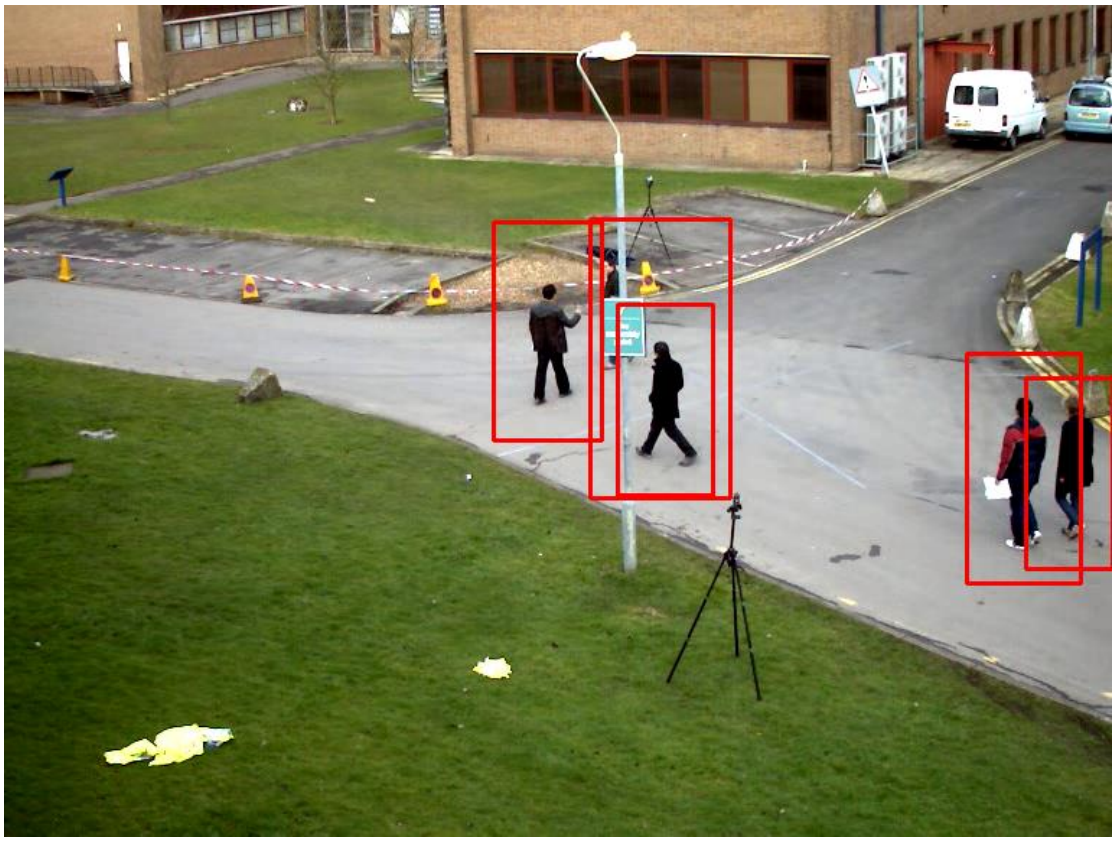
本次实验我利用自举法重新训练了 5 次，因此可以得到 6 个 SVM 模型。其中利用未使用自检法的模型进行训练时效果很不理想，就像下面这样:

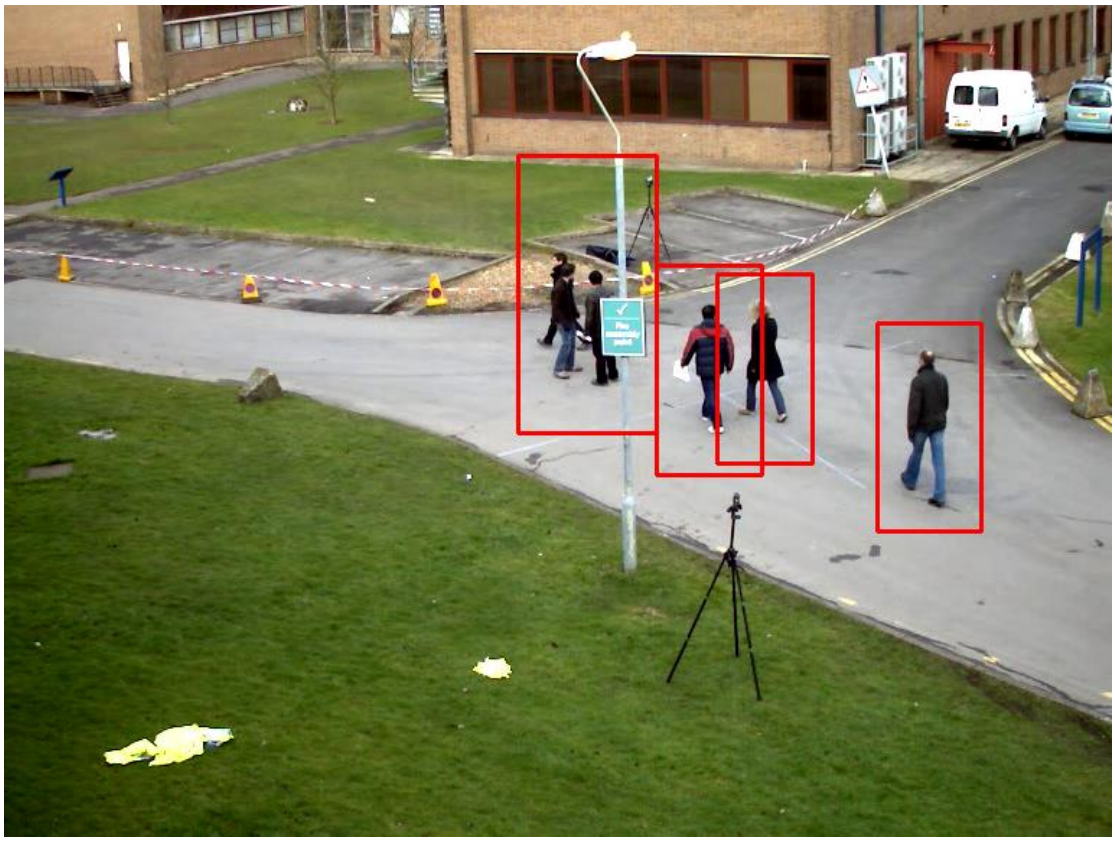


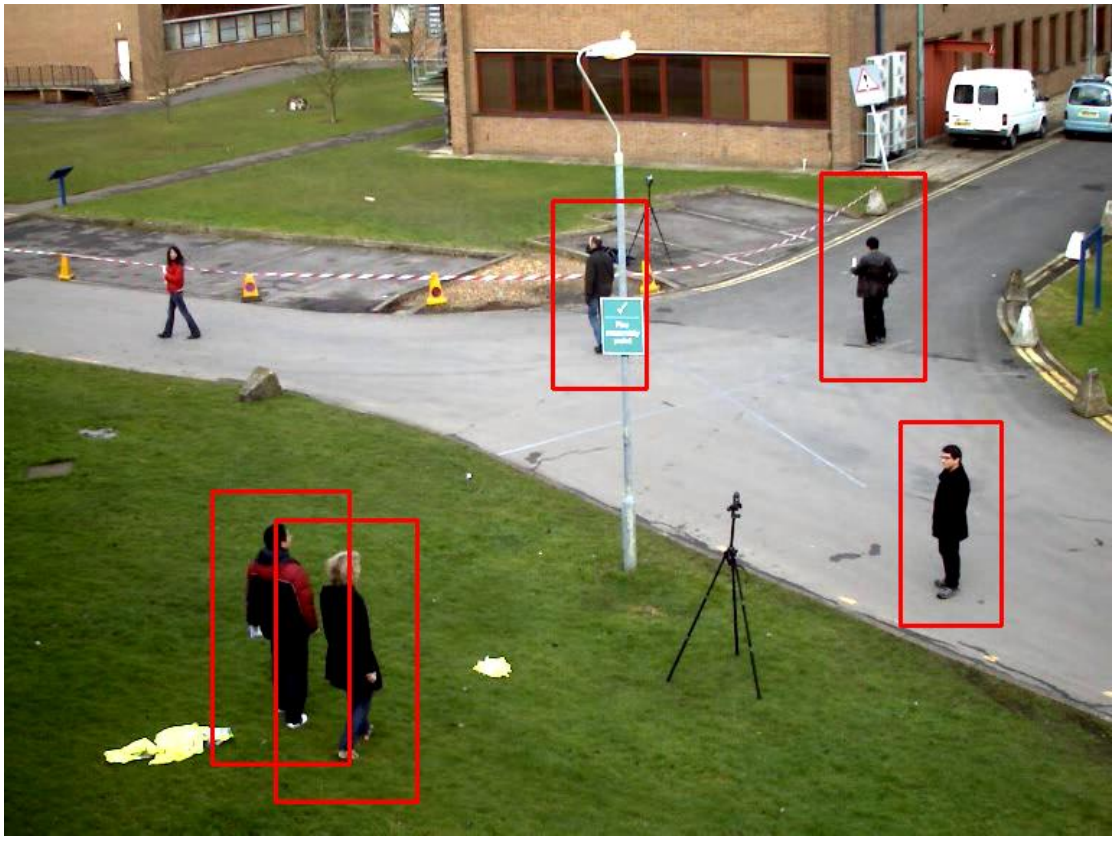




然而利用自举法重新训练出的分类测试结果就很不错了，结果如下(选取均与上例同帧)：

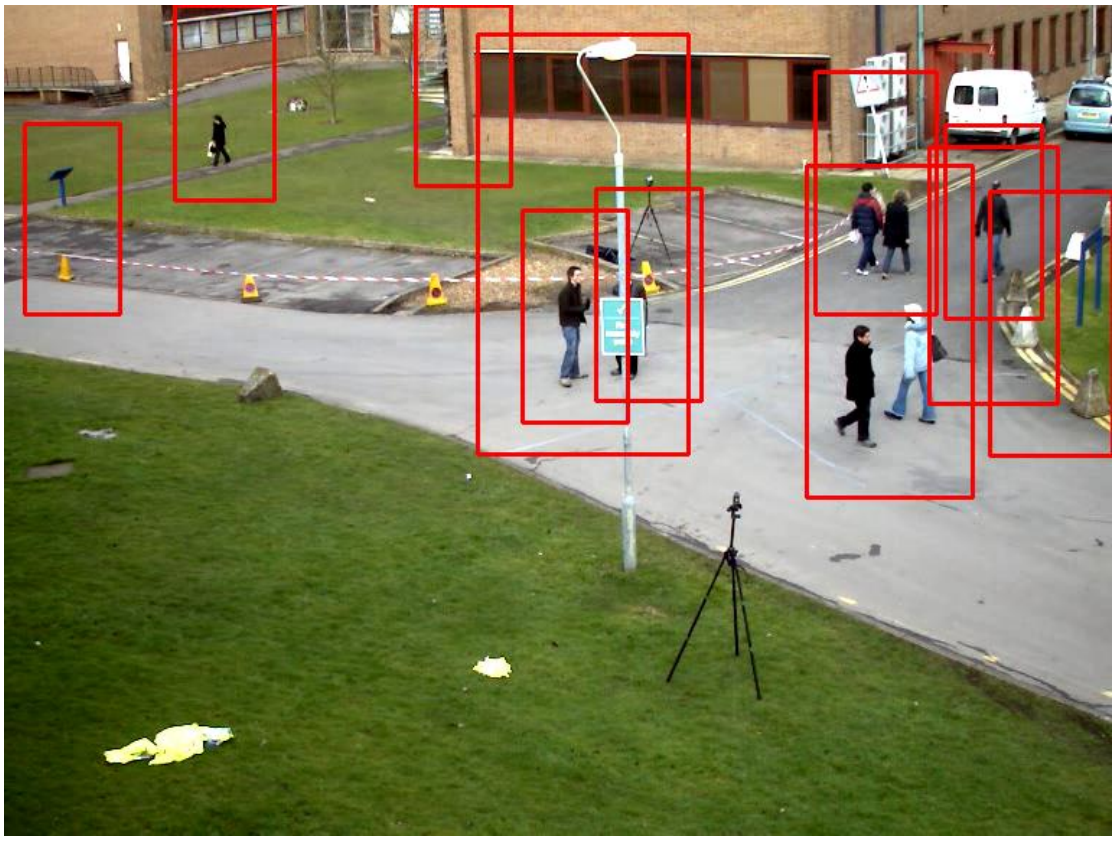






由上述的对比可以看出墙角、电线杆、路牌等均容易被分类器错识成行人，造成误检测。这结论也可被之前利用自举法得到的难例证实。但是自举法有一个问题就是在误识率减小的同时，分类器的灵敏度也下降明显。如下图：

利用自举法后的分类器没有将远处的行人识别出来，但未使用自举法的分类器随误码较多，但灵敏度较高，同一帧就能检测出远处的行人。如下图：



另外，自举法虽然能提高正确率，但是在正样本数量不变的情况下，多次使用自举法对于正确率的提升其实意义不大，反而会大幅降低灵敏度。综合比较来看，自举法使用一次到两次效果最佳。下面是五次自举法检测到的难例个数：

次数	难例个数
1	6151
2	220
3	312
4	184
5	135

可以看出第一次自举法对于正确率提高贡献最大。另外在实验中发现每次运行程序得到的难例个数相同，这也可进一步验证此 HOG-SVM 算法是基于严格的数学运算的。

2、性能分析

本实验最终得到的分类器已经几乎达到在同样的数据集情况下的最优解。无论在图片测试还是视频测试中，都能取得非常不错的效果。但是仍有一些图片不能正确识别或者错误识别，如下图：

我觉得想要解决这种情况需要增加正样本集的数量，或者使用其他的算法。

另外本次实验我使用的语言是 python，因为之前学 opencv 是通过网课学习的，网课教学使用的语言便是 python，python 语言在处理复杂工程时的能力在本次实验中表现的淋漓尽致。我的主函数只花了大约 200 行便完成了功能，如果对应 C++可能远远超过这些。但是在实际应用中，由于行人检测对应现实中的应用往往具有实时性的要求。在运行速度上 python 便不占优势，我的主函数运行时间，经过测算大约为 371 秒，测试单张图片所用时间约为 450ms，实时性不高。因此在编写可用软件时则需要用速度更快的 C++进行。

七、实验心得

本次实验属于计算机视觉类的实验，需要使用 OpenCV 作为主要工具。OpenCV 是一个基于 BSD 许可（开源）发行的跨平台计算机视觉库，实现了图像处理和计算机视觉方面的很多通用算法。在学习 OpenCv 的过程中给我最大的收获就是图像处理的实质还是数学的演算，在计算机视觉领域可以说一切操作都有其数学原理在背后支撑。

实验中我也遇到了很多困难，由于我们班只有我一个人做该项实验，因此也无人交流。但是经过大量的互联网查阅也能最终解决问题，其中最重要的信息来源还是其 OpenCv 官网的信息，官网的 OpenCV library 像一本字典一样，为我解释函数的运行和参数的调配。其中让我最为纠结、耽误我时间最多的问题还是之前提到的数据库所提供的正样本 OpenCv 无法读取的问题，好在最后在别人的博客上获得了解决方法。由此也可见，无论做什么事，交流与合作都很重要。

最后感谢老师给我机会去初步接触计算机视觉这一领域，本次实验让我受益匪浅！

八．参考目录

Histograms of Oriented Gradients for Human Detection, Navneet Dalel,Bill Triggs, CVPR2005

https://opencv.org/

https://docs.opencv.org/3.4.0/d5/d77/train_HOG_8cpp-example.html

https://www.cnblogs.com/hrlnw/archive/2013/08/06/2826651.html

https://blog.csdn.net/amds123/article/details/53696027

https://www.cnblogs.com/zhazhiqiang/p/3595266.html

九、附录代码

1、工具类代码:

（1）获取负样本

```
import cv2 as cv
import random



W = 64
```

```
H = 128
sample_time = 10
name_num = 0

# 创建负样本
str_fa = r'D:/programme/INRIAPerson/INRIAPerson/'  # source 文件路径
str_son = r'C:/Users/WangZhuo/Desktop/Pedestrian Detection/'  # 样本
路径
f = open(r'D:/programme/INRIAPerson/INRIAPerson/Train/neg.lst', 'r')
# source_list 路径
fw = open(r'C:/Users/WangZhuo/Desktop/Pedestrian
Detection/Train_64x128/neg.txt', 'w')  # 样本 list 路径
str_dir = f.read().split("\n")
str_dir.remove(str_dir[len(str_dir)-1])

for string in str_dir:
    map_name00 = 0
    string = str_fa + string

    img = cv.imread(string)
    h, w, c = img.shape
    if (w >= W) & (h >= H):
        while (map_name00 <= sample_time):
            x = random.randint(0, 1000) % (w - W)
            y = random.randint(0, 1000) % (h - H)
            imgROI = img[y:y + H, x:x + W]
            str_son_final = str_son + r'Train_64x128/neg/' +
str(name_num) + r'.png'
            cv.imwrite(str_son_final, imgROI)
            fw.write(str(name_num) + ".png" + "\n")
            map_name00 = map_name00 + 1
            name_num = name_num + 1
f.close()
fw.close()
```

（2）利用 Matlab 获取 opencv 可读的正样本：

```
m=0;
phns ='D:\programme\INRIAPerson\INRIAPerson\train_64x128_H96\pos.lst'
fpn = fopen (phns, 'rt');
while feof(fpn) ~=1
file = fgetl(fpn);
fromname = strcat('C:/Users/WangZhuo/Desktop/Pedestrian Detection/',file);
toname = strcat('C:/Users/WangZhuo/Desktop/Pedestrian
Detection/Train_64x128/pos/',int2str(m)) ;
```

```
toname = strcat(toname, '.png');
i = imread(fromname);
imwrite(i, toname);
m = m+1
end;
```

（3）主函数代码

```python
import cv2 as cv
import cv2
import numpy as np

basic_path = r"C:/Users/WangZhuo/Desktop"  # 文件夹所在目录


def load_images(dirname, list):  # 通过文本目录索引图片，加载图片数组
    img_list = []
    f1 = open(list)
    img_name_dir = f1.read().split("\n")
    f1.close()
    img_name_dir.remove(img_name_dir[len(img_name_dir) - 1])  # 最后一个元素为空
    for string in img_name_dir:
        string = dirname + r'/' + string
        img_list.append(cv.imread(string))
        print("Loading " + string)
    print("Loading fininshed")
    return img_list


def computeHOGs(img_lst, gradient_lst, wsize=(128, 64)):
    hog = cv.HOGDescriptor()
    # hog.winSize = wsize
    for i in range(len(img_lst)):
        print("computeHOGing " + str(i) + ".png")
        h, w, c = img_lst[i].shape
        if w >= wsize[1] and h >= wsize[0]:
            # 产生一个 128x64 的矩形框
            roi = img_lst[i][(h - wsize[0]) // 2: (h - wsize[0]) // 2 + wsize[0],
                  (w - wsize[1]) // 2: (w - wsize[1]) // 2 + wsize[1]]
            gray = cv.cvtColor(roi, cv.COLOR_BGR2GRAY)
            gradient_lst.append(hog.compute(gray))
    print("computeHOGing fininshed")
    # return gradient_lst
```

```python
def get_svm_detector(svm):  # 返回分类器参数
    sv = svm.getSupportVectors()  # get the support vectors
    rho, _, _ = svm.getDecisionFunction(0)
    sv = np.transpose(sv)
    return np.append(sv, [[-rho]], 0)


# train_dir:训练集所在目录
# train_list:训练集目录
# bin：训练所用的 bin 文件
# time：第几次训练
# hard_adr：错误图片所存目录，一般默认
def train(train_dir, train_list, bin, time, hard_adr=basic_path +
r"/Pedestrian Detection/hard_neg/"):
    test_list = []
    hog = cv2.HOGDescriptor()
    hog.load(bin)

    f2 = open(train_list)
    test_list_name = f2.read().split("\n")
    f2.close()
    test_list_name.remove(test_list_name[len(test_list_name) - 1])

    i = 0
    for string in test_list_name:
        string = train_dir + string
        test_list.append(cv.imread(string))
    f3 = open(hard_adr + "hard_" + str(time) + ".txt", 'w')
    print(len(test_list))
    for img in test_list:
        rects, wei = hog.detectMultiScale(img, winStride=(4, 4),
padding=(8, 8), scale=1.05)
        # winStride：步幅  pad：内边距  scale：向外重合多少倍
finalThreshold：探测多少次才允许定结论
        for (x, y, w, h) in rects:
            imgROI = img[y:y + h, x:x + w]
            imgROI = cv2.resize(imgROI, (64, 128),
interpolation=cv2.INTER_CUBIC)  # 将检测错误的地方变成 64x128 存储
            cv.imwrite((hard_adr + "hard_" + str(time) + "/" +
"hard_" + str(time) + "_" + str(i) + ".png"), imgROI)
            print("Writing " + hard_adr + "hard_" + str(time) + "/" +
"hard_" + str(time) + "_" + str(i) + ".png")
```

```
            f3.write("hard_" + str(time) + "_" + str(i) + ".png" +
'\n')
            i = i + 1
    f3.close()
    print("Finished")


neg_list = []
pos_list = []
gradient_lst = []
labels = []
t1 = cv.getTickCount()
# get HOG and Label
neg_list = load_images(basic_path + r"/Pedestrian
Detection/Train_64x128/neg",
                       basic_path + r"/Pedestrian
Detection/Train_64x128/neg.txt")
computeHOGs(neg_list, gradient_lst)
for i in range(len(neg_list)):
    labels.append(-1)  # 负样本用-1 标识

pos_list = load_images(basic_path + r"/Pedestrian
Detection/Train_64x128/pos",
                       basic_path + r"/Pedestrian
Detection/Train_64x128/pos.txt")
computeHOGs(pos_list, gradient_lst)
for i in range(len(pos_list)):
    labels.append(+1)  # 正样本用+1 标识

# get SVM
# 这里的 SVM 初始化采用的 default，参考网址：
https://docs.opencv.org/3.4.0/d5/d77/train_HOG_8cpp-example.html
svm = cv2.ml.SVM_create()
svm.setCoef0(0)
svm.setCoef0(0.0)
svm.setDegree(3)
criteria = (cv2.TERM_CRITERIA_MAX_ITER + cv2.TERM_CRITERIA_EPS,
1000, 1e-3)
svm.setTermCriteria(criteria)
svm.setGamma(0)
svm.setKernel(cv2.ml.SVM_LINEAR)
svm.setNu(0.5)
svm.setP(0.1)  # for EPSILON_SVR, epsilon in loss function?
svm.setC(0.01)  # From paper, soft classifier
```

```
svm.setType(cv2.ml.SVM_EPS_SVR)  # C_SVC # EPSILON_SVR # may be also
NU_SVR # do regression task
print("Training, loading...")
svm.train(np.array(gradient_lst), cv2.ml.ROW_SAMPLE,
np.array(labels))
hog = cv2.HOGDescriptor()
hog.setSVMDetector(get_svm_detector(svm))  # 为 SVM 分类器设置系数
hog.save('myHogDector.bin')
now_bin = 'myHogDector.bin'

train_time = 1 # 自举法训练次数，参考：http://masikkk.com/article/SVM-
HOG-HardExample/
now_time = 1
while now_time <= train_time:

    train(basic_path + r'/Pedestrian Detection/',
          basic_path + r'/Pedestrian Detection/Train/neg.lst',
          now_bin, now_time)

    neg_list = load_images(basic_path + r"/Pedestrian
Detection/hard_neg/" + "hard_" + str(now_time),
                           basic_path + r"/Pedestrian
Detection/hard_neg/" + "hard_" + str(now_time) + '.txt')
    computeHOGs(neg_list, gradient_lst)
    for i in range(len(neg_list)):
        labels.append(-1)
    print("No "+str(now_time)+": Training, loading...")

    svm.train(np.array(gradient_lst), cv2.ml.ROW_SAMPLE,
np.array(labels))
    hog = cv2.HOGDescriptor()
    hog.setSVMDetector(get_svm_detector(svm))
    hog.save('myHogDector_' + str(now_time) + '.bin')
    now_bin = 'myHogDector_' + str(now_time) + '.bin'
    now_time = now_time + 1
t2 = cv.getTickCount()
time = (t2-t1)/cv.getTickFrequency()
print("All Finished")
print("time : %s s" % (time))
```

（4）测试函数代码（视频）

```
import cv2 as cv
import cv2
```

```
t1 = cv.getTickCount()
basic_path = r"C:/Users/WangZhuo/Desktop"  # 文件夹所在目录
hog = cv2.HOGDescriptor()
hog.load('myHogDector.bin')
cap = cv.VideoCapture("1.avi")
i = 0
while True:
    ok, img = cap.read()
    rects, wei = hog.detectMultiScale(img, winStride=(4, 4),
padding=(8, 8), scale=1.05)
    for (x, y, w, h) in rects:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)
    cv.imwrite(basic_path + r"/Pedestrian Detection/Test/Frames/" +
str(i) + r".png", img)
    i = i + 1
    cv.imshow('a', img)
    t2 = cv.getTickCount()
    time = (t2 - t1) / cv.getTickFrequency()
    print("time : %s ms" % (time * 1000))
    if cv2.waitKey(33) == 27:  # esc 键
        break
cv2.destroyAllWindows()
```