PDC（行人检测项目）Demo版工作总结

Author : 马原野
Date ： 2017年5月3日
Version : 初稿

目录

前言

前一段时间完成了PDC项目的Demo版本，主要基于Opencv 2.4.13完成以下工作。

1. GMM进行前景提取;
2. 对活动区域计算HOG特征;
3. SVM对HOG进行分类;
4. 训练SVM分类器从而提高识别率;

识别效果比较一般，初步满足演示需求。现整理一下资料，一则和大家分享一下前一段工作的收获；二则希望大家能够积极的提一些意见，一起讨论学习;

操作环境介绍

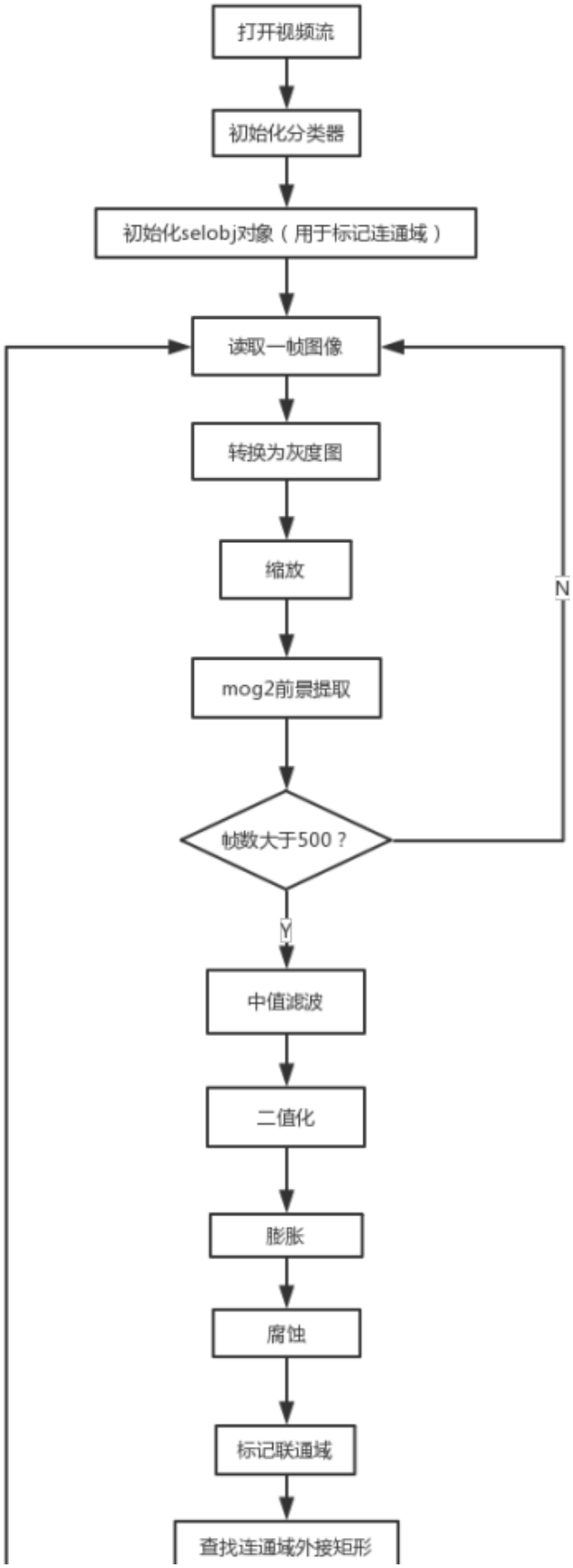
- OS： Ubuntu 16.04 LTS
- Opencv 版本： 2.4.13

工作内容概述

首先跳过编译安装opencv的步骤，如需了解此过程可以查看这里，或者自行搜索相关条目。在树莓派上编译安装opencv操作步骤和PC端相同（也可以用交叉编译的方法，感兴趣的话可以自行搜索）。

程序总体流程图如图1所示。其中

- 500帧的设置是为了mog2进行背景建模，可以根据实际情况进行调节；
- 转换为灰度图并进行缩放是为了减少需要处理的数据量，从而加快处理速度；
- 连通域提取部分用的是伟鑫师兄的算法（在findContours.cpp中），selobj对象保存了连通域标记部分所需要的一些结构；
- 膨胀腐蚀是为了消除目标内部的空洞，核半径可以根据实际情况进行调节；
- 设置ROI之前先进行缩放是为了保证待检测图像和HOG的窗口大小相同，又因为窗口为64*128,因此最好保证检测目标图像大小的长宽比和窗口相同；



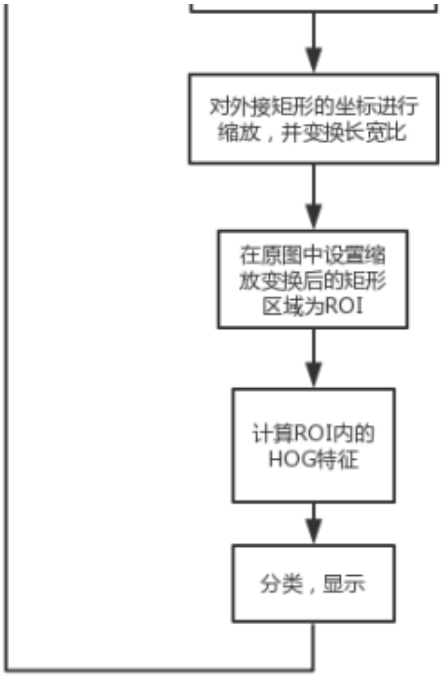


图1 程序总体流程图

标记连通域

由于Opencv提供的连通域标记方法是基于轮廓查找的，不管是在速度、效果、易用性方面都不是很理想，而且恰好伟鑫师兄已经做过了这部分的工作，因此就直接拿来用了。在工程中过封装在src/findContours.cpp文件中。

此部分对外只有一个接口，函数原型如下，其中selobj需要根据待处理图像的大小进行初始化。

```
void OnePass(Mat & fgmask,   //前景图片，由MOG提取

  connectionAreaSelObj & selobj,// selobj对象，保存标记连通域时的临时信息

  vector<Rect> &obj_rect,  //查找到的连通域外接矩形列表

  Mat & noiseSuppressionImg); //经过噪声抑制的前景
```

HOG+SVM简介

关于HOG

关于hog的原理可以看这篇博客，关于OpenCV的实现可以看这篇文章。

我个人的理解是在一张图片中以指定的窗口大小（winSize）滑动，计算窗口内的hog特征，通过对特征进行分类，从而判断有没有人存在。而在窗口内又以块（blockSize）为单位以blockStride为步长进行滑动，每个block又包含若干个Cell，分别在每个Cell中计算九个方向的梯度。具体可以看这篇图文介绍。

则:

一个块（block）包含A=(blockSize.width/cellSize.width)*(blockSize.height / cellSize.height)个胞元（cell），所以一个块（block）含有9A个梯度直方图。

一个窗口包含B=((windowSize.width-blockSize.width)/(blockStrideSize.width)+1)*((windowSize.height-blockSize.height)/(blockStrideSize.height)+1)个块（block），所以一个窗口包含9AB个梯度直方图。

其中winSize等数据在hog初始化的时候指定，因此需要计算9*4*105=3780维的hog特征向量。

```
HOGDescriptor hog(Size(64,128),Size(16,16),Size(8,8),Size(8,8),9);
```

函数原型如下

```
HOGDescriptor(Size win_size=Size(64, 128),
    Size block_size=Size(16, 16),
    Size block_stride=Size(8, 8),
    Size cell_size=Size(8, 8),
    int nbins=9,
    double win_sigma=DEFAULT_WIN_SIGMA,
    double threshold_L2hys=0.2,
    bool gamma_correction=true,
    int nlevels=DEFAULT_NLEVELS);
```

cell中计算方向那里我没有搞懂（cell中的九个向量怎么算出来的？？），小伙伴们有谁懂得话可以交流一下。

关于SVM

关于SVM的原理可以看这篇文章，个人理论水品有限SVM的细节的东西没有搞明白，只大概理解为用梯度下降去使cost函数收敛于某一指定的范围内时，求目标函数的参数W和B。

训练SVM分类器

SVM的训练主要就是依次读取所有正样本，负样本，计算HOG特征后，将特征和分类标

签交给训练器训练(调用svm.train())。这部分代码的src/image_detect.cpp中有很详细的注释，并且在train目录下的README.md中有具体的训练方法，在train/source下有训练代码以及辅助训练的脚本等小工具。

SVM分类器的两种分类方式

第一种是HOG的多尺度检测分类，此种方式需要在hog描述子中设置好分类器，然后可以处理任意大小的图片，在图片中找出人所在位置。此种方法适合在图片中搜索人的位置，执行速度较慢；

```
myHog.setSVMDetector(myDetctor);

myHog.detectMultiScale(img, found, 0, Size(8,8), Size(32,32), 1,05, 2);
```

第二种是单窗口的HOG特征检测,此种方式，单独对窗口区域计算hog特征，并用svm分类器进行分类。此种方式只对窗口大小的图像进行hog特征提取，执行速度较快，实际使用中选用此方法；

```
vect<float> descriptor;

Mat testFeaturMat;

myHog.compute(img, descriptor, Size(8,8));

for (int i=0; i<descriptor.size(); i++)

  testFeatureMat.at<float>(0,i) = descriptor[i];

svm.predict(testFeaturMat);
```

识别速度问题

缩减数据量

程序中比较耗费时间的有，GMM背景提取、标记连通域以及检测目标三方面。这三方面耗时都和图像尺寸等数据量有密切关系，因此首先想到的就是缩减数据量。

摄像头采集的图像为640*480的RGB三通道图像，处理的时候数据量比较大，速度较慢。因此首先将3通道的彩色图进行灰度转换，转换为单通道的灰度图。在再将图像尺寸从640*480缩放到320*240。缩放之前和缩放之后处理一帧图像耗时分别如图1和图2。

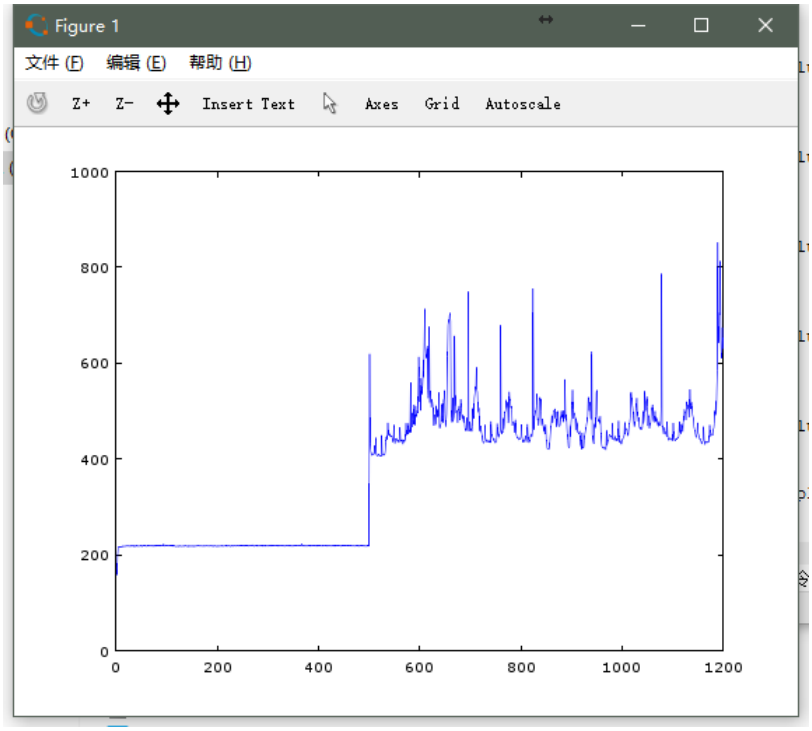


图1 缩减数据量之前处理一帧图像耗时统计（横轴是帧序，纵轴是时间消耗单位:ms）

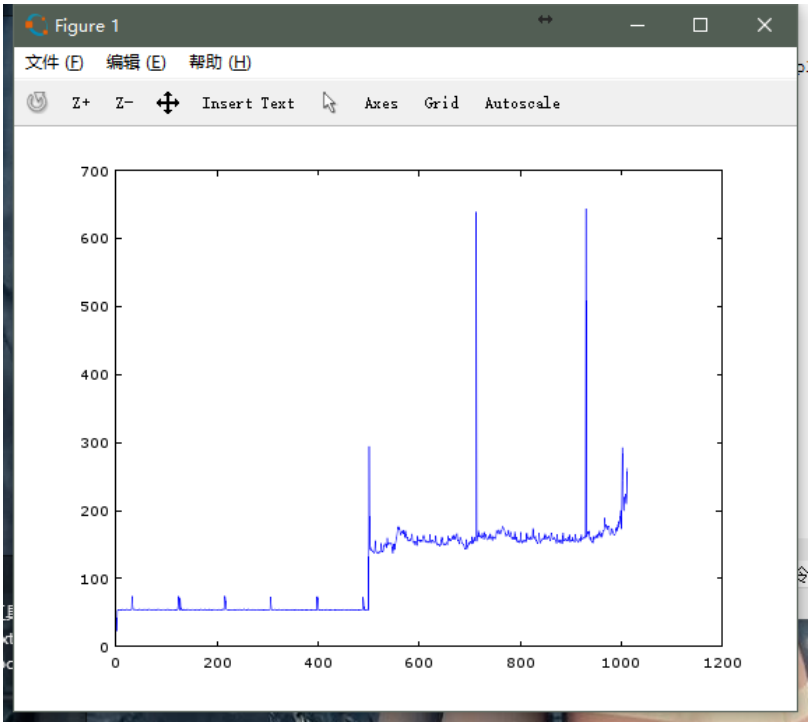


图2 缩减数据量之后处理一帧图像耗时统计

归一化

由于目标大小不一，只能使用多尺度的HOG检测，而多尺度检测是对目标图像进行多尺度的缩放，再用指定窗口在图像上进行滑动分别提取hog特征并进行分类，最后返回目标矩形列表。处理速度较慢。因此对目标进行归一化后，在窗口内提取HOG特征，不进行窗口滑动，进而能节约不少时间。

具体做法如下

- 设置宽高比为1:2；
- 缩放图像到窗口大小：64*128；

其中设置宽高比的步骤如下

1. 找到目标矩形的中心（（x0+x1）/2 , (y0+y1)/2）；
2. 设置矩形的宽是高的一半；
3. 重新为矩形元素赋值；

保存临时文件

调试过程将处理的中间文件保存下来，能很方便的查看程序执行效果，并快速定位问题。保存的临时文件主要包含两部分，处理的实时视频和从原图中抠去的目标图像；

保存视频

视频的保存主要应用Opencv提供的VideoWriter类，但是只能一个窗口一个窗口进行保存，为了将保存实时的处理效果图像，需要将原图、缩放之后的灰度图、二值图等在同一个Window显示并将它们写入同一个视频文件。程序中采用如下方法。

1. 定义一个能够包含原图、缩放之后的灰度图、前景二值图、前景二值图经过噪声抑制之后的图像的大Mat类型；
2. 分别根据图像大小，大图中设置ROI，并将要显示的图像拷贝到相应的ROI中；
3. 二值图需要用cvtColor函数进行色彩空间的转换；
4. 将大图保存进视频文件并在window中显示；

保存目标抠图

将ROI的图像保存为图片文件，可以单独进行后续分析，可以很方便查看目标查找是否准确。

此部分用一个全局变量frame_count保存帧序号；通过sprintf函数生成要保存目标的文件名；最后通过imwrite函数将目标图片保存到指定文件。这里提取的目标也可以作为svm分类器的训练集。

```
sprintf(filename, "./%d.jpg", frame_count++);

imwrite(filename, sRoi);
```

总结

Demo版基本实现行人检测功能，能够在检测的行人时进行开灯等操作。但在识别精度、速度和鲁棒性方面还有待继续完善。而且使用的GMM2背景建模方案对光照等变化较为敏感，容易发生花屏现象。

个人觉得下一步可以考虑

- 使用跟踪识别相结合的方式（TLD）
- 尝试其他前景提取方案
- CNN

测试时保存的视频

4-24-7.avi_20170425_103932.mov

附录

【1】 hog+svm，SVM训练方法
【2】 Opencv 中SVM的介绍
【3】 代码仓库 或者 主仓库