1. 绪论

1.1 课题研究背景及意义

随着计算机科学技术的不断发展，人工智能的概念被提了出来，使得计算机可以在很多场景上代替人类。机器视觉就是人工智能的一个主要应用领域，其主要目标便是通过对图像的处理和分析，模拟人类的视觉识别能力并做出相应的判断。

近年来，随着城市建设的不断发展，以及人们对高品质生活的不断追求，机器视觉领域得到了快速的发展，市场也有巨大的需求。行人检测作为机器视觉领域的一个重要研究方向，该技术在道路交通，智能安防以及娱乐产业等很多领域都有着及其重大的研究意义。其主要应用在以下几个领域:

1. 智能驾驶

我国的汽车保有量增长迅速，相应的交通事故也不断增长，交通事故的主要原因也是人为疏忽导致，行人检测装置检测到行人时就会提醒司机或者自动采取制动措施，从而避免交通事故的发生。行人检测在智能驾驶的作用主要体现在辅助驾驶和无人驾驶上。行人检测技术需要相关的图像采集设备或者雷达等，设备采集到数据后处理器应用相应的图像识别算法分析数据最终得出是否有行人的结论，无人驾驶汽车则要做出相应的措施去避让行人，辅助驾驶系统则通过语音提示的方式告诉司机，司机再做出合适的处理。

1. 安防监控

在很多时候我们需要对特定场合的行人进行监控，比如监狱外围区域，私人住宅的行人进行监控，发现可疑人物就会及时报警；军事基地还可以监控外来人员非法入境；或者监控中发现行人出现异常举动，就可以通知公安、消防等快速介入，保障生命财产不受损失。尤其在金融机构、人员流动大的公共场所更加需要智能化的安防监控系统。早期，监控主要通过人来对显示屏上的图像进行监视，不仅需要大量的人力和物力，而且监控效率并不能得到保证。因此，行人检测技术就有了用武之地，系统可以自动完成行人检测和人的行为分析，实现了对可疑人物、危险行为的准确预警。

1. 机器人导航

通过行人检测技术，机器人可以找到感兴趣的区域，然后导航到目的区域，典型的应用就是现在的无人机可以对行人进行跟随，可以在完全无人操作的情况，主动跟随目标，或者在行人的周围定点环绕。

本文主要研究的是行人检测技术在智能驾驶上的应用，ADAS(高级驾驶辅助系统)现在已经开始广泛的应用在我们的汽车上，为我们驾驶过程中提供了很多辅助功能。行人检测效果图如图1-1，行人识别就是辅助驾驶系统的一个主要功能，行人识别技术应用在辅助驾驶的难点是检测的精度和速度。所以人们对行人检测技术的算法性能提出了更高的要求，在保证检测的精度的情况下，还必须提高检测的速度，保证检测的实时性。在早期提出的算法中，鉴于汽车设备环境的限制，在精度和实时性上面都不适合应用在辅助驾驶上，在最近几年一些优秀的检测算法被提出了，其中基于CENTRIST特征的行人检测算法不管是检测精度和速度上都让行人检测技术应用在汽车辅助驾驶系统有了较好的效果。而且GPU的并行运算可以进一步优化检测的速度，从而保证了检测系统的实时性。

1.2 课题研究现状分析

行人检测算法在最近的十几年发展迅速，国内外的研究人员对行人检测算法的准确性和实时性进行了大量研究，接下来对这些研究工作进行分析。其中主要的研究来自于国内外的高校和科研机构，国内主要有清华大学、南京大学、中科院等，其中中科院计算机科学重点实验室及自动化研究所都取得了巨大成绩。除此之外，国内许多科技公司也在该领域投入了大量的人力物力，其中包括目前比较火热的百度，成立了百度大脑，并将行人检测技术应用在了无人驾驶上，还有类似商汤科技在人脸识别和目标识别领域都取得了巨大成就，并将技术转换为商业产品应用在了我们生活中，为我们的生活提供了便捷。国外的加州理工，麻省理工，谷歌，脸书等高校和企业都在进行行人检测课题的研究；每年都有很多优秀的论文发表在计算机视觉权威的机构上。



图1-1

1.2.1 行人检测算法的研究现状

行人检测目前最常用的方法是基于统计学习的方法，根据大量的样本构建行人检测分类器。提取特征主要有目标的灰度，边缘，纹理，颜色，梯度直方图等信息。分类器主要包括神经网络、SVM、AdaBoost以及当前十分火热的深度学习。统计学习目前的难点有：行人的姿态、服饰各不相同、复杂的背景、不同的行人尺度以及不同的关照环境；提取的特征在特征空间中的分布不够紧凑；分类器的性能受训练样本的影响较大；离线训练时的负样本无法涵盖所有真实应用场景的情况。

2005年法国研究人员Dalal在2005的CVPR发表的HOG+SVM的行人检测算法，基于HOG特征的行人检测提出，使得行人检测领域有了较大突破，HOG+SVM作为经典算法也别集成到OpenCV里面去了，可以直接调用实现行人检测。

基于统计学习方法的行人检测技术很重要的一点就是样本集，而当前关于行人检测主要的数据库有[MIT数据库](http://cbcl.mit.edu/software-datasets/PedestrianData.html)，INRIA数据库，[Caltech行人数据库](http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/)等。其中最常用的是INRIA和Caltech数据库，INRIA数据库是目前使用的最多的静态行人检测数据库，提供原始图片及相应的标注文件。训练集有正样本614张（包含2416个行人），负样本1218张；测试集有正样本288张（包含1126个行人），负样本453张。Caltech数据库是目前规模较大的数据库，采用车载摄像头拍摄，约10个小时左右，视频的分辨率为640x480，30帧/秒。标注了约250,000帧（约137分钟），350000个矩形框，2300个行人，另外还对矩形框之间的时间对应关系及其遮挡的情况进行标注。数据集分为set00~set10，其中set00~set05为训练集，set06~set10为测试集（标注信息尚未公开）。

HOG特征描述了一幅图像的方向梯度信息，提取该特征采用的滑动窗口扫描图像的方式，将窗口划分为大小相同的图像块（block），以块为单位提取图片的特征，这种方法可以保证提取到稳定的HOG特征，HOG+SVM的结合在INRIA数据库上的检测精度接近90%，但是HOG特征的提取计算量高，所以此方法的检测速度很慢，无法实现实时检测。

2009年，Dollar等提出了积分通道特征（Integral channel feature），该方法使用了梯度幅度特征，图像的LUV特征，及梯度直方图特征，为了简化计算使用了积分图像的方法实现特征的快速计算，该方法由于提取了多种特征来描述图像，所以检测精度可以达到86%，而且检测速度也可以得到提高。但是其复杂的特征提取导致检测还是无法做到实时性。

2011年吴建鑫等提出了基于轮廓信息的CENTRIST特征，作者通过大量的实验证明了获取行人边缘轮廓信息就能够准确描述一个行人的特征，而且CENTRIST特征的计算也比较简单，检测效果与积分通道特征相比也相差无几，而且作者在分类器上使用的是级联分类器，首先是一个线性SVM分类器快速找出可能的行人区域，再将结果交给第二层的HIK SVM分类器检测，因此保证了算法的检测精度，所以此方法是一个检测精度和速度都比较可行的一个算法。

1.3 本文组织结构及主要工作

由于主要研究的是行人检测技术应用在汽车辅助驾驶系统上，然而在这种应用场景下，汽车上的设备有限，所以很难做到实时性检测，这就要求我们设计出一套实时性高的行人检测系统。针对这个问题，我们提出了基于BING和改进的CENTRIST的行人检测系统，从而达到实时检测的目的。本文的主要研究内容如下：

（1）第一章绪论介绍了行人检测技术的国内外现状，以及其常见的应用领域，以及行人检测技术遇到的一些难点。

（2）第二章介绍了BING算法代替滑动窗口检测目标的原理，以及BING算法是如何帮助我们减少不必要的重复计算的。

（3）第三章介绍了基于CENTRIST特征的检测算法，并与基于HOG特征的行人检测算法做了比较，证明了CENTRIST特征不但可以减少计算量，还可以进一步提高检测的准确率。

（4）第四章介绍了使用BING和CENTRIST进行行人检测与其他方法在检测精度和速度上的实验数据对比。

（5）第五章是总结与展望，总结了本文的主要工作，对工作中的成果和不足进行了分析，并对未来的发展做了自己的思考。

第二章 BING似物性算法原理

本文提出的是一种结合似物性检测和统计学习的行人检测算法。传统的行人检测算法大都采用滑动窗口产生候选框，然后将候选框的特征输入到训练好的分类器去进行分类，从而判定一个候选框内是否有行人，BING是通过对图像进行预处理，产生带评分的建议窗口通过分类模型进行评估得分，实现检测行人的功能。

3.1 似物性采样的概念

受到生物学的启发，发现人在面对复杂场景时，人眼可以快速，准确的将注意力放在比较明显的视觉对象上，并且优先理解显著性物体。这种基于视觉显著性的物体检测框架受到越来越多关注。算法首先是能够很好地定位图中的显著区域，识别出主要物体。似物性算法通常是采用训练学习或对输入图像定性分析，对不同大小尺寸的窗口进行评分排序，然后提取钱N个独立窗口的坐标信息，最后将得到的候选框交给目标检测算法进行进一步分类。因此，用似物性检测产生的建议窗口，是行人检测预处理阶段的输出结果，使得后面的分类算法的计算量更小。

3.1.1 似物性采样算法分析

目前使用较多，性能较好的似物性算法主要是基于窗口的检测，此方法通过一个公式来打分，根据分数来确定窗口包含物体的可信度，然后把窗口的分数进行排序。下面我们主要会介绍程明明老师提出的BING算法。

BING算法使用梯度作为输入特征，首先训练一个一级分类器，再根据尺寸大小和一级分类器得分训练出36种不同尺寸的二级分类器。BING算法采用了特殊的计算方式，将梯度二值化，使得算法的检测速度加快，实验证明，BING算法可以很高效的产生一系列类别独立，高分辨率的对象窗口，通过使用1000个建议窗口，我们的对象检测准确率高达96.2%，通过增加建议窗口数量或者考虑颜色空间来计算BING特征，实验结果可以提高到99.5%。

3.2 BING算法分析

BING（Binarized normed gradient）是由程明明教授在2014年在CVPR上提出的一种高效的目标建议生成算法，算法的效率可以达到300fps。在VOC2007数据集可以达到96.2%的探测率。

BING算法是基于一个独立事物都拥有很好的闭合边缘特征和中心，相反背景都是杂乱的没有统一特征和纹理。作者提出了无论什么形状和类别的图像，只要是一个独立的物体，那么把它的图像响应窗口调整到一个小的固定大小的尺寸（常用的是8\*8的尺寸），那么它的NG特征（二值化梯度幅值）会有很明显的闭合边缘共性如图3-1所示。而且BING算法采用了一定的技巧使得只需要一些原子操作计算（例如加法，按位移动等）就能完成部分计算。

BING特征启发于能够在识别对象之间感知它的人类视觉系统，因此引入了一个64维的梯度特征幅值特征，二值化的梯度幅值特征（BING）很有效的获取到图像窗口的对象状态。为了能够找到图像中的一般对象，我们扫描一个定义好的量化窗口（依据尺度或者是纵横比）。

每一个窗口通过一个线性模型获得得分：

（1）

*l=(i,x,y)*  （2）

代表过滤器得分，代表NG特征，l代表坐标，i表示尺度，（x,y）表示窗口位置。运用非极大值抑制（NMS），我们为每个尺度提供一些建议窗口。相对于其他窗口（例如：100\*100），一些尺度（10\*500）的窗口包含对象的可能性是很小的。因此我们定义对象状态得分（校准过滤器得分）：

（3）

其中，，针对不同窗口尺度i的窗口，得到不同的独立学习系数。使用校准的非常快的，通常只需要在最终的建议窗口重排之后进行。

3.2 标准梯度计算

标准梯度算子使用的是一个一维离散微分模板在一个方向上或同时在水平和垂直两个方向对图像求梯度，是最简单的梯度计算方法，计算速度较快。比如，我们在计算一幅图的梯度信息时，首先用[1,0,-1]对原始图像进行卷积运算得到水平方向的梯度，同理用算子对图像做卷积运算就能得到竖直方向的梯度，然后进一步简化，在同一像素取水平和竖直方向梯度较大的值作为这一点的梯度。这里我们以Lena图作为例子计算了其梯度特征，如图3-2所示。



图3-2 梯度特征

3.3 训练目标模型和参数

上一节介绍了我们使用的是标准梯度来作为图像的描述特征，下面介绍如何使用梯度特征从训练数据集中训练出一般目标的模型，在这里我们使用的分类器是由台湾大学林智仁设计的一个快速高校训练方法，首先我们需要准备好训练数据和训练模型以及参数。

训练数据中包括正样本集和负样本集，这里我们使用的是VOC2007数据库中的图像作为我们的样本库，数据库已经标定好了每幅图中目标的坐标信息。以这些准确的目标框为基础，以下面的计算公式为准则，就得到正样本图像。

>=50%

负样本则是通过在每幅图的背景中随机产生的，作为负样本不能包括目标或者不能和目标有过多的重叠的信息，所以采用下面的公式作为准则，符合标准的框就留下，不符合的就直接舍弃了。

<=50%

按照这两个不等式的判断，训练样本图像集就分离成了正负两个样本框集合，按照下图3.3中的步骤对训练数据集进一步处理得到可用的训练数据。



图 3.3 生成正负样本数据集

按照此准则我们将VOC2007样本集就分离成了正负样本图像的集合，最后分别把这些正负样本图像缩为8\*8的大小，并计算其梯度值，最终得到了训练的样本数据分类器采用的是Liblinear,相对与Libsvm，它的计算复杂度低，训练时间少。在大量数据的情况下，Liblinear和Libsvm性能相当。所以这里使用的是Liblinear来进行分类。训练过程分为两步，级联的第一级SVM可以对候选窗口进行粗过滤，不区分候选框的大小，仅仅是根据正负样本生成的训练数据，通过SVM分类器训练出目标模型w，这是一个64维向量。

第二级SVM则是针对不同大小框包含目标的概率不同而训练出的参数。所以在预测图像中是否有目标时会将图像进行一系列缩放，生成很多大小不同的子图，如图3-6所示。



图3-6 经缩放的一系列子图

由图可知前面的方正的子图中能检测到包含目标的框的可能性肯定比细小的子图要大。所以为了能够在提供较少候选框的前提下，使候选框包含目标的概率更高，就要对缩放后的各个尺度的子图训练相应的参数，使得在不同子图中的预测框分数加入一个惩罚项，因此我们需要训练每个尺寸对应的参数和。因此整个检测过程就是首先由w模型计算每个预测框的得分，再根据对应子图尺寸的参数进行校正，使得概率大的变得更大，概率小的变更小，使得检测的质量提高，最后通过NMS筛选得出最终的预测框。

3.3 二值化

为了使得计算速度更快，我们使用了二值化的方式进行了优化，将标准梯度特征和训练出的模型分别进行二值化，使浮点型数据变成0和1的序列，这直接降低了计算难度，用简单的位操作就可以完成计算。

由Liblinear分类器训练出的模型w是一个64维的向量，这个向量有64个浮点数组成。而浮点数的计算比较麻烦，所以需要想办法对数据进行处理，降低计算量。我们采用了二值化近似w:首先将模型w用一组基来表示(这组基一共个)，的值直接影响计算量的大小，因此我们取，为了实现w的简化，将64维向量全部用0和1来表示。具体的推导过程如下：

(4)

(4)式将w模型二值化近似，其中，。现将64维的浮点数根据式子(d)和(a)转化为-1或1。

, 1 (d)

, -1 (d)

基底可以转化为如下所示：

因此将式子(3-12)带入到(4)中就可以得到：

此时浮点型的64维模型就被二值化近似为了0和1的组合，刚刚好64维的0或1数字可以被保存在一个int64的整型数据中，每一位存储二值化后对应位的数值。那么检测一个框中是否有行人的公式就要写成如下形式了：

梯度值的二值化：计算梯度时得到的每个点的梯度值就是一个0~255的一个整数，用一个字节来存储，字节型数据有下面的特点，左边位的权值越大，右边位的权值越小，也就是说高位的值才是决定字节型数据的重要因素，所以在这里就用几个高位的值来近似一个字节的数的值。图像中某一位置L处的NG特征计算出来是一个8\*8的框，如图3.12所示：



图 3.12 NG特征

可见NG特征按位平面分层，并对其二值化，位置l处的BING值就可以近似如下式所示：

由于每个位对应的权值不同，所以针对不同的位平面要乘上对应的权值。表示位置l处的BING值。通过二值化处理后，模型w就是一个int64的整型数据，NG特征可以用BING近似。然而对于每个BING值要分别找出各个位平面上的8\*8的正方形框，就要对图像的64个值进行遍历。有一种巧妙的方法来计算这个值，我们发现，相邻点的BING值有一定的关系，可以通过位操作滑动的方式，求出每个位置各个位平面上的8\*8的框，而无需遍历。

图3.13中的红色方框表示点L处第k个位平面的BING值，记作，绿色框表示框的最后一行，记作。蓝色框就是点L，这样我们就可以用一个简单的64位整型变量和byte变量来表示BING特征和它的最后一行，这样使得特征计算更有效。



图3.13 L点的BING值



(a) (b)

图3.14 相邻点间的相关性

由于相邻的BING值之间有一定的联系，例如图3.14(a)所示，位于(x,y)点的BING值与它相邻的(x,y-1)除了最后一排的数据不同之外，其余7排是完全一样的，所以当直到点(x,y-1)的值就可以通过移位操作求出(x,y)前56位的BING值。同理和点(x-1,y)的最后一排就只相差一位，如图(b)所示，黄色的框左移一位，右边补上点(x,y)在这个位平面的值，于是求出了

。综合以上两步就可以求出(x,y)出的BING值，就不用遍历64位值。具体的算法表示如下图3.15：



图3.15 BING值计算方法

按照以上介绍的二值化方法处理后，目标模型w，以及待检测图像的每个点的BING值都可以用64位整型数据来表示，每个点的评分就是w与求内积。

3.4 过滤候选框

经过预测后，第一级的模型w会对每一个框进行评分，分数的大小表示这个框包含目标的概率大小。然而BING算法使用的是8\*8模型 ，所以会产生很多的框，必须通过一定的方法来把一些窗口给过滤掉，否则的话检测速度不会有明显提升。

非极大值抑制是(Non-maximum Suppressin)是由Alexander Neubeck和Luc Van Gool提出的一种高效过滤方法，该方法在局部搜索极大值。局部区域代表一个邻域，邻域有两个可变的参数，一个是邻域的维数，另一个是邻域的大小。非极大值抑制被应用在了许多计算机视觉算法中，如：3D重建，视频跟踪，目标识别等。在BING算法中，滑动窗口提取特征经模型w计算后，每个窗口会得到一个分数。但是滑动窗口明显会产生许多没用的窗口。这个时候就需要在一定邻域中找出分数最高的那些窗口，然后舍弃分数低的窗口。算法的步骤如下：

(1) 把BING算法计算后的输出的预测框，按照分数高低进行降序排序。

(2) 每个框对应一个标记信息。

(3) 当得到一个预测框时，先查看其标记值是否为1,如果为1，则在其4\*4的邻域内，全部标记为0。

(4) 最后对所有预测框中标记为1输出，否则就舍弃。直到获得所有的框。

BING算法，由于使用了二值化加速，同时使用二值化梯度特征值的计算过程。所以作者测试速度可以达到300fps，所以这样的处理速度可以满足实时性的要求。下图3-9是BING的检测效果。



图 3-9 BING检测效果

3.5 BING算法应用于行人检测

在之前的章节，我们提到了传统的HOG+SVM的行人检测算法，现在我们做了实验来看一下其效果。我们直接用的是OpenCV自带的关于HOG和SVM的库函数，在一台普通的PC机上运行得到的结果如图3-10所示。



(a) (b)

图3-10 行人检测测试图和检测结果

从图（b）的检测结果来看，图片中有两个人，但是OpenCV的检测算法只能够检测出其中的一个人，另一个人由于和背景区分不明显，所以没有能够被检测出来。接下来我们使用BING算法先对图片进行预检测，再在预检测的框内使用HOG+SVM对人进行检测，最终得到结果如图3-11所示。图(a)表示BING检测得到的建议框，(b)是在预检测框内检测出的行人。从图中我们不难看出，经过BING算法预处理过后，第二步检测的目标区域就缩小了很多，所以把BING和行人检测算法结合起来就可以在速度和准确性上有较好的提升。



(a)BING预检测结果 (b)HOG检测结果

图3-11 BING+HOG检测效果

3.6 本章小结

本章详细介绍了似物性检测算法BING(二值化梯度幅值特征)，采用预测框策略代替传统方法中的滑动窗口方法，先从图片中提取可能的目标窗口。BING是利用一般目标都具有封闭曲线特征，同时结合目标在梯度空间的共性，将细节丢失后，闭合曲线或多或少像个圆形，而非目标并没有这个特点。BING算法包含计算梯度特征，训练模型参数，二值化，非极大值抑制这几个主要步骤。并通过实验数据证明BING算法对物体的检测有比较好的效果。

第二章 基于CENTRIST特征的行人检测算法原理

2.1引言

本章首先阐述了CENTRIST特征的特点，以及基于CENTRIST特征的行人检测算法的主要步骤，首先是如何提取图像的CENTRIST特征，以及采用级联分类器对图片进行滑动窗口检测，检测图片中是否有行人，最后比较了CENTRIST特征和HOG特征在检测准确率和速度方面的差距。

2.2 CENTRIST特征简介

2.2.1 CENTRIST特征的原理

CENTRIST（C4）特征能够很好的描述行人的特征，而且特征的提取相对传统方法计算更加简单，待特征提取完毕后就可以直接交给分类器进行判断的出结果。C4在目前最高精确度下可以达到20帧每秒的检测速度，并且是在只使用一个处理线程和不使用GPU等硬件的情况下达到的。能达到实时而精确的检测源于以下两点：第一，相邻像素差值的符号是描述轮廓的关键信息；第二，CENTRIST描述算子非常适合做人体检测，因为它编码了符号信息并且可以隐式地表达全局轮廓。使用CENTRIST描述子和线性分类器，我们提出了一种不需要显式生成特征向量的计算方法，它不需要图像的预处理或特征向量的归一化，只需要O(1)时间去测试一个图片区域。

C4特征是针对行人的局部轮廓提出的，相邻像素的大小关系是编码的关键，具体像素的值对该特征的计算影响并不大。图片中的行人的衣服上的图像，绘画等纹理性特征对行人检测有较大影响。因此首先用Sobel边缘检测算子对这些纹理性的特征进行消除，同时又能较好的保留行人的整体轮廓。所以我们使用Sobel算子先对图像进行平滑处理，用Sobel梯度值代替像素的大小，这样就可以消除这些局部的高频纹理特征带来的不理影响。因此通过Sobel算子平滑后的原始图像*Ｉ*可以得到一个新的图像*Ｉ*’ ，*Ｉ*和*Ｉ*’的对应关系如下图2-1：



图 2-1

如图2-2所示，原始图像（a）经过Sobel进行平滑处理后，图像很好地过滤掉了一些不重要的纹理特征，（b）很好的描述了行人的边缘轮廓信息。



图 2-2

C4特征中一个重要的概念是CT（Census Transform）值，图像上某个像素的CT值由该像素的灰度值和其九邻域的像素的灰度值比较而得来，具体计算的方法如图2-1所示。若相邻的像素的灰度值比中心像素的灰度值大，则把这个位置的值置为0，否则置为1。然后将邻域内的八个数字按照从上到下从左往右的一个顺序排列起来得到一个八位的二进制数，最后将这个二进制数转为十进制数，这个十进制数就是中间那个像素的CT值。而且由于是一个八位的二进制数得到的，所以这个CT值的取值范围就是0到255。



图 2-3

我们通过计算CT值来代替图像的灰度值，从而得到新的图像，我们称为CT图像。为了更好的描述，在遍历一张图像时，使用一个256维度的CT值直方图来统计每个CT值的在图像中出现的频率。最终得到的这个直方图就称之为CENTRIST特征。

在实际应用中，把一个图像分为若干个小的图像块，我们取的是一个108\*36大小的检测窗口，然后把这个块分割为9\*4的图像小块（block），每2\*2的邻接小块组成一个超级块（super-block），所以通过计算得到一共有24个超级块，计算特征时就以一个超级块为单位，则整张图片的CENTRIST描述特征就是256\*24=6144维大小的直方图。（由于CT值的计算是与3\*3的模板进行匹配，因此超级块的边缘像素不会参与CT值的计算）。

2.2.2 CENTRIST特征的计算过程

上一节我们介绍了特征的计算原理，本节我们会介绍提取一副图像的CENTRIST特征的过程，包含以下步骤：

1. 读取原始图片到内存中；
2. 采用滑动国定大小的检测窗口对图片提取CENTRIST特征。但是由于真实图像中行人的大小各异，为了能够不漏检，因此我们必须要对图像进行缩放，使得各个尺度的行人能够被滑动窗口检测到，如图2-5所示，红色的矩形框就是检测窗口。
3. 使用Sobel算子对图像进行处理，构建Sobel图像；
4. 根据前面介绍的 计算CT值的方法计算CT图像，从而提取行人的轮廓信息；
5. 检测窗口在CT图像上滑动，并计算得到窗口的CENTRIST特征；

整个流程如图2-6所示，接下来我们会介绍如何利用我们提取到的CENTRIST特征对行人进行检测。



图 2-5



图 2-6

2.3 级联分类器

基于CENTRIST特征的行人检测算法使用了线性SVM分类器和HIK SVM分类器串联得到的级联分类器来实现快速、准确的检测。首先要对分类器进行训练。

2.3.1 SVM原理

支持向量机（SVM）是90年代中期发展起来的基于统计学习理论的一种机器学习方法，通过寻求结构化风险最小来提高学习机泛化能力，实现经验风险和置信范围的最小化，从而达到在统计样本量较少的情况下，亦能获得良好统计规律的目的。通俗来讲，它是一种二类分类模型，其基本模型定义为特征空间上的间隔最大的线性分类器，即支持向量机的学习策略便是间隔最大化，最终可转化为一个凸二次规划问题的求解。2005年，Dalal等研究员，将HOG特征和支持向量机结合起来应用在行人检测取得了巨大成功。

支持向量机的基本思想：定义一个线性最优超平面，使得超平面可以很好的将数据集分开。如图2-7是一个线性二分类的例子，一个二维平面，平面中有两种不同的点，分别用两种不同的颜色表示，一种为红色的点，另一种为蓝色的点，红色的直线是一个超平面，由图可见直线将平面内的点完全分开，这条直线就是我们要求解的超平面，下侧的全是-1，上侧的全是1。



图 2-7

接着我们可以假设超平面为：

其中 是法向量，为偏置项，()决定了超平面方程，样本到超平面的平均距离为：

如果()表示的超平面可以对样本集进行分类，即，如果=1则，

如果=-1，则。令

所有满足上公式取等号的样本点我们称之为支持向量，正负两类的支持向量到超平面的距离之和为:

其中为点到平面的间隔，为了是最大，就是求满足公式的前提下求解最优参数()，也就是如下所示

*s.t.*

因此当最大，就可以使得分类间隔最大。等价于让取最小值，因此公式(4-5)可以改写为：

*s.t.*

因此问题就转化为了求在约束条件下求最优解的问题，而且这是一个凸二次优化问题，我们使用拉格朗日乘子法和KKT条件来求w和b，首先构造拉格朗日函数：

其中a=为拉格朗日乘子向量。根据拉格朗日对偶性，原始问题的对偶问题是极大极小问题：

所以，为了得到对偶问题的解，需要先求对w,b的极小值，再求对a的极大值。首先求，我们对拉格朗日函数分别对w,b求偏导并令其等于0，因此求解等式就得到了：

*w=*

*=0*

再将以上两式带入拉格朗日函数之中就得到:

接下来就是在对上式求其对a的极大值，因此最终问题就转换为了求下式的解的问题了

*s.t. =0*

而求解此问题就需要用到SMO算法了，SMO算法是支持向量机学习的一种快速算法，其特点是不断地将原二次规划问题分解为只有两个变量的二次规划子问题，并对子问题进行解析求解，知道所有变量满足KKT条件为止，这种启发式的方法得到原二次规划问题的最优解，因为子问题有解析解，所以每次计算子问题都很快，虽然计算子问题次数很多，但在总体上还是高效的。关于SMO算法的具体步骤此处就不做多的介绍了，如需了解请查阅相关资料。假设已经求得了对偶最优化问题的解a=，就可以根据式2-5和2-6求到(w,b)的解。因此就可以将超平面写成

*=0*

分类决策函数就可以写成：

*f(x)=sign()*

由此方程式可以知道分类决策函数只依赖于输入x和训练样本输入的內积，此式子被称为线性可分支持向量机的对偶形式。因此可以得到这样的结论，对于线性可分训练数据集，可以先求解对偶问题的解a，再求解原始问题的解w,b；从而得到分离超平面及分类决策函数。这种算法称之为线性可分支持向量机的对偶学习算法，是线性可分支持向量机学习的基本算法。

之前我们介绍了支持向量机的基本原理，但是对于线性不可分的样本集，我们需要将数据集映射到高维空间去，从而将数据线性分开，但是由于映射到高维空间，数据的维度会爆发式的增加，使得计算量很大，所以需要引入核函数来降低计算量。那么接下来就是要在高维空间学习找到线性可分的超平面，经常使用的核函数主要有以下几种：

（1）线性核函数：

（2）多项式核函数：

（3）高斯径向核函数：

（4）Sigmoid核函数：

其实核函数不只以上介绍的这几种，只要改核函数是一个正定核，那么就可以把其作为支持向量机的核函数。

2.3.2 HIK SVM原理

在目标检测领域，最重要的应该是分类器模型，优秀的分类器模型对系统能够出色完成分类任务是最关键的一步，之前SVM取得了较好的分类效果，也成功应用在了目标检测的众多领域。但是在实际应用中，许多情况核函数的选择才是关键，因此如何选择核函数是分类模型的重要一步。

当前在计算机视觉领域，人们大量运用直方图作为图像特征描述符。其中以直方图交叉核函数（HIK）作为核函数的SVM分类器被证明在处理直方图特征的分类能够获得更好的效果。直方图交叉核成功的原因主要有以下两点：

（1）第一，在计算机视觉领域，有很多特征描述都是使用直方图结构进行表述，如常见的：SIFT，HOG，CENTRIST特征等，由于直方图简单明了，可以有效的统计特征，因此获得了广泛的应用。直方图交叉核，以下简称HIK，可以用于比较直方图之间相似度，已经被证明比其他常用的方法有着更加好的效果。

（2）第二，在常用的分类器模型中，SVM的训练速度快，需要的样本数据量小，分类效果好，但是缺点是分类速度慢，而决策树模型则在分类速度上有着明显优势。然而，使用HIK核函数的SVM则兼具两者的优点。

在近年来基于HIK的SVM分类器被广泛引用在了许多快速检测方法中，首先介绍以下直方图相交核函数相关的公式：

*=*

上式就是直方图交叉核函数的数学表达式，其中直方图相交就是在求两个直方图的对应bin中的较小的值并把所有比较得到的较小值累加起来就得到了最后的结果。由于最后的求和相当于对其归一化的得到的结果，所以的结果越大就表明两者之间就越相似。正式因为直方图交叉核函数在直方图匹配运算时的优越性，所以近年来很多快速检测算法都使用基于HIK的SVM分类器。

2.3.3 直方图相交核函数的半正定性

要使用HIK作为SVM的核函数就必须证明核函数必须要满足正定性。Boughorbel证明在正实数的情况下HIK是正定的，下面我们将介绍HIK的半正定性。

首先假设非负的实数集{x>=0|x},接下来需要证明HIK核函数具备半正定性n表示样本点的个数，d表示特征的维度，作为特征的第j维向量。首先当d=1的时候，有n个样本点，，…,,并且假设其中,所以核函数有以下性质：

由此可以得到A=,其中R和A的定义如下：

其中A是通过K变换得到的一个半正定的矩阵，其中R满足半正定性，因此K也是半正定的。

2.3.4 HIK的快速计算

SVM采用的是监督学习的方式训练分类器，因此需要构造训练集N个样本{}，其中,而整个训练过程就是在无论多少维的空间寻找最大间隔分界平面。而求解SVM的过程我们在上2.3.1已经介绍了SVM的原理，我们得到最终求解的判别式是：

h(x)=

根据上一节得出的结论，非负实数下的HIK核函数满足Mercer定理，我们使用一个比较直观的方法表示直方图的数值。对于一个维度为d的直方图x，每个维度的数值都小于一个上界v，因此我们可以用一个二值化的向量B来表是直方图的一个维度。

B(x)=[1,1,…,0,0]

由B(x)可知整个直方图的特征空间就被扩展成了d\*v维，原有的直方图就转换为

B(x)=[B(), B(),…, B()]

扩展后的特征空间表示形式，方便了直方图相交中计算两者间最小值的方式，直方图相交核的计算值就可以转换为。这种特征空间的扩展方式使得核函数值的求解变得很简单。本文通过HIK SVM分类模型的检测阶段的判别式，证明整个直方图相交核的快速计算方法。我们假设要检测的特征向量是q，带入式得到:

*f(q)=*

在实际的计算过程中，由于输入的特征向量需要与每个支持向量进行计算，假设有n个支持向量，且d维的支持向量，就得到*o(nd)*的检测时间复杂度。为了达到快速检测的效果，我们引入了一个T表，我们令,，最终得到表达式：

*f(q)=*

我们发现经过T表简化后的决策函数只需要O(d)的时间复杂度就可以得到结果，在通用的SVM中我们得到的是系数w，所以判别式也可以表示成：

*f(q)=*

有1,2二式，可以得到：，由此计算时就可以通过T表直接得到的值，所以此方法显著提高了分类器的检测速度，而此处的T表也就是我们接下来训练的基础。通过一系列转换，最终通过快速检测T表替代了原有的w系数向量。使用T表的目的是在检测过程中不用计算输入向量与每个支持向量的核函数计算结果，通过查表的方式就可以得到检测值。T表本身是通过近似得到的，所以直接计算输入特征向量每一维在T表中的关联值，并将d维的值累加起来就得到了检测的最终结果，因此检测过程的时间复杂度就由原来的O(nd)提高到了O(d)。

2.3.5 HIK SVM快速训练算法

在前面我们介绍了通过建立一个T表来提高HIK SVM的检测速度，本节我们将介绍如何提高HIK SVM分类器的训练速度，如何建立T表来减少运算，此算法(ICD)是吴建鑫在2010年提出的。

其实HIK SVM的快速训练法还是在原理传统的SVM训练方法的基础上进行了适当的改进，具体实现还是在通过对偶问题求取极值得到最优解，所以引入直方图特征扩展的方式，SVM的优化公式就变成了：

*w+C*

其中是基于L2范式的代价函数，参数C就是控制误差和最大间隔分界平面在真个样本空间的均衡值。

将原始问题转换为求解更容易的对偶问题，则原始问题3-23转换为等价的对偶形式：

*Q’a-*

*s.t. 0*

其中*Q’=Q+D,=*，D是对角矩阵，。在线性核函数的情况下求解对偶问题的最优化问题最终会得到支持向量a的值，每一次迭代都会使g(a)逐渐变小。同样的，此过程也是求解g(a)的极小值时，最终得到所需要求解的支持向量。具体的算法实现如图3-2。

然而由于HIK SVM的核函数不同，并且为了保证检测和训练速度，所以采用了特殊的优化方式，我们在原来的算法基础上替换了3-2中的1，4，9行部分，把T表嵌入到整个优化过程中去，替代w和向量B()。将T表带入到算法中后，可以使原始算法中初始化部分简化掉，整个过程就只需要对T表进行迭代插值，不再需要考虑向量w和向量B()之间的乘法运算。这样就很大程度地提速了整个优化过程，经过实验证明，使用T表还可以使得算法的迭代次数明显降低，从而进一步加快训练速度；并且也不需要额外的存储区来保存支持向量，如图3-3表示了具体的替代部分计算方法。



图 3-2



图 3-3

2.3.3 训练级联分类器

(1)线性SVM的训练

训练分类器时正样本集P我们用的是108\*36的包含行人的图像，负样本集我们使用的是同样大小的不包含人的图像，把正负样本集输入训练线性分类器训练分类器H1，然后用H1分类样本集N，从而获得一个新的负样本集，再次使用P和训练得到分类器H2，如此反复训练多个分类器，使负样本N中的所有区域都被至少一个分类器认为是负样本。然后使用P和所有的负样本集训练最终的支持向量机分类器。

(2)HIK SVM的训练

CENTRIST特征是一种直方图特征，所以使用更适合处理直方图的直方图交叉核支持向量机相比与线性分类器可以获得更好的分类效果。在训练过程中，采用前面线性分类器检测样本集N得到的一个新的负样本集，然后用正样本集P和训练得到HIK SVM。

2.3.4 使用级联分类器做行人检测

我们将提取的CENTRIST特征输入到训练好的级联分类器中，然后分类器会输出检测区域是否有行人。检测过程主要有以下两个步骤：

(1)级联分类器的第一级分类器是线性SVM分类器，检测过程还是使用滑动窗口的方式进行，每经过一个滑动窗口，分类器就会判断窗口是否有行人，如果有行人就将该窗口的位置坐标记录下来，最终将整张图像包含人的窗口保存起来。

(2)级联分类器的第二级分类器是HIK SVM分类器，HIK SVM会对第一级分类器检测有行人的窗口进行进一步检测。同理，若窗口不包含行人，窗口就向后滑动，如果有行人就记录窗口的位置，并最终输出。

采用两级分类器的主要目的就是提高分类的准确性，因为线性SVM的分类能力有限，但却可以将大部分不包含行人的窗口快速排除掉，同时保留几乎所有包含行人的窗口，而HIK SVM则可以较为准确地对图像进行分类，所以使用两级分类器可以达到较好的检测结果。

2.3.5 如何使用线性SVM进行快速行人检测

假设有一个我们已经训练好的线性分类器，由于C4算法的特征可以分为8\*3个超级块，所以可以把划分成24个，1组成。对应地C4特征也可以被划分为24个小单元。传统算法是将分类器模型与特征向量f进行内积运算，如下所示：

如果结果大于0就表示图像块内包含行人，小于0则不包含行人。计算公式(2-1)需要提出特征向量f，但是进行这个计算会很耗时，Lampert等则提出了新的方法【】，该方法计算复杂度较低。假设检测窗口的尺寸为(h,w)，进一步将检测窗口划分为9\*4的单元格cell，每个单元格的大小为()=()，把四个单元格合并为一个图像块。原始图像用O来表示，其对应的Sobel和CT图像分别用S和C表示，(t,l)则代表检测窗口左上角在图像中的位置，所以公式2-1的就可以变换为：

此时表示其低k个维度的值，而且k的取值范围为[0,255]。因为每个检测窗口有36个单元格，而相邻的四个单元格为一个块，所以每个窗口有24个块。另外每个块内，计算时排除了每个块的边缘像素。为了简化对公式（2-2）的计算，我们需要引入一个辅助图像，这个辅助图像使用A来表示，我们定义辅助图像像素点的灰度值为：

=

因此公式（2-2）就变为了：

观察上式可以直到，每个块内的计算可以通过简单的算术操作就可以完成，这样就使得只用将辅助图像的值累加起来就完成了计算，而不用显示地提取特征向量。为了进一步将工作简化为一副辅助图像，定义：

最终就得到了如下公式：

此时只需要通过一副辅助图像就能计算出结果了，这明显加快了计算速度并减小了内存开销。

2.3.6 基于CENTRIST和HOG特征的行人检测对比

HOG是计算机视觉领域我们常用的一种描述图像局部纹理的特征，它强调的是目标的局部纹理，而CENTRIST特征则主要针对的是行人的边缘轮廓，并且可以过滤掉对行人检测不利额高频局部纹理性特征，所以与HOG相比C4算法在检测精度上效果更好，另一方面，提取HOG特征的时候，需要在图像的预处理操作和归一化操作上花费较多的时间，而C4算法则不需要，使得C4算法在速度上更具优势。

我们使用C4算法和opencv自带的HOG算法对同一副图像进行了检测结果如图2-9所示。被检测的图像有4个行人，HOG算法得到的结果出现了三次漏检，两次误检。相反使用C4算法则把四个人全部正确检测出来。因此，我们有理由相信使用C4算法进行行人检测能够取得更好的精度。在文献【4】中数据表明FPPI等于1时，HOG算法的检测精度只有74.4%，C4算法则达到了83.5%，这正一步证明C4算法更适合用于行人检测。



(a)HOG检测结果 (b)C4检测结果

图2-9

这两种算法在英特尔i5处理器的计算机上处理不同尺寸的图像的检测速度，实验结果见表2-1。从表中的结果可以发现不管什么尺寸的图像，C4算法的速度都更快，特别是处理尺寸为640\*480时，更是达到两倍于HOG的速度。

|  |  |  |
| --- | --- | --- |
| 图像大小 | HOG算法处理速度(fps) | C4算法处理速度(fps) |
| 320\*240 | 20.5 | 31.6 |
| 640\*480 | 3.5 | 7.5 |
| 1280\*960 | 1.0 | 1.6 |

表2-1

根据以上实验结果可知，为了得到更好的行人检测效果，我们应该使用C4作为我们的检测算法，因为不管是精度还是速度在目前的传统机器学习方法中，C4的表现都算很优秀的。

2.5 本章小结

本章主要介绍的是基于CENTRIST特征的行人检测算法的原理，主要包括特征提取和使用级联分类器检测行人两个部分。首先我们介绍了C4特征的提取方法，包括使用Sobel边缘检测算子对各个尺度的图像进行计算，得到对应的Sobel图像，再计算CT图像，根据CT图像就可以提取CENTRIST特征了。级联检测器部分包括线性SVM和HIK SVM，我们分别介绍了其分类原理，以及分类器训练算法。

第四章 基于BING和C4的行人检测

受启发与于神经学，人眼在识别物体之前会对物体进行大概的感知，然后才会仔细去观察该物体是什么。这就是通过“粗略一看”和“定睛一看”结合而来的物体识别，在第二章介绍了BING建议对象生成算法，可以在整张图片中提取可能存在物体的区域，这样我们就把图片中很多无用的图像区域过滤掉，但是此时得到的结果还比较模糊，并不知道感兴趣区域是些什么，这时就需要使用基于CENTRIST特征的行人检测算法来做进一步检测，所以本文将BING和C4算法结合起来，通过BING算法对图像进行预处理，获得感兴趣区域，然后将得到的感兴趣框交给C4算法进一步判断。本文对C4算法进行了一定的改进，普通的C4算法会对整张原始图像进行多尺度的滑动窗口检测，多尺度和滑动窗口都会使得检测速度降低不少，会进行许多重复的计算。改进的C4算法则是利用BING得到的窗口表明该窗口已经包含一个有比较明显轮廓的物体的区域，现在就只需要判断这个物体是不是一个行人，所以我们不需要再无目标地进行滑动窗口检测了，而是直接将BING产生的感兴趣窗口交给C4算法判断是否有行人，如果是行人就在将其坐标保存下来，并在原始图像中标注出来。

4.1 BING和改进C4的行人检测流程

到目前为止，我们已经介绍了两个算法的相关原理，现在就需要将两者结合起来得到实时和准确的行人检测，据此我提出了两种方法：

方法一：将BING算法训练的似物性分类器作为一级分类器。并且采用滑动窗口，将大量区域通过BING分类器过滤掉，保留似物性高的目标框，然后再将目标框投入到下一阶段的C4行人检测算法中去进行详细识别。

方法二：不使用滑动窗口，把输入图片经过BING算法进行预处理，输出建议窗口，再通过C4算法识别窗口中的行人。

通过实验证明第二种方法效果更好，因为第一种方法还是进行了大量的重复计算，那么采用第一种方法后，整个行人检测的过程就如4.1所示



图4.1 行人检测流程

第一步：输入待检测图像

第二步：通过BING算法对输入图像进行似物性检测生成对象建议窗口

第三步：将第二步中得到的得分高的建议窗口输入到C4算法中，进行行人检测

第四步：对得到的检测结果进行后续处理，对统一区域的检测窗口进行抑制，将最终识别出来的行人框输出到图像中。

4.2 BING检测目标框流程

在第二章我们已经介绍了BING算法的原理，本节我们会介绍BING算法得到感兴趣区域的具体实现，包括分类器训练，特征计算，以及使用OpenMP加速等。

4.2.1 训练分类器

分类器的数据集我们使用的是VOC2007，首先我们需要在网上下载VOC2007，VOC2007主要的目录如图4-1所示：



图 4-1

文件夹Annotations中包含的是图片中目标所处区域的坐标，ImageSets文件夹则是保存的通过txt文件记录的样本集中哪些是训练样本，哪些是测试样本，其中有2501个训练样本，4952个测试样本。另外JPEGImages文件夹则是保存的所有使用到的JPG格式的图片。

我们使用的是C++作为编程语言实现算法，首先我们需要加载所有的训练数据和测试数据，并将训练数据中按照第二章的原理分为正负样本，保存在对应的数据结构中。接下来就会通过Liblinear库训练分类器。训练过程分为了两步。

(1)第一步，我们将已经封装好的正负样本数据输入到训练SVM的函数中得到一个8\*8的线性模型，我们将其转换为一个64维的向量并保存在磁盘中。

(2)第二步，­由于第二阶段的训练需要在第一阶段的基础上，所以我们首先需要加载模型一，然后用该模型对图形进行过滤打分，过程中会将原始图像缩放为28种尺寸，然后用8\*8的模板去扫描图像计算出每个框的得分，然后将结果通过NMS进行过滤，然后将结果分为正负样本集，按照对应的尺寸训练第二级分类器。

此过程需要注意的是由于不同尺寸的图片包含物体的可能性不同，所以我们需要针对不同的尺寸加入一个惩罚因子，也就是我们的第二级分类器的作用。训练完同样将模型保存在磁盘中。

4.2.2 动态选择遍历窗口

首先我们会将原始图片进行尺度变换，生成一系列缩放子图，BING算法在预测感兴趣框时用8\*8的检测框，在被缩放的子图上进行遍历。对子图所有点的BING值进行打分。在这里之所以使用8\*8的框是作者对BING在计算速度上的改进，因为目前64位整型数据是目前计算机支持的最大位数据类型。其他的有8位，16位，32位数据类型。这就导致检测框的大小最大就是8\*8。但是由于更小的检测框检测范围很小，过度抽象，检测效果并不准确。

4.2.3 OpenMP并行加速计算

由于算法中有大量的图像矩阵的计算，由于数据量巨大，所以普通的串行计算方式运行起来效率比较低下，无法满足我们的实时性需求，所以并行计算在此时就发挥了巨大作用，与普通串行计算相比，并行计算将计算任务分配到计算机的多个处理器协同处理，从而提高计算效率。当前人们采用的并行计算技术主要分为两类：一是基于CPU多核多线程的并行计算；二是基于GPU的通用并行计算。此处我们只介绍前者，根据并行粒度的不同可以分为“共享式内存结构”和“分布式内存结构”，其中OpenMP就是第一种的代表，已经被广泛地应用在数据处理和科学计算中。采用OpenMP最大的优点就是编程简单，源程序改变小的优点。

OpenMP是一种API，用于编写可移植的多线程程序，并且编写简单。OpenMP可以广泛地在Windows和Linux等多种平台上使用，在Visual Studio2012中使用只需要在项目属性中设置下支持OpenMP的选项即可。具体的编程实现也是相当的简单，只需要在源程序中的for循环中加入#pragma omp parallel for语句即可实现并行计算。如图4-2所示，OpenMP并行计算的思路是主线程将共享内存里的数据分配到不同的子线程里进行计算，每个子线程完成计算后再将结果返回到主线程，主线程再将结果分配到各个子线程进行下一步的计算。在使用OpenMP编码时，需要对不同子线程内的变量属性进行区分，避免不同线程里面的私有变量被其他线程的计算所影响。



图4-2 OpenMP计算架构示意图

使用OpenMP编程时有五个要求如下：

(1)for循环中的循环变量必须是有符号整型。

(2)for循环中比较操作符必须是<,<=,>=,>。

(3)for循环中的第三个表达式，必须是整数的加减，并且加减的值必须是一个循环不变量。

(4)如果for循环中的比较操作为<或<=，那么循环变量只能增加；反之亦然。

(5)循环必须是单入口、单出口，也就是说循环内部不允许能偶达到循环意外的跳转语句，exit除外。异常的处理也必须在循环体内处理。

4.3 C4算法检测过程

由于我们前面使用了BING算法对图像进行预处理，所以我们对C4算法进行了改进。经过了预处理之后，我们已经知道了图像中物体的位置，物体所处的目标框，接下来我们就需要对目标框进行检测，判断目标框中的物体是不是一个行人。通过对C4算法的性能评估，发现算法中最耗时，效率最低的就是多尺度检测和滑动窗口检测部分，所以BING算法会直接过滤掉一些无用的背景，使得C4算法滑动检测的区域减小。

C4算法的详细检测过程主要分为以下几个步骤：

（1）程序读取已经训练好的线性分类模型和HIK SVM模型以及待检测的图像，把它们保存在内存中；

（2）将步骤（1）中读取的图像按照一定比例缩放，构建图像金字塔；

（3）对缩放的图像使用Sobel边缘检测算子进行处理，消除对行人检测不理的纹理性特征，比如颜色等，从而得到一副Sobel图像；

（4）对Sobel图像的每个像素计算CT值，构建一副CT图像；

（5）构建图像的积分图像，其中我们要按照第三章介绍的简化计算需要构建一个辅助图像A来作为我们使用线性SVM和CT值进行矩阵运算的结果，即为第一级线性分类器的检测结果。然后在辅助图像A中计算积分图像；

（6）使用第二级分类器HIK SVM进行第一轮分类器得分大于0的窗口进行检测；

（7）使用非极大值抑制对结果进行处理，从而得到最终的目标检测结果；

（8）把检测到的行人标注在原始图像上。

以上就是C4算法的主要步骤，整个流程图如图4-3所示。

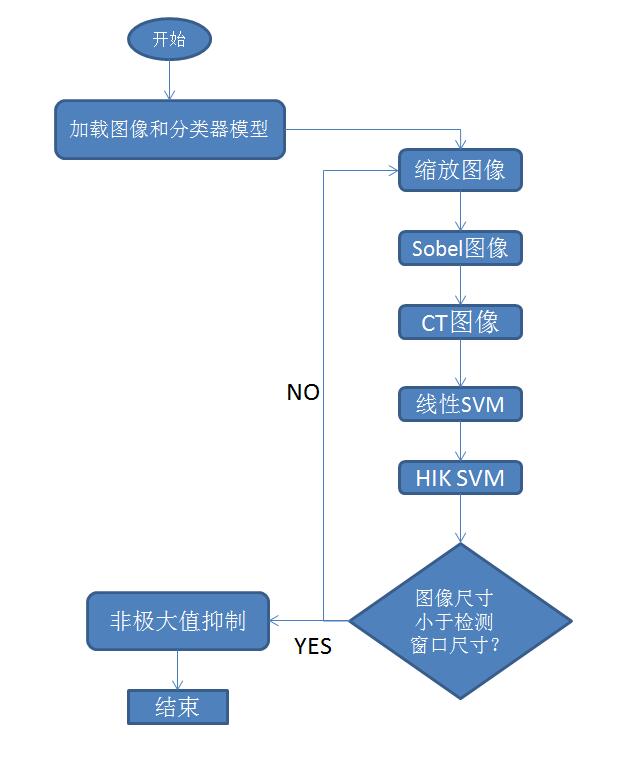
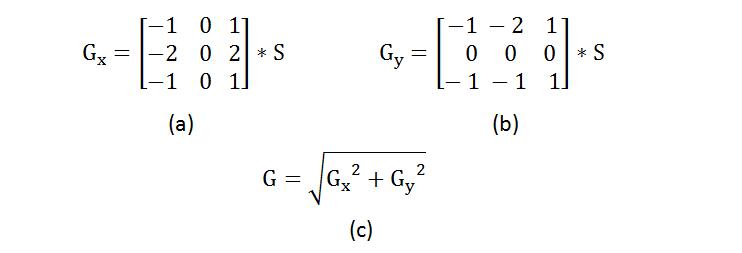


图4-3 C4算法检测流程图

4.3.1 计算Sobel图像

Sobel算子是一个离散微分算子。它结合了高斯平滑和微分求导，用来计算图像灰度函数的近似梯度。对于图像的边缘，像素值会发生显著的变化，表示这一改变的方法就是使用一阶导数，具体的计算是使用卷积的方式实现，假设被处理的图像为S，在两个方向对图像进行求导：（1）水平方向求导，将S与一个奇数大小的内核进行卷积。比如内核大小为3时，的计算结果为图4-4（a）；（2）垂直方向求导：将S与一个奇数大小的内核进行卷积。比如，内核大小为3时，的计算结果为图4-4（b）。然后对于图像的每一个像素点，结合以上两个结果求出近似梯度，如图4-4（c）。

 图 4-4 Sobel计算公式

值得注意的是在计算近似梯度的时候需要开平方计算，而且此操作比较费时。本文对此有小小的改变，由于Sobel图像是为计算每个像素的CT值做准备的。计算CT值只需要比较Sobel图像相邻像素点灰度值的大小，所以此处就不需要进行开方操作，因此计算Sobel图像时直接按照公式（4-1）计算即可。

(4-1)

4.3.2 计算CT图像

计算出原始图像的Sobel图像后，接下来需要计算出Sobel图像中每个像素对应的CT值，用CT值来代替Sobel图像对应位置的灰度值就得到了CT图像，计算时需要注意边缘像素。

由第三章介绍的CT值计算方法可以直到，Sobel图像中某个像素的值对应的CT值是根据该像素8邻域的灰度值的比较计算得到的，而Sobel图像上与边界像素相邻的像素点却不足八个，因此我们选择不对边界像素进行计算。

计算CT值过程会得到一个八位的二进制数，然后再将这个数转换为十进制值才得到了CT值。比如式4-2。

(00100110)=1x+1x+1x= (4-2)

通过式子（4-2）可以看到一个CT值的计算仅仅需要八次比较和求和操作。图像计算效果如图4-5所示。



图4-5 CT图像结果

4.3.3 辅助图像计算

根据之前的介绍可以直到，采用线性SVM检测时只需要构造一幅辅助图像即可，不用显式提取CENTRIST特征就能快速完成对行人检测。这种检测方法需要构建辅助图像，然后再根据辅助图像计算积分图像，进而完成线性SVM对行人的检测。

辅助图像的计算公式如下式（4-3）：

(4-3)

公式中A就代表的要计算的辅助图像，表示之前我们训练好的线性SVM模型，由24个，组成，代表模型第k个维度的数值，C表示CT图像，和分别代表检测窗口中单元格的高度和宽度。

用CT图像中的像素值作为索引，查找线性SVM中的值并累加起来就求得了辅助图像的对应位置的灰度值。训练的线性SVM是一个6144位的向量，向量的长度对应于CENTRIST特征的维度。

4.3.4 积分图像的计算

由于是使用滑动窗口的方式来检测行人，所以原本我们需要对每个目标窗口计算Sobel，计算CT，最后得到CENTRIST特征，但是这样就会导致整个检测过程会进行大量重复的计算，从而影响检测效率，所以积分图像就派上了用场，积分图像在视觉领域应用非常广泛。

积分图像的定义是取图像左上测的全部像素计算累加和，并用这个累加和替换原图像中的每一个像素值，通过这种方式得到的图像就是积分图像。如图4-6所示，图中有四个点，

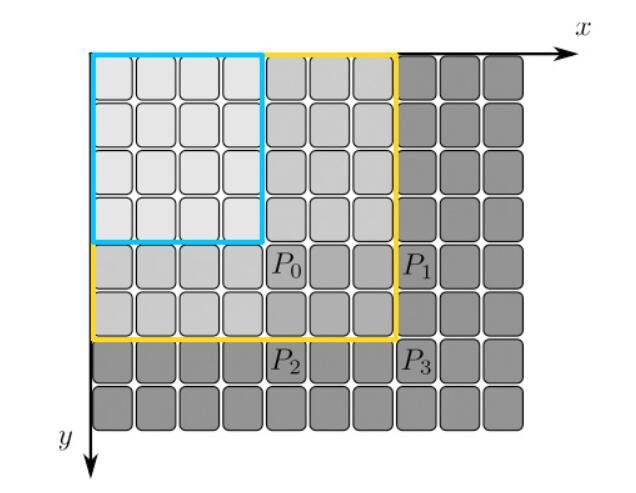


图4-6 积分图像计算示意图

其中点的左上侧就是蓝色的矩形框所包含区域，所以点的得分就是蓝色框中所有像素点的值的累加和，的左上侧就是黄色框所包含的区域。由此可知计算整幅图像的积分图像就只需要对原始图像扫描一次。因为对于同一行（列）的相邻两像素，当前像素的积分值等于上一像素的积分值加上当前像素的值加上前一像素积分值，由此可知积分图像是一个包含像素累加和的新图像。

得到积分图像后，当我们需要求某一个检测窗口的线性分类器的分类得分时，若得分大于或者等于0，则线性分类器判定检测框中包含行人，反之不包含行人。但是得到积分图像了就需要计算某一检测框内的像素值的累加和，假设一幅图中有ABCD四个点，其积分图像中A(*x*2,*y*1)点的值为其左上侧的所有像素的值的累加和，也就是蓝色区域中所有像素点的值累加，同理积分图像中的B(*x*2,*y*1)、C(*x*1,*y*2)、D(*x*2,*y*2)点值分别是绿色、紫色和黄色区域像素值的累加和。ABCD四点的位置关系如图4-7所示。

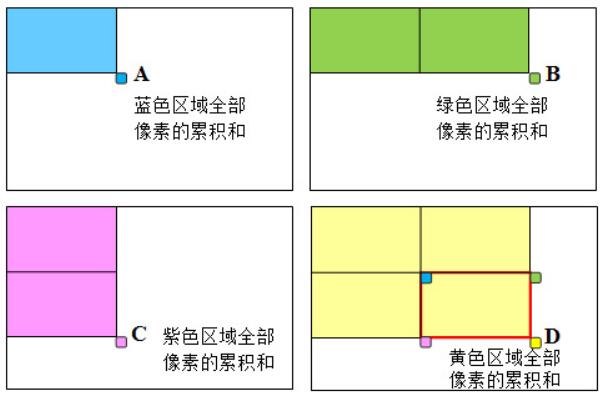


图4-7 利用积分图像计算检测框像素值总和

计算由ABCD组成的感兴趣区域的累加值就只需要按照如下公式（4-4）即可：

S=sum(*x*2, *y*2)- sum(*x*1, *y*2)- sum(*x*2, *y*1)+ sum(*x*1, *y*1) (4-4)

公式表达的即是：D-C-B+A。显然，计算量不受区域尺寸影响。所以，如果需要在多个尺寸的区域上计算像素累加和，最好采用积分图像。由此可以发现，我们只需要初始化图像时计算一次图像的得分，以后滑动窗口检测就只需要查积分图像，通过简单的加减法就可以得到结果。