PaperTest检测报告简明打印版

比对结果（相似度）：
总体相似度：16 % *（相似字数占总字数的百分比）*
红色相似度：11 % *（句子相似度70%-100%的字数占总字数的百分比）*
橙色相似度：5 % *（句子相似度40%-70%的字数占总字数的百分比）*

编　号：5ACF76DBBA2CB09QD
标　题：基于BING和C4的快速行人检测
作　者：艾伦
长　度：30006 字符(不计空格)
段　落：853个
句　子：380句
词　语：12395个
时　间：2018-4-12 23:10:19
查真伪：http://www.papertest.com/cha

全文简明报告：

机器视觉领域，无论是对图片分类，还是目标识别，传统的方法都是采用训练好的分类器对输入图像进行滑动窗口检测。 滑动过程中会产生大量窗口，并需要用分类器对窗口进行检测，从而达到检测识别的目的。 {46%：因此，提升检测算法速度的关键就是减少检测窗口的数量。} 针对这一问题，本文通过对似物性方法和传统行人检测方法的研究，提出了一种基于BING和C4的快速行人检测算法。

{73%：针对传统方法耗时的问题，有人提出在检测前加入预处理过程，通过产生的预测框来代替遍历图像的方法，开创了一个新的研究方向。} {61%：2014年由程明明等人在CVPR上发表了似物性采样算法BING，此方法代表了当前图像预处理领域的最高水平。} {73%：BING算法使用8*8大小的检测框，因为计算机中最大数据类型是64位的，刚好BING值经过二值化处理后可以保存在int64中。} {94%：针对图像中大小不同的行人，在8*8的检测窗口基础上增加不同尺寸的检测窗口，防止漏检。}

{43%：C4算法是吴建鑫等提出的一种快速、准确的行人检测算法。} {42%：算法采用的是基于行人轮廓信息的CENTRIST特征，再结合级联分类器对输入图像进行检测。} {45%：C4算法相对于HOG+SVM，在检测速度和检测精度上都更有优势。} 首先需要通过缩放构建图像金字塔，接着用Sobel算子构建Sobel图像，然后计算每个像素的CT值得到CT图像。 检测阶段，使用线性SVM检测行人，通过构建辅助图像和积分图像就能方便地得到线性SVM的分类结果。 {49%：由于线性SVM分类的准确率较低，所以需要使用HIK SVM进一步检测。}

基于 BING和 C4的完整行人检测方法把 BING方法作为行人检测的预处理阶段，将产生的对象建议窗口交给准确的行人检测方法 C4进行精确检测， 我们在 INRIA数据库和自己拍摄的图像库中进行了实验，证明算法在检测速度和精度上都满足要求。

关键词： 行人检测； BING； C4； SVM

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。

{71％：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:

1.1.1 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。

1.1.2 智能安防

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； {90％：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。

1.1.3 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， {100％：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。

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。 {97%：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

从上世纪90年代以来，交通事故已经成为重要的死亡因素，2003年，美国交通部分发表的报告显示只是在欧洲的交通事故中，就大概有150，000以上的人受伤和7，000人死亡。 为了减少交通事故的发生，ADAS被提出，它们作为辅助系统帮助驾驶员做出正确的决定或提醒驾驶员忽视的一些危险情况。 随着技术的发展，许多学者都加入到行人检测的研究中。

{48%：当前主流的行人检测算法主要有三种：} 1.基于模板特征匹配法； 2.基于统计学习和神经网络；3.基于人体显着部件模板。 这三种方法各有不同，相互之间各取所长，因此我们在实际应用中可以相互借鉴，相互补充。 {49%：其中基于统计学习的行人检测算法是根据大量的行人样本对其提取特征训练行人检测分类器。} {80%：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。} {70%：分类器主要包括神经网络、SVM、AdaBoost、级联分类器 以及当前十分火热的深度学习 。} 统计学习方法目前的难点： 图像中行人特征，行人尺度复杂，各种特征提取方法得到的特征有限，分类性能受限于训练样本，模型的泛化能力差。

行人检测的的主要难点有以下5个方面：

（1）道路上的行人具有非刚性的特性，不是静止不变的物体。 道路上的行人着装、姿态、行为的变化很大，所以我们很难找到一个固定的模板比对后直接判断。 {100%：所以就要求我们选择行人的特征时，要选择区分度高的作为训练样本。}

{93%：（2）由于摄像头的位置始终相对行人的拍摄角度是变化的。} 所以即使行人的姿势是没有变化的，成像后表现的特征也会因为视角呈现很大的差异。

（3）道路两旁的环境复杂，所以经常会出现人被其他物体遮挡，那么行人检测算法就只能从图像中部分人体部位进行判断。 这些部分行人信息就很难在训练的分类器得到很好的分类效果，使得检测精度降低。

（4）拍摄到的公路上的背景经常会出现与行人很相似的特征，导致一定误检。 比如路边的树，电杆，垃圾桶。 {40%：另外光照条件也是一个很重要的因素。}

（5）行人检测算法的另外一个主要问题就是实时性。

{69%：2005年法国科研人员 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，} {97%： 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，INRIA数据库，Caltech行人数据库 等。 虽然有这些公开的数据库可以供我们使用，但其覆盖的场景有限，所以

我们必须针对自己的应用场景添加自己的样本数据。 比如我们要做ADAS的行人检测，那么针对国内特殊的路面情况，有些电瓶车，三轮车或者渣车等都是国外数据库不包含的。 目前国内有家叫 MINIEYE的辅助驾驶系统生产商就在建立自己的国内的路况图像数据库，当然建立这个数据库的成本很大， 所以由于成本限制，本文只是在INRIA数据库的基础上添加了少量训练样本。 {70％：INRIA数据库主要是针对静态行人检测的数据库，包含了原始图片和相应的标注文件。}

HOG特征描述了一幅图像的方向梯度信息，将窗口划分为大小相同的图像块（block）， {44％：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，而且是滑动窗口检测，有很多重复计算，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， {46％：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 似物性检测研究现状

物体识别的流程如图1-2，通常来说我们要先对图像进行预处理进行多尺度缩放，再用滑动窗口扫描，通过分类器计算窗口得分情况。 本文使用物体感知来对图像做预处理，先找到可能包含物体的区域，再使用分类器对其进行分类，减少检测区域。

图1-2 物体识别流程图

人类大脑拥有超强的学习能力，所以对周围的事物有很强的辨识能力，这种能力不是对所有物体都一样的，人们可以迅速的从环境中找到自己感兴趣的区域，过滤掉无用的区域。 如果这种能力借鉴到机器视觉中就是在识别具体某个事物之前先预先感知某个对象。 其中似物性检测 就是实现这样的功能。 似物性采样方法致力于找出包含图像中所有对象的小量建议窗口，为后面的精确检测提供建议区域。 {50％：通过产生可能对象的建议窗口，似物性采样已经被证明是一个有效的减少分类器搜索区域的方法。} Cheng等提出的BING算法是一种简单有效的方法，这种方法很适合在行人检测中使用，300fps的对象查找速度使得行人检测的实时性有了无限可能。

1.4 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1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2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3章介绍了C4行人检测算法，并与HOG+SVM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4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5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2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再通过准确的行人检测模型进行评估， 实现检测行人的功能。

2.1 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，并且优先理解显着性物体。 这种基于视觉显着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着区域，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， 然后提取前 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。

2.1.1 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。 {43％： 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， } {89％： 高分辨率的对象窗口，通过产生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%。

2.2 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。

图2-1 BING算法演示图

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）， {43%：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2-1所示。} {71%：图（a）中红色框内图像为完整物体区域，绿色框内为背景或者包含部分物体区域，（b）原图像不同尺寸的NG图，} {45%：（c）中红色框为物体的8*8的NG特征图。} {57%：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。}

{68%：BING特征受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维的梯度幅值特征，} {87%：二值化的梯度幅值特征（BING）可以很有效的描述图像窗口的对象状态。} {71%：为了使模型有较好的泛化能力，我们使用一个定义好的量化窗口（依据尺度或者是纵横比）作为检测窗口。} {48%：每一个窗口的BING特征输入到一个线性模型ω∈R^64进行打分：}

$$s_l=[\omega, g_l] \quad (2\text{-}1)$$

$$l=(i, x, y) \quad (2\text{-}2)$$

{80%：s_l代表过滤器得分，g_l代表NG特征，l代表坐标，i表示尺度，（x，y）表示窗口位置。} {64%：使用非极大值抑制（NMS）进行后处理，我们为每个尺度生成一些建议窗口。} 相对于其他窗口（例如： {89%：200*200），一些尺度（10*500）的窗口不太可能包含对象。} {74%：所以我们定义对象状态最终得分（校准过滤器得分）：}

$$o_l=v_i*s_l+t_i \quad (2\text{-}3)$$

{85%：其中v_i，$s_l\in R$，针对不同窗口尺度i的窗口，再一次使用线性SVM训练不同的独立学习参数。} 对一级分类器检测结果进行校准是非常快的，把最终的建议窗口进行重排序后就可以进行。

2.2 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〖1，0，-1〗^T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2-2所示。

图2-2 梯度特征图

2.3 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。 以这些知道准确位置的目标框为基础，使用下面的计算公式对图像进行采样，就可以获得正样本图像。

(S_派∩S_标准)/(S_派∪S_标准)]=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， 随机采样获得的图像使用下面的公式进行判断，满足条件的图像框就留下，不满足的就舍弃。

(S_派∩S_标准)/(S_派∪S_标准)[=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2-3中的步骤对训练数据集进一步处理得到可用的训练数据。

{58％： 图2-3 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， {45％：它具有计算复杂度低，训练时间少的优点。} 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， {42％：仅仅是根据我们生成的正负样本数据，训练出一个 SVM分类器模型 w，这个模型是一个64维向量。}

第二级SVM则是为了对不同尺寸窗口包含对象的概率进行校准而训练出的参数。 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2-4所示。

{100％： 图2-4 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。 所以为了减少子图尺寸的影响，我们在不同子图中的预测框分数加入一个惩罚项，那么我们需要训练每个尺寸对应的参数v_i和t_i。 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。

2.3 二值化

我们知道计算机进行位运算速度很快，所以我们应该尽量使用位操作来完成计算，首先我们将训练的目标模型进行二值化， 用0或1的序列来表示浮点数，同样提取的图像的标准梯度特征也用二进制值表示， 这样就使得整个计算过程加速不少。

目标模型w是我们使用LIBLINEAR训练出来的一个由64个浮点数组成的向量。 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: 首先将目标模型 w用一组 N_w个的基来表示，计算量的大小与 N_w的值直接相关， 因此我们取 N_w=2，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：

$\omega \approx \sum_{i=1}^{2} [\beta_j \alpha_j]$ (2-4)

(2.4) 式将w模型二值化近似，其中$\alpha_j \in \{-1, 1\}^{64}$，$\beta_j \in R$。 {87%：现将64维的浮点数根据式子(2.5)和(2.6)转化为-1或1。}

$\omega_j \geq 0$， 1 (2-5)

$\omega_j \leq 0$， -1 (2-6)

基底α_j可以转化为如下所示：

$[\alpha]_j = [\alpha_j]^+ - ([\alpha_j]^+)$ (2-7)

$[\alpha_j]^+ \in \{0, 1\}^{64}$

$([\alpha_j]^+) = I - [\alpha_j]^+ \quad I \in \{1\}^{64}$ (2-8)

因此将式子（2.8）带入到（2.7）中就可以得到：

$\omega \approx \sum_{i=1}^{2} [\beta_j (2[\alpha_j]^+ - I)]$ (2-9)

此时浮点型的64维模型就用0或1的序列来表示了，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，{42%：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：

$w, b \approx \sum_{i=1}^{2} [\beta_j (2[\alpha_j]^+, b - |b|)]$ (2-10)

梯度值的二值化： 前面介绍的梯度计算方法求出来的梯度值是一个0~255范围的一个整数，刚好可以用一个字节来存储，八位的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2-5所示：

图2-5 NG特征

{43%：可见NG特征二值化后按位平面分层，位置l处的BING值就可以近似如下式所示：}

$g_l = \sum_{k=1}^{N_g} [2^{8-k} b_{(k, l)}]$ (2-11)

式子中乘了一个值是因为每个位平面的权值不同。 $b_{(k, l)}$表示位置l处的BING值。 {42%：通过二值化处理后，目标模型和图像的NG特征都可以用一个int64的数据来表示。} 然而我们发现检测框是一个8*8的矩形框，所以需要整个框的BING值，遍历显然是很好耗时的。 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，可以通过移位操作的方式，求出每个位置各个位平面上的8*8的框，而无需遍历。

{42%：图2-5中的红色方框表示点L处第k个位平面的BING值，记作$b_{(k,}$ } {91%：l)，绿色框表示框的

最后一行，记作r_(k，l)。｝｛66％：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2-6 L点的BING值

（a） （b）

｛41％：图2-7 相邻点值的相关性｝

｛41％：由于相邻的BING值之间有一定的联系，例如图2-7（a）所示，｝｛45％：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，｝｛87％：其余7排是完全一样的，所以当知道点（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r_(x，y)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

r_(x，y)。 ｛74％：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2-8：

图2-8 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B_(x， ｛45％：y)都可以用64位整型数据来表示，每个点的评分就是w与B_(x，y)求内积。｝

2.4 过滤候选框

第一级分类器模型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可能性。 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 Suppression）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。 ｛54％：局部区域是指一定大小的邻域，邻域有维数和大小两个可变的参数。｝｛41％：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。 ｛47％：通过滑动窗口的方法提取的图像的BING特征经过两级分类器后会得到一个分数，用于判断窗口中是否包含对象。｝但是滑动窗口的检测方法明显最终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。

｛85％：（2）每个框对应一个标记信息。｝

｛93％：（3）从得分最高的框开始，检查在其邻域内的检测框，如果标记为1，则重新标注为0；｝ 同样的方法处理剩余的窗口。

｛91％：（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。 {50%：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2-9是BING的检测效果。

图2-9 BING检测效果

2.5 BING算法应用于行人检测

{47%：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。} {73%：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2-10所示。}

（a） （b）

图2-10 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2-11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2-11 BING+HOG检测效果

2.6 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， {75%：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。

{73%：第3章 基于CENTRIST特征的行人检测算法原理}

3.1 引言

{54%：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。

3.2 CENTRIST特征简介

3.2.1 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。 {46%：在不使用任何硬件加速的情况（单线程，不用GPU）下，C4算法在保证高精确度的情况下可以有20帧每秒的速度。} {100%：能达到实时而精确的检测源于以下两点：} {74%：第一，相邻像素大小关系是描述轮廓的关键信息；} {49%：第二，CENTRIST描述子通过编码的形式表达了相邻像素的关系即轮廓信息，所以很适合做行人检测。} {74%：针对 CENTRIST特征和线性SVM的组合，我们提出了一种不需要显式生成特征向量的计算方法，} {94%：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边缘检测算子消除这些纹理特征，这样也不会丢失关键的轮廓信息。 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3-1：

图3-1 Sobel平滑图像

如图3-2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。

{49%：图3-2原图经Sobel算子处理后的结果}

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3-1所示。 {42%：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。} {90%：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，} {99%：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。} {97%：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3-3

对原图中的每个像素计算对应的CT值，这样我们就得到一个同样大小的图像，我们称为CT图像。 {46%：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。} {62%：最终得到的这个直方图就称之为CENTRIST特征。}

{69%：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，} {69%：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），} {60%：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。} {69%：（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）。}

3.2.2 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：

读取原始图片到内存中；

{49％：采用滑动固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，为了不产生漏检，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， {100％：如图3-4所示，红色的矩形框就是检测窗口。}

{55％：使用Sobel算子对图像进行处理，构建Sobel图像；}

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；

{47％：将目标窗口中的CT值转换为一个6144维的CENTRIST特征；}

整个流程如图3-5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3-4 缩放后的检测结果

图 3-5 CENTRIST特征计算效果图

3.3 级联分类器

{43％：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。} {59％：首先要对分类器进行训练。}

3.3.1 SVM原理

{76％：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，简单来说就是一个分类器，} 是一种通用的前馈网络模型，深度学习出来之前它是表现最好的一种学习算法。 {100％：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，} {90％：即支持向量机的学习策略便是将间隔最大化问题最终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。 {58％：如图3-6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，} {94％：分别用两种不同的颜色表示，红色的表示正样本，蓝色表示负样本，红色的直线是一个超平面，} {50％：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 3-6 线性分类图

接着我们可以假设超平面为：

$f(x)=w^T x+b$ (3-1)

其中w=（w_1; w_2; {50%：w_d）是法向量，b为偏置项，（w，b）决定了超平面方程，样本到超平面的平均距离为：}

r=|w^T x+b|/‖w‖ (3-2)

{46%：如果（w，b）表示的超平面可以对样本集进行分类，即(x_i，y_i)∈D，如果y_i=1则w^T x+b]0，}

如果y_i=-1，则w^T x+b[0。

{■(w^T x+b≥1 ，y_i=1@w^T x+b≤-1 ，y_i=-1)┤ (3-3)

{43%：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:}

γ=2/‖w‖ (3-4)

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

〖〖max〗┬(w，b)=〗〖2/‖w‖〗

s. t. y_i(w^T x_i+b)≥1 i=1，2，…，m (3-5)

{40%：因此当‖w‖^(-1)最大，就可以使得分类间隔最大。} 等价于让‖w‖取最小值，因此公式（3。5）可以改写为：

〖〖min〗┬(w，b)=〗〖1/2‖w‖^2〗

s. t. y_i(w^T x_i+b)≥1 i=1，2，…，m (3-6)

{43%：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，} {49%：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：}

L(w，b，a)=1/2‖w‖^2-∑_(i=1)^n〖a_i y_i(w*x_i+ b)+∑_(i=1)^n a_i〗 (3-7)

其中a=〖（a_1，a_2，… {75%：，a_N）〗^T为拉格朗日乘子向量。} {96%：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：}

〖max〗┬a〖min〗┬(w，b)〖L（w，b，a）〗 (3-8)

{64%：所以，先求出L（w，b，a）对w，b的极小值就可以得到对偶问题的解，再求对a的极大值。} 首先求〖min〗┬(w，b)〖L（w，b，a）〗，我们对拉格朗日函数分别对w， {95%：b求偏导并令其等于0，因此求解等式就得到了：}

w=∑_(i=1)^n〖a_iy_ix_i〗

∑_(i=1)^n〖a_iy_i〗=0 (3-9)

{45%：再将以上两式带入拉格朗日函数之中就得到:}

〖min〗┬(w, b)〖L（w, b, a）〗=-1/2∑_(i=1)^n∑_(i=1)^n〖a_ia_jy_iy_ j(x_ix_j)+∑_(i=1)^n a_i〗 (3-10)

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

〖min〗┬a L(w, b, a)=1/2∑_(i=1) ^n∑_(i=1)^n〖a_ia_jy_iy_ j(x_ix_j)-∑_(i=1)^n a_i〗

s. t. ∑_(i=1)^n〖a_iy_i〗=0 a_i≥0, i=1, 2, … , N (3-11)

{43%：而求解此问题就需要用到SMO算法了，} {88%：SMO算法是支持向量机学习的一种快速优化算法，} {100%：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，} {84%：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，} {67%：由于子问题有解析解，所以每次子问题的计算都很快，即使子问题计算次数很多，但整个算法效率还是很高的。} 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。 {86%：假设已经求得了对偶最优化问题的解a=〖（a_1, a_2, …} , a_l）〗^T，就可以根据式3.5和3.6求到（w, b）的解。 因此就可以将超平面写成：

∑_(i=0)^n〖a_iy_i(x*x_i)+b〗=0 (3-12)

分类决策函数就可以写成：

f（x）=sign（∑_(i=0)^n〖a_iy_i(x*x_i)+b〗） (3-13)

{76%：由此方程式我们发现分类决策函数只依赖于输入x和训练样本输入的内积，此式子被称为线性可分支持向量机的对偶形式。} {63%：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；} {69%：由此我们就求出了分离超平面及分类决策函数。} {97%：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，{42%：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： K(x_i, x_j)=〖x_i〗^T x_j

（2）多项式核函数： K(x_i, x_j)=[（〖x_i〗^T x_j）+1]^q

（3）高斯径向核函数： K(x_i，x_j)=exp（-γ‖x_i-x_j‖^2）

（4）Sigmoid核函数： K(x_i，x_j)=tanh（γ*〖x_i〗^T x_j+τ）

其实核函数不只以上介绍的这几种，只要该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。

3.3.2 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。 {47%：SVM是国外的Vapnik提出的一种基于结构风险最小化理论的统计学习方法。} 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。

{42%：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：

∑_(i=1)^n〖c_i k_HI〗（q，x_i）

k_HI (x，y)=∑_(j=1)^dmin(x_j，y_j） （3-14）

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k_HI (x，y)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。

{42%：3.3.3 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 R}，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， {44%： d表示特征的维度， x_(i， j)作为特征x_i的第j维向量。} 首先当d=1的时候，有n个样本点x_1，x_2，…，

x_i R，并且假设x_i≤x_(i ')其中1≤i[i^ ' ≤n，所以核函数有以下性质：

〖K〗_(ii ')=min(x_i，x_(i '))=x_(min（i，i ' ）) （3-16）

由此可以得到A=R^T KR，其中R和A的定义如下：

R_ij={■(1，如果i=j@-1，如果i=j-1@0，其他情况)┤ A_ij={■(x_1，如果i=j=1@x_i-x_(i-1)，如果i=j]1@0，其他情况)┤

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。当d=1时，k_HI为正定核。由于满足Mercer定理 的核通过迭代相加后亦然满足该定理，所以当d]1时可以得到同样的结论。

3.3.4 HIK的快速计算

{42％： 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{x_i，y_i}，} 其中y_i∈{-1，1}，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3.3.1已经介绍了SVM的原理，我们得到最终求解的判别式是：

h（x）=∑_(i=1)^m〖a_iy_ik(x，x_i)+b〗 （3-17）

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 （3-18）

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（x_1），B（x_2），…，B（x_d） （3-19）

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k_HI=〖B(x)〗^TB（y）。 {42％：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）=∑_(i=1)^n〖a_iy_ik_HI(q，x_i)-θ〗 （3-20）

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。 {43％：为了达到快速检测的效果，我们引入了一个T表，我们令c_i=a_iy_i，T_(l，k)=∑_(i:} k≥x_(i，k))〖c_ix_(i，l)〗+k∑_(i: k[x_(i，l))c_i，最终得到表达式：

f（q）=∑_(l=1)^dT_(，ql)-θ （3-21）

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）=w〗^T B(q)-θ=∑_(l=1)^d〖〖w_l〗^T B(q_l)-θ〗 (3.22)

有1，2二式，可以得到： 〖w_l〗^T B(q_l)=T_(l，ql)，由此计算时就可以通过T表直接得到〖w_l〗^T B(q_l)的值， 所以此方法显着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。 {99%： 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）减少到了O（d）。

3.3.5 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：

〖min〗┬w〖1/2 w^T〗 w+C∑_(i=1)^n〖ε（w， B(x_i)，y_i）〗 (3-23)

{70%：其中ε（w，B(x_i)，y_i）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。}

{46%：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：}

〖min〗┬a〖g(a)=1/2 a^T〗 Q’a-e^T a

s. t. 0≤a_i≤U (3-24)

其中Q’=Q+D， Q_(ii‘)=y_i y_(i’)〖B（x_i）〗^T B（x_(i‘)）， D是对角矩阵， D_ii=1/2C。 {40%：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3-7。

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3-7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（x_i）。 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（x_i）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3-8表示了具体的替代部分计算方法。

图3-7 普通算法原理

图 3-8 ICD算法原理

3.3.6 训练级联分类器

（1）线性SVM的训练

{45％： 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，｝｛61％： 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 N_1，｝｛57％： 再次使用 P和 N_1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。｝｛71％： 然后使用P和所有的负样本集N_i训练最终的支持向量机分类器H_lin。｝

（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。 ｛44％： 在训练过程中，采用前面线性分类器 H_ lin对样本集 N进行分类得到的负样本组成一个新的负样本集 N_ final，｝ 然后用正样本集 P和 N_ final训练得到 HIK SVM。

{54％： 3.3.7 使用级联分类器做行人检测｝

{42％： 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。｝｛92％： 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， ｛100％： 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。

{48％： 3.3.8 如何使用线性SVM进行快速行人检测｝

{57％： 假设有一个我们已经训练好的线性分类器ω∈R^6144，由于 C4算法的特征可以分为8*3个超级块，｝所以可以把ω划分成24个ω_(i，j)∈R^256，1≤i≤8， 1≤j≤3组成。 ｛44％： 对应地C4特征也可以被划分为24个小单元f_(i，j)。｝ 传统算法是将分类器模型ω^T与特征向量f进行内积运算，如下所示：

$$\omega^T f=\sum_{i=1}^{8}\sum_{j=1}^{3}\left[\left[\omega_{(i,j)}\right]^T f_{(i,j)}\right] \qquad (3\text{-}25)$$

{42％： 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。｝ ｛70％： 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。｝｛43％：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（h_s， w_s）=（h/9， ｛95％： w/4），把四个单元格合并为一个图像块。｝｛44％：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3-1的就可以变换为：｝

ω^T f=∑_(i=1)^8∑_(j=1)^ 3∑_(x=2)^(2 h_s-1)∑_(y=2) ^(〖2 h〗_(s-1))〖ω_(i， j)〗^(C（t+(i-1) h_s+x， l+(j-1) w_s+y））（3-26）

此时ω^k表示其低k个维度的值，而且k的取值范围为0，255。 ｛98％：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：

〖A_(i， j)〗^(x， y)=〖ω_(i， j)〗^(C（x， y）） （3-27）

因此公式（3-2）就变为了：

ω^T f=∑_(i=1)^8∑_(j=1)^ 3∑_(x=2)^(2 h_s-1)∑_(y=2) ^(〖2 h〗_(s-1))〖A_(i， j)〗^(t+(i-1) h_s+x， l+(j-1) w_s+y)（3-28）

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：

A(x， y)=∑_(i=1)^8∑_(j=1)^3〖ω_(i， j)〗^(C((i-1) h_s+x，(j-1) w_s+y））（3-29）

最终就得到了如下公式：

ω^T f=∑_(x=2)^(2 h_s-1)∑_(y=2)^(〖2 h〗_(s-1))〖A（t+x， l+y）〗 （3-30）

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。

3.3.9 基于CENTRIST和HOG特征的行人检测对比

HOG特征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， ｛74％：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3-9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。 ｛42％：在文献2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%，｝｛46％：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3-9 HOG和C4检测结果

我们用英特尔i5处理器的计算机用这两种算法检测不同尺寸的图像，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。

表3-1 HOG和C4算法计算时间

图像大小HOG算法处理速度（fps）C4算法处理速度（fps）

320*24020.531.6

640*4803.57.5

1280*9601.01.6

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。

3.4 本章小结

{49％：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， {47％：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4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， {91％：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，} {41％：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4-2所示：

图4-2 VOC2007目录图

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， {100％：哪些是测试样本，其中有2501个训练样本，4952个测试样本。} {94％：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。 {40％：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。

{82%：（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第一个模型，} 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。 {40%：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。 {46%：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。 {100%：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，所以选择8*8大小是比较合适的。

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。 {96%：采用OpenMP最大的优点就是编程简单，源程序改变小的优点。}

{75%：OpenMP是一种API，用于编写可移植的多线程程序，并且编写简单。} {43%：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。} {52%：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4-3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， {74%：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4-3 OpenMP计算架构示意图

使用OpenMP编程时有五个要求如下：

{90%：（1）for循环中的循环变量必须是有符号整型。}

{91%：（2）for循环中比较操作符必须是[，[=，]=，]。}

{95%：（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。}

{83%：（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。

{90%：（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。} {100%：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；

{50%：（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作为我们使用线性 SVM和 CT值进行矩阵运算的结果， {56%：即为第一级线性分类器的检测结果。} {54%：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；

{41%：（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。

{40%：以上就是C4算法的主要步骤，整个流程图如图4-4所示。}

图4-4 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。 {100%：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。} {59%：对于图像的边缘，像素值会发生显着的变化，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，} {69%：假设被处理的图像为 S，在两个方向对图像进行求导：} {41%：（1）水平方向求导，将S与一个奇数大小的内核G_x进行卷积。} {71%：比如内核大小为3时，G_x的计算结果为图4-

5（a）；} （2）垂直方向求导：{72%：将S与一个奇数大小的内核G_y进行卷积。}{85%：比如，内核大小为3时，G_y的计算结果为图4-5（b）。}{72%：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4-5（c）。}

图 4-5 Sobel值计算公式

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。{94%：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。

S(x，y)=〖G_x〗^2+〖G_y〗^2 (4-1)

4.3.2 计算CT图像

{41%：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，}{49%：用CT值来代替 Sobel图像对应位置的灰度值就得到了 CT图像，}{46%：计算时需要注意边缘像素。}

{70%：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）=1x2^1+1x2^2+1x2^5=〖（38）〗_10 (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4-6所示。

图4-6 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）：

A(x，y)=Σ_(i=1)^8 Σ_(j=1)^3〖ω_(i，j)〗^(C((i-1)h_s+x，(j-1)w_s+y)) (4-3)

公式中 A就代表的要计算的辅助图像，ω表示之前我们训练好的线性 SVM模型，ω由24个ω_(i，j) R^256，1≤i≤8，1≤j≤3组成，〖ω_(i，j)〗^k代表模型第 k个维度的数值，C表示 CT图像，h_s和 w_s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。 {

41％： 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， {44％： 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。 }

{85％： 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。 } {100％： 如图4-7所示，图中有四个点， }

{58％： 图4-7 积分图像计算原理图 }

{40％： 其中P_0点的左上侧就是蓝色的矩形框所包含区域，所以P_0点的得分就是蓝色框中所有像素点的值的累加和，P_3的左上侧就是黄色框所包含的区域。 } {70％： 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。 } {73％： 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， } {83％： 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， {91％： 其积分图像中 A（x2， y1）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， } {90％： 同理积分图像中的 B（x2， y1）、C（x1， y2）、D（x2， y2）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4-8所示。

{45％： 图4-8 利用积分图像计算检测框像素值总和 }

{90％： 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（x2， y2）- sum（x1， y2）- sum（x2， y1）+ sum（x1， y1） （4-4）

公式表达的即是： D-C-B+A。 {93％： 显然，计算量不受区域尺寸影响。 } {100％： 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。

4.3.5 使用HIK SVM检测行人

使用线性SVM对图像进行检测可以快速排除那些不包含行人的检测窗口。 但是线性SVM判定时检测结果往往不够准确，有很多误检，因此这些窗口需要通过级联分类器的第二级HIK SVM进行进一步的分类。

采用线性SVM并行对检测窗口进行计算得到结果后，再根据得到的结果让HIK SVM求解。 由于HIK SVM训练好的模型是一个T表代替了w模型，所以第二阶段只需要查T表，并求和，就可以得到第二级分类器的结果。 得到第二级分类器检测的结果后，我们把得分大于零的检测框保存起来。 经过第二级分类器的进一步检测后，我们就可以去除掉很多不包含行人的检测框，有效地提高了检测精度。 假设有一幅原始图像如图4-9（a）所示，其线性SVM检测后的结果为4-9（b），HIK SVM 检测结果为4-9（c）。

（a）原始图像 （b）线性SVM检测结果 （c）HIK SVM检测结果

{48％： 图4-9 线性SVM和HIK SVM检测单个行人的结果对比}

4.3.6 用NMS处理结果

本文行人检测算法中的滑动窗口的滑动步长我们选取的是2个像素，因此多个相邻检测窗口可能都包含同一个行人。 因此一个行人会有多个检测结果，显然不符合我们的要求。 因此我们需要对级联分类器的检测结果进行后处理，过滤掉多余的检测结果。 {52％： 通常采用非极大值抑制的方法来实现。}

{65％： 非极大值抑制的定义是在局部区域内搜索极大值，抑制不是极大值的元素。} {67％： 这个局部指的是一个邻域，这里我们讨论的NMS算法是用于目标检测中的提取分数最高窗口的算法。} {79％： 例如在基于滑动窗口的行人检测中，提取窗口内图像的特征经分类器分类识别后，每个窗口都会得到一个分数。} {73％： 但是滑动窗口会导致很多窗口与其他窗口存在包含或者大部分交出的情况，此时NMS就用来选取邻域里分数最高，并且抑制分数低的窗口。}

{70％： 首先通过检测算法计算出所有窗口的得分，对这些窗口的得分进行从小到大的排序去除分数最高的序号，} {92％： 循环计算最高和次高分数窗口与最高分数窗口的交叉面积与两者间最小面积的比例，} 若超过overlap那么就把这一窗口抑制了。

4.4 BING-C4实验评估

4.4.1 应用场景行人库的建立

{42％： 本文使用的行人数据库使用的是INRIA行人数据库和自己录的一部分视频提取的图像，所有图像的分辨率都为640*480。} 包含不同年龄，不同衣着，不同姿态的行人。 {43％： 其中训练BING分类时，我们使用的是VOC2007的数据集，C4的模型我们则直接用了C4源码提供的模型。}

4.4.2 BING建议窗口数量实验

由于BING算法的建议窗口的探测率与提出建议窗口的数量有关，所以如何选取建议窗口的数量，对于检测的速度和精度都非常重要。 原则就是在尽量少的建议窗口数量的情况下要能达到很好的行人检测效果。 如图4-10，通过改变 BING建议窗口数量，人别在我们的行人数据库上进行行人检测测试， 我们发现太少的建议窗口，会使得检测区域覆盖不够全面，造成大量漏检。 随着建议窗口数量的增加，漏检率组件下降，到达5000左右时，漏检率已经逼近使用滑动窗口对整图进行检测， 在 INRIA数据库下平均 miss rate为0.16，不同数量窗口 Miss Rate统计如图4-8， ，所以本方法选择5000建议窗口作为标准参数。

图4-10 建议窗口数量对检测精度影响图

4.4.3 BING-C4检测速度对比实验

BING-C4方法的核心目的是通过BING似物性生成算法，减少行人分类器的识别空间，从而使检测熟读大幅度提高，把算法应用到什么实际项目中。 {48％： 通过上一节的实验，我么选择5000个建议窗口，保证了C4的识别

性能。} 在保证良好的检测性能的基础上，我们更加关心检测速度的改进。 {44％：为了体现对比性，选择了和过去经常使用的行人检测算法，经典的HOG+SVM算法，以及单独使用C4算法。}

{82％：本文的实验环境是Intel Core i7处理器，8GB内存。} 针对以上的2种算法和本文BING-C4算法进行检测速度的对比。 在两种数据库上BING-C4算法不但能保证较低的miss rate，而且检测速度也有很大提升。 本文比较了HOG+SVM与BING-C4的检测速度和效率，如表4-1所示，可见该方法在速度上有了很大的改进。

{40％：表4-1 两种行人检测算法在INRIA数据库的处理结果}

检测方法检测时间（每帧图）总检测时间平均漏检率

HOG+SVM0.572s420.37s45.85%

BING+C40.19s66.15s16.32%

为了更加直观的展现本方法最终的检测结果，我们自己在路面上采集了图像，得到的最终行人检测效果如图4-11所示。

图4-11 行人检测实验效果

4.4.4 结果分析

从实验结果可以看到，检测结果还有些漏检和误检，而且检测结果与图像质量有很大关系，如果行人和背景区分不明显就很容易漏检。 由于 BING- C4算法是在 BING生成的建议框中进行行人检测，所以我们看有的检测结果虽然检测到了人， 但是标注框却没有精确覆盖目标区域，如上图第三排第一张图，它的标注框远超过了行人真实的大小。 但是在实际应用中，已经可以帮助车辆识别路上的行人，而且由于检测速度较快，所以应用在汽车的ADAS上是可行的。 由于道路上情况很复杂，所以检测结果还有很多误检，所以必须多采集数据，将那些分类错误的图像再进行训练。

4.5 本章小结

本章介绍了基于BING和C4的行人检测算法的具体实现和实验结果，以及工程中我们使用的一些加速检测算法的技巧，比如OpenMP、积分图像等。 其中 OpenMP作为我们使用的并行计算方式来加速某些耗时的串行计算，当然也可以使用 GPU来实现并行加速，通过实验证明该方法可以保证行人检测的精度和速度， 并分析了实验结果，以及算法需要注意和改进的地方。

第5章 结论与展望

5.1 论文主要工作

行人检测是 ADAS和自动驾驶的核心技术，而且由于汽车硬件设备性能的限制，如何有效而又快速的识别行人是这类问题的难点， 用深度学习处理精度很高，但是对计算设备的性能要求很高，所以当前考虑到设备， 其实用性可能不是很高。 如果将来显卡成本降低，深度学习必将取代传统的基于统计学习的学习算法。

正是考虑到以上的困难，本文使用的基于似物性采样和基于统计学习的分类方法来解决检测速度的问题，首先介绍了似物性采样的相关概念和技术，并着重介绍了其中代表性的BING算法。 在第二章，我们了解到了BING是如何产生建议检测框，以及二值化梯度特征得到的BING特征的，使用二级级联SVM训练的BING模型等。

第三章我们详细介绍了核心的 C4算法，其中包括 CENTRIST特征为何能够很好的描述行人，还与之前应用广泛地 HOG特征进行对比， {40％：实验证明，在检测结果和检测精度上， CENTRIST特征都更有优势。} {40％：并详细介绍了C4算法中使用的线性分类器和HIK SVM的原理以及训练方法。}

第四章我们对BING+C4算法的完全实现过程进行了讲解，其中包含我们具体编程的一些东西，以及算法中有的技巧的应用来加速计算过程。 最后我们使用这个算法利用INRIA数据库以及我们自己拍的一些图像，对算法的精度和速度进行了评估，实验证明算法在保证一定精度的情况下，速度比较可观，可以应用在ADAS上。

5.2 未来研究工作展望

{40％：通过研究发现，目前常用的目标检测算法，都是基于滑动窗口进行检测。} {41％：本文提出的基于BING和C4的目标检测方法，可以减少算法识别的窗口数量，提升检测速度。} BING不光可以应用在行人检测还可以用在其他领域。 相对于传统的10万数量级的滑动窗口，BING提出的建议窗口数量只有几千个，大大降低了运算量，所以我们还可以将BING和其他许多分类算法结合起来。 比如BING+深度学习，BING+DPM，都可以起到提高检测速度的作用。 而且实验发现检测的精度依赖于建议窗口的覆盖率。 所以希望将来可以进一步改进算法，提高精度，提高速度。

2016年 CVPR上的行人检测综述总结了行人检测10年来的进展成果，提出了一种以人眼识别能力为基准的参考指标， 但是相对于人眼识别的速度和准确度上，那么计算机视觉的目标检测还有很大的提升空间。 当前深度学习在目标检测方面的应用取得了很好的效果，期待将来硬件的发展可以支持深度学习模型应用在ADAS上面。