基于边缘图的快速物体定位

学　　生：　周　航
学　　号：　11123610
指导老师：　沈　为
专　　业：　通信工程
完成年月：　2015 年 06 月

目录

摘要

物体检测是计算机视觉领域一个研究热点，用于检测图像中是否有待检测物体以及它的位置。一般来说，物体检测的目的在于检测图像或视频中出现的特定语义对象，例如行人，建筑，人脸等。在某些领域，如行人检测和车辆检测，已经有了较深入的研究。然而，目前流行的物体检测方法中，大数据情况下物体检测的速度不够快，比如需要在大型图像数据库中快速检测并定位图像中出现的物体的情况。

基于滑动窗口的方法是目前主流的检测方法。由于滑动窗口的方法需要在图像的每个候选的物体位置进行特征提取和分类，因此在计算时间上有很大的开销。

为了提高物体检测定位系统的实时性，本文主要提出了“边缘框”的算法，即直接从边缘产生物体的候选区域框。边缘聚类形成轮廓，根据完全位于候选区域框内的轮廓数量来决定候选区域的可能性。在最后，本文讨论了检测方法在检测不同种类的物体的准确性和效率，并分析了该方法的试用范围和应用前景。

关键词：物体候选区域，物体定位，边缘检测

Abstract

Object detection is a research hotspot in the field of Computer Vision, aiming at seeking whether an object exists in an image, and if so where in the image it occurs. Object detection is an important technology related to computer vision that focus on detecting instances of semantic objects, such as humans, buildings, or cars, in digital images and videos. Well-researched domains include pedestrian detection and car detection. However, the problem of big data is insufficient under this context, most popular method may hard to deal with the tasks of detecting objects in large scale image sets.

The dominant approach to the detection currently is sliding windows. However, feature extraction and object classification has to be performed at every location and scale in an image, which increases the computational cost.

To improve the real-time appearance in object detection, this paper put forward “Edge Boxes” method, which generates object bounding box proposals using edges. Edges cluster into contours. We define a scoring box based on the number of contours it wholly encloses to decide the possibility of the candidate object. The accuracy and efficiency of this method is tested in variety of detection task. In the last of this paper, the application and prospects of the method is analyzed.

Key words: object proposals, object detection, edge detection

第一章 绪论

1.1 课题研究的意义和应用前景

1.1.1 研究意义

众所周知，当前是信息时代，信息的获得、加工、处理以及应用都有了飞跃发展。人们认识世界的重要知识来源就是图像信息。在很多场合，图像所传送的信息比其他形式的信息更丰富、真切和具体。人眼和大脑的协作使得人们可以获取、处理以及理解视觉信息，人类利用视觉感知外界环境信息的效率很高。事实上，据一些国外学者所做的统计，人类所获得的外界信息有 80%左右来自眼睛摄取的图像。由此可见，视觉作为人类获取外界信息的主要载体，计算机要实现智能化，就必须能能够处理图像信息。尤其是近年来，以图形、图像、视频等大容量为特征的图像数据处理广泛应用于医学、交通、工业自动化等领域。

1.1.2 边缘检测简介

边缘检测是图像处理和计算机视觉中的基本问题，目的是标识数字图像中亮度变化明显的点。图像属性中的显著变化通常反映了属性的重要事件和变化，这些包括深度上的不连续、表面方向不连续、物质属性变化和场景照明变化等。边缘检测是图像处理和计算机视觉中，尤其是特征提取中的一个研究领域。

1.1.3 物体检测简介

物体（object）具有以下 3 个特点：（1）在图像中，物体的边界是闭合的；（2）物体的外观与背景环境不同；（3）有时候物体从背景环境中凸显出来。

一般说来，物体检测可以分为两种：静态物体检测和动态物体检测。前者检测在图像中出现的物体的实例，后者目的在于将图像序列中某种运动的区域分割出来。本文偏重讨论静态物体的实例。

1.2 课题研究国内外现状

1.2.1 物体检测的现状

当前热点的物体检测问题包括人脸检测和行人检测等。人脸检测被认为是研究最成功的一类物体检测问题，在近十几年中有多种多样的人脸检测算法被开发出来并投入使用。2002 年 Yang，Kriegman 等人对早期的相关研究做了总结[1]，提出人脸检测可分为基于特征、基于模板以及基于外观的三类方法。同时这样的分类也适用于现今出现的大多数人脸检测方法。基于特征的检测方法尝试寻找人脸中有明显特征的部位，如眼睛、鼻子、嘴等来进行检测。其中著名的方法包括[2][3]等。基于模板的方法会构建一个目标的模板去匹配图像中的物体，比如著名的 AAM 模型[4]就属于这类方法，该类方法可以应对目标物体有较大形状变化的情况。基于外观的方法是比较通用同时也是比较流行的一种方法。这种方法一般采用扫描图像中重叠的矩形区域进行物体检测，其中比较著名的是 Viola 和 Jones 提出的快速目标检测框架[5]，可以同时保证准确性和高效性。其次被广泛关注的物体检测问题是行人检测和车辆检测。其中一种著名的行人检测算法在 2005 年由 Dalal 和 Triggs[6]提出，并在提出后迅速流行起来。该方法使用 HOG 特征和 SVM 分类器进行行人检测，可以获得较高的准确率。其他行人检测算法以及一些通用物体检测算法同时也迅速被提出，如[7][8][9]等。

1.2.2 物体检测的难点

判断每个位置是否存在物体的最为常用的方法是滑动窗口，但是这个方法有很高的计算开销。考虑大数据情况下的物体检测时，滑动窗口的检测速度不够快，比如需要在大型图像数据库中快速检测并定位图像中出现的某种物体的情况。

由于物体候选区域这种技术无需对滑动窗口的每个位置和大小进行计算来判断物体是否存在，因此物体候选区域可以提高物体检测的计算效率。采用物体候选区域生成器来替换检测子的目的是为了减少计算开销，因此候选区域生成器的方法计算速度更快。

本文提出了“边缘框”的算法，直接从边缘产生物体的候选区域框。边缘（edge）不仅稀疏，而且含有大量图像的信息，因此使用边缘在计算上具有很多优势。

1.3 内容安排

本篇毕业设计论文主要内容是自然图像边缘图的计算，基于边缘图的候选区域特征计算以及快速物体定位算法设计与实现。

第二章的内容是自然图像边缘图计算，主要介绍结构随机森林的训练和预测算法。同时，还比较了不同边缘检测算法的优缺点，以及选用结构随机森林分类器检测边缘的优点。

第三章的内容是基于候选区域的特征计算，主要介绍某一候选区域已知的情况下计算其特征（评分值）的算法。针对该算法，第三章引出了边缘组、相似度的概念，以及构建两个数组来快速寻找横跨（交叉）边界的边缘组的算法；同时详尽地介绍了候选区域框的特征计算方法。

第四章的内容是快速物体定位算法设计与实现，主要介绍物体定位准确度的定义、滑动窗口步径的设置，以及迭代搜索的算法。

第五章的内容是本算法的实验结论及程序实现。从实验结果中，可以看出“边缘框”与其他算法相比，具有很高的物体检测率。同时，第五章还介绍了 Piotr Dollár 的工具箱以及程序的运行过程、演示系统、总体结构图和代码的实现等。

第六章是本文的总结。

第二章 自然图像边缘图计算

2.1 快速边缘检测算法简介

边缘检测是计算机视觉研究领域的一个热点问题。获取到的物体边缘可以解决很多问题，比如物体定位、物体分割、获取物体轮廓等。传统的方法是：采用基于局部信息的特征，使目标检测和识别算法具有较强的鲁棒性。局部特征是根据某种显著准则定义的，与其领域相区别的图像表示方法，通常与一种或几种图像属性（颜色、纹理、方向）的改变度相关联。常用的局部特征包括 SC(Shape Context)、SIFT(Scale Invariant Feature Transform)、SURF 等。常用的边缘检测方法有：计算色彩梯度值并进行非最大值抑制获得边缘（但是无法识别纹理边缘及看不清的轮廓），使用大量特征（亮度、色度、纹理梯度等）获得边缘，谱聚类获得边缘等。

运用学习机制的边缘检测是计算机视觉领域的一个重要方向。随机森林是机器学习领域中一种有效的组合学习模型。在目标检测识别算法中，随机森林被证实在分类中具有很好的性能和效果。[9]中提出了基于霍夫森林模型的目标检测算法，该算法使用霍夫森林对霍夫图像进行投票，获得了很好的目标检测效果。本文提出了一种以随机森林作为分类器的结构模型学习算法。使用含类别信息的局部特征来构建随机森林，从每个决策树的根节点开始，递归地分裂每个节点直至节点不能再分裂（形成叶子节点）。在每个叶子节点中，存储到达该节点的局部特征的偏移量和类别信息、该节点的类别信息。利用类别监督信息可以更好地优化码本模型，获得更可靠的概率投票，加快目标检测速度。随机森林的其中一种方法：将图像分割成多个图像块（patch），对每个图像块判断中心区域是否有边缘。然后将这些小块联合起来得到整个边缘图。图像块中的边缘相互具有关联，各具形状，比如：直线、平行线、T 型线、Y 型线等。本文提出的监督学习机制称为结构学习，采用结构随机森林分类器，对图像进行训练和预测边缘图。下面将介绍边缘检测的详细算法。

2.2 随机决策森林

随机森林，指的是利用多棵树对样本进行训练并预测的一种分类器，见图 2-1。该分类器最早由 Leo Breiman 和 Adele Cutler 提出，并被注册成了商标。简单来说，随机森林就是由多棵 CART（Classification And Regression Tree，分类和回归树）构成的。对于每棵树，

它们使用的训练集是从总的训练集中有放回采样出来的。这意味着，总的训练集中的有些样本可能多次出现在一棵树的训练集中，也可能从未出现在一棵树的训练集中。在训练每棵树的节点时，使用的特征是从所有特征中按照一定比例随机地无放回的抽取的。根据 Leo Breiman 的建议，假设总的特征数量为M，这个比例可以是$\sqrt{M}$，$\frac{1}{2}\sqrt{M}$或$2\sqrt{M}$。

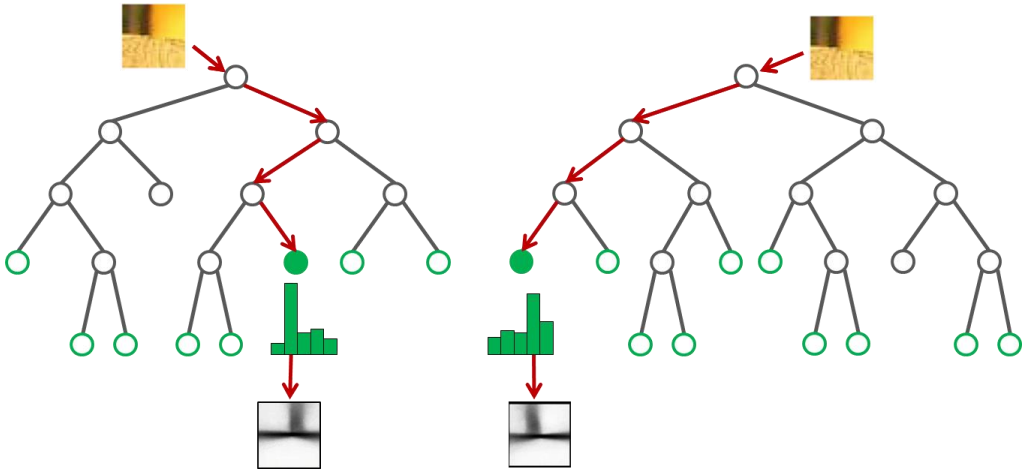


图 2-1 随机森林示意图

2.2.1 训练决策树以及信息增益

每一棵树都通过递归的方式进行训练。为了将结果划分为叶子的左节点或右节点，下面定义了信息增益的公式（1）来决定叶子的划分。

对于分类问题：

$$I(j) = H(S_j) - \sum_{k\in\{L,R\}} \frac{|S_j^k|}{|S_j|} H(S_j^k) \tag{1}$$

其中，$H(s) = -\sum_y p_y log(p_y)$，$p_y$指节点$j$各类所占比例，$H(s)$就是某个节点训练集的信息量（香农熵）。通俗地说，根节点的信息增益是该节点的信息量减去左右两个节点的信息量。信息增量$I(j)$取最大的时候，分类越纯。

本文在训练过程中用来分裂节点，因此采用的是分类算法。

2.2.2 随机化

如果训练过程中只采用一棵树，那么很有可能会使这棵树出现过拟合的现象。过拟合（overfitting）的定义：训练数据应用到决策树时，正确分类率很高；测试数据应用到决策

树时，正确分类率很低。因此，对多棵去相关的树进行训练，并将它们的输出结合起来的方式可以解决过拟合的问题。

对于每棵树，它们使用的训练集是从总的训练集中随机地有放回采样出来的；在训练每棵树的节点时，使用的特征是从所有特征中按照一定比例随机地无放回的抽取的。这两处的随机抽样保证了特征分类的准确性。

2.3 结构随机森林

以下是随机森林的训练过程：

(1)给定训练集S，测试集T，特征维数F。确定参数：使用到的 CART 的数量t，每棵树的深度d，每个节点使用到的特征数量f，终止条件：节点上最少样本数s，节点上最少的信息增益m。

(2)从S中有放回的抽取大小和S一样的训练集$S(j)$，作为根节点的样本，从根节点开始训练。

(3)如果当前节点上达到终止条件，则设置当前节点为叶子节点：如果是分类问题，该叶子节点的预测输出为当前节点样本集合中数量最多的那一类$c(j)$，概率p为$c(j)$占当前样本集的比例；如果是回归问题，预测输出为当前节点样本集各个样本值的平均值，然后继续训练其他节点。如果当前节点没有达到终止条件，则从F维特征中无放回的随机选取f维特征。利用这f维特征，寻找分类效果最好的一维特征k及其阈值th，当前节点上样本第 k 维特征小于th的样本被划分到左节点，其余的被划分到右节点。继续训练其他节点。

(4)重复(2)、(3)直到所有节点都训练过了或者被标记为叶子节点。

(5)重复(2)、(3)、(4)直到所有 CART 都被训练过。

本文采用的结构随机森林与已有的随机森林算法有点区别，见图 2-2。训练结构随机森林具有挑战性：首先，结构输出的空间维数很高，复杂性大；其次，结构分类器的信息增益很难定义。

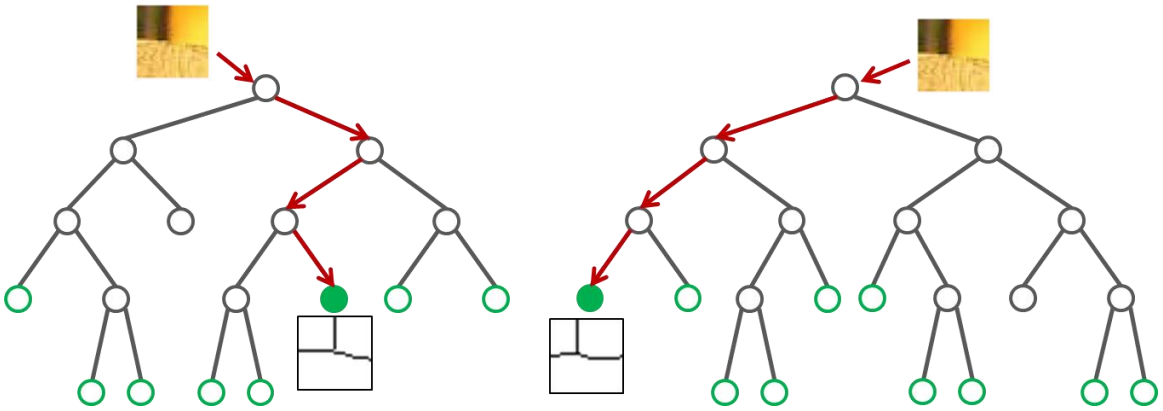


图 2-2 结构随机森林示意图

通过实验已证明，一个节点分类器不需要很精确（因为有多个节点分类器级联）。本文提出的方法是：对每个节点处的分类$y \in Y$，用一个映射函数映射到一组离散的分类c中（$c \in C$，其中$C = \{1,2,\dots,k\}$）。其含义是：相似的分类y聚类到同一个分类中。对于二分类（分为左右两个子节点），$k = 2$。对于每个节点的分类Y计算信息增益。由于很难定义一个函数对分类Y计算其信息增益，因此本文提出，建立一个过渡的空间Z，在这个Z空间中，我们可以进行比较。

2.3.1 中间的映射Π函数

对于边缘检测，一个 16×16 大小的模板代表一个节点处的一种分类。定义映射函数：

$$z = \Pi(y) \tag{2}$$

其中，映射方式为：模板y中的任意两个像素点组成一对，如果这一对点属于模板中同一块（图像中相同的分割部分），则输出 1；否则为 0。

这样，一个分割模板就映射为$C_{16\times16}^{2} = 32640$维的特征。为了提高计算效率，则进行降维，从中任意选取$m = 256$维。接下来，可以通过采用主成分分析法（PCA）进一步降维降至最多 5 维。实际中，我们降至 2 维，进行二分类。

2.3.2 信息增益标准

已知Z平面的z向量，根据第二节中提出的信息增益公式，可以对模板y进行分类。

已知Z平面的z向量，也可以构建相应的分类C平面。本文采用 PCA 量化方式量化z向量，选用$log_2(k)$个重要的变量。重要的变量指某个维度上值最大时的这个维度。实际中选用$k = 2$。

2.3.2 组合模型

在Z域上，若k不为 2，那么z_k的中点所对应的模板y_k就是所求的分类。

对于组合模型，即结构随机森林来说，每一棵树训练时选取n个模板，这n个模板只需粗糙地获取z_k中心点足以能选取n个合适的y_k。对于边缘提取，在训练过程中使用默认的组合模型；输出的时候通过一定的方式将多个结果合并起来。具体过程见下节。

2.4 边缘检测

2.4.1 边缘检测具体过程

具体的训练流程：

第一步，将图像数据集对应的基于分割的数据集中所有图像分为多个可重叠的 32×32 的分割图图像块。每一个分割图图像块中获取位于中心的 16×16 大小的模板（mask）。定义模板：映射至Z域后的m维特征。一个模板映射为$C_{16\times16}^2 = 32640$维的特征。为了提高计算效率，则进行降维，从中任意选取$m = 256$维。接下来，通过采用主成分分析法（PCA）进一步降维降至 2 维（即进行二分类）。根据二分类，对所有的分割图像块贴上标签 0 或 1，归为两类。定义边缘块：由 16×16 大小的二值像素点构成，1 代表有边缘，0 代表无边缘。模板通过一定的算法可以得到边缘块。在训练过程中，边缘块是从边缘图像中获取的，而边缘图像可以直接从分割图像数据集的图像中直接获取。

第二步，计算每个原图图像块（32×32）的特征。首先，对某块原图图像块进行处理，使每个像素都具有K维信息，K取 13，包括 3 个颜色通道（RGB）、2 个梯度幅值通道、8 个方向通道。接下来，对整块原图图像图像块特征进行降维：32×32×13/4=3328 维。再加上每个通道包含 5×5 分辨率的成对像素信息，因此每个图像块共含有 3328+13×$C_{5\times5}^2$=7728 维特征。至此，每个特征对应一个边缘块以及一个标签（0 或 1）。

第三步，构建决策树。以下是训练决策树的过程。在原图图像块集的任意选取若干个原图图像块，通过第二步的方式获取每一个原图图像块的特征，并对所有的特征在每一维度上进行排序，并进行二分类，随后通过标签的类别计算信息熵。在这 7728 维中信息熵最大情况所分类的维度就作为根节点，将图像块分为两类。这样的分类方式，时间开销少，而且分类效果好。每个节点中存放维度信息、阈值和边缘块。按照相同的方式，对左右两

个节点中的图像块继续进行分类，分类终止的条件：树的深度达到一定的条件（最大为 64）或者分类到只剩一个分节点（左节点或右节点）或者某个节点的图像块数量足够少（小于等于 8 个图像块时停止分裂节点）。

第四步，构建结构森林。第三步是对一棵树进行构建分类器。本算法实际构建了 4 棵树，每棵树的图像块数据是有放回地从图像块集中获取的。这样能保证不会出现过拟合现象。

具体的边缘检测流程：

第一步：输入一张检测图像。将检测图像分割为有重叠的 32×32 大小的原图图像块。对每个图像块计算其 7728 维的特征。

第二步：将这些图像块分别放入多个决策树中，可得到多个 16×16 大小的边缘图像块。对每个决策树的输出边缘图像块进行平均处理，然后将多棵树输出的分类结果进行合并，得到最后的边缘图像。

2.4.2 与其他边缘检测算法的比较

常用的边缘算子，比如 Sobel，Laplace、Prewitt、Canny 算子等，都是基于图像中像素的梯度信息得到的，某些人眼无法识别的梯度也能显示出来；同时，很多边缘在检测过程中会出现很多散点，因此获取到的边缘图像并不是很准确。见图 2-3。

基于结构森林的快速边缘检测算法具有学习性质，而边缘图是人为标定的，因此检测出来的边缘图的效果更好。见图 2-4。

图 2-3 基于结构森林的边缘检测

图 2-4 基于像素的边缘检测

2.5 本章小结

本章节介绍了获取自然图像的边缘图的算法。本文的边缘图获取算法主要采用的是结构随机森林的训练和预测的方式。同时，本章还比较了不同边缘检测算法的优缺点，以及选用结构随机森林分类器检测边缘的优点。下文将介绍在已有边缘图像的情况下，一种全新的计算物体候选区域特征的方法。

第三章 基于候选区域的特征计算

3.1 特征计算流程

这一章将详细地描述计算某一候选区域特征的算法。为了可以有效地将完全位于框内的轮廓从非框内的轮廓中分离出来，本文通过建立并使用基于边缘组的数组来提高效率，然后定义基于边缘的计分函数。

已知源图像（见图 3-1），首先对每个像素计算其边缘响应。使用第二章得到的结构边缘检测器[12, 15]就能得到边缘响应。结构边缘检测器在预测物体边界方面具有良好的效果，并且计算效率也很高。然后使用单一尺度变化以及论文[17]中介绍的边缘强化来减少运行时间。接下来，对边缘响应进行非最大值抑制（NMS）的计算，这与边缘响应正交，能够找到边缘的峰值点，得到图 3-2 的边缘图像。输出的结果是一个稀疏的边缘矩阵图。每一个边缘像素p具有边缘方向矢量向度m_p和方向θ_p。将那些像素的边缘方向矢量向度$m_p > 0.1$的定义为边缘。（对边缘进行阈值化是为了提高计算效率。）轮廓（contour）的定义：一组连续的边界或曲线或直线组成的边缘集合。边缘图像可以直观地被看成源图像的轮廓图。

图 3-1 原图像

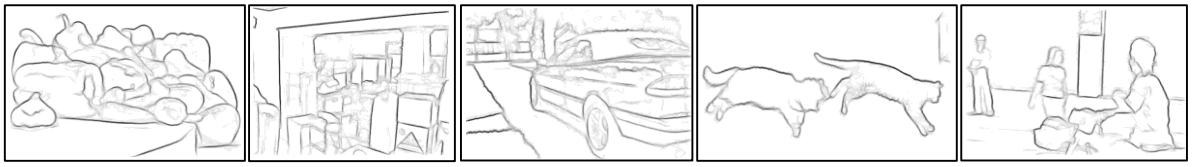


图 3-2 边缘图像

3.2 边缘组

边缘组（edge group）由具有高相似度的边缘聚集而来。定义：若 8 相连的边缘的方向夹角差之和大于某一个阈值（比如$\pi/2$），则将这 8 相连的边缘连接起来。同时，较小的边缘组应与较大的边缘组合并。根据边缘图可以计算得到所有边缘组的集合S。

3.3 相似度

对任意一组边缘组$s_i \in S$，计算两两边缘组的相似度。相似度的定义：对于任意的s_i和s_j，相似度$a(s_i, s_j)$由这两个边缘组的中心x_i、x_j以及平均方向θ_i和θ_j计算得到。计算公式如下：

$$a(s_i, s_j) = |\cos(\theta_i - \theta_{ij})\cos(\theta_j - \theta_{ij})|^{\gamma} \tag{3}$$

其中，θ_{ij}是中心x_i和中心x_j边缘方向之间的夹角。γ的值可以被用来调整相似度在方向上改变的的敏感性；实际操作中，γ取 2。如果两个边缘组的距离大于 1 个像素大小，那么它们之间的相似度设置为 0。为了提高效率，当相似度仅仅大于一个小的阈值（比如 0.05）时，这个相似度就被记录下来；而小于这个阈值时，相似度设置为 0。

边缘组和相似度的计算开销相当少，而且边缘组的细节使实验结果具有鲁棒性。

3.4 边界框的计分算法

已知边缘组集合S和任意两个边缘组之间的相似度，根据一定的计分算法就可以计算任意候选框区域b的特征值。首先，计算所有边缘组s_i中所有边缘p的边缘矢量向度m_p的和m_i（即$m_i = \sum m_p$）。然后，选取每个边缘组s_i中任意一个像素p的位置$\overline{x_i}$。对$p \in S_i$，选取某一确切的像素位置也是可以的。

对每个边缘组s_i，计算一个权值$w_b(s_i) \in [0,1]$，用来表示：若s_i完全处于候选框b以内，则$w_b(s_i) = 1$；若不是，则$w_b(s_i) = 0$。记S_b为横跨候选框b边界的边缘组的集合。通过在第 3.5 节中描述的有效数据结构来发现S_b。然后，对所有的边缘组$s_i \in S_b$，设置$w_b(s_i)$为 0。同样地，对所有满足包含$x_i \notin b$的边缘组s_i，$w_b(s_i) = 0$，因为所有满足$w_b(s_i) = 0$条件的像素要么在边界框b的外面要么$s_i \in S_b$。对于剩下的满足$x_i \in b$且$s_i \notin S_b$的边缘组，对$w_b(s_i)$按照以下方法进行计算：

$$w_b(s_i) = 1 - \max_{T} \prod_{j}^{|T|-1} a(t_j, t_{j+1}) \tag{4}$$

其中，T是一条有序路径，其长度为$|T|$，起始位置为某一个横跨边缘的边缘组$t_1 \in S_b$，结束位置为某一边缘组$t_{|T|} = s_i$。如果没有这样的路径，那么令$w_b(s_i) = 1$。因此，公式（4）

能够找到任意边缘组s_i和横跨边缘框边界的边缘组两者之间的具有最高相似度的路径。由于大多数两两边缘组之间的相似度为0，因此这个做法能够提高计算效率。

使用计算得到的权值w_b，定义以下的计分公式（5），h_b为候选区域的特征值（评分）：

$$h_b = \frac{\sum_i w_b(s_i) m_i}{2(b_w + b_h)^K} \qquad (5)$$

其中b_w为候选区域框的宽度，b_h为候选区域框的高度。要注意的是，由于边缘是由一行像素点组成的，因此分母用的是周长而不是面积。尽管如此，K取1.5的原因是为了抵消更大窗口平均具有更多边缘而产生的偏差。

在编程中，采用积分图来加速计分公式（5）中分子部分的计算速度。积分图用于计算对于任意$\overline{x_i} \in b$的所有m_i之和。接下来，对于所有$\overline{x_i} \in b$的s_i，若$w_b(s_i) < 1$，则$(1 - w_b(s_i))m_i$的值从以上所有m_i之和中减掉。以上算法大大提高了计算速度，因为对于大多数的边缘组s_i，有$w_b(s_i) = 1$，而且所有这样的边缘组s_i不需要一个个考虑。

最后，据观察，框中心的边缘相对于框边界附近的边缘并没有起很大作用。为了解决这个现象对结果产生的偏差，在原有计分公式（5）h_b的基础上减去从位于框b中心的框b^{in}的边缘梯度值：

$$h_b^{in} = h_b - \frac{\sum_{p \in b^{in}} m_p}{2(b_w + b_h)^K} \qquad (6)$$

其中，b^{in}中的宽度为$b_w/2$，高度为$b_h/2$。b^{in}边缘梯度的和可以通过积分图进行计算。在第 5 章实验中，可观察到h_b^{in}比h_b的计算开销稍大，但是h_b^{in}具有更好的物体检测的准确度。

3.5 寻找跨边界的边缘组

在第 3.4 节中，假设横跨框边界的边缘组集合S_b已知。由于要评价的边缘组的数量很大（第 4 章），因此建立一个寻找S_b的有效算法非常重要。对于较大的候选区域框，详尽地搜索框的每一个边界所有像素的方式将带来很大的运算量。

因此，下面将提出一个有效的理论，即候选区域框的每条边依靠两个额外的数组来寻找相交的边缘组。下面将描述寻找从像素点(r, c_0) 到(r, c_1)的一条水平方向的边界的情况。垂直方向的边界可以通过同样的方法进行处理。对于水平方向的边界，对每一行像素建立两个数组。第一个数组存储边缘组s_i的有序表L_r，这个有序表指向图像第r行。如果边缘组

的下标从一个像素点变换到另一个，那么有序表L_r的下标也增加一个。有以下结论：有序表L_r的长度比图像的宽度要小得多。如果边缘组之间的像素非边缘，那么这个有序表中就添加一个 0。第二个数组K_r的长度跟图像的宽度相同，用来存储对应每一列c列第r行的有序表L_r的下标。因此，如果位于坐标(r, c)的像素p是边缘组s_i的一部分，那么$L_r\big(K_r(c)\big) = i$。因为大多数的像素不属于边缘组，因此使用这两个数组，通过对有序表L_r的下标$K_r(c_0)$到下标$K_r(c_1)$探索能够有效地找到横跨的边缘组的序列。

3.6 本章小结

本章节介绍了在某一候选区域已知的情况下计算其特征的算法。针对该算法，本章节引出了边缘组、相似度的概念，以及构建两个数组来寻找横跨边界的边缘组的算法。本章节还详尽地介绍了候选区域框的特征计算方法。下文将介绍如何快速选取候选区域的算法。

第四章 快速物体定位算法设计

4.1 物体定位准确度

寻找物体轮廓时，应考虑采用目标检测的算法。一些检测算法需要高精度的物体轮廓，而另一些可以容忍边界框位置所产生的误差。边界框精度的典型测量方法是使用“交集除以并集”（IoU）的计算方式。IoU 用来表示候选区域框与物体真实值所在框的交集面积除以它们并集的面积。评价物体定位算法时，设定 IoU 的阈值为 0.5。这主要用于决定物体是否正确地检测出来[11]。在图 4-1 中可以看到，IoU 的值为 0.5 时检测是相当松散的。由 IoU 值为 0.5 时得到的检测算法得到的特征值也较低。在图 4-2 中可以看到，IoU 值的选取不同，会造成误检率和漏检率的不同。因此有以下结论：物体定位时，选择 IoU 值大于 0.5 时是比较合适的。

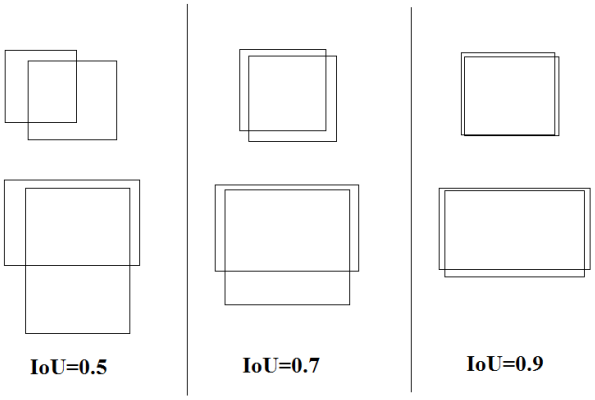


图 4-1 以上是 IoU 值分别为 0.5,0.7 和 0.9 任意几个的候选区域框的图。选取 IoU 值为 0.7 时的物体检测准确度比较合适。

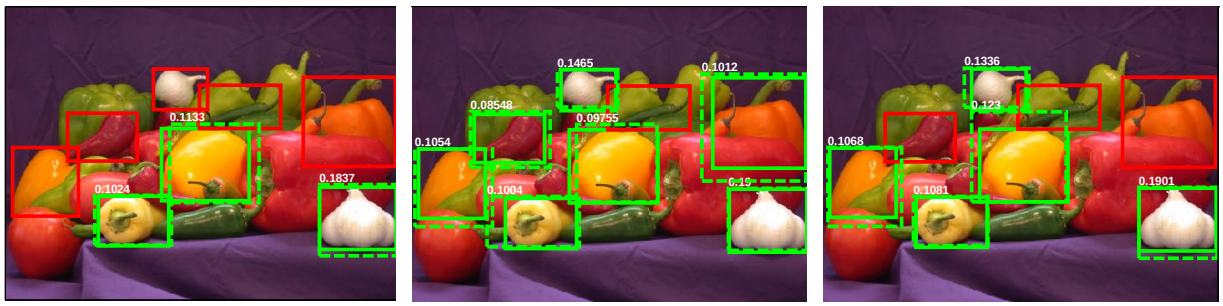


图 4-2 以上是 IoU 值分别为 0.5，0.7 和 0.9 时，物体的检测情况

4.2 物体轮廓探索策略

这一节在基于合适的 IoU 值δ下，提出了一个通过候选区域检测子来对搜索物体轮廓的策略。对于δ值更大的情况，能够生成一个更为集中于物体可能出现位置的边界框的集合。对于δ值更小的情况，由于已经假设物体检测器能够允许框位置适度的误差，边界框可能出现的位置有很多种可能性。因此，根据实际情况，可以在一个精度更高而物体候选数量更少，与一个精度更差而物体候选数量更多之间找到一个权衡。实际上，不同算法中的候选区域特征的计算对δ值有一些偏差，比如说对于候选区域“Objectness”[10]应选取有更小的δ值，对于候选区域“Randomized Prim”[14]应选取更大的δ值。本算法同时也能明确地控制准确度。

接下来，使用滑动窗口对每一个位置、滑窗面积和纵横比（高宽比）寻找物体的候选区域。步径的参数是α，α与候选区域框的 IoU 值有关。当滑动窗口的平移长度（步径）、划窗面积大小和纵横比确定下来时，滑动窗口移动一个步径后就能得到所期望的 IoU 值α。划窗面积大小的范围是介于框面积最小值$\sigma = 1000$个像素值和整幅图像面积之间的。纵横比的范围是$1/\tau$到τ，一般情况下$\tau = 3$。第 5 章节中将讨论：对大多数的δ值，α的值取0.65是最为理想的。若δ需要取高精度，$\delta > 0.9$，那么α应增加到 0.85。

4.3 迭代搜索

滑动窗口对整幅图像中物体候选区域进行搜寻之后，大于某一阈值的特征值h_b^{in}所对应的候选区域位置保存下来，其余的候选区域去除。接下来，对剩余的候选区域进行优化处理，以获取最合适的区域。优化的方法：使用迭代搜索算法，对每一个位置、面积大小和纵横比再次搜索候选区域以寻找最大的h_b^{in}。随着迭代次数的增加，滑动窗口的步径长度就减少为原先步径长度的一半。一旦移动步径的长度小于 2 个像素，则停止搜索。

候选区域框优化之后，对所得到的最大分值（特征值）记录下来并进行排序。接下来对排序好的边界框内的边缘进行非最大值抑制处理（NMS），对近乎同一个位置产生的多个候选框选择最优的一个，其他的去除，然后再重新排序。如果一个候选框的 IoU 值很高，而且大于某一个β值，那么这个候选框就被移除。在第 5 章实验已验证，设置$\beta = \delta + 0.05$时，无论δ值取何，都能够实现较高的准确度。

4.4 本章小结

本章节介绍了快速物体定位的算法。针对该算法，本章节引出了物体定位准确度的定义、滑动窗口步径的设置，以及迭代搜索的算法。第 2、3、4 这三章提出的算法可以得到某一张图像中的物体位置，而且运算速度接近实时。下文将分析该算法的实验结果。

第五章 实验

5.1 测试条件

为了与已有算法[13,11,14,10]的实验条件相同，本算法采用 PASCAL VOC 2007 数据集。这个数据集包含 9963 张自然图像。

验证数据集上将显示不同变量下的结果，测试数据集将显示本方法与其他方法的比较结果。

5.2 不同变量情况

接下来对验证数据集做了多组实验，测试各个变量对检测准确率的影响情况。图 5-1[20]（a，b）是生成 1000 个候选区域框时，控制变量α、变量β和 NMS 阈值对检测准确率影响情况的算法曲线，其中α和β用来控制滑动窗口的步径长度。

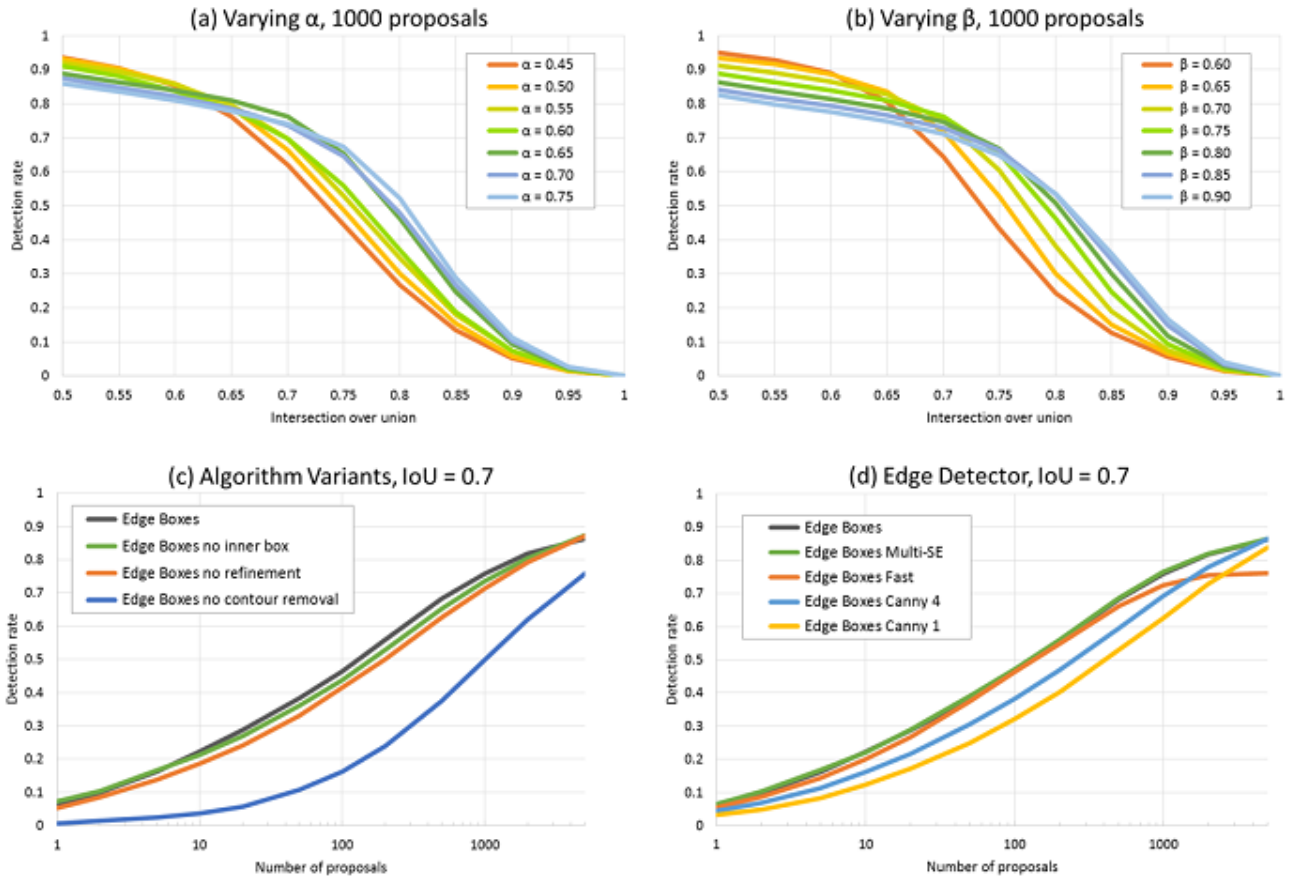


图 5-1 不同控制变量的检测准确率的比较结果。（a）不同的α值下，检测准确率随 IoU 值δ变化的曲线图（默认α的值为 0.65）。（b）不同β值（β控制 NMS 的阈值，β默认值为0.75）

下，检测准确率随 IoU 值δ变化的曲线图。（c）不同计算因子从算法中去除后（包括去除内部的边缘框、去除边缘框位置的优化步骤、去除横跨边界框的轮廓），检测准确率随候选区域框数量变化而变化的曲线图。（d）使用不同边缘检测子（包括本文中提出的算法使用的单尺度结构边缘[15]、多尺度结构边缘、论文[17]中介绍的运算速度极快且不需要边缘增强的算法，检测准确率随候选区域框数量变化而变化的曲线图。

	IoU=0.5		IoU=0.7		IoU=0.9				
	AUC	Recall	AUC	Recall	AUC	Recall	Runtime	α	β
Edge boxes 50	**.64**	**96%**	.36	55%	.04	5%	**.25s**	.65	.55
Edge boxes 70	.58	89%	**.45**	**76%**	.06	9%	**.25s**	.65	.75
Edge boxes 90	.38	59%	.28	46%	**.15**	**28%**	2.5s	.85	.95

表 5-1 本文提出算法的 3 个变量的准确度测量与运行时间：边缘框 50（Edge Boxes 50），边缘框 70（Edge Boxes 70）和边缘框 90（Edge Boxes 90）。准确度测量的方法包括：曲线下面积、1000 个候选区域下的召回率。α和β值见表右侧。其他参数保持不变。

α值的增加，使滑动窗口步径变短，候选区域框的采样密度增加，因此更多的候选区域框需要计算特征值，程序运行时间同时增加，见表 5-1。$\alpha = 0.65$时，物体检测的召回率和 AUC 比$\alpha = 0.70$和$\alpha = 0.75$情况下的值稍好一点。因此，在 IoU 值δ可以较低的时候，$\alpha = 0.65$足够提供很好的物体检测准确度。我们可以通过控制 IoU 值δ来调整参数β的值。设置$\beta = \delta + 0.05$时，无论δ值取何，都能够实现很好的物体检测准确度。在表 5-1 中，可以看到β对运行时间的影响最小。

表 5-1 中分别是三个变量：边缘框 50（Edge Boxes 50），边缘框 70（Edge Boxes 70）和边缘框 90（Edge Boxes 90）表示不同 IoU 值$\delta = 0.5$，0.7和0.9和不同α及β值下，AUC 和召回率的值。IoU 值δ较高时，候选区域框很紧密，因此参数α的值必须调整合适，通过提高运行时间来寻找更加紧密的结果。否则，对于要求不高的 IoU 值δ，参数α的值就设置为固定的某个值就可。

图 5-2 上图中，中间的图像代表：算法中没有去除中间部分轮廓得到的特征图；右边的图像代表：算法中去除中间部分轮廓得到的特征图。如果没有移除中间部分的轮廓，特征图像就会缺少清晰的峰值。为了便于观察，特征值的峰值经过归一化处理。图中的矩形框就是相对应的用于查找候选区域的滑动窗口。

第二个实验测试项目是测试算法中的不同算法的边缘检测子对物体检测准确率的影响，见图 5-1（c，d）。本实验中设置δ值为中间值0.7，观察物体检测准确率随候选区域数量增加而变化的曲线图。本论文提出核心算法是：计算候选区域框特征值时，横跨边界框边界的轮廓应该被移除，这样得到的特征值更加准确。实验证明，如果这些横跨边界框边界的轮廓不移除，物体检测的准确率就会下降很多。为了能更直观地观察到这个结果，我们用一个计分算法来表示特征值，用作结果。是否考虑横跨边界框轮廓的两种情况的结果显示在图 5-2[20]。考虑横跨边界轮廓的图像在物体边缘聚类附近有很强的峰值响应，更有可能形成物体。注意图中左下角对应的货车，对于去除边界轮廓的情况，图中的峰值响应很大。当边界轮廓不移除时，图中到处都能观察到强的响应。

如果框中心的边界框不移除，即不使用h_b^{in}而使用h_b，可以发现在准确度上有所降低。如果不进行边界框位置的优化，那么结果的准确度也会有所降低。原始的边缘检测子的选择也很重要，见图 5-1（d）。如果使用基于梯度值的 Canny 边缘[19]检测子，而不采用结构边缘检测子[15]生成最初的边缘矩阵图，物体检测的准确度就会降低。如果采用多尺度结构边缘而不采用单一尺度边缘，物体检测的准确度稍微有些提升。由于单一尺度边缘的运算比多尺度边缘的运算更快，因此接下来的实验中，均采用单一尺度边缘作为边缘检测子。实验中，每张图像使用单一尺度边缘检测子检测边缘的时间约为 0.25 秒。

如果想要加快物体检测速率，则需要调整参数。检测 1000 个候选区域框的情况下，物体检测准确度会稍微降低一些。举例说明，可以选取一个较低的a值 0.625，同时将候选区

域边缘框优化算法的阈值 0.01 增加到 0.02。而且，如果本文提出的算法不采用结构边缘检测子[17]的边缘强化算法，检测每张图像物体的时间将减少到 0.09 秒。在图 9（d）中可以看到，标注为“边缘框快速算法（Edge Boxes Fast）”的曲线，如果检测的候选区域框数量少于 1000，那么得到的物体检测精确率与边缘框算法的物体检测精确率几乎相等。

5.3 与其他算法的结果比较

	AUC	N@25%	N@50%	N@75%	Recall	Time
BING	.20	292	-	-	29%	**.2s**
Rantalankila	.23	184	584	-	68%	10s
Objectness	.27	27	-	-	39%	3s
Rand.Prim’s	.35	42	349	3023	80%	1s
Rahtu	.37	29	307	-	70%	3s
Selective Search	.40	28	199	1434	**87%**	10s
CPMC	.41	15	111	-	65%	250s
Edge boxes 70	**.46**	**12**	**108**	**800**	**87%**	**.25s**

表 5-2 在 IoU 阈值为 0.7 的情况下，本文提出的算法边缘框 70（Edge Boxes 70）与其他方法的比较。结果按照曲线下面积（AUC）的大小按照从小到大的顺序排列。其他度量方式包括实现 25%、50%和 75%召回率的候选区域数量，以及在 5000 个候选区域框的情况下最大的召回率。在多种测量结果中，边缘框算法基本上是最好的方法。时间开销结果由论文[18]得到。

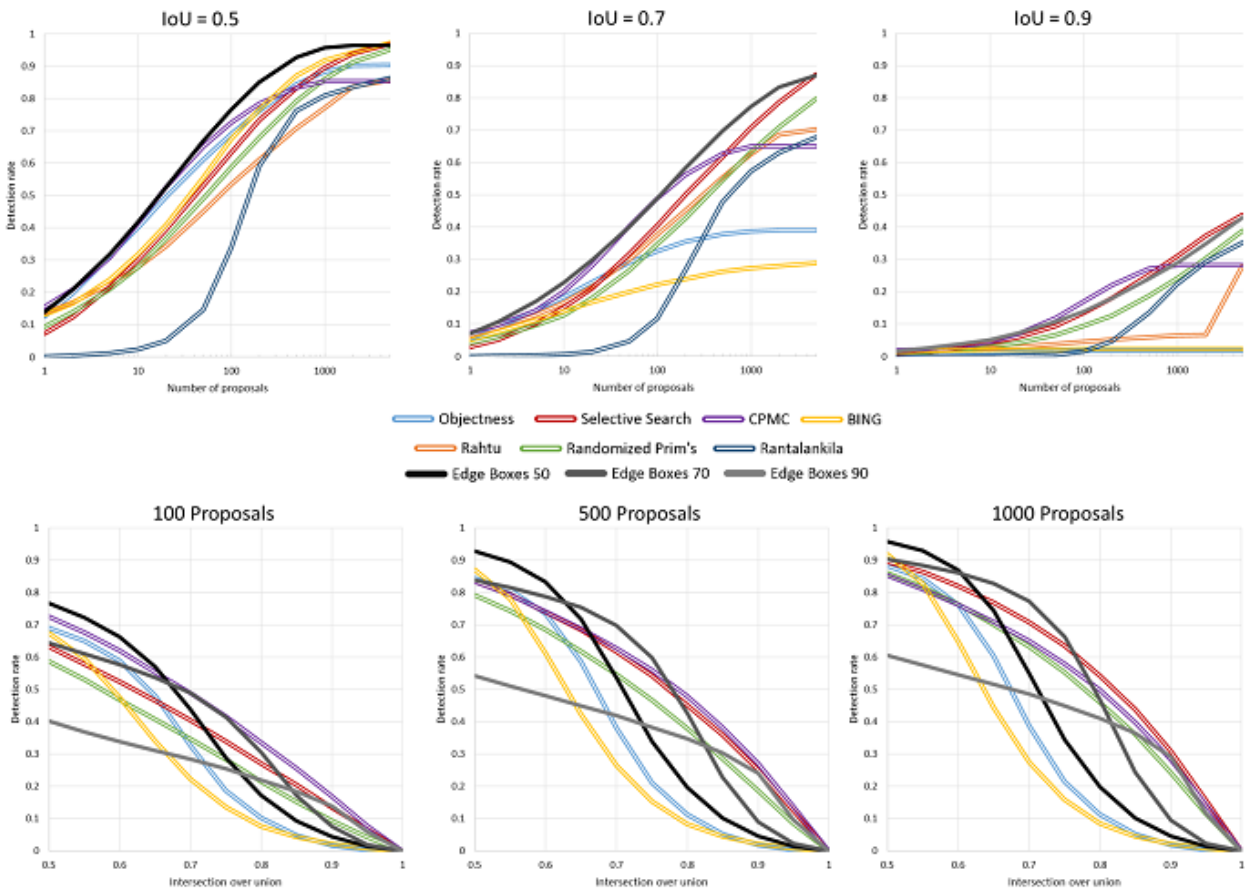


图 5-3 不同算法，包括“候选区域（Objectness）”、选择性探索（Selective Search）[11]、Randomized Prim’s[14]和 Rahtu[13]。本算法的变量用$\delta = 0.5, 0.7$和 0.9 分别代表 Edge Boxes 50、Edge Boxes 70 和 Edge Boxes 90，并进行检测。上三个图表示：已知不同 IoU 阈值，物体候选区域框随物体检测率变化而变化的曲线图。下三个图表示：已知不同候选区域框的数量，观察检测率和 IoU 阈值的关系。

本文提出的算法与其他算法进行了比较，见表 5-2。图 5-3[20]中上面三个图表示：在不同 IoU 阈值的情况下，选用不用边缘特征算法时，物体检测率随候选区域框数量变化而变化的曲线图。无论 IoU 值或候选区域数量选择多少，边缘框的算法都比其他算法的物体检测准确率效果好。选择性搜索（Selective Search）[11]这个算法在物体检测准确率上的效果与本文的边缘框算法的效果差不多，尤其是在较大的 IoU 值和更多候选区域数量的边缘框的情况下。论文[12]中的 CPMC 算法能够生成高质量的候选区域，但是候选区域数量生成较少，因此不能实现很高的召回率。论文[16]中提出的 BING 算法的时间开销少，但是只能生成宽松的候选区域框，因此只在较低的 IoU 的值的情况下选用。反观我们的算法，边缘框对于不同 IoU 阈值以及不同物体候选区域数量，都能实现很好的物体检测准确率。实

际上，在表 5-2 中能观察到，为了实现在 IoU 值为 0.7，召回率为 75%的情况下的物体检测准确率，我们若采用边缘框算法，那么我们需要 800 个候选区域；若采用选择性搜索（Selective Search）算法，那么需要 1400 个候选区域；若采用 Randomized Prim's 算法，那么需要 3000 个候选区域。对于其他算法，若要实现召回率为 75%，那么至少需要 5000 个候选区域。除了选择性搜索（Selective Search），本文提出的边缘框算法比其他所有算法在最大召回率（87%）上和曲线下面积（AUC）上实现了更高的物体检测准确率。

图 5-3 中上三个图表示：已知不同边界框候选区域数量，我们观察物体检测率随 IoU 阈值变化而变化的曲线图。与图 9（a，b）相似的是，如果这些图设置了合适的 IoU 阈值δ参数，那么物体检测率就比较高。对于不同的 IoU 阈值，除了边缘框 70 算法（Edge Boxes 70），没有一个算法或者某一组参数的设置能够始终使物体检测准确率保持很高。对于边缘盒 70 算法，IoU 阈值在 0.5 到 0.8 之间都能够有很高的物体检测准确率。对于需要很高的 IoU 阈值，边缘框 90（Edge Boxes 90）算法的物体检测效果最好。

表 5-2 比较了本文提出的边缘框算法与其他算法在时间开销和其他总结数据的情况。边缘框算法的时间开销（包括计算边缘图）为 0.1 秒/每张图像。表 5-2 表明，本文算法的时间开销更少，且物体检测的精确度也更高。唯一可以与边缘框算法比较精确度的算法是选择性搜索（Selective Search）算法和 CPMC 算法，但是这两个算法的时间开销很大。唯一可以与边缘框算法比较时间开销的算法是 BING 算法，但是 BING 算法在 IoU 值为 0.7 的情况下与其他所有的算法比较时的物体检测准确率最低。

最后，图 5-4 是定性的物体检测结果图。许多误检发生在图像的背景中，这些误检大多是小物体或者完全封闭的物体。

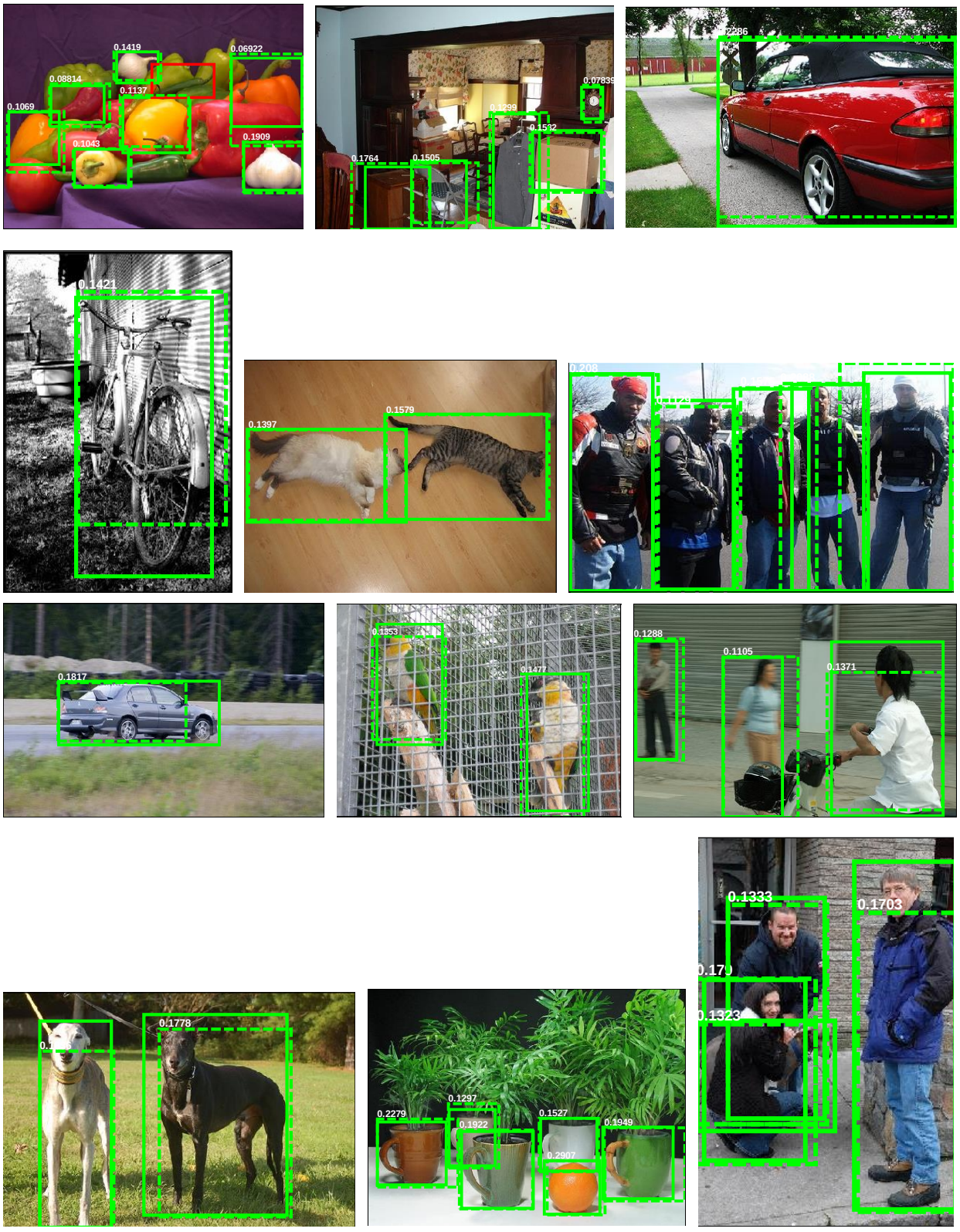


图 5-4 上图是定性的物体检测结果图。绿色实线框是相对于标定框来说的最接近的物体候选区域框。标定框由绿色虚线和红色实线标定，其中绿色虚线表示物体已找到，红色实线表示物体未找到。IoU 阈值为 0.7 时，能够比较恰当得检测出物体的位置。结果展示了

边缘盒 70（Edge Boxes 70）有 1000 个物体候选区域的情况。在这种设置前提下，本文提出的方法能够得到 75%的正确物体位置。

5.4 演示系统

5.4.1 设计平台

选择 MATLAB 作为系统的设计平台。MATLAB 自产生之日起就具有方便的数据可视化功能，以将向量和矩阵用图形表现出来。可用于科学计算和工程绘图。另外，新版本的 MATLAB 还着重在图形用户界面（GUI）的制作上做了很大的改善，使得本系统能够更形象具体地得到展示。

5.4.2 Piotr Dollár 的 ToolBox 安装及数据集

运行代码的过程中需要使用 Piotr Dollár 写的一些已经封装好的函数，因此需要安装这个 toolbox。

安装的过程如下：

第一步：在 http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html 上下载 Version3.30 的 Toolbox；

第二步：安装 Toolbox：>>addpath(genpath('C:\toolbox'));% Matlab 路径

>>savepath;

>>toolboxCompile; %必须进行编译

%compile 时选择数字 2，使用 VS2010 等编译，显示 Done compiling

注：通过"which xxx.m"来检验这个 Toolbox 是否安装成功。

第三步：下载图片数据集：BSR_bsds500.tgz。

5.4.3 主界面与测试界面

打开主界面，如图 5-5。在图像显示的位置预置上海大学图标。

图 5-5 基于快速边缘图的快速物体定位的主界面

点击“打开演示界面”按钮，弹出图像检测的界面，如图 5-6。

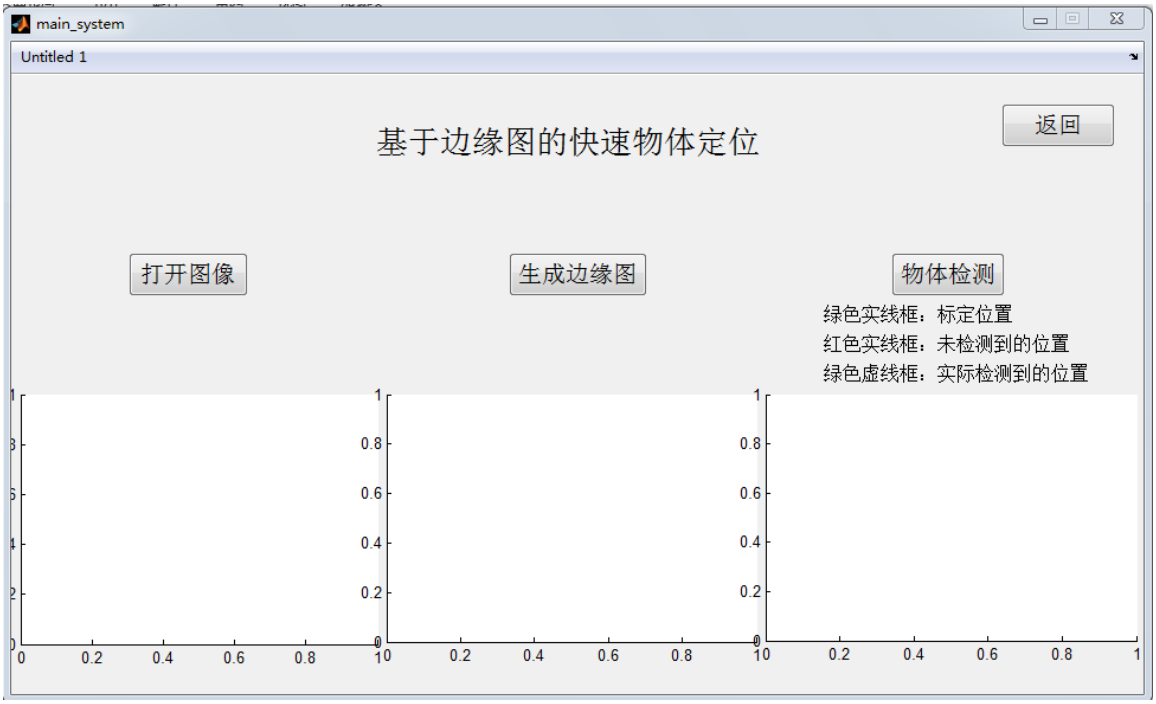


图 5-6 基于快速边缘图的快速物体定位的测试界面

点击“打开图像”按钮，弹出选择图像的菜单。选择任意一张图像之后，在左侧方框中得到显示。

点击“生成边缘图”按钮，能显示对应选择的图像的边缘图像。

点击“物体检测”按钮，能显示对应的物体检测结果（标定位置、未检测位置以及实际

检测到的位置）。

显示结果如图 5-7。图中，检测出来的候选框上方的数字为特征值（评分）。

图 5-7 基于快速边缘图的快速物体定位的结果界面

点击“返回”，返回界面。点击“测试数据集”按钮，弹出测试数据集结果的界面，如图 5-8。

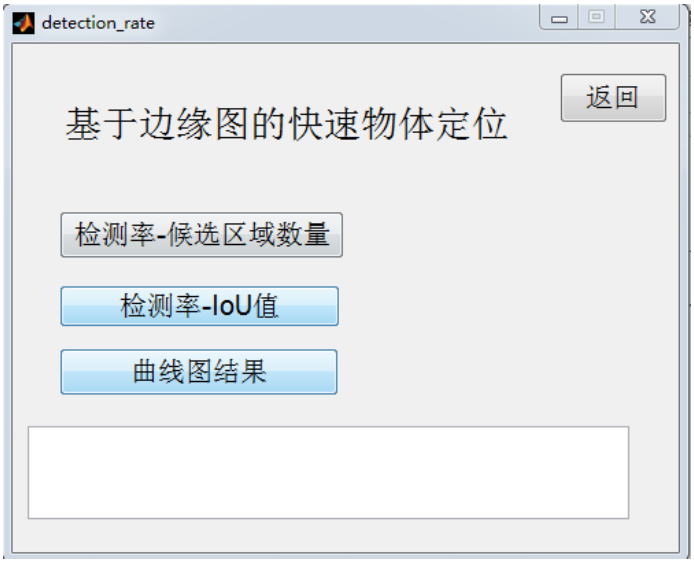


图 5-8 基于快速边缘图的快速物体定位的测试数据集界面

点击“检测率-候选区域数量”、“检测率-IoU 值”、“曲线图结果”，分别显示测试 BSDS500 数据集的结果，见图 5-9 和 5-10。

① 检测率-候选区域数量的输入参数是：nm='EdgeBoxes70';

boxesEval('data',data,'names',nm,'thrs',.7,'show',2);%IoU 阈值：0.7

② 检测率-IoU 值以及曲线图结果的输入参数是：nm='EdgeBoxes70';

boxesEval('data',data,'names',nm,'thrs',.5:.05:1,'cnts',1000,'show',3);

%IoU 阈值从 0.5 至 1 每次变化 0.05；候选区域数量为 1000

③ 数据集运行后结果：EdgeBoxes70 T=0.70 A=0.46 M=823 R=0.86。

其中：T 表示设定的 IoU 阈值；A 表示平均曲线下面积（AUC）；M 表示召回率≥0.75 时最少的候选区域个数；R 表示最大的曲线下面积。

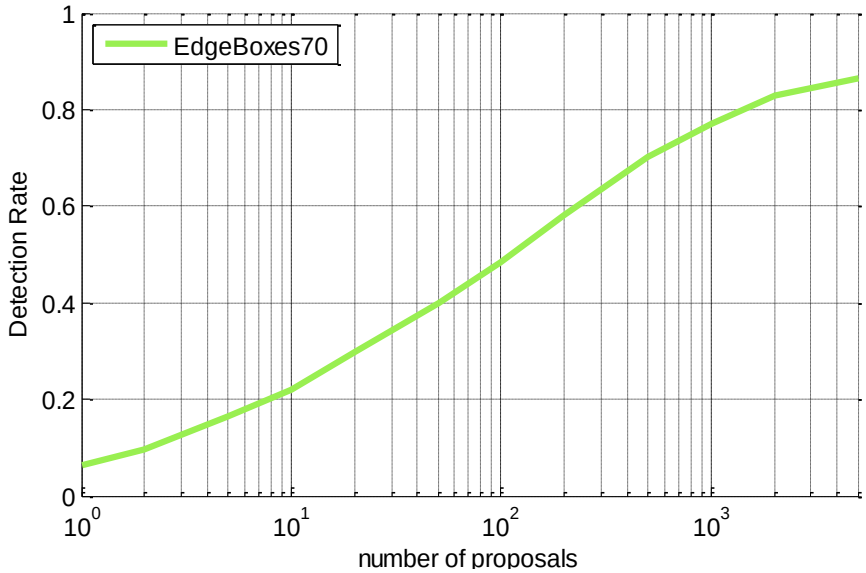


图 5-9 物体检测率与候选区域个数的关系图

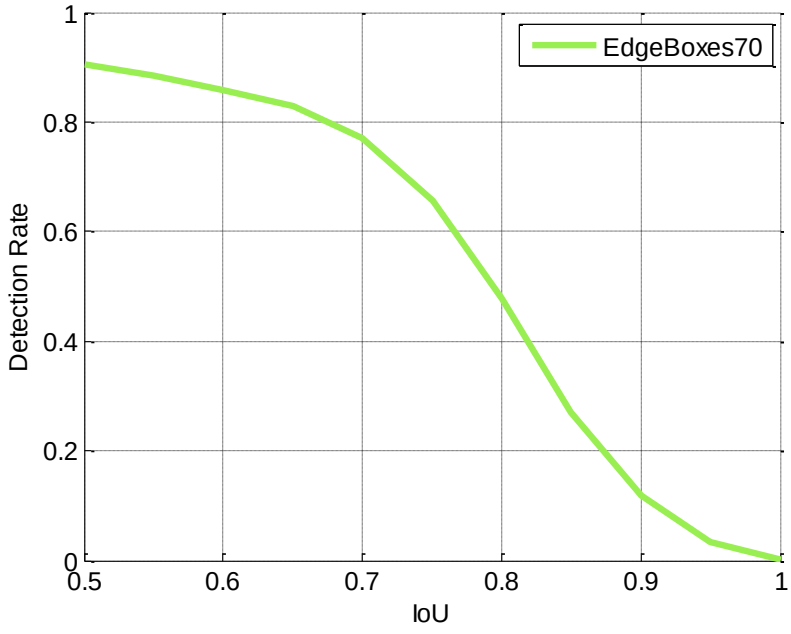


图 5-10 物体检测率与 IoU 值的关系图

5.5 物体定位算法的具体实现

5.5.1 总体结构图

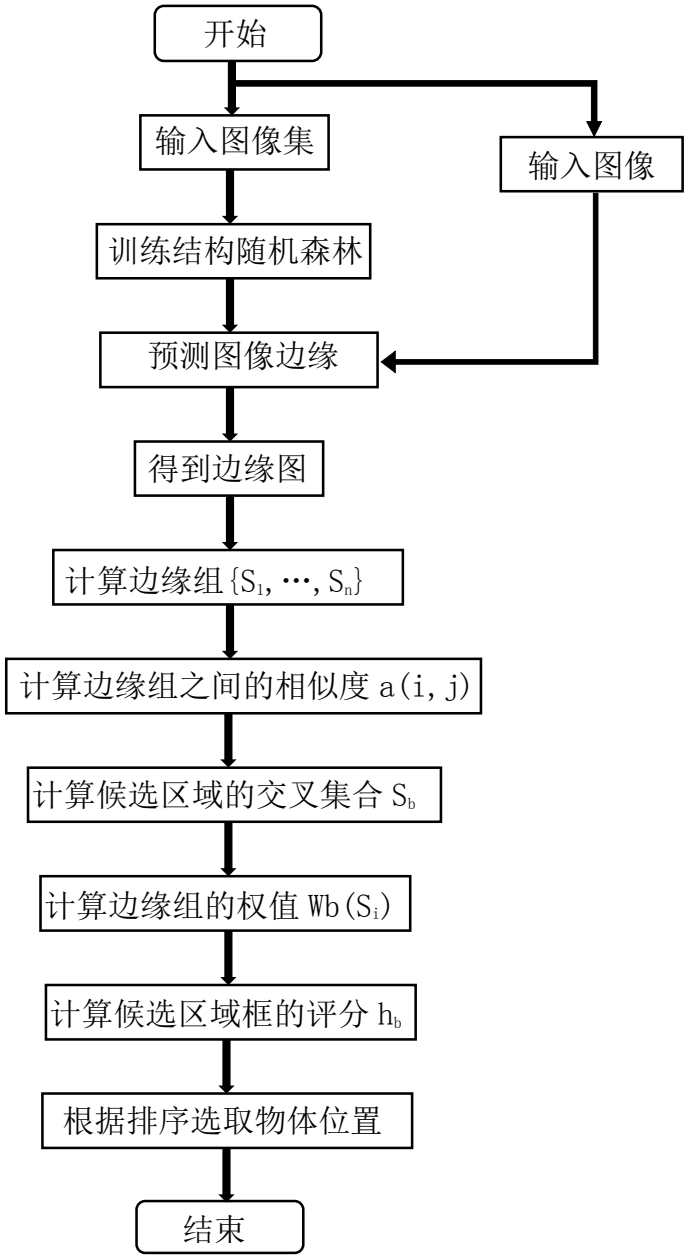


图 5-11 总体流程图

5.5.2 程序设计与代码

（1）读取图像文件

```
[filename,pathname]=uigetfile({'*.png';},'载入图像');
                                                %获得图像文件的文件名和路径
if isequal(filename,0)|isequal(pathname,0)      %没有选中图像
    errordlg('没有选中文件','出错');
    return;
else                                            %选中图像
    file=[pathname,filename];                   %获得图像路径
    I=imread(file);                             %读取图像信息
    axes(handles.axes1);                        %设置图像的显示位置 axes1
    imshow(I);                                  %将图像显示在 axes1 上
    handles.img=I;                              %将图像传递给句柄，以便其他控件使用
    guidata(hObject,handles);                   %更新句柄中的信息
end
```

（2）获取边缘图像

```
%% 设置训练过程的参数
opts=edgesTrain();                  %随机森林的默认参数设置
opts.modelDir='models/';            %设置训练好的森林模型的位置
opts.modelFnm='modelBsds';          %设置随机森林的名称
opts.nPos=5e5; opts.nNeg=5e5;       %正负样本个数（哪些是轮廓，哪些不是）
opts.useParfor=0;                   %训练决策树是否要并行计算

%% 训练边缘检测子
model=edgesTrain(opts);             %如果随机森林已经训练好，那么直接使用

%% 设置边缘检测参数
model.opts.multiscale=0;            %若需高精度，那么设置 multiscale 为 1
```

```
model.opts.sharpen=2;                %若需运算速度最快，那么设置 sharpen 为 0
model.opts.nTreesEval=4;             %若需运算速度最快，那么设置 nTreesEval 为 1
model.opts.nThreads=4;               %随机森林评价的阈值个数
model.opts.nms=0;                    %若设置为 1，则增加 NMS 运算

%% 检测图像的边缘
E=edgesDetect(handles.img,model);    %检测图像的边缘
axes(handles.axes2);                 %设置图像的显示位置 axes2
imshow(1-E);                         %显示图像的边缘
guidata(hObject,handles);            %更新句柄中的信息
```

（3）检测图像中的物体位置

```
%% 下载实现训练好的边缘检测子模型，并设置参数
model=load('models/forest/modelBsds'); model=model.model;
model.opts.multiscale=0; model.opts.sharpen=2; model.opts.nThreads=4;

%% 设置目标检测的参数
opts = edgeBoxes;                    %设置目标检测的默认参数
opts.alpha = .65;                    %划窗搜索的步径长度参数
opts.beta  = .75;                    %目标检测轮廓的 NMS 阈值参数
opts.minScore = .01;                 %候选区域特征值最小能检测到的值
opts.maxBoxes = 1e4;                 %划窗检索过程中能检测候选区域最大个数

%% 检测候选区域
bbs=edgeBoxes(handles.img,model,opts);
                %bbs 为获取到的候选区域个数，以及对应到特征值

%% 展示检测的结果（对于 Peppers 图像）
gt=[122 248 92 65; 193 82 71 53; 410 237 101 81; 204 160 114 95; ...
  9 185 86 90; 389 93 120 117; 253 103 107 57; 81 140 91 63];
```

```
                                        %Peppers 图像中对应的 8 个真实值（实际位置）
    gt(:,5)=0; [gtRes,dtRes]=bbGt('evalRes',gt,double(bbs),.7);
    % 获取所有在 IoU 为 0.7 范围内的标定物体位置信息（gtRes）和对应的候选区域信息
（dtRes）
    axes(handles.axes3);                    %设置图像的显示位置 axes2
    bbGt('showRes',handles.img,gtRes,dtRes(dtRes(:,6)==1,:));
    % 显示结果
    % dtRes(dtRes(:,6)==1,:)表示选择满足条件的候选区域
    guidata(hObject,handles);               %更新句柄中的信息
```

5.6 本章小结

本章节介绍了本算法的实验结论及程序实现。从实验结果中，可以看出“边缘框”与其他算法相比，具有很高的物体检测率。同时，本章还介绍了 Piotr Dollár 的工具箱以及程序的运行过程、演示系统、总体结构图、代码的实现等。

第六章 总结

6.1 本文主要内容

由于巨大的图像搜索空间，很多计算机视觉任务很难达到实时的目的。本文提出的物体对象性（objectness）可以有效的减少图像的搜索空间，不仅具有很高的物体检测率，而且也可以达到实时检测的速度。

为了加深论文[20]的印象，我同时还阅读了[21, 22, 23]，对物体候选区域的理解更为深刻。很多提取物体候选区域的文献，例如 BING 等，都是基于学习理论的方法。此外，本文的许多内容，甚至数学公式，都是基于作者的直觉直接建立的，而不经过学习。

“候选区域”（proposal）这个概念非常重要。当我们看到一幅图像的时候，我们绝对不会像传统检测算法那样，从图像左上角开始扫描图像，而是一眼“纵观全局”，直接发现目标“大概的位置”，然后进一步细看。候选区域正是由人类这一特质启发而提出的。显而易见，这样的方式，速度会很快。但是，如果候选区域提取的不准，那么检测率就不会高了。

为了提高物体检测定位系统的实时性，本文主要提出了“边缘框”的算法，即直接从边缘产生物体的候选区域框。边缘聚类形成轮廓，根据完全位于候选区域框内的轮廓数量来决定候选区域的可能性。

6.2 展望及应用前景

在本篇文章的基础上，还是可以继续进行探索。由于本文仅在获取边缘图像中使用了学习算法，而物体检测的过程中没有采用学习算法，也因此没有训练过程，在检测多个物体或者有物体重叠时，仍然有误检或漏检的概率，因此本算法仍然有待完善。

本算法大大提高了目标检测的准确率和速度，在大数据的时代下，可以应用到多方多面：互联网在线目标检测、数据库中寻找对应物体等，在教育、司法、游戏等多个领域中可实现智能化。

英文文献

BING: Binarized Normed Gradients for Objectness Estimation at 300fps

Ming-Ming Cheng1 Ziming Zhang2 Wen-Yan Lin3 Philip Torr1

1The University of Oxford 2Boston University 3Brookes Vision Group

Abstract

Training a generic objectness measure to produce a small set of candidate object windows, has been shown to speed up the classical sliding window object detection paradigm. We observe that generic objects with well-defined closed boundary can be discriminated by looking at the norm of gradients, with a suitable resizing of their corresponding image windows in to a small fixed size. Based on this observation and computational reasons, we propose to resize the window to 8×8 and use the norm of the gradients as a simple 64D feature to describe it, for explicitly training a generic objectness measure.

We further show how the binarized version of this feature, namely binarized normed gradients (BING), can be used for efficient objectness estimation, which requires only a few atomic operations (e.g. ADD, BITWISE SHIFT, etc.). Experiments on the challenging PASCAL VOC 2007 dataset show that our method efficiently (300fps on a single laptop CPU) generates a small set of category-independent, high quality object windows, yielding 96.2% object detection rate (DR) with 1,000 proposals. Increasing the numbers of proposals and color spaces for computing BING features, our performance can be further improved to 99.5% DR.

1. Introduction

As one of the most important areas in computer vision, object detection has made great strides in recent years. However, most state-of-the-art detectors still require each *category specific* classifiers to evaluate many image windows in a sliding window fashion [17, 25]. In order to reduce the number of windows each classifier needs to consider, training an objectness measure which is *generic over categories* has recently becomes popular [2, 3, 21, 22, 48, 49, 57]. *Objectness*

is usually represented as a value which reflects how likely an image window covers an object of *any category* [3]. A generic objectness measure has great potential to be used in a pre-filtering process to significantly improve: i) computational efficiency by reducing the search space, and ii) detection accuracy by allowing the usage of strong classifiers during testing. However, designing a good generic objectness measure method is difficult, which should:

• achieve **high object detection rate** (DR), as any undetected objects at this stage cannot be recovered later;

• produce **a small number of proposals** for reducing computational time of subsequent detectors;

• obtain **high computational efficiency** so that the method can be easily involved in various applications, especially for realtime and large-scale applications;

• have **good generalization ability** to unseen object categories, so that the proposals can be reused by many category specific detectors to greatly reduce the computation for each of them.

To the best of our knowledge, no prior method can satisfy all these ambitious goals simultaneously.

Research from cognitive psychology [47, 54] and neurobiology [20, 38] suggest that humans have a strong ability to perceive objects before identifying them. Based on the human reaction time that is observed and the biological signal transmission time that is estimated, human attention theories hypothesize that the human vision system processes only parts of an image in detail, while leaving others nearly unprocessed. This further suggests that before identifying objects, there are simple mechanisms in the human vision system to select possible object locations.

In this paper, we propose a surprisingly simple and powerful feature “BING” to help the search for objects using objectness scores. Our work is motivated by the fact that objects are stand-alone things with well-defined closed boundaries and centers [3, 26, 32]. We observe that generic objects with well-defined closed boundaries share surprisingly strong correlation when looking at the norm of the gradient (see Fig. 1 and Sec. 3), after resizing of their corresponding image windows to small fixed size (*e.g.* 8×8). Therefore, in order to efficiently quantify the objectness of an image window, we resize it to 8×8 and use the norm of the gradients as a simple 64D feature for learning a generic objectness measure in a cascaded SVM framework. We further show how the binarized version of the NG feature, namely binarized normed gradients (**BING**) feature, can be used for

efficient objectness estimation of image windows, which requires only a few atomic CPU operations (*i.e*. ADD, BITWISE SHIFT, etc.). The BING feature's simplicity, contrast with recent state of the art techniques [3,22,48] which seek increasingly sophisticated features to obtain greater discrimination, while using advanced speed up techniques to make the computational time tractable.

We have extensively evaluated our method on the PASCAL VOC2007 dataset [23]. The experimental results show that our method efficiently (300fps on a single laptop CPU) generates a small set of data-driven, category-independent, high quality object windows, yielding 96.2% detection rate (DR) with 1,000 windows (≈ 0.2% of full sliding windows). Increasing the number of object windows to 5,000, and estimating objectness in 3 different color spaces, our method can achieve 99.5% DR. Following [3, 22, 48], we also verify the generalization ability of our method. When training our objectness measure on 6 object categories and testing on other 14 *unseen* categories, we observed similar high performance as in standard settings (see Fig. 3). Compared to most popular alternatives [3,22,48], the BING features allow us to achieves better DR using a smaller set of proposals, is much simpler and 1000+ times faster, while being able to predict *unseen* categories. This fulfills afore mentioned requirements of a good objectness detector. Our source code will be published with the paper.

3. Methodology

Inspired by the ability of human vision system which efficiently perceives objects before identifying them [20, 38, 47, 54], we introduce a simple 64D norm of the gradients (NG) feature (Sec. 3.1), as well as its binary approximation, *i.e*. binarized normed gradients (BING) feature (Sec. 3.3), for efficiently capturing the objectness of an image window.

To find generic objects within an image, we scan over a predefined *quantized window sizes* (scales and aspect ratios1). Each window is scored with a linear model $w \in R^{64}$(Sec. 3.2),

$$s_l =< w, g_l >, \tag{1}$$

$$l = (i, x, y), \tag{2}$$

where s_l, g_l, l, i and (x, y) are filter score, NG feature, location, size and position of a window respectively. Using non-maximal suppression (NMS), we select a small set of proposals from each size i. Some sizes (*e.g*. 10 × 500) are less likely than others to contain

an object instance (*e.g.* 100 × 100). Thus we define the objectness score (*i.e.* calibrated filter score) as

$$o_l = v_i \cdot s_l + t_i, \tag{3}$$

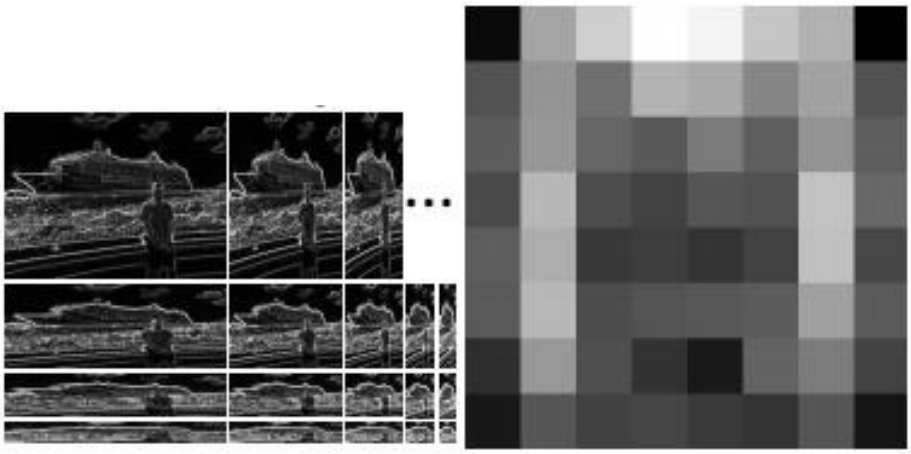
where v_i, $t_i \in \mathrm{R}$ are sperately learnt coefficient and a bias terms for each quantised size i (Sec. 3.2). Note that calibration using (3), although very fast, is only required when re-ranking the small set of final proposals.

3.1. Normed gradients (NG) and objectness

Objects are stand-alone things with well-defined closed boundaries and centers [3, 26, 32]. When resizing windows corresponding to real world objects to a small fixed size (*e.g.* 8 × 8, chosen for computational reasons that will be explained in Sec. 3.3), the norm (*i.e.* magnitude) of the corresponding image gradients becomes a good discriminative feature, because of the little variation that closed boundaries could present in such abstracted view. As demonstrated in Fig. 1, although the cruise ship and the person have huge difference in terms of color, shape, texture, illumination *etc*., they do share clear correlation in normed gradient space. To utilize this observation for efficiently predicting the existence of object instances, we firstly resize the input image to different *quantized sizes* and calculate the normed gradients of each resized image. The values in an 8 × 8 region of these resized normed gradients maps are defined as a 64D ***normed gradients (NG)*** feature of its corresponding window.

Our NG feature, as a dense and compact objectness feature for an image window, has several advantages. Firstly, no matter how an object changes its position, scale and aspect ratio, its corresponding NG feature will remain roughly unchanged because of the normalized support region of this feature. In other words, NG features are insensitive to change of translation, scale and aspect ratio, which will be very useful for detecting objects of arbitrary categories.

(a) source image　　(c) 8 × 8 NG features



(b) normed gradients maps (d) learned model $\mathbf{w} \in R^{8\times8}$

Figure 1. Although object (red) and non-object (green) windows present huge variation in the image space (a), in proper scales and aspect ratios where they correspond to a small fixed size (b), their corresponding normed gradients, i.e. a NG feature (c), share strong correlation. We learn a single 64D linear model (d) for selecting object proposals based on their NG features.

3.2. Learning objectness measurement with NG

To learn an objectness measure of image windows, we follow the general idea of the two stages cascaded SVM [57].

Stage I. We learn a single model **w** for (1) using linear SVM [24]. NG features of the ground truth object windows and random sampled background windows are used as positive and negative training samples respectively.

Stage II. To learn v_i and t_i in (3) using a linear SVM [24], we evaluate (1) at size *i* for training images and use the selected (NMS) proposals as training samples, their filter scores as 1D features, and check their labeling using training image annotations (see Sec. 4 for evaluation criteria).

Discussion. As illustrated in Fig. 1d, the learned linear model **w** (see Sec. 4 for experimental settings), looks similar to the multi-size center-surrounded patterns [36] hypothesized as biologically plausible architecture of primates [27, 38, 54]. The large weights along the borders of **w** favor a boundary that separate an object (center) from its background (surrounded). Compared

to manually designed center surround patterns [36], our learned **w** captures a more sophisticated, natural prior. For example, lower object regions are more often occluded than upper parts. This is represented by **w** placing less confidence in the lower regions.

Algorithm 1 Binary approximate model **w** [28].

Input: **w**, N_w

Output: $\{\beta_j\}_{j=1}^{N_w}$, $\{a_j\}_{j=1}^{N_w}$

Initialize residual: ε = **w**

for j = 1 to N_w **do**

a_j =sign(ε)

$\beta_j =< a_j, \varepsilon > / ||a_j||^2$ (project ε onto a_j)

$\varepsilon \leftarrow \varepsilon - \beta_j a_j$ (update residual)

end for

3.3. Binarized normed gradients (BING)

To make use of recent advantages in model binary approximation [28, 59], we propose an accelerated version of NG feature, namely binarized normed gradients (BING), to speed up the feature extraction and testing process. Our learned linear model $\mathbf{w} \in R^{64}$ can be approximated with a set of basis vectors $\mathbf{w} \approx \sum_{j=1}^{N_w} \beta_j a_j$ using Alg. 1, where *Nw* denotes the number of basis vectors, $a_j \in \{-1, 1\}^{64}$ denotes a basis vector, and $\beta_j \in R$ denotes the corresponding coefficient. By further representing each a_j using a binary vector and its complement: $a_j = a_j^+ - \overline{a_j^+}$, where $a_j^+ \in \{0, 1\}^{64}$, a binarized feature b could be tested using fast BITWISE AND and BIT COUNT operations (see [28]),

$$< \mathbf{w}, b > \approx \sum_{j=1}^{N_w} \beta_j \left(2 < a_j^+, b > - |b|\right). \quad (4)$$

Algorithm 2 Get BING features for $W \times H$ positions.

Comments: see Fig. 2 for illustration of variables

Input: binary normed gradient map $b_{W \times H}$

Output: BING feature matrix $\mathbf{b}_{W\times H}$

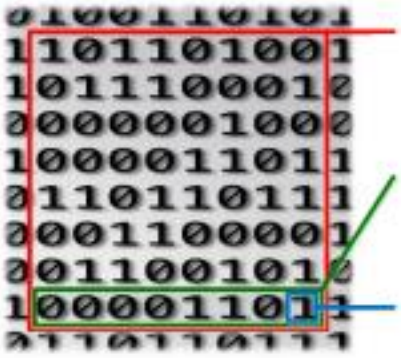
Initialize: $\mathbf{b}_{W\times H} = 0$, $\mathbf{r}_{W\times H} = 0$

for each position (x, y) in scan-line order **do**

$\mathbf{r}_{x,y} = (\mathbf{r}_{x-1,y} \ll 1) \mid b_{x,y}$

$\mathbf{b}_{x,y} = (\mathbf{b}_{x,y-1} \ll 8) \mid \mathbf{r}_{x,y}$

end for



$\mathbf{b}_{k,i,x,y} \in \{0, 1\}^{8\times 8}$

shorthand: $\mathbf{b}_{x,y}$ or $\mathbf{b}_{k,l}$

$\mathbf{r}_{k,i,x,y} \in \{0, 1\}^{8}$

shorthand: $\mathbf{r}_{x,y}$ or $\mathbf{r}_{k,l}$

$b_{k,i,x,y} \in \{0, 1\}$

shorthand: $b_{x,y}$

Figure 2. Illustration of variables: a BING feature $\mathbf{b}_{x,y}$, its last row $\mathbf{r}_{x,y}$ and last element $b_{x,y}$. Notice that the subscripts *i*, *x*, *y*, *l*, *k*, introduced in (2) and (5), are locations of the whole vector rather than index of vector element. We can use a single atomic variable (INT64 and BYTE) to represent a BING feature and its last row, enabling efficient feature computation (Alg. 2).

The key challenge is how to binarize and calculate our NG features efficiently. We approximate the normed gradient values (each saved as a BYTE value) using the top *Ng* binary bits of the BYTE values. Thus, a 64D NG feature $\mathbf{g}_l$ can be approximated by *Ng* ***binarized normed gradients (BING)*** features as

$$\mathbf{g}_l = \sum_{k=1}^{N_g} 2^{8-k}\mathbf{b}_{k,l} \tag{5}$$

Notice that these BING features have different weights according to its corresponding bit position in BYTE values.

Naively getting an 8 × 8 BING feature requires a loop computing access to 64 positions. By exploring two special characteristics of an 8 × 8 BING feature, we develop a fast BING feature calculation algorithm (Alg. 2), which enables using atomic updates (BITWISE SHIFT and

BITWISE OR) to avoid the loop computing. First, a BING feature $\mathbf{b}_{x,y}$ and its last row $\mathbf{r}_{x,y}$ could be saved in a single INT64 and a BYTE variables, respectively. Second, adjacent BING features and their rows have a simple cumulative relation. As shown in Fig. 2 and Alg. 2, the operator BITWISE SHIFT shifts $\mathbf{r}_{x-1,y}$ by one bit, automatically through the bit which does not belong to $\mathbf{r}_{x,y}$, and makes room to insert the new bit $b_{x,y}$ using the BITWISE OR operator. Similarly BITWISE SHIFT shifts $\mathbf{b}_{x,y-1}$ by 8 bits automatically through the bits which do not belong to $\mathbf{b}_{x,y}$, and makes room to insert $\mathbf{r}_{x,y}$.

Our efficient BING feature calculation shares the *cumulative* nature with the integral image representation [52]. Instead of calculating a single scalar value over an arbitrary rectangle range [52], our method uses a few atomic operations (*e.g.*ADD, BITWISE, *etc*.) to calculate *a set of binary patterns* over an 8 × 8 fixed range.

The filter score (1) of an image window corresponding to BING features $\mathbf{b}_{k,l}$ can be efficiently tested using:

$$s_l \approx \sum_{j=1}^{N_w} \beta_j \sum_{k=1}^{N_g} C_{j,k}, \tag{6}$$

where $C_{j,k} = 2^{8-k}(2 < a_j^+, \mathbf{b}_{k,l} > - |\mathbf{b}_{k,l}|)$ can be tested using fast BITWISE and POPCNT SSE operators.

Implementation details. We use the 1-D mask [−1, 0, 1] to find image gradients g_x and g_y in horizontal and vertical directions, while calculating normed gradients using $\min(|g_x| + |g_y|, 255)$ and saving them in BYTE values. By default, we calculate gradients in RGB color space. In our C++ implementation, POPCNT SSE instructions and OPENMP options are enabled.

中文翻译

二值化归一化梯度快速检测物体候选区域（BING，检测率为 300 张每秒）

摘要

训练一般的物体来产生一组物体候选区域的方式，可以加快常用的滑动窗口的检测速度。我们可以观察到，一般的物体具有封闭的边界，通过将这个区域调整到某一个固定的尺寸，便可以通过观察归一化后的梯度与其他环境区分开来。基于这种肉眼可以观察到的区别，以及为了提高计算速度，我们将这个候选区域的大小调整到固定的 8×8，而且使用归一化后的梯度值描述为 64 维特征来表示，即训练一个一般的物体候选区域的特征。

我们接下来展示二值化的特征（BING）如何能够通过小运算（加法、比特位位移等）来提高物体的检测速率。我们在 PASCAL VOC 2007 数据集上进行了实验，展示了本算法的高效性。本算法产生了一个不依赖于数据集类别的高质量物体候选区域，检测的 1000 个候选区域中有 96.2%的物体检测率（DR）。如果提高候选区域的数量，以及增加 BING 在颜色空间的计算，本算法可以达到 99.5%的检测率。

1. 介绍

目标检测是计算机视觉上一个重要的领域，近年来有很多进展。然而，最先进的检测器仍然需要特定类别的分类器来评价许多图像的窗口。为了减少滑动窗口的数量，训练一个适用于任意类别的一般的物体候选区域的评价方式变得越来越受欢迎。物体候选区域的值的大小表示这个区域存在任意类别的物体的可能性的大小。一个一般的物体检测方式具有很多潜力，它可以改善：1）通过减少滑动窗口的搜索提高计算效率；2）通过使用强分类器来提高检测效率。然而，设计一个好的一半物体检测方式比较困难，它应具有以下特质：

•实现很好的物体检测率（DR）；

•生成较少的物体候选区域来减少检测子的使用；

•获得很高的计算效率，这样的话，这个算法就可以应用到其他领域，尤其是对于实时和大数据计算；

•能够检测未知类别物体的能力，这样就可以重复使用这个检测子而不需要重新训练新的检测子。

我们知道之前没有方法可以同时满足以上的目标。

在本文中，我们提出来了一个简单却效果很好的特征 BING，使用候选区域的得分来寻找物体。物体的定义是：物体独立于环境，具有封闭的边界。我们根据这个思路，观察调整尺寸到 8×8 大小的一般物体的归一化的梯度，可以发现这些物体的边界具有关联性（见图 1 和第 3 节）。因此，为了有效地量化图像中物体的候选区域，我们将候选区域调整到 8×8 大小，并且使用归一化后的梯度用作 64 维特征，通过一个级联的 SVM 分类器来训练一个一般的物体候选区域。我们还展示了通过小运算（加法、比特位位移等），这个二值化的归一化后的梯度特征（BING）能用于提高检测窗口的效率。与其他很多为了提高区分能力而具有很高复杂性的特征相比，BING 的特征很简单，而且计算速度也快。

我们在 PASCAL VOC 2007 的数据集上评价了本算法。实验结果显示了本算法的高效性（单核 CPU 下检测速度为 300 张每秒），同时生成了一个小数据集驱使的，不依赖类别的，高质量的检测窗口，以及在 1000 个候选区域框下具有 96.2%的检测率（其中，约 0.2%的全屏大小的检测窗口）。如果提高候选区域的数量至 5000，以及处理彩色图像（3 个通道）时，本算法可以达到 99.5%的检测率。本算法还具有不依赖于数据集类别特点。当我们训练 6 类图像的物体位置，去检测其他未知的 14 类图像时，检测率与检测同一类物体时的效果差不多。与其他的算法相比较，BING 特征使用少量的候选区域实现了更高的检测率，同时，检测子既简单又比其他算法快至少 1000 倍，而且不依赖于训练图像集的类别。因此，本算法满足了先前提到的要求。

2. 相关工作

3. 方法

由于人的视觉有一种特性，看图像时在辨认物体前会先察觉到这个物体，因此我们提出了一个简单的 64 维归一化后的梯度特征，以及其二值化后的特征，即 BING 特征，来加快图像窗口获取物体候选区域的速度。为了找到图像中的物体，我们使用了预定义好的量化窗口（已知长宽和长宽比）。每一个窗口用一个线性模型$\mathrm{w} \in R^{64}$来得到一个值（见 3.2 节）：

$$s_l =< w, g_l >, \tag{1}$$

$$l = (i, x, y), \tag{2}$$

其中，s_l是通过滤波器后的值，g_l是归一化后梯度的特征 NG，l是候选区域的所有特征，i是候选区域的大小，(x, y)是候选区域的位置。使用非最大值抑制 NMS，我们可以从同一个大小的一些类似的候选区域 i 中选择一个。有些大小的候选区域（比如 10×500）不太可

能包含物体，而某些候选区域（100×100）可能包含物体，因此我们定义了一个候选区域分值函数（即标准的滤波函数）：

$$o_l = v_i \cdot s_l + t_i, \tag{3}$$

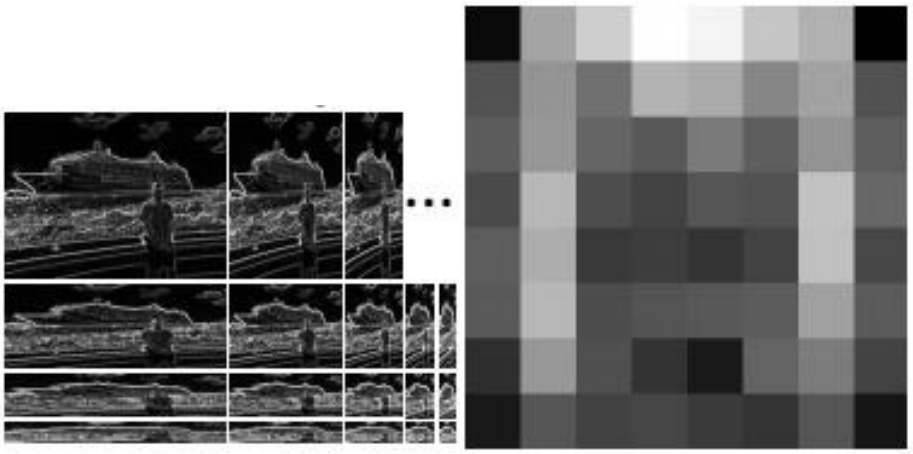
其中，对每一个量化尺寸i 来说，v_i是学习参数，t_i是偏差参数。注意：公式(3)的计算速度很快，它只对重新排列最后选取的一组候选区域计算分值。

3.1 归一化梯度（NG）和物体候选区域

物体独立于环境，具有封闭的边界。当调整窗口的大小为固定的大小时（比如 8×8 大小，用于后续计算的方便，见 3.3 节），对应图像的梯度归一化值（即梯度大小）成为了具有区分能力的特征，因为封闭区域内的特征与外界特征不同。见图 1，虽然“游轮”与“人”有不同的颜色、形状、纹理和亮度，但是它们的归一化特征有一些共同点。将这一点特征用于有效检测物体的存在，我们首先将输入图像的大小调整到不同量化的尺寸，然后计算每个调整后图像的归一化梯度值。这些 8×8 大小的区域的值就被定义为对应窗口的 64 维的归一化梯度（NG）特征。

归一化梯度（NG）特征，是图像窗口的一个密集的候选区域特征，具有很多优点。首先，无论物体如何改变位置、特征尺寸和长宽比，它对应的 NG 特征基本保持不变，因为这个特征经过归一化。换句话说，NG 特征对平移、尺度和长宽比不敏感，这对检测任意类别的物体非常有效。这些不敏感的属性就应该是一个好的物体轮廓生成器所具有的特性。其次，NG 特征的密集特性使它在计算和验证上具有很高的效率，因此 NG 特征可以用于实时物体检测。

(a) 原图像　　(c) 8×8 大小的归一化梯度



(b) 归一化梯度映射　　(d) 学习模型 $\mathbf{w}\in R^{8\times8}$

图 1 虽然物体（红色框）和非物体（绿色框）的窗口在原图像中差别很大（a），但是通过长宽比的改变，它们可以映射到同样的大小（b），它们对应的归一化梯度，即 NG 特征（c），拥有非常相似的特征。我们基于候选区域的 NG 特征，学习一个单一的 64 维的线性模型（d）用来选择物体的候选区域。

3.2. 用 NG 特征来学习候选区域的评价方式

为了学习候选区域的评价方式，我们采用以下两级级联的支持向量机（SVM）。

阶段 1：我们使用线性 SVM 来学习单一模型的 **w**。标定好的真实数据的每个窗口对应的 NG 特征和任意对背景采样窗口对应的 NG 特征分别用作训练的正样本和负样本。

阶段 2：为了使用一个线性 SVM 来学习公式（3）中的v_i和t_i，我们首先需对训练图像在尺寸i下评价公式（1）：使用选择好的候选区域（经过非最大值抑制 NMS）作为训练图像。这些候选区域对应的滤波值作为 1 维特征，并且对使用训练图像来标注的分类进行检查（见第 4 节的评价标准）。

讨论：如图 1d 所示，已学习好的线性模型 **w** 与多尺度围绕中心点的模式相似，这个模式是根据生物学上物种结构合理地假设出来的。**w** 边界的权重值更大，因为它将物体与背景区分了开来。与人工标定的模型不同，我们学习的 **w** 更精细，并且具有自然的先验性。举个例子，图像中物体的位置越处于下方，那么物体被挡住的可能性也越大。用 **w** 表示，即在越下方的区域，物体存在的可能性可以更低点。

算法 1 二值近似模型 w。

输入： **w**, N_w（基的个数）

输出： $\{\beta_j\}_{j=1}^{N_w}$, $\{\mathrm{a}_j\}_{j=1}^{N_w}$

初始化偏差值： $\varepsilon = \mathbf{w}$

for j = 1 to N_w **do**

a_j =sign(ε) （基向量）

$\beta_j = < a_j, \varepsilon > / ||a_j||^2$ （将ε映射到a_j上）

$\varepsilon \leftarrow \varepsilon - \beta_j a_j$ （更新偏差值）

end for

3.3. 二值化归一梯度（BING）

为了利用最近提出的模型二值化近似的优势，我们提出了一个加速版本的 NG 特征，即 BING，用来加速特征提取和测试过程。我们学习好的线性模型 $\mathbf{w} \in R^{64}$ 可以通过算法 1 中的基向量 $\mathbf{w} \approx \sum_{j=1}^{N_w} \beta_j \mathbf{a_j}$近似得到。其中，$N_w$表示基向量的个数，$a_j \in \{-1, 1\}^{64}$ 表示一组基向量，$\beta_j \in R$ 表示相对应的系数。a_j还可以通过以下公式：$a_j = a_j^+ - \overline{a_j^+}$来表示，其中$a_j^+ \in \{0, 1\}^{64}$。这样一来，一个二值化特征可以仅通过位运算的求和以及相乘运算使用论文[28]中得到的式子：

$$< \mathbf{w}, b > \approx \sum_{j=1}^{N_w} \beta_j \left(2 < a_j^+, b > - |b|\right). \quad (4)$$

算法 2 从长宽分别为 W 和 H 的窗口获取 BING 特征。

注释： 图 2 介绍变量

输入： 二值化的归一梯度图 $b_{W \times H}$

输出： BING 特征矩阵 $\mathbf{b}_{W \times H}$

初始化： $\mathbf{b}_{W \times H} = 0$, $\mathbf{r}_{W \times H} = 0$

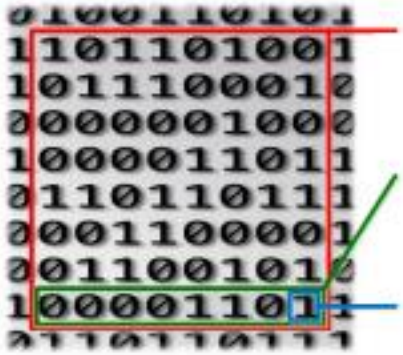
对每个位置(x, y)一行一行浏览：**do**

$\mathbf{r}_{x,y} = (\mathbf{r}_{x-1,y} \ll 1) \mid b_{x,y}$（移动 $\mathbf{r}_{x-1,y}$ 来插入 $b_{x,y}$）

$\mathbf{b}_{x,y} = (\mathbf{b}_{x,y-1} \ll 8) \mid \mathbf{r}_{x,y}$（移动 $\mathbf{b}_{x,y-1} \ll 8$ 来插入 $\mathbf{r}_{x,y}$）

（对每一“页”8×8 大小的二值数据每 8 个临时放置到 INT64 中）

end for



$\mathbf{b}_{k,i,x,y} \in \{0, 1\}^{8\times8}$

简记：$\mathbf{b}_{x,y}$ or $\mathbf{b}_{k,l}$

$\mathbf{r}_{k,i,x,y} \in \{0, 1\}^{8}$

简记：$\mathbf{r}_{x,y}$ or $\mathbf{r}_{k,l}$

$b_{k,i,x,y} \in \{0, 1\}$

简记：$b_{x,y}$

图 2 变量的介绍：一个 BING 特征 $\mathbf{b}_{x,y}$，其对应的最后一行 $\mathbf{r}_{x,y}$ 以及最后一个元素 $b_{x,y}$。

其中，公式（2）和（5）中的下标 i, x, y, l, k 是这个 BING 特征的位置，而不是该向量元素的下标。我们使用单一的变量（INT64 和字节）来表示一个 BING 特征及其最后一行，以提高计算效率（算法 2）。

这个算法最重要的是如何进行二值化以提高 NG 特征的计算效率。我们将归一化梯度值进行近似后得到一系列 8 比特的二值数值。因此，一个 64 维的 NG 特征$\mathbf{g}_l$可以近似表示为以下 BING 公式：

$$\mathbf{g}_l = \sum_{k=1}^{N_g} 2^{8-k}\mathbf{b}_{k,l} \tag{5}$$

注意，这些 BING 特征在不同二值化后的位置i上有不同的权重。

获得一个 8×8 大小的 BING 特征需要循环地计算 64 个位置的图像块。通过 8×8 大小的 BING 特征的两个特点，我们提出了一个快速计算 BING 特征的算法（算法 2）。我们仅仅使用位运算中的位移和或运算就可以避免循环计算，提高了运算速度。首先，BING 特征 $\mathbf{b}_{x,y}$ 及其最后一行 $\mathbf{r}_{x,y}$ 各自需要存储到单一的 INT64 和字节中。然后，临近的 BING 特征和相对应的行有相近的关系。见图 2 和表 2，位运算符从 $\mathbf{r}_{x,y}$ 上移一位到 $\mathbf{r}_{x-1,y}$，并留出空间给比特 $b_{x,y}$ 进行比特位的或操作。同样地，位运算符从 $\mathbf{b}_{x,y}$ 移 8 位到 $\mathbf{b}_{x,y-1}$，并留出空间给 $\mathbf{r}_{x,y}$ 放置。

高效的 BING 特征计算方法具有自然图像的积累特性，呈现了整幅图像的连贯性。处理 8×8 的候选区域的计分值时，我们并不是仅计算一个值，而是使用位运算（或运算、位运算等）得到一系列的二值序列。

公式（1）中图像窗口对应的 BING 特征 $\mathbf{b}_{k,l}$ 得到的滤波器的值可以有效地通过以下公式进行测试：

$$s_l \approx \sum_{j=1}^{N_w} \beta_j \sum_{k=1}^{N_g} C_{j,k}, \tag{6}$$

其中，$C_{j,k} = 2^{8-k}(2<a_j^+, \mathbf{b}_{k,l}>-|\mathbf{b}_{k,l}|)$，即 $C_{j,k}$ 可以通过快速按位计算，以及通过 SSE 的位 1 计数继续测试。

实施细节：我们使用 1 维的滤波器[−1, 0, 1]来查找图像的水平方向梯度值 g_x 和竖直方向梯度值 g_y，同时使用 $\min(|g_x| + |g_y|, 255)$来表示归一化梯度值，并将这个值存入一个字节的存储变量中。我们默认采用彩色图像。在公开的 C++代码中，已开启 POPCNT SSE 和 OPENMP 的指令运算。

致谢

本课题在选题及研究过程中得到沈为老师的悉心指导。

特别感谢张之江老师和沈老师在科研上对我尽心的指导与帮助。从大二进入图像处理实验室以来，我尽早地接触了图像处理的方方面面，编程能力也逐渐提升，这过程离不开导师、学长的帮助。在张老师的鼓励下，我积极投入到编程的实战中。在沈老师的指导下，我边阅读英文文献边做程序开发，实现过：VL-feat 库实现 HOG 算法，基于 LBP 的实时人脸区分算法以及基于局部约束线性编码的图像分类算法等。沈老师渊博的知识、勤恳的科研态度令我敬佩不已；在这过程中我得到了沈老师的各种帮助，对图像处理方面有了一定的了解。

毕业设计中，候选区域获取的一些较难理解的部分以及运行程序过程中所产生的问题，沈为老师积极给予答复；结构随机森林的训练过程，赵凯学长向我描述了详细过程，以及代码的某些参数设置，及时地给予了解答。在此谨向所有帮助过我的人致以最诚挚的谢意！

参考文献

[1] Yang Ming-Hsuan, Kriegman David J, Ahuja Narendra. Detecting faces in images: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24(1): 34-58

[2] Fischler Martin A, Elschlager Robert A. The representation and matching of pictorial structures. IEEE Transactions on Computors, 1973, 100(1): 67-92

[3] Yuille Alan L. Deformable templates for face recognition. In Journal of Cognitive Neuroscience, 1991, 3(1): 59-70

[4] Cootes Tim F, Taylor Christopher J. Statistical models of appearance for medical image analysis and computer vision. In Medical Imaging, 2001:236-248

[5] Viola Paul, Jones Michael J. Robust real-time face detection. International journal of computer vision, 2004, 57(2): 137-154

[6] Dalal Navneet, Triggs Bill. Histogram of oriented gradients for human detection. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005: 886-893

[7] Lampert Christoph H, Blaschko Matthew B, Hofmann Thomas. Beyond sliding windows: Object localization by efficient subwindow search. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008: 1-8

[8] Russakovsky Olga, Ng Andrew Y. A steiner tree approach to efficient object detection. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010: 1070-1077

[9] Gall J, Lempitsky V. Class-specific Hough Forests for Object Detection[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition，Miami，FL，USA，2009：1022-1029.

[10] Alexe, B., Deselaers, T., Ferrari, V.: Measuring the objectness of image windows. PAMI 34(11) (2012)

[11] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. IJCV 88(2) (2010) 303–338

[12] Carreira, J., Sminchisescu, C.: Cpmc: Automatic object segmentation using constrained parametric min-cuts. PAMI 34(7) (2012)

[13] Marr, D.: Vision: A computational investigation into the human representation and processing of visual information. Inc., New York, NY (1982)

[14] Manen, S., Guillaumin, M., Van Gool, L., Leuven, K.: Prime object proposals with randomized prims algorithm. In: ICCV. (2013)

[15] Doll´ar, P., Zitnick, C.L.: Structured forests for fast edge detection. In: ICCV. (2013)

[16] Cheng, M.M., Zhang, Z., Lin, W.Y., Torr, P.: BING: Binarized normed gradients for objectness estimation at 300fps. In: CVPR. (2014)

[17] Doll´ar, P., Zitnick, C.L.: Fast edge detection using structured forests. CoRR abs/1406.5549 (2014)

[18] Canny, J.: A computational approach to edge detection. PAMI (6) (1986) 679–698

[19] Hosang, J., Benenson, R., Schiele, B.: How good are detection proposals, really? In: BMVC. (2014)

[20] C. Lawrence Zitnick and Piotr Dollár Edge Boxes: Locating Object Proposals from Edges. ECCV 2014.

[21] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip Torr. BING: Binarized Normed Gradients for Objectness Estimation at 300fps. CVPR 2014.

[22] Doll´ar, P., Zitnick, C.L.: Fast edge detection using structured forests. CoRR abs/1406.5549 (2014)

[23] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? CVPR, pages 73–80, 2010.