摘　要

目标跟踪是当前计算机视觉领域的研究热点之一。经过几十年的发展，目标跟踪在许多方面已经取得了很多进展，但其在复杂场景下的跟踪可靠性依旧是当前的研究难点之一。本文研究了现有的典型 Meanshift 目标跟踪算法，并针对复杂场景中的目标遮挡这个跟踪难点，对 Meanshift 跟踪算法进行了研究和改进，主要研究成果如下：

针对复杂场景下的目标遮挡问题，本文分两部分对典型的 Meanshift 算法进行了改进，提出了一种基于置信图的 Meanshift 迭代算法。

⑴借鉴 HOG 算法的思想，将普通颜色直方图改进为分块颜色直方图。分块颜色直方图由于将目标的整体区域颜色特征划分成了数个局部区域的颜色特征，具有很强的局部特征表达能力。

⑵基于直方图对比，提出了置信图的概念。置信图实质上为概率密度图，它代表了目标区域的特征在一幅新图像中存在的概率。最后利用 Meanshift 算法在置信图中迭代找到该帧图像的目标区域。

实验结果表明，由于对目标的颜色直方图进行了细致的划分，并且得到了可靠的置信图，该算法能有效地解决目标遮挡问题，具有较好的鲁棒性。

关键词：分块颜色直方图；HOG；置信图；Meanshift；目标跟踪

Abstract

Target tracking is one of the hot spots in the field of computer vision. After several decades of development, target tracking has made much progress in many aspects, but its reliability is still one of the problems in the complex scene.This paper analyses the existing typical Meanshift target tracking algorithm. And then aiming at the difficulty of object occlusion in complex scenes, we improve the Meanshift tracking algorithm. The main research results are as follows:

Aiming at the problem of object occlusion in complex scenes, this paper improves the typical Meanshift algorithm in two parts, and puts forward a new Meanshift iterative algorithm based on confidence map.

(1)Referring to the idea of HOG algorithm, this paper improves the color histogram into block color histogram. Block color histogram has a batter ability of describing local features than the color histogram.

(2)Based on the histogram comparison, this paper puts forward the concept of confidence map. The confidence map is probability density map in fact. It represents the probability of the target area features in a new image.

The experimental results show that due to the divided color histogram of the target and the reliable confidence map,the proposed algorithm is robust and can effectively solve the problem of target occlusion.

Key Words: Block color histogram; HOG; Confidence map; Meanshift； Target tracking

目 录

第 1 章 绪论.....1
1.1 研究背景和意义.....1
1.2 国内外研究现状.....2
1.2.1 目标跟踪发展历程.....2
1.2.2 目标跟踪算法分类.....3
1.3 本文研究内容.....5
第 2 章 目标跟踪基本理论.....6
2.1 目标跟踪原理.....6
2.2 相关技术介绍.....8
2.2.1 直方图.....8
2.2.2 反向投影.....9
2.2.3 Meanshift 算法.....10
2.2.4 HOG 特征.....11
2.3 本章小结.....12
第 3 章 基于置信图的 Meanshift 目标跟踪.....13
3.1 改进的颜色直方图特征.....13
3.1.1 分块颜色直方图的概念.....13
3.1.2 分块颜色直方图特征提取.....13
3.2 置信图.....14
3.2.1 直方图对比.....14
3.2.2 生成置信图.....15
3.3 本文算法流程实现.....16
3.4 本章小结.....17
第 4 章 本文目标跟踪算法实验.....18
4.1 实验平台开发环境.....18
4.2 实验结果与分析.....18
4.2.1 参数选择.....18
4.2.2 目标未被遮挡情况.....20
4.2.3 目标部分被遮挡情况.....21
4.2.4 目标大面积被遮挡情况.....22
4.3 本章小结.....23
第 5 章 结论.....24

参考文献.....25
致　谢.....28
附录 A：程序清单.....29

第 1 章　绪论

1.1　研究背景和意义

现代社会计算机和其它数码产品的快速发展，一方面让图像和视频变得更容易获取，另一方面也对图像的智能处理提出了更高的要求。计算机视觉使用计算机来帮助人们分析图像和视频内容，可以根据实际需求对图像和视频进行自动处理。自从Marr等[1]于1982年提出第一个使用计算机来描述视觉的框架后，计算机视觉受到了许多领域的研究人员的关注。当前的计算机视觉领域有许多方向，而基于视觉的目标跟踪一直是其中的一个热门分支[2,3]。视觉目标跟踪是在视频或连续图像序列中找到感兴趣的目标并计算出其在每一帧中的位置，进而将结果输出以完成其它高级任务的一种技术。随着计算机性能的逐渐提高和摄像终端的推广，目标跟踪受到了越来越多的关注，也应用到了更多的领域。当前基于视觉的目标跟踪主要应用在如下范围：

⑴军事方面

精密制导火器是当代战争中的重要作战装备之一。红外制导技术[4]是常用的末端制导技术之一，其使用红外成相来检测和跟踪目标，因而具有不易受无线电干扰和可在低能见度下工作的优势。近些年来无人机在军事任务中得到了较多应用，如无人机目标侦察和跟踪[5]等，这其中也应用了目标跟踪技术。

⑵安防监控

智能视频安防监控系统[6]被广泛用于银行、超市，小区等各类场所，以监控可能出现的异常情况。相较于人工监视和事后取证的传统监控系统，智能监控系统对视频中的运动目标进行跟踪和识别分析，并在出现非正常情况时及时发出警报，从而减少人力成本并提高效率。

⑶智能交通

当前城市交通流量越来越大，智能化的交通流量控制和监控也越来越重要。交通流量监测、车辆定位等都需要目标跟踪的参与，相关的典型系统主要有VISATRAM、VIPS等[7,8]。

⑷机器人自主视觉

工业机器人利用相机获取的视觉信息对目标进行定位和跟踪以完成各种工业任务。在各类恶劣工作环境如喷漆、放射污染、高温高压等情况下，有视觉功能的机器人可以代替人类完成危险劳动。

⑸人机交互

近年来比较流行的体感游戏机让玩家可以通过肢体运动来与游戏中的角色进行互动，典型的如微软的Xbox 360和Xbox One，以及索尼的Play Station 4等。这类机器

的核心技术之一便是人体肢体的识别和跟踪，配套的相机（Kinect或PlayStation Camera）拍摄玩家的影像并进行分析、提取玩家的运动信息并作为游戏机的输入信息以代替传统的手柄等控制器，可以得到更好的游戏体验，如图1-1所示。在工业设计和3D建模等领域，Leapmotion等设备可以跟踪双手的状态，并将其转化为操作控制信号，进而更好地查看建模效果，提高工作体验和效率。

图1-1 Kinect使用示意图

⑹医学影像

如今的许多医学检查采用影像方式，使用目标跟踪等计算机视觉技术能有效协助医生对影像的分析[2]。如在医学超声波成像和核磁共振成像中，由于噪声较大，很难使用肉眼直接识别。使用目标跟踪方法能有效地提升目标检测的准确性，以提升疾病诊断结果的可靠性。另外，在对药物在体内的扩散机制等的研究中也大量使用到了医学影像[9]，传统的人工分析工作量过大且严重依赖研究人员的专业知识，使用目标跟踪技术能有效减小工作量且能得到更准确的结果。

经过多年的研究和发展，目标跟踪技术已经在很多方面得到了实际应用，但其在很多方面还存在许多不足。当前的主要问题在于实际应用场景通常比较复杂，存在很多如遮挡、光照变化、目标尺度变化和运动模糊等的干扰，现有的目标跟踪算法在这类烦杂场景下的跟踪精度还有待提升。

1.2 国内外研究现状

1.2.1 目标跟踪发展历程

目标跟踪技术的研究最早可以追溯到20世纪50年代[10]。随后Kalman算法[11]于1960年被提出，这为不少目标跟踪算法的基础之一。70年代Singer等人[12,13]将Kalman滤波理论应用到目标跟踪算法中来，获得了较多关注和相关研究。90年代提出的均值滤波（Meanshift, MS）[14]算法和粒子滤波（Particle Filter, PF）[15]算法，使目标跟踪算法的研究有了进一步发展。Meanshift算法使用迭代来获得目标中心的最优值，其简单有效。自从Comaniciu等人[16]使用Meanshift结合颜色直方图进行目标跟踪后，Meanshift算法就因其简单且效果不错而受到了广泛研究。许多研究者在Meanshift基础

上做出了大量改进，以提高算法的跟踪精度[17,18]。Particle Filter算法将目标跟踪中的目标位置看作是一个概率分布问题，其通过带有权重的离散采样点来计算系统的后验概率密度，进而得到目标位置。Meanshift和Particle Filter已经成为目标跟踪领域两个经典的算法框架，大量的相关研究均是在其基础上进行的改进。

Henriques等人[19]在频域下使用傅里叶变换加速图像卷积计算，极大地提高了使用相关性跟踪的速度，后续改进的KCF[20]算法和时空上下文算法（STC）[21]等算法在保证速度的同时，提升了跟踪精度。也是当前研究热点之一。

2000年以后，基于机器学习（Machine learning, ML）的跟踪算法开始大量出现，该类算法将目标跟踪看成是一个二分类问题：前景目标和背景。该类算法充分利用机器学习在分类上的精度优势，在目标跟踪上可实现较高的精度，因而也成为目前研究的热点和主要方向之一。分类器的训练主要分为离线学习和在线学习。相较于离线学习，在线学习可以更好地应对目标和背景的变化，是当前研究的重点。

常见的用于目标跟踪的机器学习的算法有Boosting[19,22-24]算法、支持向量机（Support vector machine, SVM）[25,26]算法和P-N学习算法[27]等。Boosting算法原理是将若干弱分类器组合为一个强分类器，提高了分类器的精度。Kalal等人[27]于2010年提出P-N学习并将其用在目标跟踪中。P-N学习引入约束条件来修正错误分类样本，以提高分类器的精度。Kalal等人随后提出Tracking-Learning-Detection（TLD）跟踪算法[28]，将P-N学习与LK光流法和串级分类器相结合，在跟踪精度和速度上均得了较好的效果。Boris等人提出的多实例学习（MIL）跟踪[29]将多个图像块组成一个小的样本集，并用这些样本集组成训练集来训练分类器，这样可以实现更好的分类和跟踪效果。当前的不少目标跟踪算法为了提高精度，也常常会结合多种方法来进行跟踪。

1.2.2 目标跟踪算法分类

目标跟踪一直发展到现在，期间出现了许多算法，按研究侧重点来讲也有多种不同的分类方式。

按目标信息来源来分，主要有两类处理方式：自底向上（Bottom-up）和自顶向下（Top-down）。

自底向上的方式的特点是不依赖目标的已有信息，直接从图片序列中获取跟踪目标的有关信息，所以也称为数据驱动（Data-driven）。典型的算法如背景差分（Background subtraction）法和相关匹配（Correlation matching）算法等。自底向上的算法适合于特定应用场景下的目标跟踪，如跟踪工厂流水线上运动的产品。

自顶向下的方式需要利用已有信息，如算法初始化时给定的模板或是针对特定应用场景而预先训练好的模型等。在跟踪程中算法在图像中搜索与给定的模板或预先训练好的模型最匹配的区域作为输出。自顶向下的方式在目前研究得比较多，特别是在当前结合机器学习的跟踪算法中，基本都是基于自顶向下的方法。

按应用场景来分，目标跟踪主要有以下几种类型：

⑴按跟踪目标的数量，可以分为单、多目标跟踪。对于军事或其它领域，对目标的打击或监视通常为单目标跟踪。而对于安防监控等场合，则更多地需要同时监控和跟踪多个目标。多目标跟踪中，目标之间可能存在众多干扰等问题，使其相较单目标跟踪更为复杂。

⑵按跟踪目标的性质分为刚体跟踪和非刚体跟踪。刚体目标是指目标本身具有刚性结果，不易发生弯曲等形变，如汽车等。对应地，非刚体目标是指容易发生形变的目标，如人等。刚体目标在跟踪过程中其各个子区域的结构关系稳定、不会发生变化，因而容易进行表征，对其进行跟踪也较容易。而非刚体在跟踪过程中的子区域的结构关系会发生变化，如人在行走过程中四肢会变化，这会给目标表征带来困难，影响跟踪效果。

⑶按相机是否运动分为相机静止的目标跟踪和相机运动的目标跟踪。在相机静止时，背景图像通常也为静止，因为使用简单的帧间差分法即可分离静止背景和运动的前景目标，方便进一步处理。相机运动时，图像序列中的背景和前景目标都会有变化，因而处理起来就会更加困难。

按目标的表征方式来看，目标跟踪又可以分为以下几类：

⑴基于区域的跟踪(Region-based tracking)通过特定几何形状的区域的匹配来实现跟踪，是较常见的一类跟踪算法。区域跟踪算法通常使用模板匹配来进行跟踪目标，其在图像中寻找与模板相似度最高的区域，并将其作为目标区域。模板由手工选取或自动检测，一般模板为矩形，模板使用的信息可以为灰度信息、颜色信息、边缘信息等。该类方法在具有很好的跟踪精度和跟踪稳定性，但其缺点在于对目标遮挡等变化比较敏感，且运算量较大不利于实时性。典型的算法如Lucas等人提出的光流法[30]。

⑵基于特征的跟踪（Feature-based tracking）与基于区域的跟踪类似，也是通过匹配来实现目标跟踪。与基于区域的模板匹配不同的是，本类算法在匹配时使用的是提取后的特征，而不是某个区域。基于特征的跟踪使用的特征通常有Harris角点[31]、SIFT 算子[32]、SUSAN算子[33]等。为了保证跟踪效果，在进行特征匹配时通常会引入约束条件，如利用目标的速度、加速度等来进行约束，因而这类跟踪通常与Kalman滤波器联合使用。相较基于区域的跟踪，其对目标的部分遮挡、尺度变化、形变和光照变化等不太敏感，因此也受到较多的研究和应用。但是该类算法同样存着对噪声以及特征选择等比较敏感的问题，同时在部分特征提取过程也存着计算量过大不利于实时性的问题。

⑶基于主动轮廓模型的跟踪(Active contour-based tracking)也称作基于动态边缘模板的目标跟踪，它的核心思想是目标的边缘可以用按一定约束条件变形的封闭的曲线来表示。这条曲线变形后能与目标的轮廓一致，从而实现对目标的跟踪。典型的算法之一为Kass提出的Snake模型算法[34]。Snake算法通过对能量函数最小化而逐步

调整Snake曲线并使其最终与目标轮廓一致，因而可以较好地处理目标形状的变化。该模型通常与Kalman滤波方法配合使用，以提高跟踪效果。主动轮廓跟踪同时考虑灰度和轮廓信息，因而能较好地跟踪目标。其缺点是不能对多个目标进行跟踪，且当目标进行快速运动时其跟踪精度不够好。

⑷基于检测的跟踪（Tracking by detection）是近些年来研究较多的方目标跟踪方法。基于检测的跟踪使用目标信息来建立检测器，并在后续的图像序列中进行使用该检测器进行目标检测以实现跟踪。这类方法通常使用机器学习方法来训练目标检测器，并进行在线更新。

当前目标跟踪受到了国内外许多研究机构的关注。国外的不少高校如麻省理工学院、卡内基梅隆大学，斯坦福大学等都有计算机视觉的研究小组，并从事目标跟踪等相关研究，相关的研究项目有VSAM（Video surveillance and monitoring）等。国内的不少高校和科研院所也在目标跟踪方面进行了不少研究并取得了不少收获，值得一提的是中科院自动化所在安全监控等应用领域有不少理论成果和实际应用。

目前国际上很多计算机视觉领域的权威期刊和会议都将目标跟踪作为重要的研究内容之一。相关的顶级期刊包括PAMI（Pattern recognition and machine learning）和IJCV（International journal of computer vision）等。另外，计算机视觉领域三大顶级会议如CVPR（Computer vision and pattern recognition）、ICCV（International conference on computer vision）和ECCV（European conference on computer vision）也收录了不少相关研究的论文。

1.3 本文研究内容

由前文内容可知，目标跟踪在很多领域都有着应用需求，也有了许多研究成果。但是，实际应用场景通常比较复杂，在复杂场景下的跟踪依旧很有挑战性。本文在前人研究成果的基础上，针对复杂场景下的单目标跟踪进行研究。由于复杂场景下目标跟踪待解决的问题较多，目前也并不存在一个能解决所有问题的算法，故本文着重针对复杂场景下跟踪的遮挡问题进行了研究。

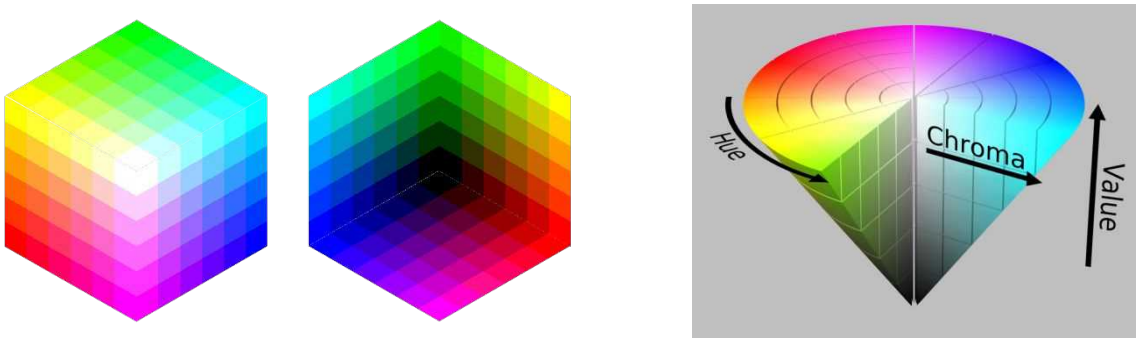
第 2 章　目标跟踪基本理论

2.1　目标跟踪原理

当前目标跟踪研究较多的是根据已知目标区域信息来进行跟踪，即给定第一帧图像和目标区域后，算法在后续图像中自动地输出目标位置。这类目标跟踪算法通常由三个主要模块构成：目标表征（Object representation）、搜索策略（Search mechanism）和模型更新（Model update）。

目标表征是指目标区域的表示方法，其是跟踪算法中最核心的内容之一，典型的如有直方图、Snake 边缘模型、能量谱等。目标表征通常与所选择的目标特征有较密切的关系，如颜色特征通常使用直方图表征方式，而边缘特征通常使用 Snake 模型等。在复杂场景中，一种特征通常无法有效表征目标，故通常会使用多个目标算法配合，如颜色、边缘、光流和纹理等。以下分别对典型的特征进行说明。

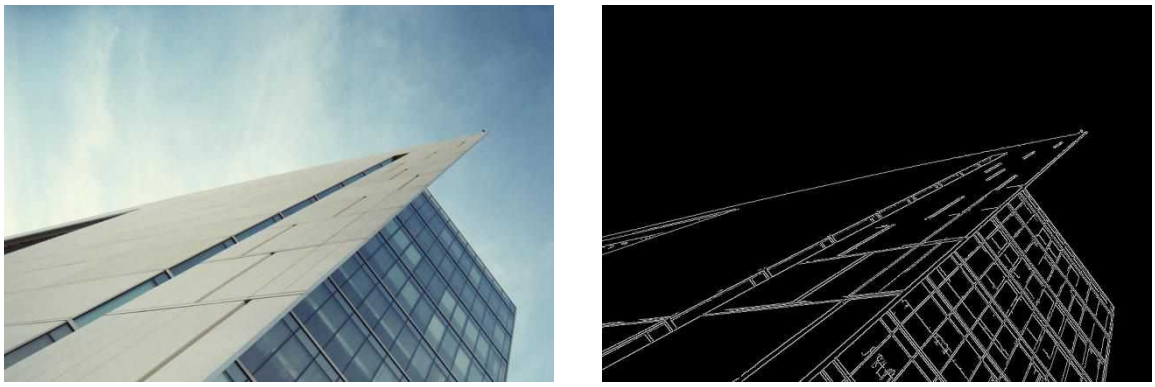
⑴颜色。颜色是最基本的视觉特征之一，其在各类目标跟踪算法中得到了广泛应用，如 Meanshift 跟踪算法和粒子滤波算法等。在计算机视觉中，颜色空间有多种，如 RGB颜色空间、HSV颜色空间、Lab颜色空间等。RGB 颜色空间将任一种颜色分解成红（Red, R）、绿(Green, G)、蓝(Blue, B)三种分量，是当前彩色图像和视频的主流表示方法，如图 2-1（*a*）所示。但由于这三个各个颜色分量相互存在较强的相关性，因而容易受到光照等的影响。HSV 空间使用色相（Hue, H）、饱和度(Saturation, S)、明度（Value, V）来表示颜色，如图 2-1（*b*）所示。将色相、饱和度和明度分解后，可减弱各分量的相关性，也在不少方面得到了应用。



（*a*） RGB 颜色空间　　（*b*） HSV 颜色空间

图 2-1 颜色空间示意图

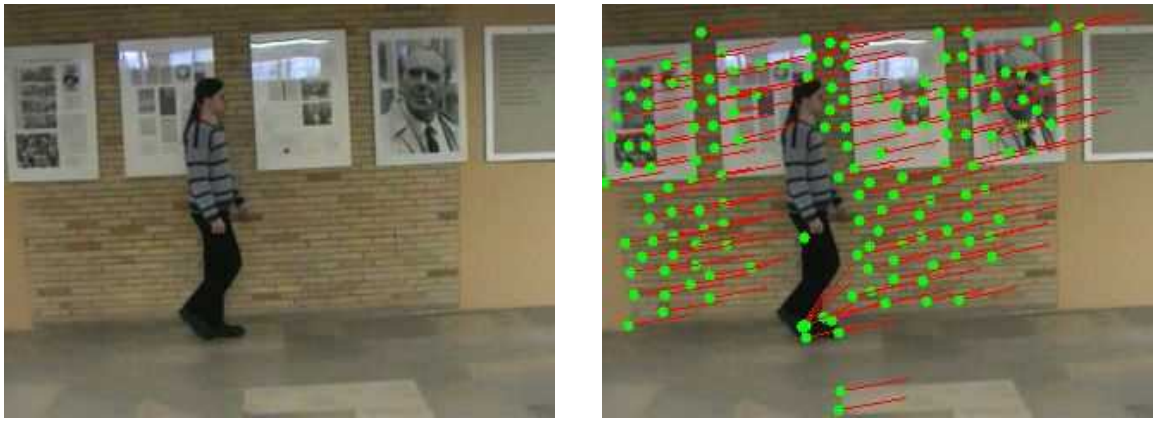
⑵边缘。边缘也是图像的重要特征之一，人和动物的视觉神经系统里均有对边缘信息响应的感受器[2]。在计算机视觉领域里，目标边缘被看作是像素点之间亮度的剧烈变化。边缘特征受光照变化影响不大，也常用于目标跟踪中。典型的边缘检测方法有 Canny 算子[35]和形态学检测方法等。在基于轮廓的目标跟踪算法中，Snake 检测算法[34]可以自动提取目标边缘，如图 2-2 所示。



（*a*）原图　　（*b*）结果图

图 2-2 边缘检测原图和结果图

⑶光流。光流表示了特定区域内每一点的灰度变化，其较多地用于运动检测。光流分为稀疏光流[30]和稠密光流[36]，两者的区别是前者只检测一组特征点。光流法存在的问题是运算量较大，且容易受光照干扰。当前使用较多计算光流的方法是 Lucas - Kanande 光流法[30]。如图 2-3 所示，由于摄像机在移动，所以光流法跟踪到了移动的背景。



（*a*）原图　　（*b*）结果图

图 2-3 光流跟踪原图和结果图

⑷纹理。纹理也是目标跟踪中常用的一个特征，其对光照变化也不太敏感。与边缘类似，纹理也可看做是像素点间的亮度变化。典型的描述方法有灰度共生矩阵[37]和局部二值模式（Local Binary Pattern, LBP）[38]。前者使用图像中两个像素的相关性信息，而后者使用中心点与其邻域内其它点的灰度大小信息。

搜索策略是指根据目标表征在新的一帧图像中寻找目标位置的策略。由于不同的算法其目标表征方式有差异，其对应的搜索策略也不完全一样。总的来看，目标跟踪算法中使用的搜索策略可以分为两大类：最优匹配和判别模型。

传统目标跟踪算法较多地使用最优匹配的方法，其利用目标特征来寻找与给定目标最相似的图像块，并将其作为目标搜索结果。典型的有如其 Meanshift 方法和 CSK 类方法[39]。Meanshift 方法采取梯度下降搜索的方法，其具有运算量小但容易陷入局部最优的问题；CSK 方法（包括 KCF[20]和 STC[21]等类似算法）在频域上加

速卷积运算来寻找极值，能达到较高的跟踪速度。

而近些年随着机器学习的兴起，许多跟踪算法使用一个图像判别模型来区分前景目标和背景。该类算法典型的有如基于AdaBoost的目标跟踪算法、基于P - N学习的跟踪算法等。对于采取判别模型的算法，其搜索策略通常是采取稠密采样的方法，即对整个图像区域进行搜索，其运算量较大，但通常能有更好的精度。

模型更新指在跟踪过程中对目标模型进行更新。由于目标在跟踪过程中通常会发生变化，因对目标模型进行更新非常重要，如 Matthew 在 LK 算法中使用初始帧和与当前帧的前几帧的信息来更新模型、TLD 算法使用 P-N 学习来更新模型等。模型更新的一个难点是在更新目标模型的同时不引入干扰。

典型的目标跟踪流程如下：

1.第一帧，利用给定的目标信息来初始化目标表征模型和其它变量；

2.后续帧，利用建立好的目标表征模型，使用搜索策略在图像中寻找目标的新位置并输出；

3.对目标表征模型进行更新，同时更新其它相关变量；

4.若跟踪未结束，跳转到第 2 步。

2.2 相关技术介绍

2.2.1 直方图

直方图广泛运用于很多计算机视觉运用当中，通过标记帧与帧之间显著的边缘和颜色的统计变化，来检测视频中场景的变化。在每个兴趣点设置一个有相近特征的直方图所构成“标签”，用以确定图像中的兴趣点。边缘、色彩、角度等直方图构成了可以被传递给目标识别分类器的一个通用特征类型。色彩和边缘的直方图序列还可以用来识别网络视频是否被复制。

简单说，直方图就是对数据进行统计的一种方法，并且将统计值组织到一系列事先定义好的bin当中，bin 的数值是从数据中计算出的特征统计量，这些数据可以是诸如梯度、方向、色彩或任何其他特征。直方图的意义如下：

⑴直方图是图像中像素强度分布的图形表达式。

⑵它统计了每一个强度值所具有的像素个数。

由于颜色是图像中最显著的特征信息，所以在目标跟踪中，颜色经常被用来描述对象。统计颜色信息的直方图称为颜色直方图，如图 2-4 所示。颜色直方图表示了目标区域中各颜色出现的频率，因此对目标的变形、尺度变化和旋转等具有一定的鲁棒性。如果目标颜色不变，仅外形有一定变化，则颜色直方图变化也不大。如果跟踪行人这种外形稍许变化，颜色几乎不变的目标，那么利用颜色直方图作为特征来跟踪是比较有效的。但是颜色直方图也有一定缺点，比如两类物体仅颜色相似，

但是外形差别很大的话，那么利用颜色直方图作为目标跟踪的特征就不能达到理想的跟踪效果。

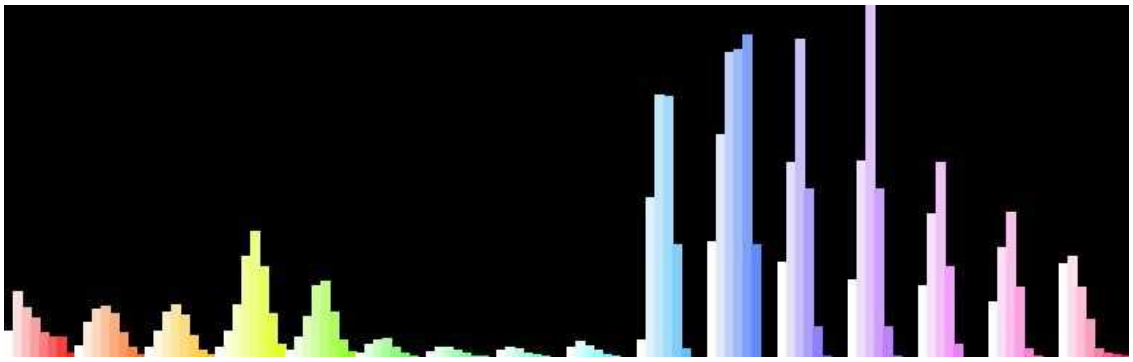


图 2-4 某图像 H-S 通道的颜色直方图

2.2.2 反向投影

如果一幅图像的区域中显示的是一种结构纹理或者一个独特的物体，那么这个区域的直方图可以看作一个概率函数，其表现形式是某个像素属于该纹理或物体的概率。反向投影（Back projection）就是一种记录给定图像中的像素点如何自适应直方图模型像素分布方式的一种方法。反向投影工作原理如下：

首先获取一幅测试图像，如图 2-5（*a*）所示。计算该图像的色调（H）通道的颜色直方图。然后对测试图像中的每个像素 $p(i,j)$，获取色调数据并找到该色调 $h(i,j)$ 在直方图中的 bin 位置，查询颜色直方图中对应 bin 的数值并将数值存储在新的反向投影图像中，如图 2-5（*b*）所示。

（*a*）原图

（*b*）结果图

图 2-5 反向投影原图和结果图

反向投影中存储的数值代表了测试图像中该像素属于目标区域的概率，在图 2-5（*b*）中，颜色越亮代表概率越高，颜色越暗代表概率越低。反向投影用于在输入图像中查找与特定图像最匹配的区域，也就是定义模板图像出现在输入图像的位置。

2.2.3 Meanshift 算法

由均值漂移（Meanshift）方法最初由 Fukunaga 等人提出，是一种非参数概率密度梯度估计算法。在经过 Cheng[40]的进一步研究后受到了许多关注，并在目标跟踪等领域得到了广泛应用。Meanshift 本身是一个迭代算法，其通过计算当前点的漂移均值并移动到对应点，然后重复计算漂移均值并移动，直到满足停止条件。后来，Conmaniciu 等人[18]将 Meanshift 应用到目标跟踪中，使该算法受到了广泛关注。

对于 d 维空间中的 n 个点，点 x 处的 Meanshift 向量表示为

$$m_h(x)=\frac{1}{k}\sum_{x_i\in S_k}(x_i-x) \tag{2-1}$$

其中，k 表示在这 n 个点中，只有 k 个点落入区域。的定义为满足下式的 y 点的集合。

$$S_h(x)=\{y:(y-x)^T(y-x)\le h^2\} \tag{2-2}$$

可见，Meanshift 算法计算当前点到区域内所有点的漂移量，然后取平均。因此，Meanshift 算法会使当前点向点密度最高的区域移动。如图 2-6 所示。

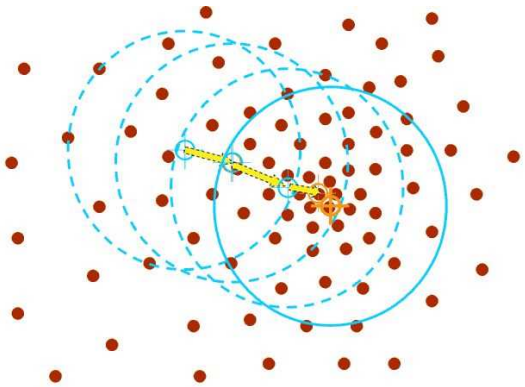


图 2-6 Meanshift 原理图

考虑到离当前点较近的样本点对估计更有效，在计算 Meanshift 向量时需要加上权值。Meanshift 算法使用径向对称的核函数进行加权

$$K(x)=\begin{cases}\frac{1}{2}c_d^{-1}(d+2)(1-\|x\|^2),\|x\|\le 1\\ 0,\text{其它}\end{cases} \tag{2-3}$$

其中，c_d 是 d 维单位球的体积。

引入核密度梯度估计后，Meanshift向量变为

$$M_h(x)=\frac{\sum_{t=1}^{n}(x_i-x)g\left[\left\|\frac{x-x_i}{h}\right\|^2\right]}{\sum_{t=1}^{n}\left[\left\|\frac{x-x_i}{h}\right\|^2\right]} \tag{2-4}$$

Comaniciu 等人的 Meanshift 跟踪方法使用颜色直方图作为目标表征。对给定的目标区域，其 RGB 通道分别为三个特征空间，各分为 16 份，每份为这个空间的一个子特征值，则整个区域的特征值有 $16^3=4096$ 个。

目标模板和候选区域的直方图之间的相似性使用 Bhattacharyya 系数来评估

$$\rho(p,q)=\sum_{u=1}^{m}\sqrt{p_u(y_0)q_u} \tag{2-5}$$

其中，y_0 为前一帧中目标的中心，p_u 和 q_u 都是密度函数。$\rho(q,p)$ 的值为 0 到 1 之间，$\rho(q,p)$ 越大，表示两个模板越相似。

Meanshift 跟踪算法计算量小，采用直方图建模因而对目标旋转和形变不敏感，但其无法有效处理目标尺度变化。

2.2.4 HOG 特征

方向梯度直方图（Histogram of oriented Ggradient，HOG）[41]是应用在计算机视觉和图像处理领域，用于目标检测的特征描述器，如图 2-7 所示。此方法用来计算局部图像梯度的方向信息的统计值。HOG 特征描述器的作者 Navneet Dalal 和 Bill Triggs 是法国国家计算机技术和控制研究所（INRIA）的研究员，他们在 2005 年的 CVPR 上首次提出了方向梯度直方图的概念。

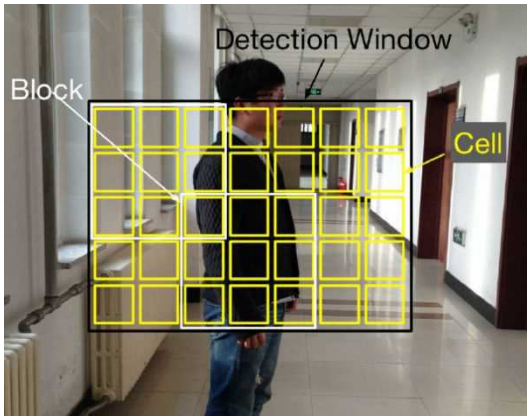


图 2-7 HOG 特征描述器

HOG 描述器最重要的思想是：在一幅图像中，局部目标的表象和形状能够被梯度或边缘的方向密度分布很好地描述。

下面是 HOG 算法的具体步骤：

1.梯度计算。梯度计算是 HOG 算法的第一步，Dalal 和 Triggs 指出，一些复杂的卷积核，如 3×3 的 sobel 卷积核在实验中并没有很好的效果，反而一维离散梯度模板是最简单而高效的方法，卷积核定义为 $[-1,0,1]$ 和 $[-1,0,1]^T$。梯度计算定义如下：

$$\begin{aligned} G_x(x,y)&=H(x+1,y)-H(x-1,y)\\ G_y(x,y)&=H(x,y+1)-H(x,y-1)\end{aligned} \tag{2-6}$$

上面的公式中，G_x 和 G_y 分别是像素的水平梯度和垂直梯度，H 为输入图像。遍历图像中的每个像素点来计算每个像素点的梯度。

像素点 $p(x,y)$ 梯度大小定义为：

$$G(x,y)=\sqrt{G_x(x,y)^2+G_y(x,y)^2} \tag{2-7}$$

像素点 $p(x,y)$ 梯度方向定义为：

$$\alpha(x,y)=\tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y)}\right) \tag{2-8}$$

2.分块直方图。在计算完成梯度之后，接着就是建立分块直方图。将窗口分割为数个相互之间有重叠的矩形块（block），每一个 block 块分割成几个固定数目的 cell 细胞单元。在每一个 cell 中统计这个 cell 本身的方向梯度直方图。

3.构建描述器。将一个 block 中包含的所有 cell 直方图相连得到 block 的直方图向量，归一化 block 的直方图向量。最后，将检测窗口中的各 block 直方图直接相连，得到完整的 HOG 向量，也就是 HOG 描述器。

2.3 本章小结

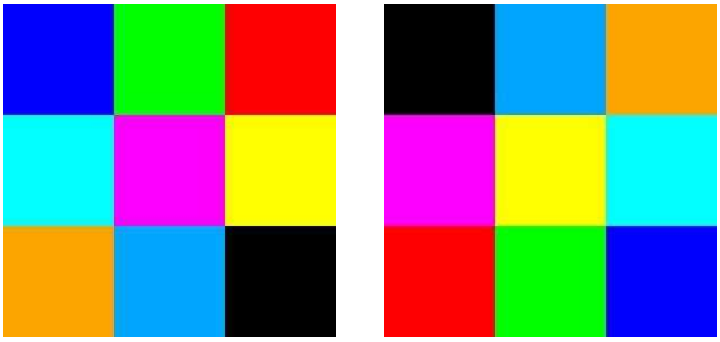
本章介绍了目标跟踪领域的一些基本技术。首先介绍了颜色直方图的概念，然后介绍了反向投影图的的概念及其功用，随后详细讲解了 Meanshift 算法的基本原理，最后仔细描述了 HOG 特征的概念以及提取 HOG 特征的过程。

第 3 章　基于置信图的 Meanshift 目标跟踪

3.1　改进的颜色直方图特征

3.1.1　分块颜色直方图的概念

第 2 章第 1 节中介绍的普通颜色直方图作为一种目标跟踪特征，在跟踪系统中能够有效地提取目标的颜色信息，而且对目标物体的小范围内形变不敏感。但是这种颜色直方图统计的是目标所在的整体区域的颜色信息，无法有效地表达出目标区域不同位置的颜色分布状况。例如图 3-1 所示，两幅图均由同样的 9 个颜色小块组成，所以这两幅图的整体颜色直方图是一样的。也就是说在目标跟踪中，普通的颜色直方图将认为这两幅图是一样的。但是很明显这两幅图不一样，9 个颜色小块在图上的排列顺序是不同的。



（*a*）　　（*b*）

图 3-1　颜色块排列顺序不一致，但整体颜色直方图一样的两幅图像

由此可见，如果单一的以目标整体区域的颜色直方图作为特征，将无法辨认图 3-1 中的两幅图像。此外，如果在跟踪过程中目标被部分遮挡，也将影响目标区域的整体颜色直方图，从而影响目标跟踪的准确性。为了解决上述问题，本文将 HOG 算法的分块思想引入到了颜色直方图中，将目标区域划分成互不重叠的子单元 box，将数个 box 组成一个块 block，block 之间相互重叠。提取并记录每个 block 的颜色直方图特征，最后将所有的 block 特征组合形成整块目标区域的特征向量。

由于将目标的整体区域颜色特征划分成了数个局部区域的颜色特征，因此能有效地辨别图 3-1 中的两幅图像。而且，如果目标局部被遮挡，那么只会影响被遮挡部分的局部颜色特征，而不会影响其它未被遮挡部分的颜色特征。在目标跟踪过程中，合理地利用各子区域的颜色特征将会提高整体算法的鲁棒性。

3.1.2　分块颜色直方图特征提取

分块颜色直方图在颜色空间和颜色统计方法上与传统颜色直方图是一致的，主要区别在于将目标区域做了进一步划分，原来的整体目标区域变成了多个小区域 block 的联合，因此能更有效地体现颜色特征。

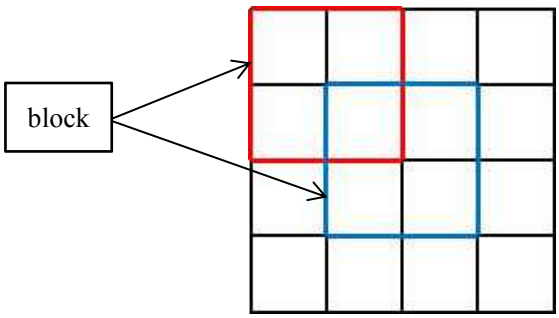


图 3-2 3×3 分块子区域

如图 3-2 所示，假设目标整体区域为宽高 $w \times h$ 的窗口，提取分块颜色直方图时，首先将该窗口划分成$m \times n$个大小为$x \times y$的重叠子区域block，对每一个block单独提取颜色直方图，然后将各子区域 block 的短颜色直方图按照固定顺序连接成一个长颜色直方图作为目标区域的颜色特征。在 HOG 算法中不同的子块 block 划分数目和重叠步长对目标识别的准确率有影响，同样在分块颜色直方图中，根据目标特点不同，使用不同参数划分的子区域 block 对目标特征的表达性能也不一样。经过试验发现，将目标区域划分成互不重叠的 4×4 的 box 小格，使用 2×2 个 box 组成重叠子区域 block，子区域 block 的重叠步长为 1 个 box，这样的划分方式能普遍满足各种目标物体的跟踪需求。

对于每一个子区域block，在HSV空间中提取颜色信息。在构建颜色直方图时，遍历子区域block中的每一个像素，通常情况下忽略亮度信息V，仅根据像素的 H，S 值投票至颜色直方图。

得到每一子区域 block 的颜色直方图后，逐一进行归一化并按照各自的位置顺序连接起来，组成完整的 9 维分块颜色直方图向量，每一维均是短颜色直方图。

在计算分块颜色直方图特征之前，可以先对图像进行降维处理，降维因子 scale 可以自定义选择。当选择合适的降维因子 scale 之后，既能加快处理速度又能保证有效地提取分块颜色直方图特征。

3.2 置信图

本文所提出的置信图实质上为概率密度图，它代表了目标区域的特征在一幅新图像中存在的概率，置信图的灰度值越大则概率越高。置信图的概念与反向投影类似但不完全一样，置信图相比反向投影图有着更优越的表达性能，在目标跟踪中有着出色的表现。

3.2.1 直方图对比

对于直方图来说，一个不可或缺的工具便是用某些具体的标准来比较两个直方图的相似度。要对两个直方图（比如说H_1和H_2）进行比较，首先必须选择一个衡量直方图相似度的对比标准（$d(H_1,H_2)$）。常用的对比方式有如下 4 种：

1.相关（Correlation）

$$d(H_1,H_2)=\frac{\sum_I(H_1(I)-\overline{H}_1)(H_2(I)-\overline{H}_2)}{\sqrt{\sum_I(H_1(I)-\overline{H}_1)^2\sum_I(H_2(I)-\overline{H}_2)^2}} \tag{3-1}$$

其中：

$$\overline{H}_k=\frac{1}{N}\sum_J H_k(J) \tag{3-2}$$

且 N 等于直方图中 bin 的个数。

2.卡方（Chi-Square）

$$d(H_1,H_2)=\sum_I\frac{(H_1(I)-H_2(I))^2}{H_1(I)} \tag{3-3}$$

3.直方图相交（Intersection）

$$d(H_1,H_2)=\sum_I\min(H_1(I),H_2(I)) \tag{3-4}$$

4.Bhattacharyya 距离

$$d(H_1,H_2)=\sqrt{1-\frac{1}{\sqrt{\overline{H}_1\overline{H}_2N^2}}\sum_I\sqrt{H_1(I)\bullet H_2(I)}} \tag{3-5}$$

本文选用相关（Correlation）方法，得到取值范围为 0-1 的相似度系数 $d(H_1,H_2)$。

3.2.2 生成置信图

在获得目标区域完整的 9 维分块颜色直方图向量 v_0 之后，读取下一帧图像 frame，按照如下步骤对图像进行处理：

1.用降维因子 scale 对图像进行降维处理；

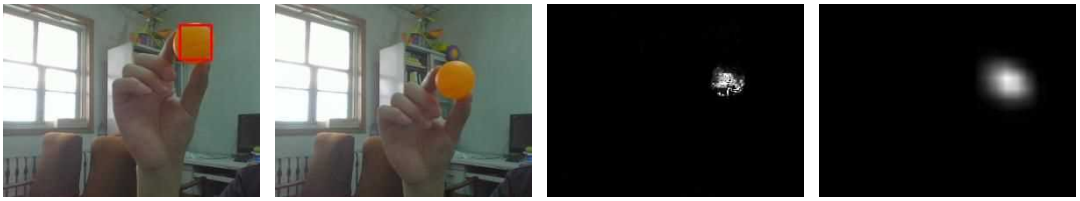
2.按照从左到右，从上到下的顺序用宽高为 $w\times h$ 的检测窗口按照自定义的步长 stride 去移动检测窗口；

3.根据检测窗口的宽高 $w\times h$ 和移动步长 stride 来计算横向和纵向能够遍历的次数 n_w 和 n_h；

4.求取当前检测窗口下的 9 维分块颜色直方图向量 v，采用 3.2.1 节所述的直方图对比方法对 $v[i]$ 和 $v_0[i]$（$i=0,1,\cdots 8$）进行直方图对比，得到相似度系数 $d_i(v_0[i],v[i])$（$i=0,1,\cdots 8$），对 $d_i(v_0[i],v[i])$（$i=0,1,\cdots 8$）相加并求出均值 d_{mean}，将 $d_{mean}\times 255$ 作为灰度值依次写入到置信图相应的像素中，置信图为宽高分别为 n_w 和 n_h 的灰度图；

5.若遍历未结束，跳转到第 2 步骤；

6.对置信图的灰度值归一化到区间 0-255，并且采用线性插值（INTER_LINEAR）方法进行上采样，调整到与图像 frame 的尺寸一致。



（*a*）　（*b*）　（*c*）　（*d*）

图 3-3 反向投影图和置信图对比

图 3-3 从左到右依次为标出目标区域的第 1 帧图、第 2 帧图 frame、frame 的反向投影图、frame 的置信图。

由图 3-3（*c*）可知，反向投影图并不能精确地反映原图的目标区域特征在下一帧图像 frame 中的位置，目标区域位置分散。而由图 3-3（*d*）可以看出，置信图很好地反映出了原图的目标区域特征在下一帧图像 frame 中的位置，目标区域位置集中。置信图相比反向投影图有着很明显的优势。

3.3 本文算法流程实现

本章的前两节详细介绍了本文算法的各主要流程，本节将介绍基于置信图的 Meanshift 目标跟踪算法的流程，算法流程图如下：

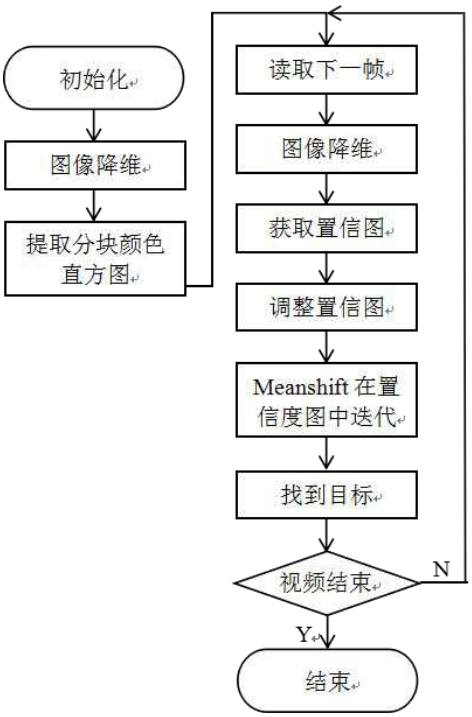


图 3-4 本文算法流程图

首先在第一帧图像中用矩形框选中目标区域，接着选择合适的降维因子 scale 对图像进行降维，然后对目标区域提取分块颜色直方图特征。读取下一帧图像，用选择好的降维因子 scale 对该帧图像进行降维处理，自定义检测窗口移动步长 stride，用检测窗口配合移动步长来遍历整幅图像从而生成置信图，最后用线性插值法调整

置信图大小，并利用 Meanshift 算法在置信图中迭代从而寻找目标区域。这样循环往复直到视频序列全部读取完毕。

不同降维因子 scale 和不同的检测窗口移动步长 stride，对于算法的处理速度和精度有着很大的影响。所以，在下一章本文将会着重讨论这两个参数的选取。

3.4 本章小结

本章主要是介绍了分块颜色直方图的概念和分块颜色直方图特征的提取方法，并且提出了置信图的概念，详细介绍了直方图对比方法和置信图的生成过程并与反向投影图进行对比，最后总结出了基于置信图的 Meanshift 目标跟踪流程。至此已将本文算法的全部流程详细介绍完毕。实验结果表明，置信图相比反向投影图有着更优越的特征表达性能，在目标跟踪中有着出色的表现。

第 4 章　本文目标跟踪算法实验

4.1　实验平台开发环境

本文使用 C++编程予以实现，编程环境为 Visual Studio 2010 旗舰版，基本的图像处理部分使用了著名的开源计算机视觉库 OpenCV，其版本号为 2.4.9。目标跟踪数据集选用国际标准的目标跟踪数据集 Visual Tracker Benchmark。计算机的处理器为 Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz，RAM 为 8.00GB，操作系统为 64 位 Windows 7 旗舰版。

4.2　实验结果与分析

实验采用 3×3 分块重叠的子区域 block 提取颜色直方图特征，利用 Meanshift 算法在置信图中进行迭代，参见第 3 章。

4.2.1　参数选择

本节针对两个重要参数：降维因子 scale 和检测窗口移动步长 stride，来讲解这两个参数对算法的处理速度和精度的影响。

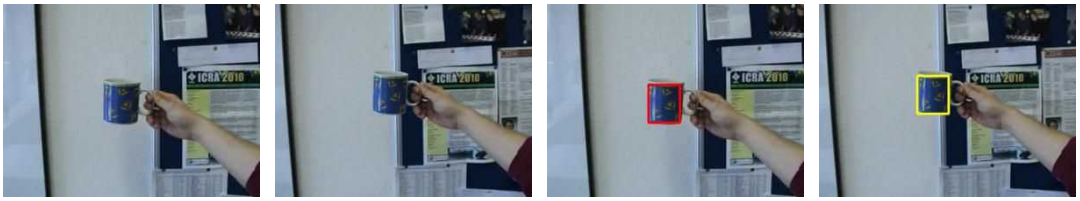
由于图像的数据量庞大，例如一幅 320×240 的图像就有多达 76800 个像素点。因此，降维因子 scale 的作用就是成倍地降低图像的像素数量来减少计算量，加快算法运行速度。但是减少的像素点过多容易导致目标区域的特征点数量也过多减少，从而影响算法的精度。所以，选择合适的降维因子 scale 很有必要。

本文选用 OpenCV 中的 resize 函数来对图像进行降维处理。resize 函数的原理是采用区域插值（INTER_AREA）方法来缩小图像。

在用检测窗口遍历图像的过程中，若移动步长 stride 过小，虽然生成的置信图非常精确，但是计算量巨大，耗时长，不能满足实时处理。若移动步长 stride 过大，虽然生成置信图的时间很快，但是生成的置信图不够精确，影响到了算法的精度，因此也不合适。所以，移动步长 stride 参数也关系到算法的处理速度和运行精度。

图 4-1 从左到右依次为第 1 帧原图、第 2 帧原图、标出目标区域的第 1 帧图、scale=1，stride=1 时，找出了目标区域的第 2 帧图。

本节采用 2 幅 320×240 大小的图像，如图 4-1（a）（b）所示。选择当 scale=1，stride=1 时，算法处理图 4-1（b）得到图 4-1（d）的运行时间 t_0 和图 4-1（d）中黄色方框的中心点 p_0 作为参考标准。当选择不同的 scale 和 stride 时，看算法处理图 4-1（b）的运行时间 t 和相应的结果图像中黄色方框的中心点 p 与 p_0 的欧氏距离 d，下文把 d 叫作精度，d 越小结果越精确。实验数据表格如表 4-1 所示。

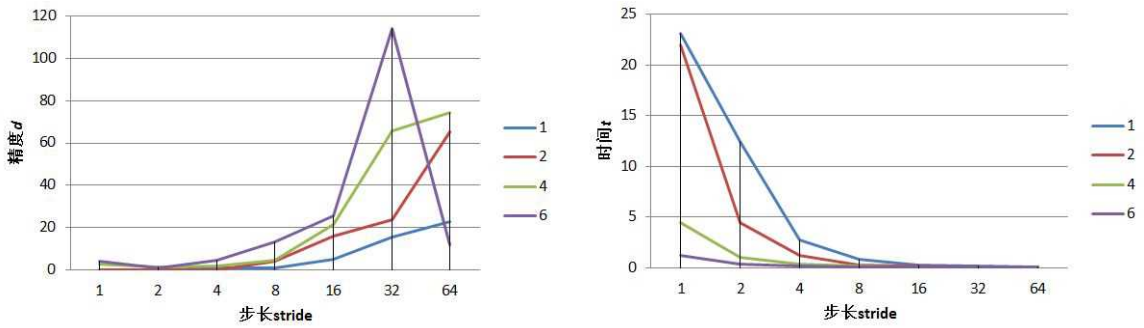


（*a*）　（*b*）　（*c*）　（*d*）

图 4-1 实验测试图

表 4-1　不同的 scale 和 stride 参数选择

stride scale	1	2	4	6
1	$t_0 = t = 23.11$s $d = 0.00$	$t = 21.92$s $d = 0.00$	$t = 4.480$s $d = 2.82$	$t = 1.266$s $d = 4.12$
2	$t = 12.46$s $d = 0.00$	$t = 4.471$s $d = 0.00$	$t = 1.065$s $d = 1.41$	$t = 0.355$s $d = 1.00$
4	$t = 2.804$s $d = 1.00$	$t = 1.225$s $d = 0.00$	$t = 0.381$s $d = 2.00$	$t = 0.167$s $d = 4.47$
8	$t = 0.847$s $d = 1.00$	$t = 0.340$s $d = 4.00$	$t = 0.174$s $d = 4.47$	$t = 0.139$s $d = 13.42$
16	$t = 0.317$s $d = 5.00$	$t = 0.183$s $d = 15.81$	$t = 0.122$s $d = 21.40$	$t = 0.101$s $d = 25.50$
32	$t = 0.176$s $d = 15.52$	$t = 0.129$s $d = 23.85$	$t = 0.120$s $d = 65.62$	$t = 0.067$s $d = 113.85$
64	$t = 0.126$s $d = 23.02$	$t = 0.112$s $d = 65.49$	$t = 0.061$s $d = 74.22$	$t = 0.082$s $d = 11.71$



（*a*）*d* 随 scale、stride 的变化图　　（*b*）*t* 随 scale、stride 的变化图

图 4-2 精度 *d*、时间 *t* 随 scale、stride 的变化图

从图 4-2（*a*）可以看出，整体趋势是步长 stride 越大、降维因子 scale 越大，精度越差，反之越好。但是在步长 stride 较小时，不论降维因子取何值，精度 *d* 都较小，也就是说算法的精度都很好；步长 stride 越大，精度 *d* 随着降维因子 scale 的不同相差也越大。

同理，从图 4-2（*b*）可以看出，整体趋势是步长 stride 越小、降维因子 scale 越小，处理时间越长，反之越短。但是在步长 stride 超过某一值后，不论降维因子取何值，处理时间均较短；步长 stride 越小，处理时间 *t* 随着降维因子 scale 的不同相差也越大。

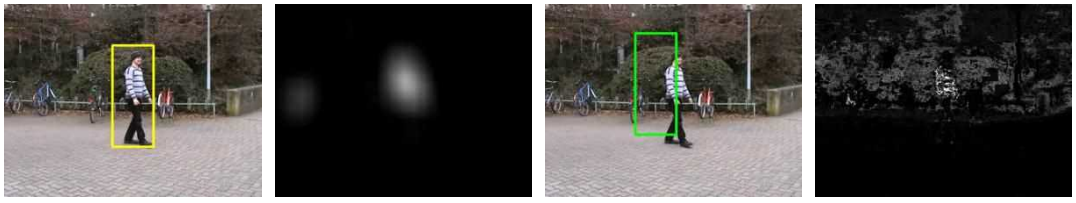
综合图 4-2 两幅图可以得出，当 scale=2，stride=6 时，算法既能保持较好的精度，又能大幅缩短运行时间，所以本文在下节的实验中就采取了该值。

4.2.2 目标未被遮挡情况

该情况下采用的是 Visual Tracker Benchmark 中的 BoBot 数据集的部分图片序列，针对的主要是图片中移动的人物目标，背景较为复杂。该图片序列分辨率为 320×240 像素，RGB 三通道彩色图像。

（a） 第 1 帧 初始目标帧

（*b*） 第 100 帧



（*c*） 第 619 帧

（*d*） 第 924 帧

图 4-3 （*b*）~（*d*）本文算法比较传统 Meanshift 算法

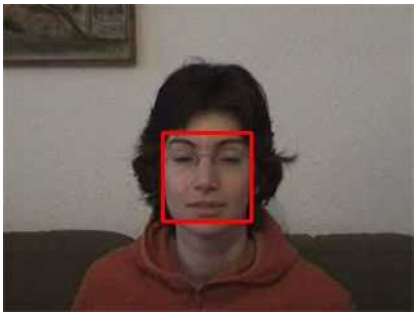
图 4-3 从左到右依次为本文算法跟踪结果图和置信图，传统 Meanshift 算法结

果图和反向投影图。

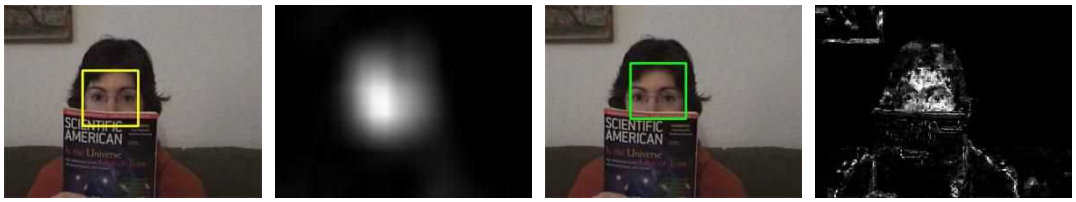
由图 4-3 可知，在第 100 帧，两种算法都能较好地找到目标区域。在第 619 帧，置信图仍能够较好地反映目标区域位置，而反向投影图的目标区域已被严重干扰。在第 924 帧，本文算法依然精确地找到了目标区域位置，而传统 Meanshift 算法已经完全跟丢了目标。

4.2.3 目标部分被遮挡情况

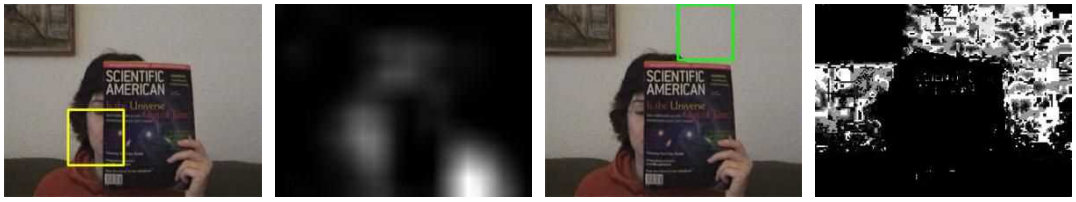
该情况下采用的是 Visual Tracker Benchmark 中的 faceocc1 数据集的图片序列，针对的主要是图片中被遮挡的人脸目标，遮挡情况由轻微遮挡到大面积遮挡，变化范围较大。该图片序列分辨率为 352×288 像素，RGB 三通道彩色图像。



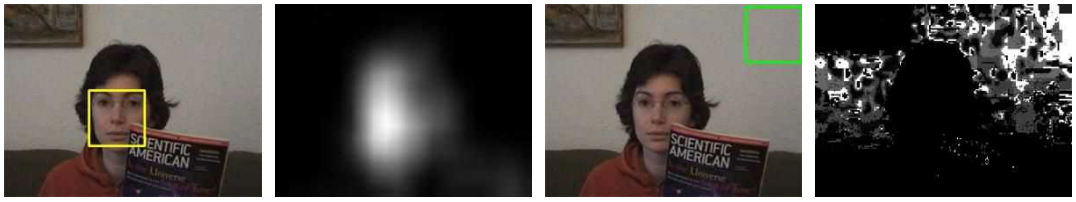
（a） 第 1 帧 初始目标帧



（*b*） 第 41 帧



（*c*） 第 555 帧



（*d*） 第 899 帧

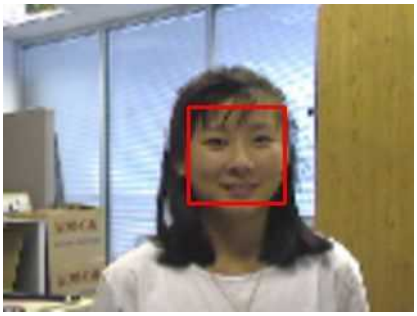
图 4-4 （*b*）~（*d*）本文算法比较传统 Meanshift 算法

图 4-4 从左到右依次为本文算法跟踪结果图和置信图，传统 Meanshift 算法结果图和反向投影图。

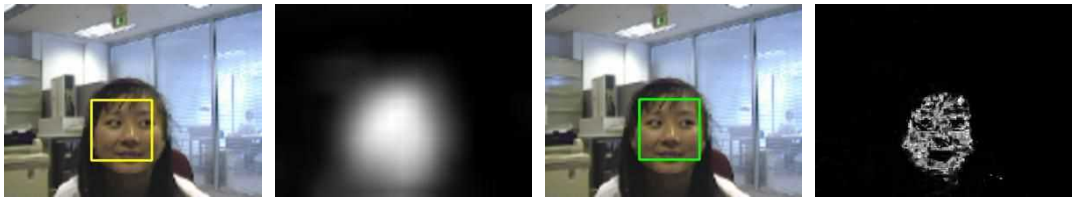
由图 4-4 可知，在第 41 帧，由于目标区域仅被轻微遮挡，两种算法都能较好地找到目标区域。在第 555 帧，目标区域被严重遮挡，由于手部的颜色与面部颜色接近，置信图出现了误差，但是由于 Meanshift 算法是在局部寻找极值，所以手部的干扰并不会造成影响，置信图仍能够较好地反映目标区域位置；而反向投影图的目标区域已被严重干扰，跟踪错误。在第 899 帧，目标区域几乎恢复到了未被遮挡状态，本文算法依然精确地找到了目标区域位置，而传统 Meanshift 算法已经无法再次找到目标了。

4.2.4 目标大面积被遮挡情况

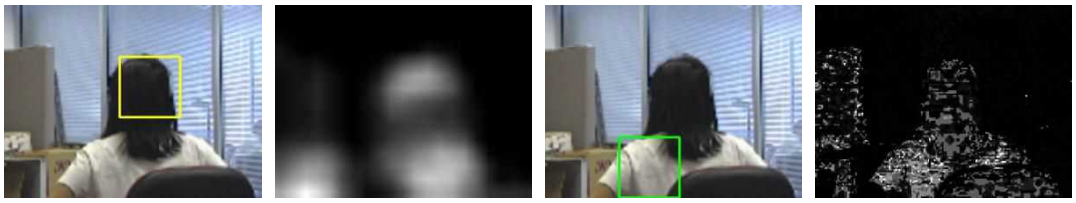
该情况下采用的是 Visual Tracker Benchmark 中的 Babenko 数据集的部分图片序列，针对的主要是图片中移动的人脸目标，背景颜色与人脸颜色很相似。该图片序列分辨率为 320×240 像素，RGB 三通道彩色图像。



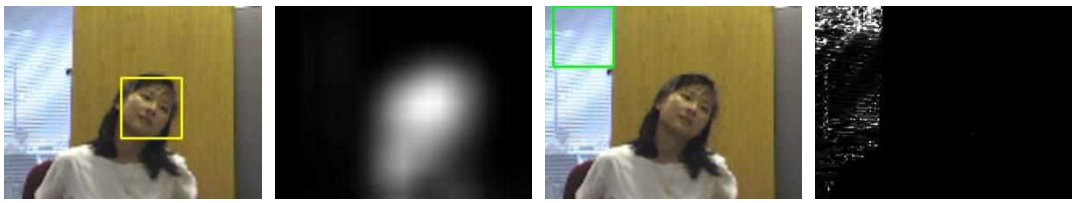
（b） 第 1 帧 初始目标帧



（*b*） 第 47 帧



（*c*） 第 105 帧



（*d*） 第 308 帧

图 4-5 （*b*）~（*d*）本文算法比较传统 Meanshift 算法

图 4-5 从左到右依次为本文算法跟踪结果图和置信图，传统 Meanshift 算法结果图和反向投影图。

由图 4-5 可知，在第 47 帧，虽然人脸外形稍微变大，但两种算法都能较好地找到目标区域。在第 105 帧，人脸完全背向镜头，但是由于在初始目标区域中，分块颜色直方图包含了黑色头发的颜色特征，所以即使人脸完全背向了镜头，本文算法也能检测出黑色的头发。而反向投影图的目标区域已经丢失。在第 308 帧，图像背景颜色和初始区域目标颜色相近，但是本文算法的置信图仍然给出了精确的目标位置，这是因为分块颜色直方图细致地划分了初始目标区域的颜色特征。反向投影图依然无法找到目标区域。

4.3 本章小结

本章主要讨论了本文算法参数的选择以及本文算法和传统 Meanshift 的跟踪结果的对比，从目标未被遮挡、目标部分被遮挡和目标大面积被遮挡三个方面对两种跟踪算法的性能进行了实验。实验结果表明，本文算法由于采用了分块颜色直方图特征和置信图，相比于传统 Meanshift 算法，在目标跟踪性能上有了很大的提升，验证了算法的有效性和可行性。

第 5 章　结论

本文研究了现有的典型 Meanshift 目标跟踪算法，并针对复杂场景中的目标遮挡这个跟踪难点，对 Meanshift 跟踪算法进行了研究和改进，主要研究成果如下：

针对复杂场景下的目标遮挡问题，本文分两部分对典型的 Meanshift 算法进行了改进，提出了一种基于置信图的 Meanshift 迭代算法。

首先，借鉴了 HOG 算法的主体思想，将普通颜色直方图改进成了分块颜色直方图。普通颜色直方图作为一种目标跟踪特征，在跟踪系统中能够有效地提取目标的颜色信息，而且对目标物体的小范围内形变不敏感。但是这种颜色直方图统计的是目标所在的整体区域的颜色信息，无法有效地表达出目标区域不同位置的颜色分布状况。分块颜色直方图由于将目标的整体区域颜色特征划分成了数个局部区域的颜色特征，如果目标局部被遮挡，那么只会影响被遮挡部分的局部颜色特征，而不会影响其它未被遮挡部分的颜色特征。在目标跟踪过程中，分块颜色直方图提高了整体算法的鲁棒性。

然后，基于直方图对比的方法，创新地提出了置信图的概念。置信图实质上为概率密度图，它代表了目标区域的特征在一幅新图像中存在的概率，置信图的灰度值越大则概率越高。置信图的概念与反向投影类似但不完全一样，置信图相比反向投影图有着更优越的表达性能，在目标跟踪中有着出色的表现。

最后利用 Meanshift 算法在置信图中进行迭代，最终找到该帧图像的目标区域。

实验结果表明，由于对目标的颜色直方图进行了细致的划分，并且得到了可靠的置信图，该算法能有效地解决目标遮挡问题，具有较好的鲁棒性。论文仍然有一些不足并需要进一步完善。后续可进行如下工作：

1. 加入 SURF 特征点提取过程。由于在目标跟踪过程中不可避免地会发生目标的大小的改变，本文算法的跟踪窗口并不能随目标大小的改变而改变。SURF 算法具有旋转不变性和尺度不变性，因此，加入 SURF 算法后能有效地改善跟踪性能。

2. 采用并行处理技术加速图像处理过程。由于图像处理的数据量非常大，在实际应用中如果想做到实时目标跟踪，就要加快图像处理的速度。采用并行处理技术能够将单一线程处理变为多线程同时处理，从而加快了图像处理速度。

参考文献

[1] David Marr. A computational investigation into the human representation and processing of visual information[J]. Vision, 1982: 125-126.

[2] A. Yilmaz, O. Javed, M. Shah. Object tracking: A survey[J]. ACM Computing Surveys, 2006, 38(4): 13-es.

[3] 侯志强, 韩崇昭. 视觉跟踪技术综述[J]. 自动化学报, 2006, 32(4): 603-617.

[4] 葛炜, 曹东杰, 郝宏旭. 红外制导技术在精确打击武器中的应用[J]. 兵工学报, 2010, (S2): 117-121.

[5] 姜长生. 无人机侦察/打击一体化的关键技术[J]. 电光与控制, 2011, (02): 1-7.

[6] 乔传标, 王素玉, 卓力,等. 智能视觉监控中的目标检测与跟踪技术[J]. 测控技术, 2008, (05): 22-24.

[7] Zhigang Zhu, Guangyou Xu, Bo Yang, et al. VISATRAM: A real-time vision system for automatic traffic monitoring[J]. Image and Vision Computing, 2000, 18(10): 781-794.

[8] Benjamin Coifman, David Beymer, Philip Mclauchlan, et al. A real-time computer vision system for vehicle tracking and traffic surveillance[J]. Transportation Research Part C: Emerging Technologies, 1998, 6(4): 271-288.

[9] 高婷婷. 基于多目标跟踪的医学影像分析[D]. 西安：西安电子科技大学, 2012.

[10] Pa West. Proceedings of the Society of Photo-Optical Instrumentation Engineers[J]. Journal of Modern Optics, 1976, 23(10): 844-844.

[11] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems[J]. Journal of basic Engineering, 1960, 82(1): 35-45.

[12] Robert A Singer. Estimating optimal tracking filter performance for manned maneuvering targets[J]. IEEE Transactions on Aerospace and Electronic Systems, 1970, (4): 473-483.

[13] Yaakov Bar-Shalom. Tracking methods in a multitarget environment[J]. IEEE Transactions on Automatic Control, 1978, 23(4): 618-626.

[14] Dorin Comaniciu, Peter Meer. Mean shift analysis and applications[C]// The Proceedings of the Seventh IEEE International Conference on Computer Vision (ICCV). Kerkyra: IEEE, 1999,2: 1197-1203.

[15] Michael Isard, Andrew Blake. Condensation—conditional density propagation for visual tracking[J]. International journal of computer vision, 1998, 29(1): 5-28.

[16]Dorin Comaniciu, Peter Meer. Mean shift: A robust approach toward feature space analysis[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24(5): 603-619.

[17]彭宁嵩, 杨杰, 刘志. Mean-Shift 跟踪算法中核函数窗宽的自动选取[J]. 软件学报. 2005, 16 (9): 1542-1550.

[18]李培华. 一种改进的 Mean Shift 跟踪算法[J].自动化学报. 2007, (33): 4.

[19]Helmut Grabner, Horst Bischof. On-line boosting and vision[C]// 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). New York : IEEE, 2006,1: 260-267.

[20]João F Henriques, Rui Caseiro, Pedro Martins, et al. High-speed tracking with kernelized correlation filters[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2015, 37(3): 583-596.

[21]Kaihua Zhang, Lei Zhang, Qingshan Liu, et al. Fast visual tracking via dense spatio-temporal context learning[C]// Computer Vision – ECCV 2014. Zurich: Springer, 2014: 127-141.

[22]Padmavathy Kankanala, Sanjoy Das, Anil Pahwa. AdaBoost: An ensemble learning approach for estimating weather-related outages in distribution systems[J]. Power Systems, IEEE Transactions on, 2014, 29(1): 359-367.

[23]Robert E Schapire. The strength of weak learnability[J]. Machine learning, 1990, 5(2): 197-227.

[24]Helmut Grabner, Christian Leistner, Horst Bischof. Semi-supervised on-line boosting for robust tracking[M]// Computer Vision – ECCV 2008. Marseille: Springer, 2008: 234-247.

[25]Yu-Dong Cai, Pong-Wong Ricardo, Chih-Hung Jen, et al. Application of SVM to predict membrane protein types[J]. Journal of Theoretical Biology, 2004, 226(4): 373-376.

[26]Paolo Mantero, Gabriele Moser, Sebastiano B Serpico. Partially supervised classification of remote sensing images through SVM-based probability density estimation[J]. Geoscience and Remote Sensing, IEEE Transactions on, 2005, 43(3): 559-570.

[27]Zdenek Kalal, Jiri Matas, Krystian Mikolajczyk. Pn learning: Bootstrapping binary classifiers by structural constraints[C]// 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). San Francisco: IEEE, 2010: 49-56.

[28]Z. Kalal, K. Mikolajczyk, J. Matas. Tracking-Learning-Detection[J]. IEEE Trans Pattern Anal Mach Intell, 2012, 34(7): 1409-1422.

[29]陈东成,朱明,高文,等.在线加权多示例学习实时目标跟踪[J]. 光学 精密工程,2014,22(6):1661-1667.

[30]Bruce D Lucas, Takeo Kanade. An iterative image registration technique with an application to stereo vision[C]// IJCAI. Vancouver: William Kaufman Inc. 1981,81: 674-679.

[31]Krystian Mikolajczyk, Cordelia Schmid. Scale & affine invariant interest point detectors[J]. International journal of computer vision, 2004, 60(1): 63-86.

[32]David G Lowe. Object recognition from local scale-invariant features[C]// 1999 The proceedings of the seventh IEEE international conference on Computer vision (ICCV). Kerkyra: IEEE, 1999,2: 1150-1157.

[33]Stephen M Smith, J Michael Brady. SUSAN—a new approach to low level image processing[J]. International journal of computer vision, 1997, 23(1): 45-78.

[34]Michael Kass, Andrew Witkin, Demetri Terzopoulos. Snakes: Active contour models[J]. International journal of computer vision, 1988, 1(4): 321-331.

[35]Myung-Cheol Roh, Tae-Yong Kim, Jihun Park, et al. Accurate object contour tracking based on boundary edge selection[J]. Pattern Recognition, 2007, 40(3): 931-943.

[36]Berthold K Horn, Brian G Schunck. Determining optical flow[C]// 1981 Technical symposium east. International Society for Optics and Photonics, 1981: 319-331.

[37]Robert M Haralick, Karthikeyan Shanmugam, Its' Hak Dinstein. Textural features for image classification[J]. IEEE Transactions on Systems, Man and Cybernetics, 1973, (6): 610-621.

[38]Timo Ojala, Matti Pietikainen, David Harwood. Performance evaluation of texture measures with classification based on Kullback discrimination of distributions[C]// 1994 Proceedings of the 12th IAPR International Conference on Pattern Recognition. Jerusalem: IEEE, 1994,1: 582-585.

[39]Martin Danelljan, Fahad Khan, Michael Felsberg, et al. Adaptive color attributes for real-time visual tracking[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Columbu : IEEE. 2014: 1090-1097.

[40]Yizong Cheng. Mean shift, mode seeking, and clustering[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1995, 17(8): 790-799.

[41]Dalal N， Triggs B. Histograms of oriented gradients for human detection [C]// Proceedings of Conference on Computer Vision and Pattern Recognition. Los Alamitos：IEEE Computer Society Press，2005,1:886-893.

致　谢

四年的大学学习生活即将结束，回想大学四年的时光，我的心中充满了快乐与感激。在论文成稿之际，谨向所有关心帮助我的人致以最真诚的感谢！最应该感谢的是我的导师张红颖副教授。在学习期间，张老师给予了我学习上的极大帮助和支持。张老师学识渊博，严谨认真，在学术上对我悉心指导，让我在科研方法和编程能力上都得到很大的提高。另外，张老师的为人处世和工作态度也值得我学习，是我在今后工作生活中的榜样。

附录 A：程序清单

```
#pragma once
#include <vector>
#include <iostream>
#include <opencv2/opencv.hpp>
using namespace cv;
class WinBox
{
public:
    WinBox();
    vector<Mat> computeHist(Mat& win, Size box);
    Mat confidence(Mat& img, Size box, vector<Mat>& winHist, int sc = 1, int st = 1);
private:
    int histSize[2];            //bin 数量
    float hranges[2];           //直方图的最小值与最大值
    float sranges[2];           //直方图的最小值与最大值
    const float* ranges[2];     //hranges 的地址值要赋给 ranges[0]
    int channels[2];            //通道
};

#include "WinBox.h"
WinBox::WinBox()
{
    histSize[0] = 60;           histSize[1] = 80;
    hranges[0]  = 0;            hranges[1]  = 180;
    sranges[0]  = 0;            sranges[1]  = 255;
    ranges[0]   = hranges;      ranges[1]   = sranges;
    channels[0] = 0;            channels[1] = 1;
}
//计算给定窗口 win 的分块直方图，并保存到 winHist 中
vector<Mat> WinBox::computeHist(Mat& win, Size box)
{
    vector<Mat> winHist;
```

```
    for (int i = 0; i < 3; ++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            //从左上开始遍历 2*2 box 小块
            Rect rect(j*box.width, i*box.height, 2*box.width, 2*box.height);
            Mat box = win(rect), boxHist;
            calcHist(&box, 1, channels, Mat(), boxHist, 2, histSize, ranges);
            normalize(boxHist, boxHist, 0.0, 1.0, NORM_MINMAX);
            winHist.push_back(boxHist);
        }
    }
    return winHist;
}
//读取下一幅完整图像，用窗口 win 遍历图像(步长为 1 像素)
//计算当前窗口 win 下的直方图，并和上幅图像得到的直方图依次进行对比
//将对比结果求均值，归一化，作为灰度值写入置信图的一个像素点
Mat WinBox::confidence(img, Size box, vector<Mat>& winHist, int scale, int stride)
{
    //初始化矩形区域(也就是 win 大小)
    Rect rect(0, 0, 4*box.width, 4*box.height);
    //用窗口扫描图像
    vector<vector<double>> row_size;
    for (int i = 0; i < img.rows; i += stride)
    {
        if (i+4*box.height > img.rows) break;
        vector<double> every_row;
        for (int j = 0; j < img.cols; j += stride)
        {
            Rect rect_temp = rect + Point(j,i);
            if (static_cast<int>(rect_temp.br().x) > img.cols) break;
            else
            {
                Mat winTemp = img(rect_temp);
                vector<Mat> winHistTemp = computeHist(winTemp, box);
```

```
                double sum = 0;        //相似度的和
                for (int t = 0; t < 9; ++t)
                {
                    double value = cv::compareHist(winHist[t], winHistTemp[t], 0);
                    if (value < 0) value = 0;
                    sum += value;
                }
                sum /= 9.0;            //求均值
                sum *= 255.0;
                every_row.push_back(sum);
            }
        }
        row_size.push_back(every_row);
    }
    int nr = row_size.size();
    int nc = row_size[0].size();
    Mat backproj = Mat::zeros(nr, nc, CV_8UC1);  //置信图
    for (int i = 0; i < nr; ++i)
    {
        uchar* data = backproj.ptr<uchar>(i);
        for (int j = 0; j < nc; ++j)
            data[j] = static_cast<uchar>(row_size[i][j]);
    }
    //调整置信图
    normalize(backproj, backproj, 0.0, 255.0, NORM_MINMAX);
    resize(backproj, backproj, Size(img.cols*scale,img.rows*scale), 0, 0, 1);
    return backproj;
}

#include "WinBox.h"
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;
int num = 0;
const int scale = 2;                //图片缩放因子
```

```
const int stride = 6;                    //窗口移动步长
const int img_width = 320;               //图片宽度
const int img_height = 240;              //图片高度
bool isWrite = false;
stringstream ss;

int main()
{
    VideoCapture cap;
    Rect rect_init;
    cap.open("E:\\Datasets\\Person\\src\\img%4d.jpg");
    rect_init = Rect(138, 80, 40, 52);
    Rect rect = rect_init;
    //讲初始跟踪区域划分成 4*4 小格，求出小格的大小
    const Size box(rect_init.width/scale/4, rect_init.height/scale/4);
    Mat frame, frameHSV, imgROI, backproj;
    WinBox win_box;
    vector<Mat> hist;
    while (1)
    {
        ++num;
        cap >> frame;
        if (frame.empty()) break;
        double timer0 = static_cast<double>(getTickCount());//计时
        if (num == 1)
        {
            //画出初始跟踪窗口
            rectangle(frame, rect, Scalar(0,0,255), 2);
            if (isWrite)
            {
                ss.str("");
                ss << "D:\\MY_RESULT\\" << num << ".jpg";
                imwrite(ss.str(), frame);
            }
            imshow("Frame", frame);
```

```
        waitKey(1);
        //按缩放因子缩小图片，减小计算量
        resize(frame, frame, Size(img_width/scale,img_height/scale), 0,0,1);
        //中值滤波处理
        medianBlur(frame, frame, 7);
        cvtColor(frame, frameHSV, CV_BGR2HSV);
        //计算缩小后的初始跟踪区域
        Rect temp(r_i.x/scale, r_i.y/scale, r_i.width/scale, r_i.height/scale);
        imgROI = frameHSV(temp);
        //计算缩小后初始跟踪区域的直方图
        hist = win_box.computeHist(imgROI, box);
        waitKey(0);
    }
    else
    {
        Mat frame_show = frame;//专门用来显示的
        resize(frame, frame, Size(img_width/scale,img_height/scale), 0,0,1);
        //中值滤波处理
        medianBlur(frame, frame, 7);
        cvtColor(frame, frameHSV, CV_BGR2HSV);
        //计算置信度图
        backproj = win_box.confidence(frameHSV, box, hist, scale, stride);
        if (isWrite)
        {
            ss.str("");
            ss << "D:\\MY_GT\\" << num << ".jpg";
            imwrite(ss.str(), backproj);
        }
        //显示置信度图，置信度图的大小和原始图像的大小一致
        imshow("Backproj", backproj);
        waitKey(1);
        //meanshift 停止迭代的条件
        TermCriteria criteria(MAX_ITER + TermCriteria::EPS, 1000, 0.001);
        //进行 meanshift 迭代，并讲找到的物体区域更新到 rect
        meanShift(backproj, rect, criteria);
```

```
            //在图像上画出更新后的跟踪区域
            rectangle(frame_show, rect, Scalar(0,255,255), 2);
            if (isWrite)
            {
                ss.str("");
                ss << "D:\\MY_RESULT\\" << num << ".jpg";
                imwrite(ss.str(), frame_show);
            }
            imshow("Frame", frame_show);
            waitKey(1);
        }
        timer0 = ((double)getTickCount()-timer0) / getTickFrequency();
        cout << "用时：" << timer0 << "  秒" << endl;
    }
    waitKey(0);
    return 0;
}
```