多区域客流统计系统研究

刘君宇

院 （系）：航天学院　　专　　业：自动化

学　　号：1130410305　　指导教师：王常虹 教授

2017 年 6 月

哈爾濱工業大學

毕业设计(论文)

题　　目 多区域客流统计系统研究

专　　业 自动化

学　　号 1130410305

学　　生 刘君宇

指 导 教 师 王常虹 教授

答 辩 日 期 2017.06.24

摘　要

客流统计系统的基本功能是统计通过一条检测线的人数。通过精确统计客流的实时流量，可以实现对店铺、公司等的科学管理，也可以用于指导客流量较大的场所实现客流的科学引导，在刑侦领域还可以减轻公安干警抓捕过程中长期蹲守的负担。同时客流统计系统可以作为一个基本平台来实现人脸识别等拓展功能，从而大大增强其扩展实用价值。但目前已经存在的统计方法和系统在长时间工作和复杂情况下体现出了不同的缺点，实用性和可靠性存在不同程度的问题，难以胜任当前精确化、数据化的计数要求。

针对目前系统存在的问题，本文设计了一款基于最新跟踪和检测算法的新型在线客流统计系统。主要工作和成果包括以下几个方面：

分析了常用的跟踪算法和检测算法，重点对系统具体可能使用的候选跟踪算法和检测算法进行了介绍，并使用系统工作环境下提取的测试视频对算法进行了检验和筛选，选出了 KCF 跟踪算法和 SSD 检测算法作为本文的核心算法。对 SSD 的检测器进行了重新优化训练。

针对本文中的情景，以及核心算法的特点，对选用的检测和跟踪算法进行了整合重构。KCF 跟踪算法实时性好，但准确性相对较差，SSD 检测算法则相反。本文采用定期检测并将检测目标和实时跟踪结果匹配的方法，实现了两种方法的优点相结合。

实现了整个系统的基本功能，并进行了改进升级。设计了系统的整体结构和系统的匹配算法、人数统计算法、多线程接口，完成了系统基本功能的实现。在长期测试之后对系统的进一步改进，主要有系统的可视化界面设计、算法改进和稳定性设计。

与一款市场上的产品效果进行了离线视频的效果对比，并验证了本系统的在线实时性。本系统的准确性比对比产品高出 34.6%，同时对人数不超过 5 人的画面处理速度达到了 30fps，满足了实时性的要求。最后，经过长时间实时视频流测试，在待处理图像的队列积压 300 张图像的情况下，在画面人数减少到 3 人以下后，可以在 3s 时间内恢复到队列中只有 3 张待处理图像。

关键词：客流统计；目标跟踪；深度学习；SSD 算法；KCF 算法

Abstract

The pedestrain counting system's basic function is to accurately count the number of people cutting through a test line. With the real time number of pedestrain, we can process scientific management on shops and companies, make effective guidance of huge flow of people in public area, or release the burdon of the police to monitor the movement of suspectors. Also, the pedestrain counting system can be used as a basic algorithms platform to achieve extra functions like face detection,which will considerably increase the utility of the system. The products in the market now show varies of problems in long-time working and complex environment performance, and so still can't meet the requirment of .pricise and stability.

Aiming to solve the problems of today's pedestrain counting systems, a new type of system is designed in this paper. The main work and achievements cover following:

The theories of core algorithms used in this system are introduced, especially the available ones. This paper used relative videos to test and select from these algithms.KCF and SSD are selected as the core algorithms. The training process of detector is also covered in this paper.

According to the situation in this paper and the characters of core algorithms, integration and reconstruction are made to the selected algorithms. KCF meets the requirement of real-time operation, but has lower accuracy, and SSD shows opposite property. This paper uses the strutegy of matching the targets of detection and tracking regularly to combine the advantages of two methods.

the detailed implementation of the whole system is proposed, update of the system is also did later. The overview of system structure is in the first place, then the implementations and procedures of matching algorithms, counting algorithms, and multi-threading design are made to accomplish the basic functions. After long-time testing, further improvements to this system are made, including visual design, algorithms's improments and robust design.

Comparisions was did with another system in the market. In an off-line test comparing with a similar product in the market, this system's accuracy went up by34.6%,and speed reached 30 fps for image with no more than 5 people, satisfying the requirment of real-time processing.After long-time on-line test,we get the result that when the queue went up to 300 images, the system can process them and shorten the queue to the length of 3 in 3s when there are less than 3 people in the images.

Keywords: client counting, tracking, deep learning, SSD algorithm, KCF algorithm

目　录

第 1 章 绪　论

1.1 课题背景及研究的目的和意义

当前社会，机器的自动化和智能化正在加速。随着计算机技术的普及和发展，针对特定方面应用，现代通信与信息技术、行业技术、计算机网络技术、智能控制技术开始实现功能集成，机器正在取代人工实现越来越多的繁重工作。在这波浪潮下，本文设计了一种基于计算机视觉的新式客流统计系统，用来替代人工统计的繁重工作，并带来准确性的大幅提升和成本的降低。

客流统计系统的基本功能是统计通过一条检测线的人数。在商铺、酒店、办公场所以及旅游景点等区域有广泛的需求和应用。通过精确统计客流的实时流量，可以实现对店铺、公司等的科学管理，改善购物和工作环境，合理调动人力资源，估计品牌和地段的价值，帮助管理者实施正确的决策。也可以用于指导客流量较大的场所实现客流的科学引导，有效控制人流的密度，防止踩踏等事故的发生。在安全和刑侦方面，通过对需要蹲守地点的房间门进行检测，可以在犯罪嫌疑人进出房门时第一时间得到消息，减轻公安干警长期蹲守的负担。对于公交车、地铁、铁路运输等公共交通系统而言，详细地掌握实时的各线路站点客流量情况，对于线路的调整和车辆资源的合理应用也有着极大的参考价值。因此对于更快更准确的客流统计系统的开发很有必要。但传统的统计方法在长时间工作和复杂情况下体现出了不同的缺点，实用性和可靠性存在不同程度的问题，难以胜任当前的精确化、数据化的计数要求。

本项目基于对图像提取、检测、跟踪等算法优点的综合应用，设计了一种新的客流统计系统的架构，和传统产品相比，在实时的基础上，实现了室内环境下客流统计的准确性得到大幅提升，并可以对同一画面中多个指定区域，分别进行进出门的客流量统计。本系统的算法决定了系统对于统计对象有灵活处理的能力，通过简单的更换模型文件就可以实现对人以外的其他任意对象进行检测和统计。为了增加系统的商业化价值，系统进行了可视化界面设计，增强了人机交互性能，添加了统计数据的科学展示和图片的输出模块。系统在实现功能之后还进行了长时间的稳定性测试，对容易引起系统长时间运行发生崩溃现象的模块进行了修改。最后，本系统可以作为其他很多扩展功能的平台，如在行人检测的基础上可以提取图像做到实时人脸识别，在安保、刑侦等领域具有很强的实用价值。

1.2 客流统计系统的研究现状及分析

客流统计系统主要分为传统统计方式和机器视觉方式两种。传统方法主要使用人工和一些简单的机械结构或者传感器实现，而机器视觉方式主要依赖于基于计算机的数字图像处理技术。本节将对两种方式进行展开介绍。

1.2.1 传统客流统计方式

传统的客流统计方式受到当时技术和硬件的限制，以及出于成本的考虑，主要有人工统计、红外线感应统计、辊闸方式和重力感应方式等。

人工统计即为由人眼目测来统计客流量，这种方法存在长期运行效率低，成本高，在客流较大的情况下误差大的弊端。红外感应客流统计设备可以分为：红外对射方式、红外反射方式等设备，其功能的主要的实现方式是：从红外感应区域经过的人体会切断或阻挡红外线使其电阻发生变化，或是通过检测人体发出的特定红外线来直接判断人体数量。这种方式安装方便，成本适中，但无法统计客流进出方向，容易受到干扰，容易漏检。三辊闸方式主要采用机械方式，行人进入相关场所需要经过翻滚闸口，翻滚闸正向或反向滚动一次，由此记录一个进或者出的人员，这种方式统计准确，能判断方向，但会阻碍通行。重力感应主要是在地板上安装重力感应装置，当人体踩到重力感应的地板时，计算客流人数，但这种方式应用场合受限，无法对方向进行判断，并且目前成本很高，可靠性差。

几种传统方法各自具有的优缺点如表 1-1 所示：

表 1-1 传统客流统计方式优缺点

统计方式	优点	缺点
人工统计	短期效率高，成本较低	长期运行效率低，成本高，稳定性差
红外线感应	成本适中，安装方便	无法统计客流进出方向，容易受到干扰，容易漏检
辊闸方式	准确率高，能判断方向	阻碍通行，影响美观
重力感应方式	不影响美观	安装要求和成本高，可靠性差

1.2.2 基于机器视觉的产品现状

随着计算机技术的不断进步，机器视觉技术也得到了持续的发展。从产生的时间来看，计算机视觉的概念产生于二十世纪中叶，它当时的主要工作内容是进行较为基础的统计模式识别。经过长时间的积累进步，该技术包含的内容明显丰富，并发展成为一门独立的学科，被广泛应用到了很多领域[1]。计算机视觉技术目前已经实现使用多种滤波和特征点提取等方法，对图像中的目标进行识别和提取，在很多环境中可以取得非常好的效果。目前市场上已经出现了使用摄像头的客流统计系统（如图 1-1）。在显示画面中确定一个检测框或者检测线，在画面中确定出行人的个数和位置，分别对每个行人的轨迹进行跟踪和记录，并在画面和数据栏显示。在轨迹通过检测线或者检测框的时候进行判断方向并计数，从而实现客流统计的功能。

图 1-1 基于机器视觉的客流统计产品效果图

使用计算机或者嵌入式设备对摄像头采集的图像进行处理得到客流信息的统计产品是一种新型的客流统计产品类型。与传统方式相比，使用机器学习方法具有成本较低，准确性很高，能应对复杂情况，体积较小，不影响美观和通行以及方便数据存储和调用等优点。但目前市场上的基于计算机视觉的客流统计系统很难找到实际的效果展示视频，而展示视频也往往只对应背景光线不强烈，画面中的人数不多的情况。基于机器学习的视频客流统计系统按使用的摄像机类型，可以分为单目客流统计和双目客流统计两种。

目前绝大多数的视频客流统计都是单目客流统计，单目客流统计的最大优点就是图像预处理算法简单，设备成本低，而对硬件计算能力的需求也相对较低。由于单目相机得到的坐标数据限制在二维，无法对物体的距离和高度进行判断，实际的准确度也就会比较低，一般能达到 80%左右，在背景发生较大变化的情况下则会降低到 60%或以下。但由于算法相对简单，这种方式可以处理的图像分辨率也就更大，覆盖的范围也可以更广。目前市场上和实际安装的大部分摄像头都是单目摄像头，所以单目相机的客流统计系统的安装和移植成本也大大降低。

双目客流统计，使用了两个相同的摄像机镜头，类似人的双眼，两个摄像头取得的图像，经过一系列的计算，得到合成的图像。在双目客流中，最大的优势就是获取了实际场中的第三维信息，如物体的距离和人的高度信息，原理的示意图如图 1-2。因此，只用检测高度在人的合理范围之内的高度图像，因为人的头部往往在最高位置，所以很容易就可以取得人的位置信息，而且有效排除灯光，阳光等干扰。由于可以判断每个物体的距离，双目摄像机可以解决目前单目相机的跟踪算法很难处理的目标之间的遮挡问题。但这种系统安装硬件成本较高，同时与目前街道和室内占大多数的单目监视摄像机没有兼容性，需要增加更换摄像机才能使用，性价比不高，会造成硬件的浪费。

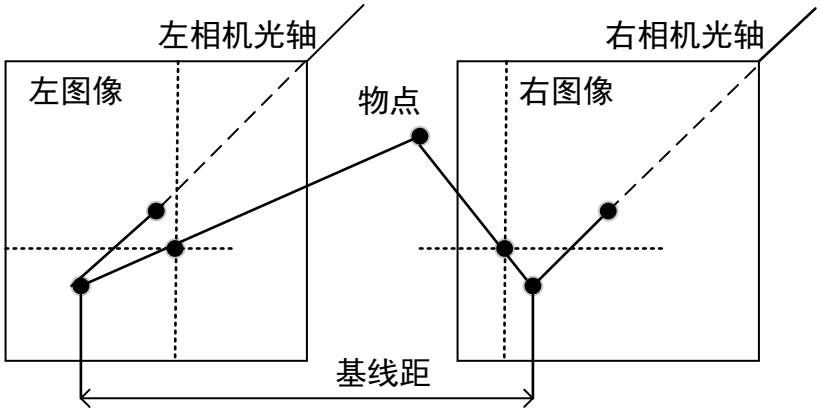

图 1-2 双目定位原理示意图

目前的客流统计产品常用的一种方法是对热成像摄像机的图片实施目标跟踪算法。这种方案具有硬件成熟，对设备要求低，可直接做成嵌入式系统或者一体化系统的优点。但是实际由于可能无法识别热源类型，且无法分辨距离较近的个体，统计的准确度相对较差。另一种常用的方法是采用传统视觉方法进行图像分割和目标检测，之后进行跟踪。这里常用的检测方法有模板匹配、背景建模等，但无论准确率还是速度都不能令人满意，模型也一般不具有普适性。关于这些方

法在 2.3 节中进行了详细的论述。

总的来说，由于采用的算法和设备的局限性，现在基于机器视觉的产品往往也很容易出现漏检和跟丢等情况。

1.3 本文的主要研究内容

1.比较和筛选不同算法

传统的检测算法都是基于经典数字图像处理的图像分割和机器学习分类算法，效率很低，效果也很难达到实际应用的标准。近年来提出的的基于深度学习的检测算法主要有：R-CNN 系列、SPP-Net、YOLO、SSD(Single Shot MultiBox Detector)等；本项目中可能使用的跟踪算法主要有：光流法、基于特征的跟踪算法、基于相关滤波器的跟踪算法。总的来说检测算法的识别正确率很高，但在目前的硬件条件下，速度无法满足实时性要求，而跟踪算法速度很快但需要给出初始目标，并存在漏检、错检问题，同时往往速度和准确性之间存在明显的矛盾。因此本项目将进行实际测试，选择效果较好的跟踪和检测算法进行结合，采用合理的程序框架，在准确率和实时性之间找到一个最优点。

2.算法库的封装和检测器的训练

相关论文提出的算法对应的代码往往是一个示例程序，且使用的编程语言和环境也多种多样，无法直接在项目中使用。基于深度学习的检测相关的示例中的分类器（神经网络）往往对应很多类别以便针对实际情景进行选择和处理，而本系统中实际上只需要分为人和背景两大类，需要选择一个浅层网络，并对参数进行调整。因此，需要将示例程序中的算法部分摘出并封装为库，以便在项目主程序中调用。

由于本项目使用的检测算法将基于深度学习理论，所有参数都是自适应的，需要使用合适的样本和方法对检测器进行训练，并引入监测机制和预防方法防止过度训练。

3.监控视频编解码和实时图像的获取

本项目将使用网口接收监控摄像头的信号。由于按照现有的视频标准传输数据，现有的网络带宽远远不能满足要求，这里采用了 H.265 视频压缩技术保证高清图像的实时传输。摄像头采集到的视频为 RGB24 格式。第一次压缩后转换为 YUV 格式，进而打包转换为可以通过网络传输的格式进行传输。到达计算机后

需要进行流解码首先获得 YUV 格式图片。之后需要再一次解压缩成为方便进行图像处理的 RGB 格式，并保存为 OpenCV 常用的 Mat 数据型，以便进一步处理。

4.整体系统的设计和优化

本项目的核心是整体系统的设计。本项目首先需要解决如何将检测和跟踪算法结合的问题，进而需要设计一个能够利用检测和跟踪准确计数通过检测线的人数的方法，最终需要解决图像和统计数据的显示问题，以实现所需的功能。

同时，从本项目预期实现的功能来看，本系统还需要能够连续工作几天甚至几个月，因此对代码的强壮性有很高的要求；本项目需要对传入的视频图像进行实时处理，因此在实现基本功能的基础上，本项目还需要对系统做稳定性和实时性的优化。在实现基本功能之后，本项目还需要继续开发系统的拓展功能，提高系统的商业价值。

1.4 本文的章节安排

论文共分六章，除绪论外各章内容安排如下：

第二章以特征提取方法为前导，介绍了目前具有代表性的跟踪算法，并重点对基于相关滤波器的算法进行了介绍。

第三章首先介绍了目标定位算法，之后分三大类介绍了当前主流的各种检测算法，重点介绍了基于深度学习的检测算法。

第四章详细介绍了整个系统基本功能的实现。介绍了系统使用的软件环境和硬件设备和视频解码的解决方案，给出了系统的解码模块构造。在系统设计上，首先给出了系统的整体结构，之后分别对系统的匹配算法、人数统计、多线程设计进行了介绍。在长期测试之后对系统进行了系统的可视化界面设计和稳定性设计，给出了最终成型系统的结构。

第五章给出了对检测器的训练过程和对跟踪和检测算法的实际测试和筛选过程，对系统性能进行了测试，最后与一款市场上的产品效果进行了对比。

1.5 本章小结

本章介绍了项目的背景和意义，同类产品的发展情况，以及本项目开发的现实需求和意义。同时给出了本项目开发的主要任务和整体框架，以及项目采用的实现方法对同类产品的优势。

第 2 章 基于相关滤波器的跟踪技术

2.1 引言

本文中系统基于室内固定的单目摄像机拍摄的实时画面运行，这种情景具有：画面内范围小，需要处理的数据相对较少；短期内背景变化小，不受户外天气等因素影响；长期运行下，背景和前景具有一定的周期性变化的特点。基于项目的特点和需求，本文选择跟踪算法作为多数帧的处理方法，用来得到行人的运动轨迹，对于算法有着实时性好和准确度高的基本要求。目前世界上每年都在提出大量跟踪算法，但基本都需要用到特征提取和匹配的算法。本章首先对特征提取和匹配的方法进行了介绍，之后给出了几类跟踪算法的原理，最后对选用的基于相关滤波器的跟踪算法进行了详细的阐述。

2.2 特征检测

2.2.1 图像卷积的定义

由于连续帧中的同一目标始终是变化的，前后目标匹配需要寻找目标的不变特征。传统算法中最重要的不变特征就是角点。最近新兴的一种基于神经网络的特征提取方式打破了传统的特征提取模式，具体内容在第三章有详细的论述。

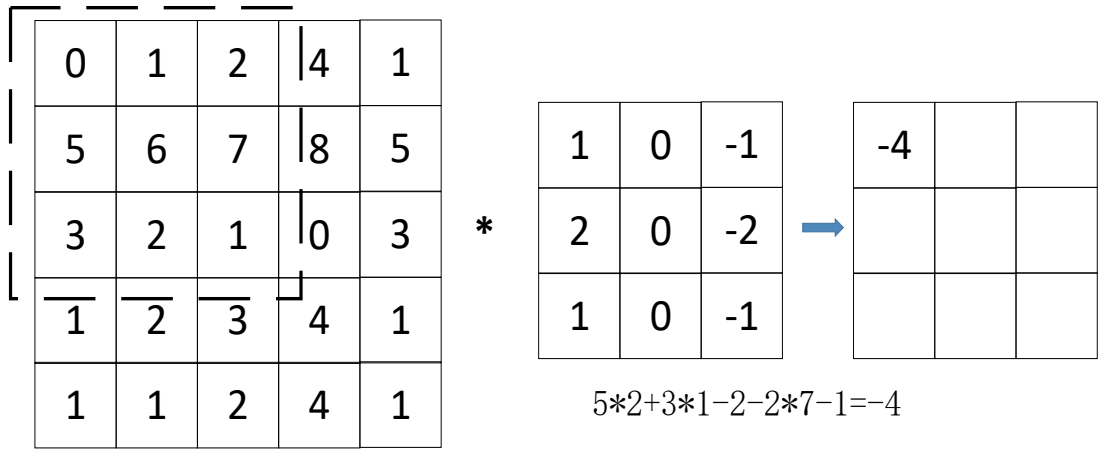

图 2-1 图像卷积算法示意图

由于在本文的算法中被频繁的使用，在开始介绍各种特征提取方法之前，需要简单介绍图像中的卷积操作。在图像中，卷积反映了在空间不同位置的输入信

号在滤波器的作用下，对输出影响的叠加结果。在空间域中，图像区域 $f(i,j)$和滤波器 $g(i,j)$的离散卷积运算定义为：

$$f * g = \sum_{a,b} f(i-a, j-b)g(a,b) \tag{2-1}$$

为了简便使用，直接使用对称将式子内部乘积项简化为了$f(a,b)g(a,b)$，即图像和滤波器对应位置的像素灰度值相乘并求和作为结果，如图 2-1 所示。

2.2.2 角点检测

角点是图像中的极值点，可以认为是图像中灰度剧烈变化的点，一般对应两条线的交点，属于物体运动中可以取得的不变特征，也是目标匹配和建模的重要方法。常用的角点提取方法有 Shi-Tomasi，Harris，SIFT 和 SURF 等，其中 Harris 方法基于相邻像素的灰度变化，算法简单高效，但对于尺度变化、光照等情况处理能力很差。SIFT 和 SURF 方法不仅具有尺度不变性，还能在旋转、亮度变化等情况下保持较好的效果，但计算复杂度相对较高，有时检测到的特征点较少，对光滑处理过的图像效果较差。这里以效果最好的 SIFT 检测为例进行介绍。

高斯卷积核是能做到尺度变换的线性核，在 SIFT 中，用不同尺度下的高斯差分核和图像卷积，得到了高斯差分尺度空间。

$$\mathrm{D}(x,y,\sigma) = \big(\mathrm{G}(x,y,k\sigma) - \mathrm{G}(x,y,\sigma)\big) * \mathrm{I}(x,y) \tag{2-2}$$

即建立了同一幅图像在不同尺度下的特征集合。在寻找角点的时候，包括中心点(x,y)的八邻域，以及两个相邻尺度下同一位置各 9 个点都被算入，只有一个点在 DOG 空间 26 个邻域中是最大或者最小时，才被确认为角点[2]。

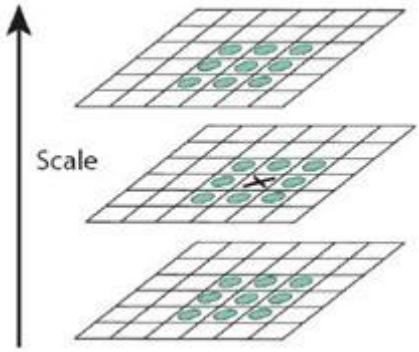

图 2-2 DOG 空间邻域

为了保证旋转不变性，SIFT 使用了关键点描述子的概念。以每个角点为中心，取 16×16 区域内的所有梯度值，分为 4×4 的小格，对所有梯度值取八个方

向上的累加值，SIFT 都把八个方向的梯度信息保存下来，故每个描述子保存了 4×4×8=128 个特征，最后将每个特征点的主轴方向都转到同一个规定的方向，从而使特征对于旋转情况有很好的处理能力。在对描述子进行归一化后，即可减轻光照等因素的影响。

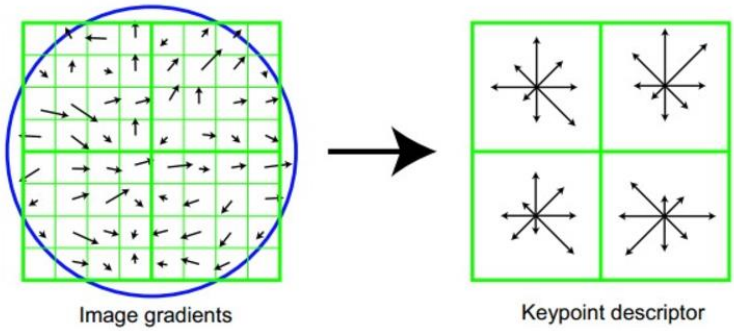

图 2-3 特征描述子的建立

在完成特征点的提取之后，需要使用特征匹配方法将不同帧中的相同特征点进行匹配，最常用的特征匹配方式就是距离匹配。主要有欧氏距离、绝对值距离和切氏距离三种，分别对应几何距离，坐标绝对值的差之和以及两个坐标距离最大方向上的距离。距离匹配对于变化较小的情况非常实用，快速有效，因此 Harris 角点的匹配可以直接使用距离匹配。

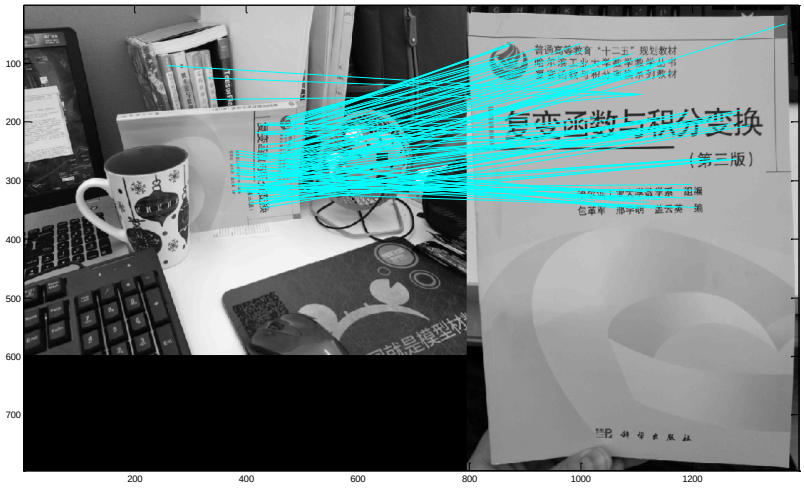

图 2-4 SIFT 角点匹配效果

SIFT 的描述子中考虑了旋转的因素，因此在匹配的时候是对 4×4 个角点分别用优先 k-d 树找到最相近的角点，然后对角点进行匹配，取出最近的距离除以第二近的距离，如果小于一个阈值，就认为是一对匹配角点。阈值取的越小，找到的角点越少。

2.2.3 HOG 特征

以 HOG（方向梯度直方图，Histogram of Oriented Gradient, HOG）为代表的特征是另一类常用的图像特征。这类特征提取方法不是以像素为参考对象，而是对选取的局部图像进行整体研究，进而提取特征。这里以效果最好的 HOG 特征为例进行介绍。

首先将图像的三通道分别做归一化处理，减少光照的影响,然后求出图像的选取范围内每个像素的梯度大小和方向。在 x，y 方向上的梯度大小可以分别用相邻像素的灰度差表示，则每个像素的梯度大小和方向分别表示为：

$$G(x,y)=\sqrt{G_x(x,y)^2+G_y(x,y)^2}$$
$$\alpha(x,y)=\tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y)}\right) \quad (2\text{-}3)$$

与 SIFT 方法类似，将这个区域分成一个个细胞单元(cells)，对每个 cell 做灰度直方图。将 360 度平均分为 n 个区域，则以这 n 个区域作为直方图每个矩形条的分界。根据经验，每个矩形条的大小直接用范围内的梯度大小做和得到。

定义区域（block）为相邻几个 cell 的集合，使用相邻 cell 的特征向量组合得到。而各个 block 之间有重叠区域，用来提取更多的特征。最终，将选取范围内所有 block 都作为区域的一个特征。特征的数量和质量与 cell 和 block 的大小直接相关。

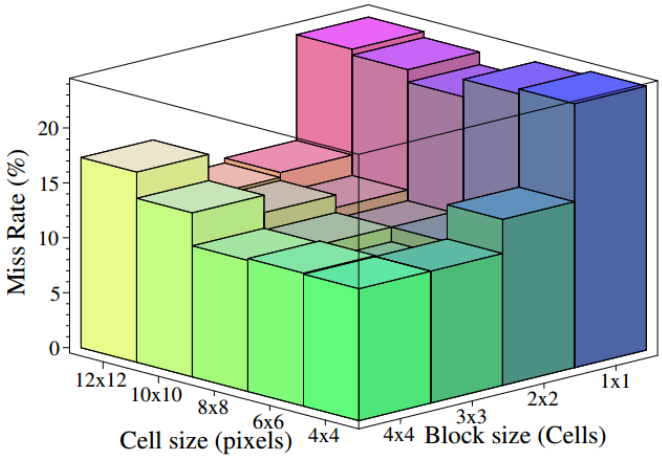

图 2-5 cell 和 block 大小与提取效果关系图

根据论文测试结果，当选择 cell 大小为 6×6，block 为 3×3 时，特征提取的损失最少，效果最好。

2.3 目标跟踪算法

目标跟踪是通过分析视频图片序列，对起始帧标出的候选目标区域实施匹配，定位出这些目标在视频序列中的坐标。简单来说，就是在序列图像中为目标进行定位。跟踪算法的发展历程很长，但所有算法本质上都是从后续的帧中标出与真实目标最接近的区域，因此所有算法都会有预测和定位，以及对相似度的判断机制。目标跟踪算法是目前计算机视觉领域一个焦点，在虚拟现实、智能监控、人机交互、机器感知等领域中有着重要的研究与应用价值[3]。

1.光流法

光流法是基于角点的跟踪算法。在使用 2.2 节的内容取出图像的角点之后，光流法的策略是用图像上的二维运动分析物体的真实三维运动。光流法需要假设前后帧之间的亮度是一致的，且物体没有出现较大的运动

根据一般常识，物体在相邻帧的运动是连续的，设 t 时刻物体上一点对应图像上的像素灰度为 I(x,y,t)，则经过△t 之后，对应物体上同一点的像素移动了(△x,△ y)，灰度值表示为 I(x+△x,y+△y,t+△t),根据定义，有：

$$\mathrm{I}(x,y,t)=\mathrm{I}(x+\triangle x,y+\triangle y,t+\triangle t) \tag{2-4}$$

经过泰勒展开和忽略高阶微小量之后，可以得到光流约束方程：

$$-\frac{\partial E}{\partial t}=\frac{\partial E}{\partial x}\frac{dx}{dt}+\frac{\partial E}{\partial y}\frac{dy}{dt}=\nabla \mathrm{E}dv \tag{2-5}$$

由上式可得，物体上一个点对应图像上灰度随时间的变化可以用场景的灰度变化率与这个点的运动速度的点积来表示。在根据场景加入其它的约束条件之后，就可以对移动的分量进行定量，从而实现对目标点的跟踪。

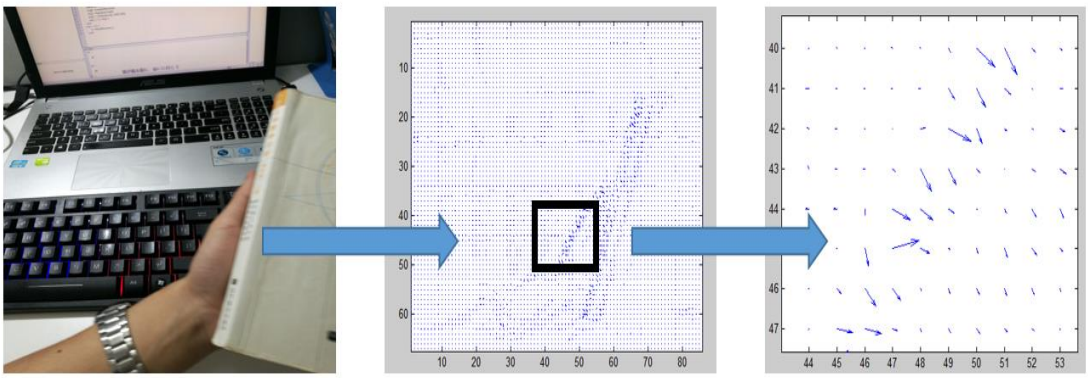

图 2-6 光流法跟踪效果

后来出现了改进版的 L-K 金字塔光流法，在多个尺度的图像上取相邻点，能够应对尺度变化和灰度改变。但总体来说，光流法对灰度改变敏感的情况没有实质改变。光流法跟踪的效果如图 2-6 所示。中间图为跟踪目标的矢量图，反映了所有特征点的运动方向，右图为局部放大后的图像。

2.基于相关滤波器的跟踪方法

基于相关滤波器的算法一般可以分为以下几步：第一，在起始帧取一些样本训练一个回归器，用来判断一个窗口中对是否有跟踪对象的响应。第二，在接下来的每一帧中都在前一帧的目标附近采集窗口，用回归器判断每个样本的响应。最后，用非最大值抑制法取出响应最强的窗口作为目标位置，并更新回归器。

常用的相关滤波器是 MOSSE 相关滤波器。由卷积定理可知，空间域的卷积计算可以转换为频域中的乘积形式。在本文中，空间域变量使用小写字母表示，频域变量使用大写字母。用 G_i 表示滤波器的输出，F_i 表示滤波器的输入，则 MOSSE 滤波器在频域中的表达可以写成：

$$\mathrm{H}_i = \frac{G_i}{F_i} \tag{2-6}$$

在确定滤波器公式时，首先从目标中提取多个图像训练组,分别记为 $f_1, f_2, ...f_t$。以它们作为训练样本，得到的滤波器响应是一系列指定的高斯函数 $g_1, g_2, ...g_t$，算法的最终目标就是找到一个相关滤波器参数的最优配置，使得算法的均方差ε最小：

$$\varepsilon = \sum_{j=1}^{t} \left\| h_t \cdot f_j - g_j \right\|^2 = \frac{1}{MN} \left\| H_t F_j - G_j \right\|^2 \tag{2-7}$$

其中等式第二项左侧是空间域的方程式，最右侧是转换到频域后的方程式，正是这个等式，使我们得以将问题变换到频域里求解。滤波器在频域里的解如下：

$$H_t = \frac{\sum_{j=1}^{t} G_j F_j}{\sum_{j=1}^{t} F_J F_j} \tag{2-8}$$

一般来说，g_j 可以是任意形式的输出，本算法中使用的输出 g_j 是峰值位于目标中心的二维高斯型函数，从而使得后续帧可以直接使用图像的峰值作为目标位置。这个方法的好处在于：一是运算简洁，基本都是矩阵运算；二是引入快速傅里叶变换(FFT)，大大提高了运算效率。这种算法满足了对实时性的需求，这也是基于相关滤波器的算法快速性的主要原因[7]。

在得到上述的相关滤波器后，对于新一帧中输入的样本集 Z, 求出相关性得分 y，让 y 取最大响应值时对应的位置 z 作为新的目标位置。

$$\mathrm{y} = \mathcal{F}^{-1}(H_t Z) \tag{2-9}$$

由于图像中的目标是有位移和形变的，对于当前帧 j，在每次找到新的一帧 j+1 中目标位置之后，需要对滤波器模型进行更新,更新的公式如下：

$$\begin{aligned} \mathrm{H}_{j+1} &= \frac{A_{j+1}}{B_{j+1}} \\ \mathrm{A}_{j+1} &= \eta \mathrm{G}_{j+1} \odot F_{j+1} + (1-\eta) A_j \\ \mathrm{B}_{j+1} &= \eta \mathrm{F}_{j+1} \odot F_{j+1} + (1-\eta) B_j \end{aligned} \tag{2-10}$$

2.4 基于相关滤波器的跟踪算法

目前效果最好，发展最迅速的跟踪算法是基于相关滤波器的算法系列。经过对资料的整理，最终筛选出了两种可能的跟踪算法作为本文的候选算法，分别为 KCF 和 DSST，两者都是基于相关滤波器的目标跟踪算法的改进，提高了精度，减少了计算量。两者分别为 2014 年算法的第三名和第一名。但由上一节可知，每个相关滤波器都只有一个峰值，因此只能实现对一个目标的跟踪。在需要多目标跟踪的情况下，必须使用多个跟踪器，造成计算时间的线性增长，这也是实际应用中制约跟踪速度的最重要因素。

2.4.1.KCF 跟踪算法

KCF（Kernel Correlation Filter,核相关滤波算法），是一种鉴别式追踪方法，这一类方法的一般步骤是在跟踪的同时用目标画面做样本训练一个目标检测器，使用这个检测器判断在下一帧各个预测位置是否存在跟踪目标，再用新的目标图像作为样本更新训练集，进而更新目标检测器。在训练目标检测器的时候定义目标区域为正样本，目标周围的区域为负样本，并认为越靠近目标的区域内正样本出现的概率越大。KCF 算法会更注重于图像的负样本采集，算法把正样本以不同的位置和尺度变换来得到负样本。算法产生大量的负样本，之后使用分类器学习这些负样本，就可以让分类器得到各种条件下的评估先验知识。

KCF 算法有几大重要的特点。第一，算法使用循环矩阵采集周围区域的正负样本，利用脊回归方法训练相关滤波器，之后用循环矩阵在傅里叶空间用对角化

的性质将矩阵运算转化为向量的 Hadamad 积（元素的点积），从而大大降低了运算复杂度，提升了运算速度，从而使算法超过了实时性要求。第二，算法将线性空间的脊回归通过核函数映射到非线性空间，通过求解一个对偶问题和常见的约束，这个问题同样可以使用循环矩阵在傅里叶空间的对角化来简化运算。第三，它给出了一种将多通道数据融入算法的方法[8]。

KCF 跟踪算法的概要流程图如图 2-7 所示：

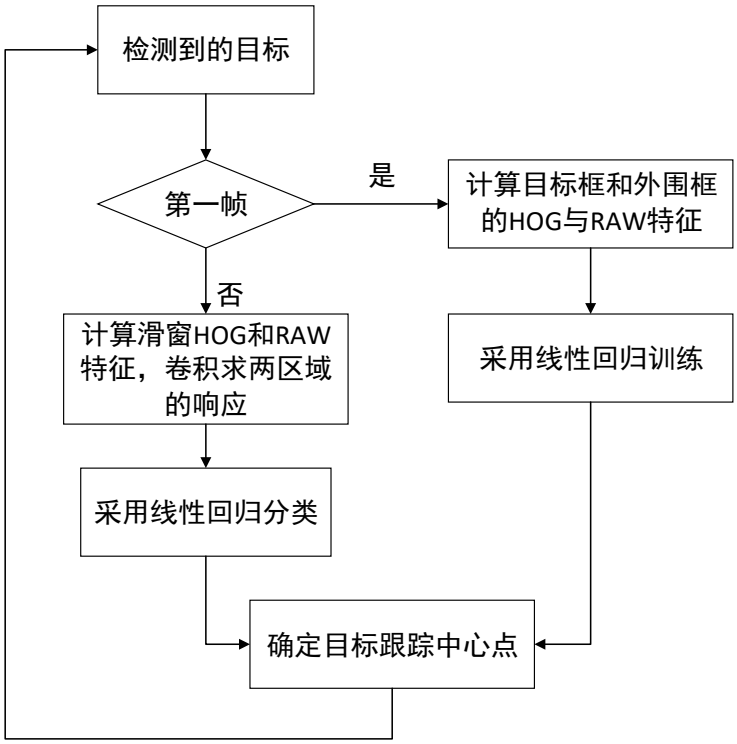

图 2-7 KCF 跟踪算法概要流程图

KCF 将跟踪问题化为了一个线性回归模型，令目标输入向量为 x，输出为 $f(x)=w^{T}x$,则算法训练的要求就是找到一系列的权重 w，使得输出与期望值 y 之间的差最小，需要满足：

$$min(\sum_{i}(f(x_i-y_i)^2+\lambda\|w_i\|^2) \tag{2-11}$$

其中，λ 是为了防止训练过度的 L1 正则项系数，这里使用了脊回归的策略，在 KCF 的文章中给出上述问题的解为：

$$w=(X^{H}X+\lambda I)^{-1}X^{H}y \tag{2-12}$$

其中 X^{H} 是 X 矩阵的共轭转置。这个式子本身求解计算量很大，对速度影响很严重，文章中引入了循环矩阵的方法来加速计算。（2-13）中的 X 为一个循环

矩阵，构造方式为：

$$X = C(x) = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_n & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \cdots & x_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \cdots & x_1 \end{bmatrix} \tag{2-13}$$

每一行都是由样本的特征向量元素循环构成。由循环矩阵的性质，X 可以分解为：

$$X = Fdiag(\hat{x})F^H \tag{2-14}$$

其中$\hat{x}$由 x 向量的傅里叶变换得到，F 为离散傅里叶矩阵：

$$F = \frac{1}{\sqrt{K}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 \\ 1 & w^2 & w^4 & w^6 \\ 1 & w^3 & w^6 & w^9 \end{bmatrix} \tag{2-15}$$

且 F 矩阵满足：

$$FF^H=F^HF=I \tag{2-16}$$

由循环矩阵的性质，式（2-16）可以写成：

$$w = (Fdiag(\hat{x} \odot \hat{x}^*)F^H + \lambda Fdiag(\delta)F^H)^{-1}X^H y \tag{2-17}$$

利用矩阵性质，可以整理为：

$$w = C(F^{-1}(\frac{\hat{x}^*}{\hat{x} \odot \hat{x}^* + \lambda\delta})) \tag{2-18}$$

可以解出 $F(w)$:

$$\mathcal{F}(w) = \frac{\hat{x}}{\hat{x} \odot \hat{x}^* + \lambda\delta} \odot \mathcal{F}(y) = \frac{\hat{x} \odot \hat{y}}{\hat{x} \odot \hat{x}^* + \lambda\delta} \tag{2-19}$$

利用上式，经过傅里叶逆变换即可求得 w。利用傅里叶变换简化参数最优解的求解是 KCF 的核心，之后文章还引入了核来解决非线性空间的问题，即将问题的求解引入一个新的空间。将 w 用 x 和一个对偶空间 α 的线性组合表示。在非线性情况下，输出与输出之间的关系表示为：

$$\begin{aligned} f(z) &= w^T\varphi(z) \\ \mathrm{w} &= \sum\nolimits_i \alpha_i \varphi(x_i) \end{aligned} \tag{2-20}$$

$\varphi(z)$为输入 z 的非线性变换，$\varphi(x)$为训练样本的非线性变换。则合并为：

$$f(z) = \sum\nolimits_i \alpha_i \varphi(x_i)\, \varphi(z) = \alpha^T \varphi(z)\varphi(x) \tag{2-21}$$

令$\mathrm{k}(z,x) = \varphi(z)\varphi(x)$,这里的 $k(z,x)$即称为为核，则原式可以写成：

$$f(z) = \alpha^T k(z,x) \tag{2-22}$$

f(z)为输入的非线性变换结果，但 f 相对于核是线性的，再利用循环矩阵的性质，解出 α 为：

$$\begin{aligned} \alpha &= (K + \lambda I)^{-1} y \\ &= (F diag(\hat{k}) F^H + diag(\lambda\delta) F^H)^{-1} y \end{aligned} \tag{2-23}$$

利用解线性问题时同样的变换方式最终可得 α 的傅式变换为：

$$\hat{\alpha} = \left(\frac{1}{k + \lambda\delta}\right) \odot \hat{y} = \frac{\hat{y}}{\hat{k} + \lambda\delta} \tag{2-24}$$

KCF 的跟踪速度很快，超过了实时的速度，但为此牺牲了重检测的时间，因此框选的精度可能受到影响，同时也无法判断目标是否已经跟丢。当目标丢失时，跟踪框会停留在丢失的位置。

2.4.2.DSST 跟踪算法

DSST(Accurate Scale Estimation for Robust Visual Tracking)夺得了 2014 年 VOT 比赛的第一名，性能优异，算法简洁，与上一小节所述的 KCF 都是基于相关滤波器的算法。这篇文章提出的算法是基于 MOSSE 的改进。

DSST 算法设计了两个不同功能的相关滤波器，分别实现目标的位置跟踪和尺度变换跟踪，分别命名为位置滤波器（translation filter）和尺度滤波器（scale filter），前者用来确定新一帧中目标的位置，后者用来应对目标发生尺度变换的问题。两个滤波器相互独立，可以分别选择不同的特征种类和特征计算方式进行训练[9]。该方案的亮点是尺度估计的方法可以移植到其他算法中去。

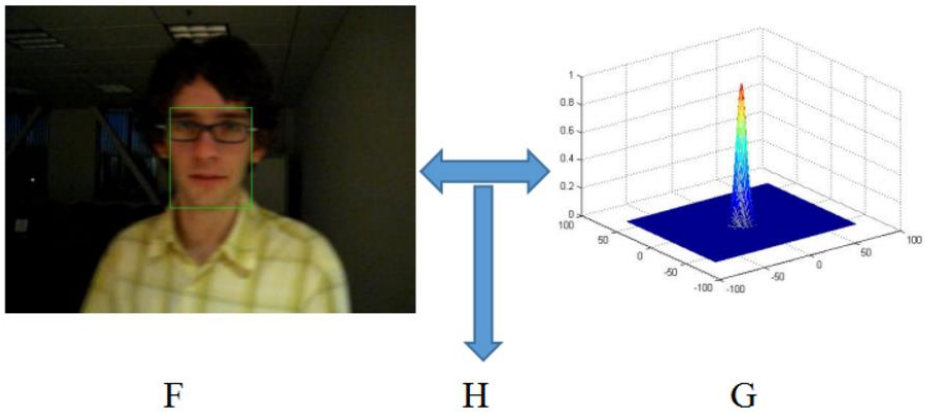

图 2-8 相关滤波器示意图

DSST 算法的内容简洁明了。如图 2-8 所示，通过左侧的图像训练组的目标可以提取特征 F，在论文中使用的是 HOG 特征，右侧是人工设定的高斯型函数 G，使用式（2-6）可以得到一个相关滤波器 H。把从下一帧的图像训练组提取的特征组 Z 作为输入，通过相关滤波器 H 按照式（2-7）计算，把得到响应值 y 最大的特征作为候选目标。具体的相关滤波器构建过程中，算法把输入信号（从下一帧提取的一个训练组）转化为一个 d 维的特征向量（比如 SIFT，HOG 等特征），建立最小代价函数来得到最优参数配置。代价函数如下：

$$\varepsilon = \left\| \sum_{l=1}^{d} h^l \cdot f^l - g \right\|^2 + \lambda \sum_{l=1}^{d} \|h^l\|^2 \tag{2-25}$$

其中，l 表示特征向量的维度，λ 为 L2 正则项系数，作用是防止 f 频谱的权值过大，导致式（2-26）中的频域解分子为零。

$$\mathrm{H}^l = \frac{\overline{GF^l}}{\sum_{k=1}^{d} \overline{F^k} F^k + \lambda} = \frac{A^l{}_t}{B_t} \tag{2-26}$$

每次训练每个像素点都要求解 d×d 维的线性方程组，计算复杂度很高，为了提高速度，算法采用更新上式中的分子 $\mathrm{A}^l\mathrm{t}$，分母 Bt 的策略得到近似结果:

$$\begin{aligned} \mathrm{A}^l{}_t &= (1-\eta)\mathrm{A}^l{}_{t-1} + \eta \overline{G_t} \mathrm{F}^l{}_t \\ \mathrm{B}_t &= (1-\eta) B_{t-1} + \eta \sum_{k=1}^{d} \overline{F^k{}_t}\, \mathrm{F}^l{}_t \end{aligned} \tag{2-27}$$

其中，η 是学习率。在新的一帧中，可以对滤波器相应采用非最大值抑制策略，得到目标位置：

$$\mathrm{y} = \mathcal{F}^{-1}\left(\frac{\sum_{l=1}^{d} \overline{A^l} Z^l}{B + \eta} \right) \tag{2-28}$$

DSST 算法的创新点在于增加了尺度搜索的独立滤波器和目标估计方法。具体操作方法是：先利用位置相关滤波器确定出目标可能的位置集合，再使用尺度相关滤波器，以各个可能的位置为中心，取出不同尺度的训练组，根据得分得到目标最可能的位置和大小。

2.5 本章小结

本章详细介绍了常用的跟踪算法。跟踪算法需要对前后帧的目标进行匹配，所以需要提取边缘、角点等特征，本章介绍了用于边缘提取的 Sobel 算子、M-H 算子和 Canny 算子，以及用于角点检测的 Harris 算子和 SIFT 算子。特征匹配使用的方法主要是向量的距离匹配，对于特征向量过多的情况，可以使用 PCA 等方式进行降维，减少计算量。视觉跟踪算法主要包括光流法和基于相关滤波器的跟踪算法等，本文中均进行了介绍。光流法的基本思想是用物体的真实运动与在图像上的运动做映射，重点捕捉物体特征的运动方向。目前效果最好的算法集中在基于相关滤波器的算法中，基于相关滤波器的跟踪算法的基本思想是寻找一个滤波器，使得在当前帧的输入下，输出的二维高斯函数的峰值能够出现在目标的中心点，并用当前帧的目标更新滤波器。DSST 和 KCF 算法分别为 2014 年 VOT 算法比赛中的一、三名，均是基于相关滤波器的算法。本文对两种算法的原理进行了详细介绍。

第 3 章 基于深度学习的检测技术

3.1 引言

如第二章所述，本文中系统的工作环境相对温和，同时存在长期周期性的变化。本文选择跟踪算法处理多数帧得到对应的轨迹，但由于摄像机拍摄范围有限，画面内行人身份、位置和数量更新很快，需要引入检测算法来及时重新初始化图中的跟踪对象。而且即使在本文的这种工作环境下，跟踪算法仍然不能避免遮挡、炫光等因素造成的跟丢目标现象，需要及时检测进行矫正。目前世界上每年都在提出大量检测算法，但基本的步骤都是一样的。基于项目的特点和需求，本文同样进行了检测算法原理的介绍。

3.2 目标定位

目标的定位始终是目标检测算法的第一步，目标定位在本质上是背景与前景的区分。最简单的目标定位方式即为直接用当前帧图像与一张模板图像做比较，差异较大的像素区域即为目标所在的区域。但这种前景提取方式受到光线和阴影的影响非常大，且当多个目标位置相近或者重叠的时候很难做出区分。

3.2.1.选择性搜索

选择性搜索（selectve search），即基于颜色、纹理、大小等多种特征将图像分割为多个小块，然后逐渐将临近的小块组合，进行区域合并，我们认为最后结果中同一个目标都被分割到了同一个区域。选择性搜索是一种基于区域生长的算法，这种算法用到了图（graph）的数据结构。

图是一种常用数据结构，由顶点集 V(vertices)和边集 E(edges)组成，可以表示为 G=(V,E)。每个顶点 vϵV，在图像领域常常用来表示单个的像素点，连接一对顶点的边表示为(v_i, v_j)ϵE，边的权重 w(*i*,*j*)在图像领域常用来表示顶点之间的距离。在选择性搜索中使用的是无向图，即边对于两侧的顶点是一样的。这里把树(tree)定义为一种特殊的图，树的任意两个顶点之间都有路径相连接，但是没有回路，所以任意两个顶点之间只有一条路径。定义所有连接指定顶点的树中，权重之和最小的为最小生成树（MST，Minimum Spanning Tree），图 3-1 中粗线连接

部分即为一棵最小生成树。在选择性搜索中，每一个像素都作为图的一个顶点，然后逐渐合并，最终得到多个 MST，对应多个分割区域，实际上可以认为分割为了多个森林(forest)。

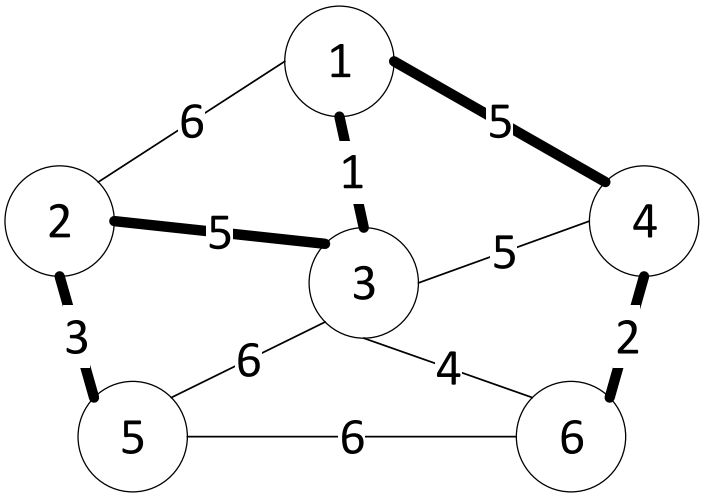

图 3-1 最小生成树示意图

在具体算法的操作中，合并的条件是一个设定的阈值。如果两个区域的差异小于这个值，就认为两个区域属于同一个对象，合并两个区域。如此进行下去，最终得到的多个区域就认为对应不同的对象或背景。由于图像每个部分的特征不同，为了尽量将相似区域合并，需要自适应产生阈值。首先定义一个区域的类内差异 Int(C)：

$$\mathrm{Int}(\mathrm{C}) = \max_{\varepsilon\epsilon(MST,E)} \varepsilon \tag{3-1}$$

定义中使用的是权重最大边的权重，对应 MST 内部最大梯度，或者认为是最大灰度变化，因此也代表了 MST 能允许的最大差异。两个区域的类间差异 $\mathrm{Diff}(C_1,C_2)$定义为：

$$\mathrm{Diff}(\mathrm{C}_1,\mathrm{C}_2) = \min_{v1\in C_1, v2\in C_2, (v1,v2)\in E} w(v1,v2) \tag{3-2}$$

故判断两个区域可以合并的条件即为：

$$\mathrm{Diff}(\mathrm{C}_1,\mathrm{C}_2) \le \min(\mathrm{Int}(\mathrm{C}_1),\mathrm{Int}(\mathrm{C}_2)) \tag{3-3}$$

其中 C_1 和 C_2 是孤立的像素值时，$\mathrm{Int}(C)\equiv 0$，所以算法需要在开始合并的时候预设一个阈值，当生长进行到一定程度之后再去掉。合并后得到的所有块都分别对应一个候选窗口，利用仿射变换可以将这些区域大小变换为分类器需要的输入矩阵尺寸，进行下一步分类。

3.2.2.基于神经网络的目标提取

神经网络是一种典型的自适应参数模型，对各种复杂的情况都有很好的适应性。神经网络的最基本构成单位是神经元。图 3-2 中每个圈代表一个神经元，可以看出，所有的神经元都是多输入单输出，而神经元的输出(也被称为激活值)可以统一表示为：

$$a = f(\sum_i \mathrm{w}_i x_i + \mathrm{b}) \tag{3-4}$$

其中 w 是每个输入对应的权值，b 是神经元对应的偏置。同一级别的神经元组成一层。在前向神经网络中，每一层神经元的输入都是上一层神经元的输出。多层神经元相互连接就构成了神经网络。

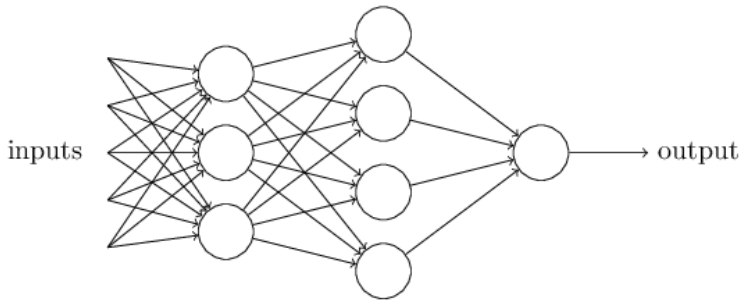

图 3-2 神经元层示意图

神经网络的主要工作层有 FC 层(full connected layers)，卷积层(convolution layers)和池化层(pooling layers)三种。

FC 层的每个神经元的输入都包含前一层的所有输出，因此 FC 层的数量越多需要的计算量越大，自适应参数的确定也会越困难。卷积层和池化层的每一个感受野(filter)都是 W×H，尺寸小于上一层 feature map 或者输入层尺寸。feature map 是指图像与一个卷积层进行卷积操作的结果，或者操作结果与卷积层作用的结果，在训练之后，这个结果中会包含图像的各种特征，便于进一步的处理。卷积层和池化层在上一层的 feature map 上移动，在每个位置计算得到一个值作为新的 feature map 对应层的一个点，而使用 K 个感受野组成的卷积层就可以得到 K 维的 feature map. 卷积层和池化层的不同点在于，卷积层的计算方式和图像处理中的滤波器一样，取的是离散卷积值；而池化层大小一般为 2×2 或 3×3，每次移动的跨度与边长一致，取出的则是作用范围内的最大值或者平均值。取出最大值的池化层叫做 max pooling 层(如图 3-3)，实际应用得更多。

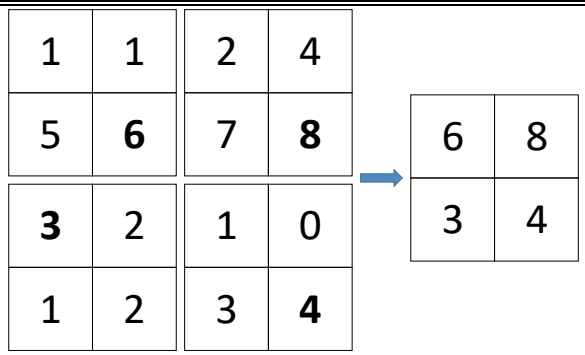

图 3-3 Max Pooling 层效果示意图

基于神经网络的定位使用的是 RPN 网络。这个网络的结构就是将多个卷积层和池化层依次排列，提取出图像的特征，最终送入 FC 层输出目标框位置参数(*x,y,w,h*)，分别对应框的中心点坐标和框的宽度、长度。这里框的大小和长宽比使用了预设的固定值，共有 9 种组合，每个位置上的一个预设框称为这个位置的一个 *anchor*。由于卷积层具有良好的位置不变性，这里使用了映射的方式，用第 5 个卷积层的输出中的一个点对应原图上的一个区域，则直接用这个点对应以这个区域为中心的 9 个预设框。第 5 层卷积层的输出为 256 维，下一个卷积层直接将输出变为 4×9=36 维，feature map 平面上每个点都对应 9 个区域的 4 个参数，之后连接一个全连接层并设置损失函数（*loss function*）进行打分，损失函数越大，代表网络输出和设定的真实输出差距越大。经过训练之后，网络可以给出每个框对应一个前景目标的概率，即实现对目标的定位。

3.3 目标检测算法

检测算法的种类很多，但基本的步骤都是一样的。首先进行图像分割和特征提取，之后投入分类器进行分类，最后对图像分割的结果进行重定位，得到一个比较精确的目标位置，完成检测任务。这些方法大致可以分为 3 类。

1.传统的目标检测算法

传统的目标检测算法使用传统数字图像处理方法，主要有背景建模和模板匹配两种方式。这种检测算法也是目前主流产品中使用的方法。

背景建模方式的基本原理是确定出画面中的背景，从中分离出前景中的目标。分为帧差模型、背景统计模型和 codebook 模型三种。其中后两种模型较为复杂，首先对图像中的像素点进行统计，然后根据经验设定一个阈值。当像素点满足某个阈值条件就被确定为背景。这两种模型实现了动态的背景建模，适用于提取变

化的背景和前景。帧差模型则最为直接，直接指定视频中一幅图像为背景，将当前帧和背景进行比较，从而提取出前景中的物体。显然这种方法不适用于变化的背景，但是在固定摄像头这种场景下是非常实用的。

图 3-4 背景建模背景（左）、检测图（中）、前景（右）

背景建模方法的缺点很明显。无论采取哪种方案，背景建模方法都是基于最简单的像素比较，对于光线的变化很敏感，容易出现误检的情况。并且这种方法只能简单的将前景提取出来，但无法对有遮挡的目标进行分割和识别

模板匹配方式是另一种基本的图像检测算法。它不是基于像素，而是特征。从标准图像中提取出一些特征向量作为模板，向量的提取和实际的场景有关。再对检测的视频采用同样的算法进行特征提取，将提取的特征与模板进行比较。即计算图像向量与模板的特征向量之间的距离，距离代表了与各个分类的相似程度，距离最小的一个分类即认为是待检测目标最可能的分类。图像的分割和定位采用了滑窗(sliding window)的方式，即从图像的左上角开始，取一个固定大小的矩形窗格作为感兴趣区域（ROI），并依次向后滑动，计算该区域提取出的向量与模板向量之间的距离。这里采用 Hausdorff 距离的定义方式。当计算出的距离小于一个规定的阈值时，就认为检测到了一个目标。

定义一个点 x 到一个点集 Y 的距离 d(x，Y)为该点到点集中每个点的距离中的最小值，即：

$$\mathrm{d}(\mathrm{x},\mathrm{Y}) = \min_{y\in Y}\|x-y\| \tag{3-5}$$

其中‖•‖为距离范数。定义与 Y 具有相同的元素数的点集 X，则 X 集合到 Y 集合的直接 Hausdorff 距离为 X 中每个元素到 Y 集合的最短距离之和：

$$\mathrm{d}(\mathrm{X},\mathrm{Y}) = \sum_{x\in X}\min_{y\in Y}\|x-y\| \tag{3-6}$$

这种方式比背景建模的方式得到的效果要好一些，但是由于需要根据实际人工选取特征向量，加大了设计时的工作量，准确度方面也很难达到要求。

2.基于深度学习的检测算法

传统的机器学习方法局限于在原始形态下对原始数据进行处理，几十年来机器学习一直依赖于严谨的建模和相当的专业知识[11]。Hinton 等人在 2006 年首次提出了深度学习的概念。深度学习使用模拟人脑结构的神经网络，对负载的输入数据进行高效处理，并可以自适应的学习和调整参数，从而有效解决多类的复杂智能问题[12]。针对深度置信网络(DBN)提出的非监督贪心逐层训练算法，为解决深层结构自适应发散的难题带来希望，随后提出深度神经网络结构。此外 Lecun 等人提出的卷积神经网络 LeNet 是第一个真正的多层结构学习算法，利用图像空间相对关系使网络稀疏化，降低了计算强度和发散的概率，提高了训练性能[13]。

深度学习目前在语音识别、计算机视觉、自然语言处理等方面得到了广泛的关注和应用。目前在世界范围引起广泛讨论和关注的 AlphaGo 也是一套基于深度学习引擎的博弈算法系统[14]。2012 年，Hinton 引入 CNN 解决 ImageNet 问题并取得了巨大成功，此后 ImageNet 等数据集便成为了深度学习理论发展和突破的引擎，在计算机视觉为代表的各个领域掀起了一股研究热潮。Google，Microsoft，Facebook，IBM，Twitter，yahoo 和 Adobe 等纷纷成立研究机构开始进行深度学习的实际项目研发，如：Microsoft 基于深度学习开发的视觉系统[15]。

深度学习目前在目标检测上得到了迅速的发展和应用。目标检测算法使得使用视频检测进行客流统计的准确性和可靠性得到了飞跃性的提升。深度学习检测使用海量样本对神经网络参数进行自适应训练，使网络具有目标分类的能力[16]。这种检测方式增加了方法的在模糊图像、形变和其他复杂条件下的鲁棒性(如图 3-5)，而目标特征的提取直接包含在了网络之中，不需要人为进行选取。但局限于当前主流计算机的性能，目前检测算法还不能做到实时性。

图 3-5 基于深度学习的运动目标检测

基于深度学习的检测方法的核心是一个主要作为特征提取器和分类器的深度网络。目前流行的深度网络结构一般为输入层+多个卷积层+2 个全连接层+一个 softmax 分类层，往往分类越多越复杂的情况下网络的层数越深。图 3-6 是一个最经典的 LeNet 的浅层网络的层次示意图。

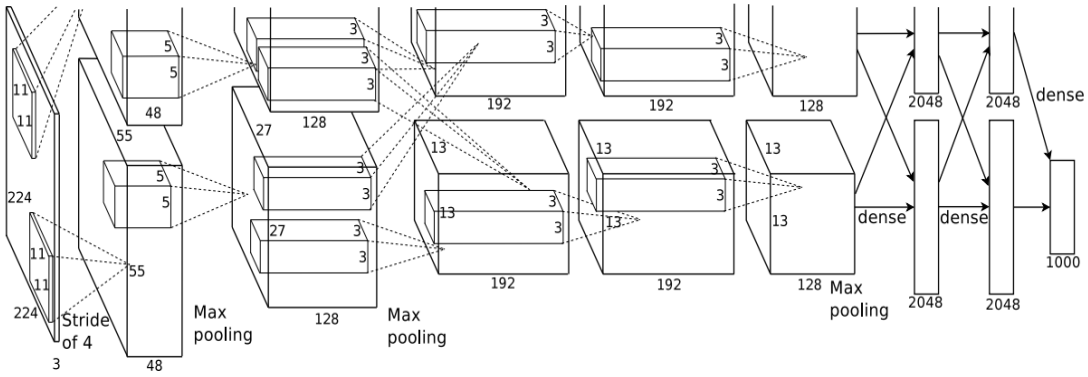

图 3-6 LeNet 结构图

最左侧是一个输入层，每个神经元都只有一个输入，整个层用来读入图片所有像素的三通道值。所有的值将被存入一个指定大小的向量以供后续的层进行处理。中间长方体代表的是 feature map，其内部的小长方体代表的是卷积层(convolutional layers)和池化层(pooling layers),这两种神经网络层是提取图像特征向量的主体。这里的卷积层和池化层可以用 FC 层来取代，但是相比之下卷积层和池化层具有减少计算量，减少训练发散，增强分类的位移不变性、不破坏图片空间结构等优点。现在已经出现了完全使用卷积层取代 FC 层的神经网络结构。卷积层右侧是两个全连接层(FC 层)，FC 层主要用来合并和压缩卷积层提取的特征向量，降低维度直到和所需要的分类数一致。网络最后的 FC 层不宜过多，否则由于参数过多很容易导致系统过拟合。

最终的 softmax 层有与分类数一致的神经元数，用来将分类的结果归一化，最终每个神经元输出的就是这个样本被分为对应类的概率。softmax 层的神经元激活值计算公式为：

$$a_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{3-7}$$

其中 z_j 代表第 j 个 softmax 神经元(对应第 j 类概率)的输入$\sum_k w_i x_i + b$。理想情况为正确分类输出 a=1，其他分类输出为 0.显然神经网络训练的最终目的就是使得输入图像对应的正确分类输出概率最高，其他分类输出概率远低于正确分类，具体的训练方法将在 5.2 节进行介绍。

3.4 基于深度学习的检测算法

每年都会诞生大量基于深度学习的检测算法，但其中的大部分算法都只适用于一些特定的场景，在使用之前需要对于算法的适用性进行检验。经过对资料的整理，最终筛选出了两种可能的检测算法，SSD 和 Faster R-CNN 进行比较。

3.4.1.Faster R-CNN 算法

Faster R-CNN 算法是 R-CNN 系列算法效果最新和效果最好的。R-CNN 算法抛弃了传统的滑窗式定位方法，首先引入了选择性搜索提取出 2000 个 bounding box 作为候选区域，如 3.2.1 小节所述，但定位和分类的速度和准确度还是较低。在 Fast R-CNN 中，为了使分类器可以做到比较好的效果，放弃了 SVM 分类，而是将选择性搜索的结果和图像同时作为输入传递给分类，直接使用网络的后端进行分类，大大加快了速度。Faster R-CNN 算法则使用了 RPN 网络，直接使用网络进行区域分割和前景的定位，提取候选窗口。

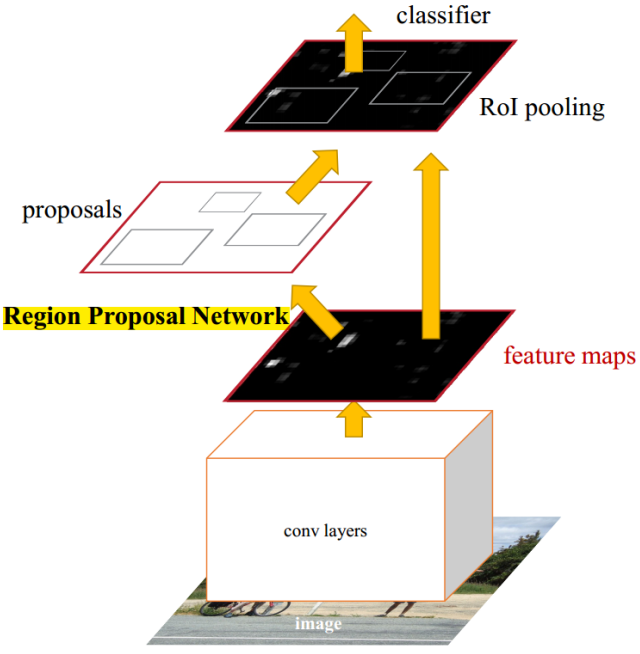

图 3-7 Faster R-CNN 结构示意图

和 Fast R-CNN 的策略相同，这些候选窗口和深度学习网络提取的特征一起作为输入进入最后的 FC 层进行分类，但 Faster R-CNN 将 RPN 和分类网络的前 5 个卷积层作为公共层，节省了 RPN 网络的主要计算量，显著压缩了计算时间

和显存占用。

为了使网络能够达到最优参数配置，这里需要引入一个损失函数的概念。定义 Faster R-CNN 的损失函数为：

$$L(\{p_i\}\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (3\text{-}8)$$

这里 N_{cls} 和 N_{reg} 分别代表网络的分类数和 feature map 的 *anchor* 数，i 是 feature map 一个像素上的 *anchor* 编号，L_{cls} 和 L_{reg} 分贝是分类和定位的损失函数，pi 和 pi^* 分别代表网络得到的分类和真实分类，t_i 和 t_i^* 分别是衡量 4 个位置参数与 anchor 之间偏差的量，即算法通过 t 对检测对象的定位进行调整。t 的四个分量分别为：

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a$$
$$\mathrm{t_w} = \log(\mathrm{w}/\mathrm{w_a}), \quad \mathrm{t_h} = \log(\mathrm{h}/\mathrm{h_a}) \quad (3\text{-}9)$$

3.4.2.SSD 算法

SSD 算法则是一种一次循环算法，将感兴趣区域的划分和调整放入神经网络进行处理，去掉了第一次搜索的过程。SSD 方法的核心就是预测物体位置，以及其归属类别的打分；同时，在 feature map 上使用小的卷积核，去预测边框的偏移来修正边框的位置和大小。

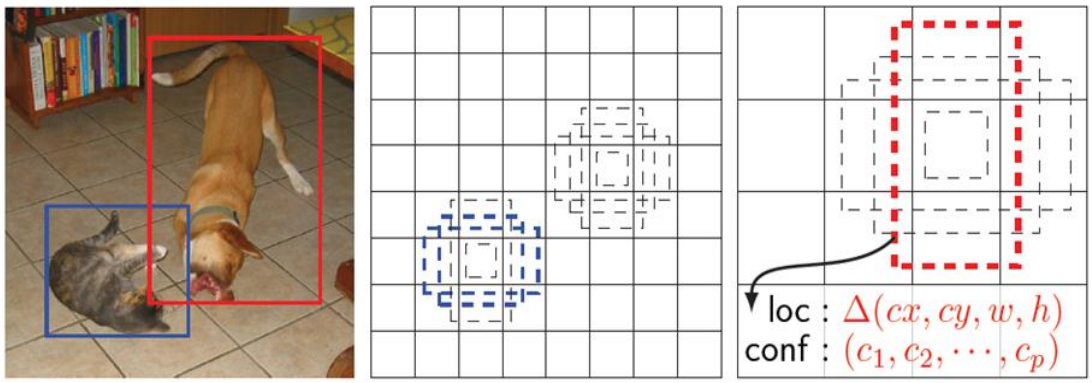

图 3-8 不同尺寸的匹配情况

SSD 的最大特点是将 5 层后每个尺度的 feature map 都提取预测框并进行分类，如图 3-9 所示。根据 feature map 的定义，越靠后的 feature map 每个像素点对应的原图区域就越大。而 SSD 为了加快目标定位的速度，放弃了 selective search 的方式，转而对不同层次的 feature map 画相同形状大小的框代表位置，如图 3-

8，这样用不同大小的 feature map 可以避免将相同图像放缩到不同尺寸后重复放入网络进行分类，大大提高了检测的速度。新增加的每一个卷积层的特征图都会经过一些小的卷积核操作，从而得到相应的默认窗口对应于不同物体类别的偏移（shape offsets）和置信度。在大小为$m \times n$，有 p 个通道数的特征图上，使用大小为$3 \times 3 \times p$的卷积核，如果特征图上的每个点对应 k 个默认窗口的物体共有 c 个分类，那么就需要使用$(c+4)k$个这样的卷积核，最后有不多于$(c+4)kmn$个输出。其中的 4 是记录 offset 的四个参数，c 包含了背景(负样本)。

在训练阶段，需要给定输入图像和每个物体的 ground truth(标出正确位置的矩形框)。每个默认窗口与任意一个 ground truth 的 IOU(交并比，IOU=(A∩B)/(A∪B))大于 0.5，就表示两者的匹配很好，应该作为正样本，小于 0.5 的就作为负样本。显然，一个 ground truth 可能会有多个默认窗口对应。所以在预测阶段，直接预测每个默认窗口的偏移以及窗口对于各个分类对应的得分，最后使用非极大值抑制的方式来得到最后的检测结果。图 3-8 的例子是：给定输入图像(左)及 ground truth(左图的矩形框)，分别在两种尺度（feature map 的大小分别为8×8，4×4）下的匹配情况。与猫匹配的默认窗口（8×8）有两个，与狗匹配的默认窗口（4×4）有一个。

SSD 的网络框架也是基于传统的用于分类的基础网络结构，如 AlexNet、VGG 等，使用通用的结构(如前 5 个卷积层等)作为基础网络，然后在这个基础上增加其他的层。为了在保证准确性的基础上提高速度，这里使用了改进的 VGG-16 网络来作为 SSD 的基网络，将所有的卷积层替换为了 2×2 和 3×3 的小卷积层，去掉了所有的 dropout 层和最后一层 fc 层。

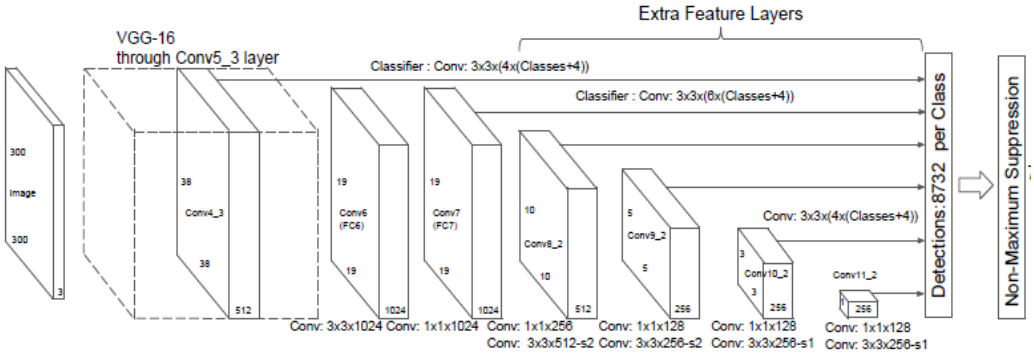

图 3-9 SSD 结构示意图

训练的目标函数和 Fast R-CNN 类似，分为两部分：计算每个默认窗口的位置偏差与目标分类的置信度，以及相应的回归结果（位置回归）。置信度计算是

采用 Softmax Loss，位置回归则是采用 Smooth L1 loss。

$$L(x,c,l,g)=\frac{1}{N}(L_{conf}(x,c)+\alpha L_{loc}(x,l,g)) \quad (3\text{-}10)$$

提出SSD的论文中利用不同层的特征图来模仿学习不同尺度下物体的检测，并定义了推荐的特征层尺度选取公式。假定要使用 m 层的特征图来做分类和定位，最底层的特征图的尺度值为$s_{min}=0.2$，最高层的为$s_{max}=0.95$，其他层尺度通过下式计算得到：

$$s_k=s_{min}+\frac{s_{max}-s_{min}}{m-1}(k-1),k\in[1,m] \quad (3\text{-}11)$$

对于预测的矩形框的形状大小选择，算法使用了不同的比率值 $a_r\in\{1，2，\frac{1}{2}，3，\frac{1}{3}\}$来确定宽度和高度：${w_k}^a=s_k\sqrt{a_r}$，${h_k}^a=s_k/\sqrt{a_r}$。另外对于比率=1 的情况额外再指定一个尺度为$s_k'=\sqrt{s_k s_{k+1}}$，就是每个 feature map 总共有 6 种不同尺度的默认窗口。每个默认窗口的中心位置要设置成$(\frac{i+0.5}{f_k},\frac{j+0.5}{f_k})$，其中$i,j$分别表示在大小为$f_k\times f_k$的特征图上点的位置。

显然图像中的负样本数量远多于正样本，训练过程中采用了“Hard Negative Mining”的策略来减少负样本的比例，减轻训练过程中的发散情况，提高算法的实用性。具体方案为根据默认窗口的置信度排序，只选取置信度较高的窗口作为负样本，最后将正负样本的比率控制在 1:3 左右。

3.5 本章小结

本章主要介绍了检测算法的基本原理，检测算法始终包括目标的定位、分类和边框回归三个步骤。本章首先给出了常用的定位方法，包括滑窗法，选择性搜索和基于神经网络的定位方法。之后介绍了传统检测算法和基于深度学习的检测算法。传统检测算法包括背景建模和模板匹配，基本思想都是使用一个模板作为背景，区别于背景的其他目标则定义为前景。基于深度学习的检测算法则使用深度网络作为特征提取和分类的主体，由于参数的自适应性，经过训练后，深度网络的分类准确性远高于其他方法。本章重点介绍了基于深度学习的算法中的 SSD 和 faster R-CNN 算法，并给出了两种算法的损失函数。

第 4 章 多区域客流统计系统设计

4.1 引言

检测和跟踪算法是本系统的核心算法，但功能的实现主要是进行算法的整合重构。为了在实时性前提下实现更好的统计效果，系统需要使用快速性好、准确率高的跟踪算法。但由于单目相机中缺少深度坐标，造成跟踪算法对遮挡的处理能力有限，需要对目标进行重检测防止丢失。系统使用了先进的目标跟踪和目标检测方法，并对方法进行整合重构。因此检测和跟踪只能作为整个系统的两个库进行使用，而主体程序的设计则是本项目的重中之重。机器视觉系统的开发需要相应的软硬件支持，项目需要对使用的编程环境和硬件进行筛选，以获得更好的效果。摄像头在接入之后通过 IP 和端口登陆之后，即可在浏览器中显示当前的实时画面。但是用实时视频流做二次开发的时候需要自行解码成为连续的图片帧后才能作为程序输入使用。本章介绍了系统的总体设计、开发和工作环境、基本模块的实现以及进一步的改进升级。

4.2 编程环境和硬件系统的选择

Windows 环境下的界面操作更为友好，针对用户的服务更加完善，但相对来说功能更加冗杂；而 Linux 系统下的系统操作更加灵活，且系统的稳定性更好，开源代码的获取也更加方便。从技术成熟度和系统对于长期运行稳定性的要求两个方面考虑，本项目选择了 Linux 系统。根据目前收集的用户体验反馈，本项目选择了当前在本领域比较稳定和成熟的 Ubuntu 14.04 系统，并安装了官方推荐使用的 CUDA 7.5 版本 GPU 计算平台。由于本项目不涉及对算法本身的大规模优化，项目选择了具有成熟模板的 caffe 深度学习框架来训练检测算法。

C++编程方面，由于本系统与操作系统间不存在过多的系统命令交换，本项目选择了先在 Code::Blocks 的 IDE 下编程实现各个库，最后合并移植为终端程序的策略，充分利用 IDE 编程自动联想和强大的调试功能，同时也避免了 IDE 编程对于新的编译文件的查找链接能力较弱的问题。程序使用的最重要的库是 OpenCV, 包含了大部分目前流行的视觉算法和操作，同时支持在 Windows 和 Linux 环境下的编译和使用。

硬件方面，由于需要对图像中的目标进行检测，需要采用方便处理的数字信号传输图像。本项目使用了海康威视的 DS-2CD2355F 摄像头，500 万像素实时图像，具有夜间红外成像功能，可以 24 小时实时统计，捕捉图像为数字格式，使用网线传输，具有优秀的高清图像压缩传输功能，满足了当前系统的需要，而价格也是满足条件的硬件中最便宜的。相机具体参数如表 4-1。

表 4-1 摄像头参数列表

型号		DS-2CD2355F(D)-I(S)
摄像机	快门	1/3 秒至 1/100,000 秒
	镜头	4mm,水平视场角:90° (2.8mm,6mm,8mm,12mm 可选)
	调整角度	水平:0°~360°；垂直:0°~75°； 旋转:0°~360°
	日夜转换模式	ICR 红外滤片式
压缩标准	视频压缩标准	H.265/H.264/MJPEG
	编码类型	H.265 Main Profile
	压缩输出码率	32 Kbps~8Mbps
图像	最大图像尺寸	2560×1920
	帧率	50Hz:2560×1920@12.5fps, 25fps(2048 × 1536, 1920 × 1080, 1280 × 720)
接口	通讯接口	1 个 RJ45 10M/100M 自适应以太网
	电源供应	DC12V±25% / PoE(802.3af)

摄像头的安装共有两种方案，分别为有倾角的斜向安装和垂直向下的安装方式，在安装中主要要考虑监视范围、准确性和对实时性的影响等因素。目前一般的监视摄像头主要采用斜向安装，以便获得更广的视场和更多的有用数据，但由于画面中存在较大的空间变化，前景目标会出现较大的尺度变化。

由 2,3 章内容可知，本文使用的核心算法对于过小目标识别能力有限，且引入尺度变化的情况下会造成算法的复杂度增加。同时，距离跨度大也会造成目标之间以及目标和背景之间很容易出现遮挡、阴影等情况，给跟踪算法带来极大的挑战。在使用斜向安装的摄像头实际测试中，远处的目标基本没有被检测和跟踪，可以认为是无效数据。斜向检测跟踪的实际效果如图 4-1：

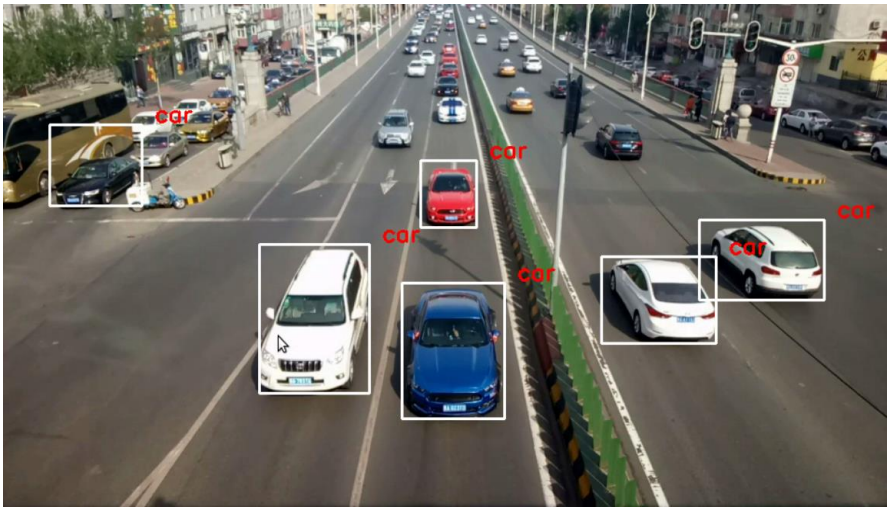

图 4-1 斜向检测效果

采用摄像头垂直监视会造成监视范围变小，但画面中目标不会出现较大的尺度变化，对于简化算法，增强系统实时性具有很大的作用，理论上甚至可以做到简化检测网络。另外，在垂直拍摄的情况下选择行人的头肩作为检测和跟踪的目标，基本可以消除目标之间的遮挡情况，提升系统的准确性。综合上述分析，本系统选择牺牲视野范围，采用垂直向下的拍摄角度。

由于本项目中摄像头安装高度不超过 5m，这个型号的摄像头分辨率已经可以满足条件。摄像头在多个室内地点完成安装，信号通过路由器接入局域网，可以使用浏览器通过 IP 地址进行实时访问(如图 4-2)。

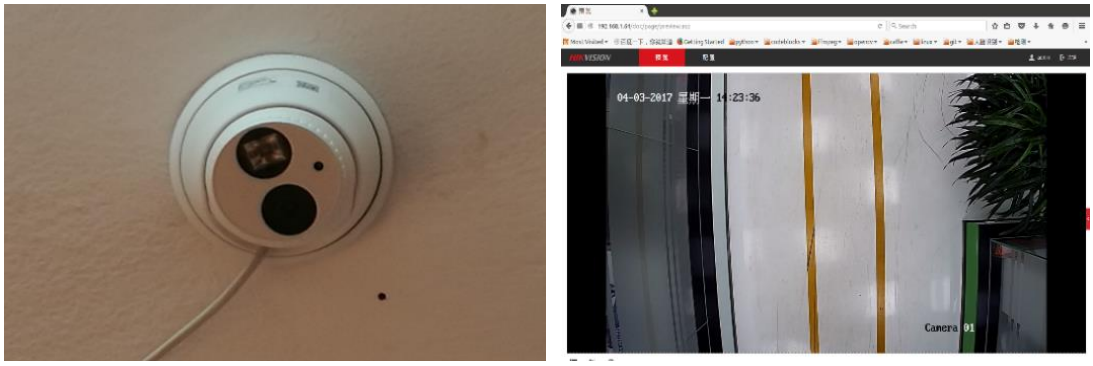

图 4-2 安装的摄像头(左)和浏览器实时图像(右)

4.3 整体流程

整体系统框架设计是决定本项目能否成功的关键一步。程序的基本功能实现流程如图 4-3 所示。首先对程序中检测线、通过检测线的正方向和摄像头进行初

始化，之后开始从摄像头获取流，并解码。解码后的图片可以使用 OpenCV 相关库进行显示。

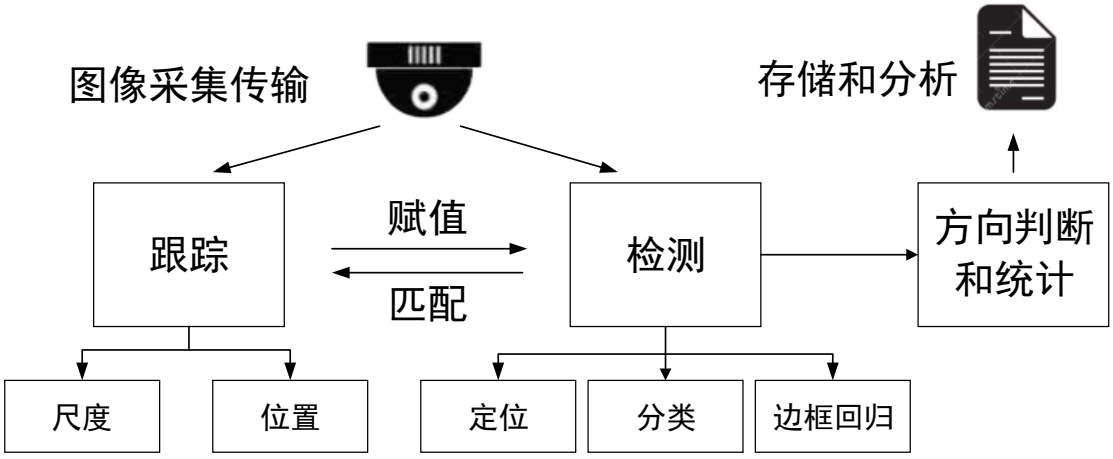

图 4-3 系统整体结构

初始化跟踪器后，每次使用检测器检测一帧，将检测到的所有目标坐标传送给跟踪器，然后更新通过的人数信息。在之后的若干帧中只使用跟踪器更新目标位置。由于检测器检测到的目标顺序和跟踪器跟踪目标顺序之间没有对应关系，需要对两者进行匹配，判断检测器检测到的哪些目标是原来跟踪器跟踪的，哪些目标是新进入画面的，以及跟踪器跟踪的哪些目标已经离开画面，之后才能重置跟踪器，继续进行跟踪。

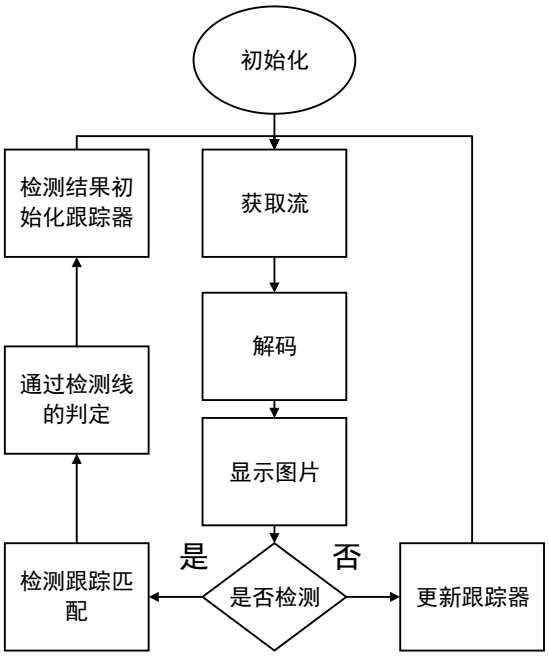

图 4-4 核心程序框架图

对应于上述的流程，系统设计基本功能实现的 UML 图如图 4-5：

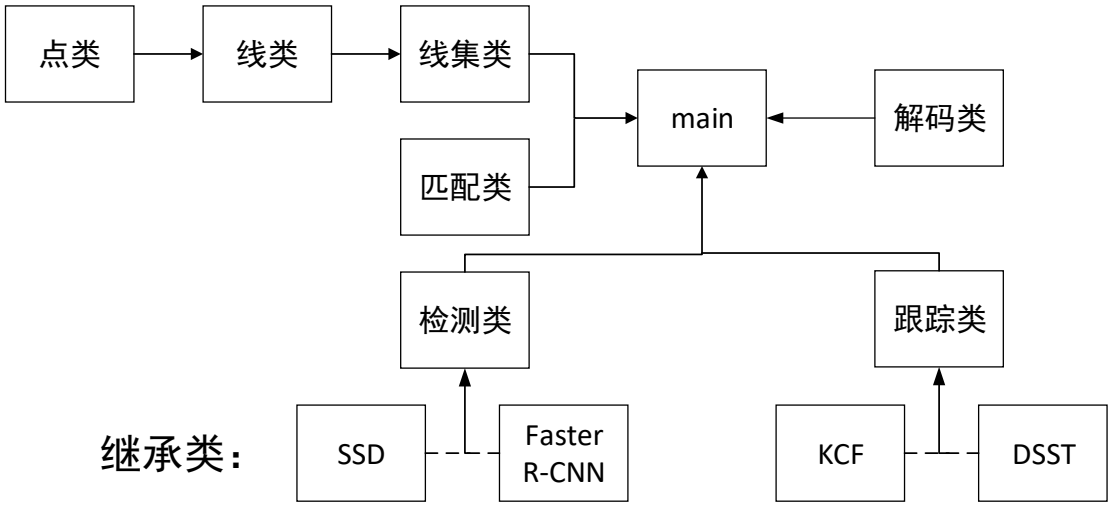

图 4-5 系统类结构图

系统的核心算法，即跟踪和检测算法，由继承关系写成，首先分别设置检测类和跟踪类，设置跟踪和检测分别必须的函数，之后将可能用到的算法分别封装为对应的继承类，在调用时根据输入的方法名称即可调用对应的类，方便在测试时更换方法使用。这样做的另一个好处是在未来出现更合适的算法之后可以方便程序的更新，减少程序的耦合性。由于系统使用的跟踪算法是单目标的跟踪算法，且目前世界上出现的效果较好的跟踪算法也都是单目标跟踪，造成实际系统在执行任务过程中消耗时间随跟踪目标数成线性增长，这也是系统在拥挤人群画面中实时性降低的主要原因。未来出现有效解决这个问题的跟踪算法后，就可以通过继承直接调用而无需对系统本身进行修改。

系统的轨迹记录由三个递进包含的类构成。首先是点类用来记录每次需要记入轨迹中的点。线类中包含一个点类构成的链表成员变量，用来记录每个通过的行人的轨迹。线集类中按照编号顺序包含了所有的轨迹。匹配类中包含一个函数，用来把前一帧的跟踪目标框和当前帧的检测框做匹配，并放入线集类对象对应的轨迹线中，具体的原理将在下一小节详述。匹配类另一个函数用来将轨迹线画在图内，最终在主程序中显示。

为了保证内存不会积累和数据不会溢出，系统中所有随时间积累的变量全部采用 long 型或者 double 数据型，同时循环进行定量的清零操作。对于储存所有轨迹的线集类成员对象，系统提供了接口用来在每天定时清零，清零的时间可调。

4.4 系统基本模块实现

4.4.1 解码和实时图像获取

为了保证系统的稳定性和方便对图像进行操作，本项目使用的摄像头采用 H.265 压缩标准，能实现高清图片的高速实时传递。H.265 标准采用了先进的技术来解决码流处理时间和码流质量、时延和算法的复杂度之间的矛盾，最大程度实现系统的最优化。这个标准具体的研究内容涵盖了压缩率的进一步提高，以及鲁棒性和容错性的进一步增强。该标准从信道获取时间和随机接入时延入手，压缩系统时延，同时降低算法的复杂度。在 H.264 标准下进行优化后，可以用小于 1Mbps 的速度发送标清数字图像；H.265 标准则可以实现利用低于 2Mbps 的速度，传送等效于 720P 的高清音视频，对于本系统的任务已经可以满足要求。收到信号后，本系统设计了解码库进行解码，解码包含两层回调函数的使用。在输入相机 IP 地址、端口以及用户名和密码实现注册和登陆之后，第一层调用为从摄像头的地址获得视频数据流，第二层为从数据流解码出 YU12 格式的图像。具体流程如图 4-6：

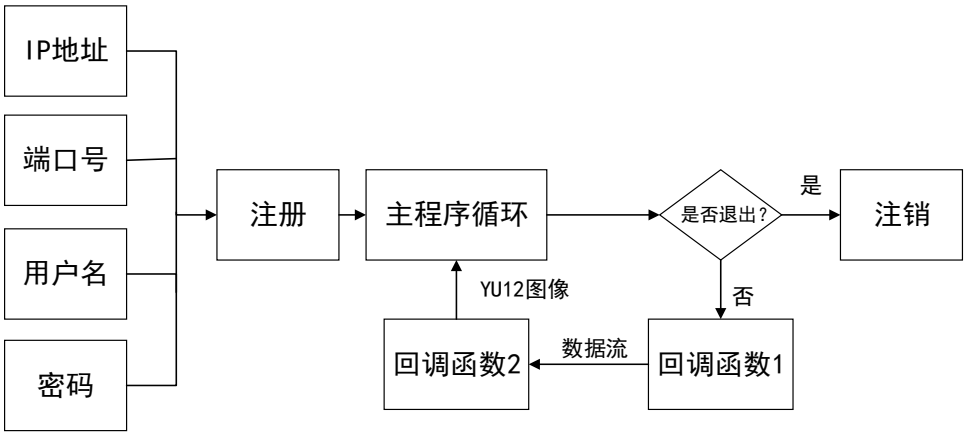

图 4-6 实时解码程序流程

在得到的 YU12 格式图片的基础上，需要继续转换为 YUV 格式，然后转换为 RGB 格式图片，并按 BGR 通道顺序存储在 Mat 数据型变量中。YUV 格式是电视信号和摄像头信号图像的一种存储方式。其中，Y 表示亮度，即灰度值；U 和 V 表示色度，描述图片的色彩和饱和度。YV12 格式图片是一种进一步压缩的格式。将 Y、U、V 分量分别打包，依次存储。其每一个像素点的 YUV 数据都是 4 个 Y 分量共用一组 UV，如图 4-7 所示，因而 V 和 U 两个通道的矩阵高度

已经压缩为原来的 1/4，故相对于三通道矩阵大小一致的 RGB 图像来说，在传输中可以大幅压缩数据量。之后需要继续转换为 OpenCV 库方便处理的图像，即 RGB 三通道图片，并且转换为 Mat 矩阵类型进行存储，以便下一步处理。

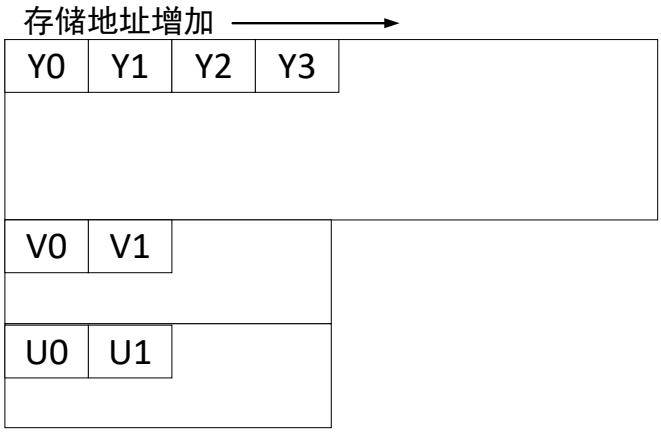

图 4-7 YV12 格式图片存储示意图

为了方便使用 OpenCV 库做进一步的处理，同时也为了适应检测和跟踪算法的输入格式，需要从 YUV 格式转换到 RGB 格式。转换使用的一个常用的经验公式如下：

$$\begin{aligned} R &= Y + 1.14V \\ G &= Y - 0.39U - 0.58V \\ B &= Y + 2.03U \end{aligned} \qquad (4\text{-}1)$$

实际使用中可以直接使用 OpenCV 库中的函数，最终解码得到实时的监控图像，如图 4-8：

图 4-8 视频解码得到的图片

4.4.2 匹配算法

本项目使用的匹配算法原理如下：首先定义跟踪框和检测框的位置为每个框的中心，然后将跟踪器对应的 K 个跟踪框位置和检测对应的 L 个检测框位置分别放在链表中，输入到匹配类。定义第 i 个跟踪框位置和第 j 个检测框之间的关联距离为：

$$D_{ij} = \sqrt{(x_{di} - x_{tj})^2 + (y_{di} - y_{tj})^2} \tag{4-2}$$

在程序实现中，根号和平方运算会消耗大量的计算时间，而在图像运算中，由于计算的元素都是量子化的，根据情况，可以对定义进行一定调整。为了减少计算量，（4-2）式可以简化为：

$$D_{ij} = |x_{di} - x_{tj}| + |y_{di} - y_{tj}| \tag{4-3}$$

所有检测框分别和所有跟踪框之间计算关联距离，则所有的关联距离构成 K×L 的关联距离矩阵 A：

$$A = \begin{bmatrix} D_{11} & D_{12} & \cdots & D_{1(L-1)} & D_{1L} \\ D_{21} & D_{22} & \cdots & D_{2(L-1)} & D_{2L} \\ \vdots & \vdots & D_{kl} & \vdots & \vdots \\ D_{(K-1)1} & D_{(K-1)2} & \cdots & D_{(K-1)(L-1)} & D_{(K-1)L} \\ D_{K1} & D_{K2} & \cdots & D_{K(L-1)} & D_{KL} \end{bmatrix} \tag{4-4}$$

之后的操作对 A 进行。首先定义一个阈值 threshold，关联距离大于这个阈值就认为两个点之间没有关联。从矩阵中取出认为有效的关联距离，最多可以从矩阵中取出 min{K，L}个彼此不共行且不共列的值，对应矩阵短边的长度。取出的距离对应的两个框我们认为即对应同一个目标。出现没有检测框按对应的跟踪框则认为被跟踪的目标已经离开画面，而出现没有对应跟踪框的检测框则认为是新进入画面的目标。最终将匹配的检测点放入原来的轨迹，未匹配的检测点作为新轨迹的起点，未匹配的跟踪轨迹认为已经结束，和检测线进行是否通过的判断。最终的输出为更新后的轨迹。

取出关联距离的方案有两种：

1.每次都取出矩阵中小于阈值且最小的一个距离，同时把这个距离同行同列的距离全部排除，直到取出了 min{K，L}个距离或者没有满足要求的值为止。

2.使用简化的方法，总体思路一样，但每次取值仅限制在同行或者同列中，

以此来简化运算。

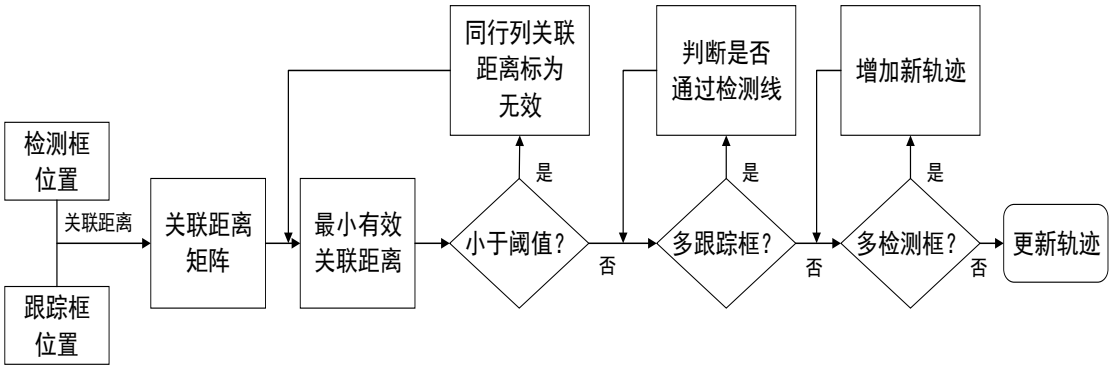

图 4-9 匹配算法实现流程图

很容易看出，第二种算法的精确度比第一种要差，具体需要在程序中实际验证第二种算法的准确度。经过在视频中的验证，第二种方案不仅可能出现丢落目标的情况，甚至可能由于优先取局部最优解造成全局性的错误。因此这里的计算采用第一种方案。

4.4.3 通过人数的计量

最后，为实现人数的统计功能，需要设计一个方法来检测行人是否过检测线，从哪个方向通过检测线。目前大多数产品的方案是采用轨迹经过检测线两侧区域的任意两个点即判定通过检测线，但这种方案在每帧都需要对点的位置进行判定，并且在一些行人折返和做出复杂轨迹的情况下会发生重复计数的情况，如图 4-10 左，由于行人两次通过检测线，此时行人会被计数两次，但实际上行人等同于并未通过检测线。为了保证实时性要求，有必要设计一种新的判定方案，减少程序耗时，增加对复杂情况的判定准确性。

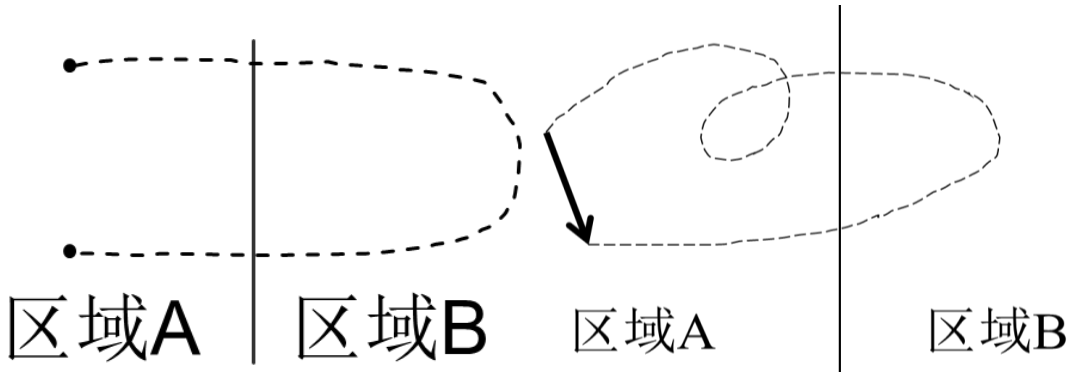

图 4-10 是否通过检测线的判断方法示意图，左侧为传统方案，右侧为本项目方案

一种改良方案是每条检测线都在最后计算通过检测线的次数的奇偶，奇数为通过，偶数为不通过，但仍然不能满足时间优化的要求，反而增加了计算量。本项目设计的解决方案为：不从区域入手，而是直接分析每条轨迹的首尾两点连成的固定向量，即行人在画面中的位移，这样计算也保证了只在行人离开画面之后对轨迹进行一次判断，大大节省了时间。若向量横跨检测线两端，则进入人数或离开人数计数加一，其他情况计数不变。显然，这种方式解决了前面提到的两个问题。

判断行人是否通过检测线实际上是对两条线段是否相交进行判断，即判断两条线段的端点是否分别在另一条线段的两侧。由于行人在画面中是有面积的，而检测框和跟踪框框选的位置可能会出现偏移，这里需要对线段相交情况留出余量，即若线段相交但其中一条的一端过于接近另一条线段，就认为两者并不相交。故首先对两条线段是否相交做出判断，这里使用的方法是叉积法，两条线段的叉积正负可以反映两条线段在旋转方向上的顺序。如图 4-11，以要进行判断的线段 AB 和 CD 为对角线做出四边形 ACBD。

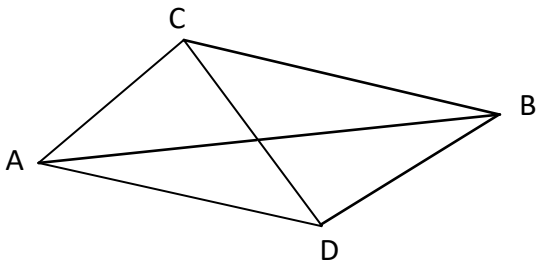

图 4-11 线段 AB 与 CD 形成的四边形

判断 CD 在 AB 两端的条件为：

$$(AB \times AC) \cdot (AB \times AD) \leq 0 \quad (4\text{-}5$$

即若 AB 旋转到 AC 和 AD 的方向，分别需要向不同的方向旋转。同理，判断 AB 在 CD 两端的条件为：

$$(CD \times CA) \cdot (CD \times CB) \leq 0 \quad (4\text{-}6$$

在得出两条线段相交的结论之后，需要对相交是否有足够的余量进行判断。首先定出一个阈值 T 表示能接受的端点到检测线的最小距离，之后分别算出轨迹向量上到两个端点 A 和 B 距离为 T 的点 A’和 B’，若 AA’和 BB’也分别和检测线相交，则认为轨迹通过了检测线且留出了足够的余量。

在确定行人的位移穿过了检测线之后，判断进出方向时使用向量比较的方法。

给出一个正方向向量，用图 4-10 右侧方案得到的实际位移向量和它点乘，当结果为正说明人沿着正方向离开画面，反之则沿着反方向离开。检测线和正方向矢量的标注如图 4-12，注意在真正工作时只有检测线在图中画出。

图 4-12 检测线和正方向矢量

为实现本系统的基本功能，还需要能实现在一个画面内对多条检测线进行客流统计。由于不需要对行人轨迹进行更多操作，且在一副画面中能设计的检测线数量是很有限的，实际在增加有限条检测线后增加的内存占用也几乎可以忽略，而增加的判断次数造成的计算复杂度增加也可以忽略。因此本系统采用固定长度的向量储存检测线和正方向矢量的参数。如果对应一条检测线的参数全部为 0，就认为这条检测线没有使用。

4.4.4 多线程设计

在系统，每个正在独立运行的程序都是一个进程，进程之间一般是独立的，不相互交换数据。每个进程至少包含一个线程，线程可以理解为代码运行的上下文，每个线程都可以实现程序中的部分功能，线程之间可以相互独立，可以比较容易的交换数据，但是所有线程都依赖于进程存在，进程结束所有线程都会关闭。通常由操作系统负责多个线程的调度和执行。多线程（英语：multi-threading），是指从软件或硬件层面实现一个进程中同时运行多个线程的技术，具有多线程并行能力的计算机在硬件的支持下可以在同一时间运行多个线程，即同时执行一个程序中的多个功能，从而提升进程整体的处理速度和效果。使用多个处理器核心的 CPU 在线程间有较多数据交换的情况下，由于需要跨核调用数据，速度会比

预想要慢。但一个进程对应线程数受到 CPU 核心数限制，若线程过多，则运行不会报错，但实际运行速度达不到真正的多线程效果，实际因为会出现多个线程在队列中依次处理的情况，运行速度会比单线程慢。

在单线程情况下，整个程序分为解码和图像操作两大部分。由于解码的函数调用了两层回调函数，而回调函数中不应出现占用时间很长的函数操作，否则会出现程序报错。这种情况决定了不能把对图像的操作放在解码图像的回调函数中。因此需要对回调函数和主程序加上类似线程锁的结构，在解码一帧之后才能进行图像操作，在图像输出之后才能继续解码下一帧，这个操作通过设置两个用于判断的全局变量来实现，如图 4-13：

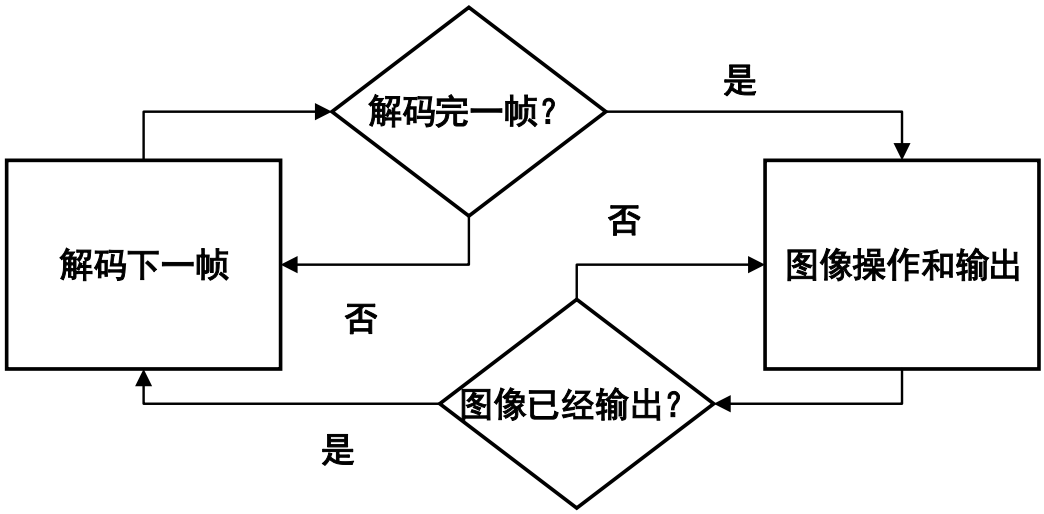

图 4-13 单线程流程示意图

在处理视频的时候，由于无论间隔多长时间，读入的都是视频中连续的下一帧，这种设计是能够满足需求的。但根据本项目的在线处理实时性要求，实际需要引入多线程来处理程序。原因在于，由于单线程程序中必定存在“锁”的结构，实际程序在执行的时候的延迟会直接影响系统的性能。证明如下：

假设程序使用帧率为 20 的实时视频，则对每帧的处理时间最多为：

$$T_{max} = \frac{1}{20} = 0.05s \tag{4-7}$$

由于 OpenCV 对于图像矩阵的处理使用了多线处理的方法，根据实际，忽略每帧显示所用的时间，而匹配矩阵的运算相对于图像百万像素级的运算同样可以忽略，因此只需要考虑解码和图像的跟踪检测时间，。每帧解码用时约为 0.01s 级别，跟踪每个目标用的时间最多为 0.0015s，检测用的时间约为 0.3~0.4s 级别。

人步行的速度约为 1.2m/s，设画面中有 5 个目标，摄像头挂在 3m 高度，拍摄范围直径约为 3.5m，最终显示画面的宽度为 400 像素。可以看出跟踪帧此时并不会对系统实时性造成很大影响。实时显示的两帧中，一个目标移动的像素数为：

$$s = \frac{1.2 * 0.05}{3.5} * 400 \cong 7pix \tag{4-8}$$

而检测帧时间内跳过了数帧，则画面中人卡顿后会跳到画面的另一处，移动的距离约为：

$$s = \frac{1.2 * 0.35}{3.5} * 400 \cong 50pix \tag{4-9}$$

在画面中已经对应相当长的距离，对匹配算法会造成很严重的干扰，也会对显示的流畅性造成严重影响。同样的，跟踪目标数过多也会造成这种效果。因此当画面中人数较多时，视频显示会出现严重的跳帧，匹配准确性严重降低。

本项目使用多线程后的程序结构如图 4-14 所示。主线程用来处理图像解码和图像的显示，线程 2 用来对图像做检测和跟踪以及匹配等处理。两个线程之间不使用线程锁进行锁定，而是引入两个全局变量的 queue 结构来传递图像。主线程将解码的图像存入 queue1,线程 2 在每次处理完一帧之后自动从 queue1 读入；而线程 2 将画好轨迹的图像存入 queue2，主线程读入后进行显示。其中 queue1 的作用显然是为了保证最终输出的图像不会出现跳帧的情况，而 queue2 的作用则是作为缓冲，提高主线程显示的流畅性。同时，queue2 保证了如果需要将输出图像进行传输等后续操作，可以将这些操作放进负担较轻的主线程中，而不占用线程 2 处理跟踪和检测的时间。

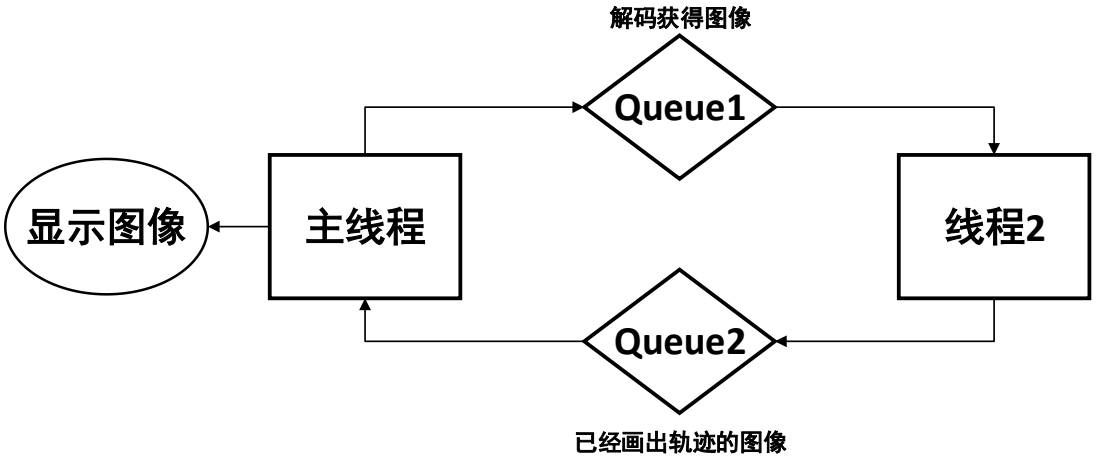

图 4-14 多线程程序结构示意图

为了保证线程安全和不影响内存，规定在 queue 大小小于 3 时不能进行

dequeue 操作，queue 大小大于 500 时自动清零。为了保证多线程的运行效率，将每个线程各自固定在一块 CPU 上，一定程度上节省了每次调用时分配 CPU 的时间，但由于数据在不同的 CPU 之间进行传递，会造成各个函数执行时间增加。在第 4 章提到的较低训练配置下，本项目使用的双线程一般会对应函数执行时间为原来的 2 倍左右。

由上面的参数可知，跟踪和解码即使在时间倍增的情况下依然能超过实时性要求，因此 queue1 在线程 2 处理检测帧的时候会积累图像，而在处理跟踪帧的时候大小会减少，实现了动态平衡。接着使用上面证明时候的数据，这次所有函数的执行时间都乘以 2，具体计算如下：

每帧检测对应时间内可以读入的图像数为：

$$N_d = \frac{0.6}{0.05} = 12 \tag{4-10}$$

但每帧实时对应时间内可以处理的跟踪帧数为：

$$N_t = \frac{0.05}{0.02 + 0.003} \cong 2 \tag{4-11}$$

即只要在每帧检测后面跟上 13 帧或更多跟踪，就可以实现动态平衡地对视频进行在线处理。13 帧对应的位移像素为：

$$7 * 13 = 51pix \tag{4-12}$$

如果检测线设在图像的中心附近区域，则可保证在人进入画面后能在线的两侧分别被检测到，从而保证对人数的正常计数，否则需要提高检测的频率，视频的显示连贯性会降低。当然，在摄像头位置更高的情况下，需要的检测频率更少，此时也可以继续降低视频帧率来节约计算资源。

4.5 系统的改进升级

截止上一节的工作已经实现了本系统的基本功能。本节将从系统的稳定性和功能拓展角度出发进行改进和升级。

4.5.1 可视化设计

为了真正形成可以推向市场的产品，在实现基本功能之后，需要对产品进行可视化和可操作化的设计，即显示和操作都在 GUI 中完成，从而达到面向用户，增强用户体验的要求。

1.界面设计

本系统的界面设计使用了 QT 框架。QT 是一个跨平台的 C++图形用户界面设计框架，可以免费使用，拥有自身的 IDE，同时框架的模块化程度很高，复用性好，使用安全，方便开发。同时 QT 有专门的类来支持 XML，简化了大量数据存储和调用的过程。

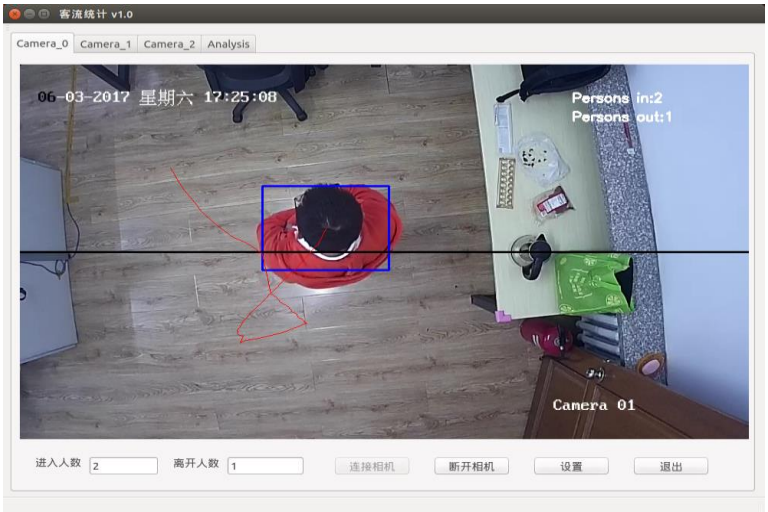

图 4-15 主机端界面

本系统的主机端的界面如图 4-15。主界面主要用来显示实时画面和统计数字，进行与主程序连接、结束程序。界面共可以接收三路相机信号，使用左上的三个按钮来选择显示的相机。

Signal/slot 机制是 QT 框架的主要特点。它取代了传统的 callback 函数，使得各个元件和窗口之间的协同工作变得简单安全。在两个元件内分别设置一个 signal 和 slot 函数，然后用 connect 函数将发出 signal 的函数和接收的 slot 函数绑定，则 slot 函数只会在收到这个 signal 的时候做出相应的动作，与消息映射机制相比，操作更简便，由于不存在占用系统消息的风险，安全性也有所提升。一个 signal 可以和多个 slot 相连接，增强了系统的执行效率。程序在点击、输入等操作以及多窗口的使用过程中充分利用了 signal/slot 机制的优势。

点击主界面的设置按钮可以打开参数设置界面(如图 4-16 左)，界面可以对摄像机参数、主程序位置，是否截图以及截图位置，算法灵敏度，还有检测线和正方向进行设置。点击主界面的 analysis 按钮可以进入统计图界面，每半个小时更新一组数据，用来显示每半个小时出入的客流量。点击生成报告可以生成 txt 版的数据报告，用于进行客流统计数据的存储和进一步分析。

图 4-16 参数调整界面(左)和统计界面(右)

2.通信设计

系统用来实现功能的程序(以后称为主程序)已经完成，主程序内部本身的耦合度已经较高，且有些库函数(如多线程库)与 QT 框架有冲突。如果直接将程序移植到 QT 界面程序中，需要对代码进行大规模的改动，因此需要设计一种新的模式来解决这个问题。本文提出的解决方案是保持两个程序整体框架不变，使用 Socket 通信机制实现前端和后端需要交换的信息的传递。

Socket 通信是一种网络进程通信的解决方案。与本地进程间通信相比，网络通信可以使程序的结构更加灵活，将多个子程序分布在各个子计算机，然后通过局域网连接到主机，形成星形结构，大大降低了主机的计算负担。同时 Socket 使用三次握手机制，降低了误码和丢包的风险。在本系统中，Socket 服务器端设在主界面程序中，socket 客户端设在主程序中，以保证一个界面可以对应多个主程序。

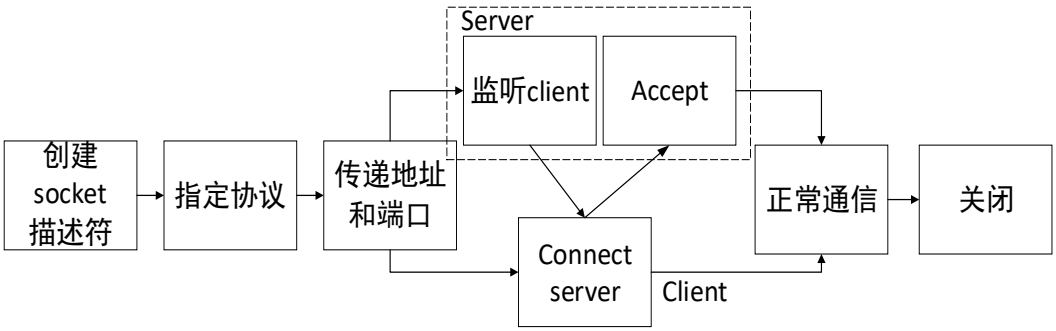

图 4-17 socket 通信机制

主界面程序的执行顺序如下。首先从 XML 文件中读取保存的参数，用这些参数信息写成 bash 文件(如图 4-18)之后打开一个终端界面，在终端上执行该文

件。这个 bash 文件的功能是在延迟 3s 后导入一个使用的数据库，之后进入主程序路径并打开主程序，从 main 形参处传入主程序初始化所需的参数。这些参数依次为：相机的 IP、端口、账号、密码、界面程序的 IP 和端口号，以及图片的存储路径。同时初始化主界面程序的通信服务器端口，保证服务器比客户端先打开。由于在 socket 通信中自带阻塞函数，需要保证通信在开始后一直通畅，一个程序中出现多个 socket 可能使得 socket 阻塞从而导致系统崩溃，所以每开一路相机需要增开一路 socket 通信构成循环。

```
#!bin/bash
#client_shell.sh
#sleep 3
export LD_LIBRARY_PATH="/home/junyu/Desktop/client2.0/HCNetSDKCom"
cd /home/junyu/Desktop/client2.0/build
./client 192.168.1.65 8000 admin 1234567890a 127.0.0.1 6666 /home/
```

图 4-18 bash 代码

由于每路相机只有一路 socket 通信，需要保证所有需要的信息都同时通过一路 socket 通信。在主程序发送向主界面的的通道中需要传递的信息包括处理完的图像和统计的人数，主界面发送到主程序的通道中需要传递设置面板中所有能修改的信息，包括检测线的数据，正方向，是否取图等。由于不是所有同时发送的信息都是此时的有效信息，需要额外发送一个指令来指示出当前要进行的操作。为了提高指令的容错能力和增强指令编辑的灵活性，指令集采用了二进制位编写。指令集如表 4-2。另外，在系统设置不变，正常传递图片的情况下，发送的指令值为-1.

表 4-2 信息指令集

数值 \ 数位	3	2	1	0
0	不改检测线	数据不清零	不取图	退出程序
1	改检测线	数据清零	取图	指令有变化

上文提到由于 socket 函数自带阻塞，故在指令没有改变的情况下仍需要保证服务器(界面程序)能持续向客户端(主程序)发送信息，即使用心跳包维持通信。这使得在本系统指令集下，服务器的指令在发送后，如果客户端不能及时处理，下一个收到的指令将会是-1，从而终止当前执行的指令，导致指令无法执行甚至引起程序的不稳定。为了消除这种情况，本系统设计了比较复杂的指令传输流程保证正常通信，如图 4-19：

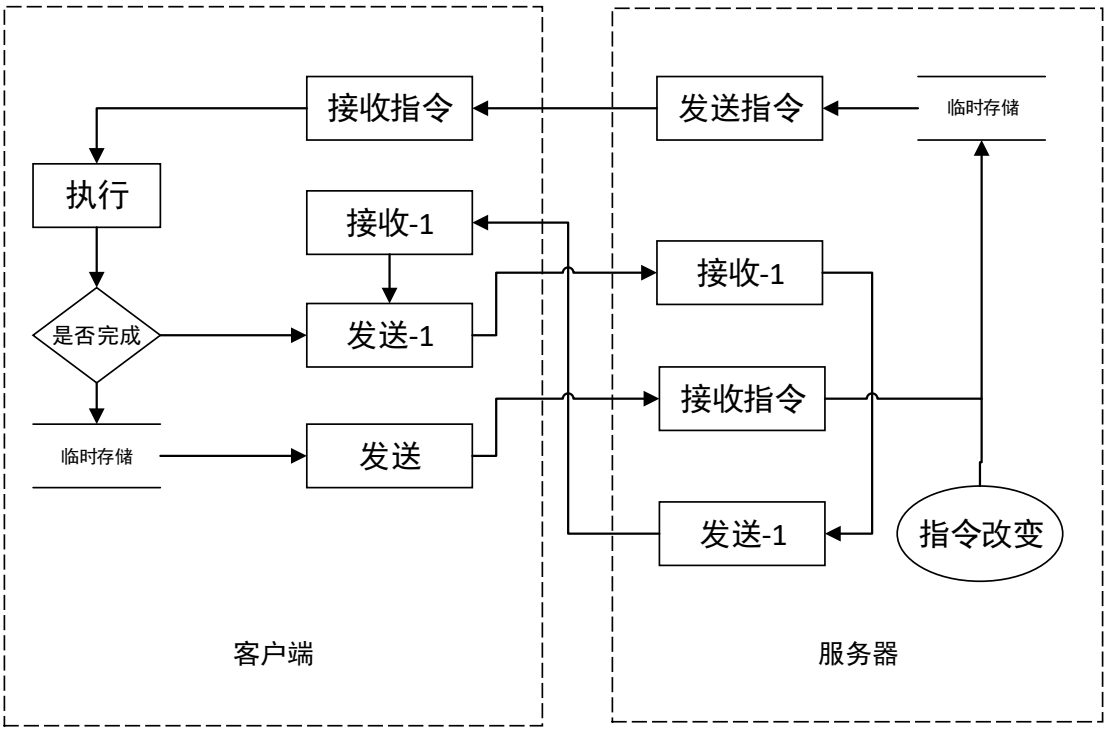

图 4-19 指令传递流程

最终，在本系统中，两个进程之间的数据流图如图 4-20：

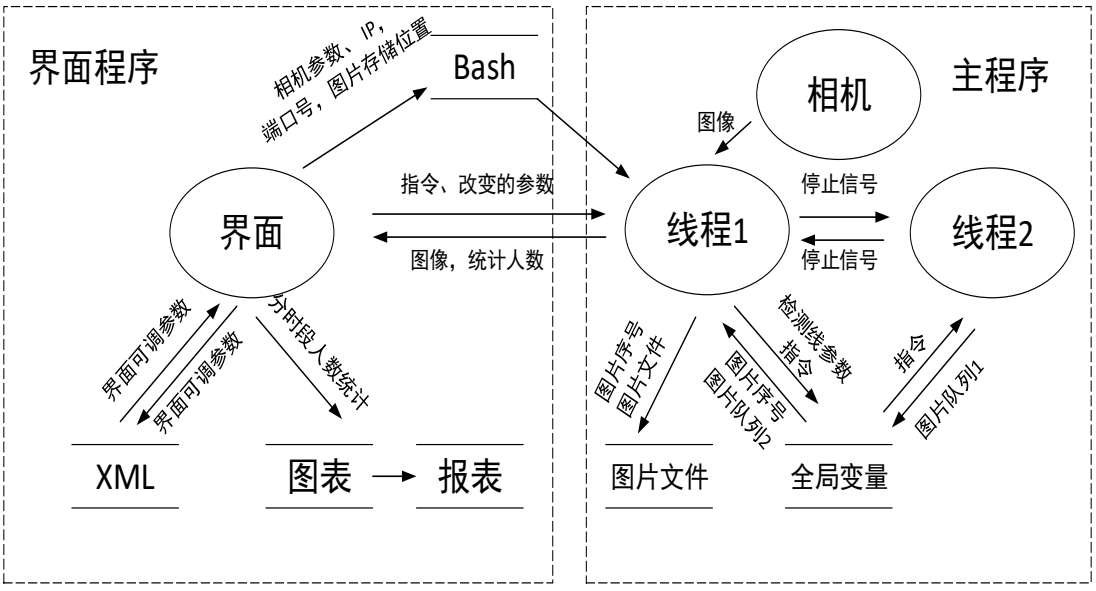

图 4-20 系统数据流图

4.5.2 鲁棒性设计

由于本系统需要实现长期正常运行，在完成功能设计之后，本系统进行了长期的稳定性测试，排查了导致程序长期运行崩溃的设计缺陷，并进行了处理。

1.queue 结构的重写

在设计之初，主程序的两个 queue 结构都是直接使用 STL(标准模板库)。在程序开发中，STL 库中包含的数据结构使用方便，封装完善，可以适应所有数据类型，但是由于内部的核心程序被封装，在工程中使用时，可能导致意想不到的错误。在本系统中实际使用的时候，由于网络的原因可能造成数据积压，在积压达到一定数量时观察到了 queue 的不正常清零，进而出现了图像的通道数错误的 Assert 报错。由 5.5 节内容可知，queue 大小为零时会导致两个线程同时对一块内存进行操作，从而导致程序崩溃。

本系统中编写了专用的 MatQueue 类，专门用来储存系统指定大小的 Mat 类矩阵，严格控制内存泄露，并在其中加入了控制 queue 大小和防止清零的机制。

2.边界处理

KCF 跟踪算法一个特点是在跟踪目标丢失后就会停在目标丢失的位置，使用 5.3 节的匹配算法时，行人出画面后跟踪框停在行人出画面的位置，这时如果在图像相近的区域出现新的目标被检测到，则两个框就可能被匹配，从而造成错检，但同时在边缘也需要找回跟丢目标。而如果目标在画面中没有被检测到，则跟踪框就会被认为离开了画面，而此时轨迹线是不完整的，从而会造成错误计数。

图 4-21 跟丢（左）和重新找回目标（右）

对于第一个问题，本系统的解决方案是划定图像的四边为边缘。当跟踪框中心超过边缘区域时，就认为目标已经离开，从而下次的检测框不再和这个目标进行匹配。对于第二个问题，本系统设定了二次检测的机制。如果跟踪框对应的目

标在画面的非边缘区域没有被检测到，则直接使用上一次的跟踪框来为下一次的跟踪器赋初始位置，如果下一次检测这个目标仍然没有被检测到，则认为目标已经离开画面，对目标轨迹进行相应的处理。

4.6 本章小结

本章首先介绍了整个系统的开发和工作的软硬件条件，本系统在 LINUX 环境下开发，需要 GPU 加速支持，使用的摄像头支持 H.265 网络传输，为系统的实时性提供了基础。系统的整体策略为定期检测，用检测的结果为跟踪算法赋初值，同时使用距离匹配算法的策略，保证了跟踪的连续性。系统的图像解码经过了多种格式的数据转换，最终转换为 OpenCV 库可以直接处理的 Mat 类型。本文设计了新式的行人通过方向判断模式，直接使用轨迹的起点和终点进行判断，节省了计算量，排除了对同一个人反复进出的重复计数。为了保证系统的实时性，引入了多线程机制，将图像的接收和处理分为两个线程。在经过长时间的系统压力测试之后，系统在实现基本功能后的界面化设计、算法改进和稳定性设计，避免了系统耦合度的增加，对行人更加复杂的轨迹分析能力增强，同时系统可以做到长时间稳定运行。到此为止整个系统的实现已经完成。

第 5 章 系统测试和验证

5.1 引言

本章首先介绍了对检测算法的检测器的训练过程，之后对跟踪和检测算法进行了实际验证和筛选。验证方式为用录制的测试视频以及视频中提取的图片对算法进行实测检验。视频 1 为正常光线人数适中，视频 2 为正常光线人流拥挤的情况，视频 3 为晚上光线很差，相机使用红外摄像的情况。这里程序开发使用的条件为：NVIDIA 750M 的 GPU，搭配 CUDA 7.5 运算平台。本章第二部分将本系统与一个市场上的同类产品进行了离线视频的效果比较，并记录了结果。

5.2 检测器的训练

在筛选检测算法之前，需要对检测器进行训练。基于深度学习的检测算法准确率与检测器的训练直接相关。由于从头开始训练检测器需要消耗大量的时间和采集海量的样本，考虑时间和成本，本项目借用了 caffe 维护团队提供的一个初始模型，在这个基础上进一步进行训练。初始模型为一个检测门类有 1000 类的大型神经网络，由于本项目需要的是对人的检测，因此使用大量人的样本继续训练，加强网络对人的识别能力。目前本项目已经拥有十余个在不同高度和场景下采集的行人视频。具体来说，拍摄视频的摄像头高度在 2 米至 5 米之间，角度接近垂直向下对准地面，场景有室内和室外，部分内容截图见图 5-1。

图 5-1 视频数据集截图

为了对模型进行训练，首先要写一个标定程序来采集样本。程序功能主要是：播放视频，在需要的位置按下空格视频暂停，然后再当前帧用鼠标将目标框出，。基于本项目的要求，需要标的是行人的俯视图信息，即以头和两肩为标志框其外

接矩形，以完成一个标准的头肩检测框，框选的质量直接影响训练出的检测器效果。为了保证准确度，只标定完全在画面中的人。

待完成一张图种所有行人的标定后，按下回车，视频继续播放，同时程序会将标出的检测框的信息存储到txt文件中。首先是图片名，然后是标签(p表示人)，后面数字是检测框坐标，依次为 x_min,y_min,x_max,y_max，即左上角和右下角坐标。

采集到的图片数量有限，为了增强训练效果，使用了图像扩充。最简单的扩充方法，就是对图片进行沿各个中心坐标轴的翻转，这里考虑到扩充图像数据应有意义，只沿以图像中心 x 轴和 y 轴对原始图像进行了翻转。这样数据集就扩充了 3 倍。而对应的检测框也只需要对应的进行翻转即可，不需要再次采集。

为了获得最适应本项目的图片用来提取样本，本系统在程序实现基本功能后，给程序添加了采集图片的功能，每当画面中检测出人的时候自动保存图片，用于今后提取样本对模型做进一步的训练。

首先要对使用的分类网络进行训练。训练采用 fine-tuing 的方法，即用之前提到的模型开始训练。因为基网络部分主要的作用是提取图像中的特征，而这个模型经过了大量的训练，网络前端对低级特征的提取已经做得相当成熟，所以训练主要会对网络提取人的特征的能力进行训练。

训练网络使用的代价函数为：

$$\mathrm{C}(\mathrm{w},\mathrm{b}) = \frac{1}{2N}\sum_{x} ||y(x) - a||^2 \tag{5-1}$$

其中，N 为训练用的样本数，w 和 b 分别是网络的权重和偏置，a 为输出层以 w 和 b 为变量的激活函数，代表输出的各个分类概率，y(x)是样本 x 对应的正确标签，训练的目标即为使代价函数达到最小。根据当前计算机的计算能力和各种优化方法的实际效果，这里选择了随机梯度下降法(SGD)方法，此时

$$\Delta\mathrm{C} = \nabla\mathrm{C} \cdot \Delta\mathrm{w} \tag{5-2}$$

为保证$\Delta\mathrm{C}$为负值，此处令$\Delta\mathrm{w} = -\lambda\nabla\mathrm{C}$即可，其中 λ 为学习率，反映了梯度下降的速度。根据经验，一般为了控制梯度的增长，防止发散和局部最优解，需要衰减权重，这里引入 L2 正则化方法，此时代价函数变为：

$$\mathrm{C}(\mathrm{w},\mathrm{b}) = \frac{1}{2N}\sum_{x} ||y(x) - a||^2 + \frac{\lambda}{2N}\sum w^2 \tag{5-3}$$

显然此时式（5-2）会引入一个负值项，抑制权重增长。权重的更新公式为：

$$w = \alpha w - \lambda \nabla C \qquad (5\text{-}4)$$

其中 α 为小于 1 的常量，一般取 0.9，用来减小 w 原值的影响。

接下来对 SSD 检测器进行训练。将网络的 FC 分类层替换为一系列卷积层，用来输出指定大小的分类和位置信息向量。同样使用 fine-tuning。λ 需要压得很低，设定为 0.001，使用 step 策略，即每经过 5000 次训练，学习率减少 10 倍，最终的学习率将达到 0.00001。动量项(momentum)设为 0.9，权重衰减(weight_decay)设为 0.0005，迭代总次数为 2 万次，使用 64 次迭代的梯度算平均值来更新一次参数。图 5-2 是迭代过程中代价函数(cost)变化曲线，每隔 20 次迭代显示一次。

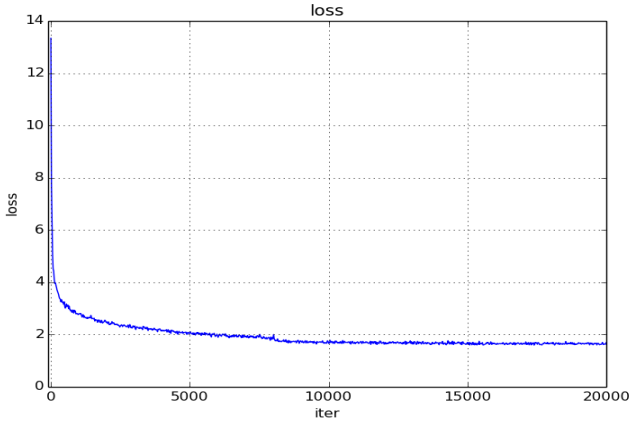

图 5-2 loss 随迭代次数变化

从图 5-2 可以看出，loss 起始很高，但是在前几百次迭代时迅速下降到 3 左右，后面迭代时 loss 缓慢下降，最终下降到了 2 以下，20000 次迭代时 loss 的准确值为 1.69689，这已经相当小的值了。在迭代到 2000 次时进行的第一次测试 mAP(mean average precision，代表检测器对每个分类检测准确度的平均值)就已经达到了 0.9，图 5-3 是程序是单独对迭代 20000 次的模型再次测试所得到的结果。可以看到，mAP 的值又有较大的提升。

```
I0226 19:52:06.700549  4360 caffe.cpp:155] Finetuning from /home/ztq/caffe-ssd-2/models/VGGNet/VOCPeopl
eCount1 /SSD_people_300x300/VGG_VOCPeopleCount1_SSD_people_300x300_iter_20000.caffemodel
I0226 19:52:07.125953  4360 net.cpp:761] Ignoring source layer mbox_loss
I0226 19:52:07.127382  4360 caffe.cpp:251] Starting Optimization
I0226 19:52:07.127403  4360 solver.cpp:294] Solving VGG_VOCPeopleCount1_SSD_people300x300_train
I0226 19:52:07.127410  4360 solver.cpp:295] Learning Rate Policy: multistep
I0226 19:52:07.127434  4360 blocking_queue.cpp:50] Data layer prefetch queue empty
I0226 19:52:07.783874  4360 solver.cpp:332] Iteration 0, loss = 1.77869
I0226 19:52:07.783916  4360 solver.cpp:433] Iteration 0, Testing net (#0)
I0226 19:52:07.799990  4360 net.cpp:693] Ignoring source layer mbox_loss
I0226 20:00:22.750957  4360 solver.cpp:546]     Test net output #0: detection_eval = 0.905163
I0226 20:00:22.751066  4360 solver.cpp:337] Optimization Done.
I0226 20:00:22.751075  4360 caffe.cpp:254] Optimization Done.
```

图 5-3 数据集训练结果

5.3 检测算法的验证

由第三章，选出的两种检测算法分别为 Faster R-CNN 和 SSD。两种算法的效果验证标准为如下两点：

a) 时间效率

b) 检测准确率

由于检测不需要使用连续的视频，所以检测的方案为：在各种情况的视频分别截取 10 张图片，编写测试程序，对截图进行检测，分别计数每个场景下所有目标的总数和没有检测出来或者错检的目标总数，图片中显示不全的目标不计入。同时将每帧耗时显示在控制台。这里本项目同样调用 clock()函数计时。检测的一个示例如图 5-4。两种检测算法在准确率上都获得了比较好的效果，但实际在计算量上两种算法差异较大，处理图像的 FPS 也相差较多。

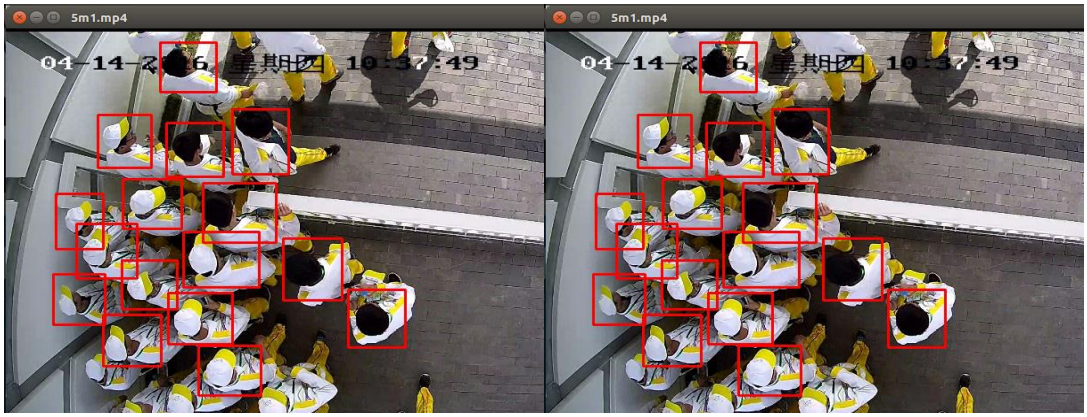

图 5-4 SSD(左)和 Faster R-CNN(右)检测效果比较

最终得到的结果如表 5-1 所示：

表 5-1 对 SSD 和 Faster R-CNN 的效果比较结果

		截图 1(正常)	截图 2(拥挤)	截图 3(夜间)
SSD	准确率	53/53	182/186	30/30
	时间(tick)	334582	367421	354123
Faster R-CNN	准确率	53/53	180/186	30/30
	时间(tick)	523710	582340	564265

由于检测算法耗时很长，接近甚至超过了图片的传入速度，会明显影响到系统的实时性，必须综合考虑时间和准确率因素，因此需要构造一个比值来对比两

种算法的效率：

$$\frac{\eta_{RCNN}}{\eta_{SSD}}=\frac{\mathrm{t}_{\mathrm{SSD}}\cdot acc_{RCNN}}{\mathrm{t}_{\mathrm{RCNN}}\cdot acc_{\mathrm{SSD}}}\times 100\% \tag{5-5}$$

t 和 acc 分别表示两种方法的时间和准确率。若结果大于 1，则 Faster R-CNN 方法更好，反之则为 SSD。这里的准确率直接使用漏检情况最严重的视频二截图对应数据。得到结果约为 0.673。因此认为 SSD 算法的效率更高，选用 SSD.

5.4 跟踪算法的验证

由第二章，选出的两种跟踪算法分别为 KCF 和 DSST，需要比较的点如下：

a) 时间效率

b) 跟踪效果，特别是遮挡、光线较差的情况

验证的方案为：先录制测试视频，之后写出一个测试程序，打开指定测试视频后用鼠标在第一帧画出初始跟踪目标的位置，之后开始播放视频，逐帧跟踪，画出跟踪轨迹和每帧中的目标框。当跟踪目标丢失的情况下，跟踪线和画出的框会消失。计时方面，由于目前的跟踪算法都只能做到每个跟踪器处理一个跟踪目标，每帧消耗的处理时间会与目标数成线性关系。本项目在每个跟踪循环内部的前面后后面各调用一次 clock()函数，用两个数值之差对应的 CPU tick 数来表示每次跟踪器更新所用的时间的相对大小，根据经验，每个 tick 大致对应 1μs.取 3 次调用的平均时间作为算法的代价时间。最终得到的结果如表 5-2 所示：

表 5-2 对 KCF 和 DSST 的效果比较结果

		视频 1(正常)	视频 2(拥挤)	视频 3(夜间)
KCF	准确率	5/5	17/17	3/3
	时间(tick)	66290	226508	40263
DSST	准确率	5/5	15/17	3/3
	时间(tick)	64915	194940	39081

可以看到，在拥挤画面的情况下，DSST 算法的准确度出现了问题。根据实际情况，两种跟踪算法平均到每个目标的时间效率只相差 0.03s，小于 25 帧实时视频流的图片传入时间间隔 0.04 秒，因此跟踪时间差别的影响可以忽略，本项目可以只考虑准确率的影响。可以认为，在本项目的工作环境下，KCF 算法的效果要略胜一筹，因此本项目选择 KCF 算法作为跟踪算法。

5.5 效果检验

1.离线视频检验

首先进行了同一目标的长时间跟踪测试。实际工作中，行人在画面出现时间很短，因此本系统定义只需要对同一目标持续跟踪 15s 以上即满足要求。在一个测试视频中存在一个自始至终都出现的目标（图 5-5 右下角），经过测试，在整个 28s 的视频中经过了较大量客流，期间目标始终被有效跟踪，证明系统对同一个目标长时间跟踪效果满足要求。

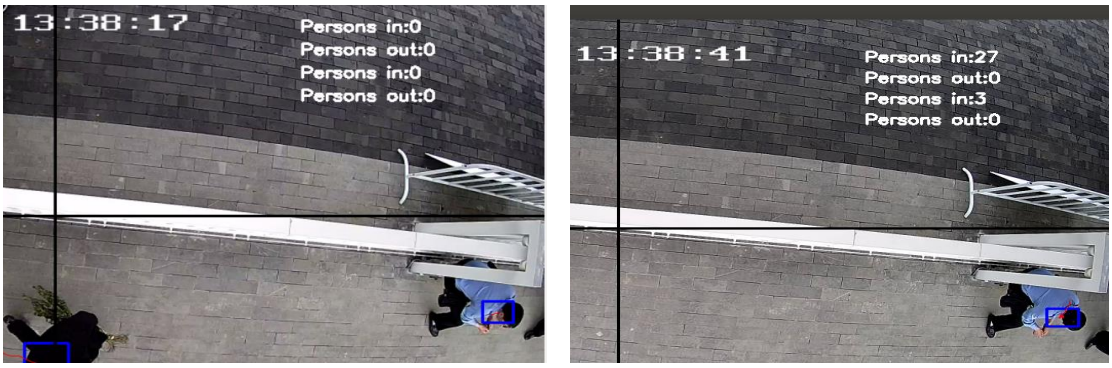

图 5-5 长时间目标跟踪效果

为了将本系统的算法与传统算法进行可量化的比较，需要保证使用的视频可以用人工进行反复验证，并且效果对比主要是针对准确率而不是实时性，所以这里需要使用录制好的离线视频，而没有使用实时视频流。同时为了使比较结果明显，采用了画面上直接带有传统算法统计数据的视频，而本系统的算法统计结果也会显示在画面上相邻的位置，方便肉眼进行比较。另外，一些在平时的实时统计中比较难以出现的特殊情况，也可以使用离线视频的来模拟，从而对算法进行验证和处理。

为了对两种方案进行量化的效果比较，定义准确率为进入人数准确率和离开人数准确率的平均值：

$$Acc = \frac{1}{2}\left(\frac{正确统计进入人数}{实际进入人数} + \frac{正确统计离开人数}{实际离开人数}\right) \times 100\% \qquad (5\text{-}6)$$

系统中可能出现由于核心算法问题而漏掉的目标，也可能出现在跟踪过程中跟丢的目标，而且还可能出现由于第一次被检测到的时间过晚造成被判断为没有过检测线。另外，也可能出现过线后方向判定错误的情况。为了给各种情况赋予

相同的权重，并增加错误的可视化数值来提高可比性。定义错检率为：

$$Err = (\frac{漏检目标数}{总人数} + \frac{错检进入人数}{实际进入人数} + \frac{错检离开人数}{实际离开人数}) \times 100\% \quad (5\text{-}7)$$

使用两个测试视频分别进行测试，两个视频的截图见下图.画面左上角黑色字体为传统算法得到的结果，白色字体为使用本系统算法实际处理视频得到。由于视频的画面不是从初始时间开始记录，故在计算传统方法的统计人数时需要减去视频第一帧的值。

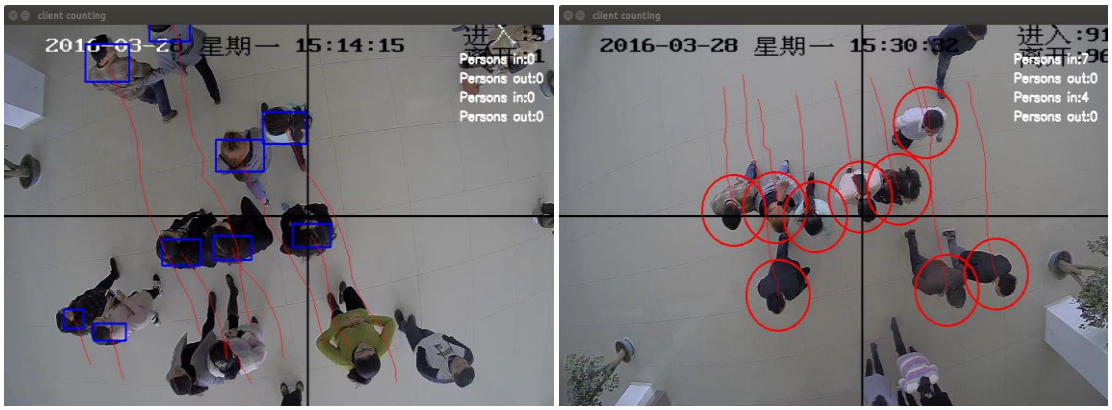

图 5-6 对比测试用视频

对比效果如表 5-3，由于缺乏对比方法录制时的实际帧率和当前播放视频的帧率之比，这里的平均处理时间只统计本系统的，具体方法为在视频中随机采样 10 张图求平均处理时间计算 FPS，达到实时性即认为满足要求。

表 5-3 对比测试效果

客流统计方法	测试视频	准确率 (%)	错检率 (%)	平均处理时间(ms)
传统	视频一	60.17	39.53	-
	视频二	64.72	35.28	-
本系统	视频一	100	0	42.23
	视频二	94.12	5.88	49.10

视频中的行人彼此之间距离很近，而传统算法对近距离目标的区分度很差，造成这两个视频中使用传统算法的统计效果很差。之后使用拥挤人群的测试视频对系统进行了离线压力测试，检测系统对密集人群的区分效果以及处理的实时性。

图 5-7 压力测试视频

整个视频通过 123 人，跟丢 2 人，测试效果如表 5-4：

表 5-4 压力测试结果

准确率 (%)	错检率 (%)	平均处理时间(ms)
98.37	1.63	125.74

2.在线视频测试

经过离线测试之后，对在线视频的测试主要集中在实时性上。经过长期测试，该系统的处理帧率在 8.3~50 之间浮动，且在待处理图像队列积压的图像达到 300 张的情况下，在 3s 内可以将待处理队列缩减到正常的 3 张。

图 5-8 实时测试效果

5.6 本章小结

本章首先给出了对检测算法的检测器的训练过程，训练结果中，SSD 检测器的 mAP 达到了 0.9 以上，效果很理想。在训练之后，对跟踪和检测算法进行了实际验证和筛选过程。验证方式为用录制的测试视频以及视频中提取的图片对算法进行实测检验。对跟踪算法的验证侧重于算法的准确性和对复杂条件的适应性；而检测算法的实时性则是重点考虑的条件。经过检验，选出了准确性更好的 KCF 跟踪算法和实时性、准确性的综合效果较好的 SSD 检测算法。本章对系统的长时间跟踪效果进行了测试，在 24s 的视频中对一个目标进行了有效跟踪，达到设定要求。随后本章将完成后的本系统与一个市场上的同类产品进行了离线视频的效果比较，本系统的准确性比对比产品高出 34.6%，同时在画面中人数不超过 5 人的情况下处理速度达到了 30fps，满足了实时性的要求。最后，经过长时间实时视频流测试，在待处理图像的队列积压 300 张图像的情况下，在画面人数减少到 3 人以下后，可以在 8s 时间内恢复队列中只有 3 张待处理图像的最小值。

结　论

本文从目前客流统计系统的需求和相关技术发展出发，围绕系统的实时性和可靠性，对设计的系统进行了详细的分析和介绍，其主要包括的内容有：

第一，从发展的角度对当前具有代表性的跟踪和检测算法进行了研究，针对每类方法所存在的问题进行了详细的阐述，从而确定了本系统在设计中可以利用的方法体系。

第二，针对各种检测和跟踪算法的特点和不足进行了整合重构，搭建了一个综合运用检测和跟踪算法的框架，充分协调了准确性和实时性的矛盾。用实际系统测试的方式，从当前效果较好的跟踪算法和检测算法中选出了效果最好的方法代入框架。

第三，在设计的框架基础上进行了系统各个部件的设计。使用自行设计的匹配算法有效的将核心算法结合起来，运用多线程设计解决了视频流实时解码与图像处理互锁的矛盾，创造新的轨迹判定方法提高了对复杂行人轨迹的判定准确度，利用数据的定期清零机制维持了系统的长期稳定运行。

第四，对系统进行了拓展和稳定性设计。为系统设计了界面，增加了数据处理和输出的功能，利用通信机制解决了强耦合性的问题。对系统进行长期稳定性测试之后，改进了系统的数据结构，优化了系统的轨迹判定算法，提高了系统的鲁棒性，从而使系统所具有的客流统计算法统计精度高、算法实时性好，长期运行稳定，达到了比市场上现有产品更好的效果。

由于目前主流跟踪算法只能实现单目标跟踪，对每个目标都需要使用一个跟踪器，实际每帧处理时间会受到跟踪人数影响。一旦出现有效的多目标跟踪算法，系统的实时性将进一步增强。

参考文献

[1].姚树春. 计算机视觉技术的推广与应用[J]. 中国新通信. 2016 年第 17 期

[2].V Bonato，E Marques，GA Constantinides. A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection [J]. Circuits and Systems for Video Technology，2008 年 18 卷 12 期

[3].管皓，薛向阳，安志勇. 在线单目标视频跟踪算法综述[J]. 小型微型计算机系统. 2017 年第 38 卷第 1 期

[4].Na Wang，Guo-Yu Wang. Shape Descriptor with Morphology Method for Color-based Tracking[J]. International Journal of Automation and computing. 2007 年第 4 卷第 1 期

[5].刘军学，屈桢深，任行行，郭隽，闻帆. 基于改进运动历史图像的多运动目标实时跟踪[J]. 计算机应用. 2008 年第 28 卷第 B06 期

[6].虞旦，韦巍，张远辉. 一种基于卡尔曼预测的动态目标跟踪算法研究[J]. 光电工程. 2009 年第 36 卷第 1 期

[7].DS Bolme，JR Beveridge，BA Draper，YM Lui, Visual object tracking using adaptive correlation filters[J], Computer Vision & Pattern Recognition,2010 年,119 卷第 5 期

[8].JF Henriques，C Rui，P Martins，J Batista，High-Speed Tracking with Kernelized Correlation Filters[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence. 2015 年 37 卷第 3 期

[9].M Danelljan，G Häger，FS Khan，M Felsberg, Accurate Scale Estimation for Robust Visual Tracking[R], British Machine Vision Conference, 2014

[10].Sánchez A, V.David. Advanced support vector machines and kernelmethods[J]. Neurocomputing,2003,55 卷,1 期

[11].LeCun, Y，Bengio, Y，Hinton，G. Deep learning[J]. NATURE. 2015 年 5 月第 521 卷第 7553 期

[12].孙志远，鲁成祥，史忠植，马刚.深度学习研究与进展[J].计算机科学.2016 年第 43 卷第 2 期

[13].孙志军，薛磊，许阳明，王正. 深度学习研究综述[J]. 计算机应用研究. 2012 年第 29 卷第 8 期

[14].邱元阳.从 AlphaGo 看深度学习[J].中国信息技术教育 2016 年第 7 期 22-22 页
[15].刘栋，李素，曹志冬. 深度学习及其在图像物体分类与检测中的应用综述[J]. 计算机科学. 2016 年第 43 卷第 12 期
[16].Ding, Jianwei，Huang, Yongzhen，Liu, Wei，Huang, Kaiqi, Severely Blurred Object Tracking by Learning Deep Image Representations[J]. IEEE Transactions on Circuits and Systems for Video Technology. 2016 年第 26 卷第 2 期

哈尔滨工业大学本科毕业设计(论文)原创性声明

本人郑重声明：在哈尔滨工业大学攻读学士学位期间，所提交的毕业设计(论文)《多区域客流统计系统研究》，是本人在导师指导下独立进行研究工作所取得的成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明，其它未注明部分不包含他人已发表或撰写过的研究成果，不存在购买、由他人代写、剽窃和伪造数据等作假行为。

本人愿为此声明承担法律责任。

作者签名：　　日期：　　年　　月

致　谢

衷心感谢导师王常虹教授对本人的精心指导。他对我的学业和工作提出了很多建设性的意见，关心我的发展，时刻督促我的进步，他的言传身教将使我终生受益。

感谢屈桢深副教授，对我进行了耐心的指导，为我提供了良好的学习和研究环境，并创造了一个又一个磨练自己、打开眼界的机会。

感谢我的家人，尽全力为了家而奋斗，并在我二十多年的生命里时刻全力支持我的发展。生养之恩，当用一生报答。

感谢董韵佳同学，在我最迷茫的时候指明了我应该前行的方向，帮助我走出阴霾，在大学剩下的时光里走出了一路完全不同的精彩。

感谢陪伴了我四年的同学们。有了他们，我在不断攀登的路上没有感到孤独。特别感谢我的舍友：李繁荣、邱果、严培熠、于翔宇，和他们的兄弟情谊我永远不会忘记。

感谢实验室同学们的热情帮助和支持。周纪强、张天琦、梁亮、楚翔宇，学长们的谆谆教诲让我收获颇丰，言谈举止之间都为我做出了榜样。我会像他们一样，为了自己的梦想继续不断前行。徐超凡、程兴、江丽、潘健岳，大四的同学们和我一起经历了半年多的成长，也已经成为了在成长路上相互搀扶鼓励的战友，感谢他们在平时对我的照顾和帮助。