

中文图书分类号：TP391

密 级：公开

UDC：004

学 校 代 码：10005



硕士专业学位论文

PROFESSIONAL MASTER DISSERTATION

论 文 题 目：多场景视频智能处理系统及调度管理算法
研究

论 文 作 者：冯小琴

领 域：软件工程

指 导 教 师：黄樟钦

论文提交日期：2019 年 5 月

UDC: 004
中文图书分类号: TP391

学校代码: 10005
学 号 : S201625102
密 级 : 公开

北京工业大学硕士专业学位论文

(全日制)

题 目: 多场景视频智能处理系统及调度管理算法研究
英文题目: RESEARCH ON MULTI-SCENE VIDEO
INTELLIGENT PROCESSING SYSTEM
AND SCHEDULING MANAGEMENT
ALGORITHM

论 文 作 者: 冯小琴
领 域: 软件工程
研 究 方 向: 物联网与嵌入式系统
申 请 学 位: 工程硕士专业学位
指 导 教 师: 黄樟钦
所 在 单 位: 软件学院
答 辩 日 期: 2019 年 5 月
授 予 学 位 单 位: 北京工业大学

独 创 性 声 明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京工业大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签 名: _____
日 期: 年 月 日

关于论文使用授权的说明

本人完全了解北京工业大学有关保留、使用学位论文的规定，即：学校有权保留递交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

(保密的论文在解密后应遵守此规定)

签 名: _____ 日 期: 年 月 日

导师签名: _____ 日 期: 年 月 日

摘要

随着时代的进步和经济的高速发展，社会安全引起了人民的重视。提高视频监控信息化水平，促进新兴科技在视频监控领域的应用，已成为并将长期作为社会安全工作的重点。视频智能处理旨在解决视频监控人为监督、分析的现状，将计算机视觉和图像分析技术，集成到视频处理系统，完成多样化的需求。多路视频智能处理，具有数据量大、算法复杂度高的特点，面对严苛的实时性需求以及软硬件资源的限制，提出一个多场景视频智能处理系统解决方案具有重要的研究意义。本文的主要工作如下：

根据视频处理的重复性和时序性特点，研究并提出了基于并发流水线的 MSSM 视频调度管理算法。为了保证面向不同应用场景的扩展性，算法部署在通用视频前端预处理过程中，包括整体并行化方案设计、并发流水线的算法架构、视频调度任务槽设计、数据共享设计、基于任务复用的任务调度算法设计、视频存储管理以及视频智能跳帧算法等，本文称之为场景规则下的视频调度管理算法（MSSM）。结合视频智能处理实时反馈，对任务和数据进行调度和管理，达到合理调度系统资源，视频处理信息最大化的目标，

基于调度管理算法的场景性因素引入，本文针对单帧人数统计这一应用场景，研究并优化了基于轻量级网络的视频智能处理算法。采用 MobileNet 系列作为特征提取网络，引入小尺度特征以及全局信息；使用优化的 SSD 作为后端检测框架，在精度和效率上进行均衡。其一作为调度管理算法的数据基础，满足一定的实时性，其二也为深度学习模型在边缘终端部署提供了参考，达到了更低的延迟、更少的模型参数以及优良的鲁棒性。

最后，针对本文提出的多场景视频智能处理及调度管理相关算法设计，提出了基于边缘计算的多场景视频智能处理系统解决方案。搭建基于单帧人数检测的多路视频智能处理系统。提炼需要验证的算法模块或者方案设计，结合相应的软硬件优化技术，对系统的性能和效果进行测试，并对结果进行了分析和总结，达到多路视频处理系统的高性能，稳定运行的目标。

关键词：高清视频监控；调度算法；智能处理；边缘计算；并发流水线

Abstract

With the rapid development of economy, social security has attracted more and more attention over time. Improving the level of video surveillance and promoting the application of emerging technologies in the field of video surveillance have become and will continue to be the focus of social security work for a long time. Video Intelligent Processing aims to solve the current situation of human supervision and analysis of video surveillance. It integrates computer vision and image analysis technology into video processing system to meet diverse needs. Intelligent multi-channel video processing has the traits of large amount of data and high complexity. Facing the real-time requirements and the limitation of hardware and software resources, it is of great significance to propose a solution of intelligent multi-scene video processing system. The main work of this paper is as follows:

According to the repeatability and time sequence of video processing, the MSSM video scheduling management algorithm based on concurrent pipeline is put forward. In order to ensure the expansibility to different application scenarios, the algorithm is deployed in the pre-processing process of general video front end, including the overall parallelization scheme design, the algorithm architecture of the concurrent pipeline, the video scheduling task slot design, the data sharing design, and the task scheduling algorithm design based on task reuse, video storage management and video intelligent frame skipping algorithm, etc., which is called video scheduling management algorithm (MSSM) based on scenario scheduling. Combining with the real-time feedback of video intelligent processing, the tasks and data are scheduled and managed, so as to achieve the goal of reasonably scheduling system resources and maximizing information of video processing.

Based on the introduction of scenario factors of scheduling management algorithm, this paper studies and optimizes the video intelligent processing algorithm based on lightweight network for the application scenario of single-frame population statistics: MobileNet is used as feature extraction network, small-scale features and global information are introduced, optimized. SSD is optimized as back-end detection framework to balance accuracy and efficiency. The algorithm can meet the requirement of instantaneity as the data foundation of the scheduling management algorithm as well as provide the reference by the deep learning model to deploy in the edge terminal,

achieving lower latency, fewer model parameters and better robustness.

Finally, aiming at the multi-scenario video intelligent processing and algorithm design related to scheduling management proposed in this paper, a solution of multi-scenario video intelligent processing system based on edge computing is proposed. By building multi-channel video intelligent processing system based on single-frame population, extracting the algorithm module or scheme design that needs to be verified, combining with the corresponding software and hardware optimization technology, testing the system performance and effect, and analyzing and summarizing the results, the goal of high performance and stable operation of multi-channel video processing system is achieved.

Keywords: HD video surveillance, Scheduling algorithm, Intelligent processing, Edge computing, Concurrent pipelining

目 录

摘 要	I
Abstract	III
第 1 章 绪论	1
1.1 课题背景及研究意义	1
1.2 国内外研究现状	2
1.2.1 高清视频监控研究现状	2
1.2.2 视频智能处理研究现状	3
1.2.3 调度管理算法研究现状	6
1.3 本文主要内容	7
1.4 论文组织结构	8
第 2 章 高清视频处理及深度学习相关技术	9
2.1 高清视频前端处理相关技术	9
2.1.1 高清视频流采集	9
2.1.2 高清视频编解码	9
2.1.3 高清视频存储	10
2.1.4 图像预处理	11
2.2 边缘计算相关技术	12
2.3 深度学习加速相关技术	13
2.3.1 深度学习介绍	13
2.3.2 加速相关技术	13
2.4 本章小结	14
第 3 章 多场景高清视频处理系统的总体设计	15
3.1 总体分析	15
3.1.1 研究对象	15
3.1.2 需求分析	16
3.2 系统架构设计	17
3.3 系统总体设计	19
3.4 本章小结	20
第 4 章 基于并发流水线的 MSSM 视频调度管理算法研究	21
4.1 并发流水线性能分析	21

4.2 MSSM 总体架构	23
4.2.1 并行化方案设计	24
4.2.2 算法总体架构	25
4.3 MSSM 相关算法设计	26
4.3.1 视频调度任务槽设计	26
4.3.2 数据共享设计	27
4.3.3 基于任务复用的任务调度算法设计	28
4.3.4 视频存储管理算法设计	29
4.3.5 视频跳帧采样算法设计	31
4.4 本章小结	33
第 5 章 基于轻量级网络的视频智能处理算法研究	35
5.1 单帧人数统计算法对比和不足	35
5.1.1 传统机器学习方法	35
5.1.2 深度学习方法	36
5.2 头部检测网络优化设计	36
5.2.1 MobileNet 特征提取网络	37
5.2.2 SSD 目标检测网络	39
5.2.3 评价标准	41
5.3 算法验证	42
5.3.1 实验环境及数据集制作	42
5.3.2 网络模型训练和测试	44
5.3.3 实验结果分析	45
5.5 本章小结	46
第 6 章 多场景视频智能处理系统实现与测试	47
6.1 系统开发环境	47
6.2 关键模块设计与实现	49
6.2.1 参数结构设计	49
6.2.2 任务复用设计	50
6.2.3 共享数据设计	51
6.3 系统前端设计	51
6.4 系统实验及结果分析	52
6.4.1 视频解码性能对比实验	52
6.4.2 视频增量跳帧算法对比实验	56

6.4.3 图像数据获取对比实验	57
6.4.4 系统整体效果	59
6.5 本章小结	60
结 论	61
参 考 文 献	63
攻读硕士学位期间取得的研究成果	67
致 谢	69

第1章 绪论

1.1 课题背景及研究意义

随着新一代信息技术的发展，以及城镇化加速进步的社会现状。智慧城市借助着物联网、大数据、云计算、人工智能等技术，应用于城市智能化服务，实现对城市各个领域的精细化管理，集约化的利用城市资源^{[1][2][3]}。其中，智慧城市应用领域里的智慧安防、视频智能监控成为了研究重点。视频智能监控主要是为了解决视频大数据下人为监督、分析，其耗费时间长，成本高，且效果一般的问题^[4]。为了加强社会安全防范，实现全方位监控，监控摄像机的数量、覆盖面积在不断增加，遍布在城市的每个角落，采集图像数据，监督着城市的安全隐患^[5]。利用图像智能分析技术，实时的、高效的提取出视频里的有效信息，结合相应的预警机制，提高报警的精确度和响应时间，实现无人监督的视频智能监控系统，为社会安全等提供更强大的屏障，真正实现智慧城市、智慧安防。

视频智能监控包括高清视频监控和视频智能处理两个关键组件。高清视频监控是指高分辨率、高质量的视频数据采集、存储和展示，是视频智能处理的数据来源之一^[6]。利用图像分析技术，对视频监控的图像序列进行高速的分析和理解，代替人力进行不间断的全方位监控。后期还能及时调出对应的视频内容，用于违法场景佐证等^[7]。视频大数据的产生以及分析任务的复杂性，促使人们不再满足于单路摄像机采集的单源视频分析，更倾向于分析多源视频，从而获取大场景和海量视频中包含的深层有效信息^{[8][9]}。

基于云计算的视频智能处理系统主要由视频采集终端和云端视频处理服务器组成。将视频处理分析放在云端服务器，资源充足，可支持多路视频并行处理，且速度极快^{[10][11]}。但存在几类问题：①视频转发：应用场景如某些地理位置较高的位置，存在无网络或者网络较差的情况，视频无法转发，不能保证实时处理；此外，高清视频数据量大，网络质量不稳定导致传输效率和传输质量得不到保障②信息安全：应用场景如某些机密场所，视频信息涉及个人隐私或者企业隐私，视频被拦截或者篡改，都将造成不可预知的后果③视频处理任务集中在云端处理，增加了云端服务器的计算压力，视频处理延时较大。边缘计算应运而生，将计算任务部署至就近的计算节点，就近的提供信息互联、数据交互服务，满足业务实时、智能、安全与隐私保护等关键需求等^{[12][13]}。虽然一些视频终端的边缘节点并没有计算能力或者部署成本很高，但是结合软硬件技术，合理的调度和管理视频处理任务，将有效的提高整个视频智能处理的效率和精度，这是现在及未来持

续的一个研究热点。

本文的研究课题也即是边缘计算下的多场景视频智能处理系统。其主要技术难点在于：①视频结构化数据量大，处理方式费时且复杂，视频采集，编码，传输，解码，存储都会占用较大的资源，且耗时长短不一；②不同的后期任务需求，比如目标检测或者目标识别，往往需要基于深度学习的算法模型才能保证其效果。但是深度学习的模型结构复杂，参数巨大，很难满足任务的实时需求；③不同场景下的边缘视频智能处理管理和调度，其场景的规则没有加入系统设计考虑，算法在不同场景下运用不够灵活；④目前，深度模型在硬件部署存在两个问题，其一是硬件资源有限，模型计算缓慢或者无法运行；其二是，模型结构多样化，目前只有相对少的产业实现深度学习模型的定点化部署，配置复杂且成本昂贵。

综上所述，本文提出了一个多场景视频智能处理系统的解决方案，具有一定的研究意义和现实意义。其主要研究及解决的问题要点是：①在不同的应用场景下，其视频处理需求如人脸检测、行为分析、轨迹分析等，任务存在一致性，也即是针对视频中的运动目标做一系列的检测和分析，其中运动目标的检测是视频智能处理的前导步骤。本文提出面向多场景规则的调度管理算法（MSSM: Multi-Scenario Scheduling Management），将利用这一特性进行设计，具有场景扩展性；②根据多路视频处理过程的重复性和时序性，提出并发流水线的算法架构设计，结合任务复制、数据共享、视频跳帧等技术，协调多路视频并行处理，降低视频处理的冗余，达到视频处理信息最大化的目标；③针对系统第二大组件——视频智能处理算法，本文研究优化一个基于轻量级网络的视频智能处理算法，其在效率和精度上有一个很好的均衡。在系统中扮演两个角色：其一作为系统调度管理算法的数据基础，满足一定的实时性需求；其二，为视频智能处理算法在边缘终端移植部署，提供一个算法选择方案，具有可延展性；④提出一套多场景视频智能处理系统的解决方案，并搭建对应的实验平台进行算法验证和性能分析。

1.2 国内外研究现状

1.2.1 高清视频监控研究现状

视频监控的发展经历了从模拟到数字、低清到高清、人工到智能化的不同阶段。依托于不同的行业高清标准、前端采集技术、高清编码标准、高清传输技术、高清显示技术、人工智能技术等发展^[6]。国外视频监控技术研究早在 19 世纪 60 年代开始，应用领域主要包括国防领域、工业生产、社会安全等。期间 Intel 公司致力于视频智能监控项目，并最早提出视频监控智能化的概念，成为行业领域的

领头羊。国内由于科技水平的限制，研究起步较晚，随着经济的迅速发展以及科技投入的加大，相继涌现出各类研究成果。目前，中科院自动化研究所模式识别国家重点实验室，开展并成立了视频智能监控系统研究组。为了促进本国科研人员的研究热情，国内很多研究所、大学等都举办了一些学术会议，分享最新研究成果。如北京理工大学、北京航空航天大学等相继加入研究。

高清技术的发展推动着智能图像处理等技术的发展。自二十世纪开始，高清视频监控迎来了重大的转型，旨在将高清视频监控与人工智能结合，克服传统视频监控清晰度低、成本大以及维护困难等问题，已广泛应用于社会安全、交通调度、国家安防等各个领域，掀起了构建智慧城市的热潮^[2]。以下是目前主流高清视频监控结合人工智能的应用场景：

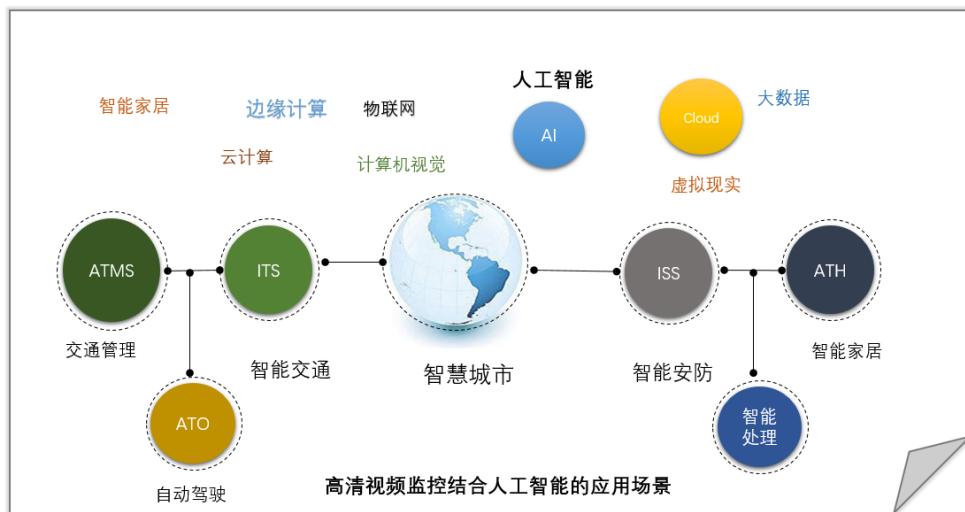


图 1-1 高清视频监控+AI 应用场景

Figure 1-1 HD video surveillance + AI application scenario

1.2.2 视频智能处理研究现状

目前，多场景下的视频智能处理数据来源大多来自于社会视频监控。系统的基本功能包括视频的采集、传输、存储、显示等功能，而视频内容的监督和分析需要人工实现，耗费大量人力物力同时，错误率和延时都很高。视频智能处理是指借助计算机强大的处理能力，结合计算视觉图像处理技术，对摄像机拍摄的图像序列行自动分析，抽取视频场景中的目标进行检测、跟踪等，提高工作效率。以下为国内外几种比较典型的视频智能处理技术研究现状：

(1) 运动检测技术(Motion Detection)

运动检测相对与静态目标检测，通常会利用视频帧间的联系，正确分割出运动目标区域或轮廓。其检测效果直接或者间接影响后续的分析工作，如目标跟踪、

异常检测、行为分析等，是视频智能处理的前导步骤。常见的运动检测方法分为背景检测方法和目标检测方法^{[14][15]}。背景检测方法又进一步分为四种：背景统计法、背景更新法、卡尔曼滤波法以及背景建模法^[16]；目标检测算法常用的有光流法、帧间差分法、背景减法^[13]等，这三类针对运动目标检测有比较好的效果，常常作为前导模型，用以初步定位，再结合基于深度学习的目标检测方法进行精准定位。此外，目标检测在深度学习领域上也有很大的发展，主要分为两类：基于区域的检测算法（Faster-RCNN 等）、基于回归的检测算法（YOLO 等）。其在检测精度和效率上一直有质的提升。

（2）目标跟踪技术(Object Tracking)

目标跟踪技术是计算机视觉的一项重要任务，是指对视频的图像序列进行计算分析，对其存在的目标状态位置持续进行推断的过程^[17]。通过对图像进行目标检测和精准定位，生成目标在视频序列中的运动轨迹。进而判断目标的移动范围和方向。其应用领域包括无人飞行器、人机交互、疑犯追踪等。

（3）行为识别技术(Behavior Understanding)

行为识别技术是视频智能处理的关键技术之一。首先，通过目标检测技术提取出目标在图像的准确位置，结合目标跟踪技术得到其运动轨迹，最后对其运动模式进行进一步分析和识别。因为考究到广泛的技术和知识点，其在人工智能、计算机视觉领域一直是研究热点亦是难点^[18]。目前技术仍然属于初级阶段，划分为两大流派：① Two-Stream:融合图像本身的静态特征以及视频帧的连续时序特征，进行行为识别^[19]；② C3D：由 Facebook 提出的基于深度学习的端到端行为识别网络。两类算法都是基于深度学习提出^[20]，相对与传统方法（iDT），在精度和速度上都有很大的提高。通过对其行为轨迹、速度等的进步挖掘和分析，其警情预测、医疗监护以及环境检测领域上有很大的应用前景^[21]。

视频智能处理系统是集物联网、计算机视觉、大数据、人工智能等多学科技的智慧结晶，是一个新兴的应用领域和研究方向。大华、英伟达、赛灵思等多家大公司都已出产相应的视频智能处理芯片以及对应的视频智能处理系统解决方案。目前的视频智能处理解决方案主要分为两大类：

（1）基于云计算的视频智能处理系统

目前基于云计算的视频智能处理系统基本结构如下图所示：

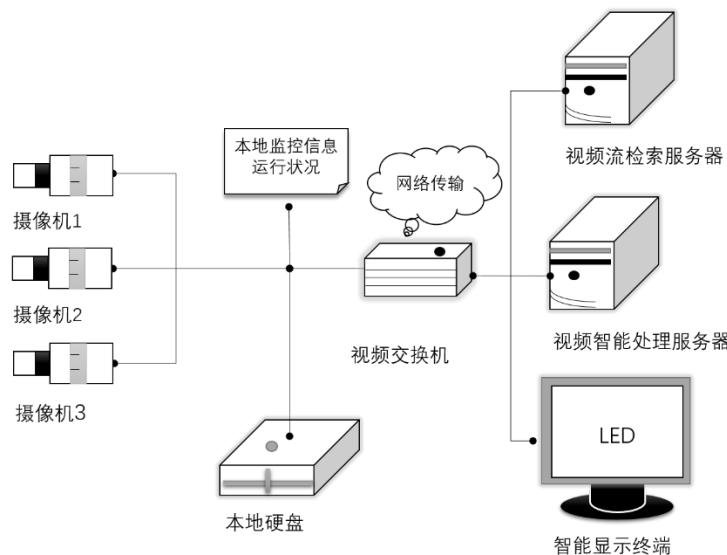


图 1-2 基于云计算的视频智能处理系统

Figure1-2 Intelligent video processing system based on cloud computing

这类处理系统一般包含两个部分：本地视频采集终端和云端视频智能处理服务器。其中本地终端根据视频流协议采集视频数据，存储至本地，转发至云端等；云端视频智能处理服务器将接收转发来的视频流，对其进行解码、预处理、智能分析等。主要存在两类问题：①视频数据实时网络转发，数据量大容易造成巨大的传输压力和数据丢包等。②增加云端的计算压力。一般会选择构建分布式的调度管理系统来对视频进行分发处理，平衡系统传输和处理的压力。

（2）基于边缘计算的视频智能处理系统

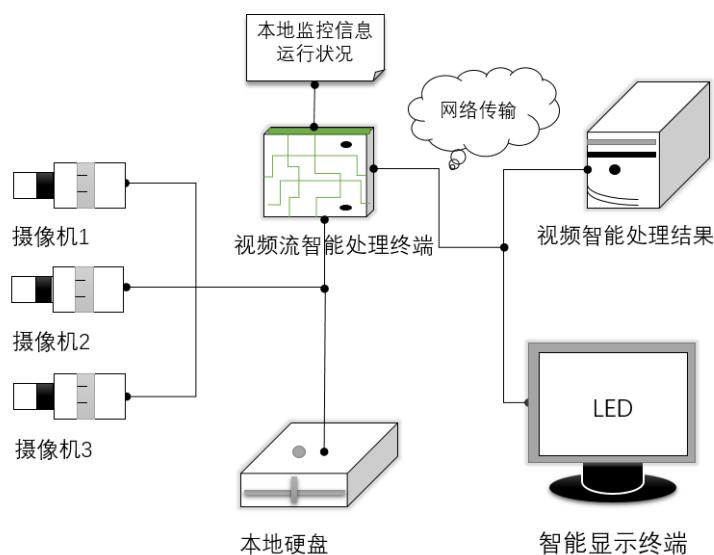


图 1-3 基于边缘计算的视频智能处理系统

Figure1-3 Intelligent video processing system based on edge calculation

上图为基于边缘计算的视频智能处理系统基本结构，对比基于云计算的视频

智能处理系统，其往往有很低的成本和很高的处理效率。将视频中智能处理计算过程部分或者全部放至视频采集终端或者更近的计算节点，有效的降低视频处理延时和开销，主要应用在视频数据传输压力大且隐私保护的场景下。这类系统的研究成果，以海康、赛灵思、华为等为首，以物联网为大背景提出的视频智能处理系统。

1.2.3 调度管理算法研究现状

随着计算机软硬件技术的不断进步，传统的视频处理系统，存在规模小、功能单一、系统控制不灵活、速度慢等特点。研究人员提出了基于硬件架构、软件定义等方向的视频调度管理算法，旨在满足严苛的实时约束和高质量的视频处理。以下是几类主流调度管理算法的研究现状：

(1) 分布式视频调度管理算法

分布式视频处理面向海量用户，提供常用的视频处理服务。下图为分布式视频处理系统的常用算法架构。企业常将其整体设计为视频云，承载视频的后端处理，同时也可作为共有云的方式向外界开放。其中，系统判断上层服务的业务类型，向视频处理子系统发起处理任务，子系统根据任务类型调度相应的任务结点进行处理。同时处理子系统将事件的回调通知返回给上层服务。相关的接口规范包括 SDK、Scheduler、Worker Group、Notifier、Configserver、Dashboard webtool 等，实现对用户管理、任务调度、任务追踪、事件回调、资源管理调度等核心功能的需求^[22]。

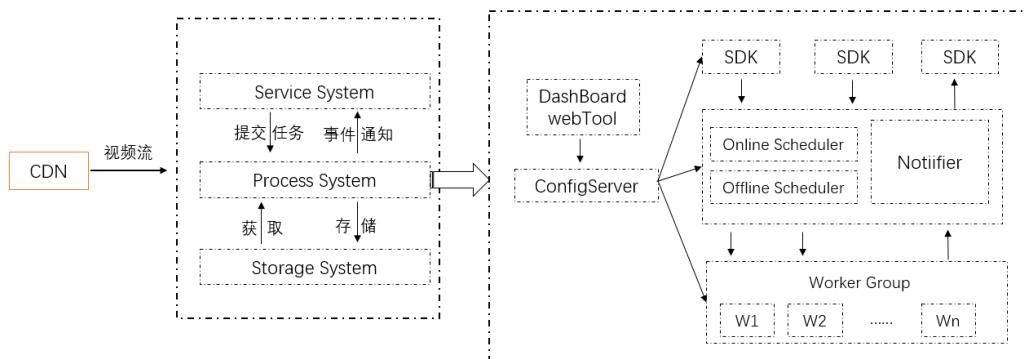


图 1-4 分布式视频处理系统的算法架构

Figure 1-4 Algorithm architecture of distributed video processing system

(2) 边缘视频调度管理算法

视频流处理属于计算密集型任务，为了降低云端视频处理的延时和传输成本，有研究学者提出了基于边缘计算的视频监控框架，将视频处理的部分或者全部任务转移到就近的处理节点完成。边缘设备的计算能力和存储能力有限，需要结合

一定的调度管理算法，提高任务处理的实时性和稳定性，合理利用系统资源，达到信息最大化的目标。边缘视频调度管理算法是由边缘节点和节点管理器组成^[26]。边缘节点管理器连接多个边缘节点，位于操作系统层之上，涉及了整个应用层的资源和任务调度，常涉及到底层硬件技术的优化，例如目前的基于 FPGA 硬件加速的视频编码、转码设计等、基于 FPGA+DSP 结合的视频处理方案等。

(3) 并发流水线调度管理算法

流水线技术是指将处理任务进行分解，其子任务顺序执行在不同的处理节点，不同的处理节点并行执行^[23]。在任意时刻，单个子任务只占用单个处理节点。从本质上讲，流水线技术是一种时间并行技术，因为其并行处理，需要考虑到多并发的问题，又称并发流水线，采用延伸重叠的方式实现，将任务的处理过程进一步细化，提高系统资源利用率。算法适用于高并发、并行，任务执行时间长且重复的应用场景。视频处理的长时性和流动性使得采用并发流水线的算法架构非常有利。

并发流水线有两个很重要的构成元素，任务队列和数据队列。其并发并行的特点，会存在流水线冲突的问题，也即是对于具体的流水线来说，由于“相关”的存在，使得任务队列中的下一个子流程不能在指定的时钟周期执行。会带来一系列的问题，其一就是错误的执行结果；其二是系统会出现停滞、阻塞现象，从而降低流水线的效率和系统的吞吐率等。因此在设计算法时，需对其的性能进行分析，同时设计相关的策略使得算法的性能最优。

1.3 本文主要内容

本文主要研究以下几方面内容：

(1) 多路视频处理系统的调度管理算法的研究

针对多路视频处理的重复性以及时序性特点，提出并发流水线的算法架构，在此基础上提出系统的并行化方案以及总体架构设计。针对不同场景下的视频处理，提出基于场景规则的视频调度管理算法，包括任务槽管理单元设计、数据共享设计、视频存储、智能跳帧等相关算法设计，旨在结合不同应用场景下的视频智能处理结果反馈信息，达到视频处理加速以及视频处理信息最大化的目标，在不同的应用场景下具有扩展性。

(2) 基于轻量级网络的视频智能处理算法研究

本文提出的调度管理算法基于不同场景规则进行定义，因此视频智能后端处理技术研究，对系统整体算法设计和部署极为关键。本文提出优化的基于轻量级网络的视频智能处理算法，在精度和效率上进行均衡，提供调度管理算法的数据

基础，为深度学习算法模型在智能终端部署提供一个轻量级的模型选择方案。本文基于单帧人数统计这一应用场景，进行算法的验证和评估。

(3) 多场景视频智能处理系统的研究

结合多场景视频智能处理系统的需求分析，提出了系统总体架构、流程设计等。针对提出的系统设计目标，搭建实验所需的环境，采集数据，部署相应的调度管理算法以及智能处理算法，验证各类算法模块设计的有效性。根据实验结果，对本文提出的调度管理算法和视频智能处理算法进行分析和总结。

1.4 论文组织结构

本文共包含六章，各个章节的组织结构如下：

第1章，首先对多路视频智能处理技术以及应用系统技术的背景进行介绍，提出课题的研究目标以及研究意义；其次针对目前多场景下视频智能处理、高清视频监控、以及调度管理算法进行国内外研究现状的总结。

第2章，主要描述了视频智能处理系统及调度管理涉及到的关键技术，包括高清视频前端处理、边缘计算、深度学习的相关技术。对各类技术进行细致的介绍和分析，旨在为本文的方案和算法设计提供理论研究基础。

第3章，提出了本文研究课题多场景视频智能处理系统的总体设计，包括系统需求分析、总体架构设计、系统流程设计，下文将基于此系统架构，设计对应的算法模型以及对比实验。

第4章，根据多路视频处理的特点，提出基于并发流水线的算法架构，在此基础上定义系统的调度管理算法——基于场景规则的视频调度管理算法(MSSM)。首先，进行并行流水线性能分析，给出并行化方案设计、整体算法结构；再具体介绍基于场景规则的各类相关调度管理算法设计。旨在达到多路视频处理加速以及视频处理信息最大化的目标，并在第5、6章对相应的算法模块进行实验验证。

第5章，根据第2章相关深度学习加速方法的研究和分析，针对单帧人数统计这一应用场景，提出优化的基于轻量级网络的视频智能处理算法，在精度和效率上进行平衡。提供调度算法的数据基础；也为视频智能处理算法在终端部署提供一个轻量级解决方案，并通过实验进行了相应的算法模型验证。

第6章，首先，针对本文研究课题提出的系统解决方案，给出系统关键模块设计，搭建系统实验环境，提炼关键的算法模块，进行实验验证和结果分析，对系统的总体效果和性能进行评估。

结论，归纳总结本文的研究工作以及初步结论，分析当前研究成果的不足之处，对今后潜在的工作做了相关的探讨与展望。

第2章 高清视频处理及深度学习相关技术

上一章介绍了本文研究课题的国内外研究现状，通过对课题目标的研究和分析，本章将介绍课题涉及的相关技术理论，包括高清视频前端处理、边缘计算以及深度学习等。首先介绍了高清视频前端处理相关技术，包括视频采集、存储、编解码、图像预处理四个方面的介绍；然后介绍了深度学习的工作原理以及相关的加速技术；最后阐述了边缘计算相关技术的应用场景以及相关算法的优点分析。

2.1 高清视频前端处理相关技术

目前基于各类视频处理框架的前端处理过程，可划分为多个子流程：视频采集、编解码、存储、图像获取、图像预处理等，数据来源为高清数字摄像头。高清视频流数据，具有高帧率、高分辨率、连续性的特点，为了提升用户体验，涌现出各类采集方法、编解码方法、存储架构等。为了实现后期的视频内容智能处理，往往还涉及到视频流到目标图像的预处理过程。

2.1.1 高清视频流采集

视频采集的方法按照其信号的不同^[24]，划分为两类：①数字信号采集：主要采用视频采集芯片将模拟信号转换为数字信号，完成图像数据的采集、刷新等，这类方法相对来说功能全面、性能和可靠性都有显著的提高，其缺点主要是成本较高，如 VGA 图像采集卡；②模拟信号采集：图像采集选择通用视频 A/D 转换器实现，其优点是成本低并且易于实现，但是其对 CPU、处理器的要求较高。

2.1.2 高清视频编解码

高帧率、高分辨率的原始视频数据未经过压缩，数据量极大^[25]。视频编码的关键是基于视频图像数据的连续性和强关联性，去除帧间相似信息，用尽量少的比特数据表征视频信息。目前，有效的视频编码标准主要有 MEPG-2、MEPG-4、WMV、H261、H.264/AVC、H265/HEVC、AVS 等。视频编码技术一方面以混合编码为框架提升压缩特性；另一方面，则不断的向可伸缩编码、多视点编码等分支方向发展^[26]。

其中，H264/H265 是一种视频压缩标准，其中 H265 相对于 H264 提高了大约两倍的压缩比，显著提高视频质量。目前图像压缩技术包括帧内压缩、帧间压

缩以及熵编码技术。以下是 H264 的编码器框架：

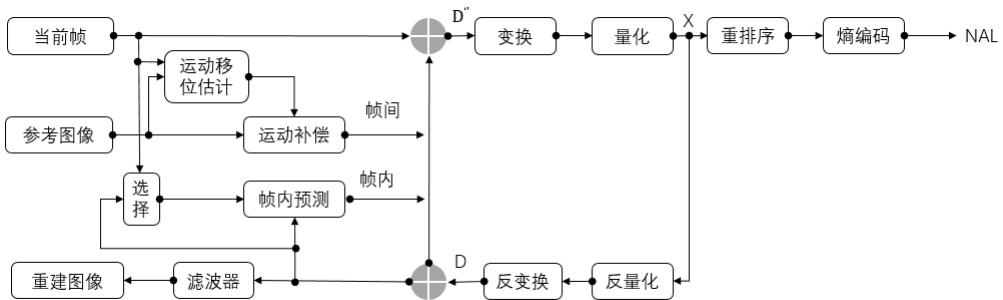


图 2-1 H264 视频编码器的原理框架

Figure 2-1 Principle framework of H264 video encoder

编码器采用变换和预测的混合编码技术，将输入的数据按照帧内或者帧间预测编码的方法进行处理。经过一系列的运动补偿、熵编码、滤波等过程得到压缩数据。对应的有 H264 解码器框架如下，也即是编码的逆过程。目前，针对运动补偿、熵编码部分都已研究出优化算法，加速编解码过程^{[27][28][29]}。

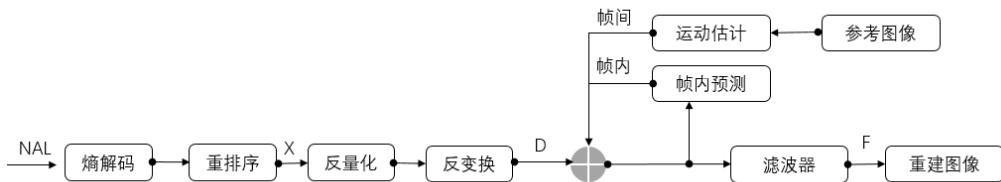


图 2-2 H264 视频解码器的原理框架

Figure 2-2 Principle framework of H264 video decoder

2.1.3 高清视频存储

高清视频监控除了需要对视频进行智能处理之外，还需要定期将视频存储至本地或者上传至视频云，实现后期回放、取证等功能。视频监控其性质是一个长时任务，视频数据即使经过编码，也具有数据量大的特点，对硬件存储可靠性和稳定性都提出了很高要求。随着高清技术的普及，720P、1080P 的视频随处可见，智能和高清的双向需求，促进了存储介质和存储架构设计的发展。目前市场上的主流存储架构如下：

(1) 基于嵌入式架构的存储系统

监控终端内置数据硬盘用以存放本地数据，容量基本在 1-2T 左右，成本低^{[30][31]}。系统集成性较高，如视频预采集、视频预处理、视频检索等。其面向中小型企业监控，视频终端的数量控制在几十路之内。

(2) 基于 X86 架构的存储系统

基于存储局域网（SAN）设计的存储架构，一般可分为IPSAN和FCSAN。相对NVR，其支持更多的节点存储，支持便捷扩展、且技术成熟。主要面向中大型企业高清视频监控，其催生出基于平台的NVR系统，支持客户端查询，扩展性、透明性好^[32]。

（3）基于云技术的存储方案

云存储是一个综合了存储设备、服务器、虚拟现实、物理地址映射、客户端等多个部分构成的复杂系统。是在云计算发展下衍生出来的应用技术，将其数据计算、管理的概念迁移至数据存储结构。该系统围绕存储设备，通过用户软件层向外提供数据存储的接口和相关业务服务。主要应用在大型企业的海量视频存储，其有较好的扩展性和稳定性。

2.1.4 图像预处理

目前，基于图像的视频智能处理是主流方法，其中数据质量的好坏将直接影响模型分析的效果，在分析处理之前需要经过预处理，如灰度变换、几何变换、图像增强等，去除或者减少图像噪声信息，增强图像中有效数据信息；高清视频数据分辨率一般是720P以上，基于图像处理过程对实时性的要求，要求图像的输入分辨率在一定范围，需要对解码后的原始数据进行裁剪、缩放等，才能进下一步分析。以下是几类常见的图像预处理算法：

表2-1 常见的图像预处理算法

Table 2-1 Common image pre-processing algorithm

名称	常见方法	效果
灰度化	分量法、最大值法、平均值法、加权均值	减少处理的数据量、提高处理速度
几何变换	平移、转置、镜像、旋转、缩放 (最近邻插值、双线性插值和双三次插值)	修正数据位置偏差，增加数据多样性
图像增强	空间域法、频率域法（灰度变换、直方图 修正、图像平滑、锐化） (灰度变换、图像平滑、锐化等)	改善图像清晰度，丰富信息质量

另一方面，由于视频数据具有连续性强、数据量大的特点，标准的视频帧率是25FPS，也即是每秒25帧。实际上，因为本地硬件资源有限，视频数据的高冗余性，往往需要对视频数据进行预处理筛选，保证其实时性和处理的有效性。本文提出了一种基于场景规则的增量视频跳帧采样算法，对待处理的视频数据进行预筛选，提高了视频处理的速度和有效性，合理的利用系统资源。

2.2 边缘计算相关技术

目前，全世界范围部署了成千上万的高清视频监控终端，数据以每秒量级的速度增长，海量的视频处理对计算能力和算法架构部署提出了巨大的挑战。以云计算为核心的集中式大数据处理时代，其关键技术已经不能高效处理设备所产生的数据。边缘计算应运而生，旨在解决目前云端数据处理，网络传输压力大、信息泄露、处理延时等问题^[12]。将数据处理的计算过程，边缘化的分布到数据源头的网络边缘侧，就近的提供计算、存储、信息交互等服务，平衡云端的计算压力。以下为边缘计算与云计算的关系：

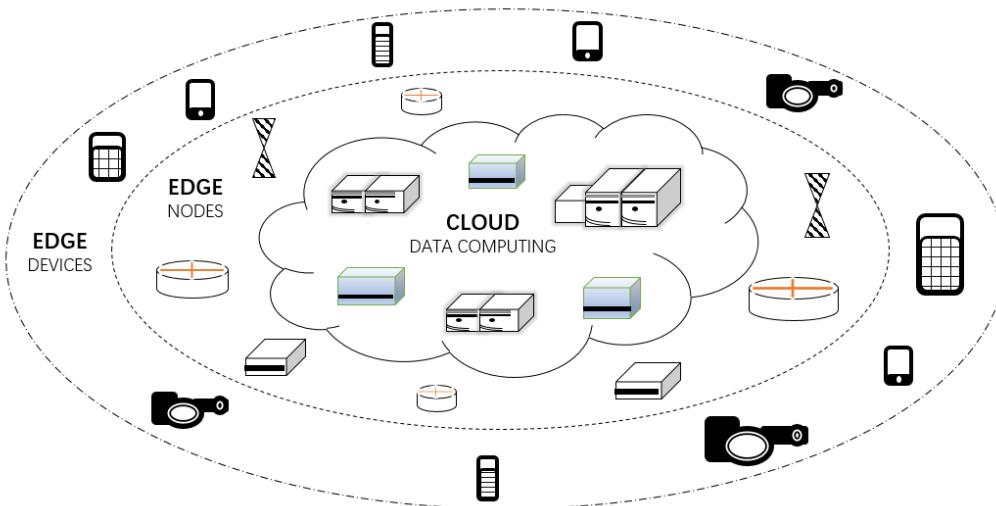


图 2-3 边缘计算与云计算的关系

Figure 2-3 Relationship between the edge of the cloud computing and

边缘计算中的“边缘”是一个相对的概念，是指从数据源到计算中心中间的任意计算资源或者网络资源。以下为边缘计算的技术特点：

- (1) 分布式和低延迟计算。基于云技术的视频智能处理，其数据源通过边缘设备传输至云端进行处理，存在传输和处理延时，影响用户实时决策。比如自动驾驶领域，要求响应时间在毫秒级。边缘计算则可以利用边缘节点的部署，减少数据传输和处理的反馈延时。
- (2) 计算资源分配。与数据中心的服务器相比，边缘终端的计算能力和存储能力相对受限，往往无法执行大量完整的数据分析，且处理过程也增加了终端的损耗。但是，并非所有的数据都需要执行计算，将数据的预处理过程放在边缘节点，执行数据的简单过滤，将有效的利用计算资源和网络资源，应对网络传输中的数据爆炸等问题。
- (3) 智能优化。边缘技术的思想不仅在广泛应用在物联网应用中，同时启发了工业生产的运作模式。比如，工业领域位于底层、嵌入式设备中的计算资源，往

往都能作为边缘计算的资源利用。合理构建生产或者数据处理的整个流水线架构，将计算分层执行，可实现业务流程的智能优化。比如著名的霍尼韦尔六层架构。

2.3 深度学习加速相关技术

2.3.1 深度学习介绍

深度学习作为机器学习的一个分支，其工作原理在于利用神经网络模拟人脑进行分析学习。相对于浅层学习来说，神经网络的多层非线性结构，使其具备强大的特征表达能力以及对复杂任务的建模能力，在图像分类、机器翻译、语音识别等应用领域，都取得出色的成果。其中卷积神经网络作为深度学习的基本结构，包含输入层、隐藏层、输出层多个重叠基本构成单元，旨在通过很少的人工输入，自动的学习到数据低层到高层不同维度下的抽象特征，在一定程度上替代了传统机器学习复杂的特征工程。以下是卷积神经网络的基本结构，包括卷积层、池化层、正则化层、全连接层、分类或者回归层^[33-36]。

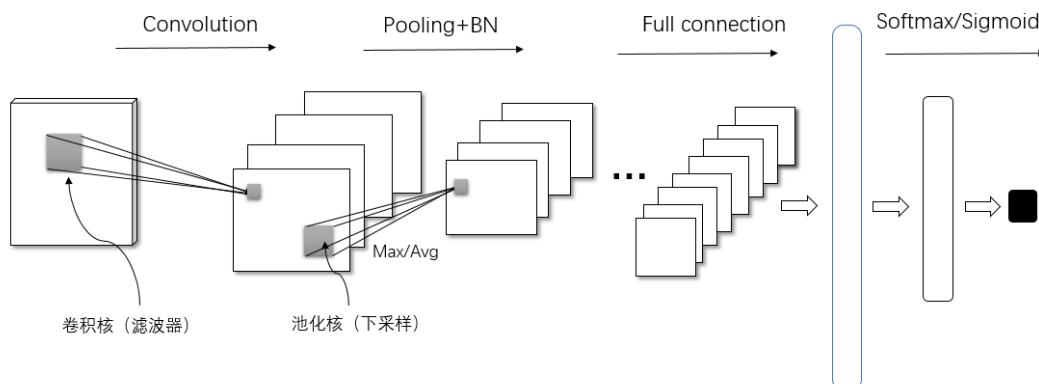


图 2-4 卷积神经网络的基本结构

Figure 2-4 Basic structure of convolutional neural networks

2.3.2 加速相关技术

摩尔定律证明了计算机大规模计算能力和存储能力的发展趋势，促进深度学习成为最活跃的学术研究领域。然而目前构建的深度卷积神经网络模型往往涉及到巨大的训练数据集，复杂的模型结构以及超长的训练时间，要求极高的计算资源和内存资源支撑。面向一些实时应用，如在线学习、增量学习、终端部署和低延时运行的需求，减少大规模网络结构的存储和计算成本变得至关重要。针对深度学习加速的问题，大量学者提出了各种研究方法和成果，包括训练加速、深层模型的并行化框架，动态模型等^{[37][38]}。其中异构计算方法则是近年来研究一大突

破，将深度学习算法模型在数据中心或者边缘计算领域的部署。借助协处理器硬件引擎，包括 GPU、FPGA 或者 ASIC 等，支持定制，其性能有很大的提高。目前市面上流行的 AI 处理芯片或者加速器包括 Nvidia 的 CUDA 加速库或者 Xilinx 的 XDNN 等。

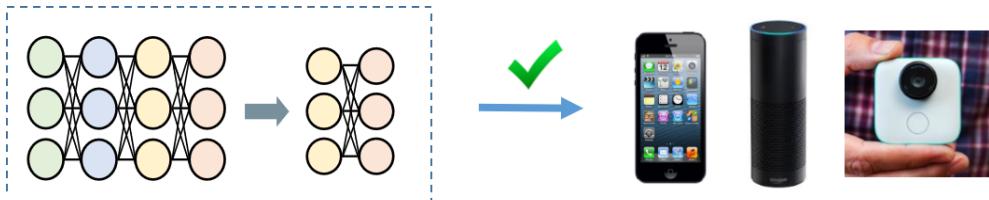


图 2-5 深度学习加速应用场景

Figure 2-5 Model compression application scenario

(1) 模型压缩

结合了各个学科的前沿技术和设计思想，包括体系结构设计、硬件加速设计等。目标是通过减少模型参数总数和模型浮点运算总数达到减少模型的资源需求，保持（无损压缩）或者尽可能少的影响（有损压缩）模型的精确度。降低了网络模型部署在各类嵌入式设备的难度和成本，大大加速了人工智能的发展进程。从参数和结构的角度上，目前模型压缩方法包括：参数修剪和共享；低秩因子分解；转移/紧凑卷积滤波器；知识蒸馏^[39]。

(2) 轻量级网络

针对深度学习加速，最近几年提出了一种轻量级卷积网络结构，这些网络主要的设计目标是一一在保证一定的精度情况下，尽可能减少网络规模（模型参数、模型浮点运算数），让移动设备具有 AI 计算的能力。最近在算法架构探索方面取得了很多进展，包括超参数优化、各种网络修剪方法、连通性学习、引入遗传算法，强化学习思想等，致力于改变内部卷积块的连接结构、计算方法等。其中近几年经典的有 SqueezeNet、MobileNet、ShuffleNet、IGCV、DenseNet 等^{[40][41]}。本文在视频的后期处理选用了轻量级网络，提高视频处理的实时性。

2.4 本章小结

本章围绕高清视频处理以及深度学习相关技术理论展开了研究和分析，主要包含三个部分。首先对高清视频前端处理过程涉及的关键技术进行了介绍；然后描述了边缘计算的应用场景以及技术特点分析，旨在解决传统云计算处理架构存在的问题；最后，总结了深度学习的相关工作原理以及对应的模型加速技术，为本文的轻量级视频智能处理算法提供理论基础。

第3章 多场景高清视频处理系统的总体设计

本文的研究课题是多场景视频智能处理系统及调度管理算法的研究，其中系统划分为两个部分：视频前端预处理以及视频智能处理。为了满足不同应用场景下整体架构的延展性，将视频智能处理部分进行抽象化。本章内容主要介绍了多场景高清视频处理系统的总体设计。首先，介绍了系统的总体分析，包括系统的研究对象以及具体需求分析；然后，介绍了整个系统的架构设计；最后针对系统的总体流程进行设计。

3.1 总体分析

3.1.1 研究对象

本课题提出的系统研究对象是多场景下的高清视频监控数据，常用的高清视频数据分辨率有 1080P、960P、720P 等，应用场景如公交车、风景区、商场入口等。其中，往往会对不同的监控方向部署多路摄像头，也即是多场景下的多路视频处理问题。这类研究对象有以下几类特点：

- (1) 数据量大，随着高清技术的普及，视频监控作为一个长时任务，即使经过编码，其数据量依然量级增长。给视频存储、视频传输、视频实时处理带来了巨大的挑战。
- (2) 格式种类多，视频处理的数据来源，可分为实时采集和本地视频数据。根据视频编解码方式的不同，对应不同的视频封装格式，部分格式支持特定的功能，涉及到不同的视频前端处理库以及算法的选择，甚至处理流程的变换。
- (3) 目标流动性强，不同的视频处理任务，如人数统计、行为分析等，其目标的出现时间和方位不固定，极少数应用场景的目标范围具有规律性。可以根据不同应用场景的目标范围移动规律，制定合适的数据预筛选策略，降低处理冗余性。
- (4) 目标环境复杂，视频监控的实际应用场景下，其研究目标环境随着时间光线的变换、目标遮挡、姿态的可变性等，存在多样性和复杂性，同时监控设的不同其视频包含的信息也有所区别。
- (5) 重要性，目前视频监控的应用场景大多会根据监控方向和范围，部署多个摄像头终端，其不同视频通道的视频数据存在一定的重要性层次。视频处理由于本地资源有限，应参考摄像头的重要性参数，合理分配对应的处理资源。

3.1.2 需求分析

下面将针对功能、非功能以及研究需求，三个方面展开需求分析，其中本文的视频智能处理系统基于边缘计算提出，下图是本文的系统功能结构，其中研究对象为多场景下的多路视频：

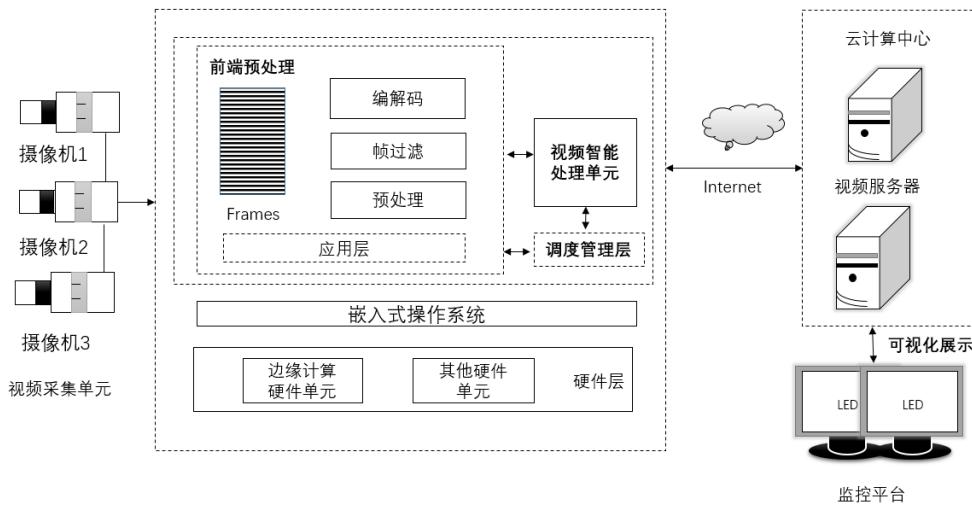


图 3-1 基于边缘计算的视频智能处理系统功能架构

Figure 3-1 Architecture of video intelligent processing system based on edge computing

(1) 功能性需求

① 多路视频前端处理

基于本文提出的边缘计算系统架构，将视频处理的前端过程作为本文调度管理算法的设计过程。其系统前端处理功能应包括：多路视频采集，视频存储，视频转码、视频缩放、图像预处理等，提供视频智能处理的数据前端。

② 视频单帧人数检测

当视频前端处理过程准备好图像数据后，执行单帧人数统计前向运算，实现多路视频的人数检测功能，应满足一定的实时性和精度要求。

③ 数据可视化展示

系统提供实时视频流以及对应视频处理结果的反馈，包含视频显示、切换，视频单帧人数、处理帧率、处理性能展示，系统参数配置等功能。

(2) 非功能性需求

安全性：系统应用于科研课题，且涉及到用户肖像等个人隐私，其应用于工业或者用户的商业经营，数据信息安全性应首要考虑，因此需要设计一定的访问控制和安全策略，防止信息泄露。

可靠性：系统具有长时处理的特点，处于 24 小时供电状态。因此系统应保证持续工作的稳定性，包括硬件设备功耗控制、系统运行相关日志等。此外，系

统为多路视频处理，其算法实现应满足一定的稳定性。

标准性：系统使用的编码格式、传输协议、搭建方式等遵循国际统一标准。

先进性：本文的研究内容为相关领域的前沿性课题，系统使用的技术以及相关理论具有前瞻性。

兼容性：系统面向不同的数据来源、数据格式、处理数量等，具有一定的扩展性，提供对应的功能扩展接口，可应用于不同的场景。

(3) 研究需求

①视频调度管理算法的研究

多路视频处理具有重复性和时序性，面对有限的系统资源以及严苛的实时需求，本文研究了一种基于边缘计算的流水线算法架构，结合基于场景规则的调度管理算法，合理的划分视频处理流程，智能化调度系统的任务和数据，达到加速视频处理，降低视频处理冗余性的目标。其中，面向不同的应用场景，其调度管理算法的设计应具有可扩展性、有效性和稳定性。

②视频智能处理算法的研究

视频智能处理算法在本文系统研究中扮演两个关键角色：作为视频处理调度管理算法的数据基础，调度管理参考不同场景下视频处理结果，其视频处理算法应具有实时性；面向实际应用场景，需要在边缘终端进行智能处理算法模型的部署，目前基于深度学习的处理方法，存在计算量大、模型复杂等特点，而终端的资源有限，功能集成度高，因此设计一类轻量级的视频智能处理算法至关重要，为算法部署提供一个轻量级解决方案。

③多场景视频处理系统的研究

作为本文的研究课题领域之一的多场景视频处理系统，对系统的总体设计、功能设计、系统流程设计等给出一个具体的解决方案。针对系统涉及到的算法模型，进行部署，搭建实验系统，对算法进行实验验证和结果分析。

④算法性能优化

现有的调度管理算法以及视频智能处理算法面向不同的应用场景有各自的优势，在大量实验的基础上，误差尚有缩减的余地、性能尚有一定的提升空间，需要根据具体实验，结合软件、硬件优化技术进行改进优化，所以提高算法的精度与速度是一大主要问题。

3.2 系统架构设计

多场景视频智能处理系统主要是包含视频前端处理以及视频智能处理后端两个部分。本文为了改善传统视频处理延时大、传输不稳定、云端计算压力大等

问题，研究出一套基于场景规则的调度管理算法，结合边缘计算的思想，部署在视频前端处理部分，既能保证面向多场景下的扩展性，也达到加速视频处理、减少视频处理冗余性的目标。视频智能处理后端则根据不同场景灵活设计和部署，可选择部署在本地边缘处理芯片或者云端视频处理服务器中。

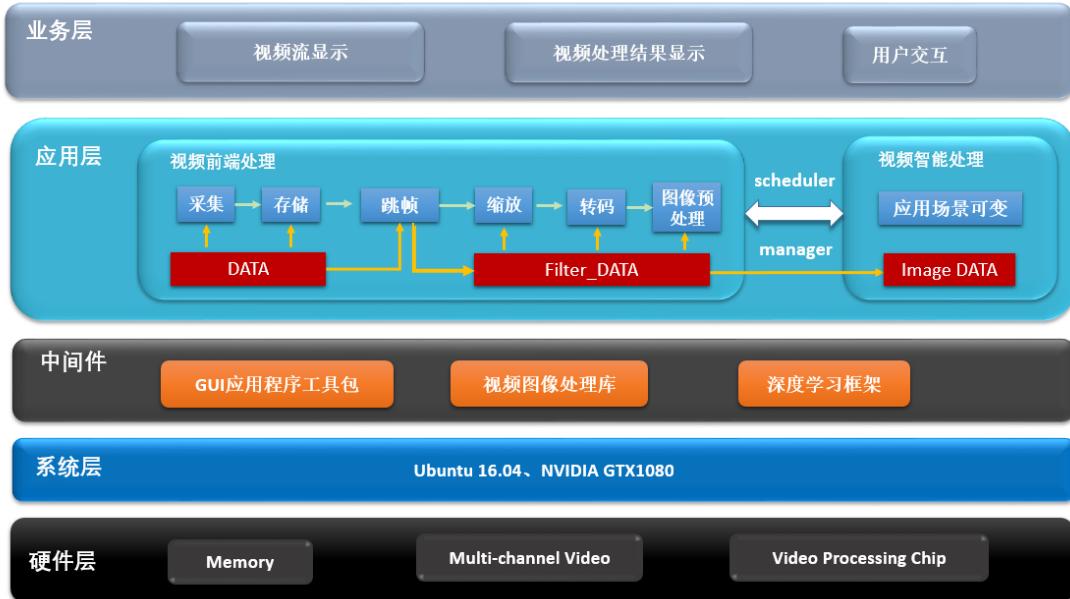


图 3-2 多场景高清视频处理系统总体架构

Figure3-2 Multi-scene HD video processing system architecture

以上为多场景视频智能处理系统的总体架构，其中视频智能处理模块抽象化，便于系统扩展。主要包括硬件层，系统层，中间层，应用层，业务层等几大关键模块，下面将分别进行阐述：

- (1) 硬件层：针对多场景下的多路视频处理，其硬件层包括存储器、传感器、智能接口、多路摄像头、网络接口、智能处理芯片等；
- (2) 系统层：操作系层采用 Linux16.04，内核版本是 4.15.0-generic，其中 GPU 处理器是 Nvidia GTX1080 处理器，支持 cuda9.0 以上的编译版本。
- (3) 中间件：采用开源 Python 套件 Anaconda4.0 进行视频处理后端的开发。其包含大量的数据科学处理包以及应用依赖项；其中 GUI 套件选择 PyQt，其跨平台的特性满足了用户的开发需求；视频处理库选择 FFmpeg、libyuv、OpenCV 等开源框架，具有跨平台特性；视频智能处理的深度学习框架，本文选择了基于 Tensorflow、Theano 以及 CNTK 后端的 Keras 框架作为系统的视频处理模型框架，具有操作简单、上手容易、文档资料丰富、环境配置容易等优点，简化了神经网络构建代码编写的难度。
- (4) 应用层：定义整个视频处理系统的功能模块。其中，视频前端处理包括视频流采集、存储、解码、缩放、图像预处理等操作；视频智能处理则包括加载图

像数据、模型文件、前向运算等。系统的调度管理算法部署在应用层，其操作对象包括两类：①任务调度管理②数据调度管理，建立在实际的应用层需求之上。根据视频智能处理结果的反馈信息，调度视频前端的数据和任务，具体算法设计包括任务调度单元设计、数据共享设计等。

(5) 业务层：根据用户需求的上层功能定义层，包括用户交对终端的控制交互、视频智能处理结果的展示方式等。其中不同场景下的结果展示方式和平台可以不同，根据场景应用需求，还可定义一些统计操作或者反馈信息。

3.3 系统总体设计

上一节针对多路视频智能处理系统的总体架构给出了介绍，并具体阐述了各个层级结构的具体内容和所处角色。本节将根据系统的总体架构，进一步提出处理系统的总体流程设计。结合本文提出的两个研究点，在总体流程上进行细化。其中调度管理层的内容体现在应用层的程序模块之中。该系统的具体程序流程如下图所示：

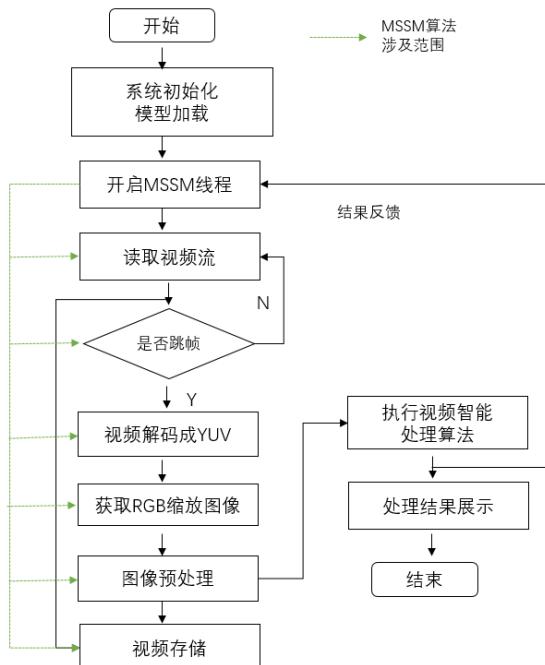


图 3-3 视频智能处理系统程序流程设计

Figure 3-3 Video intelligent processing system program flow design

(1) 系统初始化包括设备状态检查、文件系统检查、数据队列初始化、主进程初始化、图像界面初始化等过程；模型加载，是加载之前训练保存的视频智能处理模型，包括模型结构、参数等。

(2) 开启调度管理线程，包括视频前端处理线程和视频后端处理线程。视频前

端处理线程、包括视频采集、解码、存储、转码压缩等处理过程。结合提出的调度管理算法策略，将对系统的数据、资源、任务等进行全局分配，达到加速视频处理，减少视频处理冗余的目标，同时减少视频智能处理后端的计算压力。其主要作用在视频前端处理的各个应用程序之中。

(3) 一旦图像共享数据队列有高于阈值的缓存数据，视频智能处理后端开始工作，其图像的预处理过程均在视频前端完成，视频智能处理可以部署在服务器或就近的结算节点，待处理视频数据的减少以及处理过程的分布设计，降低了视频智能处理节点的计算压力。

(4) 视频智能处理，选用基于图像的智能分析，在图像数据准备好后，将数据送入加载好的处理模型之中，进行前向运算，得到处理结果。处理结果其一是直接送入显示终端，进行显示或者其他统计分析；其二是将结果返回给系统进程，用以系统根据结果进行相应的调度和管理。

3.4 本章小结

本章围绕多场景下高清视频智能处理系统的总体设计展开了讨论，主要包含三个部分。首先，对系统的研究对象以及需求分析进行了介绍，其中包括系统功能性需求、非功能性需求、系统的研究需求；然后，给出系统的总体架构设计，并对系统架构涉及到的关键层进行角色分析和介绍；最后，提出了系统的流程设计。后文将对本章提出的需求进行相应的探讨设计和算法验证。

第4章 基于并发流水线的MSSM视频调度管理算法研究

根据多路视频处理的时序性和重复性，提出基于并发流水线的架构设计，该方法通过合理的安排视频处理步骤，结合软硬件加速技术，加速视频处理。此外，将视频处理过程边缘化，将减小云端处理压力。视频处理的调度管理对象包括处理任务和数据，本文提出基于场景规则的调度管理算法（MSSM：Multi-Scenario Scheduling Management），面向不同应用场景具有扩展性。本章内容主要介绍了基于并发流水线的MSSM视频调度管理算法设计，在第三章的系统总体设计之上，首先介绍了并发流水线的性能评估指标等；然后介绍了MSSM的总体架构设计；最后针对MSSM的各个算法模块进行了详细的设计。

4.1 并发流水线性能分析

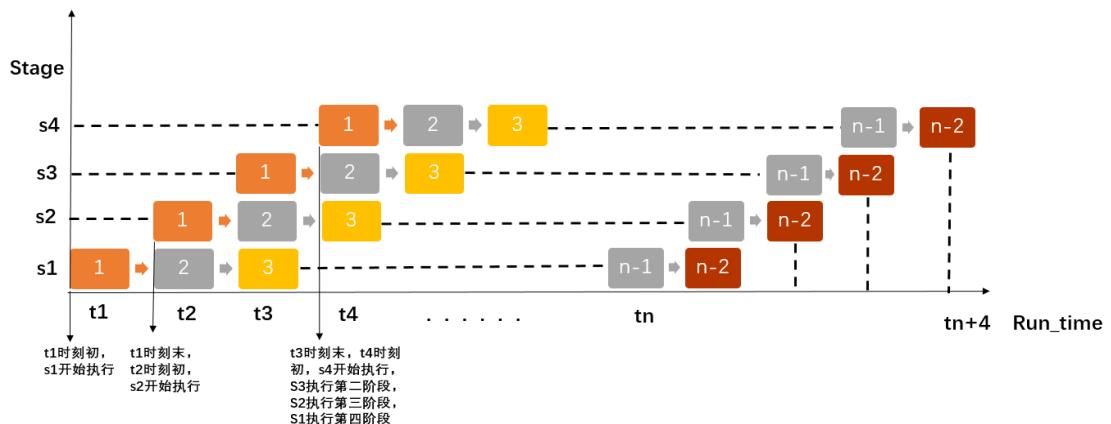


图 4-1 流水线时间-空间描述图

Figure 4-1 Pipeline time-space description

以上是流水线的时空图表示方法，其中横坐标代表处理子任务在流水线中的时间位置；纵坐标代表处理子任务在流水线的空间位置。通常基于并发流水线的软件系统均有各自的适用条件和优缺点，而评价一个流水线系统，可以从以下三个常用的评价指标考虑：

(1) 吞吐率 (Throughput Rate)

定义：表示单位时钟周期内流水线处理的task数量

$$TP = \frac{n}{T_k} \quad (4-1)$$

其中：n 为任务数； T_k 为完成 n 个任务所用时间

理想情况下，假设流水线的各级任务执行时长一致，单位时钟周期为 Δ_t ，在

任务连续处理的情况下，完成所有任务所需总时长为：

$$T_k = (k + n - 1)\Delta_t \quad (4-2)$$

其中 k 为流水线的级数， n 为任务的总数量，因此流水线吞吐率为：

$$TP = \frac{n}{(k + n - 1)\Delta_t} \quad (4-3)$$

$$\text{当 } n \text{ 趋向无穷大时, } TP_{max} = \lim_{n \rightarrow \infty} \frac{n}{(k+n-1)\Delta t} = \frac{1}{\Delta t} \quad (4-4)$$

反之，假设各段任务的执行时长不一致，比如四级流水线的任务执行时长分别为 Δ_{t1} 、 Δ_{t2} 、 Δ_{t3} 、 Δ_{t4} 。此时流水线的最大吞吐率如下，受限于最慢的子任务时长，称为瓶颈子任务。

$$TP_{max} = \frac{1}{\max\{\Delta_{t1}, \Delta_{t2}, \Delta_{t3}, \Delta_{t4}\}} \quad (4-5)$$

以上都是相对比较理想的状态，事实上，系统往往会有不稳定的因素，导致流水线不能连续流动，其时长会有跳动性的变化，因此实际的吞吐率 TP 总比最大吞吐率 TP_{max} 小。

(2) 加速比

定义：定义一批处理任务，其顺序执行和流水线并行的时长比^[34]

$$S = \frac{\text{顺序执行时间}(T_0)}{\text{流水线执行时间}(T_k)} \quad (4-6)$$

假设流水线的各级任务执行时长一致，单位时钟周期为 Δ_t ，在任务连续处理的情况下，完成 n 个任务所需总时长为：

$$S = \frac{k * n * \Delta_t}{(k + n - 1) * \Delta_t} = \frac{k * n}{(k + n - 1)} \quad (4-7)$$

当趋向无穷大时或者 $n \gg k$ 时， $S_{max} = k$

理想情况下，线性流水线每个子任务执行时长一致，且任务定义的数目远比其段数大，其加速比趋近于最大值。

(3) 效率 (Efficiency)

定义：在流水线的运行时间角度上定义，具体来讲是指流水线的有效时间占整个过程的时间利用率。往往因为系统资源，以及第二章提到的冲突问题造成整个流水线并不是满负荷运转。

以下是理想的情况下的线性流水线，一共包含 T 个时间段，每个段时长一致，均为 φ_0 ，即

$$\varphi_1 = \varphi_2 = \varphi_3 = \dots = \frac{n * \Delta_{t0}}{T} = \frac{n}{k + n - 1} = \varphi_0 \quad (4-8)$$

那么整个流水线的效率计算公式如下：

$$\varphi = \frac{\varphi_1 + \varphi_2 + \dots + \varphi_k}{k} = \frac{k * \varphi_0}{k} = \frac{k * n * \Delta_{t0}}{k * T} \quad (4-9)$$

通过对时空图的分析，分子是 n 个任务实际占用的总面积，分母是图中 k 段流水线总时间 T 围成的总面积。进一步换算：

$$\varphi = \frac{n * \Delta_{t0}}{(k + n - 1) * \Delta_{t0}} = \frac{n}{k + n - 1} = TP * \Delta_{t0} \quad (4-10)$$

当 $n \gg k$ 的时候， φ 才趋近于 1，且对于线性流水线假设每段经过的时间相等时，流水线的效率是正比于吞吐率。因此在实际划分流水线的段数的时候，需要进一步考虑每段的执行效率。针对本文的课题研究，在设计基于并发流水线的算法架构时，应考虑以下两个方面的设计，优化并发性能。

(1) 任务调度算法

能随时安全的暂停、继续、停止任务线程。在流水线架构中，多个线程并行执行，也就需要一个安全、有效的线程同步机制，也即是对应的任务调度算法，此为流水线运行的核心。包括数据队列的缓存情况，系统的资源占用情况，任务的紧急情况，外部请求等，这些往往都是调度算法设计需要考虑到的因素。

(2) 生产消费者队列控制

这里的队列是指任务队列和数据队列，任务队列划分为处理时间一致，且内存开销相对平衡，其并行的数量根据系统资源、并行时间、复杂度等因素设计；数据队列，当数据缓存队列满了或者没有新的数据到来的时候，能自动的安全的调用对调用线程进行阻塞；恢复正常之后，自动唤醒等待线程，这是数据传递的核心。

4.2 MSSM 总体架构

为了保证应用场景的扩展性和灵活性，本文提出基于场景规则的调度管理算法（MSSM：Multi-Scenario Scheduling Management），部署在并发流水线的架构之上。其中场景规则不同于以往的先验知识，结合实时的视频处理结果，互相权衡和约束，是本文算法设计的思想基础。为了面向不同的应用场景具有扩展性，本文的调度管理算法主要集中在视频智能处理前端，对相关功能的任务和数据进行管理和调度。下图 4-2 为多路视频前端处理的系统架构。

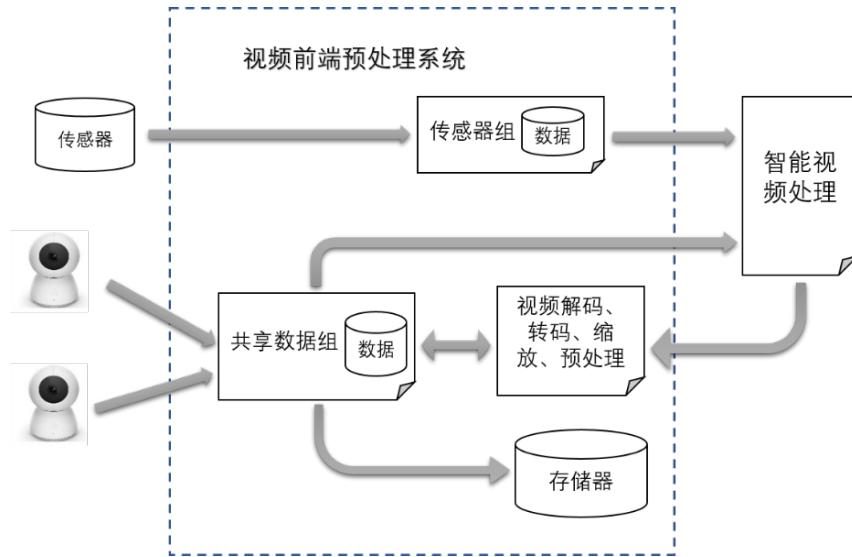


图 4-2 多路视频前端处理的系统架构

Figure4-2 System architecture for multi-channel video front-end processing

这样设计的优点是：针对通用前端处理模块的设计，应用到多场景具有可延展性。以下先给出整体的并行化方案设计、算法总体架构，再具体介绍各个调度管理模块的算法设计。

4.2.1 并行化方案设计

方案 1：多路视频流的数据并行：处理过程对于每个视频流来说都是独立的，数据和任务的管理和调度独立实现，结构如下图所示：

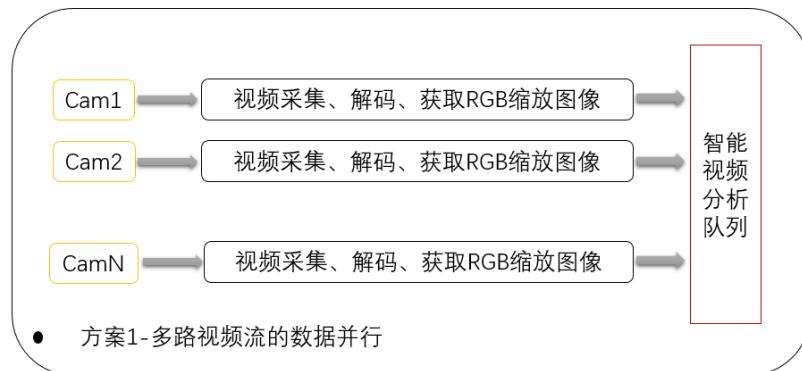


图 4-3 多路视频的数据并行方案

Figure4-3 Multi-channel video data parallel solution

方案 2：多路视频数据任务并发并行：提取出处理子过程，各个视频流不再独立处理，共享任务队列和数据队列：

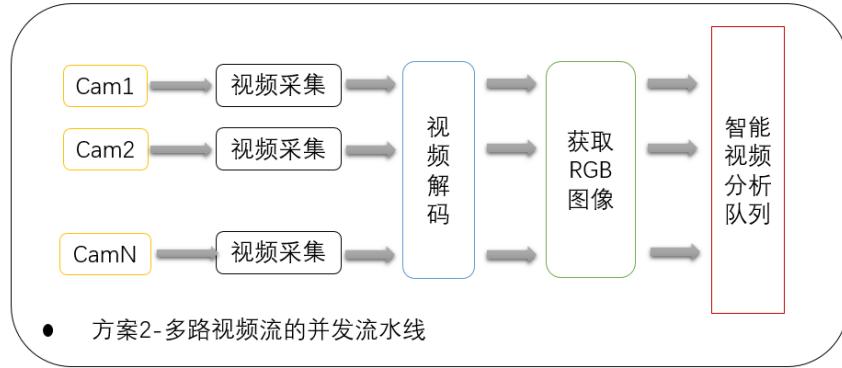


图 4-4 多路视频的并发流水线并行方案

Figure 4-4 Concurrent pipeline parallel scheme for multi-channel video

第二类方案的所有模块处于异步状态，模块之间的数据和任务是共享的且独立于数据源，减少任务和数据队列建立的开销。这类方案明显的一个优点是：对于每一块的调度是非常灵活的。比如一段时间内视频的行人目标可能比较少，就可以通过调度算法，增大视频的跳帧参数，减少视频的解码任务队列，增大视频存储队列等，从而实现资源的最大化利用。其额外的需要针对数据和任务有一个同步管理的算法设计。本文选择第二类并行化方案。

4.2.2 算法总体架构

基于第二类并行化方案，其算法和数据模块均可以复用，降低系统开发难度和成本，充分利用多核的并行处理能力。下图是算法的整体架构：

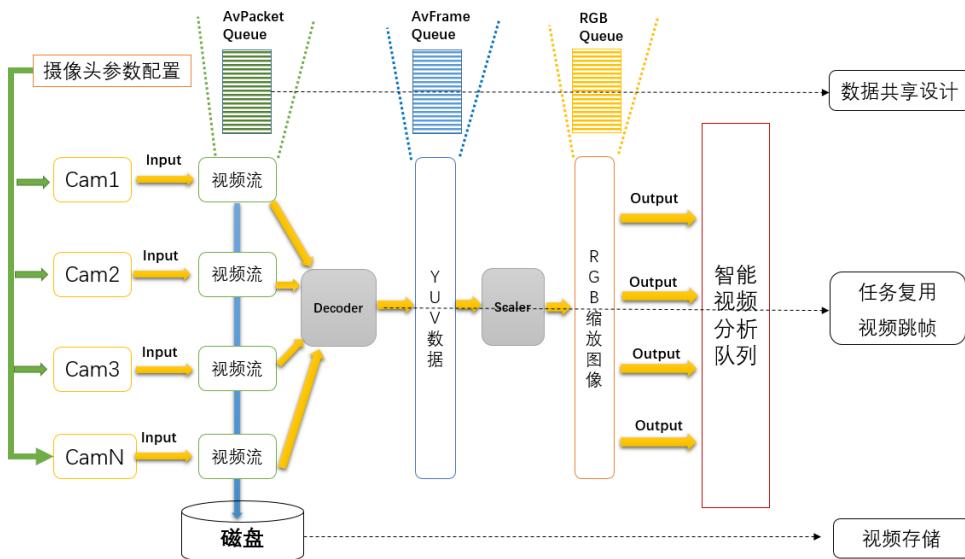


图 4-5 基于并发流水线多路视频前端处理系统架构

Figure 4-5 Concurrent pipelined multi-channel video processing system architecture

结合本文的应用场景，多路视频通用的前端处理主要包括五个功能模块：视频采集模块；视频解码模块；视频存储模块；视频图像获取；图像预处理模块。系统内的所有模块都处于异步状态。数据和任务均独立于数据来源设计，实现了多对一再到一对多的数据同步。

4.3 MSSM 相关算法设计

4.3.1 视频调度任务槽设计

所谓任务槽是指流水线子级上抽象出来的并行任务队列的上一层管理单元，是一类数据结构。视频调度任务槽包括管理单元和子任务队列。通过调度算法，管理和更新任务槽状态，进而创建和管理下层子任务。子任务，是一系列并行队列，独立于数据来源。其中数据并行的设计确保了子任务的并行和同步。这样设计的优点是，系统中的任务进行统一配置和管理，具有很好的扩展性，也能方便的对系统任务执行的状态进行统计，用于后期用户界面显示等。以下是单路视频前端预处理的任务槽设计图：

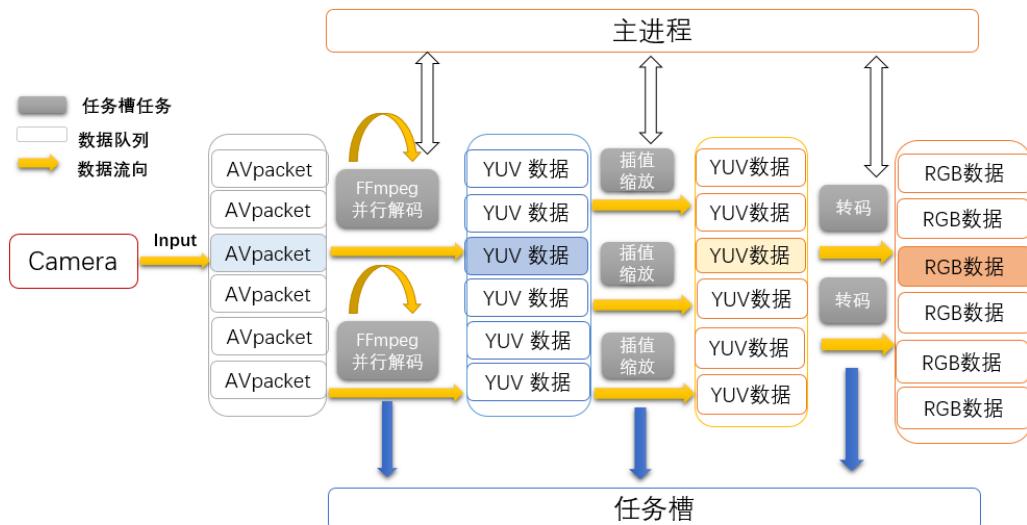


图 4-6 单路视频前端预处理的任务槽设计

Figure 4-6 Task slot design for single-channel video front-end preprocessing

对于多路视频而言，每个视频流的处理过程是一致的，视频流读入以后，根据下文的跳帧策略，选择视频数据加入后续处理，包括解码、缩放、转码、预处理等操作。本文中，每类子任务对应一个任务槽管理单元，其队列数据结构设计可以复用，同时也是作为各路视频处理的共享任务资源，执行过程独立于数据来源。如不同路视频解码得到的 yuv 数据，通过转码任务槽得到 rgb 数据。本文采用 FFmpeg 做视频的相关处理，其中视频流的转码、缩放预处理等通过实验对比，

选用了不同的处理方法，用以划分子任务，提高前端任务流水的效率，具体对比实验见第六章。如下是任务槽和子任务的数据结构设计。任务槽及其子任务作为系统关键的数据结构，其属性和方法见下表：

表 4-1 任务槽数据结构设计

Table 4-1 Task slot data structure design

属性	说明
Task_ID	任务槽 ID，系统分配
Task_Type	任务类型：视频解码、yuv2rgb 等
Task_State	任务状态：挂起、运行
Task_Priority	任务优先级，用户设定
Number_of_Children	子任务数，用户设定
Children_Tasklist	子任务 ID 列表，系统分配
Process_Data_address	待处理数据队列地址，系统分配
Result_Data_address	结果数据队列地址，系统分配
Change_Flag	子任务调度标志，0 正常运行；1 增加子任务；2 减
Change_List	调度序列，根据 Change_Flag 赋予意义
Run()	任务内容

任务槽管理了一系列的子任务队列，子任务的属性和方法见下表：

表 4-2 任务槽子任务结构设计

Table 4-2 Task slot subtask data structure design

属性	说明
Subtask ID	任务 ID，系统分配
Task_State	任务状态：挂起、运行
Run()	任务内容

以上是任务槽以及其子任务的主要属性和方法定义。主程序通过以下的调度管理规则以及子任务的运行状态等更新和管理任务槽的状态，进一步管理子任务。

4.3.2 数据共享设计

本文提出的并发流水线的算法架构，其数据和任务的都是并行和复用的。数据的共享设计是指多路视频的处理数据不再独立设计存储队列，而是位于同一数据队列或者多个数据队列，为了得到输出结果的一致性，其共享数据需要设计一定的机制去同步和标记数据来源，建立数据通道映射，使其外部透明，内部有序。其中数据队列的并行也保证了处理任务的并行和同步。数据分为两类：

(1) 单源数据：是指从摄像头读入的视频码流数据，即每个摄像头对应一个视频数据队列，主要有两点考虑：视频存储，需存储完整码流，共享则增加额外计算；视频跳帧，跳帧算法应用于每个视频流，共享则增加额外计算；

(2) 混合数据：是指数据来源无关的任务共享的数据队列，比如待处理的解码数据，yuv 数据等；

目前两类数据采用同一种数据结构，数据结构定义如下：

表 4-3 数据共享数据结构设计

Table 4-3 Data sharing data structure design

属性	说明
Data_Address	数据地址，系统分配
Total_Frame	数据大小，包含多少帧数据
Current_Frame	当前帧存放位置
Frame_Num	第 Num 帧位置
Frame_State	第 Num 帧数据状态
Channel_Info	第 Num 帧数据通道
Frame_Info	第 Num 帧数据具体内容

其中，数据地址系统初始化数据结构时分配，对应任务槽对应不同的数据地址和任务队列；当前帧存放位置，数据循环存储；数据内容包含了数据的状态（处理/未处理），避免重复处理，channel_info 字段标志了数据的通道来源，frame-info 则是对应的数据内容。

4.3.3 基于任务复用的任务调度算法设计

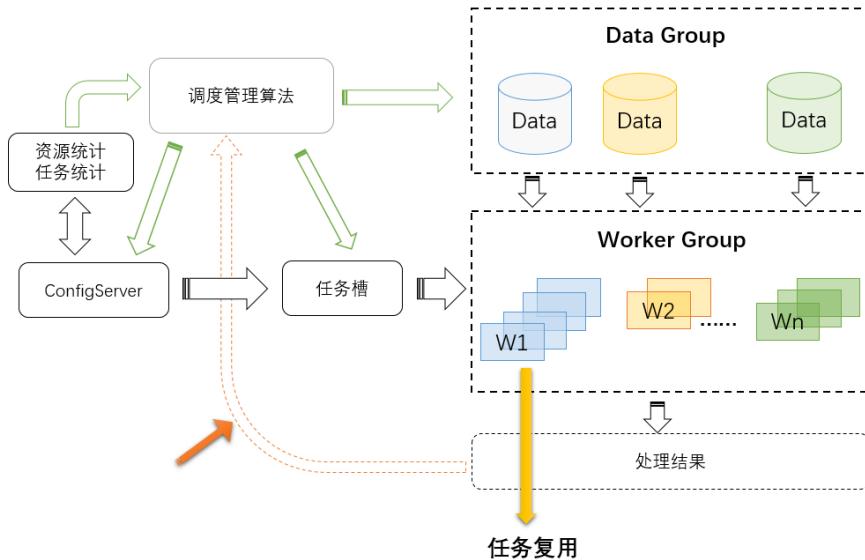


图 4-7 任务复制+分布式资源调度算法架构

Figure 4-7 Task Replication + Distributed Resource scheduling algorithm architecture

本文将多路视频处理流程进行划分，便于部署在不同的边缘计算节点，每个节点的执行独立于数据来源，节点之间数据空间共享。基于此本文提出了并发流

水线+任务复用+数据共享的算法设计思想。任务复制作为二者之间的连接纽带。参考分布式系统处理架构，得到图4-7的基于任务复制的分布式资源调度算法架构，分布式是指任务和数据分布处理。

在多路视频并行处理下，多路数据共享存储空间；处理任务独立数据来源，可处理多路视频数据，也即是任务复用。这样设计的好处是：①保证了处理过程的并行性；②减少任务创建和通信开销 ③统一任务的管理和调度。在设计一个多任务处理算法调度系统时，需要考虑处理系统本身所能支持的最大计算量。任务划分的大小以及任务的数量都需要在实验中考虑其资源占用和运行时间（按最大值）得出。当任务初步设计好后，其调度和管理状态仍然支持动态更新，可参考的规则可以是资源调度和场景规则，资源调度需要考虑 CPU、Memory、磁盘、网络等状态，场景规则是视频智能处理后端结果目标变化、环境变化范围等反馈信息，动态调整数据的处理数量以及处理任务的配置。针对各个功能模块，本文定义了以下算法规则：

（1）ConfigServer思想

类似于集群管理器，对底层的任务槽、worker组进行注册、管理。同时进行任务统计和资源监控等。对整个系统的处理过程进行初始化、配置和注册，初始化包括系统设备检查、文件目录创建、任务槽、共享数据存储结构初始化等；配置和注册包括注册任务槽和配置任务槽的参数。后期，监督系统运行情况，更新任务槽的状态，同步和管理底层数据和任务。

（2）中间态的表征结果

本文将视频前端的处理过程划分为多个子过程进行处理，其产生的数据即为中间态的表征结果，数据的存在形式见下一小节。

（3）Worker组和容错机制

Worker组可以理解为任务槽管理的一系列并行子任务队列，其设计与数据来源无关，Worker自行定义任务的类型。其上层管理单元是任务槽，当一个worker组内的子任务宕机了，完全不影响整体的运行进程，因为任务槽会更新状态，可以再次激活也可分配给其他worker子任务执行（存在多个）。

4.3.4 视频存储管理算法设计

在第二章介绍了各类视频存储架构的应用场景，本文针对基于NVR的存储架构设计其管理算法，包括存储结构以及存储格式两个方面。视频检索是一个长期研究的问题，如何降低视频检索的延时，提升用户体验，其涉及到存储结构和检索算法的设计。这类视频存储算法设计一般需要考虑四个问题：

①负载均衡：视频存储的时候，系统资源占用较大，需要考虑多个视频介质的负载均衡。比如，视频通道的分辨率不同将影响视频存储介质可存储视频文件总量。

②存储结构化：在视频大数据下，便于后期快速检索，视频数据存储需要有一定的结构化设计，同时避免数据移动。

③视频修复：为了保证视频数据的正确解析和播放，在视频存储的时候，需要添加视频的头尾信息。但是在某些紧急情况下，视频不能正确的写尾，因此还需要设计视频修复策略，保证视频存储的正确性。

④视频存储格式：对应不同的视频播放平台等往往需要有一定的格式选择，如果是直播平台，一般还应支持在线转码等。

基于上述四个方面的考虑，本文设计面向多场景的多路视频存储策略，包含：

(1) 视频存储目录结构

为了便于检索，固定视频通道对应的存储位置。系统运行时，定期检查存储空间，一旦存储空间少于阈值，启动删除前期视频数据再进行处理。视频存储有一定的目录结构，设计如下图。某些时候是通过建立一定的地址映射实现。



图 4-8 视频存储文件目录结构设计

Figure 4-8 Video storage file directory structure design

(2) 视频存储状态表

视频存储状态表表明了当前视频通道的存储情况，用以视频修复检查，其一表明视频当前正在执行存储，其二表明该时间的视频错误退出没有写尾，以文件的形式存在每个视频通道目录下。如下所示：

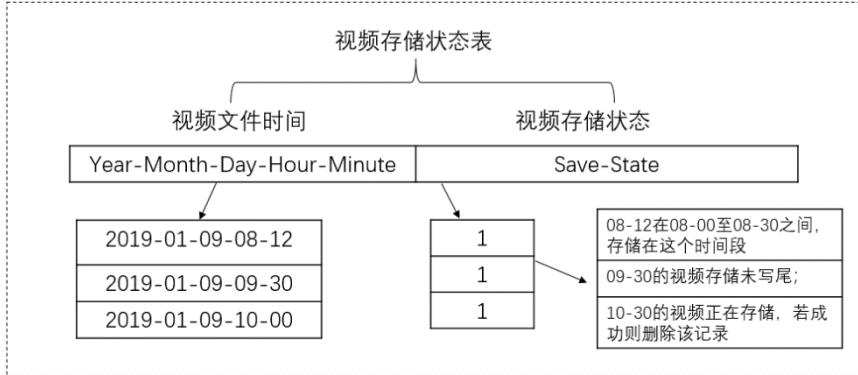


图 4-9 视频存储状态表结构设计

Figure 4-9 Video storage state table structure design

设置视频的存储间隔为半个小时，意思是一个小时内的 0-30 分钟为一个视频，即使只有片段，存储参数也可调整。另外，视频存储只有在出现紧急情况才会导致视频不能正确写尾，那么视频修复的时间，一般有两种方式：①在系统资源比较充足的时候，将视频修复任务调度，实现空闲视频修复，需要调度算法支持；②在系统初始化一段时间内进行视频修复，实现简单，但会占用系统时间。

(3) 视频封装格式

视频编码和视频封装对应的视频流数据的生产过程和包装过程，其包装过程主要是封装了视频信息，包括将视频和音频集成在一起，其封装方式并不影响视频质量。视频文件的封装格式多种多样，常对应了不同的编码格式和文件格式，且不同的视频封装格式往往还支持某些特定功能，因此需要结合视频后期任务的特点选择视频封装格式。

4.3.5 视频跳帧采样算法设计

视频智能监控利用视频智能技术对监控视频进行一系列分析处理。在不同的应用场景下，不同的视频处理需求如人脸检测、行为分析等，其中目标检测是视频智能处理的前导步骤，然而不同场景下检测目标的数量是动态变化的。例如分析任务是人头检测，人流量在不同的时间段有很大的偏差，那么不同情况下的计算需求存在差别。人流量稀疏的地段，比如工作日景区入口处，如果每一帧都检测，无疑是在这类空白帧的计算上耗费很大的资源，且多路视频同时处理，视频通道由于监控的范围不同，其重要性可能不同。这时候，本文通常会结合视频通道加权以及跳帧算法来甄选出待处理的视频帧，以减少视频处理的计算压力。

(1) 基于关键帧的跳帧算法

本文的视频编码格式为 h264，包含三类编码视频帧：I 帧：完整编码；P 帧：

前向预测帧，参考之前 I 帧，生成的包含差异部分的帧；B 帧：前后向预测帧：参考前后的帧编码生成的帧。其中 I 帧是关键帧，是对整个图像的完整编码，解码时候无需参考前后帧，即可重构图像。因此，基于关键帧的跳帧算法也即是仅解码视频流的关键帧数据做智能分析，其中每一秒关键帧的数量可以进行控制。这样设计的特点是：①算法简单，易于实现；②减少视频帧的解码压力（无需参考其他帧）；③提取的关键帧不一定是信息最大化的“关键帧”，因此存在冗余分析或者漏检情况；④关键帧的数量会影响视频码流的压缩比。

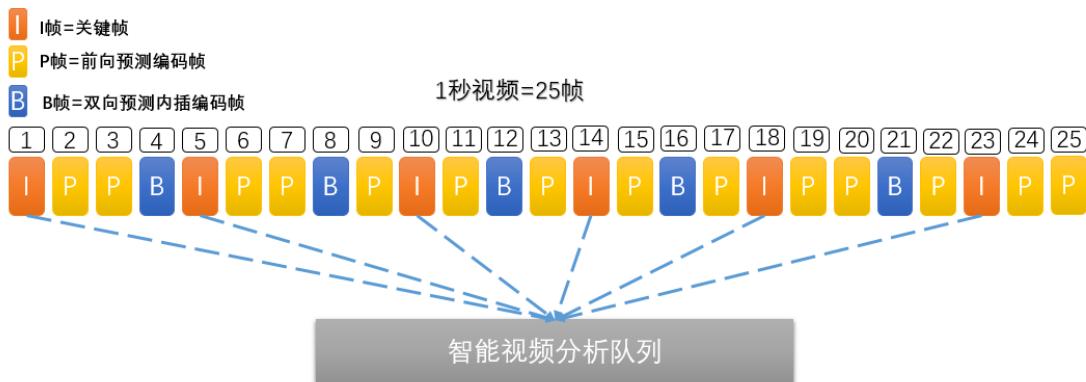


图 4-10 基于视频关键帧的跳帧算法

Figure 4-10 Frame skipping algorithm based on video key frame

(2) 基于固定步长的跳帧算法

此类算法主要是根据用户设置一定的间隔选取视频帧用于解码分析，算法的特点同上述基于关键帧的跳帧算法一致，但是不同的步长会增大视频的解码压力，且依然存在关键信息漏检的概率较大。假设从每一秒的第一帧开始检测，以下是根据步长得出的检测帧数：

$$\text{frame} = \frac{n}{\lambda} \quad (4-11)$$

视频帧的处理率，也称使用率为：

$$\text{PR} = \frac{1}{\lambda} * 100\% \quad (4-12)$$

其中，n 是视频帧数，λ 是用户设置的跳帧间隔。

这类跳帧算法的视频解码压力并没有得到缓解，比如解码 B 帧，需要依赖前后帧。因此一般实现的时候，是对视频采取完全解码，解码之后再根据步长选择分析处理。算法简单，容易实现，存在冗余分析或者漏检情况。

(3) 随机的跳帧算法

与固定步长的跳帧算法做法类似，实现的时候采取全解码，设置随机数种子，随机选取参与分析的帧。根据最优解搜索理论，其实往往随机更能够拿到比较关键的信息帧。但极端情况下，也存在关键信息漏检。因此也有研究提出了一种合

并算法：随机开始选取视频帧检测，后期可根据情况选择固定步长跳帧处理。

(4) 基于场景规则的增量跳帧算法

综合考虑视频解码压力和关键信息最大化两个因素，本文提出了基于场景规则的增量跳帧算法，结合了1、2两类跳帧算法，加入基于场景下的规则设计。主要思想是：建立关键帧处理步长与视频处理结果之间的函数约束关系，动态变化跳帧参数。针对本文的应用场景人数统计，因此本文设计检测人数与关键帧检测步长之间的关系如下：

$$d_n = \begin{cases} 1, & n_{people} = 0 \\ d_{n-1} + 1, & n_{people} \neq 0 \end{cases} \quad (4-13)$$

其中 d_n 为第n次检测之前最后连续检测目标数为0的统计计数，其中初始化为1，一旦没有存在检测目标，则为上一次计数加1，存在则重置为1；

$$\gamma_n = \begin{cases} 1, & d_{n-1} = 0 \\ \gamma_{n-1} + [(1 - \varepsilon) * \min(d_{n-1}, \max_step)], & d_{n-1} > 0 \end{cases} \quad (4-14)$$

其中， γ_n 为第n次的跳帧步长； d_{n-1} 为公式4-13定义连续目标检测统计参数，连续检测不到目标，则步长增量变大，一旦检测到目标，则保持步长为1， ε 为视频通道的加权参数， $\varepsilon \in [0,1]$ ， ε 越大，视频通道越重要，检测不到目标的时候，步长变化相对较小； \max_step 为最大跳帧步长，连续帧数越来越大的时候，控制持续跳帧间隔无限增大，后期可以根据不同的场景选择不同的参数。

对于这类算法设计的评价指标以及算法验证，将在第六章给出。本文提出的基于场景规则的增量跳帧算法有以下几类优点：

- (1) 设置视频通道的加权参数，增加重要视频通道的处理概率；
- (2) 动态的调整跳帧参数，减少冗余视频帧的处理，极大的节省计算资源，最大化多路视频中有效视频的处理效率。
- (3) 关键帧的数量可以控制，以平衡视频的压缩比，且处理的均为关键帧，不存在帧间压缩解码，减少视频解码压力。

4.4 本章小结

本章围绕多路视频智能处理系统的调度管理问题展开研究。首先，展开了并发流水线的性能评估及技术难点分析，为方案设计提供技术和理论参考；然后，针对系统整体的调度管理算法，给出了一个并行化方案的选择和设计，进一步提出整个多路视频前端预处理的算法架构。最后，详细介绍了调度管理系统中每个子模块的算法设计，包括视频调度任务槽的设计、数据共享设计、视频存储设计、视频跳帧采样算法设计、任务调度算法设计。其中，算法设计参考的规则均基于不同应用场景下的视频智能处理结果，因此整个调度管理算法被称为MSSM

(Multi-Scenario Scheduling Management) 算法，算法模块的验证将在第六章呈现。下一章将介绍视频调度系统一大组件——视频智能处理算法。

第 5 章 基于轻量级网络的视频智能处理算法研究

上一章设计了基于并发流水线的 MSSM 视频调度管理算法，其算法设计建立在视频智能后端处理结果反馈之上。为了保证视频调度的实时性，选择合理的视频智能处理算法十分关键。基于深度学习的视频智能处理算法成为主流的研究方向。然而，随着智能分析任务的难度增加，深度学习模型结构在不断加深，其需要的计算资源和内存资源也在量级增长，深度学习加速成为了一大研究热点。本章针对视频智能处理算法选择了单帧人数统计这一应用场景，提出了优化后的基于轻量级网络的视频智能处理后端算法。其一是提供调度管理算法的数据基础，其二为视频智能处理算法在终端部署，提供一个轻量级解决方案。

5.1 单帧人数统计算法对比和不足

目标检测已经成为计算机视觉领域最热门的研究方向之一，人数统计也成为了视频智能监控的一大需求，提出了一系列经典算法。虽然当前的检测精度都已经取得较好的成果，但是在实际应用场景中，由于目标遮挡，轮廓非刚性、姿态的可变性、图像分辨率、环境复杂性等因素，给准确的人数统计提出了巨大的挑战。此外，由于嵌入式智能终端的资源有限，深度学习等图像算法计算量巨大，使得算法应用成为了一大瓶颈。如何设计出一种简单有效、鲁棒性强、实时性好、精确度高的人数统计算法成为了一大挑战。目前图像处理相关技术包括背景分析、特征提取、目标检测、分割等，人数统计的准确率得到了很大的提高。根据其手段的不同，主要分为两大类：①间接法，又称基于特征的方法，是指建立目标特征和人数之间的函数关系来进行度量计算。②直接法，核心思想是提取图像特征，通过学习一个分类算法模型实现目标位置检测和分类。

5.1.1 传统机器学习方法

在人数统计领域上，传统方法在间接法和直接法中均有体现。在间接法中，建立目标特征和人数之间的函数关系来进行度量计算。Hashemzadeh M^[42]等提出了根据关键点和前景分割得到的人群密度和人群遮挡特征去估计人数；常庆龙^[43]等提出：提取前景变化和角点信息特征，通过后向传播(BP)网络优化特征提取，完成人数估计。在直接法中，首先提取目标的特征，如尺度不变特征变换 SIFT、Haar-like，方向梯度直方图 HOG 等，均为目标检测中常用到的特征。经典的

Navneet Dala 提出的 HOG+SVM 的行人检测模型^[44], 通过提取图像的 HOG 特征, 建立 SVM 的分类模型; 类似还有 Paul Viola^[45]提出的基于 Haar-Like+Adaboost 级联分类模型; 上述传统机器学习提取特征的方法, 对应为浅层学习, 精度达不到要求, 虽然提出了更好的方法, 如基于人体部件的部分形变模型 DPM 算法^[46], 分别提取人的头部, 四肢和躯干几个部分的特征, 并以其在整个身体的位置, 其距离远近来判断是否为行人, 该方法在环境复杂的场景下有较好的检测效果。但是因为多个分类模型的存在, 检测效率难以提高。

5.1.2 深度学习方法

目前, 深度学习的卷积特征结合传统机器学习分类成为目前的主流方法, 具有精度高, 泛化能力强等特点。基于各类深度卷积神经网络提出的人数统计算法, 通过建立更深更复杂的 CNN 卷积网络提取低层到高层的特征, 主要分为 one-stage 和 two-stage 两类: two-stage 是指基于区域建议的目标检测方法, 以 R-CNN (Region with CNN features) 为代表的算法, 其变种有 Fast-RCNN, Faster-RCNN, R-FCN 等, 这类算法采用了 proposal+classifier 的思想, 即是生成一定的候选区域, 再结合分类器进行目标分类, 符合类别的采用 NMS 等算法修正其边框位置。其二是端到端的检测方法, 包括 SDD、YOLO、STDN 等算法, 这类方法一般是基于回归的目标检测方法, 将问题转化为边界框预测和类别预测的回归, 从而实现端到端的目标检测和识别。与基于区域建议的目标检测算法相比, 在速度方面有了明显的提高, 且精度也慢慢接近于 two-stage 的效果, 在实际运用中作为第一选择。

经典的特征卷积网络有 AlexNet, VGG, GoogleNet, ResNet, DenseNet^[54-58]等, 无论是从网络的深度还是宽度, 都有了很大的变化, 这些网络也都在 ImageNet 比赛中拔得头筹, 其中 VGG, ResNet 等更是提出了一系列的优化结构, 为后续模型优化和压缩提供了思路。但是这种网络往往在模型参数巨大, 运行速度慢, 一般的硬件平台由于其计算资源有限, 不能用于实际应用。

5.2 头部检测网络优化设计

本文的视频智能处理算法设计主要通过构建一个全方位的人头检测数据集, 基于目前流行的轻量级网络 MobileNet^[47]实现图像特征提取, 替换 SSD^[48]检测框架的 VGG 部分, 将其中的普通卷积替换为深度可分离卷积, 引入小尺度特征以及全局信息; 替换激活函数为 SeLU(scaled exponential linear units)^[49], 提高模型检测的鲁棒性。实验证明, 在数据多样性扩展的情况下, 基于 MobileNet 改进后

的网络结构在人数统计应用上具有与复杂网络相当的检测精准度，但却有更低的延迟，优良的鲁棒性，同时模型参数数量得到有效降低。

5.2.1 MobileNet 特征提取网络

Mobilenet 网络主要是 Google 为了解决移动嵌入式终端无法应用的问题，提出的一种轻量级的深层神经网络。其核心思想就是通过分解冗余卷积层，达到降低网络参数量级，使得模型在精度和复杂度上得到均衡。

(1) mobilenetV1

该网络加入了深度可分离卷积核的设计，代替传统卷积操作，通过引入两个全局超参数：宽度参数和分辨率参数，在效率和精度之间有效的进行平衡，这两个参数允许本文根据实际问题，为其应用选择合适的模型。

①深度可分离卷积

将标准卷积分解为一个深度卷积和一个点卷积。这类分解可以实现和传统卷积一样的效果，但是可以有效的降低模型的大小。下图为传统卷积和深度可分离卷积的对比。

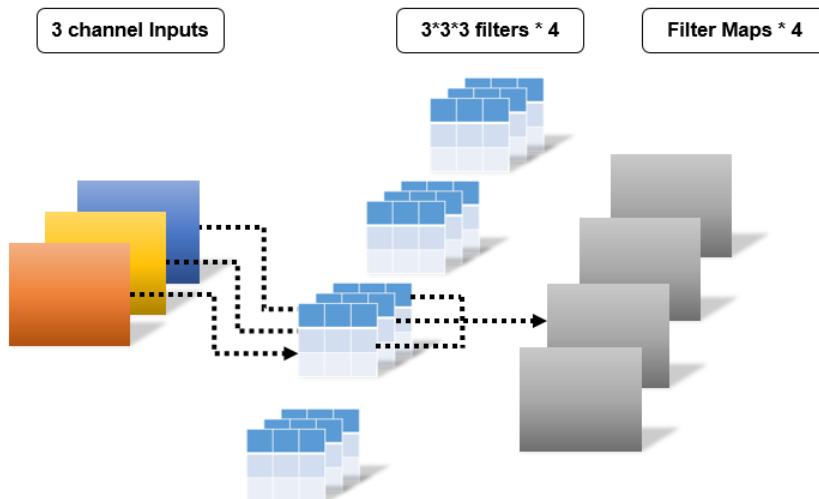


图 5-1 标准卷积运算

Figure 5-1 Standard convolution operation

标准卷积运算：输入为 $M \times M$ 的三通道 rgb 图像，经过 $3 \times 3 \times 3 \times 4$ 卷积核的卷积过程（假设输入通道：3，输出通道：4），最终输出 4 个特征图，具体大小还要参考 Padding 和 strides 步长参数。

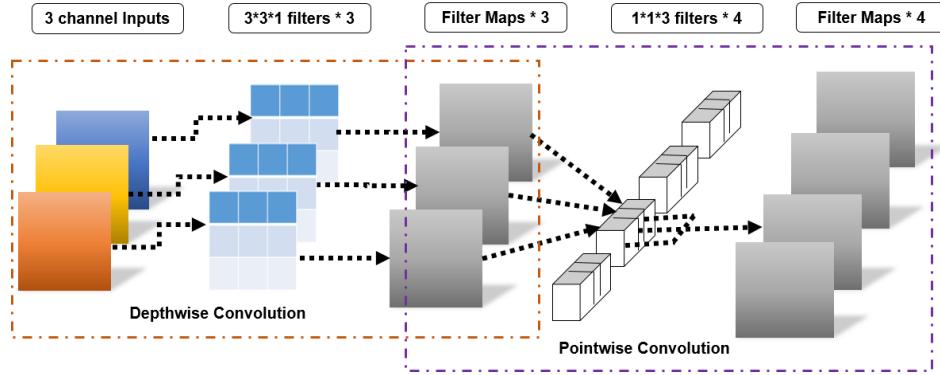


图 5-2 深度可分离卷积运算

Figure 5-2 Depthwise convolution operation

可以发现，不同于常规卷积操作，Depthwise Convolution 的对每个通道做单独的卷积，也即是一个通道只被一个卷积核卷积。Depthwise Convolution 完成后的特征图数量与输入层的通道数相同，无法扩展特征图，而且这种运算对输入层的每个通道独立进行卷积运算，没有有效的利用不同通道在相同空间位置上的特征信息。因此需要 Pointwise Convolution 来将这些特征图进行组合生成新的 Feature map。这样分离操作，计算量大大减少，两者计算量比值(CCR=DW Conv Cost / Std Conv Cost)关系如下公式计算(此处省略了宽度超参数和分辨率超参数)

$$CCR = \frac{G_k^2 * M * G_f^2 + M * N * G_f^2}{G_k^2 * M * N * G_f^2} \quad (5-1)$$

$$CCR = 1/N + 1/G_k^2 \quad (5-2)$$

其中 N 是 feature maps 的通道，通常会比较大，一般会大于 10。 G_k^2 是卷积核的 size，一般为 3*3，该比值是小于 1 的数。因此 depthwise separable convolution 相比于标准的卷积会减少计算量。

此外，增加两个超参数后的网络计算量公式如下：

$$G_k^2 * \alpha M * \rho G_f^2 + \alpha M * \alpha N * \rho G_f^2 \quad (5-3)$$

其中， α 为宽度参数，用以控制输入输出的通道数，减少 featuremap 的数量，取值为 0~1； ρ 为分辨率参数，用以控制数据层的分辨率；二者共同作用，可以进一步减少计算量。此外，上述核分解方法不仅可以减少计算量，还适配市面上绝大部分的移动端 CPU 指令加速硬件。

(2) mobilenetV2

在 MobileNetV1 的基础上，提出了升级版，一方面保证了模型准确性，另一方面减少了计算量^[50]，在本文的应用场景下，基于以下两种基本结构，该特征提

取网络将引入全局信息，丰富数据的特征信息：

①Linear bottlenecks, 用线性变换层替换通道数较少层中的 ReLU 激活函数，降低信息损耗。

②Inverted residual structure, 使用一个快捷链接（shortcut）链接了 block 的输入与输出（element-wise add）相较于原始的残差结构，这里颠倒了内部数据维度的变换顺序，这样做可以节省内存，同时丰富特征信息。下图为原始 residual structure 和 Invert residual structure 之间的对比：

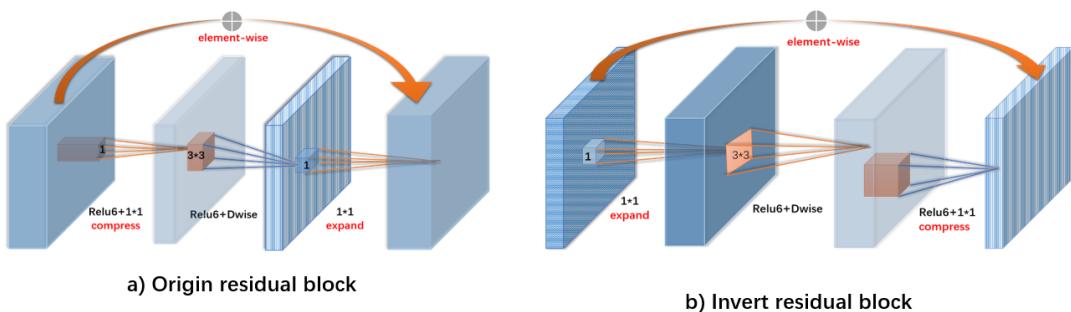


图 5-3 改进残差结构对比

Figure 5-3 Improved residual structure comparison

对比原始的 RBlock, MobilenetV2: ①经过一个 1×1 的 Conv layer, 将 feature map 的通道‘扩张’；②经过 3×3 Conv layer 提取特征信息；③经过一个 1×1 的 Conv layer, 将 feature map 通道‘压缩’回原始大小。原始设计则是先‘压缩’再‘扩张’。

5.2.2 SSD 目标检测网络

SSD 是一种单次检测模型，核心思想是在多尺度特征图上，一次回归得到目标的位置和类别。通过分析，模型存在小目标检测不准的问题，其中影响小目标检测的因素主要是，特征图的分辨率、全局信息和特征提取能力，因此本文在基础网络的选择上考虑到了特征提取能力因素，其中 Mobilenet 系列的深度可分离卷积以及 residual block 结构，可以提高用于 SSD 检测的特征图的高分辨率低层特征表达能力。此外原始 SSD 框架，激活函数使用的是 ReLU。本文通过分析对比，将其修改为 SeLU 激活函数，引入自归一化的特性。其中 shaohua.等人对 SeLU 和其他激活函数做了对比，证明其有效性和鲁棒性，甚至超过了 Batch Normalization^[49]。

(1) SSD 检测框架

SSD (Single Shot MultiBox Detector) 属于 one-stage 方法，其中 MultiBox 指明其为多框预测，对目标类别和位置的预测都是基于卷积特征。相比于之前的

YOLO，其改进方向主要是：①提取多尺度的特征图来做检测 ②设置不同尺度和长宽比的先验框用以提取目标位置。以下为 SSD 的基本框架：

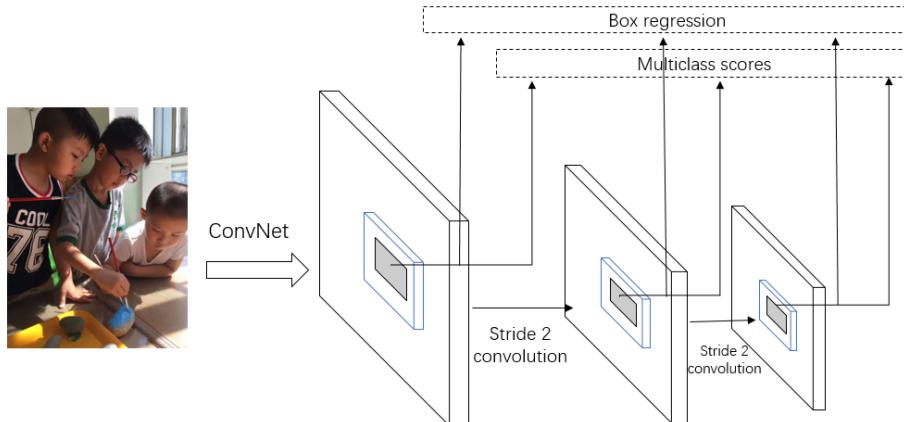


图 5-4 SSD 检测基本框架

Figure 5-4 Basic framework of SSD detection

(2) ReLU 和 SeLU 激活函数对比

神经元加入了激活函数，就具有了非线性表示的能力，这也是神经网络与线性分类器的最大的不同之处。相比最原始的激活函数 Sigmoid，ReLU 具有更宽的兴奋边界，同时其稀疏激性可以使得 ReLU 经过多次的 BP 训练，其梯度降低的幅度不会很大，可以很好的传递梯度，适合训练深度神经网络。

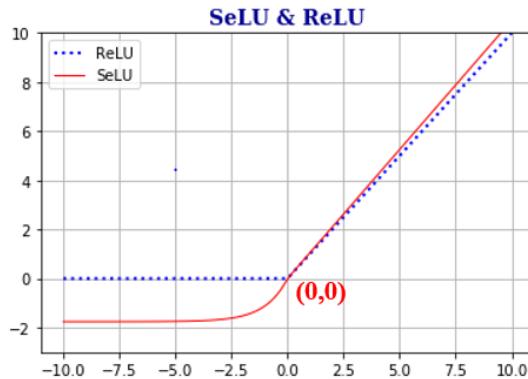


图 5-5 ReLU 和 SeLU 激活函数的对比

Figure 5-5 Comparison of ReLU and SeLU activation functions

但是，实验证明，ReLU 激活函数容易导致训练中断，其强制的稀疏处理会减少模型的有效参数容量（在 $x < 0$ 的时候，其负梯度置零，不能再次激活）。此外，ReLU 和 Sigmoid 的一个相同点是结果全是正值，降低了特征的表达能力。

2017 年，发表了一种新的激活函数 SeLU，引入了自归一化的属性。以下为 SeLU 和 ReLU 激活函数的数学公式：

$$ReLU_{(z)} = \begin{cases} z & z > 0 \\ 0 & z \leq 0 \end{cases} \quad (5-4)$$

$$SeLU_{(z)} = \gamma \begin{cases} z & z > 0 \\ \alpha(\exp(z) - 1) & z \leq 0 \end{cases} \quad (5-5)$$

where $\gamma \approx 1.0507$, $\alpha \approx 1.673$

其中 SeLU 单元主要使用一个函数，建立前后两层神经网络的映射关系，同时将参数变换到固定的均值和方差以达到归一化的效果。相比 ReLU 有以下优点：①强收敛属性：即使数据存在噪声和干扰，通过多层的前向和反向传播之后，还是会较快的收敛②带有正则化的效果，增强算法鲁棒性③其函数激活值的方差存在上下界，有效防止梯度消失和梯度爆炸。

(3) 模型结构

本文选用基于 mobileNet 的 SSD 检测框架，左边部分是特征提取网络 Mobilenet，它囊括了 conv1-conv13 层的输出。在 conv13 之后添加了 8 层卷积，选取其中 4 层以及 Mobilenet 的 conv11 和 conv13 作为 SSD 的尺度特征图，引入小尺度特征以及全局信息，最后归于一个预测单元，其激活函数采用 SeLU 函数。（注意，此处省略了 MobileNetV2 版本结构）

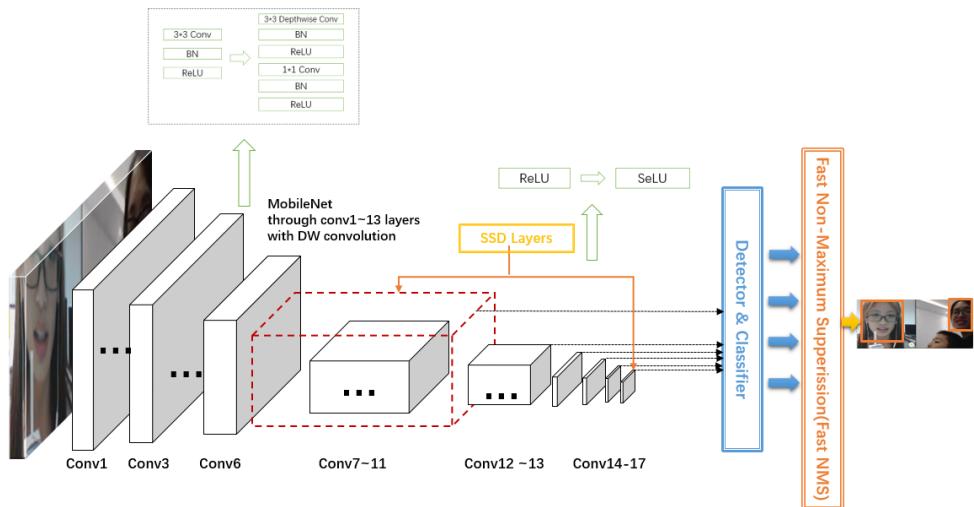


图 5-6 MobileNet_SSD 检测模型结构

Figure 5-6 Detection model structure of MobileNet_SSD

5.2.3 评价标准

本文的应用场景是单帧人数统计，实际上一个目标检测模型，关于目标检测的模型评价指标，一般是 IOU (Intersection Over Union) [51]，可以理解为系统预测出来的框与真实框的重合程度。

$$\text{IOU} = \frac{\text{Detection Result} \cap \text{GroundTruth}}{\text{Detection Result} \cup \text{GroundTruth}} \quad (5-6)$$

其评价该检测目标是否有效的时候，一种常见的方式是设置一定的 IOU 阈值，只要算法找到检测框的 IOU 大于这个阈值，则为有效检测，则计数。

针对本文的应用场景——单帧人数检测，为了对比同一段视频在不同模型下检测到的准确的人数数量，且便于比较，本文选择了 RMSE 作为评价指标^[52]，其计算公式是：

$$RMSE(X, f) = \sqrt{\frac{1}{m} \sum_{i=1}^m (f(x^i) - y^{(i)})^2} \quad (5-7)$$

其中 $y^{(i)}$ 为每一张图的原始人数， $f(x^i)$ 为检测出的人数， m 为处理的视频帧数。此外，本文使用每秒检测帧数 (frames per second, FPS) 衡量智能处理速度，并以 25 fps 作为实时性考量临界值。

5.3 算法验证

5.3.1 实验环境及数据集制作

(1) 实验开发环境

实验硬件环境为 Ubuntu16.04+cuda8.0+GTX1080。软件环境使用了 Keras 作为深度学习框架定义不同的模型，后端是 TensorFlow 实现，支持 GPU 计算。

(2) 数据集制作

构建好人头检测网络后，需要通过大量的全方位人头检测数据集来训练和调优，用以视频智能处理模块加载。在训练目标检测网络模型时需要用到特殊的数据集格式，目前常用的数据集格式有 ImageNet、COCO、PASCAL-VOC2007、PASCAL-VOC2012 等。本文的模型数据格式选择了 PASCAL-VOC^[53] 数据格式。PASCAL-VOC 数据集包含 5 个文件夹，分别为 JPEGImages、Annotations、ImageSets、SegmentationClass、SegmentationObject。在针对目标物体的检测和识别的应用中，涉及到了其中的三个文件夹：① JPEGImages：包含了用于训练和测试网络的物体图片，这些图片尺寸大小可以是不一致的，以 jpg 为图片后缀，并按照一定格式统一命名；Annotations：用于存放 xml 标签文件，该文件内容

主要包含了文件名、图像长宽及通道、物体类别等信息；ImageSets：文件夹存放各个数据集合的 txt 文件。如 Main 下的 train.txt 中记录的是用于训练的图片集合。

表 5-1 图像数据集列表

Table 5-1 Image dataset list

DataSet	Labels	Datasize	Pixelsize
Pascal VOC 2007-06	21	9963(2097)	500*375
Pascal VOC 2012-11	21	17125(5138)	500*375
INRIA	2	2416	96*160
MyData	2	5000	300*375

本文定义的人头包括正脸、侧脸，仰头，低头等姿势，而目前这方面针对人头统计的数据集，并没有很多。为了构建人头的全方位数据集，采用了 INRIA 行人数据集、PASCAL-VOC 2007+2012 数据库内标注为人的数据。但是符合条件的数据还比较少，因此，在网上抓取了大量包含目标（人）的图像数据，爬取的数据包含上述姿势，同时关注点包含面部肌肤以及发色，脖子轮廓等。使用 LabelImg 工具进行重新标注。以下为数据标注过程：

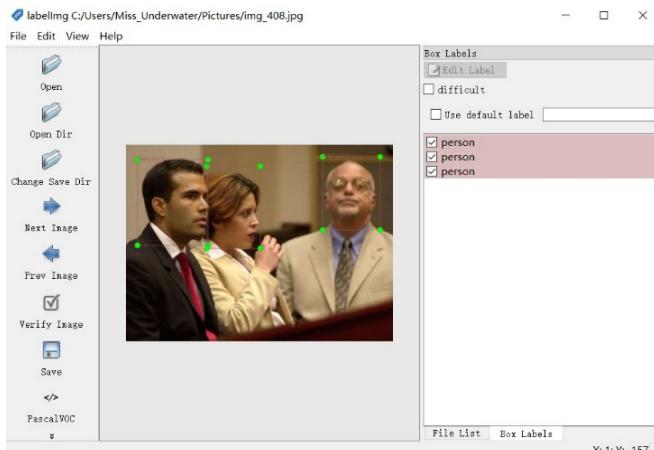


图 5-7 LabelImg 数据集标注界面

Figure 5-7 Data set labeling interface of LabelImg

以上标注过程，同时为了拿到全方位以及特征丰富的数据，标注的时候本文不会仅仅关注面部肌肤，肩部以上的特征都会加入目标框，同时对于 PASCAL-VOC 数据也进行了重新标定，这样可以获得更丰富和准确的特征信息，避免数据集信息的过拟合，增大模型的鲁棒性。

经上述方法处理得到的数据样本，按照 8:2 的比例划分训练集和验证集。此外，本文在将数据放入模型的时候，采用了一系列数据增强的工作，比如随机裁剪，亮度变换等，以增强数据的复杂性。

5.3.2 网络模型训练和测试

(1) 网络模型训练

基于 Keras 深度学习框架，搭建上文提出的检测网络模型。因为数据集类别比较单一和数量较少，本文选了 MobileNet 在 ImageNet 数据集上做预训练，再做 finetune。为了在网络参数，检测效果及实时性上比较，本文选择了现在比较流行的 VGG 卷积网络，和 AlexNet 等网络一样，该网络认为：卷积神经网络的深度增加和小卷积核的使用对网络的最终分类识别效果有很大的作用，但是也因此丧失了检测的实时性，难以用于实际。

本文搭建了 VGG16+SSD,MobileNet+SSD 两类检测框架，其中图片的输入大小为 300*300。模型的参数包括 bbox 的个数、scale 范围、aspect_ratios 大小、位置偏移等。基础特征网络的权重选择了 ImageNet 预训练权重做 finetune，训练轮数均为 10000+。其中得到网络的损失值变化曲线图，如下所示：

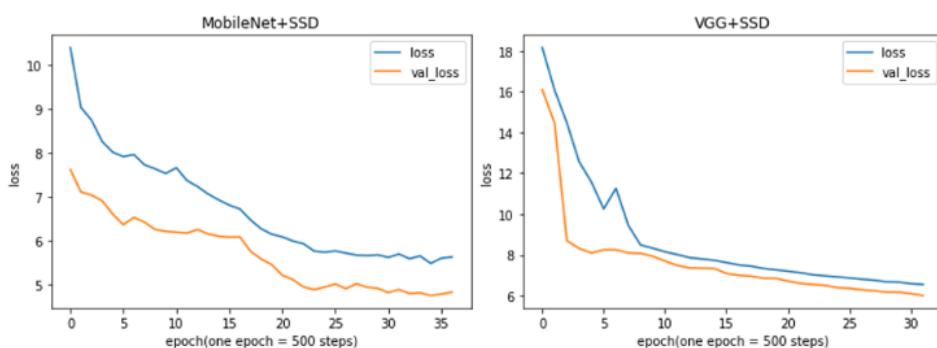


图 5-8 损失值变化曲线对比图

Figure 5-8 Loss value curve comparison chart

(2) 网络模型测试

训练结束后，可以得到对应最优表现的网络模型文件，这些文件存储着网络的图结构、参数名以及参数值信息。编写相应的测试代码，选用一段视频做人数统计，根据每帧视频真实人数和检测人数计算 RMSE，并对模型的运行参数和效率做了一系列统计。下图为 MobileNetV1+SSD 的部分可视化结果，图像右上角统一显示检测到的人脸个数：



图 5-9 实验结果可视化展示

Figure 5-9 Visualization of experimental results

5.3.3 实验结果分析

为了验证本文方法在参数大小、模型准确率、检测效率等几个方面有较好的均衡，本文对比了 VGG+SSD 和 MobileNet+SSD 的相关实验结果，如下表所示：

表 5-2 实验结果对比列表 (IOU=0.7)

Table 5-2 Comparison of experimental results IOU=0.7

BaseModel	params	RMSE(m=500)	FPS
VGG16(ReLU)	26M	0.89	46
MobileNetV1(ReLU)	7M	1.25	67
MobileNetV2(ReLU)	6M	0.93	85
VGG16(SeLU)	26M	0.82	47
MobileNetV1(SeLU)	7M	1.19	67
MobileNetV2(SeLU)	6M	0.87	85

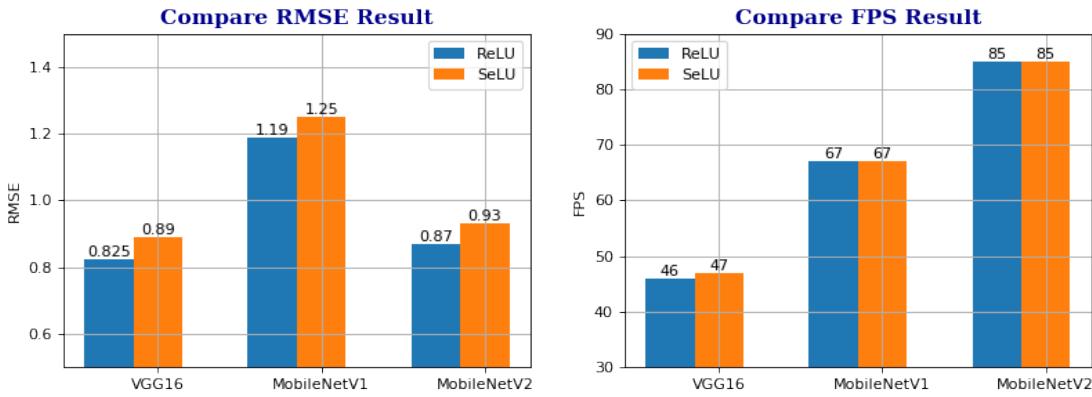


图 5-10 实验结果对比柱状图

Figure 5-10 Experimental results comparison histogram

通过网络训练过程损失的可视化结果，本文可以发现 MobileNet 以及 VGG 都能较快的收敛到一个范围，图中仅展示了 0~40epoch 的结果(1 epoch = 500steps)。各个模型在用一段视频上进行验证，本将视频的每一帧存为图片，抽取了本文 MobileNet+SSD 的部分结果展示，有以下几类现象：①能较好的识别低头和部分遮挡的人 ②存在识别错误的情况 ③存在人脸重复检测的情况 ④对比最后的评价标准 RMSE，MobileNet 相对比 VGG 稍微差一点。在实时性测试上，本文测试了 Inference 一帧的时间，相比 VGG 有很大的提高。在实际应用的时候，往往涉及到视频采集，编码，预处理，解码等模块，效率会相应的损失，其中第六章给出了部分视频前端处理的效率分析。

5.5 本章小结

本章围绕视频智能处理算法的设计展开研究。针对视频智能处理的应用场景——基于人头检测的单帧人数统计，研究并提出了一种基于轻量级网络的改进目标检测算法模型。首先，对目前的单帧人数统计算法进行了比较，提出其中的不足之处；之后，针对本文提出的算法模型进行具体介绍：首先，详细介绍了该网络结构的两个组件：MobileNet 特征提取网络和优化的 SSD 目标检测网络，并介绍了实验的评价标准；然后，介绍了相关实验准备：①实验数据集准备：公开数据+个人数据；③实验环境；最后，搭建实验模型，进行训练、测试和相应的结果分析。实验证明，基于轻量级网络改进后的网络结构在单帧人数统计应用场景上，其精度和效率得到了有效的均衡，也即是更低的延迟，优良的鲁棒性，同时模型参数数量得到有效降低。

第6章 多场景视频智能处理系统实现与测试

以上几章展开了本文研究课题相关算法设计和系统总体设计的讨论,本章将具体介绍系统关键模块设计以及对应算法模型实验。首先,介绍了系统的开发环境;然后介绍系统的关键模块设计与实现;其次,介绍了系统的前端设计;最后,对第4章涉及到的算法模块进行相关实验验证和结果分析。最终的实验结果证明了前序章节算法设计的正确性。

6.1 系统开发环境

(1) 实验环境

实验测试环境考虑两个方面因素:①实验参数配置:对摄像头的参数进行配置,包括分辨率,权重分配,存储时间范围等②实验环境的目标数量和目标质量:不考虑环境因素,应包含以下样本数据:正面人头、侧面人头、仰头、低头、部分遮挡等;目标数量则可以动态控制。实验环境的多样性也是为了验证本文第三章提出的视频跳帧算法等,是否达到视频信息最大化、处理加速的目标。

本系统的开发环境是 Ubuntu 16.04 LTS 系统,以下是系统参数:

表 6-1 实验系统参数

Table 6-1 Experimental system parameters

设备	参数
CPU	Intel® Core™ i7-8550U CPU 1.80GHZ
内存	16GB
硬盘	SATA1.0T
显卡	Nvidia GTX1080
网络交换机	百兆以太网 8 路

本系统选用了海康威视网络摄像头,通过 8 路网络交换机与主机连接,以下是摄像头的音视频参数配置:



图 6-1 网络摄像头音视频参数配置

Figure6-1 Webcam audio and video parameter configuration

图 6-2 为本文的实验 demo 环境，其中多路摄像头连接网络交换机接入 PC 端，对应的设备参数如上一节所述，整体系统功能的验证则基于本文提出的单帧人数检测模型实现，实现原理是面向多个角度的头部检测，虽然第 5 章的实验结果仍然存在不足，但通过实验整体效率和精度的对比来看，其轻量级的视频智能处理算法由于其模型参数大大降低，将更好的运用到实际生活中。

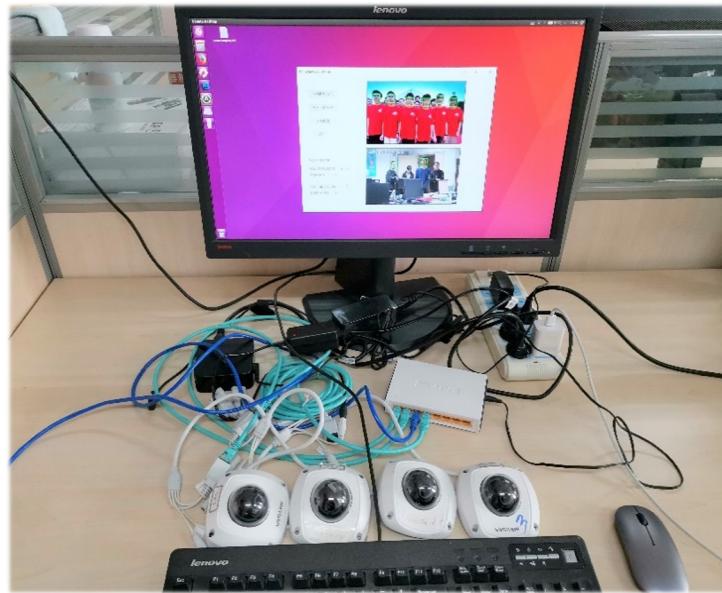


图 6-2 实验环境搭建

Figure6-2 Experimental environment construction

(2) 实验数据

本节包含整个系统的算法验证和性能分析，实验比较的过程中，实时视频不具有稳定性，因此本文设计了几类数据，用以性能测试，下面的部分实验，将使用这些数据进行对比实验，以下是数据的属性列表：

表 6-2 实验数据参数列表

Table 6-2 Experimental data parameter list

数据名	编码	分辨率	视频帧总长	人数	帧率/关键帧间隔
1-1	H264	720P 960P 1080P	198	1	25/4
2-1	H264	720P 960P 1080P	198	2	25/4
3-1	H264	720P 960P 1080P	198	3	25/4
4-1	H264	720P 960P 1080P	198	4	25/4
5-1	H264	720P 960P 1080P	198	5	25/4

采用 FFmpeg 进行视频数据采集，之后再使用转换命令将数据扩展为其他分辨率，其中分辨率、帧率、关键帧间隔可以通过摄像头进行参数配置。分辨率为 720P (1280*720)、960P (1280*720)、1080P (1920*1080)。

6.2 关键模块设计与实现

前两节介绍了视频智能处理系统的架构和流程设计，将视频智能处理模块抽象，使其面向多场景具有可延展性。本节将针对前面提出的算法设计给出相应的参数设计、任务设计以及数据设计，其调度管理算法也正是基于此部署。

6.2.1 参数结构设计

表 6-3 系统参数结构设计

Table 6-3 System parameter structure design

参数	说明
cam(i)_enable	Cam(i)使能
cam(i)_IP	Cam(i)IP 地址
cam(i)_save_enable	Cam(i)存储使能
cam(i)_FPS	Cam(i)视频帧率 (fps; 每秒关键帧个数)
cam(i)_weight	Cam(i)视频通道权重
cam(i)_save_path	Cam(i)视频存储路径
cam(i)_save_time	视频存储间隔 (分钟)
min_disk_space	磁盘最小容量 (MB)
min_data_delete	数据删除参数 (天)
image_pixel	图像数据分辨率
video_keyframe_num	视频跳帧最大值 (keyframe)
IOU_value	人头检测阈值

6.2.2 任务复用设计

第三章的任务调度算法是基于任务复用实现，将任务封装成为 worker 组，内部有序、外部透明，便于拓展和管理。系统集成多种功能，包括各类信息交互和反馈机制，比如多路视频采集、存储、处理结果可视化反馈等。功能之间存在并行、交叉的情况。本文选用多进程或者多线程实现不同的系统模块，下图是本文的多进程、线程规划设计图：

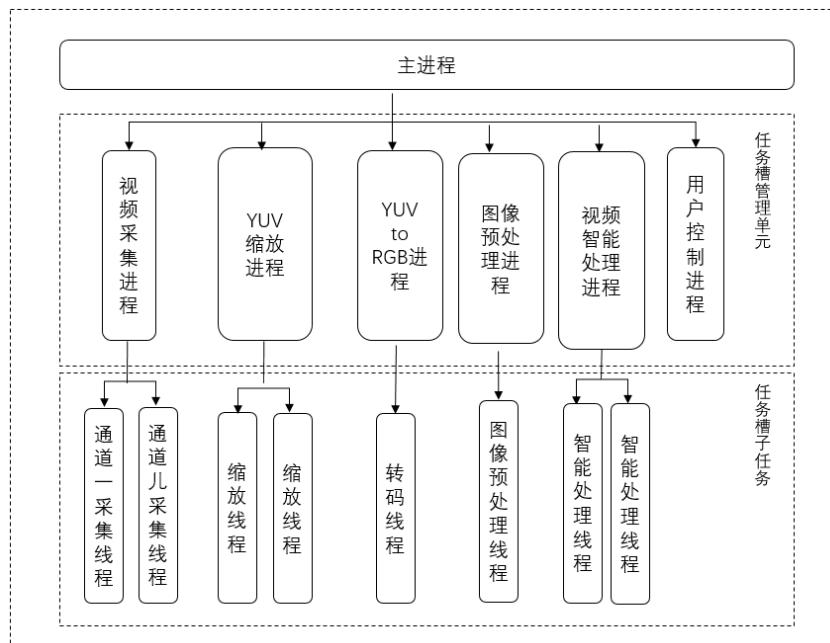


图 6-3 视频智能处理系统多线程设计

Figure 6-3 Video intelligent processing system multi-threaded design

多路视频同时处理，单个通道视频流由一个独立的解码线程处理，其中解码线程内部使用 FFmpeg 进行处理，支持内部解码并行，充分利用多核的计算能力。解码过程，参照本文的跳帧策略筛选数据进行解码；解码后将进行 yuv 缩放、转rgb、图像增强等预处理工作，获取视频处理图像数据。一对一线程往往会造成大量系统资源且造成线程之间不同步，等待执行等情况。任务复用则支持根据不同的任务执行时间和 CPU 占用等配置线程数量，便于系统扩充，且任务线程之间的创建和销毁等互不影响。结合 3.2.1 的流程设计，系统主进程启动之后，便开始进程和线程的初始化。其中进程和线程的管理是通过任务槽进行管理，任务槽是抽象的数据结构，其属性设计第 4 章已经说明，具体实现是采用类设计，系统初始化的时候进行相关初始化，此后根据系统的调度管理策略更新任务槽的状态，进一步管理进程和线程。

6.2.3 共享数据设计

根据第三章提出了多路视频处理算法架构，多路视频并行处理，部分数据共享存储空间。以下是涉及到的数据列表，包括单源数据和多源数据。其中共享数据的数据结构设计在第三章已经给出。

(1) 共享数据列表

表 6-4 共享数据列表

Table 6-4 Shared data list

数据名	说明
视频 h.264 码流数据	单源数据，每路视频对应一个数据队列
yuv 数据	多源数据，根据跳帧策略得到的解码数据
yuv 缩放数据	多源数据，yuv 缩放得到
rgb 数据	多源数据，yuv 数据转码得到

(2) 共享数据实现

经过分析，共享数据有以下几类特点：

- ① 数据队列支持循环存储，数据更新频率高
- ② 数据大小不固定（数据结构大小设置）
- ③ 数据被多个任务同时访问（读写互斥）
- ④ 数据地址在系统初始化固定（任务通过地址调取相关数据）

这四个特点决定了共享数据的实现需要满足：①数据访问灵活，支持地址访问②支持互斥访问③支持再分配。本文的数据队列属于多路视频共享地址空间，数据输入均为同一类型的任务队列，输出也为同一类型任务队列。上一节以及第三章的任务复用设计，那么需实现不同任务队列之间的数据通信，通过分析，本文采用共享内存实现数据共享。每个进程都会维护自己的内存地址，使用共享内存，则可以在进程建立的时候将内存地址进行绑定，达到共享内存地址空间的目的。此外，共享内存并不需要每次都重新建立映射，而是在整个进程运行期间一直保持，直到通信完毕，因此其通信效率是最高的。此外多个进程共享同一地址空间，需要建立同步机制实现数据的互斥访问。

6.3 系统前端设计

整个界面划分为三个部分：①视频控制：包含三类功能按钮：摄像头的显示控制按钮：单路视频以及多路视频；系统参数配置按钮，用以配置系统相关参数，包括跳帧参数等；系统注销按钮；视频跳帧参数配置，其中“-1”表示其不处理，

“1”表示间隔 1 帧处理等；②视频显示：支持视频实时显示，若选择处理，则一并在视频显示界面显示视频处理结果，如人脸检测结果(标出人的位置)；③视频处理结果显示：实时显示当前视频的处理结果，包括检测人数和处理速度 FPS。

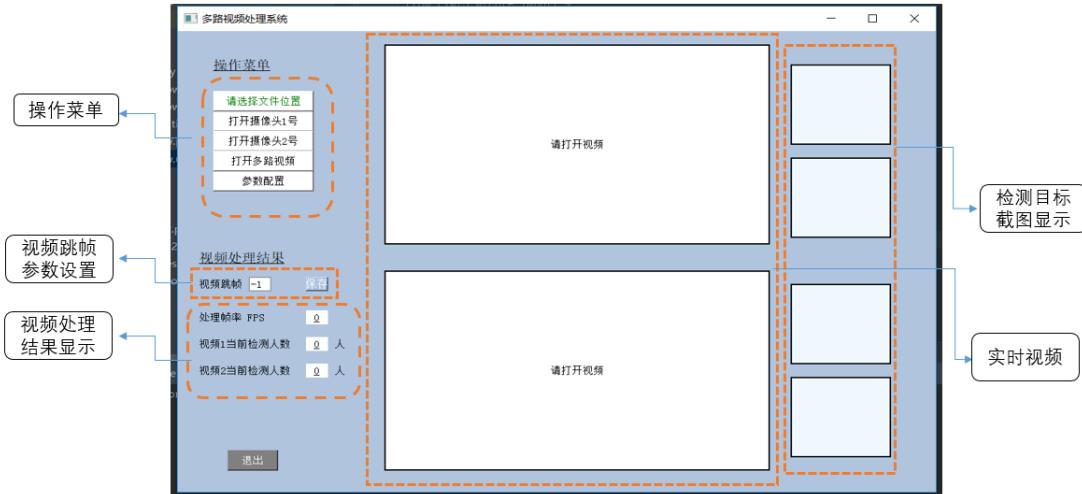


图 6-4 多路视频智能处理前端设计

Figure 6-4 Multi-channel video intelligent processing front-end design

系统参数配置主要参考 6.2.1 的系统参数设计，主要涉及多路摄像头 IP 地址，权重，存储参数等的配置，其中也包括视频智能处理的部分参数配包括 IOU 检测阈值，图像分辨率参数等。

6.4 系统实验及结果分析

6.4.1 视频解码性能对比实验

(1) 实验过程描述

本节是测试视频解码带来的影响，视频码流(AVPacket)到视频帧(AVFrame)这一过程，视频内部解码的部分过程如宏块解码是可以并行的，目前也针对此研究出很多方法，本文采用 FFmpeg 代码实现，FFmpeg 针对视频解码提供了并行机制，通过对不同分辨率、不同视频源、是否并行下的解码时间，探究视频解码的影响。

第 1 组：同一视频源不同分辨率解码性能

①选择的视频数据为 1-1；纵坐标代表每帧的解码时间，其中点坐标为关键帧的解码时间，其余为非关键帧的解码时间；横坐标为视频帧的序号；这里采用非并行的方式进行测试。

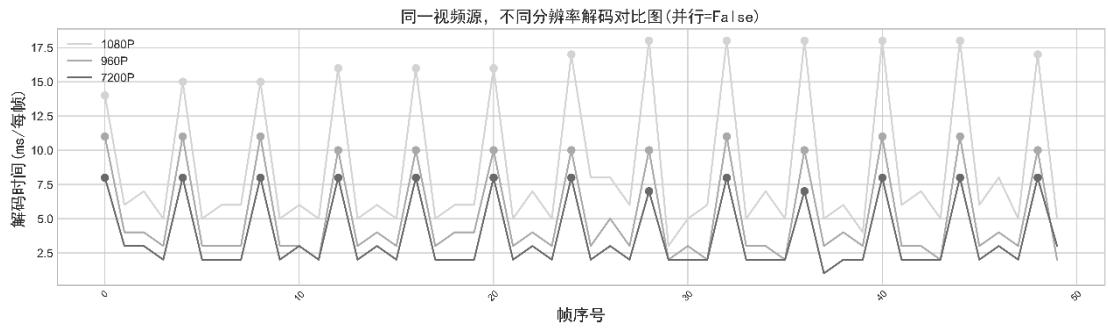


图 6-5 同一视频源，不同分辨率解码对比实验 1

Figure6-5 Same video source, different resolution decoding comparison experiment1

②选择的视频数据为 1-1; 和上述实验的不同之处在于这里采用 FFMEPG 内部解码线程并行，其中变量为分辨率和并行的线程数，纵坐标为视频全部解码的时间（从视频读取到解码结束，包含一些额外操作，但不存在影响）；横坐标：并行的线程数。

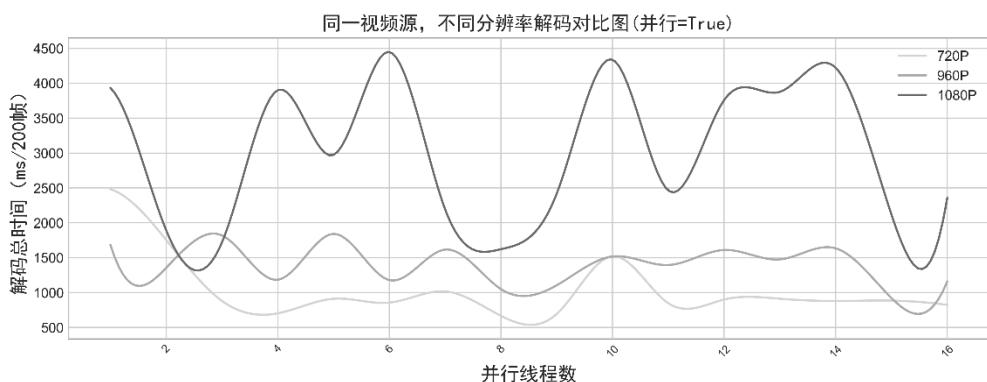


图 6-6 同一视频源，不同分辨率解码对比实验 2

Figure6-6 Same video source, different resolution decoding comparison experiment2

第 2 组：不同视频源同一分辨率解码性能

①选择的视频数据为 1-1, 3-1, 5-1; 纵坐标代表每帧的解码时间，其中点坐标为关键帧的解码时间，其余为非关键帧的解码时间；横坐标为视频着帧的序号；分三个分辨率进行非并行解码测试：

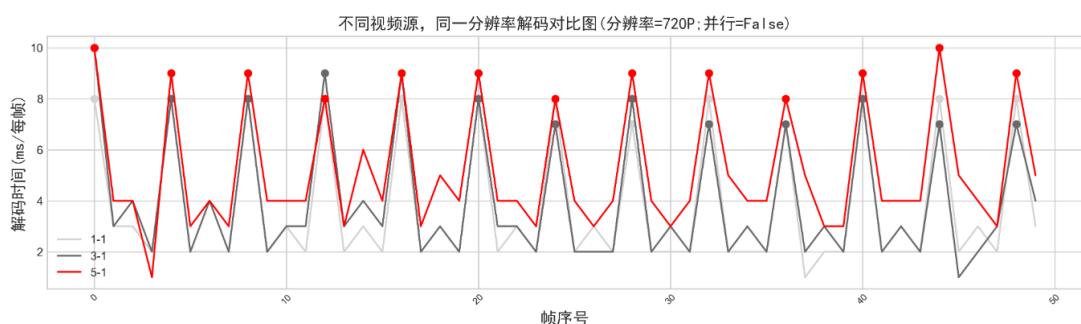


图 6-7 不同视频源的解码性能 (720P)

Figure6-7 Decoding performance of different video sources (720P)

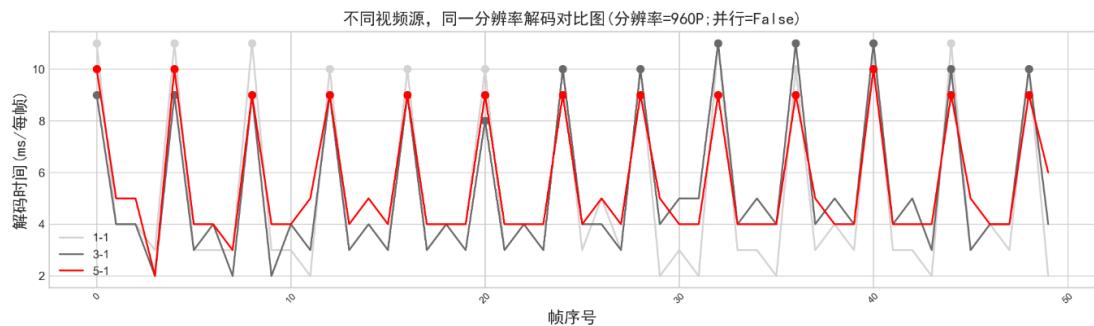


图 6-8 不同视频源的解码性能 (960P)

Figure6-8 Decoding performance of different video sources (960P)

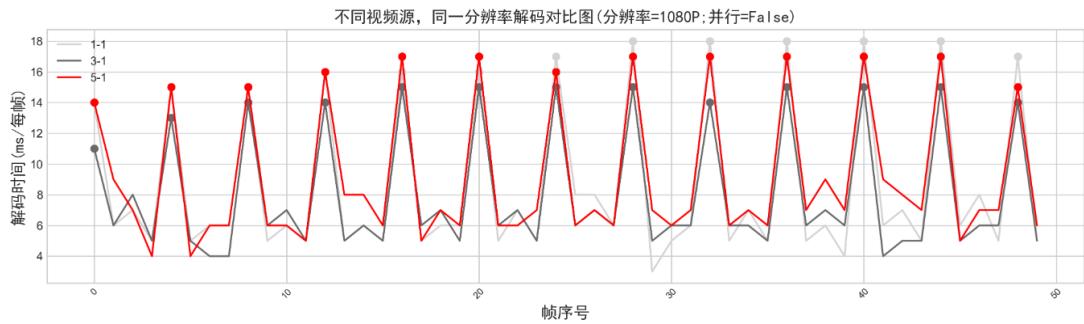


图 6-9 不同视频源的解码性能 (1080P)

Figure6-9 Decoding performance of different video sources (1080P)

②选择的视频数据为 1-1, 3-1, 5-1; 和上述实验的不同之处在于这里采用 FFMEPG 内部解码线程并行, 其中变量为分辨率和并行的线程数, 纵坐标为视频全部解码的时间 (从视频读取到解码结束, 包含一些额外操作, 但不存在影响); 横坐标: 并行的线程数。

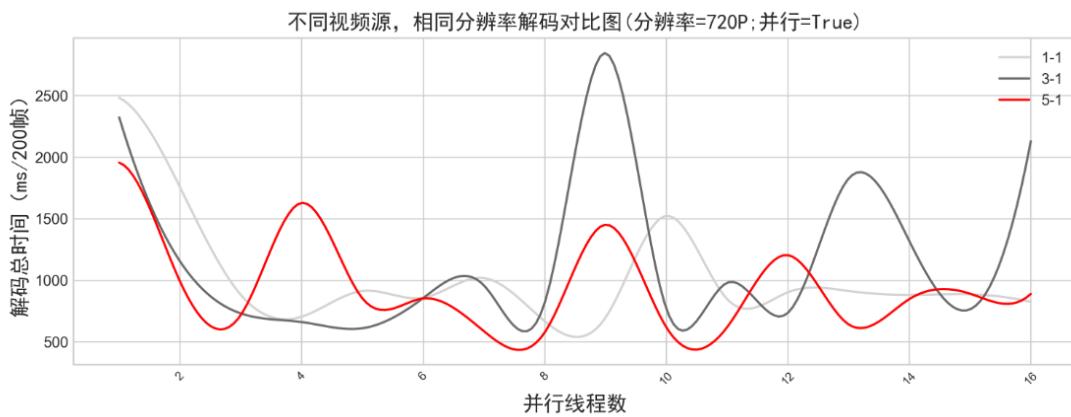


图 6-10 不同视频源的解码性能 (720P)

Figure6-10 Decoding performance of different video sources (720P)

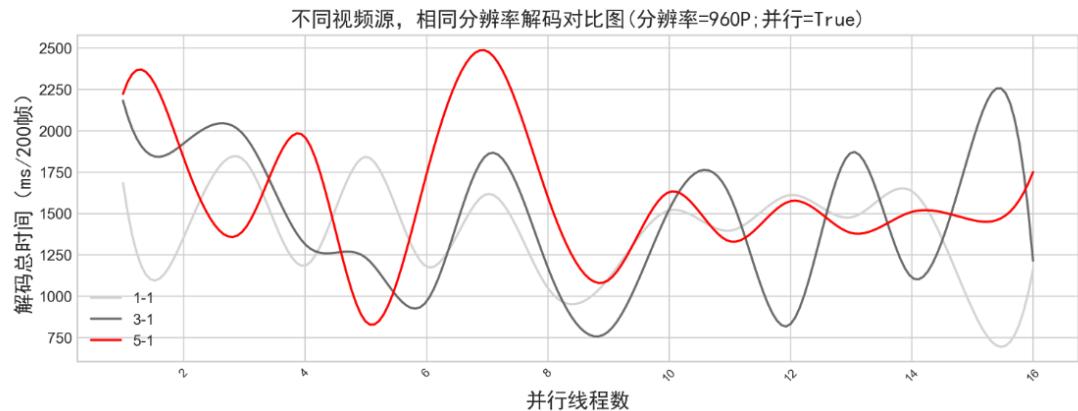


图 6-11 不同视频源的解码性能 (960P)

Figure 6-11 Decoding performance of different video sources (960P)

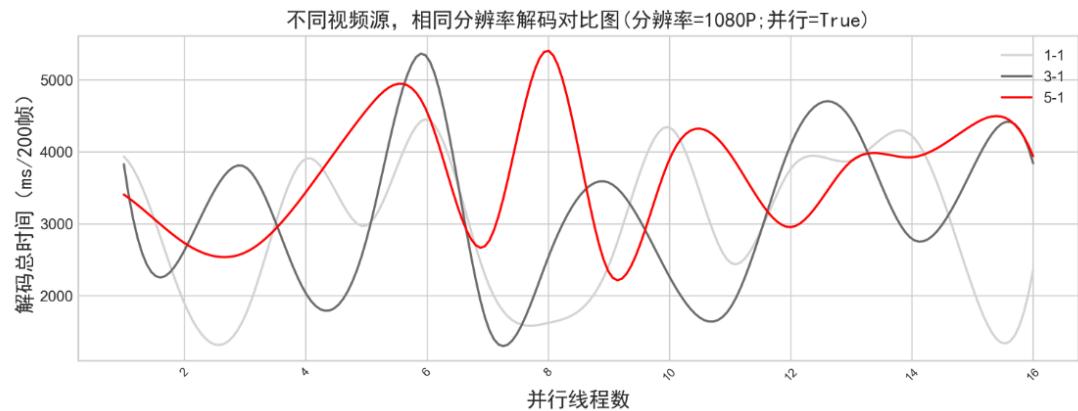


图 6-12 不同视频源的解码性能 (1080P)

Figure 6-12 Decoding performance of different video sources (1080P)

(2) 实验结果分析

以上为视频解码性能对比，通过两组实验，得到以下几点结论：

- ①关键帧解码时间大于非关键帧，且两者时间均在一定的范围内，合理的设置关键帧的间隔将降低视频整体解码时间，但是会造成视频压缩比变化；
- ②同一视频源不同分辨率的解码时间随着分辨率的增加而增加；分辨率越大，视频信息量越大；
- ③同一视频源不同分辨率下的视频解码时间趋势走向一致；不同视频源同一分辨率的视频解码时间走势则有所不同，说明视频编解码与视频内容相关；
- ④同一视频源不同分辨率下或者不同视频源的同一分辨率，开启并行解码，随着并行线程的增多，其解码时间不是线性下降的，因此需要根据视频内容和分辨率选取合适的解码线程参数；

6.4.2 视频增量跳帧算法对比实验

(1) 实验过程描述

视频跳帧算法的不同会影响视频处理智能的整体效果，包括处理器的资源利用率，视频帧处理的冗余性，为了探究不同视频跳帧策略的影响和效果，在不同数据集下，本文选用了第 4 章列举的算法进行对比实验：对同一视频数据下不同跳帧策略的帧使用率、解码过程系统占用率，视频处息熵进行分析比较。其中帧使用率 A_{UP} 表明视频帧处理使用的占比，其定义如下：

$$A_{UP} = (Used_frames / Total_frames) * 100\% \quad (6-1)$$

其中 $Used_frames$ 表示处理的视频帧数， $Total_frames$ 表示视频的总帧数；视频信息熵（Video Information Entropy），表示处理视频帧包含的有效信息，视频帧具有高帧率，连续性的特点，并且每一帧包含的信息也是有限的，不需要每一帧都处理，为了合理的选取视频帧进行处理，本文定义是视频处理信息熵验证跳帧算法的正确性。其定义如下：

$$VIE = \sum_i^n \tanh(d_i) \quad (\text{st. } d_i \text{ 是处理帧包含的目标数}) \quad (6-2)$$

系统占用率使用 LINUX 系统命令获取，通过命令“top -d 刷新间隔 | grep 源程序”得到对应进程的 CPU、MEM 等占用率。

本节实验选用视频 5-1-960P，提前标好视频帧的目标数量，再使用不同的跳帧策略进行算法验证，这里仅部署了跳帧策略，视频智能处理部分未加入，数据解码采用内部 4 路并行线程，下图展示了三类跳帧算法的 CPU 占用、视频帧使用率以及视频处理信息熵情况：

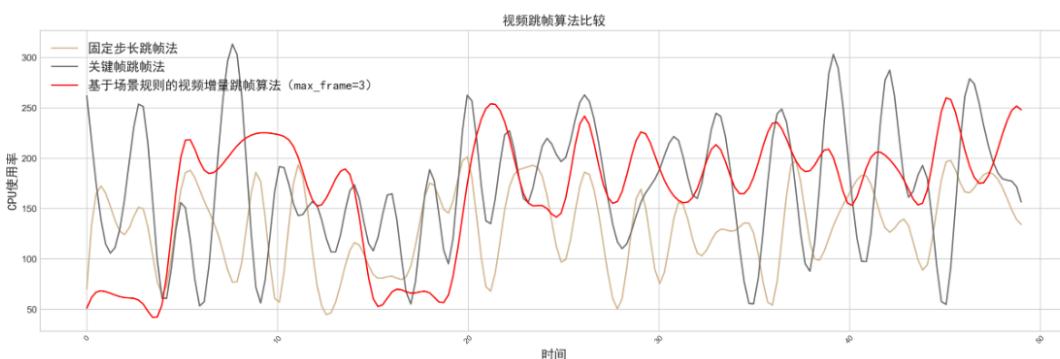


图 6-13 不同视频跳帧算法 CPU 占用率对比

Figure 6-13 Comparison of CPU usage of different video frame skipping algorithms

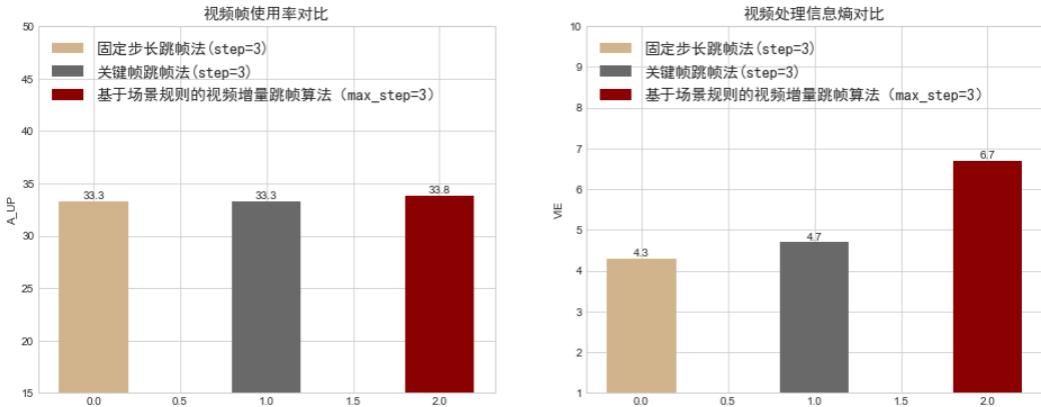


图 6-14 不同视频跳帧算法帧使用率 (A_UP) 和视频处理信息熵 (VIE) 对比

Figure6-14 Comparison of frame usage rate (A_UP) and video processing information entropy (VIE) of different video frame skipping algorithms

(2) 实验结果分析

以上为不同视频跳帧算法的对比实验，通过部署不同的跳帧算法，对系统 CPU 占用率、视频帧使用率、视频处理信息熵等进行比较，得到以下结论：

①基于固定步长和关键帧的 CPU 使用率，由于其视频处理的间隔差一致，因此相对振幅比较小；而基于场景规则的视频增量跳帧跳帧算法，则在视频处理的一段时间表现很低，主要是这段时间的视频目标不存在，因此不存在解码现象。因此，合理的跳帧算法可以使得视频帧处理冗余性降低，就能将系统资源用以别的处理过程；

②基于场景规则的视频增量跳帧算法在视频帧使用率和视频处理信息熵上也有提高，其中视频信息熵是最关注的地方，在视频帧使用率一致的情况下，其视频信息熵得到的有效提高在，证明本文提出的基于场景规则的视频增量跳帧算法是有效、可行的。

6.4.3 图像数据获取对比实验

(1) 实验过程描述

基于图像的智能分析，其读入图像格式如第 2 章所述，一般为 rgb 图像数据。本文提出的并发流水线设计将智能处理图像数据获取划分为两个步骤：yuv 尺寸缩放；yuv 转码 rgb。本节实验主要是探究 FFmpeg、opencv、libyuv 数据处理的性能分析，其中 libyuv 是 Google 开源实现各种 yuv 转码、缩放、旋转等操作的库，支持跨平台编译运行，支持 SSE、AVX、NEON 等 SIMD 指令加速。实验中，视频解码部分采用 FFmpeg 处理，解码后得到的 yuv 数据则选用三类库进行缩放转码的时间效率分析。编写了三个处理函数以及对应的调用脚本：ffmpeg_test；

`opencv_test; libyuv_test` 实现 $w * h$ 的 yuv 数据得到 $h/2 * w/2$ 的 rgb 图像数据，其中处理时间是 yuv data 到 rgb data 区间的时间。实验分为四组，如下所述：

①选择的视频数据为 1-1-1080P，包含 198 帧数据；纵坐标代表 yuv 转换缩放 rgb 的处理时间，表示三类视频处理库的处理效率，横坐标为视频解码线程数：

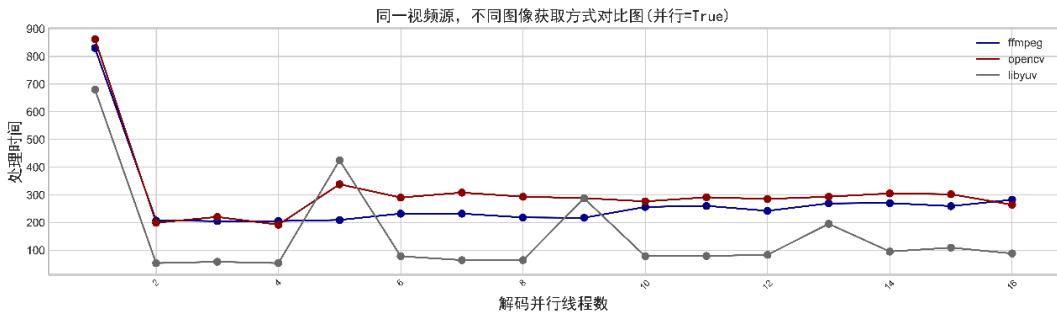


图 6-15 同一视频源，不同视频处理库性能对比图((并行=True)

Figure 6-15 Comparison of performance of different video processing libraries for the same video source(parallel=True)

②选择的视频数据为 1-1-1080P，包含 198 帧数据；纵坐标代表 yuv 转换缩放 rgb 的每帧处理时间，表明三类视频处理库的处理效率，同时控制不同的并行线程数，得到不同的结果：

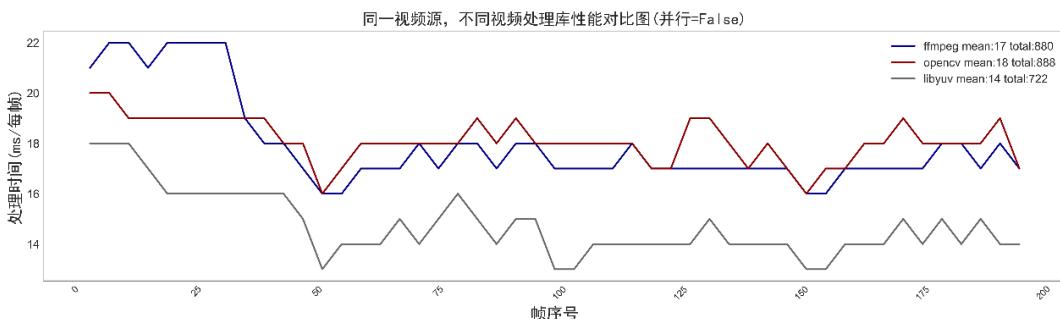


图 6-16 同一视频源，不同视频处理库性能对比图(并行=False)

Figure 6-16 Comparison of performance of different video processing libraries for the same video source (parallel=False)

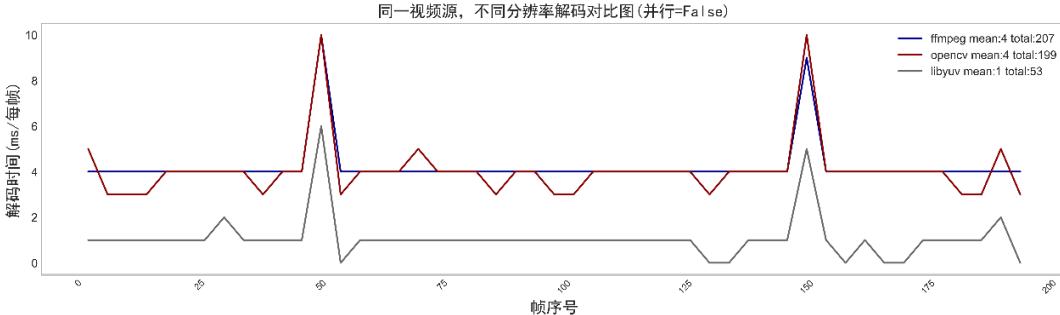


图 6-17 同一视频源，不同视频处理库性能对比图(并行线程=2)

Figure 6-17 Comparison of performance of different video processing libraries for the same video source (parallel thread = 2)

③选择的视频数据为5个视频的1080P，每个视频包含198帧数据；纵坐标代表yuv转换缩放rgb的处理时间，横坐标为解码并行线程数

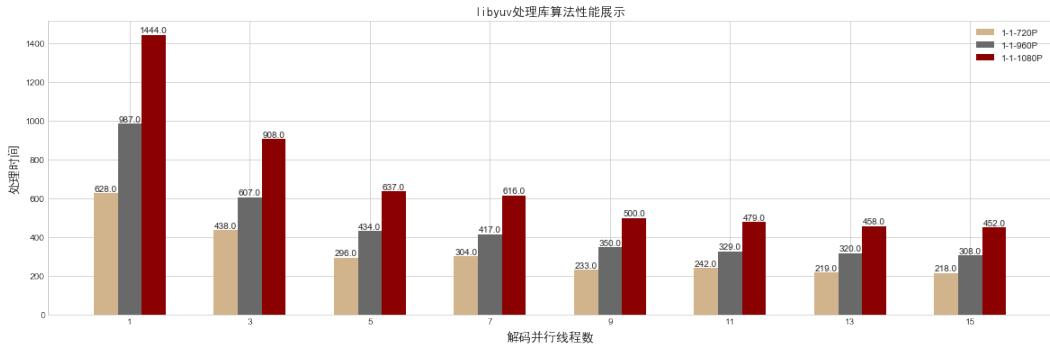


图 6-18 libyuv 处理库算法性能分析

Figure 6-18 libyuv processing library algorithm performance analysis

(2) 实验结果分析

根据以上实验结果，得到以下实验结论：

①在同一视频源上，三类视频处理库，在视频解码单线程的情况下，性能差别不大，但是最好的是 libyuv；控制视频并行解码的参数，其效果最好的是 libyuv，结合视频解码的实验，可以发现视频并行解码会加速视频解码数据的生成，并行的加速了后续处理过程；

③同一视频源，随着分辨率的递增，其 libyuv 的处理时间上升；

④基于以上实验结果，多路视频处理可以选用 libyuv 库；

6.4.4 系统整体效果

系统支持智能处理的参数选择功能，通过左边菜单栏的配置，显示对应的结果，这里主要展示单路视频选择和多路视频(≥ 4)以上的系统效果。

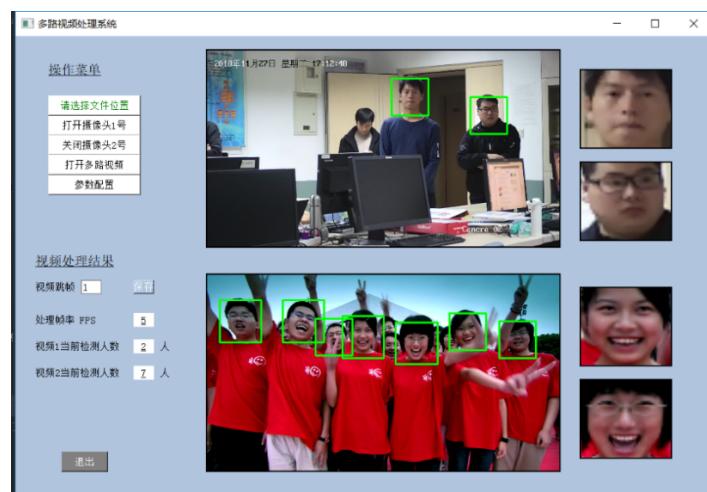


图 6-19 系统整体效果展示 1

Figure 6-19 System overall effect display 1

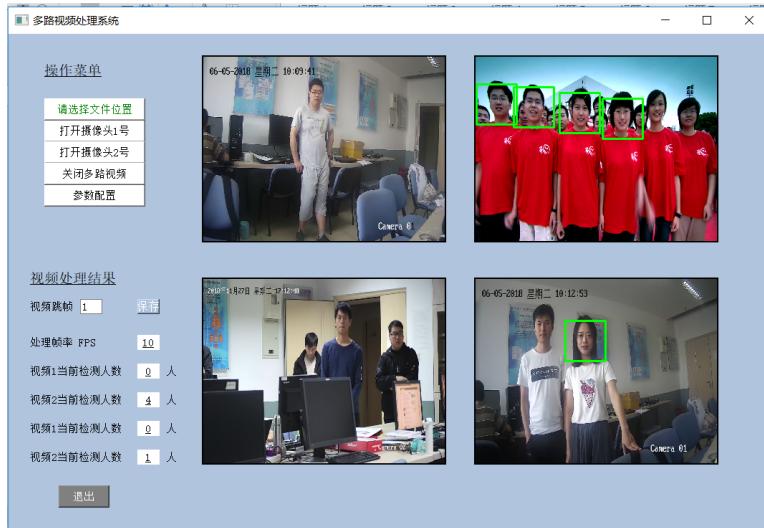


图 6-20 系统整体效果展示 2

Figure 6-20 System overall effect display 2

其中，图 6-19 是同时打开两路视频的效果，图 6-20 是同时打开四路视频的效果，通过对视频跳帧参数的控制，进一步控制视频帧的过滤。同时显示对应的处理性能和实际效果。

6.5 本章小结

本章针对本文研究课题的算法模型和系统总计设计，围绕系统的关键模块实现以及算法验证实验展开讨论。首先，介绍了视频处理系统的软硬件开发环境以及相关的实验数据准备；提出系统的关键模块设计与实现；对第 4 章涉及到的算法模块进行相关实验的验证和结果分析，包括视频解码性能分析、不同视频跳帧算法对比、图像数据获取的对比实验等。从不同的角度去分析和验证系统涉及到的各类型任务设计，给出了系统的整体效果演示，对实验结果进行剖析，分析其不足之处，对以后潜在的工作做了探讨和展望。最终的实验结果证明了前序章节算法设计的正确性和有效性。

结 论

随着物联网、大数据、计算机视觉等技术的高速发展，促进了一系列的人工智能应用领域发展。其中，智慧城市应运而生，其应用领域里的智慧安防、智能视频监控成为了研究重点。智能视频监控主要是为了解决视频大数据下人为监督、分析，其耗费时间长，成本高，且效果一般的问题。结合目前流行的各类视频智能处理技术实现对视频内容的目标检测、分类、识别，进行警情预测等。经过调研和分析，随着视频大数据的兴起，视频存储、视频传输、云端处理压力的增大，云计算技术增长比例远小于数据增长的速度。基于边缘计算的视频智能处理技术被提出，旨在将视频处理的全部或者部分过程分布式的部署至就近的计算结点，降低云端计算压力。但是边缘终端的计算能力和存储能力有限，合理的选择轻量级视频处理算法以及结合相应的系统调度管理算法，将合理的利用系统资源、加速视频处理、达到视频处理信息最大化的目标。本文对于视频智能监控领域的相关发展有一定的研究价值和意义。

本文总结了高清视频监控、视频智能处理、调度管理算法的应用场景及国内外现状，同时基于高清视频前端处理、边缘计算、深度学习加速等相关技术，展开本文研究课题多场景视频智能处理系统及调度管理的相关算法研究。论文的主要研究结果如下：

(1) 分析了目前广泛流行的云端视频智能处理架构的优缺点：其一是云端计算能力与数据增长速度不匹配的问题，造成云端处理计算压力巨大，且视频传输压力大；其二是其分布式的处理架构启发了本文将视频处理的过程分布式的部署在边缘计算节点，平衡计算压力；进一步提出基于边缘计算的视频智能处理系统架构。

(2) 分析对比了当前多路视频处理系统的算法架构和优缺点。根据多路视频处理的重复性和时序性，提出并发流水线的算法架构。结合多场景下的视频智能处理反馈，提出基于场景规则的视频调度管理算法——MSSM，包括任务槽管理单元设计、数据共享设计、视频存储、智能跳帧等相关算法设计，使其兼顾视频处理的冗余性、视频信息最大化、均衡系统资源、加速视频处理、多场景下的扩展性等目标。

(3) 针对边缘终端的计算、存储能力有限的问题，结合单帧人数检测这一应用场景，提出优化的基于轻量级网络的视频智能处理算法。通过对其特征提取网络、检测网络的改进以及数据的扩充，对比目前流行的大型深度模型，其在精度和效率上进行了权衡。提供了调度管理算法的数据基础，为深度学习算法模型

在智能终端部署提供一个轻量级的模型选择方案。

(4) 提出了面向不同应用场景的多路视频智能处理系统解决方案，包括其系统架构设计、流程设计、关键模块设计、前端设计等。同时对系统的各个算法设计进行了实验验证和结果分析。

受限于实验对象和环境的展示，本文的研究工作实际上存在着一些不足，可作为下一步的工作方向：

(1) 本文研究并没有涉及到硬件加速技术，目前针对视频处理，涌现出很多的硬件加速技术，比如利用 FPGA 的并行计算能力进行加速。可以考虑软硬件结合的方式，更好的实现多路视频加速处理。

(2) 本文提出的基于场景规则的调度管理算法，针对不同的应用场景具有扩展性，但仍然存在设计上的不足和遗漏。同时，本文对算法进行了分模快的实验验证，虽然在不同的数据集下得到了一定的结论，由于实验环境的单一性，其结论仍然需要进一步考察。后期的研究工作可以扩展至更复杂的应用场景，验证算法的正确性和鲁棒性。

(3) 针对本文轻量级算法模型研究：其一是在数据集的选取上面，不同场景下的人群特征有一定的倾向性，比如公交车上的人群，身体存在遮挡，但是头部信息相对会比较完整；通道口的人群，不同的通道，人群密度和移动的方式不同，单人通道的人特征很全，很适合做人数统计以及识别相关。其二是在模型压缩方面做更多的研究工作，比如在模型结构、模型参数精度变换、分布式并行计算等方向展开研究。同时针对不同场景的人数统计应用，提出不同的模型结构，或者提取一定的先验知识，加入模型训练。算法够能够自适应不同场景下的处理目标，是未来的一个研究方向。

参 考 文 献

- [1] Nam T, Pardo T A. Conceptualizing smart city with dimensions of technology, people, and institutions[C]// International Digital Government Research Conference: Digital Government Innovation in Challenging Times. 2011.
- [2] Neirotti P, Marco A D, Cagliano A C, et al. Current trends in Smart City initiatives: Some stylised facts[J]. Cities, 2014, 38(5):25-36.
- [3] Su K , Li J , Fu H . Smart city and the applications[C]// International Conference on Electronics. IEEE, 2011.
- [4] Foroughi H, Aski B S, Pourreza H. Intelligent video surveillance for monitoring fall detection of elderly in home environments[C]//2008 11th international conference on computer and information technology. IEEE, 2008: 219-224.
- [5] Morozov AA . Development of a method for intelligent video monitoring of abnormal behavior of people based on parallel object-oriented logic programming[J]. Pattern Recognition & Image Analysis, 2015, 25(3):481-492.
- [6] Feng J, Yang L, Bian H, et al. Design of high-reliability and HD video remote surveillance system[C]//Tenth International Conference on Digital Image Processing (ICDIP 2018). International Society for Optics and Photonics, 2018, 10806: 1080647.
- [7] 黄凯奇, 陈晓棠, 康运锋, et al. 智能视频监控技术综述[J]. 计算机学报, 2015, 20(6):1093-1118.
- [8] Xu Z, Mei L, Hu C, et al. The big data analytics and applications of the surveillance system using video structured description technology[J]. Cluster Computing, 2016, 19(3): 1283-1292.
- [9] Zhu X , Loy C C , Gong S . Learning from Multiple Sources for Video Summarisation[J]. International Journal of Computer Vision, 2016, 117(3):247-268.
- [10] Yang Y , Yao X , Gao Y , et al. Parallel Multi-source Video Processing Based on Software Pipeline[C]// 3rd International Conference on Mechatronics, Robotics and Automation. 2015.
- [11] Zhu X , Loy C C , Gong S . Learning from Multiple Sources for Video Summarisation[J]. International Journal of Computer Vision, 2016, 117(3):247-268.
- [12] Shi W, Jie C, Quan Z, et al. Edge Computing: Vision and Challenges[J]. IEEE Internet of Things Journal, 2016, 3(5):637-646.
- [13] Lopez P G, Montresor A, Epema D, et al. Edge-centric Computing:Vision and Challenges[J]. Acm Sigcomm Computer Communication Review, 2015, 45(5):37-42.
- [14] Karasulu B, Korukoglu S. Moving object detection and tracking by using annealed background subtraction method in videos: Performance optimization[J]. Expert Systems with Applications, 2012, 39(1): 33-43.
- [15] Robin C , Lacroix S . Multi-robot target detection and tracking: taxonomy and survey[J]. Autonomous Robots, 2016, 40(4):729-760.

- [16] 石红波, 黄山, 张洪斌, et al. 基于综合背景提取方法与阴影抑制的智能监控系统[J]. 计算机工程与科学, 2012, 34(12).
- [17] Yilmaz A, Javed O, Shah M. Object tracking: A survey[J]. Acm computing surveys (CSUR), 2006, 38(4): 13.
- [18] Vishwakarma S, Agrawal A. A survey on activity recognition and behavior understanding in video surveillance[J]. The Visual Computer, 2013, 29(10): 983-1009.
- [19] Simonyan K, Zisserman A. Two-stream convolutional networks for action recognition in videos[C]//Advances in neural information processing systems. 2014: 568-576.
- [20] Tran D, Bourdev L, Fergus R, et al. Learning spatiotemporal features with 3d convolutional networks[C]//Proceedings of the IEEE international conference on computer vision. 2015: 4489-4497.
- [21] De Souza C R, Gaidon A, Vig E, et al. Sympathy for the details: Dense trajectories and hybrid classification architectures for action recognition[C]//European Conference on Computer Vision. Springer, Cham, 2016: 697-716.
- [22] Bolosky W J, Fitzgerald R P, Douceur J R. Distributed schedule management in the Tiger video fileserver[J]. Acm Sigops Operating Systems Review, 1997, 31(5):212-223.
- [23] Teifel J, Manohar R. Static tokens: Using dataflow to automate concurrent pipeline synthesis[C]//10th International Symposium on Asynchronous Circuits and Systems, 2004. Proceedings. IEEE, 2004: 17-27.
- [24] Robinovitch S N, Feldman F, Yang Y, et al. Video capture of the circumstances of falls in elderly people residing in long-term care: an observational study[J]. Lancet, 2013, 381(9860):47-54.
- [25] Simpson F F, Herendeen J, Mousseau R P, et al. System and method for performance data collection in a virtual environment: U.S. Patent 8,239,526[P]. 2012-8-7.
- [26] Wu C Y, Singhal N, Krahenbuhl P. Video compression through image interpolation[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 416-431.
- [27] Chen Y, Wen Z, Wen J, et al. Efficient software H. 264/AVC to HEVC transcoding on distributed multicore processors[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2015, 25(8): 1423-1434.
- [28] Shen T, Lu Y, Wen Z, et al. Ultra fast H. 264/AVC to HEVC transcoder[C]//2013 Data Compression Conference. IEEE, 2013: 241-250.
- [29] 蔡晓霞, 崔岩松, 邓中亮,等. 下一代视频编码标准关键技术[J]. 电视技术, 2012, 36(2):80-84.
- [30] Wu C Y, Singhal N, Krahenbuhl P. Video compression through image interpolation[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 416-431.
- [31] 段军棋, 蒋丹. 远程视频监控系统的设计与实现[J]. 电子科技大学学报, 2002, 31(5):523-528.

- [32] 张南平, 杨照芳, 夏红霞. IPSAN 基于 IP 的存储区域网络[J]. 计算机技术与发展, 2003, 13(1):18-20.
- [33] Witten I H , Frank E . Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)[M]. 机械工业出版社, 2005.
- [34] LeCun Y, Bengio Y, Hinton G. Deep learning[J]. nature, 2015, 521(7553): 436.
- [35] Nussbaumer H J. Fast Fourier transform and convolution algorithms[M]. Springer Science & Business Media, 2012.
- [36] Gong Y, Wang L, Guo R, et al. Multi-scale orderless pooling of deep convolutional activation features[C]//European conference on computer vision. Springer, Cham, 2014: 392-407.
- [37] Gu S, Lillicrap T, Sutskever I, et al. Continuous deep q-learning with model-based acceleration[C]//International Conference on Machine Learning. 2016: 2829-2838.
- [38] Iandola F N, Moskewicz M W, Ashraf K, et al. Firecaffe: near-linear acceleration of deep neural network training on compute clusters[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 2592-2600.
- [39] Han S, Liu X, Mao H, et al. EIE: efficient inference engine on compressed deep neural network[C]//2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). IEEE, 2016: 243-254.
- [40] Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856..
- [41] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826.
- [42] Hashemzadeh M, Farajzadeh N. Combining keypoint-based and segment-based features for counting people in crowded scenes[J]. Information Sciences, 2016, 345: 199-216.
- [43] 常庆龙, 夏洪山, 黎宁. 一种基于归一化前景和角点信息的复杂场景人数统计方法[J]. 电子与信息学报, 2014(2).
- [44] Dalal N, Triggs B. Histograms of oriented gradients for human detection[C]//international Conference on computer vision & Pattern Recognition (CVPR'05). IEEE Computer Society, 2005, 1: 886--893.
- [45] Viola P, Jones M. Rapid object detection using a boosted cascade of simple features[J]. CVPR (1), 2001, 1: 511-518.
- [46] 高飞, 丰敏强, 汪敏倩, et al. 基于热点区域定义的人数统计方法研究[J]. 计算机科学, 2017(S1):183-188+211.
- [47] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.
- [48] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European

- conference on computer vision. Springer, Cham, 2016: 21-37.
- [49] Klambauer G, Unterthiner T, Mayr A, et al. Self-normalizing neural networks[C]//Advances in neural information processing systems. 2017: 971-980.
- [50] Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.
- [51] Ahmed F, Tarlow D, Batra D. Optimizing expected intersection-over-union with candidate-constrained CRFs[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1850-1858.
- [52] Chai T, Draxler R R. Root mean square error (RMSE) or mean absolute error (MAE)?[J]. Geoscientific Model Development Discussions, 2014, 7: 1525-1534.
- [53] Vicente S, Carreira J, Agapito L, et al. Reconstructing pascal voc[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 41-48.
- [54] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.
- [55] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
- [56] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
- [57] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
- [58] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700-4708.

攻读硕士学位期间取得的研究成果

论文成果：

- [1] Xiaoqin Feng, Rong Xie, Junyang Sheng, Shuo Zhang. Population Statistics Algorithm Based on MobileNet[C]//2019 4th International Conference on Intelligent Computing and Signal Processing(ICICSP 2019)(已录用， EI 收录源)
- [2] Rong Xie, Xiaoqin Feng. A method of quick edge detection based on Zynq[C]//2018 International Conference on Modeling, Algorithm and Artificial Intelligence (MAAI 2018)(已录用， EI 收录源)
- [3] Junyang Sheng, Xiaoqin Feng. Research on the Internet of Things Platform for Smart and Environmental Protection[C]//2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems(CCIS 2018)(已录用， EI 收录源)

参与的科研项目：

- [1] 国家发改委项目:基于 IPv6 移动物联网的交通车辆管控终端研制及应用示范.参与起止时间 2017 年 7 月-2018 年 7 月. 主要负责公交视频监控系统的方案设计、部分算法研究；终端传感器采集、平台通信程序等开发工作。

致 谢

时光荏苒，转瞬间，研究生的三年学习之旅即将落下帷幕，临近毕业，回忆在此三年的点点滴滴，不免感慨万千。感谢这三年以来，我的老师、同学和家人，正是因为你们的不离不弃，我才能克服重重困难，不断进步，临近毕业，致以我最真挚的祝福和感谢。

感谢我的导师黄樟钦教授在学业和生活上给与的支持和帮助。刚上研一的时候，自己的理论和工程能力都很欠缺，是老师安排了一系列的科研训练，亲自指导，为我后面进入实际项目打下了基础；之后进入实际项目，老师亲自指导工程设计、代码测试、文档编撰、实地考察等，和本文一起工作在项目的最前线，我很感动，同时也收益匪浅。本文从开题开始一直到论文定稿，老师准备了各类学术论文讨论会、中期检查会等，对本文的论文都给予了详细的指导。因此，我想借此机会向黄老师表达深深的祝福和感谢。

感谢张硕学长、谢蓉、邓旺华、秦泽鹏对我论文工作的指导和帮助，感谢你们提出的中肯建议使我能够顺利完成论文，也感谢你们的鼓励支持。与此同时，感谢我的朋友陈梦瑶、李倩、李洋、盛俊阳、杨恩源等对我的关心和帮助，这三年是你们与我一起前进，倾心相助，感谢让我遇到你们，让我的研究生生活变得多姿多彩。感谢多年来家人的陪伴和理解，即使身在异乡，依然对我无微不至。感谢人生路上每一个遇到的人，仅在此表达自己最真挚的感谢与祝福。