**中图分类号：TP37**

**论文编号：10006SY1306213**



硕士学位论文

面向MOOC的移动视频学习平台的设计与实现

作者姓名 王 翊

学科专业 计算机软件与理论

指导教师 吴文峻教授

培养院系 计算机学院

**The Design and Implementation of a Video-based Mobile Learning Platform Matching on MOOC**

A Dissertation Submitted for the Degree of Master

**Candidate: Wang Yi**

**Supervisor: Prof. Wu Wenjun**

School of Computer Science and Engineering

Beihang University, Beijing, China

**中图分类号：TP37**

**论文编号：10006SY1306213**

硕士学位论文

面向MOOC的移动视频学习平台

的设计与实现

作者姓名 王 翊 申请学位级别 工学硕士

指导教师姓名 吴 文 峻 职 称 教 授

学科专业 计算机软件与理论 研究方向 多媒体

学习时间 自 2013 年 9 月 2 日 起 至 2015 年 12 月 日 止

论文提交日期 2015 年 12 月 日 论文答辩日期 2015 年 12 月 日

学位授予单位 北京航空航天大学 学位授予日期 2016 年 1 月 8 日

关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在导师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或证书等而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中做出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名 日期 年 月 日

关于学位论文使用授权的声明

本人完全同意北京航空航天大学有权使用本学位论文(包括但不限于其印刷版和电子版)，使用方式包括但不限于：保留学位论文，按规定向国家有关部门(机构)送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名 日期 年 月 日

指导教师签名 日期 年 月 日

摘 要

大规模开放式在线课程（MOOC）是一种当今非常流行的授课方式。通过教师将课程视频及教学材料上传到网络，学生可以学习课程、完成作业、参与课程讨论。这种教学方式灵活性强，自主性高，受到广大师生的好评。这种授课方式决定了用户应该可以在任何时间学习网络上的MOOC课程。如果MOOC平台仅仅在Web端发展，则用户仍然只能在PC端进行学习；为了满足用户随时随地学习课程的需求，本文设计了一种面向MOOC的移动视频学习平台，为用户学习课程提供很大的便利。

传统的视频传输协议具有缓冲时间长、清晰度固定、自适应能力差等缺点；一种基于HTTP的动态自适应流（DASH）协议可以有效减少视频的缓冲时间、增强用户体验。DASH技术先将视频切分为小分段，然后将视频分段转码为不同码率的副本；通过客户端的带宽选择算法，客户端可以自行选择合适的码率并下载对应的视频。

虽然这种DASH协议具备很多优点，但是它并没有针对MOOC视频进行优化；鉴于此，本文尝试对DASH协议进行实现，并对DASH系统的几个关键技术进行重新研究，使之可以提供更优质、质量更稳定的MOOC视频。

本文的主要研究内容为以下两个方面。

首先，研究一种适用于MOOC的视频切分算法。相比于传统视频切分算法，本算法考虑到了视频的内容与质量的关系，并有针对性的对其进行转码，在保证视频质量的前提下降低了每个视频分段的大小，提升了带宽利用率，提高了用户体验。

其次，研究一种带宽适配算法，根据当前下载速率的大小判断可用带宽，并自动适配最合适的码率，根据该码率下载视频分段，可以有效地提高视频播放器对带宽波动的适应性。这种带宽适配算法对提升用户体验有一定的帮助。提升DASH视频流的稳定性以及在带宽不稳定情况下的视频质量。

**关键词**：MOOC，DASH，视频分类，带宽适配

**Abstract**

Nowadays, Massive Open Online Course (MOOC) has becoming a very popular teaching method. Students who participate in MOOC can learning courses, do homework, take part in discussion through MOOC platform. Because of its flexible, independent characteristics, this kind of teaching method is highly recommended by most teachers and students. If MOOC platform is only being developed in Web terminal, these characteristics will be restricted. In order to avoid that, i.e. to meet the needs of learning everywhere, this paper tries to design an video-based mobile learning platform.

Traditional video stream transmission methods has many disadvantages, including long caching time, fixed resolution and inadaptability. In this paper, we using Dynamic Adaptive Streaming over HTTP (DASH) to transmit MOOC video, which could reduce buffer time, thus enhance the user experience. DASH cuts MOOC videos into small segments at first, then transcodes segment into several copies of different rate. The bandwidth adaptation algorithm makes client be able to select proper video bitrate.

Although this DASH system has many advantages, it is not optimized for MOOC videos. This paper attempts to optimize the DASH system in several algorithms so that it can provide better and more stable MOOC stream.

This paper is divided into two parts.

Firstly, this paper tries to develop a video classification algorithm based on MOOC. Compared to traditional algorithm, this new algorithm takes the relationship between the video content and quality into consideration and reduces the size of each video segment in guarantee of video quality, thus improves the bandwidth utilization and user experience.

Secondly, this paper proposes an improved bandwidth adaptation algorithm. It can determine available bandwidth according to current download rate and automatically adapt the most appropriate rate, which can remarkable enhance the user experience.

**Keywords:** MOOC, DASH, Video Classification, Bandwidth Adaptation

目 录

[第一章 绪论 1](#_Toc438988576)

[1.1 研究背景及意义 1](#_Toc438988577)

[1.2 国内外研究现状 2](#_Toc438988578)

[1.2.1 DASH技术 2](#_Toc438988579)

[1.2.2 视频评价 2](#_Toc438988580)

[1.2.3 视频切分算法 3](#_Toc438988581)

[1.2.4 码率调节算法 4](#_Toc438988582)

[1.3 研究目标与内容 5](#_Toc438988583)

[1.4 论文组织 5](#_Toc438988584)

[第二章 视频切分算法设计 7](#_Toc438988585)

[2.1 视频切分算法综述 7](#_Toc438988586)

[2.2 MOOC视频分段的类别 7](#_Toc438988587)

[2.2.1 讲师分段 8](#_Toc438988588)

[2.2.2 PPT分段 9](#_Toc438988589)

[2.2.3 混合分段 9](#_Toc438988590)

[2.2.4 视频资料分段 9](#_Toc438988591)

[2.2.5 其他分段 9](#_Toc438988592)

[2.3 视频分段算法 9](#_Toc438988593)

[2.4 视频分类算法 12](#_Toc438988594)

[2.4.1 运动区域计算 13](#_Toc438988595)

[2.4.2 人脸识别 15](#_Toc438988596)

[2.4.3 相似度计算 17](#_Toc438988597)

[2.5 码率划分策略 18](#_Toc438988598)

[2.5.1 用户主观感受的客观测量方法 18](#_Toc438988599)

[2.5.2 码率划分方案 20](#_Toc438988600)

[2.6 小结 23](#_Toc438988601)

[第三章 码率调节算法设计 24](#_Toc438988602)

[3.1 码率调节算法综述 24](#_Toc438988603)

[3.2 码率调节原则 24](#_Toc438988604)

[3.1.1 防止视频缓存过多 24](#_Toc438988605)

[3.1.2 防止视频缓存过少 25](#_Toc438988606)

[3.1.3 平滑调节带宽 25](#_Toc438988607)

[3.1.4 内容对QoE的影响 26](#_Toc438988608)

[3.3 码率调节算法 26](#_Toc438988609)

[3.2.1 带宽测量方法 26](#_Toc438988610)

[3.2.2 下载速率数据优化 27](#_Toc438988611)

[3.2.3 码率调节方法 29](#_Toc438988612)

[3.4 小结 29](#_Toc438988613)

[第四章 系统设计与实现 31](#_Toc438988614)

[4.1 系统总体设计 31](#_Toc438988615)

[4.2 平台服务器设计 31](#_Toc438988616)

[4.2.1 视频接收模块 32](#_Toc438988617)

[4.2.2 视频划分模块 32](#_Toc438988618)

[4.2.3 视频分类模块 33](#_Toc438988619)

[4.2.4 视频转码、存储模块 36](#_Toc438988620)

[4.2.5 代理服务器模块 37](#_Toc438988621)

[4.3 平台客户端设计 41](#_Toc438988622)

[4.3.1 客户端主要结构及主体界面 41](#_Toc438988623)

[4.3.2 数据收发模块设计方案 42](#_Toc438988624)

[4.3.3 欢迎界面设计 43](#_Toc438988625)

[4.3.4 登录界面设计及模块实现 45](#_Toc438988626)

[4.3.5 主界面设计 46](#_Toc438988627)

[4.3.6 全部课程浏览模块设计 46](#_Toc438988628)

[4.3.7 课程讨论模块设计 47](#_Toc438988629)

[4.3.8 分割视图界面设计 48](#_Toc438988630)

[4.3.9 已选课程浏览模块设计 49](#_Toc438988631)

[4.3.10 视频观看界面设计 49](#_Toc438988632)

[4.4 平台视频播放器设计 49](#_Toc438988633)

[4.4.1 视频地址解析 50](#_Toc438988634)

[4.4.2 视频解码 50](#_Toc438988635)

[4.4.3 音频解码 52](#_Toc438988636)

[4.4.4 时间轴控制及音视频同步 53](#_Toc438988637)

[4.5 小结 53](#_Toc438988638)

[第五章 实验与数据分析 55](#_Toc438988639)

[5.1 视频切分算法准确度 55](#_Toc438988640)

[5.2 码率调节算法实验 57](#_Toc438988641)

[5.3 在线学习平台用户体验实验 59](#_Toc438988642)

[5.4 小结 59](#_Toc438988643)

[总结与展望 62](#_Toc438988644)

[一、总结 62](#_Toc438988645)

[二、展望 62](#_Toc438988646)

[参考文献 64](#_Toc438988647)

[攻读硕士学位期间取得的学术成果 67](#_Toc438988648)

[致 谢 68](#_Toc438988649)

图 目

[图1 视频分类模块的决策树 8](#_Toc438988650)

[图2 四种不同视频分段的截图示例 8](#_Toc438988651)

[图3 计算MOOC视频颜色分布的耗时和Pearson相关系数 11](#_Toc438988652)

[图4 某段视频的颜色分布差值图（n=5时情形） 11](#_Toc438988653)

[图5 图4中帧编号在2400～3000之间的颜色分布差值图 12](#_Toc438988654)

[图6 运动区域的计算过程 14](#_Toc438988655)

[图7 不同视频分段的典型运动区域图 15](#_Toc438988656)

[图8 仅采用面部识别时的错误案例 16](#_Toc438988657)

[图9 双重人脸识别的例子 16](#_Toc438988658)

[图10 讲师位置的分析 17](#_Toc438988659)

[图11 对一系列PPT分段及视频资料分段的SSIM值的比较 19](#_Toc438988660)

[图12 PPT分段码率与视频质量关系分析图 21](#_Toc438988661)

[图13 讲师分段码率与视频质量关系分析图 22](#_Toc438988662)

[图14 视频资料分段码率与视频质量关系分析图 23](#_Toc438988663)

[图15 移动学习平台系统总体结构 31](#_Toc438988664)

[图16 视频接收模块流程图 32](#_Toc438988665)

[图17 视频划分模块流程图 33](#_Toc438988666)

[图18 运动区域计算模块流程图 35](#_Toc438988667)

[图19 人脸识别模块流程图 36](#_Toc438988668)

[图20 相似度计算模块流程图 36](#_Toc438988669)

[图21 视频转码存储模块流程图 37](#_Toc438988670)

[图22 原始数据路径及代理服务器的生效过程 39](#_Toc438988671)

[图23 在线学习平台客户端结构及关系图 42](#_Toc438988672)

[图24 在线学习平台客户端欢迎界面 44](#_Toc438988673)

[图25 在线学习平台客户端登录界面 45](#_Toc438988674)

[图26 在线学习平台客户端界面 47](#_Toc438988675)

[图27 全部课程信息模块结构及关系 47](#_Toc438988676)

[图28 课程讨论模块结构及关系图 48](#_Toc438988677)

[图29 在线学习平台客户端播放器界面 50](#_Toc438988678)

[图30 视频/音频解码流程图 51](#_Toc438988679)

[图31 环形缓冲区示例 52](#_Toc438988680)

[图32 码率调节算法实验结果一 58](#_Toc438988681)

[图33 码率调节算法实验结果二 58](#_Toc438988682)

[图34 250kbps实验：码率选择图 60](#_Toc438988683)

[图35 250kbps实验：客观质量图 60](#_Toc438988684)

[图36 250kbps实验：视频截图 60](#_Toc438988685)

[图37 3000kbps实验：码率选择图 61](#_Toc438988686)

[图38 3000kbps实验：客观质量图 61](#_Toc438988687)

[图39 3000kbps实验：视频截图 61](#_Toc438988688)

表 目

[表1 PSNR至MOS值的粗略转换 20](#_Toc438988689)

[表2 测试用PPT分段各类参数 20](#_Toc438988690)

[表3 测试用讲师分段各类参数 21](#_Toc438988691)

[表4 测试用视频资料分段各类参数 22](#_Toc438988692)

[表5 码率设置 23](#_Toc438988693)

[表6 视频分段的清晰度对应码率 25](#_Toc438988694)

[表7 受试者对不同视频质量的测试视频的打分情况 26](#_Toc438988695)

[表8 带宽选择算法所用变量含义 28](#_Toc438988696)

[表9 代理服务器数据库的定义 40](#_Toc438988697)

[表10 数据库MOOCDatabaseUser的定义 43](#_Toc438988698)

[表11 数据库MOOCDatabaseCourse的定义 44](#_Toc438988699)

[表12 登录请求的变量设置 45](#_Toc438988700)

[表13 分类实验正确识别数统计结果 55](#_Toc438988701)

[表14 分类实验正确识别数分类统计结果 55](#_Toc438988702)

[表15 错误识别的具体样例 56](#_Toc438988703)

第一章 绪论

1.1 研究背景及意义

MOOC（massive open online courses）[1]即大型开放式网络课程，是2000年之后才发展出来的概念，最远可追溯至20世纪60年代。最初的MOOC课程于2007年8月由犹他州州立大学的大卫‧怀利教授开设，来自8个国家的50名学生选修并最终学完了这门课程。近年来，凭借工具资源丰富、课程制作便捷、课程价格合理或免费、面向群体广泛、互动性较强、参与自主性高的特点[2][3]，MOOC这一概念迅速发展，并发展出数十家同时在线万人以上的MOOC平台，其中以Coursera、Udacity、edX三大课程提供商发展而来的三大MOOC平台最为火热。如今，数百家来自世界各地的大学已经逐渐参与到MOOC平台的建设中[4]，未来将会有更多的学校参与进来。

移动终端[5]，主要指可以在移动中使用的微小型设备，以智能手机、平板电脑为这类设备的代表，具有便携、易于扩展、功能强大的特点。这些移动设备主要采用Android、iOS等操作系统。如今，这些操作系统越来越开放，基于开源系统开发的个性化应用软件层出不穷，如游戏软件、办公软件、金融平台等，极大程度地满足了各式用户的需求。这些软件的发展，也反过来推动移动终端飞速发展。这种双重促进，使得移动终端在近几年越来越火热，成为最受欢迎的电子设备。

在移动终端，MOOC平台应主要具备视频观看、课程讨论、全部课程浏览等功能[6]，这些功能所需的数据存储、界面设计、通信等技术均有对应的实现方案。但是，移动端的存储空间及屏幕通常较小，并且移动端的数据来源通常是单位费用较高的蜂窝移动通信网络，设计移动端MOOC平台时需要考虑以上因素并针对其进行优化，以保证用户使用移动终端学习MOOC课程时可以获得与PC终端相近的用户体验质量QoE（Quality of Experience）。本文尝试针对上述移动端的特点，结合MOOC课程及MOOC平台的特点，研发一种QoE较高、消耗带宽小于PC终端的移动端MOOC视频平台。

本文的主要研究内容在于如何将DASH[7](Dynamic Adaptive Streaming over HTTP, HTTP动态自适应流媒体)协议应用到MOOC平台的设计与实现中。DASH技术将视频切分为基于HTTP的小视频分段，并将视频分段转码为多种不同的码率；客户端播放视频时，可以自动选择最适合当前网络状况的码率的视频分段，避免视频的卡顿现象，以提升QoE。这种技术可以根据网络质量自动变更视频码率并无缝衔接不同码率的视频，是第一种基于HTTP的自适应视频流解决方案国际标准。随着时间的推移，DASH的发展越来越迅速，也有更多学者开始研究如何将DASH应用于各种各样的场景。由于DASH是一种协议，基于DASH协议开发的系统大多并不开源，所以本文需要首先实现一种适用于MOOC的DASH系统。

DASH技术要求服务端提前将视频切分为视频分段。现有的DASH标准对切分方法没有精确的定义，仅仅将视频按固定时长切分，并将切分后的视频分段转码为固定的几种码率。这种切分方法不仅浪费带宽，而且无法有效提升QoE。由于视频的内容具有冗余，并且不同的内容具有不同的冗余度，所以根据不同的内容设计不同的转码方案可以有效节约带宽，并且与转码前的视频相比，可以保证QoE基本相同或者不会出现大幅下降。因此，本文需要研究一种对视频的切分、分类、转码算法，以达到上述目的。

DASH技术要求客户端对所处网络的网络质量进行测量，并根据当前情况自动选择最合适的码率进行播放。与之相关的带宽选择算法决定了客户端的稳定性；好的带宽选择算法对带宽变化有适当的灵敏度，它可以保持视频质量不出现较大波动、视频缓存长度保持恒定水平，为用户提供高水平的用户观看体验。本文的另一个研究重点即是研究一种具备上述特点的带宽选择算法，以尽量提高用户使用移动端MOOC平台学习MOOC课程时的QoE。

本课题是基于北京航空航天大学软件开发环境国家重点实验室承接的北京市教委的教改项目，本设计是基于该项目所建北航学堂开发的移动终端。视频的切分、分类、转码算法可以减小每个视频分段所需码率，提高带宽利用效率；码率选择算法可以增强用户观看视频时的质量稳定性；对局域网内数据请求的优化可以大幅提高视频质量。三部分相互结合，相得益彰，可以极大提高QoE，使用户可以更好地学习MOOC课程。

1.2 国内外研究现状

1.2.1 DASH技术

上文已经对DASH技术进行了简单介绍；至今为止，已经有很多学者在HTTP视频流上做了大量研究工作。除了苹果主导研究的HLS（HTTP Live Streaming）协议[8]，MPEG-DASH也是一种流行的HTTP视频流[7]，它们均为DASH技术的一种，工作原理为将视频进行切分后播放，优点为传输迅速、用户体验良好。

1.2.2 视频评价

当DASH播放器播放MOOC视频时，算法有必要对该DASH播放器播放出的视频质量进行衡量，以对算法进行评估及改进。目前，对视频质量进行量化的方法主要有主观评价[9],[10]及客观评价两大类。

对视频的主观评价，主要步骤是选取一批对视频领域无知识基础，并且性别、年龄分布均衡的测试者，令其观看测试视频，并对所有测试视频进行打分，最终对评分取平均值，即可获得对每个视频的平均意见分MOS（Mean Opinion Score）[11]。与客观评价相比，这种评价方法更容易代表人类的真实感受，对质量量化的准确性更高；与之对应的是，这种方法需要组织大量测试者，所以成本较高，评测准确性受用户主观感受影响较大，获取批量数据将耗费较多时间。现有的主要主观评测方法分为双激励损伤度分级法DSIS（Double Stimulus Impairment Scale）、双激励连续质量分级法DSCQS（Double Stimulus Continuous Quality Scale）、单激励法SSM（Single Stimulus Methods）、单激励连续质量分级法SSCQE（Single Stimulus Continuous Quality Evaluation）四种，四种方法的主要区别在于测试视频集的设置、分数的设置等。

对视频的客观评价，主要是采用一系列算法判断视频的质量或者与未压缩视频的差距。这种方法现在有一些成熟实现，运算速度快捷，不需要任何成本。但是由于客观评价算法计算得到的值与质量没有本质关系，需要通过一系列主观评价后使评测数据与视频质量建立关系，所以准确度取决于算法研究者的经验；同时，有时视频质量与计算值的关系并非简单的函数关系，所以对某些视频的质量评估会于实际值有较大差异。现有的比较成熟的评价视频质量的参数有峰值信噪比PSNR（Peak Signal to Noise Ratio）及结构相似性指标SSIM（Structural Similarity Index Measurement）[12]，很多学者也对这两种技术进行了评估及比较[13]。

1.2.3 视频切分算法

视频切分算法为上文所述DASH技术的一种创新，旨在为基于MOOC的视频切分及分类转码提供依据。

视频切分算法主要采用了颜色频率统计、人脸识别、相似度分析等算法，关于这三方面的研究进展已经很多[14][15]。本文尝试在现有研究的基础上，研究一种适合MOOC视频的视频切分算法，并针对这三方面进行改进，从而使该算法的执行效率得到提升。上文提及到的人脸识别技术[16]，是基于人的面部特征信息进行识别的一种技术。该技术通过识别人脸的特征（五官、轮廓等），以提取出人脸的位置、大小等信息。如今的人脸识别技术集成了多种计算机领域，识别精度可以达到相当高的水平。

DASH技术带来了一些会影响视频质量的变量。一些学者[17]认为，初始延迟、卡顿、视频质量变动是影响用户观看基于DASH的视频的QoE的三个重要因素。很多研究者在如何从这三个角度优化QoE做了大量研究[18]。然而，对于基于MOOC的DASH系统，目前的研究甚少。据本文作者了解，针对基于内容的DASH系统的研究只有一篇文章[19]，它描述了如何通过将视频根据运动量分类来设计视频切分算法。这种分类算法是基于非固定类型视频设计，这种设计方案并不适用于MOOC平台。

1.2.4 码率调节算法

码率调节算法为上文所述动态自适应流的一个步骤，主要目的是通过对当前网络环境进行采样，以选择最适合的码率播放给用户。近期的文献[20]将带宽适配算法划分为四个步骤：估计可用网络带宽、估测数据平滑处理、量化带宽以判断合适的带宽并选取合适的视频码率、估计下一段视频的下载时间。

基于DASH的播放器需要为下一段视频选取码率时，首先以及最重要的工作是为码率调节算法提供当前的带宽数据。最近的一篇文章[21]描述了带宽测量的难度：用户所处的网络环境通常是不稳定的，而由于网络的特性，真正的网络状况通常隐藏在应用层之下，很难通过程序去捕捉；作为一种替代方案，算法可以捕获近期的最大带宽数据，但是对过去的测量通常不能代表现在的网络状况，这导致算法很难准确判定下一段视频应使用哪一种带宽，从而造成算法的效果较差。

DASH算法的带宽估计算法可以分为两大类：主动探测方法、被动方法。根据DASH系统的特性不同，研究者选取的探测方法也略有不同。

一篇文章[22]介绍了一种名为QDASH的DASH算法，该技术采用主动探测方法探测带宽；该文作者介绍，通过发送TCP探测包，并计算往返时延RTT（Round Trip Time），可以一定程度上判断网络的状况。这种方法的优点在于，它可以快速探测到带宽的变动，从而使QDASH的动态性更强。但是另一篇文章[23]指出，这种方法对带宽的判定是一种估计，估计带来的偏差可能会导致严重的正反馈，从而导致码率调节算法的偏差，这会导致用户观看视频质量的大幅波动甚至大幅降低，使QoE变差；由于这种探测算法涉及到传输层的参数调节，导致这种测量方法很难植入DASH系统中。

另外一些DASH算法[24]通过被动测量方法估计可用的带宽。当需要下载视频分段时，播放器将记录并通过合适的算法计算平均下载速率，并以该值作为对当前网络带宽的估计值。播放器将保留相当长一段时间的数据，作为判断网络带宽变化趋势的依据。与主动探测方式相比，该方法不涉及复杂的网络技术，易于实现，算法占用资源少，适合使用。

1.3 研究目标与内容

论文主要研究面向MOOC的移动视频学习平台的设计与实现。本文主要采用基于DASH的视频流技术传输并显示视频，并尝试研究如何优化现有的DASH算法以提高带宽的利用率、减少视频质量变化并提升QoE。论文主要从客户端、服务端两大方面入手，解决如下的问题：

1、服务端方面：如何对视频的切分算法进行优化，以使得服务端在最大化视频压缩程度的前提下，提供给用户质量最高的视频。

2、客户端方面：如何选择适当的码率，使得客户端既可以在限定时间内下载完对应码率的视频分段，又可以保证最大化视频分段的质量，同时尽量减少码率的变动，以最大化QoE。

针对以上两方面的问题，本文提供了以下解决方案：

1、服务端方面：利用视频内容的特性，研究出一种新型的视频切分算法。该算法具有按内容切分视频、将视频分段进行分类并按分类转码视频分段的功能，可以在尽可能减少带宽消耗的前提下提升视频质量，以提升QoE。另外，本文需要调查用户视频观看体验与码率的关系，并根据该调查对视频进行码率划分，使划分结果可以覆盖合理的QoE区间。

2、客户端方面：在采用基于缓冲区的视频播放器的同时，尝试研究一种带宽选择算法，该算法可以使视频播放器播放质量更高更稳定、卡顿更少的视频。

围绕上述研究目标，本文的研究内容包括：视频切分算法优化、带宽选择算法优化。

1.4 论文组织

论文各章的主要内容如下：

第一章：绪论。介绍论文的研究背景；介绍与论文研究相关的DASH技术、视频评价算法、视频切分算法、码率调节算法的国内外研究现状；阐述论文的研究目标、研究思路，简单安排论文的内容及组织安排。

第二章：视频切分算法设计。针对MOOC视频的特点，提出一种新的视频切分算法，该算法可以根据内容将视频进行切分，并将切分后的视频分段进行分类，最后按特定的码率表根据视频分段的分类将其转码。这种算法可以减少视频分段的大小，并让视频客户端具有按需下载视频的能力，可以大幅提升用户的实际体验。

第三章：码率调节算法设计。提出一种新的码率调节算法，该算法可以根据用户当前所处网络状况，自动请求适当码率的视频，可以避免视频播放器出现卡顿现象，减少QoE下降程度。

第四章：系统设计与实现。详细描述移动视频学习平台的服务端、代理服务端、客户端的设计方案以及系统架构，并描述每个模块的具体工作职能及工作流程。

第五章：实验及数据分析。利用上文设计的系统，从服务端的视频切分算法、客户端的码率调节算法以及用户体验三个角度进行实验，论证本文所述算法的性能或准确率，并比较本文算法的具体优化效果。

第六章：总结与展望。对全文进行总结，并归纳本文研究点的不足之处，为下一步研究做好准备。

第二章 视频切分算法设计

本章主要讨论对MOOC视频进行DASH适配过程中涉及到的视频划分算法设计及优化。

2.1 视频切分算法综述

相对于整体质量，用户更加关注视频的主体区域，即感兴趣区域ROI（Region of interest）[25]的质量；而不同内容的视频，感兴趣区域各自不同。所以，视频码率和视频内容是影响用户对视频质量评价的两个重要因素。虽然DASH对视频码率的范围没有明确的限制，但过大的带宽消耗会明显增加用户端缓冲所需要的时间。因此，对视频切分算法的优化将主要着眼于两方面：尽量减少QoE损失和降低视频带宽开销。

DASH协议规定，在处理视频时，算法需要先按固定的时长将视频切分为多个分段，再按照固定的码率表将视频分段转换为多种码率的副本。这种切分操作有两个弊端。一方面，不同内容的出现时长各不相同，这种切分方法会导致视频分段中可能包含多种码率差异较大的内容，这将导致该分段的平均码率较大，转码时产生一定的带宽浪费。另一方面，视频的内容呈现多样性，按固定的码率表对其进行转码无法体现这种多样性：对某些较低码率的分段，这种方法会显著提高带宽消耗而QoE并没有相应提高；对码率较高的视频分段，当码率表的码率分布显著低于该视频分段的码率时，这种方法会导致视频质量显著降低。

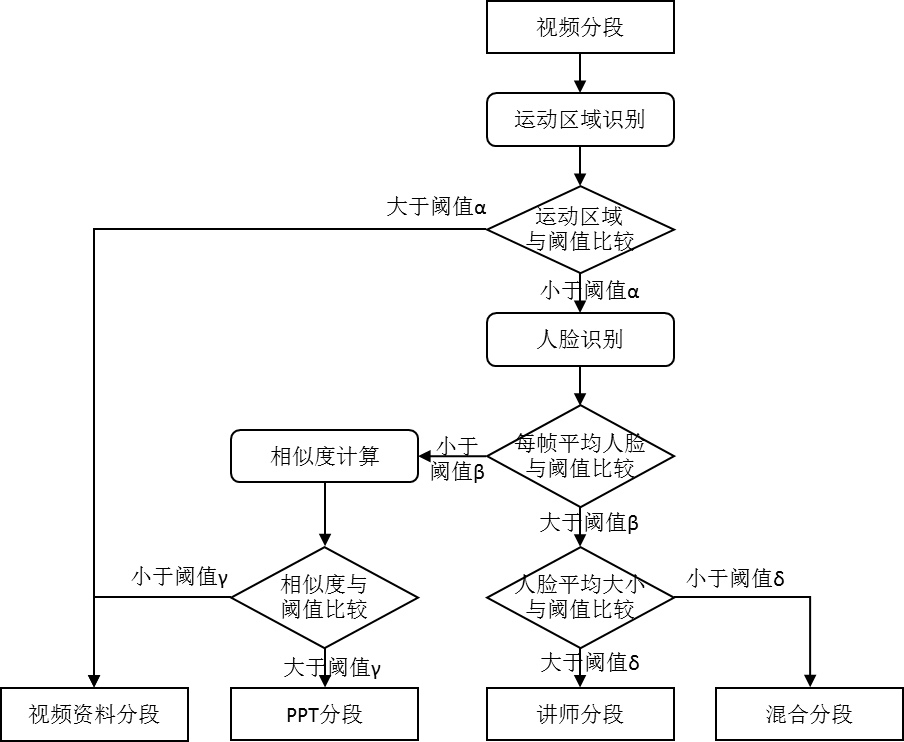
针对这些问题，本文设计的视频切分算法采用如下方案。

对于相同的内容，其相邻帧之间颜色分布应无较大变化；在不同内容的切换处，颜色分布会出现较大变化。所以，先根据颜色的变化寻找视频内容的切换处，再对视频进行精确切分，可以获得与内容高度相关的划分结果，显著提升视频划分的准确率。对划分结果，算法进一步根据人脸、运动量等参数判断其内容，并根据该内容对应的码率表对其进行转码工作，可以将每个视频分段的码率控制在合理范围之内，以在保证QoE的前提下大幅减小带宽消耗。该算法的决策树如图1所示。

下文将详细阐述每个步骤的具体实现方案，以及算法中所有参数的测定方案。

2.2 MOOC视频分段的类别

MOOC视频的特点为以讲师授课、展示PPT为主，偶尔穿插视频资料，视频整体的运动量较小，其平均码率一般处于较低水平。结合绪论中对MOOC视频的定义，可以将MOOC视频的分段分为以下几类。如图2为视频分段的典型截图示例。



**图1 视频分类模块的决策树**



**图2 四种不同视频分段的截图示例**

**左上：讲师分段 右上：PPT分段 左下：混合分段 右下：视频资料分段**

2.2.1 讲师分段

讲师分段（Instructor-Centered Segment）的内容一般只包含在固定位置授课的讲师，画面中不存在文字资料，讲师的移动范围有限。此类分段的画面一般由1个讲师组成，少数分段的画面存在2个以上讲师。讲师是此类视频分段的主体；一般情况下，讲师全身应占据至少1/3的画面面积，讲师面部应至少占据1/6的画面面积。

2.2.2 PPT分段

PPT分段（Slides-Presentation Segment）的内容一般只包含讲师展示的静态演示文稿，同期音频一般为对画面的实时讲解。此类分段的画面一般由静止的图片和文字构成，部分分段包含少量动态背景、动态图片及动画效果。一般情况下，上述动态内容在此类分段中占比不大，并且上述内容所在区域占据的面积不大、存在时间也较短。

2.2.3 混合分段

混合分段（Instructor-Slides Segment）的内容一般包含讲师和讲师实时讲解的资料。这里的资料可以是PPT、动态视频或者动画等等多种形态。此分段画面分为两个区域：第一个区域由讲师的半身或全身构成，讲师在此区域内小幅移动；第二个区域由图片，文字或动态内容占据，通常为讲师当前讲授的内容。由于画面一侧的资料画面一般处于静止，所以此类分段的运动量取决于画面另一侧的讲师占据画面的大小。

2.2.4 视频资料分段

视频资料分段（Motion Segment）的内容一般为讲师选取的视频资料。该类分段一般由运动量较大的视频资料构成，资料多样性丰富，码率一般较大。此类分段的内容可能出现人脸，且人脸出现的数量、大小、时间均不确定，分类时容易与讲师分段/混合分段产生混淆。部分视频资料运动量较小甚至没有运动，由于其码率与PPT分段区别不大，可将其作为带有渐进动画的PPT分段处理。

2.2.5 其他分段

其他包括所有上述四类分段之外的分段，包括但不限于无实际意义的全黑片头、全黑片尾、淡入淡出、视频特效等等，通常出现在视频头部、视频尾部及内容切换处。由于这些无意义的片段一般对QoE影响较小，并且码率不高，故在分类过程中，通常将这些分段分类为PPT分段，部分运动量较大的分段可以分类为视频资料分段。

2.3 视频分段算法

如上文所述，不同内容的视频分段颜色差异较大，相同内容的视频分段内，颜色分布变动较小。若直接将视频等长切分，在视频切换处的视频分段将不易于处理。如果将视频按内容切分，不仅视频分段将易于分类，位于切换处的视频分段的质量也更加容易控制。所以本文认为，研究一种可以按内容切分视频的分段算法是相当重要的。

在设计视频分段算法时，本文采用如下方法：统计每帧的颜色分布，并逐帧计算颜色差异值，选取差异值较大的时间点作为切分点，即可得到根据内容准确划分的视频分段。由于视频图像对几何变换及图像质量的变化不敏感，所以该方法可以用于检测相邻两帧图像在颜色分布上的差异。令为第i帧的颜色分布矩阵，为此帧内颜色为(r,g,b)的点的个数，则此帧的颜色分布矩阵可表示为：



由于需要对全部RGB色域进行统计，单次统计操作消耗内存较大，消耗较多服务器资源，所以这种算法效率较低。实际应用中，通常将统计的粒度适当增大，即以（n=0-8）为单位，将全部RGB色域划分为等大小的区域，在区域内的颜色点合并统计，以减少内存开销。令为第i帧的颜色分布矩阵，统计粒度为n，为的元素，则此帧的粒度为n的颜色分布矩阵可表示为：



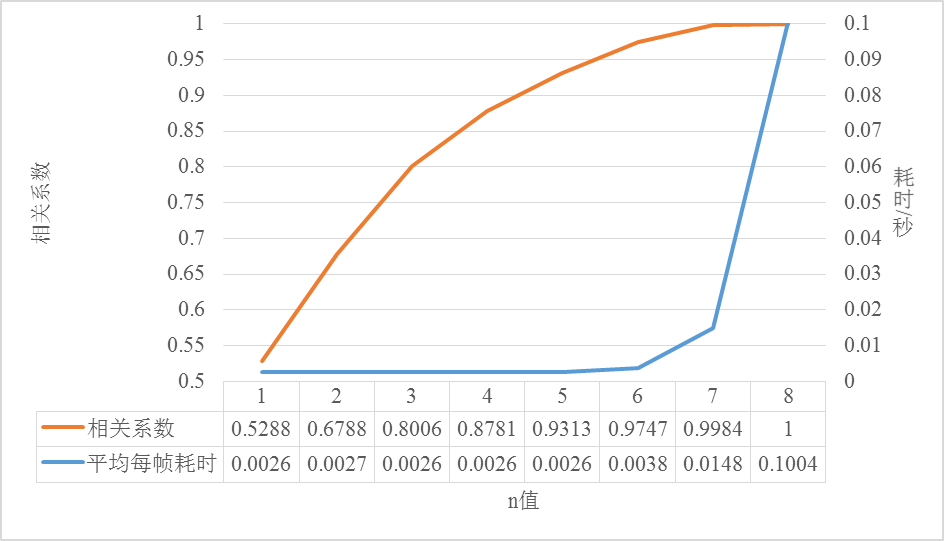
第i与i+1帧的颜色分布差值可表示为：



若该值大于阈值，则记录此时间点，将该时间点作为切分点即可。

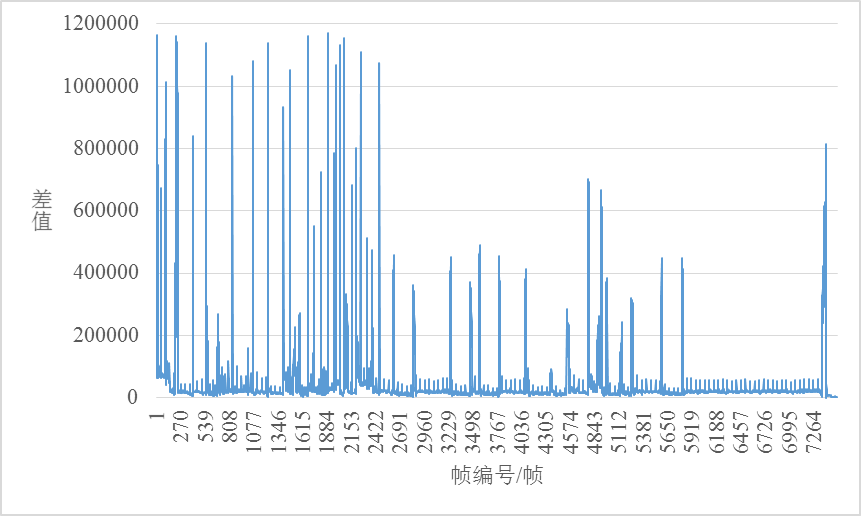
不同的n值对应不同的阈值。若每帧的大小为p\*q像素，则此算法的时间复杂度与n无关，为o(pq)，空间复杂度为o()，故n的取值决定计算速度及计算精度；过大的n值会导致颜色特征不明显，过小的n值则会消耗过多资源。需要说明的是，当n=0时，矩阵只有一个元素，该元素的值为该视频的分辨率，无实际意义，故下文不对该值进行讨论。

下图3为计算某段MOOC视频的时间消耗及与原始数据（n=8时）比较的Pearson相关系数。可以看出，n=1至5时，计算时间几乎相同；从n=6开始，计算时间呈指数级上升。而作为原始数据的精简版本，n=1至7时的颜色分布数据均有一定损耗，而n=5至7时数据的相关系数大于0.9，具有极强相关性。因此，实际计算时取n=5可以获得较好效果。



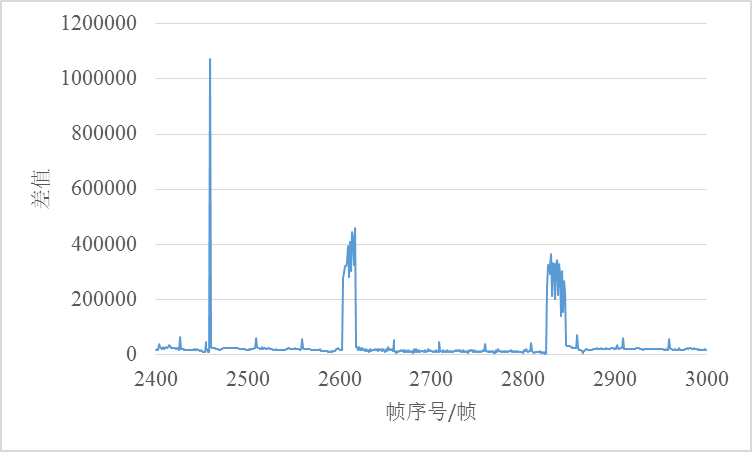
**图3 计算MOOC视频颜色分布的耗时和Pearson相关系数**

结合多个视频的数据分析，文章认为，大部分视频分界点处的颜色分布差值都在以上，因此将上文提到的切分阈值设为符合大多数视频的要求。下图4为上文对MOOC视频计算后得到的n=5时的颜色分布差值图。可以看到，全部峰值都在以上，符合上文所述阈值设定。事实上，后期人工对该视频标记的划分点也与此图中的峰值完全符合。



**图4 某段视频的颜色分布差值图（n=5时情形）**

视频的切换方式通常可以分为直接切换和渐变切换，两种切分方式具有不同的颜色变化特征。如图5为上图4中帧编号2400至3000之间的颜色分布差值图，其中第一个峰值（帧编号2459至2460处）为视频内容的直接切换处，具有明显高于其他值的特征，在图中呈尖状分布；第二、第三个峰值（帧编号2606至2620处及2828至2849处）为视频内容的渐变切换处，此处的颜色变化略低于直接切换，但是会持续一段时间，在图中呈梯形分布。通常渐变切换处的视频码率较高，但用户对此处的质量变化不敏感，故对渐变处存在两种处理方法：可以将其与后续视频分段一并处理，也可以将其独立处理，并赋予较低的码率。由于二者在QoE方面区别不大，但是第一种方案的处理速度较快、效率较高，故切分算法采用第一种方案。



**图5 图4中帧编号在2400～3000之间的颜色分布差值图**

据此处理后可以得到一系列切分点，根据这些点可以将视频切分为多个分段。对于切分后较长的分段，切分算法将进一步对其进行定长切分，以保证视频分段长度在5～10秒之间；对于切分后较短的分段，切分算法将该分段与附近相邻分段合并处理，具体合并方式取决于三者的实际码率。通常情况下，按此操作切分后，长度较短的分段一般为视频资料分段、切换速度极快的PPT分段或其他分段，用户并不会关心这些较短分段的内容，所以将这些分段与前面或后面的分段合并处理并不会影响QoE，反而会降低这些较短分段的码率，可以减少带宽消耗。

2.4 视频分类算法

视频分类的目标是将具有相同特性的视频分段归并在一起，以便将相同特性的视频分段转码为相同的码率。而分类工作的核心是提取视频的特征并判断视频分段的特征是否与特定分类吻合，进而对其进行分类。所以，视频分类工作的准确度取决于特征的定位、特征提取的精度以及特征阈值的选取。通过2.1节的分析，MOOC视频分段主要分为讲师分段、PPT分段、混合分段、视频资料分段四类，划分这些分段的主要依据为视频的运动量、人脸参数等。其中，由于PPT分段运动量较小、视频资料分段运动量较大，通过计算运动量可以有效区分这两类视频；而讲师分段及混合分段均包含人脸，二者的人脸大小及分布范围各不相同。根据这些特性，算法可以对讲师分段及混合分段进行分离。

需要指出的是，由于视频内容的多样性，对于无限定范围的内容识别，现有的非机器学习图像检测技术尚无较好的解决方案，而涉及机器学习的检测技术又需要建立异常庞大的内容库。作为一个轻量化的算法，这个视频分类算法只能对特定内容进行识别，并存在一定的误识别率。本文认为，对于某些严格意义上属于错误分类的特殊情形，只要这种情形具有一定的实际意义，即可忽略这些特殊情形，以增强算法的执行效率。下文会在适当的位置指出并逐一分析这些特殊情形。

2.4.1 运动区域计算

只采用2.3.2及2.3.3两种特征计算方式进行识别，亦可以完成识别工作。但是在后期的分类实验中发现，由于部分视频资料分段包含形态各异的人脸，当进行识别时，此类视频分段易被误分类为讲师分段/混合分段。讲师分段或混合分段的最高码率远低于视频资料分段的最高码率，所以，此类误分类情形将极大降低视频分段的质量，进而对QoE造成影响。而采用运动区域计算作为识别工作的第一步，可以有效地减少视频资料分段导致的误识别现象。

在分析各类视频分段的特性后发现，每类视频的运动区域各不相同。这里的运动区域，是指视频分段任意相邻两帧的像素发生变化的区域的大小。如果将某视频分段内某相邻两帧记作和，则这两帧的运动区域矩阵可以记作D：



如果将1视作白色，0视作黑色，则该矩阵可以视作一张黑白图像，白色区域为A与B像素不同的区域，可以称之为运动区域。如图6为某视频内连续两帧及这两帧的运动矩阵，可以看到，左下图中未处理的运动区域分布较为分散，连续区域较小，分布不均匀，需进一步处理才可用于识别分类。实践中，先对白色区域进行腐蚀，然后对其进行膨胀，可以去除大部分较小的区域并将较大的区域进一步扩大，使紧邻的运动区域合并。然后取其中最大的区域，以此进行识别，可以最大程度地体现分段的特性。如图6右下所示，经过处理后的运动区域区域图已经几乎没有噪点，区域分布非常集中。



**图6 运动区域的计算过程**

**上：两张相邻帧图片 下：对应的运动区域图 下左：处理前 下右：处理后**

下图7分别为PPT分段、讲师分段、混合分段、视频资料分段的处理后的典型运动区域图。可以看出，由于PPT分段多为静止图文，故运动区域很小；讲师分段的运动区域为半身人形，多集中在画面中间，占据区域约为全部区域的1/3至1/4；混合分段的运动区域为全身人形，通常分布在画面一侧，约占全部区域的1/10至1/8；视频资料分段的运动区域非常广，通常占满全部区域，只有少数视频资料分段的运动量小于全部区域的一半。由此可见，绝大多数情况下，运动区域大于全部区域一半的视频分段属于视频资料分段，并且其他视频分段的运动区域小于全部区域的一半。由于视频内容的连续性，绝大部分单一内容的视频分段，每帧的特性基本相同，所以对视频分段进行等间隔抽样，以抽样结果计算得到的运动区域大小的平均值可以很好地反映该视频分段的特性。

据此，可以将图1中阈值设置为0.5，也即将符合此条件的视频分段提前分类为视频资料分段，可以很好地解决视频资料分段误识别为其他分段的情况。

需要指出的是，如果某种非视频资料分段的视频分段的运动区域值大于1/2，则该分段会被错误分类为视频资料分段。虽然在计算分类准确率时需要将其计算为误识别情形，但是该识别方式有其必然性。事实上，由于运动量大，视频压缩算法对该分段的压缩比率一般较低，所以一般码率较高。此时，无论是对其进行ROI转码（ROI区域较大，效果不好）还是对其采用正确的分类对应的码率进行转码（码率远低于原值）都会导致QoE大幅降低；而将其分类为视频资料分段，在客户端所处的网络质量较好的情况下，可以保证用户获取到的视频分段清晰度较高，进而提升QoE。



**图7 不同视频分段的典型运动区域图**

**左上：讲师分段 右上：PPT分段 左下：混合分段 右下：视频资料分段**

2.4.2 人脸识别

由于MOOC视频的特性，大部分视频分段中包括占据面积较大的人体图像，故对人脸进行识别并判断人脸在全部区域中占据的面积，对正确区分这些视频分段非常重要。

在人脸识别模块，本文采用了基于Harr分类器[15]的人脸识别技术。Harr分类器识别人脸的主要工作原理是：在寻找人脸的过程中，一个子窗口将会从左上角开始，在待识别的图片中逐行从左向右滑动。每滑动到一个位置，算法就对窗口内的部分计算harr特征，再利用级联的决策树计算harr特征，并判断该特征是否与人脸的特征相同或者相近。这种方法相对比较简单，基于这种方法的成熟算法已经有很多种。然而此种方法的误识别率较高，会生成莫名其妙的结果。图8提供了一些误识别的案例。

由于Haar分类器丰富的特征库，我们可以采用该分类器识别多种图形，例如人体全身、肩部以上人体、半身人体等等。这为本文改进人脸识别方案提供了思路：在进行人脸识别时，可以嵌套识别两种特征，先识别较大特征，再在较大特征的范围内识别较小特征。以此思路，本文在设计人脸识别步骤时，采用先识别肩部以上人体，再识别人脸两步骤识别操作，可以大幅减少识别错误的情形。由于混合分段主要包含一个完整人体，讲师分段主要包含胸部以上人体，所以此种识别方法可以正确识别出MOOC视频分段中包含的人体。如图9所示，蓝色框体内为肩部以上人体的识别结果，绿色框体为人脸的识别结果。



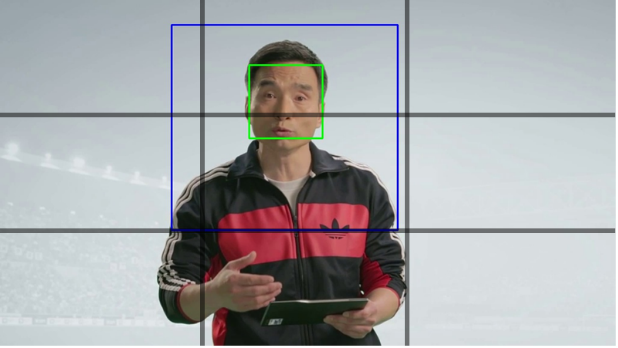
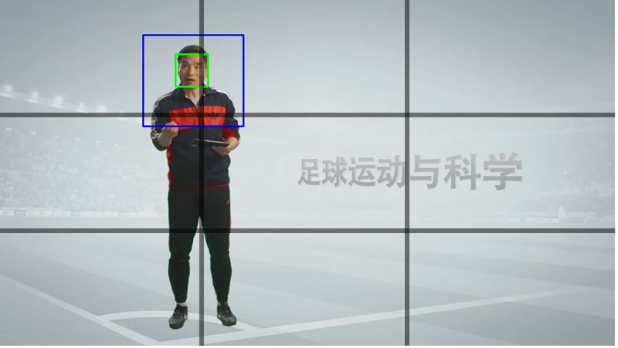
**图8 仅采用面部识别时的错误案例**



**图9 双重人脸识别的例子**

通过人脸识别可以获取到人脸的数量及每个人脸的大小、位置，根据这些数据可以将视频分段进行分类。事实上，讲师分段及绝大部分混合分段都有且仅有一个人脸，混合分段可能不包含人脸也可能包含多个人脸，绝大部分PPT分段不包含人脸。一般包含多个人脸的混合分段都包含较剧烈的运动，所以此部分分段可以在2.3.1步骤中被选出，因此不会对此步操作造成较大影响。混合分段有可能出现大于1个人脸的情形，这些非讲师人脸一定出现在资料一侧，并且具有不活动或活动范围极小、面部占据空间有限、出现时间短于讲师的特点。

根据以上分析，我们可以将每帧平均人脸数在1～2的视频分段选取出来，即图1内的范围，并进一步判断人脸在这些视频分段的位置、大小。图10为混合视频和讲师视频的典型截图。事实上，如果将讲师视频的任意一帧分为9等份，如图10左侧图例，则讲师半身区域在其中应至少占1份的面积，并且讲师的大部分处于中间一列中；同样，如果将混合视频也分为9等分，如图10右侧图例，则讲师全身应至少有一半在左部分或右部分一列，并且上半身区域面积一定小于1份的面积。综上所述，本文可以对以上两类视频进行如下定义：讲师上半身区域面积大于画面总面积1/9，并且讲师主体处于画面中部的是讲师视频；讲师上半身区域至少有一半处于画面左部或右部的是混合视频。

**图10 讲师位置的分析**

**左侧：讲师视频 右侧：混合视频**

经过以上分析，我们可以确定包含人脸的讲师视频及混合视频的区分方式，即可设定图1内阈值为1/9。其他未区分的视频则需要进入下一步继续分析。

2.4.3 相似度计算

到此步骤为止，还有PPT分段和部分视频资料分段没有分类。这些余下的分段的主要特性在于：二者都没有可识别的人脸，并且PPT分段的运动量远小于大部分视频资料分段。因此，此步骤需要一种可以快速计算视频分段综合运动量的算法。

在此，我们采用相似度计算来确定视频分段的平均运动量。相似度计算，是通过比较每帧间的相似程度，来计算视频分段整体的运动量大小。这里我们采用结构相似性指标SSIM来计算该特征。这种做法有两个好处：一方面，SSIM是一种比较成熟的算法，计算速度较快；另一方面，该特征可以准确体现视频分段内运动量的大小。

SSIM通过将图片转化为对比度、亮度和结构三个维度，计算并比较两个图片这三个维度的差异，以判断二者之间的相似程度。SSIM的计算方法为：对于相同尺寸的两个图片x与y，如果记x和y的均值分别为和，方差分别为和，二者的协方差为，每像素占用b比特空间，则二者的亮度相似指标为：



二者的对比度相似指标为：



二者的结构化相似指标为：



其中三个常数一般设为：



则x与y的SSIM值可由以下公式计算：

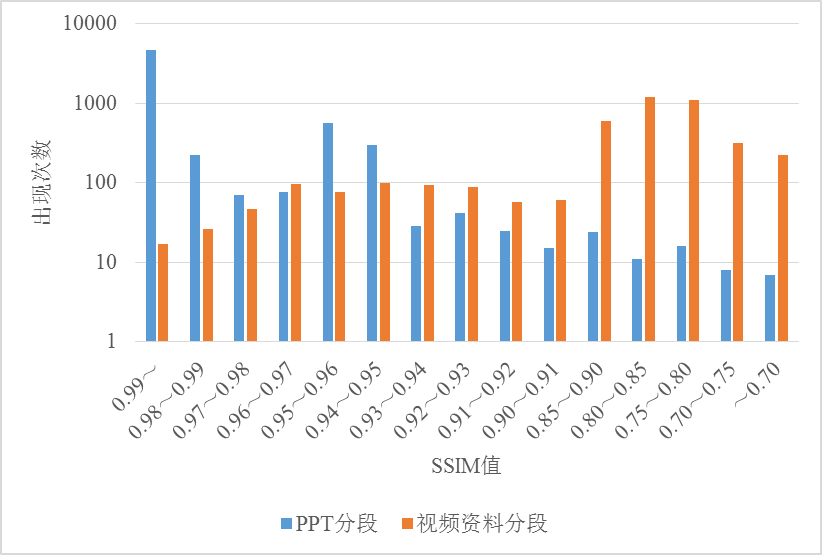


为确定划分PPT分段和视频资料分段的阈值，本文截取了5个MOOC课程的13个典型视频分段，其中包括7个PPT分段及6个视频资料分段，总长度为938.4秒，总帧数为10247帧。本实验通过计算其SSIM值，并观察不同视频分段的SSIM值，以判断是否具有合适的分界点。对这13个分段的计算花费了78.8秒，平均每帧仅花费7.7毫秒的计算时间，可见对SSIM值的计算是十分迅速、便捷的。如图11即为这些样本的统计结果，由于统计样本分布较为集中，故本图表选用了对数坐标轴以便于观察。统计发现，大部分PPT分段的SSIM值分布在0.94以上，集中在0.99～1区间，有4711帧的SSIM值落在此区域，占总帧数的77%；而大部分视频资料分段的SSIM值分布在0.9以下，分布较为分散、均匀。据此，本文认为，将区分PPT分段和视频资料分段的阈值，即图1中的阈值设定在0.94，可以较精确地区分视频资料分段和PPT分段。

2.5 码率划分策略

在根据视频特征对视频进行分类之后，该算法需要将视频转换成适当的码率以供客户端使用。为了提高QoE，该算法会根据QoE将不同分类的码率划分为数档，每一档代表不同的清晰度，提供给不同网络质量的客户端，而转码操作涉及到的码率表需要根据QoE进行调整，因此通过实验对其进行确定十分重要。

2.5.1 用户主观感受的客观测量方法



**图11 对一系列PPT分段及视频资料分段的SSIM值的比较**

用户主观感受，是指用户在使用产品时的QoE，代表用户在使用产品过程中该用户感受到的完成全部过程的难易程度。在讨论QoE时，需要先将该参数量化，以便于讨论用户主观感受与该产品某些参数的联系。量化QoE的主要方法有两种：一种方法主要采集用户对产品的评价并使用该数据对QoE与其他数据进行分析，另一种方法在采集用户评分的基础上，将某种可以计算的指标与用户评分建立函数关系，之后可以用该指标估算当前的QoE状况。由于数据量比较大，用户感受的评估可能准确度较差，故本文在讨论码率划分策略时主要采用第二种方法。

当用户观看基于DASH的在线视频时，本文认为，用户的QoE主要与包括视频质量、视频卡顿情况、视频加载时间在内的情况相关。由于此处划分的是一段内容独立的视频分段，所以不存在视频的卡顿、加载时间等情形，只需讨论视频质量即可。在比较视频质量时，本文主要采用峰值信噪比PSNR将原视频的每一帧与压缩后的视频进行比较，以得出视频的大致质量损失；同时，本文选用MOS值来测量QoE的大小。二者的关系可以参见表1[26]。需要说明的是，这种关系以原始视频作为5分，所以这种评估方式要求原始视频的质量要较高，才能达到较好的评估效果。下文在选取与质量评估相关的视频的时候，都将保证采用质量非常高（平均码率大于1000至1500kbps）的视频，以保证实验的准确性。

**表1 PSNR至MOS值的粗略转换**

|  |  |
| --- | --- |
| PSNR值（dB） | MOS值 |
|  | 5（非常好） |
|  | 4（好） |
|  | 3（一般） |
|  | 2（差） |
|  | 1（非常差） |

2.5.2 码率划分方案

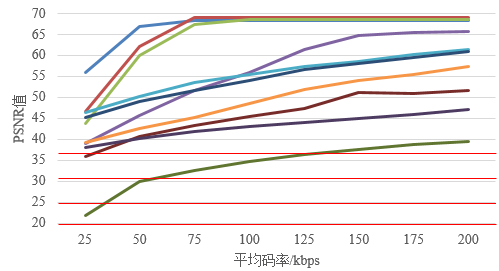
码率划分的目的是将原视频分段文件转码为多种不同码率的副本。在确定码率的等级时，需要根据用户体验划分码率；此处可以采用上文提及的MOS指数确定码率对应的用户体验，结合上文提及的MOS-PSNR转换表，即可获得客观的视频质量划分方案。本文认为，在划分码率时，可以根据每种视频分段的码率特点，将码率划分为2至4档，每一档的码率大小由对应的MOS评分确定。下面将对每类视频进行分类讨论。

PPT视频分段的运动一般不大；通常PPT分段中的运动部分集中在背景、动画特效等，所以码率一般比较小，压缩空间有限。故PPT分段可以仅划分为较少档次。为了确定PPT分段的码率设置，本文选取了4门MOOC课程的10个PPT分段，码率在20～300kbps之间，这10个分段的信息可以参见表2。将这些分段分别压缩为25kbps至200kbps之间的8种码率，每间隔25kbps设置一种码率。将压缩后的视频与原视频逐帧进行比较并计算平均PSNR值，以判断码率设置。

**表2 测试用PPT分段各类参数**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 速率（帧/s） | 5 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 |
| 码率（kbps） | 22 | 38 | 41 | 65 | 114 | 121 | 123 | 174 | 246 | 291 |
| 时长（s） | 48 | 100 | 68 | 30 | 3 | 660 | 3 | 4 | 2 | 38 |

计算结果如图12所示。参照表1可以看出，绝大部分分段的MOS值都在5分，仅有少数分段的MOS值为2至4。为了保证所有PPT分段的质量都达到一定标准，本文认为，将PPT分段的码率分为清晰、高清两档，分别设置为50kbps和100kbps，即足以达到较高的播放质量。



**图12 PPT分段码率与视频质量关系分析图**

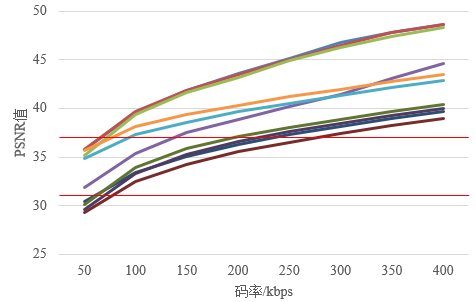
讲师分段的视频运动主要集中在讲师，其他区域运动比较缓慢，所以此分段的码率一般比PPT分段高，但比视频资料分段低。为了确定讲师分段的码率划分，此处本文采用与上文处相似的实验方案；本文选取了3门MOOC课程的10个讲师分段，码率在300至900kbps之间，这10个分段的信息可以参见表3。将这些分段分别压缩为50kbps至400kbps之间的8种码率，每间隔50kbps设置一种码率。

**表3 测试用讲师分段各类参数**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 速率（帧/s） | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 |
| 码率（kbps） | 346 | 362 | 419 | 421 | 556 | 608 | 642 | 697 | 764 | 810 |
| 时长（s） | 39 | 9 | 3 | 60 | 38 | 39 | 12 | 5 | 258 | 19 |

计算结果如图13所示。参照表1可以看出，可以看出，300kbps以上的分段客观评价都是5分，部分50至300kbps之间的分段为4分，一半的50kbps分段为3分。因为在50kbps时过半分段评价为4分、在150kbps时过半分段评价为5分、在300kbps时全部视频评分为5分，据此，本文认为，可以将讲师分段设为普通、清晰、高清三档，分别对应50kbps、150kbps、300kbps。

需要说明的是，由于混合分段的特性与讲师分段相同，并且混合分段的讲师移动范围通常没有讲师分段大，而且混合分段有大部分区域是保持不动的，所以混合分段的视频码率通常比讲师视频要低，最坏情况可以参照讲师分段。因此，本文认为，将混合分段的码率设置为与讲师分段相同的分档即可。



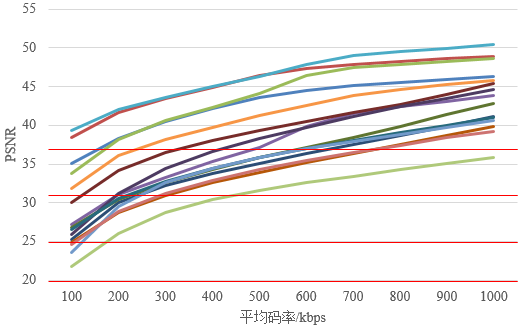
**图13 讲师分段码率与视频质量关系分析图**

视频资料分段的内容多样性非常丰富，因此分段的视频质量是最不可控的。所以视频资料分段的码率设置应倾向偏多分档、每档间距较大。本文同样采用了与上文相同的实验来确定该类分段的码率；本文选取了3门MOOC课程的15个视频资料分段，码率在500至1800kbps之间，这15个分段的信息可以参见表4。将这些分段分别压缩为100kbps至1000kbps之间的10种码率，每间隔100kbps设置一种码率。

**表4 测试用视频资料分段各类参数**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 速率（帧/s） | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 |
| 码率（mbps） | 0.5 | 0.5 | 0.6 | 0.6 | 0.7 | 0.8 | 1 | 1.1 | 1.1 | 1.2 | 1.2 | 1.3 | 1.4 | 1.4 | 1.8 |
| 时长（s） | 13 | 14 | 71 | 6 | 22 | 9 | 6 | 37 | 9 | 18 | 15 | 23 | 6 | 69 | 9 |

计算结果如图14所示；图中红线将整个图表划分为了5分、4分、3分、2分区域。由图可知，在200kbps时，全部分段的评分达到了3分；而过半视频达到4、5分的码率为300kbps、500kbps；绝大部分视频达到5分为800kbps。另外，为了照顾网络环境非常差的情形，本文认为，应加设50kbps档位，以保证用户在任何环境下都能观看视频。因此，在划分视频资料分段的码率时，应将该分段的清晰度划为流畅、普通、清晰、高清、超清五档，并分别按50、200、300、500、800kbps划分。



**图14 视频资料分段码率与视频质量关系分析图**

综上所述，本文认为下表5中的码率设置是合理的，按此码率表进行转码可获得较好的效果。需要说明的是，如果任意视频分段的码率低于该类任意一个码率设置，则在转码时将码率表的设置全部舍弃或者只保留比该视频分段码率低的码率设置，将舍弃的设置用原始视频码率代替。

**表5 码率设置**

|  |  |
| --- | --- |
| 视频分段 | 码率设置（kbps） |
| PPT分段 | 50、100 |
| 讲师分段 | 50、150、300 |
| 混合分段 |
| 视频资料分段 | 50、200、300、500、800 |

2.6 小结

本章主要讲述了对DASH中视频切分模块的设计方案。本文将MOOC视频分段划分为四类：PPT分段、讲师分段、混合分段、视频资料分段，每一类分段都有其独特的特征。据此，本文提出了一种视频划分算法，该算法根据视频的内容将其切分并分类。然后将分类后的视频分段根据码率表数据转码为不同清晰度的视频分段副本供客户端取用。通过此方法，可以在保证QoE的前提下，大幅减少大部分视频分段的带宽需求。

第三章 码率调节算法设计

本章主要讨论在播放视频过程中，DASH播放器中负责自动选择适当码率的码率调节算法的设计及优化。

3.1 码率调节算法综述

DASH服务端转码工作进行完毕后，会生成一个多媒体展示描述MPD（Media Presentation Description）文件供客户端下载。为了适应当前的网络环境，客户端在播放视频时，需要进行带宽适配操作，即：通过不断获取客户端的下载速率、缓冲区状况，客户端可以得到当前所处网络的网络质量；根据视频的MPD文件，选取最合适的码率，并向服务端请求对应码率的视频分段，如此往复。由此可见，为了保证QoE，如何估计当前带宽，并选取最合适的码率，是带宽适配操作的关键。如果选取的码率过大，则客户端将无法在限定时间内下载完视频分段，使得缓存逐渐减少直至变空，导致视频出现卡顿现象，QoE变差；如果选取的码率过小，则会造成缓存填充速度过快，使得客户端对带宽的感知速度明显减慢，并且无法提供高质量的视频，导致QoE将保持较低水平；如果选取的码率变化幅度、频率较大，会导致用户观看的视频质量持续抖动，造成QoE急剧降低。所以，本文尝试研究一种带宽适配算法，通过适当的调节过程，为用户提供高质量、稳定的MOOC视频，并保持较高水平的QoE。

3.2 码率调节原则

理想情况下，基于DASH的视频客户端应能在最短的时间内下载完任意分段并播放流畅、稳定的视频；据此，本文总结了码率调节的四条重要原则，当且仅当码率调节算法满足以下调控原则时，视频客户端才能达到以上理想情况。

3.1.1 防止视频缓存过多

一般情况下，缓存的大小是恒定的；移动终端的缓冲区大小更是远小于PC端。如果移动终端处于较好的网络环境下，视频播放器将很容易填满缓冲区。如果由于设计不当，导致码率调节算法来不及更新码率设置，则会导致客户端下载的视频清晰度低于最优值，缓存将迅速被较低码率的视频填满，导致用户无法观看到最高质量的视频。另外，一旦缓冲区被填满，基于下载量的带宽检测算法将无法工作，导致码率调节算法无法调整带宽，算法对带宽的灵敏度会大大降低。一般情况下，增大码率的调节频率可以防止这种情况发生。

理想情况下，该调控原则应当遵守；但是与其它原则相比，违反该原则带来的QoE损失是最小的，所以在设计算法时，可以不考虑这条原则。

3.1.2 防止视频缓存过少

一般情况下，移动终端所处的网络环境不会非常好，通常会处在单位数据费用较高、带宽波动较大的蜂窝移动网络。这种情况下，如果码率调节算法无法较快适应带宽的变动，导致当前播放器的码率设置大于实际带宽，则客户端将无法按时下载完视频分段，缓冲消耗速度将会过大，缓冲区的视频分段将逐渐变少直至没有视频可以读取，这意味着用户将经历一定时间的视频卡顿。

由于视频卡顿非常影响用户体验，所以在码率调节算法力所能及的范围内应尽力避免这种情况的发生。通常情况下，增大码率的调节频率，可以令算法对带宽波动的灵敏度提高。

3.1.3 平滑调节带宽

在观看视频时，用户更希望能观看到视频质量相差不多的视频；如果用户可以感知到视频质量在频繁变动，则QoE将显著降低。为验证这条假设，本文选取一段MOOC视频的讲师分段，长为5秒，将其转码为3种不同质量的码率，具体码率设置可以见表6。考虑到受试者的心理，单个测试的视频时长设置为20秒，即包含4个分段，每个分段的设定可以参见表7。该实验选取了10名受试者，每位受试者独立观看5段测试视频，并按照5分制MOS的规定进行打分。实验的平均评分结果可以见表7。

**表6 视频分段的清晰度对应码率**

|  |  |
| --- | --- |
| 清晰度 | 码率 |
| 流畅 | 80KBps |
| 普通 | 180KBps |
| 高清 | 280KBps |

由表7可见，编号1的视频，所有的分段质量都是相同的，得分也最高。编号2、3的视频都有两个高清分段、两个流畅分段，不同之处在于流畅分段出现的时间，编号3的视频中间两个分段的视频质量是相同的，而编号3的视频得分也略高于编号2。以上数据都表明，无论视频质量如何，用户都更希望看到视频质量变动较少的视频。而编号4、5的视频都有两个普通分段、一个流畅分段和一个高清分段，在编号5的视频里，流畅分段出现的时间更晚，而得分却大大低于编号5的视频。这表明，视频质量与用户体验的关系不是线性的，而是指数关系的。这也验证了上文中QoE-SSIM公式所建立的关系。

**表7 受试者对不同视频质量的测试视频的打分情况**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 分段1 | 分段2 | 分段3 | 分段4 | 平均评分 |
| 1 | 普通 | 普通 | 普通 | 普通 | 4.07 |
| 2 | 高清 | 流畅 | 高清 | 流畅 | 3.36 |
| 3 | 流畅 | 高清 | 高清 | 流畅 | 3.40 |
| 4 | 普通 | 流畅 | 普通 | 高清 | 3.40 |
| 5 | 普通 | 普通 | 流畅 | 高清 | 3.07 |

综上所述，用户更希望看到质量相对稳定的视频，而不是质量频繁波动的视频，尽管平均视频质量可能不变甚至更高。这就与前两个原则里频繁调整视频质量的要求相悖；算法需要找到一个调控频率的平衡点以同时满足这三条原则。

3.1.4 内容对QoE的影响

虽然用户对视频质量的调节很敏感，但是在视频内容的变动处、无实际意义处，质量的变动不易被察觉到，例如视频切换、渐变切换、黑屏等等。事实上，由于渐变切换处、黑屏处本身就较难判断视频质量，并且视频切换处的画面变化很大，因此在这些位置前后的视频质量变化不易被用户察觉。因此，当播放器缓冲模块正在处理相同内容的视频分段时，调控算法仍然应当遵循上文提到的第三条原则；但是当模块运行到上述位置时，可以适当增大调控幅度，这对用户体验的影响微乎其微。

以上四个原则是调控算法设计时需要考虑的重点；完全根据以上原则设计的算法，可以保证调控效果较好、用户体验较高。

3.3 码率调节算法

根据以上四条原则，本文提出以下码率调节算法。

3.2.1 带宽测量方法

码率调节算法首先需要获取当前带宽的准确状况，以供后续步骤确定适当的码率。通常情况下，受到网络环境因素的影响，客户端当前的可用带宽大部分时间都处于不稳定的状态。作为码率调节的重要依据，准确测量实时带宽是整个码率调节算法的基础。上文提到，带宽估计有主动探测及被动探测两种方法。与主动探测方式相比，被动方法不涉及复杂的网络技术，易于实现，算法占用资源少。本文认为，被动方法提供的数据准确度已经足够，所以本文将采用被动方法对带宽进行测量。

带宽测量的被动方法，主要是利用下载速率数据进行适当的计算，来估计当前带宽的大小。带宽测量算法将从下载任务开始，到下载任务结束，统计单位时间内的下载速率，没有下载任务时不予统计。如果将某个视频分段下载任务的起始下载时间记作，结束下载时间记作，则任意满足的时间至的播放器数据下载量d（字节）可定义为



则至的数据下载速率（比特）为：



需要说明的是，为了得到准确的下载速率数据，在任意分段开始下载之前，及视频分段下载结束之后，这些时间不计入播放器播放MOOC视频时的下载速率计算中。播放器以1秒为周期，通过上方公式采集每个自然秒内的下载速率，并以此作为调整当前码率的依据。这种下载速率计算方法的优势，主要体现在可以精确体现当前带宽的波动；由于在用户观看视频时，移动端的后台通常没有数据请求，或者数据请求占据比例较小，这也是可以使用这种方法测量带宽的现实基础。

3.2.2 下载速率数据优化

由于上文所述测量方法对带宽的变动非常敏感，在获取到下载速率数据后，有必要对其进行优化。数据优化的目标，一方面需要减少这种特性对码率调节过程带来的扰动，另一方面也要保持数据中带宽变动的趋势。所有此处及下文用到的变量及其含义可见表8。由于算法中常数值通常为经验数值，无法通过实验确定，故下文将给出本文通过多次尝试确定的效果较好的一组常数设定。

首先，按上文计算方法，如果以1秒为周期进行采样，可得到播放视频期间的实际下载速率序列：



其中时间范围和可按如下方法确定：当视频分段s播放完毕时，算法对带宽进行一次采样，时间等于上次采样时间，等于采样点位置的时间；其他情况下，采样周期为1秒，等于采样点位置的时间，。

**表8 带宽选择算法所用变量含义**

|  |  |
| --- | --- |
| 变量 | 含义 |
|  | 分段s的码率设置序列 |
|  | 算法选定的码率序列 |
|  | 实际下载速率序列 |
|  | 去峰下载速率序列 |
|  | 加权平均下载速率序列 |
|  | 分段所属类型序列 |
|  | 缓冲区最大长度 |
|  | 缓冲区当前长度序列 |
|  | 常量 |

其次，如上文所述，当相邻视频分段的内容相同时，应避免码率的大幅调整，并降低码率调节的频率，以保证用户体验不会降低。因此，算法需要适当削减下载速率数据中的波动部分。事实上，带宽减小会导致下载速率迅速变慢，算法需要对这种情况做出迅速反应以免缓冲区缓存迅速减少，所以算法需要保留数据中的波谷；带宽增加通常是暂时的，算法可以适当削减数据中波峰的大小，以保证视频质量不出现较大波动。据此，本文可以根据下述公式得到去峰下载速率列表：



其中时效果较好。

再次，本文认为，在码率调节时，应参考一段时间的码率，以避免算法对带宽波动过于灵敏导致的视频质量频繁波动。在此本文采用加权平均值法；事实上，距离调节时间越远，数据的效度越低，应用加权平均法可有效突出距离当前时间较近的数据。因此，本文可以根据下述公式得到加权平均下载速率列表：



其他

i≥p+q

其中，时效果较好。

按此方法处理后的数据即可用于码率调控，可以获得较好的准确度。

3.2.3 码率调节方法

首先，算法需要一个可以体现缓冲区缓存多少的常数。事实上，由于缓冲区当前的缓存决定了播放器对带宽波动的耐受程度，所以当缓存较少时，应尝试提高缓存的存量，并且保证视频质量的稳定性。本文认为，当缓存少于缓冲区容量的30%时，意味着缓冲区需要尽快补充缓存，并且缓存越少，这种需求越紧急；而当缓存大于这个比例时不用调整。因此，本文设计了一个常数，该常数由一个分段函数计算得来：



若按上文描述，上式中常数和均等于0.3即可。

最终，算法可以根据上述参数估计适合下一段视频s+1的码率。如果记时刻k为下一段视频s+1的开始时间，j为比时刻k早且距离时刻k最近的采样时间点，则下一段视频的码率应为：



视频播放器只需根据码率获取对应码率的视频分段，即可获得较好的QoE。

3.4 小结

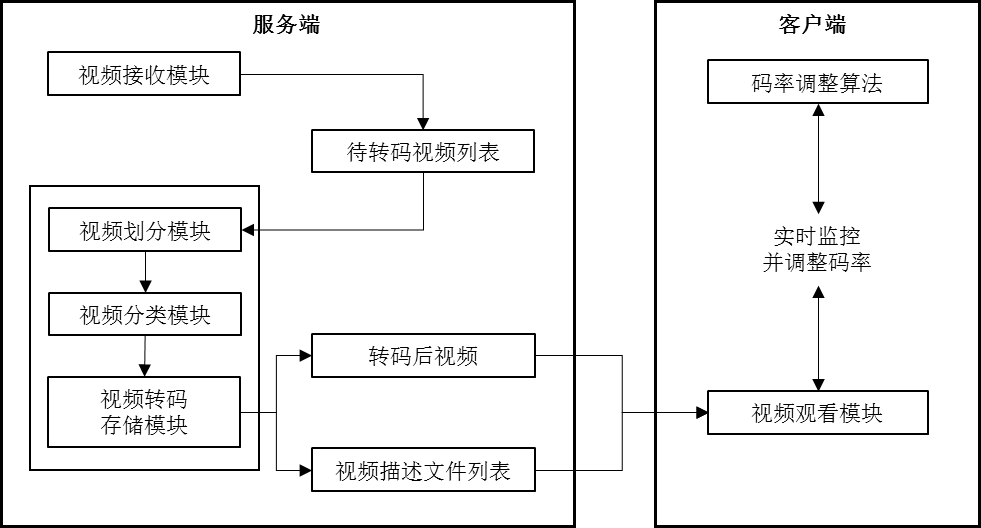
本章主要描述了一种码率调节算法及该算法的设计思路。本章先对码率调节的原则进行了分析，并定义了四种应当遵守的码率调节原则：防止缓存过多、防止缓存过少、平滑调节带宽、考虑内容对QoE的影响。根据以上四个原则，本文设计了一种基于内容及缓冲的码率适配算法，该算法可以有效地感知并突出当前带宽的变动趋势。除此之外，当缓冲区缓存数量较少时，该算法可以有效地提高缓冲区填充速度；当待调控视频分段的内容与前一视频分段相同时，该算法可以有效地减少码率的调整，使视频质量趋于稳定。通过此方法，既可以减少带宽的消耗，也可以维持并适当增加QoE，为用户提供质量较稳定的MOOC视频。

第四章 系统设计与实现

本章主要叙述面向MOOC的移动视频学习平台的整体设计与实现方案，详细介绍系统的整体架构，以及每个模块的具体功能、实现细节。如何将上文所述的优化方案及算法应用至各个模块中并协调各个模块可正常工作，以完成最终的目标，在下文都将有详细讲述。

4.1 系统总体设计

本移动视频学习平台的主要设计目标是：在服务端接收教师上传的指定课程，将其处理并转码为多个视频分段，每个视频分段可以提供不同质量的副本，将其存储至远端，以供用户浏览课程简介并学习课程。系统分为服务端、客户端两部分，主要运行流程及二者关系如图15所示。



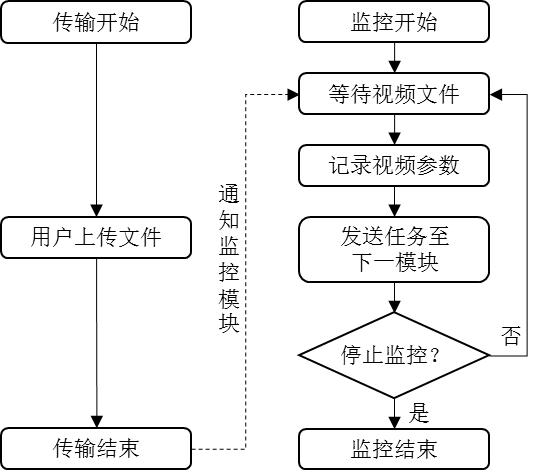
**图15 移动学习平台系统总体结构**

4.2 平台服务器设计

服务端的主要任务是接收用户传来的视频，将其按内容划分分段、分类、转码并存储至本地，根据视频分段情况生成MPD文件。其主要流程图可参见图15，按功能可将其划分为视频接收模块、视频划分模块、视频分类模块、视频转码存储模块。一般情况下，本文所述视频划分、分类模块采用Python设计，服务端使用Django写成的前端界面和基于Apache的HTTP服务器以提供上传视频、分发视频分段时的存储服务。各个模块的功能将在下文详述。

4.2.1 视频接收模块

本模块的主要作用是将文件上传、保存至服务端。服务端提供一个页面供用户选择传送文件并将其上传；服务端有一个观察者模块，该模块负责监视上传文件；待用户将原始视频文件传输至服务端后，由观察者模块生成一个视频处理任务并将该任务发送至视频划分、分类模块。该模块流程图如图16所示。



**图16 视频接收模块流程图**

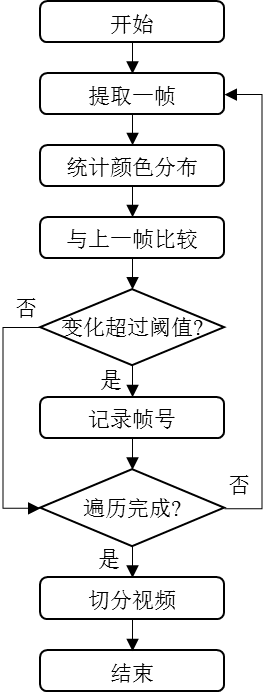
该模块上传部分由采用Django实现的上传页面实现，监控部分由采用观察者模式的模块实现。扫描模块每隔固定时间扫描一次上传文件夹，如果发现刚刚上传的视频文件后，则由处理模块调用FFMpeg对视频进行预处理。利用FFMpeg可以获取视频文件的类型、大小、时长、帧率、分辨率等参数，并将分析结果发送至下一模块，以供下一模块对视频进行处理使用。

4.2.2 视频划分模块

本模块的主要作用是将文件根据内容切分，以供下一模块进行分类工作。本模块的流程图如图17所示。

本模块主要采用OpenCV对视频颜色分布进行计算。OpenCV即开源计算机视觉库（Open Source Computer Vision Library），是一个可以运行在多种操作系统上的轻量化计算机视觉库，它实现了多种图像处理、计算机视觉的通用算法，调用方便，运算快捷。

该模块收到上一模块发来的视频信息后，使用OpenCV打开视频，并遍历视频的全部帧；对每一帧，采用OpenCV的颜色分布统计算法对每帧的颜色分布进行统计。该开源库的颜色分布统计算法较为成熟，计算速度已在上文分析过，非常迅速，故选取该算法对颜色分布进行统计可获得较好的效果。获取到这一帧的颜色分布后，将颜色分布以矩阵的形式存储在内存中。Python的科学计算模块NumPy对矩阵运算进行了优化，采用该模块对矩阵进行运算与普通存储相比，可适当提高运算速度。若这一帧不是起始帧，则用此帧与前一帧的颜色分布进行比较并计算差值的绝对值和，如果和值大于阈值，则可以认为这两帧的内容发生了较大改变，算法应该记录这一帧的序号。



**图17 视频划分模块流程图**

按此方法，算法依次计算每一帧的颜色分布及相邻帧的差值绝对值和，即可得到一个帧序号的序列，称为视频的内容分界点集。根据此集合，算法可以将该视频切分为多个视频分段；对于某些过长（大于10秒）的视频分段，可以按固定时长（5～10秒）对其进一步切分，并控制最后一个子分段的长度在3至10秒；对于某些过短（小于3秒）的视频分段，可以就近与下一个视频分段合并，此做法的依据在上文已说明。按此步骤切分，即可得到视频片段集，之后交付给视频分类模块以进行下一步操作。

4.2.3 视频分类模块

本模块主要的工作是对视频分段进行划分。需要注意的是，在进行视频划分操作后，后续的操作无需再次对视频分段进行切分；由于分类工作对文件的操作大部分是读取操作，而转码模块需要写入文件，故将二者分离并分别封装，有利于对系统的管理。

本模块的主要工作流程为：

（1）对视频分段进行运动区域计算，将运动区域大于阈值的视频分段分类至视频资料分段，其他视频分段送至（2）继续处理；

（2）对视频分段进行人脸识别，出现人脸频率小于阈值的视频分段进入（4）继续处理，大于阈值的视频分段进入（3）继续处理；

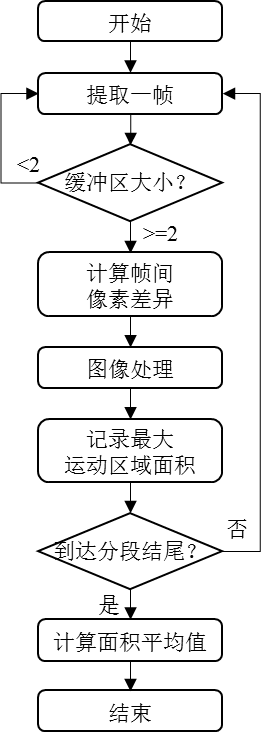
（3）继续分析人脸出现的位置及大小，人脸矩形框小于阈值的视频分段归类至混合分段，否则人脸较大，可归类至讲师分段；

（4）对视频分段进行相似度计算，相似度小于阈值的视频运动较少，可归类至PPT分段；反之运动量较大，则归类至视频资料分段。

本模块的决策树如图1所示，此处的全部阈值已在上文给出。本模块分为3个子模块：运动区域计算、人脸识别、相似度计算。

运动区域计算模块主要负责对视频的运动区域进行计算，其流程图如图18所示。该模块采用OpenCV对视频分段进行处理，先对视频分段进行遍历，将视频帧存入缓冲区。需要注意的是，由于视频内容的连续性，算法在处理对内容敏感的特征时，只需对视频进行抽样计算，即可获取能够代表该视频特性的特征。因此，实际应用时，该子模块采用每10帧取前2帧的抽取方法。当缓冲区所存放帧数达到要求时，采用OpenCV库的颜色系统转换算法将缓冲区内所有视频帧处理为灰度图片，并采用NumPy库的依次将相邻两帧的差值矩阵进行或运算，将最终得到的矩阵二值化，即可得到缓冲区内视频帧的运动区域图。对于得到的运动区域图，算法还需要对其进行腐蚀、膨胀等处理操作使分散区域变得连续并去掉较小区域，使该运动区域图更有特异性。最后即可记录最大运动区域的面积；待该视频分段抽样完毕后，取所有该面积值的平均值，即可作为该视频分段的运动区域。

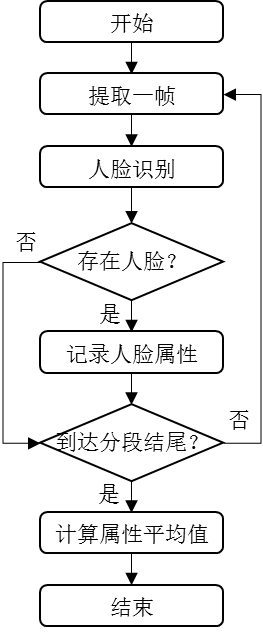
人脸识别模块主要识别人脸并标记人脸的位置及大小，其流程图如图19所示。该模块采用每10帧抽取1帧的方法，使用OpenCV对视频分段进行处理，并选用Haar半身特征库及面部特征库对人脸进行识别。人脸识别的过程中，会有一个滑动窗口在待检索图片中逐行扫描，对每一个扫描到的位置进行特征计算，并通过计算结果判断该特征是否满足人脸的标准。该算法设置初始窗口大小为20像素，每次向右/向下移动10像素；如果单次扫描没有扫描到人脸，则逐渐降低步进像素大小为7像素及4像素；如果仍然不能找到，则放弃寻找人脸，输出人脸数为0。这种方法相对比较简单，实现较简便，缺点是计算速度相对较慢，计算精度较低。为以最小的代价提高计算准确度，该算法以上文提及的先逐行扫描半身再在半身范围内扫描人脸的方法识别人脸。由于识别出的半身范围较小，在此范围内进行人脸识别消耗的时间非常少；此方法仅增加了有限的计算时间，却大幅提高了准确率。



**图18 运动区域计算模块流程图**

相似度计算模块主要计算视频分段的总运动量，其流程图如图20所示。与其他分段类型不同，视频资料分段的内容变化幅度较大，时而剧烈时而平静，所以应采用连续取样的方法对相似度进行计算。该算法采用Python图像库PIL（Python Image Library）对每相邻帧进行预处理，先将图片尺寸统一转换为256\*256像素（图片对缩放变换具鲁棒性），然后将图片切为16部分，每部分为64\*64像素。将得到的两个相邻帧的图片集两两计算SSIM值；待该视频分段的SSIM值全部计算完毕之后，计算全部数值的平均值，可以以此作为该视频分段的相似度（即运动剧烈程度）。

以上三个模块计算完毕后，根据得到的三个特征对视频分段进行分类并记录，将分类情况汇总后发送给视频转码模块，由该模块进行转码工作。



**图19 人脸识别模块流程图**



**图20 相似度计算模块流程图**

4.2.4 视频转码、存储模块

上文已经对每一类视频进行了详细的分析，并根据分析结果设定了固定的码率分配表；本模块的任务为根据视频分段的分类，按照码率分配表，将视频分段转码为多个不同清晰度（码率）的视频分段副本，并存储在服务器供客户端取用。待转码工作全部进行完成后，本模块将生成一个MPD文件供客户端使用。本模块的流程图如图21所示。



**图21 视频转码存储模块流程图**

本模块获取到分段及码率分布后，将利用FFMpeg模块对视频进行转码工作。FFMpeg不仅提供了API，还提供了可运行的版本，用户仅需使用命令行即可完成对视频或音频的编码、解码等操作。程序还将分析转码的执行结果，如果执行成功，则将转码文件存放至视频库中，然后进行下一任务；如果失败，则分析失败结果，调整参数后重新进行转码工作。待全部转码工作进行完毕后，该模块可以根据全部视频分段的访问地址、开始时间、结束时间、码率生成一个MPD列表；客户端只需根据此列表即可自由调整视频的码率。

由于存储服务器的任务比较简单，只要可以高效发送视频分段文件即可，此处服务端采用基于Apache的HTTP服务器存放文件及MPD列表；用户获取到MPD列表后从相应的HTTP地址请求适当码率的视频分段即可。

4.2.5 代理服务器模块

代理服务器，是一种可以分发视频的近端服务器。它可以采用多种分发方案：基于TCP、UDP、HTTP均有较成熟的实现方案。由于北航学堂的代理服务器主要为选修同一课程的学生提供方便，不需要具备便携性，所以可以将代理服务器按照服务器的标准进行架设并构建HTTP服务器，将为客户端增加支持代理服务器的设计增加很大便利。

一般情况下，客户端如果需要访问一些资源，则必须连接到远端唯一的服务器；对于出口带宽较小的局域网终端，这种设计会导致客户端始终无法观看到较高质量的视频。一种自然的想法即是将服务器整体迁移至本地，客户端间接通过请求本地服务器上的资源来获取信息。由于客户端访问本地资源的速度一般远快于访问远端服务器资源的速度，所以可以大大提升用户学习课程的QoE。

在客户端适配代理服务器的过程中，需要服务器对客户端进行正确引导，以达到较好效果。如果服务器向客户端提供了一个错误的代理服务器，则不仅不能提升QoE，还会大大提高用户访问资源的延迟甚至访问失败。另外，代理服务器在远端服务器登记后，需要向远端服务器请求资源。适当的算法可以让代理服务器选择需求量较大的视频分段优先进行下载，以在尽可能短的时间内尽快满足客户端的需求。本文尝试设计一种同时具备以上特点的代理服务器，使得客户端可以在较短时间内请求到资源，进而QoE得到提升。

普通的基于C/S框架的应用程序请求网络资源时，会先向指定的服务器发送资源请求信息，然后即可接收到适当的信息。其步骤可以参见图22。而要想使代理服务器生效，则需要将客户端的远端服务器地址替换为代理服务器的地址。一般来说，一个代理服务器从建立至生效要经历如下几个过程：

第一步，代理服务器与远端服务器建立连接，告知代理服务器的性质；

第二步，服务器确认该代理服务器的性质，记录该服务器的IP地址、MAC地址；

第三步，服务器将存储的MOOC视频分发至该代理服务器；

第四步，对每个客户端，服务器确认其所在位置，并判断代理服务器是否可以加快该客户端的网络请求速度；

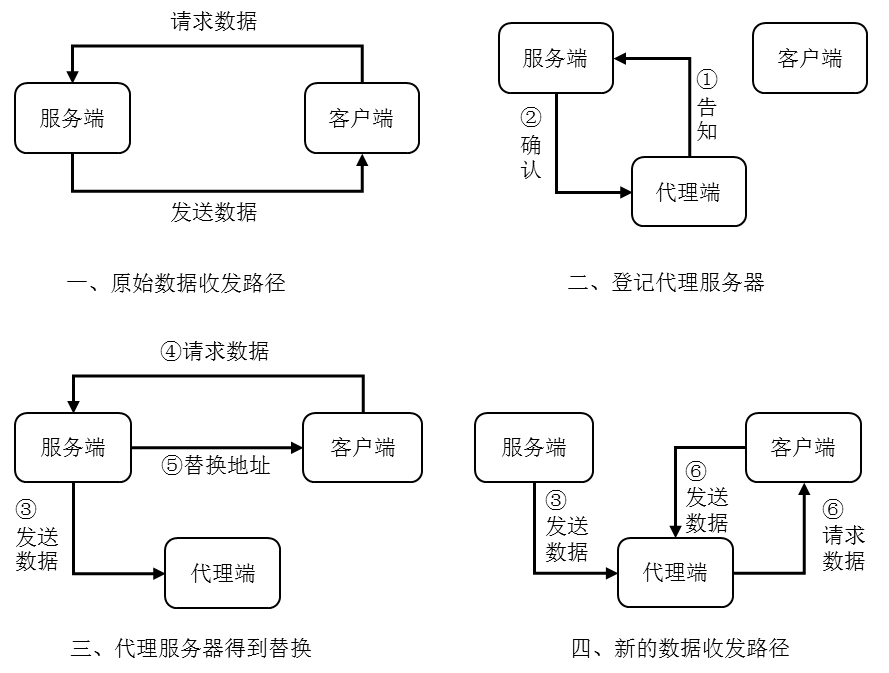
第五步，对合适的客户端，在其观看视频时，服务器将自身的地址替换为代理服务器的地址并告知客户端；

第六步，客户端从代理服务端下载数据，下载速率得到大大加强。

以上六步的过程可以参见图22。

在此处设计中存在两个难点：一方面，如何判断客户端与代理服务器的距离；另一方面，如何判断服务端应该先发送哪一个视频至代理端，可以达到最好的效果。

因为代理服务器的特性，使得它一定是距离客户端较近的服务器，否则设置代理服务器将没有意义。因此，客户端选择代理服务器的一个重要依据即是与各个服务器之间的距离。由于选取代理服务器的工作并不紧急，即使没有代理服务器也可以进行应用收发工作，因此对代理服务器的探测工作可交给客户端进行。客户端只需依次判断各个代理服务器是否存在于本地LAN中、连接速度是否高于远端服务器，即可判断是否存在可供自己使用的代理服务器。



**图22 原始数据路径及代理服务器的生效过程**

代理服务器的一项重要工作是接收并存储远端服务器的MOOC视频。对于接收视频的优先顺序，一般认为应优先发送使用频率更高的MOOC视频。因此，代理服务器在建立之后，应按访问频率逐个接收所有视频课程并存放在本地。需要注意的是，由于大部分用户不会在初次学习MOOC视频时进行大量快进操作，所以对于单个MOOC视频，应按照播放顺序依次下载视频分段。如果某个客户端向代理服务器请求某个未下载的视频或视频分段，有两种处理方法：忽略该请求，指示客户端前往远端服务器获取；立即下载涉及该请求的视频，并在下载完毕后推送给客户端。后者可能会造成客户端产生一定的视频显示延迟，但是其观看效果要好于前者。而无论是代理服务器还是客户端，对存放在远端服务器的视频的访问速度都是相同的；除非远端服务器的出口带宽要好于客户端，这种情况下客户端更不应该自行请求视频分段。然而，如果有多个客户端同时请求不同的视频，可能会造成代理服务器负载过大，从而导致每门课程的下载速度较低。所以，对于已经与某个代理服务器确认连接的客户端，其对代理服务器没有缓存的课程的请求路径，应根据代理服务器的负载决定。

本文采用如下代理服务器及代理服务设计方案：

远端服务器：设置代理服务器数据库，用于存放代理服务器的具体信息，主要包括本地IP地址和MAC地址信息。同时提供接口以发送代理服务器列表数据。

代理服务器：对远端服务器来说，代理服务器是一个客户端，二者的主要交互为发送视频请求、接收视频数据；对客户端来说，代理服务器是一个服务器，它能够以高速发送课程的视频数据。因此，代理服务器应建立视频数据库，以存放代理服务器拥有的课程名称及其他信息，并尽力保持所有课程视频处于最新状态；另外，当客户端请求代理服务器不存在的视频时，代理服务器还应当尽快请求到课程数据以提供给客户端，或者告知客户端返回远端服务器请求数据。

代理服务器应构建课程数据库，并维护课程的下载情况。数据库的各个字段可参见表9。代理服务器将先从远端服务器接收全部课程数据，然后根据用户总体的选课情况，将课程视频按选课热度一一下载至本地供用户接收。

**表9 代理服务器数据库的定义**

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 含义 |
| course\_id | 字符串 | 课程ID |
| course\_title | 字符串 | 课程名称 |
| display\_name | 字符串 | 课程编号 |
| display\_number | 字符串 | 课程学期编号 |
| display\_organization | 字符串 | 课程组织编号 |
| download\_status | 字符串 | 已下载的最后一个章节号 |
| last\_update | 日期 | 本地数据最后更新日期 |
| sections | 字符串 | 课程章节信息 |
| status | 数值 | 0-未下载，1-部分下载，2-下载完毕 |

客户端：设置代理服务器探查模块，接收到远端服务器发来的代理服务器列表后，依次检查是否存在与自身处于同一LAN的代理服务器；如果有，则与远端服务器一起，选取一个访问最快的服务器获取视频。需要注意的是，由于课程信息数据量比较小，所以自始至终该数据都是从远端服务器获取。

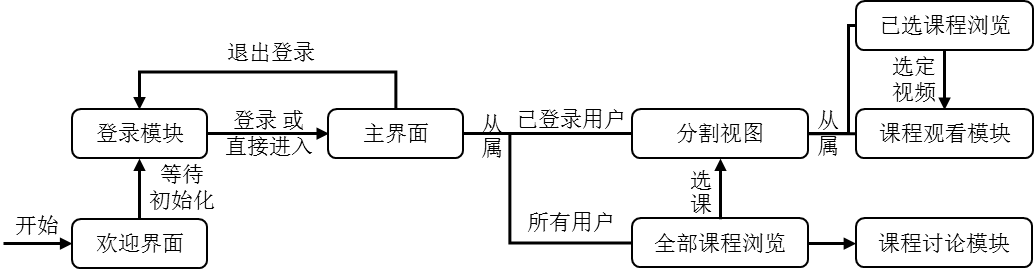
在连接至远端服务器之前，客户端需要判断最近的远端服务器。首先，客户端需要从服务器接收代理服务器列表，然后依次比较每个IP地址的MAC地址是否与表格内相同；如果相同，则该代理服务器与客户端处于同一LAN中。然后通过下载一个较小的数据包，来判断该代理服务器的连接速度是否高于远端服务器。如果较高，则证明该远端服务器具备连接价值，客户端将把自身定义的视频服务器地址改为该代理服务器。同时，当用户请求代理服务器的数据等待时间过长时，客户端会自动切换回远端服务器进行视频下载工作。

4.3 平台客户端设计

本平台的客户端以iOS平台设计，程序代码主要使用Objective-C语言完成。采用iOS平台设计的优点有很多：一方面，iOS平台已经设计好大部分UI控件的动画逻辑及UI动作的事件响应，用户只需调用相应的接口即可实现大部分功能；一方面，苹果公司对功能库的更新是非常频繁的：随着版本的更新，苹果公司逐渐加入了对JSON、识别二维码、高级网络功能等的支持，省去了引用并编译第三方API库的麻烦；一方面，iOS的开发环境XCode对开发者更加友好，不仅有完善的编码提示功能，还能在编译时协助用户完成get/set函数生成、内存管理等，更有完善的代码管理模块。综上所述，采用iOS平台设计该功能模块有助于更好地完成客户端的设计工作。

4.3.1 客户端主要结构及主体界面

本客户端主要包含如下几个模块：数据收发模块、登录模块、全部课程浏览模块、用户已选课程浏览模块、课程讨论模块、视频观看模块。其中，数据收发模块主要负责发送请求、接收数据，将数据本地化以减少网络请求；登录模块负责处理登录请求；全部课程浏览模块可以查看平台提供的全部课程及包括开课时间、选课介绍、课程序号、课程简介在内的课程信息；用户已选课程浏览模块负责提供已选课程的信息及可观看视频列表；视频观看模块负责提供视频以供用户观看，模块核心是基于DASH的视频播放器；课程讨论模块主要负责处理用户与其他用户或者课程讲师的交互。客户端所有模块的结构及关系如图23所示，该结构采用Objective-C提供的故事板（Storyboard）功能设计并由Objective-C进行管理。所有界面如未说明，均采用Objective-C的界面类UIView及界面控制类UIViewController进行功能实现，处于界面之中的控件位置由Constraints及NSLayoutView根据设定进行自动调整及适配。所有模块的具体构造将在下文一一叙述。



**图23 在线学习平台客户端结构及关系图**

4.3.2 数据收发模块设计方案

本模块将统筹负责所有其他模块的网络请求及回传接收到的数据。在线学习平台有大量的通信操作，包括验证登录信息、请求课程列表、课程详细信息、课程章节列表、视频地址等等。如果其他模块独立设计数据请求及接收功能，将产生大量冗余代码。因此，本文认为，将所有涉及网络功能的代码单独封装，不仅易于修改、可读性强，还增强了代码的逻辑性。

本文在设计初期曾经调研了多种网络开源模块，例如AFNetworking、ASIHTTPRequest等，并将其与Objective-C原生的NSURLSession模块进行比较，发现大多数开源模块都是建立在NSURLSession基础上的，故为了精简APP大小，本文决定不对网络库进行集成，仅使用objective-c自带的网络通信模块NSURLSession进行网络操作。

由于网络模块的复用特性，本文认为，使用单例模式（Singleton）[27]对其进行封装，不仅可以省去重复分配、释放模块的实例造成的资源损耗，还让模块可以对网络请求进行统筹管理，增强了该模块对网络操作的可控性。

本文将网络操作封装为MOOCConnection类，其工作流程简述如下：在该类第一次被调用时，初始化该类并创建一个实例；一旦有网络请求，该实例通过NSURLSessionDataTask类对网络操作进行初始化并将该网络请求发送至服务端；当接收到服务端的响应时，该模块根据数据的特性对其进行处理，并将可本地化的数据保存在本地数据库；收到的数据被处理完毕后，将处理结果广播给特定对象。此处的广播，是采用观察者模式[27]设计的功能模块NSNotification，它通过NSNotificationCenter进行统筹管理。一个观察者（Observer）可以在NSNotificationCenter内注册，当有广播被发送，符合广播条件的观察者即可捕获该条广播进行处理。这种信息处理方式虽然具有不可控性，但是发送网络请求的功能模块可以不必阻塞等待，并且可以在任何时间处理收到的数据，灵活方便。

上文提及，将获取到的网络数据进行本地化存储，可以使客户端先从本地数据库中提取信息并与服务器比对信息的有效性，可以有效地减少数据下载量。经过调研，Objective-C原生的CoreData功能可以有效地处理这种本地化请求。CoreData能够将Objective-C对象转化成数据，保存在轻量化的SQLite数据库文件中，也能够将保存在数据库中的数据还原成Objective-C对象，可以很好地满足对象-关系映射（ORM）的需求。这个模块虽然在iOS5之后才出现，但是不影响这个模块的强大；类似于Hibernate框架，数据操作时，该模块只需要调用相关的成员函数即可完成数据库操作，操作便捷，易于使用。

本文需要使用两个数据库：登录用户信息数据库，键名实体化为MOOCDatabaseUser类；课程信息数据库，键名实体化为MOOCDatabaseCourse类。对这两个数据库的处理操作封装为MOOCCourseData类，该类实现了对每类数据库的基本操作：插入数据（集）、删除数据（集）、修改数据（集）、获取数据（集）。两个数据库的键值设计可以见表10和表11。需要说明的是，由于数据库委托给CoreData管理，故本数据库的主键由CoreData管理，本表中不出现主键。

**表10 数据库MOOCDatabaseUser的定义**

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 含义 |
| language\_code | 字符串 | 用户使用语言 |
| last\_update | 日期 | 本地数据最后更新日期 |
| mail\_address | 字符串 | 用户邮箱地址 |
| user\_full\_name | 字符串 | 用户姓名 |
| user\_name | 字符串 | 用户名 |

4.3.3 欢迎界面设计

欢迎界面的作用是为了初始化客户端以及确认客户端到服务器的连接并从服务器获取Token。在用户等待初始化的过程中，用户可以看到北航学堂的主要信息，点击“关于我们”将弹出北航学堂的“关于”网页。本文将欢迎界面封装为MOOCWelcomeView，继承于UIViewController，该界面的样式可见图24。

**表11 数据库MOOCDatabaseCourse的定义**

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 含义 |
| about | 字符串 | 课程简介 |
| course\_end | 日期 | 课程结束日期 |
| course\_id | 字符串 | 课程ID |
| course\_image | 字符串 | 课程图片（本地存放地址） |
| course\_image\_url | 字符串 | 课程图片地址 |
| course\_start | 日期 | 课程开始日期 |
| course\_title | 字符串 | 课程名称 |
| display\_name | 字符串 | 课程编号 |
| display\_number | 字符串 | 课程学期编号 |
| display\_organization | 字符串 | 课程组织编号 |
| enrollment\_date | 日期 | 选课日期（如果可用） |
| enrollment\_end | 日期 | 选课截止日期 |
| enrollment\_start | 日期 | 选课开始日期 |
| is\_full | 布尔值 | 课程是否已满 |
| last\_update | 日期 | 本地数据最后更新日期 |
| registered | 布尔值 | 是否已注册 |
| sections | 字符串 | 课程章节信息 |



**图24 在线学习平台客户端欢迎界面**

4.3.4 登录界面设计及模块实现

初始化完毕之后，客户端跳转到登陆界面。登陆界面允许用户使用用户名和密码登录、注册新用户或者浏览课程。需要注意的是，如果用户选择浏览课程，则用户将无法进入已选课程浏览及课程观看模块中，只能查看北航学堂全部课程的课程信息。登录时，用户将输入用户名及密码；输入完成后，客户端将把用户名及密码以POST的形式发送给服务器以验证其正确性。此处发送的POST请求中需要设置的变量将在表12中给出。待请求发送完毕，服务器将以json形式返回登录结果；如果登录成功，则客户端进入主界面并保存当前用户的信息至本地，否则提示用户登录失败原因。此处获取的用户信息与表10相同。本文将登录界面封装为MOOCLoginView，继承于UIViewController，界面样式可见图25。

**表12 登录请求的变量设置**

|  |  |
| --- | --- |
| 名称 | 值 |
| Method | POST |
| Body | email=（用户名）&password=（密码） |
| Header/content-type | application/x-www-form-urlencoded |
| X-CSRFToken | （初始化时获取的令牌值） |



**图25 在线学习平台客户端登录界面**

4.3.5 主界面设计

主界面的功能是为用户管理子功能列表以及在上方标题栏内提供退出登录、帮助等功能的按钮，从属于主界面的子功能主要包括：全部课程浏览模块、包含于分割视图的已选课程模块和视频观看模块。由于iOS对UI的设计进行了深度封装，在设计该界面时用户只需要在故事板里标明从属关系即可，主界面会自动把子功能以标签的形式显示在下方。本文将主界面模块封装为MOOCMainView类，继承于UITabBarController。

4.3.6 全部课程浏览模块设计

全部课程浏览模块的作用是显示北航学堂提供的所有课程。在用户第一次进入该模块时，客户端会先向服务器请求全部课程的简要信息，包括课程名称、课程图片、课程ID等等，请求到数据后将其存放至本地。一般情况下，北航学堂的课程不是很多，每个课程的图片、名称、ID仅占据很小的空间，故一次性请求全部课程的信息可以减少网络请求，使用户不需要在看更多课程时再次请求其他课程信息。如果用户不是第一次进入该模块，则需要先与服务端比对本地课程列表信息的有效性，当有效性得到确认时再予以显示，否则需要更新所有课程的信息。

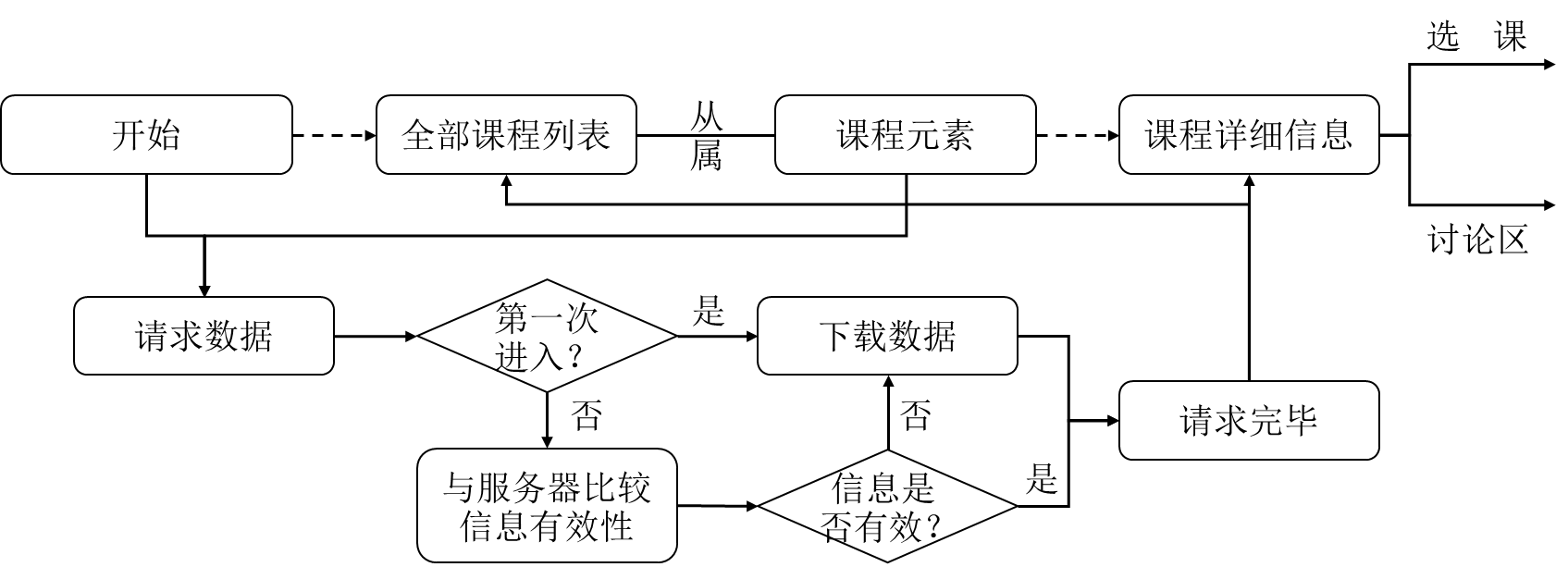
本模块采用基于瀑布流的课程列表设计方案，界面可参见图26左图。瀑布流（瀑布流式布局），是一种非常受欢迎的页面布局方案，视觉表现为由参差不齐的内容块堆叠而成的多栏布局。随着页面向下滚动，这种布局还会不断提前加载内容块，并将已下载好的内容块附加至当前页面的尾部。这种布局方式具有用户操作简单、界面漂亮、可展示内容多的特点，是当今很流行的布局方案之一。本文将该模块封装为MOOCCourseView类，继承于UICollectionViewController类；对于瀑布流内的每个元素，本文封装为MOOCCourseViewCell类，继承于UICollectionViewCell类，由需要加载至瀑布流的元素进行复用。

当用户点击到某一门课程后，客户端将显示该课程的详细信息，该界面如图26右图。因为详细信息数据量比较大，所以不能在请求课程列表的时候同时对详细信息进行请求，以避免用户等待时间过长。此处请求的数据种类与表11基本相同。与课程列表的设计逻辑类似，如果是第一次查看该课程的详细信息，需要先向服务端进行请求，等待服务器发送信息后显示并存至本地；如果不是第一次查看则需要比对本地课程信息的有效性。同时，该模块提供选课操作，如果当前课程没有满员，则用户可以选修该课程；用户点击选课按钮后，客户端向服务器发送选课请求，待服务端确认后，客户端跳转至已选课程页面。课程详细信息页面提供讨论区的入口，用户可以点击标题栏的讨论按钮进入讨论区，与学习课程的其他同学及课程讲师交流。本文将课程详细信息界面封装为MOOCCourseDetailView类，继承于UIViewController类。本模块的工作流程图见图15。

**图26 在线学习平台客户端界面**

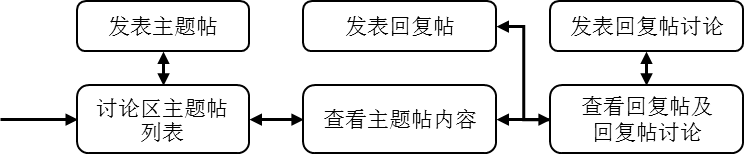
**左图：所有课程界面（瀑布流） 右图：课程详细信息界面**



**图27 全部课程信息模块结构及关系**

4.3.7 课程讨论模块设计

课程讨论模块主要用于用户之间及用户与课程讲师的交互行为。首先本文需要介绍一下北航学堂讨论区的主要结构。北航学堂讨论区主要采用三级结构构成：基础一层为主题帖；用户可以在主题帖下发回复帖；可以在回复帖下发讨论帖，针对该回复帖的内容进行讨论。本文在实现讨论区功能时也采用了这种结构；讨论区的主结构及讨论区子模块之间的关系见图28。



**图28 课程讨论模块结构及关系图**

帖子列表主要采用iOS的TableView进行实现，它是一种苹果设计的一维表视图布局方案，能在使用时根据用户的移动位置动态加载并复用列表，所以即使数据量非常庞大，该模块对数据的处理也可以非常迅速高效。表视图的表内元素可以自由定义，元素内容可以是静态的也可以是动态的，每个内容的具体布局也可以根据需要自由选择，是一种非常灵活的布局方案。由于主题贴的内容是支持RTF（富文本格式）的，可能包括图片、超链接、文字样式等多种信息在内，所以主题贴需要单独显示在一个页面；讨论区主题帖和主题帖的回复帖采用TableView实现，其中单个回复帖和该回复帖的讨论将显示在TableView的同一个Cell中。用户在主题帖列表页面可以进入发布主题帖页面，在查看主题帖页面和查看回复帖页面可以进入发布回复帖页面，在查看回复帖页面可以进入讨论回复帖页面。

本文将讨论区主题帖列表封装为MOOCForumDiscussion类，将查看主题贴页面封装为MOOCForumDetails类，将查看回复帖及回复帖讨论页面封装为MOOCCommentDetails类；发表主题帖、主题贴回复、讨论主题贴回复的页面被封装为MOOCAddThread类、MOOCReplyThread类、MOOCReplyComment类。其中MOOCForumDiscussion类及MOOCForumDetails类继承于UITableViewController类，其余均继承于UIViewController类。

4.3.8 分割视图界面设计

分割视图（SplitView）是iOS系统独有的一种界面布局方案，它由两个视图构成：列表视图和详细视图。分割视图可以在屏幕的一侧显示列表视图，列表内包含多种供用户选择的选项；当用户选择表中的元素后，屏幕上的详细视图将显示与点击的选项相关的详细信息。在设备处于侧置状态（Landscape Orientation）下，列表视图将可以隐藏或者固定在屏幕一侧，另一侧为详细视图；在设备处于正置（Portrait Orientation）或倒置（Upside Down Orientation）状态下，列表视图将默认不予以显示，标题栏的按钮可以将其唤出，详细视图将填满整个屏幕。由于这种布局方案需要一定的空间，所以这种视图只能用于iPad。

本文将分割视图中的列表视图定义为已选课程列表，将详细视图定义为视频观看界面。而本文将分割视图的控制模块封装为MOOCMyCourseSplitView类，继承于UISplitViewController类。

4.3.9 已选课程浏览模块设计

本模块是从属于分割视图界面的子模块，主要供用户查看已选的课程、选择想要观看的视频。由于MOOC的课程采用四级列表设计，所以此处需要使用五级列表，分别用于显示已选课程列表、课程章列表、每章的分节列表、每节的小节列表、每小节的元素（视频）列表。因为一般情况下，每小节只存在一个视频，存在两个视频的情况不多，故本文将后两个列表合并；如果遇到某小节有多个视频的情况，则客户端将在小节列表中列出所有可供观看的视频。用户点击每个列表的某项之后均将跳转至对应的下一级列表，当用户点击小节列表的视频后，分割视图中的视频观看界面将显示用户所选的视频。

由于此处整合代码需要整合界面，涉及工作量较大，而不如分别实现每个列表，故本文将已选课程列表、课程章列表、每章的分节列表、每节的小节列表分别封装为MOOCMyCourseList、MOOCChapterList、MOOCSectionList、MOOCPositionList类，均继承于UITableViewController类。

4.3.10 视频观看界面设计

本模块是从属于分割视图界面的子模块，主要用于显示用户选择的视频。本文将本模块封装为MOOCMyCourseShow类，继承于UIViewController类，类内代码主要用于接收已选课程浏览模块发送的视频地址、控制播放器的各种行为及调整播放器的样式。

4.4 平台视频播放器设计

本平台的视频播放器主要用于播放DASH视频。由于iOS原生的MPMoviePlayer只支持苹果HLS协议，所以本文需要自行实现一种可以用于上文所述DASH视频的播放器。现有用于解析音视频的API主要为FFMpeg开源库，本文在编程实现视频播放器时主要使用该开源库中的libavcodec视频编解码库，它包含了非常先进的音视频编解码算法，具备较高可移植性和较高的编解码质量。如图29为客户端播放视频的画面。



**图29 在线学习平台客户端播放器界面**

4.4.1 视频地址解析

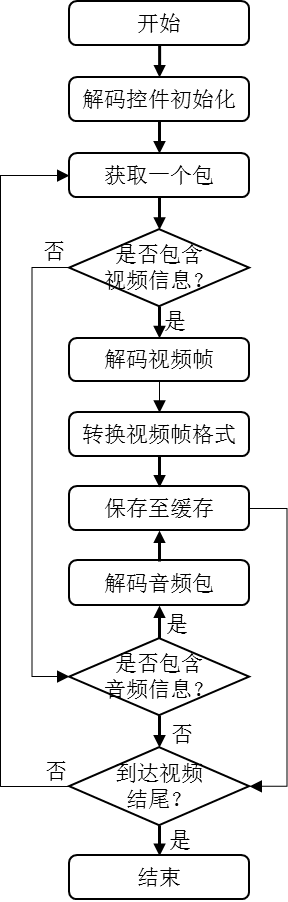
当播放器获取到播放地址后，需要对其进行解析。DASH规定在MPD里需要将视频描述为分段的集合，每个分段的具体码率分配也需要在其中写明。MPD会以文件形式保存在服务器中，播放器在播放视频时先根据地址将MPD文件下载到本地，再对其进行解析。解析时，播放器会将该视频的分段信息、视频码率信息等有用的信息存储到本地，播放时根据当前需要，依次下载对应码率的视频分段。

4.4.2 视频解码

当播放器需要播放某一视频分段时，播放器需要将其解码后逐帧显示出来。视频播放的原理，是将一动态的过程切分为图像，并以极快的速度连续播放，以使人看起来像是连续的场景。本平台的播放器需要完成这一过程。

任意视频分段均包含一系列信息，以供视频播放器对其进行解码。首先，视频文件的头部包含该视频的编码信息及播放信息，包括视频长度、帧速率、分辨率等等。然后，视频文件包含一系列的流；通常情况下，视频文件至少包含视频流及音频流，部分视频包含多个字幕流以便用户观看与自己母语不同的视频。而上文提到的视频流，其基本元素为帧，视频编码算法会将部分帧进行简化存储以提高压缩率。因此，播放器在播放视频时需要通过特定的解码器将其解码出来。常见的视频编解码算法有H.26X系列、MPEG系列等等。

在代码角度，基本元素是以包（Packet）的形式封装在流里面的，而每个流的包是混合存放的，因此，当代码接收到一个包之后，其内容是不确定的，必须通过代码循环处理。处理时，需要将包内的数据转换为特定格式的图片，才能够显示出来。本部分的流程图如图30所示。

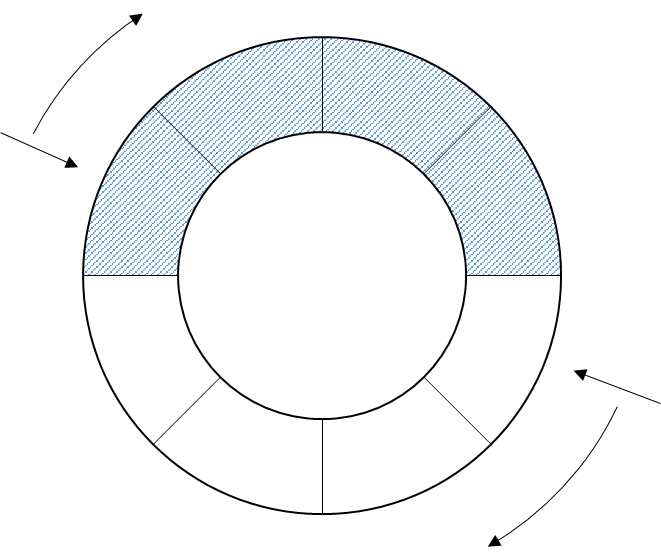


**图30 视频/音频解码流程图**

当播放器获取到一段视频后，首先调用avformat\_open\_input()打开文件，并调用avformat\_find\_stream\_info()分析视频内流的数量及信息，从在这些流中查找出视频流并调用avcodec\_find\_decoder()函数判断视频流的编码格式，待获取到相关信息后调用avcodec\_open2()打开对应的编解码器以对这个视频流进行解码。视频编解码器解码得到的视频帧最初是AVFrame格式，播放器需要先调用av\_frame\_alloc()对缓冲区进行初始化，调用avpicture\_get\_size()获取缓冲区的大小并调用av\_malloc()对该缓冲区进行初始化。最终调用avpicture\_fill()将缓冲区分配给AVFrame类的实体化变量用于缓冲视频帧。至此，视频的读取工作已经准备完毕。

读取时，需要调用av\_read\_frame()循环读取内容包，如果该包的内容是需要读取的视频帧，则使用avcodec\_decode\_video2()对该帧进行解码，并储存至AVFrame类型的缓冲区中。然而，iOS并不能直接处理该格式的视频帧，需要将其转换为iOS可读的格式。此处可以采用iOS平台比较流行的OpenGL渲染图像方案，先将AVFrame格式的视频帧转换为YUV格式，然后将该帧保存在特定的待播放缓冲区中，直至需要时交付给OpenGL，即可渲染出图像。这种方法的效率较高，对性能的要求相对较小，是一种良好的渲染方案。

本客户端采用环形缓冲区存放视频帧，如图31所示。在初始化时，环形缓冲区申请一段连续内存空间作为存放数据的空间，并初始化两个判断缓冲区存储状况的头指针及尾指针。当提取数据时，环形缓冲区将尾指针指向的元素推送出去，尾指针顺时针移位一个单位；存放数据时，缓冲区在头指针指向的空位存入新数据，并将头指针顺时针移位一个单位。当二者重合时为空，当头指针比尾指针的位置小1时为满。这种缓冲区方案可以提高内存的重复利用率，有效地避免多次请求及释放空间。



**图31 环形缓冲区示例**

4.4.3 音频解码

数字音频流是一种包含大量采样点的流，每个采样点包括了采样时刻的声音波形信息。在算法将声音还原时，只需要根据每个采样点的声音及采样频率即可将声音还原。一般情况下，音频流每秒的采样点个数为每秒22050个或44100个。如果音频流需要对多声道音频进行编解码，则单个音频流内会包含多重采样信息。上文提到，基本元素是以包为单位的。音频信息则被拆分成每秒40个左右的包，每个包都包含20～25毫秒的音频信息。当我们调用FFMpeg模块对音频进行解码时，我们并不能预先知道每个包内的采样信息的声道数；一般情况下，FFMpeg模块不会将不同声道的音频信息拆开发送，所以在处理音频信息时不需要分开处理每个声道。

播放器需要先对加载的视频进行首先调用avformat\_open\_input()打开文件，并调用avformat\_find\_stream\_info()分析视频内流的数量及信息，从在这些流中查找出音频流并调用avcodec\_find\_decoder()函数判断音频流的编码格式，待获取到相关信息后调用avcodec\_open2()打开对应的编解码器以对这个音频进行解码。解码前，需要根据音频的特征设置音频格式、采样率、声道、频率等参数。

读取时，同样需要调用av\_read\_frame()循环读取内容包，如果该包的内容是需要读取的音频，则将获取到的音频数据保存至缓冲区并等待渲染。从包内解码得到的数据为二进制格式，需要iOS对其进行处理然后播放。本文在此采用iOS的CoreAudio模块实现声音的播放操作。此处声音的解码流程可参见图30。

4.4.4 时间轴控制及音视频同步

无论是音频流还是视频流，播放时都要在固定的时间内显示规定的内容，以正常显示音视频信息。音频播放过快会出现杂音，播放过慢会出现卡顿；视频播放速度过快或者过慢都会影响用户体验。因此，选取适当的时间轴对正常播放视频非常重要。由于音频与视频使用不同的播放模块，二者的播放速度必须保持相同，因此如何同步时间轴是播放器设计面临的第二个问题。

同步时间轴有三种方法：分别同步二者至电脑时钟；同步音频到视频；同步视频到音频。由于iOS中CoreAudio模块及音频本身的特性，在规定的时间内播放规定的音频采样数比较容易；视频的编解码方式导致解码得到帧的顺序与真实的播放顺序可能略有不同，如果将视频同步到音频，则可以较为简便地根据音频位置选择正确的视频帧。本文采用同步视频到音频的解决方案。实际设计中，音频的解码及播放速度良好；解码视频需要消耗大量资源，导致部分设备解码视频帧的速度慢于播放的速度，例如iPad mini，此时则会出现卡顿情况。这种情况下，为了正常播放视频，则需要将部分视频帧甩掉；采用视频同步到音频的方式能够方便地甩掉过时的视频帧（跳帧）。

4.5 小结

本章叙述了基于MOOC的在线学习系统的具体设计和实现。本系统主要以服务端/客户端架构（C/S）实现，服务端主要处理并存储MOOC视频，客户端则实现了一个完整的基于MOOC的移动学习平台iOS端。该移动学习平台可以查看课程、选修课程、学习课程、参与课程讨论，界面简约，易于使用。其中学习课程功能，本文单独实现了一个基于DASH的视频播放器，可以在性能较强的设备上正常播放视频，并自动适应当前带宽，可以提供较强的用户体验。另外，本系统设计了一种代理服务器，它可以在局域网内起到替代远端服务器的作用，可以大大增强用户体验。

第五章 实验与数据分析

本章主要讨论上文所述服务端的分类算法准确率以及客户端的码率调节算法性能及综合用户体验。本文通过三部分实验，对以上各个优化方案及系统进行有效性、合理性论证，并得出相应的结论。

5.1 视频切分算法准确度

本实验主要用于评估视频切分算法的分类准确度。该实验先对每个MOOC视频进行人工分类，然后对该视频进行切分、分类，并将机器分类结果与人工结果进行比较，进而获得分类正确率。每个切分正确、分类正确的分段记作一次正确分类。根据分类结果，本文可以继续对算法进行适当的改进。

由于MOOC是一种新兴的概念，网络上并没有针对MOOC的数据集，所以本文需要自行组织一份科学、合理的数据集。实验针对MOOC视频的特点，分别选取了PPT分段较多、讲师/混合分段较多及视频资料分段较多的视频9个，分别从4门MOOC课程中选出，总计时长3449.92秒，包含讲师分段57个、混合分段46个、PPT分段30个、视频资料分段29个。需要注意的是，某些第二章所述的特殊情形，本实验将把这些分段舍弃，不对其进行统计。

对这9个MOOC视频的分类统计结果如表13所示。

**表13 分类实验正确识别数统计结果**

|  |  |  |
| --- | --- | --- |
|  | 个数 | 百分比 |
| 分段数总计 | 162 | 100.000% |
| 正确识别数 | 151 | 93.210% |
| 错误识别数 | 11 | 6.790% |

从表14可以看到，算法的分段正确识别数量占总数的93.2%、正确识别时长占总时长的97.6%，具有较好的识别效果。对于详细的视频分类结果，可以看到，讲师视频的正确识别个数占比最低，为91.2%。

**表14 分类实验正确识别数分类统计结果**

|  |  |  |  |
| --- | --- | --- | --- |
| 视频分类 | 真实值 | 正确识别值 | 正确率 |
| 讲师分段 | 57 | 52 | 91.228% |
| 混合分段 | 46 | 43 | 93.478% |
| PPT分段 | 30 | 29 | 96.667% |
| 视频资料分段 | 29 | 27 | 93.103% |

在表15中，本文列出了错误分类的11个分段的具体误识别情形。对错误识别样例的分析，我们得出，导致错误识别的情形大致可以分为以下几个原因。

**表15 错误识别的具体样例**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 识别值 | 真实值 | 时长 |
| 1 | PPT分段 | 视频资料分段 | 2.04 |
| 2 | PPT分段 | 混合分段 | 17.92 |
| 3 | 视频资料分段 | 混合分段 | 11.16 |
| 4 | 视频资料分段 | 讲师分段 | 1.56 |
| 5 | PPT分段 | 混合分段 | 5.88 |
| 6 | 混合分段 | 讲师分段 | 3.36 |
| 7 | 视频资料分段 | 讲师分段 | 7.88 |
| 8 | 视频资料分段 | 讲师分段 | 3.64 |
| 9 | 混合分段 | 讲师分段 | 23.96 |
| 10 | 混合分段 | PPT分段 | 0.4 |
| 11 | PPT分段 | 视频资料分段 | 5.08 |

第一，讲师的脸部不够清晰及讲师的脸部一直背对镜头，导致讲师面部无法被识别，造成错误识别的情形。在错误识别样例中，有相当多的讲师分段被误识别为其他分类；部分混合分段被识别为PPT分段或者视频资料分段也是这个原因。如果讲师的面部无法识别出来，由于对人体其他部位的识别均有较大难度，所以几乎不存在改进方法，也是本算法过于依赖人脸的一个短板。

第二，算法对讲师的位置判断有误差，导致讲师分段中讲师的特性符合混合分段中讲师的特性，造成误分类的情形。部分混合分段被误识别为讲师视频的情形也与这个原因有关。这种情形，可以通过对增强区分讲师视频及混合视频的条件来解决。

第三，视频资料分段与PPT分段的误识别情形也经常出现。这是由于某些视频资料分段的运动量过小或者PPT分段的背景在动导致。本文认为，尽管可以通过对视频资料分段及PPT分段进行更精确的条件设置来避免此类误分段，但是这种分类方式有其客观合理性。视频资料分段与PPT分段的实际区别就在于运动量，如果PPT视频的运动量非常大，则将其分类至视频资料分段并无不妥；同理，如果视频资料分段长时间静止不动的话，将其分类至PPT分段也不是没有依据。严谨起见，本文将这些分类情形依然定义为误分类。

第四，PPT分段或视频资料分段中包含人脸。大部分包含人脸的PPT分段或视频资料分段均可以被剔除，但是如果这些不包含讲师的分段长时间出现连续、较大的人脸或画像，且人脸或画像在移动（尽管肉眼可能无法分辨），则该分段将有很大概率被分类为包含人脸的混合分段或讲师分段。对于PPT分段，由于现有的算法无法分辨画像与人脸的区别，所以这种情况难以改进；其他情况下的误识别，可以通过调整参数来减少。对于视频资料分段，可以通过判断人脸的位移情况、出现时长来与讲师分段区分。

综上所述，本算法仍有一定的改进空间；但是现有的准确率证明，该算法已经可以用作对MOOC视频进行分类，并具备一定的水平。

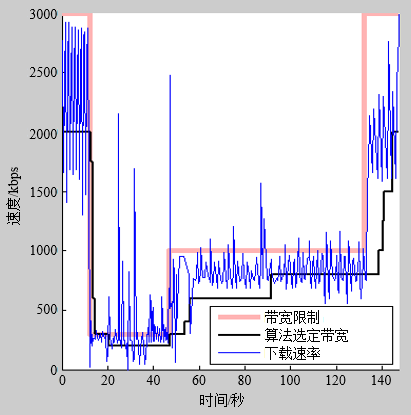
5.2 码率调节算法实验

本实验主要用于评估带宽处于变动状态下时，带宽选择算法的性能。

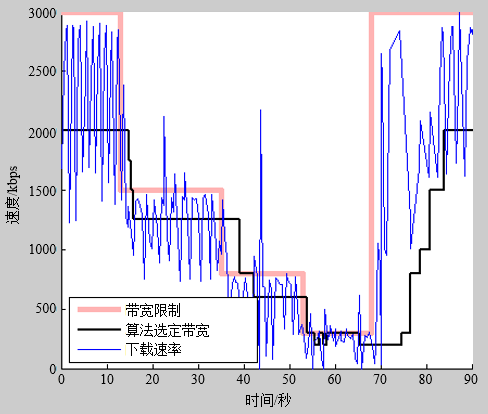
本实验将不在iOS客户端上进行；本文作者单独编写一个基于PC的视频下载软件，用于下载视频，该软件采用文中所述的带宽调节算法。服务器采用基于Apache的HTTP服务器存放视频分段数据。服务端与客户端之间采用损伤仪控制带宽的大小。为了体现该算法的真实效果，本文在切分视频时，不能采用上文的视频切分算法对其进行分类，否则将无法精确体现该算法的特性。此处本文选取了一段长为3分钟的视频，从始至终运动量均很大，平均码率为1831kbps，每个视频分段采用12级码率分布，从低到高依次为100、150、200、300、400、600、800、1000、1250、1500、1750、2000kbps。采用这种码率设置对视频进行转码后，将视频分段存储至服务器。需要说明的是，本视频中所有视频分段的类型均设置为视频资料分段，以观察极端情况下该算法的性质。

在进行带宽稳定性实验时，分别考虑两种情况：带宽缓降速升、缓升速降，在带宽选择算法运行稳定后缓慢/迅速降低损伤仪的带宽设定，一段时间后再缓慢/迅速恢复带宽设定，以判断带宽选择算法的灵敏度及稳定性。故此处设置两组实验：第一组对损伤仪的带宽限制方案为3000kbps(10s)→300kbps(40s)→1000kbps(85s) →3000kbps(至结束)，第二组的限制方案为3000kbps(10s)→1500kbps(25s)→750kbps(28s)→250kbps(25s) →3000kbps(至结束)，记录算法选择的带宽及下载速率，并分析图像。

实验结果见图32和图33。由两个实验结果可见，当带宽减小时，本算法非常快地向下调节视频码率至低于损伤仪的设定值。这种行为符合预期，可以有效地避免视频加载过慢，导致播放时出现卡顿现象。当带宽增大时，本算法向上的调节速度比较慢，通常从增大带宽后5-10秒开始向上调节，整个调节过程需要至少10秒左右才能完成。这种行为也符合预期，由于用户并不想看到视频质量的频繁变动，提高视频码率会明显增加视频质量抖动的风险。因此，谨慎地向上调节带宽可以避免用户体验的大幅下降。



**图32 码率调节算法实验结果一**



**图33 码率调节算法实验结果二**

综上所述，该算法对带宽减小的灵敏度非常高，对带宽增大的灵敏度较低，符合算法设计之初的预期，能够较好地提高用户体验，避免用户体验的大幅下降。

5.3 在线学习平台用户体验实验

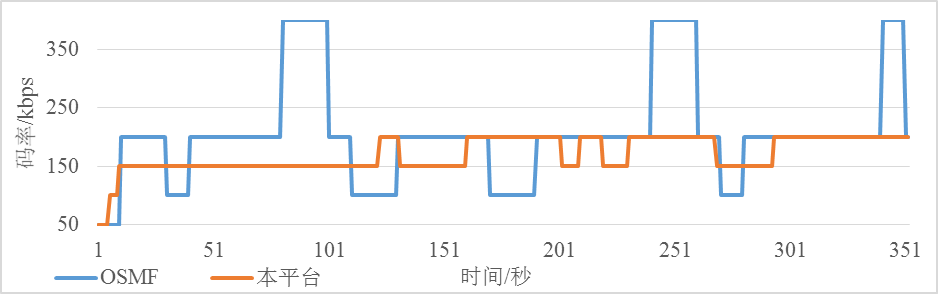
本实验主要用于评估带宽充足及带宽不足的情况下，在线学习平台的客观用户体验。

本文选取一段长5分52秒的MOOC视频作为实验视频，该视频码率为1013kbps。实验中，本文分别使用以Adobe公司出品的开源媒体框架OSMF[24]（Open Source Media Framework）算法为代表的传统切分方法和本文所述的视频切分算法对该视频进行处理。根据该视频的特点，在采用传统切分方案时，我们将其划分为50，100，200，400，700，1000kbps六档。在进行切分、转码、存储操作之后，将得到的两组处理后的视频分别存放在服务器中；实验时，使用客户端同时对两段视频进行播放，并记录视频的码率以供分析。同时，客户端实时记录播放视频与原视频的PSNR值，以比较所有测试视频的质量区别。客户端与服务器之间采用损伤仪限制带宽。在此，本文进行两组实验，分别考察带宽不足及带宽充足的情形。第一组实验将损伤仪的带宽限制设定为250kbps，第二组实验设定为3000kbps。

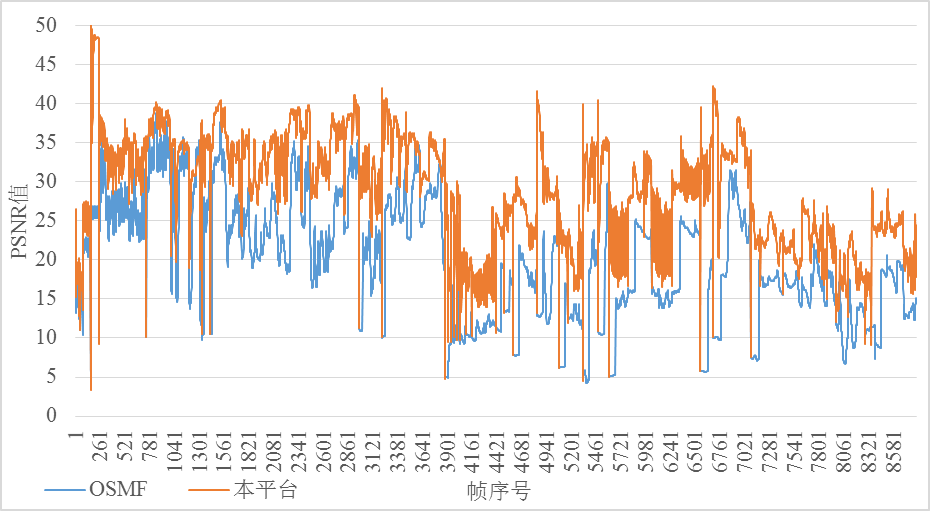
第一组实验结果见图34至图36，第二组实验结果见图37至图39。从图34中可以看到，在带宽不足的情形下，在相同内容的播放区间内，本文设计的平台可以较好地控制带宽波动，并保持恒定的带宽。图36为播放时的视频截图，结合图35进行分析，可以认为，本文设计的平台的视频质量要好于传统算法。并且，从图38和图39中可以看到，在带宽充足的情形下，本文设计的平台提供的视频质量与传统算法基本一致。同时可以从图37中看到，本文设计的平台减少了相当一部分的带宽开销。

5.4 小结

通过三组实验，本章对上文提出的各种算法及优化方案进行验证。对于视频切分算法，本文通过分析大量视频的切分结果来判断分类的准确率，得到该算法准确率完全可以满足一般使用的结论；对于码率调节算法，本文通过限制带宽研究其特性，得到该算法特性与预期相符的结论；对于视频客户端的用户体验实验，本文通过比较客观质量的变化以及带宽的消耗，得出结论：无论是从带宽消耗角度还是视频质量角度，本系统都要优于采用普通算法的系统。综上所述，本论文研究的基于MOOC的在线学习平台完全符合预期，并在DASH技术上做出了较大的改进，与MOOC视频配合效果较好。



**图34 250kbps实验：码率选择图**

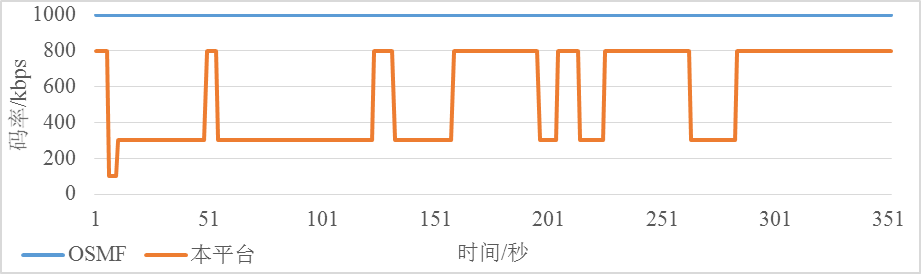


**图35 250kbps实验：客观质量图**

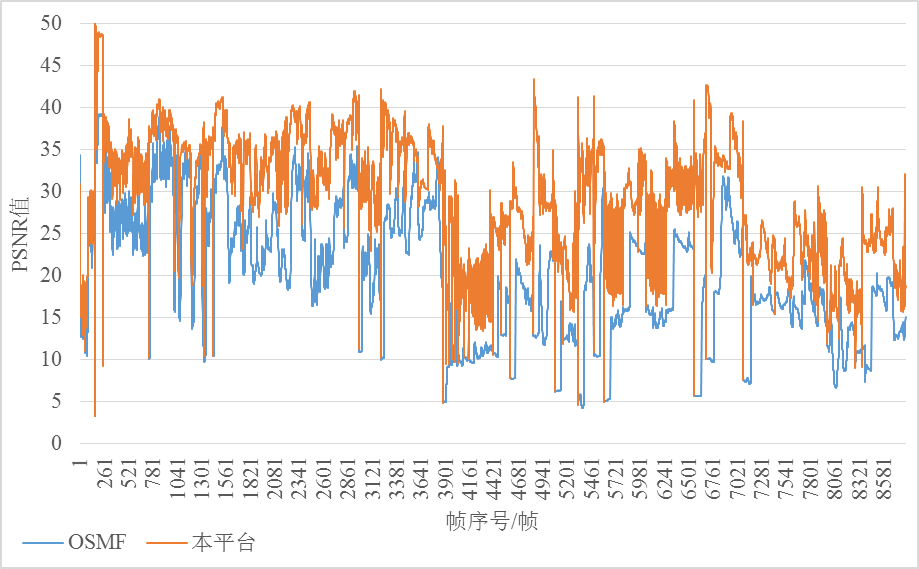


**图36 250kbps实验：视频截图**

**左半部分：视频截图 右半部分：视频局部放大图 上半部分：传统算法 下半部分：本平台**



**图37 3000kbps实验：码率选择图**



**图38 3000kbps实验：客观质量图**



**图39 3000kbps实验：视频截图**

**左半部分：视频截图 右半部分：视频局部放大图 上半部分：传统算法 下半部分：本平台**

总结与展望

一、总结

本论文实现了一个面向MOOC的移动视频学习平台。该平台利用DASH技术构建；DASH的多个模块均未针对MOOC进行优化。本文对DASH的切分模块、码率调节模块、局域网内共享模块进行了改进，并将这些改进后的算法写入了这个移动学习平台，使得该移动学习平台具备抗网络抖动的能力，可以为用户提供质量更好的视频，进而提高QoE。具体的工作总结如下：

* 视频切分方面。提出了一种基于内容的视频切分算法。该算法针对不同内容的视频码率具有差异的特点，先将视频切分，然后根据内容对视频进行分类并转码，在保证用户体验的同时提升了带宽的利用率。
* 带宽选择方面。提出了一种带宽选择算法。该算法可以根据下载速率判断当前的可用带宽，进而估计出最适合当前网络状况的视频码率，并向服务器请求对应码率的视频分段。这种算法可以提高播放器对网络抖动的稳定性。
* 最后，本文开发了一种面向MOOC的移动视频学习平台，在文中详细介绍了该平台的框架，以及平台每个模块的实现方案及细节。通过一系列实验，本文证明了本文所做的工作是有效的。

二、展望

对于本文的工作，今后还有很多工作需要改进，主要在以下几个方面：

* 提高视频分类的准确度。本文中的视频分类模块采用了相似性检测、人脸识别、运动量检测三种算法，均可以采用更高效的算法对其进行改进。例如，采用机器学习算法学习并判断运动量分布、相似性与视频分类的关系；采用更精确前沿的其他视频质量客观评估算法估计视频的质量等等，这些都对增强视频切分算法的准确率有所帮助。另外，本文认为，对其他视频分段可以进一步讨论，对这些分段进一步处理，以达到更高的用户体验水平。
* 带宽选择算法的进一步优化。由于水平有限，当前带宽选择算法的参数均为经验值。相信经过大量实验的精细测算，带宽算法将更加稳定、抗抖动能力更强。
* 在线视频学习平台的功能改进。本文设计的基于MOOC的在线视频学习平台，目前功能仍然不够完善，例如播放器的功能不够全面、用户不能选择视频清晰度的倾向等等。这个应用的性能以及代码的封装仍有待进一步完善，以达到更高的水平。

参考文献

1. Mackness J, Mak S, Williams R. The ideals and reality of participating in a MOOC[J]. 2010.
2. Breslow L, Pritchard D E, DeBoer J, et al. Studying learning in the worldwide classroom: Research into edX’s first MOOC[J]. Research & Practice in Assessment, 2013, 8(1): 13-25.
3. King C, Robinson A, Vickers J. Online education: Targeted MOOC captivates students[J]. Nature, 2014, 505(7481): 26-26.
4. Knox J, Bayne S, MacLeod H, et al. MOOC Pedagogy: the challenges of developing for Coursera[J]. Association for Learning Technology Online Newsletter, 2012, 28(08.08): 2012.
5. 孟群智. 移动网络环境下移动终端设备的发展与应用[J]. 网络安全技术与应用, 2013 (7): 49-50.
6. 徐苑苑. 云计算环境下的开放课程应用研究[D]. 华东师范大学, 2013.
7. Lohmar T, Einarsson T, Fröjdh P, et al. Dynamic adaptive HTTP streaming of live content[A]. World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a[C]. IEEE, 2011: 1-8.
8. Lai C F, Chao H C, Lai Y X, et al. Cloud-assisted real-time transrating for HTTP live streaming[J]. Wireless Communications, IEEE, 2013, 20(3): 62-70.
9. ITU-T RECOMMENDATION P. Subjective video quality assessment methods for multimedia applications[J]. 1999.
10. ITU-T Recommendation J.140. Subjective picture quality assessment for digital cable television systems [J]. 1998.
11. 佟雨兵, 胡薇薇, 杨东凯, 等. 视频质量评价方法综述[J]. 计算机辅助设计与图形学学报, 2006, 18(5): 735-741.
12. Wang Z, Bovik A C, Sheikh H R, et al. Image quality assessment: from error visibility to structural similarity[J]. Image Processing, IEEE Transactions on, 2004, 13(4): 600-612.
13. Hore A, Ziou D. Image quality metrics: PSNR vs. SSIM[A]. Pattern Recognition (ICPR), 2010 20th International Conference on[C]. IEEE, 2010: 2366-2369.
14. Mas J, Fernandez G. Video shot boundary detection based on color histogram[J]. Notebook Papers TRECVID2003, Gaithersburg, Maryland, NIST, 2003.
15. Lienhart R, Maydt J. An extended set of haar-like features for rapid object detection[A]. Image Processing. 2002. Proceedings. 2002 International Conference on.[C] IEEE, 2002, 1: I-900-I-903 vol. 1.
16. 徐晓艳. 人脸识别技术综述[J]. 电子测试, 2015 (10).
17. Liu Y, Dey S, Gillies D, et al. User Experience Modeling for DASH Video[A]. Packet Video Workshop (PV), 2013 20th International[C]. IEEE, 2013: 1-8.
18. Huang T Y, Johari R, McKeown N, et al. A buffer-based approach to rate adaptation: Evidence from a large video streaming service[A]. Proceedings of the 2014 ACM conference on SIGCOMM[C]. ACM, 2014: 187-198.
19. Hu S, Sun L, Gui C, et al. Content-aware adaptation scheme for QoE optimized dash applications[A]. Global Communications Conference (GLOBECOM), 2014 IEEE[C]. IEEE, 2014: 1336-1341.
20. Li Z, Zhu X, Gahm J, et al. Probe and adapt: Rate adaptation for http video streaming at scale[J]. Selected Areas in Communications, IEEE Journal on, 2014, 32(4): 719-733.
21. Huang T Y, Handigol N, Heller B, et al. Confused, timid, and unstable: picking a video streaming rate is hard[A]. Proceedings of the 2012 ACM conference on Internet measurement conference[C]. ACM, 2012: 225-238.
22. Mok R K P, Luo X, Chan E W W, et al. QDASH: a QoE-aware DASH system[A]. Proceedings of the 3rd Multimedia Systems Conference[C]. ACM, 2012: 11-22.
23. Akhshabi S, Anantakrishnan L, Begen A C, et al. What happens when HTTP adaptive streaming players compete for bandwidth?[A]. Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video[C]. ACM, 2012: 9-14.
24. Liu C, Bouazizi I, Gabbouj M. Rate adaptation for adaptive HTTP streaming[A]. Proceedings of the second annual ACM conference on Multimedia systems[C]. ACM, 2011: 169-174.
25. Brinkmann R. The art and science of digital compositing: techniques for visual effects, animation and motion graphics[M]. Morgan Kaufmann, 2008.
26. Klaue J, Rathke B, Wolisz A. Computer Performance Evaluation: Modelling Techniques and Tools[M]. Springer Berlin Heidelberg, 2003: 255-272.
27. Gamma E, Helm R, Johnson R, et al. Design patterns: elements of reusable object-oriented software[M]. Pearson Education, 1994.

攻读硕士学位期间取得的学术成果

1. Wang Y, Wu W, Lou Y H. MOOC-DASH: A DASH System for Delivering High-Quality MOOCs Videos [A]. Multimedia, IEEE International Symposium on[C]. 2015 (Accepted on 2015-09-18)
2. 吴文峻，王翊，楼奕华．一种用于传输高质量在线课程视频的动态自适应流系统[P]．中国专利: 201510673176.8，2015-10-18

致 谢

时光荏苒，日月如梭，研究生的生活即将结束，学生的身份即将成为过去时。在这期间，我从一个对计算机知识一知半解的数学系本科生开始，成长为一名具备一定专业技能的IT工程师，这期间付出了很多的汗水和努力。在就读北航这短短两年半的时间里，我学到了很多新知识，我的眼界也得到了开阔。而我之所以能走到今天，离不开大家的帮助。在此，我要感谢我的指导教师、曾经帮助过我的所有人以及和我一同成长、奋斗的同窗好友们。

首先，我要感谢我的导师吴文峻教授。由于一系列的巧合，我有幸成为了吴文峻教授的学生，并加入了北航计算机学院软国重的大家庭。他渊博的知识、严谨的态度以及不断进取的精神给我留下了非常深刻的印象。他一直教导我们，要严于律己、深入研究、不止于结果。每次我遇到困难的时候，他总是能够从更加专业、前沿的角度为我提供研究方向及解决方案。再次感谢吴文峻教授两年半以来对我的指导和帮助。

在这里，我要特别感谢我的父亲、母亲。在报考北航、就读北航、寻找工作期间，我曾经遇到了很多、很大的困难、挫折。他们以长者的身份为我指引方向，我以他们为榜样大步前行。依靠着他们的支持、鼓励及帮助，我从入读北航一步一步走来，直到今天。在不远的将来，这份亲情，我会尽力报答。

在这里，我要感谢楼奕华学长。凭借着他在计算机知识及科学研究上深厚的功底及造诣，他给予了我很大帮助。在我遇到困难的时候，他耐心、认真的帮助我解决问题，给我带来了莫大的帮助。未来，我将以他为奋斗的目标，在计算机技术方面不断深入探索，以争取尽快达到他的高度。

在这里，我要特别感谢姚婷学姐，准备考研以及刚就读北航时，她给了我莫大的帮助。可以说，没有她，也许我就会失去入读北航的机会。

在这里，作为同级同学，感谢三年来郭帅同学在各方面给予的照顾。感谢成睿、张海阔学长，他们在我平时的工作中曾经给予过很大的帮助。我还要一并感谢胡征慧学姐、吴利钢、陈永全、于海亮、魏赵栋学长、韩勇、王鑫、黄剑琨、李勃舒学弟、王珊珊、周晓懿、孙时、谢琦、刘丽萍以及周萱学妹三年来给我带来的帮助。能与你们共同学习，是我最大的荣幸。