**毕 业 论 文**

┊┊┊┊┊┊┊装┊┊┊┊┊┊┊订┊┊┊┊┊┊┊线┊┊┊┊┊┊┊

**二○一九年六月**

移动设备流媒体系统实现

专业班级：

姓 名：

指导教师：

信息科学技术学院

**摘 要**

目前随着第四代数字通信技术(4G)的快速发展和对下一代数字通信技术的不断研究，流媒体的服务在生产和生活中日益重要。同时，流媒体技术极大地推动了音视频等多媒体在互联网的应用。本文主要研究流媒体服务体系以及Android平台下的视频开发技术，构建一个基于Android平台的移动设备流媒体客户端的系统。

本文深入研究了Android 操作系统开发架构，移动多媒体开发框架，流媒体服务逻辑结构及以移动流媒体协议等相关技术，在此基础上完成基于Android 的移动流媒体客户端的设计与开发。本系统主要完成三个子系统，视频采集子系统，视频播放子系统和视频裁剪子系统。对于视频采集子系统提取 H.264 视频数据和 ARMNB 音频数据，来实现视频录制的功能。而视频播放子系统作为本文系统的核心，在基本播放模块主要实现本地视频播放，视频点播以及视频实时直播观看三部分，实现了流媒体技术在播放功能上顺序流式传输和实时流式传输两方面的应用。此外还进行监控视频播放窗口的滑动和全屏原屏切换等播放功能的优化。在视频裁剪子系统的实现过程中，采用Java封装FFmpeg命令行加载FFmpeg库，通过视频裁剪的命令将视频进行相应裁剪，来实现相应的功能。

经过后期的测试和实际应用，本系统成功实现了移动视频采集、流媒体实时播放、视频裁剪、视频保存等功能。在实际的运行环境中，播放流畅，延迟较小，具有较高的实用价值和很强的可移植性。

**关键词：流媒体；Android；实时传输；FFmpeg；播放控制**

**ABSTRACT**

With the rapid development of the fourth generation of digital communication technology (4G) and the continuous research of 5G networks，streaming media services are increasingly important in production and life. At the same time，streaming media technology has greatly promoted multimedia such as audio and video in the Internet. Applications. This paper mainly studies the streaming media service system and the video development technology under the Android platform，and builds a mobile device streaming client system based on the Android platform.

This paper deeply studies the Android operating system development architecture， mobile multimedia development framework，streaming media service logic structure and related technologies such as mobile streaming media protocol. On this basis，the design and development of Android-based mobile streaming media client is completed. The system mainly completes three subsystems a video acquisition subsystem，a video playback subsystem and a video cropping subsystem. The video capture subsystem extracts H.264 video data and ARMNB audio data for video recording. As the core of the system，the video playback subsystem mainly implements three parts： local video playback，video on demand and live video live broadcast. The streaming media technology implements sequential streaming and real-time streaming in the playback function. Aspect of the application. In addition，the playback function of the monitoring video playback window and the full screen original screen switching are optimized. In the implementation process of the video cropping subsystem，the FFmpeg library is loaded in the Java package FFmpeg command line，and the video is cropped by the video cropping command to implement the corresponding functions.

After the later testing and practical application，the system successfully realized the functions of mobile video capture，streaming media real-time playback， video cropping，video saving and so on. In the actual operating environment，the playback is smooth，the delay is small，and it has high practical value and strong portability.

**Key words：Streaming Media；Android；Real-time transmission；FFmpeg；Playback control**

**目 录**

[第1章 绪论 1](#_Toc11308575)

[1.1 研究背景及意义 1](#_Toc11308576)

[1.2 课题平台选择 1](#_Toc11308577)

[1.3 国内外研究现状 2](#_Toc11308578)

[1.4 论文组织结构 3](#_Toc11308579)

[第2章　移动流媒体基本理论 5](#_Toc11308580)

[2.1 移动流媒体服务体系 5](#_Toc11308581)

[2.1.1 移动互联网接入标准 5](#_Toc11308582)

[2.1.2 流媒体体系结构 5](#_Toc11308583)

[2.1.3 流媒体服务逻辑结构 6](#_Toc11308584)

[2.2 移动流媒体协议介绍 6](#_Toc11308585)

[2.2.1 RTP实时传输协议 6](#_Toc11308586)

[2.2.2 RTCP实时传输控制协议 7](#_Toc11308587)

[2.2.3 RTSP实时流媒体传输协议 7](#_Toc11308588)

[2.3 Android系统特点 7](#_Toc11308589)

[2.3.1 Android框架 7](#_Toc11308590)

[2.3.2 Android应用程序 8](#_Toc11308591)

[2.3.3 Android多媒体框架 9](#_Toc11308592)

[第3章 移动流媒体系统方案分析与设计 11](#_Toc11308593)

[3.1 系统需求分析 11](#_Toc11308594)

[3.1.1 系统功能需求 11](#_Toc11308595)

[3.1.2 系统性能需求 12](#_Toc11308596)

[3.2.3 系统整体设计 12](#_Toc11308597)

[3.2 流媒体系统设计 13](#_Toc11308598)

[3.2.1 系统环境概述 13](#_Toc11308599)

[3.2.2 系统功能模块设计 14](#_Toc11308600)

[3.2.3 系统用例说明 18](#_Toc11308601)

[第4章 移动流媒体系统的实现 22](#_Toc11308602)

[4.1 系统开发环境 22](#_Toc11308603)

[4.1.1 软件开发平台搭建 22](#_Toc11308604)

[4.1.2 移植FFmpeg到Android 22](#_Toc11308605)

[4.2 流媒体客户端的实现 23](#_Toc11308606)

[4.2.1 用户界面实现 24](#_Toc11308607)

[4.2.2 视频采集模块实现 24](#_Toc11308608)

[4.2.3 视频播放模块实现 26](#_Toc11308609)

[4.2.4 视频裁剪模块实现 34](#_Toc11308610)

[第5章 移动流媒体系统测试与结果分析 36](#_Toc11308611)

[5.1 功能测试 36](#_Toc11308612)

[5.1.1 视频播放子模块测试 36](#_Toc11308613)

[5.1.2 视频录制子模块测试 37](#_Toc11308614)

[5.1.3 视频裁剪子模块测试 37](#_Toc11308615)

[5.2 性能测试 38](#_Toc11308616)

[5.3 结果分析 38](#_Toc11308617)

[结论 39](#_Toc11308618)

[参考文献 41](#_Toc11308619)

[致 谢 42](#_Toc11308620)

**移动设备流媒体系统实现**

# 第1章 绪论

随着第四代数字通信技术(4 G)的快速发展和对下一代数字通信技术的不断研究，同时移动电话等移动设备的处理能力正在迅速提高，人们不再满足于文本和语音信息的交换。视频媒体信息的服务在生产和生活中日益重要。移动流媒体将传统的网络流技术应用于无线传输领域，正在变成世界上移动业务的研究热点之一。

## 1.1 研究背景及意义

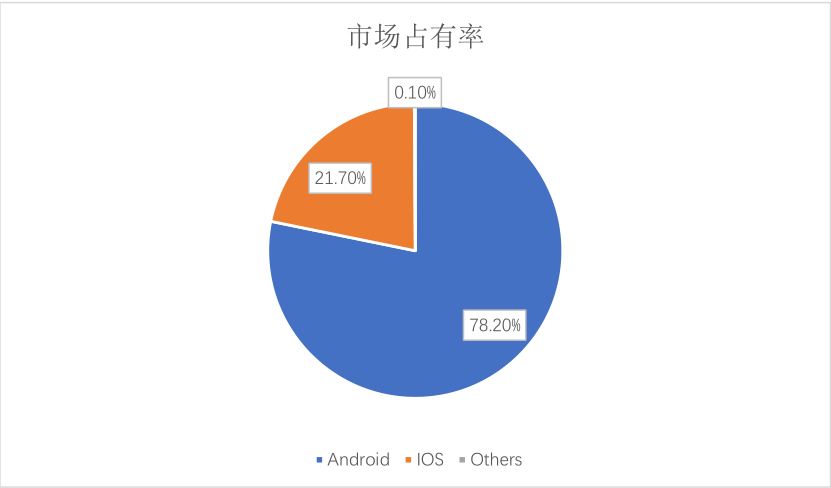
流媒体(Streaming Media)是指通过网络按时间顺序传输和播放的连续音频和视频数据流。作为从因特网开发的多媒体应用技术，流媒体技术是一种通过流传输网络上的多媒体文件的技术。流媒体传输的主要特征是流媒体形式的多媒体数据传输，生成的图像和声音信息被压缩并放在网络流媒体服务器上。然后，客户端通过缓冲区接受媒体数据，然后在播放时播放并继续从服务器下载剩余数据[1] 。流媒体技术不是单一技术，它融合了流媒体数据的采集、压缩、存储和网络通信等相关网络技术。

目前，4G网络正在突飞猛进，5G网络已接近现实，移动流媒体服务支持高速数据传输。提供至少数十Mbps的带宽，为移动流应用提供良好支持。目前，移动流媒体，包括视频会议，视频会话，直播新闻，实时监控，移动多媒体娱乐，远程教育等应用，已成为仅次于语音服务的第二大移动应用。

视频编码技术的标准化是工业化的前提和基础。目前，下一代视频编解码技术（如H.264标准）的压缩效率是以前标准的两倍多。H.264不仅保留了以往压缩技术的精髓，而且还具有其他压缩技术无法实现的优势，受到大范围的发展和应用。

## 1.2 课题平台选择

目前市场上手机操作系统主要包括：Apple公司的iOS、 Google公司的Android、 Software公司的Window mobile和RIM公司的BlackBerry以及Linux等。根据Kantar Worldpanel于2019年3月发布的中国移动手机市场占有率报告如图1.1所示



**图1.1 智能手机市场占有率**

根据数据可以看出国内智能手机市场Android手机占有大多数市场份额，而且Google公司在 2007 年正式公开的 Android 操作系统是以开源的 Linux 系统为基础，将C语言内置到底端，让硬件操作速率达到最好；应用层方面则选择扩展功能厉害，操作比较方便的 Java 编程语言，并且开放了源代码。因此，Android操作系统一经推出便受到各大公司的强烈支持，打破了iOS在智能移动手机领域的领导地位。因此，本课题采用Android系统作为开发平台。

## 1.3 国内外研究现状

2018年，在第三届全国党报网站高峰论坛发布的统计数据显示[2]，2017年中国移动互联网基础服务建设成果速度最快。4G网络建造全面展开，4G站点数量增加652000个，总的个数已经是328000个。中国开始进行5G三期试验，开始部署6G网络，窄带物联网也进入快速发展阶段。智能手机市场日益饱和，智能硬件和智能终端正在快速增长。智能机器人，无人机，智能家居和自动驾驶等技术取得了重大突破。中国已经形成了世界上最大的移动互联网应用市场。截至2017年12月底，监测移动应用程序403万个，移动应用市场达到7865亿元。

近年来，无线通信技术进入了一个活跃的发展时间点，多媒体技术已进入快速发展和应用时间点。移动流媒体详细使用传统的流媒体技术，并在移动终端平台上得到了扩展，并且取得了巨大的成果。

在当今的移动流媒体播放系统中，国内外各大企业都做了大量的开发和使用。RealNetworks是流媒体市场的先驱，其Helix开源项目计划在各种终端上开发rm格式的播放软件。目前，Storm Video Player以其小格式，高加载速度和全面支持高清解码，使人们对1080P网络高清视频的浏览成为现实：播放器具有5秒的加速度，可将原始视频从15秒缓冲到5秒。国内外许多视频平台，如YouTube，Bilibili和QIY，已经推出了用于移动设备的流媒体播放器。移动终端上的流视频应用可以满足真实移动，紧急或临时场合的公司，家庭和其他地方的视频监控和管理。另一方面，它在4G网络下发挥全覆盖和高速带宽的特点，并且还利用了移动终端的即时传输，快速和个性化的优点，具有广阔的市场和前景。

目前，在全球消费者固定网络IP流中，视频服务已达到80％，而全球移动数据流在2021年占80％。随着AR，VR和其他服务的发展，视频业务将以更快的速度增长。诸如车辆互联网，物联网和工业互联网等应用将加快增强移动多媒体交易。使用单播的传统移动通信可以支持移动多媒体服务，但效率低且不经济。5 G考虑在标准化中优化支持多播/广播模式的技术，包括网络存储管理和数据管理。网络发现、应用、认证、移动性管理和会晤管理以及用户面等功能，加快了开发支持组播与广播方式的移动多媒体业务，努力实现通信网络和广电网络融合[3]。我相信移动流媒体技术在不多的之后会有更多的创新空间。

## 1.4 论文组织结构

本文在对目前的处于移动智能设备平台的流媒体播放器系统进行调查，讨论该技术的长短处，为了满足当前用户对于流媒体播放设备的需求，来实现移动设备流媒体系统。本文的论文组织结构如下：

第1章：先研究系统的背景知识，分析流媒体技术在国内外的发展情况，根据市场和用户需求选择合适的开发平台，对流媒体技术和系统开发有了初步了解。

第2章：对移动流媒体各种关键技术进行了解和掌握。认识3GPP下的目前使用和开发的4G和5G标准，对流媒体技术运用的网络协议进行学习，掌握Android架构的特点，深入了解基础理论。

第3章：执行软件开发的需求分析和软件设计。分析系统对功能和性能的要求，并对要实现的功能提供基本的了解; 对系统的功能模块进行设计，将实现的功能具体化到各个模块，对实现的模块进行流程分析和步骤分解，最后，使用用例分析详细描述每个功能。

第4章：对移动流媒体的系统功能进行研发和实现。介绍各个功能模块在开发过程中遇到的问题和解决方案，介绍各个功能的核心代码，展示功能实现的界面截图。

第5章：进行系统软件测试和对运行结果进行分析。对系统进行功能，性能等方面的测试，找出软件漏洞，使系统达到用户需求的标准。

结论：总结并思考本次毕业设计项目的工作。发现不足之处，去寻找解决方案，让系统更加优化，展望未来，去实现接下来的工作。

# 第2章　移动流媒体基本理论

## 2.1 移动流媒体服务体系

### 2.1.1 移动互联网接入标准

第三代合作伙伴计划（3GPP）是一个标准化组织，其开发协议移动电话。其最重要的工作是开发和维护[4]：

1. GSM和相关的2G和2.5G标准，包括GPRS和EDGE
2. UMTS和相关的3G标准，包括HSPA
3. LTE和相关的4G标准，包括LTE Advanced和LTE Advanced Pro
4. 下一代和相关的5G标准
5. 与访问无关的方式开发的演进IP多媒体子系统

端到端的包交换流媒体服务(Packet Switched Streaming Service，PSS) 是3GPP移动网络交互式流服务框架中定义的规范。移动流媒体系统有RTP，RTCP，RTSP，HTTP和SDP等网络运输和发展协议。PSS框架规定了移动流媒体服务端和移动流媒体客户端之间的通信内容，可以根据不一样的网络环境进行调整和处理。流媒体服务端和客户端之间的通信是单播和多播。流媒体服务端需要使用单播或多播，具体取决于特定的应用程序大小。

### 2.1.2 流媒体体系结构

3GPP提出的端到端分组交换流媒体服务规范根据层次结构将流媒体系统划分为三个级别：流媒体服务器，移动通信网络，流媒体客户端（移动终端设备等）。

流媒体服务器通常存在于因特网上并提供流服务。公共互联网中的服务器除了可以为办公PC和个人笔记本电脑提供流媒体服务，还可以为移动通信网络中的移动设备提供各种服务。

移动通信网络主要有3G(HSPA、HSPA +、LTE（E-UTRA）)、4G(E-UTRA、LTE Advanced Pro)、5G(NR、LTE-M)。

流媒体客户端是直接与用户交互的流媒体系统的一部分。在流媒体客户端中，它负责从服务器要求数据，然后靠着移动通信网络来接收数据，解码和播放。流媒体客户端播放时的稳定性，播放的清楚性和流畅性以及图像画面将直接影响用户体验。因此，在设计过程中，开发人员需要注意这些情况，而这些情况同时是移动流媒体系统设计的重难点。

### 2.1.3 流媒体服务逻辑结构

3GPP指定的移动流媒体系统采用Client/Server（C/S）模式。服务器端可以分为流媒体服务平台，管理平台和编码服务平台。C/S模型流传输服务器通常使用RTP /RTSP实时流传输协议来向客户端提供流传输服务。传统的流媒体系统采用基于单播传播的C/S结构。媒体内容存储在媒体服务器中心中。当客户端需要得到数据时，它可以传输服务器提供的流方向。服务器的负载能力是整个系统的负载能力。 目前，有许多关于流媒体服务器的研究。然而，由于智能移动终端的不同性和困难性，研究流媒体客户端对于流媒体系统也是极其重点的。

## 2.2 移动流媒体协议介绍

流式传输技术作为流媒体技术中核心的部分。其中它的使用实现了流媒体客户端和服务器之间的通信。目前，RTP(Real-Time Transport Protocol，实时传输协议)、RTSP(Real-Time Streaming Protocol，实时流媒体协议)、RTCP(Real-Time Transport Control Protocol，实时传输控制协议)是常用的流媒体传输协议[5]，其在TCP/IP协议中的位置如图2.1所示

|  |  |
| --- | --- |
| 应用层(HTTP、RTSP、MMS) | |
| 传输层 | RTP、RTCP、RSVP |
| TCP、UDP |
| 网络层(IP) | |
| 网络接口层 | |
| 物理层 | |

**图2.1 流媒体传输协议在TCP/IP协议中的位置**

### 2.2.1 RTP实时传输协议

RTP主要是一种实时传输服务，为用户提供连续的媒体数据。它基于多播或单播网络。 RTP协议提供对等通信，并且可以通过多播网络执行实时数据传输。这种对等和多播模式确保了时间信息和实现流的同步。 RTP通常使用UDP进行数据传输，具有实现简单，不需要路由器支持，控制信息占用带宽小的优点。

RTP 协议是位于传输层上的协议。它可以建立在TCP / UDP协议上，但通常它使用RTP和UDP进行实时传输。在音频和视频传输过程中，使用UDP端口号和学校。 它还继承了 UDP 协议小传输延迟的优点。当需要发送音视频数据时，将数据封装成RTP数据包，然后将RTP数据包封装成UDP数据包，然后封装成IP数据包进行传输。接收音频和视频数据包时，还需要通过UDP套接字输入RTP数据包，然后提取音频和视频数据。

### 2.2.2 RTCP实时传输控制协议

实时传输控制协议（Real-time Transport Control Protocol，RTCP）是 RTP 的关联协议。传输期间供给的功能包括数据流控制和拥塞控制。该实现使用与分组传输相同的分发机制并使用控制信息。它会定期发送给会话中的每个参与者。在RTP会话中，每个参与的会话器周期性地发送包含统计数据的RTCP分组，例如丢失的数据分组的数量和发送的数据分组的数量。服务器可以通过这些统计信息来实时地改变数据流传输速度，甚至可以改变有效负载的种类。RTCP 与网络应用中的 RTP 结合使用，以提供对底层网络的监控以及会话期间端点外的带外通信。RTCP定义了五种类型的消息，其功能由不同的消息实现：SR（Sender Report），RR（Receiver Report），BYE，SDES（Source Description）和APP（Application-defined）。差异主要通过头部的 PT（Packet Type）来区分。

### 2.2.3 RTSP实时流媒体传输协议

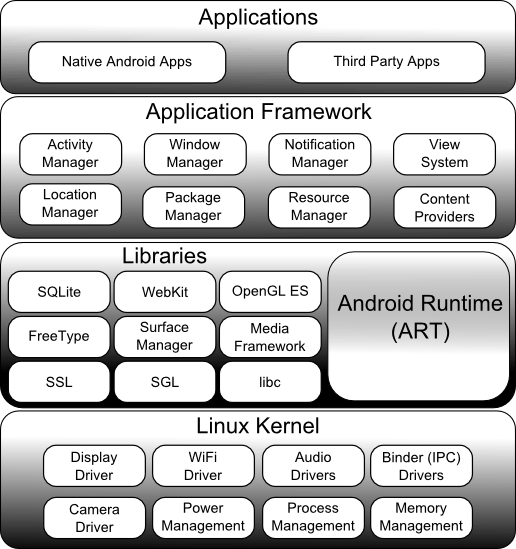
RTSP是位于应用层的流媒体渲染协议，通过它可以控制流媒体数据。因为它是应用层协议，所以它不负责数据本身的传输。传输数据由底层传输。 通常，RTSP与RTP协议一起用于音频和视频的实时传输。RTSP协议具有可扩展的框架，用于控制音频和视频回放端的音频和视频，例如暂停，播放，快进等。RTSP协议是一种有状态的对等协议，在进行RTSP数据流请求时维护会话状态。这些功能与 HTTP 协议有很大不同。另外，它类似于HTTP 协议。

## 2.3 Android系统特点

Android是由谷歌开发的移动操作系统。它基于Linux内核和其他开源软件的修改版本，主要用于智能手机和平板电脑等触摸屏移动设备。Android最初由Google于2005年收购，于2007年推出，并于2008年9月推出首款商用Android设备。操作系统已经开发了多个主要版本，当前版本为9 “Pie”。

### 2.3.1 Android框架

Android的系统架构采用分层结构[6]，从上到下分为四层，分别是应用程序层（Application Layer）、应用程序框架层（Application Framework）、系统运行库层（Libraries）和Linux内核层（Linux Kernal），如图2.2所示。



**图2.2 Android框架**

1. 应用程序层，该层由在Android Dalvik虚拟机上运行的应用程序组成。这层应用程序是用Java编写的，其中包括流行的移动应用程序，如电话簿，地图和浏览器。Android嵌入式平台应用程序比其他移动电话系统平台更注重灵活性和个性。例如，开发人员可以根据自己的爱好设计个性化电话簿，地图和其他应用程序，而不是像其他移动电话系统那样直接体现这些系统软件。

2. 应用程序框架层，主要由开发人员可以直接调用的应用程序组件组成，如消息管理器，资源管理器，视图管理器等。 这些组件是我们开发 Android应用程序的基础，它是 Android 系统。开发人员可以直接轻松打包应用程序组件，我们可以继承这些组件来开发个性化应用程序。

3. 系统运行库层，该层主要由两部分组成：系统库和Android运行时。该层是为应用程序框架层提供支持的层。

4. Linux内核层，Android嵌入式系统基于Linux 2.6内核。该层主要包括驱动程序，内存管理，进程管理和网络协议等部件。

### 2.3.2 Android应用程序

Android应用程序主要包含以下组件：

1. Activity：该组件是Android应用程序的窗口，主要负责与用户交互。

2. Service：此组件没有可视界面，主要在后台运行。通过使用Service，您可以在后台运行一些复杂的操作和服务，而不会影响前端界面的用户体验。

3. Broadcast Receives：该组件主要用于接收广播信息和响应信息。 通常，系统发送广播信息，例如呼叫，低电量等。当然，应用程序也可以发送广播消息。应用程序可以从多个广播消息发送者接收消息。此操作没有用户界面，但消息响应可以启动界面活动以与用户交互。

4. Content Providers：该组件主要用于向其他应用程序提供数据，允许应用程序与其他应用程序共享数据。

除了这些组件之外，每个Android应用程序还必须具有配置文件：AndroidManifest.xml。在此配置文件中，定义了有关Android应用程序中使用的组件和组件的一些约束信息。 它还涉及应用程序对网络，存储卡等的访问控制，只有配置具有一定的权限，应用程序无法实现其目的以执行相应的操作，否则，即使它应用于程序中。

### 2.3.3 Android多媒体框架

从图2.2可以看出，多媒体框架层位于系统运行时的库中。库的这一层通常由C\C ++实现，并由Java JNI调用。Android多媒体框架使用OpenCore开源框架。其中H. 264解码器是目前开源H. 264中最好的解码器之一。在多媒体方面，Android2.3更新了音频API，增加了VP8、WEBM等多媒体格式，可以使用AAC和ARM录制更高质量的音频。

OpenCore是用C++实现的，有大量代码定义了功能齐全的操作系统移植层。从宏观上看，框架主要分为两个方面：

PVPlayer：提供多媒体播放功能，实现多种音频和视频流格式的播放。PVAuther：为多种音频、视频流格式和静态图像捕获提供多媒体流记录。

Android以SDK的形式向开发人员提供了这两种功能。通过这个多媒体框架，程序员可以在SDK上快速开发各种多媒体应用程序，如音乐播放器、视频播放器、流媒体播放器、音频和视频采集等特性。

OpenCore功能强大，PVPlayer不仅可以播放文件，还可以播放网络媒体流，具有媒体流控制、音频和视频流解码和文件解析功能。PVAuther主要负责媒体流记录，主要基于实时流协议RTSP，包括流同步、音频和视频编码以及文件编写。

OpenCore的底层实现有一个巨大的函数库和代码结构。我们不需要在开发过程中研究它的实现过程。Android提供了用于上层开发的媒体API，主要包括两个类：MediaPlayer类和MediaRecorder类。

MediaPlayer主要负责播放音频和视频，MediaRecorder主要负责录制音频和视频。MediaPlayer有一个复杂的生命周期。在使用MediaPlayer进行应用程序开发时，必须考虑它的生命周期，否则程序将会出现意外错误。

# 第3章 移动流媒体系统方案分析与设计

## 3.1 系统需求分析

开发中最重要的任务是分析用户的需求。需求分析是软件开发的重要组成部分。其基本任务是首先确定用户的具体需求。 本文利用功能和性能来分析系统的要求。

### 3.1.1 系统功能需求

流媒体技术提供技术支持。第四代无线网络通信提供了速度快、带宽大的传输服务的功能，智能手机为多媒体技术的实施提供平台支持。 因此，该系统构建了基于Android移动终端的流媒体传输系统。

1. 本地视频可以通过Android平台的流媒体播放器播放，用户可以获取视频信息并在客户端上观看。
2. 在Android平台上实现视频点播的播放，即播放的在线视频可被用户任意观看，每次都可从头到尾进行播放。
3. 在Android平台上实现视频直播的播放，即观看诸如电视直播的视频节目，播放正在实时发生的画面。
4. 通过Android平台，对视频播放功能进行优化，实现动态切换全屏/原屏、横屏/竖屏的自由切换。
5. 通过Android平台，监控视频播放器窗口的滑动，实现亮度加减，音量加减，快进和快退功能。
6. Android平台的视频客户端调用摄像头录制视频，以便用户可以根据日常需要在生活中逐位记录。
7. 通过Android平台可以实现获取手机上的所有视频，选择某一视频，可以进行客户端播放。
8. 通过Android平台可以实现视频长按选择打开方式，实现用开发的播放器进行播放。
9. 通过Android平台实现对内容信息的分享，将内容分享自微信、QQ、知乎、邮箱、微博等其他平台。
10. 通过Android平台对FFmPeg解码移植，实现获取视频的第一帧预览图像，剪裁视频存放在本地文件夹。

### 3.1.2 系统性能需求

1．界面美观，操作简便

本系统旨在满足一些用户需求，为用户提供便利。因此，在设计系统时必须要考虑用户体验。如果系统不太可用，那么即使系统具有更强大的功能。它不能被用户接受，系统只会变得无用。该系统的目的是为用户提供流畅的音视频播放，然后每个功能模块需要紧密匹配，界面设计满足用户的使用习惯，界面美观，操作简单。

2．实时传输

系统的目标是构建实时流媒体传输系统，因此实时性能是系统性能要求的基本要求。它具有强大的实时性能，不仅可以改善用户体验，还可以为应急响应做出决策。此外，实时传输强劲，视频播放流畅，不存在干扰，乱码，数据丢失等现象，提升了用户体验。

3．视频图像质量

系统的目的是让用户清楚地看到手机拍摄的视频图像。 如果图像质量很差，则用户几乎看不到屏幕上的模糊图像。即使它具有强大的实时性和完善功能，也无法为用户提供服务。该系统没有达到预期的效果，注定是一个失败的系统。

4．视频编码和解码效率

系统可以从采集和回放功能分为采集和回放，最关键的任务是捕获和播放。 根据技术分析，H.264视频编解码技术具有良好的编解码比，因此为了保证系统的平滑性，必须具有较高的编解码效率[7]。

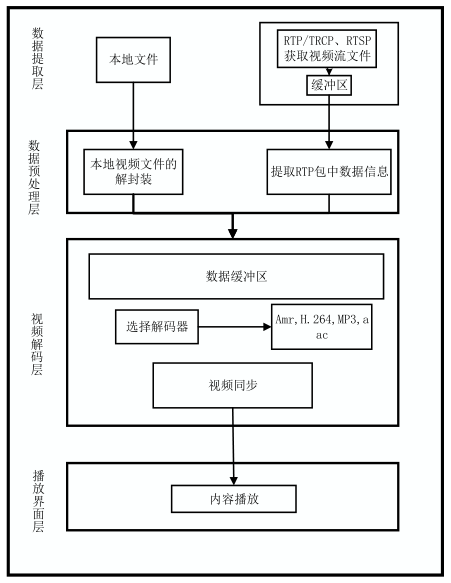
5．用户体验流畅性

系统的目的是让用户看到流畅清晰的图片，查看完整的视频图像信息。每个模块必须紧密匹配，不会发生错误。

### 3.2.3 系统整体设计

流媒体播放器必须在播放远程视频监控视频或本地视频文件后获取视频数据。然后对所获得的音频和视频数据进行解码，并根据视频文件的回放过程将解码后的数据分为四个阶段进行处理，采用分层播放结构设计。

由于远程监控视频与本地视频文件不同，为了保持上层解码的一致性，前两个文件被预处理，两个文件以相同的格式生成并传递给上层解码。根据上述特征，在文件解码过程中，终端采用分层结构，每层独立执行其功能，降低了层间的耦合度，有利于功能扩展。分层结构分为信息提取层，信息预处理层，视频信息解码层和回放接口层，如图3.1所示。



**图3.1 流媒体视频播放系统整体框图**

数据提取层功能包括网络视频文件和本地视频文件的采集。网络流媒体文件需要从服务器获取流媒体数据信息，而本地视频文件只需要读取本地文件信息。

数据预处理层根据其所属的媒体格式对本地视频文件进行解码，并获取文件字幕或音视频信息，并将所获得的内容存储到相应的解码缓冲区中。网络流媒体内容需要提取RTP头信息，然后将信息帧存储在解码缓冲区中的RTP中。

## 3.2 流媒体系统设计

### 3.2.1 系统环境概述

1．硬件环境

采集终端：Android智能手机，相机，H.264硬编码，4G无线网卡。

服务终端：该系统主要在主流PC上运行。

手机播放端：Android智能手机，4G无线网卡

2．软件环境

采集终端：Android2.2及以上。

服务器端：Windows，Java 运行环境，流媒体播放器。

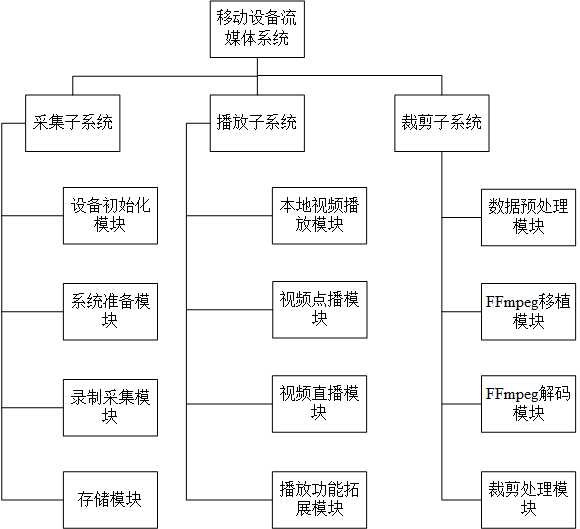
播放客户端：PC：流媒体播放器；手机播放端：移植FFMPEG，Android 1.5以上。

3．开发环境

Windows，JDK，Android Studio，Android SDK，Android NDK，ADT，WireShark1.6.2。

### 3.2.2 系统功能模块设计

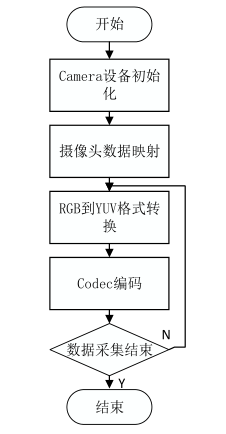
经过之前的系统需求分析，流媒体系统客户端主要分为视频捕获子系统，视频播放子系统和视频裁剪子系统。系统的整体功能模块划分图如图3.3所示。



**图3.3 系统模块图**

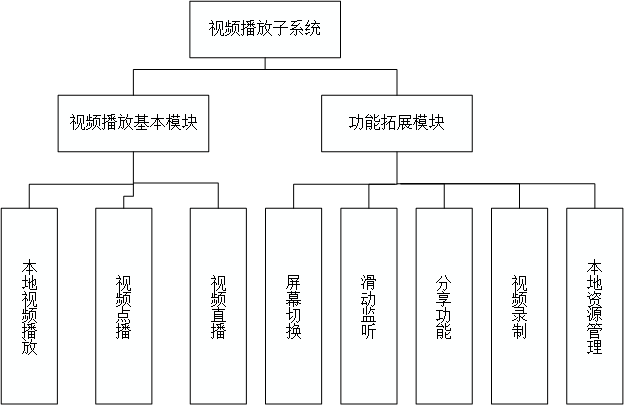
论文将重点关注的是：视频播放子系统和视频裁剪子系统的设计。而对于视频采集子系统则是要简要介绍其实现视频录制的功能。

对于录像功能的实现主要是：数据采集层包括相机的传输和接收控制信息，也就是说，用户发送的命令控制相机传输到相机驱动模块封装后的数据预处理层、和功能来控制相机。摄像头主要用于采集视频信号，捕获的视频数据通过核心帧V4L2的vidioc\_dqbuf操作放入缓冲队列中，然后读取缓冲队列中的数据并进行编码。摄像头数据采集控制流程图如图3.4所示。



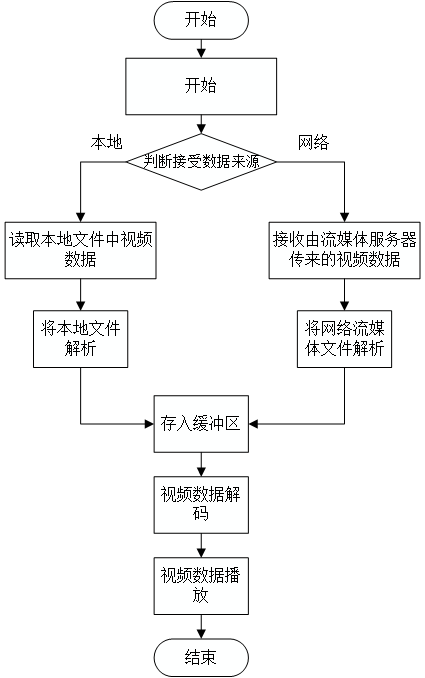
**图3.4 摄像头数据采集控制流程图**

视频播放子系统分为视频播放基本模块和功能扩展模块。在视频播放的基本模块中，该模块是流媒体播放子系统的核心模块。主要负责视频播放，视频播放控制，使用户可以根据自己的需要控制视频暂停，播放，停止，快进等控制选项。该模块具体分为数据接收模块，数据预处理模块和解码播放模块。在功能扩展模块中，模块是流媒体播放子系统的扩展模块，提供视频播放辅助功能。 它主要分为横竖屏切换、全屏原屏转换，视频滑动监听模块，分享功能模块和本地视频资源管理模块，如图 3.5 所示。



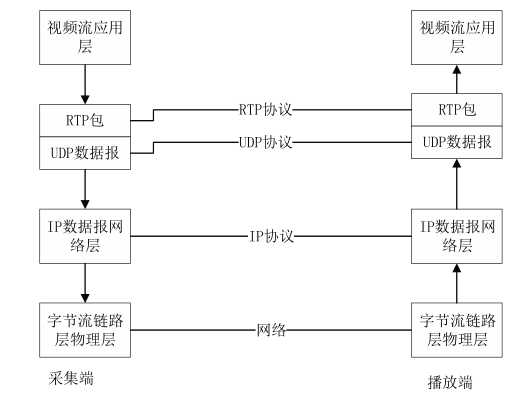
**图3.5 视频播放功能模块图**

对于视频播放基本功能模块， 图3.6为本模块的流程图。 在系统中播放视频的具体过程如下： 首先，执行系统初始化，然后视频判断接收源。 根据接收的数据源是本地文件还是网络文件，系统解析数据并将解析的数据放入缓冲区以进行下一步。



**图3.6 视频播放流程图**

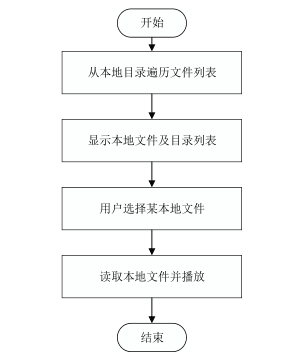
对于网络视频流的传输解析过程如图3.7所示



**图3.7 网络传输解析过程**

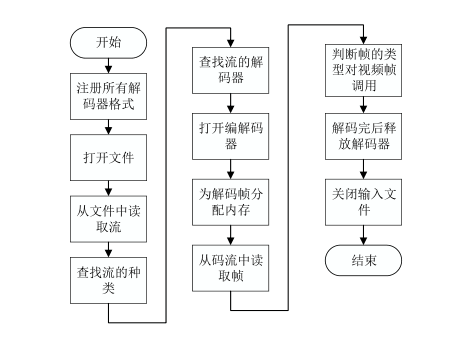
主要是为扩展流媒体播放子系统的功能而设计的。下面将主要介绍本地资源管理和视频录制的设计流程。

本地资源管理主要使用递归显示本地目录方法向用户显示本地目录结构和视频文件。具体过程如图3.8所示。



**图 3.8 本地资源管理流程**

视频裁剪子系统中主要是调用用Java语言封装好的FFmpeg类库来对视频进行解码操作，之后使用FFmpeg中的系统命令来获取对应帧的图像以及剪裁所需时间段对应的视频流。然后将结果保存到存储位置方便调用。其中FFmpeg的解码流程如图3.9所示。



**图3.9 FFMPEG解码流程**

### 3.2.3 系统用例说明

通过对程序的功能模块图进行进一步详细解读，可得到更为详细，更明确的播放器的基本操作事件需求。

1．视频文件播放

**表3.1 视频文件播放用例说明**

|  |  |
| --- | --- |
| 描述项目 | 描述内容 |
| 用例配额 | 视频文件播放 |
| 用例描述 | 单击播放按钮 |
| 参加人员 | 用户 |
| 前置条件 | 视频进来播放状况 |
| 后置条件 | 选中的视频开始来播放 |

从表3.1可知，本用例主要用来描述视频文件能否正常地播放。

2．视频文件停止

**表3.2 视频文件停止用例说明**

|  |  |
| --- | --- |
| 描述项目 | 描述内容 |
| 用例配额 | 视频停止 |
| 用例描述 | 当视频文件处于播放状况时，点击停止按钮 |
| 参加人员 | 用户 |
| 前置条件 | 视频正常播放状态 |
| 后置条件 | 视频停止播放 |

从表3.2可知，本用例主要用来描述某视频文件正在播放时，是否可以点击停止按钮，转变为停止状态。

3．视频文件全屏/原屏切换

**表3.3 视频文件全屏/原屏切换用例说明**

|  |  |
| --- | --- |
| 描述项目 | 描述内容 |
| 用例配额 | 视频全屏/原屏 |
| 用例描述 | 点击按钮，如果是原始屏幕，单击进入全屏。如果是全屏，单击后返回到原屏 |
| 参加人员 | 用户 |
| 前置条件 | 视频处于播放状态 |
| 后置条件 | 当前视频从全屏切换为原屏或从原屏变为全屏 |

从表3.3可知，此用例主要用于描述是否正在播放视频文件，单击切换按钮，以及是否可以切换全屏和原始屏幕。

4．视频文件录制

**表3.4 视频文件录制用例说明**

|  |  |
| --- | --- |
| 描述项目 | 描述内容 |
| 用例配额 | 视频录制 |
| 用例描述 | 点击视频录制按钮，进入视频录制界面 |
| 参加人员 | 用户 |
| 前置条件 | 视频已播放完毕回到主界面 |
| 后置条件 | 视频进入录制阶段 |

从表3.4可知，本用例来描述视频文件能否进行录制，将录制文件存储到本地。

5．视频文件共享

**表3.5 视频文件共享用例说明**

|  |  |
| --- | --- |
| 描述项目 | 描述内容 |
| 用例配额 | 视频文件共享 |
| 用例描述 | 点击视频共享按钮，共享内容到其他平台 |
| 参加人员 | 用户 |
| 前置条件 | 视频已播放完毕回到主界面 |
| 后置条件 | 视频进入共享阶段 |

从表3.5可知，本用例主要用来描述视频文件内容能否分享到其他应用平台。

6．本地视频文件管理

**表3.6 本地视频文件管理录制用例说明**

|  |  |
| --- | --- |
| 描述项目 | 描述内容 |
| 用例配额 | 本地视频文件管理 |
| 用例描述 | 点击获取按钮，获取本地视频文件，选择本地视频文件进行播放 |
| 参加人员 | 用户 |
| 前置条件 | 视频已播放完毕回到主界面 |
| 后置条件 | 打开本地文件管理 |

从表3.6可知，该用例主要用于描述获取本地视频文件和选择本地视频文件以进行回放。

7．视频文件裁剪

**表3.7 视频文件裁剪用例说明**

|  |  |
| --- | --- |
| 描述项目 | 描述内容 |
| 用例配额 | 视频文件裁剪 |
| 用例描述 | 将视频拖动到裁剪界面，进行所需内容的裁剪 |
| 参加人员 | 用户 |
| 前置条件 | 视频已播放完毕回到主界面 |
| 后置条件 | 当前视频开始进行裁剪 |

从表3.7可知，本用例主要用来描述视频文件裁剪到所需时间段，将裁剪好的视频存储到本地文件。

8．视频文件点播

**表3.8 视频文件点播用例说明**

|  |  |
| --- | --- |
| 描述项目 | 描述内容 |
| 用例配额 | 视频文件点播 |
| 用例描述 | 输入视频文件的链接进行网络视频文件的播放实现 |
| 参加人员 | 用户 |
| 前置条件 | 输入所需视频文件地址 |
| 后置条件 | 视频播放结束 |

从表3.8可知，本用例主要用来描述播放的在线视频可被用户任意观看，每次都可从头到尾进行播放。

# 第4章 移动流媒体系统的实现

本章主要介绍了流媒体播放器各模块的具体实现。前几章重点介绍了相关技术和主要设计。在本章中，将会仔细描述实施过程中遇到的困难。

## 4.1 系统开发环境

### 4.1.1 软件开发平台搭建

本文中使用的开发平台为 Android Studio、Android SDK，Android移动手机平台、Windows 10操作系统。

搭建过程：

1.安装 JDK，在系统环境变量PATH里面添加JAVA\_HOME的环境变量，其变量值为 Java 的安装路径。

2.安装 Android SDK，解压到所需文件夹中。

3.在官方网站上下载Android Studio安装包，单击安装，然后选择所需的安装路径。安装完成后，进行配置，选中“启动Android Studio”，然后单击“完成”以启动AS。 接下来按照界面要求进行选择，指定 SDK 的本地路径。最后，配置AS以首次运行环境，以便它可以成功编译和运行APP。

4.打开Android Studio，在工具栏中找到App选项，点击会弹出 Edit Configurations选项，点击进入，然后在设置页面中找到 Deploymeng target Options下的Target选项，然后选择为USB Device，然后点击OK使用USB进行真机调试。

### 4.1.2 移植FFmpeg到Android

FFmpeg是一个开放源代码包，它是在Linux操作系统上开发的，但是可以在其他操作系统上运行[8]。Andorid是一个嵌入式操作系统，源代码包不能在嵌入式操作系统上编译和运行。而Andorid的接口SDK是用Java语言实现的，所以基于Andorid的第三方应用开发应该使用Java语言，而FFmpeg源代码是C语言，所以运行FFmpeg来运行单一的Andorid系统是非常困难的。

当谷歌发布SDK时，它已经指出Android虚拟机Dalvik支持JNI编程。JNI编程是指开发人员开发的应用程序，可以使用JNI调用C语言动态库，这意味着可以使用Java+C编程模型实现第三方应用程序。用C语言开发的程序将被编译成动态链接库(.so)。如何打包。所以应用程序也是一个技术问题。谷歌提出了一个跨生产平台NDK，它帮助开发人员用Java代码编译so文件，大大减轻了开发人员的负担。

在本项目开发中采用Java封装FFmpeg命令行，而没有采用一般的Java+C语言来进行开发，使之在Android项目中轻松执行FFmpeg和FFprobe命令。采用开源项目Bravobit FFmpeg-Android加载FFmpeg库：

1．包括依赖项：

dependencies {

implementation 'nl.bravobit：android-ffmpeg：1.1.6'

}

2．检查是否支持FFmpeg：

if (FFmpeg.getInstance(this).isSupported()) {

// ffmpeg is supported

} else {

// ffmpeg is not supported

}

3．在此示例代码中，运行ffmpeg -version命令：

FFmpeg ffmpeg = FFmpeg.getInstance(context);

// to execute "ffmpeg -version" command you just need to pass "-version"

ffmpeg.execute(cmd， new ExecuteBinaryResponseHandler() {

@Override

public void onStart() {}

@Override

public void onProgress(String message) {}

@Override

public void onFailure(String message) {}

@Override

public void onSuccess(String message) {}

@Override

public void onFinish() {}

});

## 4.2 流媒体客户端的实现

用户需要通过客户端播放视频文件(包括本地文件和网络文件)。该部分包括4个模块：用户中心、视频中心、视频播放和视频裁剪。用户中心登录注册可以从开放源码中借用，以减少工作负载。本节主要介绍视频中心的UI实现、视频播放的实现以及视频剪辑的实现。

### 4.2.1 用户界面实现

用户对客户端的可用性有着强烈的需求，UI的控制和设计体现了美观和易用性。由于屏幕大小的限制，移动设备需要更多的UI接口。Android 系统中，各种 UI 控件都继承了 android.view.View 类[9]。

Android 事件处理机制包含几个方面：

1．事件：用户在图形界面上的处理，可以封装以类的方式出现。

2．事件源：事件发生的位置，即按钮，文本框等各个控件。

3．事件处理：接受并处理事件对象，并且事件处理通常实现特定的接口类。比如实现 android.view.View.OnClickListener 接口，对点击事件监听。

在用户主界面中，要将实现的功能简单明了的展示给用户，其最终用户启动主界面如图4.1所示：



**图4.1 用户启动主界面**

### 视频采集模块实现

1．设置权限

如果要在App中使用摄像头，必须先在Manifest中声明摄像头权限

<uses-permission android：name="android.permission.CAMERA" />

<uses-feature android：name="android.hardware.camera" />

在录制视频的时候，同时需要录制音频，所以也要声明录音权限

<uses-permission android：name="android.permission.RECORD\_AUDIO" />

2．自定义相机的一般步骤

1) 是否有摄像头、是否有访问摄像头的权限。

2) 创建相机预览类，继承SurfaceView并实现SurfaceHolder界面以预览相机的即时图像。

3) 使用相机预览类，构建将预览视图与界面控件组合在一起的预览布局。

4) 连接显示器的聆听者以拍摄照片或视频以响应用户的操作，例如按钮点击事件。

5) 拍摄照片和视频并保存输出。

6) 使用相机后，要正确释放相机以供其他应用程序使用。

3．录制视频

录制视频需要注意管理Camera以及MediaRecorder类。通过摄像头采集视频时，必须处 Camera.lock()和Camera.unlock()的调用以允许MediaRecorder访问摄像头设备，另外还有 Camera.open() 和Camera.release()的调用。

// 配置MediaRecorder

mRecorder.setAudioSource(MediaRecorder.AudioSource.CAMCORDER);

mRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);

mRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG\_4);

mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR\_NB);

mRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.H264);

视频录制结果如图 4.2 所示



**图4.2 视频录制界面**

### 4.2.3 视频播放模块实现

1. 本地视频播放功能实现
2. 获取本地视频路径

对于扩展存储设备 SD 卡内的文件的路径：

调用Environment.getExternalStorageDirectory()，则返回外置的SD的路径。

String uri=Environment.getExternalStorageDirectory().getPath()+"/geyou3.mp4";

VideoView video\_view = (VideoView)findViewById(R.id.view);//加载视频

video\_view.setVideoURI(Uri.parse(uri));

自Android 6.0起，Google就开始对系统权限施加严格的要求。 因此，开发人员需要调用权限应用程序的代码，弹出一个小窗口，并动态地向用户申请权限。 故读写手机内存文件也需要动态分配权限。

// 读与写的权限先定义到静态字符数组中：

private static String[] PERMISSIONS\_STORAGE = {

Manifest.permission.READ\_EXTERNAL\_STORAGE，

Manifest.permission.WRITE\_EXTERNAL\_STORAGE};

首先判断当前版本是否是Android6.0以及以上，如果是则确定它是否含有了写入文件的权限，如果没有则需要调用动态申请权限的代码，ActivityCompat.requestPermission方法的第一个参数是目标Activity，填写this即可，第二个参数是String[]字符数组类型的权限集，第三个即请求码：

if (Build.VERSION.SDK\_INT > Build.VERSION\_CODES.LOLLIPOP) {

if (ActivityCompat.checkSelfPermission(this， Manifest.permission.WRITE\_EXTERNAL\_STORAGE) != PackageManager.PERMISSION\_GRANTED) {

ActivityCompat.requestPermissions(this， PERMISSIONS\_STORAGE， REQUEST\_PERMISSION\_CODE);

}

}

回调函数，请求权限后调用onRequestPermissionResult函数，第一个形参为请求码，第二个形参是刚刚请求的权限集，最后一个参数是请求结果，其中0表示授权完成，-1表示授权错误：

@Override

public void onRequestPermissionsResult(int requestCode， @NonNull String[] permissions， @NonNull int[] grantResults) {

super.onRequestPermissionsResult(requestCode， permissions， grantResults);

if (requestCode == REQUEST\_PERMISSION\_CODE) {

for (int i = 0; i < permissions.length; i++) {

Log.i("MainActivity"， "申请的权限为：" + permissions[i] + "，申请结果：" + grantResults[i]);

}

}

}

对于本地文件路径存储在Android Studio文件夹中的文件：

将视频文件保存在raw文件夹下

String uri= "android.resource：//" + getPackageName() + "/" +R.raw.video1;

VideoView video\_view = (VideoView)findViewById(R.id.view);//加载视频

video\_view.setVideoURI(Uri.parse(uri));

1. 获取路径解码播放

播放控件使用Android自带的 VideoView 控件，采用MediaController媒体控制器实现播放。

//VideoView全局变量

private VideoView video\_view;

//将VideoView切成全局变量

video\_view = (VideoView) findViewById(R.id. video\_view);

//设置视频的来源

video\_view.setVideoURI(Uri.parse(uri));

//实列化媒体控制器

MediaController mediaController=new MediaController(this);

video\_view.setMediaController(mediaController);

mediaController.setMediaPlayer(video\_view);

video\_view.requestFocus();

video\_view.start();

本地视频播放实现效果如图 4.3 所示：



**图4.3 本地视频播放界面**

1. 视频点播功能

网络视频文件的播放必须通过网络通信进行控制，以实现与服务器的交互和网络媒体数据的传输。在将视频文件上传并存储到流服务器之后，流传输服务器可以响应客户端的相应按需请求。压缩编码过程发送给客户端用户后转换为HTTP的数据流。在客户端一侧再生的视频流通过取得 HTTP 流地址的VideoView 设定[10]。播放控件使用Android自带的 VideoView 控件，使用MediaController媒体控制器实现播放。

网络视频播放模块的核心代码：

//输入网络视频地址

mediaUri=Uri.parse(((EditText)findViewById(R.id.id\_internetUri)).getText().toString());

videoView1.setMediaController(new MediaController(InternetActivity.this));

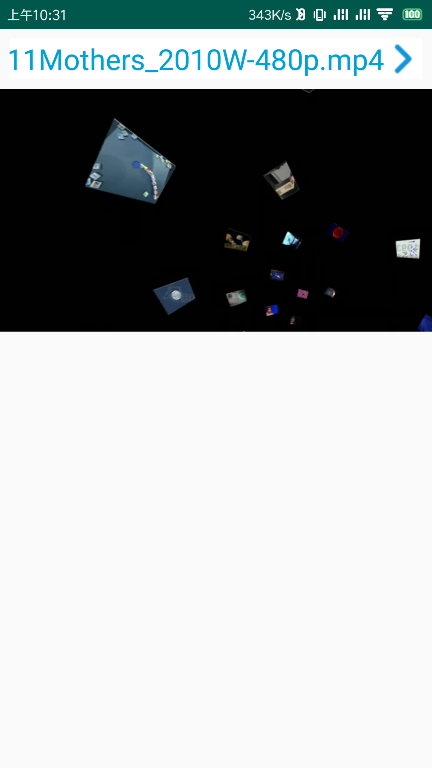
videoView1.setVideoURI(mediaUri);

videoView1.requestFocus(); // 获取焦点

videoView1.start();

客户端在收到流媒体服务器发送过来的视频流之后，经过一系列的各种组合解码操作，这个过程则必须采用底层Android的相关库。

视频点播结果如图 4.4 所示：



**图4.4视频点播界面**

1. 视频直播功能实现

Android Studio集成Ijkplayer，将其作为moudle导入我们需要使用播放器的工程project中，由于仅仅实现简单的网络直播播放功能，这里生成Fastvideoplay来将Ijkplay的功能进行简化。

IjkVideoView 这个类是使用Ijkplayer播放的View，这里为了操作方便，直接使用IjkVideoView作为直播的播放器。

之后在引用Fastvideoplay类时，直接调用其内置的功能即可实现：

try {

IjkMediaPlayer.loadLibrariesOnce(null);

IjkMediaPlayer.native\_profileBegin("libijkplayer.so");

playerSupport = true;

} catch (Throwable e) {

Log.e("GiraffePlayer"， "loadLibraries error"， e);

}

视频直播结果如图 4.5 所示



**图4.5视频直播界面**

1. 本地视频资源管理功能

在Activity Action中有一个“ACTION\_ GET\_ CONTENT”字符串常量。 此常量允许用户选择特定类型的数据并返回数据的URI。 我们利用这个常量，然后将类型设置为“video / \*”以获取Android手机中的所有视频。

Intent intent = new Intent();

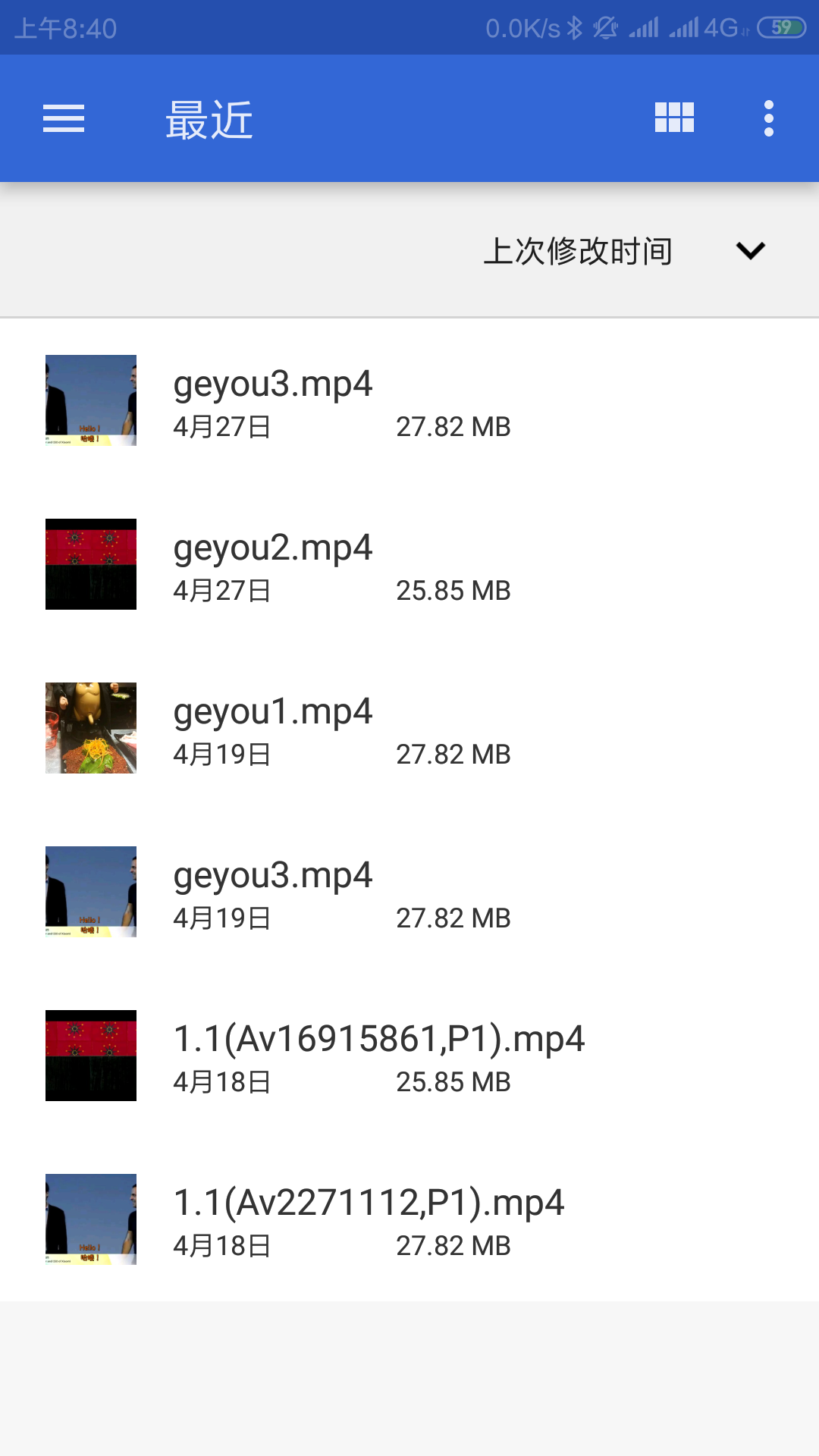
intent.setType("video/\*");

intent.setAction(Intent.ACTION\_GET\_CONTENT);

startActivityForResult(intent， 1);

之后从媒体管理器返回，利用mediaUri = data.getData()获取本地视频文件路径，然后在视频播放端使用VideoView进行播放。

本地视频资源管理结果如图 4.6 所示



**图4.6 本地视频资源管理界面**

1. 横屏竖屏功能实现

横竖屏功能的实现有两种方法：

第一种是横屏竖屏可以专门用两个xml文件分别对应竖屏和和横屏的界面，切换水平和垂直屏幕时，程序调用活动的 onCreate方法来加载相应的布局。切换屏幕时需要重新绘制Activity，这就需要用到两个系统自带的函数onRestoreInstanceState和onSaveInstanceState，前者恢复数据，后者切换屏幕销毁时保存数据。

另一种是设置权限，在AndroidManifest.xml中对Activity属性进行设置android：configChanges属性：

android：configChanges="keyboardHidden|orientation|screenSize"

该系统采用第二种方法，其横竖屏结果如图4.7所示：



**图4.7 横竖屏切换界面**

1. 视频播放动态切换全屏/原屏功能实现

第一种方法为：

设置全屏：

WindowManager.LayoutParams attrs = getWindow().getAttributes();

System.out.println("fullScreen的值：" + fullScreen);

if (fullScreen) {

attrs.flags &= (~WindowManager.LayoutParams.FLAG\_FULLSCREEN);

getWindow().setAttributes(attrs);

取消全屏设置：

getWindow().clearFlags(WindowManager.LayoutParams.FLAG\_LAYOUT\_NO\_LIMITS);

mPreferences.edit().putBoolean("fullScreen"， false).commit();

} else {

attrs.flags |= WindowManager.LayoutParams.FLAG\_FULLSCREEN;

getWindow().setAttributes(attrs);

getWindow().addFlags(

WindowManager.LayoutParams.FLAG\_LAYOUT\_NO\_LIMITS);

mPreferences.edit().putBoolean("fullScreen"， true).commit();

}

另一种方法则是在manifest里面配置：<activity android ： theme = " @ android：style / Theme.NoTitleBar.Fullscreen" />只在当前Activity内显示全屏，<application android：theme = "@android ： style /Theme.NoTitleBar Fullscreen"/>为整个应用配置全屏显示。

该系统采用第一种方法，其全屏/原屏切换结果如图4.8所示：

**图4.8 全屏/原屏切换结果界面**

1. 分享内容功能实现

这里采用的是Android 系统原生 API实现分享功能，首先创建一个 Intent ，指定其 Action 为 Intent.ACTION\_SEND，这表示要创建一个发送指定内容的隐式意图。然后指定需要发送的内容和类型。针对分享内容的不同，在代码实现上会有所差异，对于图像等文件的分享，要设置实现分享图片的路径。

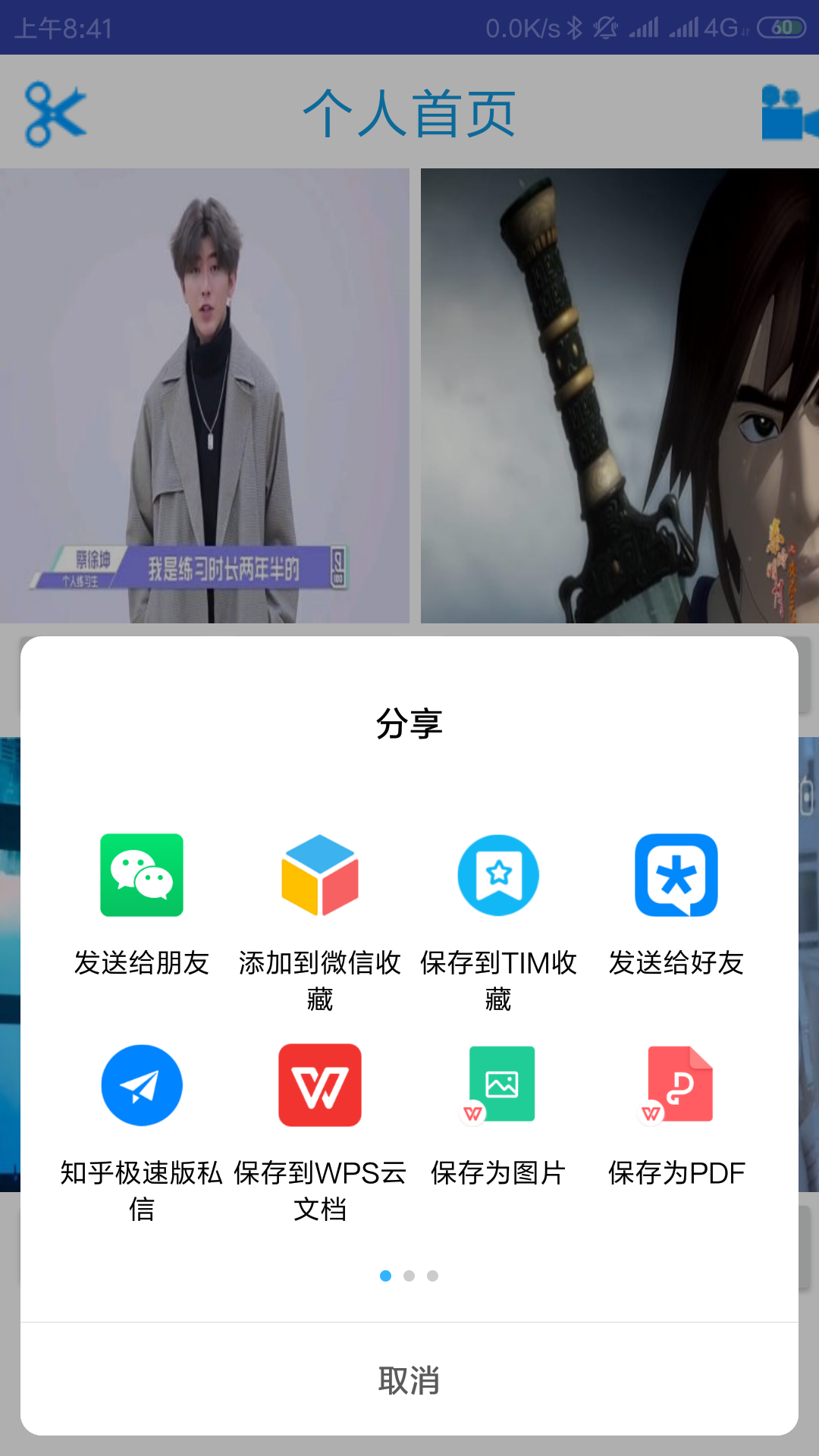
Intent textIntent = new Intent(Intent.ACTION\_SEND);

textIntent.setType("text/plain");

textIntent.putExtra(Intent.EXTRA\_TEXT， "这是一段分享的文字");

startActivity(Intent.createChooser(textIntent， "分享"));

分享内容结果如图 4.9 所示：



**图4.9 分享内容结果界面**

### 4.2.4 视频裁剪模块实现

1. 获取视频图片帧功能实现

利用FFmpegAndroid开发库可以直接用Java的API来执行ffmpge的命令行ffmpeg的commend命令。

在FFmpegInterface.java类中找到PAI后，看到参数String[] cmd即控制台的命令行参数，只要将我们的命令行传入，设置好回调，ffmpeg就能触发我们的回调了。

FFmpeg中提取图片命令为：

String cmd = "-y"+"-i"+ videoFilename+"-vframes"+ "1"+"-ss"， hour + "：" +

min+ "：" + sec+ "-f"+"mjpeg"+"-s"+ width + "\*" + height+"-an"， thumbFilename ;

String[] command = cmd.split(" ");

调用该FFmpeg命令行即可得到视频对应的某一帧。

1. 视频裁剪功能实现

视频帧的获取可以通过Android提供的MediaMetadataRetriever类来实现。

MediaMetadataRetriever mediaMetadataRetriever = new MediaMetadataRetriever();

mediaMetadataRetriever.setDataSource(context， videoUri);

获取视频的帧，帧其实就是图片，一帧可能会有N多张图片组成。通过调用MediaMetadataRetriever类中的getFrameAtTime()获取视频的帧的Bitmap。

public Bitmap getFrameAtTime(long timeUs， int option) {

if (option < OPTION\_PREVIOUS\_SYNC || option > OPTION\_CLOSEST) {

throw new IllegalArgumentException("Unsupported option： " + option);

}

return \_getFrameAtTime(timeUs， option);

}

视频裁剪的命令如下：

ffmpeg -ss START -t DURATION -i INPUT -vcodec copy -acodec copy OUTPUT

其中：

DURATION 是表示视频的持续时间而不是结束时间

INPUT 表示输入的包含文件名的文件路径 e.g： sdcard/xxx/xx/input.mp4

OUTPUT 表示输出的包含文件名的文件路径 e.g： sdcard/xxx/xx/output.mov

String cmd = "-ss " + start + " -t " + duration + " -i " + inputFile + " -vcodec copy -acodec copy " + outputFile + "/" + outputName;

String[] command = cmd.split(" ");

视频裁剪结果如图 4.10 所示：



**图4.10 视频裁剪结果界面**

# 第5章 移动流媒体系统测试与结果分析

本章中对实现的流媒体系统开展相应的软件测试，软件测试的目的是尽可能多的发现软件中存在的不足，最后只有满足测试内容的高质量的系统才能供用户去使用[11]。例如系统无法响应用户需求或者系统处于不稳定的状态，都会给用户带来使用上的不便。在本系统的软件测试采用的是黑盒测试的方法，针对的是把程序看成一个黑盒子，完全不考虑系统的内部工作过程，只检查程序功能能否按照设计的功能正常使用。

## 5.1 功能测试

在本环节中，功能测试主要是对前面系统设计和实现的功能模块进展相应的测试，通过相应的测试来发现各个功能在实现中的不足和缺点，以便进行相应的优化和改正。主要功能模块的测试环节及内容如下：

### 5.1.1 视频播放子模块测试

1．本地视频播放测试

**表5.1本地视频播放测试说明**

|  |  |
| --- | --- |
| 测试项目 | 本地视频播放 |
| 测试内容 | 视频文件是否能够播放 |
| 测试过程 | 点击播放按钮 |
| 测试重点 | 功能是否存在缺点之处 |
| 测试结果 | 选中的视频播放 |

2．网络视频播放测试

**表5.2 网络视频播放测试说明**

|  |  |
| --- | --- |
| 测试项目 | 网络视频播放 |
| 测试内容 | 网络视频是否能够加载进行播放 |
| 测试过程 | 输入网络视频地址，点击跳转按钮 |
| 测试重点 | 功能是否存在缺点之处 |
| 测试结果 | 网络视频加载成功可以进行播放 |

3．视频屏幕切换功能测试

**表5.3 屏幕切换功能测试说明**

|  |  |
| --- | --- |
| 测试项目 | 视频屏幕切换 |
| 测试内容 | 视频播放时能否实现原屏和全屏的动态转换 |
| 测试过程 | 播放过程中点击全屏按钮 |
| 测试重点 | 功能是否存在缺点之处 |
| 测试结果 | 原屏全屏切换正常 |

### 5.1.2 视频录制子模块测试

1．视频录制功能测试

**表5.4 视频录制功能测试说明**

|  |  |
| --- | --- |
| 测试项目 | 视频录制 |
| 测试内容 | 调用手机摄像头，能够录制一段视频 |
| 测试过程 | 先获取摄像头权限，再点击录制按钮，进行视频录制 |
| 测试重点 | 功能是否存在缺点之处 |
| 测试结果 | 能够正常录制视频 |

### 5.1.3 视频裁剪子模块测试

1. 获取视频某一帧图片测试

**表5.5 获取视频某一帧图片测试说明**

|  |  |
| --- | --- |
| 测试项目 | 获取视频帧图片 |
| 测试内容 | 从上传的视频中，截图视频的某一帧作为缩略图 |
| 测试过程 | 选取所需帧的时间，运行相关程序代码 |
| 测试重点 | 功能是否存在缺点之处 |
| 测试结果 | 相应帧图片获取成功 |

1. 视频裁剪功能测试

**表5.6 视频裁剪功能测试说明**

|  |  |
| --- | --- |
| 测试项目 | 视频裁剪 |
| 测试内容 | 裁剪一定长度的视频，存放到本地 |
| 测试过程 | 将视频拖动到裁剪位置，选取所需视频，然后进行剪切保存 |
| 测试重点 | 功能是否存在缺点之处 |
| 测试结果 | 裁剪相应视频内容成功 |

## 5.2 性能测试

与普通播放器相比较来说，流媒体播放器最主要的不同点在于其能够实现实时的视频内容的播放，用户可以一边加载一边观看，不需要一次全存储完全部内容。所以，对于移动流媒体播放系统来说，在性能测试中，最重要的是检查系统的稳定性和实时性这两方面。

对于稳定性来说，在软件测试的环节，系统保持稳定正常运行，满足基本需求。

对于实时性来说，在校园Wifi正常的状况下，通过腾讯网络助手测得平均网络传播速度为263.8KB/s，网络视频在正常观看前，只需要5~6秒的加载时间。在播放过程中，播放流畅，没有间断，没有卡顿。表5.7为性能测试说明。

**表5.7 性能测试说明**

|  |  |  |
| --- | --- | --- |
| 性能 | 预期值 | 实际值 |
| 安装内存占用 | 30 | 25.6 |
| 播放内存占用 | 150 | 110 |
| 中央处理器占用 | 15% | 13.4% |
| 后台内存占用 | 50 | 35 |
| 电量消耗(1h) | 15% | 8% |
| 软件启动所需时间 | 3 | 2.4 |
| 网络视频加载所需时间 | 6 | 5.5 |

## 5.3 结果分析

在已知的移动流媒体系统功能和特点的基础上，对移动流媒体播放系统进行了功能测试和性能测试。在软件测试过程中系统运行较为稳定，未发现严重性的 问题。在加载速度，占用内存，耗用时间等方面都在用户能接受的范围内，能够正常工作，满足上线要求，符合功能需求。

# 结论

一、技术分析

随着移动互联网的高速发展，人们对于移动端视频的需要和要求越来越高和越来越多。Android 移动操作系统因为源代码开源且市场需求量广大被选为本文应用的操作系统；此外，本篇文章中在视频直播功能模块中，集成了开源播放器Ijkplayer，引入Ijkplayer的播放视频的控件IjkVideoView来进行直播平台的搭建；同时在视频裁剪子系统中使用了开源框架 FFmpeg，通过分析FFmpeg的代码结构和其提供的操作行命令，来对系统的功能完成开发和实现。

在整个流媒体系统的研究和设计过程中，重点实现了以下的内容：

1．了解流媒体发展的背景，对国内外移动流媒体的现状进行分析和研究，做好知识、技能等相关准备，说明系统的研究内容和研究方向。

2．理解系统关键性技术，了解移动流媒体服务体系相关内容和工作原理，掌握移动流媒体中RTP、RTSP 和 RTCP 等网络传输协议，分析Android系统的特点和功能。

3．对系统进行需求分析，首先对于系统的功能需求来说，介绍系统要实现的各个功能，而对于系统的性能需求来说，是为了让系统满足用户的要求，以方便上线工作。

4．根据需求对系统进行总体设计，然后对各个部分进行具体的、详细的设计。对系统的功能模块进行设计，将实现的功能具体化到各个模块，对实现的模块进行流程分析和步骤分解。

5．以系统需求和系统设计为蓝图，采用Android Studio、Windows系统等开发工具，通过移植FFmpeg到Android上来完成系统客户端以及服务器各模块的实现，重点实现视频播放模块和视频裁剪模块。

6．搭建系统测试环境，通过丰富的测试用例来开展功能测试，并检查系统的完整性和实时性来实现性能测试，分析测试结果。

二、非技术分析

本系统中，针对目前社会上4G以及5G的快速发展，对系统所涉及的社会效益、文化效益等相关技术因素进行思考。

1．在经济效益方面，本系统实现的客户端可以为用户提供视频直播以及视频点播，在4G以及5G发展的大潮中，具有广大的市场前景，能够为用户提供良好的播放效果，收到用户支持。

2．在文化效益方面，本系统实现的客户端在视频播放中可以弘扬社会主义核心价值观，给用户提供技能培训，课程学习等方面的视频，丰富人们的文化生活。

3．在学习能力培养方面，后期可以提供弹幕发送等增强用户体验的功能，可以使用户在观看学习视频时发送弹幕来方便交流，使用户在学习时相互督促。

作为一名未来的软件工程师，要学会理论与实践并重，去用工程师的眼光来分析事情，遵守职业道德，去做有利于国家和人民的工作，用自己的能力去为社会发展做出自己的奉献，在历史的大潮中，用技术去为中华民族的伟大复兴发一分光，尽一份力。

三、展望

对于测试结果的数据分析，流媒体系统已实现了基本的功能要求和性能要求，但由于各种因素的干扰，系统中仍存在一些问题。经过仔细观察和分析结果，可以从以下几个方面进行加以完善：

1．该系统主要用于移动视频播放。客户界面的美观，易操作性和影响用户体验的更丰富的个性化功能（如视频搜索和弹幕实现）需要进一步优化和改进。

2．该系统可以实现类似于自动爬取各大网站上最流行的视频文件，然后发送给用户。此外，还可以发展允许用户上传创作视频内容。

3．该系统可以实现对在线直播视频相关内容的裁剪，使用户在观看直播的同时截取自己感兴趣的内容保存到本地。

4．该系统可以发展用来支持更多的移动操作系统，如 iOS，Window mobile，BlackBerry等，并可以与HTML5 技术结合实现跨平台操作，使整个系统在兼容和扩展上有更好的发展。

5．该系统中的视频文件可能会在不同的手机机型，系统以及网络环境中发生不同的结果。所以，在试验中应充分认识到各种原因的干扰，并应在多种环境的综合下进行更全面的试验，以获得更完整的试验结果。 随着4G网络以及5G网络的向前发展，人们对移动视频的希冀必然会上升，希望有更好的播放质量，更快的加载速率。因此系统有很大的潜力，特别是面对各个用户间的不同需求。

# 参考文献

[1] 吴伟. UMTS移动流媒体业务及其关键技术[J]. 电信网技术, 2005(9):8-11.

[2] 移动互联网蓝皮书《中国移动互联网发展报告(2018)》在天津正式发布[J]. 信息技术与网络安全, 2018(7).

[3] 邬贺铨. 5G时代的移动多媒体——在BIRTV2018主题报告会上的演讲[J]. 现代电视技术, 2018, 207(09):21-22+55.

[4] Giordani M , Polese M , Roy A , et al. [IEEE 2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net) - Capri, Italy (2018.6.20-2018.6.22)] 2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net) - Initial access frameworks for 3GPP NR at mmWave frequencies[J]. 2018:1-8.

[5] 杨小明. 基于Android流媒体播放器的设计与实现[D]. 南京邮电大学,2012.

[6] 李刚. 基于Android平台的智能手机流媒体播放器的研究及实现[D]. 南京邮电大学.

[7] 杨光平. 基于Android的移动流媒体实时传输系统设计与实现[D]. 西安电子科技大学, 2012.

[8] 解贺亮. 基于Android平台的流媒体播放器的设计与实现[D]. 电子科技大学.

[9] Designing challenge questions for location‐based authentication systems: a real‐life study[J]. Human-centric Computing and Information Sciences, 2015, 5(1):17.

[10] Shiddiqi AM, Pratama P, Ciptaningtyas HT. Video Streaming Application Using Mobile Media Application Programming Interface. TELKOMNIKA: Indonesian Journal of Electrical Engineering, 2010; 8(3): 293-300.

[11] 张海藩,牟永敏. 软件工程导论（第6版）[M]. 清华大学出版社, 2013.

# 致 谢

转眼间，我在大连海事大学已经度过了人生中最美好最青春的四年时光。我很感恩在这里遇到了谆谆教诲的老师们和团结友爱的同学们，很感谢母校提供给我们良好的学习环境，向上的学习氛围，也很幸运能见证海大的110周年校庆。在这里我从一个懵懵懂懂的少年成长为一个对自己人生负责的青年。

在这里我首先要感谢我的指导老师张海昕老师，张老师在学术和技术方面给了我很大的帮助，他治学严谨，学识渊博，诲人不倦。不仅如此，他那平易近人的人格魅力也对我有着潜移默化的影响。在此，感谢张老师对我论文的指导，感谢这三年来对我的培养与教育。

特别感谢我的父母，是他们在我成长过程中不断付出心血与无微不至的关怀才有了今天的我。

感谢张志恒、郭彦均、刘宇轩、于林啸雄、陈诚等同学，他们在学习中给了我大量的启发和帮助，同时给四年的本科生活带来了很多乐趣。

还要感谢在我本科四年指导我教诲我的所有老师们，感谢你们在学习和生活上对我的帮助。

最后，向审阅本文和参与答辩的各位专家评审老师们表示最诚挚的感谢!