Software Engineering and Applications 软件工程与应用, 2013, 2, 104-108
http://dx.doi.org/10.12677/sea.2013.25019 Published Online October 2013 (http://www.hanspub.org/journal/sea.html)

Real-Time Video Transmission and Decoding Based on Android Mobile Phone

Xiaojia Wu, Hua Ye, Ya Su, Yanlan Yang

School of Automation, Southeast University, Nanjing
Email: wuxiaojiaemma@163.com

Received: Sep. 1st, 2013; revised: Sep. 11th, 2013; accepted: Sep. 17th, 2013

Abstract: This paper discusses video transmission and decoding of the Android mobile phone video surveillance application. Mobile client communicates with server through Socket to receive video data, and restructures the video data to a video frame according to the frame number. This paper designs to use frame dropping mechanism to solve the problem that undecoded video frame filling the buffer because of the phone's weak processing power. Then, it completes real-time decoding and image format conversion of the video data in H.264 format by FFmpeg codec library which was migrated to the Android system. The test results show that the software runs well, the picture is clear and smooth.

Keywords: Video Transmission; H.264 Decoding; Android; Video Surveillance

基于 Android 手机的实时视频传输和解码

吴晓佳，叶　桦，苏　雅，仰燕兰

东南大学自动化学院，南京
Email: wuxiaojiaemma@163.com

收稿日期：2013 年 9 月 1 日；修回日期：2013 年 9 月 11 日；录用日期：2013 年 9 月 17 日

摘　要：本文主要探讨了 Android 手机视频监控软件的视频传输和解码。手机客户端通过 Socket 与平台服务器通讯，接收视频数据，并按照帧号重组成视频帧。本文设计采用丢帧机制解决了由手机处理能力弱导致待解码视频帧占满缓存区的问题，然后通过移植到 Android 系统中的 FFmpeg 解码库，实现了 H.264 格式视频数据的实时解码和图像格式转换。经测试结果表明软件的运行良好，画面清晰流畅。

关键词：视频传输；H.264 解码；安卓操作系统；视频监控

1. 引言

视频监控作为安防系统的重要组成部分，在民用安全、城市交通和重大事件中发挥着巨大的作用，越来越受到人们的重视。传统的视频监控系统大多通过线缆或光纤将视频信号传输到监控中心，但由于网线的限制，不利于监控系统的迅速搭建[1]，人们只能在一定设备环境下进行监控。随着信息化程度的提高，人们已经不满足于这种视频监控模式，迫切需要一种更加方便，实时的视频监控模式。Android 系统的产生和 3G 网络的普及为此提供了技术支持。Android 是 Google 公司推出的基于 Linux 的开源操作系统，凭借其良好的稳定性、开放性和可移植性，广泛应用于智能手机和平板电脑[2]。3G 网络的普及应用为流媒体无线传输业务提供了保证。本文在 Android 操作系统

的智能手机上，利用无线网络通信技术，H.264 编码技术和 FFmpeg 视频解码解决方案，研发了一个视频监控应用软件，正好满足了这一需求。用户只要随身携带手机，在有无线网络的情况下就可以通过本软件随时，随地查看监控画面，更加快捷，方便，具有较大的应用前景。本文主要探讨了 Android 手机视频监控软件的视频传输和解码。

2. 整体方案设计

2.1. 系统结构

本系统由摄像头，监控终端，平台服务器和智能手机客户端四个部分组成，系统结构如图 1 所示。摄像头负责拍摄采集视频数据，传输到终端；摄像头带有云台，可以在上下左右四个方向的转动，并具有调焦功能。一个监控终端可以同时控制几个摄像头，一个摄像头即为一路视频。监控终端将采集到的视频数据按 H.264 格式编码后上传到平台服务器，并控制着摄像头的转动和调焦。平台服务器作为终端和手机客户端之间的沟通中介，负责将终端上传的视频数据转发到手机客户端；同时将手机客户端的指令转发到终端。手机客户端主要负责接收和解析来自平台的数据包，解码 H.264 视频帧后，将视频画面播放显示，也可以根据用户需求发出相应指令控制摄像头转动和调焦。

2.2. 软件架构

基于 Android 的手机视频监控软件的架构如图 2 所示，由视频解码，视频显示和通讯三个模块构成。其中通讯模块负责与服务器之间的数据收发。主要分成三个部分，一是接收来自服务器的所有数据，并对数据进行处理，当解析到设备信息，推送到用户界面显示，当解析到视频数据则将数据存储到视频缓冲区中；二是将用户操作产生的控制命令传送到服务器端；三是向服务器发送和接收心跳，保持手机客户端和服务器之间的稳定连接。视屏解码模块负责从缓冲区中读取视频数据，送入解码器进行解码，并调整解码后图像的分辨率和格式适应不同型号的手机。视频显示模块将解码获得的图像绘制到界面上实现视频显示。

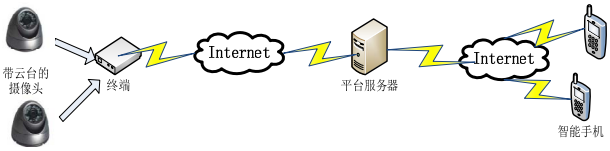

Figure 1. System structure diagram
图 1. 系统结构图

3. 通讯模块实现

3.1. 通讯协议

手机视频监控最重要的一个性能指标就是实时性。因此，在选择网络通信协议时必须着重考虑这个因素。TCP 和 UDP 是目前最为常用的两种网络通讯协议。TCP 是一种面向连接的、可靠的、基于字节流的运输层通信协议，服务器和客户端的传输数据之前必须先建立连接。TCP 提供超时重发，丢弃重复数据，检验数据，流量控制等功能，保证数据能从一端传到另一端。用户数据报协议(UDP)是一种无连接的传输层协议，UDP 在传输数据报前不用在客户和服务器之间建立一个连接，且没有超时重发等机制，故而传输速度很快，同时它不提供可靠通信保证，数据包可能会丢失。分析比较两者的特点，本软件在 Android 手机客户端和平台服务器之间采用 UDP 协议连接，以保证视频传输的实时性，提高视频传输速度[3]。

UDP 应用在网络通讯中，首先要建立套接字 DatagramSocket，通过 DatagramSocket 就可以接受和发送 UDP 数据报(DatagramPacket)。DatagramPacket 中包含了要发送的数据信息和目的地址的信息，DatagramSocket 根据该信息把 UDP 数据报发送到目的地址[4]。

3.2. 视频数据处理

3.2.1. 视频组包

在视频传输的过程中，关键帧的数据量是非常大

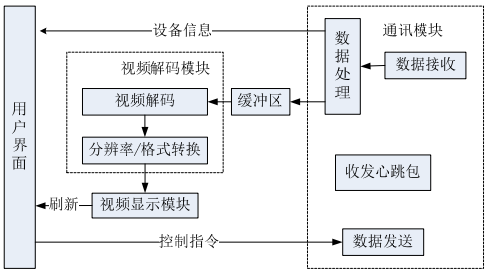

Figure 2. Software architecture diagram
图 2. 软件架构图

的，根据视频分辨率和码率的不同可以达到几万甚至几十万个字节，不可能在一个数据包中传输，往往需要分成独立的 NALU 单元后封装在几个数据包中发送。因此，在接收到数据后，手机客户端需要将收到的数据包解析，然后把帧号相同的视频数据重新组成一帧。客户端在接到第一帧数据的时候先解析出帧号，包号(当一帧数据被分成几包，包号表示此数据包是一帧图像的第几包)和视频数据等信息，然后将包号和视频数据信息保存到构建的 Package 对象中，并将 Package 对象存储到缓冲区，记录下当前帧号为 P_FrameId 后，开始接收下一包数据。当接收到的新数据包的帧号与 P_FrameId 相同，表明视频数据是同一帧的数据，将其以同样的方式存储到缓存区中，当收到新的数据包的帧号与 P_FrameId 不同时，就认为上一帧的数据已经收完，就可以把缓冲区中的数据组包。将缓冲区清空后，再将收到的数据包信息存入缓冲区，同时更新 P_FrameId。

```
Package{
Int P_Id;//包号
Byte[] P_data;//视频数据
}
```

服务器发送的视频数据包在网络传出的过程中时，由于网络不稳定等因素有可能出现数据包乱序到达的情况，因此在数据包组包的过程中，设计采用冒泡算法根据包号将数据包排序。

3.2.2. 丢帧机制

由于 CPU 处理能力和内存的限制，手机解码速度可能滞后于接收到视频数据的速度，视频缓冲区也不能过大，当过多的视频数据滞存在缓冲区中时，不仅会占满视频缓冲区，也会影响视频监控的实时性。为解决这个问题，本文采取了丢帧机制，视频的帧率为 15FPS，软件设计视频缓冲区最多可以容纳 30 个视频帧，当视频帧占满缓冲区时就要丢弃一部分视频帧。以前的一些视频监控软件采用的丢帧方式是当发现缓冲区满以后，将新收到的视频帧丢弃。这种丢帧方式虽然也能保证软件正常运行，但是没有考虑到编码方式的特征，影响了手机的实时性，也可能造成一段时间的监控盲区。本文采取的丢帧机制是，丢弃较早到达的视频帧，容纳新的数据来保证视频监控的实时性。同时考虑监控终端采用的 H.264 视频编码标准，在编码过程中只使用了 I 帧和 P 帧。其中 I 帧是关键帧，可以独立解码；P 帧是参考帧，是这一帧跟之前一个关键帧(或 P 帧)的差别，在接收到 I 帧的基础上，P 帧才能进行解码。从两者的重要性考虑，首先丢弃 P 帧，再丢弃 I 帧。在既定的缓冲区中保留更多的可用视频帧，具体处理流程如图 3。

3.3. 心跳模块

手机视屏监控要求具有稳定性，为保证手机客户端和平台服务器之间长时间稳定的连接，本文设计采用心跳模块。心跳功能由 Service 实现，Service 是 Android 的服务组件，没有用户界面，在后台运行，对用户完全透明[5]，不影响前台界面的正常运行。

在使用 Service 前先要在 Androidmainfest.xml 文件中进行注册，注册的代码如下：

```
<service
    android:name = "com.video.service.HeartbeatServi
                    ce "
    android:enabled = "true"
    android:exported = "false"
    android:label = "HeartbeatServiceOFvideo">
    </service>
```

在登陆到平台以后调用 context.startService()函数开启 Service，在 Service 中开启一个线程，定时向平台发送心跳包，平台在收到心跳包后给以回复，当心跳线程连续三次没有收到回复的时候，向广播接收组件(BroadcastReceiver)发送广播告警信息，Broadcast Receiver 在收到广播信息后，调用 context.stopService()

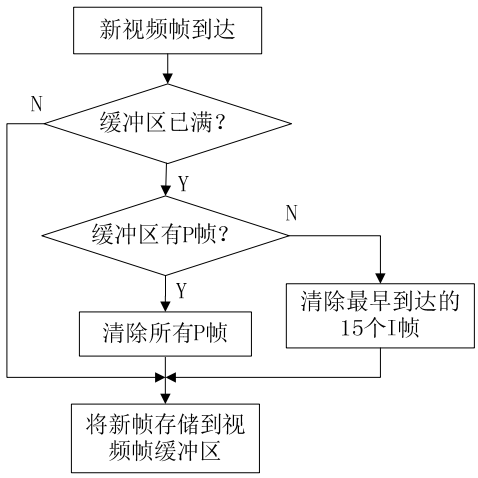

Figure 3. Frame dropping mechanism
图 3. 丢帧机制

函数将 Service 结束，然后跳转到登陆界面并给出断线提醒，用户可以重新连接平台。

当手机客户端在接收视频数据的时候，客户端和服务器之间有很频繁的数据交互，不需要用心跳包来来维持与服务器之间的连接，心跳包的存在反而会给客户端和服务器都带来负担，因此，在接收视频数据的时候，客户端不再验证心跳包，直到视频数据终止传输。

4. 视频解码模块实现

4.1. FFmpeg 解码库移植

Android 平台自身的多媒体框架 OpenCore 只支持 3gp、MP4 两种视频格式，无法解码终端编码的 H.264 格式视频。同时 Android 没有开放 OpenCore 的底层视频编解码 native API，利用 Java 来处理有关音视频编解码的工作显然力不从心，所以本文移植 FFmpeg 来完成解码工作[6]。FFmpeg 是一个集录制、转换、音/视频编码解码功能为一体的完整开源解决方案[7]。FFmpeg 是用 C 语言编码的，但是在 Windows 平台下，Android 环境不能直接支持 C 语言的开发。因此，还需要利用 Android NDK 将其编译为动态链接库后才能在 Android 平台下进行开发。

4.2. 视频解码

视频解码模块基于 FFmpeg 底层解码函数调用，由 C 语言编程实现的；主要负责从缓冲区中读取视频帧，将其进行解码，并将解码后的视频数据交送视频显示模块。监控终端上传的视频是 H．264 视频编码的分辨率为 352 × 288，YUV420 格式的视频帧。针对本项目在 FFmpeg 解码方案的基础上设计 H.264 解码器。

H.264 解码器主要成为四步：初始化解码器，解码视频帧，视频分辨率/格式转换，释放解码器。

初始化解码器的主要工作是调用 av_register_all()注册多种音视频格式的编解码器及各种文件；针对视频编码方式的单一性，通过 avcodec_find_decoder (CODEC_ID_H264)将解码器设置为 H.264 模式，并为解码器和解码过程中需要用到的结构分配内存，最后打开解码器。

解码视频帧所用的函数是 avcodec_decode_video2()，其返回值表示解码出一幅图像所消耗的数据帧的字节长度，由此可以判断解码是否成功，当函数返回值为−1 时，表示解码不成功。解码成功后获得的图像存储在 AVFrame 结构中，而图像的大小和格式信息则在 pAVCodecCtx 结构中。

视频分辨率/格式转换是必不可少的一个步骤，Android 手机只支持 RGB 格式的图像显示，平台传送过来的帧解码后是 YUV 格式的视频数据，只有转成 RGB 格式后才能在手机上播放；源视频的分辨率为 352 × 288，手机型号种类繁多，分辨率也各不相同，为适应不同型号的手机，获得更好的视觉效果，也需要将对视频的分辨率做调整。FFmpeg 中提供了 sws_scale()函数来进行图像缩放和格式转换。转换的过程中还用到了一个很重要的结构 SwsContext，在转换前需要在此结构中指明原图像和目标图像的分辨率大小和视频格式，以及转换过程中采用的算法等信息。

释放解码器是在解码完成后的必要操作，初始化解码器时所分配的内存到解码完毕后都要释放掉。不及时的内存释放会导致程序出现内存泄露而崩溃。

视频解码模块的主要流程可以从图 4 中看到。

数据帧通过解码和格式转换后，以 RGB 格式存储在数组中。本软件将 RGB 数组转成 Bitmap 后采用 SurfaceView 作为绘图容器将图像画在手机屏幕上。SurfaceView 可以直接从内存或者 DMA 等硬件接口取

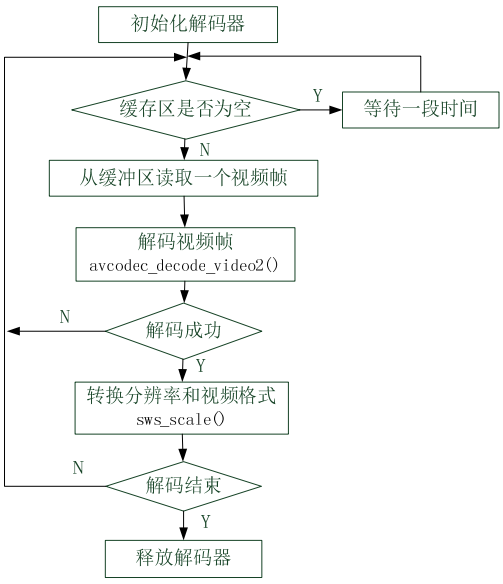

Figure 4. Flowchart of decoding
图 4. 解码流程图

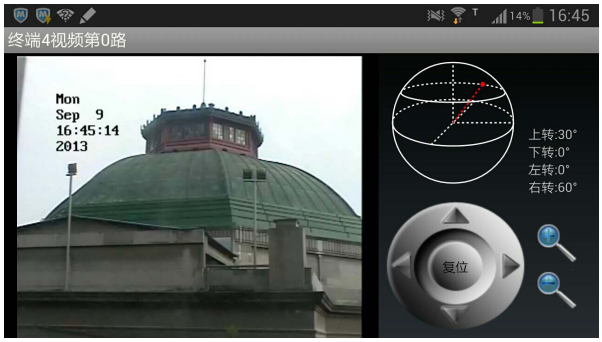

Figure 5. Interface of monitoring software
图 5. 监控软件界面

得图像数据，具有双缓冲机制，效率高，能够实现良好的显示效果。

5. 软件运行

本软件在型号为三星 GT-N7108 的手机利用 WIFI 无线网络运行并测试。经测试，本软件在视频为 15 帧/s，码率 256 bit/s 的条件下能够获得良好的运行结果，视频流畅播放，有很好的实时性和稳定性，满足设计的基本需求。视频监控软件的运行效果如图 5。

6. 总结

本系统在当前主流的智能手机操作系统——Android 平台上移植了 FFmpeg 解码库，开发出一个视频监控应用，实现了视频数据接收，视频解码显示以及云台控制等功能。本文详细分析了软件架构设计，主要论述了视频数据接收和视频解码模块的设计和实现。在数据处理过程中，不同性能的手机解码能力存在差距，本文采用丢帧机制解决了由手机处理能力弱导致待解码视频帧占满缓存区的问题，提高了软件的适用性，实时性和稳定性。最后，本文探讨了 H.264 视频解码器的实现。软件实现了友好的操作界面，具有良好的稳定性和实时性，满足了当前市场的需求，具有较大的市场潜力。

参考文献 (References)

[1] 李昂, 宋海声, 苏小芸. 基于 Android 的视频监控系统设计与实现[J]. 电子技术应用, 2012, 38(7): 138-143.
[2] 宋强, 齐贵宝, 宋占伟. 基于 Android 系统的 H.264 视频监控设计[J]. 吉林大学学报, 2012, 30(3): 272-278.
[3] 赵飞, 叶震. UDP 协议与 TCP 协议的对比分析与可靠性改进[J]. 计算机技术与发展, 2006, 16(9): 219-221.
[4] 张雷. 基于 Android 的物联网音视频通信系统关键技术研究[D]. 南京邮电大学硕士论文, 2012.
[5] 苏雅, 仰燕兰, 吴晓佳, 叶桦. 基于 Android 系统的手机定位软件的设计与开发[J]. 计算机科学与应用, 2013, 3: 17-22.
[6] 摆云. UDP 协议与 TCP 协议的对比分析与可靠性改进[J]. 计算机技术与发展, 2006, 16(9): 219-221.
[7] 杨钊. 基于 Android 的视频采集系统的设计与实现[F]. 西安电子科技大学硕士论文, 2012.