基于 RTSP 服务器的望远镜流媒体实时传输系统的设计

王超　王昊京
（中国科学院长春光学精密机械与物理研究所　吉林省长春市　130033）

摘　要：本文针对望远镜的观测需求，设计并实现了一种能够用于望远镜实时流媒体数据传输的 RTSP 服务器系统。系统能够对相机图像进行实时的 H.264 编码，并且将编码后的数据进行 RTSP 推送。本文重点介绍了 H.264 实时编码及 RTSP 服务器的设计与实现。通过实验验证该系统能够对相机设备进行实时图像数据推送，并能够允许多个用户并发访问推送数据。

关键词：RTSP 服务器；流媒体实时传输；H.264

在传统的望远镜观测系统中，图像数据通过 JPEG 格式进行传送。由于压缩技术的限制，图像的压缩比例低，占用的存储空间和带宽都比较大。随着望远镜观测系统获取图像数量的增加和实时性传输需求的增强，使用 JPEG 压缩技术已经无法满足系统需求，因此将流媒体实时传输技术应用到望远镜的图像传输系统中具有很好的实用性。

本文设计了一种用于实时观测相机图像的流媒体数据传输系统。系统利用 OpenCV 进行视频采集，将采集到的数据进行 H.264 压缩编码，最后将编码数据通过 RTSP 服务器进行实时流媒体推送。

1 实时传输系统技术介绍

1.1 RTSP协议

实时流媒体协议 RTSP（real time stream protocol）是一种应用广泛的实时流媒体传输协议，主要负责处理服务器和客户端之间的数据通信，能够建立和控制媒体的时间同步流。RTSP 协议主要由两部分构成，分别是 RTP 和 RTCP。其中 RTP 协议用来传输媒体数据，RTCP 则是负责在 RTP 传输过程中提供传输控制信息 [1]。RTSP 协议本身不具有流媒体推送的能力，主要依靠其中的 RTP 和 RTCP 来进行数据的传输。在网络结构中，RTSP 属于应用层协议，需要相应的链路层协议配合才能进行数据的推送。

1.2 H.264

H.264 编码技术是一种高效视频压缩编码技术，能够支持流媒体数据进行分组传输 [2]。H.264 技术具有较高的编码压缩率，在相同带宽下，用其进行压缩编码能够传输更多的数据。同时 H.264 技术具有较强的抗干扰性，能够适应不同的网络环境，常用于数据流的传输过程，尤为适合图像的传输。经压缩编码后，图像的帧结构由 NALU 组成。每个 NALU 中都分为两部分，一部分是 NAL 头文件信息，用以储存数据传输的相关信息。另一部分是原始图像编码后的数据信息。

2 望远镜流媒体传输系统的实现

2.1 系统的整体设计

流媒体实时传输系统采用客户端 - 服务器工作模式。系统中的 H.264 编码模块选用 FFmpeg 中的 H.264 编码库。网络传输协议选用 TCP 协议。流媒体传输系统的整体结构如图 1 所示，主要包括三部分：视频采集编码模块、流媒体服务器模块和流媒体客户端模块。

前端模块主要负责图像的获取和处理，在功能上分为两部分，分别是图像采集和 H.264 编码器。两部分各占用一个线程单独进行处理，这样可以保证在获得图像信息后，能够实时的进行 H.264 编码。图像获取过程使用 OpenCV 对相机的原始图像信息进行采集，采集到的每一帧图像都会被转存为 cv::Mat 格式，便于接下来编码器的编码操作。帧数据在进入编码器之前会经过数据格式转换，该过程将每一帧的图像信息提取出来，通过色度空间转换转变为 YUV420 格式，之后再进入编码器进行 H.264 编码。编码器通过调

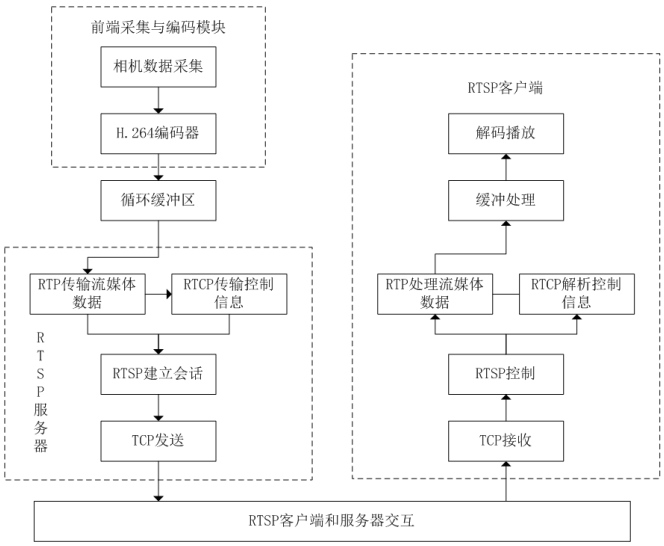

图 1：流媒体传输系统整体架构

用编码函数，对图像进行压缩编码，然后将数据存储到内存中等待 RTSP 服务器调用。

RTSP 服务器通过指针调用内存中的编码数据。在接收到数据后，RTSP 服务器首先对数据格式进行判别处理，然后将 H.264 数据进行转换操作，再将图像数据封装成 RTP 数据包、将控制信息封装成 RTCP 协议支持的数据包，最后使用 RTSP 服务器对外推流。

如图 2 所示为 RTSP 客户端和服务器的交互过程。客户端首先向服务器发送 Options 请求报文，等待服务器响应。待服务器对此进行响应后，客户端会向服务器端发送 Describe 请求报文、Setup 请求报文，服务器通过接收这些报文，逐步建立与客户端的连接 [3]。在客户端与服务器建立连接的过程中，客户端会向服务器端发送 Play 请求，即请求服务器进行 RTSP 数据推送。在对该请求进行响应后，RTSP 服务器便会进行 RTP 和 RTCP 的推送。推送完成后，客户端和服务器通过 Turndown 报文结束连接 [4]。

为了保障数据传输的可靠性，本系统在数据链路层使用 TCP 协议进行数据发送。客户端和服务器在 SETUP 过程中确定了使用 TCP 协议进行后续的 RTSP 数据推送工作。当流媒体数据进行传输时，服务器不会构建新的 TCP 连接，而是直接使用此前通信过程中的 TCP 通道。服务器会对 RTP 数据包进行封装处理，在原 RTP 包的基础上添加辨识符、通道号和 RTP 包的大小等信息，然后经由 TCP 通道发送给客户端，发送示意图详见图 3。

2.2 编码器的基本框架

本系统的编码器主要以 FFmpeg 中提供的 libx264 为基础进行图像信息的 H.264 压缩编码。前端视频采集装置在获得图像信息后，经过色度空间转换，再通过指针传入到 H.264 编码器中。H.264 编码器对接收到的图像数据进行判别和预处理，并将图像数据存储在

AVFrame 结构体中，然后通过调用 avcodec_encode_video() 函数进行 H.264 编码 [5]。编码后的数据存放在 AVPacket 结构中，待 RTSP 服务器进行推送。

2.3 RTSP服务器的设计与实现

RTSP 服务器是流媒体实时传输系统的核心部分，其主要职能是对图像数据进行 RTSP 封装处理及推流。RTSP 服务器的主要功能为客户端和服务器之间的通信、构建流媒体数据、发送 RTSP 流媒体数据。RTSP 服务器的基本设计流程如图 4 所示。

RTSP 服务器在运行时，首先会监听当前状态下的 TCP 连接情况，当监听到来自客户端的连接请求时，服务器会建立新的 socket，并绑定相应端口号。然后服务器进入 RTSP 的事件循环中，待接收到客户端发送的 connection 信息后，与客户端进行连接 [6]。

在等待连接的同时，RTSP 服务器会开辟新的线程来构建流媒体传输数据。RTSP 服务器在接收到来自编码器的数据后，自身会构造一个 AVFrame 类型的帧结构，用来进行后续的流媒体推送。该帧结构主要包括图像编码数据、编码器的头文件信息、P 帧、I 帧、时间戳等。

在与客户端连接成功后，RTSP 服务器通过解析接收到的信息，获取流媒体 ID，建立流媒体会话，通过会话传递流媒体数据。在会话建立后，系统会创建数据推送地址，建立 TCP 连接，并构建动态监听器，监听来自客户端的返回值。若返回值正确，便开始进行图像数据的 RTSP 推送。待推送结束后，服务器关闭 RTSP 数据推送，移除流媒体会话。

3 流媒体的实时获取与处理

实时性对于望远镜流媒体传输系统来说至关重要，其主要体现在编码和推送两方面。本系统通过使用多线程、环形缓存和时间同步等技术，使前端模块和 RTSP 服务器模块的实时性都得到了提升，以达到减少传输时延的目的。

3.1 H.264实时编码

开源编码库中的 H.264 编码程序是基于视频文件进行编码，这种编码方式只能对内存中已有的图像信息进行压缩编码，无法对从相机处实时获取的原始数据进行软编码。基于这种情况，我们改进了 H.264 编码器数据输入部分的程序结构，以适应系统实时性的需求。

通过对 Parse 函数的内部结构进行修改，使其不再将读取文件信息作为获取待编码数据的来源，而是通过直接读取系统内存中的数据流进行编码。相机将采集的图像信息直接存储在共享内存中，H.264 编码器会从共享内存中逐帧读取数据，将内存中的数据进行压缩编码，并在每一帧图像的编码数据前添加相应的关键帧、SPS 和 PPS 等前缀 [7]，使得编码后的数据流具备完整的 H.264 格式。

为使系统在编码后能够直接进行 RTSP 推流，编码器对输出的数据不再保存成 H.264 的文件格式，而是直接将其存储在缓存中。RTSP 服务器从该缓存中读取编码信息后，直接进行下一步的推流操作。这样不仅能减少处理过程的时延，还可以节约内存空间，减少系统产生不必要的输出。

3.2 RTSP实时推流

本系统设计并实现了一个 RTSP 类，用来完成从初始化参数配置到数据实时推送的完整过程。RTSP 推送类主要包括 H.264 数据调用、RTSP 服务器构建和图像推送等操作。通过构建 RTSP 对象，进而建立一个完整的 RTSP 实时推流过程。

RTSP 推流过程依靠多线程处理来实现实时性推送。主线程中构建了一个 RTSP 对象，用于初始化 RTSP 推送类、H.264 编码器函数、推送函数、图像参数以及存储空间。推送函数中定义了一个子线程，用来进行数据的处理和推送。主线程中打开 RTSPServer 函数，完成服务器的构建。然后主线程进入获取图像数据的循环中，

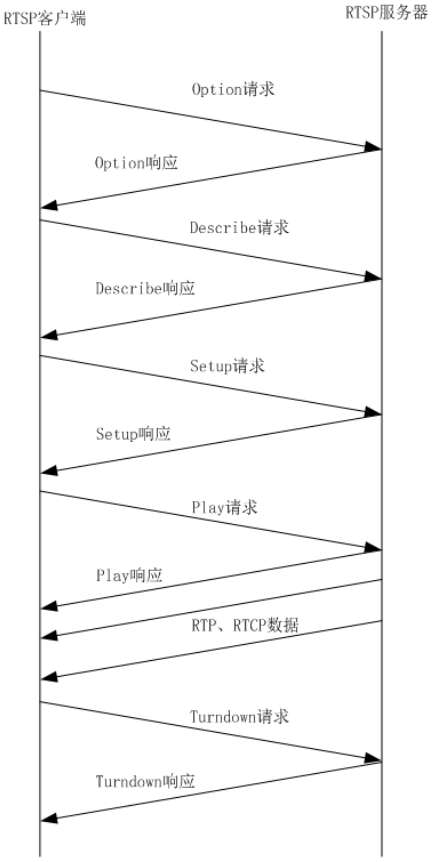

图 2：RTSP 服务器与客户端之间的通信过程

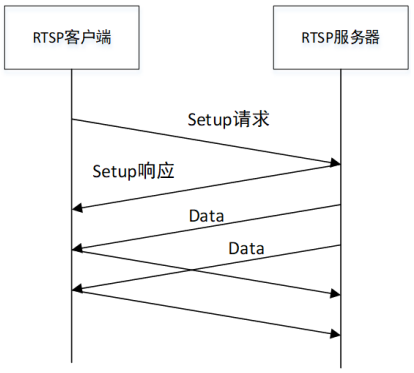

图 3：基于 TCP 协议的 RTSP 数据推流

不断的将获取到的图像数据存放到推送类的指针成员中。最后主线程中加入推送对象的停止和退出操作。这样便完成了主线程中实时推送环境的搭建。

在推送函数中定义的子线程主要负责构造 RTSP 推送的数据。子线程调用主线程中前端相机获取的原始图像数据，将图像数据转换成 YUV420 格式，并进行 H.264 编码。再将编码后的数据通过主线程中的 RTSP 服务器进行推送，使用 TCP 协议传输到客户端。

3.3 图像实时缓存区

在对原始图像数据进行 H.264 编码时，由于前端模块获取图像数据的速度和 H.264 编码器的读入速度并不相同，两者数据直连会造成时延。因此图像数据需要经过缓存区再进入到编码器中。线性缓存区不能同时进行读写两种操作，数据需写满缓冲区后，再进行读出，这会造成在编码器读取数据时，图像数据因等待写入而丢失，

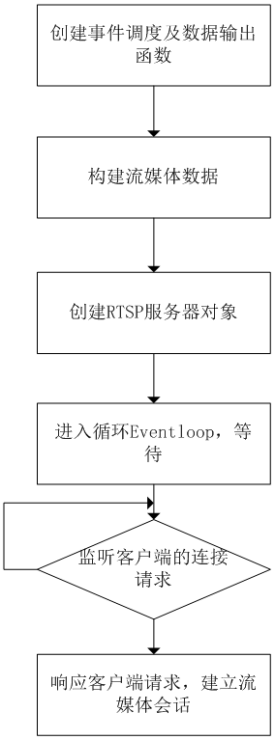

图 4：RTSP 服务器的基本设计流程

无法保证系统的实时性。为此系统在前端输入设备和编码器之间设置一个环形缓冲区。环形缓冲区能够同时进行读写操作，实现图像数据的先入先出，这样便解决了两个线程同时处理的问题。并且缓冲区不需要再拷贝内存，对数据的解析率高。但环形缓冲区的大小也是有限的，如果读写数据的速率相差很大，可能会导致数据溢出，这时便需要分别计算图像获取操作和 H.264 编码操作的耗时，通过主动延迟某一方面的单次耗时，配合系统的环形缓冲区来确保数据不会丢失。

4 系统测试与结果

流媒体实时传输系统使用 Visual Studio 2015 作为开发工具，在 Windows 平台下进行运行。采用高清鱼眼相机作为前端采集与编码模块的视频输入设备，使用 VLC 播放器作为客户端。

运行流媒体实时传输系统后，终端界面会显示出当前流媒体系统的 RTSP 推流地址和如图 5 所示的前端实时图像窗口。使用 VLC 中的网络串流功能，输入实时传输系统的推送地址和预先设定好的端口号，即可完成视频的实时播放。实验结果如图 6 所示，VLC 播放器的显示画面与前端输入图像一致，证明基于 RTSP 服务器的流媒体实时传输系统中的数据采集、编码、推送、播放等各个功能皆能正常运行，系统整体达到实时传输效果。

系统在使用的过程中，支持多个接收端并行观看，使用不同的网络串流客户端，通过访问相同的 RTSP 播放地址，便能同时观看前端流媒体实时数据。

5 结束语

本文讨论了使用 RTSP 技术实现流媒体数据实时传输的全过程，设计并实现了一种基于 RTSP 服务器的流媒体实时传输系统。系统能够对图像数据进行 H.264 实时编码和 RTSP 实时推送。在兼顾传输实时性的同时，系统采用了 TCP 协议对 RTSP 数据进行传输，保证了数据传输的可靠性。因此本系统能够为望远镜的观测图像提供实时、可靠的传输服务。

图 5：前端实时图像窗口

图 6：流媒体实时传输系统客户端测试结果

参考文献

[1] 张洪坤．移动式视频监控系统的设计与实现 [D]．东南大学，2017.
[2] 张宛怡，张尧，周启炜．基于 H.264 标准的实时视频压缩系统设计 [J]．数字通信世界，2019(05)：127-128+140.
[3] 胡李镇．基于 RTSP 的嵌入式流媒体服务器设计与实现 [D]．华中科技大学，2017.
[4] 茅炎菲，黄忠东．基于 RTSP 协议网络监控系统的研究与实现 [J]．计算机工程与设计，2011, 32(07)：2523-2526+2530.
[5] 黄天驰．基于 H.264 与 H.265 的低延时视频监控系统的设计与实现 [D]．贵州大学，2018.
[6] 彭湛博．无人机实时高清图传系统的设计与实现 [D]．西安电子科技大学，2018.
[7] 魏崇毓，张宏琳．基于 Live555 的手机实时直播系统设计与实现 [J]．计算机工程与设计，2016, 37(05)：1156-1160.

作者简介

王超，男，硕士学位，研究实习员。研究方向为图像处理与软件开发。
王昊京，男，博士，副研究员。研究方向为图像处理与软件开发。