Airborne Multi-channel Video Synchronization Playback Technology Based on C/S Mode

Zhao Hao*, Ying An, and GuoWang Zhang
Chinese Flight Test Establishment, Xi’an, Shaanxi, 710089, China
*Corresponding author’s e-mail: haoc036@avic.com

Abstract. In the flight test, the airborne video data is an important part of the flight test data. At present, the airborne recorded video data is an independent M2T file, and the existing video playback software can only play back a single video file. In order to solve the problem of synchronous playback of airborne multi-channel video data, a multi-channel video synchronous playback technology based on C/S mode is proposed. The server performs video file reading control, and the client performs video decoding and display. Firstly, the basic concept of FFmpeg is introduced. Then the overall design of the system is discussed. The design and implementation of the server and the client are emphasized. The system is designed with C++ language, and the TCP/IP protocol is used in the server to interact with the client. Software design uses modularity and multi-threaded to improve processing efficiency and ensure functional scalability. At present, the software has been successfully applied to the simultaneous playback of multiple types of missions and multiple channels of video. The actual application results show that the system is stable and can effectively improve the efficiency of model flight tests.

1. Introduction

Flight test is an important means for verifying the design index requirements of aviation products, inspecting the quality of aviation products and conducting research on new aviation theories and technologies. In the flight test, the video data can fully, intuitively, time-effectively and objectively reflect the human-computer interaction, which is an important basis for the flight test engineer to give the aviation weapon equipment appraisal/final flight test conclusion[1]. At present, each channel of the airborne video data is recorded as an independent M2T file in the form of TS stream[2][3]. After the flight, the video data is downloaded to the disk array for storage through the ground offload device. Researchers download the video data they care about and then use existing video playback software like Baofengyingyin to play it. However, the existing video playback software can only realize single-channel video data playback. In some key flight test subjects, researchers need to perform synchronous comparative analysis of multiple video images. If multiple video softwares are used to replay each channel of video separately, accurate positioning is impossible and the process is cumbersome.

Based on the above analysis, a C/S mode[4][5] multi-channel video data synchronous playback scheme is proposed, which reduces the link of downloading video data to the local. The server side reads the multi-channel video data frame by frame, and then reassembles the video data and sends it to the client using TCP/IP protocol[6][7][8]. After the client receives the video data, it unpacks and sends each channel of video data to the decoder for decoding. The decoded YUV[9] data is displayed through SDL[10][11], thereby realizing multiple channels of video data Synchronized playback.
2. FFmpeg

FFmpeg[12][13] is an open source, cross-platform audio and video streaming solution. It is the most complete set of multimedia support libraries in free software. It has high portability and codec quality. It provides a complete solution for audio and video conversion, decoding and streaming. FFmpeg includes 8 types of library files: libavcodec, libavdevice, libavformat, libavutil, libpostproc, libswresample, and libswscale. Libavcodec contains all FFmpeg audio/video codec libraries, and related data structures include AVFormatContext, AVCodecContext, AVCodec, AVPacket, AVFrame and AVPicture, etc. Libavformat implements streaming media protocols (udp, rtp, rtmp, rtsp, etc.), media containers (mp4, AVI, FLV, etc.) and basic I/O access. Libavutil is a utility library that contains secure mobile string functions, random number generators, data structures, additional mathematical functions, encryption and multimedia-related functions. Libavfilter is a general audio and video post-processing library. Libavdevice provides a general framework for everything from acquisition and rendering to common multimedia input/output devices, and supports multiple input and output devices. Libswresample implements audio resampling and mixing. Libswscale implements color format conversion and scaling.

3. Overall design

The system adopts the C/S design mode, and the server uses TCP/IP protocol to exchange data with the client to ensure the reliability of connection and data transmission. The server is listening on port 2333. The client sends a request to establish a connection to realize data exchange. The system function structure chart is shown as in Figure 1. The server includes a command parsing module, a file reading module, a package module and a data sending module. The command parsing module mainly realizes the functions of receiving and parsing client commands, and completing the pause, continue and progress jump functions. The file reading module uses FFmpeg to read the video file frame by frame. The grouping module realizes regrouping each channel of video data so that the client can perform demultiplexing and decoding. The client is divided into an unpacking module, a video decoding module, an image display module and a human-computer interaction module. The unpacking module realizes receiving and parsing server data to obtain a frame of video data and send it to the corresponding decoder. The video decoding module uses FFmpeg to decode a frame of video data to obtain YUV data. The image display module uses SDL to display YUV image data. The human-computer interaction module mainly implements response interface operations and sends instructions such as pause, continue, and progress jump to the server.

![Multi-channel video synchronization playback system](image)

**Figure 1. System function structure diagram**

The server running process is as follows:
1) Network initialization, monitoring client connection requests.
2) When a client sends a request, add the client's information to the client list, and open a separate thread for the client to perform data exchange;
3) When receiving the file information sent by the client, first initialize FFMPEG, then open the corresponding files, and send the video configuration information to the client;

4) Read the video data frame by frame cyclically, group it and send it to the corresponding SOCKET;

5) When a client disconnects, remove the client information from the client list and close the corresponding thread.

The client operation process is as follows:
1) Network initialization, initialize SDL;
2) Send a request to the server to establish a data connection;
3) Send the aircraft number, date, flight number, etc. to be replayed to the server;
4) Receive server data and unpack. If it is video configuration information, initialize each channel of FFMPEG, otherwise get a frame of data and current time;

5) Send a frame of video image to FFMPEG for decoding to obtain YUV data;

6) Use SDL to display YUV image data.

The operation flowchart of the server and client system is as follows.

![System operation flow chart](image)

Figure 2. System operation flow chart

### 4. Software Implementation

#### 4.1. Server design

In response to the requirement of synchronous playback of multiple channels of video, a structure is defined to represent the relevant information of the client. The member variables of the client structure are as Table 1:
Table 1. The member variables of the client structure

| Variable                  | Description                                                      |
|---------------------------|------------------------------------------------------------------|
| SOCKET s;                 | //Data interaction socket                                        |
| char cmd[10];             | //Command                                                        |
| HANDLE hThread;           | //Thread handle                                                  |
| CString strFile[4];       | //Video data path, up to four                                    |
| CFFMPEG [4];              | //Decoder corresponding to each channel                         |

The TCP initialization process is: use WSAStartup to initialize Winsock; create a socket; use bind to bind the socket; call listen to monitor. The non-blocking asynchronous socket WSAAsyncSelect() is used to realize network communication, and the message-based asynchronous access strategy is adopted for network events, which can conveniently handle network communication. When data is received, FD_READ message will be triggered. When there is a client request, the FD_ACCEPT message will be triggered, and a unique SOCKET will be generated for data transmission. When the client is disconnected, it will trigger the FD_CLOSE message, traverse the client list to find, if the SOCKET corresponding to the current client is found, it will be removed from the client list.

The server and client data exchange protocol includes video file configuration and video data sending. The video file configuration information sending protocol is defined as follows: [synchronization word] [number of channels] [total seconds] [image width] [image height] [encoding format]. The sync word is "FEGH1234", and the other fields are 2 bytes, totalling 18 bytes. The video data sending protocol is defined as follows: [number of channels] [current seconds] [video data].

Multi-channel video synchronization playback adopts two strategies: frame synchronization and time stamp synchronization. Frame synchronization is to read one frame of data of multiple channels in sequence each time and send it to the client for decoding and display. The time stamp synchronization strategy is based on the first video time stamp, and the other channels use av_seek_frame to realize the video frame positioning of the corresponding time stamp.

The total time calculation method is: pFormatCtx->duration/AV_TIME_BASE. The current time is based on the first video. The calculation method is: the DTS of the current video packet minus the start time of the video file, and then the number of seconds of the current video frame is obtained after conversion. The formula is: $T_n = (\text{packet->dts} - \text{pFormatCtx->streams[videoIndex]->time_base.num}) / \text{pFormatCtx->streams[videoIndex]->time_base.den}$. The av_seek_frame is used for frame jump, when the progress jump received from the client is $T_d$, the calculation formula is $T_r = \text{pFormatCtx->streams[videoIndex]->start_time} + \text{av_rescale}(T_d, \text{pFormatCtx->streams[videoIndex]->time_base.den}, \text{pFormatCtx->streams[videoIndex]->time_base.num})$.

In order to make full use of system resources, the server uses multi-threaded concurrency technology. The main thread exchanges instructions with the client, and the independent threads of each client realize the reading and sending of multiple channels of video data. The main thread controls the suspension and resumption of the child threads. The server completes the analysis after receiving the client instruction. When a client sends a progress jump instruction, av_seek_frame is used to move the file pointers of each path to the video frame at the corresponding time point. When a client issues a pause or resume instruction, SuspendThread or ResumeThread is used to suspend or resume the corresponding thread.

4.2. Client design

The client TCP initialization process is: use WSAStartup to initialize Winsock; create a socket; use connect to connect. The commands for interaction between the client and the server include: (1) The video file information is the string "FILE# aircraft# date# flight number"; (2) Pause is the "Stop" command; (3) The continue is the "GoOn" command; (4) The progress jump is the string "MOVE# time".
In order to avoid the time-consuming video decoding and display resulting in loss of video data packet reception, the client uses multi-threaded concurrent technology. The client uses the user UI interface operation as the main thread, data reception, video decoding and image display as sub-threads, and the main thread performs operations such as pause and dragging the progress bar. After the data receiving sub-thread receives the data, the data packet is stored in the FIFO queue. The video decoding thread extracts a frame of video from the queue for decoding, and sends a message to the image display sub-thread after the decoding is completed.

The process of FFMPEG decoding video files is as follows:
1. Initialize FFMPEG:
   \[\text{av_register_all()}\]
2. Open the corresponding file:
   \[\text{avformat_open_input(&pFormatCtx, filepath, NULL, NULL);}\]
3. Find audio and video stream information:
   \[\text{avformat_find_stream_info(pFormatCtx, NULL);}\]
4. Find the video decoder of the corresponding format:
   \[\text{pCodec = avcodec_find_decoder(pCodecCtx->codec_id);}\]
5. Read one frame of data:
   \[\text{av_read_frame(pFormatCtx, packet);}\]
6. Open the video decoder of the corresponding format:
   \[\text{avcodec_open2(pCodecCtx, pCodec, NULL);}\]
7. Use the corresponding decoder to decode a frame of video data:
   \[\text{avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);}\]

On the server side, after receiving the video file information sent by the client, call (1)-(4) to complete the reading of the video data file. After the initialization is completed, call (5) to read one frame of image in turn. Perform (1) on the client to complete the initialization, then receive the video file configuration information sent by the server, initialize the pCodecCtx required for decoding, and then call (6) to open the decoder. After receiving a frame of video data cyclically, call (7) to realize video decoding.

Simple DirectMedia Layer(SDL) is a cross-platform multimedia development library based on C language, which provides a variety of functions for controlling audio and video input and output. SDL shows the YUV data flow as follows:
1. Initialize SDL:
   \[\text{SDL_Init(SDL_INIT_VIDEO | SDL_INIT_TIMER);}\]
2. Create SDL display window:
   \[\text{screen = SDL_CreateWindowForm(m_hwnd);}\]
3. Create a renderer based on the window:
   \[\text{sdlRenderer = SDL_CreateRenderer(screen, -1, 0);}\]
4. Create texture to display YUV data:
   \[\text{sdlTexture = SDL_CreateTexture(sdlRenderer, SDL_PIXELFORMAT_IYUV, 1, width, height);}\]
5. Set the pixel data of the texture:
   \[\text{SDL_UpdateTexture(sdlTexture, NULL, pFrameYUV->data[0], pFrameYUV->linesize[0]);}\]
6. Copy the texture data to the rendering target:
   \[\text{SDL_RenderCopy(sdlRenderer, sdlTexture, NULL, NULL);}\]
7. Display screen:
   \[\text{SDL_RenderPresent(sdlRenderer);}\]

After receiving the video configuration information sent by the server, the client executes (1)-(4) to realize SDL initial display settings, and calls (5)-(7) to display the YUV image.
5. Application effect
The actual application effect is shown in Figure 3. Select the aircraft, date and flight number, click the start button to establish a connection with the server and receive server video data. Click the pause/continue button and drag the progress bar to send instructions to the server to control the playback progress. The application effect shows that the system can realize the synchronous playback of the airborne multi-channel video.

![Figure 3. Actual application effect](image)

6. Conclusion
In order to solve the problem of synchronous playback of multi-channel video data on the test plane, a multi-channel video synchronous playback system based on C/S mode was designed. The server and the client use TCP/IP protocol for data exchange. The server uses FFmpeg for multi-channel video file reading and packet transmission. The client receives a frame of video data from the server, uses FFmpeg to decode and displays it through SDL. This technology has been applied to the simultaneous playback of multi-channel video of flight test of multiple models to realize the functions of pause, continue and control the progress. The correctness and reliability have been verified to meet the requirements of multi-channel video synchronization playback to ensure the efficient flight test of the model. Smooth progress plays an important role.

Acknowledgments
Authors wishing to acknowledge assistance or encouragement from colleagues, this work is supported by the National Defense Science and Technology Industry Administration National Defense Basic Research Program Key Project under grants JCKY2016205B006.

References
[1] Lichter M, Bateman A, Balas G. (2009) Flight Test Evaluation of a Run-Time Stability Margin Estimation Tool. In: AIAA Guidance, Navigation, and Control Conference.
[2] Pereira C M M, Mello R F D. (2014) TS-stream: clustering time series on data streams. Journal of Intelligent Information Systems, 42(3):531-566.
[3] Wang H, Zhang Y. (2009) The Digital TV User Behavior Information Gathering System Based on the TS Stream of MPEG-2 System. In: International Conference on Management & Service Science. IEEE.

[4] M X Zang, Q Z Chen, JG Jiang. (2010) Network Secure Transmission Based on Authentication for C/S Mode. In: International Forum on Information Technology & Applications. IEEE.

[5] Zhang J P, Zhu X D, Liang X. (2009) C/S and B/S Mixed Style and the Application. In: Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on. IEEE.

[6] B J Youm, J Park. (2005) TCP/IP PROTOCOL OVER IEEE-1394 NETWORK FOR REAL-TIME CONTROL APPLICATIONS. In: Prague:16th Triennial World Congress. pp.37-42.

[7] Lakshman T V, Madhow U. (1997) The performance of TCP/IP for networks with high bandwidth-delay products and random loss. IEEE/ACM Transactions on Networking, 5(3):336-350.

[8] Balakrishnan H, Seshan S, Amir E, RH Katz. (1995) Improving TCP/IP performance over wireless networks.

[9] J Mukherjee, M K. Lang, S K Mitra. (2005) Demosaicing of images obtained from single-chip imaging sensors in YUV color space. Pattern Recognition Letters, 26(7): 985-997.

[10] Wang J J, Wang Z M. (2013) H.264 streaming media player system based on SDL. Computer Systems Applications, 22(12): 51-54.

[11] Li K, Li L, Lan S Y. (2014) Multiple video streams real-time conversion and playback based on FFmpeg and SDL. Computer Technology and Development, 24(4): 65-68.

[12] Qin Y. (2011) The research of the video format conversion technology based on the FFmpeg. Computer Knowledge and Technology, 7(12): 2912-2913.

[13] Zeng H, Fang Y. (2013) Implementation of video transcoding client based on FFmpeg. Advanced Materials Research, 756-759: 1748-1752.

[14] Courbin P, Lupu I, Goossens J. (2013) Scheduling of hard real-time multi-phase multi-thread(MPMT) periodic tasks. Real-Time Systems, 49(2):239-266.