Special Treatment of Video Image Based on FFmpeg

Gaohe Li

School of Economic Management

Xi'an Shiyou University, Xi'an, China

gaoheli@xsyu.edu.cn

Key words: FFmpeg; VC++; graphics and image processing; eliminate objects in video; mask objects in video; replace the background sound in video

Abstract: FFmpeg is an open source computer program that can be used to record and convert digital audio/video, and convert it into a stream. It contains very advanced audio/video codec library libavcodec, and in order to ensure high portability and decoding quality, a lot of code in libavcodec is developed from scratch. In this paper, it is realized eliminating objects in video, masking objects in video and replacing the background sound in video by using FFmpeg, as well as in the video image to add company logo, to add live broadcast time, to increase watermark , and so on. It has high practical value.

1 Introduction

In video processing, you sometimes need to do some special processing. For example, masking objects in video. That is also to make it disappear, and the rest of the video remains the same in normal. This special requirement is mainly applied to the video that the leaders inspected military bases, such as requiring the screening of sensitive equipment or background; Or it can be used in film post-production, for example, to screen out items that should not appear on the screen. After weeks of hard work, the target was finally fulfilled. In addition, the project also completed eliminating objects in video, and replacing the background sound in video, adding the company logo in video, adding real-time playing time and other functions. The implementation of the project is described in detail below.

2 FFmpeg

FFmpeg is an open source computer program that can be used to record and convert digital audio/video. It includes very advanced sound/video encoding library libavcodec and so on. FFmpeg is developed under the Linux platform, but it can also be compiled in other operating system environments, including Windows, Mac OS, and so on[1]. You need to install MinGW under Windows. Specific installation process can reference the author website address：

http://www.31kj.net/templets/company/techonology/ index_tech.htm.

FFmpeg structure system is more complex. In short, FFmpeg contains the following main contents:[2]

libavcodec: various types of sound/image encoding and decoding;

libavutil: some common utility functions;

libswscale: scaling and color mapping of video scenarios;

libpostproc: post-effect processing;

ffmpeg: a tool provided by the project that can be used for format conversion, decoding, or TV card real-time coding, etc.

ffsever: an HTTP multimedia real-time broadcast streaming server;

ffplay: a simple player that uses ffmpeg library to parse and decode，SDL display;

The Decoding Process about FFmpeg video player is shown in figure 1.

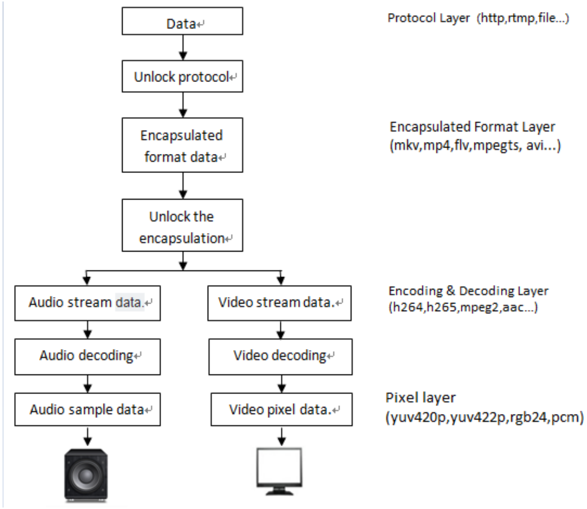
Data
Unlock protocol
Encapsulated format data
Unlock the encapsulation
Audio stream data.
Video stream data.
Audio decoding
Video decoding
Audio sample data
Video pixel data.
Protocol Layer (http,rtmp,file...)
Encapsulated Format Layer (mkv,mp4,flv,mpegts, avi...)
Encoding & Decoding Layer (h264,h265,mpeg2,aac...)
Pixel layer (yuv420p,yuv422p,rgb24,pcm)

fig.1 The Decoding Process about FFmpeg Video Player[2]

In the following, the process is introduced to realized eliminating objects in video, masking objects in video and replacing the background sound in video by using FFmpeg, as well as in the video image to add company logo, to add live broadcast time, to increase watermark , and so on.

3 Eliminate Objects in the Video Image

To achieve the elimination effect to video object, we should first understand the principle of Opencv video playing. OpenCV(open source computer vision library) is an open source computer vision and machine learning library. OpenCV provides a common infrastructure for computer vision applications and accelerates the use of machine awareness in commercial products. OpenCV is a BSD-licensed product that makes it easy for companies to use and modify code.[3] As we all know, the continuous video screen is actually a continuous stream of pictures, and video is playing at 25 frames per second. This can be verified by the following procedure:

```
CvCapture * capture;                                   //video source object.
IplImage *frame = NULL;                                // video's current frame.
char url[]="rtmp://live.hkstv.hk.lxdns.com/live/hks";  // Hong Kong TV video source test, or file name.
capture = cvCaptureFromAVI(url);
……
if(frame_no>370 & frame_no <432)                       //select the range of frame images you care about.
{
      frame = cvQueryFrame(capture);                   // get the current frame.
      sprintf(tmpfile,"%s\\%d.bmp",PicSavePath, frame_no);
      cvSaveImage(tmpfile, frame);
```

```
        ++ frame_no;                                    //The frame no.
    }
```

If we run the above program , we will get all of video's image files. The process of eliminate objects in video is actually to extract a single image in the video files, and then use the drawing tools to modify and save images, finally according to the original order to pick up these images, and generate the modified video. Modify the extracted image files, such as eliminating a person portrait object in the video image, we need to remove the portrait from continuous each image by plotting tools. The workload is very big, because it will produce 25 video images in one second, so it will is produced 1500 video images in one minute, and these image files is larger in space. And we pay attention to a skill here, if the video image background is relatively static, such as the announcer broadcast images. If we want to eliminate the announcer portrait, it is the relatively smaller workload. We can make a rectangle image without the announcer's portrait, and it has only background behind the announcer, and the rectangle can covers the area of the announcer's portrait. While we regain our image file and generate video image, we can replace the area of announcer portrait using the rectangular background, so to reach to eliminate the object target. The effect is shown in figure 2.

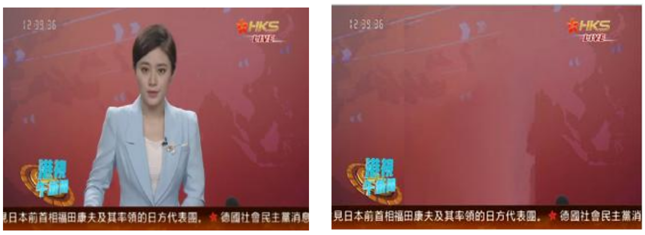

fig.2 Eliminating an Object in the Video Image

4 Mask the Objects in the Video Image

If you know the principle of eliminating objects in the video image above, the process of masking the objects in the video image is relatively simple. That is, the area is replaced by using random data instead of the background image, as shown in figure 3.

To facilitate the completion of the task, you may need to get video current (or total) playing time, as well as the current (total) frames .The following statement will be useful:

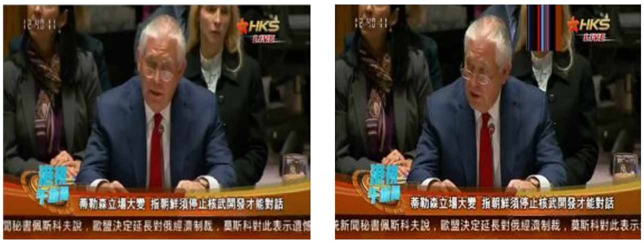

fig.3 Mask an Object in the Video Image

```
double fps=cvGetCaptureProperty(capture, CV_CAP_PROP_FPS );          // the frame rate of video
int vfps = 1000 / fps;                   //calculate the time per frame, microsecond/frame,1000 ms =1 second
double frames=cvGetCaptureProperty(capture,CV_CAP_PROP_FRAME_COUNT);
                                                                    //the total number of frames in video
```

```
……
char *filename2="TestData/map2.bmp";
if(count_tmp>860 & count_tmp<1140)
{
      cvSaveImage(filename1,frame);
      // the rectangular scope of the graphic file is filled with random data
      pixHandle2(filename1,300,230,355,288);
      image2 = cvLoadImage(filename2);
      Mat M(image2);
      cv::cvtColor(M, dstYuvImage, CV_BGR2YUV_I420);
}
```

5 Replac Background Audio in Video

Replacing the background audio we're talking about now is to use the original sound, which means we're going to insert our background audio data directly into the original video file. The best example about this is the soundtrack to a film. Of course, this process is very complicated. Here we can only introduce the principle and process. The process of replacing background audio in video is to stop the original background audio stream from the position which we hope, and insert our own background audio stream in the file instead, or to replace all original video audio by our own audio. In the latter case, In simple terms, the principle is to define a video input, such as the video of MP4, and define an audio input, such as an MP3 audio. The two are then synthesized to form a new MP4 file. So you get rid of the MP4 voice, and you switch to the MP3 voice. The technical key here is that when you write video or audio to a file, you must pay attention to the timing of both.[4][5]

```
int compare_tag =
av_compare_ts(cur_pts_v,ifmt_ctx_v->streams[videoindex_v]->time_base, cur_pts_a,
ifmt_ctx_a->streams[audioindex_a]->time_base) ;
if(compare_tag<= 0)
{
     ifmt_ctx=ifmt_ctx_v;
     stream_index=videoindex_out;
     if(av_read_frame(ifmt_ctx, &pkt) >= 0)
     {
     ……      // Process video data.
     }
}
else{
     ifmt_ctx=ifmt_ctx_a;
     stream_index=audioindex_out;
     if(av_read_frame(ifmt_ctx, &pkt) >= 0)
     {
     ……      // Processing audio data
```

```
        }
    }
```

6 Add Watermark

Video adds watermark, such as adding the company Logo in video. The principle is to make a transparent Logo picture first. The image is then added to the video stream in advance of video recording.[6]

```
AVFrame *pFrame, *pFrameYUV;
AVFilterContext *buffersink_ctx;
const char *filter_descr= "movie=res/logo.png[wm];[in][wm]overlay=5:5[out]";
……
// Initialize the filter to prepare watermark.
if ((ret = init_filters(filter_descr)) < 0)    // Init_filters is a custom function, slightly.
return -1;
……
// Decoding the video[7]
if (avcodec_decode_video2(dlg->pCodecCtx_Video, pFrame, &got_picture, packet) < 0)
{
        printf("Decode Error. \n");
        continue;
}
if (got_picture)
{
pFrame->pts = av_frame_get_best_effort_timestamp(pFrame);
if (av_buffersrc_add_frame(dlg->buffersrc_ctx, pFrame) < 0) {
      printf( "Error while feeding the filtergraph\n");
      break;
}
ret = av_buffersink_get_frame(dlg->buffersink_ctx, pFrameYUV);
if (ret < 0)
      continue;
FILE *fp_yuv=fopen("TestData/output.yuv","wb+");
……
fseek(fp_yuv,0,SEEK_END);
fwrite(dst_data[0],   y_size, 1, fp_yuv);
fwrite(dst_data[1],   y_size/4, 1, fp_yuv);
fwrite(dst_data[2],   y_size/4, 1, fp_yuv);
fflush(fp_yuv);
```

The effect of the video after adding watermark is shown in figure 4.

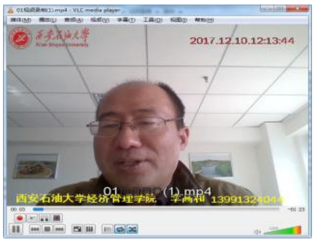

fig.4 Increases Watermark Logo and Time, Chinese character in the Video Image.

7 conclusion

To eliminate objects, masking objects, replacing the background audio, add watermark and so on are very multifarious work, understanding its working principle is only a first step, we need also a lot of knowledge of FFmpeg, even OpenCV knowledge, as well as the C++ programming knowledge and experience. This article does not describe add Chinese characters work in video, the more complex, the author wrote a paper specifically for it, which will be published in the Journal of Xi 'an Shiyou University (natural science edition). The readers who are interested in it can refer to that Journal. As for adding system time in video, this feature is provided in OpenCV, which is relatively simple, and the author will not repeat it. For the actual presentation of all the content of the paper, readers can visit http://www.31kj.net.The website also includes FFmpeg, OpenCV and other graphic image knowledge introduction.

References

[1] ffmpeg. Baidu encyclopedia [EB/OL]. https://baike.baidu.com/item/ffmpeg/2665727? fr=aladdin, 2018/2018-4-15

[2] Lei Xiao hua. Introduction to video /audio data processing: RGB, YUV pixel data processing. [EB/OL]. https://blog.csdn.net/leixiaohua1020/article/details/50534150, 2016-1-29/2018-4-15

[3] About. OpenCV team[EB/OL]. https://opencv.org/about.html, 2018/2018-3-25

[4] Lei Xiao hua. Introduction to video /audio data processing: PCM audio sampling data processing. [EB/OL]. https://blog.csdn.net/leixiaohua1020/article/details/50534316, 2016-1-29/2018-4-15

[5] Lei Xiao hua. Introduction to video /audio data processing: AAC audio stream parsing. [EB/OL]. https://blog.csdn.net/leixiaohua1020/article/details/50535042, 2016-1-30/ 2018-4-15

[6] Rafael C. Gonzalez, Rchard E. Woods. Digital image processing (third edition) [M]. Beijing: Electronic industry press, 2011.6

[7] Lei Xiao hua. Introduction to video /audio data processing: H.264 video code flow resolution. [EB/OL]. https://blog.csdn.net/leixiaohua1020/article/details/50534369, 2016-1-31/2018-4-15