An Available and Simple Android-Based Short Video Processing Method

Junlin Lu

East China Normal University, Shanghai 200062, China

lujunlin6@gmail.com

Keywords: Video processing, Android, Short video, Opengl.

Abstract. This paper introduces an available method, which displays the short video production processing on Android. Including three such steps as raw video recording, video editing, and video exporting. The general method those short-video-processing apps like Meipai, Duopai and Miaopai use need NDK to develop their own recorder and player. Which is complicated and confusing. However, the method this paper introduces is much simpler and effective as well.

1. Introduction

Many micro-video apps have appeared on Android in China since the Vine [1] became popular. Chinese micro-video apps like Maipai [2], Miaopai [3] and Duopai [4] have attracted many young people to upload their mv. All the apps above have three core steps to make video: recording, editing, and exporting. After a study of those three Chinese apps, a general method can be discovered. They create their own recorder and player with android-NDK [5], in order to record several square videos whose resolution and size are both 480*480, then concatenate them into one video, before previewing it by their own player, then editing it by adding background video and audio with some rendering like screen blend or source over [6], and a logo ending with Gaussian-blur [7]. NDK is powerful but coded by C/C++.The work is hard. However, the method this paper introduces is different from that general one. This method uses MediaRecorder [8], the Android primary class, to record raw videos whose resolution and size are both 640*480. Then edit them without concatenating, crop every frame grabbed [9] from the raw videos to 480*480 during editing step [10], then output one final mv in exporting step. Compared with the previous one, this method is easier, quicker, and effective as well.

2. Preparation

2.1 MediaRecorder.

I just use MediaRecorder to capture raw camera data and raw audio data for the convenience of recording, this idea make the recording work much easier than create a new one, without worrying about the video quality and the synchronizing issues between images and audios. The detail use of this class is in [8].Just follow the document to configure those parameters in a specific sequence [11].

2.2 FFmeg.

FFmpeg [12] is a complete, cross-platform solution to record, convert and stream audio and video. It is a good choice for mobile app video processing. There are already good wrapper for IOS, such as iFrameExtractor [13], FFmpeg-iOS-Encoder [14]. However for Android, the tools is neither so handy nor updated in time. Some tool just wrap a simple graphic interface for user to input command line. Some's last commit is long time ago. Which causes a big problem for developers who want to develop an app which has a small function to process videos.

2.3 JavaCV

As in [9], JavaCV is an open-source github project. It uses wrappers from the JavaCPP Presets [15] of commonly used libraries by researchers in the field of computer vision (OpenCV [16], FFmpeg [12], libdc1394 [17], PGR FlyCapture [18], OpenKinect [19], videoInput [20], ARToolKitPlus [21], and flandmark [22]). It provides utility classes to make their functionality easier to use on the Java platform, including Android. It also provides sample usage [23] for you to learn. JavaCV is used to

grab out frames from raw videos for editing and is used to record the processed frames into one square video.

JavaCV has useful tools for video processing, but not enough. It just wraps limited tools. Until recently, the author add a filter called FFmpegFrameFilter, I can only choose opengl to cut and flip the image.

2.4 OpenGl ES 2.0

As in [10], OpenGl ES 2.0 is supported by Android 2.2 and higher, which is a stable version. When editing video, like rendering effect, adding background video, adding logo emerging gradually with Gaussian-blur background. The result is previewed on screen in real time. OpenGl ES 2.0 is needed to do this. More details are in [24],[25],[26].

3. Implementation

This paper divide micro video processing into three steps: recording, editing, and exporting. Recording is capturing raw videos via camera and microphone. Editing is making video by choosing some renders or some filters or some music to make video beautiful. Exporting is outputting the final production.

3.1 Recording

Although MediaRecorder make things simple, some issues have to be solved. First, micro video is known as a 480*480 square video. But the resolutions Android supports are all rectangles like 640*480,720*480 [27]. Second, because of the square video, the recording view must look like a square view. In fact the camera still record data as a rectangle, so we still need to crop the raw video. Third, the recording view is upright when user hold his phone to record something, but most video players on Android plays the raw video MediaRecorder records in 90 counterclockwise rotation for the widescreen experience. In a word, there are three issues needing to solve:

Crop image frame into square, that is, from 640*480 to 480*480.

Adjust the recording view to make it look like a square but is actually a widescreen view to hold data, in this way the image in the raw video doesn’t look like a stretch.

Rotate the image to make it playing normally.

3.2 Editting

Other than concatenating several raw videos into one video, which are directly used for editing. JavaCV is used to grab image frames and sample frames from raw videos for OpenGl to render with effect and background video, displaying on GlSurfaceView at the same time the sound is played synchronously. There are also several issues:

Sample frame grabbed from the raw videos or the background music should be played when the OpenGl rendering. No matter how many channels the sample frame has, it is necessary for AudioTrack to convert into one channel.

When the user click one option to preview a specific rendering effect. The effect should be switched instantly and the previous video and sound should stop and release related resources.

The preview process can be paused when the user click on the screen to see one static image. When the user click again, the GlSurfac View keeps rendering with the audio playing.

Several vertice shaders and fragment shaders should be written for opengl es to use, providing the user different options. These shaders can be learned from GPUImage Android [6].

The video has a logo ending with the bluring video image as background. Gaussian blur is used in shaders but optimized for GPU processig. One entire gaussian blur on one image is rendered twice, one do x offset, another do y offset. So framebuffer in opengl is needed to do off-srcreen rendering.

3.3 Exporting

After user has chosen what effect and music (or not at all) he want, click the “next” button to Exporting time. In this time, the final 480*480, 30fps video must be exported. Remember that raw videos are not be concatenated into one video. These things should be done now:

Read every pixels with opengles2.0, recording them as the image frame.

Record every sample frame into the video. Make sure the timestamp correct. Background music should be added to original audio if there is it.

Rotate the image from the raw video and the image from background video to make the final video image diplays upright. The ending logo generated in advance and the gaussian-blur effect should also be recorded into the final video.

4. Experiment

Android Studio with gradle 2.3 is used on Mac system for the app development. All the apps are tested on XiaoMi HM Note 1S.The app is compared with Meipai and Miaopai. Duopai has some protection mechanism, that is, the videos are deleted when the user switch to home interface. So I collect data about Meipai and Miaopai. Three apps, including my app, record two raw videos then add a background video to render, and then export into a final video.

4.1 Meipai

MeituPic's short video community. Having the version of Android and IOS. More than 100 million users worldwide are using Meipai. I use Meipai record two short videos.

As we can see in Table 1, videos captured by Meipai don't reach 30 fps, because H.264 is the encoder that allow recorder to change fps according to lighting condition considering exposure. My app also has this problem, but in the same condition, mine can get to nearly 30 fps (29.86). In table 2, we can see the fps go down to 20.73. And in table 3, with background video added, the fps of the final video go back to 25.2. Rapid play can be felt when watch the video. The reason that the audio stream is not seen in table 2 is that background audio is added from another mp4 file. Background video can be cut to adapt to the length of the captured video, whose size does not matter. The total length of the background video is 30 second, long enough for any video whose maximum length is 30 s and minimum length is 10 s.

Table 1 Raw Videos

Parameters	Video Details	
	Raw video1	Raw video2
Video Encorder	H.264	H.264
Video Resolusion	480*480	480*480
Preview Size	480*480	480*480
Audio Encoder	AAC	AAC
Audio Channel	2	2
Sample Rate	44100Hz	44100Hz
Frame Rate(fps)	21.45	20.52
Video Size	762KB	1MB
Data Rate	2.24Mb/s	2.16Mb/s

Table 2 Concatenated Video And Background Video

Parameters	Video Details	
	Concatenated video	Background Video
Video Encorder	H.264	H.264
Video Resolusion	480*480	480*480
Preview Size	480*480	480*480
Audio Encoder	AAC	
Audio Channel	2	
Sample Rate	44100Hz	
Frame Rate(fps)	20.73	30
Video Size	1.8MB	
Data Rate	2.18Mb/s	

Table 3 Final Video

Parameters	Video Details
	Final Video
Video Encorder	H.264
Video Resolusion	480*480
Preview Size	480*480
Audio Encoder	AAC
Audio Channel	2
Sample Rate	44100Hz
Frame Rate(fps)	25.2
Video Size	1.5MB
Data Rate	1.75Mb/s

4.2 My App

As described above, I use a new method, a combination of JavaCV, OpenGl ES 2.0, and MediaRecorder and so on, to do short video processing. All the experiments is conducted on Xiaomi HM NOTE 1S with Android 4.4. The uniform operation is: capture two raw videos, then output the final video.

From the beginning of recording step. My app is different from Meipai. The Resolutions of the raw videos are 640*480, which lead to another road to Rome. However, the fps reach nearly 30 fps. The final video's fps is 29.97, but its data rate is 583.67 kb/s.

Table 4 Raw Videos

Parameters	Video Details	
	Raw video1	Raw video 2
Video Encorder	H.264	H.264
Video Resolusion	640*480	640*480
Preview Size	640*480	640*480
Audio Encoder	AAC	AAC
Audio Channel	2	2
Sample Rate	44100 Hz	44100 Hz
Fps	29.86	29.86
Video Size	380 KB	400 KB
Data Rate	2.47 Mb/s	2.28 Mb/s

Table 5 Background Video and Final Video

Parameters	Video Details	
	Background Video	Final Video
Video Encorder	H.264	H.264
Video Resolusion	480*480	480*480
Preview Size	480*480	480*480
Audio Encoder		AAC
Audio Channel		2
Sample Rate		44100Hz
Fps	30	29.97
Video Size		2.2 MB
Data Rate		583.67 kb/s

There are some other features of my app:

Configurable

Json is used as script to configure how to render a video by configuring renderer,steps where you can specify some video as effect or mask of the effect, beginning time and end time of the background video,cover image and ending logo.

Less Code

By the help of third party project. The code is less much, convinient for maintenance.

5. Conclusion

It is an available method to process short video with no need to implement your own recorder or player. Besides, the quality of the video is fine as well. However, the video frame rate may vary as the light condition varies, though it has good performance on the test phone. Black frames can be produced if the light is dim. JavaCV has added the newest wrapper of avfilter of ffmpeg which can drop the invalid frame. The app will do that in further.

References

[1] Information on: https://vine.co/
[2] Information on: http://www.meipai.com/
[3] Information on: http://www.miaopai.com/
[4] Information on: http://www.duopai.me/
[5] Information on: https://developer.android.com/tools/sdk/ndk/index.html
[6] Information on: https://github.com/CyberAgent/android-gpuimage
[7] Gedraite, E.S, Hadad, M. Investigation on the effect of a Gaussian Blur in image filtering and segmentation
[8] http://developer.android.com/reference/android/media/MediaRecorder.html
[9] Samuel Audet, http://bytedeco.org/, 2014
[10] http://developer.android.com/guide/topics/graphics/opengl.html
[11] http://developer.android.com/guide/topics/media/camera.html#capture-video
[12] Lei Xiaohua, Jiang Xiuhua , Wang Caihong, Design and Implementation of a Real-Time Video Stream Analysis System Based on FFMPEG, 2013
[13] https://github.com/lajos/iFrameExtractor
[14] https://github.com/chrisballinger/FFmpeg-iOS-Encoder
[15] https://github.com/bytedeco/javacpp-presets
[16] Culjak, I, Abram, D,Pribanic, T. , Dzapo, H. , Cifrek, M. A brief introduction to OpenCV,2012
[17] http://damien.douxchamps.net/ieee1394/libdc1394/
[18] http://www.ptgrey.com/products/pgrflycapture/
[19] http://openkinect.org/wiki/Main_Page
[20] http://www.muonics.net/school/spring05/videoInput/
[21] http://studierstube.icg.tugraz.at/handheld_ar/artoolkitplus.php
[22] http://cmp.felk.cvut.cz/~uricamic/flandmark/
[23] https://github.com/bytedeco/javacv#sample-usage
[24] Hwanyong Lee, Nakhoon Baek, Implementing OpenGL ES on OpenGL,2009
[25] http://docs.gl/es2/glActiveTexture
[26] https://open.gl/introduction
[27] http://developer.android.com/reference/android/hardware/Camera.Parameters.html#getSupportedVideoSizes()