Lab Report: Face Detection Recorder

Name: Yina Zhu
Instructor: Bilge Soran
Steele Lab
University of Washington
Jun~Nov 2016

Table of Contents

MainActivity

The major coding part for this project is in **app** > **java** > **MainActivity.java.**

MainActivity > onCreate

OnCreate function is the only function in MainActivity, which loads the layout activity_main.xml. There are three buttons on the layout and each is connected with an activity java file in record folder (CameraActivity.java, RecordActivity.java and FaceBlurActivity.java).

CameraActivity

The major coding part for this project is in **app** > **java** > **CameraActivity.java.**

CameraActivity > VIDEO_REQUEST_CODE

```
private final int VIDEO_REQUEST_CODE = 100;
```

This field is a private, and immutable int field, which a self-defined value 100. We use this int field to check whether the video is successfully recorded, and whether the video is the correct version we want.

CameraActivity > onCreate

This method is a default method, which created with CameraActivity.java. It will be automatically called after the app started.
You can ignore the first line of codes, which just a routine call for all onCreate method. The second line is to set up the interface as we designed in **activity_camera.xml**

CameraActivity > captureVideo

This method is called we used click **"Capture Video!" button**, and main function of it is to **capture by Native Camera API**, and **save it back to device**.

- In this project, we use Intent as the major tool to capture the video. The parameter, *MediaStore.ACTION_VIDEO_CAPTURE,* indicates the use of video capture for the intent.

- The second and third lines of codes is to specify the location we are going to store the video. Next, we convert it into Uri format, which can be accepted with Intent.
- The forth line adds the address into Intent, and the fifth line specifies the quality of video for the Intent
- The last line starts the capture video activities, and passes **VIDEO_REQUEST_CODE** for later confirmation

CameraActivity > onActivityResult

This method will be called automatically after **startActivityForResult**

- The **if statement** checks if the video is successfully captured, and if the video is captured as required with the **VIDEO_REQUEST_CODE**.
- In the second part of the codes, first of all, it creates a reference for VideoView in order to display the processed video. Then, it get the video address by calling **getFilePath** method. Next, display the video into VideoView, which users can watch the video through the app
- The third part of the codes, it sets up **media controller**. These three lines of codes are the standard routine in Android Development to set up media controller.
- It starts playing video by call **start** method with **the VideoView object**.

RecordActivity

The major coding part for this project is in **app** > **java** > **RecordActivity.java.** Most of the code is from https://github.com/bytedeco/javacv/blob/master/samples/RecordActivity.java, please check https://github.com/bytedeco/javacv/issues for specific issues. I'm only documenting the additional functions I wrote here:

RecordActivity > convertToBGR

Since the frames recorded here are originally in YUV_NV21 mode, this method is converting the frames to BGR mode. It also calls rotate function because the frame is originally in landscape mode and thus needs to be rotate to portrait mode.

RecordActivity > rotate

Takes in an IplImage and returns a rotated one. Better choice would be using FFmpegFrameFilter here to avoid using IplImage, but the current version of FFmpegFrameFilter has a problem, details here: https://github.com/bytedeco/javacv/issues/509

FaceBlurActivity

The major coding part for this project is in **app** > **java** > **RecordActivity.java.** Most of the code is from https://github.com/bytedeco/javacv/blob/master/samples/RecordActivity.java, please check https://github.com/bytedeco/javacv/issues for specific issues. I'm only documenting the additional functions I wrote here:

FaceBlurActivity > onCreate

This method is a default method. It will be automatically called after the app started. The second line is to set up the interface as we designed in **activity_faceblur.xml** When the button is clicked, Init_Classifier() and readAndBlur() will be called. The text in textView will also be changed to "Done!" after finished.

FaceBlurActivity > readAndBlur

Initialize FFmpegFrameGrabber and FFmpegFrameRecorder. Use grabber to grab every frame of the record, feed the frame into ProcessVideo() function and record the returned frame using recorder.

FaceBlurActivity > rotate

Takes in an IplImage and returns a rotated one. Better choice would be using FFmpegFrameFilter here to avoid using IplImage, but the current version of FFmpegFrameFilter has a problem, details here: https://github.com/bytedeco/javacv/issues/509

FaceBlurActivity > rotateWithFrame

Convert Frame to Iplimage, feed into rotate function and convert the returned Iplimage back to Frame.

FaceBlurActivity > ProcessVideo

Takes in a Frame, convert to Mat, feed the converted Mat into FaceDetector

FaceBlurActivity > Init_Classifier

When called, read the xml file used for CascadeClassifier faceDetector and initialize faceDetector.

FaceBlurActivity > FaceDetector

Takes in a Mat, convert to grayscale and use faceDetector to detect and blur the face.

Issues for fellow students to continue working:

1) RecordActivity.java currently can only record videos with low resolution and frame rates.

   Two possible solution:
   a) Change RECORD_LENGTH to a positive number. For instance, if you change RECORD_LENGTH to 15, then RecordActivity and record a video for 15 seconds at longest. We need videos that are more than 20 minutes but if you try setting RECORD_LENGTH to a number bigger than 20, it will crash an tell you there is no enough space to allocate a native array. This is because if RECORD_LENGTH is positive, RecordActivity is retrieving frames from the camera and store them in an array, and later will record the frames altogether in an array instead of recording the frames while retrieving frames. Here the goal is to figure out how to allocate enough space to store all the frames before saving them.

   b) RECORD_LENGTH is still set to 0, which means RecordActivity is recording the frames while retrieving them. Because it takes time to convert a frame to BGR mode and save it, the frame rate and the resolution is hard to improve. One thing to try is using parallelism such as AsyncTask.

2) FaceBlurActivity.java reads a video and writes to a new one, but apparently the resolution of the new saved video is lower than the original one:
   This may be the problem of FFmpegFrameRecorder itself. I did not do enough research to tell the reason why so please do more research on it to see whether there is a way to improve video quality.

Fall 2016 by Yina Zhu