Design Of Video Bitrate Analyzer Based On Swift

Xiuyu Zhong [1*]，Zhongyi Luo [2]

[1,2] School of Computer Science, Jiaying University, Guangdong, China

Abstract. Network video is very popular, users often upload video, Video site allows specific coded video to maintain the original specification at a certain peak rate, and then to add lower-level specifications to the code. At present, many analysis tools can only get the maximum rate of the whole video, and cannot get the rate map, nor support the new video format. This design proposes using the Swift language to display the video bitrate analysis data in different forms, FFmpeg/ffprobe applications is called to analyze video stream. The experimental results show that the design can get the information of all frames of the video to depict the column chart, and can display the corresponding time point of the frame, so that users can modify the area exceeding the rate limit, so as to achieve the goal of no forced transcoding. In addition, this design supports HEVC format video.

1 Introduction

The network video is very popular, users shoot what have been seen and heard, record game video, and upload the video to share. At the beginning, because of bandwidth, hardware and other reasons, the video sites will force transcoding to users' uploaded videos, compress into different specifications to choose for users. The resulting problem is not smooth (60 fps to 24 fps) and indistinct (1080p to 720p). With the progress of technology, video sites relax the terms and no longer force transcoding, allow specific coded video to maintain its original specifications at a certain rate of peak value, and then transcode new low-level specifications. Video bitrate analysis is particularly important before uploading video.

Many scholars have made research on video rate analysis, Dongyan Zhang and Hui He proposed a server-side-based rate allocation algorithm for content delivery networks [1]. Hyoungseok Kim and Joonseok Park proposed new video quality assessment metric using intensity variation analysis [2]. Jie Wang and Linge Li proposed algorithm to speed up the transcoding procedure, which mainly utilizes information from the input video stream such as Code Unit depths, Prediction Unit partitions [3]. Selvaraj Kesavan and J. Jayakumar proposed client-driven three-level optimized rate adaptation algorithm for adaptive HTTP media streaming, the algorithm controlled and minimized the effects of buffer stalls and overflow resulting from the brief network variations occurring between consecutive segments [4]. NK Karn，H Zhang and F Jiang presented a comprehensive analysis of a dynamic network environment for streaming of 3D MVD over Internet [5]. Jian Chen and Shuai Zheng proposed a frame-level distributed video coding system based on encoder rate control [6]. YSDL Fuente，R Skupin and T Schierl proposed the technique which generated a single video bit stream out of the selected tiles so that a single hardware decoder can be used [7]. Jacob Søgaard and Muhammad Shahid presented insights into a study that investigates perceptual preferences of various adaptive video streaming scenarios through crowdsourcing based and laboratory based subjective assessment [8].

These studies mainly improved the efficiency of analysis from algorithm, the specific application was less. In addition, ordinary tools such as Mediainfo can only get the maximum of the entire video, but not the rate distribution map. Under the Windows system, there is a very powerful video rate analysis tool Bitrate Viewer, it can analyze a variety of video coding, and display the results according to three different methods of data processing: Frame, GOP and Second [9]. Under the macOS system, Udo Gerber has been transplanted. But both have not been updated for years and unable to support new video coding, such as HEVC. And the price of both is too expensive, which is prohibitive.

2 Brief introduction of Swift

Swift is a powerful and intuitive programming language, which is created by Apple and can be used to develop APP for iOS, Mac, Apple TV and Apple Watch. Swift can provide real-time feedback, and can be seamlessly integrated into existing Objective-C code. Swift code guarantees security from design, developers can write secure and reliable code, have a very rich APP experience while saving time. Swift is free and open source, it provides binary files for the OS X and the Linux platform，these files can compile code in iOS, OS X, watchOS, tvOS and Linux. Writing Swift code is a fun interactive process, because Swift syntax is concise, but expressive enough to develop software that runs as fast as lightning [10].

* Corresponding author: Xiuyu Zhong: mzzxy@jyu.edu.cn

3 Design of Analyzer

3.1 Procedure Design

Ffmpeg is uses to analyze video streams, it is an open source computer program which can be used to record and convert digital audio and video, and transform them into into streams [11]. The analysis process has seven steps, including calling FFmpeg/ffprobe applications to analyze the first video stream, extracting information from each frame, extracting information from the video stream, calculating the video stream, parsing the JSON data, displaying the details of each data, and dragging the video file to the program window. The details are shown below.

（1）Analyze the information of the first video stream by calling FFmpeg/ffprobe applications.

（2）Extract information from each frame, including timestamp , duration, size and type, the data are displayed in three formats: Frame, GOP and Second.

（3）Extract information of the video stream, including width, height, average frame rate , time base, length of time, average bitrate.

（4）Calculate the minimum frame rate, maximum frame rate and total number of frames.

（5）Use Codable protocol of Swift 4 to analyze JSON data directly.

（6）Click the column chart data to display the details of each item data.

（7）Drag video files to the program window.

3.2 Module Design

(1) Frame module design

The module calls FFmpeg/ffprobe applications to analyze the first video stream, each frame is a statistic, Ffprobe extracte information by default.

(2) GOP module design

Using one frame as segmentation, All frames before the first frame to the second frame are combined into a statistical item, and so on. The program is as follows.

```
extension Array where Element: TypeEquatable & DurationEquatable {
func eachSlice<S>(transfrom: (ArraySlice<Element>) -> S) -> [S] {
var result = [S]()
var sliceDuration = CMTimeValue(0)
var startIndex = 0
for i in 0 ..< count {
sliceDuration += self[i].duration
if (self[i].type == .I && i > 0) || i == (count - 1) {
result.append(transfrom(self[startIndex ..< i]))
sliceDuration = self[i].duration
startIndex = i
}
}
return result
}
}
```

(3)Second module design

Using second as segmentation, All frames from seconds to seconds are combined into a statistical item, and so on. The program is as follows.

```
extension Array where Element: DurationEquatable {
func eachSlice<S>(duration: CMTime, transfrom: (ArraySlice<Element>) -> S) -> [S] {
var result = [S]()
var sliceDuration = CMTimeValue(0)
var startIndex = 0
for i in 0 ..< count {
sliceDuration += self[i].duration
if (sliceDuration > duration.value && i > 0) || i == (count - 1) {
result.append(transfrom(self[startIndex ..< i]))
sliceDuration = self[i].duration
startIndex = i
}
}
return result
}
}
```

(4) Time formatted output module design

```
extension CMTime: CustomStringConvertible {
public var description: String {
let seconds = CMTimeGetSeconds(self)
let hour = Int(seconds / 3600)
let minute = Int(seconds % 3600 / 60)
let second = Int(seconds % 60)
let millisecond = Int(seconds * 1000 % 1000)
if hour > 0 {
return String(format: "%i:%02i:%02i:%03i", hour, minute, second,millisecond)
        }
 else {return String(format: "%02i:%02i:%03i", minute, second, millisecond)}
}
}
```

(5) Drag module design

```
protocol DragViewDelegate {func dragged(with file: URL) }
class DragView: NSView {
var delegate: DragViewDelegate?
convenience init() {
self.init(frame: .zero)
registerForDraggedTypes([.fileURL])
                    }
override func draggingEntered(_ sender: NSDraggingInfo) -> NSDragOperation {
layer?.backgroundColor = NSColor.gray.cgColor
let sourceOperationMask = sender.draggingSourceOperationMask()
return sourceOperationMask.contains(.generic) ? .generic : []
                    }
override func draggingEnded(_: NSDraggingInfo) {
layer?.backgroundColor = NSColor.clear.cgColor
}
override func performDragOperation(_ sender: NSDraggingInfo) -> Bool {
let pasteboard = sender.draggingPasteboard()
if pasteboard.types!.contains(.fileURL) {
let files = pasteboard.readObjects(forClasses: [NSURL.self]) as! [URL]
if supportedFileTypes.contains(files.first!.pathExtension.lowercased()) {
delegate?.dragged(with: files.first!)
return true
}
}
return false
}
}
```

4.Test Results and Future Work

The video bitrate analyzer displays the results according to the three methods of data processing: Frame, GOP and Second. The test results are shown below.

4.1 Test results

The bitrate analysis results are shown in figure 1, figure 2 and figure 3.

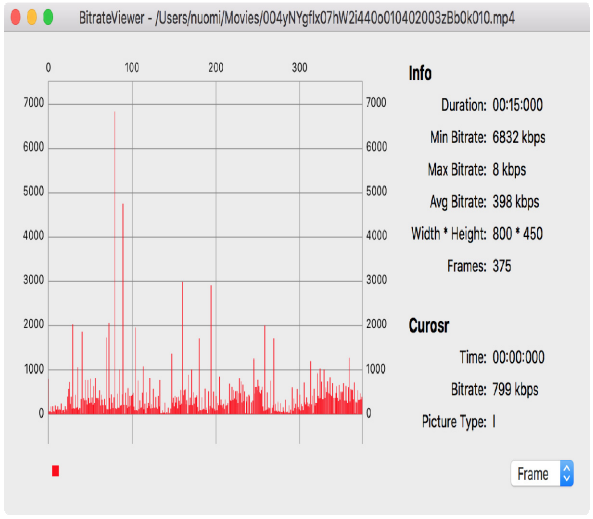

Figure 1. The bitrate analysis result of Frame mode

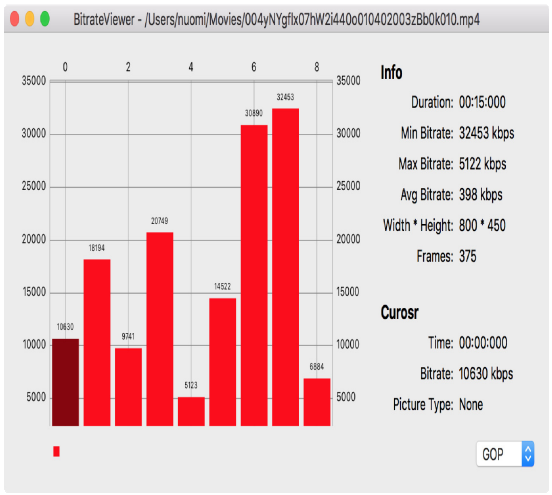

Figure 2. The bitrate analysis result of GOP mode

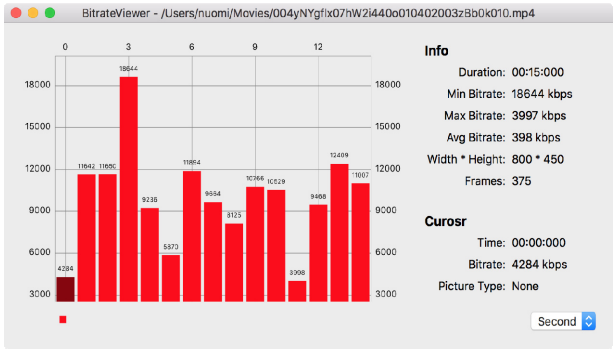

Figure 3. The result bitrate analysis of Second mode

4.2 Future Work

The analysis of video stream uses ffprobe directly, because ffprobe is a CLI program, Swift needs to take over cout and store it in memory temporarily when it calls, after waiting for the program to quit completely, saves the cout result to the JSON file, finally, through the parsing tool JSON in Swift 4, parses the JSON into the variables of the program running. These steps are obviously superfluous. The C++ hybrid programming should be used, analyze the video file directly and save the result to the variable. If it is needed to be saved, it is saved to disk in JSON form.

Because in the AutoLayout layout, DragView is on the top level, thus covering BarChartView, column chart data can be clicked sometimes and not clicked sometimes. In addition, only MP4 and flv containers are supported, and flv is not fully supported.

Acknowledgments

The authors want to thank construction project of college students' practice education base in local universities of Education Ministry and the teaching quality and teaching reform project in universities of Guangdong Province for their general support for the research (with grant NO. [2013]48 Teacher's letter and [2016]233 High letter of Yue Jiao respectively).

References

1. Dongyan Zhang,Hui He. Bitrate allocation among multiple video streams to maximize profit in content delivery networks. Personal and Ubiquitous Computing,June 2016, Volume 20, Issue 3, pp 385–396
2. Hyoungseok Kim,Joonseok Park. Efficient video quality assessment for on-demand video transcoding using intensity variation analysis . The Journal of Supercomputing, January 2018 ,Volume 74, Issue 1,pp 1-15
3. Jie Wang,Linge Li. Efficient algorithms for HEVC bitrate transcoding. Multimedia Tools and Applications,December 2017, Volume 76, Issue 24, pp 26581–26601
4. Selvaraj Kesavan,J. Jayakumar. Effective client-driven three-level rate adaptation (TLRA) approach for adaptive HTTP streaming. Multimedia Tools and Applications,April 2018, Volume 77, Issue 7, pp 8081–8114
5. NK Karn , H Zhang , F Jiang. User-perceived quality aware adaptive streaming of 3D multi-view video plus depth over the internet.Multimed Tools Appl (2018). https://doi.org/10.1007/s11042-018-5744-8
6. Jian Chen, Shuai Zheng,Qing Hu, Yonghong Kuo. A frame-level encoder rate control scheme for transform domain Wyner-Ziv video coding.

Multimedia Tools and Applications, October 2017, Volume 76, Issue 20, pp 20567–20585

7. YSDL Fuente， R Skupin， T Schierl. Video processing for panoramic streaming using HEVC and its scalable extensions. Multimedia Tools and Applications, February 2017, Volume 76, Issue 4, pp 5631–5659
8. Jacob Søgaard,Muhammad Shahid,Jeevan Pokhrel,Kjell Brunnström. On subjective quality assessment of adaptive video streaming via crowdsourcing and laboratory based experiments. Multimedia Tools and Applications,August 2017, Volume 76, Issue 15, pp 16727–16748
9. http://www.winhoros.de/docs/bitrate-viewer/
10. https://developer.apple.com/cn/swift/
11. https://ffmpeg.org/doxygen/trunk/index.html