doi:10.3969/j.issn.1672-4348.2020.06.013

基于安卓系统的均衡器算法的仿真及应用

林金阳,庄伟达

(福建工程学院 微电子技术研究中心,福建 福州 350118)

摘要:针对安卓系统原生 Audio Effect 框架不支持 32 bit 音频的问题,提出了采用 FFmpeg 开源音视频框架自定义新的 Audio Effect 框架。选用高阶音频参数多波段均衡器算法的巴特沃斯部分作为滤波,对该算法进行分析、验证和应用测试。仿真和测试结果显示,该方法可以提高均衡准确性,达到设计要求。

关键词:安卓;Audio Effect 框架;FFmpeg 音频均衡器;应用层开发

中图分类号:TP37　　**文献标志码**:A　　**文章编号**:1672-4348(2020)06-0583-04

Simulation and application of equalizer algorithm based on Android

LIN Jinyang, ZHUANG Weida

(Research Center for Microelectronics Technology, Fujian University of Technology, Fuzhou 350118, China)

Abstract: In response to the problem that native Audio Effect framework of the Android system does not support 32-bit audio, a new Audio Effect framework was proposed by using the FFmpeg open source audio and video framework. The Butterworth part of its high-order audio parameter multi-band equalizer algorithm was used as the filter, and the algorithm was analyzed, verified and tested in application. Simulation and test results show that this method can improve the accuracy of equalizer and meet the design requirements.

Keywords: Android; Audio Effect framework; FFmpeg audio equalizer; application layer development

近年来,数字均衡器算法随音频质量需求的提升也在不断发展,但与应用平台的结合研究相对较少。如今基于安卓系统的智能手机市场占有率达 80%以上,数字均衡器算法与安卓系统的结合应用很重要[1],对安卓系统平台音效处理的研究也尤为必要[1-2]。

目前主流安卓系统中的音频均衡器部分,其原生 Audio Effect 框架仅能支持 16 bit 音频数据处理,对更为流行的 32 bit 音频缺乏支持。FFmpeg 是一个集多媒体转码、音视频流化传输于一体的第三方开源框架,能够实现对音频数据及视频数据的编解码工作,支持各种格式的转换,广泛应用于各类音乐播放软件、视频播放软件以及直播软件中[3-5]。FFmpeg 开源音视频框架包含有许多均衡器算法,如 allpass、anequalizer、bandpass、bass、biquad 以及 treble 等[6]。本研究选用安卓平台深入分析高阶音频参数多波段均衡器算法,并进行仿真验证和应用研究。

1 高阶音频参数多波段均衡器分析与仿真

1.1 算法运行流程分析

音频均衡器算法实际上就是滤波器原理设计,可分为 3 步。首先,获得具体的模拟域的传输函数模型,根据每个类型的滤波器推导出具体的设计参数;其次,根据双线性变换法把第一步得到的滤波器数字化;最后,根据频率变换把数字化后的数字低通滤波器转换到对应的中心频率点。

收稿日期:2020-09-24
基金项目:平板显示国家地方联合工程实验室开放基金项目(KF1802,GY-Z18038)
第一作者简介:林金阳(1981—),男,福建莆田人,副教授,博士,研究方向:电子信息显示。

FFmpeg 音视频框架中的高阶音频参数多波段均衡器算法便是遵循该设计方法所得,该算法中共包含 3 种原型滤波器类型,依次为巴特沃斯型、切比雪夫 I 型以及切比雪夫 II 型[7]。

在使用时,FFmpeg 根据音频数据的带宽不同选用相应的类型进行数据处理,最后通过差分方程计算得到新的音频数据并输出,本研究选用与平台音频均衡器框架结合较好的巴特沃斯型作为滤波器,具体过程如下:

(1)根据系统输入的增益大小进行带宽增益的计算,在高阶音频参数多波段均衡器算法中,该部分为 compute_bw_gain_db;

(2)计算滤波器设计参数进而得到滤波器系数,在高阶音频参数多波段均衡器算法中,此部分为 bp_filter;

(3)最后,经由差分方程计算获得经过处理后的输出数据,在高阶音频参数多波段均衡器算法中,差分方程计算由 fo_section 部分完成[8]。

分析高阶音频参数多波段均衡器算法源码可知,当音频数据输入,系统首先需要根据用户设置的峰值增益 G 计算出对应的带宽增益 G_B;再根据每个类型的滤波器传输函数模型推导出滤波器设计参数,不同类型的滤波器所需计算的设计参数也不一样,从而获得不同的数字低通坡型滤波器系数 $a_0 \cdots a_4$ 和 $b_0 \cdots b_4$,以及带通均衡器二阶和四阶子块的系数 $a_0 \cdots a_4$ 和 $b_0 \cdots b_4$;最后由输入的音频数据以及计算所得滤波器系数根据系统的差分方程计算得到新的输出,至此用户所设增益才算生效。图 1 为 FFmpeg 的运行流程图。

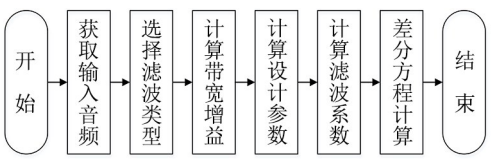

图 1 FFmpeg 主要部分运行流程

Fig.1 Workflow of the major part of FFmpeg

1.2 带宽增益计算

在响应用户设置增益时,多频段音频均衡器的相邻频段通常会相互影响,使得音频均衡器系统无法得到一个较为平坦的通带。所以,在高阶音频参数多波段均衡器算法中,针对不同类型的滤波器特点设计相应的带宽增益计算方法。相对切比雪夫 I 型以及切比雪夫 II 型滤波器,巴特沃斯型与 Android 系统结合的效果更好,故本研究仅选用巴特沃斯型滤波器进行算法仿真及音频均衡器开发[9]。

针对巴特沃斯型滤波器,高阶音频参数多波段均衡器算法设计其带宽增益计算公式为:

$$G_B = \begin{cases} G + 3, G \leqslant -6 \\ G * 0.5, -6 \leqslant G \leqslant 6 \\ G - 3, G \geqslant 6 \end{cases} \tag{1}$$

1.3 巴特沃斯型音频均衡算法部分

针对巴特沃斯型滤波器的零极点进行计算可得,巴特沃斯型滤波器模拟域传输函数为:

$$H_a(s) = \left[\frac{g\beta + g_0 s}{\beta + s}\right]^r \prod_{i=1}^{L}\left[\frac{g^2\beta^2 + 2g\,g_0\,s_i\beta s + g_0{}^2 s^2}{\beta^2 + 2\,s_i\beta s + s^2}\right] \tag{2}$$

其中,各设计参数定义为:

$$g = G^1/N \tag{3}$$

$$g_0 = G_0^1/N \tag{4}$$

$$\beta = \varepsilon^{-1}/N\ \Omega_B = \varepsilon^{-1}/N\tan(\Delta\omega/2) \tag{5}$$

$$\Omega_B = \tan(\Delta\omega/2) \tag{6}$$

式中,设左边界频率 f_1 转换到数字域为 ω_1,右边界频率 f_2 转换到数字域为 ω_2,则 $\Delta\omega = \omega_1 - \omega_2$。

$$\varepsilon = \sqrt{(G^2 - G_B^2)/(G_B^2 - G_0^2)} \tag{7}$$

$$u_i = (2i - 1)/N, i = 1,2,\cdots,L \tag{8}$$

$$s_i = \sin(\pi \times u_i/2), i = 1,2,\cdots,L \tag{9}$$

$$D_i = \beta_2^2 + 2 \times s_i \times beta + 1, i = 1,2,\cdots,L \tag{10}$$

根据高阶音频参数多波段均衡器算法源码可知,其算法中 N 为 4,L 为 2。

得到上述模拟域传输函数以及各设计参数后,应用双线性变换公式便可推出系数计算式[10],在高阶音频参数多波段均衡器算法源码中,其系数计算部分分为两种情况:

当 $c_0 = \cos\omega_0$ 的值为 1 或-1 时,

$$\begin{cases} a_0 = 1 \\ a_1 = 2 \times c_0 \times (\beta^2 - 1)/D_i \\ a_2 = (\beta^2 - 2 \times \beta \times s_i + 1)/D_i \\ b_0 = (g^2 \times \beta^2 + 2 \times g \times g_0 \times s_i \times \beta + g_0^2)/D_i \\ b_1 = 2 \times c_0 \times (g^2 \times \beta^2 - g_0^2)/D_i \\ b_2 = (g^2 \times \beta^2 - 2 \times g \times g_0 \times s_i \times \beta + g_0^2)/D_i \end{cases} \tag{11}$$

当 $c_0 = \cos\omega_0$ 的值不为 1 或-1 时,

$$\begin{cases} a_0 = 1 \\ a_1 = -4 \times c_0 \times (1 + s_i \times \beta) / D_i \\ a_2 = 2 \times (1 + 2 \times c_0^2 - \beta^2) / D_i \\ a_3 = -4 \times c_0 \times (1 - s_i \times \beta) / D_i \\ a_4 = (\beta^2 - 2 \times s_i \times \beta + 1) / D_i \\ b_0 = (g^2 \times \beta^2 + 2 \times g \times g_0 \times s_i \times \beta + g_0^2) / D_i \\ b_1 = -4 \times c_0 \times (g^2 + g \times g_0 \times s_i \times \beta) / D_i \\ b_2 = 2 \times [g^2{}_0 \times (1 + 2 \times c_0^2) - g^2 \times \beta^2] / D_i \\ b_3 = -4 \times c_0 \times (g^2 - g \times g_0 \times s_i \times \beta) / D_i \\ b_4 = (g^2 \times \beta^2 - 2 \times g \times g_0 \times s_i \times \beta + g_0^2) / D_i \end{cases} \tag{12}$$

1.4　差分方程计算

通过参数计算后,得到根据用户所设增益大小计算而来的滤波器系数 $a_0 \cdots a_4$ 和 $b_0 \cdots b_4$,差分方程的作用就是将前文计算所得的滤波器系数与原始音频的数据进行计算,得到新的音频数据。在高阶音频参数多波段均衡器算法中,也有其自己选用的差分方程式,差分方程的具体计算式为:

$$\begin{aligned} y[n] = {} & b_0 \times X[n] + b_1 \times X[n-1] + \\ & b_2 \times X[n-2] + b_3 \times X[n-3] + \\ & b_4 \times X[n-4] - a_1 \times y[n-1] - \\ & a_2 \times y[n-2] - a_3 \times y[n-3] - \\ & a_4 \times y[n-4] \end{aligned} \tag{13}$$

式中 $y[n]$ 为输出,$X[n]$、$X[n-1]$、$X[n-2]$、$X[n-3]$、$X[n-4]$ 、$y[n-1]$、$y[n-2]$、$y[n-3]$、$y[n-4]$ 为历史音频数据。

根据式(13)计算完成后,输出的音频数据便为与用户设计的增益相匹配的输出,从而可根据用户的偏好调整音乐效果。目前大多数音频均衡器的增益大小设置为[-15,+15]或[-6,+6],而可调频率范围为 20 ~20 000 Hz,用户根据个人的听感喜好,调节不同频段的增益大小,系统根据用户的设置,处理完音频数据后输出,便完成了整个音频数据处理的全部流程。

2　仿真与应用分析

2.1　FFmpeg 音频均衡器仿真

本研究根据 FFmpeg 音视频框架中的高阶音频参数多波段均衡器算法原理,选择将其在 MATLAB 平台进行算法的仿真分析。

图 2 为巴特沃斯频响曲线,其中曲线为频响曲线,小圆点为所设置增益大小。选定算法后,便可对算法做正确性的验证。以正弦波作为输入波形进行测试。图 3 为输入波形频率为 125 Hz 和 500 Hz 的相应图,横轴为采样点,纵轴为振幅大小。从图 3 可以看出,该算法经仿真后,可以按照系统设定增益进行工作,正确实现均衡器的功能,符合预期要求。

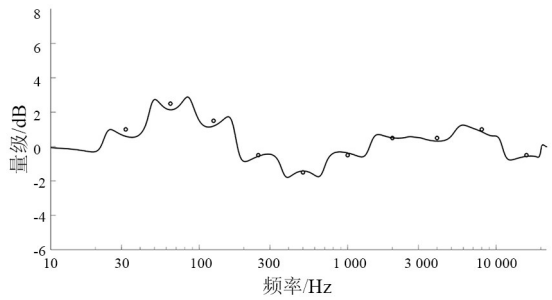

图 2　巴特沃斯频响曲线

Fig.2　Frequency response curve of the Butterworth type equalizer algorithm

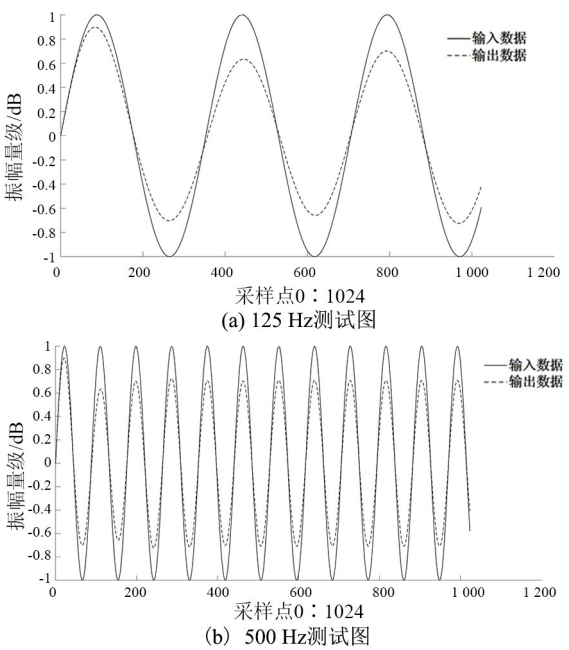

图 3　不同输入波形频率的测试图

Fig.3　Diagrams of different input waveform frequencies

2.2　应用分析

根据仿真验证,在 Android Studio 下进行相关开发操作,首先点击获取音乐列表进行音乐选择,其次选择所需处理声道,可单独选择左声道或右声道,也可选择两个声道同时处理的立体声,然后

设置各频段增益,最后点击生成音乐便可在指定路径下生成处理后的输出文件并进行播放。得到处理后的文件后,可将输入以及输出文件载入 Adobe Audition 软件中,通过该软件得到输入以及输出文件的音频数据,同时依据音频数据便可绘制出其相应的频率分析曲线。

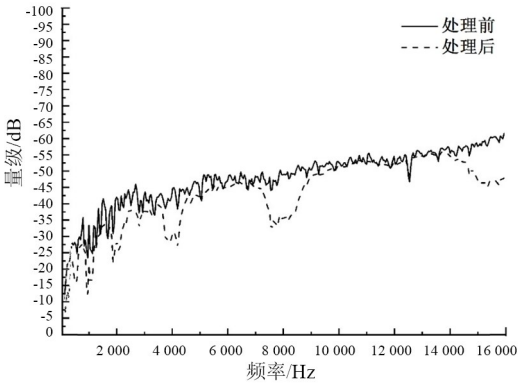

图 4 音效处理频率分析曲线

Fig.4 Frequency analysis curves of the audio processing

将左声道设置为-15 dB 增益,右声道设置为+15 dB 增益,对左右声道的处理前后频率进行分析。处理前后左、右声道频率分析曲线对比如图 4 所示,从图 4 可以看出,正负增益情况均表现正确。由此可见,将 FFmpeg 音视频框架中所包含音频均衡器算法应用于 Android 平台的均衡器开发的方案是可行的,

3 结语

从应用层方向验证了 FFmpeg 音视频框架中的高阶音频参数多波段均衡器算法在 Android 平台实现均衡器功能的可行性。采用 JNI 的开发方式,通过 FFmpeg 的命令行方式进行了算法的应用方案测试,实验结果表明,高阶音频参数多波段均衡器算法可以在应用层实现均衡器的功能。

但是,对于高阶音频参数多波段均衡器算法的底层移植方案,在测试时还是未能达到开发要求,后续可进一步对原生 Audio Effect 框架深入分析,并完善算法从浮点到定点的转换工作;完成算法移植后,可仿照原生 Audio Effect 框架进行新框架的自定义开发,解决其仅支持 16 bit 支持音频数据处理的问题。

参考文献:

[1] SOLYMAN A, ATTAR H, KHOSRAVI M, et al. A low-complexity equalizer for video broadcasting in cyber-physical social systems through handheld mobile devices[J]. IEEE Access, 2020, 3: 67591-67602.

[2] 冯启朋, 杨飞然, 杨军. 基于 Android 平台的音效系统设计与实现[J]. 网络新媒体技术, 2016, 5(4): 16-22.

[3] 井洪亮. 基于 Android 的 H.264/AVC 解码器的设计与实现[D]. 哈尔滨: 哈尔滨工业大学, 2010.

[4] 刘振. 基于 Android 平台的 HEVC 到 H.264 视频转码设计与实现[D]. 南京: 南京邮电大学, 2016.

[5] ZHONG X, LUO Z. Design of video bitrate analyzer based on Swift [C]//EITCE 2018, 2018: 1-4.

[6] ZENG H, SHI L, ZHANG Z. Research and Implementation of video codec based on FFmpeg[C]//2016 International Conference on Network and Information Systems for Computers, 2016, 54: 184-188.

[7] 吴礼仲. 音频均衡器算法研究与实现[D]. 西安: 西安电子科技大学, 2010.

[8] VÄLIMÄKI V, REISS J. All about audio equalization: solutions and frontiers[J]. Applied Sciences, 2016, 6: 1-46.

[9] JOT J. Proportional parametric equalizers-application to digital reverberation and environmental audio processing[C]//Proceedings of the 139th Convention of the Audio Engineering Society, 2015: 1-8.

[10] 周耀辉, 王芸波, 朱维新, 等. IIR 数字滤波器设计[J]. 电力自动化设备, 2010, 30(9): 129-131.

(责任编辑: 方素华)