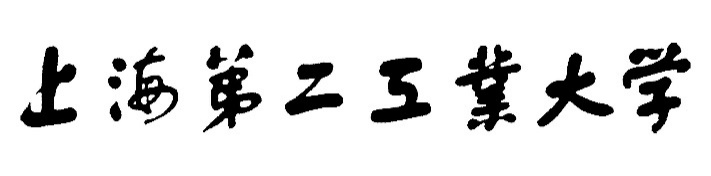
****

**本科毕业设计（ 论文 ）**



题 目：基于Android平台的数字音频及图像分析

处理APP

学 号：　　20124822309

姓 名：　　　乔乙桓

班 级：　　 12电信A1

专 业： 电子信息工程

学 院：　　 工学部

入学时间：　　　2012级

指导教师：　　　胡金艳

日 期： 2016年5月 5 日

毕业设计（论文）独创性声明

本人所呈交的毕业论文是在指导教师指导下进行的工作及取得的成果。除文中已经注明的内容外，本论文不包含其他个人已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中作了明确说明并表示谢意。

作者签名：

日期：

基于Android平台的数字音频及图像分析处理APP

摘要

Android目前已成为全球最普遍手机操作系统，三星、华为、步步高等手机制造商已经凭借Android获得了巨大的成功。

本论文主要研究如何在Android平台上，实现数字音频以及图像分析处理。音频侧重分析，需要实时显示波形；除了音频的采集使用Google提供的接口外，凡是涉及到的数字信号处理（DSP）算法一律采用Java编程；分析功能包括：时域分析、滑动平均处理、FFT、FFTshift以及平均谱。图像侧重处理，使用Google提供的ColorMatrix，处理功能包括：修改图像的饱和度、色相以及亮度。另外，还实现了包括将采集到的音频存储为WAV文件格式，以及可以选择JPEG、PNG、WEBP三种格式对图像进行压缩和存储。

关键词: Android；Java；FFT；DSP；采样定理；平均谱；滑动平均

Digital Audio and Image Analysis and Processing Based On Android Platform

ABSTRACT

Android system has become the world's most widely used mobile operating system. Samsung, Huawei, BBK and other mobile phone manufacturers rely on Android system to get a great success.

This thesis studies how to achieve digital audio and image processing on the Android platform. The Audio part focuses on analysis, which needs to display the real-time waveform. Except that the audio acquisition uses the interface provided by Google, all the Digital Signal Processing (DSP) algorithms use Java programming. The audio analysis includes time-domain analysis, moving average processing, FFT, FFT shift and the average spectrum. The Image part focuses on processing. It uses the ColorMatrix provided by Google. The image processing includes adjustments of the saturation, hue and luminance. In addition, the storage function is implemented for users to save the recorded audio into the WAV format, and to compress and save the processed image in one of the JPEG, PNG, WEBP formats.

Key words: Android; Java; FFT; DSP; Sampling theorem; Average spectrum; Moving Average

**目录**

[1 绪论 1](#_Toc451763738)

[1.1 研究背景和意义 1](#_Toc451763739)

[1.2 国内外发展现状 2](#_Toc451763740)

[1.3 可行性分析 4](#_Toc451763741)

[1.3.1 需求分析 4](#_Toc451763742)

[1.3.2 运行环境 5](#_Toc451763743)

[1.3.3 开发工具 6](#_Toc451763744)

[1.4 论文主要研究内容 6](#_Toc451763745)

[2 数字音频处理的基本原理 7](#_Toc451763746)

[2.1数字音频的基本概念 7](#_Toc451763747)

[2.2 音频信号的采集 8](#_Toc451763748)

[2.3 涉及到的主要分析方法 13](#_Toc451763749)

[2.3.1时域分析 13](#_Toc451763750)

[2.3.2频域分析 13](#_Toc451763751)

[2.4 本章小结 15](#_Toc451763752)

[3 数字音频分析的APP设计 16](#_Toc451763753)

[3.1 系统软件设计 16](#_Toc451763754)

[3.1.1 音频的数据走向 16](#_Toc451763755)

[3.1.2 绘制波形 17](#_Toc451763756)

[3.2 音频处理系统架构 19](#_Toc451763757)

[3.2.1 项目结构 19](#_Toc451763758)

[3.2.2 系统架构 20](#_Toc451763759)

[3.2.3 系统进程视图 21](#_Toc451763760)

[3.2.4 系统界面设计 22](#_Toc451763761)

[3.3 音频分析结果 25](#_Toc451763762)

[3.4 本章小结 25](#_Toc451763763)

[4 数字图像的基本原理 26](#_Toc451763764)

[4.1 数字图像的基本概念 26](#_Toc451763765)

[4.2 涉及的主要处理方法 27](#_Toc451763766)

[4.3 本章小结 28](#_Toc451763767)

[5 数字图像处理的APP设计 29](#_Toc451763768)

[5.1 软件设计 29](#_Toc451763769)

[5.2图像处理结果 30](#_Toc451763770)

[5.3 本章小结 30](#_Toc451763771)

[6 结论 31](#_Toc451763772)

[致谢 32](#_Toc451763773)

[参考文献 33](#_Toc451763774)

[附录 34](#_Toc451763775)

[FFT源码 34](#_Toc451763776)

# 

# 1 绪论

## 研究背景和意义

声音是一种重要的信息交换途径。在这个信息发达的社会中，人类广泛地将语音信号转换为数字信号进行声音的传送、合成、存储、还有识别。高品质的窄带通信系统、智能机器人、翻译机等应用都用到了数字信号处理技术。伴随微电子技术以及大规模集成电路的快速发展，基于数字信号处理的系统正逐步走向实用化。数字语音信号处理是一门刚新起的学科，它是语音学和数字信号处理的结合。它同语言学、人工智能、计算机科学以及模式识别等学科有着密切联系。数字语音处理理论的发展离不开这些学科的发展，而数字语音信号处理技术的进步也会促进相关领域的进步。进行数字信号处理是为了获得到音频的特征参数，以便高效地存储或传输；或者是通过一定处理运算以实现某种用途的要求，例如语音合成、语音识别等。

图像可以用来表达信息以及传递信息。据不完全统计，视觉信息在人类接收到的信息中所占比例高达。因此数字图像处理已经成为信息科学、地理气象等诸多学科的重要工具。数字图像处理是利用计算机实现对图像的处理，因此也可以称作“计算机图像处理”。对图像进行处理可以改善图像的最终呈现效果，以提高图片的质量。利用数字图像处理可以从图片中能够提取出它包含的某些有用信息,来使得计算机更加容易分析图像。如果对图像进行编码和压缩，可以减少图像占用的存储空间，便于存储和传输。目前，数字图像处理已经成为计算机视觉、地质学、气象学等学科主要学习和研究的对象。

本论文音频部分主要可以显示声音的时域以及频谱波形。波形其实就是波的图像，声音不同波形也会不同。当声音的幅度发生变化时，波形会呈现不断起伏的波峰；当声音的频率发生变化，波形将会像弹簧一样的伸缩。悦耳的声音，其波形也美丽而圆滑；嘈杂的声音其波形也会混乱而繁杂。波形有很多种类，针对不同的情况要选择适当的方法进行测量。为了在Android平台实现相关功能有必要介绍下Android系统的背景。

Android操作系统的底层是是Linux核心，它主要应用于手机等移动设备。目前国内对Android开发人才的需求也在迅速增长。随着技术的不断进步，Android手机越来越接近小型电脑，甚至比它的功能更强大。因为电脑不方便携带，而手机可以在某些场合下可以完成电脑所能完成的任务，所以手机软件在未来的IT行业中占有重要地位。对于Java语言而言，Android系统给了Java一个新的机会。以前Java语言主要用做服务器端编程语言，Java EE平台一直是移动、电信、银行、电子商务应用的首选平台。但作为客户端语言开发，Java语言却表现不是很好，虽然Java自己的的界面开发库，但对于从事客户端开发的人员而言，大多不愿意选择Java语言。而Android系统的出现改变了这种糟糕的局面。Android作为一款非常优秀的手机、平板电脑操作系统，主要采用Java语言开发客户端。这就使得越来越多的开发者选择Java。不过运行Android系统的手机计算能力有限，不适合在Android平台上，部署大型企业级应用，却更可能以纯粹客户端应用出现，然后通过互联网与大型应用进行交互，用作大型企业应用的客户端，例如，淘宝、QQ、微信、美图秀秀等Android客户端，它们都是这种发展趋势下的产物。

研究本论文从理论层面，用户通过将音频信号、图像信息转换到变换域研究其特征和处理方法，这不仅验证了信号与系统等相关课程的理论，还加深了知识的理解与应用。另外，将采集到的声音与图像进行分析处理，方便数据的存储与传输，通过对信号的处理来满足手机用户的实际使用需求。作为本论文的研究的最后成果——数字信号分析处理开发库，脱离硬件平台的束缚，有利于进行二次开发，不必反复检查修改底层代码，可以快速实现开发与维护。

## 国内外发展现状

大约在一两千年前，人们才开始对声音的研究，由于当时并没有合适的测量仪器,所以在很长一段时间内，人类只能通过耳来倾听和用口模仿来进行研究的。所以这种声音的研究常被人叫做“口耳之学”，所以对语音的描述也只能停留在定性的描述上。真正意义上的研究从1876年开始的，贝尔发明了电话，首次使用声电，实现了音频的远距离传输。1939年Homer Dudley成功的发明了第一个声码器，奠定了语音产生模型的基础。19世纪60年代，核姆霍兹使用传统发的声学方法，对元音和歌唱进行了研究，从而奠定了语音的声学基础。19世纪40年代，美国的Haskins实验室成功研制 了“语音回放机”，该仪器可以把薄膜片上手工绘制的图转换为语音，从而进行语音合成。20世纪40年代，语谱图仪问世了。它将语音转换为可见的语谱图，从而得到了“可见语音”[7]。

20世纪50年代，人们开始利用计算机完成数字信号处理的相关工作，这使得过去受人力和时间限制的语音统计工作获得解放。从此语音信号的处理不论在理论研究，还是在技术应用都取得了惊人的进展。数字信号处理可以用来识别人声、鉴别语种、合成语音等，但其中最具挑战性的是语音识别。

国内语音识别的研究的工作是从20世纪50年代开始的，以中科院声学所为代表开始研究的汉语全音节识别，20世纪80年代该研究取得了突破进展，部分汉语输入系统已经很实用。20世纪年代初，在国家的支持下成立了专门的语音识别项。中科院模式识别研究室对于不限定人汉语、连续语音听写系统的出错率能限制到10%以下。为将后发展民族化语音产业打下非常坚实的技术基础。

20世纪70年代，我国工程师开始使用计算机编程。最初使用机器语言进行编程，烧入晶体管的108乙机运行，这种工作有时需要数周甚至更久才能调试出一个小程序。随着技术的发展，转为使用Nov1200、z80、Inter8086等，编程时使用的汇编语言也转变到BASIC、FORTRAN等语言[10]。数字信号处理研究的基础是FFT和Filter，但不同的机器或平台需要使用不同的语言，这样就做了大量重复性的工作同时还极其耗费时间和精力。到了21世纪，我国工程师开始使用MATLAB语言。 MATLAB编程除了语法简单，还有全世界的精英为它开发的强大工具箱。基础的处理程序都包含在工具箱中不需要用户去开发。对于普通用户而言，只需要利用工具箱来实现自己的算法。MATLAB不需要编译链接等操作，输入程序后即可运行。使用MATLAB处理音频图像前需要掌握一下几方面知识，其中包括音频图像处理的理论部分、数字信号处理理论基础和MATLAB编程。为了在Android平台实现在数字信号处理的相关算法，不仅需要了解Android操作系统底层库对于音频图像的支持，也需要参考国内外相关Android应用。例如，腾讯计算机系统有限公司在交新版的QQ软件中加入了语音聊天变音功能，天天动听音乐播放器在播放音乐时可以调节均衡器，支付宝新增的声波支付功能，美图秀秀中所有修改照片的功能背后其实都是对数字图像的处理。表1-1围绕音频处理的基础功能，将Android与其他的两个平台进行了对比[12]。

表1-1 不同平台对于数字音频的支持

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Functions | Superpowered Audio SDK(Cross Platform) | Core Audio by Apple | OpenSL ES  (Android) |
| Audio Decoder | MP3 | ✓ | ✓ | ✓ |
| AAC | ✓ | ✓ | ✓ |
| WAV | ✓ | ✓ | ✓ |
| AIFF | ✓ | ✓ | ✓ |
| Audio Filters | Low-Pass, High-Pass | ✓ | ✓ |  |
| Low-Shelf, High-Shelf | ✓ | ✓ |  |
| Bandpass | ✓ | ✓ |  |
| Notch | ✓ | ✓ |  |
| Audio Effects | Echo | ✓ | ✓ |  |
| Gate & Roll | ✓ |  |  |
| Time Stretching | ✓ | Low quality |  |
| FFT | Complex | ✓ | ✓ |  |
| Real | ✓ | ✓ |  |
| Audio Quality and Formats | Floating Point 32-bit | ✓ | ✓ | Partially |

## 1.3 可行性分析

### 1.3.1 需求分析

1. 功能需求：采集音频信号，并保存至手机内部存储，实时观测音频的时域或频域波形；对图像进行适当处理，以改变图像的饱和度、色相和亮度，能够将处理的图像保存至手机内部存储
2. 性能指标：要求音频的采样频率为，单声道，以2.5ms~为一帧音频(若音频信号是采样率为，采样的深度为位，每帧数据的时隙间隔为，单通道，则一帧音频数据需要的存储空间为：size = 44100 x 16bit x 0.02s x 1 = 14112 bit = 1764 byte)
3. 可靠性需求：在安静的环境下能够粗略的显示音频的频域谱，对于相对嘈杂的环境有一定抗干扰能力
4. 将来可能的扩展要求：能够增加更多音频图像处理功能，在高性能的前提下保证低电量的消耗

几乎任意一部Android手机都支持录音功能，而Google提供了开放的API(AudioRecorder)允许开发者通过申请相应权限来使用手机的话筒进行录音。为了保证平台的统一性，Google要求所有手机制造商保证至少支持44100Hz、16位深度、单声道的采样，并在采样前进行抗混叠滤波。调用接口得到的数据属于PWM编码后的数据，而WAV无需压缩数据，所以只需要简单的查阅WAV文档，先写好文件头，接下来直接存储采集到的数据即可。

利用Google提供显示动画的API(SurfaceView)接口可以十分便捷、快速地更新屏幕上显示的波形。

本设计采用快速傅里叶变换，可以将采样得到的时域信号转换为频谱。使用平均谱对嘈杂环境有一定的抗干扰能力。

在Android中通常使用Bitmap来存放一幅图像，利用Google提供的ColorMatrix接口可以十分便捷的控制图像的饱和度、色相以及亮度。Java提供了File以及OutputStream等类，可以用来读写文件，这样就可以读取并保存图像。此外，Bitmap类里有用于压缩的compress函数，这样就可以使处理后的图像占用更小的存储空间。

应用设计时考虑到Android的生命周期，只要用户关闭录音程序便杀掉所有线程，以减少电量的损耗。

经过分析以上需求均可满足。

### 1.3.2 运行环境

该应用适用于Android5.0.1以上，无需网络，但手机需要支持录音才能够正常使用。

### 1.3.3 开发工具

1. Eclipse IDE for Java Developers，Version: Mars.2 Release (4.5.2)；
2. JDK,Java(TM) SE Runtime Environment (build 1.8.0\_20-b26)；
3. Android Development Toolkit，Version: 23.0.7.2120684；
4. Android Series Develop Kits，Android5.0.1(API21) SDK Platform；

## 1.4 论文主要研究内容

开发基于Android平台的手机软件，能够对数字音频和数字图像进行分析和处理。数字音频部分主要包括将采集到的音频以多种方式显示到手机屏幕上，并通过做滑动平均，观测信号的变化趋势，还可以通过FFT变换从时域转换到频域，为了观测的需求分别做了FFTShift以及平均谱处理，最后将音频以PWM编码、WAV文件格式保存到手机的内部存储。数字图像部分主要包括：图像饱和度的增强、图像色相以及亮度的改变。论文内容安排如下：

第一章首先介绍了数字信号处理的研究意义以及国内外发展。

第二章介绍了数字音频处理的基本概念以及涉及到的主要分析方法，此章重点为音频信号的采集。

第三章则分别从数据走向、模块设计、系统框架入手，详细的说明了程序音频部分的开发思路。

第四章介绍了数字图像处理的基本概念以及主要处理方法，此章会重点介绍如何利用ColorMatrix实现图片的色饱和度、色相及亮度的处理。

第五章则详细的介绍了图像的处理流程。

第六章对论文全篇进行了总结。

# 2 数字音频处理的基本原理

## 2.1数字音频的基本概念

21世纪人类已经全面进入了数字化时代，数字信号处理已经被广泛地应用到诸多范围，无时无刻的不在影响着人类的日常生活。不同的信号会有不同的表现方式。日常生活中的物理信号有不同的表现形式，常见的表现形式有声、光、电等为了分析处理这些信号，需要使用传感器将这些非电信号转变为电信号来进行处理，比如将麦克风采集到的声音转变为电信号，将数字照相机采集到的物体的光信息转变为二维坐标下的电信号。

在实际中，物理信号转换后的电信号在任意时刻都有数值，并且是连续的，则称该信号为连续时间信号，也叫模拟信号，如正弦信号声音信号及图像信号等，如图2-1所示。由于模拟信号在任意时刻的数值都是连续的，所以并不适合放到计算机中进行处理。

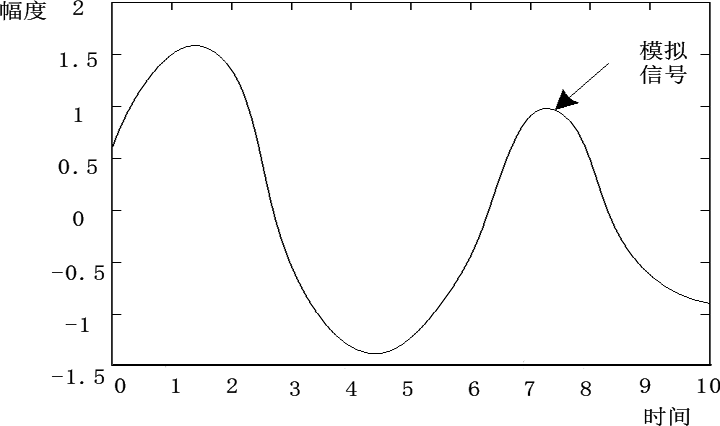


图2-1 模拟信号实例

如果信号在时间轴上是离散的，则称该信号为离散时间信号，例如，x(n) ={0.25，1.3，0 .5，-0.5，-1,-0.48}，其中0.25表示处在零位置的信号取值为0.25，x(n)如图2-2所示。各地区人口统计的数据、股票交易、以及每天的气象预报数据等等都是离散序列。离散信号十分适合放到计算机中进行处理，离散信号是信号处理主要的研究对象。但有时处理的是连续时间信号，这是就要对该信号进行数字化处理，通过采佯、量化、编码将模拟信号转变为数字信号，而这一过程的实现要用到模数转换器(A/D)。



图2-2 离散时间信号实例

采样时需要满足何种条件，才能将采样得到的数字音频信号不失真的还原为采样前的模拟信号才是人们所关心的。下节重点介绍的奈奎斯特采样定理将会解决了这个棘手的问题。

## 2.2 音频信号的采集

模拟信号在使用数字系统处理前，需要先进行采样，这时系统需要A/D转换器和D/A转换器，如图2-3所示。



图2-3 模拟信号数字处理框图

模拟信号首先要经过抗混叠滤波器，它能够滤除高频成分，避免频谱混叠。处理完成后，还要将数字信号还原为模拟信号。模拟信号转换为数字信号转换的第一步是将离散序列转换为与之成比的阶梯信号，阶梯信号需要零阶保持。之后利用低通滤波使阶梯信号变平滑，平滑后的信号即还原为模拟信号。

对模拟信号进行等间隔的采样，其作用可以看作是将模拟信号送入一个周期为Ƭ的矩形脉冲串PT(t) 的电子开关，开关每隔Ƭ秒闭合一次，。开关闭合时间，则变为理想采样，此时脉冲串会成为单位冲激串，它在每个采样点上强度为1，理想采样是将乘以Ƭ为周期的冲激函数，采样公式为

式中是单位冲激信号，当且仅当当时，才可能出现非零值，因此采样信号可表达为

其中Ƭ为采样周期，其倒数1/Ƭ=fs，称为采样频率。采样信号在相应的t=nƬ采样处上，信号的强度准确地等于对模拟信号的采样值。

信号、、的频谱可表达为

式中，,称为采样角频率，单位是弧度/秒(rad/s)。

因此

式(8)表明，采样信号的频谱是由一系列形状相同的原信号的频谱进行周期延拓而产生。幅度为1/Ƭ加权，而相邻两个组成部分的中心频率间会相隔一个采样频率，即延拓周期为采样频率。若是带限信号，其最高截至频率为，如图2-4(a)所示。如果满足，即，基带谱与其周期延拓形成的频谱不发生重叠，如图2-4(b)所示，可以利用低通滤波器滤掉高频成分取出和原始信号完全一样的频谱，否则会造成频谱混叠，如图2-4(c)所示。所以，若是想要将采集到的数字音频信号完整地还原为原始信号，采样时必须满足奈奎斯特采样定理。

若模拟信号的频带是有限的，且位于[0，fc]内，采样频率与信号的最高截止频率满足fs≥2fc，那么由采样信号可以不失真的恢复出原连续信号。这就是柰奎斯特采样定理。

模拟信号经过采样后还不是数字信号，还需要经过量化和编码。由采样信号 可以得到一串采样点上的样本数据，这一串样本数据可看为时域离散信号序列，用表示，即

如图2-5表示了由模拟信号经过采样后变为离散信号序列。



图2-4 采样信号的频谱



图2-5 模拟信号采样转换的时域离散信号序列

例如：模拟信号，其中f=50Hz，选采样频率fs=200Hz，将t=nƬ代入中，得到离散时间序列：

当n=…，0,1,2,3,…时，得到序列如下

通过 A/D 转换，将模拟信号转换为离散时间信号。准确地说，离散事件信号还不是数字信号，要是计算机能够对信号直接进行处理，还将离散时间序列进行量化和编码。以式(10)为例，按照6位二进制码进行量化编码（包含以为符号位），则得到数字信号如下

用十进制表示的为

显然量化编码后的与原来的不同。这样所产生的误差称为量化误差，可以通过增加量化比特数来降低这种误差，使用N bit，计算机可以表示个可能的值。比特数越大，数字信号所代表的数值与模拟信号真实数值便会越靠近，但在计算机处理时的所需的时间也会随之增加。

图2-6 演示了3位模拟信号的采样、量化及量化误差间的关系。



图2-6 三比特A/D转换及串行数字比特流

一般地说，用离散时域序列表示数字信号更好，因为直观的反映了信号的增减变化，而编码后的数字信号则不能。因此，在对数字信号分析时大多采用离散时域序列进行分析。在不混淆的睛况下，我们也将离散时间序列称为数字信号。

对于数字序列，一个重要的概念就是数字频率。如果是由一个周期为的模拟正弦信号采样来的，其模拟角频率弧度/秒(rad／s)，设采样时间为Ƭ，则

令，则称为序列的数字频率，单位为弧度。设为采样频率， ，则

说明数字频率是模拟角频率对采样频率的归一化。

## 2.3 涉及到的主要分析方法

### 2.3.1时域分析

时域分析：不经过任何处理直接显示时域波形，如图2-7所示，在有嘈杂的环境下很难辩认出原始信号。为了观测信号的变化趋势，通常需要对观测到的信号进行滑动平均。一个四项滑动平均的差分方程可以表示为

经过滑动平均消除背景噪声的干扰后的时域波形，如图2-8所示，噪声基本消失，可以真实地看到原始信号的变化趋势，但如果原始信号的频率过高也会造成很大的信息损失。

### 2.3.2频域分析

频域分析：时域波形只能传达很少的信息到人类的眼里，所以要经过快速傅里叶变换得到频谱图，如图2-9所示，在安静的环境下可以粗略地观测信号不同频率分量在频谱的分布，但是频谱图比较杂乱，没有达到理想的效果。经过fftshift坐标转换后的频谱，如图2-10所示，直接经过FFT变换所产生的频谱，由于离散傅里叶变换的对称性，观测到的频谱并不能真实反映真正的频率位置，最高频率位于中间，零频位于两侧，经过位置左右交换后的频谱图中间显示为零频直流分量，而两侧是最高频率，最高应为采样频率的一半。

不幸的是，这看起来也像噪声一样混乱。假设采样个数为256，因为这256个点并没有提供足够的信息来获得一个表现好的曲线。使用更长的FFT也不能解决这个问题。如果使用2048个点的做FFT变换，频谱将会产生1025个点。即使原始的2048个点包含更多的信息，在频谱中出现更多的样本也只会由于同样的因素而稀释了有用信息。更长点的FFT变换只会获得更好的频谱分辨率，但得到的同样是噪音水平的。

答案就是使用更长的原始信号但却不增加频谱的点数。可以将输入信号截断分成许多由256个点组成的片段来实现。每个片段都乘以汉明窗，并进行256点的DFT，然后转换到极坐标系下。由此产生的频谱之后会被平均来形成一个129点的频谱。如图2-11展示了一个平均8次后的频谱，用来代替图2-10。图2-12是FFT平均谱优化图像。平均谱的增强是显而易见的；噪声被降低到了一个水平，在这种水平下允许观测者仅观测对信号感兴趣的特征。只有频域的幅值大小才会以这种方式做平均；频谱的相位通常会被丢弃，因为它并不包含有用的信息。随机噪声的降低与分成片段的数目的平方根成比例。而100个片段是典型的，有些应用可能会平均数百万个片段来产生若特征。

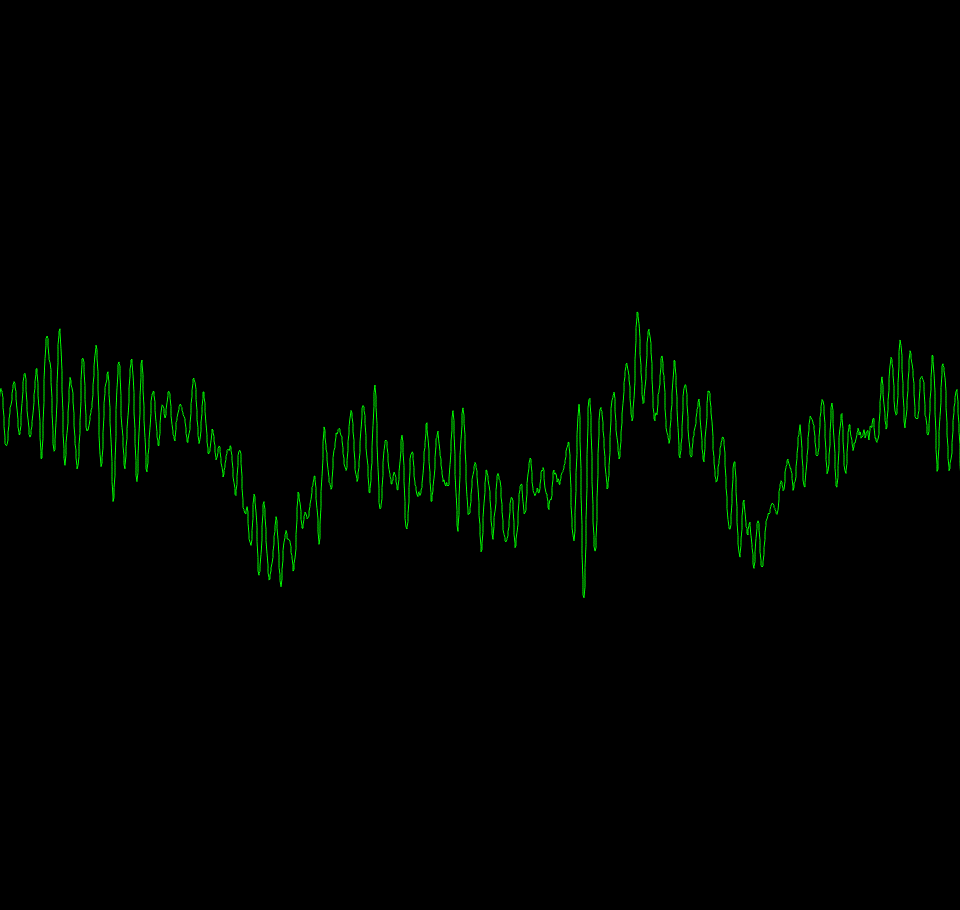


图2-7 时域波形

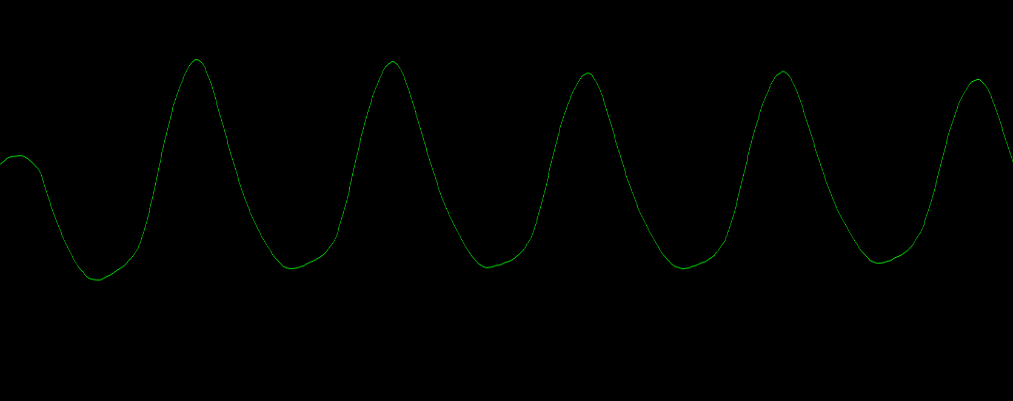


图2-8 时域滑动平均

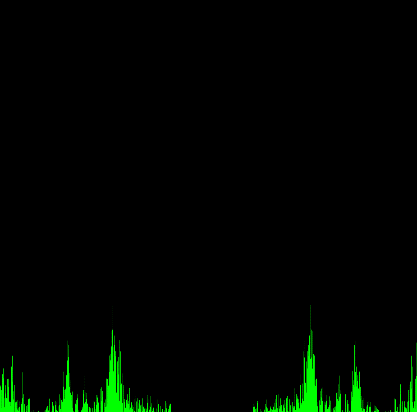


图2-19 FFT频谱

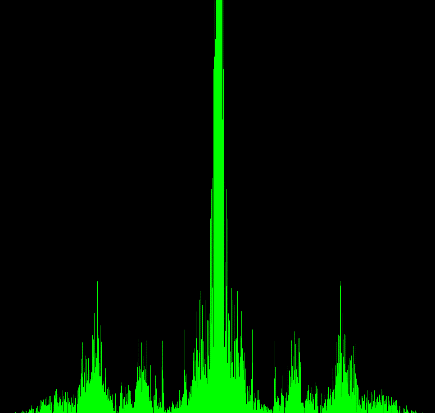


图2-10 FFTshift频谱

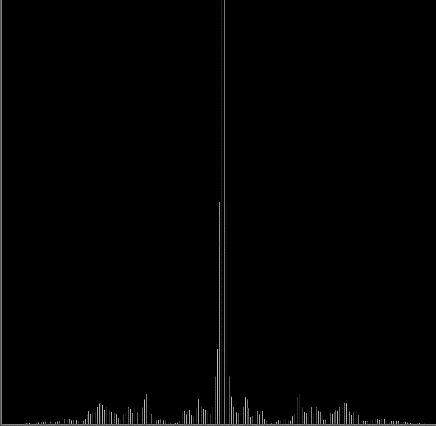
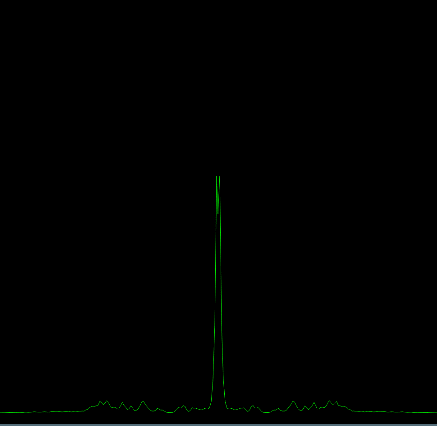
 

图2-11 FFT平均谱 图2-12 FFT平均谱优化图像

## 2.4 本章小结

本章主要介绍了数字音频的基本概念、音频信号的采集以及涉及到的主要分析方法三方面内容，着重介绍了音频的采集。本论文所设计的数字音频分析的APP就是以此为理论基础进行的。

# 3 数字音频分析的APP设计

## 3.1 系统软件设计

### 3.1.1 音频的数据走向

采集到的数据首先将会存放至缓存区，程序中需要通过Android提供的接口及时取出数据，取出的音频数据是以short型的数组存在，之后将采集到的音频数据分为两种情况分发，因为采集到的数据是16位的，而存储或绘制需要不同类型的数据，所以需要根据不同的需求将数据转换为相应的数据类型。

当用户没有打开数字信号处理功能时，会直接将数据分发给存储模块和绘图模块，存放模块需要byte型数组，byte占一个字节short占两个字节，均为有符号类型，经过转型的数组长度为原来的两倍，将采集到的short型数组分为低8位和高8位，short型数组的一个元素将会分别放置在byte型数组相邻的两个位置，数据的流向、使用的函数以及对应的数据类型如图3-1所示



图3-1 不经过DSP的数据走向

当用户打开数字信号处理功能后，首先将采集到的数据通过深表复制将数据传给数字信号处理模块，为了保证数据的精确度，处理前需要把数据转换为double型数组，然后再根据用户所选择的处理类型进行相应的处理，最后通过另一个分发通道将数据通过深表复制和数据类型转换分发至存储模块和绘图模块。如图3-2详细的标注出数据的走向，使用的函数以及对应的数据类型。



图3-2 经过DSP的数据走向

### 3.1.2 绘制波形

为了满足使用者观测的需要，在绘制波形时会按照用户的设置实时修改波形的参数以及绘制的风格。波形参数主要用于调整幅度和一帧数据的长度，通过适当地手动调整，可以使波形恰当的显示在屏幕上，以便观测者观测感兴趣的部分，或解决由于音源或噪声过大而导致波形超出屏幕范围的情况。使用者可以选择一种自己喜欢的绘制风格来绘制波形，其中包括块状图、柱状图以及曲线图。块状图会显示出所有检测到的信息，但不适合在噪音过大的环境下使用，由于绘制的范围较大，所以会影响手机的性能帧率会降低；柱状图就像大部分有显示波形功能的播放器一样，仅仅显示主要的信息，也可能会丢失主要信息，但是却十分美观，尤其是在欣赏音乐这种情况下；而曲线图将会把每个数据连接在一起，这里看到的波形和常见的示波器显示的波形是一致的，适合于时域的显示。

最强大的功能是用户不仅可以观测时域波形，还可以选择显示音频信号的频谱。这里有三种方法供用户选择，包括一般快速傅里叶变换(FFT)、FFTshift以及平均谱。FFT变换后所显示的波形往往是对称的，而采集的音频的最高频率也只有采样频率的一半，应位于波形的中央，一般幅度最低，而两侧的高频区幅度往往最高。用户如果需要看到和实际频率所匹配的波形，需要调整数据宽度x，使屏幕仅显示左半侧，或者使用FFTshift将零频移到屏幕的中央。可惜的是这两只方法并不能够显示足够多的信息，且容易受到噪声的干扰，所以这里增加了一个强大的功能，就是平均谱。打开平均谱后，虽然会降低谱分辨率，但用户将会观测到信号真实的频谱分布，并能够抵消一部分噪声的干扰，对于系统的性能并不会有什么影响。将平均谱与曲线图结合是观测频谱的最佳组合。

图3-3 波形绘制流程图

## 3.2 音频处理系统架构

### 3.2.1 项目结构

项目工程结构如图3-4、3-5、3-6所示，src用于放Java原码，res用于放资源文件，包括布局layout、图片drawable、菜单menu、音乐raw、字符串资源values等。src目录下又分为android、audio、image三个子文件夹，它们分别用来存放活动窗口activity、音频处理audioProcess、以及图像处理imageProcess。audio下面分为analysis和display，分别用于分析音频以及显示波形。AndroidManifest.xml是Android应用的清单，可以用于向手机申请使用权限，并注册配置Android组件。

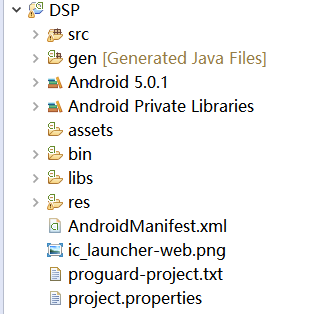


图3-4 项目结构总图

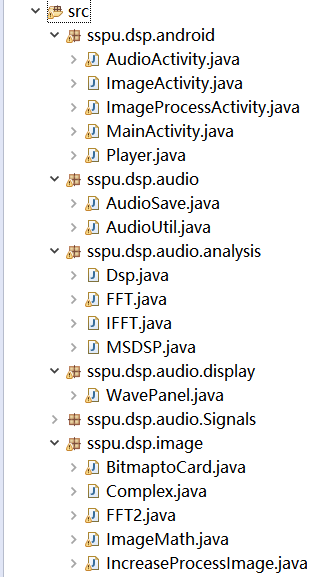


图3-5 项目结构源码图

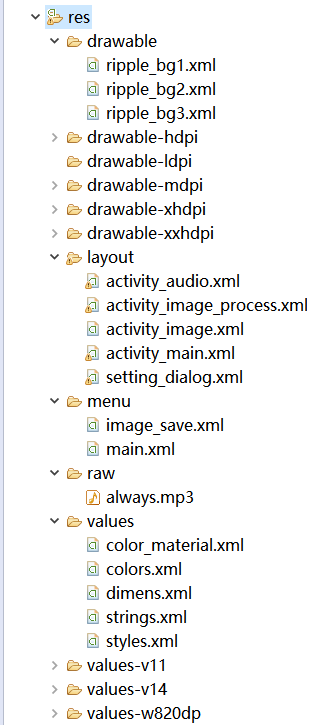


图3-6 项目结构资源图

### 3.2.2 系统架构

Android系统的底层是Linux核心，Android平台由操作系统、中间件、用户界面和应用软件4层组成，它采用软件层叠(Software Stack)的方式进行构建。这种叠层结构分隔了层与层，使得各层分工明确。这种分工降低了层与层之间的耦合，当某层的下层或更下层发生变化时，上层不需要做任何改变。图3-7显示了Android系统的系统结构。

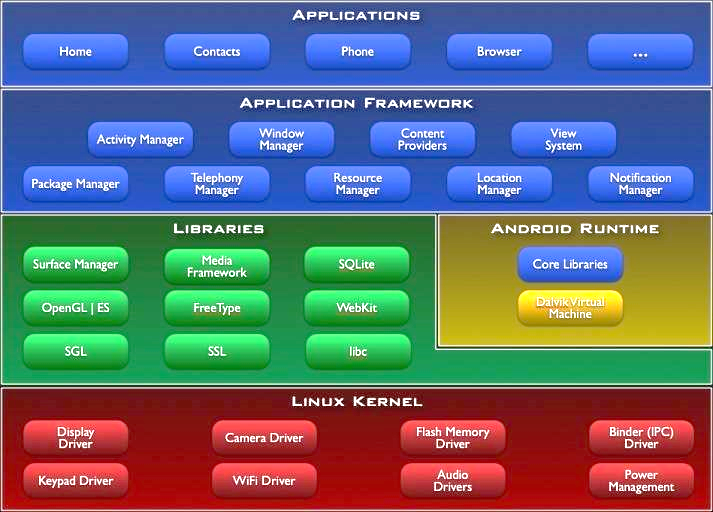


图3-7 Android系统体系结构

系统框图分为四层从下到上分别为音频采集模块、数字信号处理基础算法实现模块、音频工具集合、绘制模板、音频活动，见表3-1。音频采集模块使用Android开放的API实现录音，数字信号处理基础算法实现模块依据信号与系统理论以及Matlab算法，完全使用Java实现。音频工具集合调用了基础算法实现模块，为上层的活动提供接口，使得Android程序与Java算法模块解耦。音频活动属于Android的Activity组件，作为音频程序的入口，负责与用户交互，完成界面上的响应，将用户的操作传递给对应的模块进行处理，最后在反应到屏幕上的波形显示上。

表3-1 系统结框图

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AudioActivity | | | | | |
| AudioUtil | | | | | WavePanel |
| Dsp | MSDSP | FFT | IFFT | AudioSave |
| AudioRecorder | | | | | |

### 3.2.3 系统进程视图

为了使系统符合满足功能和性能上的需求，将数据的采集以及分发设计为并行模式，使用多线程来实现，为了解决线程安全问题，程序中使用了Java关键字synchronized进行对象同步，避免多个线程同时操作同一个数据造成数据错误的问题。由于有多个线程同时运行，可能导致存储模块没有及时得到采样的数据而造成音频的不连贯，所以在程序中加入链表，保证音频数据的连贯性，一旦获取到数据会及时将链表中的数据清空，防止内存泄漏。当手机的内存占用较大时，不建议同时开启绘制与存储，这样可能会造成声音不流畅。系统的进程如图3-8。



图3-8 音频系统进程图

### 3.2.4 系统界面设计

音频分析功能用户可能会时常用到，而在某些情况下为了及时捕捉信号，应用需要提供快速入口，来迅速打开“音频处理及分析”窗口，为了满足用户的这种需求，本应用将入口图标设计为双图标模式，分别用于打开“主菜单”以及“音频处理及分析”界面。如图3-8。



图3-9 双图标设计

为了用户得到更好的体验，应用的每个界面均采用Android5.0(lollipop)以来设计体验最佳的主题风格——Material。应用该主题界面上凡是可以获取触摸焦点的位置均会出现类似水波纹(ripple)扩散的效果，如图3-10、3-11，顶部的ActionBar位于图层最上层，有阴影渲染，并被设计为卡片风格。ActionBar可以提示用户所处界面的功能，同时提供返回主界面的快捷入口，并将常用设置放到这里，减少了屏幕的占用，增加了主要内容的占屏比重。如图3-12、3-13、3-14、3-15。音频处理界面增加了屏幕旋转功能，横屏观看时户将会获得更大的视野。

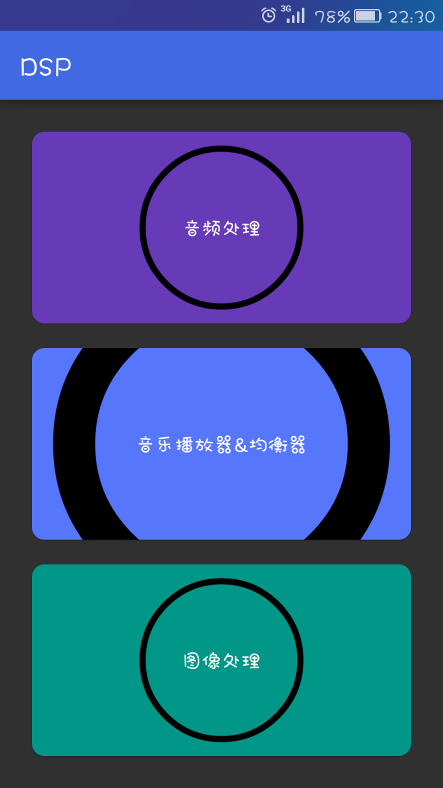


图3-10 主界面



图3-11 水波纹效果

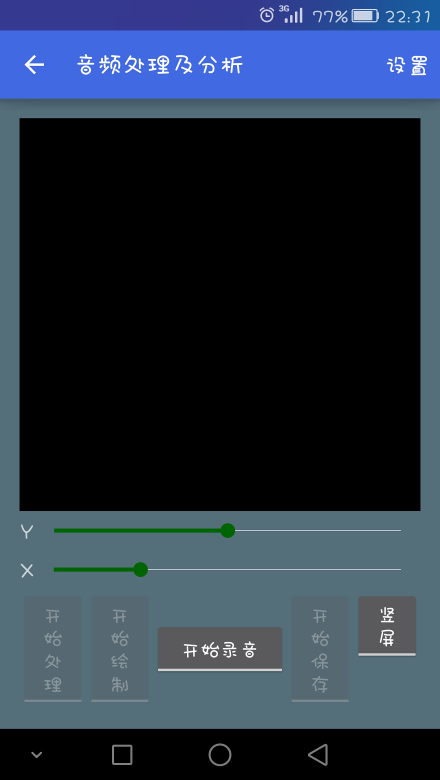


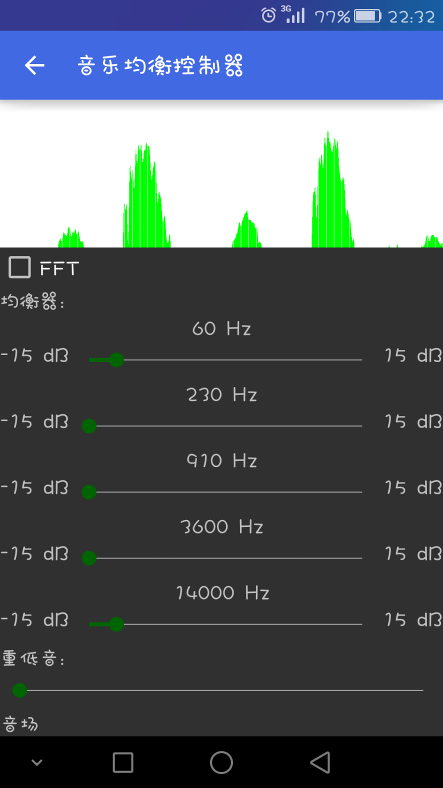
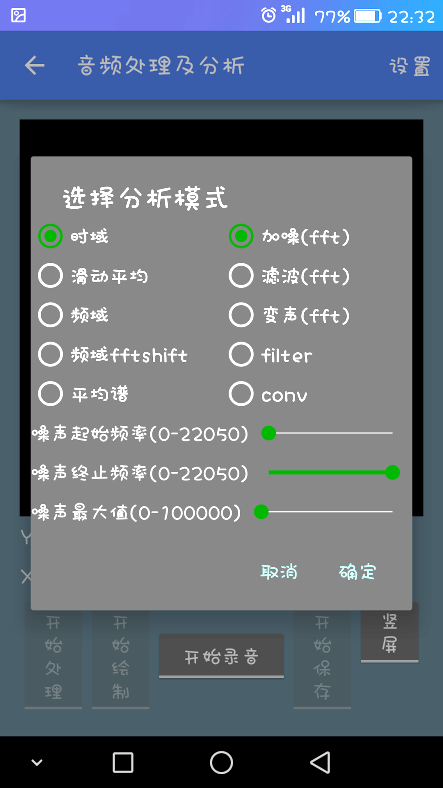
图3-12 音频分析处理界面(Portrait) 

图3-13 音乐均衡控制器



13-14 设置对话框1



图3-15 设置对话框2

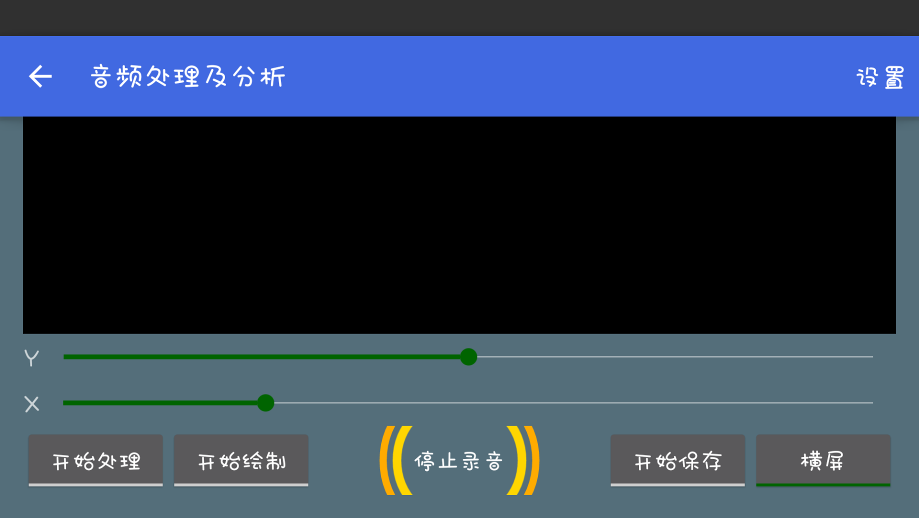


图3-16 音频分析处理界面(Landscape)

## 3.3 音频分析结果

打开数字音频和图像分析处理APP，使用音频处理模块，可以实现波形的实时预览，拖动下方的滑动条来调整波形的高度和宽度，放大波形中最感兴趣的那部分。同时还实现了将录制的音频以WAV的文件格式保存至内部存储，以便回放音频。单击绘制面板波形将会以三种不同的方式显示。单击顶部的设置菜单，可以弹出设置分析类型的对话框，在里面用户可以分别设置不同的分析方法来观测波形。退回到主界面我进入音乐播放界面，将会循环播放一段优美的音乐，同时用户还可以使用下方的均衡控制器来调整音乐不同频段的增益，例如用户可以实现消除人声，放大重低音等。在欣赏音乐的同时用户还可以像上一个模块一样观看到正在播放音乐的波形。

## 3.4 本章小结

本章分为四个小节，前两节侧重介绍音频的软件设计和系统的结构，第三节简要的介绍了音频分析结果。通过对程序总设计的介绍，可以更清晰的了解数字信号在实际中是如何搭建框架的；通过对个个模块的详细介绍，可以更加深入的理解数据是怎样获取的，经过什么处理，需要达到什么效果，以及最终是以什么形式输出的。

# 4 数字图像的基本原理

## 4.1 数字图像的基本概念

除了用于描述音频的一维离散序列外外，还可以用二维序列来描述图像。二维离散序列即矩阵 (matrix)，矩阵中的每个值都有对应的像素点(pixel)，它记录了该处图像上的颜色信息。在二值黑白图像，矩阵则表示一个区域内像素的灰度级(gray scale level)。对于一幅量化值为8 *bit*的黑白图像，可以表示个灰度级，每个灰都级表示0（黑）到255（白）中间的某个灰度。灰度图像能够产生形状，如4-1所示。16x16像素的图像，每个像素占8 *bit*，该图像的大小为16\*16\*8bit(或16xl6x256)。图4-2中的矩阵列出了图4-1中的每个像素的灰度值，其中较低的数值对应图4-1的暗处。而彩色图像每个像素用三个参数描述，包括红、绿、蓝三种色彩。



图4-1 16\*16\*256数字灰度图

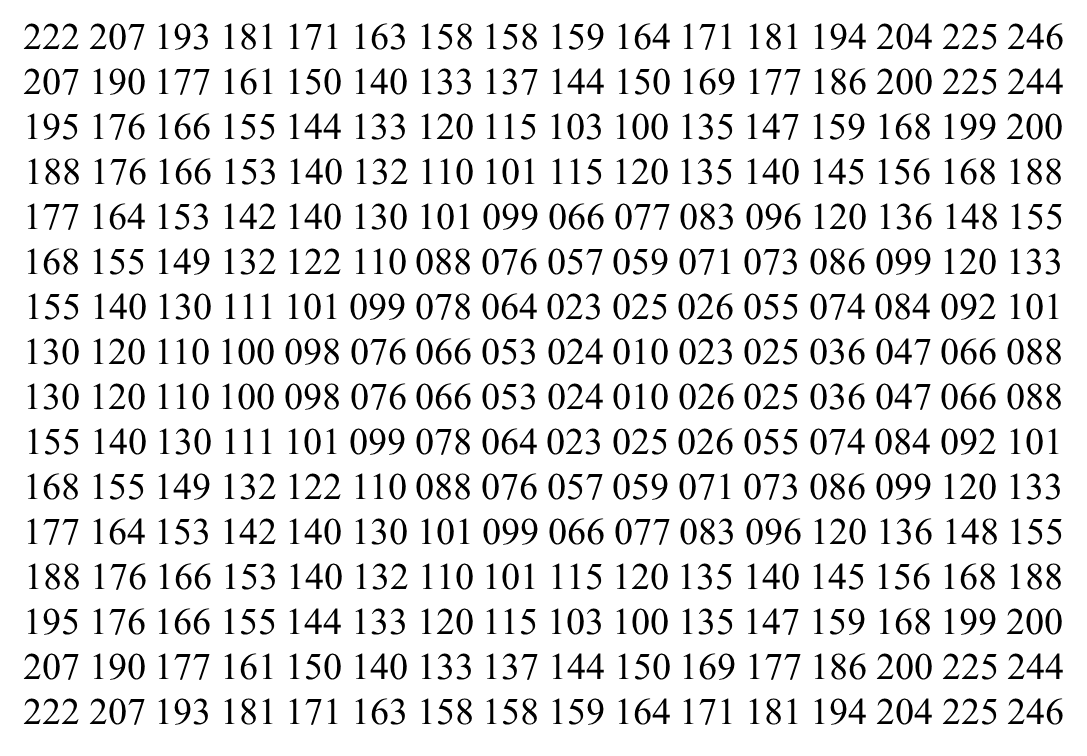


图4-2 数字图像灰度值

## 4.2 涉及的主要处理方法

数字图像处理的目的是为了达到某种期望的效果，它主要在计算机上完成处理，因此也可以成为计算机图像处理。数字图像处理可以分为空域处理法和变换域处理法。空域处理就是在空间内对图像之间进行处理，往下又分为领域处理法以及点处理法。变换域处理法包括傅里叶变换、离散余弦变换、以及小波变换等。通过以上某种变换，可以将图像的空域矩阵转换为变换域矩阵，之后在变换域中完成图像的相关处理。处理完变换域矩阵后，还要通过对应的反变换将变换域矩阵转换为空域矩阵，这样就得到了处理后的图像。数组图像处理的主要研究图像压缩、编码、复原等。数字图像处理的目的是按照人们的需求，将图像中感兴趣的部分进行突出，并减弱或去掉不感兴趣的部分，从而使有用信息。

在Android程序中，通常通过Bitmap来存放图像的颜色信息，使用ColorMatrix及Matrix来实现一幅图像的颜色变换和空间变换，例如图像增强、复原、提高饱和度以及裁剪、旋转等效果。

如式(1)所示M是一个5\*4的颜色矩阵。在Android中， M是以一维数组进行存放的。

(1)

一幅图像通过红、绿色、蓝色、透明度三个参数共同表示最终的显示效果。rgba放在5\*1的颜色向量Color中，通过改变Color就可以控制图像的颜色效果。颜色向量Color如式(2)。

(2)

若要修改图像的颜色，只需要修改该图像颜色向量。假设改变后的颜色向量为，则如式(3)所示的颜色向量计算公式。

(3)

从式(3)可以看出，矩阵K的第一行可以改变图像的红色分量，第二行可以改变图像的绿色分量，第三行可以改变图像的蓝色分量，第四行可以改变图像的透明度，第五列则可以改变图像的颜色偏移量。

矩阵K的第五列一般用来修改图像的颜色成分。从式(4)可以看出矩阵M1可以使图像的红色和绿色分量均增加100，这样处理效果会使得图像看起来偏黄些。

*M1=* (4)

此外，将颜色数值乘以某个系数也可以达到改变颜色分量的效果。如图(5)所示的矩阵M2，绿色分量变成原先的2倍，这样图像看起来就绿些。

M2= (5)

## 4.3 本章小结

本章主要介绍了数字图像的基本原理，包括是数字图像是什么，怎样在计算机里描述一幅图像，数字图像处理是什么，基本的处理方法有哪些，以及在Android中如何通过ColorMatrix修改一幅图像。本论文所设计的数字图像处理的APP就是以此为理论基础进行的。

5 数字图像处理的APP设计

## 5.1 软件设计

上一章我们介绍了颜色矩阵的工作原理，现在我们就可以使用颜色矩阵对图片进行处理了。数字图像处理的基本流程图如图5-1。



图5-1 图像处理流程图

## 5.2图像处理结果

打开数字音频及图像处理APP的最后一个模块，选择一张想要修改的图片，进入处理界面后，拖动滑块修改图像的饱和度、色相以及亮度。用户在调整相关参数后可以选择不同的格式进行压缩，包括jpg、png、webp三种格式。jpg格式对于多色彩的还原比较真实；而png格式更适合矢量图形，如果你的图片背景是透明的可以选择这种格式；webp是谷歌定义的一种格式，同样的一幅图片使用这种压缩方式在保证图像质量的同时，会占用更小的存储空间，用于网页时可以加快图片加载速度。图像修改色相前后的对比如图5-2所示。图像处理后偏粉色系,比原图更加赏心悦目。

图5-2 图片处理前后效果对比

## 5.3 本章小结

本章介绍了数字图像处理的APP设计，实现了第四章中介绍的部分处理方法。最后还展示了图片处理前后的效果对比图。

# 6 结论

本论文设计并开发完成了一款基于Android平台的数字音频和数字图像分析处理APP。数字音频及图像分析处理APP设计为三个模块，分别为音频分析模块、音频处理模块、以及图像处理模块。

使用音频分析模块，可以实现波形的实时预览，拖动下方的滑动条来调整波形的高度和宽度，放大波形中最感兴趣的那部分。同时还实现了将录制的音频以WAV的文件格式保存至内部存储，以便回放音频。单击绘制面板波形将会以三种不同的方式显示。单击顶部的设置菜单，可以弹出设置分析类型的对话框，在里面用户可以分别设置不同的分析方法来观测波形。

使用音频处理模块用户将进入音乐播放界面，此时会循环播放一段音乐，同时用户还可以使用下方的均衡控制器来调整音乐不同频段的增益，例如用户可以实现消除人声，放大重低音等。在欣赏音乐的同时用户还可以像上一个模块一样观看到正在播放音乐的波形。

使用图像处理模块，用户需要先选择一张想要修改的图片，进入处理界面后，拖动滑块修改图像的饱和度、色相以及亮度。单击活动条右上方的保存按钮，选择一种压缩方式，单击确认即可将修改的图片保存至手机内部存储。

虽然取得了较为满意的结果，但仍有很多功能可以增加和提升，如：测量声音的最大分贝值，以模拟电表的形式显示当前测量的情况，并能够将分贝值记录到图标里显示等等、性能加倍提升、电量更加节省、界面更加友好、以及达最佳的用户体验。

# 致谢

在此我要感谢指导老师胡金艳的帮助与关心，在您的悉心指导下我才能够顺利完成毕业设计，并不断地完善它，同时也向李刚、李兴华、郑君里、蒋文蓉、 颜文英、孙秋冬等众多老师及同学给予的帮助表示感谢，你们不仅在我身边鼓励我，还帮助我解决了许多实际问题，让我学到了许多有用的知识。谢谢你们的关心，我会加倍努力。

# 参考文献

1. 方勇.数字信号处理：原理与实践(第2版) [M]. 北京:清华大学出版社,2010.8.
2. 丁玉美,高西全.数字信号处理(第2版).西安:西安电子科技大学出版社,2001.
3. 李兴华.Java开发实战经典[M]. 北京:清华大学出版社,2009.8.
4. 张秀丽, 王冠, 张巧文等. 滑动窗口FFT实时频域虚拟低音增强算法[J]. 价值工程,2016,(1):110～111.
5. 谭子尤，张雅彬. 离散傅里叶变换快速算法的研究与MATLAB算法实现 [J]. 中国科技信息,2006,(22):316～321.
6. 胡航.语音信号处理（第4版）[M]．哈尔滨：哈尔滨工业大学出版社，2007．
7. 张雪英.数字语音处理及Matlab仿真[M]. 北京：电子工业出版社，2012.
8. 阮秋埼.数字图像处理（第3版）[M]. 北京：电子工业出版社，2011.
9. Rafael C.Gonzalez等著，阮秋埼译. 数字图像处理的MATLAB实现（第2版）[M]. 北京：清华大学出版社，2013.
10. 龚声蓉等著.数字图像处理与分析（第2版）[M]. 北京：清华大学出版社，2014.
11. 宋知用.MATLAB在语言信号分析与合成中的应用 [M]. 北京：北京航空航天大学出版社，2013.
12. 李刚.疯狂Android讲义（第2版）[M]. 北京：电子工业出版社，2013.
13. 郭霖.第一行代码Android [M]. 北京：人民邮电出版社，2014.
14. 蒋文蓉.Android手机编程.-上海第二工业大学计算机与信息学院，2014.1
15. Steven W. Smith.The Scientist and Enginner’s Guide to Digital Signal Processing[M]. California Technical Pub, 1997.
16. Oppenheim A V,Schafer RW.Discrete-Time Signal Processing[M]. Gelwood Cliffs, Nj:Prentice-Hall,1975.
17. Bellanger M.Digital Processing of Signals (Theory and Practice)[M]. New York: John Wiley & Sons,1984.

# 附录

## FFT源码

**public** **class** FFT {

**int** N, m;// m表示N以2的多少m次幂,m为次数N为幂,2为底数

**double**[] cos, sin;

/\*\* N必须为2的幂 \*/

**public** FFT(**int** N) {

**this**.N = N;

**this**.m = (**int**) (Math.*log*(N) / Math.*log*(2));

**if** (N != (1 << m))

**throw** **new** RuntimeException("FFT length must be power of 2");

// 预先计算运算流图的旋转因子cos,sin分别表示旋转因子的实部与虚部

cos = **new** **double**[N / 2];

sin = **new** **double**[N / 2];

**for** (**int** i = 0; i < N / 2; i++) {

cos[i] = Math.*cos*(-2 \* Math.***PI*** \* i / N);

sin[i] = Math.*sin*(-2 \* Math.***PI*** \* i / N);

}

}

/\*\*

\* 实现快速傅里叶变换

\* **@param** x 输入数据的实部

\* **@param** y 输入数据的虚部

\* **@return** 输出X(k)的实部虚部将会分别放在x,y里

\*/

**public** **void** fft(**double**[] x, **double**[] y) {

**int** i, j, k, n1, n2, a; **double** c, s, e, t1, t2;

j = 0; //把二进制位倒序排列如001->100,110->011

n2 = N / 2;

**for** (i = 1; i < N - 1; i++) {

n1 = n2;

**while** (j >= n1) {

j = j - n1;

n1 = n1 / 2;

}

j = j + n1;

**if** (i < j) {

t1 = x[i]; x[i] = x[j];

x[j] = t1; t1 = y[i];

y[i] = y[j]; y[j] = t1;

}

}

// FFT

n1 = 0;

n2 = 1;

**for** (i = 0; i < m; i++) { // m=蝶形运算的级数

n1 = n2; n2 = n2 + n2;

a = 0;

**for** (j = 0; j < n1; j++) {

c = cos[a]; s = sin[a];

a += 1 << (m - i - 1);

**for** (k = j; k < N; k = k + n2) {

t1 = c \* x[k + n1] - s \* y[k + n1];

t2 = s \* x[k + n1] + c \* y[k + n1];

x[k + n1] = x[k] - t1;

y[k + n1] = y[k] - t2;

x[k] = x[k] + t1;

y[k] = y[k] + t2;

}

}

}

}

/\*\*

\* 计算幅度

\* **@param** x 实部

\* **@param** y 虚部

\* **@return** 输出结果直接覆盖原来的实部

\*/

**public** **void** abs(**double**[] x, **double**[] y) {

**for** (**int** i = 0; i < x.length; i++) {

x[i] = Math.*sqrt*(x[i] \* x[i] + y[i] \* y[i]);

}

}

**public** **double**[] getValue(**double**[] x, **double**[] y) {

**double**[] temp = **new** **double**[x.length];

**for** (**int** i = 0; i < x.length; i++) {

temp[i] = Math.*sqrt*(x[i] \* x[i] + y[i] \* y[i]);

}

**return** temp;

}

}