# 基于FFT的音频可视化及图像压缩

## 摘要

本文基于数字信号处理课程上学习到的知识，将快速傅里叶变换算法用于实际应用中的音频可视化及图像压缩。通过matlab中的fft2和ifft2实现了离散傅里叶变换并通过低通滤波的方式去除高频信息，以此来压缩图像。并且在Android Studio通过编写本地快速傅里叶变换算法，实现了音频信号转频谱图（音乐可视化）以及图片经过二维离散傅里叶变换转到频率域的幅度谱，再通过裁切幅度谱并经过傅立叶逆变换的方式来压缩图片,最后将以上功能封装成手机APP。matlab及Android Studio代码已在Github上开源（[WPENGxs/DSP: My DSP homework (github.com)](https://github.com/WPENGxs/DSP)）。

**关键词：**快速傅里叶变换；信号处理；安卓；图片压缩；

## 引言

快速傅里叶变换算法（FFT），是快速计算序列DFT或其逆变换的方法，早在1895年该算法就被推导出来，但该算法的基本思想直到1965年才得到普及。该算法被广泛应用与工程、科学和数学领域，同时也被IEEE科学与工程计算期刊列入20世纪十大算法。FFT算法的基本思想是以分治法为策略并通过递归的方式将长度为N的离散傅里叶变换分解为长度为A的B个较短序列的离散傅里叶变换（N=AB），该思想将算法的时间复杂度变为，极大的提升了计算的速度，可以说这个算法改变了世界。而在本论文中，我将使用FFT来实现音频可视化（将音频信号的函数转化为频谱图），并将一张二维图片通过裁切幅度谱的方式去除高频信息，达到压缩图片的目的。同时以上功能全部集成在手机APP中，可以在手机本地运行这一伟大的FFT算法。

### 一、matlab实现压缩图片

在matlab上实现FFT和IFFT极为方便，只需要调用matlab官方集成好的fft2和ifft2函数即可。在对图片进行FFT后将数据通过低通滤波后在进行IFFT即可达到压缩图片的效果。在matlab上实现效果图如下：

图形用户界面, QR 代码

描述已自动生成

在该过程中使用的均为matlab提供的函数，故不特别说明。

### 二、Android实现音频可视化

在Android Studio实现音频可视化，需要将音频信号实时读取，并计算出当前信号的频谱图，然后将频谱图实时更新即可得到可视化的音频。在这里我使用了Android的开源框架：XXPermissions以及AudioVisualizeView。其中XXPermixxions为安卓权限请求框架，由于音频信号的获取需要手机的录音访问权限和手机存储权限，故在此使用这个框架集成权限请求。AudioVisualizeView为音频展示框架，在此使用此框架来读取音频并通过FFT计算出我们需要的图形，然后在手机视图中展示给用户。核心代码如下：

1. AudioVisualize.doPlay(R.raw.sound);
2. Toast.makeText(AudioActivity.this,"Play raw sound(default)",Toast.LENGTH\_SHORT).show();

由于时间的限制，在这里只做了播放固定音乐，没有让用户选择播放音频。软件实机运行效果图如下：

文本

描述已自动生成

### 三、Android实现基于FFT的图片压缩

与上文的matlab实现思路相同，手机上基于FFT的图片压缩也是将图片先通过FFT计算出频率域的幅度谱，在通过裁切幅度谱的方式消除高频信号，之后将数据通过IFFT还原成图片。在手机上实现的难点在于安卓不会像matlab那样自带FFT与IFFT的集成算法，这里的算法需要我们自己通过java实现。这里我在Android Studio中通过创建了三个类：Complex类、FFT类和FourierTransformer类来实现该算法。Complex类为复数类，用于实数与复数在java中的实现。FFT类中通过Complex类实现快速傅里叶变换和快速傅里叶逆变换的算法。而FourierTransformer类集成图片压缩算法，输入参数为Bitmap，通过Bitmap的getPixels()方法获取每一个像素的值，以此来做傅里叶变换和逆变换，最后将结果汇总并重新生成一张Bitmap。实现的核心代码如下：

1. Toast.makeText(ImageActivity.this,"In the calculation...",Toast.LENGTH\_SHORT).show();
2. FourierTransformer fourierTransformer=new FourierTransformer();
3. Bitmap co\_bitmap=fourierTransformer.convert(bitmap);
4. Bitmap af\_bitmap=fourierTransformer.recover(bitmap,false);

实机运行图片如下：

手机屏幕截图

描述已自动生成

其中从上到下第一张图片为用户选择的原图，第二张图片为图片的幅度谱展示，第三张为压缩后的图片展示。右侧的文字显示了两张图片的详细信息。可以看出这张图片在压缩前为4194304字节，压缩后为2097152字节，压缩效率良好且图片无明显失真。

## 实验结果与分析

本次使用FFT实现了两个功能：音频可视化和图片压缩。音频可视化在安卓平台上实现，图片压缩在matlab和安卓上均有实现。总体来说实现效果好，不仅深入了解了FFT及IFFT算法，同时将算法带入到实际应用中。

## 结论

FFT在生活中有很多应用，本次实验仅选取了其中两种作为展示。音频可视化在许多音乐软件中也可以见到，算是FFT在我们身边最容易见到的应用。而图片压缩算法，在安卓上有比使用FFT压缩率更高的算法，因此使用FFT压缩图片也仅仅是展示FFT的应用，而FFT作为信号处理的算法，对于图片更多的应用还是降噪处理。

## 参考文献

[1]於玺.快速傅里叶变换在信号处理中的应用[J].信息记录材料,2021,22(10):184-186.DOI:10.16009/j.cnki.cn13-1295/tq.2021.10.088.

[2] M. Heideman, D. Johnson and C. Burrus, "Gauss and the history of the fast fourier transform," in IEEE ASSP Magazine, vol. 1, no. 4, pp. 14-21, October 1984, doi: 10.1109/MASSP.1984.1162257.

[3] J. Dongarra and F. Sullivan, "Guest Editors Introduction to the top 10 algorithms," in Computing in Science & Engineering, vol. 2, no. 1, pp. 22-23, Jan.-Feb. 2000, doi: 10.1109/MCISE.2000.814652.

[4] T. Mantoro and F. Alfiah, "Comparison methods of DCT, DWT and FFT techniques approach on lossy image compression," 2017 International Conference on Computing, Engineering, and Design (ICCED), 2017, pp. 1-4, doi: 10.1109/CED.2017.8308126.

[5] Jorge L. Vago, Herman C. Vermeulen, Antonio Verga, "Fast Fourier transform based image compression algorithm optimized for speckle interferometer measurements," Opt. Eng. 36(11) (1 November 1997) https://doi.org/10.1117/1.601542