**成绩评定：**

**教师签名：**

****

课程设计报告书

**题目： 声波绘影**

**学 院 电子与信息学院**

**专业班级 2023级信息工程创新班**

**小组成员 孙艺、马梓豪、李昊峻**

**指导教师 杨俊美**

**起始日期 2025.4.1**

|  |  |
| --- | --- |
| 教  师  评  语 | 教师签名：  日期： |
| 成  绩  评  定 |  |
| 备  注 |  |

1. **项目内容摘要**

这份项目报告详细介绍了“声波绘影”项目，该项目旨在通过软件实现音频信号的实时频谱可视化，将声音转化为动态频谱视频，从而形象化、具象化声音内容。报告阐述了项目的背景，指出其通过解决零门槛自动化处理、创新视觉表达、教育场景可视化、低成本个性化创作及情感化数字纪念等五大痛点，填补了音频内容在视觉传播、教育应用和情感表达领域的空白。在设计思路部分，报告论证了纯软件实现方案的合理性，选择了Python作为编程语言，并采用基于FFT的频谱图作为核心音频可视化方案。同时，报告还构建了项目文件链路系统，包括本地计算与边缘机链路计算，并展示了链路图及系统流程图。在音频可视化部分，报告介绍了FFT原理及其优势，并展示了相关代码。工程整合部分，报告详细描述了文件传输链路、文件组成框架及文件读写、视频生成等部分的代码实现。最后，报告通过测试方案及结果验证了项目的可行性和效率，并进行了总结与致谢。

1. **设计原理与步骤**

**2.1项目背景**

声波绘影是一个奇妙的项目，它能够将用户提供的一段音频在本地计算或发送到中心机后，实时地生成这段音频的频谱，屏幕上的频谱会随着音频信号的变化而随之跳动，从而实现将一段声音形象化、具象化的目标。视频信号与音频信号也会经过调制与解调的方式发送回用户。它能以一种简单的方式，赋予每个人用计算机软件创造艺术的能力。

声波绘影通过将音频转化为动态频谱视频，解决了五大痛点：1）零门槛自动化处理，替代复杂专业软件，降低学习成本；2）创新视觉表达，用动态频谱增强社交媒体内容吸引力；3）教育场景可视化，将抽象声学概念转化为直观教学素材；4）低成本个性化创作，满足定制化视觉需求；5）情感化数字纪念，让声音记忆转化为可保存的艺术品，填补了音频内容在视觉传播、教育应用和情感表达领域的空白。

**2.2设计思路**

**2.2.1硬件与软件方案论证与选择**

方案一：纯软件实现。成本较低，易于实现，且有众多功能全面丰富的库函数可供调用，可以满足本项目中复杂的数据与信号处理需求。但软件模拟无法完全复现真实物理系统，且对于项目中传输与调制解调等在真实系统中必要的流程在纯软件实现的方式下无法较好地展示出来。

方案二：借助单片机、LCD屏幕等工具软件硬件结合实现。其结构更加完善，更加接近项目的理想状态，但硬件成本较高，且数据处理与实时运算能力不如PC。此外，不同模块间通过WiFi、蓝牙等方式通过射频信号发送文件，连接与数据传输难度较大。

综上所述，鉴于实现难度、项目成本、组员个人能力等因素，本组决定采用纯软件实现方案。

**2.2.2编程语言方案论证与选择**

方案一：选择C/C++。C语言执行效率较高，速度更快，操控硬件能力更强，但学习成本较高，调试难度大，且图表生成能力不如其它编程语言。

方案二：选择Python。Python语言简明易懂，学习成本低，库函数丰富，图标生成能力与文件处理能力强，且项目成员对Python语言也更为熟悉。但python语言执行效率较低，占用运算资源较多。

方案三：选择MATLAB平台实现。MATLAB专为数据处理设计，运算能力与绘图制表能力强大，但是软件资源获取难度较大，且编程语法规则与主流编程语言差别较大，学习成本高。

本项目的核心是在软件平台上模拟项目正确读取、传输并可视化音频信号的全流程，因此对编程语言的运算、数据处理能力与制图能力要求较高。且项目代码量大，易于调试的编程语言可以大大减少项目工作量，因此，在综合考虑项目特点与项目成员个人能力情况后，我们选择Python语言作为编程语言。

**2.2.3音频可视化方案论证**

方案一：基础波形可视化​​

采用传统的振幅-时间波形显示方式。这种方式实现简单直观，能直接显示音频信号的强度变化。然而，经Python模拟测试，波形图无法展示声音的频率成分，也无法直观体现音乐的旋律特征。对于语音和音乐分析项目，它无法清晰展现不同乐器的音色差异或语音频谱特性。

方案二：频谱图可视化​​

采用STFT技术生成时频谱图。该方案通过快速傅里叶变换(FFT)将时域信号转换为频域信息，能够直观展示不同频率的能量分布。虽然频谱计算需消耗一定算力，但现代计算机完全能够胜任实时处理要求，且能保留声音的完整频谱特征。

我们的音频可视化系统主要面向音乐和语音分析场景，而频谱图能直观展示音乐的和声结构、乐器的频率分布等关键特征。相较于简单波形图，频谱图虽实现稍复杂，但对音频特征的表达能力成倍提升。通过调整FFT窗口大小(nperseg)和重叠量(noverlap)，可精确控制时间和频率分辨率的平衡。

同时，频谱可视化在专业应用(如音频编辑软件、音乐可视化软件)中已成为标准方案，符合现代音频处理技术发展趋势。因此，我们选择基于STFT的频谱图作为核心音频可视化方案，既能满足项目实时性要求，又能提供专业级的音频分析能力。

**2.2.4项目文件链路构建方案**

我们的项目文件链路方案分为两部分：本地计算与边缘机链路计算。本地计算模拟的是高算力中心机的视频生成。它接受用户发送的wav文件后在本地实时生成各个采样点的png格式频谱图，并将若干张频谱图拼接生成mp4视频文件，在生成视频过程中做到音视频同步。

边缘机链路计算模拟的是低算力的边缘机（如麦克风、小型终端）接收到音频信号后，将其发送至中心机生成视频并将视频文件与音频信号返回至边缘机的全过程。边缘机首先将输入的wav音频文件转化为bin格式的二进制文件，随后将二进制文件传输至高算力中心机，中心机以与本地计算同样的方式生成mp4文件后返回边缘机。

受限于全软件实现方案，我们无法模拟边缘机与中心机进行文件传输的真实过程。我们也曾尝试用代码实现边缘机与中心机间通过调制解调实现bin文件传输的过程，但传输效果欠佳。我们会在未来思考边缘机与中心机之间文件传输更优的解决方案。

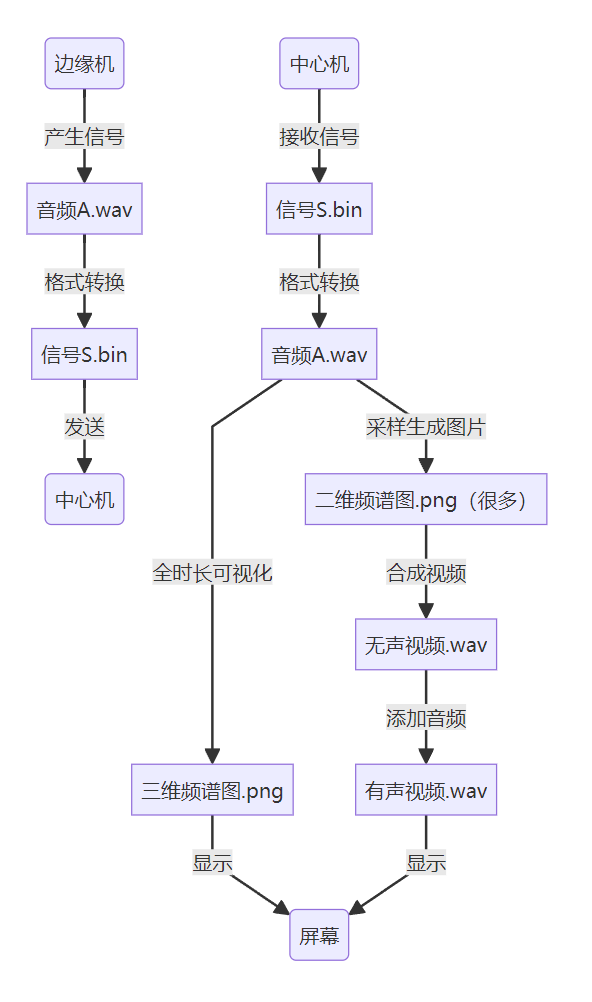
**2.2.5链路图及系统流程图展示**

图 1：链路图及系统流程展示图

**2.3音频可视化部分程序设计**

**2.3.1** **FFT原理介绍及其优势**

FFT是离散傅里叶变换（DFT）的高效算法实现，用于将时域信号转化为频域信号：

其中，是时域信号采样点，是采样点数，是频率索引。

在音频可视化系统中，我们采用快速傅里叶变换(FFT)技术作为核心分析手段。FFT本质上是将音频信号从时间域转换到频率域的高效数学工具。想象一下，任何复杂的声波都可以被分解成不同频率、不同振幅的正弦波组合，就像一道复杂的菜肴可以被分解为各种原料和调味料一样。FFT就是那个精准"分解"声音的"分析器"。

FFT算法的核心优势在于其计算效率。对于个采样点的音频片段，传统傅里叶变换需要的计算复杂度，而FFT只需的复杂度。这使得我们可以实时处理较长的音频片段，不会造成卡顿。例如，在标准音频采样率(44.1kHz)下处理0.1秒的音频片段时，FFT的处理速度比传统方法快数百倍。

**2.3.2 FFT生成频谱图部分代码展示**



图 2：FFT生成三维频谱图部分代码展示图

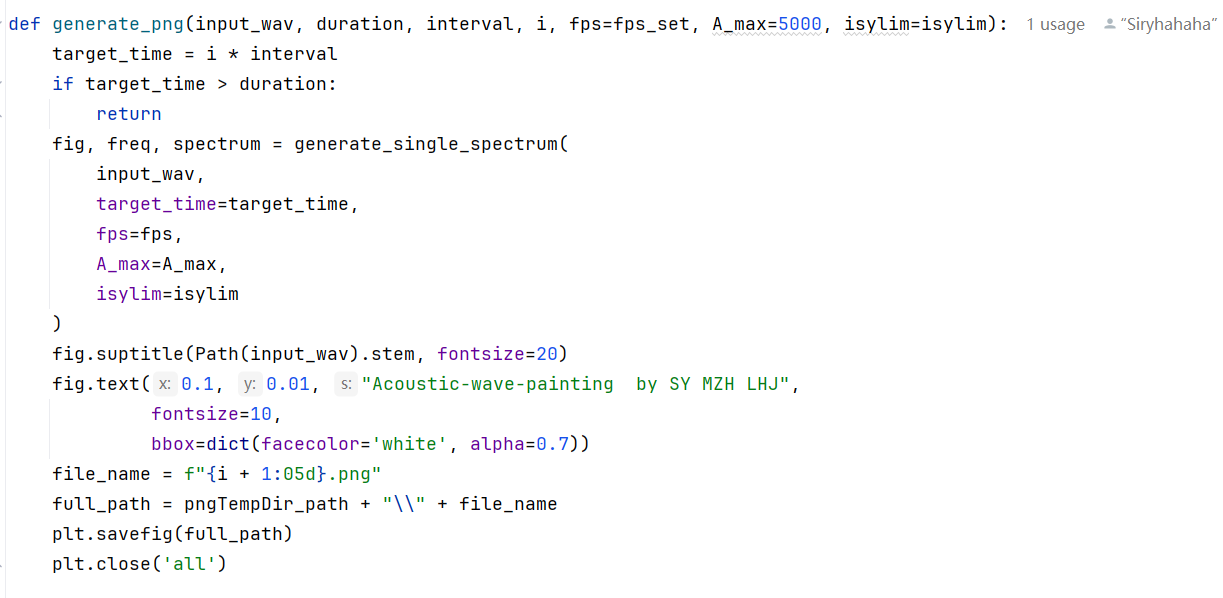


图 3：FFT生成二维频谱图部分生成代码展示图

**2.4项目文件链路系统以及工程整合**

**2.4.1文件传输链路方案**

文件传输链路方案分为全本地链路和异地链路两种。

全本地链路的流程如下：选择wav文件输入 - 选择帧率 - 开启生成频谱视频，即可生成全频段频谱图和频谱视频，以上资源均可查看或保存。

异地链路的流程如下：

边缘设备（低算力）：选择wav文件输入 - 不生成频谱视频 - 保存bin文件 - 发送到中心设备

中心设备（高算力）：接收bin文件 - 选择bin文件输入 - 选择帧率 - 开启生成频谱视频。以上资源可查看或保存。

**2.4.2具体文件传输路径**

项目的外部导入路径为：wavInput\_path、binInput\_path，此变量的值表示接收外部导入wav或bin文件的路径。

本地转化存储：转化后保存为wav和bin格式wavOutput\_path和binOutput\_path，此变量的值表示接收保存在终端中的wav或bin文件的路径。

生成三维频谱图：输入音频转化后生成三维频谱图pngTAF\_path，此变量的值表示三位频谱图的保存路径。

采样生成二维频谱图：对音频信号转化采样生成二维频谱图。存储在pngTempDir\_path，表示生成频谱视频的二位频谱图的临时存储路径。

生成频谱视频：按帧率合成静默视频到mp4Silent\_path变量中的路径，再合并音频到mp4Output\_path变量中的路径。

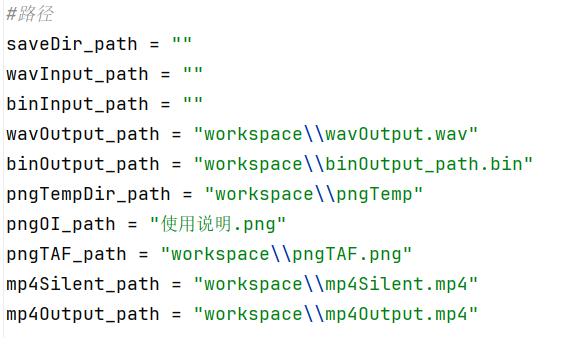


图 4：路径变量代码汇总

**2.4.3文件组成框架**

工程文件主体共由4部分组成：使用说明.png、main.py、lib文件夹workspace文件夹，此外还包含了一些示例文件可供调用。

使用说明.png：，此图片阐述了文件的操作指南，以及两种链路的方案。

main.py：此为项目主函数，主要包含UI搭建、文件读取保存、绘制函数接入等部分代码。

lib文件夹：包含constants.py、fileIO.py、painting.py。constants.py引用了所有项目所需要的外部库，并包含存储路径、标志位、帧率等全局变量。fileIO.py主要处理wav、bin、png等格式的文件的读取和转化，视频合成与音频添加。painting.py文件中包含对音频进行进一步处理的函数，包括得到音频基本参数（频率范围、最大幅值等），绘制三维频谱图、绘制二维频谱图。

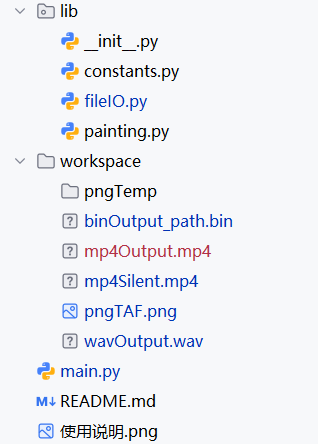
****workspace文件夹包含了工作文件，包括生成的wav音频文件和bin二进制文件、三维频谱图、采样二维频谱图、静默和有声视频。workspace会在开始运行时初始化，结束后清空采样频谱图的文件夹，以节约存储空间。

图 5:项目文件汇总展示图

**2.4.4部分代码展示**

图 6：文件读写部分代码展示图

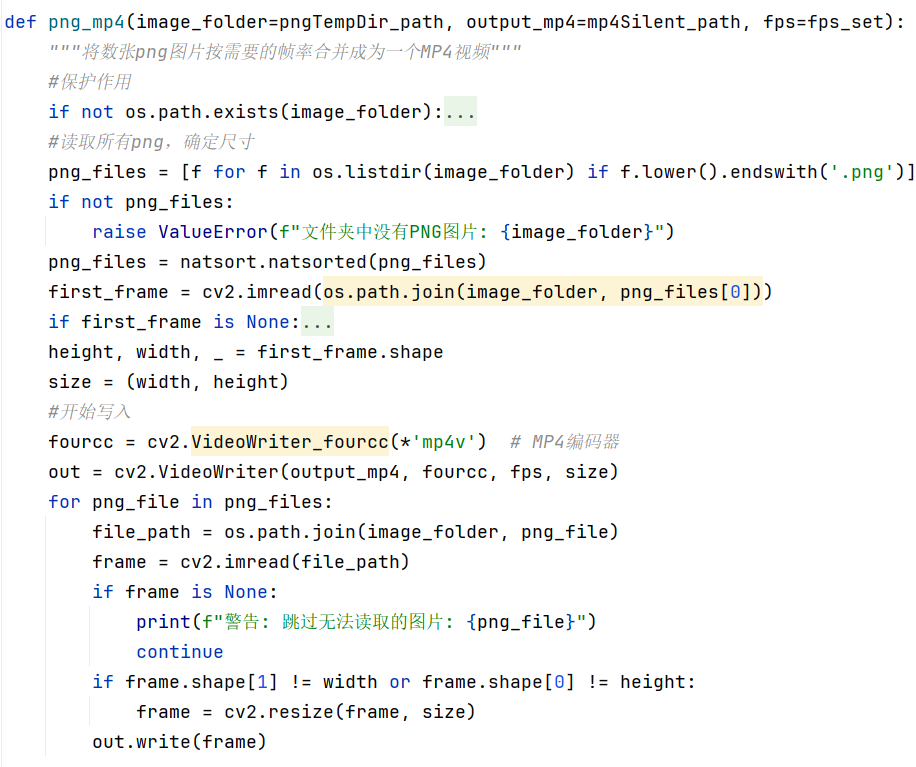
1. **设计结果与分析**

图 7：图片合成为视频部分代码展示图

**3.1测试方案以及条件**

测试方案：分别测试全本地链路频谱视频（fps=24）、三维频谱图生成速度和边缘机二进制bin文件生成速度，例程音频时长分别为3s、6s、9s、16s（二进制bin文件生成特别添加141s例程），各三次取平均值。

测试条件：笔记本PC终端，处理器Ryzen7 7840HS，RAM 16GB，Pycharm编译器。测试过程中未开启其他大负载，未对算力做特别调度，手动计时。

**3.2音频可视化测试**

表 1：全本地链路频谱视频和三维频谱图生成速度（fps = 24）（单位：s）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 音频时长 | **3s** | **6s** | **9s** | **16s** |
| 第一次测试 | **31.50** | **63.26** | **94.53** | **139.64** |
| 第二次测试 | **31.40** | **64.05** | **94.66** | **140.32** |
| 第三次测试 | **30.94** | **63.88** | **94.20** | **139.33** |
| 均值 | **31.28** | **63.73** | **94.46** | **139.76** |

表 2：边缘机二进制bin文件生成速度（单位：s）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 音频时长 | 3s | 6s | 9s | 16s | 141s |
| 第一次测试 | **2.20** | **3.35** | **4.45** | **7.14** | **59.12** |
| 第二次测试 | **2.08** | **3.12** | **4.26** | **7.03** | **58.48** |
| 第三次测试 | **2.11** | **2.98** | **4.95** | **7.02** | **58.86** |
| 均值 |  |  |  |  |  |

1. **设计总结与心得**

本次项目从2025年3月31日开始，过去的两个半月中，我们得到了很多外部的帮助，在此特别表达感谢。

首先是信号与系统课程的指导老师杨俊美，她不仅教授了我们傅里叶变换等课程知识，还在项目起步的时候对可行性报告提出了一些建议，她的建议“可以事先生成视频，然后和音乐对齐来播放。”最后的确极大加快了视频文件的输出速度。

其次是Deepseek等AI模型，虽说这些模型在工程中后期难以应对多文件编译和复杂链路的理解，闹出了一些笑话，但其在UI框架搭建、库函数说明、文件规范性等方面的确教会了我们很多很多。

还要特别电子与信息学院无线电爱好者协会，为我们小组的组会提供了会议室和一些设备。

最后，本项目能在我们三个没什么水平的大二学生手中诞生，不能不感谢伟大的开源互联网，我们愿将从开源项目中学到的东西再回馈到开源互联网中。本项目已在github开源，开源地址：https://github.com/Siryhahaha/Acoustic-wave-painting.git，欢迎浏览并留下一个star!