Project Freedom Extended Report

Gabriel Aldous
MAS4115

2023-12-13

Contents

1 Executive Summary

1.1 Project Overview

Project Freedom is a easy to use CLI program that users can interact with in order to perform their own experiments with the Fast Fourier Transform and related operations on both images and audio files. Basic filtering operations are supported for images and audio, and there are numerous ways to visualize how the files have changed and preview those changes. Additionally, the application supports easy modification of the file system, and the classes employed to perform these image modifications can easily be repurposed by other programmers to add additional functionalities.

In addition to the basic manipulations, there is an implemented advanced 'hybridization' technique for both audio and image files. Image files can be 'hybridized' so that the image appears to be one thing up close, and another when viewed from far away, while audio files can be modified to make it appear that one sound is being heard within a different environment.

This project is implemented in Python, and supports building on Python versions 3.9-3.12 (though it was only tested on version 3.12). It relies on Numpy, Matplotlib, Scipy, and PyGame, and instructions for setting up and running the program are available in the README file. To view the results of various operations performed by Project Freedom, see the github repository.

1.2 Mathematical Principles

The fundamental operation which this program seeks to foster a greater understanding of is called the Fourier Transform, or more specifically the Fast Fourier Transform (FFT). This technique is often used in signal processing applications, as is the case in this project. The FFT is utilized to turn some signal data (such as an image or an audio file) into a different domain called the frequency domain. Simply speaking, the FFT takes the input and restructures it so that information about which frequencies are more prevalent is easily accessible.

In the context of the custom hybridizations, audio hybridization is performed using a convolution, while image hybridization is performed by filtering the high detail portions of one image (the edges & lines) and overlaying it on the low detail portion of another image (the general colors & shapes). This results in an image which can trick human perception and appear differently from different viewing distances.

The FFT has applications in many areas, notably in image compression for-

mats like JPEG and MP3, filtering operations, wireless communication, and much more. For more details about how the FFT works, and the linear algebra supporting it, view the mathematics explanation section of this document.

2 Motivation

When considering the potential avenues of exploration for Project Freedom, there were 2 main paths which I considered. The first identifying malware files utilizing neural networks, was interesting but I didn't feel comfortable installing thousands of files of malware onto my laptop, even within a VM environment. That left this option of exploration, which was appealing due to how it ties into my current coursework (Fourier Analysis) and future coursework (a CS technical elective about 3D Audio). Additionally, this was motivated by my Fourier Analysis course not covering the Fast Fourier Transform due to time constraints, so this became a nice way to apply linear algebra and the skills I learned in that course to a practical application.

3 Technical Report

This portion of the report shall primarily cover how to setup the repository, and show off various results from the program execution.

3.1 Installation & Use

To run Project Freedom yourself, you can install the program with the following commands from within a linux terminal.

```
git clone https://github.com/Sn00pyW00dst0ck/project_freedom.git
cd project_freedom
pip3 install -r requirements.txt
```

Note: on some systems 'pip3' may need to be replaced by 'pip'.

With the project installed, the program can be run by executing the following command from within the terminal in the BASE directory for the project (if you run the executable from anywhere else, then the program may attempt to read files from locations which do not exist, and may crash):

```
python3 ./src/main.py
```

Note: on some systems 'python3' may need to be replaced by 'python'.

When the project is executed, the user will be presented with a series of menu options to determine which actions they wish to preform. The menu systems are designed to walk the user through the operations which can be performed.

3.2 Program Execution Results

While I will briefly discuss some program execution results here, it is worth noting that audio files cannot be adequately represented within a PDF document. For this reason, I highly recommend viewing the 'samples' folder within the GitHub repository to listen to the execution results which I will be discussing, particularly in regards to audio.

3.3 Audio Outputs

The audio outputs created by Project Freedom mostly need to be heard to be believed, though there are some graphs which can be shown here. Keep in mind, the visualizations for each chart are different, particularly in the axis scales, in other words, not all charts are drawn to scale! That said, I highly recommend viewing the GitHub repository for this portion of the report.

3.4 Image Outputs

The image outputs created by Project Freedom are much easier to visualize.

4 Mathematical Explanation

4.1 Explanation of Fourier Transform and Fast Fourier Transform

4.2 Code Implementation

Within the source code of Project Freedom, the implementation of the FFT is offloaded to Numpy and Scipy libraries. The reasoning for this is two-fold. These libraries are widely tested and will provide accurate results, and secondly is that these libraries will provide a more robust implementation (both in terms of speed and versatility).

To see a sample of how FFT is used within the Python code, we examine the high_pass_filter of the Audio_Processor class:

```
def high_pass_filter(self, cutoff_frequency):
"""
Apply a high pass filter to the loaded audio
effectively isolating the high frequencies only.

Args:
    cutoff_frequency (int):
        The frequency at which to apply the filter.
```

```
    """
    if self.audio_data.ndim == 1:
        fft_audio = fft(self.audio_data)
        # Get the frequencies corresponding to each point in the FFT
        frequencies = np.fft.fftfreq(len(fft_audio), d=1.0/self.sample_rate)
        # Create a mask for frequencies above the cutoff
        filter_mask = np.abs(frequencies) >= cutoff_frequency
        # Apply the mask to the FFT
        filtered_fft = fft_audio * filter_mask
        self.audio_data = ifft(filtered_fft).real
        return

    for channel in range(self.audio_data.shape[1]):
        # Apply FFT to the audio signal
        fft_audio = fft(self.audio_data)

        # Get the frequencies corresponding to each point in the FFT
        frequencies = np.fft.fftfreq(len(fft_audio), d=1.0/self.sample_rate)

        # Create a mask for frequencies above the cutoff
        filter_mask = np.abs(frequencies) >= cutoff_frequency

        # Apply the mask to the FFT
        filtered_fft = fft_audio * filter_mask

        # Apply Inverse FFT to obtain the filtered audio signal
        self.audio_data[:, channel] = ifft(filtered_fft).real
```

This method has been chosen for examination as it is a typical use case for FFT within the project. The program determines if the FFT will need to be used one time, or multiple times based on how many channels of data are present (mono vs stereo). Then, the FFT operation is utilized to convert a channel from the normal space to the 'frequency space' (represented as fft_audio). Further processing is utilized to get the individual frequencies from this, which can create a mask to filter out certain frequencies which are unwanted. The filter is applied with element wise multiplication, and then the inverst FFT is utilized to restore the audio to a format which can be played.

5 Lessons Learned

Implementing Project Freedom has re-inforced the concepts which I learned within Fourier Analysis, while also allowing me to further apply those skills in a practical setting. Meanwhile, writing this extended report has alllowed me to further cement some linear algebra topics and solidify my understanding of the Fourier Transform. This will prove invaluable to me in future coursework.

Beyond that, I tackled some interesting programming issues during the creation of this repository which have given me practice in setting up a proper Python development setting and solving interesting program organization challenges. This practice could prove valuable in the future, whether in grad school or a industry setting.