REPORT

Astro Sonification was an exhilarating experience as getting into the basic of sonification through the interactive assignments which helped to create the building blocks which finally helped me to create my first sonification tool. Getting introduction to new things were challenging but fun to get myself into through the weekly assignments. I further have shared my experience and challenges faced through-out the project -

Assignment 1 -

Through this assignment, I was introduced to how the MIDITime library is used to transform lunar crater data into a musical format by mapping numerical data like crater diameter into pitch and velocity within the D minor scale. It helped me to understand that design of functions ensures clarity and precision in handling specific tasks like pitch tuning and velocity mapping. The outcome of MIDI output is well-translated scientific data in music form, providing a way of interpreting trends.

**Challenges**- Going through all the resources and tutorials was a little challenging. Error handling was hard as it was a new thing for me with MIDITime and other libraries. Getting hands on python also took some time. Although the resources were good enough to get a basic understanding, further researching sonification was also time-consuming.

Assignment 2 -

This enabled a captivating way to represent stellar light curve information and use the Lightkurve library to process light curve data for Kepler-62b. A light curve and pixel data download may be started with Matplotlib being used to plot, effectively both the original as well as smoothed light curves of both. Sonification represents translating normalized flux values in audio frequencies and producing the audio signal that is to be frequency modulated. The use of amplitude modulation ensures two quite different audio representations as outputting the flux of star change within time.

**Challenges** - The library, LightKurve, with fixed parameters for range frequency, duration, and sampling rate, does reduce the flexibility. Understanding this LightKurve library itself was a bit challenging, took time as every new thing is a slow process. Error handling, the problem of failed downloads, and incompatible data formats became a common issue and needed solving. Further enhancement can include better documentation, user-configurable parameters, and exception handling, which would significantly improve usability and adaptability of code.

Assignment 3 -

Assignment 3 helped me to efficiently handle the analysis and visualization of high-resolution galaxy image data. One can begin by loading the image data and selecting a specific image for processing. The RGB values of each pixel are extracted and saved into a CSV file, ensuring accessibility and further usability of the pixel data. You can also visualize the image and its red channel intensity using Matplotlib, providing clear insights into the distribution of pixel values.

**Challenges** – Difficulties in moving forward were nothing new since it was hard to understand the workflow of the assignment. There was also the difficulty of error handling because not used to python and coding in general. Further improvement would be to have markdown explanations, dynamic parameterization for file paths and image selection, and basic exception handling which would increase its usability and robustness, and make it versatile enough for a variety of datasets or workflows.

Assignment 4 -

It helped me explore an innovative approach to sonifying galaxy image data by converting pixel RGB values into sound, thus mapping the red, green, and blue components to frequency, volume, and duration, respectively, using a sine wave generator from the Pydub library. I created a nuanced auditory representation of the image. I also understood how to handle large datasets efficiently by processing pixel data in batches, reducing memory usage. Moreover, I added the saving of an intermediate file, so any progress is saved and minimizes data loss if something goes wrong. The output is a composite WAV file that contains the sonified galaxy image.

**Challenges** – New libraries were challenges too. The workflow of the script and revisiting the previous assignments were a regular need. Though tutorials and exiting github repositories on image sonification were of great help. It can improved by including detailed documentation, dynamic file path handling, and exception management for file I/O operations. These additions would enhance usability and reliability, making the code more versatile and user-friendly.

Assignment 5 -

Finally, I learned how to develop a comprehensive image-to-sound conversion tool that exhibits flexibility through different types of sonification modes like brightness-based pitch modulation, color-based sound effects, and spatial stereo mapping, thus handling files in both .npy and common image formats (.png,.jpg). I even used functions like pixel\_to\_sound\_brightness and pixel\_to\_sound\_spatial, which gave creative mappings of image properties - such as brightness, colour, and spatial positioning of the image - to its corresponding sound parameters, frequency, duration, and panning in stereo.

**Challenges** - Potentially inefficient processing of big image files or high-resolution data due to a long computation time, given that a nested loop structure goes over every pixel. In addition, when working with big .npy files, there are memory constraints for running without proper hardware support. Improving the efficiency in handling pixel operations can be optimized using techniques for parallelization.

Potential Improvements for the ImageSonify Tool-

**1. Performance Optimization**

* **Parallel/Multi-threaded Processing**: Utilize batch processing to expedite the generation of pixel-by-pixel sound by parallelizing or multi-threading large images.
* **Automatic Resizing Option**: Offer automatic resizing and an option to allow dimensions adjustment before processing.
* **Memory Optimization**: Optimize data structures with less memory usage, eliminating the duplication of unnecessary memory while handling .npy files or large resolutions images.

**2.** **Ease of Use**

* **GUI**: Design a minimal GUI that allows users to input images, select modes, and output paths without forcing them to input at the command line.
* **Interactive Feedback**: Allow for previewing sound effects on selected rows or image sections before processing the whole image.
* **Automated File Handling**: Auto-detect the input file types and set default output paths to avoid human mistakes.

**3. Extended Sonification Features**

* **Advanced Sonification Modes**:
* Implement pixel clusters or patterns that control rhythm or tempo changes.
* Include support for user-defined mappings, for example, mapping specific colour ranges to instrument tones.
* **Sound Layers**: Create multiple sound modes, such as brightness and spatial mapping, and layer them to provide richer auditory outputs.

**4. Scalability and Flexibility**

* **Handling of Large Datasets**: Implement mechanisms for chunking .npy datasets or image batches to avoid overwhelming memory.
* **Cross-Platform Compatibility**: The use of libraries should be done without platform-specific dependencies, hence ensuring it is smooth on all the operating systems.

**5. Visualization Integration**

* **Real-time Visualization**: It can provide a real-time spectrogram or waveform of the sound produced along with the image in order to have better connection between the visual and audio data.
* **Pixel-to-Sound Mapping Visualization**: Displaying heatmap or animation would depict which pixel are mapped into sounds while producing.