Operating System: Windows 11

Environment Version: Visual Studio Code 1.83.1

TIFF Viewer Application

The TIFF Viewer reads and displays uncompressed TIFF image files. It uses Java Swing GUI for users to interact with and perform image processing operations on 24-bit RGB full-color TIFF images.

Libraries and Tools Used

The application is written in Java and utilizes the following libraries:

1. Java AWT (Abstract Window Toolkit): A set of APIs used for creating window-based applications. It provides components for building GUIs such as buttons, menus, and windows.
2. Swing: Built on top of AWT, Swing provides a richer set of components than AWT. It is used to create more sophisticated GUI elements that are platform-independent.
3. ImageIO: Part of the Java standard library, it is used for reading and writing images in various formats including TIFF, which is crucial for the application's core functionality.
4. Java Logging API: Utilized for logging errors and other information, which is essential for debugging and maintaining the application

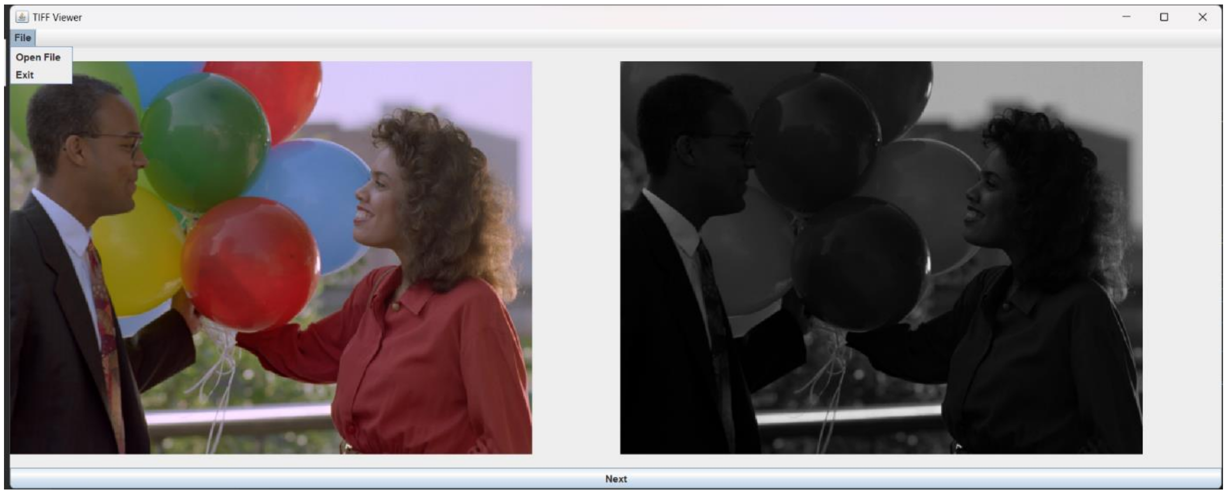
Graphical User Interface

The GUI is the main interface between the user and the application. It consists of:

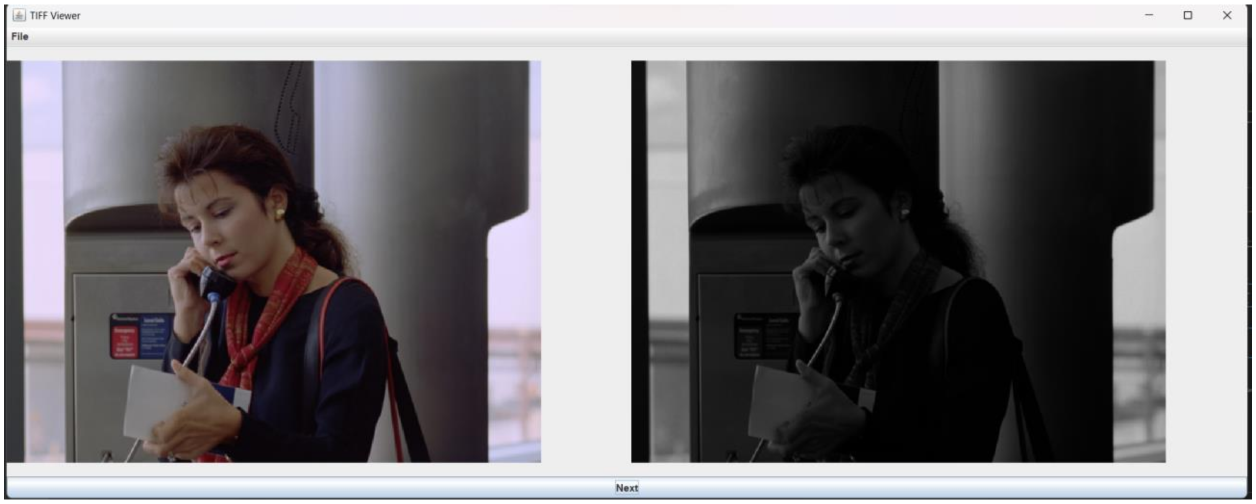
- A main window (JFrame) that houses all other components.
- A menu bar (JMenuBar) with a "File" menu that includes "Open File" and "Exit" items for file operations.
- A panel (JPanel) that contains two labels (JLabel) to display the original and processed images side by side.
- A "Next" button (JButton) allowing users to cycle through the image processing operations.

Application Workflow

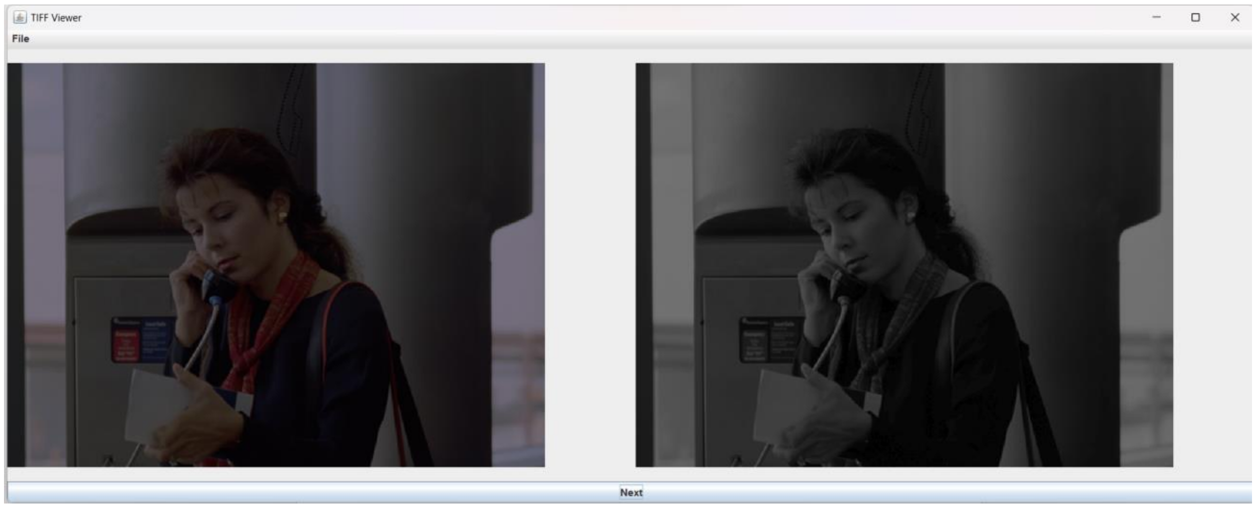
Upon launching, the application presents a window where users can choose to open a TIFF image file. Once an image is selected, the following sequence of operations can be performed:



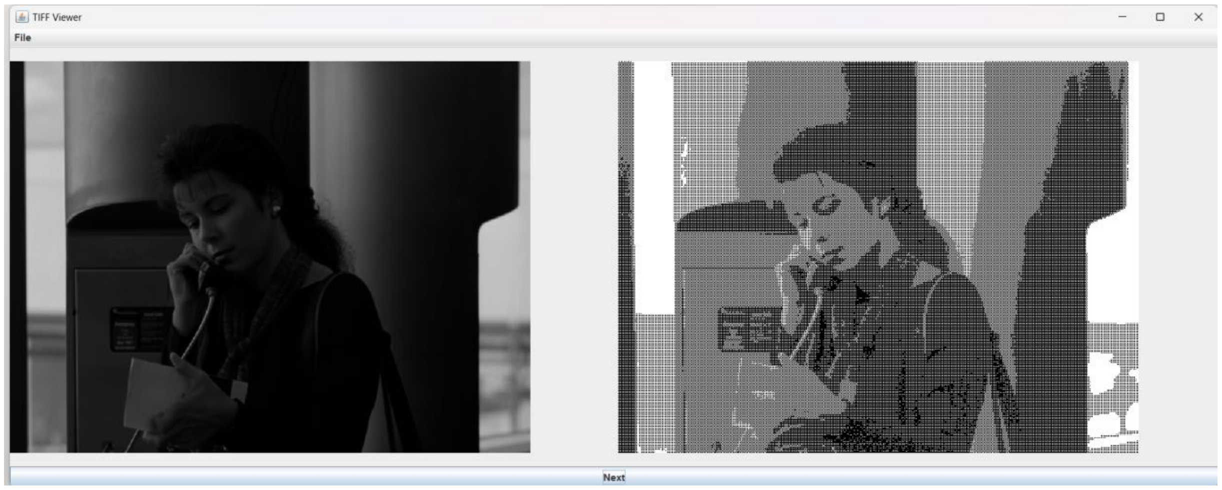
Display Original and Grayscale Images: The original colored image is displayed on the left, and its grayscale counterpart is displayed on the right.



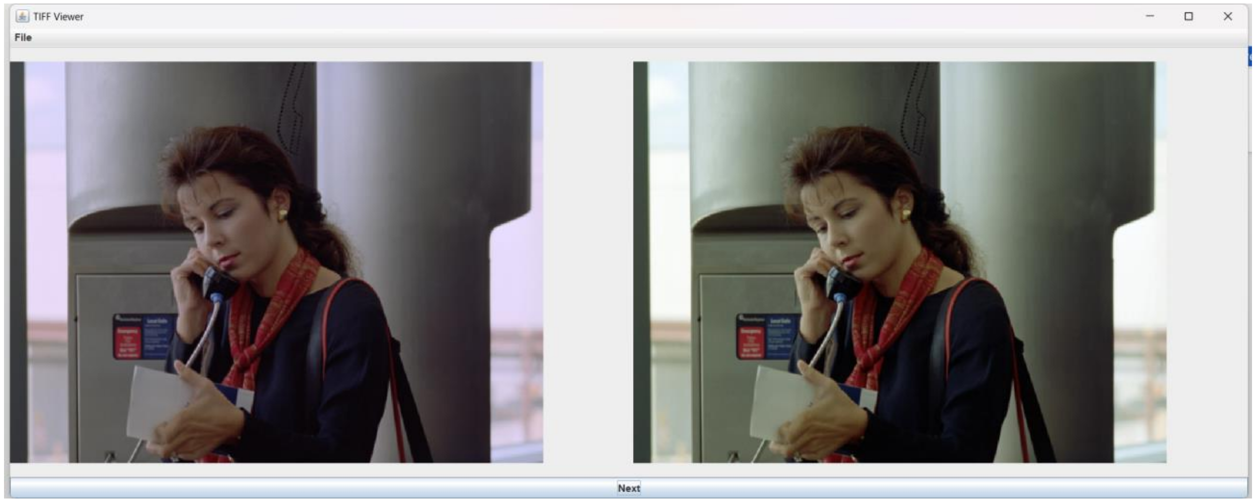
Brightness Reduction: The brightness of both images is reduced by 50%.



Ordered Dithering: The grayscale image undergoes ordered dithering, displayed on the left, with the right side showing the result of the dithering process.



Auto Level: The original colored image is auto-leveled to enhance its visual quality.



Dither Matrix Discussion:

The dither matrix typically contains values that determine the thresholds at which colors are changed. The matrix is tiled over the image, and for each pixel, the corresponding value in the dither matrix is used to decide whether to render the pixel as black or white. The choice of a dither matrix significantly affects the visual quality of the dithered image.

In the TIFF Viewer application, a simple 2x2 matrix is used:

$$\begin{matrix} 0 & 128 \\ 192 & 64 \end{matrix}$$

The matrix values range from 0 (black) to 255 (white), providing a basic threshold for the dithering. If the intensity is > the dither matrix entry, we print an on dot at the entry. This particular matrix has been chosen for its simplicity and relatively good performance on a variety of images. However, for higher-quality dithering, larger matrices with more levels can be used to create a more detailed and less coarse result.

Auto Leveling Algorithm Design

The algorithm first determines the minimum and maximum intensity values for each color channel across the entire image. Then, it stretches the histogram of each channel so that the minimum and maximum intensity values map to 0 and 255, respectively. This enhances the contrast of the image.

$$NewValue = \frac{255 \times (Value - MinValue)}{MaxValue - MinValue}$$

Each step is designed to be independent, allowing the algorithm to process the images in a sequence or separately as needed. The design is such that it can be extended to include additional image processing functions in the future.

Waveform Viewer Application:

The Waveform Viewer is a Java-based application designed to read, process, and display the waveform of stereo .WAV audio files. It provides a graphical user interface (GUI) that allows users to select and visualize audio data in a waveform format, offering insights into the audio's properties, such as entropy and average Huffman code length.

The Waveform Viewer program is designed to allow users to view the waveform of stereo audio files.

Graphical User Interface

1. Main Window (JFrame)

- Represents the primary window of the application titled "Waveform Viewer".
- Closing the window will terminate the application (setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)).

Waveform Panel

- This is a custom component (WaveformPanel) added to the center of the main window.
- Displays the waveforms of the left and right audio channels in visual form.
- The waveform rendering considers the width of the panel, channel data, and the gap between channels for accurate representation.
- Displays the sample rate and total number of samples as textual information.

Menu Bar (JMenuBar)

- Contains a "File" menu which further contains an "Open" menu item.

Open Menu Item

- When selected, it triggers a file selection dialog (JFileChooser) for the user to select an audio file.
- If a file is selected and approved, it attempts to load and render the waveform of that audio file.

Loading and Parsing the Waveform

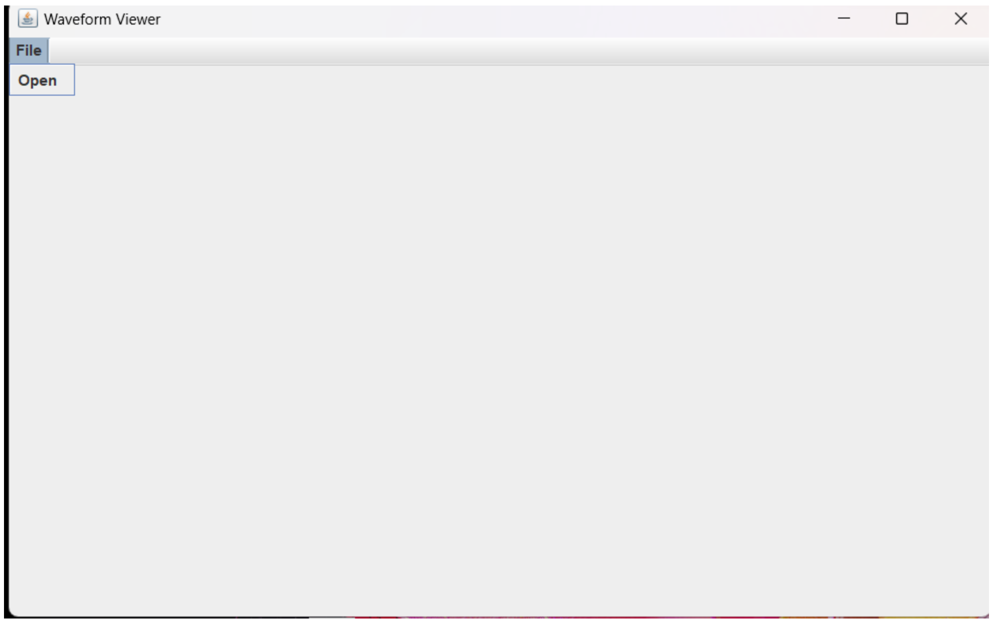
- The selected audio file is read into an AudioInputStream.
- Checks if the file has 2 channels (stereo). If not, it raises an exception indicating that only stereo files are supported.
- Audio data is then converted into bytes and further split into left and right channel samples.
- The samples are normalized and used to update the WaveformPanel.

Waveform Rendering

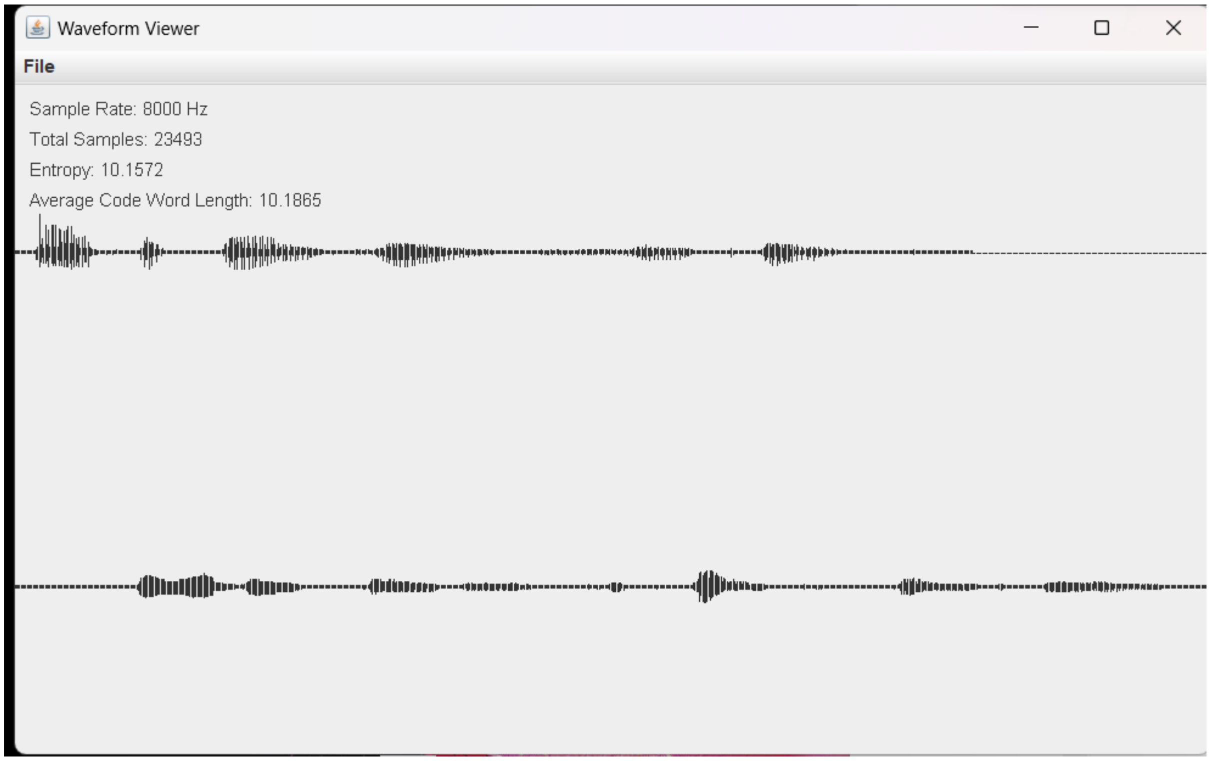
- For each sample in the left and right channels, lines are drawn representing the amplitude of the sound at that point.
- The left channel's waveform is drawn above the center of the panel, while the right channel's waveform is drawn below.
- A gap separates the waveforms of the two channels.

Application Workflow

Upon launching, the application presents a window where users can choose to open a .WAV audio file.



After the user selects a .WAV file it compress the audio samples in the file using Huffman coding. Next, the GUI will display the Entropy alongside the Average Code World Length



```
Sample Rate: 8000 Hz
Total Samples: 23493
Entropy: 10.1572
Average Code Word Length: 10.1865
```

Huffman coding is implemented to demonstrate a real-world application of data compression techniques. The code builds a Huffman tree based on the frequency of occurrence of each audio sample, then generates Huffman codes for each unique sample. This process is visualized by the average code word length metric displayed in the GUI.