

# Information and Coding

Lab work nº 1

João Monteiro (102690), João Sousa (103415), João Gaspar  
(107708)

Departamento de Eletrónica, Telecomunicações e  
Informática

Universidade de Aveiro



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Methodology</b>	<b>2</b>
2.1	Text Processing . . . . .	2
2.2	Audio Processing . . . . .	4
2.3	Image Processing . . . . .	6
<b>3</b>	<b>Results</b>	<b>8</b>
3.1	Text Data . . . . .	8
3.2	Audio Data . . . . .	12
3.3	Image Data . . . . .	17
<b>4</b>	<b>Discussion</b>	<b>22</b>
4.1	Outcomes, Strengths, and Limitations of the Methods . . . . .	22
4.2	Learning Experience: Challenges and Successes . . . . .	23
<b>5</b>	<b>Conclusion</b>	<b>24</b>

# Acronyms

**SFML** Simple and Fast Multimedia Library

**MSE** Mean Squared Error

**SNR** Signal-to-Noise Ratio

**XML** Extensible Markup Language

**I/O** Input/Output

**PPM** Portable Pixel Map

**PSNR** Peak Signal-to-Noise Ratio

# List of Figures

3.1	Histogram illustrating the diversity of words in the Portuguese language, with an entropy value of 10.3228. . . . .	9
3.2	Character frequency distribution in Portuguese text, reflecting an entropy of 3.62677. . . . .	9
3.3	Word frequency histogram for the English text, indicating an entropy value of 9.72633. . . . .	10
3.4	Character frequency histogram for the English text, indicating an entropy value of 3.57883. . . . .	11
3.5	Audio file characteristics extracted from <code>sample01.wav</code> . . . .	12
3.6	Waveform visualization of the audio signal extracted from <code>sample01.wav</code> . . . . .	13
3.7	Amplitude Histogram for Left Channel. . . . .	14
3.8	Amplitude Histogram for Right Channel. . . . .	14
3.9	Amplitude Histogram for Mono Channel. . . . .	14
3.10	Amplitude Histogram for Side Channel. . . . .	14
3.11	Comparison of original and quantized waveforms. The original waveform is shown in white, while the quantized waveform is displayed in red. . . . .	15
3.12	The waveform at the top represents the original signal, while the waveform at the bottom corresponds to the quantized signal. . . . .	15
3.13	Waveform of audio quantized to 16 bits. . . . .	16
3.14	Waveform of audio quantized to 4 bits. . . . .	16
3.15	Loaded Airplane image. . . . .	17
3.16	Comparison of RGB channels and grayscale version of the image. . . . .	18
3.17	Graphical representation of the distribution of pixel intensities. . . . .	19
3.18	Comparison of original image and blurred images with different kernel sizes. . . . .	20
3.19	Visualization of the difference image between two images. . . . .	21
3.20	Comparison of grayscale image and images with different quantization levels. . . . .	21

# Chapter 1

## Introduction

This project focuses on the manipulation and analysis of three distinct types of data: text, audio, and images. The primary goal is to explore different data processing techniques, including transformation, compression, and visualization, while gaining deeper insights into their practical applications. Each section of the project is dedicated to specific objectives, methodologies, and results for the respective data types.

- **Text Processing:** This section emphasizes text manipulation, encompassing techniques such as reading, transforming, and analyzing text data. The goal is to extract valuable insights, like word frequency distributions.
- **Audio Processing:** Here, the focus is on manipulating audio data through techniques like quantization and visual representation. The objective is to assess the effects of these transformations on audio quality.
- **Image Processing:** This part involves applying transformations to images, including resizing, filtering, and quantization, with an analysis of how these changes influence image quality.

To facilitate the use of this project, a comprehensive README file is provided. This document includes detailed instructions on setting up and executing each module, covering all dependencies and installation requirements.

## Chapter 2

# Methodology

### 2.1 Text Processing

In this section, we will discuss several key transformations and algorithms that can be applied to analyze and manipulate text data. The goal was to extract word and character frequencies and histograms while allowing the user to modify the text data interactively through different transformation options.

#### 2.1.1 Key Transformations and Algorithms

- **Transformation Options:** The program was designed to be flexible, allowing users to choose which transformations to apply via a menu. This allows the user to test different combinations of transformations, offering a dynamic exploration of text statistics.
- **Lowercase Conversion:** One of the transformations to the text is converting all characters to lowercase. This step was crucial to ensure that words with different capitalizations (e.g., “Word” and “word”) can be treated in the same way during frequency analysis. This standardizes the text data, making the frequency counts more accurate. A loop was used to iterate through each character in the text, and the `tolower()` function from `<cctype>` was applied to convert each character to lowercase.
- **Punctuation Removal:** The other transformation is the removal of punctuation marks. Punctuation symbols, such as commas, periods, and exclamation marks, have no impact on the meaning of words and may alter frequency counts if they are not eliminated. The removal ensures that only the meaningful content of the text remains for analysis. The transformation was implemented using the `std::remove_if` function in combination with `iswpunct()`, a function that checks whether

a character is a punctuation mark. This allowed the program to eliminate all punctuation symbols from the text efficiently.

- **Extensible Markup Language (XML) Tag Removal:** Given that some input text files contained XML data, another key transformation was removing all XML tags. These tags are metadata, not part of the actual content, and need to be stripped before proceeding with text analysis. This option only functions if applied before lowercase conversion or punctuation removal. A regular expression

`(<[^\>]*)>`

was used to match all XML tags, and the `regex_replace()` function from the `<regex>` library was applied to remove them.

- **Character Frequency Count:** Once the text is normalized and cleaned, the frequencies of each character can be counted. This is essential for calculating metrics like entropy and for generating histograms. The text is iterated over, and for each character (excluding spaces), its frequency is tracked using a `std::map<wchar_t, int>`. Each character is used as the key, and its occurrence count is incremented as it is found.
- **Word Frequency Count:** In addition to character frequency, word frequency is another vital part of the analysis, by identifying the most common words. A parsing algorithm was used where the text is split based on spaces to identify individual words. For each word found, a `std::map<wstring, int>` tracks the frequency. This word segmentation, followed by frequency counting, allows us to analyze the distribution of words effectively.
- **Entropy Calculation:** Entropy provides a measure of the randomness or unpredictability of the text. By calculating the entropy based on character and word frequencies, we assess the uniform distribution within the text data. The Shannon entropy formula was applied:

$$H(X) = - \sum p(x) \log_2(p(x))$$

where  $p(x)$  is the probability of a character or word occurring, calculated from the frequency maps.

### 2.1.2 Tools and Libraries

- **Standard C++ Libraries:** For basic string manipulation, file Input/Output (I/O), and algorithm implementation (`<map>`, `<string>`, `<algorithm>`, `<cctype>`, `<locale>`).

- **Regular Expressions:** The `<regex>` library was used for removing XML tags from the text.
- **Simple and Fast Multimedia Library (SFML):** This was used to generate and display visual histograms of character and word frequencies.
- **Gnuplot:** For creating the histograms in graphical form, the program generates data files and plots using this external tool.
- **Filesystem (`<filesystem>`):** For traversing the directory structure and reading all text files in a given folder, this standard library feature was used.

Each of these tools and libraries was chosen with the goal of trading off efficiency against ease of implementation using the C++ framework. In addition, the flexibility of SFML allowed intuitive visual representations in the presentation of data.

## 2.2 Audio Processing

This section describes the methodology used for processing audio data throughout the project. The main objective was to explore a variety of audio processing techniques, including loading audio files, waveform visualization, quantization, and assessing audio quality.

### 2.2.1 Key Transformations and Algorithms

- **Loading Audio Files:** Audio files, particularly in the `.wav` format, were loaded using the SFML (Simple and Fast Multimedia Library) audio module. The `sf::SoundBuffer` class was employed to store the audio samples efficiently.
- **Extracting Raw Audio Samples:** After loading, raw audio samples were extracted from the sound buffer using the `getSamples()` method. This allowed direct access to the sample data for further processing and analysis.
- **Visualizing Waveforms:** The waveform of the audio data was visualized by plotting the amplitude of the samples over time. Using SFML's graphics module, a real-time graphical representation was created to observe the behavior of the audio signal.
- **Creating Amplitude Histograms:** Amplitude histograms were generated to study the distribution of sample values. The samples were grouped into bins, making it possible to observe how often different



amplitude levels occurred. Gnuplot was utilized to generate these histograms, which were saved as image files for later use in the report.

- **Quantization of Audio Samples:** Uniform quantization was applied to reduce the number of unique amplitude levels in the audio data. This was achieved by adjusting the number of bits used to represent each sample, thereby reducing the resolution of the amplitude values.
- **Quality Assessment:** The quality of the quantized audio was evaluated through metrics such as Mean Squared Error (Mean Squared Error (MSE)) and Signal-to-Noise Ratio (Signal-to-Noise Ratio (SNR)). These metrics provided a quantitative comparison between the original and quantized audio, highlighting any potential degradation in audio quality.

### 2.2.2 Justification for Methods

The SFML library was selected due to its simplicity and effectiveness in handling audio playback and manipulation. It allowed for easy loading and rendering of audio data. Visualizing the waveform was essential for understanding the time-domain characteristics of the audio signal, while amplitude histograms offered insights into the distribution of sample values.

Gnuplot was chosen for its capability to produce high-quality plots efficiently. By scripting Gnuplot commands, it was possible to automate the creation of visualizations, making the process both efficient and reproducible.

Quantization was implemented to explore the trade-offs between audio quality and file size, allowing for the study of lossy compression effects. The selected metrics, MSE and SNR, were effective in quantifying the impact of quantization, enabling clear comparisons between the original and processed audio samples.

### 2.2.3 Tools and Libraries

The following tools and libraries were used throughout the audio processing tasks:

- **SFML (Simple and Fast Multimedia Library):** Used for loading audio files, playback, and waveform visualization.
- **Gnuplot:** Employed to create detailed amplitude histograms and other data visualizations.
- **C++ Standard Library:** Utilized for general data handling, including structures such as `std::vector` for storing audio samples and `std::map` for managing histogram data.

## 2.3 Image Processing

This section outlines the approach employed for processing images in Portable Pixel Map (PPM) format in our project. The first goal of this project task was to analyze the images, extract information, and create histograms to illustrate the distribution of grayscale intensity values in the image pixels.

### 2.3.1 Code Organization

- **Trab3:** This file serves as the main program for this part of the project and includes a menu for interacting with the application.
- **ImageProcessor:** To maintain better code organization, we created an ImageProcessor class that utilizes OpenCV for loading, processing, and displaying images. It also integrates Gnuplot to generate visual histograms, which provide a graphical representation of image data.

### 2.3.2 Key Transformations and Algorithms

In this project, several key transformations and algorithms were applied to analyze and manipulate image data:

- **Loading and Displaying Images:** Using OpenCV, images in the PPM format were loaded into the program. This was the foundation for subsequent processing and analysis tasks.
- **Color Channel Separation:** After loading the image, it was separated into its Red, Green, and Blue (RGB) channels. This allowed us to individually analyze the intensity distributions for each color channel.
- **Grayscale Conversion:** The RGB image was converted to a grayscale image. This transformation simplifies the analysis by focusing on intensity values rather than color, which is useful for tasks such as histogram generation and filtering.
- **Histogram Calculation:** A histogram of grayscale intensity values was generated to illustrate how pixel intensities are distributed across the image. This was done using both OpenCV and Gnuplot, providing visual insight into the contrast and brightness of the image.
- **Gaussian Blur:** A Gaussian blur filter was applied to the grayscale image with various kernel sizes. This filter smooths the image by averaging neighboring pixel values, reducing noise and detail. Each kernel size resulted in different levels of blurriness, enabling a comparison of their effects.

- **Image Quantization:** The grayscale image underwent uniform quantization to reduce distinct intensity levels. We tested quantization levels like 2 and 8, simplifying the image while introducing varying detail loss, which helped us explore the trade-off between image quality and data compression.
- **Difference Calculation:** We compared two images by calculating the absolute pixel-wise difference, which highlighted significant variations. We also used metrics like MSE and Peak Signal-to-Noise Ratio (PSNR) to quantify differences and evaluate transformation quality.

## Chapter 3

# Results

### 3.1 Text Data

This section presents the results obtained from the text processing tasks applied to both English and Portuguese datasets after several key transformations, such as XML tag stripping, lowercase conversion, and punctuation removal, were applied to the text.

#### 3.1.1 Portuguese Text Analysis

- **Word Frequencies:** After the text-preprocessing had been performed, that is, after removing the XML tags, converting to lowercase, and removing punctuation, we analyzed word frequencies in the Portuguese dataset. As expected, common articles and prepositions such as "de", "a" and "que", appeared most frequently.
- **Word Entropy:** The word entropy for the Portuguese text was 10.3228. These results indicate a relatively high diversity of words in the Portuguese language, reflecting its linguistic complexity.

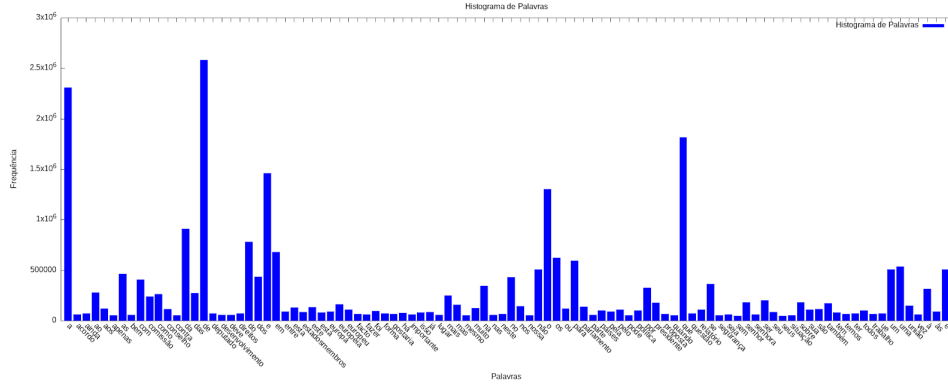


Figure 3.1: Histogram illustrating the diversity of words in the Portuguese language, with an entropy value of 10.3228.

- **Character Frequencies:** We also checked the character distribution in the individual characters of the Portuguese dataset. The vowels 'a', 'e', and 'o' ranked among the top most frequent characters, followed by common consonants such as 's', 'r', and 'n'. These are typical results when it comes to letter frequencies of the Portuguese language.
- **Character Entropy:** The character entropy for the Portuguese text was calculated as: 3.62677. This value indicates that the text has a moderately diverse character set, reflecting the variety of letters used in the language.

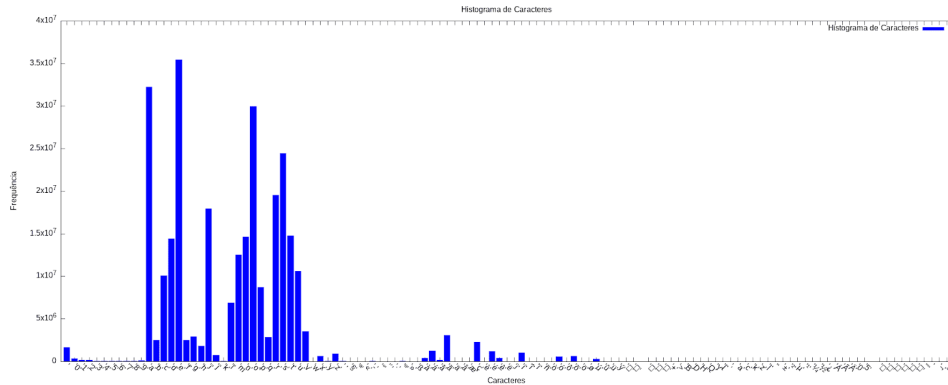


Figure 3.2: Character frequency distribution in Portuguese text, reflecting an entropy of 3.62677.

- **Impact of Transformations:**
  - **Lowercase Conversion:** This transformation allowed us to treat uppercase and lowercase letters as identical, it ensured that word

counts were not artificially inflated due to case differences. For example, variations like "A" and "a" were standardized, preventing the analysis from treating them as distinct entities.

- **Punctuation Removal:** We ensure that words with attached punctuation (e.g., "palavra.") were correctly identified as the same entity. This step improved the clarity of the text by removing extraneous symbols that do not contribute to meaning but could inflate the frequency analysis.
- **Impact on Character Entropy:** Without applying these transformations (lowercase conversion and punctuation removal), the entropy of character frequencies was 10.9659. This is significantly higher than 10.3228 reflecting a larger diversity of unique characters introduced by case variations and punctuation.
- **Impact on Word Entropy:** Similarly, without these transformations, the entropy of word frequencies was calculated to be 3.8997. After applying them, the entropy dropped to 3.62677, reflecting a more streamlined and consistent set of words for analysis.

### 3.1.2 English Text Exploration

- **Word Frequencies:** Transformations similar to Portuguese text were applied to English text. The most frequent words were articles and conjunctions such as "the", "of", and "to", which is typical of natural language text.
- **Word Entropy:** The word entropy for the English text was 9.72633. The entropy values reflect a slightly lower diversity of both characters and words compared to the Portuguese text.

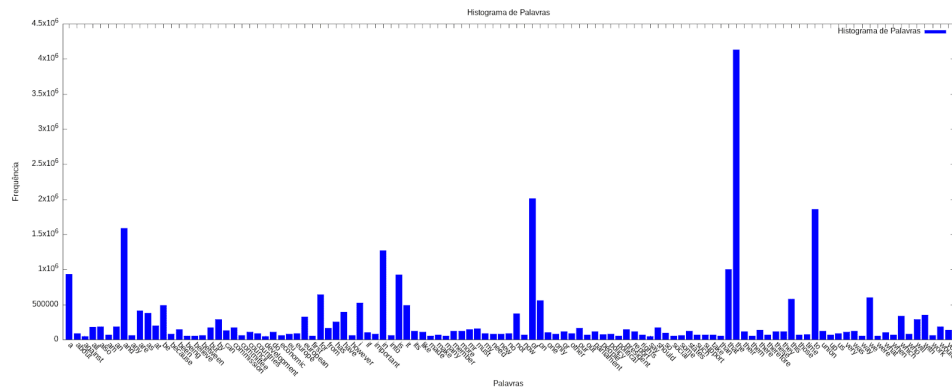


Figure 3.3: Word frequency histogram for the English text, indicating an entropy value of 9.72633.

- **Character Frequencies:** The most frequent characters in English texts are 'e', 'o', 'a' vowels and consonants 't', 'n', and 's'. This is simply because these letters happen to be the most used in texts written in the English language, with common words and grammatical structures using them.
- **Character Entropy:** The character entropy for the English text was calculated as 3.57883. This slightly lower entropy than the Portuguese text indicates a weaker variation of the set of characters for the English text.

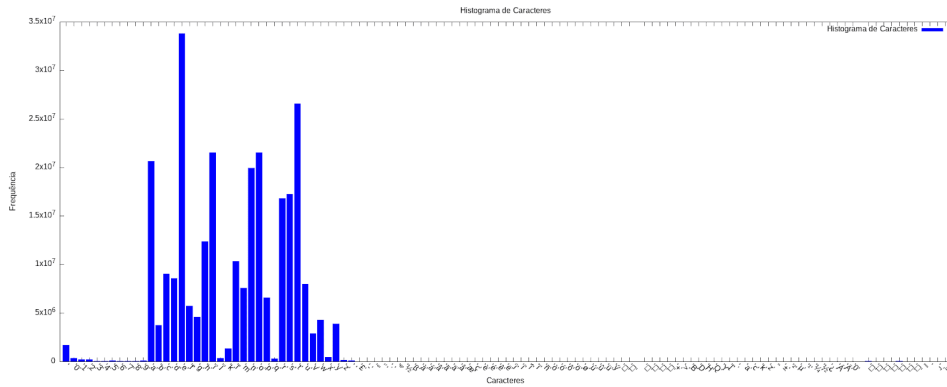


Figure 3.4: Character frequency histogram for the English text, indicating an entropy value of 3.57883.

- **Impact of Transformations:**
  - **Lowercase Conversion:** For both word and character analysis, converting all text to lowercase was essential. It ensured that variations only in capitalization (e.g., "The" and "the") were treated as the same entity, thus producing a more accurate frequency distribution.
  - **Punctuation Removal:** Removing punctuation was vital for accurately segmenting words and characters. This prevented punctuation attached to words or characters (e.g., "word." and "word,") from being treated as separate entities.
  - **Impact on Character Entropy:** Without applying these transformations (lowercase conversion and punctuation removal), the entropy of character frequencies was calculated to be 10.9659. This higher value indicates a greater variety of unique characters compared to 10.9659 due to case variations and punctuation.
  - **Impact on Word Entropy:** Similarly, converting all characters to lowercase ensured uniform treatment of letters (e.g., 'A' and

'a') in the word frequency analysis, preventing skewed results. Without these transformations, the entropy of word frequencies was calculated to be 9.72633. After applying them, the entropy dropped to 3.57883, reflecting a more streamlined and consistent set of words for analysis.

### 3.1.3 Comparison and Similarities

- **Character Frequency Comparison:** By analyzing both texts, one could tell how the character distribution of the Portuguese text was somewhat vaster, evidenced by the higher value of entropy at 3.62677 for the Portuguese text compared with that in the English text at 3.57883. Indeed, this is reasonably close to the expected difference, given that Portuguese uses a more significant portion of available letter space because words tend to be longer and more grammatically complex than those of English due to their verb conjugations and agreements.
- **Word Frequency Comparison:** The comparative entropy of the word proves a higher diversity in the Portuguese text, with 10.3228, against the English text, which is 9.72633. This reflects the more varied vocabulary usage in Portuguese, with more unique word forms.

## 3.2 Audio Data

This section presents the results obtained from the audio processing tasks performed on the audio file `sample01.wav`. The analysis includes waveform visualizations, amplitude histograms, quantization, and the evaluation of audio quality through MSE and SNR metrics.

### 3.2.1 Audio File Characteristics

The audio file used in this project had the following characteristics (see Figure 3.5):

```
Audio file information:  
Sample rate: 44100 Hz  
Number of channels: 2  
Duration: 29.35 seconds  
First 10 raw audio samples: 269 -1811 1372 1478 896 3714 -1539 4201 -5361 2669
```

Figure 3.5: Audio file characteristics extracted from `sample01.wav`.

These details provided a understanding of the audio data's structure and allowed for subsequent transformations and analysis.



### 3.2.2 Waveform Visualization

A waveform visualization was generated from the raw audio samples. This graph shows the variation in amplitude over time. The waveform is consistent with a typical audio signal. It does not reveal any notable patterns beyond the expected behavior. Each wave represents the amplitude of the audio as it varies over time (see Figure 3.6).

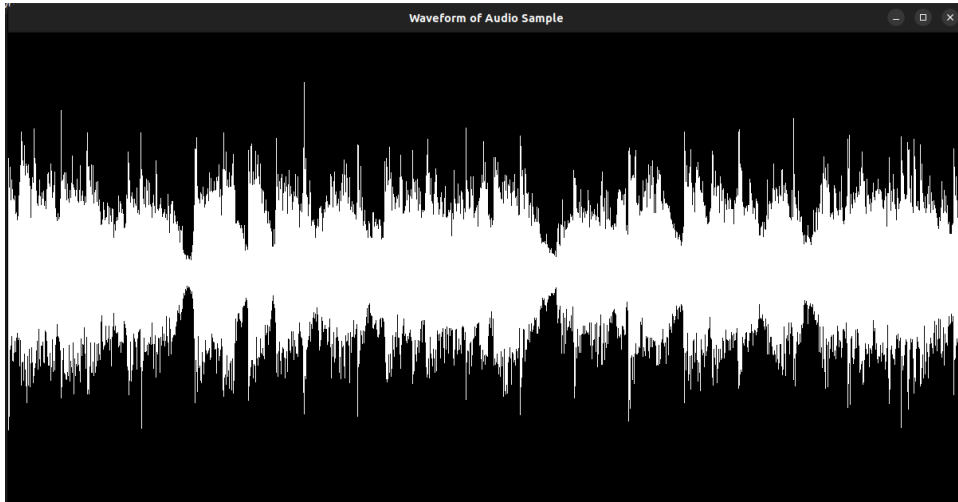


Figure 3.6: Waveform visualization of the audio signal extracted from `sample01.wav`.

### 3.2.3 Amplitude Histogram Analysis

Amplitude histograms were generated for the left and right channels, as well as for the mono and side channels derived from the stereo audio. The following observations were made:

- **Left and Right Channels:** Both channels operated within similar amplitude ranges, as expected. The amplitude distributions were smooth and broad, indicating a variety of amplitude values in the audio content (see Figure 3.7 and Figure 3.8).

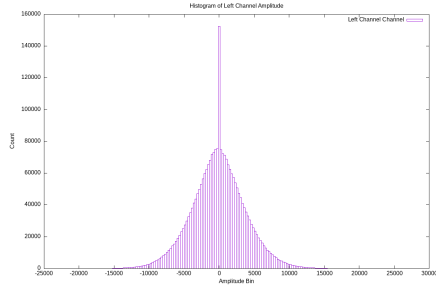


Figure 3.7: Amplitude Histogram for Left Channel.

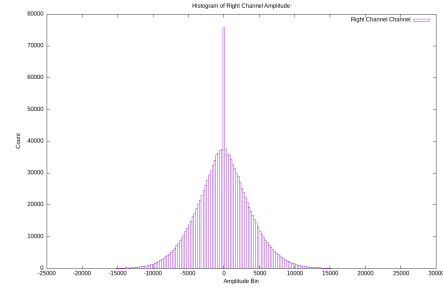


Figure 3.8: Amplitude Histogram for Right Channel.

- **Mono Channel:** The mono channel is the average of the left and right channels (see Figure 3.9). It displayed a distribution similar to the individual channels. This confirms that it maintained the general characteristics of the stereo signal.
- **Side Channel:** The side channel is calculated as the difference between the left and right channels (see Figure 3.10). It had a more concentrated distribution around zero. This indicates smaller, less varied amplitude values compared to the left and right channels.

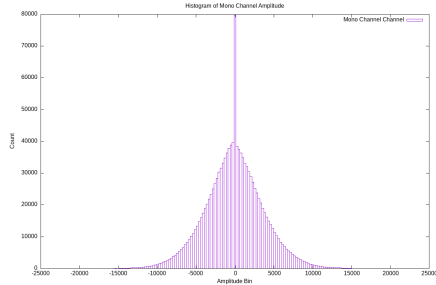


Figure 3.9: Amplitude Histogram for Mono Channel.

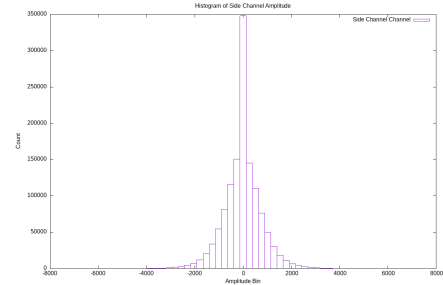


Figure 3.10: Amplitude Histogram for Side Channel.

The side channel showed a pronounced central peak. This suggests that the stereo signal had less variation between the left and right channels, which is consistent with most stereo audio recordings.

### 3.2.4 Quantization of Audio Samples

The audio file underwent uniform quantization with varying bit depths. This allowed the exploration of how reducing bit resolution affects the audio signal. The quantization process used user-defined bit values. The visual differences between the original and quantized waveforms were notable.

When quantizing the audio to 4 bits, a significant difference was observed between the original (white) and quantized (red) waveforms (see Figure 3.11 and Figure 3.12). The quantized waveform exhibited more pronounced "steps," corresponding to the reduced number of amplitude levels. Additionally, the quantized signal introduced noticeable noise, which was perceptible during playback.

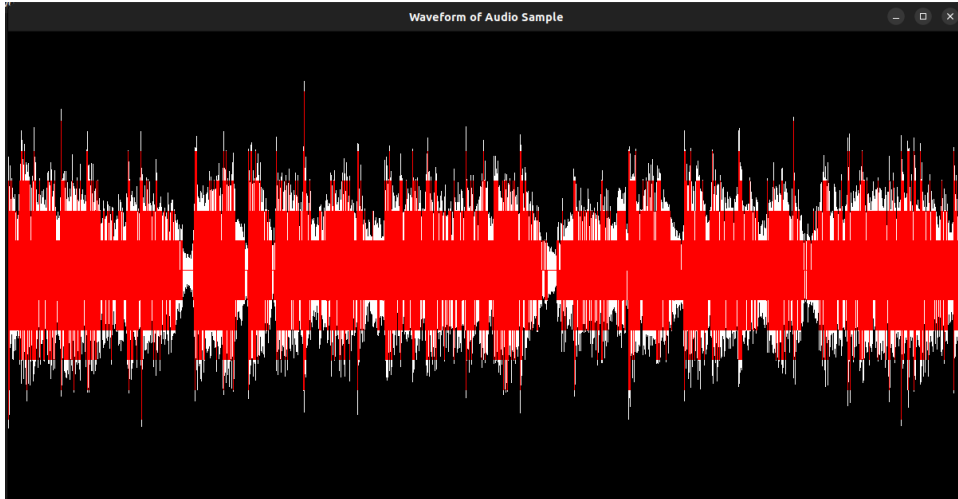


Figure 3.11: Comparison of original and quantized waveforms. The original waveform is shown in white, while the quantized waveform is displayed in red.

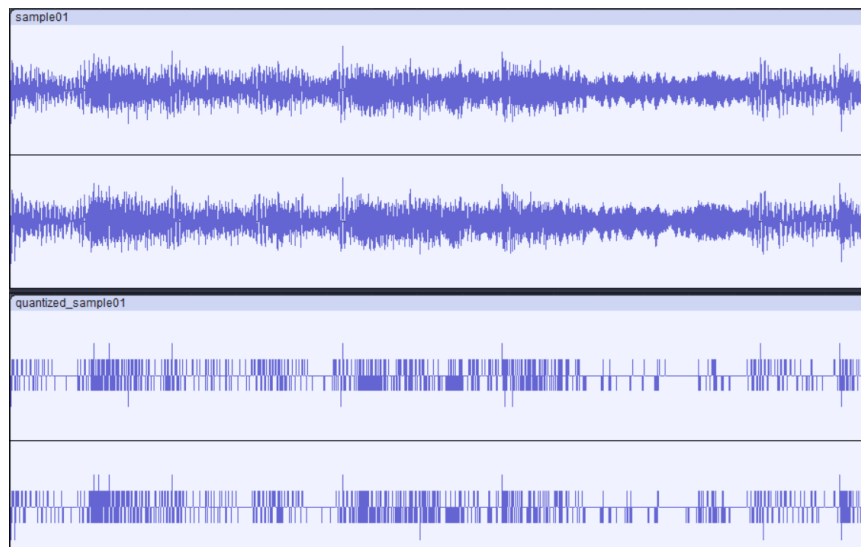
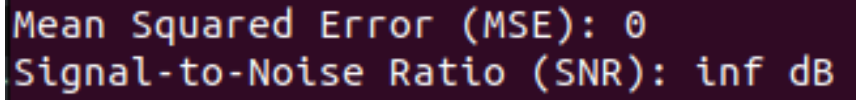


Figure 3.12: The waveform at the top represents the original signal, while the waveform at the bottom corresponds to the quantized signal.

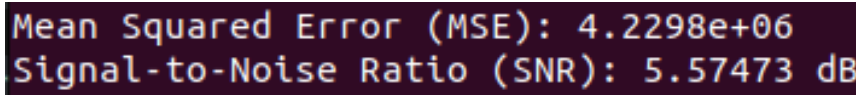
### 3.2.5 Quality Assessment: MSE and SNR Metrics

The quality of the quantized audio was assessed using MSE and SNR. The results varied depending on the bit depth used during quantization.



```
Mean Squared Error (MSE): 0
Signal-to-Noise Ratio (SNR): inf dB
```

Figure 3.13: Waveform of audio quantized to 16 bits.



```
Mean Squared Error (MSE): 4.2298e+06
Signal-to-Noise Ratio (SNR): 5.57473 dB
```

Figure 3.14: Waveform of audio quantized to 4 bits.

As expected, the 16-bit quantization produced no significant error, preserving the original audio quality without introducing noise (see Figure 3.13). In contrast, the 4-bit quantization resulted in substantial degradation of the audio signal. The higher MSE and much lower SNR indicate this. The SNR of 5.57 dB for the 4-bit audio indicates a significant amount of noise relative to the signal (see Figure 3.14).

### 3.2.6 Performance Analysis

- **Processing Time:** The processing time for audio quantization varied according to bit depth and file duration. Quantization at 2 bits was the fastest due to the simplicity of the quantization levels. On the other hand, 4-bit quantization took longer, as it involved more rounding operations. Despite handling more data, 16-bit quantization was efficient because it preserved the original sample values.
- **Trade-offs:** Quantization at 2 bits resulted in high compression with significant quality loss, while 16-bit quantization maintained the original audio quality but achieved minimal compression. A balance was observed with 4-bit quantization, which had a longer processing time and intermediate audio quality.
- **Comparison:** Music files, with high amplitude variation, showed greater sensitivity to compression, leading to more distortions. On the other hand, speech files maintained reasonable quality even with 4-bit quantization.

### 3.3 Image Data

This section presents the results obtained from processing tasks on sample images, including loading and displaying images, RGB channel extraction, grayscale conversion, histogram analysis, and image quantization. Evaluation metrics such as MSE (Mean Squared Error) and PSNR (Peak Signal-to-Noise Ratio) were used to assess image quality.

#### 3.3.1 Load an Image

To start the image processing tasks, the first step was to load and display an image file using OpenCV. This enabled easy access to and visualization of the image data. For this project, we used images in PPM, which served as a foundational dataset for further transformations and analyses in the subsequent sections.



Figure 3.15: Loaded Airplane image.

#### 3.3.2 Image Channels and the Grayscale Version

After loading the image, we separated it into its respective RGB color channels — Red, Green, and Blue — using OpenCV functions. Each channel was displayed individually to observe the distribution of color intensities. Additionally, we converted the image to grayscale, creating a single-channel representation. This grayscale version simplifies further processing steps by focusing on intensity values rather than color, which is useful for tasks like histogram analysis and filtering.

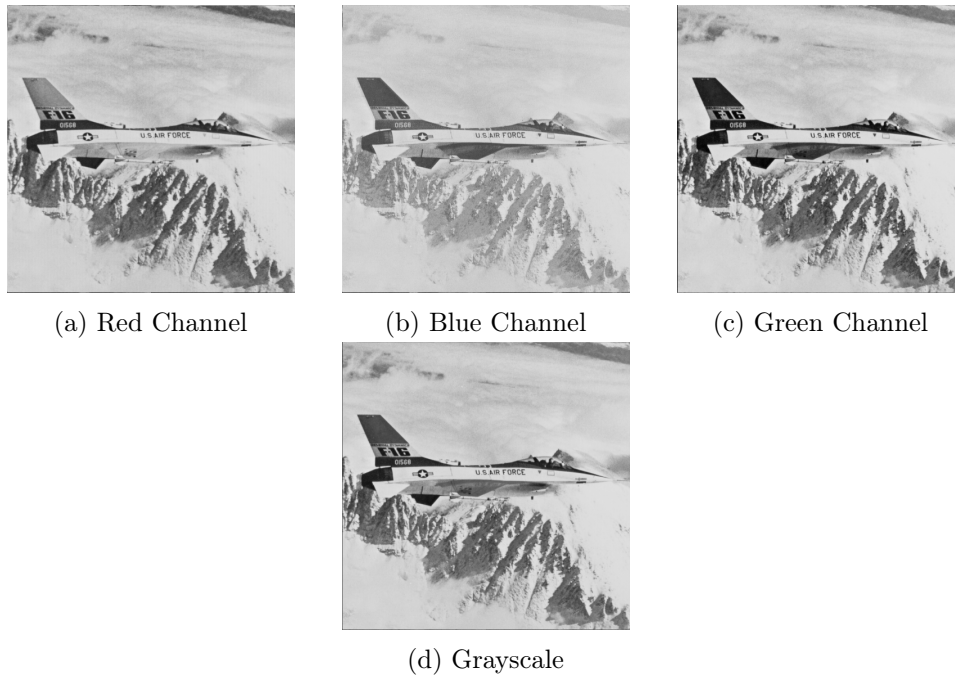


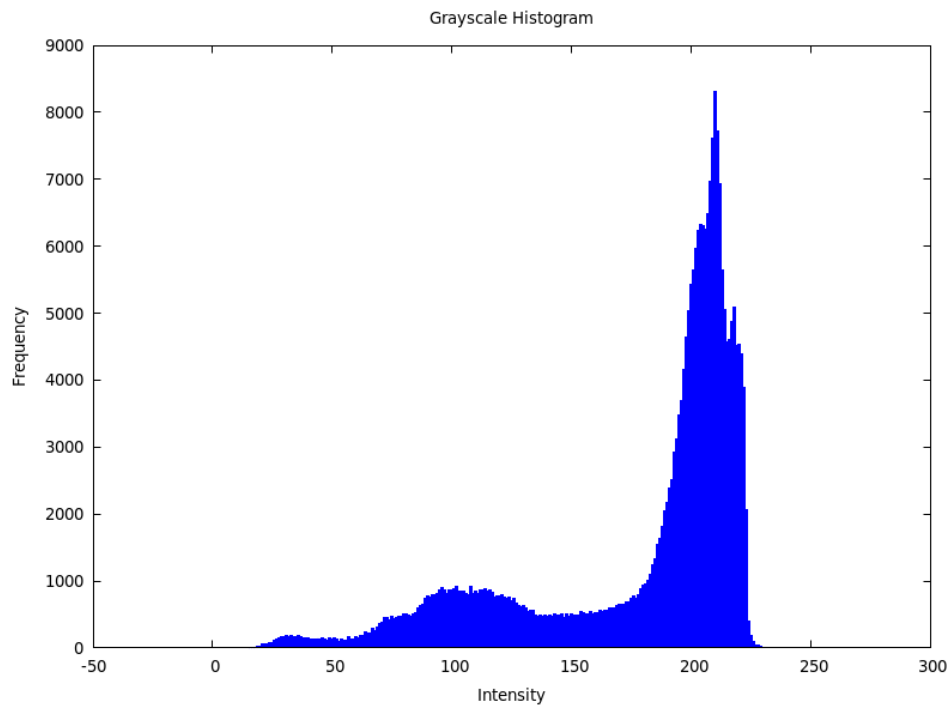
Figure 3.16: Comparison of RGB channels and grayscale version of the image.

### 3.3.3 Distribution of Pixel Intensities

To analyze the distribution of pixel intensities in the image, we calculated the histogram of its grayscale version. This histogram visually represents how pixel values are spread across various intensity levels, providing insights into the image's contrast and brightness. By using OpenCV and Gnuplot, we generated and visualized the histogram, highlighting areas of both high and low intensity concentration.



(a) Grayscale image



(b) Histogram of pixel intensities for the grayscale image

Figure 3.17: Graphical representation of the distribution of pixel intensities.

### 3.3.4 Gaussian Blur Filter with Different Kernel Sizes

To analyze the effects of blurring, we applied a Gaussian blur filter to a grayscale image using various kernel sizes. This filter smooths the image by averaging the pixel intensities within a neighborhood defined by the kernel size. Smaller kernels produced subtle smoothing, while larger kernels produced a more pronounced blur. By comparing images blurred with different kernel sizes, we observed how the degree of blur affects image clarity and detail preservation.

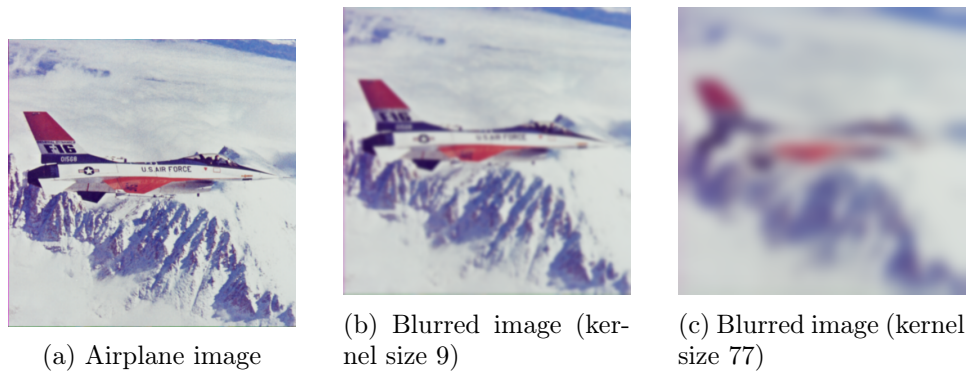


Figure 3.18: Comparison of original image and blurred images with different kernel sizes.

### 3.3.5 Difference Between Two Images

To measure differences between two images, we implemented a function that calculates the absolute difference on a pixel-by-pixel basis. This comparison highlights areas where pixel values vary, providing a clear view of changes between the images. We then calculated two key metrics: MSE and PSNR, which quantify the similarity between images and assess the quality of transformations.

In the code, we translated the MSE formula as follows:

```

1  double calculateMSE(const cv::Mat& img1, const cv::Mat&
    img2) {
2      cv::Mat diff;
3      cv::absdiff(img1, img2, diff);
4      diff.convertTo(diff, CV_32F);
5      diff = diff.mul(diff);
6      cv::Scalar sumDiff = cv::sum(diff);
7      double mse = sumDiff[0] / (double)(img1.total());
8      return mse;
9  }
```

For PSNR, we used the following code:

```

1  double calculatePSNR(const cv::Mat& img1, const cv::Mat&
    img2) {
2      double mse = calculateMSE(img1, img2);
3      if (mse == 0) {
4          return INFINITY; // image 100% identical
5      }
6      double psnr = 10.0 * log10((255 * 255) / mse);
7      return psnr;
8  }
```





Figure 3.19: Visualization of the difference image between two images.

### 3.3.6 Image Quantization

Image quantization reduces the number of distinct intensity levels in an image, simplifying its representation while introducing some loss in quality. We experimented with different quantization levels:

- **Original:** The original image before any alteration.
- **2 levels:** Highly abstract, with only black and white tones, leading to significant detail loss.
- **8 levels:** Good balance between compression and visual quality, preserving most details.

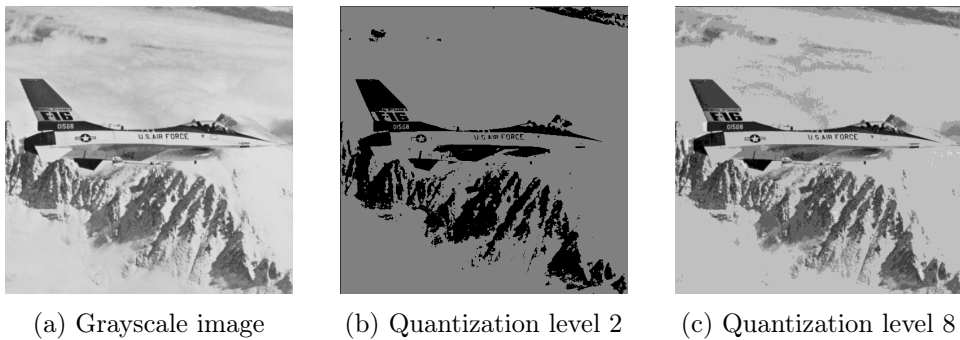


Figure 3.20: Comparison of grayscale image and images with different quantization levels.

Quantization effectively reduces image complexity, trading off some visual quality for data compression.

## Chapter 4

# Discussion

### 4.1 Outcomes, Strengths, and Limitations of the Methods

Text data preprocessing involved key transformations like removing XML tags, converting text to lowercase, and stripping punctuation. These steps were essential for normalizing the data and reducing noise, leading to improved frequency distributions. Notable strengths of the project include the use of visualization tools to analyze text data, with histograms effectively illustrating character and word frequency distributions.

Similarly, the ways in which audio data was loaded and preprocessed gave us a better idea about how different transformations affect the quality of representations. Visualization tools such as Gnuplot and waveform generation helped to understand data distributions and signal behavior. The quantization techniques showed the relationship between file size and quality, which is one of the important factors in data compression.

However, there were also limitations during the project:

- **Audio Format Selection:** One critical limitation was the use of compressed audio formats like MP3 in testing. MP3 files already undergo lossy compression, which introduces artifacts and alters the original waveform. This affected the accuracy of our quantization and quality analysis. For more precise results, it is important to use uncompressed audio formats such as WAV, which retain the full original quality of the audio.
- **Limited Depth in Quality Metrics:** While metrics such as MSE and SNR were effective in analyzing degradation in signal quality, these measurements alone don't always reflect perceived quality. Human auditory perception is more complex, and some distortions may be more noticeable to listeners despite having small numerical errors.

- **Computational Complexity:** Quantizing larger audio files, especially at lower bit depths, proved computationally expensive. Real-time processing could be slow, especially when handling more detailed datasets, such as long audio tracks. Heavy computational issues were also experienced when working with large volumes of text data, particularly during histogram creation and entropy calculations.

## 4.2 Learning Experience: Challenges and Successes

The learning experience during this project was both rewarding and challenging. One of the key challenges was handling large datasets, such as long audio files and multiple text files. Analyzing these datasets and generating detailed histograms, especially for character and word distributions, proved to be computationally expensive, leading to performance bottlenecks.

On the positive side, the integration of visualization tools worked very well. The waveform visualizations and amplitude histograms provided clear, intuitive insights into the nature of the signals and helped to highlight the differences between original and quantized data. Similarly, bar graphs generated with SFML effectively illustrated linguistic diversity in the text, revealing significant differences in character and word usage between Portuguese and English texts. Additionally, collaborating with team members allowed for a diverse range of perspectives and problem-solving approaches, enriching the overall learning process and enhancing the quality of the project outcomes.

## Chapter 5

# Conclusion

In summary, the project provided key takeaways in terms of data manipulation, compression, and quality analysis. Working with quantization techniques offered a practical perspective on how data compression introduces quality loss in audio files. Key lessons learned include the importance of choosing the right file formats for analysis (i.e., uncompressed formats for accuracy), as well as the trade-offs between file size and perceptual quality.

This project also showed how small changes, like changing text to lower-case or removing punctuation, can have a big impact on linguistic diversity metrics. Visualization tools, like SFML-generated graphs, were essential for understanding frequency distributions and comparing the complexity of English and Portuguese texts. Similar to data compression, these transformations reduced artificial diversity, improving the clarity of analysis.

When manipulating data, small changes in bit depth can dramatically alter quality. This was especially apparent in audio quantization, where reducing from 16 to 4 bits resulted in audible distortions.

The use of tools like SFML, Gnuplot, and C++ allowed for an in-depth exploration of data processing and visualization. By expanding future work with more advanced perceptual quality metrics and improving computational efficiency, more refined and optimized approaches to data manipulation and compression can be achieved. Future work could also focus on advanced text quality metrics and optimizing computational efficiency to further explore and compare multilingual texts.