UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELECTRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

Information and Coding (2024/25)

Lab work nº 1 — Due: 28 Oct 2024

Intro

- All programs should be implemented in C++. If you are not familiar with C++, start by taking a look of one of the many available tutorials[1].
- The Standard Template Library (STL) will be also a valuable resource[2].
- The comprehensive C++ reference is available at `https://en.cppreference.com`
- Use a github (or equivalent) repository to manage the development of your software.

Part I - Text data manipulation

The goal of this project is to familiarize yourself with manipulating text data programmatically.

T1 - Read the content of a text file (in C++, reading a text file is typically done using file streams).

- Open and read a text file line by line or as a stream of characters.
- Store the content in a suitable data structure.
- Make sure the file opens successfully, and handle any errors.
- Print the content to check if it has been read correctly.

T2 – Apply some basic transformations on text.

- Before performing statistical analysis, you can normalize the text by applying basic transformations such as:
  - Converting to lowercase: This ensures that case differences don't affect the word counts.
  - Removing punctuation: Punctuation marks are typically ignored when calculating word frequencies.

[1] see, for example, `https://www.tutorialspoint. com/cplusplus/index.htm`

[2] see, for example, `https://www.tutorialspoint.com/cplusplus/cpp_stl_tutorial.htm`

T3 - Counting character frequencies (to calculate the frequency of characters, use a data structure such as `std::map` to count occurrences of each character).

- Iterate through the text and count how many times each character appears.
- Store this information in a data structure.
- Print out the character frequencies.
- Although not mandatory, you can use a plotting library to visualize the frequencies.

T4 - Counting word frequencies

- Tokenize the text into words using whitespace as the delimiter.
- Store the word counts.
- Print out the character frequencies.
- Although not mandatory, you can use a plotting library to visualize the frequencies.

Further extensions

- You can add functionality to handle different encodings (e.g., UTF-8) if working with non-ASCII characters.
- Consider calculating the frequency of n-grams (bigrams, trigrams) for a more complex analysis.

Additional resources

- Text Preprocessing Concepts: Refer to resources on NLP (Natural Language Processing) for more advanced text manipulation techniques.

Part II – Audio data manipulation

The objective of this part of the project is to familiarize yourself with reading and manipulating audio data.

T1 - Reading and loading audio files (eg. using the library SFML - `https://www.sfml-dev.org` or `libsndfile` - `https://libsndfile.github.io/libsndfile/`).

- Load an audio file (preferably a `.wav` file).
- Extract raw audio samples from the file.
- Get basic information about the audio file, such as the sample rate, number of channels, and duration.

T2 - Visualizing the audio waveform (visualize how the amplitude of the sound changes over time).

- Plot the waveform using the audio samples.
- Ensure the x-axis represents time, and the y-axis represents amplitude.
- Compare your results with some open source available programs (eg. `https://www.audacityteam.org`).

T3 - Histogram of amplitude values (The histogram shows the distribution of amplitude values across the entire audio file. It provides insight into how often certain amplitude levels are present in the audio data.)

- Create a histogram of the amplitude values (raw audio samples). Consider the visualization of the left and right channels, as well as the histogram of the average of the channels (the mono version, i.e., (L + R)/2, also known as the MID channel) and the difference of the channels (i.e., (L − R)/2, also known as the SIDE channel)[3], when the audio is stereo (i.e., when it contains two channels).
- Use a bin size that provides meaningful information about the distribution. Instead of having an histogram bin for each different sample value, have bins that gather together 2, 4, 8, . . . , 2^k values.

T4 - Quantization of audio (quantization involves reducing the precision of the audio samples. This process is typically used in lossy compression techniques but will be implemented here to understand its effect on audio quality.).

- Implement uniform quantization to reduce the number of bits used to represent each audio sample (reducing the number of distinct amplitude levels).
- Plot the waveform before and after quantization.

T5 - Comparing two audio samples. A commonly used metric for comparing audio signals is the Mean Squared Error (MSE), which calculates the average squared difference between the corresponding sample points of two signals. Another option is to compute the Signal-to-Noise Ratio (SNR), which quantifies the amount of noise introduced in the processed signal compared to the original.

- Calculate the MSE and SNR between the original and processed audio samples.
- Use these metrics to assess the quality of the quantization.
- Experiment with different quantization levels to see how they affect the quality.

Further Extensions:

- Frequency analysis: Implement a basic frequency analysis using the Fourier Transform to analyze the frequency content before and after transformations.
- Noise addition: Add artificial noise to the audio samples and then compare the noisy and original samples using MSE and SNR.

[3] In both cases, use integer division.

Part III - Image Data Processing

The goal is to work with image data in a similar way to the previous parts on text and audio.

T1 - Reading and loading image files. In C++, you can use libraries like OpenCV (https://opencv.org/) for image manipulation, which makes it easy to read, manipulate, and display images. OpenCV is widely used for computer vision and image processing tasks. Therefore, start by installing this library and look at the tutorials. Note: before trying to download and build the OpenCV library from scratch, find out if there are prebuilt packages for your system.

- Load an image from a file (e.g., PNG or PPM – we will talk about image coding later).
- Display the image using OpenCV functions.

T2 - Visualizing the image channels and the grayscale version of an RGB image.

- Split the RGB image into the corresponding three channels.
- Convert the image from RGB to grayscale.
- Display the color channels and grayscale image and compare it with the original color image.

T3 - Calculating image histogram. An image histogram provides a graphical representation of the distribution of pixel intensities.

- Calculate the histogram of pixel intensities for the grayscale image.
- Visualize the histogram to analyse the distribution of intensity values.

T4 - Applying basic image filters (Image filters are used to enhance or extract features from images).

- Apply a Gaussian blur filter to the image with different kernel sizes.
- Display the blurred image and compare it with the original.

T5 - Calculating the difference between two images.

- Implement a function to calculate the absolute difference between two images.
- Compute and display the Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) between the two images.
- Display the difference image to visually observe where the differences are.

T6 - Image quantization. To reduce the number of bits used to represent each image (i.e., to perform uniform scalar quantization).

- Implement the quantization function on grayscale images.
- Experiment with different numbers of quantization levels.
- Compare the original image with the quantized one using MSE and PSNR to evaluate the quality.

Part IV – Report and peer code review

T1 – Report. The report should not simply be a code walkthrough but instead focus on what the implemented software can achieve, along with a thoughtful analysis of the results. The report must include:

- Introduction:
  - Briefly describe the purpose of the project and the steps carried out in each part (text, audio, and image processing).
  - Summarize the goals of each step (data manipulation, transformation, compression).
- Methodology:
  - For each data type (text, audio, image), describe the key transformations and algorithms applied.
  - Explain why specific methods were chosen (e.g., why quantization was chosen for image compression or why a certain number of levels were used in quantization).
  - Describe the tools and libraries used in the process.
- Results:
  - Text data:
    - Summarize the results from text data analysis, including word frequencies, character histograms, and any transformations applied.
    - Discuss how the transformation (e.g., lowercase conversion, punctuation removal) impacted the word frequency analysis.
  - Audio data:
    - Provide results from the audio manipulation tasks. For example, compare the original and quantized audio, and discuss how quantization impacted the quality using measures such as MSE and SNR.
    - Show waveform visualizations, histograms, and any other relevant plots.
  - Image data:
    - Summarize the results of image manipulations, such as histogram calculations, resizing, and quantization.
    - Include MSE and PSNR comparisons between original and processed images to highlight how the various transformations (e.g., quantization, filtering) affected image quality.
    - Include visualizations such as before-and-after images and difference images.

- Performance analysis:
    - If possible, include processing time for different steps (e.g., time taken for quantization).
    - Discuss the trade-offs between processing time, compression efficiency, and quality loss. For example, "Higher compression ratios resulted in lower PSNR values, indicating a loss in quality."
    - Compare the results across different datasets (e.g., different audio or image files).
- Discussion:
    - Analyze the outcomes and identify the strengths and limitations of the methods implemented.
    - Propose possible improvements or optimizations for future work.
    - Reflect on the learning experience: What was challenging? What worked well?
- Conclusion:
    - Summarize the key takeaways from the project.
    - Highlight what was learned about data manipulation, compression, and image/audio quality analysis.
- Appendices:
    - Include sample code snippets (optional, if necessary to explain results).
    - Add links to repositories or source code for peer review.

T2 - Peer code review

- Review a peer's report and code, providing detailed feedback.
- Reflect on the strengths and areas for improvement in your peer's work.
- Incorporate insights from peer review to improve your own coding and reporting practices.