universidade de aveiro
theoria poiesis praxis

Análise do Grupo 3

Primeiro Mini-projeto IC

Peer review

Unidade Curricular: Informação e Codificação (IC)

DETI

Ano Letivo 2024/25

102690, João Monteiro, P2

103415, João Sousa, P2

107708, João Gaspar, P2

Índice

Visão Geral

A primeira impressão ao abrir a pasta do projeto deste grupo foi a desorganização que a mesma continha. O seu conteúdo todo espalhado sem qualquer tipo de identificação direta. O ficheiro README.md pouco ajudou. Ficámos um pouco “não entendo nada do que se passa aqui” porque, de facto, a desorganização é tanta que a questão “por onde começamos?” era a única coisa que nos passava pela cabeça.

Para terem uma ideia da nossa visão (a do INFERNO):

```
Universidade/Mestrado/g3 via 🐍 v3.10.12
❯ l
 1.1.cpp                    airplane_grayscale.png                  ruido_artificial.cpp
 1.cpp                      airplane.ppm*                           Sample/
 1_python.py                airplane_quantized.png                  sample01_com_ruido.wav
 2.1.1*                     arial.ppm*                              sample01_processado.wav
 2.1.1.cpp                  audio1.sh*                              sample01.wav
 2.1.cpp                    audio.sh*                               samples_output.txt
 2.2.cpp                    combined_histograms.png                 textofrances.txt
 2.3.cpp                    data.sh                                 textofrances.txt_char.csv
 2.4.cpp                    entropies.txt                           textofrances.txt_char_histogram.png
 2.5.cpp                    entropy_characters.png                  textofrances.txt_word.csv
 2.cpp                      entropy_words_bigrams_trigrams.png      textofrances.txt_word_histogram.png
 3*                         entropy_words.png                       textolituania.txt
 3.1.2*                    'ep-00-01-17(1).txt'                     textolituania.txt_char.csv
 3.1.2.cpp                  es.rar                                  textolituania.txt_char_histogram.png
 3.1.3*                     FFT.cpp                                 textolituania.txt_word.csv
 3.1.3.cpp                 'IC_Lab_1_RelatoÌ'$'\302\201''rio.pdf'   textolituania.txt_word_histogram.png
 3.1.4*                     imagem.sh*                              textoportugal.txt*
 3.1.4.cpp                  imagens/                                textoportugal.txt_char.csv
 3.1.5.cpp                  Imagens_Output/                         textoportugal.txt_char_histogram.png
 3.1.6.1.cpp                imagens_quantizadas.sh                  textoportugal.txt_word.csv
 3.1.6.cpp                  makefile                                textoportugal.txt_word_histogram.png
 3.cpp                      quantization                            trab1.pdf
 airplane_blurred_15x15.png README.md
```

Figura do estado da organização

Texto

Task 1

No código fornecido, o ficheiro é aberto corretamente com suporte para UTF-8, o que está bem implementado para a leitura de textos. A leitura é feita linha a linha e o código verifica se o ficheiro foi aberto corretamente, com tratamento de erro em caso de falha.

Task 2

No código, as transformações ao texto são aplicadas corretamente. O texto é convertido para minúsculas e as pontuações são removidas. Apesar disto as tags XML não são removidas , o que leva a um mau tratamento dos dados.

Task 3

O código armazena a frequência de cada caráter, apenas caracteres alfanuméricos são contabilizados, dado que pontuações são removidas previamente.

Task 4

O código faz a tokenização do texto. Palavras comuns são filtradas antes de contar a frequência, o que ajuda a melhorar a análise. A contagem de palavras também é armazenada.

Extensões Adicionais

O código lida corretamente com codificação UTF-8, permitindo a leitura e processamento de textos que podem conter caracteres especiais. Além disso, o código implementa a contagem de bigramas e trigramas.

Resultados

Nos histogramas de palavras, as palavras muito comuns como "de", "la", "ir" têm grande destaque. Para uma análise mais focada nas palavras, poderia ser interessante filtrar mais palavras para reduzir o impacto destas palavras comuns e ver melhor a distribuição das palavras mais relevantes.
Em resumo, os resultados mostram padrões esperados e interessantes sobre a estrutura dos textos em diferentes línguas.

Áudio

Disclaimer: Tenho de fazer make de cada um dos programas para cada tarefa, o que gera muitos arquivos e dificulta a análise do código.

Task 1

Após bastante tempo do início da aula consegui, finalmente, entender como correr o programa após uma leitura do ficheiro “Makefile” visto que, o README.md, que deveria ter as noções básicas para o entendimento do programa, não o ter.
A primeira tarefa está de certa forma repetida em três ficheiros diferentes sendo apenas um usado para a versão final. *2.ccp*, *2.1.cpp* e *2.1.1.cpp* onde apenas o segundo é usado para compilar. Em si, o código parece congruente e apresenta resultados que parecem estar certos.

Task 2

Nesta tarefa, o código também parece estruturado e correto para o que o enunciado pede porém, quando compilo o programa e dou run, fico sem saber a qual amostra de áudio a waveform se refere. Será à primeira, à quarta ou à quinquagésima?
Abaixo está a figura resultante ao correr o programa compilado correspondente à tarefa 2 (sem qualquer tipo de legenda).

Figura relativa à waveform da sample01.wav

Task 3

Na tarefa três o problema persiste. Não há qualquer tipo de indicação para qual a amostra de áudio usada. Na figura abaixo (output), não há qualquer referência a legendas, ou seja,

estou a olhar para o quê? O que representam os eixos? Apesar disto o código parece bem formulado.

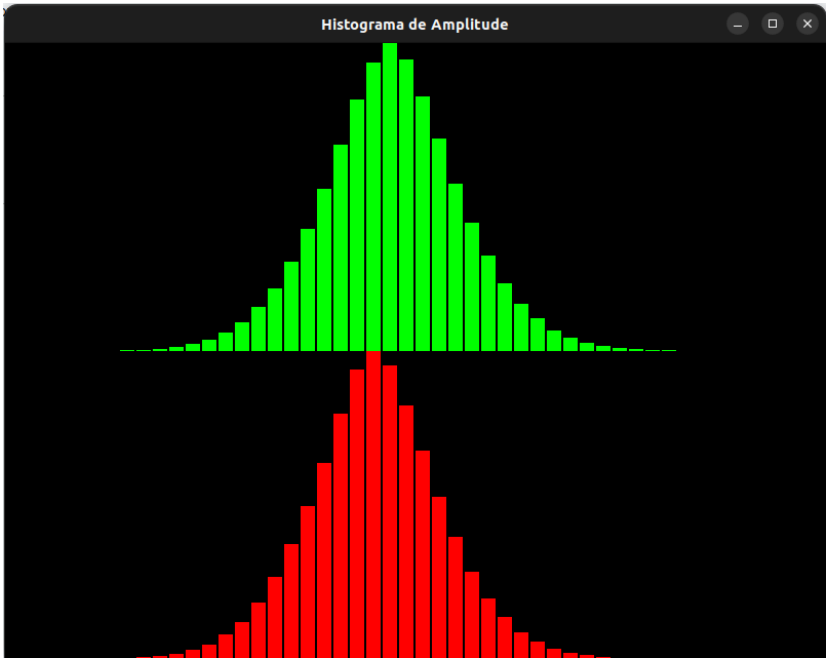

Figura relativa aos histogramas de ambos os canais da sample01.wav

Task 4

O código também parece ok, porém talvez fosse melhor que o próprio utilizador colocasse o número de bits para a quantização. Relativamente aos resultados (figura abaixo), o azul representa a onda original e a vermelha a quantizada. A quantização é de 4 bits. A meu ver e segundo o resultado que obtivemos no nosso projeto, a mesma não está bem feita. A quantização de 4 bits não deveria ser tão redutora.

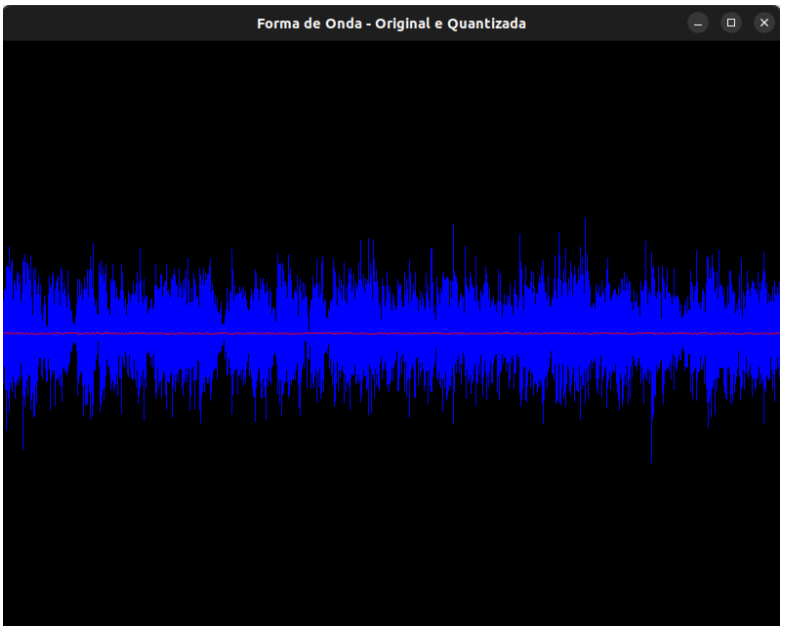

Figura relativa à waveform original (a azul) e quantizada (a vermelho) da sample01.wav

Task 5

Na última tarefa o código está bom, no entanto, não compre com todas as sub-tarefas propostas. Acabaram por não fazer a análise com os diferentes níveis de quantização. Os resultados objetivos para as duas métricas parecem bem obtidos.

Imagem

Disclaimer: Tenho de fazer make de cada um dos programas para cada tarefa, o que gera muitos arquivos e dificulta a análise do código.

Task 1 e 2

O código cumpre corretamente os requisitos da tarefa, como o dar load das imagens, visualizar a imagem e divisão por canais de cor (RGB), obter a imagem em grayscale.
Utiliza a biblioteca OpenCV para desempenhar estas funçÕes e guardar estas imagens numa pasta diferente.

Task 3

O histograma é gerado, mas não tem legendas nem dá opção de selecionar outra imagem para visualizar o histograma sem ser trocando no código o ficheiro que vai ser analisado.

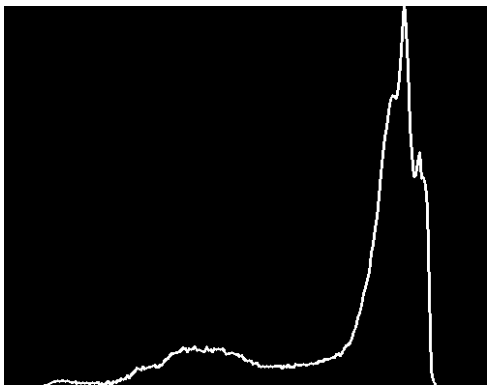

Figura do histograma gerado

Task 4

Conseguiram cumprir a tarefa, embora não haja uma opção dinâmica, sem ser alterando o código, para selecionar a imagem a que queremos aplicar o desfoque e o kernel size.

Task 5

O código cumpre os requisitos pois fornece tanto a comparação quantitativa (MSE e PSNR) quanto a comparação visual através da exibição das imagens, permitindo uma análise clara das diferenças entre as imagens. A implementação é funcional e eficiente, mas novamente faltou a parte de torná-la dinâmica.

Task 6

Esta implementação parece correta, embora não ofereça a opção se apenas aplicar quantização a uma imagem, processa várias e salva-as numa pasta.

```
----------------------------------
Imagem quantizada salva como 'quantizadas/girl/girl_quantized_8.png'
Imagem: girl | Níveis de Quantização: 16
Tempo de Quantização: 4.84943 ms
Erro Médio Quadrático (MSE): 20.7352
Relação Sinal-Ruído de Pico (PSNR): 34.9637 dB
----------------------------------
```

Figura que mostra no terminal a quantização das imagens

Conclusão

O código para o texto cumpre todos os requisitos. Todas as funcionalidades necessárias, como leitura do ficheiro, normalização do texto, contagem de caracteres e palavras, e a extensão para bigramas e trigramas, estão bem implementadas. A contagem de palavras e caracteres e o tratamento de codificação UTF-8 está feito de forma correta.

Embora a lógica seja clara, o código poderia incluir mais comentários explicativos, adicionar documentação sobre o fluxo do código. Poderia também melhorar a organização da pasta para uma melhor compreensão, pois tem 2 ficheiros com códigos quase idênticos por alguma razão.

O código para o aúdio cumpre com a maior parte dos requisitos. A desorganização seria e é um aspeto a melhorar (e bastante). Para além de tornar a leitura e a compreensão lenta, não é uma boa prática em programação. Antes mesmo de compilar e correr o código, existem já ficheiros que resultam do output do mesmo, o que indica que não houve uma “limpeza” da pasta.

O código para a imagem analisada cumpre todos os requisitos propostos nas tarefas da parte III do trabalho, abordando a manipulação e processamento de imagens, embora que, novamente reforçando, de forma muito desorganizada e pouco dinâmica.