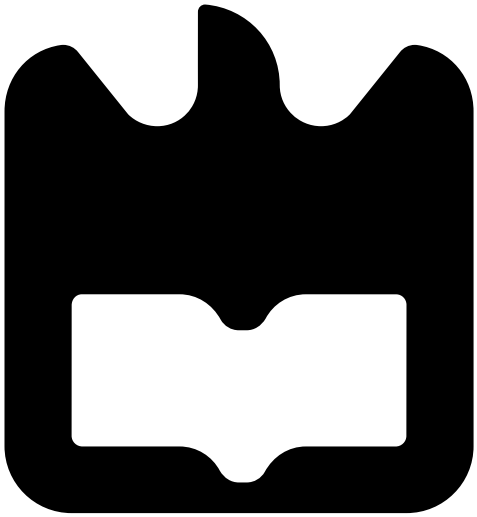

Relatório do Trabalho Laboratorial nº 2

Informação e Codificação (2025/26)

Pedro Miguel Miranda de Melo (114208)
Nome do Aluno 2 (Número Mec.)
Nome do Aluno 3 (Número Mec.)
Departamento de Eletrónica, Telecomunicações e Informática (DETI)
Universidade de Aveiro

Novembro de 2025

Conteúdo

1 Introdução

Este relatório documenta o desenvolvimento e os resultados obtidos no âmbito do Trabalho Laboratorial nº 2 da unidade curricular de Informação e Codificação (2025/26), lecionada no Departamento de Eletrónica, Telecomunicações e Informática (DETI) da Universidade de Aveiro.

O projeto foca-se em duas áreas principais: a manipulação básica de imagens digitais utilizando a biblioteca OpenCV e a implementação de um sistema de codificação entrópica (Codificação Golomb) aplicado à compressão sem perdas (*lossless*) de sinais de áudio e imagem. O desenvolvimento foi realizado em C/C++, complementando as ferramentas desenvolvidas no trabalho anterior com novas funcionalidades.

O código-fonte completo do projeto está disponível publicamente no seguinte repositório GitHub: LINK_PARA_O_VOSSO_REPOSITORIO.

2 Parte I: Manipulação Básica de Imagens com OpenCV

A primeira parte do trabalho consistiu na familiarização com a biblioteca OpenCV através da implementação de programas para realizar operações fundamentais em imagens digitais, manipulando diretamente os seus píxeis. Foi instalada a versão 4.x da biblioteca, utilizando os pacotes pré-compilados disponíveis para o sistema operativo.

2.1 Programa `extract_channel`

Este programa tem como objetivo extrair um canal de cor específico (Azul, Verde ou Vermelho) de uma imagem de entrada, gerando uma imagem de saída em tons de cinza correspondente a esse canal.

2.1.1 Funcionalidade e Utilização

O programa lê uma imagem a cores (representada internamente em formato BGR pelo OpenCV). De seguida, cria uma nova imagem monocromática (`CV_8UC1`) com as mesmas dimensões. Percorrendo a imagem original píxel a píxel, o valor do canal especificado pelo utilizador (0 para Azul, 1 para Verde, 2 para Vermelho) é copiado para a posição correspondente na imagem de saída. A leitura e escrita dos píxeis é feita usando o método `Mat::at<>()`.

A sintaxe de utilização é a seguinte:

```
./bin/extract_channel <imagem_entrada> <imagem_saida> <numero_canal>
```

Listing 1: Sintaxe de Uso do extract_channel

Onde `numero_canal` deve ser 0, 1 ou 2. O formato da imagem de saída deve ser um que suporte imagens monocromáticas, como `.pgm` ou `.png`.

Exemplo de Teste: Para extrair o canal Vermelho (índice 2) da imagem `airplane.ppm` e guardá-lo como `airplane_red.png`:

```
./bin/extract_channel img/airplane.ppm out/airplane_red.png 2
```

2.2 Programa `image_operations`

Este programa engloba um conjunto de operações geométricas e de intensidade sobre imagens, implementadas através da manipulação direta dos píxeis, sem recurso a funções específicas do OpenCV para essas transformações. Foram criados executáveis separados para cada operação para maior clareza.

2.2.1 Negativo da Imagem (`image_negative`)

Esta operação inverte os valores de intensidade de cada canal de cor. Para uma imagem de 8 bits por canal, o valor do píxel negativo P'_{cor} é calculado a partir do original P_{cor} como:

$$P'_{\text{cor}} = 255 - P_{\text{cor}}$$

A operação é aplicada independentemente a cada um dos canais B, G, R.

Utilização:

```
./bin/image_negative <imagem_entrada> <imagem_saida> [view]
```

Listing 2: Sintaxe de Uso do image_negative

O argumento opcional `view` permite visualizar a imagem resultante.

2.2.2 Espelhamento da Imagem (`image_mirror`)

Esta funcionalidade permite espelhar a imagem horizontal ou verticalmente.

- **Espelhamento Horizontal (h):** O píxel na posição (r, c) da imagem espelhada recebe o valor do píxel $(r, \text{largura} - 1 - c)$ da imagem original.
- **Espelhamento Vertical (v):** O píxel na posição (r, c) da imagem espelhada recebe o valor do píxel $(\text{altura} - 1 - r, c)$ da imagem original.

Utilização:

```
./bin/image_mirror <imagem_entrada> <imagem_saida> <h | v> [view]
```

Listing 3: Sintaxe de Uso do image_mirror

O terceiro argumento especifica o tipo de espelhamento ('h' ou 'v').

2.2.3 Rotação da Imagem (`image_rotate`)

Implementa a rotação da imagem por qualquer ângulo múltiplo de 90 graus (positivo, negativo ou zero). O programa normaliza o ângulo fornecido para um equivalente em $\{0, 90, 180, 270\}$ graus no sentido horário e calcula a posição do píxel de origem correspondente a cada píxel da imagem de destino. Para rotações de 90 ou 270 graus, as dimensões da imagem (altura e largura) são trocadas.

Utilização:

```
./bin/image_rotate <imagem_entrada> <imagem_saida> <angulo>
```

Listing 4: Sintaxe de Uso do image_rotate

Onde `angulo` é um inteiro múltiplo de 90.

2.2.4 Ajuste de Intensidade (`image_intensity`)

Permite aumentar ou diminuir o brilho geral da imagem. O programa aceita um valor percentual no intervalo $[-100, 100]$. Este valor é mapeado para um ajuste aditivo A no intervalo $[-255, 255]$:

$$A = \text{round}(\text{percentagem} \times 2.55)$$

Este valor A é somado a cada canal de cor (B, G, R) de cada píxel. A função `saturate_cast<uchar>` garante que o resultado final permaneça no intervalo válido $[0, 255]$.

Utilização:

```
./bin/image_intensity <imagem_entrada> <imagem_saida> <percentagem_ajuste>
```

Listing 5: Sintaxe de Uso do image_intensity

Onde `percentagem_ajuste` é um inteiro entre -100 e 100.

3 Parte II: Classe de Codificação Golomb

Esta parte focou-se na implementação de uma classe C++ para a codificação Golomb, uma técnica de codificação entrópica eficiente para fontes com distribuições geométricas ou de Laplace.

3.1 Classe `GolombCodec`

4 Parte III: Codec Áudio Lossless

Nesta secção, foi desenvolvido um codec de áudio sem perdas (*lossless*), utilizando a codificação Golomb implementada na Parte II para comprimir os resíduos de predição.

4.1 Arquitetura do Codec (`audio_golomb_codec`)

4.2 Análise de Desempenho e Compressão

5 Parte IV: Codec Imagem Lossless (Grayscale)

Aplicando os mesmos princípios da Parte III, foi desenvolvido um codec sem perdas para imagens em escala de cinza.

5.1 Arquitetura do Codec (`image_golomb_codec`)

5.2 Análise de Desempenho e Compressão

6 Conclusões