

Mestrado em Engenharia de Computadores e Telemática  
Departamento de Eletrónica, Telecomunicações e Informática Universidade de Aveiro

# Lab Work Nº2

IC - Informação e Codificação



26/11/2023

Gabriel Negri - 97157  
Marco Magalhães - 101061  
Pedro Sugiyama - 101094

## Parte I - Manipulação de ficheiros de imagem

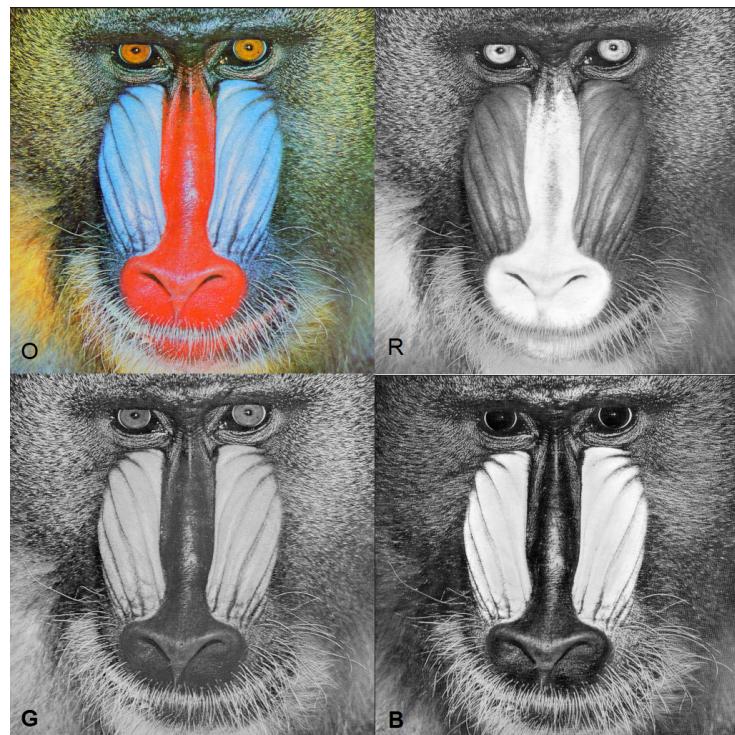
### 1- ppm\_extract

```
./mainExtract < input file > < channel > < output file >
```

O programa **ppm\_extract** possibilita a extração de um dos canais de cor, conforme indicado pelo usuário por meio da variável **<channel>**, de uma imagem fornecida como entrada (**<input file>**). Além disso, o programa permite a criação de uma nova imagem (**<output file>**) que contém exclusivamente o canal extraído.

A indicação do **<channel>** a ser extraído é feita através de uma das opções:

- **-r**: Extração do canal vermelho da imagem.
- **-g**: Extração do canal verde da imagem.
- **-b**: Extração do canal azul da imagem.



O - Imagem original / R - Canal Vermelho extraído

G - Canal Verde extraído / B - Canal azul extraído

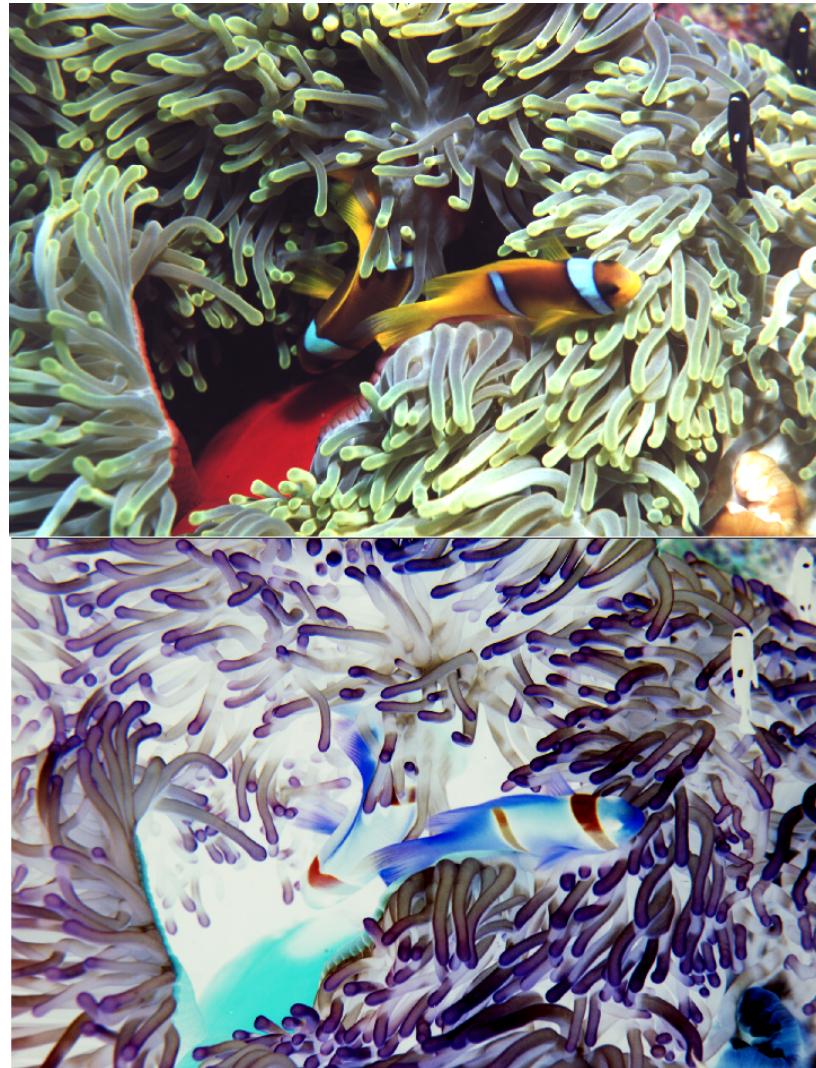
## 2 - **img\_tools**

```
. /mainTools < input file > < tool > [params] < output file >
```

O programa **img\_tools** permite a aplicação de diferentes efeitos em uma imagem passada como argumento (**input file**), além da criação de uma nova imagem (**outputfile**) com o efeito aplicado.

Os seguintes efeitos, **<tool>**, estão disponíveis:

1. **-n**: Cria a versão negativa da imagem.



2. **-m**: Cria a versão espelhada da imagem. É necessário passar um dos seguintes parâmetros:

- a. **H**: Espelhar horizontalmente.
- b. **V**: Espelhar verticalmente



Imagen Original

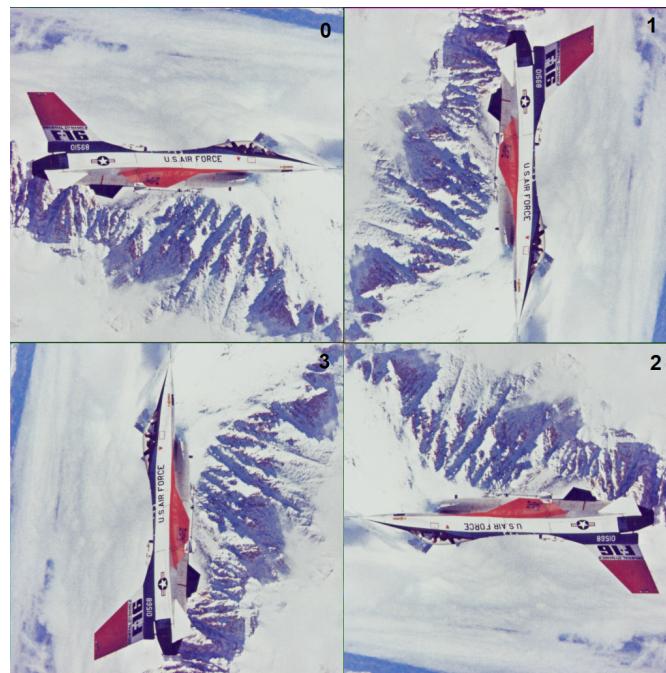


Imagen espelhada Horizontalmente



Imagen espelhada Verticalmente

3. -r: Rotaciona a imagem em múltiplos de 90°, dado o número de rotações como parâmetro (**[params]**).



Representação de 0,1,2,3 rotações

4. -I: Altera os valores de intensidade de uma imagem, dado um fator como parâmetro (**[params]**).

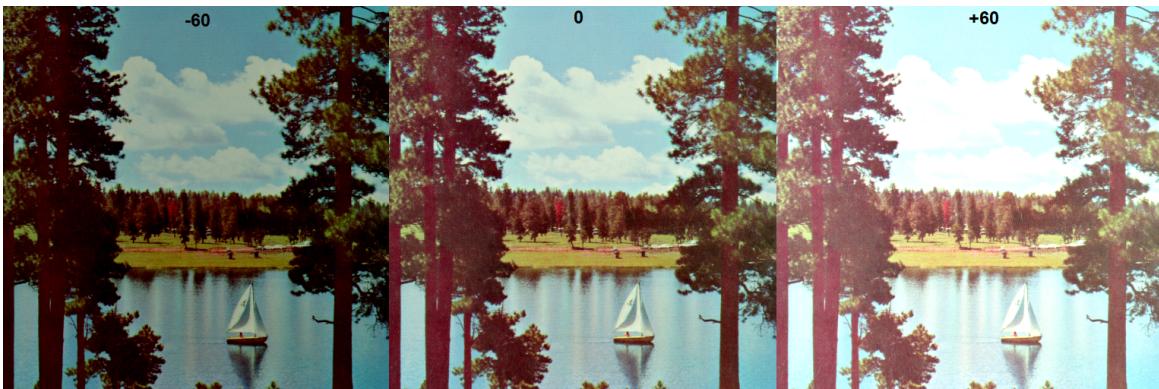


Imagen com fatores: -60, 0, +60.

## Parte II - Codificação de Golomb

### 3 - Golomb

A classe **golomb** dispõe de uma interface para codificação e descodificação de números inteiros baseada na codificação de Golomb.

Para configurar uma codificação/descodificação, basta instanciar um objeto dessa classe com os parâmetros *m* e *mode* desejados.

Os *modes* disponíveis são:

- (0): Sinal e Módulo.
- (1): Valores Positivos/Negativos intercalados.

O parâmetro *m* define o módulo da divisão dos inteiros codificados.

Para verificar o funcionamento dos métodos de codificação/decodificação pode-se utilizar o programa *GolombMain*:

```
./GolombMain < m > < mode > < n >
```

## Parte III - Codecs de áudio

### 4 - sndCodec - Predictive audio Golomb encoding (lossless)

#### a) Implementação básica (módulo m fixo)

O Codec de áudio ainda não está totalmente funcional, mas nesta secção será explicada a lógica usada e o que o grupo pensa ter sido motivo para as falhas.

Para a codificação/decodificação de Golomb dos ficheiros de áudio, foi criada a classe *GolombStream* para representar o ficheiro de áudio codificado.

Para implementar o Codec, foi criada a classe *sndCodec*, que aproveita do código Golomb feito anteriormente, e o programa *sndCodecMain.cpp* para testes.

```
bin/sndCodecMain <operation> <fileIn> <fileOut> <mode> [m]
List of operations:
-e      encode <fileIn> to <fileOut>.
-d      decode <fileIn> to <fileOut>.
List of params:
mode:   (0) -> sign and magnitude (1) -> positive/negative interleaving.
m:      optional golomb modulus parameter.
```

O funcionamento interno do Codec envolve o uso do polinômio presente nos slides para cálculo dos valores de predição, em ordem 3, da seguinte forma:

- $P(0) = 0$
- $P(1) = X(0)$
- $P(2) = 2*X(1) - X(0)$
- $P(n) = 3*X(n-1) - 3*X(n-2) + X(n-3)$

Sendo que  $P(n)$  é o valor previsto para a amostra  $n$ ,  $X(n)$  é o valor efetivo da amostra  $n$  e  $n$  é um inteiro positivo.

Atualmente, a codificação preditiva já estava funcional, produzindo ficheiros binários com tamanhos inferiores aos ficheiros de áudio .wav utilizados para

testes, a depender do valor de m escolhido (em geral 1024 ou 2048). No entanto, a descodificação gerava um processo infinito sem nunca escrever para o ficheiro de áudio.

Isto foi atribuído a erros na função BitStream::hasNext. Na tentativa de consertá-la, outros erros foram introduzidos na escrita para ficheiros binários, de forma a impossibilitar a codificação, e por isso não foi possível progredir na implementação.

### **b) Implementação com m dinâmico**

Devido ao impasse ao qual se chegou na alínea anterior, não houve tempo para implementar a funcionalidade do módulo dinâmico. Uma das possibilidades cogitadas foi iniciar o Codec com um valor de m padrão e armazenar a média (em módulo) dos últimos n valores residuais produzidos pelo preditor. Ao fim de cada um de tais ciclos, o valor de m seria alterado para uma potência de dois mais próxima.

Do lado da descodificação, o processo seria bastante parecido, sendo que o Codec teria que novamente analisar a média a cada n amostras para alterar o módulo da descodificação adequadamente.