EEL 7825 - Projeto Nível II em Cont e Proc de Sinais I Daniel Augusto Figueiredo Collier Exercícios - Aula 3

Desafio:

Na página 171 do livro texto é apresentado o que chamamos de separação de planos de bits de imagens. Em muitos casos pode ser interessante esta separação (ex.: técnicas de marcas d'água, compactação de imagens, etc). O SIP não dispõe de uma função pronta que permita ao usuário separar os 8 planos de bits de uma imagem com 256 tons de cinza. Desta forma, fica o desafio para vocês. Tentem criar uma função utilizando apenas os comandos do Scilab (ou seja, nada de linguagem C) que permita a separação dos 8 planos de bits para imagens em tons de cinza. **DICA:** você poderá fazer uso das funções de operações binária disponíveis no Scilab.

Para fazer a separação dos 8 planos de bits o ideal seria trabalhar com os números convertidos para alguma base na potência de 2 e realizar operações lógicas sobre os números como em linguagem de baixo nível. Pelo que eu procurei de pacotes binários não encontrei nada parecido com em outras linguagens de script como Matlab e Octave.

O que eu fiz foi primeiramente uma função que converte um decimal para uma representação que adotei "binária". Isso porque não defini nenhum tipo novo, o número "binário" de saída na verdade é um vetor em que o primeiro elemento é o bit mais significativo e mascara com zeros até completar 8 bits..

Exemplo:

```
dec2bin(10) = 000010101
```

A partir desse ponto, basicamente foi uma adaptação da função para receber como parâmetro um vetor de pontos e ter como saída um vetor de bits para cada número de entrada:

Exemplo:

dec2bin(1:3) =

- 0. 0. 0. 0. 0. 0. 1.
- 0. 0. 0. 0. 0. 1. 0.
- 0. 0. 0. 0. 0. 1. 1.

A função de separação de bits então é uma função de conversão para binários em que o vetor de entrada é o valor de intensidades da imagem e cada plano é uma coluna de bits em que é feito o processo para voltar a ser uma matriz da dimensão da imagem original.

Para verificar o funcionamento da minha função peguei uma figura na segunda edição do livro (em pdf) que tinha como exemplo na página 88. Assim ficou melhor para comparar os resultados obtidos.

Exemplo:

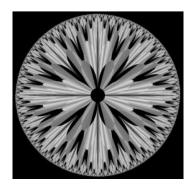


Figura 1: Imagem original.

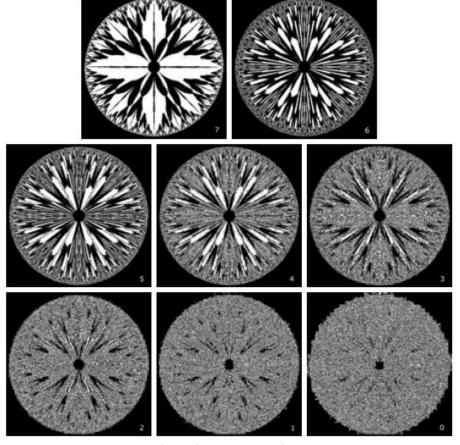


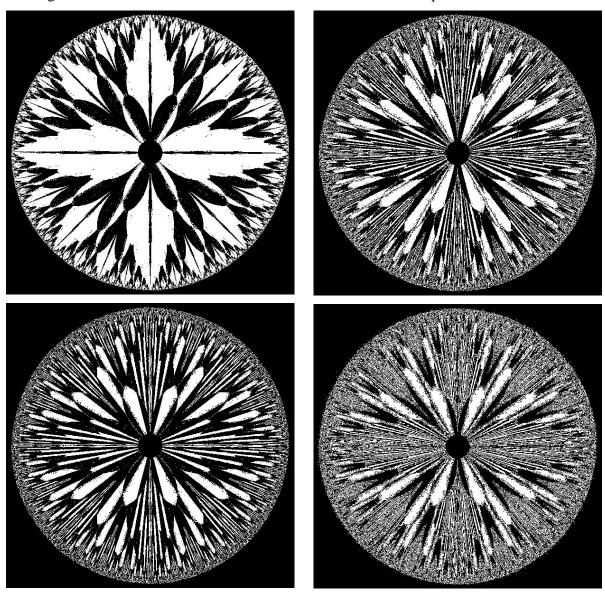
Figura 2: Imagens do livro.

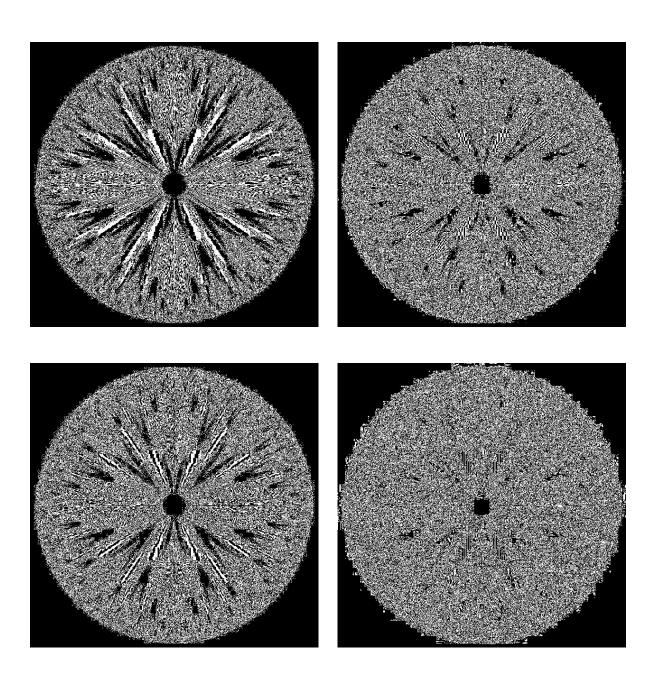
Com a função implementada tive problemas com o ajuste dos valores de entrada. O problema foi ao usar a função *int* para pegar a parte inteira dos valores da matriz, o que na verdade fez ocorrer alguns erros de arredondamento. Para fazer o ajuste correto da entrada fiz:

```
I = imread('fractalg.jpg')-1;
Ou:
I = gray_imread('fractalg.jpg');
I = round(255*I);
```

Fiz diversas comparações para conferir se era possível realmente utilizar das duas maneiras e todas deram que sim.

Na primeira tentativa as imagens não ficaram como o exemplo do livro devido à forma de obtenção da imagem por captura de tela. Então dei uma procurada e achei o site com as imagens do livro e fiz o teste em cima dela. O resultado foi o esperado.





As imagens do livro podem ser encontradas no link:

http://www.imageprocessingplace.com/DIP/dip_image_dabases/book_images_downloads.htm

```
A listagem dos programas:
function B = slicing(I, n)
// SLICING_
// B: plano "n" da imagem
// I: imagem em tons de cinza de 0 a 255
// n: plano da imagem de 0 a 7
// Uso:
// I = imread('figura.jpg');
// I = I -1;
// n = 7;
// B = slicing(I,n);
// imshow(B,[]);
//
s = size(I);
B = dec2bin(I(:));
// divisão do plano
B = matrix(B(:,8-n), s);
endfunction
function b = dec2bin(n)
// DEC2BIN_
// b: vetor com número binário
// n: número em decimal de 0 a 255
//
// Uso:
// b = dec2bin(10);
// b = 0 0 0 0 1 0 1 0
//
n = n(:)';
base = 2i
// mascarar os zeros
mascara = 7;
// divisores
d = 2^{1:mascara};
// quocientes
q = int(n'*(d.\1));
// restos
r = [n' q] - base.*[q zeros(length(n),1)];
// vetor de saída
b = r(:,8:-1:1);
[m n] = size(b);
i = eye(n,n);
k = (ones(n,1)*[d(mascara:-1:1) 1]).*i;
R = (b*k)*ones(n,1);
```

endfunction