Relatório Octave Tarefa 7

Nome: Thiago Heron Albano de Ávila

1. Exercício 1: Modelos de Ruído

1.1 Introdução

Utilizando a função Matlab/Octave imnoise, gere imagens com os seguintes modelos de ruído a partir de uma imagem em escala de cinza de entrada:

- Ruído gaussiano com média 0 e variância 0.01;
- Ruído gaussiano com média 30 e variância 0.01
- Ruído gaussiano com média 50 e variância 0.05
- Ruído com distribuição de Poisson
- Ruído sal-e-pimenta afetando 2% dos pixels
- Ruído sal-e-pimenta afetando 10% dos pixels.

1.2 Desenvolvimento

Nesse exercício não há muito o que ser comentado, a partir de uma imagem de entrada em escala de cinza, utilizado a função **imnoise** passando o tipo de modelo de ruído junto com seus parâmetros solicitados. Infelizmente, novamente não consegui executar esse exercício localmente. Porém, consegui desenvolver pelo Octave Online, e obter os resultados descritos na subseção 1.3.

A seguir, um exemplo do código utilizado para realizar o primeiro teste:

```
I = imread("images/input/lena_cinza.bmp");

I1 = imnoise(I, "gaussian", 0, 0.01);
imshow(I1); title("Ruído Gaussiano, 0, 0.01");
```

1.3 Resultados

1.3.1 Ruído Gaussiano com média 0 e variância 0.01

Ruído Gaussiano, 0, 0.01

1.3.2 Ruído Gaussiano com média 0.30 e variância 0.01

Ruído Gaussiano, .30, 0.01

1.3.3 Ruído Gaussiano com média 0.50 e variância 0.05

Ruído Gaussiano, .50, 0.05

1.3.4 Ruído com Distribuição de Poisson

Ruído com Distribuição de Poisson

1.3.5 Ruído Sal-e-Pimenta afetando 2% dos pixels

Ruído Sal-e-Pimenta 2%

1.3.6 Ruído Sal-e-Pimenta afetando 10% dos pixels

2. Exercício 2: Restauração com Filtro de Mediana

2.1 Introdução

Crie uma função Matlab/Octave que:

- Receba uma imagem em escala de cinzas com ruído sal e pimenta;
- Receba um valor N que representa as dimensões de um filtro NxN;
- Restaure a imagem completa, utilizando um filtro espacial de MEDIANA de tamanho N×N;
- Retorne como resultado o nome do arquivo gerado com a nova imagem filtrada;
- Atenção: a função deve funcionar para qualquer N escolhido, respeitando as dimensões da imagem.

2.2 Desenvolvimento

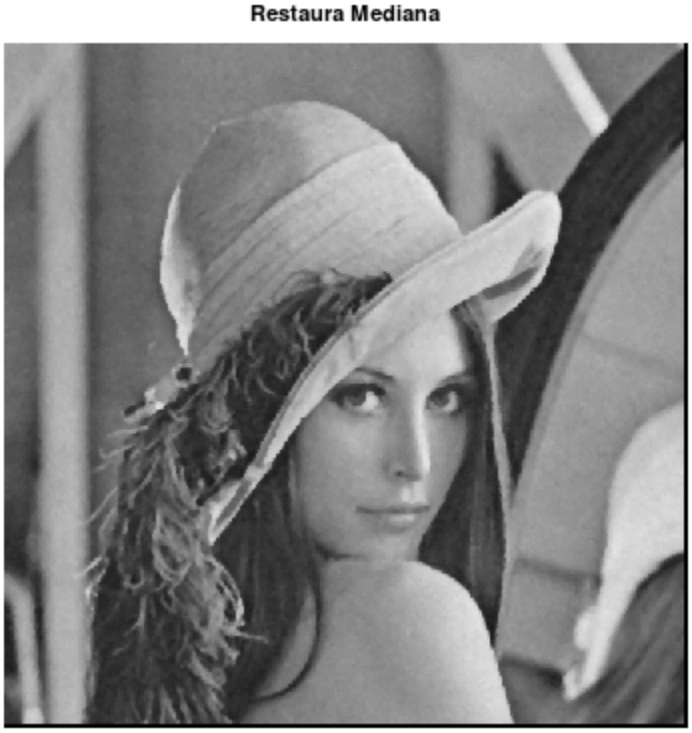
Para o desenvolvimento desse código, basicamente a partir da imagem “lena_sal_e_pimenta.jpg” é utilizado duas estruturas de for aninhados, onde a primeira percorre a imagem de entrada e a segunda armazena os valores a serem aplicados da dimensão do filtro.Por fim, esse filtro espacial da mediana é aplicado sobre a imagem, retornando o nome do resultado.

Esse exercício em específico consegui executar localmente sem a necessidade do Octave Online, porém foi extremamente demorado para um filtro considerado pequeno, **N=3.**

2.3 Resultados

2.3.1 Imagem de Entrada: “lena_sal_e_pimenta.jpg”

2.3.2 Imagem com aplicando filtro da mediana com N = 3.



3. Exercício 3: Quantização de cores com YCbCr

3.1 Introdução

Crie uma função Matlab/Octave com o seguinte formato: A função deve:

- 1. Ler a imagem de entrada
- 2. Converter de RGB para YCbCr
- 3. Salvar os canais Y, Cb, Cr em três matrizes separadas
- 4. Realizar quantização separadamente para cada canal de acordo com o número de bits indicado como parâmetros da função
- 5. Combinar os canais quantizados em uma nova imagem YCbCr
- 6. Retornar a imagem resultante ao espaço de cores RGB
- 7. Apresentar e salvar a imagem quantizada

Assinatura:

Quantiza(entrada, saida, bitsY, bitsCb, bitsCr)

Além disso, realizar o seguinte teste:

- **Teste: 1**. Quantizar com 8 bits em Y, 2 bits em Cb e 2 bits em Cr e verificar a imagem resultante

3.2 Desenvolvimento

Para o desenvolvimento desse exercício, primeiramente teve como entrada a imagem “lena.png” em RGB disponibilizada no Google Drive, após isso aplicado a função **rgb2ycbcr** sobre ela, separando nos três canais Y, Cb, e Cr.

```
YCbCr = rgb2ycbcr(imageInput);
Y = YCbCr(:, :, 1);
Cb = YCbCr(:, :, 2);
Cr = YCbCr(:, :, 3);
QY = (Y.*bitsY);
QCb = (Cb.*bitsCb);
QCr = (Cr.*bitsCr);
imageQuantizada(:, :, 1) = QY;
imageQuantizada(:, :, 2) = QCb;
imageQuantizada(:, :, 3) = QCr;
```

3.3 Resultados

3.3.1 Imagem Original: “lena.png”

Image RGB

3.3.2 Imagem YCbCr

Image YCbCr