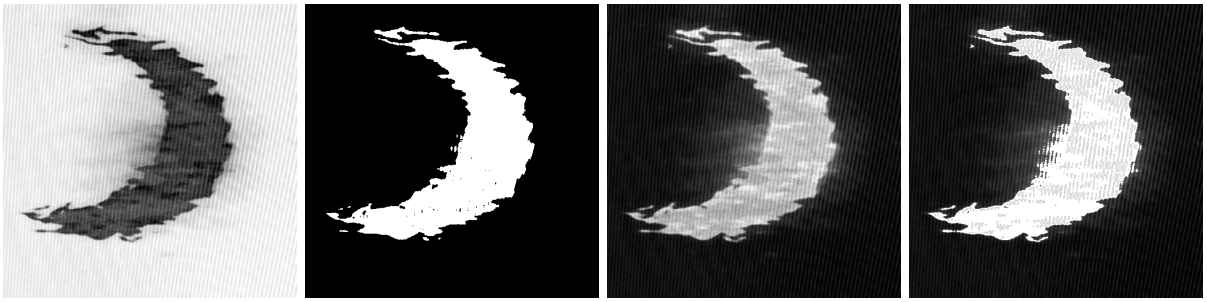
Prática 3 – Transformações e Histograma

—

Ampliando as Possibilidades de Manipulação de Imagens através de Limiarização, Negativos, Potências e Operações Intervalares

Objetivo A

Imagem original | ImBinarizada | ImInvertida | ImFinal



1. **Leia a imagem**:
   Leia a imagem “moon.png” e guarde na variável “im”.
2. **Plotar o histograma:**
   Abra uma janela e mostre o histograma da imagem em um gráfico.
3. **Aplique a limiarização:**
   Faça a limiarização da imagem usando um valor fixo. Após visualizar os resultados, faça a mesma operação para um valor dinâmico inserido pelo usuário. Salve a imagem resultante na variável “imBinarizada”.
4. **Aplique o negativo na imagem original:**
   Operação ponto a ponto que inverte a intensidade de pixels.
5. **Aplicando transformações de intensidade:**
   Aplique a operação de potenciação por “1.3” **somente** na região da lua, ou seja, apenas onde for BRANCO na imagem “imBinarizada”.
   Após visualizar os resultados, inverta a operação, aplicando apenas no fundo.

Objetivo B

Entrada:

Processo:

Saída:

Elabore um algoritmo de PDI que receba as duas imagens de entrada “dance.png” e “dance_depth.png” de uma câmera de captura RGB-D e gere as duas imagens de saída.

Dicas:

1. O histograma será muito útil.
2. Utilize a técnica de fatiamento por níveis de intensidade fixos para chegar no resultado esperado.

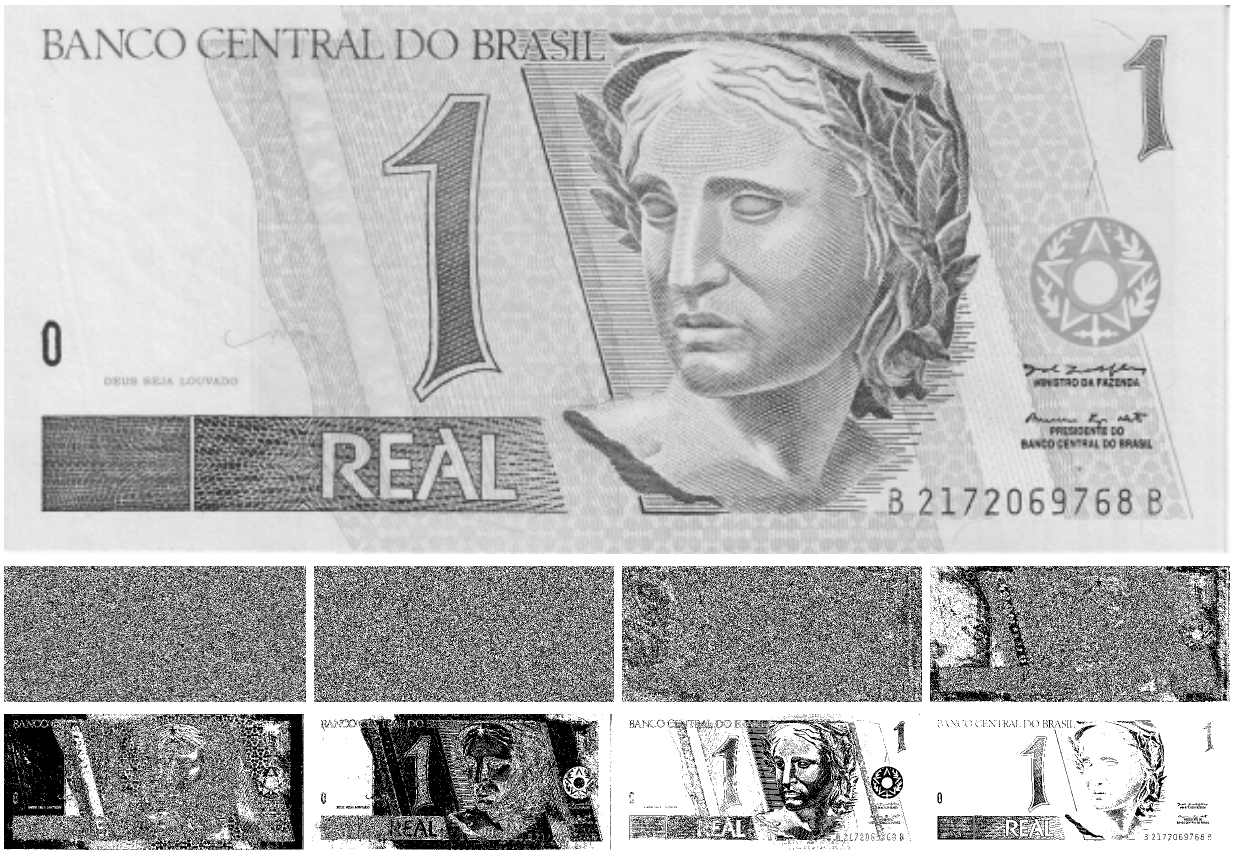
Objetivo C

lake.png

imContraste

1. **Leia a imagem**:
   Leia a imagem “lake.png” e guarde na variável “im”.
2. **Plotar o histograma:**
   Abra uma janela e mostre o histograma da imagem em um gráfico.
3. **Aplique o alargamento de contraste:**
   Criar e mostrar a imagem “imContraste” aplicando em “im” o alargamento de contraste.
4. **Exibir os resultados:**
   Mostre todas as imagens em uma nova janela.

Desafio 02 - Fatiamento bit-a-bit

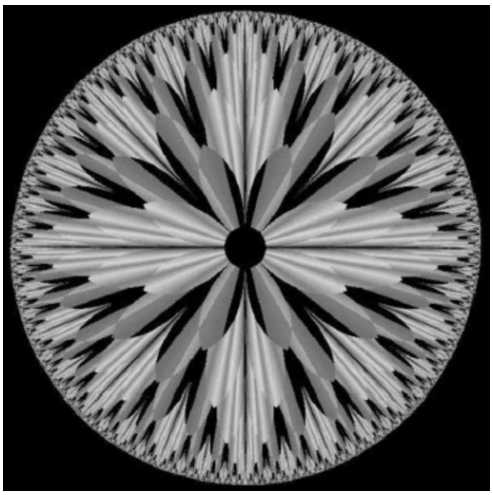


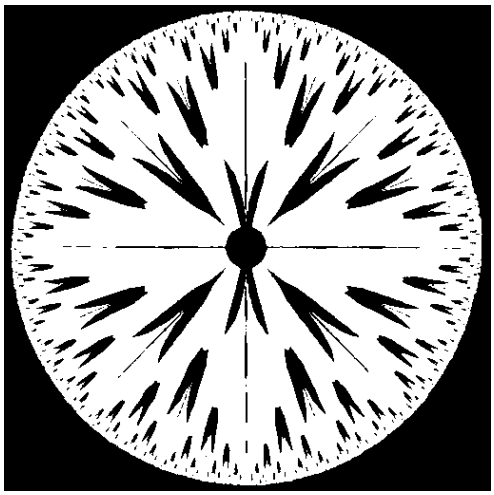
O fatiamento bit-a-bit é uma técnica usada para separar os valores dos pixels em uma imagem de acordo com os seus bits individuais. Em outras palavras, cada pixel é representado por um conjunto de bits, e o fatiamento bit-a-bit envolve separar esses bits para análise ou processamento.

Como Funciona:

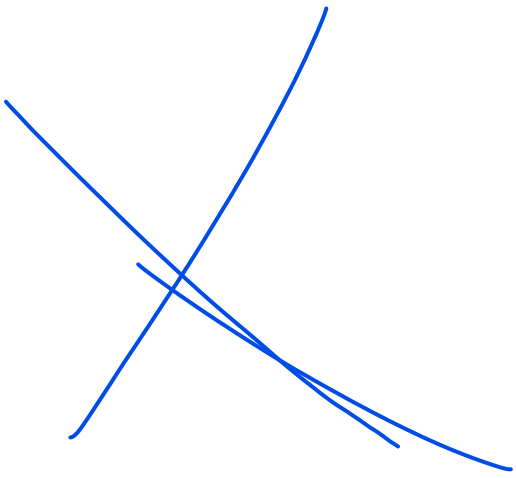
1. **Bits de Pixel**: Cada pixel em uma imagem digital é representado por um conjunto de bits. Por exemplo, em uma imagem em tons de cinza, cada pixel pode ser representado por 8 bits, variando de 0 a 255.
2. **Separação de Valores**: Ao definir limites dentre as fatias da imagem, por exemplo os bits 6 e 7 da imagem “real.png”, a imagem pode ocupar ¾ a menos de espaço e ainda assim conservar os detalhes mais importantes da imagem original.
3. **Aplicações**: O fatiamento bit-a-bit tem várias aplicações. Pode ser usado para realçar detalhes sutis em uma imagem, separar regiões com intensidades similares, ou até mesmo segmentar objetos com base em características específicas.

fractal.png **imSaida**





1. **Leia a imagem**:
   Leia a imagem “fractal.png” e guarde na variável “im”.
2. **Decomposição Bit a Bit:**
   Realize a decomposição da imagem em 8 imagens diferentes através do fatiamento bit a bit e exiba todas as imagens resultantes.[1]
3. **Geração de imSaida:**
   Crie a imagem "imSaida" utilizando os planos de bits mais significativos 7 e 8.
4. **Exibir os resultados:**
   Mostre todas as imagens em uma nova janela.



[1] Para isso você deve transformar o valor do pixel de decimal para binário. E em seguida bolar alguma maneira de pegar cada bit individualmente.