

Trabalho 1

v2 (atualizada 15/03/2018)

Desenvolva seu código sem olhar o de colegas. Plágio não será tolerado.

Gerador de imagens

Nesse trabalho você deverá implementar um gerador de imagens por meio de funções. Você deverá implementar usando **python** com as bibliotecas **numpy** e **random**.

Leia com calma as instruções de cada etapa.

Seu programa deverá permitir ao usuário fornecer parâmetros para a geração automática de imagens. Para isso, o programa deverá:

1. Receber os parâmetros:

- a) nome do arquivo com a imagem referência,
- b) tamanho lateral da imagem da cena C (a imagem da cena será quadrada portanto terá tamanho $C \times C$),
- c) tipo de função a ser utilizada f (1, 2, 3, 4 ou 5);
- d) parâmetro Q ;
- e) tamanho lateral da imagem digital N (a imagem digital será quadrada portanto terá tamanho $N \times N$), sendo $N \leq C$;
- f) número de bits por pixel B a considerar na etapa de quantização, sendo que $1 \leq B \leq 8$;
- g) semente S a ser usada nas funções aleatórias.

2. Gerar a imagem da cena, f , conforme a escolha do tipo de função e parâmetros;

3. Gerar a imagem digital, g , com amostragem definida por N e quantização definida por B ;

4. Comparar a imagem gerada com a imagem referência;

5. Exibir na tela a raiz do erro médio quadrático entre as duas imagens.

Imagem cena, imagem digital

Imagem cena : as funções a serem usadas para geração da imagem na cena f são

1. $f(x, y) = (x + y)$;
2. $f(x, y) = |\sin(x/Q) + \sin(y/Q)|$;
3. $f(x, y) = \left\lceil (x/Q) - \sqrt{y/Q} \right\rceil$;
4. $f(x, y) = \text{rand}(0, 1, S)$:

A função aleatória deve ser uniforme entre 0 e 1, e usa uma semente S inicializada uma única vez antes de sortear o primeiro número.

5. $f(x, y) = \text{randomwalk}(S)$,

A semente S é inicializada uma única vez antes de sortear o primeiro número. Então, considere que $f(x, y) = 0$ para todo x, y . A função de passo aleatório inicia atribuindo o valor 1 à posição $(x = 0, y = 0)$, ou seja $f(0, 0) = 1$. A seguir, são dados passos aleatórios, primeiramente em x , sorteando um número inteiro dx entre -1 e 1 , e atribuindo $x = [(x + dx) \bmod C]$, bem como $f(x, y) = 1$. Em seguida realiza passo em y , sorteando dy entre -1 e 1 e atribuindo $y = [(y + dy) \bmod C]$ e então $f(x, y) = 1$. Note que é importante empregar o módulo para evitar sair dos limites da matriz.

O número total de passos deverá ser $1 + (C \cdot C)/2$

Usar o pacote `random` para o sorteio dos valores; A imagem de cena f deve ser computada usando valores reais.

Amostragem e quantização : a seguir, a imagem da cena deverá ser “digitalizada”, gerando uma matriz g com elementos inteiros de 8 bits. Essa matriz deverá ter dimensão lateral de forma que $N \leq C$. Além disso, deverá ser calculada de forma a considerar diferentes valores de bits por pixel, aceitando valores de bit por pixel entre 1 e 8. Como a resolução de g pode ser menor do que a de f você deverá calcular g utilizando a operação de máximo local. Exemplo, considere uma matriz com $C = 4$:

$$\begin{bmatrix} 5 & 15 & 36 & 0 \\ 18 & 0 & 0 & 1 \\ 0 & 100 & 154 & 0 \\ 0 & 99 & 159 & 100 \end{bmatrix}$$

Para $N = 2$, teríamos:

$$\begin{bmatrix} 18 & 36 \\ 100 & 159 \end{bmatrix}$$

Essa operação pode também ser descrita por: $g(i, j) = \max[f(x, y)]$ para $x, y = (i \cdot d), \dots, (i \cdot d) + d$.

Onde d é uma variável que define quantos pixels na direção vertical e horizontal serão considerados para realizar a operação de máximo. Por exemplo, para saber, no exemplo anterior o valor do pixel $(1, 0)$ (linha 1, coluna 0) da imagem digital, teríamos

$$g(1, 0) = \max [f(x, y)] \text{ para } x = (1 \cdot d), \dots, (1 \cdot d) + d - 1 \text{ e } y = (0 \cdot d), \dots, (0 \cdot d) + d - 1.$$

A variável d pode ser calculada como a quantidade relativa de pixels nas direções vertical e horizontal entre a imagem cena e a imagem digital. Assim podemos fazer: $d = C/N = 2$, e:

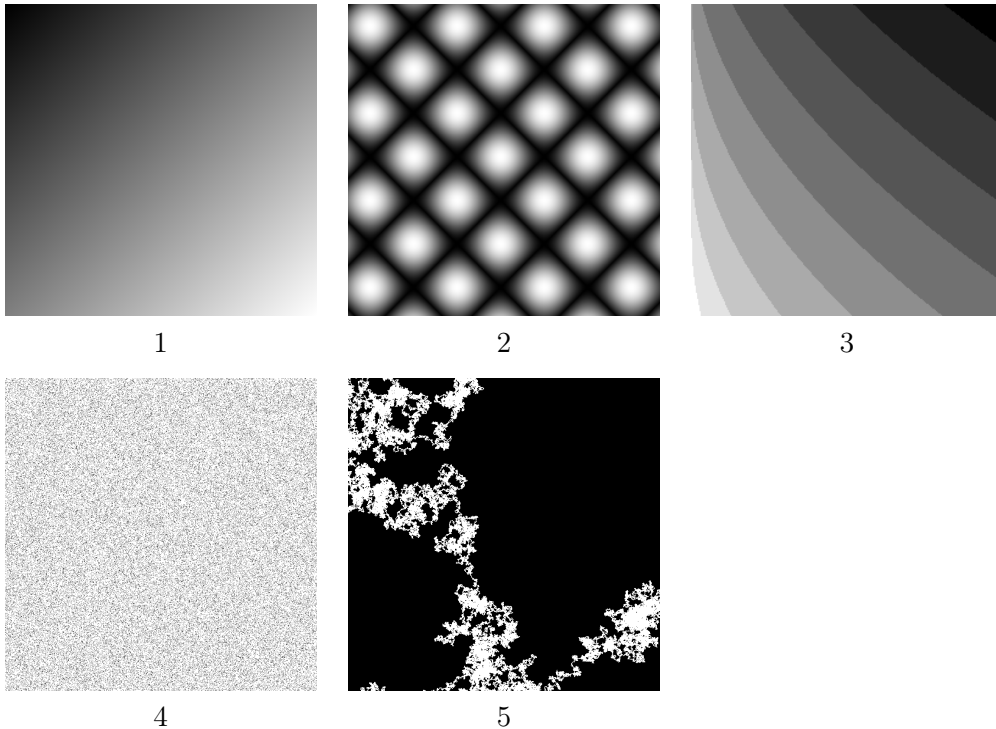
$$g(1, 0) = \max [f(x, y)] \text{ para } x = (1 \cdot 2), \dots, (1 \cdot 2) + 2 - 1 \text{ e } y = (0 \cdot 2), \dots, (0 \cdot 2) + 2 - 1.$$

Resolvendo para $i = 1$ e $j = 0$: $g(1, 0) = \max [f(2 : 3, 0 : 1)]$, o que quer dizer obter o máximo na submatriz relativo as linhas de 2 a 3 e nas colunas de 0 a 1:

$$\max \left(\begin{bmatrix} 0 & 100 \\ 0 & 99 \end{bmatrix} \right) = 100.$$

Além disso, note que f pode conter valores muito maiores do que 2^8 . Assim, você deverá quantizar a imagem realizando uma operação de deslocamento de bit. Para isso, primeiro converta os valores de f para um inteiro de 8 bits de forma que o valor máximo em f passe a ser $(2^8) - 1 = 255$ e o restante dos valores sejam escalados da mesma maneira. Então realize deslocamento dos bits de forma que sejam úteis apenas os B bits menos significativos por pixel, sendo o restante preenchido por zeros.

Veja exemplos de figuras geradas pelas 5 diferentes funções (OBS: imagens com menos do que 8 bits/pixel foram normalizadas para 0-255 para melhor visualização)



Comparação

Você deverá carregar uma imagem de referência R para comparar com a imagem g gerada pelo seu programa. A comparação será feita seguindo a equação raiz do erro médio quadrático (RMSE). O erro deverá ser impresso na tela, com arredondamento para 4 casas decimais.

$$RMSE = \sqrt{\sum_i \sum_j (g(i,j) - R(i,j))^2}$$

A imagem de referência R está armazenada no formato de matriz `numpy`. Você deverá carregar e converter para o formato `uint8` para garantir a comparação. Por exemplo:

```
import numpy as np

filename = str(input()).rstrip()
R = np.load(filename)
```

Submissão e instruções

No sistema Run.Codes deve incluir apenas o arquivo `.py`

1. **É obrigatório comentar seu código.** Como cabeçalho insira seu nome, número USP, código da disciplina, ano/semestre e o título do trabalho. Haverá desconto na nota caso esse cabeçalho esteja faltando, além de descontos para falta de comentários.
2. **É obrigatório organizar seu código em funções.** Utilize uma função para cada tipo de imagem diferente a ser gerada (1,2,3,4,5).