**Introdução ao Processamento de Imagens**

**Trabalho 1**

**21/08/2024**

# **Introdução.**

# Neste trabalho você explorará as técnicas de processamento de imagens discutidas em aula. As tarefas incluirão perguntas conceituais, atividades de programação e discussão de resultados. O trabalho deve ser feito individualmente.

# **Operações Básicas com Imagens.**

## Usando a biblioteca Opencv **cv2.imread()**, implemente a função read\_img(path, grayscale=True) no arquivo common.py. A função deve carregar uma imagem a partir do caminho (path) especificado e, opcionalmente, convertê-la para escala de cinza. A imagem carregada deve ser retornada como um array NumPy com valores do tipo float normalizado para o intervalo [0, 1].

## Usando a função cv2.imwrite() da biblioteca Opencv, implemente no arquivo common.py a função save\_img(img, path). Ela deve rescalar o conteúdo da imagem para valores entre 0 e 255 e então salva-la no caminho (**path**) especificado.

## Usando a função read\_img(), leia a imagem lenagray.jpg. Defina uma nova imagem J, 256 × 256 da seguinte forma: a metade esquerda de J, ou seja, as primeiras 128 colunas, devem ser iguais à metade direita da imagem lenagray.jpg. A metade direita de J, ou seja, as colunas de 128 colunas restantes, devem ser iguais à metade esquerda da imagem do lenagray.jpg. Salve a imagem modificada usando save\_img().

## Usando a função read\_img(), leia a imagem lenacolor.jpg. Defina uma nova imagem J, com as mesmas dimensões da imagem original. Faça com que o canal vermelho da nova imagem seja igual ao canal azul da imagem original, faça o canal verde da nova imagem igual ao canal vermelho da imagem original, e faça o canal azul da nova imagem igual ao canal verde da imagem original. Salve a nova imagem usando save\_img().

# **Transformações de Intensidade.**

## Matematicamente, que a transformações usamos para melhorar o contraste de áreas escuras de imagens? Explique o funcionamento.

## Implemente no arquivo transforms.py a função gamma(img, lambda). Esta função deve ler uma imagem usando read\_img() e retornar a imagem transformada e normalizado. Teste a função usando a imagem acromáticas de sua escolha e realize o salvamento com save\_img() definidas no arquivo **common.py**. Imprima e discuta os resultados em seu relatório.

## Matematicamente, qual é a função de transformação de intensidade que usamos para gerar negativos de imagens? Explique o seu funcionamento.

## Implemente no arquivo transforms.py a função negative(img). Esta função deve ler uma imagem usando read\_img() e retornar o seu negativo normalizado. Teste a função usando a imagem lenacolor.jpg e realize o salvamento com save\_img() definidas no arquivo **common.py**. Imprima e discuta os resultados em seu relatório.

## Por que correções de brilho no espaço RGB podem ser problemáticas? Qual o espaço de cor mais apropriado?

## Como podemos interpretar o histograma de uma imagem em tons de cinza? Explique quando podemos inferir que uma imagem tem baixo contraste observando seu histograma.

## Qual a finalidade da especificação de histograma. Explique matematicamente o procedimento de transformação da imagem de entrada.

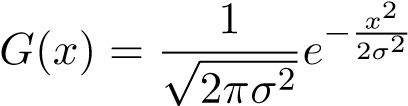
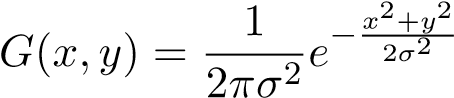
## Qual a finalidade da equalização de histogramas? Qual é a função de transformação de intensidade envolvida? Explique o procedimento.

## No arquivo transforms.py Implemente a função histeq(img). Essa função recebe uma imagem como entrada e retorna a imagem equalizada. Se a imagem for colorida a equalização deve ser feita apenas na intensidade no espaço HSI. Para esse exercício é proibido usar **funções numpy** ou **Opencv** integradas para calcular histogramas ou realizar a equalização. Para fins de teste, utilize duas imagens de sua escolha, uma colorida e uma em escala de cinza, ambas com baixo contraste. Compare os resultados da sua implementação com os obtidos com cv2.equalizeHist(), plotando-os lado a lado.

## Explique o procedimento para realizar a especificação de histogramas de imagens.

# **Filtragem Espacial.**

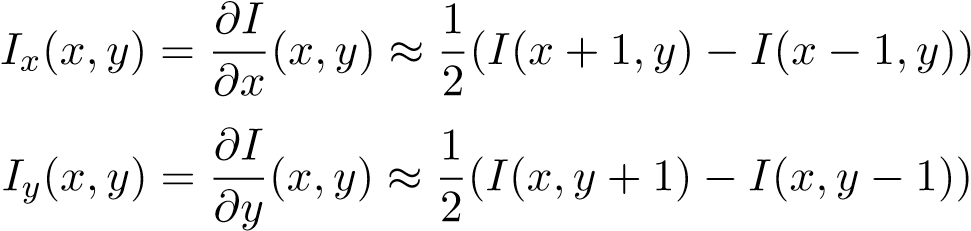
## Os filtros espaciais gaussianos 1D e 2D são derivados das funções gaussianas 1D e 2D especificadas abaixo. Para um filtro gaussiano 2D com uma dada variância σ2, a convolução pode ser reduzida por operações sequenciais de um kernel 1D. Prove que uma convolução por um filtro Gaussiano 2D é equivalente a convoluções sequenciais de um filtro Gaussiano 1D vertical e horizontal. Especifique a relação entre o filtro gaussiano 2D e 1D, especialmente a relação entre suas variâncias.

1D kernel : 2D kernel :

## Implemente a função convolve(img, kernel2d, padd=False) no arquivo filter.py que realiza a convolução da imagem com o kernel 2D fornecido. Compare seus resultados com os obtidos com cv2.filter2d()

## Teste a implementação da função convolve() com a imagem gracehopper.png e um kernel gaussiano 3 × 3. Plote as imagens de saída em seu relatório. Descreva de forma sucinta o que a filtragem gaussiana fez com a imagem.

## Considere imagens como funções *I*(*x,y*): R2 →R. Quando trabalhamos com detecção de bordas (veremos a frente no curso), precisamos prestar atenção às aproximações das derivadas:



Construa kernels de convolução com base nas aproximações acima: (i) , (i) . Implemente a função **edgedetection(img)** no arquivo filter.py. Ele deve retornar a magnitude do gradiente. As componentes devem ser calculadas com os kernels e . Teste a função **edgedetection()** com a imagem **lenagray.jpg**, salve os resultados e os discuta em seu relatório.

## Os filtros de Sobel computam uma aproximação para a derivadade uma imagem. Implemente a função **sobel(img)** no arquivo **filters.py.** Teste a função **sobel()** com a imagem **lenagray.jpg** salve os resultados e discuta em seu relatório. Compare com os resultados obtidos no exercício 4.4.

## Chamamos de *steerable filter* um operador que computa a derivada em qualquer direção.

* 1. Revise o conceito de derivadas direcionais e derive um kernel que detecta transições em termos de um ângulo *α*, tomando como base o grandiente aproximado obtido pelos kernels de Sobel.
  2. Implemente a função steerablefilter(img, alpha) em **filters.py**. Teste sua função com  e a imagem **gracehopper.png**. Salve os resultados e discuta em seu relatório.