

Processamento de Imagens



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



Transformações e Filtragens de Imagens

Responsável pelo Conteúdo:

Prof. Me. Josivan Pereira da Silva

Revisão Textual:

Prof.^a M.^a Sandra Regina Fonseca Moreira

UNIDADE

Transformações e Filtragens de Imagens



- Processamento de Imagens;
- Introdução ao Domínio da Frequência.



OBJETIVO DE APRENDIZADO

- Entender os fundamentos do processamento de imagens relacionados aos algoritmos clássicos, convolução, *kernel*, filtros passa-baixa, passa-banda, passa-alta, média e mediana. Números complexos, ângulo, fase, frequência, espectro, Transformada de Fourier e DFT, FFT.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:



Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Processamento de Imagens

A imagem do *link* a seguir é definida como uma matriz de *pixels*. E o que é *pixel* mesmo?

Pixel significa *Picture Element*, ou seja, o menor elemento de uma imagem que permite operações. A Figura a seguir define uma imagem formada por uma matriz de *pixels*.



Estrutura de uma imagem: matriz de números que significam cores alocadas nos *pixels*, disponível em: <https://bit.ly/2042xvS>

O *link* anterior mostra, nesse caso, uma imagem de 35x35 *pixels* com valores pretos ou brancos.

O que muda dessa imagem em preto e branco para uma imagem colorida?

Essa imagem tem apenas um canal e cada posição (x, y) permite uma cor branca ou preta. Na imagem colorida, cada *pixel* pode ser formado por uma composição RGB que permite muitas possíveis cores.

O *link* a seguir mostra a estrutura de uma imagem colorida com 3 canais RGB.



Estrutura de uma imagem colorida, existem valores em RGB para cada *pixel*. Disponível em: <https://bit.ly/3olorBh>

Acontece que eu posso mudar esses valores de *pixel*, tanto na imagem de 1 canal quanto na de 3 canais (só a forma de inserir o valor muda).

Para uma imagem onde eu tenho 0 ou 1 como cores possíveis, pode ser que eu consiga colocar 0.5, por exemplo, e essa cor já diferencia.

E eu posso fazer cálculos para alterar todos, ou parte dos *pixels* de uma imagem de modo automatizado. Esse é o chamado Processamento de Imagens.

Vou demonstrar essa teoria de forma prática, utilizando o HTML5.

Se eu colocar todos os 3 valores RGB de 1 *pixel* iguais e fazer isso para todos os *pixels* de uma imagem, a tendência é que ela fique em tons de cinza.

Dá até para fazer uma conversão de colorida para tons de cinza utilizando essa ideia. Vamos entender isso do início...

Você já sabe como uma cor é representada no computador, no modelo RGB por exemplo. No modelo RGB, se definirmos esse valor para um *pixel*: (200, 0, 0) ele terá a cor Vermelha, se definirmos a cor (0,0,0) será Preta e se definirmos (255,255,255) teremos a cor Branca, agora repare que todos os valores RGB desligados são equivalentes ao Preto e todos têm o mesmo valor, ou seja, Zero. Repare que se ligarmos todos os valores RGB temos Branco e todos os valores continuam iguais, só que agora são Preto. Acontece que entre Branco e Preto temos o Cinza, ou seja, uma mistura de Branco com

Preto. Na prática, se deixarmos todos os valores RGB iguais, mas diferentes de 0 ou 255, teremos cinza, por exemplo: (50,50,50) ou (120,120,120) e por aí vai.

Podemos fazer cálculos para mudar um pouco os valores RGB dos *pixels* e mudarmos a aparência da imagem de entrada, transformando-a em outra imagem.

Quando essas operações com *pixels* (mudança de cor de *pixels*) com base em cálculos são feitas por todos os *pixels* de uma imagem, acabamos mudando completamente a aparência da imagem e esse é o conceito de processamento de imagem, que é muito utilizado para melhorar as informações contidas nas imagens de maneira a facilitar a extração de informação (MACIEL *et al.*, 2015). Veremos dois exemplos disso: uma conversão de imagem colorida para tons de cinza e uma binarização (conversão para preto e branco) de imagem colorida.

A Figura 1 mostra a conversão da imagem colorida para tons de cinza.



Figura 1 – Processamento de imagem que converte uma imagem colorida em tons de cinza

Fonte: Reprodução

Para transformar uma imagem colorida em uma imagem em tons de Cinza, podemos pegar os valores RGB de cada *pixel* e mudarmos pela média dos três valores, por exemplo, se um *pixel* tem os valores (130, 10, 156) a média seria aproximadamente 99, então mudaríamos os valores do *pixel* em questão para (99,99,99). Fazendo isso com todos os *pixels* da imagem, ela seria convertida para uma imagem em tons de Cinza, como mostra a Figura 1.

A partir da Imagem em tons de Cinza, podemos obter a imagem em Preto e Branco. Primeiro convertemos a imagem colorida em tons de Cinza e depois em Preto e Branco.

A Figura 2 mostra a conversão da imagem colorida para preto e branco.



Figura 2 – Processamento de imagem que converte uma imagem colorida em imagem preto e branco

Fonte: Reprodução

Para conseguir o resultado da Figura 2, pegamos o *pixel* e verificamos seu valor, após, fazemos o seguinte:

Primeiro estabelecemos um valor de corte que podemos chamar de *Threshold*.

Depois falamos que os *pixels* com valores maiores que esse *Threshold* têm cor Branca, e os *pixels* com valores menores ou iguais a esse *Threshold* têm cor Preta.

Exemplo, se você pegar 127 para o valor de *Threshold* e um *pixel* na imagem em tons de Cinza que tenha os valores (86,86,86), ao aplicarmos o cálculo, esse *pixel* mudará para Preto e ficará assim (0,0,0). Já um *Pixel* que tenha os valores (218, 218, 218) mudará para Branco e ficará assim (255,255,255), ou seja, somente teremos *pixels* com valores (0,0,0) e (255,255,255) na imagem final que será em Preto e Branco, como na Figura 2.

Convolução de Imagens

A Convolução é uma operação que tem duas imagens operando, uma imagem/matriz a ser processada e uma imagem/matriz utilizada para processar, chamada de *Kernel* ou matriz de convolução, ou máscara, (NUNES, 2014). Cada *pixel* da imagem a ser processada é alterado por todos os *pixels* da matriz processadora (*kernel*), de maneira que são feitas multiplicações entre os *pixels* das duas matrizes e os resultados dessas multiplicações são acumulados em um somatório, sendo que cada *pixel* recebe esse valor do somatório. Todos os *pixels* da imagem processada são alterados, mas ficam com valores diferentes, pois os produtos entre as duas matrizes são feitos levando em consideração os valores dos vizinhos do *pixel* em questão. A Figura 3 mostra a operação de convolução alterando os valores dos *pixels* de uma imagem.

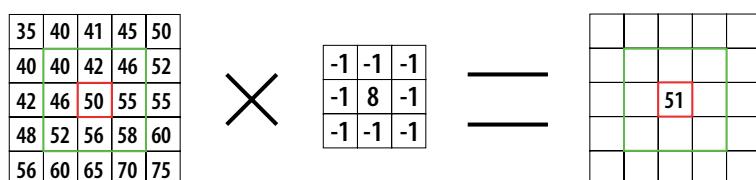


Figura 3 – Ilustração da operação de convolução



Esse vídeo explica em detalhes a convolução de imagens. A convolução é a base de um filtro.
Disponível em: https://youtu.be/WJqM_6HWbdI

Filtragem de Imagens

Irei mostrar dois filtros de imagem: um que serve para borrar um pouco a imagem (filtro de mediana) e outro que serve para detectar as bordas (filtro de Sobel) da imagem, ambos utilizam o conceito de convolução de imagem.

A Figura 4 mostra o filtro de suavização mediana, borramento suave, (FACON, 2016), a imagem original à esquerda, e o resultado à direita.



Figura 4 – Processamento de suavização de imagem com filtro de mediana

Fonte: Reprodução

Podemos chamar o filtro de Mediana de um filtro que deixa passar as frequências baixas e médias.

A Figura 5 mostra o filtro detector de borda Chamado Sobel (FILHO; NETO, 1999), a imagem original à esquerda, e o resultado à direita.



Figura 5 – Processamento de detecção de bordas em imagem com filtro de Sobel

Fonte: Reprodução

Podemos chamar o filtro de Mediana de um filtro que deixa passar as frequências altas.

Executando programas de Processamento de Imagens

Instalando o *Python* e *OpenCV*

Primeiramente, entre no site python.org/downloads e baixe o *Python*, como mostrado na Figura 6.

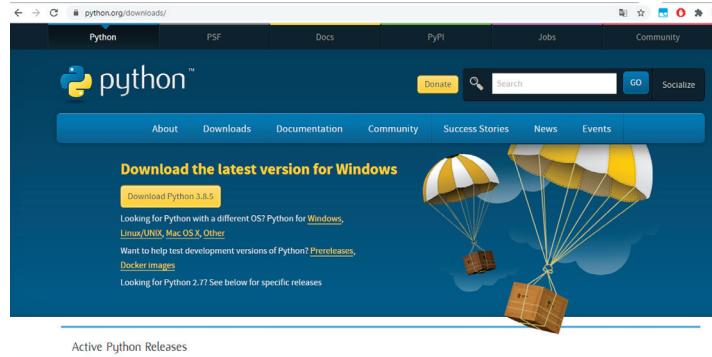


Figura 6 – Site e botão para download do Python

Fonte: Divulgação | Python

O arquivo executável para a instalação do *Python* é ilustrado na Figura 7.

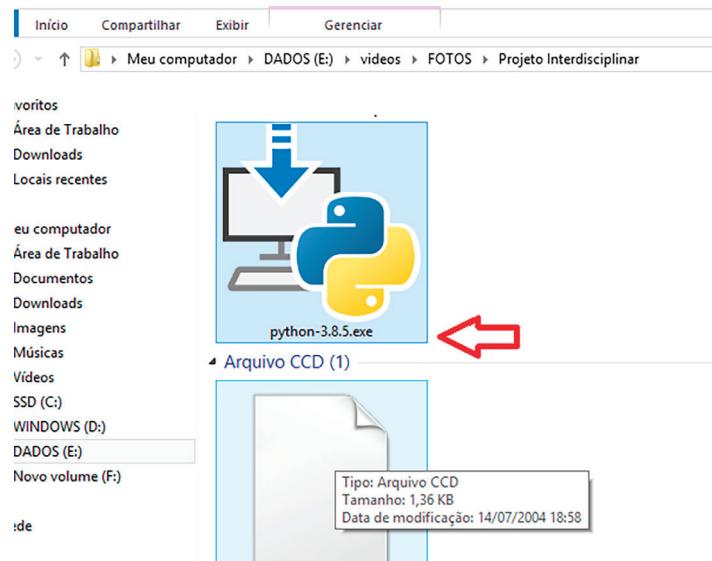


Figura 7 – Arquivo do executável para a instalação do Python

Fonte: Acervo do conteudista

Clique no executável para instalar e, em seguida, selecione a opção *Customize Installation* e em *Install launcher for all users*, como mostra a Figura 8.

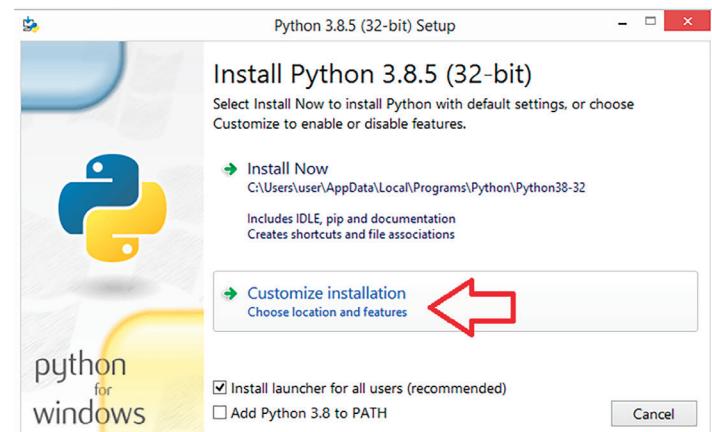


Figura 8 – Início da Instalação do Python

Fonte: Acervo do conteudista

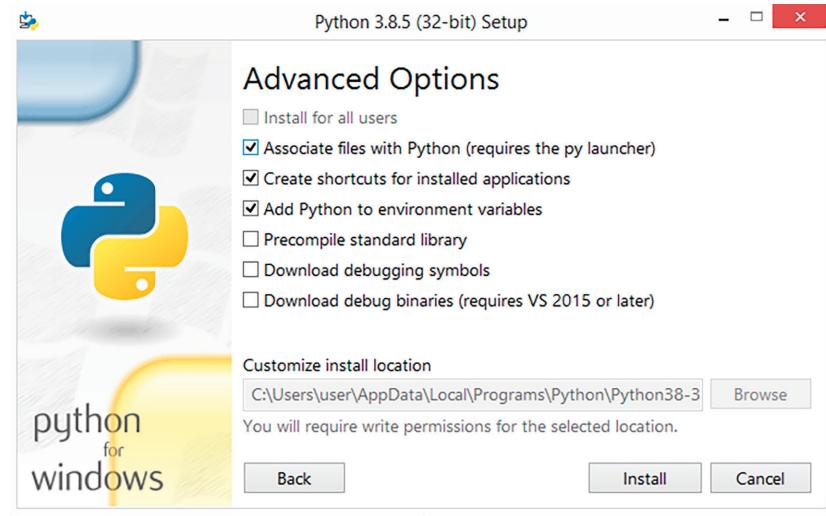


Figura 9 – Configurações essenciais do *Python*

Fonte: Acervo do conteudista

Selecione as caixas de diálogo *Associate files with Python*, *Create shortcuts for installed applications* e *Add Python Enviroment variables*, como mostra a Figura 9.

Depois de instalado o *Python*, procure o atalho do Programa chamado *Idle*, ou procure-o em seu computador como mostra a Figura 10.



Figura 10 – Procurando o *Idle* para rodar os programas

Fonte: Acervo do conteudista

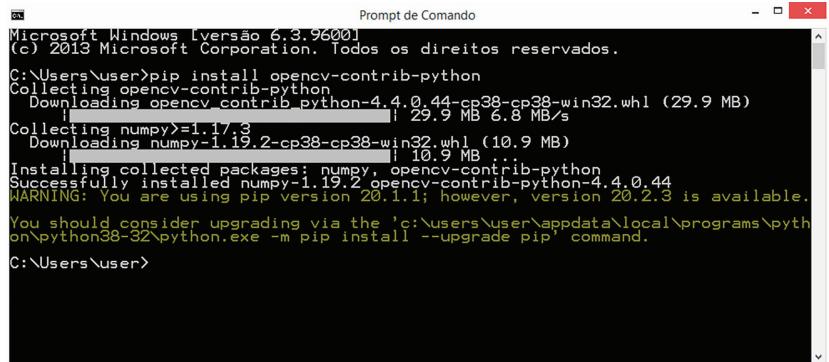
Agora vamos instalar o *OpenCV*.

Abra o *prompt* de comando e execute o comando:

```
pip install opencv-contrib-python
```

Depois aperte *enter*.

Você verá algo semelhante à janela mostrada na Figura 11.



```

Microsoft Windows [versão 6.3.9600]
(c) 2013 Microsoft Corporation. Todos os direitos reservados.

C:\Users\user>pip install opencv-contrib-python
Collecting opencv-contrib-python
  Downloading opencv_contrib_python-4.4.0.44-cp38-cp38-win32.whl (29.9 MB)
    100% |████████████████████████████████| 29.9 MB 6.8 MB/s
Collecting numpy>=1.17.3
  Downloading numpy-1.19.2-cp38-cp38-win32.whl (10.9 MB)
    100% |████████████████████████████████| 10.9 MB 10.9 MB
Installing collected packages: numpy, opencv-contrib-python
Successfully installed numpy-1.19.2 opencv-contrib-python-4.4.0.44
WARNING: You are using pip version 20.1.1; however, version 20.2.3 is available.
You should consider upgrading via the 'c:\users\user\appdata\local\programs\python\python38-32\python.exe -m pip install --upgrade pip' command.

C:\Users\user>

```

Figura 11 – Instalação do *OpenCV*

Fonte: Acervo do conteudista

Executando Programas de Processamento de Imagens

Depois de feitas as instalações do *Python* e do *OpenCV*. No *Idle*, salve esse programa como GRAY.PY e certifique-se de que a imagem está na mesma pasta, ou mude o caminho da imagem para o local mais apropriado. Execute esse programa no *Idle* com a tecla F5:

```

import cv2

image = cv2.imread('lena_color_128_128_.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imshow('Original image',image)
cv2.imshow('Gray image', gray)

cv2.waitKey(0)
cv2.destroyAllWindows()

```

 Os programas demonstrados nesse texto serão mais bem explicados na videoaula da unidade.

Esse programa executa a conversão de uma imagem colorida para uma em tons de Cinza, como mostra a Figura 12.

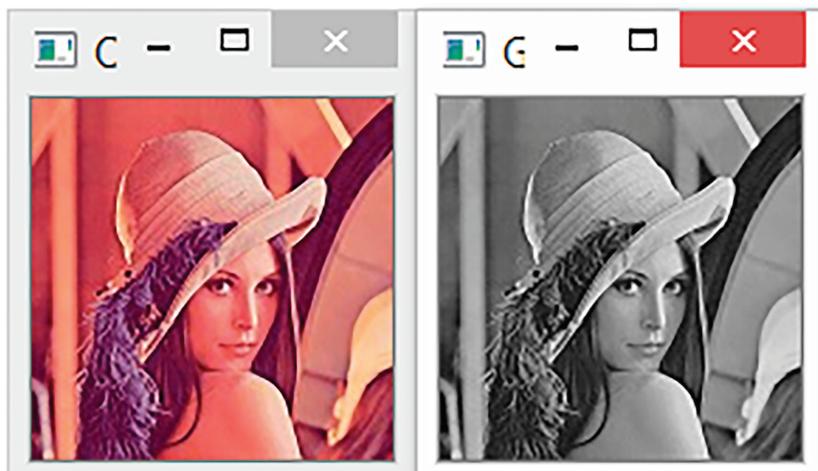


Figura 12 – Conversão da imagem colorida na imagem em tons de cinza

Fonte: Reprodução

Depois de testar o primeiro programa (GRAY.py), salve o programa abaixo como BW.py e rode esse programa no *Idle* com a tecla F5:

```
import cv2

originalImage = cv2.imread('lena_color_128_128_.jpg')
grayImage = cv2.cvtColor(originalImage, cv2.COLOR_BGR2GRAY)

(thresh, blackAndWhiteImage) = cv2.threshold(grayImage, 127, 255, cv2.THRESH_BINARY)

cv2.imshow('Black white image', blackAndWhiteImage)
cv2.imshow('Original image', originalImage)
cv2.imshow('Gray image', grayImage)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

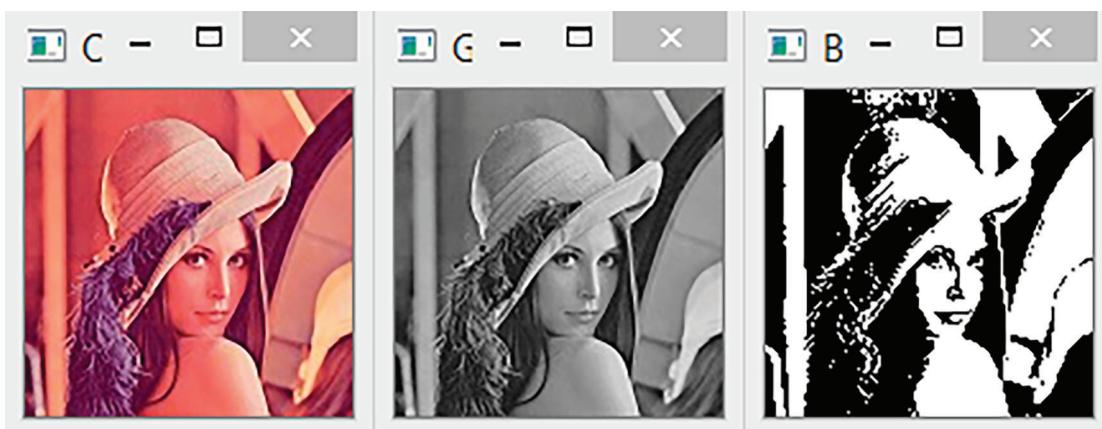


Figura 13 – Conversão da imagem em tons de cinza para a imagem em preto e branco

Fonte: Reprodução

Executando outros programas que representam filtros, teremos os seguintes resultados:

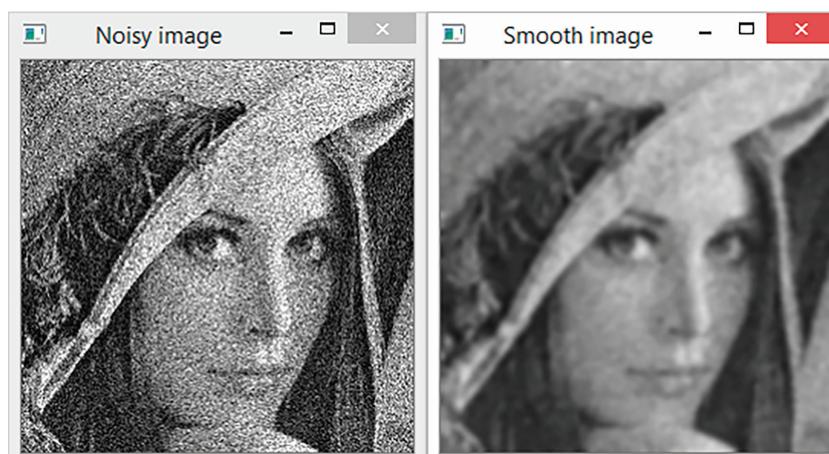


Figura 14 – Filtragem Gaussiana de imagem com ruído

Fonte: Reprodução

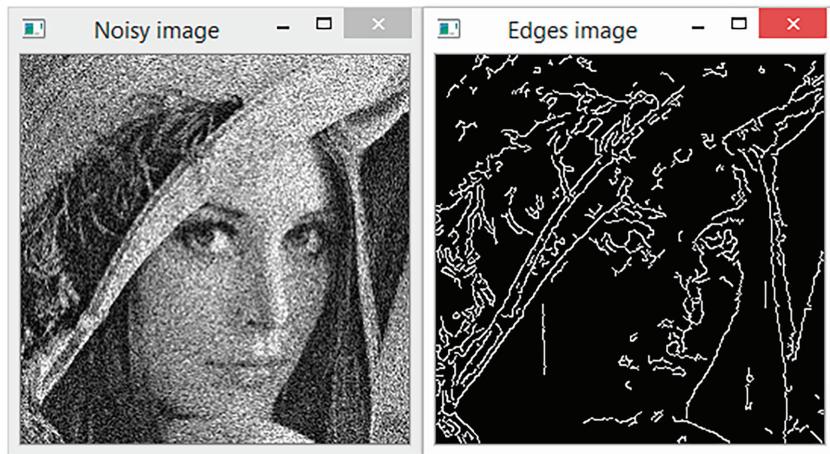


Figura 15 – Filtragem de Canny para detectar bordas da imagem

Fonte: Reprodução



Os resultados das Figuras 14 e 15 podem ser obtidos com os códigos e imagem que estão disponíveis no link: <https://bit.ly/39MHu9z>

Esses processamentos serão explicados em mais detalhes na videoaula desta unidade.

Introdução ao Domínio da Frequência

Funções Seno e Cosseno com Ângulo, Fase e Frequência

O gráfico abaixo é da função seno.

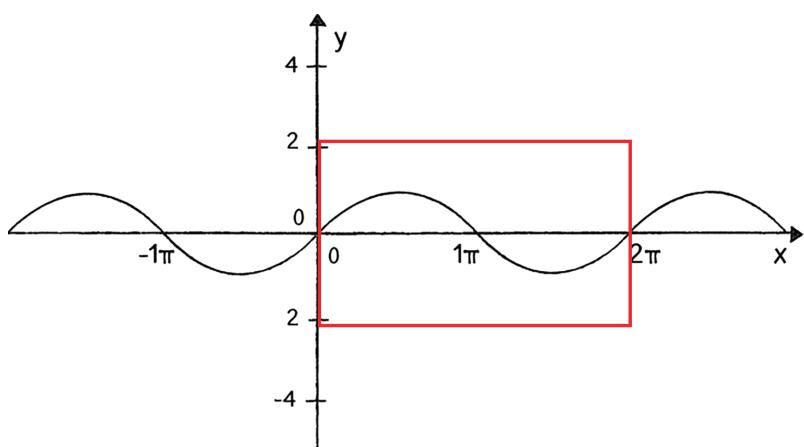


Figura 16 – Função Seno

Fonte: Adaptada de montereyinstitute.org

Na Figura 16, repare que na parte vermelha ela começa embaixo no valor zero, depois sobe e cruza o zero em 1π , desce e cruza o zero novamente em 2π . Esse intervalo de 0 a 2π é o período da função seno; a partir daí, a função sempre se repete subindo, descendo e cruzando o zero.

Repare que a função no valor 0, está embaixo, mas podemos deslocar a função para que ela inicie em alta no valor 0. Para isso, é necessário que mudemos sua Fase.

Baseado no conceito de fase, podemos ver que a função cosseno é uma variação da função seno com a fase modificada, por isso, a função cosseno está no alto em 0, enquanto a função cosseno está embaixo no 0. Isso é ilustrado na Figura 17.

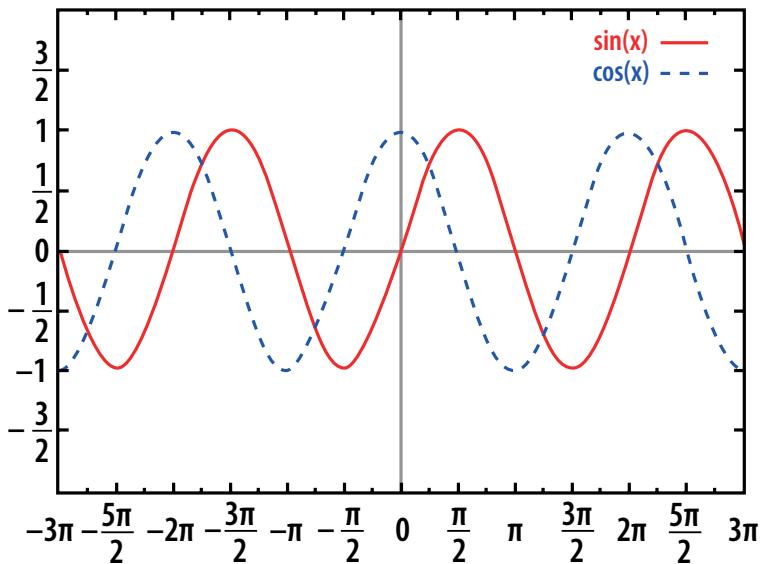


Figura 17 – Funções Seno e Cosseno

Os ângulos nessas funções são representados pelos valores de 0 a 2π , ou seja, o seu período. Sendo 0 o ângulo de 0 grau, π é equivalente a 180 graus e 2π equivalente a 360 graus.

A frequência das funções seno e cosseno é a velocidade em que as funções cruzam o zero. Se elas cruzam o zero mais lentamente, a frequência é baixa, se elas cruzam o zero mais rapidamente, a frequência é alta. Por exemplo, na Figura 18, podemos ver que a frequência da função seno que está em cor verde é maior que a função seno que está em cor vermelha.

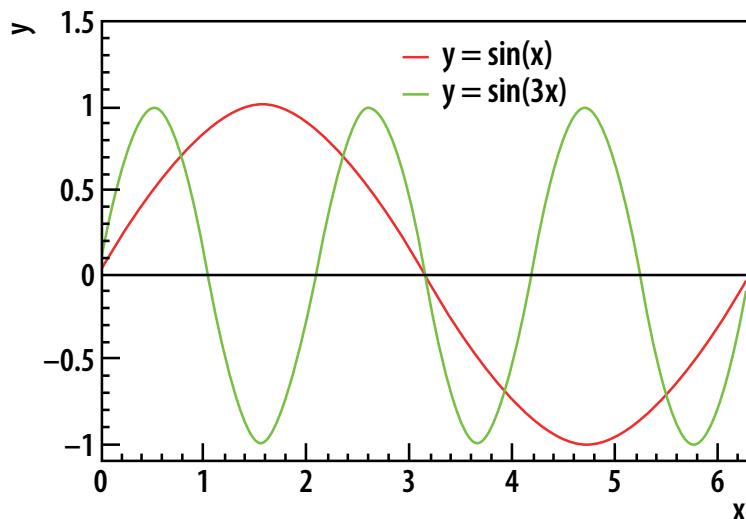


Figura 18 – Diferentes Funções Seno com diferentes frequências

Números Complexos

Os números complexos surgiram com a necessidade de se resolver equações que representam raízes de números negativos, o que não é possível com o conjunto dos números reais, assim, foi criado o conjunto dos números complexos.

Às vezes, o conjunto dos números complexos é chamado de conjunto dos números imaginários, pois foi definido que a raiz de -1 seria chamada de número imaginário, ou seja, $i = \sqrt{-1}$. Desse modo, cada número complexo tem uma parte real e uma parte imaginária.

Os números complexos podem ser representados de três formas: a forma algébrica ($z = a + bi$), composta por uma parte real a , e uma parte imaginária b ; a forma geométrica, representada no plano complexo, conhecido também como plano de Argand-Gauss; e a sua forma trigonométrica, conhecida também como forma polar.

A Figura 19 apresenta a representação gráfica de um número complexo.

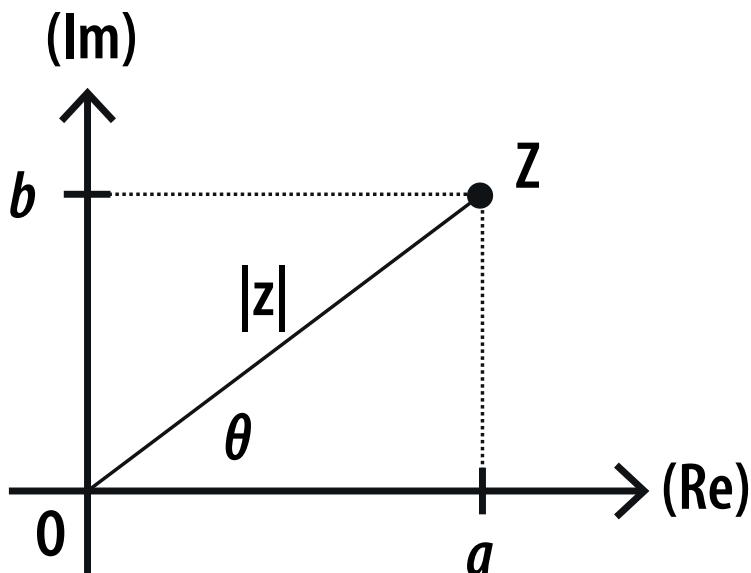


Figura 19 – Representação de um número complexo no Plano Cartesiano em sua forma geométrica

Para a representação de uma imagem em termos de frequência (lembrando que as frequências de uma imagem são suas cores), pode-se representá-la como um sinal descrito por várias funções senos e/ou cossenos e aproveitar as geométricas e trigonométricas de um número complexo para representar uma imagem de uma outra forma, ou seja, seu espectro de frequências.

Assim, é possível pegar uma imagem e mudar a forma como ela é representada para a forma em frequência, depois aplicar filtros na sua forma em frequência e retornar a imagem filtrada para o seu domínio original, que é chamado domínio espacial.

Isso é feito, pois há ocasiões que processar a imagem para filtrá-la no domínio espacial fica muito complicado e demanda muito cálculo/processamento, daí fazer esses cálculos no domínio da frequência pode ser mais simples. O que justifica fazer os cálculos no domínio da Frequência e depois retornar com a Imagem resultante para o domínio espacial.

Exemplo de transformação para o domínio da Frequência:

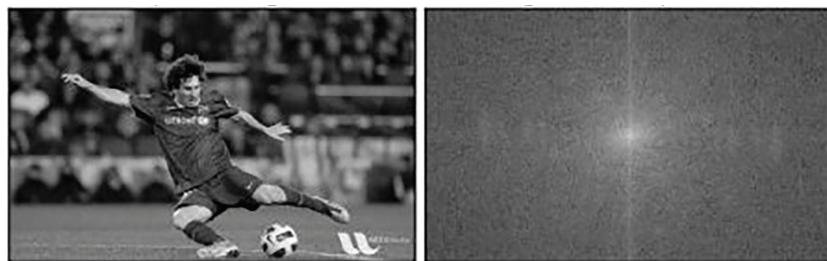


Figura 20 – Imagem da esquerda no domínio espacial (Input Image) e imagem da direita, mesma, no domínio da frequência (Magnitude Spectrum)

Fonte: docs.opencv.org

Filtragem no Domínio da Frequência

A Figura a seguir mostra a imagem de um caminhão à esquerda e sua representação no domínio da frequência à direita.



Imagen da esquerda no domínio espacial e imagem da direita, a mesma, no domínio da frequência, disponível em: <https://bit.ly/3rifNv9>

Já a Figura a seguir apresenta a filtragem realizada.



Imagen à esquerda, no domínio espacial, imagem ao centro, o *Kernel* utilizado para processar a imagem no domínio da frequência, e a imagem resultante, já retornada para o domínio espacial, disponível em: <https://bit.ly/2LhDGDF>

Esse conceito apresentado nas duas figuras anteriores é chamado de Transformada de Fourier e Transformada Inversa de Fourier.

É importante ressaltar que os programas demonstrados neste texto serão detalhados na videoaula da unidade.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

▶ Vídeos

Computação Gráfica – Aula 05 – Processamento de Imagens – I

<https://youtu.be/gw9ws3QbSy8>

Computação Gráfica – Aula 03 – Fundamentos de Cor

<https://youtu.be/C-Xsq5q6ba0>

☰ Leitura

Processamento de Imagens: Métodos e Análises

<https://bit.ly/3oQIltY>

Filtragem Espacial

<https://bit.ly/39K09CT>

Suavização de imagens

<https://bit.ly/39KxnBV>

Referências

FACON, J. **Técnicas de Processamento Digital de Imagens Aplicadas à Área da Saúde.** XIII Escola Regional de Informática da SBC – Paraná, 2016.

FILHO, O. M.; NETO, H. V. **Processamento dial de imagens.** 1999. Disponível em: <<https://www.ogemarques.com/wp-content/uploads/2014/11/pdi99.pdf>>. Acesso em: 15/10/2020.

MACIEL, A. M.; VINHAS, L.; CÂMARA, G. Aplicação de técnicas de processamento digital de imagens usando a extensão espacial *PostGIS Raster* em imagens de sensoriamento remoto. **Anais XVII Simpósio Brasileiro de Sensoriamento Remoto – SBSR**, João Pessoa-PB, Brasil, 25 a 29 de abril de 2015, INPE, 2015.

NUNES, L. D. P. **Algoritmos para Processamento de Imagens Visando Implementação em FPGA.** 2014. Disponível em: <<https://www.lume.ufrgs.br/bitstream/handle/10183/101192/000931913.pdf?sequence=1>>. Acesso em: 15/10/2020.



Cruzeiro do Sul
Educacional