

FIAP

Sistemas de Informação

COGNITIVE COMPUTING, COMPUTER VISION AND IOT SYSTEMS

Prof. Arnaldo Jr / Prof. Yan Coelho

Apresentação do professor

FIAP

Yan Gabriel Coelho

Formação:

Graduado em engenharia elétrica – UNINOVE

Pós-graduado em Inteligência Artificial e Aprendizagem de Máquina - UNINOVE

Licenciatura em matemática – UNINOVE (ATUAL)

Experiência como professor:

FIAP

Sequencial

ESTADO

OBJETIVO

Etc...

Experiência

Técnico em Hardware

Coordenador de cursos de TI (ATUAL)

Hobbies:

Séries, Cultura Maker, Jogos, PC, entre outros.



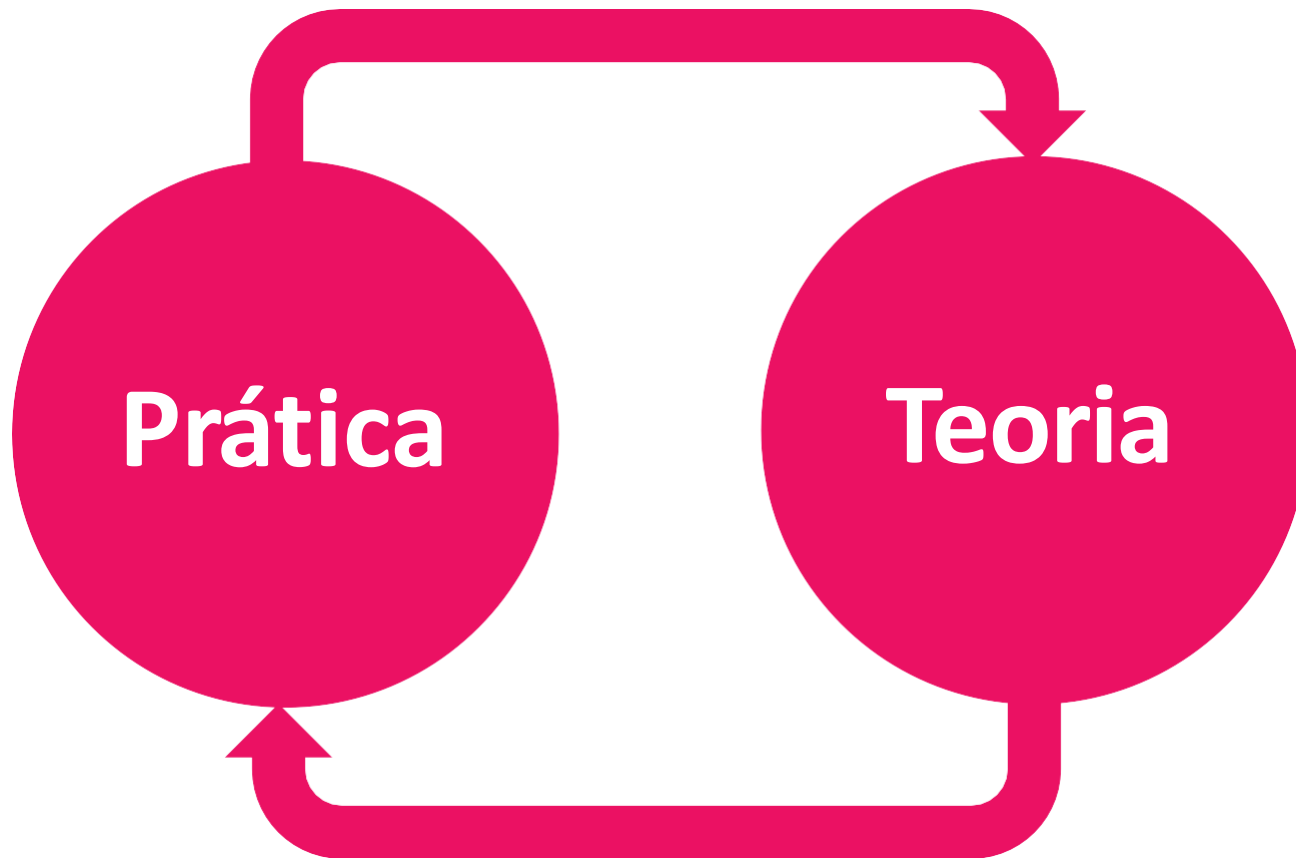
proyan.coelho@fiap.com.br



Microsoft Teams

Dinâmica do curso

- Metodologia baseada em projetos (PBL) e hands-on.

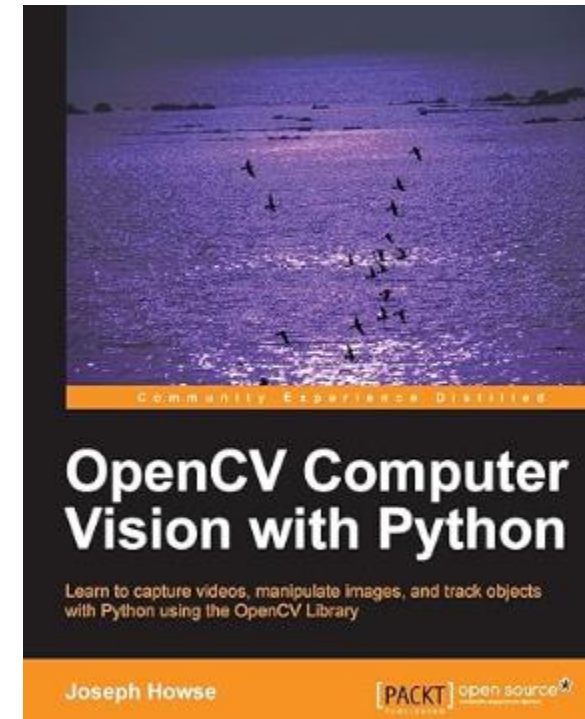
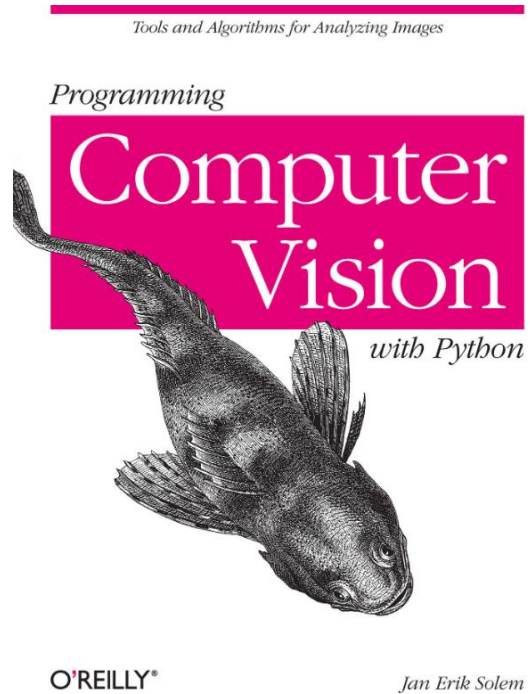


Mão na massa!!!!

Ementa

- Visão computacional
 - Processamento de imagens, manipulação, filtros, transformadas... with Python
- IA
 - Machine Learning, classificadores, Aprendizado, DL, redes neurais, redes convolucionais... Etc. with Python
- IoT
 - Flasher, atuadores, sensores, comunicação... Etc. with C++

Computer Vision



Objetivos (CV)

Ao final da disciplina o aluno será capaz de:

- Compreender os conceitos de Processamento digital de imagens.
- Conhecer as principais técnicas para detecção e segmentação de objetos.
- Conhecer e aplicar técnicas de visão computacional.

O que preciso instalar para acompanhar esse curso?

FIAP



```
jupyter sample_plot (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib inline

In [2]: import sys
import matplotlib

print(f"Python Version: {sys.version}")
print(f"NumPy Version: {np.__version__}")
print(f"Pandas Version: {pd.__version__}")
print(f"Matplotlib Version: {matplotlib.__version__}")

Python Version: 3.9.7 | packaged by conda-forge | (default, Sep 29 2021, 19:22:19)
[Clang 11.1.0 ]
NumPy Version: 1.21.4
Pandas Version: 1.3.4
Matplotlib Version: 3.5.0
```

```
settings.py -- transfer-app
EXPLORER
TRANSFER-APP
  > colleges
  > courses
  > majors
  > pages
  > references
  > static
  > staticfiles
  > templates
  > transfer_project
    > __pycache__
    > _init_.py
    > asgi.py
    > settings.py 2, M
    > urls.py
    > wsgi.py
  > users
  > .env
  > .env-example
  > .gitignore
  > OUTLINE
  > TIMELINE
  > Python 3.8.9 64-bit
  > 0 1 2
  > Ln 16, Col 1 Spaces: 4 UTF-8 LF Python
  >
transfer_project > settings.py > ...
1 # transfer_project/settings.py
2
3 from pathlib import Path
4 import os
5
6 import environ
7
8 # Build paths inside the project like this: BASE_DIR / 'subdir'.
9 BASE_DIR = Path(__file__).resolve(strict=True).parent.parent
10 # Create env object from the environ package
11 env = environ.Env(DEBUG=(bool, False))
12 # read .env file
13 env_file = os.path.join(BASE_DIR, ".env")
14 if os.path.exists(env_file):
15     environ.Env.read_env(env_file)
16
17 # if production deploy, import django-heroku package
18 if env("DEPLOY")=="prod":
19     import django_heroku
20
21 # Quick-start development settings - unsuitable for production
22 # See https://docs.djangoproject.com/en/3.8/howto/deployment/checklist/
23
24 SECRET_KEY = env("SECRET_KEY")
25
26 DEBUG = env("DEBUG")
```


The logo for Google Colab, featuring the word "colab" in a lowercase, rounded font. The "co" is yellow with a subtle gradient, and the "lab" is orange. The letters are bold and modern.

1. Introdução a processamento de imagens

Objetivos da aula:

- Conhecer o que é uma imagem digital
- Conhecer como fazer leitura e exibição de imagens
- Conhecer algumas propriedades de imagens
- Conhecer canais de cores de imagens

Representação e visualização de imagem

Uma imagem digital nada mais é que uma matriz de linhas e colunas, onde cada posição desta matriz contém o valor de um *pixel*.

O valor de cada pixel representa a intensidade de cor naquele ponto específico.

Vamos lá...

```
# Importando a biblioteca OpenCV
```

```
import cv2
```

```
#import a biblioteca Numpy8 bits
```

```
import numpy as np
```

```
#linha magica para imprimir graficos no notebook
```

```
%matplotlib inline
```

```
from matplotlib import pyplot as plt
```

```
print ("OpenCV Versão : %s " % cv2.__version__)
```

Exibir a imagem

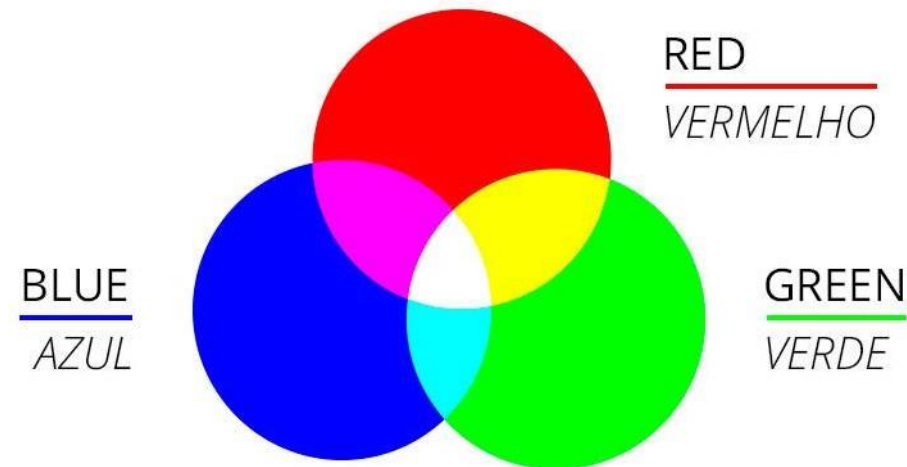
```
image = cv2.imread("NATUREZA_1.jpg")
```

```
plt.imshow(image)
```

Sem eixos

```
#para nao imprimir os eixos  
image = cv2.imread("NATUREZA_1.jpg")  
  
plt.axis('off')  
plt.show()
```

A imagem colorida possui três dimensões: as linhas e as colunas da matriz, bem como os canais da imagem. Uma imagem colorida geralmente possui três canais: R (Red - vermelho) G (Green - verde) B (Blue - azul)



Mas porque a imagem é mostrada de modo estranho pelo pacote matplotlib?

Porque a OpenCV representa os canais da imagem na ordem B - G - R, e não R - G - B como é esperado pela maior parte das bibliotecas.

Assim, para podermos visualizar corretamente uma imagem do OpenCV com matplotlib, precisamos inverter os canais, como no código abaixo:

```
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
plt.imshow(image_rgb)
```

```
plt.axis('off')  
plt.show()
```



cv2.cvtColor()

Representação da imagem(SHAPE)

```
# Mostrando a representação interna da imagem
print("Dimensões da imagem: ", image_rgb.shape)
print("Quantidade de linhas: ", image_rgb.shape[0])
print("Quantidade de colunas: ", image_rgb.shape[1])
print("Camadas de cores: ", image_rgb.shape[2])
```

Representação da imagem(MATRIZ)

```
# Mostrando a representação interna da imagem  
print("Dimensões da imagem: \n", image_rgb)
```

Matriz

A matriz acima é a representação da imagem de forma numérica, é o valor de cada pixel da imagem. Com esta imagem fica complicado. Vamos tentar analisar separando os canais de cores de um pixel específico.

```
(b, g, r) = image [50, 50]  
print('O pixel (50, 50) tem as seguintes cores:')  
print('Vermelho:', r, 'Verde:', g, 'Azul:', b)
```

Desafio 1

Abra a imagem "**img3x3.png**" e plote suas componentes externas (shape) e internas (matriz).

Como você está relacionando as posições da matriz com os pixels da imagem??

Imagem em tons de cinza

Em muitos casos trabalhamos com imagens na escala de cinza, logo, a imagem possui apenas 1 canal de cor.


```
import cv2
import numpy as np
```

```
from matplotlib import pyplot as plt
```

```
# Carregando a imagem na versão tons de cinza (grayscale) de um arquivo
imagem_cinza = cv2.imread("img3x3.png", cv2.IMREAD_GRAYSCALE)
```

```
# ou use o argumento 0, tem o mesmo efeito de importar na escala de cinza
#imagem_cinza = cv2.imread("img3x3.png", 0)
```

```
plt.imshow(imagem_cinza)
```

```
plt.axis('off')
plt.show()
imagem_cinza
```

Desafio 2

Eita! Alguma coisa está errada nesse plot, era esperado uma imagem na escala de cinza. Por que apareceu isso, como corrigir?

Alterando o tamanho de uma imagem

O redimensionamento da imagem pode ser feito na OpenCV através do comando `cv2.resize(imagem, tamanho, interpolação)`

O tamanho é dado por uma tupla (W,H), onde W é a largura (número de colunas) e H é a altura (número de linhas)

Alterando o tamanho de uma imagem

```
# Carregando a imagem na versão colorida de um arquivo
import cv2
import matplotlib.pyplot as plt

imagem = cv2.imread("NATUREZA_1.jpg")
image = cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)

print("Dimensões da imagem: ", image.shape)

imagem2 = cv2.resize(image, (600,400), cv2.INTER_LINEAR)
print("Novas dimensões da imagem: ", imagem2.shape)

plt.imshow(imagem2)
plt.show()
```

Alterando os valores dos pixels de uma imagem

Range de valores

Antes de alterar os valores dos pixels temos que entender que a OpenCV trabalha com valores de 8 bits para cada componente de cor ou escala de cinza, quer dizer que os valores possíveis estão no range entre **0** e **2^8-1** , que é a mesma que dizer entre **0** e **255**.

Desafio 3

Implemente um código que faça a alteração do pixel(0,0) para a cor Magenta - RGB (255,0,255);

