Disciplina: TÓPICOS ESPECIAIS III
*Deep Learning* para Sensoriamento Remoto

## Introdução e conceitos básicos:

- processamento de imagens, reconhecimento de padrões e visão computacional

- bibliotecas Python:
  - skimage (scikit image)
  - tkinter
  - openCV

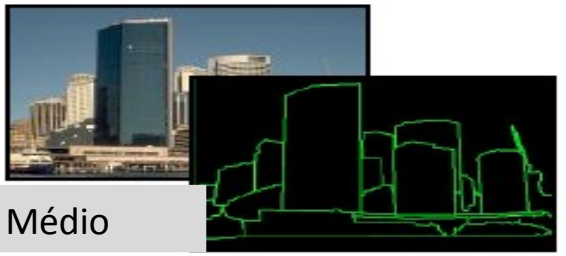
# Níveis de Processamento digital de imagens para a visão computacional:

- Os métodos de **baixo nível** geralmente usam pouco conhecimento sobre o conteúdo ou a semântica das imagens. Envolvem operações como a redução de ruído, o aumento do contraste, a extração de bordas e a compressão de imagens.

- Os métodos de **alto nível** envolvem tarefas como a segmentação das imagens em regiões ou objetos de interesse, descrição desses objetos de modo a reduzi-los a uma forma mais apropriada para representar o conteúdo da imagem e reconhecimento ou classificação desses objetos.

Baixo

Entrada: imagem;
Saída: imagem

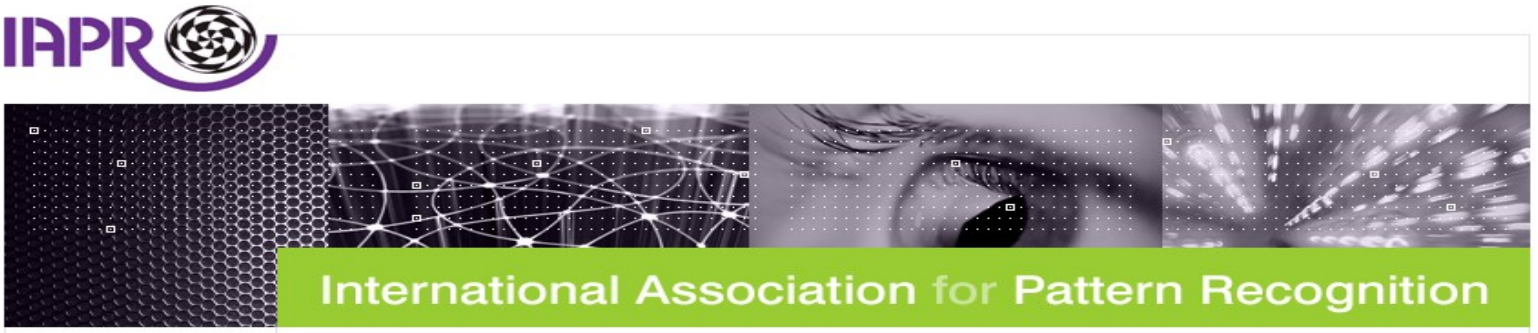


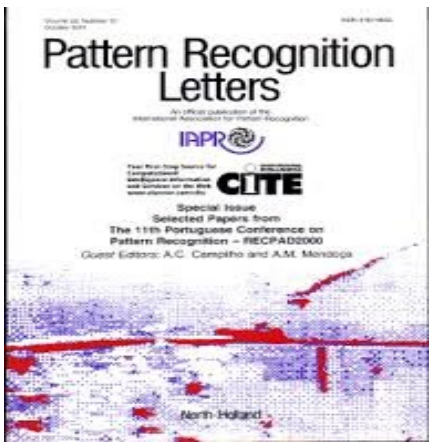
Médio

Entrada: imagem;
Saída: atributos extraídos das imagens (bordas, contornos, identificação de objetos)

Alto

"Pessoas andando com guarda chuva"



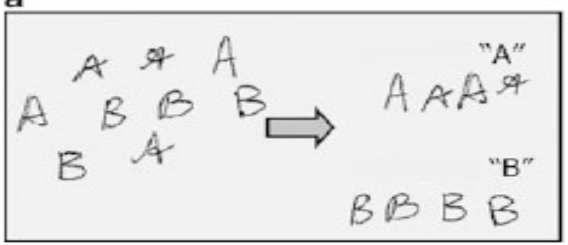

The International Association for Pattern Recognition (IAPR) is an international association of non-profit, scientific or professional organizations (being national, multi-national, or international in scope) **concerned with pattern recognition, computer vision, and image processing in a broad sense**. Normally, only one organization is admitted from any one country, and individuals interested in taking part in IAPR's activities may do so by joining their national organization.

Reconhecimento de Padrões e Classificação:

- Capacidade humana inata.
- Tarefas corriqueiras:
  - reconhecer pessoas,
  - reconhecer objetos,
  - etc.

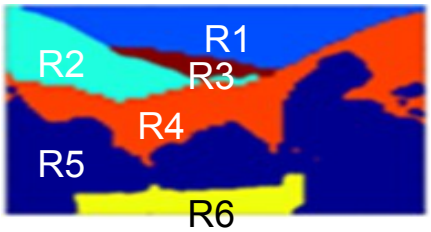
Criar classes após análise dos objetos:

Separar objetos em classes já conhecidas:



Segmentação

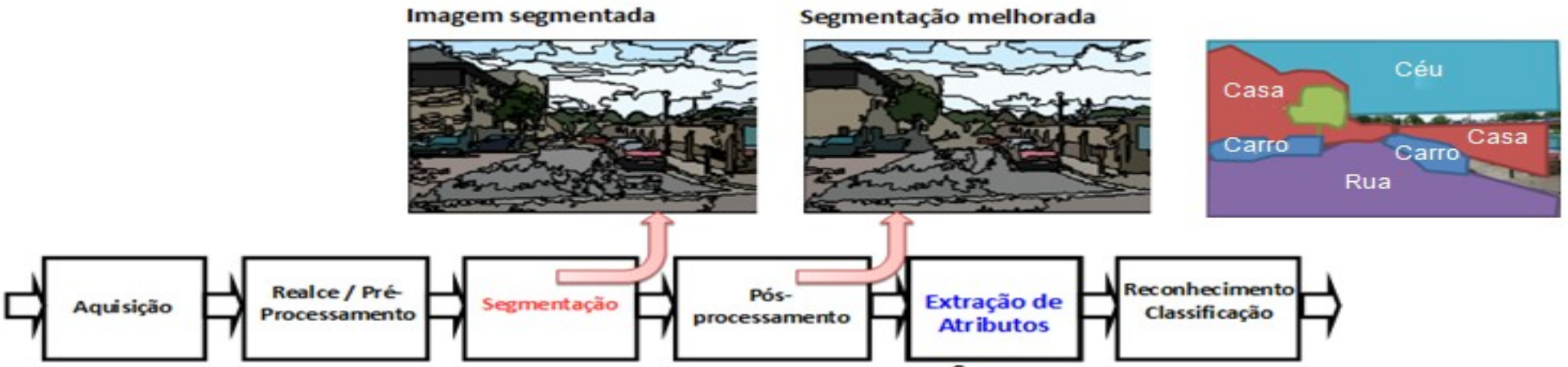
Seja I a imagem de entrada:




A **segmentação** particiona I em *n* regiões conexas R1, R2,..., Rn,

tal que:

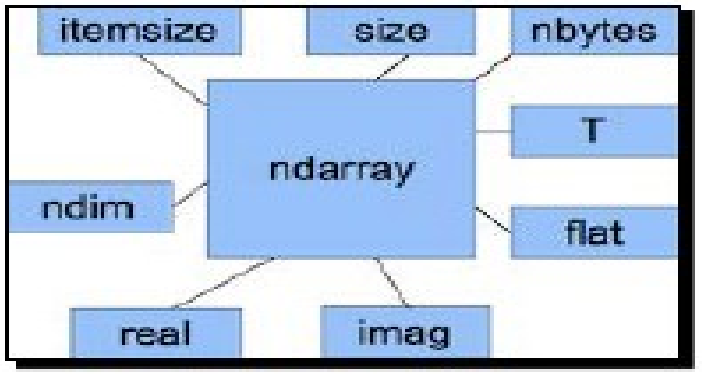
| | |
|---|---|
| - Pixels de uma região devem satisfazer uma propriedade comum. | $P(R_i)$ = VERDADEIRO para i = 1, 2,.....n |
| - a segmentação deve ser completa. | $\bigcup_{i=1}^{n} R_i = I$ |
| - as regiões devem ser disjuntas.. | $R_i \bigcap R_j = \varnothing \quad \forall i = 1,2,\ldots n$ |
| - Regiões adjacentes Ri e Rj não podem ser unidas numa única região. | $P(R_i \cup R_i) = FALSO \quad para \quad i \neq j$ |




**- características externas: tamanho, forma...**

**- características internas : cor, texturas,...**

Alguns atributos da classe ndarray:



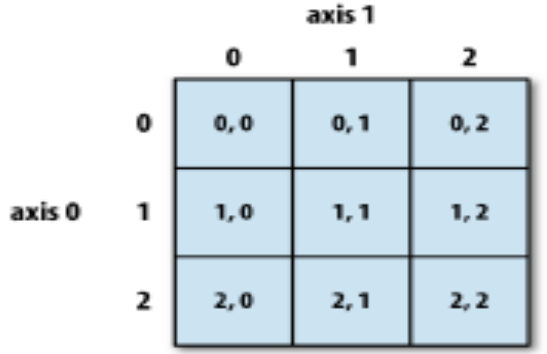

Instância da classe

| | |
|---|---|
| `ndarray.dtype` | Data type of elements |
| `ndarray.ndim` | Dimension of array |
| `ndarray.shape` | Shape of array, (*n*, *m*) for (*n*, *m*) array |
| `ndarray.size` | Size of array, $n \times m$ |
| `ndarray.itemsize` | Size in bytes of each element |
| `ndarray.data` | Buffer data containing actual element |
| `ndarray.reshape` | Reshapes keeping $n \times m$ constant |

Tipos de dados numéricos:

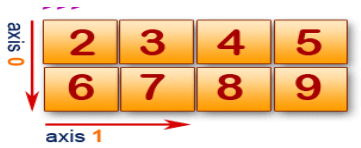
| dtype | Variants | Description |
|---|---|---|
| `int` | `int8, int16, int32, int64` | Integers |
| `uint` | `uint8, uint16, uint32, uint64` | Unsigned (nonnegative) integers |
| `bool` | `Bool` | Boolean (True or False) |
| `float` | `float16, float32, float64, float128` | Floating-point numbers |
| `complex` | `complex64, complex128, complex256` | Complex-valued floating-point numbers |

Posição dos elementos de arrays:

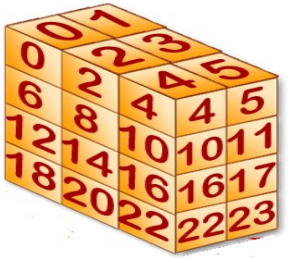



```
import numpy as np
```

```
# array 2D
x = np.array([[2,3,4,5],[6,7,8,9]])
valor =x [[0],[2]]
print(valor)
```




```
# array 3D
x = np.array([ [[0,1], [6,7],[12,13],[18,19]], [[2,3],[8,9],
[14,15],[20,21]],
[[4,5],[10,11],[16,17],[22,23]] ])
print(x)
 valor = x[[0],[2],[0]]
 print(valor)
```



# Scientific Computing Tools for Python

SciPy refers to several related but distinct entities:

- The *SciPy ecosystem*, a collection of open source software for scientific computing in Python.
- The *community* of people who use and develop this stack.
- Several *conferences* dedicated to scientific computing in Python - SciPy, EuroSciPy and SciPy.in.
- The SciPy library, one component of the SciPy stack, providing many numerical routines.

## The SciPy ecosystem

Scientific computing in Python builds upon a small core of packages:

- Python, a general purpose programming language. It is interpreted and dynamically typed and is very suited for interactive work and quick prototyping, while being powerful enough to write large applications in.
- NumPy, the fundamental package for numerical computation. It defines the numerical array and matrix types and basic operations on them.
- The SciPy library, a collection of numerical algorithms and domain-specific toolboxes, including signal processing, optimization, statistics and much more.
- Matplotlib, a mature and popular plotting package, that provides publication-quality 2D plotting as well as rudimentary 3D plotting
- pandas, providing high-performance, easy to use data structures.
- SymPy, for symbolic mathematics and computer algebra.
- scikit-image is a collection of algorithms for image processing.
- scikit-learn is a collection of algorithms and tools for machine learning.
- h5py and PyTables can both access data stored in the HDF5 format.

NumPy
Base N-dimensional array package

SciPy library
Fundamental library for scientific computing

Matplotlib
Comprehensive 2D Plotting

IP[y]: IPython

IPython
Enhanced Interactive Console

Sympy
Symbolic mathematics

pandas
Data structures & analysis

scikit-image
image processing in python

O módulo *io* contém utilitários para ler e armazenar imagens em diversos formatos.

- skimage.io.imread(fname[, as_gray, plugin, …])
  onde
    fname : endereço / nome do arquivo
    [, …] : parâmetros opcionais

- skimage.io.imsave(fname, arr[, plugin, …])
  onde
    fname : endereço / nome do arquivo
    arr : variável a ser armazenada
    [, …] : parâmetros opcionais

```
import matplotlib.pyplot as plt
import skimage.io

img = skimage.io.imread('C:/... /imagem1.tif')
ncol, nlin,k = img.size
I= np.zeros([m,n])
for j in range(0,nlin):
    for i in range(0,ncol);
            R = img[i,j,0]/255.0
            G = img[i,j,1]/255.0
            B = img[i,j,2]/255.0
            #  intensidade
            I[i,j] = ( R + G + B) / 3
plt.imshow(I)
plt.show()
```

Função: bloco de código

```
import matplotlib.pyplot as plt
import skimage.io

def intensidade (img):
    # nome_da_função (argumento)
    ncol, nlin,k = img.size
    I= np.zeros([m,n])
    for j in range(0,nlin):
        for i in range(0,ncol);
            R = img[i,j,0]/255.0
            G = img[i,j,1]/255.0
            B = img[i,j,2]/255.0
            #  intensidade
            I[i,j] = ( R + G + B) / 3
    return I

img1 = skimage.io.imread('C:/... /imagem1.tif')
# Calcular intensidade para a primeira imagem
I1 = intensidade(img1)
```

# Tkinter


Python provides various options for developing graphical user interfaces (GUIs). The most important features are listed below.

- **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
- **wxPython** − This is an open-source Python interface for wxWidgets GUI toolkit. You can find a complete tutorial on WxPython here .
- **PyQt** −This is also a Python interface for a popular cross-platform Qt GUI library. TutorialsPoint has a very good tutorial on PyQt here .
- **JPython** − JPython is a Python port for Java, which gives Python scripts seamless access to the Java class libraries on the local machine http://www.jython.org .

There are many other interfaces available, which you can find them on the net.

## Tkinter Programming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Disponível em: *tutorialspoint.com/python3/python_gui_programming.htm*

# Busca de imagem com tkinter: arquivo Exemplo_tkinter.txt

```
import skimage
import numpy as np
import matplotlib.pyplot as plt
import os

from tkinter import filedialog
from tkinter import *
root = Tk()
root.title( "Ler imagem")
root.geometry('450x50')
filename = filedialog.askopenfilename()
Imagem = skimage.io.imread(filename)

Img_bin I= np.uint8(Imagem/255)

fig = plt.figure()
fig1 = plt.subplot(121)
fig2 = plt.subplot(122)
fig1.imshow(Imagem )
fig2.imshow(Img_bin)

plt.show()

root.mainloop()
```


http://opencv.org/)

- Criado por volta de 1999.
- biblioteca open source para visão para processamento computacional de imagens e computacional
- Cada módulo lida com um grupo específico de aplicação.

Os módulos comuns são:

- `core`: Core OpenCV data structures and functionalities
- `imgproc`: Image processing
- `imgcodecs`: Image file reading and writing
- `videoio`: Media input/output routines
- `highgui`: High-level graphical user interface
- `video`: Video analysis
- `calib3d`: Camera calibration and 3D reconstruction
- `features2d`: Working with 2D features description and matching
- `objdetect`: Object detection such as faces
- `ml`: Machine learning
- `flann`: Clustering and searching in higher-dimensional spaces
- `photo`: Computational photography
- `stitching`: Stitching images together
- `shape`: Shape matching
- `superres`: Super-resolution enhancement
- `videostab`: Video stabilization
- `viz`: 3D visualization

# Image Processing in OpenCV

- **Changing Colorspaces**

  Learn to change images between different color spaces. Plus learn to track a colored object in a video.

- **Geometric Transformations of Images**

  Learn to apply different geometric transformations to images like rotation, translation etc.

- **Image Thresholding**

  Learn to convert images to binary images using global thresholding, Adaptive thresholding, Otsu's binarization etc

- **Smoothing Images**

  Learn to blur the images, filter the images with custom kernels etc.

- **Morphological Transformations**

  Learn about morphological transformations like Erosion, Dilation, Opening, Closing etc

- **Image Gradients**

  Learn to find image gradients, edges etc.

- **Canny Edge Detection**

  Learn to find edges with Canny Edge Detection

- **Image Pyramids**

  Learn about image pyramids and how to use them for image blending

- **Contours in OpenCV**

Leitura/gravação de imagem com openCV:
- cv2.imread( )
- cv2.imwrite( )

**Exemplo:**

```
import numpy
import cv2
img = cv2.imread('F:/Aulas/ImagensTeste/input.jpg')
print(type(img))
print(img.shape)
print(img.dtype)

cv2.imwrite('F:/Aulas/ImagensTeste/output.jpg',img)
```

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('C:/.../pessoa.jpg')


# Converter imagem em escala de cinza
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray,'gray')
plt.show()

# Visualizar a nova imagem
cv2.imshow("gray image", gray)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Salvar a imagem processada :
cv2.imwrite('C:/.../ /pessoa_cinza.jpg'), gray)
```

```
(b, g, r) = cv2.split(image)

b = image[:, :, 0]
g = image[:, :, 1]
r = image[:, :, 2]

height,width =g.shape

image_copia = cv2.merge((b, g, r))
```

Detecção de contorno

- Ex. Gradiente: Sobe (*)l
  Canny edge detection

Segmentação

- Ex. Watershed (*)
  Region adjacency graph (RAG) + Normalized cuts
  Histogram of oriented gradients (HOG)

- Detecção de características
  - Haar Cascade
  - Scale-Invariant Feature Transform (SIFT)

Exemplos:

- contornos.txt
- watershed.txt
- ncut.txt
- frutas.txt
- detecção_face_olhos.txt
- sift.txt

Ball, John. E. et al. **Comprehensive survey of deep learning in remote sensing: theory, tools, and challenges for the community**. In: Journal of Applied. Remote Sensing. 11(4), 042609(2017), doi: 10.1117/1.JRS.11.042609.

Zhang, Liangpei et. al. **Deep Learning for Remote Sensing Data, a technical tutorial on state of the art**. in: IEEE Geoscience and Remote Sensing Magazine, vol. 4, no. 2, pp. 22-40, June 2016, doi: 10.1109/MGRS.2016.2540798.

Li, Y., Zhang, H., Xue, X., Jiang, Y., & Shen, Q. (2018). **Deep learning for remote sensing image classification**: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(6), [e1264].

Tsagkartakis, Grigorios et al. **Survey of Deep-Learning Approaches for Remote Sensing Observation Enhancement**. In: *Sensors*, 19, 3929 (2019), doi: 10.3390/s19183929.