Visão Computacional

Com Python e OpenCV

Professor: Austeclynio Pereira - 2024

Visão Computacional com Python - Essencial

- Início em 03/10/2024;
- Término em 05/11/2024;
  - Aulas em 03/10;**08/10;10/10**;17/10;**22/10;24/10**;31/10;05/11
- Horário: das 9h às 12h (8 sessões);
- Dias da semana: 3as e 5as feiras, observar as folgas;
- Avaliação: tarefas ao longo curso;

Bibliografia

1. Programming Computer Vision with Python – Solem, Jan Erik. Makron O'Reilly, 2012.

2. Computer Vision with OpenCV – Howe. Kenneth, Wiley, 2014

3. https://aws.amazon.com/pt/what-is/computer-vision/ em 29/01/2024.

4. https://aws.amazon.com/pt/rekognition/

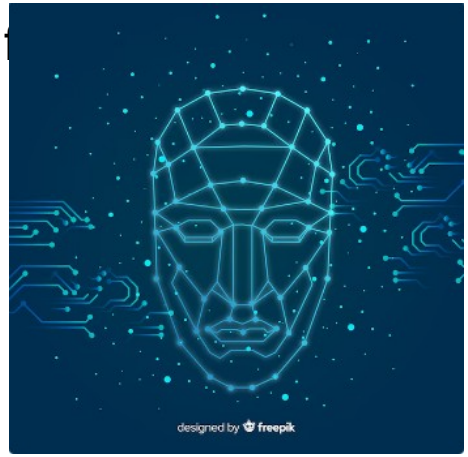
5. **https://opencv.org/**

Foco do Curso

- Capacitar para desenvolver soluções em visão computacional utilizando a linguagem Python e o OpenCV;
- O propósito é orientar ao uso das ferramentas existentes e não em criar novas ferramentas;
- Não será detalhado o processo de reconhecimento facial e sim o de identificação;

Identificação facial

- No rosto humano, é possível identificar, pelo menos, cerca de 80 pontos nodais que permitem medir variáveis como:
  - comprimento ou largura do nariz
  - distância entre os olhos
  - espessura dos lábios
  - tamanho da mandíbula etc.
- A identificação antecede ao reconhecimento f



Reconhecimento Facial

- Dividido nas seguintes tarefas:
  - Detecção da face
  - Extração das configurações da face usando *deep learning*
  - Treinamento do modelo de reconhecimento facial com as configurações obtidas
    - No treinamento do modelo apresentam-se imagens do que é(positivas) e as do que não é(negativas)
  - Reconhecimento da face alvo

O que é?

- A Visão Computacional é uma tecnologia que permite as máquinas identificarem e reconhecerem imagens automaticamente e descrevê-las com **exatidão** e **eficácia**;

- As aplicações de visão computacional usam IA e *machine learning* (ML) para processar imagens e dados de vídeo para **identificação de objetos em geral, humanos ou não humanos**;

- Possibilita a classificação, a recomendação, o monitoramento e a detecção.

- Os dados de imagens e vídeos não são estruturados e a organização deles pelos computadores é complexa. Portanto, as aplicações de visão eram caras e pouco acessíveis para a maioria das organizações.

- Com o aumento do poder computacional, e seu baixo custo, organizações passaram a usar a visão computacional em larga escala;

O que é?

- A Visão Computacional foi originalmente fundada como uma subdisciplina do campo da Inteligência Artificial na década de 1970 no MIT;
- O objetivo era criar um sistema que tivesse as mesmas capacidades perceptivas que o sistema visual humano possui;
- O sistema visual humano pode interpretar facilmente qualquer cena com pouco esforço;
- Discrimina perfeitamente entre milhares de categorias e pode encontrar objetos em cenas dentro de um intervalo de tempo de apenas centenas de milissegundos;
- Ele alterna facilmente entre vários tipos de processos de reconhecimento com flexibilidade e rapidez, **cuja complexidade e dinâmica não foram bem compreendidos ainda;**

O que é?

- Engloba conhecimentos de :
  - Cálculo Vetorial
  - Álgebra Linear
  - Probabilidade e Estatística
  - Equações Diferenciais
  - Geometria Diferencial
  - Análise Numérica
  - Geometria Euclidiana

O que é?

Como vemos.

232	182	143	151	151	148	148	143	145	139	143	136	139	136	134	132	129	130	126	124	115	116	115	104	109	102	100	101
244	218	160	149	145	147	145	143	139	142	140	139	134	134	130	131	125	120	120	116	110	110	107	100	100	97	95	97
246	233	196	145	145	146	141	141	137	134	140	133	133	125	131	125	114	121	116	116	109	101	95	101	97	87	89	91
248	242	222	161	142	140	145	137	138	135	129	127	127	122	124	118	116	113	102	110	99	102	98	94	91	88	91	90
252	246	234	192	143	139	136	134	133	129	131	127	124	121	117	114	111	105	108	95	101	102	86	88	91	84	84	99
252	249	242	215	151	137	134	134	129	126	126	121	120	116	113	111	108	104	99	94	102	93	89	96	79	87	92	112
252	248	242	227	169	134	135	124	122	120	125	121	116	115	105	112	102	99	92	98	93	88	89	74	87	65	97	111
253	246	244	236	192	134	125	123	119	120	118	116	112	107	110	95	104	94	89	96	84	86	79	77	65	79	105	119
252	250	246	238	210	144	126	118	120	115	116	116	98	105	103	102	96	93	91	82	80	79	75	70	82	81	108	119
250	251	247	239	219	161	127	117	117	109	105	107	100	104	99	100	98	79	98	70	75	80	72	65	86	83	113	124
252	249	247	241	226	177	122	120	116	106	108	110	91	103	93	99	89	88	79	80	72	74	76	65	84	87	109	123
249	249	247	241	231	191	131	116	110	109	106	98	95	90	102	83	78	86	85	80	70	75	69	81	79	96	122	121
248	249	247	244	238	206	133	121	101	103	94	97	91	87	87	83	83	82	77	78	81	61	73	65	86	99	118	120
247	250	248	244	237	215	149	115	102	105	91	94	80	91	79	83	81	70	71	75	74	71	78	74	76	108	117	119
250	247	246	243	239	218	159	108	100	87	100	88	92	83	85	77	81	63	80	70	63	73	70	78	81	110	120	116
248	245	244	241	239	224	170	113	103	94	89	86	84	83	74	81	68	78	76	66	66	70	73	65	92	108	115	123
248	244	244	242	237	226	179	123	98	94	84	74	88	77	71	76	71	78	68	67	63	72	72	75	94	109	115	124
247	244	245	241	238	221	183	123	95	87	89	73	77	79	71	65	78	56	69	66	62	61	70	69	90	113	118	118
247	246	244	242	236	219	185	120	100	84	82	79	66	67	76	72	69	55	61	56	65	57	70	78	95	106	119	116
246	245	244	241	231	216	190	126	91	86	77	77	72	71	76	60	69	60	57	52	66	55	62	75	87	110	110	117
245	244	244	237	231	221	189	133	97	83	70	73	62	59	77	44	65	66	60	70	51	43	67	75	95	107	116	111
244	244	241	237	230	222	188	133	90	83	77	77	59	78	60	67	62	61	66	72	62	51	62	71	96	105	115	108
242	242	237	236	232	219	187	126	83	79	70	64	58	66	63	67	54	65	51	65	58	54	62	73	77	92	107	94
241	241	238	236	229	216	186	125	85	77	70	66	64	53	63	55	56	53	67	39	52	25	23	9	11	51	66	77
241	239	237	237	228	214	185	127	92	83	64	66	69	62	61	65	32	42	12	7	6	15	65	123	146	160	167	172
240	239	237	236	225	208	178	123	89	67	72	67	49	54	27	10	7	23	103	142	162	167	169	168	171	172	172	178
238	236	236	229	221	203	174	125	77	82	55	33	23	9	79	135	163	173	174	175	174	170	171	167	167	172	169	173
235	235	231	228	215	198	165	122	84	43	14	57	132	166	176	175	179	177	176	178	178	173	169	172	167	168	171	162
231	231	227	223	210	191	163	110	44	95	159	174	175	179	178	180	183	180	179	177	175	175	174	173	169	168	171	156
230	226	225	220	202	187	169	151	175	180	182	177	182	182	183	184	184	184	181	182	181	178	182	179	172	161	160	155
223	224	220	213	198	191	185	186	182	182	178	179	184	185	191	189	189	192	188	192	193	194	192	187	179	161	153	147
220	219	213	203	191	182	181	177	176	173	175	180	182	184	192	192	193	195	200	203	203	206	205	202	192	164	150	151
212	209	200	188	177	173	174	171	169	165	173	176	180	187	191	192	195	195	201	203	207	210	208	212	201	177	147	143

Como o computador vê.

Como funciona?

- Usa a tecnologia de inteligência artificial (IA) para simular as capacidades do cérebro humano que são responsáveis pelo reconhecimento e classificação de objetos;

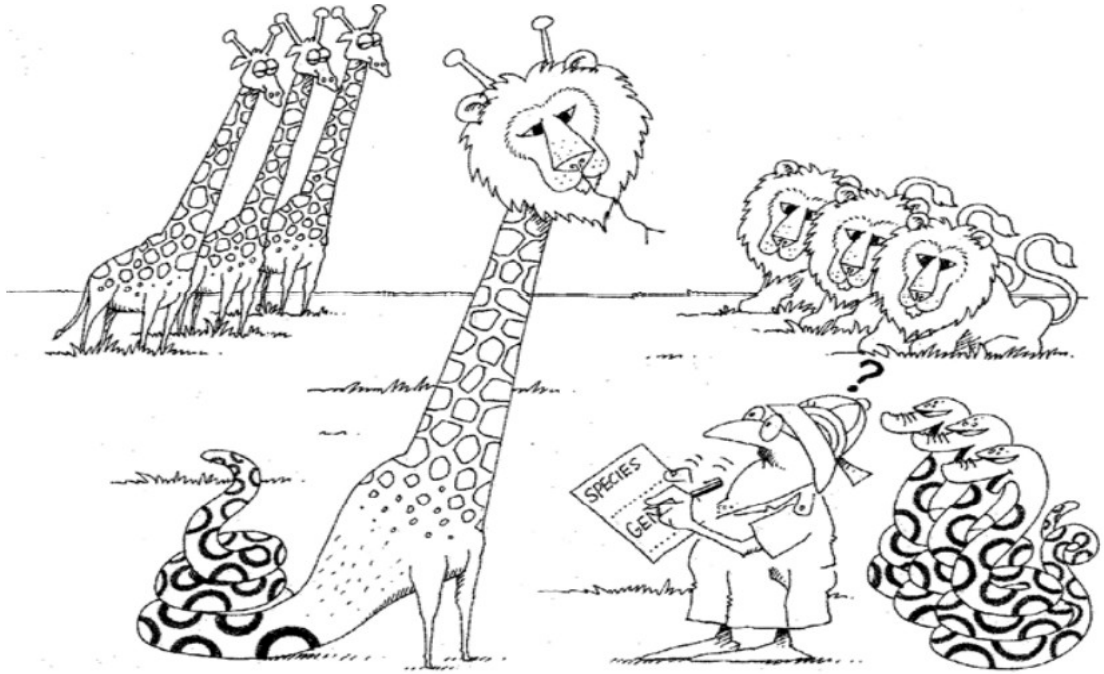


Figura extraída do livro Object-Oriented Analysis and Design with Applications

Como funciona?

- Computadores são treinados para reconhecer dados de imagens, produzindo grandes quantidades de informações;

- Os algoritmos de *machine learning* (ML) **identificam padrões comuns** nessas imagens ou vídeos e aplicam esse conhecimento para identificar imagens desconhecidas com exatidão;

- Por exemplo, se os computadores processarem milhões de imagens de cachorros, eles começarão a criar padrões de identidade que possam detectar, com exatidão, um cachorro, em uma imagem;

Como funciona?

- Em resumo, a visão computacional usa as seguintes tecnologias:
  - **Machine Learning(ML)**
    - O ML usa redes neurais. As redes neurais do ML são compostas de camadas de módulos de software chamados neurônios artificiais.
    - São usados cálculos matemáticos para processar, automaticamente, diferentes aspectos dos dados da imagem e desenvolver gradualmente uma compreensão combinada da imagem.
  - **Redes Neurais Convolucionais(CNN)**
    - São um subconjunto do aprendizado de máquina utilizadas com mais frequência para tarefas de classificação e visão computacional.
    - Oferecem uma abordagem mais escalável para tarefas de classificação de imagens e reconhecimento de objetos, aproveitando princípios da álgebra linear, especificamente a multiplicação de matrizes, para identificar padrões dentro de uma imagem.
    - Como um humano tentando reconhecer um objeto à distância, a CNN primeiro identifica contornos e formas simples antes de preencher detalhes adicionais, como cor, formas internas e textura. Por fim, ela repete o processo de previsão em várias iterações para melhorar a exatidão.

    Obs.: Convolução é uma operação matemática entre duas funções para produzir uma terceira.

Como funciona?

- Em resumo, a visão computacional usa as seguintes tecnologias(continuação):
  - **Redes Neurais Recorrentes(RNN)**
    - São semelhantes às CNNs, mas podem processar uma série de imagens para encontrar ligações entre elas. Embora as CNNs sejam usadas para análise de imagem única, as RNNs podem analisar vídeos e entender as relações entre as imagens.

Visão Computacional e o Processamento de Imagens

- Diferenças entre a visão computacional e o processamento de imagens:
  - O processamento de imagens usa algoritmos para alterar imagens, incluindo nitidez, suavização, filtragem ou aprimoramento.
  - A visão computacional não altera uma imagem, mas dá sentido ao que vê e realiza tarefas, como a rotulagem.
  - Em alguns casos, pode-se usar o processamento de imagem para modificar uma imagem para que um sistema de visão computacional possa entendê-la melhor.
  - Em outros casos, pode-se usar a visão computacional para identificar imagens ou partes de uma imagem e, em seguida, usar o processamento de imagem para modificá-la ainda mais.

Algumas tarefas da Visão Computacional

- A seguir, algumas das missões da visão computacional:
  - **Aquisição da imagem:**
    - Consiste na captação da imagem seja por câmeras, filmadoras, *scanners* etc.
  - **Classificação das imagens:**
    - Permite que os computadores vejam uma imagem e classifiquem com exatidão em qual classe ela se enquadra.
    - A visão computacional entende as classes e as rotula, por exemplo, como árvores, aviões ou edifícios. Um exemplo prático é uma câmera reconhecer rostos em uma imagem e enquadrá-los.
  - **Segmentação:**
    - Responsável por particionar a imagem em regiões de interesse. A segmentação pode reconhecer se há mais de um objeto em uma imagem ou quadro.
  - **Rastreamento de Objetos:**
    - Usa modelos de ML para identificar e rastrear itens pertencentes a categorias. Por exemplo, o rastreamento de objetos pode ser usado para vigilância humana e exames de imagens médicas.

Algumas tarefas da Visão Computacional

- Algumas das tarefas da visão computacional (continuação):
  - **Detecção de Objetos:**
    - Usa a classificação para reconhecer, classificar e organizar imagens.
    - A detecção de objetos é usada em processos industriais e de fabricação para controlar aplicações autônomas e monitorar linhas de produção.
    - Usada também para processar transmissões de vídeo ao vivo de câmeras para detectar pessoas e objetos em tempo real e fornecer alertas aos usuários finais.

Visão Computacional Aplicações

- Algumas aplicações da visão computacional:
  - **Agricultura**: Analisam a forma, a cor e a textura das culturas para análise posterior.
  - **Reconhecimento facial**: Utilizado em sistemas de segurança e acesso através da identificação de pessoas em imagens e vídeos.
  - **Carros autônomos**: Permite a percepção e o entendimento do ambiente em todo o redor do veículo(360º).
  - **Fábricas**: Aplicada na inspeção automática de produtos para garantir a qualidade e identificar defeitos em itens como alimentos, produtos eletrônicos e peças de automóveis.
  - **Imagens Médicas**: Usada em detecção de padrões e classificação de imagem para diagnósticos. Predominante em patologia, radiologia e oftalmologia. Microsoft InnerEye fornece diagnósticos rápidos e precisos.
  - **Educação**: Permite que os professores identifiquem aqueles alunos desatentos. Também pode inibir, por análise de comportamento corporal, práticas ilegais(colar, por exemplo).

Visão Computacional Aplicações

- Algumas aplicações da visão computacional(continuação):
  - **Análises médicas:** registro de imagens pré-operatórias e intra-operatórias, realizando estudos de longo prazo da morfologia cerebral das pessoas à medida que envelhecem; detecção de tumores; medição do tamanho e forma de órgãos internos; análise cromossômica; contagem de células sanguíneas.
  - **Segurança automotiva:** reconhecimento de sinais de trânsito, detecção de obstáculos inesperados, como pedestres na estrada ou rua.
  - **Vigilância**: monitoramento de intrusos, análise de tráfego rodoviário, monitoramento de piscinas para vítimas de afogamento;
  - **Reconhecimento de gestos**: identificando posturas de mão de fala em nível de sinal, identificando gestos para interação humano-computador ou teleconferência.
  - **Reconhecimento de impressões digitais e biometria**: autenticação automática de acesso, bem como aplicações forenses.
  - **Robótica**: reconhecimento e interpretação de objetos em uma cena, controle de movimento e execução através de *feedback* visual.

Visão Computacional Aplicações

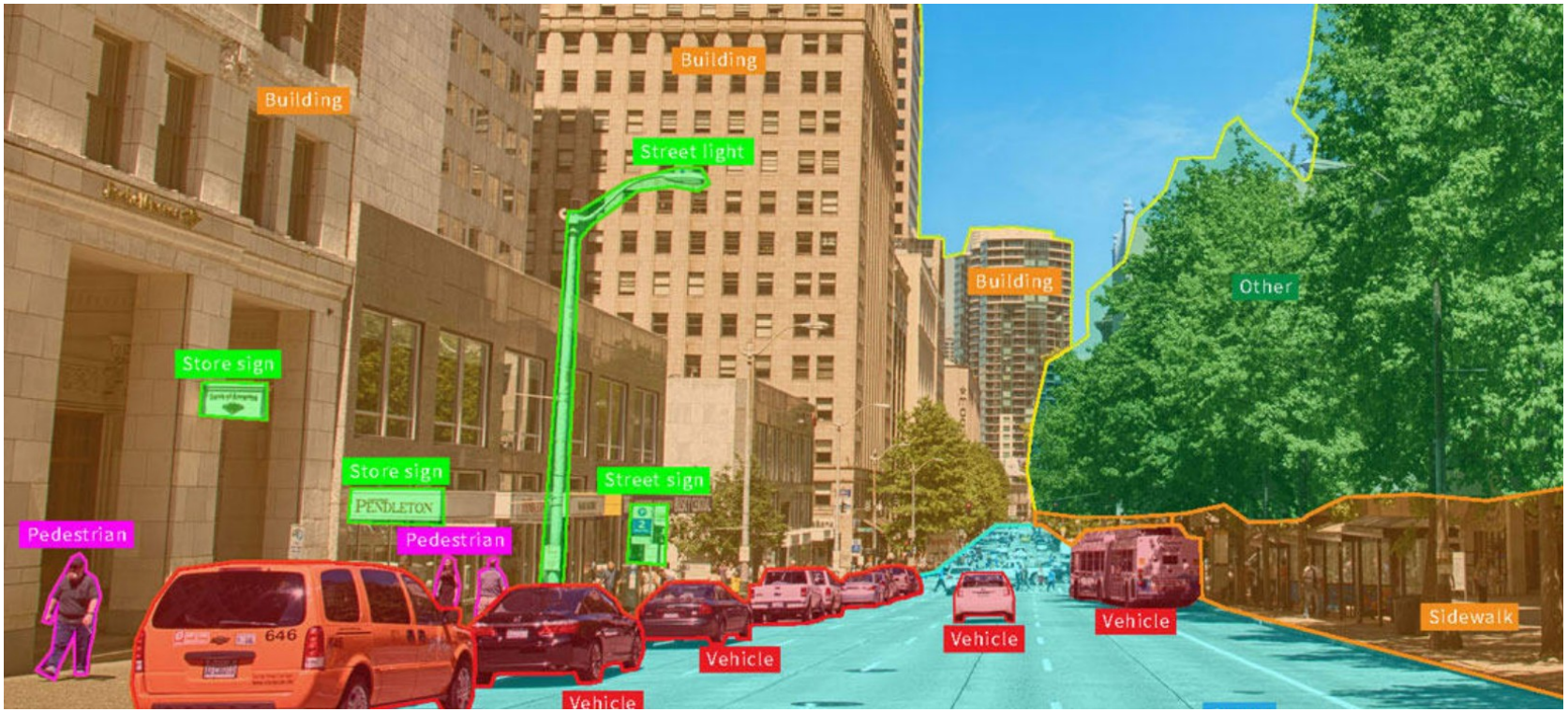
- Algumas aplicações da visão computacional(continuação):
  - **Cartografia:** elaboração de mapas a partir de fotografias, síntese de mapas meteorológicos.
  - **Imagens de radar:** detecção e identificação de alvos, orientação de helicópteros e aeronaves em pouso, orientação de veículos pilotados remotamente (RPV), mísseis e satélites a partir de sinais visuais.
  - **Sensoriamento remoto:** análise de imagens multi-espectrais, previsão do tempo, classificação e monitoramento de ambientes urbanos, agrícolas e marinhos a partir de imagens de satélite.
  - **Inspeção de máquinas:** inspeção de defeitos e falhas de peças:
    - inspeção rápida de peças para garantia de qualidade
    - usando visão estéreo com iluminação especializada para medir tolerâncias em asas de aeronaves ou carrocerias de automóveis
    - peças; ou procurar defeitos em peças fundidas de aço usando visão de raios X; identificação de peças em linhas de montagem.

Visão Computacional Aplicações

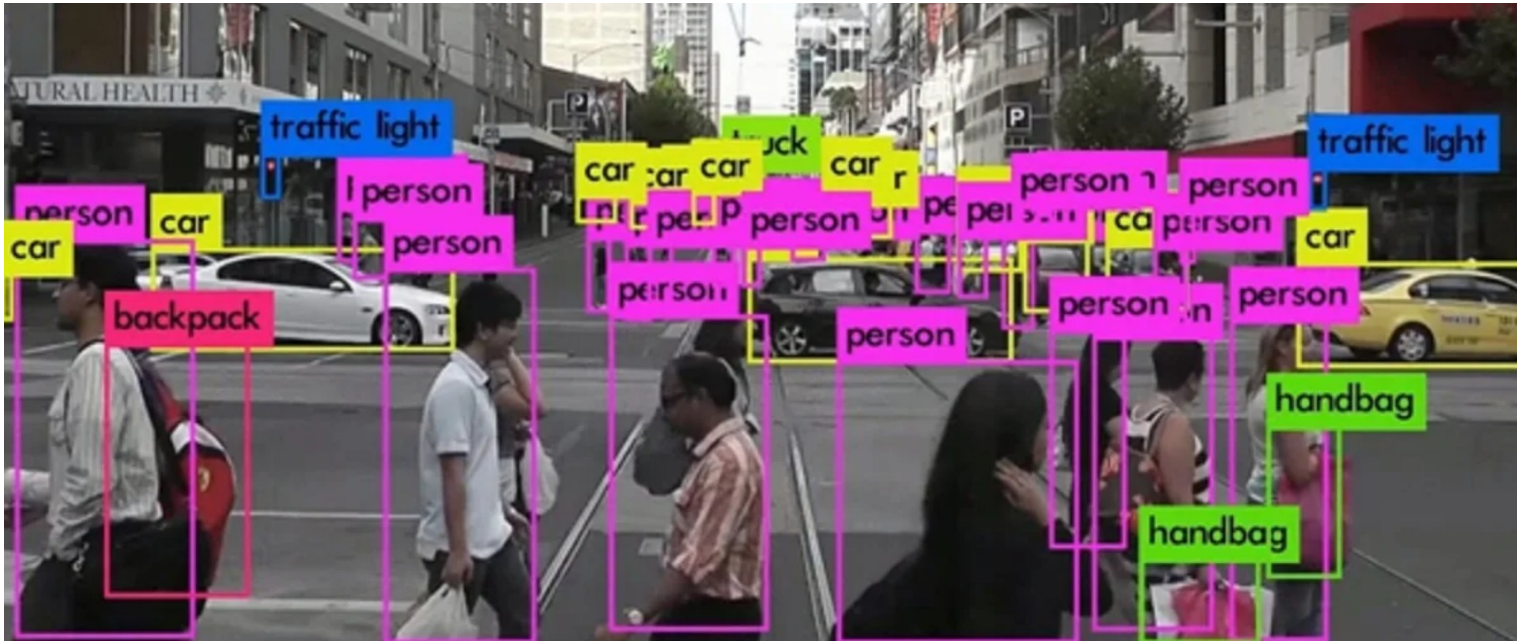
- Algumas aplicações da visão computacional(continuação):
  - **Transporte**: Usadas para detectar infrações por direção errada e em sistemas inteligentes de transporte para análise do fluxo de tráfego.
  - **Drones:** Para identificar bases inimigas e atacá-las.

Visão Computacional
Exemplos

Visão Computacional
Exemplos



Visão Computacional
Exemplos



Visão Computacional
Exemplos

Visão Computacional
Exemplos

Python Instalação

- Vá em www.python.org e clique na guia *downloads*;
- Baixar a versão mais atual, no momento do curso a 3.12.5;
- Após o *download*, executar o arquivo de instalação;
- Após a instalação, é criado um ícone na área de trabalho;
- Além do interpretador Python, também é instalada uma IDE que permite a escrita e a execução dos programas;

Python Instalação

- Instalar a biblioteca *readline* que implementa um conjunto de funções para recursos interativos de edição;
- Para tal, abra um *prompt* de comando do Windows;
- Digitar:
  - pip install pyreadline

OpenCV

- OpenCV (Open Source Computer Vision Library) é uma biblioteca, com uma grande gama de funções, orientada à visão computacional;
- Originalmente desenvolvido pela Intel;
- É multiplataforma e licenciada como software gratuito e de código aberto sob a Licença Apache 2;
- A partir de 2011, OpenCV passa a utilizar os recursos da GPU(Graphics Processing Unit) para operações em tempo real;
- Integração com as linguagens Python, Java e MATLAB;
- Documentação em https://opencv.org/;

OpenCV
Instalação sob o Python

- Para tal, abrir um *prompt* de comando do Windows e digitar:
  - **pip install opencv-python**
- A biblioteca NumPy também será instalada nesta operação, uma vez que o OpenCV dela depende;
- NumPy é uma biblioteca que suporta o processamento de grandes, e multi-dimensionais, *arrays*;
- O NumPy tem como objetivo fornecer um objeto *array* até 50x mais rápido que as listas tradicionais em Python;
- Arrays são usados com muita frequência em ciência de dados, onde velocidade e recursos computacionais são muito importantes.
- Ciência de Dados: é um ramo da ciência da computação onde estuda-se como armazenar, usar e analisar dados e derivar informações deles.

Conceitos básicos sobre Imagens

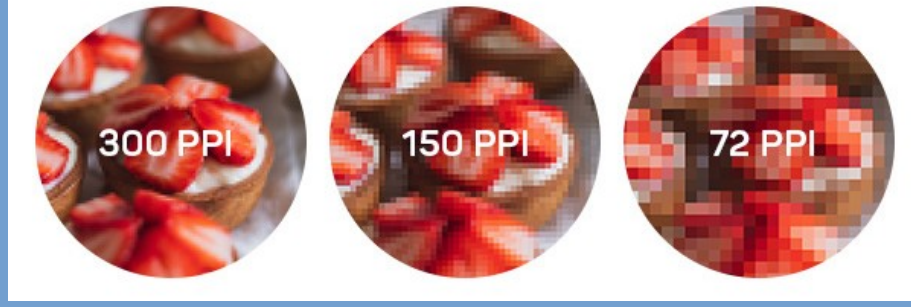
- Uma imagem é composta por pixels;
- Um pixel(*picture element*) é o menor ponto que forma uma imagem digital, sendo que um conjunto de pixels, com várias cores, formam a imagem inteira;
- O pixel é atômico, não existe nada menor que um pixel;
- Uma imagem com resolução de 1920 x 1080, significa que a imagem possui 1920 colunas e 1080 linhas de pixels. Total de 2.073.600 pixels;
- A título de comparação, uma tela 4K (3.840 x 2.160 pixels) conta com 8.294.400 pixels e uma 8K (7.680 x 4.320 pixels) 33.177.600 pixels;

Conceitos básicos sobre Imagens

- A nitidez de uma imagem depende de quantos pixels a compõem;
- Logo, quanto maior a densidade de pixels por polegada (PPI) de uma tela, maior o potencial que esse dispositivo tem de exibir imagens bem definidas;
- **Pixels podem ter tamanhos variados, e é por isso que telas do mesmo tamanho podem ter resoluções diferentes;**
- Por exemplo, em um quadrado de uma tela de 4K cabem quatro vezes mais pixels do que em uma Full HD;
- Diminuindo o tamanho do pixel, é possível inserir mais pixels na imagem, obtendo maior precisão no controle de cores;

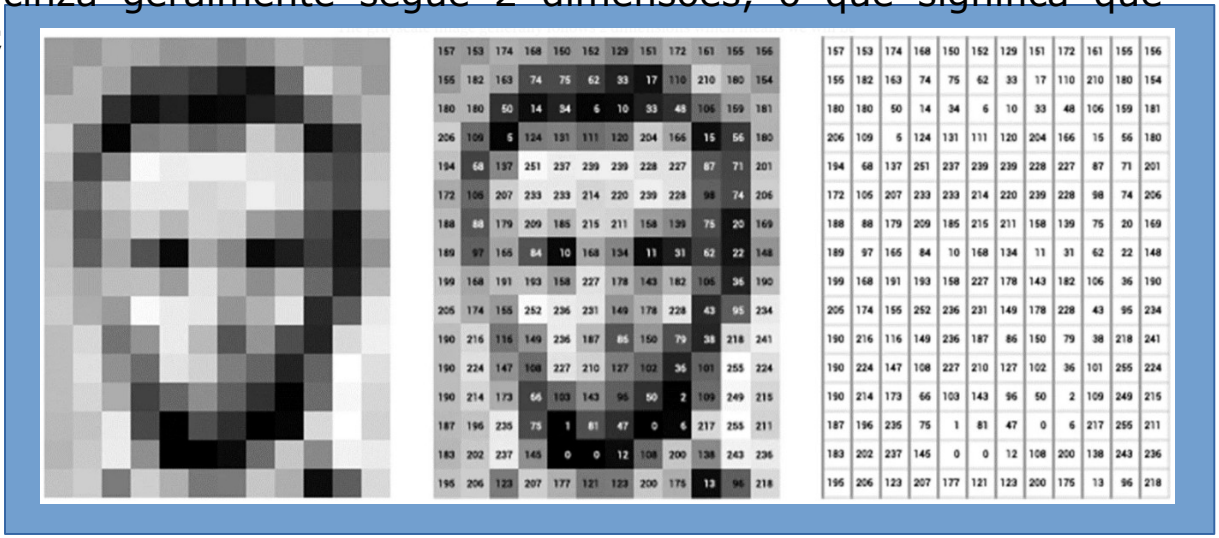
Conceitos básicos sobre Imagens

- Em câmeras, o termo megapixel(MP) associa-se à capacidade do sensor de capturar pixels em uma foto;
- Uma câmera com 48 MP é capaz de captar imagens com até 8.000 pixels de largura por 6.000 de altura.
- DPI (dots per inch) indica a definição de uma imagem impressa;
- PPI (pixels per inch) indica a definição de uma imagem digital;



Conceitos básicos sobre Imagens

- Os pixels são representados de três formas: binária, coloridos ou em tons de cinza(grayscale);
- Na representação binária é 0, para a cor branca, ou 1 para a cor preta;
- Em uma imagem em tom de cinza (grayscale), cada pixel possui um valor de 0 a 255;
- O 0 corresponde ao “preto” e 255 ao “branco”. Os valores entre 0 e 255 representam os tons de cinza;
- A imagem em tons de cinza geralmente segue 2 dimensões, o que significa que teremos linhas e colunas;

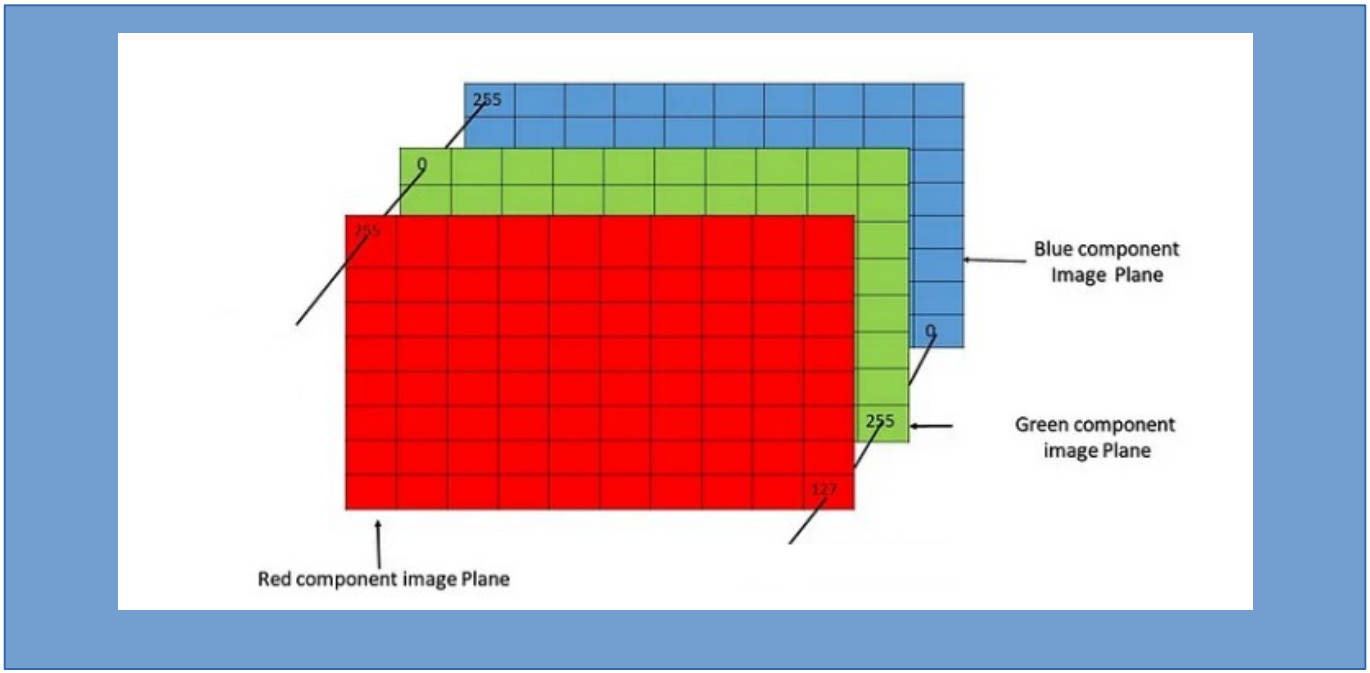


Conceitos básicos sobre Imagens

- Imagens coloridas são bastante diferentes para o processador ler quando comparadas às imagens em escala de cinza;
- As imagens coloridas têm **3 canais** diferentes chamados RED, GREEN e BLUE, também chamados de canais RGB;
- No OpenCV utiliza-se o padrão **BGR**;
- Cada uma das cores é representada por um número inteiro de 0 a 255, que indica "quanto"(ou partes) dessa cor existe;
- Geralmente são utilizados inteiros de 8-bits para representar as intensidades de cores;
- Combina-se então esses valores em uma tupla RGB na forma (red,green,blue);
- Outros modelos de cor: HSV(hue, saturation e brightness) e CMYK(Cyan, Magenta, Yellow e Black);

Conceitos básicos sobre Imagens

- Na visão computacional, esta tupla será representada por 3 diferentes matrizes, uma para cada cor;
- Um pixel, em determinada posição, seria representado, por exemplo, como: (127,255,0);



Conceitos básicos sobre Imagens

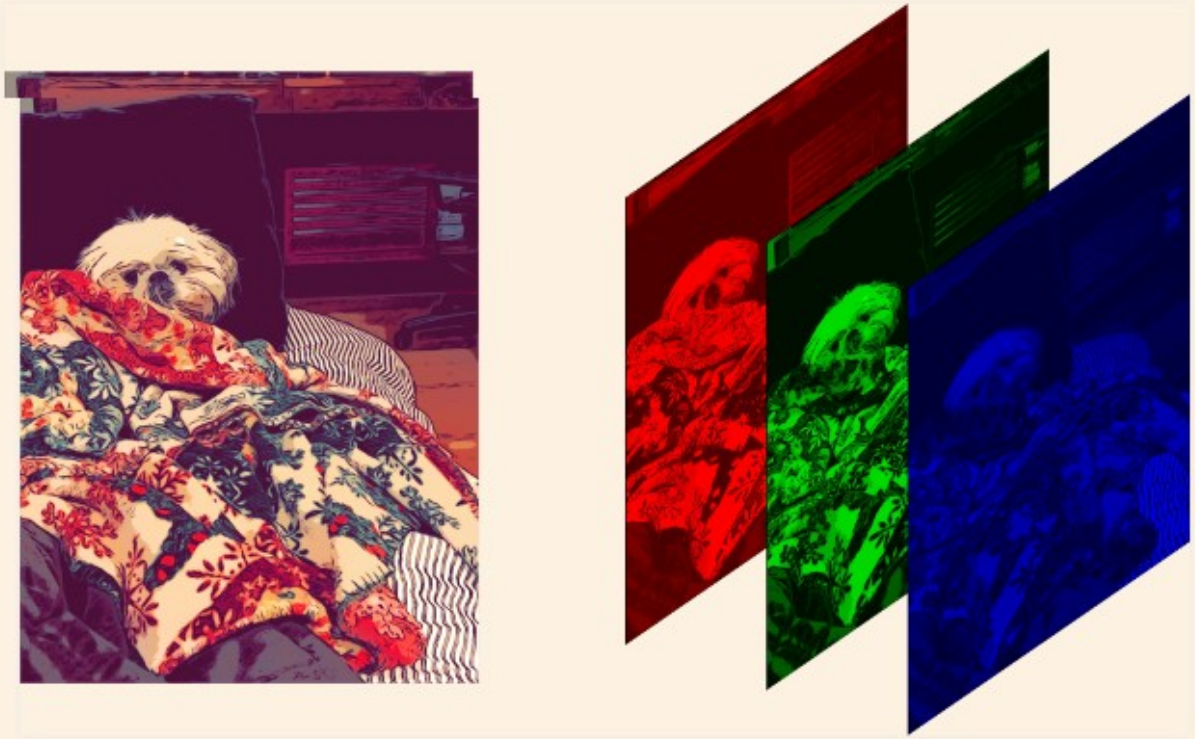
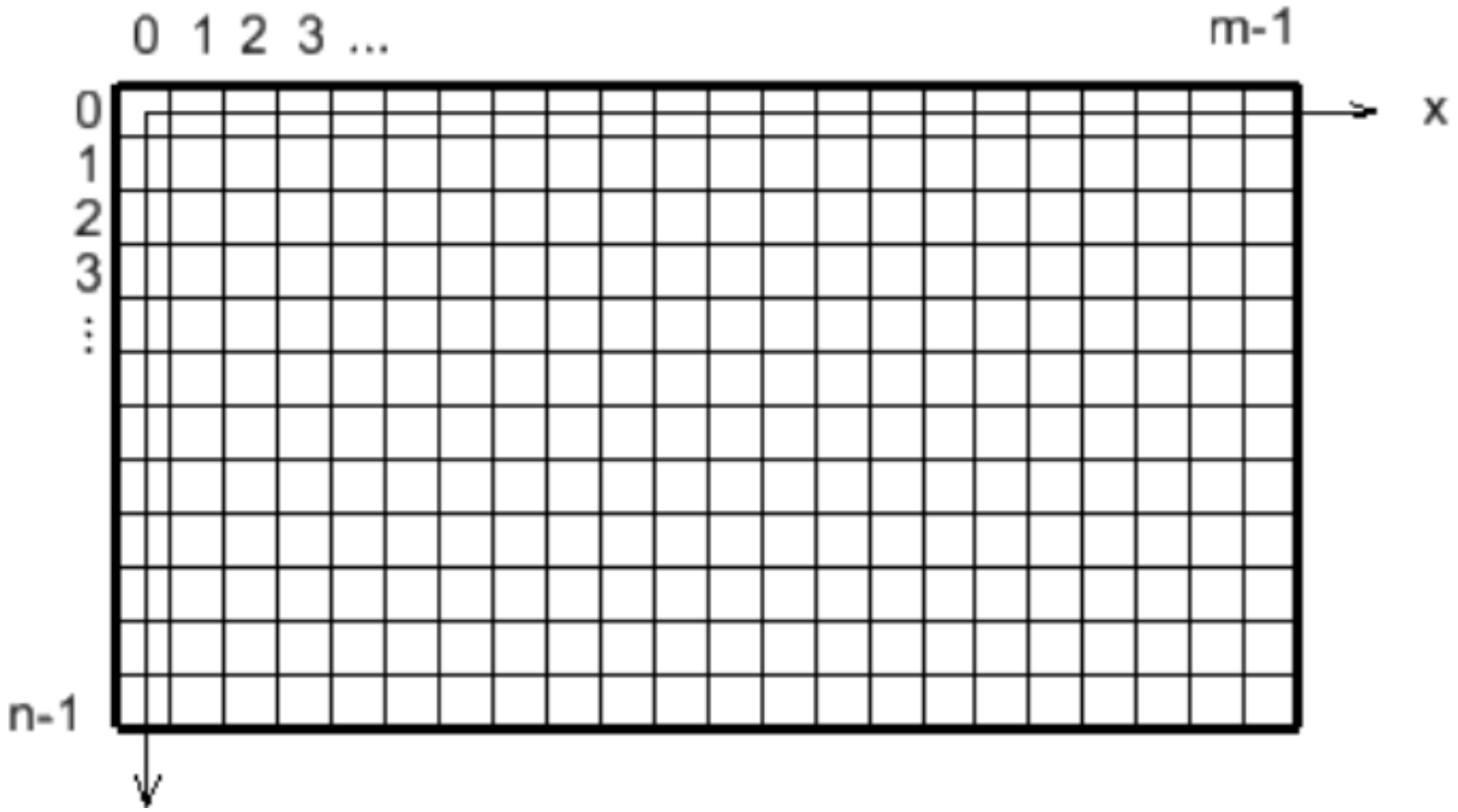


Image credit: Diane Rohrer

Pixels Coordenadas



OpenCV
Algumas funções

imread(filename,flags)	Lê uma imagem com filename. Flags: 0 – em grayscale; 1 – em cores
imshow(window-name,image)	Apresenta a imagem em uma janela do SO de nome window-name.
waitKey(time-in-milliseconds)	Tempo de exposição da imagem. Se 0, tempo ilimitado.
destroyAllWindows()	Libera todos os recursos alocados.
imwrite(filename, img)	Grava a imagem. Suporta *.bmp, *.dib , *.jpeg, *.jpg, *.png,*.webp, *.sr,*.tiff, *.tif
shape	Informa as dimensões, em **pixels(y,x,n)**, e o número de canais usados na imagem, se houver.
size	Informa o tamanho, em pixels(**y * x**), da imagem.

```
import numpy as np
import cv2
# Load a color image in grayscale
img = cv2.imread('Visao_Computacional_Logo_Imagem.png',0)
assert img is not None, "Arquivo não encontrado!"
print(img.shape)   # y,x
print(img.size)
cv2.imshow('OpenCV - GrayScale',img)
cv2.waitKey(5000)
# Load a color image
img = cv2.imread('Visao_Computacional_Logo_Imagem.png',1)
print(img.shape)  # y,x,n
print(img.size)
cv2.imshow('OpenCV - Color',img)
cv2.waitKey(5000)
#Write image
cv2.imwrite("Visao_Computacional_Logo_Imagem_Copia.png", img)
cv2.destroyAllWindows()
```

Executar: VisaoComputacional_OpenCV_Exemplo01.py

OpenCV algumas funções exemplos

- Carregando e modificando uma figura (tamanho 143(y) x 237(x)):

```
import numpy as np
import cv2
# Load a color image
img = cv2.imread('BGR_Azul_Verde_Vermelho.png',1)
print(img.shape)
# General Pixel Value (BGR)
print(img[100,70])   # img[y,x]
print(img[100,210])
print(img[100,140])

# By Matrix
print(img[100,70,0])  # img[y,x,n]
print(img[100,70,1])
print(img[100,70,2])

# By Matrix
print(img[140,140,0])
print(img[140,140,1])
print(img[140,140,2])

# By Matrix
print(img[50,210,0])
print(img[50,210,1])
print(img[50,210,2])
```

Executar: VisaoComputacional_OpenCV_Exemplo02.py

```
cv2.imshow('OpenCV - Color',img)

cv2.waitKey(5000)


# Modify Pixel Color

for i in range(100):     # y

    for j in range(100):  # x

        img[i,j]=[0,0,0]  # altera para cor preta



cv2.imshow('OpenCV - Color _ Modified',img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OpenCV algumas funções exemplos

- Outra maneira:

```
import numpy as np
import cv2
# Load a color image
img = cv2.imread('BGR_Azul_Verde_Vermelho.png',1)
cv2.imshow('OpenCV - Color _ Modified',img)
cv2.waitKey(2000)
# By Matrix
print("@Inicio normal")
print(img.item(100,70,0))   # pixel específico
print(img.item(100,70,1))   # pixel específico
print(img.item(100,70,2))   # pixel específico

print(img[100,70])
print("@Fim normal")

for i in range(img.shape[0]): # número de linhas
        for j in range(img.shape[1]): #número de colunas
                img.itemset((i,j,2),0)   # modifica
cv2.imshow('OpenCV - Color _ Modified',img)
print("@Inicio preto")
print(img.item(100,140,0))
print(img.item(100,140,1))
print(img.item(100,140,2))

print(img[100,140])
print("@Fim preto")
cv2.waitKey(10000)

for i in range(img.shape[0]):
        for j in range(img.shape[1]):
                img.itemset((i,j,2),255)
cv2.imshow('OpenCV - Color _ Modified',img)
print("@Inicio vermelho")
print(img.item(100,140,0))
print(img.item(100,140,1))
print(img.item(100,140,2))

print(img[100,140])
print("@Fim vermelho")
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar:

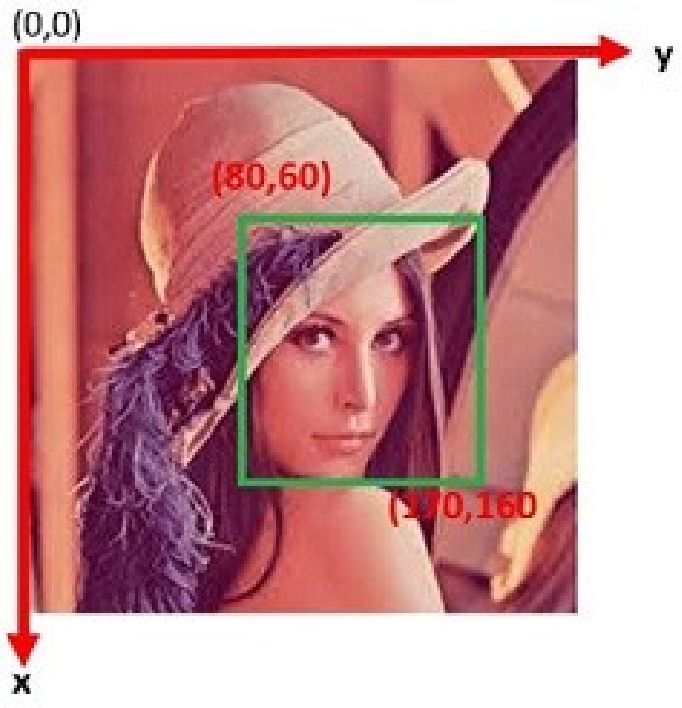
VisaoComputacional_OpenCV_Exemplo03.py

OpenCV
ROI - Region of Interest

- Permite obter acesso a algumas partes da imagem(região);
- Essa região é chamada de Região de Interesse(ROI);
- Aqui pode-se obter acesso a mais de um pixel por vez;
- Para tal, indicam-se o início e o fim da região, formando um quadrilátero;
- A região pode ser tanto lida quanto modificada;

OpenCV
ROI - Region of Interest

- Para selecionar a ROI, deve-se mencionar o intervalo de linhas (xi:xf) e colunas (yi:yf) que constitui a região de interesse;
- Na figura do próximo *slide,* com 253 por 424 *pixels,* a ROI é o cálice de cachaça;
- **Atenção aos eixos!**



OpenCV
ROI - Region of Interest

OpenCV
ROI - Region of Interest

- Exemplo:

```
import cv2
import numpy as np
img = cv2.imread('ROI_Conjunto_Bebidas.png',cv2.IMREAD_COLOR)
print(img.shape)
ROI = img[180:230,160:190]
cv2.imshow('ROI',ROI)
cv2.waitKey(5000)


img[180:230,160:190] = 255
cv2.imshow('ROI',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar:

VisaoComputacional_OpenCV_Exemplo04.py

OpenCV
ROI - Region of Interest

- Exercício 1:

  Isolar a bebida de seu interesse(não vale a cachaça!).

OpenCV
split e merge de imagens

- Os canais da imagem podem ser divididos em planos individuais usando a função split().
- Os canais podem ser mesclados usando a função merge().
- A função split() retorna um array multicanal.

OpenCV
Desenhando formas e texto

- Exemplo, retângulos e texto:

```
import cv2
import numpy as np

img = cv2.imread('ROI_Conjunto_Bebidas.png',cv2.IMREAD_COLOR)
image = cv2.resize(img, (1280, 720))

pt1 = (400, 40)
pt2 = (800, 300)
color = (0, 255, 255)
thickness = 1
lineType = cv2.LINE_4

img_rect = cv2.rectangle(image, pt1, pt2, color, thickness, lineType)
cv2.imshow("Conjunto_Bebidas", img_rect)
cv2.waitKey(2000)

thickness = -1
img_rect = cv2.rectangle(image, pt1, pt2, color, thickness, lineType)
cv2.imshow("Conjunto_Bebidas", img_rect)
cv2.waitKey(3000)

text = "Minhas Bebidas!!!"
onde = (450, 170)
fontFace = cv2.FONT_HERSHEY_SIMPLEX
fontScale = 1
color = (0,0,255)
img_text = cv2.putText(img_rect, text, onde, fontFace, fontScale, color, lineType)
cv2.imshow("Conjunto_Bebidas",img_text)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar: VisaoComputacional_OpenCV_Exemplo7.py

OpenCV
Desenhando formas e texto

- Exemplo, linhas e círculos:

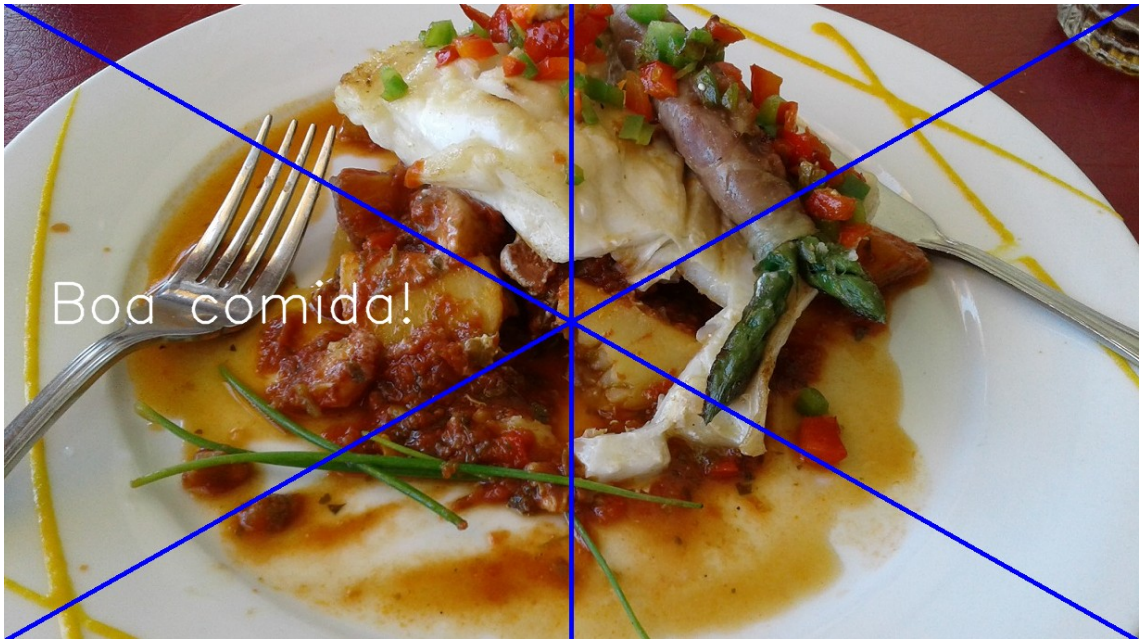
```
import numpy as np
import cv2 as cv
# Create a black image
img = np.zeros((512,512,3), np.uint8)

cv.line(img,(0,0),(511,511),(255,0,0),5)
cv.circle(img,(447,63), 63, (0,0,255), -1)

for x in range(63, img.shape[0], 63):
    cv.circle(img,(x,189), 63, (0,255,0), 1)

#Para desenhar um polígono, primeiro precisa-se das coordenadas dos vértices.
#Transforme esses pontos em uma matriz de formato ROWSx1x2 onde ROWS é o número de vértices e deve ser do tipo int32.
#Aqui desenha-se um pequeno polígono com quatro vértices na cor amarela.
pts = np.array([[10,50],[20,80],[70,40],[90,10]], np.int32)
cv.polylines(img,[pts],True,(0,255,255))

font = cv.FONT_HERSHEY_SIMPLEX
cv.putText(img,'Formas',(10,500), font, 4,(255,255,255),2,cv.LINE_AA)

cv.imshow("Desenhando_Formas",img)
print (img.shape)
cv.waitKey(0)
cv.destroyAllWindows()
```

Executar:
VisaoComputacional_OpenCV_Exemplo71.py

OpenCV
Desenhando formas e texto

- Exercício 2:

  Usando a figura, realizar os desenhos.



OpenCV
Desenhando formas e texto

- Exercício 2 – Uma solução:

```
import numpy as np
import cv2 as cv

image = cv.imread('VisaoComputacional_Imagem_Refeicao_Chile.jpg',1)
assert image is not None, "Arquivo não encontrado!"
y,x,c = image.shape
print(image.shape)
cv.line(image,(0,0),(x-1,y-1),(255,0,0),4)
cv.line(image,(x-1,0),(0,y-1),(255,0,0),4)
a=int(x/2)
b=int(y/2)
cv.line(image,(a,0),(a,y-1),(255,0,0),4)

font = cv.FONT_HERSHEY_SIMPLEX
cv.putText(image,'Boa comida!',(50,b), font,2,(255,255,255),2,cv.LINE_AA)
cv.imshow('OpenCV - Color',image)


#cv.imwrite("VisaoComputacional_Imagem_Refeicao_Chile_Copia.png", image)
cv.waitKey(0)
cv.destroyAllWindows()
```

OpenCV
resize e rotate

- É possível aumentar ou diminuir uma imagem com o uso da função resize();
- Diminuindo, ajuda a reduzir o tempo de treinamento de uma rede neural;
- Quanto maior o número de pixels, maior o número de nós de entrada, o que aumenta a complexidade do modelo;
- É importante ter em mente a proporção original da imagem, se desejar manter a mesma na imagem redimensionada também.
- Reduzir o tamanho de uma imagem exigirá reorganização dos pixels.
- Aumentar o tamanho de uma imagem exige reconstrução da imagem. Isso implica na necessidade de interpolar novos pixels.

OpenCV
resize e rotate

- Sintaxe:
  - resize(img, dsize, dest, fx, fy, interpolation)
    - img – é a imagem.
    - dest – tamanho do array de saída(opcional).
    - dsize - é o tamanho desejado para a imagem. Pode ser uma nova altura e largura.
    - fx: fator de escala para o eixo Horizontal(opcional).
    - fy: fator de escala para o eixo Vertical(opcional).
    - interpolação: forma de interpolação(opcional).

OpenCV
resize e rotate

- Em matemática, denomina-se interpolação o método que permite construir um novo conjunto de dados a partir de um conjunto discreto de dados pontuais, previamente conhecidos;
- Tipos de interpolação:
  - INTER_NEAREST - Uma interpolação do vizinho mais próximo.
  - INTER_LINEAR - Uma interpolação bilinear (usada por padrão)
  - INTER_AREA -Reamostragem usando relação de área de pixel. É um método preferido para dizimação da imagem, mas quando a imagem é ampliada, é semelhante ao, Método INTER_NEAREST.
  - INTER_CUBIC - Uma interpolação bicúbica sobre uma vizinhança de 4x4 pixels
  - INTER_LANCZOS4 - Uma interpolação Lanczos sobre uma vizinhança de 8x8 pixels
- Os métodos de interpolação preferíveis são o INTER_AREA, para redução, o INTER_CUBIC e o INTER_LINEAR para zoom;

OpenCV
resize e rotate

- Exemplo, usando alturas e larguras nas novas imagens:

```
import cv2
import numpy as np

image = cv2.imread('VisaoComputacional_Imagem_RottWeiler.jpg')
cv2.imshow('Original Image', image)
print (image.shape)

# Reduz a imagem
down_width = 300
down_height = 200
down_points = (down_width, down_height)
resized_down = cv2.resize(image, down_points, interpolation= cv2.INTER_LINEAR)

# Aumenta a imagem
up_width = 900
up_height = 450
up_points = (up_width, up_height)
resized_up = cv2.resize(image, up_points, interpolation= cv2.INTER_LINEAR)

# Apresenta imagens
cv2.imshow('Resized Down by defining height and width', resized_down)
cv2.waitKey()

#Tecle enter para ver
cv2.imshow('Resized Up image by defining height and width', resized_up)
cv2.waitKey()

#press any key to close the windows
cv2.destroyAllWindows()
```

Executar:

OpenCV
resize e rotate

- Exemplo, usando escalas nas novas imagens:

```
import cv2
import numpy as np

image = cv2.imread('VisaoComputacional_Imagem_RottWeiler.jpg')
cv2.imshow('Original Image', image)
print (image.shape)
cv2.waitKey()

# Reduz a imagem 0.5 vez
scale_down = 0.5
scaled_f_down = cv2.resize(image, None, fx= scale_down, fy= scale_down, interpolation= cv2.INTER_LINEAR)

# Aumenta a imagem 1.5 vezes
scale_up_x = 1.5
scale_up_y = 1.5
scaled_f_up = cv2.resize(image, None, fx= scale_up_x, fy= scale_up_y, interpolation= cv2.INTER_LINEAR)

# Apresenta imagens
cv2.imshow('Resized Down by defining height and width', scaled_f_down)
cv2.waitKey()

#Tecle enter para ver
cv2.imshow('Resized Up image by defining height and width', scaled_f_up)
cv2.waitKey()

#press any key to close the windows
cv2.destroyAllWindows()
```

Executar: VisaoComputacional_OpenCV_Exemplo81.py

OpenCV
resize e rotate

- A translação e a rotação de imagens estão entre as operações mais básicas na edição de imagens;
- Ambas se enquadram na classe mais ampla de transformações afins;
- O OpenCV usa funções de transformação afins para operações de translação e rotação;
- A transformação afim é uma transformação que pode ser expressa na forma de uma multiplicação de matrizes seguida por uma adição de vetores;
- Existem duas funções warpAffine() e warpPerspective(), onde pode-se ter todos os tipos de transformações;
- Para encontrar a matriz de transformação para rotação, existe a função getRotationMatrix2D();

OpenCV
resize e rotate

- A rotação é uma operação de três etapas:
  - Primeiro, obtém-se o centro de rotação. Normalmente é o centro da imagem que se está tentando girar.
  - A seguir, cria-se a matriz de rotação 2D. O OpenCV fornece a função getRotationMatrix2D() para este propósito.
  - Por fim, aplica-se a transformação afim à imagem, usando a matriz de rotação que criada na etapa anterior.
  - A função warpAffine() no OpenCV faz este trabalho.
- Após aplicar a transformação afim, todas as linhas paralelas na imagem original permanecerão paralelas também na imagem de saída;

OpenCV
resize e rotate

- Função para a rotação 2D:
  - getRotationMatrix2D(centro, ângulo, escala)
- Onde:
  - centro: o centro de rotação da imagem de entrada
  - ângulo: o ângulo de rotação em graus
  - escala: um fator de escala que aumenta ou diminui a imagem de acordo com o valor fornecido
  - Se o ângulo for positivo, a imagem será girada no sentido anti-horário
  - Se o ângulo for negativo, a imagem será girada no sentido horário

OpenCV
resize e rotate

- A função warpAffine(),sintaxe;
  - warpAffine(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]])
- Onde:
  - src: a imagem de entrada
  - M: a matriz de transformação
  - dsize: tamanho da imagem de saída
  - dst: a imagem de saída
  - flags: combinação de métodos de interpolação como INTER_LINEAR ou INTER_NEAREST
  - borderMode: o método de extrapolação de pixels
  - borderValue: o valor a ser utilizado no caso de borda constante, tem valor padrão 0

OpenCV
resize e rotate

- Exemplo, usando escalas nas novas imagens:

```
import cv2

# Reading the image
image = cv2.imread('VisaoComputacional_Imagem_RottWeiler.jpg')
cv2.imshow('Original Image', image)
print (image.shape)
cv2.waitKey()


# dividing height and width by 2 to get the center of the image
height, width = image.shape[:2]
print(image.shape[:2])
# get the center coordinates of the image to create the 2D rotation matrix
center = (width/2, height/2)

# using cv2.getRotationMatrix2D() to get the rotation matrix
rotate_matrix = cv2.getRotationMatrix2D(center=center, angle=45, scale=1)

# rotate the image using cv2.warpAffine
rotated_image = cv2.warpAffine(src=image, M=rotate_matrix, dsize=(width, height))

cv2.imshow('Original image', image)
cv2.imshow('Rotated image', rotated_image)
# wait indefinitely, press any key on keyboard to exit
cv2.waitKey()

#press any key to close the windows
cv2.destroyAllWindows()
```

Executar: VisaoComputacional_OpenCV_Exemplo82.py

OpenCV
resize e rotate

- Exercício 3:

  Usando as figura da refeição no Chile, pratique o resize e o rotate. Altere escalas, ângulos para a rotação etc.

OpenCV
Thresholding

- *Thresholding* é uma função que consiste em converter uma imagem com tons de cinza em uma imagem binária com base em um valor limiar;
- Para cada pixel, o mesmo valor de *threshold* é comparado. Se o valor do pixel for menor que o *threshold*, ele recebe o valor 0, caso contrário, ele recebe um valor máximo;
- Algoritmos de *thresholding* pegam uma imagem de origem(src) e um valor de limite (*thresh*) como entrada e produzem uma imagem de saída (dst);
- Caso src(x,y) > *thresh*, então dst(x,y) recebe algum valor, denominado valor máximo(maxValue). Caso contrário, dst(x,y) receberá o valor 0;

OpenCV Thresholding

- Sintaxe:

  img_ret,th = threshold(src, thresh, maxval, type, dst)

  - Onde:

    src: array de entrada

    dst: array de saída, com o mesmo tamanho

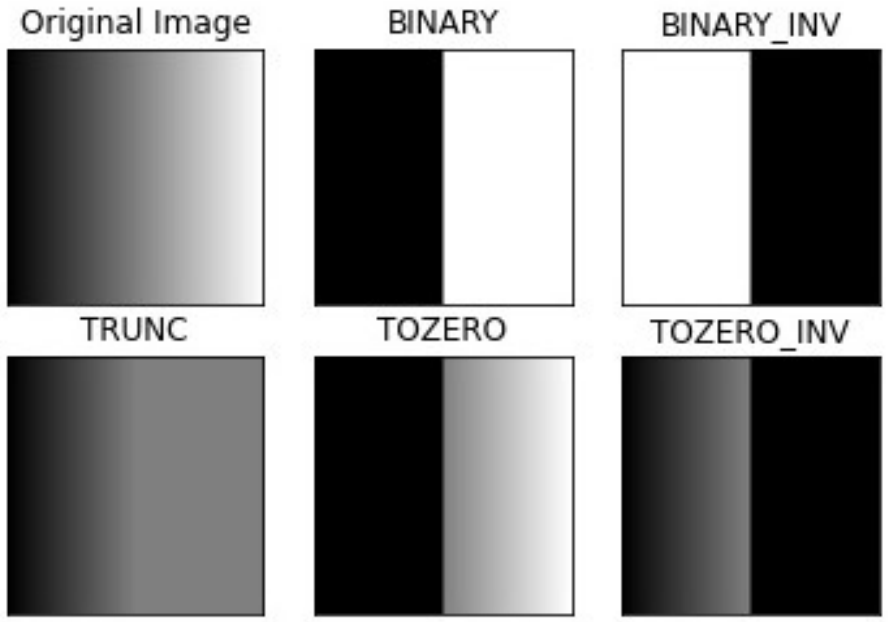
    thresh: valor do threshold

    maxval: valor máximo

    type: tipo do thresholding (THRESH_BINARY,THRESH_BINARY_INV,THRESH_TRUNC, THRESH_TOZERO,THRESH_TOZERO_INV)

  - A função threshold() retorna o threshold usado e a imagem produzida.

OpenCV Thresholding



OpenCV Thresholding

- Exemplo:

```
import cv2

# Read image
src = cv2.imread('VisaoComputacional_Imagem_RottWeiler.jpg',cv2.IMREAD_GRAYSCALE);

# Basic threhold example
th, dst = cv2.threshold(src, 0, 255, cv2.THRESH_BINARY);
print('THRESH_BINARY')
cv2.imwrite('VisaoComputacional_Imagem_RottWeiler-Threshold-Example.jpg', dst);
img = cv2.imread('VisaoComputacional_Imagem_RottWeiler-Threshold-Example.jpg')
cv2.imshow('Imagem_RottWeiler-Threshold-Example',img)
cv2.waitKey()

# Thresholding with maxValue set to 128
th, dst = cv2.threshold(src, 0, 128, cv2.THRESH_BINARY);
cv2.imwrite("VisaoComputacional_Imagem_RottWeiler-thresh-binary-maxval.jpg", dst);
img = cv2.imread('VisaoComputacional_Imagem_RottWeiler-thresh-binary-maxval.jpg')
cv2.imshow('Imagem_RottWeiler-Threshold-Example',img)
cv2.waitKey()

# Thresholding with threshold value set 127
th, dst = cv2.threshold(src,127,255, cv2.THRESH_BINARY);
cv2.imwrite("VisaoComputacional_Imagem_RottWeiler-thresh-binary.jpg", dst);
img = cv2.imread('VisaoComputacional_Imagem_RottWeiler-thresh-binary.jpg')
cv2.imshow('Imagem_RottWeiler-Threshold-Example',img)
cv2.waitKey()

# Thresholding using THRESH_BINARY_INV
th, dst = cv2.threshold(src,127,255, cv2.THRESH_BINARY_INV);
print('THRESH_BINARY_INV')
cv2.imwrite("VisaoComputacional_Imagem_RottWeiler-thresh-binary-inv.jpg", dst);
img = cv2.imread('VisaoComputacional_Imagem_RottWeiler-thresh-binary-inv.jpg')
cv2.imshow('Imagem_RottWeiler-Threshold-Example',img)
cv2.waitKey()
```

OpenCV Thresholding

```
# Thresholding using THRESH_TRUNC
th, dst = cv2.threshold(src,127,255, cv2.THRESH_TRUNC);
print('THRESH_TRUNC')
cv2.imwrite("VisaoComputacional_Imagem_RottWeiler-thresh-trunc.jpg", dst);
img = cv2.imread('VisaoComputacional_Imagem_RottWeiler-thresh-trunc.jpg')
cv2.imshow('Imagem_RottWeiler-Threshold-Example',img)
cv2.waitKey()

# Thresholding using THRESH_TOZERO
th, dst = cv2.threshold(src,127,255, cv2.THRESH_TOZERO);
print('THRESH_TOZERO')
cv2.imwrite("VisaoComputacional_Imagem_RottWeiler-thresh-tozero.jpg", dst);
img = cv2.imread('VisaoComputacional_Imagem_RottWeiler-thresh-tozero.jpg')
cv2.imshow('Imagem_RottWeiler-Threshold-Example',img)
cv2.waitKey()

# Thresholding using THRESH_TOZERO_INV
th, dst = cv2.threshold(src,127,255, cv2.THRESH_TOZERO_INV);
print('THRESH_TOZERO_INV');
cv2.imwrite("VisaoComputacional_Imagem_RottWeiler-thresh-to-zero-inv.jpg", dst);
img = cv2.imread('VisaoComputacional_Imagem_RottWeiler-thresh-to-zero-inv.jpg')
cv2.imshow('Imagem_RottWeiler-Threshold-Example',img)
cv2.waitKey()

cv2.destroyAllWindows()
```

Executar:VisaoComputacional_OpenCV_Exemplo9.py

OpenCV Thresholding

- Exemplo:

```
import cv2

# Read image
src = cv2.imread("Rosto_de_Menina_001.png", cv2.IMREAD_GRAYSCALE);

assert src is not None, "Arquivo não encontrado!"

# Original image
cv2.imshow("ORIGINAL",src);
cv2.waitKey()

# Basic threhold example
th, dst = cv2.threshold(src, 100, 255, cv2.THRESH_BINARY);
cv2.imshow("THRESH_BINARY",dst);
cv2.waitKey()

# Thresholding with maxValue set to 128
th, dst = cv2.threshold(src,00,128, cv2.THRESH_BINARY);
cv2.imshow("THRESH_BINARY",dst);
cv2.waitKey()

# Thresholding with threshold value set 127
# Pixel values below 127 would be changed to Black
# Pixel values above 127 would be changed to White (255)
th, dst = cv2.threshold(src,127,255,cv2.THRESH_BINARY);
cv2.imshow("THRESH_BINARY",dst);
cv2.waitKey()
```

OpenCV Thresholding

```
# Thresholding using THRESH_BINARY_INV
th, dst = cv2.threshold(src,127,255, cv2.THRESH_BINARY_INV);
cv2.imshow("THRESH_BINARY_INV",dst);
cv2.waitKey()

# Thresholding using THRESH_TRUNC
th, dst = cv2.threshold(src,127,255, cv2.THRESH_TRUNC);
cv2.imshow("THRESH_TRUNC",dst);
cv2.waitKey()

# Thresholding using THRESH_TOZERO
th, dst = cv2.threshold(src,127,255, cv2.THRESH_TOZERO);
cv2.imshow("THRESH_TOZERO",dst);
cv2.waitKey()

# Thresholding using THRESH_TOZERO_INV
th, dst = cv2.threshold(src,127,255, cv2.THRESH_TOZERO_INV);
cv2.imshow("THRESH_TOZERO_INV",dst);
cv2.waitKey()
cv2.destroyAllWindows()
```

Executar:

VisaoComputacional_OpenCV_Exemplo95.py

OpenCV Thresholding

- Exercício 4:

Executar o thresholding para as figuras da refeição no Chile, Tigre e das Bebidas.

Atenção com o formato das imagens!

OpenCV Filtros

- Os filtros são encarregados de tarefas como remover ruído, aumentar contraste ou ressaltar característica da imagem como cantos bordas e agrupamentos;
- Ao aplicar um filtro passa-baixa, pode-se remover qualquer ruído da imagem;
- Ao aplicar um filtro passa-alta, ajuda na detecção das bordas;
- Para filtrar existe a função filter2D(), que possibilita realizar uma convolução, denominada convolução kernel, da imagem original com uma matriz MxN, onde M e N são inteiros ímpares, por exemplo 3×3, 5×5, 7×7 etc;
- A convolução kernel é usada para realizar operações matemáticas em cada pixel de uma imagem para obter o efeito desejado (como desfocar ou aumentar a nitidez de uma imagem);
- Desfoca-se para reduzir certos tipos de ruído em uma imagem. Por esse motivo, o desfoque costuma ser chamado de suavização;
- Para remover um fundo que não interessa, pode-se desfocar intencionalmente partes de uma imagem, como é feito no modo “Retrato”, em câmeras de dispositivos móveis.

OpenCV Filtros

- Pode ser usado o método blur() para suavizar a imagem, em vez do filter2D, bastando informar a dimensão da matriz kernel;

- Maiores detalhes do funcionamento em https://setosa.io/ev/image-kernels/;

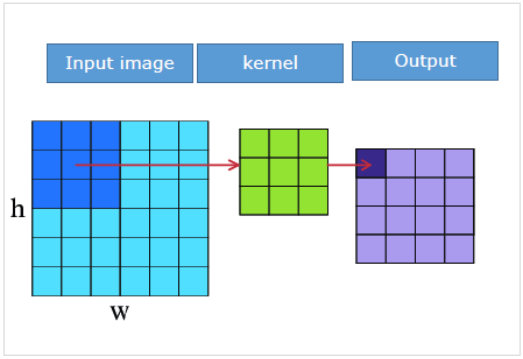
- Técnicas de filtragem adicionais:

  BilateralFilter: Reduz o ruído indesejado mantendo as bordas intactas.

  BoxFilter: É uma operação de desfoque médio.

  GaussianBlur: Elimina o conteúdo de alta frequência, como ruído e bordas.

  MedianBlur: Em vez da média, ele pega a mediana de t
  kernel e substitui o valor central.



OpenCV Filtros

Exemplo:

```
import cv2
import numpy as np
img = cv2.imread('Varios_Madeira_001.jpg')

cv2.imshow('Original', img)
cv2.waitKey(0)

kernel2 = np.ones((5, 5), np.float32) / 25
img2D = cv2.filter2D(src=img, ddepth=-1, kernel=kernel2)
cv2.imshow('filter2D', img2D)
cv2.waitKey(0)

img_blur = cv2.blur(src=img, ksize=(5,5))
cv2.imshow('Blurred', img_blur)
cv2.waitKey(0)

gaussian_blur = cv2.GaussianBlur(src=img,ksize=(5,5),sigmaX=0, sigmaY=0)
cv2.imshow('Gaussian Blurred', gaussian_blur)
cv2.waitKey(0)

median = cv2.medianBlur(src=img, ksize=5)
cv2.imshow('Median Blurred', median)
cv2.waitKey(0)

bilateral_filter = cv2.bilateralFilter(src=img, d=9, sigmaColor=75, sigmaSpace=75)
cv2.imshow('Bilateral Filtering', bilateral_filter)
cv2.waitKey(0)
```

OpenCV
Filtros

- Exercício 5:

Aplique os filtros para as imagens da Refeição, da Menina, do Tigre e do Rottweiler;

OpenCV
Detecção de bordas

- A detecção de bordas é uma técnica do processamento de imagens usada para identificar os limites (bordas) de objetos ou regiões dentro de uma imagem;
- As bordas estão entre os recursos mais importantes associados às imagens;
- A estrutura subjacente de uma imagem é conhecida através de suas bordas;
- Os pipelines de processamento da visão computacional, portanto, usam extensivamente a detecção de bordas em aplicativos;
- **Mudanças repentinas na intensidade dos pixels caracterizam as bordas;**
- **Procura-se essas mudanças nos pixels vizinhos para detectar as bordas;**
- Dois importantes algoritmos de detecção de bordas existem no OpenCV:Sobel Edge Detection e Canny Edge Detection;

OpenCV
Detecção de bordas

- Antes de aplicar as técnicas de detecção de bordas deve-se suavizá-la, reduzindo-se, desta forma, a quantidade de conteúdo de alta frequência, como ruídos;

OpenCV
Detecção de bordas

- Exemplo:

```
import numpy as np
import cv2 as cv

img = cv.imread('VisaoComputacional_Imagem_RottWeiler.jpg')

img_blur = cv.GaussianBlur(img,(3,3),0) # filtro
edges = cv.Canny(img_blur,100,200)

cv.imshow('Original image', img)
cv.waitKey()
cv.imshow('Canny image', edges)

cv.waitKey()
cv.destroyAllWindows()
```

Executar:
VisaoComputacional_OpenCV_Exemplo10.py (Algoritmo Canny)

OpenCV
Detecção de bordas

Exemplo:

```
import cv2

# Read the original image
img = cv2.imread('VisaoComputacional_Imagem_RottWeiler.jpg')
# Display original image
cv2.imshow('Original', img)
cv2.waitKey()

# Convert to graycsale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Blur the image for better edge detection
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)

cv2.waitKey()
# Sobel Edge Detection
sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5) # Sobel Edge #Detection on the X axis
sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5) # Sobel Edge #Detection on the Y axis
sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5) # Combined X and Y #Sobel Edge Detection
# Display Sobel Edge Detection Images
cv2.imshow('Sobel X', sobelx)
cv2.waitKey()
cv2.imshow('Sobel Y', sobely)
cv2.waitKey()
cv2.imshow('Sobel X Y using Sobel() function', sobelxy)
cv2.waitKey()
```

OpenCV
Detecção de bordas

- Exemplo:

```
# Canny Edge Detection
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) # Canny Edge Detection
# Display Canny Edge Detection Image
cv2.imshow('Canny Edge Detection', edges)
cv2.waitKey()

cv2.destroyAllWindows()
```

Executar:

VisaoComputacional_OpenCV_Exemplo101.py (Algoritmo Sobel)

OpenCV
Detecção de bordas

- Exercício 6:

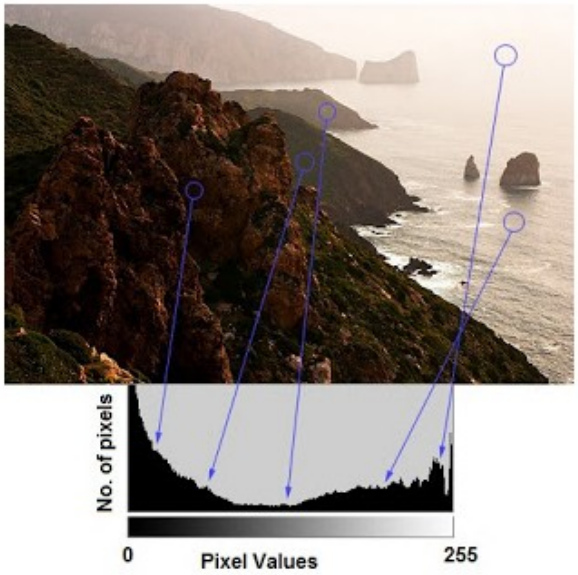
Executar o Algoritmos de Canny e Sobel para as figuras da refeição no Chile, Tigre, Menina, James Brown e das Bebidas.

OpenCV
Instalação de Biblioteca

- Emitir o comandos, no prompt do DOS:
  - pip install dlib==19.24.2
  - pip install face_recognition

OpenCV Histograma

- O histograma é um gráfico que dá uma ideia geral sobre a distribuição da intensidade tonal de uma imagem;
- Apresenta valores de pixels(variando de 0 a 255) no eixo X e a quantidade correspondente de pixels da imagem no eixo Y;
- Usando o histograma, pode-se entender a distribuição de contraste, brilho e intensidade de uma imagem específica;
- A região esquerda do histograma mostra a quantidade de pixels mais escuros na imagem e a região direita mostra a quantidade de pixels mais brilhantes;
- No histograma, pode-se ver que a região escura é maior do que a região mais clara, e a quantidade de tons médios é muito menor.



OpenCV Histograma

- Um histograma pode não contar apenas as intensidades das cores, mas também quaisquer características da imagem que deseja-se medir como gradientes, direções etc.

- A função calcHist() calcula o histograma de uma imagem;
- Sintaxe:
  - calcHist(imagem, canais, máscara, faixa)
    - imagem: É a imagem alvo.
    - canais: É o índice do canal para o cálculo do histograma. Para imagem em tons de cinza, seu valor é [0]. Para imagens BGR, é [0], [1] ou [2].
    - máscara: A imagem da máscara é fornecida como "None" para imagem completa. Para uma determinada região de imagem, cria-se uma imagem de máscara.
    - faixa: normalmente é [0,256].

OpenCV Histograma

- Para visualizar o Histograma, sob a forma de *plot*, deve ser instalada a biblioteca matplotlib. Para tal, emitir o seguinte comando no prompt do DOS:

  - pip install -U matplotlib

OpenCV Histograma

- Exemplo:

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread('VisaoComputacional_Imagem_RottWeiler.jpg')

cv.imshow('Histograma 1', img)
cv.waitKey(0)

color = ('b','g','r')
for i,col in enumerate(color):
    hist = cv.calcHist([img],[i],None,[256],[0,256])
    plt.plot(hist, color = col)
    plt.xlim([0,256])
plt.show()
cv.waitKey(0)
img = cv.imread('VisaoComputacional_Imagem_Refeicao_Chile.jpg')
cv.imshow('Histograma 2', img)
cv.waitKey(0)

for i,col in enumerate(color):
    hist = cv.calcHist([img],[i],None,[256],[0,256])
    plt.plot(hist, color = col)
    plt.xlim([0,256])
plt.show()
cv.waitKey()
cv.destroyAllWindows()
```

Executar: VisaoComputacional_OpenCV_Exemplo11.py

OpenCV
Transformações Morfológicas

- Transformações morfológicas são um conjunto de operações que processam imagens com base em suas formas;
- Elas aplicam um elemento estruturante a uma imagem de entrada e geram uma imagem de saída;
- As operações morfológicas mais básicas são duas:
  - Erosão
  - Dilatação

OpenCV
Transformações Morfológicas

- Erosão:
  - Elimina os limites do objeto em primeiro plano.
  - Feita uma convolução kernel com a imagem.
  - Um pixel na imagem original (1 ou 0) será considerado 1 somente se todos os pixels sob o kernel forem 1, caso contrário, ele será erodido (tornado zero).
  - Assim, todos os pixels próximos ao limite serão descartados dependendo do tamanho do kernel.
  - Portanto, a espessura ou o tamanho do objeto em primeiro plano diminui.
- Sintaxe:
  - cv.erode(src, kernel, dst, anchor, iterations)
    - src e dst são matrizes das imagens de entrada e saída do mesmo tamanho.
    - kernel é uma matriz de elementos estruturantes usados para erosão.
    - anchor é -1 por padrão, o que significa que o elemento âncora está no centro.
    - iterations referem-se ao número de vezes que a erosão é aplicada.

OpenCV
Transformações Morfológicas

- Dilatação:
  - É exatamente o oposto de erosão pois aumenta a área do objeto.
  - Usado para acentuar características.
  - Feita uma convolução kernel com a imagem.
  - Um elemento de pixel na imagem original é '1' se pelo menos um pixel sob o kernel for '1'.
  - Ele aumenta o tamanho do objeto em primeiro plano.
- Sintaxe:
  - cv.dilate(src, kernel, dst, anchor, iterations)
    - src e dst são matrizes das imagens de entrada e saída do mesmo tamanho.
    - kernel é uma matriz de elementos estruturantes usados para erosão.
    - anchor é -1 por padrão, o que significa que o elemento âncora está no centro.
    - iterations referem-se ao número de vezes que a erosão é aplicada.

OpenCV
Transformações Morfológicas

Exemplo:

```
import cv2 as cv
import numpy as np

kernel = np.ones((5,5),np.uint8)
img = cv.imread('Imagem_Linux_001.jpg',0)

erosion = cv.erode(img,kernel,iterations = 1)
dilation = cv.dilate(img,kernel,iterations = 1)

cv.imshow('Original', img)
cv.waitKey(0)
cv.imshow('Erosion', erosion)
cv.waitKey(0)
cv.imshow('Dilation', dilation)
cv.waitKey(0)

img = cv.imread('Varios_Dados_001.jpg',1)

erosion = cv.erode(img,kernel,iterations = 1)
dilation = cv.dilate(img,kernel,iterations = 1)

cv.imshow('Original', img)
cv.waitKey(0)
cv.imshow('Erosion', erosion)
cv.waitKey(0)
cv.imshow('Dilation', dilation)
cv.waitKey(0)

cv.destroyAllWindows()
```

OpenCV
Espaços de Cor

- Um espaço de cores é um modelo matemático que descreve como as cores podem ser representadas;
- Descrito em uma faixa específica, mensurável e fixa de cores possíveis e valores de luminância;
- O OpenCV suporta os seguintes espaços de cores:
  - RGB: Um valor de cor é obtido pela combinação de valores das cores vermelha, verde e azul. Cada uma é representado por um número que varia entre 0 e 255
  - Espaço de cores HSV: H, S e V representam Matiz, Saturação e Valor. O valor de Matiz está entre 0 e 179, enquanto os números S e V estão entre 0 e 255.

OpenCV
Espaços de Cor

- CMYK: Representam Ciano, Magenta, Amarelo e Preto. Todos os valores são representados na escala de 0 a 100%.

- LAB: Tem três componentes que são L para luminosidade, A que variam de verde a magenta e B para componentes de azul a amarelo.

- YCrCb: Utilizado em sistemas de vídeo digital e compressão de imagem. Baseado no modelo de cores RGB (Red, Green, Blue) e comumente utilizado em formatos de vídeo como o MPEG e o JPEG. Luminância (Y), a crominância vermelha (Cr) e a crominância azul (Cb).

OpenCV
Espaços de Cor

- cvtColor()
  - método usado para converter uma imagem de um espaço de cor para outro. Existem mais de 150 conversões de espaço de cores.
- Sintaxe:
  - cvtColor(src, code[, dst])
  - Onde:

    src: é a imagem cujo espaço de cor deve ser alterado

    code: é o código de conversão do espaço de cor

    dst: é a imagem de saída do mesmo tamanho que a imagem src

  - Códigos em:
    https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html

OpenCV
Espaços de Cor

- Exemplo:

```
import cv2

img = cv2.imread('VisaoComputacional_Imagem_Refeicao_Chile.jpg')

img1 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY )
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
# Displaying the image
cv2.imshow('original', img)
cv2.waitKey(0)
cv2.imshow('Gray', img1)
cv2.waitKey(0)
cv2.imshow('HSV', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar: VisaoComputacional_OpenCV_Exemplo12.py

OpenCV
Contornos e contadores

- Contorno é uma curva que une todos os pontos contínuos ao longo do limite de uma imagem;
- Os contornos são úteis para a análise da forma e detecção de objetos humanos ou não humanos;
- Antes de encontrar contornos, deve-se aplicar a detecção de limite(threshold) ou borda canny;
- Para melhor precisão, utilizam-se imagens binárias;
- **No OpenCV, encontrar contornos é como encontrar um objeto branco em um fundo preto;**

OpenCV
Contornos e contadores

- Sintaxe:
  - findContours(image, mode, method, contours)
  - Onde:

    image: A imagem de entrada binária obtida em etapa anterior

    mode: É o modo de recuperação de contorno.

    - **RETR_EXTERNAL**: Recupera apenas os contornos externos extremos.
    - **RETR_LIST**: Recupera todos os contornos sem estabelecer quaisquer relações hierárquicas.
    - **RETR_CCOMP**: Recupera todos os contornos e os organiza em uma hierarquia de dois níveis.
    - **RETR_TREE:** Recupera todos os contornos e reconstrói uma hierarquia completa de contornos aninhados.

OpenCV
Contornos e contadores

- Sintaxe(continuação):

  method: Define o método de aproximação de contorno.

  - **CHAIN_APPROX_NONE**: Armazena absolutamente todos os pontos de contorno.
  - **CHAIN_APPROX_SIMPLE:** Comprime segmentos horizontais, verticais e diagonais e deixa apenas seus pontos finais.

- Parâmetros de retorno:

  img - uma imagem de canal único de 8 bits. Pixels diferentes de zero são tratados como 1's. Pixels zero permanecem 0's, então a imagem é tratada como binary.

  contours - contornos detectados. **Cada contorno é armazenado como um vetor de pontos**.

  hierarchy - vetor de saída opcional, contendo informações sobre a topologia da imagem. Ele tem tantos elementos quanto o número de contornos.

OpenCV
Contornos e contadores

- O que é hierarquia?
  - Os contornos em uma imagem podem ser organizados de acordo com os relacionamentos que eles compartilham entre si;
  - Pode haver contornos que ficam um dentro do outro;
  - Então o contorno externo pode ser chamado de pai e o contorno interno de filho;
  - Esse relacionamento agora pode ser usado para organizar os contornos de acordo com a hierarquia;

OpenCV
Contornos e contadores

- Após detectar os vetores de contorno, eles são desenhados sobre a imagem original usando o método drawContours();
- Sintaxe:
  - drawContours(image, contours, contourIdx, color,thickness)
  - Onde:

    image: Imagem de destino.

    contours: Todos os contornos de entrada. Cada contorno é armazenado como um vetor de pontos.

    contourIdx: Parâmetro que indica um contorno a ser desenhado. Se for negativo, todos os contornos são desenhados.

    color: Cor dos contornos.

    thickness: Espessura da linha de desenho do contorno.

OpenCV
Contornos e contadores

- Exemplo:

```
import cv2
import numpy as np

image = cv2.imread('Varios_Ovos_001.jpg')
cv2.imshow('Original', image)
cv2.waitKey(0)
kernel = np.ones((5,5),np.uint8)
img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Cinza', img)
cv2.waitKey(0)

#img = cv2.erode(img,kernel,iterations = 1)
#cv2.imshow('Erosao', img)
#cv2.waitKey(0)

img = cv2.Canny(img, 30, 200)
cv2.imshow('Canny', img)
cv2.waitKey(0)

contours, hierarchy = cv2.findContours(img,cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
print("Numero of Contornos = " ,len(contours))

cv2.drawContours(image, contours, -1, (0, 255, 0), 1)
cv2.imshow('Contours', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar: VisaoComputacional_OpenCV_Exemplo13.py
Executar novamente retirando os comentários.

OpenCV
Contornos e contadores

- Exemplo:

```
import cv2

image2 = cv2.imread('Varios_Dados_001.jpg')
assert image2 is not None, "file could not be read, check with os.path.exists()"
cv2.imshow('Original', image2)
cv2.waitKey(0)

img_gray2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
cv2.imshow('Cinza', img_gray2)
cv2.waitKey(0)

suave = cv2.blur(img_gray2, (7, 7))  # Aplicam-se os filtros
cv2.imshow('Filtrado', suave)
cv2.waitKey(0)

ret, thresh2 = cv2.threshold(suave, 100, 255, cv2.THRESH_BINARY) # Transforma para binária
cv2.imshow('Imagem Binaria', thresh2)
cv2.waitKey(0)

contours3, hierarchy3 = cv2.findContours(thresh2, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)

image_copy4 = image2.copy()
cv2.drawContours(image_copy4, contours3, -1, (0, 255, 0), 2, cv2.LINE_AA)
print("Numero of Contornos = " ,len(contours3))
cv2.imshow('Contours', image_copy4)

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

Executar VisaoComputacional_OpenCV_Exemplo131.py

OpenCV
Contornos e contadores

- Exemplo:

```
import cv2
image2 = cv2.imread('Rosto_de_Menina_001.png')
#image2 = cv2.imread("James_Brown_001.png");

cv2.imshow('Original', image2)
cv2.waitKey(0)

img_gray2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
cv2.imshow('Cinza', img_gray2)
cv2.waitKey(0)

ret, thresh2 = cv2.threshold(img_gray2, 100, 255, cv2.THRESH_BINARY) # Transforma para binária
cv2.imshow('Threshold', thresh2)
cv2.waitKey(0)

contours4, hierarchy4 = cv2.findContours(thresh2, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
image_copy5 = image2.copy()
cv2.drawContours(image_copy5, contours4, -1, (0, 255, 0), 2, cv2.LINE_AA)

cv2.imshow('EXTERNAL', image_copy5)
cv2.waitKey(0)
print(len(contours4))
cv2.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo132.py
Executar com a imagem do James Brown

OpenCV
Contornos e contadores

- Exemplo:

```
import cv2

#image2 = cv2.imread('VisaoComputacional_Imagem_RottWeiler.jpg')
#image2 = cv2.imread("James_Brown_001.png");
image2 = cv2.imread('Rosto_de_Menina_001.png')

cv2.imshow('Original', image2)
cv2.waitKey(0)

gray = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
cv2.imshow('Cinza', gray)
cv2.waitKey(0)

blur = cv2.bilateralFilter(src=gray, d=9, sigmaColor=75, sigmaSpace=75)  # Aplicam-se os filtros
cv2.imshow('Filtrado', blur)
cv2.waitKey(0)

ret, thresh2 = cv2.threshold(blur, 100, 255, cv2.THRESH_BINARY) # Transforma para binária
cv2.imshow('Imagem Binaria', thresh2)
cv2.waitKey(0)

canny = cv2.Canny(thresh2, 70, 150)  # Detecta as bordas
cv2.imshow('Bordas Detectadas', canny)
cv2.waitKey(0)
```

OpenCV
Contornos e contadores

- Exemplo(continuação):

```
contours5, hierarchy5 = cv2.findContours(thresh2, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_NONE)
image_copy6 = image2.copy()
cv2.drawContours(image_copy6, contours5, -1, (0, 255, 0), 2, cv2.LINE_AA)
print("Numero of Contornos = " ,len(contours5))
# see the results
cv2.imshow('CCOMP', image_copy6)

cv2.waitKey(0)
print(len(contours5))
cv2.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo133.py

Executar com as outras imagens.

OpenCV
Contornos e contadores

- Exemplo:

```
import cv2
import numpy as np

image = cv2.imread('Varios_Dados_001.jpg')
#image = cv2.imread('Varios_MCUS_001.jpg')

assert image is not None, "file could not be read, check with os.path.exists()"
cv2.imshow('Original', image)
cv2.waitKey(0)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray', gray)
cv2.waitKey(0)

#blur = cv2.GaussianBlur(gray, (11, 11), 0)
blur = cv2.bilateralFilter(src=gray, d=9, sigmaColor=75, sigmaSpace=75)  # Aplicam-se os filtros
cv2.imshow('Filtrado', blur)
cv2.waitKey(0)

ret, thresh2 = cv2.threshold(blur, 100, 255, cv2.THRESH_BINARY) # Transforma para binária
cv2.imshow('Imagem binaria', thresh2)
cv2.waitKey(0)
```

OpenCV
Contornos e contadores

- Exemplo(continuação):

```
canny = cv2.Canny(thresh2, 70, 150)  # Detecta as bordas
cv2.imshow('Bordas Detectadas', canny)
cv2.waitKey(0)

(cnt, hierarchy) = cv2.findContours(canny.copy(), cv2.RETR_EXTERNAL,  cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(image, cnt, -1, (0, 255, 0), 2)
print("Objetos na imagem : ", len(cnt))
cv2.imshow('Final', image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo134.py

OpenCV
Contornos e contadores

Exemplo Fatal:

```
iimport cv2
import numpy as np

image = cv2.imread('Varios_Dados_001.jpg')
#image = cv2.imread('Varios_MCUS_001.jpg')

assert image is not None, "file could not be read, check with os.path.exists()"
cv2.imshow('Original', image)
cv2.waitKey(0)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray', gray)
cv2.waitKey(0)

#blur = cv2.GaussianBlur(gray, (11, 11), 0)
blur = cv2.bilateralFilter(src=gray, d=9, sigmaColor=75, sigmaSpace=75) # Aplicam-se os filtros
cv2.imshow('Filtrado', blur)
cv2.waitKey(0)

ret,thresh2 = cv2.threshold(blur, 100, 255, cv2.THRESH_BINARY) # Transforma para binária
cv2.imshow('Imagem binaria', thresh2)
cv2.waitKey(0)
```

OpenCV
Contornos e contadores

- Exemplo Fatal(continuação):

```
#canny = cv2.Canny(blur, 30, 150, 3)
canny = cv2.Canny(thresh2, 70, 150)# Detecta as bordas
cv2.imshow('Bordas Detectadas', canny)
cv2.waitKey(0)


kernel = np.ones((7, 7), np.uint8)
dilated = cv2.dilate(canny, kernel, iterations=2) # Dilata as bordas
cv2.imshow('Bordas dilatadas', dilated)
cv2.waitKey(0)


(cnt, hierarchy) = cv2.findContours(dilated.copy(), cv2.RETR_EXTERNAL,  cv2.CHAIN_APPROX_SIMPLE)

cv2.drawContours(image, cnt, -1, (0, 255, 0), 2)


print("Objetos na imagem : ", len(cnt))
cv2.imshow('rgb', image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo136.py

OpenCV
Área e perímetro

- Cálculo de área e perímetro de um objeto são possíveis pelos métodos contourArea() e arcLength();

- Sintaxe:
  - Área:
    - contourArea(cnt)
      - Onde:
        - » cnt: é uma matriz numpy dos pontos de contorno de um objeto na imagem.
  - Perímetro:
    - arcLength(cnt,True)
      - Onde:
        - » cnt: é uma matriz numpy dos pontos de contorno de um objeto na imagem.
        - » True: flag que indica se a curva está fechada ou não.

OpenCV
Área e perímetro

- Exemplo:

```
import cv2
import numpy as np

# Load the image
image = cv2.imread('Varios_MCUS_001.jpg')
assert image is not None, "file could not be read, check with os.path.exists()"
cv2.imshow('Original', image)
cv2.waitKey(0)

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray', gray)
cv2.waitKey(0)

blur = cv2.bilateralFilter(src=gray, d=9, sigmaColor=75, sigmaSpace=75)
ret, thresh2 = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
cv2.imshow('threshold', thresh2)
cv2.waitKey(0)

# Find the contours of the objects in the image
```

OpenCV
Área e perímetro

- Exemplo:

```
# Loop through the contours and calculate the area of each object
for cnt in contours:
    area = cv2.contourArea(cnt)

# Draw a bounding box around each object and display the area on the image
    x, y, w, h = cv2.boundingRect(cnt)
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
    cv2.putText(image, str(area),(x, y),cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 1)

# Show the final image with the bounding boxes  and areas of the objects overlaid on top
cv2.imshow('image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo137.py

OpenCV
Área e perímetro

- Exemplo:

```
import cv2
img = cv2.imread('Imagem_Triangulo_001.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(gray,150,255,0)

contours,hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
print("Numero de contornos:",len(contours))
cnt = contours[0]

# compute the area and perimeter
area = cv2.contourArea(cnt)
perimeter = cv2.arcLength(cnt, True)
perimeter = round(perimeter, 4)
print('Area:', area)
print('Perimetro:', perimeter)
img1 = cv2.drawContours(img, [cnt], -1, (0,255,255), 3)
x1, y1 = cnt[0,0]
cv2.putText(img1, f'Area:{area}', (x1, y1+50), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
cv2.putText(img1, f'Perimetro:{perimeter}', (x1, y1+20), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)

cv2.imshow("Image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo16.py

OpenCV
Área e perímetro

Exemplo:

```
import cv2
import numpy as np

img1 = cv2.imread('Imagens_Figuras_Geometricas_001.jpg')
assert img1 is not None, "file could not be read, check with os.path.exists()"
img = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray', img)
cv2.waitKey(0)

ret,thresh = cv2.threshold(img,10,255,0)

contours,hierarchy = cv2.findContours(thresh, 1, 2)
print("Numero de figuras na imagem:",len(contours))

for i, cnt in enumerate(contours):
   M = cv2.moments(cnt)
   if M['m00'] != 0.0:
      x1 = int(M['m10']/M['m00'])
      y1 = int(M['m01']/M['m00'])
   area = cv2.contourArea(cnt)
   perimeter = cv2.arcLength(cnt, True)
   perimeter = round(perimeter, 4)
   print(f'Area da figura {i+1}:', area)
   print(f'Perimetro da figura {i+1}:', perimeter)
   img1 = cv2.drawContours(img1, [cnt], -1, (0,255,255), 3)
   cv2.putText(img1, f'Area :{area}', (x1, y1), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)
```

OpenCV
Área e perímetro

- Exemplo(continuação):

```
cv2.imshow("Image", img1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo161.py

OpenCV
Retângulo delimitador

- Existem dois tipos de retângulos delimitadores:
  - Retângulo delimitador reto
    - É um retângulo reto que não considera a rotação do objeto. Ela é encontrada pela função boundingRect().
  - Sintaxe: x,y,w,h = boundingRect(cnt)
  - Onde:
    - cnt – matriz dos contornos
    - x,y são coordenada superior esquerda do retângulo e (w,h) sua largura e altura.

OpenCV
Retângulo delimitador

- Existem dois tipos de retângulos delimitadores(continuação):
  - Retângulo rotacionado
    - Aqui, o retângulo delimitador é desenhado com área mínima, portanto, considera a rotação também.
    - Obtido pela funções minAreaRect() e boxPoints().
    - Sintaxe: center(x,y),w,h,a = minAreaRect(cnt)
    - Onde :
      - cnt – matriz dos contornos.
      - x,y indicam a coordenada do ponto central e (w,h) sua largura e altura.
      - a é o ângulo de rotação.

OpenCV
Retângulo delimitador

- Existem dois tipos de retângulos delimitadores(continuação):
  - Retângulo rotacionado - boxPoints()
    - Antes de desenhar o retângulo, necessário converter os 4 cantos para o tipo inteiro.
    - Sintaxe: x,y,w,h = boxPoints(rect)
    - Onde :
      - rect – retorno da função minAreaRect().
      - x,y,w,h – pontos para desenhar o retângulo.

OpenCV
Retângulo delimitador

- Exemplo:

```
import cv2
import numpy as np
import math
img = cv2.imread('Imagem_Raio_001.jpg',1)
img1 = cv2.imread('Imagem_Raio_001.jpg',cv2.IMREAD_GRAYSCALE)
assert img1 is not None, "file could not be read, check with os.path.exists()"


ret,thresh = cv2.threshold(img1,127,255,0)
contours,hierarchy = cv2.findContours(thresh, 1, 2)

cnt = contours[0]
x,y,w,h = cv2.boundingRect(cnt)
img2= cv2.rectangle(img.copy(),(x,y),(x+w,y+h),(0,255,0),2)
perimetro = cv2.arcLength(cnt,True)
print(math.trunc(perimetro))

cv2.imshow("Image", img2)
cv2.waitKey(0)

rect = cv2.minAreaRect(cnt)
box = cv2.boxPoints(rect)
box = np.intp(box)
cv2.drawContours(img,[box],0,(0,0,255),2)

perimetro = cv2.arcLength(box,True)
print(math.trunc(perimetro))

cv2.imshow("Image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OpenCV
Contornos, área e perímetro

- Exercício 7:

Contem e apresentem as dimensões dos objetos apresentados.

OpenCV
Contornos

- Exercícios Soluções (baseado no VisaoComputacional_OpenCV_Exemplo136):

```
import cv2
import numpy as np

image = cv2.imread('Varias_Porcas_001.jpg')
#image = cv2.imread('Varios_Parafusos_001.jpg')

assert image is not None, "file could not be read, check with os.path.exists()"
cv2.imshow('Original', image)
cv2.waitKey(0)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray', gray)
cv2.waitKey(0)

blur = cv2.bilateralFilter(src=gray, d=9, sigmaColor=75, sigmaSpace=75) # Aplicam-se os filtros
cv2.imshow('Filtrado', blur)
cv2.waitKey(0)

ret,thresh2 = cv2.threshold(blur, 100, 255, cv2.THRESH_BINARY) # Transforma para binária
#ret,thresh2 = cv2.threshold(blur, 100, 255, cv2.THRESH_TOZERO) # Transforma para binária
cv2.imshow('Imagem binaria', thresh2)
cv2.waitKey(0)

canny = cv2.Canny(thresh2, 70, 150)# Detecta as bordas
cv2.imshow('Bordas Detectadas', canny)
cv2.waitKey(0)
```

OpenCV
Contornos

- Exercícios Soluções(baseado no VisaoComputacional_OpenCV_Exemplo136):

```
kernel = np.ones((7, 7), np.uint8)
dilated = cv2.dilate(canny, kernel, iterations=2) # Dilata as bordas
cv2.imshow('Bordas dilatadas', dilated)
cv2.waitKey(0)

(cnt, hierarchy) = cv2.findContours(dilated.copy(), cv2.RETR_EXTERNAL,  cv2.CHAIN_APPROX_SIMPLE)

cv2.drawContours(image, cnt, -1, (0, 255, 0), 2)

print("Objetos na imagem : ", len(cnt))
cv2.imshow('rgb', image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

OpenCV
Contornos, área e perímetro

- Exercícios Soluções (baseado no VisaoComputacional_OpenCV_Exemplo138):

```
import cv2
import numpy as np
import math

# Load the image
#image = cv2.imread('Varios_MCUS_001.jpg')
image = cv2.imread('Varios_Parafusos_Diferentes_001.jpg')

assert image is not None, "file could not be read, check with os.path.exists()"
cv2.imshow('Original', image)
cv2.waitKey(0)

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray', gray)
cv2.waitKey(0)

blur = cv2.bilateralFilter(src=gray, d=9, sigmaColor=75, sigmaSpace=75)
ret, thresh2 = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
cv2.imshow('threshold', thresh2)
cv2.waitKey(0)

# Find the contours of the objects in the image
contours, hierarchy = cv2.findContours(thresh2, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
print("Objetos na imagem : ", len(contours))
```

OpenCV
Contornos, área e perímetro

- Exercícios Soluções(baseado no VisaoComputacional_OpenCV_Exemplo138):

```
# Loop through the contours and calculate the area of each object
for cnt in contours:
    perimetro = cv2.arcLength(cnt,True)
    print(math.trunc(perimetro))

# Draw a bounding box around each object and display the perimeter on the image
    x, y, w, h = cv2.boundingRect(cnt)
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
    cv2.putText(image, str(math.trunc(perimetro)),(x, y),cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 1)

# Show the final image with the bounding boxes and perimeter of the objects overlaid on top
cv2.imshow('image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OpenCV
Retângulo delimitador e perímetro

- Exercícios Soluções (baseado no VisaoComputacional_OpenCV_Exemplo138_1):

```
import numpy as np
import math

# Load the image
image = cv2.imread('Varios_MCUS_001.jpg')
#image = cv2.imread('Varios_Parafusos_Diferentes_001.jpg')

assert image is not None, "file could not be read, check with os.path.exists()"
cv2.imshow('Original', image)
cv2.waitKey(0)

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray', gray)
cv2.waitKey(0)

blur = cv2.bilateralFilter(src=gray, d=9, sigmaColor=75, sigmaSpace=75)
ret, thresh2 = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
cv2.imshow('threshold', thresh2)
cv2.waitKey(0)

# Find the contours of the objects in the image
contours, hierarchy = cv2.findContours(thresh2, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
print("Objetos na imagem : ", len(contours))
```

OpenCV
Retângulo delimitador e perímetro

- Exercícios Soluções(baseado no VisaoComputacional_OpenCV_Exemplo138_1):

```
# Loop through the contours and calculate the perimeter of each object
for cnt in contours:
    rect = cv2.minAreaRect(cnt)
    box = cv2.boxPoints(rect)
    box = np.intp(box)
    cv2.drawContours(image,[box],0,(0,0,255),2)
    perimetro = cv2.arcLength(box,True)
    x, y, w, h = cv2.boundingRect(cnt)
    cv2.putText(image, str(math.trunc(perimetro)),(x, y),cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 1)


# Show the final image
cv2.imshow('image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OpenCV
Contornos, área e perímetro

- Exercícios Soluções (baseado no VisaoComputacional_OpenCV_Exemplo139):

```
import cv2
import numpy as np

#image = cv2.imread('Varios_Dados_001.jpg')
#image = cv2.imread('Varios_MCUS_001.jpg')
#image = cv2.imread('Varias_Porcas_001.jpg')
#image = cv2.imread('Varios_Parafusos_001.jpg')
image = cv2.imread('Varios_Lapis_001.jpg')

assert image is not None, "file could not be read, check with os.path.exists()"
cv2.imshow('Original', image)
cv2.waitKey(0)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray', gray)
cv2.waitKey(0)

blur = cv2.bilateralFilter(src=gray, d=9, sigmaColor=75, sigmaSpace=75) # Aplicam-se os filtros
cv2.imshow('Filtrado', blur)
cv2.waitKey(0)

ret,thresh2 = cv2.threshold(blur, 130, 255, cv2.THRESH_BINARY) # Transforma para binária

cv2.imshow('Imagem binaria', thresh2)
cv2.waitKey(0)
```

OpenCV
Contornos, área e perímetro

- Exercícios Soluções(baseado no VisaoComputacional_OpenCV_Exemplo139):

```
canny = cv2.Canny(thresh2, 70, 150) # Detecta as bordas
cv2.imshow('Bordas Detectadas', canny)
cv2.waitKey(0)


kernel = np.ones((7, 7), np.uint8)
dilated = cv2.dilate(canny, kernel, iterations=2) # Dilata as bordas
cv2.imshow('Bordas dilatadas', dilated)
cv2.waitKey(0)


(cnt, hierarchy) = cv2.findContours(dilated.copy(), cv2.RETR_EXTERNAL,  cv2.CHAIN_APPROX_SIMPLE)


cv2.drawContours(image, cnt, -1, (0, 255, 0), 2)


print("Objetos na imagem : ", len(cnt))
cv2.imshow('rgb', image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

OpenCV
Template Matching

- Técnica de *template matching* é usada para detectar uma ou mais áreas em uma imagem que corresponde a uma amostra ou modelo;
- Implementada pelo método matchTemplate();
- Sintaxe:
  - matchTemplate(image, templ, method)
  - Onde:
    - image - é a imagem de entrada na qual o padrão templ (template) deve ser localizado.
    - method deve ser um dos seguintes valores:
      - TM_CCOEFF
      - TM_CCOEFF_NORMED
      - TM_CCORR
      - TM_CCORR_NORMED
      - TM_SQDIFF
      - TM_SQDIFF_NORMED

OpenCV
Template Matching

- Se a imagem de entrada for do tamanho (LxA) e a imagem do *template* for do tamanho (lxa), a imagem de saída terá um tamanho de (L-l+1, A-a+1);

- Esse retângulo é a região de *template;*

- Apesar de pontos positivos, a correspondência de modelos falha se houver fatores de variação nas imagens de entrada, incluindo alterações na rotação, escala, ângulo de visão etc;

- Se as imagens de entrada contiverem essas variações utiliza-se detectores de objetos como HOG + Linear SVM, Faster R-CNN, YOLO, DLIB etc;

OpenCV
Template Matching

- Exemplo pesquisando o *template* em uma imagem simples :

```
import cv2
import numpy as np
img = cv2.imread('Grupo_Caxias_001.jpg',1)
cv2.imshow('Original',img)
cv2.waitKey(0)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

template = cv2.imread('Template_Caxias_001.jpg',1)
cv2.imshow('Template',template)
cv2.waitKey(0)
template = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)

w,h = template.shape[0], template.shape[1]
matched = cv2.matchTemplate(gray,template,cv2.TM_CCOEFF_NORMED)
threshold = 0.8
loc = np.where( matched >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img, pt, (pt[0] + h, pt[1] + w), (0,255,255), 1)
cv2.imshow('Matched with Template',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo_15

OpenCV
Template Matching

- Exercício :

Ajustar para procurar Wally e as cartas do baralho.

Alterem o threshold.

OpenCV
Template Matching

- Exemplo, pesquisando o *template* em imagens de um diretório :

```
import cv2
import numpy as np
import glob
import os

#path = "*.jpg"
path = "E:/UDTInovUERJ/Cursos_FAPERJ/Visão_Computacional/fontesImagens/*.JPG"
template = cv2.imread('Template_Caxias_001.jpg',0)

cv2.imshow('Template',template)
cv2.waitKey(0)

w,h = template.shape[0], template.shape[1]
print(w)
print(h)

for file in glob.glob(path):
```

OpenCV
Template Matching

- Exemplo(continuação):

```
    fileName_absolute = os.path.basename(file)
    print(fileName_absolute)
    img= cv2.imread(fileName_absolute)
    #print(img)
    cv2.imshow('Archive',img)
    w1,h1 = img.shape[0], img.shape[1]
    print(w1)
    print(h1)
    cv2.waitKey(0)
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    matched = cv2.matchTemplate(gray,template,cv2.TM_CCOEFF_NORMED)
    threshold = 0.8
    loc = np.where( matched >= threshold)

    for pt in zip(*loc[::-1]):
        cv2.rectangle(img, pt, (pt[0] + h, pt[1] + w), (0,255,255), 0)
        cv2.imshow('Matched with Template',img)

cv2.destroyAllWindows()
print("FIM")
```

Executar VisaoComputacional_OpenCV_Exemplo_153

OpenCV
Detecção de Objetos

- O OpenCV usa classificadores em cascata baseados em recursos Haar para a detecção de objetos;
- É um algoritmo baseado em aprendizado de máquina, onde uma função em cascata é treinada a partir de muitas imagens positivas e negativas;
- Ele é então usado para detectar objetos em outras imagens. O algoritmo usa o conceito de Cascata de Classificadores;
- Classificadores pré-treinados para rosto, olho etc podem ser baixados de https://github.com/opencv/opencv/tree/master/data/haarcascades;
- Baixar haarcascade_frontalface_default.xml e haarcascade_eye.xml desta URL;

OpenCV
Detecção de Objetos

- O método detectMultiScale() da classe CascadeClassifier detecta objetos na imagem de entrada;
- Retorna as posições dos rostos detectados na forma de Retângulo e suas dimensões (x, y, w, h);
- Depois da obtenção desses locais, esses locais, pode-se usá-los para detecção de olhos, já que os olhos estão sempre no rosto!

OpenCV
Detecção de Objetos

- Sintaxe:

detectMultiScale(image,slacefactor,minneihbors);

- Onde:
  - Image: A imagem a ser utilizada (em tons de cinza);
  - ScaleFactor:
    - Especifica o quanto o tamanho da imagem é reduzido em cada escala da imagem.
    - Caso igual 1.03, significa a redução do tamanho em 3%, com isso aumenta-se a chance de um tamanho correspondente ao modelo para detecção ser encontrado;
  - MinNeighbors:
    - Especifica quantos vizinhos cada retângulo candidato deve ter para retê-lo. Pode afetar a qualidade dos rostos detectados.
    - Valores mais altos resultam em menos detecções, mas com qualidade superior.
    - Quanto maiores os valores, menor será o número de falsos positivos e menor será o erro em termos de detecção falsa de rostos. No entanto, há uma chance de perder alguns traços faciais pouco claros também

OpenCV
Detecção de Objetos

- Exemplo:

```
import numpy as np
import cv2
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread('Grupo_Caxias_001.jpg')
#img = cv2.imread("Imagem_Pessoas_Juntas_001.jpg")
#img = cv2.imread("Imagem_Pessoas_Juntas_002.jpg")
cv2.imshow('Original',img)
cv2.waitKey(0)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.3, 5)  # Alterem para testar mais

for (x,y,w,h) in faces:   # Procura faces
 img = cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,255),2)
 roi_gray = gray[y:y+h, x:x+w]
 roi_color = img[y:y+h, x:x+w]
 eyes = eye_cascade.detectMultiScale(roi_gray)
 for (ex,ey,ew,eh) in eyes: # Procura olhos nas faces
     cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,0,255),2)
cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

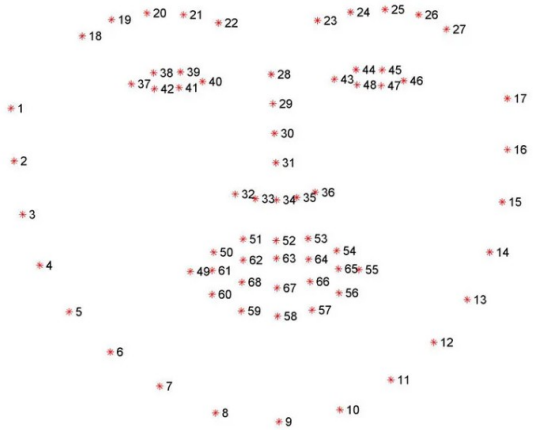
Executar: VisaoComputacional_OpenCV_Exemplo_17 :

OpenCV
DLIB

- Poderosa ferramenta de aprendizado de máquina e visão computacional desenvolvida em C++, mas com uma interface acessível em Python;
- Amplamente utilizada para tarefas como reconhecimento facial, detecção de objetos e aprendizado profundo;
- Aqui estão alguns pontos-chave sobre seu funcionamento:
  - **Detecção de Faces**: Utiliza técnicas como HOG (Histogram of Oriented Gradients) e redes neurais convolucionais (CNN) para detectar faces em imagens e vídeos.
  - **Reconhecimento Facial**: Gera codificações faciais (vetores de 128 dimensões) que representam características únicas de cada rosto, permitindo a comparação e reconhecimento de faces.
  - **Detecção de Objetos**: Além de faces, pode ser treinada para detectar outros objetos personalizados em imagens.
  - **Ferramentas de Processamento de Imagens**: Inclui funcionalidades para carregar, salvar, redimensionar e converter imagens entre diferentes espaços de cores.
  - **Algoritmos de Aprendizado de Máquina**: Implementa internamente vários algoritmos como SVM (Support Vector Machines), KNN (K-Nearest Neighbors) e redes neurais, facilitando a criação de sistemas de reconhecimento e detecção com poucas linhas de código.

OpenCV
DLIB

- Amplamente utilizada para detecção de *landmarks* faciais;
- *Landmarks* são pontos específicos no rosto que ajudam a identificar características como olhos, nariz, boca e contorno do rosto;
- A dlib pode detectar até 68 pontos de referência faciais em uma imagem;
- Como usar:
  - **dlib.get_frontal_face_detector()**



OpenCV DLIB

Exemplo – Face Detector:

```
import dlib
import cv2

detector = dlib.get_frontal_face_detector()

image_path = "Grupo_Caxias_001.jpg"
#image_path = ("Imagem_Pessoas_Juntas_001.jpg")

image = cv2.imread(image_path)
cv2.imshow('imgPre',image)
cv2.waitKey(0)

faces = detector(image)

for face in faces:
    x, y, w, h = face.left(), face.top(), face.width(), face.height()
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)

cv2.imshow('imgPos',image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OpenCV
DLIB

- Para usar landmarks baixar:
  - **shape_predictor_68_face_landmarks.dat**
- Para invocar:
  - **dlib.shape_predictor(shape_predictor_68_face_landmarks.dat)**

OpenCV DLIB

- Exemplo – Landmarks:

```
import numpy as np
import cv2
import dlib

detector = dlib.get_frontal_face_detector()

JAWLINE_POINTS = list(range(0, 17))
RIGHT_EYEBROW_POINTS = list(range(17, 22))
LEFT_EYEBROW_POINTS = list(range(22, 27))
NOSE_POINTS = list(range(27, 36))
RIGHT_EYE_POINTS = list(range(36, 42))
LEFT_EYE_POINTS = list(range(42, 48))
MOUTH_OUTLINE_POINTS = list(range(48, 61))
MOUTH_INNER_POINTS = list(range(61, 68))

predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")  #Já baixado

# Read the image
#image_path = "Imagem_Pessoas_Juntas_001.jpg"
image_path = "Grupo_Caxias_001.jpg"

image = cv2.imread(image_path)
faces = detector(image)
print("Achadas {0} faces!".format(len(faces)))
```

OpenCV DLIB

- Exemplo – Landmarks:

```
for face in faces:
    x, y, w, h = face.left(), face.top(), face.width(), face.height()
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)
    dlib_rect = dlib.rectangle(int(x), int(y), int(x + w), int(y + h))
    landmarks = np.matrix([[p.x, p.y]
                for p in predictor(image, dlib_rect).parts()])
    landmarks_display = landmarks[RIGHT_EYE_POINTS + LEFT_EYE_POINTS + NOSE_POINTS + JAWLINE_POINTS +
     MOUTH_OUTLINE_POINTS]
    for idx, point in enumerate(landmarks_display):
      pos = (point[0, 0], point[0, 1])
      cv2.circle(image, pos, 2, color=(0, 255, 255), thickness=-1)

cv2.imshow("Landmarks achados", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar: VisaoComputacional_OpenCV_DLIB_004

Mudar as imagens!

OpenCV
Reconhecimento Facial

- Aspectos-chave do reconhecimento facial:
  - Captura da imagem: uso de câmeras para capturar imagens faciais.
  - Extração de características: identificação e extração de características faciais distintas.
  - Correspondência facial: comparação das características extraídas com modelos armazenados.
  - Verificação/identificação: determinação da identidade ou verificação da pessoa com base na comparação.

OpenCV Reconhecimento Facial

- Onde aplicar?
  - Controle de acesso: conceder ou restringir acesso a áreas seguras com base no reconhecimento facial.
  - Aplicação da lei: identificar suspeitos e pessoas desaparecidas usando imagens de vigilância.
  - Smartphones: desbloquear dispositivos e autorizar pagamentos por meio de reconhecimento facial.
  - Dispositivos pessoais: personalizar perfis de usuário e preferências em dispositivos pessoais.

OpenCV
Reconhecimento Facial

- Onde aplicar?
  - Insights do cliente: analisar dados demográficos e comportamento do cliente para personalizar estratégias de marketing.
  - Experiências de compra personalizadas: oferecer recomendações com base em clientes reconhecidos.
  - Identificação do paciente: garantir a identificação precisa do paciente e reduzir erros médicos.
  - Monitoramento de saúde: usar análise facial para detecção precoce de certas condições de saúde.
  - Monitoramento do motorista: aumentar a segurança monitorando a atenção e o estado de alerta do motorista.

OpenCV
Reconhecimento Facial

- Etapas:
  - A primeira tarefa é detectar rostos em uma imagem ou fluxo de vídeo.
  - Uma vez que a localização exata ou as coordenadas do rosto são identificadas, o rosto é extraído para processamento posterior.
  - Após cortar o rosto da imagem, as características são extraídas usando embeddings do rosto.
  - Uma rede neural processa a imagem do rosto e gera um vetor representando as características mais essenciais do rosto, são os embeddings de rosto.
  - Durante o treinamento, a rede aprende a produzir vetores semelhantes para rostos de aparência semelhante.
  - **Por exemplo, múltiplas imagens de uma pessoa em diferentes momentos resultarão em vetores semelhantes, já que as principais características faciais permanecem relativamente constantes**.

OpenCV
Reconhecimento Facial

- Etapas:
  - Treinar tal rede requer dados extensos e poder computacional.
  - Passam-se todas as imagens por essa rede pré-treinada para obter os respectivos embeddings, que são então salvos em um arquivo para uso futuro.
  - Com embeddings de rosto salvos para cada rosto em nossos dados, o próximo passo é reconhecer uma nova imagem que não esteja em nossos dados.
  - Calcula-se o embedding de rosto para essa nova imagem usando a mesma rede e compara-se com os embeddings salvos.
  - Um rosto é reconhecido se seu embedding for mais próximo ou similar a qualquer embedding salvo.
  - Este processo identifica e reconhece rostos de forma eficiente comparando vetores no espaço do embedding.

OpenCV
DLIB face-recognition

- Uma das mais simples e eficazes para o reconhecimento facial em Python;
- Construída sobre os algoritmos de última geração da dlib;
- Pontos importantes sobre seu funcionamento:
  - Detecção de Faces: Utiliza técnicas avançadas de aprendizado de máquina para detectar faces em imagens e vídeos com alta precisão.
  - Codificação Facial: Gera vetores de 128 dimensões que representam características únicas de cada rosto, permitindo a comparação e reconhecimento de faces.
  - Manipulação de Imagens: Pode encontrar e manipular características faciais como olhos, nariz, boca e contorno do rosto.
- Uso Simples: Oferece uma API fácil de usar e uma ferramenta de linha de comando para realizar reconhecimento facial em pastas de imagens;

OpenCV
DLIB face-recognition

- Como usar:
  - Para carregar imagens:
    - **imagem = face_recognition.load_image_file(imagem)**
  - Para identificar as faces:
    - **face-locations = face_recognition.face_locations(imagem)**
  - Para fazer o encoding das imagens:
    - **face_encodings =face_recognition.face_encodings(imagem,face_locations)**
  - Para comparar as imagens:
    - **face_recognition.compare_faces(known_encodings,face_encoding[,tolerance=tolerância])**
      - Por padrão, os rostos são considerados correspondentes se a distância euclidiana entre os vetores dos rostos for 0,6 ou menos(valor padrão!).
      - Usar valores de tolerância menores que 0,6 tornará a comparação mais rigorosa.

OpenCV
DLIB face-recognition

Exemplo – Identificação facial puramente:

```
import face_recognition
import cv2

# Carregar a imagem
image = face_recognition.load_image_file("Imagem_Pessoas_Juntas_002.jpg")
#image = face_recognition.load_image_file("Imagem_Pessoas_Juntas_001.jpg")
#image = face_recognition.load_image_file("Grupo_Caxias_001.jpg")
# Encontrar todas as faces na imagem
face_locations = face_recognition.face_locations(image)
print(len(face_locations))
# Carregar a imagem usando OpenCV para desenhar retângulos
image_cv = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

# Desenhar retângulos ao redor das faces detectadas
for (top, right, bottom, left) in face_locations:
    cv2.rectangle(image_cv, (left, top), (right, bottom), (0, 255, 0), 2)
font = cv2.FONT_HERSHEY_SIMPLEX
y,x,c = image.shape
b=int(.95*y)
cv2.putText(image_cv,'Pessoas na Foto! ' + str(len(face_locations)),(50,b), font,1,(0,0,255),2,cv2.LINE_AA)

# Mostrar a imagem com as faces detectadas
cv2.imshow("Faces Detectadas", image_cv)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar: VisaoComputacional_OpenCV_DLIB_009

OpenCV
DLIB face-recognition

- Exemplo – Reconhecimento facial:

```
import face_recognition
import cv2

# Carregar a imagem
image = face_recognition.load_image_file("Grupo_Caxias_001.jpg")

# Encontrar todas as faces na imagem e fazer o encoding
face_locations = face_recognition.face_locations(image)
face_encodings = face_recognition.face_encodings(image, face_locations)

# Carregar imagens de referência e aprender a reconhecer
known_image = face_recognition.load_image_file("Template_Caxias_001.jpg")
known_encoding = face_recognition.face_encodings(known_image)[0]

# Iniciar variáveis
known_encodings = [known_encoding]
known_names = ["Paulo Cientista"]

# Carregar a imagem usando OpenCV para desenhar retângulos
image_cv = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
```

OpenCV
DLIB face-recognition

- Exemplo – Reconhecimento facial(continuação):

```
# Reconhecer faces na imagem
for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
    matches = face_recognition.compare_faces(known_encodings, face_encoding)
    name = "?"

    if True in matches:
        first_match_index = matches.index(True)
        name = known_names[first_match_index]

    # Desenhar retângulo ao redor da face e adicionar o nome
    cv2.rectangle(image_cv, (left, top), (right, bottom), (0, 255, 0), 2)
    cv2.putText(image_cv, name, (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

# Mostrar a imagem com as faces reconhecidas
cv2.imshow("Faces reconhecidas", image_cv)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Executar: VisaoComputacional_OpenCV_DLIB_010

OpenCV
DLIB face-recognition

- Exercício 8:

  Precisamos das fotos individuais de cada um de nós e a foto de todos juntos, vamos fazer o reconhecimento facial.

OpenCV
Capturando vídeo de uma câmera

- Realizado pela função VideoCapture() que retorna um objeto VideoCapture;
- Precisa de um índice de dispositivo como parâmetro. O computador pode ter várias câmeras conectadas!
- Elas são enumeradas por um índice começando em 0 para webcam embutida;
- Após a câmera ser aberta, pode-se ler os quadros sucessivos dela pela função read() que lê o próximo quadro disponível e retorna um valor (True/False);
- O quadro é renderizado no espaço de cor desejado com a função cvtColor() e exibido na janela OpenCV;.

OpenCV
Capturando vídeo de uma câmera

- Para capturar o quadro atual em um arquivo de imagem, utiliza-se a função imwrite();
- Para salvar a transmissão ao vivo da câmera em um arquivo de vídeo, utiliza-se a função VideoWriter();
- Sintaxe: VideoWriter(filename, fourcc, fps, frameSize)
  - Onde:
    - filename – nome do arquivo.
    - fourcc - código padronizado para codecs de vídeo. O OpenCV suporta: DIVX, XVID, MJPG, X264 etc.
    - fps e framesize dependem do dispositivo de captura de vídeo.
- A função VideoWriter() retorna um objeto de fluxo VideoWrite, no qual os quadros capturados são sucessivamente gravados em um loop.

OpenCV
Capturando vídeo de uma câmera

- Exemplo:

```
import cv2 as cv
cam = cv.VideoCapture(0)   # Aciono a câmera

if not cam.isOpened():
 print("Error opening camera")
 exit()
while True:
 # Capture frame-by-frame
 ret, frame = cam.read()
 # if frame is read correctly ret is True
 if not ret:
     print("Error in retrieving frame")
     break
 cv.imshow('Que Horror!', frame)


# Press 'q' to quit
 if cv.waitKey(10)== ord('q'):
     break
cam.release()
cv.destroyAllWindows()
```

Executar Visao_Computacional_OpenCV_Exemplo_18 (necessário ter a câmera conectada!)

OpenCV
DLIB video com landmarks

- **Executar VisaoComputacional_OpenCV_DLIB_001**

OpenCV
Reproduzindo um vídeo

- A função VideoCapture() também pode recuperar quadros de um arquivo de vídeo gravado em alguma mídia;
- Basta substituir o índice da câmera pelo nome do arquivo de vídeo a ser reproduzido na janela OpenCV;
- Sob o prompt do DOS, emitir o comando:
  - **pip install ffpyplayer**

OpenCV
Reproduzindo vídeo pré-gravado

- Exemplo (reproduzindo um vídeo)::

```
import cv2

def redim(img, largura): #função para redimensionar uma imagem
 alt = int(img.shape[0]/img.shape[1]*largura)
 img = cv2.resize(img, (largura, alt), interpolation = cv2.INTER_AREA)
 return img

from ffpyplayer.player import MediaPlayer

file="Video_Curso_002.mp4"
video=cv2.VideoCapture(file)
player = MediaPlayer(file)
while True:
 ret, frame=video.read()
 audio_frame, val = player.get_frame()
 if not ret:
     print("End of video")
     break
 cv2.waitKey(20)
 cv2.imshow("Video", redim(frame,1280))
 if val != 'eof' and audio_frame is not None:
     img, t = audio_frame
video.release()
cv2.destroyAllWindows()
```

OpenCV
Reproduzindo vídeo pré-gravado

- Exemplo (gravando cada frame em disco):

```
import cv2
import os
cam = cv2.VideoCapture("Video_Curso_002.mp4")
frameno = 0
while(True):
    ret,frame = cam.read()
    if ret:
# if video is still left continue creating images
        name = str(frameno) + '.jpg'
        print ('new frame captured...' + name)
        #cv2.imwrite(name, frame)    # Comentei o ato da gravação em si, tire o comentário. São 275 frames!!!
        frameno += 1
    else:
        break
cam.release()
cv2.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo_182

OpenCV
Capturando vídeo de uma câmera

- Exemplo (procurando faces no vídeo):

```
import cv2
def redim(img, largura): #função para redimensionar uma imagem
 alt = int(img.shape[0]/img.shape[1]*largura)
 img = cv2.resize(img, (largura, alt), interpolation = cv2.INTER_AREA)
 return img

df = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

camera = cv2.VideoCapture(0)
while True:
    (sucesso, frame) = camera.read()
    if not sucesso: #final do vídeo
        break
    frame = redim(frame,1280)
    frame_pb = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = df.detectMultiScale(frame_pb, 1.3,2)
    frame_temp = frame.copy()
    for (x, y, lar, alt) in faces:
        cv2.rectangle(frame_temp, (x, y), (x + lar, y + alt), (0,255, 255), 3)
        cv2.imshow("Procurando faces.....", redim(frame_temp, 640))
        if cv2.waitKey(1) & 0xFF == ord("s"):
            break
camera.release()
cv2.destroyAllWindows()
```

OpenCV
inRange

- Função inRange:
  - Retorna uma matriz de elementos igual a 255 se os elementos da matriz alvo estiverem entre as duas matrizes que representam os limites superiores e os limites inferiores
  - Ou retorna uma matriz de elementos igual a 0 se os elementos da matriz fornecida não estiverem entre as duas matrizes que representam os limites superiores e os limites inferiores.
  - Sintaxe:
    - **resultarray inRange(sourcearray, upperboundarray, lowerboundarray)**
    - Onde:
      - sourcearray é o array cujos elementos devem ser comparados com os arrays que representam limites superiores e inferiores.
      - upperboundarray é o array que consiste em elementos que representam limites superiores.
      - lowerboundarray é o array que consiste em elementos que representam limites inferiores.
      - resultarray é o array que representa os elementos iguais a 255 ou 0 retornados da função inRange().
  - Melhor resultado com o padrão HSV de cores.

OpenCV
inRange

- Exemplo (rastreando o carro azul):

```
import numpy as np
import cv2
azulEscuro = np.array([100, 67, 0], dtype = "uint8")
azulClaro = np.array([255, 128, 50], dtype = "uint8")

camera = cv2.VideoCapture('Video_Curso_004.mp4')
while True:
    (sucesso, frame) = camera.read()
    if not sucesso:
        break
    obj = cv2.inRange(frame, azulEscuro, azulClaro)
    obj = cv2.GaussianBlur(obj, (3, 3), 0)
    cnts,hierarchy = cv2.findContours(obj.copy(),cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if len(cnts) > 0:
        cnt = sorted(cnts, key = cv2.contourArea, reverse = True)[0]
        rect = np.int32(cv2.boxPoints(cv2.minAreaRect(cnt)))
        cv2.drawContours(frame, [rect], -1, (0, 255, 255), 2)
        cv2.imshow("Tracking", frame)

        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
camera.release()
cv2.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo_186

OpenCV
bitwise

- Função bitwise:
  - São operações bitwise(bit a bit) de AND, OR, NOT e XOR.
  - São úteis para extrair qualquer parte da imagem e trabalhar com ROIs não retangulares.
  - Sintaxe:
    - **bitwise_conector(image1,image2[, máscara])**
    - Onde:
      - conector: and, or, not, xor.
      - image1: imagem de interesse 1
      - image2: imagem de interesse 2
      - mascara: máscara para selecionar a area de interesse

OpenCV
bitwise

- Exemplo:

```
import cv2 as cv

# Load two images
img1 = cv.imread('Imagem_Pessoas_Juntas_002.jpg')
#img2 = cv.imread('icons8-opencv-48.png')
img2 = cv.imread('LogoB4H.png')

assert img1 is not None, "file could not be read, check with os.path.exists()"
assert img2 is not None, "file could not be read, check with os.path.exists()"
cv.imshow('Logo',img2)
cv.waitKey(0)
# I want to put logo on top-left corner, So I create a ROI
rows,cols,channels = img2.shape
roi = img1[0:rows, 0:cols]
cv.imshow('ROI',roi)
cv.waitKey(0)
```

OpenCV
bitwise

- Exemplo:

```
# Now create a mask of logo and create its inverse mask also
img2gray = cv.cvtColor(img2,cv.COLOR_BGR2GRAY)
ret, mask = cv.threshold(img2gray, 10, 255, cv.THRESH_BINARY)

mask_inv = cv.bitwise_not(mask)

# Now black-out the area of logo in ROI
img1_bg = cv.bitwise_and(roi,roi,mask = mask_inv)

# Take only region of logo from logo image.
img2_fg = cv.bitwise_and(img2,img2,mask = mask)

# Put logo in ROI and modify the main image
dst = cv.add(img1_bg,img2_fg)
img1[0:img2.shape[0], 0:img2.shape[1]] = dst

cv.imshow('Final',img1)
cv.waitKey(0)
cv.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo_19

OpenCV
bitwise

- Exemplo:

```
import cv2 as cv

img1 = cv.imread('night_sky.jpg')
img2 = cv.imread('moon.jpg')


img2 = cv.resize(img2, None, fx = 0.5, fy = 0.5)
cv.imshow('IMG1',img1)
cv.waitKey(0)
cv.imshow('IMG2',img2)
cv.waitKey(0)

img_2_shape = img2.shape
roi = img1[0:img_2_shape[0],0:img_2_shape[1]]
cv.imshow('ROI',roi)
cv.waitKey(0)

img2gray = cv.cvtColor(img2,cv.COLOR_BGR2GRAY)

ret, mask = cv.threshold(img2gray, 10, 255, cv.THRESH_BINARY)
cv.imshow('Mask',mask)
cv.waitKey(0)
```

OpenCV
bitwise

- Exemplo:

```
mask_inv = cv.bitwise_not(mask)
cv.imshow('Mask Invertida',mask_inv)
cv.waitKey(0)

# Now black-out the area of moon in ROI
img1_bg = cv.bitwise_and(roi,roi,mask = mask_inv)
cv.imshow('IMG1_bg',img1_bg)
cv.waitKey(0)

# Take only region of moon from moon image.
img2_fg = cv.bitwise_and(img2,img2,mask = mask)
cv.imshow('img2_fg',img2_fg)
cv.waitKey(0)
# Put moon in ROI and modify the main image
dst = cv.add(img1_bg,img2_fg)
img1[0:img_2_shape[0], 0:img_2_shape[1]] = dst

#Create resizable windows for our display images
cv.imshow('img2_fg',img2_fg)
cv.waitKey(0)
cv.imshow('Imagem Final',img1)
cv.waitKey(0)
cv.destroyAllWindows()
```

OpenCV
bitwise e inRange

- Exemplo:

```
import cv2
import numpy as np

def nothing(x):
    pass

# Open the camera
cap = cv2.VideoCapture(0)

# Create a window
cv2.namedWindow('image')

# create trackbars for color change
cv2.createTrackbar('lowH','image',0,179,nothing)
cv2.createTrackbar('highH','image',179,179,nothing)

cv2.createTrackbar('lowS','image',0,255,nothing)
cv2.createTrackbar('highS','image',255,255,nothing)

cv2.createTrackbar('lowV','image',0,255,nothing)
cv2.createTrackbar('highV','image',255,255,nothing)
```

OpenCV
bitwise e inRange

- Exemplo:

```
while(True):
    ret, frame = cap.read()

    # get current positions of the trackbars
    ilowH = cv2.getTrackbarPos('lowH', 'image')
    ihighH = cv2.getTrackbarPos('highH', 'image')
    ilowS = cv2.getTrackbarPos('lowS', 'image')
    ihighS = cv2.getTrackbarPos('highS', 'image')
    ilowV = cv2.getTrackbarPos('lowV', 'image')
    ihighV = cv2.getTrackbarPos('highV', 'image')

    # convert color to hsv because it is easy to track colors in this color model
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    lower_hsv = np.array([ilowH, ilowS, ilowV])
    higher_hsv = np.array([ihighH, ihighS, ihighV])
    # Apply the cv2.inrange method to create a mask
    mask = cv2.inRange(hsv, lower_hsv, higher_hsv)
    # Apply the mask on the image to extract the original color
    frame = cv2.bitwise_and(frame, frame, mask=mask)
    cv2.imshow('image', frame)
    # Press q to exit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Executar VisaoComputacional_OpenCV_Exemplo_192

OpenCV
bitwise e inRange

- Exemplo:

```
import cv2
import numpy as np

# Função para detectar a cor amarela
def detect_yellow(image_path):
    # Carregar a imagem
    image = cv2.imread(image_path)

    # Converter a imagem para o espaço de cor HSV
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Definir o intervalo de cor para amarelo
    lower_yellow = np.array([23, 93, 0])  # Intervalo inferior do amarelo
    upper_yellow = np.array([45, 255, 255])  # Intervalo superior do amarelo

    # Criar a máscara para a cor amarela
    yellow_mask = cv2.inRange(hsv_image, lower_yellow, upper_yellow)

    # Aplicar a máscara na imagem original
    yellow_result = cv2.bitwise_and(image, image, mask=yellow_mask)
```

OpenCV
bitwise e inRange

- Exemplo:

```
# Mostrar os resultados
    cv2.imshow('Original Image', image)
    cv2.waitKey(0)
    cv2.imshow('Yellow Mask', yellow_mask)
    cv2.waitKey(0)
    cv2.imshow('Detected Yellow Color', yellow_result)
    cv2.waitKey(0)

    # Esperar até que uma tecla seja pressionada e fechar as janelas
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Caminho da imagem
image_path = 'Imagens_Figuras_Geometricas_001.jpg'

# Detectar a cor amarela
detect_yellow(image_path)
```

Executar Executar VisaoComputacional_OpenCV_Exemplo_195

OpenCV
bitwise e inRange

- Executar Executar VisaoComputacional_OpenCV_Exemplo_194

OpenCV
Exercícios

- Exercício 9:
  - Utilizando a figura Varios_Parafusos_Diferentes_002, escreva um código para identificar quais e quantos deles fogem do padrão de tamanho(altura e largura).
  - O padrão é a figura Varios_Parafusos_Diferentes_Padrao_002
- Exercício 10:
  - Utilizando a figura Varios_Botoes_Diferentes_001, escreva um código para contar o número desses botões de acordo com a cor.
- Ambos valerão nota!