

Exercícios Práticos 1



Exercício 1.1 – Leitura e visualização de uma imagem;

OpenCV methods: *imread()*, *namedWindow()*, *imshow()*, *waitKey()*

Exercício 1.2 – Leitura e visualização de um vídeo ou imagens de uma câmara;

OpenCV methods: *VideoCapture()*; *VideoCapture.get()*; *VideoCapture.read()*;

Exercício 1.3 – Redimensionamento de imagens;

OpenCV method: *resize()*.

Exercícios Práticos 2



Exercício 2.1 – *Croma*key (blue screening);

$$\text{ImageOut} = \text{objectImage} \times \text{Mask} + \text{background} \times \text{!Mask}$$



OpenCV methods: *add()*; *multiply()*; *addWeighted()*

Exercício 2.2 – Filtragem de média e mediana;

OpenCV methods: *blur()*; *medianBlur()*;

Exercício 2.3 – Detecção de contornos;

OpenCV methods: *cvtColor()*; *Sobel()*; *Canny()*.

Exercícios Práticos 3



Exercício 3.1 – Histograma de uma imagem;

OpenCV method: *calcHist()*;

Matplotlib (python 2D plotting library) method: *bar()*; *show()*;

Exercício 3.2 –Binarização de uma imagem;

OpenCV method: *threshold()*;

Exercício 3.3 – Operadores morfológicos;

OpenCV methods: *getStructuringElement()*; *morphologyEx()*;
dilate(); *erode()*.

Exercícios Práticos 4



Exercício 4.1 – Labeling;

Local toolbox: `bwLabel` and `psColor`

Used methods: *`bwLabel.labeling()`; `psColor.CreateColorMap()`;
`psColor.Gray2PseudoColor()`;*

OpenCV methods: `findContours()`; `drawContours()`

Exercício 4.2 – Extração de características;

OpenCV methods: `contourArea()`; `moments()`; `arcLength()`;
`boundingRect()`

Exercício 4.3 – Classificação;

Exercícios Práticos 5



Exercício 5.1 – Cálculo do Gradiente

5.1.1 – Determinar o módulo e a fase do gradiente com base num operador diferencial, por exemplo, *Sobel* (exercício 2.3);

5.1.2 – Com base no módulo do gradiente, determine uma imagem de contornos (binarização, exercício 3.2);

Compare os resultados com o algoritmo de *Canny*.

Exercício 5.2 – Extração de características de textura

5.2.1 – Calcule a densidade de contornos;

5.2.2 – Determine um histograma de amplitude e orientação de contornos.

Compare estas características entre várias imagens.

Exercícios Práticos 6



Exercício 6 – Detecção de Movimento

Entrada: Duas imagens monocromáticas $I_n(r, c)$ e $I_{n-k}(r, c)$ ou $I_n(r, c)$ e $B_n(r, c)$ e o limiar τ ;

Saída: imagem binária, I_{out} e conjunto de caixas, B , com a localização dos objectos detectados

Algoritmo com 5 passos:

1. Calcular imagem binária (pixels activos)

$$I_{out}(r, c) = \begin{cases} 1 & \text{se } |I_n(r, c) - I_{n-k}(r, c)| > \tau \\ 0 & \text{caso contrário} \end{cases}$$

2. Realizar operação morfológica de fecho usando um pequeno disco

3. Realizar extracção de componentes conexos sobre I_{out}

4. Remover as regiões com área pequena (ruído)

5. Para cada região, determinar a caixa rectangular que a contém (*bounding box*)

Exercícios Práticos 7



Exercício 7 – Detecção do Campo de Movimento Esparso

Entrada: Duas imagens monocromáticas $I_n(r, c)$ e $I_{n-k}(r, c)$;

Saída: Conjunto de pontos de interesse e os respectivos vetores de movimento.

Algoritmo:

1. Calcular um conjunto de pontos de interesse na imagem do instante anterior $I_{n-k}(r, c)$, por exemplo: cantos;
OpenCV method: *goodFeaturesToTrack()*;
2. Determinar a correspondência destes pontos na imagem do instante atual $I_n(r, c)$;
OpenCV method: *calcOpticalFlowPyrLK()*;
3. Determinar os vetores de movimento e representá-los graficamente;
Matplotlib method: *quiver()*.

Exercícios Práticos 8



Exercício 8.1 – Segmentação de cor com k-médias

Utilizar a função `cv2.kmeans()` para realizar a segmentação de cor, onde cada pixel é representado pelas suas componentes *RGB*.

Exercício 8.2 – Detecção de círculos com base na transformada de Hough

Utilizar a função `cv2.HoughCircles()` para detectar objectos circulares.