Exercícios Práticos 1

Exercício 1.1 – Leitura e visualização de uma imagem;

OpenCV methods: *imread()*, *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 – *Cromakey* (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()*; *GaussianBlur()*;

Exercícios Práticos 2 (cont.)

Exercício 2.3 – Transformações Geométricas

OpenCV methods: *getRotationMatrix2D(); warpAffine();*

$$\begin{bmatrix} x \\ y \end{bmatrix} = T \begin{bmatrix} v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_x \\ t_{21} & t_{22} & t_y \end{bmatrix} \begin{bmatrix} v \\ w \\ 1 \end{bmatrix}$$

(t_x, t_y) definem os parâmetros de translação;

Exemplo de rotação:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \end{bmatrix} \begin{bmatrix} v \\ w \\ 1 \end{bmatrix}$$

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 3 (cont.)

Exercício 3.4 – *Labeling*;

Local toolbox: bwLabel and psColor
Used methods: *bwLabel.labeling(); psColor.CreateColorMap(); psColor.Gray2PseudoColor()*;

OpenCV methods: *findContours()*; *drawContours(); connectedComponents()*

Exercício 3.5 – Extracção de características;

OpenCV methods: *contourArea()*; *moments()*; *arcLength()*; *boundingRect(); connectedComponentsWithStats()*

Exercício 3.6 – Classificação;

Exercícios Práticos 5

Exercício 5.1 – Deteção de contornos;

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

Exercício 5.2 – Cálculo do Gradiente

5.2.1 – Determinar o módulo e a fase do gradiente com base num operador diferencial, por exemplo, *Sobel*;

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

Compare os resultados com o algoritmo de *Canny*.

Exercícios Práticos 6

Exercício 6.1 – Extração de informação de cor

6.1.1 – Converta uma imagem em formato RGB para o espaço de cor HSI e visualize cada componente;

6.1.2 – Determine um histograma de cor.

Compare esta característica entre várias imagens.

Exercício 6.2 – Extração de características de textura

6.2.1 – Calcule a densidade de contornos;

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

Compare estas características entre várias imagens.

Exercícios Práticos 6 (cont.)

Exercício 6.3 – Deteção de tom de pele

Pretende-se realizar a deteção de zonas da imagem com a cor do tom de pele, como por exemplo, faces ou mãos, tipicamente é um tom relativamente característico na componente de crominância.

Procedimento:

1. Converta a imagem do formato RGB (normalmente, o formato devolvido pelo método de leitura de ficheiros) para um formato que separe as componentes de luminância e crominância, como por exemplo, HSI, rg_normalizado ou YCbCr.
2. Na componente de crominância, detete qual o valor médio correspondente ao tom de pele.
3. Determine uma imagem binária calculando os pixéis com valores de crominância perto do valor determinado anteriormente (sugestão, utilize a função do openCV *inRange*);
4. Realize melhoramento (pós-processamento) à imagem binária obtida anteriormente e selecione as regiões com áreas significativas, desenhando os contornos dessas regiões as suas *bounding box*.

Exercícios Práticos 7

Exercício 7.1 – 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)

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

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 (cont.)

Exercício 7.2 – Deteção 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 respetivos 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 7 (cont.)

Exercício 7.3 – Deteção de Segmentos de Vídeo

Entrada: Vídeo com várias mudanças de cena/shots;

Saída: Informação da localização das mudanças de segmentos de vídeo;

Algoritmo:

1. Determinar os histogramas de imagens consecutivas (exercício 3.1);
2. Realizar o gráfico com a diferença entre os histogramas (por exemplo, utilizar a distância L_1);
3. Determinar um limiar para detetar mudanças de segmentos;
4. Indicar em que instantes/*frames* existe mudança.

Exercícios Práticos 8

Exercício 8.1 – Segmentação de cor com k-médias

Utiliar a função *cv2.kmeans()* para realizar a segmenação de cor, onde cada pixel é representado pelas suas components *RGB*.

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

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