Workout 1

Work 1.1 – Reading and showing an image

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

Work 1.2 – Reading and showing a video or images from a camera;

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

Work 1.3 – Image resize with a finger print image

OpenCV method: *resize()*.

Workout 2

Work 2.1 – *Cromakey* (blue screening)

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

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

Work 2.2 – Image smoothing and median filtering

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

Work 2.3 – Edge detector

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

Workout 3

Work 3.1 – Image Histogram

OpenCV method: *calcHist()*;

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

Work 3.2 – Image Binarization

OpenCV method: *threshold()*;

Work 3.3 – Morphological Operators

OpenCV method: *getStructuringElement()*; *morphologyEx()*;

dilate(); *erode()*.

Workout 4

Work 4.1 – Labeling

Local toolbox: bwLabel and psColor

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

OpenCV 2.X methods: *findContours(); drawContours();*

OpenCV 3.X method: *connectedComponents()*

Work 4.2 – Feature Extraction;

OpenCV 2.X methods: contourArea(); moments(); arcLength(); boundingRect()

OpenCV 3.X method: *connectedComponentsWithStats()*

Work 4.3 – Classification;

Workout 5

Work 5.1 – Image Gradient

5.1.1 – Compute the modulus and phase of the image gradient based on a differential operator, for example, *Sobel* (work 2.3);

5.1.2 – Based on the modulus of the gradient, compute a contour image (thresholding, work 3.2);

Compare the results with the *Canny* algorithm.

Work 5.2 – Color Feature Extraction

5.2.1 – Convert an image in RGB format to the HSI color space and visualize each component;

5.2.2 – Compute a color histogram.

Compare these features between several images.

Workout 5 (cont.)

Work 5.3 – Texture Feature Extraction

5.3.1 – Compute the edge density (*edgeness*);

5.3.2 – Compute the normalized histogram of magnitudes and orientations of contours.

Compare these features between several images.

Workout 6

Work 6 – Motion Detection

Input: Two monochrome images $I_n(r,c)$ and $I_{n-k}(r,c)$ or $I_n(r,c)$ and $B_n(r,c)$ and a threshold τ;

Output: binary image, I_{out} and a set of bounding boxes, ***B***, with the location of detected objects;

5 - Step Algorithm:

1. Compute the binary image (active pixels)

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

2. Perform a morphological closing of I_{out} using a small disc;
3. Perform connect components extraction;
4. Remove regions with small area (noise);
5. For each remaining active region, compute the bounding box;

Workout 7

Work 7 – Sparse Motion Field Detection

Input: Two monochrome images $I_n(r,c)$ and $I_{n-k}(r,c)$;

Output: Set of interesting points and their corresponding motion vectors;

Algorithm:

1. Compute a set of interesting points in the previous image, $I_{n-k}(r,c)$, for example, *corners*;

   OpenCV method: *goodFeaturesToTrack()*;

2. Determine the correspondence of these points in the current image, $I_n(r,c)$;

   OpenCV method: *calcOpticalFlowPyrLK()*;

3. Determine the motion vectors and represent them graphically;

   Matplotlib method: *quiver()*.

Workout 8

Work 8.1 – K-means Color Segmentation

Use the OpenCV function *kmeans()* to perform color segmentation, where each pixel is represented by its *RGB* components (use also other color space, for example, HSI).

Work 8.2 – Circle Detection with Hough Transform

Use the OpenCV function *HoughCircles()* to detect circular objects, for example, *coins*.