Master in Informatics and Computing Engineering | 5th Year
EIC0104| Computer Vision | 2015/2016 – 1st Semester
Jorge Silva / Luís Teixeira

OpenCV exercises

1. Images – read, write and display

a) Read the name of a file containing an image in 'jpg' format and show it in a window, whose name is the name of the file. Test whether the image was successfully read. Display the height and width of the image, on the console.

b) Read a color image in 'jpg' format and save it in 'bmp' format.

2. Images – creation

a) Analyse and interpret the result of the following code:

```
Mat img1, img2, img3;
img1 = imread(…);
img2 = img1;
img1.copyTo(img3);
flip(img3,img2,1);
// show the 3 images
```

b) Create an image, having 50(lines)x200(columns) pixels with constant intensity, 100, except the central pixel, whose intensity must be 255. Display the image.

c) Develop a C++ class, Image, for representing an image. It must have 2 constructors: one to construct an Image object from a file, another to construct an image with a given size and constant intensity. Include also a method, getImage(), that returns the image (BE CAREFUL!).

3. Images – representation, grayscale & color spaces

a) Read a color image, display it in one window, convert it to grayscale, display the grayscale image in another window and save the grayscale image to a different file.

b) Read an image (color or grayscale) and add "salt and pepper" noise to it. Suggestion: start by determining the number of image channels. The number of noisy points must be 10% of the total number of image points.

c) Read a color image (in RGB format), split the 3 channels and show each channel in a separate window. Add a constant value to one of the channels, merge the channels into a new color image and show this image.

d) Read a color image (in RGB format), convert it to HSV, split the 3 HSV channels and show each channel in a separate window. Add a constant value to saturation channel, merge the channels into a new color image and show this image.

e) Analyze and run the given code that illustrates alternative ways to access the pixels of an image.

4. Video – acquisition and simple processing

a) Display the video acquired from the webcam (in color) in one window and acquire and save a frame when the user presses the keyboard.

b) Display the video acquired from the webcam (in color) in one window and the result of the conversion of each frame to grayscale in another window.

5. Image enhancement – histogram equalization

a) Take a low contrast image and plot its histogram.

b) Enhance the image constrast using:
b1) simple histogram equalization, or
b2) CLAHE,
and show the resulting enhanced images and their histograms.

6. Image enhancement – filtering

Take a noisy image and filter it (try different filter sizes), using:

a) a mean filter;

b) a Gaussian filter;

c) a median filter;
d) a bilateral filter.

7. Edge detection

Detect the edges of an image using:

a) the Sobel filter (cv::Sobel()); try different thresholds;

b) the Canny filter (cv::Canny()); try different thresholds;

c) compare the outputs of the two filters when the same thresholds are used;

d) the Laplacian filter (cv::Laplacian()); try different apertures;
notes: 1) in order to visualize the result it may be necessary to rescale the resulting values;
2) to isolate the edges it is necessary to detect the zero crossings in the result.

8. Hough transform – line and circle detection

a) Compare the functionality of cv::HoughLines() and cv::HoughLinesP() OpenCV functions for line detection.

b) Use cv::HoughLines() to detect lines in a binary image; try different parameter values; draw the detected lines on the image, using cv::line().

c) Use cv::HoughLinesP() to detect line segments in a binary image; try different parameter values; draw the detected line segments on the image.

d) Take an image containing coins and use cv::HoughCircles() to detect the coins in the image.