Open CV Assignment 1

**1. Describe the algorithm for detecting Canny edges.**

A. The algorithm is based on grayscale pictures. Therefore, the pre-requisite is to convert the image to grayscale before following steps.

The Canny edge detection algorithm is composed of 5 steps:

1. Noise reduction: Edge detection results are highly sensitive to image noise, so we can reduce the noise by applying Gaussian Blur to smooth it
2. Gradient calculation: The Gradient calculation step detects the edge intensity and direction by calculating the gradient of the image using edge detection operators. The result is almost the expected one, but we can see that some of the edges are thick and others are thin.
3. Non-maximum suppression: The final image should have thin edges. Thus, we must perform non-maximum suppression to thin out the edges. The algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions. The result is the same image with thinner edges. However there is still some variation regarding the edges’ intensity like some pixels seem to be brighter than others.
4. Double threshold: The double threshold step aims at identifying 3 kinds of pixels: strong, weak, and non-relevant, where:

* High threshold is used to identify the strong pixels (intensity higher than the high threshold).
* Low threshold is used to identify the non-relevant pixels (intensity lower than the low threshold).
* All pixels having intensity between both thresholds are flagged as weak.

1. Edge Tracking by Hysteresis: Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one.

**2. Convert the MatOfByte object to a byte array using a Python script.**

A. Convert the **MatOfByte** object into a byte array we can use the method toArray() as below:

//Reading the image

Mat image = Imgcodecs.imread(file);

//instantiating an empty MatOfByte class

MatOfByte matOfByte = new MatOfByte();

//Converting the Mat object to MatOfByte

Imgcodecs.imencode(".jpg", image, matOfByte);

byte[] byteArray = matOfByte.toArray();

**3. Describe OpenCV's ColorMaps.**

A. The human perception isn't built for observing fine changes in grayscale images. Human eyes are more sensitive to observing changes between colors, so you often need to recolor your grayscale images. OpenCV comes with various colormaps to enhance the visualization in your computer vision application. OpenCV defines **12 colormaps** that can be applied to a grayscale image using the function **applyColorMap** to produce a pseudocolored image.

EX: import cv2

img\_gray = cv2.imread(“pluto.jpg”, cv2.IMREAD\_GRAYSCALE)

img\_color = cv2.applyColorMap(img\_gray, cv2.COLORMAP\_JET)

**4. Using OpenCV, create a Rectangle.**

A. import cv2

image = cv2.imread('test.jpg')

height, width, channels = image.shape

start\_point = (0,0)

end\_point = (width, height)

color = (255,0,255)

thickness = 5

image = cv2.rectangle(image, start\_point, end\_point, color, thickness)

cv2.imshow('Rectangle',image)

5. **In OpenCV, how many different types of image filters are there?**

A. Filters generally use many pixel for computing each new pixel value but point operations can use one pixel to perform an image processing. The filters can use for blurred or fuzzy the local intensity of image to make it smooth.

**Type of filters:** 2 types: 1. Linear Filtering 2. Non-linear Filtering

**1. Linear filter:** The convolution of matrix pixels and kernel matrix to reduce intensity of image, which is blurring the image.

There are 3 types of linear filter:

* Box filter
* Gauss filter
* Laplace filter or Mexican hat filter

**2. Non Linear Filter:** Using some non-linear function from the source pixel value. The idea is to replace the target pixel value with its neighbor pixels value from some ordering mechanism or function.

There are many types of Non Linear Filter but in this article I will show you just 3 of them:

* Minimum Filter
* Maximum Filter
* Median Filter