**Noam Salomonski 303161194**

**Computer Vision 22928**

**Explanations for Maman11:**

**Question 1:**

Please run **ex1/ex1.py**.

a.

generate\_random\_gaussian\_matrix

parameters:

mean=10, std=5, size=(100, 100)

as requested,.

Linearly normalizing the output to be between 0 and 1, so cv2.imshow can work with it. And show it properly.

b.

draw\_histogram

using 256 bins, setting some arbitrary bin width for visual beauty.

Centering each bar on the average of its data by calculating

center = (bins[:-1] + bins[1:]) / 2

c.

read\_my\_image

reading the image twice-

once into

self.\_\_color\_image

as a color image

and once into

self.\_\_grayscale\_image

as a grayscale image

d.

detect\_edges

applying a canny edge detector by cv2.

Using thresholds: (thres1, thres2) = (300, 250), (500, 300), (1000, 250)

With lower thres1, we see finer details.

With lower thres2 we see lines less broken.

This is given thres1 > thres2.

e.

def detect\_harris(self, block\_size, ksize, k, corner\_threshold): # 1e

Applying a harris detector by cv2.

Drawing found points on the input image.

As **kSize** increases, the edges seem to get blurry. This is because the sobel kernel becomes larger, and it responds more weakly to sharp (small) edges.

I selected the parameter **k** to 0.04, as an empirical result by several posts online.

A bigger **k** should give less false corners, and more lost true corners. (recall-percision tradeoff).

**Block\_size** is the size of neighborhood to apply Harris for each point. Larger would usually mean the point would be more likely to be selected as a positive corner, given the rest of the parameters remain the same.

**Corner\_threshold** is a threshold to eliminate corners whose value is too low and would qualify as false positives. A larger value corresponds to less corners, and to more confidence in corners that pass.

**Question 2:**

Please run **ex2/blob\_detector.py**