1st Programming Assignment: Corner Detection

CSE 6239 (July 2020)

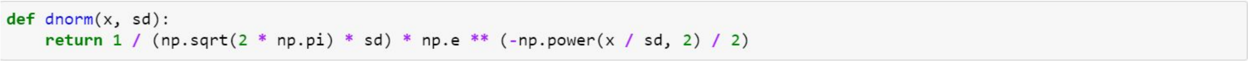
Report By:
Name: Md Mamun Hasan
Roll: 1907555

Description:

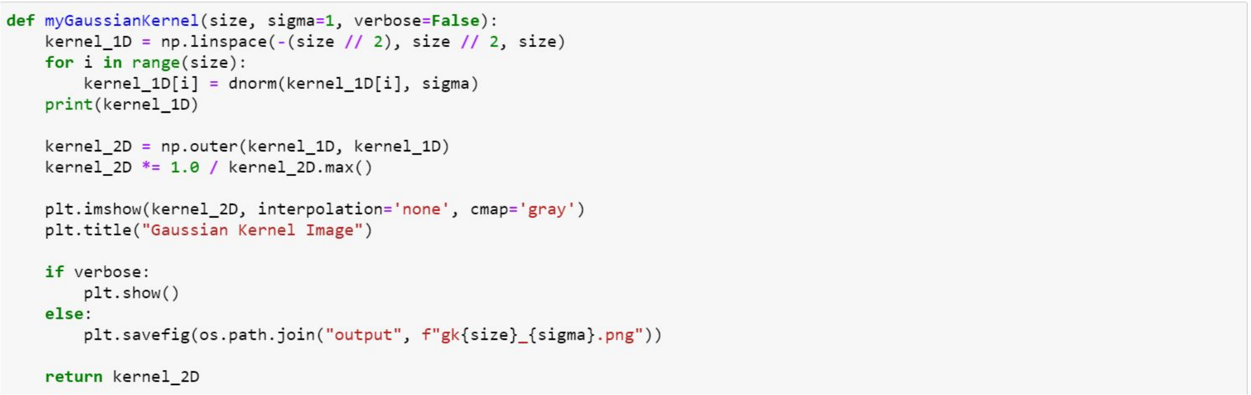
Corner detection executed for 8 images with different criteria is reported below with results.

Image: img1.png

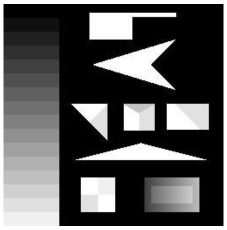
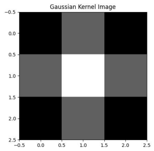
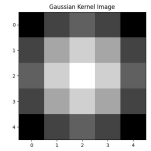
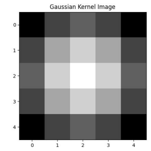
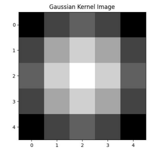
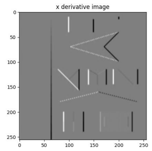
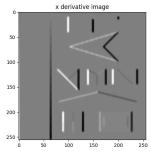
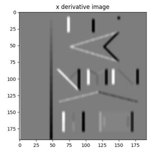
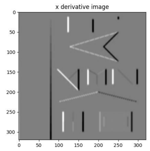
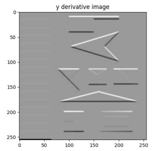
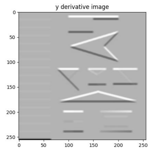
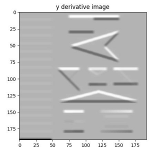
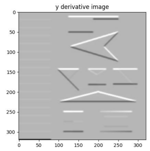
Kernel: Gaussian

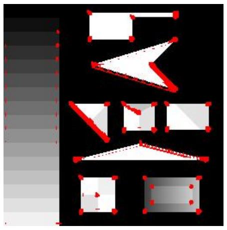
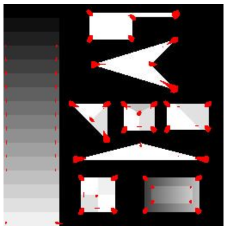
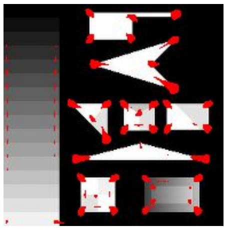
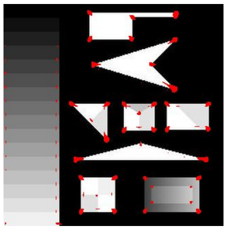
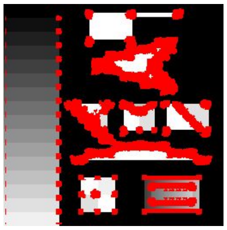
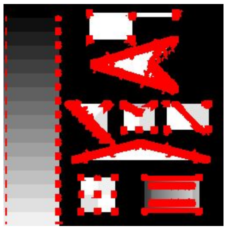
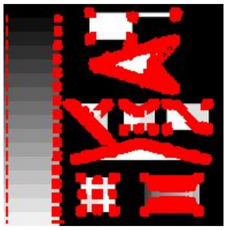
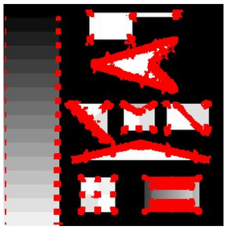
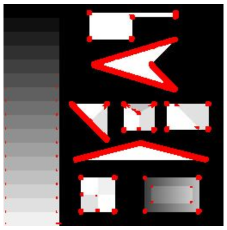
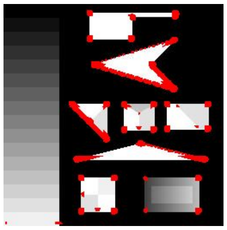
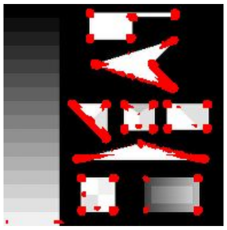
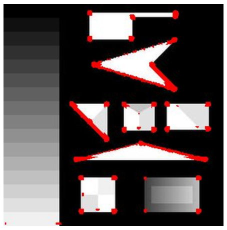


```
def dnorm(x, sd):
    return 1 / (np.sqrt(2 * np.pi) * sd) * np.e ** (-np.power(x / sd, 2) / 2)
```

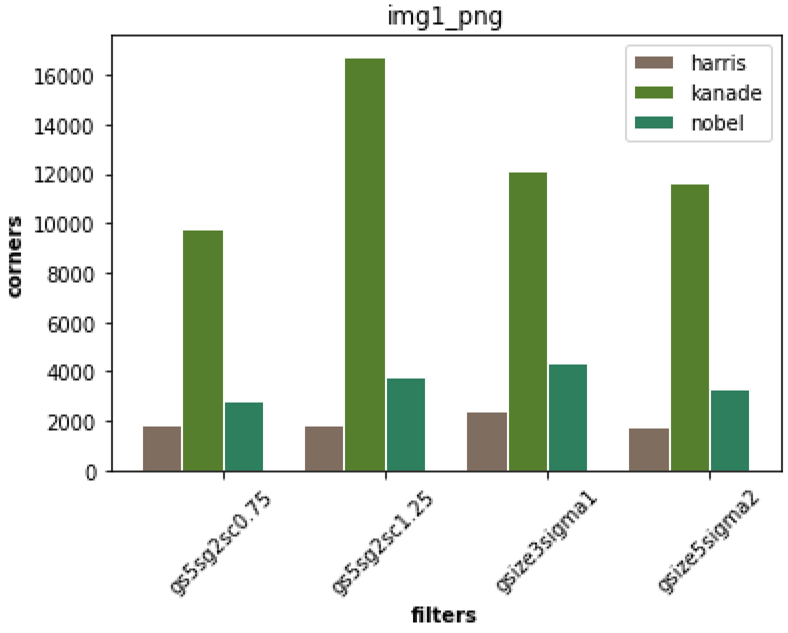


```
def myGaussianKernel(size, sigma=1, verbose=False):
    kernel_1D = np.linspace(-(size // 2), size // 2, size)
    for i in range(size):
        kernel_1D[i] = dnorm(kernel_1D[i], sigma)
    print(kernel_1D)

    kernel_2D = np.outer(kernel_1D, kernel_1D)
    kernel_2D *= 1.0 / kernel_2D.max()

    plt.imshow(kernel_2D, interpolation='none', cmap='gray')
    plt.title("Gaussian Kernel Image")

    if verbose:
        plt.show()
    else:
        plt.savefig(os.path.join("output", f"gk{size}_{sigma}.png"))

    return kernel_2D
```

	Size: 3 Sigma: 1	Size: 5 Sigma: 2	Size: 5 Sigma: 2 Scale: 0.75	Size: 5 Sigma: 2 Scale: 1.25
Kernel				
X Derivative				
Y Derivative				

Harris R Threshold = 10000.00				
Kanade R Threshold = 100.00				
Nobel R Threshold = 1.00				

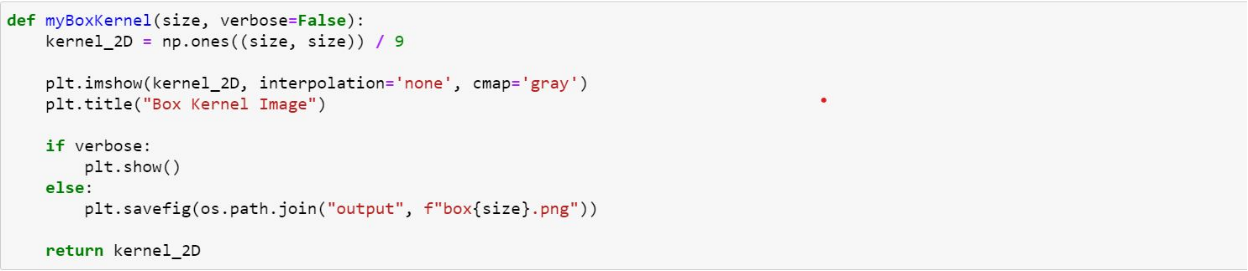
Bar chart for above **Gaussian** filter



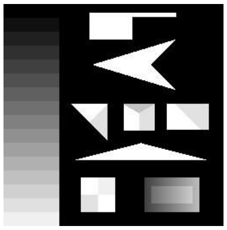
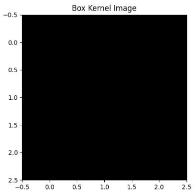
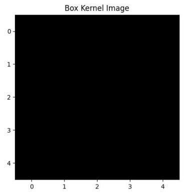
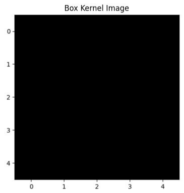
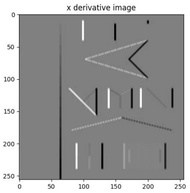
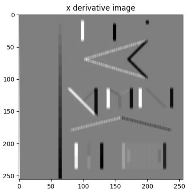
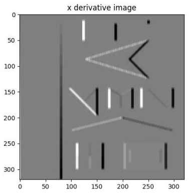
Comments:

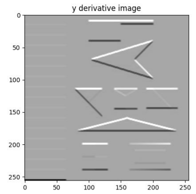
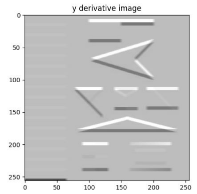
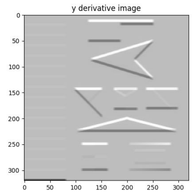
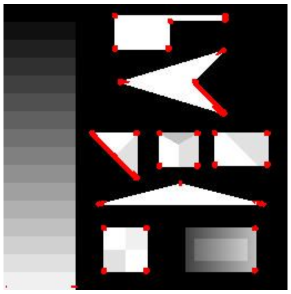
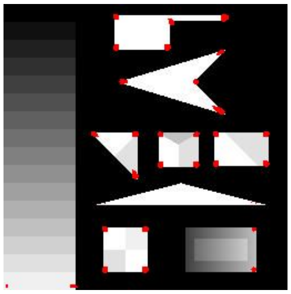
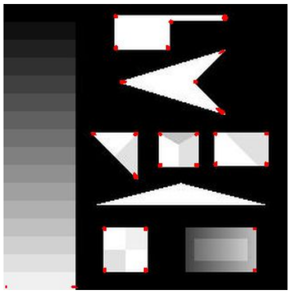
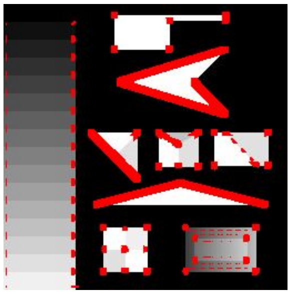
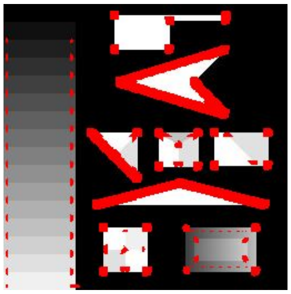
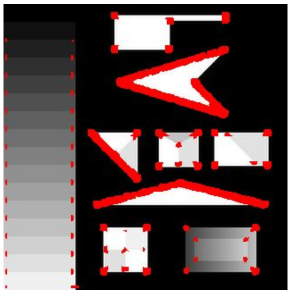
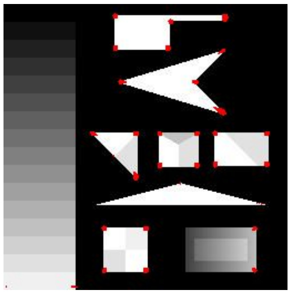
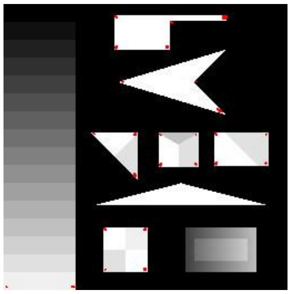
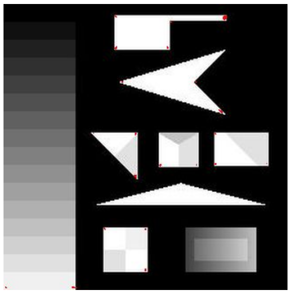
1. Harris algorithm has worked well.
2. Kanade works very bad
3. Scale up detected more accurate corners
4. 15.5% corners are found common for 3 algorithms

Kernel: Box

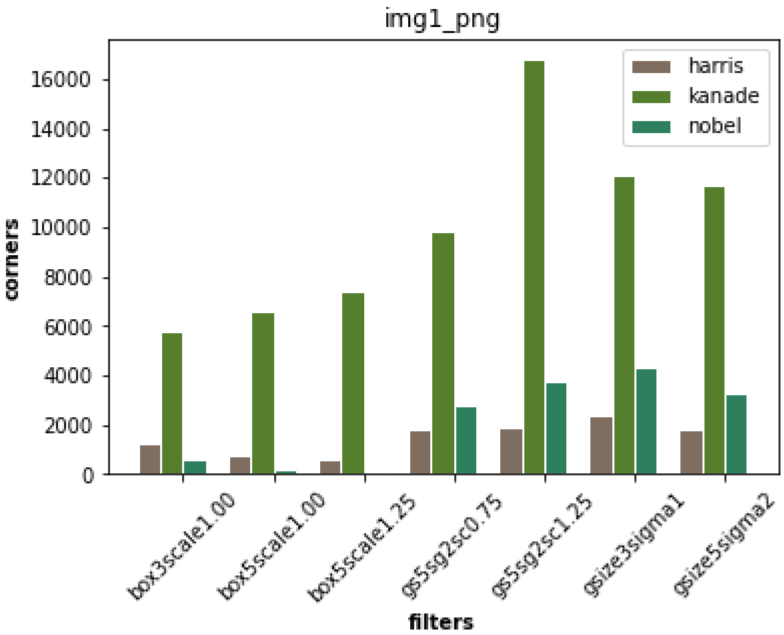


```
def myBoxKernel(size, verbose=False):
    kernel_2D = np.ones((size, size)) / 9

    plt.imshow(kernel_2D, interpolation='none', cmap='gray')
    plt.title("Box Kernel Image")

    if verbose:
        plt.show()
    else:
        plt.savefig(os.path.join("output", f"box{size}.png"))

    return kernel_2D
```

	Size: 3	Size: 5	Size: 5 Scale: 1.25
Kernel			
X Derivative			

Y Derivative			
Harris			
Kanade			
Nobel			

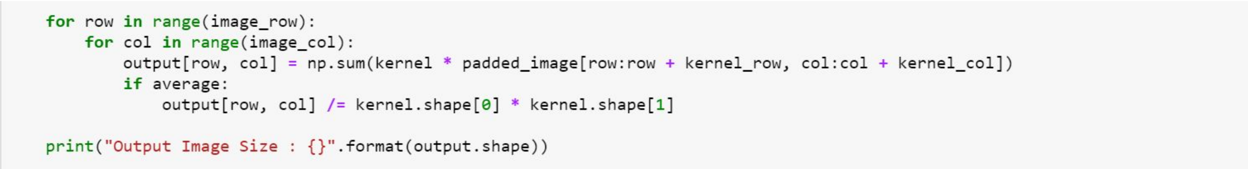
Bar chart for above **Gaussian + Box** filter



Comments:

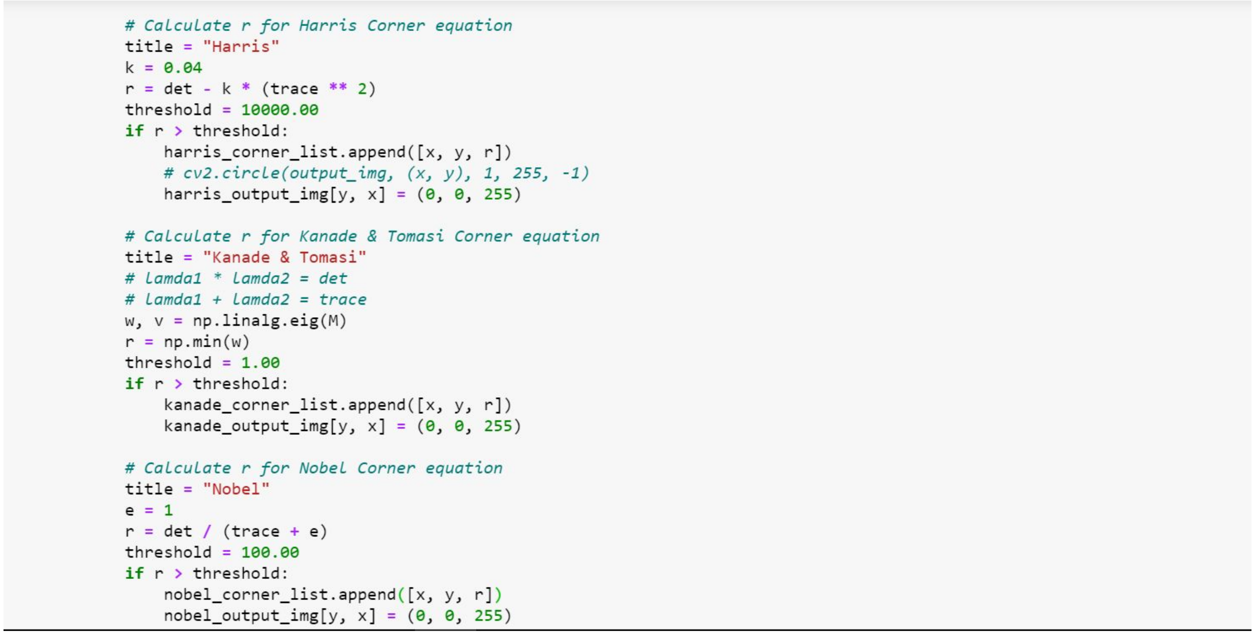
5. Less corner detected in box filter
6. Again scaling up detect more corners

Code from myImageFilter() conv by kernel:



```
    for row in range(image_row):
        for col in range(image_col):
            output[row, col] = np.sum(kernel * padded_image[row:row + kernel_row, col:col + kernel_col])
            if average:
                output[row, col] /= kernel.shape[0] * kernel.shape[1]

    print("Output Image Size : {}".format(output.shape))
```

Code for R calculation:



```
# Calculate r for Harris Corner equation
title = "Harris"
k = 0.04
r = det - k * (trace ** 2)
threshold = 10000.00
if r > threshold:
    harris_corner_list.append([x, y, r])
    # cv2.circle(output_img, (x, y), 1, 255, -1)
    harris_output_img[y, x] = (0, 0, 255)

# Calculate r for Kanade & Tomasi Corner equation
title = "Kanade & Tomasi"
# lamda1 * lamda2 = det
# lamda1 + lamda2 = trace
w, v = np.linalg.eig(M)
r = np.min(w)
threshold = 1.00
if r > threshold:
    kanade_corner_list.append([x, y, r])
    kanade_output_img[y, x] = (0, 0, 255)

# Calculate r for Nobel Corner equation
title = "Nobel"
e = 1
r = det / (trace + e)
threshold = 100.00
if r > threshold:
    nobel_corner_list.append([x, y, r])
    nobel_output_img[y, x] = (0, 0, 255)
```

Similar process applied for all other provided images and For more programming reference please visit here.