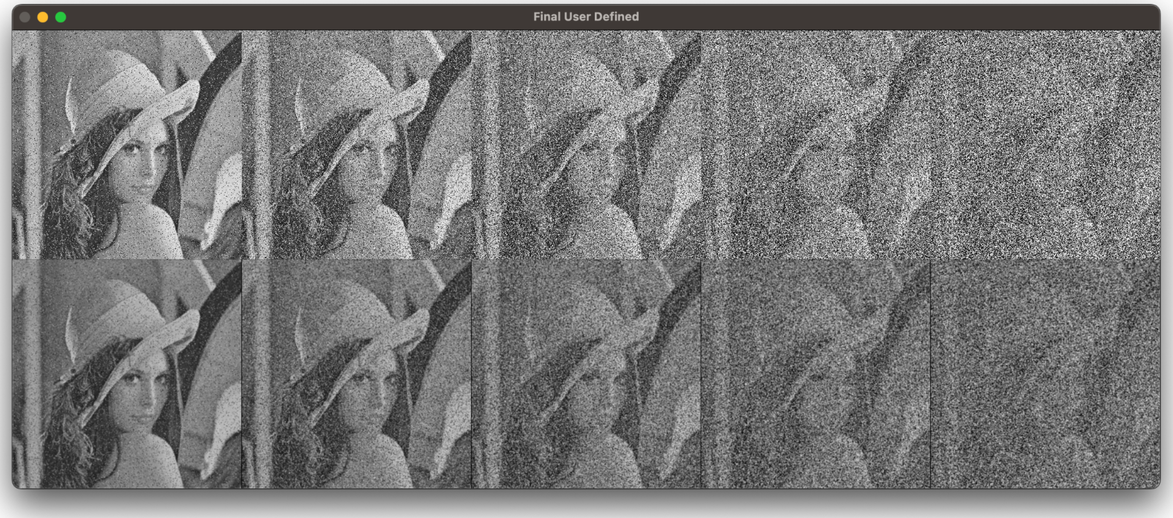
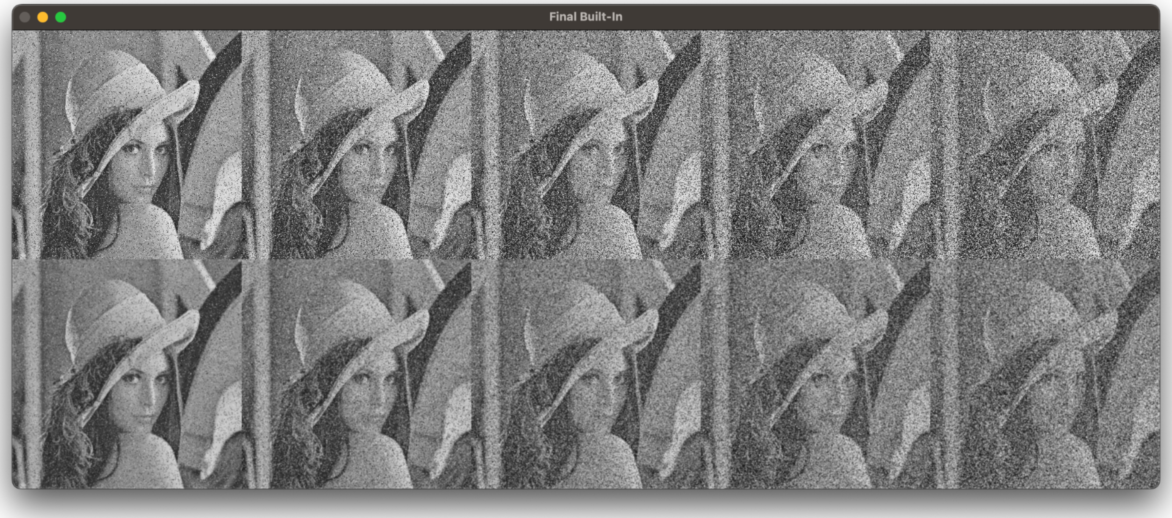
```
"""
   CS20B1097 HIMANSHU

   1. Download Lena color image, convert it to grayscale image and add salt and pepper
   noise with noise quantity 0.1,0.2 up to 1 and generate 10 noisy images.

   2. Correlate each noisy image with Gaussian filters of varying size. Filter size can be
   3 x 3, 5 x 5 and 7 x 7.
"""

import cv2
import numpy as np
from skimage.util import random_noise
import random
import copy

def salt_and_pepper_noise(input_img, noise_quantity):
   output = input_img
   height, width = img.shape
   noisy_pixels = int(height * width * noise_quantity)

   for _ in range(noisy_pixels):
       y = random.randint(0, height - 1)
       x = random.randint(0, width - 1)
       output[y][x] = random.choice([0, 255])

   return output

def find_gaussian(n):
   gaussian = np.zeros((n,n), dtype='double')
   u = v = (n//2)
   multiplier = 1/(2*np.pi)
   for i in range(n//2 + 1):
       for j in range(i+1):
           gaussian_value = multiplier * (np.exp(-(i**2 + j**2)/2))
           gaussian[u-i][v-j] = gaussian_value
           gaussian[u+i][v+j] = gaussian_value
           gaussian[u+j][v-i] = gaussian_value
           gaussian[u-j][v+i] = gaussian_value
   return gaussian

def correlation_gaussian(img, n):
   height = img.shape[0]
   width = img.shape[1]
   padded_img = np.zeros((img.shape[0] + 2*n-2, img.shape[1] + 2*n-2), dtype='uint8')

   u = v = n-1
   for i in range(height):
       for j in range(width):
           padded_img[u+i][v+j] = img[i][j]

   gaussian_filter = find_gaussian(n)
   final_image = np.zeros(img.shape, dtype='double')
   for i in range(n//2, height+n//2-1):
       for j in range(n//2, width+n//2-1):
           value = 0
           for k in range(n-1):
               for l in range(n-1):
                   value += gaussian_filter[k][l] * padded_img[i+k][j+l]

           final_image[i-n//2][j-n//2] = value

   return final_image.astype(np.uint8)



img = cv2.imread('Lena.png', 0)
cv2.imshow('Original Image', img)
filter_size = 3

# User Defined Function
img1 = copy.deepcopy(img)
noisy_images = []
correlated_images = []
for i in range(5):
   # 1) Adding salt pepper noise
   noisy_img = copy.deepcopy(salt_and_pepper_noise(img1, (i+1)*0.1))
```

```
    noisy_images.append(noisy_img)

    # 2) Correlating Noisy image with gaussian filter
    copy_noisy_img = copy.deepcopy(noisy_img)
    correlated_image = copy.deepcopy(correlation_gaussian(copy_noisy_img, filter_size))
    correlated_images.append(correlated_image)

noisy = np.concatenate(noisy_images, axis=1)
correlated = np.concatenate(correlated_images, axis=1)
final = np.concatenate((noisy, correlated), axis=0)
cv2.imshow('Final User Defined', final)


# Built-in Function
img2 = copy.deepcopy(img)
noisy_images_builtin = []
correlated_images_builtin = []
for i in range(5):
    # 1) Adding salt and pepper noise builtin
    noisy_img_builtin = copy.deepcopy(np.array(255*(random_noise(img2,
mode='s&p',amount=(i+1)*0.1)), dtype='uint8'))
    noisy_images_builtin.append(noisy_img_builtin)

    # 2) Correlating Noisy image with gaussian filter builin
    copy_noisy_img_builtin = copy.deepcopy(noisy_img_builtin)
    gaussian_filter_window = find_gaussian(filter_size)
    correlated_image_builtin = copy.deepcopy(cv2.filter2D(copy_noisy_img_builtin, -1,
gaussian_filter_window))
    correlated_images_builtin.append(correlated_image_builtin)


noisy_builtin = np.concatenate(noisy_images_builtin, axis=1)
correlated_builtin = np.concatenate(correlated_images_builtin, axis=1)
final_builtin = np.concatenate((noisy_builtin, correlated_builtin), axis=0)
cv2.imshow('Final Built-In', final_builtin)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT


Final User Defined


Final Built-In