ICE-9 Image Restoration and Reconstruction

Use this image : https://en.wikipedia.org/wiki/Lenna#/media/File:Lenna_(test_image).png

Salt and pepper noise

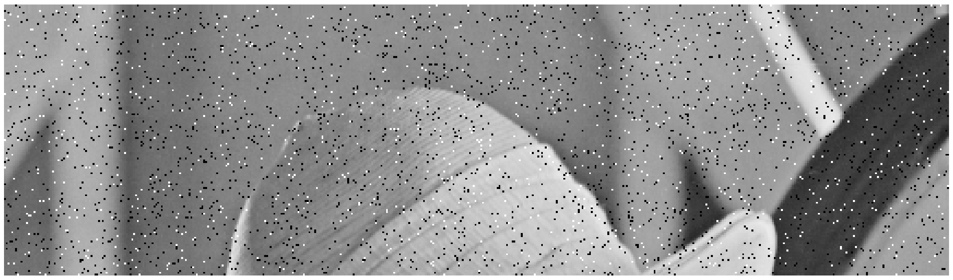
(Tutorial)

- Salt-and-pepper noise can only be added in a grayscale image. So, convert an image to grayscale after reading it
- Randomly pick the number of pixels to which noise is added (number_of_pixels)
- Randomly pick some pixels in the image to which noise will be added. It can be done by randomly picking x and y coordinate
- Note the random values generated must be within the range of the image dimensions. The x and y coordinates must be within the range of the image size Random numbers can be generated using random number generator functions like random.randint used in the code
- Color some randomly picked pixels as black setting their value to 0
- Color some randomly picked pixels as white setting their value to 255 Save the value of the image

```
import random
import cv2

from google.colab.patches import cv2_imshow


def add_noise(img):

    # Getting the dimensions of the image
    row , col = img.shape

    # Randomly pick some pixels in the
    # image for coloring them white
    # Pick a random number between 300 and 10000
    number_of_pixels = random.randint(300, 10000)
    for i in range(number_of_pixels):

        # Pick a random y coordinate
        y_coord=random.randint(0, row - 1)

        # Pick a random x coordinate
        x_coord=random.randint(0, col - 1)
        # Color that pixel to white
        img[y_coord][x_coord] = 255

    # Randomly pick some pixels in
    # the image for coloring them black
    # Pick a random number between 300 and 10000
    number_of_pixels = random.randint(300 , 10000)
    for i in range(number_of_pixels):

        # Pick a random y coordinate
        y_coord=random.randint(0, row - 1)

        # Pick a random x coordinate
        x_coord=random.randint(0, col - 1)

        # Color that pixel to black
        img[y_coord][x_coord] = 0

    return img

# salt-and-pepper noise can
# be applied only to grayscale images
# Reading the color image in grayscale image
img = cv2.imread('Lenna_(test_image).png',
                 cv2.IMREAD_GRAYSCALE)

print ("\n \n Original Image : \n \n ")

cv2_imshow(img)

img2 = add_noise(img)

print ("\n \n Image with salt and pepper noise : \n \n ")

cv2_imshow(img2)
```

```
Original Image :
```

```
Image with salt and pepper noise :
```



Question 1

Explain briefly why is noise found in an image . What are the ways to reduce it ?

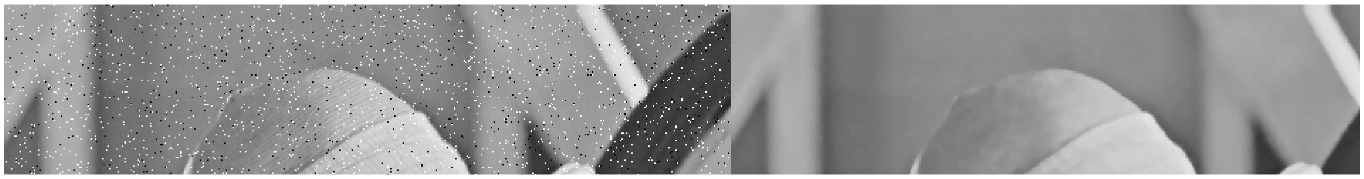
ANSWER HERE

Task 1

Use the salt pepper noise image above and reduce noise in it .

- Use Median Filter
- Before starting print the noisy image
- Reduce noise and comment your code appropriately
- Print the image obtained after you apply filter and reducing noise .

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Load the noisy image
img = cv2.imread('Lenna_(salt&pepper).png')

# Apply median filter to reduce noise
median = cv2.medianBlur(img, 5)

# Concatenate the original and filtered images for comparison
compare = np.concatenate((img, median), axis=1)

# Display the images side by side
cv2_imshow(compare)

# Wait for a key press to close the image window
cv2.waitKey(0)

# Close all open windows
cv2.destroyAllWindows()
```



Question 2

In task 1 why is Median filter chosen . What is the use of median filter here ?

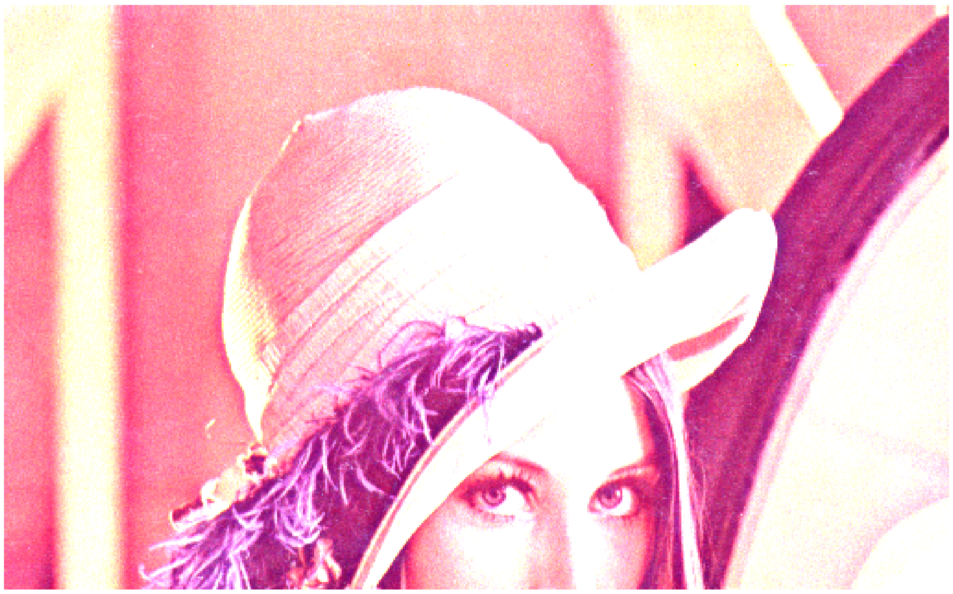
ANSWER HERE

Task 2.

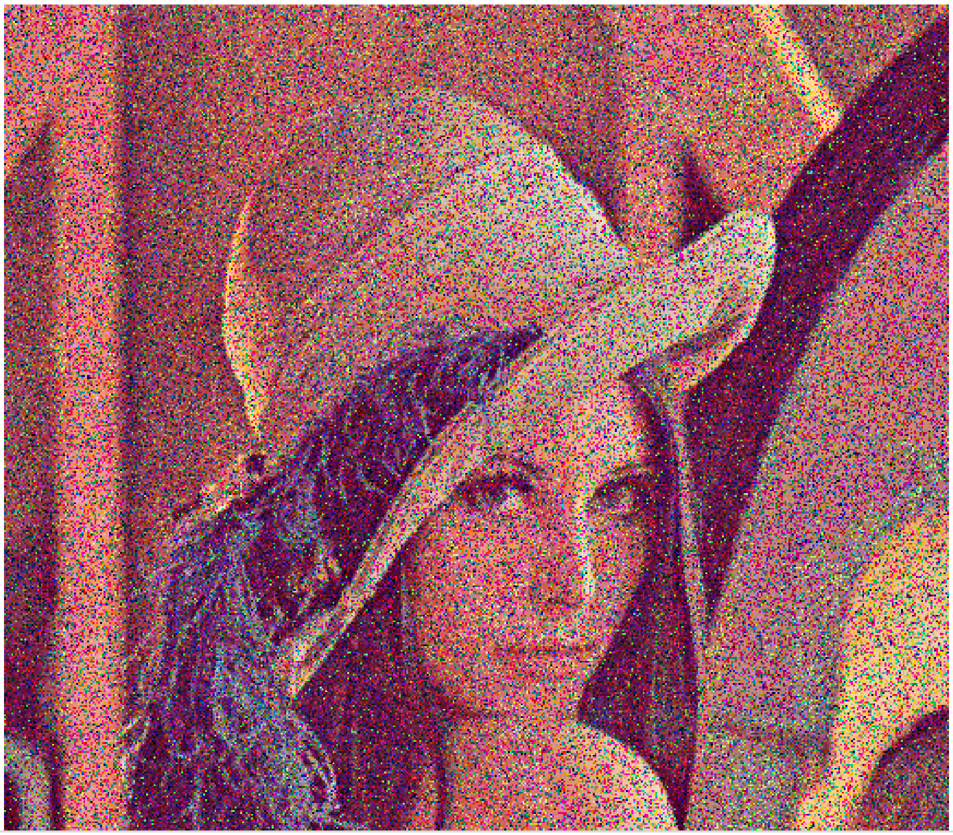
Use the image given above (lenna png) and add Poisson Noise in it

Hint : Use OpenCV-Python

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Load the image
img = cv2.imread('Lenna_(test_image).png')

# Generate Poisson noise
poisson = np.random.poisson(img.astype(np.float32))

# Add the Poisson noise to the image
noisy_img = np.clip(img + poisson, 0, 255).astype(np.uint8)

# Display the noisy image
cv2_imshow(noisy_img)
cv2.waitKey(0)
```



```
# code here

# Load the image
img = cv2.imread('Lenna_(test_image).png')

# Generate guassian noise
gauss = np.random.normal(0,1,img.size)
# Add the guassian noise to the image
gauss = gauss.reshape(img.shape[0],img.shape[1],img.shape[2]).astype('uint8')
noise = img + img * gauss

cv2_imshow(noise)
cv2.waitKey(0)
```



✓ 0s completed at 16:41