
Q1> add the following types of noise: 1. Salt and pepper noise 2. Gaussian noise

Noise: Noise means random disturbance in a signal in a computer version. Here the signal is an image. Random disturbance in the brightness and color of an image is called Image noise.

Salt-and-pepper or Impulse Noise: It is found only in grayscale images (black and white image). An image having salt-and-pepper noise will have a few dark pixels in bright regions and a few bright pixels in dark regions. This degradation caused by sharp and sudden disturbances in the image signal.

Salt and Pepper Noise

- In given Img I observed that it conains white and black dots.
- white dot is a solid noise and it has a pixel value equal to 1 or 255 in an 8bit image.
- Black dot is the pepper noise it has the pixel value equal to 0.
- So, the salt and pepper noise is random number of 1 and 0.
- White= salt =1 or 255
- Black = pepper =0

k represents the number of bits used to represent the intensity values in a digital image, then the range of possible intensity values for that image is [0, $2^k - 1$]

The probability density function (PDF):

$$p(z) = \begin{cases} P_s & \text{for } z = 2^k - 1 \quad \text{salt} \\ P_p & \text{for } z = 0 \quad \text{pepper} \\ 1 - (P_s + P_p) & \text{for } z = V \quad \text{unchanged} \end{cases}$$

where V is any integer value in the range $0 < v < 2^k - 1$
Z=pixel value
K= bit of the image
V=original img. Pixel

Gray scale Image | Noisy Image | Gray scale Image | Noisy Image

Gaussian noise

Gaussian noise is an idealized form of white noise, it is caused by random fluctuations in the signal. A noise must be considered in this formula which can be modelled as an additive function to the convolution formula as follows,

$$g(x, y) = f(x, y) + n(x, y)$$

g = degraded image
f =original img
n-=noise

Random Selection:

n = np.random.normal(loc=mean, scale=sigma, size=(x, y))

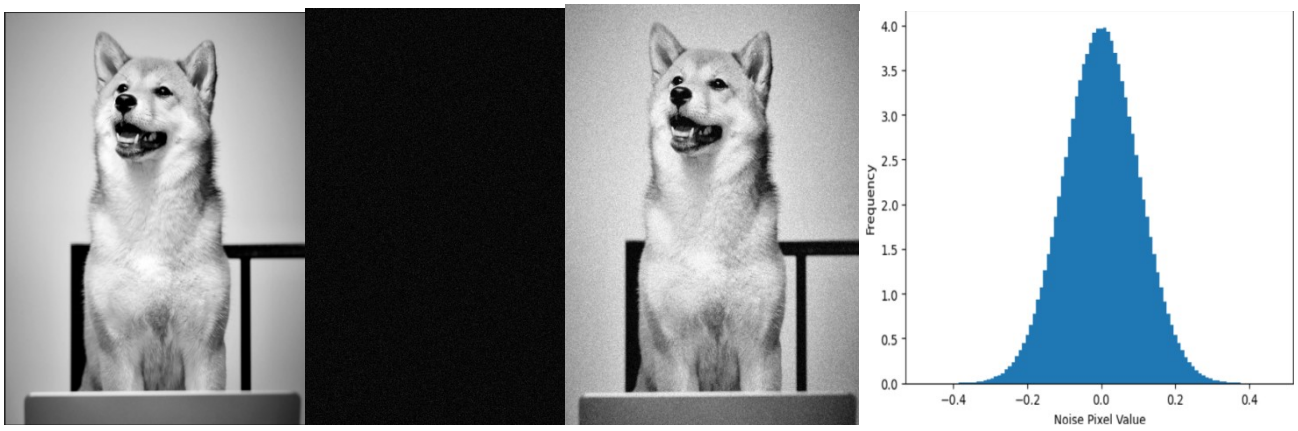
Gaussian formula

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-\bar{z})^2}{2\sigma^2}} \qquad -\infty < z < \infty$$

Img2:

Original image **Adding Noise** **Noisy Image** **Gaussian Distribution Curve**

Img1:



Observations:

- Salt & pepper images contain black and white dots.
- Gaussian noise a white noise, a random signal follows normal distribution.
- In Salt & pepper the effect of individual pixels becoming either completely white (salt) or completely black (pepper) due to errors in data transmission.
- In Gaussian it random values sampled from a Gaussian distribution to each data point in an image or signal.

filtering techniques:

Salt and Pepper Noise is eliminated through image processing technique median filtering, which replaces noisy pixels with the median value of neighboring pixels.

Gaussian Noise can be reduced through filtering technique using mean filtering.

Median filter: We can remove salt and pepper noise by using median filter.

- Process is replacing the value of a pixel by the median of the gray levels in region of that pixel.

$$s(x,y) = \underset{(i,j)\in A_{xy}}{median}\{r(i,j)\}$$

In the Median functionality the OpenCV library is used to clean up a "noisy" image with random black and white specks (salt and pepper noise). It loads the noisy image, applies a special blurring technique called median blur, and then saves the cleaned image.

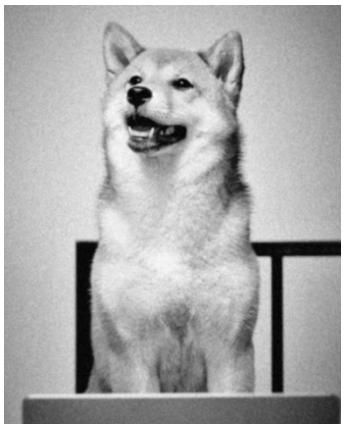
Denoised Image 1

Denoised Image 2

Mean Filter: We can remove Gaussian noise by using a mean filter.

The mean filter is a simple spatial filter that replaces the value of each pixel with the average value of its neighboring pixels within a specified kernel size.

In the Mean functionality OpenCV uses to improve a "noisy" image with unwanted visual disturbances. It loads the noisy image, applies a mean filter (averaging) technique, which smoothens the image by considering nearby pixel values.



Denoised Image 1

Denoised Image 2

References for Q1:

1. From Richard E. Woods' Digital Image Processing, Fourth Edition,
2. Gonzalez, Woods, and Eddins [2009], Digital Image Processing, Fourth Edition
3. https://uomustansiriyah.edu.iq/media/lectures/5/5_2017_03_26!05_29_23_PM.pdf

---

Q2> blur Images using: Gaussian blur & Motion Blur

Image Blurring refers to making the image less clear or distinct. It is done with the help of various low pass filter kernels.

Gaussian Blur: Gaussian blur used in graphics software, typically to reduce image noise and reduce detail.

- In Gaussian Blur operation, the image is convolved with a Gaussian filter instead of the box filter.
- The Gaussian filter is a low-pass filter that removes the high-frequency components are reduced.
- This is done by the function cv.GaussianBlur().

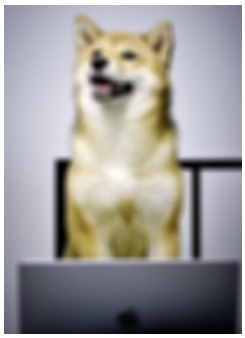
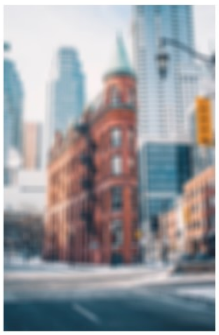
Gaussian kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Functionality:

- Gaussian blur is applied using the cv2.GaussianBlur function.
- **Gaussian Kernel:** k_size specifies the size of the Gaussian kernel. It's defined as (35, 35), which means the kernel has a size of 35x35 pixels. This determines the area over which the kernel operates.
- Adjusting the kernel size allows you to control the radius of the blur effect.
- sigma_x (Standard Deviation in X Direction): horizontal (X) direction.
- sigma_y (Standard Deviation in Y Direction): the vertical (Y) direction.

These parameters allow to fine-tune the Gaussian blur operation to achieve the desired smoothing effect on the image.

gaussian_blur_img1 **gaussian_blur_img2** **gaussian_blur_denoise_img2** **gaussian_blur_denoise_img1**

Observations:

Selecting the Right Kernel Size:

- One of the main challenges in applying Gaussian blur is picking the perfect kernel size. If it's too small, it might not effectively get rid of unwanted noise in the image.

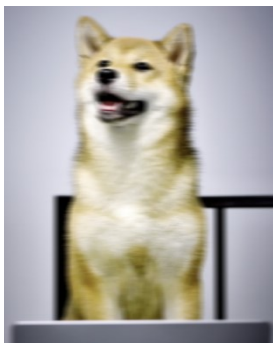
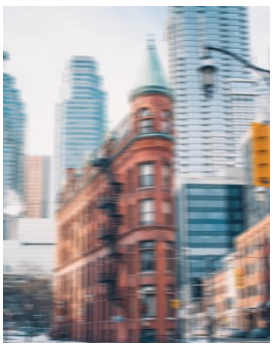
Balancing Detail and Smoothing:

- Gaussian blur does a great job at smoothing out images, but it comes with a trade-off. It tends to make fine details in the image less pronounced. This can be a bit tricky, especially when you're working on tasks like image recognition.

Challenges: Wiener Deconvolution

- Determining the exact level of noise in the image can be tricky because it varies depending on the specific noise characteristics.
- To use the Wiener deconvolution effectively, we need to find the right settings for the filter, like its size and shape. This often involves some trial and error.
- If the noise in the image doesn't follow a standard Gaussian pattern or if the image has complex issues, our denoising approach might not work as well. In such cases, we might need more advanced techniques or initial fixes to improve the results.

Motion Blur:

motion blur img1 **motion blur img2** **motion blur denoise img1** **motion blur denoise img2**

Functionality:

- **Image Loading:** The code starts by loading an image from a specified location.
- **Creating Blur Direction:** It simulates a horizontal motion blur effect, as if objects in the image are moving from left to right.
- **Kernel Setup:** A "kernel" is like a brush that defines the blur effect. The code sets up a kernel, making it wider based on a parameter called **kernel_size**.
- **Saving the Result:** The final blurred image is saved to a new file.

Challenges Faced:

- Precise kernel size selection is essential for achieving optimal blur strength, as a small size may result in imperceptible blurring, while a large size can lead to excessive distortion.
- Extending the code's capability to handle diverse blur directions and managing unintended visual artifacts requires meticulous parameter adjustments, akin to fine-tuning an artistic process.

Wiener Deconvolution for Deblur:

Wiener Deconvolution is a mathematical technique used for the restoration or deblurring of images that have been blurred by a known linear system. It is particularly useful when you have knowledge of the blur kernel (the function that describes how the image has been blurred) and when the image is corrupted by additive noise.

Deconvolution: The code applies the Wiener filter in the frequency domain to the blurred image (G) to perform the deconvolution. This step essentially reverses the blurring process and attempts to recover the original image.

- Loading the Input Image
- Convert Gray Scale
- Defining Deblurring kernel : It defines the deblurring kernel (kernel_h) assuming horizontal motion blur. It creates a 2D array representing the kernel, setting the middle row to ones and normalizing the kernel.
- Compute the wiener filter: involves applying Fourier transforms to the kernel (H) and the image (G), and then using these transforms to compute the filter.
- Performing Deconvolution and clip pixel values
- Saving Deblurred image

References:

- https://www.geeksforgeeks.org/opencv-motion-blur-in-python/
- https://www.adeveloperdiary.com/data-science/computer-vision/applying-gaussian-smoothing-to-an-image-using-python-from-scratch/
- **Gonzalez, R.C., Woods, R.E., and Eddins, S.L.** "Digital Image Processing." Chapter 11: Image Restoration and Reconstruction. Pearson, 2018.

Q3> Take the image of a dog and place it on the road in the city image.

Steps I followed placing dog Image on Road:

1: Remove BG of Dog

- Importing PIL, rembg(to remove background) Libraries.
- Open the Dog image using PIL Library to manipulate it.
- Remove Bg using rembg.
- Converting the Image to RGB mode using convert method and saving in JPEG.

2: Making Dog as Transparent:

- Load BG Removed Image.
- Converting the image to the RGBA color mode. By using Alpha Transparency.
- Iterating through each pixel in the image's pixel data. If the pixel is not black you keep it unchanged. However, if the pixel is black, you make it transparent by setting its alpha channel to 0.

3: Pasting Dog Image on Road

- I Used width and Height Dimensions are 200,200.
- Rotate the dog image by 360 degrees.
- Paste the dog image onto the road image at the specified position (X,Y).


Original Image


Bg Image


Transparency Image


Dog on Road

Challenges Faced:

- Removing the background from the dog imagecan be challenging, especially if the background has complex patterns or colors.
- Scaling the dog image to an appropriate size and positioning it correctly on the road image can be challenging.
- Precisely positioning the dog on the road image requires adjusting the x_position and y_position values.

References:

- https://www.geeksforgeeks.org/how-to-remove-the-background-from-an-image-using-python/
- https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html