Filters 2

What is noise?

Image noise is random (usually unwanted) variation of brightness or colour information in the images captured. It is degradation in image signal and may be caused by several factors, e.g. by long exposures because of the heat produced by the image sensor in the camera.

Noise Type 1: Salt and Pepper

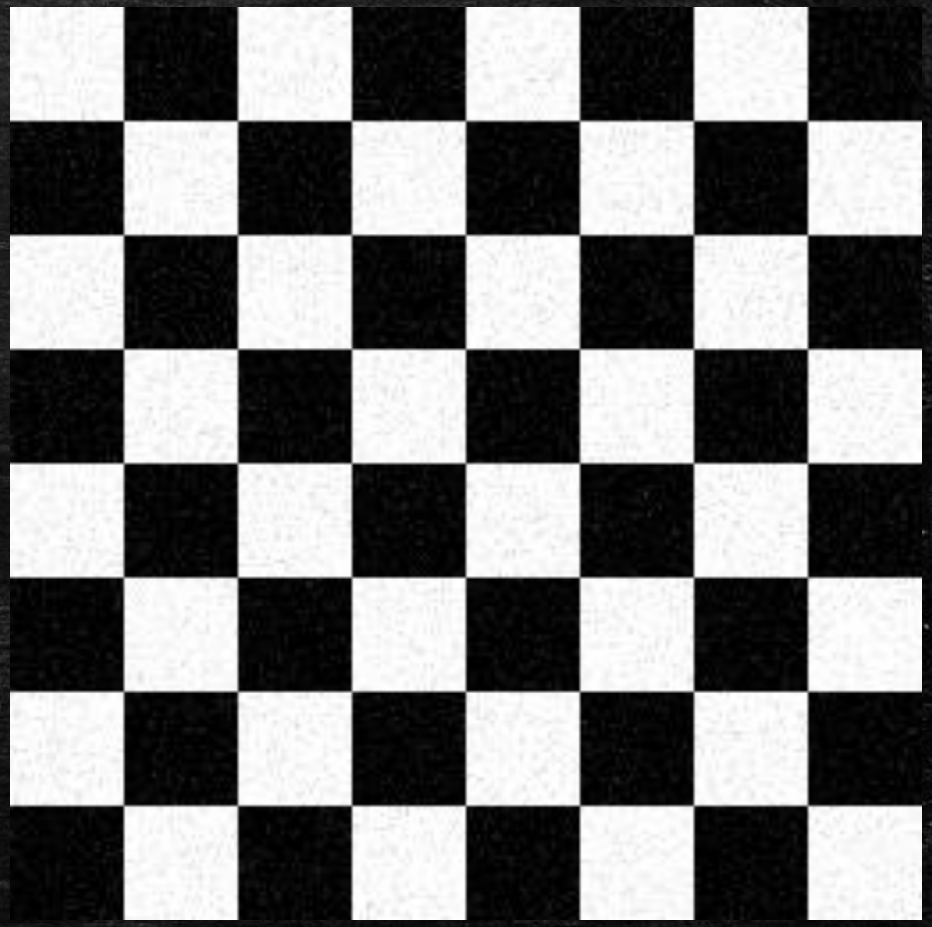
The effect is similar to sprinkling white and black dots – salt and pepper – all over the image.

One example where salt and pepper noise arises is in transmitting images over noisy digital links.

Noise Type 2: Gaussian Noise

This type of noise adds more noise to the midtones and less noise to the shadows and highlight regions of the image.

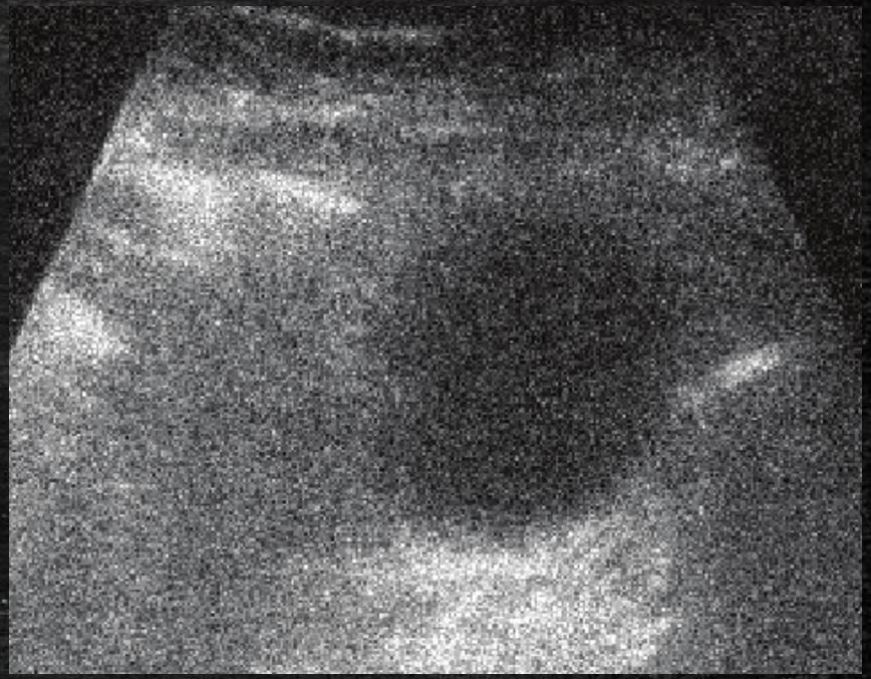
Principal sources of **Gaussian noise** in digital images arise during acquisition e.g. sensor **noise caused** by poor illumination and/or high temperature



Noise Type 3: Speckle noise

Random values multiplied by pixel values in an image.

Very evident in medical ultrasound images. Images obtained by such imaging system suffer from an interference phenomenon.



Generating Noise

Use an image and try to generate random noise. Display next to each other to better appreciate the differences. The code we need is:

```
cv2.randn(im,mean,sigma) # create the random distribution
im_noise = cv2.add(img, im) # add the noise to the original image
```

Solutions?

Image Restoration – how do we restore the image? i.e. how do we remove the noise?

Usually we know what type of errors to expect and the type of noise on the image; hence, we can choose the most appropriate method for reducing the effects

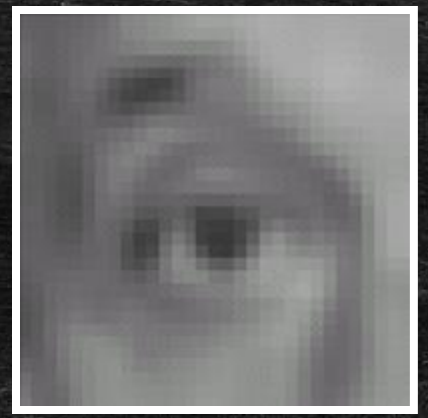
Most of the time we will be dealing with Filters in order to perform such operations. Common filters are the Mean, Gaussian and Median smoothing.

Mean filter (average)

Already covered…

Original

$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

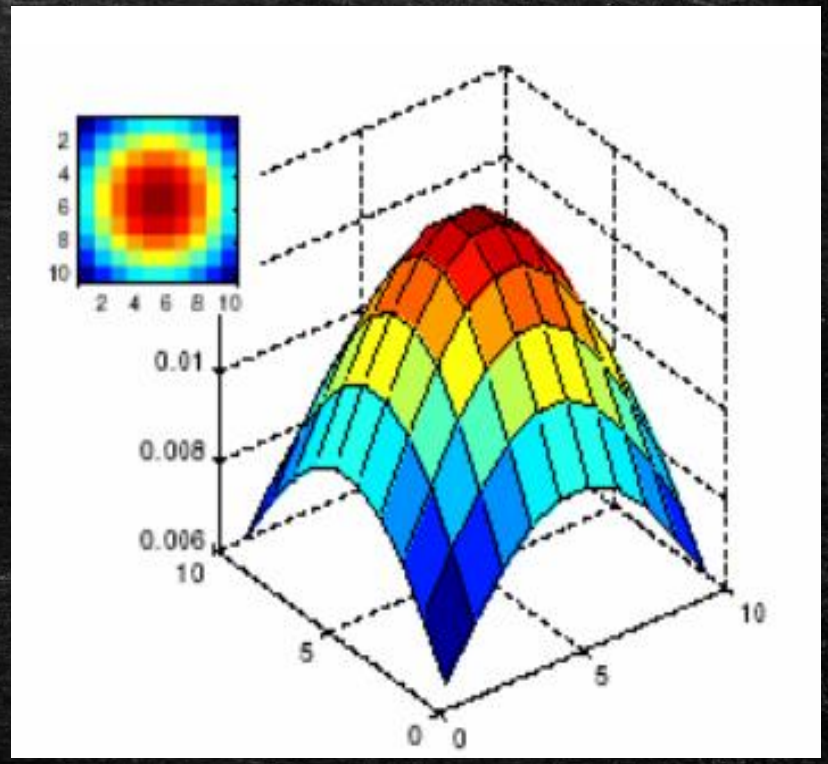


Blur (with a mean filter)

Guassian filter

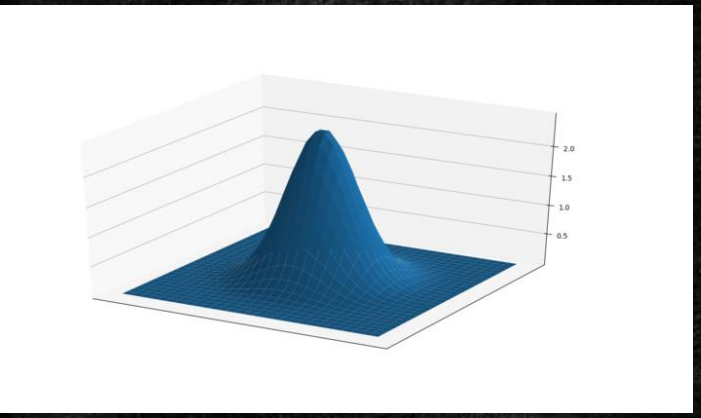
Already covered…

```
Gaussian_kernel = 1/16 [1,2,1;
                        2,4,2;
                        1,2,1]
```



Gaussian blur

- In a **Gaussian blur**, the pixels nearest the center of the kernel are given more weight than those far away from the center.
- Larger kernels have more values factored into the average, and this implies that a larger kernel will blur the image more than a smaller kernel.
- Sigma (StDev) defines the variance or ‘decay’ of the gaussian curve. It peaks at the center and decays as you move away. The bigger the sigma the slower the decay.



Python OpenCV built-in gaussian blur function

```
KernelSize = (15, 15)
GaussianBlurImg = cv2.GaussianBlur(src=im_noise, ksize=KernelSize, sigmaX=0, sigmaY=0)
```

Where :

src- image to blur.

ksize – Kernel size (5x5, 15x15). Has to be in odd numbers.

sigmaX and sigmY – Standard deviation for decay of curve.
Larger value = slower decay

....

Mean and Gaussian Filters

- These work best with Salt & Pepper Noise, and Gaussian noise
- BUT they also tend to:
  - Blur sharp edges, and
  - Destroy the very fine details in the image

  They are called **Linear filters** (can be done by sliding the kernel)

Non-Linear Filters

- Advantage over linear filters
- Can preserve edges
- Can be difficult to design, but thankfully OpenCV provides ready in-built methods e.g. Bilateral filter

Median Filters

- A relatively simple non-linear filter
- The **median filter** considers each pixel in the image in turn and looks at its nearby neighbours to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the mean of neighbouring pixel values, it replaces it with the **median** (middle) of those values.

Bilateral filter

- A **bilateral filter** is a non-linear, edge-preserving, and noise-reducing smoothing **filter** for images. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. This weight can be based on a Gaussian distribution.
- **cv2.bilateralFilter()**, is highly effective at noise removal while preserving edges.

Worksheet Intro

Use an image with salt & pepper noise and apply the 3 different types of noise removal techniques we learnt about.

Display the results labelled and in a single figure, to enable comparison.

Questions?