Computer Vision: Task 1 Report

1- Add noise to the image:

-Gaussian Noise:

Gaussian noise, also known as additive white Gaussian noise (AWGN), is a type of random noise with a probability density function that follows a Gaussian distribution.it is characterized by its mean (μ) and standard deviation (σ).

$$\varphi(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(z-\mu)^2/(2\sigma^2)}$$

- parameters:

- zis the value of the random variable.
- μ is the mean of the Gaussian distribution.
- σ is the standard deviation of the Gaussian distribution.

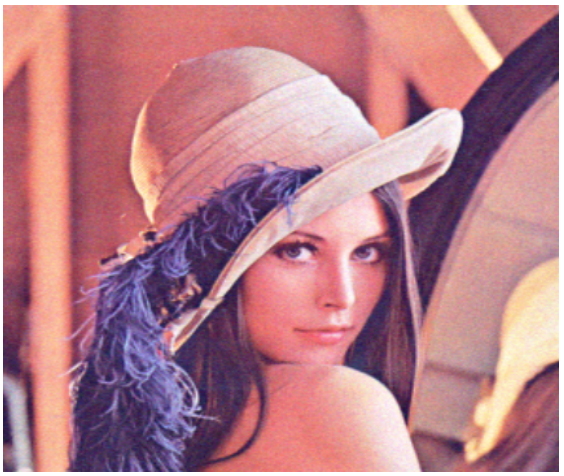
-Uniform Noise:

Uniform noise, also known as flat noise, is a type of random noise that is uniformly distributed over a specified range.and it's probability density function is constant within a specified range.

$$p(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \le z \le b \\ 0 & \text{otherwise} \end{cases}$$

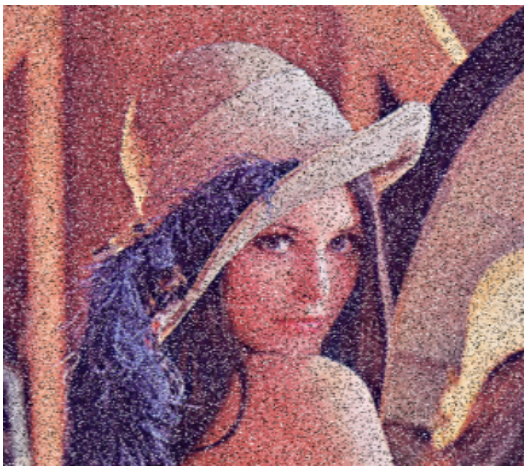
- parameters:

- z is the value of the random variable.
- a is the lower bound of the uniform distribution.
- b is the upper bound of the uniform distribution.

-Salt and Pepper Noise:
is a random noise that randomly replaces some pixels in an image with either the maximum intensity value (salt) or the minimum intensity value (pepper).

- There is no specific mathematical formula for salt and pepper noise since it is a descriptive term used to represent the presence of random isolated white and black pixels in an image

2- Frequency domain filters :

-Low-pass:
low-pass filter attenuates or removes high-frequency components in the image, allowing only the low-frequency components to pass through. This filtering
operation helps in reducing noise and smoothing the image and blurs the image.

$$D(u,v)=\sqrt{\left(u-\frac{P}{2}\right)^2+\left(v-\frac{Q}{2}\right)^2}$$

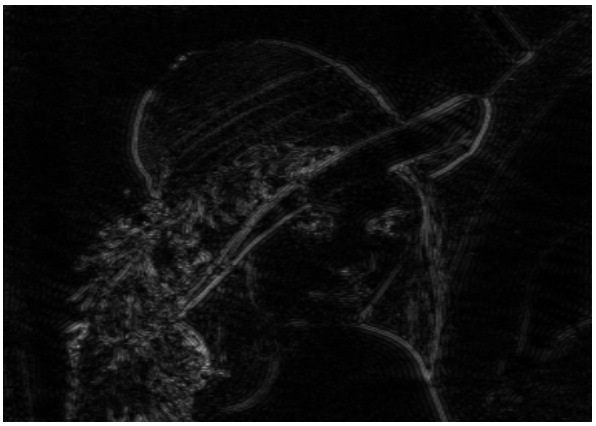
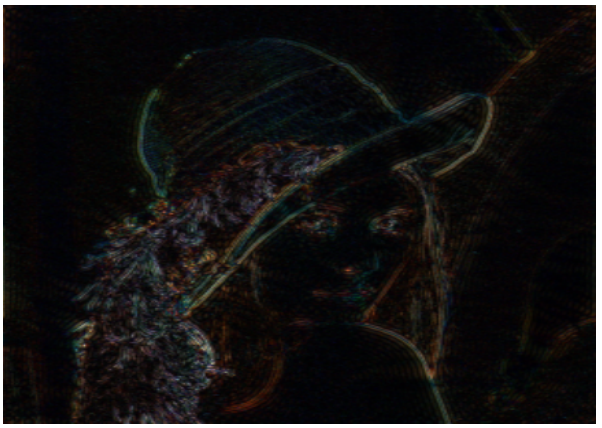
- parameters :

- D (u, v) represents the frequency response of the filter at spatial frequency (u, v).
- (u, v) are the coordinates in the frequency domain.
- P and Q are the dimensions of the input image.
- D0 is the cutoff frequency that determines the circular region in the frequency domain below which the filter allows frequencies to pass. Frequencies outside this region are attenuated (set to zero).


-High-pass:

High-pass filter attenuates or removes low h-frequency components in the image, allowing only the high -frequency components to pass through.it enhances sharp detail, but cause a reduction in contrast in the image..

$$H(u,v)=1 - D(u,v)$$

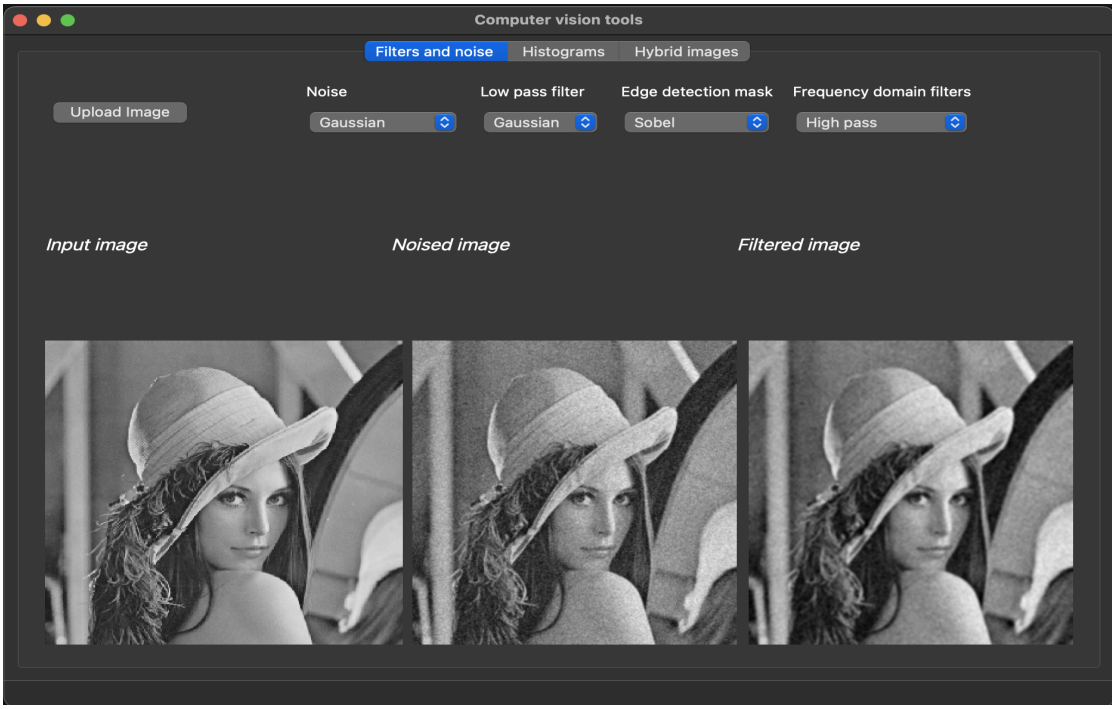
3 - Filter the noisy image using the following low-pass filter:

-Gaussian

Gaussian Filter is used in reducing noise in the image and also the details of the image

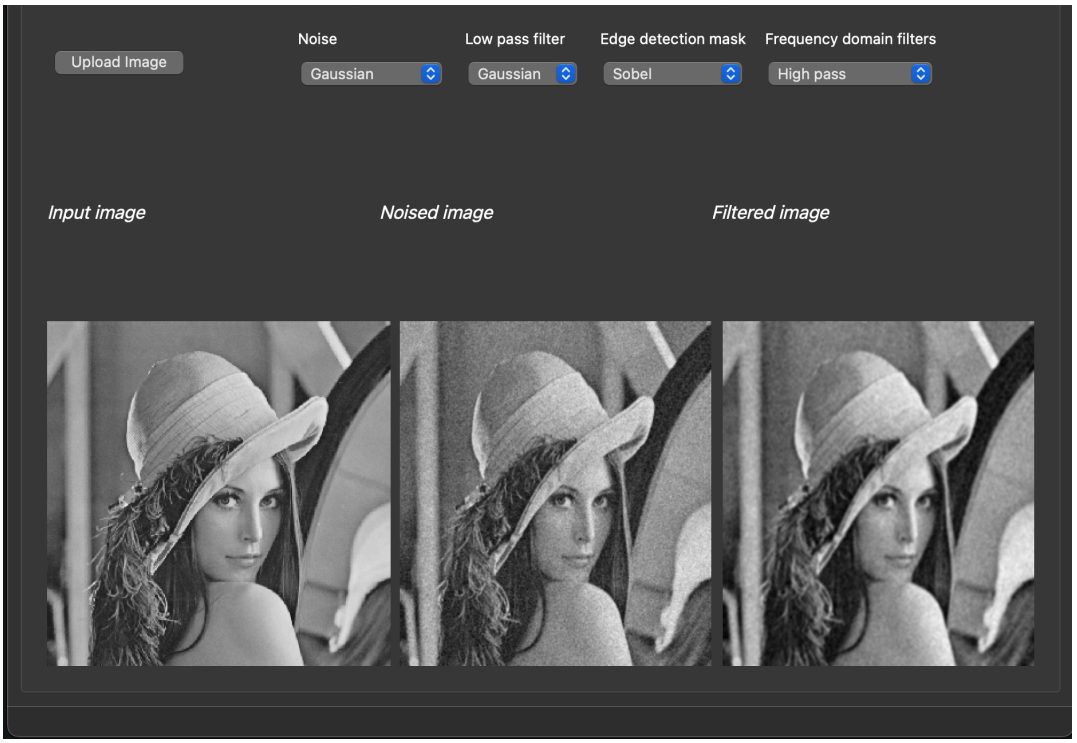
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

1. Defining the convolution function which iterates over the image based on the kernel size(Gaussian filter)
2. Defining the Gaussian function based on the size of sigma(standard deviation).



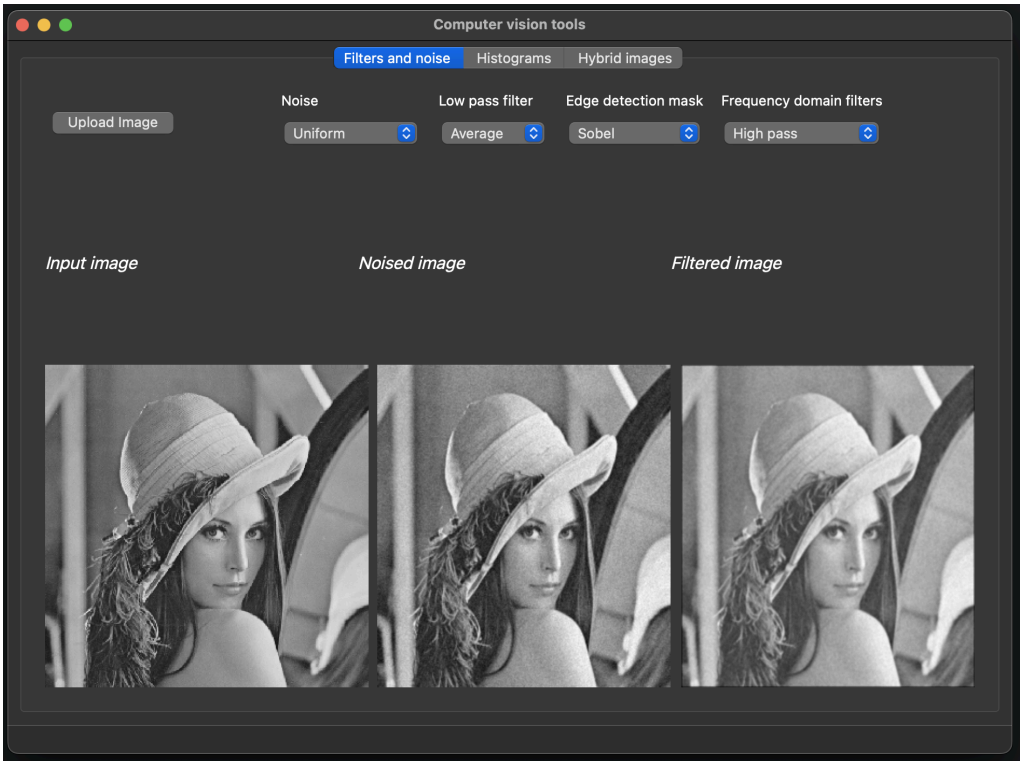
sigma = 5 and filter_size = 5

compare with built in function: `cv2.GaussianBlur(image, (5, 5), 5)`



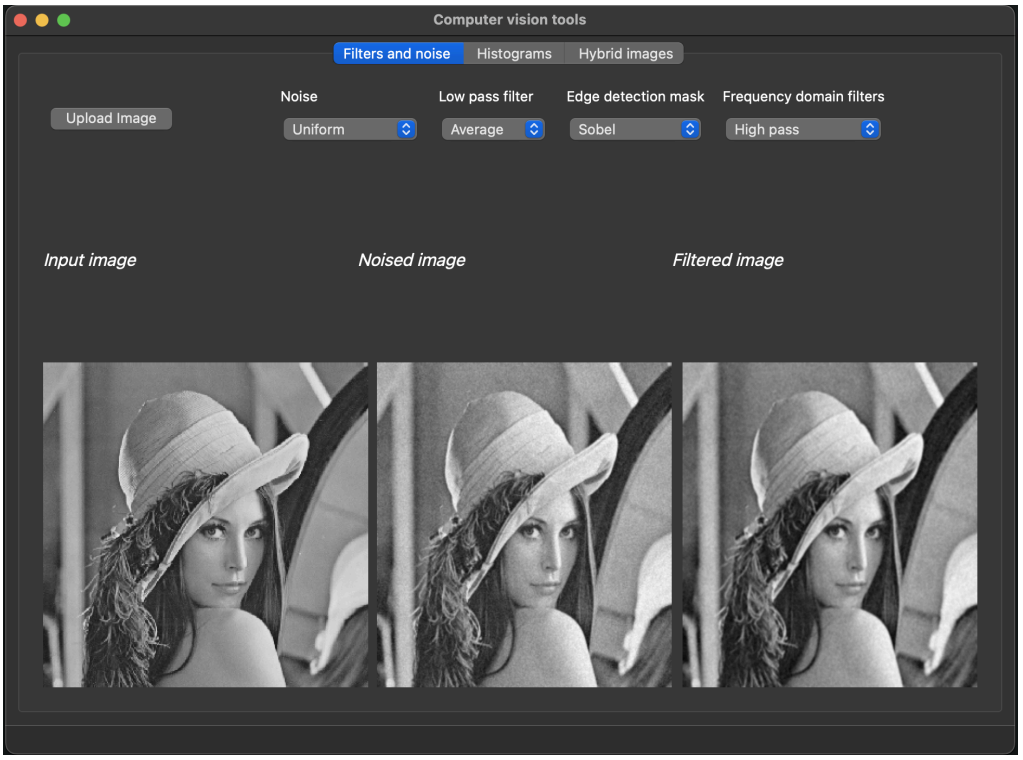
-Average

The mean filter is a simple sliding-window spatial filter that replaces the center value in the window with the average (mean) of all the pixel values in the window.



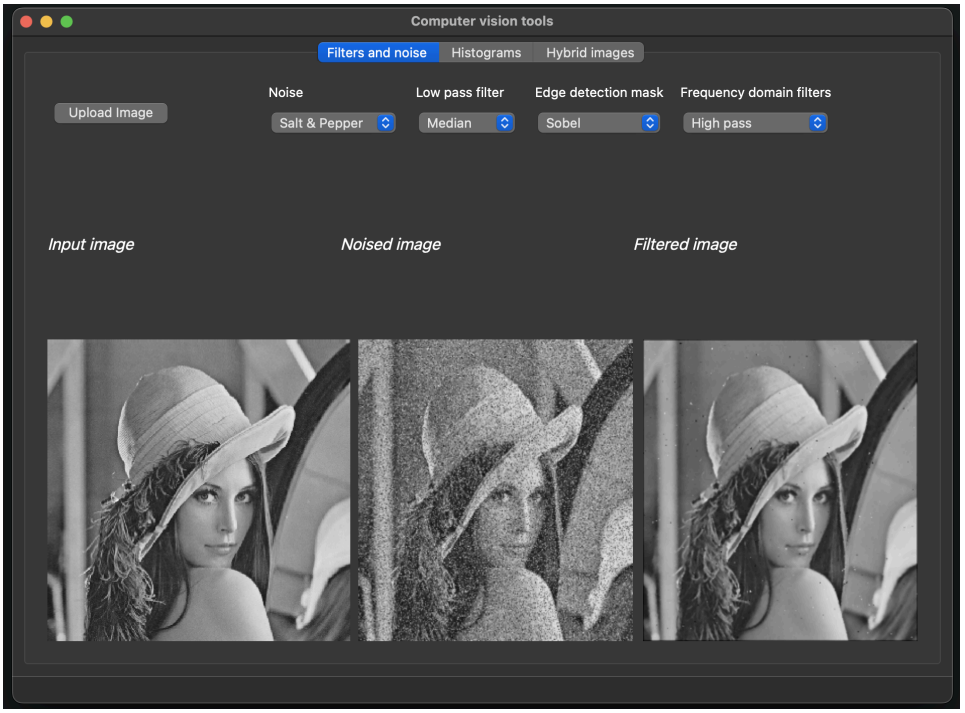
kernel size=3

compare with built in function : `cv2.blur(image, (3, 3))`



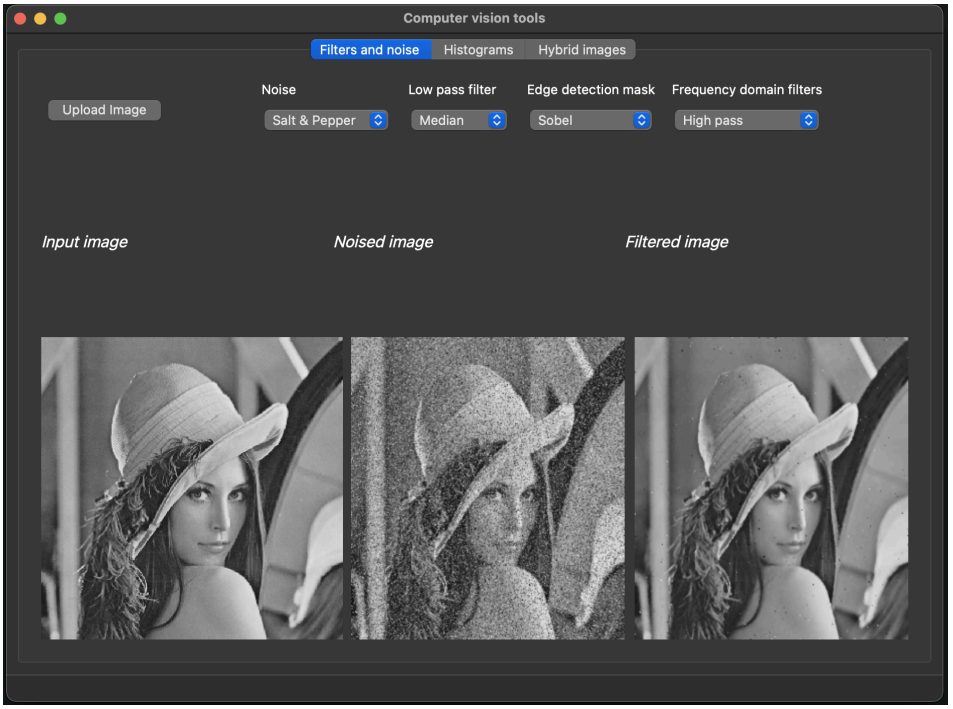
-Median .

The median filter is also a sliding-window spatial filter, but it replaces the center value in the window with the median of all the pixel values in the window. good for salt and pepper noise.



kernel size=3

compare with built in function : `cv2.medianBlur(image, 3)`



4 - Detect edges in the image using the following masks:

-Sobel

The Sobel operator is a type of gradient-based edge detection technique. It works by convolving the image with a small kernel that calculates the gradient at each pixel.

The gradient is then used to determine the edges in the image by looking for places where the gradient magnitude is high.

-Roberts

The Roberts operator is another type of gradient-based edge detection technique. It works by calculating the gradient in two diagonal directions using a small kernel.

-Prewitt

The Prewitt operator is similar to the Sobel operator, but it uses a slightly different kernel to calculate the gradient.

-Canny

The **Canny edge detector** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images.
The Canny edge detection algorithm is composed of 5 steps:

1. Noise reduction;
2. Gradient calculation;
3. Non-maximum suppression;
4. Double threshold;
5. Edge Tracking by Hysteresis.

This algorithm is based on grayscale pictures. Therefore, the prerequisite is to convert the image to grayscale before following the above-mentioned steps.

5 - Noise Reduction:

One way to get rid of the noise on the image is by applying Gaussian blur to smooth it.

Gradient Calculation

The Gradient calculation step detects the edge intensity and direction by calculating the gradient of the image using edge detection operators. Edges correspond to a change of pixels' intensity. To detect it, the easiest way is to apply filters that highlight this intensity change in both directions: horizontal (x) and vertical (y)

When the image is smoothed, the derivatives ***Ix*** and ***Iy*** w.r.t. ***x*** and ***y*** are calculated. It can be implemented by convolving ***I*** with Sobel kernels

Non-Maximum Suppression

Ideally, the final image should have thin edges. Thus, we must perform non-maximum suppression to thin out the edges.

Double threshold

- High threshold is used to identify the strong pixels (intensity higher than the high threshold)
- Low threshold is used to identify the non-relevant pixels (intensity lower than the low threshold)
- All pixels having intensity between both thresholds are flagged as weak and the Hysteresis mechanism (next step) will help us identify the ones that could be considered as strong and the ones that are considered as non-relevant.

Edge Tracking by Hysteresis

Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one

6- Normalizing the image

Normalizing an image is a common preprocessing step in image processing and machine learning tasks. It involves transforming the pixel values of an image to a common scale, usually between 0 and 1 or -1 and 1. This process is beneficial for several reasons:

1. Consistent Scale: Normalizing ensures that pixel values are on a consistent scale, making it easier for algorithms to learn patterns across different images.

2. Numerical Stability: It helps in stabilizing the learning process by preventing large input values from dominating the learning process, especially in deep learning models.

3. Improved Convergence: Normalizing input data often results in faster convergence during training, as it helps in avoiding saturation of activation functions and gradients.

4. Effective Regularization: Normalization can act as a form of regularization, preventing overfitting by constraining the input data within a specific range.

Before Normalization

After Normalization

7- Global and Local Thresholding

7.1 Global Thresholding:

- In global thresholding, a single threshold value is applied to the entire image to separate pixels into foreground (object) and background regions.
- The threshold value is determined based on characteristics of the entire image, such as histogram analysis or taking the mean of the whole image (mean in used)
- Global thresholding is simple and efficient, especially when the image has a clear distinction between foreground and background regions.

Original Image

Global Thresholding

7.2 Local Thresholding:

- In local thresholding, different threshold values are applied to different regions of the image, allowing for adaptive segmentation based on local pixel intensity variations.
- Local thresholding methods typically involve dividing the image into smaller subregions or windows (kernel) and applying a thresholding algorithm independently to each window. The kernel size is directly related to the local threshold process but inversely related to the computation time.
- Local thresholding is more robust to variations in illumination and contrast across different parts of the image, making it suitable for images with uneven lighting conditions or complex backgrounds.

- The local threshold output image depends on the size of the window (Kernel) chosen, here are some examples of the global thresholding output image using different kernels.

Kernel = 3 pixels

Kernel = 5 pixels

Kernel = 11 pixels

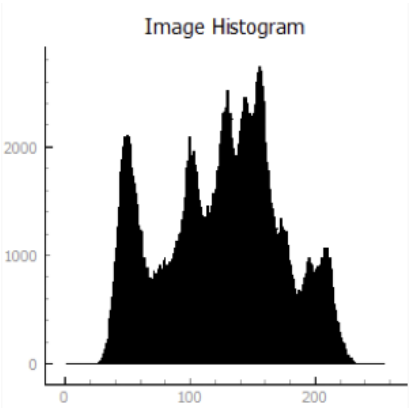
8- Histogram

A histogram is a tool in image processing. It is a graphical representation of the distribution of data. An image histogram gives a graphical representation of the distribution of pixel intensities in a digital image.

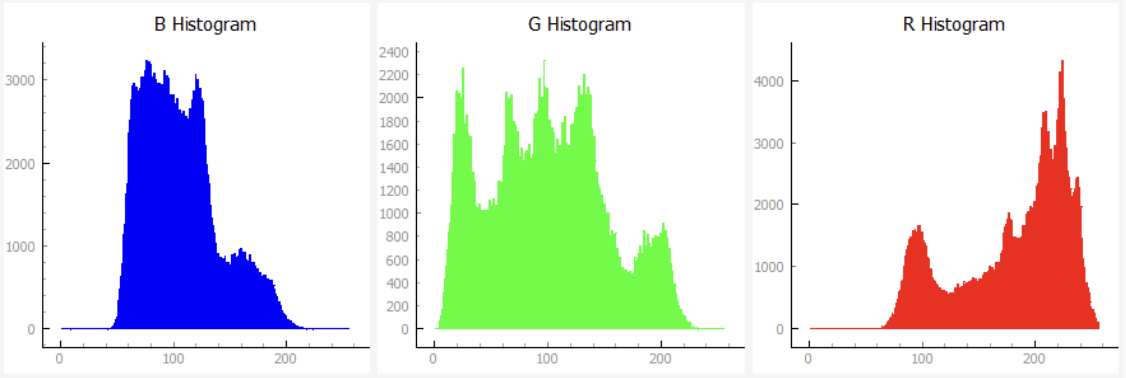
The x-axis indicates the range of values the variable can take. This range can be divided into a series of intervals called bins. The y-axis shows the count of how many values fall within that interval or bin.

When plotting the histogram, we have the pixel intensity in the X-axis and the frequency in the Y-axis.

- We calculated the histogram for the gray-scale image as we have a single channel



- We calculated the histogram for the colored image as we have three channels (B, G, R)

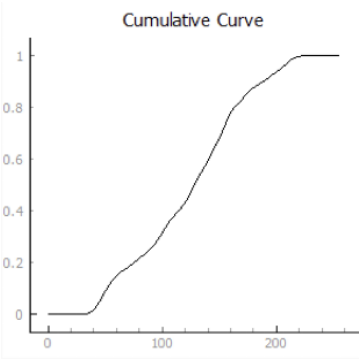


Finally, we can compare between the histograms of the input image and the edited image that can be normalized, equalized or any other edit.

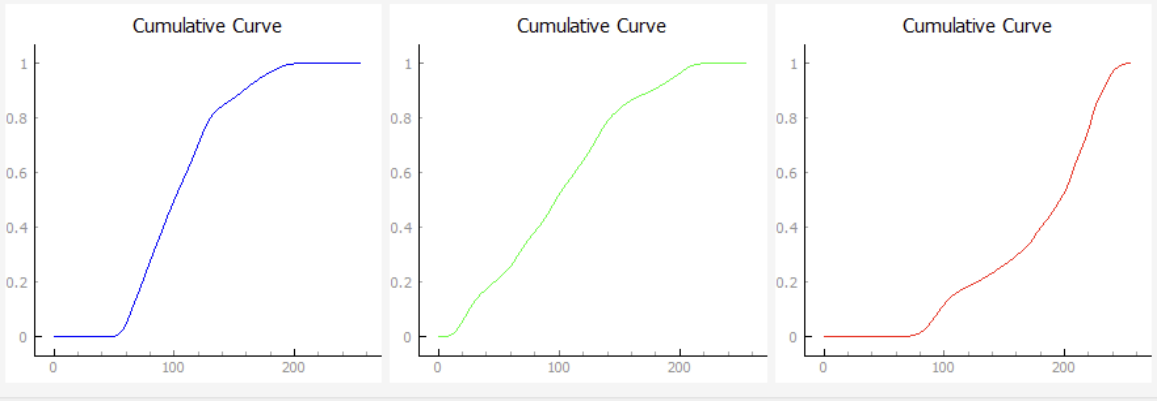
Cumulative Distribution Function

The cumulative distribution function (CDF) of an image is a function that represents the cumulative probability distribution of the pixel intensities in the image. The CDF is a plot that shows how many pixels have intensities less than or equal to a given value. It ranges from 0 to 1, where 0 represents the lowest intensity and 1 represents the highest intensity in the image.

- For the gray-scale image



- For colored image (B, G, R)

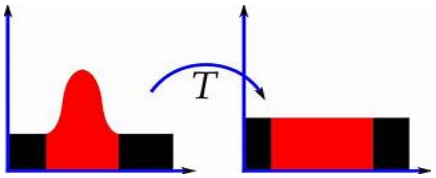


9- Histogram Equalization

Histogram Equalization is an image processing technique that is used to improve contrast in images. It accomplishes this by spreading out the most frequent (spiking) intensity values, i.e. stretching out the intensity range of the image to be more "uniform". Histogram Equalization is particularly useful in images where their backgrounds and foregrounds share similar intensity values thus improving the image's overall contrast. Of course, depending on the image, histogram equalization can yield undesirable and unrealistic effects to the image, but it does have its applications.

The general histogram equalization formula is:

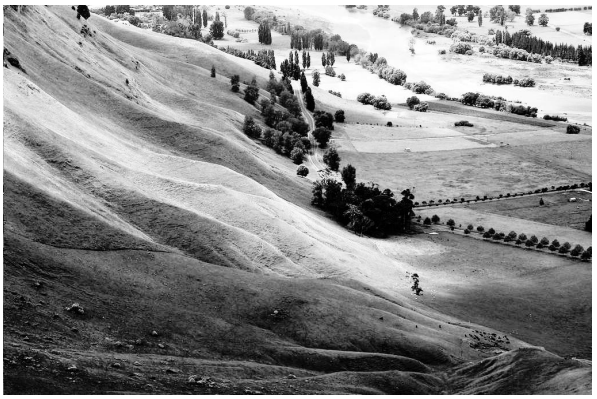
$$h(v) = \text{round}\left(\frac{\text{cdf}(v) - \text{cdf}_{\text{min}}}{(M \times N) - \text{cdf}_{\text{min}}} \times (L-1)\right)$$



where cdf_{min} is the minimum non-zero value of the cumulative distribution function, M × N gives the total number of pixels in the image (where M is width and N is height), and L is the number of grey levels used (in most cases, for an 8-bit image, it is 256).

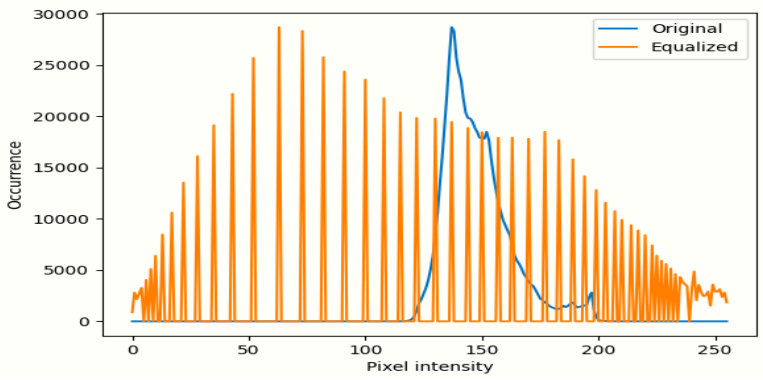
Following this formula, we can now equalize images. Using the image on the left, we apply histogram equalization to it and it produces the image on the right. As shown below, the contrast of the image greatly improves whereby the details that were obscured in the image prior to equalization are now more visible in the equalized image.

Before Equalization

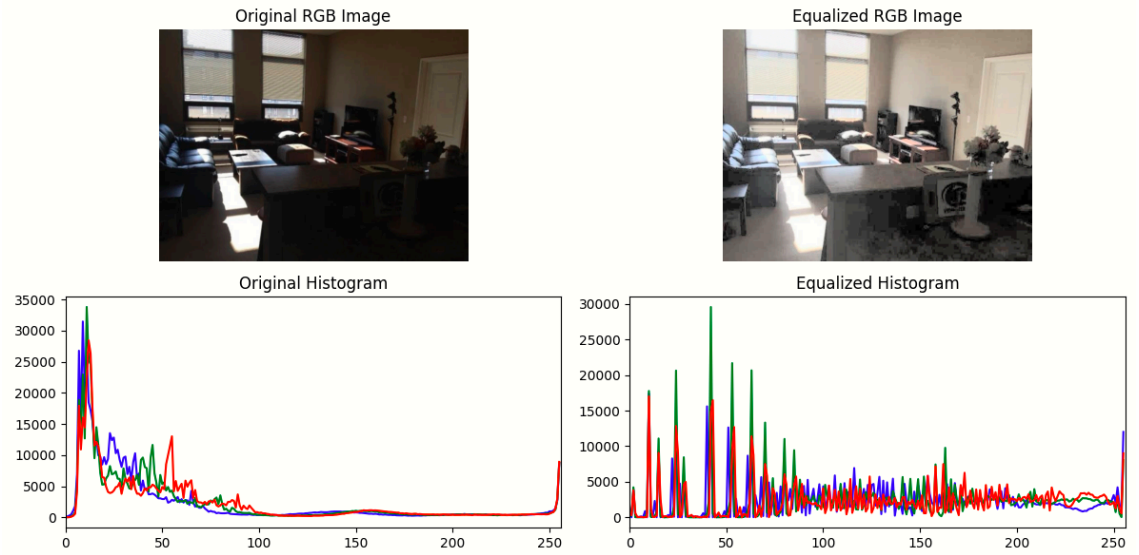


After Equalization

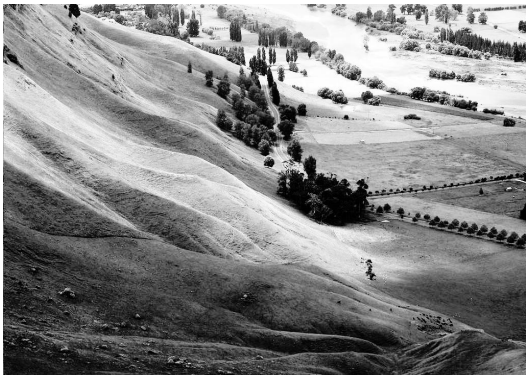
Below are the images' histograms for comparison. As shown, the image's histogram became more "uniform" and flat compared to the original's limited grey level range.



Histogram Equalization can also be applied to not only grayscale images (as seen above) but also to colored images. The process, however, requires a few alterations as most images exist in the RGB color space which does not have an intensity channel. Therefore, it is necessary to first convert the image to another color space containing an intensity or "luma" channel, e.g. HSV, LAB, or YCrCb color spaces, then equalizing that channel before converting the image back into RGB. Below, we have equalized a colored image by equalizing its Y-channel after converting it into a YCrCb image, and we showcase the differences in the RGB channels by plotting each channel's histogram:



Comparing to OpenCV `equalizeHist()`



Equalized Grayscale Image

Equalized RGB Image

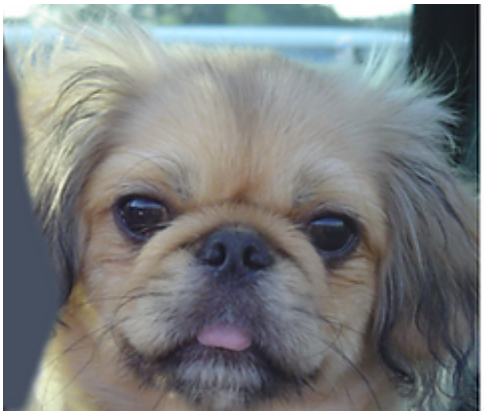
In comparison to our equalization, the results are very similar with little to no differences.

10- Hybrid Images

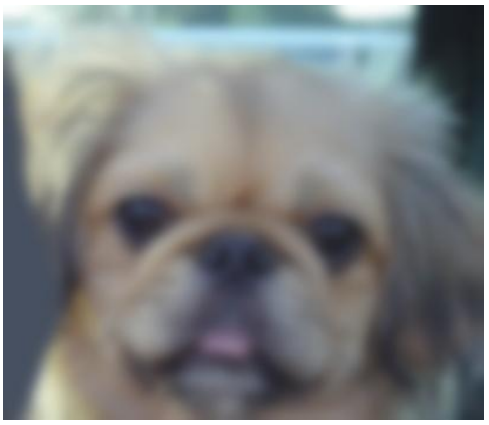
Hybrid images are a form of optical illusion that can be perceived in one of two different ways, depending on the viewing distance. The optical illusion is based on the way humans process visual input. Hybrid images are created by combining the low-pass components of one image with the high-pass components of another image, producing an image with an interpretation that changes with viewing distance. From a close viewing distance, the high-pass image dominates, and from a farther viewing distance, the low-pass image dominates.

Take, for example, the two images of the cat and dog below. First, we retrieve the low spatial frequencies from the image of the dog by applying our designed gaussian blur filter (with a sigma of about 5). Next, we retrieve the high spatial frequencies from the image of the cat by subtracting the low spatial frequencies of the image from the original image. Afterwards, the two filtered images are superimposed to create the final hybrid image by adding the two filtered images together.

Original images of the cat and dog:



Filtered images:



High-Pass Image

Low-Pass Image

The final hybrid image with hybrid image visualization to aid in the perception of the optical illusion:

Of course, to create a good illusion, it is best to use different sigma values for each image and to choose the images from which the low-pass/high-pass components are taken from. Additionally, the images have to be aligned to sell the illusion well. One such example of an unrealistic, unconvincing combination is taking the low-pass components of the cat image and the high-pass components of the dog image. This creates an unrealistic feel to the image due to the low-pass image's majority contribution of color to the image. The result is shown below.

Another example of a good hybrid image includes the combination of images of a bird and plane: