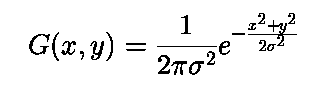
Complete the tasks using python and when submitting the answers include the code you wrote, relevant output images and word processed answers to the questions.

1. **Using the given equation for a 2D Gaussian filter, plot the output of a filter-convolved image with a sigma, σ, value of 10 [2]. What type of filter does a high sigma value produce [1]? Can you think of a way of extracting high frequency image components using only a low frequency filtered image [1]? Plot such an image and comment on it’s quality [1].**



*Hint: Use PIL.Image.open to import your image, and from scipy.signal use the convolve2d with mode=’same’ for the convolution.*

*[2] Plotting the various blurred images*

*[1] Low-pass filter.*

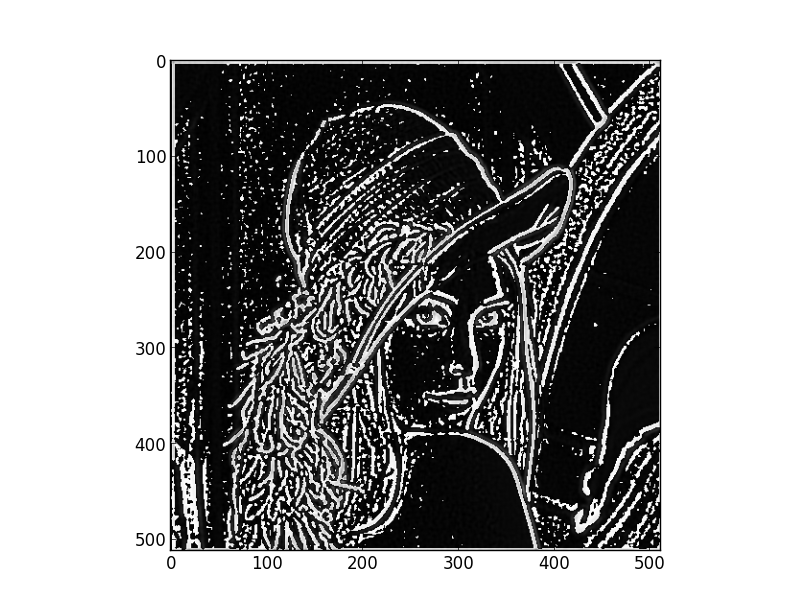
*[1] Subtract the low-pass filtered image from the original image.*

*[1] Roughly an edge-detected image plotted (far-right above plot).*

1. **Create a Difference of Gaussian (DOG) filter by applying two different sigma valued Gaussian filters to the original image and then subtracting the two outputs from each other [2]. Try to find optimum sigma values for edge detection [1]. Comment on the difference between this filter and the first [2].**

*Hint: Larger sigma values require larger Gaussian kernels.*

*[2] Produce a DOG filter and plot the output*



*[1] Optimum filter sizes have low sigma values e.g. 1 & 2. Must be different, low values.*

*Any 2:*

*[1] Higher quality filter*

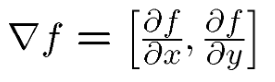
*[1] Less image noise*

*[1] More easily controlled*

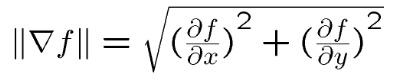
1. **Oriented Difference of Gaussian (ODOG) filters can be created by using the below equation. Where [a, b] is a directional vector of x and y respectively for defining the direction of the filter. Plot a horizontally oriented difference of Gaussian filtered image as well as a vertically oriented difference of Gaussian filtered image.**

Add output plots.

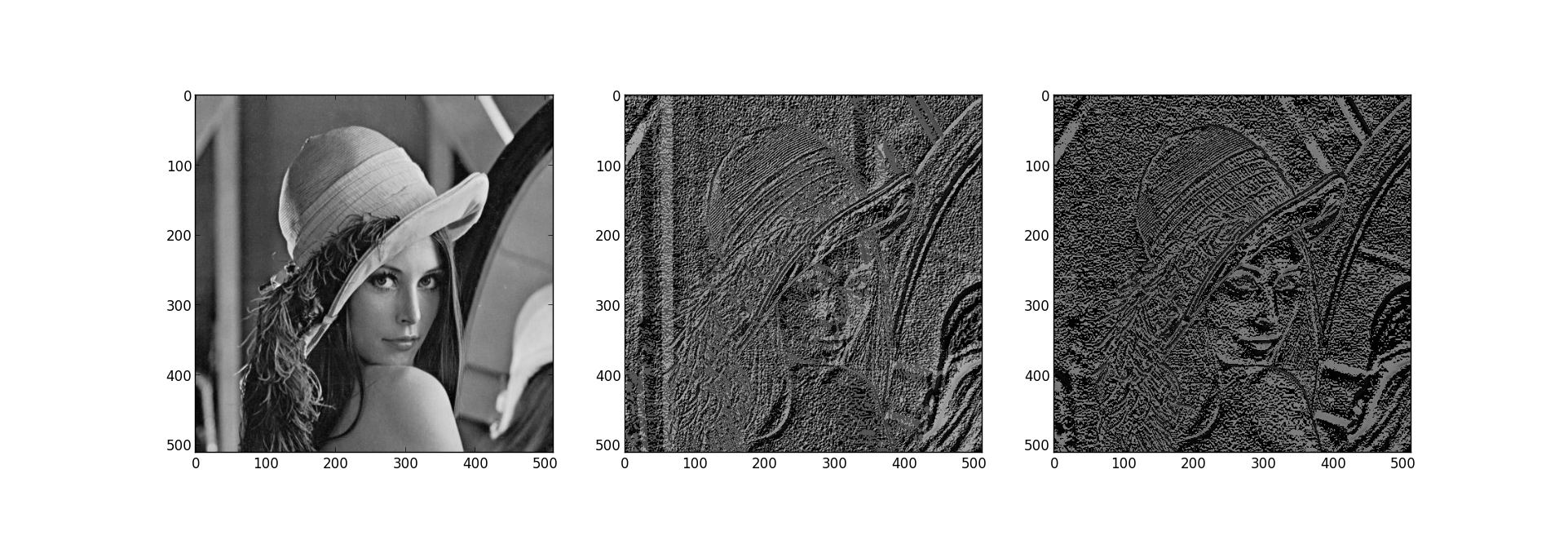
1. **Edges can also be extracted from the image by computing the contrast gradients within the image. The gradient equation is:**



**Calculate edge strength by taking the magnitude of the image derivatives[1]:**



*Hint: Use numpy.gradient function which will give you derivatives across x and y axis.*



*Original image/ Sum x & y derivatives/ Magnitude*

1. **“Canny” Edge detection program**
2. **Fourier Analysis (High-pass, low-pass, band-pass filters)**
3. **Compare to On/Off Ganglion cells - think about receptive fields.**
4. **Apply to a squarewave/sinwave grating?a**