1 Convolution

Use the function conv2 or imfilter to apply different convolution operators to an image. Using the function fspecial you may create different filter kernels to be used. Using the option 'same' in conv2 you get a result that has the same size as the first argument of the function (typically your image). The commands conv2 and imfilter are similar, but to use conv2 you need to convert the image and the kernel to double before. The imfilter command can filter uint8 images directly.

1. Examine at least 3 different filter kernels, among which there should be at least one sharpening (edge enhancing) and one smoothing filter and apply them in different sizes to the image cameraman.png, e.g. sizes 3 × 3, 7 × 7 and 31 × 31. Note that some filters are only available in one size when using fspecial. Include at least three figures in your report. One showing the original image, one figure showing the image after sharpening, and one figure showing the image after smoothing. For each filter, explain what the filter does to the image, and explain the effect of the different filter sizes.

Original Image:

Filter 1.
Average filter with a 3x3 kernel
The picture is smooth. If we increase the size of the kernel we will get a more blurry picture.

Filter 2.
Laplace filter with a 3x3 kernel
The picture is edge enhanced. Can't change filter size, but alpha is set to 0.2 (should be a number between 0 and 1).

Filter 3.
Gaussian filter with a 7x7 kernel
It is a smoothing filter. If we start with sigma as 0.5 we don't see that much of a difference. When we increase sigma the picture becomes blurrier.

2. Are the filters with filter kernels 'average', disk' and gaussian' examples of low-pass, band-pass or high-pass filters?
Low-pass filters!

3. Demonstrate how you can synthesize low-pass, band-pass and high-pass filtered images using simple arithmetics and filter kernels mentioned in Question 2.
A high-pass could be created by subtracting the low-pass filter from the unit impulse kernel.

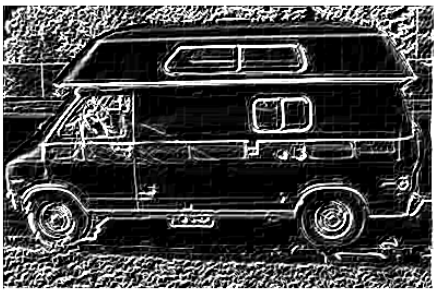
A band-pass could be created by subtracting a low-pass and high-pass filter (created above) from the unit impulse kernel.

2 The Sobel Filter

The Matlab function fspecial can produce filter kernels for Sobel filters.
4. Use this functionality to demonstrate Sobel filtering on cameraman.png and wagon.png.

You will need to do some arithmetics since the Sobel filter is not a linear filter and cannot be implemented using convolution alone. You may flip the x- and y-direction of a filter kernel using the matrix transpose command, e.g. A'.

The fast change in X-direction frequency combined with the fast change in Y-direction frequency



3 The Median Filter

Median filter is not included in fspecial function and to calculate this filter you can use function medfilt2.

5. Open the image wagon shot noise.png. Perform median filtering on the image using different sizes of the filter masks.

Filtermask 3x3:

Filtermask 11x11:

6. Compare visually the effect of median filtering to the effect of mean and Gauss filtering. Explain the differences on the image wagon shot noise.png. How does median filtering work compare to mean and Gauss filtering?

Median displays the pixel with the median intensity among the pixels inside the kernel (no smoothing) while mean and Gauss computes an average between the pixels inside the kernel.

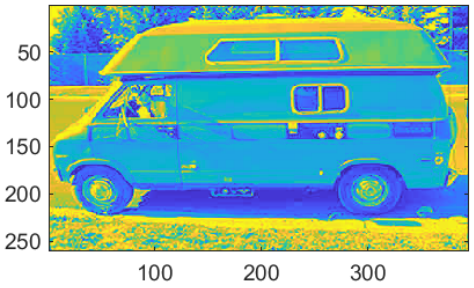
Gauss keeps a lot of the salt and pepper noise, but reduces some of it, although not as much as the mean filter which smoothens the white and black pixels by averaging with the rest of the image.

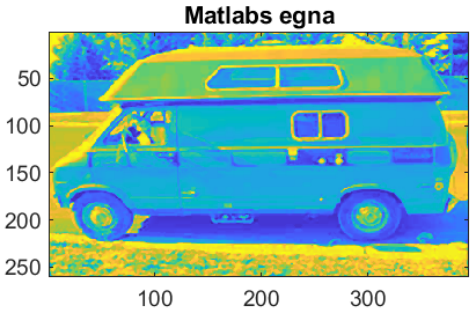
Gauss:

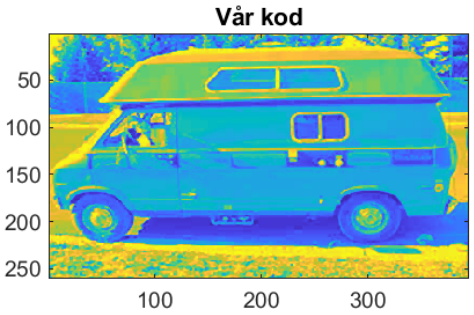
Mean:

7. In general, the median filter is more time consuming, why?
Have to compare and sort every element inside the kernel to know which one to display.

8. Implement your own code for 3x3 median filtering. You may use the Matlab function median that computes the median element of a vector. Use for instance two nested for-loops to iterate your filter for every neighbourhood in the image. The exact behaviour on the borders is not so important for this exercise and you may cut some corners here if it helps you.







9. If you implement a Gaussian filter using a large filter mask (and a large standard deviation), why do you get a black border around the image?

This is probably depends on how Matlab handles borders, or pixels outside the image. By getting a black border these are probably set to 0. In this case when the standard derivation is very large some pixels outside the image might need to be taken in count for the filtering.

4 Fast Fourier Transform

Open the image lines.png. Transform the image using the FFT and display the logarithm of the magnitude of the Fourier components using the command:

im = double(imread('lines.png'));

f = fftshift(fft2(im));

figure; imagesc(log(abs(f)));

Can you see a relationship between the lines in the Fourier spectra and the lines in the original image?
Hint: Figure 1 shows a few example images and their corresponding Fourier spectra.
The lines in the fourier spectra shows in which direction we find frequencies/changes in the original image.

10. Repeat the same procedure with the image cameraman.png. Comment on the spectrum that you see and compare it to the ordinary representation of the image. You may also try other images, for example circle.png or rectangle.png.

11. Experiment with FFT of an odd-length signal (image) of small length. For creating such a signal (image) use the command f = fftshift(fft2(rand(1,5))). What are some characteristic features of the centre value in f, i.e, f(1,3)? What are some characteristic features of the pair f(1,2) and f(1,4), as well as the pair f(1,1) and f(1,5)?

Do you notice any symmetry in these values? Do you notice the same symmetry in an even length vector?

Now we want to perform some filtering in the Fourier domain. Start by putting the frequency in f(1,2) to zero, i.e, f(1,2) = 0, and then transform the frequency vector back to an image using im = ifft2(ifftshift(f)) Is the resulting image real-valued or complex-valued? Now, try to put both f(1,2) = 0, and f(1,4) = 0 and do the IFFT. Is the resulting image real-valued or complex-valued? How should you zero out the frequencies to get a real-valued image when doing the IFFT? For these signals or very small "images", we recommend that you inspect the actual numbers by printing out the matrix in Matlab instead of viewing with imagesc.

12. Now modify the FFT representation of cameraman.png, by setting certain frequencies to 0, to create a low-pass version of the image. Use a circular filter for the best result, but feel free to simplify the task with a square pattern of your filter. You may blank out a part of a matrix using slices, e.g. A(20:30, 50:60) = 0. The resulting image should be real-valued after performing ifft2. Pay attention to the symmetry in the complex value that you observed in the previous exercise. You are not allowed to use the functions real or abs or similar ways to force a real-valued result. In your report, clearly explain your algorithm or include your code and comment. Now open the image freqdist.png. There is a pattern present in the image that should be filtered out. Remove it using notch filters in the frequency domain. This is similar to the case above, but you may need to be slightly more artistic when you blank out frequencies. Remember to take extra care to ensure the proper symmetry, so that your end result is a real-valued image after doing ifft2. Sometimes filtering produces signals that are outside the [0,255] range, so take extra care when your display the results using caxis or perhaps the command imcontrast.

13. Create a filter in the frequency-domain that suppresses the pattern in freqdist.png, but leaves the rest of the image as intact as possible. What does the filter look like? What do you see in the filtered image? Like the previous question, the resulting image should be real-valued after performing ifft. You are not allowed to use the functions real or abs or similar ways to force a real-valued result.
That's it. The exercise is done!

14. Any comments on the exercise?