Answers to questions in Lab 1: Filtering operations

Name: Arturs Kurzemnieks Program: Computer Science

Instructions: Complete the lab according to the instructions in the notes and respond to the questions stated below. Keep the answers short and focus on what is essential. Illustrate with figures only when explicitly requested.

Good luck!

Question 1: Repeat this exercise with the coordinates p and q set to (5, 9), (9, 5), (17, 9), (17, 121), (5, 1) and (125, 1) respectively. What do you observe?

Answers:

We enable specific frequencies corresponding to frequency *p*-1 in y direction and frequency *q*-1 in x direction (due to Matlab coordinates starting with 1 instead of 0). Therefore when setting (p,q) to, e.g. (5,9), we can see 5 peaks in y direction (4 full periods) and 9 peaks in x direction (8 periods), similarly for (9,5). These yield (*uc*,*vc*) values of (4,8) and (8,4), so they can be viewed as two rotations of the same vector. Since rotation in one domain becomes a rotation in the other domain, the corresponding image we get in the spatial domain is the same, just rotated as well.

Since we only set one point in the frequency spectrum, we get one frequency in each dimension.

As we have a resolution of 128x128, the maximum frequency we can represent is 64 (which gives 1 pixel stripes / period of 2 pixels), so everything above that we can only sample as some lower frequency. So for (17,121), when centering, we get the same the same frequencies with the same wavelengths as for (17,9), only symmetrical over y axis. Respectively, the (*uc*,*uv*) pairs we get are (16,8) and (16,-8), again a rotation resulting in two differently rotated images of essentially the same waveform.

Similar case for (5, 1) and (125, 1), which gives (*uc*,*vc*) pairs of (4,0) and (-4,0). Since we only have non-zero frequency in the vertical direction, we get horizontal bars. The negative frequency results in a shifted phase for the imaginary (sine) part.

Question 2: Explain how a position (p, q) in the Fourier domain will be projected as a sine wave in the spatial domain. Illustrate with a Matlab figure.

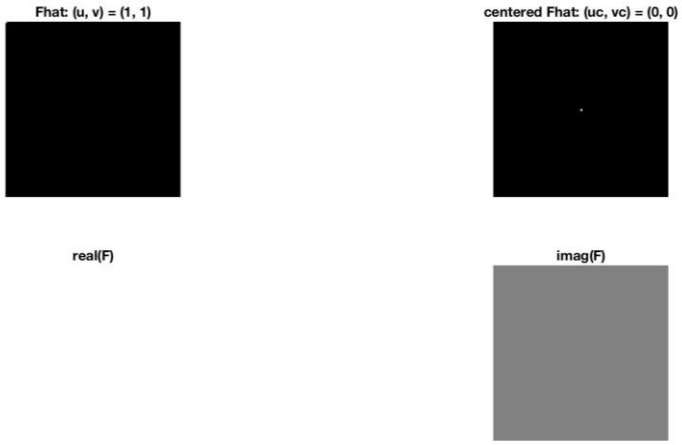
Answers:

If we expand the inverse discrete Fourier transform based on Euler's formula, we get

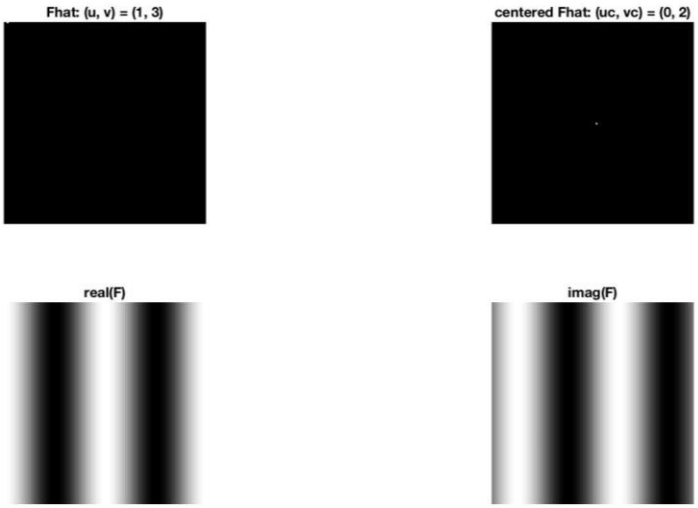
$$F(x) = \frac{1}{N} \sum_{x \in [0..N-1]^2} \hat{F}(u) e^{\frac{2\pi i u^T x}{N}} = \frac{1}{N} \sum_{u \in [0..N-1]^2} \hat{F}(u) cos\left(\frac{2\pi u^T x}{N}\right) + i sin\left(\frac{2\pi u^T x}{N}\right)$$

The position values *p* and *q* correspond to the vector *u* in the equation, so for every spatial position *x* we get some sum (divided by factor N) of real cos and imaginary sin values multiplied by the Fourier coefficient, which affects the amplitude and phase of the wave. Since we only enable one frequency pair (coefficient is non-zero only for the specific *p* and *q*), the sum yields one two-argument cosine in the real part and one two-argument sine in the imaginary part.
So from the value pair (*p*,*q*) we can get both frequency and direction. The waveform will have a frequency of *p*-1 periods in y direction and *q*-1 periods in x direction, with the orientation facing coordinates (*p*,*q*) as the two-argument functions change uniformly in that direction.
It is worth noting that the *p* and *q* values we provide are in Matlab coordinate system that starts with 1 and are mapped to frequencies *p*-1 and *q*-1 respectively.

p=1 and *q*=1 gives (0,0), i.e. 0 frequency so the cosine and sine are obviously constant, which we can see in the following example:



Similarly, for *p*=1 and *q*=3, it gives (0,2), so we have a non-zero frequency only in the x direction in which it completes 2 periods.



Question 3: How large is the amplitude? Write down the expression derived from Equation (4) in the notes. Complement the code (variable amplitude) accordingly.

Answers:

We can get the amplitude from $\hat{F}(u)$ that we set ourselves to one, so $|\hat{F}(u)| = 1$ as well. As the equation states, it is normalized by factor of $\frac{1}{N}$, which results in amplitude being $\frac{|\hat{F}(u)|}{N} = \frac{1}{N}$

If we have:

$$F(x) = \frac{1}{N} \sum_{x \in [0..N-1]^2} \hat{F}(u) e^{\frac{2\pi i u^T x}{N}}$$

we get the amplitude A as

$$A = \sqrt{Re(u)^2 + Im(u)^2}$$

where *Re(u)* = 1 and *Im(u)* = 0 are the real and imaginary parts of $\hat{F}(u) = 1$, i.e. we only have the real part by which we multiply all the cosines and sines. As we're summing over N points, we also normalize over N, giving the final amplitude of $\frac{1}{N}$ for these examples.

Question 4: How does the direction and length of the sine wave depend on p and q? Write down the explicit expression that can be found in the lecture notes. Complement the code (variable wavelength) accordingly.

Answers:
The larger *p* and *q* values we have, the larger frequency waveforms we have. The larger the frequency is, the shorter the length of the wave. Since we have a two-argument sine function, the *p* and *q* values act as coordinates, showing the direction vector the wave is pointed to.

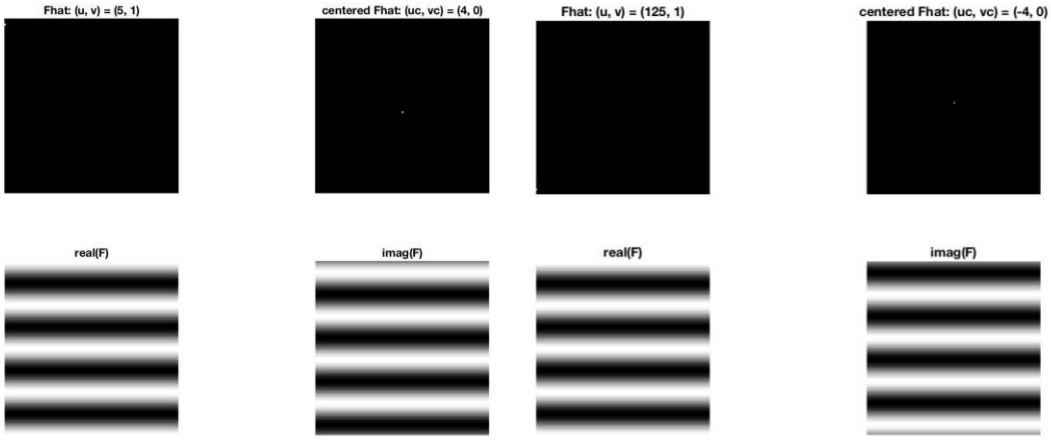
As for the wavelength, we have $\lambda = \frac{2\pi}{||\omega||}$
The frequencies corresponding to *p* and *q* are *uc* and *vc,* from those we calculate the corresponding angular frequencies and the wavelength, i.e.

$$\lambda = \frac{2\pi}{\sqrt{uc^2 + vc^2}}$$

Question 5: What happens when we pass the point in the center and either p or q exceeds half the image size? Explain and illustrate graphically with Matlab!

Answers:
We exceed the maximum frequency that can be represented in the spatial domain. Maximum frequency we can display is 2 pixels per period (1 pixel stripes), any frequencies higher than that we can't sample with the resolution, aliasing starts to occur and we happen to see some other lower frequency in the image, possibly with a phase shift.

An example can be seen with *p,q* pairs (5,1) and (125,1), which when converting to actual frequencies and zero-centering map to (4,0) and (-4,0) respectively, resulting in the same image for the real part and a phase shifted one for the imaginary part.



Question 6: What is the purpose of the instructions following the question *What is done by these instructions?* in the code?

Answers:

Zero centering. Originally, when we set $\hat{F}(u)$, we have the lowest frequencies in the upper-left corner (0,0), with values going 0..N-1 on both axes. By zero centering we shift the range from [0,N-1] to [$-\frac{N}{2}, \frac{N}{2} - 1$], and the angular frequency $\omega \in [0,2\pi)$ to $\omega \in [-\pi, \pi)$.

These instructions correspond to what fftshift does, swapping first and third quadrants, and second and fourth quadrants.

Even for *u* and *v* less than half image size we set *uc* = *u* – 1 and *vc* = *v* – 1 respectively as Matlab uses coordinate system that starts at 1, so we just do a mapping to the real frequencies.

Question 7: Why are these Fourier spectra concentrated to the borders of the images? Can you give a mathematical interpretation? Hint: think of the frequencies in the source image and

consider the resulting image as a Fourier transform applied to a 2D function. It might be easier to analyze each dimension separately!

Answers:

In the case of **F** we have a horizontal bar, so if we look at it as a 2D function, it has no changes moving in the x direction, it only changes in the y direction in which it can be viewed as a step function. Therefore in the Fourier domain we end up with a lot of different frequencies in the y direction that make up this step function, while in the x direction it is constant so we only have the zero frequency. Since for the uncentered version the origin (0,0) is at the upper-left corner (0 to 2π), we get all the values over the left border of the spectrum, where $v = 0$.

It's a similar case with **G**, where we only get change moving in the x direction and the y direction is constant, so we have all the values in the Fourier domain over the top border, where $u = 0$.

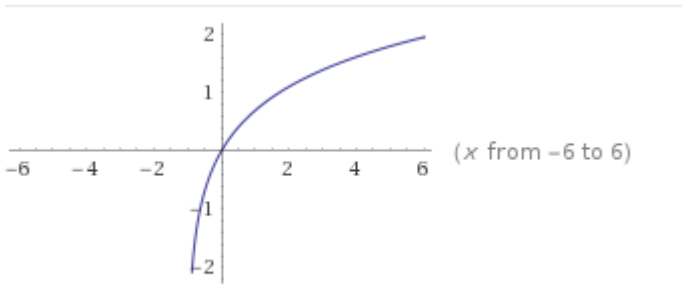
For **H** we have the bars from **F** and **G** combined (with **G** having higher amplitude), so the resulting Fourier transform is constructed the same way, as it is a linear transform, i.e. if we sum the two images and then transform, the result is the same as if we transform each one separately and then sum.
Finally for the shifted version the bars in the Fourier domain just move to the middle of the image as the 0 valued axis are now both centered there (origin in the center instead of upper-left corner).

Question 8: Why is the logarithm function applied?

Answers:

It is applied for value compression to improve visual readability when observing the magnitudes, bringing the lowest and highest values tighter together.

If we look at the graph of ***log(1 + x)***



we can see that the lowest value range is accented and thus can be more clearly distinguished, while the higher values flatten out more. Adding 1 to the term, i.e. doing ***log(1 + x)*** instead of ***log(x)***, ensures that 0 values still get mapped to 0.

Question 9: What conclusions can be drawn regarding linearity? From your observations can you derive a mathematical expression in the general case?

Answers:
As mentioned already in Question 7, the Fourier transform is linear, which we can clearly observe in the case of **H = F + G**. As the resulting transform is made up of the individual transforms of **F** and **G**, we can say that for any *f, g* the Fourier transform **F**(*f* + *g*) = **F**(*f*) + **F**(*g*)

Question 10: Are there any other ways to compute the last image? Remember what multiplication in Fourier domain equals to in the spatial domain! Perform these alternative computations in practice.

Answers:

Multiplication in spatial domain is the same as convolution in Fourier domain, i.e.

$$\mathrm{F(hf)} = \mathrm{F(h)} * \mathrm{F(f)}$$

so instead of doing point-wise multiplication on the images, we can transform each of them separately and then do a convolution between them in the Fourier domain.

Question 11: What conclusions can be drawn from comparing the results with those in the previous exercise? See how the source images have changed and analyze the effects of scaling.

Answers:

Compression in spatial domain results in expansion in Fourier domain and vice versa.
The image in question can be viewed as scaled version of the previous image in question 10, only stretched in x axis and compressed in y axis, this results in the Fourier transform doing the opposite, i.e. compressed in x axis and stretched in y axis, which is logical, as in x axis the image is now not changing so fast so less higher frequencies are needed, while in y axis the change is now faster are more higher frequencies are needed in the Fourier domain.

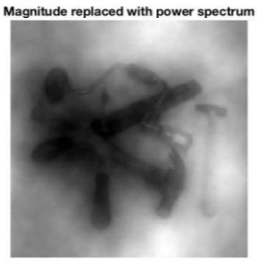
Question 12: What can be said about possible similarities and differences? Hint: think of the frequencies and how they are affected by the rotation.

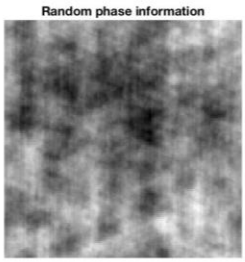
Answers:
Rotation in one domain results in the same rotation in the other domain, so ideally we should be getting the same spectrum as $\hat{F}$ before, just rotated by however degrees we rotated **F** to get **G.** In practice it is not always entirely the case, as when rotating the original image we start to get jagged edges (aliasing) as we're limited by resolution, so they're not perfectly straight lines anymore. This introduces some additional frequencies in the Fourier domain so the transforms don't look completely the same. It is most obvious in cases of 30 and 60 degrees here, while the 90 degree rotation is a perfect rotation as we can get perfectly straight lines again, so the Fourier transform actually looks the same, just rotated.

Question 13: What information is contained in the phase and in the magnitude of the Fourier transform?

Answers:

Magnitude maps to intensities in the spatial domain, i.e. the resulting grey levels, while phase maps to positioning, defining where edges will end up in the image. That's why translation of an image doesn't change the magnitude (Fourier spectrum), only the phase is shifted.





We can observe in our experiments that replacing the magnitude with the square of it preserves the edges and really only exaggerates the resulting color values. Meanwhile, if we replace the phase information, we get a seemingly random image. They have the same magnitudes, and these images should have the same histograms, but as the phase is now random, no edge information is preserved and the color values are basically randomly displaced over the image.

Question 14: Show the impulse response and variance for the above-mentioned t-values. What are the variances of your discretized Gaussian kernel for t = 0.1, 0.3, 1.0, 10.0 and 100.0?

Answers:

For t = 0.1: $\begin{matrix} 0.0133 & 0 \\ 0 & 0.0133 \end{matrix}$

For t = 0.3: $\begin{matrix} 0.2811 & 0 \\ 0 & 0.2811 \end{matrix}$

For t = 1: $\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$

For t = 10: $\begin{matrix} 10 & 0 \\ 0 & 10 \end{matrix}$

For t = 100: $\begin{matrix} 100 & 0 \\ 0 & 100 \end{matrix}$

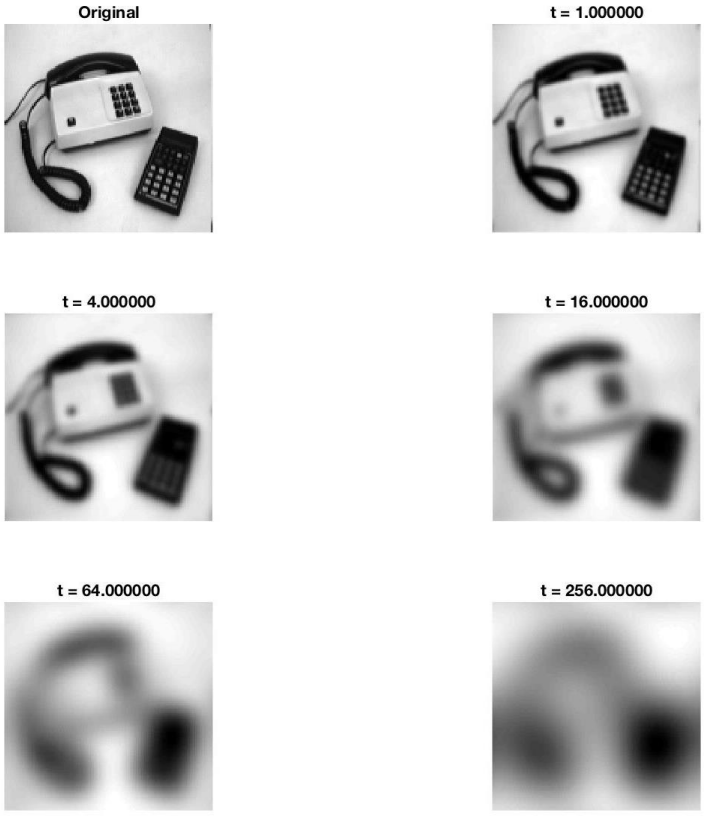
Question 15: Are the results different from or similar to the estimated variance? How does the result correspond to the ideal continuous case? Lead: think of the relation between spatial and Fourier domains for different values of t.

Answers:

For the larger t values they're similar (practically the same). The smaller the t value, the larger the error it seems. This could be due to that fact that for smaller variances the filter is sharper, with the values being tighter together, which corresponds to expansion in the Fourier domain as we need higher frequencies to represent them, so we might not be able to represent the higher frequencies anymore due to lack of resolution.

Question 16: Convolve a couple of images with Gaussian functions of different variances (like t = 1.0, 4.0, 16.0, 64.0 and 256.0) and present your results. What effects can you observe?

Answers:



The larger the variance, the more blurred the output image becomes, as the filter gets more spread out and a larger area around each pixel contributing to its output value.

Question 17: What are the positive and negative effects for each type of filter? Describe what you observe and name the effects that you recognize. How do the results depend on the filter parameters? Illustrate with Matlab figure(s).

Answers:

Gaussian filters seem to work relatively well with most noise, except for salt and pepper noise which is only smudges out over the rest of the picture, making it even worse. The good quality about Gaussian filters is that they're symmetrical in all directions, so having a rotated image technically doesn't change the response.

Gaussian noise, Gaussian filter with t=1

Gaussian noise, Gaussian filter with t=2

As the parameter t (variance) gets increased, the more spread out the filter becomes and the blurrier the image becomes after applying it.

S&P noise, Gaussian filter with t=2

It can be observed that for salt and pepper noise the Gaussian filter isn't effective and the noise is still heavily present.

The median filter looks at a small neighborhood of values (the window) and takes the median value, thus eliminating extremes. The positive is that it's very effective at getting rid of extreme values like the salt and pepper noise and it preserves somewhat clearly defined edges, but they can get a bit deformed, as the filter produces painting-like images that seem to be made out of patches of color.

S&P noise, Median filter with 4x4 window

It can be observed in this case of a 4x4 window, that all the salt and pepper noise that was applied to this image beforehand is eliminated, but the edges can look swirly or jagged, and fine detail is being lost.

S&P noise, Median filter with 3x3 window

For a smaller window size the effects are less noticeable, but some noise still gets through.

Gaussian noise, Median filter with 3x3 window

For Gaussian noise the median filter is somewhat effective, although it's still noticeable.

The ideal low-pass filter cuts away frequencies based on a given cutoff parameter. The higher the value, the higher the threshold and more frequencies are kept. This method, while supposedly it should remove high-frequency noise, it also heavily affects edges and all small details that correspond to high frequencies as well. Additionally, when removing the higher frequency components, it results in swirling patterns all over the image, as we start to see the lower frequency waveforms without the higher frequency ones to compensate and smooth out, ultimately just generating a new kind of noise in the image.

Gaussian noise, ideal low pass filter with cutoff 0.2

Gaussian noise, ideal low pass filter with cutoff 0.25

S&P noise, ideal low pass filter with cutoff 0.2

S&P noise, ideal low pass filter with cutoff 0.3

It can be seen that for salt and pepper noise the low-pass filter is mostly useless.

Question 18: What conclusions can you draw from comparing the results of the respective methods?

Answers:

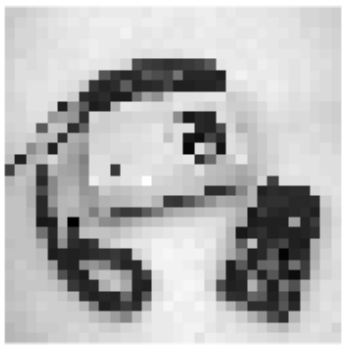
Gaussian noise in general is very hard to get rid of. You can blur it out so it's less sharp but it's still present and noticeable on smooth surfaces when using Gaussian filters, as it only gets more smudged out among local areas. Low-pass filters don't seem to do much good in any case. It's easier to run into rings and blurring than it is to get rid of the noise. Still, applying some minor degree of Gaussian blur is probably the best approach for most noise.

As for the salt and pepper noise, it's an extreme value noise, and the only type of filter that seems to work is median filtering, others just make it worse. But median filtering works really well for this type of noise, as naturally it discards extreme values in local areas. Of course it has the drawback of deforming edges, but in overall it's the closest we can get to a completely denoised image.

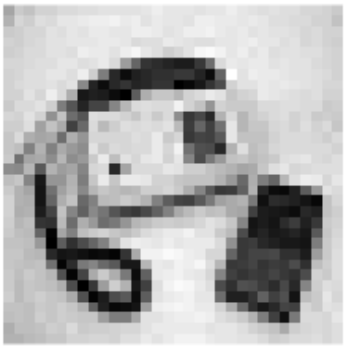
Question 19: What effects do you observe when subsampling the original image and the smoothed variants? Illustrate both filters with the best results found for iteration i = 4.

Answers:

For i = 4


Original subsampled


Gaussian (t=1)


Low-pass (cutoff=0.3)

When subsampling the original image, information loss can be seen as we're completely dropping some pixel values and distorted, jagged structures appear due to spatial aliasing.

Prefiltering before subsampling seems to preserve the image information better and we don't encounter distorted structures and the image has less artifacts.

Question 20: What conclusions can you draw regarding the effects of smoothing when combined with subsampling? Hint: think in terms of frequencies and side effects.

Answers:

When subsampling, we're losing data from the higher frequencies and introduce aliasing. Prefiltering the image brings more information to the lower frequencies that we can now preserve better when subsampling.
