Lab 1: Filtering operations

DD2423 Image analysis and Computer Vision, 2019

Federico Favia
favia@kth.se

Martin De Pellegrini
martindp@kth.se

November 20, 2019

1 Properties of the discrete Fourier transform

1.1 Basis functions

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?*

By changing the coordinates of the point defined by (p, q), which are in frequency domain, it is noticeable that the position of the impulse influences the frequency and the phase of the sine resulting from the inverse Fourier transform of the image. Therefore, increasing p and q leads to increase the frequency variation of the sinusoid wave in the spatial domain.

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.*

As we can see from Figure 1, a sine wave in spatial domain is projected as a position (p, q) in the Fourier domain, in this case we see two positions because it is the shifted spectrum due to its periodicity. For simplicity we have shown a one-dimensional horizontal sine along x-direction. This is because the magnitude Fourier transformation determines how much a frequency weighs for a certain signal and this case the position represents the frequency of the wave (and its symmetric value). If we increase the number of periods of the sine therefore the position will become larger along x-direction because the frequency of the wave is higher.

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

The amplitude is 61·10−5, as we can see from (2). Since in MATLAB the inverse Fast Fourier Tranform (iFFT) is computed with a normalization factor $1/N^2$, the steps to compute the amplitude of the image in the space domain are the follow:

$$\begin{aligned} f(x) = F^{-1}\{F(\omega)\} = \frac{1}{N}\sum F(\omega)e^{j\frac{2\pi\omega^T x}{N}} = \frac{1}{128}e^{j\frac{2\pi(ux+vy)}{128}} = \\ = \frac{1}{128}(cos(\frac{ux+vy}{20}) + jsin(\frac{ux+vy}{20})) \end{aligned} \tag{1}$$

$$A = \frac{1}{128^2}\sqrt{Re^2 + Im^2} = \frac{1}{128^2}\sqrt{cos^2() + sin^2()} = \frac{1}{128^2} = 61 \cdot 10^{-5} \tag{2}$$

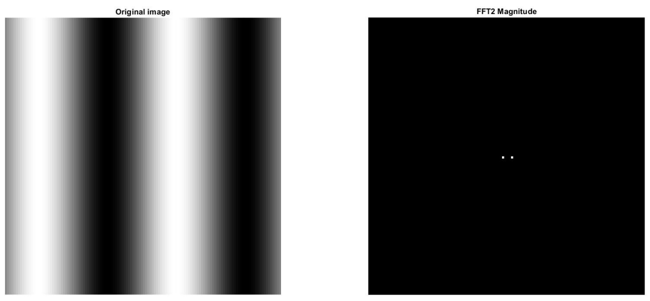

Figure 1: Sine wave in spatial domain and corresponding Fourier transform, a position, in frequency domain.

But the same result can be achieved by using the function max(abs(F(:))).

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.*

The direction of the sine wave is given by the direction of the vector defined by the two points (0, 0) and (p, q). On the other hand, the wavelength λ of the sine is inversely proportional to the frequency. Thus a an higher frequency in one of the two directions (p = w_1, q = w_2) will decrease the length of the sine, which is varying faster, as we can deduct from equation (3).

$$\lambda = \frac{2\pi}{\|w\|} = \frac{2\pi}{\sqrt{w_1^2 + w_2^2}} \tag{3}$$

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!*

When we pass the point in the center of the image, as we can see from Figure 2, the frequency of the wave is the highest in both directions, resulting in a perfect diagonal sine wave in spatial domain. While if either p or q exceeds half the image size, as we can observe from Figure 3, it is equivalent of having high frequencies in on of the two direction, negative coordinate in the corresponding shifted Fourier domain because of the periodicity of the spectrum.

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

The point (u, v) in the Fourier domain is shifted with respect the new center. The instructions in the code move the origin (0, 0) of the Fourier domain image from the top-left corner to the center of the image, since when you compute direct transform, the center of the frequency space is placed in the center of the image. This can be done thanks to the periodicity of the Fourier spectrum with a period of 2π. Also in this section of the code, the new center (uc, vc) is computed just for visualization purpose since from an implementing perspective the spectrum is shifted by using the instruction *fftshift*.

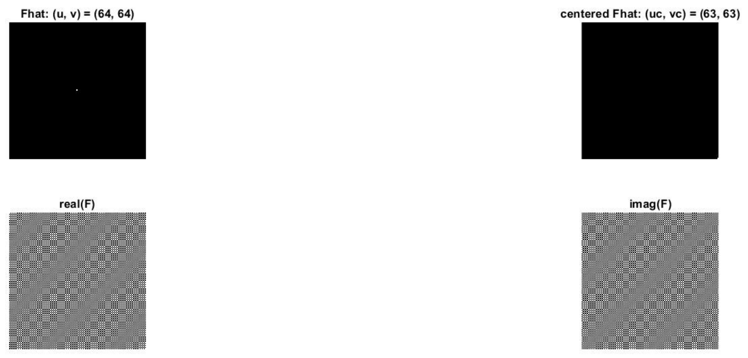

Figure 2: Position of the point (p, q) passed in the center of image in frequency domain.

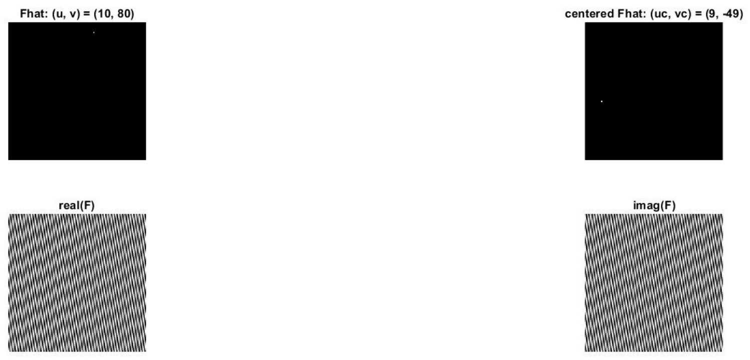

Figure 3: Position of the point (p, q) passed with one coordinate exceeding half the image size in frequency domain.

1.2 Linearity

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!*

We observe that Fourier spectra of the images are concentrated to the borders because we are assuming to show the Fourier Spectrum without shifting it, therefore for Matlab the origin is in the top-left corner of the image. In the first image F, we have only transition in y-direction, so the corresponding Fourier spectra will be in the left-border because along x-direction there is no change, but will take different values (namely a *sinc*) in y-direction. The contrary applies for the second image G. The third image H is a linear combination of the two, therefore also the resulting Fourier spectrum will behave as a linear combination of the two spectra. When we apply the shift of the spectrum thanks to its periodictiy, then it is more intuitive to visualize it.

Question 8: *Why is the logarithm function applied?*

With the logarithm of the Fourier transform, we can see many minor frequencies. Using the logarithm function helps indeed to bring out details of the Fourier transform in regions where it is very close to zero. The drawback is that, since images are represented by square pixels, discretization noise appears.

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

This exercise shows linearity property of Fourier transform because we can show through the funcion

```
showgrey(ifft2(Fhat+2*Ghat));
```

that the resulting image is the same as the input image $H = F + 2 * G$. Indeed, it is known that the Fourier Transform is linear (4). The Fourier Transform of a sum of functions, is the sum of the Fourier Transforms of the functions. Also, if you multiply a function by a constant, the Fourier Transform is multiplied by the same constant.

$$\alpha \cdot f_1(x,y) + \beta \cdot f_2(x,y) \overset{F}{\longleftrightarrow} \alpha \cdot X_1(u,v) + \beta \cdot X_2(u,v) \tag{4}$$

1.3 Multiplication

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.*

The result we obtain from the multiplication is a square in the center of the image, the relative Fourier transform is a *sinc* in the frequency domain. It is possible to achieve the same result by first computing the Fourier Transform of the two images separately, then compute the convolution between the two, because convolution in spatial domain is multiplication in Fourier domain and viceversa (Eq. 5).

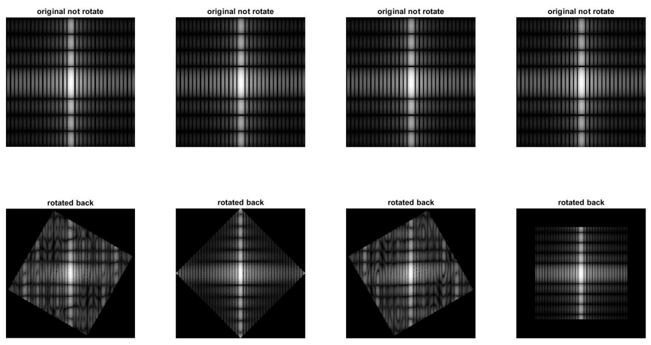

Figure 4: Impulse responses for gaussian kernels with different variances.

$$\begin{aligned} f * g &= \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\} \\ f \cdot g &= \mathcal{F}^{-1}\{\mathcal{F}\{f\} * \mathcal{F}\{g\}\} \end{aligned} \tag{5}$$

1.4 Scaling

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.*

From the scaling theorem of the 1-D Fourier Transform we know that stretching or squeezing the signal in the original domain by a factor α, the result in the transformed domain is a squeezed or stretched spectra respectively. The same happens in the 2-D formulation. In the previous section we have an image with a white stripe in the center, after computing the transformation we have a *sinc* in vertical direction. After scaling the image, we obtain a shrunken white line that in Fourier is transformed as thicker *sinc*. Furthermore, there are also variation along the x direction.

1.5 Rotation

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

We can clearly see from Figure 4 that one of the property of the Fourier Transform is rotation invariance. This means that if an image is rotated and then transformed, we achieve the same result of transforming the image and then rotating its spectrum. Therefore applying a rotation on an image will cause the same rotation on its spectra.

1.6 Information in Fourier phase and magnitude

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

From the functions applied on the test images we can observe that a spectrum in which the phase is preserved and the magnitude is replaced by a non-linear function represents better the original image, with respect to a spectrum in which the magnitude is preserved and the phase is replaced by a uniform distribution. Typically phase contains more information than magnitude and therefore it is more important to preserve it during filtering. Phase defines how waveforms are shifted along its direction (where edges will end up in the image), while magnitude defines how large the waveforms are, how much a certain spacial frequency contributes to the image (which grey levels are on either side of edge). Reconstructing an image from only the magnitude of Fourier spectrum will return almost nothing, while using only phase at least the shapes/edges are recovered, but to reconstruct perfectly the original image both information are needed.

2 Gaussian convolution implemented via FFT

2.1 Filtering procedure

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: The resulting variances computed on the discretized Gaussian kernel by the MATLAB function

```
variance(psf);
```

are pretty close to the ideal covariance matrix, since one diagonal can be approximated to zero due to the very small values:

$$C(\cdot, \cdot; t) = t \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Co-variances matrices of the discretized Gaussian kernel are shown below while impulse responses are plotted in Figure 5.

$$C(\cdot, \cdot; 0.1) = \begin{bmatrix} 0.013 & 1.1e-14 \\ 1.1e-14 & 0.013 \end{bmatrix}$$

.

$$C(\cdot, \cdot; 0.3) = \begin{bmatrix} 0.28 & 1.1.9e-14 \\ 1.19e-14 & 0.28 \end{bmatrix}$$

.

$$C(\cdot, \cdot; 1.0) = \begin{bmatrix} 1 & 8.1e-15 \\ 8.1e-15 & 1 \end{bmatrix}$$

.

$$C(\cdot, \cdot; 10.0) = \begin{bmatrix} 10 & 1.7e-16 \\ 1.7e-16 & 10 \end{bmatrix}$$

.

$$C(\cdot, \cdot; 100.0) = \begin{bmatrix} 100 & 3.3e-16 \\ 3.3e-16 & 100 \end{bmatrix}$$

.

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.*

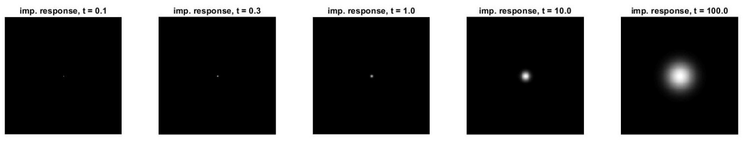

Figure 5: Impulse responses for gaussian kernels with different variances.

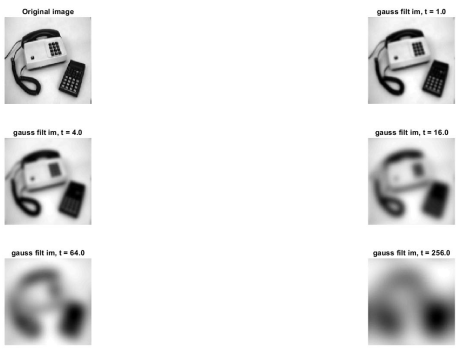

Figure 6: Gaussian smoothing with different variances on *phonecalc128*.

As it is noticeable form the matrices reported in the previous point, the variances are close to the ideal case. Only the variance of the Gaussian kernel with t = 0.1 is not close to the ideal case because of a discretization error introduced in the spacial domain. Therefore a very small variance will cause a Gaussian kernel with a sharper shape in spatial domain.

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?*

In Figure 6 we present the results when applying the Gaussian filter on the image *phonecalc128*. What we can observe is that the effect of the Gaussian filter is the smoothing of the image, meaning that it is acting as a low pass filter. As we increase the variance of the Gaussian kernel, the image is smoothed/blurred with higher strength, because the corresponding Gaussian kernel in frequency will be sharper and will cut more frequencies.

3 Smoothing

3.1 Smoothing of noisy data

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).*

We can observe the results in Figure 7.

- The gaussian smoothing filter performs better on the gaussian noisy image (best variance is 0.9), but looses small details like the keyboard, while for

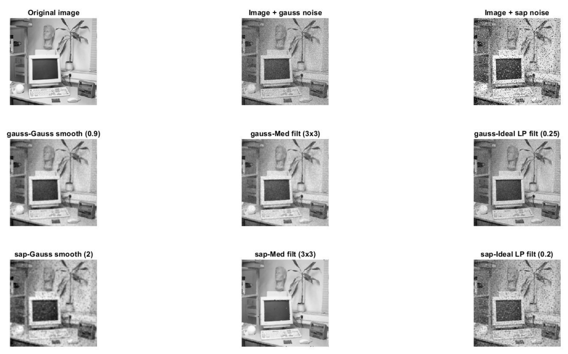

Figure 7: Original image, noisy images and result of different filters to noisy images, respectively gaussian smoothing (first column), median filtering (second column) and ideal low-pass filtering (third column) on gaussian noisy (second row) and salt and pepper (third row).

removing the salt and pepper noisy points it is needed an higher parameter equal to 2, that also leads to loose most of the high frequencies, resulting in a very blurry image.

- The median filter is particularly useful for salt and pepper noisy image but at the same time it performs not badly on gaussian noisy image, and it preserves most of significant edges. A window of 3 by 3 pixels is chosen for both images: increasing it leads to a cartoon effect even though a slightly larger windows performs better for salt and pepper noisy image.
- The ideal low-pass filter performs very poorly on salt and pepper noisy image (best cut-off frequency = 0.2 and still blocks are visible) while on the gaussian noisy with a cut-off frequency of 0.25 it is slightly better helpful to smooth the noise, even though still present and looking very similar to the noisy one.

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

The best filter for the gaussian noisy image is the gaussian smoothing filter while the median filter performs better against salt and pepper noise. This means that selecting the best filter to smooth noisy data require both:

- Knowledge on type of noise, aspect of the spectrum (lot of details means many more high frequencies) and dimension of the image.
- A trial-and-error method to choose best parameters of the filter.

3.2 Smoothing and subsampling

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 $i = 4$.*

Subsampling the original image introduce high variation between pixels, but

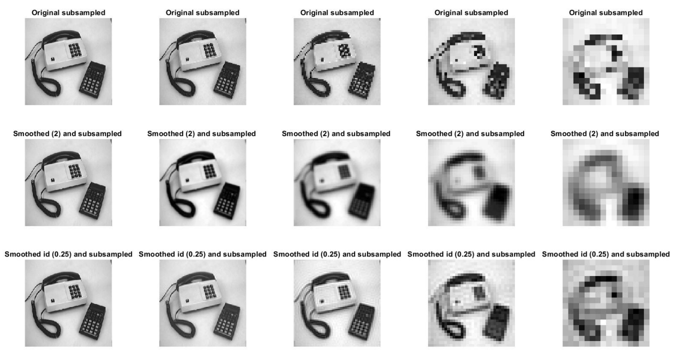
Original subsampled
Original subsampled
Original subsampled
Original subsampled
Original subsampled
Smoothed (2) and subsampled
Smoothed (2) and subsampled
Smoothed (2) and subsampled
Smoothed (2) and subsampled
Smoothed (2) and subsampled
Smoothed id (0.25) and subsampled
Smoothed id (0.25) and subsampled
Smoothed id (0.25) and subsampled
Smoothed id (0.25) and subsampled
Smoothed id (0.25) and subsampled

Figure 8: Results of downsampling (first row), gaussian smoothing and downsampling (second row), ideal low-pass filtering and downsampling (third row) on *phonecalc256*.

if before it is applied a smoothing operation, the increasing variation is less perceivable. Figure 8 shows the difference between performing only subsampling, gaussian smoothing plus subsampling and finally ideal low-pass filtering plus subsampling. At iteration #4 it is possible to observe that subsampling the original image without smoothing it before leads in a high degradation of the image, thus loosing almost all significant objects. Instead, applying a smoothing operator before subsampling allows to preserve some information such as shape of relevant objects (also at different scales), like in this case the pocket calculator. Further explanations are stated in the next question. Results with best parameters (variance = 2 for gaussian kernel, and cut-off frequency = 0.25 for ideal low-pass filter) are shown as well in Figure 8.

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.*

Usually performing smoothing before subsampling is a recommended action if we want to control the loss of information after downsampling. When an image contains fast variations (which increase if we iterate the downsampling), the subsampling operation can produce visually weird aliasing artifacts. A blurring operation instead would attenuate image sharpness (highest frequencies), thus reducing the apparent aliasing aspects. We can observe more clearly this effect in Figure 9, which takes a critical example of a brick wall. In the recommended approach (smoothing and subsampling) the shapes are less deformed. We also know from (digital) sampling theorem by Nyquist-Shannon that signals should be properly band-limited, before they are downsampled ($f_{\mathrm{c}} = 2 \cdot f_{\max}$).

References

[1] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing*, Prentice Hall, 2nd ed., 2002

[2] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2010

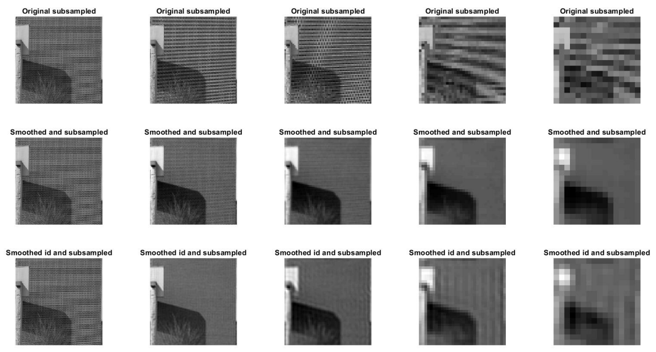

Figure 9: Aliasing effects of downsampling and comparison with smoothing and downsampling.