School of Computer Science and Communication

DD2423 Image Analysis and Computer Vision:

Lab 2: Filtering operations

The goal of this lab is to help you to

- become familiar with the two-dimensional Fourier transform, get an understanding of its properties and test these properties in practice,
- get an understanding of how to use Fourier transform on image data, expressed spatially or in the Fourier domain,
- test how different smoothing operations are influenced by various kinds of noise and how contrast sharpening can be performed in practice,
- get an understanding of how multiple linear operations can be combined into a composite linear operation,
- learn the relation between the continuous and discrete Fourier transform and test how continuous filters can be discretized spatially as well as in the Fourier domain,
- get practical experience of differences between synthetic and real data.

The purpose of exercise 1 is for you to become familiar with the characteristics of the Fourier transform. The purpose of exercise 2 is for you reflect on what you have learned.

Remaining exercises are more problem-oriented and experimental in nature and formulations are thus more concise. If you happen to have difficulties with Matlab or the environment in general contact the lab assistants during the scheduled lab hours.

Reporting: For you to efficiently report your labs, you ought to create script files that reproduce the experimental results. If some results take too much time to compute, printouts are recommended. You should also summarize results and conclusions from your experiments, writing down answers to explicitly stated questions.

For most exercises it is recommended that you create illustrations with multiple images simultaneously shown on screen. For this purpose use the embedded Matlab command `subplot` (see "`help subplot`").

As *prerequisites* to this lab exercise, you should have studied the course material regarding spatial filtering and image restoration. You should also have finished and reported the results of Lab 1.

1 Properties of the discrete Fourier transform

1.1 The continuous and discrete Fourier transform

From the definitions used during the lectures the continuous Fourier transform $\hat{f}\colon \mathbb{R}^2 \to \mathbb{C}$ of a two-dimensional signal $f\colon \mathbb{R}^2 \to \mathbb{R}$ is

$$\hat{f}(\omega) = \mathcal{F}_C(f)(\omega) = \int_{x\in\mathbb{R}^2} f(x)\, e^{-i\omega^T x} dx \tag{1}$$

and the Fourier inversion theorem states that

$$f(x) = \mathcal{F}_C^{-1}(\hat{f})(x) = \frac{1}{(2\pi)^2} \int_{\omega\in\mathbb{R}^2} \hat{f}(\omega)\, e^{+i\omega^T x} d\omega. \tag{2}$$

Correspondingly, in the discrete case the Fourier transform $\hat{F}\colon [0..N-1]^2 \to \mathbb{C}$ of a quadratic image $F\colon [0..N-1]^2 \to \mathbb{R}$ of N^2 image elements is defined as

$$\hat{F}(u) = \mathcal{F}_D(F)(u) = \frac{1}{N} \sum_{x\in[0..N-1]^2} F(x)\, e^{\frac{-2\pi i u^T x}{N}} \tag{3}$$

with the corresponding inversion theorem

$$F(x) = \mathcal{F}_D^{-1}(\hat{F})(x) = \frac{1}{N} \sum_{u\in[0..N-1]^2} \hat{F}(u)\, e^{\frac{+2\pi i u^T x}{N}}. \tag{4}$$

There expressions are symmetric with respect to the factor $1/N$ in the Fourier transform and inverse respectively.

Note that in MATLAB the fast Fourier transform (FFT) is implemented using a factor 1 in the FFT-routine, and factor $1/N^2$ in the inverse.

1.2 Definition domains

In the continuous case the spatial variable $x = (x_1, x_2)$ and the frequency variable $\omega = (\omega_1, \omega_2)$ are defined in the whole two-dimensional plane, i.e.

$$x, \omega \in \mathbb{R}^2,$$

while the corresponding discrete variables $x = (x_1, x_2)$ and $u = (u_1, u_2)$ are defined in the interval $[0..N-1]^2$, i.e.

$$x_1, x_2, u_1, u_2 \in [0..N-1] = \{0, 1, \ldots, N-1\}.$$

From the periodicity of the basis functions in the discrete case, thus follows that the discrete Fourier transform will become periodic with a period of N. To maintain the consistency with the inversion theorem we similarly have to consider the original signal as periodic with the same period.

By comparing equations (1) and (3) we can relate the continuous angular frequency variable ω to the discrete frequency variable u:

$$\omega_D = 2\pi \frac{u}{N}. \tag{5}$$

Since $u \in [0..N-1]^2$ we may from this relation treat the frequency variable ω_D as being defined in the interval $[0, 2\pi]^2$.

Furthermore, the discrete Fourier transform may similarly be treated as periodic with a period of 2π, and we may change the definition domain to the interval $[-\pi, \pi]^2$ without losing in generality. This corresponds to a *centering* operation in the discrete Fourier transform.

1.3 Basis functions

The Fourier transform can be regarded as a change of basis functions, from compact (discrete) delta functions defined on the Cartesian image domain to complex exponential function with maximum spatial extension. In the discrete case this is even more obvious noting that the discrete Fourier transform can be found by pre- and post-multiplying the image with orthogonal matrices.

Doing so we may consider the complex pixel values in the Fourier transform $\hat{F}$, as components of the discrete image F, with respect to the new basis. Specifically, the basis element corresponding to an image point at coordinates (p, q) in $\hat{F}$ is proportional to

$$\begin{pmatrix} e^{i2\pi\frac{p\cdot 0}{N}} \\ \\ \vdots \\ \\ e^{i2\pi\frac{p\cdot(N-1)}{N}} \end{pmatrix} \begin{pmatrix} e^{i2\pi\frac{q\cdot 0}{N}} & \ldots & e^{i2\pi\frac{q\cdot(N-1)}{N}} \end{pmatrix}$$

These basis vectors are the discrete counterparts to the complex exponential functions

$$e_w(x) = e^{i\omega^T x}$$

You may convince yourself that this is true by calculating the inverse discrete Fourier transform of an image $\hat{F}$, that is zero everywhere except for a single point (p, q) at which the value is one. For such an image the expansion of $\hat{F}$ only contains one term, i.e. the basis vector given by the index (p, q). To visualize this, define a 128×128 pixel image, with elements defined as

```
Fhat = zeros(128, 128);
Fhat(p , q ) = 1;
```

with $(p, q) = (5, 9)$. Display this image on screen using `showgrey`. Thereafter, compute its inverse discrete Fourier transform with

```
F = ifft2(Fhat);
```

and look at its real and imaginary parts, as well as its magnitude and phase, by writing

```
Fabsmax = max(abs(F(:)));
showgrey(real(F), 64, -Fabsmax, Fabsmax)
showgrey(imag(F), 64, -Fabsmax, Fabsmax)
showgrey(abs(F), 64, -Fabsmax, Fabsmax)
showgrey(angle(F), 64, -pi, pi)
```

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

A clever way of organizing the experiment is by writing a small procedure `fftwave`, that assembles the resulting images into a single figure in MATLAB, with the help of the command `subplot` and illustrative headings. [1] Please use this procedure as of prototype example of how the experimentation in later exercises can be arranged.

[1] A template of such a procedure can be found in `fftwavetemplate.m` in the course library.

```
function fftwave(u, v, sz)

if (nargin < 2)
  error('Requires at least two input arguments.')
end
if (nargin == 2)
  sz = 128;
end

Fhat = zeros(sz);
Fhat(u, v) = 1;

F = ifft2(Fhat);
Fabsmax = max(abs(F(:)));

subplot(3, 2, 1);
showgrey(Fhat);
title(sprintf('Fhat: (u, v) = (%d, %d)', u, v))

% What is done by these instructions?
if (u <= sz/2)
  uc = u - 1;
else
  uc = u - 1 - sz;
end
if (v <= sz/2)
  vc = v - 1;
else
  vc = v - 1 - sz;
end

wavelength = 0.0;  % Replace by correct expression
amplitude = 0.0;   % Replace by correct expression

subplot(3, 2, 2);
showgrey(fftshift(Fhat));
title(sprintf('centered Fhat: (uc, vc) = (%d, %d)', uc, vc))

subplot(3, 2, 3);
showgrey(real(F), 64, -Fabsmax, Fabsmax);
title('real(F)')

subplot(3, 2, 4);
showgrey(imag(F), 64, -Fabsmax, Fabsmax);
title('imag(F)')

subplot(3, 2, 5);
showgrey(abs(F), 64, -Fabsmax, Fabsmax);
title(sprintf('abs(F) (amplitude %f)', amplitude))

subplot(3, 2, 6);
showgrey(angle(F), 64, -pi, pi);
title(sprintf('angle(F) (wavelength %f)', wavelength))
```

- <u>*Question 2:*</u> **Explain and illustrate with a Matlab figure how a position (p, q) in the Fourier domain will be projected as a sine wave in the spatial domain.**

- <u>*Question 3:*</u> **How large is the amplitude? Write down the expression derived from Equation (4) in these notes. Complement the code (variable `amplitude`) accordingly.**

- <u>*Question 4:*</u> **How does the direction and length of the sine wave depend on p and q? Draw an illustrative figure on paper. Write down the explicit expression that can be found in the lecture notes. Complement the code (variable `wavelength`) accordingly.**

- <u>*Question 5:*</u> **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!**

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

1.4 Linearity

Define some rectangular shaped 128×128 pixel test images by

```
F = [ zeros(56, 128); ones(16, 128); zeros(56, 128)];
G = F';
H = F + 2 * G;
```

and display them with `showgrey`. Then compute the discrete Fourier transform of the images by writing

```
Fhat = fft2(F);
Ghat = fft2(G);
Hhat = fft2(H);
```

and show their Fourier spectra with

```
showgrey(log(1 + abs(Fhat)));
showgrey(log(1 + abs(Ghat)));
showgrey(log(1 + abs(Hhat)));
```

Also try to run the following

```
showgrey(log(1 + abs(fftshift(Hhat))));
```

and explain why the `fftshift` command is needed here. Another way of achieving the same effect is using the MATLAB function `showfs` located in the course library.

Question 7: **Why are these Fourier spectra concentrated to the borders of the images?**

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

Question 9: **What conclusions can be drawn regarding linearity?**

1.5 Multiplication

Try the following commands

```
showgrey(F .* G);
showfs(fft2(F .* G));
```

and explain the results. (The notation `F .* G` in MATLAB refers to point-wise multiplication of corresponding matrix elements.)

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!

1.6 Scaling

Define a test image

```
F = [zeros(60, 128); ones(8, 128); zeros(60, 128)] .* ...
    [zeros(128, 48) ones(128, 32) zeros(128, 48)];
```

and display it with `showgrey`. Determine the discrete Fourier transform and look at the magnitude with `showfs`.

Question 11: **What conclusions can be drawn from comparing the results with those in the previous exercise?**

1.7 Rotation

Activate a new figure with `figure`, but keep the Fourier spectrum of `F` in the first figure. Rotate `F` by for example $vinkel = 30°$ and show the results with

```
G = rot(F, vinkel);
showgrey(G)
axis on
```

Then calculate the discrete Fourier transform of the rotated image with

```
Ghat = fft2(G);
```

and display the results with `showfs`. To you recognize it? Finally, rotate the spectrum back by the same angle with

```
Hhat = rot(fftshift(Ghat), -vinkel);
```

and show the results by writing

```
showgrey(log(1 + abs(Hhat)))
```

Assemble the original images and their Fourier spectra into the same figure with the `subplot` command.

Question 12: **What can be said about possible similarities and differences?**

1.8 Rotational symmetry

Create a rotationally symmetric test image `F` with

```
[G H] = meshgrid(-64: 63, -64: 63);
F = G .^ 2 + H .^ 2 <= 128;
```

and display it with `showgrey`. The `meshgrid` command is often useful in MATLAB. If you are not familiar with the command, you ought to take a look at the `G` and `H` matrices with `showgrey`. Then show the Fourier transform with

```
showfs(fft2(F))
```

Question 13: **How does the rotational symmetry affect the Fourier spectrum and why?**

1.9 Translation

Activate another figure window, such that the Fourier spectrum of `F` can be kept on screen. Then translate the test image with

```
G = [F(:, 21:128) F(:, 1:20)]
```

and show the result

```
showgrey(G); axis on
```

Show also the Fourier spectrum of `G` with

```
showfs(fft2(G))
```

and the phase spectrum of `G` and `F` with

```
showgrey(angle(fft2(G)), 64, -pi, pi);
showgrey(angle(fft2(F)), 64, -pi, pi);
```

Question 14: **What similarities and differences can you observe?**

2 Information in Fourier phase and magnitude

Above we have primarily used the magnitude of the Fourier transform when we have visualized the transform as an image. This visualization technique is also what dominates in literature. As an illustration of the limitations of this way of visualizing, we will in this section devote ourselves to a bit of image manipulation, where we simply *replace* the power spectrum for a given image f with a power spectrum of the form [2]

$$|\hat{f}(\omega)|^2 = \frac{1}{a + |\omega|^2} \tag{6}$$

In the course library there is a function `pow2image` that performs this (nonlinear) function. (Take a look at the code by typing "`type pow2image`".)

Apply this function on for example the following images `phonecalc128`, `few128`, `nallo128` and study the results on screen (for very small values of $a \approx 10^{-10}$). Conclusions? As a comparison you should apply the function `randphaseimage` that keeps the magnitude of the Fourier transform, but replaces the phase information with a random distribution.

Question 15: **What information is contained in the phase and in the magnitude of the Fourier transform?**

---

3 Gaussian convolution implemented via FFT

3.1 Continuous Gaussian convolution

Gaussian convolution means that we convolve a given image f_{in} with a Gaussian kernel. In the continuous case the output image f_{ut} is given by

$$f_{ut}(x, y) = \int_{\xi=-\infty}^{\infty} \int_{\eta=-\infty}^{\infty} f_{in}(x - \xi, y - \eta)\, g(\xi, \eta;\ t)\, d\xi\, d\eta \tag{7}$$

where

$$g(x, y;\ t) = \frac{1}{2\pi t} e^{-(x^2+y^2)/(2t)}. \tag{8}$$

Correspondingly, in the Fourier domain

$$\hat{f}_{ut}(\omega_1, \omega_2) = \hat{g}(\omega_1, \omega_2;\ t)\, \hat{f}_{in}(\omega_1, \omega_2) \tag{9}$$

where $\hat{f}_{ut}$ and $\hat{f}_{in}$ are the Fourier transforms of f_{ut} and f_{in} respectively, and $\hat{g}(\cdot, \cdot;\ t)$ denotes the Fourier transform of the Gaussian function, that is

$$\hat{g}(\omega_1, \omega_2;\ t) = \int_{\omega_1=-\infty}^{\infty} \int_{\omega_2=-\infty}^{\infty} g(x, y;\ t)\, e^{-i(\omega_1 x + \omega_2 y)}\, dx\, dy = e^{-(\omega_1^2+\omega_2^2)\, t/2}. \tag{10}$$

3.2 Discretization

To discretize the Gaussian convolution, we may either choose to discretize the convolution operation (7) in the spatial domain, or proceed by discretizing the Fourier transform of the Gaussian function (10). In this exercise we will try both these methods.

[2]For continuous signals (and in the case $a = 0$) this form of Fourier spectrum can be derived as an idealized model of images that contain different types of image structures distributed on all scales. (The purpose of parameter a is simply to avoid division by zero.)

3.2.1 In the spatial domain

If we discretize the integral in (7) with the trapezoidal rule and set the step length to one (corresponding to a unit distance between neighboring image elements), we get

$$f_{ut}(x, y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f_{in}(x - m, y - m)\, g(m, n;\ t) \tag{11}$$

which corresponds to a discrete convolution with a sampled version of the Gaussian function. To perform this operation in practice, we may thus use either one of the methods below:

With spatial discretization and spatial convolution:

1. Generate a filter based on a sampled version of the Gaussian function.
2. Convolve the image with this filter using the embedded MATLAB-function `conv2`.

With spatial discretization and convolution via FFT:

1. Generate a filter based on a samples version of the Gaussian function.
2. Fourier transform the original image and the Gaussian filter.
3. Multiply the Fourier transforms.
4. Invert the resulting Fourier transform.

Of these methods the first method with filtering in the spatial domain is probably the most straightforward one. Such explicit convolution in the spatial domain generally works very well, if one exploits the fact that the Gaussian function is separable (this is however not done with the MATLAB-function `conv2`). This method is particularly suitable when the variance of the Gaussian kernel is small, and the kernel is truncated to a small filter of compact support. Since the computational work grows linearly with the number of non-zero elements in the filter, this method can be computationally expensive for large values of t, in particular if the separability is not exploited.

The other method, with linear filtering in the frequency domain, has the characteristic that it is essentially independent of the size of the Gaussian kernel. (In practice, you always generate a Gaussian kernel of the same size as the original image to be filtered; we assume here that this is a power of two and large enough for the shape of the Gaussian kernel to be kept within the given image size.) Specifically, with the image processing system we use, the second method is facilitated by an efficient implementation of FFT available in MATLAB.

For the above mentioned reasons the last method is in this case the preferable one, even if it involves more steps.

3.2.2 In the Fourier domain

If the convolution operations still are performed via a Fourier transform we may merge steps 1 and 2 of the last method, by directly expressing the filter in the Fourier domain. The method will then have the following structure:

Gaussian convolution in the Fourier domain:

1. Fourier transform the original image
2. Multiply the transformed image with the Fourier transform of the Gaussian kernel.
3. Invert the resulting Fourier transform.

However, when we perform these operations in practice there are some technical difficulties to consider:

- Firstly, the continuous definition of the filter (expressed in the whole frequency plane) must be translated to the limited part of the frequency plane that is covered by the discrete Fourier transform. With reference to section 1.2 we know that the discrete frequency variables u are projected to the continuous frequency variables ω_D according to

$$\omega_D = 2\pi \frac{u}{N}. \tag{12}$$

  Furthermore, the unlimited frequency plane is restricted to a limited interval $[-\pi, \pi]^2$. In digital signal processing non-linear transformations, known as frequency warpings, are often used for this purpose. For simplicity we will instead just use truncation.

- Secondly, we need to place the origin in the discrete Fourier transform, so as to cover an origin symmetric interval in the argument $\omega = (\omega_1, \omega_2)$ of the Fourier transformed Gaussian kernel.

$$\hat{g}(\omega_1, \omega_2;\ t) = e^{-(\omega_1^2+\omega_2^2)\,t/2}. \tag{13}$$

  In MATLAB you may simply generate a coordinate system with the command `meshgrid` and move the position of the origin with the command `fftshift`.

3.3 Filtering procedures

Write two functions that perform filtering in the Fourier domain!

Write one `Matlab`-procedure

```
gaussffta(pic, t)
```

that, **using the fast Fourier transform**, convolves the image `pic` with a two-dimensional Gaussian function of arbitrary variance t via a discretization of the Gaussian function in the spatial domain.

And write a second `Matlab`-procedure

```
gaussfftb(pic, t)
```

that performs the same Gaussian smoothing based on a definition of the filter in the Fourier domain.

Proposed test procedure: When you perform this operation you might want to make sure that your convolution procedure has an approximatively correct behavior by analyzing its impulse response. Hence, generate the impulse response explicitly

```
psf = gaussfft(deltafcn(128, 128), t);
```

for different values of t (e.g. t = 0.1, 1.0, 10.0 and 100.0). Look at the result and compute also the (spatial) covariance matrix of the Gaussian function with

```
variance(psf)
```

Question 16: **How does the result correspond to the ideal continuous case, for which the covariance matrix is t multiplied by the identity matrix?**

$$C(g(\cdot,\cdot;\ t)) = t \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{14}$$

If you find it difficult to get this to properly work, a routine `discgaussfft`, that convolves an image with the discrete version of the Gaussian kernel, is provided in the course library. This discrete kernel has (among other things) the characteristic that its discrete variance is exactly equal to t.

Question 17: **Show the impulse response and variance for the above mentioned t-values. What are the variances of your discretized Gaussian kernel for $t = 0.1$, 1.0, 10.0 and 100.0?**

Question 18: **Can you note any differences between the results of `gaussffta` and `gaussfftb`? Why are the results different from or similar to the estimated variance? Lead: First consider the results for small t values.**

Question 19: **Convolve a couple of images with Gaussian functions of different variances (like $t = 1.0$, 4.0, 16.0, 64.0 and 256.0). What effects can you observe?**

---

4 Smoothing

Load the image `office256` from the image database of the course

```
office = office256;
```

and create two noisy images using the MATLAB-functions `gaussnoise` and `sapnoise`. (These functions exist in the course function library):

```
add = gaussnoise(office, 16);
sap = sapnoise(office, 0.1, 255);
```

Show the images on screen with `showgrey`. What kind of noise do they represent? (Use the `help` command, if you are unable to guess.)

Exercise (#4.1): Try to reduce the noise in the images `add` and `sap` with

- Gaussian smoothing (using either function `gaussffta` or `gaussfftb` that you wrote in previous exercise, alternatively the function `discgaussfft` available in the course library)
- Median filtering (using the function `medfilt` in the course library)
- Ideal low-pass filtering (using the function `ideal` in the course library)

Try suitable values of the parameters of each filter respectively (standard deviation for the Gaussian filter, window size for the median filter, cut-off frequency for the ideal low-pass filter.

Question 20: **What conclusions can you draw from comparing the results of the respective method?**

Question 21: **What positive and negative effects does each filter type have?**

Question 22: **What respective similarities and differences can you observe between the different filters?**

Question 23: **How do the results depend on the filter parameters?**

Compute impulse responses and Fourier transforms of the ideal low-pass filter, using suitable values of cut-off frequency. Illustrate the results partly as grey-level images and partly as cross section graphs.

Question 24: **What are the effects you observe based on these illustrations?**

---

5 Contrast improvement (sharpening)

Create a 3×3 pixel filter mask `laplace` that approximates the Laplace operator. Compute and inspect the result of this operator applied to the image.[3]

```
laplaceblocks = conv2(blocks1 laplace, 'valid');
showgrey(laplaceblocks);
showgrey(conv2(add, laplace, 'valid'));
showgrey(conv2(sap, laplace, 'valid'));
```

If the resulting image 'laplaceblocks' is of low contrast, you will have to saturate some of the high values in the histogram!

What can be said about the noise sensitivity of the Laplace operator? Try to sharpen image `blocks1` by subtracting a suitable constant multiplied by the Laplace operator. That is, let

```
stripblocks = conv2(blocks1, [0 0 0; 0 1 0; 0 0 0], 'valid');
sharpblocks = stripblocks - koefficient * laplaceblocks;
```

for a suitable value of *koefficient*. (The convolution operation above has the effect that `sharpblocks` gets the same size as `laplaceblocks`.)

Finally, generate a sharpening mask `sharpmask` so that the sharpening operation can be performed in a single step, i.e. so that the operations above can be replaced by

```
sharpblocks = conv2(Blocks, sharpmask, 'valid');
```

Question 25: **Why can you replace the whole operation above with a single filter?**

- Try different values of *koefficient*.
- Verify that the contours really get sharper by selecting a suitable row or column in the images `stripblocks` and `sharpblocks`, and draw these with `plot` before and after sharpening. (Naturally, you should assemble the results with `subplot` to facilitate interpretation.)

[3] The argument `'valid'` considers the size of the filter mask and reduces the image size with an enough number of pixels for the results to depend only on values within the image. This procedure is suitable in this case, since the values of the Laplace operator can be significant around the boarder of the image.

Question 26: **How is the Laplace operator affected by noise?**

Laboration 2 in DD2423 Image Analysis and Computer Vision

. . . .
Student's personal number and name (filled in by student)

. . . .
Approved on (date) Course assistant