Name: Per Arne Kjelsvik email adres: p.a.kjelsvik@student.utwente.nl
Student id: s2049201
Educational Program: EEMCS

Exercise 1: Basics of convolution and Fourier transform

Matlab Preamble

- During development of your m-code, you might want to inspect images and image-like data (such as Fourier magnitude spectra) on your screen for inspection. This is done as follows:
```
figure;                % create a figure window
imshow(im);            % write the image to the current graphical object
```
- If the data is out of range (for type doubles: not between 0 and 1), and it is an intensity (grey) image, you can adapt the range of screen by (this is also useful if the range is so small that contrasts are hardly visible):
```
imshow(im,[]);         % adapt the range
```
- For color images, this does not work. If you want to adapt the scale, you use:
```
imshow(mat2gray(im));% adapt the range
```
- For the report, you have to write the images to file first. This is done as follows:
```
imwrite(im,'fname.jpg');                    % without adapting the range
imwrite(mat2gray(im),'fname.jpg');          % with adaption of the range
```
- For non-image like data, i.e. graphical data such as plots, you have to save the whole figure window to file. This is done as follows:
```
print -r300 -dpng fname.png    % save the current figure window as a png
                               % file. The resolution ,300 dpi, is an
                               % example.
```

The report and the m-code

You create a **single** m-file for all questions. The code must be such that **all** code can be executed by a single click on the "RUN" button (with the green arrow). Put the following line between each question:
```
%% Question 1  (etc)
```

You copy-and-paste all your code to the end of the report. Make sure that the line length does not exceed 95 characters since otherwise the code does not fit the width of a page.

Questions

The image, stored in the file `car_dis.png`, is distorted. The objective is to apply a filter that removes the distortion without affecting the original image.

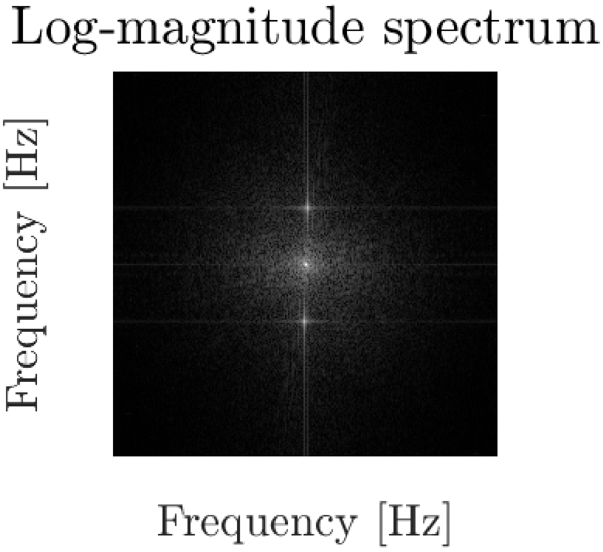
1. Write and execute an m-file that:
   - clears the workspace and closes all possible figure windows (use `clear variables` and `close all`, respectively)
   - reads the image from file (`imread`), converts it to an intensity image of type double (`im2double`), and displays the image in a figure window on the screen (`imshow`).
   - calculates the Fourier transform and shows the log-amplitude spectrum as an image on the screen. Be sure to do so in a controlled manner: (i) the origin should be in the centre, and (ii) the logarithmic scale maps the magnitudes to intensities such that the interesting part of the range of magnitudes is displayed well. Include a title, and the correct x- and y labels to the axes. Write the resulting figure window as a png image[1] to file (use: `print -r150 -dpng Imlogmag.png`). **Do not use the copy-and-paste option, nor the 'save as' of the figure window as the result is not reproducible (it will depend too much on your screen resolution).**

Note: Do not copy-and-paste text from this pdf into your matlab editor: some fonts in the pdf might slightly differ from the ones that the editor expects

[1] The PNG (Portable Network Graphics) is a lossless image file format. For image analysis, a lossless coding is often preferred as a lossy encoding, such as JPEG, might induce loss of information. Another popular format is TIFF which supports both lossless and lossy encoding.

Insert the car_dis.png

Insert the log-amplitude spectrum (Imlogmag.png)



2.a What kind of symmetry do you observe in the spectrum and how can you explain this?

> fftshift swaps the 1st and 3rd quadrants, as well as the 2nd and 4th. We see before shifting the periodicity property of the Fourier transform. Also, we say that for 2D the real part is symmetric (hermitian/complexconjugate symmetry) while the imaginary part is anti-symmetric. This is what allows the center point symmetry of the log-amplitude shifted image.

2.b Suppose that the size of the input image `im` is N x M. How large will be the fft `IM` of this image? The function `fftshift` shifts the origin of the u,v frequency domain to the centre. At which row and column index is the origin of this domain (that is, u=0, v=0) located in the matlab array `IM`, directly after application of `fft2`, and after application of `fftshift`. Give matlab expressions and the numerical result for this particular case:

directly after `fft2`:	`column =`	`IM(1,1)      %Origin of fft2 is always in element (1,1)`
	`row =`	`IM(1,1)      %Origin of fft2 is always in element (1,1)`
after `fftshift`:	`column =`	`find(IM(1,1) == IM_shift);       %Origin in (129, 129)`
	`row =`	`find(IM(1,1) == IM_shift);       %Origin in (129, 129)`

2.c Suppose that the pixel size of the image `im` is $\Delta \times \Delta$ The distortion in the image looks like an harmonic function. By using the cursor of the figure window, created in question 1a, determine, i.e. measure, the wavelength of this harmonic function (expressed in pixel size Δ; that is the result could be, for instance, `4*Delta`). Give the frequency ρ of this harmonic, also expressed[2] in pixel size Δ:

`lambda =`	`7*Delta`
`rho =`	`1/lambda  %1/(7*Delta)`

2.d Suppose that the distortion has an horizontal orientation, so that in Cartesian coordinates we have $u = 0$ and $v = \rho$. If the image size is NxM, and its fft `IM` is centred with `fftshift`, at what row and column index one of the dominant frequency components of the distortion could be expected in the log-magnitude spectrum? Give matlab expressions and the numerical result for this particular case:

`col =`	`(1/2)*N+1                %128, no horizontal offset (u=0)`
`row =`	`(1/2)*N +/- rho*M        %+/- to indicate two points, 92 and 166`

---

[2] Hint: useful information is on the slide entitled: “Discrete Fourier transform” of the lecture sheets.

2.e The image `car_dis.png` has a finite size. What effect does this finiteness has on the log-magnitude spectrum, and then especially on the part of the spectrum that is associated with the distortion?

> The effect is spectral leakage, which we can see in the log-magnitude figure: There are thin lines extending out from the distortions (leakage due to the image being finite).

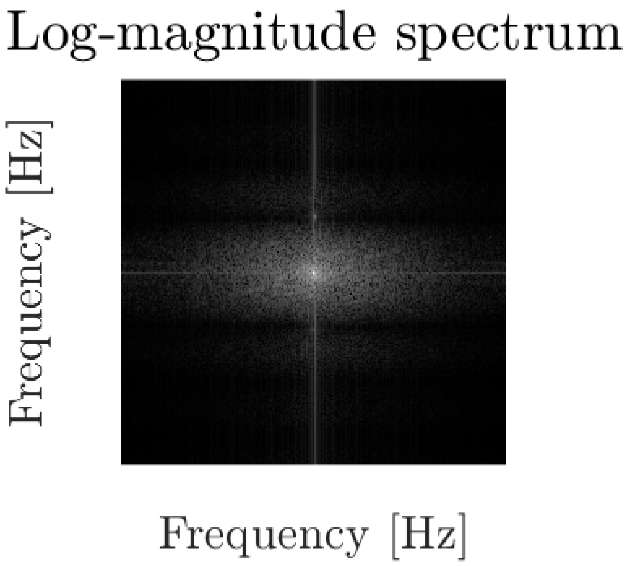
2.f And what is the explanation of this effect (why does the finiteness of the image cause this effect)?

> This is caused by a digital Fourier transform of non-periodic data. FFT has finite length (the image is finite), which produces a window on the data. A window in the time domain is convolution in frequency domain with the transform of the window. A rectangular window's transform is a Sinc function (sin(x)/x), which has infinite width. Therefore the ripples of the since function will show up as "leakage" far from any spectral peak that is not perfectly periodic in the FFT's length.

3. Extend the m-file with code that filters the image with a rectangular PSF. In Matlab the corresponding filter is called ‘`average`’. Select the horizontal and vertical sizes of the PSF such that the distortion is suppressed, but the image details are preserved as much as possible. (use `fspecial` and `imfilter`). Write the resulting image to file in JPEG format, e.g. `imwrite(imfil,'IMfil.jpg')`; Calculate, display and store the log-magnitude spectrum of the filtered image, as was done in Q1.

Insert the filtered image here:

Insert the log-amplitude spectrum of the filtered image:



3.a What is the exact size of the PSF and motivate its specific width and length.

`Nhorz =` 1

`Nvert =` 7

> PSF is often space variant (differs at the border of the image), and depends on wavelength. We measured the distortion wavelength (assuming it is a harmonic function) to have lambda=7*Delta and rho=1/(7*Delta). This motivates us to choose the PSF to be the same size as the distortion. We have almost no vertical distortion, so that's why we set Nvert as minimal as possible

3.b Explain what this filter should suppress in the spectrum. Does the filter also affect the informative part of the image? And is this inevitable, or could another choice of the PSF circumvent this?

> Wavelength frequency of the harmonic function of the distortion. Yes, the filter affects the informative part, often resulting in a ringing effect appearing near strong edges for example. Theoretically not inevitable, but practically the PSF of an optical imaging system cannot be exactly known due to insufficient information of measurement.

3.c What do you observe at the border of the image, and how do you explain this?

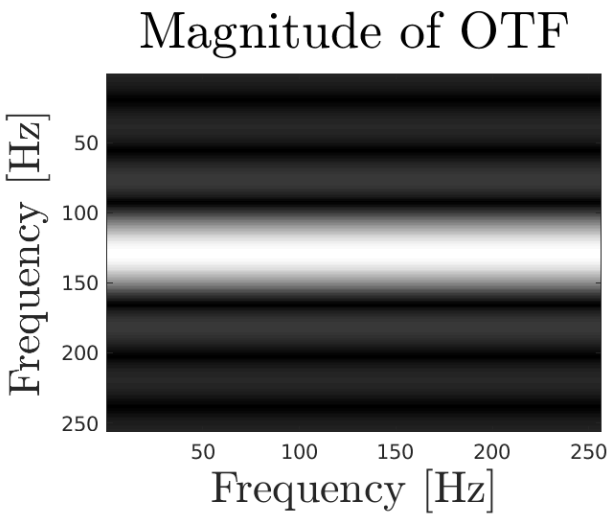
When you filter an image, zero padding can result in a dark band around the edge of the image since we get a transition from the image edge to 0. So the border we see here comes from imfilter. Using options like 'replicate' (border replication) helps with this, since then we try to extend the available information from the edges to reduce the black band around the edges.

3.d Add one or option(s) to `'imfilter'` (refer the documentation of this function) such that the result is as natural as possible (the natural result would be obtained if image data outside the image plane would have been available). Note: in future, use these option(s) whenever you want to use `imfilter`.

Option(s) used: `replicate`

4. Extend the m-file with code that calculates the otf associate to the filter found at Q3 and stores this in an array `OTF`. For that, use the function `psf2otf`. See the documentation. If you use this function with only one argument being the psf matrix, then the size of `OTF` will be equal to the one of the psf matrix (you can inspect that easily). With an additional option, you can calculate the otf at a much higher resolution. Select the second option such that the resulting array has the same size as the image in Q3. Next, shift the origin to the centre of the graph, and show the magnitude of `OTF` as an intensity image in a figure window. Apply a linear scale to map the magnitudes to intensities. Don't forget to add the title and the labels at the axes.

Insert the magnitude of the OTF image:



4.b Inspect the imaginary part of the otf , and explain the result.

`max(abs(imag(OTF(:)))) =` `0`

Explanation:

The OTF is perfectly symmetric (can be inspected by plotting it). Had there been any asymmetry, there would be some imaginary data, and we would not get 0 as the result.

4.b In Q2 you have identified the dominant frequency (and corresponding row- and column-indices) that can be associated with the distortion. Use this information, to extract from the OTF the attenuation factor of this particular frequency.

Attenuation factor is: `0.0119`

4.c Which type of this OTF represent (low pass, high pass, band pass, high emphasis, etc)?

Type of filter: `low pass filter`

4.d Is this filter rotational invariant, i.e. is the OTF rotational invariant, and would that be a desirable property in this application?

Rotational invariant (yes/no): `no`

Desirable (yes/no): `yes`

Why, or why not:

Our OTF can only handle horizontal distortions, while this is pretty good for our application, a rotational invariant OTF would better deal with the slight angle of the distortion.

5. Instead of spatial convolution we can filter in the frequency domain. The strategy will be to define a filter transfer that blocks a range of frequencies that can be associated with the distortion. All other frequencies will be passed without attenuation.

5.a Suppose that `(row,col)` are the row and column indices of one of the dominant frequency as found in Q2d. We define a rectangular area around this dominant frequency. Suppose the size of the rectangle is set to `s x s`, where `s` is an odd integer. In matlab, a rectangular range of indices can be represented by `rstart:rend`, and `cstart:cend`. These four parameters must be chosen such that `(row,col)` is in the centre of this rectangle. Give matlab expressions for `rstart`, `rend`, `cstart`, and `cend`:

`rstart =` `ceil(row-roff-s/2);           % S=9, row=129 from before, roff=38 (see full code)`

`rend =` `floor(row-roff+s/2);`

`cstart =` `ceil(col+coff-s/2);           % col=129 from before, coff = 1`

`cend =` `floor(col+coff+s/2);`

5.b An OTF matrix H with the same size of the image, and filled with 1's everywhere except for the frequencies of the rectangle can be created with[3]:

```
H = ones(size(im));
H(rstart:rend,cstart:cend) = 0;
```

However, this code will not yield an OTF with the needed symmetry. Extend the code such that the symmetry requirements are met.

Code:

```
[N,M]= size(im); s = 9; H = ones(size(im));
coff = col - 1/2*N; roff = ceil(row -(1/2*N-1/7*N));    % col and row offset of distortions

rstart  = ceil(row-roff-s/2);  rend    = floor(row-roff+s/2);
cstart  = ceil(col+coff-s/2);  cend    = floor(col+coff+s/2);

H(rstart:rend, cstart:cend) = 0; H(rstart+2*roff:rend+2*roff, cstart-2*coff:cend-2*coff) = 0;
```

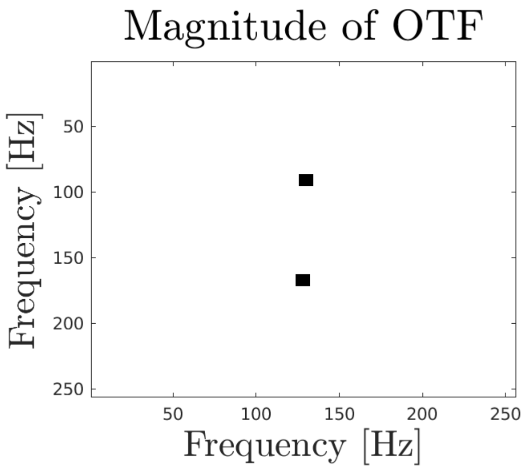
5.c Apply the filter. That is, perform a pixel-by-pixel-multiplication between H and the FT of the original image. Next, calculate the inverse Fourier transform of the resulting image and file this in your report. Very small imaginary parts (that is smaller than 10^{-14}) of the resulting image are due to numerical round-off errors. These parts may be removed. If the imaginary parts are substantial, then the matrix H did not comply with the symmetry requirements. Suppose that `imresult` is the final resulting image, then execute:

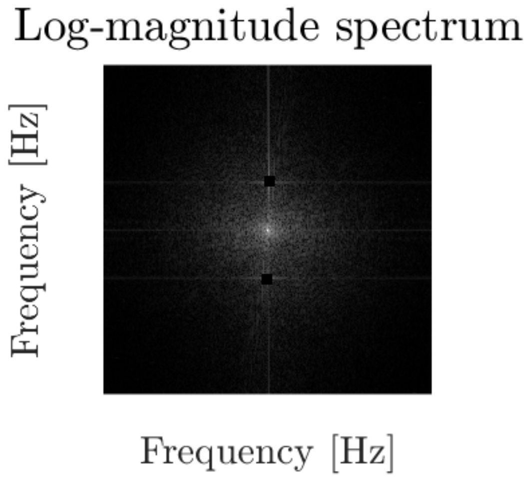
`max(abs(imag(imresult(:))))` = `0`

[3] Experienced Matlab programmers always try to avoid for-loops. Here, the colon operator (:) very useful. It produces concise and comprehensible code. Moreover, it may speed up Matlab considerably..

5.d Make graphical representations of the new OTF and the log-magnitude spectrum of the newly filtered image (as you did before).

The magnitude of the found OTF image: The log-magnitude of the newly filtered image





Show (for comparison) the distorted image and the newly filtered image:

6 In the resulting image, there will be some artefacts at the border of the image. These artefacts depend on the size of the chosen rectangle. Describe how, and why:

> We are using a square window which captures the center part of the distortion (in the image above I used s=9), but there is still the spectral leakage from the distortion. The leakage is (as described earlier) due to the finite size of the image and the image edges. If we increase the size of the rectangle, we will filter more of the distortion out, which makes more of the borders in the edges disappear.

7 Describe what the effects of the filter are on the informative part of the image.

> In return for increasing filter size, more of the information in the image will be filtered too so that we get a blurry photo with strong edge ringing effects (as discussed earlier as well). This can bee seen in the spikes of the wheel for example (with larger filter), and the front edge of the barrel etc.

Make sure the m-code fits within the PDF-margin (also the added comments)!

m-code:

```
% Per Arne Kjelsvik s2049201 - Exercise 1 NB!! Must open in Adobe Reader to see all code (scroll) !!
%% Question 1
clc
close all
clear variables

im = im2double(imread('car_dis.png'));  % Read image, convert to double
IM = fft2(im);                          % Element (1,1) is origin
IM_shift = fftshift(IM);                % Element (129,129) is origin

figure;
imshow(log(abs(IM_shift)+10), []);      % Plot log of magnitude, +10 for
title('Log-magnitude spectrum', ...     % better visualization
    'FontSize', 26, 'Interpreter', 'latex');
xlabel('Frequency [Hz]', 'FontSize', 22, 'Interpreter', 'latex');
ylabel('Frequency [Hz]', 'FontSize', 22, 'Interpreter', 'latex');
print -r150 -dpng Imlogmag.png

figure;
imshow(im)                              % Display the image

%% Question 2
% b) Matlab expression and numerical result of index of origin
[row,col]= find(IM(1,1) == IM_shift);   % Find origin in shifted spectrum

% c) Wavelength of harmonic function
Delta   = 1;
lambda  = 7*Delta;
rho     = 1/lambda;

%% Question 3
PSF = fspecial('average', [7 1]);       % 'average' for averaging filter
imfil = imfilter(im,PSF, 'replicate');  % 'replicate' used to handle edges

IMfil = fft2(imfil);                    % FFT of filtered image
IMfil_shift = fftshift(IMfil);          % Shift spectrum to center

figure;
imshow(log(abs(IMfil_shift)+2), []);
title('Log-magnitude spectrum', ...
    'FontSize', 26, 'Interpreter', 'latex');
xlabel('Frequency [Hz]', 'FontSize', 22, 'Interpreter', 'latex');
ylabel('Frequency [Hz]', 'FontSize', 22, 'Interpreter', 'latex');
print -r150 -dpng IMfillogmag.png

figure;
imshow(imfil);
imwrite(imfil,'imfil.jpg');

%% Question 4
OTF = psf2otf(PSF,[256,256]);
OTF_shift = fftshift(OTF);

figure;
imshow(abs(OTF_shift));
imagesc(abs(OTF_shift),[0 1]); colormap(gray);
title('Magnitude of OTF', ...
    'FontSize', 26, 'Interpreter', 'latex');
xlabel('Frequency [Hz]', 'FontSize', 22, 'Interpreter', 'latex');
ylabel('Frequency [Hz]', 'FontSize', 22, 'Interpreter', 'latex');
print -r150 -dpng OTF.png

% b) Inspect the imaginary part of the otf
max(abs(imag(OTF(:))));                 % Imaginary part equal to 0

%% Question 5
s       = 9;                            % Size of rectangular side, odd
[N,M]   = size(im);
H       = ones(size(im));
coff    = col - 1/2*N;                  % Column offset to distortion
roff    = ceil(row -(1/2*N-1/7*N));     % Row offset to distortion

% a) Matlab expressions for rstart, rend, cstart, cend
rstart  = ceil(row-roff-s/2);
cstart  = ceil(col+coff-s/2);
rend    = floor(row-roff+s/2);
cend    = floor(col+coff+s/2);

% b) Extend code so that symmetry requirments of H is met
% Could have used H.*flip(H) for easier calculation, but chose to do as
% below for perfect symmetry
H(rstart:rend, cstart:cend) = 0;
```