Lab 6

Image processing – 2D signals

6.1. Problems for home study

Similar problems will be given at entry test
1) Calculate a 2D DFT of a 16×16 pixel image:

a) totally black (all zeros)
b) totally white (all ones)
c) with only one white pixel at $(0, 0)$ coordinates
d) with eight vertical white strips and eight black stripes (stripe width 1 pixel)
e) the same – with horizontal stripes

Assume white is one, black is zero. **Hint:** Note that DFT is separable by dimensions – one can calculate 16 vertical transforms first, then 16 horizontal ones (or the other way round).

2) What image is represented by a spectrum with only one nonzero sample at $(0, 0)$? What value should the sample have to attain maximum value in the image of "white" i.e. 1? Beware of the scaling coefficient in IDFT!

3) An image (2D signal) with 16×16 pixels, where all pixels are black except a white pixel at $(8, 8)$ has been filtered with a linear filter defined by impulse response:

```
lp2=[ 1 1 1;...
      1 2 1;...
      1 1 1];
```

Assume the filter is non-causal, zero-phase, and:

a) tell which sample in the above is $h(0, 0)$
b) calculate the result of filtering
c) prove that the phase response is actually zero

4) Sketch the result of filtering these signals with a 5-sample median filter (the signals are one-dimensional):

a) unit impulse $\delta(n)$
b) unit step $u(n)$
c) unit step with added impulse: $u(n) + \delta(n - 2)$
d) unit step with a hole in it: $u(n) - \delta(n - 3)$

6.2. Experiments

Capturing an image and preparing it for processing

Hint: You will do the following each time you start experimenting with a new image. You'll also need to do the **"preparing"** part when using an image from a file or from the Web.

Capturing an image

In order to capture an image, you will use the USB camera which is attached to your computer.

You may use any Windows program that captures an image to file (then you'll read the file into Matlab), but the easiest way is to use VCAPG2 driver and capture the image straight into Matlab variable.

```
» myimageRGB=LCPS_getsnapshot();% calls VCAPG.MEXW64 loadable function
» imshow(myimageRGB);
```

It may happen that the driver hangs – then call `LCPS_getsnapshot(1);` to reset it.

If you use the image from a file, read the image using `imread()` – you may also drag the image to Matlab's "workspace" window.

```
» myimageRGB=imread('PathToTheFile.png');
```

Preparing an image for processing

- For a colour image you will get a 3D matrix $M \times N \times 3$ with R, G, B channels; either choose one channel, or flatten the image to greyscale with `rgb2gray`, in order to get a matrix $M \times N$.
  ```
  » myimageGray=rgb2gray(myimageRGB);
  ```
- JPG and PNG images are read as integer matrices (`uint8`). For calculations we need a floating-point variable; do the cast:
  ```
  » myimage=double(myimageGray);
  ```
  Hint: Whenever you read an image in the following exercises, you will need to repeat the above operations.
- Display the image in grayscale (`gray` instead a default colormap with false colors`jet`):
  ```
  » imagesc(myimage);colormap(gray);colorbar;
  ```

6.2.1. Spectrum of a 2D signal

Remark: In code examples we will use lowercase variable names when we mean signal, uppercase when we mean spectrum. We encourage student to adhere to this setup also in his/her own code.

Spectra of simple images

- Create an "all-white" image with size 32×16 pixels
  ```
  » img1=ones(32,16);
  ```
- Display image, calculate and display its 2D-spectrum
  ```
  » imagesc(img1)
  » IMG1=fft2(img1);
  ```

```
» figure;imagesc(abs(IMG1));
```

Note the size of the table (matrix) with spectrum, the value of its maximum, indices of the maximum in the table.

Note

Hint: Use `colorbar` to see how the values are mapped to colors.

The spectrum is easier to interpret when zero frequency is in the middle of plot. Use `fftshift` which exchanges the four quarters of a matrix (or left and right halves of a vector – useful for 1D spectra).

```
» imagesc(fftshift(abs(IMG1)));
```

Hint: `fftshift` used twice comes back to the original position of quarters – you will need to use this before restoring the image from the `fftshift`'ed spectrum.

Answer

Answer in the report: What variables are represented on two axes of the spectrum plot? What is the correct scaling (labeling) – give one out of several possibilities.

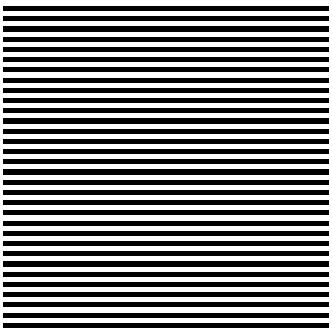

Fig. 6.1: Horizontal lines with high frequency.

Fig. 6.2: Horizontal lines with low frequency.

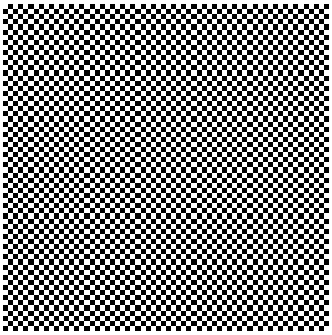

Fig. 6.3: Chequerboard with high frequency.

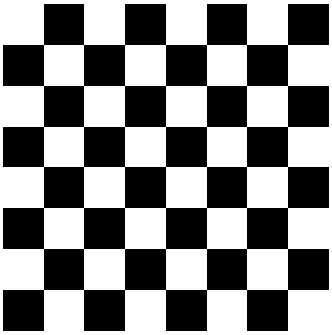

Fig. 6.4: Chequerboard with low frequency.

Capture or generate images like Fig. 6.1 – Fig. 6.4.

If generating, choose parameters from table 6.1. If capturing, try to change the distance so that you obtain images with high and low frequency as shown at Fig. 6.1 – Fig. 6.4.

How to generate a "lines" image? Start with two strips:

```
» M=...
» N=...
» MS=...%size of the strip
```

Table 6.1

Table number.	1	2	3	4	5	6	7	8	9	10	11	12	13
Rows in the image M	32	64	48	16	32	64	60	40	24	32	48	64	16
Columns in the image N	16	32	48	48	32	24	60	48	64	16	36	48	32
Size [px] of the strip/square	4	8	1	2	8	4	5	2	6	1	3	1	4

```
» strip1=ones(MS,N)
» strip0=zeros(MS,N)
```

Then repeat them `M/2/MS` times:

```
» img2=repmat([strip1;strip0],M/2/MS,1);
```

How to generate "chequerboard"? Start with two squares:

```
» sqr1=ones(MS,MS)
» sqr0=zeros(MS,MS)
```

Then make a 2x2 board and repeat it `M/2/MS` by `N/2/MS` times:

```
» img2=repmat([sqr1, sqr0; sqr0, sqr1],floor(M/2/MS),floor(N/2/MS));
```

In both cases view the 2D signal and its 2D spectrum. For spectra use `fftshift`. Use `colorbar` to understand represented values.

Answer

Answer in the report:

– Which components are visible in the spectrum? Use the term *spatial frequency (horizontal or vertical)*.
– Describe the difference between spectrum of "lines" and "chequerboard".

If you prefer a sketch over words, do a sketch with answers marked on it.

Spectrum of a live image

Prepare an image

If possible, get an image from a camera (best idea). Otherwise, you may find an interesting photo on the Web. If more interesting images are inaccessible, use ready-made `Lenna.png`.

Choose a photo subject by yourself – but we encourage students to choose different subjects, so that you may see the different results among your neighbors: a face, the room, a desk scene, a scene outside the window....

Image spectrum

Calculate and view the spectrum; remember to use `fftshift` in order to put zero frequency in the middle.

Answer

Answer in the report: Which frequency has a dominating magnitude? Which component of the image it represents?

You can see that the spectrum displayed in linear scale is domianted by one strong component, which makes other components barely visible.

Display the spectrum in a logarithmic (dB) scale.

> Wherever you convert the data to dB, consider if you take the magnitude (or magnitude ratio) or power (or power ratio, squared magnitude, variance....) In the former case use $20 \cdot \log_{10}(x)$, in the latter – $10 \cdot \log_{10}(x)$.

Hint: Matlab function `dB` returns `20*log10(x)` by default.

Answer

Answer in the report: Describe major difference between spectrum of a live image and spectra of Fig. 6.1 – Fig. 6.4.

6.2.2. Two-dimensional filtering

Low-pass filter pass only the low frequencies – the image is smoothed, sharp features are lost:

```
lp1=[ 1, 1, 1;...
      1, 1, 1;...
      1, 1, 1];
```

```
lp2=[ 1, 1, 1;...
      1, 2, 1;...
      1, 1, 1];
```

```
lp3=[ 1, 2, 1;...
      2, 4, 2;...
      1, 2, 1];
```

```
lp4=[ 1, 1, 2, 1, 1;...
      1, 2, 4, 2, 1;...
      2, 4, 8, 4, 2;...
      1, 2, 4, 2, 1;...
      1, 1, 2, 1, 1];
```

High-pass filters enhance sharp, small features:

```
hp1=[ -1, -1, -1;...
      -1,  9, -1;...
      -1, -1, -1];
```

```
hp2=[  0, -1,  0;...
      -1,  5, -1;...
       0, -1,  0];
```

```
hp3=[  1, -2,  1;...
      -2,  5, -2;...
       1, -2,  1];
```

As the image pixels are represented by positive numbers, the subtraction in the filter equation may give invalid (negative) numbers. The decision what to do with them depends on the interpretation – possibilites are: truncate at zero, normalize by shifting up, use absolute value... When the image is displayed with `imagesc`, it is automatically scaled and shifted so that the minimal value is black and maximal – white; this is not necessarily what you want!.

Edge-detection filters are performing subtraction of shifted copies of the image, to approximate the calculation of partial derivatives in a given direction.

Examples of different edge-detection filters:

```
edg1=[0,  0,  0;...
     -1,  1,  0;...
      0,  0,  0];

edg2=[-1,  0,  0;...
       0,  1,  0;...
       0,  0,  0];

edg3=[-1,  1,  1;...
      -1, -2,  1;...
      -1,  1,  1];

edg4=[-1, -1,  1;...
      -1, -2,  1;...
       1,  1,  1];

edg5=[0, -1,  0;...
     -1,  4, -1;...
      0, -1,  0];

edg6=[-1,  0, -1;...
       0,  4,  0;...
      -1,  0, -1];

edg7=[1,  2,  1;...
      0,  0,  0;...
     -1, -2, -1];
```

If you want to detect edges in all directions, you may combine results of different filters – e.g. by adding absolute values of results.

Image filtering

Table 6.2

Table no.	1	2	3	4	5	6	7	8	9	10	11	12	13
LP filter	1	2	3	4	1	2	3	4	1	2	3	4	1
HP filter	1	2	3	1	2	3	1	2	3	1	2	3	1
Edge filter	1	2	3	4	5	6	7	1	2	3	4	5	6

- Filter your live image using filters from the previous section, selected according to the table 6.2. Note coefficients of each filter used.

Note

Use `y=filter2(h,x)` to perform 2D filtering.

> The `filter2` command implements a noncausal filter – zero delay corresponds to the central element of `h`.

- Display the filtered image; find its maximum and minimum value. **Hint:** `max(max(y))` and `min(min(y))` will find the answer for a 2D matrix `y`.

Answer

Answer in the report: (for each filter)

– Describe the filtering result with own words (if you prefer, sketch it).
– How did the max/min value change? Remember to use `colorbar`, as `imagesc` displays a scaled value image.
– How to modify filter coefficients to keep the result in the original range of values?

Frequency response of a filter

Note

Choose one of the filters (LP, HP, or edge), and investigate it further.
Note the type of choosen filter

- Display filter frequency response (substitute your filter impulse respose for `h`)

» `freqz2(h)`

- Display the 2D spectrum of the image before and after filtering

  Answer in the report: Describe the filtering effects in two ways: using the term "horizontal/vertical frequency", and summarizing the visual effects.

Answer

6.2.3. Edge detection – miniproject

In a practical application of image recognition – e.g. in robotic vision – it is frequently necessary to find edges in order to recognize shapes of objects.

Here you will use edge-detection filters to enhance edges.

- Read an image with strong lines – take a photo, or (plan B) read `'bariera.JPG'`.

  For a color photo, convert it to grayscale (as in 6.2.1).
- Use different edge-detaction filters from the previous section and combine their results to get an image with strong edges.
- Choose a threshold value and do thresholding (example result – – Fig. 6.5).

  Hint: Use construct `Ab=boolean(A>threshold)` will return a binary matrix with the same size as `A`, having ones where the logical expression is true.

  Display the result of processing – if you don't like it, repeat the process with different filters or different way of combining and thresholding.

```
figure(1)
imagesc(BW); colormap(gray); colorbar
```

Note

Note the choices which give best results (in your opinion).

- *Extra* Finally you may use Hough transform to identify straight lines (we did not study this transform – just use it; if you are curious, read about it in Wikipedia).

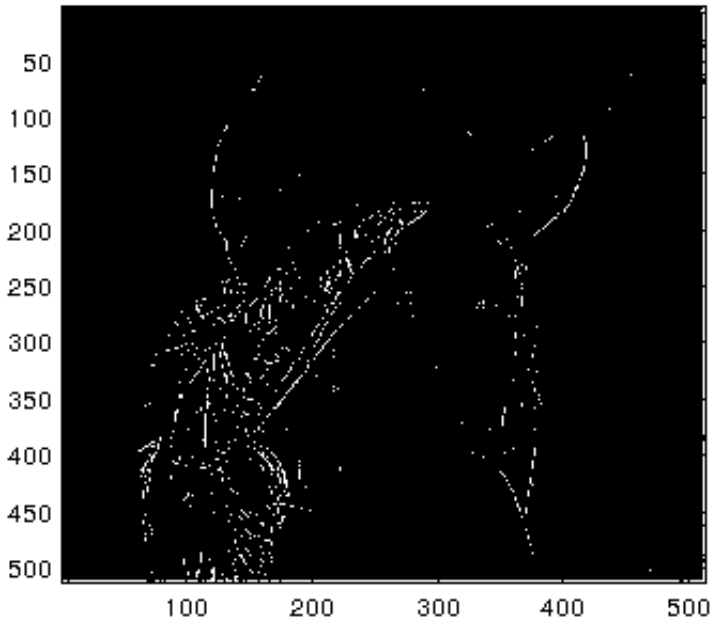

Fig. 6.5: Example of a thresholded image

```
[H,T,R] = hough(BW,'RhoResolution',0.5,'Theta',-90:0.5:89.5);
P = houghpeaks(H,9,'threshold',ceil(0.3*max(H(:))));
% The coefficient  0.3 above may be changed
%to get better results
L = houghlines(BW,T,R,P,'FillGap',5,'MinLength',7);

figure(2)
plot_hough(H,T,R);

figure(3)
imshow(BW);
hold on
plot_lines(L);
hold off
```

6.2.4. De-noising an image

We will use 2D median filter implemented in Matlab with `medfilt2` and compare it with linear filters.

Additive "salt and pepper" noise

- Add to your `image` the noise:
  » `NoisyImage=double(imnoise(uint8(image),'salt & pepper',0.2));` **Remark:** The type conversions are needed, because MATLAB® programmers assumed that images are only `uint8`, but we want to operate on `double` (double-precision floating-point) values.
- Try to remove the noise using a chosen LP linear filter and a median filter (e.g. 3x3 or 5x5 sample ones).

Answer

Answer in the report: Which filter is better at additive noise removal?

Multiplicative noise

- Add the multiplicative (so called "speckle") noise:
  » `MulNoisyImage=double(imnoise(uint8(image),'speckle'));`

Answer

Try a linear LP and median filters now.
Answer in the report: Which filter is better at multiplicative noise removal?

- *Extra* Try to apply filters to a logarithm of the image; do not forget to perform inverse-log operation before displaying the result!!

6.2.5. Dynamic range stretching

- Read a colour image. Display histograms of all the colour channels. Use a ready-made tool `imhist`.
- Stretch each channel with a linear function (e.g. `r1=r*a+c`). Choose constants `a` and `c` by yourself, so that the main part of histogram is stretched to the range 0.0-1.0. Cut smaller and bigger values:
  » `r1=min(1,max(0,r1));`

- Display the corrected image. Repeat the process if you are not satisfied.
  Answer in the report: Which details are now better visible? Did some details disappear? Whit happened to the colors?

Answer