T-61.5100 DIGITAL IMAGE PROCESSING

Exercises 2013

Contents

T-61.5100 Digital image processing, Exercise 1, Sep 24, 2013

To maximise the benefit you obtain from these exercises, please try to work on the problems home before the exercise session. At the exercise session, the course assistant will present you the model solutions to the problems. You should not use so much time on problems marked *demo*; it is sufficient if you look at the problem and just think about it a little. The point in those problems is in what the solution teaches you, you are not required to be able to achieve the solution by yourself (and it may sometimes be difficult if the problem is stated vaguely).

The exercises are not mandatory. They are only to aid you in the learning process. The solutions need not be handed in or returned otherwise, and you do not need to present solutions yourself. Consequently, you cannot earn extra points for the exam from the exercises.

demo 1. The *Mach bands* phenomenon is an example of how the human perception of brightness is not a simple function of of intensity (see Fig. 2.7 in the course book). How could the "Mexican-hat" function (see Fig. 1(a) below) be used to "explain" this phenomenon? As a simplification you can use the approximated function in Fig. 1(b) in one dimension.

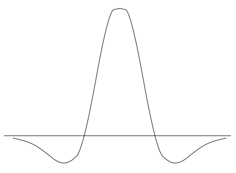


Figure 1(a)

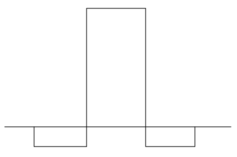


Figure 1(b)

demo 2. Suppose that a flat area with centre at (x_0, y_0) is illuminated by a light source with intensity distribution

$$i(x,y) = Ke^{-[(x-x_0)^2+(y-y_0)^2]}.$$

The reflectance $r(x,y)$ of the area is 1 and $K = 255$. If the resulting image is digitized using n bits of intensity resolution, and the eye can detect an abrupt change of eight shades of intensity between adjacent pixels, what value of n will cause visible false contouring?

3. Consider the two image subsets S_1 and S_2 shown below. For $V = \{1\}$, determine how many (a) 4-connected, (b) 8-connected, and (c) m-connected components there are in S_1 and S_2. Are S_1 and S_2 adjacent?

$$\begin{array}{c|cccc|cccc|c}
\multicolumn{1}{c}{} & \multicolumn{4}{c}{S_1} & \multicolumn{4}{c}{S_2} & \multicolumn{1}{c}{} \\ \cline{2-9}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \boxed{1} & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & \boxed{1} & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ \cline{2-9}
\multicolumn{1}{c}{0} & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & \multicolumn{1}{c}{1}
\end{array}$$

4. In the image above, compute the D_4- and D_8-distances between the two points marked with rectangles. Also compute the D_m-distance given $V = \{1\}$.

5. Assume that we have many noisy versions $g_i(x, y)$ of the same image $f(x, y)$, i.e.

$$g_i(x, y) = f(x, y) + \eta_i(x, y)$$

where the noise η_i is zero-mean and all point-pairs $(\eta_i(x, y), \eta_j(x, y))$ are uncorrelated between each image version. Then we can reduce noise by taking the mean of all the noisy images

$$\bar{g}(x, y) = \frac{1}{M} \sum_{i=1}^{M} g_i(x, y).$$

Prove that

$$E\{\bar{g}(x, y)\} = f(x, y)$$

and

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{M} \sigma^2_{\eta(x,y)}$$

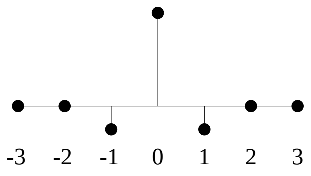
where $\sigma^2_{\eta(x,y)}$ is the variance of η and $\sigma^2_{\bar{g}(x,y)}$ the variance of $\bar{g}(x, y)$.

T-61.5100 Digital image processing, Exercise 1, Sep 24, 2013

1.

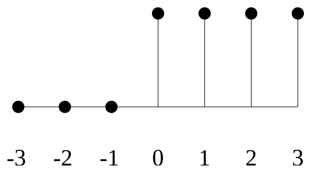
In the early visual system there are cells with an excitatory effect in the centre and an inhibitory effect in the surrounding area. This can be roughly approximated by the "Mexican hat"-function in one dimension. For this problem we use the simplest possible "hat":

$$h(i) = \begin{cases} -0.25, & \text{when } i = \pm 1 \\ 1, & \text{when } i = 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$



An edge is modelled by a step function:

$$f(i) = \begin{cases} 0, & \text{when } i < 0 \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

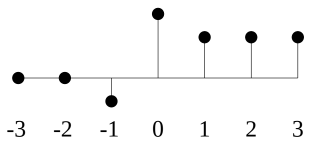


Convolution is used:

$$g(x) = \sum_{i=-\infty}^{\infty} h(i) f(x-i) = \sum_{i=-1}^{1} h(i) f(x-i) \tag{3}$$

And we have

$$g(-3) = 0, \quad g(-2) = 0, \quad g(-1) = -0.25, \quad g(0) = 0.75, \quad g(1) = 0.5, \quad g(2) = 0.5 \tag{4}$$



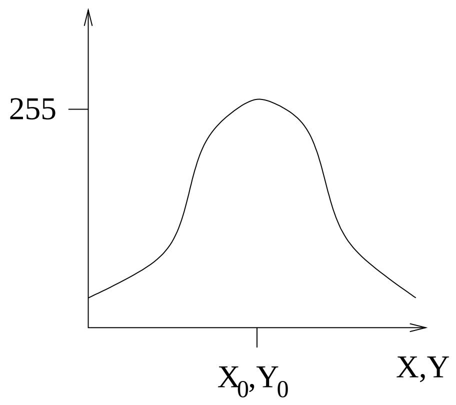
Thus, we see a darker and a lighter bands near the edge. In two dimensions these would be seen as a moat and a wall.

2.

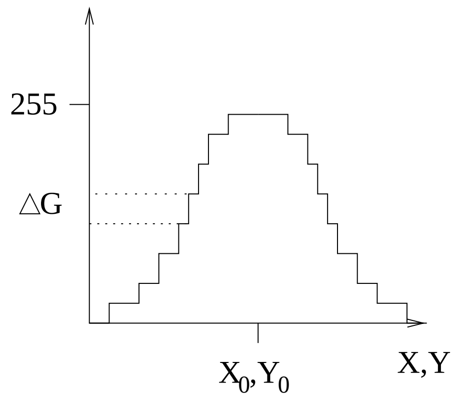
In the simple image formation model (book section 2.3.4) the image is the product of the illumination $i(x,y)$ and the reflectance $r(x,y)$, which is now given as:

$$f(x,y) = i(x,y)r(x,y) = 255e^{-[(x-x_0)^2+(y-y_0)^2]}, \tag{5}$$

and the cross section of the image looks like this



Quantisation with n bits means that we discretise the intensity so that the smallest change is $\Delta G = \frac{255+1}{2^n}$. This is what the quantised picture might look like:



Since the eye is able to perceive a sudden change of 8 intensity levels, we see a false contour, when ΔG equals 8 or is larger, i.e.

$$\begin{aligned} \Delta G &\geq 8 \\ \frac{255+1}{2^n} &\geq 8 \\ \frac{256}{8} &\geq 2^n \\ \log_2 32 &\geq \log_2 2^n \\ 5 &\geq n. \end{aligned} \tag{6}$$

So when $n \leq 5$ a false contour can be seen.

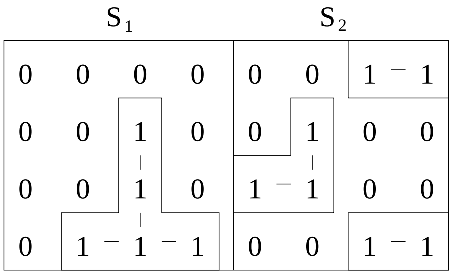
3.

A *connected component* (see book section 2.5.2) is a set in which all its pixels are *connected* with each other. Two pixels are connected if there is a *path* between them, i.e. a chain of pixels which all belong to the set and are *adjacent* in each step of the chain. Adjacency, in turn, means that two pixels are neighbours (e.g. 4-neighbours) and both belong to the set V. Here $V = \{1\}$.

a) 4-connected components, i.e. where we use 4-neighbourhood to define adjacency:

0	0	0	0	0	0	0	1	1	0
1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	1	1	0	0	0
0	0	1	1	1	0	0	1	1	1
0	0	1	1	1	0	0	1	1	1

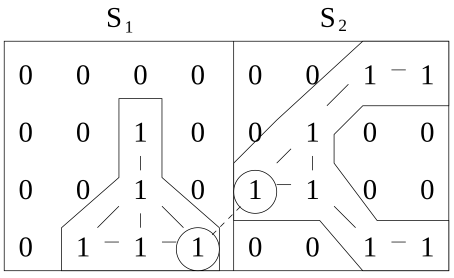
S_1: 1, S_2: 3



b) 8-connected components, i.e. with 8-adjacency:

0	0	0	0	0	0	0	1	1	0
1	0	0	1	0	0	1	0	0	1
1	0	0	1	0	1	1	0	0	0
0	0	1	1	1	0	0	1	1	1
0	0	1	1	1	0	0	1	1	1

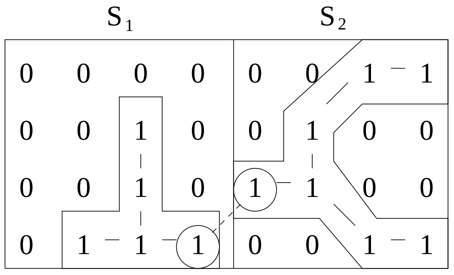
S_1: 1, S_2: 1



c) M-connected components, i.e. with mixed adjacency:

$$
\begin{array}{c|cccc|cccc|c}
\cline{2-9}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
\cline{2-9}
\multicolumn{1}{c}{0} & 0 & 1 & 1 & \multicolumn{1}{c}{1} & 0 & 0 & 1 & \multicolumn{1}{c}{1} & 1
\end{array}
$$

S_1: 1, S_2: 1



The answer is the same as in b), but now there exists only one possible path connecting each pixel-pair.

Adjacency of subsets

Follow-up question: are S_1 and S_2 adjacent in a), b) or c) ?

Two *subsets* are adjacent if there are two pixels (one in each set) which are adjacent (as pixels).

a): S_1 and S_2 are not adjacent, since no pixel of S_2 that belongs to V is a 4-neighbour of any pixel in S_1 that belongs to V.

b) and c): In both cases S_1 and S_2 are adjacent, thanks to the pixels that have been circled in the figures.

4.

D_4 ("City-block") and D_8 ("Chess-board") distances do not depend on V but only on the coordinates of the two points.

$$
\begin{aligned}
D_4(p,q) &= |x_p - x_q| + |y_p - y_q| = 6 + 3 = 9 && (7)\\
D_8(p,q) &= \max(|x_p - x_q|, |y_p - y_q|) = \max(6,3) = 6 && (8)
\end{aligned}
$$

$$D_4(p,q) = |x_p - x_q| + |y_p - y_q| = \dots \tag{9}$$

$$D_8(p,q) = \max(|x_p - x_q|, |y_p - y_q|) = \dots \tag{10}$$

(11)

$$
\begin{array}{c|cccc|cccc|c}
\cline{2-9}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \boxed{1} & 0\\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1\\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0\\
0 & 0 & \boxed{1} & 1 & 1 & 0 & 0 & 1 & 1 & 1\\
\cline{2-9}
\multicolumn{1}{c}{0} & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & \multicolumn{1}{c}{1}
\end{array}
$$

The D_m distance is defined as the shortest m-path (path defined with m-adjacency) between the two points. In this case, the distance will depend on the values of the pixels along the path, as well as the values of their neighbours because only pixels in V can be adjacent.

$$
\begin{tikzcd}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \arrow[r] & \boxed{1} & 0\\
1 & 0 & 0 & 1 & 0 & 0 & 1 \arrow[ur] & 0 & 0 & 1\\
1 & 0 & 0 & 1 & 0 & 1 \arrow[r] & 1 \arrow[u] & 0 & 0 & 0\\
0 & 0 & \boxed{1} \arrow[r] & 1 \arrow[r] & 1 \arrow[ur] & 0 & 0 & 1 & 1 & 1\\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1
\end{tikzcd}
$$

$$\begin{array}{c|cccc|cccc|c}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \boxed{1} & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & \boxed{1} & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1
\end{array}$$

$$D_m(p,q) = 7 \qquad (12)$$

We could also calculate the lengths of the 4- and 8-paths between p and q. Note however that these are not the same as the D_4 and D_8 distances. In fact, what would happen in we calculated the 4- and 8-path lengths for the current example?

5.

The noisy image is now given as

$$\underbrace{g_i(x,y)}_{\text{noisy image}} = \underbrace{f(x,y)}_{\text{original image}} + \underbrace{\eta_i(x,y)}_{\text{noise}},$$

where noise has zero mean and it is uncorrelated, i.e.

$$E\left\{\eta_i(x,y)\right\} = 0,$$

and, for $i \neq j$,

$$E\left\{\eta_i(x,y)\eta_j(x,y)\right\} = 0.$$

When we average the noisy images

$$\bar{g}(x,y) = \frac{1}{M}\sum_{i=1}^{M} g_i(x,y)$$

we get for the expectation

$$E\left\{\bar{g}(x,y)\right\} = \ldots$$

$$E\left\{\bar{g}(x,y)\right\} = E\left\{\frac{1}{M}\sum_{i=1}^{M} g_i(x,y)\right\} = E\left\{\frac{1}{M}\sum_{i=1}^{M} f(x,y) + \frac{1}{M}\sum_{i=1}^{M}\eta_i(x,y)\right\} = \frac{1}{M}\sum_{i=1}^{M} E\left\{f(x,y)\right\} + \frac{1}{M}\sum_{i=1}^{M}\underbrace{E\left\{\eta_i(x,y)\right\}}_{=0 \text{ (zero mean)}} = f(x,y)$$

and for the variance

$$\sigma^2_{\bar{g}(x,y)} = E\left\{\left(\bar{g} - E\left\{\bar{g}\right\}\right)^2\right\} = \ldots$$

$$\sigma^2_{\bar{g}(x,y)} = E\left\{(\bar{g} - E\{\bar{g}\})^2\right\} = E\left\{\left(\frac{1}{M}\sum_{i=1}^{M}(f+\eta_i) - f\right)^2\right\} = E\left\{\left(\frac{1}{M}\sum_{i=1}^{M}\eta_i\right)^2\right\} =$$

$$\frac{1}{M^2}E\left\{\left(\sum_{i=1}^{M}\eta_i\right)^2\right\} = \frac{1}{M^2}E\left\{\sum_{i=1}^{M}\left(\eta_i^2 + \sum_{j=1,\, j\neq i}^{M}\eta_i\eta_j\right)\right\} =$$

$$\frac{1}{M^2}\left[\sum_{i=1}^{M}\left(\underbrace{E\{\eta_i^2\}}_{=\sigma_\eta^2} + \sum_{j=1,\, j\neq i}^{M}\underbrace{E\{\eta_i\eta_j\}}_{=0}\right)\right] = \frac{1}{M}\sigma_\eta^2,$$

since the noise had zero mean and it was uncorrelated.

T-61.5100 Digital image processing, Exercise 2, Oct 1, 2013

Computer exercise: Image enhancement in the spatial domain

1. Set up your workspace.

It is easiest to create a separate directory for your exercise and use that as the "home" directory for Matlab. You can create a new directory, for example with the name "dip", using the command `mkdir dip` in the terminal, or by clicking Places → Home Folder in the top left menu-area. Then in the new window click File → Create Folder and type the name, e.g. "dip" and press Enter.

Matlab can be started by the command `matlab &` in the terminal, or by selecting it from the Applications menu. If you do this exercise at home on your own computer you can also use Octave, the free software alternative to Matlab. There are some instructions on how to use that on the course Noppa page.

> **Tip:** For more advanced users: if you don't the need the graphical interface, Matlab is much quicker if you start in text-mode like this: `matlab -nodesktop -nosplash`. This is also good if you run Matlab over the network, e.g. from at home over ssh. However, if you are unfamiliar with Matlab maybe the graphical mode is easier to start with.

Now in Matlab you need to move to the correct directory: `cd dip`. You can check the current working directory with the command `pwd`.

2. Open and display an image

Now open a web browser (e.g. Firefox) to download the `images2.zip` file, which is an archive package that contains all the images and Matlab files needed:

```
https://noppa.aalto.fi/noppa/kurssi/t-61.5100/viikkoharjoitukset
```

You should then be able to extract the files from the ZIP archive by double clicking on the file and choosing "extract" or similar option, or by using the `unzip` command from the Linux command line. Make sure that the files end up in the "dip" directory that we created for this exercise so that Matlab can find them.

Load the first image into Matlab by the following command:

```
I=double(imread('10.png'))/255;
```

The function `imread` reads in the image called `10.png` into the matrix `I`. Here we also convert the image in to a real-valued matrix (with double precision) and convert the range from $[0, 255]$ to $[0, 1]$. This is not always neccessary, but is usually a good idea. You can check the size of the matrix, and thus the image, by:

```
[M,N]=size(I)
```

Now you also have the number of rows, i.e. height of the image, in `M` and the number of columns, i.e. width of the image, in `N`. You can inspect for example the top left 10×20 pixel corner of the image by:

```
I(1:20,1:10)
```

This now shows the grey scale values of the first 20 rows and 10 columns. This is not very informative for a human, but gives a hint of what the computer is "seeing"!

Typically in image processing (except in the Gonzalez-Woods book!) the first coordinate value is the x-dimension, and the second the y-dimension. Since Matlab expresses everything with matrices it uses the matrix notation where the first value is the row-index ("y-coordinate") and the second value is the column index ("x-coordinate"). *So in Matlab the coordinates are reversed!* Also remember that in Matlab the first index is 1, not 0 as in most programming languages (such as C or Python).

Finally display the image in figure 1:

```
subplot(2,2,1)
imshow(I)
title('image I')
```

The command `subplot` creates a grid of 2×2 smaller figures or subplots, the third parameter says which figure of those four we are using, in this case figure 1. The command `imshow` actually shows the image and `title` is just to display a title above the image.

Tip: In future exercises we commonly will have images which have been manipulated and may not always have the correct range $[0, 1]$. The command `imshow` is very sensitive to this, and this has often caused problems for students. For convenience, Mats Sjöberg – the previous course assitant – has made a very simple image displaying function `showim` which scales the image values before passing it to the standard `imshow` function. This helper function can be found on the course exercise page in Noppa in the same ZIP package as the images. Simply copy the file `showim.m` to your working directory and it can be used as any other function:

```
showim(I)
```

where `I` is any matrix that can be displayed as an image.

Tip: Now when you are going to do more involved things in Matlab, it might be a good idea to write the code into a separate file that you can then run in Matlab in one go. Simply open a new file in a text editor, named for example `dip2.m`. Just type the Matlab commands into this file and save it to the directory that was created in the beginning (e.g. `dip`). Matlab also has a built in editor, just click the icon "New M-file".

In Matlab you can try out different things, and when you have figured out the correct command you can copy & paste it into the text file. You can then run all the commands in the text file in one go in Matlab just by typing `dip2` and pressing enter.

3. Histogram equalisation

As you can see the image `10.png` displays a 10 euro bill with a low contrast. This can be checked by looking at the histogram (in figure 2):

```
subplot(2,2,2)
imhist(I)
title('histogram of I')
```

• How can you see that the image has a low contrast just by looking at the histogram?

Matlab has a built in function for histogram equalisation: `histeq`. To see the help listing and all options of a command just type `help` in front of it. For example try `help histeq`. On the last few rows the help also suggests other related commands that might be useful. In this case for example `imhist` that we have already used. Now equalise the histogram of the image and display the new image `J` in figure 3:

```
J=histeq(I);
subplot(2,2,3)
imshow(J)
title('image J')
```

What can you say about the contrast of the new image `J` compared to the old one `I`? Again, it is instructive to look at the histogram:

```
subplot(2,2,4)
imhist(J)
title('histogram of J')
```

• What can you say about the new histogram compared to the old one, how does this explain the change in contrast?

• Section 3.3.1 in the course book explains histogram equalisation, in the continuous form we are doing the transformation from intensity level r to s by:

$$s = T(r) = (L-1)\int_0^r p_r(w)dw. \tag{1}$$

When the probability density functions for r and s are known, the new distribution is given as $p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$ (result from probability theory).

Since,

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = (L-1)\frac{d}{dr}\left[\int_0^r p_r(w)dw\right] = (L-1)p_r(r) \tag{2}$$

we get

$$p_s(s) = p_r(r)\left|\frac{1}{(L-1)p_r(r)}\right| = \frac{1}{L-1} \tag{3}$$

which is a constant, i.e. a completely uniform distribution. But why is the histogram in our Matlab/Octave programme still not completely uniform?

4. Spatial filtering

Now we start with a new image. It is always good to clear all old variables in Matlab, this can be done with `clear all`. (Octave prefers `clear -v`.) Starting a new text file might also be a good idea.

Now open the image `noisycat.png` in the same way as before and display it. You can se a furry cat sitting on a checkered cloth. Unfortunately the image has a lot of noise. On the left side of the image there is so called "salt & pepper" noise, which means random black and white pixels. On the right side there is Gaussian noise, which is random noise with the values normally distributed.

You can now try to fix this by applying different kinds of filters. This can be done with the Matlab function `imfilter` (try `help imfilter`). You should try with different kinds of masks, with size 3×3 or more. A mask is simply a matrix in Matlab that is convolved over the image matrix. For example a simple mask that only copies the central pixel (i.e. actually does nothing):

```
H = [0 0 0;
     0 1 0;
     0 0 0];

J=imfilter(I,H);
```

Try at least some kind of averaging masks, for example the average over the 8-neighbourhood. Remember to normalise the mask so that its sum is always 1!

Also try a non-linear spatial filter, for example median filtering with `medfilt2`.

Again, if you want to see many images at the same time, you can use `subplot` to create many smaller images, or you can simply create a new figure for each image if you don't mind moving all the windows around. This can be done by clicking on the "New figure" icon in the image window or by the `figure` command.

• How do the different filters perform with the different kinds of noise (i.e. the left and right side of the image)? What about the different patterns of the image, for example the cat's fur or the checkered cloth?

• Why does certain filters work better on certain kinds of noise, and in certain areas of the image?

The original noiseless image can be found in `cat.png` if you wish to compare to that. It is not possible to restore the image to perfect shape, but the quality can be improved.

• If you have the original image available, how could you numerically compare the quality of different filtering methods?

T-61.5100 Digital image processing, Exercise 3, Oct 8, 2013

Image enhancement in the frequency domain

1. Fourier transform the sequence $f(0) = 2, f(1) = 3, f(2) = 4, f(3) = 4$. Then calculate the inverse Fourier transform and compare the result with the original sequence.

2. Suppose that you form a low-pass spatial filter that averages the 4-neighbours of a point (x, y), but excludes the point (x, y) itself.

   (a) Find the equivalent filter $H(u, v)$ in the frequency domain for an $N \times N$ sized image.
   Hint: Calculate $G(u, v) = \mathcal{F}\{g(x, y)\}$, and remember that the Fourier transform $\mathcal{F}$ is a linear operator. Once you have $G(u, v)$ you can solve $H(u, v)$ from $G(u, v) = H(u, v)F(u, v)$.

   (b) Show that $H(u, v)$ is a low-pass filter.
   Hint: look at what values $|H(u, v)|$ has at the origin $(0, 0)$ and how it changes when moving away from the origin.

   (c) Consider also the phase response $\phi(u, v) = \arg H(u, v)$.
   The phase response is the phase angle $\phi(u, v)$ of $H(u, v) = |H(u, v)|e^{j\phi(u,v)}$, since $H(u, v)$ in general is a complex value (i.e. with a real and imaginary part).
   However, in this exercise, what can we say about the imaginary part?

   $$\mathrm{Im}\{H(u, v)\} = \ldots$$

   and then,

   $$\phi(u, v) = \arg H(u, v) = \ldots$$

demo 3. A Gaussian low-pass filter in the frequency domain has the transfer function

$$H(u, v) = Ae^{-(u^2+v^2)/2\sigma^2}.$$

Show that the corresponding filter in the spatial domain has the form

$$h(x, y) = A2\pi\sigma^2 e^{-2\pi^2\sigma^2(x^2+y^2)}.$$

4. Show that the continuous Fourier transform of the continuous convolution of two functions is the product of their Fourier transforms. For simplicity, assume 1-D functions. That is, show that for functions $f(x)$ and $g(x)$,

$$\mathcal{F}\{f(x) * g(x)\} = \mathcal{F}\{f(x)\}\mathcal{F}\{g(x)\}$$

T-61.5100 Digital image processing, Exercise 3, Oct 8, 2013

1.

The discrete Fourier transform and the inverse transform:

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N}, \quad f(x) = \sum_{u=0}^{N-1} F(u)e^{j2\pi ux/N}$$

Note: The version of the Fourier transform that we adopt here differs from the one presented in the book by the fact that we scale the values by the length of the sequence. This does not affect the theory in any way, one only needs to know if scaling was done when performing the inverse transform (note that we do not scale there but the scaling is done in the book).

The sequence is now $f(0) = 2, f(1) = 3, f(2) = 4, f(3) = 4$. The Fourier-transform is done using 4 points $\Longrightarrow$ $N = 4$:

$$F(0) = \frac{1}{4}\sum_{x=0}^{3} f(x)e^{0} = \frac{1}{4}[f(0) + f(1) + f(2) + f(3)] = \frac{1}{4}(2+3+4+4) = \frac{13}{4} = 3.25$$

$$\begin{aligned}
F(1) &= \frac{1}{4}\sum_{x=0}^{3} f(x)e^{-j2\pi x/4} = \frac{1}{4}[f(0)e^{0} + f(1)e^{-j\pi/2} + f(2)e^{-j\pi} + f(3)e^{-j3\pi/2}] \\
&= \frac{1}{4}[2e^{0} + 3e^{-j\pi/2} + 4e^{-j\pi} + 4e^{-j3\pi/2}] \\
&= \frac{1}{4}[2 + 3(\cos(\pi/2) - j\sin(\pi/2)) + 4(\cos(\pi) - j\sin(\pi)) + 4(\cos(3\pi/2) - j\sin(3\pi/2))] \\
&= \frac{1}{4}[2 + 3(-j) + 4(-1) + 4(j)] = \frac{1}{4}(-2+j)
\end{aligned}$$

$$F(2) = \frac{1}{4}\sum_{x=0}^{3} f(x)e^{-j2\pi 2x/4} = \frac{1}{4}(f(0) - f(1) + f(2) - f(3)) = -\frac{1}{4}$$

$$F(3) = \frac{1}{4}\sum_{x=0}^{3} f(x)e^{-j2\pi 3x/4} = \frac{1}{4}(f(0) + jf(1) - f(2) - jf(3)) = -\frac{1}{4}(2+j)$$

Then we do the inverse transform:

$$f(0) = \sum_{u=0}^{3} F(u)e^{0} = F(0) + F(1) + F(2) + F(3) = \frac{13}{4} + \frac{1}{4}(-2+j) - \frac{1}{4} - \frac{1}{4}(2+j) = 2$$

$$f(1) = \sum_{u=0}^{3} F(u)e^{j2\pi u/4} = \frac{13}{4} + \frac{1}{4}(-2+j)e^{j\pi/2} - \frac{1}{4}e^{j\pi} - \frac{1}{4}(2+j)e^{j3\pi/2} = 3$$

$$f(2) = \sum_{u=0}^{3} F(u)e^{j2\pi 2u/4} = \frac{13}{4} + \frac{1}{4}(-2+j)(-1) - \frac{1}{4} - \frac{1}{4}(2+j)(-1) = 4$$

$$f(3) = \sum_{u=0}^{3} F(u)e^{j2\pi 3u/4} = \frac{13}{4} + \frac{1}{4}(-2+j)(-j) - \frac{1}{4}(-1) - \frac{1}{4}(2+j)j = 4$$

The result is thus the original sequence.

2.

The spatial average is

$$g(x,y) = \frac{1}{4}\left[f(x,y+1) + f(x+1,y) + f(x,y-1) + f(x-1,y)\right].$$

We need the following property of the Fourier transform (see Table 4.3[1] in the book):

$$\mathcal{F}\left\{f(x-x_0,y-y_0)\right\} = \exp\left[-j2\pi(ux_0+vy_0)/N\right]F(u,v) = e^{-j2\pi ux_0/N}e^{-j2\pi vy_0/N}F(u,v)$$

a)

$$G(u,v) = \frac{1}{4}\left[e^{-j2\pi v/N} + e^{-j2\pi u/N} + e^{j2\pi v/N} + e^{j2\pi u/N}\right]F(u,v) = H(u,v)F(u,v).$$

Because

$$e^{-jx} = \cos x - j\sin x \quad \text{and} \quad e^{jx} = \cos x + j\sin x$$

we get

$$H(u,v) = \frac{1}{2}\left[\cos\frac{2\pi u}{N} + \cos\frac{2\pi v}{N}\right]$$

that is the filter transfer function in the frequency domain. ($H(u,v)$ is real because the filter is symmetric!)

b) $|H(0,0)| = 1$ and $|H|$ decreases as a function of distance from the origin. This is the characteristic of a low-pass filter. In fact this happens monotonically in the directions of u and v axis only; in the diagonal directions also the higher frequencies are emphasized:

$$\left|H(\pm\frac{N}{2},\pm\frac{N}{2})\right| = |-1| = 1$$

Look at the images!

c) The phase response can be calculated from $H(u,v)$. Because $\mathrm{Im}\{H(u,v)\} = 0$, then the phase response is 0 when $\mathrm{Re}\{H(u,v)\} > 0$, and π where $\mathrm{Re}\{H(u,v)\} < 0$:

$$\arg H(u,v) = \begin{cases} 0, & \text{if } (|u|+|v|)/N \le 0.5 \\ \pi, & \text{if } (|u|+|v|)/N > 0.5 \end{cases}$$

Look at the images!

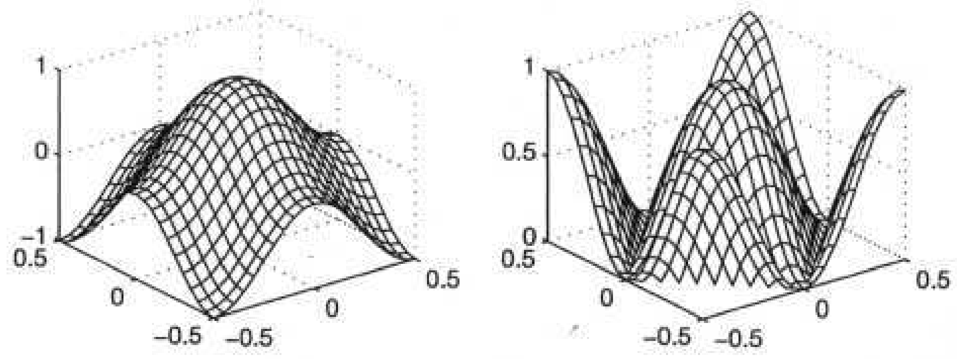
So H retains (but turns) "the chessboard" ($u = v = N/2$) but averages the horizontal and vertical lines in images.

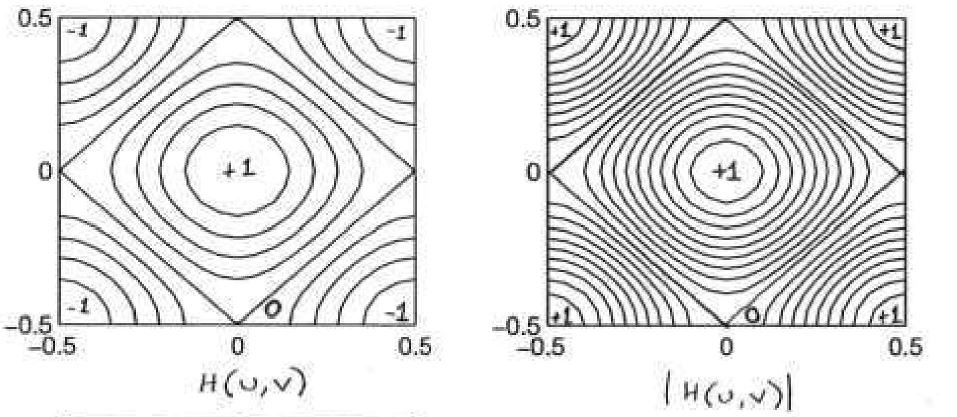
[1]Third Edition

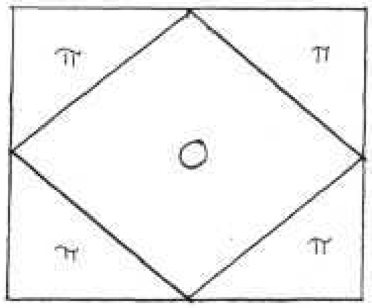
```
                        < M A T L A B (R) >
              (c) Copyright 1984-97 The MathWorks, Inc.
                        All Rights Reserved
                         Version 5.1.0.421
                            May 25 1997

  To get started, type one of these commands: helpwin, helpdesk, or demo.
  For information on all of the MathWorks products, type tour.

>> u=-0.5:.05:0.5;
>> v=u;
>> [X,Y]=meshdom(u,v);
>> Z=0.5*(cos(2*pi.*X)+cos(2*pi.*Y));
>> subplot(3,2,1), mesh(u,v,Z);
>> subplot(3,2,2), mesh(u,v,abs(Z));
>> subplot(3,2,3), contour(u,v,Z);
>> subplot(3,2,4), contour(u,v,abs(Z));
>> subplot(3,2,5), mesh(u,v,atan2(imag(Z),real(Z)));
>> print -dps
```







arg H(u,v)

3.

We start with only one variable and show first that, if

$$H(u) = e^{-u^2/2\sigma^2}$$

then

$$h(x) = \int_{-\infty}^{\infty} e^{-u^2/2\sigma^2} e^{j2\pi ux} du = \sqrt{2\pi}\sigma e^{-2\pi^2 x^2 \sigma^2}.$$

We can express the integral in the preceding equation as

$$h(x) = \int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}[u^2 - j4\pi\sigma^2 ux]} du.$$

We now make use of the identity

$$e^{-\frac{(2\pi)^2 x^2 \sigma^2}{2}} e^{\frac{(2\pi)^2 x^2 \sigma^2}{2}} = 1.$$

Inserting this identity in the preceding integral yields

$$h(x) = e^{-\frac{(2\pi)^2 x^2 \sigma^2}{2}} \int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}[u^2 - j4\pi\sigma^2 ux - (2\pi)^2\sigma^4 x^2]} du = e^{-\frac{(2\pi)^2 x^2 \sigma^2}{2}} \int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}[u - j2\pi\sigma^2 x]^2} du.$$

Next we make the change of variable $r = u - j2\pi\sigma^2 x$. Then, $dr = du$ and the above integral becomes

$$h(x) = e^{-\frac{(2\pi)^2 x^2 \sigma^2}{2}} \int_{-\infty}^{\infty} e^{-\frac{r^2}{2\sigma^2}} dr.$$

Finally, we multiply and divide the right side of this equation by $\sqrt{2\pi}\sigma$:

$$h(x) = \sqrt{2\pi}\sigma e^{-\frac{(2\pi)^2 x^2 \sigma^2}{2}} [\frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} e^{-\frac{r^2}{2\sigma^2}} dr].$$

The expression inside the brackets is recognized as a Gaussian probability density function, whose integral from $-\infty$ to ∞ is 1. Therefore,

$$h(x) = \sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2 x^2}.$$

With this result as background, we now move into two-dimensional case. By substituting directly into definition of the inverse Fourier transform we have:

$$h(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Ae^{-(u^2+v^2)/2\sigma^2} e^{j2\pi(ux+vy)} du dv = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} Ae^{(-\frac{u^2}{2\sigma^2} + j2\pi ux)} du \right] e^{(-\frac{v^2}{2\sigma^2} + j2\pi vy)} dv$$

By using the obtained result for the one-dimensional case, we now have the final result:

$$h(x, y) = (A\sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2 x^2})(\sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2 y^2}) = A2\pi\sigma^2 e^{-2\pi^2\sigma^2(x^2+y^2)}.$$

4.

Convolution:

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)\,d\alpha$$

The Fourier transform of a delay:

$$\mathcal{F}\{g(x-\alpha)\} = G(u) \cdot e^{-j2\pi u\alpha}$$

By direct substitution into the definition, we get

$$\mathcal{F}\{f(x) * g(x)\} = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)\,d\alpha \right] e^{-j2\pi ux}\,dx$$

$$= \int_{-\infty}^{\infty} f(\alpha) \left[\int_{-\infty}^{\infty} g(x-\alpha)e^{-j2\pi ux}\,dx \right] d\alpha = G(u)\int_{-\infty}^{\infty} f(\alpha)e^{-j2\pi u\alpha}\,d\alpha = G(u)F(u)$$

This is a very practical result, since it is often easier to calculate separately the transforms of f and g by FFT, multiply them and calculate the inverse transform than to calculate the convolution straight from the definition.

T-61.5100 Digital image processing, Exercise 4, Oct 17, 2013

Image enhancement in the frequency domain

This computer exercise assumes that you already know the basics of image processing in Matlab. E.g. opening an image (`imread`) and showing it on the screen (`imshow` or `showim` provided with the images). If you have forgotten something you can always check back to the first computer exercise (exercise no 2 on the course Noppa page).

1. Fourier analysis of periodic noise

Open the two images `car.png` and `noisycar.png`. The first one is the original grey-scale image of a car, and the second one has been corrupted by vertical lines of noise (periodical and sinusoidal noise). Look at the two pictures to confirm this.

Now, calculate the 2D discrete Fourier-transform of both images using the Matlab-function `fft2`. For the purpose of visualisation it is useful to shift the origo of the transformed image to the centre. Matlab has a handy function for this: `fftshift`.

To plot the Fourier transform we use the absolute value (the transform itself is imaginary):

```
imshow(abs(F_shift),[ ])
```

- The first parameter to `imshow` is the absolute value of the shifted Fourier-transformed image.
- The second parameter of the `imshow` command gives the limits of the image values. Since these are now out of the normal image range (typically $0 \ldots 255$ in Matlab) we have to specify them explicitly in the second parameter. Giving an empty vector `[ ]` is a handy shortcut to giving the minimum and maximum of the matrix values. (See `help imshow` for a longer explanation.) (Alternatively use the provided function `showim` which does the scaling itself.)

You may notice that you cannot see anything more than a white dot in the middle, this is because the peak in the middle is so strong that the lower values are almost black in comparison. A typical solution is to use a logarithmic scale, i.e. just take the `log` of all values. Try doing this!

If you compare the two Fourier spectra (i.e. from the original image and from the image with the noise), you should notice a difference. It might be difficult to see initially since the centre peak is so strong.

Hint: the Fourier of a cosine-function is two impulses with equal distance from the origin (one positive, the other negative).

The differences between the Fourier spectra of the images must be because of the noise. Considering this, how might one remove the noise by filtering in the Fourier-domain?

2. Constructing filters in the frequency domain

The purpose is now to blur the image `car.png` by low-pass filtering. We will try two different filters: the *ideal lowpass filter* (ILPF) and a *Gaussian lowpass filter* (GLPF).

Start by getting the size of the image $P \times Q$ if the image is in matrix `I`:

```
[Q,P] = size(I)
```

You should have $Q = 224, P = 400$. Again because Matlab uses matrix convention, i.e. rows $\times$ columns, they are in the oppsite order of the typical width $\times$ height, i.e. the image is of size 400×224 in this case.

Next we need to create a matrix `D` with the distance values from the centre evaluated at each point. This can be done using the equation (from Eq. 4.8-2 in the course book):

$$D(u,v) = \left[(u - P/2)^2 + (v - Q/2)^2\right]^{1/2}$$

In Matlab:

```
[u,v] = meshgrid(0:P-1,0:Q-1);
D=((u-P/2).^2 + (v-Q/2).^2).^0.5;
```

The matrix `D` should now be zero in the middle and have increasing values with increasing distance from the middle. You can visualise the matrix for example by `imshow(D,[ ])` or show a 3D version by `mesh(D)`. If Matlab is slow with `mesh`, it might be because the matrix is quite big. You can sample only every five element of the matrix in the mesh like this:

```
mesh(D(1:5:end,1:5:end))
```

The 3D mesh can be rotated with the image tools in Matlab to see from different angles.

Now the ideal lowpass filter (Eq. 4.8-1):

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

can be easily generated, for example for $D_0 = 30$:

```
D0=30
H_ideal = double(D<=D0);
```

Visualise the filter again using both `imshow` and `mesh`.

Do the same for the Gaussian lowpass filter (Eq. 4.8-6).

$$H(u,v) = \exp\left(-\frac{D^2(u,v)}{2D_0^2}\right).$$

In Matlab:

```
H_gauss = exp(-(D.^2)./(2*D0^2));
```

And visualise it in the same way. How do the filters differ in the frequency (Fourier) domain?

3. Applying the filters

Now Fourier transform the image `car.png` and shift it so that the origin is in the centre of the image. (This was done already in section 1.) Applying a filter in the frequency domain is now simply a matter of multiplying each element of the matrices:

```
G_ideal=H_ideal.*F_shift;
```

To display the transformed image, we need to first reverse the shifting using `ifftshift`, then do the inverse Fourier transform `ifft2` back to the spatial (image) domain:

```
g_ideal=real(ifft2(ifftshift(G_ideal)));
```

Just to be safe we also take the real part only – due to rounding errors some imaginary parts might still be left. The image can now be plotted as normally. Do you notice any problems with the image?

Try the same with the Gaussian filter. Can you see any difference? If so, why?

Hint: in the same way as with the transformed images you can also translate the filter matrices (`H_ideal` and `H_gauss`) to the image domain with the inverse transform and display them using `imshow` or `mesh`.

4. Remove periodic noise

Now, since we have learned how to construct filters in the Fourier domain, we can try to fix the periodic noise in section 1 from `noisycar.png`. What kind of filter would be ideal for this case?

T-61.5100 Digital image processing, Exercise 5, Oct 29, 2013

Image enhancement in the frequency domain

1. Suppose that you are given a set of images generated by an experiment dealing with the analysis of stellar events. Each image contains a set of bright, widely scattered dots corresponding to stars in a sparsely occupied section of the universe. The problem is that the stars are barely visible, owing to superimposed illumination resulting from atmospheric dispersion. If these images are modeled as the product of a constant illumination component with a set of impulses, give an enhancement procedure based on homomorphic filtering designed to bring out the image components due to the stars themselves.

demo 2. Investigate what the Butterworth low-pass filters look like in the spatial domain. Let the cutoff frequency be $D_0 = N/6$. Generate the spatial masks of size 3×3 and 5×5 of order $n = 1$ and $n = 2$.

Image restoration

demo 3. During acquisition, an image undergoes uniform linear motion in the vertical direction for a time T_1. The direction of motion then switches to the horizontal direction for a time interval T_2. Assuming that the time it takes the image to change directions is negligible, and the shutter opening and closing times are negligible also, give an expression for the blurring function, $H(u, v)$.

demo 4. Cannon [1974] suggested the power spectrum equalization filter $R(u, v)$ based on the premise of forcing the power spectrum of the restored image to equal the power spectrum of the original image:

$$S_{\hat{f}}(u, v) = |R(u, v)|^2 S_g(u, v) = S_f(u, v).$$

Find $|R(u, v)|$, the magnitude response of the restoration filter.

5. Find magnitude responses for the
   (a) inverse filter
   (b) power spectrum equalization filter
   (c) Wiener filter

in the points (u, v) of frequency domain, where signal power spectrum $S_f(u, v)$, noise $S_n(u, v)$ power spectrum and magnitude response of point spread function (PSF) $H(u, v)$ have the following values:

$\lvert H(u,v)\rvert$	$S_f(u,v)$	$S_n(u,v)$	
H	0	N	• signal power zero
H	S	0	• no noise
1.0	3000.0	0.01	• close to uv-origin
0.7	0.7	0.01	• low frequencies
0.01	0.005	0.01	• high frequencies

6. (a) Show that the application of a 3×3-sized local mean mask can be replaced by 1×3 and 3×1 masks applied sequentially. Compare the amount of additions that are needed in both cases.

   (b) Compare the amounts of additions and multiplications that are needed in a general case, where a N×N mask is replaced by 1×N and N×1 masks and masks' coefficients are not equal to ones.

   (c) Depict the 3×3 Sobel gradient masks. Show for one of the Sobel masks that it can be separated as above into two one-dimensional masks.

   (d) Is it possible to separate the 3×3 discrete Laplace-operator?

T-61.5100 Digital image processing, Exercise 5, Oct 29, 2013

1.

Let us assume that there is only a single star that is modeled as an impulse $\delta(x-x_0, y-y_0)$ where (x_0, y_0) are the coordinates of the star. A discrete impulse is defined as:

$$\delta(x-x_0,y-y_0)=\begin{cases}1 & \text{when } (x,y)=(x_0,y_0)\\ 0 & \text{otherwise}\end{cases}$$

K is the illumination and ϵ the reflectance of the sky. Then the image model is

$$f(x,y)=K(\delta(x-x_0,y-y_0)+\epsilon)$$

Homomorphic filtering:

$$f(x,y)\rightarrow\boxed{\ln}\rightarrow\boxed{\mathcal{F}}\rightarrow\boxed{H(u,v)}\rightarrow\boxed{\mathcal{F}^{-1}}\rightarrow\boxed{\exp}\rightarrow g(x,y)$$

So, first we take the logarithm of the image:

$$\ln f(x,y)=\ln K+\ln(\delta(x-x_0,y-y_0)+\epsilon)=\ln K+\ln\epsilon+\delta(x-x_0,y-y_0)(\ln(1+\epsilon)-\ln\epsilon).$$

Then we Fourier transform:

$$\begin{aligned}H(u,v)=\mathcal{F}\{\ln f(x,y)\}&=\mathcal{F}\{\ln K\epsilon\}+\mathcal{F}\{\delta(x-x_0,y-y_0)(\ln(1+\epsilon)-\ln\epsilon)\}\\&=C_1\delta(u,v)+C_2e^{-j2\pi(\frac{ux_0}{M}+\frac{vy_0}{N})},\end{aligned}$$

where $C_1=\ln K\epsilon$ and $C_2=\ln(1+\epsilon)-\ln\epsilon$.

From this result, it is evident that the contribution of illumination is an impulse at the origin of the frequency plane. It can be cancelled by high-pass filtering the image (e.g. using a notch filter). Extension of this development to multiple impulses (stars) is straightforward. The filter will be the same.

2.

In principle the filter $H(u,v)$ corresponds to a mask $h(x,y)$ of the same size as the image (e.g. $N \times N$). However, this would be very inefficient, so we try to generate a smaller $m \times m$-sized spatial mask $\hat{h}(x,y)$ instead, so that it's corresponding filter $\hat{H}(u,v)$ is as close to $H(u,v)$ as possible in the sense of the squared error.

Let $\hat{\mathbf{h}}$ be the mask ordered as an m^2-sized vector, i.e. a vector with one column of m^2 rows,

$$\hat{\mathbf{h}} = \begin{pmatrix} \hat{h}(0,0) \\ \vdots \\ \hat{h}(0,N-1) \\ \hat{h}(1,0) \\ \vdots \\ \hat{h}(N-1,N-1) \end{pmatrix}$$

The Fourier transform is linear. Let the complex transformation matrix be $\mathbf{F}$, and the transformed mask (N^2-sized complex vector) be $\hat{\mathbf{H}}$:

$$\hat{\mathbf{H}} = \mathbf{F}\hat{\mathbf{h}}.$$

Now we want $\hat{\mathbf{H}}$ to be as close as possible to the Butterworth filter $\mathbf{H}$ (also N^2-sized complex vector). In effect we want to minimize the error function,

$$\mathcal{E}(\hat{\mathbf{h}}) = ||\hat{\mathbf{H}} - \mathbf{H}||^2 = ||\mathbf{F}\hat{\mathbf{h}} - \mathbf{H}||^2.$$

The minimum can be found where the partial derivative vanishes:

$$\frac{\partial \mathcal{E}(\hat{\mathbf{h}})}{\partial \hat{\mathbf{h}}} = 0.$$

Since for any matrix, $||\mathbf{A}||^2 = \mathbf{A}^T\mathbf{A}$,

$$\begin{aligned}
\mathcal{E}(\hat{\mathbf{h}}) &= ||\mathbf{F}\hat{\mathbf{h}} - \mathbf{H}||^2 \\
&= (\mathbf{F}\hat{\mathbf{h}} - \mathbf{H})^T(\mathbf{F}\hat{\mathbf{h}} - \mathbf{H}) \\
&= (\hat{\mathbf{h}}^T\mathbf{F}^T - \mathbf{H}^T)(\mathbf{F}\hat{\mathbf{h}} - \mathbf{H}) \\
&= \hat{\mathbf{h}}^T\mathbf{F}^T\mathbf{F}\hat{\mathbf{h}} - \hat{\mathbf{h}}^T\mathbf{F}^T\mathbf{H} - \mathbf{H}^T\mathbf{F}\hat{\mathbf{h}} + \mathbf{H}^T\mathbf{H} \\
&= \hat{\mathbf{h}}^T\mathbf{F}^T\mathbf{F}\hat{\mathbf{h}} - 2\hat{\mathbf{h}}^T\mathbf{F}^T\mathbf{H} + \mathbf{H}^T\mathbf{H}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathcal{E}(\hat{\mathbf{h}})}{\partial \hat{\mathbf{h}}} &= 2\mathbf{F}^T\mathbf{F}\hat{\mathbf{h}} - 2\mathbf{F}^T\mathbf{H} \\
&= 2\mathbf{F}^T(\mathbf{F}\hat{\mathbf{h}} - \mathbf{H})
\end{aligned}$$

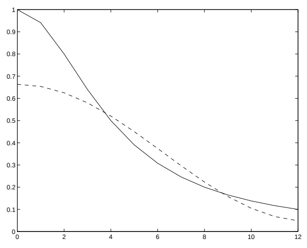
The derivative is zero when $\mathbf{F}\hat{\mathbf{h}} - \mathbf{H} = 0$, i.e.

$$\begin{aligned}
\mathbf{F}\hat{\mathbf{h}} &= \mathbf{H} \\
\mathbf{F}^T\mathbf{F}\hat{\mathbf{h}} &= \mathbf{F}^T\mathbf{H} \\
\hat{\mathbf{h}} &= (\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{H} \\
\hat{\mathbf{h}} &= \mathbf{F}^\dagger\mathbf{H},
\end{aligned}$$

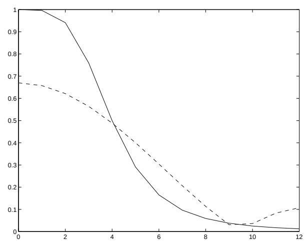
where $\mathbf{F}^\dagger = (\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T$ is the pseudo-inverse of $\mathbf{F}$.

The Matlab-code `spatial.m` that generates the spatial mask for the Butterworth filter will be published on the course Noppa page.

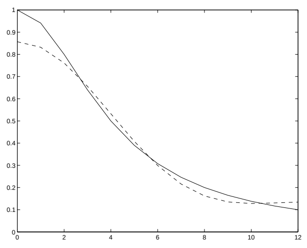
Results:



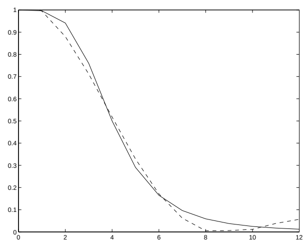
0.0404	0.0736	0.0404
0.0736	0.2075	0.0736
0.0404	0.0736	0.0404



0.0576	0.0800	0.0576
0.0800	0.1196	0.0800
0.0576	0.0800	0.0576



0.0067	0.0130	0.0156	0.0130	0.0067
0.0130	0.0404	0.0736	0.0404	0.0130
0.0156	0.0736	0.2075	0.0736	0.0156
0.0130	0.0404	0.0736	0.0404	0.0130
0.0067	0.0130	0.0156	0.0130	0.0067



0.0107	0.0244	0.0318	0.0244	0.0107
0.0244	0.0576	0.0800	0.0576	0.0244
0.0318	0.0800	0.1196	0.0800	0.0318
0.0244	0.0576	0.0800	0.0576	0.0244
0.0107	0.0244	0.0318	0.0244	0.0107

Solid line: the frequency response of the Butterworth filter.

Dotted line: the frequency response of the spatial mask.

Left column: $n = 1$. Right column: $n = 2$.

Notice that in both cases the 3×3 mask is included in the 5×5 mask. This is a general property that comes from the orthogonality. If the mask size is increased, the obtained frequency response approaches the frequency response of the Butterworth filter.

3.

Because the motion in the x- and y-directions are independent (motion is in the vertical (x) direction only at first, and then switching to motion only in the horizontal (y) direction) this problem can be solved in two steps. The first step is identical to the analysis that resulted in Eq. (5.6-10), which gives the blurring function due to vertical motion only:

$$H_1(u,v) = \frac{T_1}{\pi ua}\sin(\pi ua)e^{-j\pi ua},$$

where we are representing linear motion by the equation $x_0(t) = at/T_1$. The function $H_1(u,v)$ would give us a blurred image in the vertical direction. That blurred image is the image that would then start moving in the horizontal direction and to which horizontal blurring would be applied. This is nothing more than applying a second filter with transfer function

$$H_2(u,v) = \frac{T_2}{\pi vb}\sin(\pi vb)e^{-j\pi vb},$$

where we assumed the form $y_0(t) = bt/T_2$ for motion in the y-direction. Therefore, the overall blurring transfer function is given by the product of these two functions

$$H(u,v) = \frac{T_1T_2}{(\pi ua)(\pi vb)}\sin(\pi ua)\sin(\pi vb)e^{-j\pi ua}e^{-j\pi vb},$$

and the overall blurred image is

$$g(x,y) = \mathcal{F}^{-1}[H(u,v)F(u,v)]$$

where $F(u,v)$ is the Fourier transform of the input image.

4.

The power spectrum of the restored image $\hat{f}$ is given by

$$S_{\hat{f}}(u,v) = |R(u,v)|^2 S_g(u,v).$$

The restoration filter should force the power spectrum of the restored image to equal the power spectrum of the original image:

$$S_{\hat{f}}(u,v) = S_f(u,v).$$

We can easily solve $|R(u,v)|$ from the first equation: $|R(u,v)| = \sqrt{S_f(u,v)/S_g(u,v)}$, but we need to calculate $S_g(u,v)$ first. Let's drop the indexing (u,v) for brevity, and use the degradation model: $G = HF + N$:

$$\begin{aligned} S_g &= |G|^2 = G^*G = (HF+N)^*(HF+N) = (F^*H^* + N^*)(HF+N) \\ &= F^*H^*HF + F^*H^*N + N^*HF + N^*N \\ &= |H|^2|F|^2 + |N|^2 + H^*(\cancel{F^*N}) + H(\cancel{N^*F}) \\ &= |H|^2 S_f + S_\eta \end{aligned}$$

where the cross-terms vanish because the image and the noise are uncorrelated (i.e. $F^*N = 0$. $S_\eta(u,v)$ is the power spectrum of the noise. We can now insert this into the equation for $|R(u,v)|$:

$$|R(u,v)| = \sqrt{\frac{S_f(u,v)}{|H(u,v)|^2 S_f(u,v) + S_\eta(u,v)}} = \sqrt{\frac{1}{|H(u,v)|^2 + \frac{S_\eta(u,v)}{S_f(u,v)}}}$$

5.

Filter	**Expression**	**Magnitude response**
Inverse	$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$	$\lvert H_{\text{INV}}(u,v)\rvert = \frac{1}{\lvert H(u,v)\rvert}$
PSE (exer. 4)	-	$\lvert H_{\text{PSE}}(u,v)\rvert = \sqrt{\frac{1}{\lvert H(u,v)\rvert^2 + \frac{S_\eta(u,v)}{S_f(u,v)}}}$
Wiener	$\hat{F}(u,v) = \left[\frac{H^\star(u,v)}{\lvert H(u,v)\rvert^2 + \frac{S_\eta(u,v)}{S_f(u,v)}}\right] G(u,v)$	$\lvert H_{\text{WIENER}}(u,v)\rvert = \frac{\lvert H(u,v)\rvert}{\lvert H(u,v)\rvert^2 + \frac{S_\eta(u,v)}{S_f(u,v)}}$

Let us substitute the given values into these equations:

$\lvert H(u,v)\rvert$	$S_f(u,v)$	$S_\eta(u,v)$	$\lvert H_{\text{INV}}(u,v)\rvert$	$\lvert H_{\text{PSE}}(u,v)\rvert$	$\lvert H_{\text{WIENER}}(u,v)\rvert$
H	0	N	$1/H$	0	0
H	S	0	$1/H$	$1/H$	$1/H$
1.0	3000	0.01	1.0	≈ 1.0	≈ 1.0
0.7	0.7	0.01	≈ 1.43	≈ 1.41	≈ 1.38
0.01	0.005	0.01	100.0	≈ 0.71	≈ 0.005

6.

a) The 3×3-sized local mean mask is (scaling is omitted for simplicity)

1	1	1
1	1	1
1	1	1

The part of an image that falls under the mask is given as

a	b	c
d	e	f
g	h	i

The mask response in position e is $e^* = 1\cdot a + 1\cdot b + 1\cdot c + 1\cdot d + 1\cdot e + 1\cdot f + 1\cdot g + 1\cdot h + 1\cdot i$ $= a+b+c+d+e+f+g+h+i$. If we are using the mask $\boxed{1}\boxed{1}\boxed{1}$, the responses in positions b, e, and h are $b' = a+b+c$, $e' = d+e+f$, and $h' = g+h+i$. When we then apply the mask $\boxed{1}\boxed{1}\boxed{1}^T$, the response in position e is $e'' = b'+e'+h' = (a+b+c)+(d+e+f)+(g+h+i) = e^*$.

With a 3×3 mask we have 8 additions for each mask position. With a 1×3 mask we have 2 additions for each position. Thus using 1×3 and 3×1 masks takes a total of 4 additions for each position which is half the number of additions needed with a 3×3 mask.

b) In a general case we have N^2-1 additions and N^2 multiplications with a N×N mask. Using the separate masks takes $2(N-1)$ additions and $2N$ multiplications.

c) The 3×3 Sobel gradient masks are

$$G_x = \begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\\hline 0 & 0 & 0 \\\hline 1 & 2 & 1 \\\hline\end{array} \qquad\qquad G_y = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\\hline -2 & 0 & 2 \\\hline -1 & 0 & 1 \\\hline\end{array}$$

G_x measures horizontal edges and G_y vertical edges.

Let us use the G_x mask. The response in position e is $e^* = (g+2h+i)-(a+2b+c)$. With a one-dimensional difference mask $\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\\hline\end{array}^T$ the responses in positions d, e, and f are $d' = g-a$, $e' = h-b$, and $f' = i-c$. When we then apply the mask $\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\\hline\end{array}$ the response in position e is $e'' = (g-a)+2(h-b)+(i-c) = (g+2h+i)-(a+2b+c) = e^*$. G_y mask can be used in the same way.

d) The 3×3 discrete Laplace-operator is

$$\begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\\hline -1 & 4 & -1 \\\hline 0 & -1 & 0 \\\hline\end{array}$$

In order to separate the 3×3 discrete Laplace-operator, we must find two 3×1 vectors $\begin{array}{|c|c|c|}\hline a & b & c \\\hline\end{array}^T$ and $\begin{array}{|c|c|c|}\hline d & e & f \\\hline\end{array}^T$ whose outer product

$$\begin{array}{|c|c|c|}\hline ad & bd & cd \\\hline ae & be & ce \\\hline af & bf & cf \\\hline\end{array}$$

were the Laplace mask. For example, $ad = 0$. If we choose $a = 0$ then also $ae = 0$ which is not valid. If $d = 0$ then $bd = 0$ which is also not valid. So, we cannot separate the mask into two one-dimensional vectors.

T-61.5100 Digital image processing, Exercise 6, Nov 5, 2013

Morphological image processing

1. How can the given object be cleaned up by using morphological operations? (The outline of the "box" in the image should be closed.)

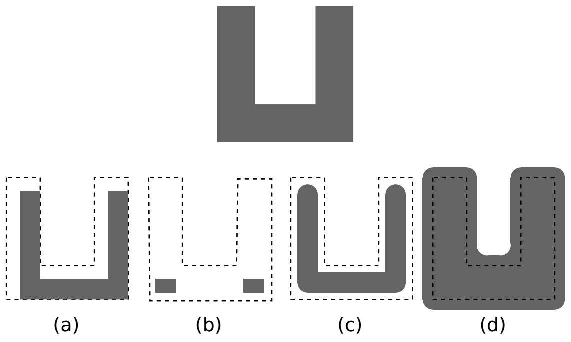
0	0	0	0	0	0	0	0	0
0	0	1	1	0	1	1	0	0
0	0	1	0	1	0	0	1	0
0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0

demo 2. (a) Give a morphological algorithm for converting an 8-connected binary boundary to an m-connected boundary. You may assume that the boundary is fully connected and that it is one pixel thick.

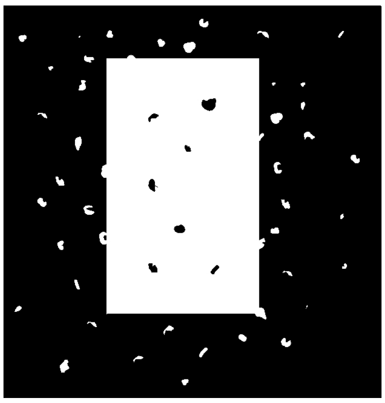
(b) Does the operation of your algorithm require more than one iteration with each structuring element? Explain your reasoning.

(c) Is the performance of your algorithm independent of the order in which the structuring elements are applied? If your answer is yes, prove it; otherwise give an example that illustrates the dependence of your procedure on the order of application of the structuring elements.

3. Give the structuring element and morphological operation(s) that produced each of the results shown in images (a) through (d). Show the origin of each structuring element clearly. The dashed lines show the boundary of the original set and are included only for reference. Note that in (d) all corners are rounded.



4. Sketch or explain what the sets C, D, E, F would look like when the following sequence of operations is applied to a given image: $C = A \ominus B$; $D = C \oplus B$; $E = D \oplus B$; $F = E \ominus B$, where B is the structuring element. The initial set A consists of all the image components shown in white. Note that this sequence of operations is simply the opening of A by B, followed by the closing of that opening by B. You may assume that B is round and just large enough to enclose each of the noise components.



T-61.5100 Digital image processing, Exercise 6, Nov 5, 2013

1.

One solution is to use the morphological operation called "closing". Closing is defined as dilation followed by erosion:

$$A \bullet B = (A \oplus B) \ominus B.$$

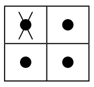
Dilation of A by B is the set of all points z such that $\hat{B}$ (B's reflection about its origin) and A overlap by at least one element,

$$A \oplus B = \{z | [(\hat{B})_z \cap A] \subseteq A\}.$$

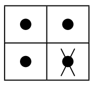
Erosion of A by B is the set of all points z such that B is contained in A,

$$A \ominus B = \{z | (B)_z \subseteq A\}.$$

The structuring element B that we choose for this exercise is



The origin is marked with the cross. B's reflection about its origin, is then called $\hat{B}$:



The result of dilation of the image A by B is

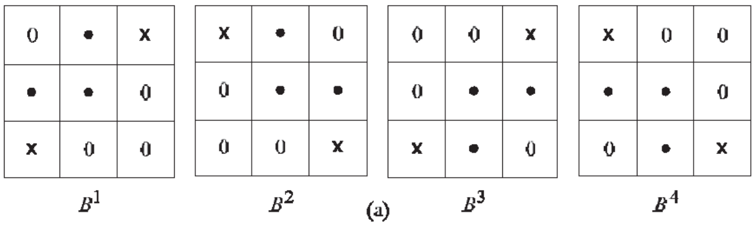
0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1
0	0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	0

The previous image after erosion is

0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0
0	0	1	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0

2.

a) With reference to the discussion in Section 2.5.2, m-connectivity is used to avoid multiple paths that are inherent in 8-connectivity. In one-pixel-thick, fully connected boundaries, these multiple paths manifest themselves in the four basic patterns shown here:



0	•	x
•	•	0
x	0	0

x	•	0
0	•	•
0	0	x

0	0	x
0	•	•
x	•	0

x	0	0
•	•	0
0	•	x

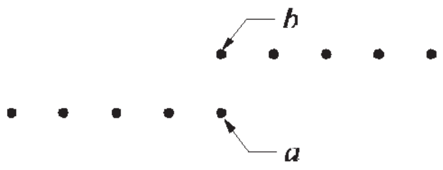
The solution to the problem is to use the hit-or-miss transform to detect the patterns and then to change the center pixel to 0, thus eliminating the multiple paths. A basic sequence of morphological steps to accomplish this is as follows:

$$\begin{aligned}
X_1 &= A \circledast B^1 \\
Y_1 &= A \cap X_1^c \\
X_2 &= Y_1 \circledast B^2 \\
Y_2 &= Y_1 \cap X_2^c \\
X_3 &= Y_2 \circledast B^3 \\
Y_3 &= Y_2 \cap X_3^c \\
X_4 &= Y_3 \circledast B^4 \\
Y_4 &= Y_3 \cap X_4^c
\end{aligned}$$

where A is the input image containing the boundary.

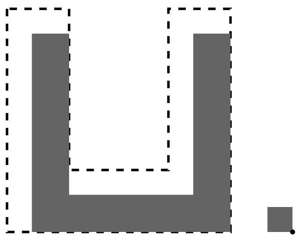
b) Only one pass is required. Application of the hit-or-miss transform using a given B^i finds all instances of occurrence of the pattern described by that structuring element.

c) The order does matter. For example, consider the sequence of points shown in next figure and assume that we are traveling from left to right. If B^1 is applied first, point a will be deleted and point b will remain after application of all other structuring elements. If, on the other hand, B^3 is applied first, point b will be deleted and point a will remain. Thus, we would end up with different (but of course, acceptable) m-paths.

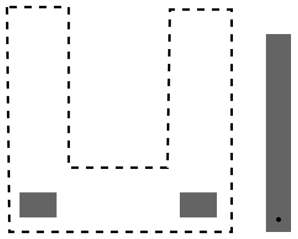


3.

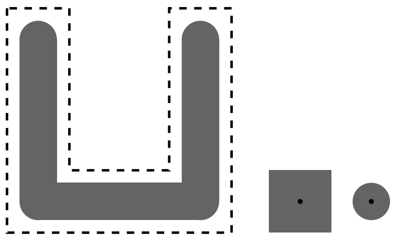
(a) Erode by a small rectangular structuring element. Note that the origin of the element is in the lower right corner (shown by small black dot.)



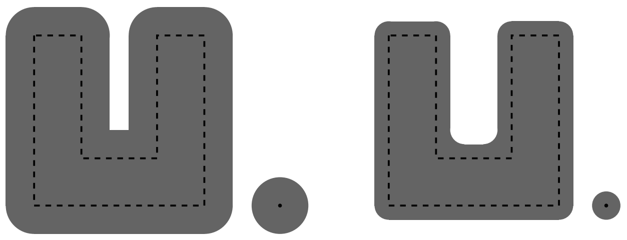
(b) Erode the original set with the tall rectangular structuring element shown.



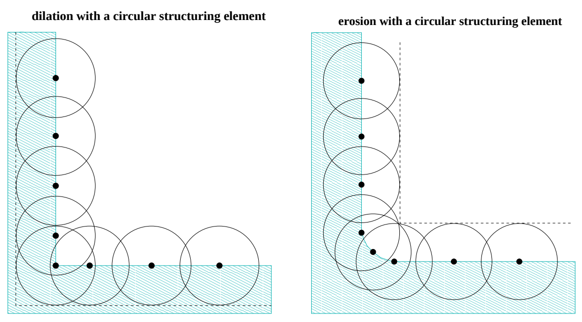
(c) First erode the image shown down to thin lines using the rectangular structuring element. Then dilate this result with the circular structuring element.



(d) First dilate the original set with the large disk shown. Then erode the resulting image with a disk of half the diameter of the disk used for dilation.



Below are zoomed in images of how the dilation and erosion of an angle occurs with a circular element. Again, the dashed line is the original border and the new border is filled.


dilation with a circular structuring element
erosion with a circular structuring element

4.

The solution is shown in the next figure. Although the images shown could be sketched by hand, they were done in MATLAB. The structuring element was chosen to be just large enough to encompass all the noise elements, as given in the problem statement.

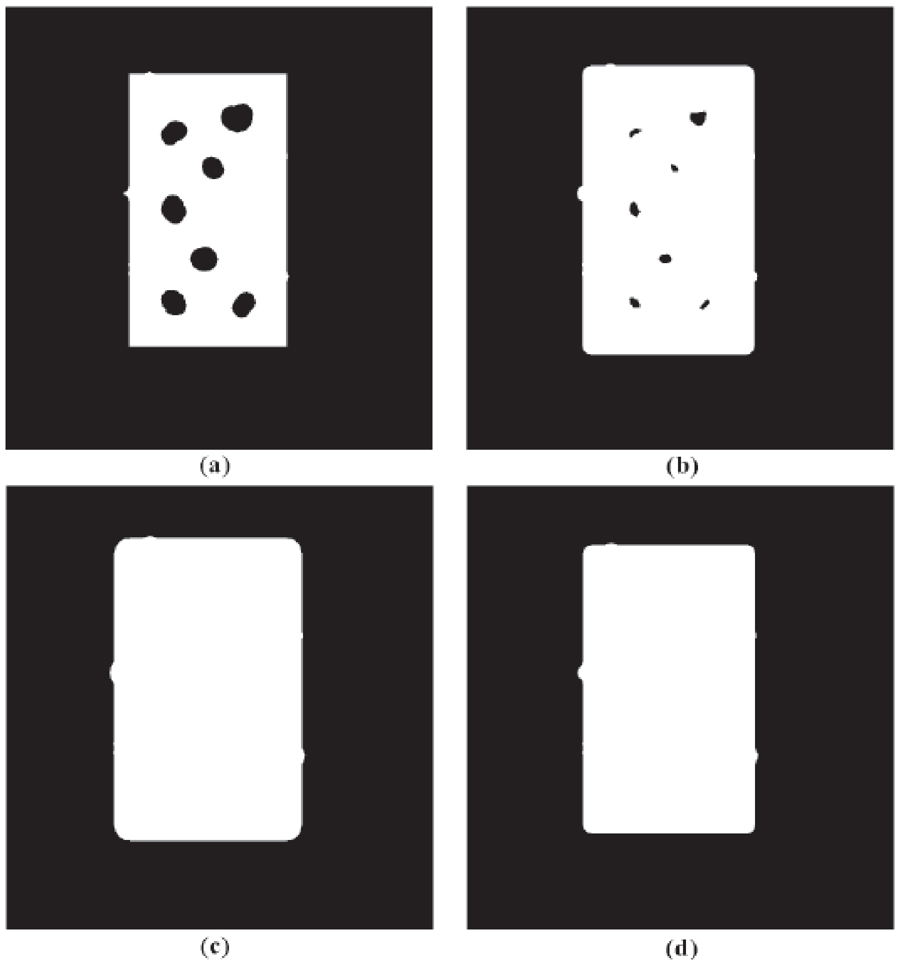
The images shown in the figure are: (a) erosion of the original, (b) dilation of the result, (c) another dilation, and finally (d) an erosion.

The first erosion (image (a)) should take out all noise elements that do not touch the rectangle, should increase the size of the noise elements completely contained within the rectangle, and should decrease the size of the rectangle. If worked by hand, one might not realise that some imperfections are left along the boundary of the object. In this case, this is not an important issue because it is scale-dependent, and nothing is said in the problem statement about this.

The first dilation (image (b)) should shrink the noise components that were increased in erosion, should increase the size of the rectangle, and should round the corners.

The next dilation (c) should eliminate the internal noise components completely and further increase the size of the rectangle.

The final erosion (image (d)) should then decrease the size of the rectangle back to the original size. Note, however, the rounded corners!



T-61.5100 Digital image processing, Exercise 7, Nov 12, 2013

Wavelets

1. Construct a fully populated approximation pyramid and corresponding prediction residual pyramid for the image
$$\mathbf{F} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}.$$
Use 2×2 block neighbourhood averaging for the approximation filter and omit the interpolation filter (see Figure 7.2 in the textbook).

demo 2. Compute the one-dimensional discrete wavelet transform (DWT) of function $f(0) = 1$, $f(1) = 4$, $f(2) = -3$, and $f(3) = 0$ with starting scale $j_0 = 1$. Then compute the inverse transform.

3. Show how $\varphi_{1,0}(x)$ and $\varphi_{1,1}(x)$ can be formed using $\varphi_{0,0}(x)$ and $\psi_{0,0}(x)$.

demo 4. From $\varphi_{0,0}(x)$, $\psi_{0,0}(x)$, $\psi_{1,0}(x)$ and $\psi_{1,1}(x)$ form the transform matrix $\mathbf{H}$ that can be used in multiresolution processing of a 4×4-sized image.

5. By using the matrix from the previous exercise, calculate the transform $\mathbf{T} = \mathbf{HFH}^T$, where the analysed image is
$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}.$$

6. Compute the Haar transform $\mathbf{T} = \mathbf{HFH}^{\mathrm{T}}$ of the 2×2 image
$$\mathbf{F} = \begin{bmatrix} 3 & -1 \\ 6 & 2 \end{bmatrix}.$$
Also compute the inverse Haar transform $\mathbf{F} = \mathbf{H}^{\mathrm{T}}\mathbf{TH}$ of the obtained result.

7. Draw wavelet $\psi_{3,3}(x)$ for the Haar wavelet function. Write an expression for $\psi_{3,3}(x)$ in terms of the Haar scaling function.

T-61.5100 Digital image processing, Exercise 7, Nov 12, 2013

1.

A mean approximation pyramid is formed by forming 2×2 block averages. Since the starting image is of size 4×4, $J = 2$, and $\mathbf{F}$ is placed at level 2 of the mean approximation pyramid. The level 1 and level 0 approximations are achieved by taking 2×2 block averages over $\mathbf{F}$ and subsampling. The completed mean approximation pyramid is

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \begin{bmatrix} 3.5 & 5.5 \\ 11.5 & 13.5 \end{bmatrix} \begin{bmatrix} 8.5 \end{bmatrix}.$$

Since no interpolation filtering is specified, pixel replication is used in the generation of the mean prediction residual pyramid levels. Level 0 of the prediction residual pyramid is the lowest resolution approximation, [8.5]. The level 2 prediction residual is obtained by upsampling the level 1 approximation and subtracting it from the level 2 (original image). Thus, we get

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} - \begin{bmatrix} 3.5 & 3.5 & 5.5 & 5.5 \\ 3.5 & 3.5 & 5.5 & 5.5 \\ 11.5 & 11.5 & 13.5 & 13.5 \\ 11.5 & 11.5 & 13.5 & 13.5 \end{bmatrix} = \begin{bmatrix} -2.5 & -1.5 & -2.5 & -1.5 \\ 1.5 & 2.5 & 1.5 & 2.5 \\ -2.5 & -1.5 & -2.5 & -1.5 \\ 1.5 & 2.5 & 1.5 & 2.5 \end{bmatrix}.$$

Similarly, the level 1 prediction residual is obtained by upsampling the level 0 approximation and subtracting it from the level 1 approximation.

$$\begin{bmatrix} 3.5 & 5.5 \\ 11.5 & 13.5 \end{bmatrix} - \begin{bmatrix} 8.5 & 8.5 \\ 8.5 & 8.5 \end{bmatrix} = \begin{bmatrix} -5 & -3 \\ 3 & 5 \end{bmatrix}.$$

The mean prediction residual pyramid is therefore

$$\begin{bmatrix} -2.5 & -1.5 & -2.5 & -1.5 \\ 1.5 & 2.5 & 1.5 & 2.5 \\ -2.5 & -1.5 & -2.5 & -1.5 \\ 1.5 & 2.5 & 1.5 & 2.5 \end{bmatrix} \begin{bmatrix} -5 & -3 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} 8.5 \end{bmatrix}.$$

2.

The DWT transform pair is given as

$$W_\varphi(j_0,k) = \frac{1}{\sqrt{M}}\sum_x f(x)\varphi_{j_0,k}(x)$$
$$W_\psi(j,k) = \frac{1}{\sqrt{M}}\sum_x f(x)\psi_{j,k}(x),$$

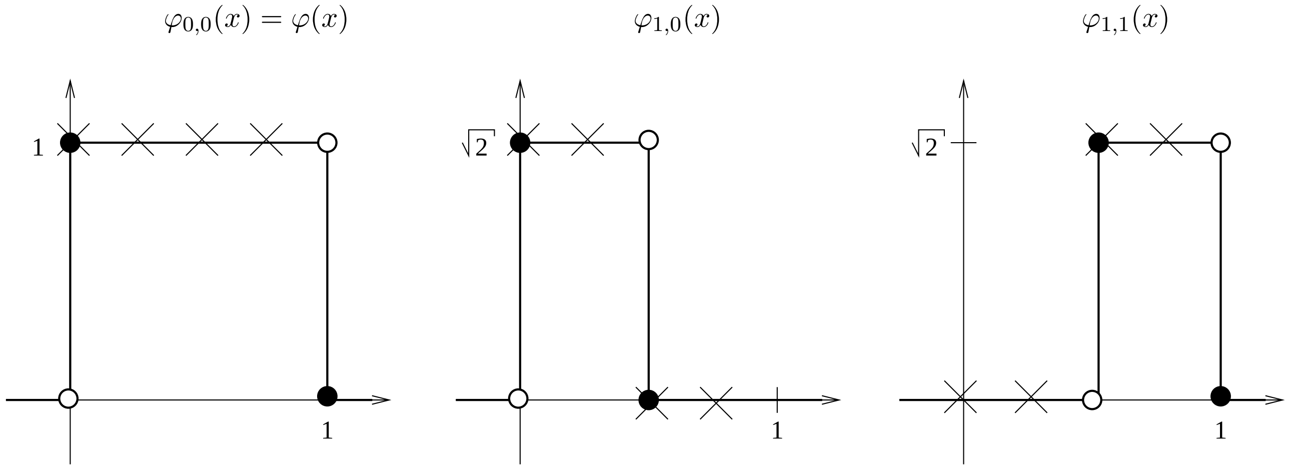
and the inverse transform as

$$f(x) = \frac{1}{\sqrt{M}}\sum_k W_\varphi(j_0,k)\varphi_{j_0,k}(x) + \frac{1}{\sqrt{M}}\sum_{j=j_0}^{\infty}\sum_k W_\psi(j,k)\psi_{j,k}(x).$$

Now $M = 4$, $J = 2$, and $j_0 = 1$, so the summations in the above formulas are performed over $x = 0,1,2,3$, $j = 1$, and $k = 0,1$. Using Haar functions and assuming that they are distributed over the range of the input sequence, we get for the scaling functions

$$\begin{aligned} \varphi_{1,0}(x) &= \sqrt{2}\varphi(2x) \\ \varphi_{1,1}(x) &= \sqrt{2}\varphi(2x-1), \\ \text{where}\quad \varphi(x) &= \begin{cases} 1 & 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

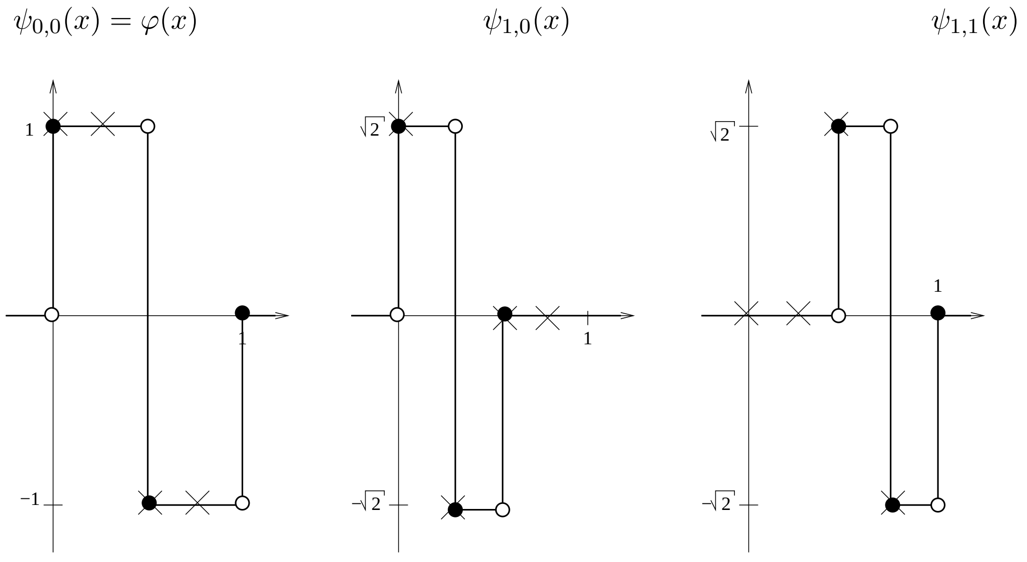
and below is shown the three functions with the four sampling points, i.e. from where we should read values for $x = 0,1,2,3$:



Similarly for the wavelet functions

$$\begin{aligned} \psi_{1,0}(x) &= \sqrt{2}\psi(2x) \\ \psi_{1,1}(x) &= \sqrt{2}\psi(2x-1), \\ \text{where} \quad \psi(x) &= \begin{cases} 1 & 0 \le x < 0.5 \\ -1 & 0.5 \le x < 1 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

and the functions with the four sampling points:



Now we can calculate the transform,

$$\begin{aligned} W_\varphi(1,0) &= \frac{1}{2}[f(0)\varphi_{1,0}(0) + f(1)\varphi_{1,0}(1) + f(2)\varphi_{1,0}(2) + f(3)\varphi_{1,0}(3)] \\ &= \frac{1}{2}[(1)(\sqrt{2}) + (4)(\sqrt{2}) + (-3)(0) + (0)(0)] = \frac{5\sqrt{2}}{2} \\ W_\varphi(1,1) &= \frac{1}{2}[f(0)\varphi_{1,1}(0) + f(1)\varphi_{1,1}(1) + f(2)\varphi_{1,1}(2) + f(3)\varphi_{1,1}(3)] \\ &= \frac{1}{2}[(1)(0) + (4)(0) + (-3)(\sqrt{2}) + (0)(\sqrt{2})] = \frac{-3\sqrt{2}}{2} \\ W_\psi(1,0) &= \frac{1}{2}[f(0)\psi_{1,0}(0) + f(1)\psi_{1,0}(1) + f(2)\psi_{1,0}(2) + f(3)\psi_{1,0}(3)] \\ &= \frac{1}{2}[(1)(\sqrt{2}) + (4)(-\sqrt{2}) + (-3)(0) + (0)(0)] = \frac{-3\sqrt{2}}{2} \\ W_\psi(1,1) &= \frac{1}{2}[f(0)\psi_{1,1}(0) + f(1)\psi_{1,1}(1) + f(2)\psi_{1,1}(2) + f(3)\psi_{1,1}(3)] \\ &= \frac{1}{2}[(1)(0) + (4)(0) + (-3)(\sqrt{2}) + (0)(-\sqrt{2})] = \frac{-3\sqrt{2}}{2} \end{aligned}$$

so that the DWT is $5\sqrt{2}/2, \ -3\sqrt{2}/2, \ -3\sqrt{2}/2, \ -3\sqrt{2}/2$.

The inverse transform is then calculated:

$$f(x) = \frac{1}{2}[W_\varphi(1,0)\varphi_{1,0}(x) + W_\varphi(1,1)\varphi_{1,1}(x) + W_\psi(1,0)\psi_{1,0}(x) + W_\psi(1,1)\psi_{1,1}(x)],$$

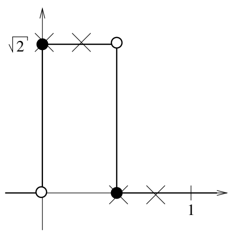
so we get

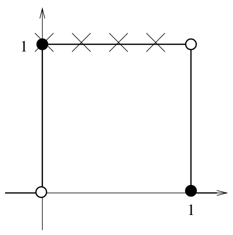
$$\begin{aligned}
f(0) &= \frac{\sqrt{2}}{4}[(5)(\sqrt{2}) + (-3)(0) + (-3)(\sqrt{2}) + (-3)(0)] = \frac{2(\sqrt{2})^2}{4} = 1 \\
f(1) &= \frac{\sqrt{2}}{4}[(5)(\sqrt{2}) + (-3)(0) + (-3)(-\sqrt{2}) + (-3)(0)] = \frac{8(\sqrt{2})^2}{4} = 4 \\
f(2) &= \frac{\sqrt{2}}{4}[(5)(0) + (-3)(\sqrt{2}) + (-3)(0) + (-3)(\sqrt{2})] = \frac{-6(\sqrt{2})^2}{4} = -3 \\
f(3) &= \frac{\sqrt{2}}{4}[(5)(0) + (-3)(\sqrt{2}) + (-3)(0) + (-3)(-\sqrt{2})] = \frac{0(\sqrt{2})^2}{4} = 0
\end{aligned}$$

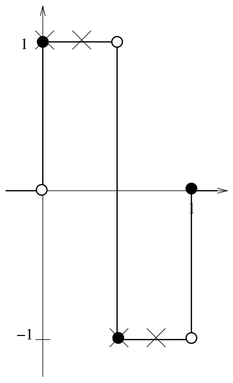
3.

How can we express $\varphi_{1,0}(x)$ as a weighted sum of $\varphi_{0,0}(x)$ and $\psi_{0,0}(x)$, i.e. what is a and b in

$$\varphi_{1,0}(x) \quad = \quad a \quad \cdot \quad \varphi_{0,0}(x) \quad + \quad b \quad \cdot \quad \psi_{0,0}(x), \qquad \text{or}$$



$= \quad a$


$+ \quad b$


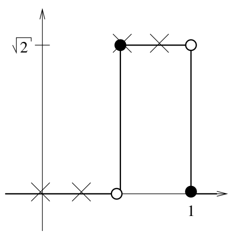
?

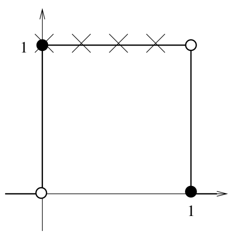
The answer is $a = b = \frac{\sqrt{2}}{2}$. Let's check some points:

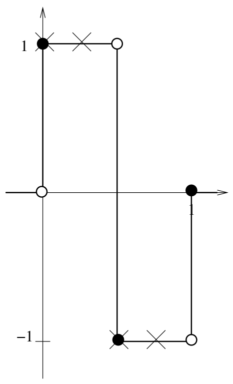
$$\begin{aligned}
\varphi_{1,0}(0) &= \frac{\sqrt{2}}{2}\varphi_{0,0}(0) + \frac{\sqrt{2}}{2}\psi_{0,0}(0) = \frac{\sqrt{2}}{2}1 + \frac{\sqrt{2}}{2}1 = \sqrt{2},\\
\varphi_{1,0}(\frac{1}{2}) &= \frac{\sqrt{2}}{2}\varphi_{0,0}(\frac{1}{2}) + \frac{\sqrt{2}}{2}\psi_{0,0}(\frac{1}{2}) = \frac{\sqrt{2}}{2}1 + \frac{\sqrt{2}}{2}(-1) = 0,\\
\varphi_{1,0}(1) &= \frac{\sqrt{2}}{2}\varphi_{0,0}(1) + \frac{\sqrt{2}}{2}\psi_{0,0}(1) = \frac{\sqrt{2}}{2}0 + \frac{\sqrt{2}}{2}0 = 0.
\end{aligned}$$

Now, what about $\varphi_{1,1}(x)$ similarly as a weighted sum,

$$\varphi_{1,1}(x) \quad = \quad a \quad \cdot \quad \varphi_{0,0}(x) \quad + \quad b \quad \cdot \quad \psi_{0,0}(x), \qquad \text{or}$$



$= \quad a$


$+ \quad b$


?

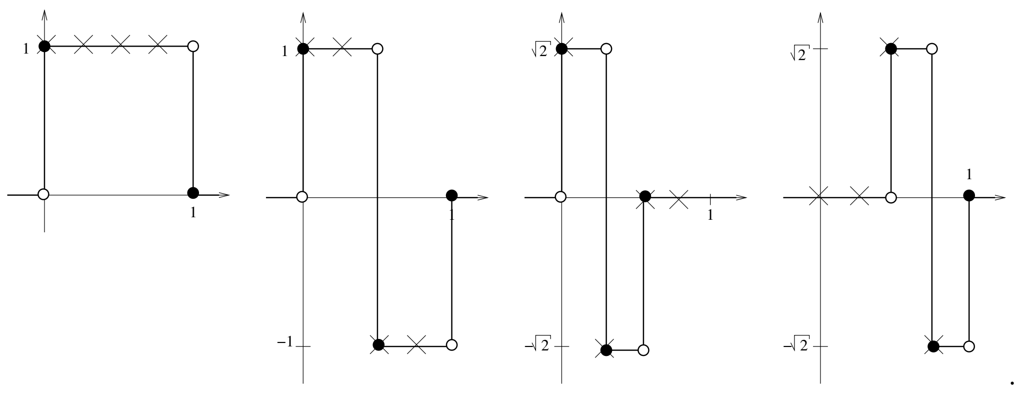
The answer is $a = \frac{\sqrt{2}}{2}$, $b = -\frac{\sqrt{2}}{2}$. Some checks:

$$\begin{aligned}
\varphi_{1,0}(0) &= \frac{\sqrt{2}}{2}\varphi_{0,0}(0) - \frac{\sqrt{2}}{2}\psi_{0,0}(0) = \frac{\sqrt{2}}{2}1 - \frac{\sqrt{2}}{2}1 = 0,\\
\varphi_{1,0}(\frac{1}{2}) &= \frac{\sqrt{2}}{2}\varphi_{0,0}(\frac{1}{2}) - \frac{\sqrt{2}}{2}\psi_{0,0}(\frac{1}{2}) = \frac{\sqrt{2}}{2}1 - \frac{\sqrt{2}}{2}(-1) = \sqrt{2},\\
\varphi_{1,0}(1) &= \frac{\sqrt{2}}{2}\varphi_{0,0}(1) - \frac{\sqrt{2}}{2}\psi_{0,0}(1) = \frac{\sqrt{2}}{2}0 - \frac{\sqrt{2}}{2}0 = 0.
\end{aligned}$$

4.

The transform matrix $\mathbf{H}$ for a 4×4 image is formed from sampling four points of each function,

$$\varphi_{0,0}(x) \qquad \psi_{0,0}(x) \qquad \psi_{1,0}(x) \qquad \psi_{1,1}(x),$$



$$\mathbf{H} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix}$$

5.

Finally we can calculate the transform matrix $\mathbf{T}$ by inserting $\mathbf{H}$ and the given $\mathbf{F}$ into the equation:

$$\begin{aligned}
\mathbf{T} &= \mathbf{HFH}^T \\
&= \left(\frac{1}{2}\right)^2 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & \sqrt{2} & 0 \\ 1 & 1 & -\sqrt{2} & 0 \\ 1 & -1 & 0 & \sqrt{2} \\ 1 & -1 & 0 & -\sqrt{2} \end{bmatrix} \\
&= \frac{1}{4} \begin{bmatrix} 3 & 3 & 3 & 0 \\ -1 & -1 & -1 & 0 \\ -\sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & \sqrt{2} & 0 \\ 1 & 1 & -\sqrt{2} & 0 \\ 1 & -1 & 0 & \sqrt{2} \\ 1 & -1 & 0 & -\sqrt{2} \end{bmatrix} \\
&= \frac{1}{4} \begin{bmatrix} 9 & 3 & 0 & 3\sqrt{2} \\ -3 & -1 & 0 & -\sqrt{2} \\ -3\sqrt{2} & -\sqrt{2} & 0 & -2 \\ 0 & 0 & 0 & 0 \end{bmatrix}.
\end{aligned}$$

6.

The 2×2 Haar transformation matrix is

$$\mathbf{H}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Then we get

$$\mathbf{T} = \mathbf{HFH}^{\mathrm{T}} = (\frac{1}{\sqrt{2}})^2 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 6 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 5 & 4 \\ -3 & 0 \end{bmatrix}.$$

Next we compute the inverse Haar transform. First, the transpose of the 2×2 Haar transformation matrix is computed:

$$\mathbf{H}^{\mathrm{T}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \mathbf{H}.$$

Then,

$$\mathbf{F} = \mathbf{H}^{\mathrm{T}}\mathbf{TH} = (\frac{1}{\sqrt{2}})^2 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 5 & 4 \\ -3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ 6 & 2 \end{bmatrix}.$$

7.

The set $\{\psi_{j,k}(x)\}$ of wavelets is defined as

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k).$$

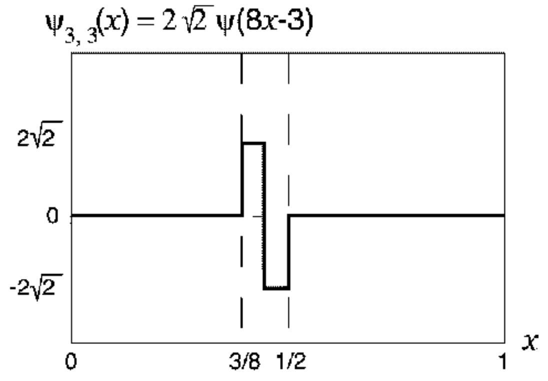
From this definition we obtain

$$\psi_{3,3}(x) = 2^{3/2}\psi(2^3 x - 3) = 2\sqrt{2}\psi(8x - 3).$$

Using the Haar wavelet function definition,

$$\psi(x) = \begin{cases} 1 & 0 \le x < 0.5 \\ -1 & 0.5 \le x < 1 \\ 0 & \text{elsewhere,} \end{cases}$$

we obtain the following plot.



To express $\psi_{3,3}(x)$ as a function of scaling functions, we notice that any wavelet function can be expressed as a sum of shifted, double-resolution scaling functions,

$$\psi(x) = \sum_n h_\psi(n)\sqrt{2}\varphi(2x - n).$$

Employing this result and the Haar wavelet vector defined in the textbook in Example 7.6 ($h_\psi(0) = 1/\sqrt{2}$ and $h_\psi(1) = -1/\sqrt{2}$), we get

$$\psi(8x-3) = \sum_n h_\psi(n)\sqrt{2}\varphi(2(8x-3)-n) = \frac{1}{\sqrt{2}}\sqrt{2}\varphi(16x-6)+\frac{-1}{\sqrt{2}}\sqrt{2}\varphi(16x-7) = \varphi(16x-6)-\varphi(16x-7).$$

Then, since $\psi_{3,3} = 2\sqrt{2}\psi(8x-3)$,

$$\psi_{3,3} = 2\sqrt{2}\psi(8x-3) = 2\sqrt{2}\varphi(16x-6) - 2\sqrt{2}\varphi(16x-7).$$

T-61.5100 Digital image processing, Exercise 8, Nov 12, 2013

Wavelets and multiresolution processing

This computer exercise assumes that you already know the basics of image processing in Matlab or Octave. E.g. opening an image (`imread`) and showing it on the screen (`imshow` or `showim` provided with the images). If you have forgotten something you can always check back to the first two computer exercises (exercise no 2 and no 4 on the course Noppa page).

1. Image pyramids

Open the image `livingroom.png` (which displays a very "modern" living room). The task is to create the first four levels of an image pyramid starting with the full 256×256 and sampling down three times. See the system scheme in Fig. 7.2 (included as `fig7_2.png` in ZIP package). For the approximation we use 2×2 neighbourhood averaging, i.e. convolve (`conv2`) with a 2×2 averaging mask. For the prediction residual you can simply use the difference between the original image and the averaged one (i.e. no need to first downsample and then upsample).

For repetitive actions you can use a for loop in Matlab:

```
for i=1:N
  ...
end
```

where `i` will loop over the values from 1 to `N`. Replace the "..." with whatever should be done for each value of `i`, i.e. `i=1`, `i=2`, and so on up to `i=N`.

A useful Matlab/Octave function in this case is `downsample(X,2)` which downsamples the matrix `X` along each column, i.e. it removes every second row. For example:

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ \cancel{5} & \cancel{6} & \cancel{7} & \cancel{8} \\ 9 & 10 & 11 & 12 \\ \cancel{13} & \cancel{14} & \cancel{15} & \cancel{16} \end{array}$$

When you need to downsample along the rows instead you simply transpose the input and output by using Matlabs transpose-operator (`'`) like this:

```
downsample(X',2)'
```

Example:

$$\begin{array}{cccc} 1 & \cancel{2} & 3 & \cancel{4} \\ 5 & \cancel{6} & 7 & \cancel{8} \\ 9 & \cancel{10} & 11 & \cancel{12} \\ 13 & \cancel{14} & 15 & \cancel{16} \end{array}$$

2. Two-dimensional discrete wavelet transform

There is a ready-made package in Matlab for wavelet processing, but it is not easy to understand what it is doing "behind the scenes". It is much more instructive and fun (?) to do it "by hand"!

You can try on any of the downloaded images (`livingroom.png`, `cameraman.png`, `weird.png`), and try them all once you have finished your program. Look at the diagram in Fig.7.24 how to do it (included as `fig7_24.png` in ZIP package).

We start with the simplest of wavelet functions, the Haar function. For the Haar function set the following variables:

```
ld = [1 1]/sqrt(2);
hd = [-1 1]/sqrt(2);
```

The `ld` and `hd` are the familiar low-pass and high-pass filters that make up the Haar transform. They are however order-reversed (due to the convolution) as is also indicated by the negative indices in the DWT diagram.

Now just follow the scheme in the diagram. The $\star h_\psi$ indicates convolution with the `hd` variable, while $\star h_\varphi$ indicates convolution with the `ld`. Remember that in the first step we are doing the convolution and downsampling along the rows, i.e. every second column is removed after the downsampling. In the second step we are working along the columns.

Just to get you going, here is how to do the first two steps in the diagram scheme. The top branch does convolution between the input image and the high-pass filter h_ψ, and then downsampling along the rows:

```
Ih = downsample(conv2(I, hd, 'same')',2)';
```

The bottom branch does convolution with the low-pass filter h_φ and then downsampling:

```
Il = downsample(conv2(I, ld, 'same')',2)';
```

Now you have to do the following steps yourself. Remember that the convolutions and downsampling should now be done along the columns!

After this plot the resulting one-scale decomposition, i.e. the four matrices $W_\varphi, W_\psi^H, W_\psi^V$ and W_ψ^D. If you have done everything correctly the first one should be a smaller version of the original (the approximation). The second should show horizontal details (such as vertical lines), the third one vertical details, and the last one diagonal components (but this is hard to see).

The last step is just to reconstruct the image again from its wavelet components by doing the inverse transform as in Fig. 7.24(c). This is now relatively easy, just "reverse" the thinking of the forward transform. You will need the `upsample(X,2)` function which works as one would expect. It inserts zeros along each column. The coefficients for the reconstruction Haar wavelet are:

```
lr = [1 1]/sqrt(2);
hr = [1 -1]/sqrt(2);
```

Display the original image and the reconstructed image and compare them. If everything goes well they should look the same.

If you have time you can try more complex wavelets, e.g. simply by changing the values of the decomposition coefficients (`ld`, `hd`) and reconstruction coefficients (`lr`, `hr`). These can be easily copied and pasted into your Matlab-code from the web page http://wavelets.pybytes.com/.

T-61.5100 Digital image processing, Exercise 9, Nov 26, 2013

Image compression

1. An image has 8 grey scale values $a_1, \ldots a_8$, whose probabilities are 0.6, 0.2, 0.08, 0.06, 0.02, 0.02, 0.01 and 0.01, respectively. Construct the Huffman code for this image, considering the fixed grey scale values as the symbols. Calculate the entropy and compare it to the average word length. Also calculate the efficiency and the compression ratio when compared to the plain binary 3-bit code.

demo 2. Devise an algorithm for decoding the following LZW encoded line:

   39 39 126 126 256 258 260 259 257 126

   Since the dictionary that was used during encoding is not available, the code book must be reproduced as the output is decoded.

3. Show the bit planes for the following 4×4 image in both direct binary code and in Gray code. Only three bits are needed. Is there a significant difference between the results? If so, why?

1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

4. Predictive coding (Fig 8.41) for the grey level line

   30 29 29 28 20 15 12 10 9 8 9 10 11 11 11 11 11

   Create delta modulation code (DM), when $\alpha = 1$ and error is coded using values ± 2. Compare required number of bits to corresponding lossless coding. What are the disadvantages of DM?

T-61.5100 Digital image processing, Exercise 9, Nov 24, 2013

1.

The Huffman code is constructed as follows. In the beginning, each symbol is its own "tree" whose weight is the probability of the symbol. Then with each step we combine two "lightest" trees (if there are several possibilities, we can choose any of them). The weight of the new tree is the sum of the weights of the combined trees. This procedure is repeated until we end up with only one tree. See Figure 1.

Now the binary code of each symbol is the path from the root node to the symbol node. The top path is marked with '0' and to the bottom path with '1'. So we obtain:

$$\begin{aligned}
a_1 &= 0\\
a_2 &= 10\\
a_3 &= 111\\
a_4 &= 1100\\
a_5 &= 11011\\
a_6 &= 110100\\
a_7 &= 1101010\\
a_8 &= 1101011
\end{aligned}$$

The entropy is

$$H = -\sum_{i=1}^{8} P(a_i)\log_2 P(a_i) = -[0.6\log_2 0.6 + \cdots + 0.01\log_2 0.01] = 1.80 \text{ bits/symbol}$$

and the average word length is:

$$\begin{aligned}
L_{\text{avg}} &= \sum_k l(r_k)p(r_k)\\
&= 0.6\cdot 1 + 0.2\cdot 2 + 0.08\cdot 3 + 0.06\cdot 4 + 0.02\cdot 5 + 0.02\cdot 6 + 0.01\cdot 7 + 0.01\cdot 7\\
&= 1.84.
\end{aligned}$$

It can be seen that the average word length of the Huffman code is quite close to the entropy, which is the smallest code word length theoretically possible (Shannon's first theorem). The efficiency of the coding tells us how close we are to the theoretical minimum:

$$\eta = \frac{nH}{L_{avg}} = \frac{1\cdot 1.80}{1.84} \approx 0.98$$

In fact the Huffman code is optimal when we are restricted to encode one symbol at a time ($n = 1$).

We can also calculate the compression ratio when moving from a representation of $b = 3$ bits to one of on average $b' = 1.84$ bits:

$$C = \frac{b}{b'} = \frac{3}{1.84} \approx 1.63.$$

and efficiency:

$$\eta = \frac{nH}{b} = \frac{1\cdot 1.80}{3} \approx 0.56.$$

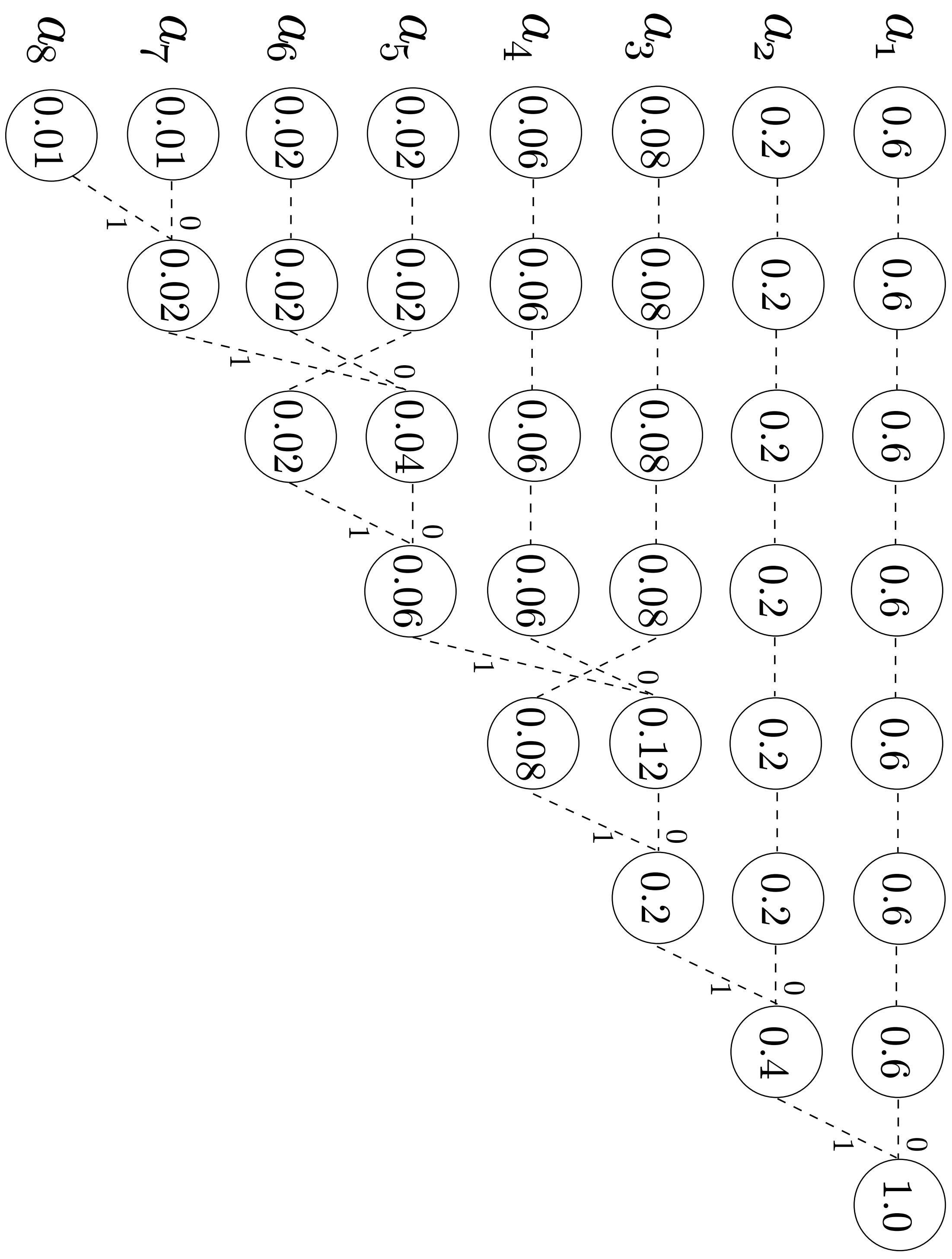


Figure 1: The Huffman tree in exercise 1.

2.

The input to the LZW decoding algorithm is

39 39 126 126 256 258 260 259 257 126

The starting dictionary, to be consistent with the coding itself, contains 512 locations - with the first 256 corresponding to grey level values 0 through 255. The decoding algorithm begins by getting the first encoded value, outputting the corresponding value from the dictionary, and setting the "recognized sequence" to the first value. For each additional encoded value, we

(1) output the dictionary entry for the pixel value(s),

(2) add a new dictionary entry whose content is the "recognized sequence" plus the first element of the encoded value being processed, and

(3) set the "recognized sequence" to the encoded value being processed.

For the given input, the sequence of operations is:

Recognized	Encoded Value	Pixels	Dict. Address	Dict. Entry
	39	39		
39	39	39	256	39-39
39	126	126	257	39-126
126	126	126	258	126-126
126	256	39-39	259	126-39
256	258	126-126	260	39-39-126
258	260	39-39-126	261	126-126-39
260	259	126-39	262	39-39-126-126
259	257	39-126	263	126-39-39
257	126	126	264	39-126-126

Note, for example, in row 5 of the table that the new dictionary entry for location 259 is 126-39, the concatenation of the currently recognized sequence, 126, and the first element of the encoded value being processed - the 39 from the 39-39 entry in dictionary location 256. The output is then read from the third column of the table to yield

$$\begin{matrix} 39 & 39 & 126 & 126 \\ 39 & 39 & 126 & 126 \\ 39 & 39 & 126 & 126 \\ 39 & 39 & 126 & 126 \end{matrix}$$

where it is assumed that the decoder knows or is given the size of the image that was received. Note that the dictionary is generated as the decoding is carried out.

3.

First we need the direct binary and Gray codes. The Gray code can be generated from the direct code by using the exclusive OR operator $\oplus$:

$$\begin{aligned} g_{m-1} &= a_{m-1} \\ g_i &= a_i \oplus a_{i+1} \quad 0 \le i \le m-2 \end{aligned}$$

decimal	direct	Gray
1	001	001
2	010	011
3	011	010
4	100	110

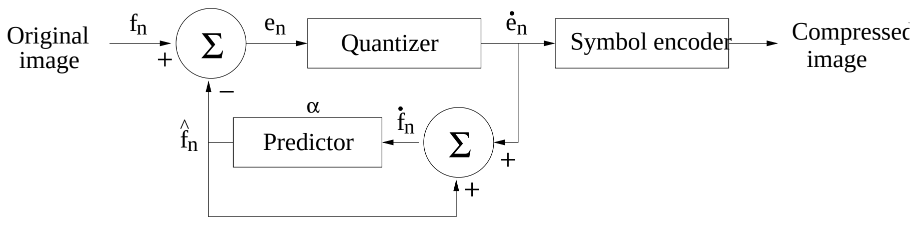
original image

1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

	direct code				**Gray code**			
bit 2	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1
bit 1	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1
	0	0	0	0	1	1	1	1
bit 0	1	1	1	1	1	1	1	1
	0	0	0	0	1	1	1	1
	1	1	1	1	0	0	0	0
	0	0	0	0	0	0	0	0

4.

The encoder:



The predictor:

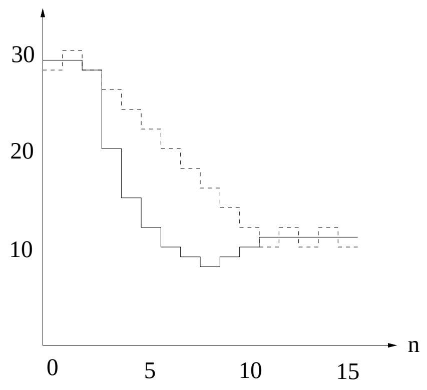
$$\hat{f}_n = \alpha \dot{f}_{n-1}, \quad \alpha = 1$$

The quantizer:

$$\dot{e}_n = \begin{cases} +2, & \text{when } e_n \geq 0 \\ -2, & \text{when } e_n < 0 \end{cases}$$

The initial values are $f_0 = \dot{f}_0 = 30$.

f_n	29	29	28	20	15	12	10	9	8	9	10	11	11	11	11	11
$\hat{f}_n$	30	28	30	28	26	24	22	20	18	16	14	12	10	12	10	12
e	−1	+1	−2	−8	−11	−12	−12	−11	−10	−7	−4	−1	+1	−1	+1	−1
$\dot{e}$	−2	+2	−2	−2	−2	−2	−2	−2	−2	−2	−2	−2	+2	−2	+2	−2
$\dot{f}$	28	30	28	26	24	22	20	18	16	14	12	10	12	10	12	10



It can be seen from the above image that DM cannot follow rapid changes and it will produce noise even in flat areas. The encoding will thus lose information.

In lossless or information preserving coding the error must be encoded exactly. If the predictor is given as

$$\hat{f}_n = \alpha f_{n-1} = f_{n-1}$$

then the error is

$$e_n = f_n - \hat{f}_n = f_n - f_{n-1}.$$

The largest change will determine the code length. In the example the largest change is $20-28 = -8$. So we need 4 bits to encode the change if it is assumed that the change can also happen in another direction to $+7$. In DM only one bit is required.

T-61.5100 Digital image processing, Exercise 10, Dec 5, 2013

Image compression

1. Where should the two reconstruction levels be placed for the optimal 2-level quantiser, if the probability density function is

$$p(s) = \begin{cases} s+1, & \text{when } -1 < s < 0, \\ 1-s, & \text{when } 0 \le s < 1, \\ 0 & \text{otherwise} \end{cases} \qquad ?$$

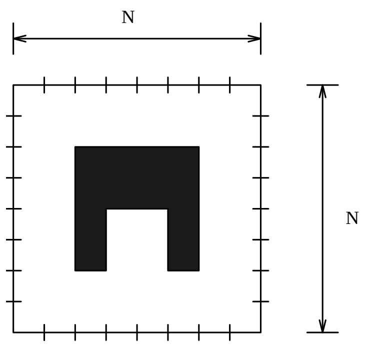
demo 2. Inspect the encoder in Figure 8.21(a) in the textbook and ignore the symbol encoder part, because it causes no error. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be a subimage to be encoded. Transformed image $\boldsymbol{y}$ will be formed from $\boldsymbol{x}$ so that first some linear orthonormed transformation is applied (for example, Fourier transformation) $\boldsymbol{y}' = \boldsymbol{A}\boldsymbol{x}$. After this, $\boldsymbol{y}$ will be replaced by m first components of $\boldsymbol{y}'$ and the result is given to the quantizer. Let the average quadratic truncation error $e_m^2 = E\left\{\|\boldsymbol{y} - \boldsymbol{y}'\|^2\right\}$, the quantization error $e_q^2 = E\left\{\|\boldsymbol{v} - \boldsymbol{y}\|^2\right\}$, and the total error $e_T^2 = E\left\{\|\boldsymbol{v} - \boldsymbol{y}'\|^2\right\}$ be defined as where $\boldsymbol{v}$ is the output of the quantizer. Let us assume that truncation and quantization errors are uncorrelated and thus additive.

   (a) Show that the truncation error equals to the energy corresponding to the excluded components of $\boldsymbol{y}'$ $e_m^2 = \sum_{i=m+1}^{n} E\{y_i'^2\}$

   (b) Show that from the fact that truncation and quantization errors are uncorrelated follows $e_T^2 = e_m^2 + e_q^2$

Image segmentation

3. Segment the image shown below using the split and merge procedure. Let $Q(R)$=TRUE if all pixels in region R have the same intensity. Show the quadtree corresponding to your segmentation.



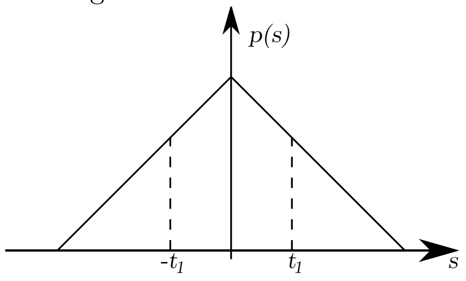
demo 4. Assume that the image consists of small, non-overlapping bubbles, which have a mean grayscale value of $m_1 = 150$ and a variance $\sigma_1^2 = 400$. The background has mean $m_2 = 25$ and variance $\sigma_2^2 = 625$. The bubbles take up about 20 % of the image. Show a method based on thresholding which separates the bubbles from the background.

T-61.5100 Digital image processing, Exercise 10, Dec 5, 2013

1.

The probability density function of the input values was given as:

$$p(s) = \begin{cases} s+1, & \text{when } -1 < s < 0 \\ 1-s, & \text{when } 0 \leq s < 1 \\ 0, & \text{otherwise} \end{cases}$$



The optimal quantiser is created by setting the reconstruction levels (codebook values) t_i so that the mean square quantisation error e_q^2 is minimised,

$$e_q^2 = E\left\{(s-t_i)^2\right\} = \sum_{i=1}^{L/2} \int_{s_{i-1}}^{s_i} (s-t_i)^2 p(s) ds, \quad s_i = \begin{cases} 0, & i=0 \\ \frac{t_i+t_{i+1}}{2}, & i=1,\dots \frac{L}{2}-1 \\ \infty, & i=\frac{L}{2} \end{cases} .$$

Here we have $L = 2$, i.e. simple Delta modulation (DM) coding. Thus, $s_0 = 0, s_1 = \infty$, and only a single level, t_1, needs to be solved. Now, the mean square error reduces to just a single integral:

$$e_q^2 = \int_0^\infty (s-t_1)^2 p(s) ds.$$

To find the minimum of that we need to solve where the derivative with regard to t_1 is zero (see page 602 in the course book):

$$\int_0^\infty (s-t_1) p(s)\, ds = 0.$$

Let's calculate the integral ...

$$\begin{aligned} &\int_0^\infty (s-t_1)p(s)\, ds = \\ &\int_0^1 (s-t_1)(1-s)\, ds + \int_1^\infty (s-t_1)\cdot 0 = \\ &\int_0^1 (-s^2+(1+t_1)s-t_1)\, ds = \\ &-\frac{1}{3}+\frac{1+t_1}{2}-t_1 \end{aligned}$$

Then put that back into the equation, and solve for t_1:

$$\begin{aligned} -\frac{1}{3}+\frac{1+t_1}{2}-t_1 &= 0 \\ t_1 = 2(-\frac{1}{3}+\frac{1}{2}) &= \frac{1}{3} \end{aligned}$$

2.

The encoder is the following:

$$\begin{tikzcd} {} \arrow[r] & \boxed{\text{Subimage}} \arrow[r] & \boxed{\text{Transform}} \arrow[r] & \boxed{\text{Truncate}} \arrow[r] & \boxed{\text{Quantizer}} \arrow[r] & {} \end{tikzcd}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{y}' = \begin{bmatrix} y'_1 \\ y'_2 \\ \vdots \\ y'_n \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \qquad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}$$

a) The average truncation (mapping) error:

$$\begin{aligned} e_m^2 &= E\left\{\|\mathbf{y}-\mathbf{y}'\|^2\right\} = E\left\{\sum_{i=1}^{n}(y_i - y'_i)^2\right\} = \sum_{i=1}^{n} E\left\{(y_i - y'_i)^2\right\} \\ &= \sum_{i=1}^{m} E\{\underbrace{(y_i - y'_i)^2}_{0}\} + \sum_{i=m+1}^{n} E\{\underbrace{(0 - y'_i)^2}_{(\star)}\} = \sum_{i=m+1}^{n} E\left\{y_i'^2\right\}. \end{aligned}$$

$(\star)$ The components that were left out were set as zeros.

b)

$$\begin{aligned} e_T^2 &= E\left\{\|\mathbf{v}-\mathbf{y}'\|^2\right\} = E\left\{\|(\mathbf{v}-\mathbf{y}) + (\mathbf{y}-\mathbf{y}')\|^2\right\} \\ &= E\left\{\|\mathbf{v}-\mathbf{y}\|^2 + 2(\mathbf{v}-\mathbf{y})^T(\mathbf{y}-\mathbf{y}') + \|\mathbf{y}-\mathbf{y}'\|^2\right\} \\ &= E\{\|\mathbf{v}-\mathbf{y}\|^2\} + 2E\{(\mathbf{v}-\mathbf{y})^T(\mathbf{y}-\mathbf{y}')\} + E\{\|\mathbf{y}-\mathbf{y}'\|^2\} \end{aligned}$$

$(\mathbf{v}-\mathbf{y})$ is the quantization error and $(\mathbf{y}-\mathbf{y}')$ the truncation (mapping) error. These are now uncorrelated, so $E\{(\mathbf{v}-\mathbf{y})^T(\mathbf{y}-\mathbf{y}')\} = 0$. Thus

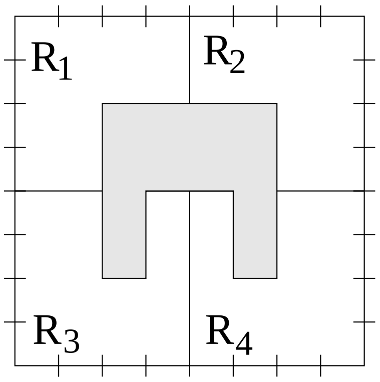
$$e_T^2 = E\{\|\mathbf{v}-\mathbf{y}\|^2\} + E\{\|\mathbf{y}-\mathbf{y}'\|^2\} = e_q^2 + e_m^2$$

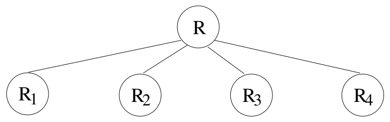
3.

The split and merge algorithm is given in the course book (section 10.4.2), repeated here a bit more clearly:

1. Initialise, $R = R_0$ = the whole image as one region. Also make R as the single root node of the quadtree.
2. *Split* any region R_i, for which $Q(R_i) = \text{FALSE}$, in to four disjoint quadrants $R_{i1}, R_{i2}, R_{i3}, R_{i4}$. Also add these four as child nodes to the R_i node in the quadtree.
3. *Merge* any adjacent regions R_j and R_k for which $Q(R_j \cup R_k) = \text{TRUE}$.
4. If no further splits or merges are possible, stop, otherwise go to step 2.

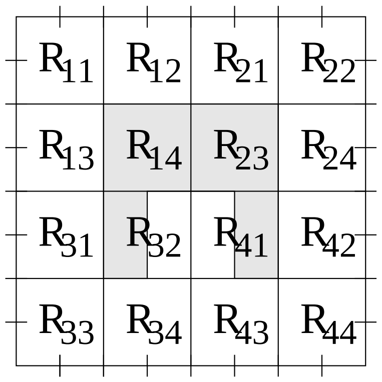
So, after the first split step (step 2.) the given image is divided into four regions:

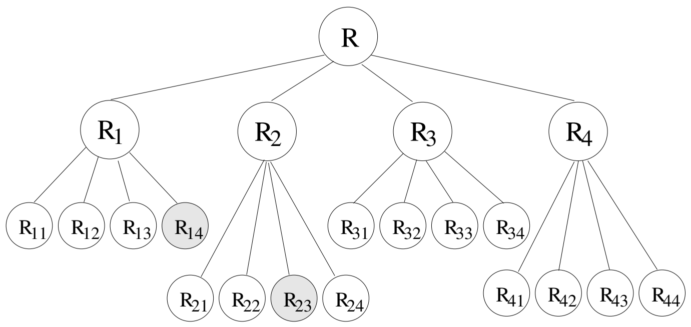




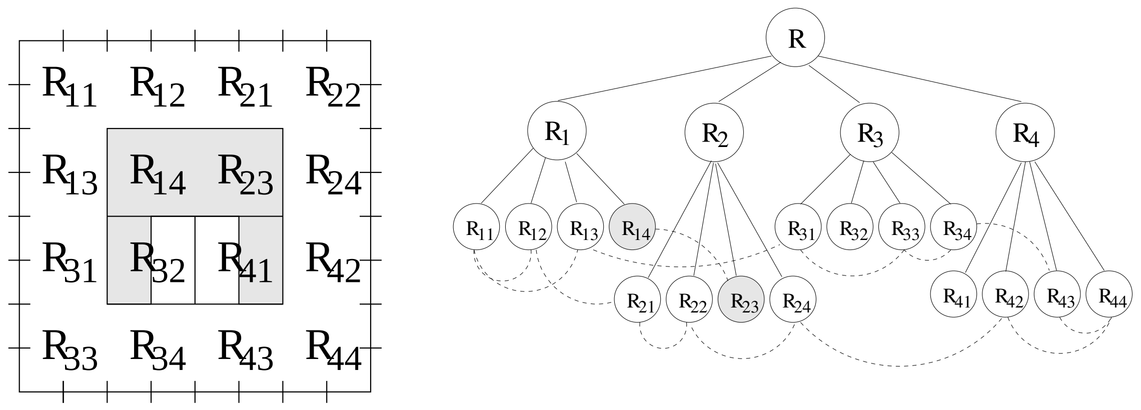
In step 3. there are no regions to merge, so we go back to step 2 ...

Now each region is further split into four regions:

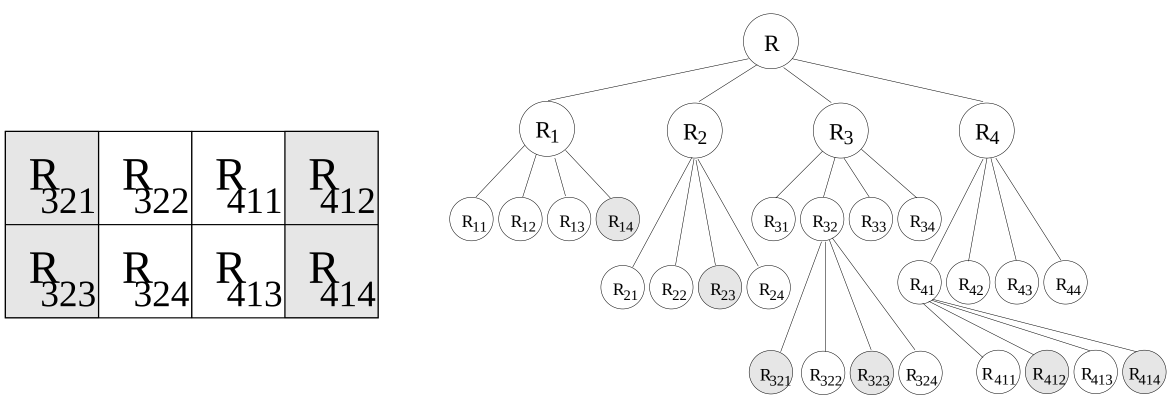




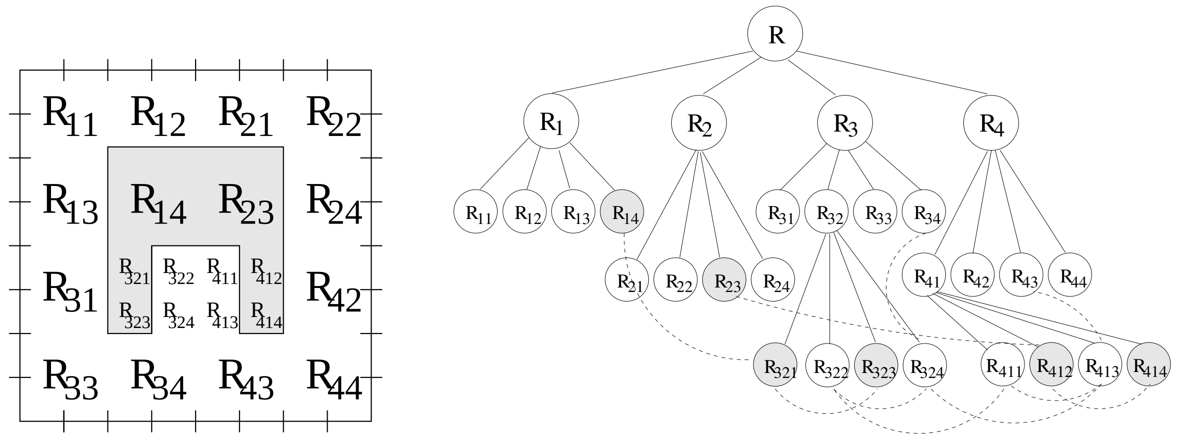
Now (for step 3.) the border regions (R_{11}, R_{12}, ...) are all homogenous with the same intensity and can therefore be merged into one region. The regions R_{14} and R_{23} are also homogenous with the same gray-scale value and are merged as well:



However, the regions R_{32} and R_{41} are not homogeneous and must be divided further, so we go back to step 2. again. Here is a zoomed-in image of the further division:



These regions are all homogeneous and can be merged (step 3.) into the two main regions from the previous step, and we are done:



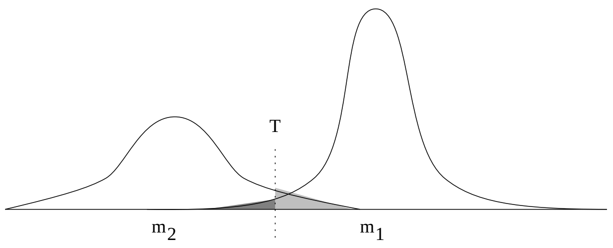
4.

The probability distribution for gray levels in class i is

$$p_i(z) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{(z-\mu_i)^2}{2\sigma_i^2}\right]$$

The probability $p(z)$ for the gray level z in the image is the joint probability of the gray levels of the bubbles and the background, $p(z) = P_1 p_1(z) + P_2 p_2(z)$. P_i is the *a priori* probability for the class i, i.e., the probability that a pixel belongs to class i without knowing its gray level value. So,

$$p(z) = P_1 \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{(z-\mu_1)^2}{2\sigma_1^2}\right] + P_2 \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left[-\frac{(z-\mu_2)^2}{2\sigma_2^2}\right]$$



If the threshold is T, the probability for the error is (see Figure)

$$E(T) = E_1(T) + E_2(T) = P_1 \int_{-\infty}^{T} p_1(z)dz + P_2 \int_{T}^{\infty} p_2(z)dz$$

The error function is minimized by setting its derivative to zero, $E'(T) = P_1 p_1(T) - P_2 p_2(T) = 0$, and we obtain

$$P_1 p_1(T) = P_2 p_2(T).$$

Because the exponentials may be tricky, we take the logarithm on both sides:

$$\ln P_1 p_1(T) - \ln P_2 p_2(T) = \ln \frac{P_1}{\sqrt{2\pi}\sigma_1} - \frac{(T-\mu_1)^2}{2\sigma_1^2} - \ln \frac{P_2}{\sqrt{2\pi}\sigma_2} + \frac{(T-\mu_2)^2}{2\sigma_2^2} = 0,$$

$$\Leftrightarrow -\sigma_2^2(T^2 - 2T\mu_1 + \mu_1^2) + \sigma_1^2(T^2 - 2T\mu_2 + \mu_2^2) + 2\sigma_1^2\sigma_2^2 \ln \frac{\sigma_2 P_1}{\sigma_1 P_2} = 0$$

$$\Leftrightarrow (\sigma_1^2 - \sigma_2^2)T^2 + 2(\mu_1\sigma_2^2 - \mu_2\sigma_1^2)T + \sigma_1^2\mu_2^2 - \sigma_2^2\mu_1^2 + 2\sigma_1^2\sigma_2^2 \ln \frac{\sigma_2 P_1}{\sigma_1 P_2} = 0.$$

When we now place the given values in the equation and observe also that $P_1/P_2 = 20\%/80\% = 0.25$, we obtain $-225T^2 - 167500T - 1.4506 \cdot 10^7 = 0$, and the final result is $\underline{T_1 \approx 100}$ and $\underline{T_2 \approx 644}$. The larger solution is due to the fact that the distribution of the bubbles is sharper (the variance is smaller) and on the right it will go (again!) under the distribution of the background.

T-61.5100 Digital image processing, Exercise 11, Dec 10, 2013

Colour image processing

This computer exercise assumes that you already know the basics of image processing in Matlab or Octave. E.g. opening an image (`imread`) and showing it on the screen (`imshow`). If you have forgotten something you can always check back to the previous computer exercises (Exercise no 2, 4 and 8 on the course Noppa page).

1. Colour space representations

Open the image `cbars.png` and store into a matrix as usual:

```
I=double(imread('cbars.png'))/255;
```

Display the image with the normal `imshow` method — you should see eight bars with different colours against a grey background. Also check the size of the matrix `I` (use the `size` command). What is unusual compared to the grey-scale images in previous exercises?

You can extract the `i`th component image by normal indexing: `I(:,:,i)`. Display the three component images as separate grey-scale images. Can you tell what colour representation Matlab/Octave uses by default?

Next, convert the image into the HSI colour space (H=hue, S=saturation, I=intensity).

Hint: try the `rgb2hsv` function — Matlab/Octave uses the name HSV for the HSI colour space (V=value). Display the three HSI component images, how do they compare to the RGB components?

Now do the same operations with a more realistic colour image, `motorcycle.png`.[1]

2. Component-wise filtering

Many grey-scale image processing methods can be trivially generalised to colour images by simply performing the processing on each colour component image separately.

Try blurring the images used in the previous task by using the same method as in Exercise 2 (`imfilter`) — but *separately* for each component image for red, green and blue. You can for example use `fspecial` to generate a 5×5 averaging mask. To do filtering for a given component `i` you can use indexing, for example:

```
J(:,:,i)=imfilter(I(:,:,i), f, 'same');
```

where `f` is the filter mask. Display the resulting colour image.

Next try the same for the images represented in the HSI colour space. Remember to convert back to RGB before displaying, by using the `hsv2rgb` function. What can you say about the result? Does it make sense to blur all components for HSI? How can we produce a better result?

[1]This image is from the mirflickr database, taken by Sonietta46 on Flickr, distributed under the Creative Commons Attribution License.

3. Colour edge detection

Again, we'll start with the image `cbars.png`, and then try the same processing with the more realistic image `motorcycle.png`.

First, calculate the gradient vectors of the RGB component images separately:

$$\nabla R = \left[\frac{\partial R}{\partial x}, \frac{\partial R}{\partial y}\right], \nabla G = \left[\frac{\partial G}{\partial x}, \frac{\partial G}{\partial y}\right], \text{ and } \nabla B = \left[\frac{\partial B}{\partial x}, \frac{\partial B}{\partial y}\right].$$

You can approximate the partial derivatives by convolving with the 3×3 Sobel masks (see e.g. `help fspecial`).

Then, show the gradient magnitudes for each component separately as grey-scale images, e.g. for the R component:

$$\|\nabla R\| = \sqrt{\left|\frac{\partial R}{\partial x}\right|^2 + \left|\frac{\partial R}{\partial y}\right|^2}.$$

Also, calculate and display the gradient sum by adding the gradient lengths for each component:

$$G_{sum} = \|\nabla R\| + \|\nabla G\| + \|\nabla B\|$$

Compare with the separate colour components and the original image.

Is there another way to calculate the gradient of the 3D colour vector?