Linear filtering.

1. Smoothing images with a Gaussian filter
The 2D Gaussian function

$$G(x,y) = G(x)G(y) = \exp\left(-\frac{x^2}{2\sigma^2}\right)\exp\left(-\frac{y^2}{2\sigma^2}\right) = \exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

$$= \exp\left(-\frac{\rho^2}{2\sigma^2}\right) = G(\rho,\varphi) \text{ (in polar coordinates)}$$

constitutes a very convenient low pass (smoothing) filter. Notice that we have constructed G(x,y) as the product of two 1−dimensional Gaussians G(x) and G(y). *Separability* is therefore trivial; this is a very desirable property, since it allows fast implementation of filtering by two consecutive 1− dimensional convolutions. The fact that the same parameter σ has been used for both the x and the y standard deviations entails the circular symmetry of G(x,y), which is evidenced by writing G in polar coordinates: notice how $G(\rho,\varphi)$ is actually a function only of the radius ρ and not of the angle φ. This results in the filter being *isotropic,* meaning that all directions in the image are treated in the same way (there is, of course, some underlying anisotropy implicit in the choice of square pixels, but this is a problem of a more fundamental order).

1a) Separability
Make a 2D smoothing filter with a standard deviation of $\sigma = 1.4$ by sampling the gaussian function f(x,y).

```
s=1.4;  %standard deviation
[x,y]=meshgrid(−round(3*s):round(3*s),−round(3*s):round(3*s));  %sample grid
g=exp(−(x.*x + y.*y)/(2*s*s));  %2D smoothing filter
g=g/sum(sum(g));  % sum of weights equals one
figure(1); subplot(2,1,1);mesh(x,y,g);  %show the filter
```

What is the size of the 2D filter g, ***size(g)****?*

Sample the 1D gaussian functions f(x) respectively f(y) to make the 1D filters gx and gy.

```
x1=−round(3*s):round(3*s);  %sample grid
gx= exp(−(x1.*x1 )/(2*s*s));  %smoothing filter in the x−direction
gx=gx/sum(gx);  % sum of weights equals one
gy=gx';  %smoothing filter in the y−direction (transpose of gx)
figure(1); subplot(2,1,2); stem(x1,gx);  %show the 1D smoothing filter
```

What is the size of the 1D filters gx and gy?

*Verify the separable property of the gaussian filter by computing g=gx∗ gy (****gg=conv2(gx,gy);****). Print out on the screen (or via the printer) the filter weights for g (2D filter) and for gg (gx∗ gy). Compare the weights!*
How do you use the separable property, i.e. how do you filter an image by gx and gy?

1b) Low−pass filtering (smoothing)
Load the image flowers.jpg. Filter the image by using the 2D filter g.

f=double(imread('flowers.jpg')); %load the image
f=f(1:256,1:256); %crop it to 256 x 256 pixels

y=conv2(f,g,'valid'); %filter the image
subplot(2,2,1); imshow(f/255); % original image
subplot(2,2,2); imshow(y/255); % smoothed image

How many multiplications per pixel are required when filtering the image using the 2D filter?

Now, use filter separability to filter the image by means of the two 1D filters gx and gy ((f* gx)* gy). Control (by computing the sum of the squared differences) that you obtain the same result that you obtained when you filtered the image with the 2D filter g.

yy=conv2(conv2(f,gx,'valid'),gy,'valid'); %filter the image
subplot(2,2,4);imshow(yy/255); %and display it

e=y-yy; %compute error
sum(sum(e.*e)) %print it on screen

How many multiplications per pixel are used when filtering the image using the 1D filters?
What is the effect of smoothing on the frequency spectrum? (suggestion: plot the DFT for one of the 1D smoothing filters.)
Gx=fft(gx,32); *%DFT of the filter*
stem(-16:15,fftshift(abs(Gx))); *%display the magnitude spectrum*

Make low pass filters gx and gy of larger size ($\sigma = 2.4$) and filter the image flowers.jpg (see previous Matlab code how to do it!).

*Explain how the smoothing effect varies when the size of the filter is changed by displaying the two smoothed images (**imshow(yy/255;)***
Plot the DFT of one of the larger 1D filters and compare with the smaller filter. Does their magnitude spectrum confirm the difference in smoothing effect?

1c) Noise reduction through smoothing
Add gaussian noise to the image flowers.jpg.

ff=imnoise(f/255,'gaussian',0,0.01).*255; %add noise

Filter the image by using the larger filters ($\sigma = 2.4$) gx and gy (see previous Matlab code how to do it) and display the noisy and the filtered image.

What is the result of the smoothing operation?

Extract the subpart of the original image and of the noisy image that also appear in the filtered image. Compare using the sum of the squared errors:
- noisy image vs. original image
- filtered image vs. original image

fs=f(8:249,8:249); %extract subpart in the original image f (*why 8:249?*)
ffs=ff(8:249,8:249); %and in the noisy image ff
e=ffs-fs; sum(sum(e.*e)) %error between noisy and original image
e=yy-fs; sum(sum(e.*e)) %error between filtered and original image

Which one is closer to the original? Why?
If you are curious: try reducing the variance of the noise to 0.001. Does filtering still help? Why?

2. Derivative filters and the extraction of edges
The partial derivatives of the Gaussian filter G(x,y) can be used (but for a sign change) as derivative (edge) filters in the x and y directions of the image.

$$-\frac{\partial}{\partial x}\{G(x,y)\} = -\frac{\partial}{\partial x}\{G(x)\}G(y) = -\frac{\partial}{\partial x}\left\{\exp\left(-\frac{x^2}{2\sigma^2}\right)\right\}\exp\left(-\frac{y^2}{2\sigma^2}\right)$$
$$= \frac{x}{\sigma^2}\exp\left(-\frac{x^2}{2\sigma^2}\right)\exp\left(-\frac{y^2}{2\sigma^2}\right) = -G'(x)G(y)$$

$$-\frac{\partial}{\partial y}\{G(x,y)\} = -\frac{\partial}{\partial y}\{G(y)\}G(x) = -\frac{\partial}{\partial y}\left\{\exp\left(-\frac{y^2}{2\sigma^2}\right)\right\}\exp\left(-\frac{x^2}{2\sigma^2}\right)$$
$$= \frac{y}{\sigma^2}\exp\left(-\frac{y^2}{2\sigma^2}\right)\exp\left(-\frac{x^2}{2\sigma^2}\right) = -G'(y)G(x)$$

Where G' is the derivative of the 1D Gaussian function G.

The gradient of the image is defined as $\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$. The magnitude of the gradient is large at the location of edges in the image.

2a) Derivative (edge) filtering
By sampling the 1D Gaussian function G (gx and gy in the code) and its derivative G' (dx and dy in the code) and using separability, we can construct derivative filters along the x and y directions of the image (dx*gy respectively dy*gx).

s=1.4; %standard deviation
x2=−round(3*s):round(3*s); %sample grid
dx= x2/(s*s).*exp(−(x2.*x2)/(2*s*s)); %derivative in the x−direction
dy=dx'; %derivative in the y−direction

x1=−round(3*s):round(3*s); %sample grid
gx= exp(−(x1.*x1)/(2*s*s)); %smoothing filter in the x−direction
gx=gx/sum(gx); % sum of weights equals one
gy=gx'; %smoothing filter in the y−direction (transpose of gx)

subplot(3,1,1); stem(x1,gx); %1D smoothing filter
subplot(3,1,2); stem(x2,dx); %1D derivative filter

Filter the image flowers.jpg in the x−direction ((f∗ dx)∗ gy) and in the y−direction ((f∗ dy)∗ gx). Show the original image and the filtered images.

fx=conv2(conv2(f,dx,'valid'),gy,'valid'); %derivative in x-direction
fy=conv2(conv2(f,dy,'valid'),gx,'valid'); %derivative in y-direction
subplot(2,2,1); imshow(f/255); %show the original image
subplot(2,2,2); imagesc(fx); colormap(gray); axis image; %show derivative image in the x-
subplot(2,2,4); imagesc(fy); colormap(gray); axis image; %and in the y-direction

Compute the magnitude of the gradient $\|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$ *and display the resulting edge filtered image.*

Edge=sqrt(fx.*fx + fy.*fy); %edge filtered image (the magnitude of the gradient)
subplot(2,2,3); imshow(Edge/255); %and display it

What is the action of derivative filters on the frequency spectrum of an image? (Suggestion: Plot the DFT of one of the derivative operators, either dx or dy).

Dx=fft(dx,32); %DFT of the dx filter
stem(-16:15,fftshift(abs(Dx))); %and display the magnitude spectrum

3a) Edge filtering in the presence of noise
Add gaussian noise to the image flowers.jpg.

ff=imnoise(f/255,'gaussian',0,0.01).*255; %add noise

Compute the edge image of the noisy image and display it (see previous Matlab code how to do it).

Is edge filtering sensitive to noise?
Does the fact that derivation is performed using derivatives of Gaussians help (think of the spectrum of the derivative filters dx, dy and how the derivative images fx and fy are computed)?