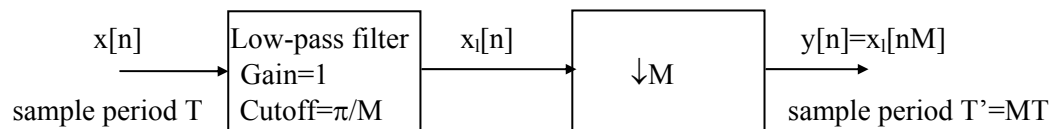


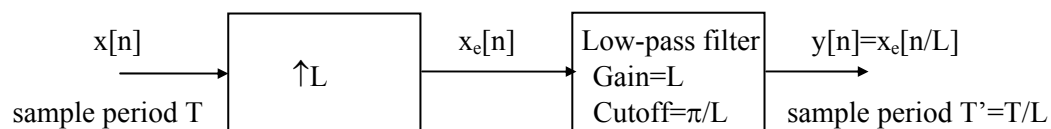
## Gaussian/Laplacian pyramids.

### 1. Downsampling and upsampling

When you compute the *Gaussian pyramid* (GP) you have to make images increasingly small (in size) (downsample, see Figure 1) and when you compute the *Laplacian pyramid* (LP) you have to make one GP image larger (upsampling, Fig. 2) before subtracting two consecutive images of the Gaussian pyramid.



**Figure 1.** Downsampling by an integer factor M, i.e. keep every M'th pixel horizontally/vertically.



**Figure 2.** Upsampling by an integer factor L, i.e. "expand" every pixel to L pixels.

When computing the Gaussian and the Laplacian pyramids M&L equal 2 giving a cutoff frequency of  $\pi/2$  for both the filters, Fig. 1 and Fig. 2.

The ideal low-pass filter can be approximated by a Gaussian function  $G(\omega) = \exp(-\omega^2/2\sigma^2)$ .

For the cutoff frequency of  $\pi/2$ , say you are satisfied with the approximation  $\sigma = \sigma_0$ .

You know the filters in the frequency domain: Gaussian,  $\sigma = \sigma_0$ . Use the scaling property of the FT to find the corresponding filters in the spatial domain: Gaussian;  $\sigma = 1/\sigma_0$ .

*Write a MATLAB script which generates the 1D gaussian filters in the spatial domain.*

That is generate one smoothing filter g1, and one interpolation filter g2. The smoothing filter is used before the subsampling of the image by 2, and the interpolation filter is used after the expansion of the image by 2. For this exercise use a  $\sigma = 1.0$  for the spatial filters g1 and g2.

```

s=1.0; %std in the spatial domain
x=-round(4*s):round(4*s); %sample grid
g1=exp(-(x.*x)/2/s/s); %smoothing filter
g1=g1/sum(g1); %gain=1
g2=2*g1; %gain=2
figure(1); subplot(2,1,1);stem(x,g1); %show filters
subplot(2,1,2);stem(x,g2);
  
```

*What is the result of smoothing and interpolation when the input function is a constant e.g.  $f=1$ ?*

*Hints: 1) compute it by Matlab:*

```

f=ones(1,30);y=conv(f,g1);subplot(3,1,1);stem(f);subplot(3,1,2);stem(y);
fz=zeros(1,60); fz(1:2:60)=f; fzi=conv(fz,g2);subplot(4,1,1);stem(fz);subplot(4,1,2);stem(fzi);
  
```

*2) write down the scalar products  $\langle f, g1 \rangle$  and  $\langle fz, g2 \rangle$  by hand when  $f=1$ .*

Using your experience in 1D, write two (own) functions for (2D) images in MATLAB: **shrink** and **expand**.

The function *shrink* should do a shrinking by an integer factor  $M$  ( $\downarrow M$ ). It will take as input parameters an image  $I$  and the factor  $M$ , the output parameter  $S$  will be the shrunk image ( **$S=\text{shrink}(I,M);$** ).

HINT: What is the size of  $f(1:30)$  compared to  $f(1:2:30)$  and what does the notation in the latter do?

The function *expand* will do an expansion by an integer factor  $L$  ( $\uparrow L$ ). It will take as input parameters an image  $I$  and the factor  $L$ , whereas the output parameter  $E$  will be the expanded image ( **$E=\text{expand}(I,L);$** ) where only every  $L$ 'th point comes from  $I$ , the rest being zero.

HINT: What is the size of the expanded image if the  $L=2$ , and the original size is  $256 \times 256$ ?...if  $L=3$ ?

Use the image *fmt* of size  $256 \times 256$ . You can generate the *fmt* image by the function *fmttest*.

You can use the command *trueSize* in order to ensure that the picture size in pixels is the same on the screen as in the memory. (Matlab normally resizes images in order to give a "nice" looking layout for displayed images).

```
fmt=fmttest(256,[0.1*pi 0.3*pi]); %generate the image fmt
```

```
figure(2); imshow(fmt); trueSize; %and display in trueSize
```

What can you say about the frequency content of the image *fmt*?

Without smoothing, use the function *shrink* to downsample the image *fmt* by a factor 2.

Display the result image.

```
fmt_s2=shrink(fmt,2); %shrink by 2
```

```
figure(3); imshow(fmt_s2); trueSize; %and display
```

Explain the effect of downsampling without smoothing.

First smooth the image *fmt* using filter *g1* and then use the function *shrink* to downsample the smoothed image by a factor 2.

Display the result image.

```
y=conv2(conv2(fmt,g1,'same'),g1,'same'); % first 2D smoothing
```

```
fmtsmooth_s2=shrink(y,2); %then shrink by 2
```

```
figure(4); imshow(fmtsmooth_s2); trueSize; %and display
```

Explain the effect of smoothing and downsampling.

## 2. Compute and display the Gaussian pyramid

Use the image *fmt*. Compute and display the Gaussian pyramid (*fmt*, *y128*, *y64*, and *y32*) up to a smallest image, say  $32 \times 32$ .

You can use the gaussian smoothing filter *g1*, and the MATLAB-function *shrink* for this:

```
fmt=fmttest(256,[0.1*pi 0.3*pi]); %generate the image fmt
```

```
%generate the Gaussian pyramid
```

```
y256_s=conv2(conv2(fmt,g1,'same'),g1,'same'); %first 2D smoothing
```

```
y128=shrink(y256_s,2); %then shrink by 2
```

```
y128_s=conv2(conv2(y128,g1,'same'),g1,'same'); %first 2D smoothing
```

```
y64=shrink(y128_s,2); %then shrink by 2
```

```
y64_s=conv2(conv2(y64,g1,'same'),g1,'same'); %first 2D smoothing
```

```
y32=shrink(y64_s,2); %then shrink by 2
```

```
%display the Gaussian pyramid in trueSize
```

```
figure(5); imshow(fmt); trueSize;
```

```
figure(6); imshow(y128); trueSize;
```

```
figure(7); imshow(y64); trueSize;
```

```
figure(8); imshow(y32); trueSize;
```

How do you interpret the images in the Gaussian pyramid?

### 3. Compute and display the Laplacian pyramid Compute the Laplacian pyramid (L256, L128, L64) from the Gaussian pyramid and display it.

Use the MATLAB-function *expand* and the gaussian interpolation filter *g2*.

```
e128_1=expand(y128,2); %first expand
e128_2=conv2(conv2(e128_1,g2,'same'),g2,'same'); %then interpolate
L256=fmt-e128_2; %take the difference
```

```
e64_1=expand(y64,2); %first expand
e64_2=conv2(conv2(e64_1,g2,'same'),g2,'same'); %then interpolate
L128=y128-e64_2; %take the difference
```

```
e32_1=expand(y32,2); %first expand
e32_2=conv2(conv2(e32_1,g2,'same'),g2,'same'); %then interpolate
L64=y64-e32_2; %take the difference
```

```
%display the Laplacian pyramid in truesize
figure(9); imagesc(L256); colormap(gray); truesize;
figure(10); imagesc(L128); colormap(gray); truesize;
figure(11); imagesc(L64); colormap(gray); truesize;
```

*How do you interpret the images in the Laplacian pyramid?*

*Why do we have the factor 4? If we did not have it, would we notice it by looking at with imagesc?*

*Try to see...*

*Final questions:*

*Why do we have to smooth the image before downsampling?*

*How do we upsample an image to get an expanded image and why do we "smooth"? (Hint: interpolation)*