Digital Image Processing Lecture 3

- DFT and the relation to continuous Fourier transform
- 2D convolution
- 2D filters: low-pass, high-pass, derivative (sobel)
- Circular convolution
- The magnitude of the gradient and edge enhancement

- Gonzales & Woods:
  - Chapter 3, 25 pages
  - Chapter 4, 65 pages

- Numbers according to Gonzales & Woods, Global Edition, 4th edition. (Numbers in other editions may vary).

Maria Magnusson, Computer Vision Lab., Dept. of Electrical Engineering, Linköping University

Another equation for the Fourier transform of the impulse train sampled signal

$$\tilde{G}(f) = \int_{-\infty}^{\infty} \tilde{g}(t)\, e^{-j2\pi ft} dt$$

$$= \int_{-\infty}^{\infty} g(t) \cdot \sum_{n=-\infty}^{\infty} \delta(t-n\Delta)\, e^{-j2\pi ft} dt$$

$$= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} g(t)\delta(t-n\Delta)\, e^{-j2\pi ft} dt \qquad \approx\text{(4-40)}$$

$$= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} g(t)\delta(t-n\Delta)\, e^{-j2\pi ft} dt$$

$$= \sum_{n=-\infty}^{\infty} g(n\Delta)\, e^{-j2\pi fn\Delta} = \sum_{n=-\infty}^{\infty} g_n\, e^{-j2\pi fn\Delta}$$

DFT (Discrete Fourier transform), symmetric

DFT

$$G_k = \sum_{n=-N/2}^{N/2-1} g_n e^{-j2\pi nk/N}, \quad -\frac{N}{2} \le k \le \frac{N}{2} - 1$$

Inverse DFT

$$g_n = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} G_k e^{j2\pi nk/N}, \quad -\frac{N}{2} \le n \le \frac{N}{2} - 1$$

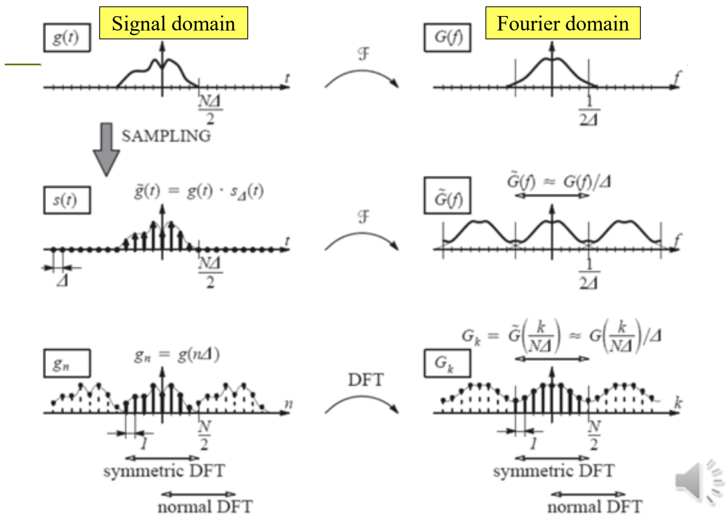
Scaling between continuous and discrete frequenses:

$$f = \frac{k}{N\Delta}$$

Comparison with previous slide gives: $G_k = \tilde{G}\left(\frac{k}{N\Delta}\right)$

if $g_n = 0$ outside n:s interval. G_k is a scaled variant of $\tilde{G}(\ )$

Continuous Fourier transform & DFT



DFT (Discrete Fourier transform), symmetric and normal

Matlab:
fftshift(fft(ifftshift( )))
ifftshift(ifft(fftshift( )))

DFT, symmetric

$$G_k = \sum_{n=-N/2}^{N/2-1} g_n e^{-j2\pi nk/N}, \quad -\frac{N}{2} \le k \le \frac{N}{2} - 1$$

$$g_n = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} G_k e^{j2\pi nk/N}, \quad -\frac{N}{2} \le n \le \frac{N}{2} - 1$$

Matlab:
fft( )
ifft( )

DFT, normal

$$G_k = \sum_{n=0}^{N-1} g_n e^{-j2\pi nk/N}, \quad 0 \le k \le N-1$$ ≈(4-42)

$$g_n = \frac{1}{N} \sum_{k=0}^{N-1} G_k e^{j2\pi nk/N}, \quad 0 \le n \le N-1$$ ≈(4-43)

About the relations between continuous Fourier transform and DFT…

- A time-limited signal can not be band-limited.
- A band-limited signal can not be time-limited.
- DFT demands a time-limited signal.
- The sampling theorem demands a band-limited signal.
- Therefore, the continuous Fourier transform can only be calculated *approximately* with sampling and DFT.
- DFT can be calculated fast with FFT (Fast Fourier Transform).
- For N points, the complexity of DFT is $O(N^2)$.
- For N points, the complexity of FFT is O(N logN).
- There are more about FFT (outside the course) in chapter 4 if you are interested.

2D DFT, normal

Can you write down the 2D Symmetric DFT?

$$F(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) e^{-j2\pi(xu/N + yv/M)}$$ ≈(4-67)

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F(u,v) e^{j2\pi(xu/N + yv/M)}$$ ≈(4-68)

$$F(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) e^{-j2\pi(xu/N + yv/M)}$$

can be separated

$$= \sum_{x=0}^{N-1} \left(\sum_{y=0}^{M-1} f(x,y) e^{-j2\pi yv/M} \right) e^{-j2\pi xu/N}$$

1D and 2D continuous and discrete convolution

Continuous convolution

$$w * f(x) = \int_{-\infty}^{\infty} w(x-\alpha) \cdot f(\alpha)\, d\alpha$$ ≈(4.24)

$$w * f(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w(x-\alpha, y-\beta) \cdot f(\alpha,\beta)\, d\alpha d\beta$$

Discrete convolution

$$w * f(x) = \sum_{\alpha=-\infty}^{\infty} w(x-\alpha) \cdot f(\alpha)$$

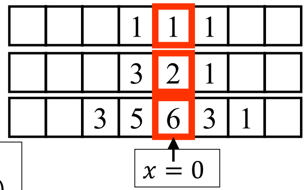
$$w * f(x,y) = \sum_{\alpha=-\infty}^{\infty} \sum_{\beta=-\infty}^{\infty} w(x-\alpha, y-\beta) \cdot f(\alpha,\beta)$$

1D discrete convolution

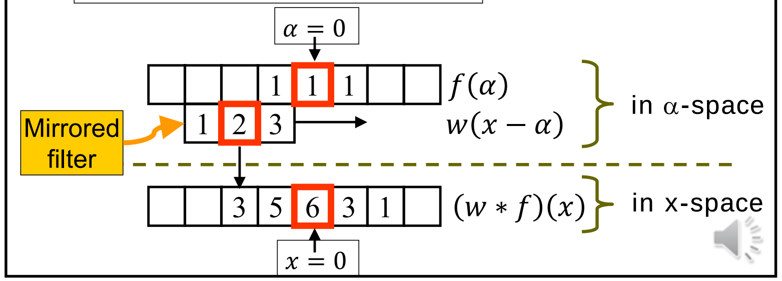
$f(x) = 0,0,0,1,1,1,0,0$

$w(x) = 0,0,0,3,2,1,0,0$

$(w * f)(x) = 0,0,3,5,6,3,1,0$



$$(w * f)(x) = \sum_{\alpha=-\infty}^{\infty} w(x-\alpha) \cdot f(\alpha)$$



The mechanics of linear spatial filtering using a 3x3 filter mask

$g(x,y) =$
$w(-1,-1)f(x-1,y-1)$
$+$
$w(-1,0)f(x-1,y)+\ldots+$
$w(0,0)f(x,y)+\ldots+$
$w(1,1)f(x+1,y+1)$

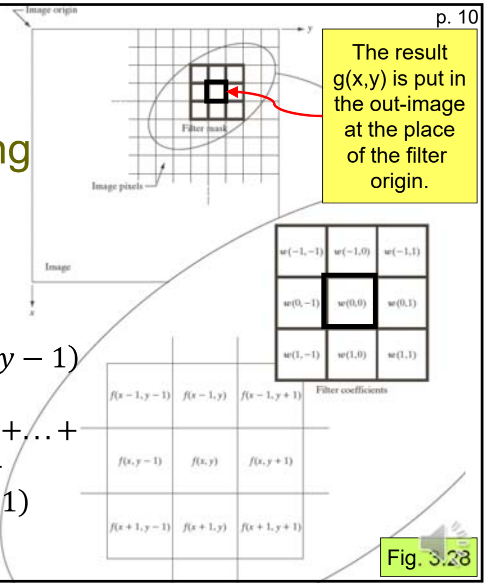


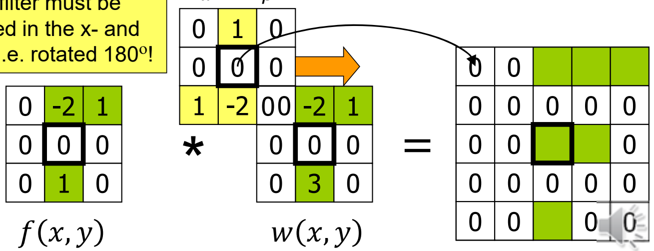
Fig. 3.28

2D discrete convolution

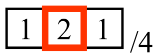
$$h(x,y) = w * f(x,y) = \sum_{\alpha=-\infty}^{\infty} \sum_{\beta=-\infty}^{\infty} w(x-\alpha, y-\beta) \cdot f(\alpha,\beta)$$

$$h(x,y) = f * w(x,y) = \sum_{\alpha=-\infty}^{\infty} \sum_{\beta=-\infty}^{\infty} f(x-\alpha, y-\beta) \cdot w(\alpha,\beta)$$

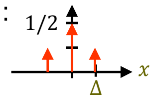
The filter must be mirrored in the x- and y-axis, i.e. rotated 180°!



Which Fourier transform has this filter?



- Put dirac impulses at every value in the filter. Set the sample distance to Δ. This gives:



$$f(x) = [1 \cdot \delta(x+\Delta) + 2 \cdot \delta(x) + 1 \cdot \delta(x-\Delta)]/4$$

- Compute the continuous Fourier transform, use the formula collection. This gives:

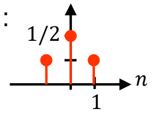
$$F(u) = \left[1 \cdot 1 \cdot e^{+j2\pi\Delta u} + 2 \cdot 1 + 1 \cdot 1 \cdot e^{-j2\pi\Delta u}\right]/4 =$$

$$= [2\cos(2\pi\Delta u) + 2]/4 = \cos^2(\pi\Delta u)$$

Alternative. Compute the DFT of the filter: [1 2 1]/4

p. 13

- Here is the filter drawn as $f_D(n)$:



- Insert $f_D(n)$ into the DFT formula:

$$F_D(k) = \sum_{n=-N/2}^{N/2-1} f_D(n)\, e^{-j2\pi nk/N} = [\cdots + 0 + 1 \cdot e^{-j2\pi(-1)k/N} + \ldots$$

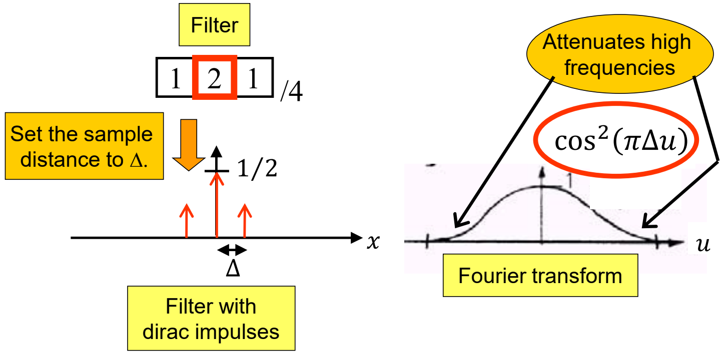
$$\ldots + 2 \cdot e^{-j2\pi(0)k/N} + 1 \cdot e^{-j2\pi(1)k/N} + 0 + \cdots]/4 =$$

$$= [2\cos(2\pi k/N) + 2]/4 = \cos^2(\pi k/N)$$

Scaling between continuous and discrete frequenses: $u = k/N\Delta$

A 1D discrete weighted averaging filter is also a low-pass filter

p. 14



Which 2D Fourier transform has this filter? [1 2 1]/4

p. 15

- Put dirac impulses $\delta(x,y)=\delta(x)\delta(y)$ at every value in the filter. Set the sample distance to Δ. This gives

$$f(x,y) = [1 \cdot \delta(x+\Delta) + 2 \cdot \delta(x) + 1 \cdot \delta(x-\Delta)] \cdot \delta(y)/4$$

- Compute the continuous Fourier transform

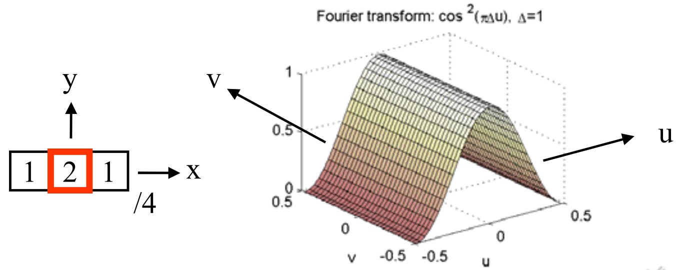
$$F(u,v) = [1 \cdot e^{+j2\pi\Delta u} + 2 + 1 \cdot e^{-j2\pi\Delta u}] \cdot 1(v)/4 =$$

$$= [2\cos(2\pi\Delta u) + 2]/4 = \cos^2(\pi\Delta u)$$

Low-pass filter in the x- (u-) direction

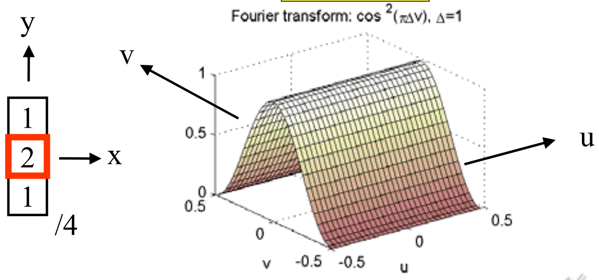
p. 16

$\cos^2(\pi\Delta u)$
$\Delta = 1$ here



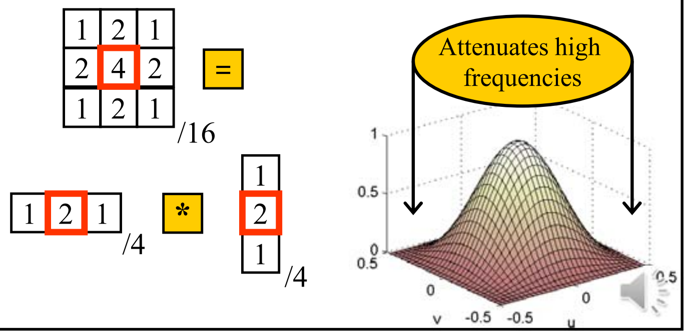
Low-pass filter in the y- (v-) direction

$\cos^2(\pi\Delta v)$
$\Delta = 1$ here



Low-pass filter in the x- (u-) direction and the y- (v-) direction

$\cos^2(\pi\Delta u) \cdot \cos^2(\pi\Delta v)$
$\Delta = 1$ here



A 3x3 averaging filter and a 3x3 weighted averaging filter

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

Fig. 3.31

A (weighted) averaging filter must be divided by the sum of the values of its coefficients.

The averaging filter have relatives of size 5x5, 7x7, 9x9, … They do all have a sinc×sinc-like Fourier transform.

Smoothing with averaging filters

Noise is attenuated. Details and edges are blurred.

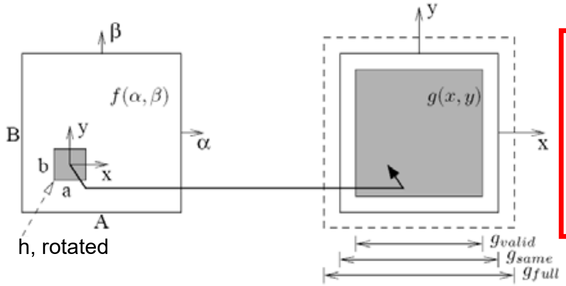
Why is it a dark frame around some images?

The smoothing is not symmetric in all directions and the sinc×sinc-like Fourier transform is not the best choice.

Similar to Fig. 3.33

Image size after 2D linear discrete convolution

- Valid: Values outside the in-image are regarded as undefined => The out-image becomes smaller than the in-image.
- Full: Values outside the in-image are regarded as zero => The out-image becomes bigger than the in-image. Or equally sized if the extra values are thrown away. (Same)



To avoid border effects, the border pixels in the image can be extrapolated before convolution.

More low-pass filter in the x- (u-) direction and the y- (v-) direction

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

/256

=

1	2	1
2	4	2
1	2	1

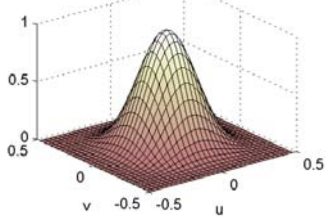
/16

*

1	2	1
2	4	2
1	2	1

/16

$\cos^4(\pi\Delta u) \cdot \cos^4(\pi\Delta v)$
$\Delta = 1$ here

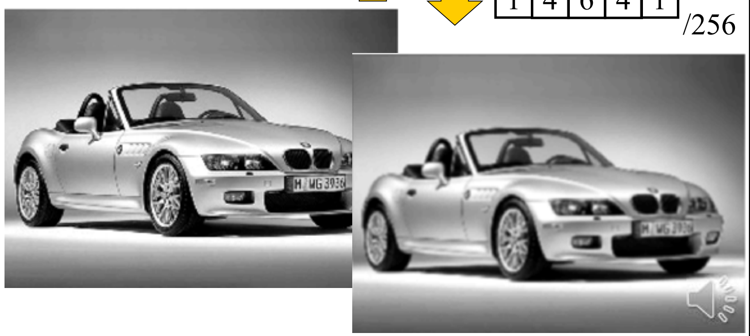


Low-pass filtering in the spatial domain

*

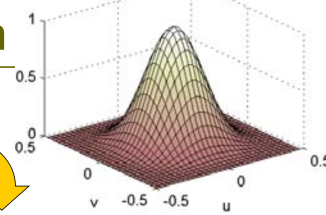
1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

/256

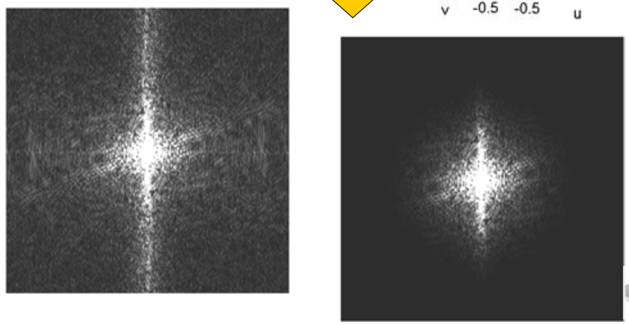


Low-pass filtering in the Fourier domain

Fourier transform



•



Computational complexity for discrete convolution

f11	f12	f13
f21	f22	f23
f31	f32	f33

$\circledast$

1	2	1
2	4	2
1	2	1

=

g11	g12	g13
g21	g22	g23
g31	g32	g33

$$g22 = f11 + 2 \cdot f12 + f13 + \ldots$$
$$2 \cdot f21 + 4 \cdot f22 + 2 \cdot f23 + \ldots$$
$$f31 + 2 \cdot f32 + f33$$

5 multiplications and 8 additions per pixel!

Computational complexity with and without separation

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

/256 =

1	2	4	2	1

*

1
2
4
2
1

/256

21 multiplications, 24 additions and 1 division per pixel

6 multiplications, 8 additions and 1 division per pixel

Gaussian low-pass filters (GLPF) designed in the Fourier Domain

Note: The u- and v-axis should rather pass the center of the image.

The filters on the previous slides belongs to a family with separable weighted averaging filters based on the binomial coefficients. They approximates Gaussian functions. However, Gaussian low-pass filters can also be designed directly in the Fourier domain.

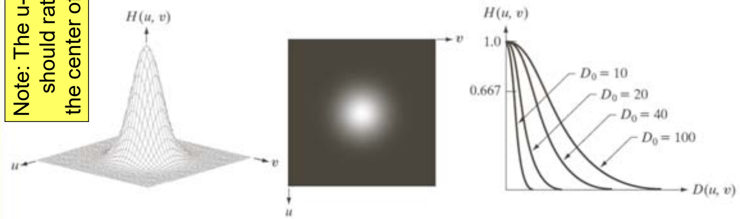


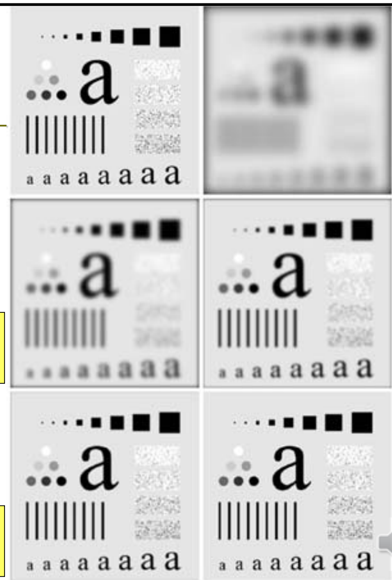
Fig. 4.43 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

$$H(u,v) = e^{-D^2(u,v)/2D_0^2} \quad (4\text{-}116)$$

Smoothing with GLPF filters

Noise are attenuated. Details and edges are blurred.

The smoothing is symmetric all directions, i.e. rotational symmetric.



Similar to Fig. 3.36-3.37

Ideal low-pass filters (ILPF) designed in the Fourier Domain

Note: The u- and v-axis should rather pass the center of the image.

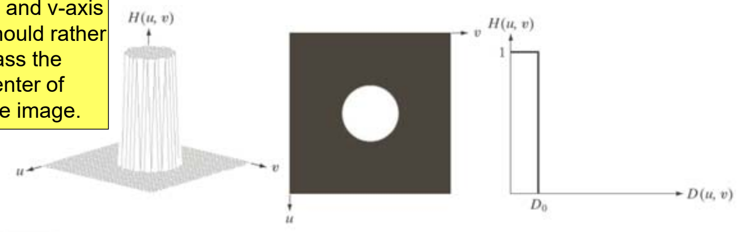


Fig. 4.39 (a) Perspective plot of an ideal lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \le D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases} \quad (4\text{-}111)$$

$$D(u,v) = \sqrt{(u-P/2)^2 + (v-Q/2)^2} \quad (4\text{-}112)$$

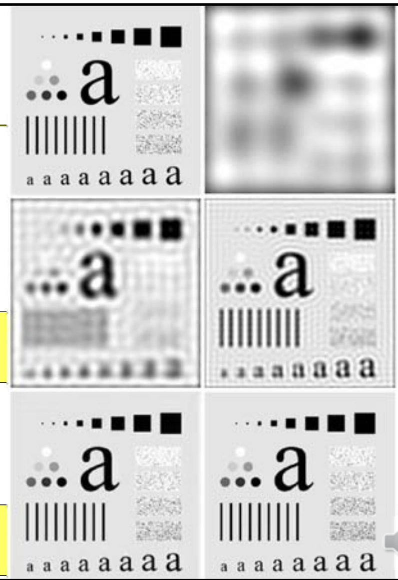
Smoothing with ILPF filters

Noise are attenuated. Details and edges are blurred.

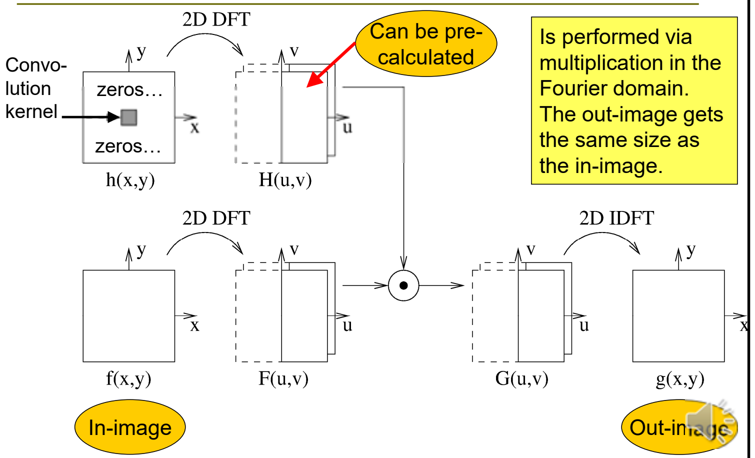
Note the Gibbs ringing artefacts!

The smoothing is symmetric all directions, i.e. rotational symmetric.

Similar to Fig. 4.41



Multiplication in the discrete Fourier domain corresponds to circular convolution



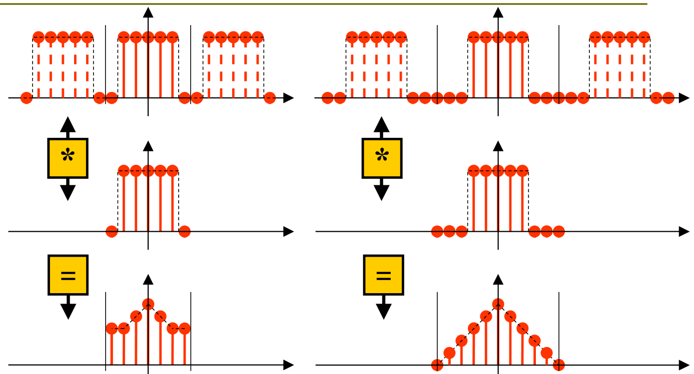
1D circular convolution

$$f *_N h(m) = \sum_{n=0}^{N-1} f(n) \cdot h\big((m-n)_N\big)$$

where $*_N$ denotes circular convolution and $(\ )_N$ denotes modulo N operation, i.e. $h(\ )$ can be viewed as periodical repeated or circular.

Theorem: $DFT\big(f *_N h(m)\big) = F(k) \cdot H(k)$

1D circular convolution

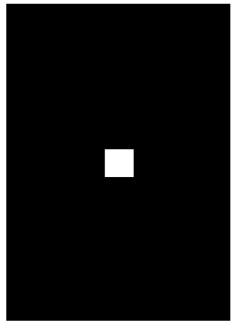


Zero-padding must be performed before circular convolution to get the same result as normal linear convolution.

Example) 2D linear and circular convolution

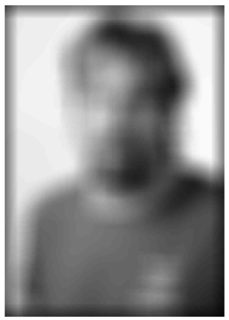
Peter Forsberg, former hockey player

In-image

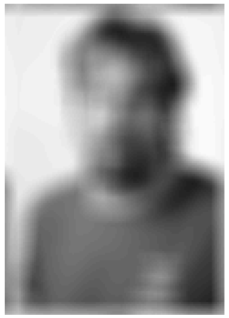


Averaging filter
Size: 15x15

Example) 2D linear and circular convolution



Out-image after linear convolution



Out-image after circular convolution

Computing the derivative = convolution with a derivative operator

$$\frac{\partial}{\partial x}$$

Fourier transform

$$j2\pi u$$

Derivative filter:

1	0	-1

$/2\Delta$

Fourier transform

$$j\sin(2\pi\Delta u)/\Delta$$

$j\sin(2\pi\Delta u)/\Delta \rightarrow j2\pi\Delta u/\Delta =$
$= j2\pi u$, for small u

A filter that have a Fourier transform looking like a straight line through the origin can be used as a derivative filter.

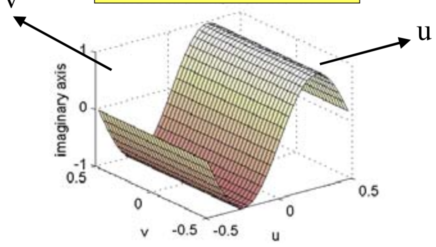
Derivative filter plus low-pass in the x- (u-) direction

$j \sin(2\pi\Delta u)/\Delta$,
$\Delta = 1$ here

1	**0**	-1

$/2\Delta$

(y ↑, x →)



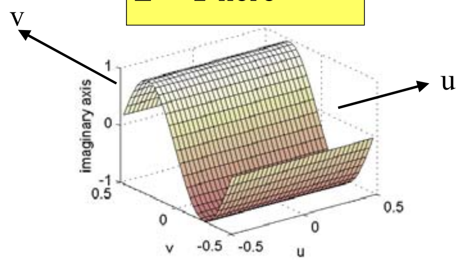
Derivative filter plus low-pass in the y- (v-) direction

$j \sin(2\pi\Delta v)/\Delta$,
$\Delta = 1$ here

-1
0
1

$/2\Delta$

(y ↑, x →)



Derivative filter in the x- (u-) direction plus low-pass filter in both directions

Sobel-x

1	0	-1
2	**0**	-2
1	0	-1

$/8\Delta$ =

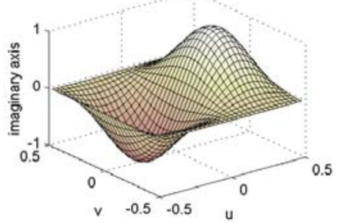
1	**0**	-1

$/2\Delta$ *

1
2
1

$/4$

$j \sin(2\pi\Delta u) \cdot \cos^2(\pi\Delta v)/\Delta$,
$\Delta = 1$ here



≈Fig. 4.38

Derivative filter in the y- (v-) direction plus low-pass filter in both directions

Sobel-y

-1	-2	-1
0	**0**	0
1	2	1

$/8\Delta$ =

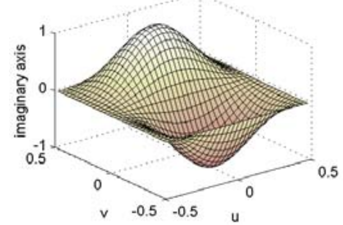
1	**2**	1

$/4$ *

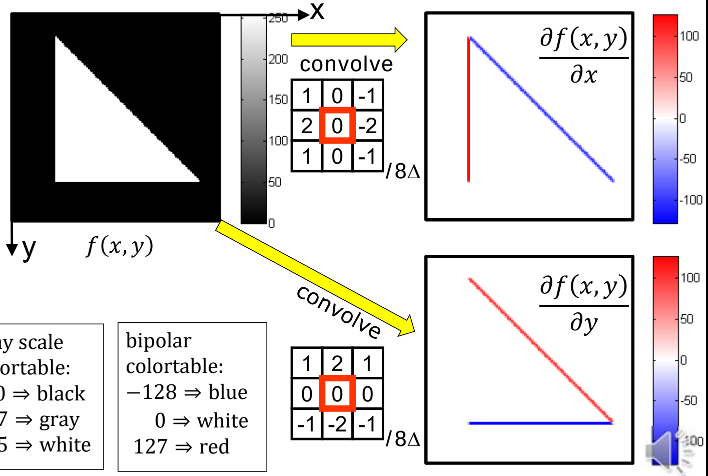
-1
0
1

$/2\Delta$

$j \sin(2\pi\Delta v) \cdot \cos^2(\pi\Delta u)/\Delta$,
$\Delta = 1$ here



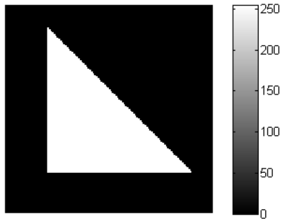
Derivative filtering

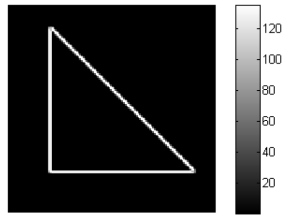


gray scale colortable:
0 ⇒ black
127 ⇒ gray
255 ⇒ white

bipolar colortable:
−128 ⇒ blue
0 ⇒ white
127 ⇒ red

The magnitude of the gradient enhances edges in the image

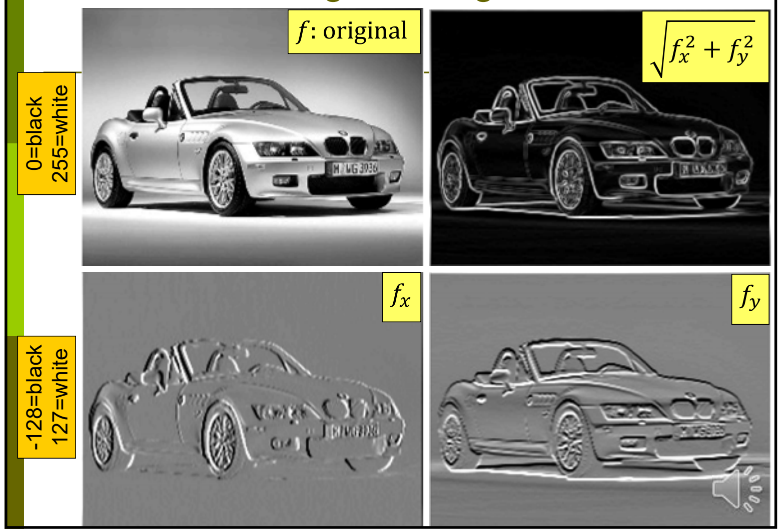




$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$$

Derivative filtering and edge enhancement



Linear discrete convolution when the center is between the pixels

$\frac{\partial}{\partial x}$ * lowpass filter = $\frac{\partial}{\partial x}$

[1 | -1] /Δ * [1 | 1] /2 = [1 0 -1] /2Δ

$\frac{\partial}{\partial y}$ * lowpass filter = $\frac{\partial}{\partial y}$

[-1 / 1] /Δ * [1 / 1] /2 = [-1 / 0 / 1] /2Δ

In a similar way as on previous slide, we can construct a Laplace filter

Lab 1 exercise

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial x} * \frac{\partial}{\partial x} + \frac{\partial}{\partial y} * \frac{\partial}{\partial y} = \ \ldots\ = \text{Mask a}$$

Mask a

0	1	0
1	−4	1
0	1	0

Mask b

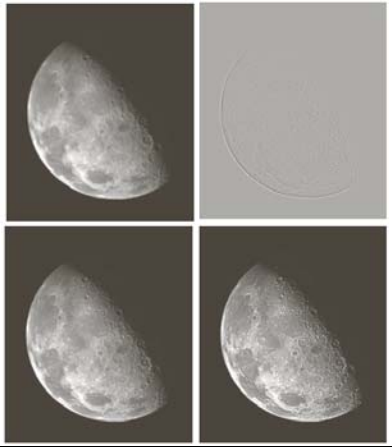
1	1	1
1	−8	1
1	1	1

Fig. 3.45

Laplace can help to give a shaper image $g(x,y) = f(x,y) + c[\nabla^2 f(x,y)]$ (3-54)

Note that c = -1 !

$f(x,y)$ $\nabla^2 f(x,y)$



$g(x,y)$ Using Mask a

$g(x,y)$ Using Mask b

Fig. 3.46

A normal approximate Laplace filter mask = a high-pass filter • -1

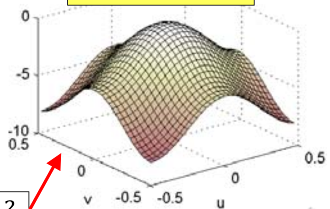
$$\Im[(1,-2,1)/\Delta^2]: \ \ [1 \cdot e^{+j2\pi\Delta u} - 2 + 1 \cdot e^{-j2\pi\Delta u}]/\Delta^2 =$$

$$= [2\cos(2\pi\Delta u) - 2]/\Delta^2 = -4 \cdot \sin^2(\pi\Delta u)/\Delta^2$$

0	1	0
1	-4	1
0	1	0

$/\Delta^2$ = [1 -2 1] $/\Delta^2$ + [1; -2; 1] $/\Delta^2$

Fourier transform



$$-4(\sin^2(\pi\Delta u) + \sin^2(\pi\Delta v))/\Delta^2$$

Low- and High-pass filtering

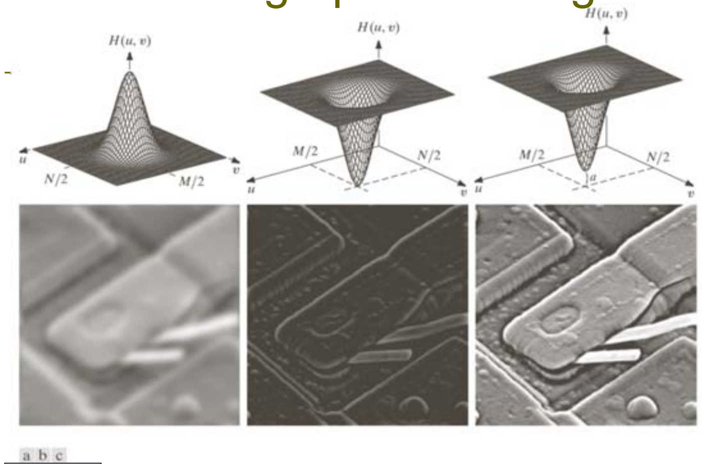


Fig. 4.30 Top row: frequency domain filters. Bottom row: corresponding filtered images obtained using Eq. (4.7-1). We used $a = 0.85$ in (c) to obtain (f) (the height of the filter itself is 1). Compare (f) with Fig. 4.29(a).