UiO : Department of Informatics
University of Oslo

INF 5860 Machine learning for image classification

Lecture 8

Background in image convolution, filtering, and filter banks for feature extraction

Anne Solberg

March 17, 2017

Today

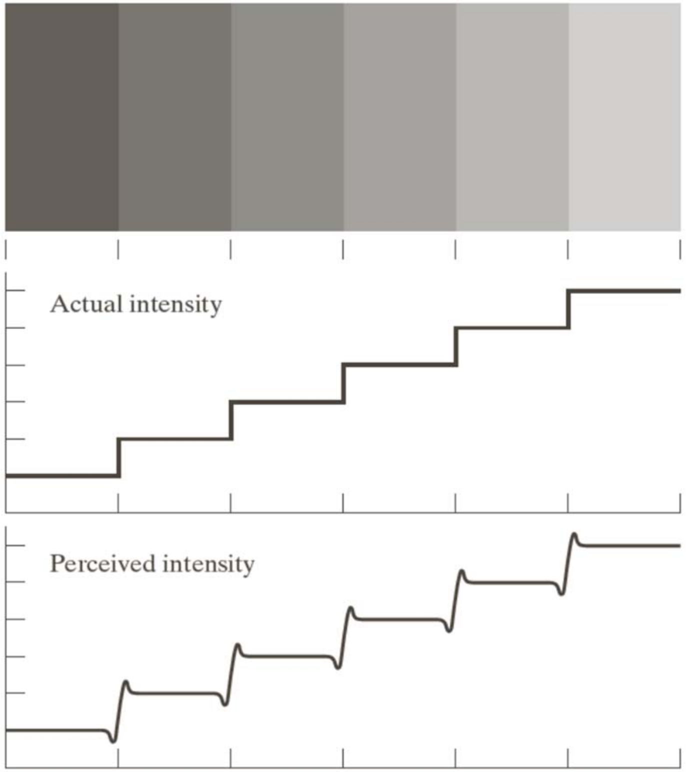
- Image filtering
- 2D convolution
- Edge detection filters
- Filtering in the frequency domain
  – Lowpass, bandpass, highpass
- Gabor-filters for feature extraction/segmentation

- PURPOSE: give a short background to those without a background in image analysis
- Convolutional nets use convolution as the basic operation.

Properties of the human visual system

- We can see light intensities over a broad range
  - The largest is about **10^{10}** times higher than the lowest we can sense.
- We can only see a certain number of levels simultaneously,
  - About 50 different gray levels, but many more colors.
- When we focus on a different area, the eye adapts and we see local intensity differences.

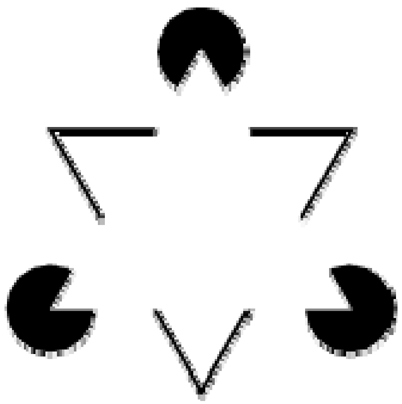
Neural processes in the retina



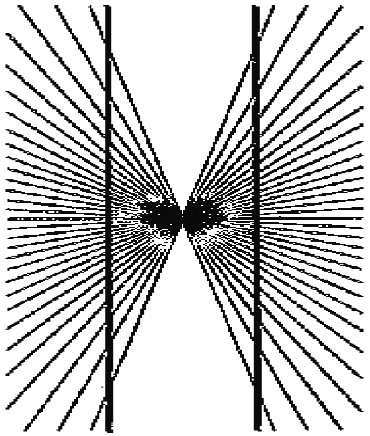
- Amplifies edges.
- Stimulating one part suspends other parts.
  - See one orientation at a time
- Increased contrast in edges between uniform regions
  - Called Mach band

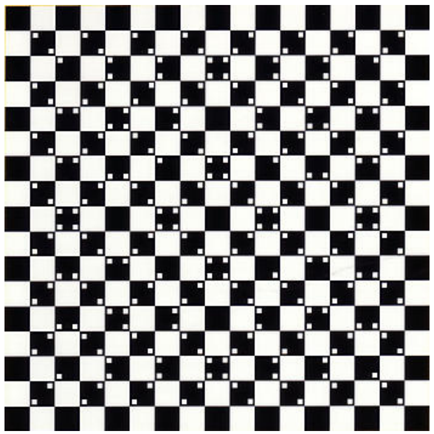
Optical illusions

- Illusional contours



Straight or curved lines





- Multistable images

Simultaneous contrast

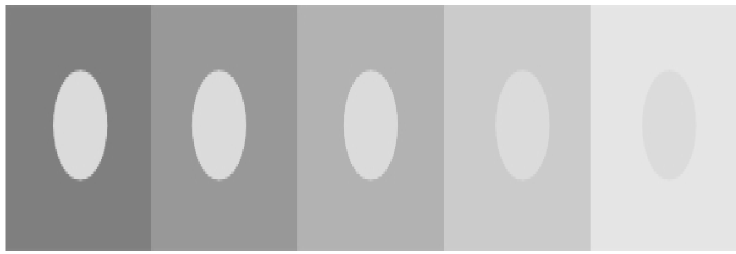


Image filtering

- One of the most used operations on images
- A filter kernel is applied either in the image domain or 2D Fourier domain.
- Applications:
  - Image enhancement
  - Image restoration
  - Image analysis – preprocessing
- Used to:
  - Reduce noise
  - Enhance sharpness
  - Detect edges and other structures
  - Detect objects
  - Extract texture information

Spatial filtering

- A filter is given as a matrix:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

- The size of the matrix and the coefficients decides the result.

Spatial filtering in general

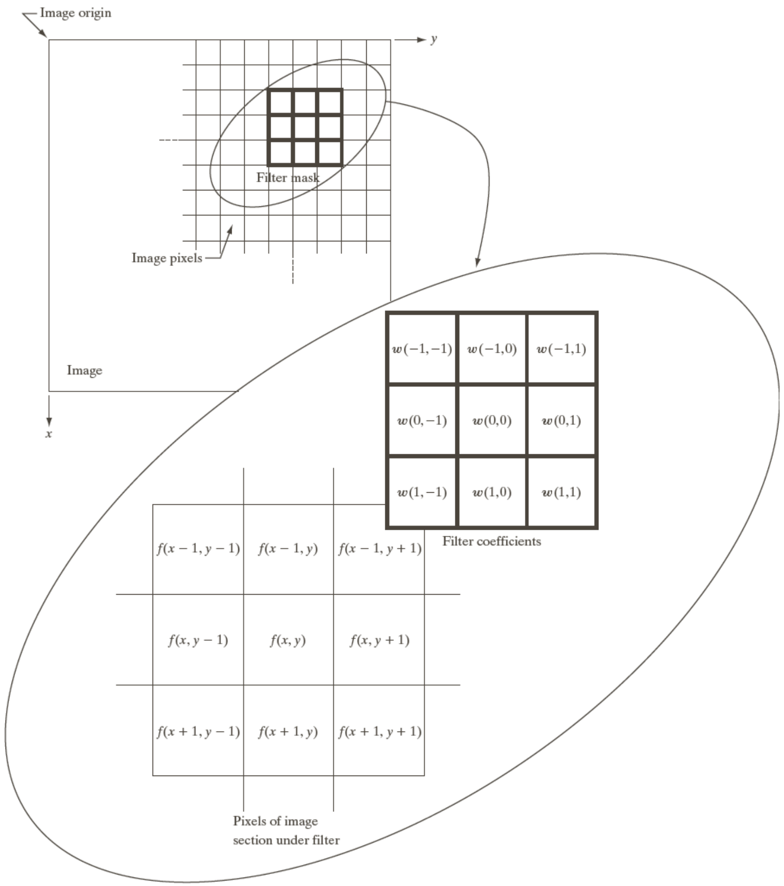


FIGURE 3.28 The mechanics of linear spatial filtering using a 3 × 3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

2-D convolution

- Output image: g. Input image f.

$$g(x,y) = \sum_{j=-w_1}^{w_1} \sum_{k=-w_2}^{w_2} h(j,k) f(x-j, y-k)$$

$$= \sum_{j=x-w_1}^{x+w_1} \sum_{k=y-w_2}^{y+w_2} h(x-j, y-k) f(j,k)$$

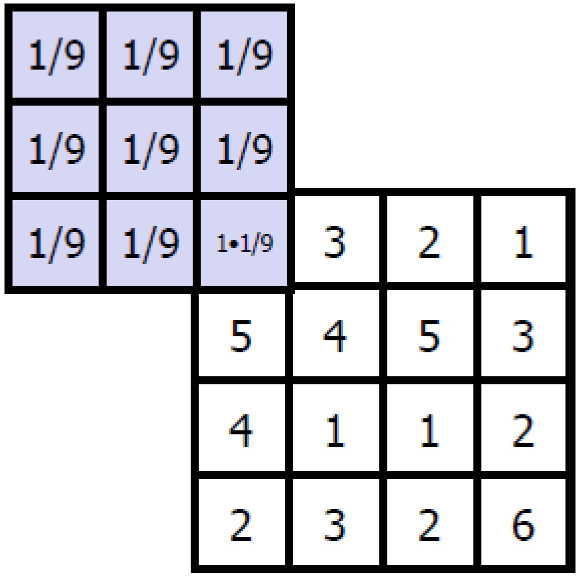
- h is a $m \times n$ filter of size $m=2w_1+1$, $n=2w_2+1$
- *m* and *n* usually odd.
- Output image: weighted sum of input image pixels surrounding pixel (x,y). Weights: *h(j,k).*
- This operation is done for every pixel in the image

Step 1: convolution: rotate the image 180 degrees

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

In this case, nothing changes

Step 2: Iterate over all locations where the filter overlaps the image

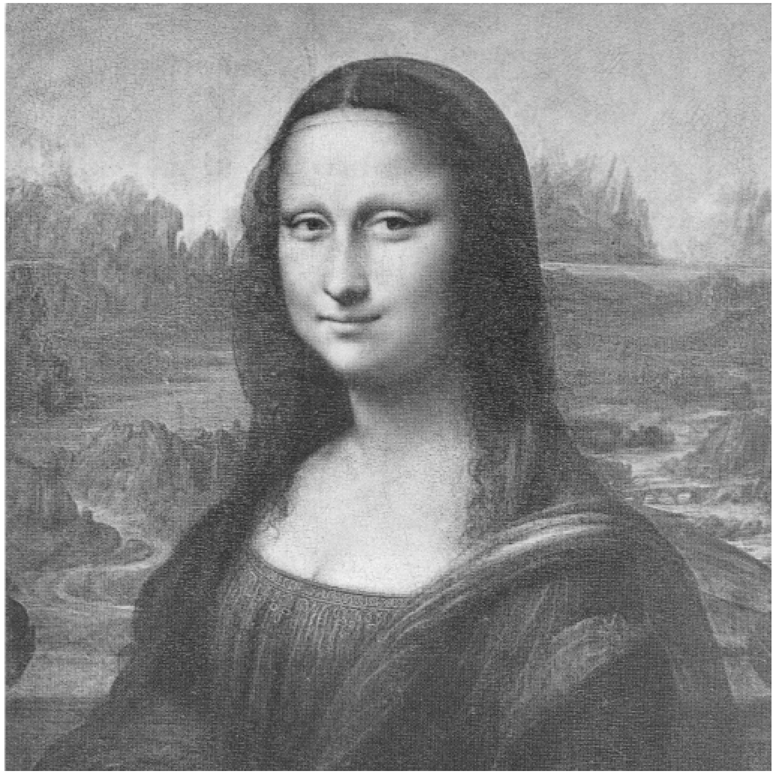


Multiply the image and the mask
Compute the result for location (x,y)

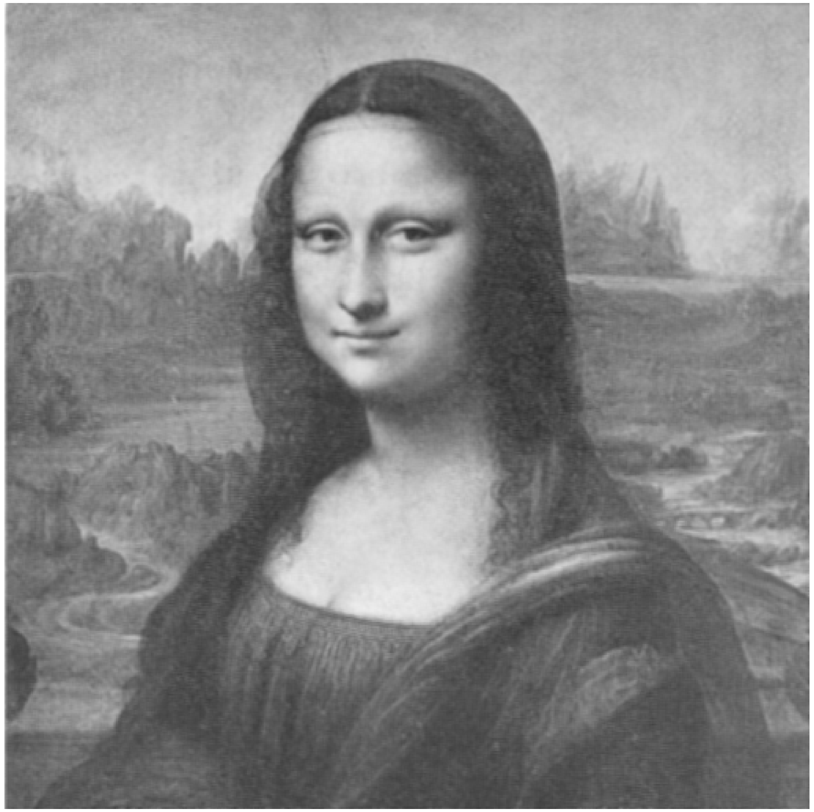
Mean filters

- 3×3: $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
- 5×5: $\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$
- 7×7: $\frac{1}{49}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

- Scale the result by the sum of the coefficients



Original



Filtered 3x3

9x9

25x25

Gradients

- Gradient of *F* along r in direction θ

$$\frac{\partial F}{\partial r} = \frac{\partial F}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial F}{\partial y}\frac{\partial y}{\partial r}$$

$$\frac{\partial F}{\partial r} = \frac{\partial F}{\partial x}\cos\theta + \frac{\partial F}{\partial y}\sin\theta$$

- Largest gradient when $\frac{\partial}{\partial \theta}\left(\frac{\partial F}{\partial r}\right) = 0$

- This is the angle θ_g where

$$-\frac{\partial F}{\partial x}\sin\theta_g + \frac{\partial F}{\partial y}\cos\theta_g = 0 \Leftrightarrow \frac{\partial F}{\partial y}\cos\theta_g = \frac{\partial F}{\partial x}\sin\theta_g$$

- $g_x = \delta F/\delta x$ and $g_y = \delta F/\delta x$ are the horisontal and vertical components of the gradient.

Gradient magnitude and direction

- Gradient direction:

$$\frac{g_y}{g_x} = \frac{\sin\theta_g}{\cos\theta_g} = \tan\theta_g \qquad \theta_g = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

- Gradient magnitude

$$\left(\frac{\partial F}{\partial r}\right)_{\max} = \left[g_x^2 + g_y^2\right]^{1/2}$$

Gradient-filters

- Prewitt-operator

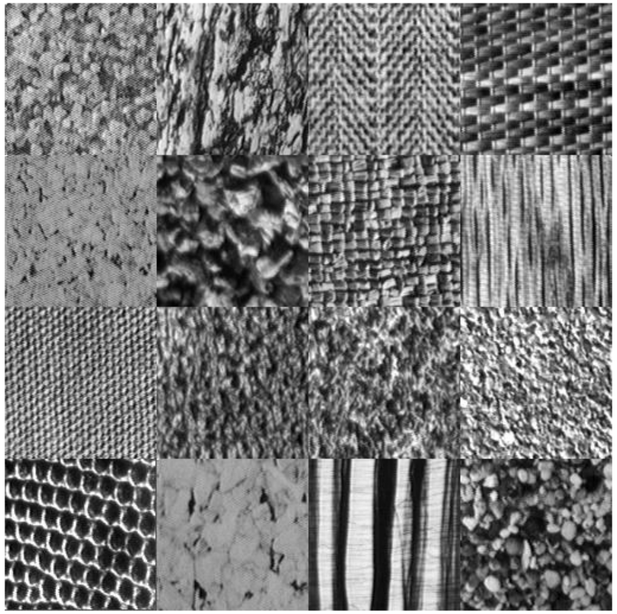
$$H_x(i,j)=\begin{bmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{bmatrix} H_y(i,j)=\begin{bmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{bmatrix}$$

- Sobel-operator

$$H_x(i,j)=\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix} H_y(i,j)=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}$$

Frequency content in images

- Low frequencies
  - Small gray level changes
  - Homogeneous ares
- High frequencies:
  - Large variations
  - Edges



Sinusoids in images

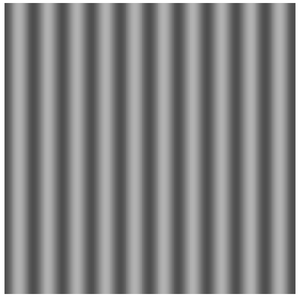
$$f(x,y) = 128 + A\cos\left(\frac{2\pi(ux+vy)}{N} + \phi\right)$$

A - amplitude

u - horisontal frequency

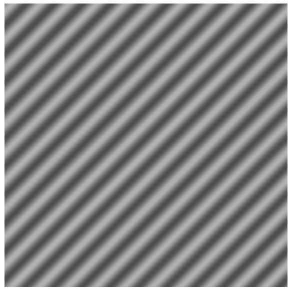
v - vertical frekvency

ϕ - phase

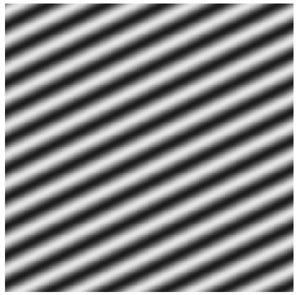


A=50, u=10, v=0

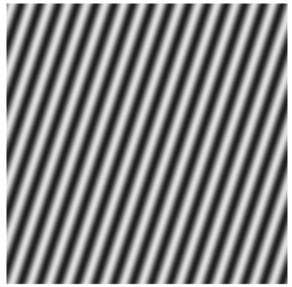
A=20, u=0, v=10



A=50, u=10, v=10



A=100, u=5, v=10



A=100, u=15, v=5

Note: u and v are the number of cycles (horisontally and vertically) in the image

2D Discrete Fouriertransform (DFT)

f(x,y) is a pixel in a N×M image

Definition:
$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$$

$$e^{j\theta} = \cos\theta + j\sin\theta$$

This can also be written:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\left[\cos(2\pi(ux/M+vy/N)) - j\sin\left(2\pi(ux/M+vy/N)\right)\right]$$

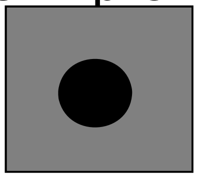
Inverse transform:

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

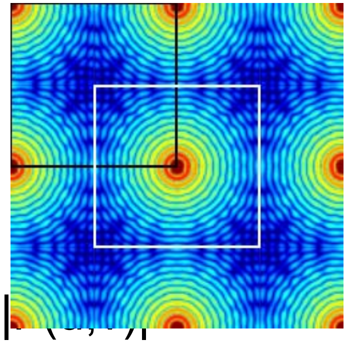
Displaying the Fourier spectrum

- Since F(u,v) is periodic with period N, it is common to translate the spectrum such that origo (u=v=0) is in the center of the image
  - Change quadrants

  OR: pre-multiply f(x,y) with $(-1)^{x+y}$



f(x,y)



|F(u,v)|

f(x,y): image domain

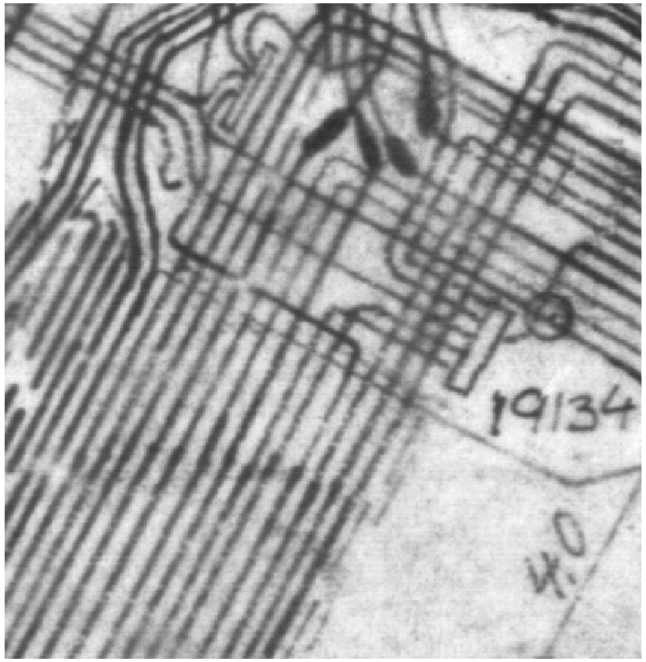
F(u,v): frequency domain

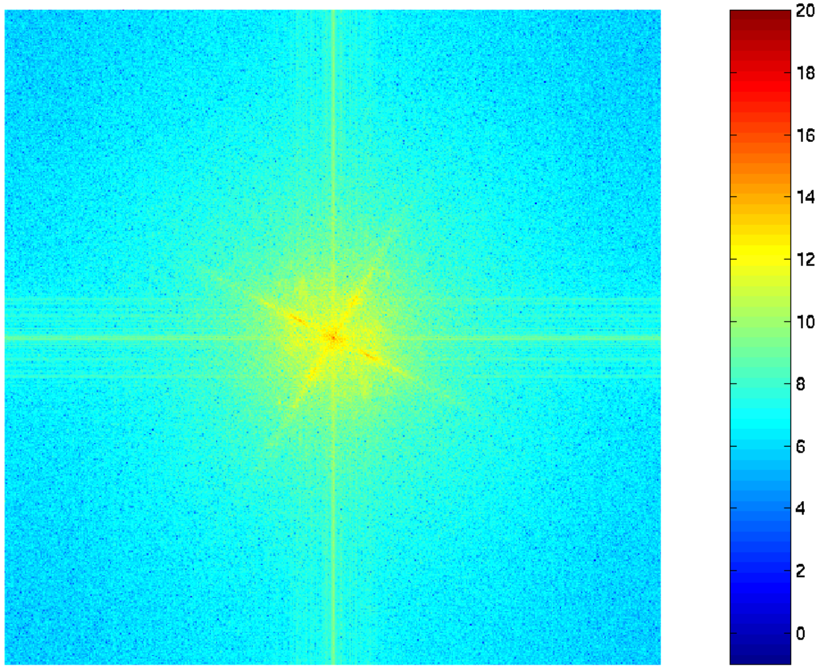
|F(u,v)| is called the spectrum of f(x,y)

(amplitude spectrum)

Power spectrum: $|F(u,v)|^2$

2D Fourier spectrum - oriented structure





The convolution theorem

$$f(x,y) * h(x,y) \Leftrightarrow F(u,v) \cdot H(u,v)$$

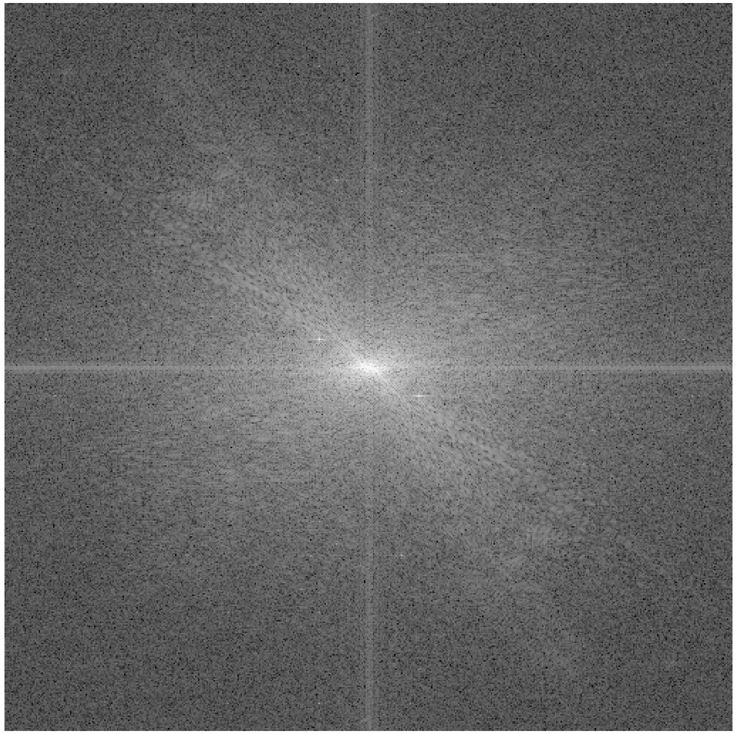
Convolution in the image domain ⇔ Multiplication in the frequency domain

$$f(x,y) \cdot h(x,y) \Leftrightarrow F(u,v) * H(u,v)$$

And the opposite:

Multiplication in the image domain ⇔ Convolution in the frequency domain

How do we filter out this effect?



Filtering in the frequency domain

1. Multiply the image by $(-1)^{x+y}$ to center the transform.
2. Compute F(u,v) using the 2D DFT
3. Multiply F(u,v) by a filter H(u,v)
4. Compute the inverse FFT of the result from 3
5. Obtain the real part from 4.
6. Multiply the result by $(-1)^{x+y}$

The ”ideal” low pass filter

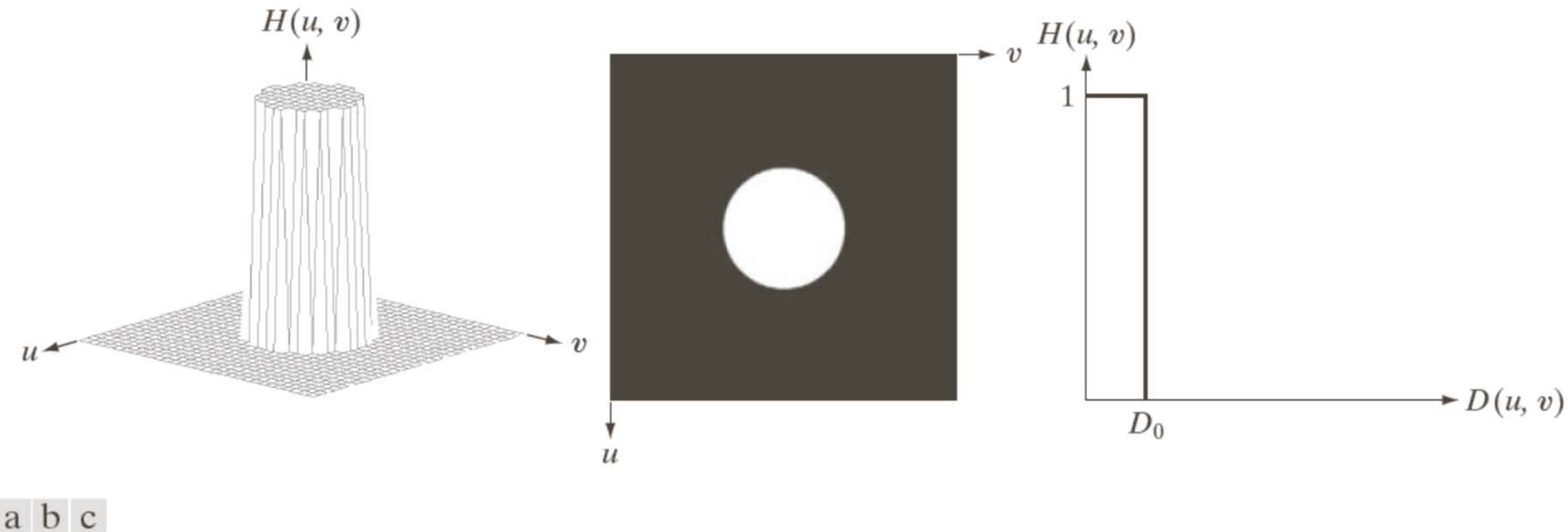


FIGURE 4.40 (a) Perspective plot of an ideal lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

Example - ideal low pass

Original

r_0=0.2

r_0=0.3

Warning: Look at these image in high resolution. You should see ringing effects in the two rightmost images.

Butterworth low pass filter

- Window-functions are used to reduce the ringing effect.
- Butterworth low pass filter of order *n*:

$$H(u,v) = \frac{1}{1 + \left[D(u,v) / D_0\right]^{2n}}$$

- D_0 describes the point where H(u,v) has decreased to the half of its maximum)
  - Low filter order (*n* small): H(u,v) decreases slowly: Little ringing
  - High filter order (*n* large): H(u,v) decreases fast: More ringing

- Other filters can also be used, e.g. Gaussian, Bartlett, Blackman, Hamming, Hanning

Gaussian lowpass filter

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

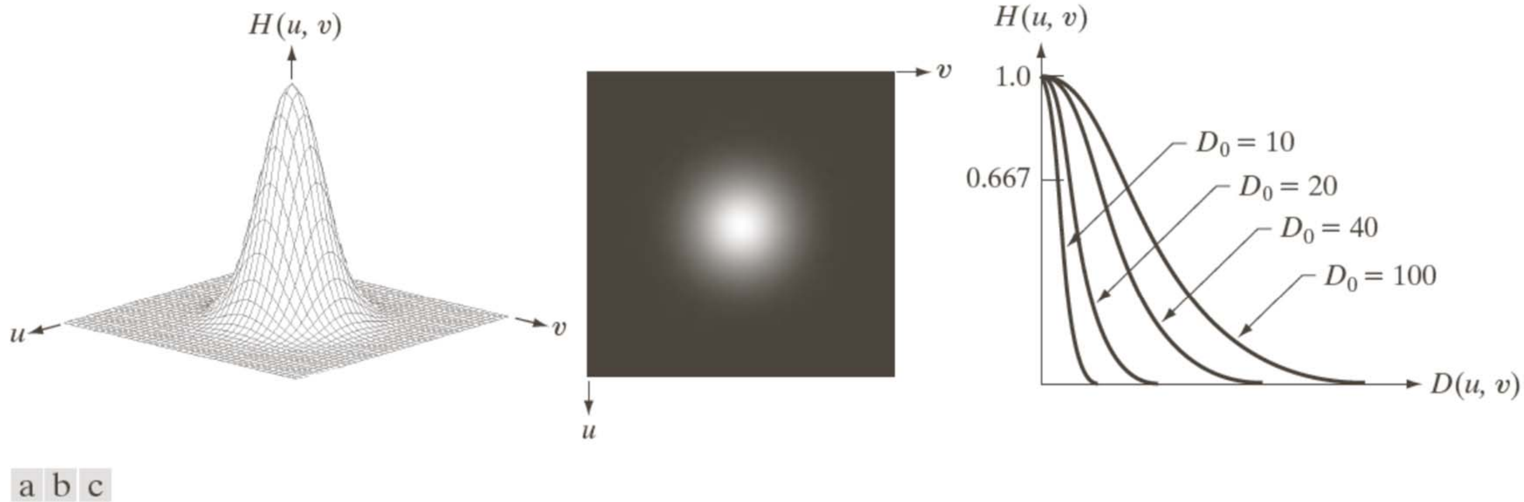


FIGURE 4.47 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

High pass filtering

- Simple ("Ideal") high pass filter:

$$H_{hp}(u,v) = \begin{cases} 0, D(u,v) \le D_0, \\ 1, D(u,v) > D_0. \end{cases}$$

or

$$H_{hp}(u,v) = 1 - H_{lp}(u,v)$$

- Butterworth high pass filter:

$$H_{hpB}(u,v) = \frac{1}{1 + \left[D_0 / D(u,v)\right]^{2n}}$$

- Gaussian high pass filter:

$$H_{hpG}(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

Ideal, Butterworth and Gaussian highpass filters

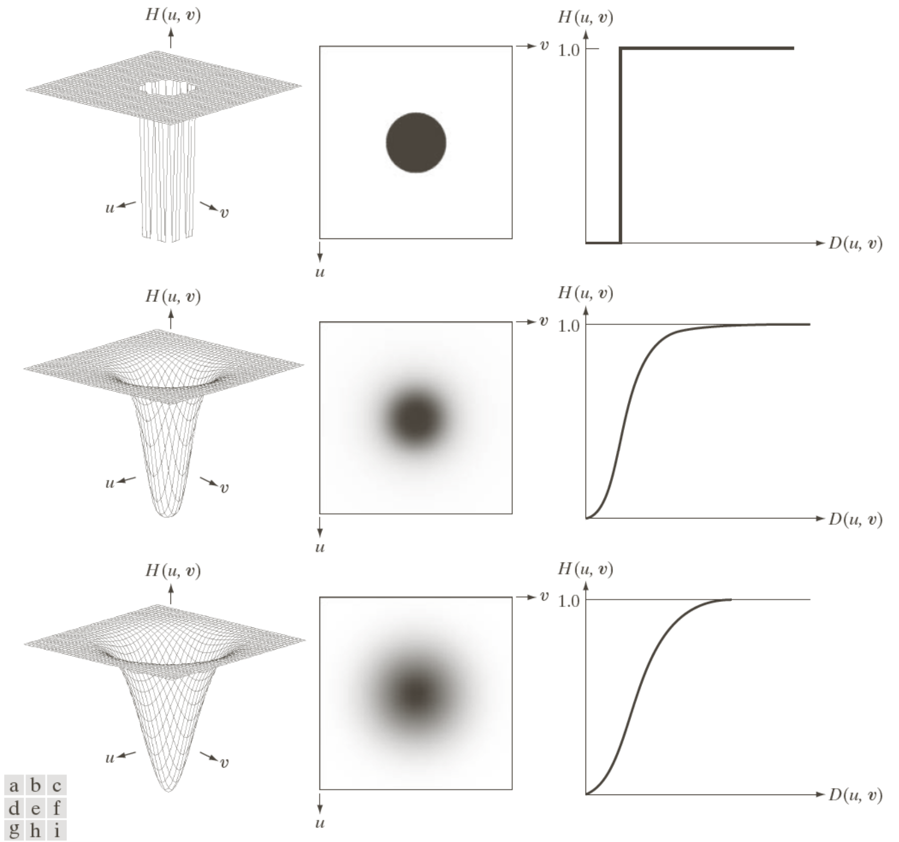


FIGURE 4.52 Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

Example – Butterworth highpass

a b c

FIGURE 4.55 Results of highpass filtering the image in Fig. 4.41(a) using a BHPF of order 2 with $D_0 = 30, 60,$ and 160, corresponding to the circles in Fig. 4.41(b). These results are much smoother than those obtained with an IHPF.

Bandpass and bandstop filters

- Bandpass filter: Keeps only the energy in a given frequency band $<D_{low},D_{high}>$ (or $<D_0-W/2,D_0+ W/2>$)

- W is the width of the band
- D_0 is its radial center.

- Bandstop filter: Removes all energy in a given frequency band $<D_{low},D_{high}>$

Bandstop/bandreject filters

- Ideal

$$H_{bs}(u,v)=\begin{cases}1 & \text{if } D(u,v)<D_0-\frac{W}{2}\\ 0 & \text{if } \mathrm{D}_0-\frac{W}{2}\le D(u,v)\le D_0+\frac{W}{2}\\ 1 & \text{if } D(u,v)>D_0+\frac{W}{2}\end{cases}$$

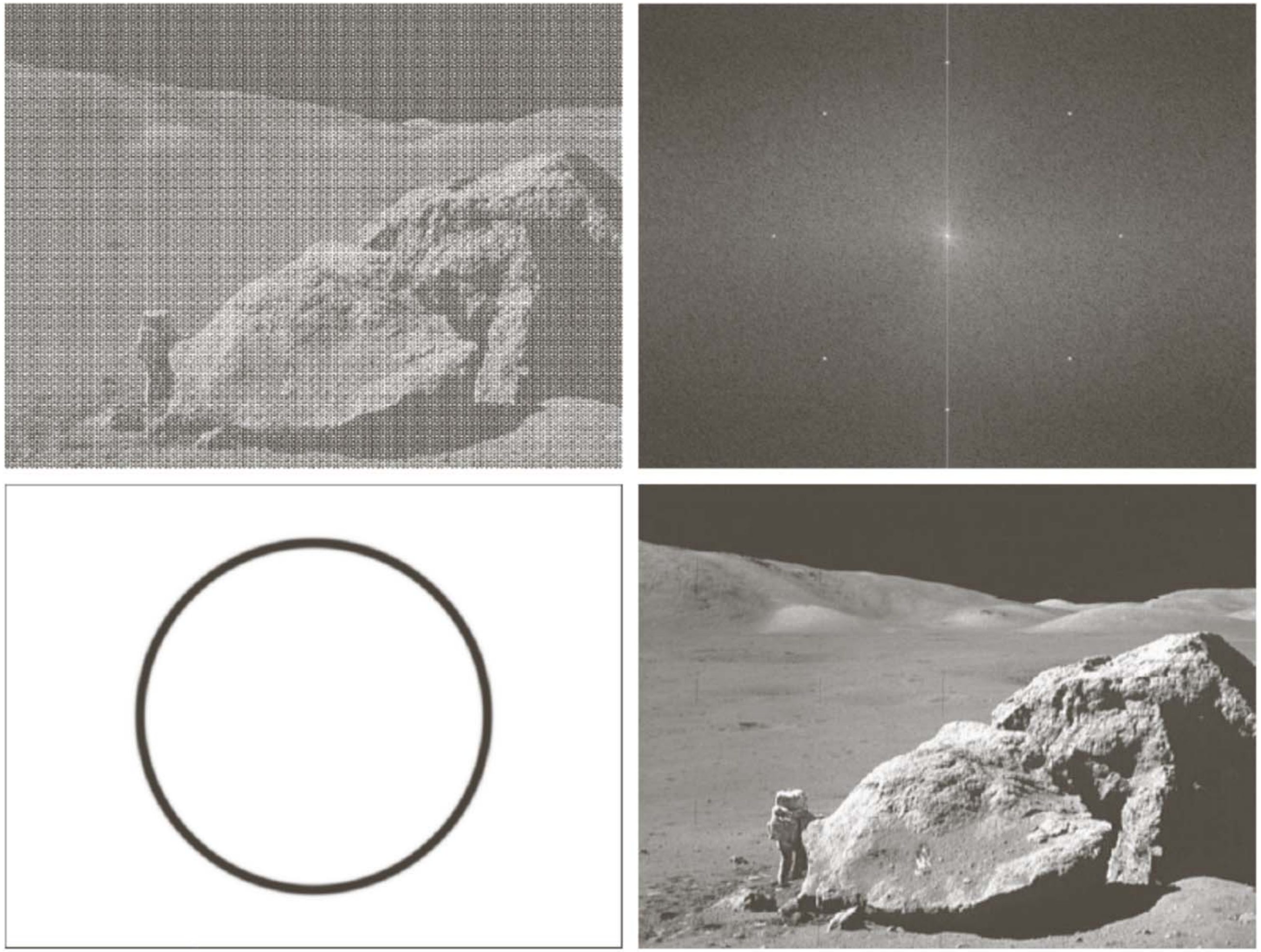
- Butterworth

$$H_{bs\mathbf{B}}(u,v)=\frac{1}{1+\left[\frac{D(u,v)W}{D^2(u,v)-D_0^2}\right]^{2n}}$$

- Gaussian

$$H_{bsG}(u,v)=1-e^{-\frac{1}{2}\left[\frac{D^2(u,v)-D_0^2}{D(u,v)W}\right]^2}$$

An example of bandstop filtering



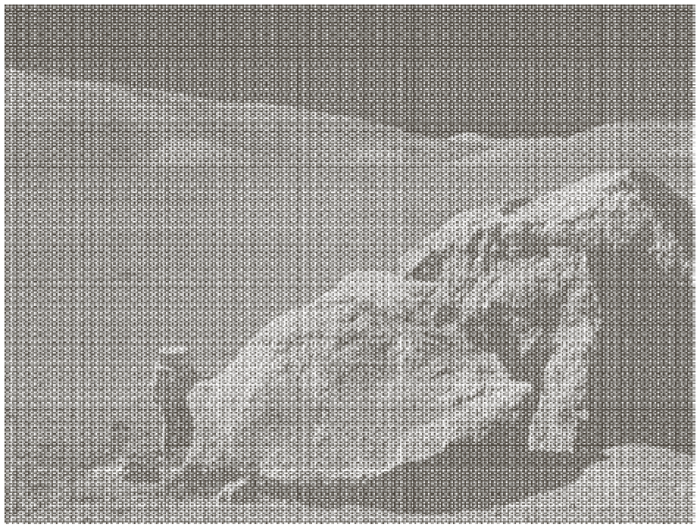
a	b
c	d

FIGURE 5.16 (a) Image corrupted by sinusoidal noise. (b) Spectrum of (a). (c) Butterworth bandreject filter (white represents 1). (d) Result of filtering. (Original image courtesy of NASA.)

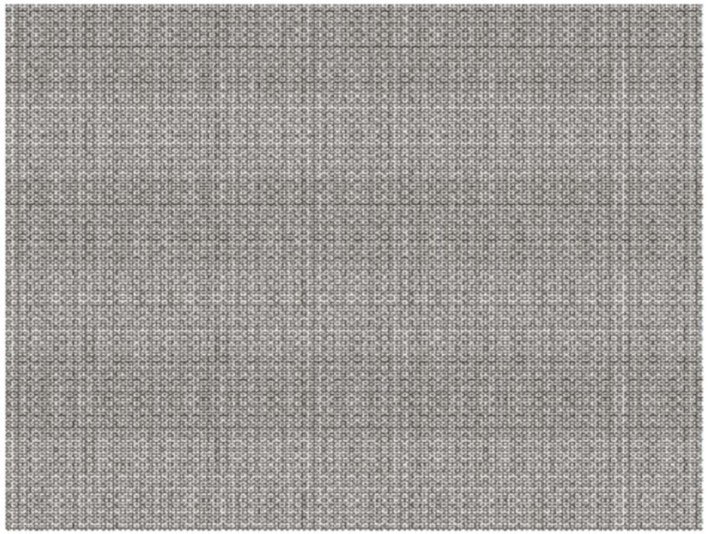
Bandpass filters

- Are defined by

$$H_{bp}(u,v) = 1 - H_{bs}(u,v)$$



Original



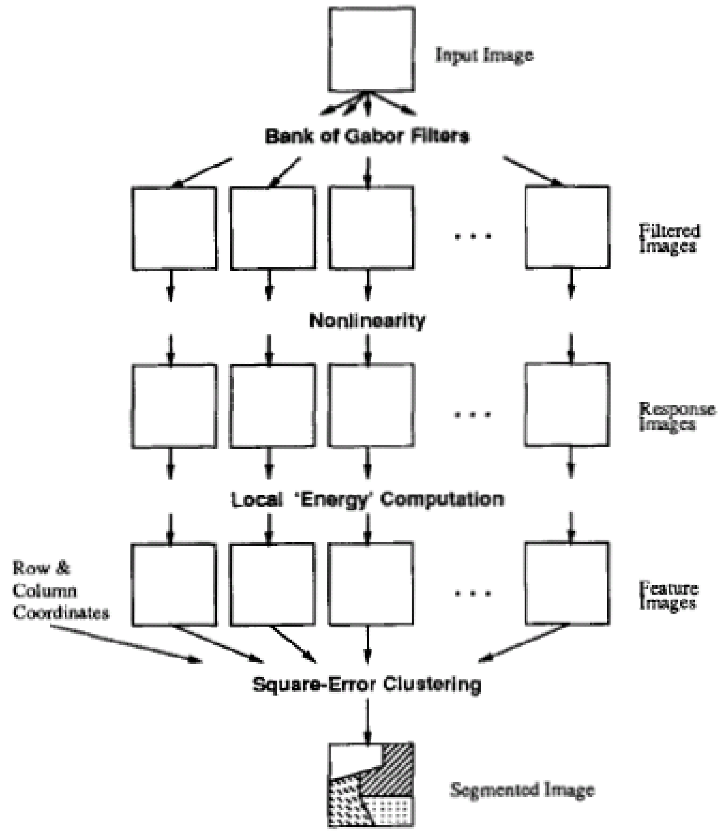
Result after bandpass filtering

Feature extraction from filter banks

- Idea: convolve the image with a set of filters to extract stuctures with different size and orientations
- In this lecture: Gabor filters, but we could use other types e.g wavelets
- These are low-level features similar to those extracted by convolutional net.
- Convolutional nets:
  - Estimate the filter coefficients themselves
  - Adapts a classifier also.

Fourier texture from filter banks

1. Compute the 2D Fourier spectrum of the image
2. Design a set of 2D bandpass filters that covers the spectrum with varying center frequency and orientations.
3. Apply each filter to the input image.
4. Apply a non-linear transform to the filtered images with the aim of creating homogeneous regions corresponding to texture with a certain frequency and orientation.
5. Segmentation or classification based on all the images from after applying the non-linear transform.



Gabor-filters - spatial domain

- A popular method for designing filters.
- A 2D Gabor function consists of a sinusoidal plane wave of some frequency and orientation, modulated by a 2D Gaussian envelope, typically in the spatial domain:

$$h(x,y) = \exp\left\{-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right]\right\}\cos(2\pi u_0 x + \phi)$$

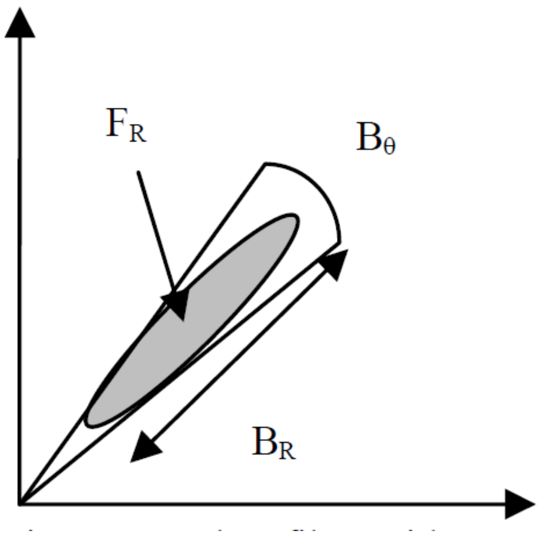
- u_0 and ϕ are the frequency and phase of the sinusoid along the x-axis (for orientation 0 degrees), and σ_x and σ_y are the width of the Gaussian envelope function.
- A Gabor filter with arbitrary orientation θ can be obtained by rotating the coordinate system.

Gabor-filters for extracting information from the Fourier spectrum

- A 2D Gabor filter H(u’,v’) with orientation θ and radial filter center F_R in the frequency domain is defined by rotating the coordinate system from (u,v) to (u’,v’):

$$H(u',v') = A\left(\exp\left\{-\frac{1}{2}\left[\frac{(u'-F_R)^2}{\sigma_u^2}+\frac{v'^2}{\sigma_v^2}\right]\right\}+\exp\left\{-\frac{1}{2}\left[\frac{(u'+F_R)^2}{\sigma_u^2}+\frac{v'^2}{\sigma_v^2}\right]\right\}\right)$$

$$u' = u\cos\theta + v\sin\theta, \quad v' = -u\sin\theta + v\cos\theta$$



- The filter consists of a sinusoidal wave modulated with a 2D Gaussian. They have high resolution both in the spatial and the Fourier domain.
- The filter is specified in terms of radial filter bandwidth B_R in octaves and angular filter width B_θ in radians and use the conversion:

$$\sigma_u = \frac{1}{\sqrt{-2\log(1/2)}}\frac{2^{B_R}-1}{2^{B_R}+1}F_R, \quad \sigma_v = \frac{\tan(B_\theta/2)}{\sqrt{-2\log(1/2)}}F_R.$$

Gabor filters - filter parameters

- The following filter parameters are normally used:
- Orientations: 0, 45, 90, 135
- For an image of size N, radial frequencies
  $1\sqrt{2}, 2\sqrt{2}, 4\sqrt{2}, 8\sqrt{2}, 16\sqrt{2}$,....... but the lowest can often be skipped
- The frequencies are one octave apart.

Gabor filters: feature images

- The purpose of the non-linear transform is to transform the band-pass filtered result, which has large amplitude fluctuations with the given frequency in regions where the selected frequency has a significant presence in the Fourier spectrum.
- The following function is often used:

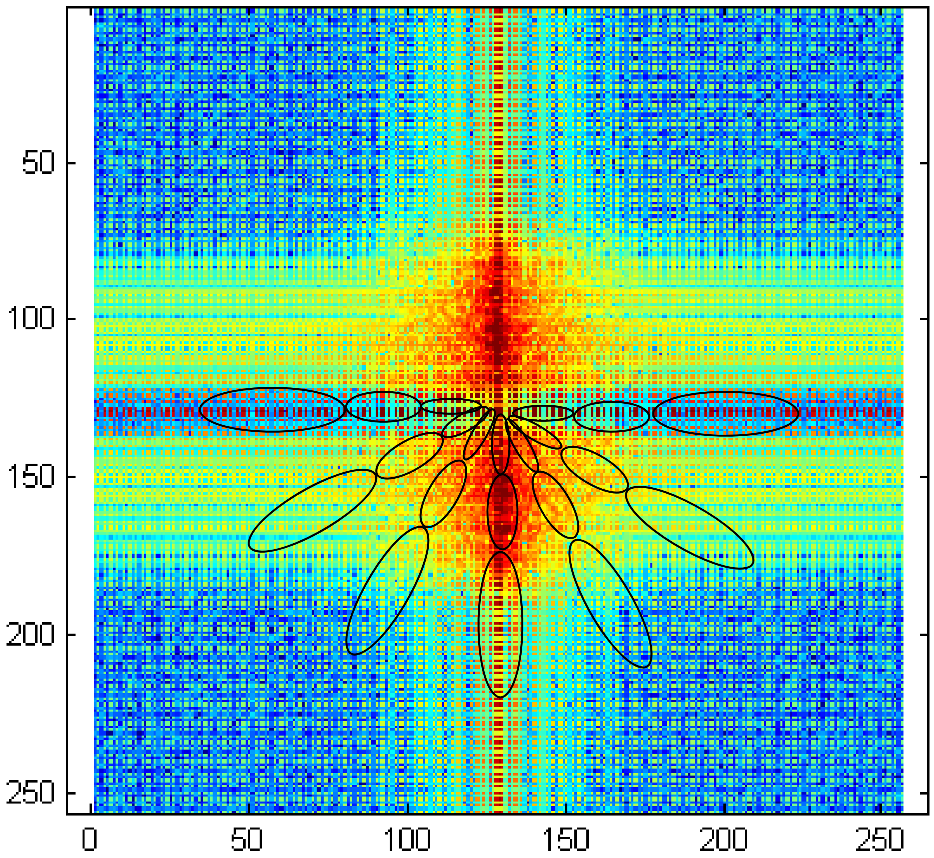
$$\psi(t) = \tanh(\alpha t) = \frac{1 - e^{-2\alpha t}}{1 + e^{-2\alpha t}},$$

- This will act as a blob detector for regions corresponding high amplitude value for the given frequency.
- This is later combined with computing texture energy as average absolution deviation in small windows:

$$e_k(x, y) = \frac{1}{M^2} \sum_{(a,b)\in W_{xy}} |\psi(r_k(a, b))|,$$

- The size of the window is proportional to the average size of the intensity variations in the image T=N/u0.

Sampling the Fourier spectrum with a filter bank of Gabor filters

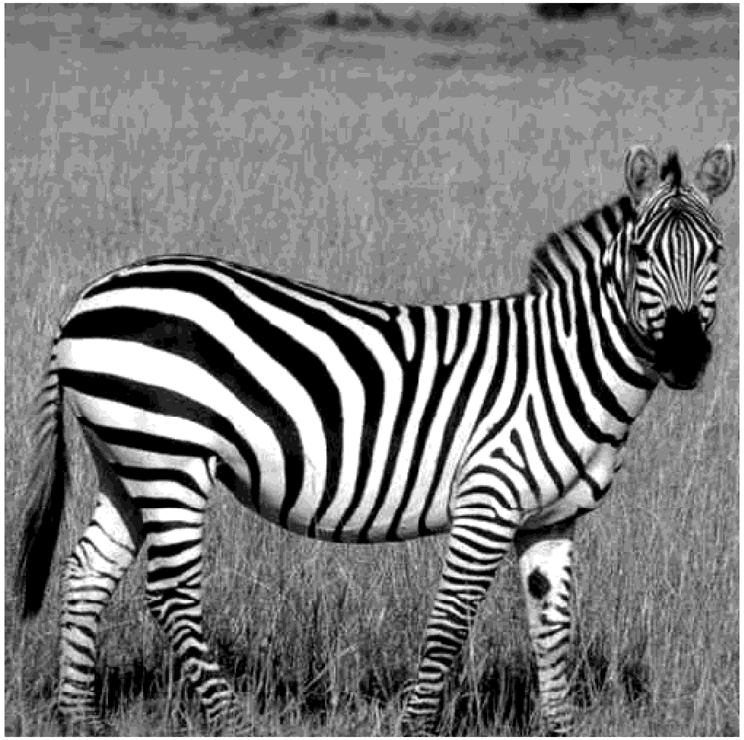


A bank of 24 filters with different center frequencies and orientation sample the Fourier domain

Orientations:
0 °, 30 °, 45 °, 60 °, 90 °, 120 °, 135 ° and 150 °

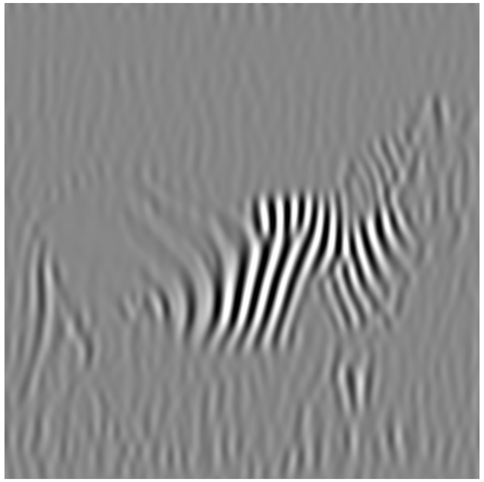
Frequencies:
u_0=0.1N, 0.15N and 0.35N
N is the image size.

Gabor filter : example

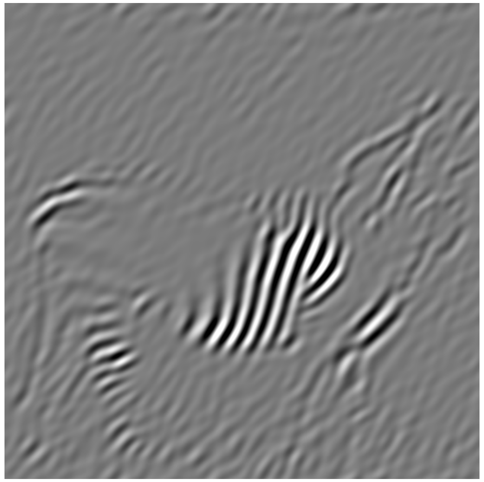


Filter with 4 Gabor filters with equal center frequency and Orientation 0, 45, 90 and 135 degrees

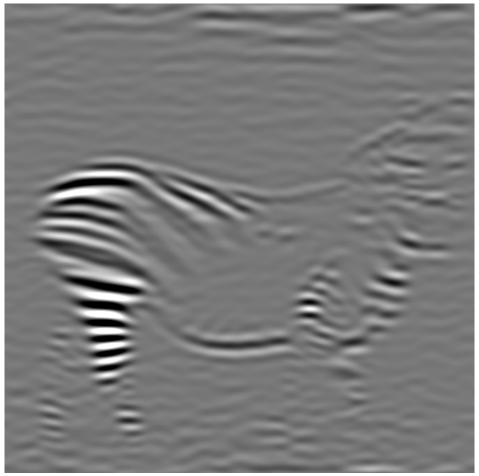
Gabor-filtered images



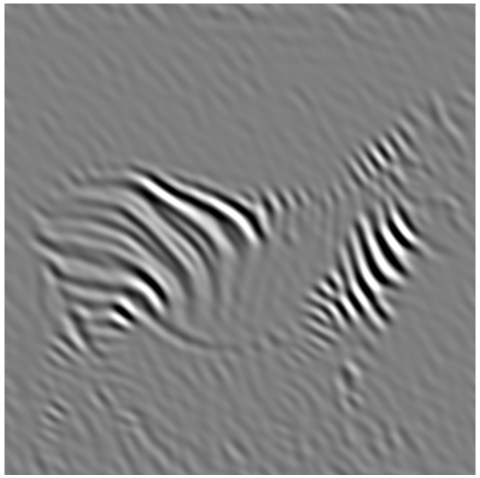
0



45



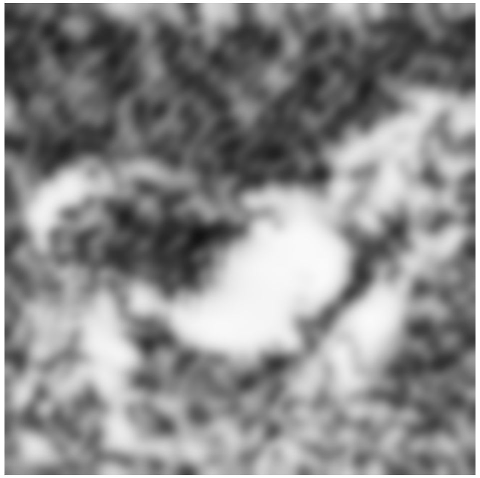
90



135

Gabor feature images after nonlinear transform

0



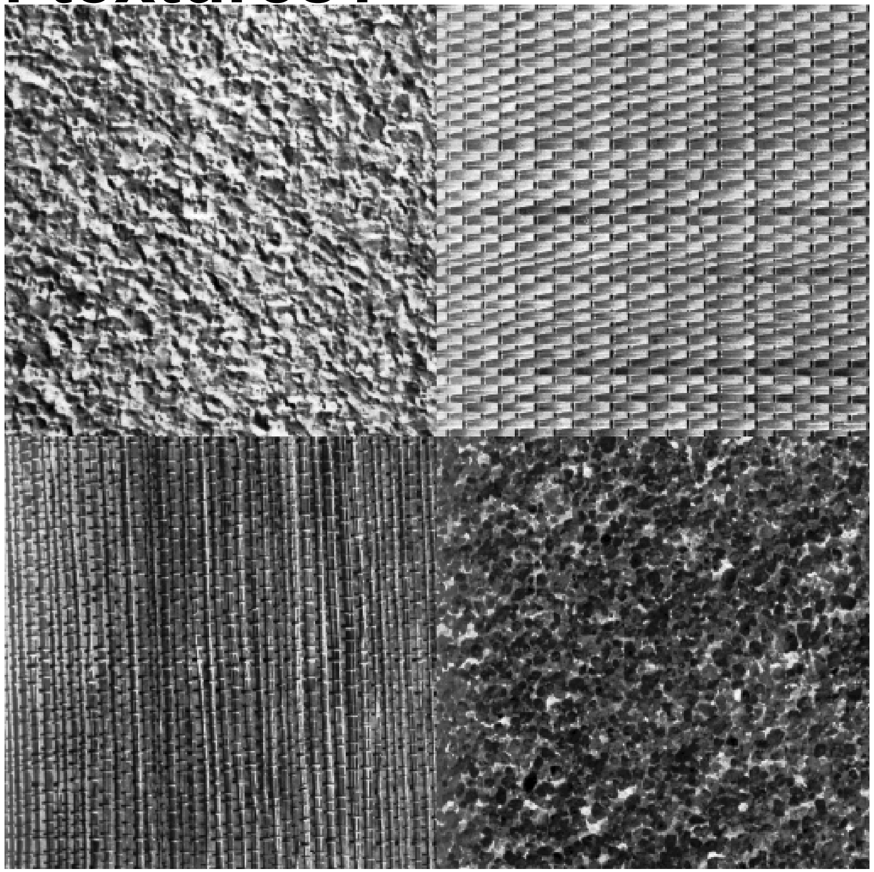
45

90

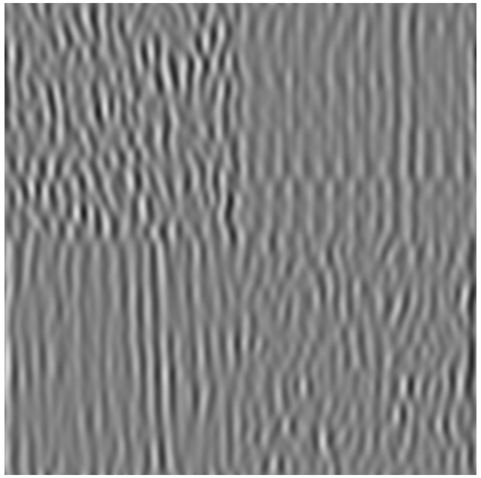
135

Simple feature combination: the average image

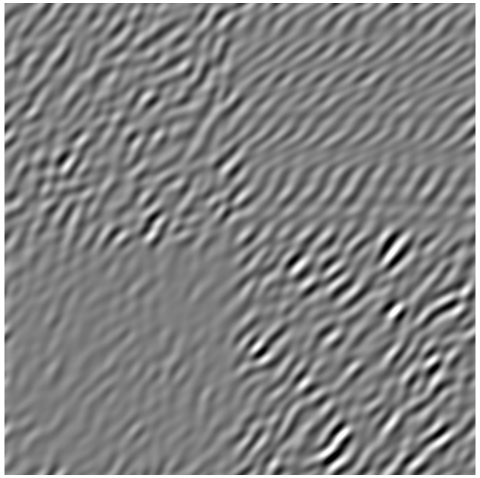
Example 2: texture segmentation Can we segment the boundaries between the 4 textures?



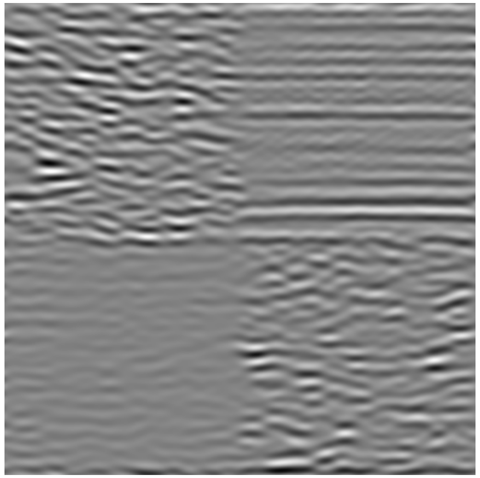
Gabor-filtered images



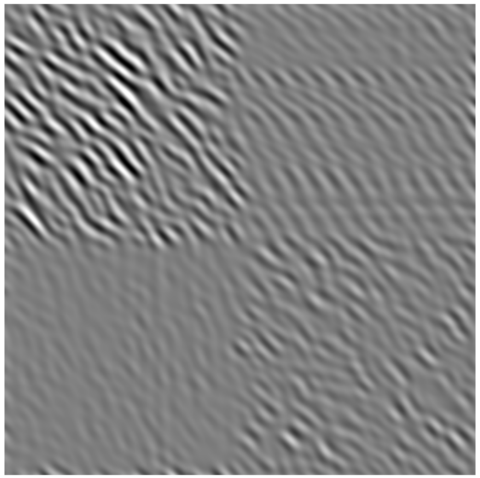
0



45



90



135

Gabor feature images after nonlinear transform

0

45

90

135

Simple feature combination: the average image

- Classification based on these images might be possible, but is not stable with respect to size, orientation etc.
- What if the object was a cat? Or horse?
- We see that this approach builds a set of primities as edges in given orientations.
- Filter coefficients must be determined by the used.
- To detect over different scales we could either resample the images and use one filter set, or use filter sets of different sizes.
- At least these filters are sensitive to orientation patterns.