HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Computer Vision

Chapter 3.2: Image transform

Nguyễn Thị Oanh
oanhnt@soict.hust.edu.vn

Computer Vision

Chapter 3.2 Image transform

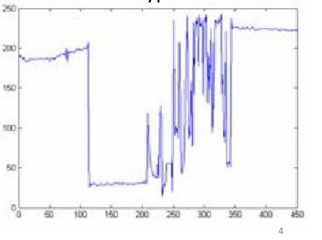
Nguyễn Thị Oanh
oanhnt@soict.hust.edu.vn

Content

- Rappel: digital image representation
- Point Processing
- Convolution and Linear filtering
- More neighborhood operators
- Image transforms
  - Frequency domain
    - Frequencies in images
    - Fourrier transform
    - Frequential Processing (frequential filters)
  - PCA (additional reading)

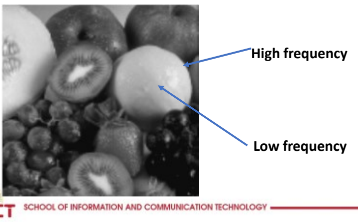
Frequencies in images

- Image:
  - matrix of pixels
  - a signal: a pixel ~ a sample in the image signal



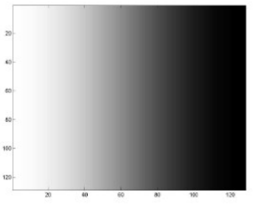
Frequencies in images

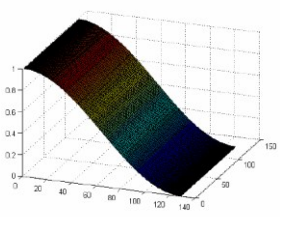
- What are the (low/high) frequencies in an image?
  - Frequency **= intensity change**
  - Slow changes (homogeneous /blur regions): low frequency
  - fast/abrupt changes (egde, contour, noise): high frequency



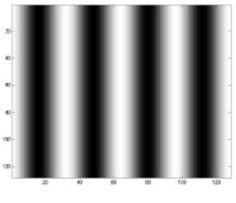
Most of energy concentrated in low frequencies

Low frequencies





High frequencies



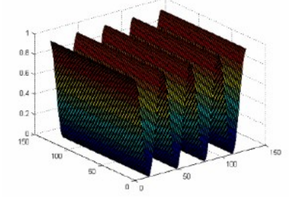


Image spectral analysis

- An image is a visual signal
  - We can analyse the frequencies of the signal
- How?
  - we will create a new « image » which contains all frequencies of the image
    - Like a 2D frequency graphic
  - The basic tool for it is the **Fourier Transform**
- We talk about the **frequency domain**, opposing to the **spatial domain** (image)

Frequencies in a signal

High frequency signal →

...

Low frequency signal →

Sum of all the above signals →

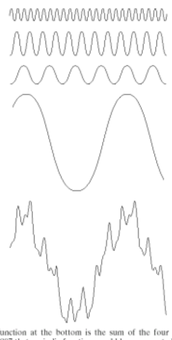


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

Fourier series

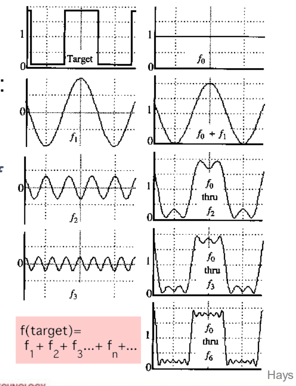
A bold idea (1807) - Jean Baptiste Joseph Fourier (1768-1830):

***Any** univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.*

Our building block:

$$A\sin(\omega t) + B\cos(\omega t)$$

Add enough of them to get any signal *g(t)* you want!

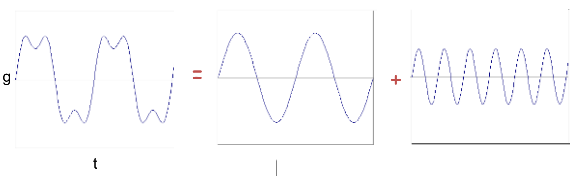


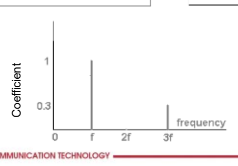
Hays

Example

$t = [0,2], f = 1$

$g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi (3f)\, t)$

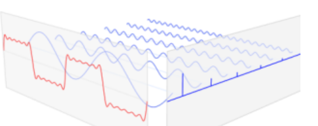




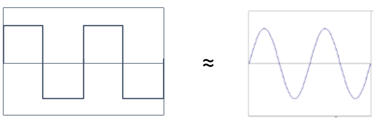
Slides: Efros

Fourier Transform

- Fourier transform is a mathematical transform that
  - Decomposes functions depending on space or time into functions depending on spatial or temporal frequency

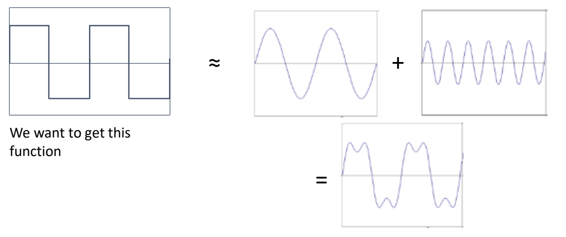


Fourier Series



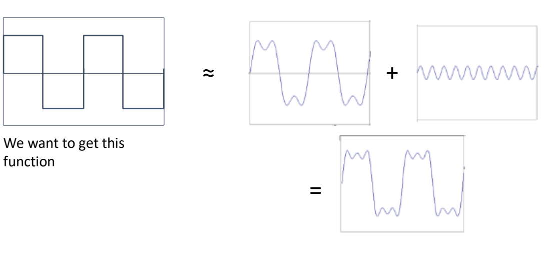
We want to get this function

Fourier Series



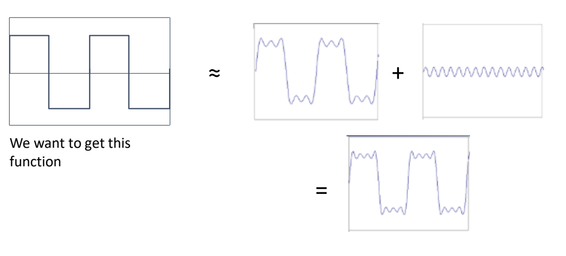
We want to get this function

Fourier Series



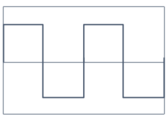
We want to get this function

Fourier Series



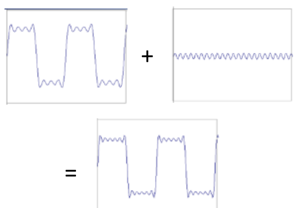
We want to get this function

Fourier Series



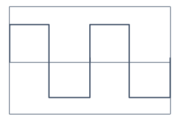
We want to get this function

≈



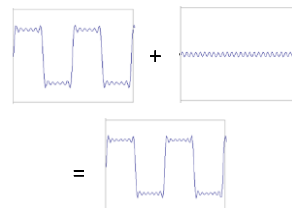
Slide by Alexei A. Efros

Fourier Series



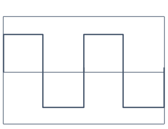
We want to get this function

≈



Slide by Alexei A. Efros

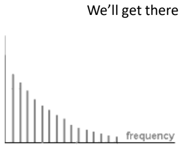
Fourier Series



We want to get this function

$$= A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)$$

We'll get there in the limit



Slide by Alexei A. Efros

Discrete Fourier transform

$$H_{f_j} = \frac{1}{N}\sum_k h_{t_k} e^{2\pi i f_j t_k}$$

$$h_{t_j} = \frac{1}{N}\sum_k H_{f_k} e^{-2\pi i f_k t_j}$$

where the t_k are the time corresponding to my signal in the time domain h_{t_k}, f_k are the corresponding frequency to my signal in the frequency domain, and N is the number of points of tl signal data.

Fourier Transform Equation Explained:
https://www.youtube.com/watch?v=8V6Hi-kP9EE&ab_channel=IainExplainsSignals%2CSystems%2CandDigitalComms

Magnitude and phase

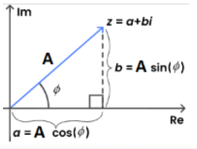
- Fourier transform stores the **magnitude** and **phase** at each frequency
  - Magnitude encodes how much signal there is at a particular frequency
  - Phase encodes spatial information (indirectly)
  - For mathematical convenience, this is often notated in terms of complex number (with real and imaginary part)

Amplitude:

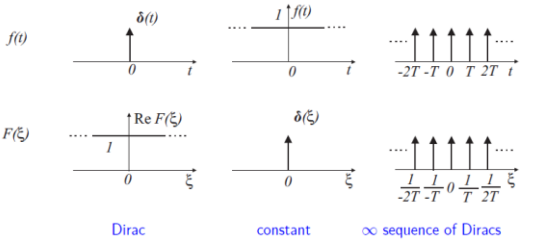
$$A = \pm\sqrt{R(\omega)^2 + I(\omega)^2}$$

Phase:

$$\phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$$

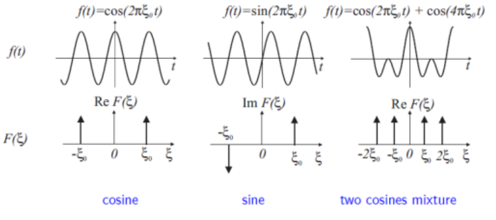


Basic Fourier Transform pairs



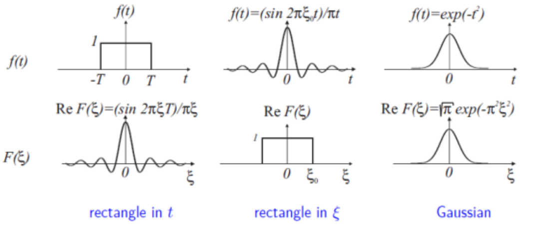
Source: Václav Hlaváč - **Fourier transform, in 1D and in 2D**

Basic Fourier Transform pairs



Source: Václav Hlaváč - **Fourier transform, in 1D and in 2D**

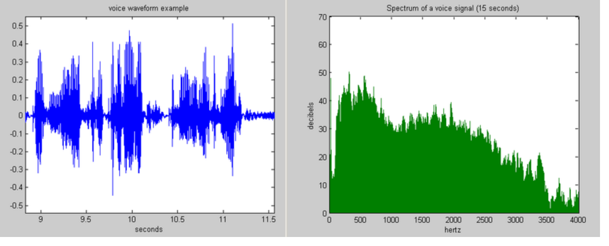
Basic Fourier Transform pairs



Source: Václav Hlaváč - **Fourier transform, in 1D and in 2D**

Example: Music

- We think of music in terms of frequencies at different magnitudes



Slide: Hoiem

2D FT - discrete

Direct transform

$$F(u,v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \exp\left[-2\pi i \left(\frac{mu}{M} + \frac{nv}{N}\right)\right],$$

$$u = 0, 1, \ldots, M-1, \qquad v = 0, 1, \ldots, N-1,$$

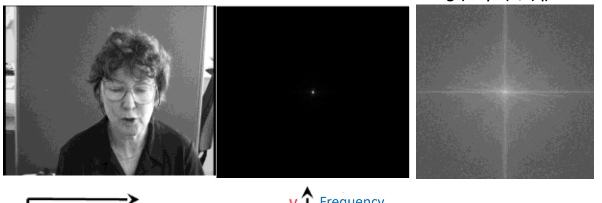
Inverse transform

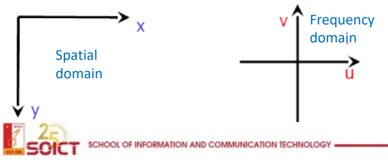
$$f(m,n) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \exp\left[2\pi i \left(\frac{mu}{M} + \frac{nv}{N}\right)\right],$$

$$m = 0, 1, \ldots, M-1, \qquad n = 0, 1, \ldots, N-1.$$

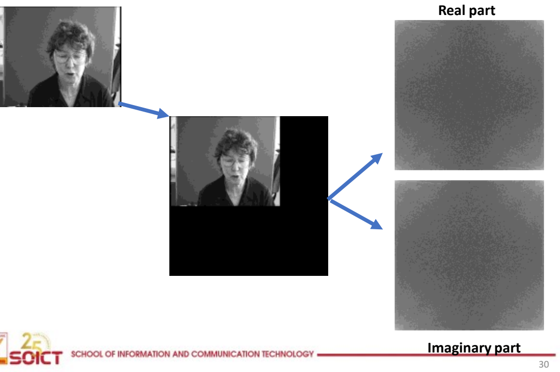
Image Fourier transform

Original image **Spectra | F(u,v) |** **Enhanced Spectra log (1+ | F(u,v) |)**



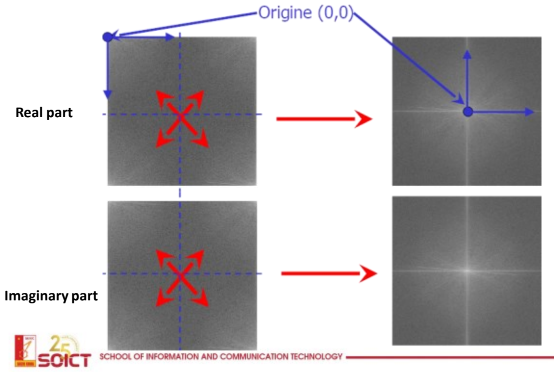


FFT: fast fourier transform

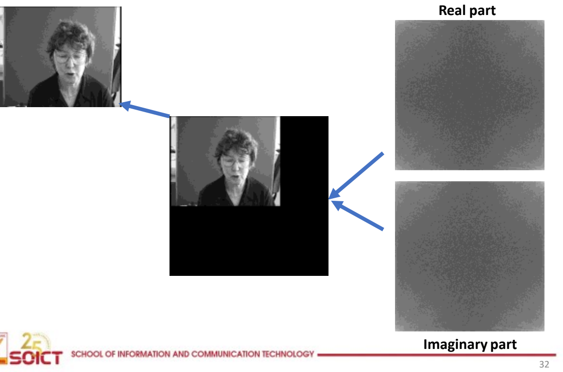


FFT

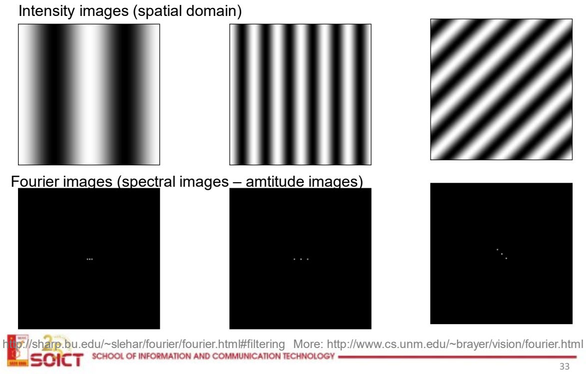
For visualization purposes we may also rearrange the quadrants of the result, so that the origin (0, 0) corresponds with the image center.



Inverse FFT

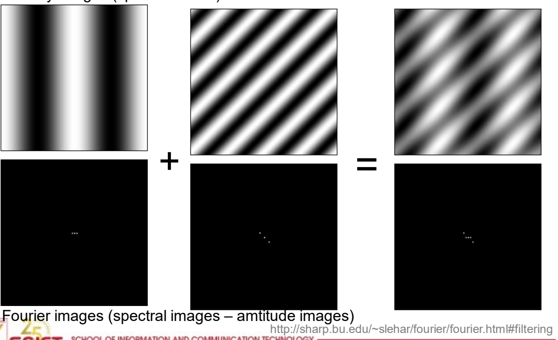


Fourier analysis in images



http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering More: http://www.cs.unm.edu/~brayer/vision/fourier.html

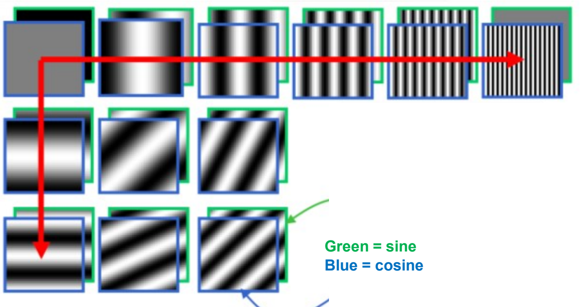
Signals can be composed



http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

Fourier Bases

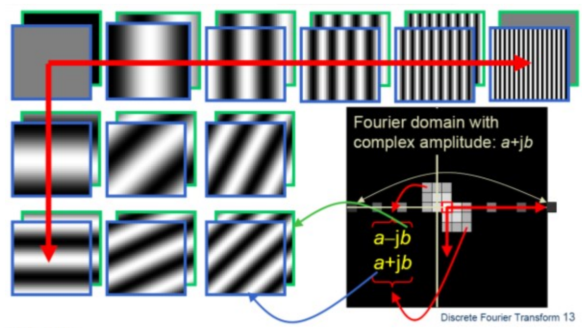
Teases away 'fast vs. slow' changes in the image.



This change of basis is the Fourier Transform

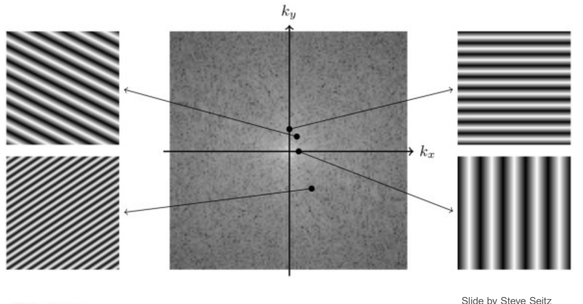
Hays

Fourier Bases



Hays

2D Fourier Transform



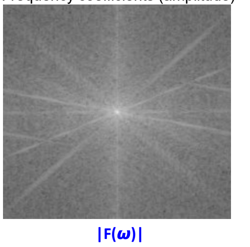
Slide by Steve Seitz

Fourier Transform of an image

Natural image

f(x,y)

Fourier decomposition
Frequency coefficients (amplitude)

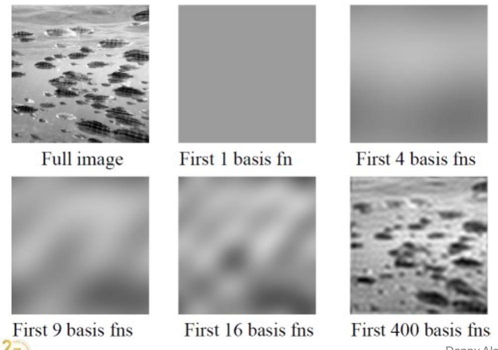


|F(ω)|

What does it mean to be at pixel x,y?
What does it mean to be more or less bright in the Fourier decomposition image?

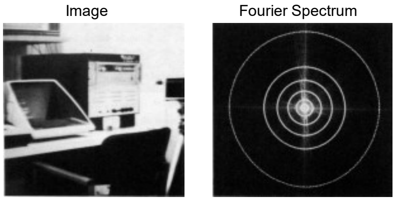
Slide by Steve Seitz

Basis reconstruction



Danny Alexander

2D Fourier transform

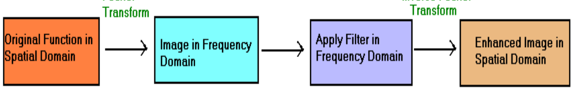


Percentage of image power enclosed in circles (small to large) :
90, 95, 98, 99, 99.5, 99.9

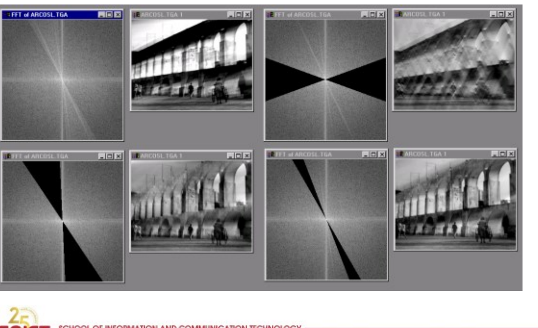
Most of energy concentrated in low frequencies

Image filtering in the frequential domain

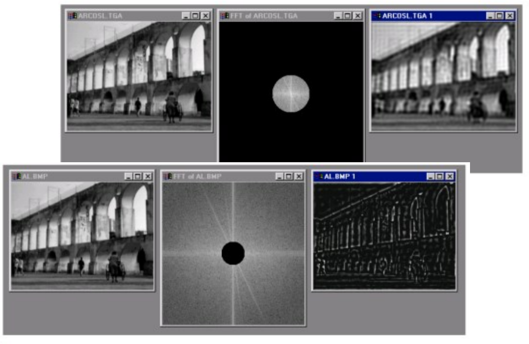
- We will be dealing only with functions (images) of finite duration so we will be interested only in Fourier Transform



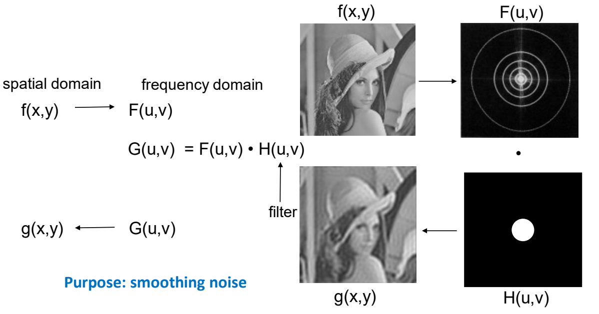
Now we can edit frequencies!



Low-pass and high-pass filtering



Low-pass filter



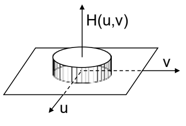
Purpose: smoothing noise

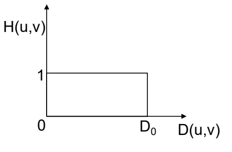
H(u,v) - Ideal low-pass filter

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

$D(u,v) = \sqrt{u^2 + v^2}$

D_0 = **cut off frequency**





Blurring - Ideal low-pass filters

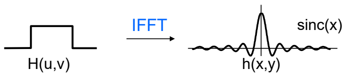
The **blurring** can be increased by increasing the **size** of **the mask**

The ringing problem

$G(u,v) = F(u,v) \cdot H(u,v)$

Convolution Theorem

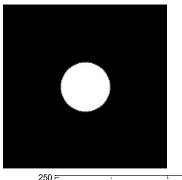
$g(x,y) = f(x,y) * h(x,y)$



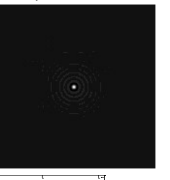
$\uparrow D_0 \longrightarrow \downarrow$ Ringing radius + $\downarrow$ blur

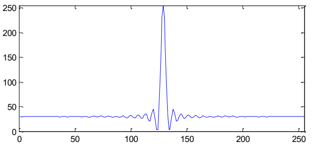
The ringing problem

Freq. domain

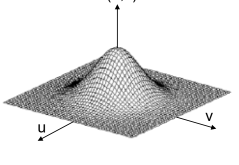


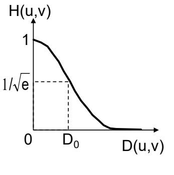
Spatial domain





H(u,v) - Gaussian filter

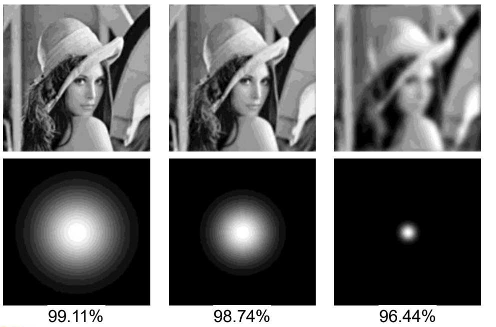




$H(u,v) = e^{-D^2(u,v)/(2D^2_0)}$ $\quad D(u,v) = \sqrt{u_2 + v_2}$

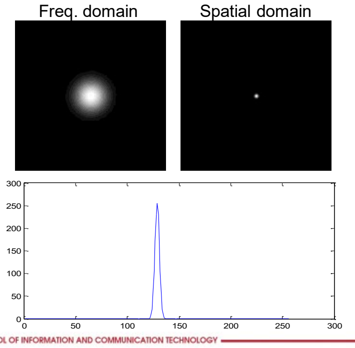
Softer Blurring + no Ringing

Blurring - Gaussain lowpass filter



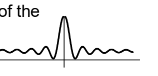
99.11% 98.74% 96.44%

The Gaussian lowpass filter



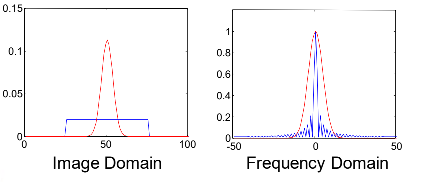
Blurring in the Spatial Domain

Averaging = convolution with $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ = point multiplication of the transform with sinc:



Gaussian Averaging = convolution with $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

= point multiplication of the transform with a gaussian.



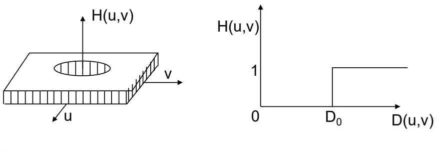
High-pass filter

H(u,v) - Ideal Filter

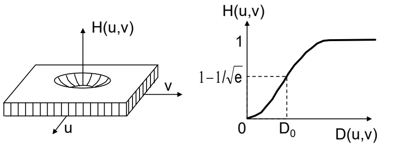
$$H(u,v) = \begin{cases} 0 & D(u,v) \le D_0 \\ 1 & D(u,v) > D_0 \end{cases}$$

$D(u,v) = \sqrt{u^2 + v^2}$

D_0 = **cut off frequency**



High-pass gaussian filter

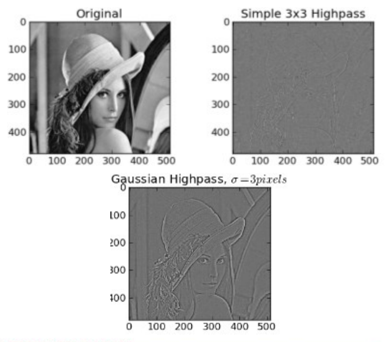


$$H(u,v) = 1 - e^{-D^2(u,v)/(2D^2_0)}$$

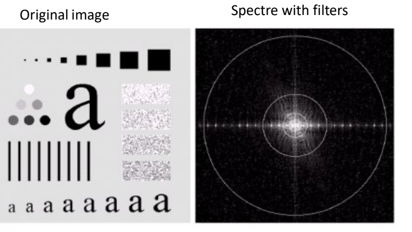
$$D(u,v) = \sqrt{u_2 + v_2}$$

High-pass filtering

$$h_1(3x3) = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



High pass filtering

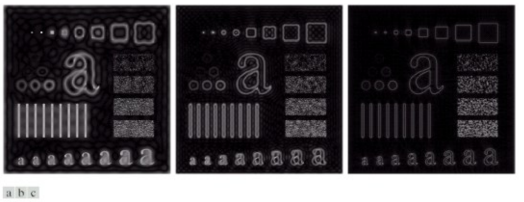


a b

FIGURE 4.11 (a) An image of size 500 × 500 pixels and (b) its Fourier spectrum. The superimposed circles have radii values of 5, 15, 30, 80, and 230, which enclose 92.0, 94.6, 96.4, 98.0, and 99.5% of the image power, respectively.

Source : Gonzalez and Woods. Digital Image Processing. Prentice-Hall, 2002.

High pass filtering



a b c

FIGURE 4.24 Results of ideal highpass filtering the image in Fig. 4.11(a) with $D_0 = 15$, 30, and 80, respectively. Problems with ringing are quite evident in (a) and (b).

Source : Gonzalez and Woods. Digital Image Processing. Prentice-Hall, 2002.

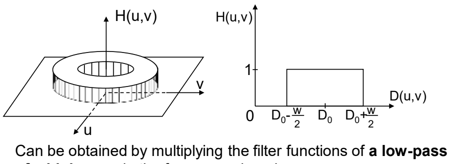
Band-pass filtering

$$H(u,v) = \begin{cases} 0 & D(u,v) \le D_0 - \frac{w}{2} \\ 1 & D_0 - \frac{w}{2} \le D(u,v) \le D_0 + \frac{w}{2} \\ 0 & D(u,v) > D_0 + \frac{w}{2} \end{cases}$$

$D(u,v) = \sqrt{u^2 + v^2}$

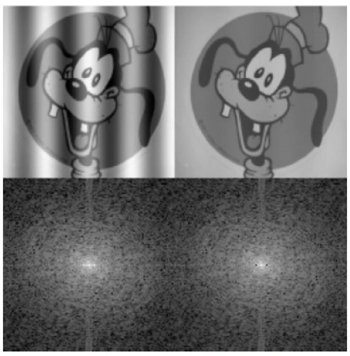
D_0 = cut off frequency

w = band-width



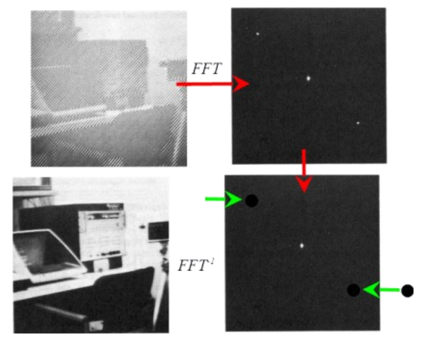
Can be obtained by multiplying the filter functions of **a low-pass and of a high-pass** in the frequency domain

Removing sinus noise

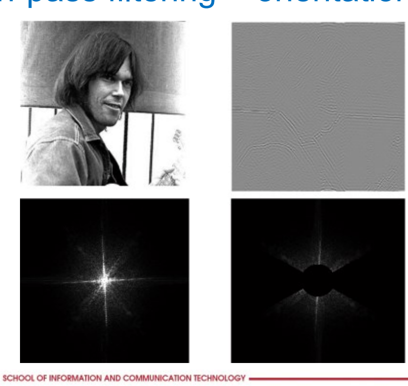


Brayer

Removing sinus noise



High-pass filtering + orientation



Hybrid Images

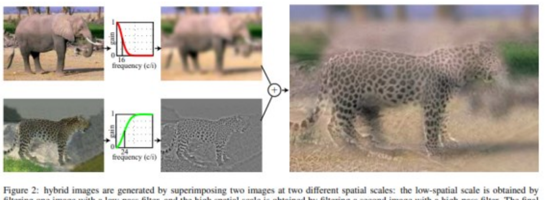
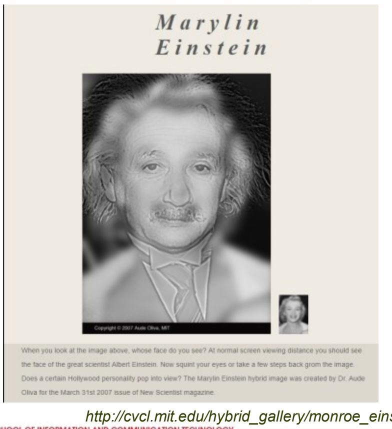


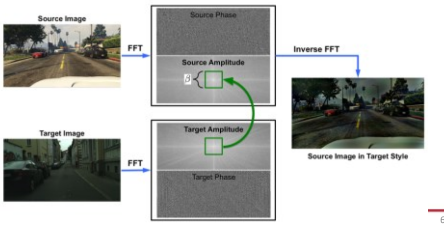
Figure 2: hybrid images are generated by superimposing two images at two different spatial scales: the low-spatial scale is obtained by filtering one image with a low-pass filter, and the high spatial scale is obtained by filtering a second image with a high-pass filter. The final hybrid image is composed by adding these two filtered images.

A. Oliva, A. Torralba, P.G. Schyns, SIGGRAPH 2006



http://cvcl.mit.edu/hybrid_gallery/monroe_einstein.html

Style transfer



Content

- Rappel: digital image representation
- Point Processing
- Convolution and Linear filtering
- More neighborhood operators
- Image transforms
  - Frequency domain
  - PCA (additional reading)
    - PCA
    - Example of using PCA for face recognition

Principle Component Analysis - PCA (Karhunen-Loeve transformation)

- **PCA** transforms the original input space into a lower dimensional space
  - By constructing dimensions that are linear combinations of the given features
- The objective: consider independent dimensions along which data have largest variance (i.e., greatest variability)

Slide by Jana Kosecka

Principal Component Analysis (cont.)

- **PCA** enables transform a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components
- The first principal component accounts for as much of the variability in the data as possible
- Each succeeding component (orthogonal to the previous ones) accounts for as much of the remaining variability as possible

Principal Component Analysis (cont.)

- PCA is the most commonly used dimension reduction technique.
- Data samples $x_1, \cdots, x_N$
- Compute the mean $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$
- Computer the covariance matrix:

$$\Sigma_x = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T$$

Principal Component Analysis (cont.)

- Compute the eigenvalues λ and eigenvectors e of the matrix Σ_x
- Solve $\Sigma_x x = \lambda x$
- Order them by magnitude:

$$\lambda_1 \geq \lambda_2 \geq .\lambda_N.$$

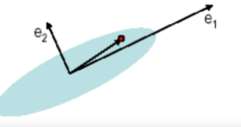
- PCA reduces the dimension by keeping direction e such that $\lambda < T$.

Principal Component Analysis (cont.)

- For many datasets, most of the eigenvalues are negligible and can be discarded.

The eigenvalue λ measures the variation
In the direction of corresponding eigenvector

Example:
$\lambda_1 \neq 0, \lambda_2 = 0.$



Principal Component Analysis (cont.)

- How to get uncorrelated components which Capture most of the variance
- Project the data onto the selected eigenvectors:

$$y_i = e_i^T (x_i - \bar{x})$$

- If we consider first M eigenvectors we get new lower dimensional representation

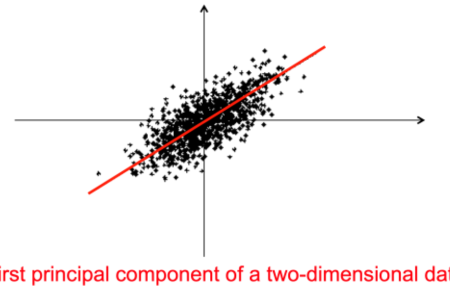
$$[y_1, \cdots, y_M]$$

- Proportion covered by first M eigenvalues

$$\frac{\sum_{i=1}^{M} \lambda_i}{\sum_{i=1}^{N} \lambda_i}$$

Slide by Jana Kosecka

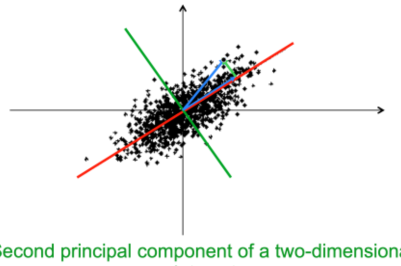
Illustration of PCA



First principal component of a two-dimensional data set.

Slide by Jana Kosecka

Illustration of PCA



Second principal component of a two-dimensional data set.

Slide by Jana Kosecka

Determining the number of components

- Plot the eigenvalues
  - each eigenvalue is related to the amount of variation explained by the corresponding axis (eigenvector)
  - If the points on the graph tend to level out (show an “elbow” shape), these eigenvalues are **usually close enough to zero that they can be ignored**

Slide by Jana Kosecka

Content

- Rappel: digital image representation
- Point Processing
- Convolution and Linear filtering
- More neighborhood operators
- Image transforms
  - Frequency domain
  - PCA (additional reading)
    - PCA
    - Example of using PCA for face recognition

The space of all face images

- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 100x100 image = 10,000 dimensions
- However, relatively few 10,000-dimensional vectors correspond to valid face images
- We want to effectively model the subspace of face images

Slide by Jana Kosecka

Eigenfaces: Key idea

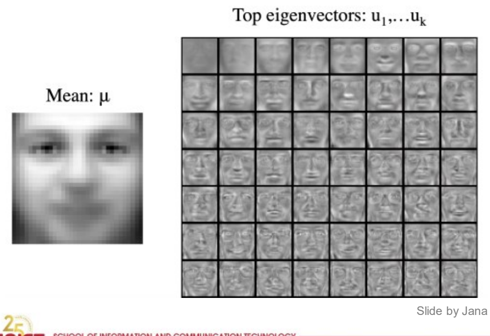
- Assume that most face images lie on a low-dimensional subspace determined by the first *k* (*k*<*d*) directions of maximum variance
- Use PCA to determine the vectors or “eigenfaces” **u**1,...,**u**k that span that subspace
- Represent all face images in the dataset as linear combinations of eigenfaces

Slide by Jana Kosecka

Eigenfaces example- Traning images

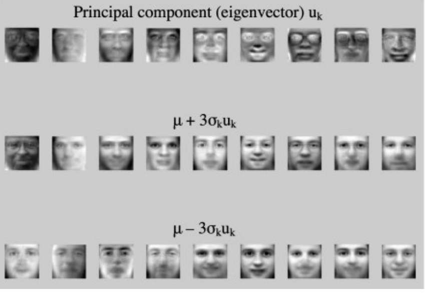
Slide by Jana Kosecka

Eigenfaces example



Slide by Jana Kosecka

Eigenfaces example



Slide by Jana Kosecka

Eigenfaces examples

- Representation

$(w_{i1},\ldots,w_{ik}) = (\mathbf{u}_1^T(\mathbf{x}_i - \boldsymbol{\mu}), \ldots, \mathbf{u}_k^T(\mathbf{x}_i - \boldsymbol{\mu}))$

- Reconstruction

$\hat{x} = \mu + w_1u_1+w_2u_2+w_3u_3+w_4u_4+ \ldots$

Slide by Jana Kosecka

Recognition with eigenfaces

- Process labeled training images:
- Find mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
- Find k principal components (eigenvectors of $\boldsymbol{\Sigma}$) $\mathbf{u}_1,\ldots\mathbf{u}_k$
- Project each training image $\mathbf{x}_i$ onto subspace spanned by principal components:
  $(w_{i1},\ldots,w_{ik}) = (\mathbf{u}_1^T(\mathbf{x}_i - \boldsymbol{\mu}), \ldots, \mathbf{u}_k^T(\mathbf{x}_i - \boldsymbol{\mu}))$

- Given novel image $\mathbf{x}$:
- Project onto subspace:
  $(w_1,\ldots,w_k) = (\mathbf{u}_1^T(\mathbf{x} - \boldsymbol{\mu}), \ldots, \mathbf{u}_k^T(\mathbf{x} - \boldsymbol{\mu}))$
- Optional: check reconstruction error $\mathbf{x} - \hat{\mathbf{x}}$ to determine whether image is really a face
- Classify as closest training face in k-dimensional subspace

Slide by Jana Kosecka


VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY
Thank you for your attention!
soict.hust.edu.vn/
fb.com/groups/soict