Thomas.Grenier@creatis.insa-lyon.fr

Digital Image Processing

Discrete 2D Processing

Département Génie Electrique
5GE - TdSi

Summary

Discrete 2D Processing

- Vector space, colour space
- Operations on images
  - Arithmetic operations
  - Set and logical operations
  - Spatial operations
    - Geometric
    - Convolution
- Image transformations
  - Unitary Transforms

Vector space and Matrix

- Vector and Matrix Operations
  - Vector
    - Spatial position of pixel $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$
    - Pixel intensities $\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$ i.e. : RBG

  ➔ once pixels have been represented as vectors, we can use the tools of vector-matrix theory
  - Euclidean distance $D(\mathbf{a},\mathbf{b}) = \left[(\mathbf{a}-\mathbf{b})^T(\mathbf{a}-\mathbf{b})\right]^{1/2}$
  - Linear transformations $\mathbf{y} = \mathbf{A}(\mathbf{x}-\mathbf{b})$

Vector space and Matrix

- Vector and Matrix Operations
  - Other vector forms
    - Joint Spatial-range domain

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_s \\ \mathbf{x}_r \end{bmatrix}$$ ← location, ← intensities

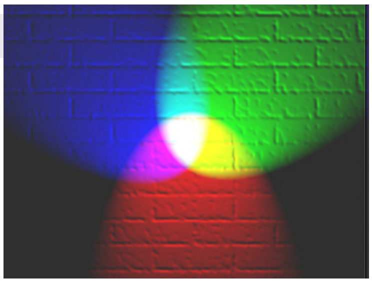
i.e. $$\mathbf{x} = \begin{bmatrix} x \\ y \\ r \\ g \\ b \end{bmatrix}$$

  - Image
    - As function $f(x,y) \Leftrightarrow f(\mathbf{x}_s) = \mathbf{x}_r$
    - As matrix (*MxN*) $\mathbf{I}[i,j] \Leftrightarrow f(i.\Delta x, i.\Delta y) = \mathbf{x}_r$
    - As vector (*MNx1)* $\mathbf{f}[k] = (\mathbf{x}_s, \mathbf{x}_r) \Leftrightarrow \mathbf{I}[k\%M, \lfloor k/N \rfloor] = \mathbf{x}_r$
    - *i.e.* adding noise: $g(x,y) = f(x,y) + \eta(x,y)$

$$\mathbf{G} = \mathbf{F} + \mathbf{N}$$
$$\mathbf{g} = \mathbf{f} + \mathbf{n}$$

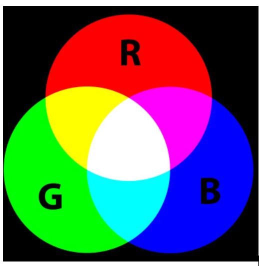
Noiseless images → f, $\mathbf{F}$, $\mathbf{f}$; noise → η, $\mathbf{N}$, $\mathbf{n}$

Color spaces

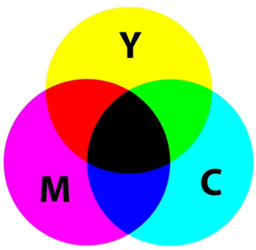


- Color fundamentals
  - Colors are seen as variable combinations of ***primary colors***: Red, Green and Blue
  - Standardization of the specific wavelength values by the CIE (Commission Internationale de l'Eclairage)
    - Red = 700nm, green=546.1nm, blue=435.8nm
  - 2 kinds of mixture
    - Mixture of light (additive primaries)
    - Mixture of pigment (subtractive primaries)

additive



subtractive



Color spaces

- 3 Characteristics used to distinguish one color from another:
  - ***Brightness***: achromatic notion of intensity
  - ***Hue***: dominant color perceived (wavelength)
  - ***Saturation***: relative purity (or the amount of white light within a hue)
- Chromaticity: hue+saturation

➔ A color can be characterized by its brightness and chromaticity

Color spaces

- Tristimulus values:
  - amount of red (X) green (Y) and blue(Z)
- Trichromatic coefficients (not spatial position!)

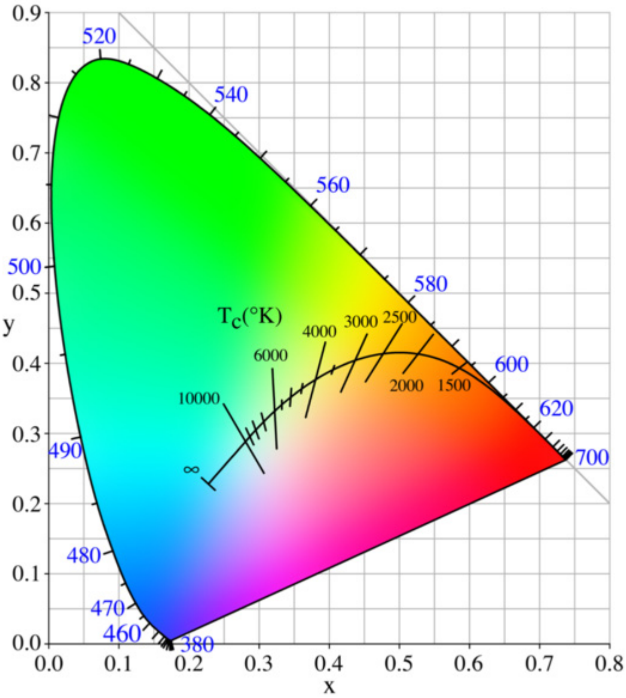
$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad z = \frac{Z}{X+Y+Z}$$

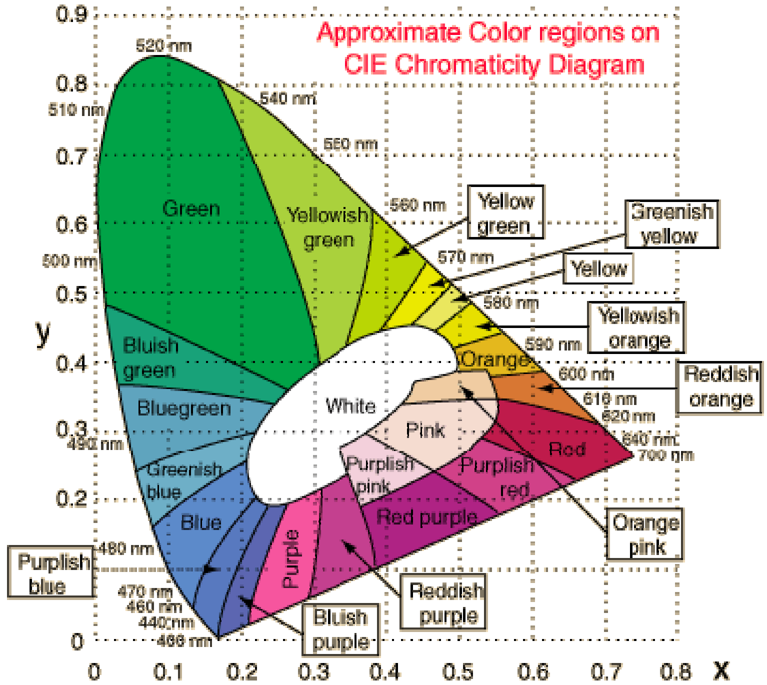
$$\Rightarrow x + y + z = 1$$

- Example:
  - a yellowish green :
    - x = 30% Red; y=60% Green; (z=10% Blue)

Color spaces

- CIE Chromaticity diagram



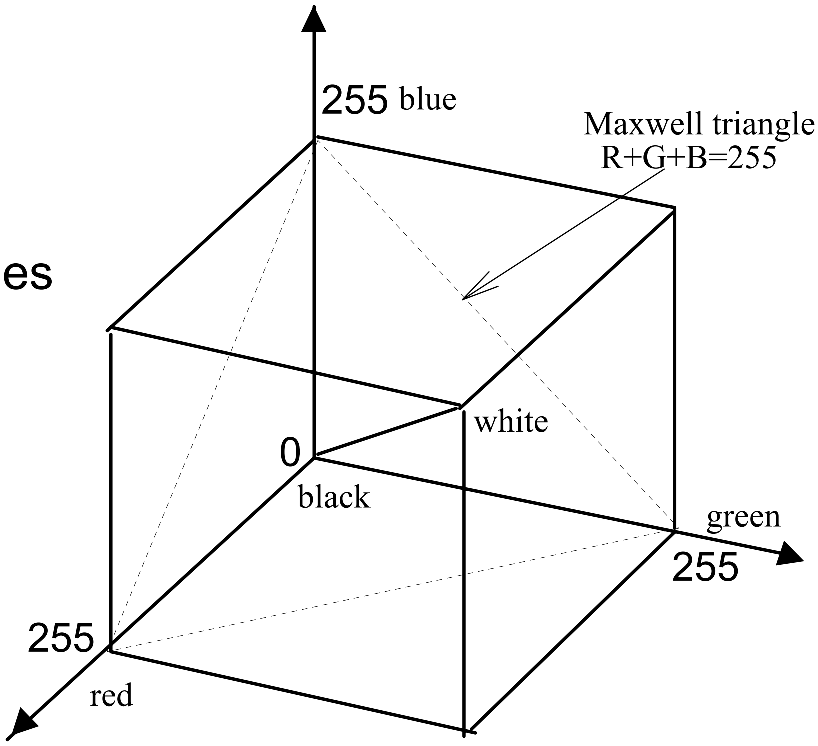


Color spaces

- A color model (also called color space or color systems) is a specification of a coordinate system.
- Color models in use may be
  - Hardware-driven (monitors, printers)
  - Application-driven (creation, color graphic animation, …)
- Commonly used color models
  - RBG (monitors, color video cameras)
  - CMY, CMYK (printing)
  - YUV (TV, MPEG, …)
  - HSI (similar to human description and interpretation of colors)
  - *Pseudocolor and Look Up Tables*

RGB color model

- Additive synthesis of color (mixtures of light)
  - Display hardware
    - monitors
      - CRT, LCD, plasma
    - graphic card
  - 24 bits (3*8 bits) Images
  - About 16M colors
  - Grey-levels images
    - R=G=B



CMY Color model

- Cyan Magenta Yellow
  - Subtractive color model
    - Describes the color reflected by an illuminated surface absorbing certain wavelengths
    - Is needed by devices that deposit colored pigments
      - i.e. on paper
    - CMYK: black color added for true black

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} \rightarrow \begin{bmatrix} C - \min(C,M,Y) \\ M - \min(C,M,Y) \\ Y - \min(C,M,Y) \\ \min(C,M,Y) \end{bmatrix} = \begin{bmatrix} C' \\ M' \\ Y' \\ K \end{bmatrix}$$

YUV Color model

- YUV: PAL; YIQ: NTSC
- Y → intensity (grey-levels!) (not yellow ☺)
- UV or IQ: chromaticity

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & -0.437 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.522 & 0.311 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

➔ YUV is better than RGB for information decorrelation

→ Compression of color images

$$\begin{bmatrix} R:33.2 \\ V:36.2 \\ B:30.6 \end{bmatrix} \leftrightarrow \begin{bmatrix} Y:93 \\ U:5.3 \\ V:1.7 \end{bmatrix}$$

HSI Color model

- Hue Saturation and …
  - Intensity HSI, Brightness HSB, Lightness HSL, Value HSV (not exactly the same)
- This model attempts to describe perceptual color relationships more accurately than RGB

RGB→HSI

$$I = \frac{1}{3}(R + B + G)$$

$$S = 1 - \frac{3.\min(R, G, B)}{R + G + B}$$

$$H = \begin{cases} \theta & if \; B \le G \\ 360 - \theta & if \; B > G \end{cases}$$

$$\theta = \cos^{-1}\left(\frac{\frac{1}{2}\left((R-G)+(R-B)\right)}{\left((R-G)^2 + (R-B)(G-B)\right)^{\frac{1}{2}}} \right)$$

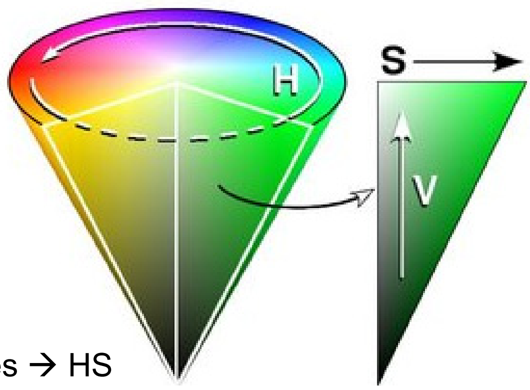
YUV→HSL

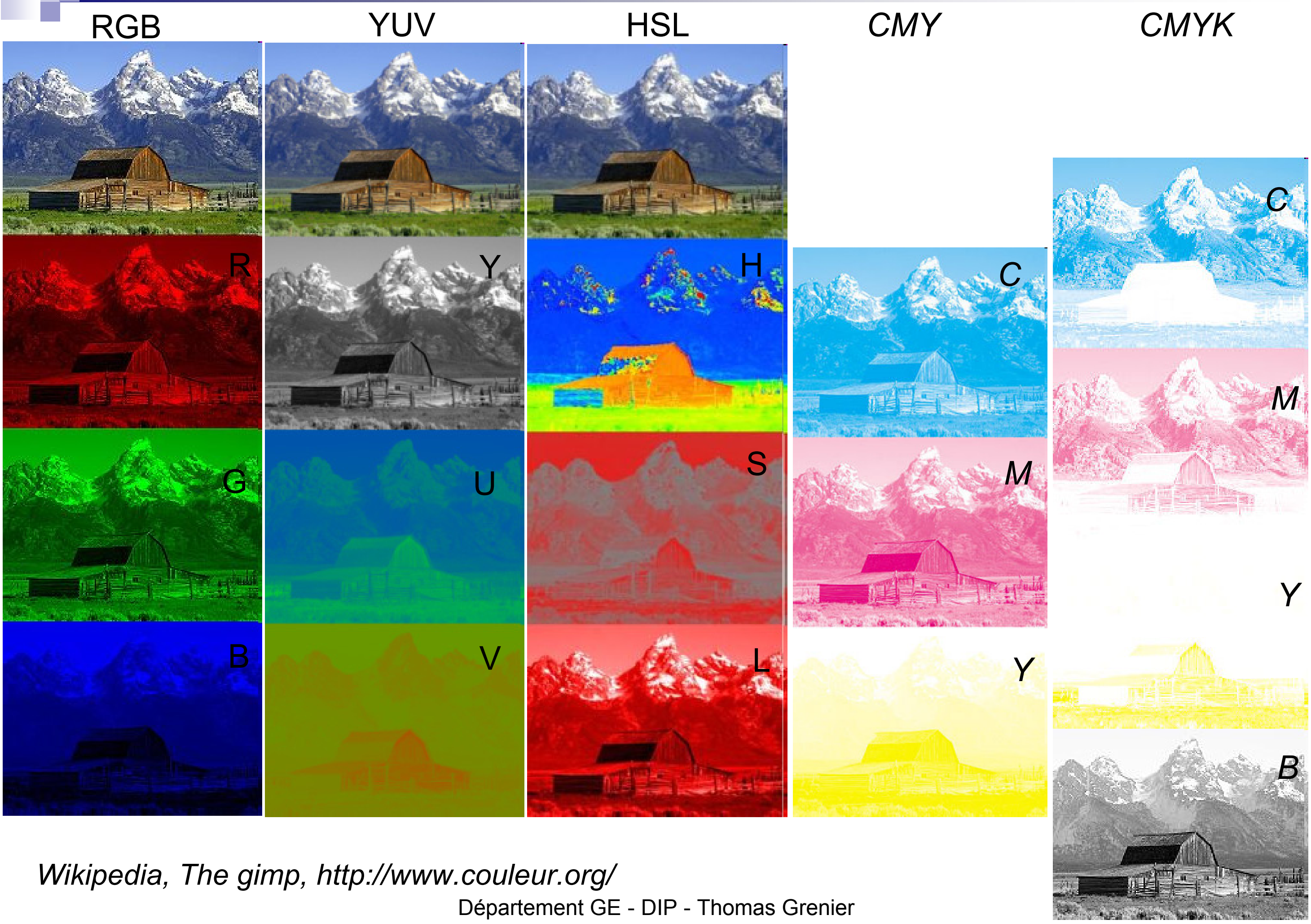
$$L = Y$$

$$H_{UV} = \tan^{-1}(V/U)$$

$$S_{UV} = \sqrt{U^2 + V^2}$$

UV→ polar coordinates → HS

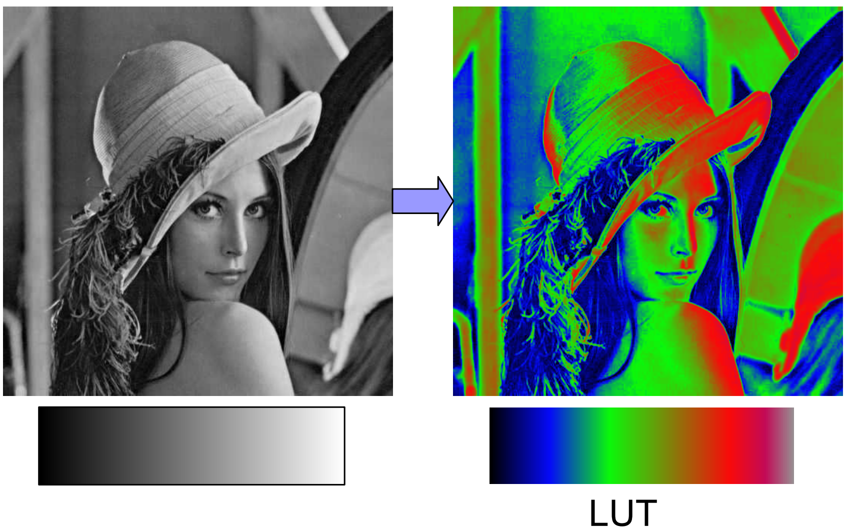


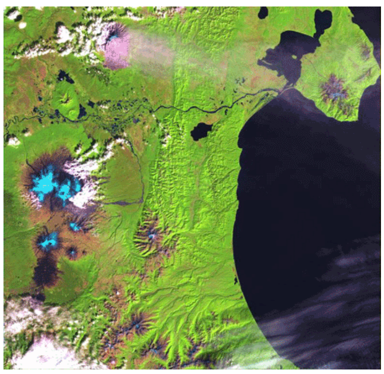


Wikipedia, The gimp, http://www.couleur.org/

Pseudocolor

- Or false color visualization consists of assigning colors to gray values based on specified criteria, function(s), table(s) (LUT)





Example of a 3-color image with mixed components

→ Image compression

Operations on images

- Arithmetic Operations
  - $+, -, \times, \div$
  - Application: Corrupted image *g* obtained by adding the noise *η* to a noiseless image *f*

$$g(x,y) = f(x,y) + \eta(x,y)$$

    - assumptions :
      - at every pair of coordinates (x,y) the noise is uncorrelated
      - the noise has zero average value

➔ Noise reduction by summing (averaging) a set of noisy images

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}} . \sigma_{\eta(x,y)}$$

- Noise Reduction

Original

$\sigma = 3{,}55$

Mean

1 $\sigma = 25{,}45$

2 $\sigma = 18{,}3$

4 $\sigma = 13{,}5$

6 $\sigma = 11{,}6$

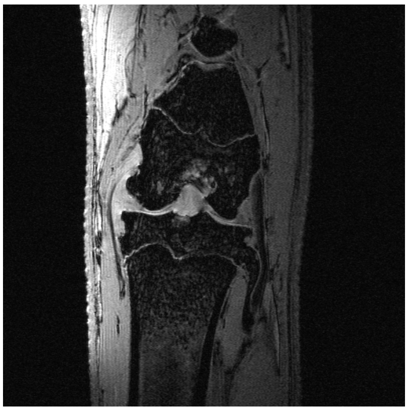
Shading correction examples

*, /

=

Shading pattern

Shading correction examples



Knee of Guinea pig
MRI, 7 Telsa

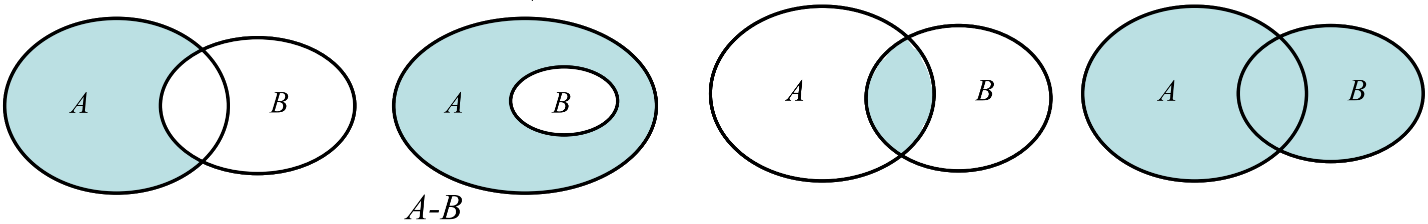
Ultrasound image

And:
- perspective
- non uniform lighting (bulb filament, spot light)

Operations on images

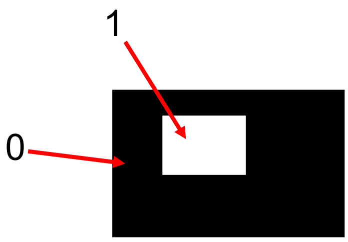
- Set and Logical Operations
  - Basic set operations
    - *a* is an element of the set A: $a \in A, b \notin A$
    - A set is represented with two braces $\{\bullet\}$
    - The set with no elements is called null or empty set: $\varnothing$
    - Some operations

difference $A - B$ | *complement* $B^c = \{w \mid w \notin B\}$ | *intersection* $A \cap B$ | *union* $A \cup B$



Operations on images

- Set and Logical Operations
  - Logical operations
    - Binary image:
      - 1→ foreground
      - 0→ background
    - Operations:
      - AND, OR, NOT, XOR, AND-NOT, …
      - Applicable on binary images or gray-level images!



  - Fuzzy sets
    - For gradual transition from 0 to 1 (and 1 to 0)

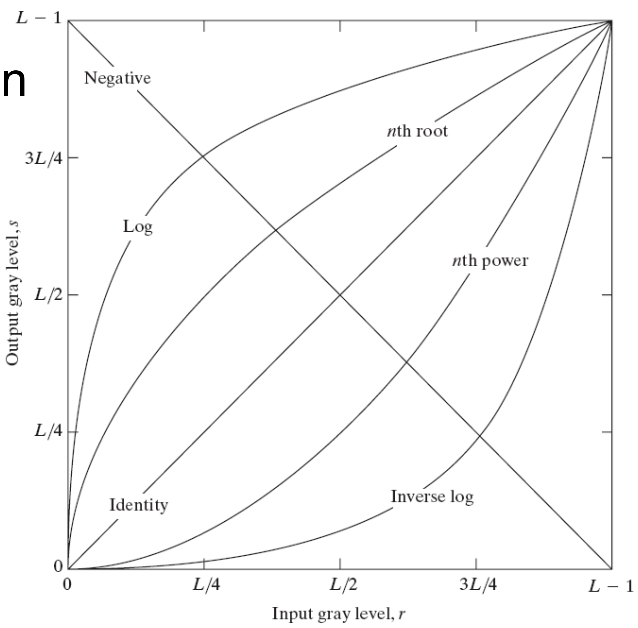
Operations on images

- Spatial Operations
  - Spatial operations are performed directly on the pixels of a given image.
  - 3 categories:
    - Single pixel operations
    - Neighborhood operations, Convolution
    - Geometric spatial transformations and image registration

Operations on images

- Spatial Operations - Single pixel operations
    - r: original intensity,
    - s: new intensity,
    - T: a transformation function

$$s = T(r)$$

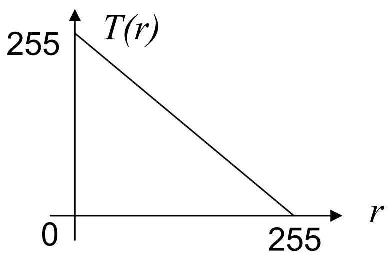


Some basic grey-levels transformation functions used for image enhancement

Range of intensities: *[0,L-1]*

Operations on images

- Spatial Operations - Single pixel operations
  Example



Original

Negative

Operations on images

- Spatial Operations - Neighborhood operations
  - S_{xy}: set of coordinates of a neighborhood centered on a point (x,y) in an image *f*.
  - Neighborhood processing generates one corresponding pixel in the output image *g* at the same (x,y) coordinates.
  - The value of that pixel in g is determined by a operation involving the pixels in S_{xy}.

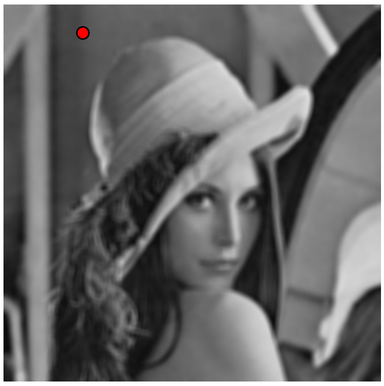
Operations on images

- Spatial Operations - Neighborhood operations

S_{xy}

S_{xy}

Mean of S_{xy}



$$g(x,y) = \frac{1}{m.n} \sum_{(r,c)\in S_{xy}} f(r,c)$$

-1,	-1,	-1,	-1,	-1
-1,	-1,	-1,	-1,	-1
-1,	-1,	**24**,	-1,	-1
-1,	-1,	-1,	-1,	-1
-1,	-1,	-1,	-1,	-1

FIR filters…

Operations on images

- Convolution

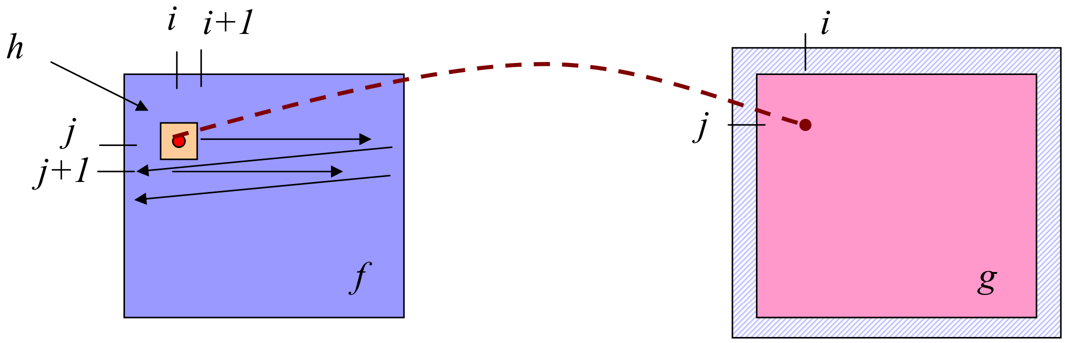
g(x,y) = h(x,y)*f(x,y) (two-dimensional convolution)

$$g(i,j) = \sum\sum_{(k,l)\in H} h(k,l)\, f(i-k, j-l)$$

Output image

Convolution mask
Impulse response

Input image



Example of *Mean Filter*

$$h(k,l) = \begin{cases} \dfrac{1}{(2M+1)(2N+1)} & if\ -M < k < M and -N < l < N \\ 0 & else \end{cases}$$

W: 2x2 neighborhood

- $k = \{0;1\}$ $l = \{0;1\}$

h(k,l) = 1 /4 for each (*k,l*)

l \ k	0	1
0	1/4	1/4
1	1/4	1/4

0	1	2	2
1	1	2	1
1	2	0	0

→

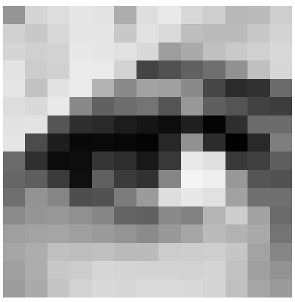
3/4	6/4	7/4	x
5/4	5/4	3/4	x
x	x	x	x

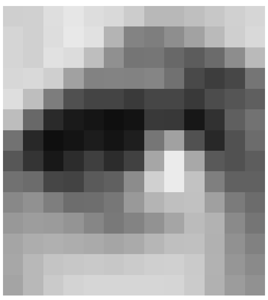
→ (rounded to the integer part)

0	1	1	x
1	1	0	x
x	x	x	x

Example of Mean Filter

2x2
Mean Filter



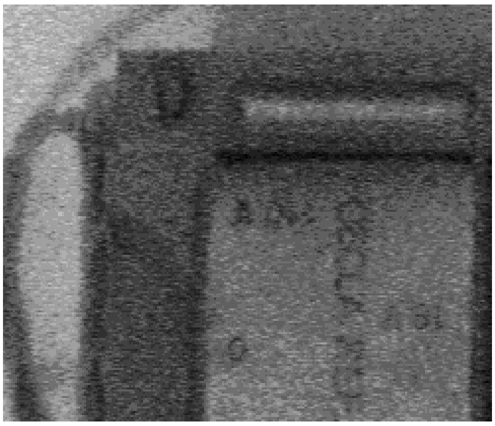


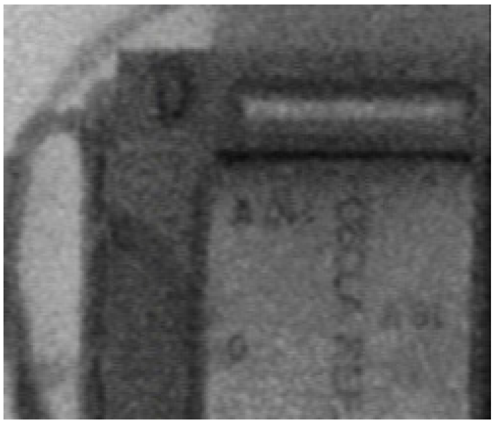
(zoom)

(… interpolation)

Example Mean Filter → noise reduction

Mean Filter 3x3 (k=-1,0,1 l=-1,0,1),
Constant value $h(k,l)$=1/9

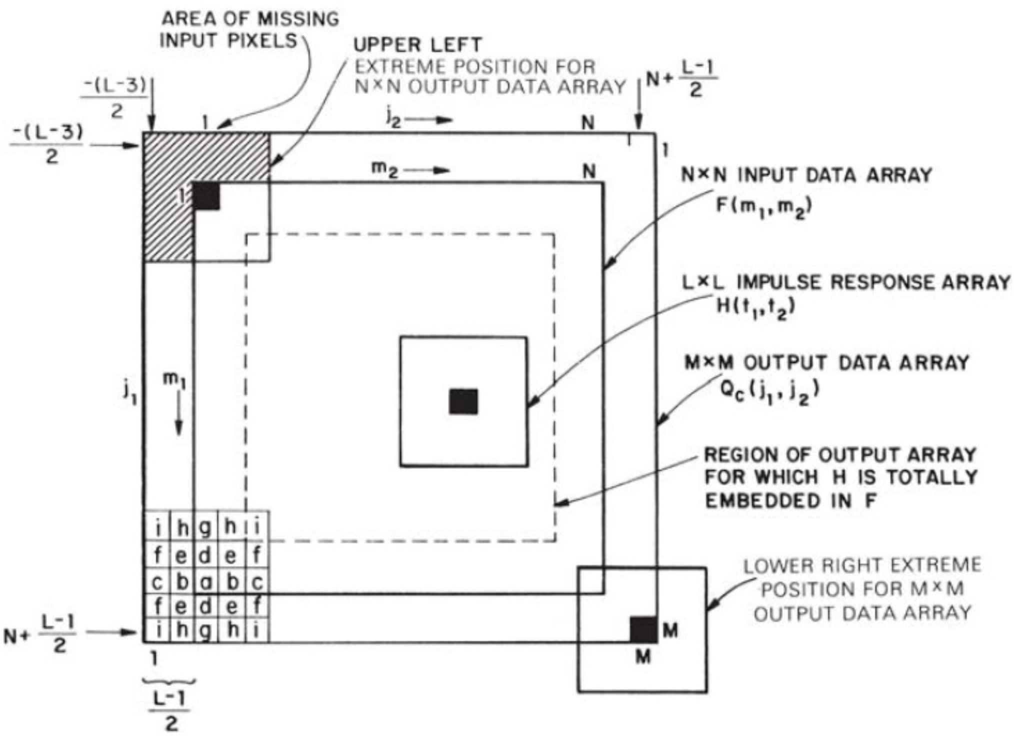




→ Problems?

Operations on images

Domain of convolution



Operations on images

- Geometric spatial transformations and image registration
  - Geometric transformations modify the spatial relationship between pixels in an image
  - In terms of digital image processing, a geometric transformation consists of
    - A **spatial transformation** of coordinates $(x',y') = T\{(x,y)\}$
    - **Intensity interpolation** that assigns values to the spatially transformed pixel

- Spatial transformations
  - Example
    - Shrink image to half its size $(x',y') = T\{(x,y)\} = (x/2, y/2)$
  - **Affine transform**:

$$[x', y', 1] = [x, y, 1].\begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

  - Higher order

$$[x', y', 1] = [x, y, x^2, y^2, xy, \ldots, 1].\mathbf{T}$$

…

Affine transform:

identity
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = x \\ y' = y \end{cases}$$

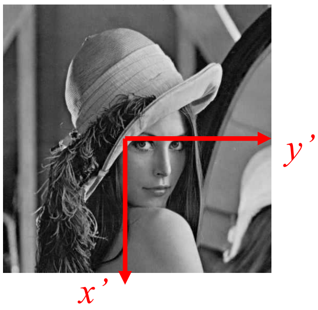
scaling
$$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = c_x.x \\ y' = c_y.y \end{cases}$$

translation
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = x + t_x \\ y' = y + t_y \end{cases}$$

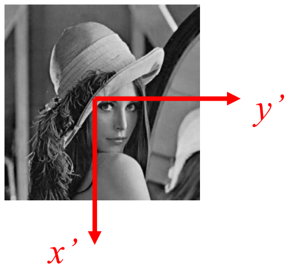
Shear (vertical)
$$\begin{bmatrix} 1 & 0 & 0 \\ s_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = x + s_x.y \\ y' = y \end{cases}$$

Shear (horizontal)
$$\begin{bmatrix} 1 & s_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = x \\ y' = y + s_y.x \end{cases}$$

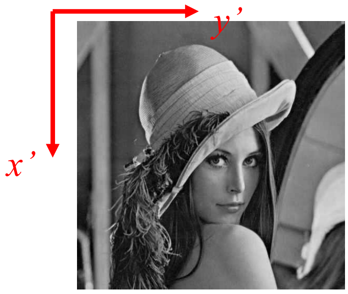
identity



scaling

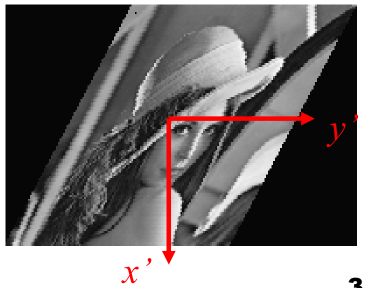


translation



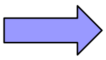
horizontal shearing

$x' = x$
$y' = y + 0.5\,x$



Affine transform, rotation

Rotation
$$T = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

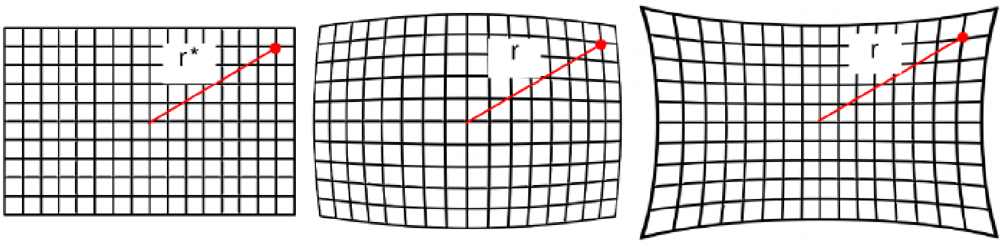


35° degrees rotation

Higher order transforms

Applications :

Lens distortion correction, perspective

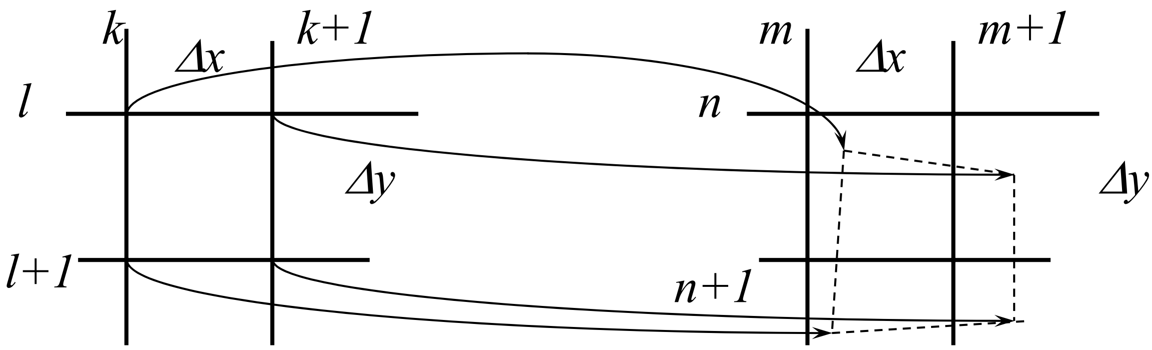


Intensity interpolation

Problem:

x,y are discrete values (sampled image): x=*kDx* , *y=lDy*

and *x'=h1(kDx , lDy)* et *y'=h2(kDx , lDy) are not* necessary multiple integer of *Dx* and *Dy*



Solution: intensity interpolation

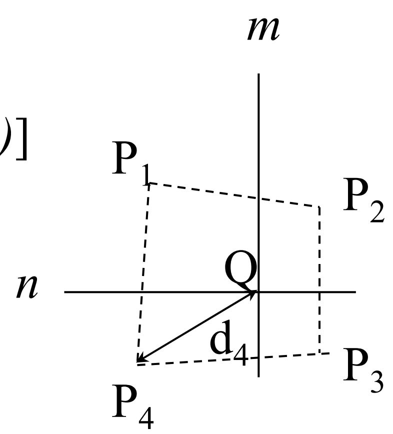
$f'(Q)=f'(m\Delta x,n\Delta y) = G[f(P_1),f(P_2),f(P_3),f(P_4)]$

With $f(P_1)=f(k\Delta x, l\Delta y)$

$f(P_2)=f((k+1)\Delta x,l\Delta y)$

$f(P_3)=f((k+1)\Delta x,(l+1)\Delta y)$

$f(P_4)=f(k\Delta x, (l+1)\Delta y)$



- Nearest neighbor: $f(Q)=f(P_k)$, k : $d_k=\min\{d_1,d_2,d_3,d_4\}$
- linear interpolation

$$f(Q)=\frac{\sum_{k=1}^{4} f(P_k)/d_k}{\sum_{k=1}^{4} 1/d_k}$$

•Bilinear interpolation, ideal interpolation, spline interpolation,....

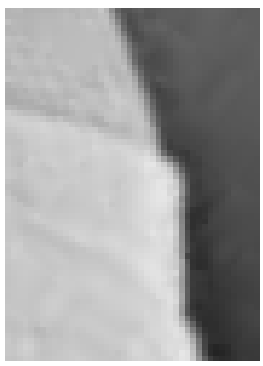
Interpolation, example with rotation

$$T=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$$

35° degrees rotation

Without interpolation

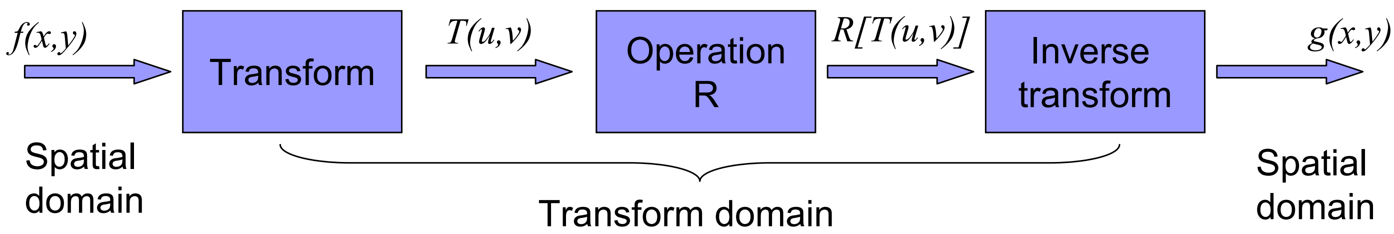
With interpolation



Images Transformation

- Previous methods work in spatial domain
- In some cases, image processing tasks are best formulated in a transform domain.
  - i.e. frequency → Fourier

→ Many transformations exist



Images Transformation

- A particularly important class of 2D linear transforms can be expressed in the general form

$$T(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y).r(x,y;u,v)$$

forward transform — Input image — Forward transformation kernel

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u,v).s(x,y;u,v)$$

Recovered image — Forward transform — inverse transformation kernel

Separable kernel: $r(x,y,u,v) = r_1(x,u).r_2(y,v)$

Symmetric kernel: $r(x,y,u,v) = r_1(x,u).r_1(y,v)$

Images Transformation, DFT

- 2-D Discrete Fourier Transform (**DFT**)

Forward kernel $r(x, y, u, v) = e^{-j2\pi(ux/M + vy/N)}$

Inverse kernel $s(x, y, u, v) = \frac{1}{MN} e^{j2\pi(ux/M + vy/N)}$

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y).e^{-2j\pi(ux/M + vy/N)}$$

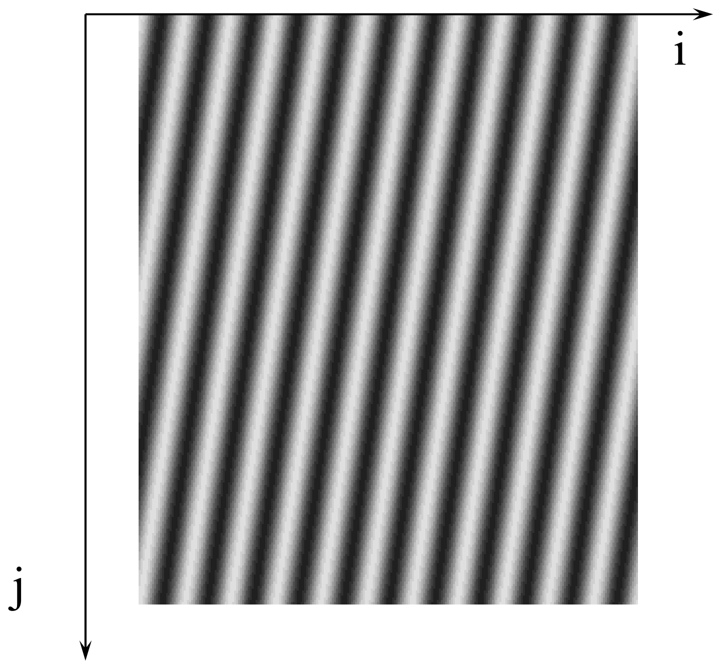
$$f(x, y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} T(u, v).e^{2j\pi(ux/M + vy/N)}$$

➔Complex numbers…

➔Modulus and phase (angle)

→ coefficients…

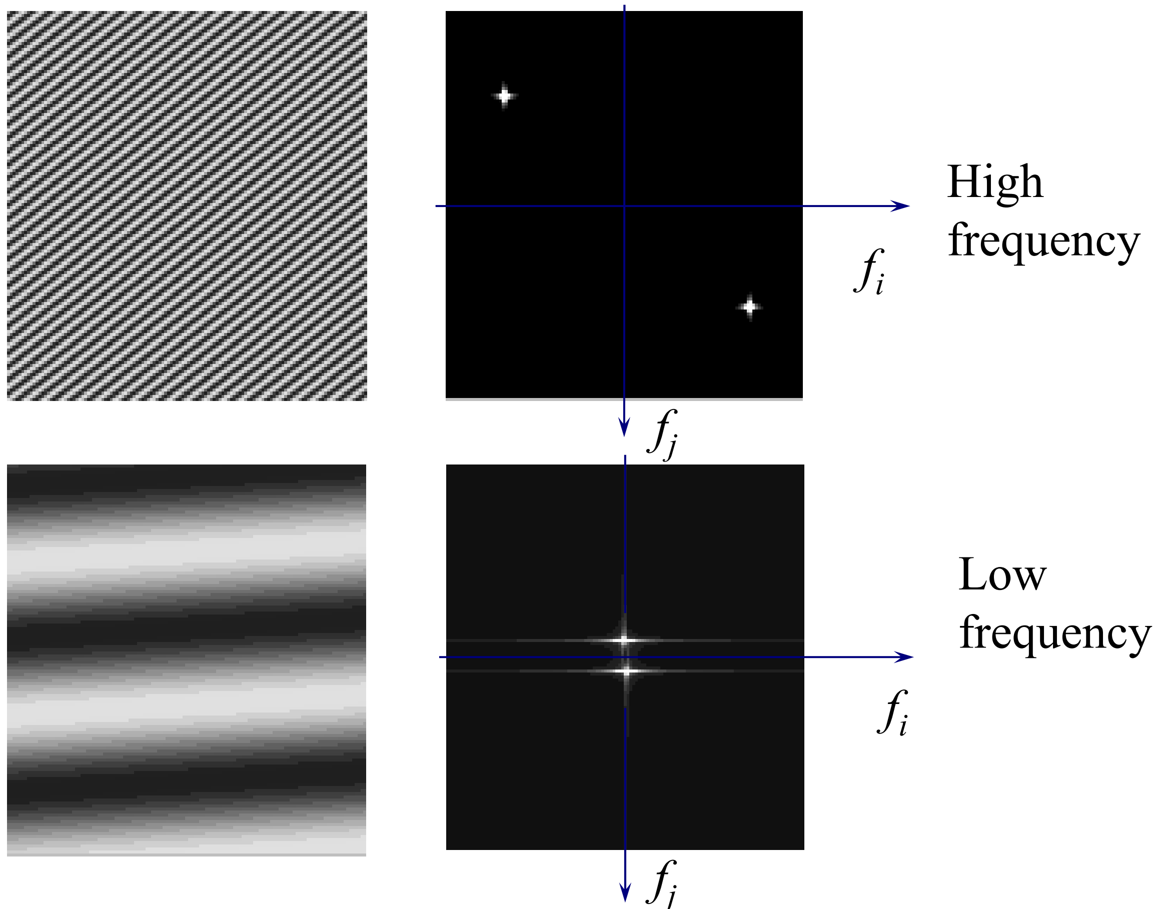
- Example: 2-D Fourier transform



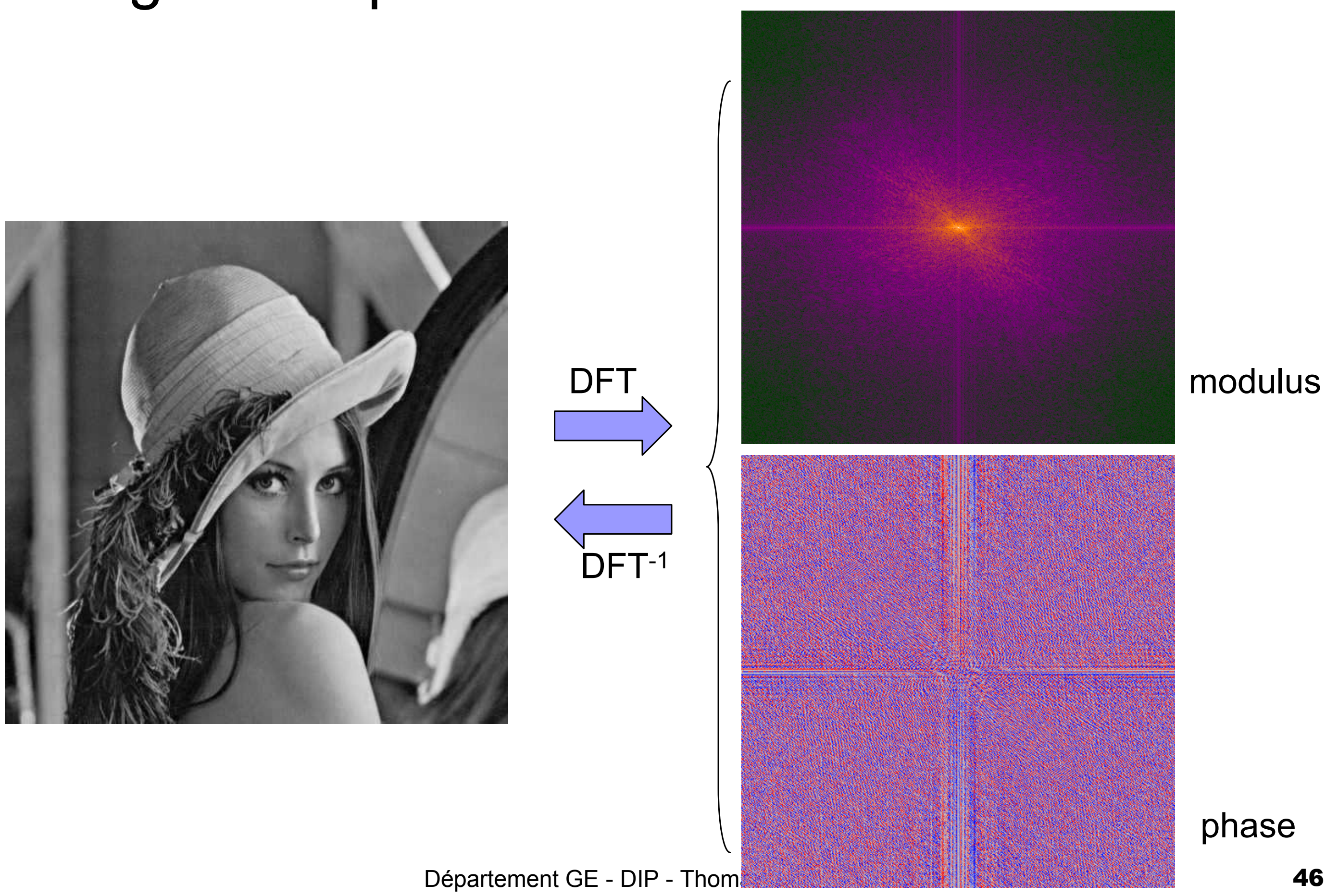
Speed (frequency) of sinusoïdal variation of pixel intensity in a given direction

$$f(i, j) = A \sin(2\pi f_i i + 2\pi f_j j + \varphi_i + \varphi_j)$$

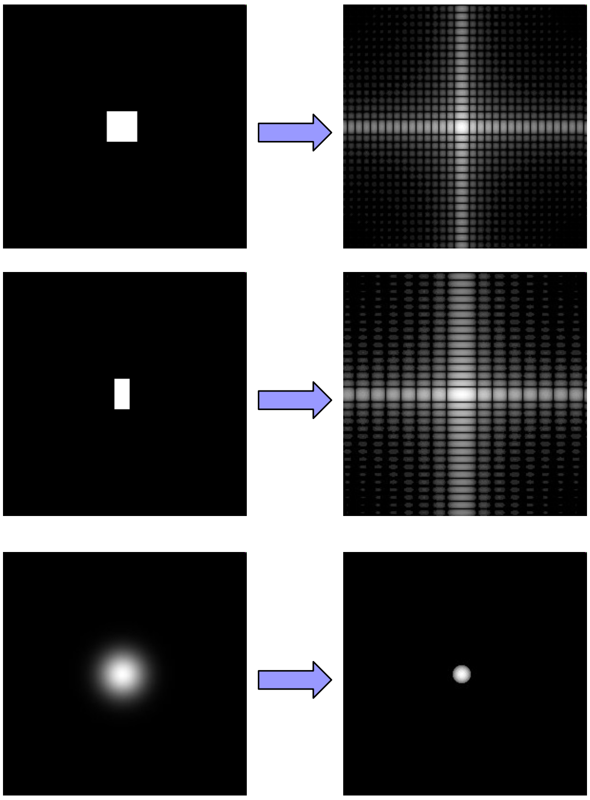
Sine images → 2 Dirac delta functions



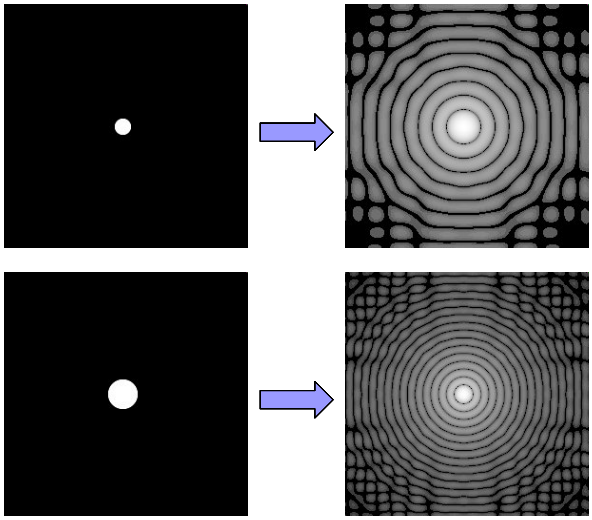
- Image example: 2-D Fourier transform

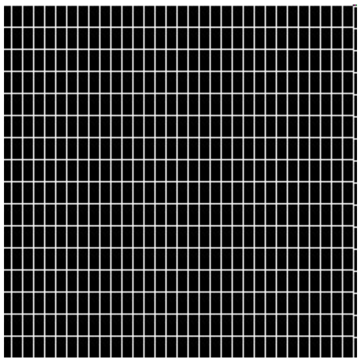


Some DFTs

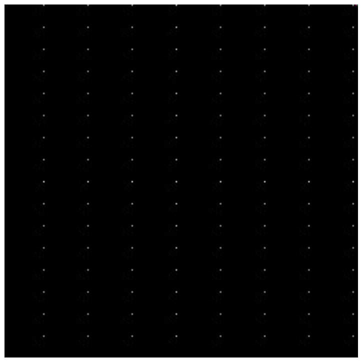


gauss





grid

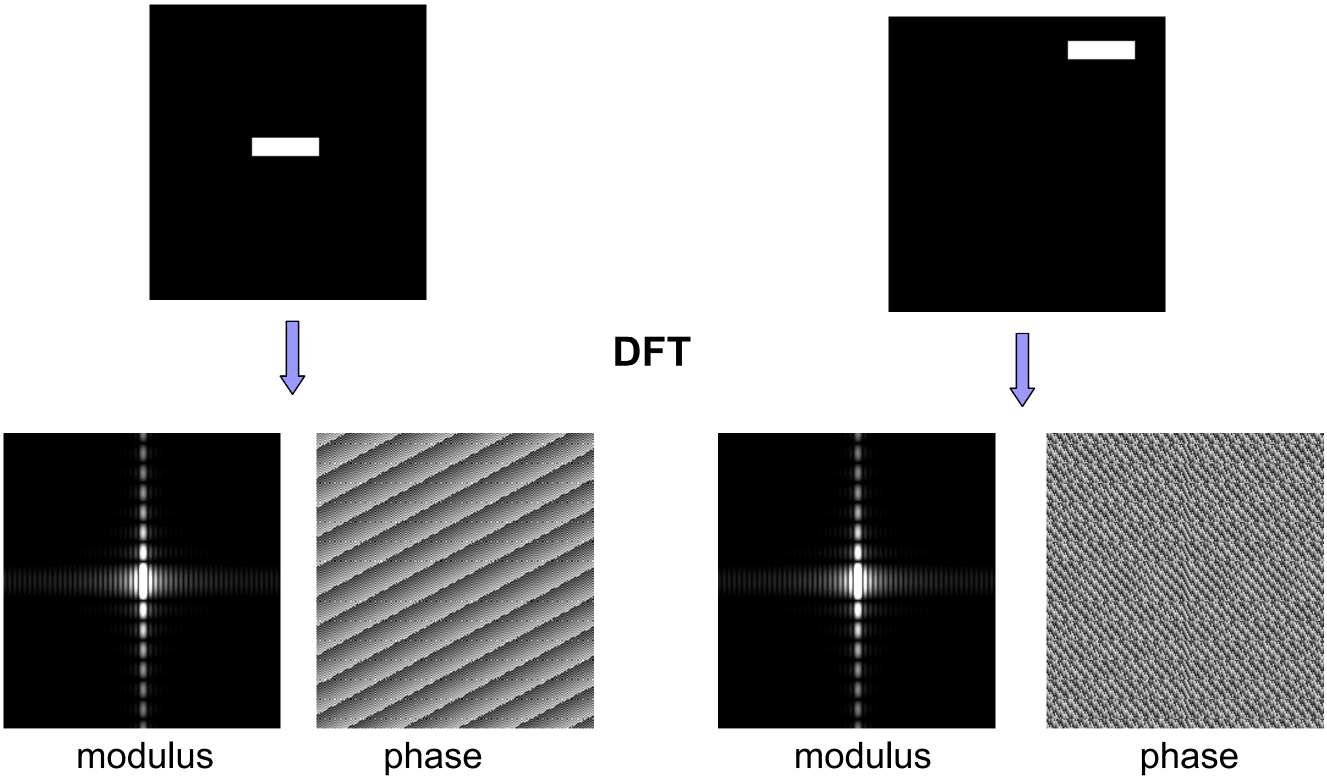


Weighted 2D Dirac comb

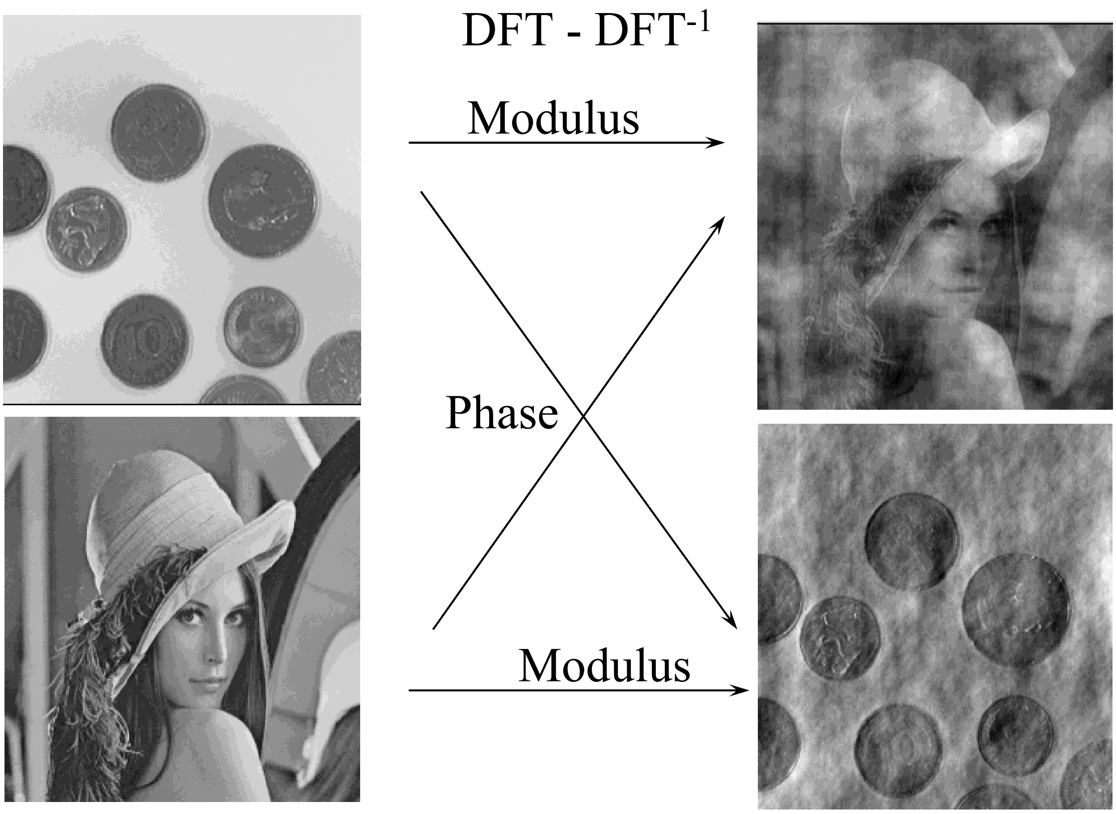
2-D DFT and DFT^{-1} properties

- The same as in the 1D case
- Periodic in the *u* and *v* directions (period = M,N)
- $F(0,0)$ = dc component = mean of grey-levels
- Energy conservation → $\Sigma\Sigma\ |f(x,y)|^2 = \Sigma\Sigma\ |F(u,v)|^2$
- $f(x,y)$ real ➢ $F(u,v)$ is conjugate symmetric $F^*(u,v) = F(-u,-v)$
  - (and Real part is even, Imaginary part is odd)
- Separable and symmetric kernel
- Fast algorithm (FFT, many forms) : $N^2.\log_2(N)$
- Circular convolution (periodic extension of function) = DFT

Phase influence:

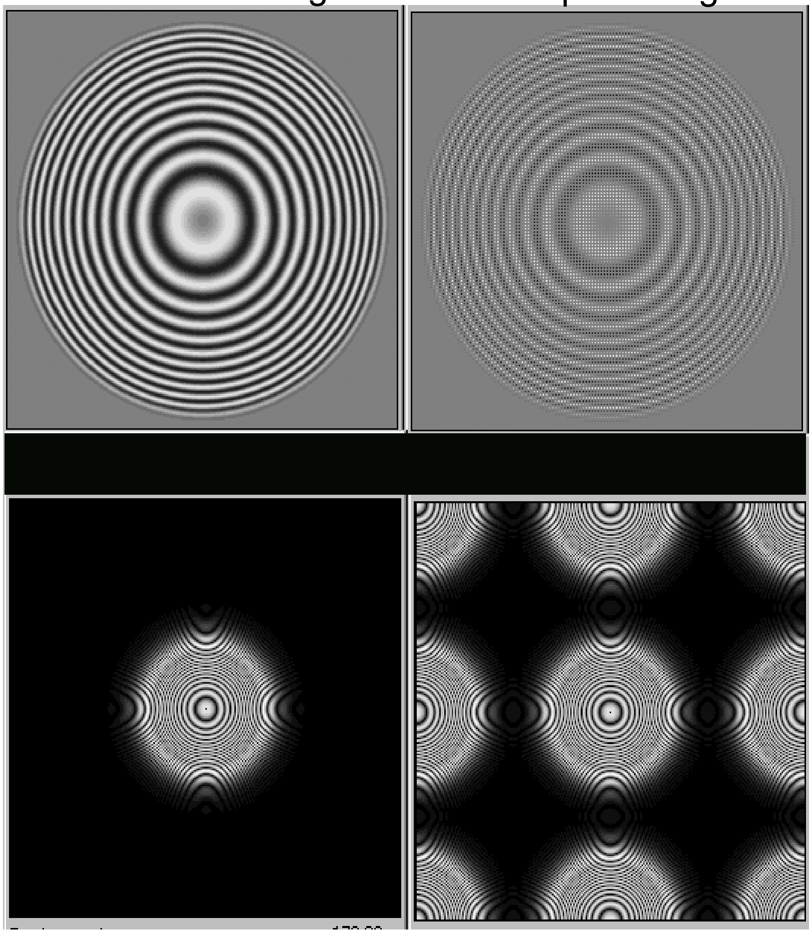


Phase influence:



□ Notes on aliasing…

Continuous image | Sampled image



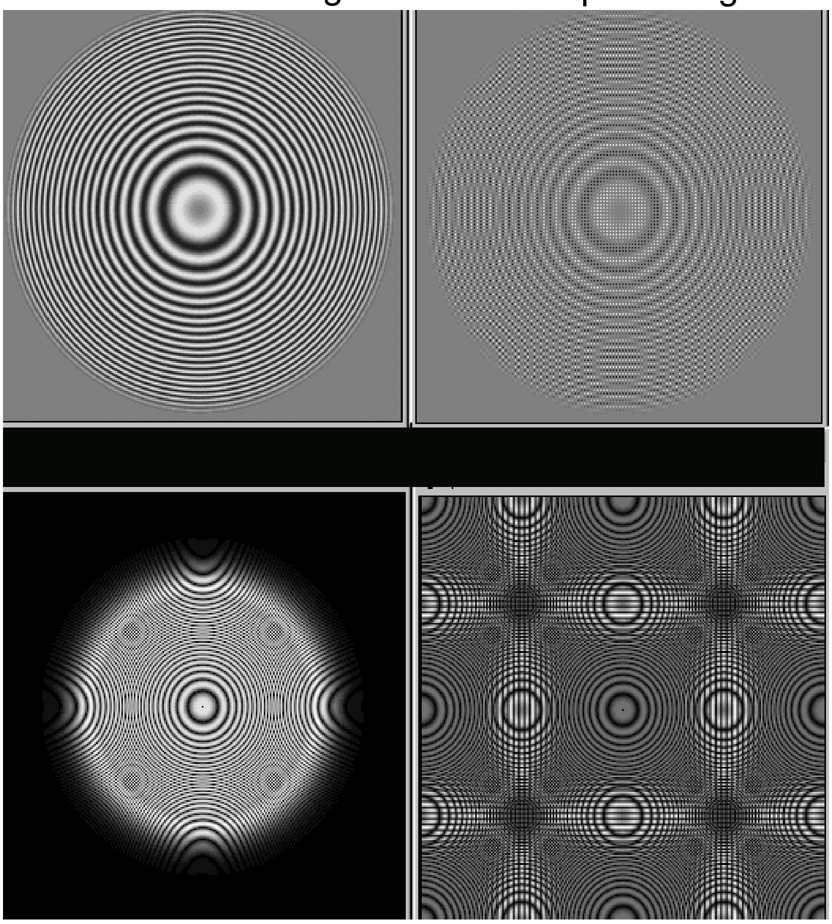
Without aliasing

Note

Periodic DFT

□ Notes on aliasing…

Continuous image | Sampled image



With aliasing

Images Transformation

- Other transformations
  - Unitary transforms
    - Radon
      - used to reconstruct images from medical computed tomography scans
    - **Cosine (DCT)**
      - JPEG, MPEG (mDCT: AAC, Vorbis, WMA, MP3)
    - Sine
    - Wavelet
    - …
  - **Hough**

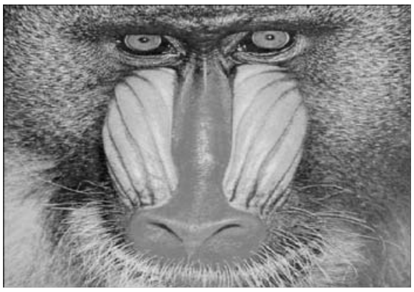
- Discrete Cosine Transform

DCT-II

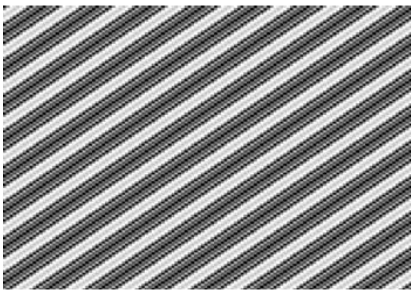
$$C(u,v)=\frac{4.c(u).c(v)}{M.N}.\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}x(i,j).\cos\left(\frac{(2i+1)u.\pi}{2N}\right).\cos\left(\frac{(2j+1)v.\pi}{2M}\right)$$

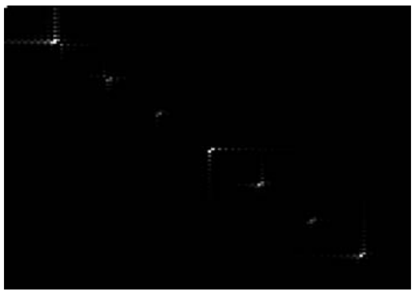
$$\text{with}\, c(m)=\begin{cases}\sqrt{2/N} & \text{if } m\neq 0\\ \sqrt{1/N} & \text{if } m=0\end{cases}$$

Examples



Sine wave



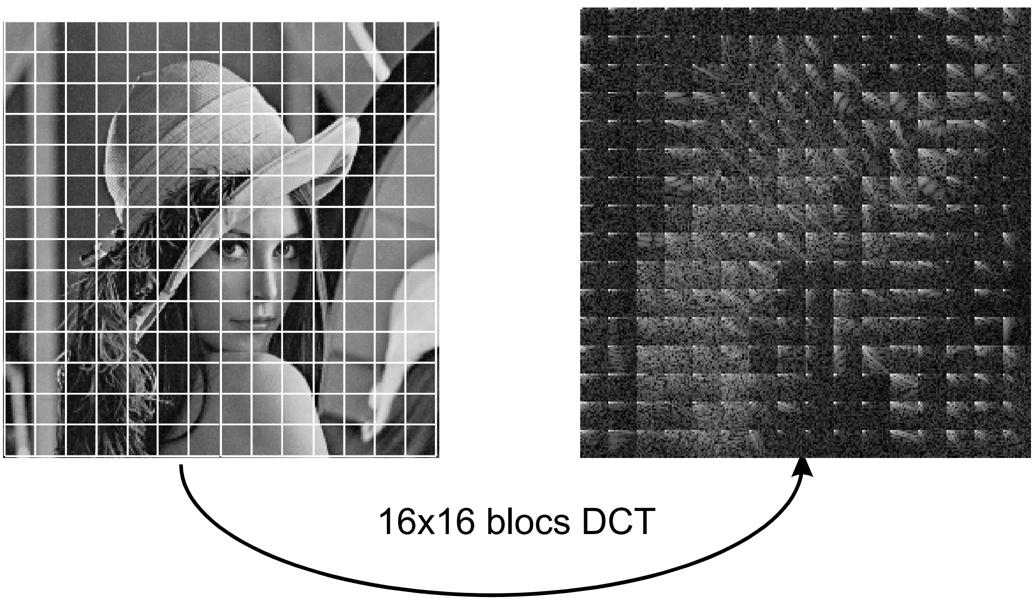


Discrete Cosine Transform, properties

- Linear, separable
- **real coefficients**
- *C(0,0)* = dc component = mean of grey-levels
- DCT concentrates most of the power on the lower frequencies
- Fast algorithms (like FFT) : $N^2.\log_2 (N)$

➔ Image compression (JPEG, MPEG)

Note on jpeg: DCT per bloc



Hough Transform

➔ Features extraction technique

- The classical Hough transform was concerned with the identification of lines in the image
- The Hough transform has been extended to identify positions of arbitrary shapes
  - Circles
  - Ellipses

Hough Transform

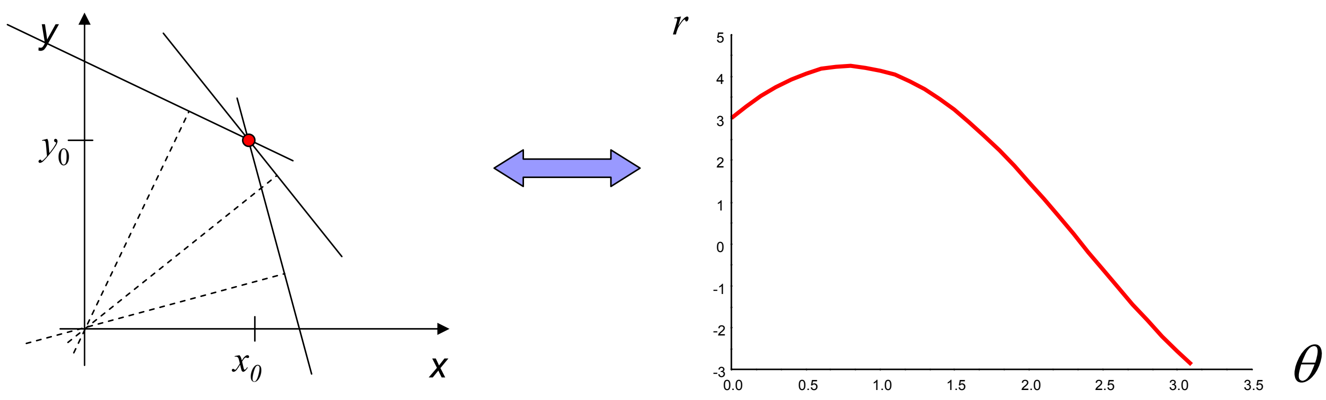
Equation of a line

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right) \longrightarrow r = x.\cos\theta + y.\sin\theta$$

Distance to origin

Angle of the vector from the origin to the closest point of the line

➔For a point with (x_0,y_0) coordinates in the image plane, all the lines that go through it verify : $r(\theta) = x_0.\cos\theta + y_0.\sin\theta$

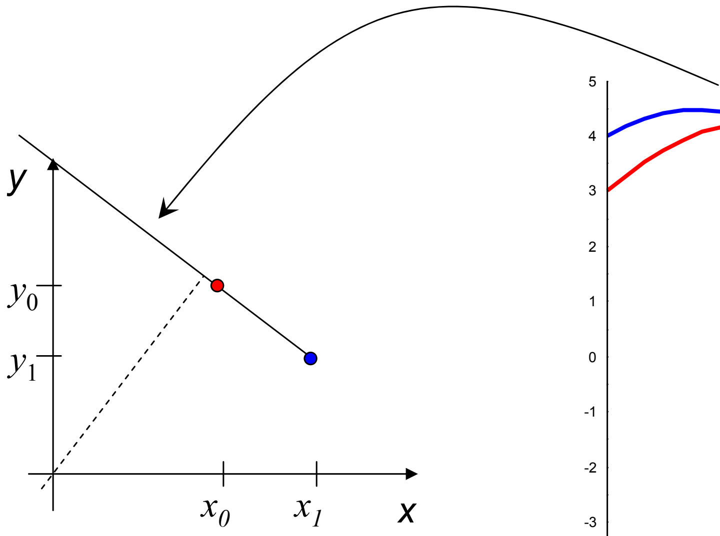


3 lines through (x_0,y_0)

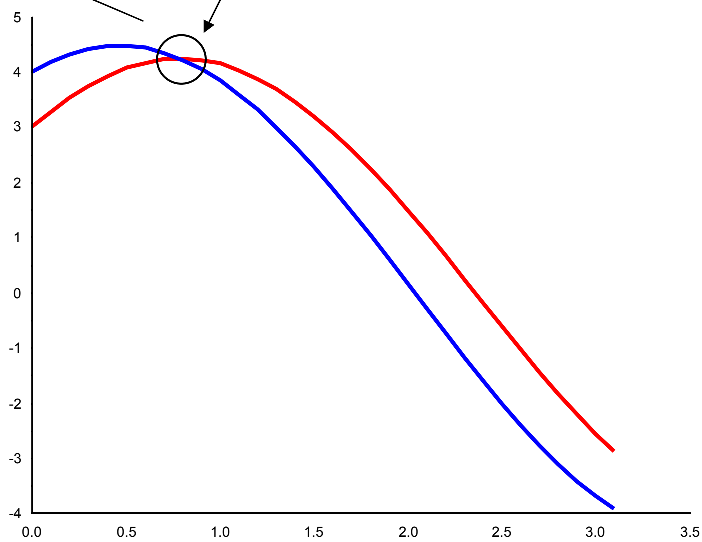
All the lines through (x_0,y_0)

Hough Transform

One point in the Hough space
➔ One line in spatial space



2 points (x_0, y_0), (x_1, y_1)
➔ One line !



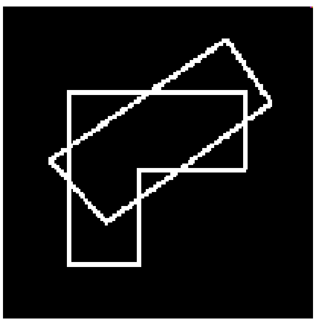
Hough space of the 2 points (x_0, y_0), (x_1, y_1)

→ $(x_0, y_0) = (3,3)$; $(x_1, y_1) = (4,2)$;

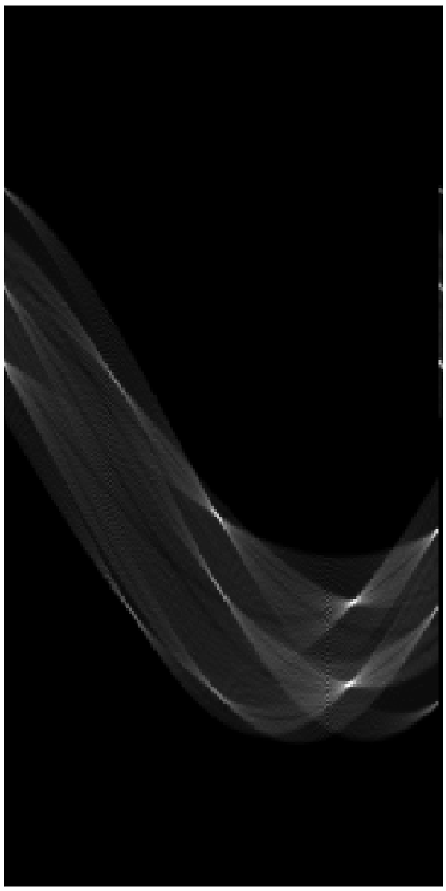
→ $(r, \theta) \sim (4.25, 0.785)$

Hough Transform

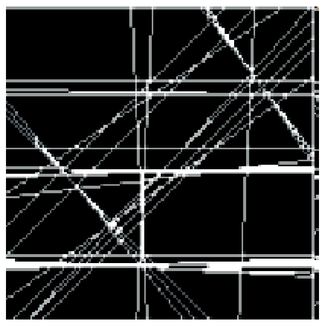
- Application: line detection



Binary image



Hough's space



Projection of some lines

→ Implementation ?
→ Sinogram and Radon