Digital Image Processing Lecture 4

- Geometrical transformations with the affine transform.
- Different types of interpolation:
  nearest neighbor, bilinear, bicubic and sinc.
- Up-sampling, down-sampling and rotation.
- Scaling pyramids.
- Gonzales & Woods:
  - Chapter 2: Image interpolation (4 pages)
    Geometric transformation (5 pages)
  - Chapter 4: Aliasing in images (7 pages)
  - Numbers according to Gonzales & Woods, Global Edition, 4th edition. (Numbers in other editions may vary).
  - Gonzales & Woods lacks rather much of the contents of this lecture.

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

Geometrical transformations with the affine transform

- Points on a straight line are transformed to points on a straight line
- Parallel lines are transformed to parallel lines
- The relation between the area of an object and its transform is constant.

Ex) The parallel projection is an affine transform, but the perspective projection is not.

$$(x \quad y \quad 1) = (v \quad w \quad 1)\mathbf{T} = (v \quad w \quad 1)\begin{pmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{pmatrix}$$

Alternative description:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} t_{11} & t_{21} \\ t_{12} & t_{22} \end{pmatrix}\begin{pmatrix} v \\ w \end{pmatrix} + \begin{pmatrix} t_{31} \\ t_{32} \end{pmatrix}$$

Lab 2

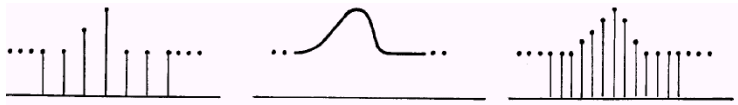
Coordinate in the out-image | 2x2-matrix | Coordinate in the in-image | Translation vector

Affine trans-forma-tions

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

Resampling consists of ...

- 1) Convolution with an interpolation function and the sampled signal gives a continuous signal.
- 2) Resampling of the continuous signal gives the resampled signal.



Signal | Interpolated signal | Resampled signal

In reality, the continuous function is only calculated in the sample points of the resampled function.

Computation of a new sample value

Original function: $g(x)$
Sampled function: $\tilde{g}(x)$
Interpolation function: $w(x)$

$$w * \tilde{g}(x) = \int_{-\infty}^{\infty} w(x-a) \cdot \tilde{g}(a)\, da$$

Think: dirac impulses

$$= \int_{-\infty}^{\infty} w(x-a) \cdot \sum_{n=-\infty}^{\infty} g(a)\delta(a-n\Delta)\ da$$

$$= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} w(x-a) \cdot g(a)\delta(a-n\Delta)\ da$$

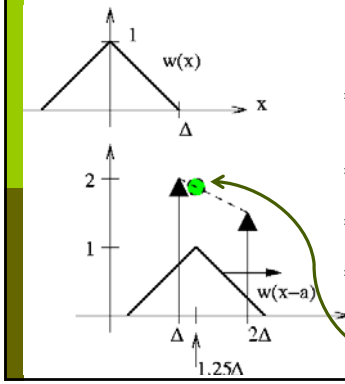
$$= \sum_{n=-\infty}^{\infty} w(x-n\Delta) \cdot g(n\Delta)$$

Think: sample values

Ex) Convolution with w(x)=Λ(x/Δ) (equals linear interpolation)

Think: Dirac impulses

$$w * \tilde{g}(x) = \int_{-\infty}^{\infty} w(x-a) \cdot \tilde{g}(a)\, da =$$



$$= \int_{-\infty}^{\infty} w(1.25\Delta - a) \cdot 2\delta(a-\Delta) + \ldots \ldots + w(1.25\Delta - a) \cdot 1.5\delta(a-2\Delta)da$$

$$= \int_{-\infty}^{\infty} w(1.25\Delta - \Delta) \cdot 2\delta(a-\Delta) + \ldots \ldots + w(1.25\Delta - 2\Delta) \cdot 1.5\delta(a-2\Delta)da$$

$$= w(0.25\Delta) \cdot 2 + w(-0.75\Delta) \cdot 1.5$$

$$= 0.75 \cdot 2 + 0.25 \cdot 1.5 = 1.875$$

Ex) Convolution with w(x)=Λ(x/Δ) (equals linear interpolation)

Think: Sample values

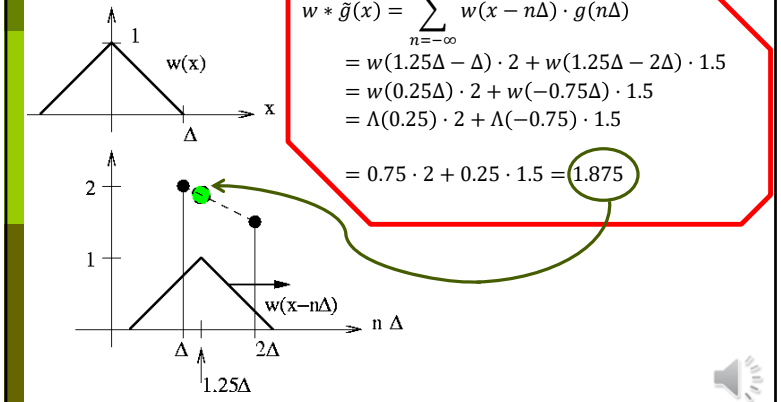
$$w * \tilde{g}(x) = \sum_{n=-\infty}^{\infty} w(x-n\Delta) \cdot g(n\Delta)$$

$$= w(1.25\Delta - \Delta) \cdot 2 + w(1.25\Delta - 2\Delta) \cdot 1.5$$

$$= w(0.25\Delta) \cdot 2 + w(-0.75\Delta) \cdot 1.5$$

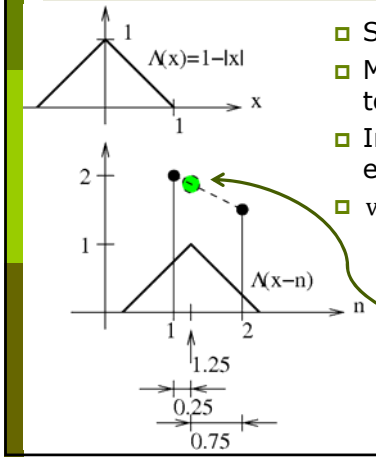
$$= \Lambda(0.25) \cdot 2 + \Lambda(-0.75) \cdot 1.5$$

$$= 0.75 \cdot 2 + 0.25 \cdot 1.5 = 1.875$$



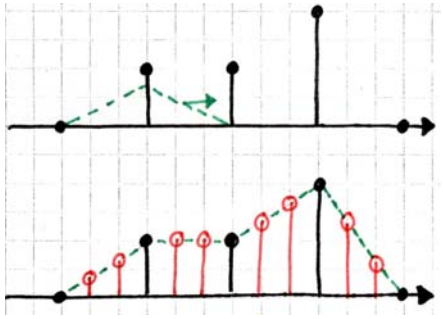
(Linear) Interpolation – very simple description

- Set Δ=1.
- Move the interpolation function to the point of interest.
- Interpolation functions are even => regard only distance
- value $= \Lambda(0.25) \cdot 2 + \Lambda(0.75) \cdot 1.5$
  $= 0.75 \cdot 2 + 0.25 \cdot 1.5 = 1.875$

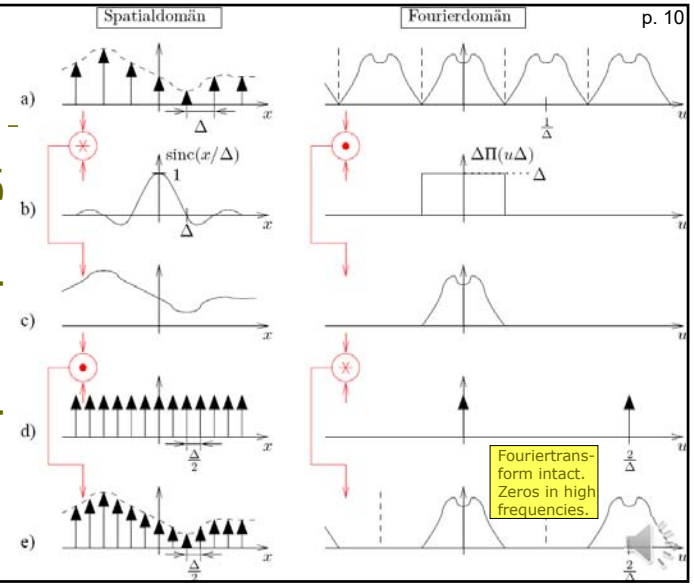


Ex) Linear interpolation

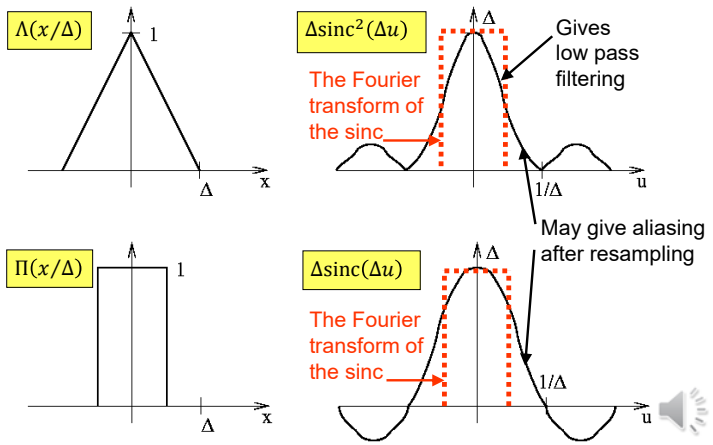
Up-
samp-
ling
a factor
of 3



Upsampling, ideal

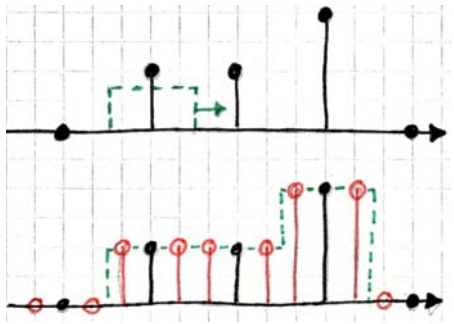


The Fourier transforms of linear and nearest neighbor interpolation



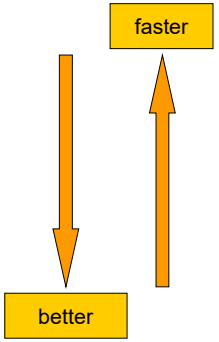
Ex) Nearest neighbor interpolation

Up-
samp-
ling
a factor
of 3



Interpolation functions

- Nearest neighbor interpolation
- Linear interpolation
- Truncated sinc interpolation
- Sinc interpolation



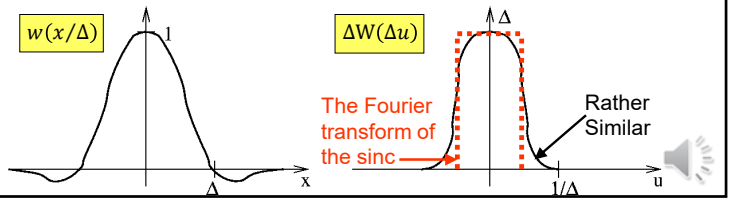
But cubic interpolation can be both faster and better than truncated sinc interpolation!

The cubic spline function

a=-0.5 approximates the sinc best in a least square manner. -0.5>a>-0.75 can give a sharper, more nice looking appearance.

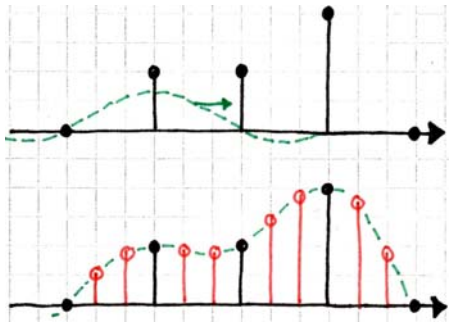
- A cubic spline function is put together by cubic splines (3:rd degree polynomials). The cubic spline function is continuous and has continuous 1:st degree derivatives in every point.

$$w(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1, & 0 \le |x| \le 1, \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a, & 1 \le |x| \le 2, \\ 0, & \text{otherwise.} \end{cases}$$



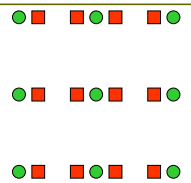
Ex) Cubic spline interpolation

Up-sampling a factor of 3

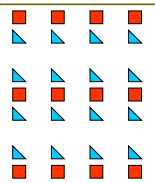


2-D up-sampling, 2 different methods

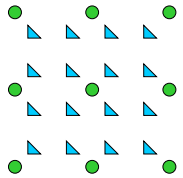
1a) 1D resampling horizontally, in x-dir:



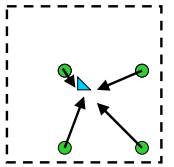
1b) 1D resampling vertically, in y-dir:



2a) 2D re-sampling directly:



2b) 2D technique:



Base area for a 2x2 interpolation function

2-D resampling

Sampled in-function: $f(x,y)$ Resulted continuous function: $g(x,y)$

Interpolation functions: $h(x)$ and $h(y)$ can be proved

Gives: $[h(y)\delta(x)] * [h(x)\delta(y)] = h(y) \cdot h(x)$

1D resamp-ling in y-dir: 1D resamp-ling in x-dir:

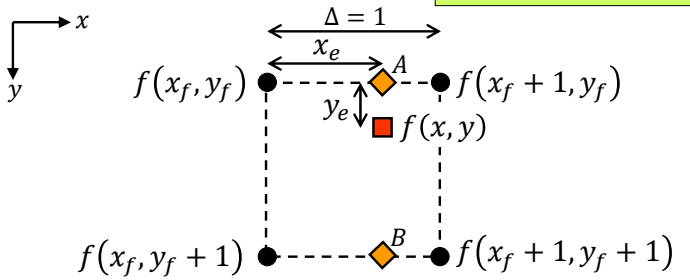
Method1: $g(x,y) = h(y)\delta(x) * [h(x)\delta(y) * f(x,y)]$
Method2: $g(x,y) = [h(x)h(y)] * f(x,y)$

Ideal 2D resampling function:
$h(x) \cdot h(y) = \mathrm{sinc}(x/\Delta) \cdot \mathrm{sinc}(y/\Delta)$

2D linear resampling function:
$h(x) \cdot h(y) = \Lambda(x/\Delta) \cdot \Lambda(y/\Delta)$

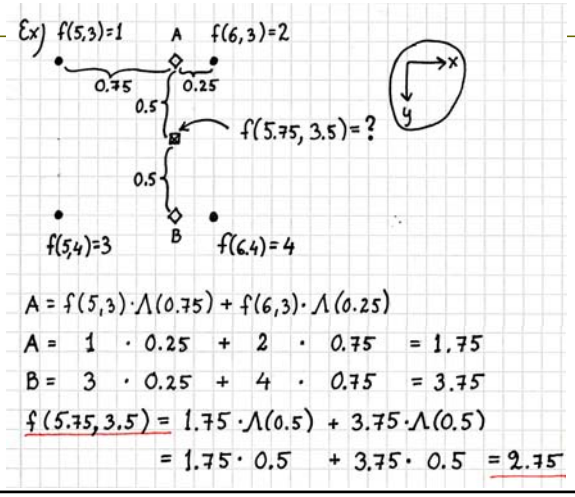
2-D bilinear inter-polation, method 1

$$\Lambda(x) = \begin{cases} 1-|x|, & |x| \le 1 \\ 0, & |x| \ge 1 \end{cases}$$



$$\begin{cases} A = f(x_f,y_f) \cdot (1-x_e) + f(x_f+1,y_f) \cdot x_e \\ B = f(x_f,y_f+1) \cdot (1-x_e) + f(x_f+1,y_f+1) \cdot x_e \\ f(x,y) = A \cdot (1-y_e) + B \cdot y_e \end{cases}$$

Ex) 2D bilinear interpolation



$A = f(5,3) \cdot \Lambda(0.75) + f(6,3) \cdot \Lambda(0.25)$

$A = 1 \cdot 0.25 + 2 \cdot 0.75 = 1.75$

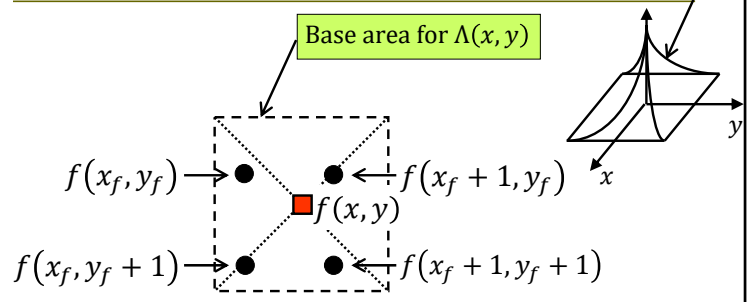
$B = 3 \cdot 0.25 + 4 \cdot 0.75 = 3.75$

$f(5.75, 3.5) = 1.75 \cdot \Lambda(0.5) + 3.75 \cdot \Lambda(0.5)$

$= 1.75 \cdot 0.5 + 3.75 \cdot 0.5 = 2.75$

2-D bilinear inter-polation, method 2

$$\Lambda(x,y) = \Lambda(x) \cdot \Lambda(y)$$

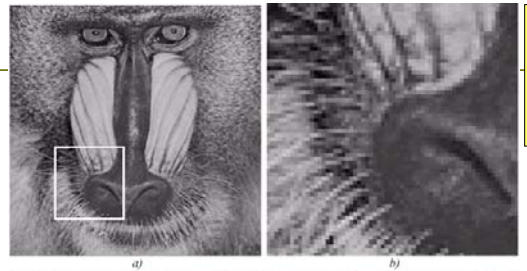


$$f(x,y) = f(x_f,y_f) \cdot (1-x_e)(1-y_e) + f(x_f+1,y_f) \cdot x_e(1-y_e) + f(x_f,y_f+1) \cdot (1-x_e)y_e + f(x_f+1,y_f+1) \cdot x_e y_e$$

2-D up-sampling, a factor 4

p. 21

original



Nearest neighbor interpolation

Bilinear interpolation

Bicubic interpolation

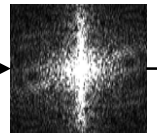
Ideal up-sampling

p. 22

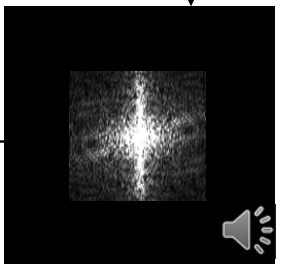
Which interpolation function corresponds to this method?

Image of size N x N

Matlab: fftshift(fft2(ifftshift( )))



Zero-pad

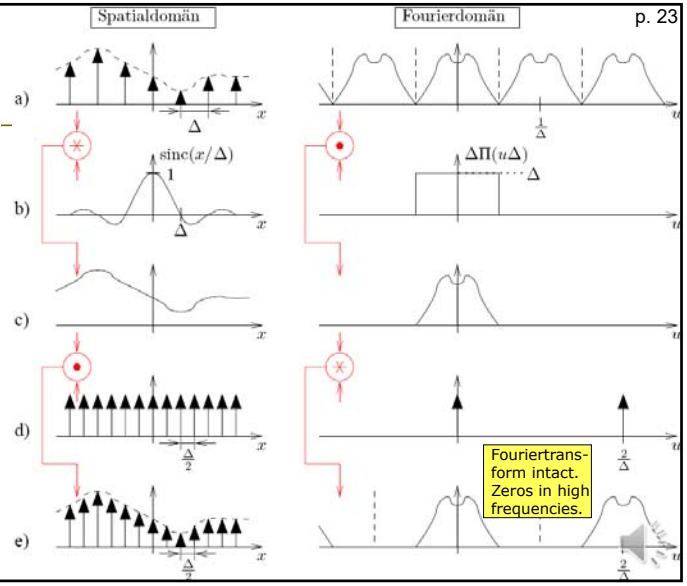


Matlab: fftshift(ifft2(ifftshift( )))

Image of size 2N x 2N

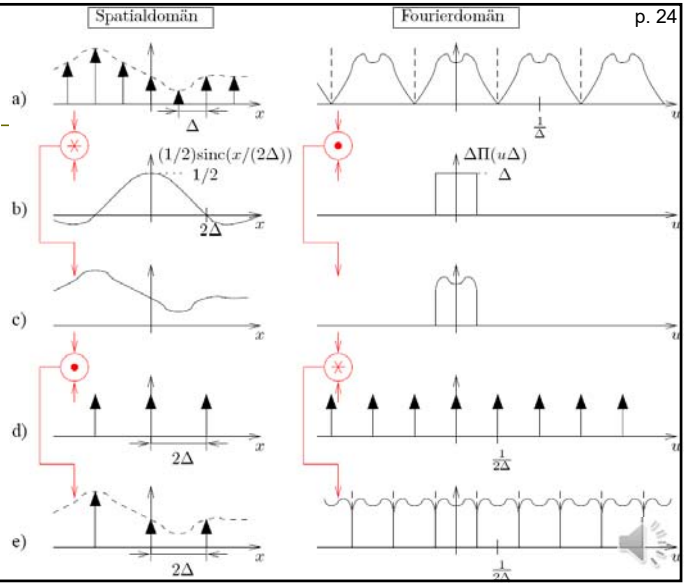
Upsampling, ideal

p. 23

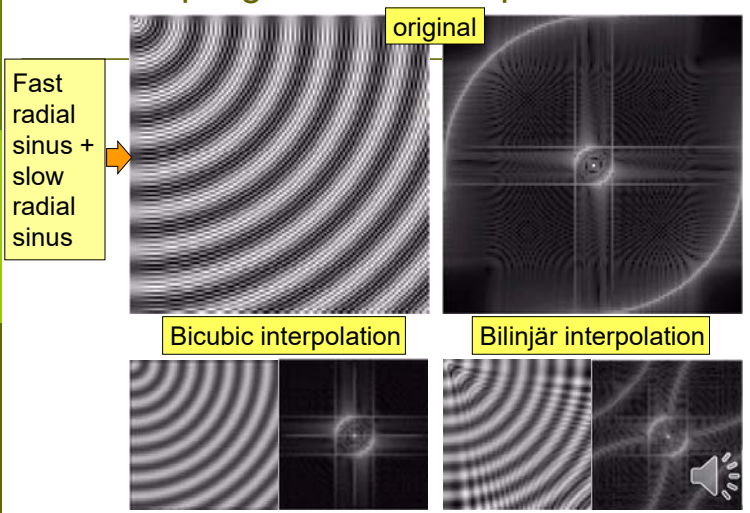


Downsampling, ideal

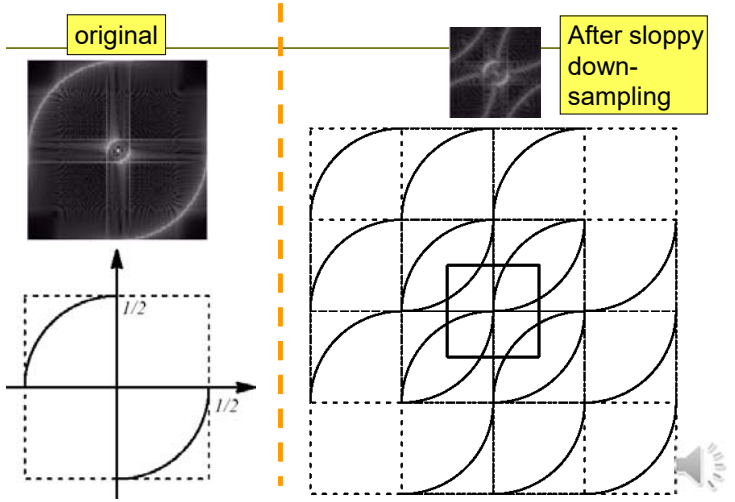
p. 24



Downsampling with diff. interpolation func.



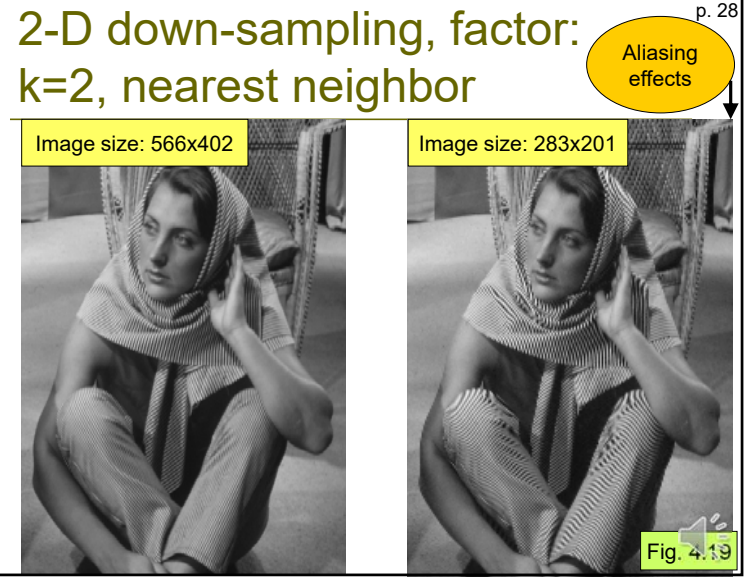
Downsampling with diff. interpolation func.



Conclusions for up- and down-sampling

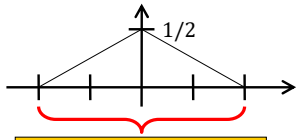
- The ideal interpolation function is the sinc function which corresponds to a rectangular function in the Fourier domain.
- During up-sampling, the edge of the rectangular function is located at the band-limit of the function to be interpolated.
- During down-sampling a factor of k, the edge of the rectangular function is at the band-limit/k of the function to be interpolated. The corresponding interpolation function is k times broader.
- Consequently, during down-sampling, the interpolation function must perform low-pass filtering. Otherwise, there will be aliasing.
- An approximate interpolation function is the triangle function, which is equivalent to linear interpolation. Its width = the width of the main lobe of the sinc. During up-sampling , the width of the triangle function is 2 pixels and during down-sampling its width is 2·k pixels.

2-D down-sampling, factor: k=2, nearest neighbor



2-D down-sampling, factor: k=2, bilinear interpolation

Aliasing almost gone



width = 2·k = 4 pixels

This is equivalent to convolution with:

$$\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix}/16 = \begin{pmatrix}1\\ 2\\ 1\end{pmatrix}/4 * (1 \quad 2 \quad 1)/4$$

i.e. low-pass filtering, followed by throwing away every second pixel

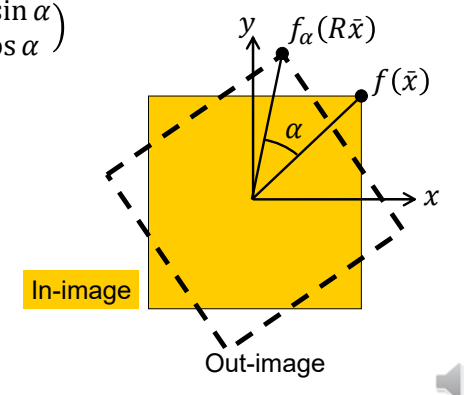
Image size: 283x201

Fig. 4.19

Rotation of an image

$$R = \begin{pmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{pmatrix}$$

$$\bar{x} = \begin{pmatrix}x\\ y\end{pmatrix}$$



2 mapping methods, here with rotation

Not a problem for rotation, however

1. For all $\bar{x}$ in the out−image $f_\alpha(\ \ )$
   Compute $R^{-1}\bar{x}$ in the in−image $f_\alpha(\ \ )$
   $f_\alpha(\bar{x}) := f(R^{-1}\bar{x})$

Inverse mapping, page 102

the value $f(R^{-1}\bar{x})$ must be interpolated in the inimage

Solution, see 5 slides ahead

2. For all $\bar{x}$ in the in−image $f(\ \ )$
   Compute $R\bar{x}$ in the out−image $f_\alpha(\ \ )$
   $f_\alpha(R\bar{x}) := f(\bar{x})$

Forward mapping, page 101

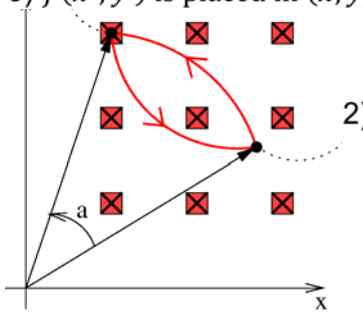
the value $f(\bar{x})$ in the inimage is spread out in the outimage

Page 102: A problem with the forward mapping is that two or more pixels in the input image can be transformed to the same location in the output image, raising the question of how to combine multiple output values into a single output pixel.

Rotation with inverse mapping

1) (x, y) is given
3) $f(x', y')$ is placed in (x, y)

Observe!
The in-image ■ and the out-image ✕ are overlaid.



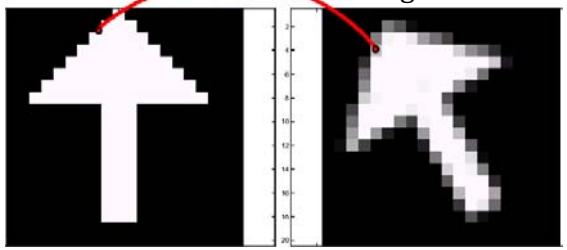
2) (x', y') is calculated, $f(x', y')$ is interpolated

$$\begin{pmatrix}x'\\ y'\end{pmatrix} = R^{-1}\begin{pmatrix}x\\ y\end{pmatrix}$$

Rotation with inverse mapping, example

$f_\alpha(\bar{x}) = f(R^{-1}\bar{x})$

the value $f(R^{-1}\bar{x})$ must be interpolated in the in−image

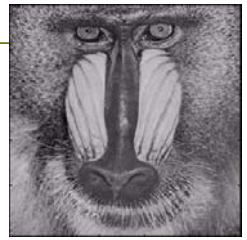


In-image f before rotation

Out-image f_α after rotation

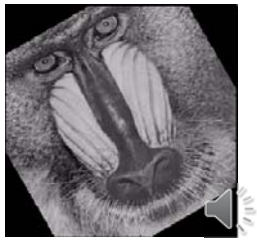
Rotation with bilinear interpolation

Rotation 30°



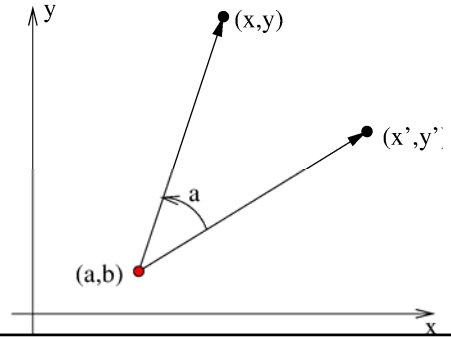
Why is the monkey more blurred after the rotations?

Rotation -30°



Rotation around (a,b)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x-a \\ y-b \end{pmatrix}$$



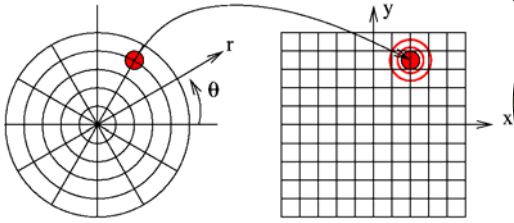
Resampling with forward mapping. Example.

Not in G&W!

- The in-image f(r,θ) is going to be resampled to the out-image g(x,y) with forward mapping.
- Every value f(r,θ) is spread out in g(x,y) using the filter weights h(x,y).
- The high density of function values in the center of f(r,θ) is handled by normalization with the weight-sum.
- Algorithm? See next slide!

Similar to a method called: "normalized convolution"

Another method, the "gridding method", recommends initial weighting with the sampling density, though



$f(r,\theta)$ in−image $g(x,y)$ out−image

Resampling with forward mapping. Algorithm.

Not in G&W!

$g(x,y) = H(x,y) = 0;$
For all (r,θ), $f(r,\theta) \neq 0$
$x = r\cos\theta\,;\ y = r\sin\theta\,;$
För all (x,y), $h(x - x_i, y - y_i) \neq 0$
$g(x,y) := g(x,y) + h(x - x_i, y - y_i) f(r,\theta);$
$H(x,y) := H(x,y) + h(x - x_i, y - y_i);$
End
End
For all (x,y)
$g(x,y) := \dfrac{g(x,y)}{H(x,y)};$
End

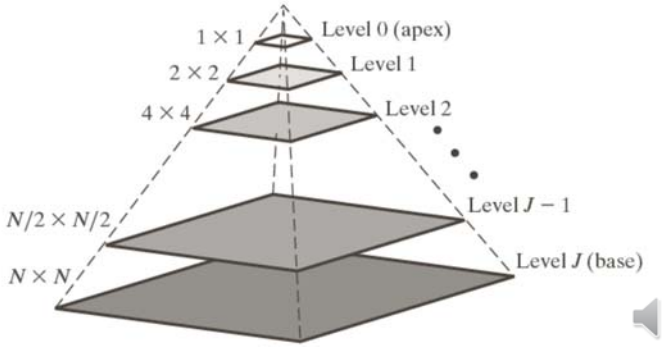
Spread out the f(r,θ) according to the weights in the filter h.

Book-keep the weights for each pixel.

Normalization with the sum of the weights

Short about image scale pyramids

- Useful when processing structures of different sizes.



An example of a pyramid

Convolve with $\begin{pmatrix}1\\2\\1\end{pmatrix} * (1 \quad 2 \quad 1)/16$

and through away every second pixel

Convolve with $\begin{pmatrix}1\\2\\1\end{pmatrix} * (1 \quad 2 \quad 1)/16$

and through away every second pixel

Conclusion about image scale pyramids

- It is often useful to work with images in different scales to process structures of different sizes.
- Image scale pyramids normally have the scale 2^i x 2^i, where i is an integer.
- Low pass filters must be used when constructing a pyramid
- It is possible to construct an ideal scale pyramid by convolving with a sinc or multiplying with a rectangle in the Fourier domain. However, beware of ringing artefacts (Gibbs phenomena).