Photogrammetry & Robotics Lab

Geometric Transformations

Cyrill Stachniss

The slides have been created by Cyrill Stachniss.
Partial slides courtesy by S. Seitz.

Rectification

- Process of correcting an image distortion by transforming the image
- Input: **distorted** image
- Output: **rectified** image

Image courtesy: SIGPAC

Rendering or Texture Mapping

- Input: **rectified** image
- Output: **warped** image (on an object)

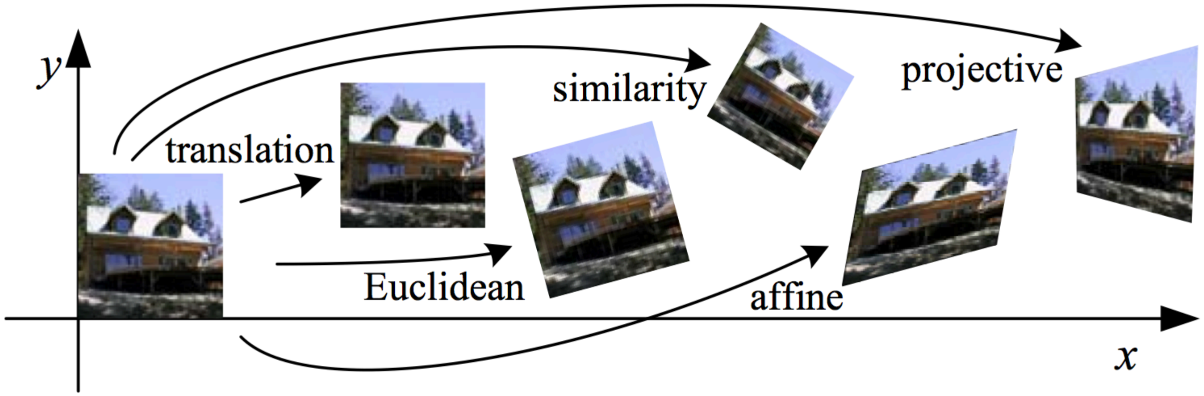


Image courtesy:
Szeliski, Jurassic Park

Registration

- Input and output are **differently distorted** images
- Goal: Alignment of both images

Geometric Transformations Between Images

- Every image has an own coordinate system
- Image $a(x, y)$ with c.s. S_a
- Image $b(x, y)$ with c.s. S_b
- Coordinates:

$$
{}^{a}\boldsymbol{x} = \begin{bmatrix} {}^{a}x \\ {}^{a}y \end{bmatrix} \quad \text{and} \quad {}^{b}\boldsymbol{x} = \begin{bmatrix} {}^{b}x \\ {}^{b}y \end{bmatrix}
$$

S_a $\qquad$ S_b

Geometric Transformations Between Images

- Goal: transformation from S_b to S_a
- This transforms $[\,{}^b x,\ {}^b y] \rightarrow [\,{}^a x,\ {}^a y]$
- This transformation is T

$$ {}^a\boldsymbol{x} = {}^a\boldsymbol{T}_b(\,{}^b\boldsymbol{x}) $$

to from

- Analogously: ${}^b\boldsymbol{x} = {}^b\boldsymbol{T}'_a(\,{}^a\boldsymbol{x})$

Geometric Transformations Between Images

- The expression ${}^{a}\boldsymbol{x} = {}^{a}\boldsymbol{T}_b({}^{b}\boldsymbol{x})$
- is a short form for

$$ {}^{a}\boldsymbol{x} = \begin{bmatrix} {}^{a}x \\ {}^{a}y \end{bmatrix} = \begin{bmatrix} T_x({}^{b}x, {}^{b}y) \\ T_y({}^{b}x, {}^{b}y) \end{bmatrix} = {}^{a}\boldsymbol{T}_b({}^{b}\boldsymbol{x}) $$

1st and 2nd dimension
of the function

Example for a Geometric Transformation

- Translation

$$
\begin{aligned}
\begin{bmatrix} {}^{a}x \\ {}^{a}y \end{bmatrix} &= \begin{bmatrix} T_x({}^{b}x,\ {}^{b}y) \\ T_y({}^{b}x,\ {}^{b}y) \end{bmatrix} \\
&= \begin{bmatrix} {}^{b}x \\ {}^{b}y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}
\end{aligned}
$$

Example for a Geometric Transformation

- Affine transformation

$$
\begin{aligned}
\begin{bmatrix} {}^a x \\ {}^a y \end{bmatrix} &= \begin{bmatrix} T_x({}^b x, {}^b y) \\ T_y({}^b x, {}^b y) \end{bmatrix} \\
&= \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} {}^b x \\ {}^b y \end{bmatrix} + \begin{bmatrix} h_{13} \\ h_{23} \end{bmatrix}
\end{aligned}
$$

Example

$$\begin{bmatrix} {}^a x \\ {}^a y \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} {}^b x \\ {}^b y \end{bmatrix} + \begin{bmatrix} h_{13} \\ h_{23} \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix} \begin{bmatrix} {}^b x \\ {}^b y \end{bmatrix} + \begin{bmatrix} 10 \\ 10 \end{bmatrix}$$

What is this transformation doing?

Example

$$
\begin{aligned}
\begin{bmatrix} {}^{a}x \\ {}^{a}y \end{bmatrix} &= \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} {}^{b}x \\ {}^{b}y \end{bmatrix} + \begin{bmatrix} h_{13} \\ h_{23} \end{bmatrix} \\
&= \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix} \begin{bmatrix} {}^{b}x \\ {}^{b}y \end{bmatrix} + \begin{bmatrix} 10 \\ 10 \end{bmatrix}
\end{aligned}
$$

- Scales the size of the image by 1/4
- Shifts the scaled image by 10 pixels in x and y direction

Example for Pixel (4,40)

$$\begin{bmatrix} {}^a x \\ {}^a y \end{bmatrix} = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix} \begin{bmatrix} 4 \\ 40 \end{bmatrix} + \begin{bmatrix} 10 \\ 10 \end{bmatrix}$$

$$= \begin{bmatrix} 11 \\ 20 \end{bmatrix}$$

pixel (4,40)

Problem

$$\begin{bmatrix} {}^a x \\ {}^a y \end{bmatrix} = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.25 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 10 \\ 10 \end{bmatrix}$$

$$= \begin{bmatrix} 10.25 \\ 10.25 \end{bmatrix}$$

pixel (1,1)

- The transformed pixel index is not an integer anymore

What to do?

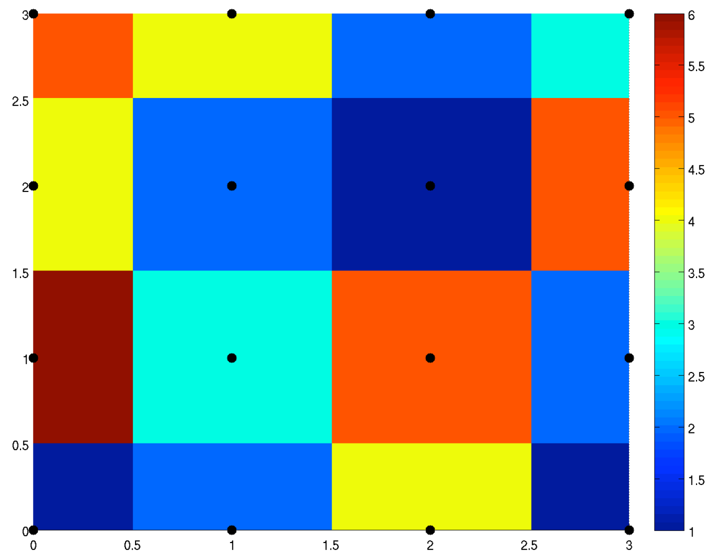
Resampling

- The transformation leads to **non-integer coordinates**
- To assign intensity values from the input to the output images, we need to **interpolate**
- Performing a discretization and quantization is called **resampling**

Nearest Neighbor Interpolation

- Choose the same color value as the closest nearby pixel
- Results in rounding the pixel position

$$a([\,{}^a x], [\,{}^a y]) = b(\,{}^b x,\ {}^b y)$$



Can We Do better?

Bilinear Interpolation

Bilinear Interpolation

Linear interpolation in x-direction, followed by the y-direction

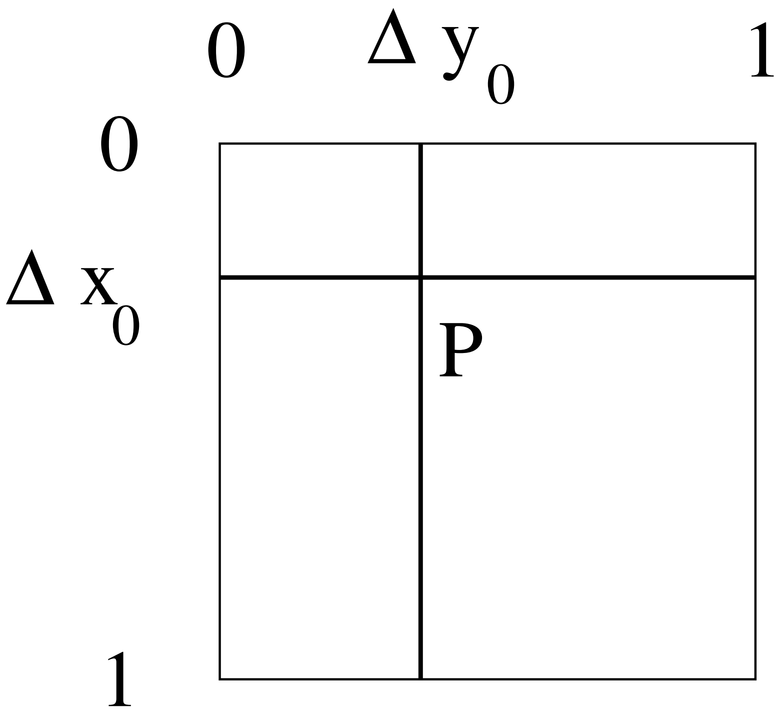
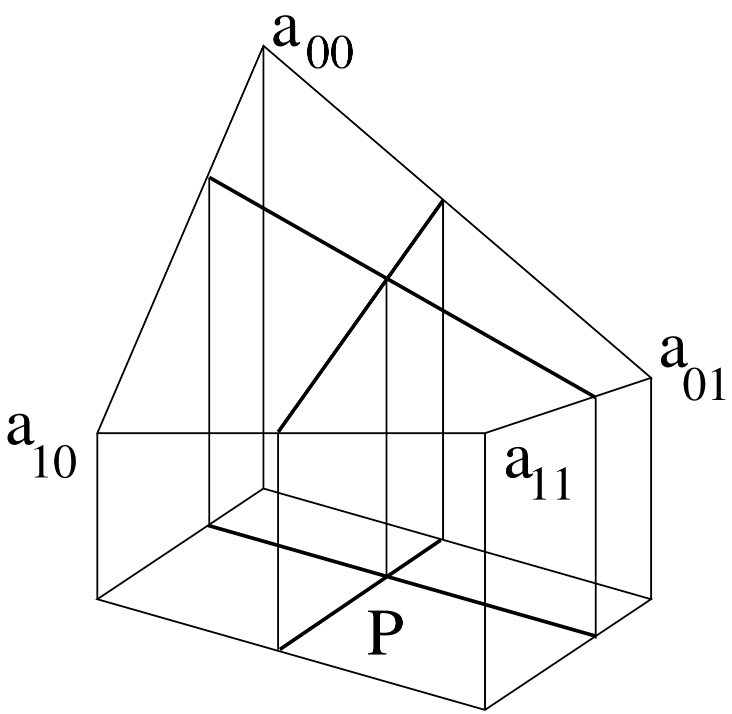


Image courtesy: Förstner

Bilinear Interpolation

Linear interpolation in x-direction, followed by the y-direction



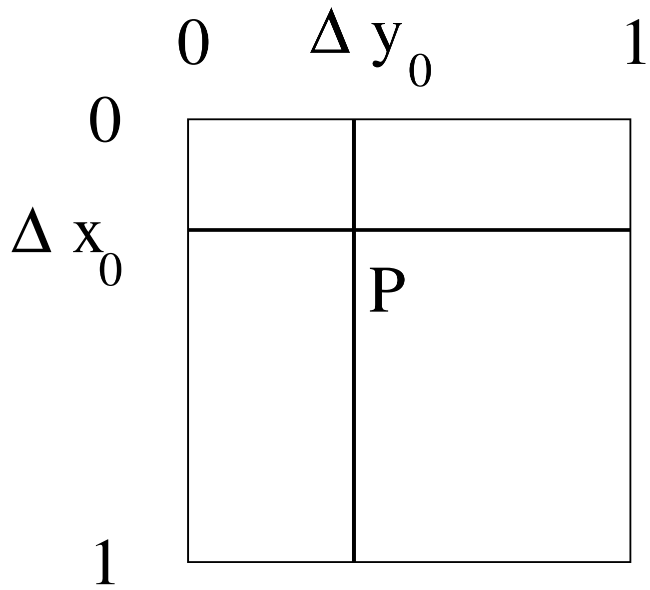


Image courtesy: Förstner

Bilinear Interpolation

Linear interpolation in x-direction, followed by the y-direction

$$
\begin{aligned}
b({}^b x, {}^b y) \quad = \quad & a_{00}(1-\Delta x)(1-\Delta y) + a_{01}(1-\Delta x)\Delta y \\
& + a_{10}\Delta x(1-\Delta y) + a_{11}\Delta x \Delta y
\end{aligned}
$$

four neighboring pixel intensity values

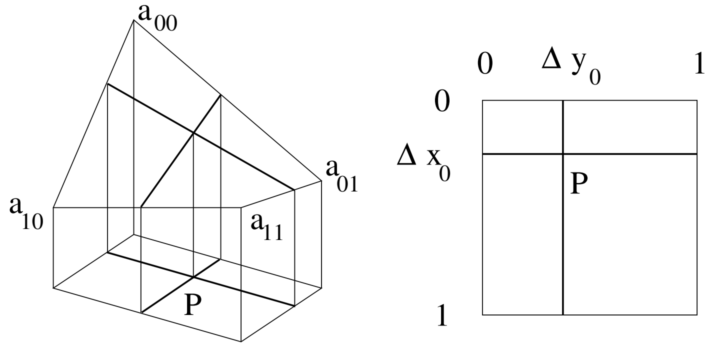
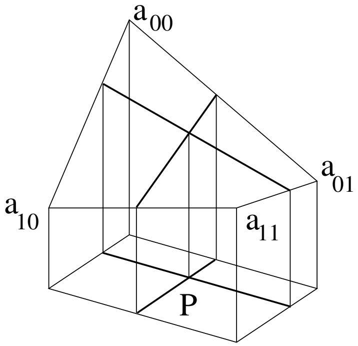


Image courtesy: Förstner

Bilinear Interpolation

Linear interpolation in x-direction, followed by the y-direction

$$
\begin{aligned}
b({}^{b}x,\ {}^{b}y) &= a_{00}(1-\Delta x)(1-\Delta y) + a_{01}(1-\Delta x)\Delta y \\
&\quad + a_{10}\Delta x(1-\Delta y) + a_{11}\Delta x \Delta y \\
&= a_{00} + \Delta x(a_{10}-a_{00}) + \Delta y(a_{01}-a_{00}) \\
&\quad + \Delta x \Delta y(a_{00}-a_{01}-a_{10}+a_{11})
\end{aligned}
$$



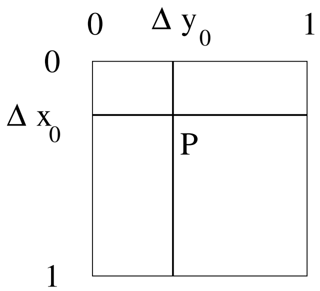
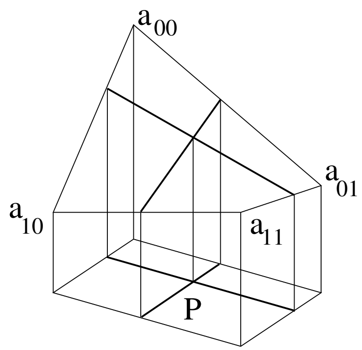


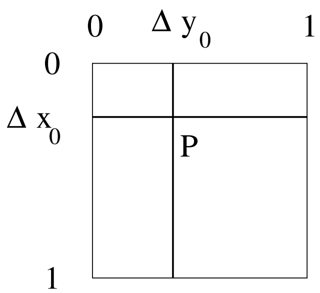
Image courtesy: Förstner

Bilinear Interpolation

Weighted average of the neighboring intensity values $a_{00}, a_{10}, a_{01}, a_{11}$

$$
\begin{aligned}
b({}^b x, {}^b y) &= a_{00}(1-\Delta x)(1-\Delta y) + a_{01}(1-\Delta x)\Delta y \\
&\quad + a_{10}\Delta x(1-\Delta y) + a_{11}\Delta x \Delta y \\
&= \underbrace{a_{00}}_{c_{00}} + \Delta x \underbrace{(a_{10}-a_{00})}_{c_{10}} + \Delta y \underbrace{(a_{01}-a_{00})}_{c_{01}} \\
&\quad + \Delta x \Delta y \underbrace{(a_{00}-a_{01}-a_{10}+a_{11})}_{c_{11}}
\end{aligned}
$$



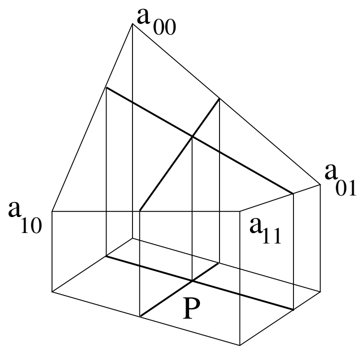


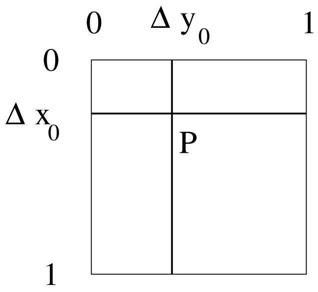
$$z = \sum_{i \le 1} \sum_{j \le 1} c_{ij} \Delta x^i \, \Delta y^j$$

Image courtesy: Förstner

Bilinear Interpolation

Weighted average of the neighboring intensity values $a_{00}, a_{10}, a_{01}, a_{11}$





$$z = \sum_{i \leq 1} \sum_{j \leq 1} c_{ij} \Delta x^i \, \Delta y^j$$

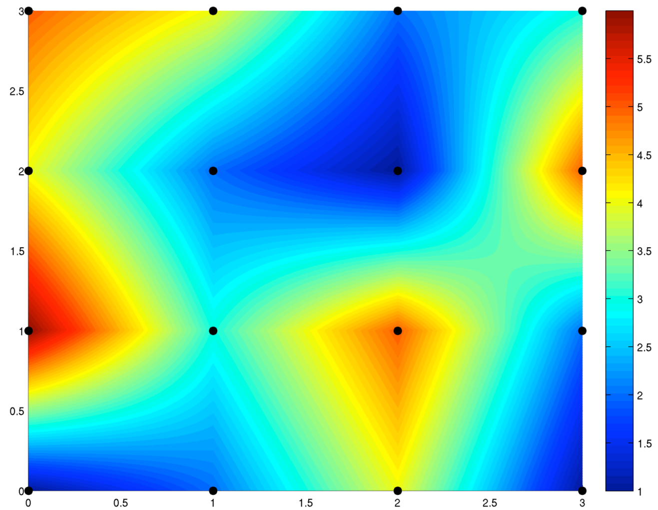
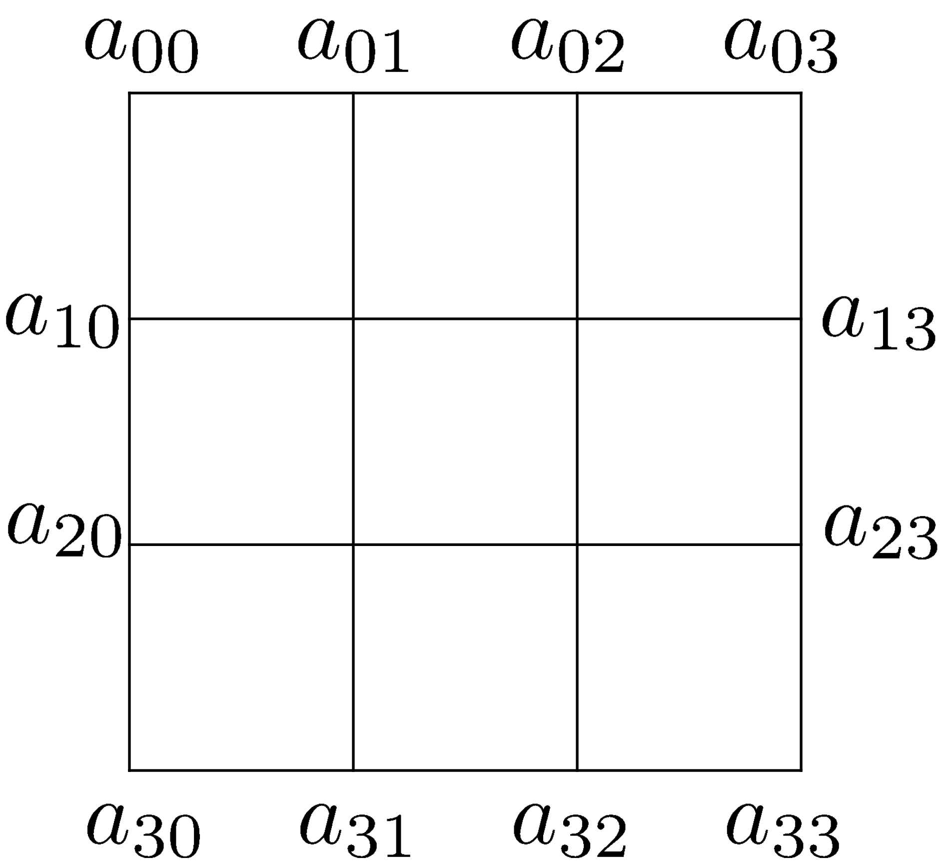


Image courtesy: Förstner, Wikipedia.org

Bicubic Interpolation

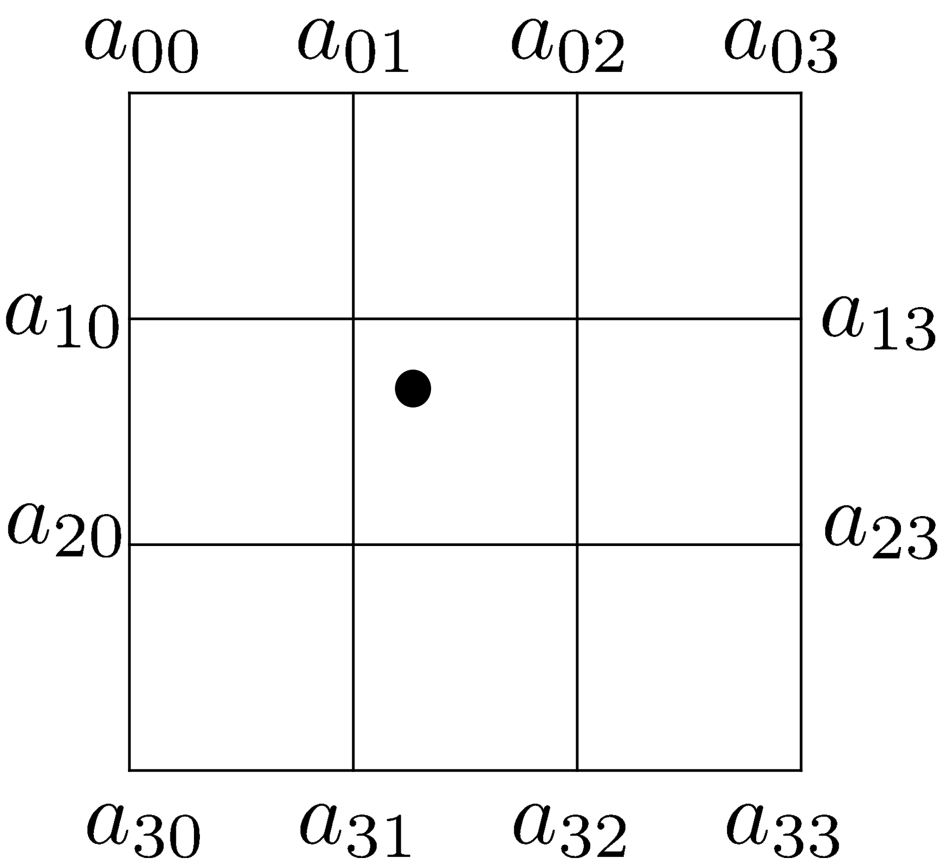
Bicubic Interpolation

- Cubic interpolation in x-direction, followed by the y-direction



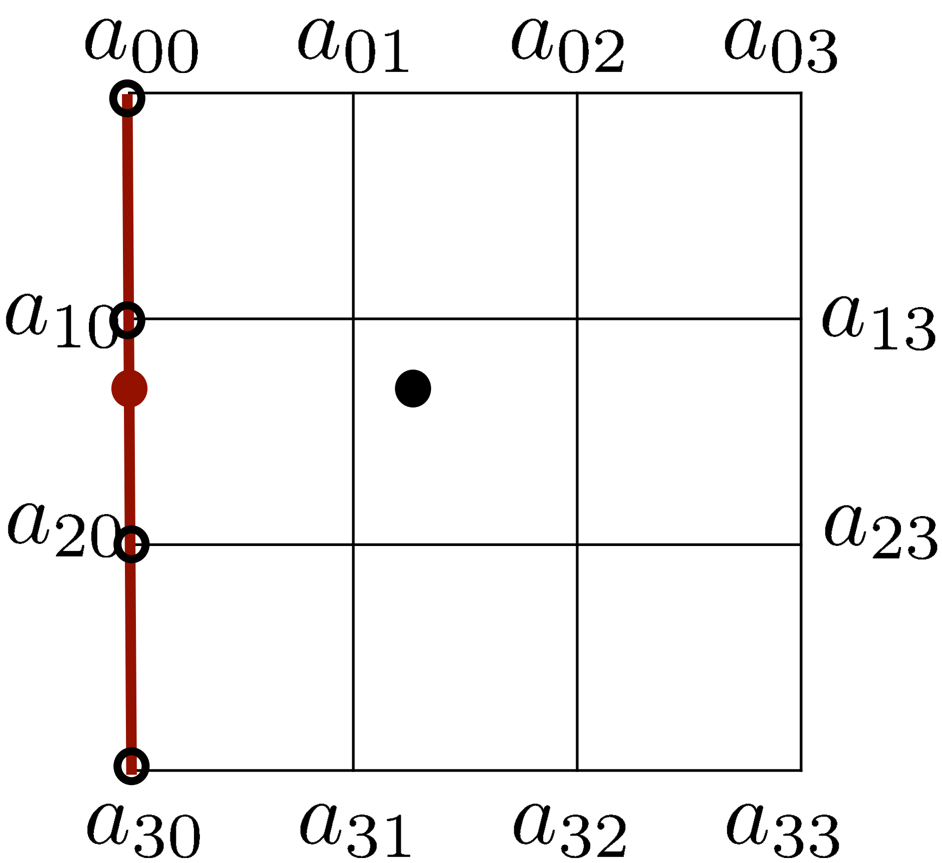
Bicubic Interpolation

- Cubic interpolation in x-direction, followed by the y-direction



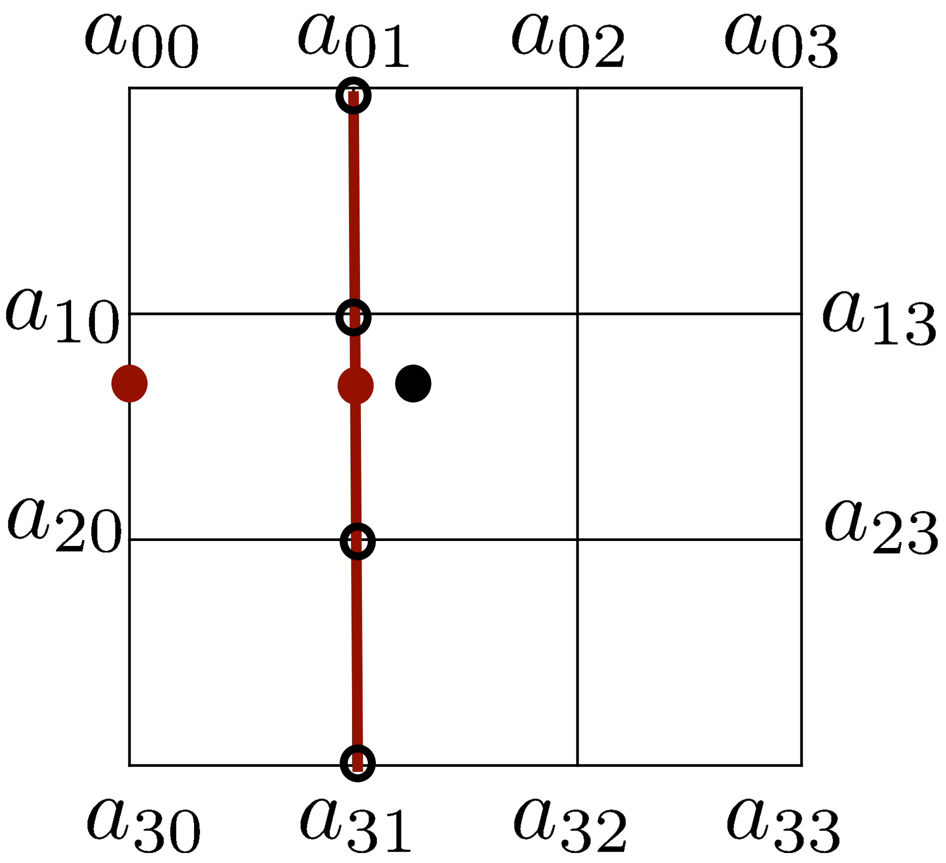
Bicubic Interpolation

- Cubic interpolation in x-direction, followed by the y-direction



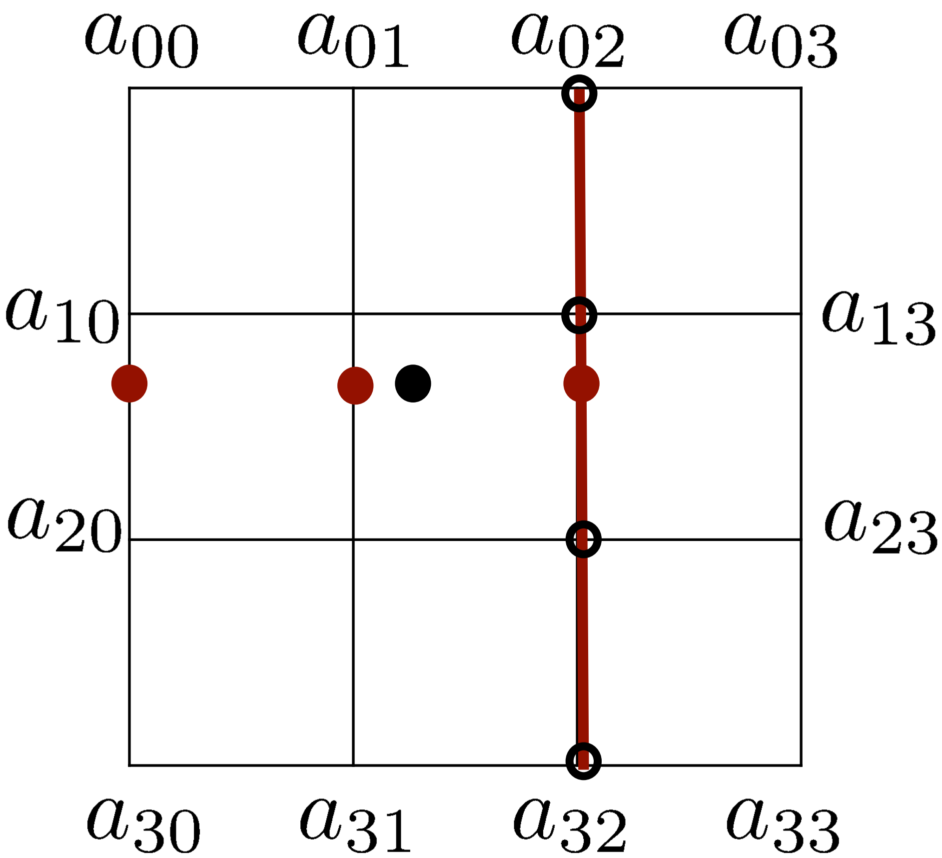
Bicubic Interpolation

- Cubic interpolation in x-direction, followed by the y-direction



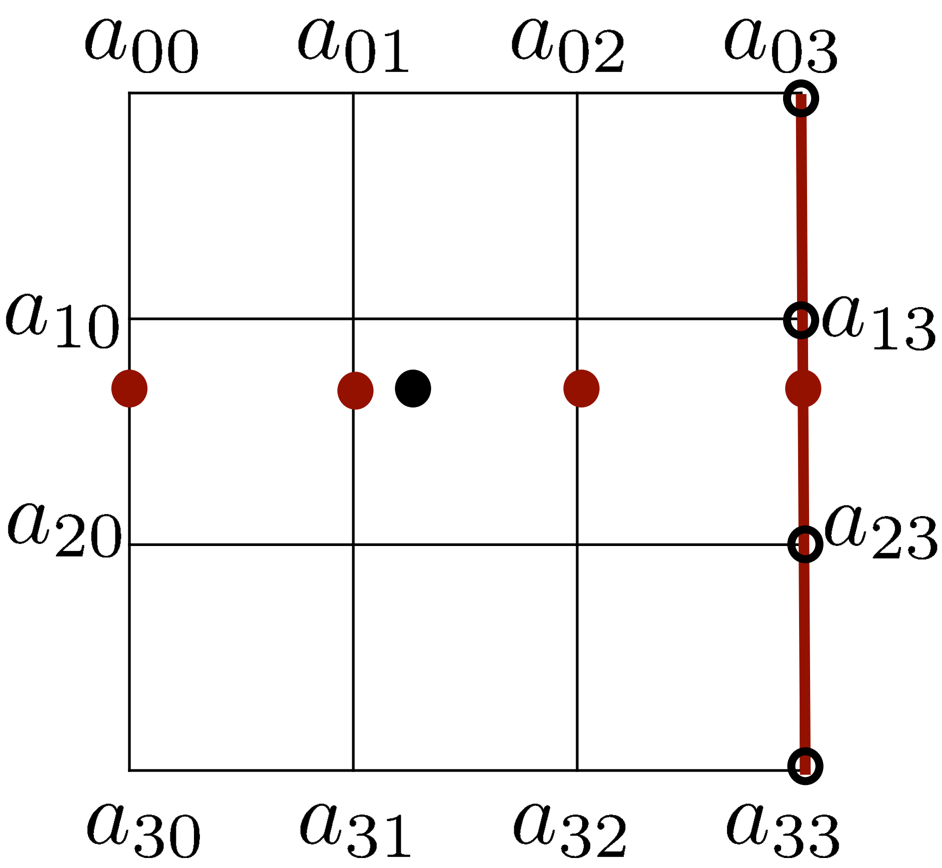
Bicubic Interpolation

- Cubic interpolation in x-direction, followed by the y-direction



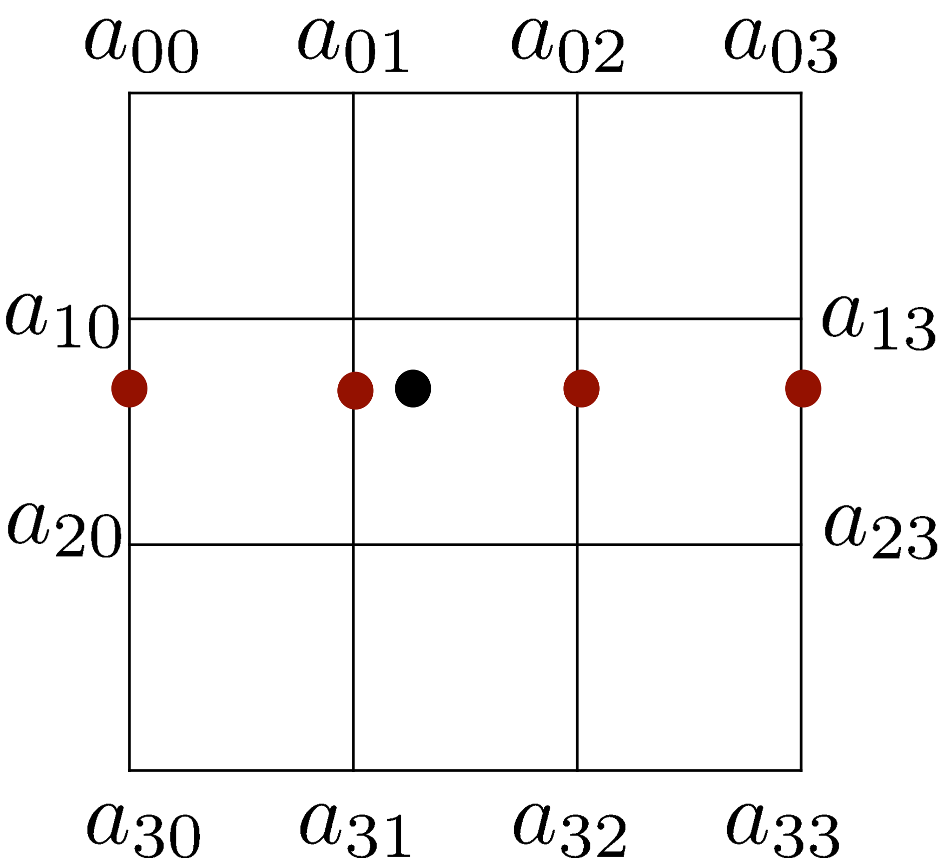
Bicubic Interpolation

- Cubic interpolation in x-direction, followed by the y-direction



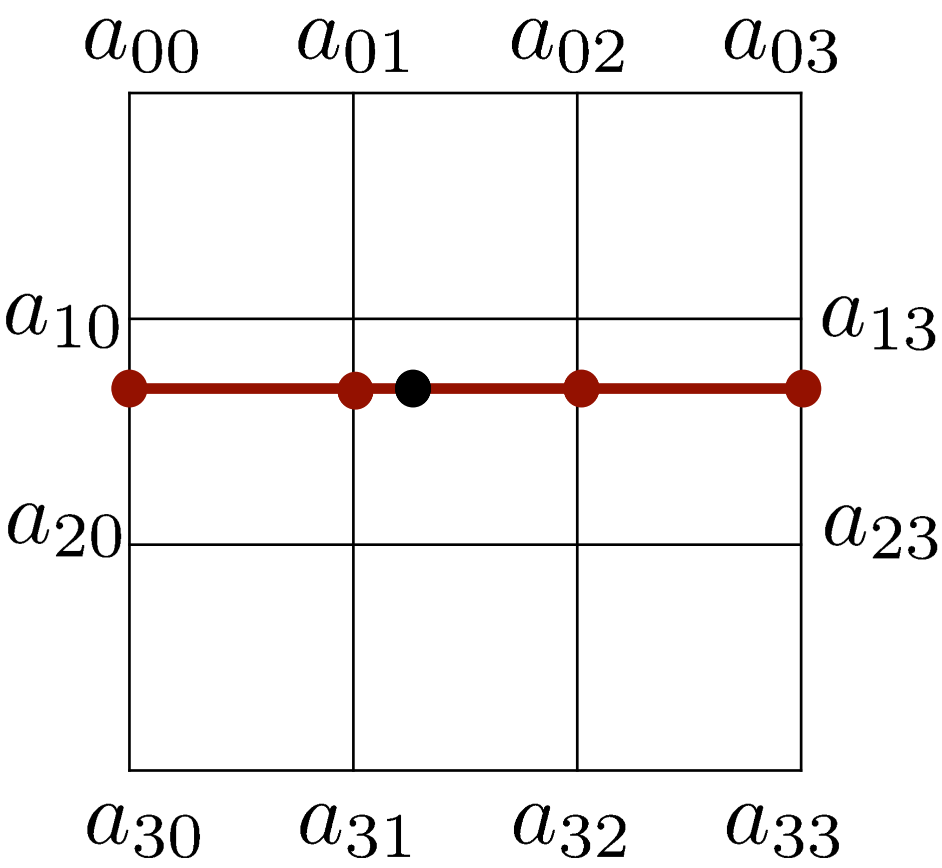
Bicubic Interpolation

- Cubic interpolation in x-direction, followed by the y-direction



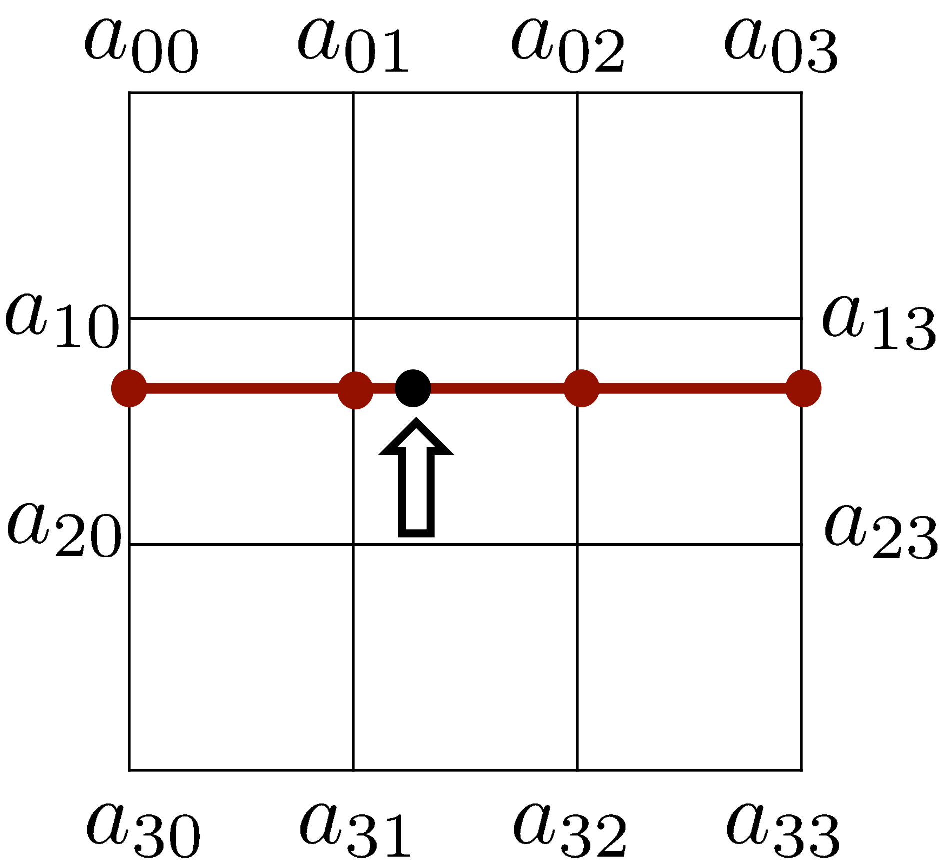
Bicubic Interpolation

- Cubic interpolation in x-direction, followed by the y-direction



Bicubic Interpolation

- Cubic interpolation in x-direction, followed by the y-direction



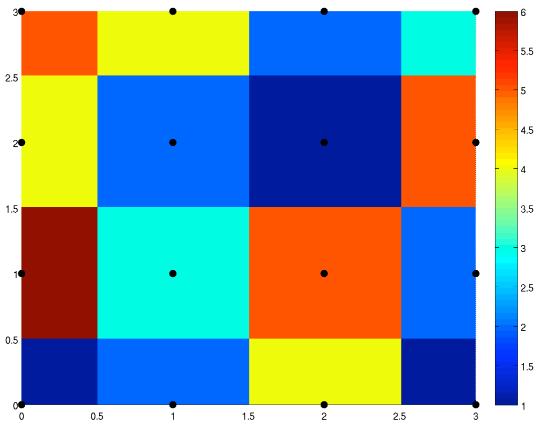
Bicubic Interpolation

- Cubic interpolation in x-direction, followed by the y-direction

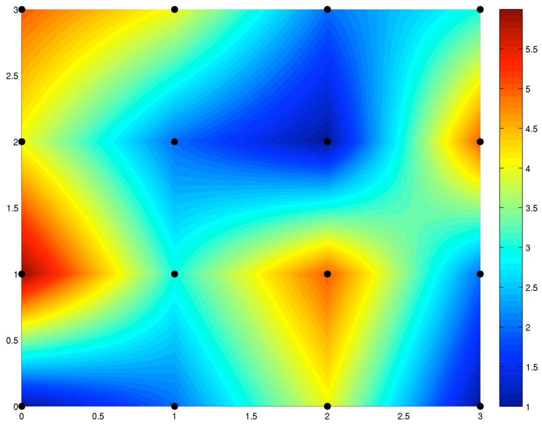
$$z = \sum_{i \leq 3} \sum_{j \leq 3} c_{ij} \Delta x^i \, \Delta y^j$$

- Result depends on 16 coefficients c_{ij}
- Yields a cubic spline interpolation

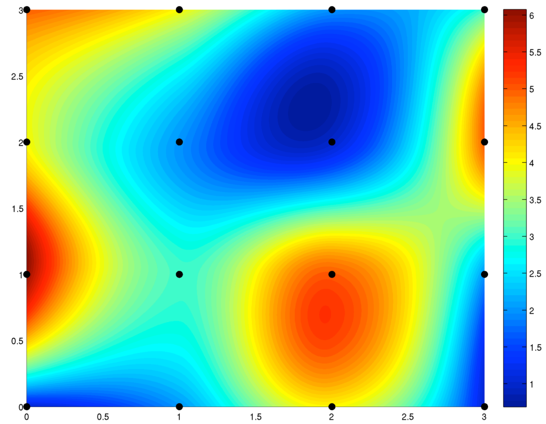
Interpolation Example



nearest neighbor
interpolation



bilinear
interpolation



bicubic
interpolation

Image courtesy: Wikipedia.org

Properties of the Individual Interpolation Methods

	Speed	Quality
NN	++	-
Bilinear	+	0
Bicubic	-	++

Interpolation Example (NN)

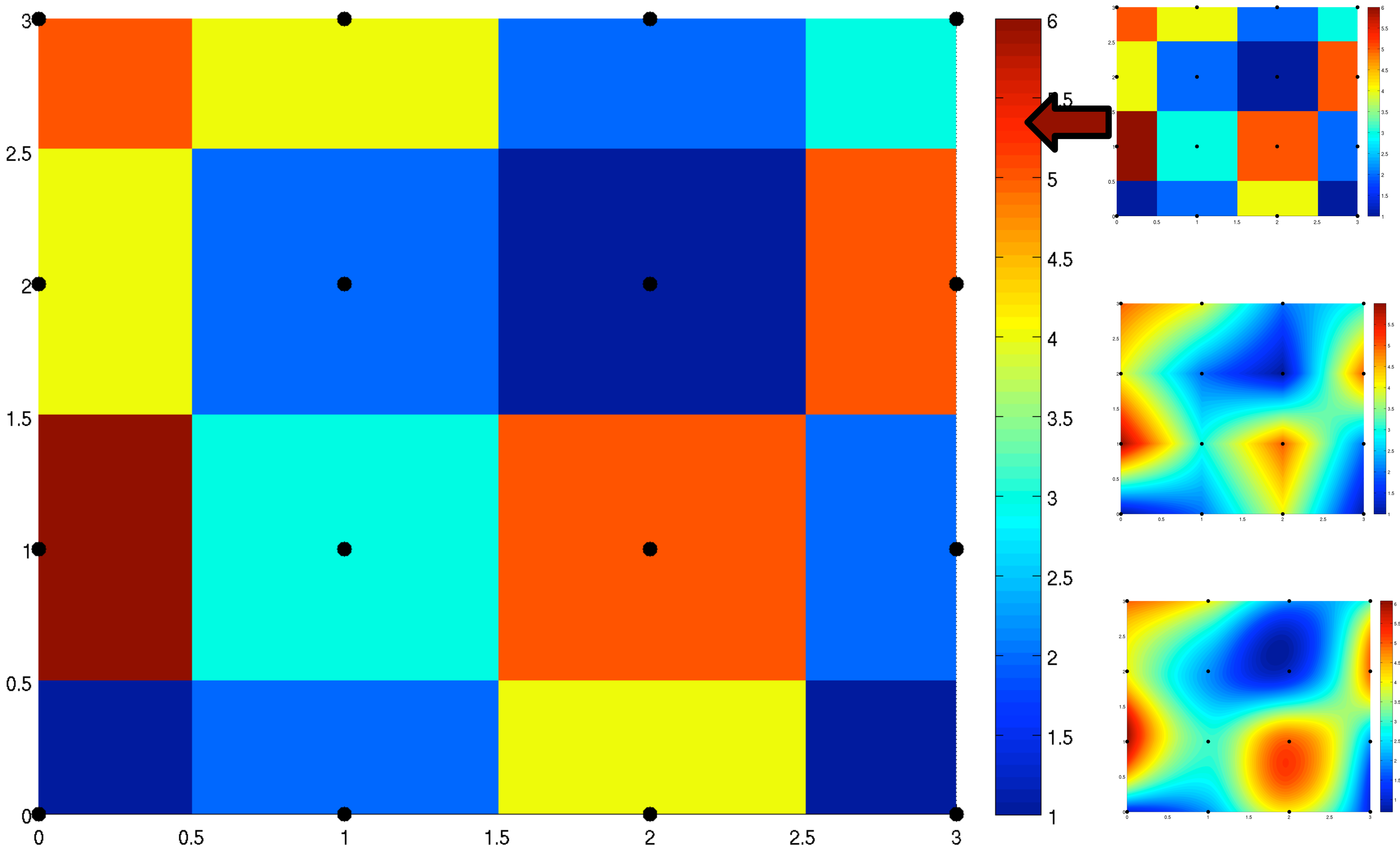


Image courtesy: Wikipedia.org

Interpolation Example (bilinear)

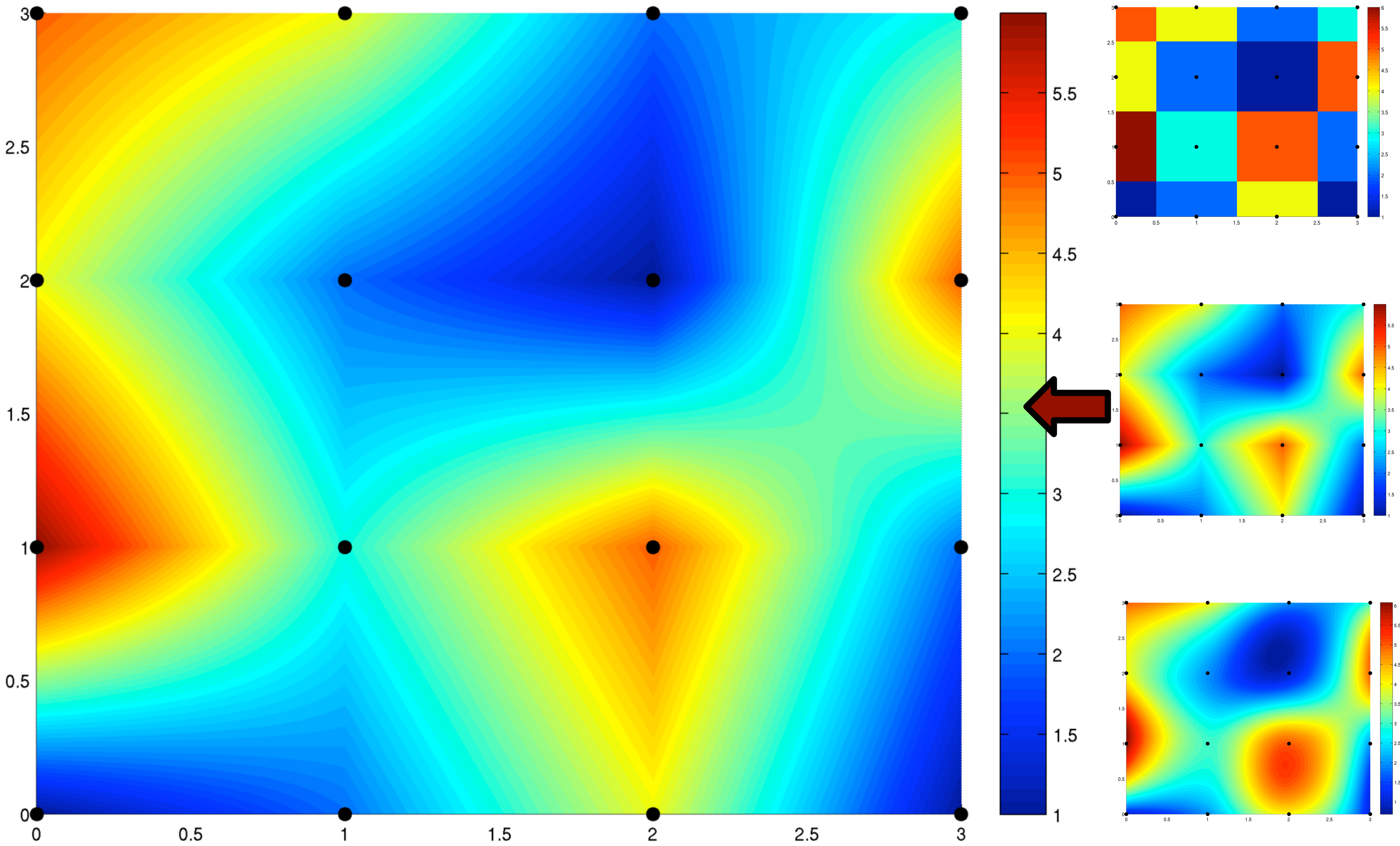


Image courtesy: Wikipedia.org 38

Interpolation Example (bicubic)

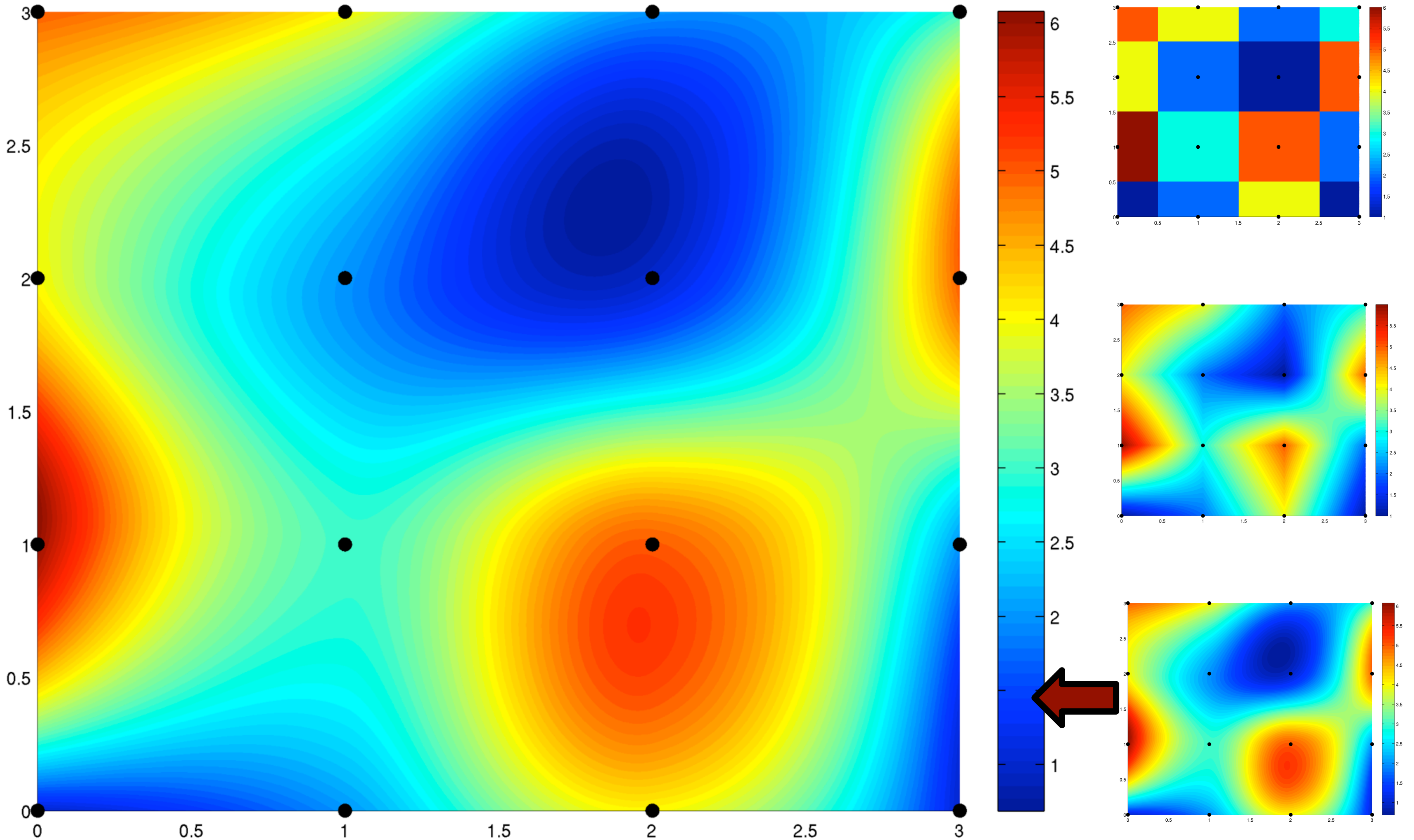
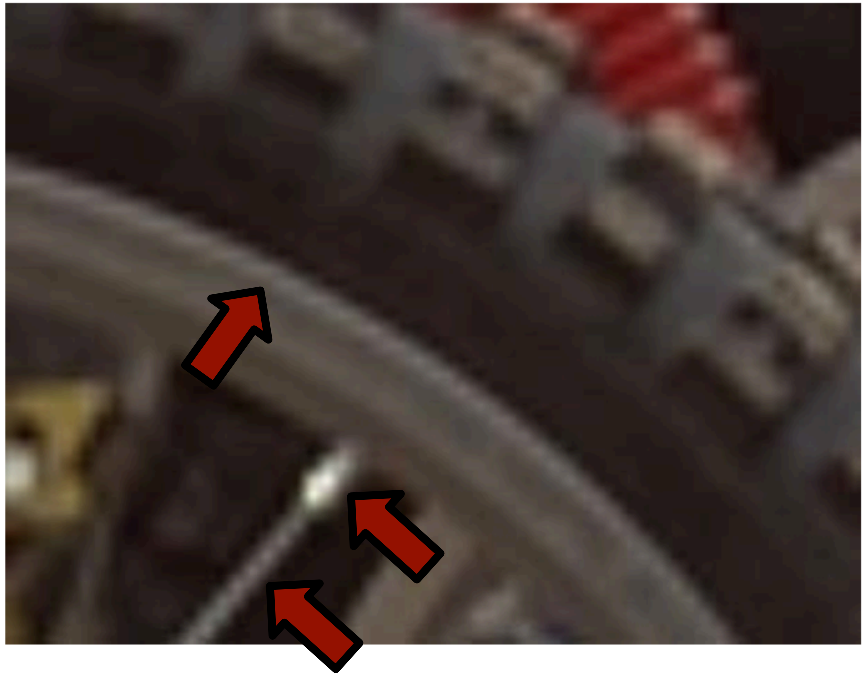


Image courtesy: Wikipedia.org

Example: Bilinear vs. Bicubic Interpolation



bilinear interpolation

bicubic interpolation

Image courtesy: Szeliski

Forward vs. Backward Warping

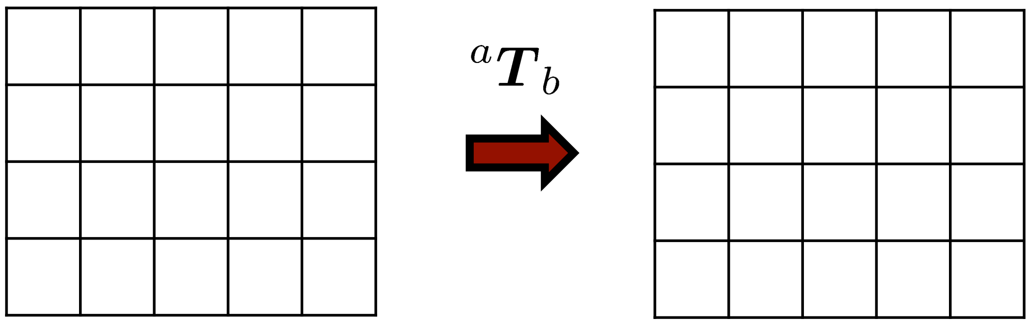
Mapping from the input to the output or from the output to the input image?

Resampling – Forward warping

Forward warping:

Compute for every pixel in the input image b a value in the output a

$$\forall\ {}^b\boldsymbol{x} \quad {}^a\boldsymbol{x} = {}^a\boldsymbol{T}_b({}^b\boldsymbol{x})$$



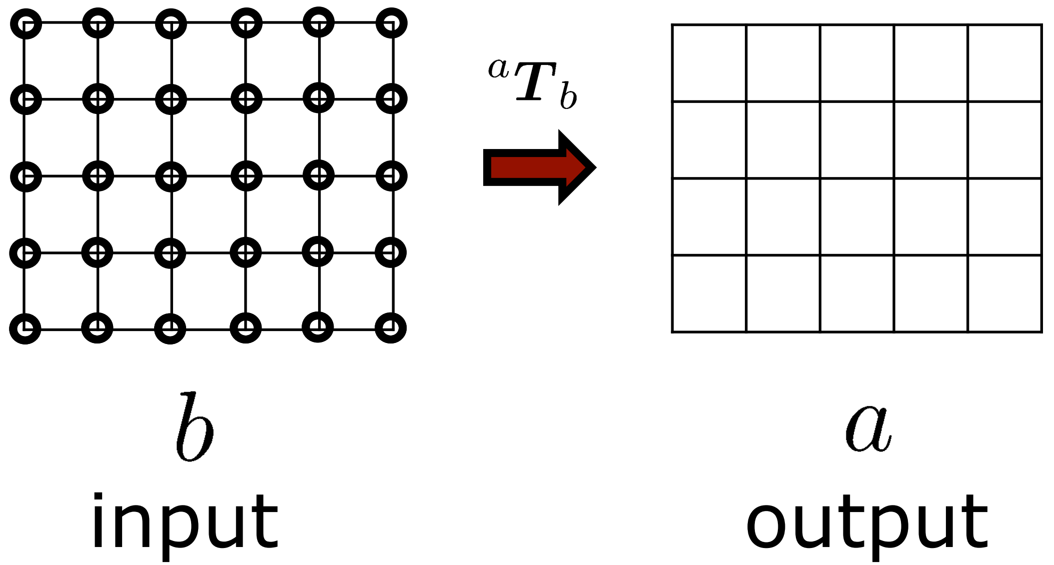
b input $\qquad$ a output

Resampling – Forward warping

Forward warping:

Compute for every pixel in the input image b a value in the output a

$$\forall\ {}^{b}\boldsymbol{x} \quad {}^{a}\boldsymbol{x} = {}^{a}\boldsymbol{T}_{b}({}^{b}\boldsymbol{x})$$

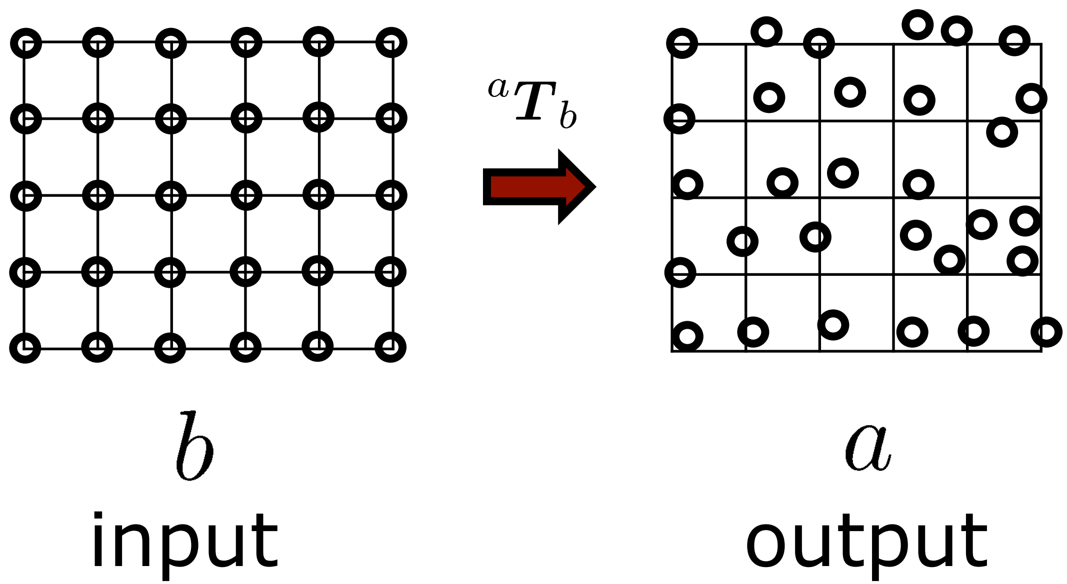


Resampling – Forward warping

Forward warping:

Compute for every pixel in the input image b a value in the output a

$$\forall\ {}^b\boldsymbol{x} \quad {}^a\boldsymbol{x} = {}^a\boldsymbol{T}_b(\,{}^b\boldsymbol{x})$$

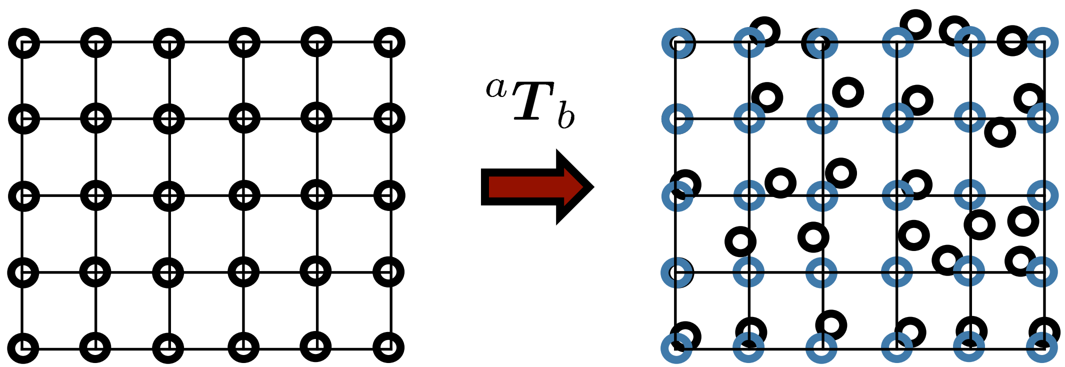


Resampling – Forward warping

Forward warping:

Compute for every pixel in the input image b a value in the output a

$$\forall\ {}^b\boldsymbol{x} \qquad {}^a\boldsymbol{x} = {}^a\boldsymbol{T}_b(\ {}^b\boldsymbol{x})$$



b input

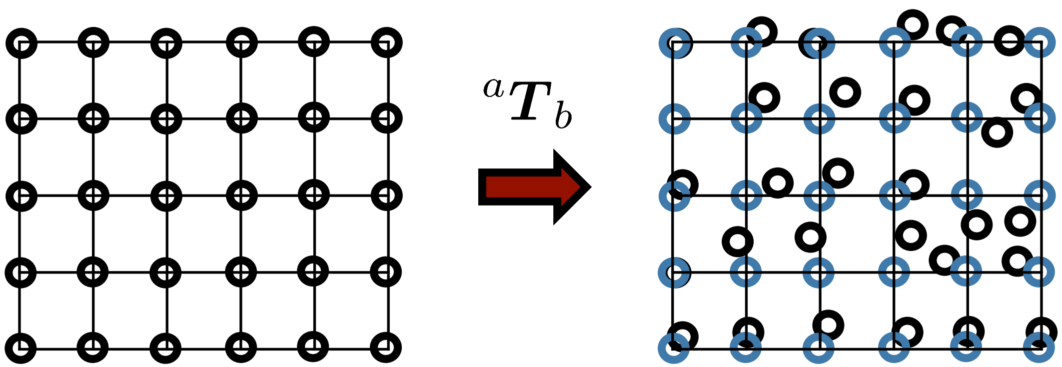
a output

Resampling – Forward warping

Forward warping:

Compute for every pixel in the input image b a value in the output a

$$\forall\ {}^b\boldsymbol{x} \qquad {}^a\boldsymbol{x} = {}^a\boldsymbol{T}_b(\,{}^b\boldsymbol{x})$$



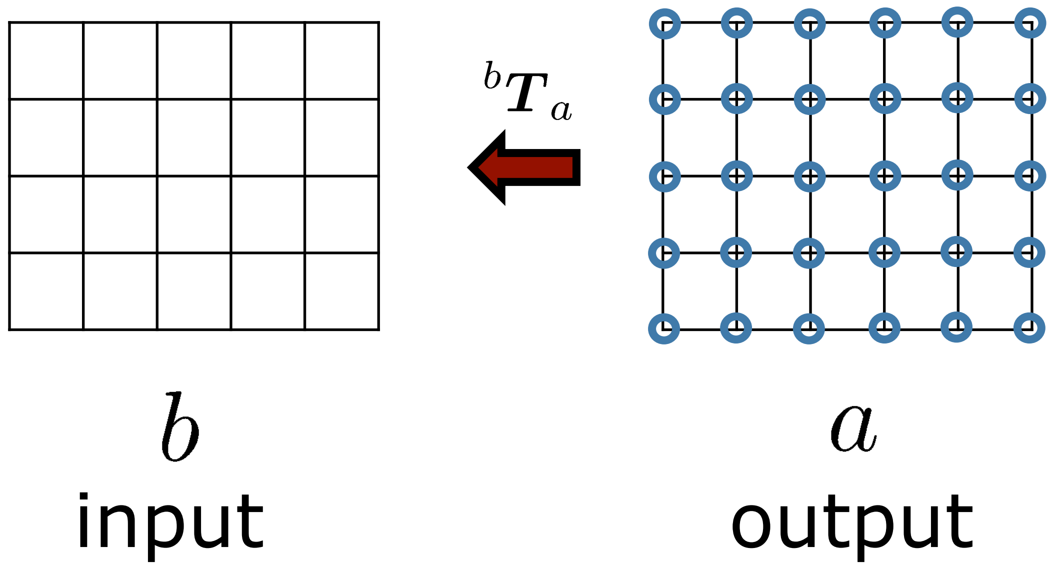
reconstruct the intensities at blue locations (**regular** grid) from the black ones (**irregular**)

Resampling – Inverse warping

Inverse warping:

Compute for every pixel in the output image a a value based on the input b

$$\forall\ {}^a\boldsymbol{x} \qquad {}^b\boldsymbol{x} = (\ {}^a\boldsymbol{T}_b)^{-1}\,(\ {}^a\boldsymbol{x}) = \ {}^b\boldsymbol{T}_a\,(\ {}^a\boldsymbol{x})$$

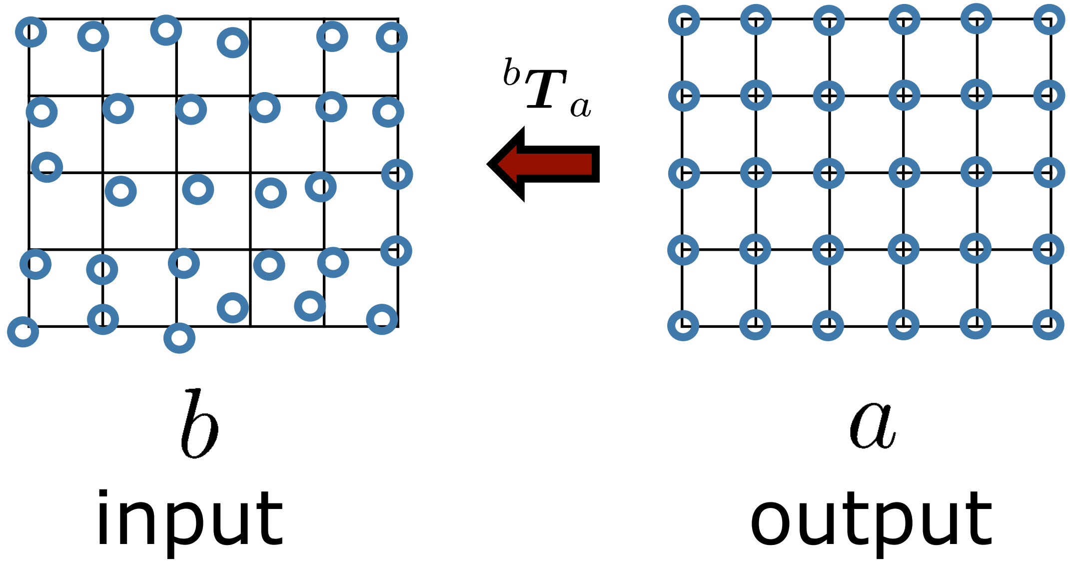


Resampling – Inverse warping

Inverse warping:

Compute for every pixel in the output image a a value based on the input b

$$\forall\ {}^a\boldsymbol{x} \qquad {}^b\boldsymbol{x} = {}^b\boldsymbol{T}_a\left({}^a\boldsymbol{x}\right)$$

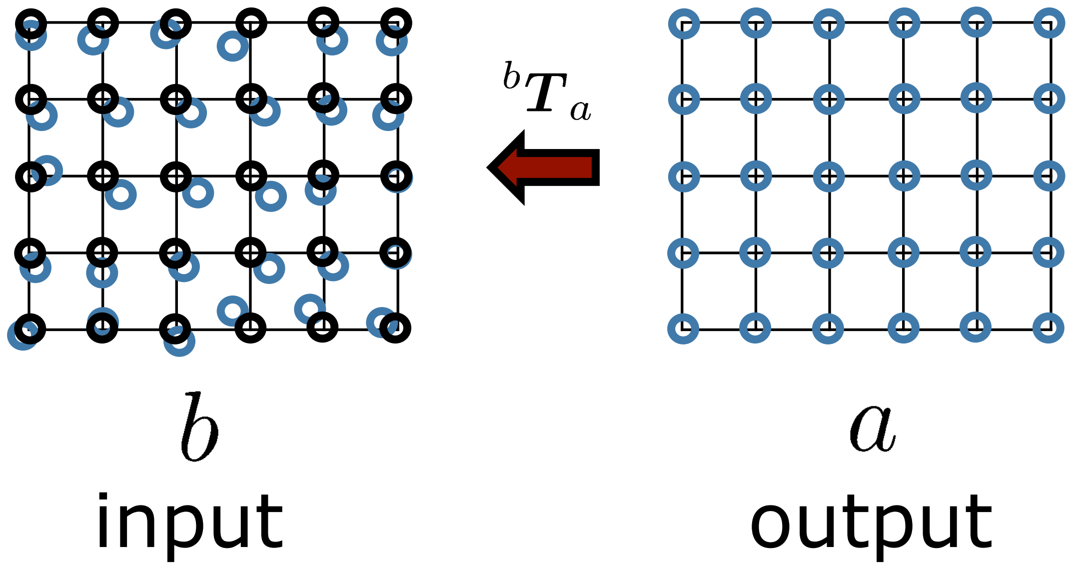


Resampling – Inverse warping

Inverse warping:

Compute for every pixel in the output image a a value based on the input b

$$\forall\ {}^a\boldsymbol{x} \quad {}^b\boldsymbol{x} = {}^b\boldsymbol{T}_a({}^a\boldsymbol{x})$$

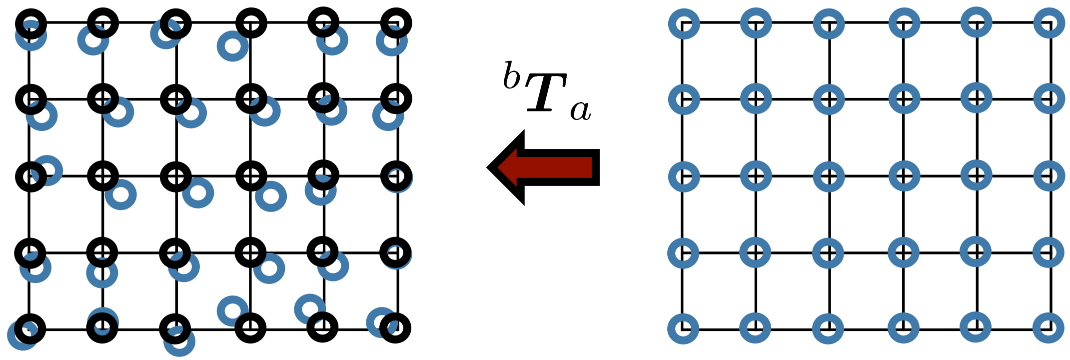


Resampling – Inverse warping

Inverse warping:

Compute for every pixel in the output image a a value based on the input b

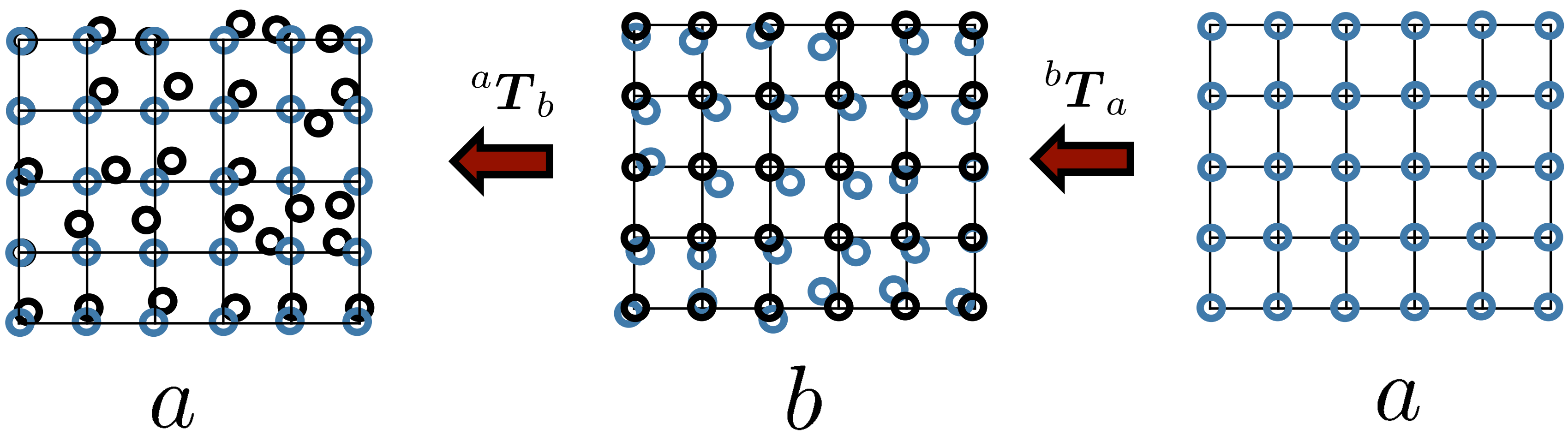
$$\forall\ {}^a\boldsymbol{x} \qquad {}^b\boldsymbol{x} = {}^b\boldsymbol{T}_a\left({}^a\boldsymbol{x}\right)$$



reconstruct the intensities at blue locations (**irregular**) from the black ones (**regular grid**)

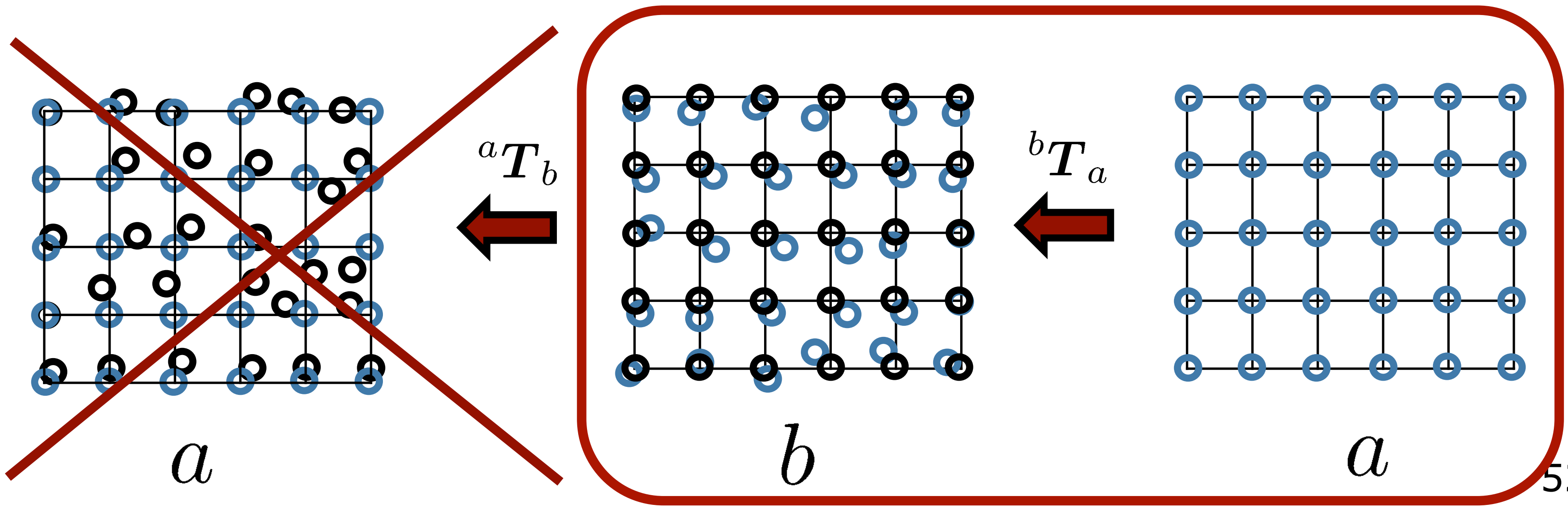
Resampling

- The forward approach can lead to missing pixels in the output image
- Inverse method allows for the direct application of bilin./cubic interpolation
- **Always use inverse warping!**



Resampling

- The forward approach can lead to missing pixels in the output image
- Inverse method allows for the direct application of bilin./cubic interpolation
- **Always use inverse warping!**



Complex Warping Example (1)

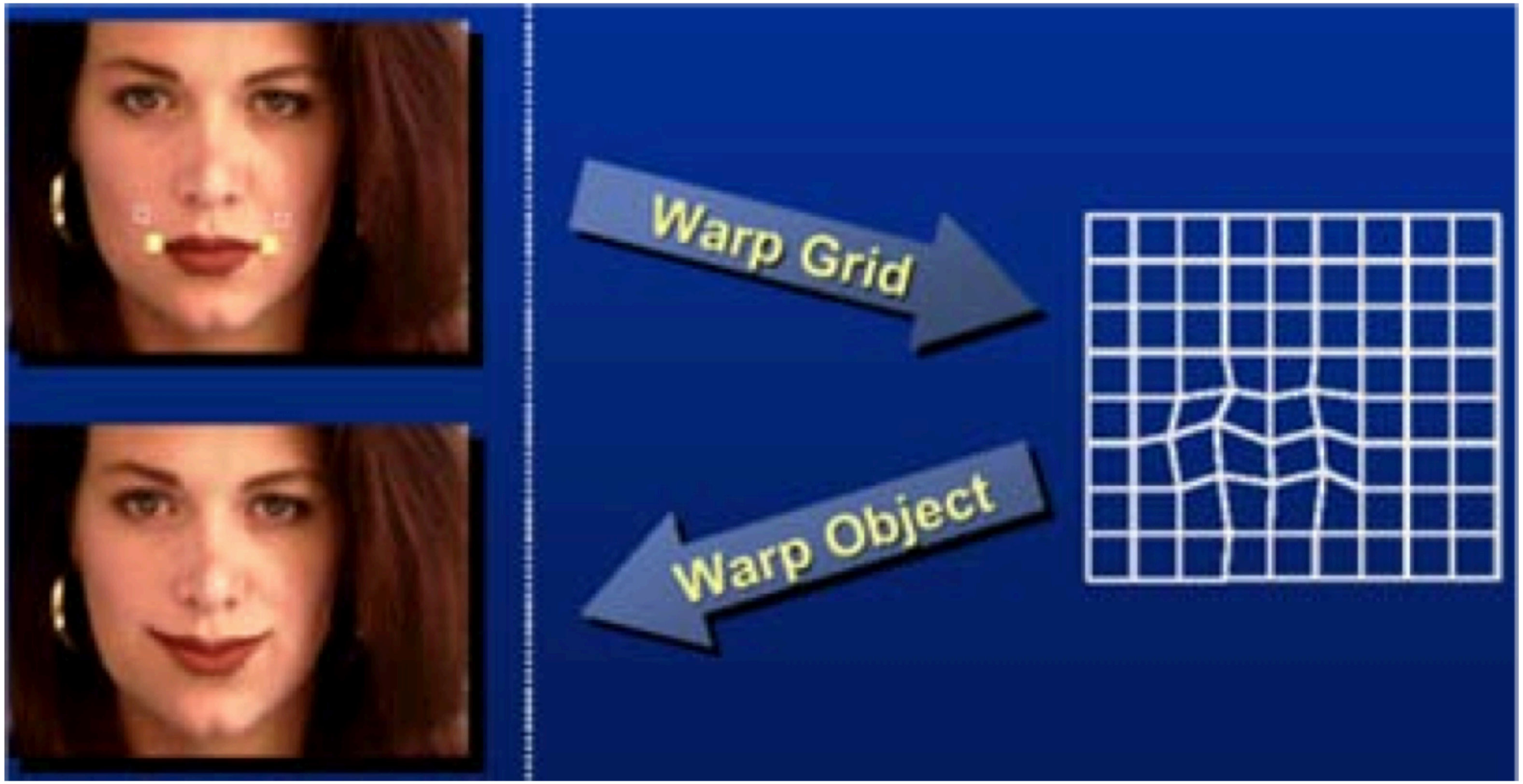
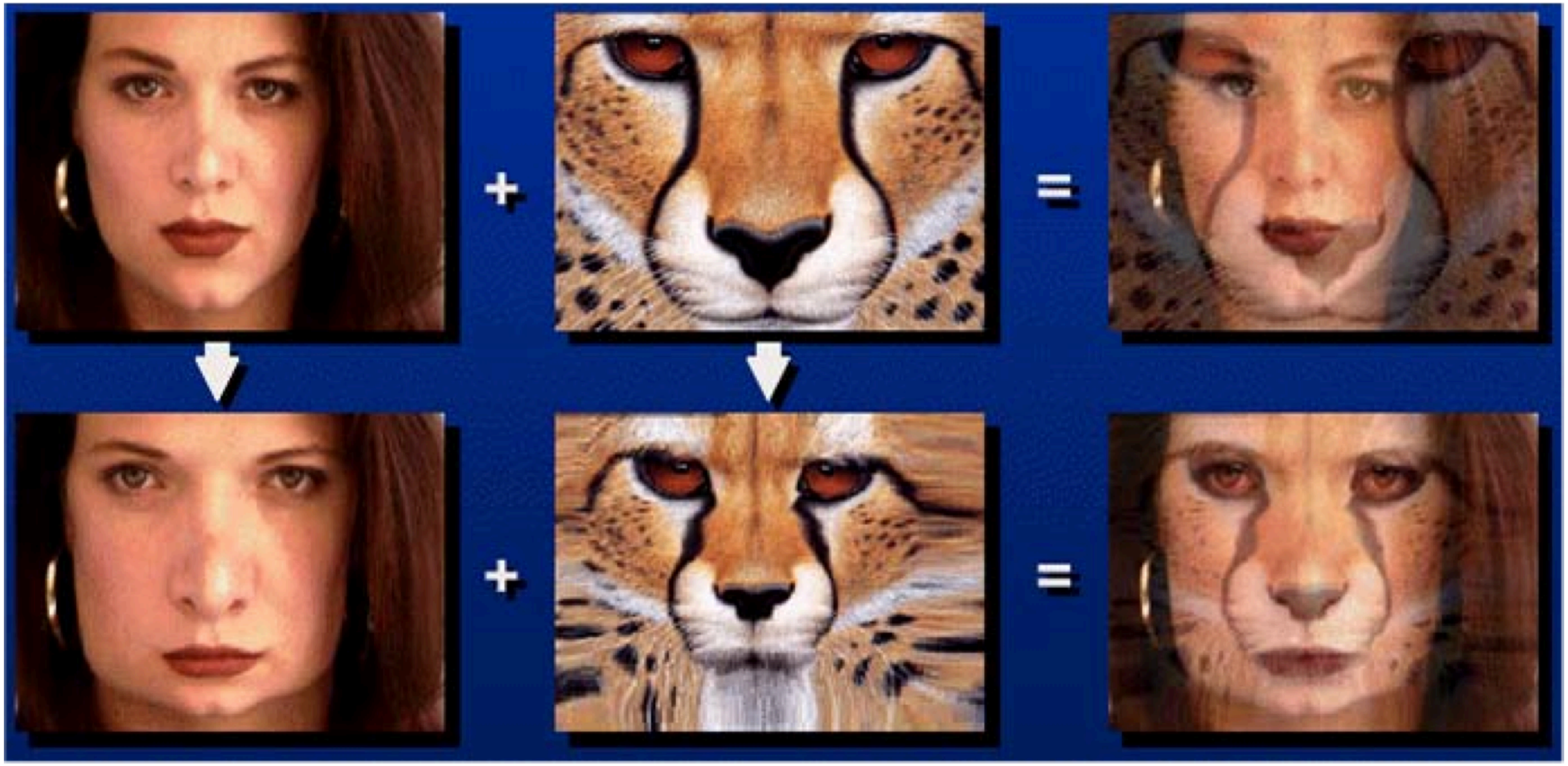


Image courtesy: Szeliski

Complex Warping Example (2)



top: simple blending; bottom: warping and blending

Image courtesy: Szeliski

Alignment for the Average Face

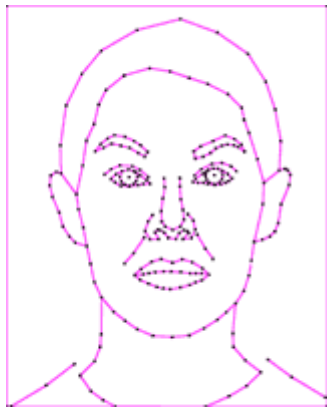


Image courtesy: Beautycheck.de (Gründl, Marberger, Braun, Scherber)

Alignment for the Average Face

avg. female

avg. male

See: http://www.beautycheck.de

Image courtesy: Beautycheck.de (Gründl, Marberger, Braun, Scherber)

Virtual Miss Germany (2002)

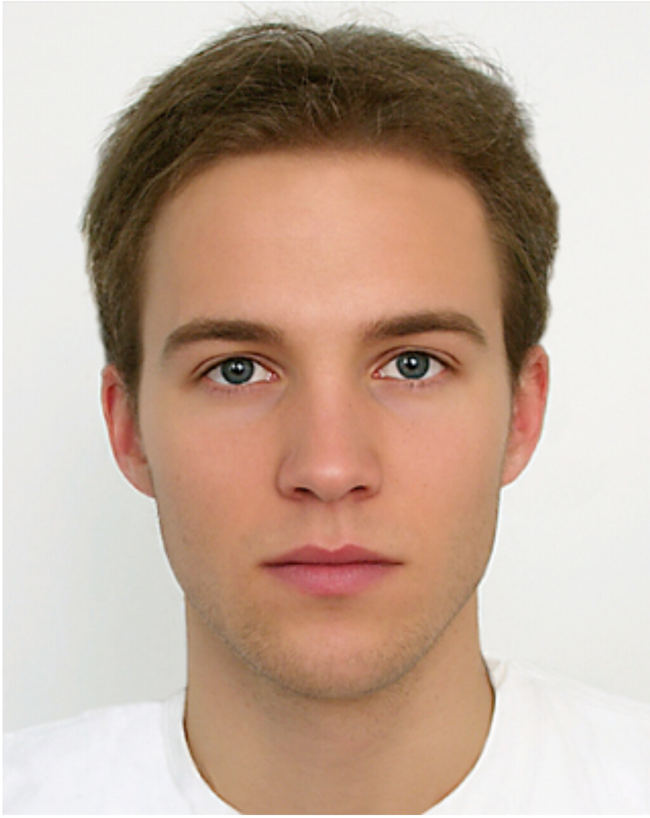
Image courtesy: Beautycheck.de (Gründl, Marberger, Braun, Scherber)

Virtual Miss Germany (2007)

Image courtesy: Beautycheck.de (Gründl, Marberger, Braun, Scherber)

Deriving a Beautiful Formula...

generated female



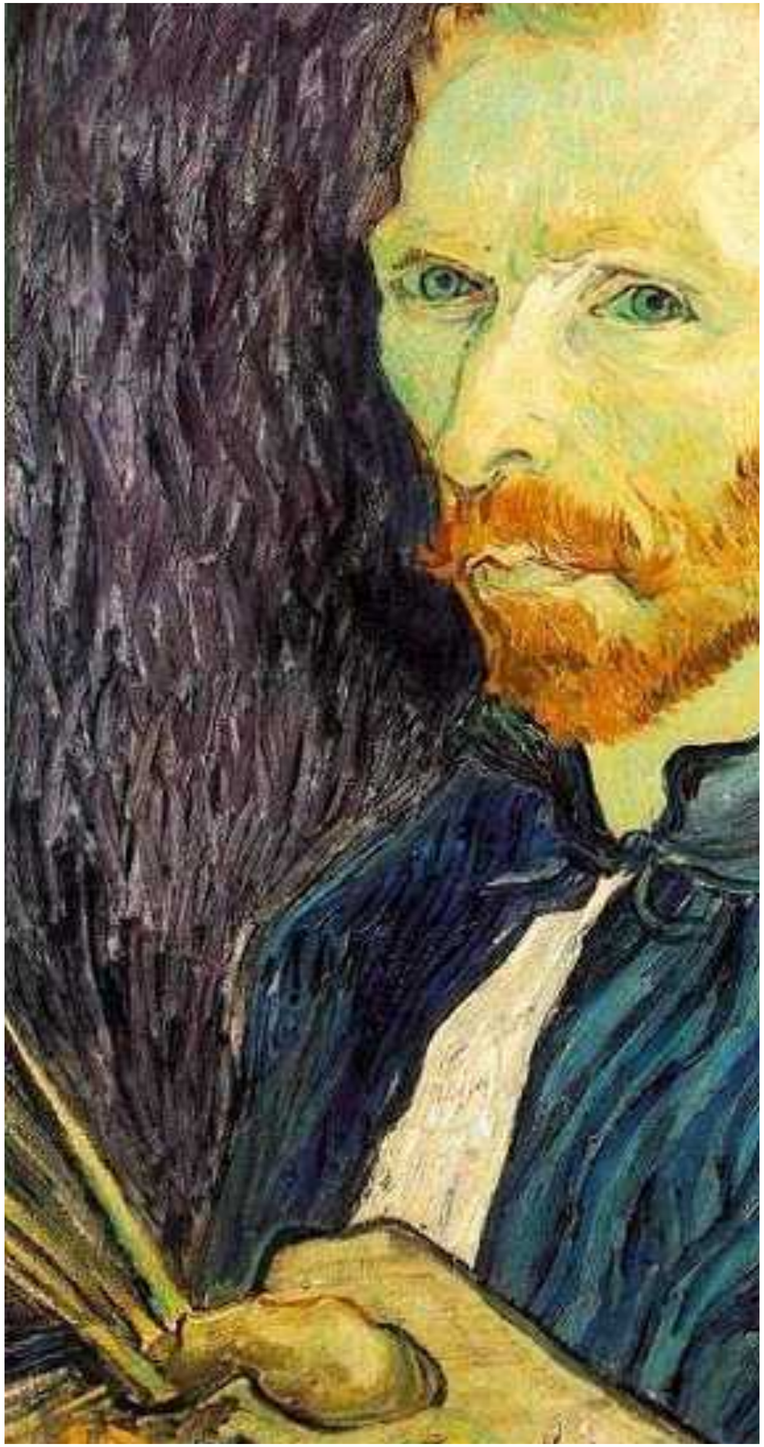
generated male

See: http://www.beautycheck.de

Image courtesy: Beautycheck.de (Gründl, Marberger, Braun, Scherber)

Image Half-Sizing

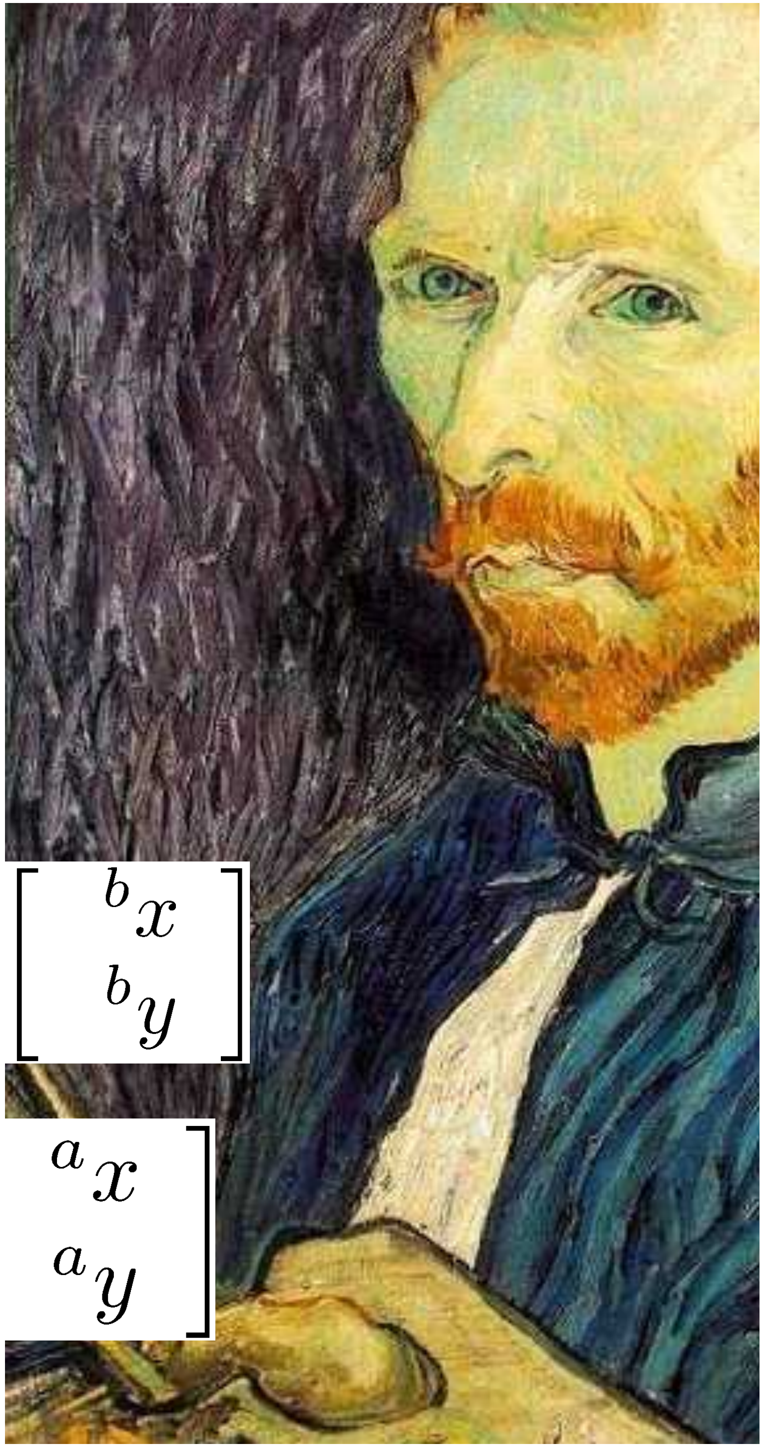
- This image is too big (to fit on the screen)
- How can we reduce it?
- How to generate a half-sized version?



Slide courtesy: Seitz

Image Half-Sizing

- This image is too big (to fit on the screen)
- How can we reduce it?
- How to generate a half-sized version?

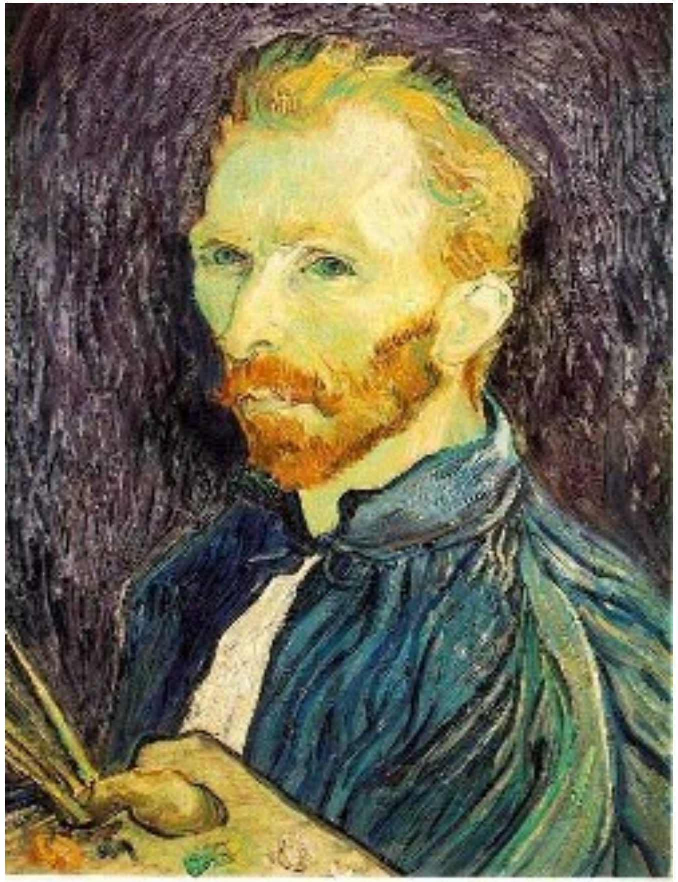


$$\begin{bmatrix} {}^{a}x \\ {}^{a}y \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} {}^{b}x \\ {}^{b}y \end{bmatrix}$$

$$\begin{bmatrix} {}^{b}x \\ {}^{b}y \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} {}^{a}x \\ {}^{a}y \end{bmatrix}$$

Slide courtesy: Seitz

Image Subsampling

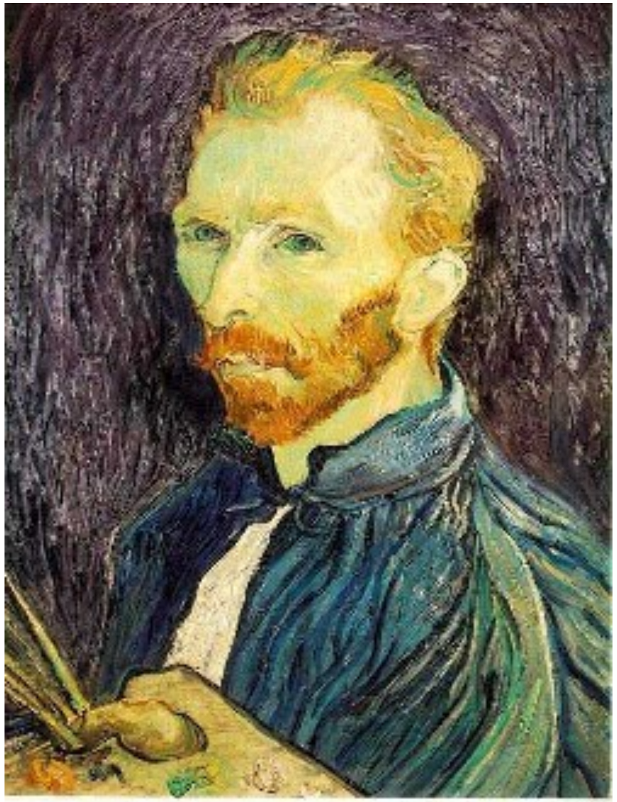


1/4

1/8

Throw away every other row and column to create a half-sized image.

Image Subsampling



1/2

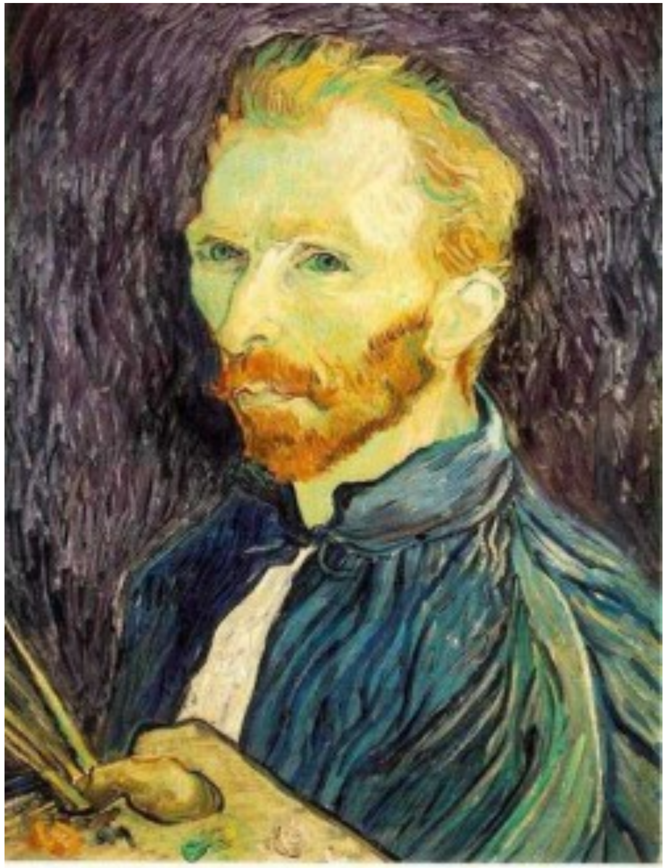
1/4 (2x zoom)

1/8 (4x zoom)

Simple subsampling → aliasing artifacts

Slide courtesy: Seitz

Apply a Binomial Filter Before Subsampling



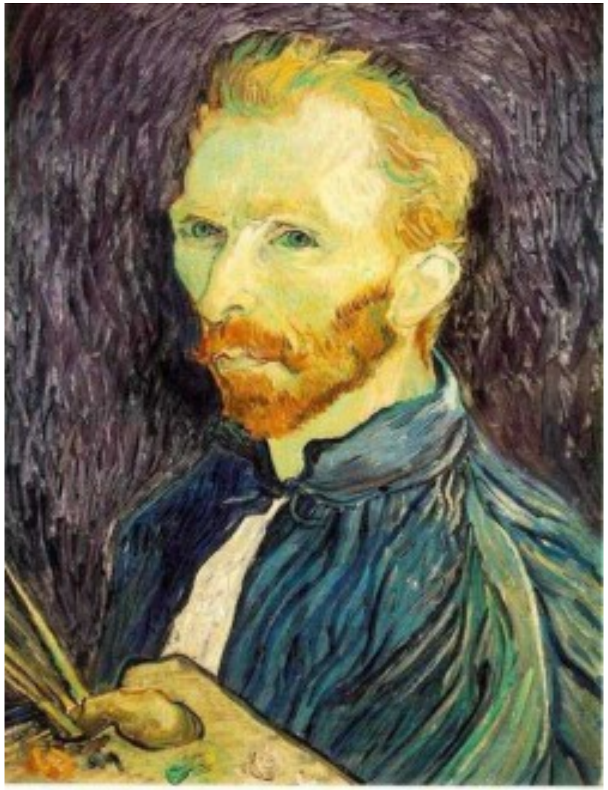
Binomial & 1/2

B & 1/4

B & 1/8

Slide courtesy: Seitz

Subsampling with Binomial Pre-Filtering

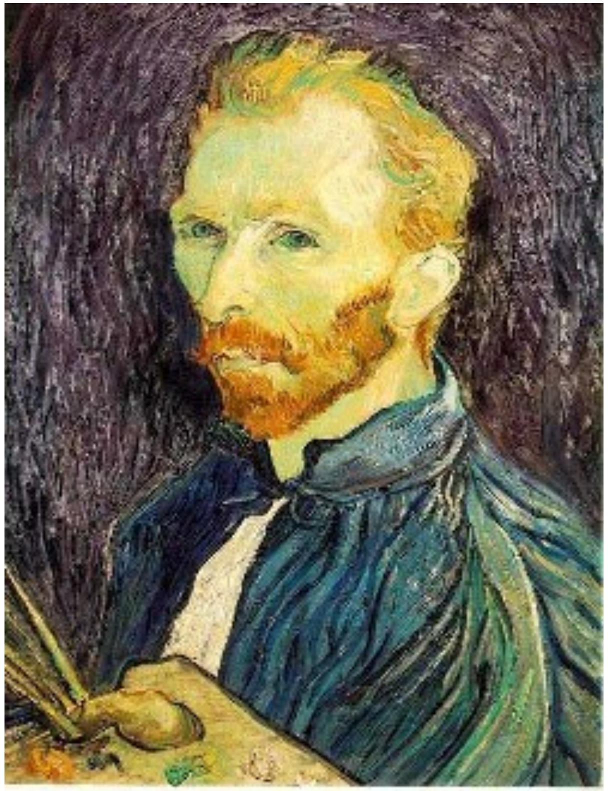


Binomial & 1/2

Binomial & 1/4

Binomial & 1/8

Compare with...



1/2

1/4 (2x zoom)

1/8 (4x zoom)

Slide courtesy: Seitz

Why is a Smoothing Step Needed?

- Simple subsampling results in aliasing and loosing details
- Smoothing combines pixel information from neighboring pixels

$$\boldsymbol{B}_2^{(2)} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & \underline{4} & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

How Much Smoothing is Needed?

- Depends on the kernel
- Depends on the width of the kernel
- Depends on the scale of the transformation

Kernel Width

- Width of a kernel is given by its standard deviation:

$$\sigma = \left(\sum_i i^2 \, w(i) \right)^{\frac{1}{2}}$$

- For the box and Gaussian filter

$$\sigma_{\boldsymbol{R}_n} = \left(\frac{n^2 - 1}{12} \right)^{\frac{1}{2}} \qquad \sigma_{\boldsymbol{B}_n} = \left(\frac{n}{4} \right)^{\frac{1}{2}}$$

Widths for Box and Binomial Filter

$$\sigma_{\boldsymbol{R}_n} = \left(\frac{n^2 - 1}{12}\right)^{\frac{1}{2}} \qquad \sigma_{\boldsymbol{B}_n} = \left(\frac{n}{4}\right)^{\frac{1}{2}}$$

n	1	2	3	4	5	7	8	9	16
$\sigma_{\boldsymbol{R}_n}$	–	0.50	0.82	1.12	1.41	2.00	2.29	2.58	4.61
$\sigma_{\boldsymbol{B}_n}$	0.5	0.71	0.87	1.00	1.12	1.32	1.41	1.50	2.00

Scale of a Transformation

- Average local scale of a transformation

$$\boldsymbol{T}(\boldsymbol{x}) = [T_x(\boldsymbol{x}), T_y(\boldsymbol{x})]^{\mathsf{T}}$$

- is given by

$$m = \sqrt{\frac{1}{2} \left\| \frac{\partial \boldsymbol{T}}{\partial \boldsymbol{x}} \right\|^2}$$

- with

$$\frac{1}{2} \left\| \frac{\partial \boldsymbol{T}}{\partial \boldsymbol{x}} \right\|^2 = \frac{1}{2} \left(\left(\frac{\partial T_x}{\partial x} \right)^2 + \left(\frac{\partial T_x}{\partial y} \right)^2 + \left(\frac{\partial T_y}{\partial x} \right)^2 + \left(\frac{\partial T_y}{\partial y} \right)^2 \right)$$

Scale Example

- Consider the transformation

$$\begin{bmatrix} T_x \\ T_y \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

- which yields a scale of

$$m = \sqrt{\frac{1}{2}(2^2 + 0^2 + 0^2 + 4^2)} = \sqrt{\frac{1}{2} \cdot 20} \approx 3.16$$

$$\left(\left(\frac{\partial T_x}{\partial x}\right)^2 + \left(\frac{\partial T_x}{\partial y}\right)^2 + \left(\frac{\partial T_y}{\partial x}\right)^2 + \left(\frac{\partial T_y}{\partial y}\right)^2 \right)$$

Consider the Scale for Resampling

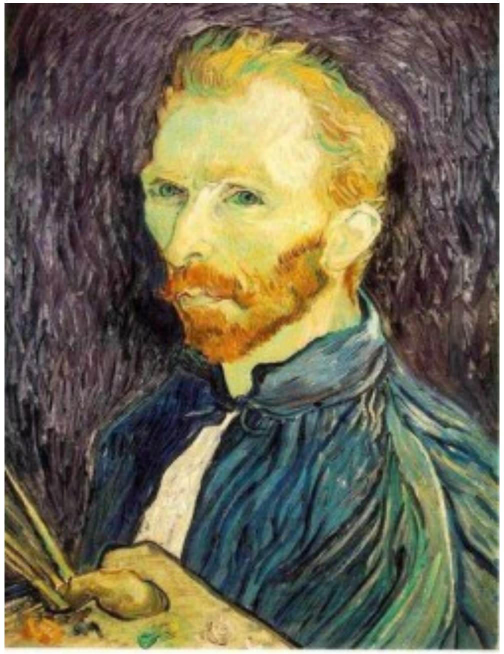
- m<1: **Image becomes smaller**
  Recommendation $\sigma \approx m/2$

- m=1: **Same scale**
  Use bilinear interpolation or bicubic for high quality results

- m>1: **Image becomes larger**
  Use bicubic interpolation

Image Pyramid

- A list of images
- Each image has half of the size of its predecessor



1 (original)

2 (size: 1/2)

3 (size: 1/4)

Image courtesy: Seitz

Image Pyramid

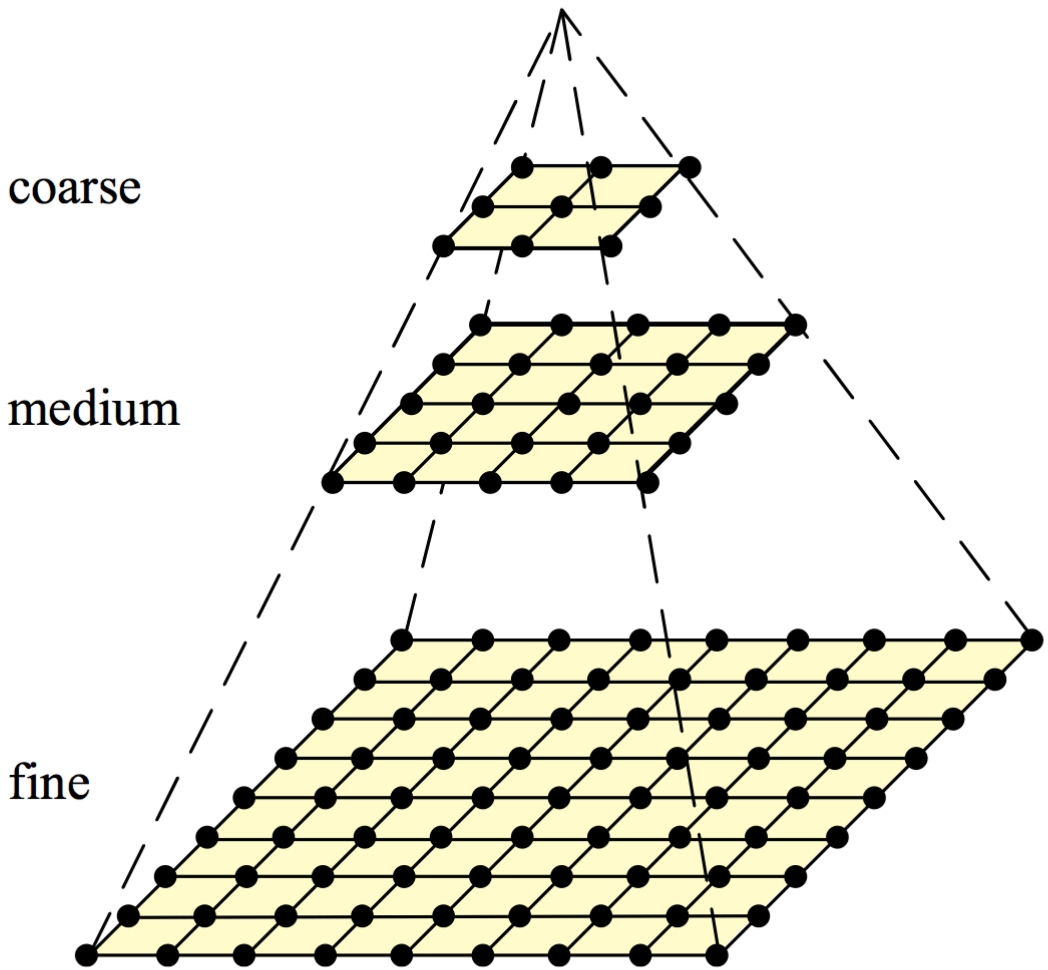


Image courtesy: Szeliski

Pyramid Example

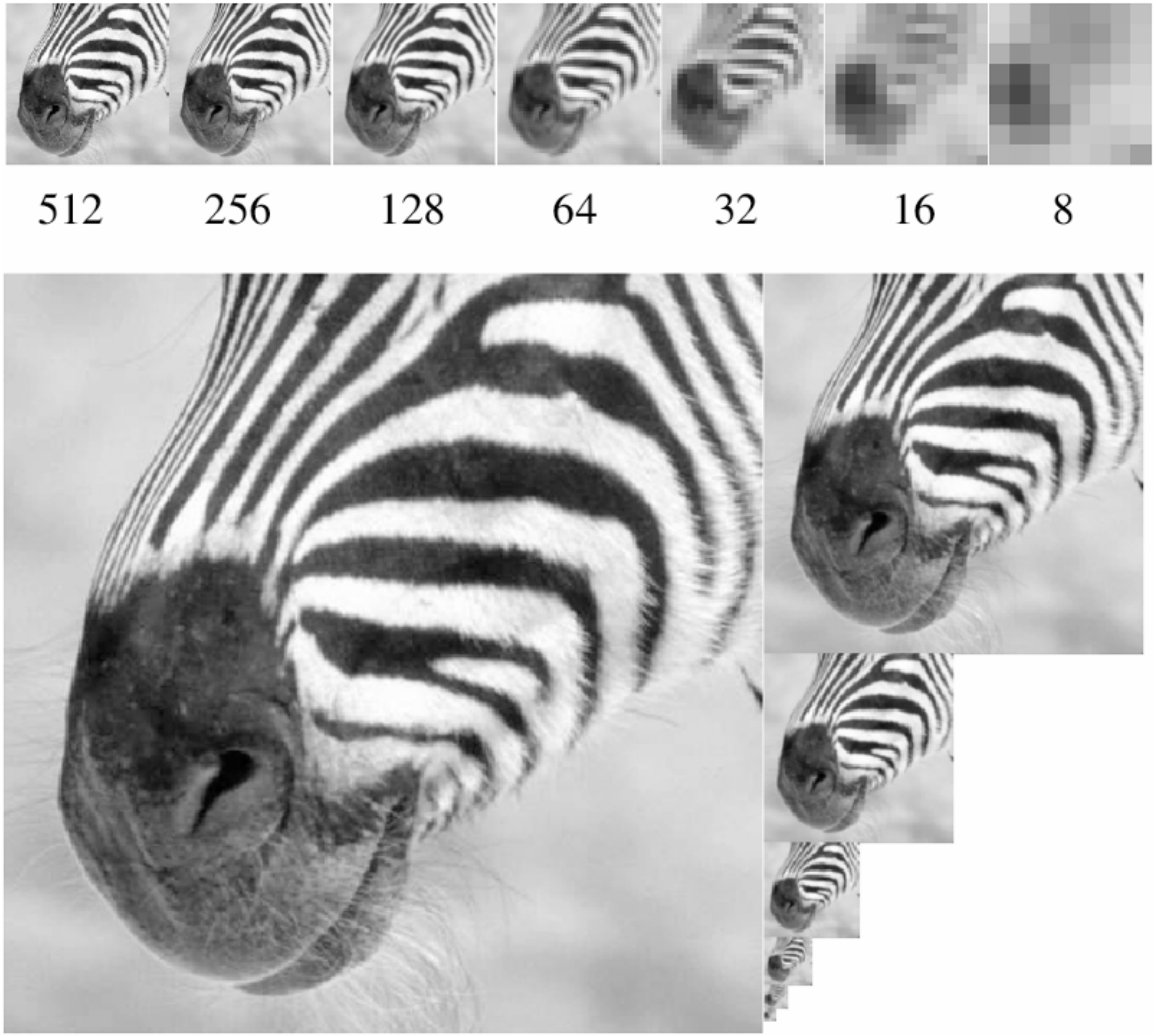


Image courtesy: Forsyth

Summary

- Geometric transformations of images
- Interpolation techniques
- Resampling
- Forward and inverse warping
- Image subsampling and smoothing
- Image pyramid

Literature

- Szeliski, Computer Vision: Algorithms and Applications, Chapter 3
- Förstner, Scriptum Photogrammetrie I, Chapter 10

Slide Information

- The slides have been created by Cyrill Stachniss as part of the photogrammetry and robotics courses.
- **I tried to acknowledge all people from whom I used images or videos. In case I made a mistake or missed someone, please let me know.**
- The photogrammetry material heavily relies on the very well written lecture notes by Wolfgang Förstner and the Photogrammetric Computer Vision book by Förstner & Wrobel.
- Parts of the robotics material stems from the great Probabilistic Robotics book by Thrun, Burgard and Fox.
- If you are a university lecturer, feel free to use the course material. If you adapt the course material, please make sure that you keep the acknowledgements to others and please acknowledge me as well. To satisfy my own curiosity, please send me email notice if you use my slides.

Cyrill Stachniss, cyrill.stachniss@igg.uni-bonn.de