Geometric transformations and **Interpolation** of images

Processing an image often implies also ***moving*** *its pixels*

- **Spatial geometric transformations**

In case of *geometric distortions*: an inverse transformation can bring each pixel to its correct position (generally not on the original sampling grid).

Two steps: 1. spatial transformation 2. interpolation

E.g.: correct lens distortions; remove perspective effects for image matching and retrieval; ... or warp an image or morph a face

- **Image scaling**

Matches *image size* to the one of the display (or a portion thereof)

E.g.: YouTube (640x480, 360p, ...) or SDTV (576) to HD (1920x1080p); photocamera e.g. 4000x3000 to HD

Interpolation changes the *sampling rate* (better, the *sampling grid*) of the image, and is needed in both cases

Geometric transformations

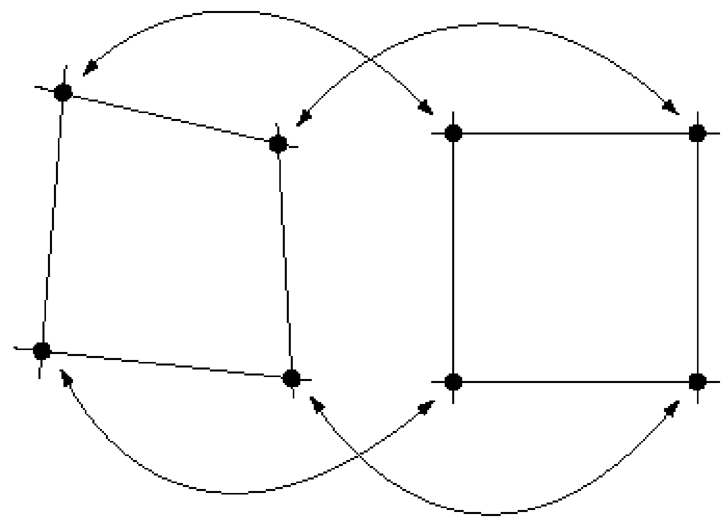
A geometric distortion may be represented by a coordinate mapping:

$$x'=r(x,y),\ y'=s(x,y).$$

If r and s are analytically available (e.g. known 3-D scene structure, known lens distortion), we can obtain an estimate $\hat{f}(x,y)$ of the ideal image $f(x,y)$ from the distorted image $g(x',y')$.

A single description for the whole image plane is generally not suitable.
→ Finite elements approach

1. A set of reference pixels (*tiepoints*) whose correct location can be determined is looked for
2. image is segmented into polygons
3. all pixels in each polygon are moved accordingly.



Corresponding tiepoints in two image segments.

Geometric transformations

A typical model used to describe geometric distortions is the **bilinear** one

$$x' = r(x,y) = a_r x + b_r y + c_r xy + d_r$$

$$y' = s(x,y) = a_s x + b_s y + c_s xy + d_s$$

The system has eight d.o.f.. If four pairs of tiepoints are known, it can be solved. Then, all the pixels in the enclosed quadrilateral region are transformed.

1: determine the distorted position (x',y') of each point (x,y) in the ideal image f via the eqs. above

2: set $\hat{f}(x,y)=g(x',y')$.

The point (x',y') in general will *not* be located on the sampling grid. We need an **interpolator**

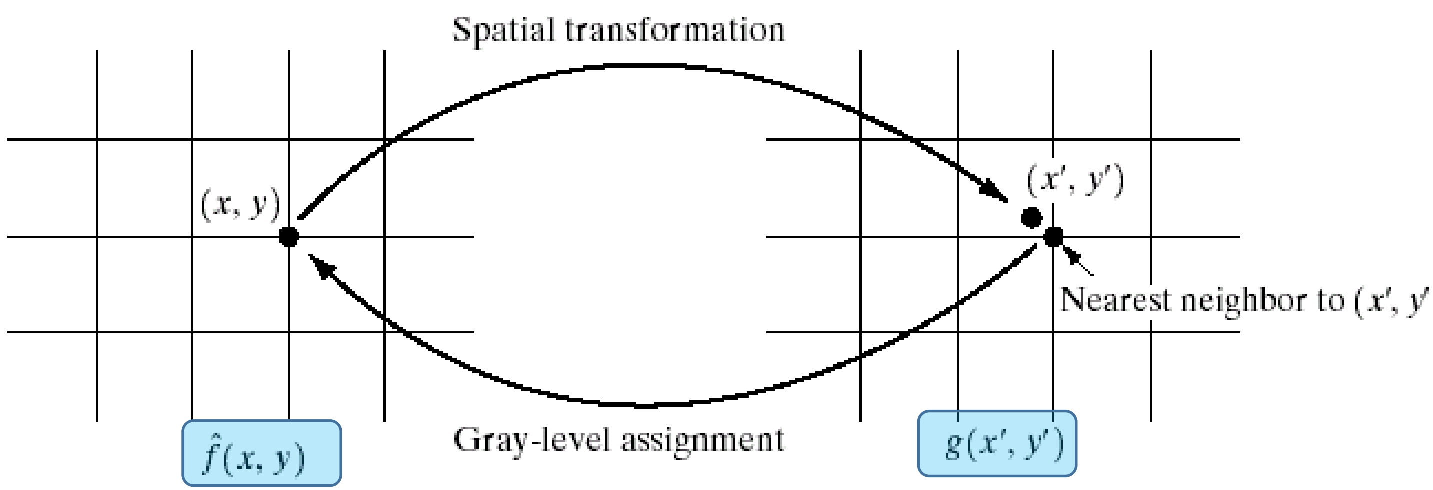


FIGURE 5.33 Gray-level interpolation based on the nearest neighbor concept.

Geometric transformations and Interpolation of images

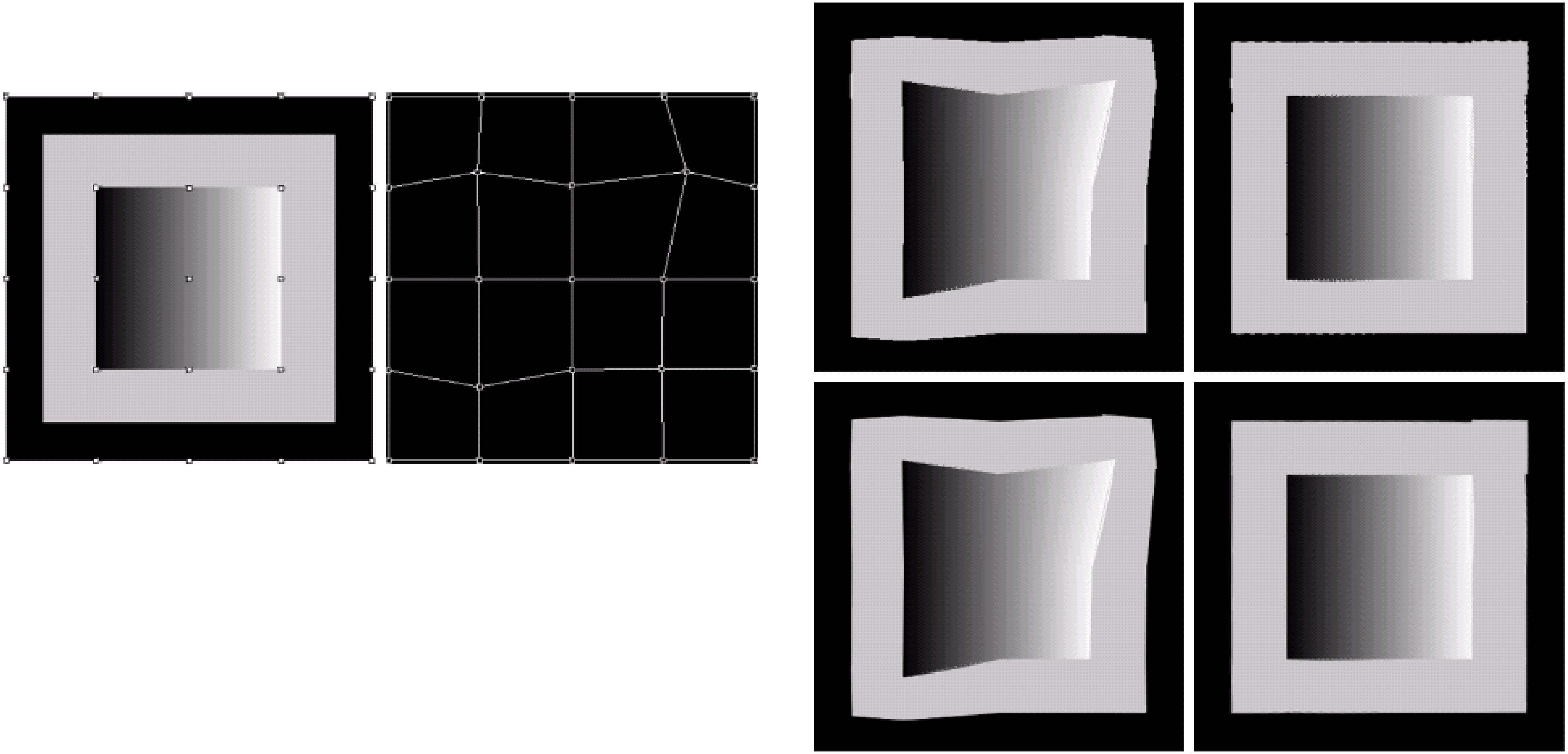


FIGURE 5.34 (a) Image showing tiepoints. (b) Tiepoints after geometric distortion. (c) Geometrically distorted image, using nearest neighbor interpolation. (d) Restored result. (e) Image distorted using bilinear interpolation. (f) Restored image.

Note: **bilinear** here has a different meaning !!!

Geometric transformations and Interpolation of images

Perspective correction

[Russ]

Figure 4.66 Correcting trapezoidal distortion due to a nonperpendicular view.

Notes:

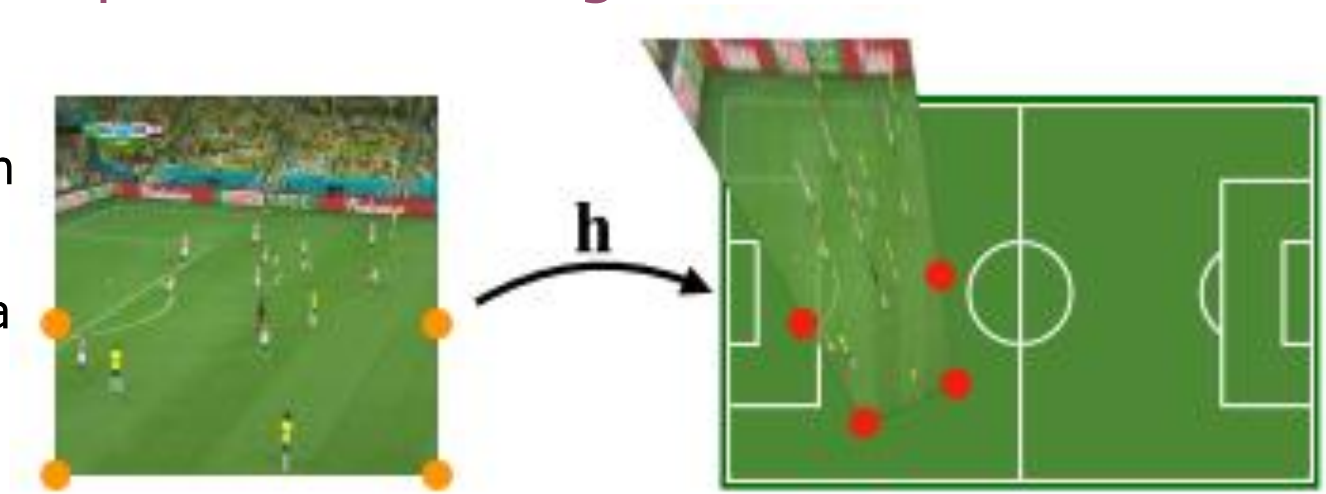
1. features that do not lie on the tilted planar surface are distorted (the light pole appears to be at an angle to the building when in fact it is perpendicular).

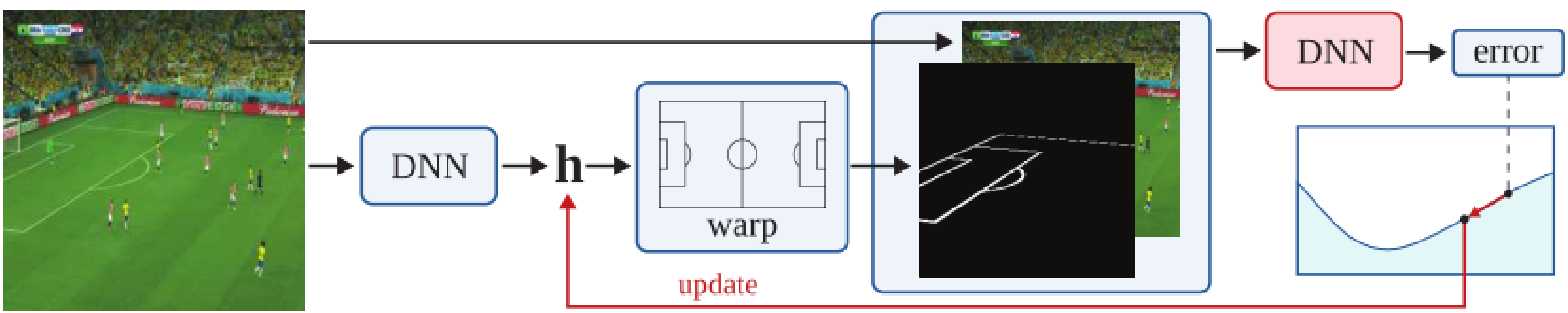
2. the right portion of the building is blurred due to enlargement and interpolation

Geometric transformations and Interpolation of images

Perspective correction

A more complex problem: determine a homography for sports video analysis





Jiang20_Accurate Sports Field Registration

Geometric transformations and Interpolation of images

A simpler tool: the **affine** transformation (six params.)

$x' = r(x,y) = a11\ x + a12\ y + b1$

$y' = s(x,y) = a21\ x + a22\ y + b2$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} + B$$

has the *collinearity* constraint: points on a line stay on a line, parallel lines remain parallel. Can be used to perform

translation ------------------------------→ $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad B = \begin{bmatrix} kc \\ kr \end{bmatrix}$

rotation (counter-clockwise) ---------→ $A = \begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) \\ \sin(\vartheta) & \cos(\vartheta) \end{bmatrix}$

scaling (possibly anisotropic) --------→ $A = \begin{bmatrix} kc & 0 \\ 0 & kr \end{bmatrix}$

shearing (e.g., parallel to *y*) ---------→ $A = \begin{bmatrix} 1 & kr \\ 0 & 1 \end{bmatrix}$

[columns (*x*) shift proportionally to rows (*y*)]

Geometric transformations and Interpolation of images

Homogeneous notation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} + B \quad \Rightarrow \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a11 & a12 & b1 \\ a21 & a22 & b2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \Rightarrow \quad P_{fin} = A\, P_{ini}$$

- To perform the transformation, for each position P_{fin} in output image find position P_{ini} in input image: $P_{ini} = round(A^{-1}\, P_{fin})$, and read gray level there. Note '*round*' → nearest neighbour interpolation

- To find *A* from tiepoints: 6 d.o.f. → 3 pairs of eqs. like the one above:

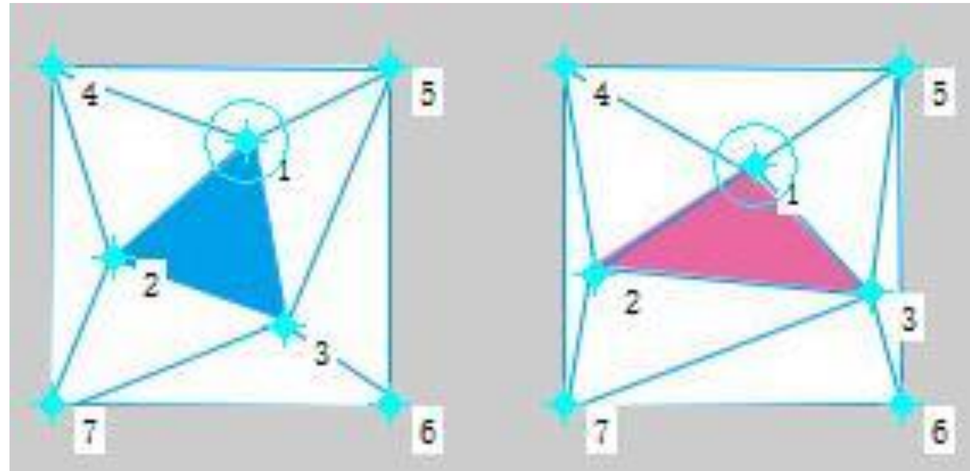
$$\begin{bmatrix} x1' & x2' & x3' \\ y1' & y2' & y3' \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} a11 & a12 & b1 \\ a21 & a22 & b2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 & x2 & x3 \\ y1 & y2 & y3 \\ 1 & 1 & 1 \end{bmatrix} \quad \Rightarrow \quad P_{fin} = A\, P_{ini} \quad \Rightarrow \quad A = P_{fin} P_{ini}^{-1}$$

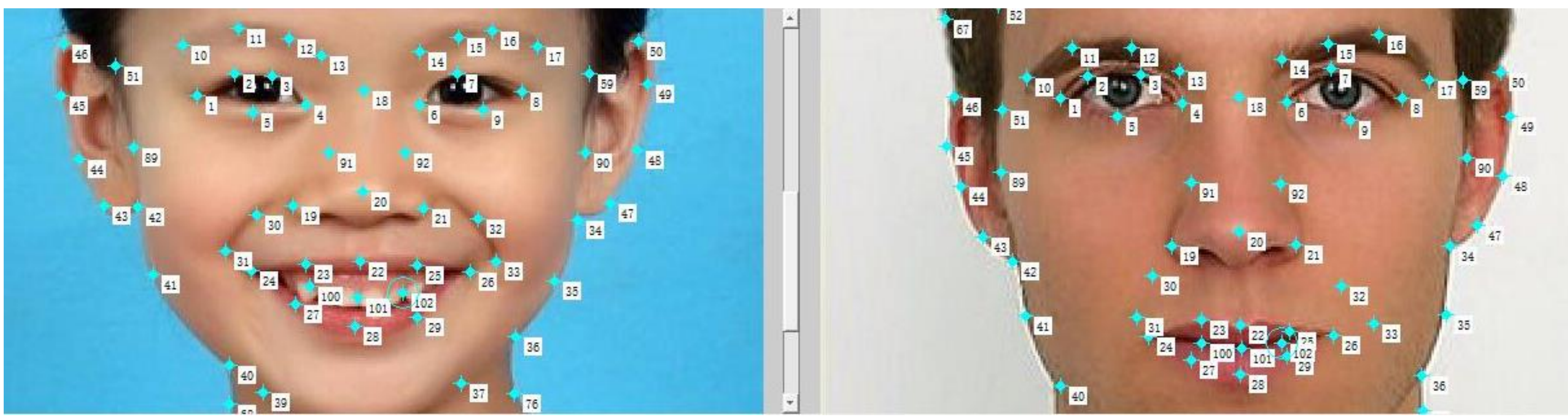
If no. of tiepoints >3, use the *pseudoinverse* to find the MMSE solution

[matlab**]**

Affine transformation: an application

Image morphing
using Delaunay triangulation





https://hypjudy.github.io/2017/04/25/image-morphing/

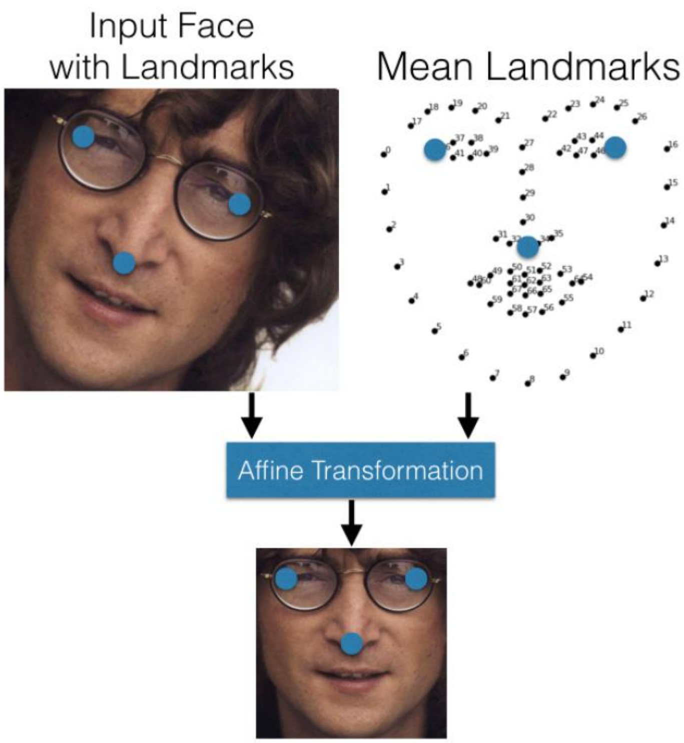
Affine transformation: an application

Preprocessing for DL-based face recognition

Face detection returns a list of bounding boxes around the faces in an image.

- Faces could be looking in different directions or under different illumination.

- To reduce the size of the input space, normalize the faces: eyes, nose, and mouth should appear at similar locations in each image.



- *DeepFace* (Facebook-Meta) (2014) frontalizes the face to a 3D model to make the image appear as looking directly towards the camera.
- *FaceNet* (Google) → *OpenFace* (CMU 2016, open source) uses a simpler 2D affine transformation (→ *OpenFace 2.2* for further tasks)

68 landmarks are detected with *dlib*'s face landmark detector. The affine transformation also resizes and crops the image to the edges of the landmarks so the input image to the neural network is 96 × 96 pixels.

Interpolation of images

Image upscaling: an *apparently simple* case of geom. transformation

In principle, image acquisition violates the sampling theorem: the real world has "infinite" spatial-frequency contents

→ Ideal sinc-based interpolation is not the theoretically right choice, apart from not being realizable

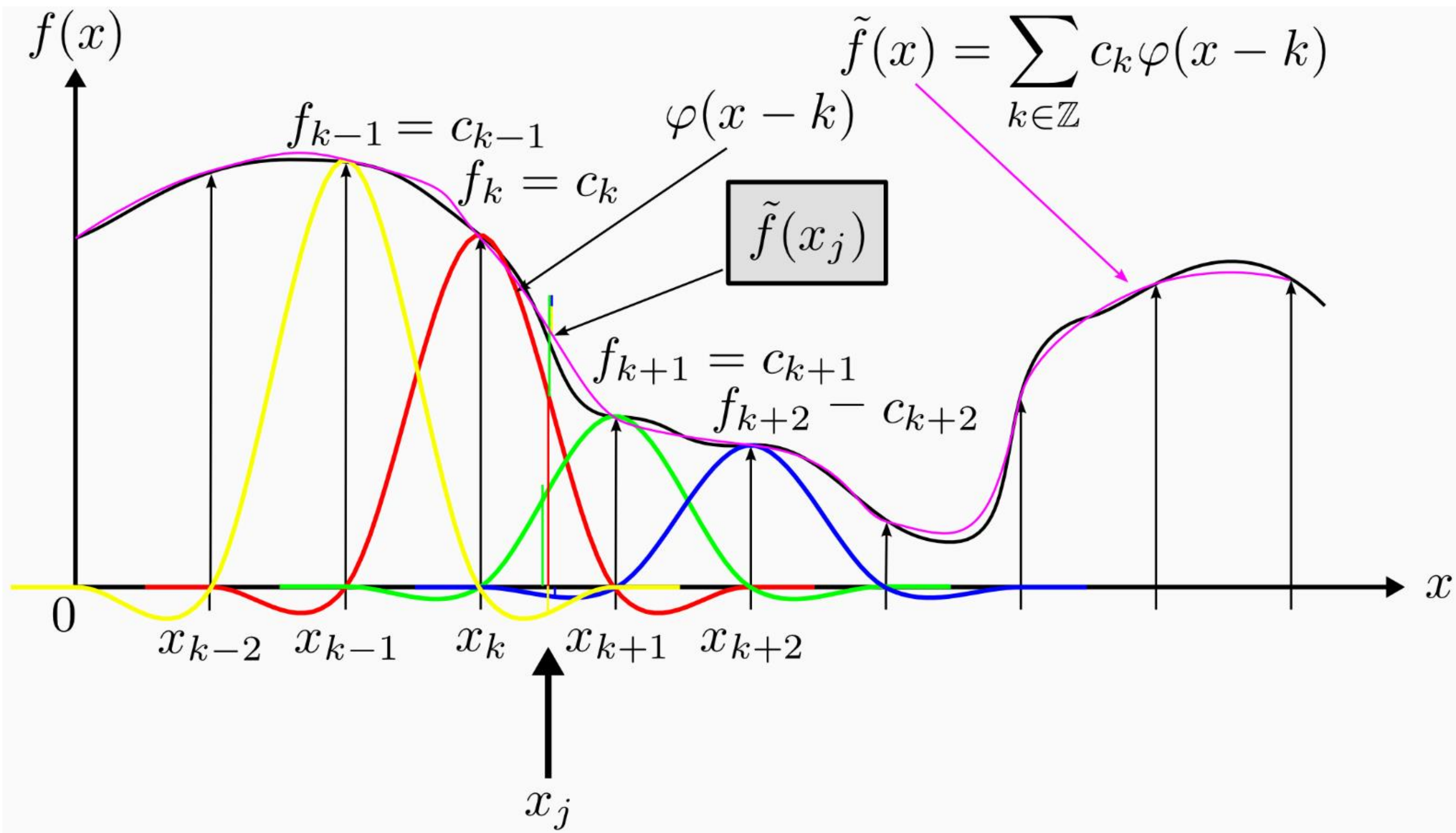
The visual system is the final reference, and has very high resolution (someone says several hundred Mpixel if we account for saccadic movements too) (http://clarkvision.com/imagedetail/eye-resolution.html)

We wish we could (re-)introduce the *correct* higher frequency contents

Interpolation of images

Shannon interpolation for Nyquist-frequency sampled **1-D** data

The **ideal** LP filter has rectangular mask with cutoff freq. at $\Omega_T/2$ → impulse response is $\varphi(x)$ = sinc(x/T). Interpolation is obtained by the **convolution** between the sampled data and $\varphi(x)$



Interpolation of images

[Pratt 113]**[dip07_2.m]**

Simplest solution: *local n-th order* interpolators approximate the sinc. Infinite convolution becomes *linear combination of a few adjacent samples*. *Weights* are a function of sample *distance* x. They can be derived by repeated convolutions of the *hold* function $R_0(x)$ with itself:

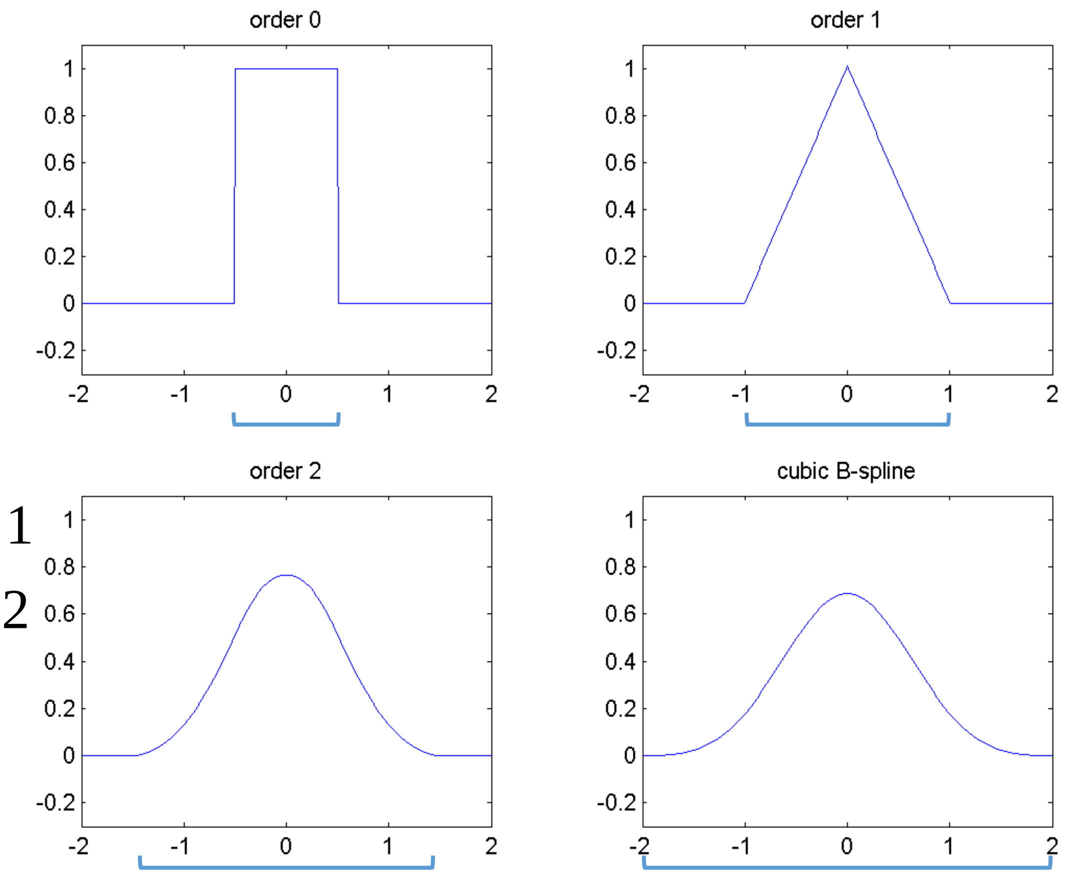
$$R_0(x) = \begin{cases} 1 & -1/2 \le x < 1/2 \\ 0 & \text{elsewhere} \end{cases}$$

$$R_1(x) = \begin{cases} x+1 & -1 \le x \le 0 \\ 1-x & 0 \le x \le 1 \end{cases}$$

...

$$R_3(x) = \begin{cases} \frac{2}{3} + \frac{1}{2}|x|^3 - x^2 & 0 \le |x| \le 1 \\ \frac{1}{6}(2-|x|)^3 & 1 \le |x| \le 2 \end{cases}$$

(cubic B − spline)



Interpolation of images

Note: some of these are *function approximators* rather than *interpolators*: the reconstructed function is not constrained to pass exactly *on* the original points. Better use the:

Keys interpolator

An often used 3rd-order interpolator based on the general expression:

$$R_C(x) = \begin{cases} A|x|^3 + Bx^2 + C|x| + D & 0 \le |x| \le 1 \\ E|x|^3 + Fx^2 + G|x| + H & 1 \le |x| \le 2 \end{cases}$$

with constraints: *R*=1 at 0, *R*=0 at -2,-1,1,2 (like a sinc), and null first derivative at -2,2 → 7 constraints, 8 unknowns → 1 d.o.f.:

$$R_C(x) = \begin{cases} (A+2)|x|^3 - (A+3)x^2 + 1 & 0 \le |x| \le 1 \\ A|x|^3 - 5Ax^2 + 8A|x| - 4A & 1 \le |x| \le 2 \end{cases}$$

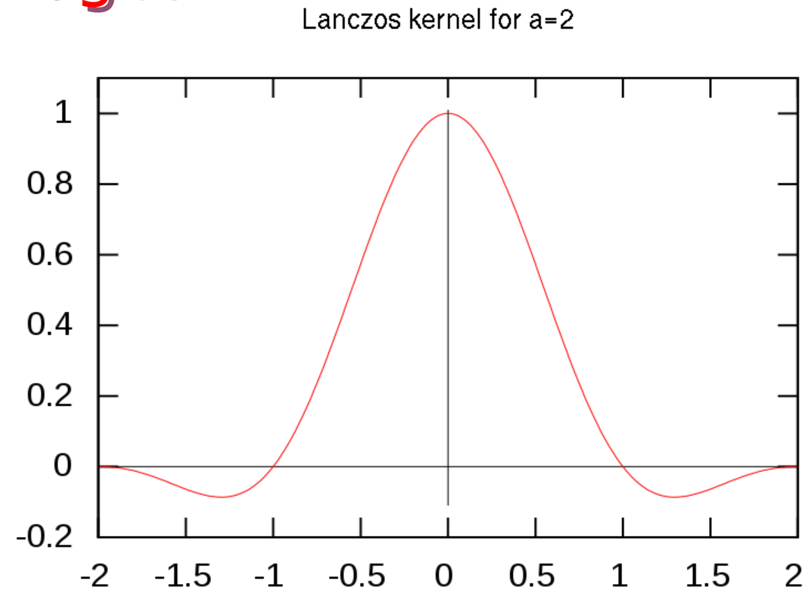
If *A*=-1/2 is selected, the solution achieved is the one with mmse for a power series expansion.

Interpolation of images

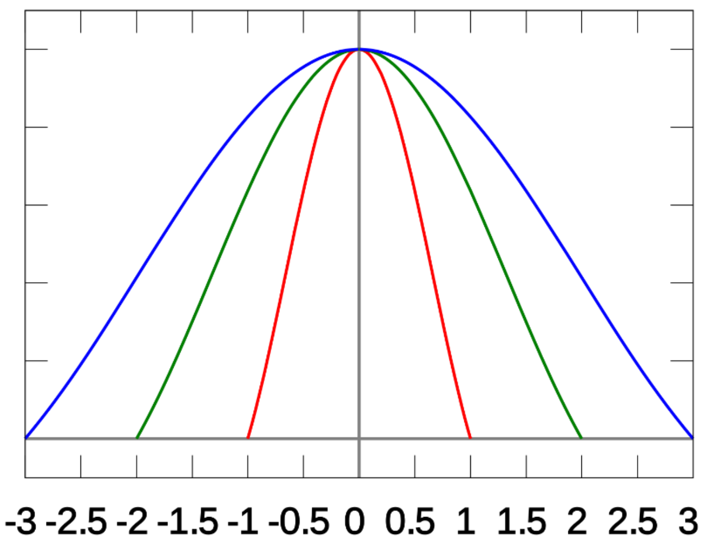
A generalization of the Keys interpolator:

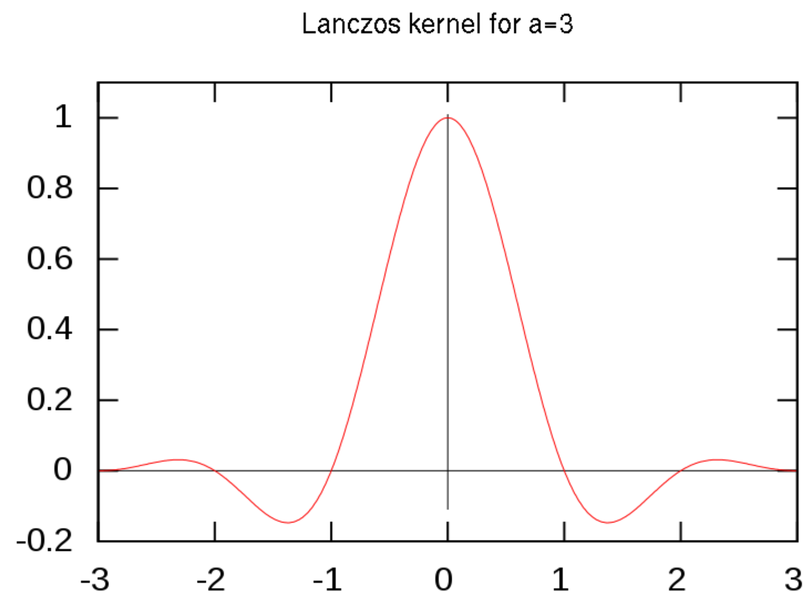
Lanczos interpolator

Coefficients are the values of a sinc windowed with the main lobe of another wider sinc



sinc(x/a) for $-a \le x \le a$ (a=1,2,3)





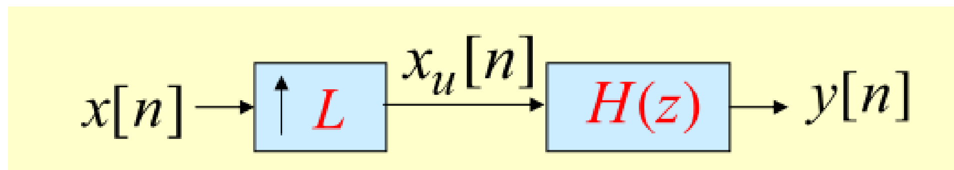
https://commons.wikimedia.org/w/index.php?curid=6897796

Interpolation of images

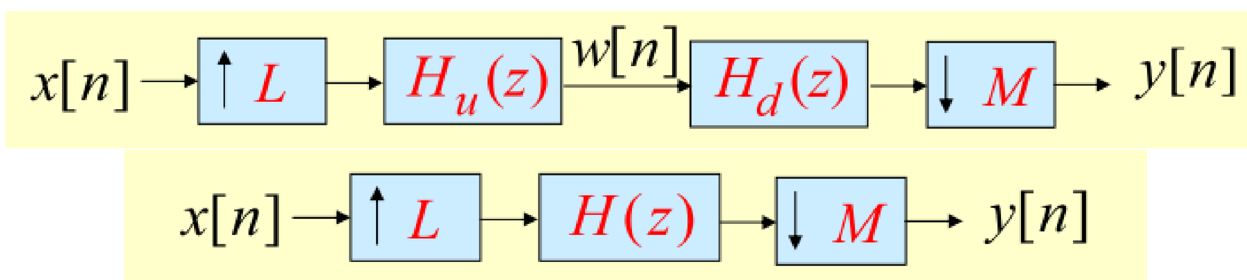
In the discrete-time domain we do not need *continuous* interpolated functions → perform **upsampling + linear anti-imaging filtering *H(z)***

H(z) may have same impulse response as R_0, R_1, R_3, R_C filters above.

For an integer factor ***L***:



For a rational factor ***L/M***, **antialias filtering + downsampling** is further needed:



Note: efficient **polyphase** realizations exist for interpolation.

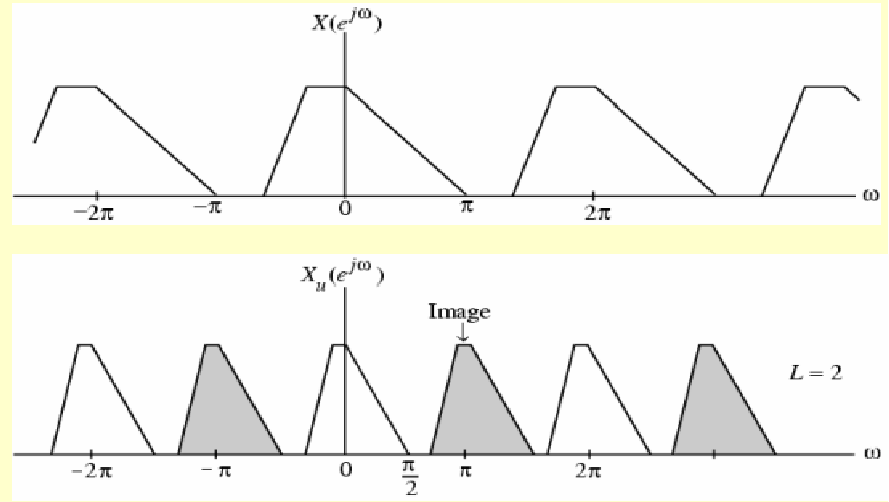
Interpolation of images

E.g., if $L=2$:

note asymmetric spectrum: signal is complex

$$X_u(z) = \sum_{n=-\infty}^{\infty} x_u[n] z^{-n} = \sum_{\substack{n=-\infty \\ n \text{ even}}}^{\infty} x[n/2] z^{-n}$$

$$= \sum_{m=-\infty}^{\infty} x[m] z^{-2m} = X(z^2)$$



For any L, an anti-imaging filter is needed with stopband edge at $\omega_s = \pi/L$ and passband edge at ω_p as close as possible to ω_s to avoid damaging the spectrum of the input signal.

2-D case: (separable) linear interpolators are derived:

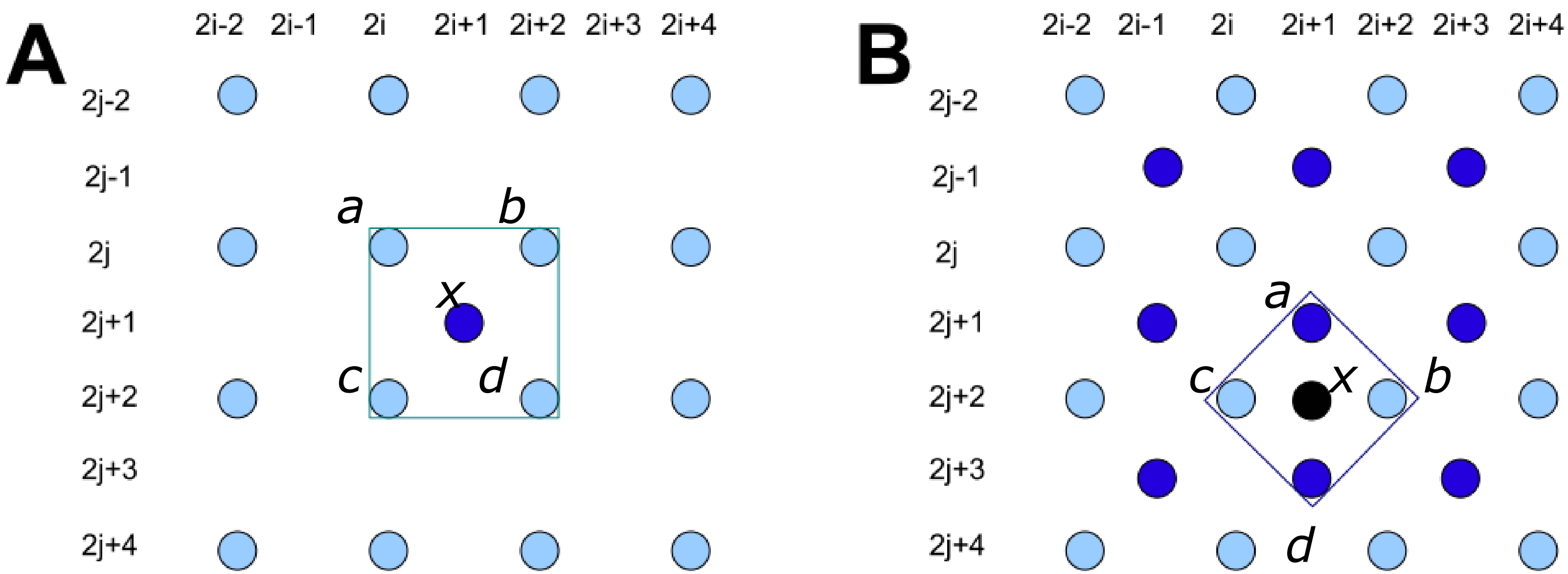
- *R0* → **nearest-neighbor**
- *R1* → **bilinear**
- Keys → **bicubic**
- → **Upsampling and 2-D anti-imaging filter**

[matlab]

Interpolation of images

Many **Nonlinear Edge-Directed** interpolation algorithms exist.

→ *iteratively duplicate* the image size by copying original pixels into an enlarged grid, then fill the gaps with weighted averages of neighbors, using weights derived by a local edge analysis.

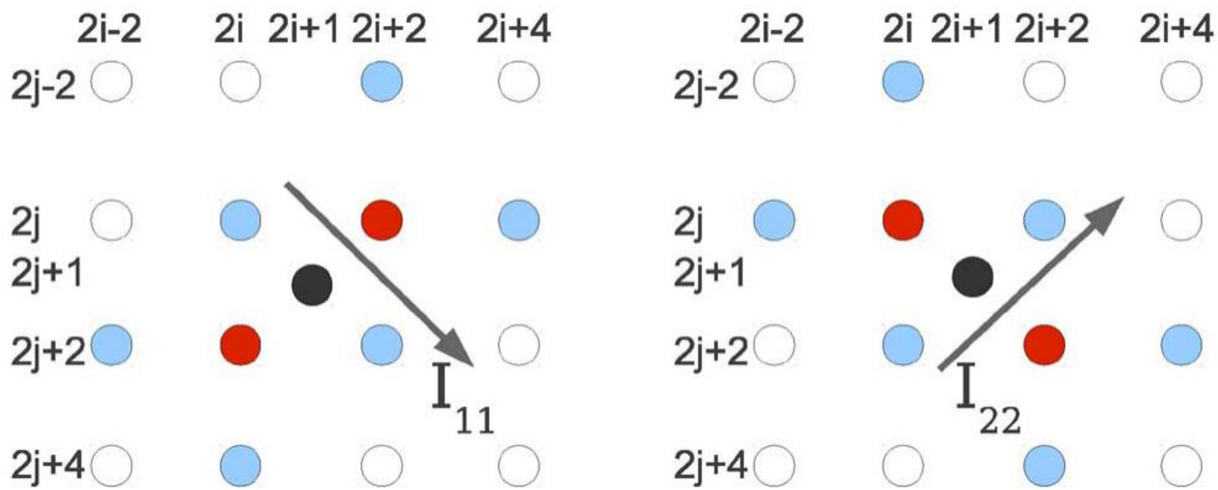


Simple solution: $x = (a+d)/2$ *if* $(|a-d| < |b-c|)$

$x = (b+c)/2$ *otherwise*

Interpolation of images

More sophisticated:



At each step, the algorithm fills the central pixel (black) with the average of the two neighbors in the direction of lowest second order derivative, I11 or I22. I11 (and I22) are estimated as:

$$\tilde{I}_{11}(2i+1,2j+1) = I(2i-2,2j+2) + I(2i,2j) +$$
$$+I(2i+2,2j-2) - 3I(2i,2j+2) - 3I(2i+2,2j) +$$
$$+I(2i,2j+4) + I(2i+2,2j+2) + I(2i+4,2j)$$

An iterative refinement is then performed

Li Orchard 2001
Giachetti 2011

Interpolation of images

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 45, NO. 5, MAY 2023 5461

Blind Image Super-Resolution: A Survey and Beyond

Anran Liu, Yihao Liu,

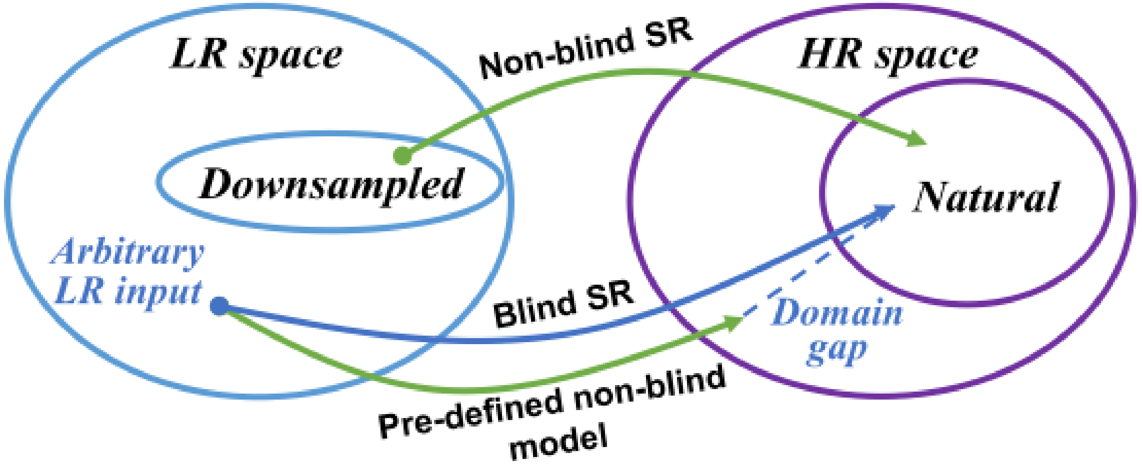


Fig. 2. Domain interpretation of differences between non-blind and blind SR. There exists a large domain gap between the SR result and desired high-quality HR, which is caused by applying a pre-trained non-blind model to LR input with degradation deviating from the assumed one (e.g., downsampling).

→ z07_Superresolution_Image_Reconstruction

Interpolation of images

A case of domain-specific superresolution:

Kriging: used in topography to estimate 3D surfaces from irregularly sampled terrain

It is quite apparent that *sinc* interpolation does not provide in general a realistic representation of terrain from sparse data

SPOTLIGHT

on Geostatistics — Professor D. G. Krige

by A. N. BROWN*

Professor D. G. Krige, of the Department of Mining Engineering of the University of the Witwatersrand, received the degree of Doctor of Engineering (*honoris causa*) from the University of Pretoria at the Autumn Graduation Ceremony in April 1981. He was honoured as an outstanding engineer for his world-renowned pioneering work in the field of statistical methods of ore valuation, or geostatistics as it has become known.

Early Life

Daniel Gerhardus Krige was born at Bothaville in the Orange Free State on 26th August, 1919. He completed his schooling at the Monument Hoërskool at Krugersdorp, the town at which his father, ds. J. J. Krige, was a well-known minister of the Nederduits Gereformeerde Kerk. He graduated as a mining engineer at the University of the Witwatersrand in 1939 at the age of 19.

Mining Experience

Through the next five years he received training and practical experience on various gold mines on the Witwatersrand, and in 1944 took up a post in the office of the Government Mining Engineer. One of his important duties during the eight years he spent in the Mines Department was to provide direct assistance to the Government Mining Engineer in preliminary and subsequent negotiations with the American and British authorities concerning the production of uranium. He was responsible for the design of the original uranium-pricing formula and other provisions incorporated in the uranium supply contracts negotiated in the period 1945

Witwatersrand for his pioneering work in this field.

In 1952 he took up an appointment with the Anglovaal mining group and progressed rapidly to the position of Group Financial Engineer. His research in the application of mathematical statistics to ore valuation continued unabated, as is evidenced by the forty or more scientific publications that stand to his credit over the years. These publications and papers have had a number of important consequences.

He is internationally known and acknowledged in mining circles throughout the world as one of the leading

https://desktop.arcgis.com/en/arcmap/latest/tools/3d-analyst-toolbox/how-kriging-works.htm

For generic images: → z07_interp_kriging

Interpolation of images

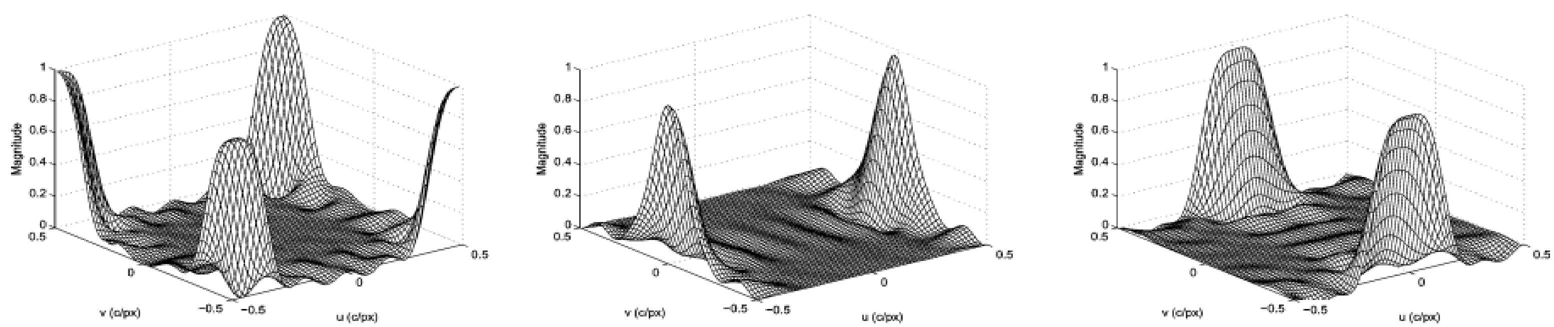
UNIVERSITÀ DEGLI STUDI DI TRIESTE

A case of domain-specific superresolution:

Bayer filter demosaicking

→ z07_Demosaicking_Color_image_demosaicking_An_overview

LEUNG *et al.*: LEAST-SQUARES LUMA–CHROMA DEMULTIPLEXING ALGORITHM FOR BAYER DEMOSAICKING



(a) (b) (c)

→ z07_Demosaicking_04 Sensors + demosaicing