ECE5554 – Computer Vision
Lecture 3c – Interpolation; Image Pyramids

Creed Jones, PhD

BRADLEY DEPARTMENT
OF ELECTRICAL & COMPUTER ENGINEERING

VIRGINIA TECH.

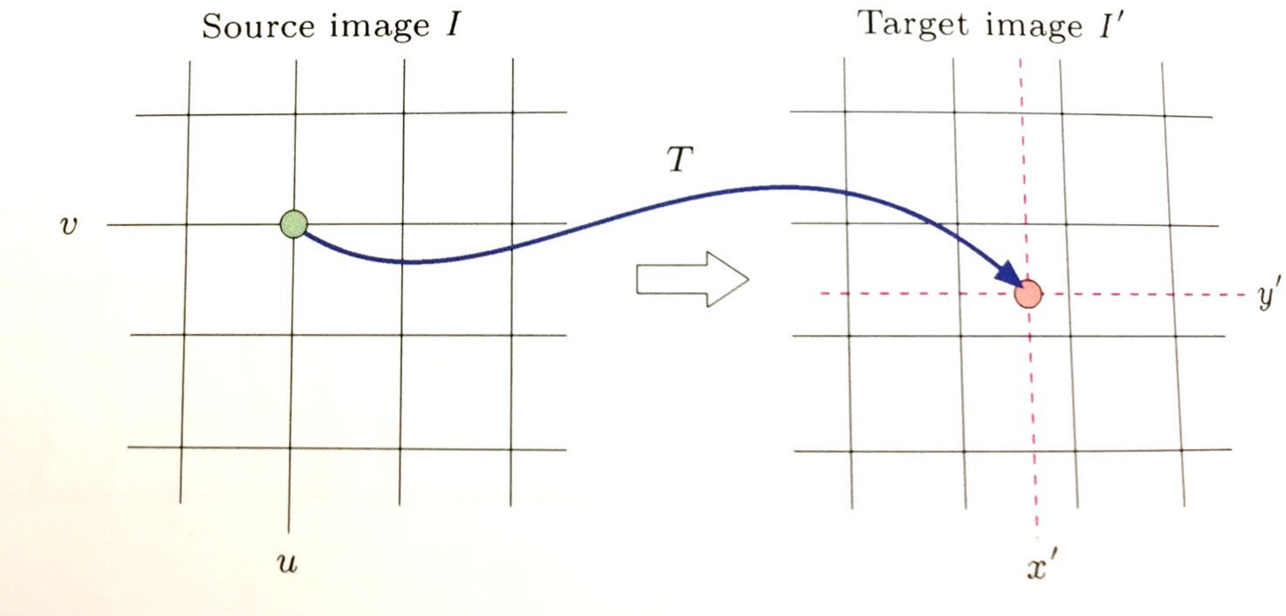
Today's Objectives

- Interpolation
  - Bilinear interpolation
  - Bicubic interpolation
  - Lanczos interpolation
  - Downsampling
- Multiresolution representations
  - Image pyramid concept
  - Gaussian pyramid
  - Laplacian pyramid

IMAGE INTERPOLATION

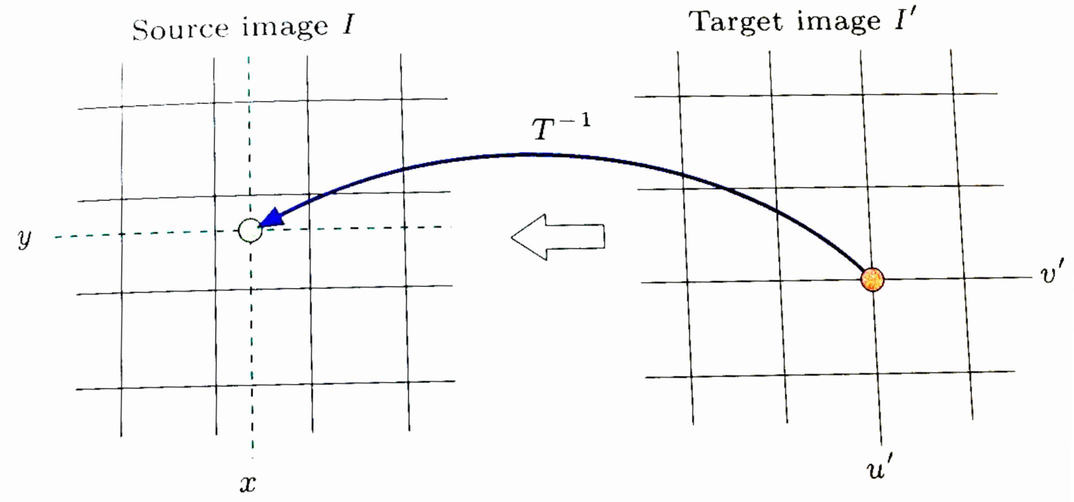
When doing Source to Target mapping, we examine each pixel value in the source image and determine where it should go in the target image

- Can result in voids in the target image (locations with no pixel values from the source image)



In Target to Source mapping, each pixel location in the target image is considered to determine which pixel value from the source image should go there

- Can result in duplicate values in the target image, or pixel values in the input image that are overwritten or not used



$$(x, y) = T^{-1}(u', v')$$

Consider target to source mapping; the complication arises when the transformation function dictates a point with non-integer coordinates

- We are scaling an image by 125% in both directions...
- $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1.25 & 0 \\ 0 & 1.25 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$
- If $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 15 \\ 18 \end{bmatrix}$, $\begin{bmatrix} x' \\ y' \end{bmatrix} \begin{bmatrix} 1.25 & 0 \\ 0 & 1.25 \end{bmatrix} \begin{bmatrix} 15 \\ 18 \end{bmatrix} = \begin{bmatrix} 18.75 \\ 22.5 \end{bmatrix}$

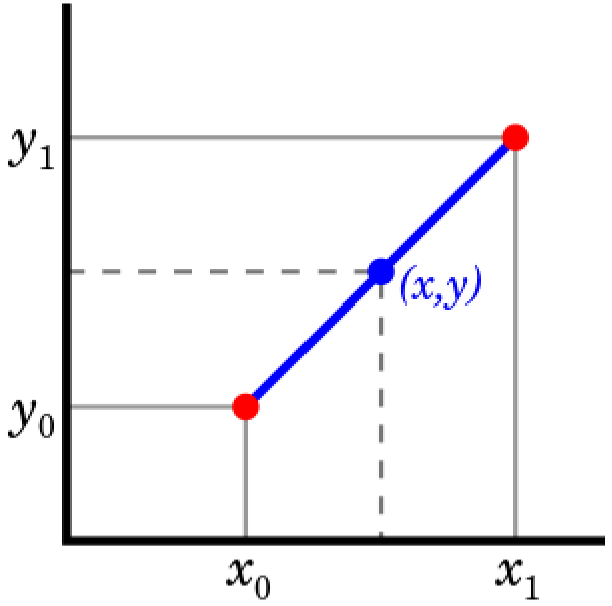
- How do I go get the source image value at this location???

Simple linear interpolation in 1D estimates desired missing function values by assuming linear behavior between known values

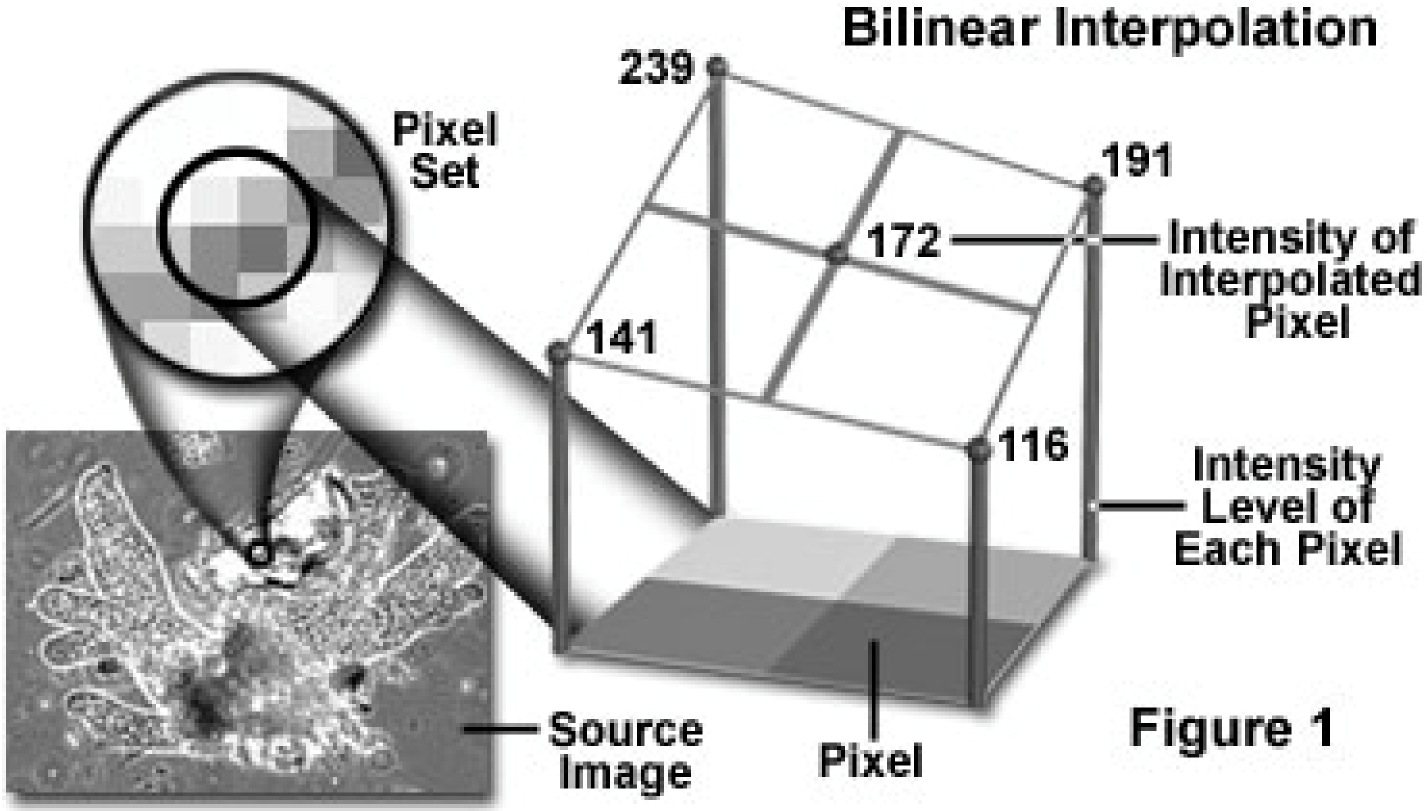
- Assume a line between each successive set of points in $\{x_1, x_2 \dots x_n\}$
- The function value at an arbitrary x_p between points x_k and x_{k+1} can be estimated as:

$$y_p \approx y_k + (x_p - x_k)\frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

- It's only an estimation…



The extension of linear interpolation to 2D is called bilinear interpolation



Bilinear interpolation finds the point between two points calculated by linear interpolation

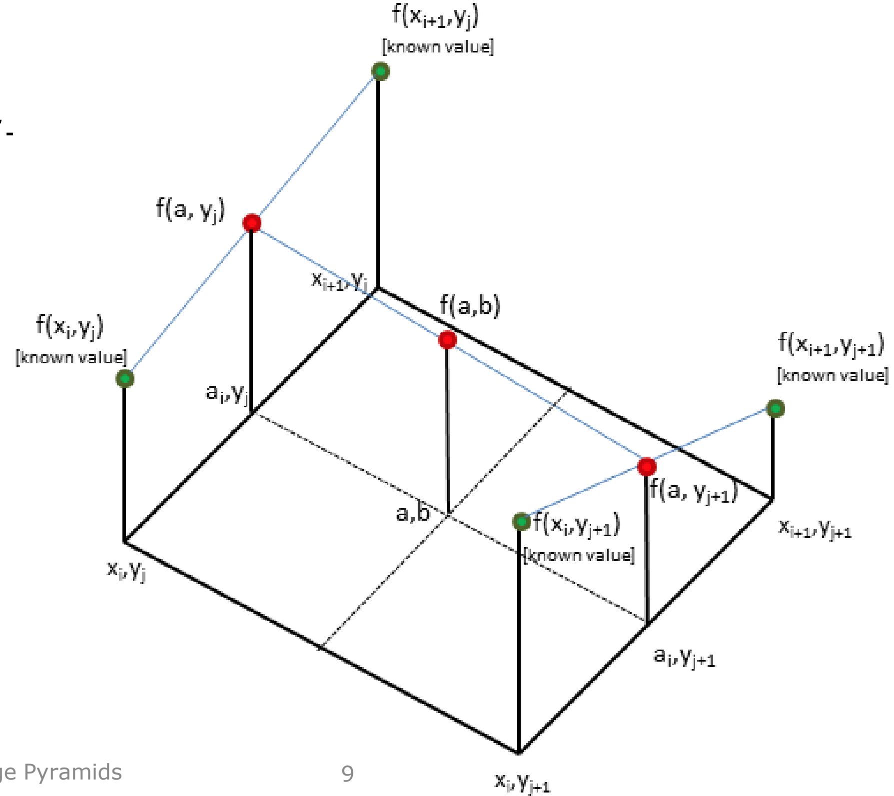
$$f(a,b) = \frac{1}{(x_{i+1}-x_i)(y_{j+1}-y_j)} \begin{bmatrix} x_{i+1}-a \\ a-x_i \end{bmatrix}^T \begin{bmatrix} f(x_i,y_j) & f(x_i,y_{j+1}) \\ f(x_{i+1},y_j) & f(x_{i+1},y_{j+1}) \end{bmatrix} \begin{bmatrix} y_{j+1}-b \\ b-y_j \end{bmatrix}$$

$$= \frac{1}{(x_{i+1}-x_i)(y_{j+1}-y_j)} \{ f(x_i,y_j)(x_{i+1}-a)(y_{j-}$$

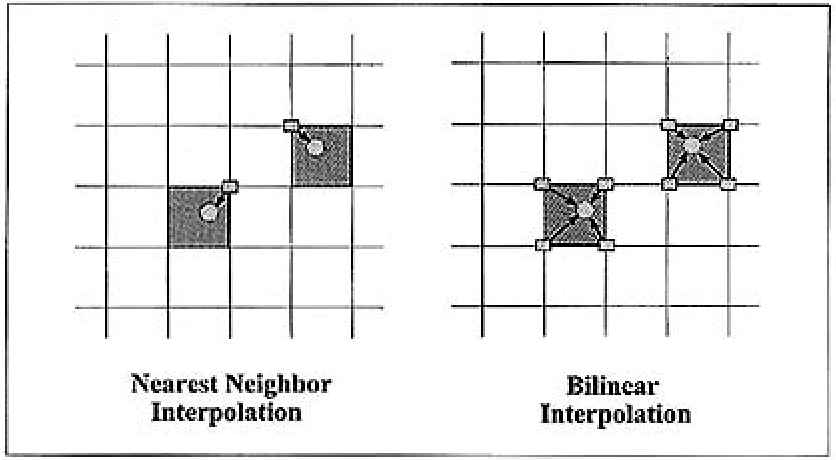
$$+ f(x_{i+1},y_j)(a-x_i)(y_{j+1}-b)$$

$$+ f(x_i,y_{j+1})(x_{i+1}-a)(b-y_j)$$

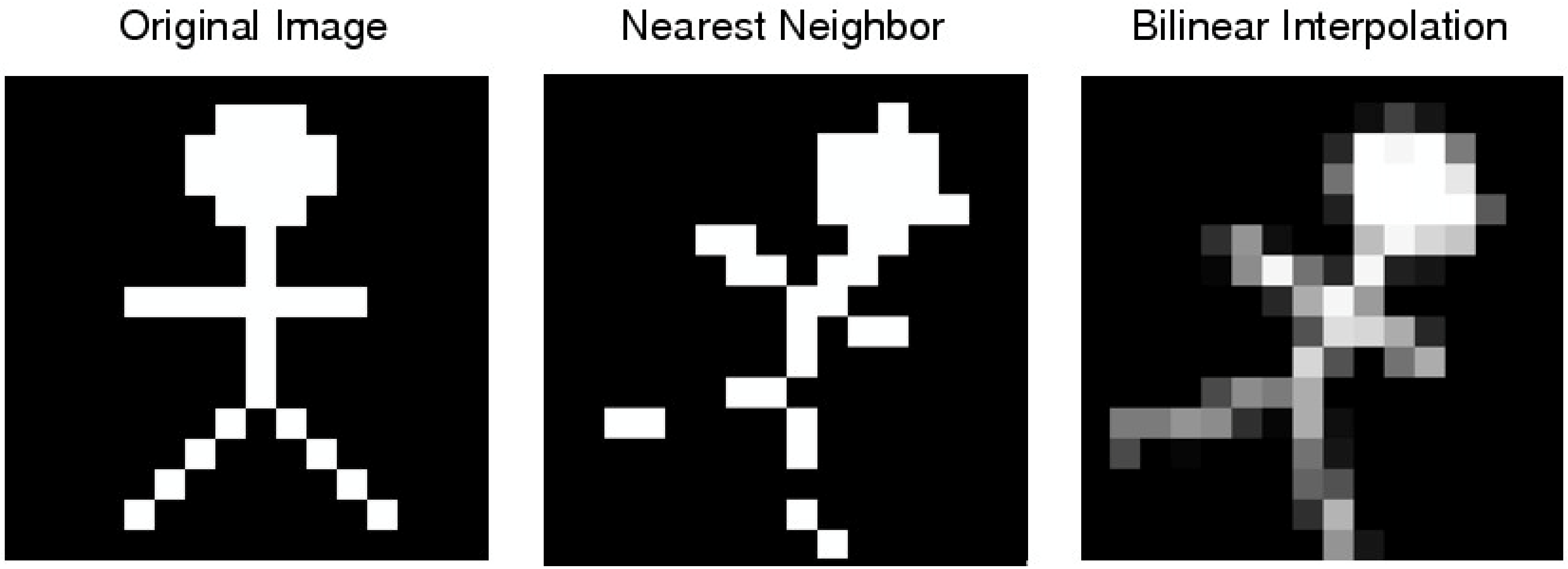
$$+ f(x_{i+1},y_{j+1})(a-x_i)(b-y_j) \}$$



Nearest-neighbor interpolation identifies which of the surrounding four pixels is closest to the desired location and uses its value



Nearest-neighbor will always use a pixel value present in the source image; bilinear interpolation generally finds new values between existing values

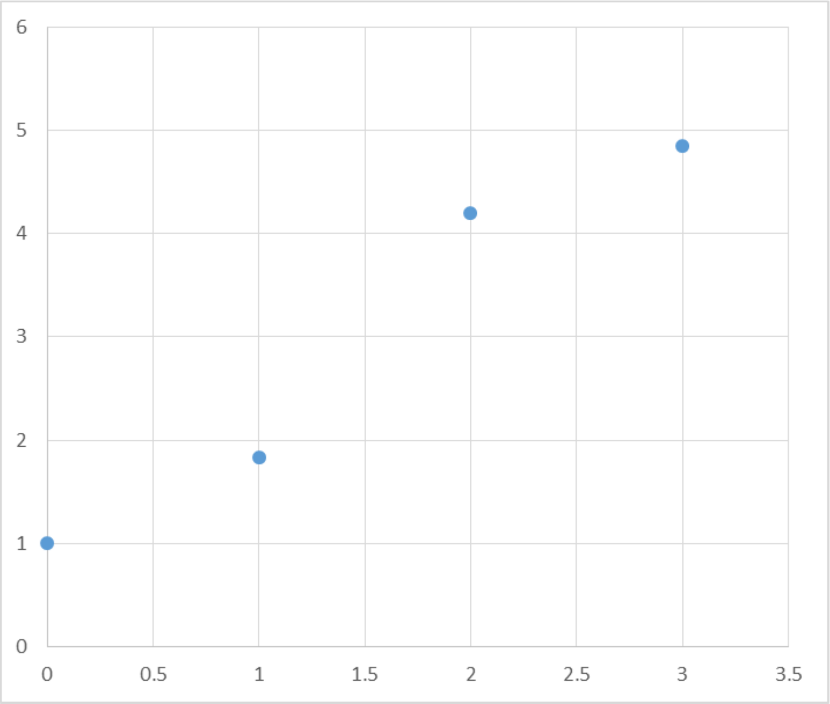


Cubic interpolation in 1D uses the values of more than the closest two points; four points are used to find the best third-order polynomial fit

$$f(x_p) = \sum_{u=\lfloor x_p \rfloor - 1}^{\lfloor x_p \rfloor + 2} w_{cub}(x_p - u) f(u)$$

where

$$w_{cub}(u) = \begin{cases} |u|^3 - 2|u|^2 + 1 & for\ 0 \le |u| < 1 \\ -|u|^3 + 5|u|^2 - 8|u| + 4 & for\ 1 \le |u| < 2 \\ 0 & for\ |u| \ge 2 \end{cases}$$

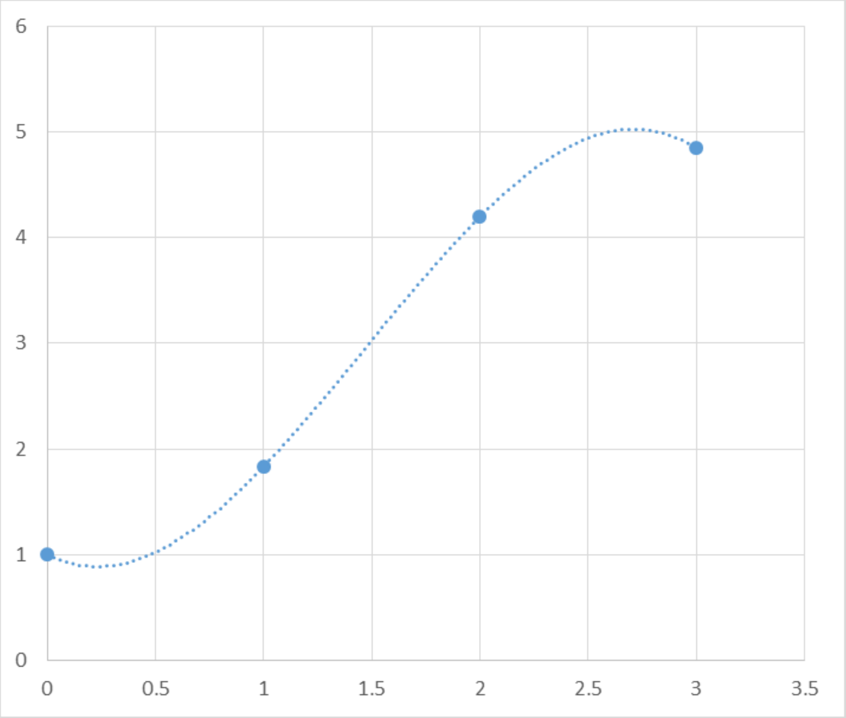


Cubic interpolation in 1D uses the values of more than the closest two points; four points are used to find the best third-order polynomial fit

$$f(x_p) = \sum_{u=\lfloor x_p \rfloor - 1}^{\lfloor x_p \rfloor + 2} w_{cub}(x_p - u) f(u)$$

where

$$w_{cub}(u) = \begin{cases} |u|^3 - 2|u|^2 + 1 & for\ 0 \le |u| < 1 \\ -|u|^3 + 5|u|^2 - 8|u| + 4 & for\ 1 \le |u| < 2 \\ 0 & for\ |u| \ge 2 \end{cases}$$

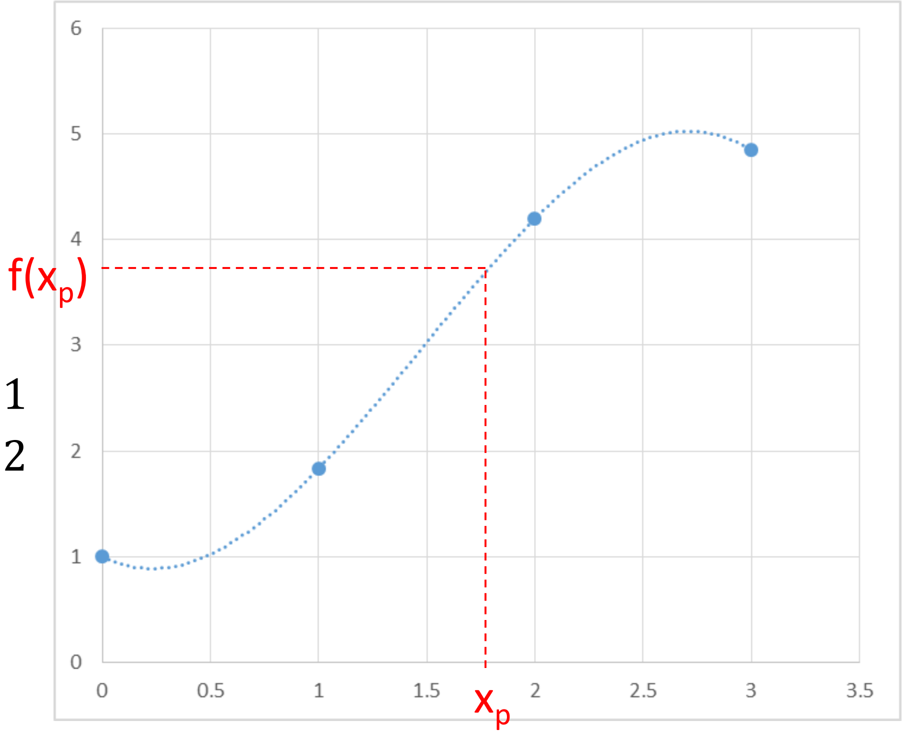


Cubic interpolation in 1D uses the values of more than the closest two points; four points are used to find the best third-order polynomial fit

$$f(x_p) = \sum_{u=\lfloor x_p \rfloor - 1}^{\lfloor x_p \rfloor + 2} w_{cub}(x_p - u) f(u)$$

where

$$w_{cub}(u) = \begin{cases} |u|^3 - 2|u|^2 + 1 & for\ 0 \le |u| < 1 \\ -|u|^3 + 5|u|^2 - 8|u| + 4 & for\ 1 \le |u| < 2 \\ 0 & for\ |u| \ge 2 \end{cases}$$



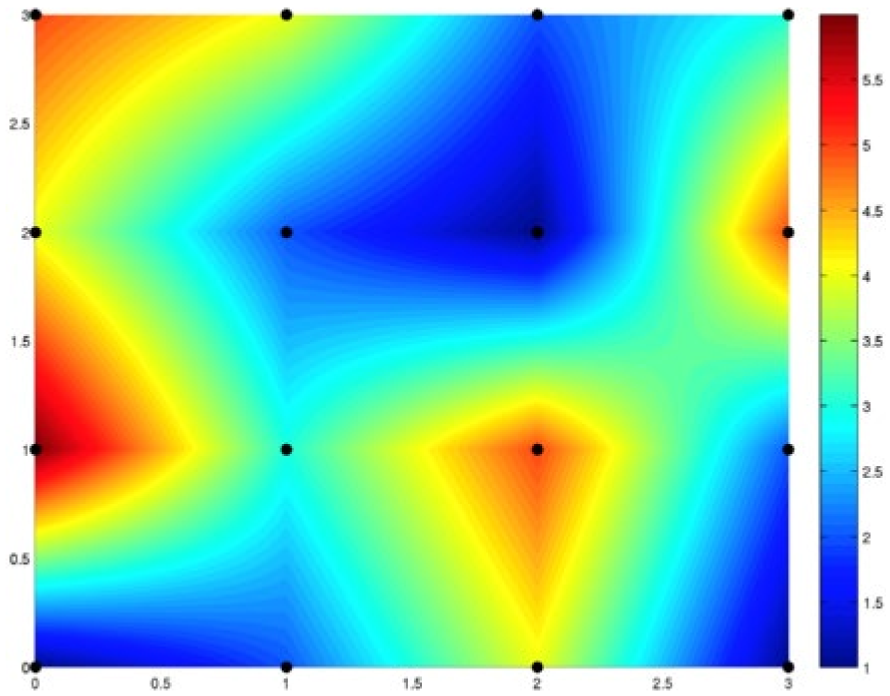
Bicubic interpolation in 2D follows the same general approach – cubic function profiles are assumed in each direction

$$f(x_p, y_p) = \sum_{j=0}^{3}\left[w_{cub}(y_p - y_j)\sum_{i=0}^{3} I(x_i, y_j)w_{cub}(x_p - x_i)\right]$$

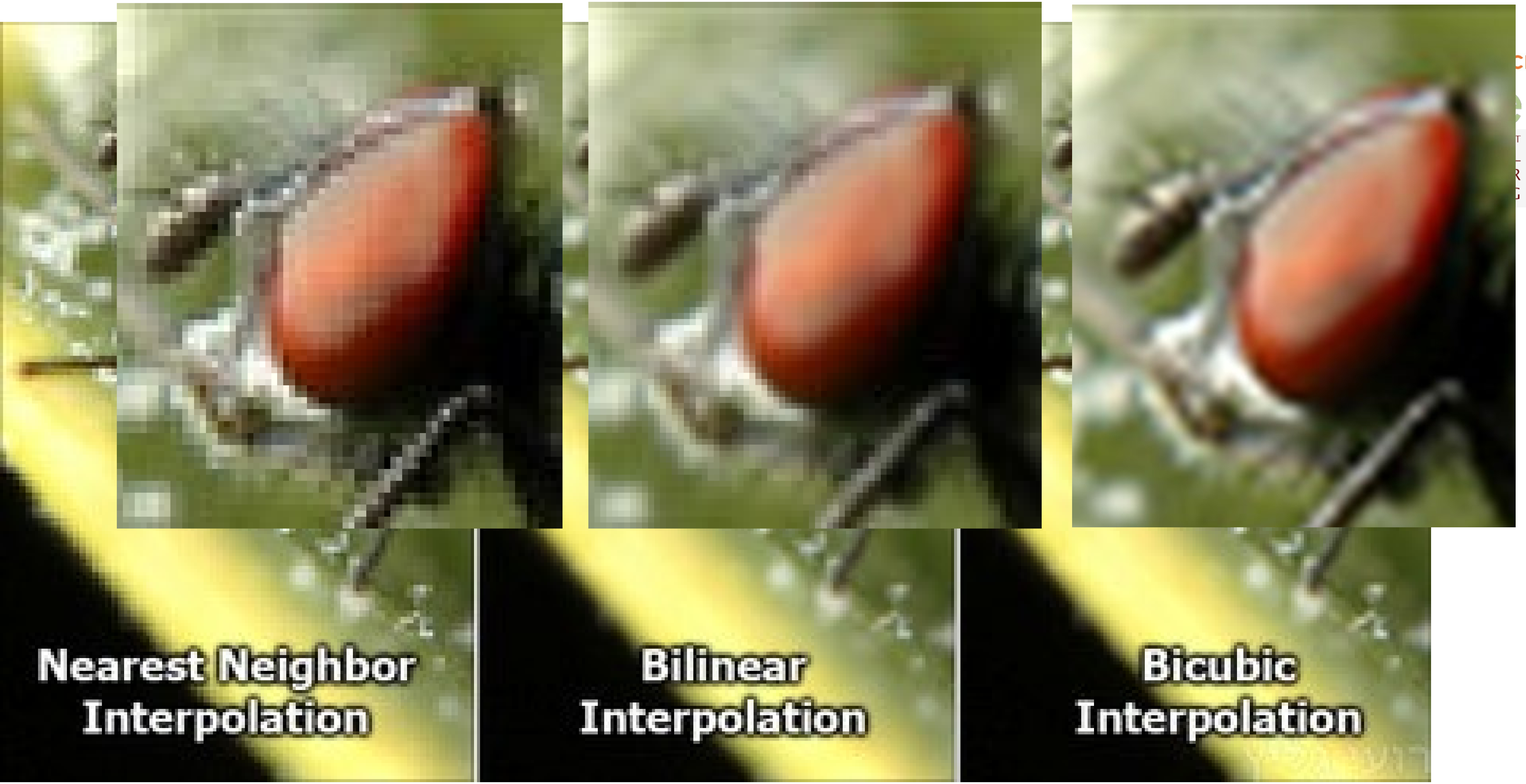
where

$$w_{cub}(u) = \begin{cases} |u|^3 - 2|u|^2 + 1 & for\ 0 \le |u| < 1 \\ -|u|^3 + 5|u|^2 - 8|u| + 4 & for\ 1 \le |u| < 2 \\ 0 & for\ |u| \ge 2 \end{cases}$$

Assumption of independent cubic function projections in each direction can lead to some complex 2D functions for the intensity value




Nearest Neighbor Interpolation
Bilinear Interpolation
Bicubic Interpolation


Nearest Neighbor Interpolation
Bilinear Interpolation
Bicubic Interpolation

Lanczos interpolation is the most popular “high-fidelity” interpolation method; commonly used in image utilities for rescaling

$$I(x,y) = \sum_{v=\lfloor y \rfloor -2}^{\lfloor y \rfloor +3} \left[\sum_{u=\lfloor x \rfloor -2}^{\lfloor x \rfloor +3} I(u,v) w_{L3}(x-u) w_{L3}(y-v) \right]$$

where the Lanczos filter value is given by:

$$w_{L3}(z) = \begin{cases} 1 & for\ |z| = 0 \\ \dfrac{\sin\left(\pi z/_3\right)}{\left(\pi z/_3\right)} & for\ 0 < |z| < 3 \\ 0 & for\ |z| \geq 3 \end{cases}$$

More powerful interpolation methods have a dramatic effect on the visual results

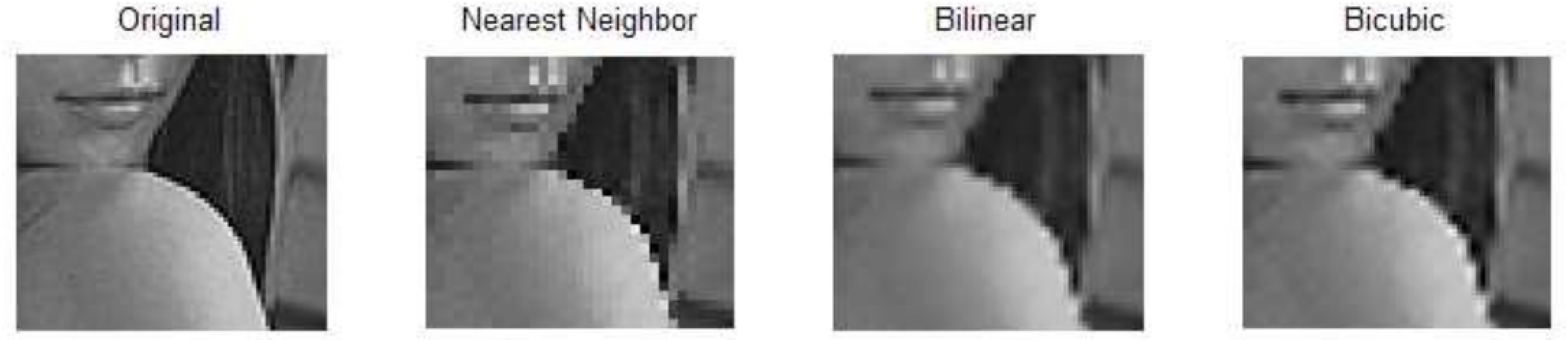


Figure 1.2: Part of Lena image downsampled and then upsampled by factor K=2.

Wittman – "Mathematical Techniques for Image Interpolation", http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.8726&rep=rep1&type=pdf

Formally, interpolation is defined as

$$I_o(i,j) = \sum_{k,l} f(k,l)h(i-rk, j-rl);$$

we can think of it as either superposition of a number of interpolation kernels or one kernel at varying phase shifts

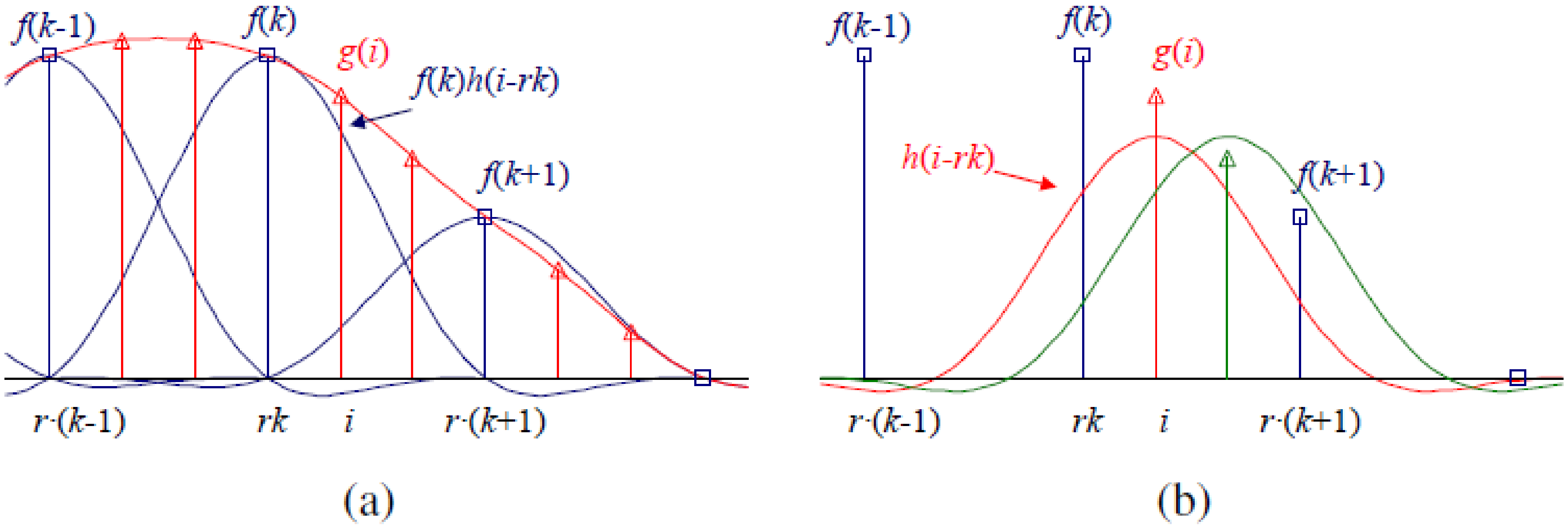
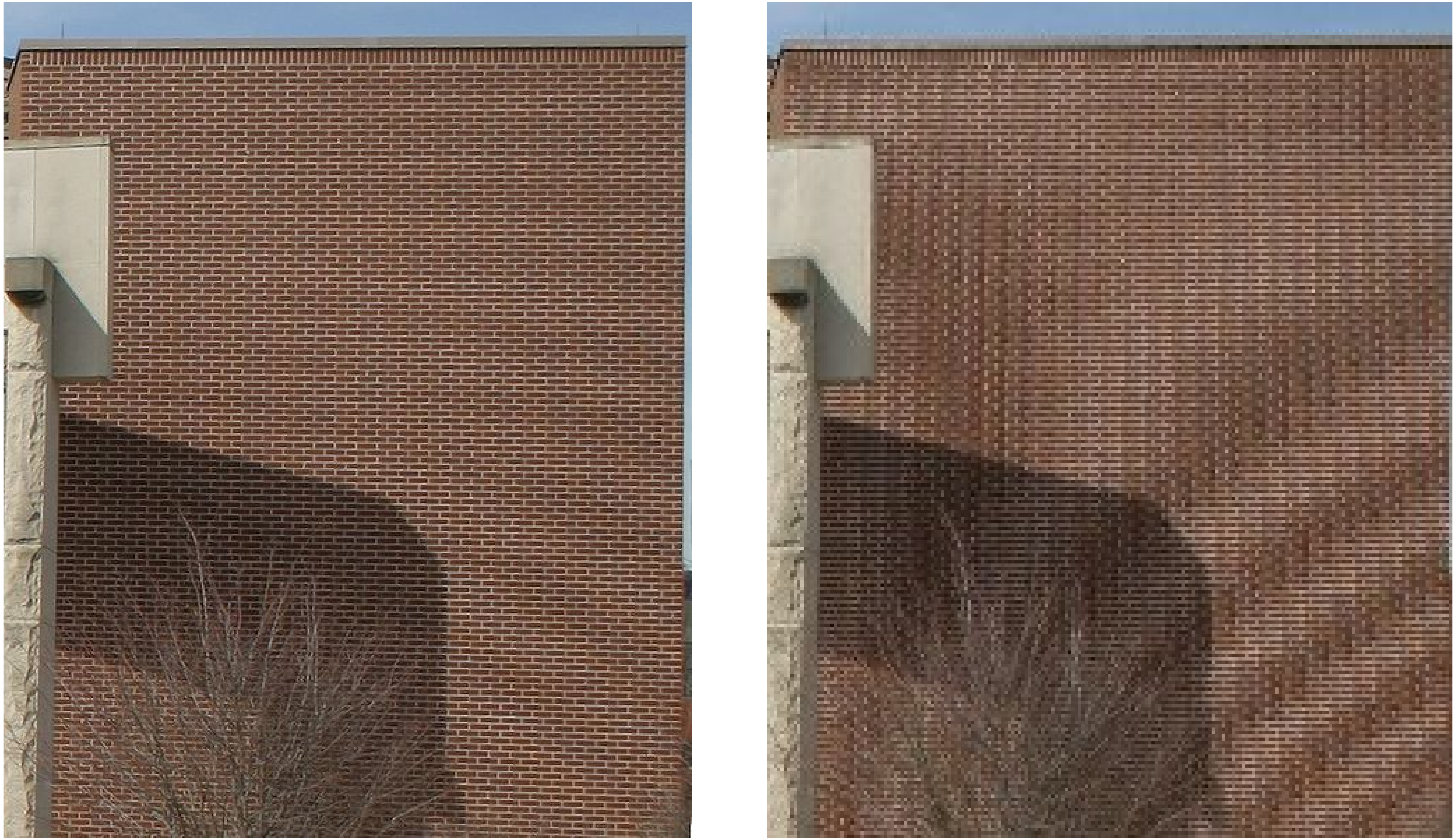
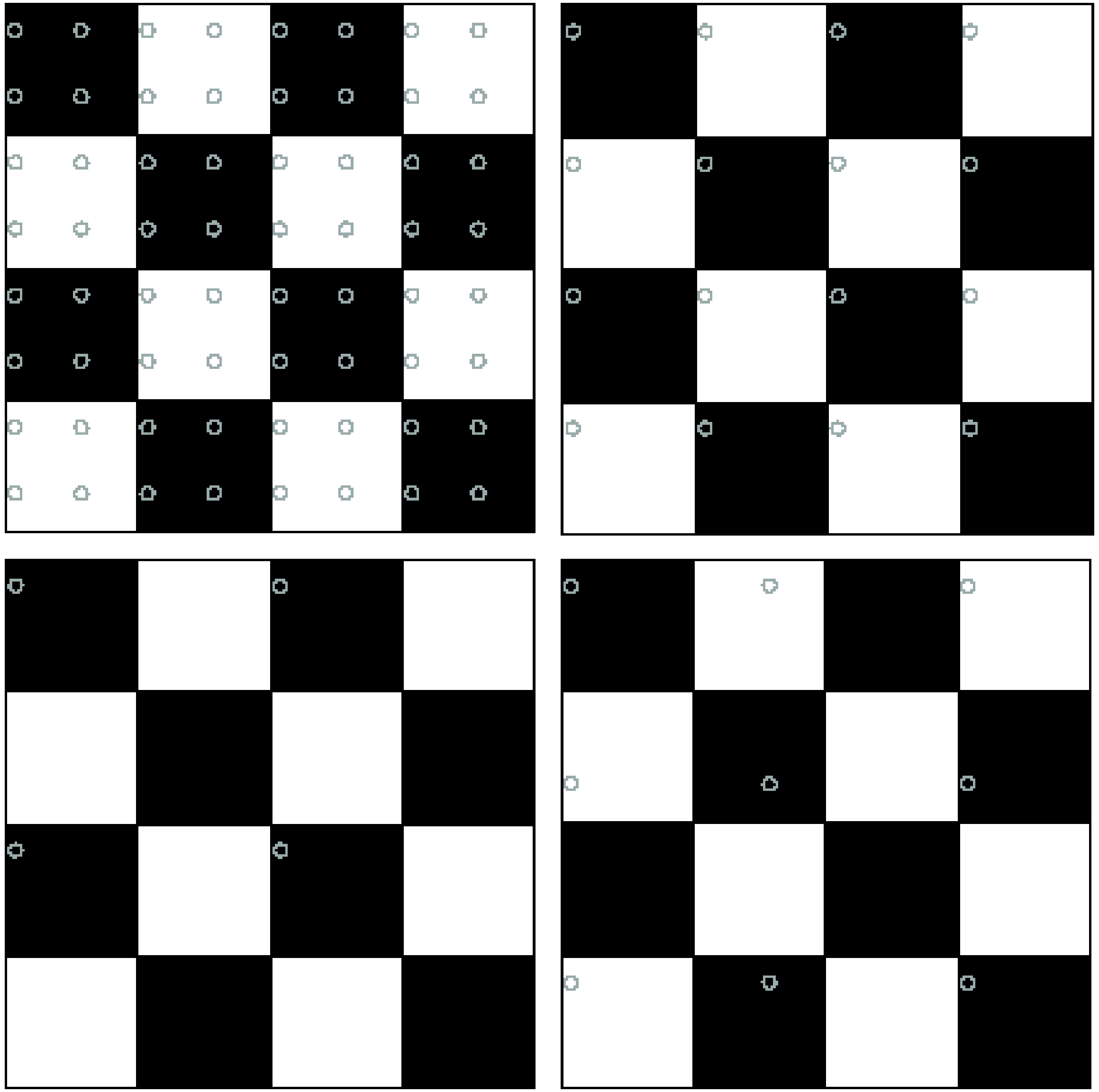


Figure 3.27 Signal interpolation, $g(i) = \sum_k f(k)h(i-rk)$: (a) weighted summation of input values; (b) polyphase filter interpretation.

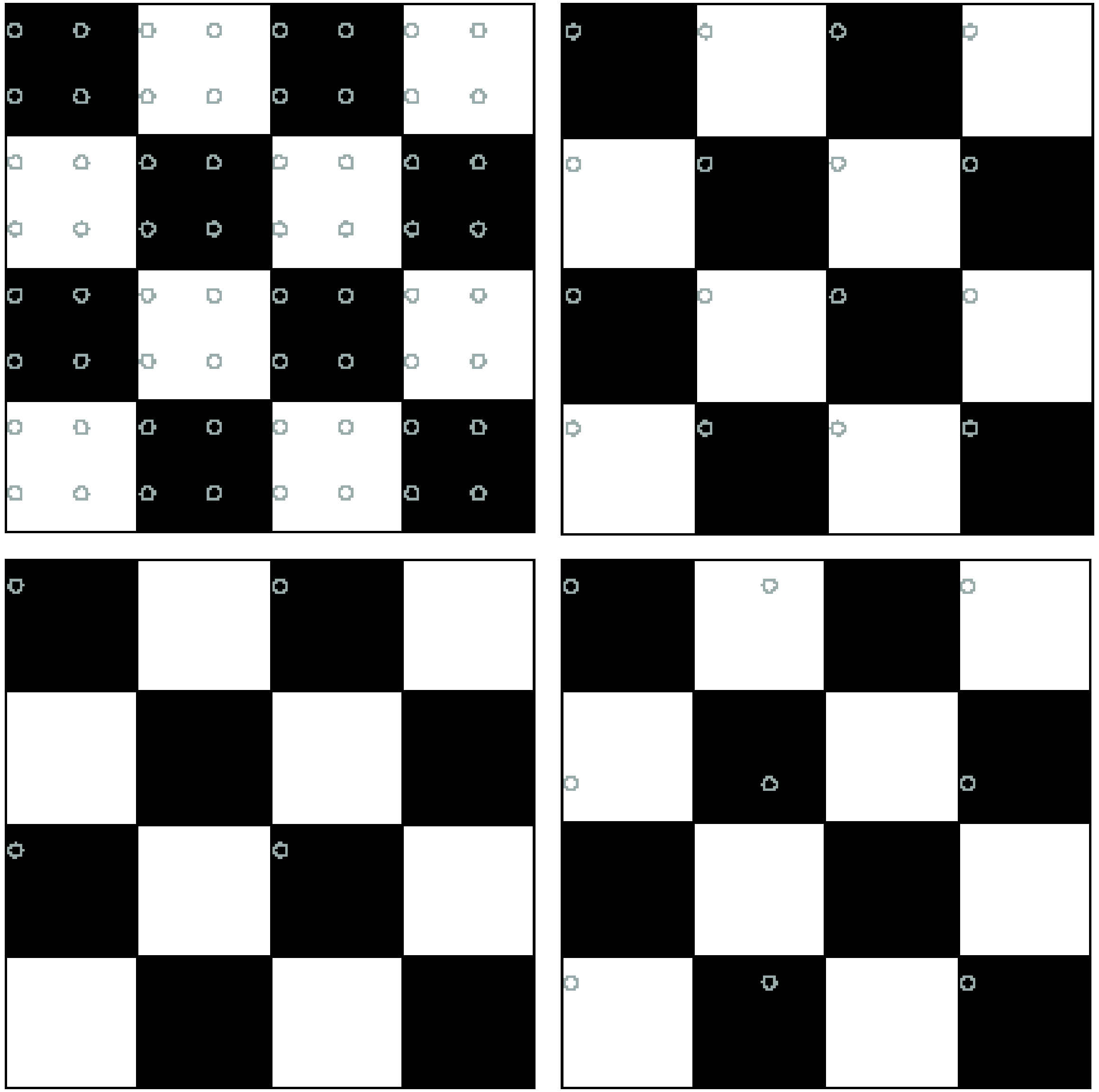


Example of aliasing producing a moiré pattern
(source: Wikipedia)

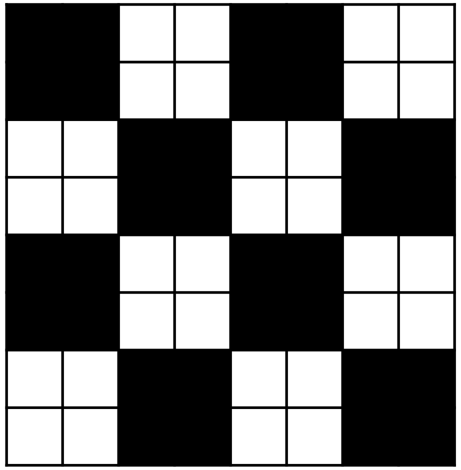


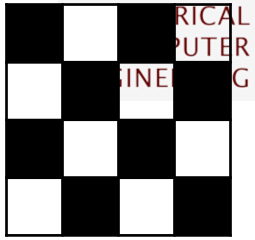
Resample the checkerboard by taking one sample at each circle. In the case of the top left board, new representation is reasonable. Top right also yields a reasonable representation. Bottom left is all black (dubious) and bottom right has checks that are too big.

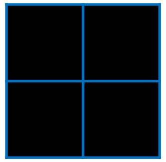
Slide credit: Darrell

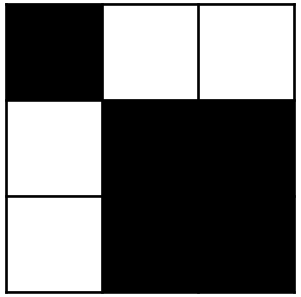


Resample the checkerboard by taking one sample at each circle. In the case of the top left board, new representation is reasonable.
Top right also yields a reasonable representation.
Bottom left is all black (dubious) and bottom right has checks that are too big.







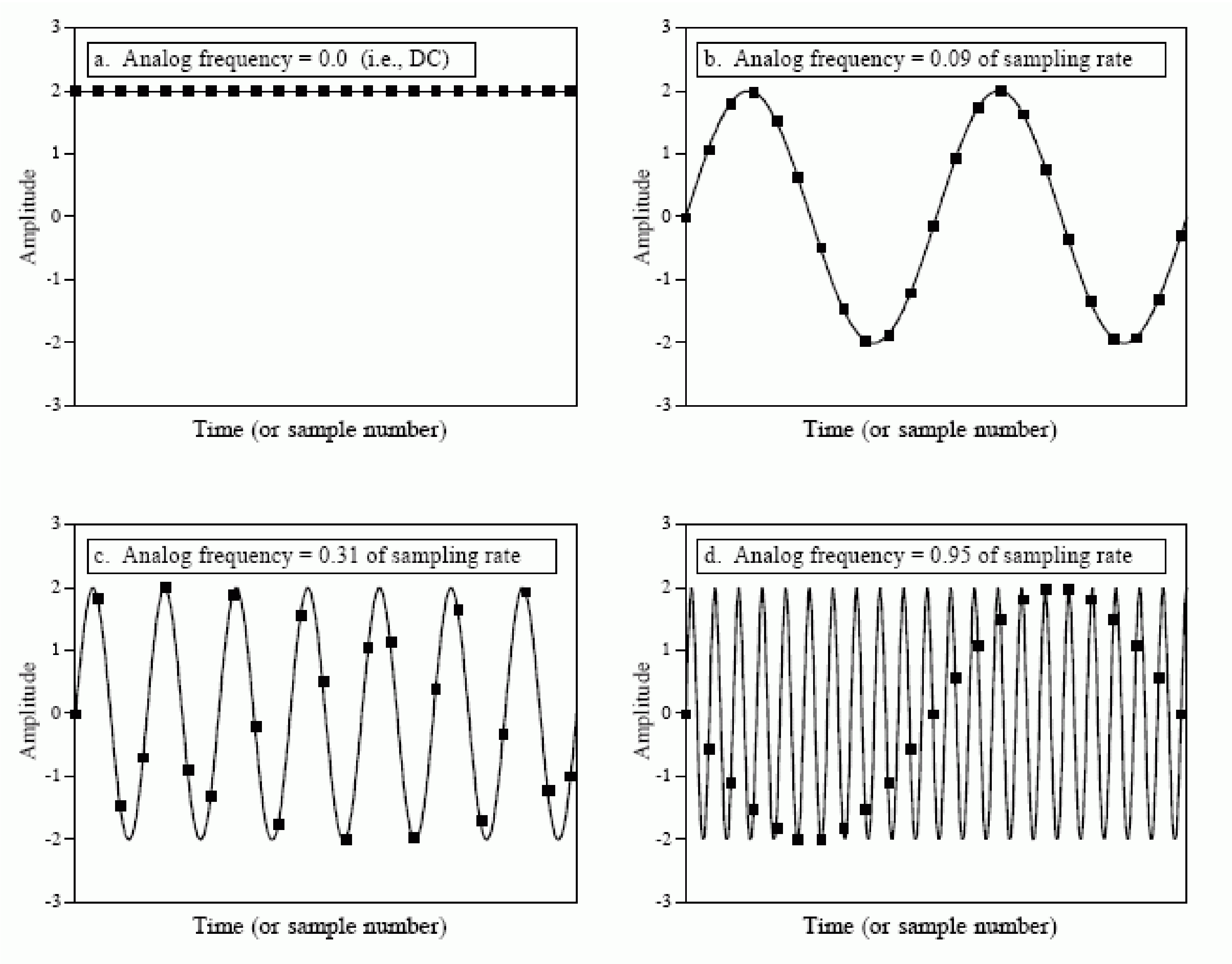


Slide credit: Darrell

A smoothing operation (low-pass filter) removes fine details from an image

Aliasing will not occur if the Nyquist criterion is satisfied

The *Nyquist sampling criterion* says that we will not have aliasing if our sampling frequency is more than twice the frequency of the highest sinusoidal frequency present



The Scientist and Engineer's Guide to Digital Signal Processing
By Steven W. Smith, Ph.D.

Figure 3-3 shows several values for $\frac{sampling\ rate}{\max frequency}$. In (a), the frequency is zero, so the ratio approaches infinity.

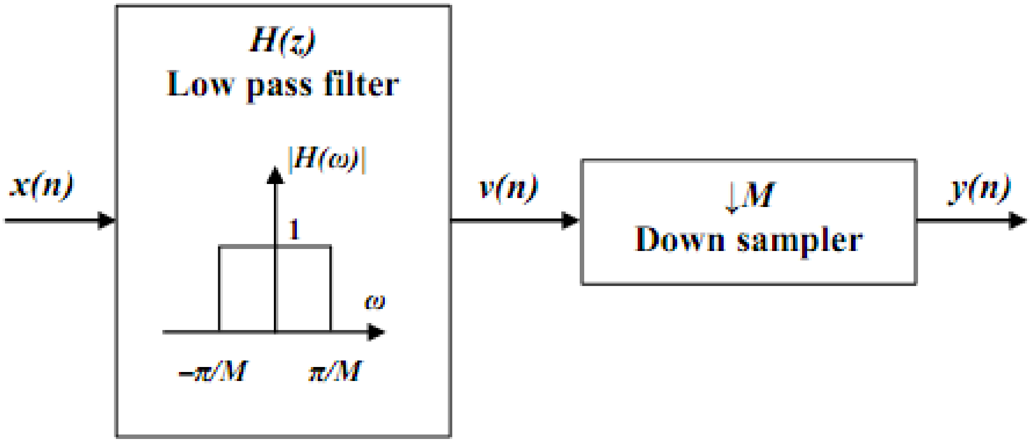
Plot (b) shows a sampling rate 11.1 times the highest frequency present. As you can see, the data well represents the sine wave.

In (c), the sine wave's frequency to 0.31 of the sampling rate. This results in only 3.2 samples per sine wave cycle. Despite the appearance, the samples are a unique representation of the analog signal. This also meets the Nyquist criterion.

In (d), the sampling rate is 1.05 samples per sine wave cycle, which is less than two. Do these samples properly represent the data? No! The apparent sine wave at the lower frequency is called aliasing.

Decimation or *downsampling* is the process of reducing the resolution in a mathematically correct manner

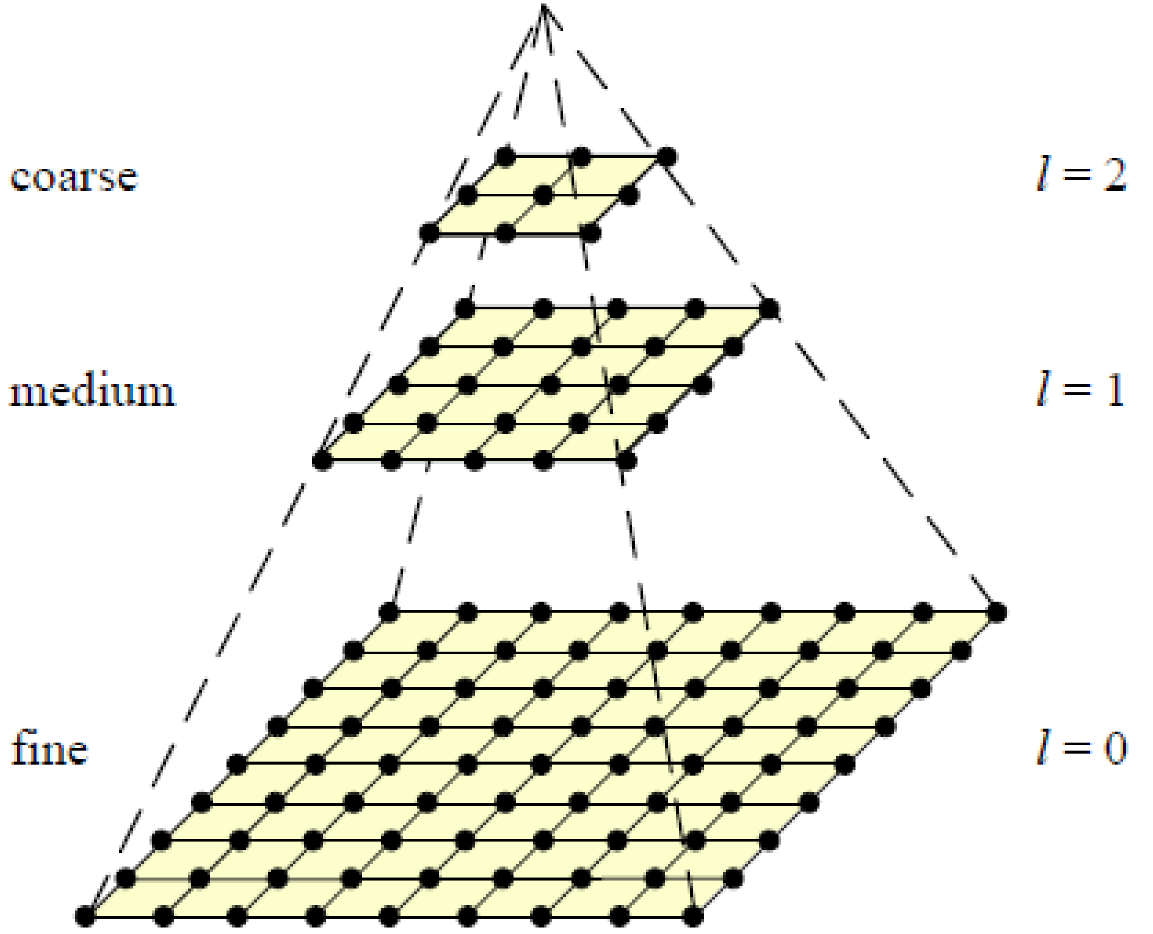
- Because we want to avoid aliasing, we will *low-pass filter* the original image to ensure that no spatial frequencies are present that will violate the Nyquist criterion <u>for the new lower sampling rate</u>
- Therefore, downsampling usually involves application of a suitable smoothing filter followed by keeping only each r^{th} sample



Some reasons for low-pass filtering

- To reduce noise
- To facilitate analysis of
  coarse (low-frequency) image content
- To allow subsampling without aliasing
- Other names for subsampling:
  - downsampling
  - decimation
- With different amounts of image blurring, image content can be analyzed at different "scales"

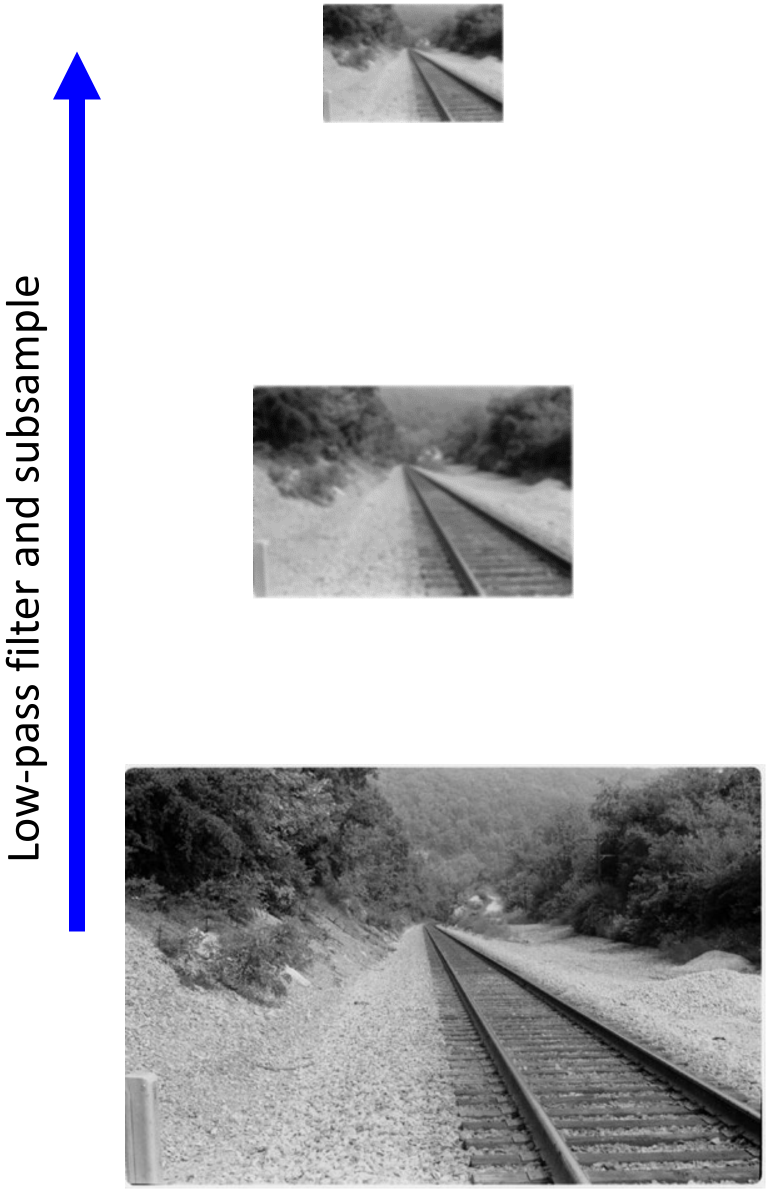
MULTIRESOLUTION REPRESENTATIONS



This pyramid shows the representation of the original image in three different *scale-space representations*

Figure 3.32 A traditional image pyramid: each level has half the resolution (width and height), and hence a quarter of the pixels, of its parent level.

An image pyramid

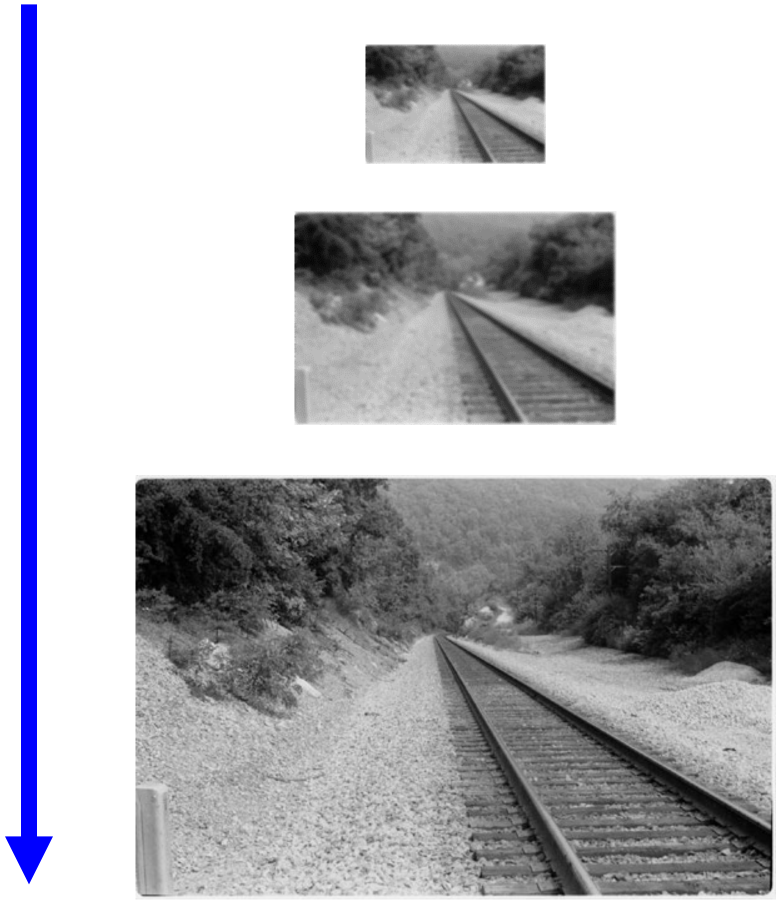


Why is the multiresolution approach useful?

- Images usually contain features of physically significant structure at different scales of resolution
- For some problems, this allows us to select a desired level of detail
- For some problems, processing at a coarse level first and continuing to finer levels can reduce the computation time greatly

Coarse-to-fine processing can lead to more efficient methods:

- Begin processing using an image that contains only very low spatial frequencies
- Continue processing, using images with increasingly higher resolution
- For each successive level, use results obtained at the previous level to guide the analysis



Some common pyramid representations:

- **Gaussian pyramids**
- **Laplacian pyramids**
  (Burt and Adelson, 1983)

- Wavelet decompositions
- Steerable pyramids

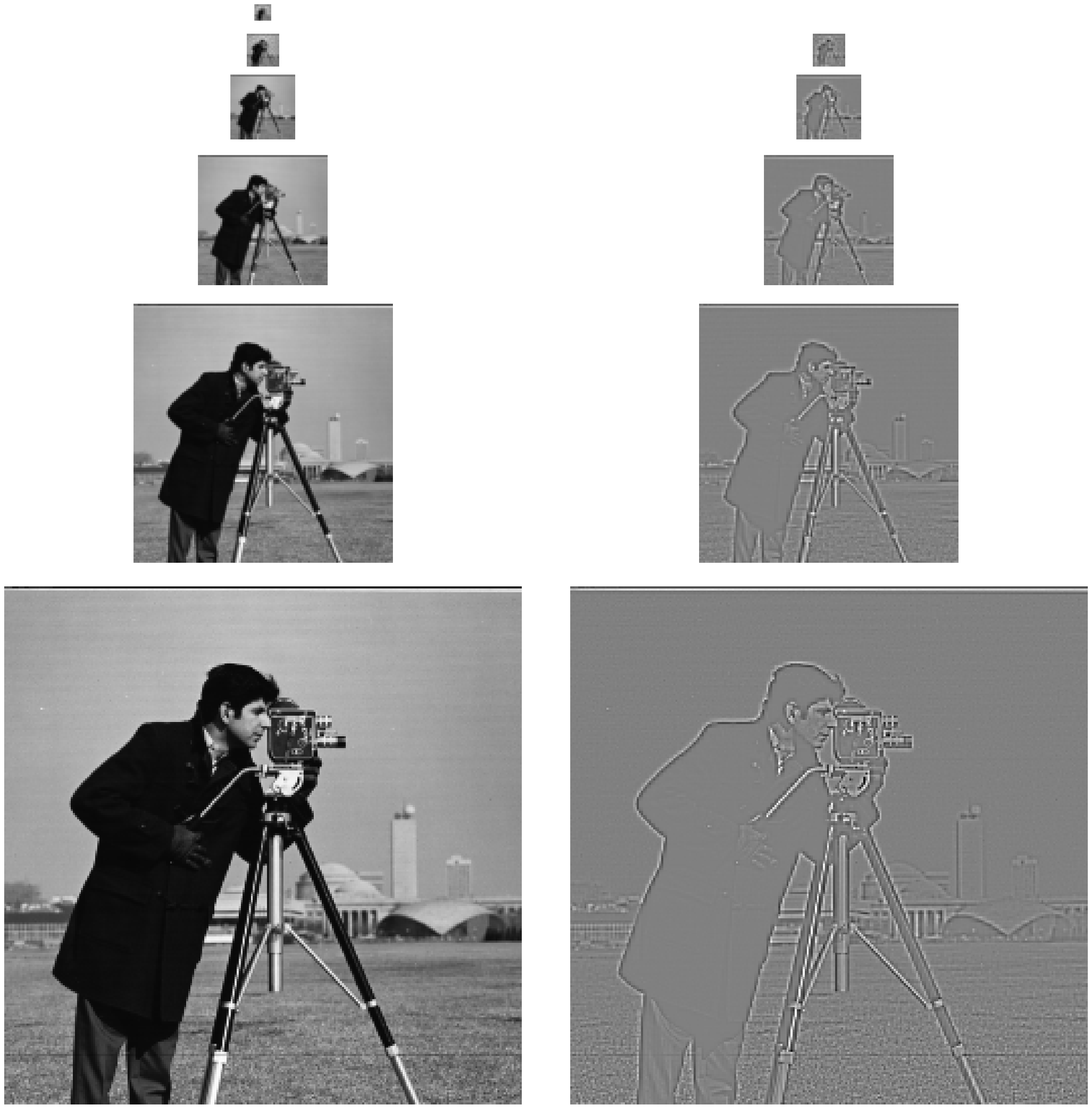
Level 4, 5

Level 3

Level 2

Level 1

Level 0



Each new level of a Gaussian pyramid is formed from the previous level by smoothing and subsampling:

$$g_k(i,j) = \sum_{m=-2}^{2}\sum_{n=-2}^{2} w(m,n)\, g_{k-1}(2i-m, 2j-n)$$

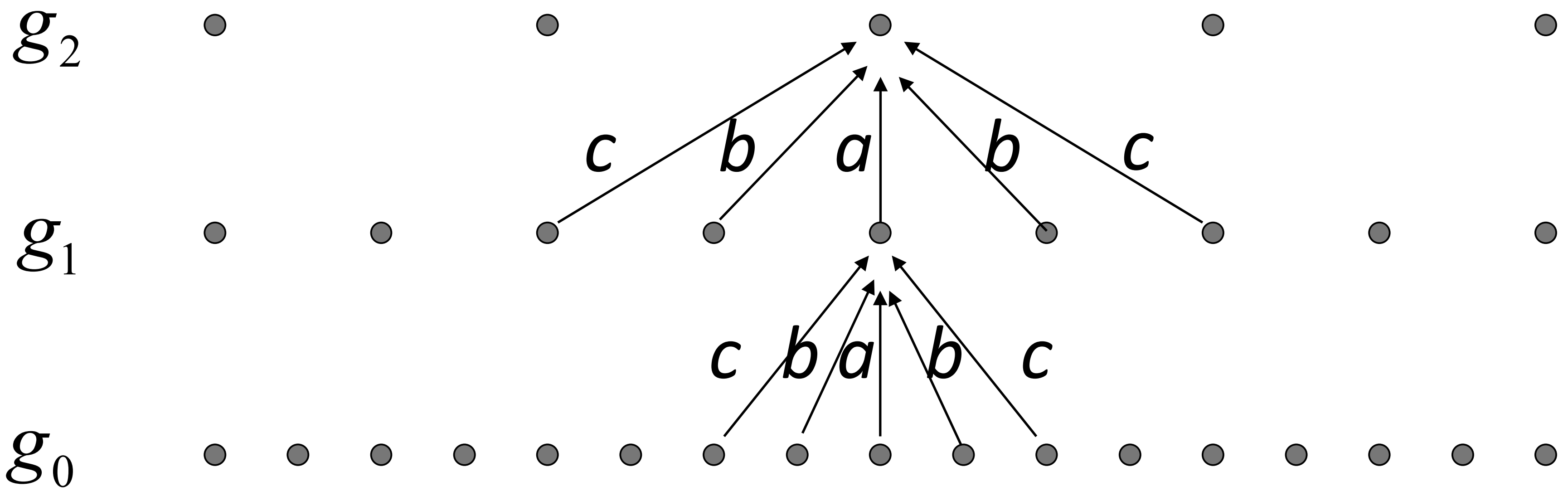
- Conceptually, this is written as $g_k = REDUCE(g_{k-1})$

- Commonly, the filter *w* is chosen to be symmetric, normalized, separable, and unimodal (to resemble a Gaussian function)

$$w = [c\ b\ a\ b\ c] * [c\ b\ a\ b\ c]$$

- e.g., *a* = 3/8, *b* = 1/4, *c* = 1/16

A (one-dimensional) look at the application of a kernel to produce the (reduced) higher layers



A Laplacian pyramid is obtained by:

$$L_k = g_k - EXPAND(g_{k+1})$$

There are 2 basic ways to realize the EXPAND operation:

1. For a given image at level k + 1, use interpolation to double the number of pixels in both directions
2. At the time the Gaussian pyramid is produced, also compute intermediate pixel values for level k + 1; use those values to compute the Laplacian image (and then discard them)

- The Gaussian pyramid has 1 more level than the Laplacian pyramid
- From the Laplacian pyramid, it is possible to reconstruct the original image (if the coarsest level of the Gaussian pyramid is available)

Gaussian Pyramid

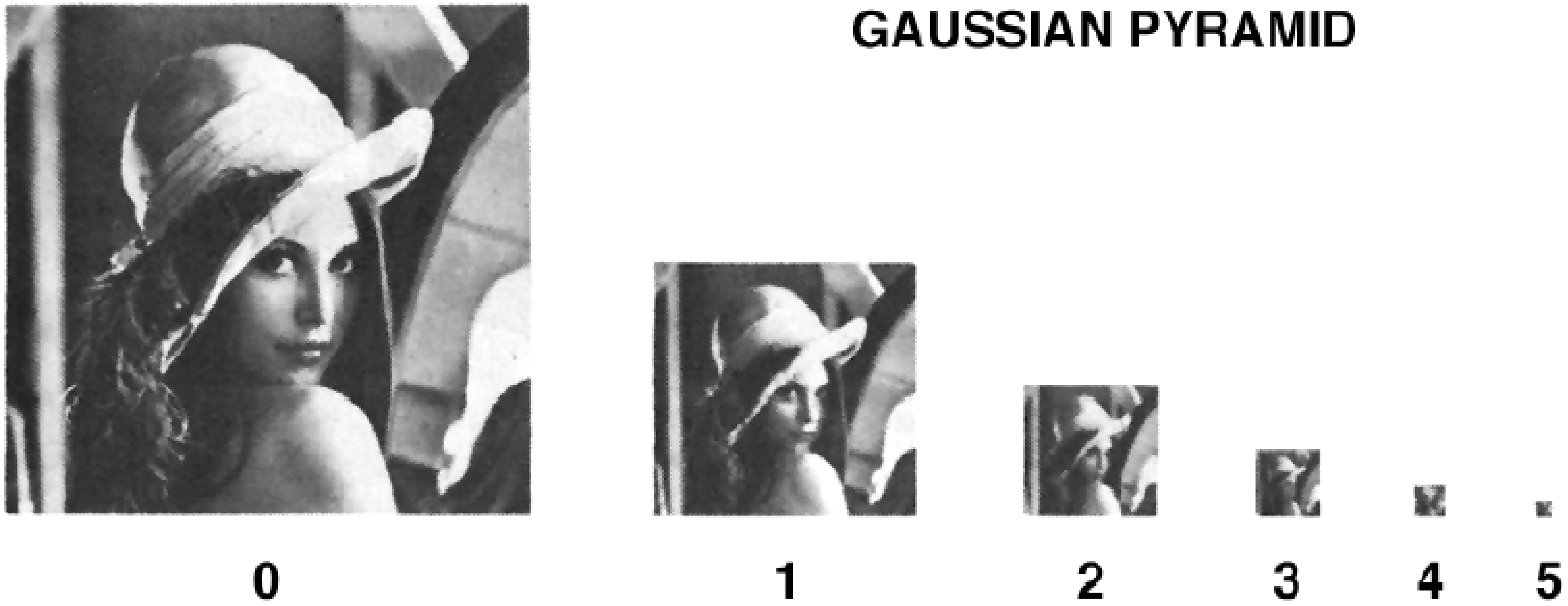


Fig. 4. First six levels of the Gaussian pyramid for the "Lady" image The original image, level 0, meusures 257 by 257 pixels and each higher level array is roughly half the dimensdons of its predecessor. Thus, level 5 measures just 9 by 9 pixels.

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

Gaussian pyramid (top)
Laplacian pyramid (bottom)

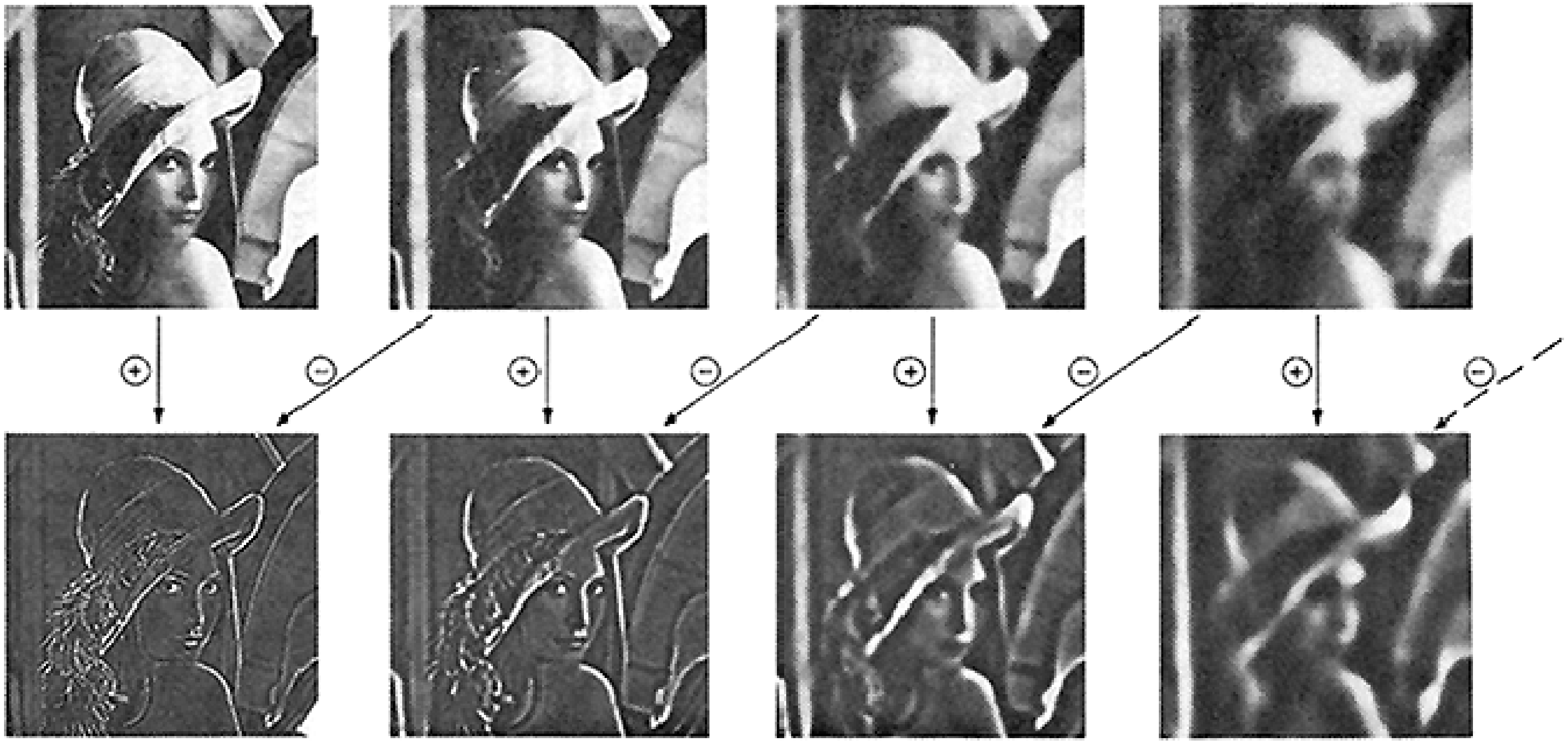
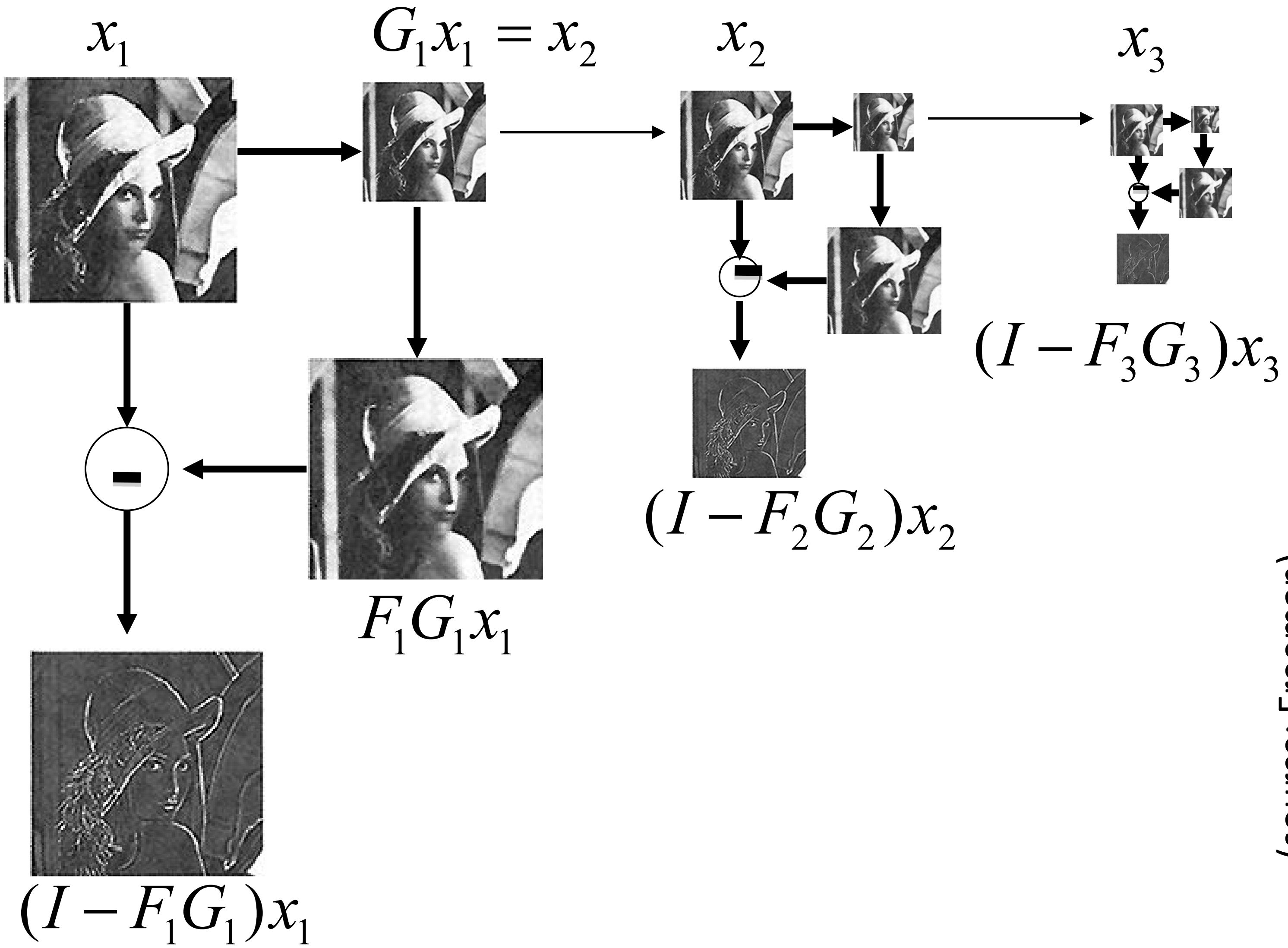


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

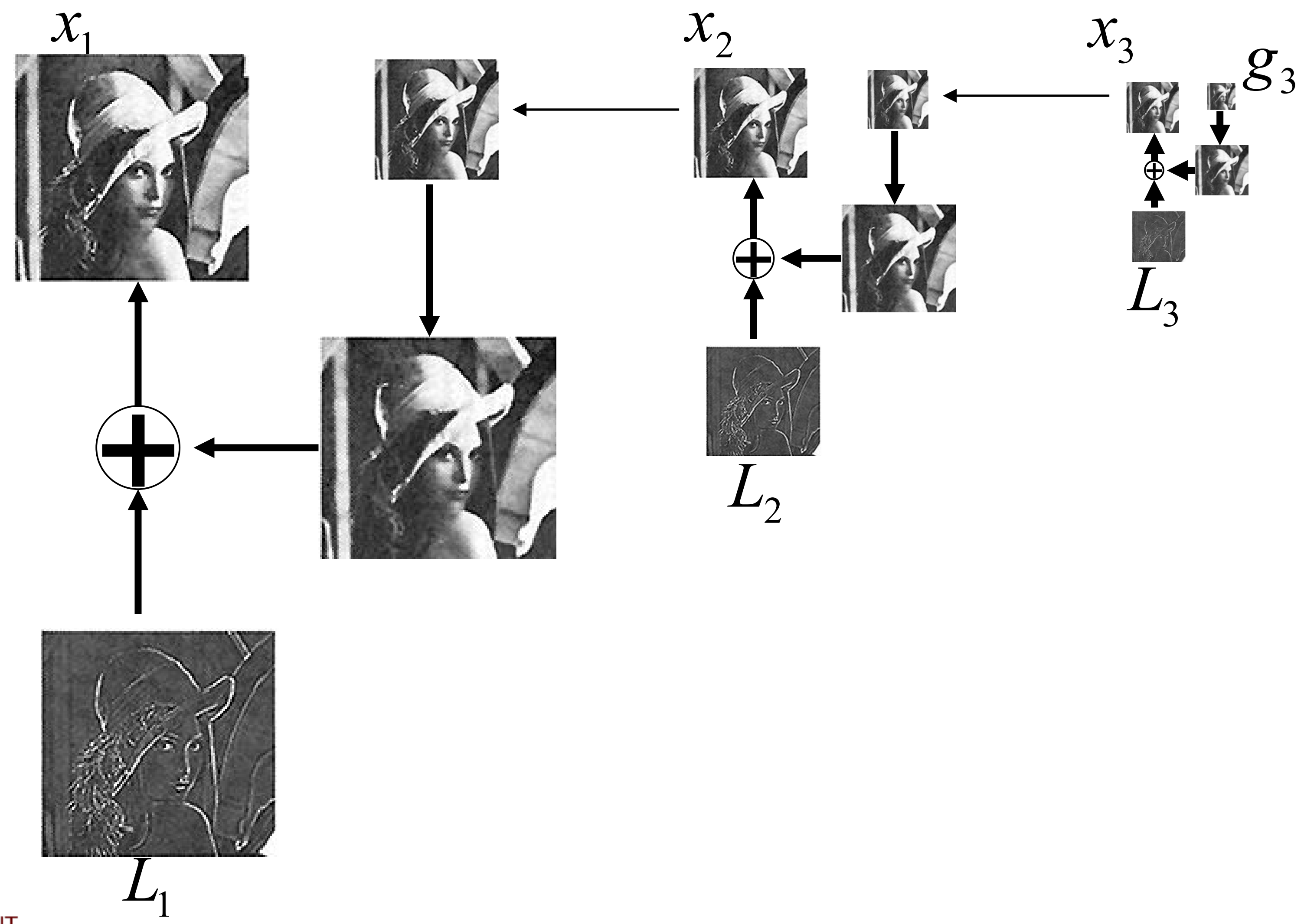
IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

Laplacian pyramid algorithm

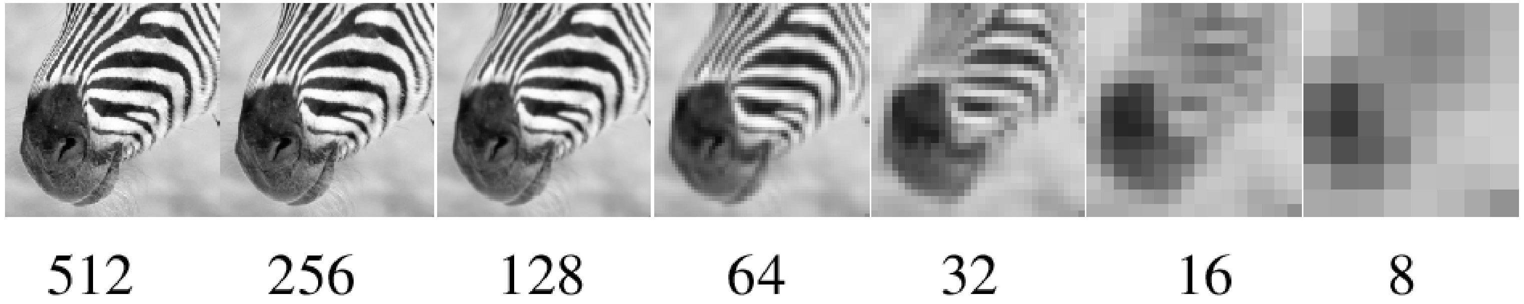


(source: Freeman)

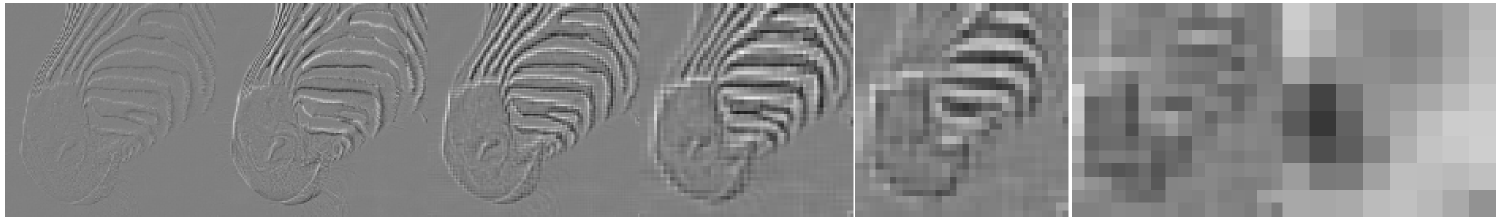
Laplacian pyramid reconstruction algorithm: recover x_1 from L_1, L_2, L_3 and g_3



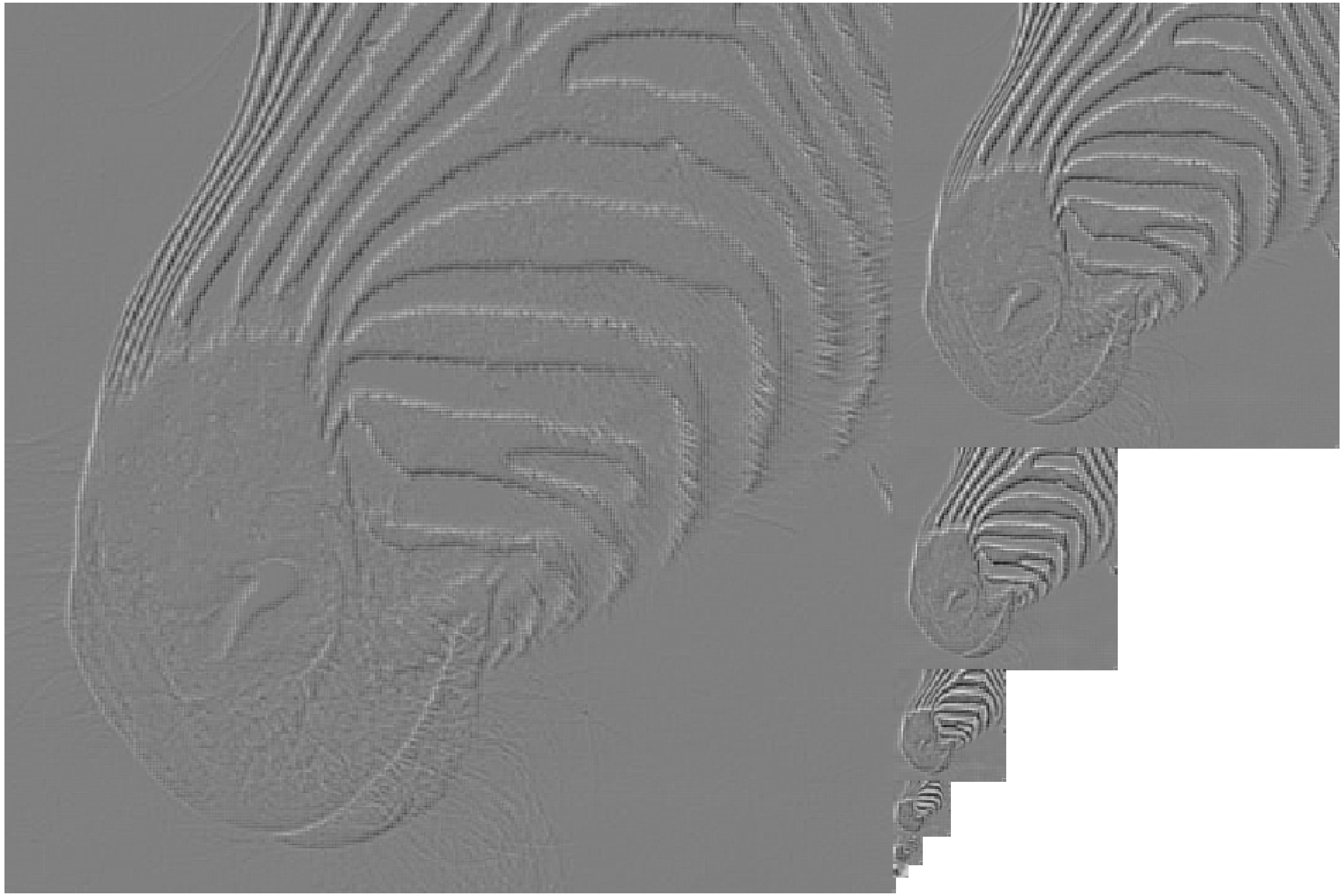
(source: Freeman)



(source: Freeman)



512 256 128 64 32 16 8



(source: Freeman)

So what are Gaussian and Laplacian pyramids used for?

- Compact representation of the image at different scales
  - Certain uses may only need a low resolution rendition
- Progressive display
  - Higher levels of the pyramid are successively smaller thumbnails
- Multiresolution processing
  - For object location, look in the lower resolution space for a rough outline of the object, then drop to the higher resolution levels for verification and precise location
- Compression
- Various advanced algorithms
  - Lucas-Kanade
  - SIFT

Today’s Objectives

- Interpolation
  - Bilinear interpolation
  - Bicubic interpolation
  - Lanczos interpolation
  - Downsampling
- Multiresolution representations
  - Image pyramid concept
  - Gaussian pyramid
  - Laplacian pyramid