

# Lecture 2.2

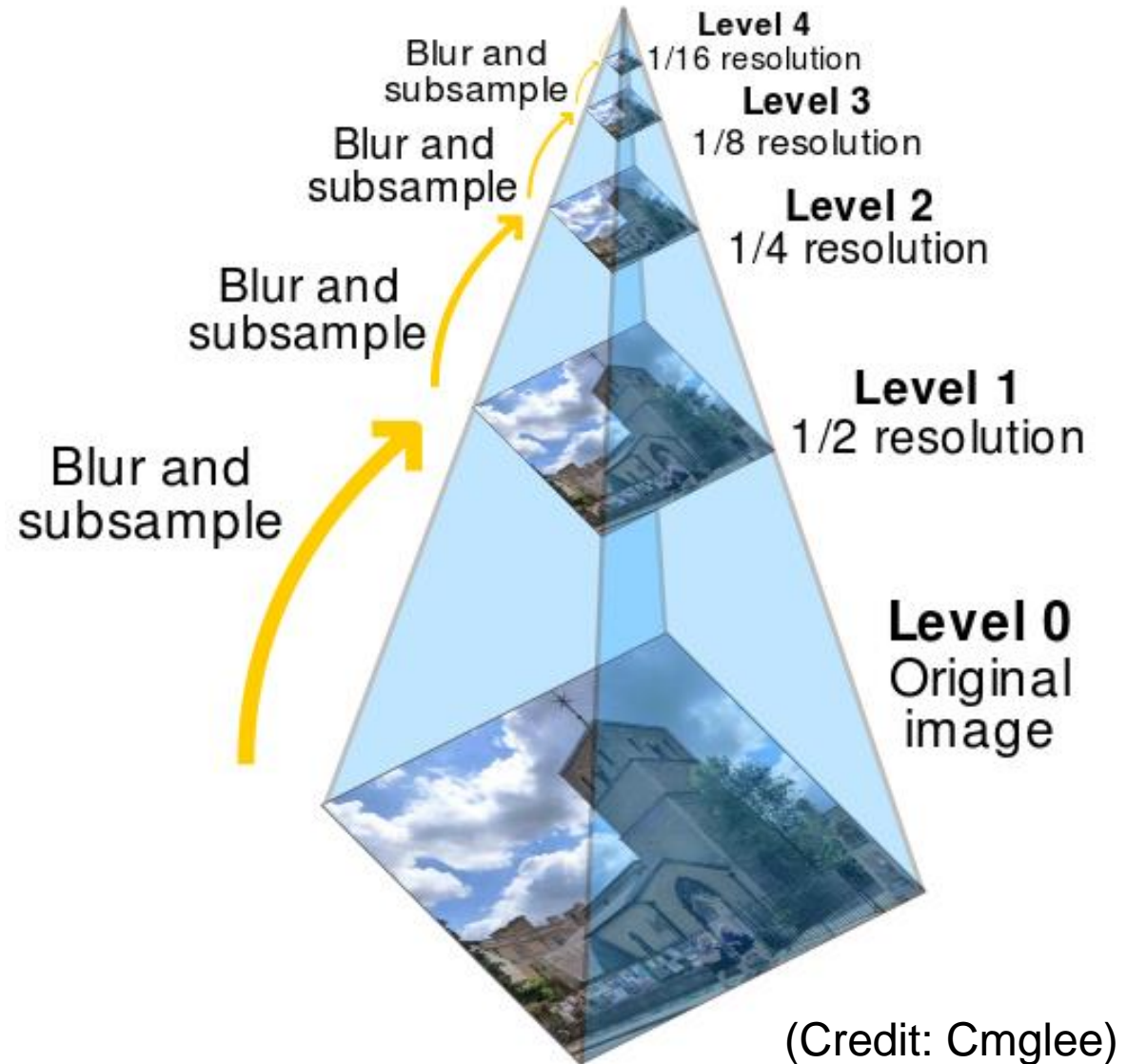
## Image Pyramids

Idar Dyrdal



# Image Pyramids

- Downsampling (decimation)
- Upsampling (interpolation)
- Pyramids
  - Gaussian Pyramids
  - Laplacian Pyramids (Lecture 2.3)
- Applications
  - Template matching (object detection)
  - Detecting stable points of interest
  - Image Registration
  - Compression
  - Image Blending
  - ...



# Image Scaling

- Assume that the image is too big for practical use:
  - Requires too much memory
  - Time consuming to process
  - Too big for the screen
  - ...
- A smaller image can be obtained by image sub-sampling



# Image Downsampling



1/2



1/4



1/8

Throw away every other row and column → image reduced to  $\frac{1}{2}$  size along each dimension.

# Image Downsampling



1/2



1/4 (2x zoom)

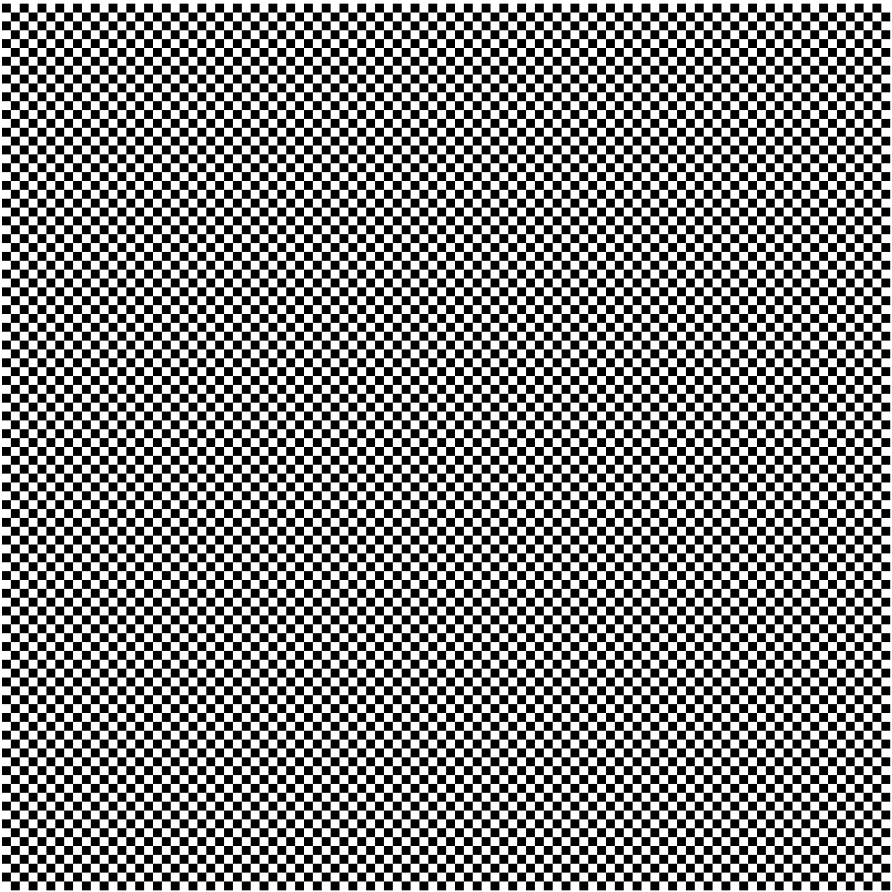


1/8 (4x zoom)

The subsampled images are of low quality. Why?

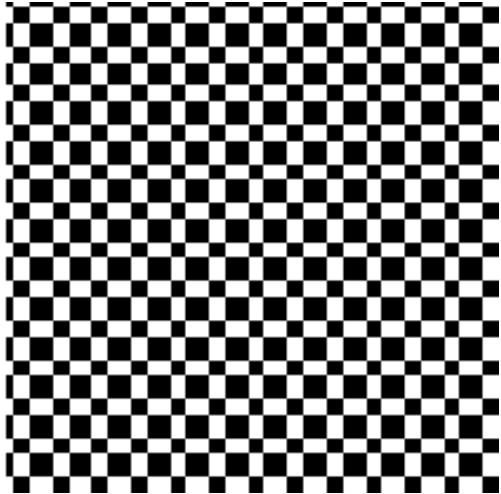


# Spatial undersampling

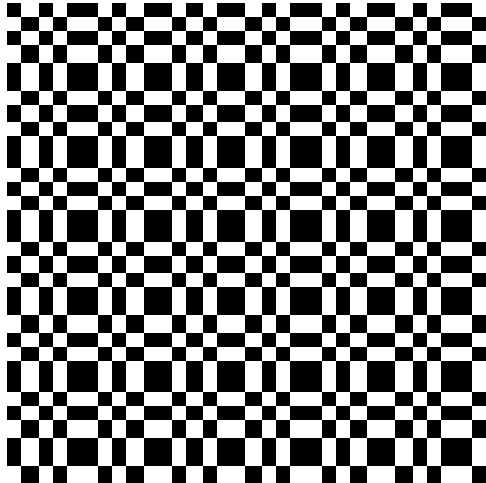


Checkerboard with 10 x 10 pixel squares

Downsampled images



1/10

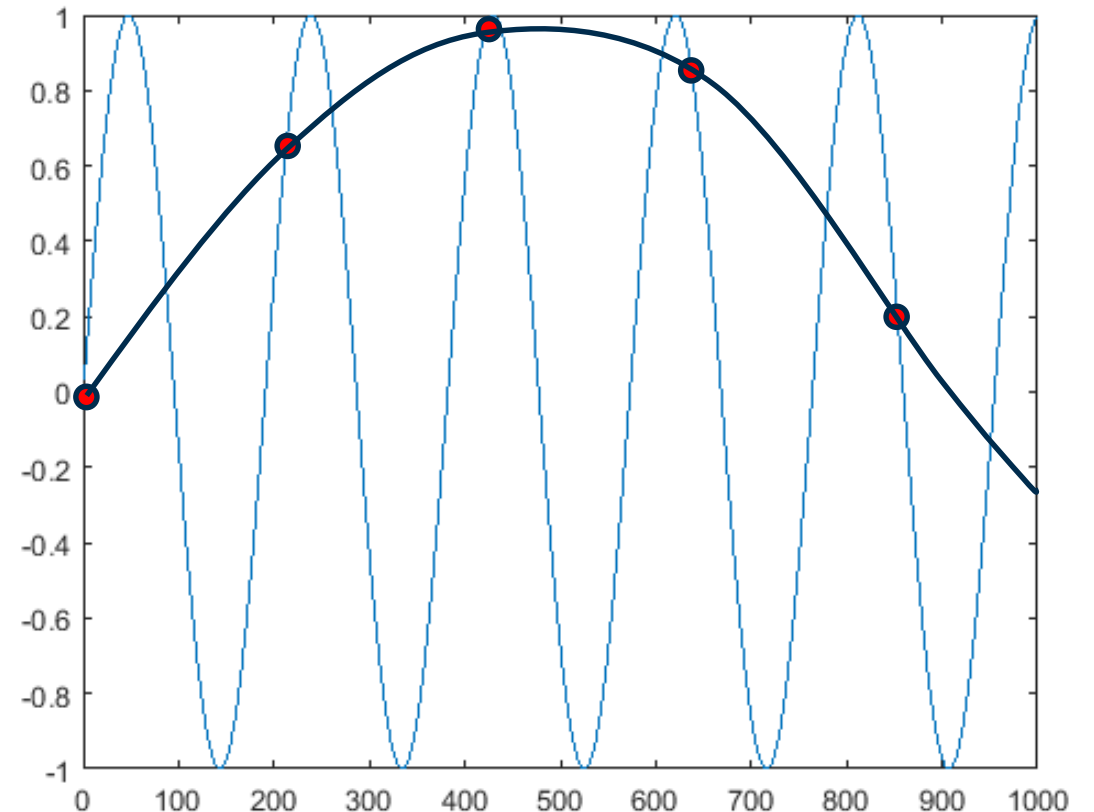


1/16

# Aliasing

- Occurs when the (spatial) sampling rate is not high enough to capture the details in the image
- High frequencies are transformed to lower frequencies (i.e. aliases)
- To avoid aliasing the sampling rate must be at least two times the maximum frequency in the image (at least two samples per cycle)
- This minimum sampling rate is called the **Nyquist rate**.

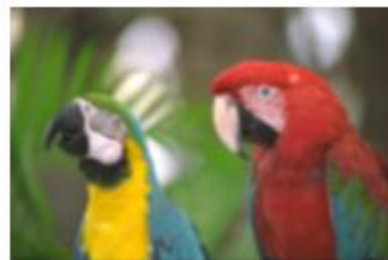
Aliasing can be avoided by low-pass filtering the image before downsampling



# Gaussian pre-filtering (low pass)



Gaussian 1/2



Gaussian 1/4



Gaussian 1/8



# Gaussian pre-filtering (low pass)



Gaussian 1/2



Gaussian 1/4



Gaussian 1/8

# Compared to downsampling without low-pass filtering...



1/2



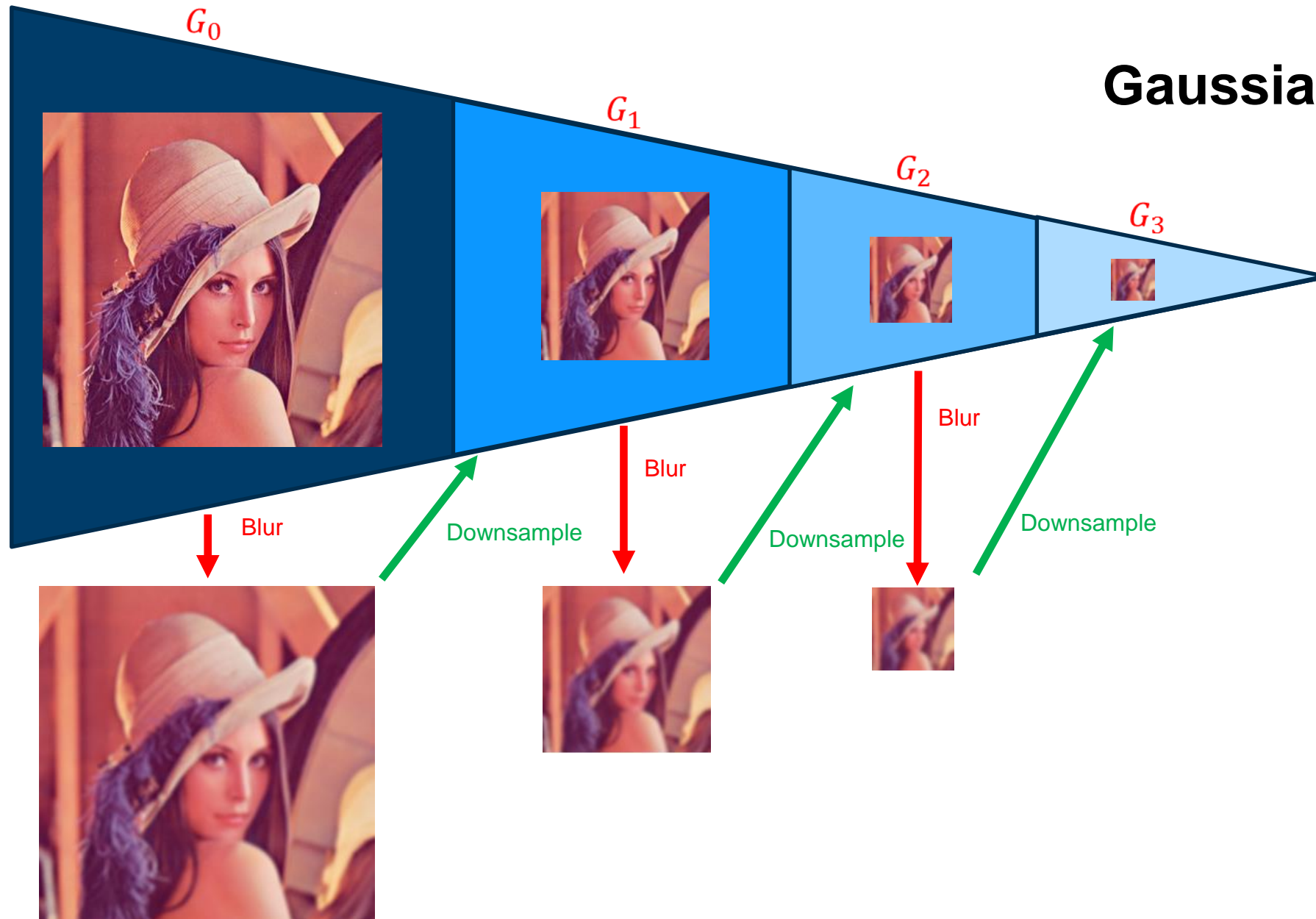
1/4



1/8

Conclusion: Low-pass filtering (i.e. smoothing with a Gaussian kernel) before subsampling the image!

# Gaussian Pyramid



# Upsampling

10 x magnification



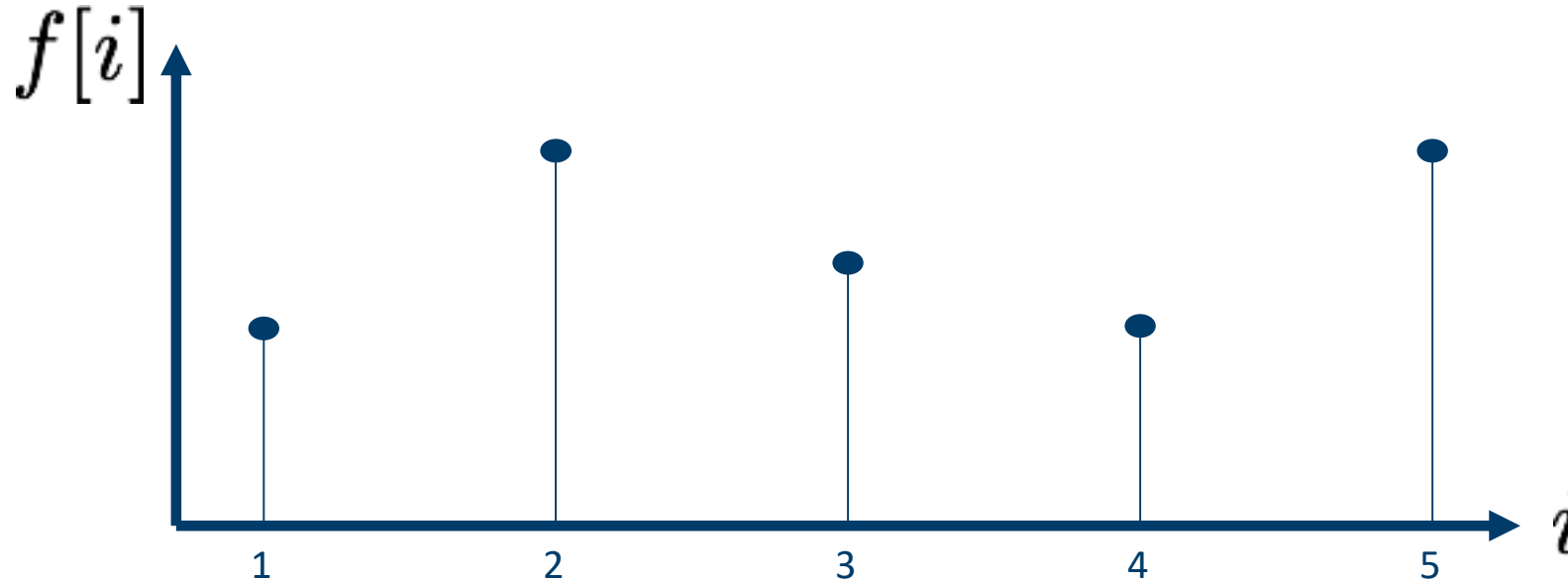
## Nearest neighbor interpolation:

- Repeat each row and column 10 times
- Fast and simple approach.

# Image interpolation

A digital image is a discrete point-sampling of a continuous function:

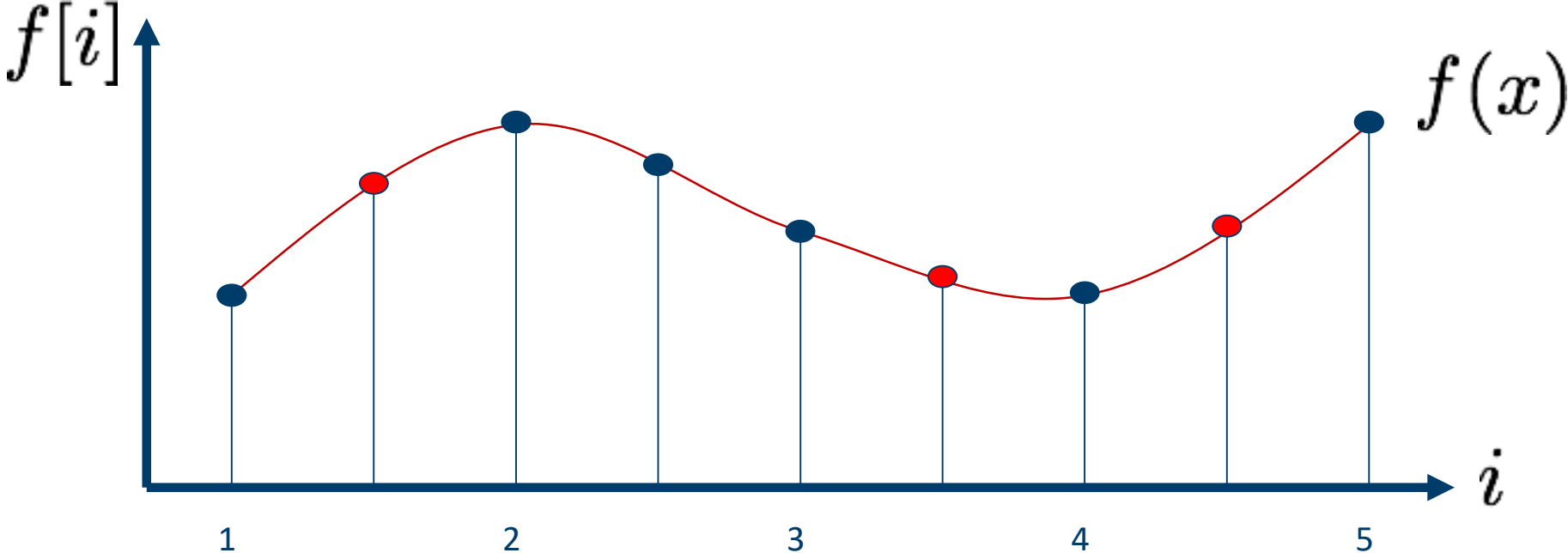
$$f[i, j] = \text{quantize}\{f(x, y)\} \text{ where } x = i\Delta x \text{ and } y = j\Delta y.$$



1D example

A new image could be generated, at any resolution and scale, if the original function could be reconstructed.

# Interpolation by convolution

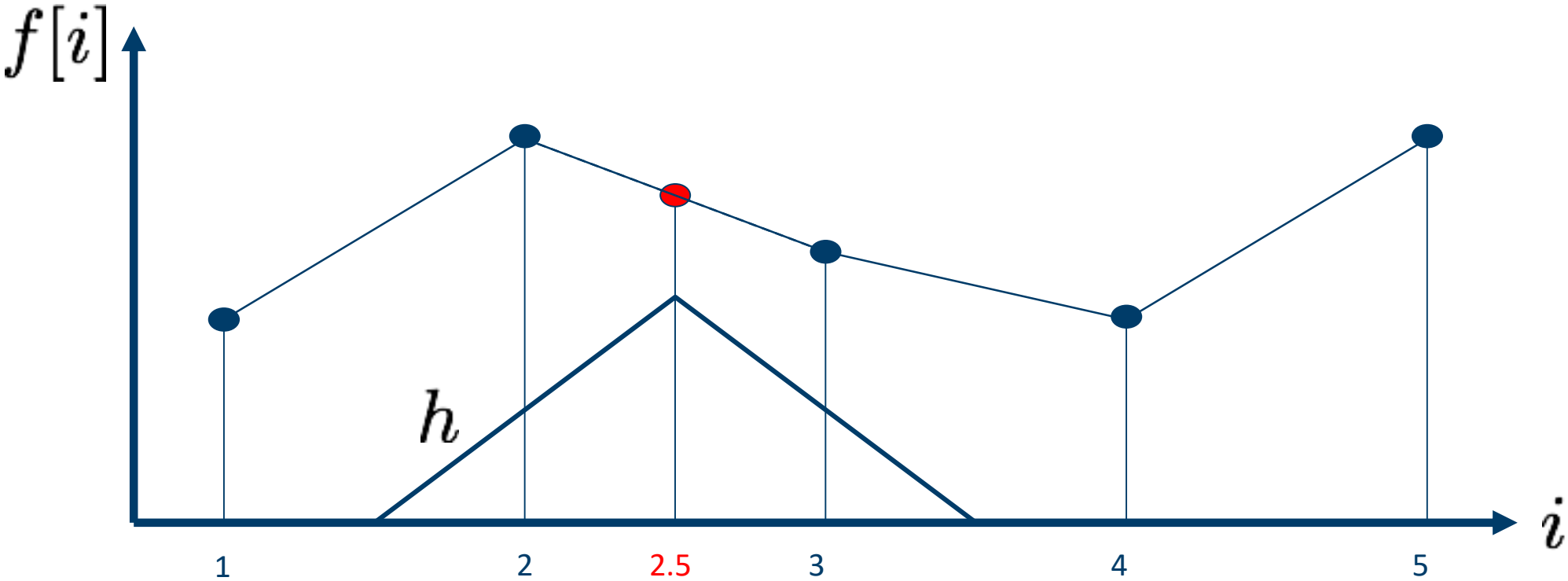


$$g[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k h[u, v] f[i - ru, j - rv]$$

r = scale factor

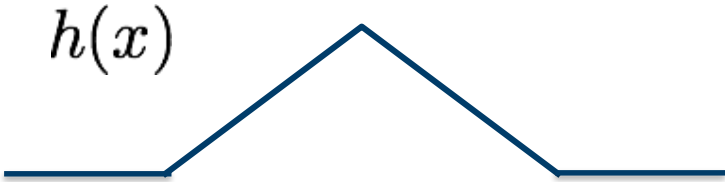


# Linear interpolation (bilinear for images)



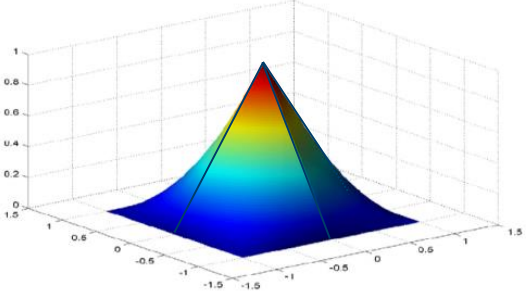
# Some kernels for signal and image interpolation

Linear:

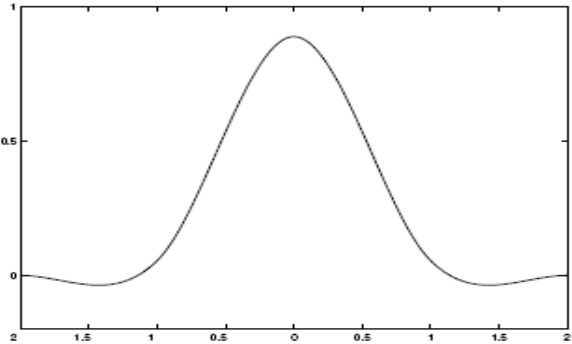


Bilinear:

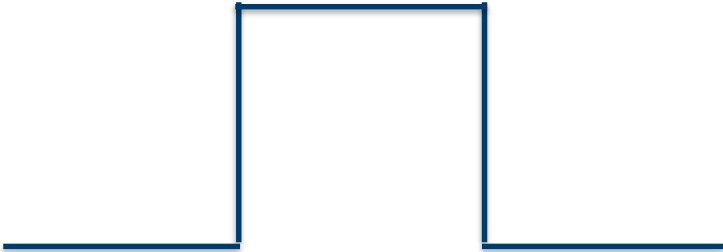
$$h(x, y)$$



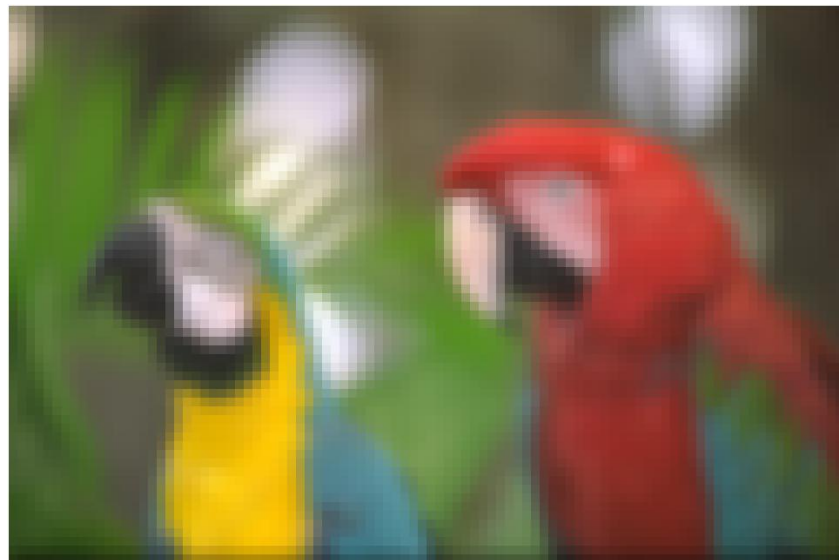
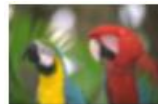
Bicubic (better choice for images):



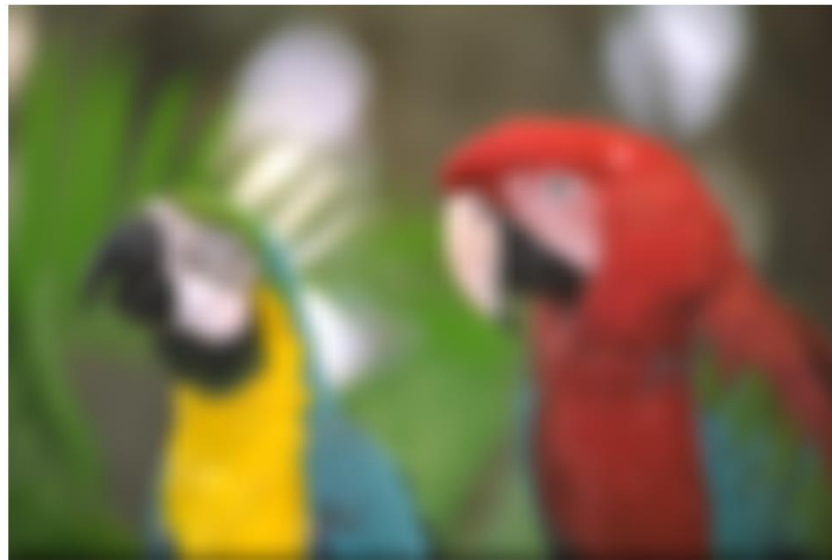
Nearest neighbor:



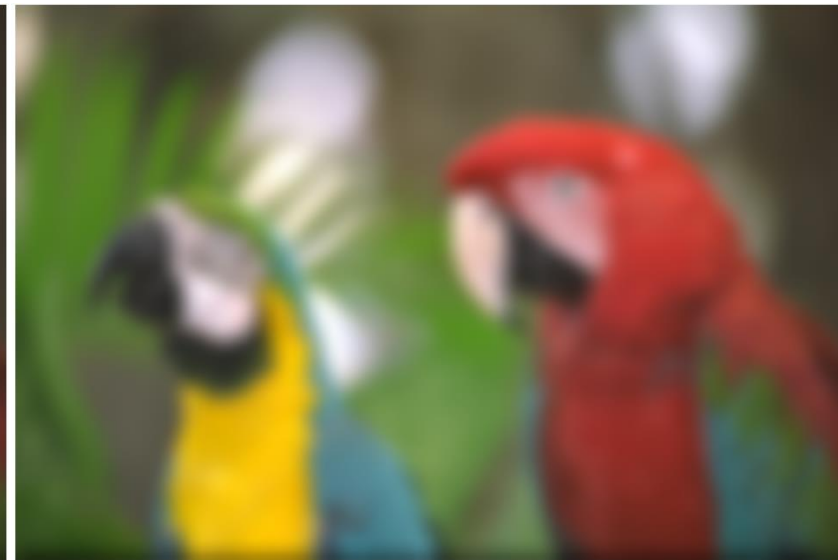
# Image interpolation - examples



Nearest neighbor



Bilinear



Bicubic

# Application: Template Matching with Image Pyramids

Input: Image, Template

1. Match template at current scale
2. Downsample image
3. Repeat 1-2 until image is very small
4. Take responses above some threshold, perhaps with non-maxima suppression.

# Summary

## Image Pyramids:

- Downsampling
- Upsampling
- Gaussian Pyramids

**More information:** Szeliski 3.5

