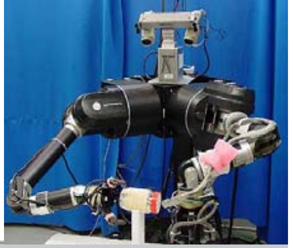
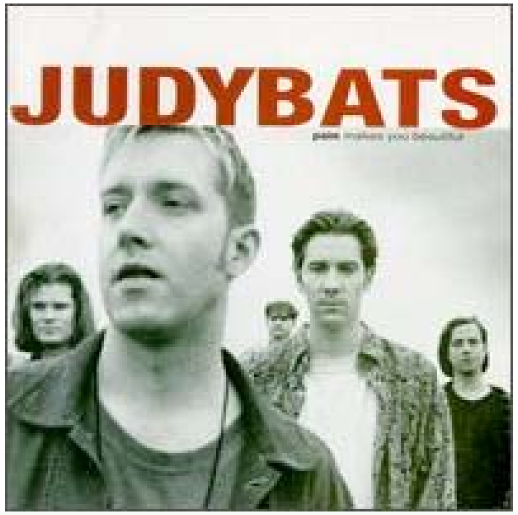
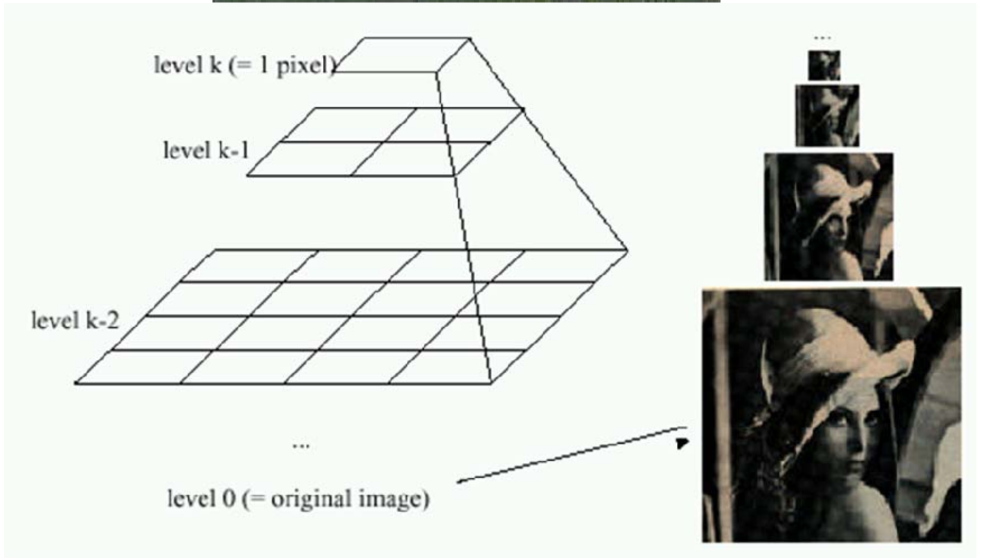
Einführung in Visual Computing
Unit 16: Multiscale Representations

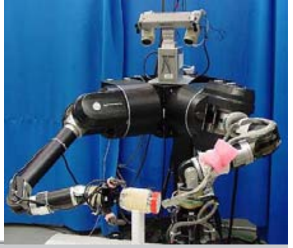


http://www.caa.tuwien.ac.at/cvl/teaching/sommersemester/evc

- Content:
  - Recap: Sampling
  - Multi-Resolution Image Representation
  - Gaussian Pyramid
  - The Laplacian Pyramid
  - Applications of Image Pyramids

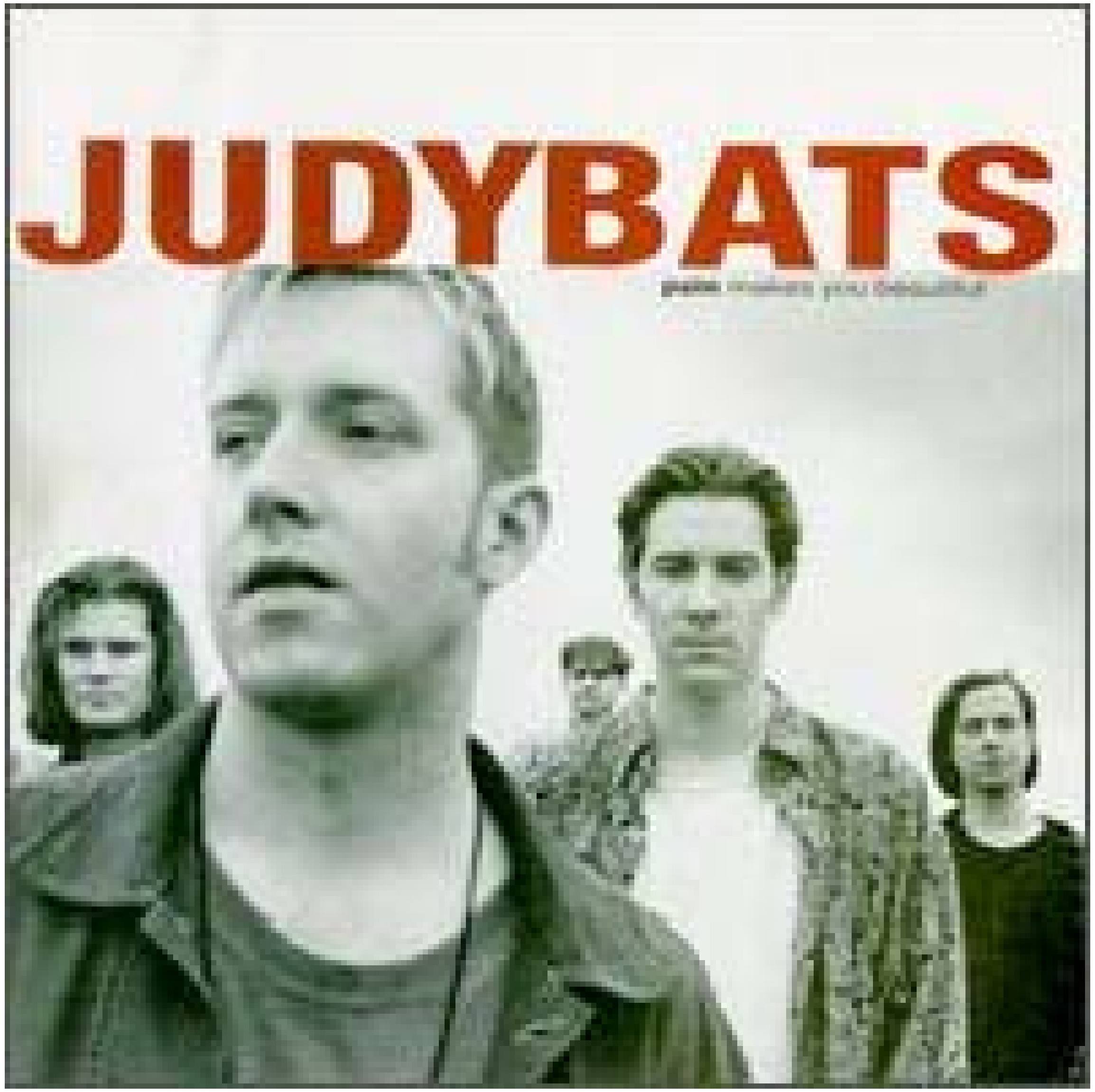




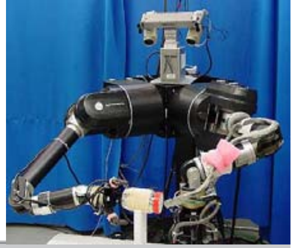


Multiscale Representations

- Image information occurs at all spatial scales



Scale Invariance



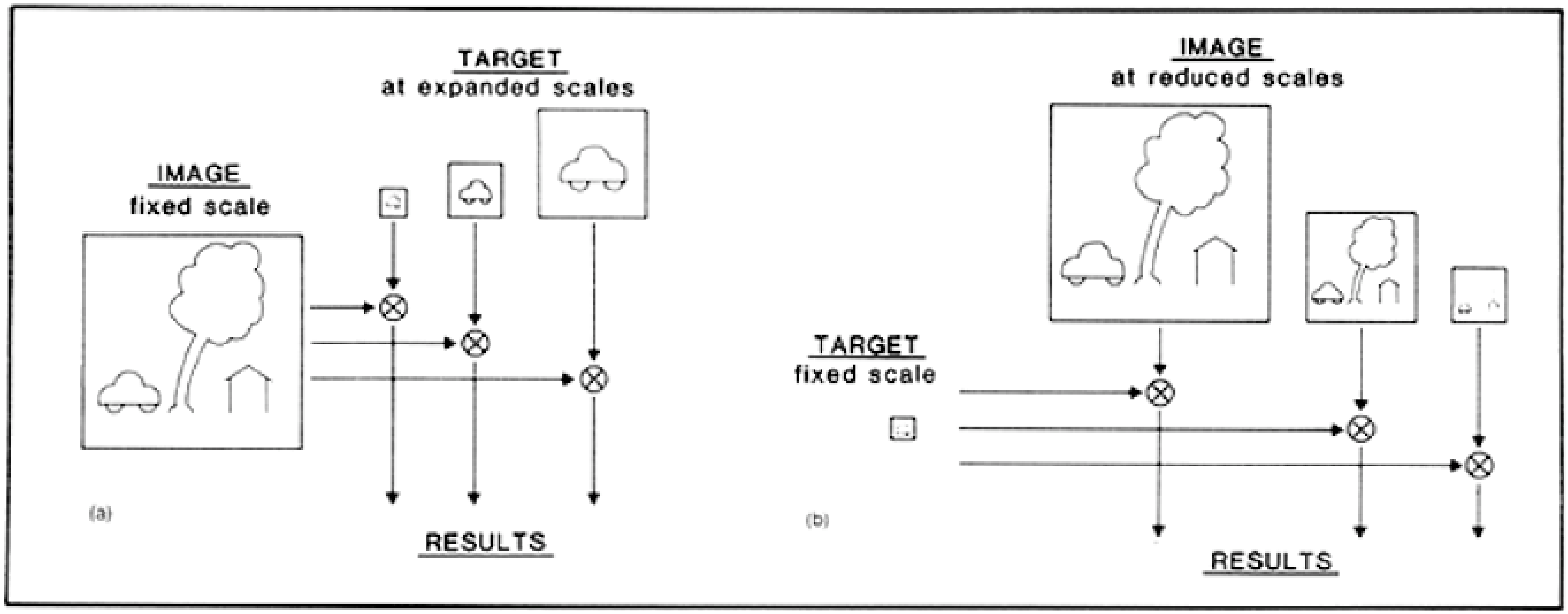
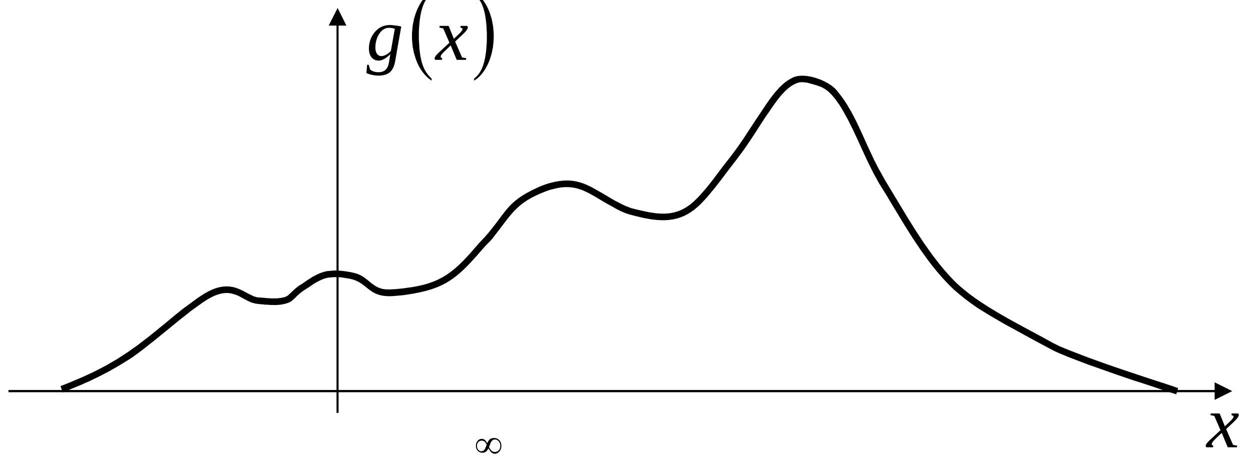


Fig. 1. Two methods of searching for a target pattern over many scales. In the first approach, (a), copies of the target pattern are constructed at several expanded scales, and each is convolved with the original image. In the second approach, (b), a single copy of the target is convolved with copies of the image reduced in scale. The target should be just large enough to resolve critical details The two approaches should give equivalent results, but the second is more efficient by the fourth power of the scale factor (image convolutions are represented by 'O').

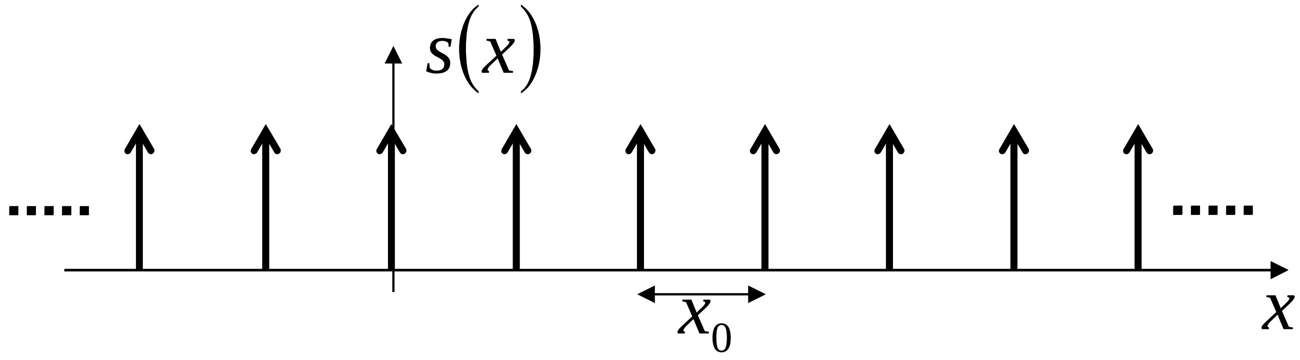
RECAP: Sampling

RECAP - Sampling Theorem

Continuous signal:



Shah function (Impulse train): $s(x) = \sum_{i=-\infty}^{\infty} \delta(x-i)$



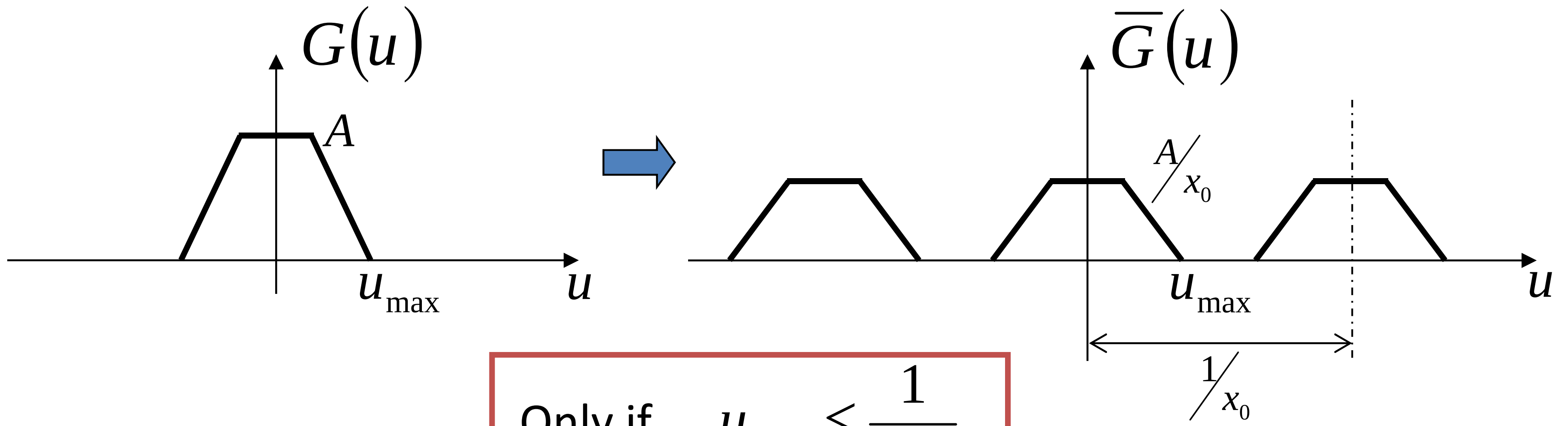
Sampled function: $\bar{g}(x) = g(x)s(x) = g(x)\sum_{i=-\infty}^{\infty} \delta(x-i)$

RECAP - Sampling Theorem

- Sampled function: $\bar{g}(x) = g(x)s(x) = g(x)\sum_{i=-\infty}^{\infty}\delta(x - ix_0)$
- Fourier transformed:

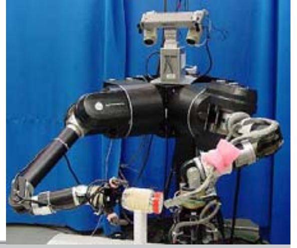
$$\overline{G}(u) = G(u) * S(u) = G(u) * \frac{1}{x_0}\sum_{i=-\infty}^{\infty}\delta\left(u - \frac{i}{x_0}\right)$$

Sampling frequency $\frac{1}{x_0}$

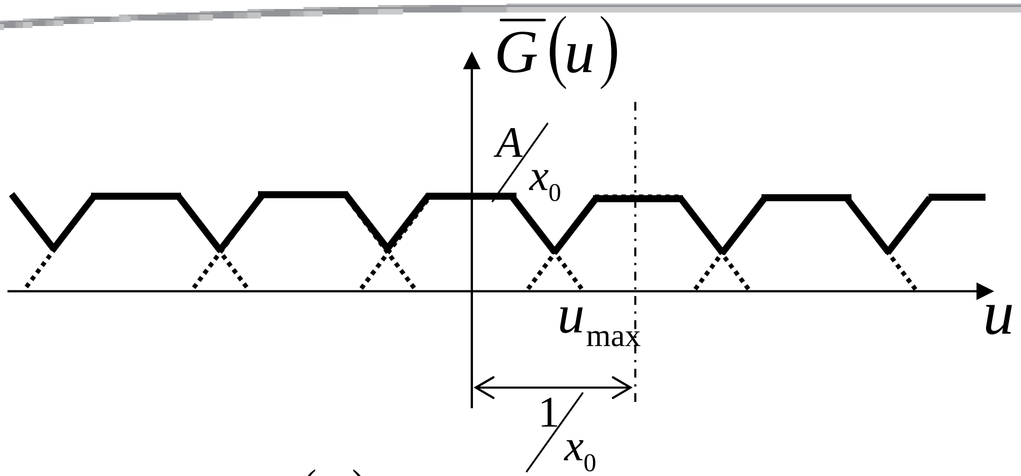


Only if $u_{max} \leq \frac{1}{2x_0}$

RECAP - Nyquist Theorem



If $u_{\max} > \dfrac{1}{2x_0}$



Aliasing

When can we recover $G(u)$ from $\overline{G}(u)$?

Only if $u_{\max} \le \dfrac{1}{2x_0}$ (Nyquist Frequency)

We can use

$$C(u) = \begin{cases} x_0 & |u| < 1/2x_0 \\ 0 & \text{otherwise} \end{cases}$$

Then $G(u) = \overline{G}(u)C(u)$ and $f(x) = \mathrm{IFT}[G(u)]$

Sampling frequency must be greater than $2u_{\max}$

Aliasing

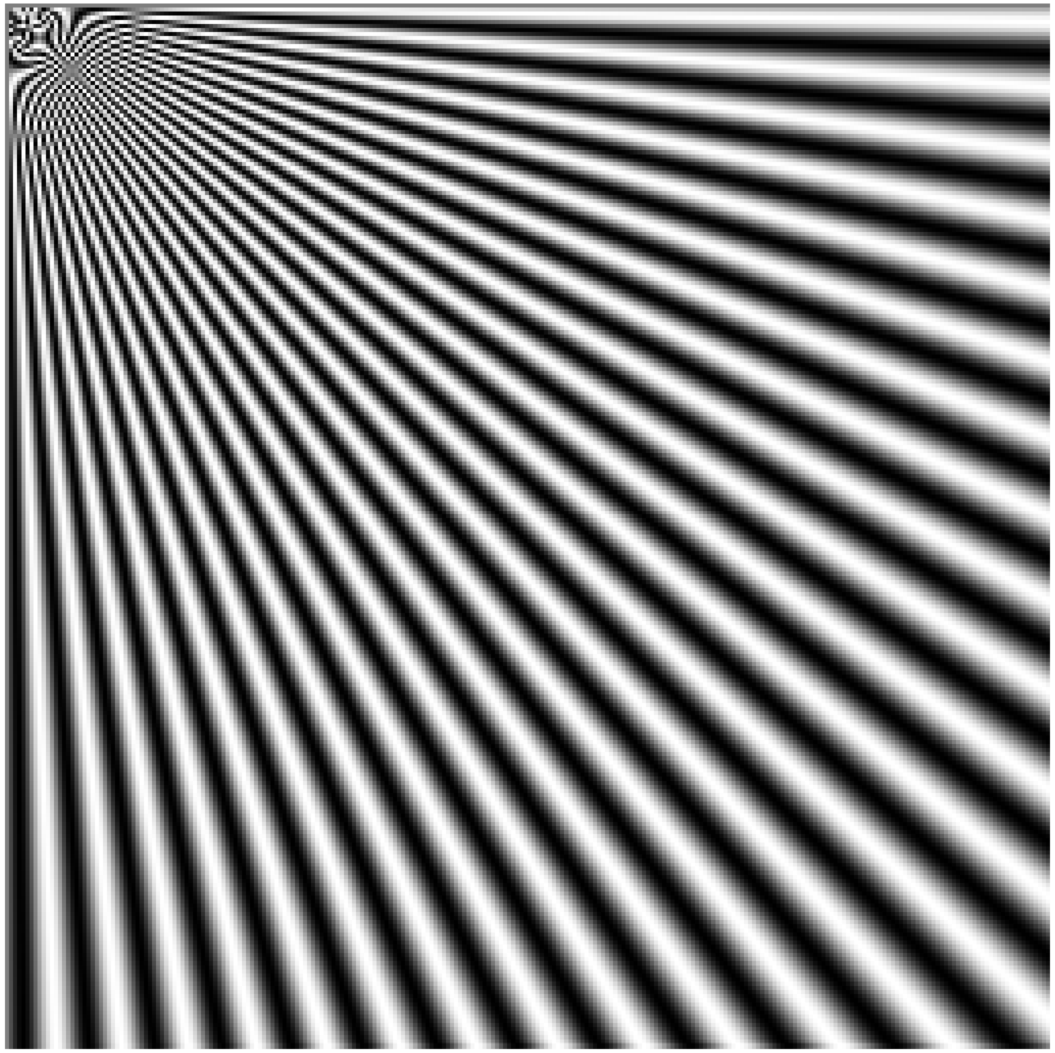
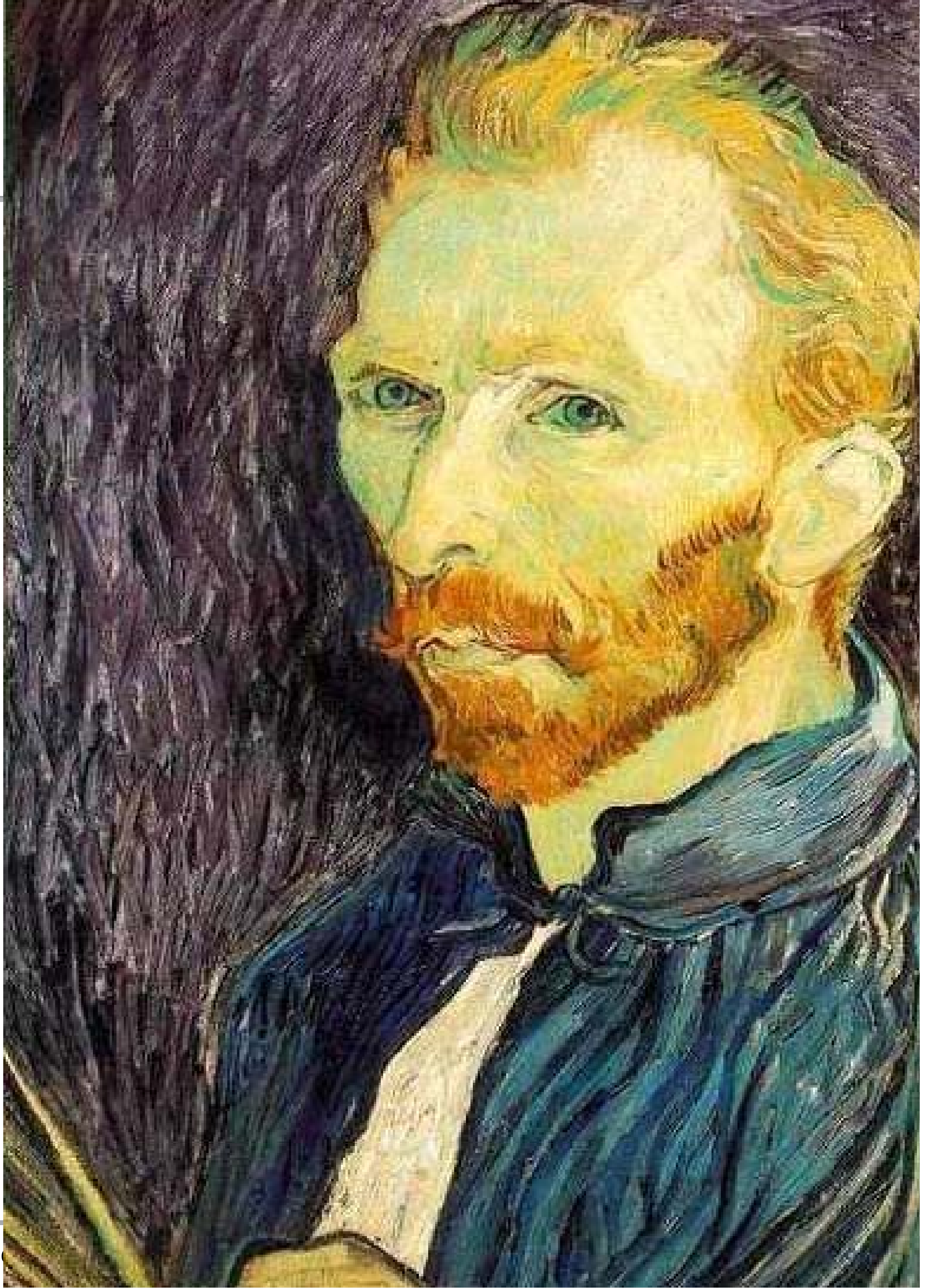


Image Scaling

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



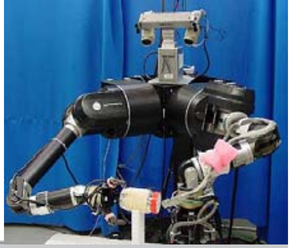
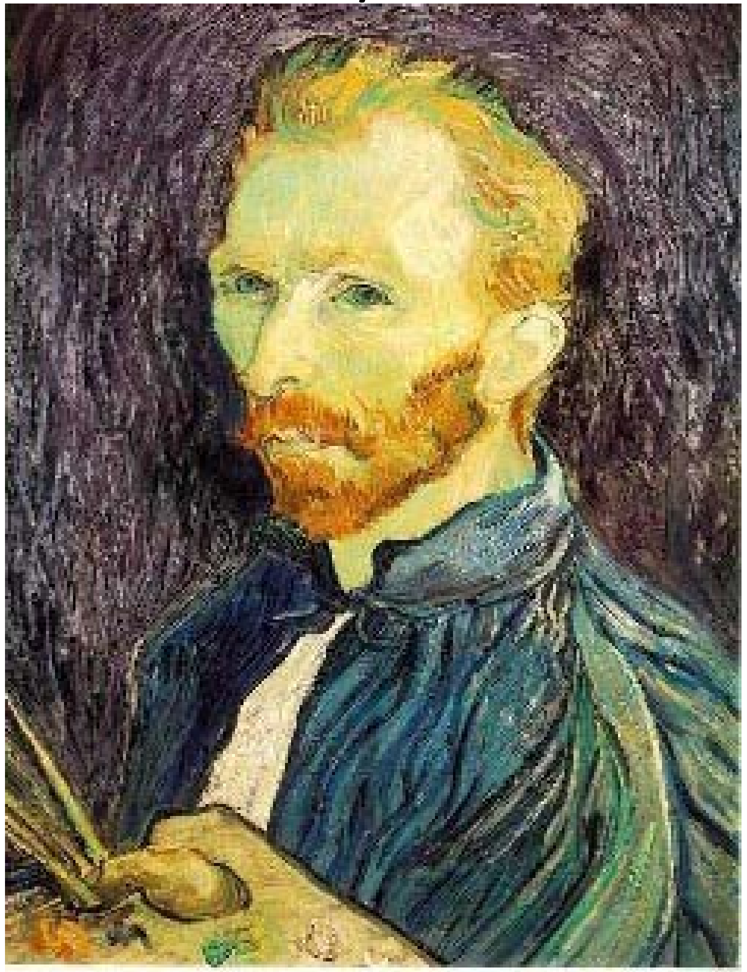


Image Sub-Sampling

1/2

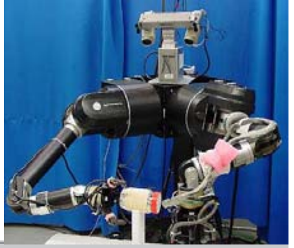


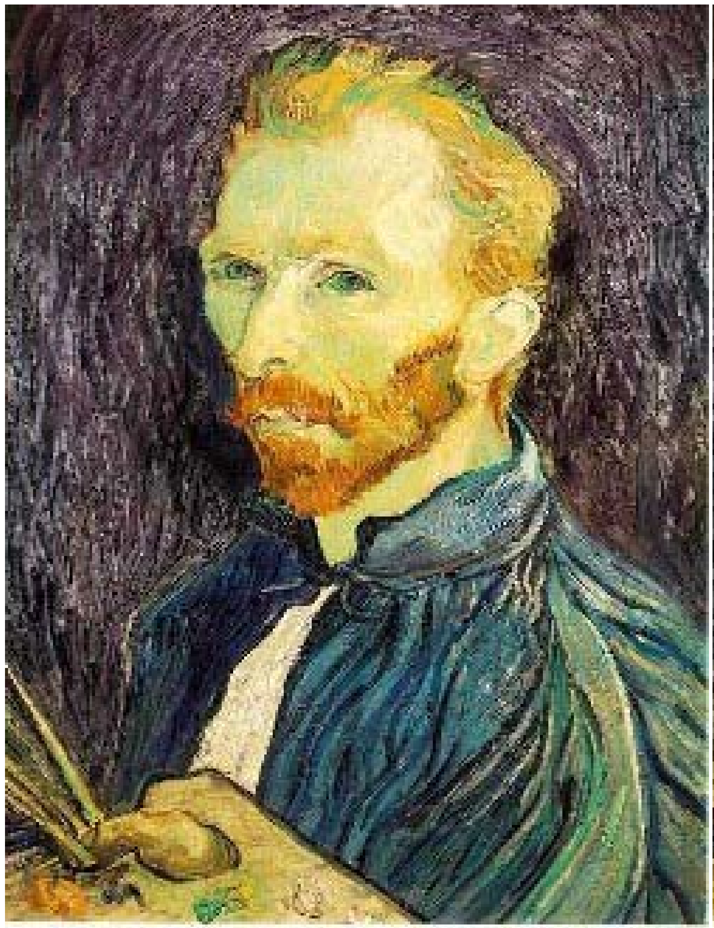
1/4

1/8

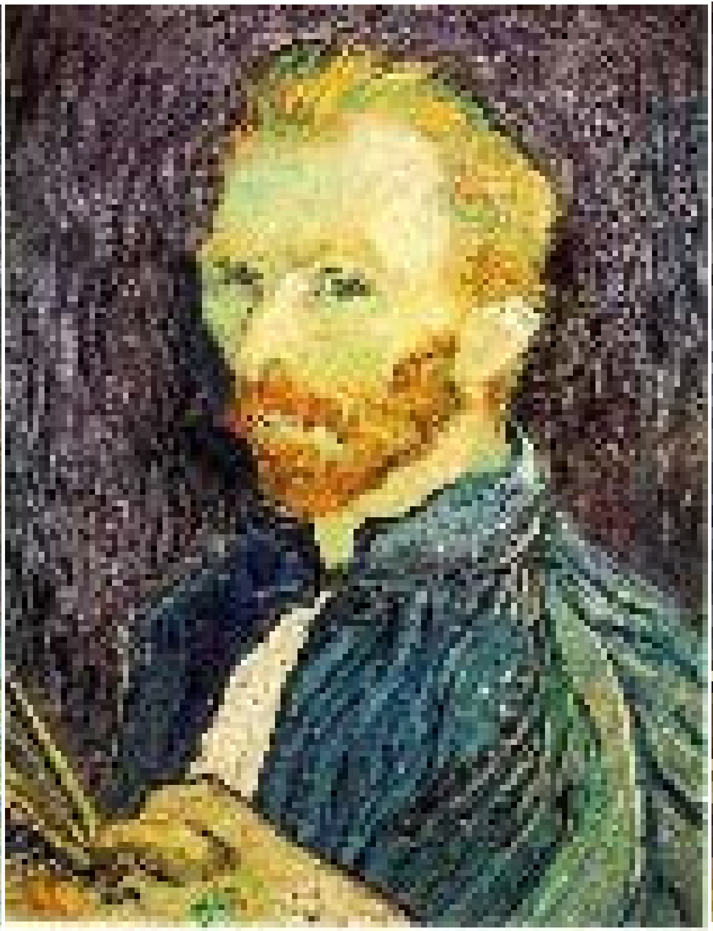
- Throw away every other row and column to create a 1/2 size (1 direction) image
  - - called image sub-sampling

Image Sub-Sampling

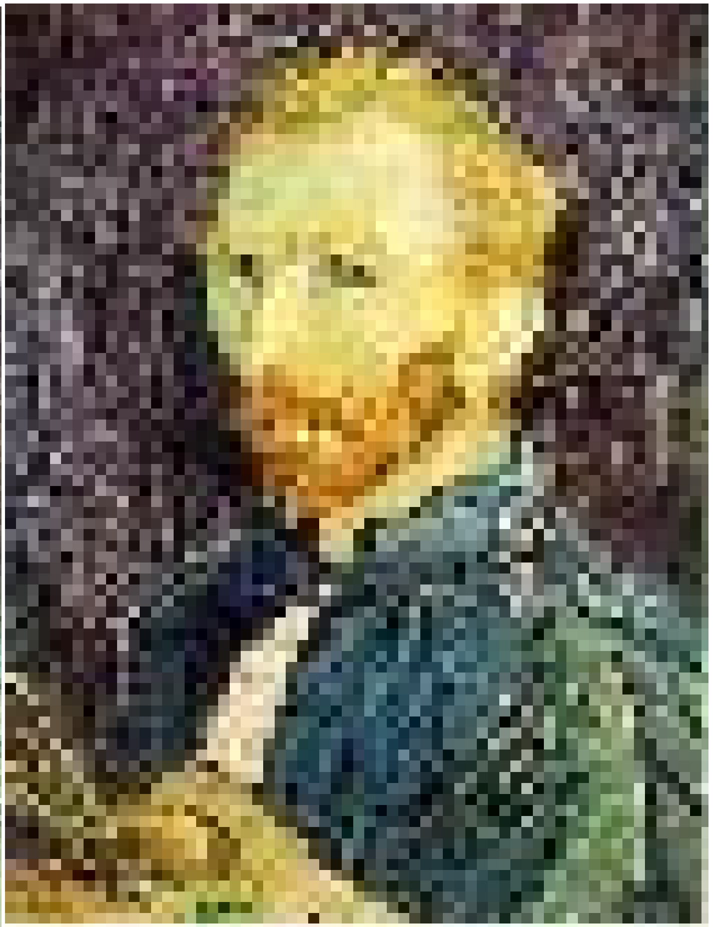




1/2

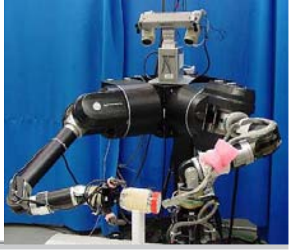


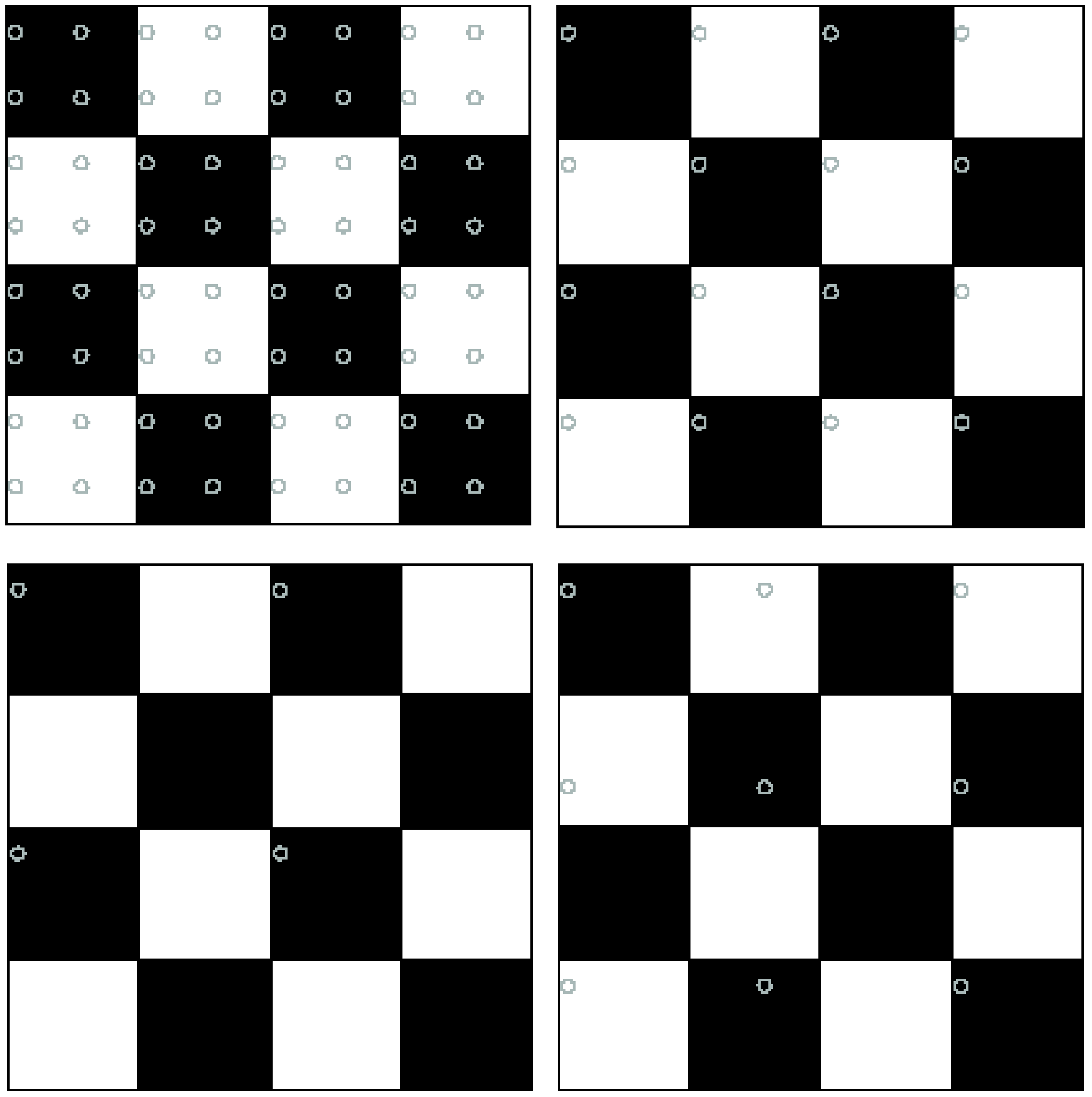
1/4 (2x zoom)



1/8 (4x zoom)

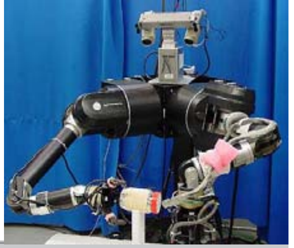
Good and Bad Sampling

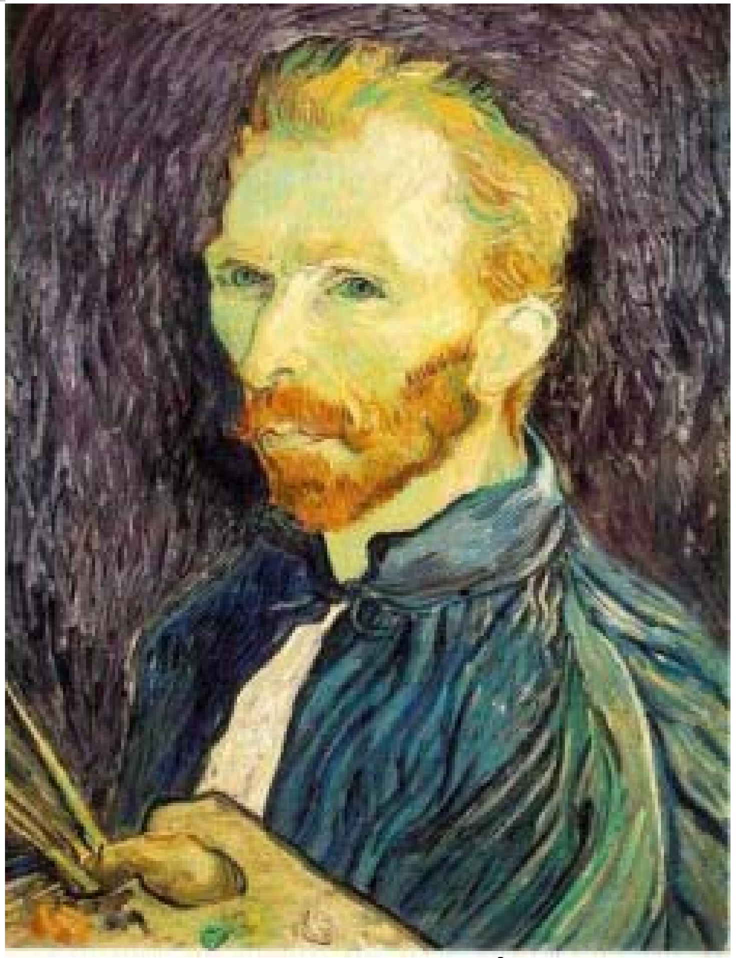




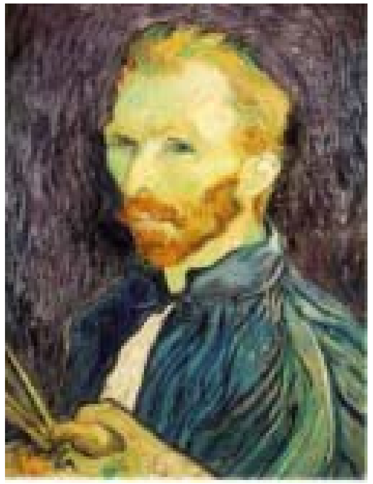
- Good sampling:
  - Sample often or,
  - Sample wisely
- Bad sampling:
  - see aliasing in action!

Sub-Sampling with Gaussian Pre-Filtering

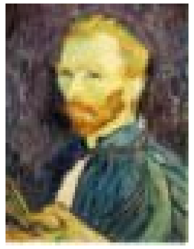




Gaussian 1/2



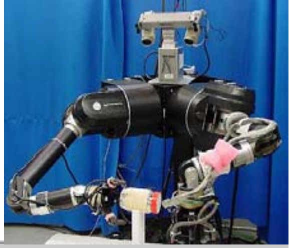
G 1/4

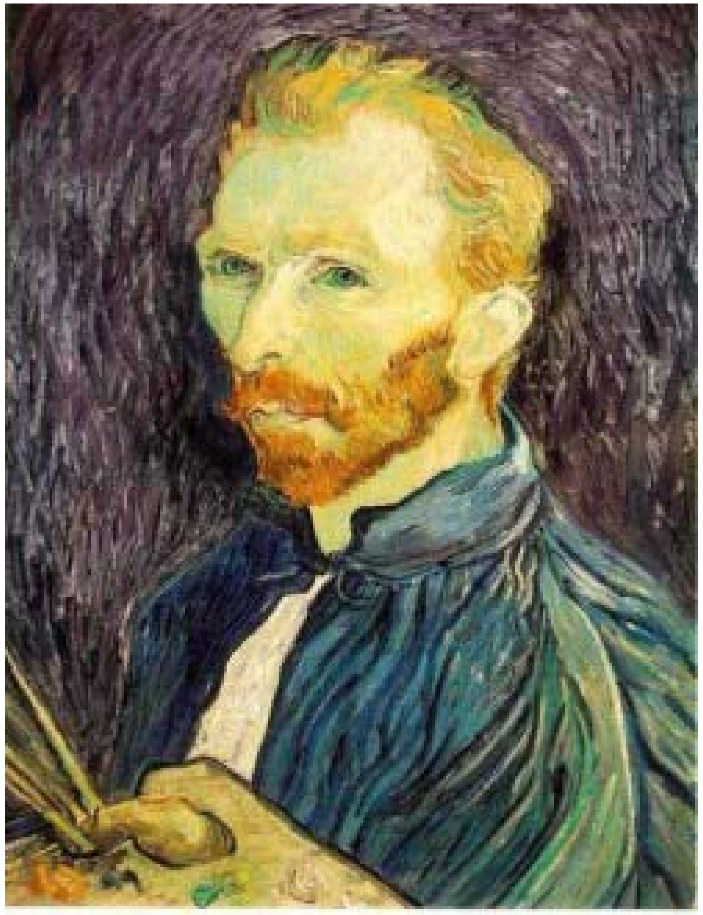


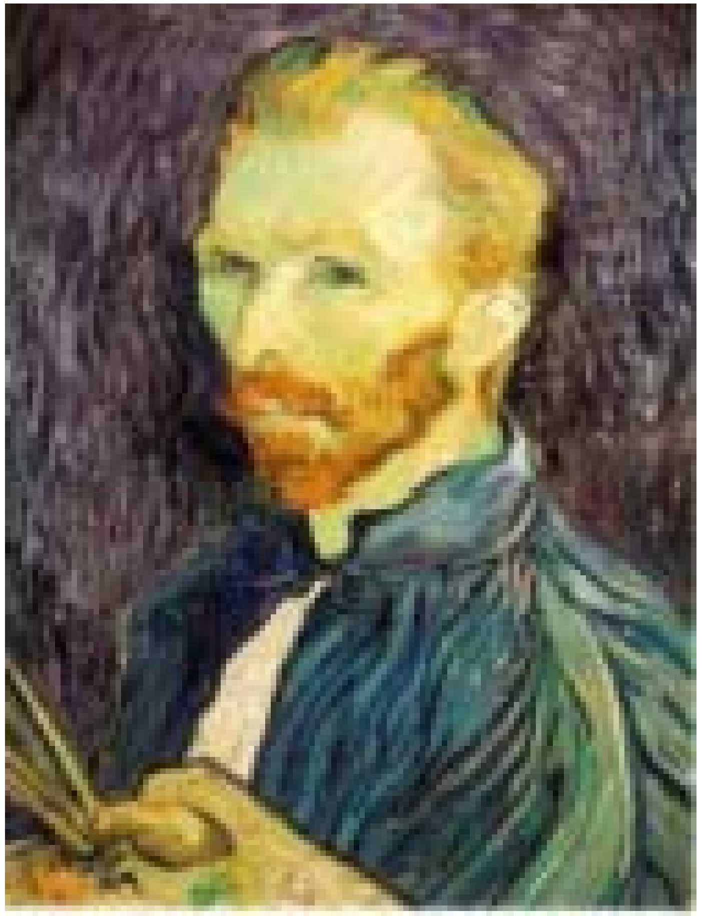
G 1/8

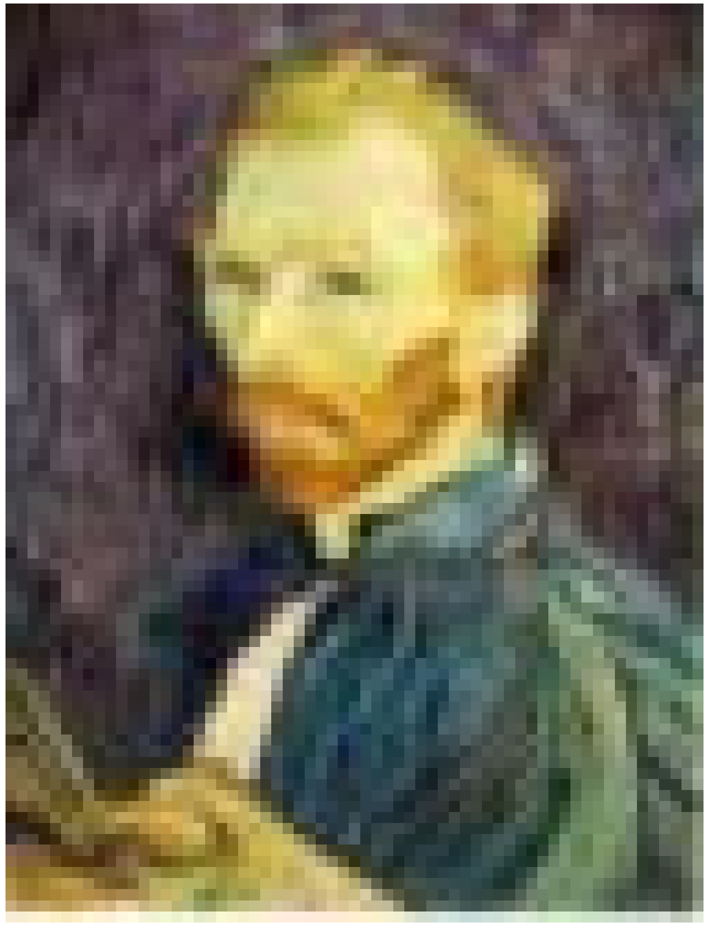
- Solution: filter the image, then subsample
  - Filter size should double for each ½ size reduction. Why?

Sub-Sampling with Gaussian Pre-Filtering

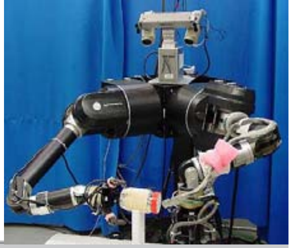


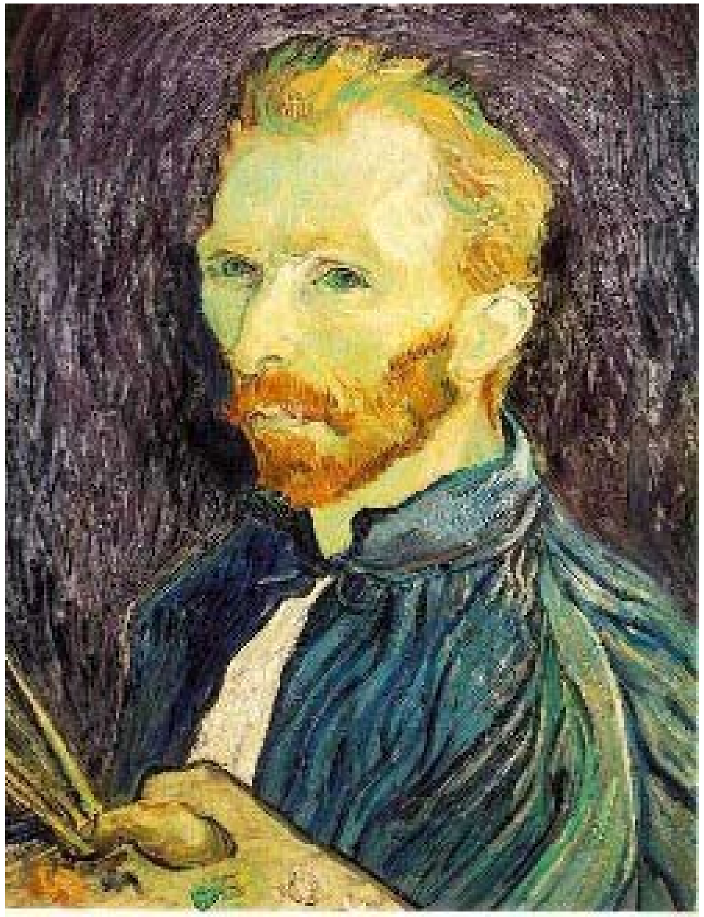

Gaussian 1/2


G 1/4

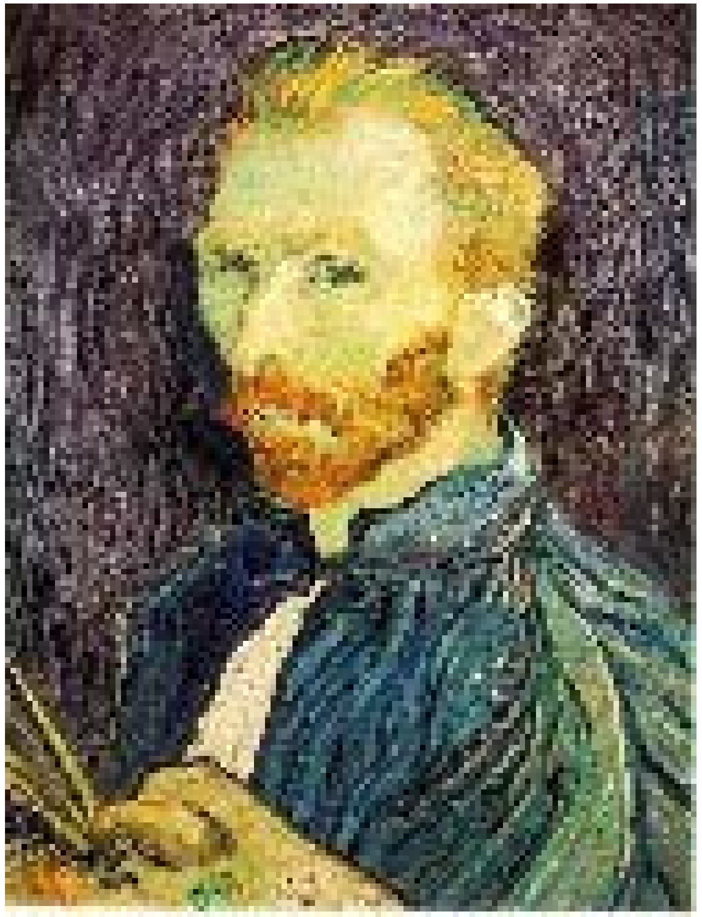

G 1/8

Compare with...

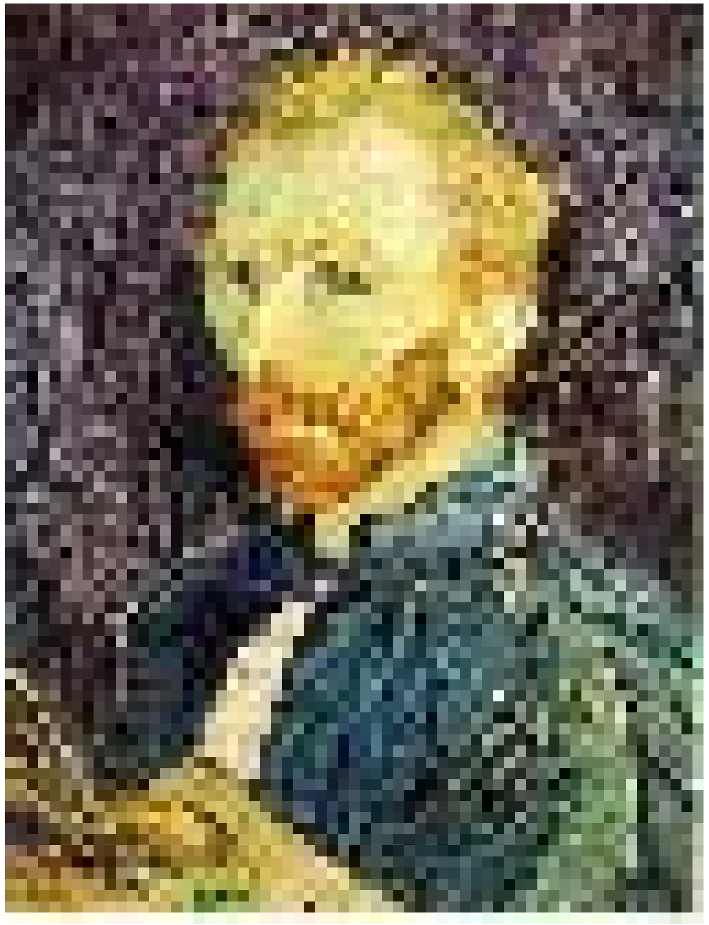




1/2



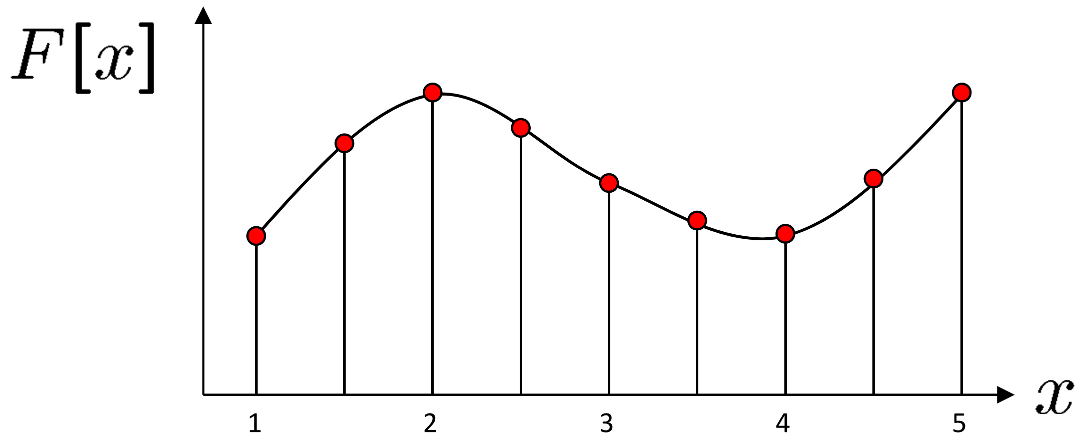
1/4 (2x zoom)



1/8 (4x zoom)

Image Resampling

- What about arbitrary scale reduction?
- How can we increase the size of the image?

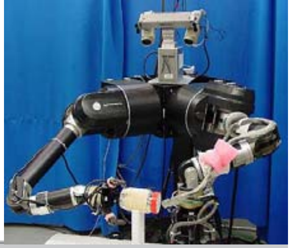


- Recall how a digital image is formed:

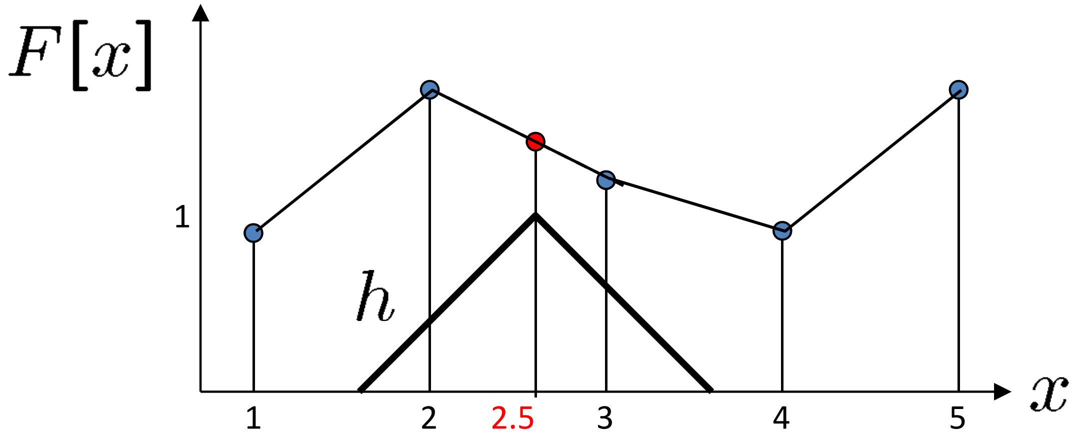
$$F[x,y] = \text{quantize}\{f(xd, yd)\}$$

  - It is a discrete point-sampling of a continuous function
  - If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Image Resampling



- So what to do if we don't know: f
  - Answer: guess an approximation $\tilde{f}$
  - Can be done in a principled way: filtering

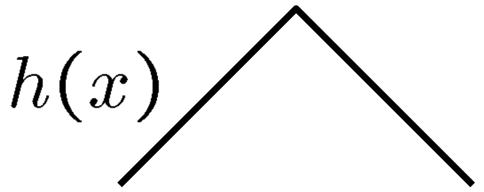


- Image reconstruction
  - Convert F to a continuous function

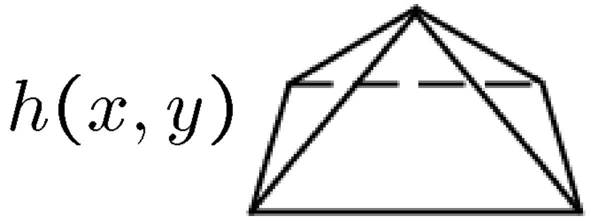
    $f_F(x) = F(\frac{x}{d})$ when $\frac{x}{d}$ is an integer, 0 otherwise
  - Reconstruct by convolution: $\tilde{f} = h \otimes f_F$

Resampling Filters

- What does the 2D version of this hat function look like?



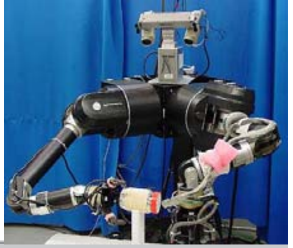
performs
linear interpolation



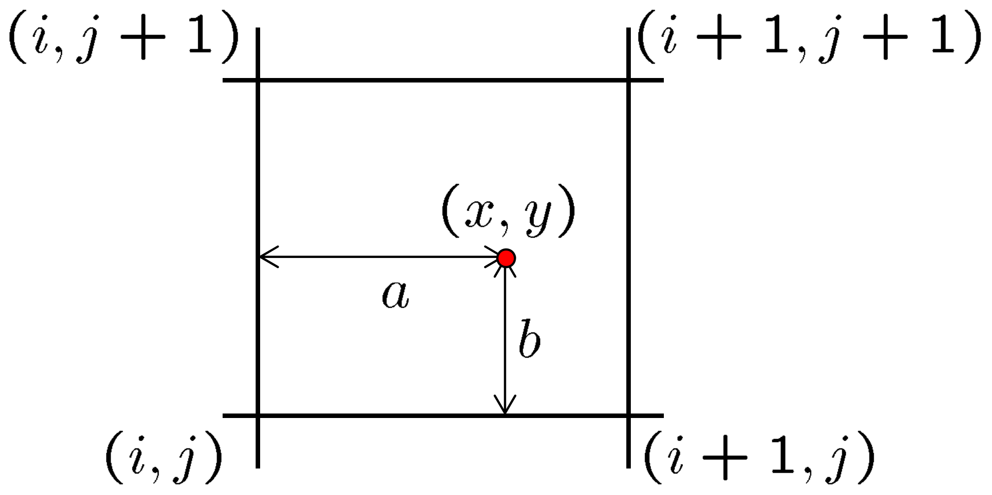
performs
bilinear interpolation

- Better filters give better resampled images
  - Bicubic is common choice

Bilinear interpolation

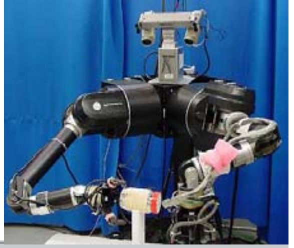


- A common method for resampling images



$$
\begin{aligned}
F(x,y) = \; & (1-a)(1-b) && F(i,j) \\
& +a(1-b) && F(i+1,j) \\
& +ab && F(i+1,j+1) \\
& +(1-a)b && F(i,j+1)
\end{aligned}
$$

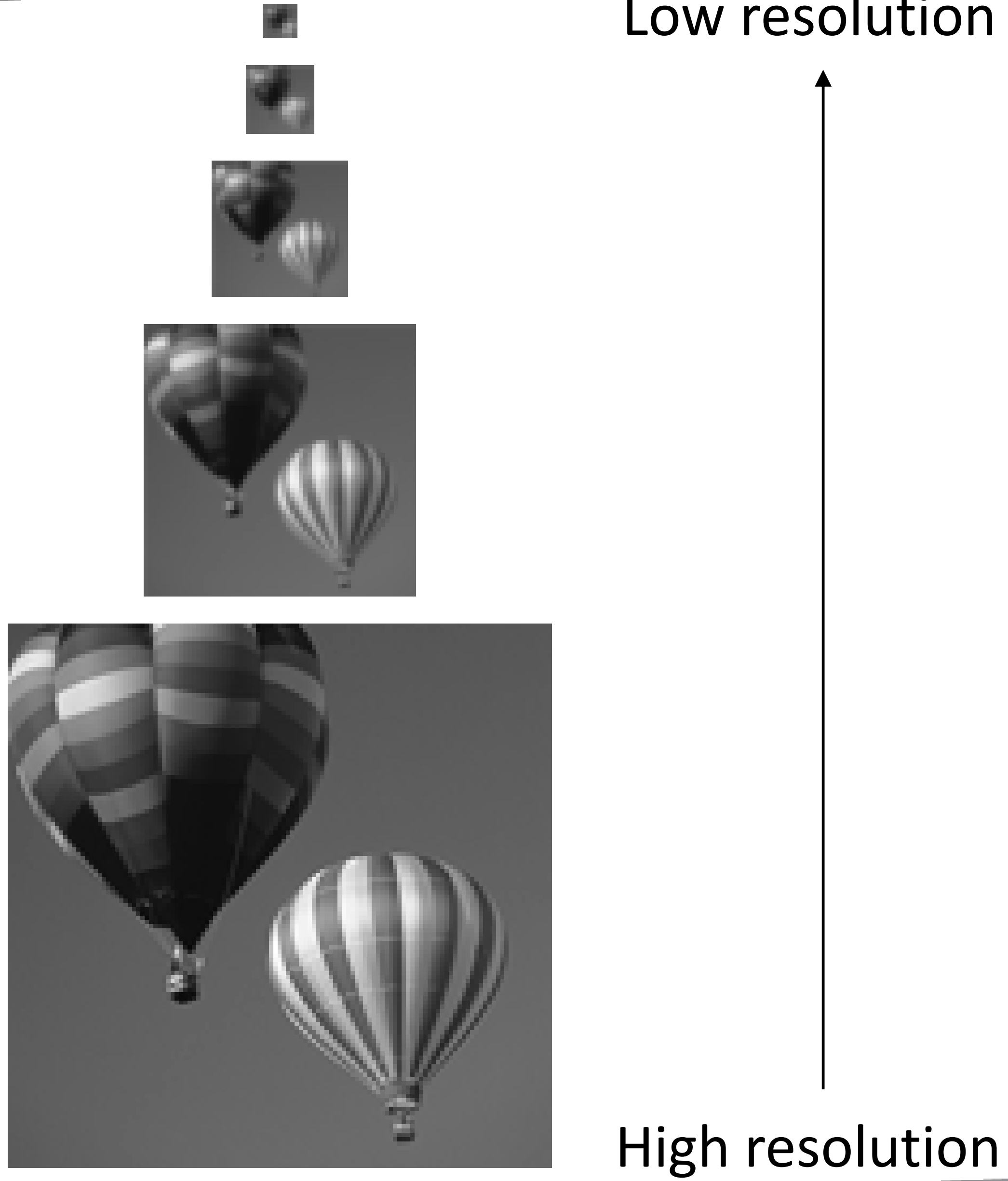
Multi-Resolution Image Representation



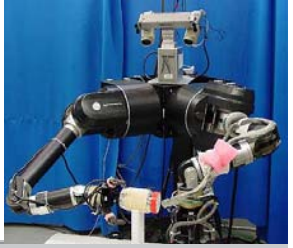
Multi-Resolution Image Representation

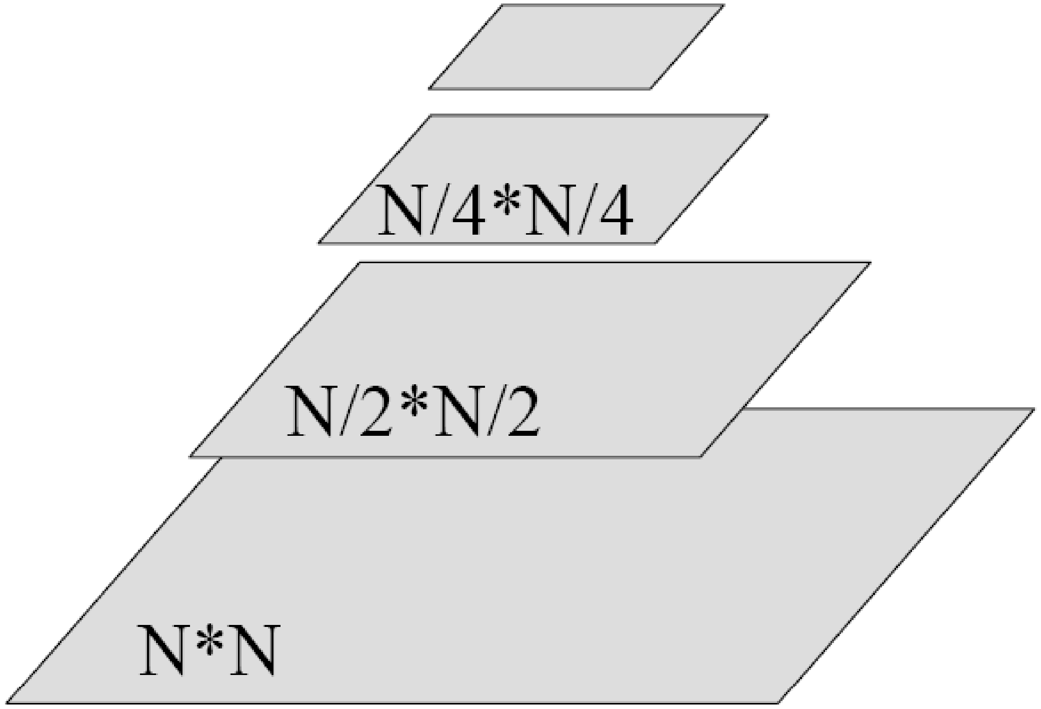
- Fourier domain tells us “what” (frequencies, sharpness, texture properties), but not “where”.
- Spatial domain tells us “where” (pixel location) but not “what”.
- We want a image representation that gives a local description of image “events” – what is happening where.
- Naturally, think about representing images across varying scales.

Multi-resolution Image Pyramids

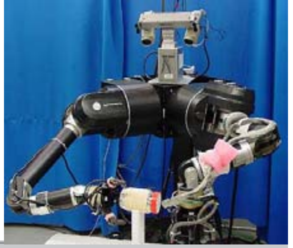


Space Required for Pyramids

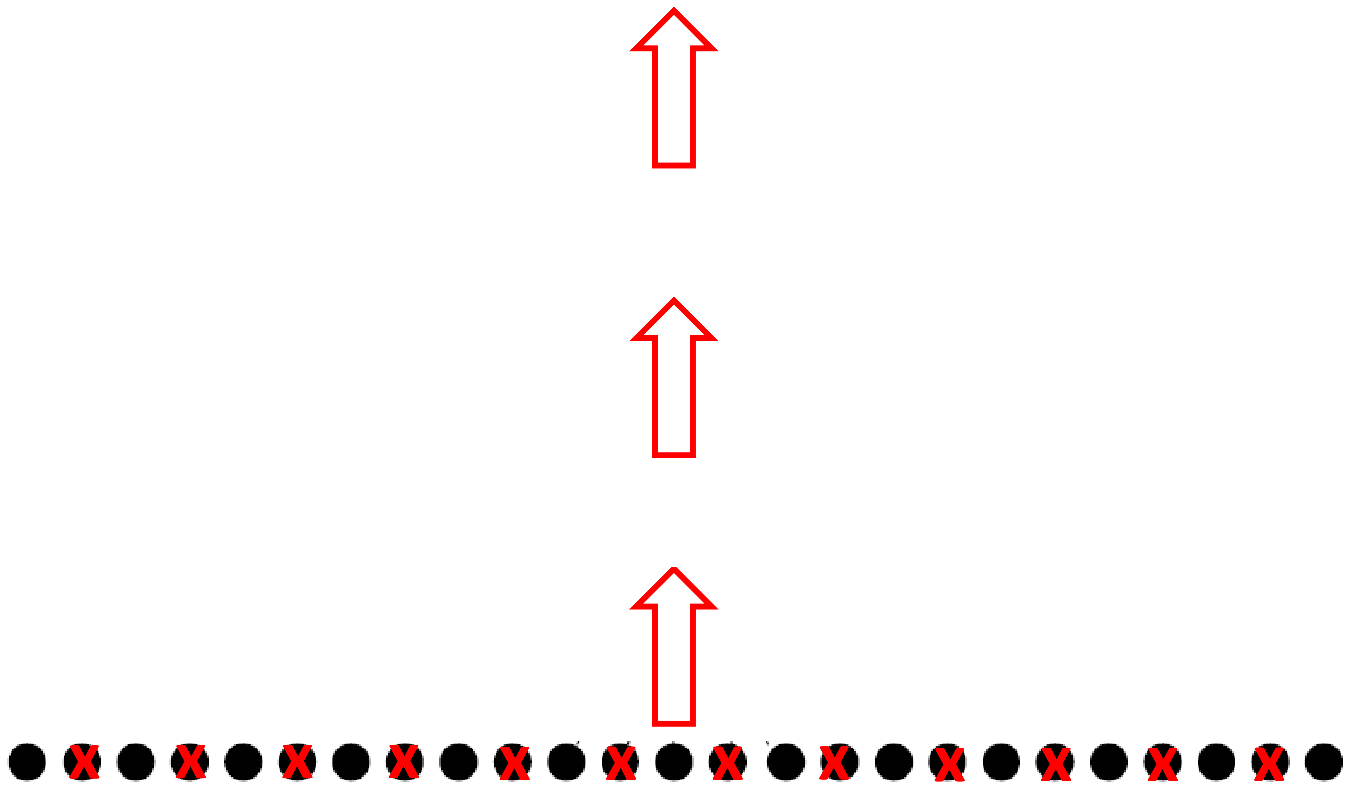




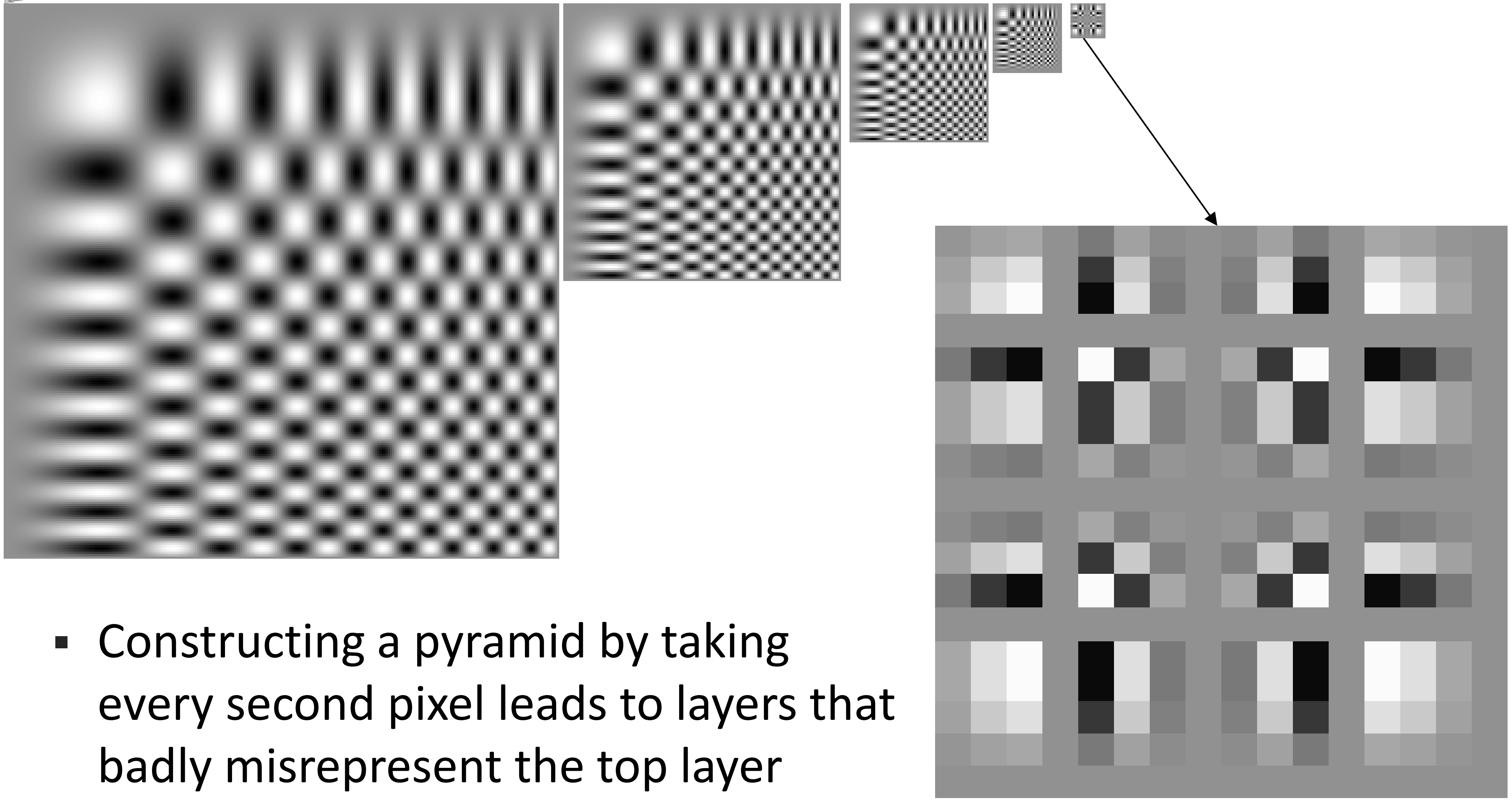
$$N^2 + \frac{1}{4}N^2 + \frac{1}{16} + N^2 + \cdots = 1\frac{1}{3}N^2$$



Pyramid Construction

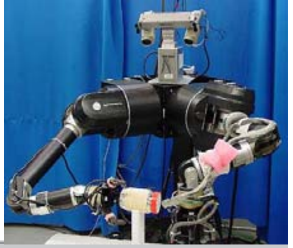


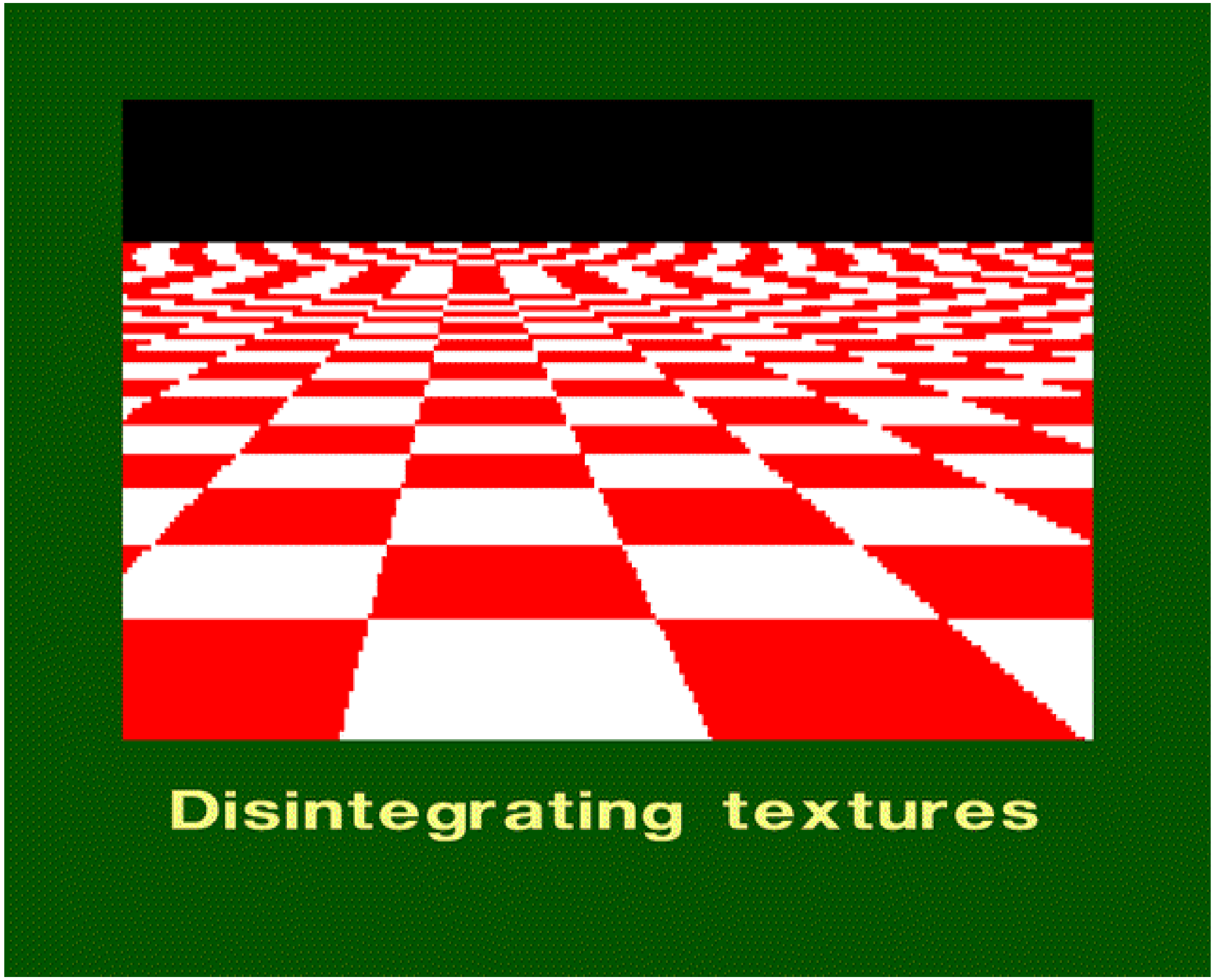
Pyramid Construction



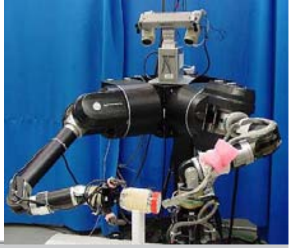
- Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer

Even worse for Synthetic Images



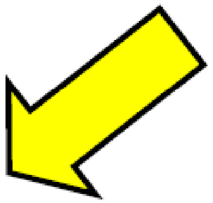


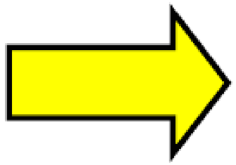
Decimation



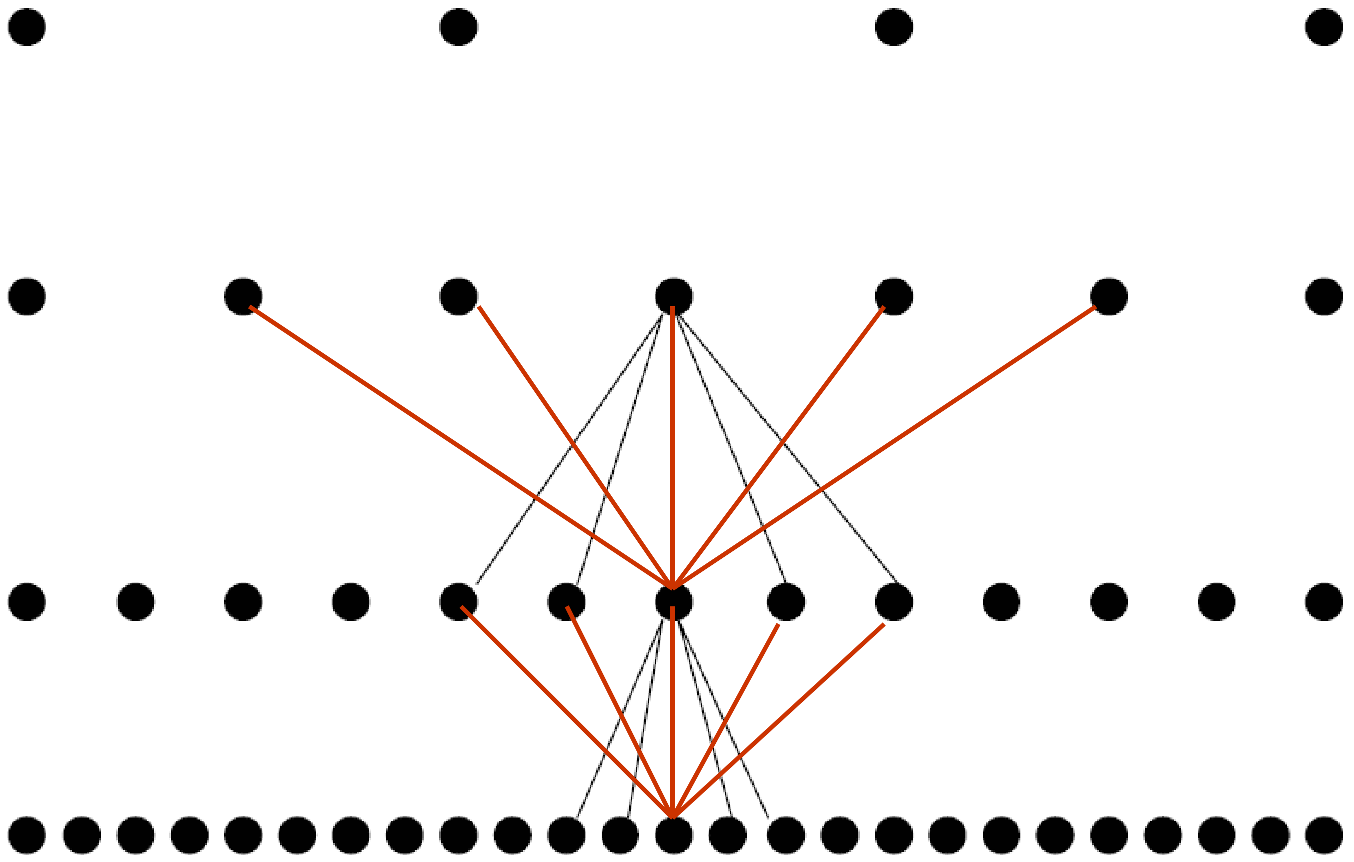
$*$

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

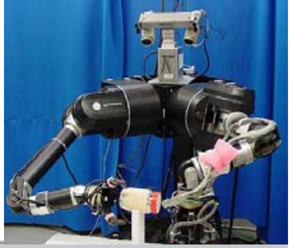


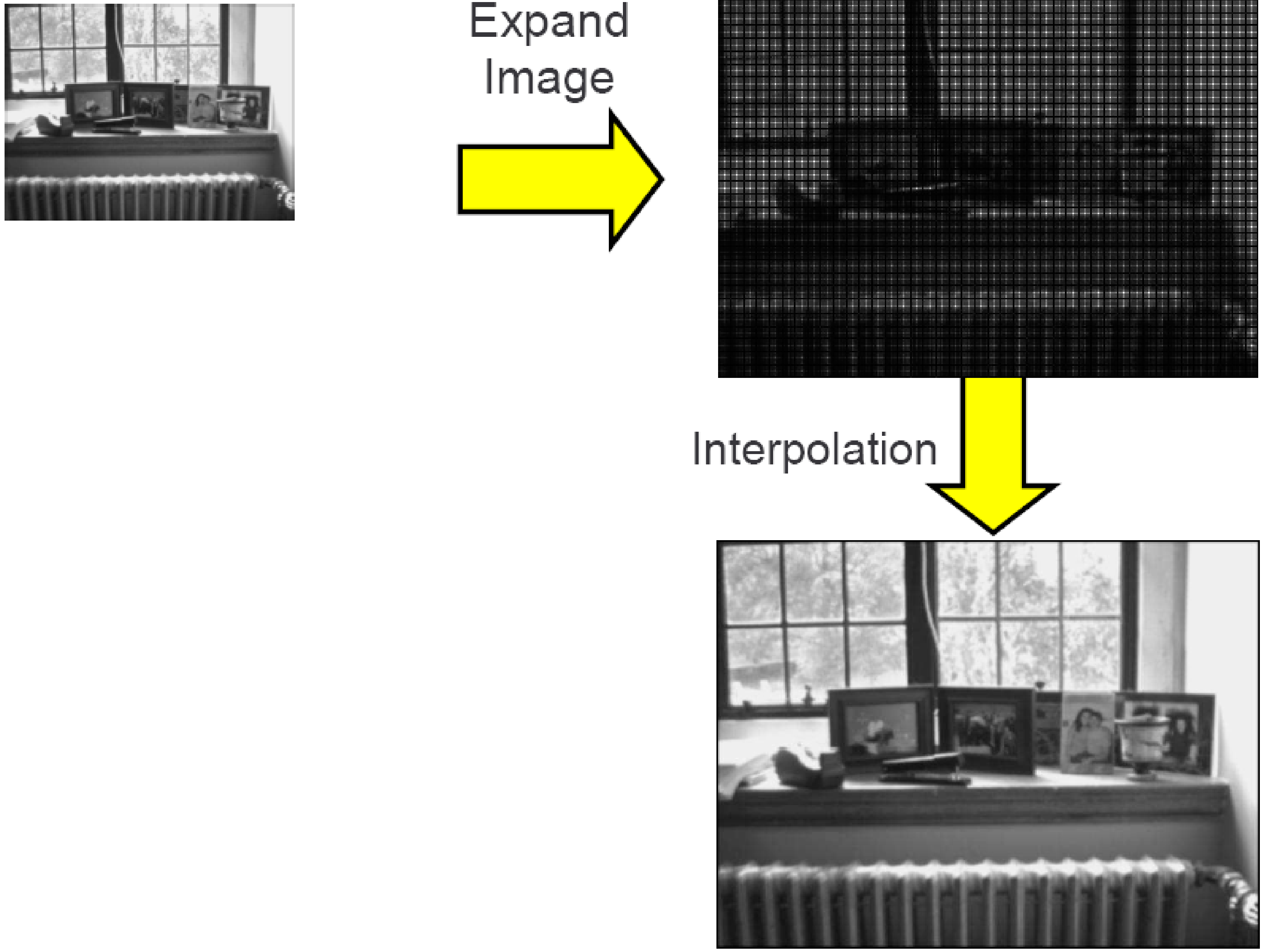


Pyramid Construction

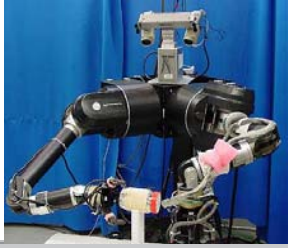


Expansion

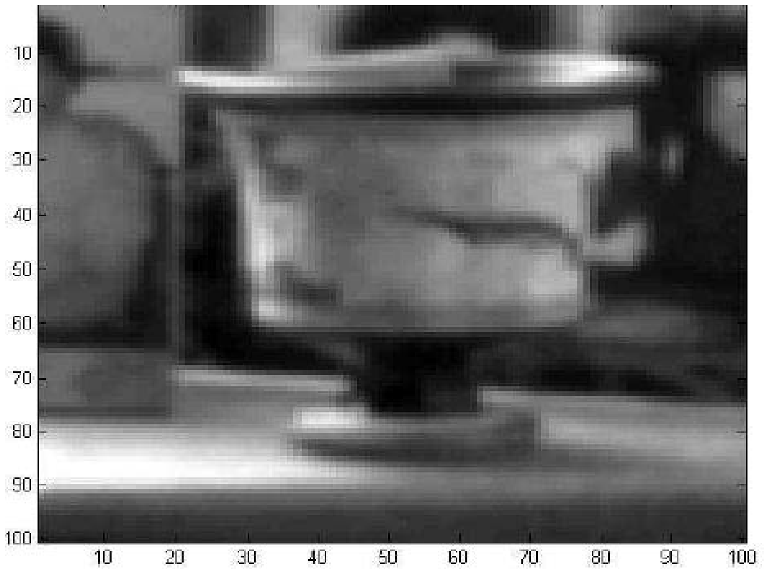




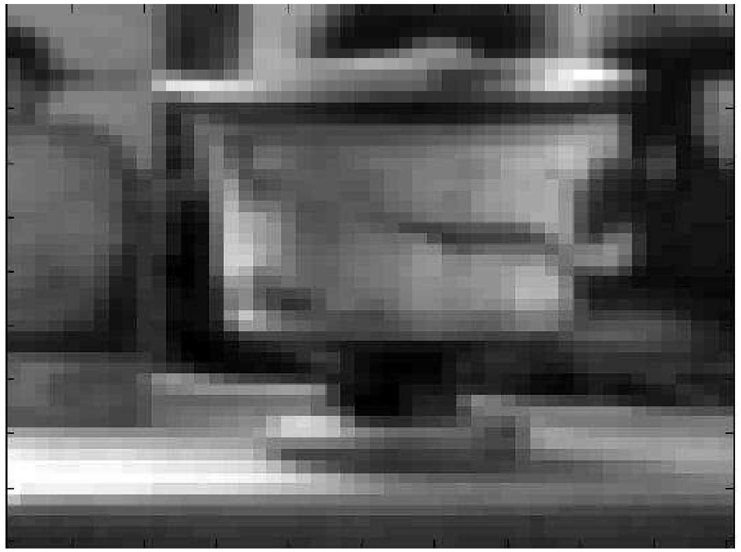
Interpolation Results



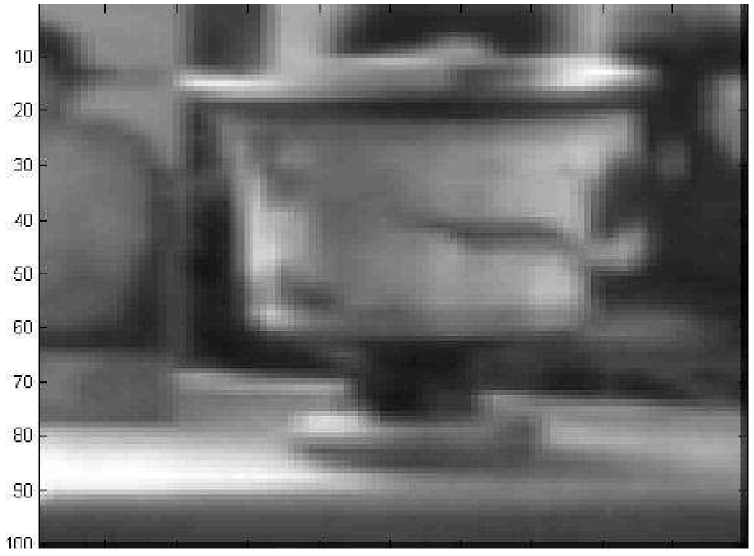
Original Image



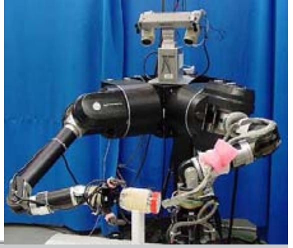
Nearest Neighbor



Bilinear Interpolation



Smoothing as Low-pass Filtering



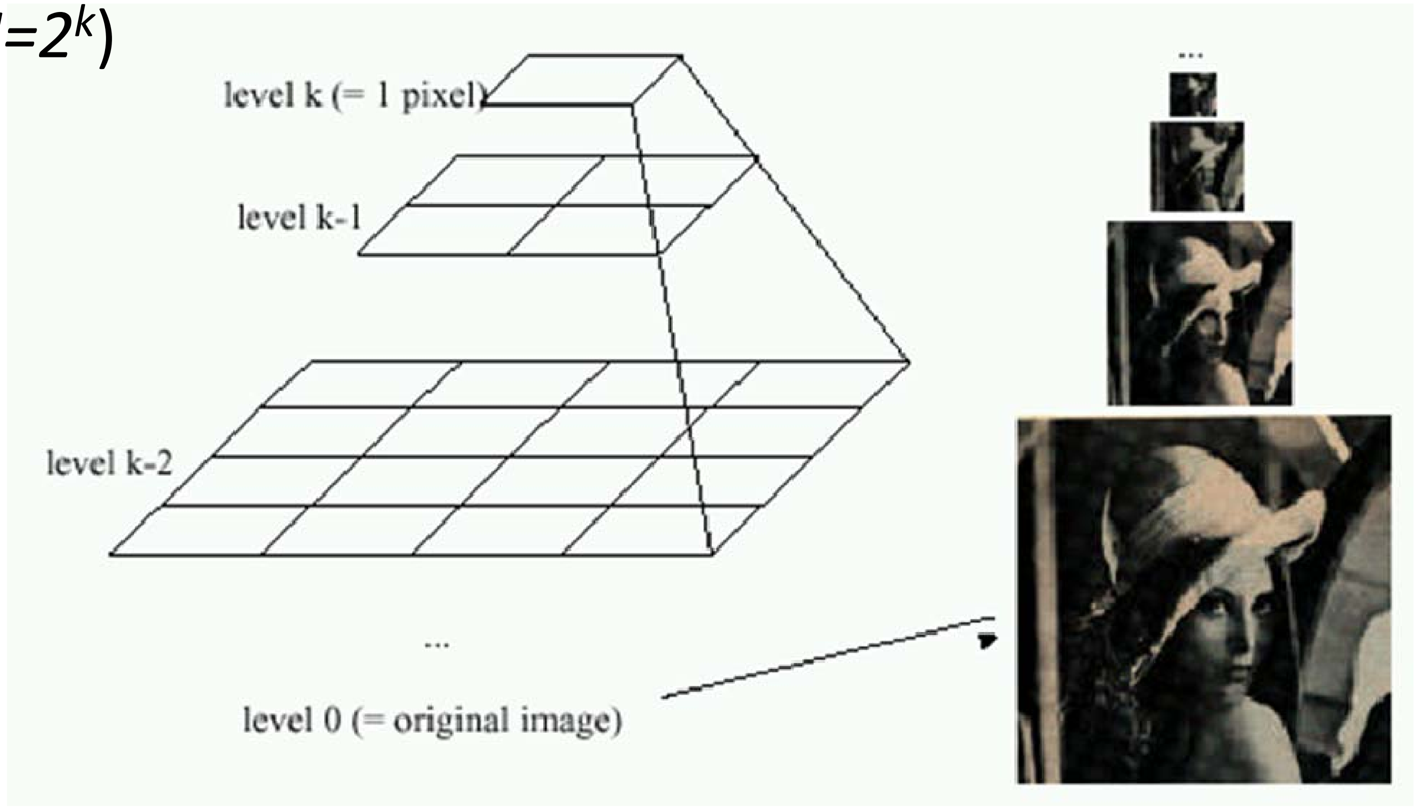
- High frequencies lead to trouble with sampling.
- Suppress high frequencies before sampling !
  - truncate high frequencies in FT
  - or convolve with a low-pass filter

- Common solution: use a Gaussian
  - multiplying FT by Gaussian is equivalent to convolving image with Gaussian.

Gaussian Pyramid

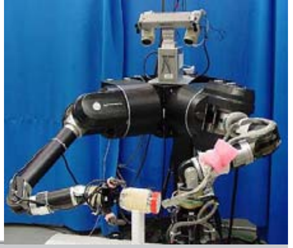
Image Pyramids

- **Idea:** Represent *NxN* image as a „pyramid“ of *1x1,2x2, 4x4,…*$2^k x2^k$ images (assuming $N=2^k$)

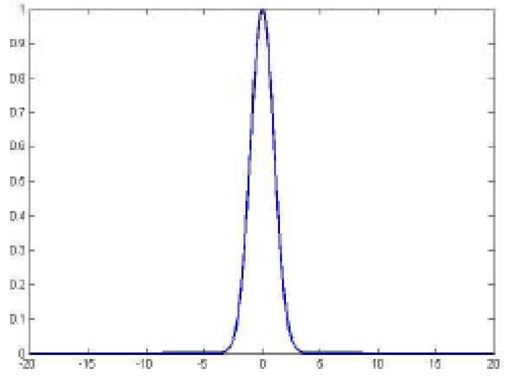


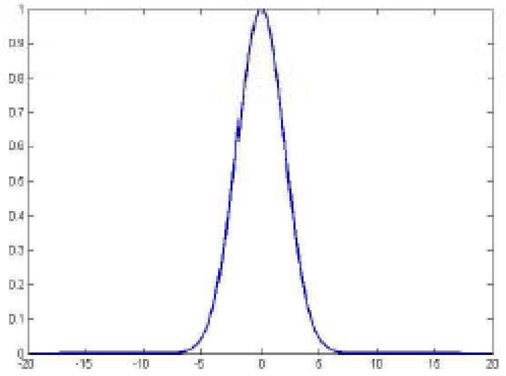
- Known as **Gaussian Pyramid** [Burt and Adelson, 1983]
  - In computer graphics: mip map [Williams, 1983]

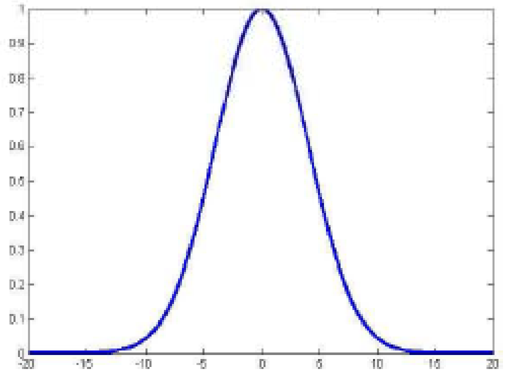
The Gaussian Pyramid



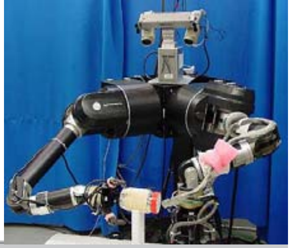
- Smooth with Gaussians because
  - a Gaussian*Gaussian=another Gaussian
- Synthesis
  - smooth and downsample
- Gaussians are low pass filters, so repetition is redundant
- Kernel width doubles with each level

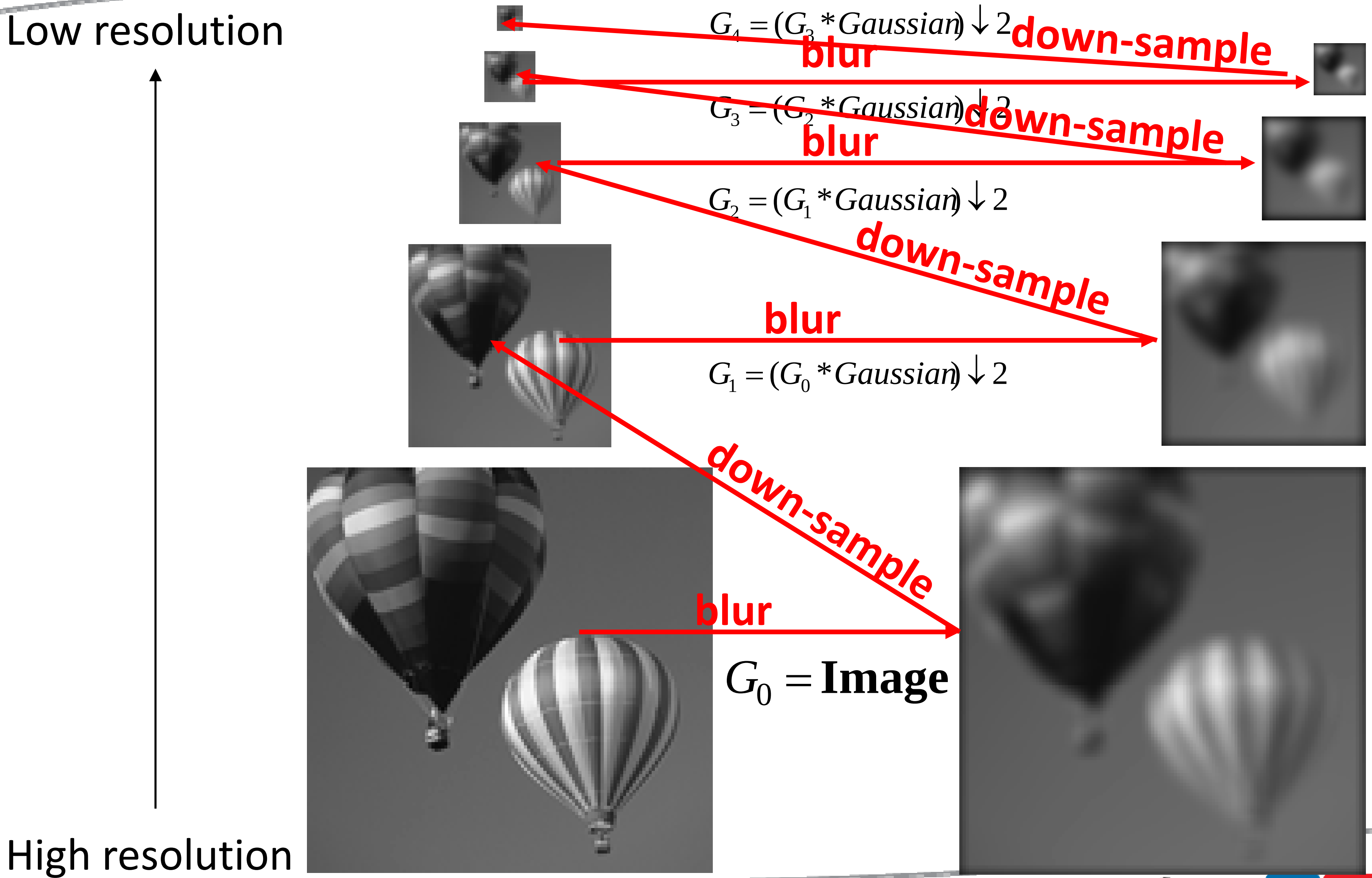

Level 1


Level 2


Level 3

The Gaussian Pyramid





$$G_4 = (G_3 * Gaussian) \downarrow 2$$

$$G_3 = (G_2 * Gaussian) \downarrow 2$$

$$G_2 = (G_1 * Gaussian) \downarrow 2$$

$$G_1 = (G_0 * Gaussian) \downarrow 2$$

$$G_0 = \textbf{Image}$$

The Gaussian Pyramid

- Smooth with Gaussians, because
  - a Gaussian*Gaussian = another Gaussian
- Gaussians are low pass filters, so representation is redundant.

GAUSSIAN PYRAMID

https://www.cs.tau.ac.il/~hezy/Vision%20Seminar/pyramid83.pdf

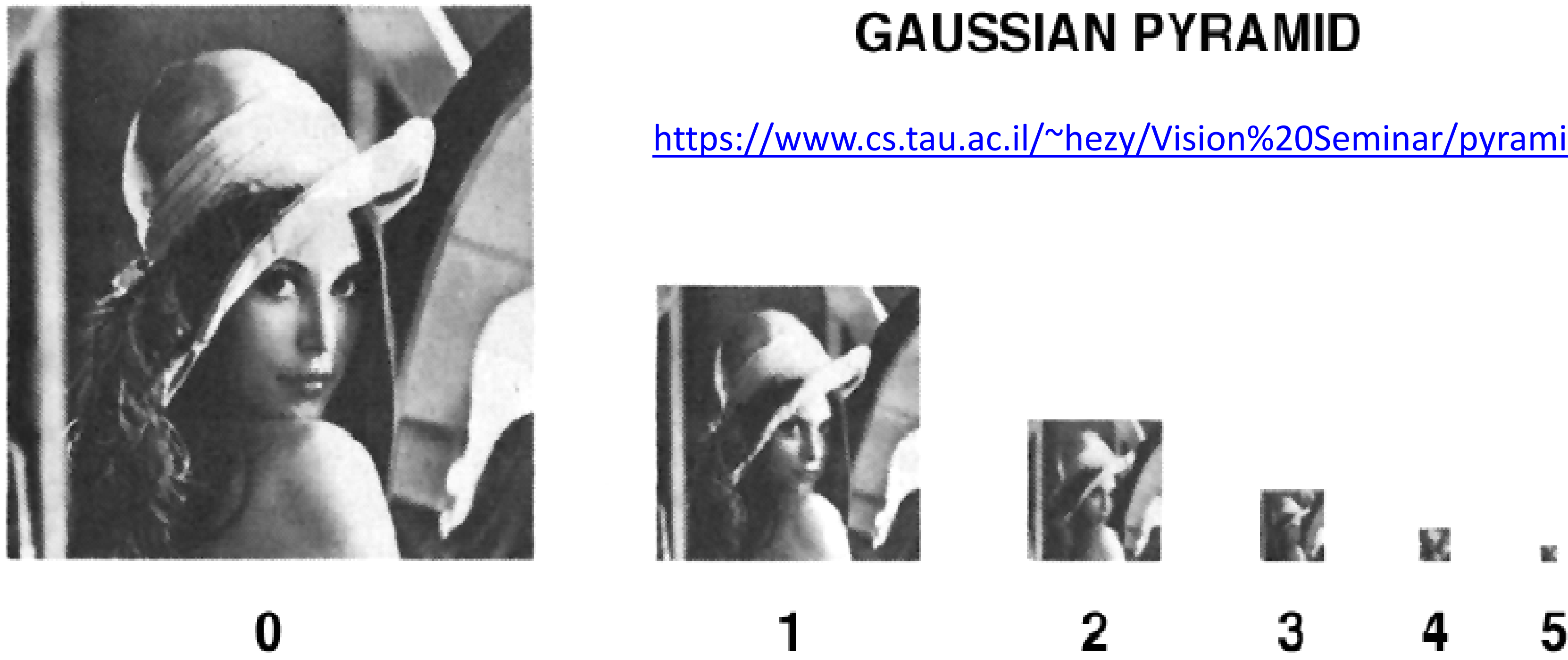
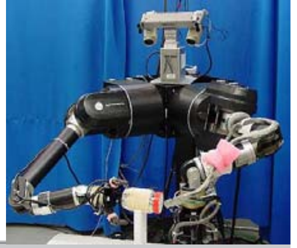
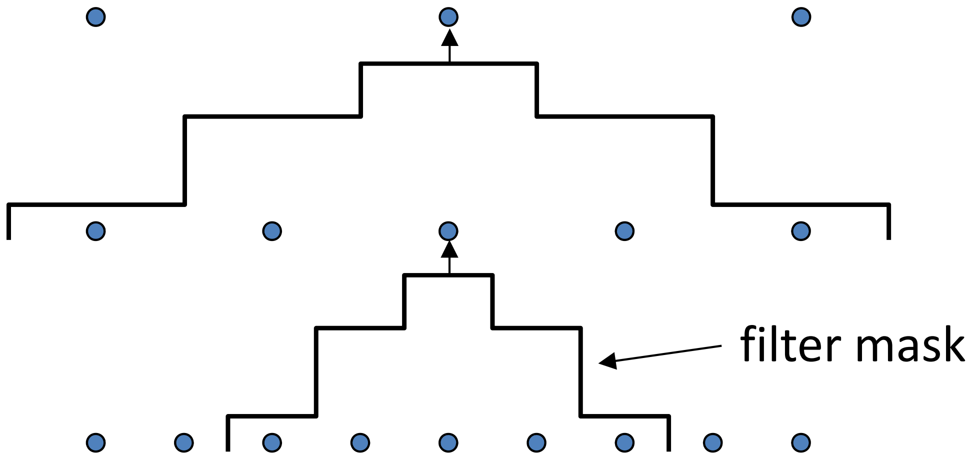


Fig. 4. First six levels of the Gaussian pyramid for the "Lady" image The original image, level 0, meusures 257 by 257 pixels and each higher level array is roughly half the dimensdons of its predecessor. Thus, level 5 measures just 9 by 9 pixels.

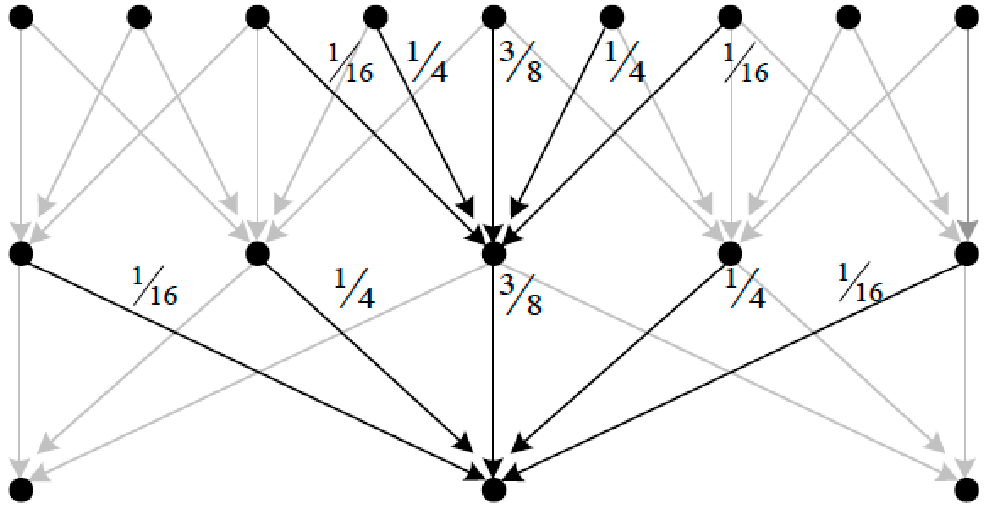
Gaussian pyramid construction

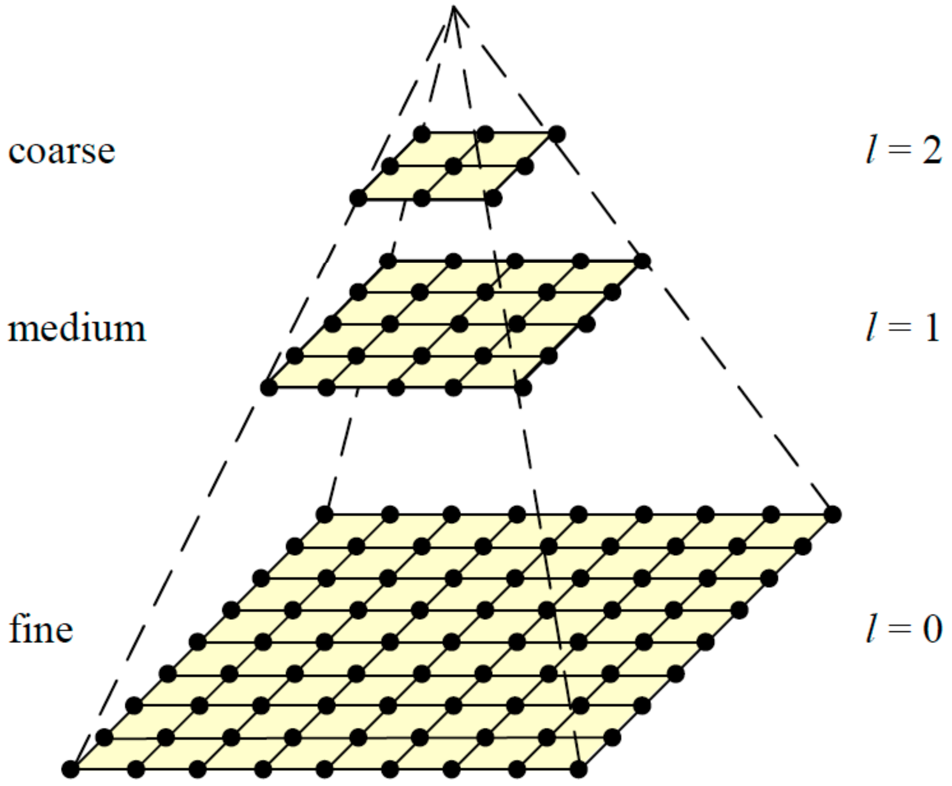




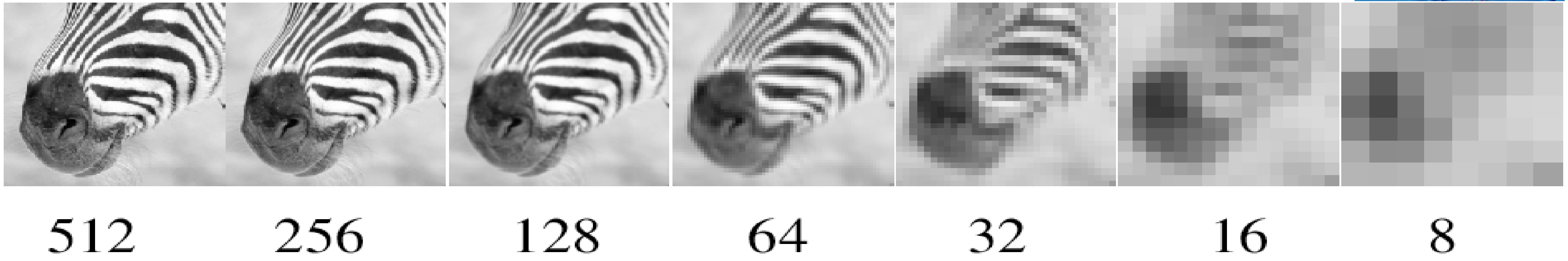
- Repeat
  - Filter
  - Subsample
- Until minimum resolution reached
  - can specify desired number of levels (e.g., 3-level pyramid)
- The whole pyramid is only 4/3 the size of the original image!

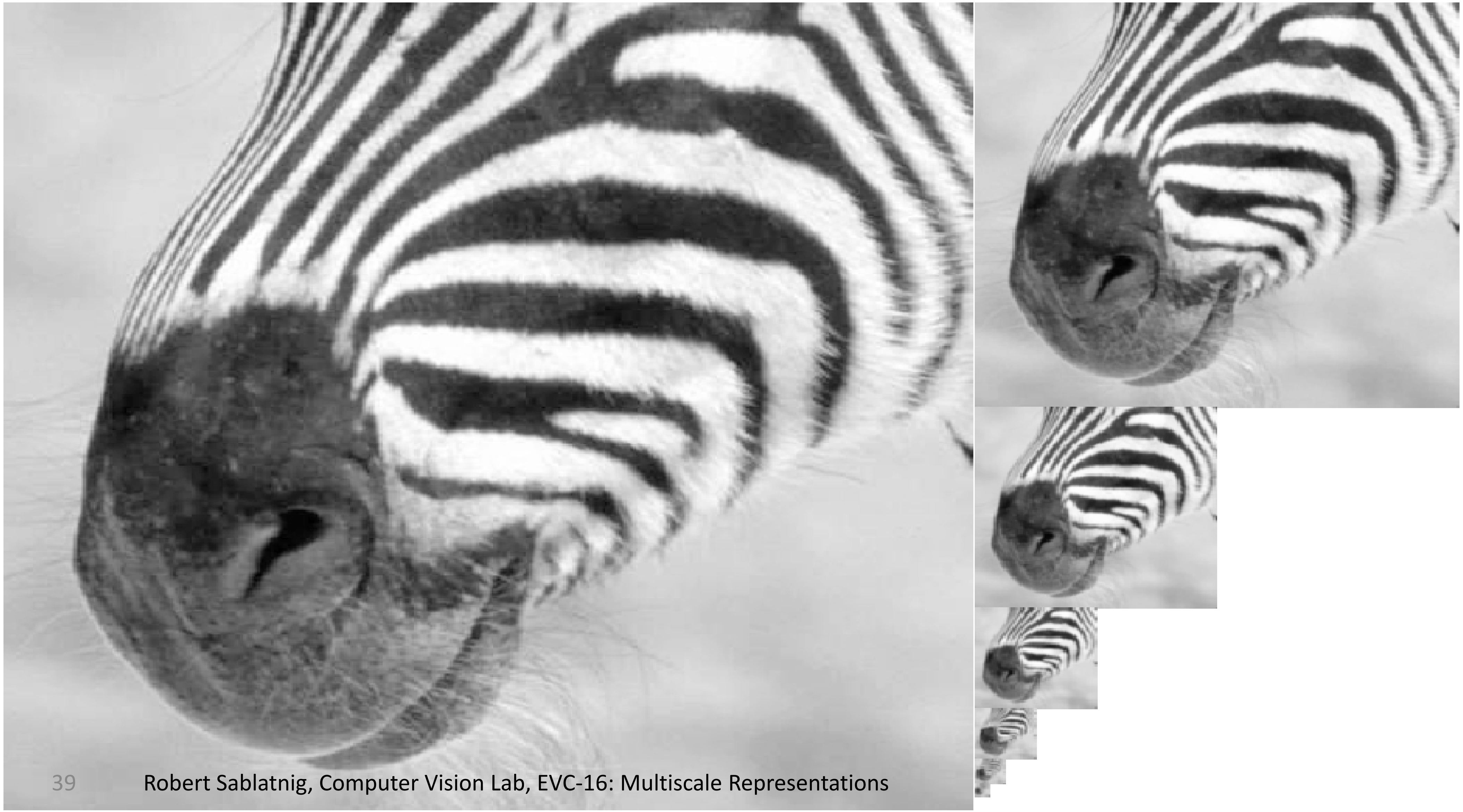
Gaussian Pyramid



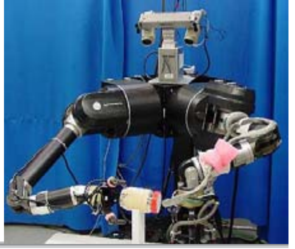


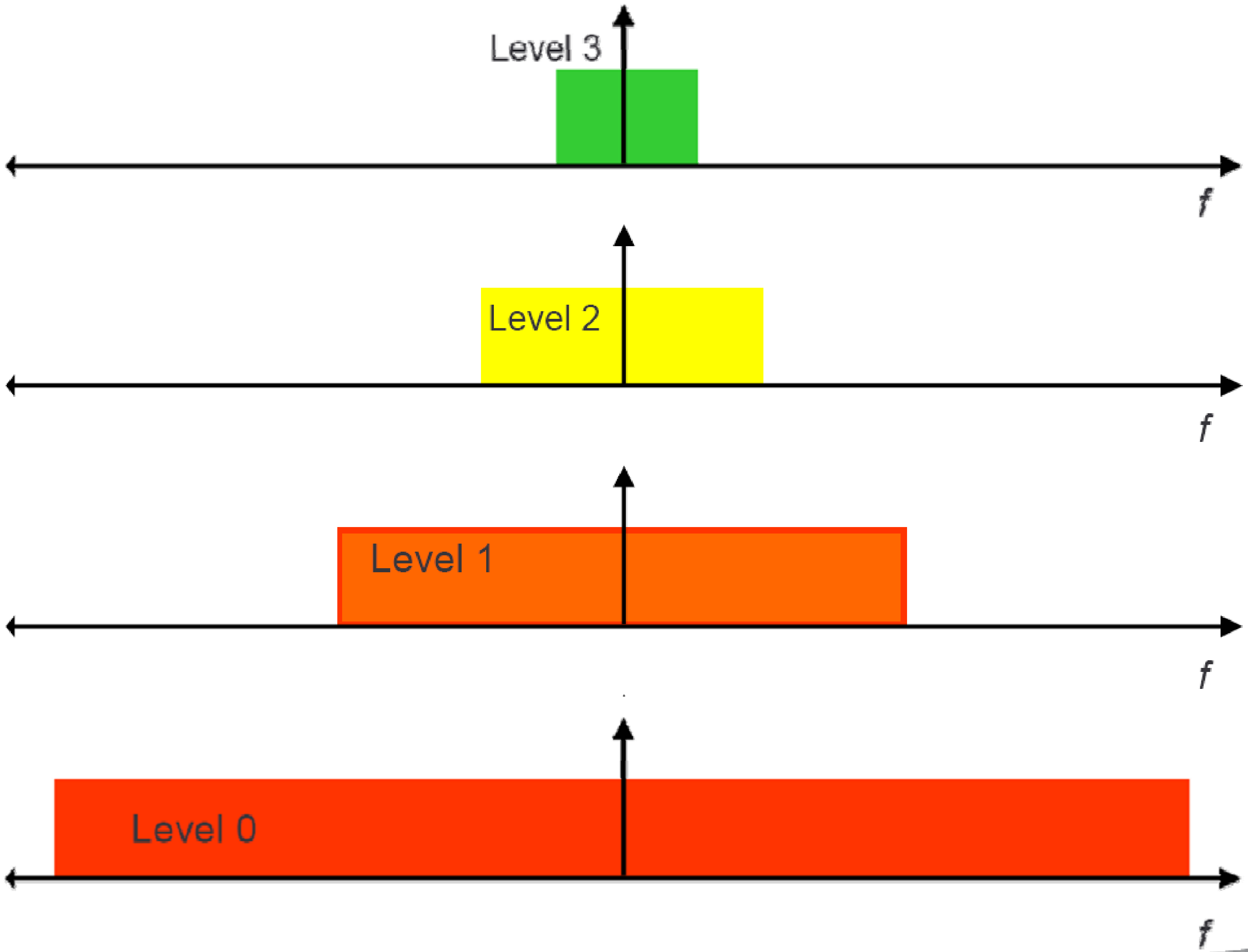
https://www.cs.tau.ac.il/~hezy/Vision%20Seminar/pyramid83.pdf IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

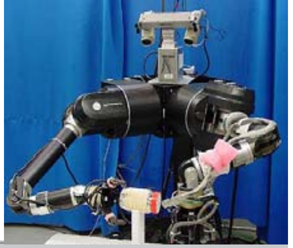

512
256
128
64
32
16
8



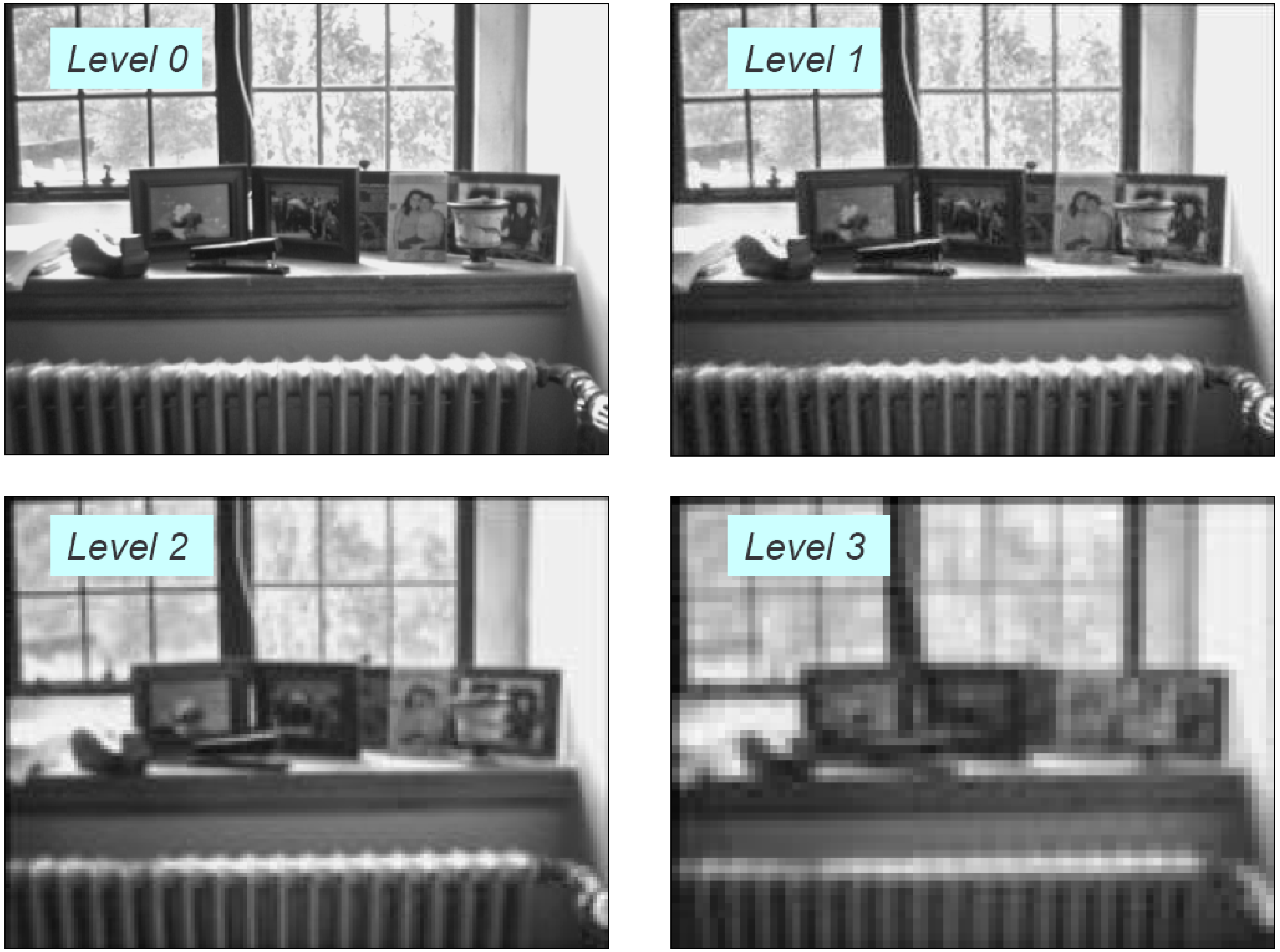
Gaussian Pyramid Frequency Composition



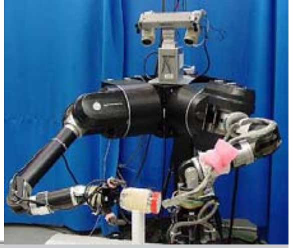




Pyramids at Same Resolution

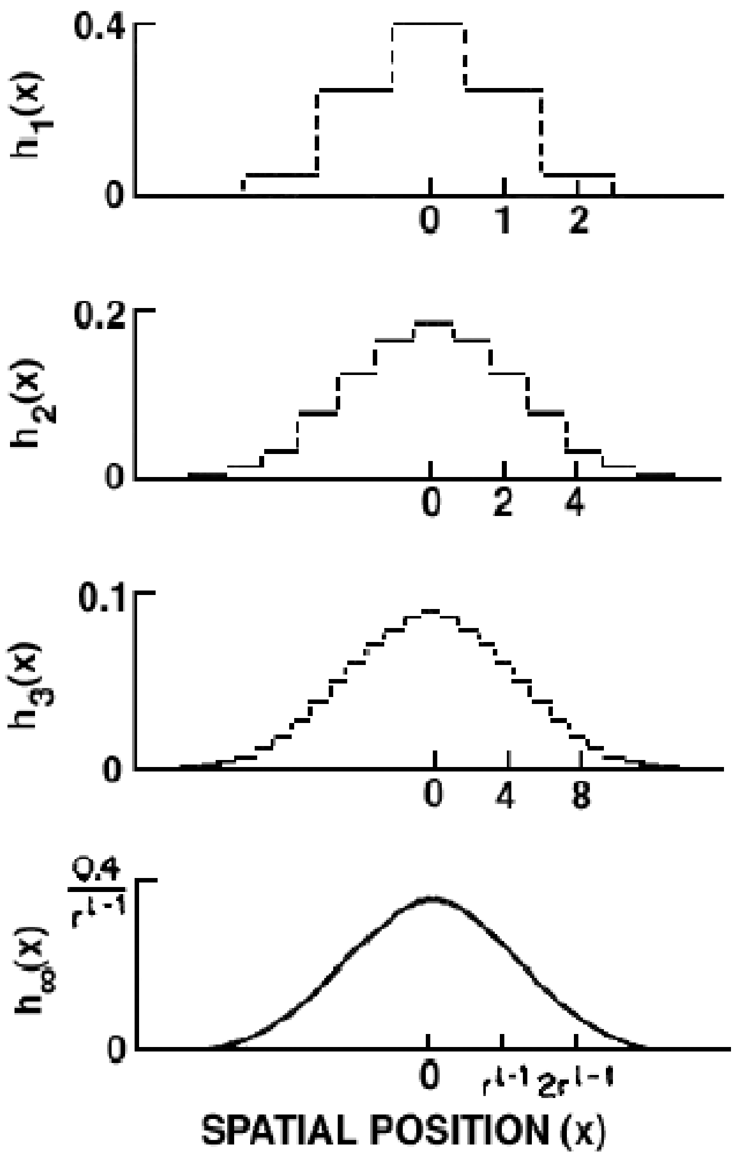


What are they good for?



- Improve Search
  - Search over translations: Classic coarse-to-fine strategy
  - Search over scale: Template matching e.g. find a face at different scales

- Pre-computation
  - Need to access image at different blur levels
  - Useful for texture mapping at different resolutions (mip-mapping)
- Image Processing
  - Editing frequency bands separately
  - E.g. image blending…



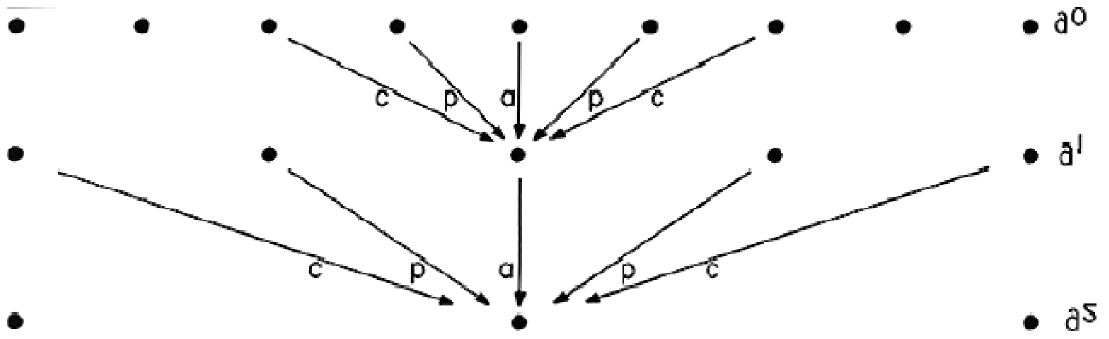
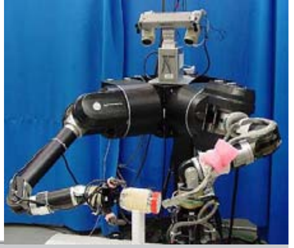


Fig. 2. The equivalent weighting functions $h_l(x)$ for nodes in levels 1, 2, 3, and infinity of the Gaussian pyramid. Note that axis scales have been adjusted by factors of 2 to aid comparison Here the parameter a of the generating kernel is 0.4, and the resulting equivalent weighting functions closely resemble the Gaussian probability density functions.

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

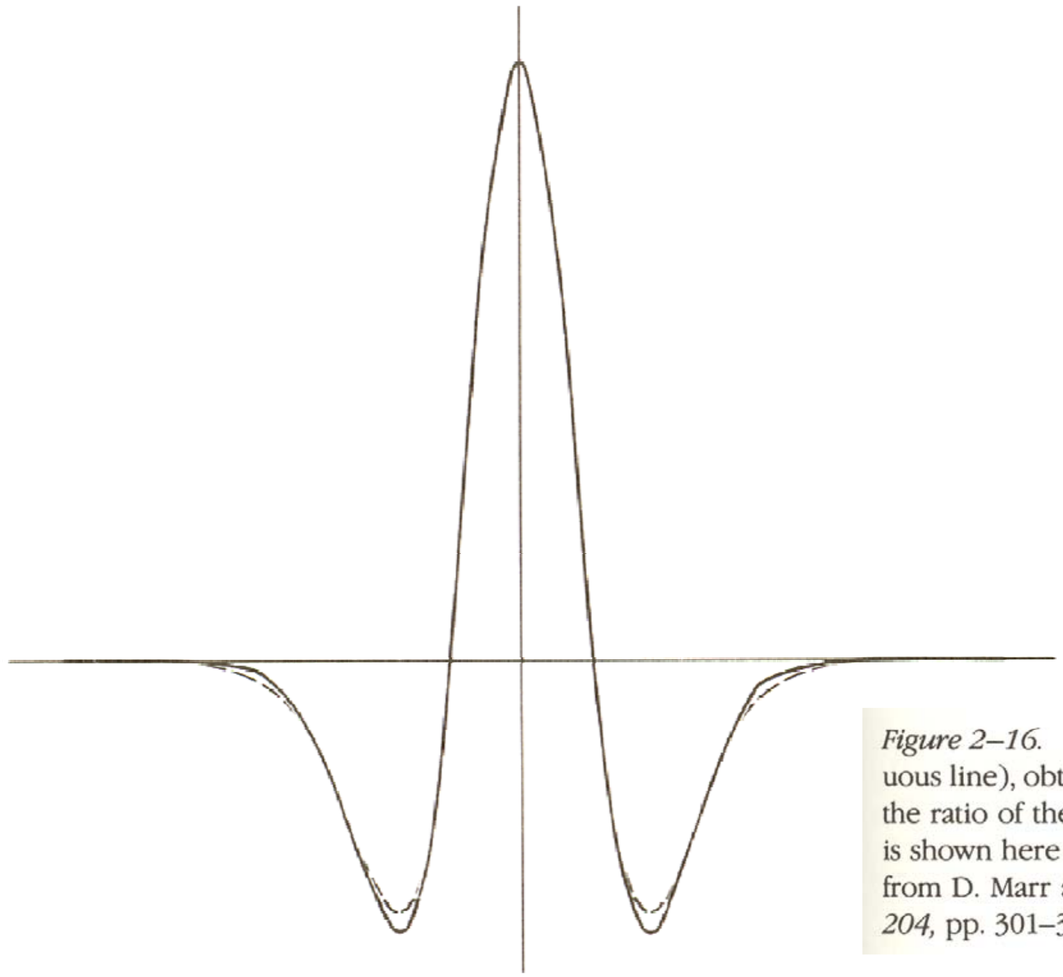
Gaussian Pyramids are used for



- Up- or down- sampling images.
- Multi-resolution image analysis
  - Look for an object over various spatial scales
  - Coarse-to-fine image processing: form blur estimate or the motion analysis on very low-resolution image, up-sample and repeat.
  - Often a successful strategy for avoiding local minima in complicated estimation tasks.

Difference of Gaussians (DoG)

- Laplacian of Gaussian can be approximated by the difference between two different Gaussians



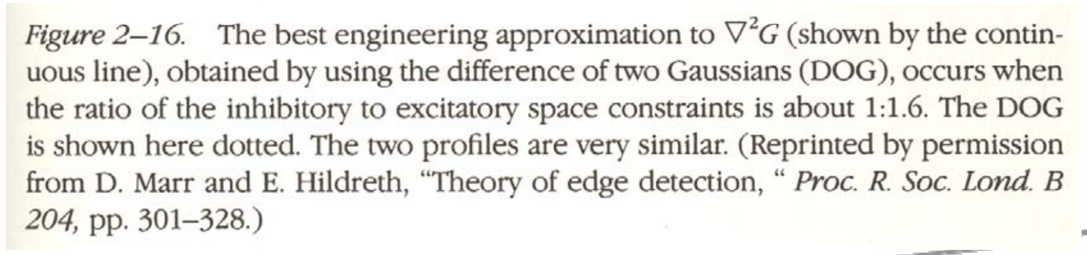
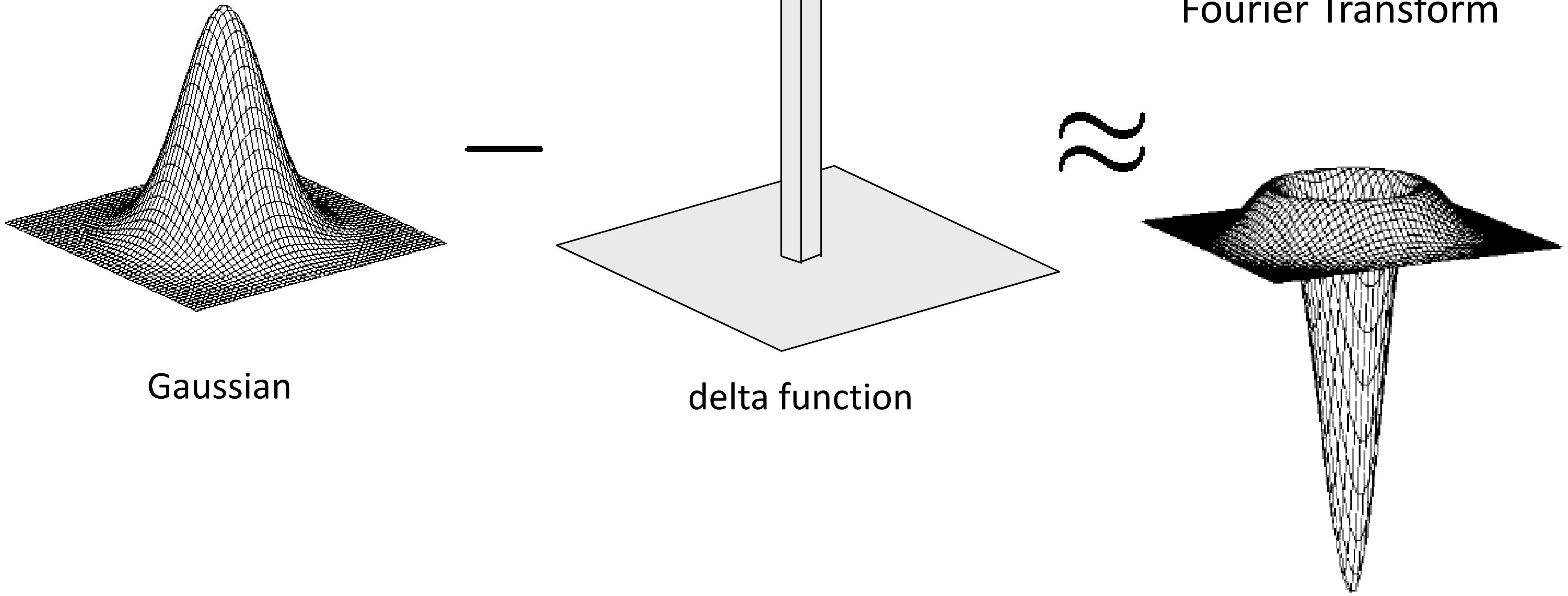
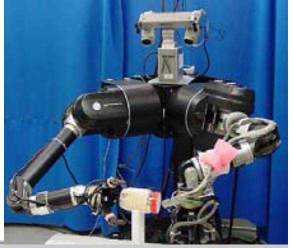


Figure 2–16. The best engineering approximation to $\nabla^2 G$ (shown by the continuous line), obtained by using the difference of two Gaussians (DOG), occurs when the ratio of the inhibitory to excitatory space constraints is about 1:1.6. The DOG is shown here dotted. The two profiles are very similar. (Reprinted by permission from D. Marr and E. Hildreth, "Theory of edge detection, " *Proc. R. Soc. Lond. B 204*, pp. 301–328.)

Gaussian – Image filter



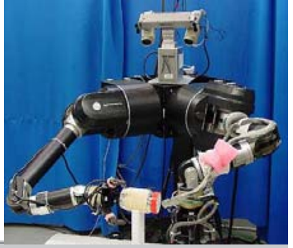
The Laplacian Pyramid



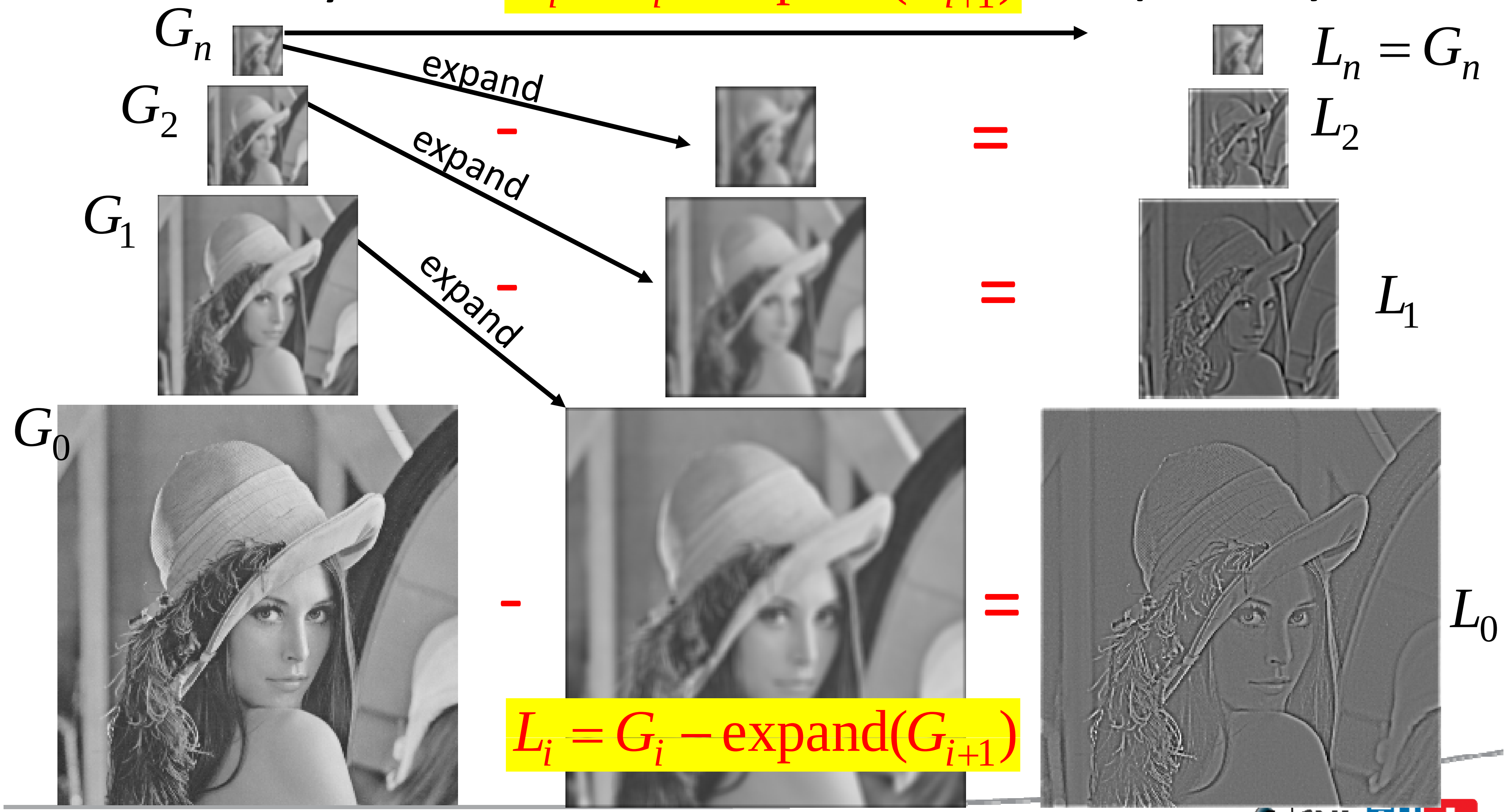
The Laplacian Pyramid

- Synthesis
  - Compute the difference between upsampled Gaussian pyramid level and Gaussian pyramid level.
  - band pass filter - each level represents spatial frequencies (largely) unrepresented at other level.

The Laplacian Pyramid

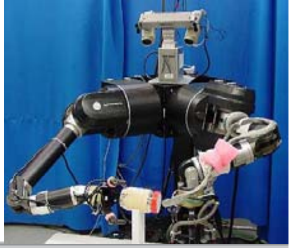


$$G_i = L_i + \mathrm{expand}(G_{i+1})$$

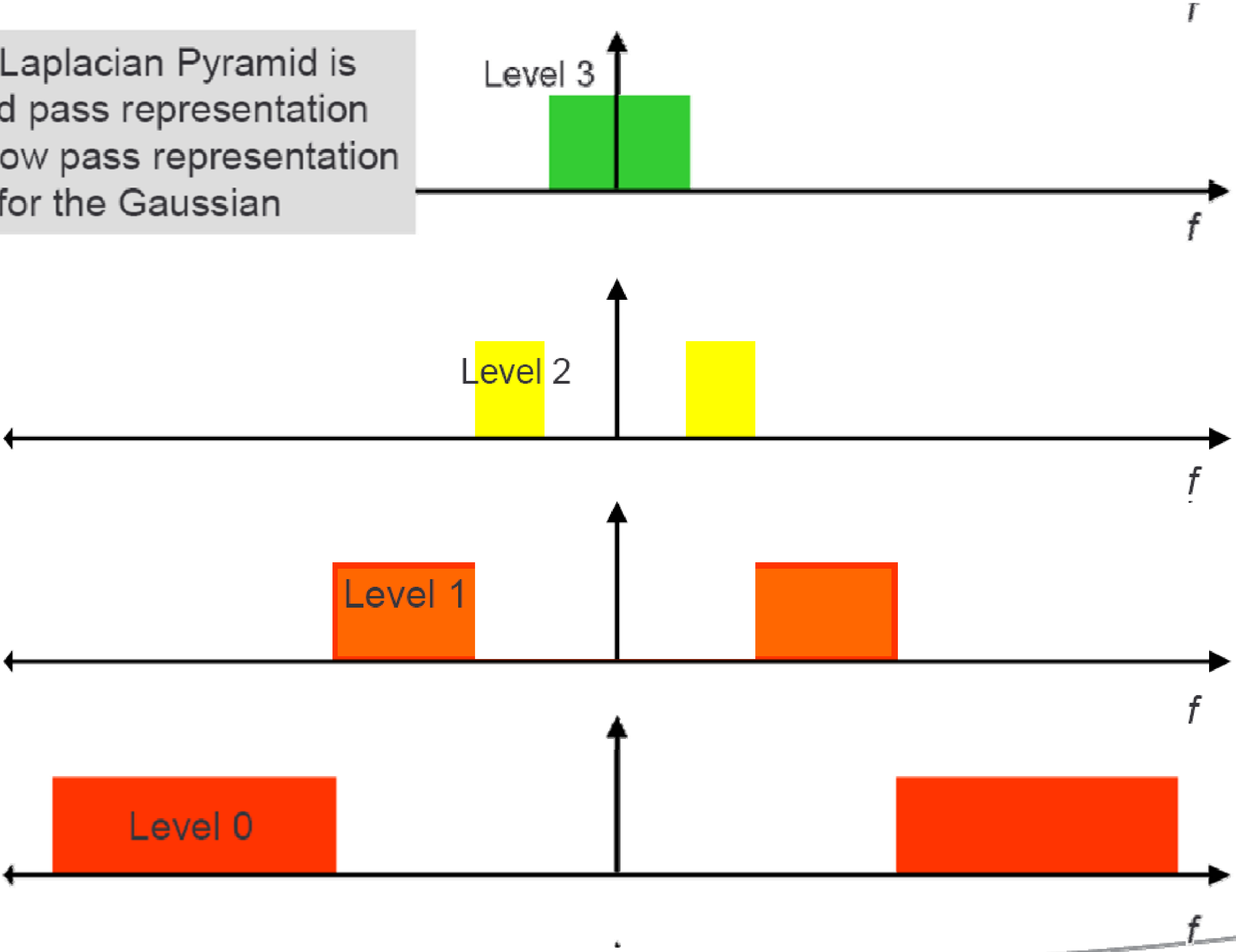


$$L_i = G_i - \mathrm{expand}(G_{i+1})$$

Laplacian Pyramid Frequency Composition

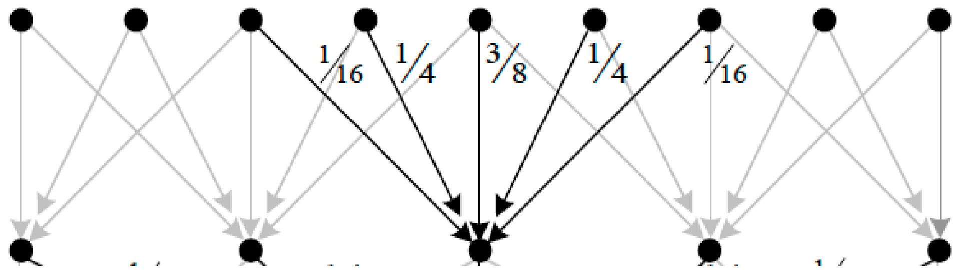


The Laplacian Pyramid is a band pass representation vice a low pass representation for the Gaussian

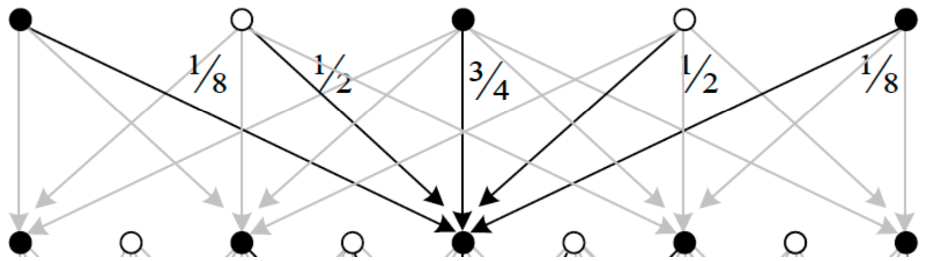


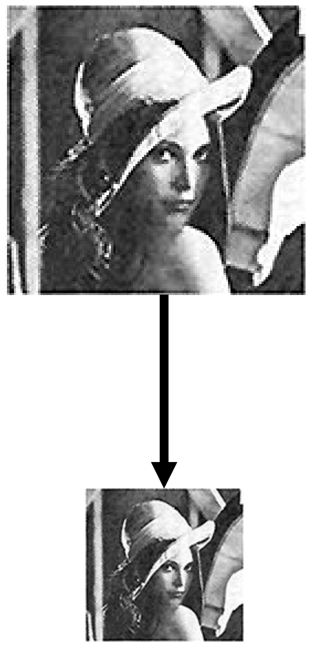
Reduce and Expand

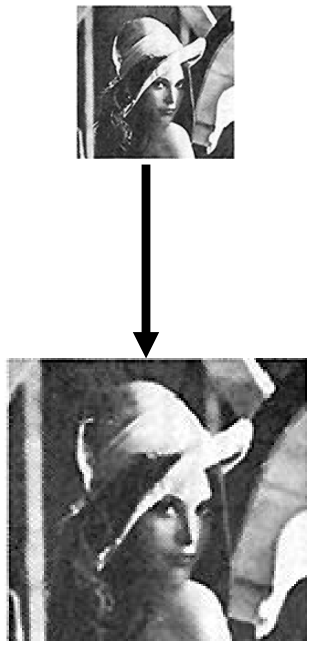
Reduce



Expand



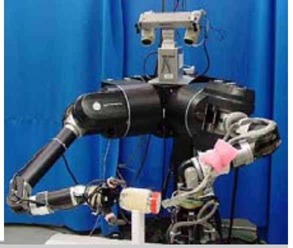




https://www.cs.tau.ac.il/~hezy/Vision%20Seminar/pyramid83.pdf

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

Showing, at full resolution, the information captured at each level of a Gaussian (top) and Laplacian (bottom) pyramid.



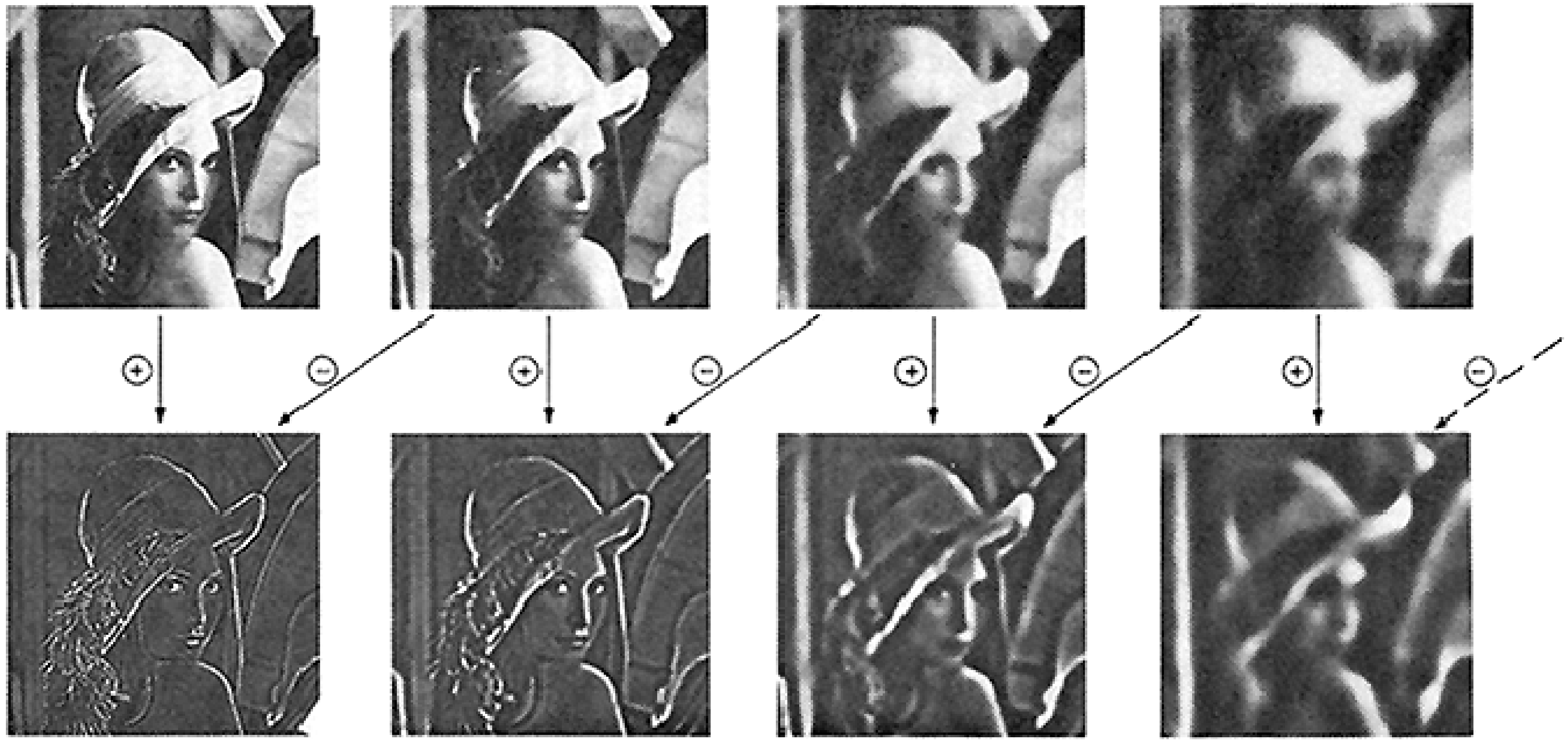
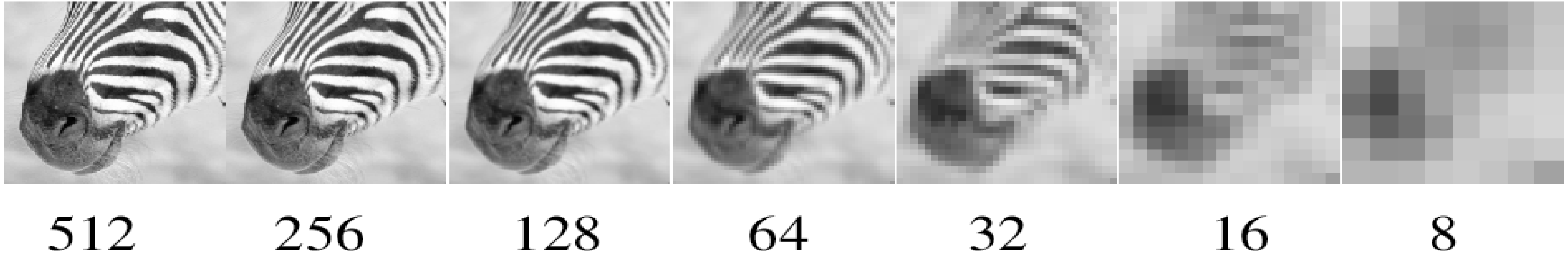
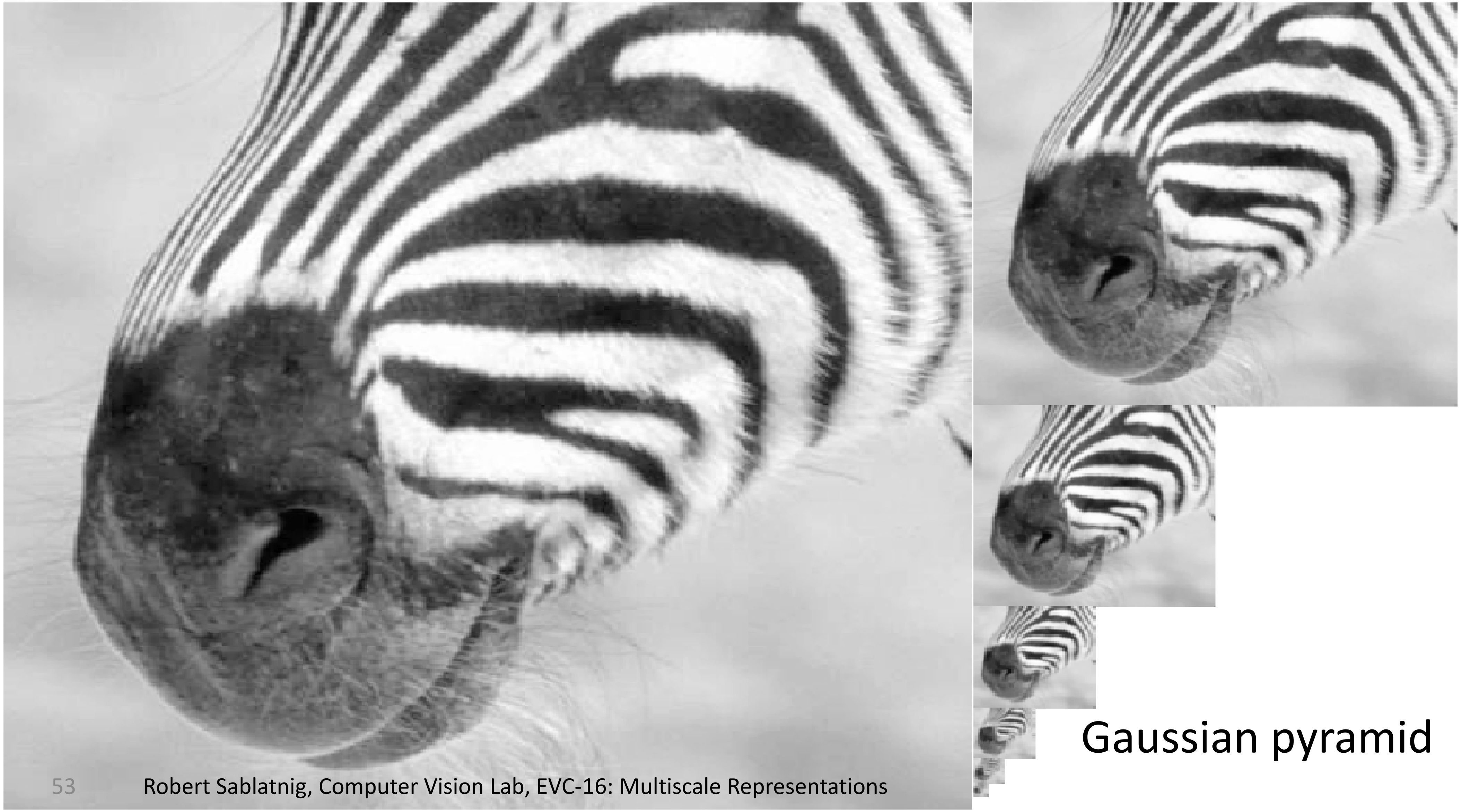
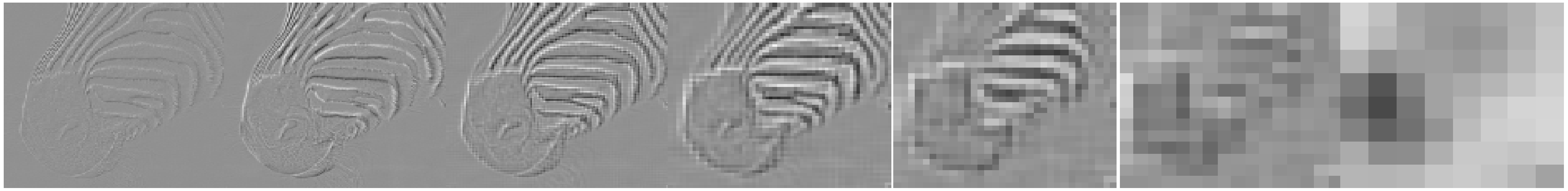


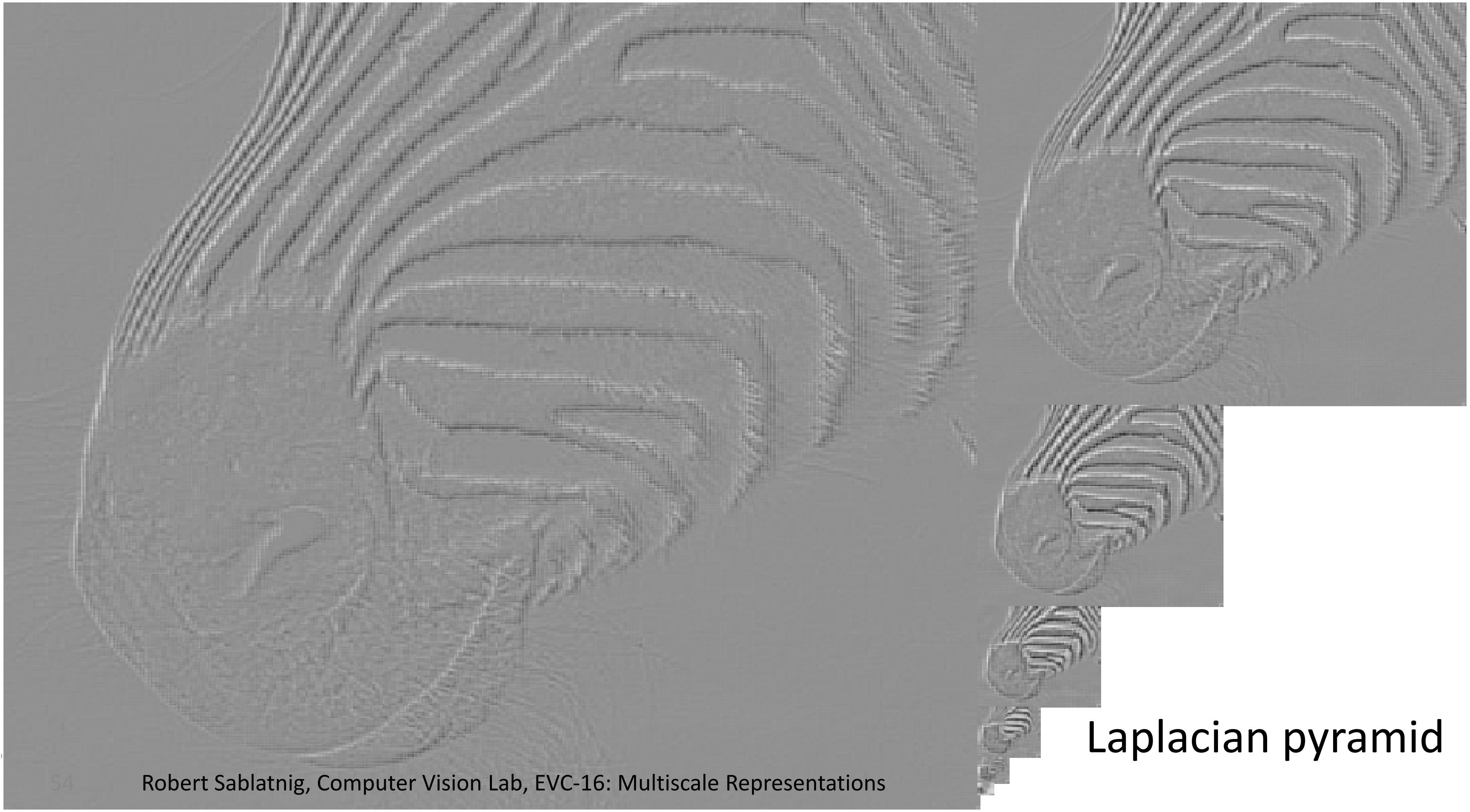
Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtainedby expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.





Gaussian pyramid

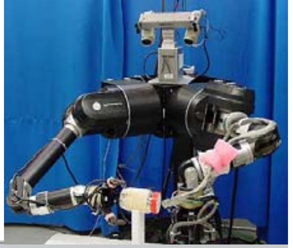




Laplacian pyramid

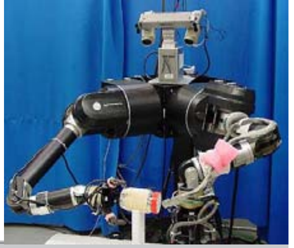
Applications of Image Pyramids

Applications of Image Pyramids



- Coarse-to-Fine strategies for computational efficiency.
- Search for correspondence
  - look at coarse scales, then refine with finer scales
- Edge tracking
  - a “good” edge at a fine scale has parents at a coarser scale
- Control of detail and computational cost in matching
  - e.g. finding stripes
  - very important in texture representation
- Image Blending and Mosaicing
- Data compression (Laplacian pyramid)

Edge Detection using Pyramids



- Coarse-to-fine strategy:
- Do edge detection at higher level.
- Consider edges of finer scales only near the edges of higher scales.

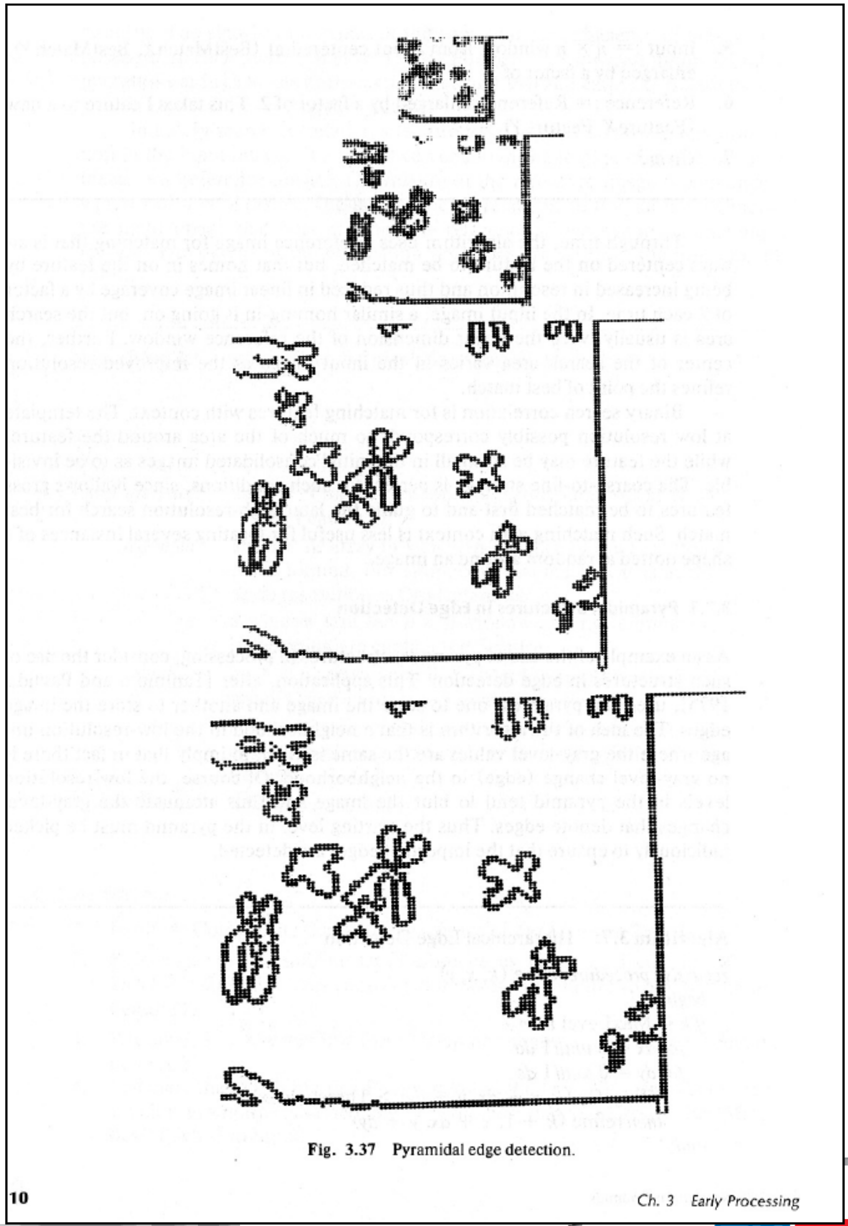
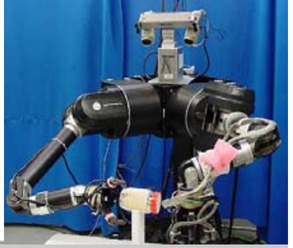
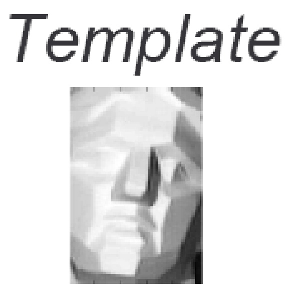


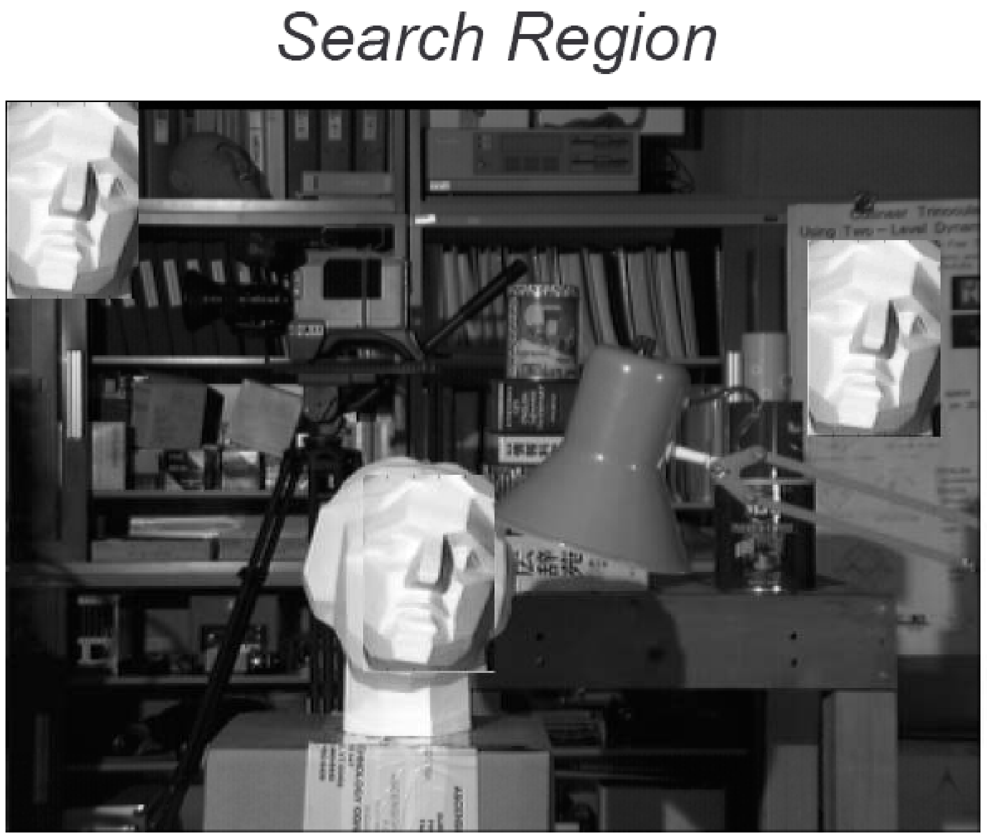
Fig. 3.37 Pyramidal edge detection.

10 Ch. 3 Early Processing

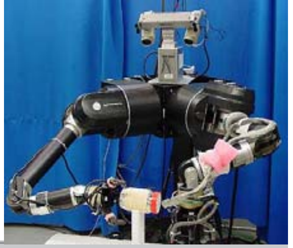
Fast Template Matching







- For an m x n image….
- For a p x q template….
- The complexity of the 2D pattern recognition task is O(mnpq)
- This gets even worse for a family of templates (e.g. to adress scale and/or rotational effects)



Fast Template Matching

Template

Search Region

Original Image

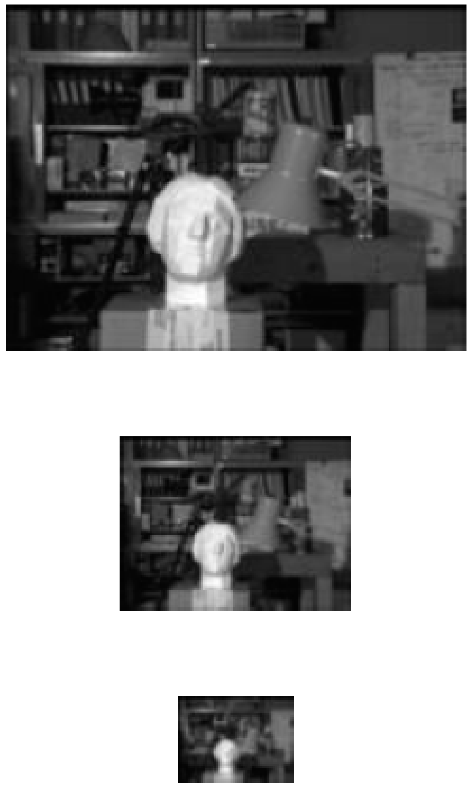
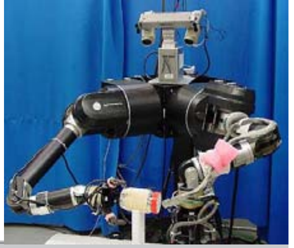
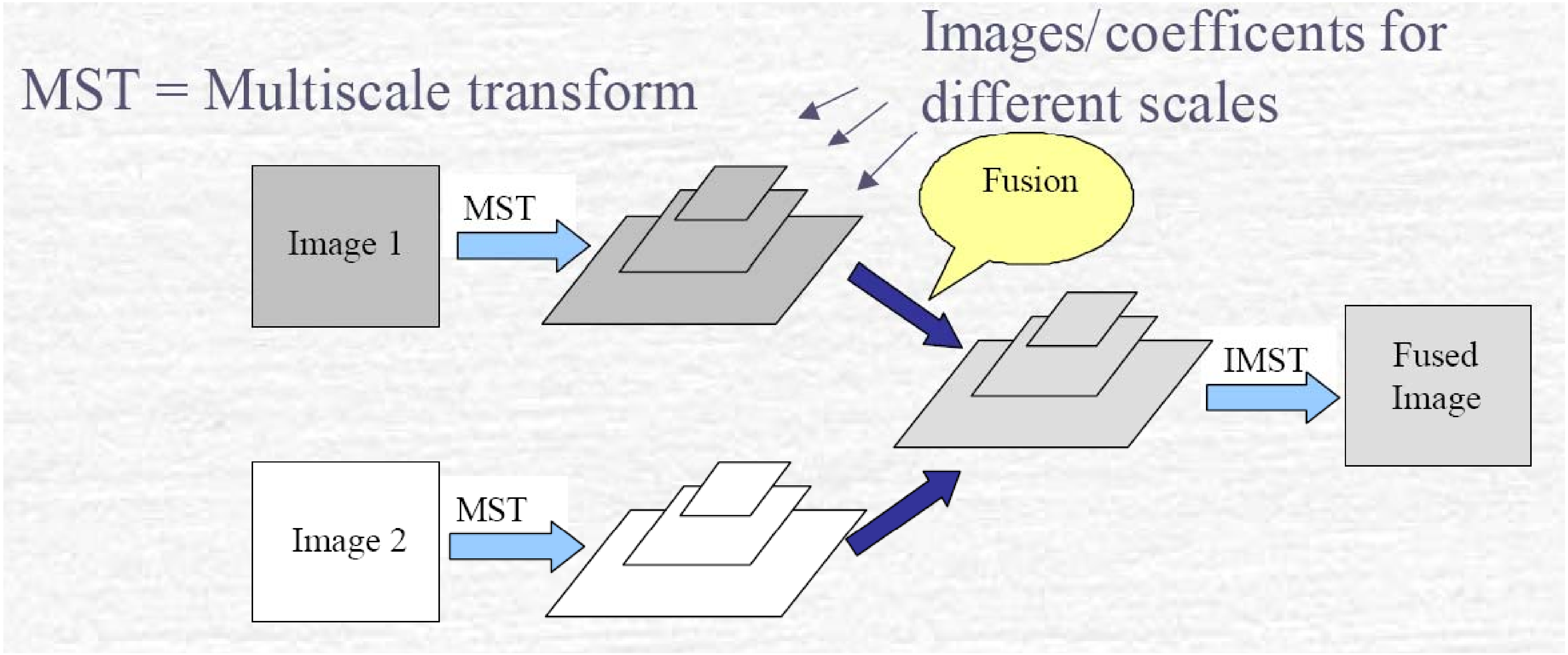


Image Fusion

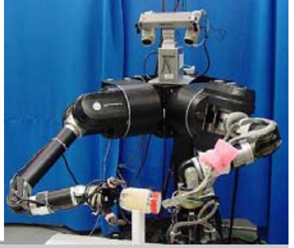


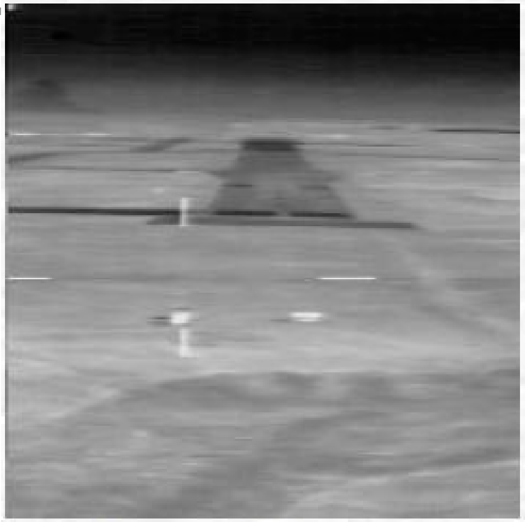

MST = Multiscale transform
Images/coefficents for different scales
Fusion
Image 1
MST
Image 2
MST
IMST
Fused Image

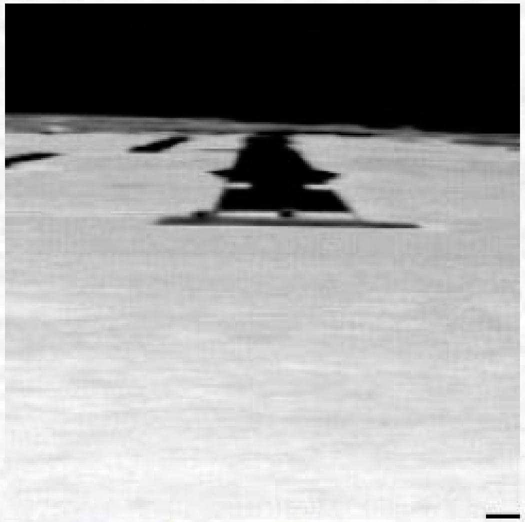
Multi-scale Transform (MST) = Obtain Pyramid from Image

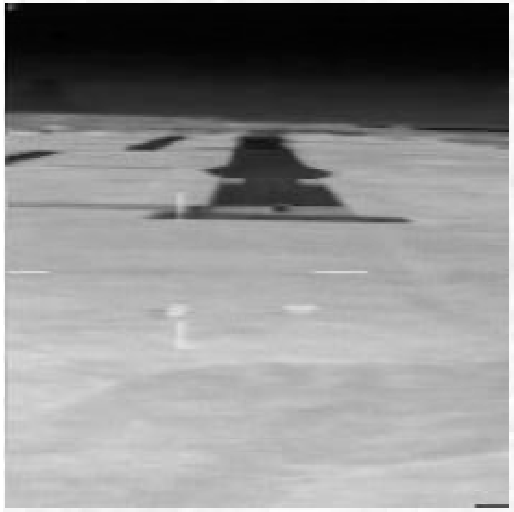
Inverse Multi-scale Transform (IMST) = Obtain Image from Pyramid

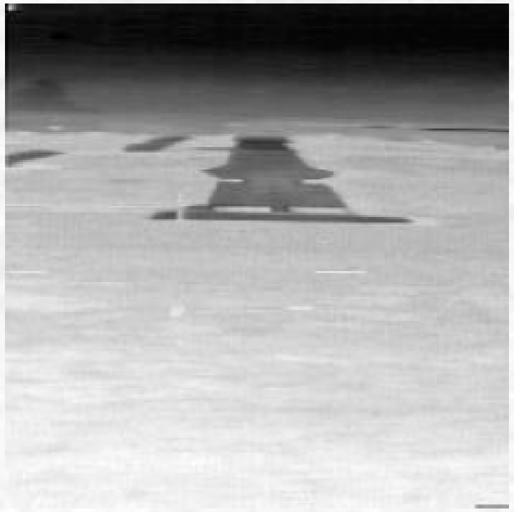
Multi-Sensor Fusion




Long Wave


Medium Wave


Averaging


Selecting Maximum

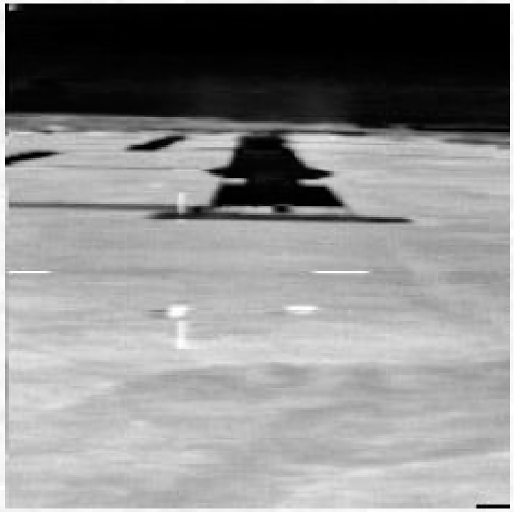

Laplacian Fusion

Image Compression

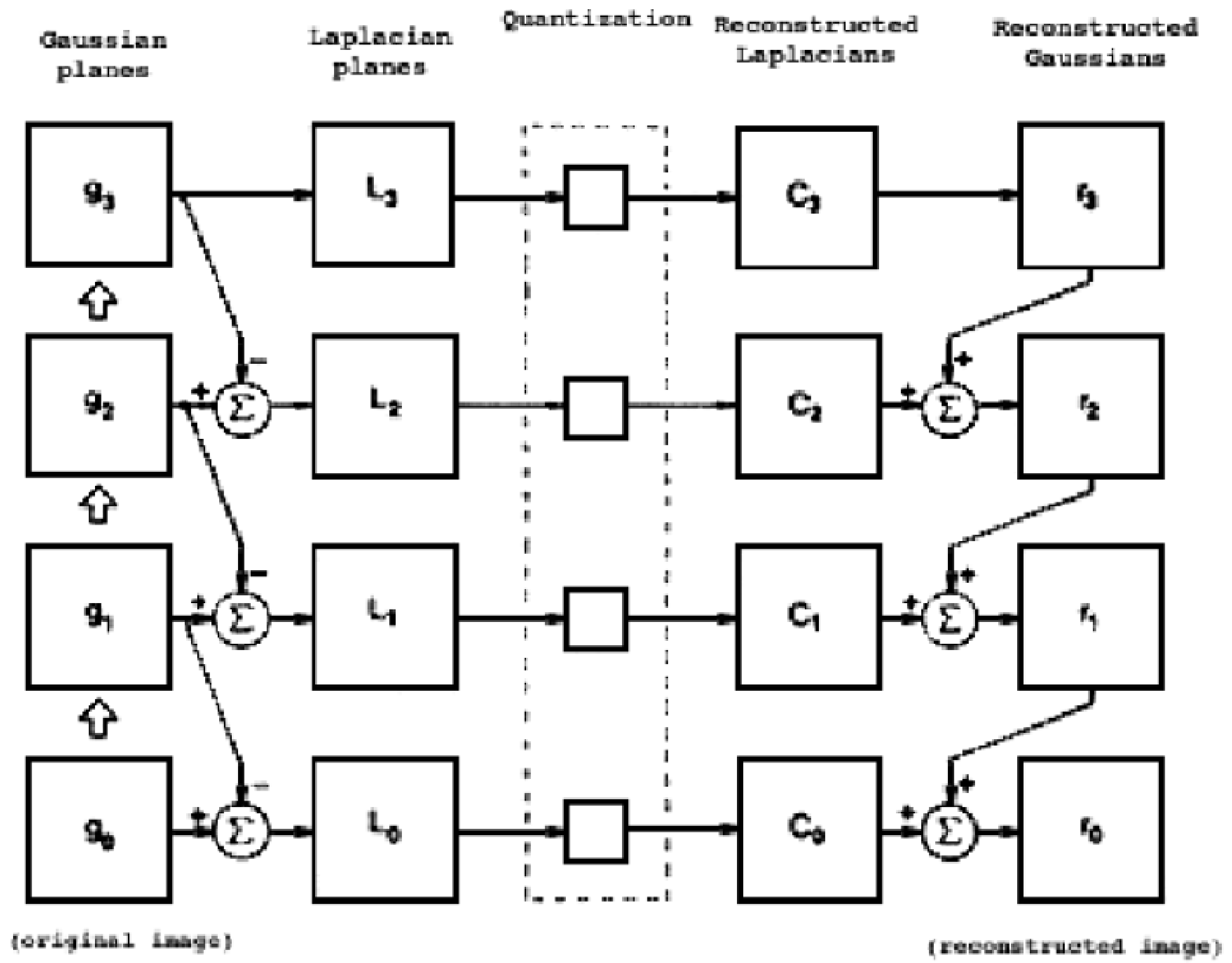
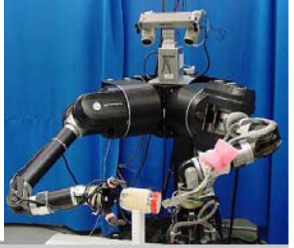
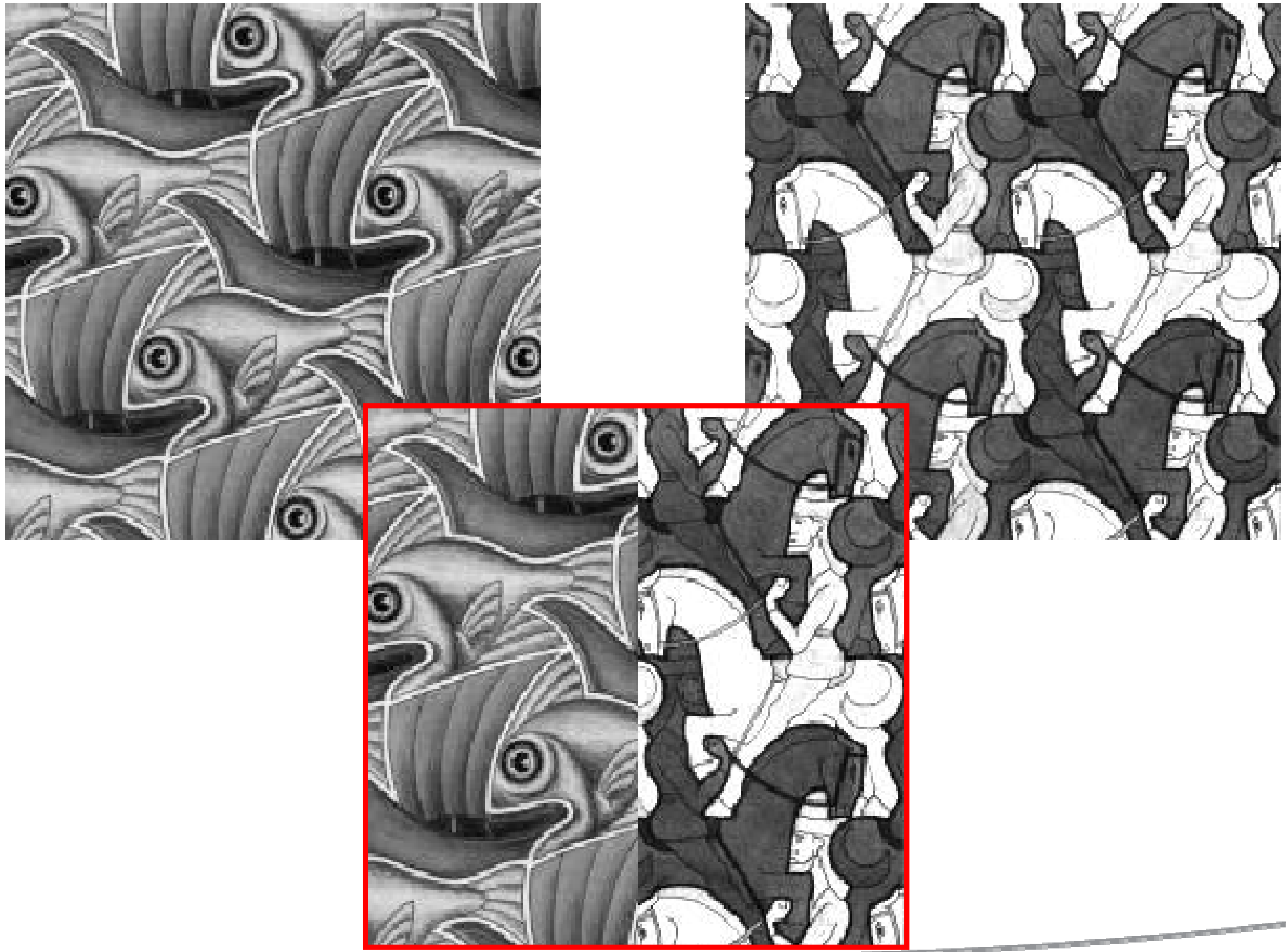
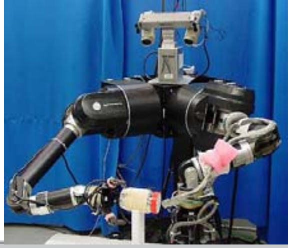


Fig. 10. A summary of the steps in Laplacian pyramid coding and decoding. First, the original image g_0 (lower left) is used to generate Gaussian pyramid levels g_1, g_2, … through repeated local averaging. Levels of the Laplacian pyramid L_0, L_1, … are then computed as the differences between adjacent Gaussian levels. Laplacian pyramid elements are quantized to yield the Laplacian pyramid code C_0, C_1, C_2, …. Finally, a reconstructed image r_0 is generated by summing levels of the code pyramid.

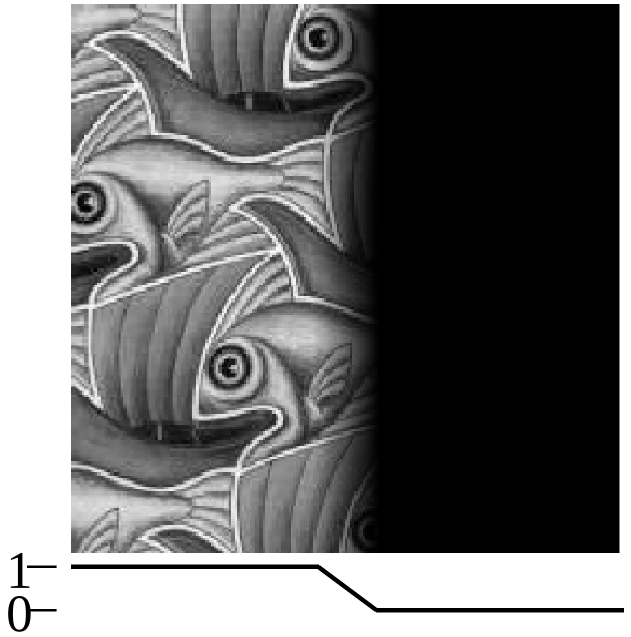
Image Blending



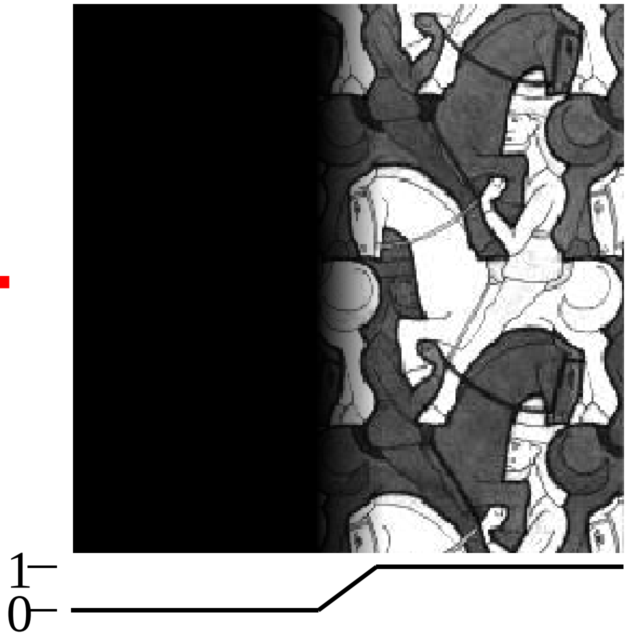




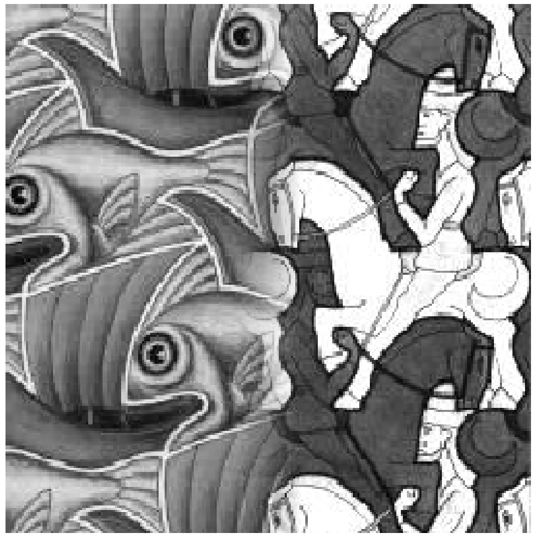
Feathering



+



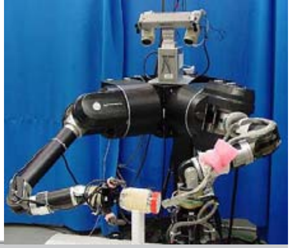
=



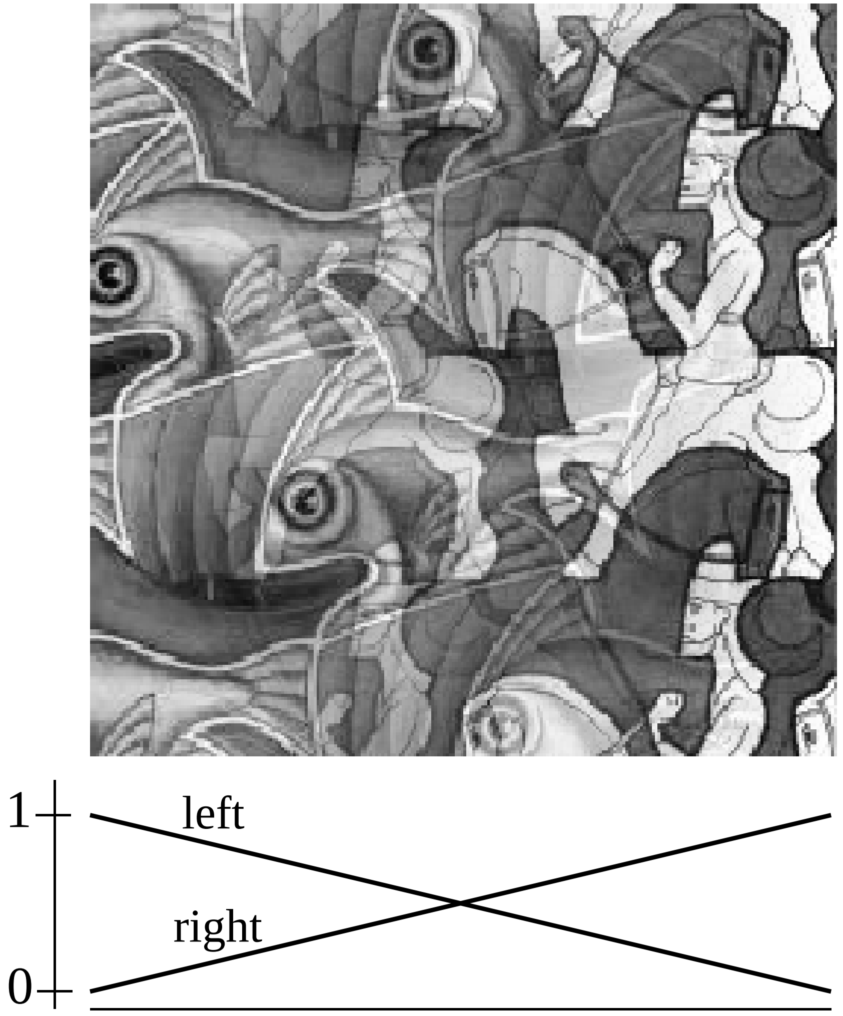
Encoding transparency

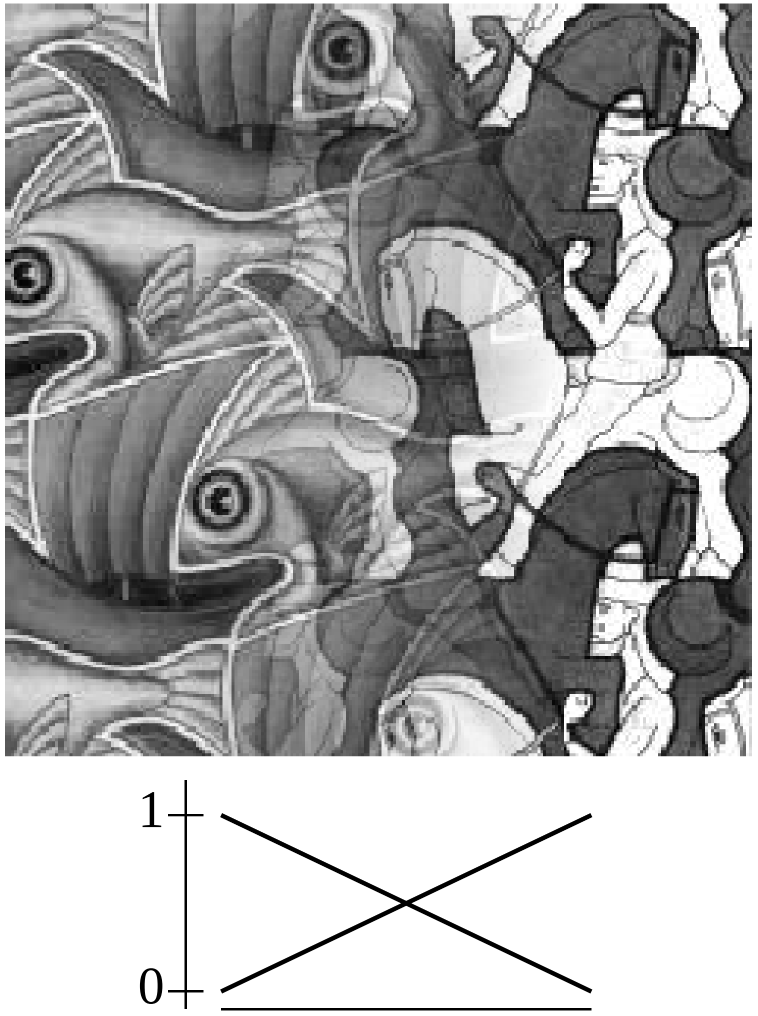
$I(x,y) = (\alpha R, \alpha G, \alpha B, \alpha)$

$I_{blend} = I_{left} + I_{right}$

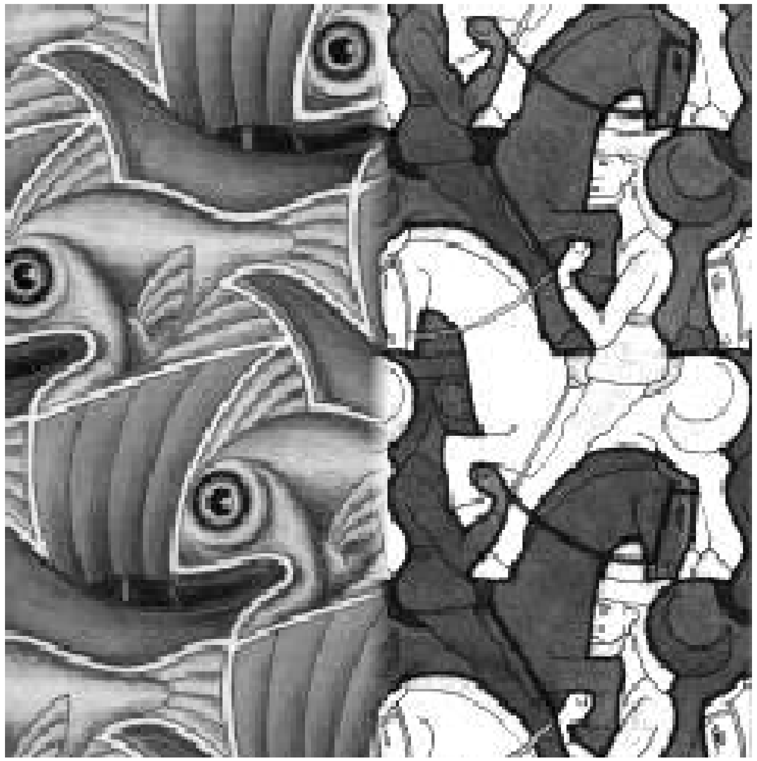


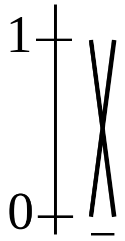
Affect of Window Size

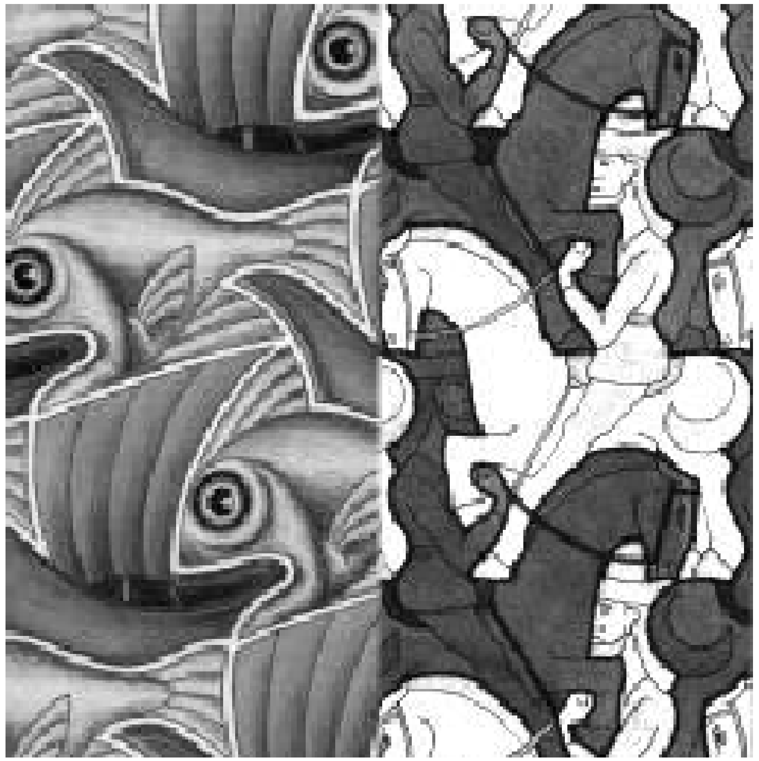


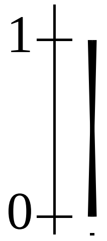


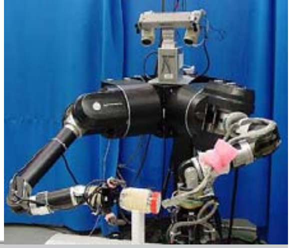
Affect of Window Size



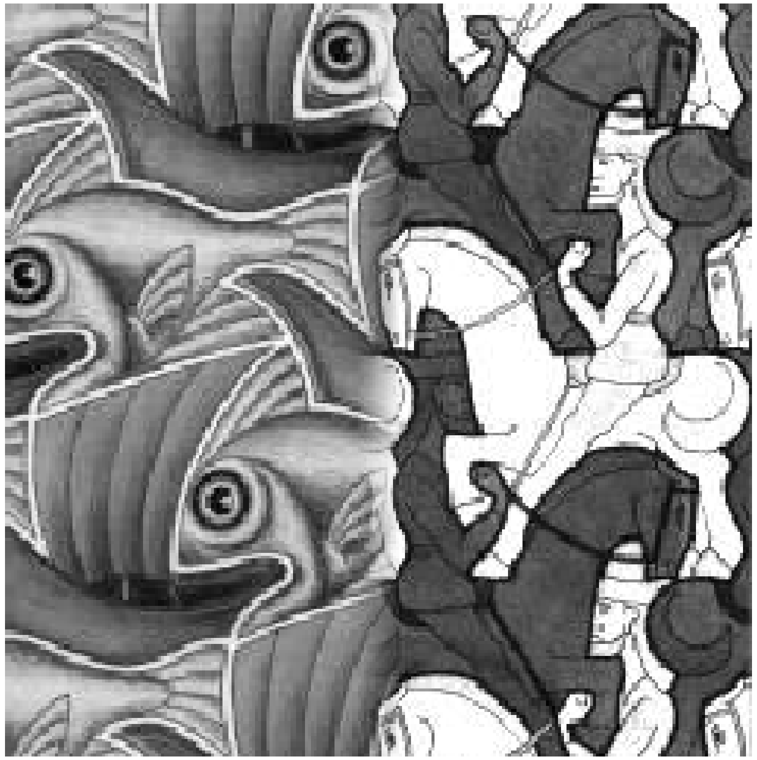


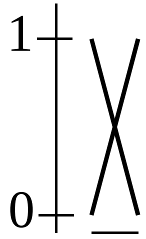






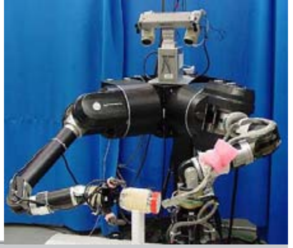
Good Window Size





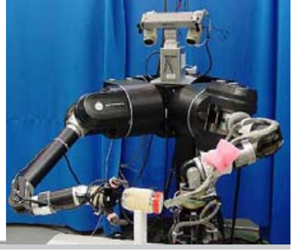
“Optimal” Window: smooth but not ghosted

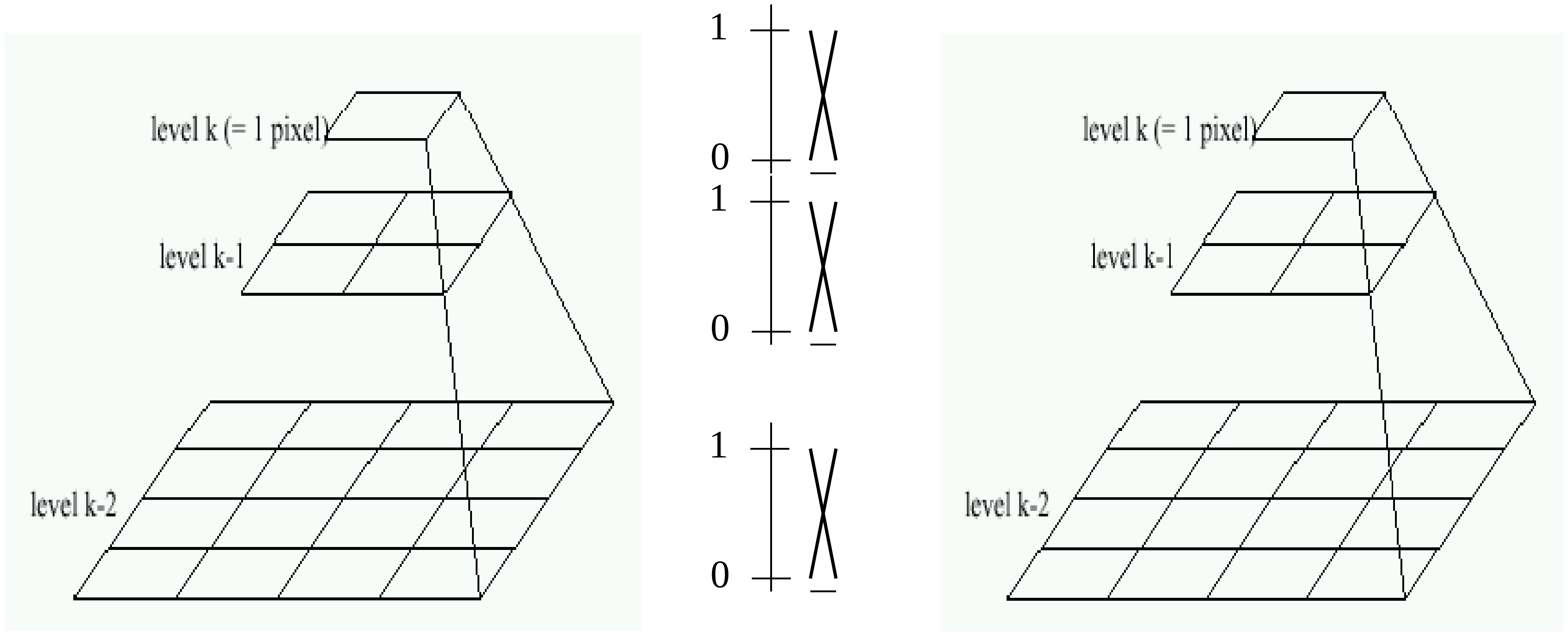
What is the Optimal Window?



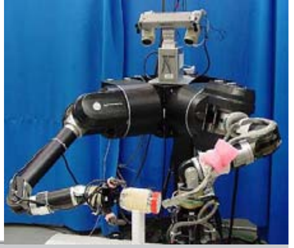
- To avoid seams
  - window >= size of largest prominent feature
- To avoid ghosting
  - window <= 2*size of smallest prominent feature

Pyramid Blending

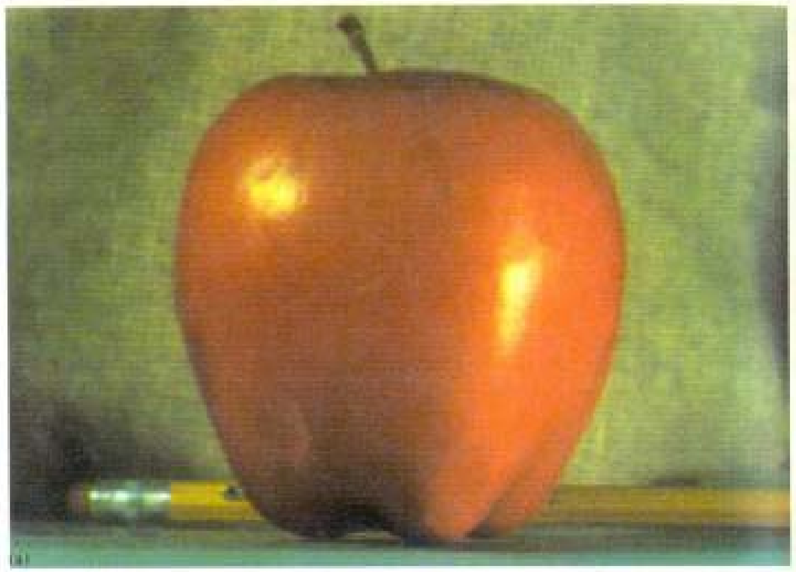


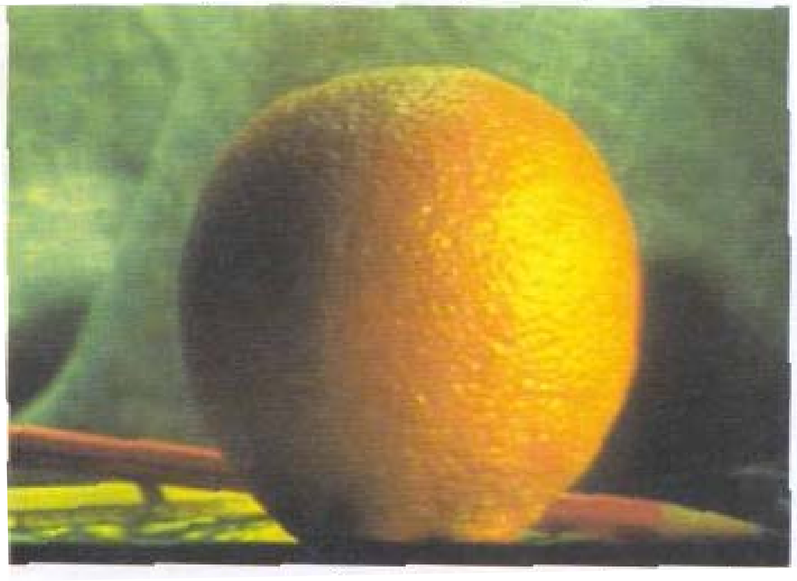


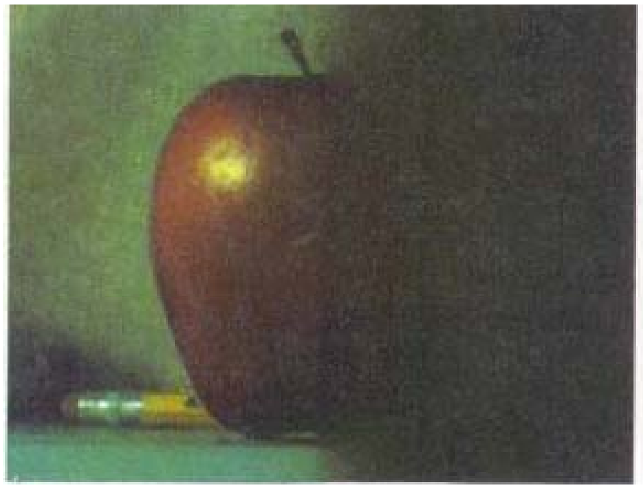
Left pyramid blend Right pyramid

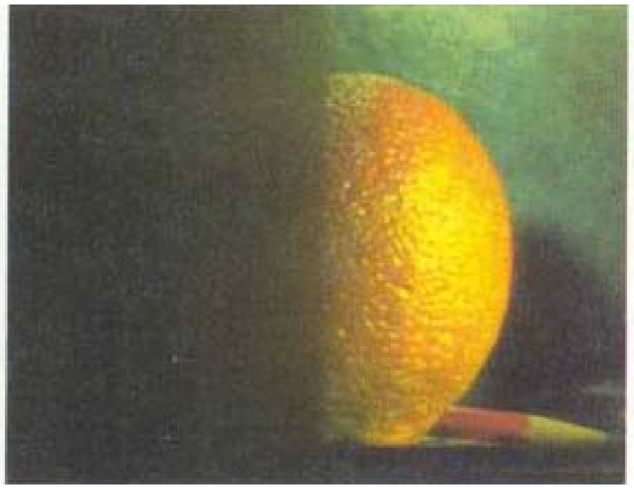


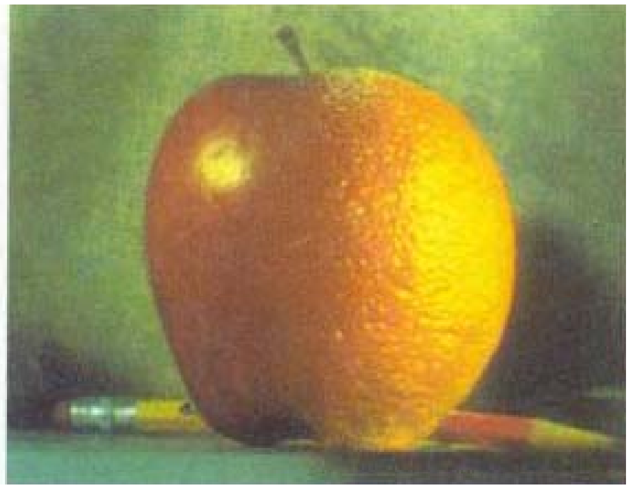
Pyramid Blending

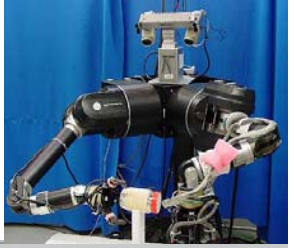








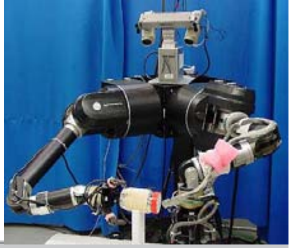




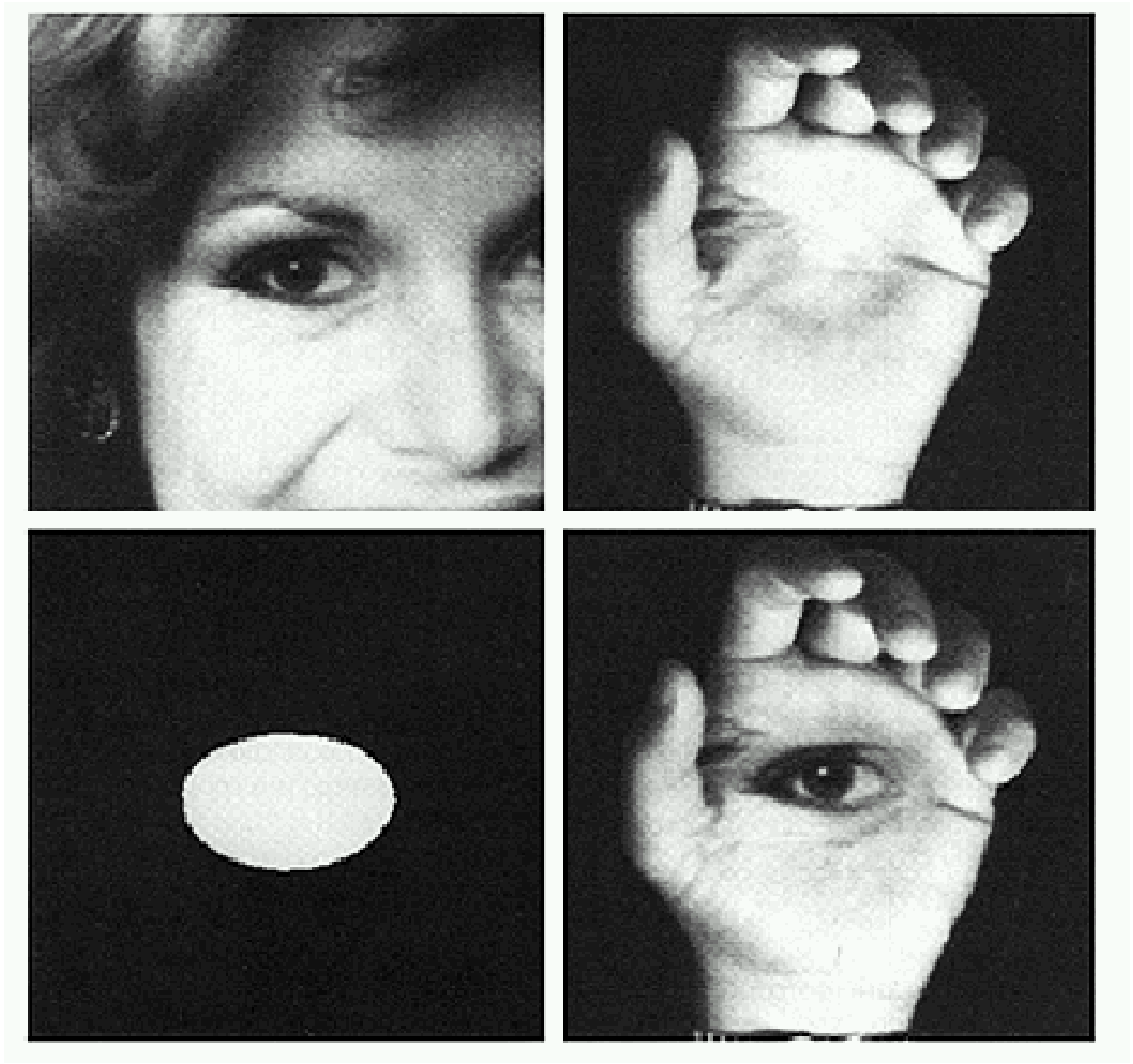
Laplacian Pyramid: Blending

- General Approach:
  - Build Laplacian pyramids LA and LB from images A and B
  - Build a Gaussian pyramid GR from selected region R
  - Form a combined pyramid LS from LA and LB using nodes of GR as weights:

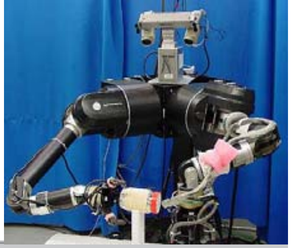
    *LS(i,j) = GR(I,j,)*LA(I,j) + (1-GR(I,j))*LB(I,j)*
  - Collapse the LS pyramid to get the final blended image

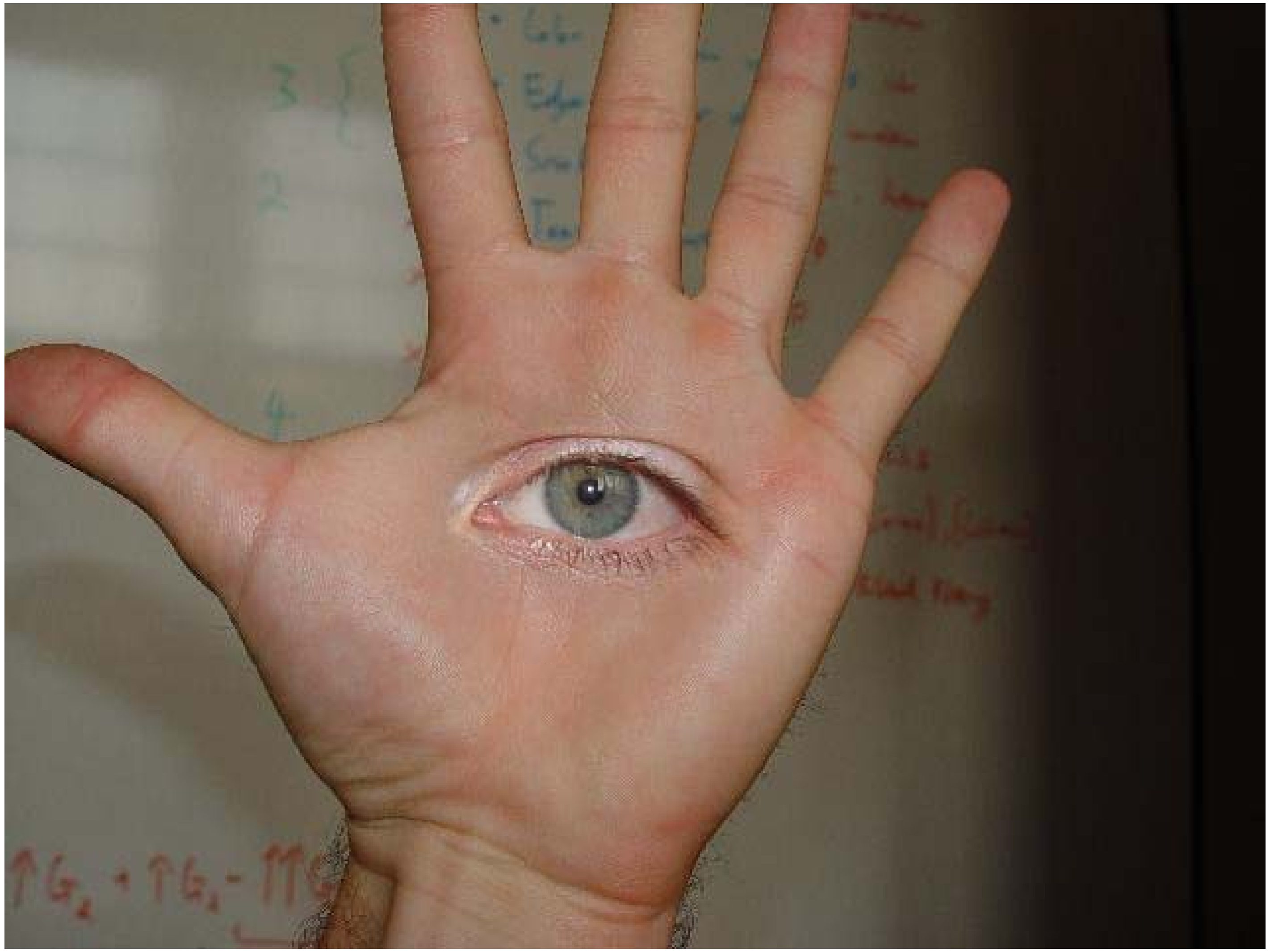


Blending Regions



Horror Photo





© prof. dmartin