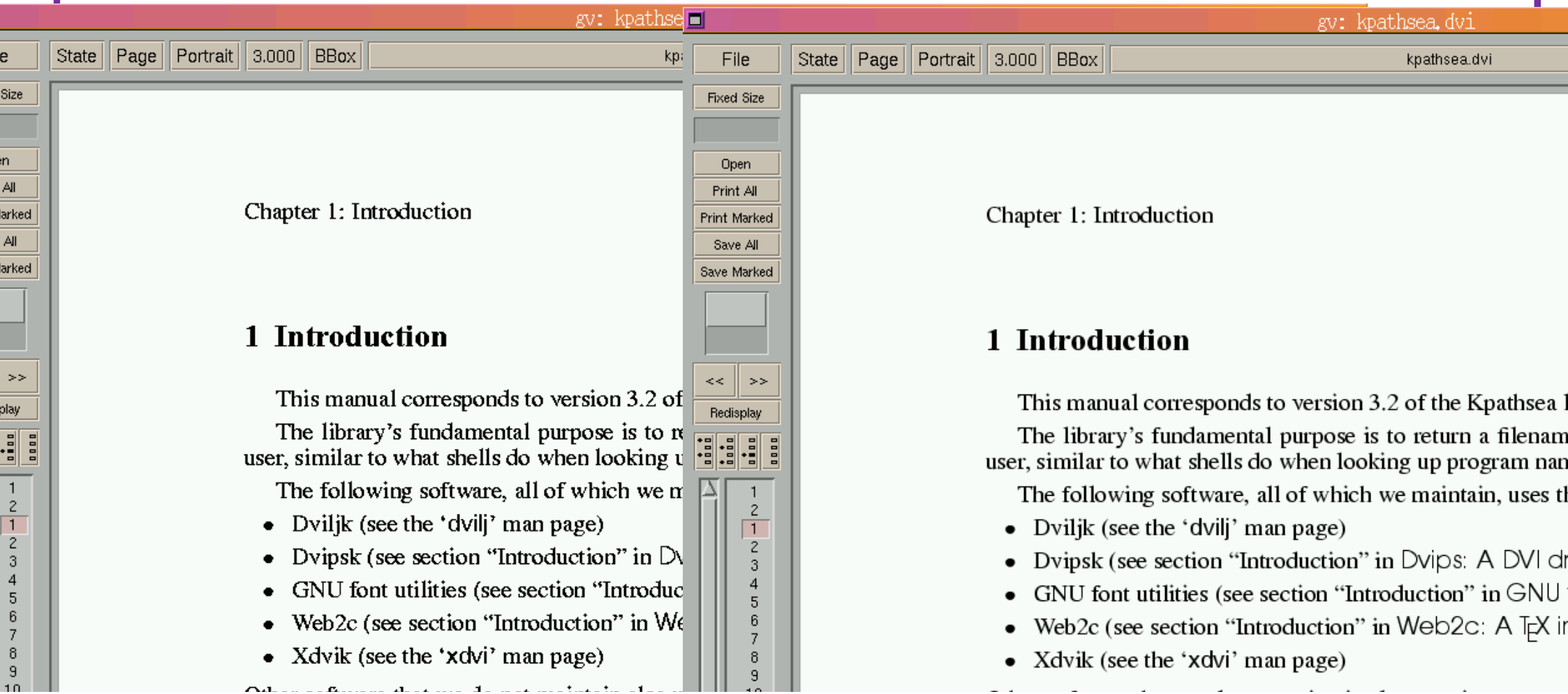
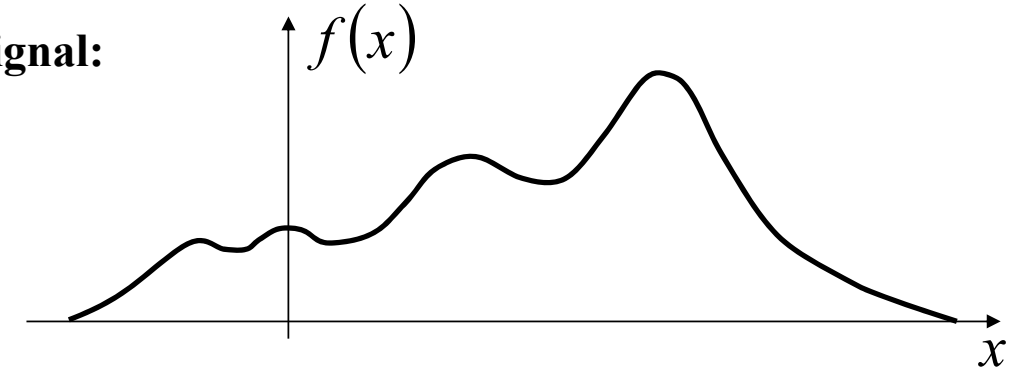
Image Resampling and Pyramids

Text Aliasing

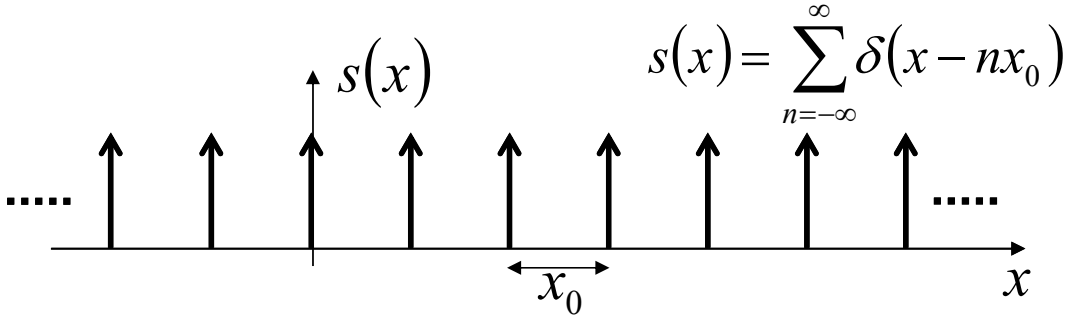


RECAP - Sampling Theorem

Continuous signal:



Shah function (Impulse train):



Sampled function:

$$f_s(x) = f(x)s(x) = f(x)\sum_{n=-\infty}^{\infty}\delta(x - nx_0)$$

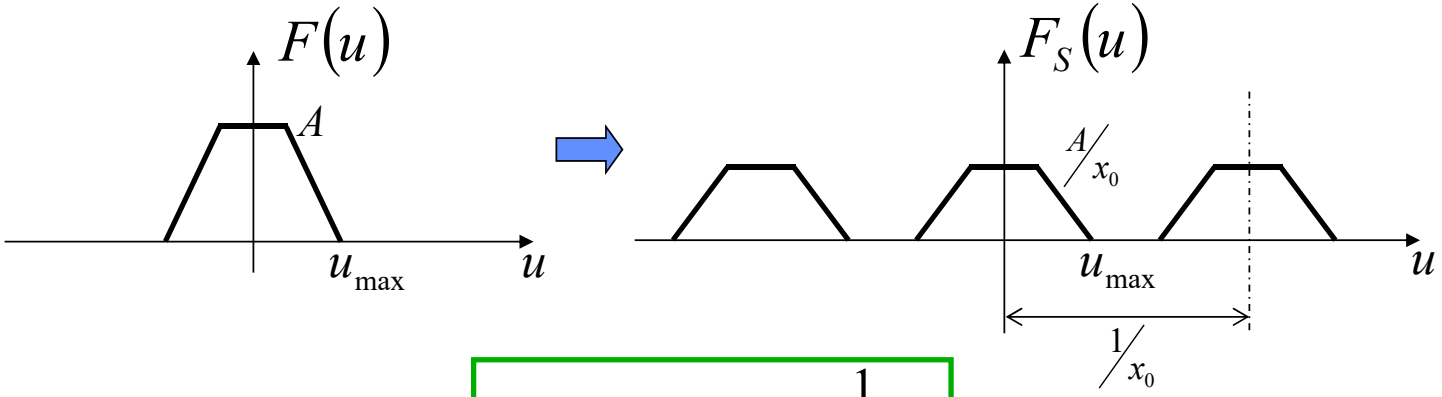
RECAP - Sampling Theorem

Sampled function:

$$f_s(x) = f(x)s(x) = f(x)\sum_{n=-\infty}^{\infty}\delta(x - nx_0)$$

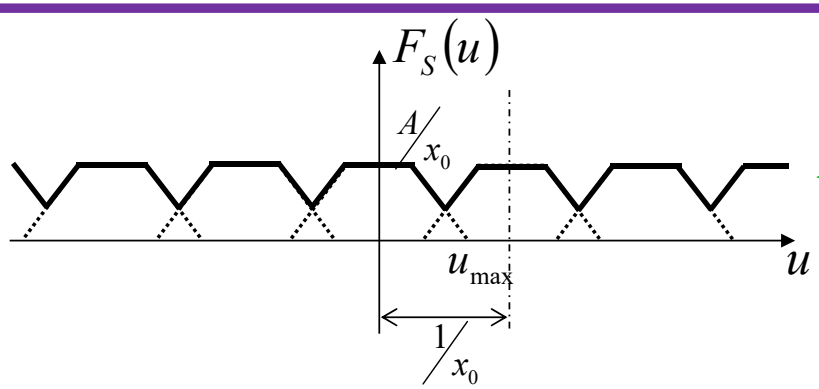
Sampling frequency $\frac{1}{x_0}$

$$F_S(u) = F(u) * S(u) = F(u) * \frac{1}{x_0}\sum_{n=-\infty}^{\infty}\delta\left(u - \frac{n}{x_0}\right)$$



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

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



Aliasing

When can we recover $F(u)$ from $F_S(u)$?

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

We can use

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

Then $F(u) = F_S(u)C(u)$ **and** $f(x) = \text{IFT}[F(u)]$

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

Aliasing

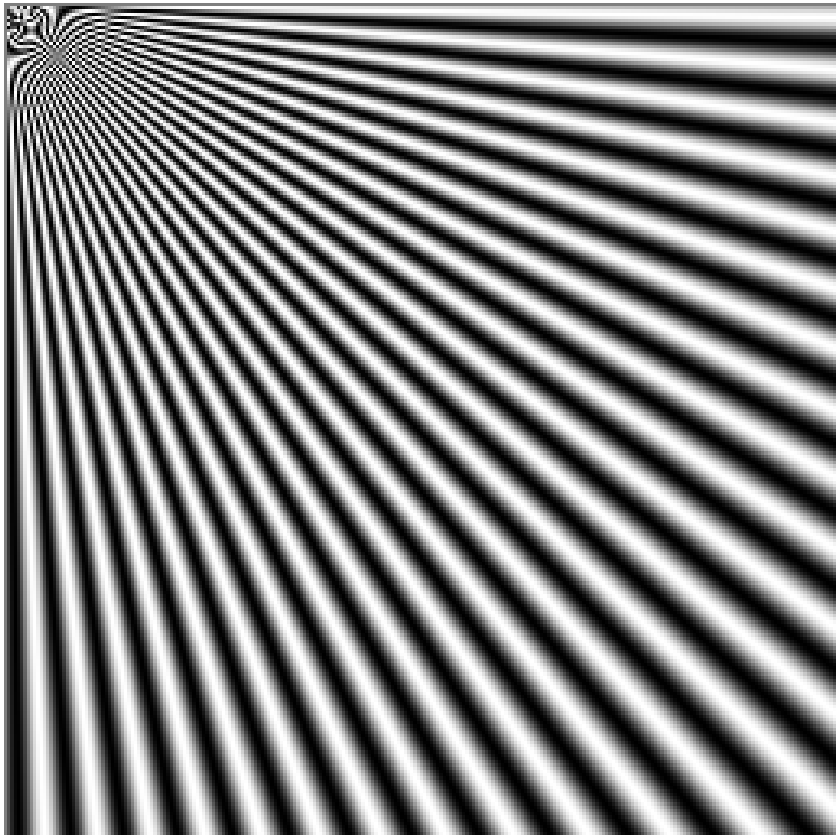


Image Scaling

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?

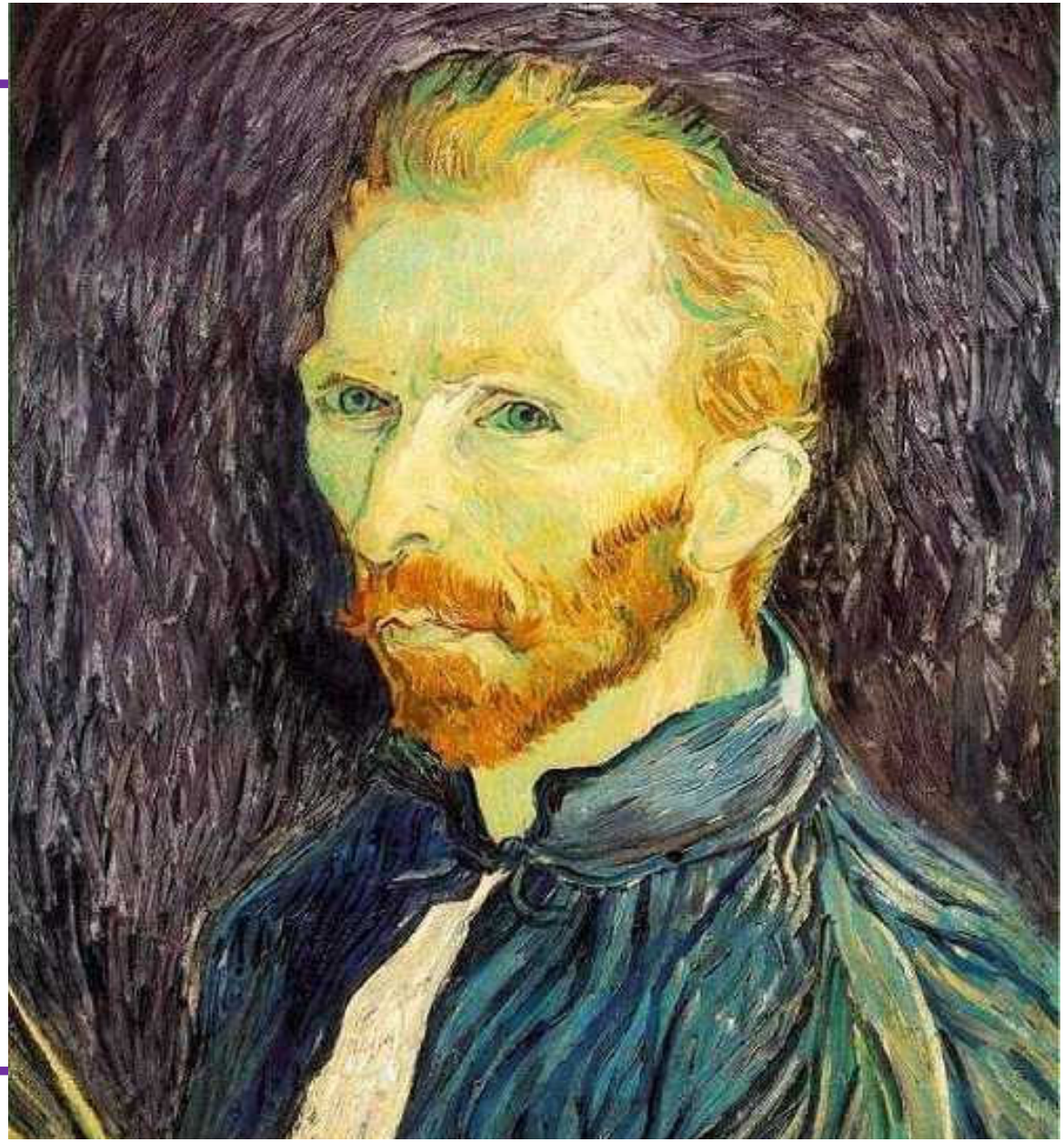
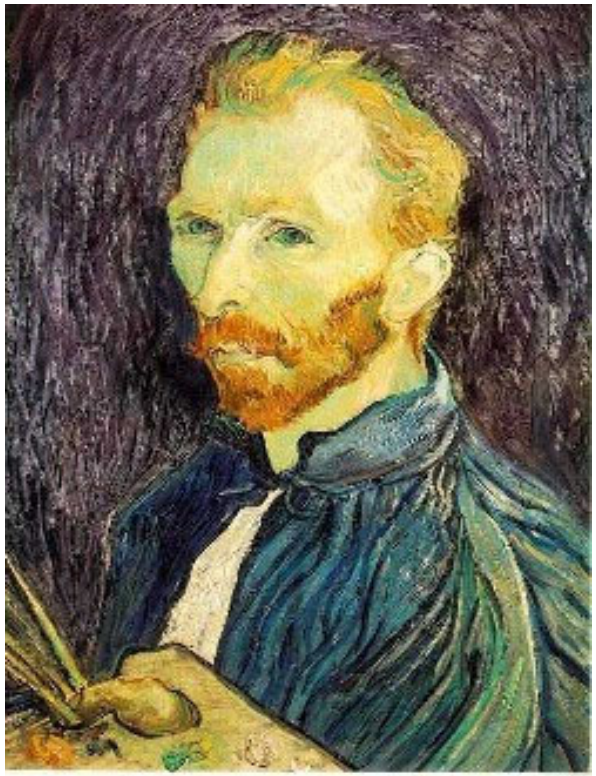


Image Sub-Sampling

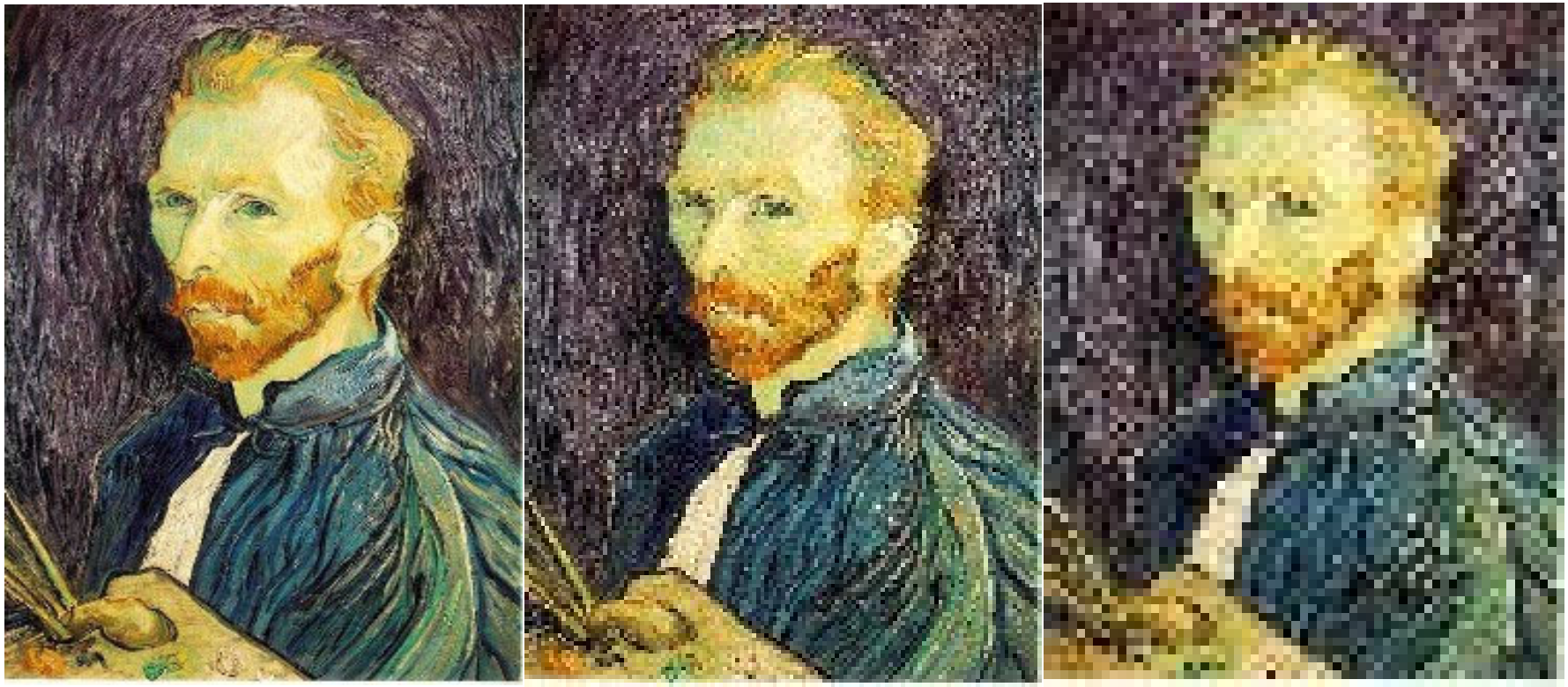


1/4

1/8

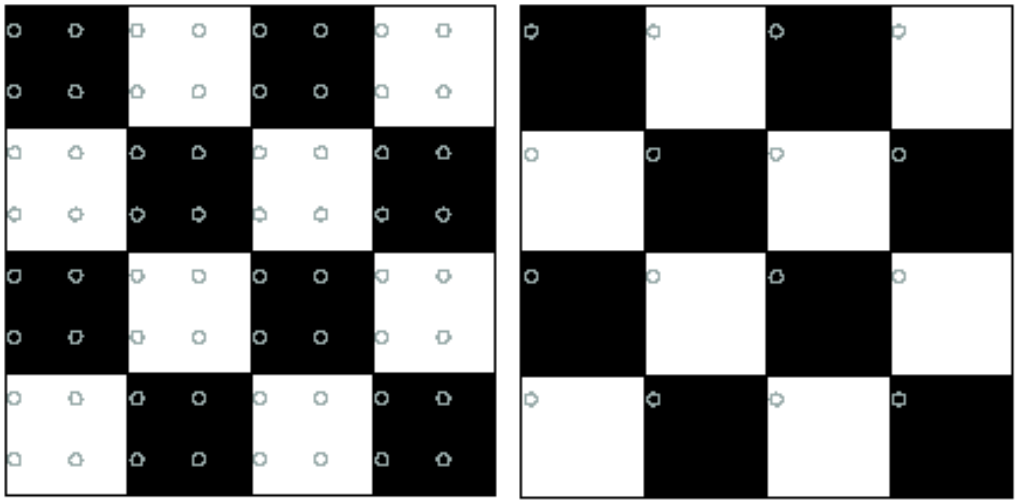
Throw away every other row and column to create a 1/2 size image - called *image sub-sampling*

Image Sub-Sampling

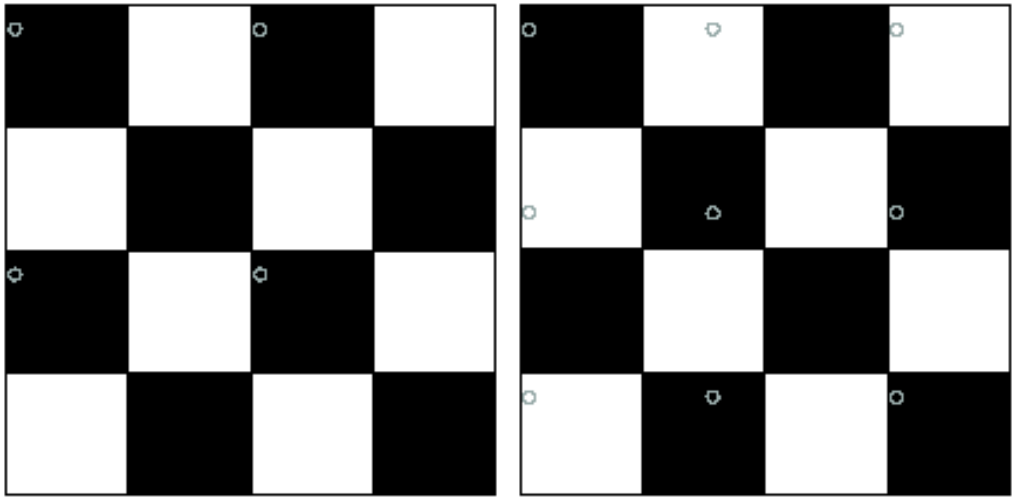


1/2 **1/4 (2x zoom)** **1/8 (4x zoom)**

Good and Bad Sampling

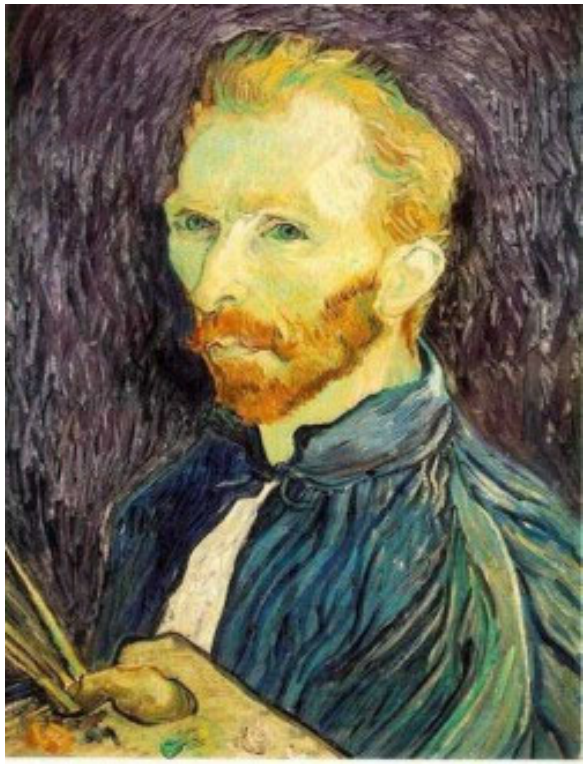


Good sampling:
- **Sample often or,**
- **Sample wisely**



Bad sampling:
- **see aliasing in action!**

Sub-Sampling with Gaussian Pre-Filtering



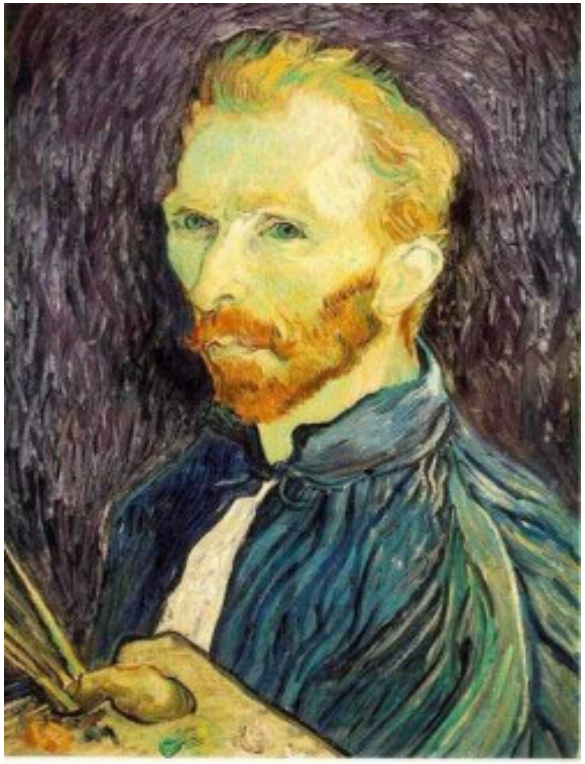
Gaussian 1/2

G 1/4

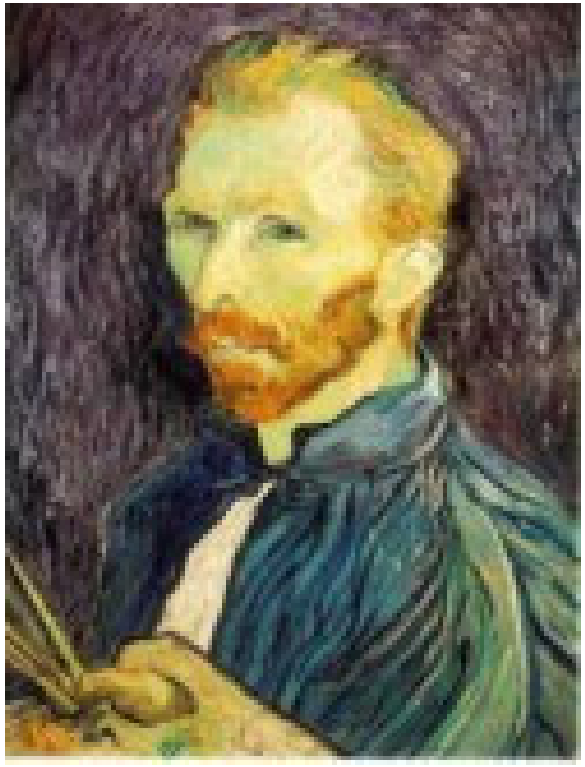
G 1/8

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

Sub-Sampling with Gaussian Pre-Filtering



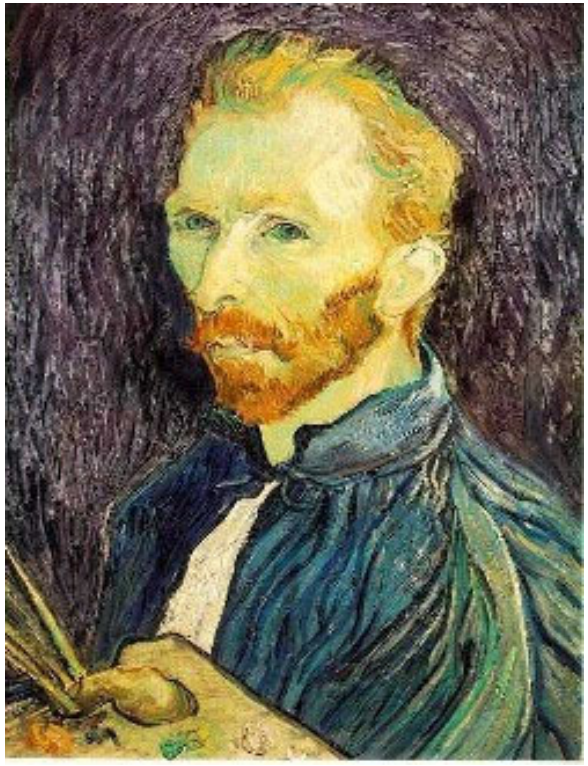
Gaussian 1/2



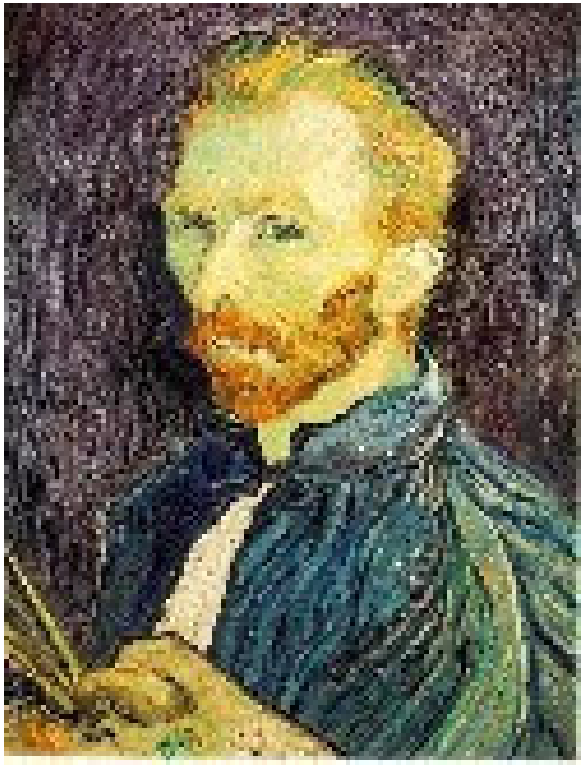
G 1/4

G 1/8

Compare with...



1/2

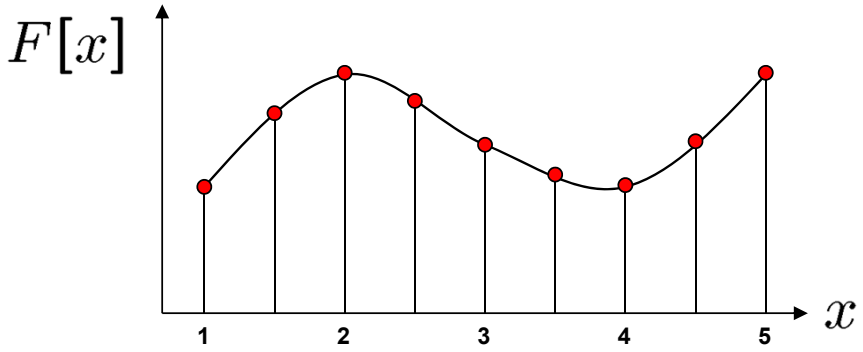


1/4 **(2x zoom)**

1/8 **(4x zoom)**

Image Resampling

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



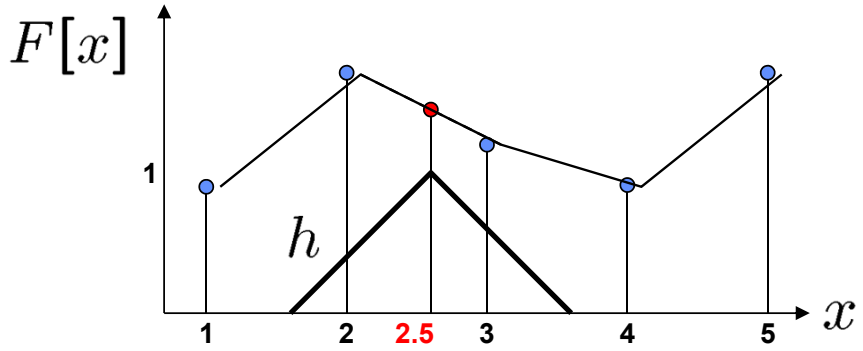
- **Recall how a digital image is formed**

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

  - **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**

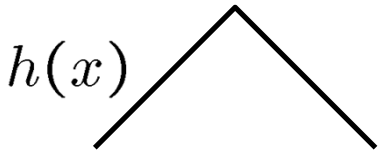
$$f_F(x) = F(\tfrac{x}{d}) \text{ when } \tfrac{x}{d} \text{ is an integer, 0 otherwise}$$

  - **Reconstruct by convolution:**

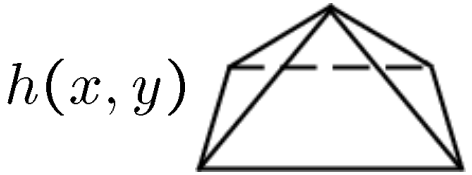
$$\tilde{f} = h \otimes f_F$$

Resampling filters

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



performs linear interpolation

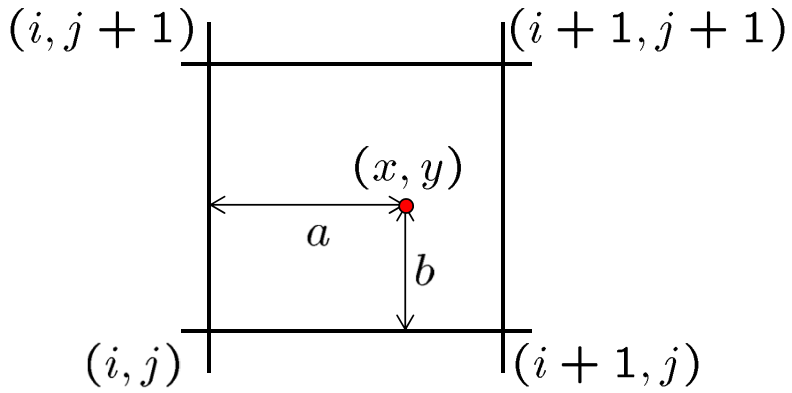


performs bilinear interpolation

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

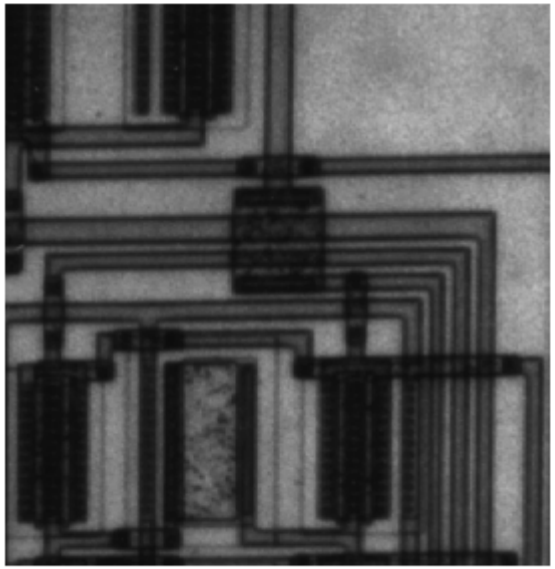
Bilinear interpolation

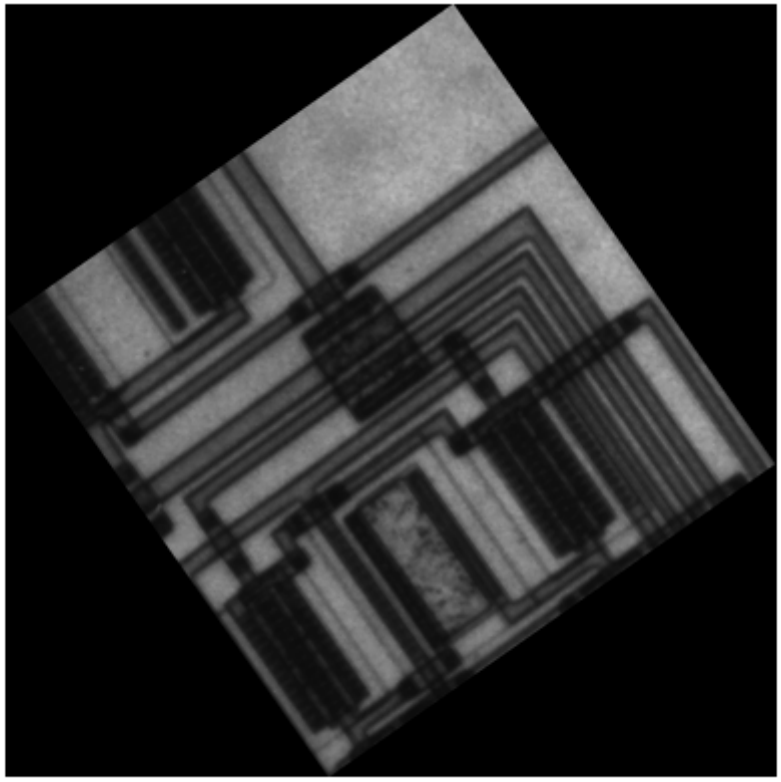
- A common method for resampling images



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

Image Rotation

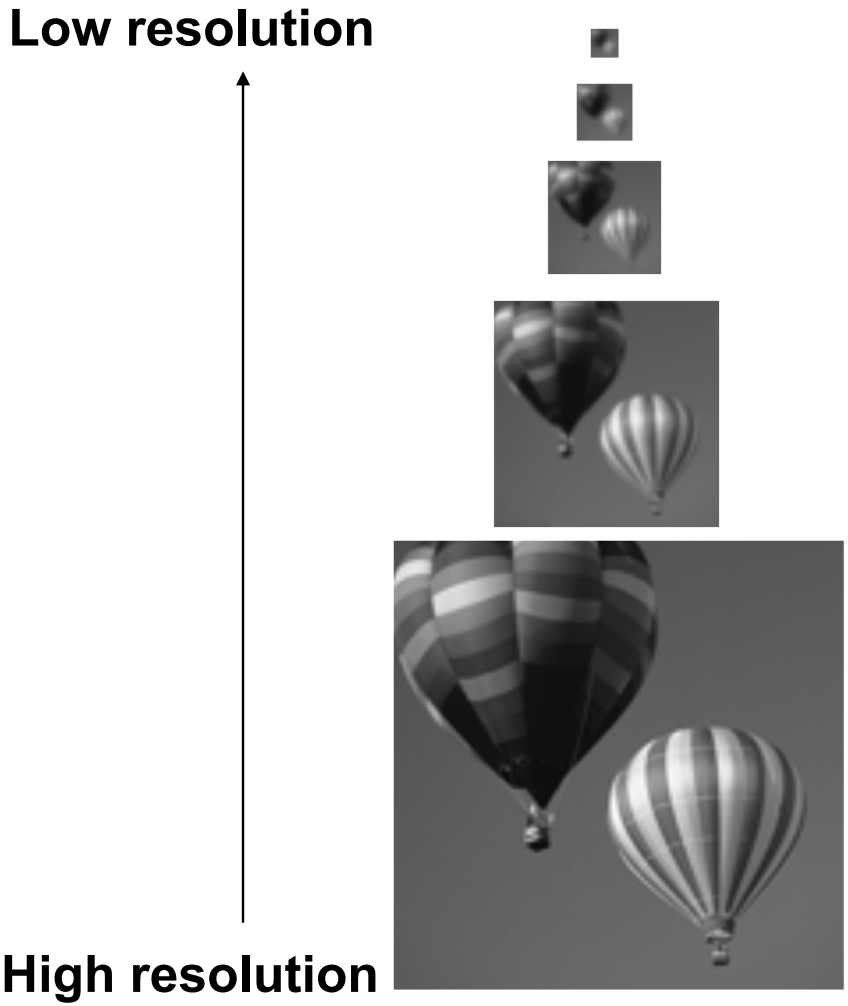




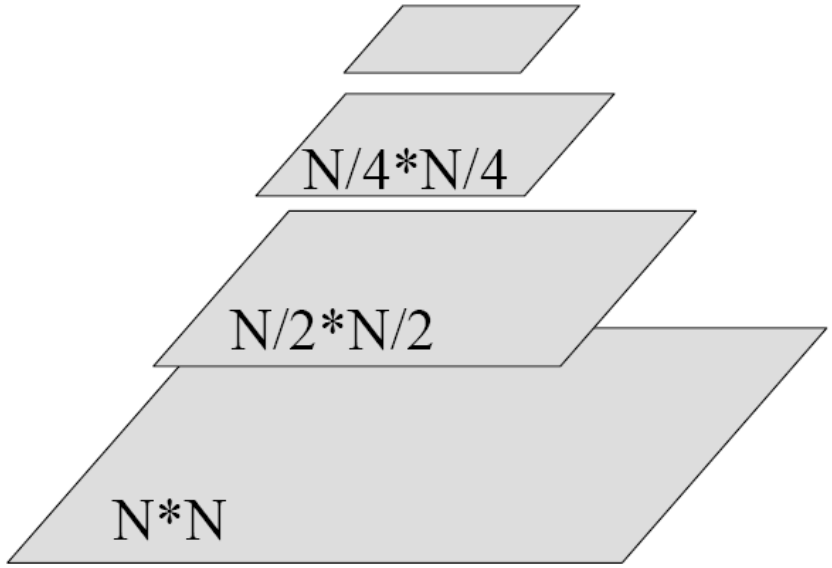
Multi-Resolution Image Representation

- **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

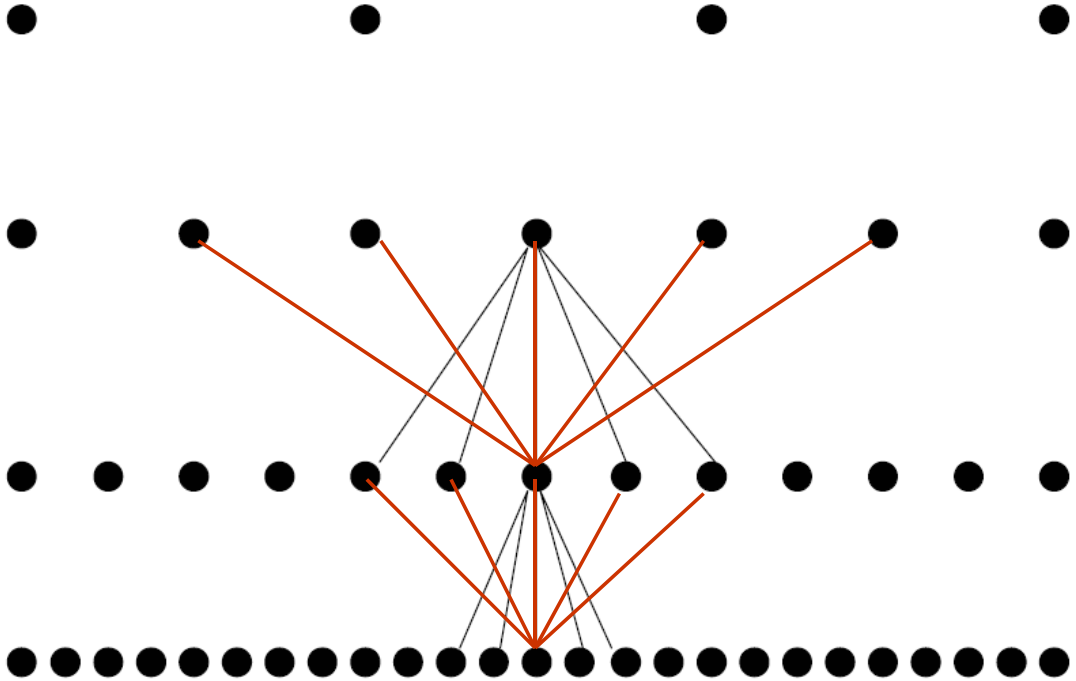


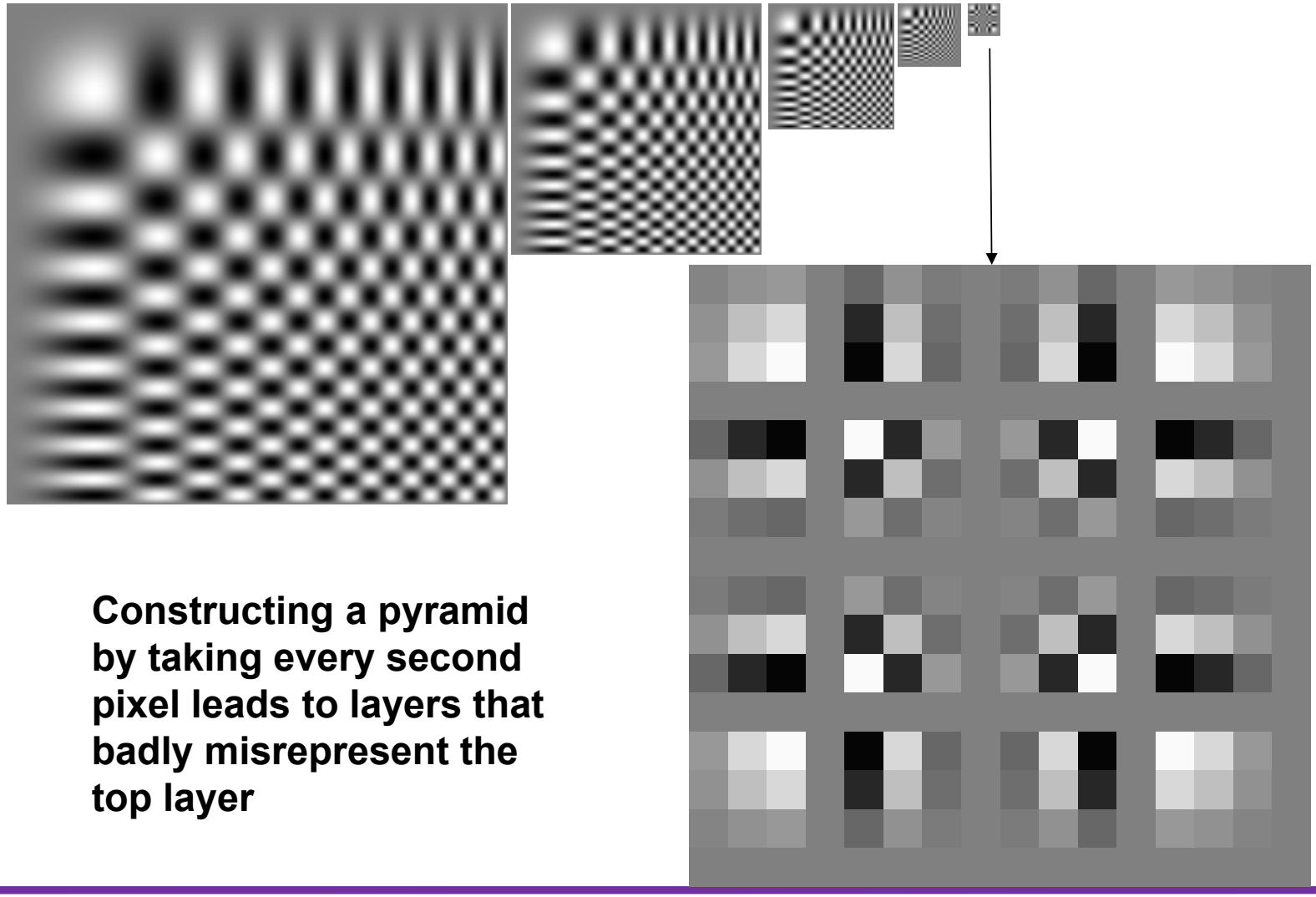
Space Required for Pyramids



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

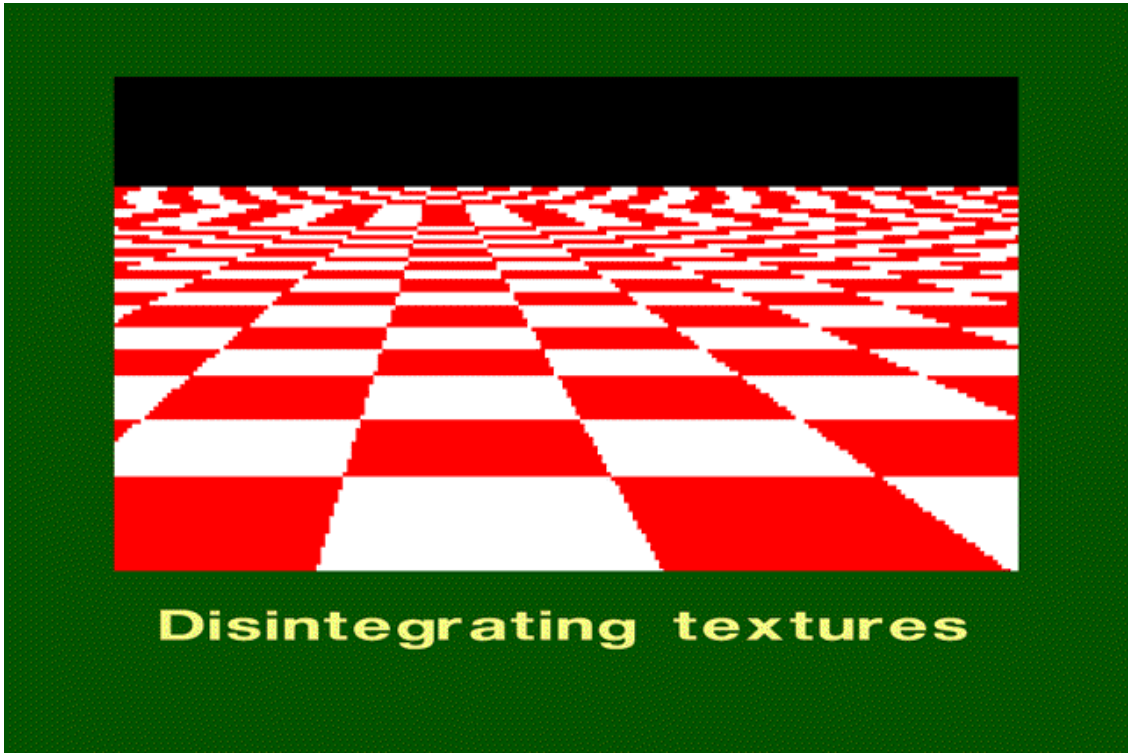
Pyramid Construction





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

Even worse for synthetic images



Downsampling

*

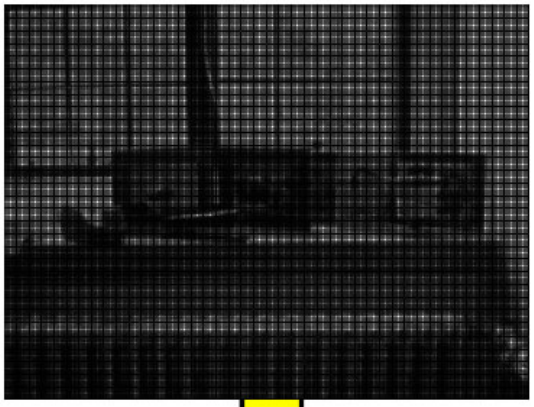
1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

Filter

Subsample

Expansion

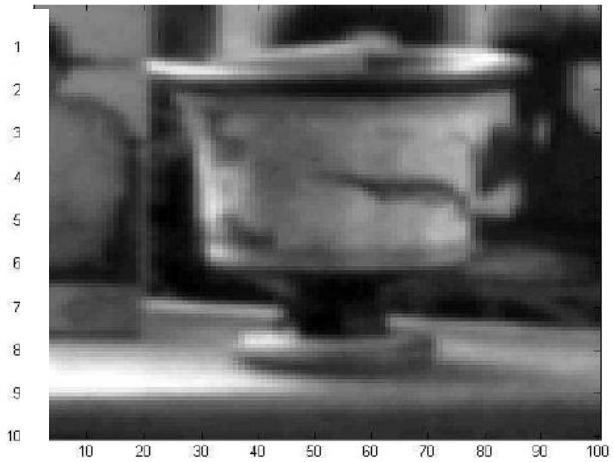
Expand Image



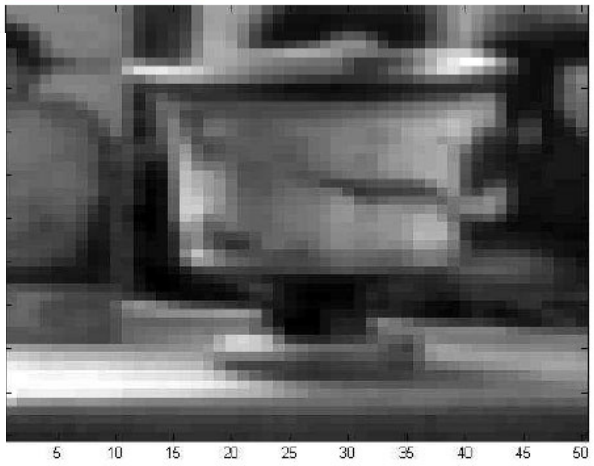
Interpolation

Interpolation Results

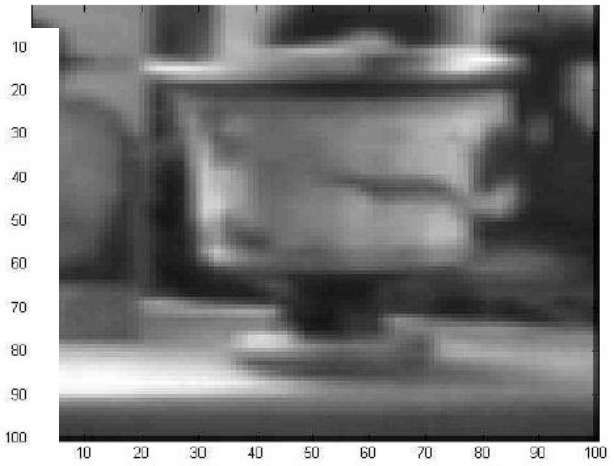
Original Image



Nearest Neighbor

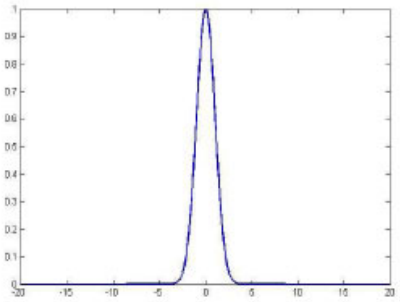


Bilinear Interpolation

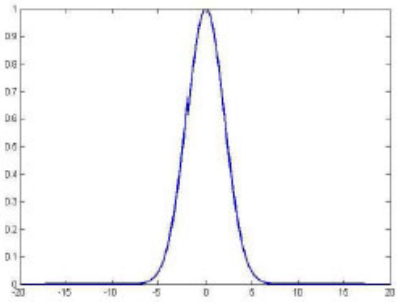


The Gaussian Pyramid

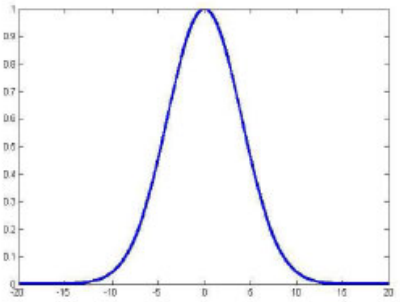
- Smooth with Gaussians because
  - a Gaussian*Gaussian=another Gaussian
- Synthesis
  - smooth and downsample
- Gaussians are low pass filters, so repetition is redundant
- Kernel width doubles with each level



Level 1



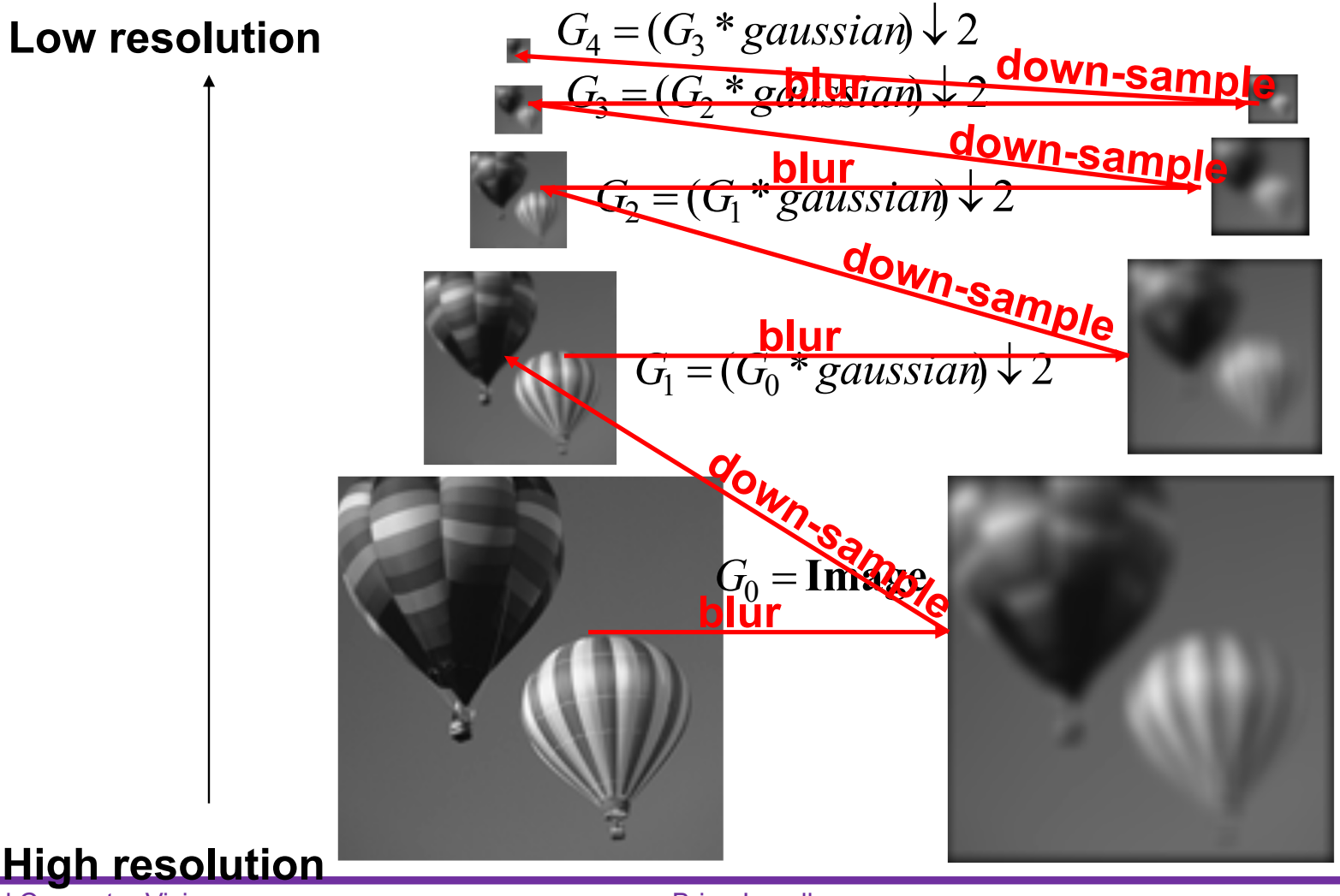
Level 2

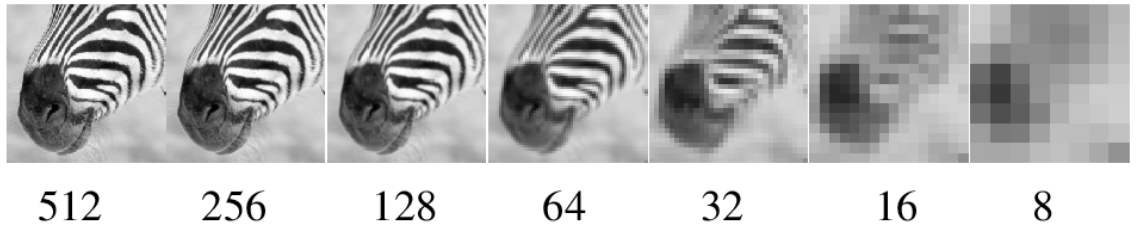


Level 3

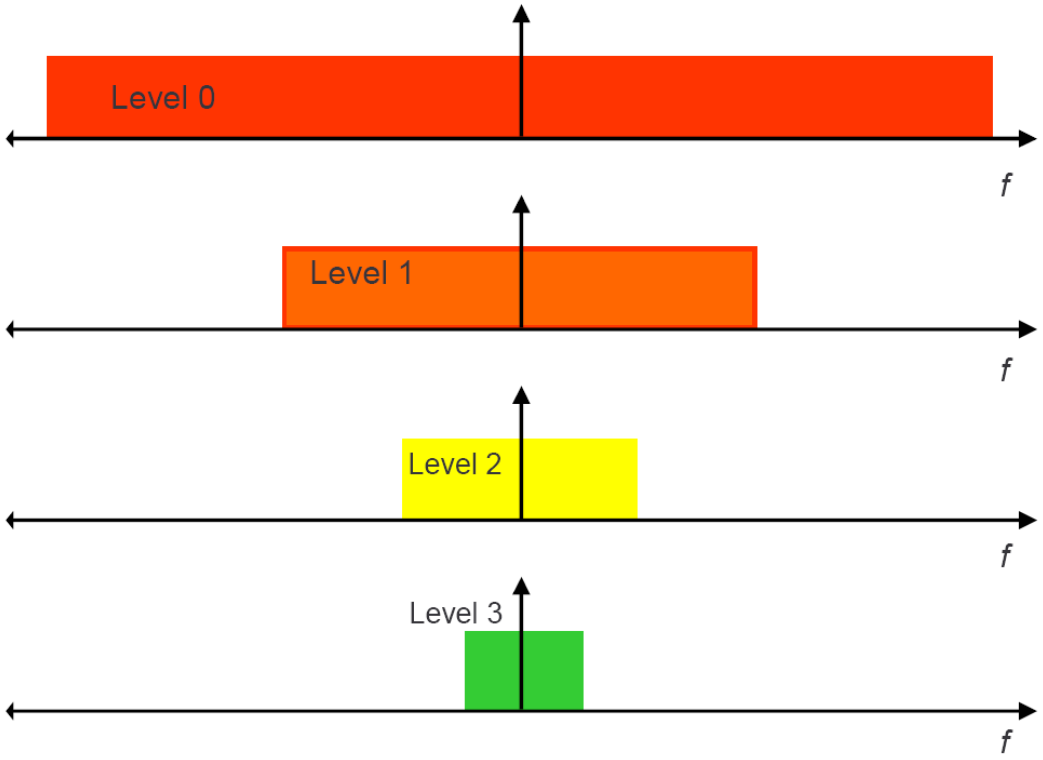
Smoothing as low-pass filtering

- 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.

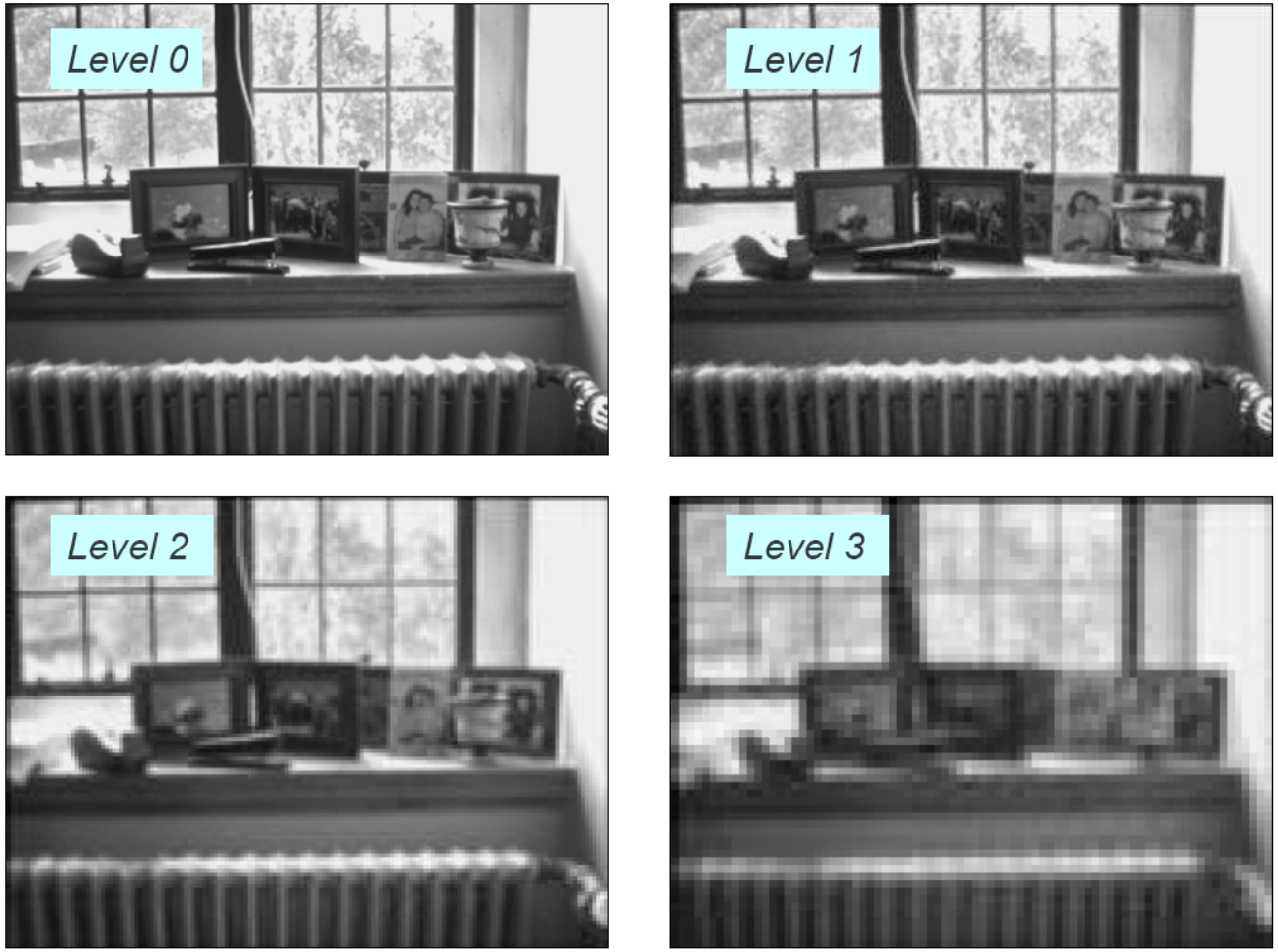

Low resolution
$G_4 = (G_3 * gaussian) \downarrow 2$
blur
down-sample
$G_3 = (G_2 * gaussian) \downarrow 2$
down-sample
blur
$G_2 = (G_1 * gaussian) \downarrow 2$
down-sample
blur
$G_1 = (G_0 * gaussian) \downarrow 2$
down-sample
$G_0 = \text{Image}$
blur
High resolution


512
256
128
64
32
16
8

Gaussian Pyramid Frequency Composition



Pyramids at Same Resolution



Difference of Gaussians (DoG)

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

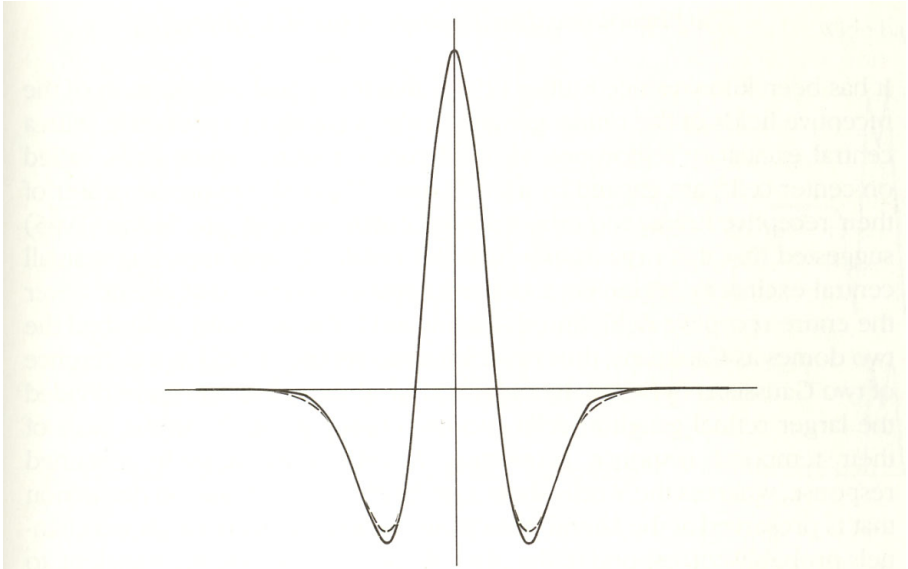
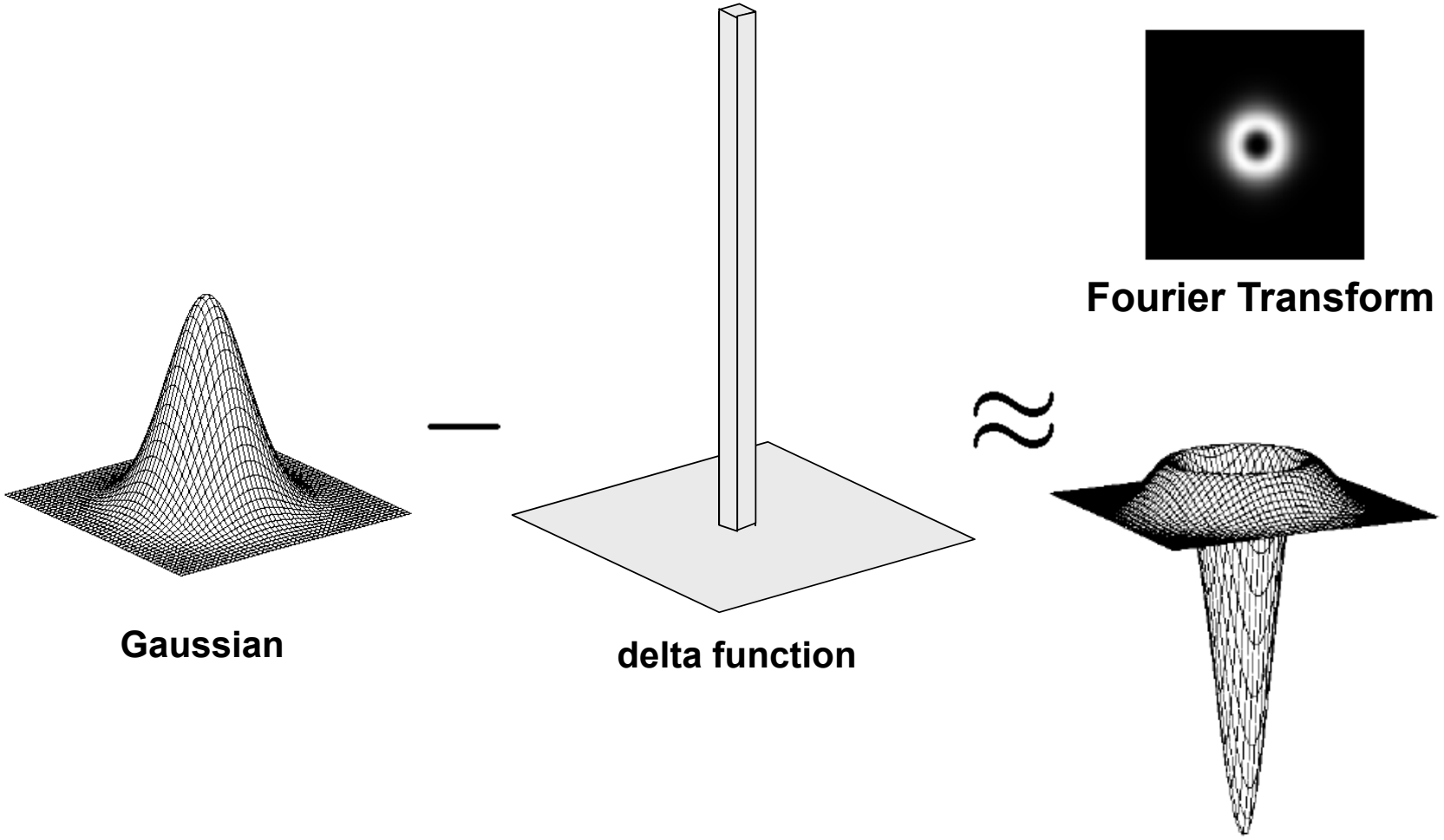


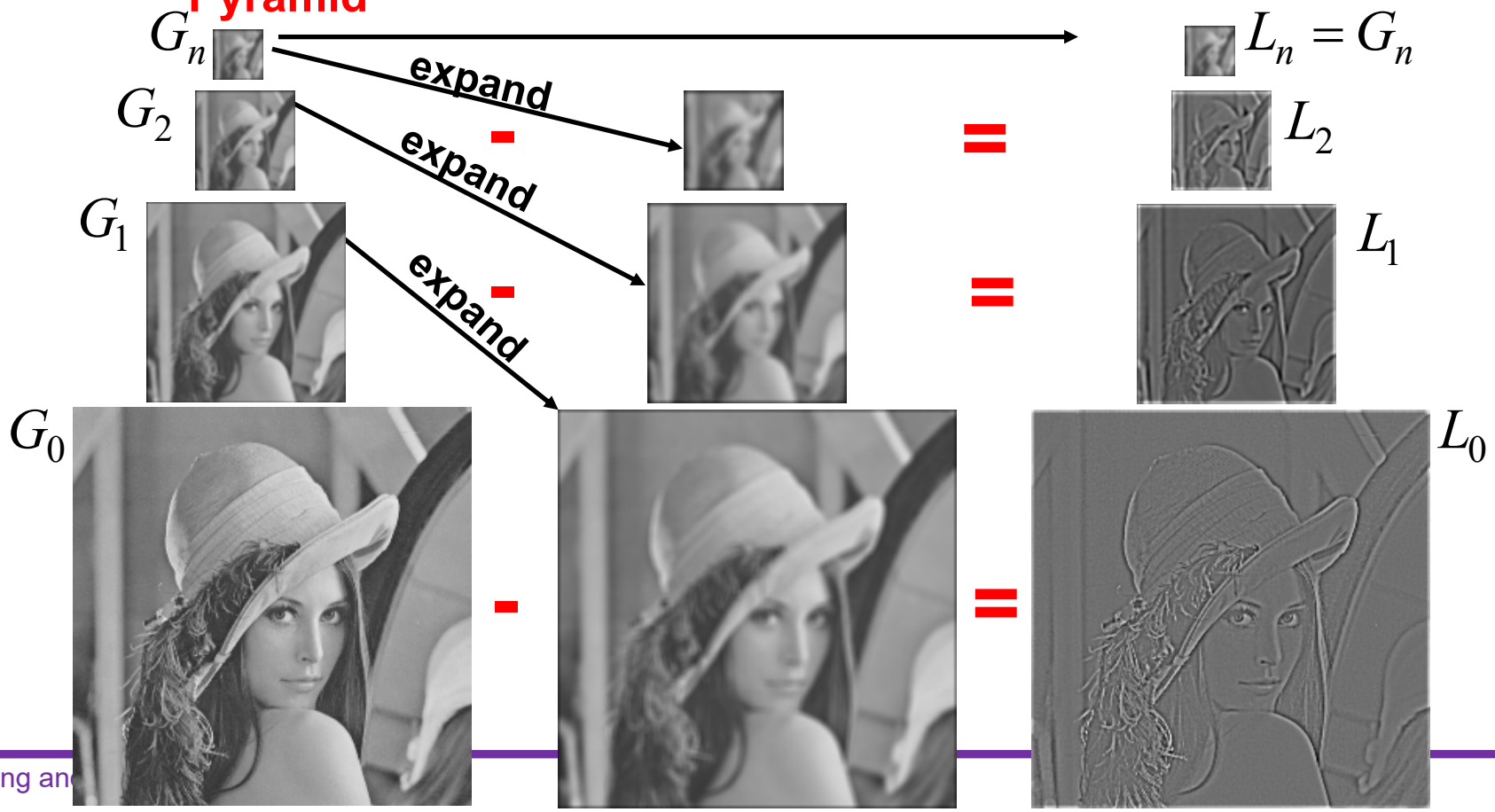
Figure 2–16. The best engineering approximation to $\nabla^2 G$ (shown by the continuous line), obtained by using the difference of two Gaussians (DOG), occurs when the ratio of the inhibitory to excitatory space constraints is about 1:1.6. The DOG is shown here dotted. The two profiles are very similar. (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

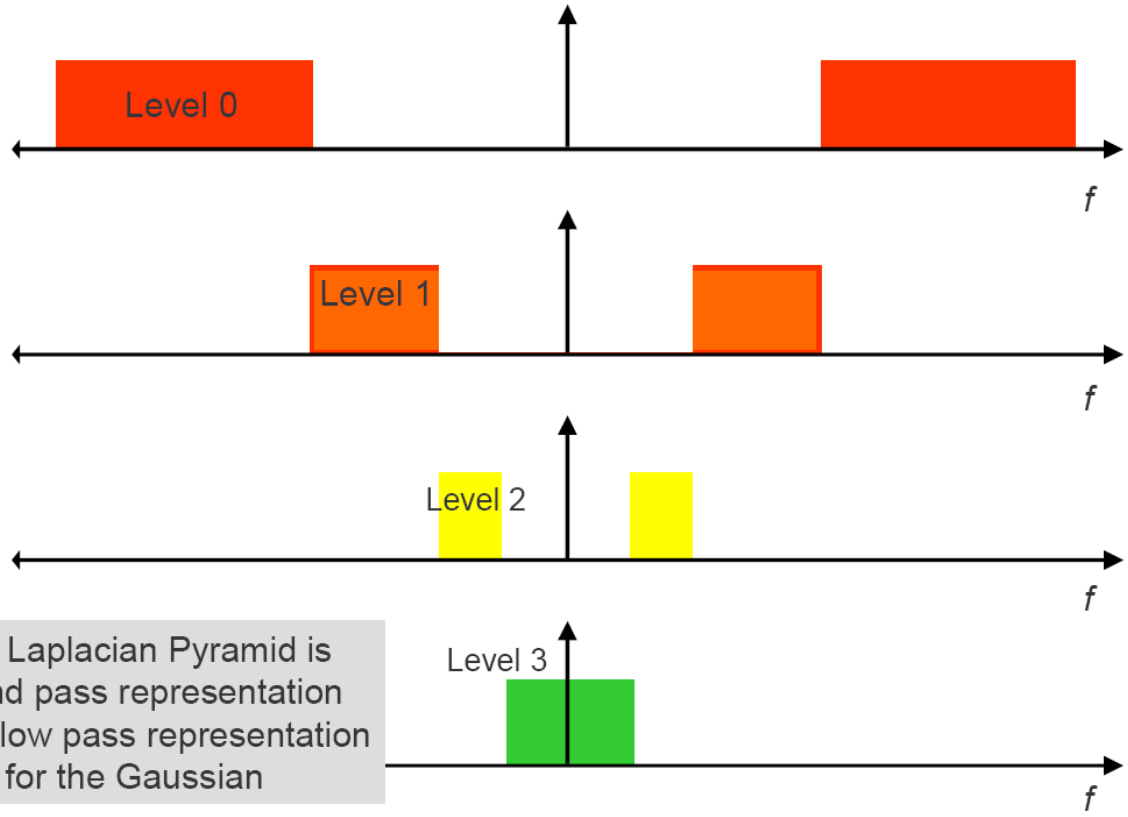


$$L_i = G_i - \text{expand}(G_{i+1})$$

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



Laplacian Pyramid Frequency Composition



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

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

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.**

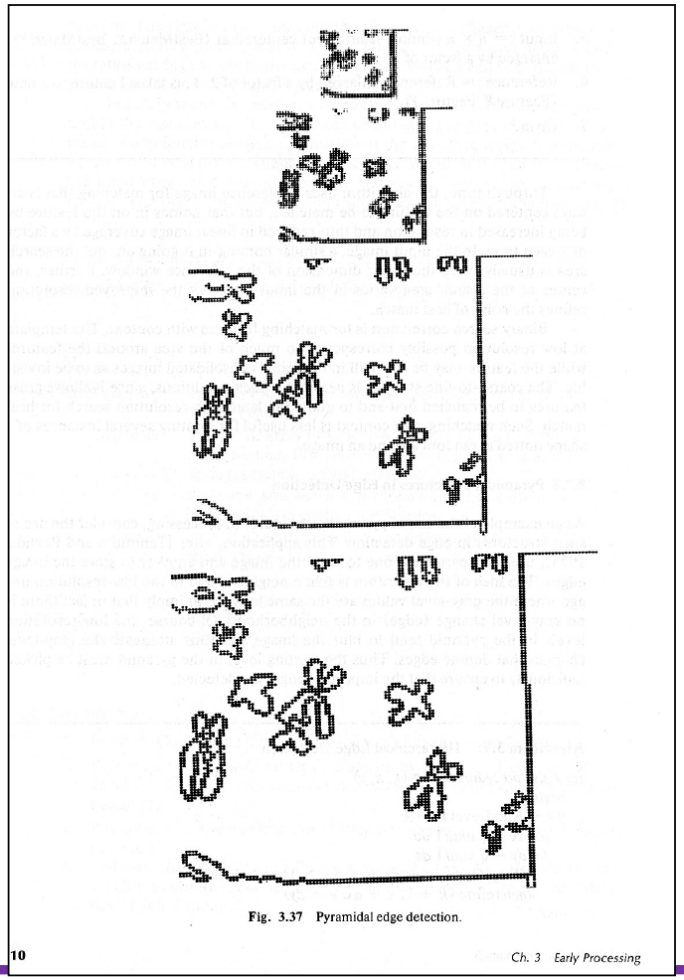


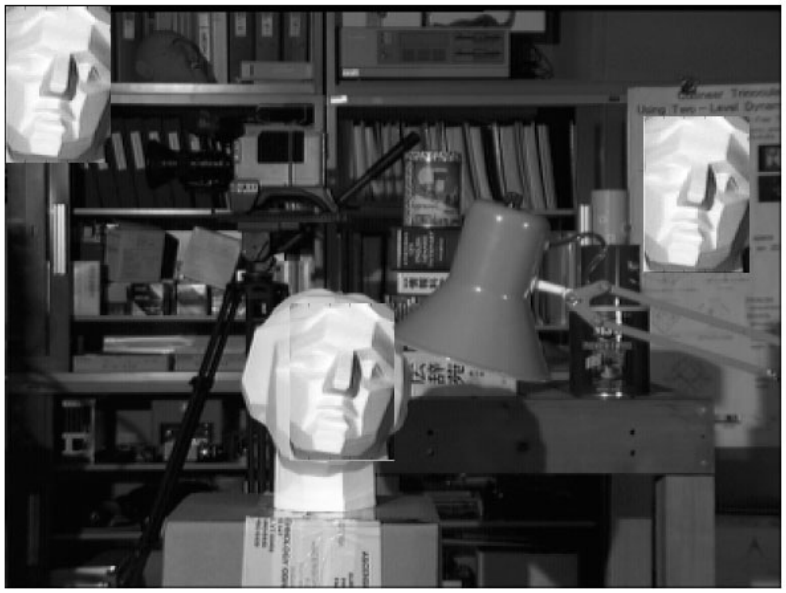
Fig. 3.37 Pyramidal edge detection.

10 Ch. 3 Early Processing

Fast Template Matching

Template

Search Region



- 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 address scale and/or rotational effects)

Fast Template Matching

Template

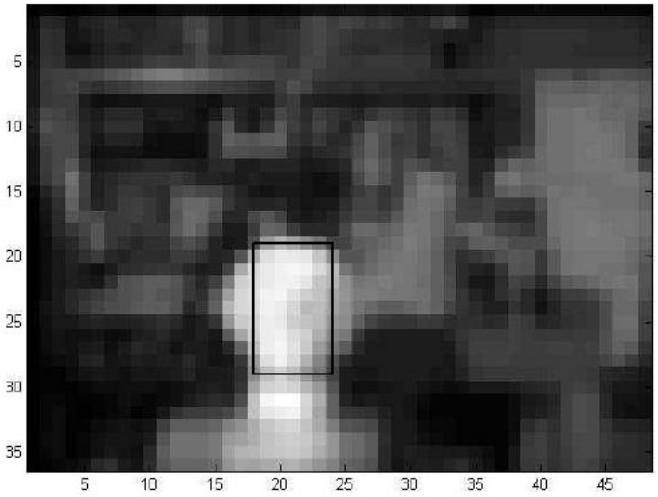
Search Region

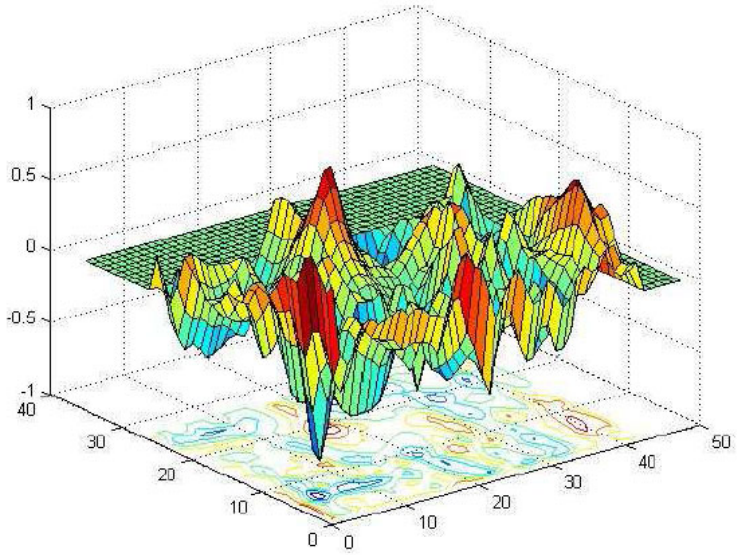
Original Image

- Multi-resolution template matching
  - reduce resolution of both template and image by creating an **image pyramid**
  - match small template against small image
  - identify locations of strong matches
  - expand the image and template, and match higher resolution template selectively to higher resolution image
  - iterate on higher and higher resolution images
- Issue:
  - how to choose detection thresholds at each level
    - too low will lead to too much cost
    - too high will miss match

Level 3 Search

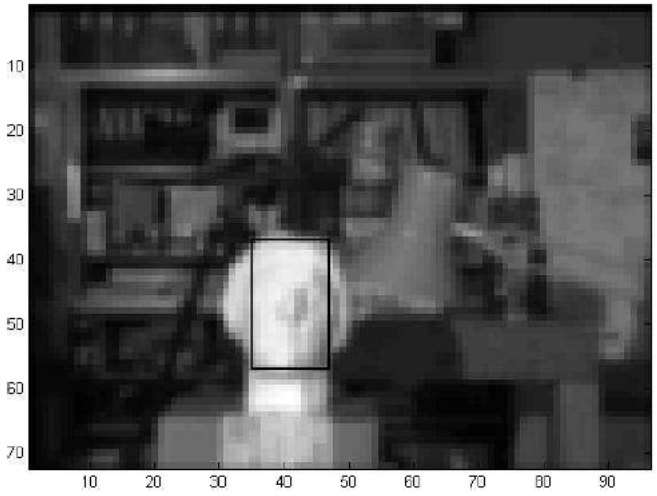
- At the lowest pyramid level, we search the entire image with the correlation template

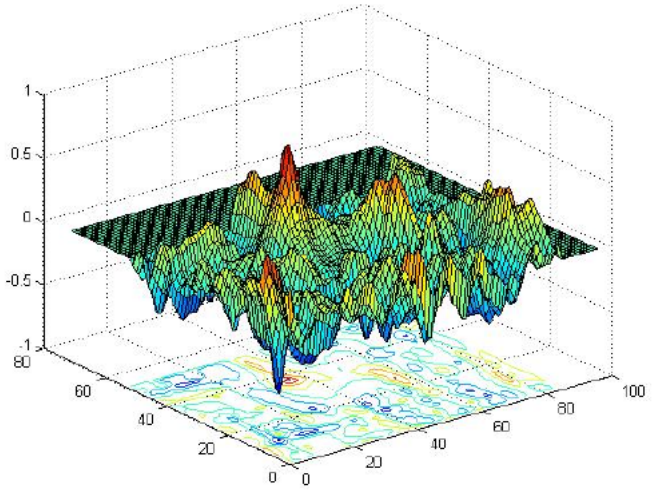




Level 2 Search

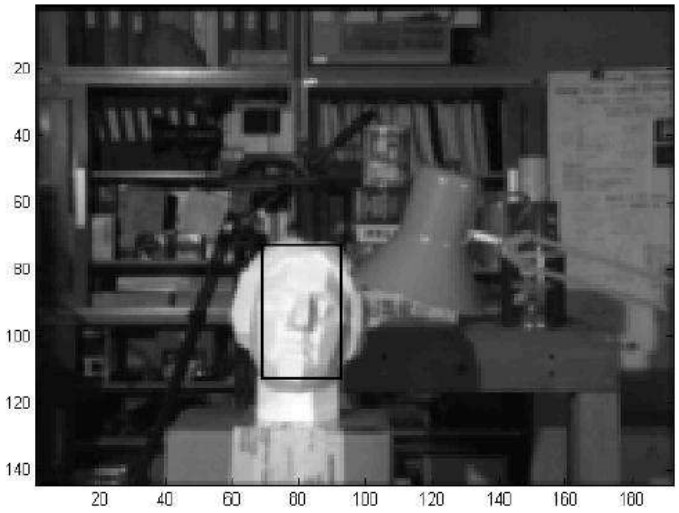
- Subsequent searches are constrained to a neighborhood of only several pixels in the x and y directions

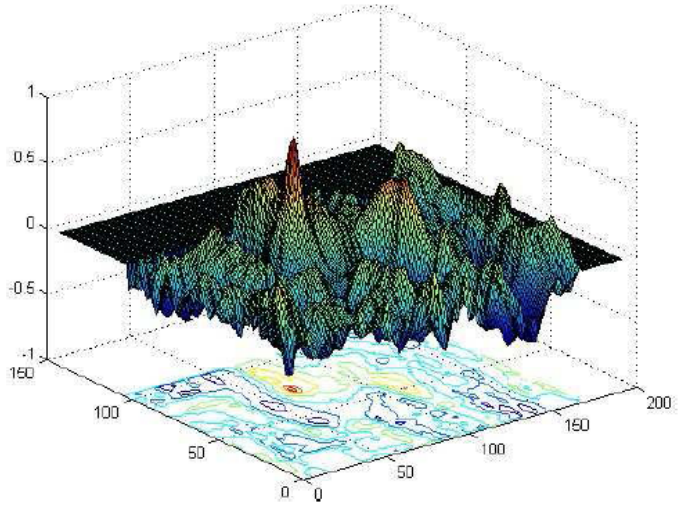




Level 1 Search

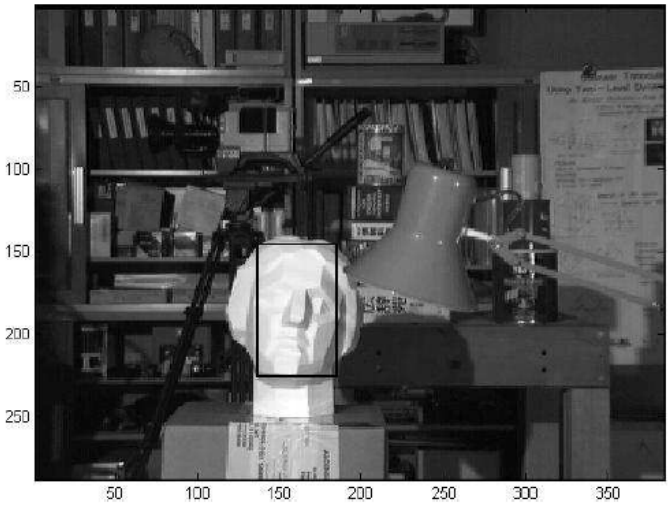
- Subsequent searches are constrained to a neighborhood of only several pixels in the x and y directions





Level 0 Search

- In the end, the total time (in Matlab) was reduced from ≈ 31 seconds to ≈ 0.5 seconds while obtaining the same template match



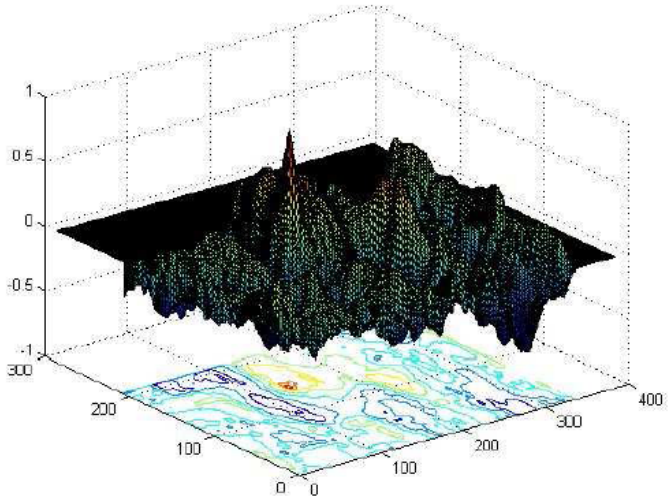
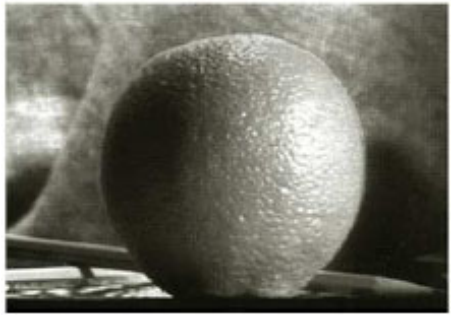


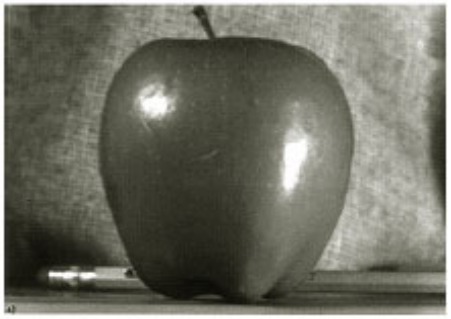
Image Blending and Mosaicing

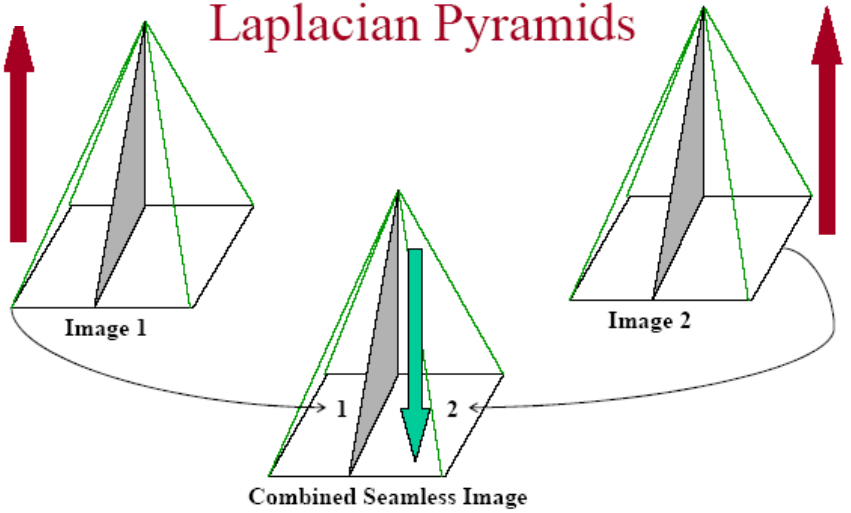
- Given two images A and B
- Construct Laplacian Pyramid L_a and L_b
- Create a third Laplacian Pyramid L_c where for each level l

$$L_c(i,j) = \begin{cases} L_a(i,j) & \text{if } i < width/2 \\ (L_a(i,j) + L_b(i,j))/2 & \text{if } i = width/2 \\ L_b(i,j) & \text{if } i > width/2 \end{cases}$$

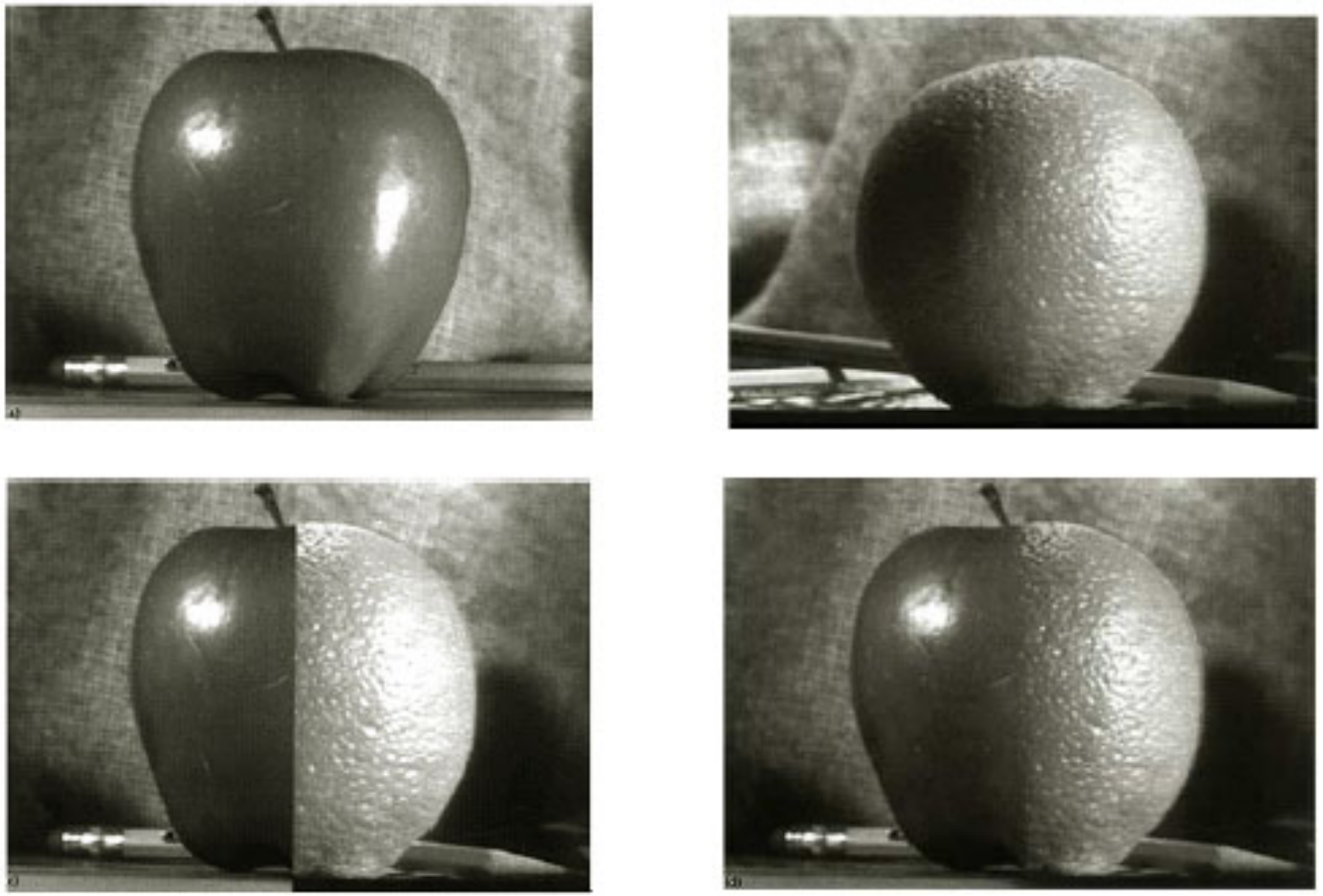


- Sum all levels L_c in to get the blended image





Blending Apples and Oranges



Pyramid blending of Regions

- Given two images A and B, and a mask M
- Construct Laplacian Pyramids L_a and L_b
- Construct a Gaussian Pyramid G_m
- Create a third Laplacian Pyramid L_c where for each level l

$$L_c(i,j) = G_m(i,j)L_a(i,j) + (1 - G_m(i,j))L_b(i,j)$$

- Sum all levels L_c in to get the blended image

Horror Photo

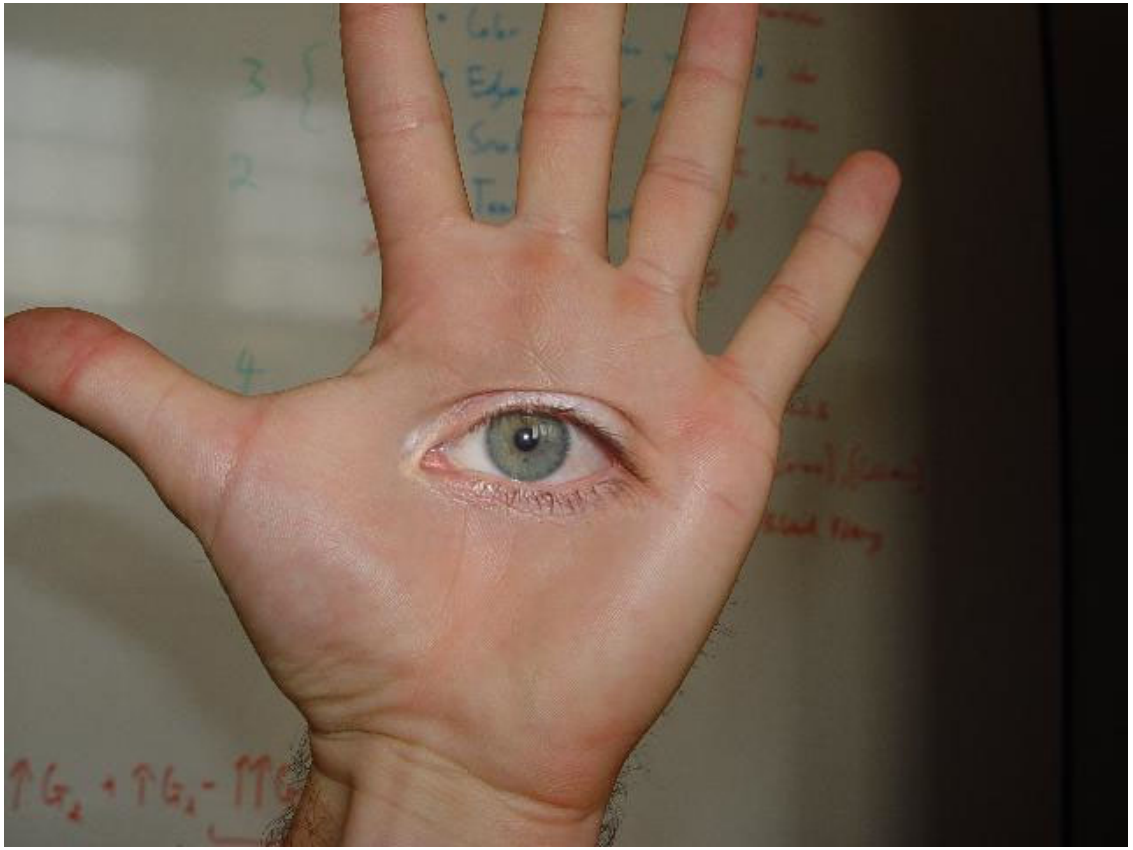
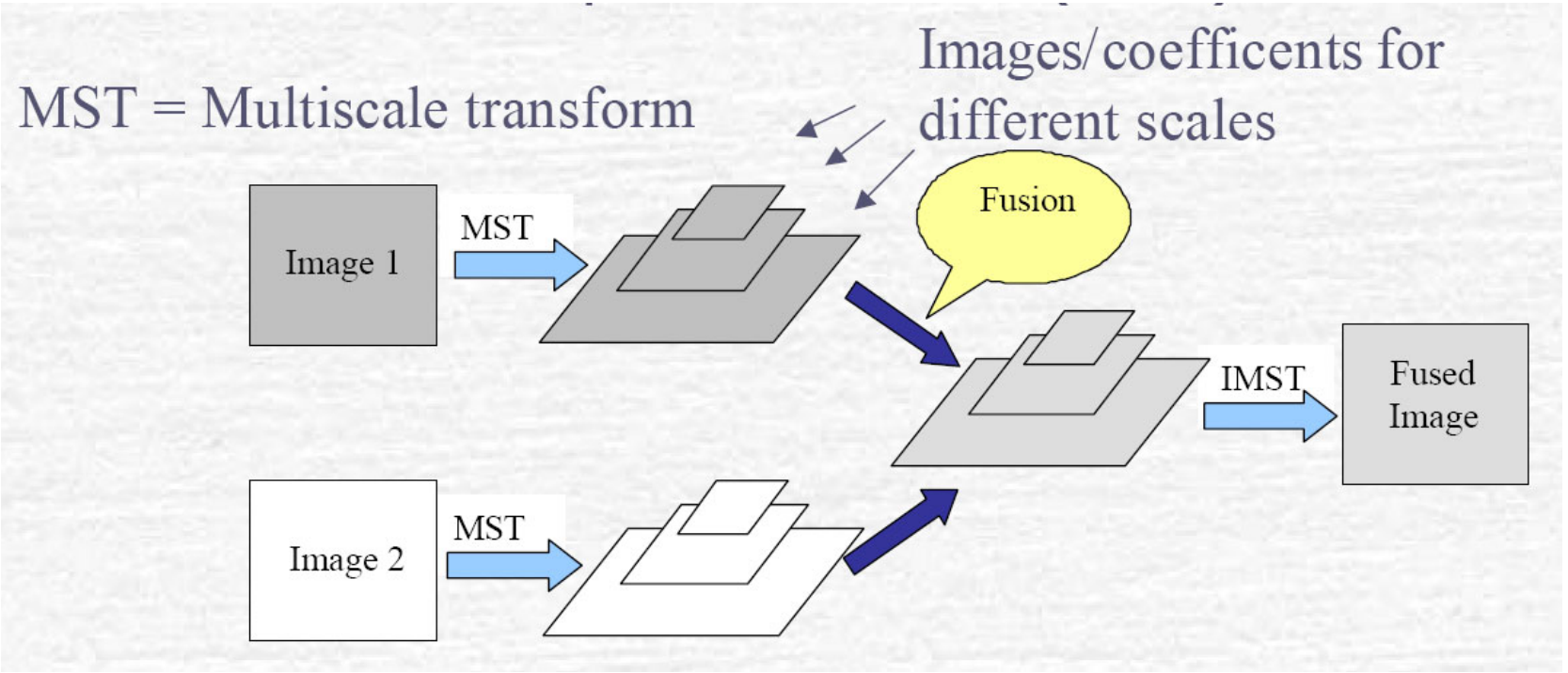
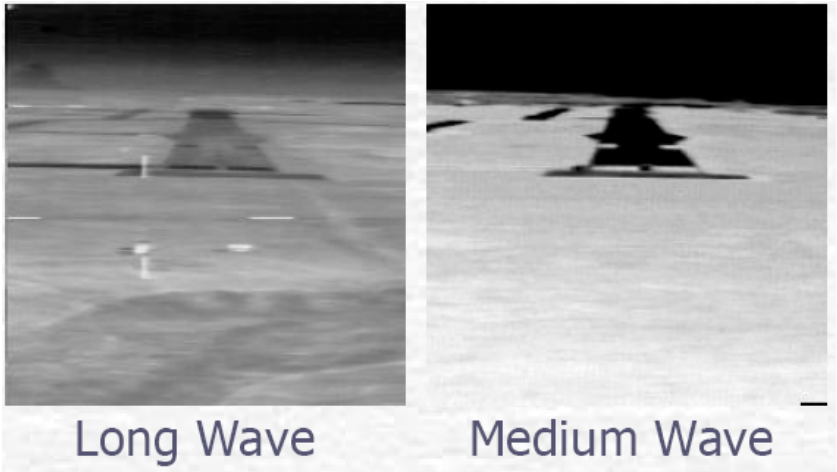


Image Fusion



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

Multi-Sensor Fusion



Matlab Impyramid()