SCALE-SPACE -

Theory and Applications

Scale is embedded in the *task*: do you want only coins or TREASURE?

SCALE-SPACE – Theory and Applications

- Scale-space theory is a framework of multiscale image/signal representation;

- Need to handle multi-scale nature of real-world objects

- Representation of multiple scales simultaneously

- Design of flexible image operators, for tasks such as, feature detection, feature classification, stereo matching, motion descriptors, shape cues and representing image/video content.

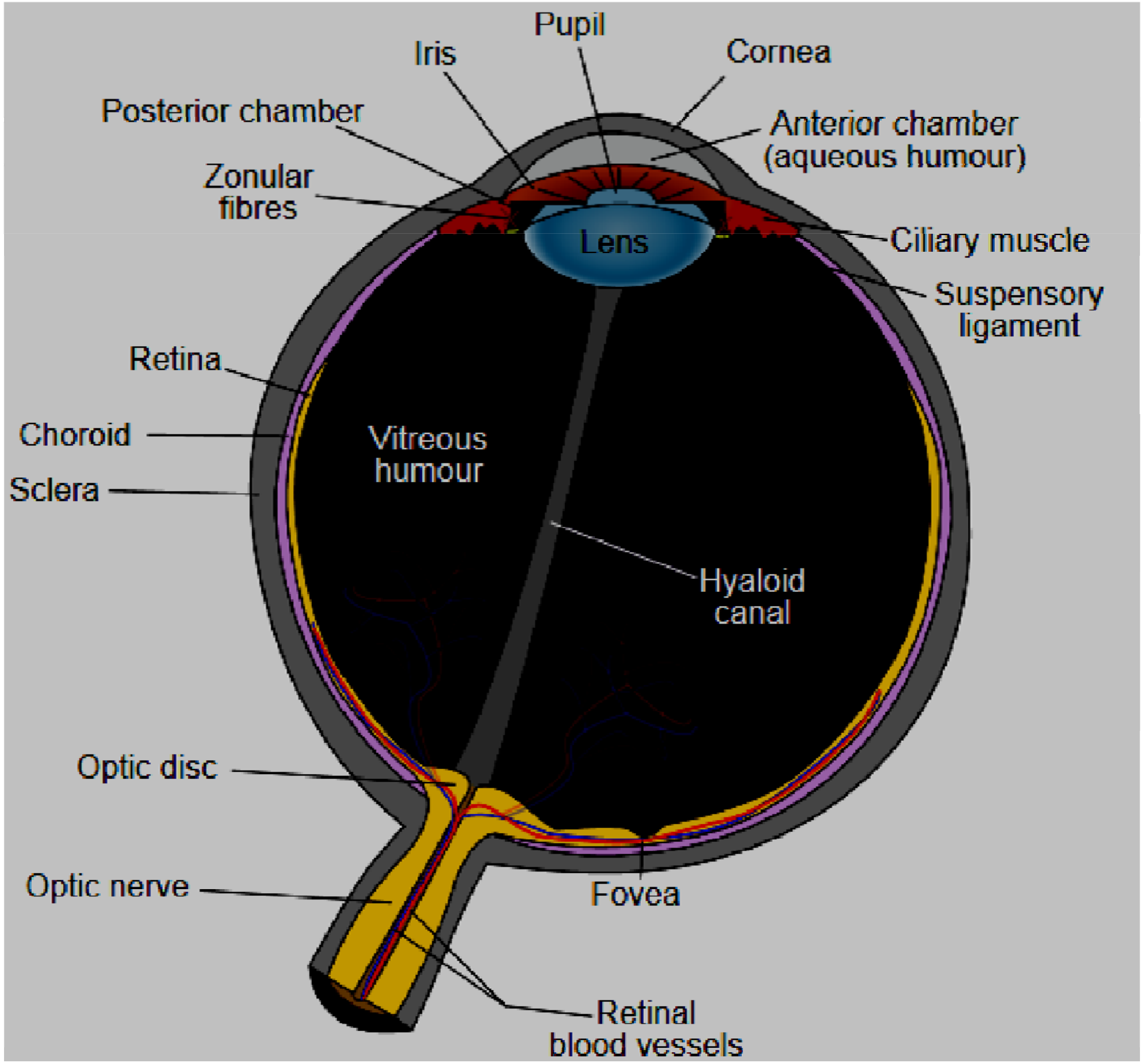
- How make modules of visual processing scale invariant ?

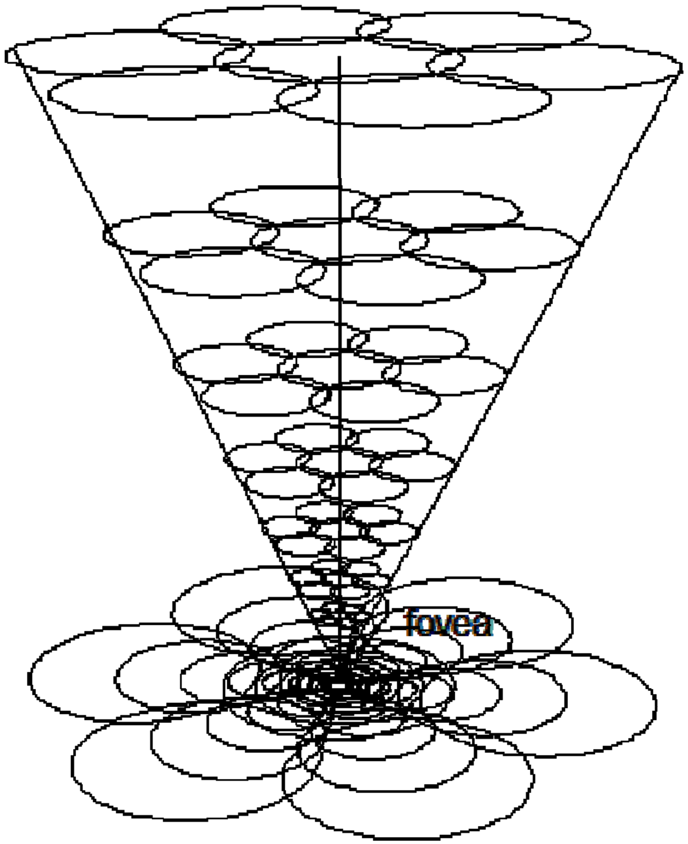
- Motivation comes from the resemblance of close receptive field profiles of the human visual system.

Scale Space in Human Vision

•

•





- Most problems in CV & IP, are faced with the question:

- What operators to use ?

- Where to apply them ?

- How large (scale or range of scales) should they be ?

- How to relate (interpret) to the actual structure in the scene?

In the absence of prior information – use empirical methods; represent data at multiple scales.

Scale-space method attempts to represent data at all scales simultaneously.

SCALE-SPACE – Theory and Applications

- Earliest stage of visual processing [Hubel and Wiesel] suggests that, the response of cells in primary visual cortex have multi-channel, multi-resolution property, orientation selectivity and response to primitive geometrical shapes structures.

- Scale-space theory specifies that convolution by the Gaussian kernel and its derivatives provide a canonical class of image operators with unique properties.

- In presence of noise and other artifacts, computing image derivatives is an ill-posed problem.

- Gaussian derivatives provide a convenient way of defining derivatives in scale-space in a well-posed manner

- Thus convolve the image with Gaussian derivative kernels.

Practical Implementation

- Convolve the image with a Gaussian Kernel

$$G(\sigma) = \frac{1}{(2\pi\sigma^2)} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

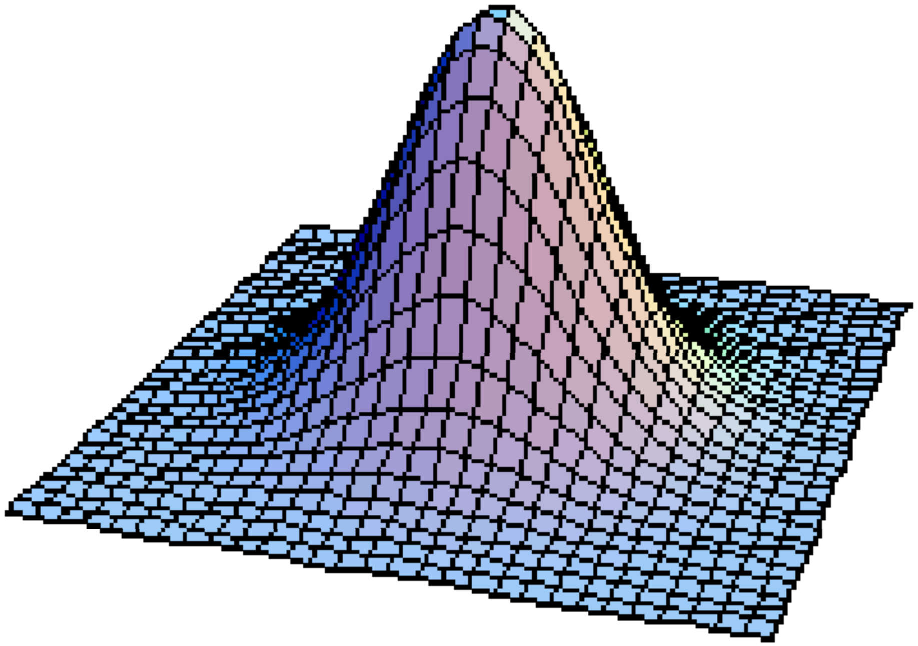


Image at increasing scales, obtained by Gaussian Convolution

$$L(\cdot,\cdot;t) = g(\cdot,\cdot;t) * f(\cdot,\cdot),$$

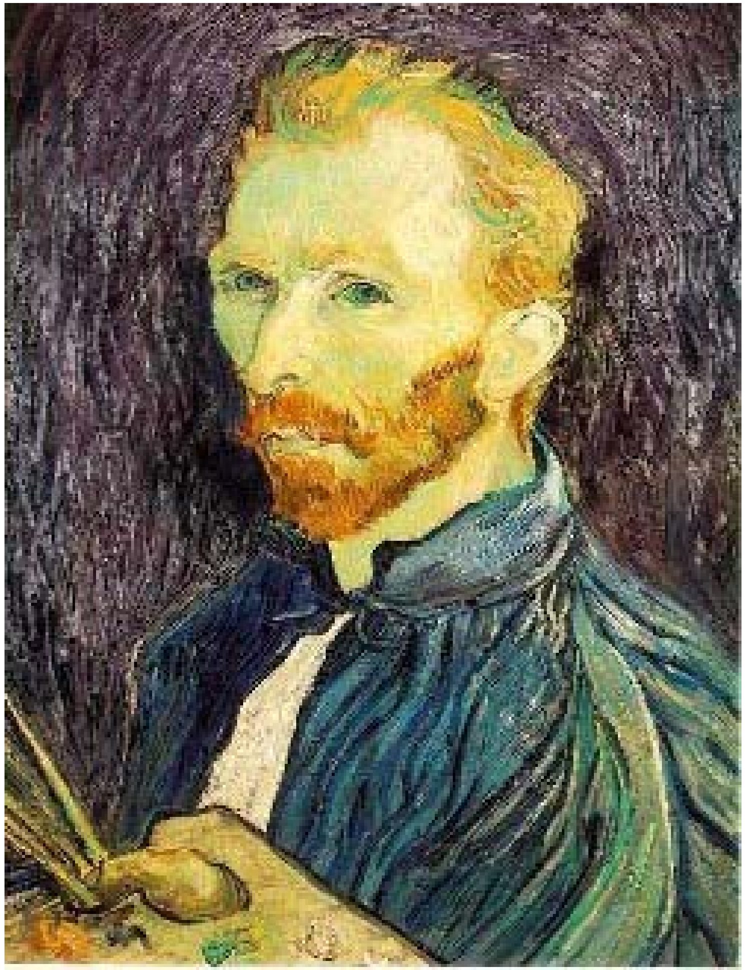
The scale-space family can be defined as the solution of the diffusion equation (for example, in terms of the heat equation): with initial condition, L(x, y; 0) = f(x, y).

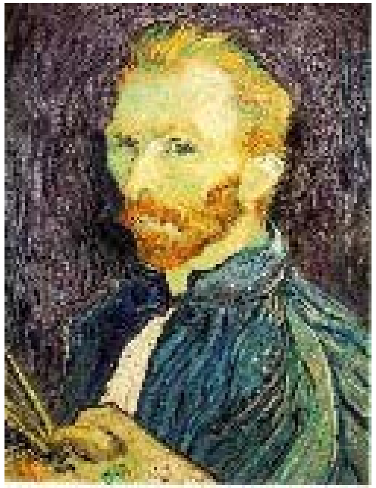
$$\partial_t L = \frac{1}{2}\nabla^2 L,$$

Why multi-scale? Why should you blur?

- **Computational efficiency**
- **Coarse-to-fine**
- **Extracting hierarchical structure**
- **First principles of physics of observations**
- **Visual system is multi-scale**

Image sub-sampling





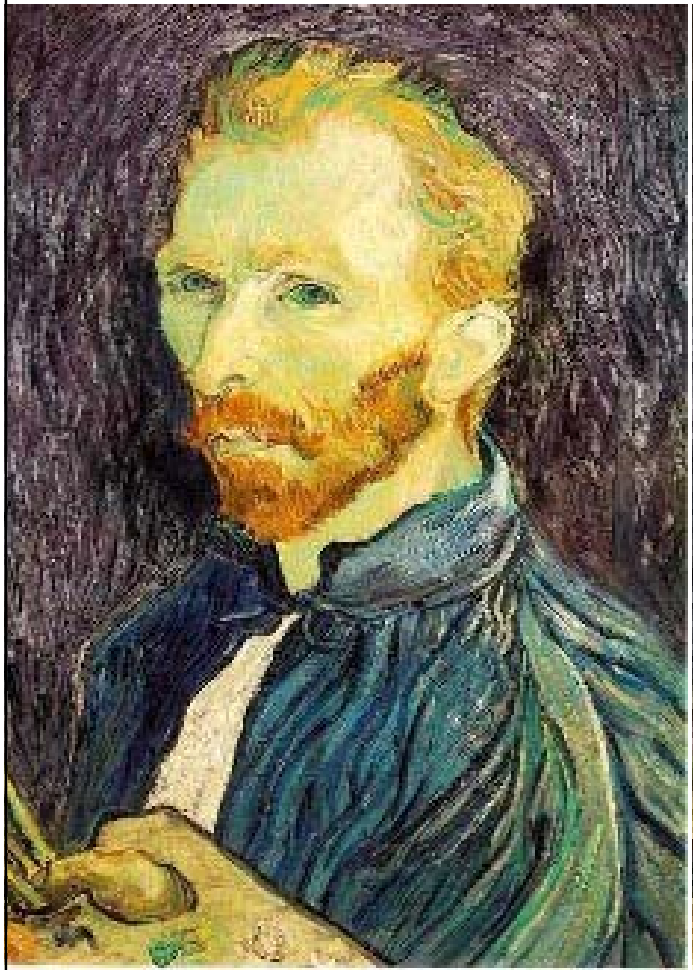
1/4

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)

Why does this look so bad?

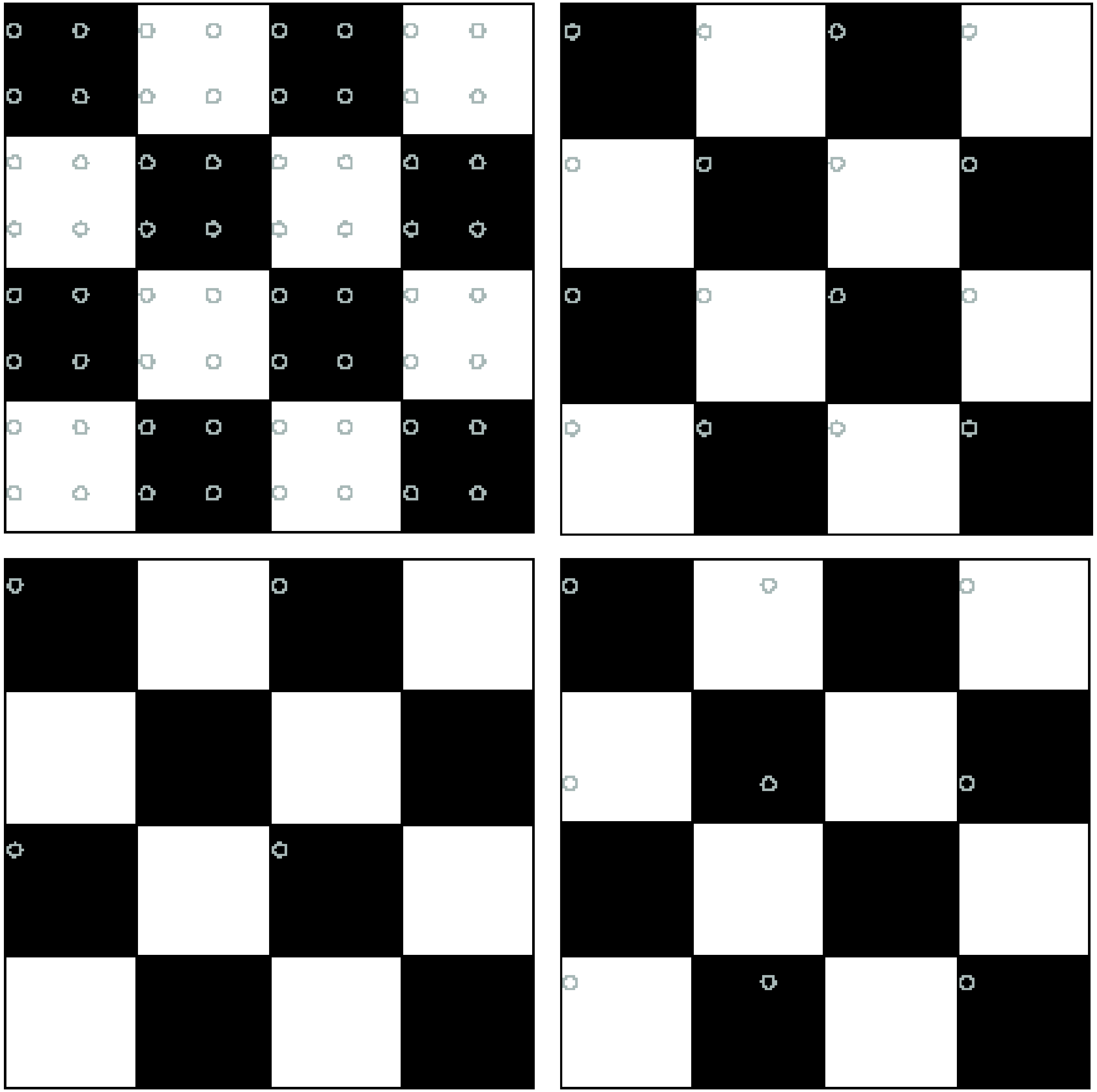
Aliasing

Occurs when you shrink an image by taking every nth (n>1) pixel.

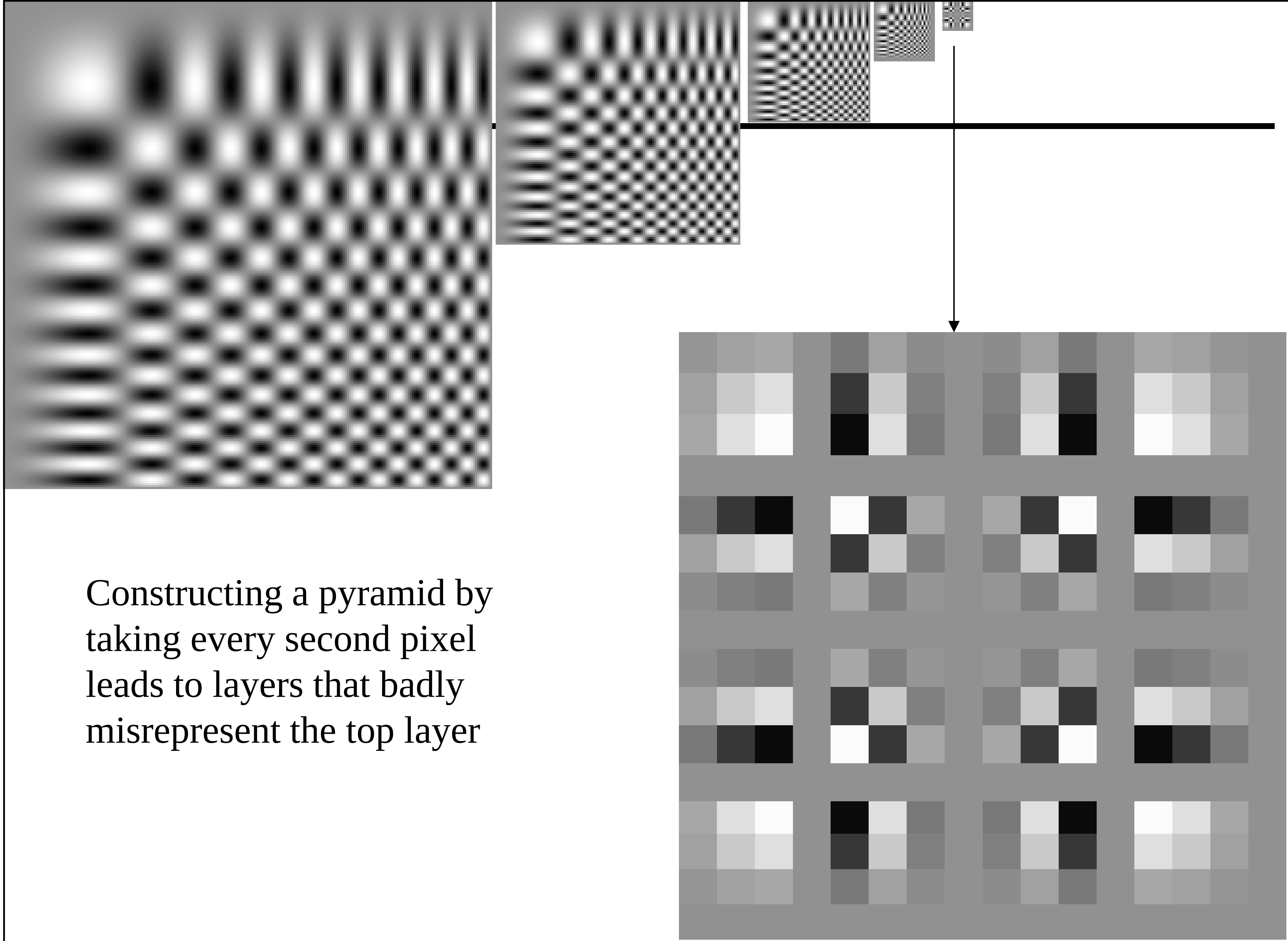
If we do, characteristic errors appear

- **Typically, small phenomena look bigger; fast phenomena can look slower**

- **Common phenomenon**
  - **Wagon wheels rolling the wrong way in movies**
  - **Checkerboards misrepresented in ray tracing**
  - **Striped shirts look funny on colour television**



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.



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

What does blurring take away?

Original

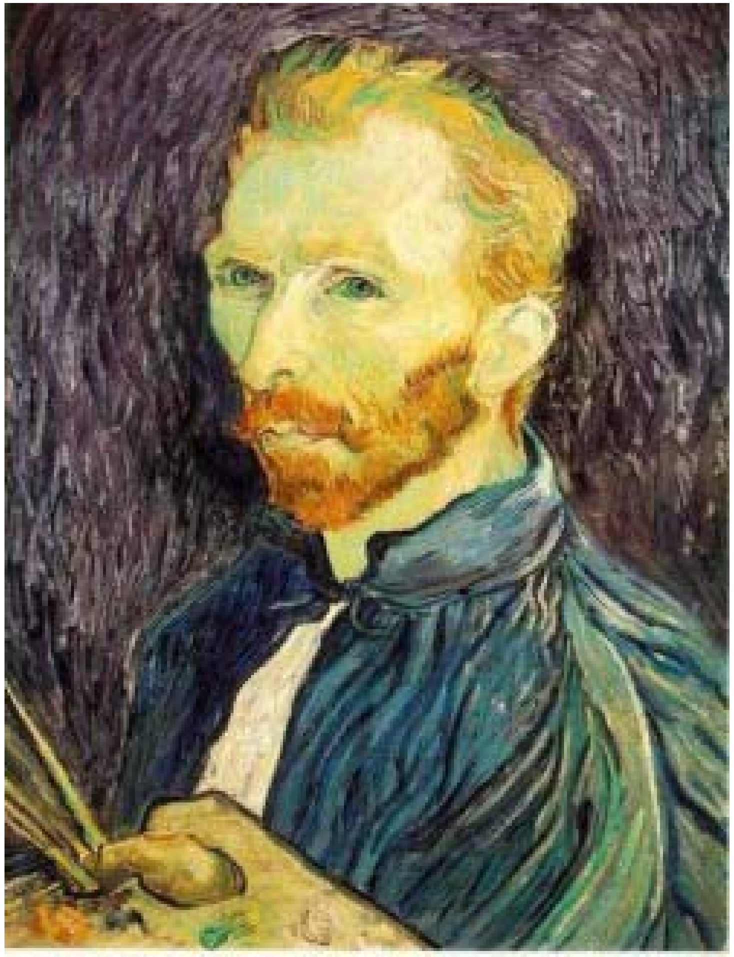
What does blurring take away?

smoothed (5x5 Gaussian)

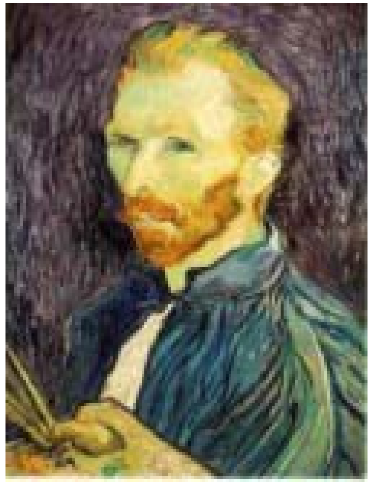
High-Pass filter

smoothed MINUS original

Gaussian pre-filtering



Gaussian 1/2



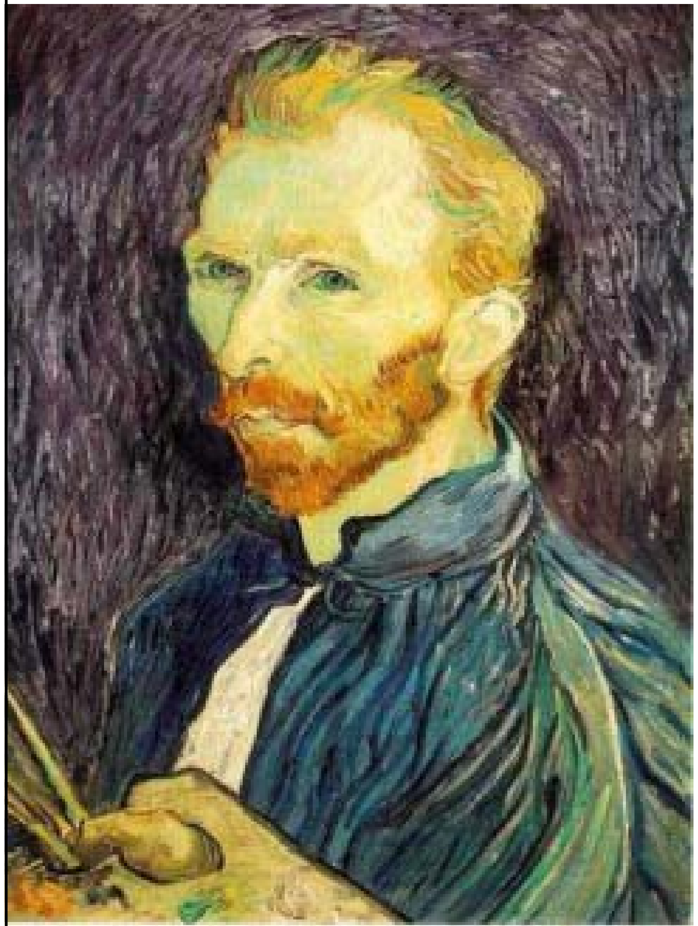
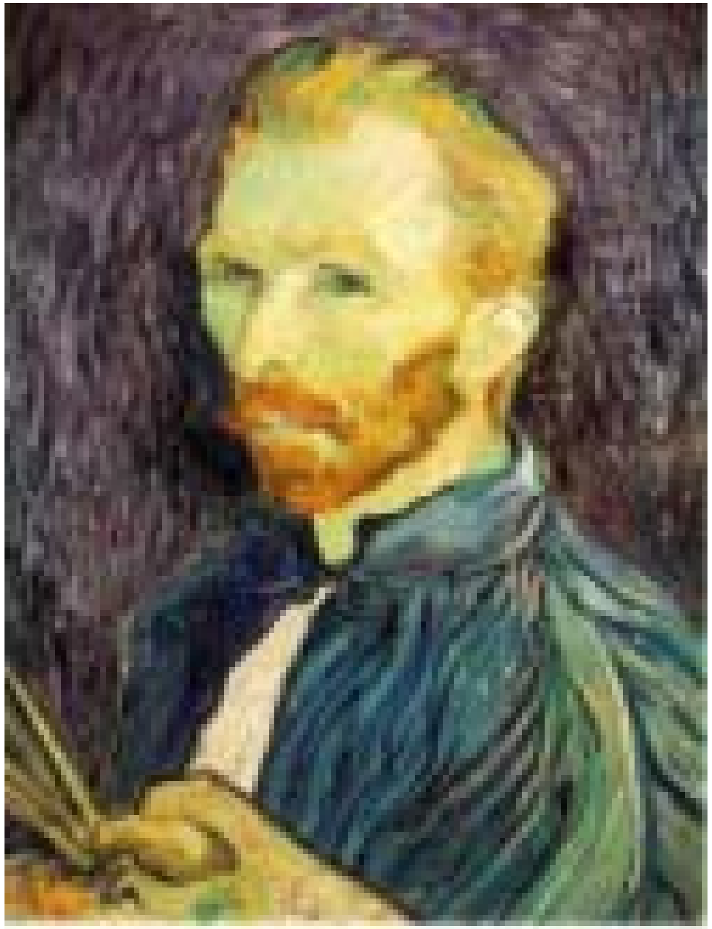
G 1/4

G 1/8

Solution: filter the image, *then* subsample

- **Filter size should double for each ½ size reduction. Why?**

Subsampling with Gaussian pre-filtering

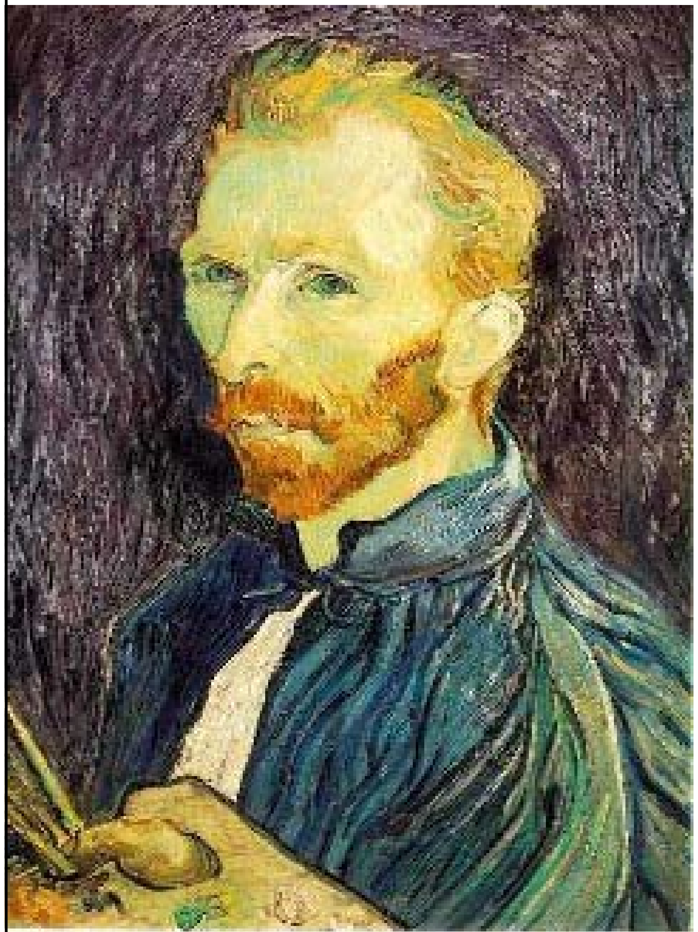




Gaussian 1/2 G 1/4 G 1/8

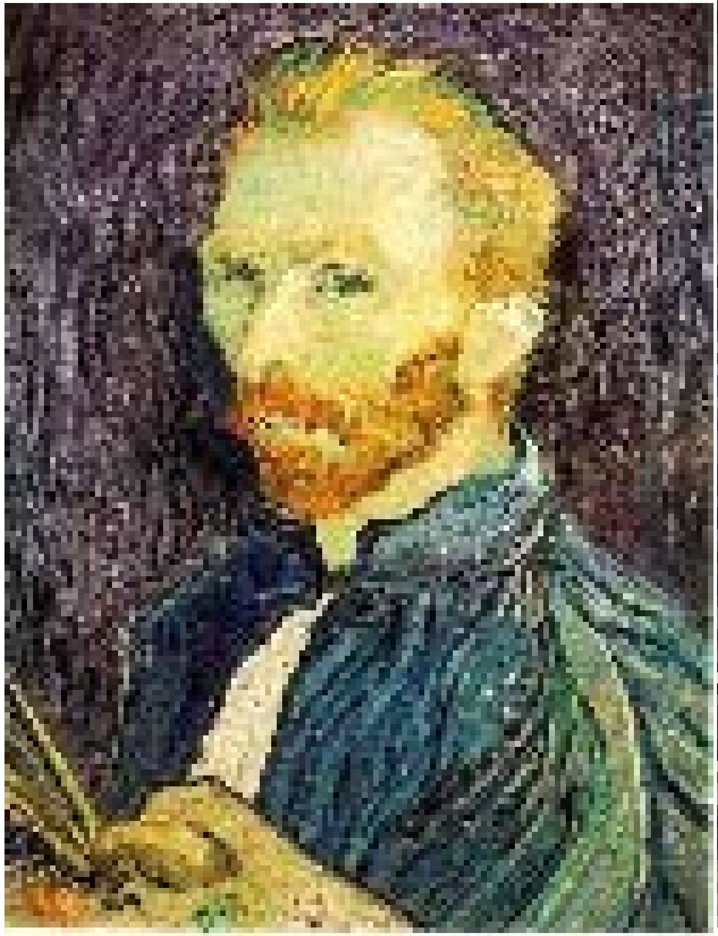
Solution: filter the image, *then* subsample

- **Filter size should double for each ½ size reduction. Why?**
- **How can we speed this up?**

Compare with...

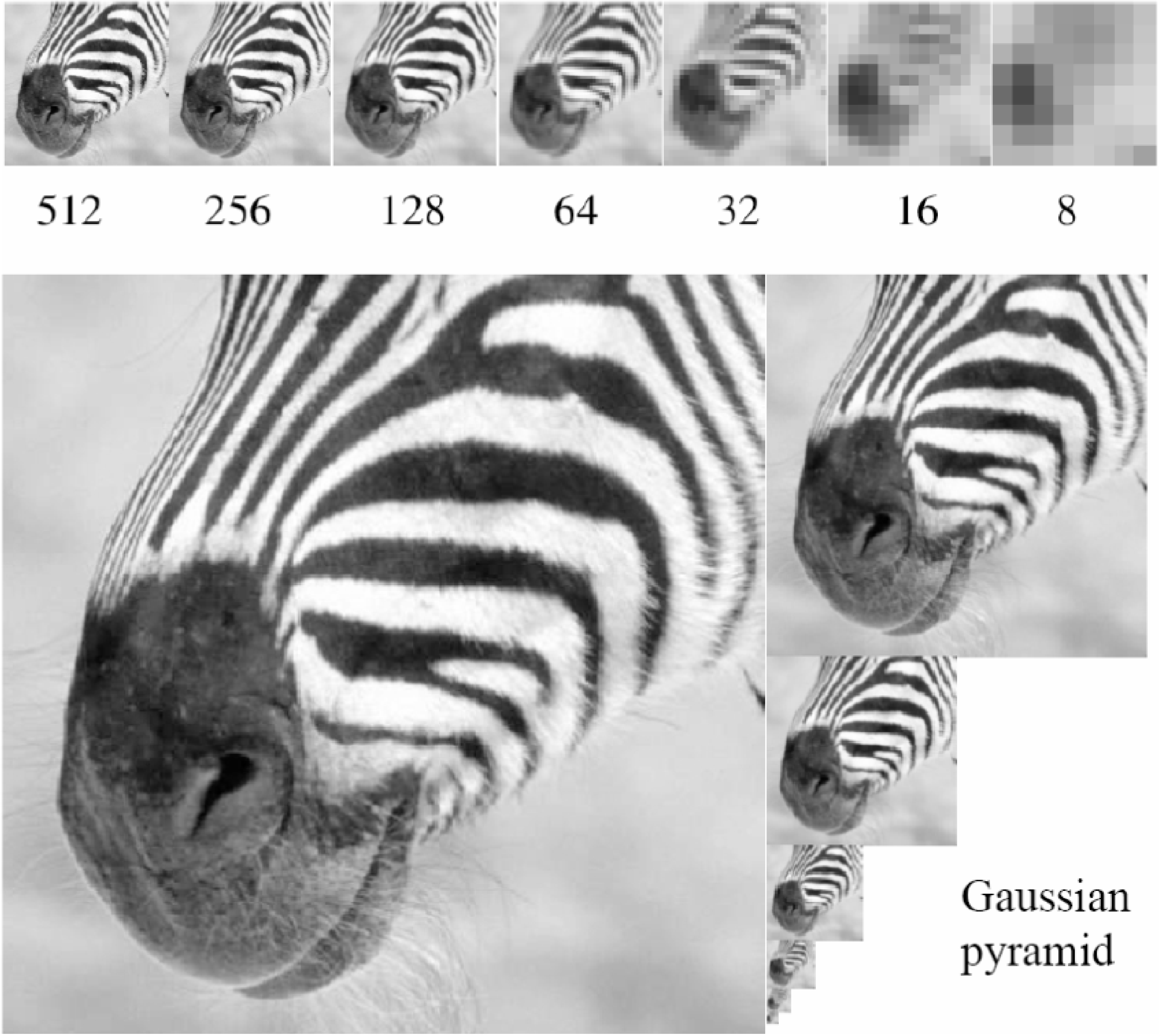


1/2



1/4 (2x zoom)

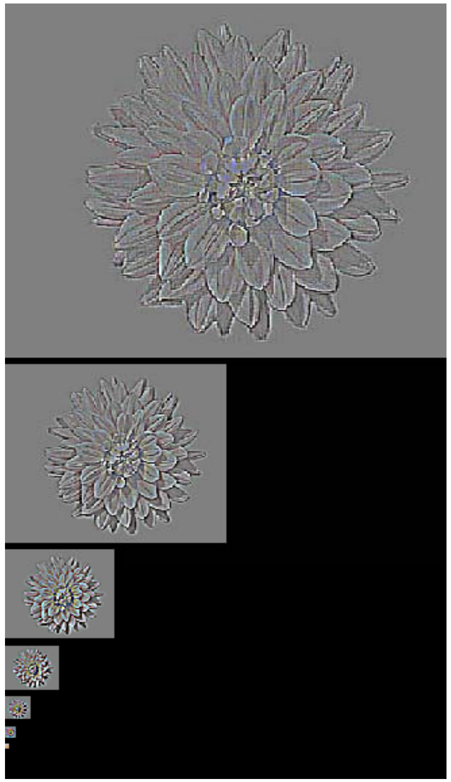
1/8 (4x zoom)



Gaussian pyramid

Image Pyramids

- Gaussian and Laplacian



The Gaussian Pyramid

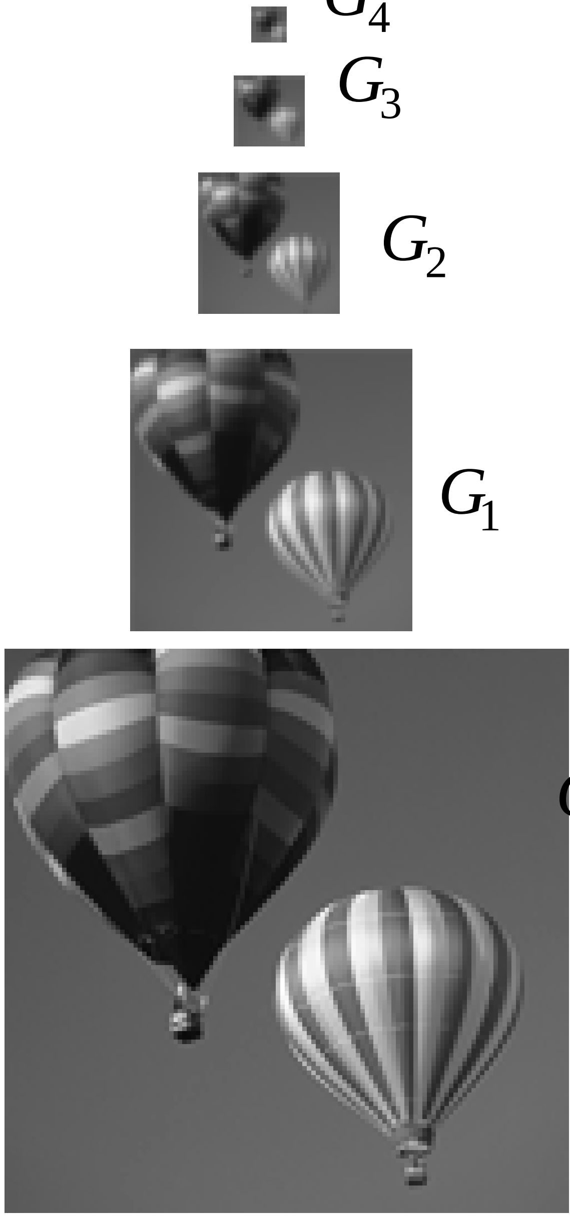
Low resolution

G_4

G_3

G_2

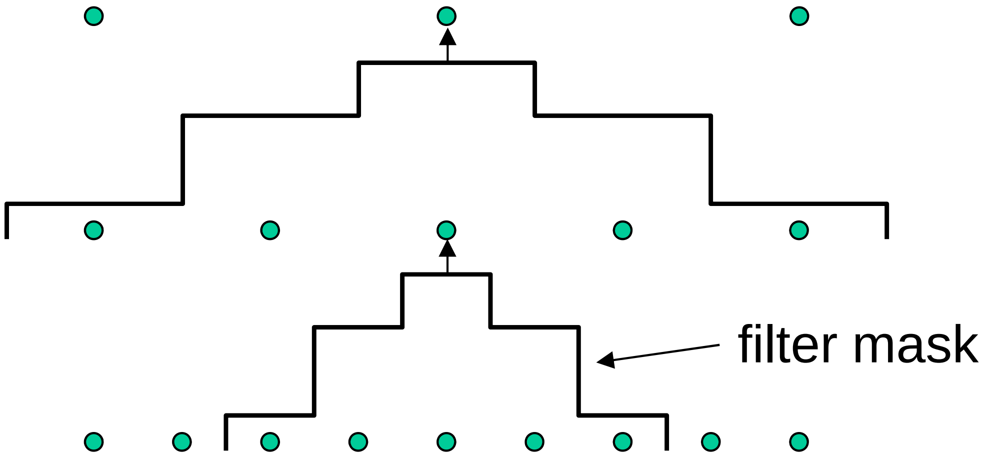
G_1



$G_0 =$ **Image**

High resolution

Gaussian pyramid construction



Repeat

- Filter
- Subsample

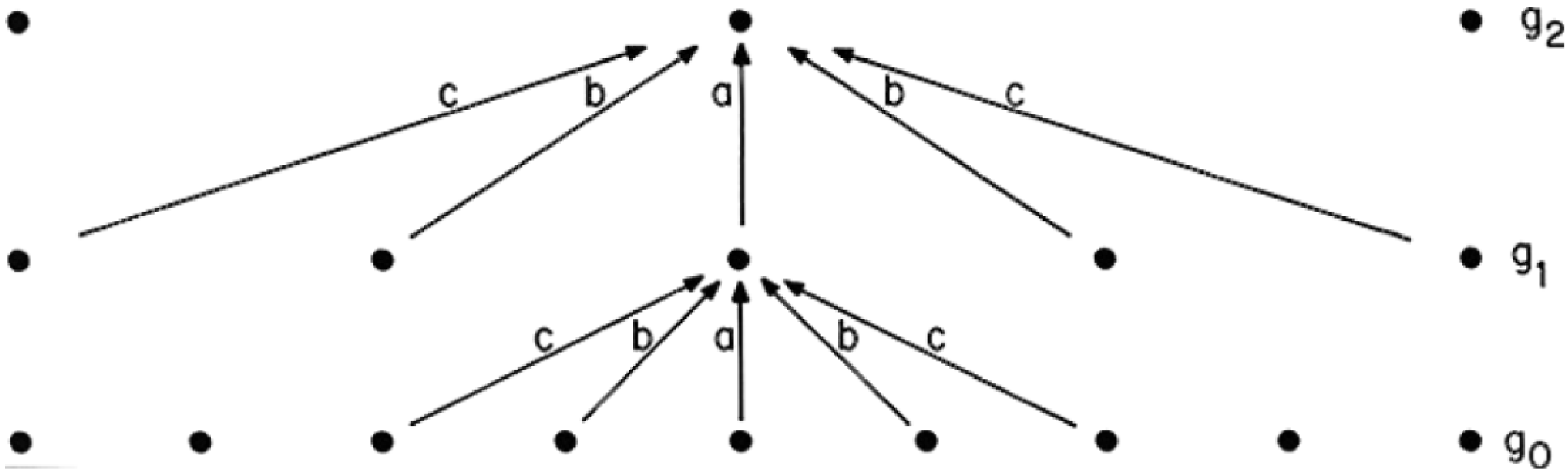
[0.05, 0.25, 0.4, 0.25, 0.05]

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!

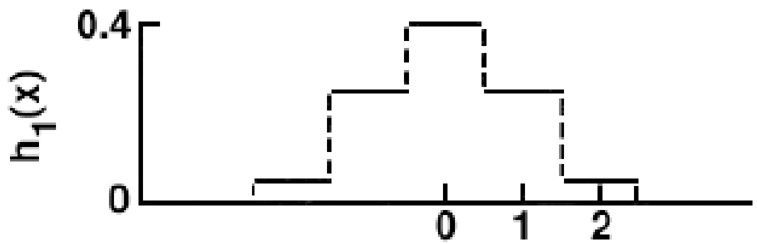
GAUSSIAN PYRAMID



g_0 = IMAGE

g_L = REDUCE [g_{L-1}]

$$g_l(i, j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m, n) g_{l-1}(2i + m, 2i + n).$$

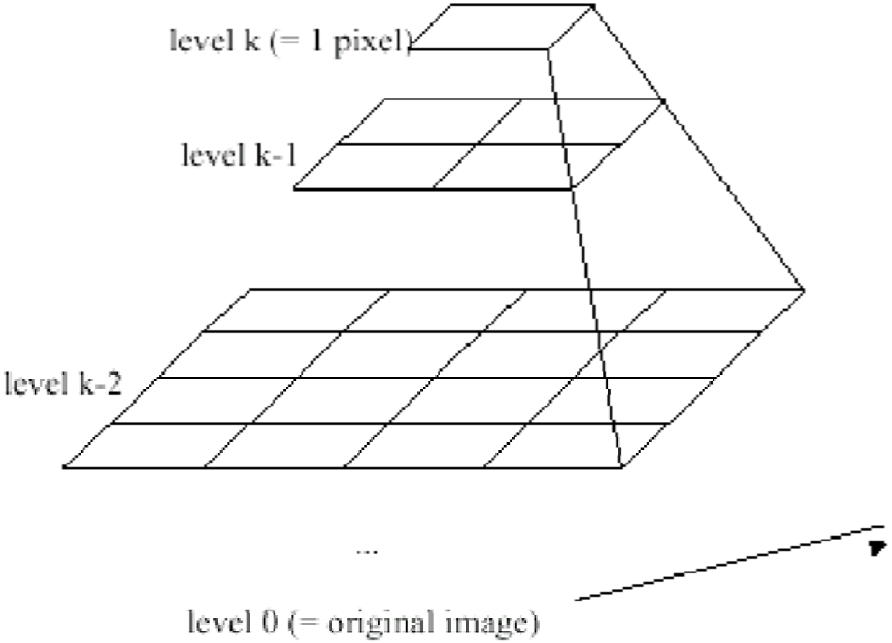


GAUSSIAN PYRAMID

0 1 2 3 4 5

Image Pyramids

Idea: Represent NxN image as a "pyramid" of 1×1, 2×2, 4×4, …, $2^k \times 2^k$ images (assuming $N = 2^k$)



Known as a **Gaussian Pyramid**

DECIMATION AND INTERPOLATION

$$y(n) = x(n) * h(n) = \sum_k h(k)x(n-k);$$

I/P signal : x(n);
Output : y(n);
Impulse response of LPF : h(n).

$$z(n) = y(2n);$$

$$z(n) = \sum_k h(k)x(2n-k);$$

$$x(n) \longrightarrow \boxed{h(n)} \xrightarrow{y(n)} \boxed{\downarrow 2} \longrightarrow z(n)$$

$$y(n) = \begin{cases} x(n/2) & : n \quad even \\ 0 & : n \quad odd \end{cases}$$

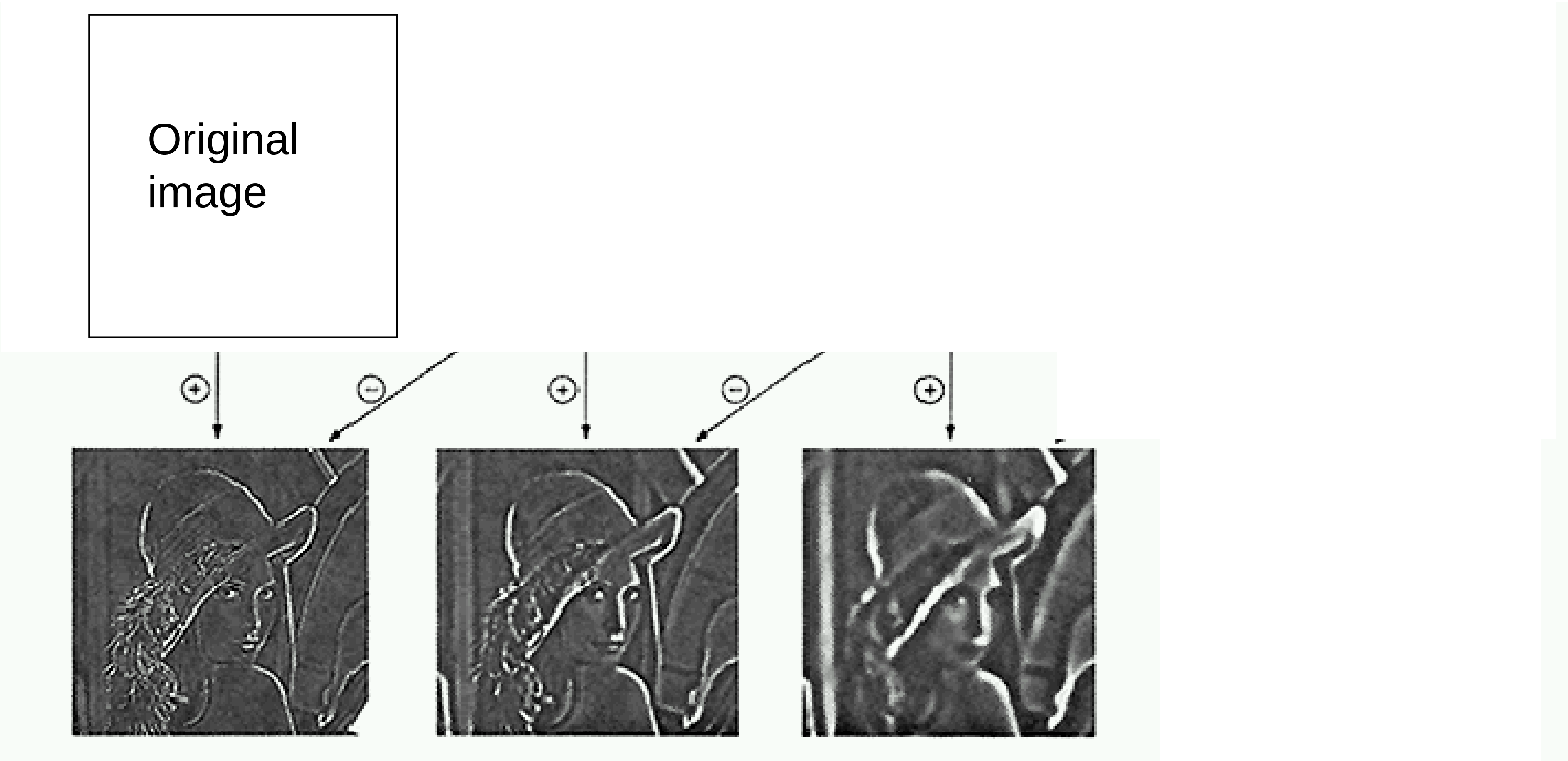
$$z(n) = \sum_k h(k)x(n-2k);$$

$$x(n) \longrightarrow \boxed{\uparrow 2} \xrightarrow{y(n)} \boxed{h(n)} \longrightarrow z(n)$$

Band-pass filtering

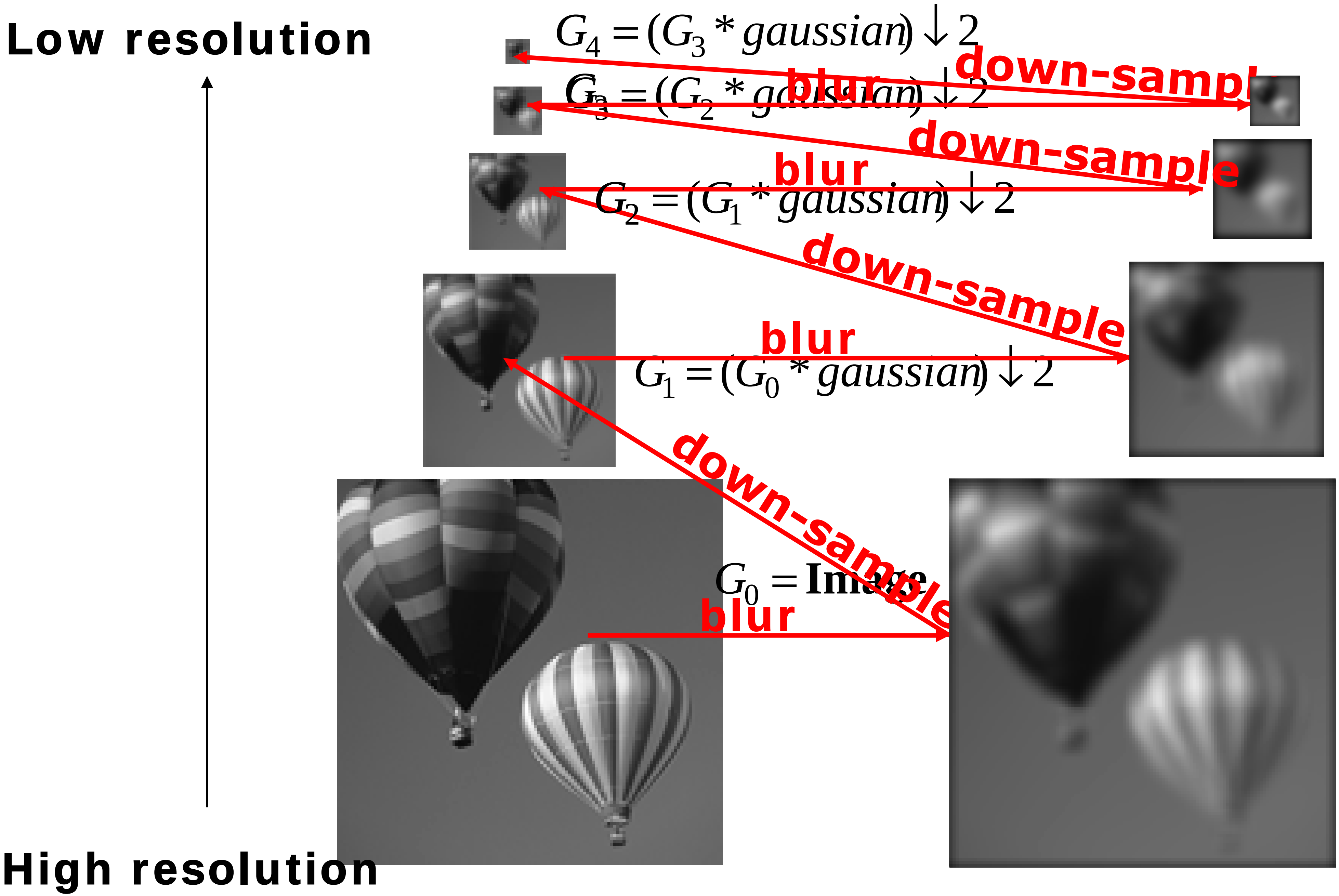
Gaussian Pyramid (low-pass images)

Laplacian Pyramid



How can we reconstruct (collapse) this pyramid into the original image?

The Gaussian Pyramid



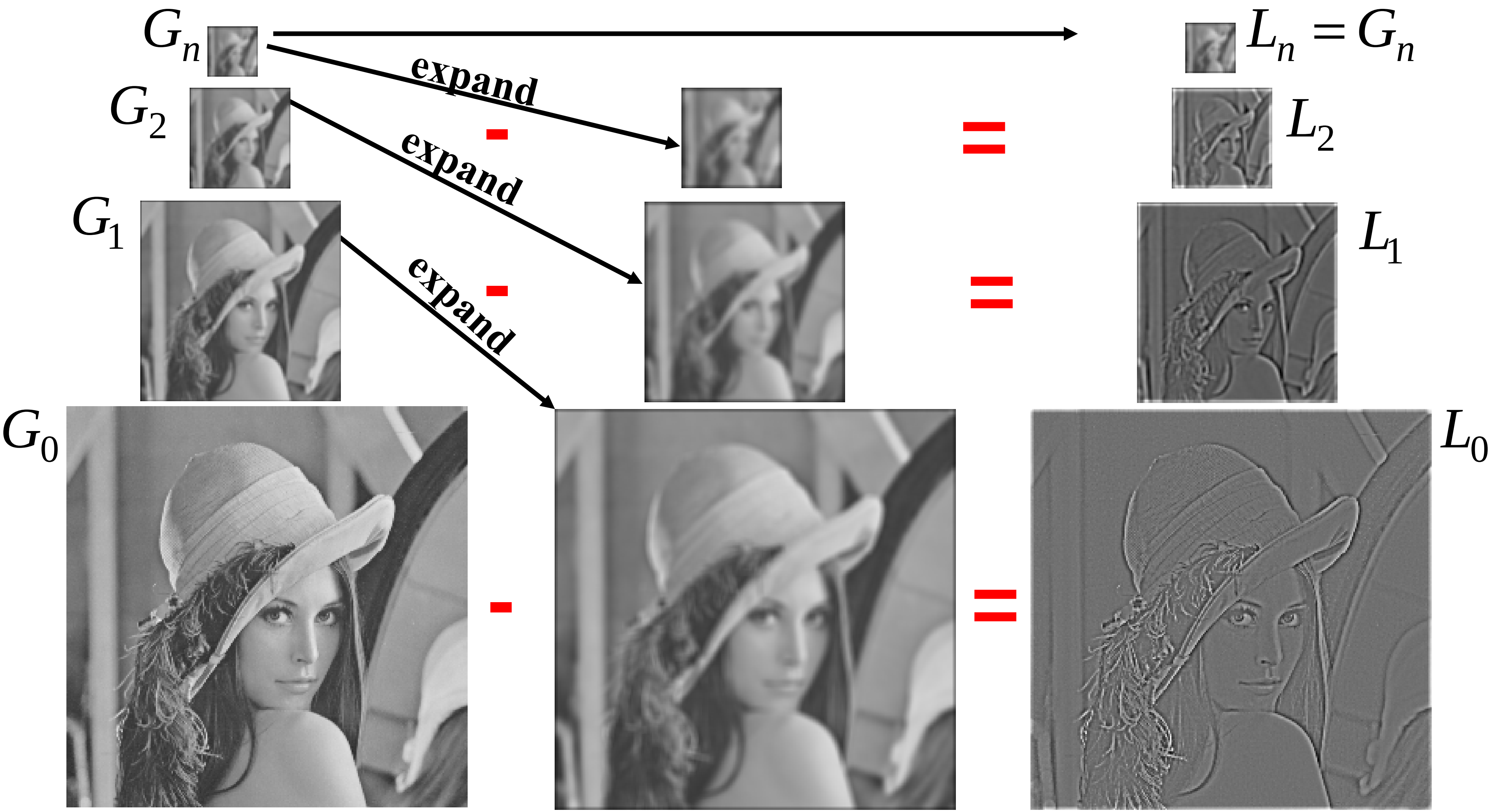
The Laplacian Pyramid

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

Gaussian Pyramid

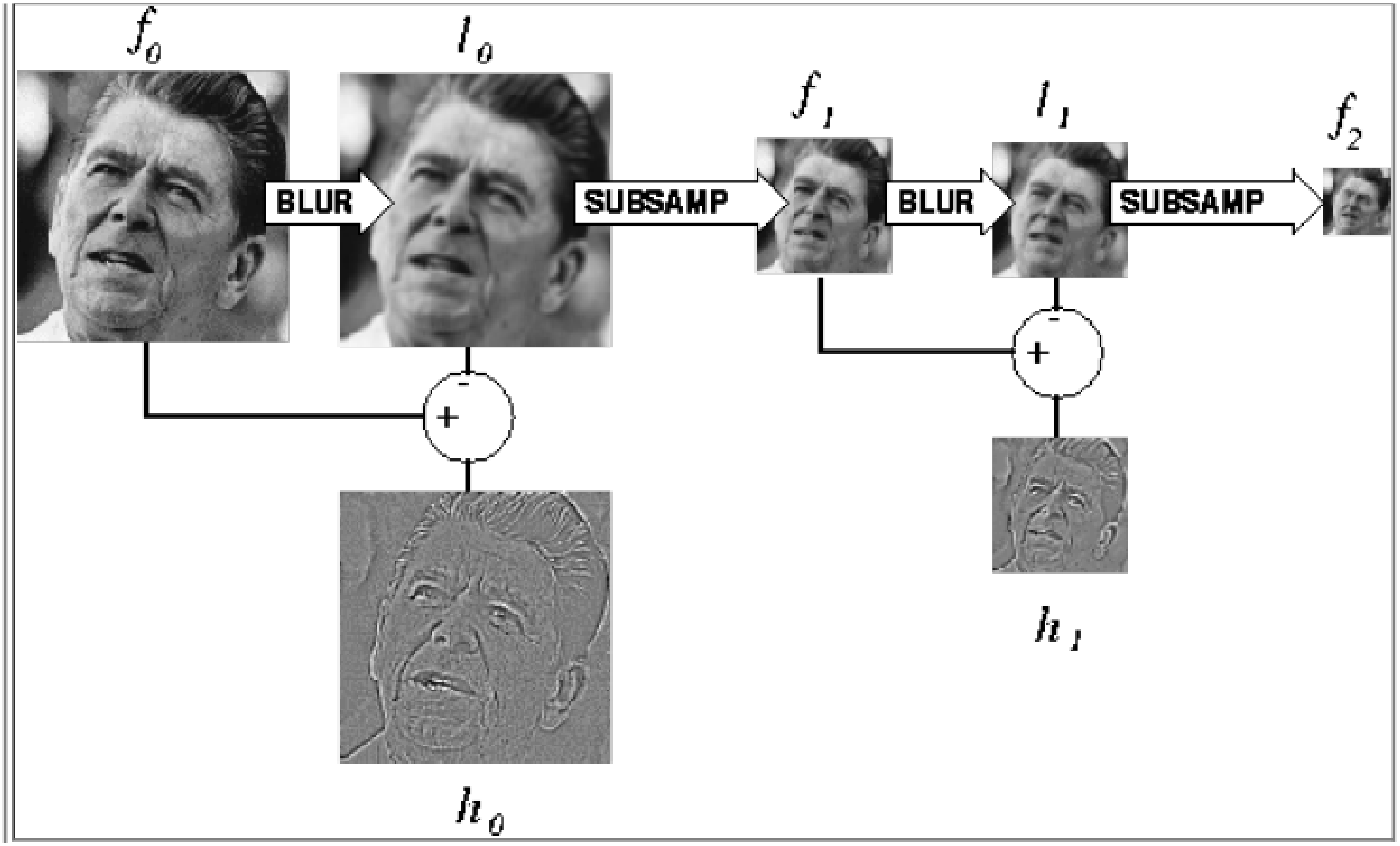
$$G_i = L_i + \text{expand}(G_{i+1})$$

Laplacian Pyramid



Laplacian Pyramid

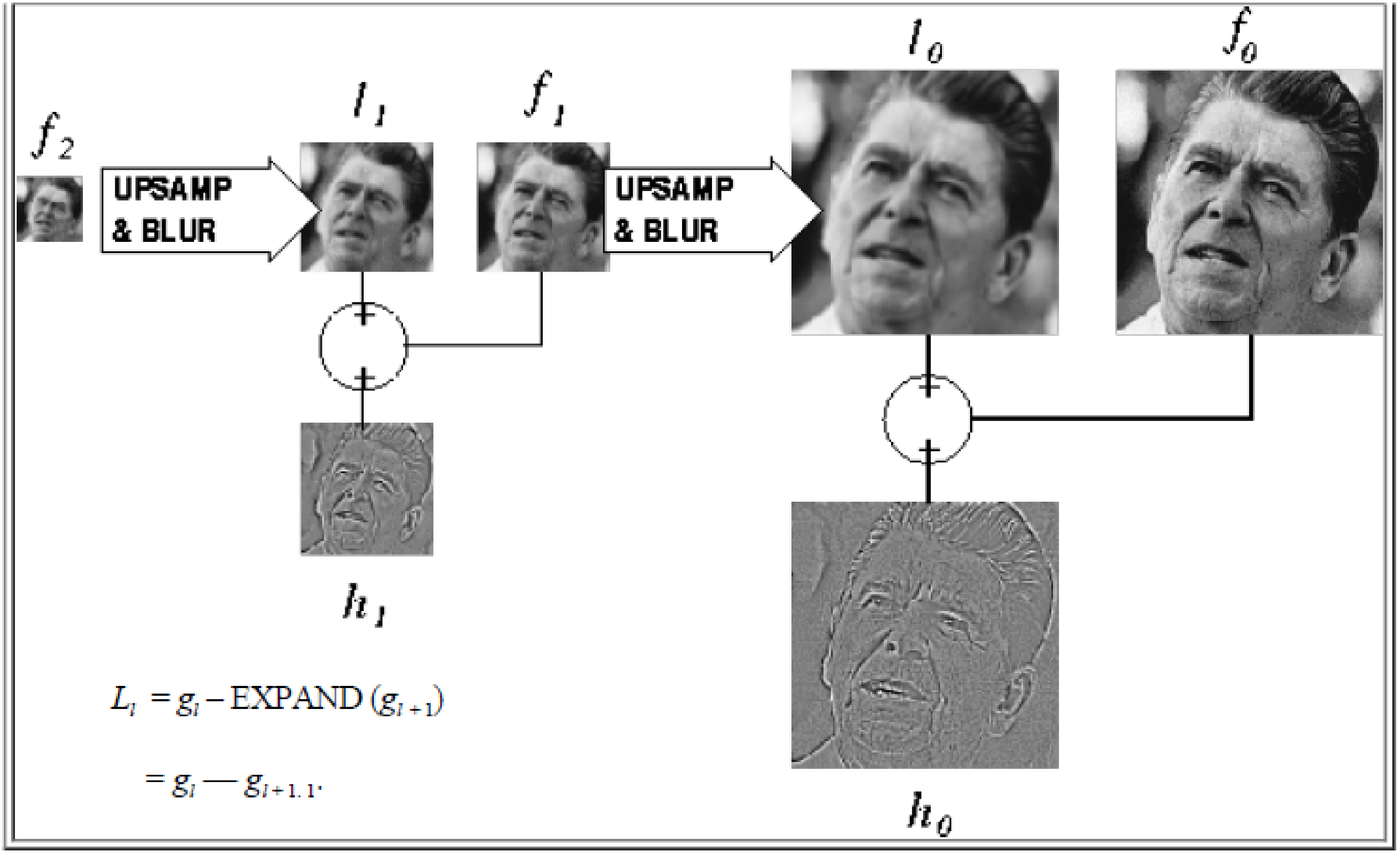
Gaussian Pyramid



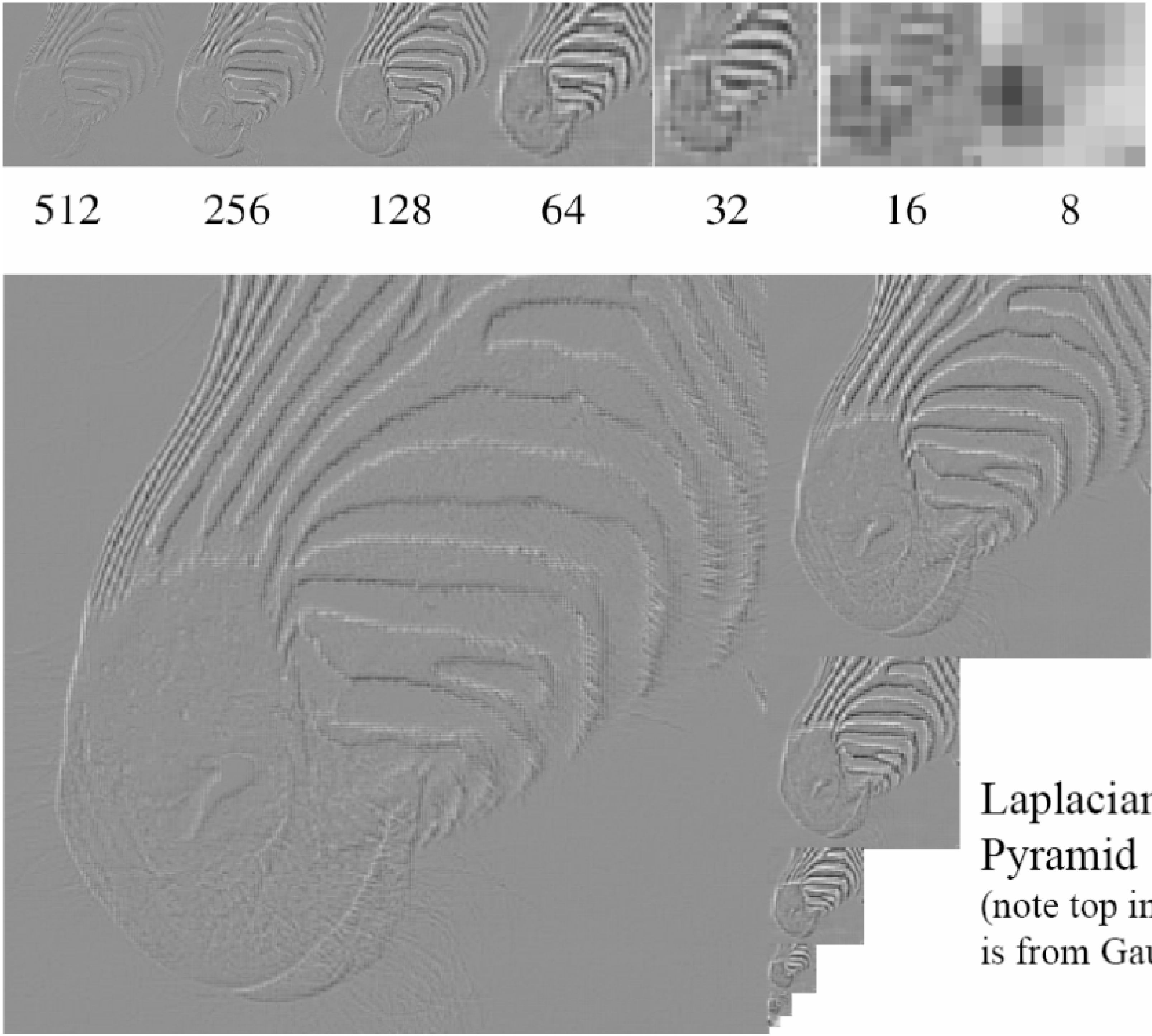
- **Laplacian Pyramid decomposition**
  - Created from Gaussian pyramid by subtraction

Laplacian Pyramid

Gaussian Pyramid

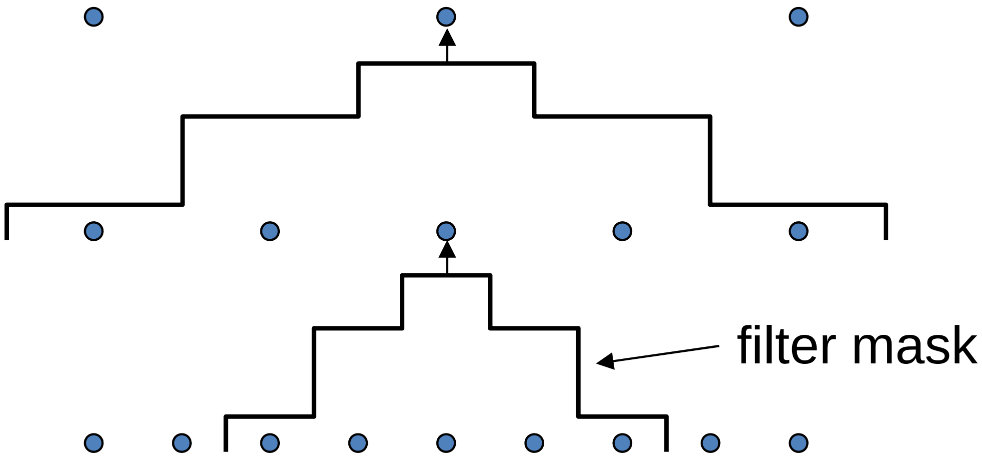


- **Laplacian Pyramid decomposition**
  - **Created from Gaussian pyramid by subtraction**



Laplacian
Pyramid
(note top image
is from Gaussian)

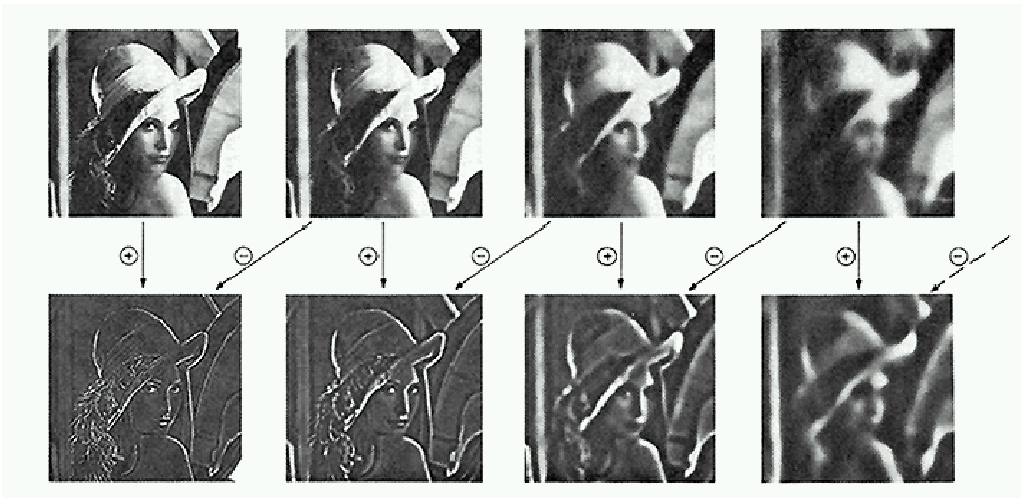
Pyramid Creation

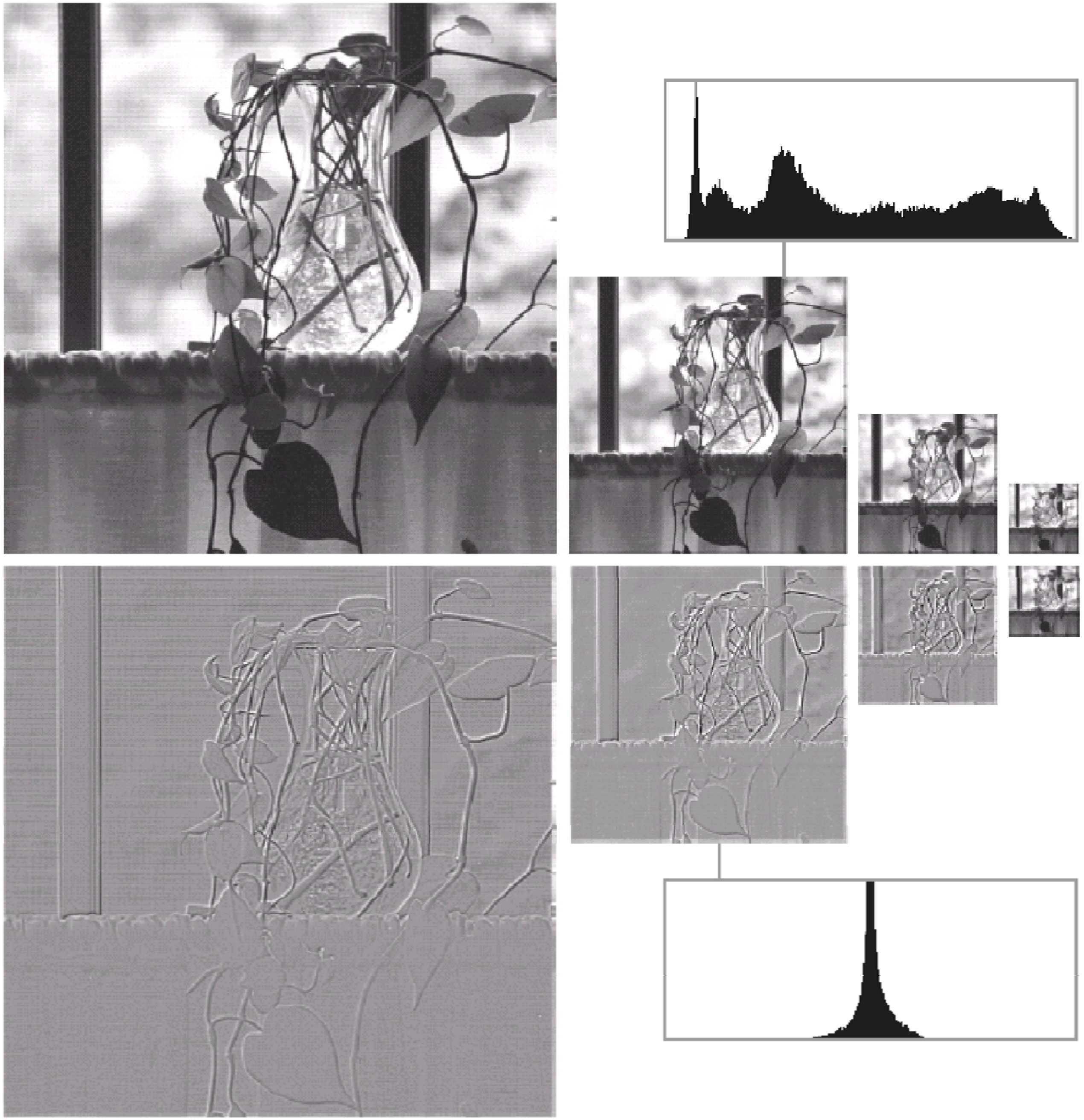


"Gaussian" Pyramid

"Laplacian" Pyramid

- Created from Gaussian pyramid by subtraction $L_l = G_l - \text{expand}(G_{l+1})$

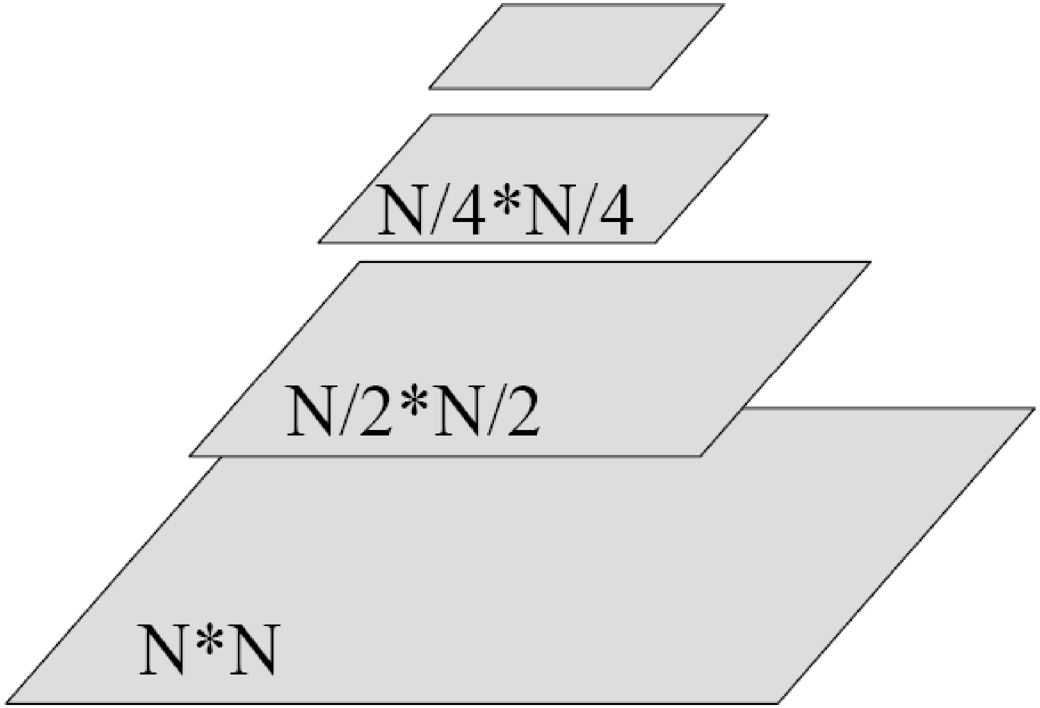




a
b

FIGURE 7.3 Two image pyramids and their statistics: (a) a Gaussian (approximation) pyramid and (b) a Laplacian (prediction residual) pyramid.

Space Required for Pyramids



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

Image pyramids

- At each level we have an approximation image and a residual image.
- The original image (which is at the base of pyramid) and its P approximation form the approximation pyramid.
- The residual outputs form the residual pyramid.
- Approximation and residual pyramids are computed in an iterative fashion.
- A (P+1) level pyramid is build by executing the operations in the block diagram P times.

Image pyramids

- During the first iteration, the original $2^J x 2^J$ image is applied as the input image.

- This produces the level J-1 approximate and level J prediction residual results

- For iterations j = J-1, J-2, …, J-p+1, the previous iteration's level j-1 approximation output is used as the input.

Image pyramids

- Each iteration is composed of three sequential steps:

1. Compute a reduced resolution approximation of the input image. This is done by filtering the input and downsampling (subsampling) the filtered result by a factor of 2.
   - Filter: neighborhood averaging, Gaussian filtering
   - The quality of the generated approximation is a function of the filter selected

Image pyramids

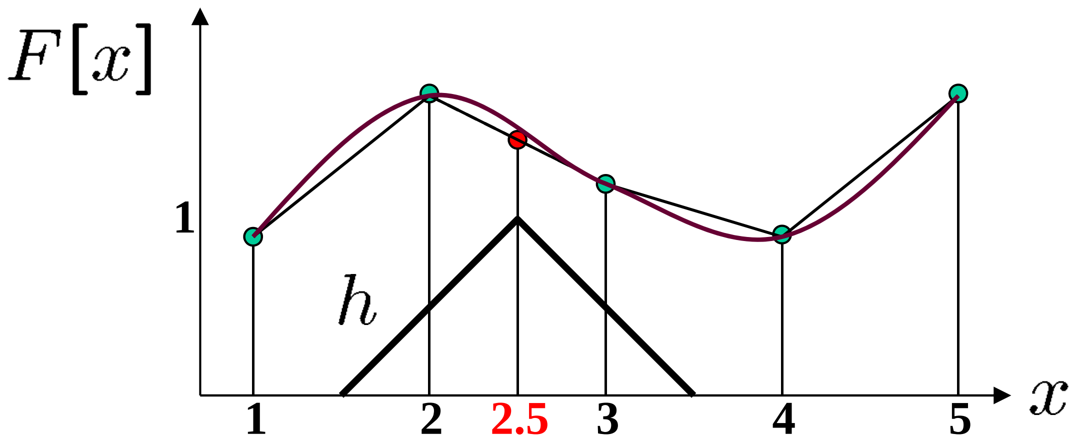
2. Upsample output of the previous step by a factor of 2 and filter the result. This creates a prediction image with the same resolution as the input.
   – By interpolating intensities between the pixels of step 1, the interpolation filter determines how accurately the prediction approximates the input to step 1.

3. Compute the difference between the prediction of step 2 and the input to step 1. This difference can be later used to reconstruct progressively the original image

Image resampling

So what to do if we don't know f

- **Answer: guess an approximatio$\tilde{f}$**
- **Can be done in a principled way: filtering**



d = 1 in this example

Image reconstruction

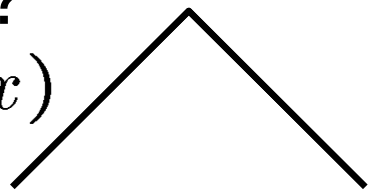
- **Convert F to a continuous function**
  $f_F(x) = F(\frac{x}{d})$ when $\frac{x}{d}$ is an integer, 0 otherwise
- **Reconstruct by cross-correlation:**

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

Resampling filters

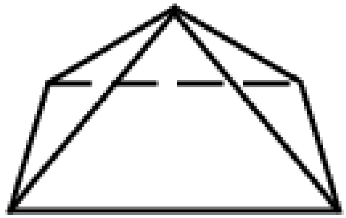
What does the 2D version of this hat function look like?

$h(x)$



performs
linear interpolation

$h(x, y)$



(tent function) performs
bilinear interpolation

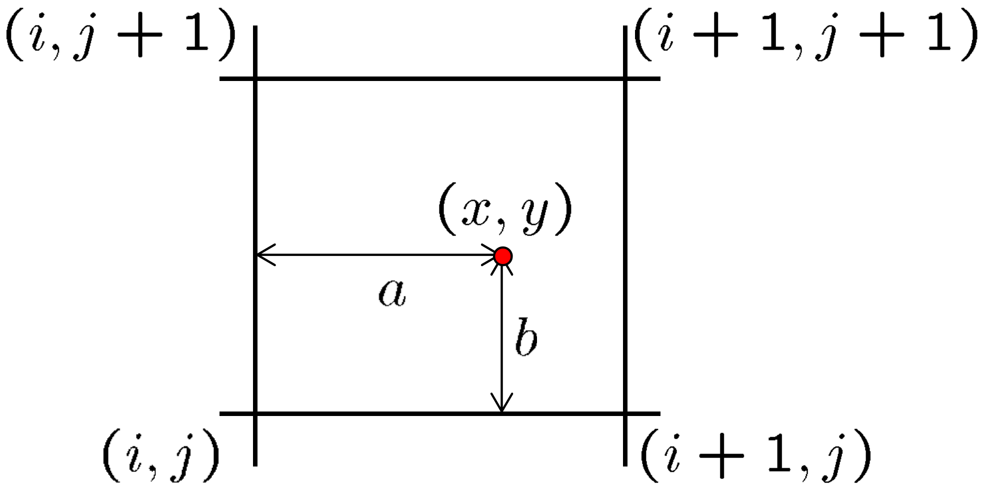
Better filters give better resampled images

- Bicubic is common choice

Why not use a Gaussian?

Bilinear interpolation

Sampling at *f(x,y):*



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

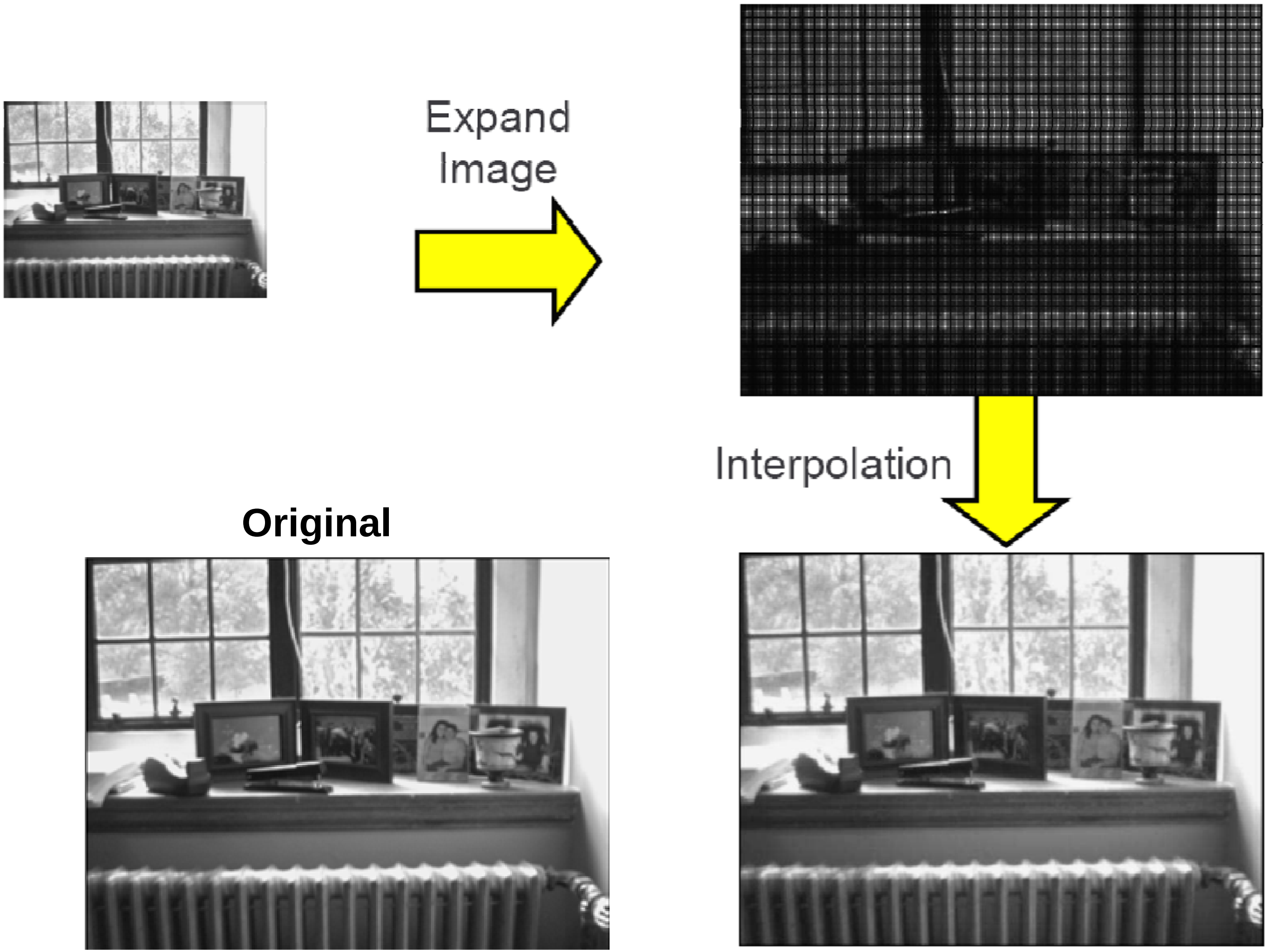
Decimation

$$* \quad \begin{matrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{matrix}$$

Filter

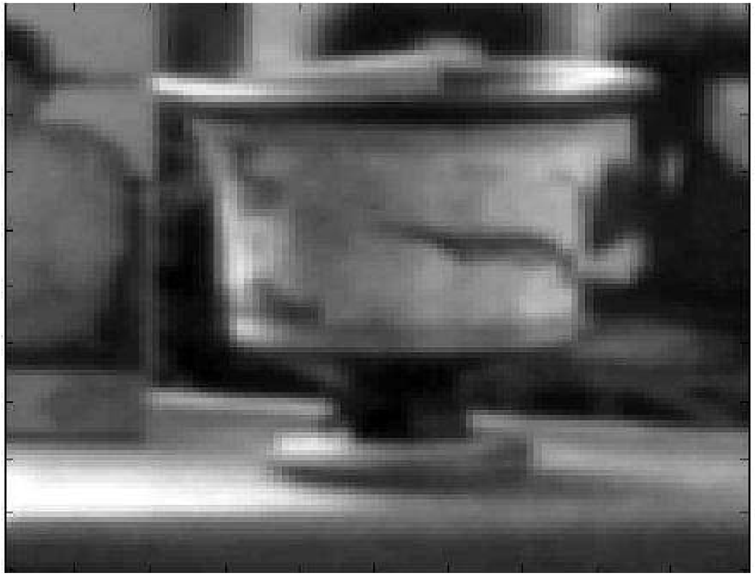
Subsample

Expansion



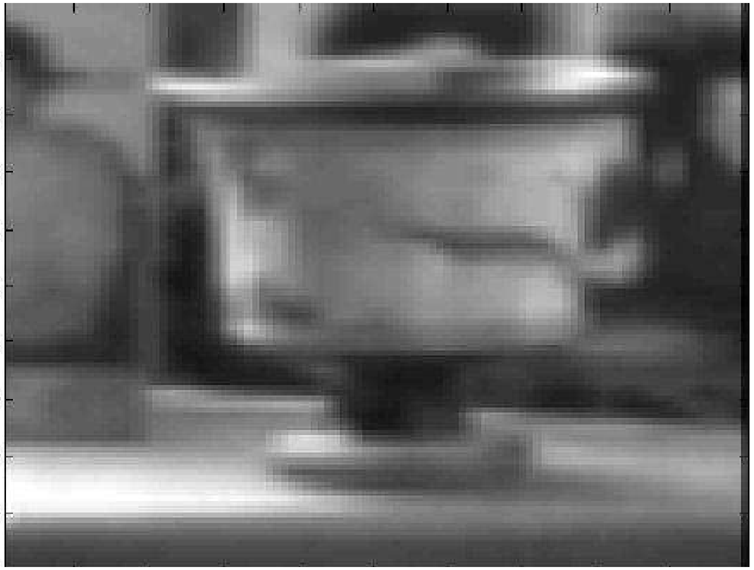
Interpolation Results

Original Image



Nearest Neighbor

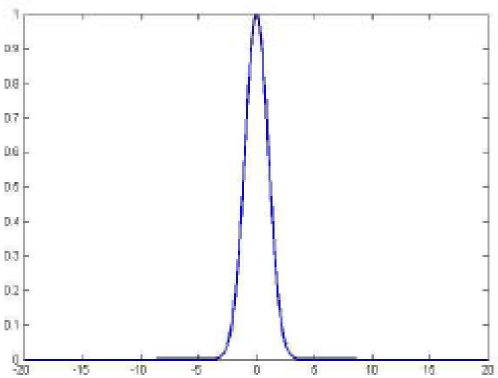
Bilinear Interpolation



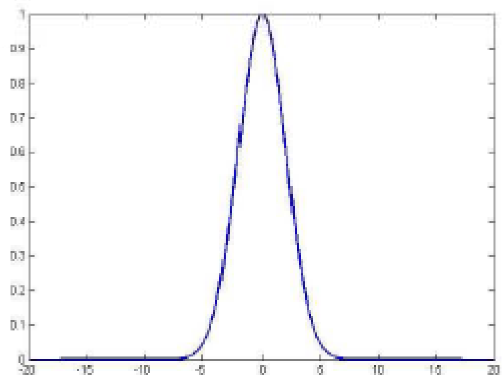
Pyramids at Same Resolution

The Gaussian Pyramid

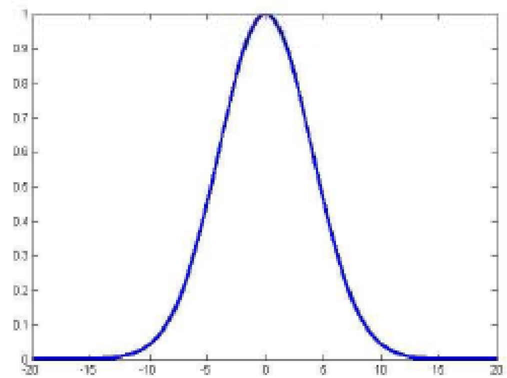
- Smooth with Gaussians because
  - a Gaussian*Gaussian=another Gaussian
- Synthesis
  - smooth and downsample
- Gaussians are low pass filters, so repetition is redundant
- Kernel width doubles with each level



Level 1



Level 2

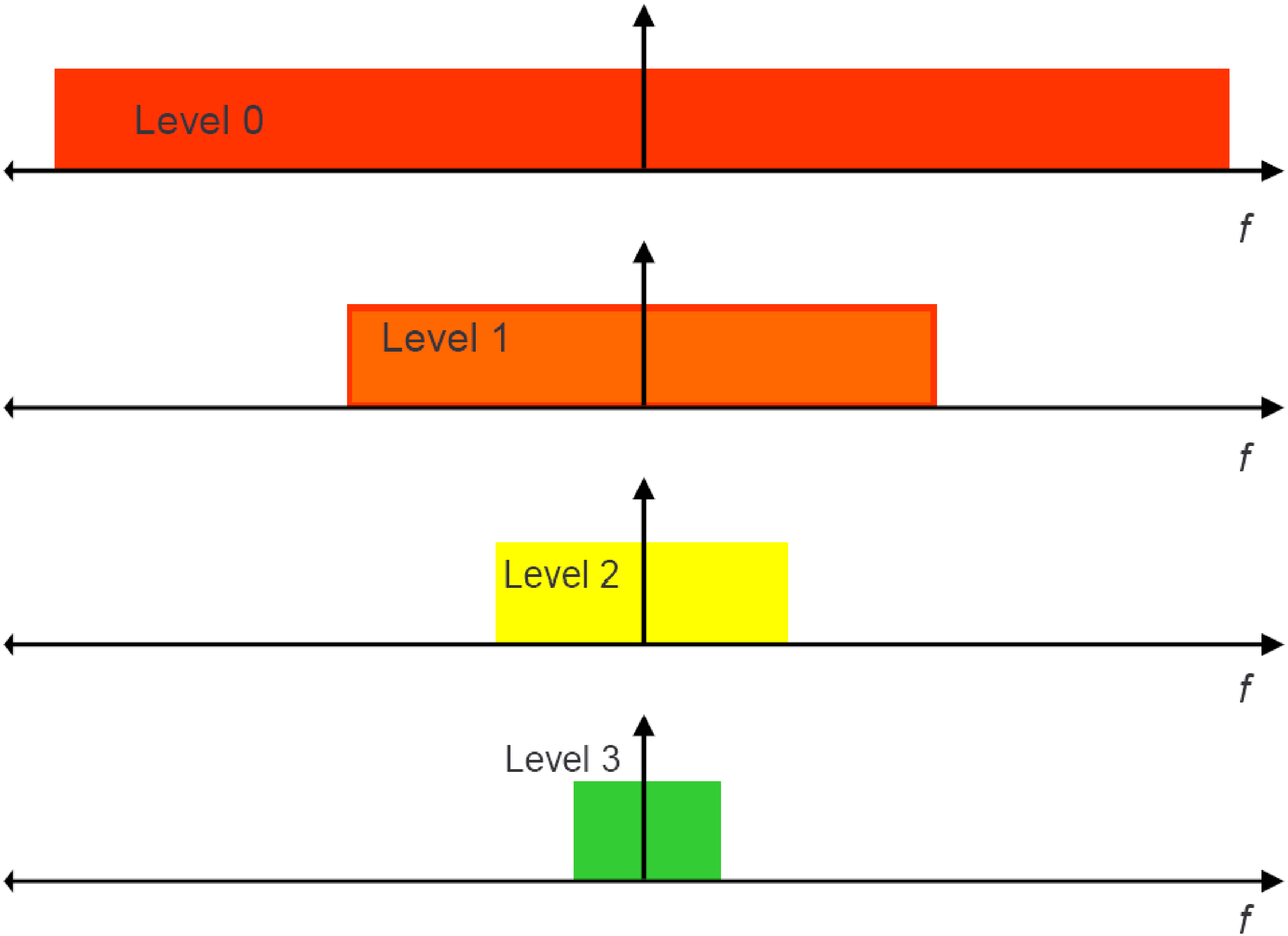


Level 3

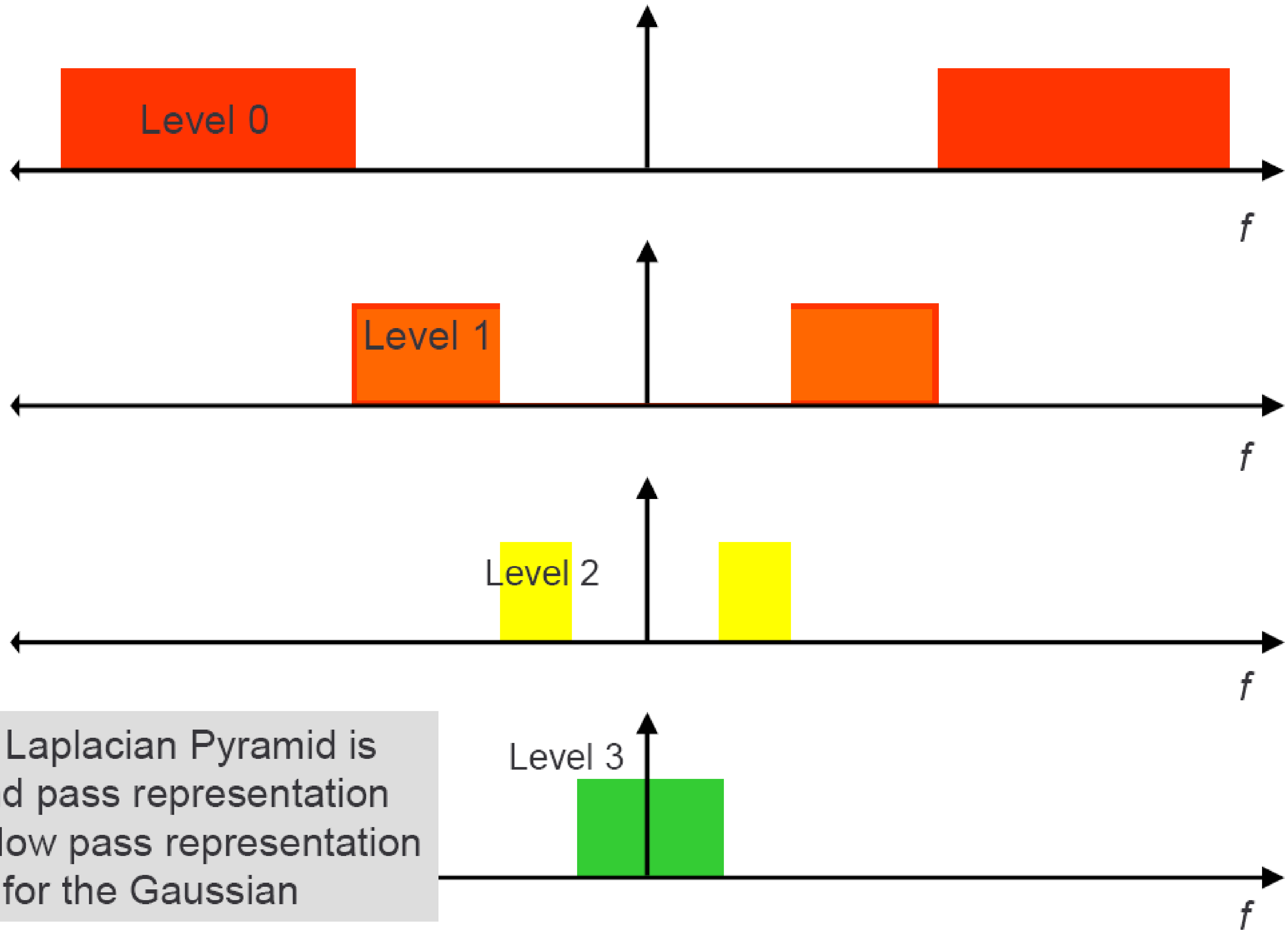
Smoothing as low-pass filtering

- **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.**

Gaussian Pyramid Frequency Composition



Laplacian Pyramid Frequency Composition



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

SCALE-SPACE AND diffusion - Theory

The inner scale is the smallest detail seen by the smallest aperture (e.g. the CCD element, a cone or rod);
The outer scale is the coarsest detail that can be discriminated, i.e. it is the whole image (field of view).

Convolution with a Gaussian necessarily increases the inner scale: the Gaussian is the operator that transforms the inner scale of the image.

The cascade property states that it is the same if one reaches a final certain scale in a single step from the input image by a given Gaussian aperture, or apply a sequence of Gaussian kernels, to reach the same scale.

The stack of images as a function of increasing inner scale is coined a linear 'scale-space'.

The generating equation of a linear scale-space is the linear diffusion equation.

The scale-space family can be defined as the solution of the diffusion equation (for example, in terms of the heat equation):

$$\partial_t L = \frac{1}{2}\nabla^2 L,$$

with initial condition, L(x, y; 0) = f(x, y).

Linear diffusion equation:

$$\frac{\partial L}{\partial s} = \vec{\nabla}.\vec{\nabla} L = L_{xx} + L_{yy}$$

Derivative to scale equals the divergence of the gradient of the luminance function, which is the Laplacian, the sum of the second partial derivatives. Soln,. Given as:

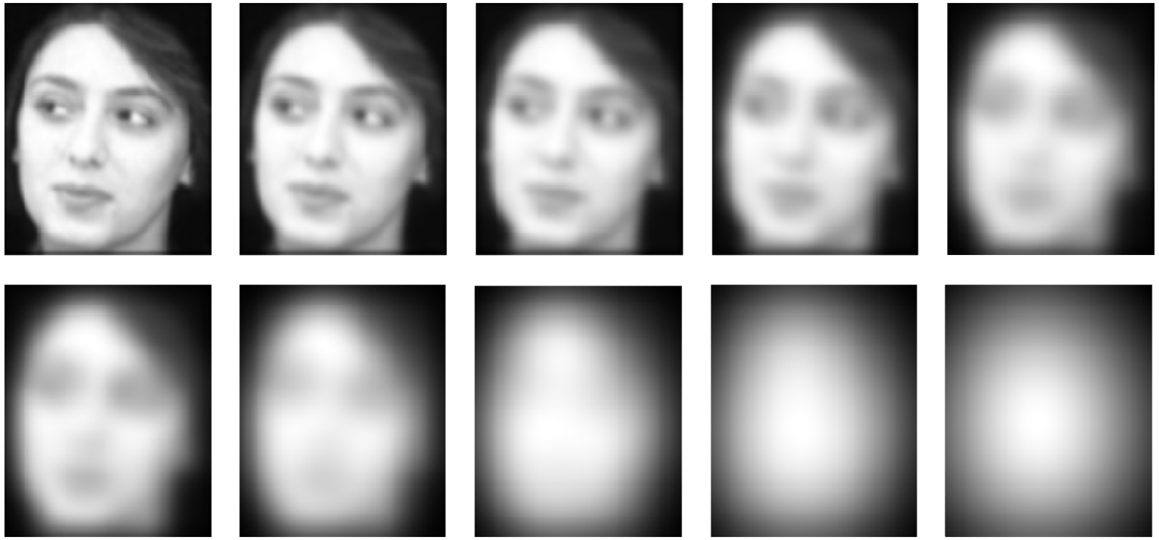
$$L(\cdot,\cdot;t) = g(\cdot,\cdot;t) * f(\cdot,\cdot),$$

The Gaussian is the Green's function of the diffusion equation.

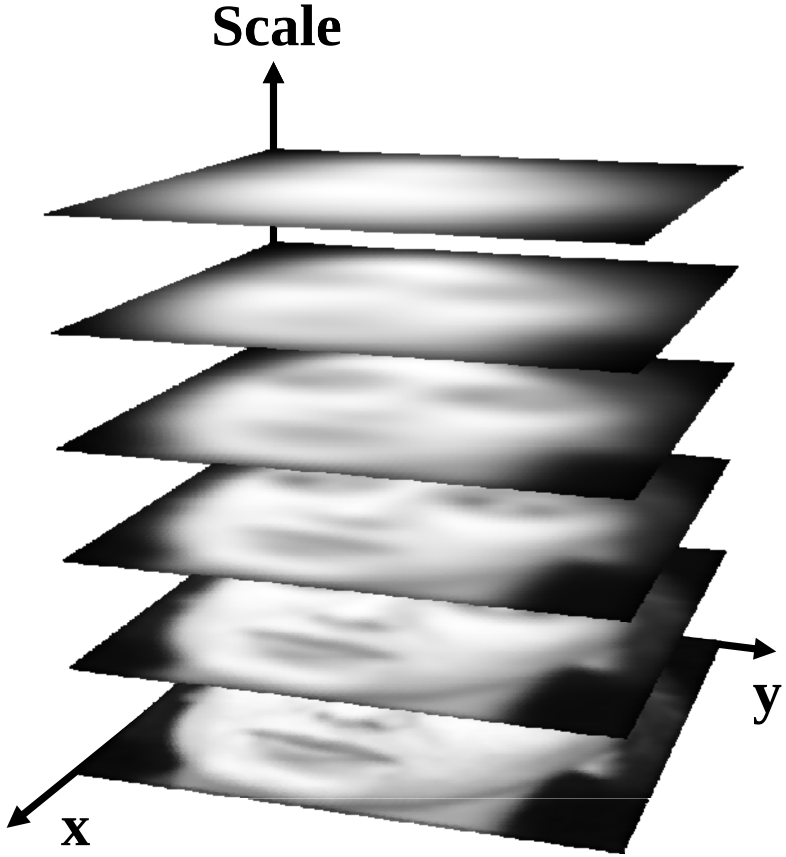
When the diffusion is equal for all directions, i.e. the sigma's of the Gaussian are equal, we call the process isotropic.

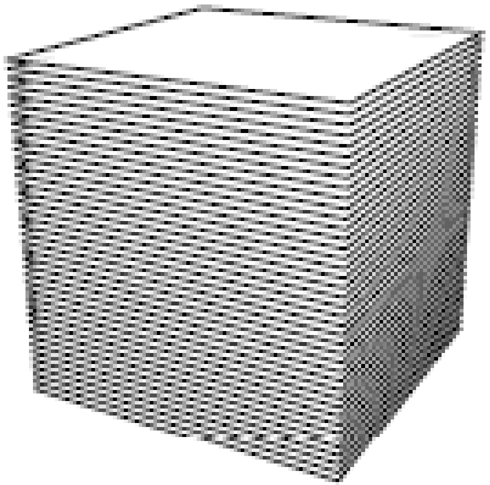
When the diffusion is equal for every point of the image, we call the process homogeneous.

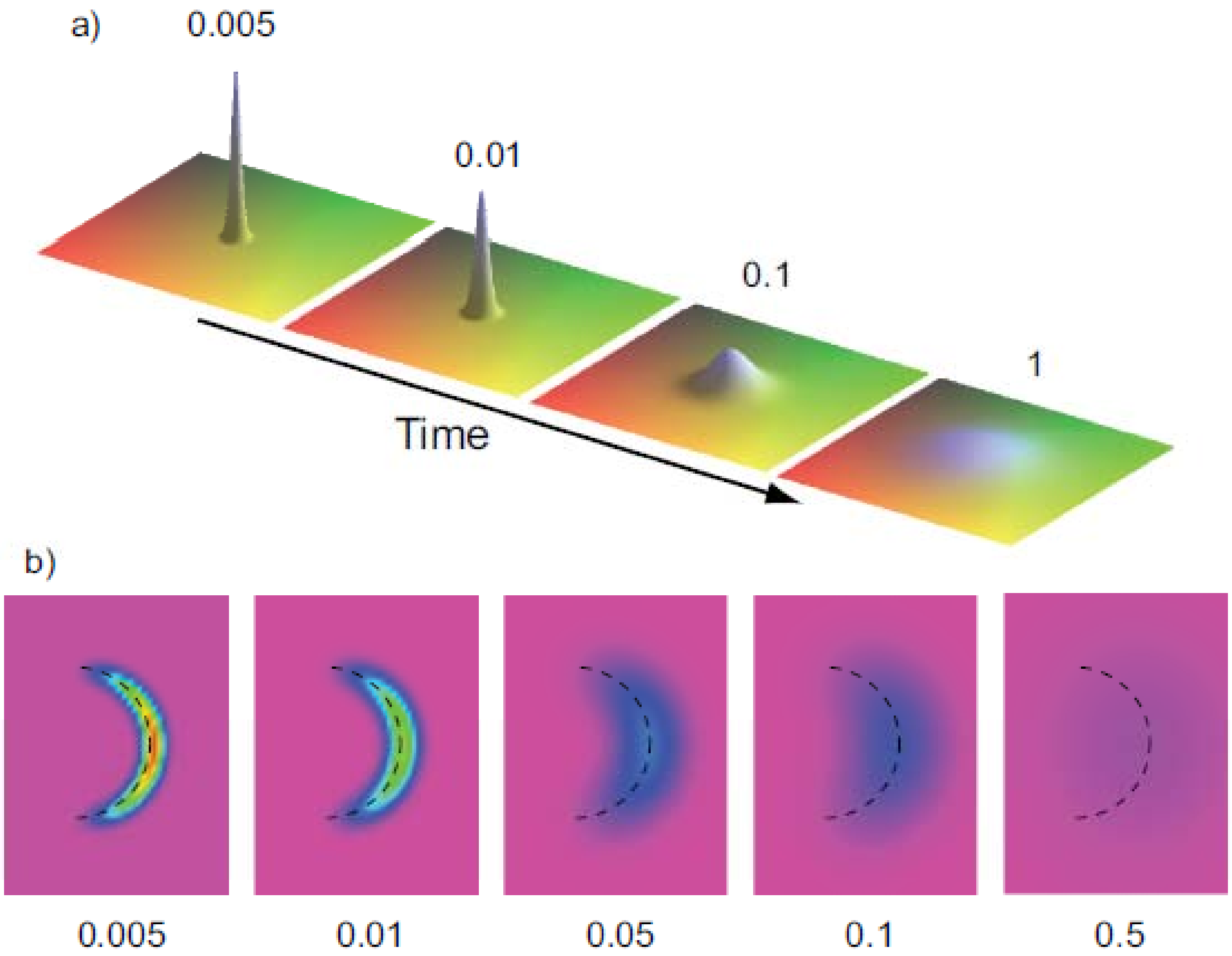
Because of the diffusion equation, the process of generating a multiscale representation is also known as image evolution.



Scale Space

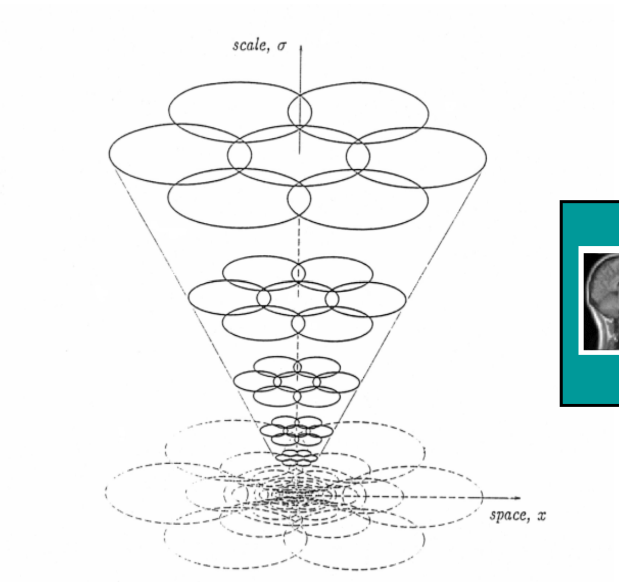


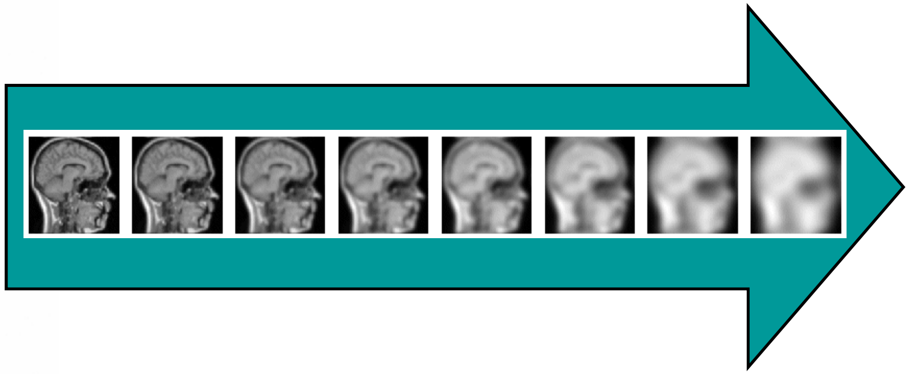


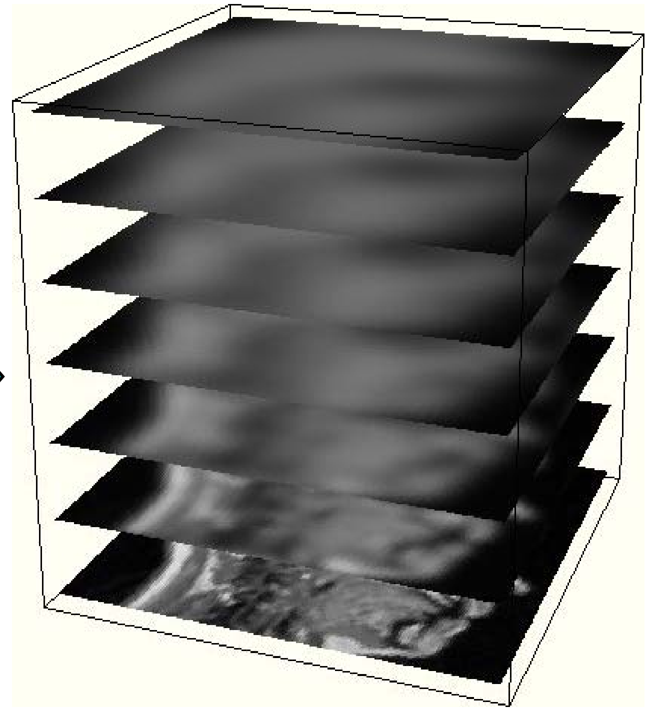


Diffusion in two dimensions.

The retina measures on many resolutions simultaneously







scale-space

Gaussian Derivatives:

It is well known that derivative operations performed on a discrete grid are an **ill-posed problem, meaning derivatives are overly sensitive to noise**.

To convert derivative operations into a well-posed problem, the image is low-pass filtered or smoothed prior to computing the derivative

Another useful result in linear scale-space theory is that:

the **spatial derivatives of the Gaussian** are solutions of the diffusion equation too, and together with the zeroth order Gaussian they form a complete **family of differential operators**.

From scale-space solution: $L(\cdot,\cdot;t) = g(\cdot,\cdot;t) * f(\cdot,\cdot),$

We, obtain scale-space derivatives, as:

$$L_{x^\alpha y^\beta}(.,.;t) = \partial_{x^\alpha y^\beta} L(.,.;t) = [\partial_{x^\alpha y^\beta}\{g(.,.;t)\}] * f(.,.)$$

Gaussian Derivatives:

$$\frac{\partial}{\partial x}(L*G) = L*\frac{\partial G}{\partial x} \qquad \frac{\partial^n G}{\partial x^n} \rightarrow \sigma^n \frac{\partial G}{\partial x}$$

From scale-space solution: $L(\cdot,\cdot;t) = g(\cdot,\cdot;t) * f(\cdot,\cdot),$

We, obtain scale-space derivatives, as:

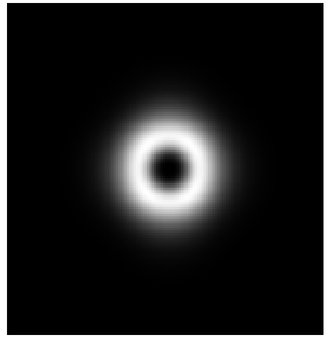
$$L_{x^\alpha y^\beta}(.,.;t) = \partial_{x^\alpha y^\beta} L(.,.;t) = [\partial_{x^\alpha y^\beta}\{g(.,.;t)\}]* f(.,.)$$

The smoothing to regularize the image is implemented as a convolution over the image and therefore this filtering operation is linear.

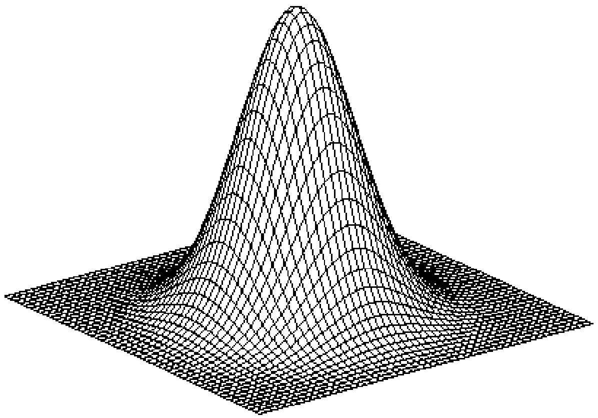
Since differentiation is also a linear operation, the order of smoothing and differentiation can be switched, which means the derivative of the convolution kernel can be computed and convolved with the image resulting in a well-posed measure of the image derivative.

Gaussian – Image filter

$$G(\sigma) = \frac{1}{(2\pi\sigma^2)} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

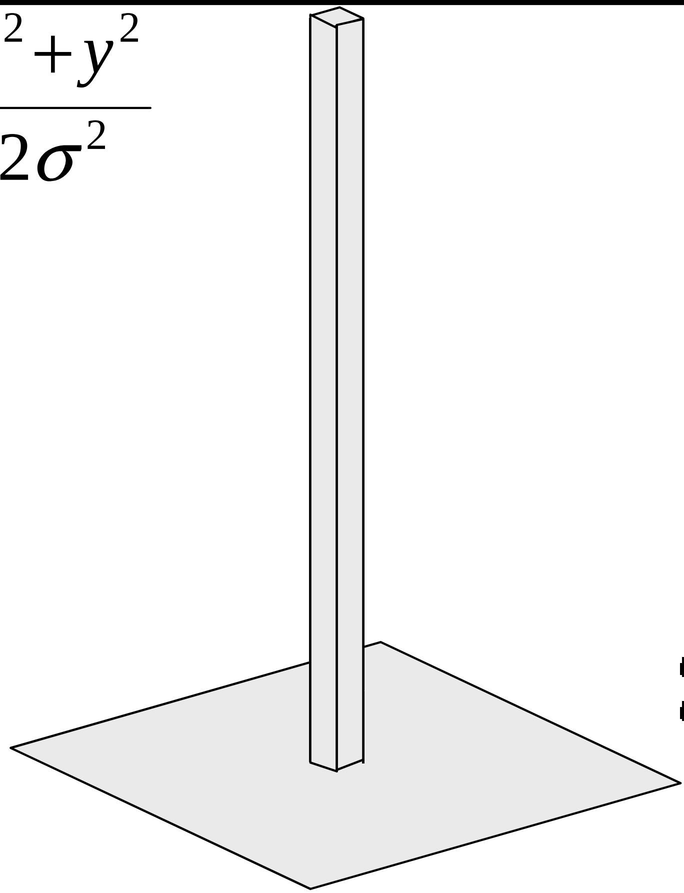


Shape of L(w) ?
Fourier Transform



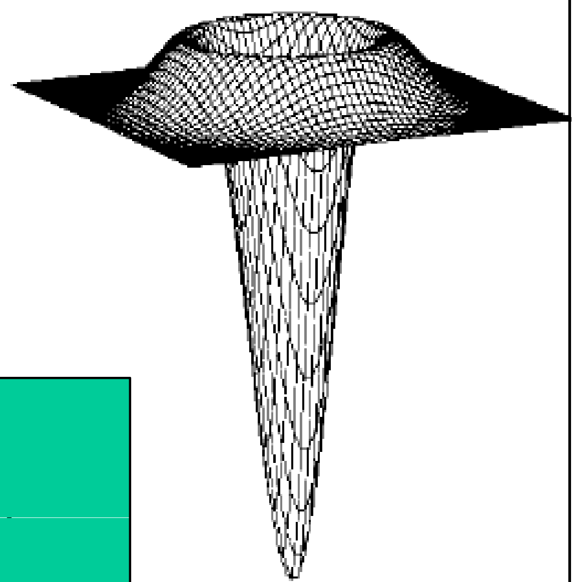
Gaussian

$-$



Delta function

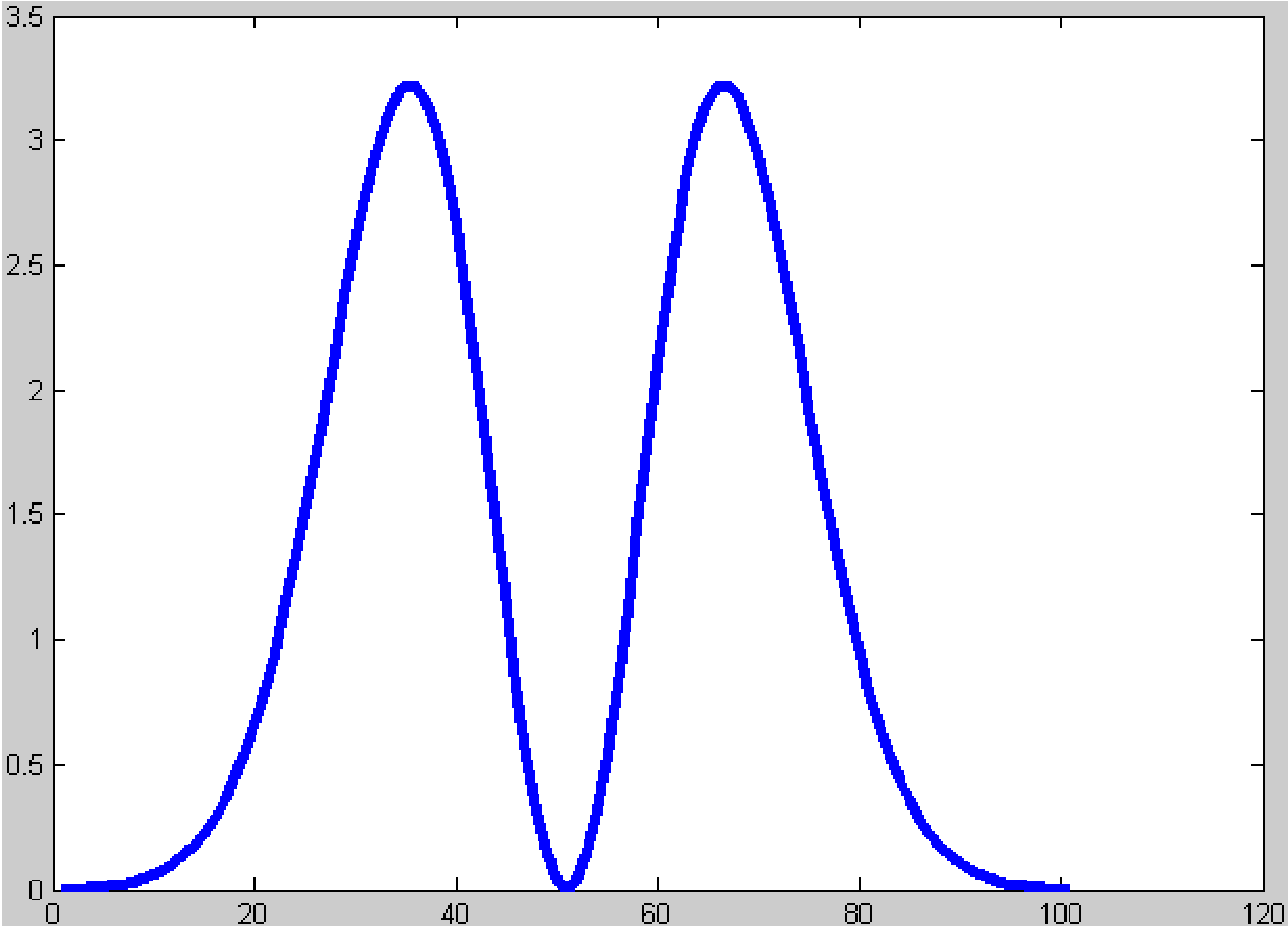
$\approx$



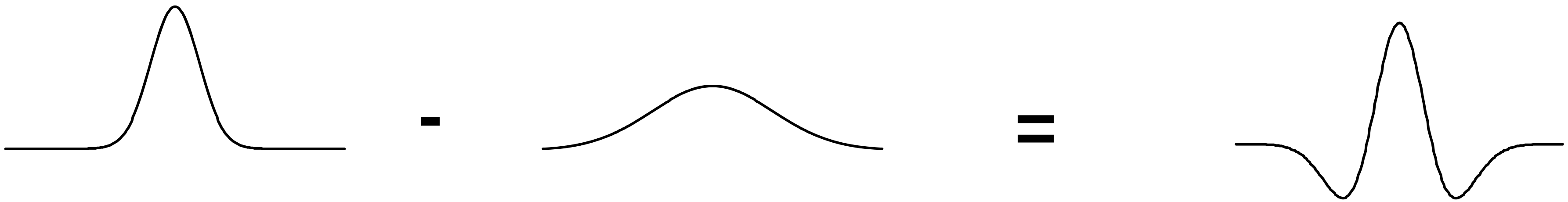
$$G(\omega) = \frac{1}{(\sqrt{2\pi}\sigma)} e^{-\frac{\sigma^2\omega^2}{2}}$$

$$L(\omega) =$$

$$G(\omega) = \frac{1}{(\sqrt{2\pi}\sigma)} e^{-\frac{\sigma^2\omega^2}{2}} \qquad L(\omega) = \frac{-\omega^2}{(\sqrt{2\pi}\sigma)} e^{-\frac{\sigma^2\omega^2}{2}}$$



Laplacian ~ Difference of Gaussians



DOG = Difference Of Gaussians

Typically, $\sigma_2 = 1.6 * \sigma_1$;

Difference of Gaussians (DoG)

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

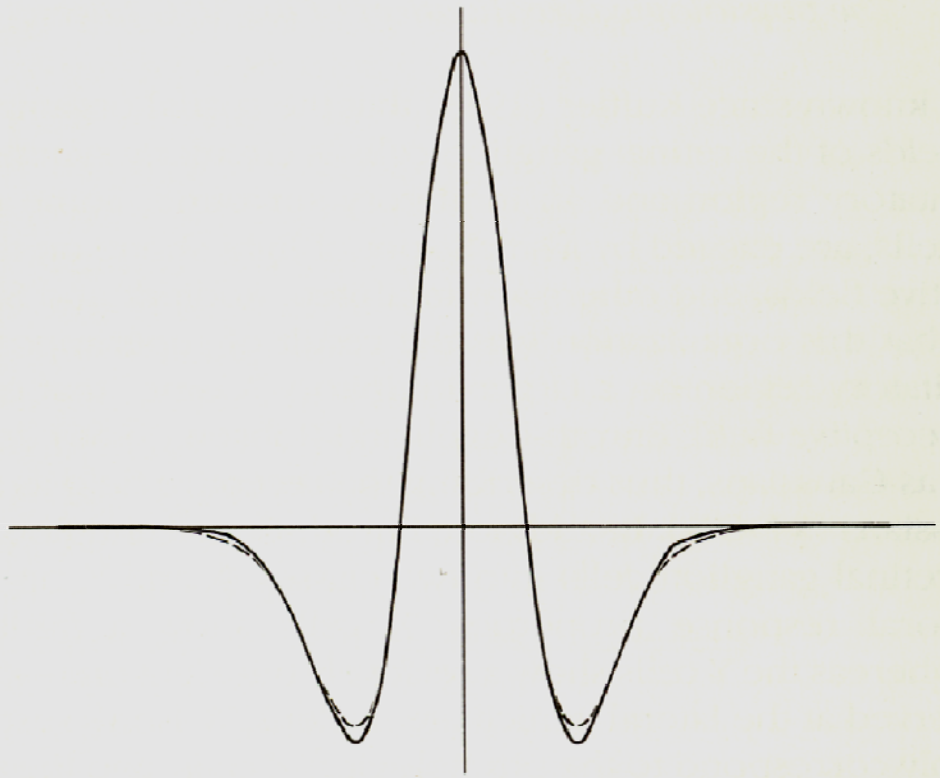


Figure 2–16. The best engineering approximation to $\nabla^2 G$ (shown by the continuous line), obtained by using the difference of two Gaussians (DOG), occurs when the ratio of the inhibitory to excitatory space constraints is about 1:1.6. The DOG is shown here dotted. The two profiles are very similar.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

Gaussian Pyramid

- Synthesis: Smooth image with a Gaussian and downsample. Repeat.
- Analysis: Take top image or search over scale
  - Face detection
- Redundant (over-complete) representation, in comparison to wavelet decomposition.
- Top levels come “for free”. Processing cost typically dominated by lowest two levels.

The Laplacian Pyramid

- **Synthesis**
  - **preserve difference between upsampled Gaussian pyramid level and Gaussian pyramid level**
  - **band pass filter - each level represents spatial frequencies (largely) unrepresented at other levels**
- **Analysis**
  - **reconstruct Gaussian pyramid, take top layer**

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

Precomputation

- Need to access image at different blur levels
- Useful for texture mapping at different resolutions (called mip-mapping)

Image Processing

- Editing frequency bands separately
- E.g. image blending…

Applications of scaled representations

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
- 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.**

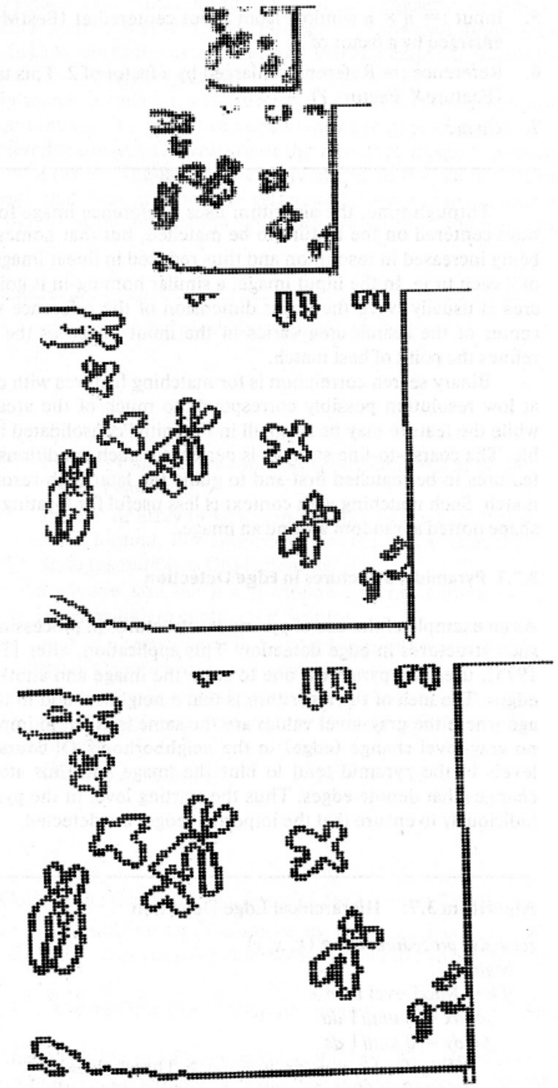


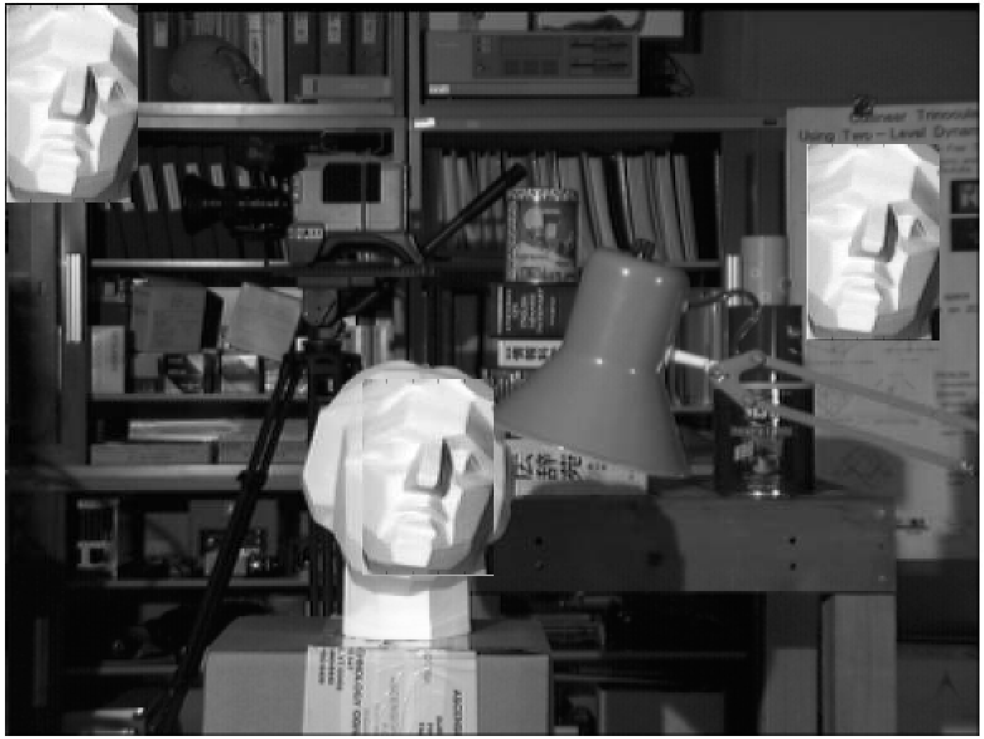
Fig. 3.37 Pyramidal edge detection.

10 Ch. 3 Early Processing

Fast Template Matching

Template

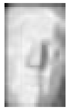
Search Region

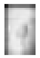


- For an m x n image…
- For a p x q template…
- The complexity of the 2D pattern recognition task is O(mnpq) ☹
- This gets even worse for a family of templates (*e.g.*, to address scale and/or rotational effects)

Fast Template Matching

Template





Search Region

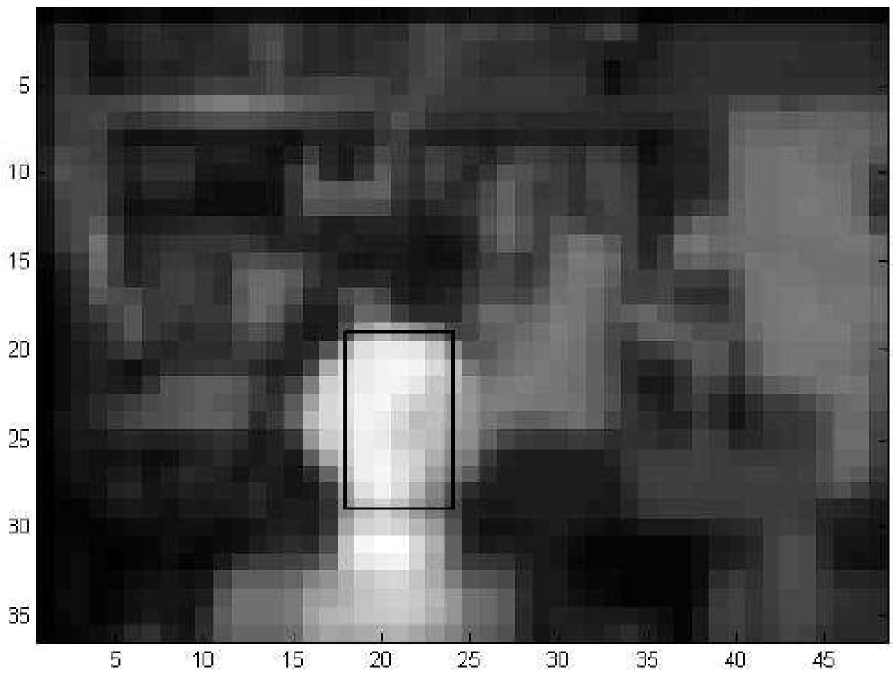
Original Image

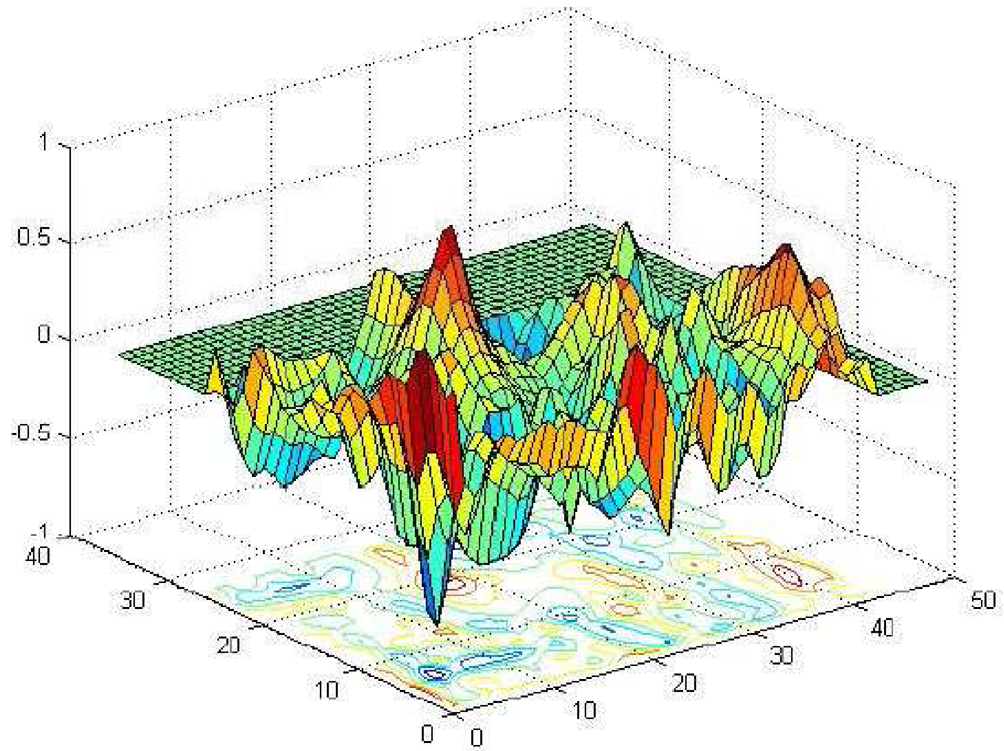
Multi-resolution correlation

- 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

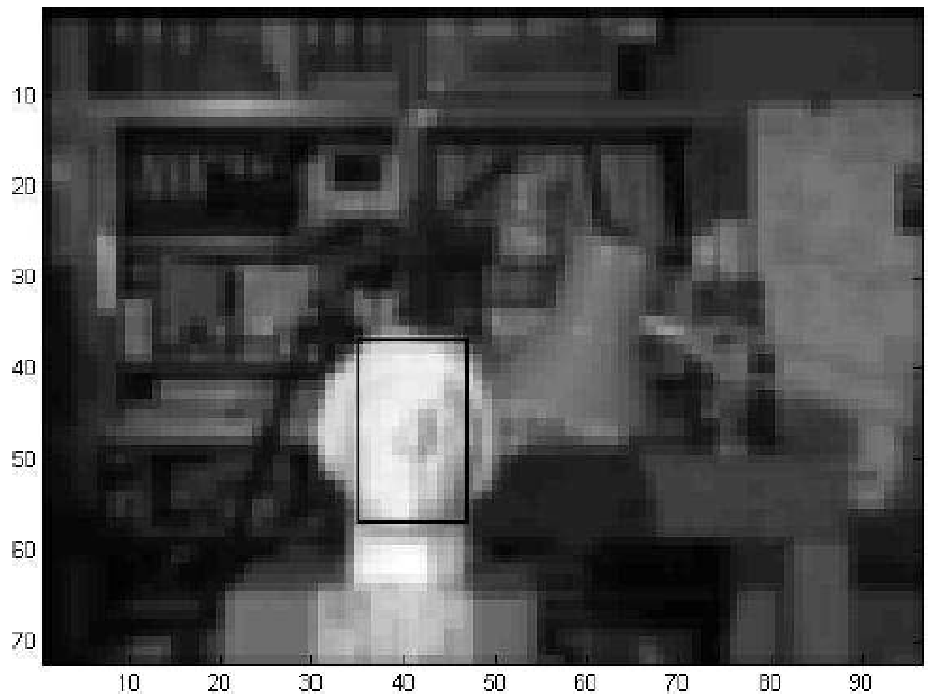
- At the lowest pyramid level, we search the entire image with the correlation template

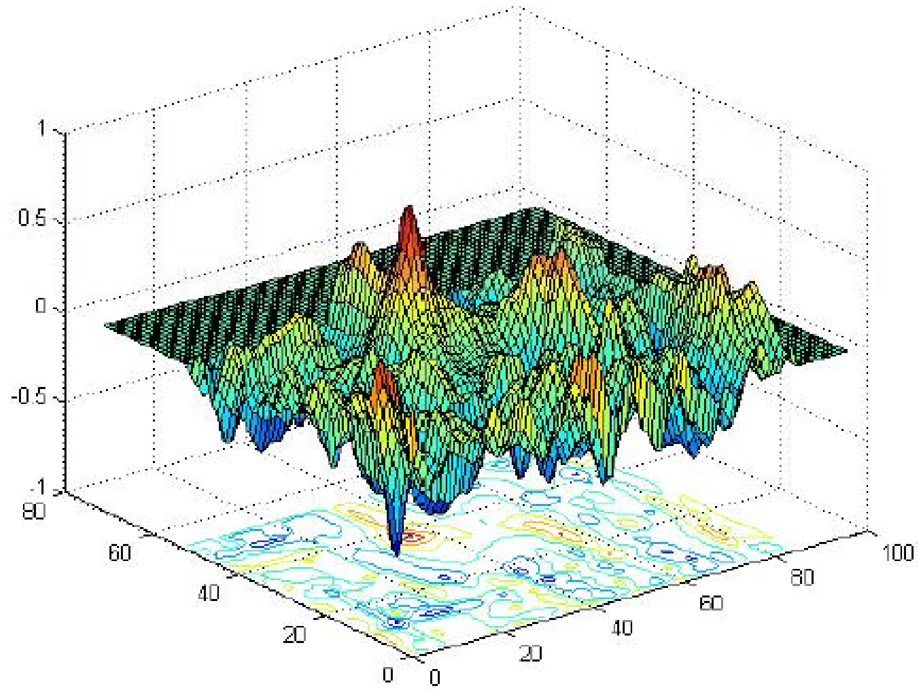




Level 2 Search

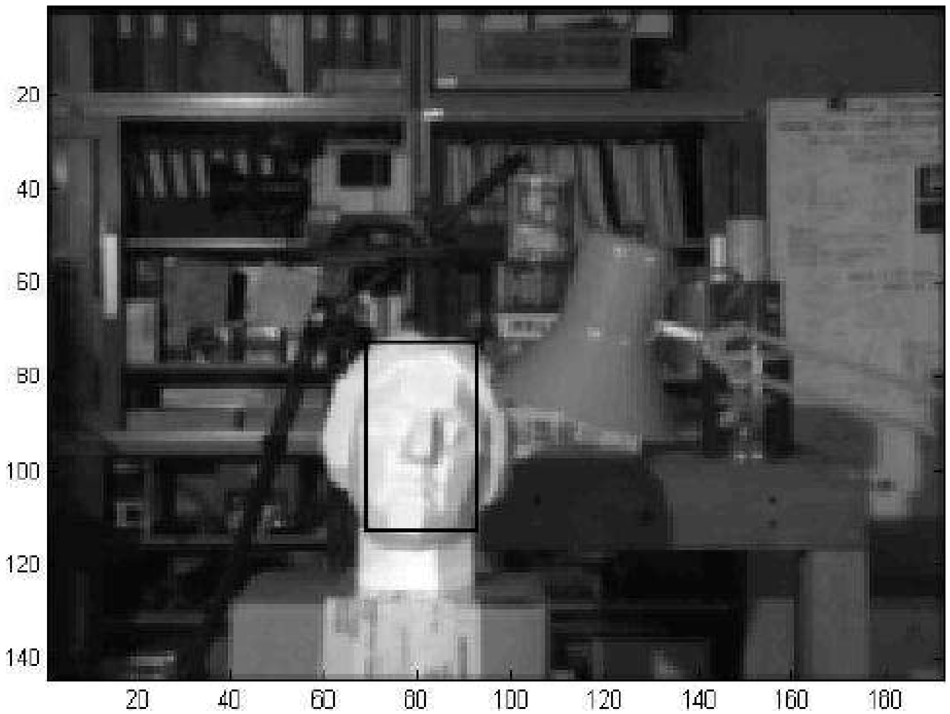
- Subsequent searches are constrained to a neighborhood of only several pixels in the x and y directions

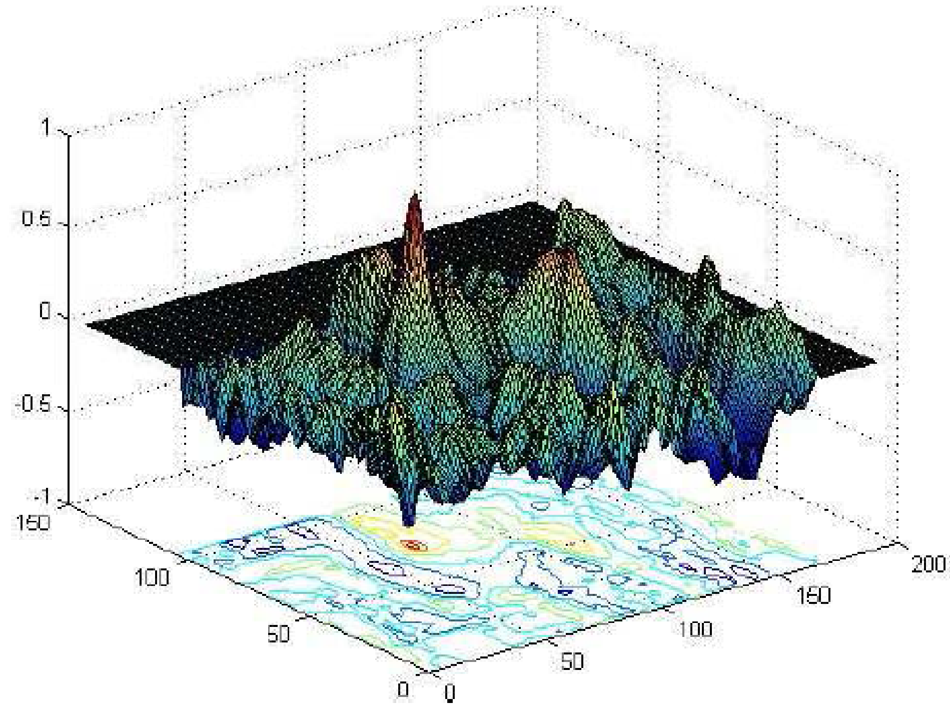




Level 1 Search

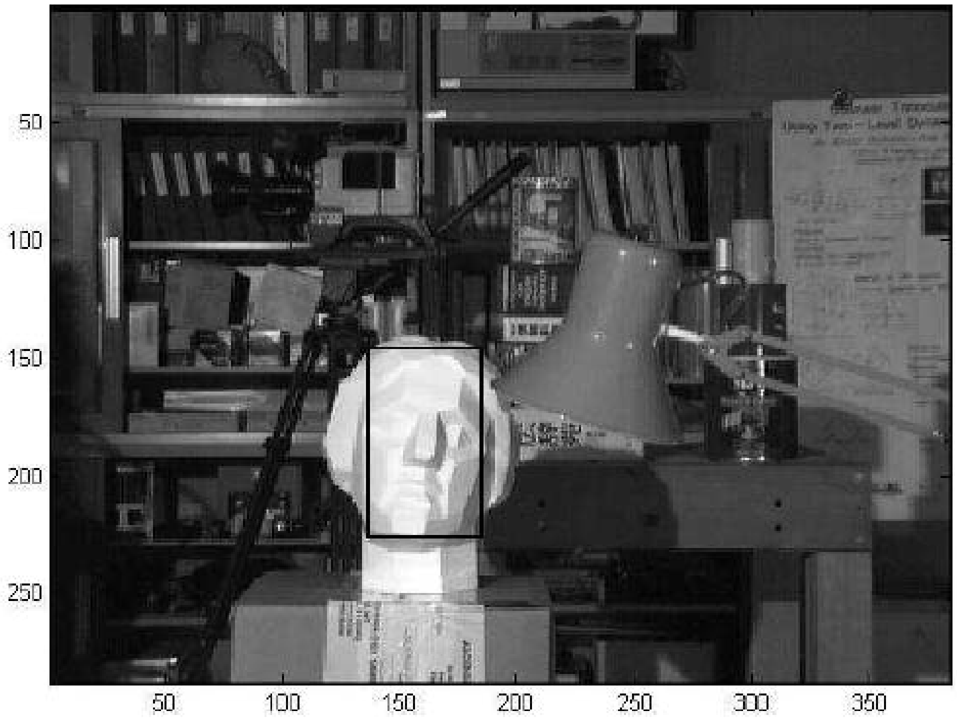
- Subsequent searches are constrained to a neighborhood of only several pixels in the x and y directions

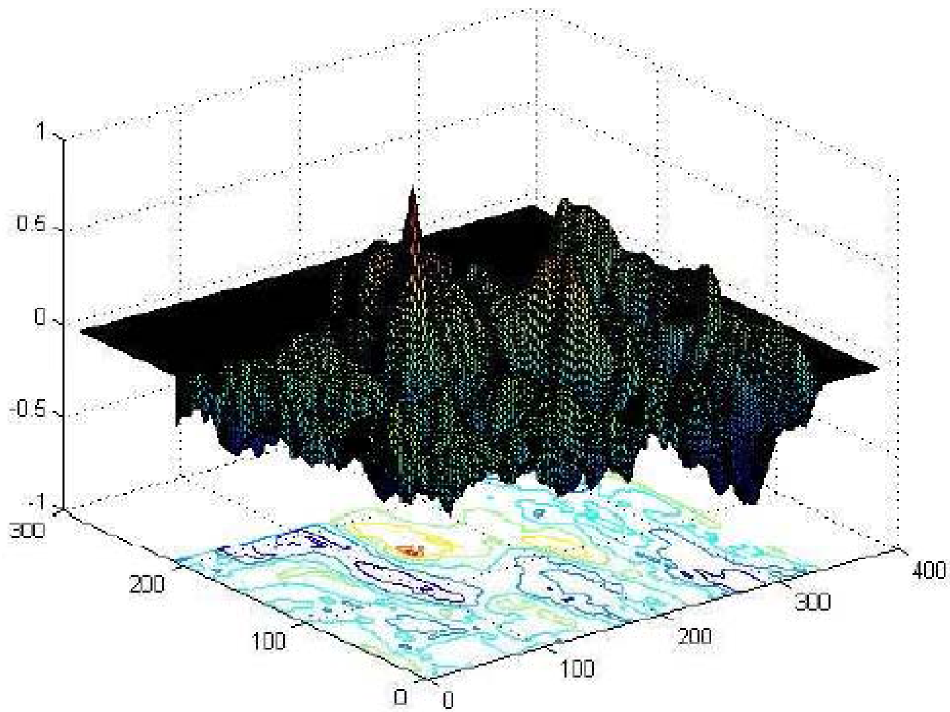




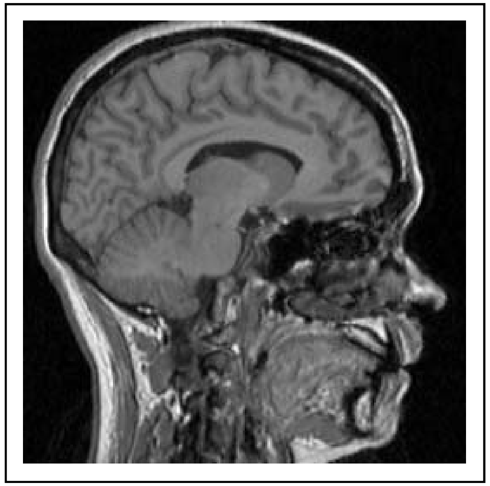
Level 0 Search

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

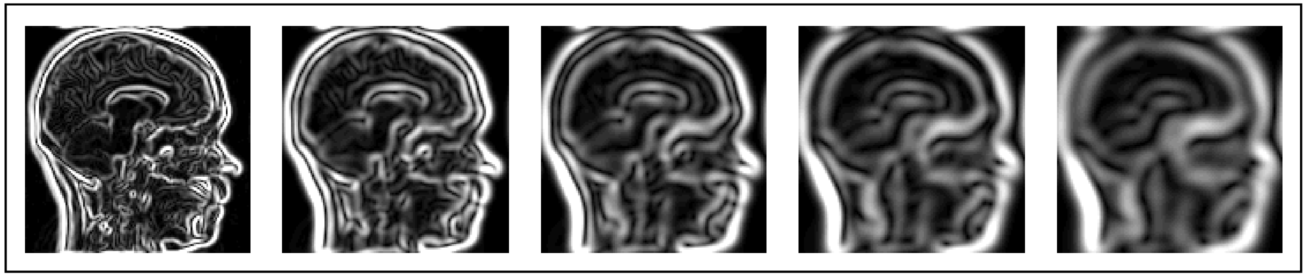




We can Calculate Derivatives and Combinations of them at all Scales

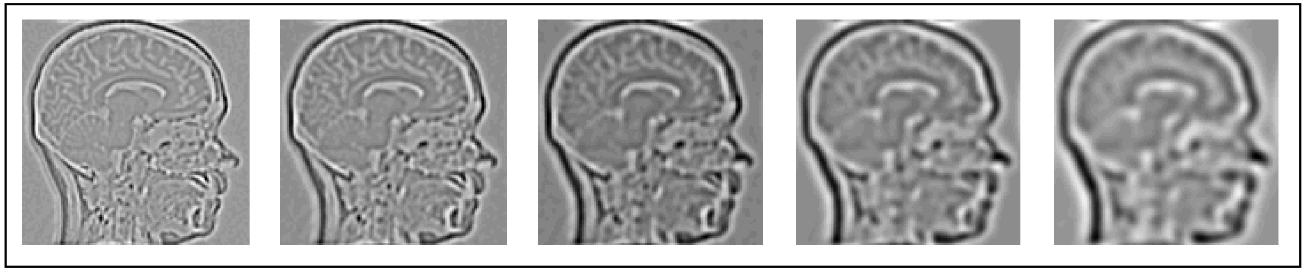


Original Image



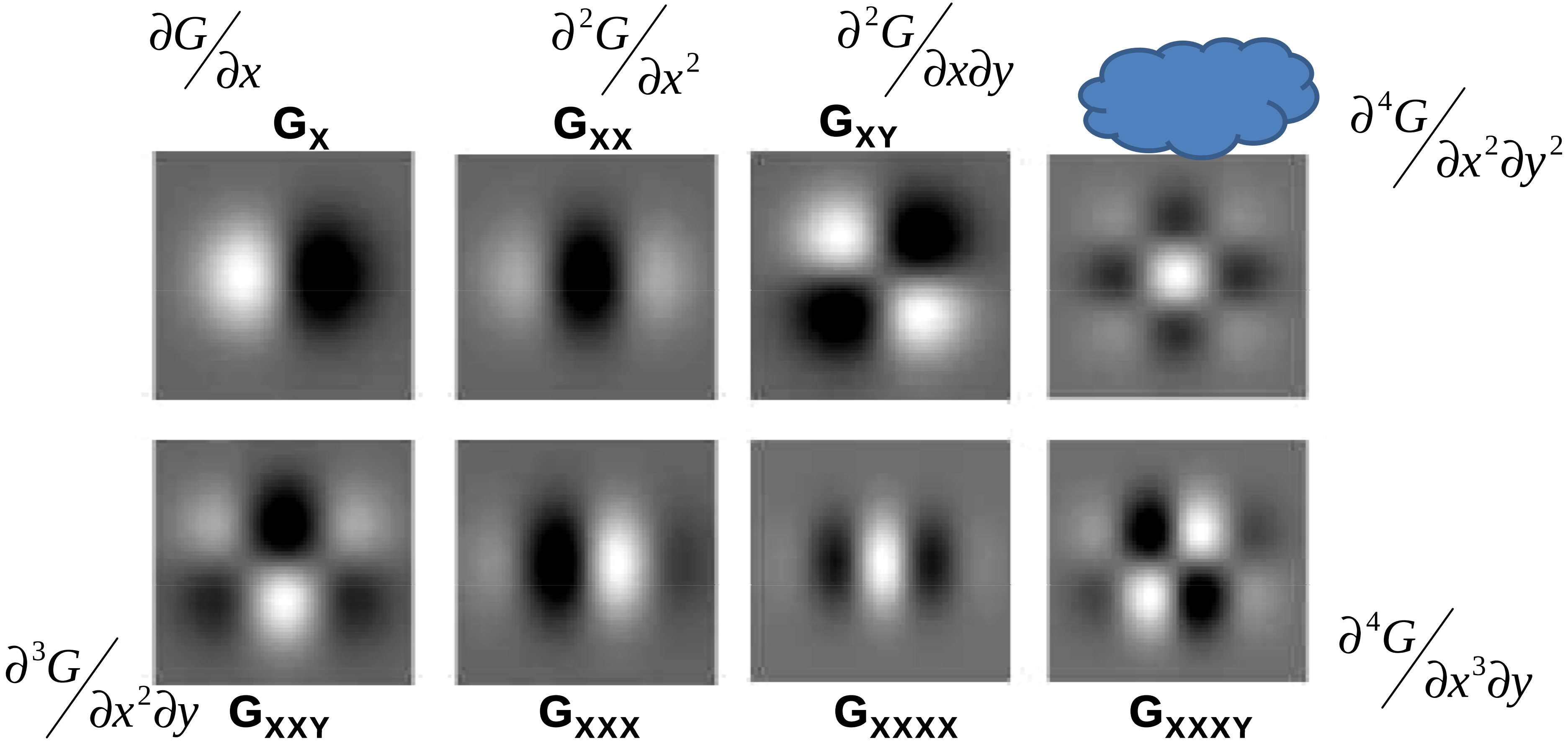
Gradient Magnitude

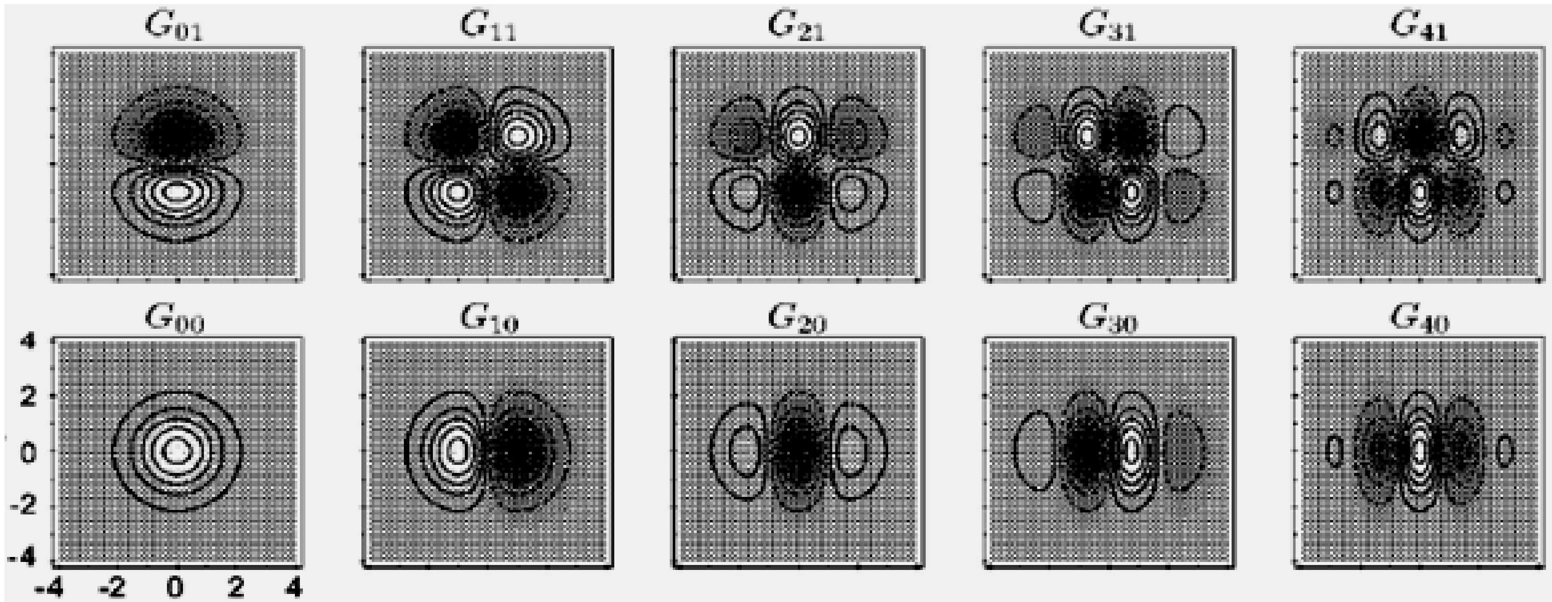
$$\sqrt{L_x^{\,2} + L_y^{\,2}}$$



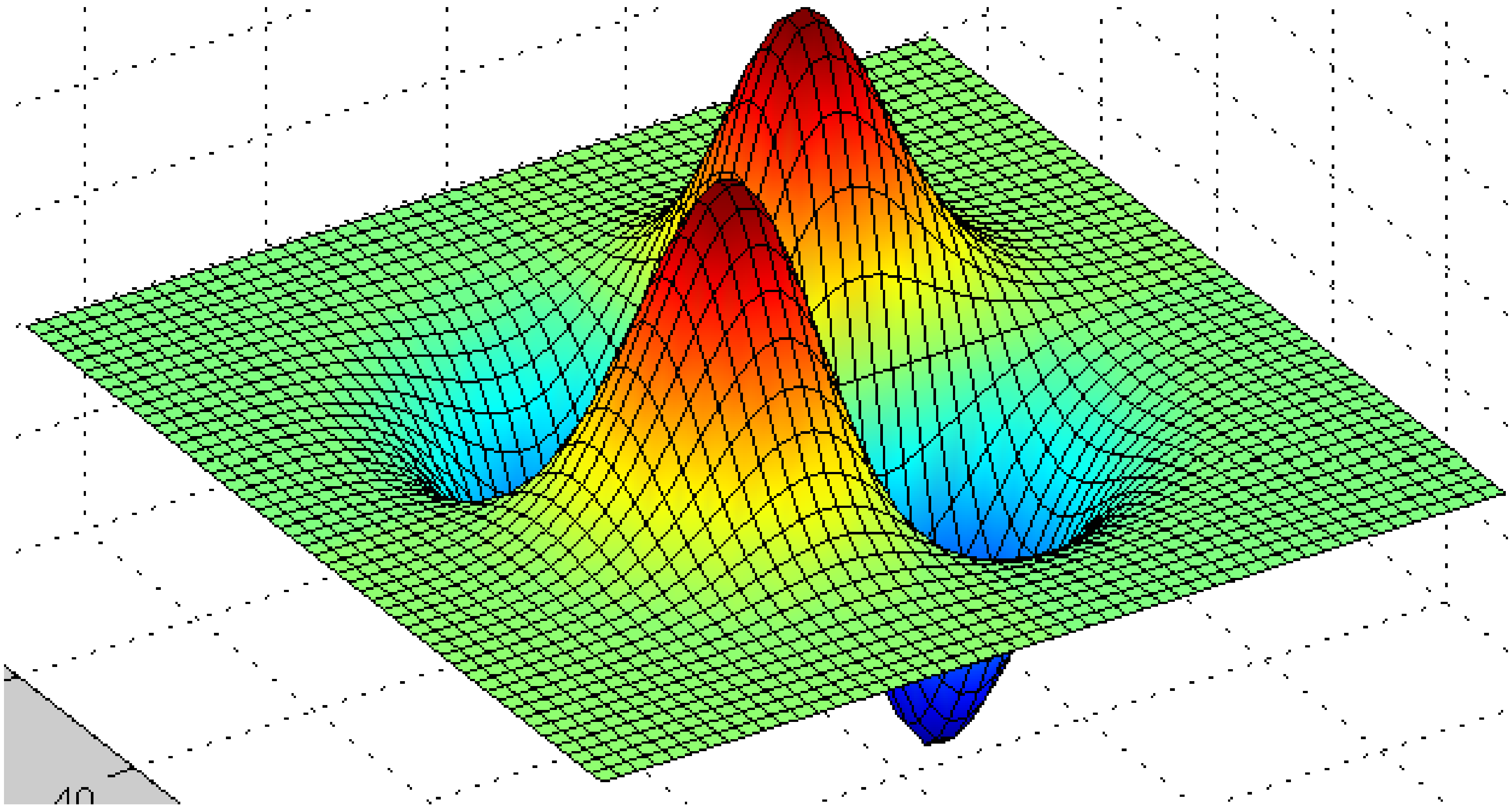
Laplacian

$$L_{xx} + L_{yy}$$



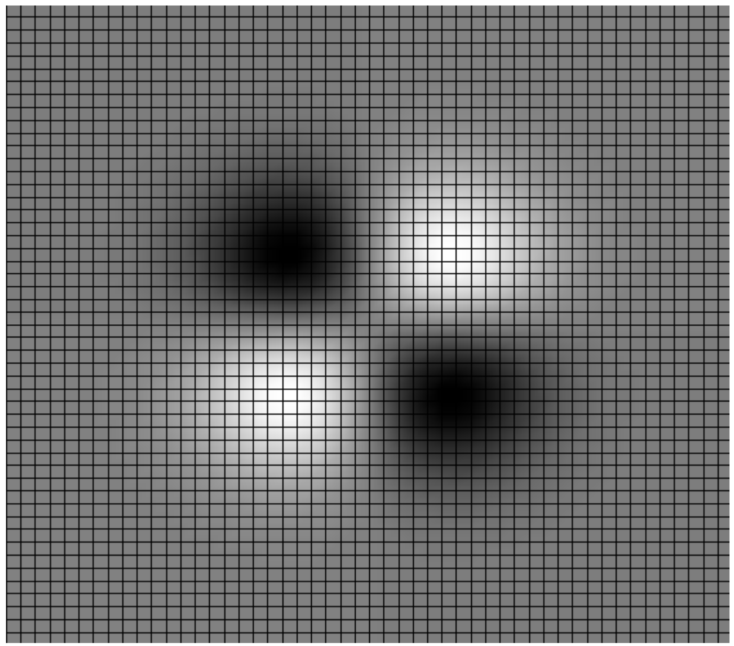


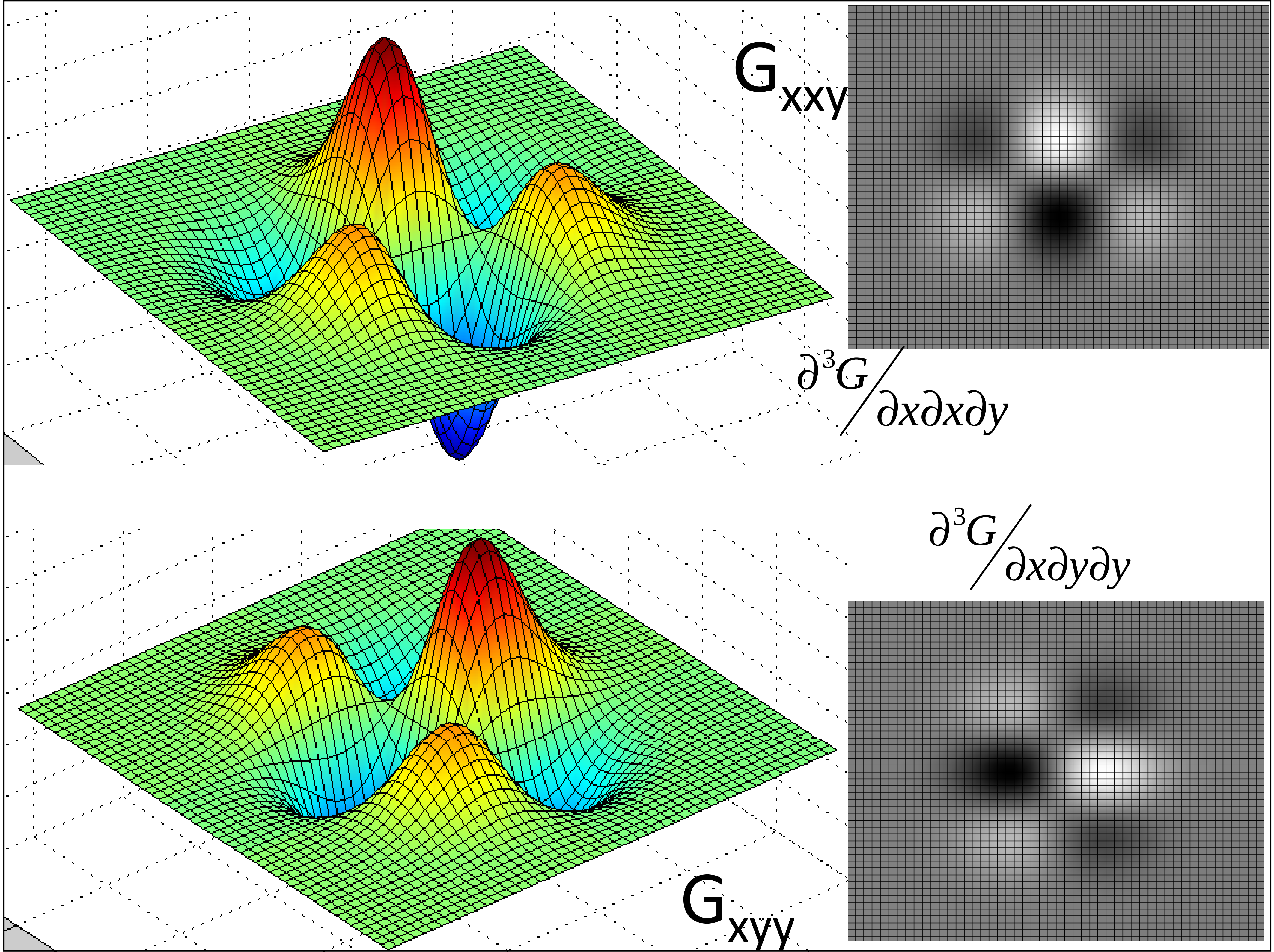
Mikolajczyk, K. and Schmid, C.: Scale and affine invariant interest point detectors, Int. Journal of Computer Vision, 60:1, 63 - 86, 2004.



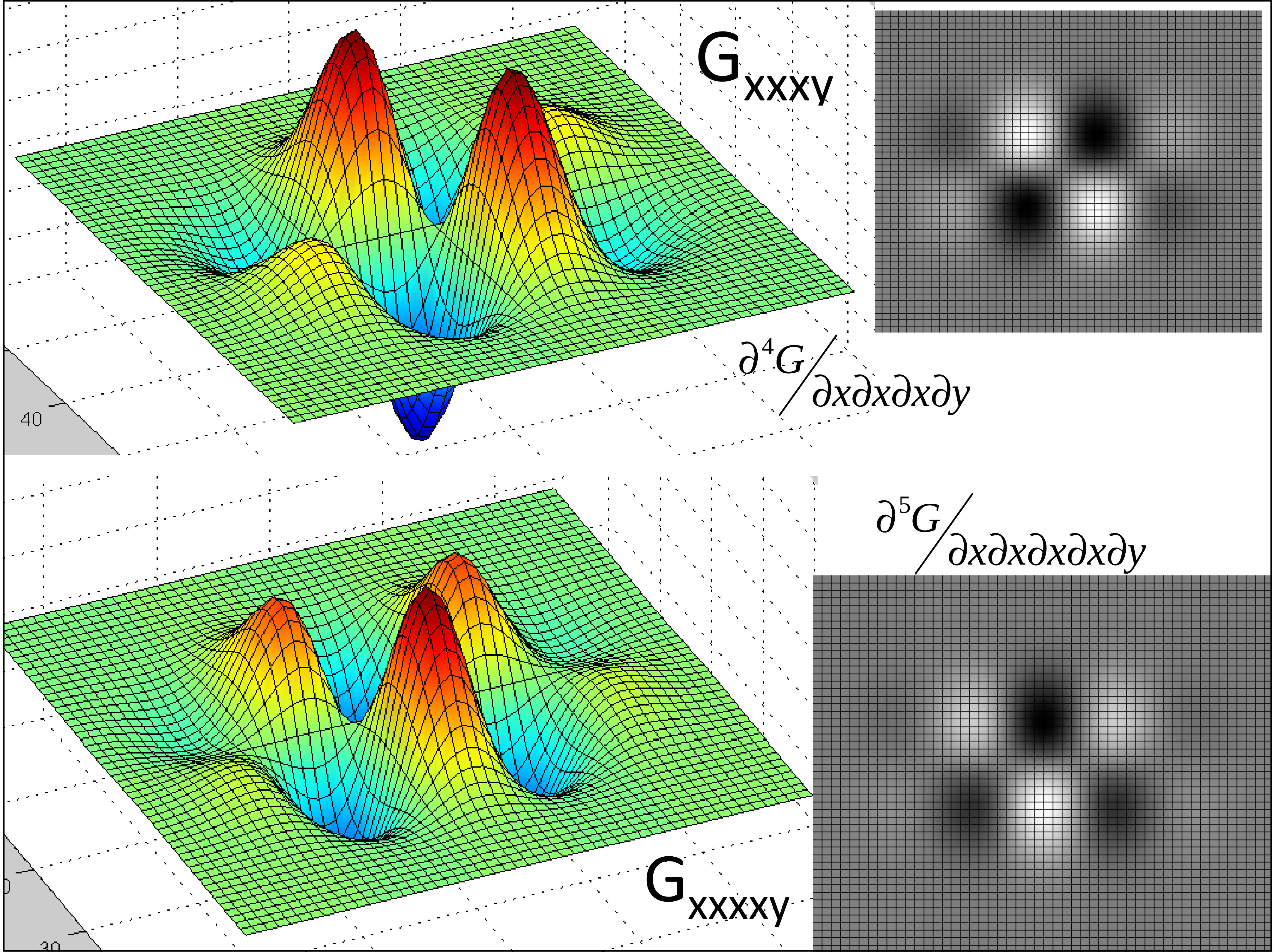
$$\partial^2 G \Big/ \partial x \partial y$$

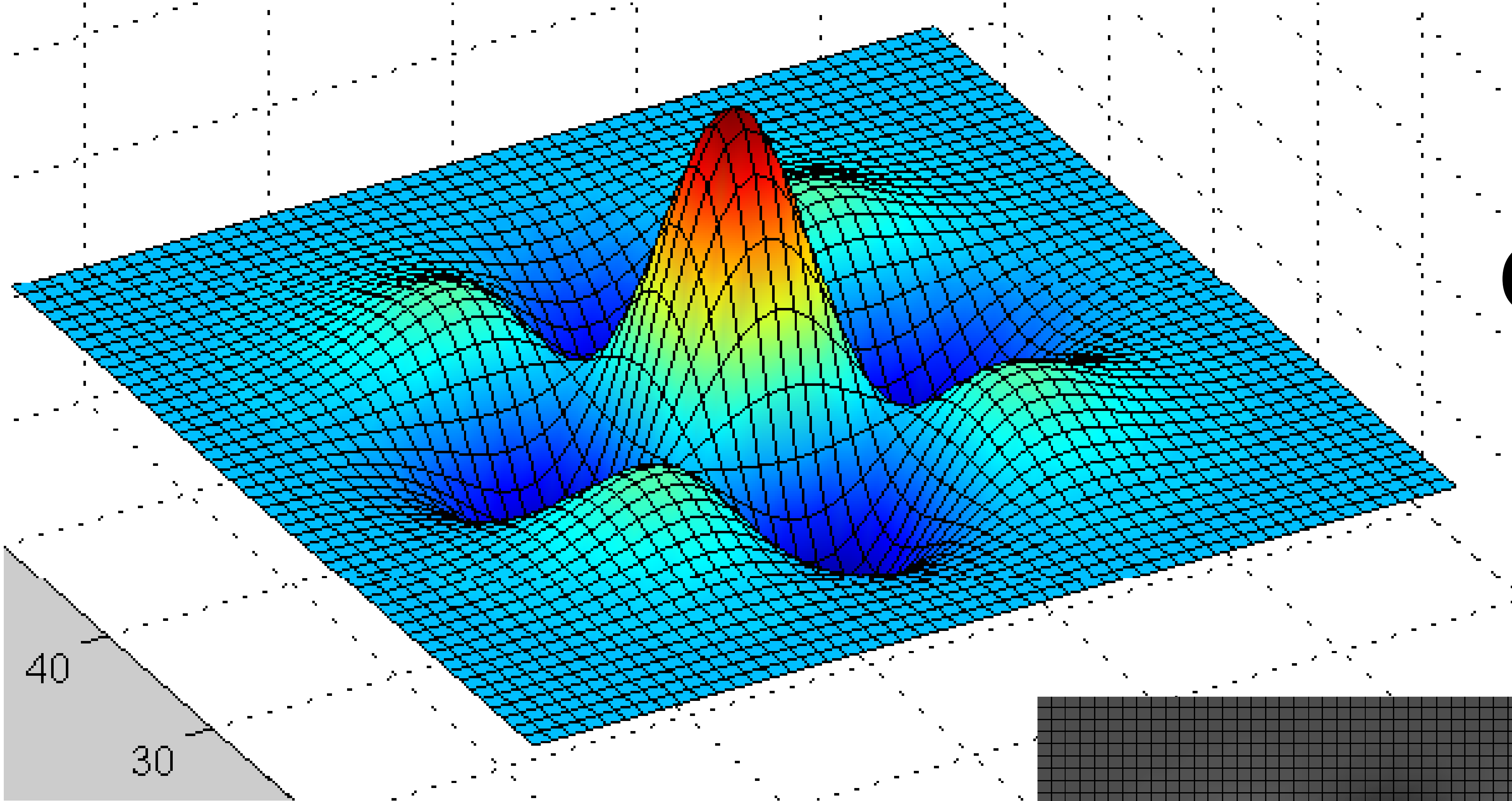
G_{xy}





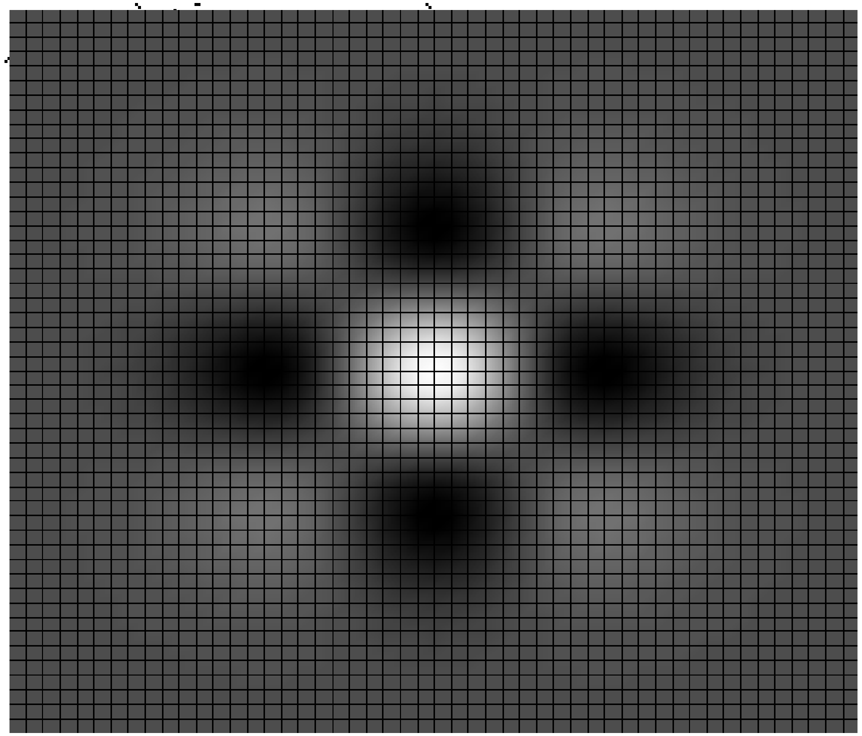
G_{xxy}
$\partial^3 G / \partial x \partial x \partial y$
$\partial^3 G / \partial x \partial y \partial y$
G_{xyy}


G_{xxxy}
$\partial^4 G / \partial x \partial x \partial x \partial y$
40
$\partial^5 G / \partial x \partial x \partial x \partial x \partial y$
G_{xxxxy}
30

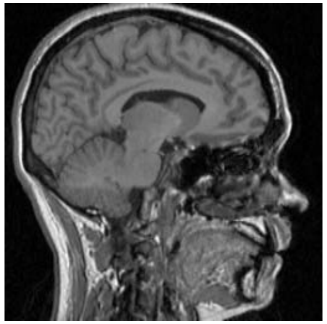


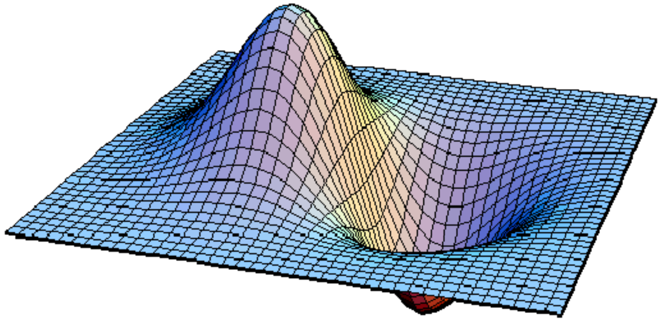
G_{xyxy}

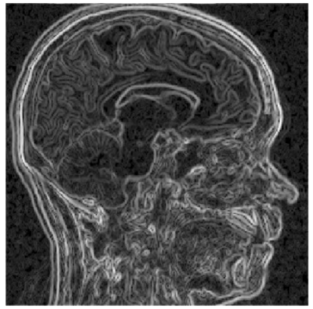
$$\partial^4 G \Big/ \partial x \partial y \partial x \partial y$$



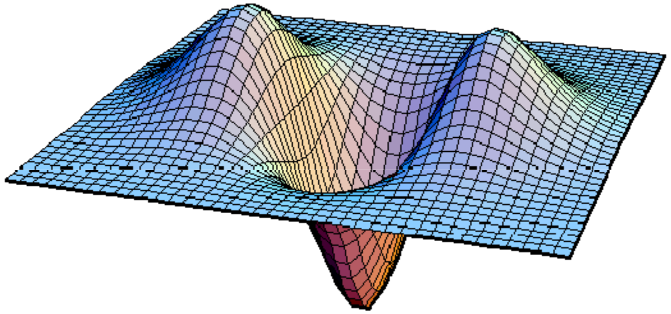
The visual system measures changes in place and time: derivatives

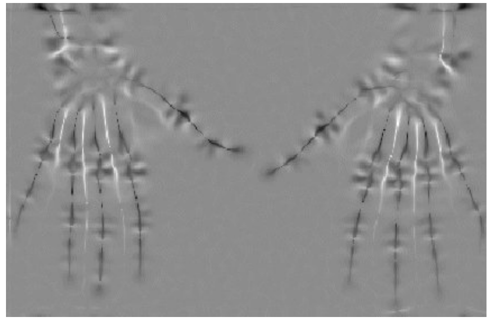




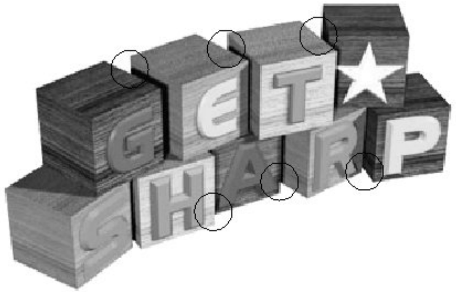


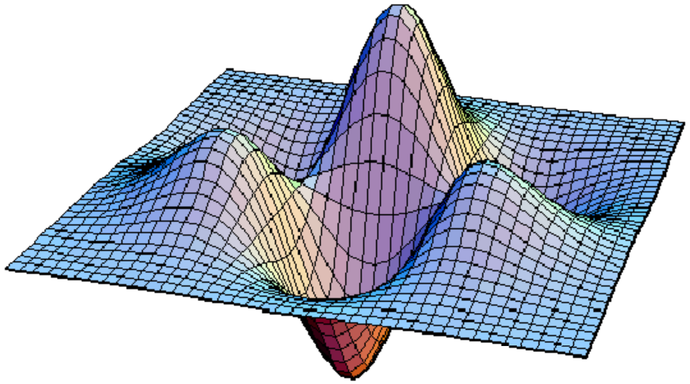
1st order

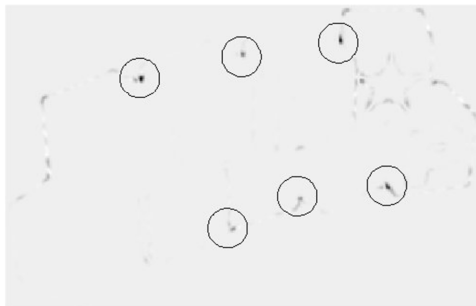




2nd order







3rd order

$$\frac{1}{(L_x^2+L_y^2)^3}\Big(-L_{xxy}\,L_y^5 + L_y^4\,(2\,L_{xy}^2 - L_x\,(L_{xxx} - 2\,L_{xyy}) + L_{xx}\,L_{yy}) +$$

$$L_x^4\,(2\,L_{xy}^2 - L_x\,L_{xyy} + L_{xx}\,L_{yy}) + L_x^2\,L_y^2\,(3\,L_{xx}^2 - 8\,L_{xy}^2 + L_x\,(-L_{xxx} + L_{xyy}) - 4\,L_{xx}\,L_{yy} + 3\,L_{yy}^2) +$$

$$L_x\,L_y^3\,(6\,L_{xy}\,(L_{xx} - L_{yy}) + L_x\,(L_{xxy} - L_{yyy})) + L_x^3\,L_y\,(6\,L_{xy}\,(-L_{xx} + L_{yy}) + L_x\,(2\,L_{xxy} - L_{yyy}))\Big)$$

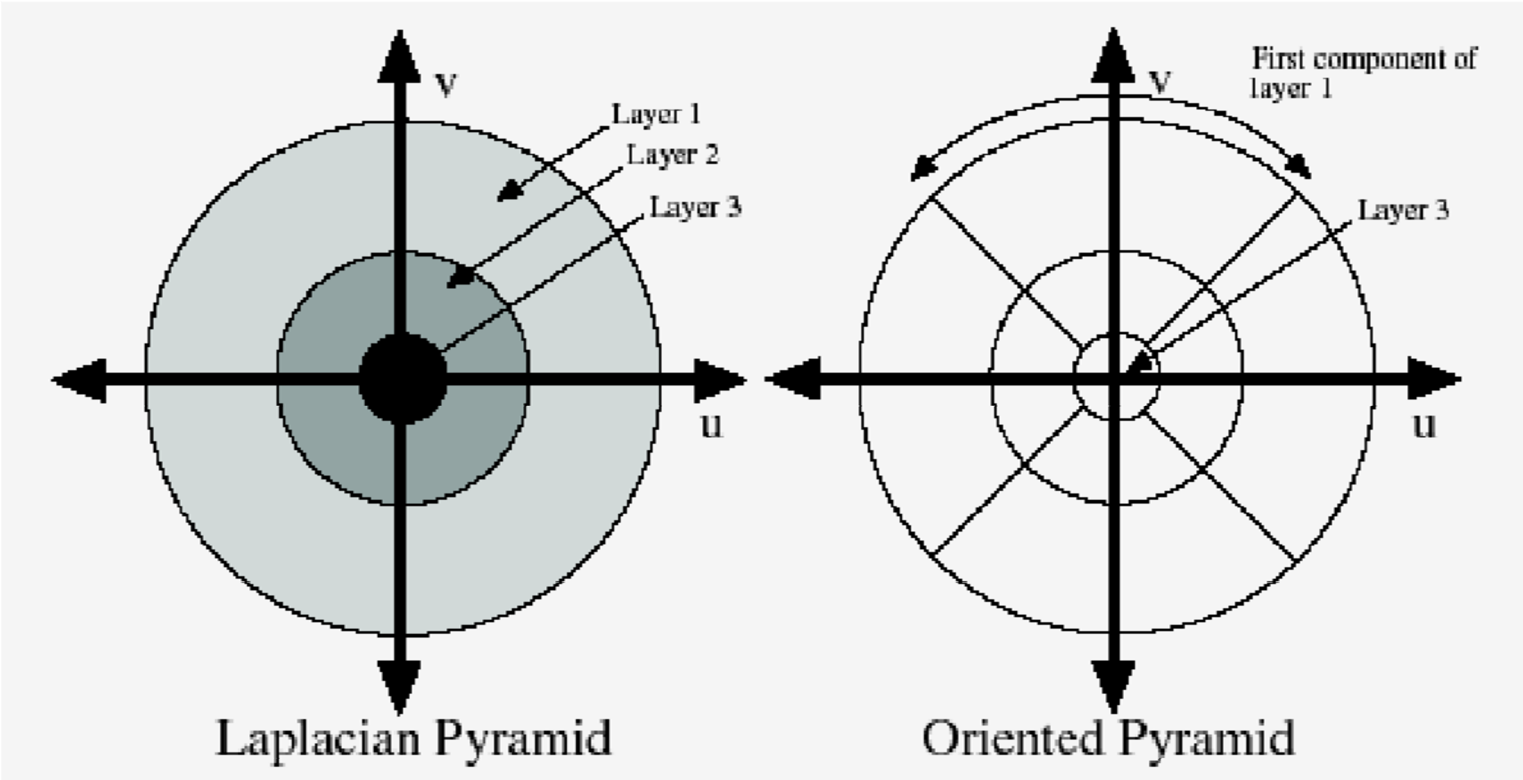
$$L_{x^\alpha y^\beta}(.,.;t) = \partial_{x^\alpha y^\beta}[L(.,.;t) * f(.,.)] = [\partial_{x^\alpha y^\beta}\{g(.,.;t)\}] * f(.,.)$$

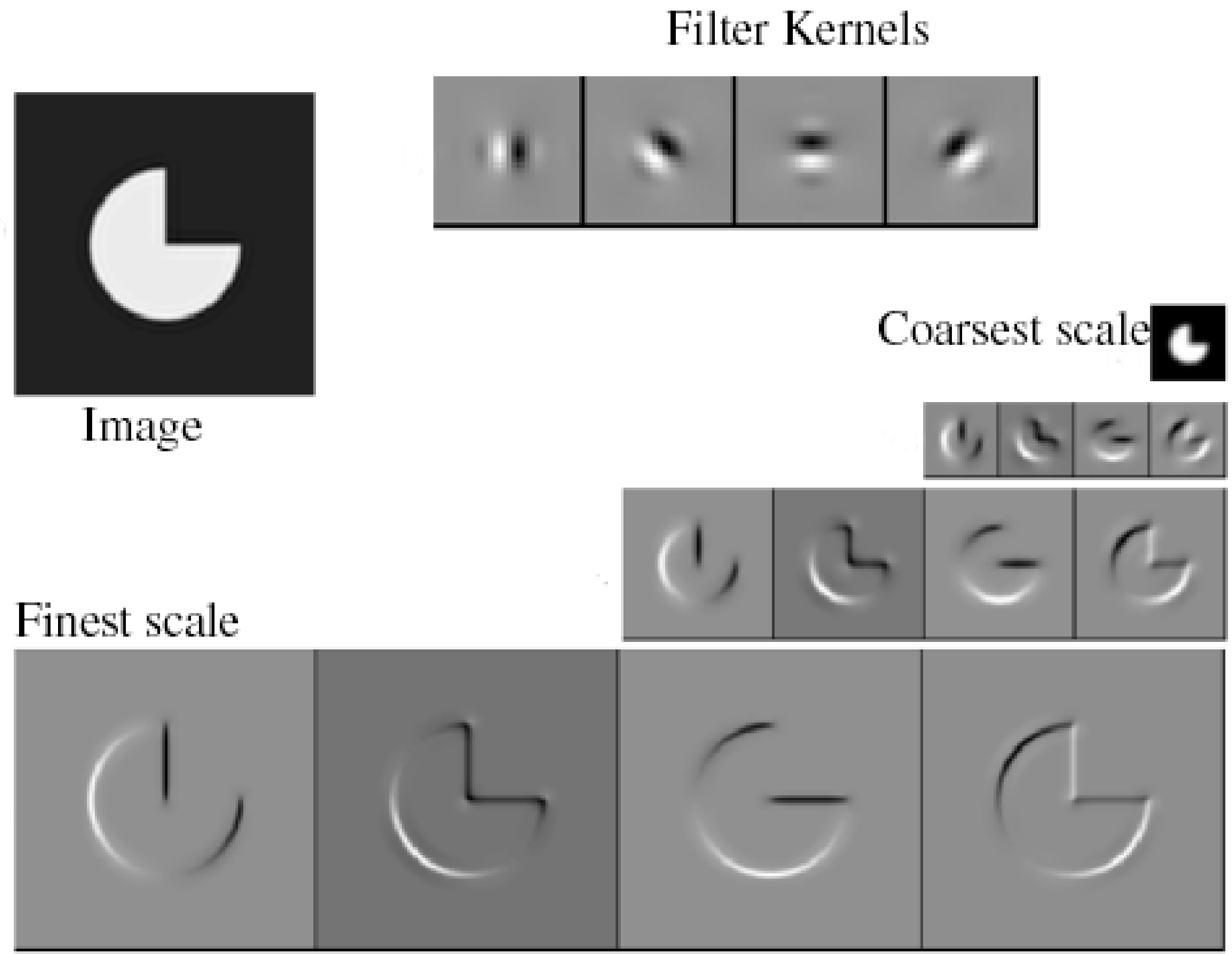
OPERATOR	Order	Purpose
Lx, Ly	**First**	**Gradient**
Lxx, Lxy, Lyy	**Second**	**Zero Crossing; Uniform Blobs; Ridges, Valleys.**
Lxxx, ,,,,	**Third**	**Corners, Ridges etc.**
$\nabla^2 L = L_{xx} + L_{yy}$	**2nd**	**ZCs**
Det_HL (DOH)	$= L_{xx}L_{yy} - L_{xy}^2$	**Saddles, Using extremas**
$\tilde{\kappa}(L)$	$= L_x^2 L_{yy} + L_y^2 L_{xx} - 2L_x L_y L_{xy}$	**Corner, using Rescaled level curvature**
Harris	**Det(μ) – κ.trace²(μ)**	**CORNER, using 2nd-moment structure tensor**

Oriented pyramids

- **Laplacian pyramid is orientation independent**
- **Apply an oriented filter to determine orientations at each layer**
  - **by clever filter design, we can simplify synthesis**
  - **this represents image information at a particular scale and orientation**

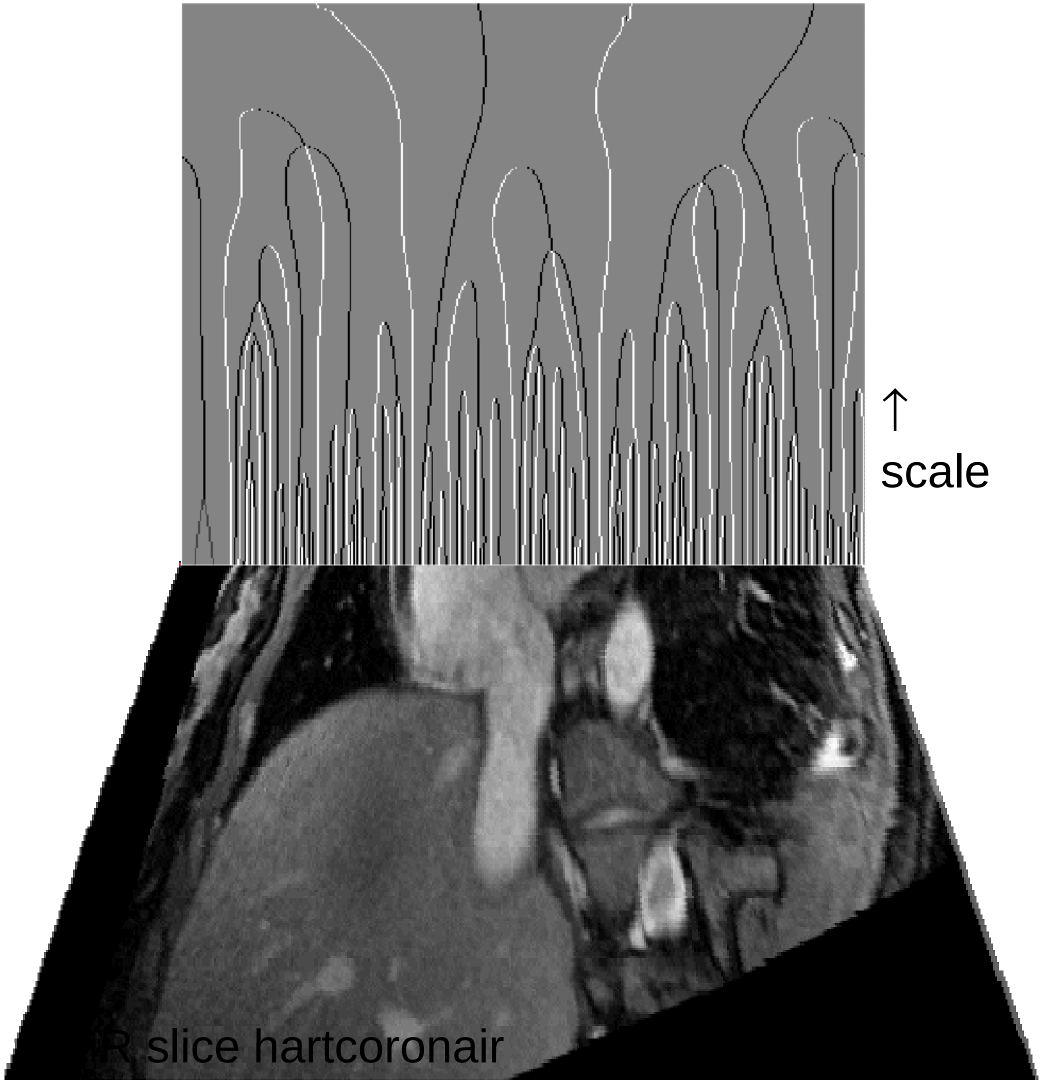
Oriented pyramids





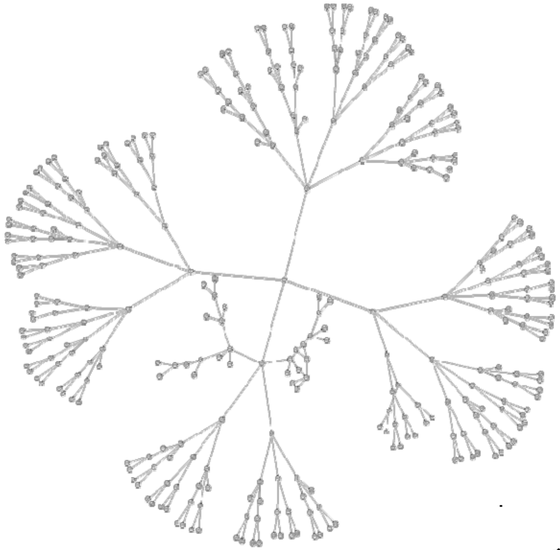
Reprinted from "Shiftable MultiScale Transforms," by Simoncelli et al., IEEE Transactions on Information Theory, 1992, copyright 1992, IEEE

- toppoints



↑
scale

- graph theory



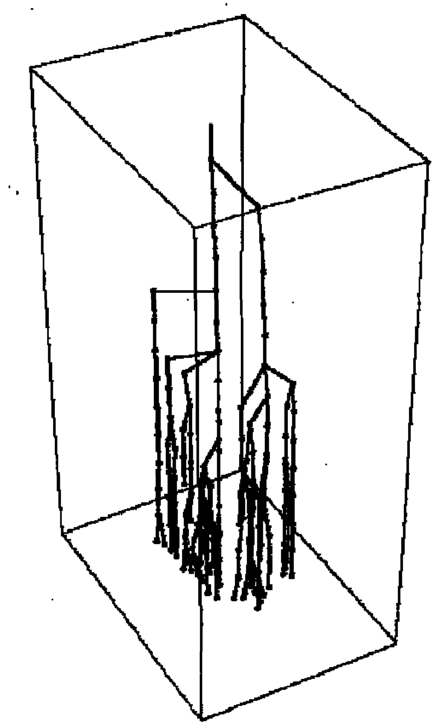
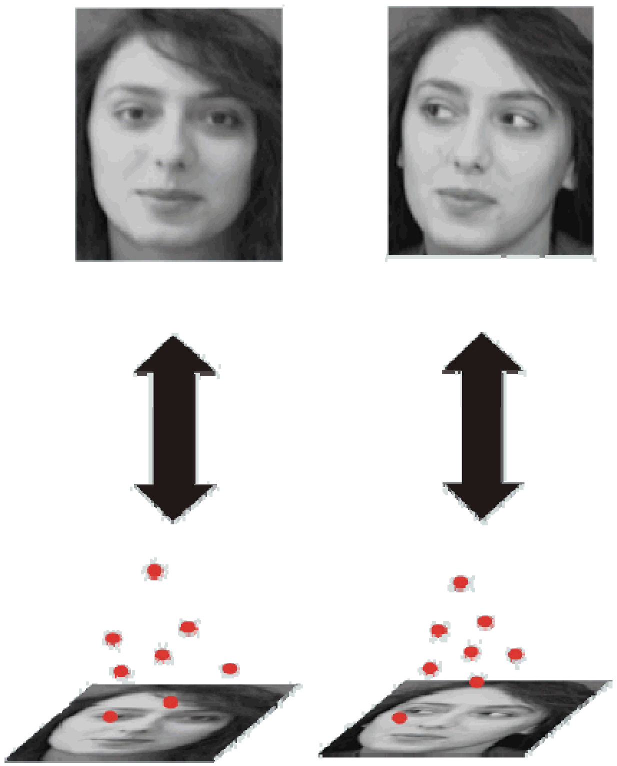


Image guided database retrieval



Point cloud matching (earth mover distance)

REFERENCES

Crowley, J, Riff O. Fast computation of scale normalised Gaussian receptive fields, Proc. Scale-Space'03, Isle of Skye, Scotland, Springer Lecture Notes in Computer Science, volume 2695, 2003.

Lindeberg, T. and Bretzner, L. Real-time scale selection in hybrid multi-scale representations, Proc. Scale-Space'03, Isle of Skye, Scotland, Springer Lecture Notes in Computer Science, volume 2695, pages 148-163, 2003.

Lowe, D. G., “Distinctive image features from scale-invariant keypoints”, International Journal of Computer Vision, 60, 2, pp. 91-110, 2004.

Burt, Peter and Adelson, Ted, "The Laplacian Pyramid as a Compact Image Code", IEEE Trans. Communications, 9:4, 532–540, 1983.

Crowley, J. and Parker, A.C, "A Representation for Shape Based on Peaks and Ridges in the Difference of Low Pass Transform", IEEE Transactions on PAMI, 6(2), pp 156-170, March 1984.

Crowley, J. L. and Sanderson, A. C. "Multiple resolution representation and probabilistic matching of 2-D gray-scale shape", IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(1), pp 113-121, 1987.

P. Meer, E. S. Baugher and A. Rosenfeld "Frequency domain analysis and synthesis of image generating kernels", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 9, pages 512-522, 1987.

Lindeberg, Tony, "Scale-space for discrete signals," PAMI(12), No. 3, March 1990, pp. 234-254.

REFERENCES

Lindeberg, T., Scale-Space Theory in Computer Vision, Kluwer Academic Publishers, 1994, ISBN 0-7923-9418-6

T. Lindeberg (1994). "Scale-space theory: A basic tool for analysing structures at different scales". Journal of Applied Statistics (Supplement on Advances in Applied Statistics: Statistics and Images: 2) 21 (2): pp. 224–270.

J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda, Uniqueness of the Gaussian kernel for scale-space filtering. IEEE Trans. Pattern Anal. Machine Intell. 8(1), 26–33, 1986.

A. Yuille, T.A. Poggio: Scaling theorems for zero crossings. IEEE Trans. Pattern Analysis & Machine Intelligence, Vol. PAMI-8, no. 1, pp. 15–25, Jan. 1986.

Lindeberg, Tony, "Principles for automatic scale selection", In: B. Jähne (et al., eds.), Handbook on Computer Vision and Applications, volume 2, pp 239--274, Academic Press, Boston, USA, 1999.

Lindeberg, T. and Garding, J.: Shape-adapted smoothing in estimation of 3-D depth cues from affine distortions of local 2-D structure, Image and Vision Computing, 15, pp 415–434, 1997.

Baumberg, A.: Reliable feature matching across widely separated views, Proc. Computer Vision Pattern Recognition, I:1774–1781, 2000.

Mikolajczyk, K. and Schmid, C.: Scale and affine invariant interest point detectors, Int. Journal of Computer Vision, 60:1, 63 - 86, 2004.

PETER J. BURT and EDWARD H. ADELSON; “The Laplacian Pyramid as a Compact Image Code”; IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-3l, NO. 4, p ??, APRIL 1983.

http://home.engineering.iastate.edu/~alexs/classes/575X_Spring_06/