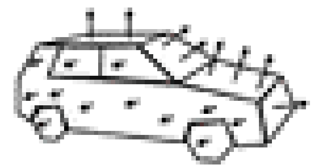
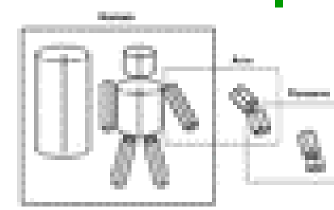
Lecture 6: Low-Level Processing and Feature Detection.

Reading Assignments:

Chapters 7 & 8.

Low-Level Processing

Remember: Vision as a change in representation.

1)  2)  3)  4) 

At the low-level, such change can be done by fairly streamlined mathematical transforms:

- Fourier transform
- Wavelet transform

these transforms yield a simpler but more organized image of the input.

Additional organization is obtained through multiscale representations.

Biological low-level processing

Edge detection and wavelet transforms in V1: hypercolumns and Jets.

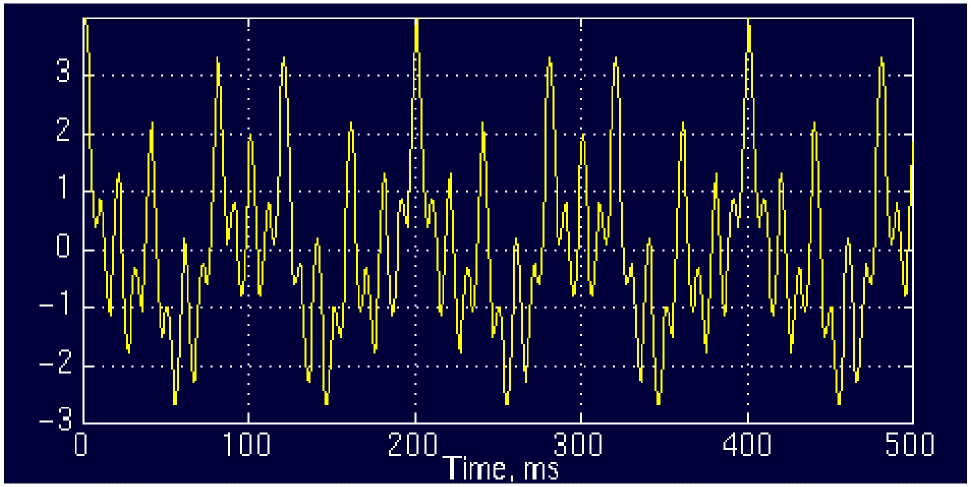
but…

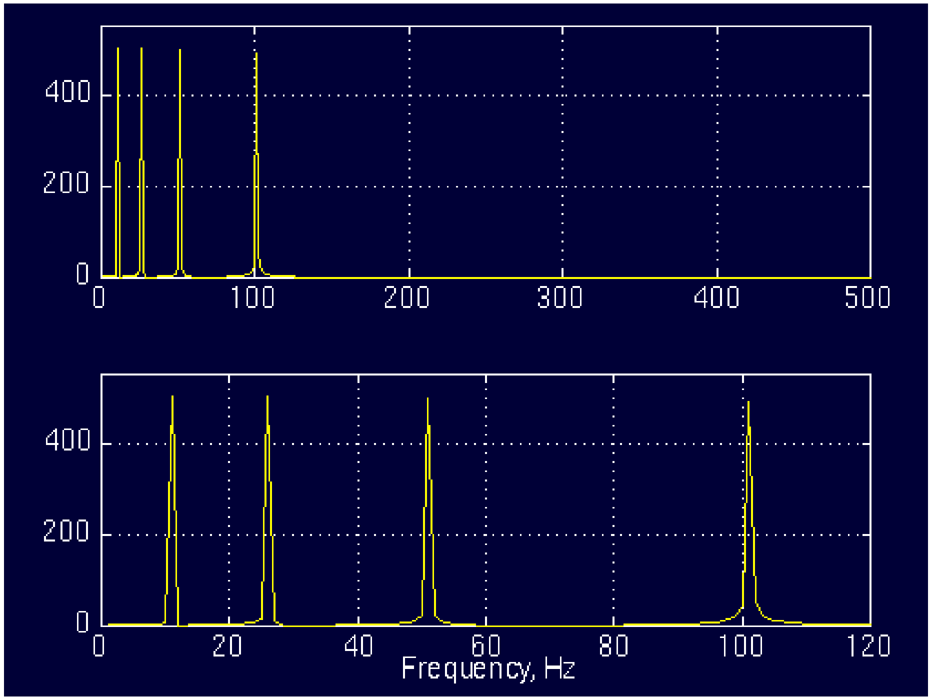
- processing appears highly non-linear, hence convolution of input by wavelets only approximates real responses;

- neuronal responses are influenced by context, i.e., neuronal activity at one location depends on activity at possibly distant locations;

- responses at one level of processing (e.g., V1) also depend on feedback from higher levels, and other modulatory effects such as attention, training, etc.

Fourier Transform



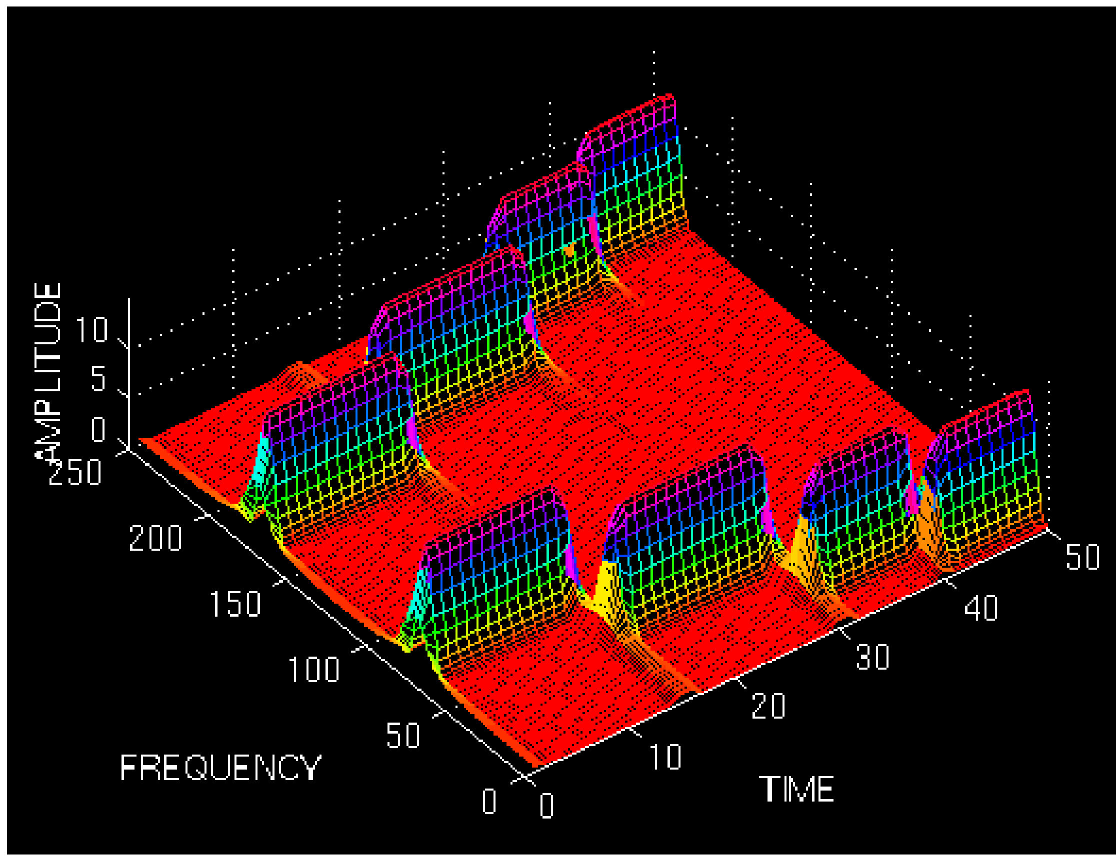


Problem

- The Fourier transform does not intuitively encode non-stationary (i.e., time-varying) signals.

- One solution is to use the short-term Fourier transform, and repeat for successive time slices.

- Another is to use a wavelet transform.

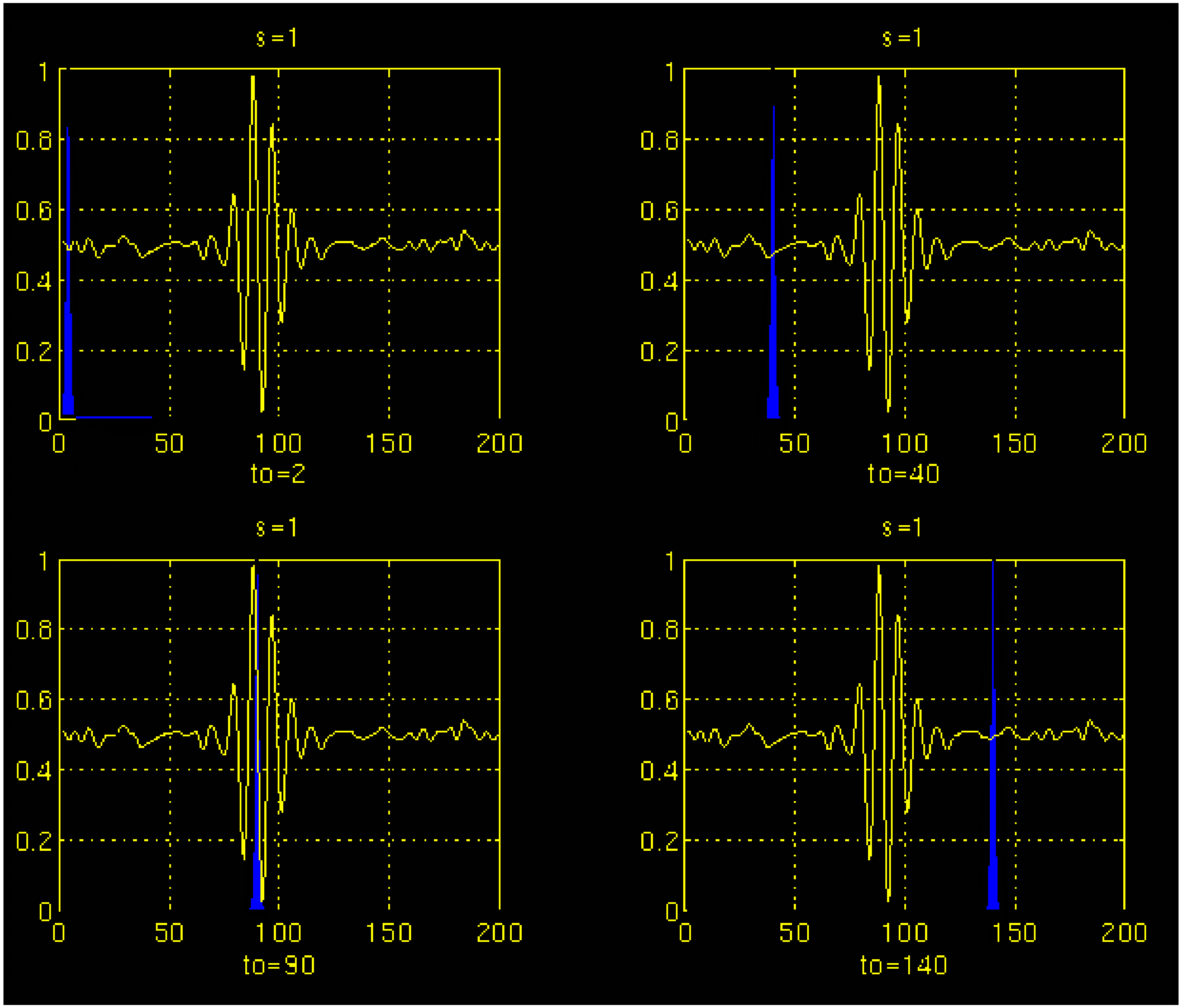


Wavelet Transform

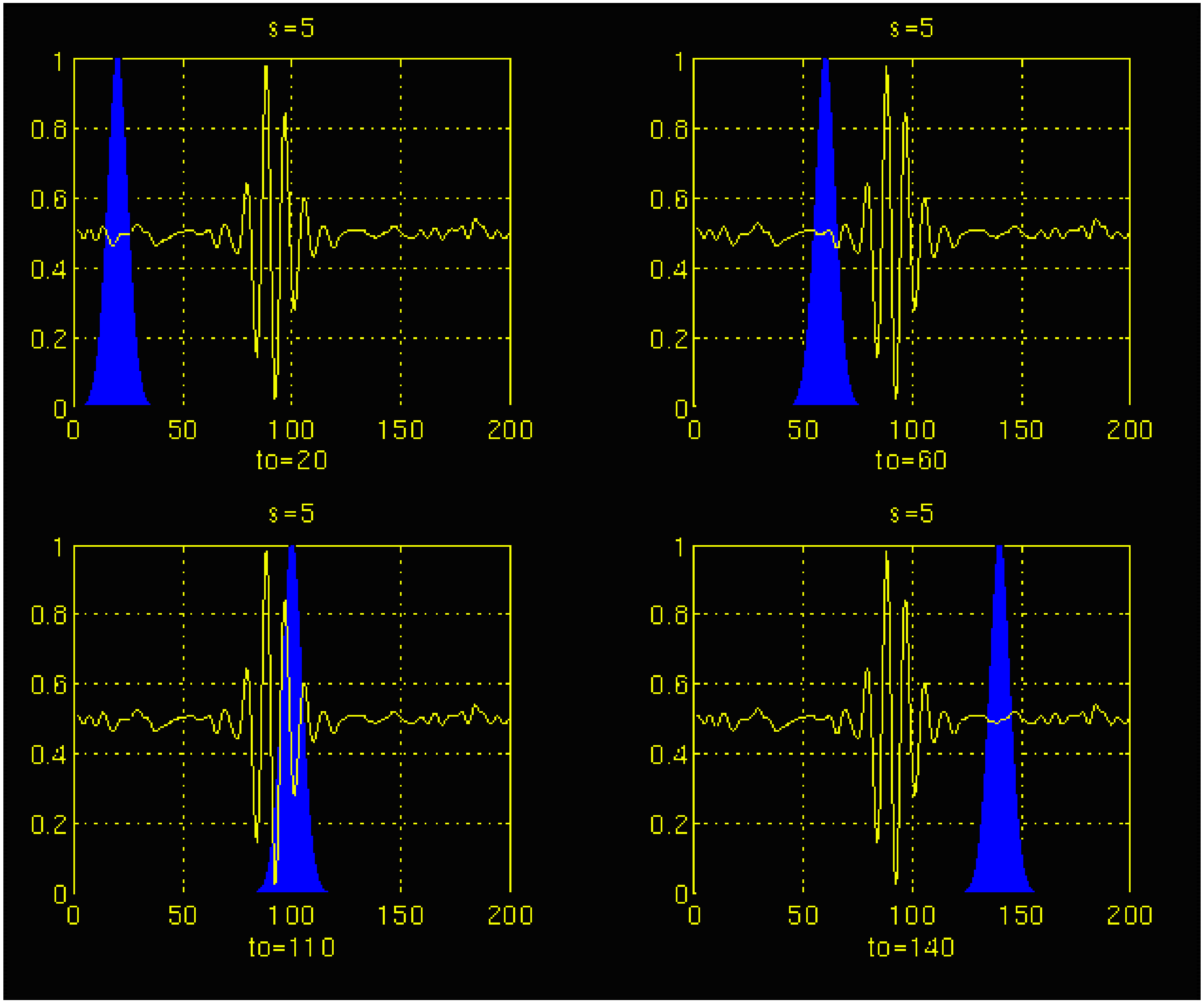
- Mother wavelet ψ: defines shape and size of window
- Convolved with signal (x) after translation (τ) and scaling (s)
- Results stored in array indexed by translation and scaling

$$CWT_x^{\psi}(\tau, s) = \Psi_x^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\psi^* \left(\frac{t-\tau}{s}\right) dt$$

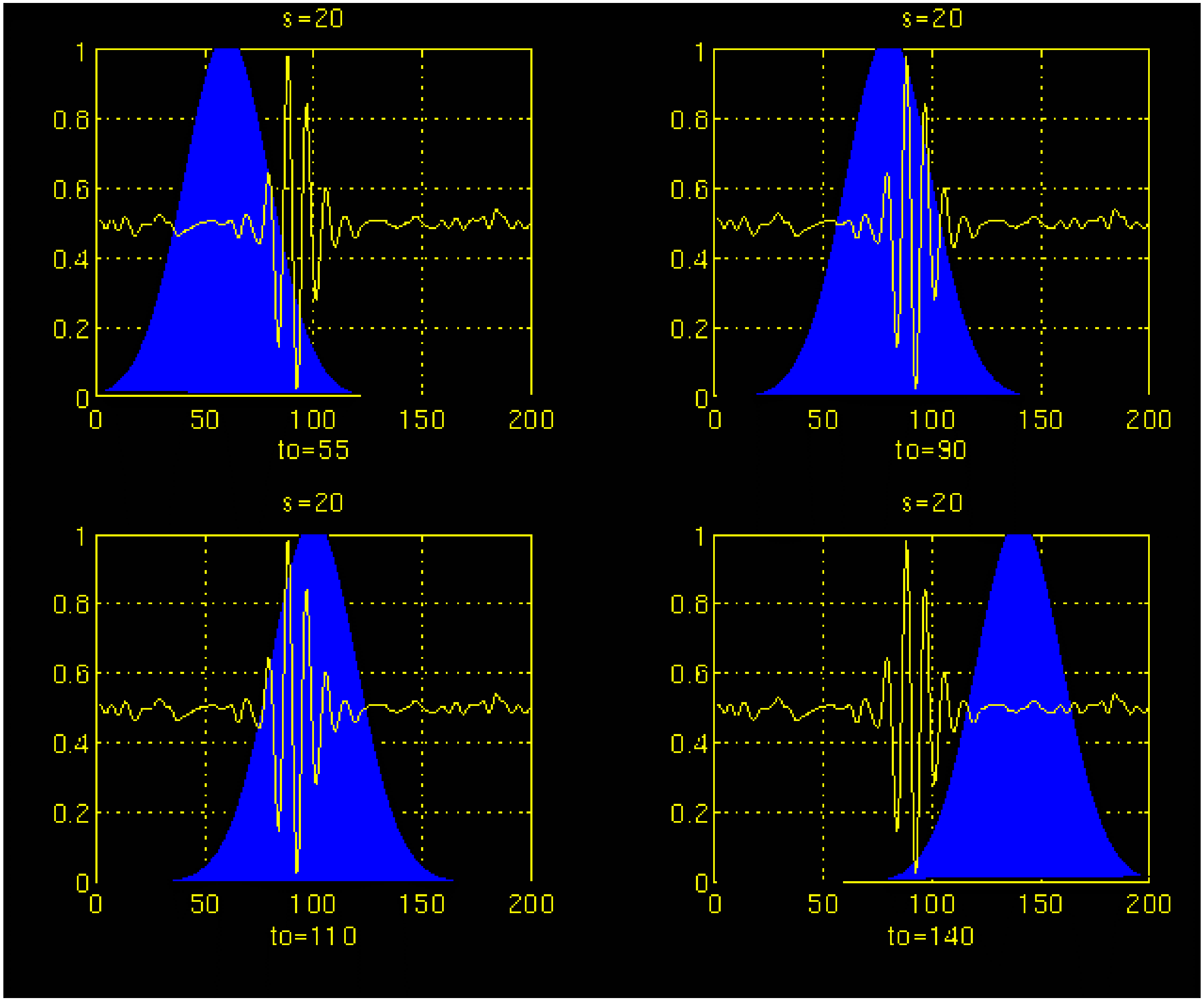
Example: small-scale wavelet is applied



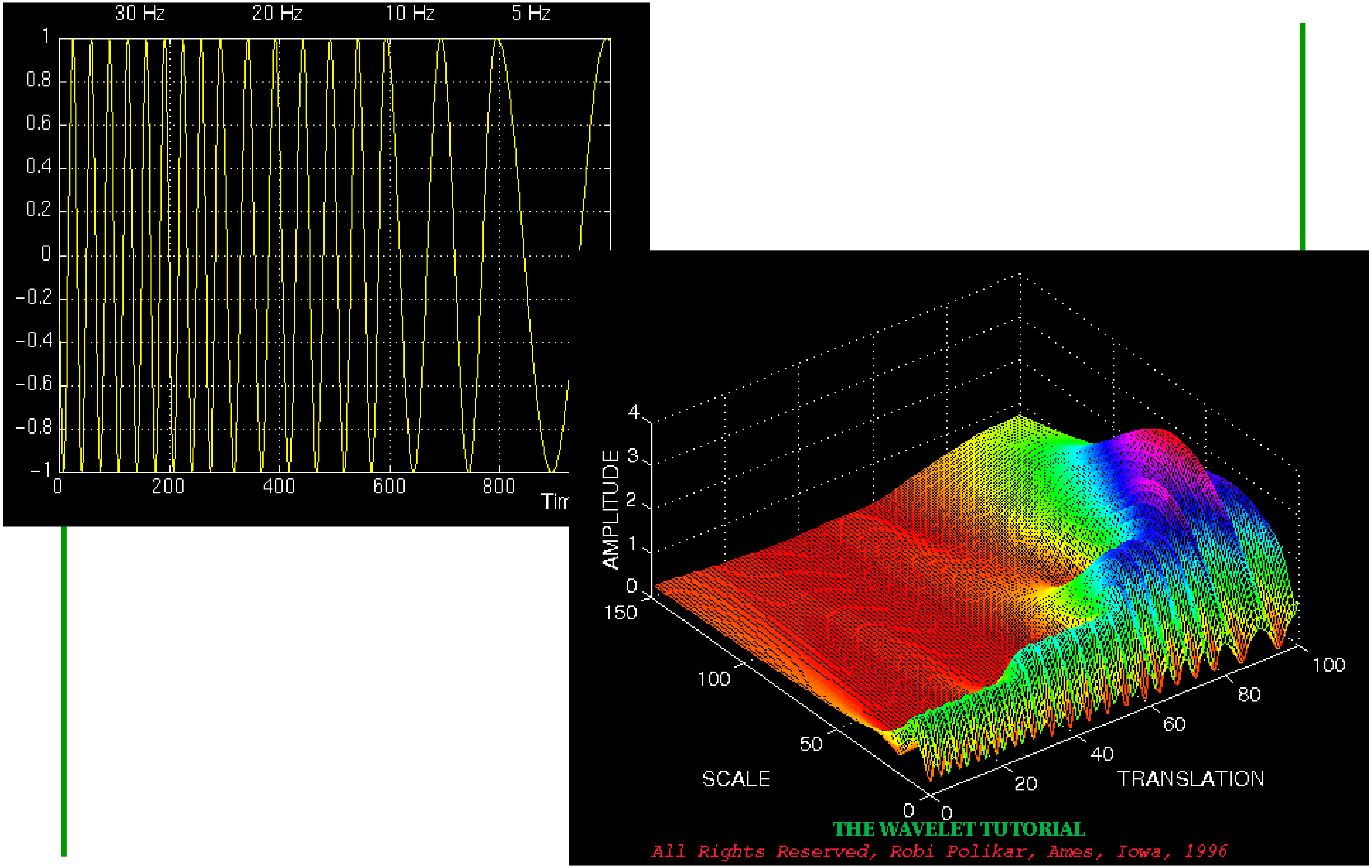
then larger-scale…



and even larger scale…



Result is indexed by translation & scale



Wavelet Transform & Basis Decomposition

We define the inner product between two functions:

$$< f(t), g(t) >= \int_a^b f(t).g^*(t)\, dt$$

then the continuous wavelet transform:

$$CWT_x^{\psi}(\tau, s) = \Psi_x^{\psi}(\tau, s) = \int x(t).\psi_{\tau,s}^*(t)\, dt$$

can be thought of as taking the inner product between signal and all of the different wavelets (parametrized by translation & scale):

$$\psi_{\tau,s} = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right)$$

Orthonormal

two functions are orthogonal iff:

$$< f(t), g(t) >= \int_a^b f(t).g^*(t)\,dt = 0$$

and a set of functions is orthonormal iff:

$$\int_a^b \phi_k(t)\phi_l^*(t)\,dt = \delta_{kl}$$

with

$$\delta_{kl} = \begin{cases} 1 & \text{if } k = l \\ 0 & \text{if } k \neq l \end{cases}$$

Basis

If the collection of wavelets forms an **orthonormal basis**, then we can compute:

$$\mu_k = < f, \phi_k > = \int f(t).\phi_k^*(t)\, dt$$

and fully reconstruct the signal from those coefficients (and knowledge of the wavelet functions) alone:

$$\begin{aligned} f(t) &= \sum_k \mu_k \phi_k(t) \\ &= \sum_k < f, \phi_k > \phi_k(t) \end{aligned}$$

thus the transformation is reversible.

Edge Detection

Very important to both biological and computer vision:

- Easy and cheap (computationally) to compute.

- Provide strong visual clues to help recognition.

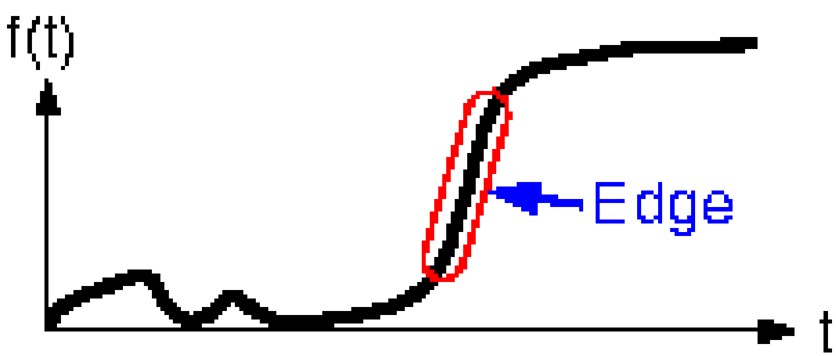
- Problem: sensitive to image noise.

why? because edge detection is a high-pass filtering process and noise typically has high-pass components (e.g., speckle noise).

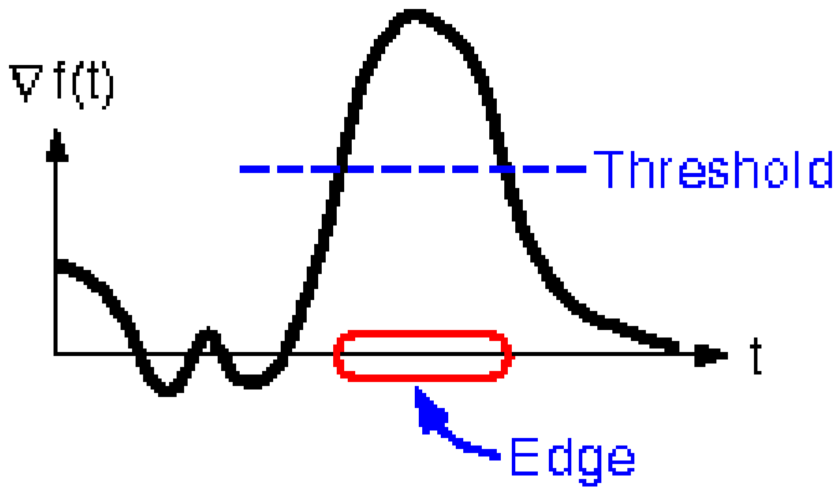
Laplacian Edge Detection

Edges are defined as zero-crossings of the second derivative (Laplacian if more than one-dimensional) of the signal.

This is very sensitive to image noise; thus typically we first blur the image to reduce noise. We then use a Laplacian-of-Gaussian filter to extract edges.



Smoothed signal



First derivative (gradient)

Derivatives in 2D

Gradient:

$$\frac{\partial f(x,y)}{\partial x} = \Delta_x = \frac{f(x+d_x,y)-f(x,y)}{d_x},$$
$$\frac{\partial f(x,y)}{\partial y} = \Delta_y = \frac{f(x,y+d_y)-f(x,y)}{d_y}, \qquad (42)$$

for discrete images:

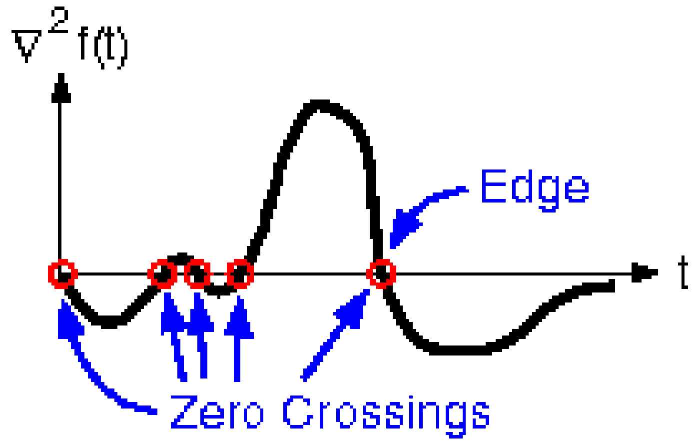
$$\Delta_x = f(i+1,j)-f(i,j),$$
$$\Delta_y = f(i,j+1)-f(i,j). \qquad (43)$$

magnitude and direction:

$$M = \sqrt{\Delta_x^2 + \Delta_y^2}, \qquad \theta = \tan^{-1}\left[\frac{\Delta_y}{\Delta_x}\right].$$

Laplacian-of-Gaussian

Laplacian:

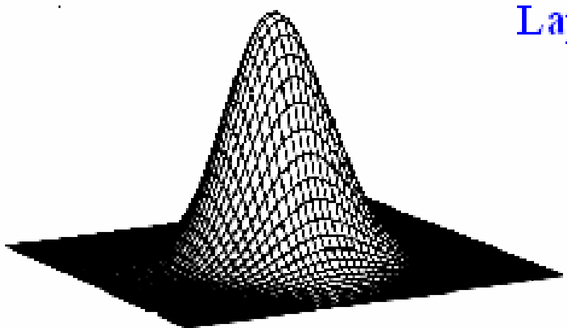


$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}. \quad (47)$$

Laplacian of the Gaussian

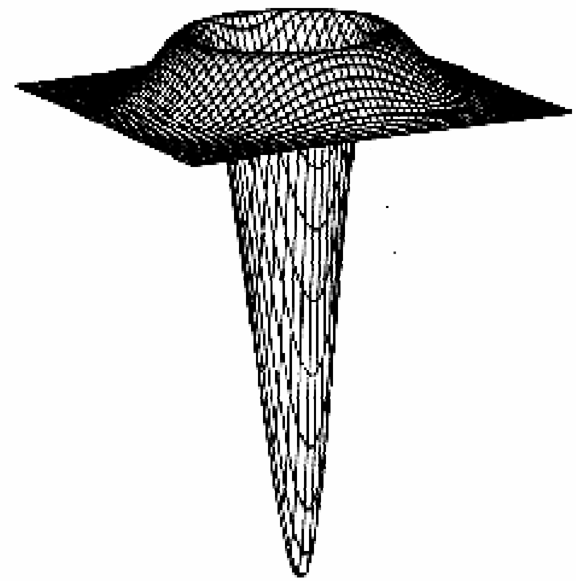
Idea: Rather than smooth image first using a Gaussian & then computing the Laplacian of the result, we can derive a mask that does both operations simultaneously

$\boxed{G(r,c)}$



Laplacian of Gaussian (L.O.G.)

$$\boxed{\nabla^2 G(r,c) = \frac{\partial^2 G}{\partial^2 r} + \frac{\partial^2 G}{\partial^2 c}}$$



Note similarity to 3x3 Laplacian mask

0	1	0
1	-4	1
0	1	0

L.O.G. mask computed by sampling the L.O.G. function on a discrete grid (mask size chosen to be identical to that of the corresponding Gaussian mask)

Another Edge Detection Scheme

- Maxima of the modulus of the Gradient in the Gradient direction (Canny-Deriche):

    - use optimal 1st derivative filter to estimate edges
    - estimate noise level from RMS of 2nd derivative of filter responses
    - determine two thresholds, Thigh and Tlow from the noise estimate
    - edges are points which are locally maximum in gradient direction
    - a hysteresis process is employed to complete edges, i.e.,
        - edge will start when filter response > Thigh
        - but may continue as long as filter response > Tlow

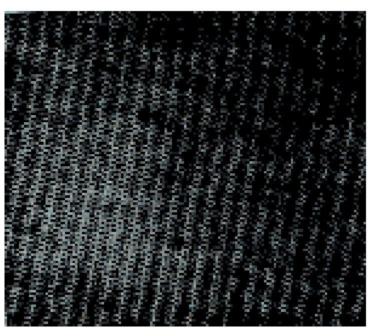
Marr-Hildreth Edge Detection Algorithm

Given: An input image I and a value for the variance σ

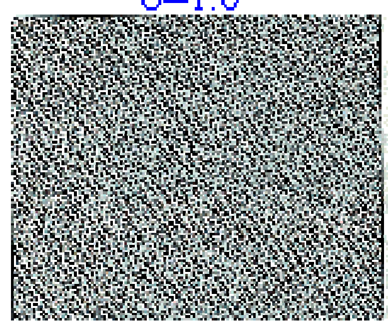
Steps:

1. Compute the L.O.G. mask corresponding to the desired value of σ
2. Apply the L.O.G. mask to image I
3. Compute zero crossings of the result
4. Label as edges all zero crossing pixels
5. Optional step: label as edges only those zero-crossings whose transition from + to - or from - to + is greater than a threshold

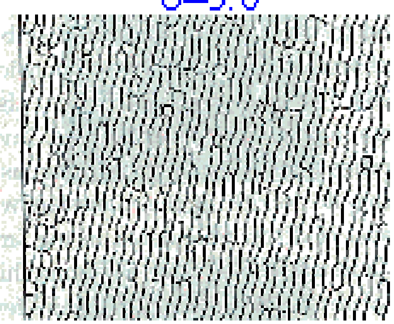
Input image



zero crossings σ=1.0

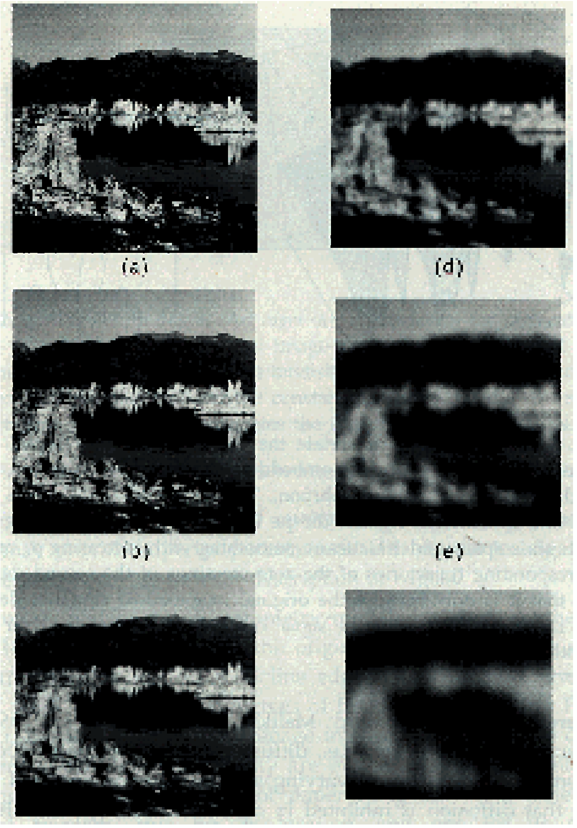


zero crossings σ=5.0



Choosing the "Right" Scale??

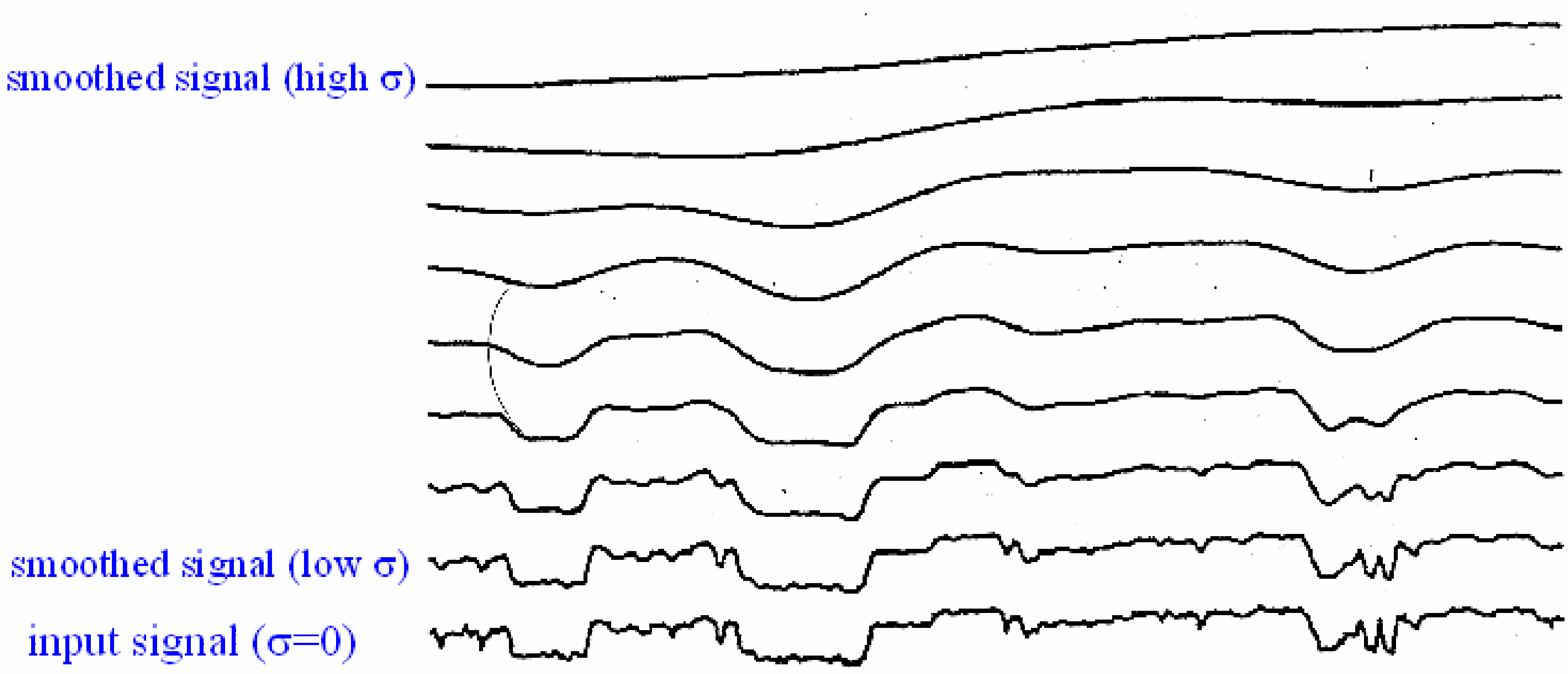
Rather than trying to decide which scale to use, we can process images at all scales & examine how the resulting image changes with increasing values of σ



Basic principles of scale-space approaches

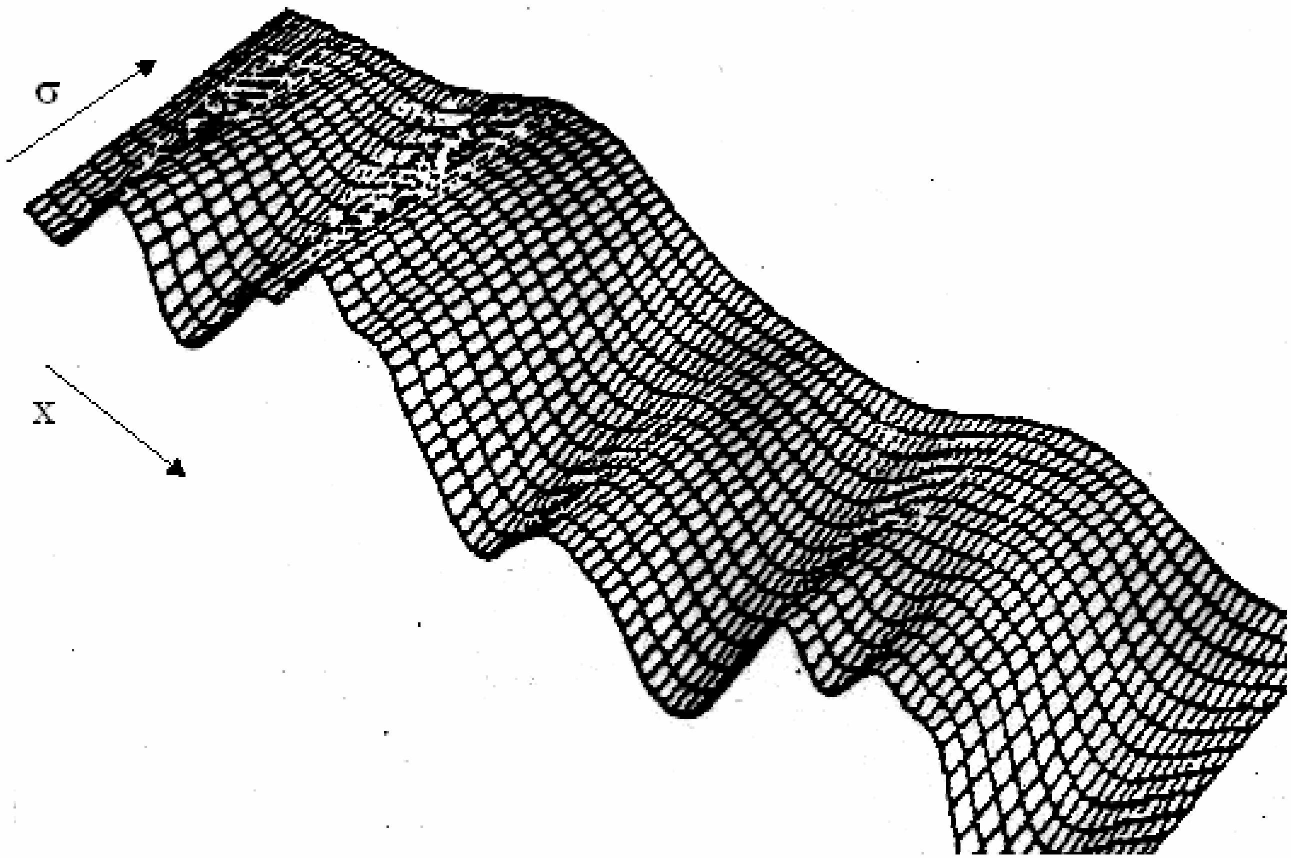
- No single scale is "special"
- Treat s as a continuous parameter
- As s increases, we should obtain "simpler" images, i.e., no "new structures" should be created
- "Structures" that remain present for a broad range of scales signify "important" features in the image

Scale Space in 1D



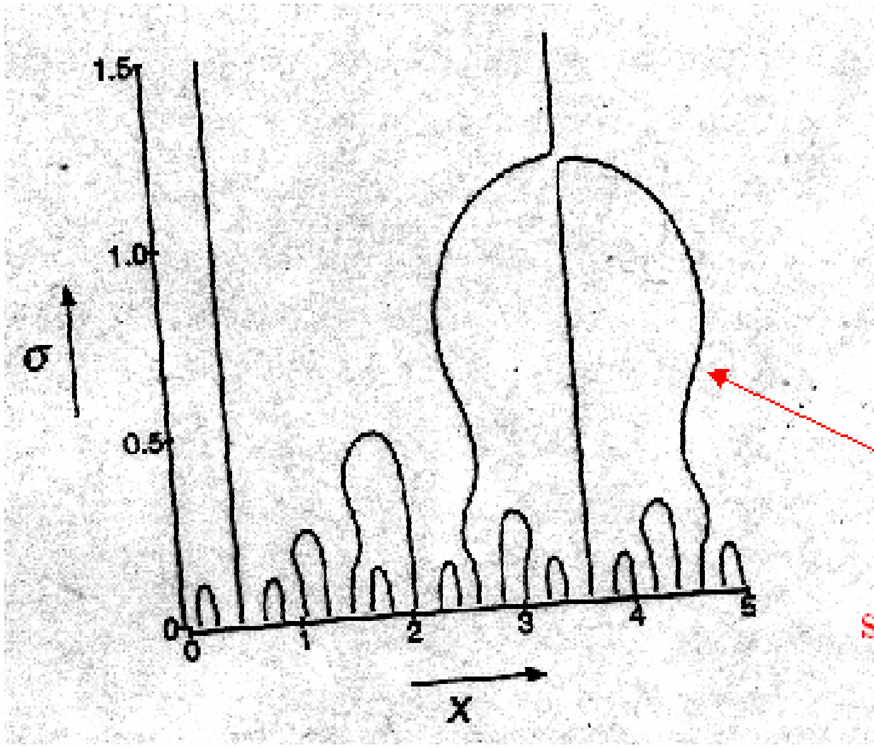
Scale Space in 1D

Resulting Scale-Space Surface



Scale Space in 1D

Idea: As the 1D image gets increasingly smoothed, the position of zero crossings moves along the smoothed curve, creating trajectory curves



Trajectory of a single zero-crossing

- For the Gaussian function, trajectories form nested, upside-down U's
- The Gaussian smoothing function is the only function that ensures no new zero-crossings are created as σ increases
- Therefore, it is the only function for which zero crossing trajectories have the nesting property

Scale Space in 2D

- Zero crossing contours form surfaces in (r,c)-σ space
- No new zero-crossings created as σ increases

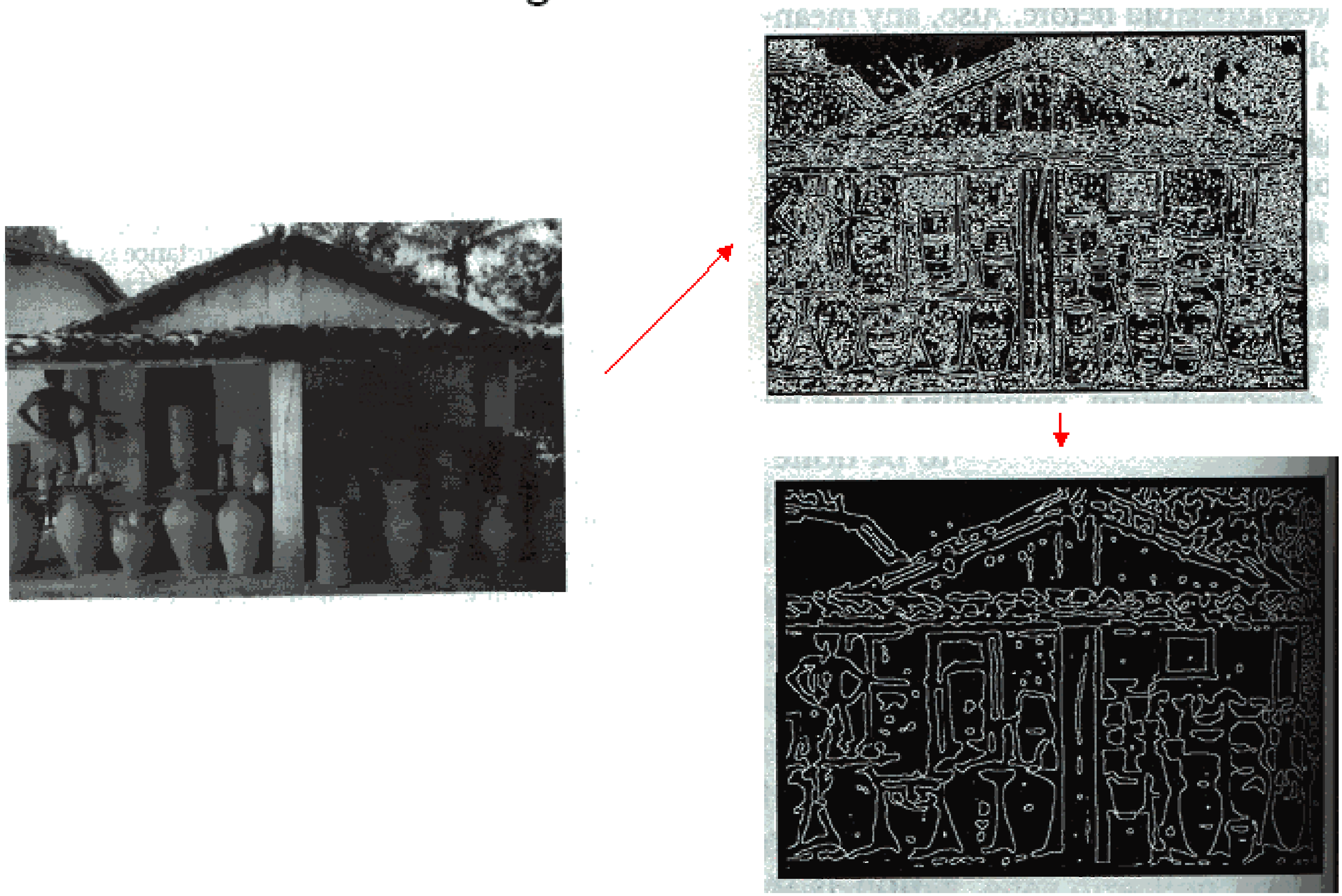
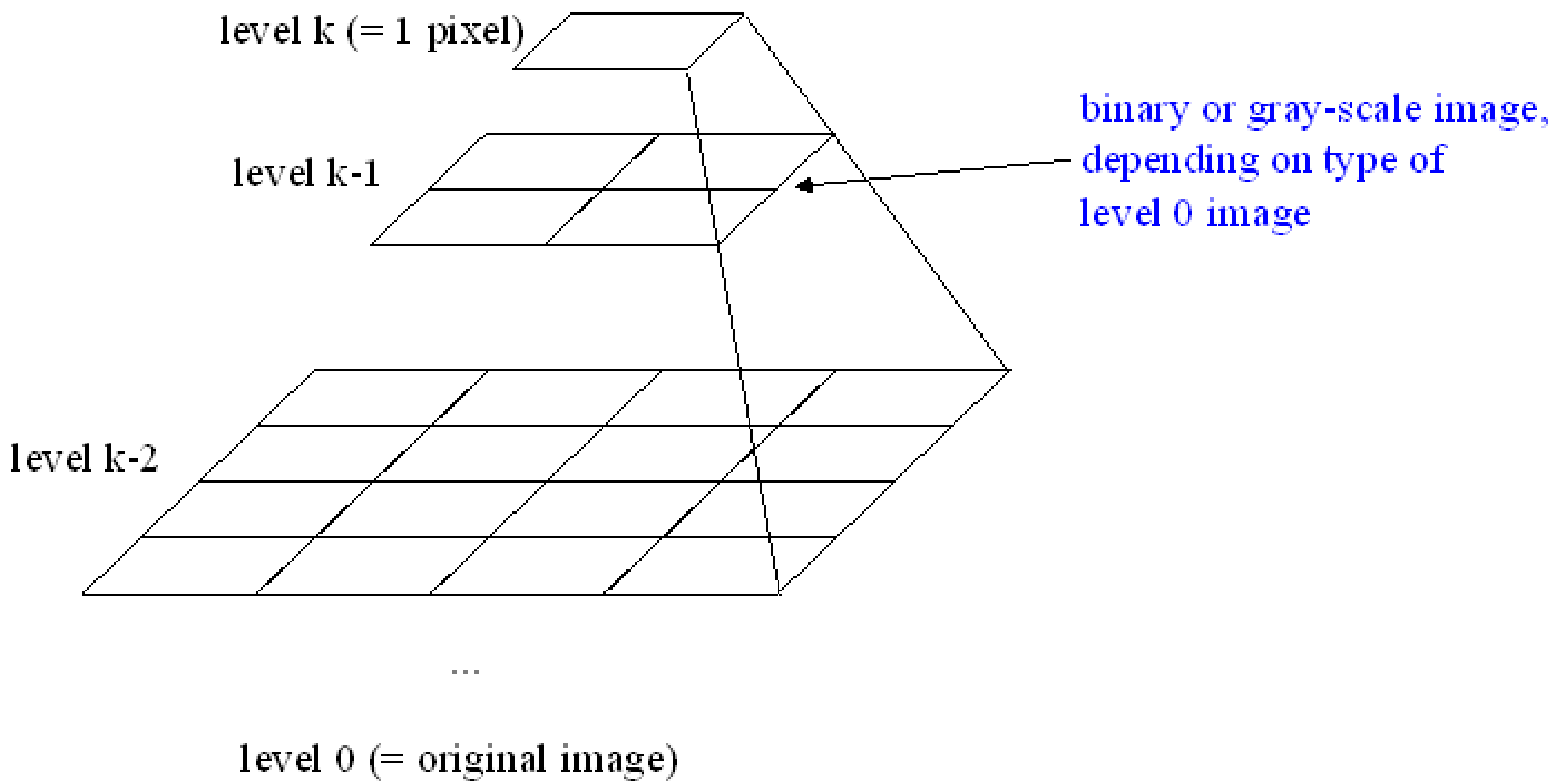


Image Pyramids

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



Quad Tree: A Binary Image Pyramid

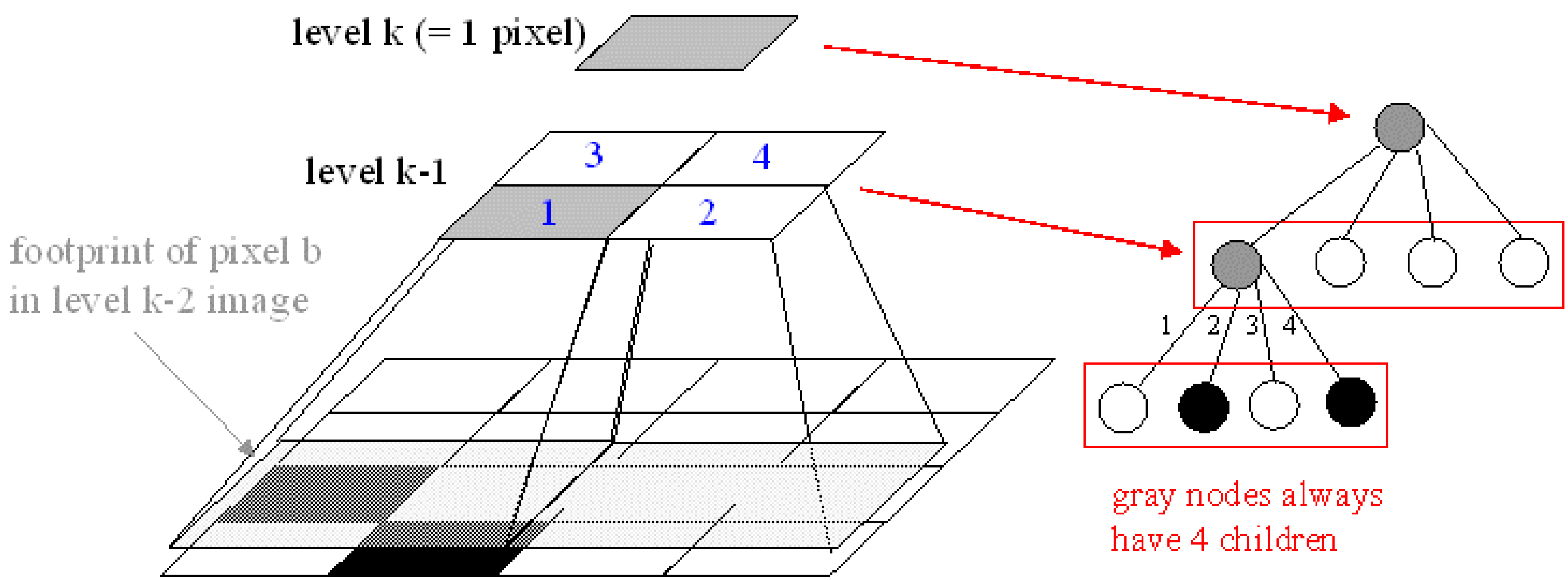
Idea---Represent binary image using a tree data structure:

- root = level k pixel
- 3 possible values for a node: white, black, gray

○ = all pixels in node's footprint are white

● = all pixels in node's footprint are black

◐ = node's footprint has black & white pixels

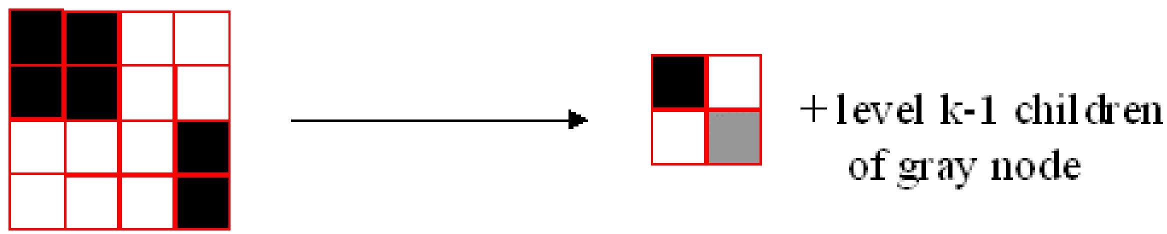


Building A Quad Tree

Two basic image operations need to be implemented:

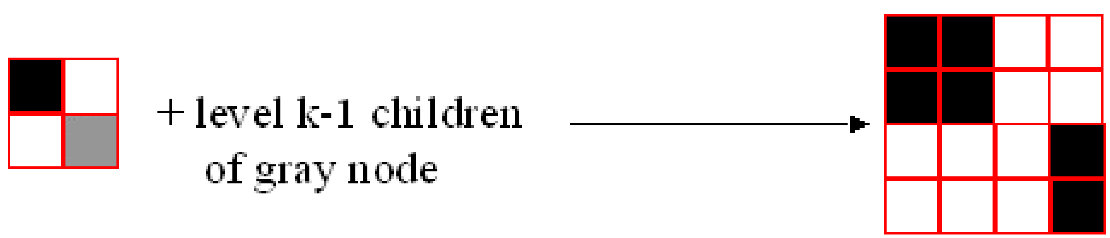
- REDUCE operation:

  Map a level k-1 image (hi-res) to a level k image (low-res)



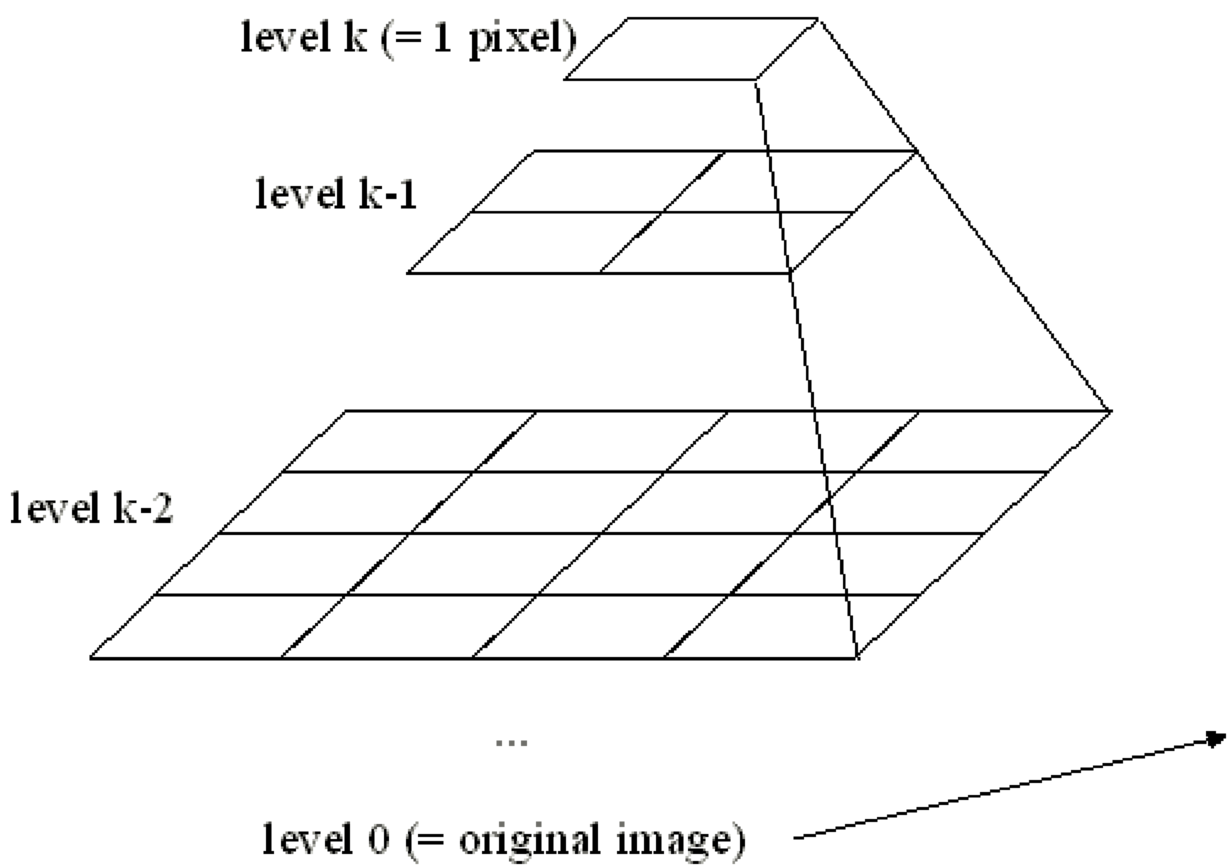
- EXPAND operation

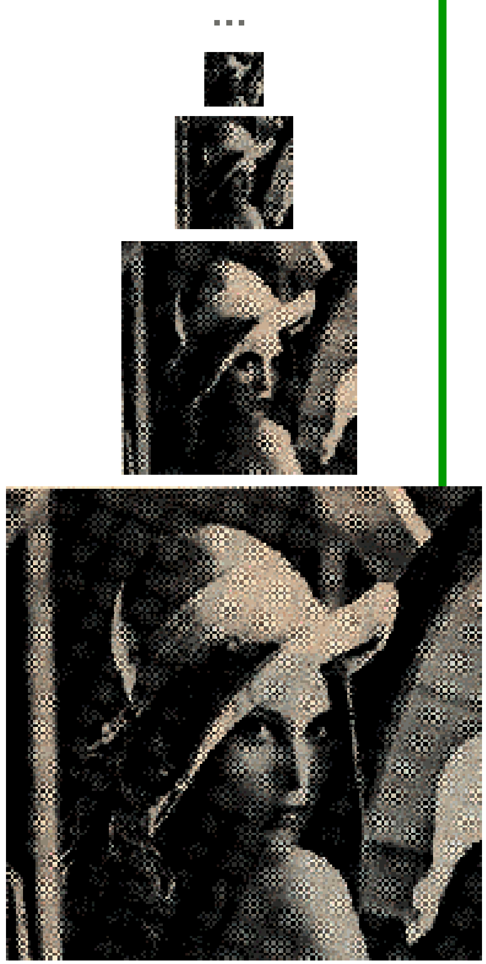
  Map a level k image (low-res) to a level k-1 image (hi-res)



Gaussian Pyramid

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





Building A Gaussian Pyramid

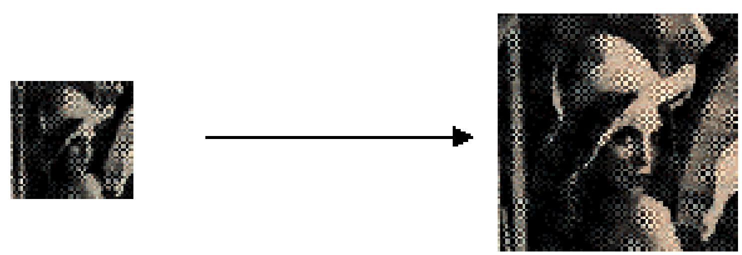
Two basic image operations need to be specified:

- REDUCE operation:

  Map a level k-1 image (hi-res) to a level k image (low-res)

- EXPAND operation

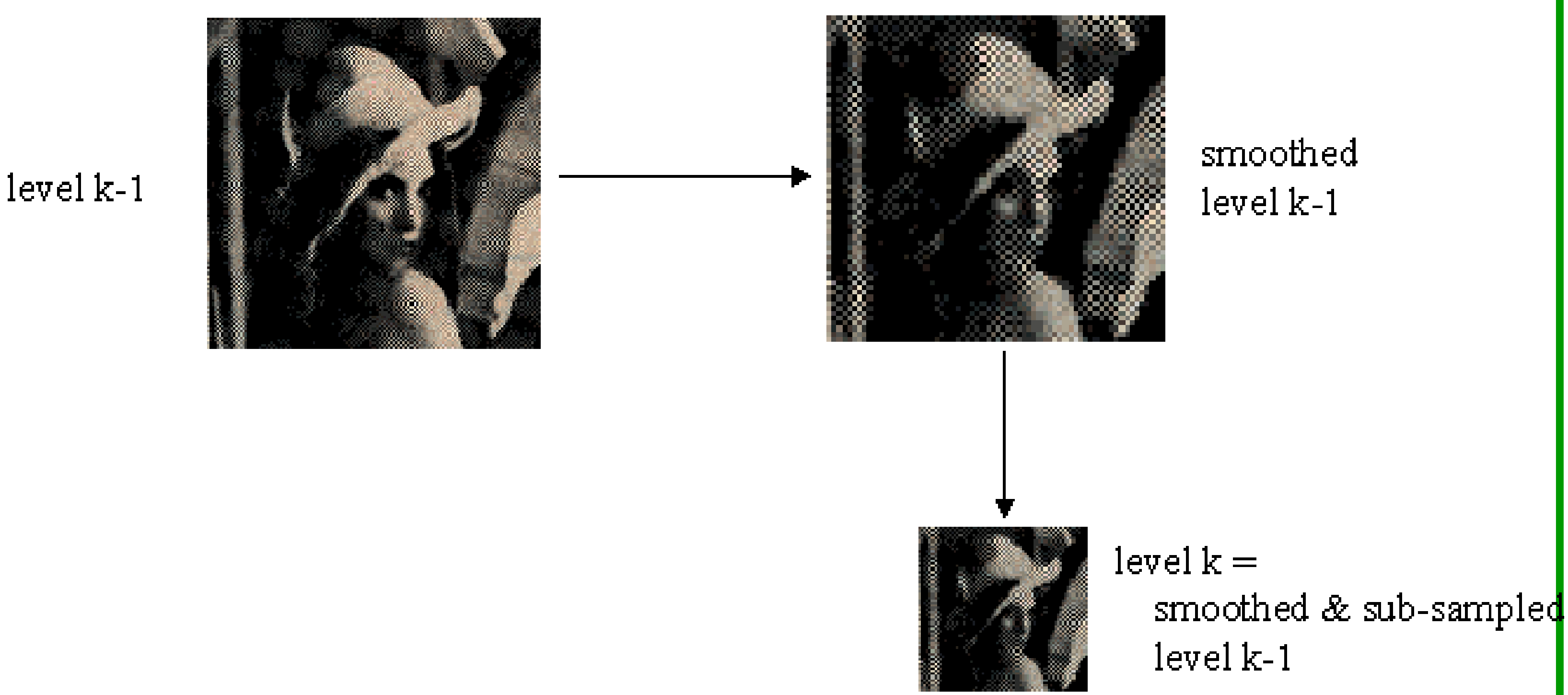
  Map a level k image (low-res) to a level k-1 image (hi-res)



Building A Gaussian Pyramid

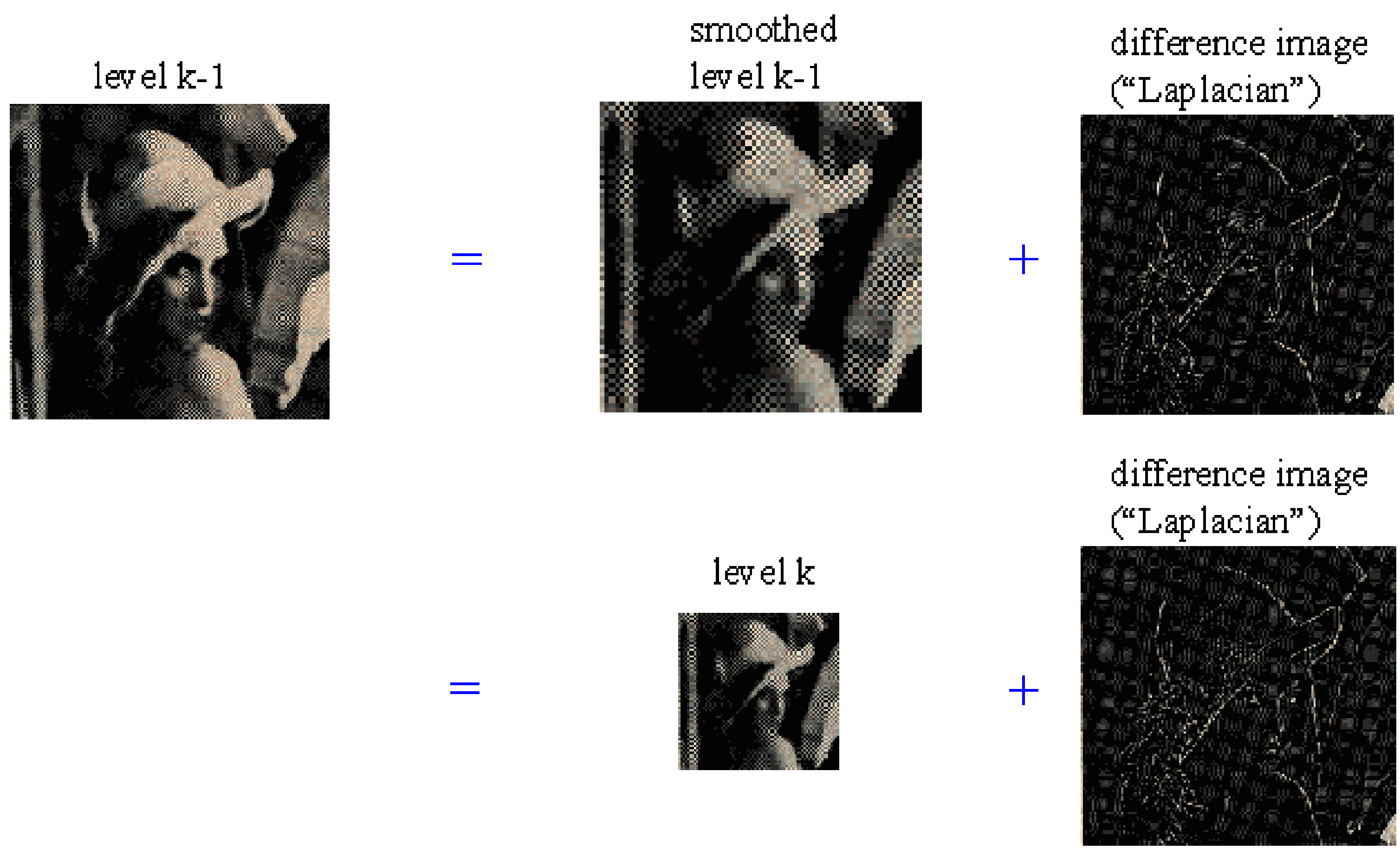
Basic idea of the REDUCE operation:

- If we smooth the level k-1 image appropriately, the resulting image will have less detail than the original
  $\Rightarrow$ we can represent the result with a lower-resolution image!!



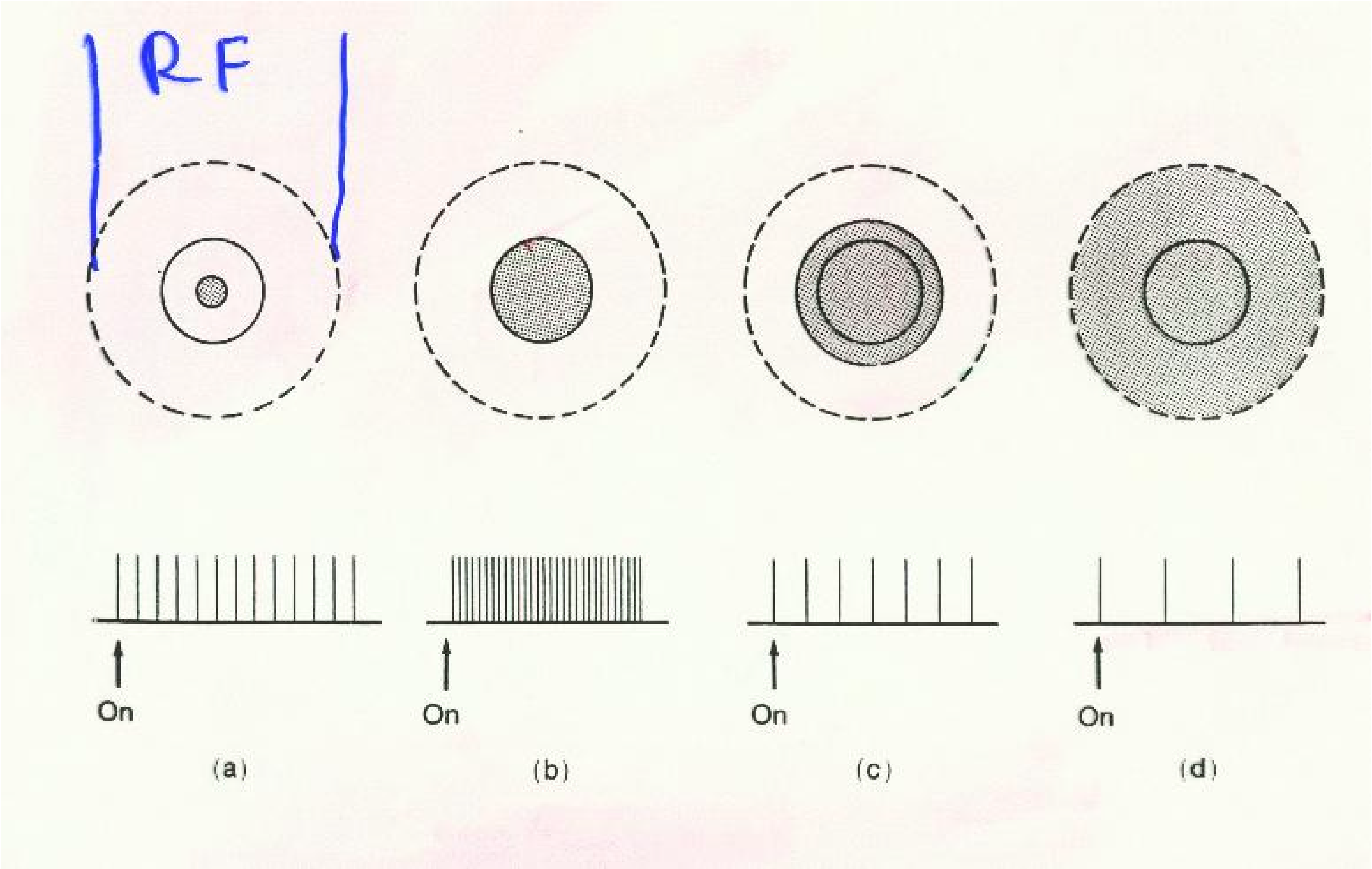
Laplacian Image Pyramid

Idea: Representing level k-1 image by a level k image & a "difference image"



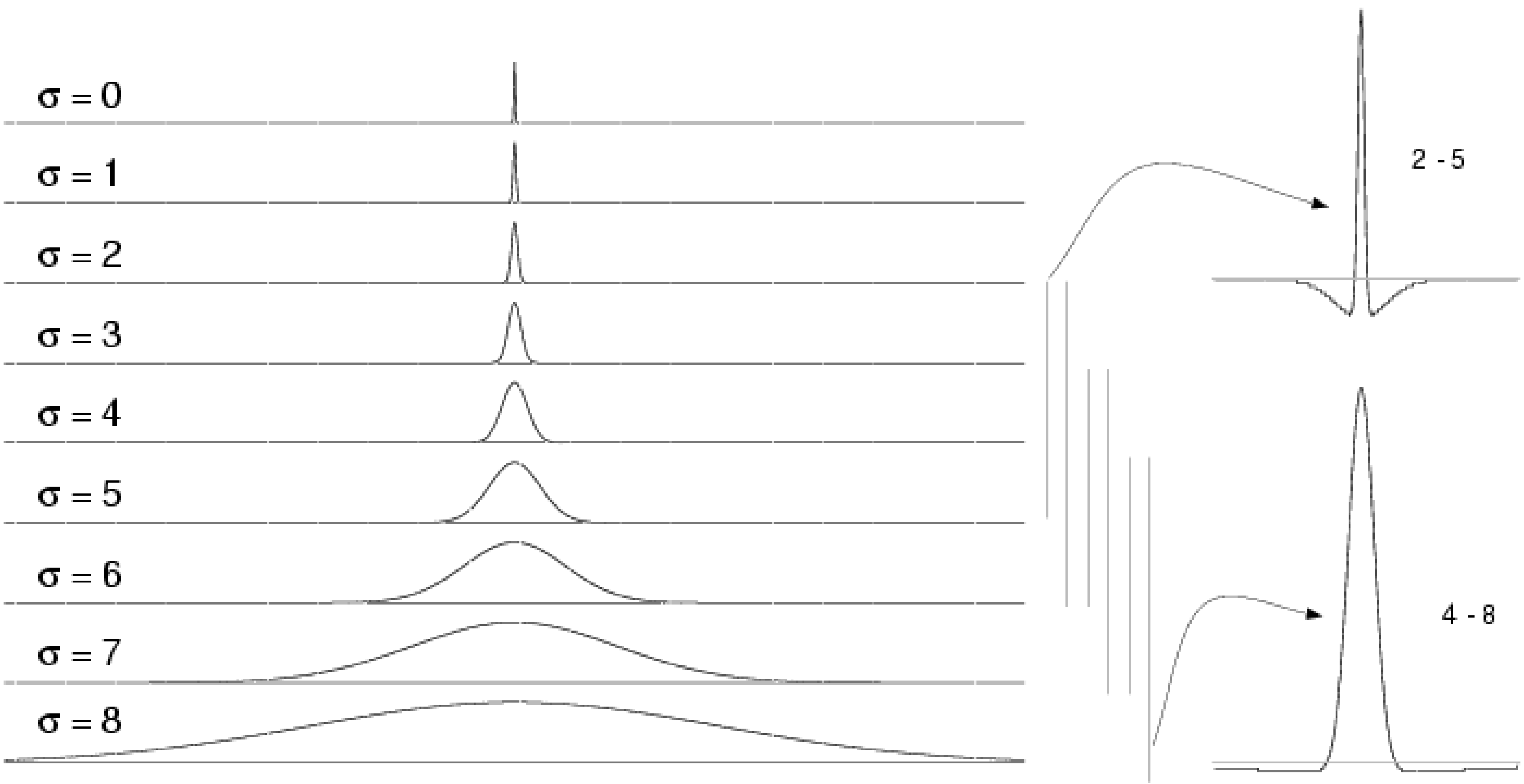
Biological Feature Extraction

Center-surround vs. Laplacian of Gaussian.



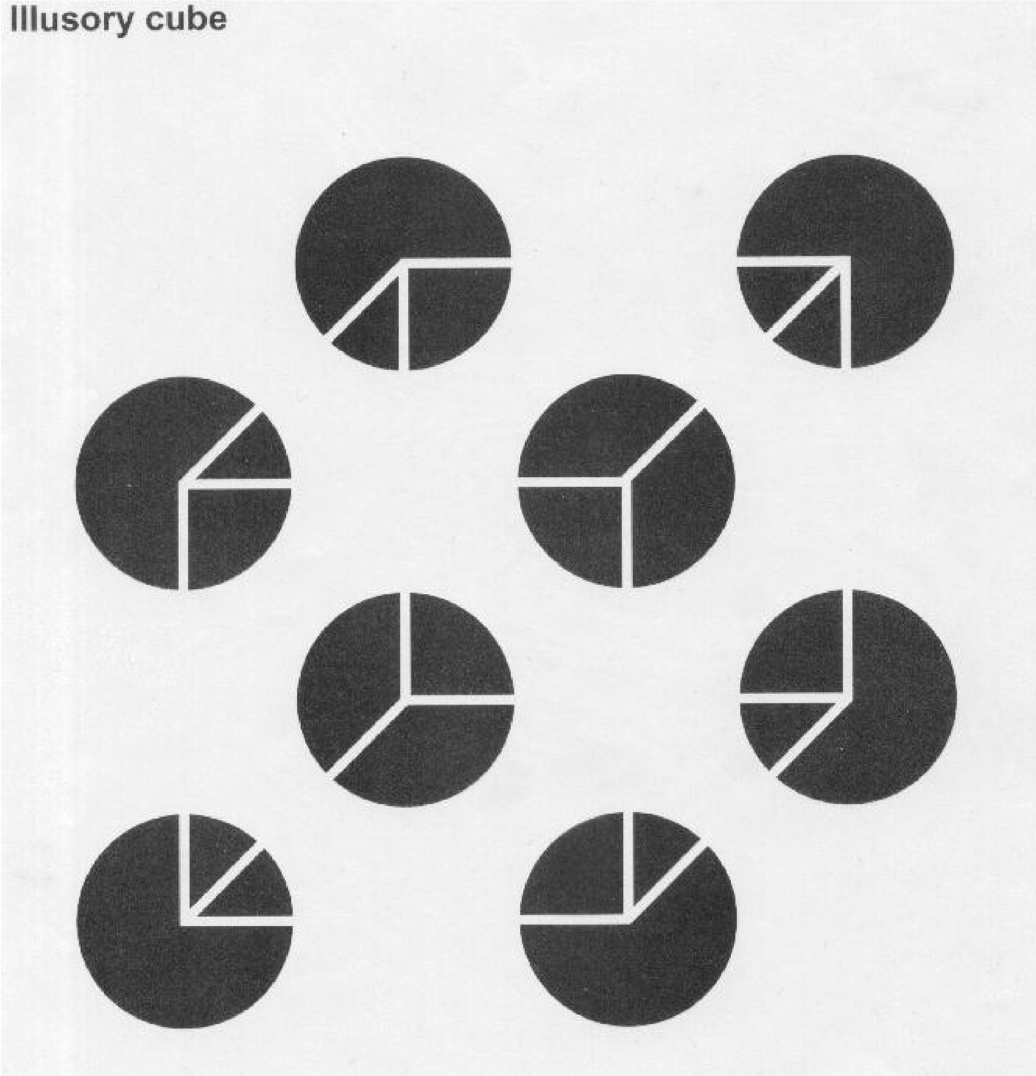
Using Pyramids to Compute Biological Features

- Build Gaussian Pyramid
- Take difference between pixels at same image locations but different scales
- Result: difference-of-Gaussians receptive fields



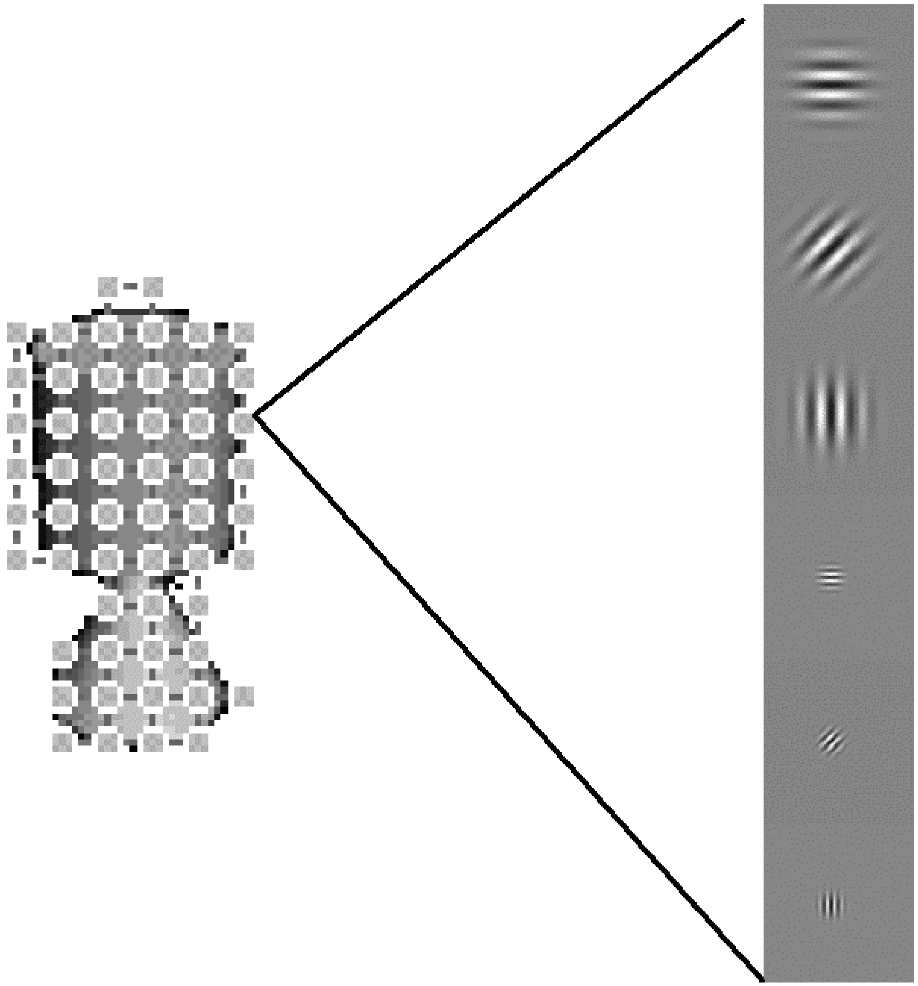
Illusory Contours

Some mechanism is responsible for our illusory perception of contours where there are none…



Gabor "jets"

Similar to a biological hypercolumn: collection of Gabor filters with various orientations and scales, but all centered at one visual location.



Non-Classical Surround

Sillito et al, Nature, 1995: response of neurons is modulated by stimuli outside the neuron's receptive field.

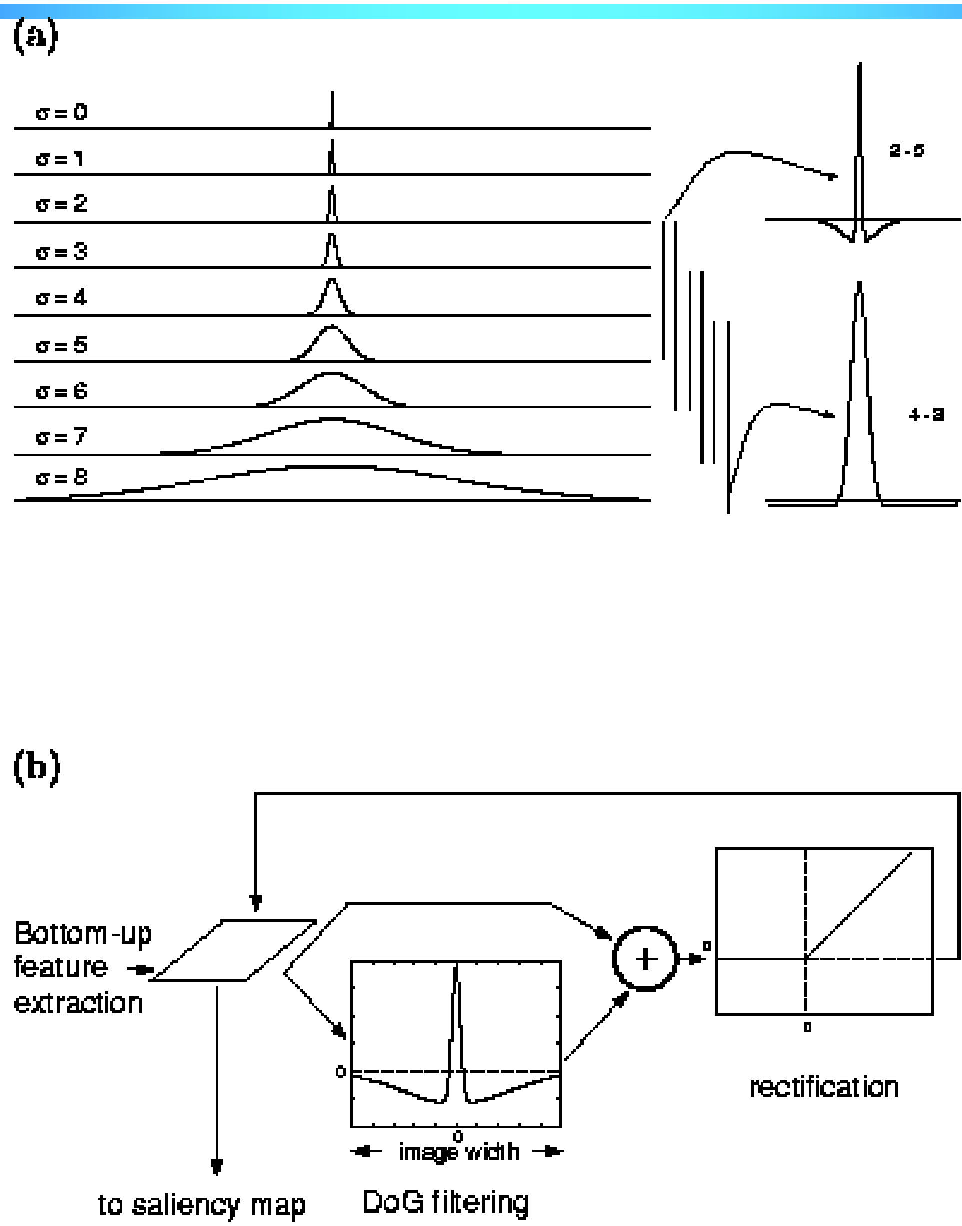
Method:

- Map receptive field location and size
- Check that neuron does not respond to stimuli outside the mapped RF
- Present stimulus in RF
- Compare this baseline response to response obtained when stimuli are also present outside the RF.

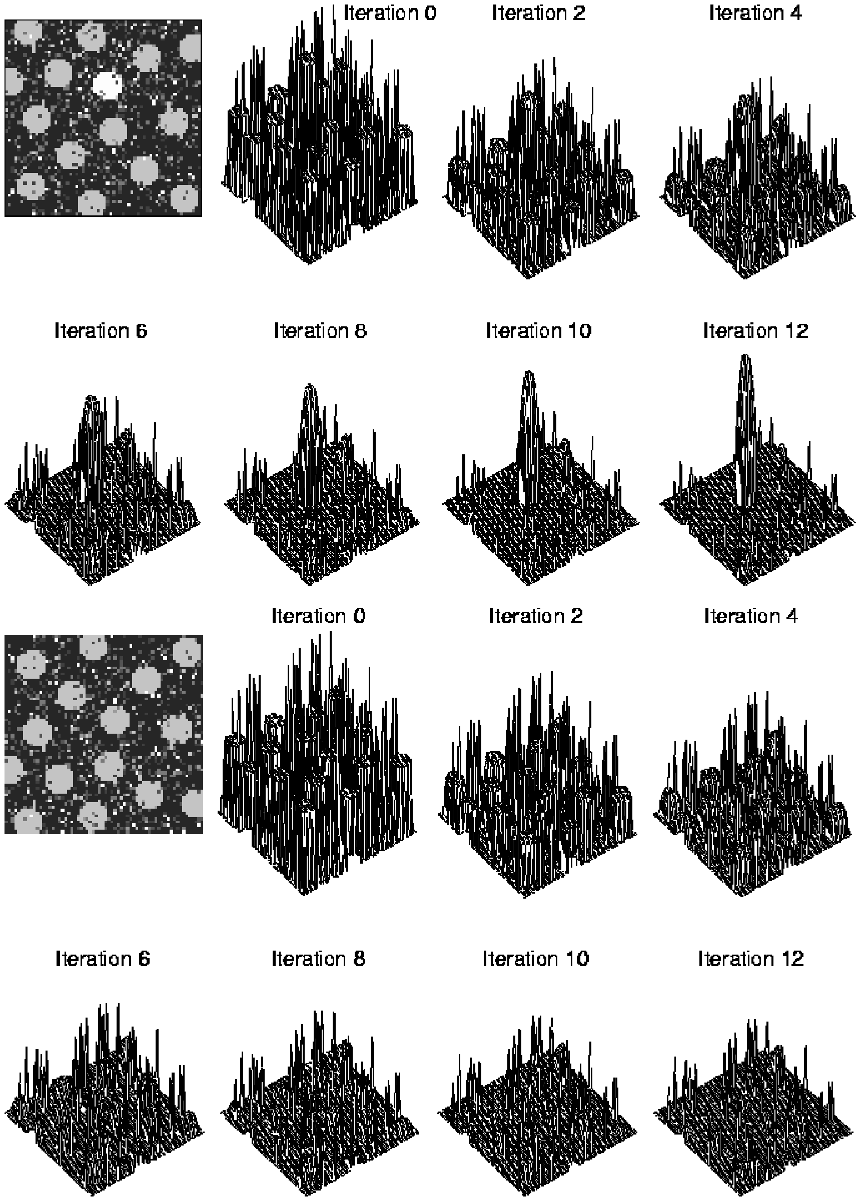
Result:

- stimuli outside RF similar to the one inside RF inhibit neuron
- stimuli outside RF very similar to the one inside RF do not affect (or enhance very slightly) neuron

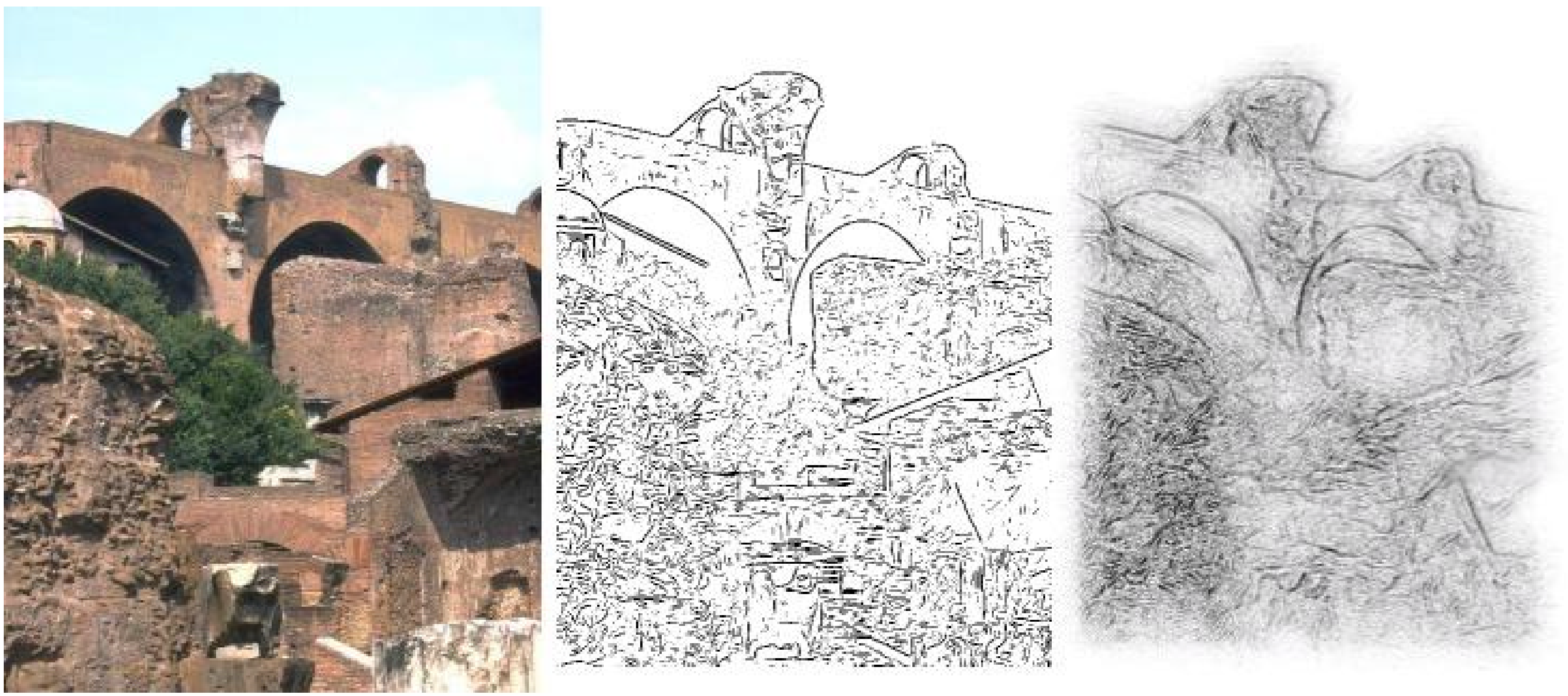
Non-classical surround inhibition



Example


Iteration 0
Iteration 2
Iteration 4
Iteration 6
Iteration 8
Iteration 10
Iteration 12
Iteration 0
Iteration 2
Iteration 4
Iteration 6
Iteration 8
Iteration 10
Iteration 12

Non-Classical Surround & Edge Detection

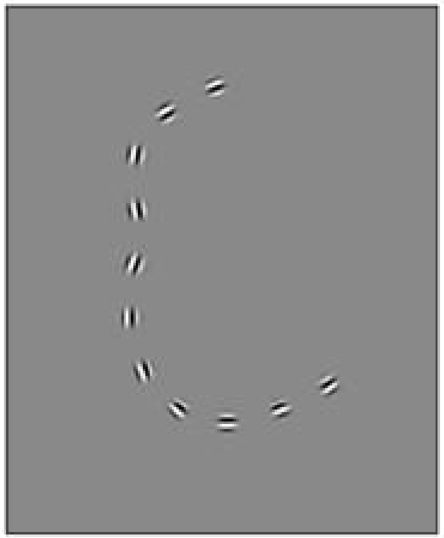


(a) Original image (b) Classical (local input) (c) Non-classical (nonlocal) input only

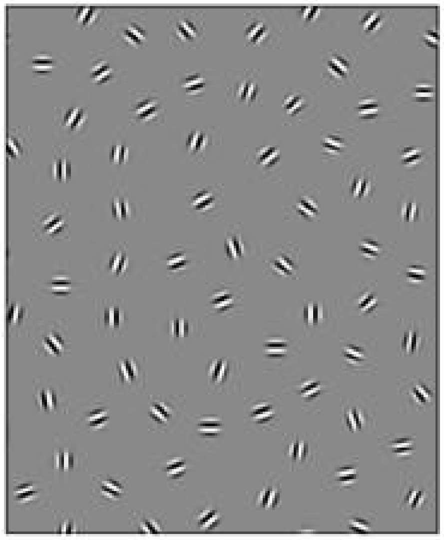
Holt & Mel, 2000

Long-range Excitation

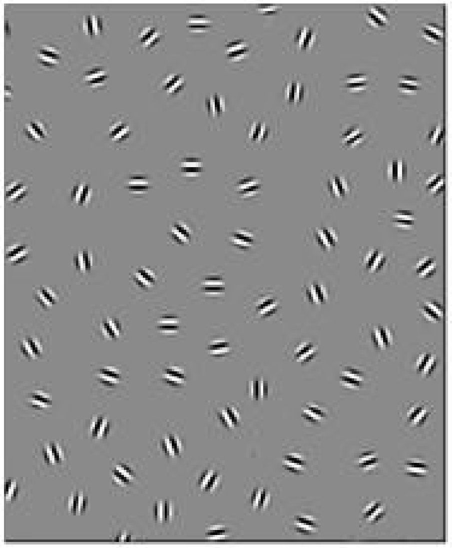
a



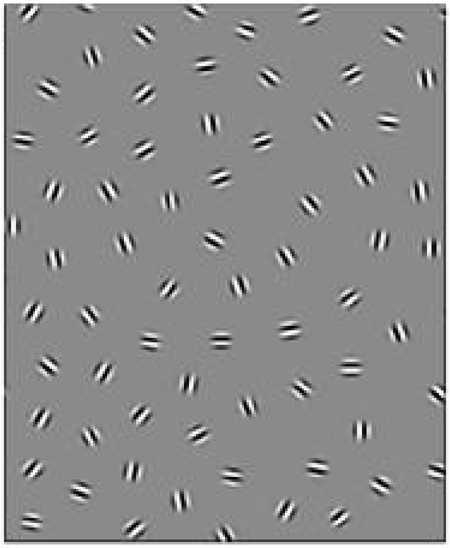
b



a



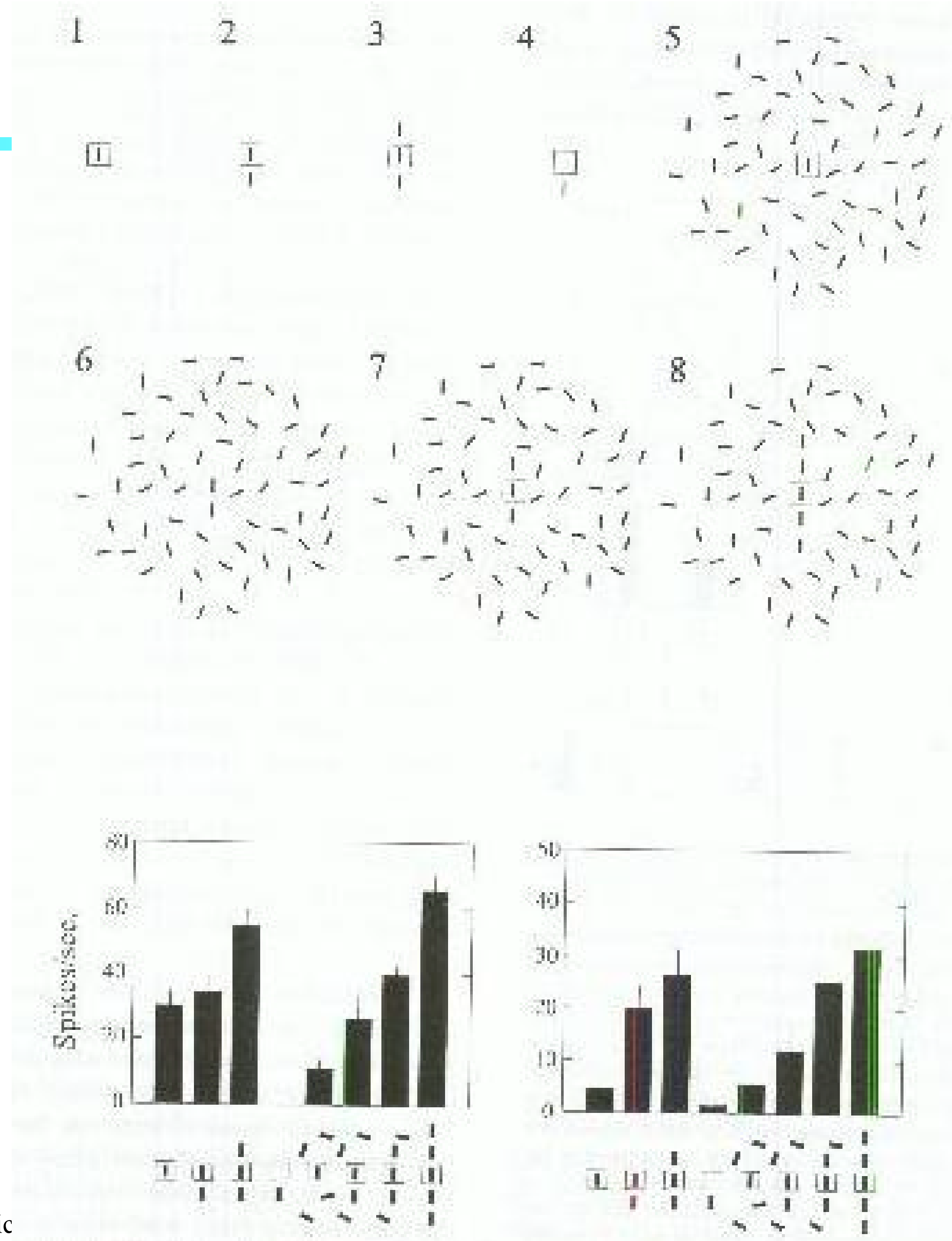
b



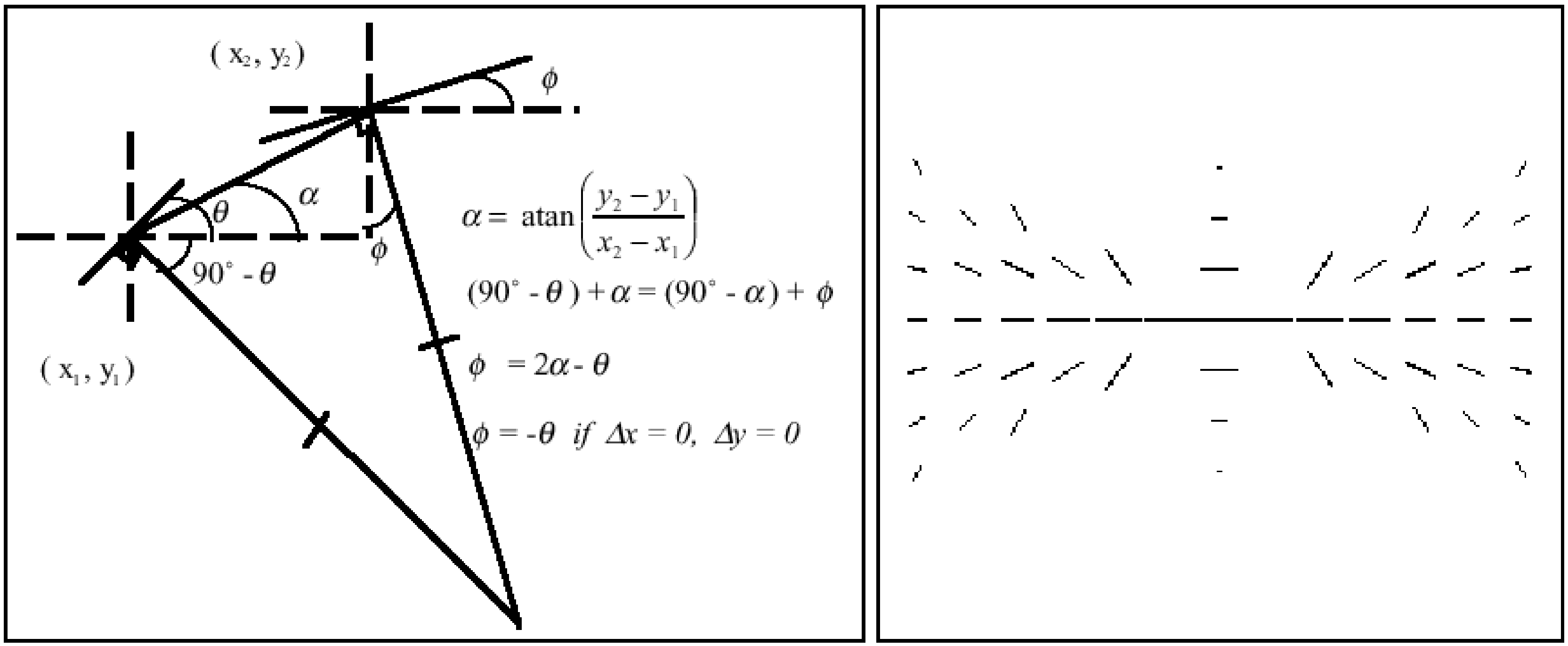
Long-range

Gilbert et al, 2000.

Stimulus outside RF
enhances neuron's
response if placed and
oriented such
as to form a contour.

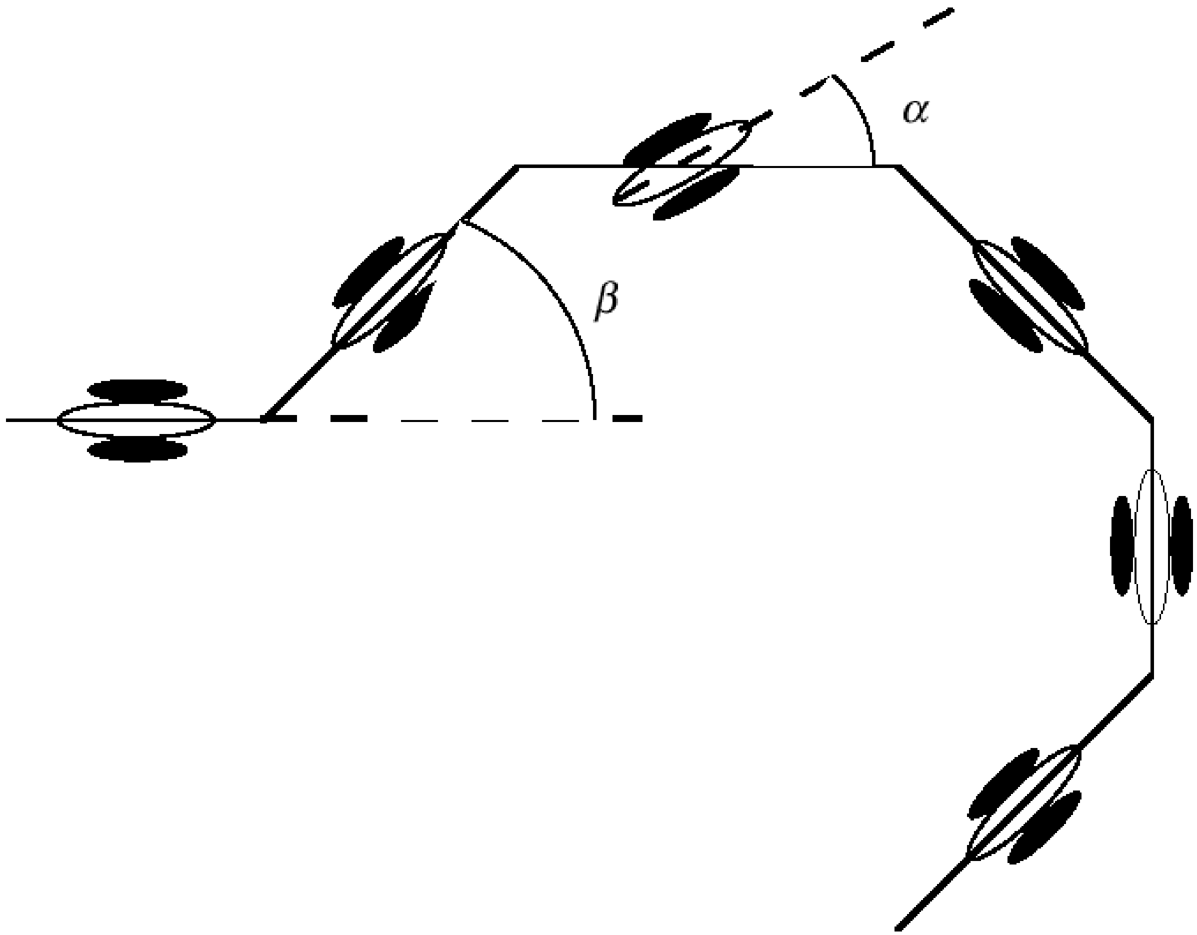


Modeling long-range connections



a b

Contour completion



Grouping and Object Segmentation

We can do much more than simply extract and follow contours…