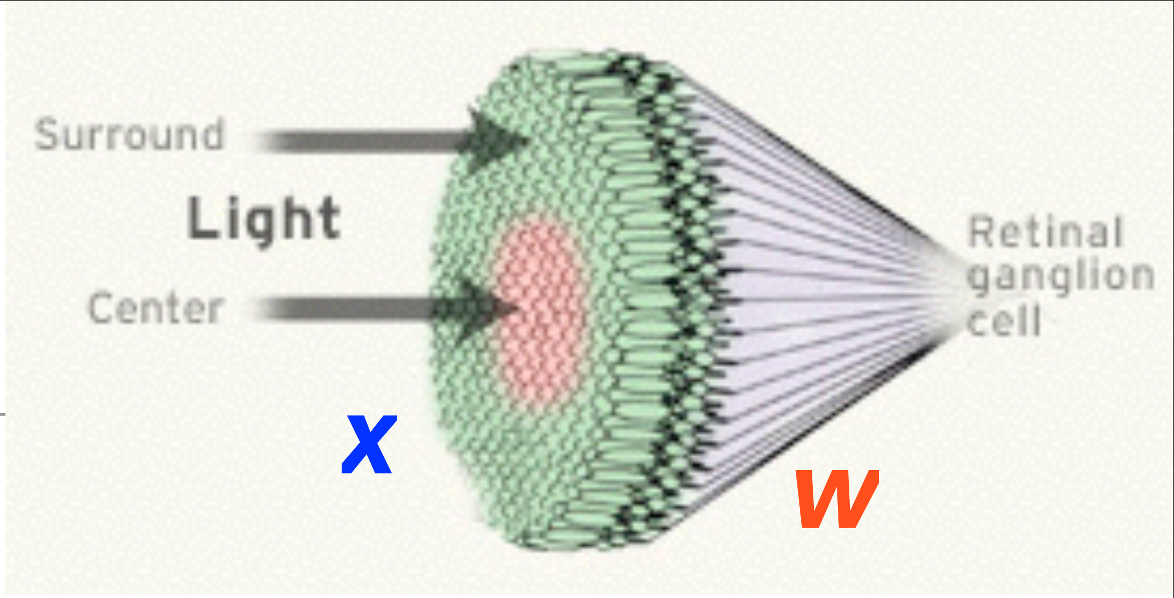
Computational Vision

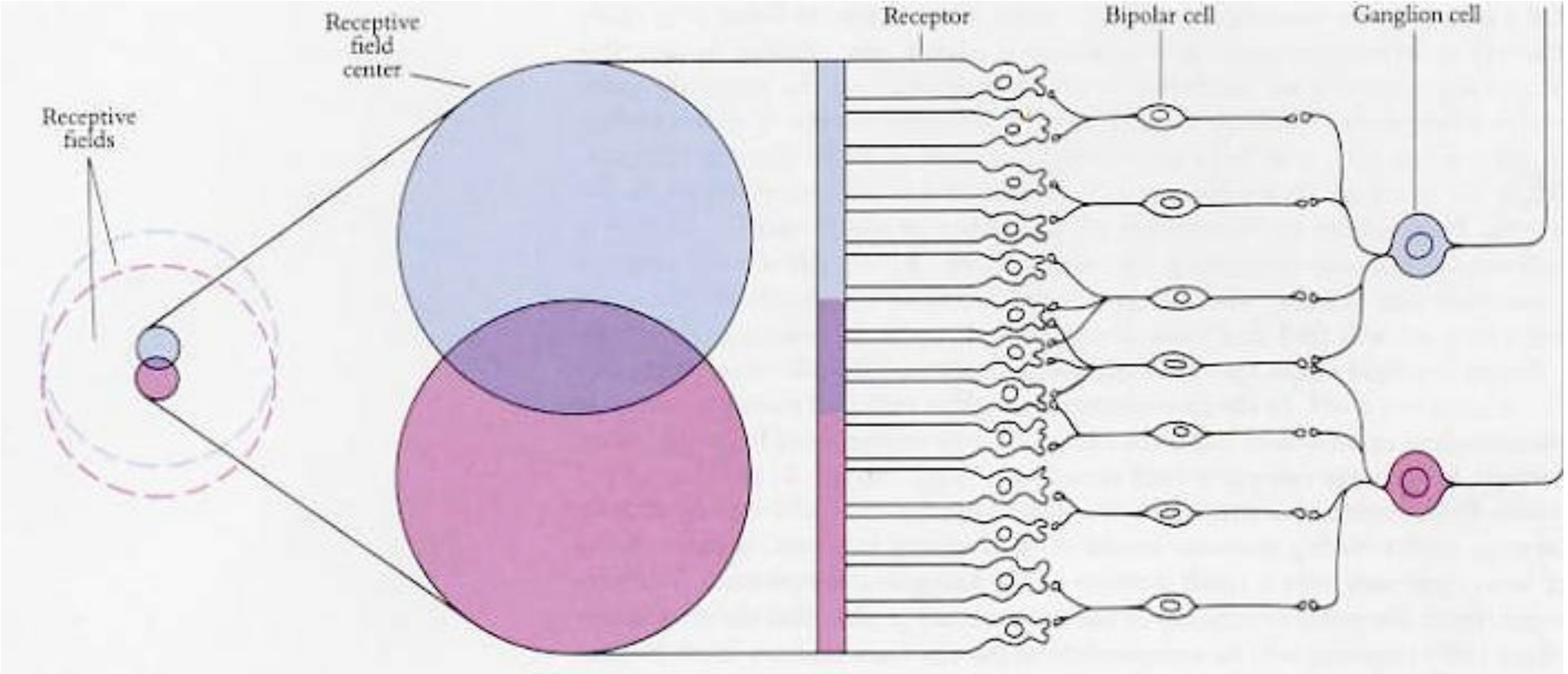
LGN

- Feature detection
- Filtering

What do neurons compute?



- Neurons detect features (=patterns or templates) that are stored in their synaptic weights



Source: Hubel

Neurons as feature detectors

- ~1M receptors
- 2.5-3.5M connecting neurons
- 0.5 M ganglion cells
- Each ganglion cell receives many inputs from the receptors
- Each receptor projects to many ganglion cells

Neurons as feature detectors

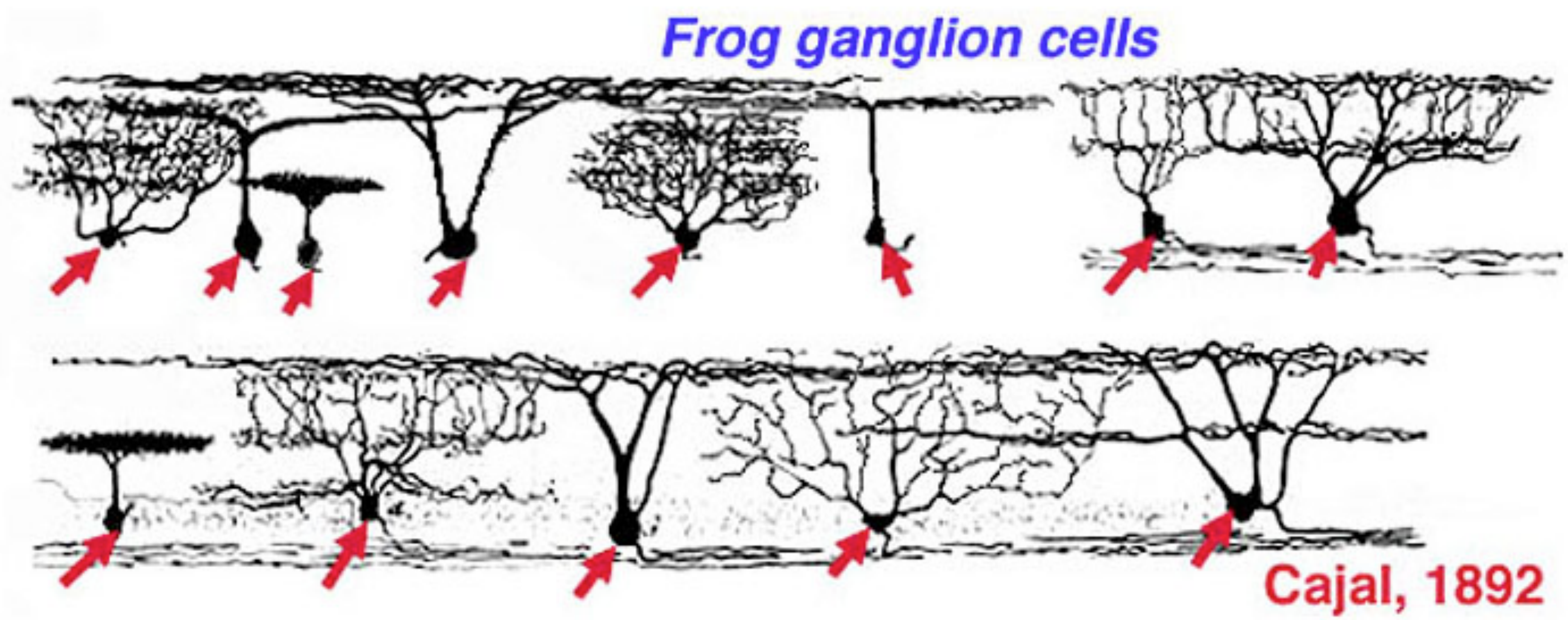
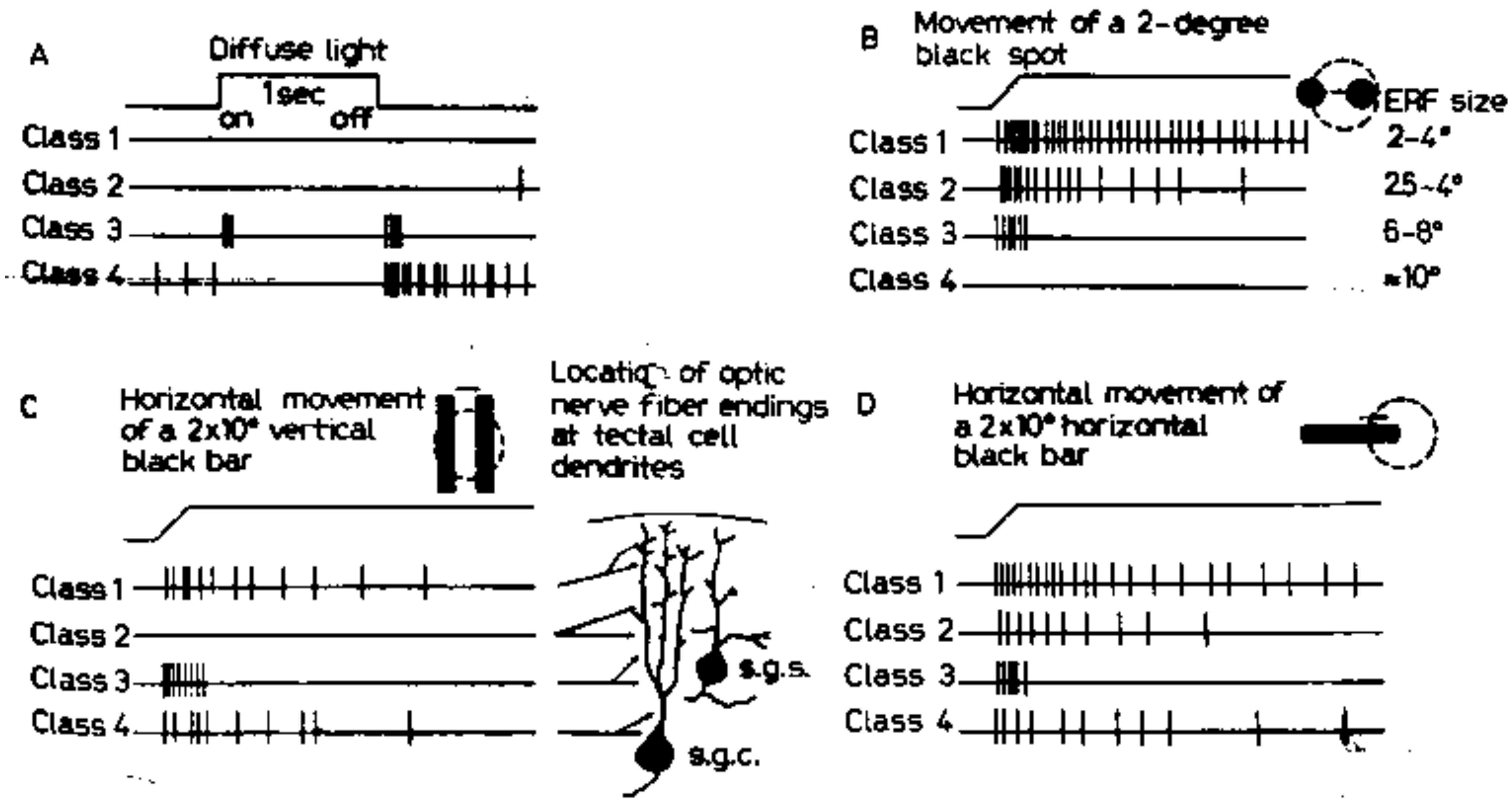


Fig. 1.Cajals drawing of ganglion cells of the frogs retina.

source: webvision

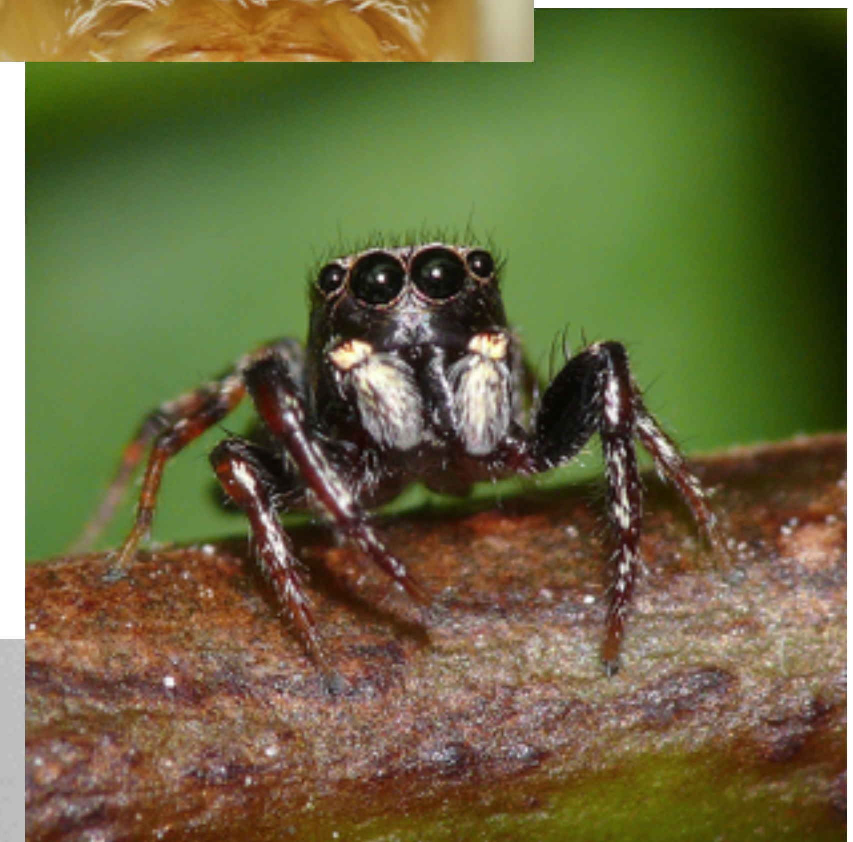
Neurons as feature detectors



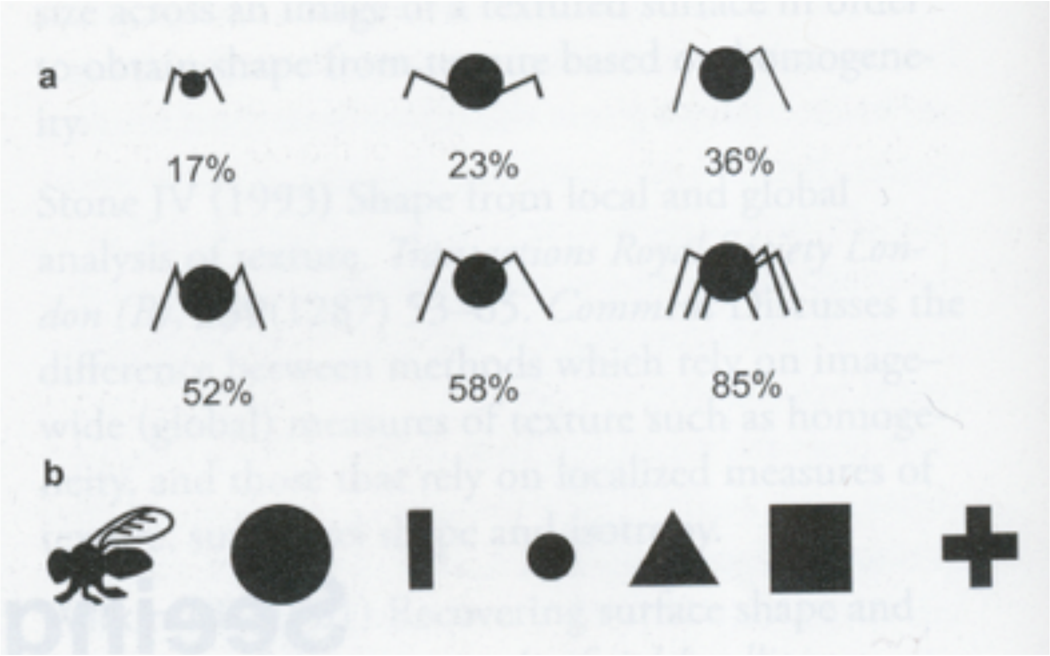
We have been tempted for example, to call the convexity detectors [class 2] “bug perceivers”. Such a fiber responds best when a dark object, smaller than a receptive field, enters that field, stops, and moves about intermittently thereafter. The response is not affected if the lighting changes or if the background (say a picture of grass and flowers) is moving, and is not there if only the background, moving or still, is in the field. Could one better describe a system for detecting an accessible bug? [Lettvin et al 1959]

Template matching by the jumping spider

- 4 pairs of eyes
- Eyes have single lenses like mammals (unlike insects with compound eyes)
- Scan visual scenes by moving body and retina (lens is fixed)
- Detection at 30-40cm
- Hunt preys



Template matching by the jumping spider



mating behavior

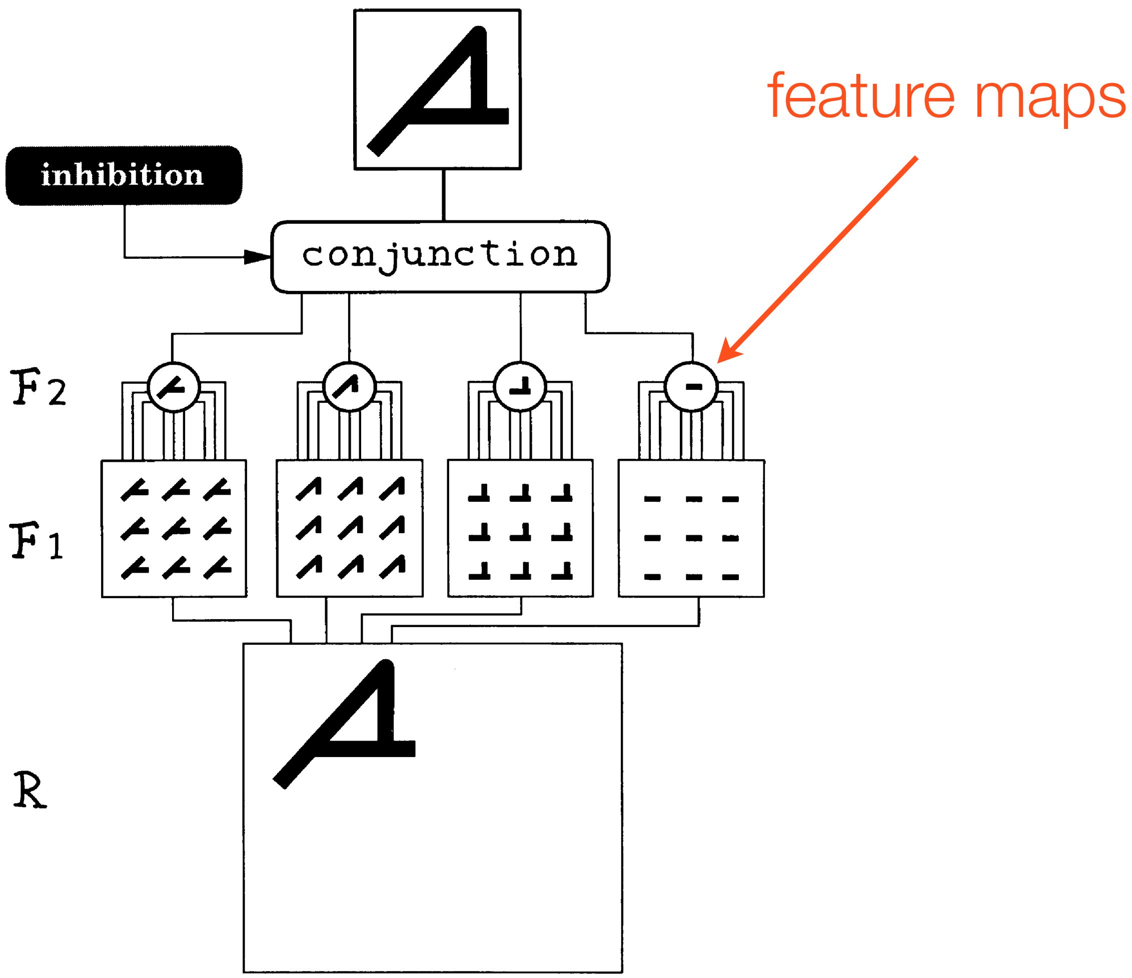
capture behavior

Drees '52

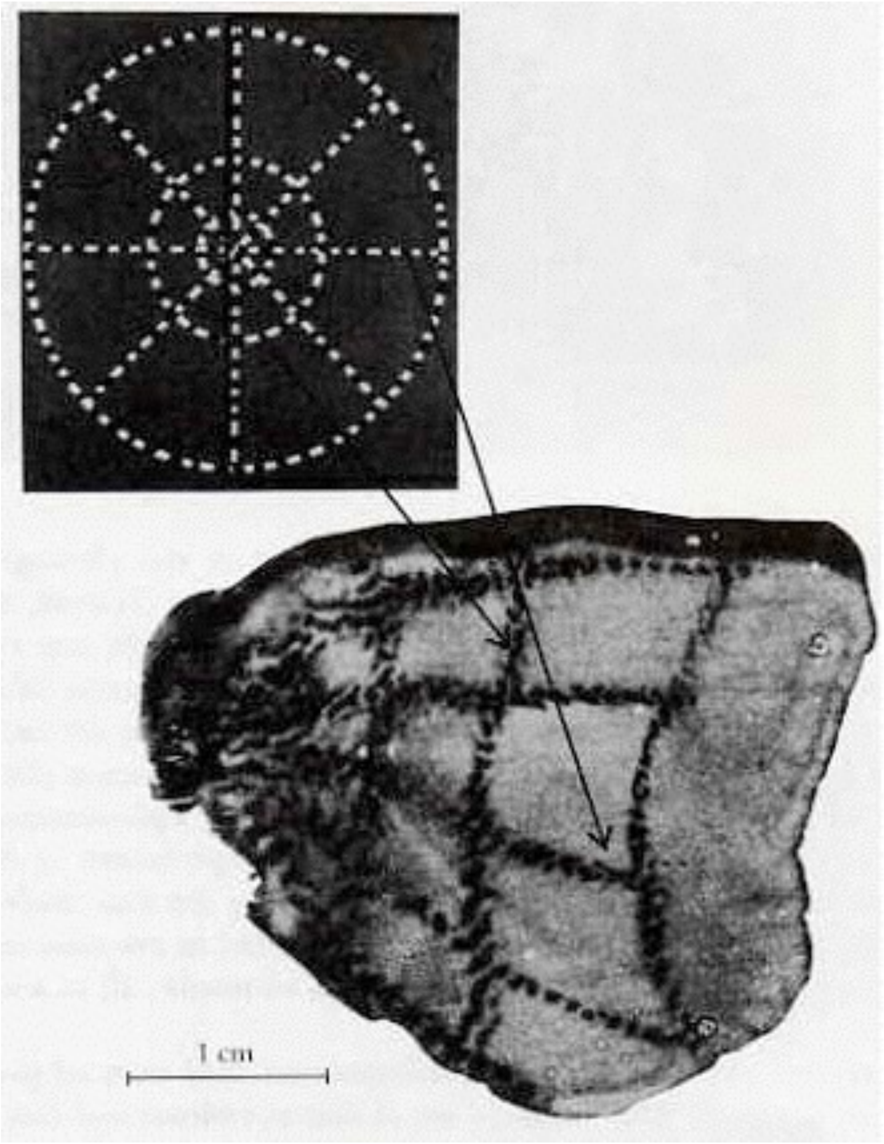
Land & Nilsson '01

Cortex vs. computers

Brains: Full-replication scheme



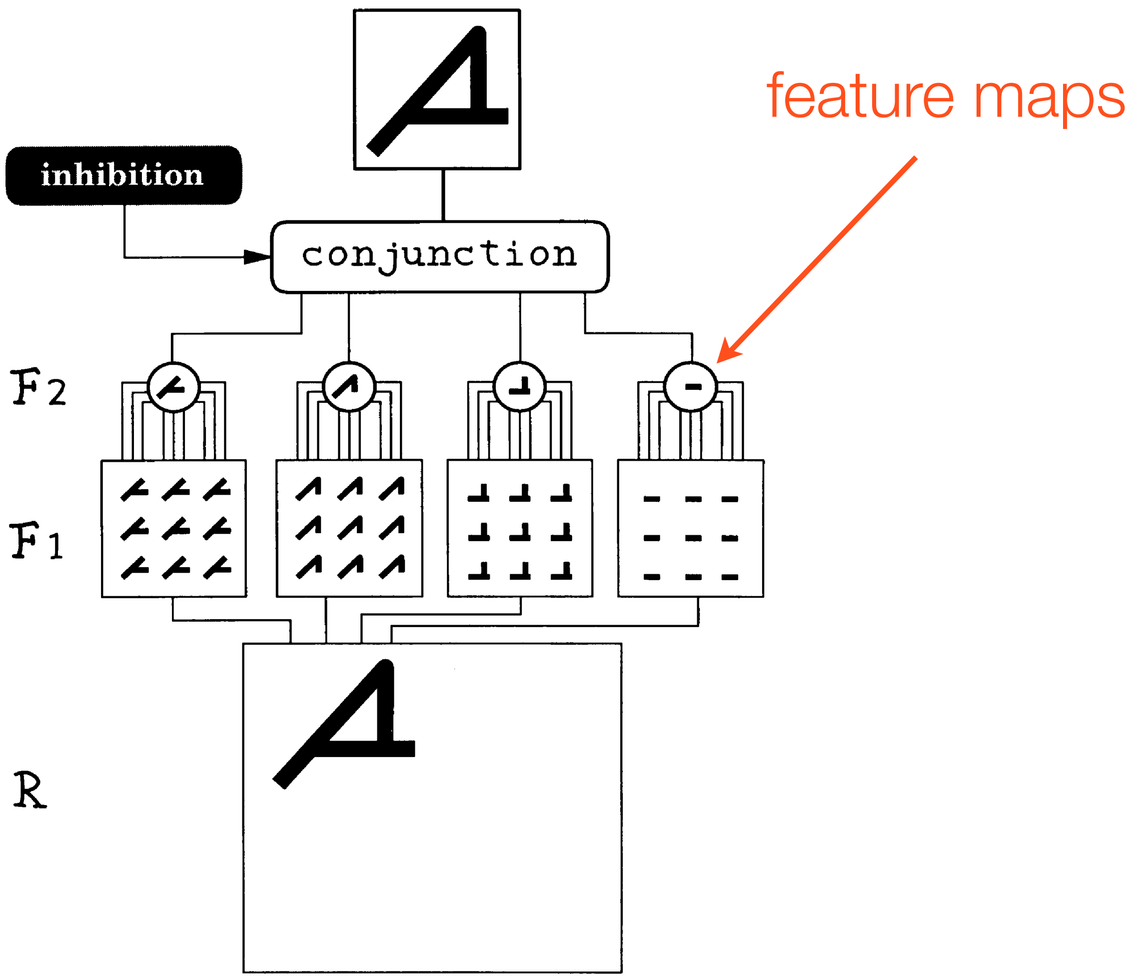
Retinotopy in early visual areas



source: Hubel

Cortex vs. computers

Brains: Full-replication scheme



Computers: Filtering/Convolution

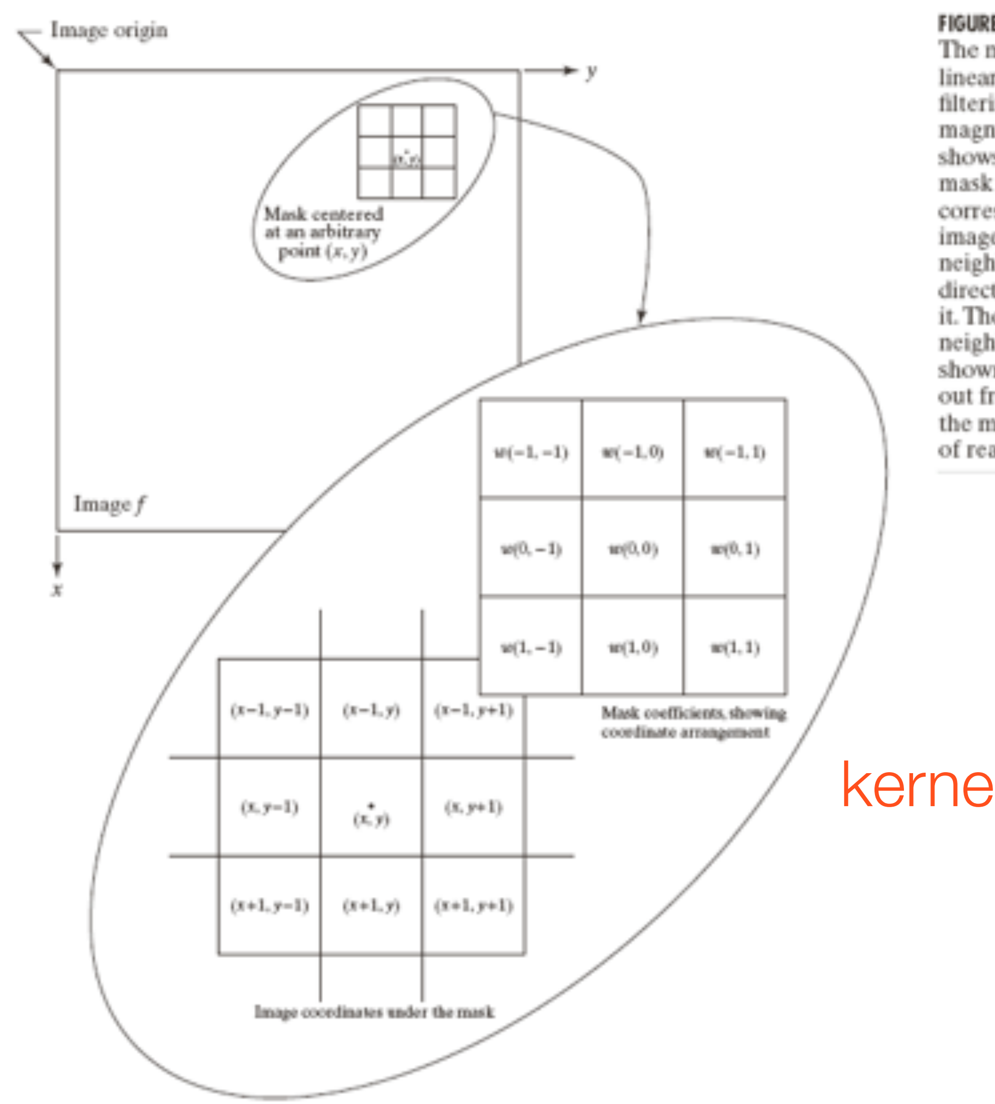


FIGURE 3.13 The mechanics of linear spatial filtering. The magnified drawing shows a 3× 3 filter mask and the corresponding image neighborhood directly under it. The image neighborhood is shown displaced out from under the mask for ease of readability.

Principles of spatial convolution/filtering

- Multiply each pixel in a neighborhood by a corresponding coefficient and sum the results to get response at each point (x,y)
- Neighborhood of size (m,n) requires nm coefficients
- Coefficients arranged as matrix called filter, mask, filter mask, kernel, or template
- Move center of the filter mask, w, from point to point in image f

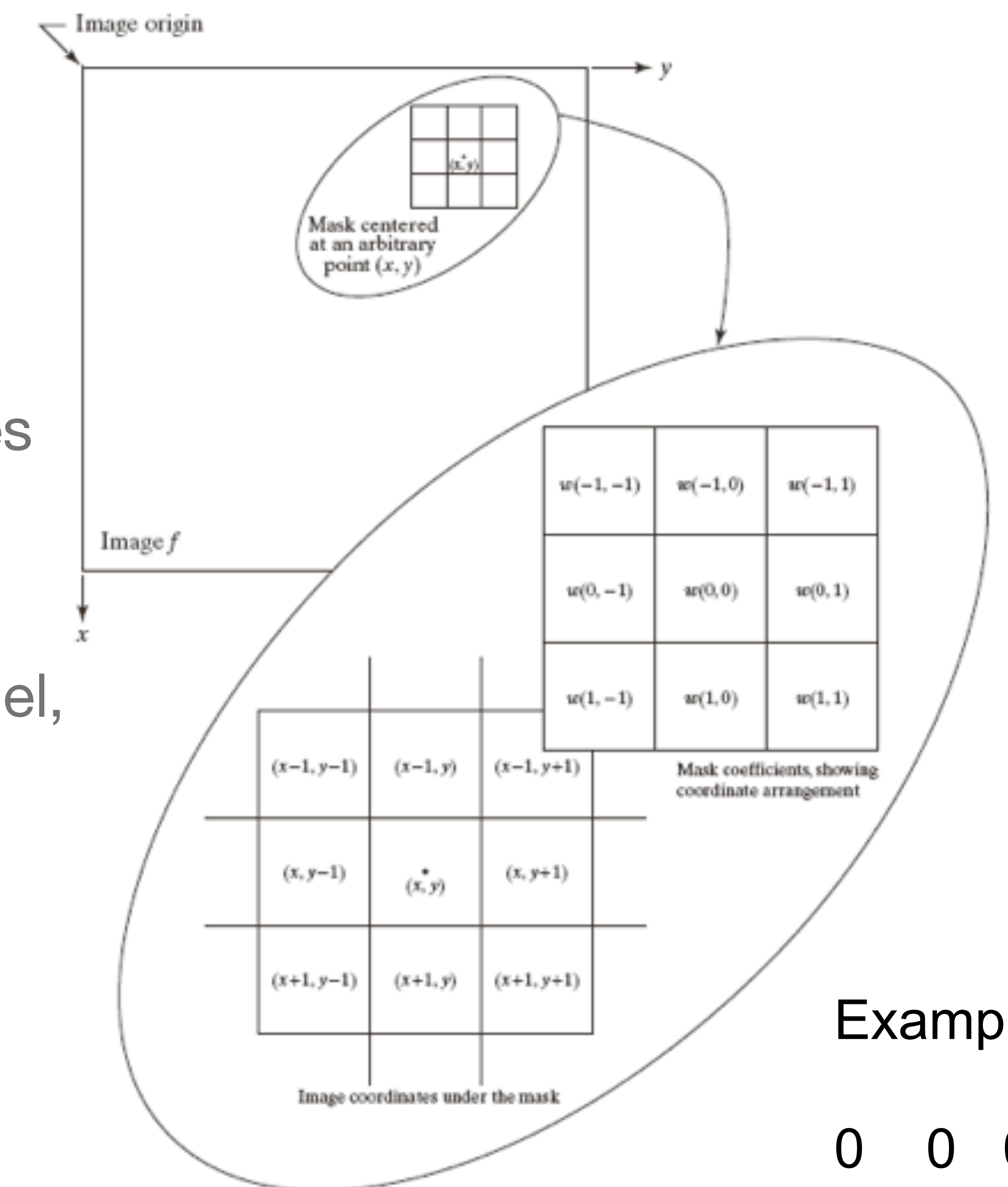


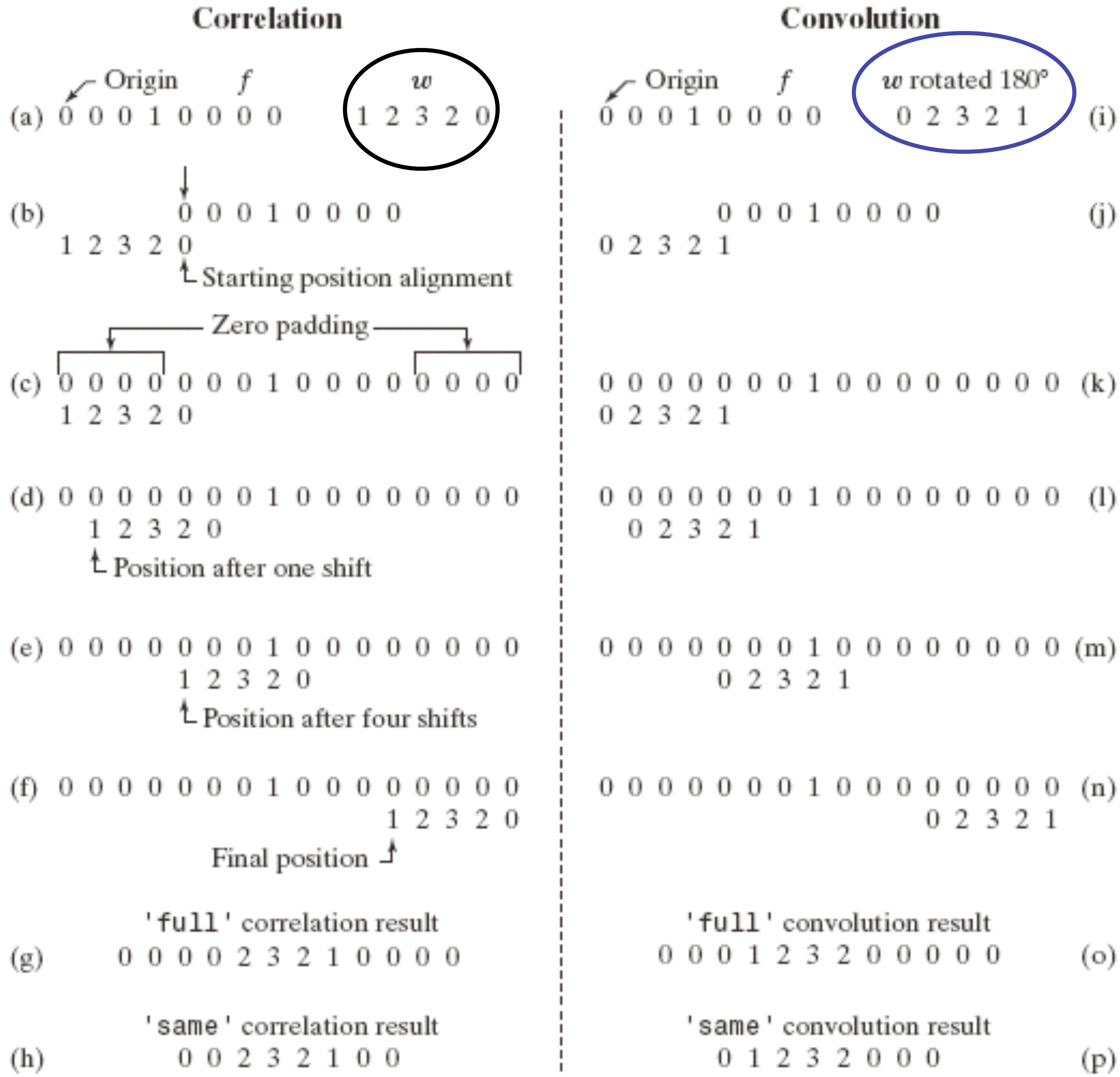
FIGURE 3.13 The mechanics of linear spatial filtering. The magnified drawing shows a 3×3 filter mask and the corresponding image neighborhood directly under it. The image neighborhood is shown displaced out from under the mask for ease of readability.

Example mask

0	0	0
1.	0	-1
0	0	0

Convolution is correlation with a rotated filter

FIGURE 3.14 Illustration of one-dimensional correlation and convolution.



imfilter

conv2

2D convolution

Origin of $f(x, y)$

$$
\begin{matrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{matrix}
\qquad
\begin{matrix}
 & w(x, y) & \\
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{matrix}
$$

(a)

$$r_{i,j} = \sum_{x,y} w_{i-x,j-y} f_{x,y}$$

Rotated w

$$
\begin{matrix}
9 & 8 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 5 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
$$

(f)

`'full'` convolution result

$$
\begin{matrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 2 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 5 & 6 & 0 & 0 & 0 \\
0 & 0 & 0 & 7 & 8 & 9 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}
$$

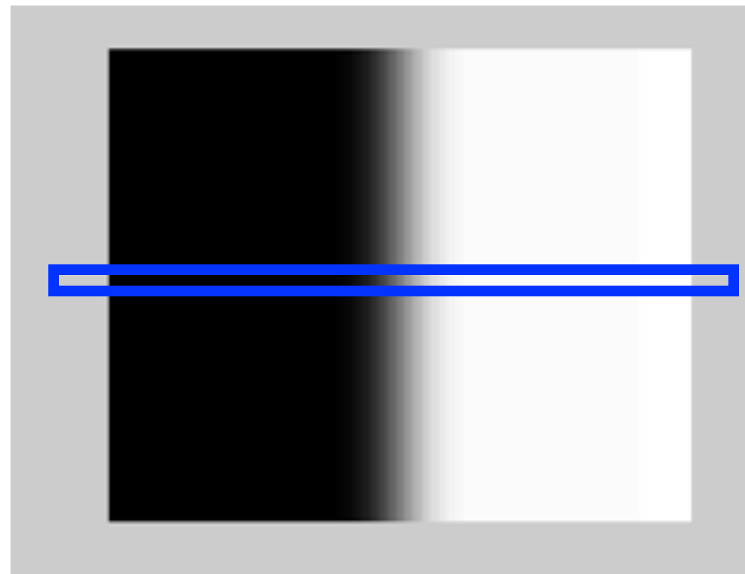
(g)

Filtering in image processing

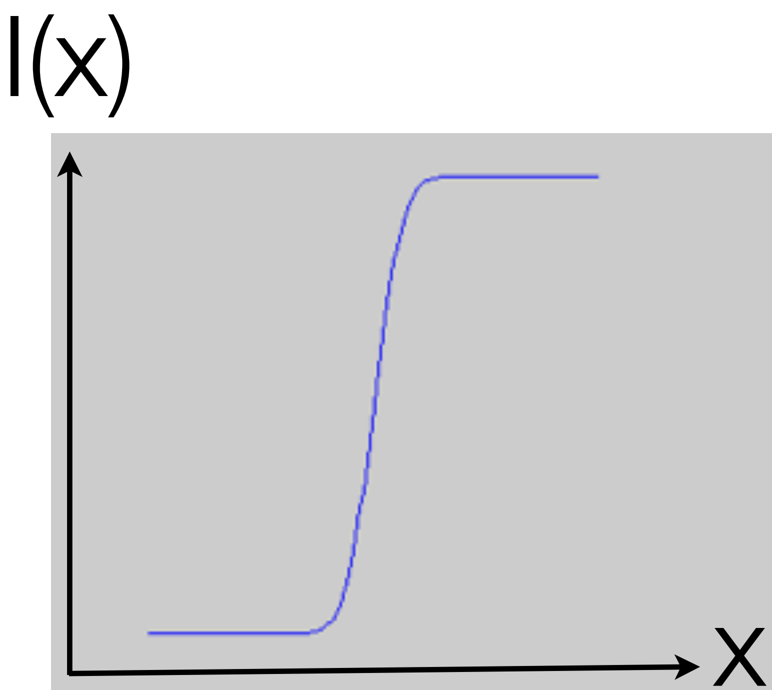
- Filtering the image is a set of dot-products
- **Insight:** Filters look like the effects they are intended to find

- **Exercise:**
  - How well does the template matching operation works for detecting faces and objects?
  - What happens when the appearance of the target object changes (small changes in size, view-point, background clutter, etc)?
  - Play with the size of the templates: What are the pros and cons of small vs. large templates?

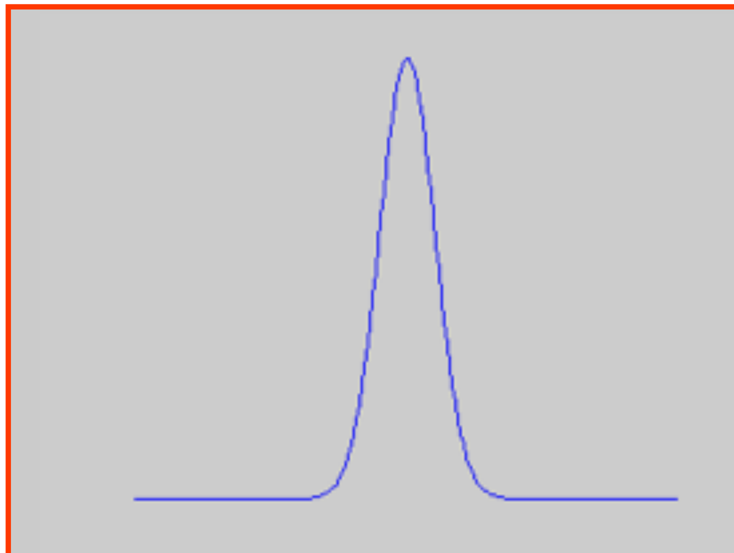
Neurons as edge detectors



edge

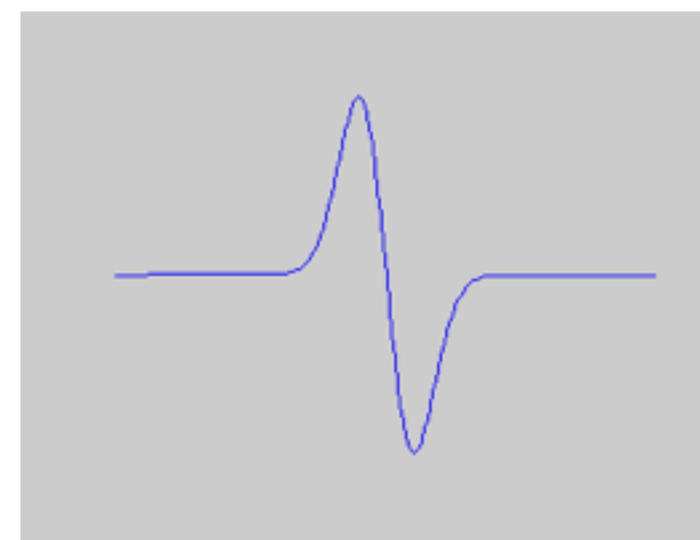


I



First derivative of I

$$\frac{\delta I}{\delta x}$$



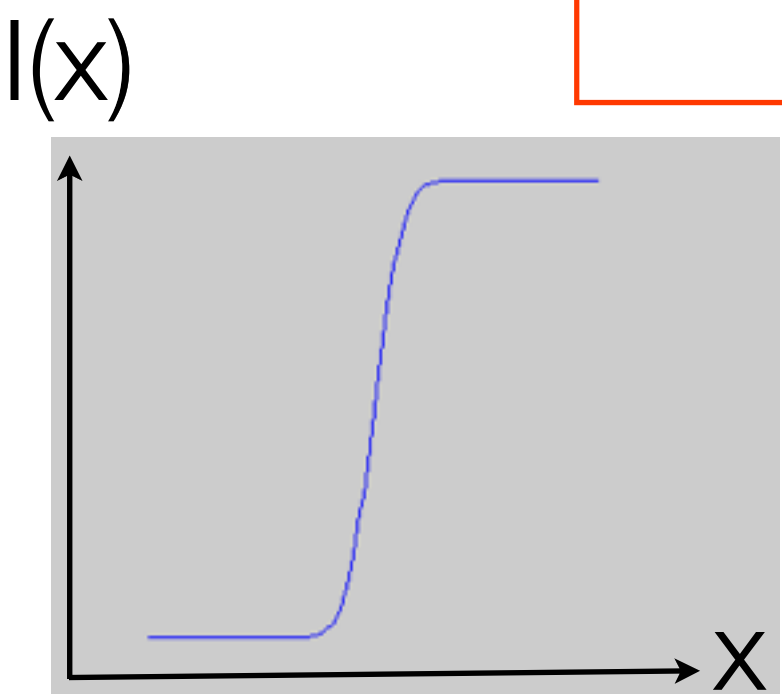
Second derivative of I

$$\frac{\delta^2 I}{\delta x^2}$$

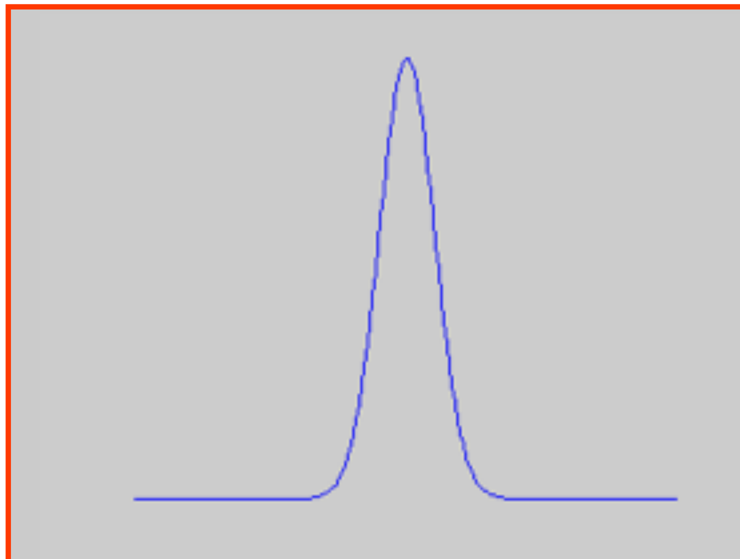
Neurons as edge detectors

$$\frac{\Delta I(x,y)}{\Delta x} \approx \frac{I(x+\Delta x) - I(x)}{\Delta x}$$

$$\approx I(x+1) - I(x)$$

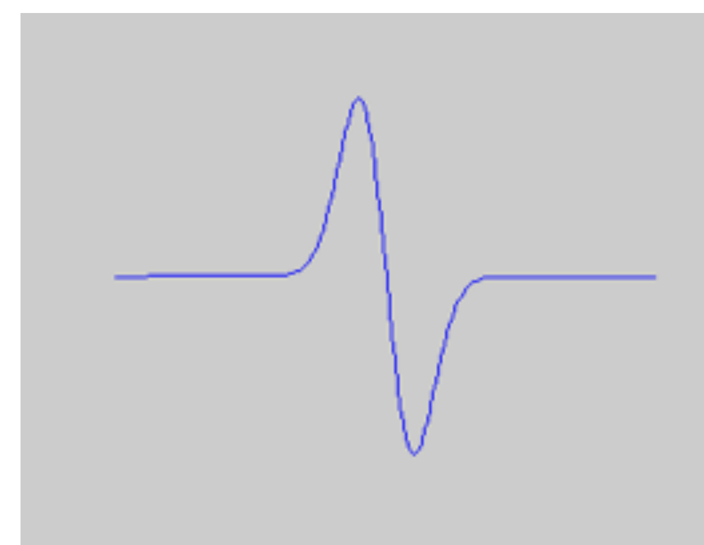


I



First derivative of I

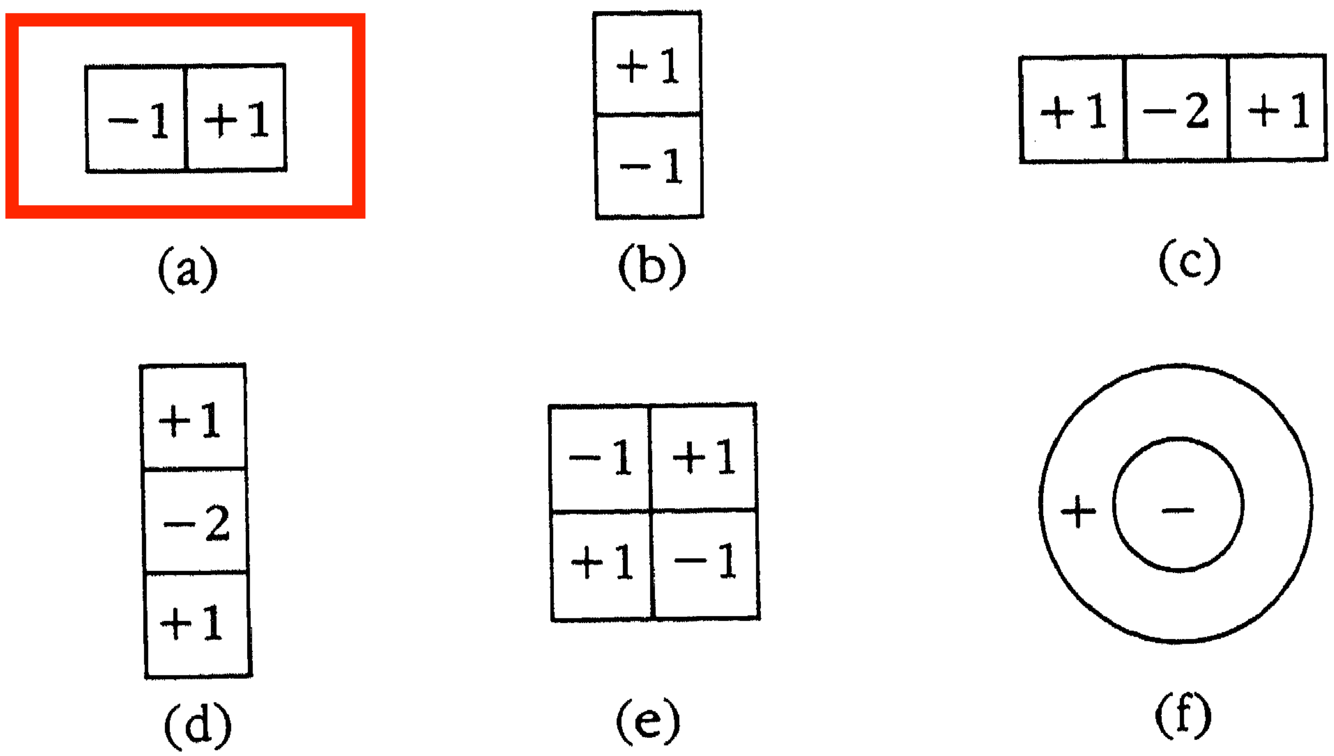
$$\frac{\delta \; I}{\delta \; x}$$

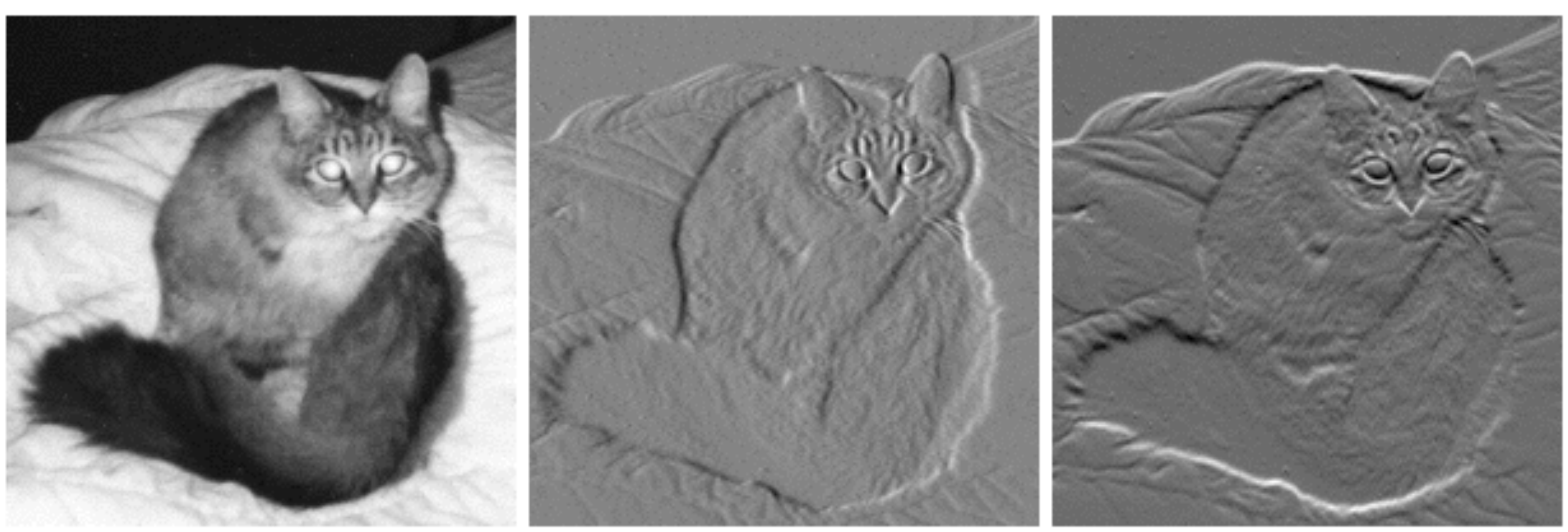


Second derivative of I

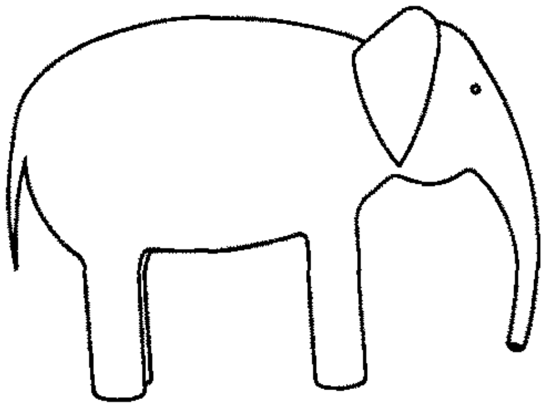
$$\frac{\delta^2 I}{\delta \; x^2}$$

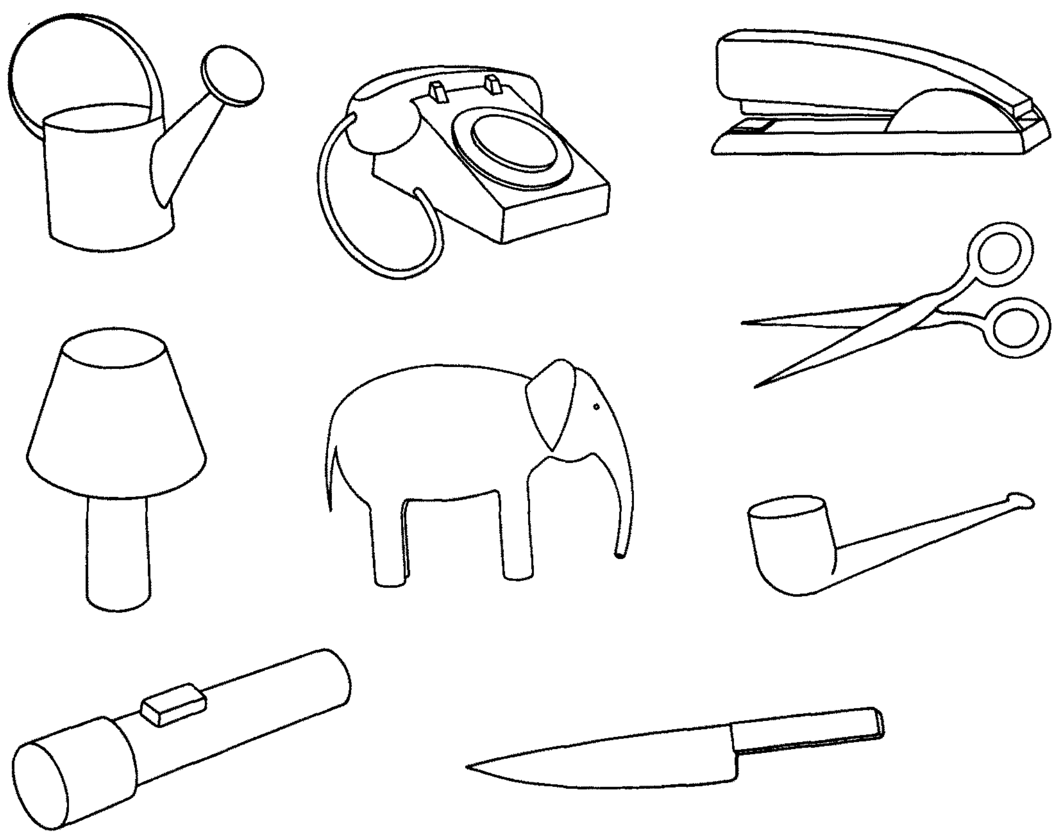
Differential operators





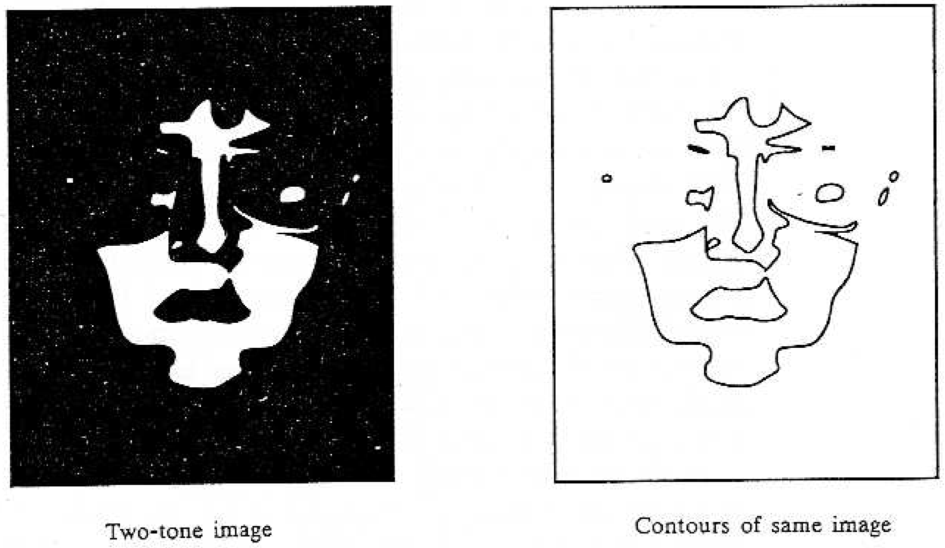
Edges and contours play a special role in vision





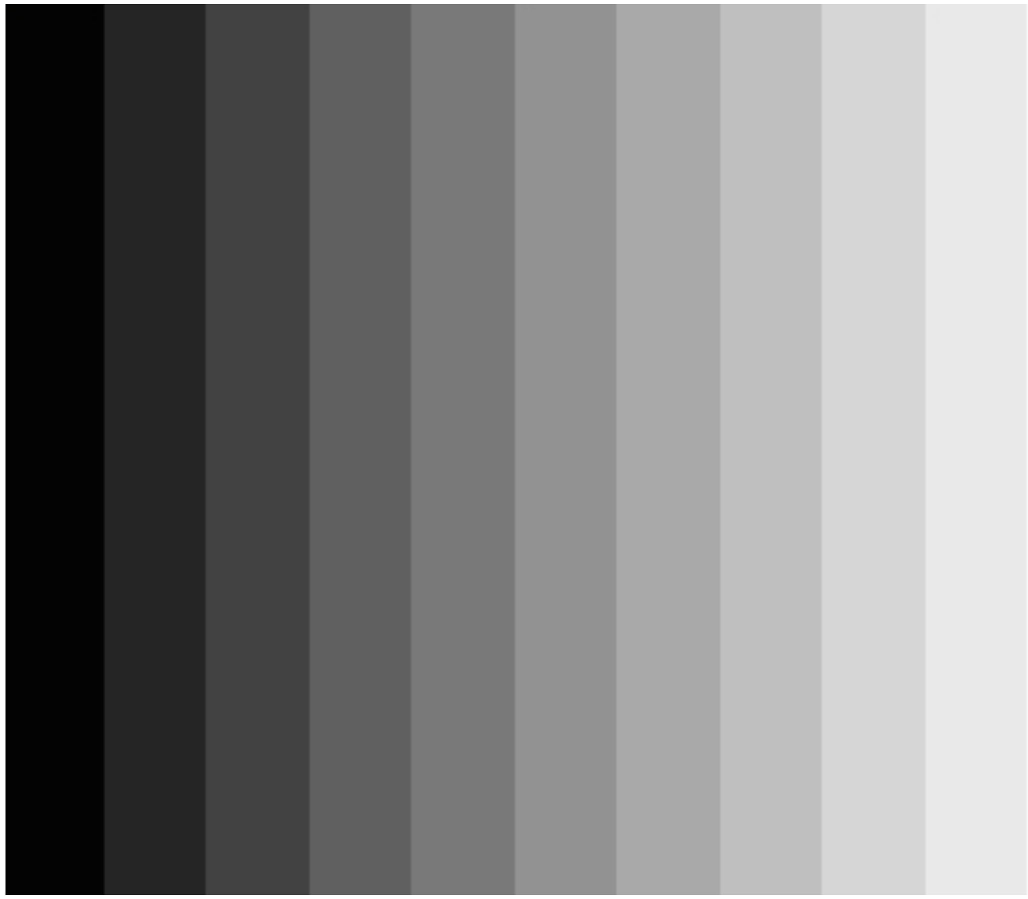
Source: Biederman

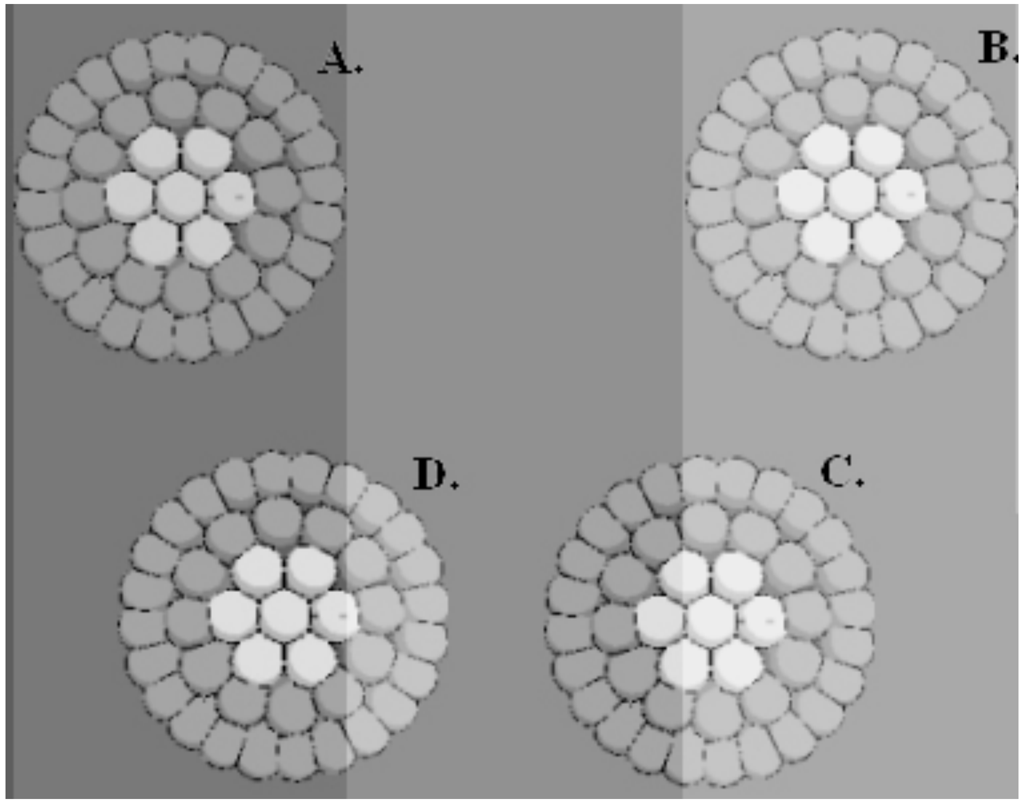
Figure 11. Nine of the experimental objects.



Source: Cavanagh '95

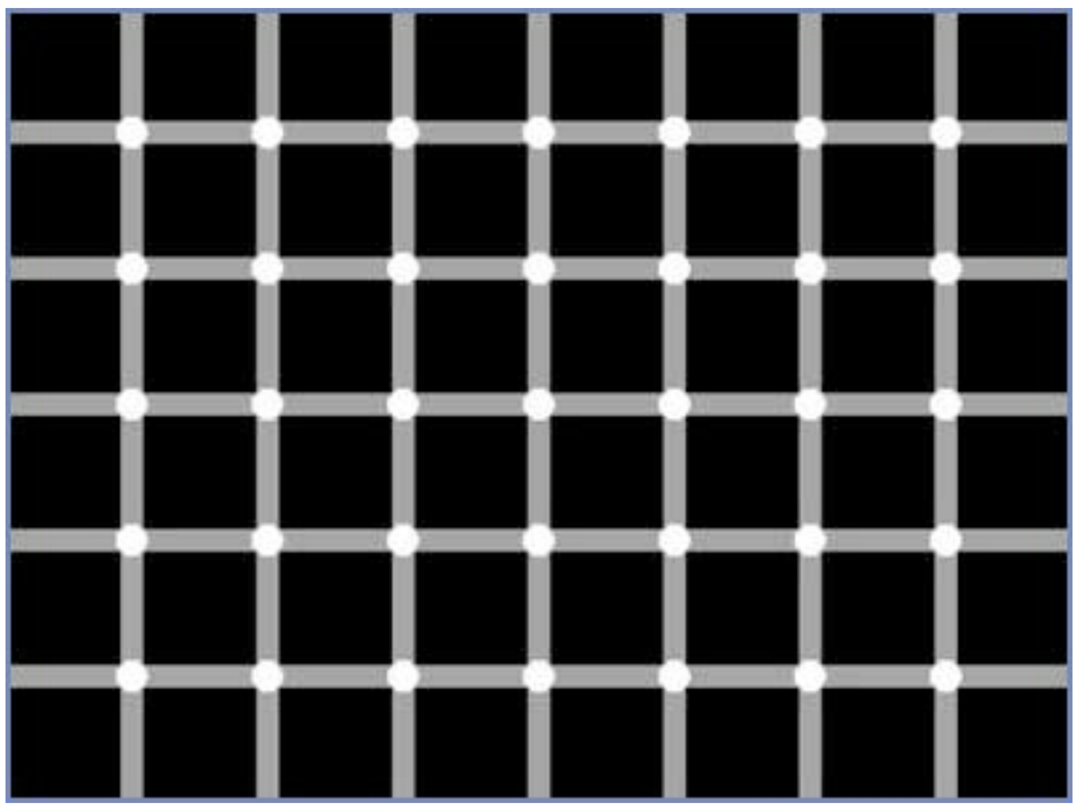
Illusions and center-surround processing

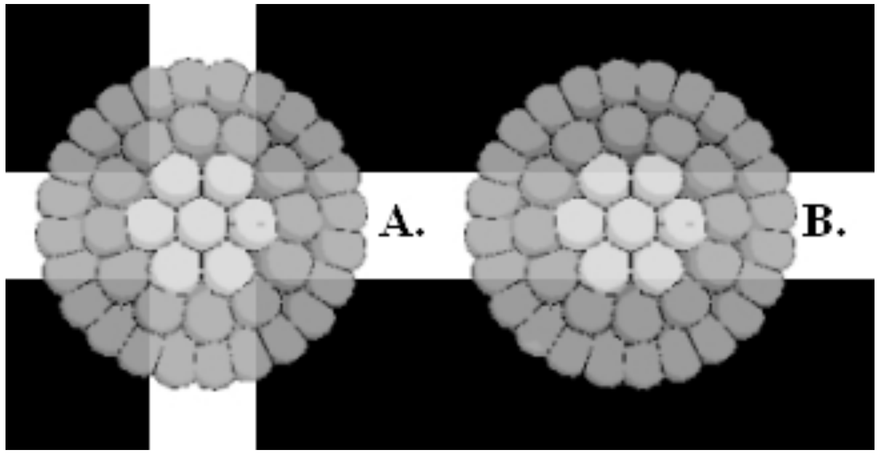




Source: http://www.nku.edu/~issues/illusions

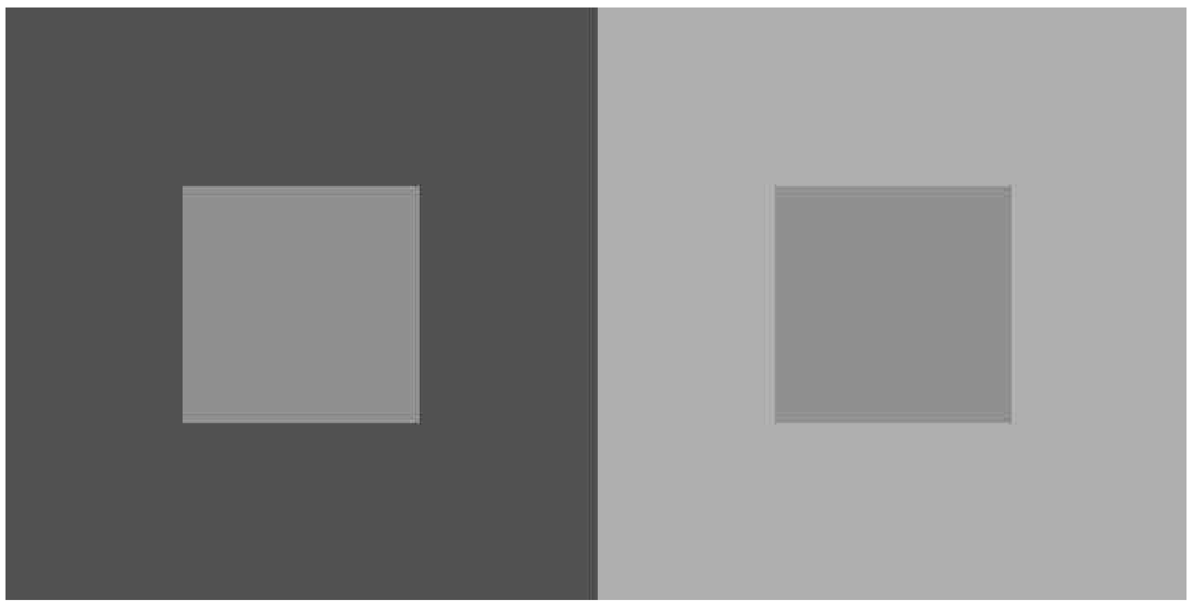
Illusions and center-surround processing

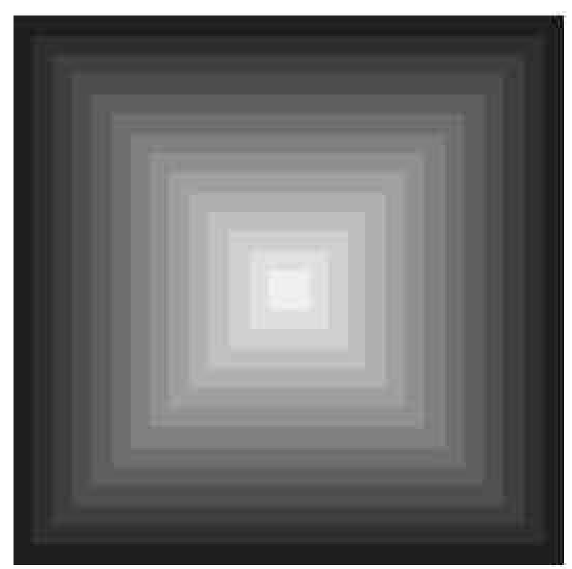




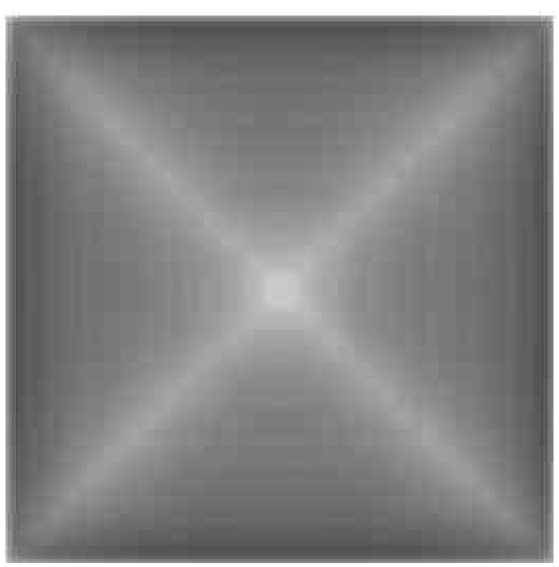
Source: http://www.nku.edu/~issues/illusions

Illusions and center-surround processing





a

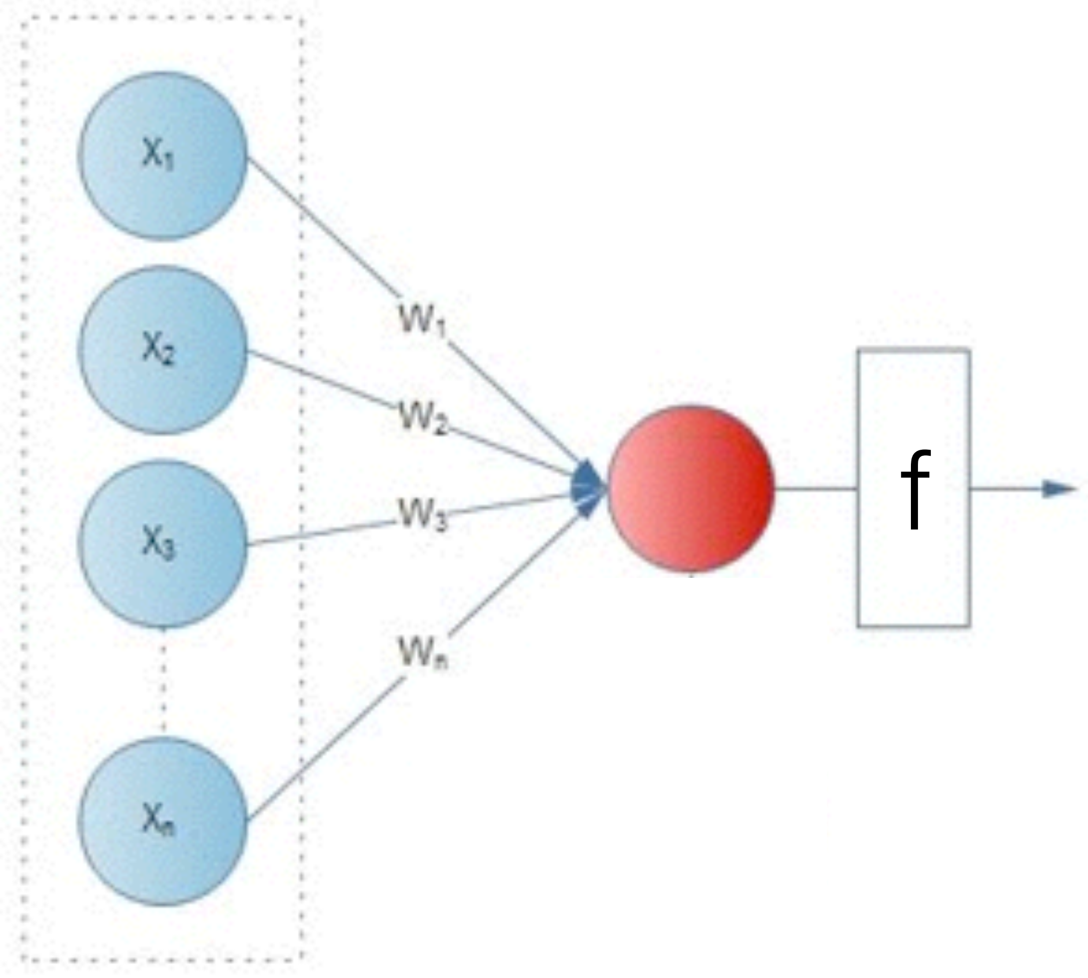


b

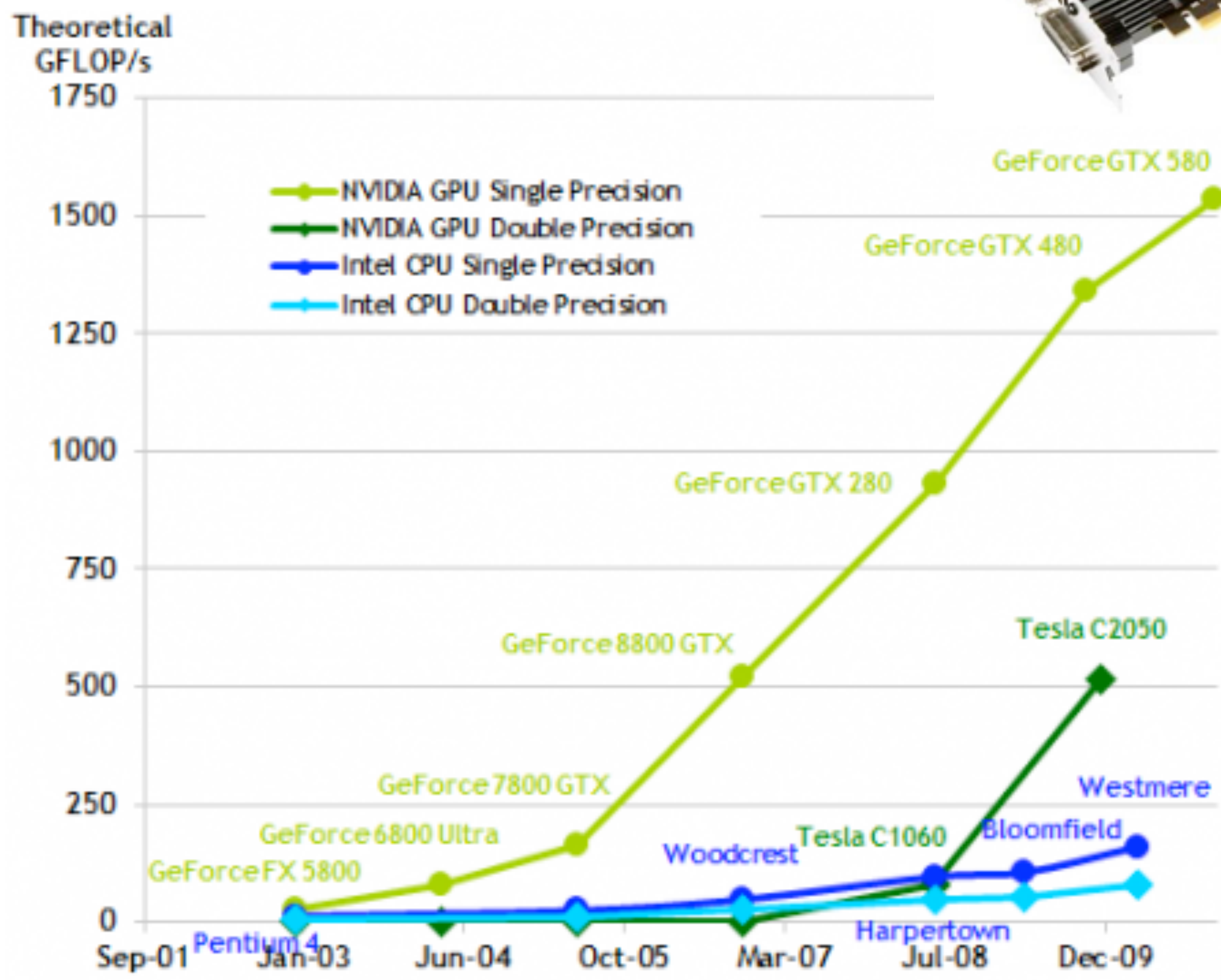
Source: Adelson (2000)

Computing with RFs: Summary

- Basic model of neural processing
- Reverse engineering computations by trying to interpret synaptic weights
- Filtering, convolution, preferred stimulus, template matching



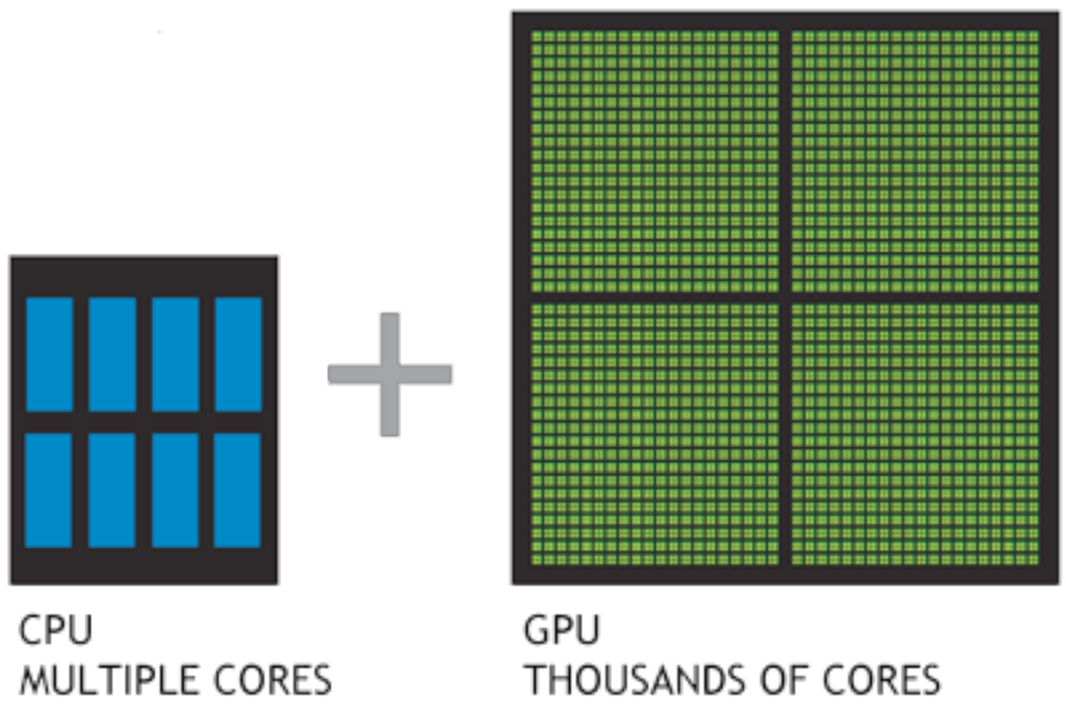
Graphics Processing Units



source: http://dual.sphysics.org/index.php/gpu/

Graphics Processing Units

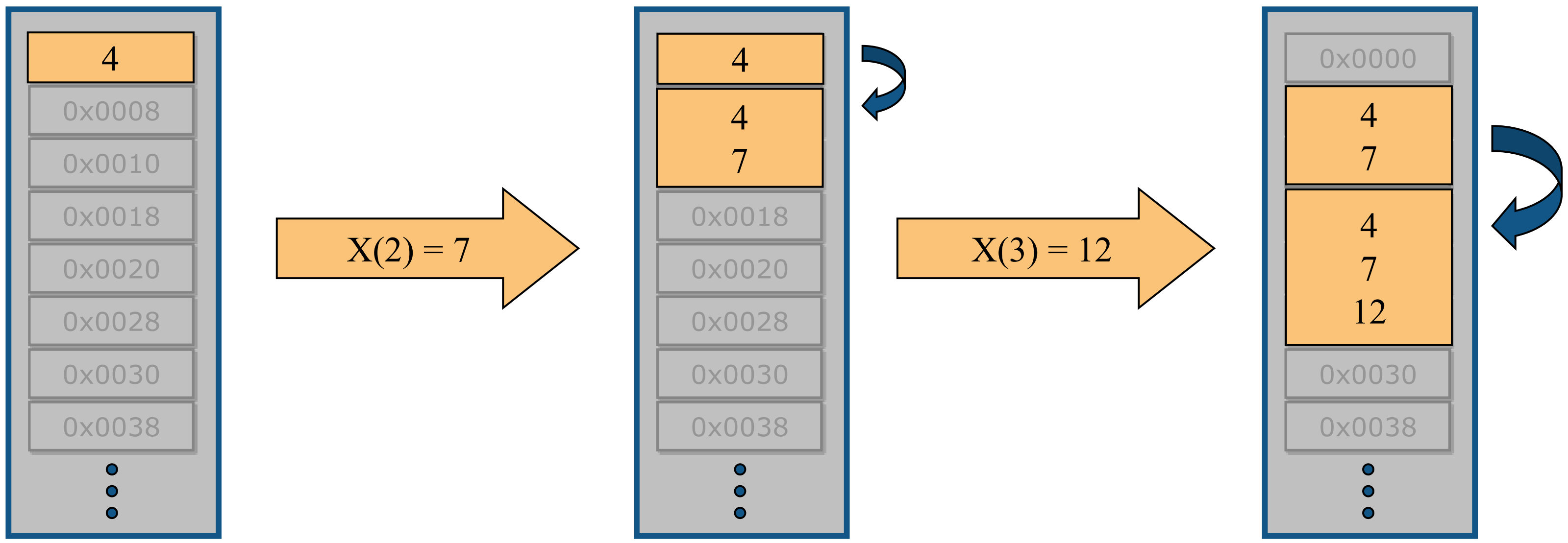
- Send Data to GPU:
    - M = magic(6);
    - G = gpuArray(M);
- Retrieve Data from GPU:
    - D = gather(G);
- Many built-In functions support for gpuArray (conv2, imfilter, etc)
- gpuarrayB = imfilter(gpuArrayA,h)



source: NVIDIA

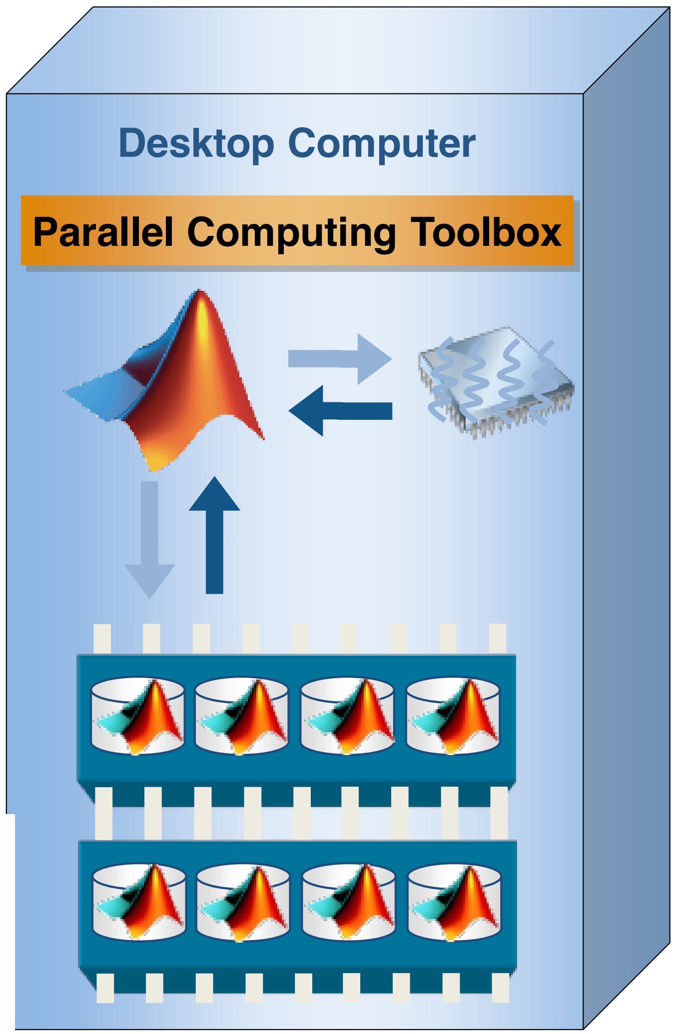
Speeding up MATLAB

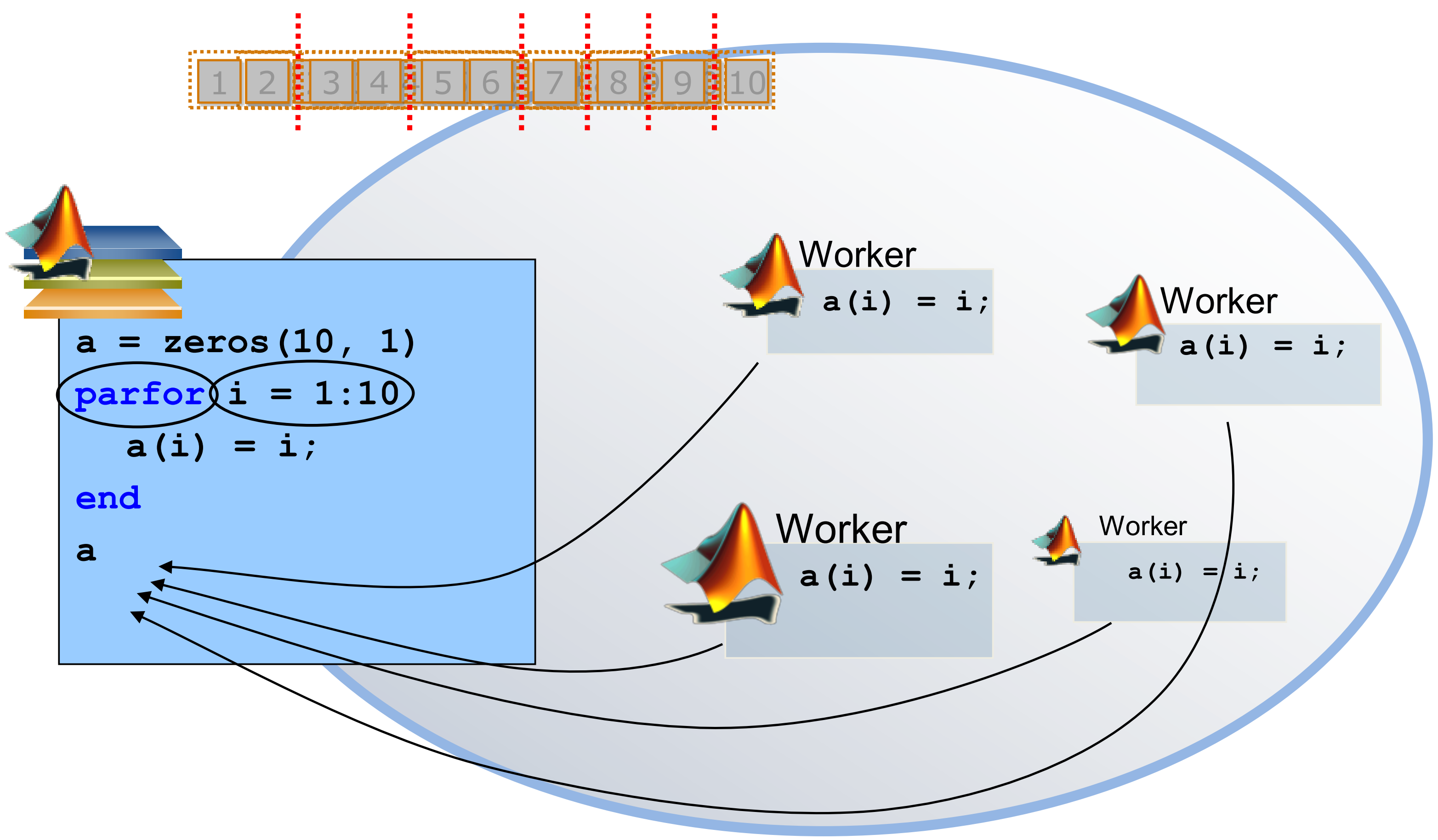
- Leveraging the power of vector & matrix operations
- Vectorize your code (MATLAB optimized for column / blo processing)
- Pre-allocate memory
- Timing functions (tic; toc;)



Speeding up MATLAB

- MATLAB Distributed Computing Server
  - matlabpool open 4
  - do stuff
  - matlabpool close





Pool of MATLAB Workers

Other best practices

- Minimize dynamically changing path
  - ‘addpath’+ ‘fullfile’, rather than ‘cd’
- Use the functional load syntax
  - x = load('myvars.mat') instead of just load('myvars.mat')
- Minimize changing variable class
  - x = 1;
  - xnew = 'hello'; instead of x = 'hello';
- File I/O
  - Disk is slow compared to RAM
- Displaying output
  - Creating new figures is expensive
  - Writing to command window is slow