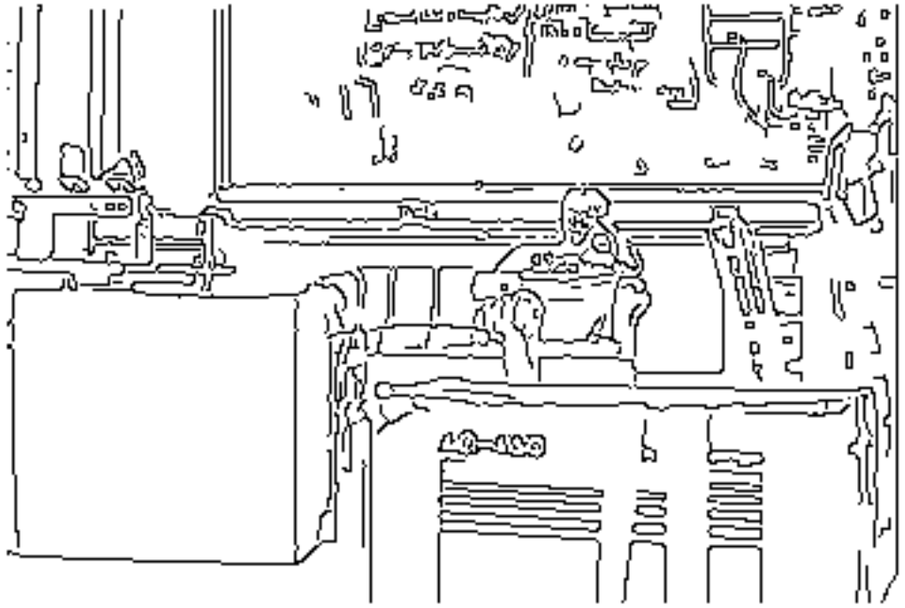
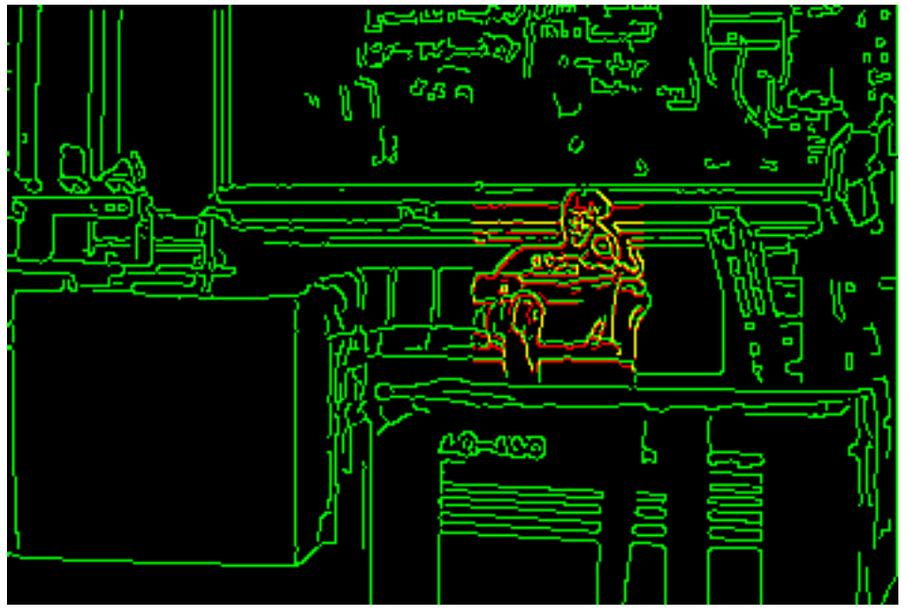
Filter Banks & Edges





Outline

- Logistics
- Edges
- Filter banks
- Efficiency (pyramids, separability, steerability)

Course website

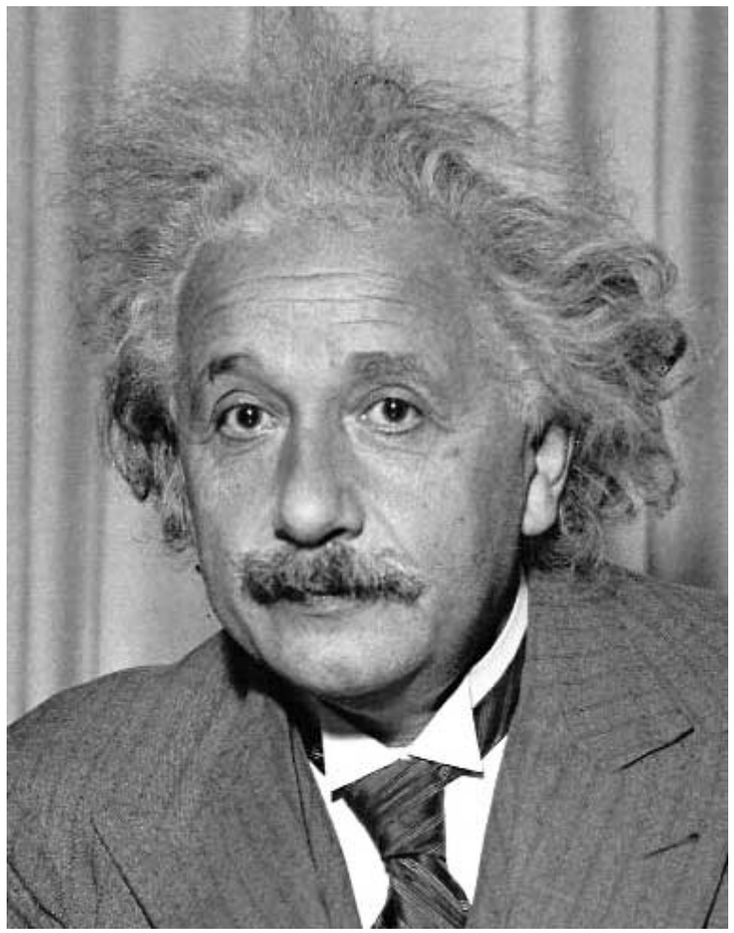
http://16720.courses.cs.cmu.edu/

HW1 out

Template matching with filters

H[i,j]

F[i,j]

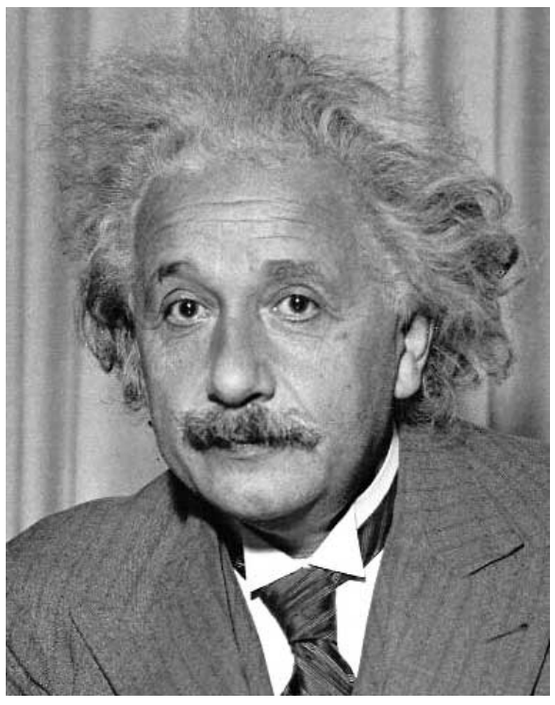


Can we use filtering to build detectors?

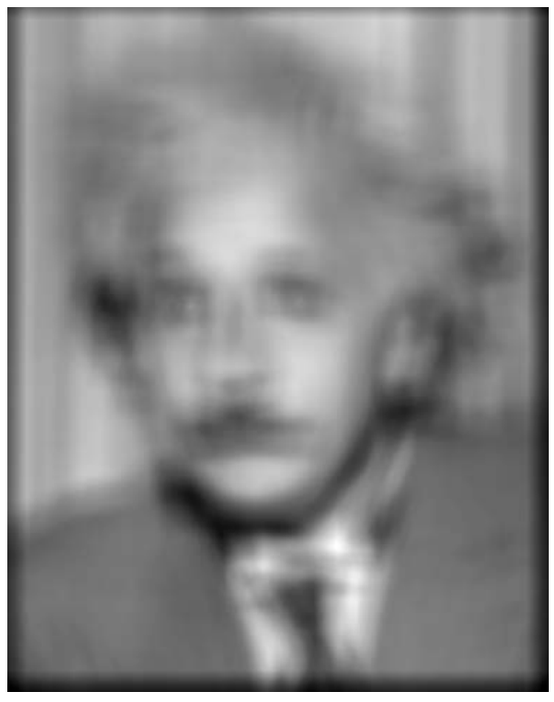
Attempt 1: correlate with eye patch

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

H



Input



Filtered Image

Attempt 1: correlate with eye patch

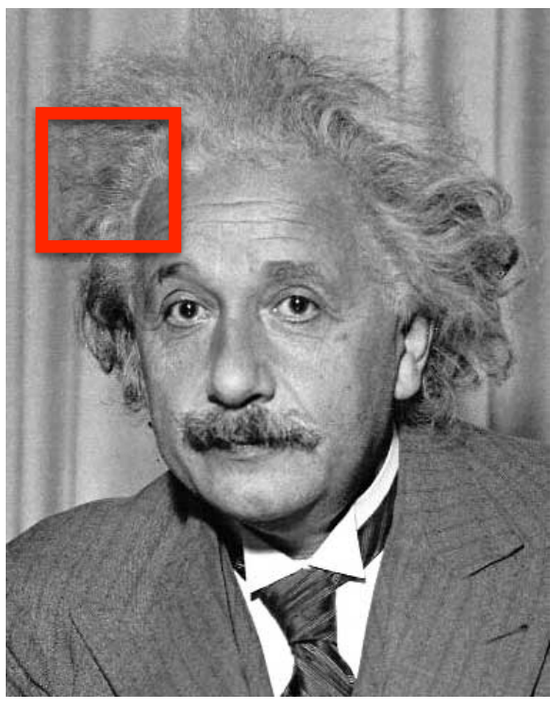
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

$$= H^T F_{ij} = ||H||||F_{ij}|| \cos\theta, \quad H, F_{ij} \in R^{(2K+1)^2}$$

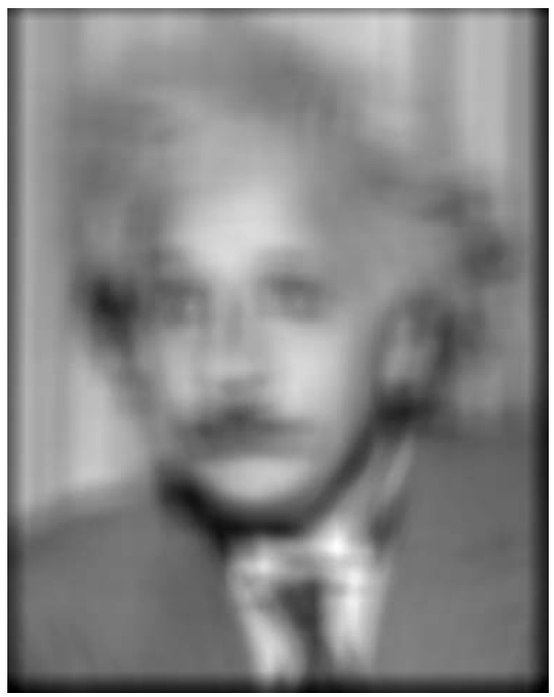
Useful to think about correlation and convolution

H

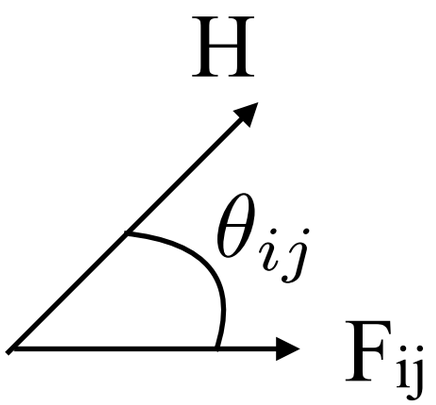
F_{ij}

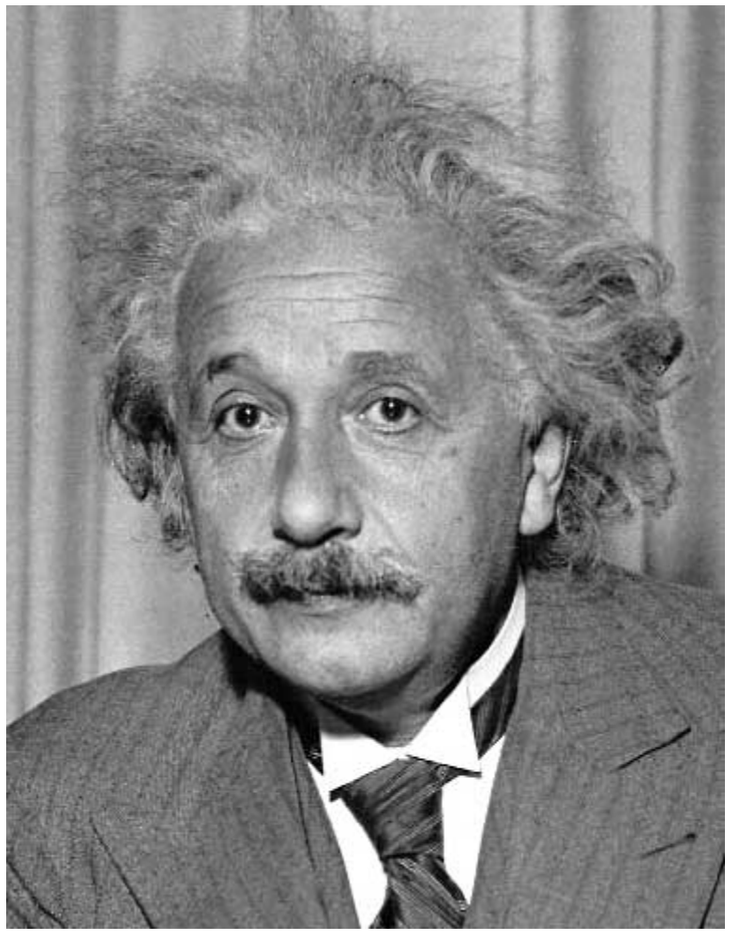


Input



Filtered Image

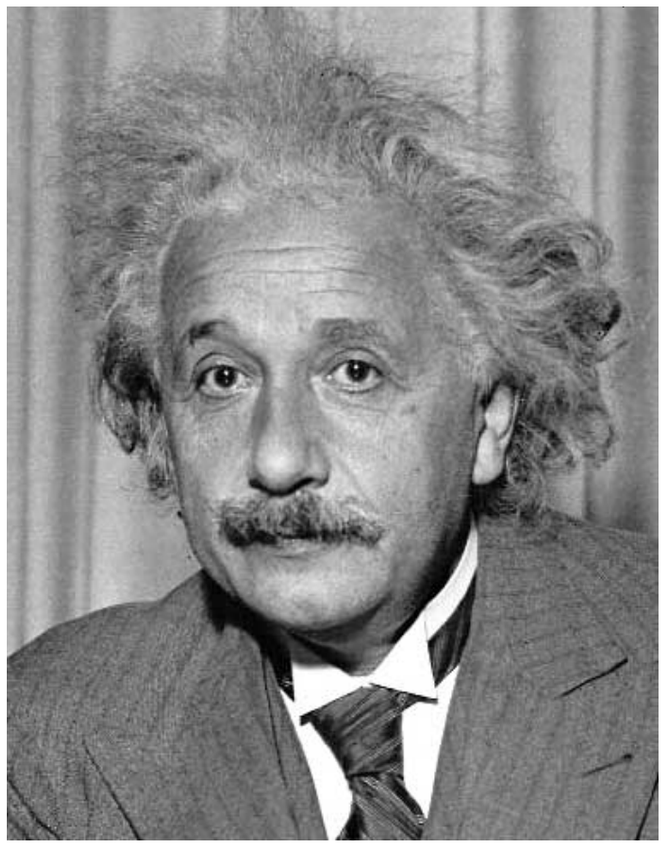




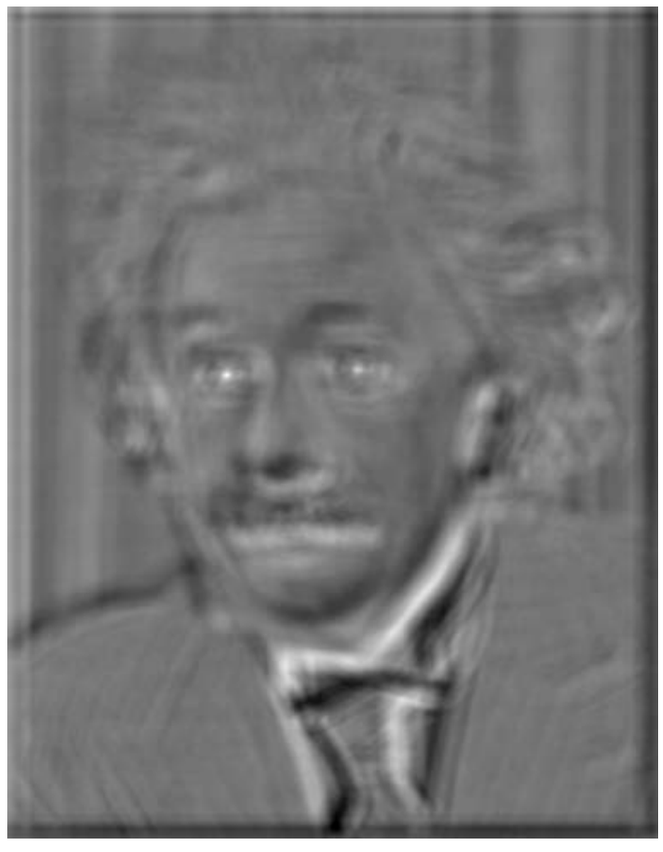
Let’s transform filter such that response on a flat region is 0

Attempt 1.5: correlate with zero-mean eye patch

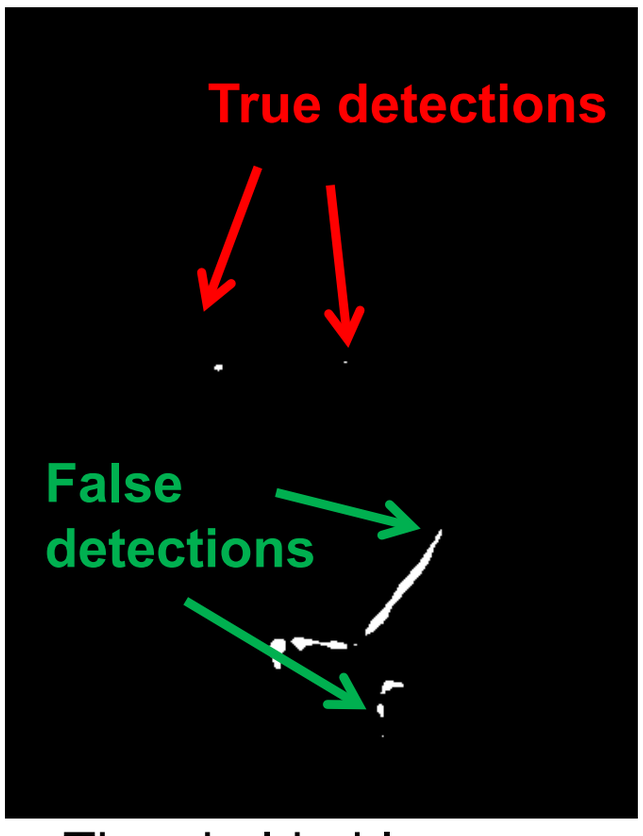
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} (H[u,v] - \bar{H})F[i+u, j+v]$$



Input



Filtered Image (scaled)

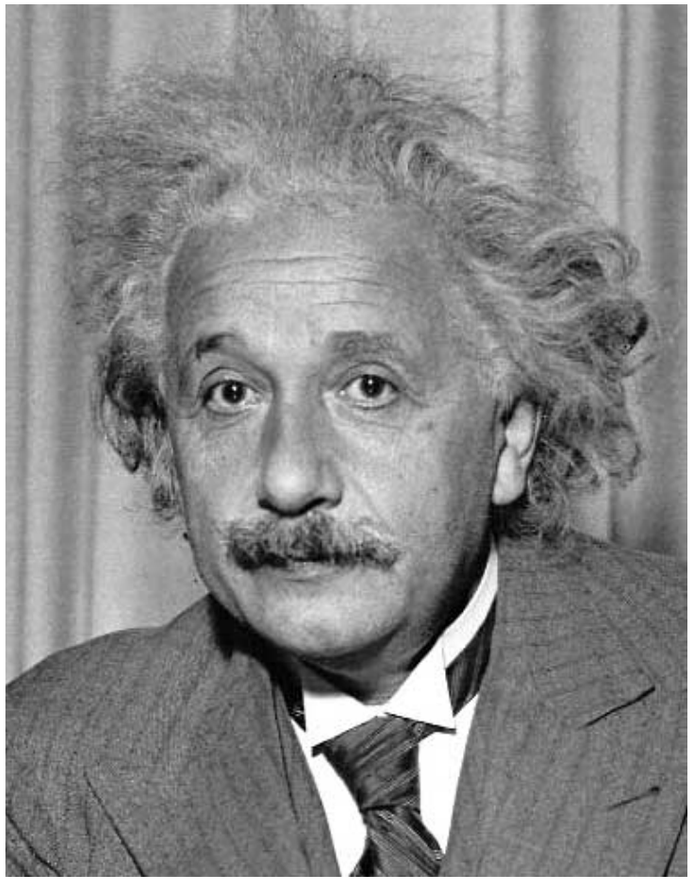


Thresholded Image

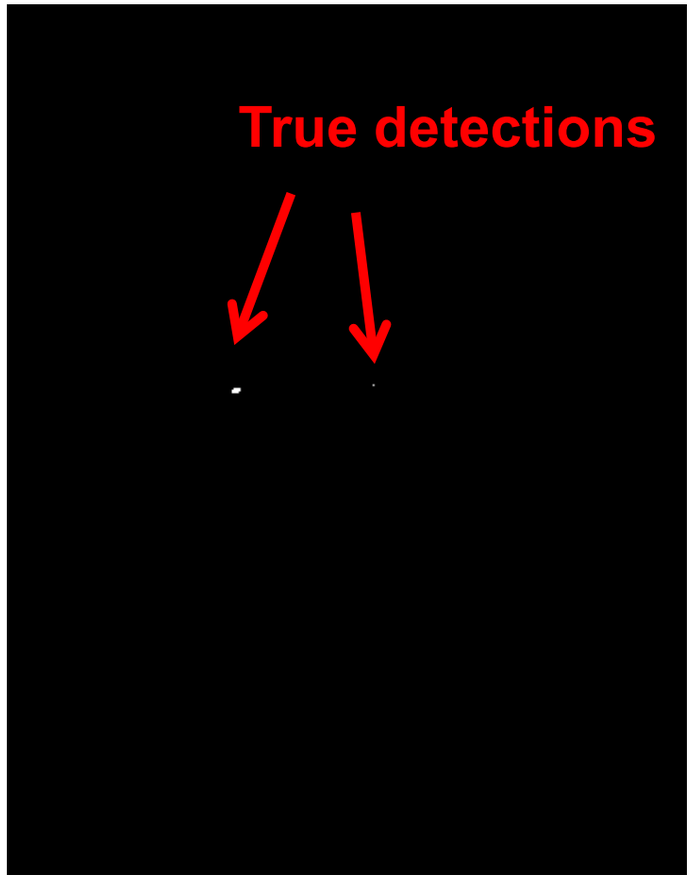
Attempt 2: SSD

$$SSD[i,j] = ||H - F_{ij}||^2$$
$$= (H - F_{ij})^T (H - F_{ij})$$

Can this be implemented with filtering?



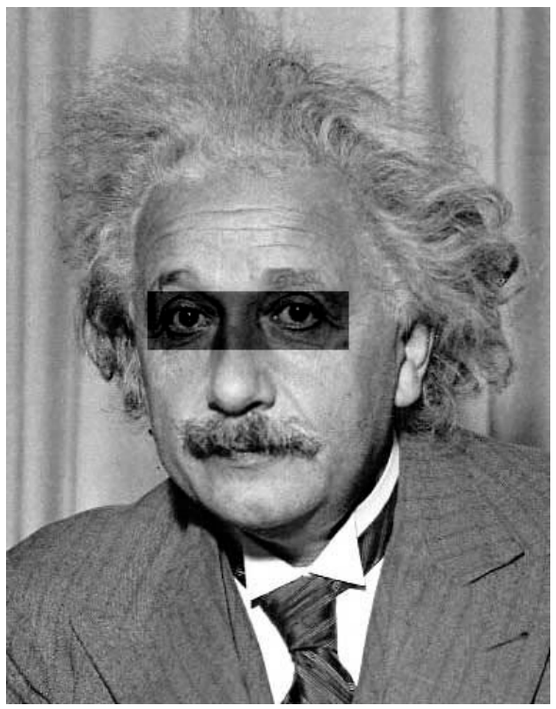
-SSD(patch,image)



Thresholded image

What will SSD find here?

-SSD(patch,image)



(where eyes have been darkened by .5 scale factor)

SSD will fire on shirt

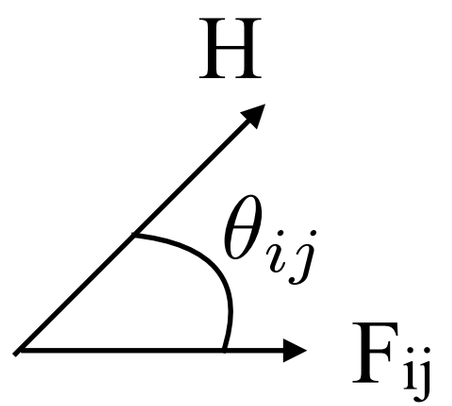
Normalized cross correlation

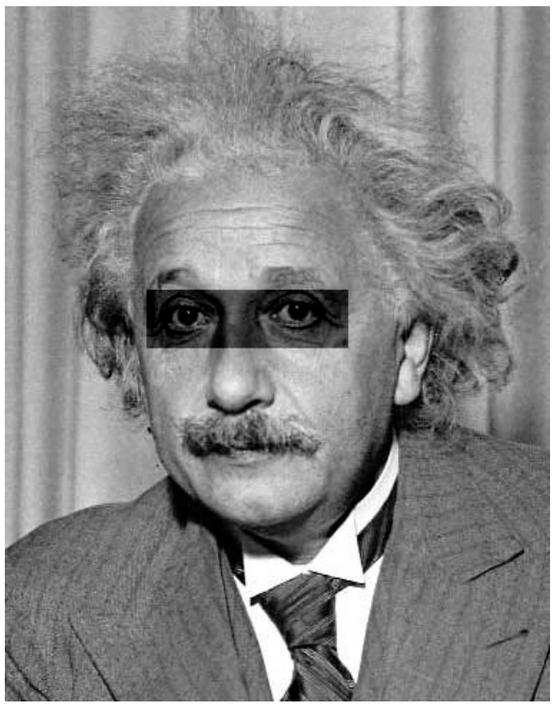
$$NCC[i,j] = \frac{H^T F_{ij}}{||H||||F_{ij}||}$$

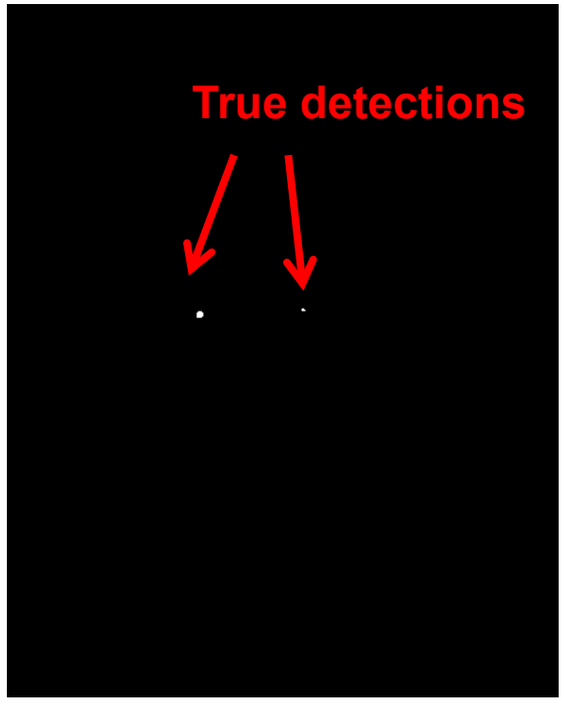
$$= \frac{H^T F_{ij}}{\sqrt{H^T H}\sqrt{F_{ij}^T F_{ij}}}$$

$$= \cos\theta$$

where H, F_{ij} are mean-centered



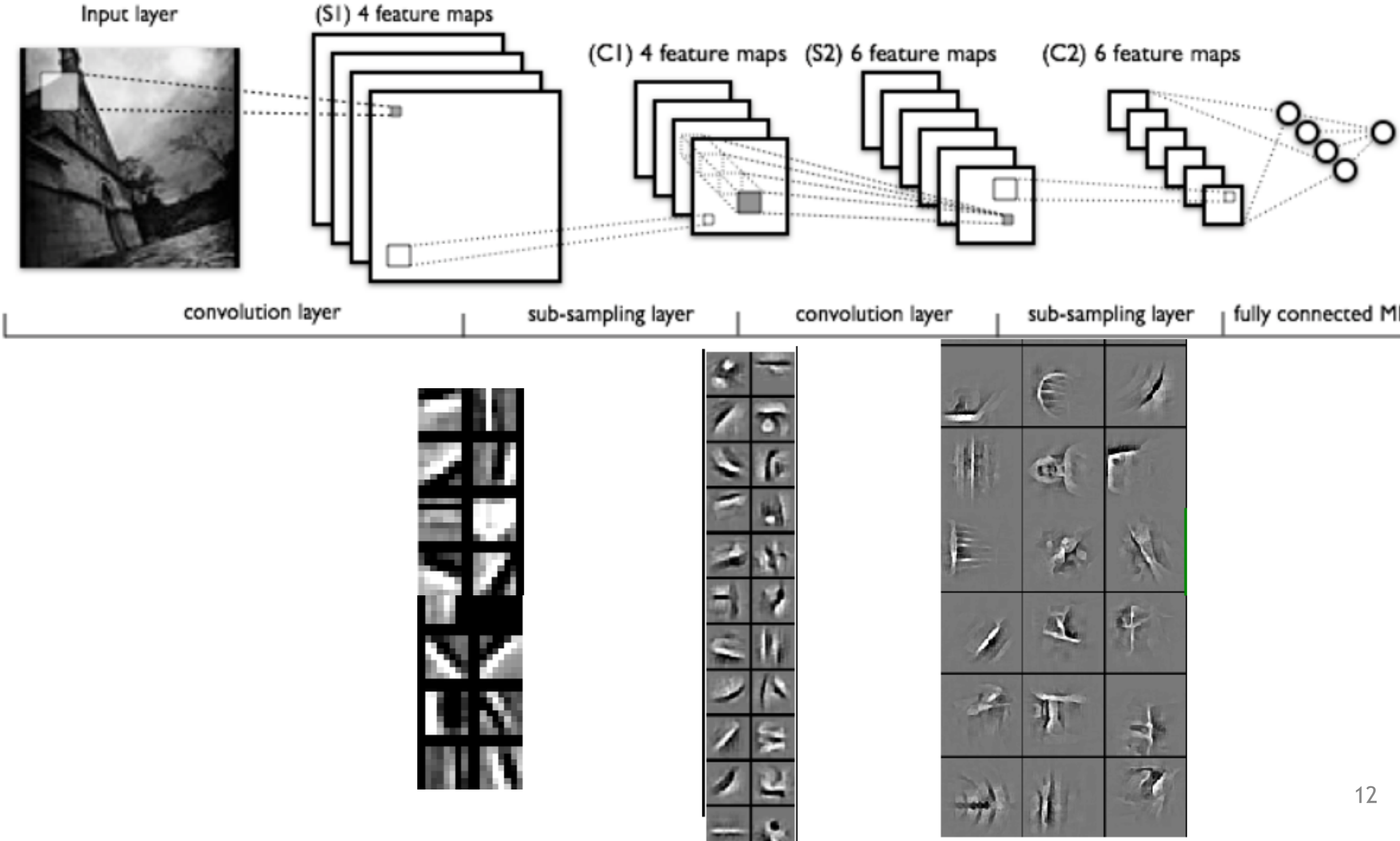




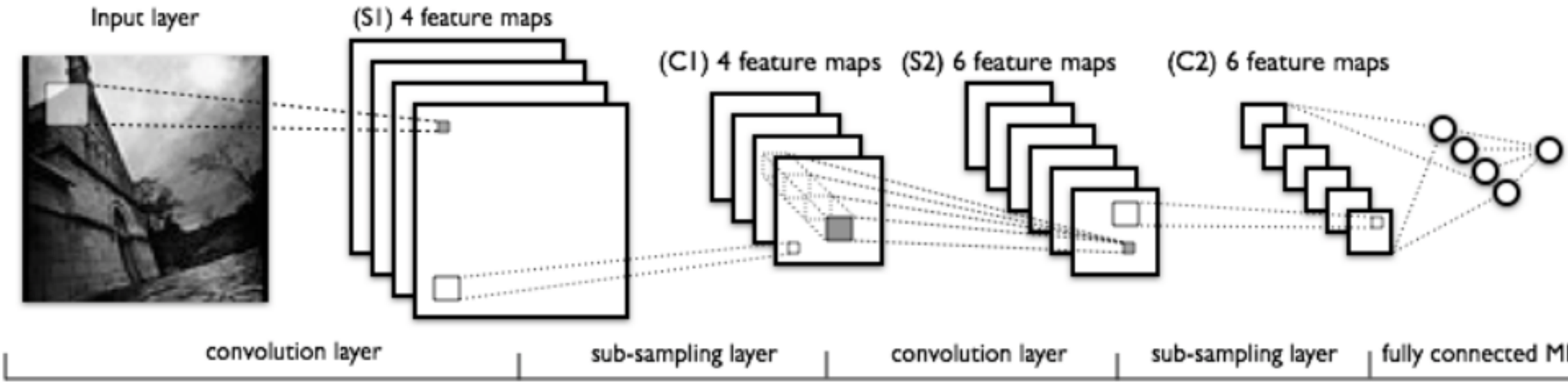
Modern filter banks

Convolutional Neural Nets (CNNs) Lecun et al 98

Learn filters from training data to look for low, mid, and high-level features



Convolutional neural nets



$$G = \max(0, F)$$

Elementwise rectification or “RELU” - rectified linear unit

Is this operation linear shift-invariant?

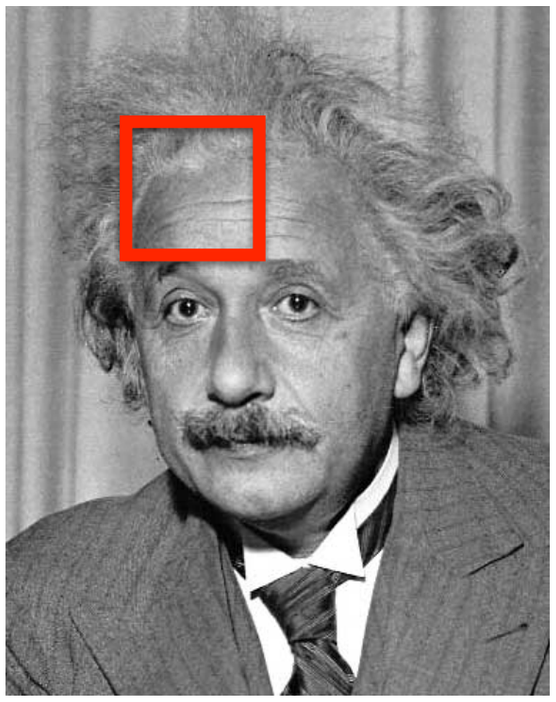
Theoretical analysis that suggests “homogenous” transformations are easy to learn (local minima = global minima)

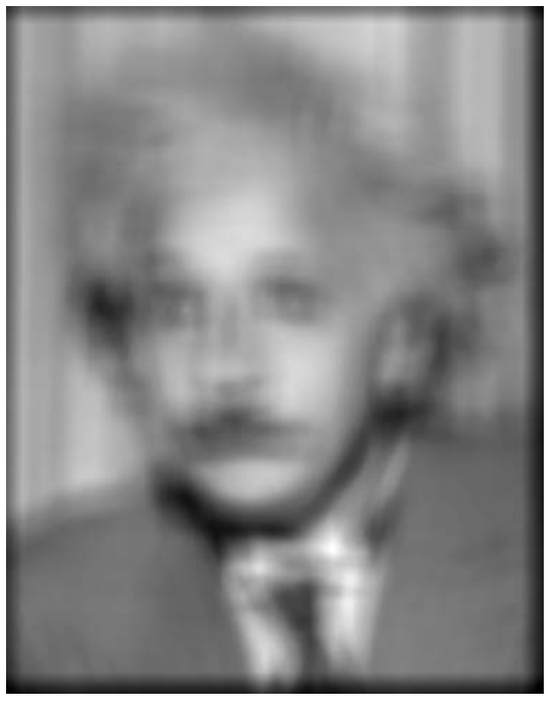
https://arxiv.org/abs/1506.07540

SSD vs CC vs NCC

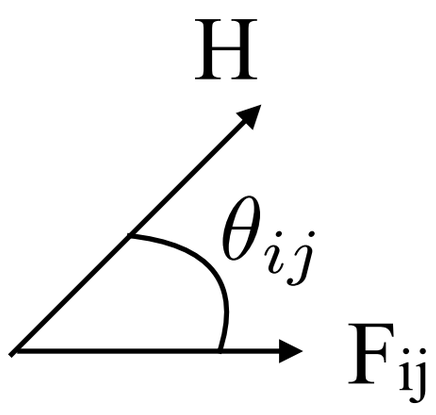
F_{ij}

H





Treat patches (H,F_{ij}) as vectors in R^N



$$SSD_{ij} = ||H - F_{ij}||^2$$
$$CC_{ij} = H^T F_{ij}$$
$$NCC_{ij} = \cos(\theta_{ij})$$

A look back

$$SSD_{ij} = ||H - F_{ij}||^2 = H^T H - 2H^T F_{ij} + F_{ij}^T F_{ij}$$
$$CC_{ij} = H^T F_{ij}$$
$$NCC_{ij} = \frac{H^T F_{ij}}{\sqrt{H^T H}\sqrt{F_{ij}^T F_{ij}}}$$

1. When would peaks of SSD align with peaks of CC ?

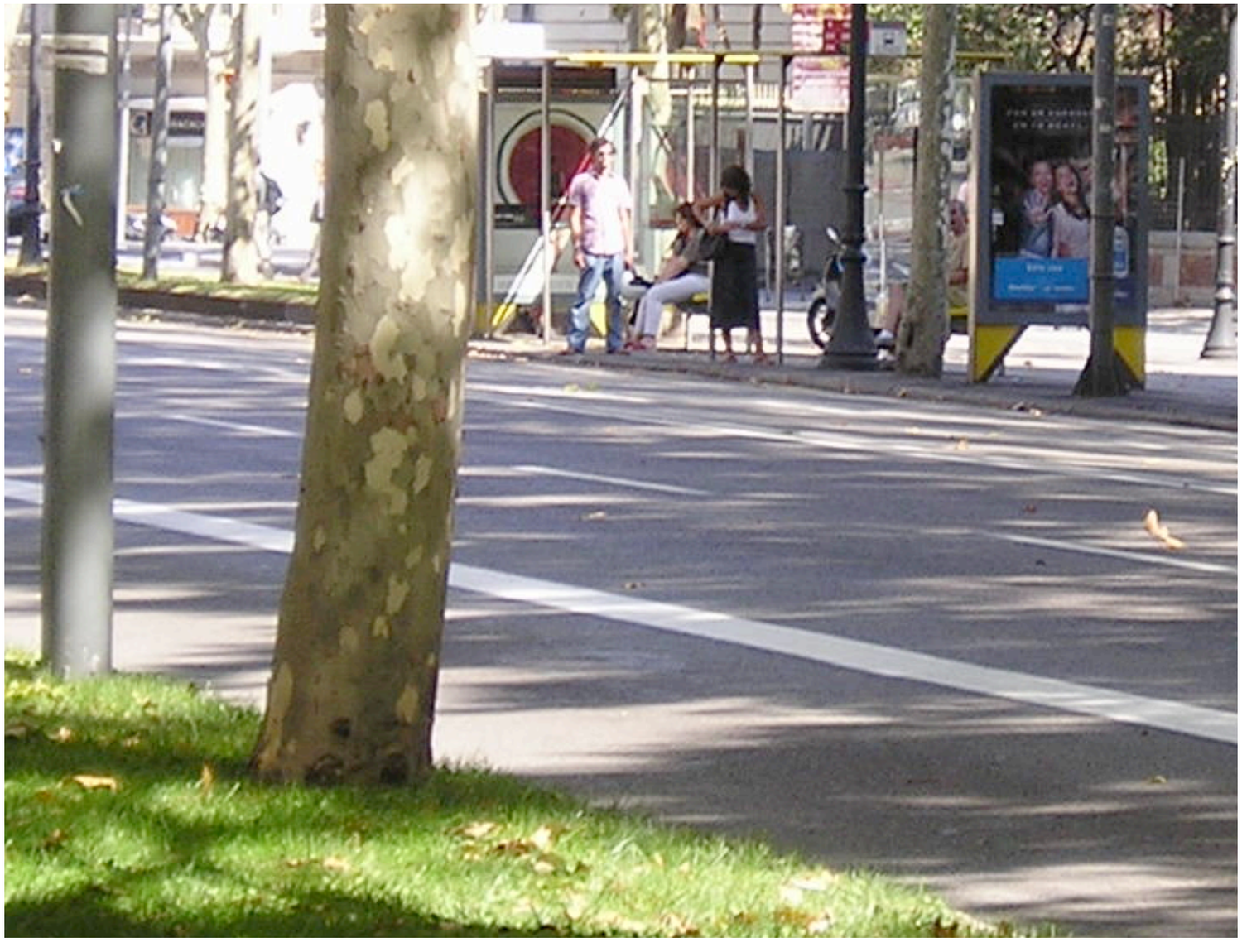
2. When would NCC outperform SSD?

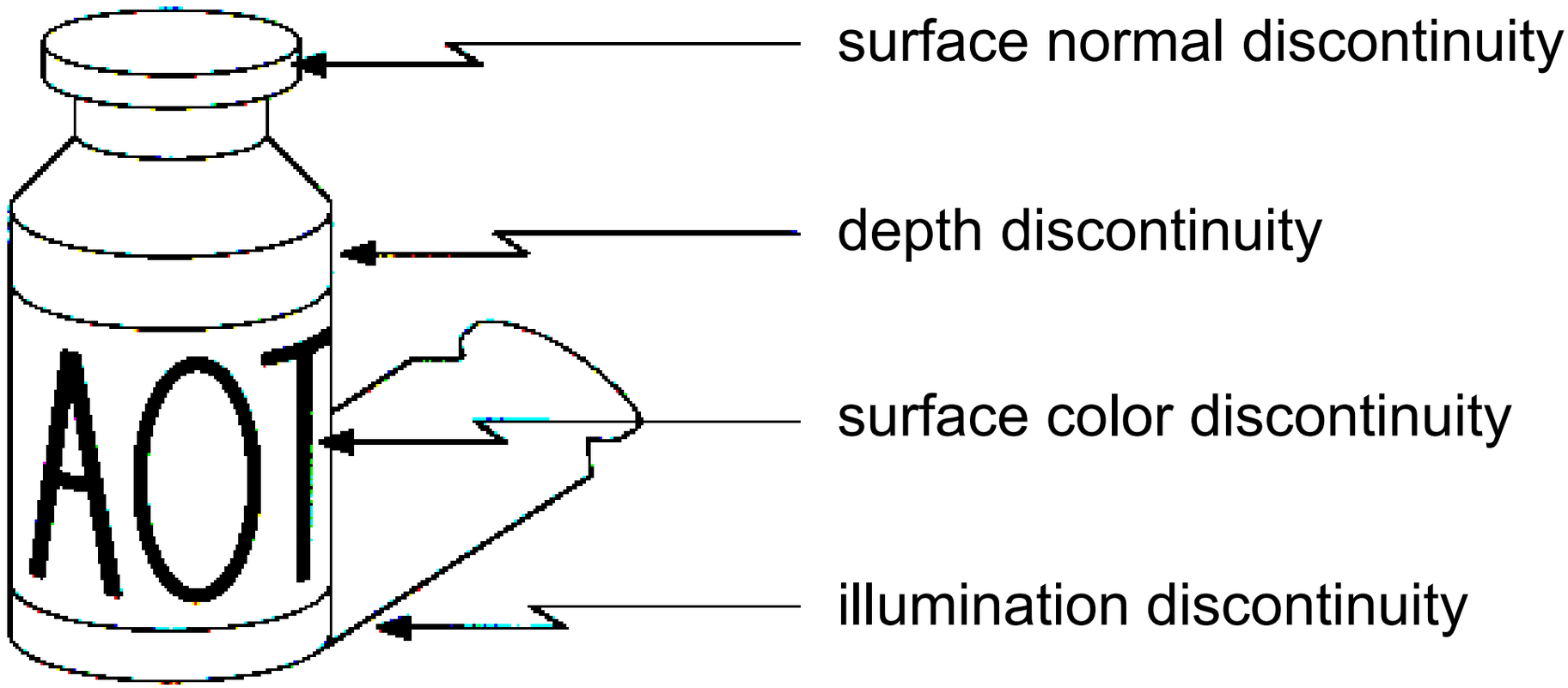
3. How can we compute SSD, NCC with pointwise operations and filtering?

Outline

- Logistics
- Edges
- Filter banks
- Efficiency (pyramids, separability, steerability)

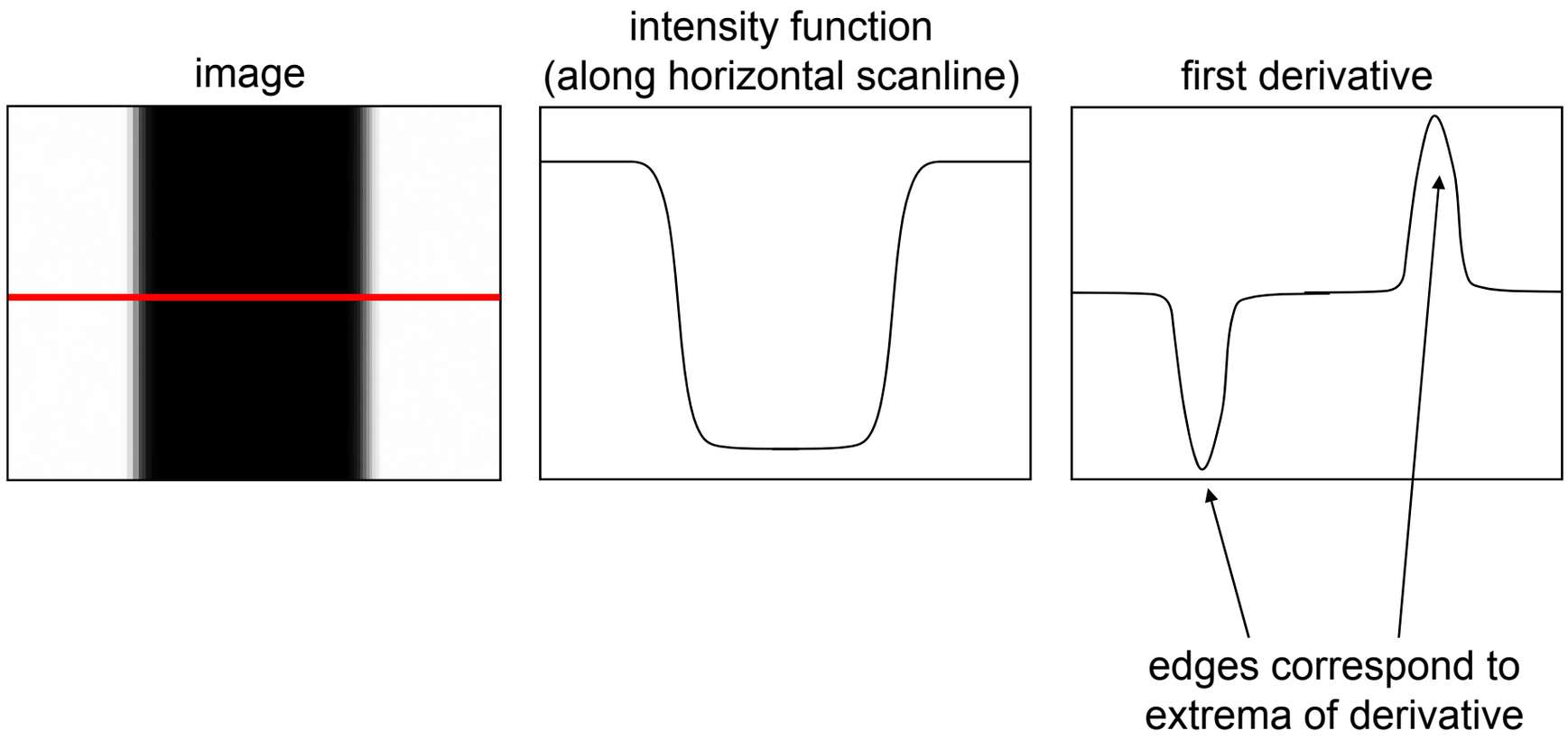
What causes edges?



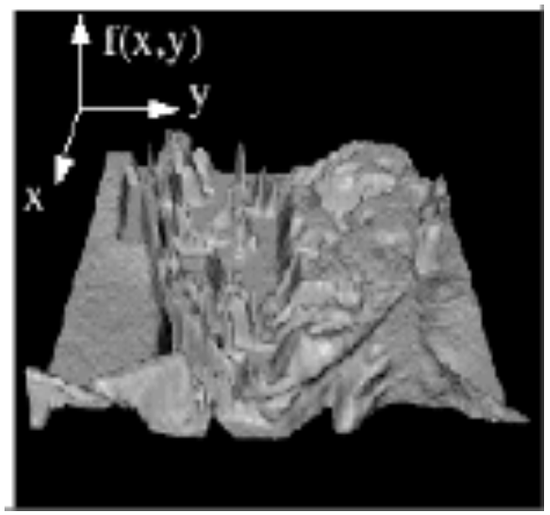

surface normal discontinuity
depth discontinuity
surface color discontinuity
illumination discontinuity

Characterizing edges

- An edge is a place of rapid change in the image intensity function



Continuous derivatives



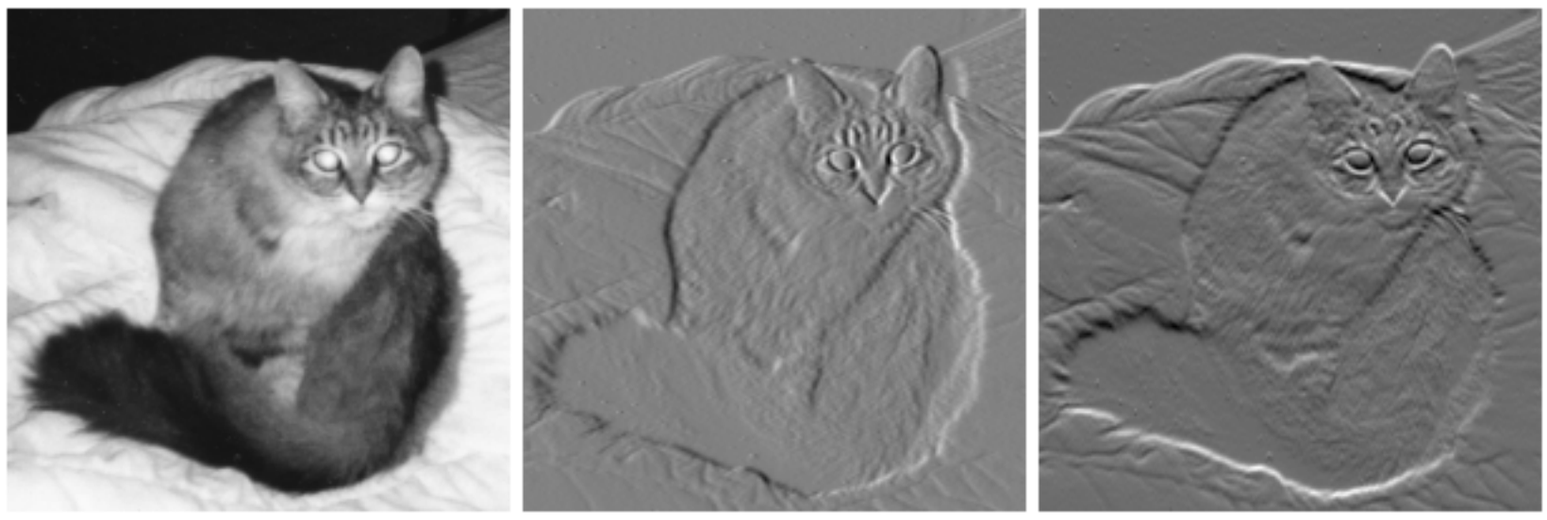
$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

$$||\nabla f|| = \sqrt{f_x^2 + f_y^2}$$

$$\alpha = \tan^{-1} \frac{f_y}{f_x}$$

Derivative filters

$$\begin{bmatrix} -1 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$



Other approximations

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

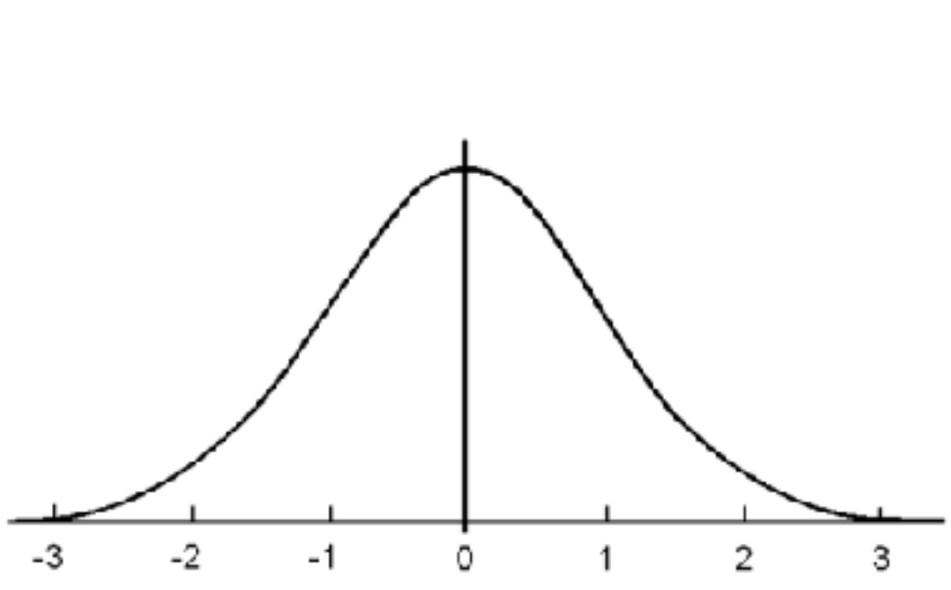
Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

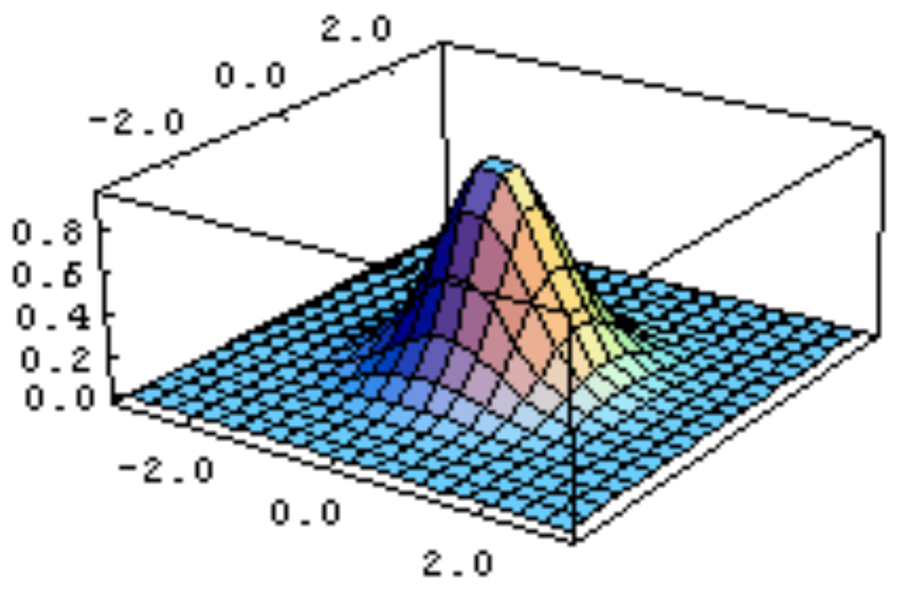
Why might these work better?

Recall: Gaussian smoother

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

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



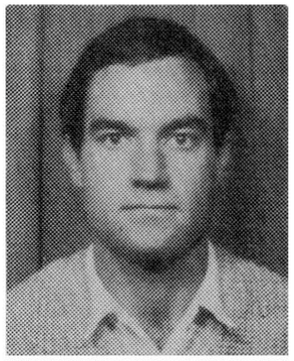


$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

A Computational Approach to Edge Detection

JOHN CANNY, MEMBER, IEEE



John Canny (S'81–M'82) was born in Adelaide, Australia, in 1958. He received the B.Sc. degree in computer science and the B.E. degree from Adelaide University in 1980 and 1981, respectively, and the S.M. degree from the Massachusetts Institute of Technology, Cambridge, in 1983.

He is with the Artificial Intelligence Laboratory, M.I.T. His research interests include low-level vision, model-based vision, motion planning for robots, and computer algebra.

Mr. Canny is a student member of the Association for Computing Machinery.

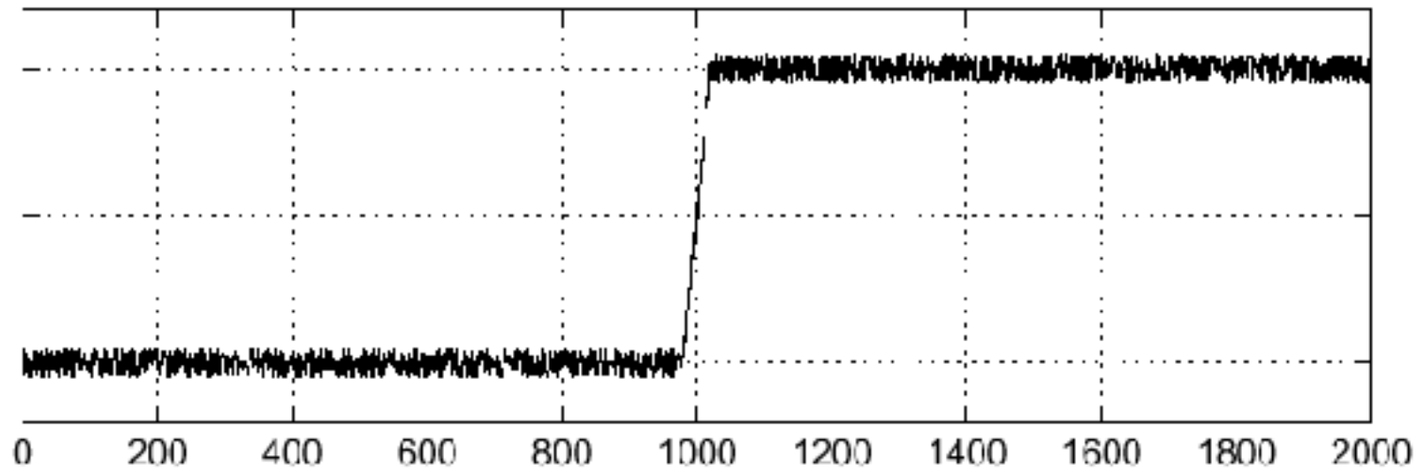
Criteria: we want to detect and localize edges

Approach: start with model (ideal step edge + Gaussian noise) and come up with optimal solution

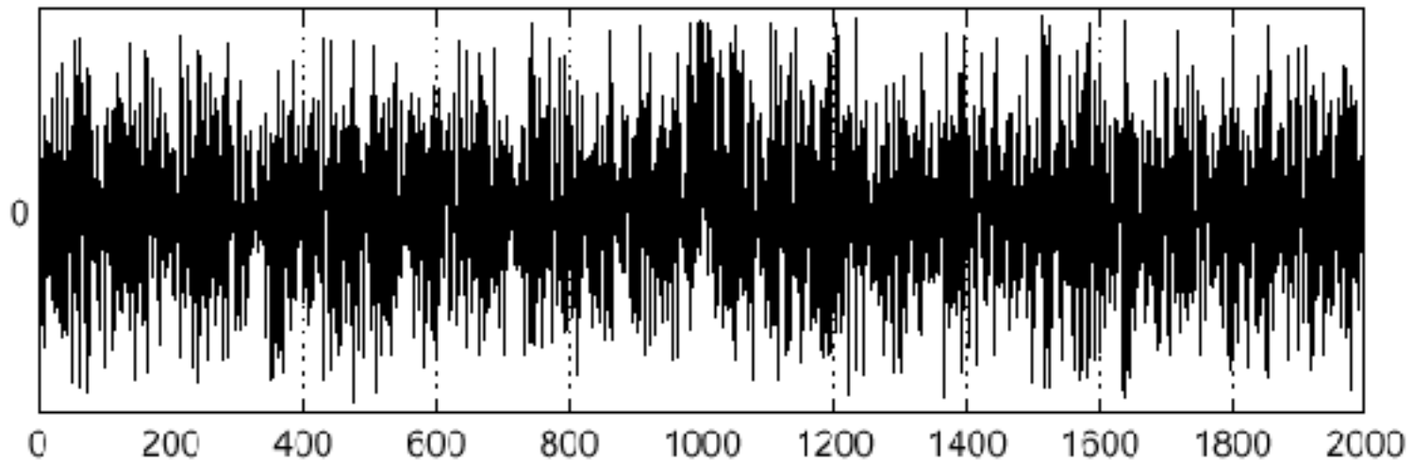
Revisiting gradients

(let's plot a single row of image as a function)

F[i]

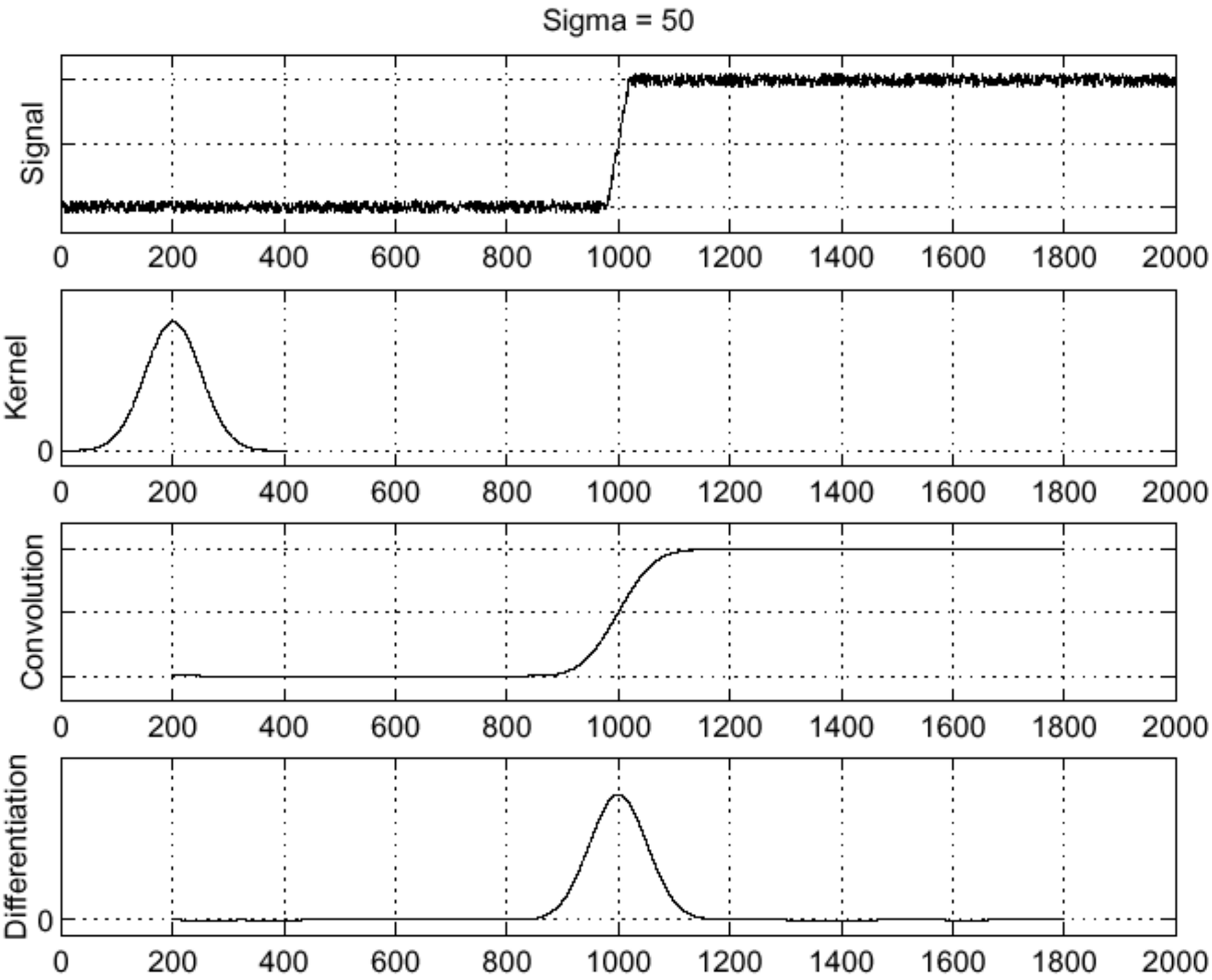


$$\frac{\partial F}{\partial i} \approx F \otimes [-1, 1]$$
$$= F * [1, -1]$$

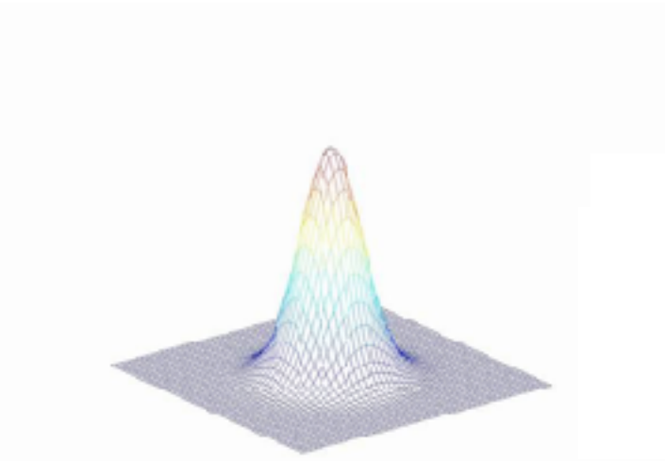


Solution: smooth first

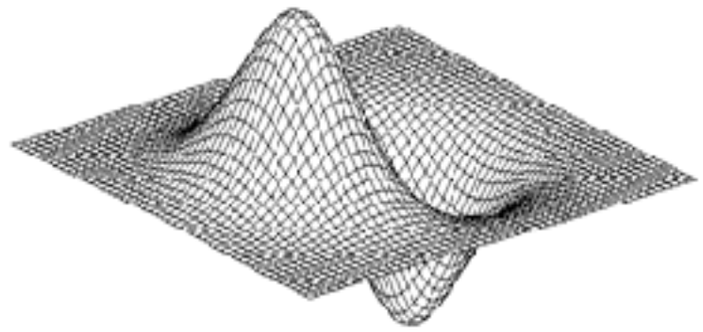
F



Derivative of Gaussian filter

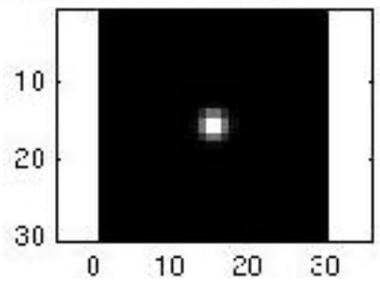


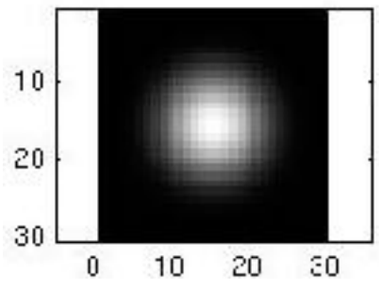
$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * [-1, 1] =$$



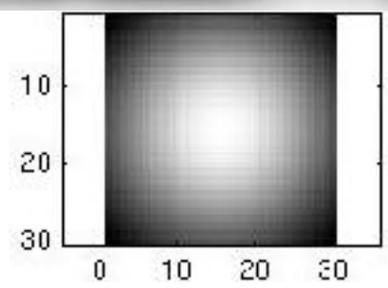
Effect of σ on Gaussian smoothing

Recall: parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.





...



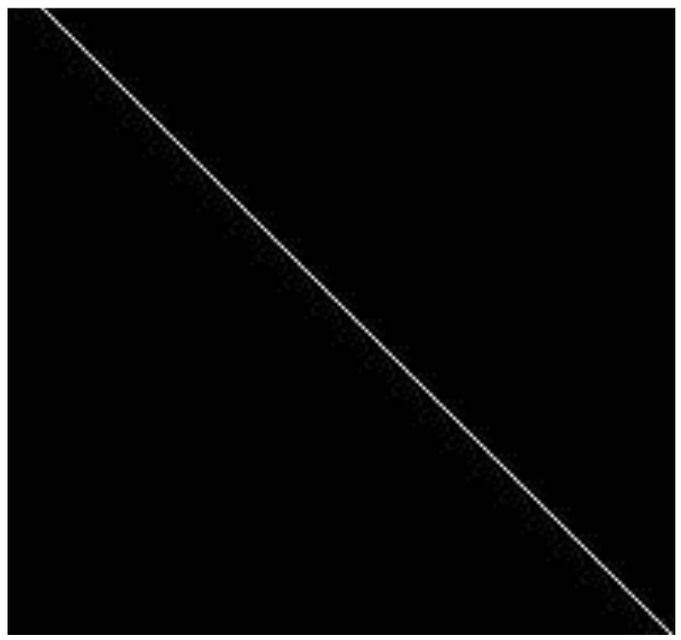
Effect of σ on derivatives

The apparent structures differ depending on Gaussian's scale parameter.

Larger values: larger scale edges detected
Smaller values: finer features detected

Fundamental tradeoff between smoothing and good localization!

Image with Edge



Edge Location

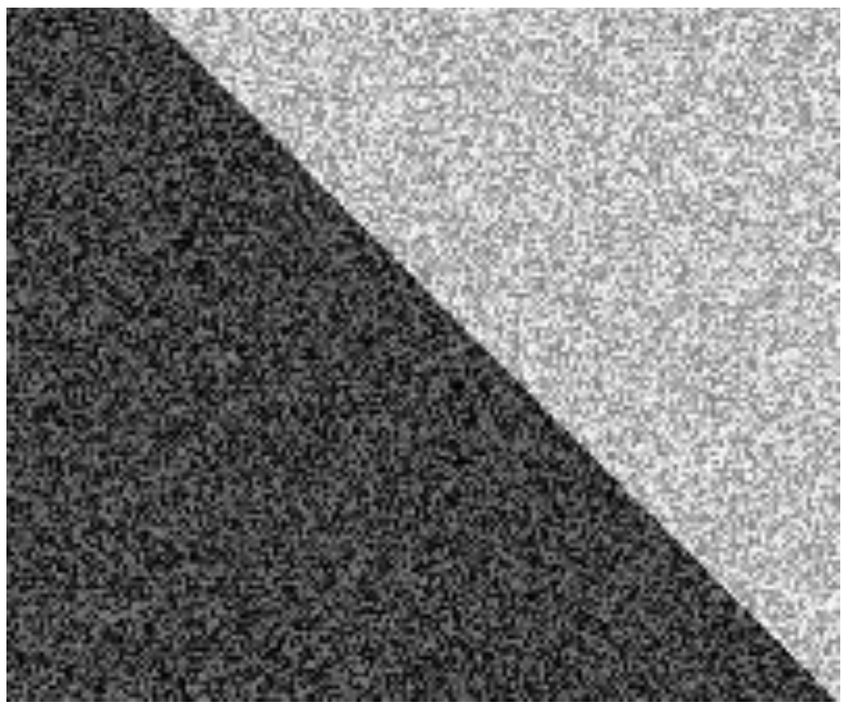
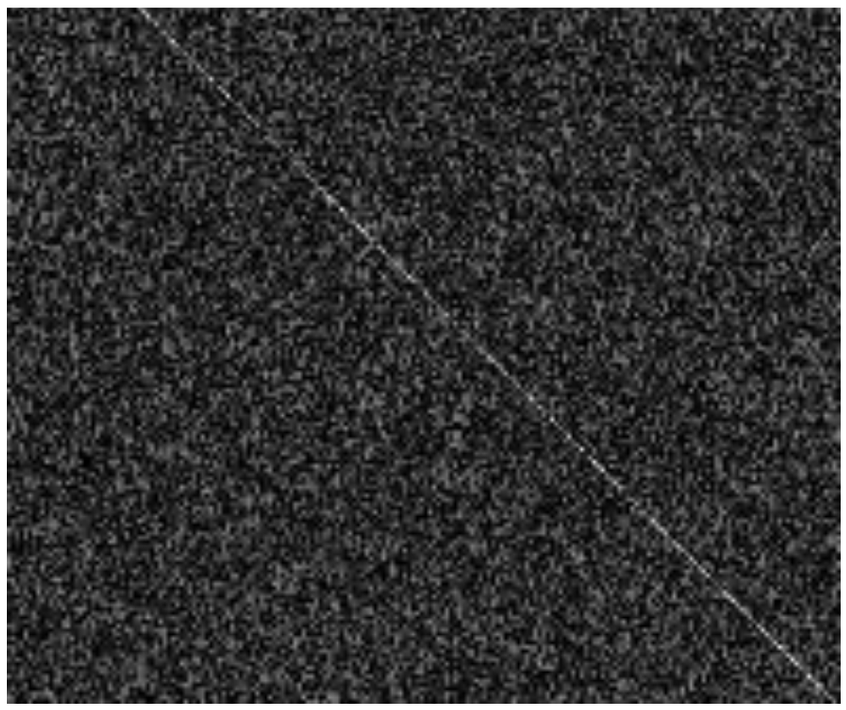
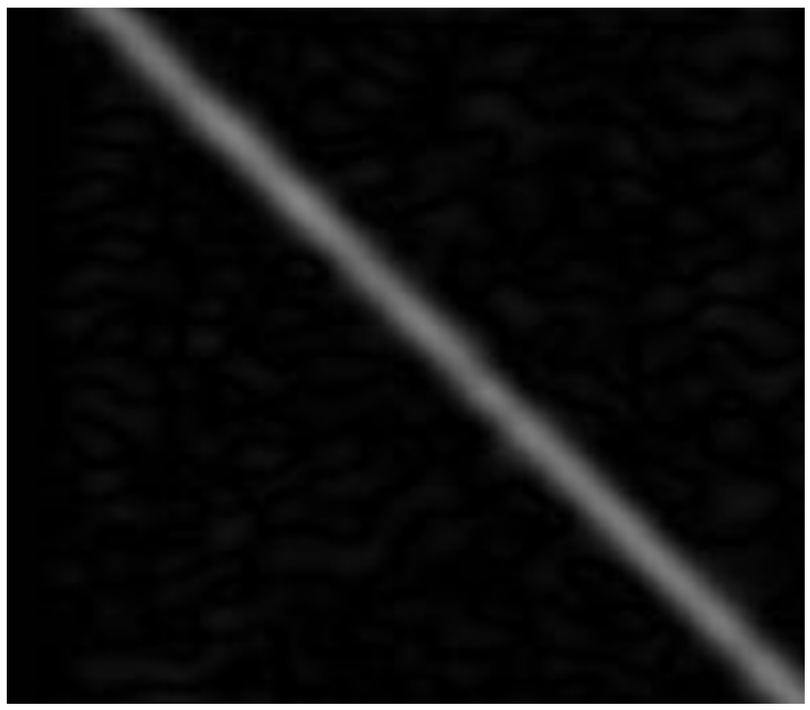


Image + Noise



Derivatives detect edge *and* noise



Smoothed derivative removes noise, but blurs edge

The Canny edge detector

original image (Lena)

Source: S. Seitz

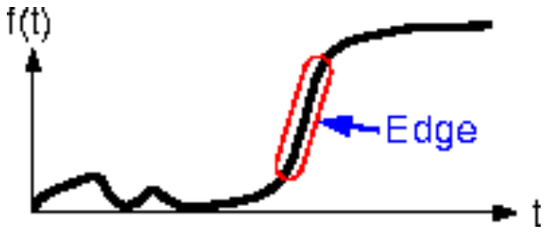
The Canny edge detector

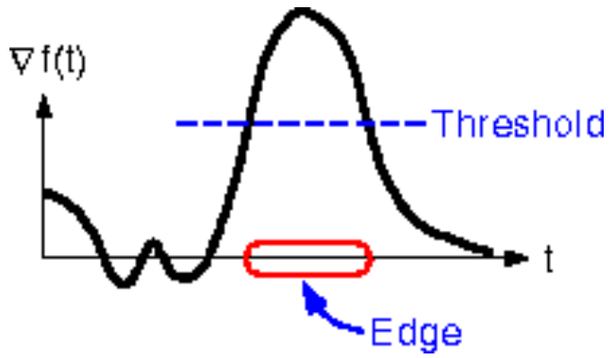
norm of the gradient

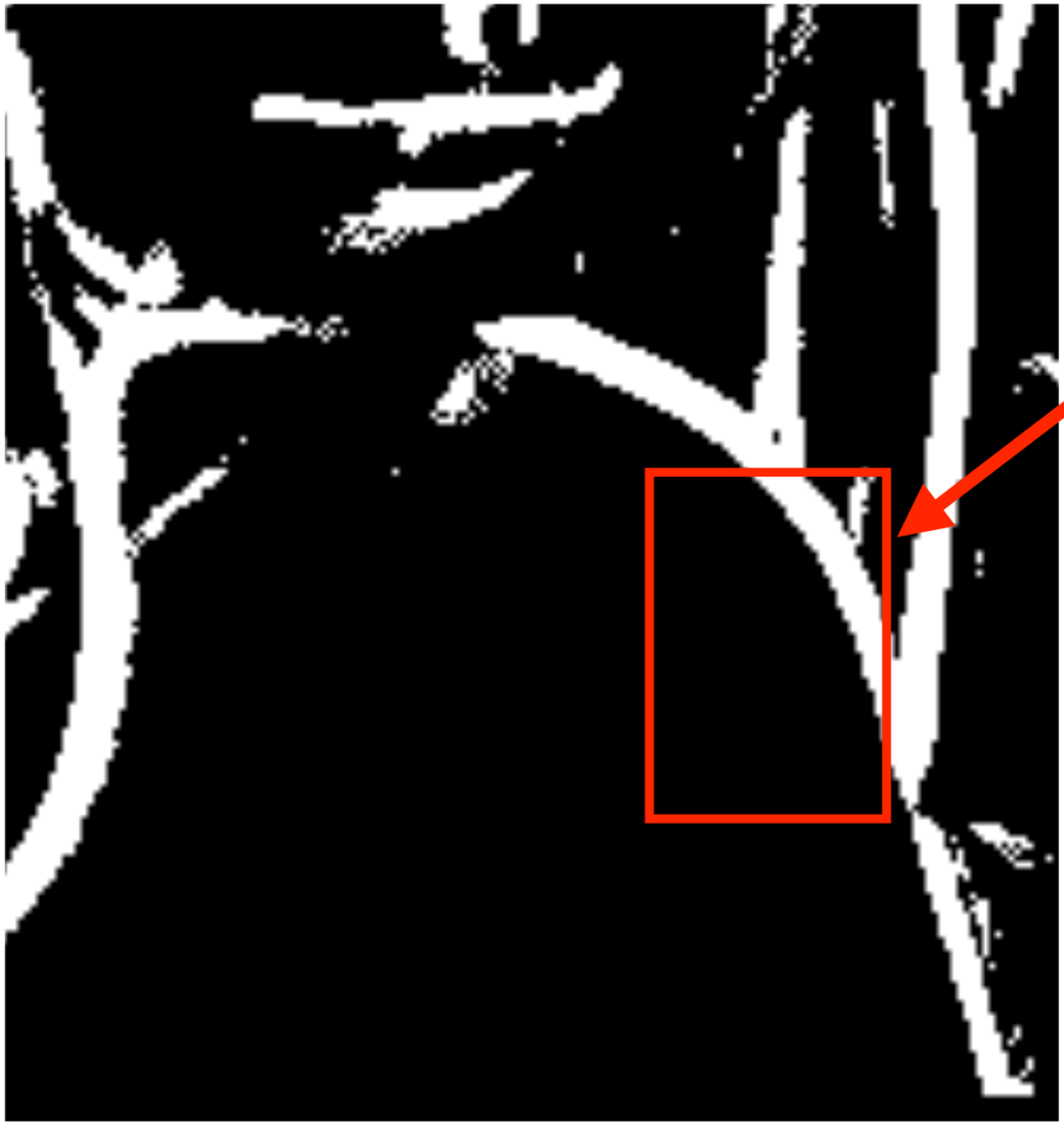
The Canny edge detector

thresholding

The Canny edge detector



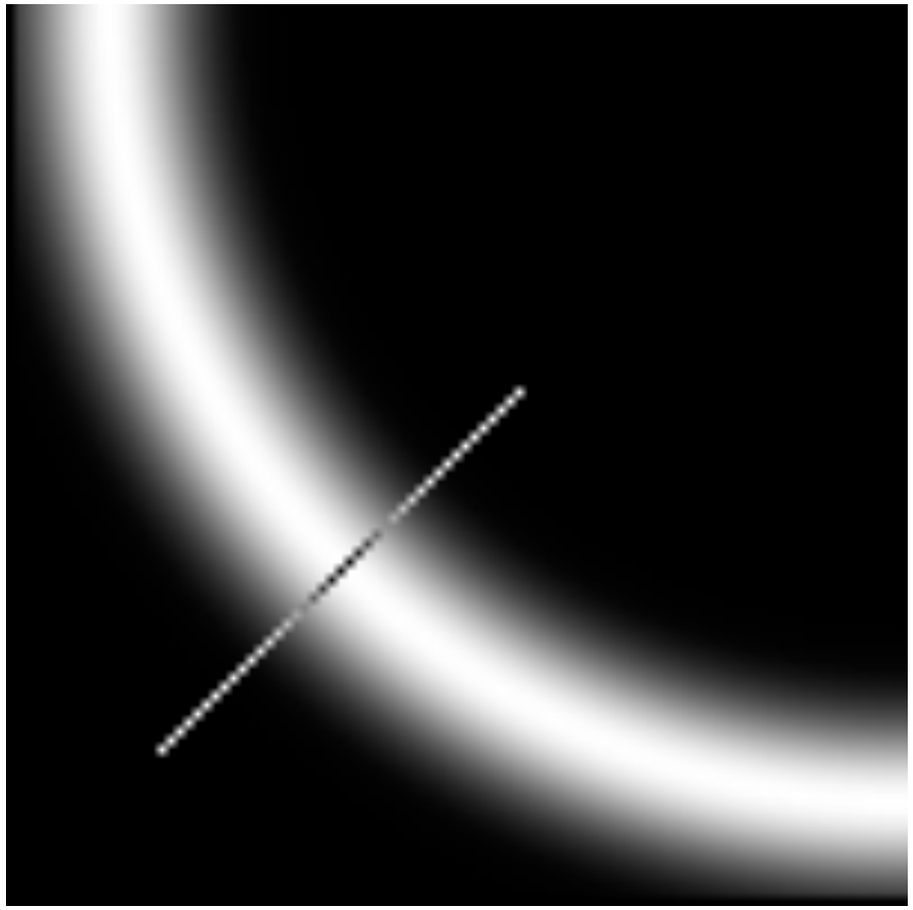


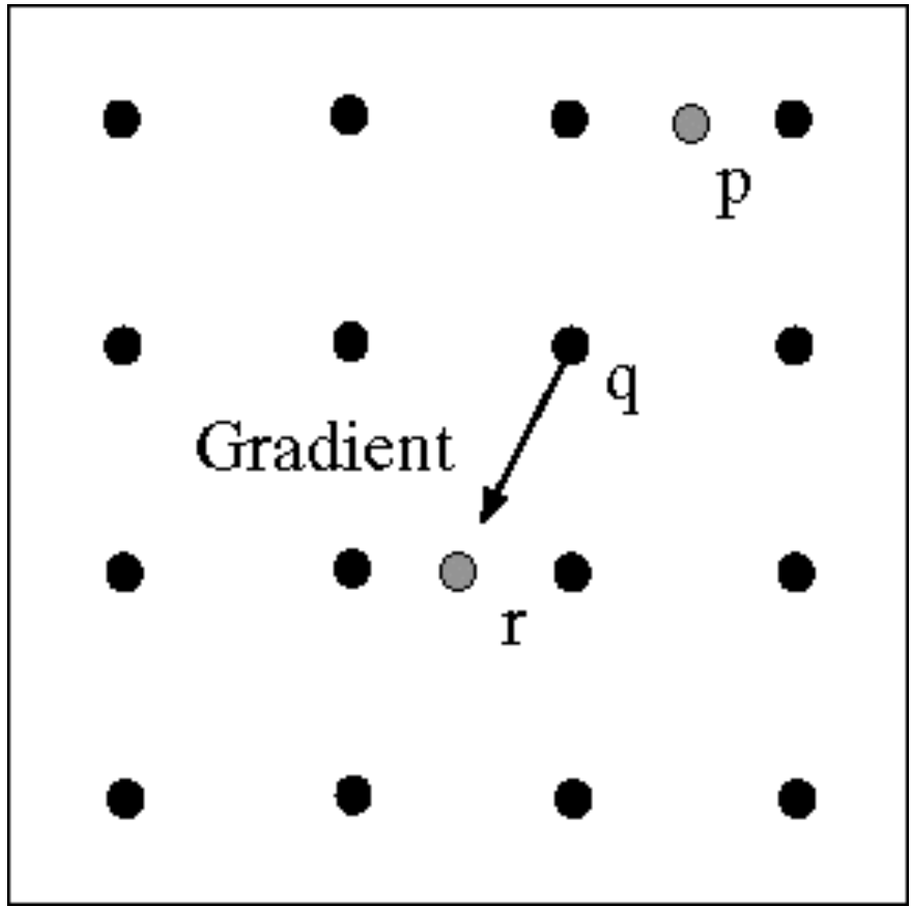


How to turn these thick regions of the gradient into curves?

thresholding

Non-maximum suppression



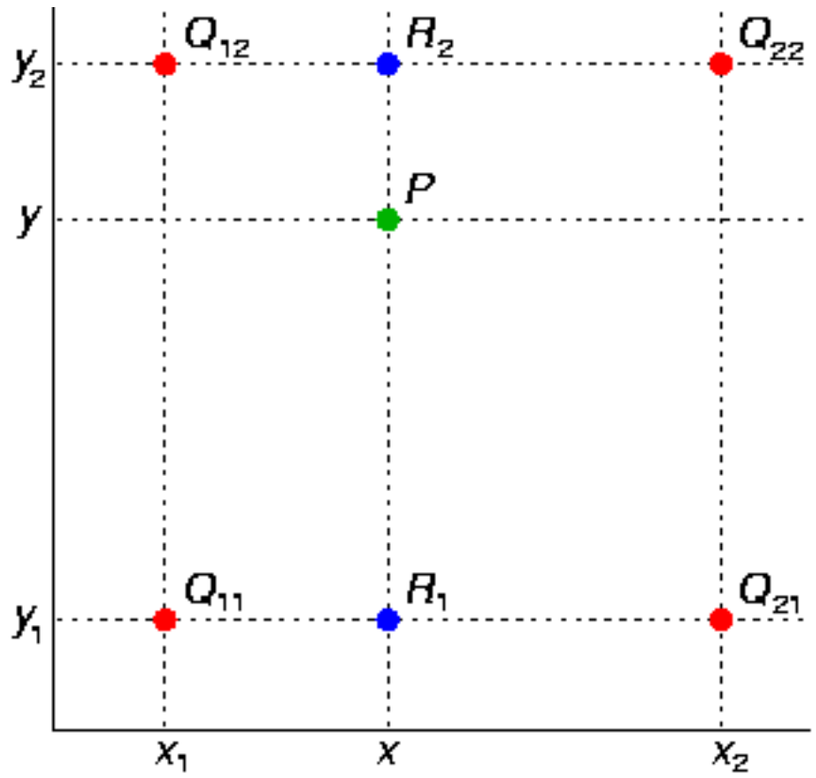


Check if pixel is local maximum along gradient direction, select single max across width of the edge

- requires checking interpolated pixels p and r

Bilinear interpolation

http://en.wikipedia.org/wiki/Bilinear_interpolation



$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

$$f(x, y) \approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2)$$

The Canny edge detector

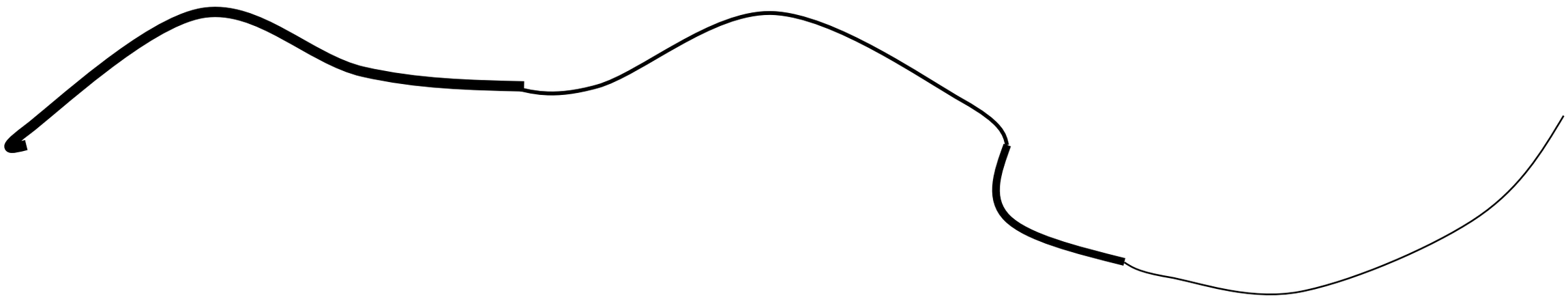
Problem: pixels along this edge didn't survive the thresholding

thinning

(non-maximum suppression)

Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
  - –drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



Source: S. Seitz

Hysteresis thresholding

original image

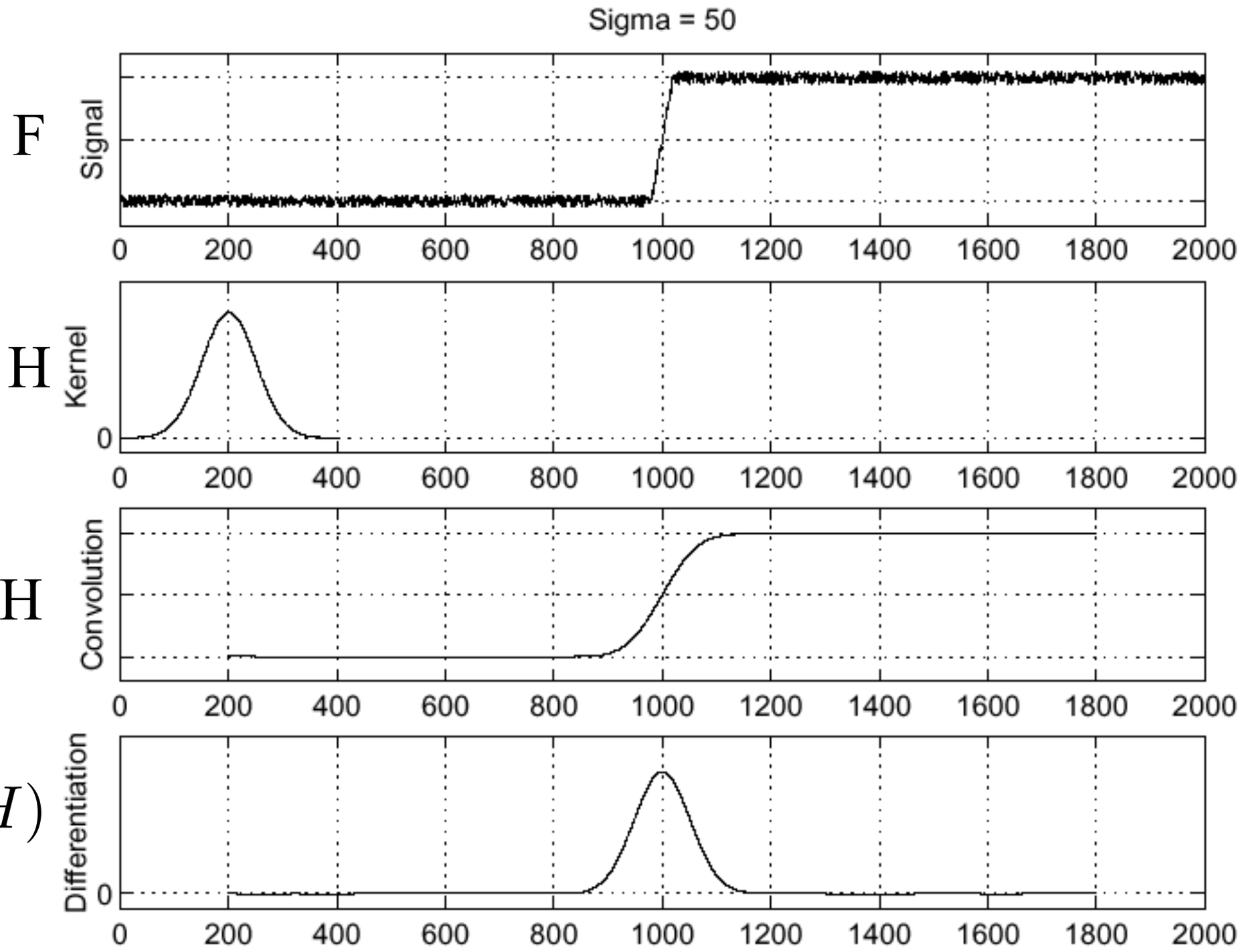
high threshold
(strong edges)

low threshold
(weak edges)

hysteresis threshold

Source: L. Fei-Fei

Alternate approach



Find peaks of above profile be setting *its* gradient equal to 0.

What does the resulting edge filter look like?

Boring math

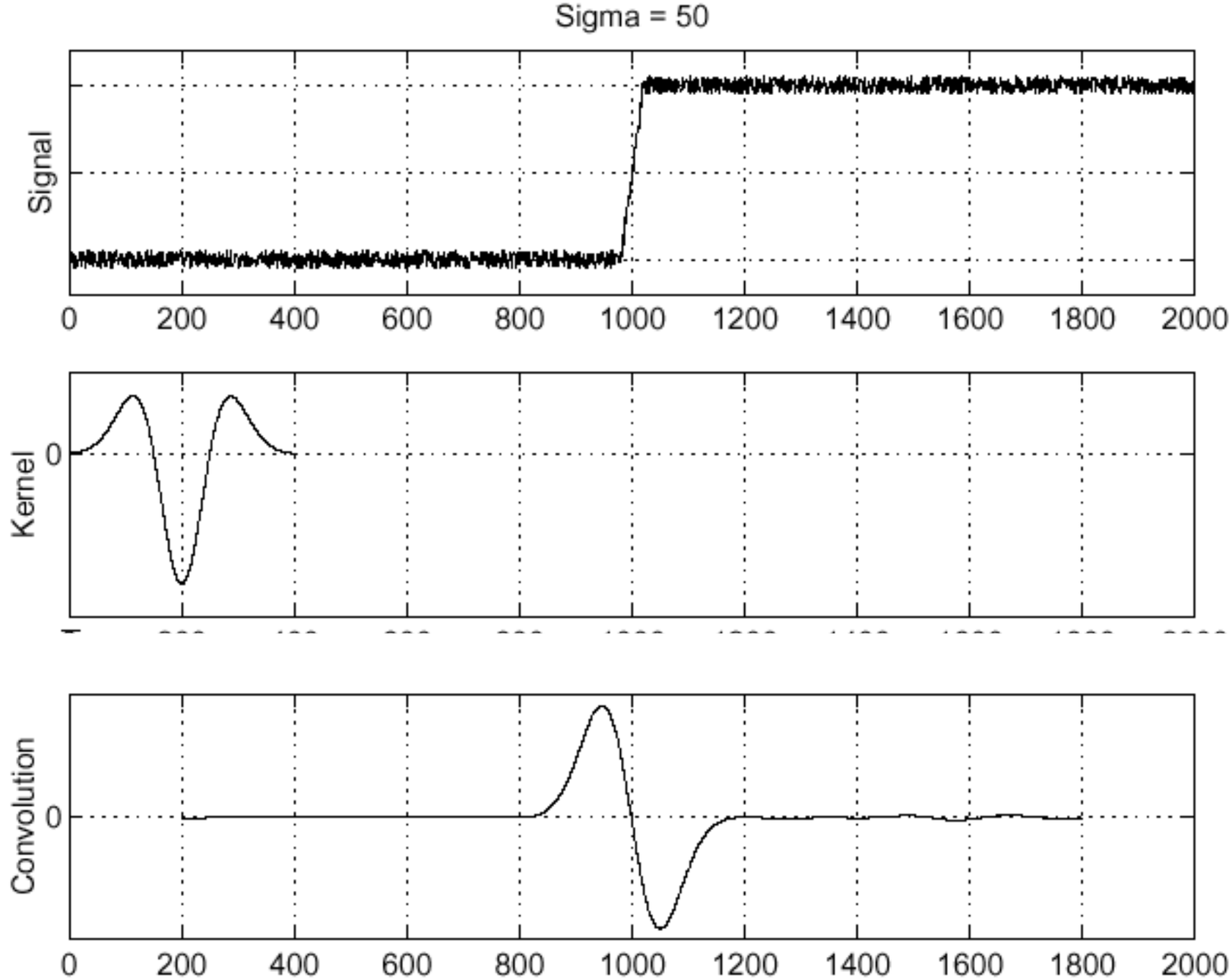
F*H*H where H = [1 -1]
Convolution with filter G = [1 -1]*[1 -1]

$$G[0] = 0 * -1 + 1 * 1 + -1 * 0 = 1$$
$$G[1] = -1 - 1 = -2$$
$$G[2] = 1 * 0 + -1 * -1 + 0 * 1 = 1$$

G = [1 -2 1]

Look for zero-crossings of second derivative

Consider smoothing (Gaussian) + second derivative [1 -2 1]

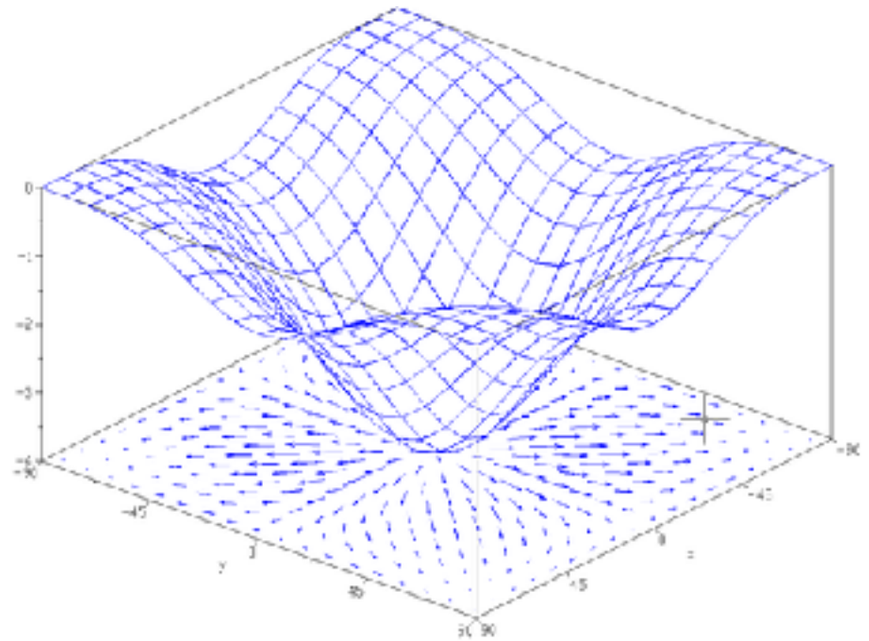


Zero-crossings of second graph

Generalization to 2D

∇^2 is the **Laplacian** operator: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Divergence ("source-ness" or "sink"-ness) of a gradient of a function (used in fluid mechanics)

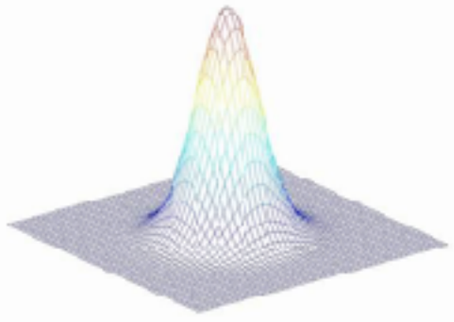


$$I_{xx} + I_{yy} = \left(\begin{bmatrix} 1 & -2 & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \right) * I = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} * I$$

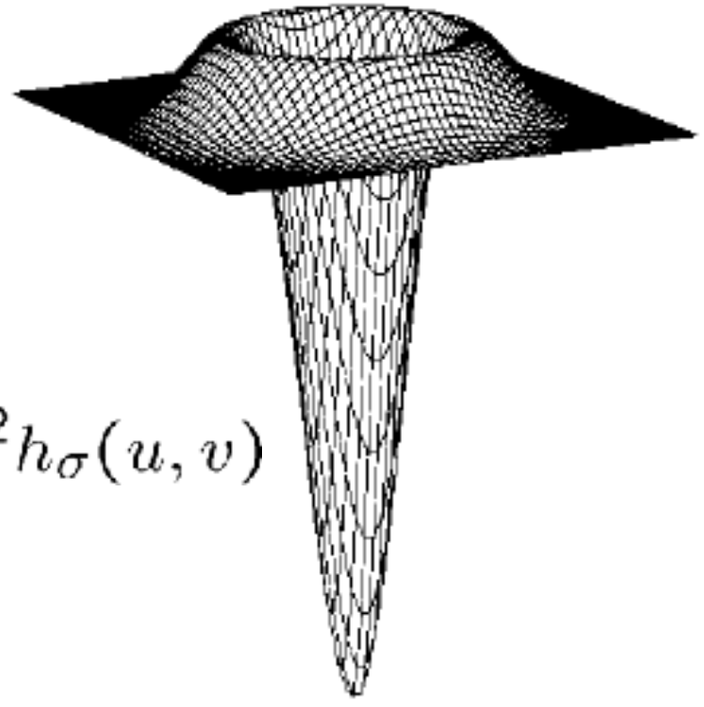
Laplacian of Gaussian (LOG)

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

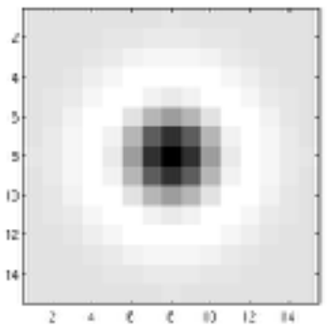
Let $h_\sigma(u,v)$ be a 2D Gaussian



Laplacian of Gaussian



$\nabla^2 h_\sigma(u,v)$

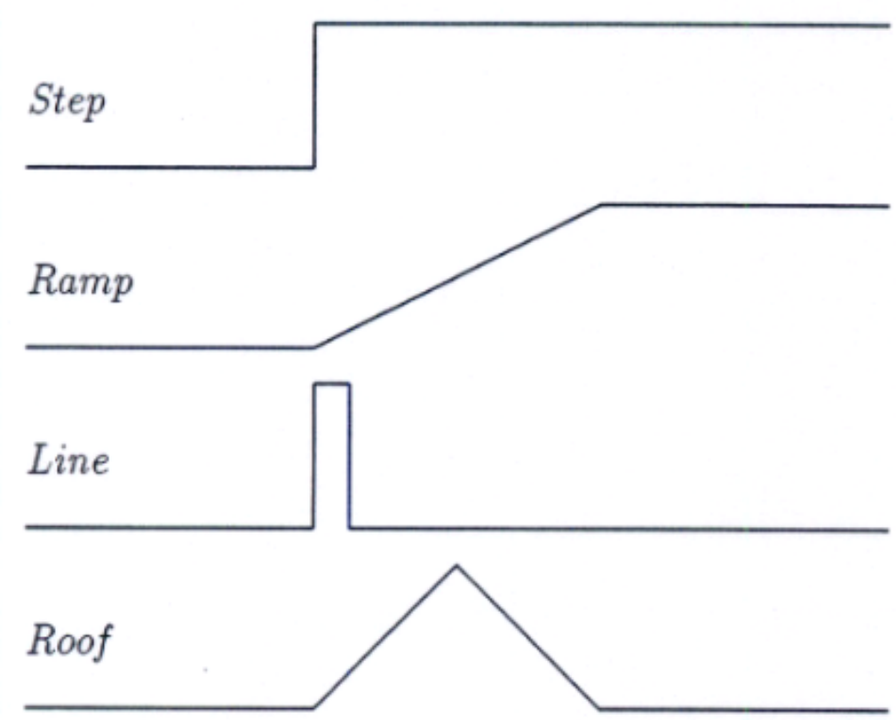


Marr & Hildreth

Outline

- Logistics
- Edges
- Filter banks
- Efficiency (pyramids, separability, steerability)

Other local features



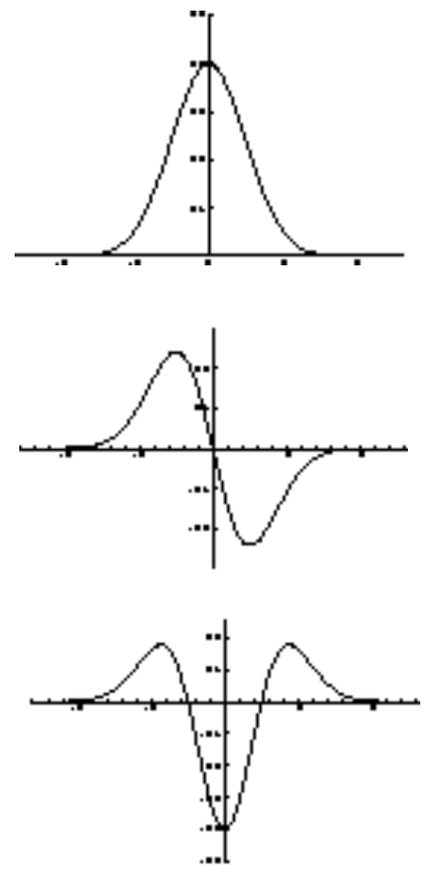
Other filters: Gaussian derivatives

$$G_0(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(\frac{-z^2}{2}) \quad ; \quad z = \frac{x}{\sigma}$$

$$G_1(x) = -\frac{1}{\sigma} z G_0$$

$$G_2(x) = \frac{1}{\sigma^2}\left(z^2 - 1\right) G_0$$

$$G_3(x) = -\frac{1}{\sigma^3}\left(z^3 - 3z\right) G_0$$



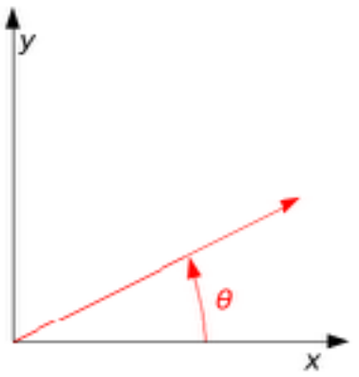
$$G_1(x) = \frac{1}{\sigma\sqrt{2\pi}} \frac{-z}{\sigma} e^{-\frac{z^2}{2}}$$

2D Gaussian derivates

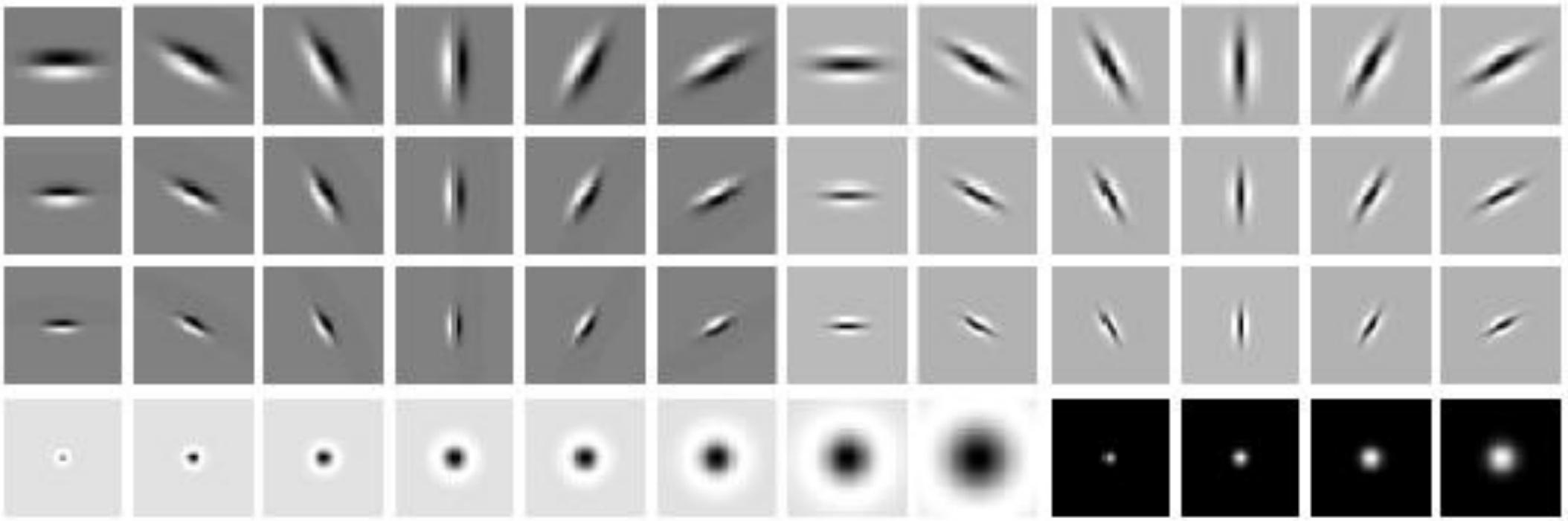
$$H_{ij}(x,y) = G_i(x)G_j(y)$$
$$H_{i,j,a,b}(x,y) = H_{ij}(x',y')$$
$$H_{\theta,i,j,a,b}(x,y) = H_{ij}(x'',y'')$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Filter banks

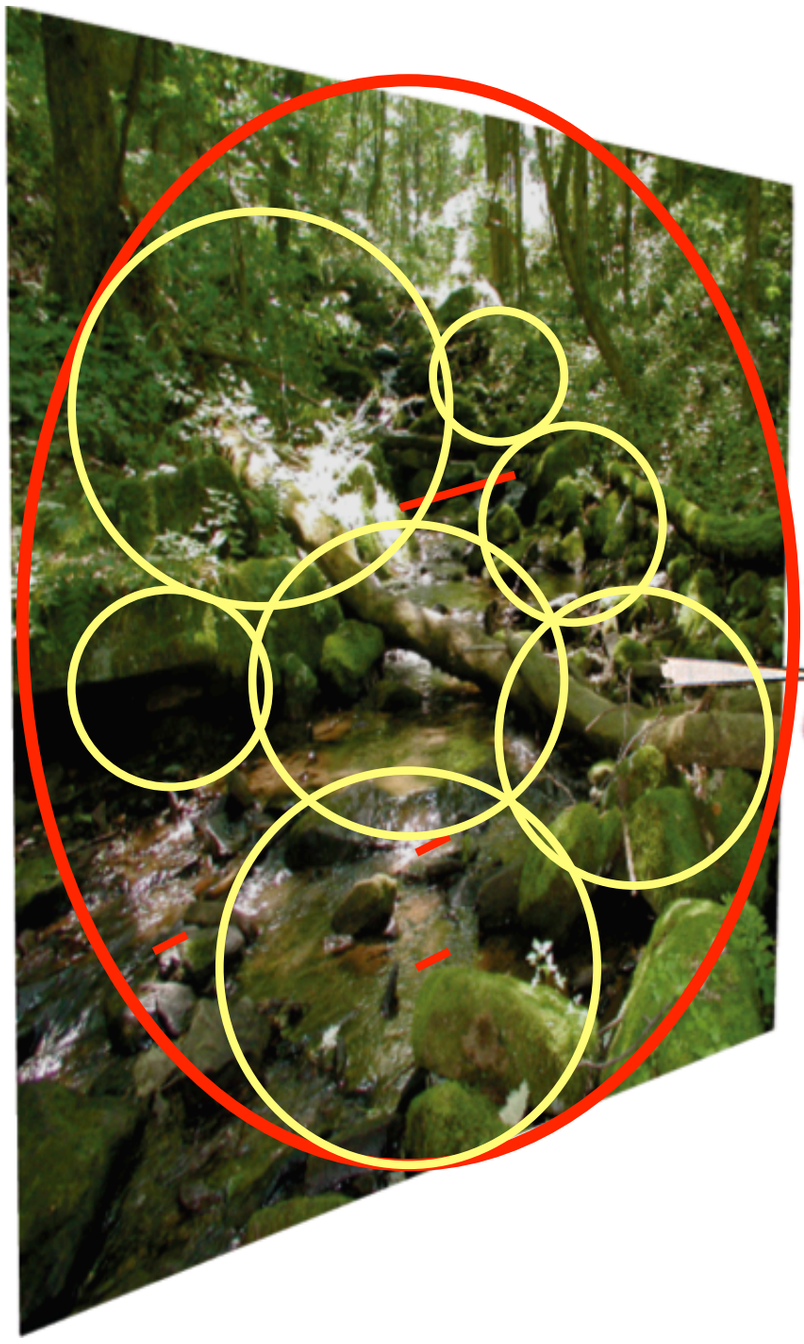


The LeungMalik filter bank has a mix of edge, bar and spot filters at multiple scales and orientations. It has a total of 48 filters - 2 Gaussian derivative filters at 6 orientations and 3 scales, 8 Laplacian of Gaussian filters and 4 Gaussian filters.

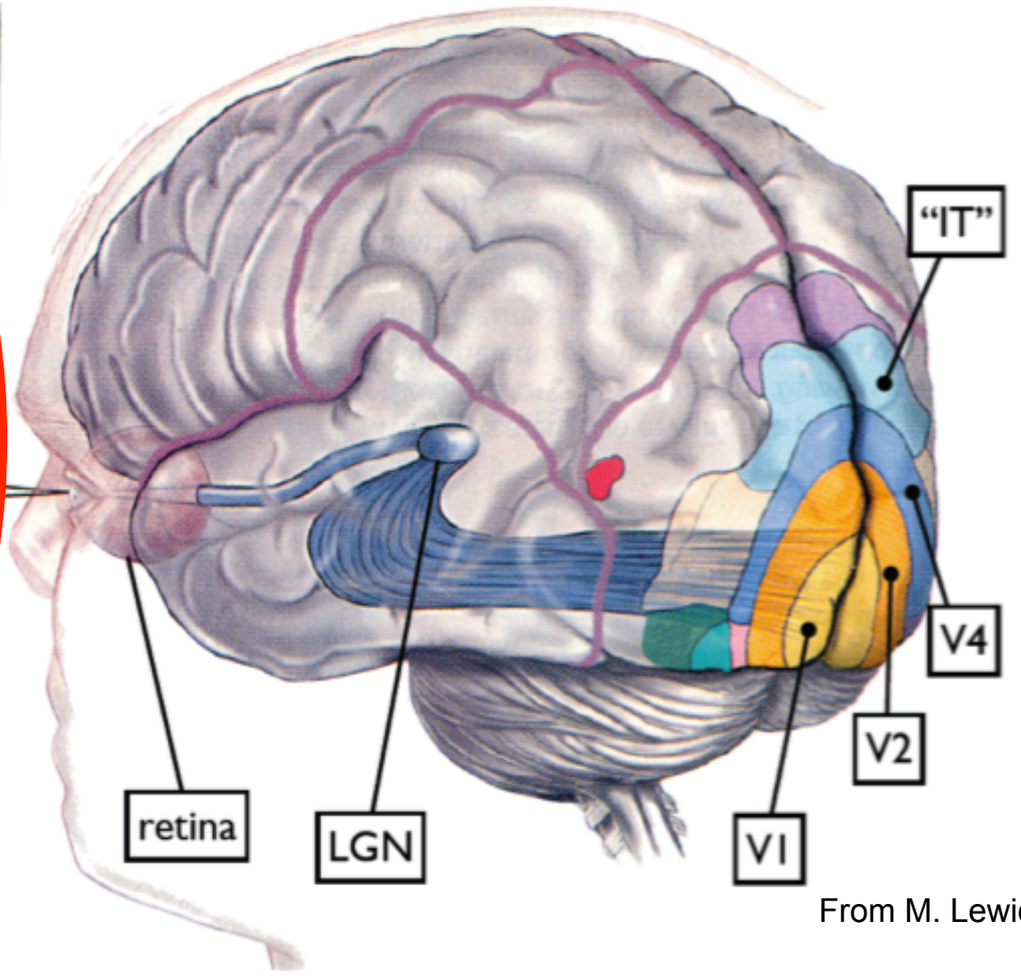
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

$$= H^T F_{ij} = ||H||||F_{ij}|| \cos\theta, \quad H, F_{ij} \in R^{(2K+1)^2}$$

Biological motivation



Some visual areas…



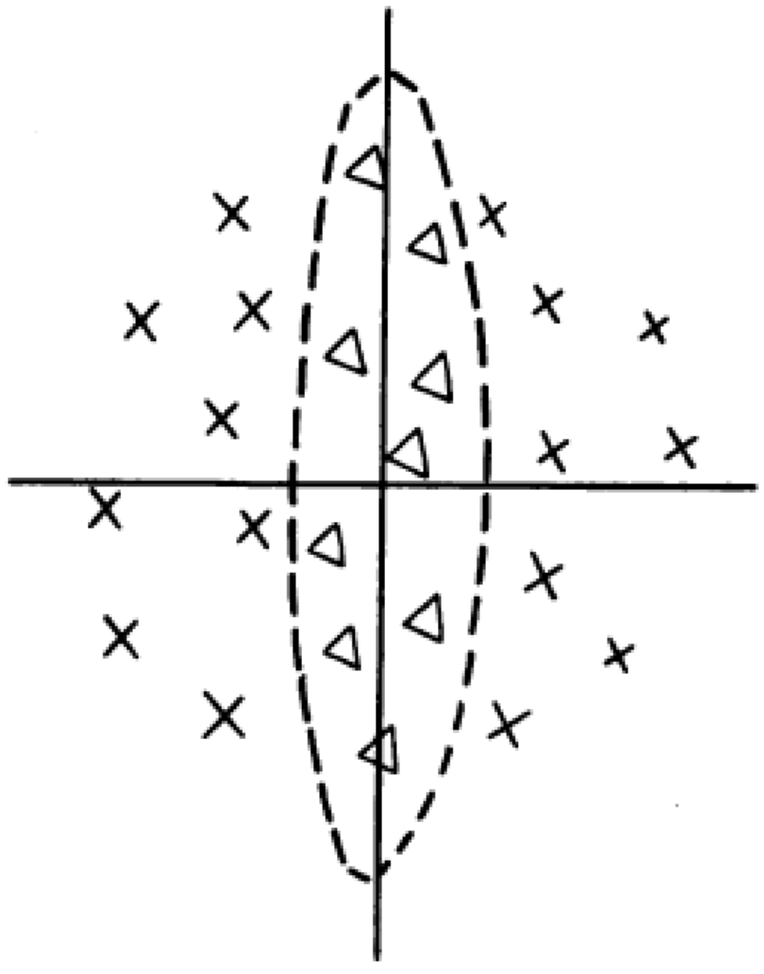
From M. Lewicky

J. Physiol. (1959) 148, 574–591

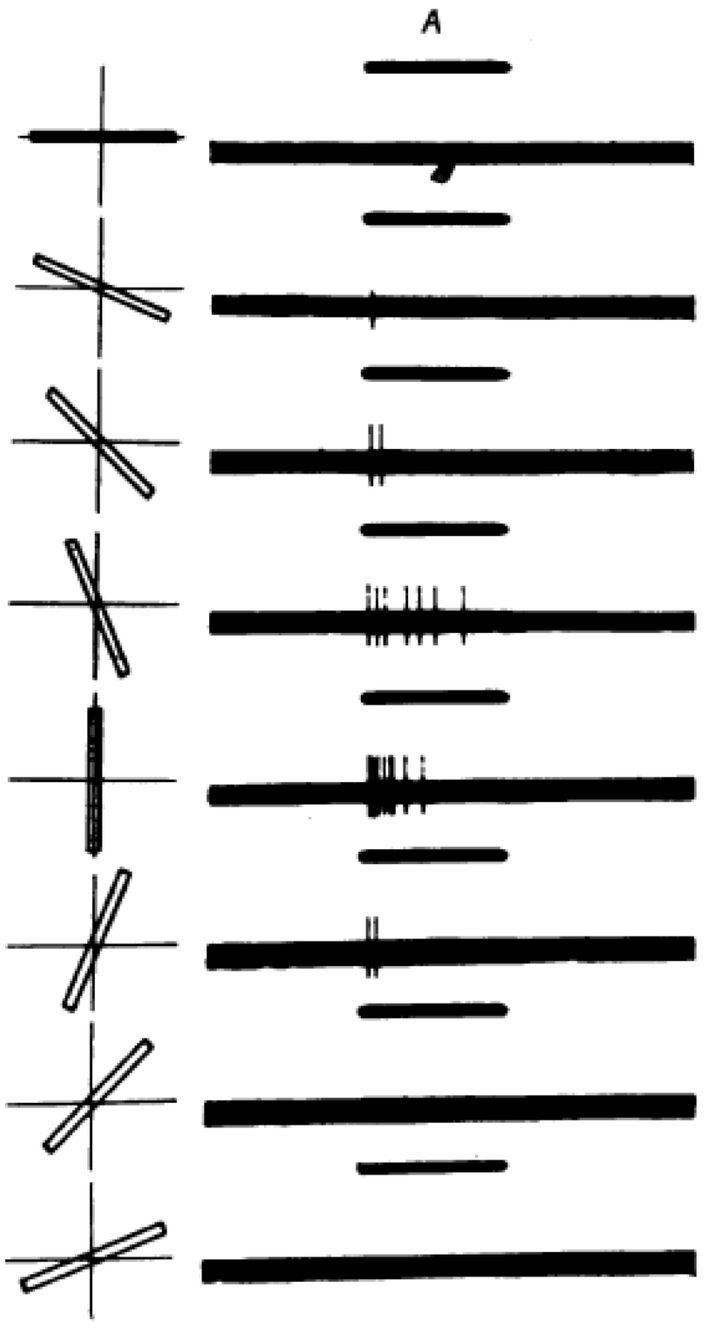
RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

By D. H. HUBEL* and T. N. WIESEL*

From the Wilmer Institute, The Johns Hopkins Hospital and University, Baltimore, Maryland, U.S.A.



Receptive field
of a cell in the cat's cortex

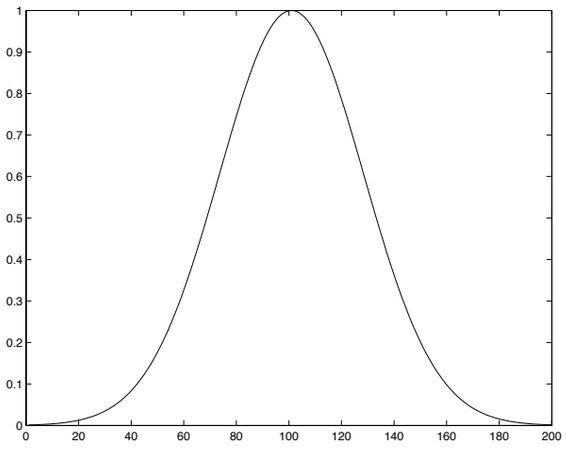


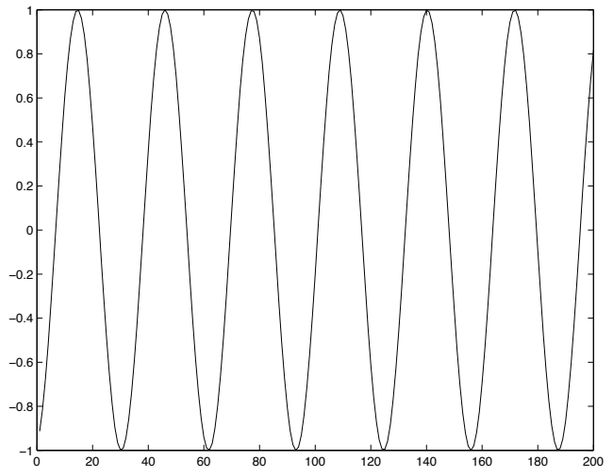
Responses to an oriented bar

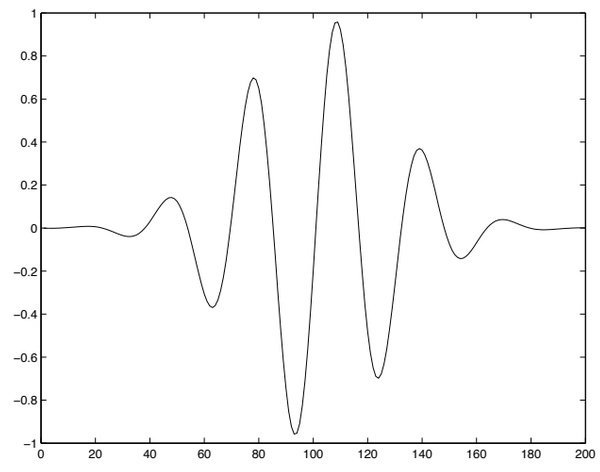
Other filters: gabors

$$h(x) = e^{-\frac{x^2}{2\sigma^2}} w(x)$$

$$w(x) = \cos(2\pi f x)$$



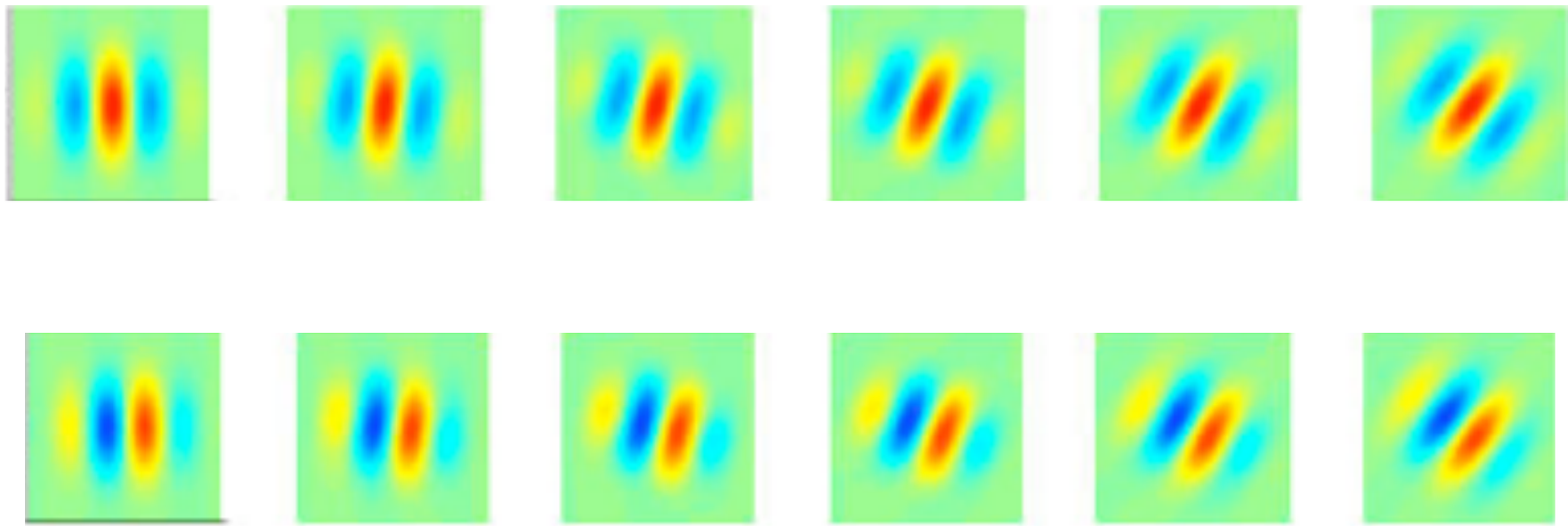




f = frequency (# cycles in a “second”)

$$\lambda = \frac{1}{f} \text{ wavelength}$$

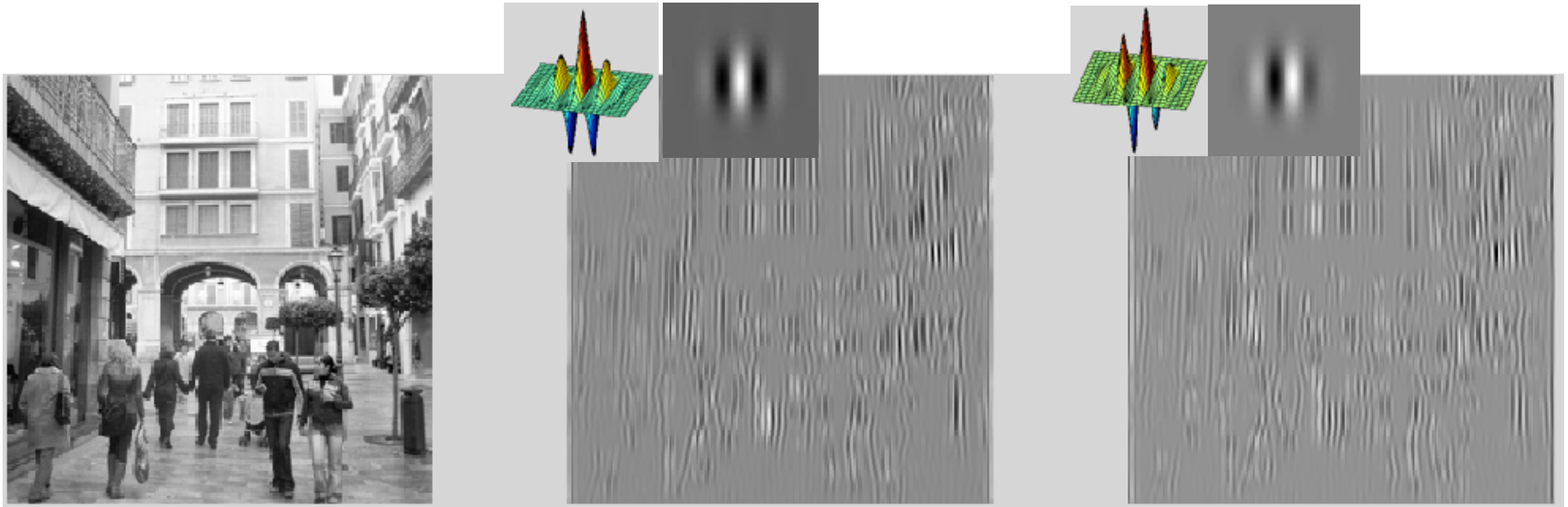
Gabor filters



It turns out, we can write cosine + sine modulated gabor filters as real and imaginary parts of a single complex filter (Fourier theory)

http://en.wikipedia.org/wiki/Gabor_filter

Gabor energy



Given an image of a grating with a particular stripe width (frequency), even and odd filters will be sensitive to precise alignment (phase)

$$Energy = (I * G_{even})^2 + (I * G_{odd})^2$$

(Magnitude of complex filter)

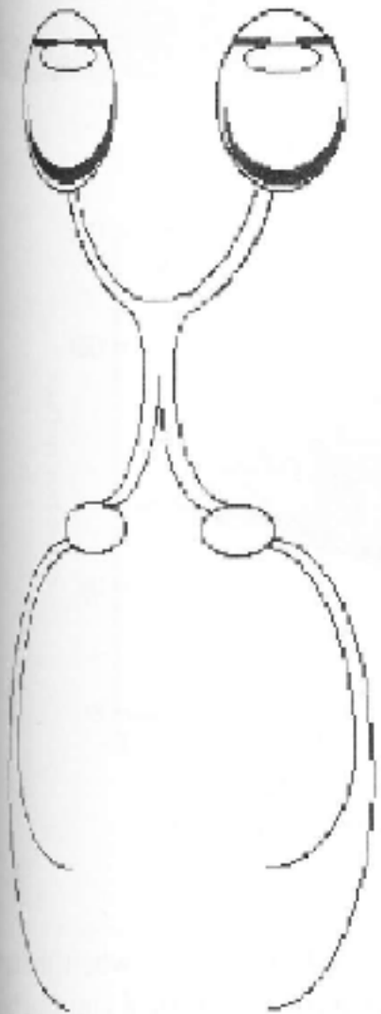
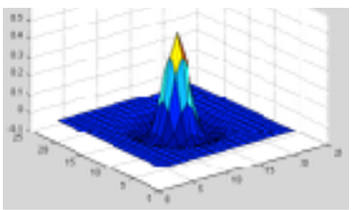
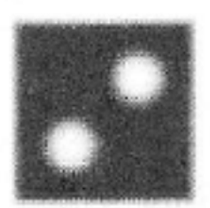
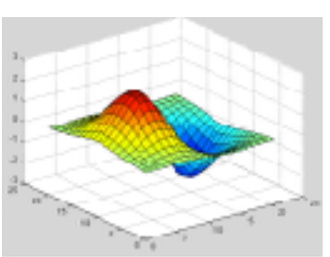
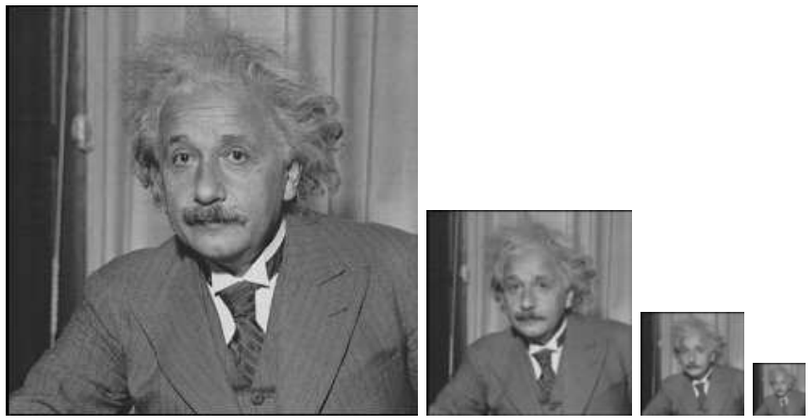
Structure		Operations	2D Fourier Plane
	World	I (x,y,t,λ)	
	Optics	Low-pass spatial filtering	
	Photoreceptor Array	Sampling, more low-pass filtering, temporal low/bandpass filtering, λ filtering, gain control, response compression	
	LGN Cells	Spatiotemporal bandpass filtering, λ filtering, multiple parallel representations	 
	Primary Visual Cortical Neurons: Simple & Complex	Simple cells: orientation, phase, motion, binocular disparity, & λ filtering Complex cells: no phase filtering (contrast energy detection)	 

FIGURE 1 Schematic overview of the processing done by the early visual system. On the left, are some of the major structures to be discussed; in the middle, are some of the major operations done at the associated structure; in the right, are the 2-D Fourier representations of the world, retinal image, and sensitivities typical of a ganglion and cortical cell.

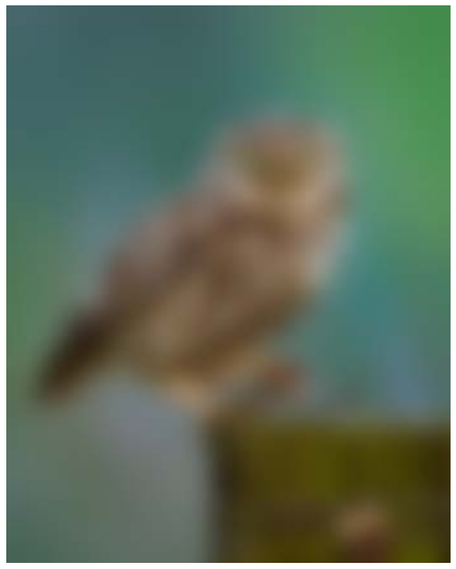
Outline

- Logistics
- Edges
- Filter banks
- Efficiency (pyramids, separability, steerability)

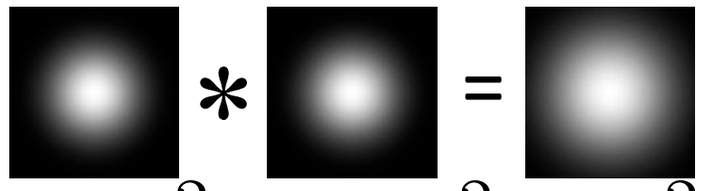
Pyramids



- Big filters (e.g., Gaussians) tend to be smooth, so the output is redundant



- Exploit property that Gaussian*Gaussian = Bigger Gaussian



$$\sigma_{a*b}^2 = \sigma_a^2 + \sigma_b^2$$

Proof: https://en.wikipedia.org/wiki/Sum_of_normally_distributed_random_variables

Burt & Adelson, 83

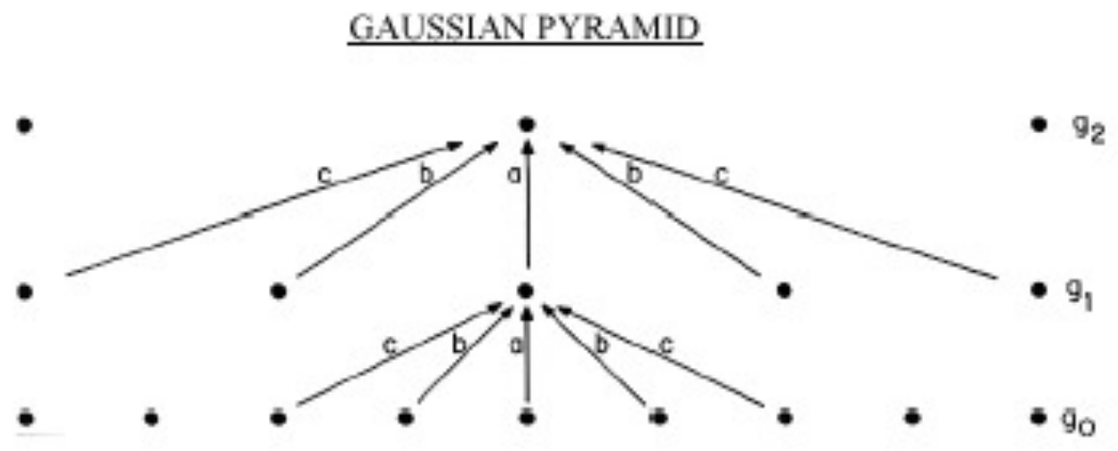


Fig 1. A one-dimensional graphic representation of the process which generates a Gaussian pyramid Each row of dots represents nodes within a level of the pyramid. The value of each node in the zero level is just the gray level of a corresponding image pixel. The value of each node in a high level is the weighted average of node values in the next lower level. Note that node spacing doubles from level to level, while the same weighting pattern or "generating kernel" is used to generate all levels.

$$H[i] = \frac{1}{16}\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix}$$

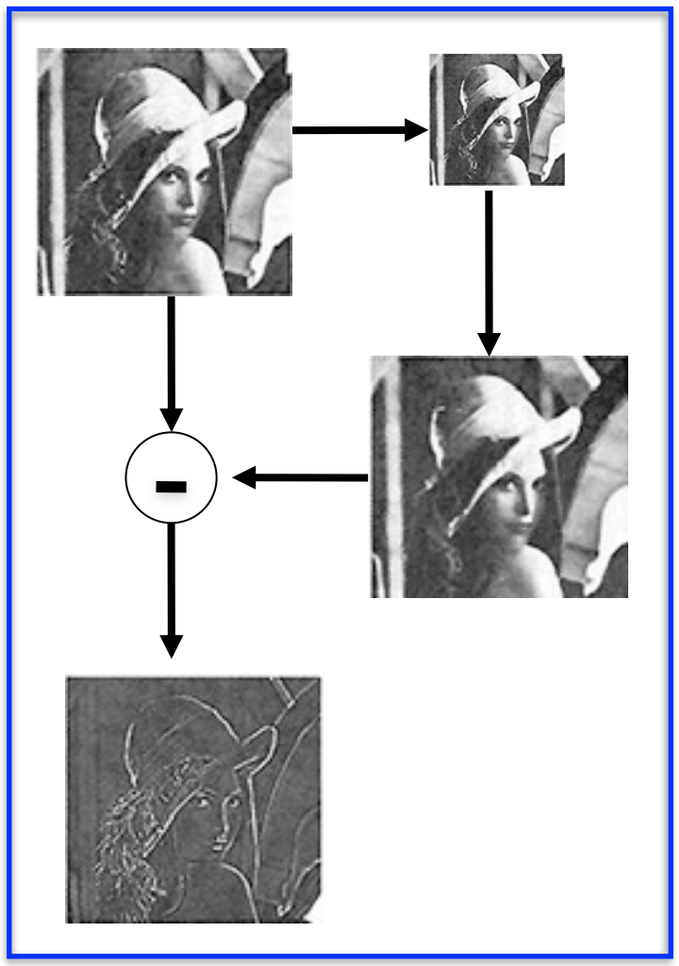
Fig. 4. First six levels of the Gaussian pyramid for the "Lady" image The original image, level 0, meusures 257 by 257 pixels and each higher level array is roughly half the dimensdons of its predecessor. Thus, level 5 measures just 9 by 9 pixels.

$$REDUCE(F)[i] = \sum_{u=-2}^{2} H[u]F[2i+u]$$

$$H[i] = \frac{1}{16}\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix}$$

Laplacian pyramid

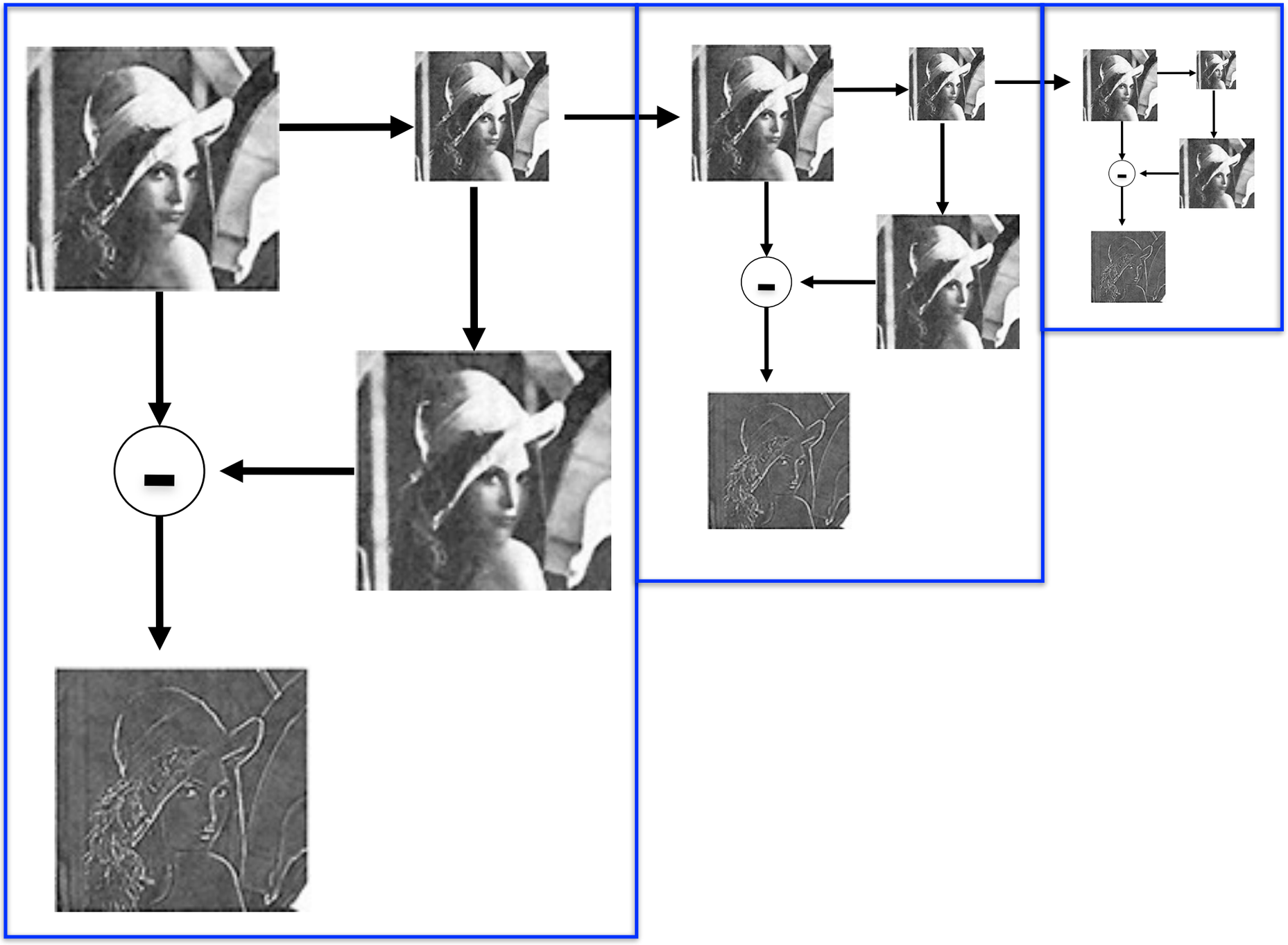
Store difference between upsampled Gaussian pyramid level and Gaussian pyramid level



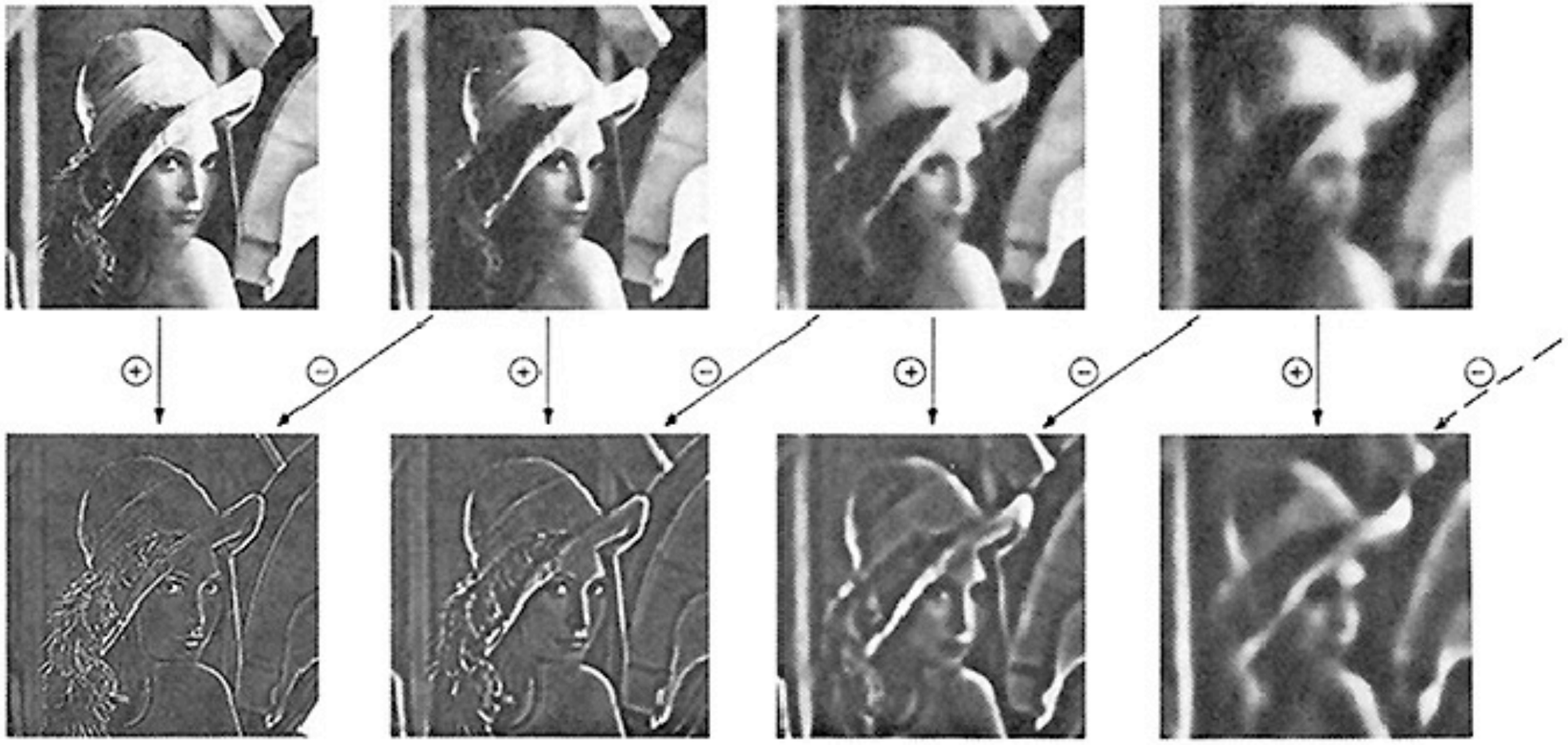
Upsampling: insert zeros between pixels and apply Gaussian filter

$$EXPAND(F)[i] = 4 \sum_{u=-2}^{2} H[u]F[(i+u)/2]$$

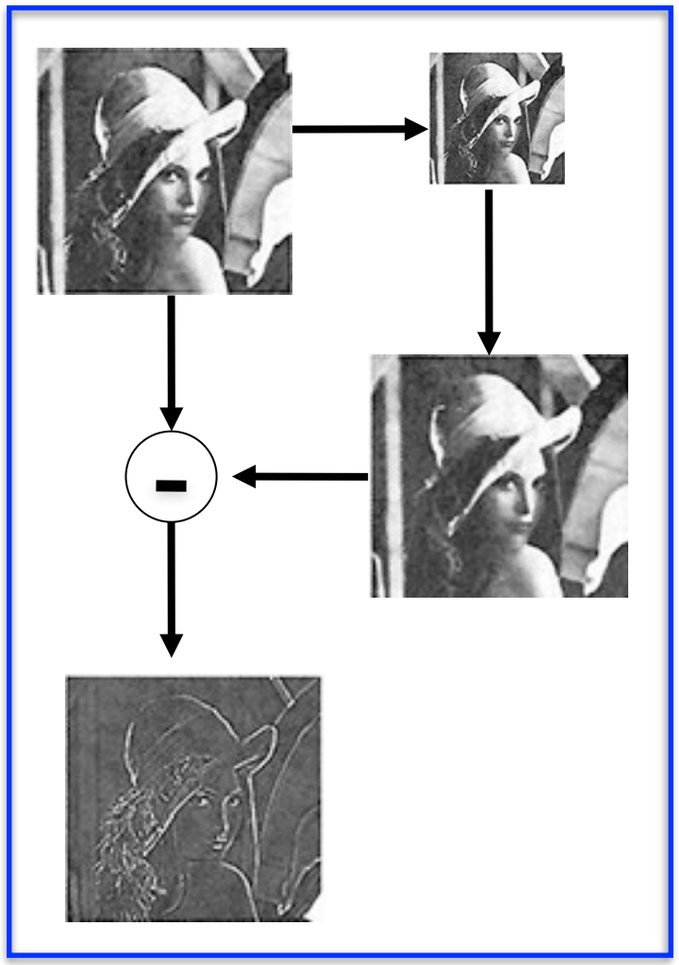
Laplacian pyramid



Laplacian pyramid

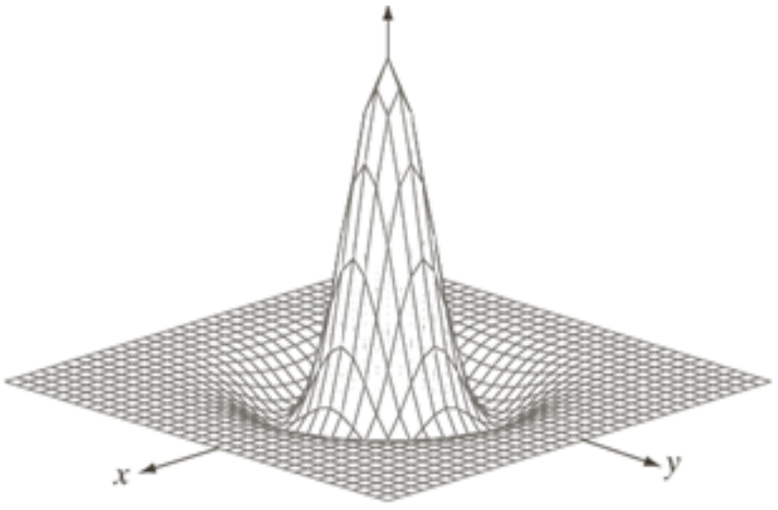


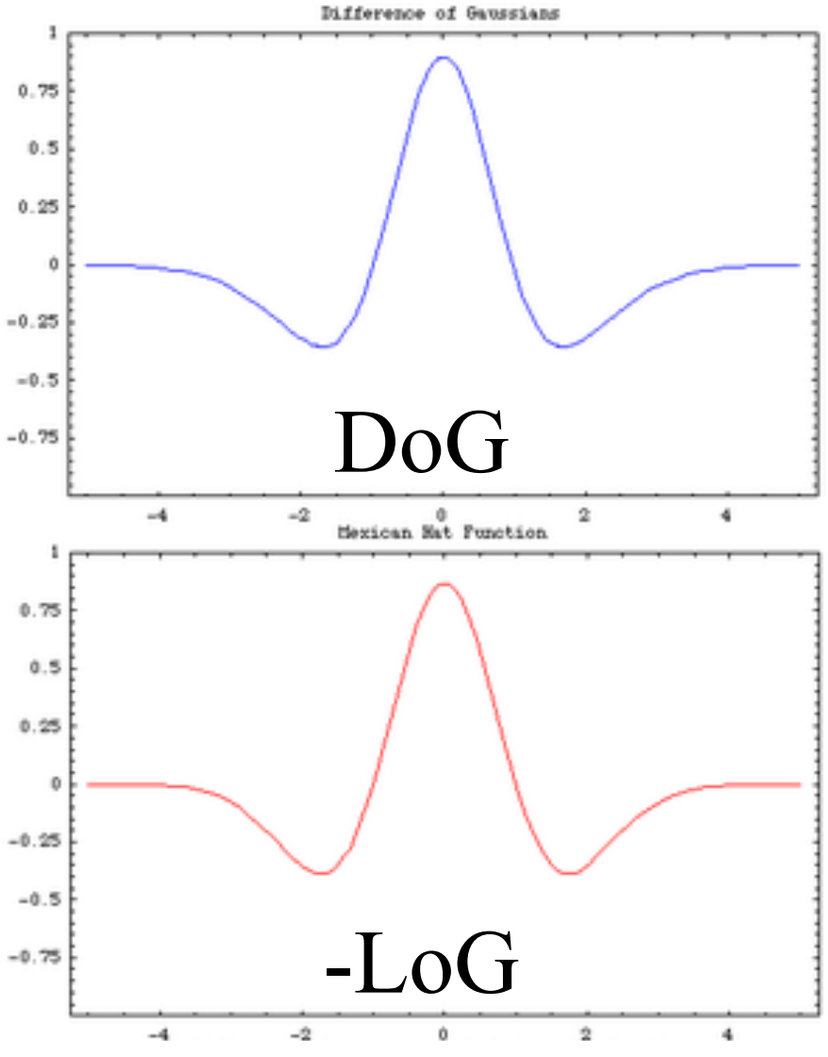
Laplacian pyramid



Can we directly produce the difference image with a linear filter?

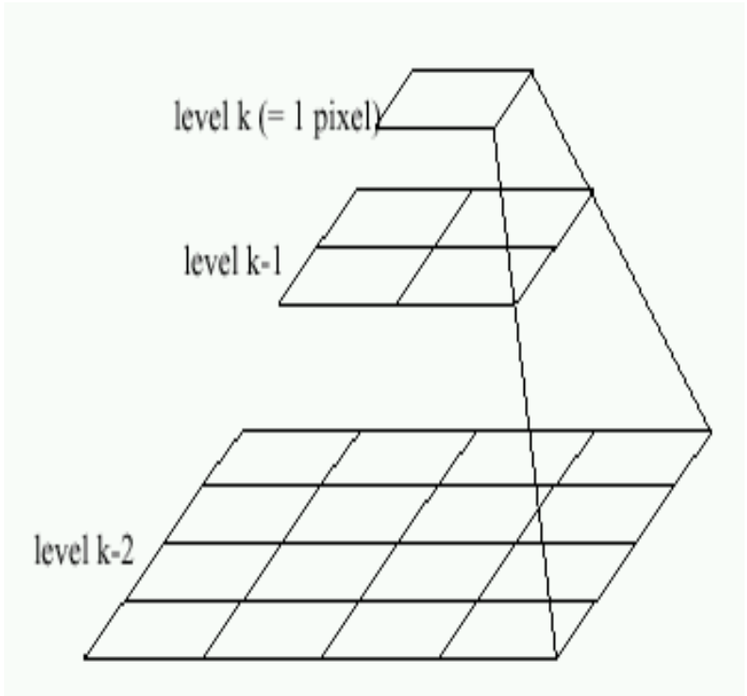
Difference of Gaussian (DoG) filter

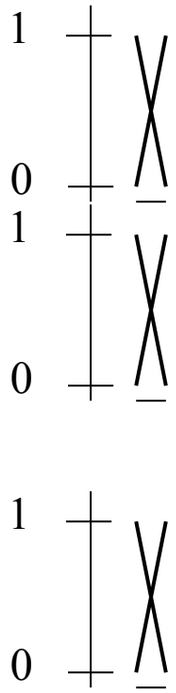


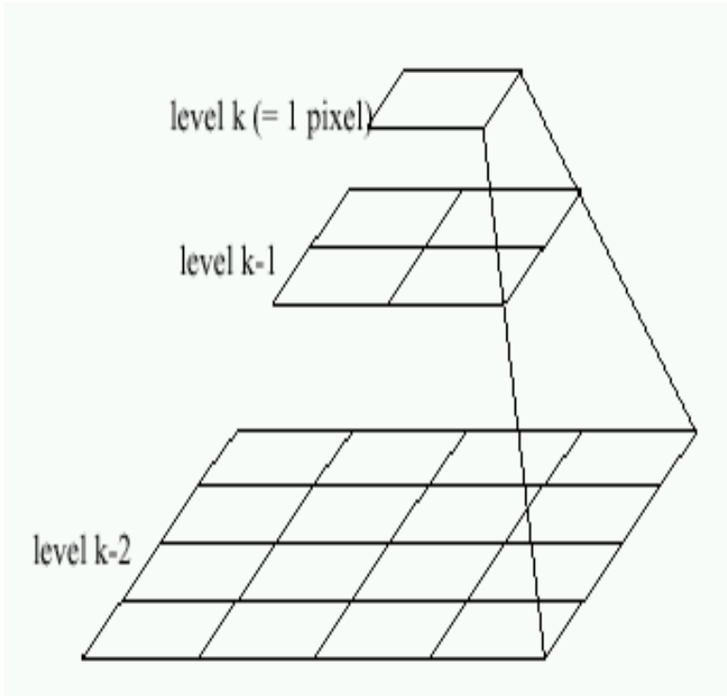


Multiresolution processing

Multi-resolution blending







Left pyramid

blend

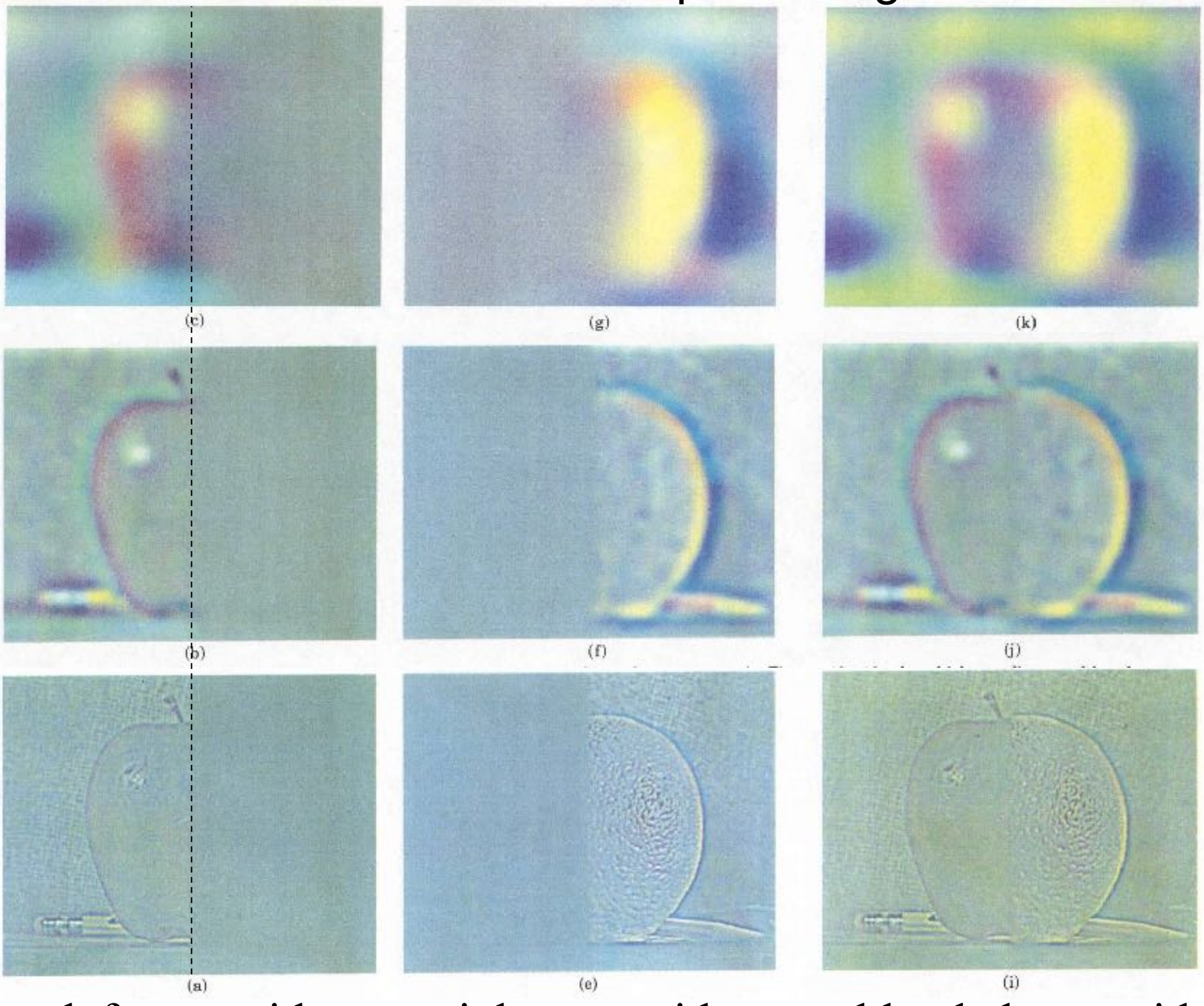
Right pyramid

Sum blended band-pass images

laplacian level 4

laplacian level 2

laplacian level 0

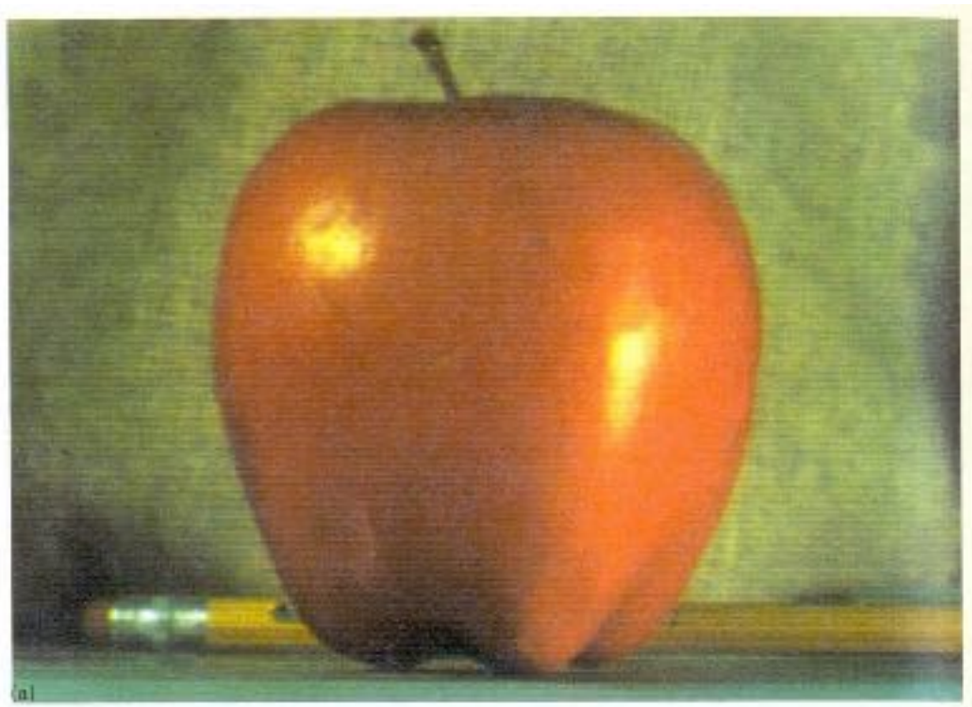


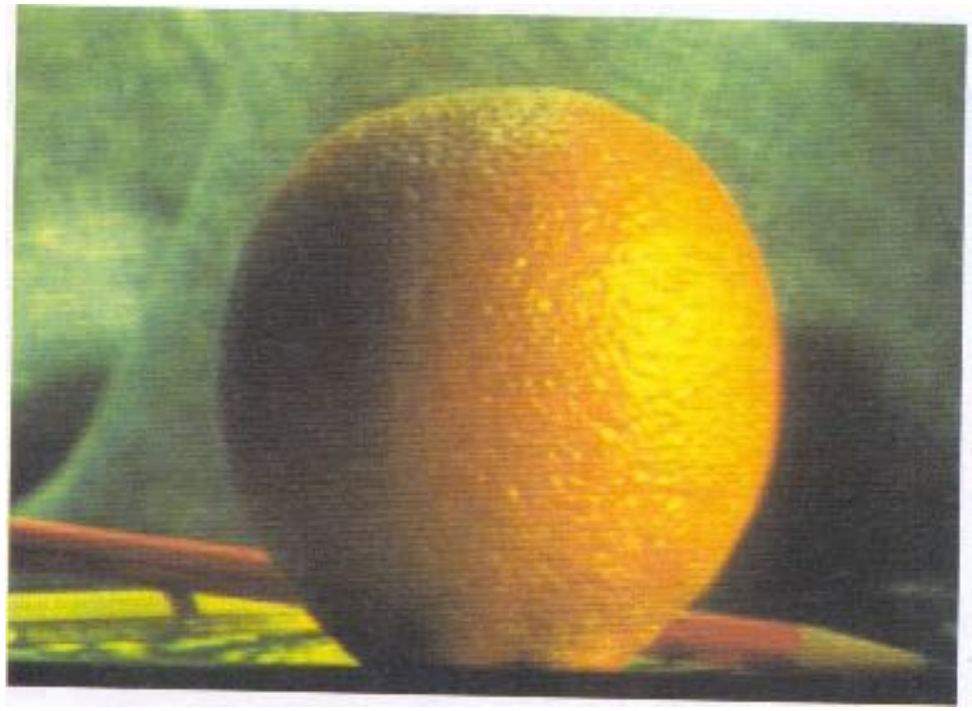
left pyramid

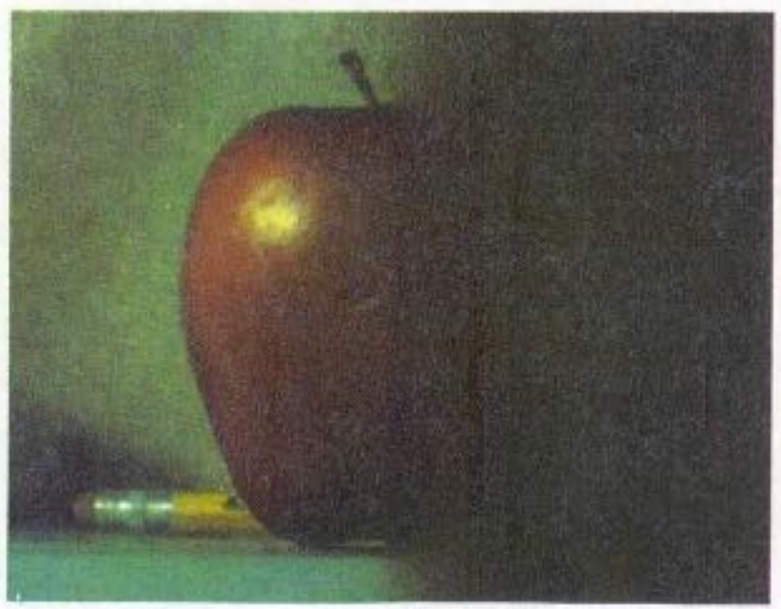
right pyramid

blended pyramid

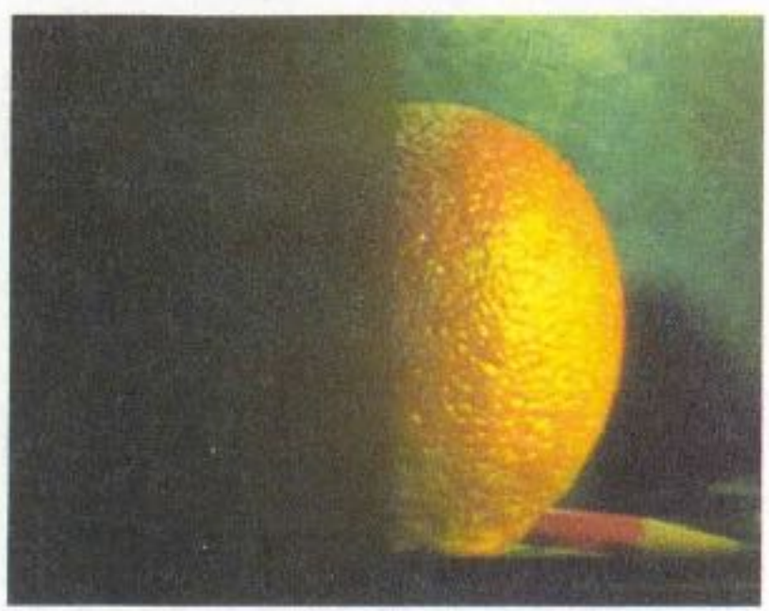
Pyramid Blending



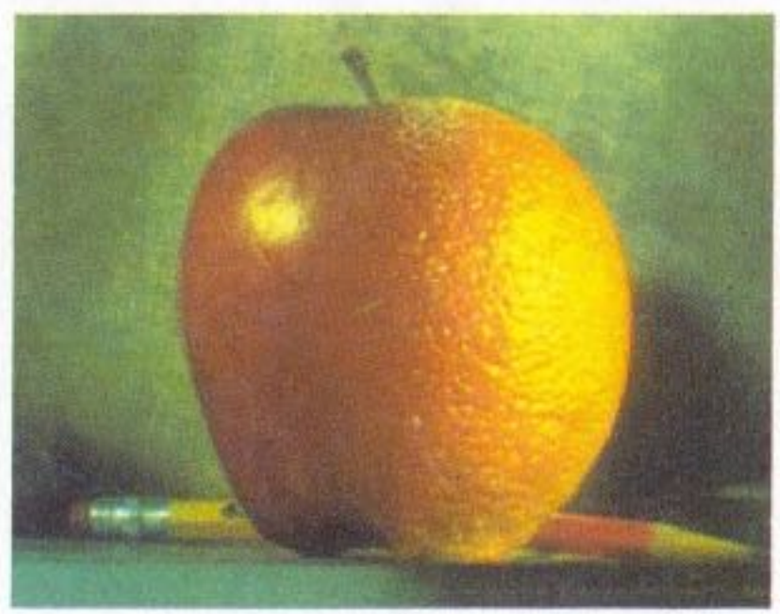




(d)

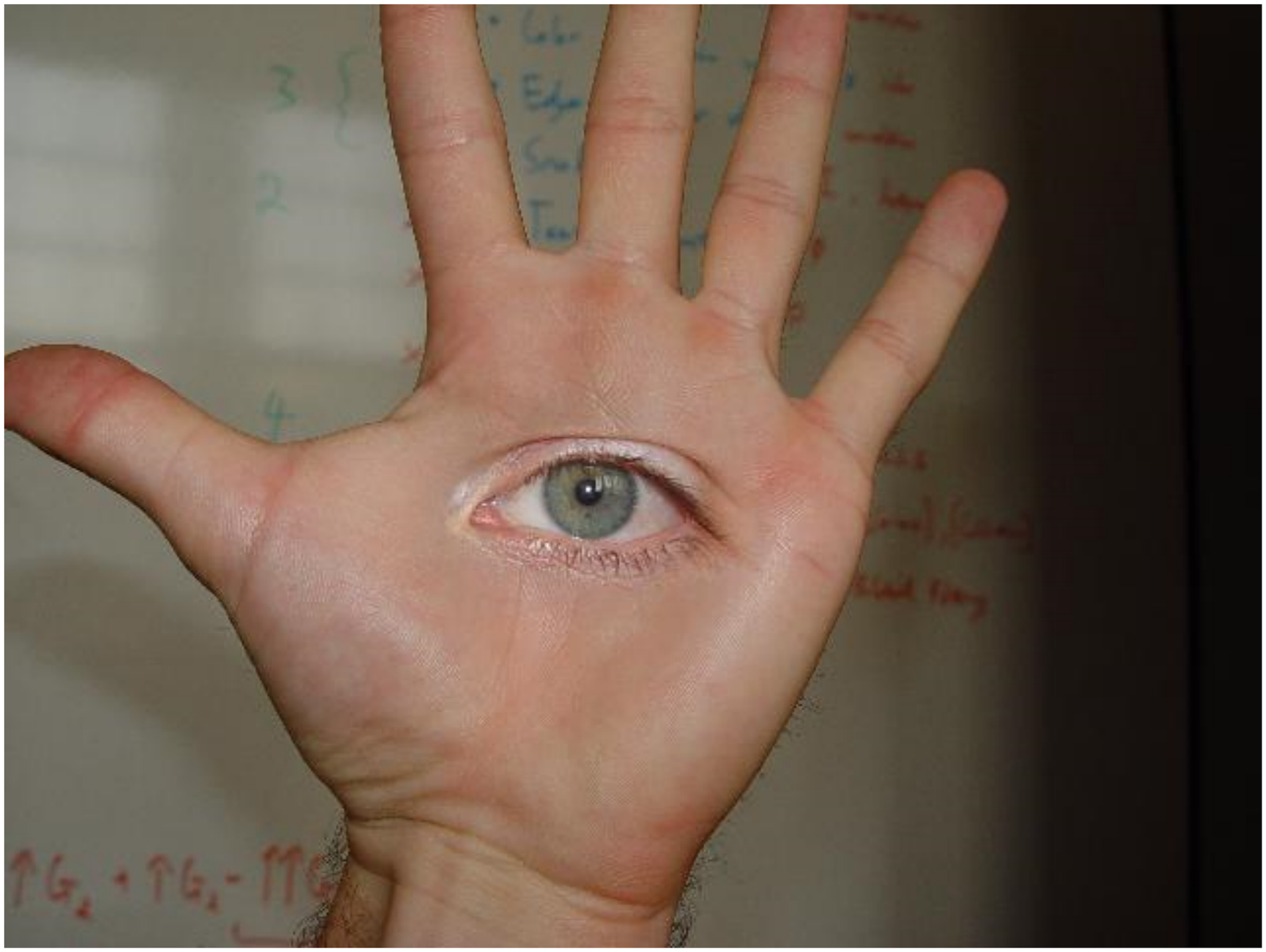


(h)



(l)

Horror Photo



© david dmartin (Boston College)

Outline

- Logistics
- Edges
- Filter banks
- Efficiency (pyramids, separability, steerability)

Separability

Image of size N^2
Filter of size M^2

Complexity of filtering?

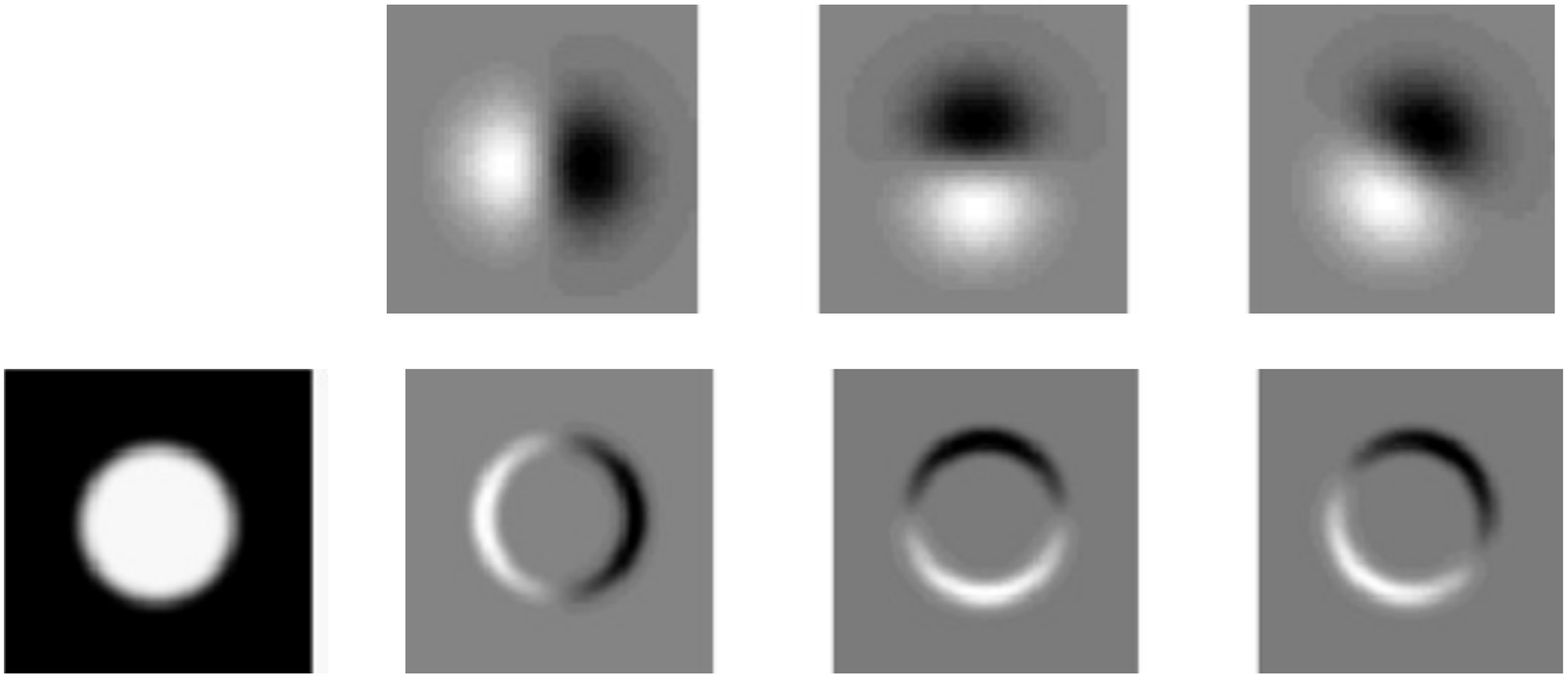
O(N^2M^2)

$$H[u,v] = H_x[u]H_y[v]$$

$$G[i,j] = \sum\sum H[u,v]F[i+u,j+v]$$

Steerability

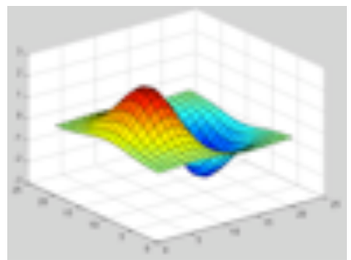
- Steerability - the ability to synthesize a filter of any orientation from a linear combination of filters at fixed orientaton



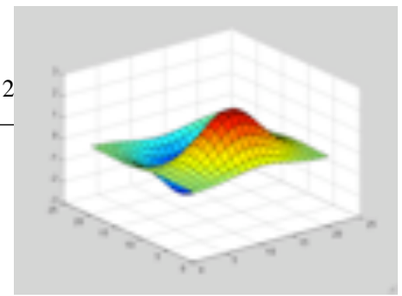
W. Freeman, T. Adelson, “The Design and Use of Sterrable Filters”, IEEE Trans. Patt, Anal. and Machine Intell., vol 13, #9, pp 891-900, Sept 1991

Derivatives of a Gaussian:

$$h_x(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$$h_y(x,y) = \frac{\partial h(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

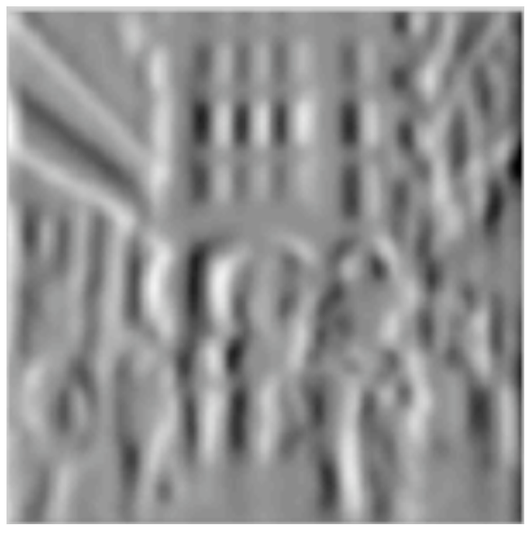


An arbitrary orientation can be computed as a linear combination of those two basis functions:

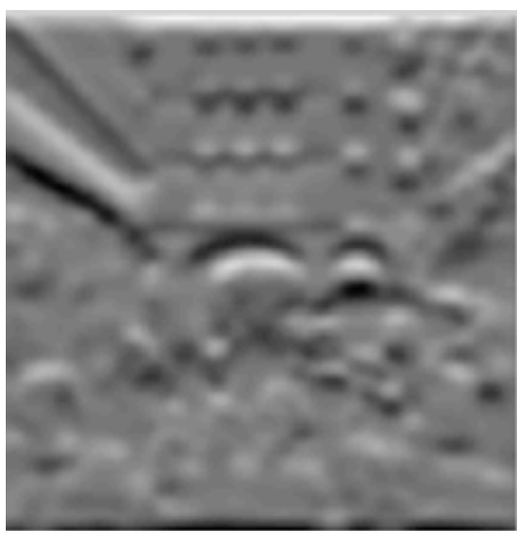
$$h_\alpha(x,y) = \cos(\alpha) h_x(x,y) + \sin(\alpha) h_y(x,y)$$

The representation is “shiftable” on orientation: We can interpolate any other orientation from a finite set of basis functions.

cos(α)



+sin(α)



=

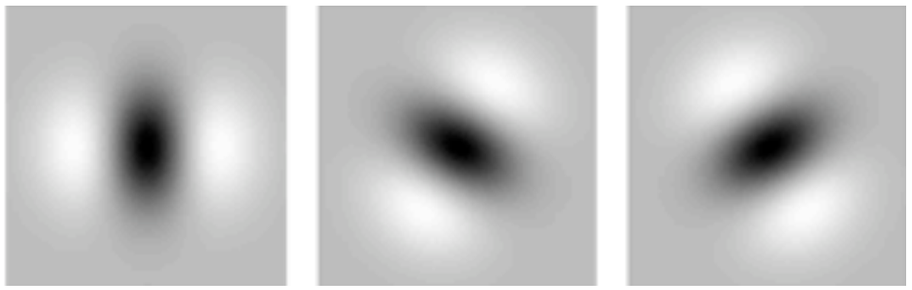
Freeman & Adelson 92

Question: how can we compute the optimal orientation at each point efficiently?

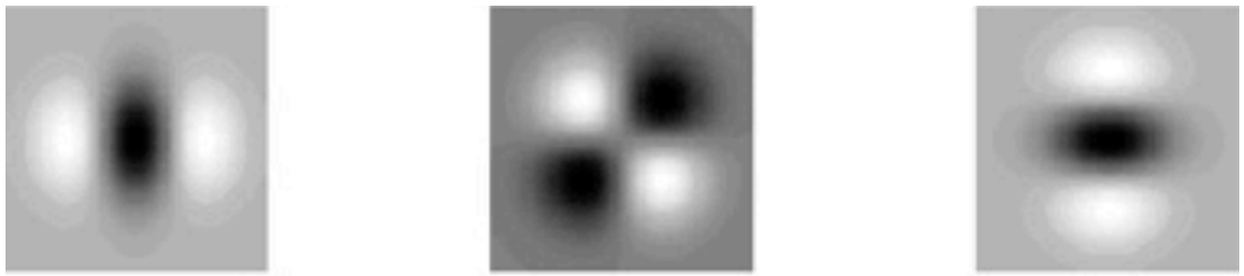
$$h_\alpha(x,y) = \begin{bmatrix}\cos(\alpha) & \sin(\alpha)\end{bmatrix}^T \begin{bmatrix}h_x(x,y) \\ h_y(x,y)\end{bmatrix}$$

Special case: second-derivatives of Gaussians

Steerable basis



Seperable and steerable



G_{2a} G_{2b} G_{2c}

G_{2a}	=	$0.9213(2x^2-1)e^{-(x^2+y^2)}$	$k_a(\theta)$	=	$\cos^2(\theta)$
G_{2b}	=	$1.843xye^{-(x^2+y^2)}$	$k_b(\theta)$	=	$-2\cos(\theta)\sin(\theta)$
G_{2c}	=	$0.9213(2y^2-1)e^{-(x^2+y^2)}$	$k_c(\theta)$	=	$\sin^2(\theta)$

Table 3: X-Y separable basis set and interpolation functions for second derivative of Gaussian. To create a second derivative of a Gaussian rotated along to an angle θ, use: $G_2^\theta = (k_a(\theta)\, G_{2a} + k_b(\theta)\, G_{2b} + k_c(\theta)\, G_{2c})$. The minus sign in $k_b(\theta)$ selects the direction of positive θ to be counter-clockwise.

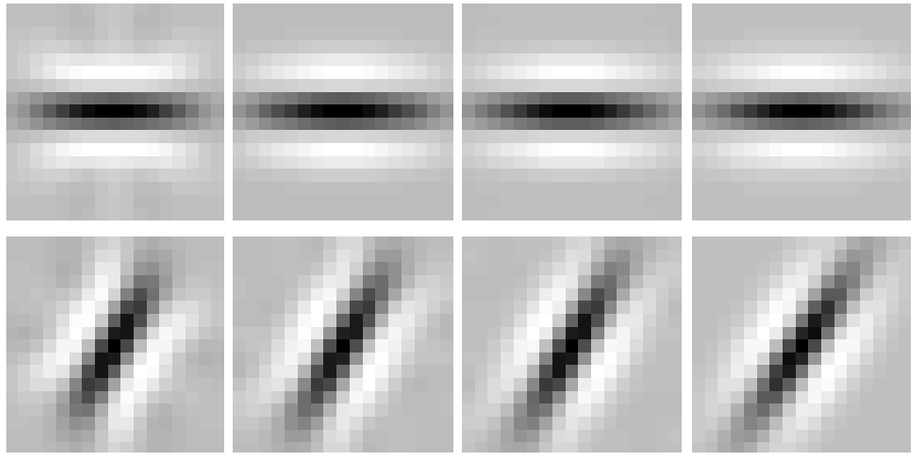
When is this possible?

When filters are smooth in “orientation space”

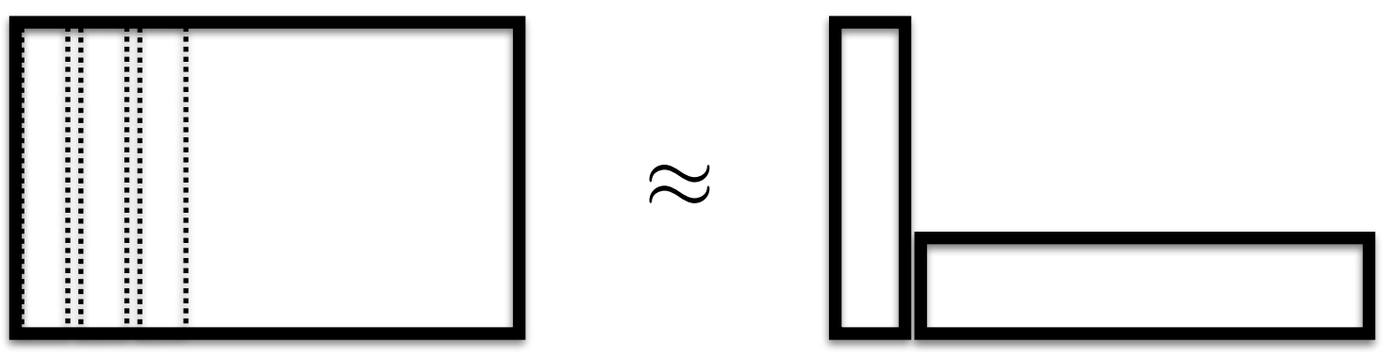
We’ll need some additional math to derive this… ignore for now

Least-squares method of steerability

Shy & Perona, CVPR94



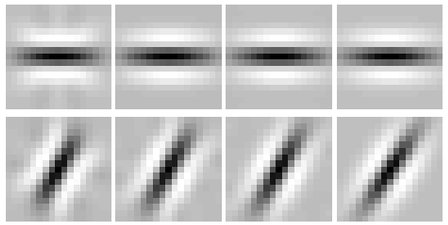
Stack bank of filters into a matrix
Apply SVD to generate low-rank approximation



Least-squares method of steerability

Shy & Perona, CVPR94

Rank 1 approximation



$$H[u,v,k] = H_s[u,v]c[k]$$

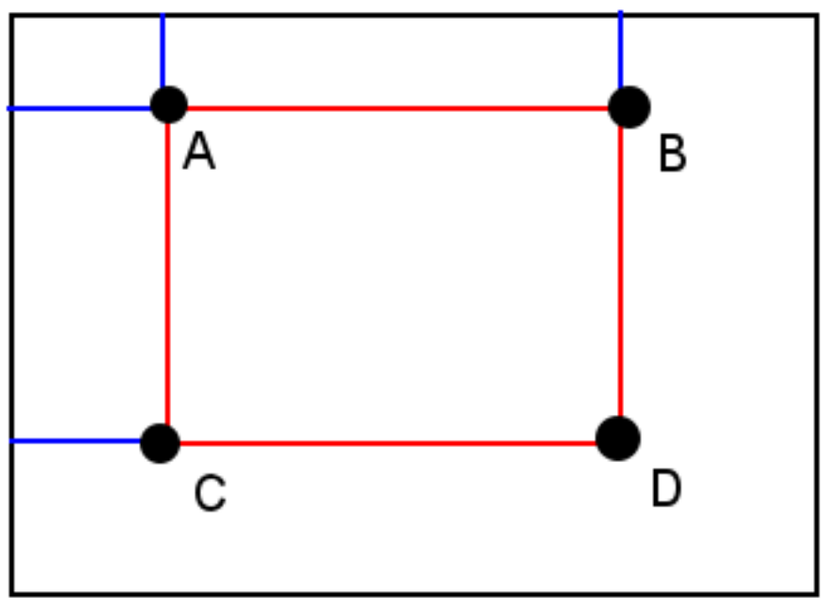
$$G[i,j,k] = \sum_u \sum_v H[u,v,k]F[i+u,j+v]$$

Box filtering with integral images

http://en.wikipedia.org/wiki/Summed_area_table

$$I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y')$$

$$I(x,y) = i(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1)$$



Sum = D - B - C + A

Reduces $O(N^2M^2)$ to $O(N^2)$

A look back

- Edges (Canny, hysteresis, LoG)
- Filter banks (Gabors)
- Efficiency (pyramids, separability, steerability)
- Next class: bag-of-words, linear algebra, frequencies