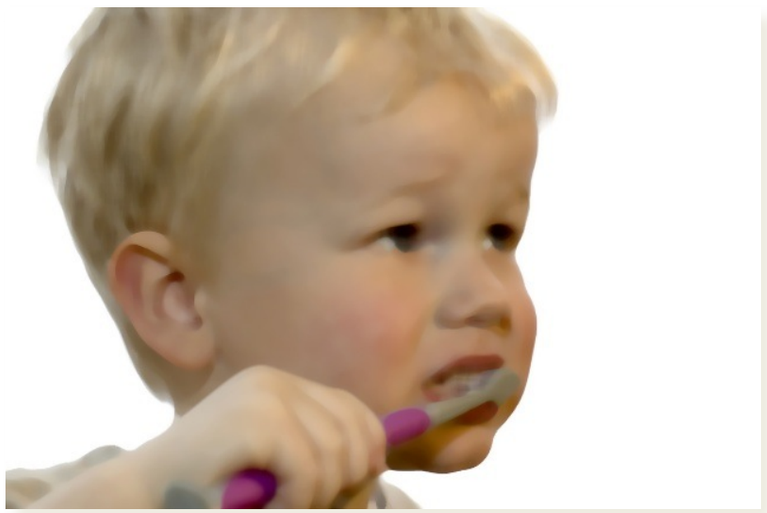
Edges

Original

Gradient Magnitude

Gaussian Blur



Median

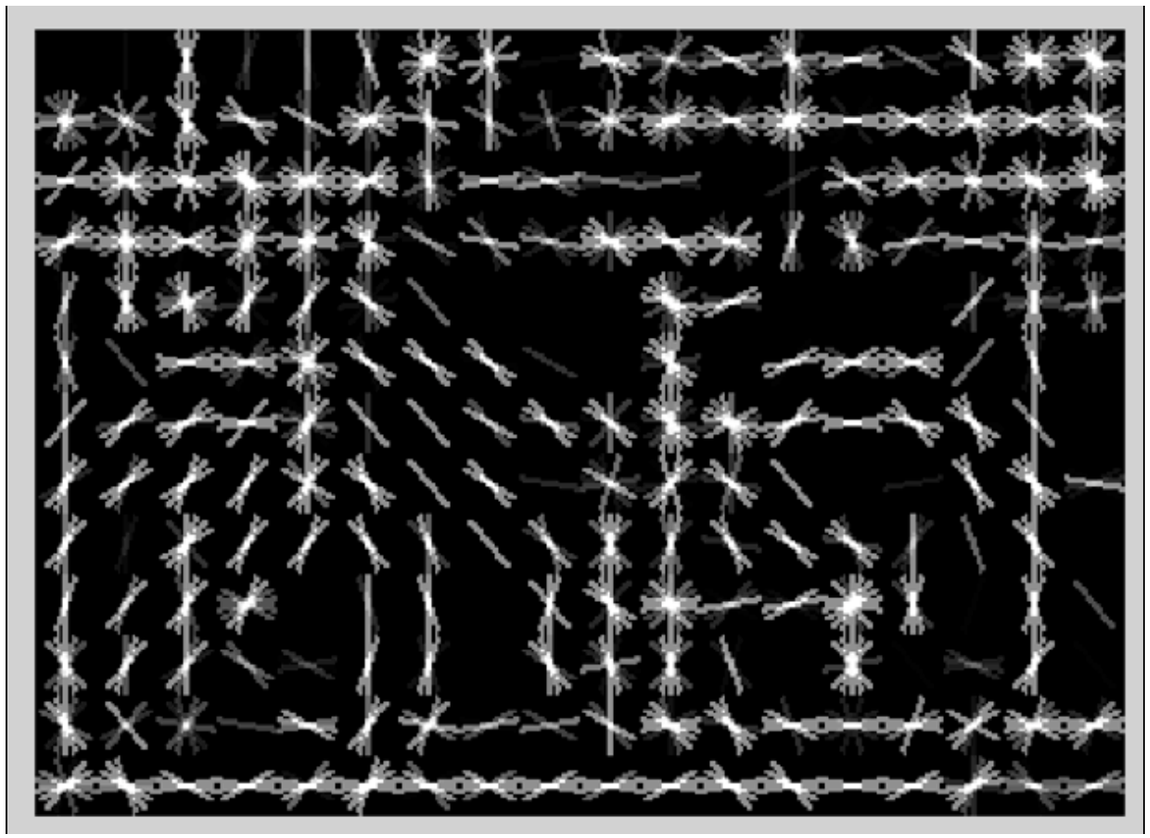
Adaptive Thresholding

Bilateral

Outline

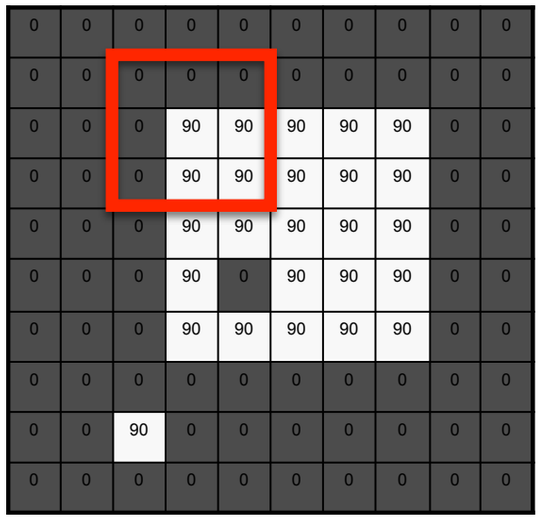
- Convolution
  - Linear Shift Invariance (LSI)
  - Convolution Properties (commutative, associative, distributive)
  - Normalized Cross-Correlation
- **Edges**
  - Canny edges (hysterisis)
  - derivative-of-gaussians (DoG)
  - laplacian-of-Gaussians (LoG)
- Efficiency
  - pyramids
  - linear approximiations (SVD, separability)
  - steerability

Where we are headed (HW1)



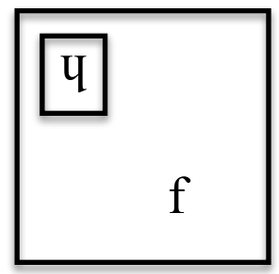
Lecture 1 Recap

Any linear shift-invariant (LSI) image operation must be representable as a convolution of an image F[i,j] with an *impulse response/ filter* H[i,j]

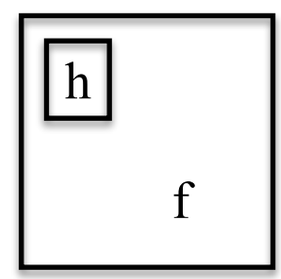


1	2	1
2	4	2
1	2	1

Convolution: $$G[i,j] = F * H = \sum_{u,v} F[u,v]H[i-u,i-v]$$



(Cross)-Correlation: $$G[i,j] = F \otimes H = \sum_{u} \sum_{v} H[u,v]F[i+u,j+v]$$



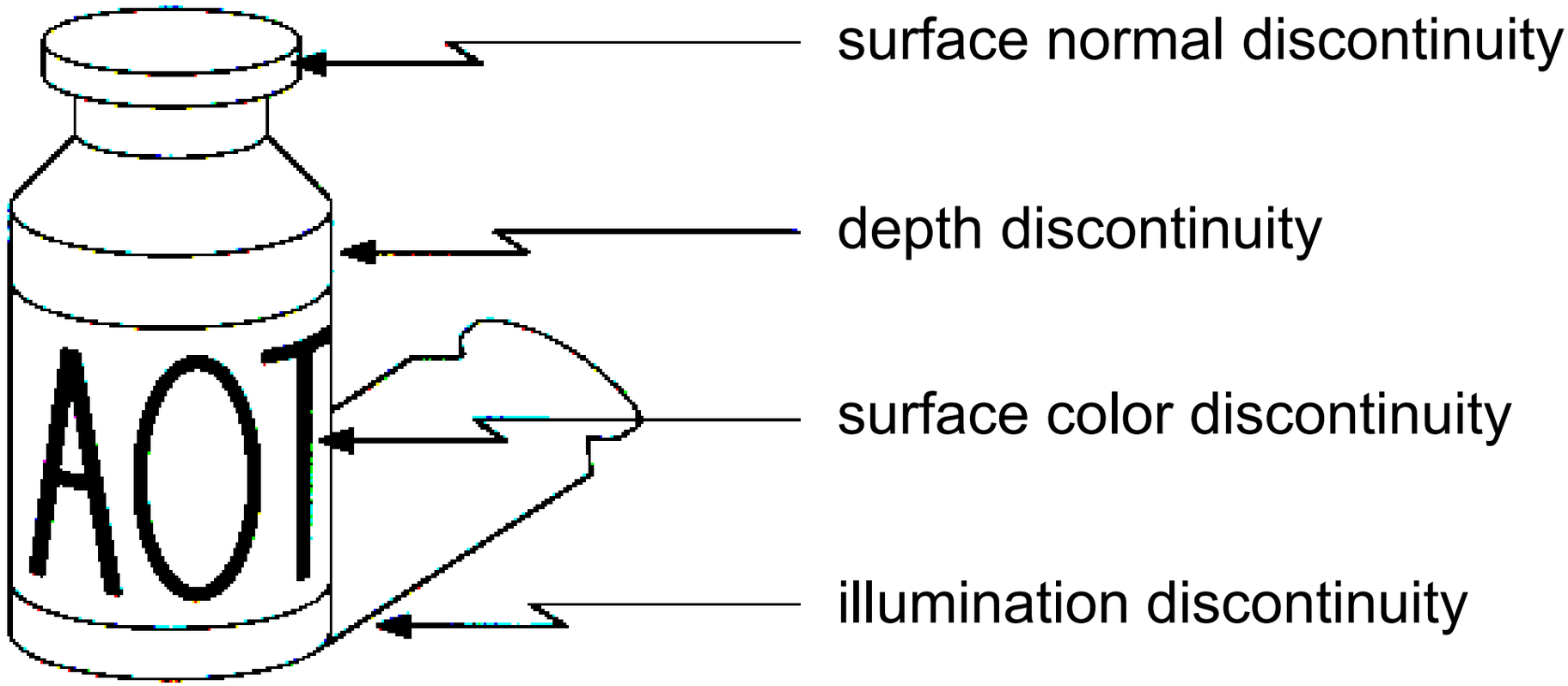
Convolution from-scratch

```
def convolve2d(image, filter):
    # Perform 2D convolution with 'valid'-size output
    # using vector dot products
    ny, nx = image.shape
    my, mx = filter.shape

    # Define output
    oy  = ny - my + 1
    ox  = nx - mx + 1
    res = np.zeros((oy,ox))

  #Flip filter left-right and up-down
  filter = filter[::-1,::-1]

  #Turn filter into a vector
  filter = filter.flatten()

  for i in range(oy):
    for j in range(ox):
      res[i,j] = np.dot(filter,image[i:i+my, j:j+mx].flatten())
  return res
```

1. How to modify code to perform “full” convolution?
2. How to modify code to perform convolution over a bank of filters?

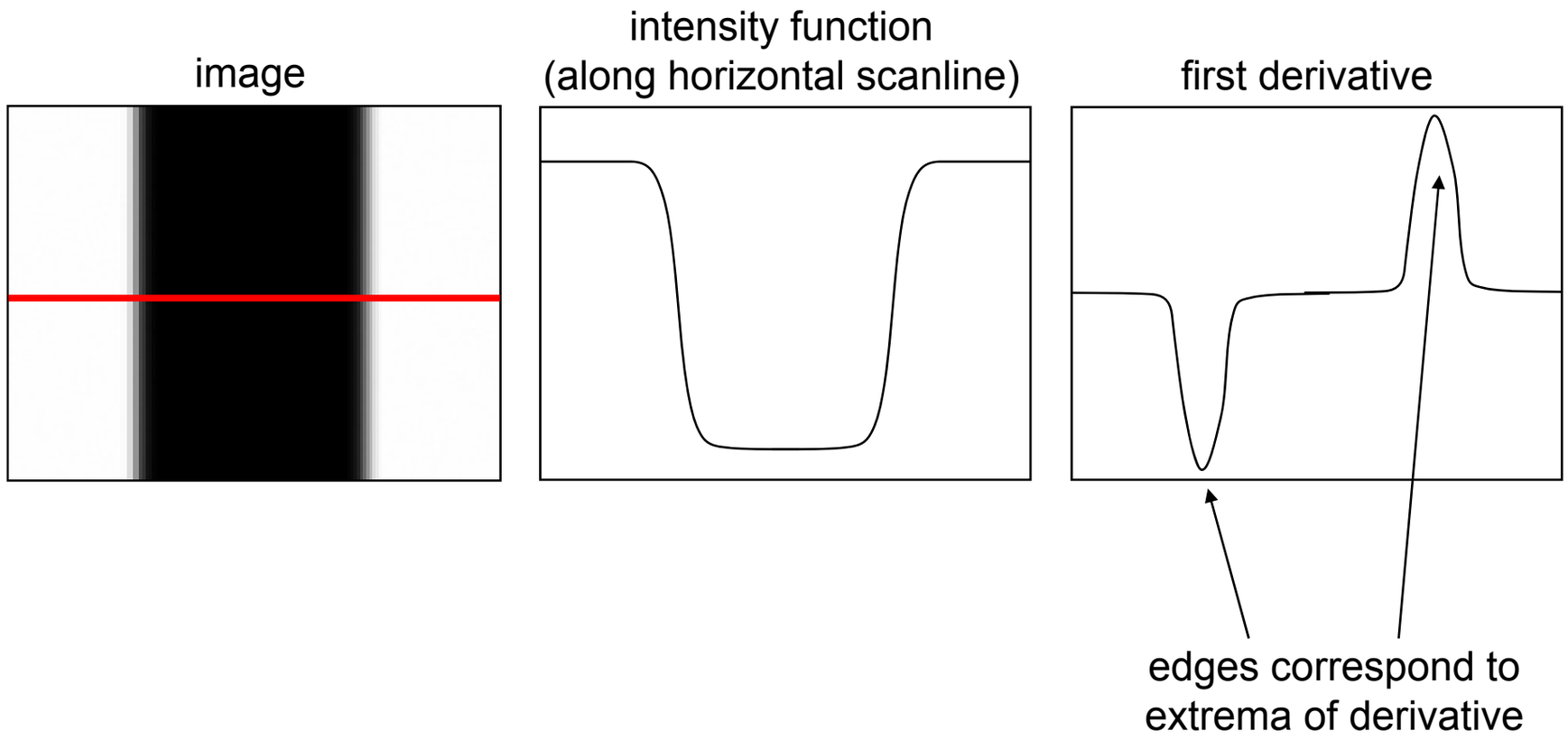
zero-pad the input image

replace vector-vector dot product with matrix-vector product

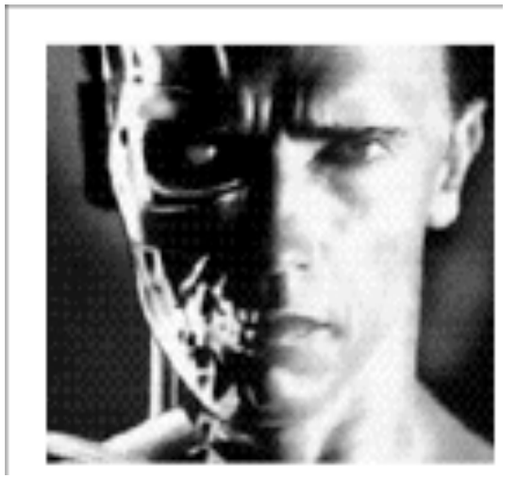

surface normal discontinuity
depth discontinuity
surface color discontinuity
illumination discontinuity

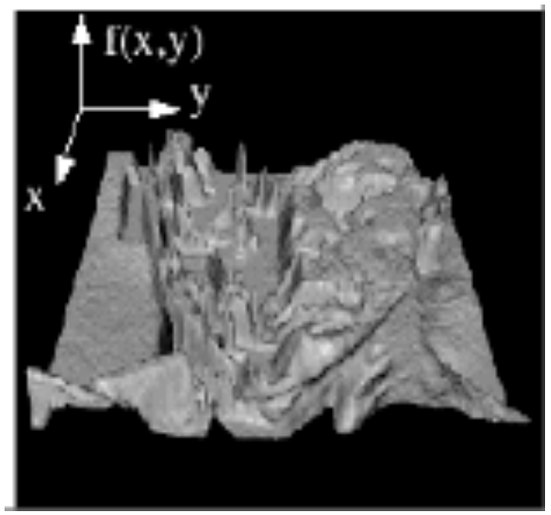
Characterizing edges

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



Continuous derivatives





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

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

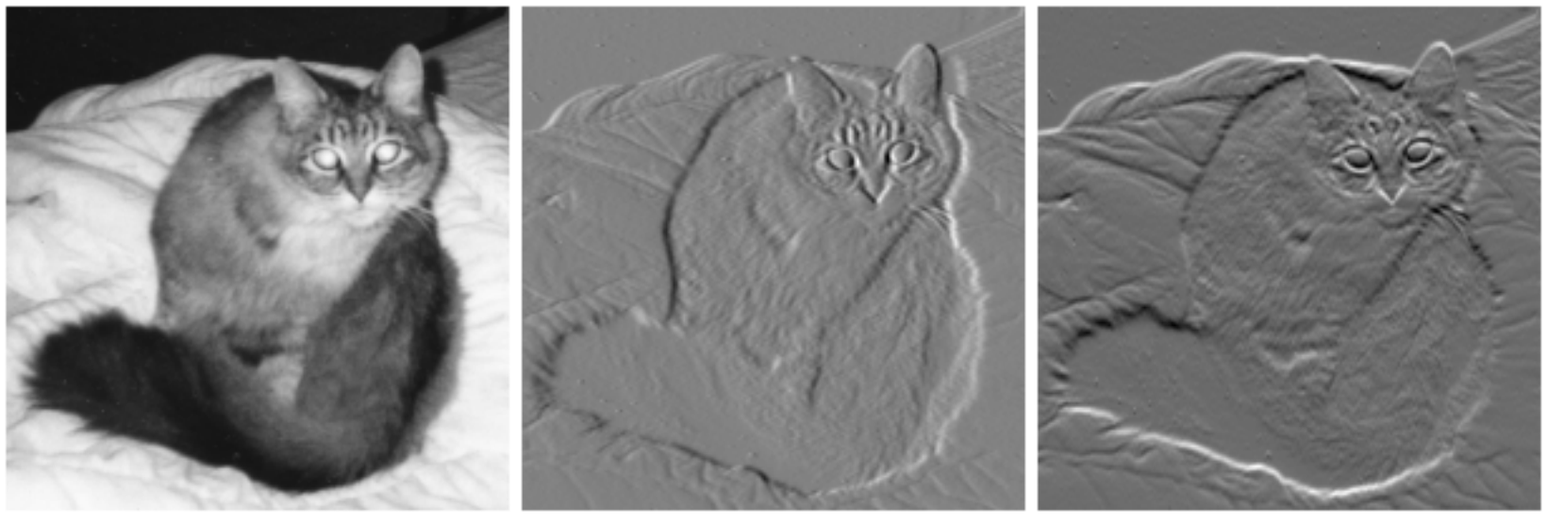
Gradient Orientation: $$\alpha = \tan^{-1} \frac{f_y}{f_x}$$

Derivative filters

Is this visualizing correlation or convolution with the given filters?

correlation!

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



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

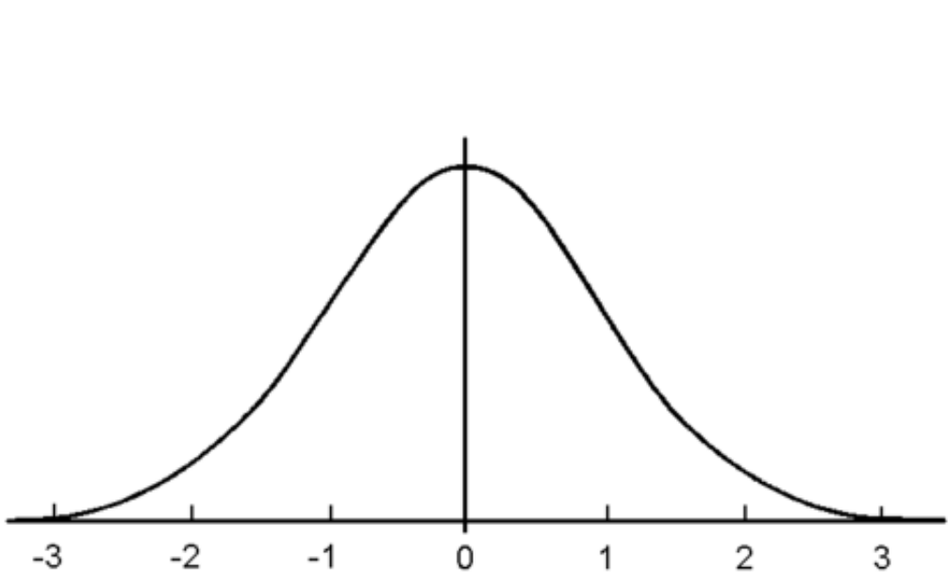
Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Why might these work better?

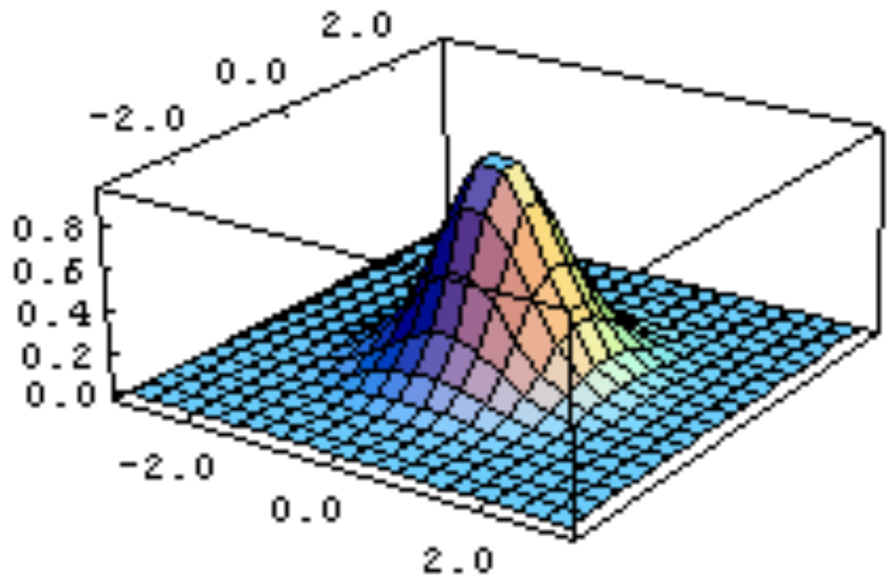
Background: smoothing with Gaussians

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

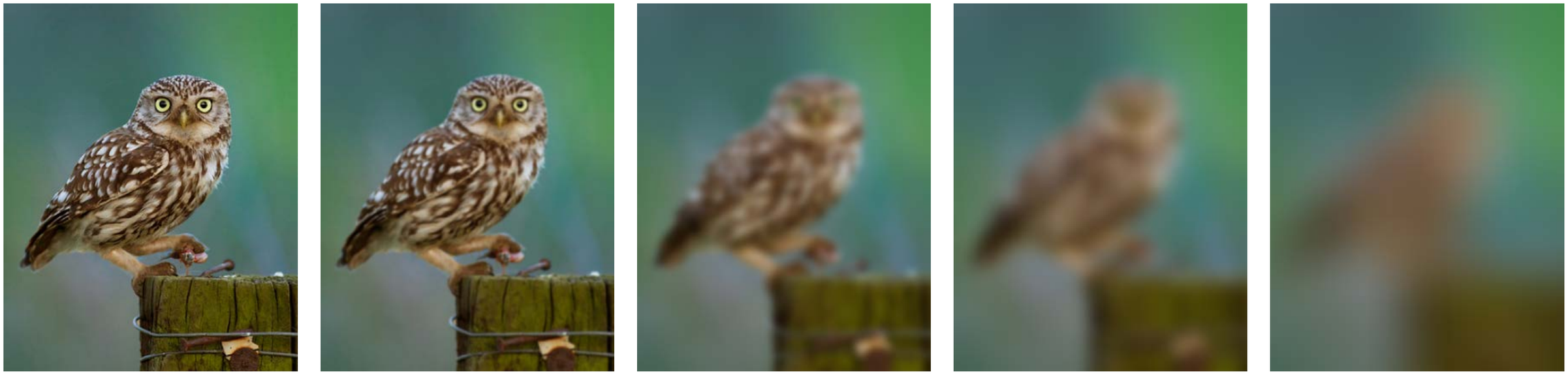
$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



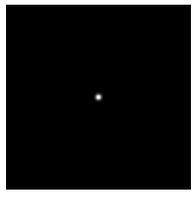
One-dimensional



Two-dimensional

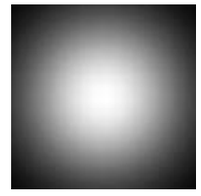


σ = 1 pixel



σ = 5 pixels

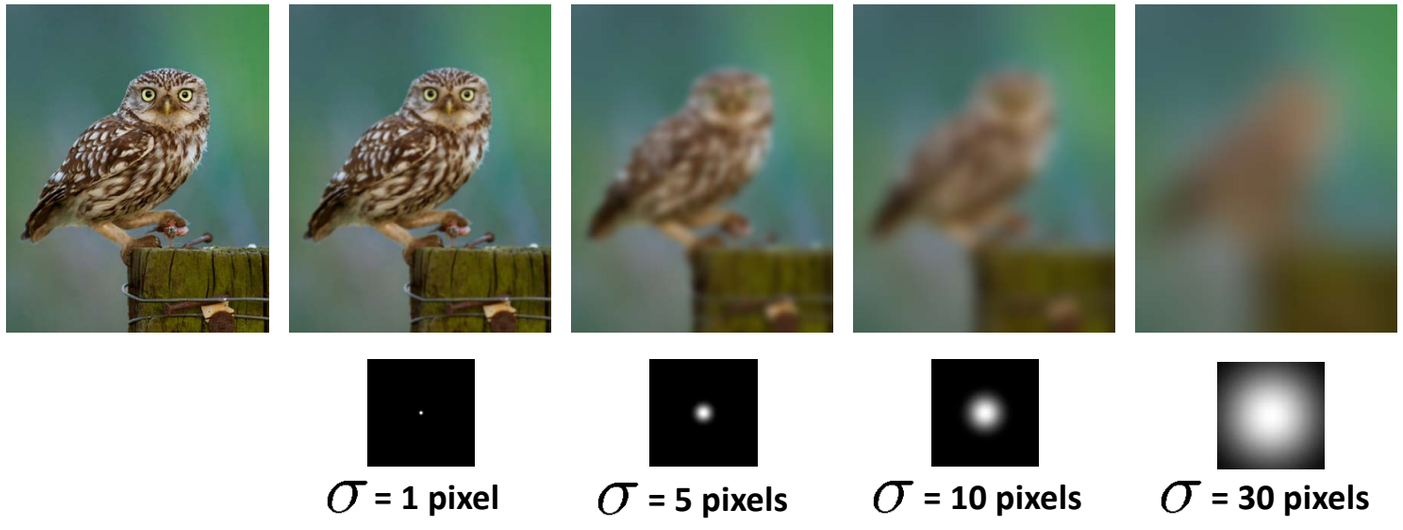
σ = 10 pixels

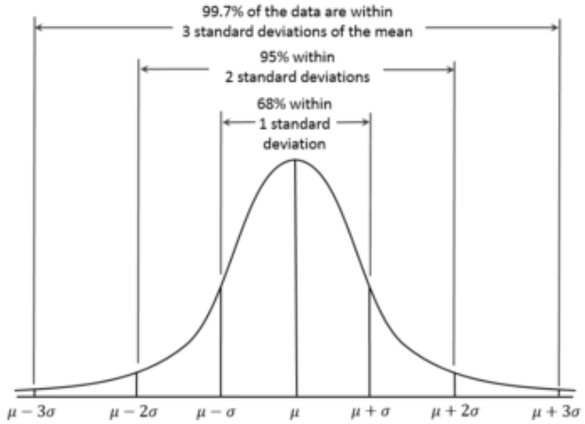


σ = 30 pixels

Question: how do we set spatial extent of a Gaussian filter
(e.g., sigma=1 pixel has lots of 0's which are inefficient?)

Rule-of-thumb





Set radius of filter to be 3 sigma

3-tap Gaussian filter: $\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$

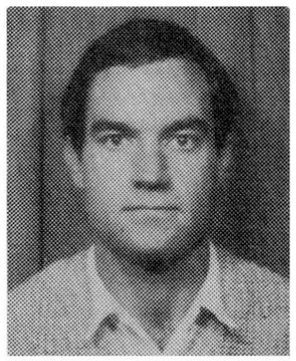
5-tap Gaussian filter: $\frac{1}{159}\begin{bmatrix}2 & 4 & 5 & 4 & 2\\ 4 & 9 & 12 & 9 & 4\\ 5 & 12 & 15 & 12 & 5\\ 4 & 9 & 12 & 9 & 4\\ 2 & 4 & 5 & 4 & 2\end{bmatrix}$

Outline

- Convolution
  - Linear Shift Invariance (LSI)
  - Convolution Properties (commutative, associative, distributive)
  - Normalized Cross-Correlation
- **Edges**
  - **Canny edges (hysterisis)**
  - derivative-of-gaussians (DoG)
  - laplacian-of-Gaussians (LoG)
- Efficiency
  - pyramids
  - linear approximiations (SVD, separability)
  - steerability

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.

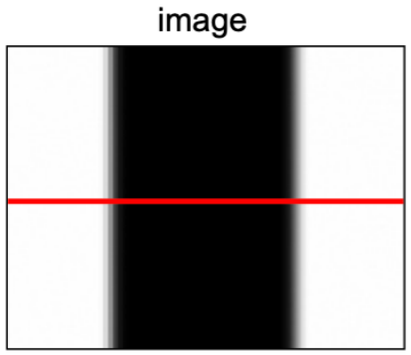
He is with the Artificial Intelligence Laboratory, M.I.T. His research interests include low-level vision, model-based vision, motion planning for robots, and computer algebra.

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

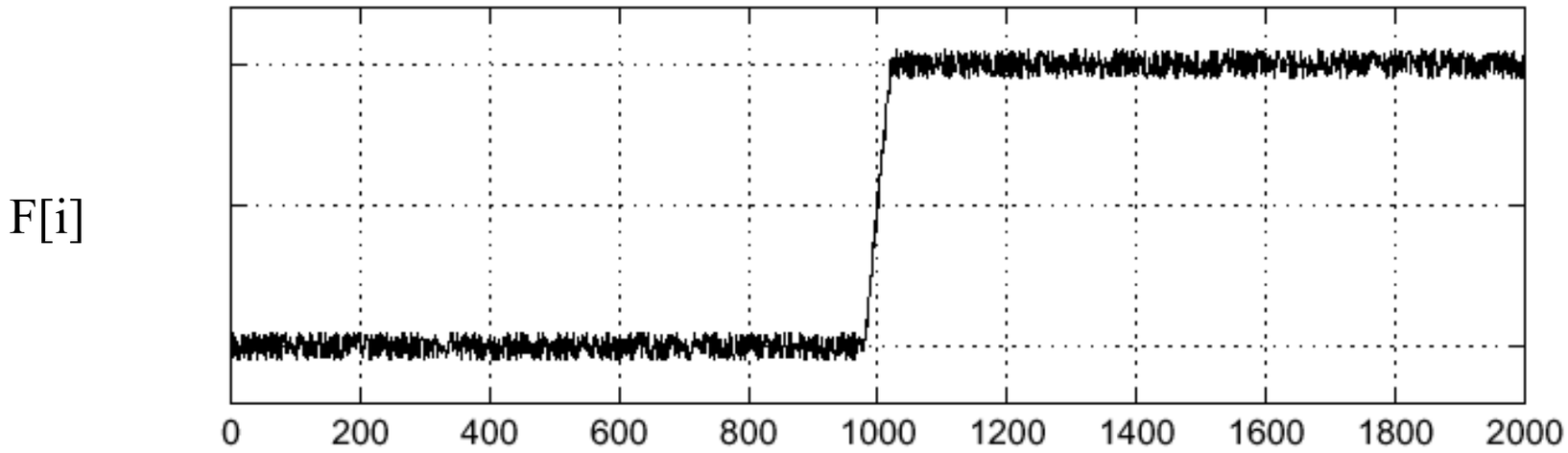
Criteria: we want to detect and localize edges

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

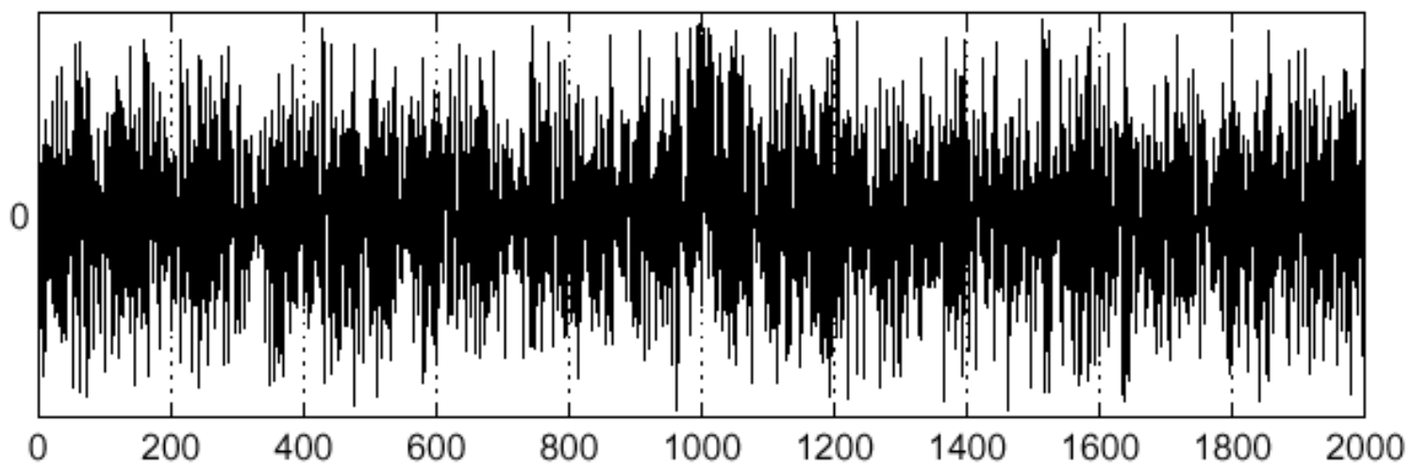
Revisiting gradients



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

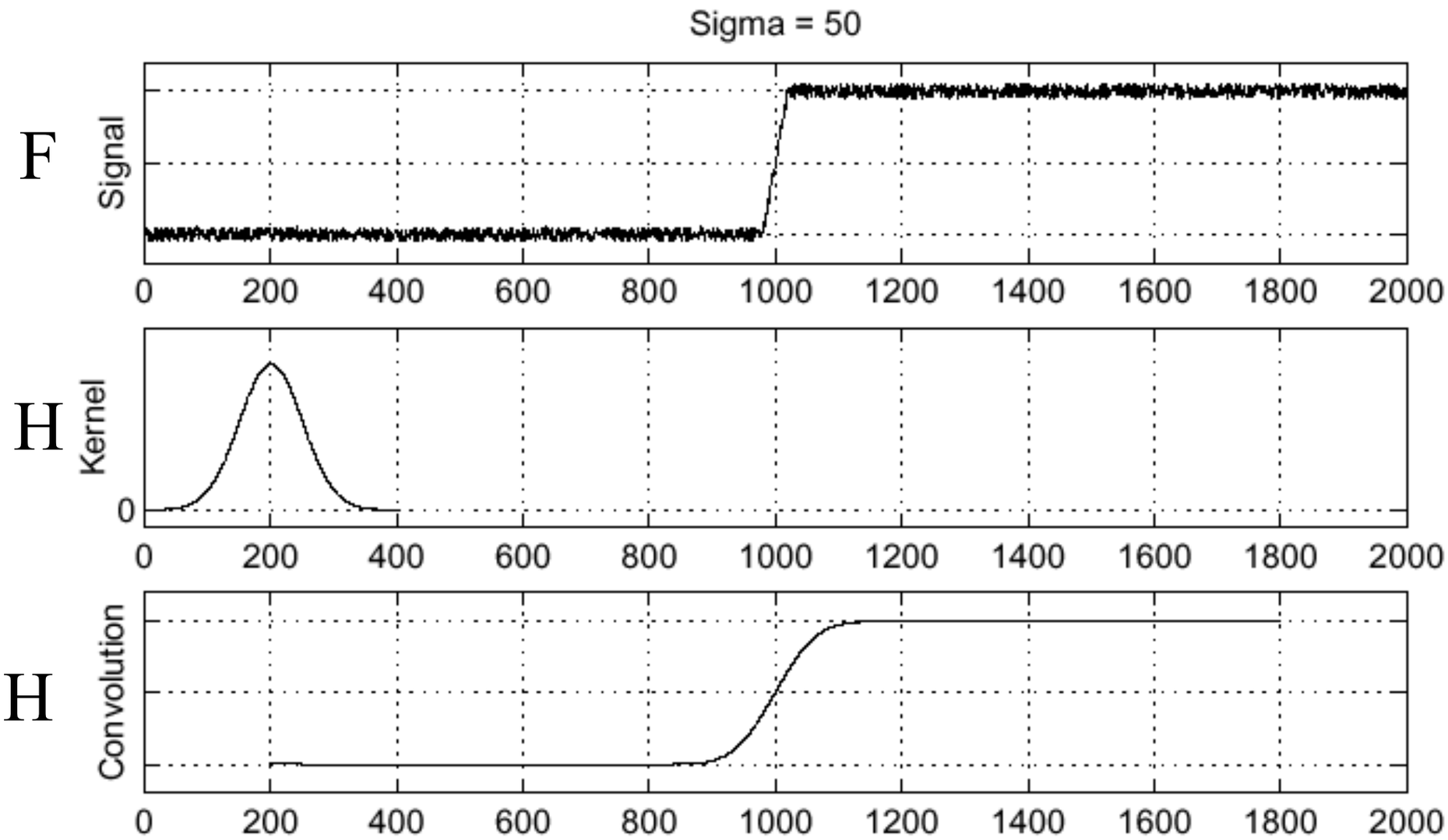


$$\frac{\partial F}{\partial i} \approx F \otimes [-1, 1]$$
$$= F * [1, -1]$$



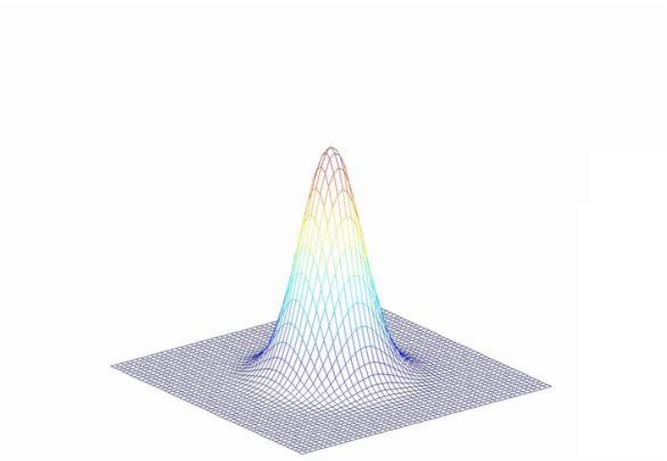
How can we find pixels with high gradients and not find the noise?

Solution: smooth first

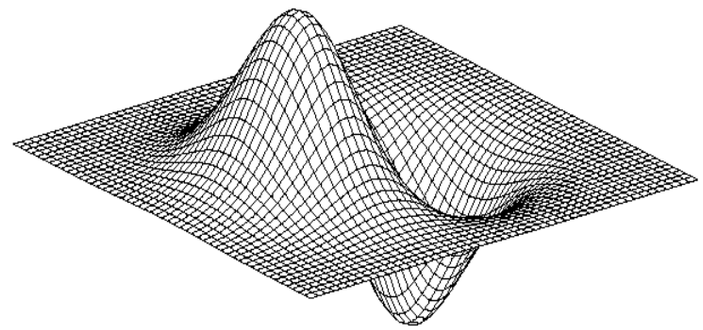


(F*H)*[1 -1]

Derivative of Gaussian filter

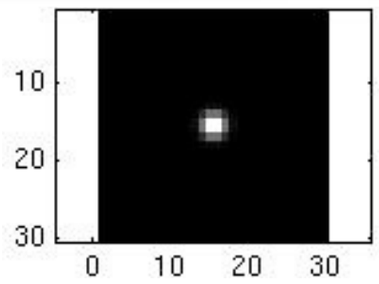


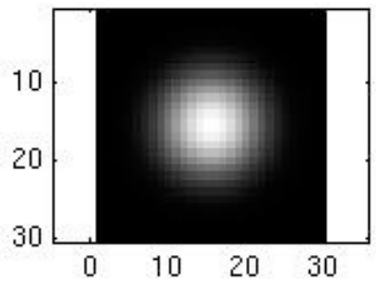
$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * [-1, 1] =$$



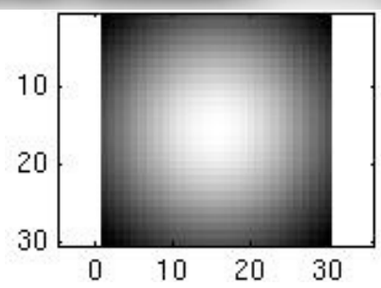
Effect of σ on Gaussian smoothing

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





...



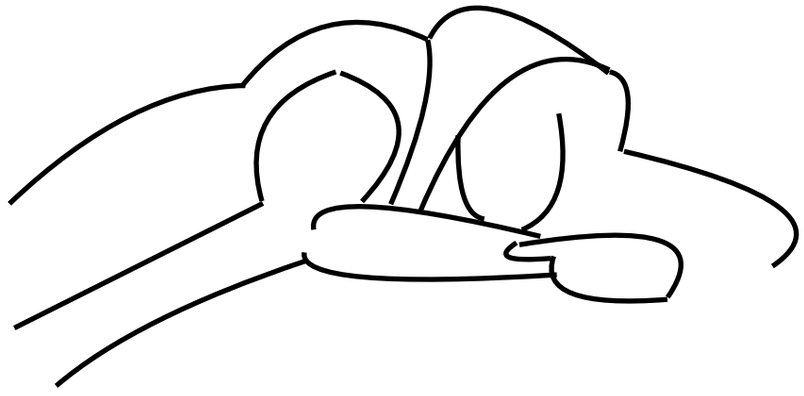
Effect of σ on derivatives

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

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

The Canny edge detector

(ideal*ish* goal)



The Canny edge detector

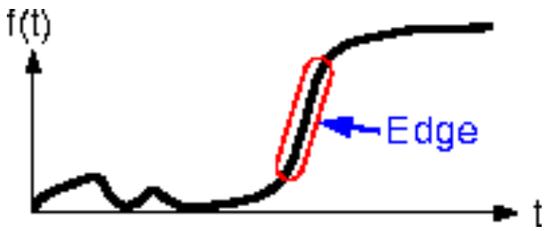
norm of the smoothed gradient

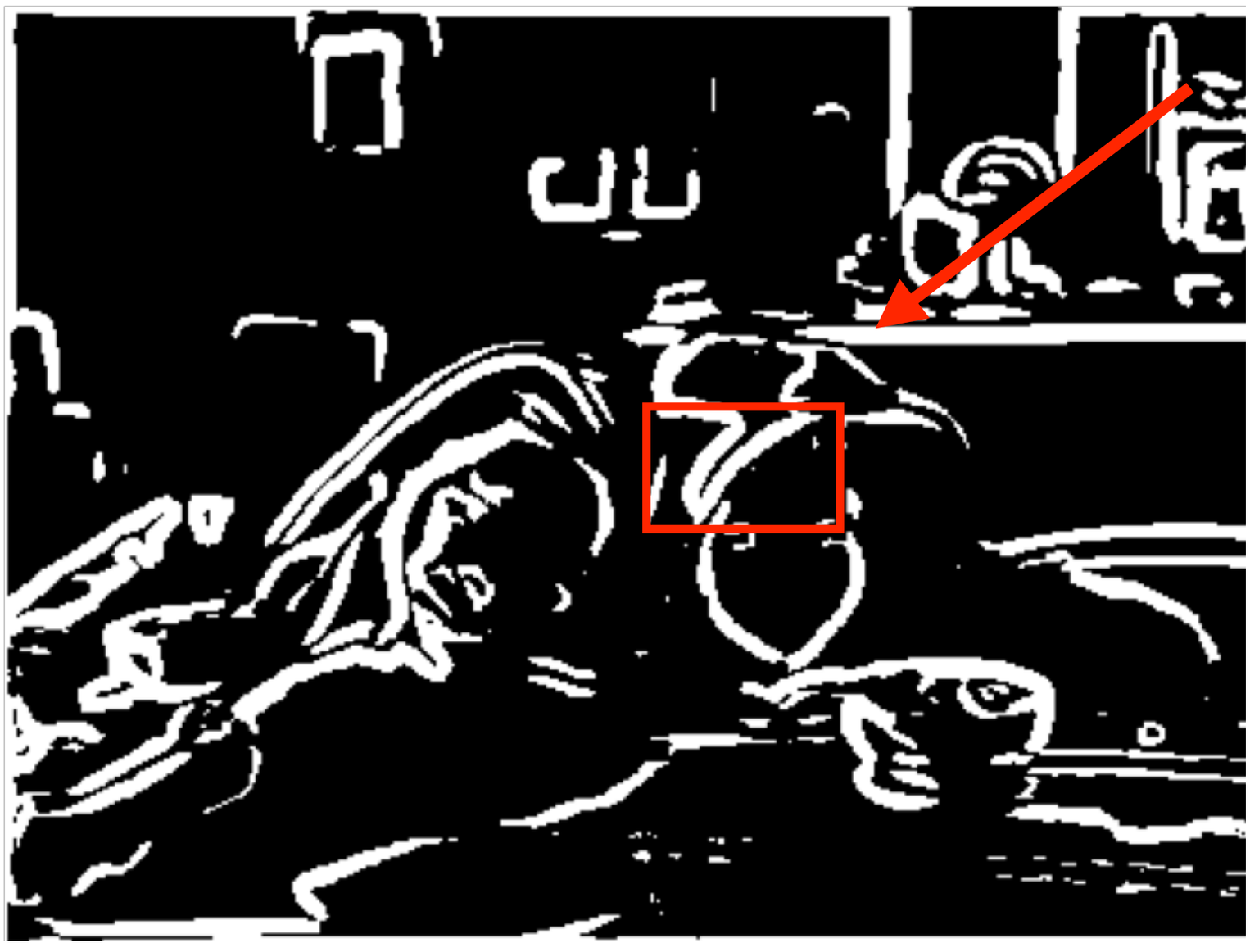
The Canny edge detector

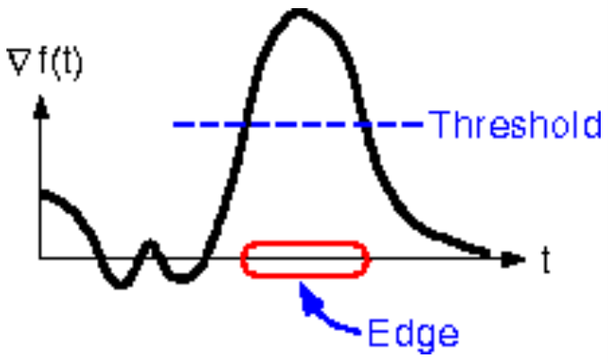
thresholding

The Canny edge detector

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

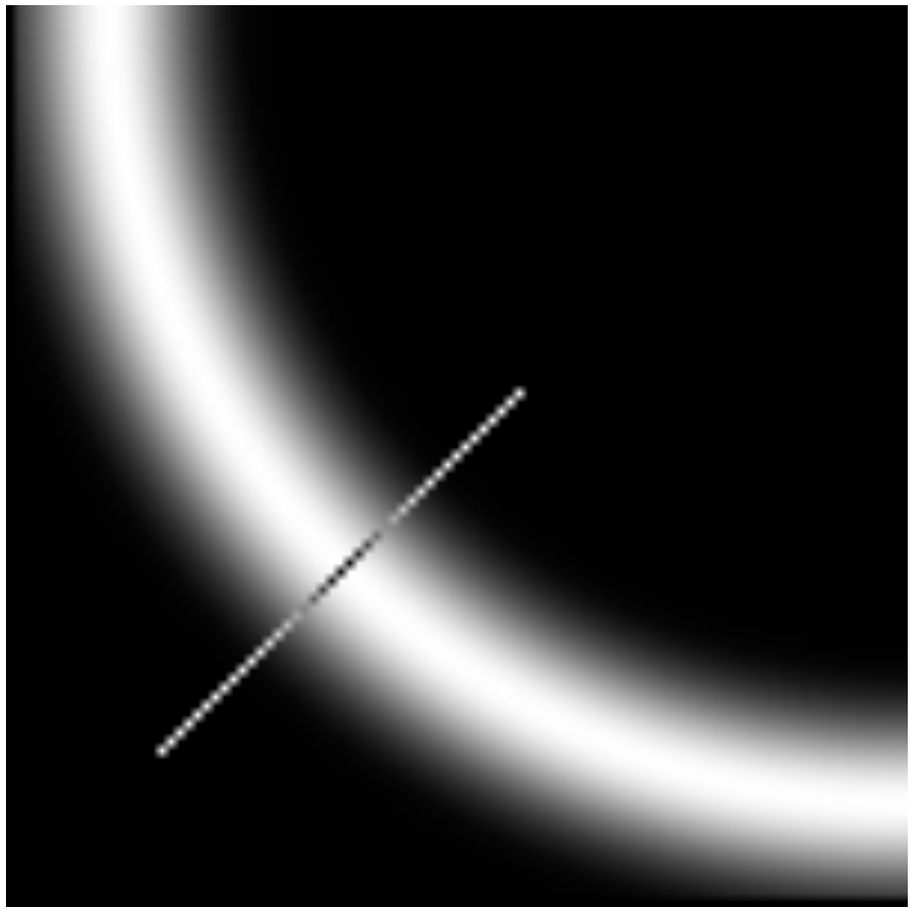


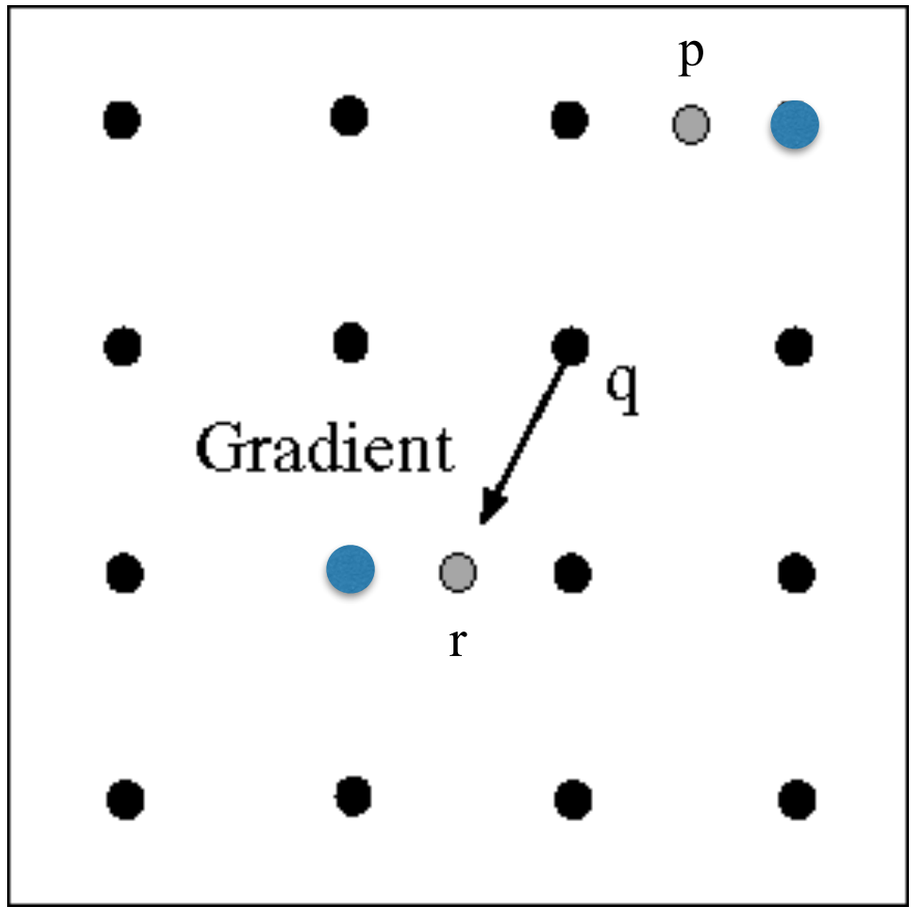




thresholding

Non-maximum suppression





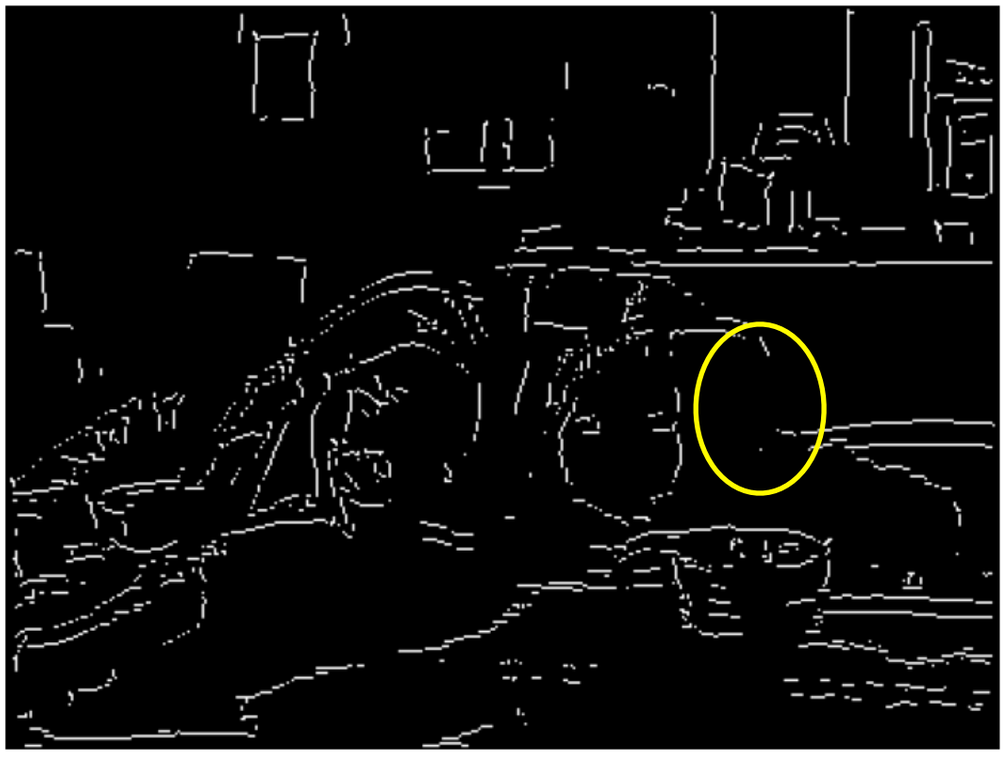
Check if pixel q is larger than neighbors (p,r) along gradient direction

- requires checking (nearest-neighbor) interpolated pixels p and r

The Canny edge detector

thinning

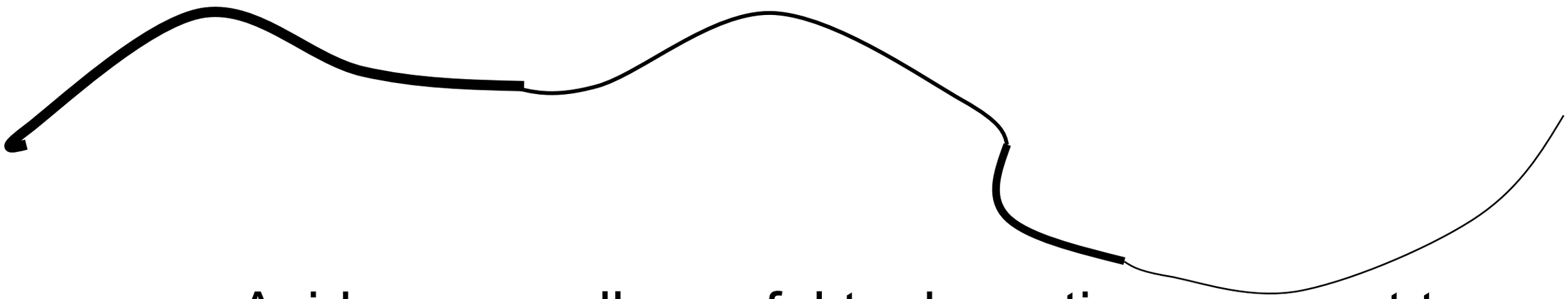
(non-maximum suppression)



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

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.

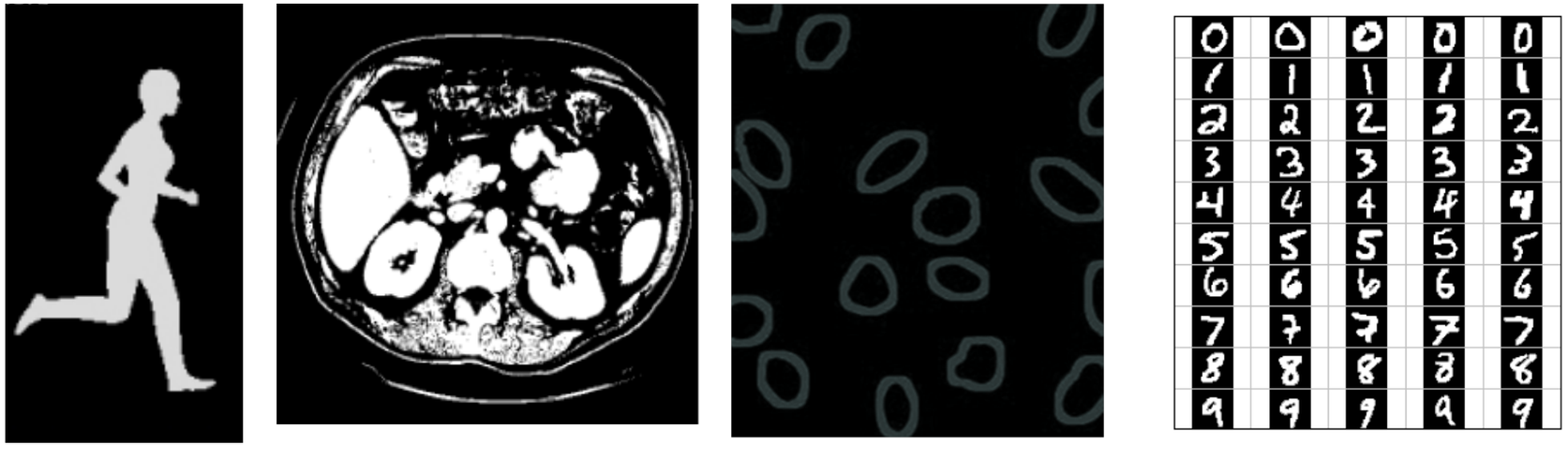


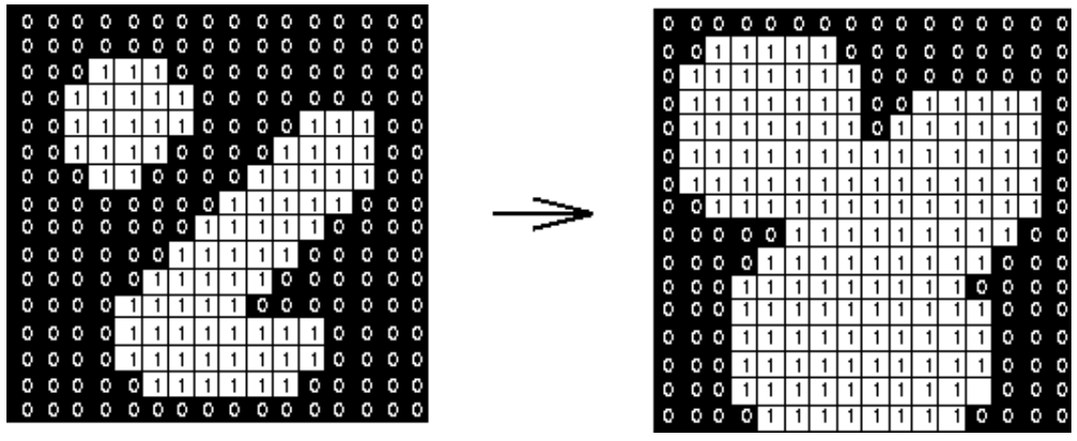
- Aside: generally useful took anytime we want to find curves in images / videos (e.g., tracking as spacetime curve fitting!)

Source: S. Seitz

Aside: morphological image processing

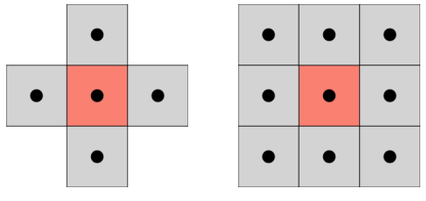
Hystersis is can be implemented through image *dilation*, which is a special case of *nonlinear binary image processing*





$$G[i,j] = \max_{u,v \in SE} F[i+u, j+v]$$

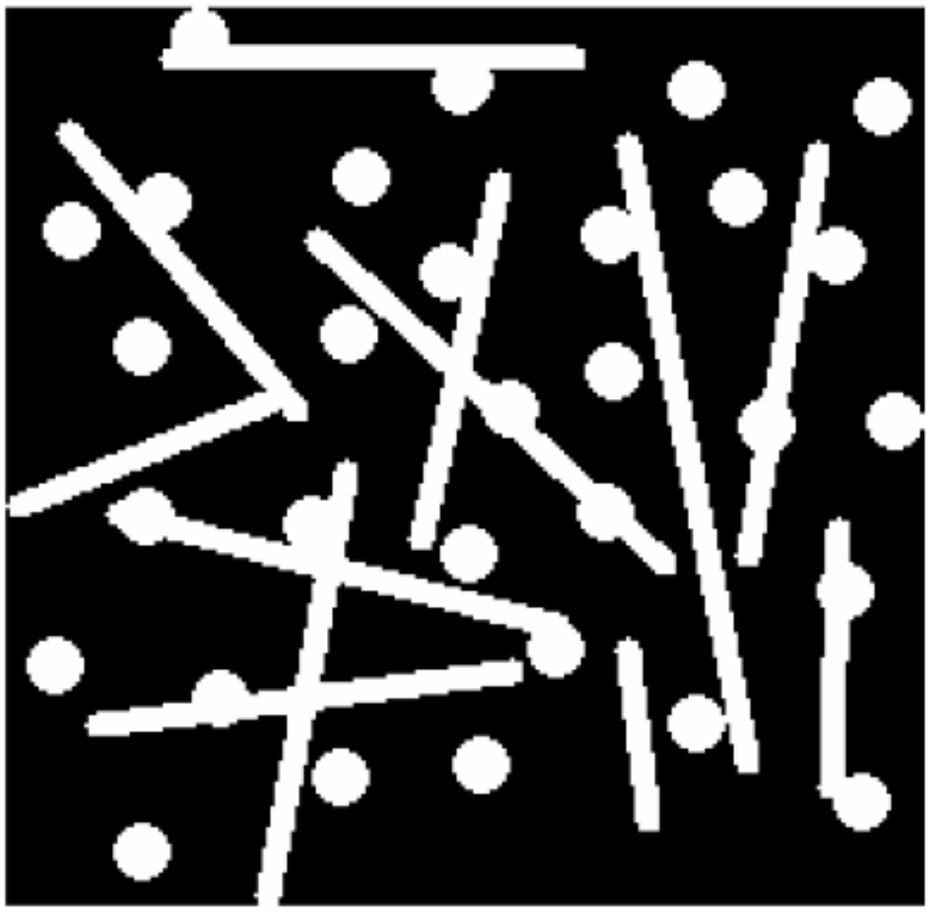
SE = "structuring element" mask



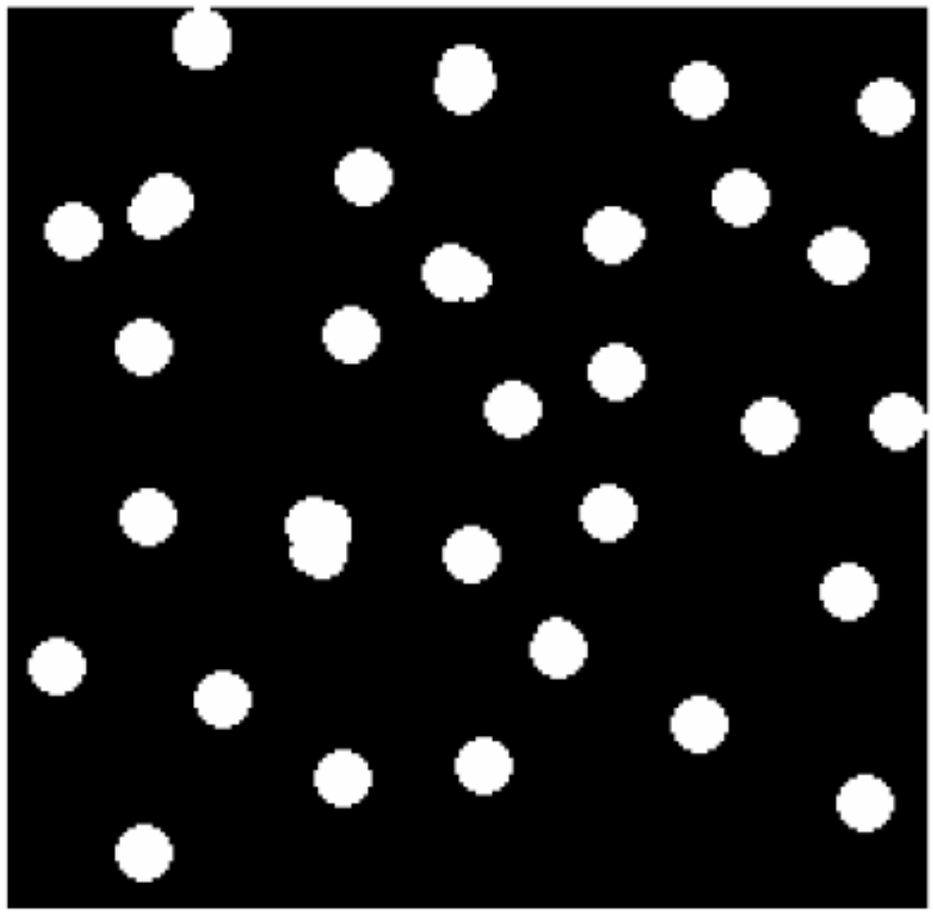
Erosion corresponds to min'ing rather than maxing

Opening

- Erode, then dilate
- Remove small objects, keep original shape



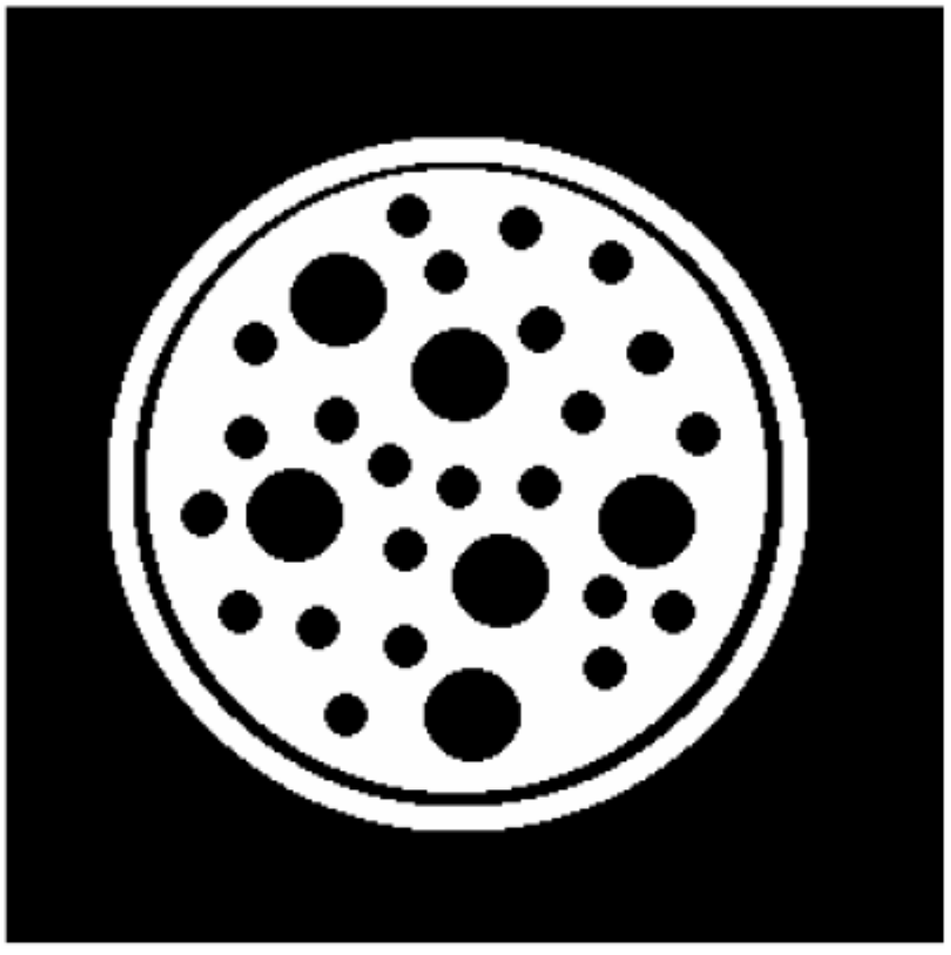
Before opening



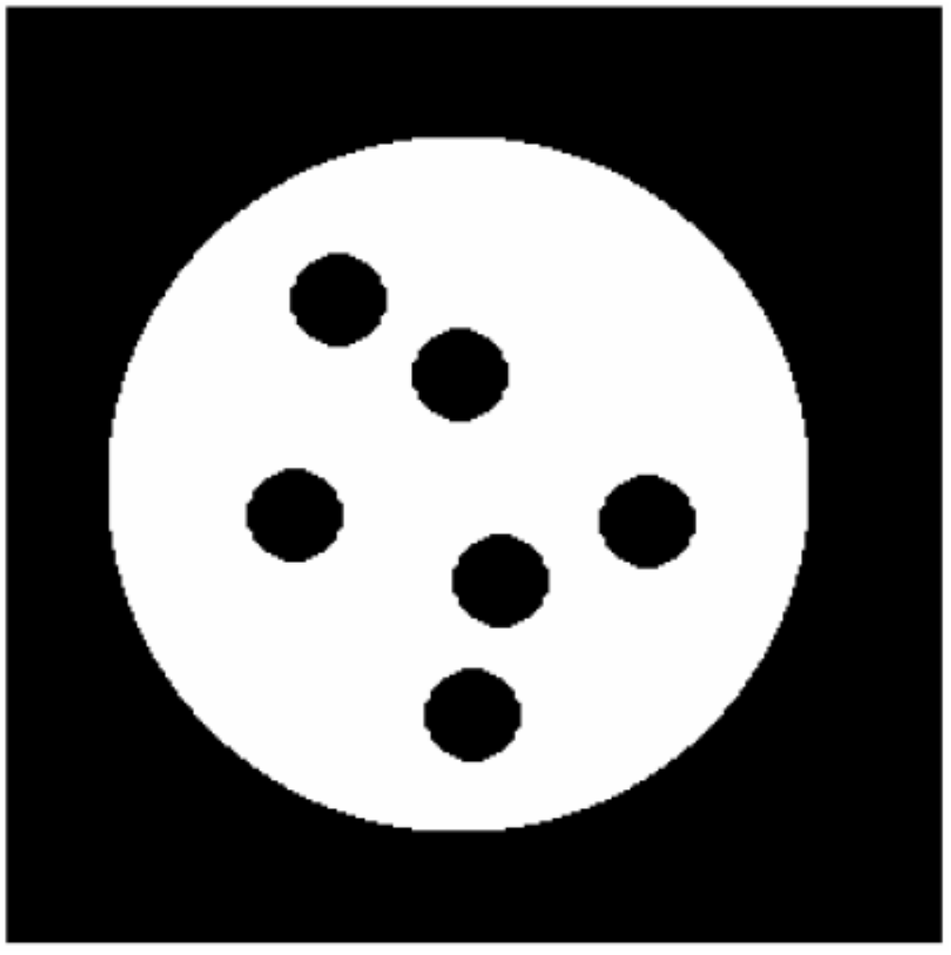
After opening

Closing

- Dilate, then erode
- Fill holes, but keep original shape



Before closing



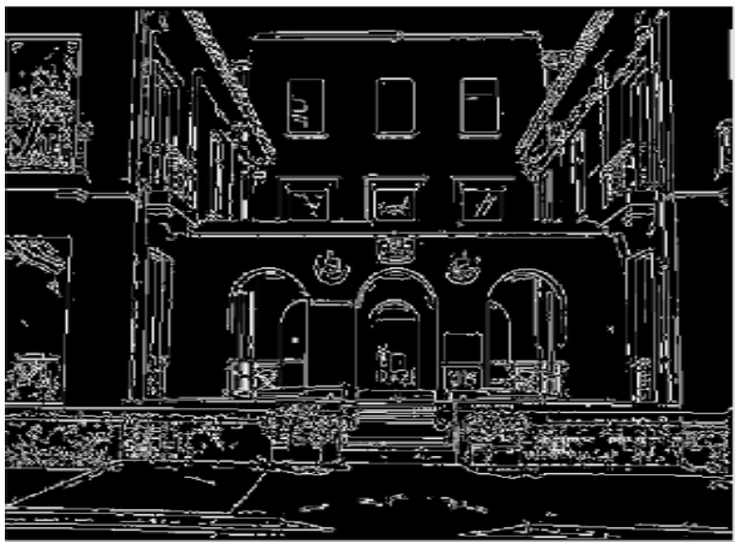
After closing

Hysteresis thresholding

original image

high threshold
(strong edges)

low threshold
(weak edges)

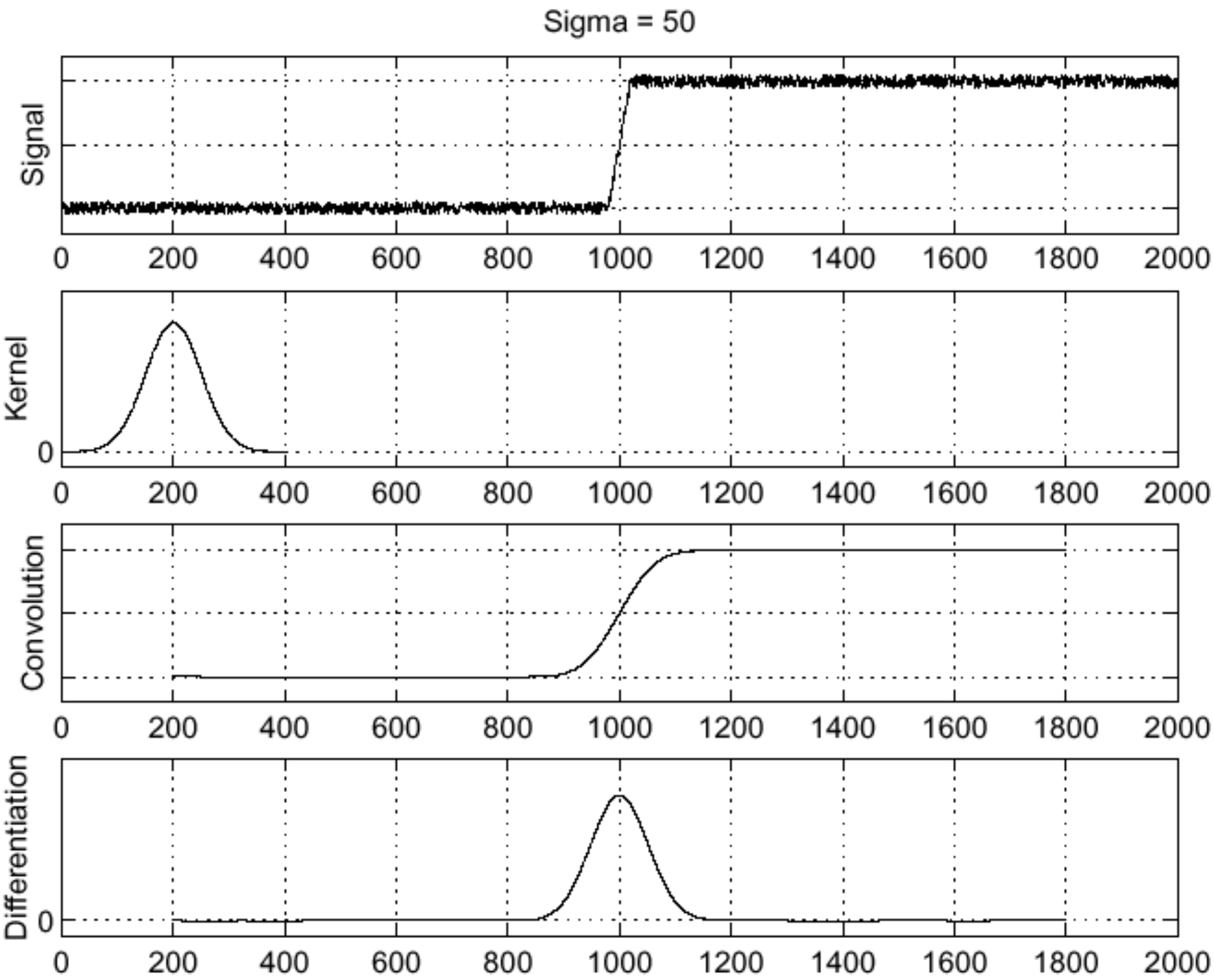


hysteresis threshold

Source: L. Fei-Fei

Alternate approach to edge-finding

F



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

What does the resulting edge filter look like?

Boring math for (finite approximation of) second-derivative filter

"Second derivative filter" G = [1 -1] convolved with [1 -1]

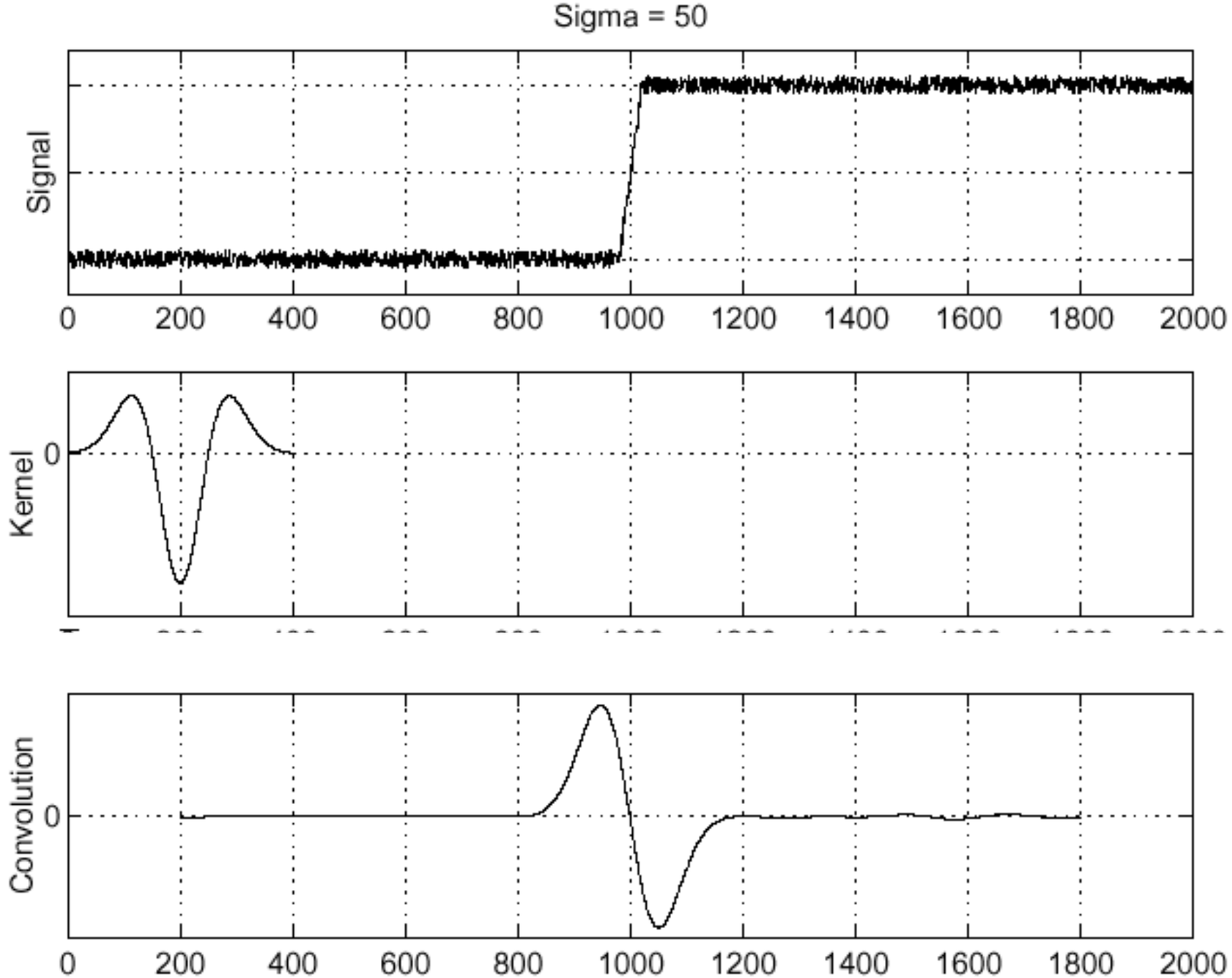
$$G[0] = 0 * -1 + 1 * 1 + -1 * 0 = 1$$
$$G[1] = -1 - 1 = -2$$
$$G[2] = 1 * 0 + -1 * -1 + 0 * 1 = 1$$

[Apologies; on this slide, "*" refers to standard multiplication and not convolution]

$$\mathrm{G} = [1\ \text{-}2\ 1]$$

Look for zero-crossings of second derivative

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

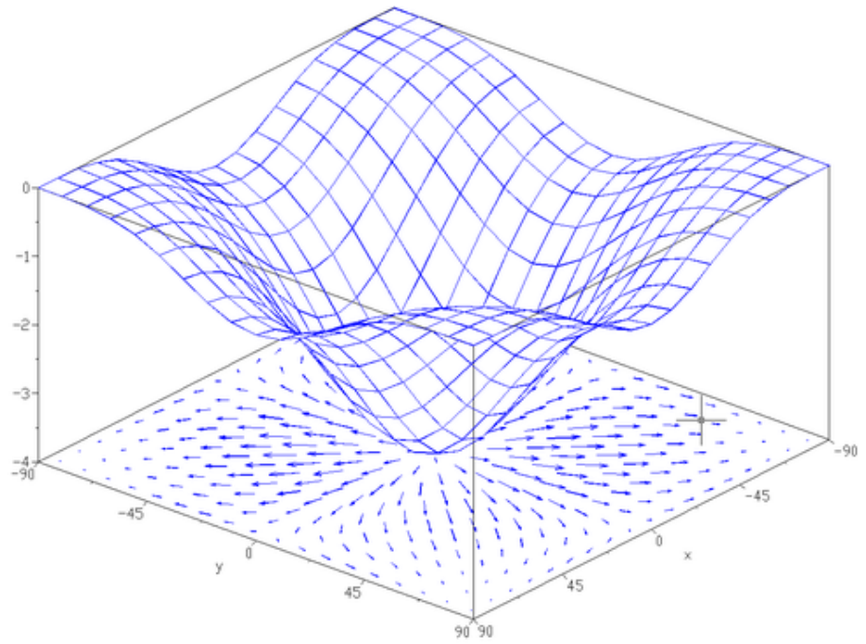


Zero-crossings of second graph

Generalization to 2D

Add second derivatives in each dimension

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



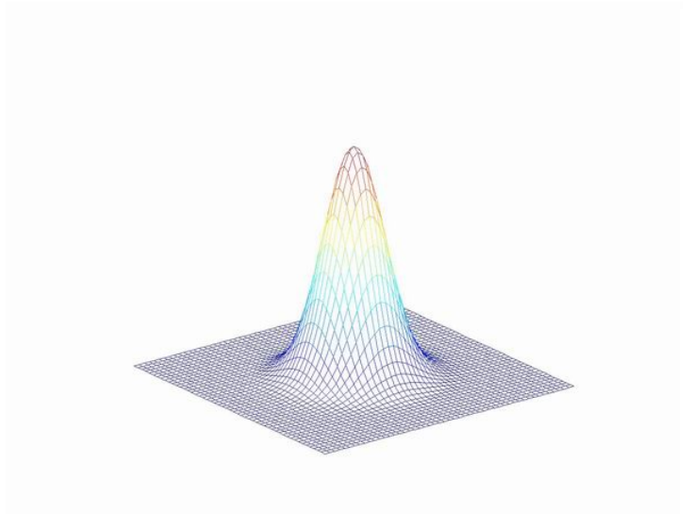
$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Aside: laplacian is equivalent to the divergence (“source-ness” or “sink”-ness) of a gradient of a function (used in fluid mechanics)

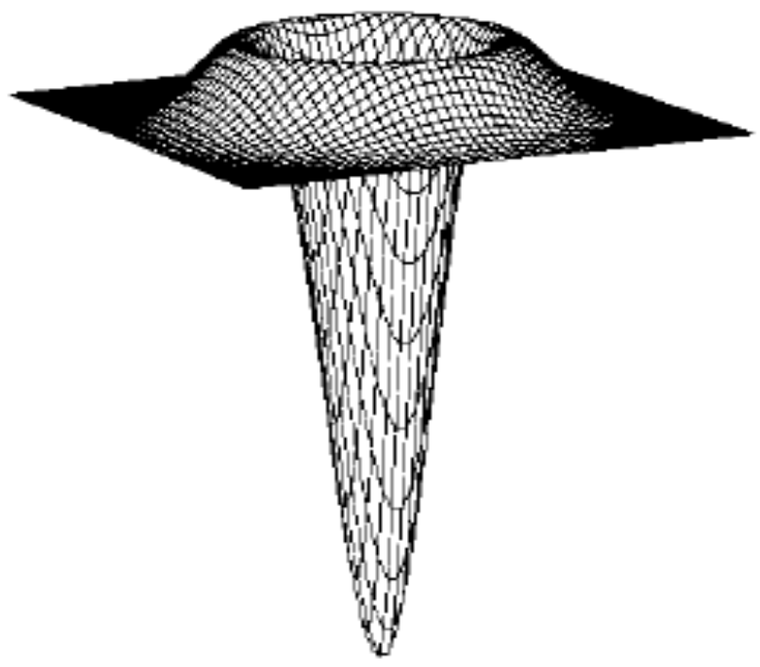
Laplacian of Gaussian (LOG)

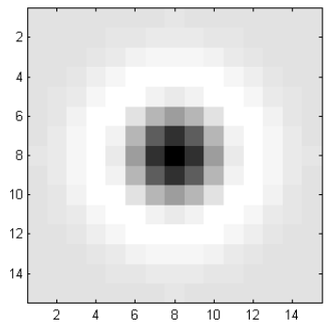
$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Gaussian



Laplacian of Gaussian





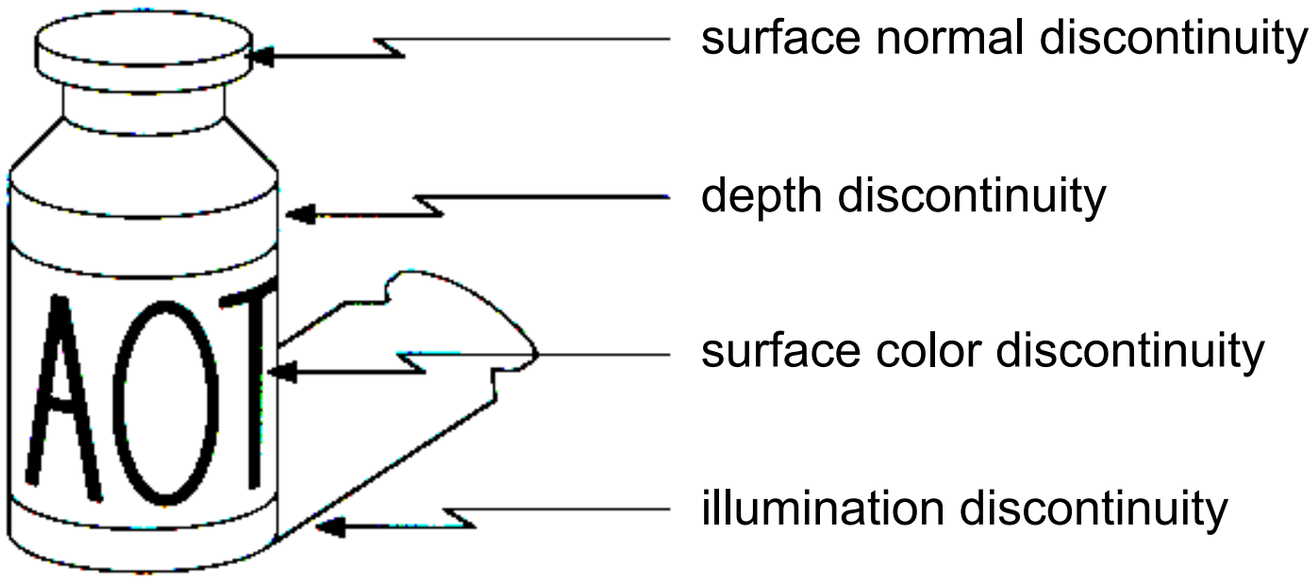
Marr & Hildreth

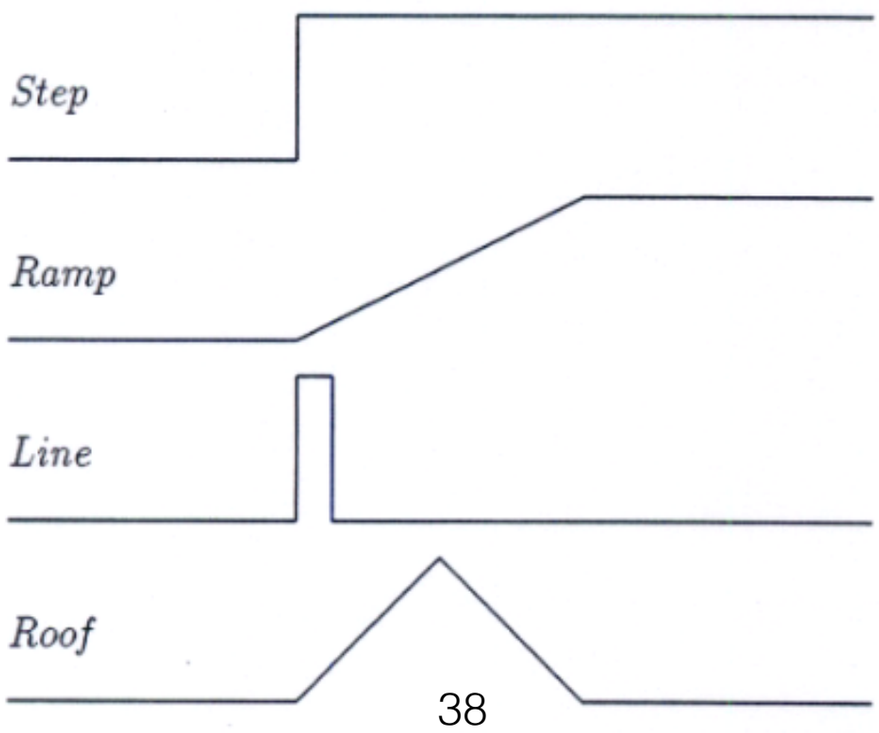
Though we've derived this to be an edge detector (by looking for where responses are 0), one could also use this as a dark-blob detector (by looking for where responses are high)

Outline

- Convolution
  - Linear Shift Invariance (LSI)
  - Convolution Properties (commutative, associative, distributive)
  - Normalize Cross-Correlation
- Edges
  - Canny edges (NMS, hysterisis)
  - derivative-of-gaussians
  - laplacian-of-Gaussians (LoG)
  - **filter banks**
- Efficiency
  - pyramids
  - linear approximiations (SVD, separability)
  - steerability

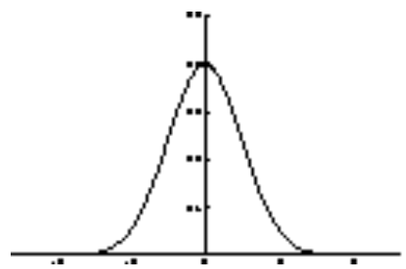
Other local features





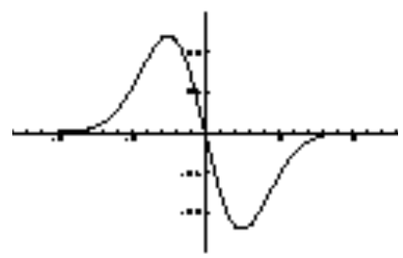
Other filters: Gaussian derivatives

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

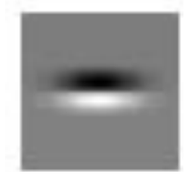


Gaussian

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

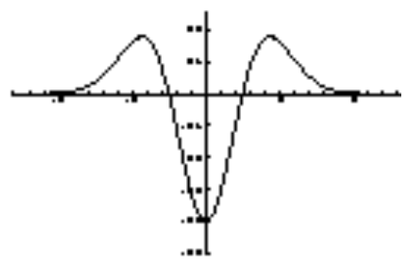


First derivative
"edges"

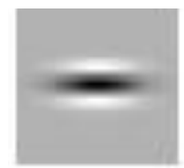


$$G_2(x) = \frac{1}{\sigma^2}\left(z^2 - 1\right) G_0$$

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



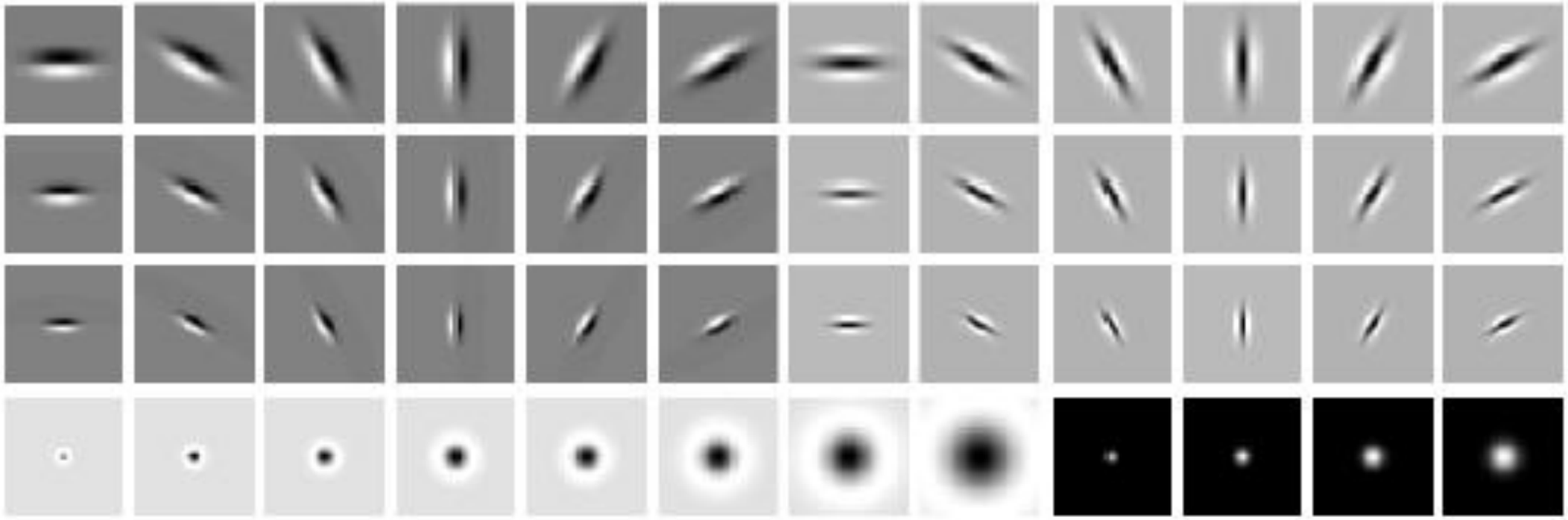
Second derivative
"bars"



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

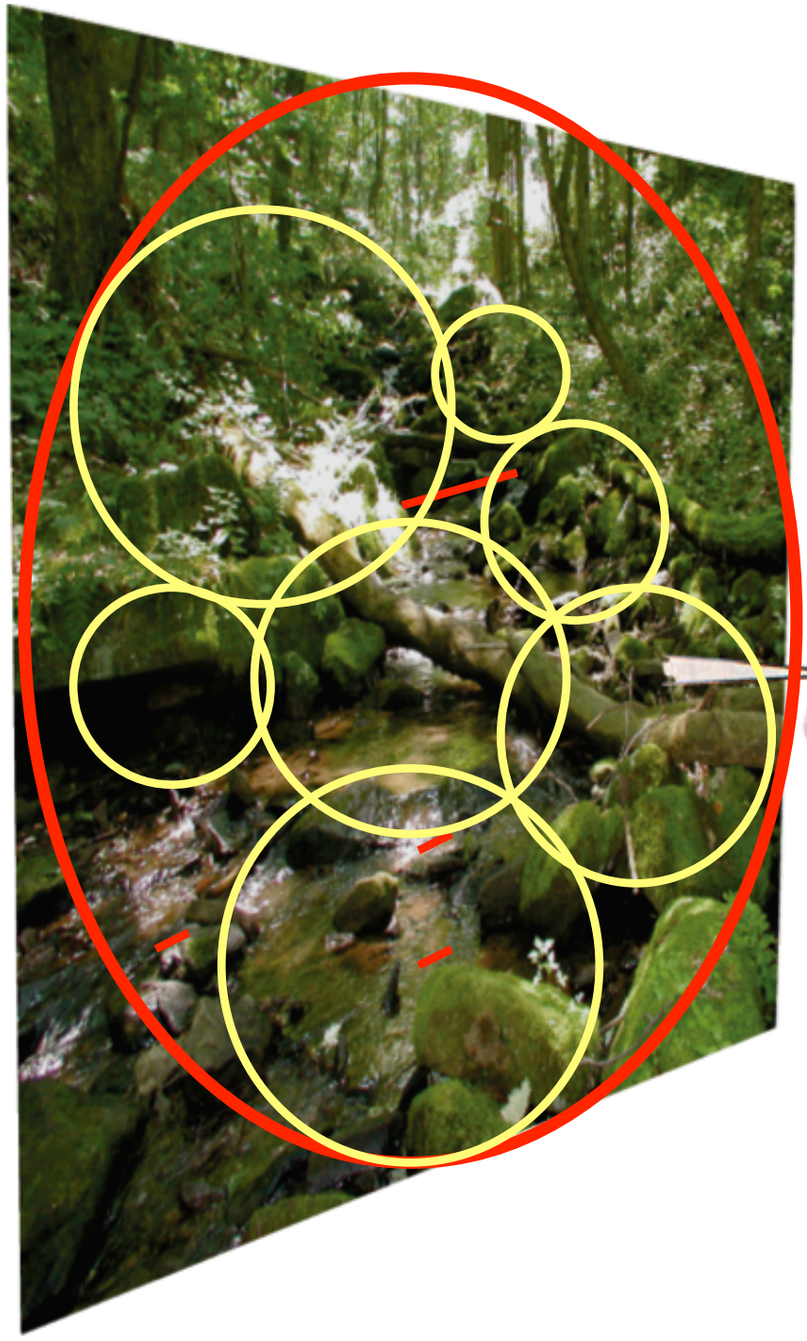
What if I want to find skinny horizontal bars or "fat" 45 degree bars…

Filter banks

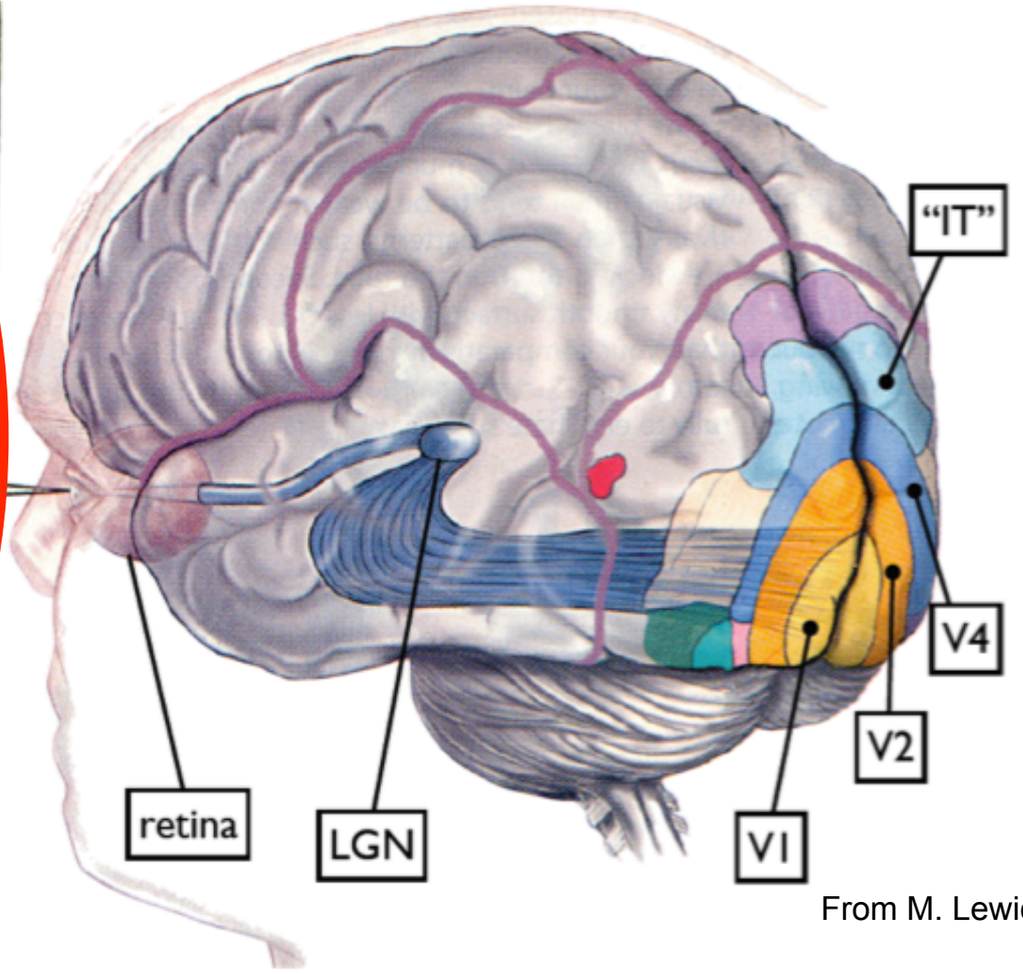


The LeungMalik filter bank has a mix of edge, bar and spot filters at multiple scales and orientations. It has a total of 48 filters - 2 Gaussian derivative filters at 6 orientations and 3 scales, 8 Laplacian of Gaussian filters and 4 Gaussian filters.

Biological motivation



Some visual areas…



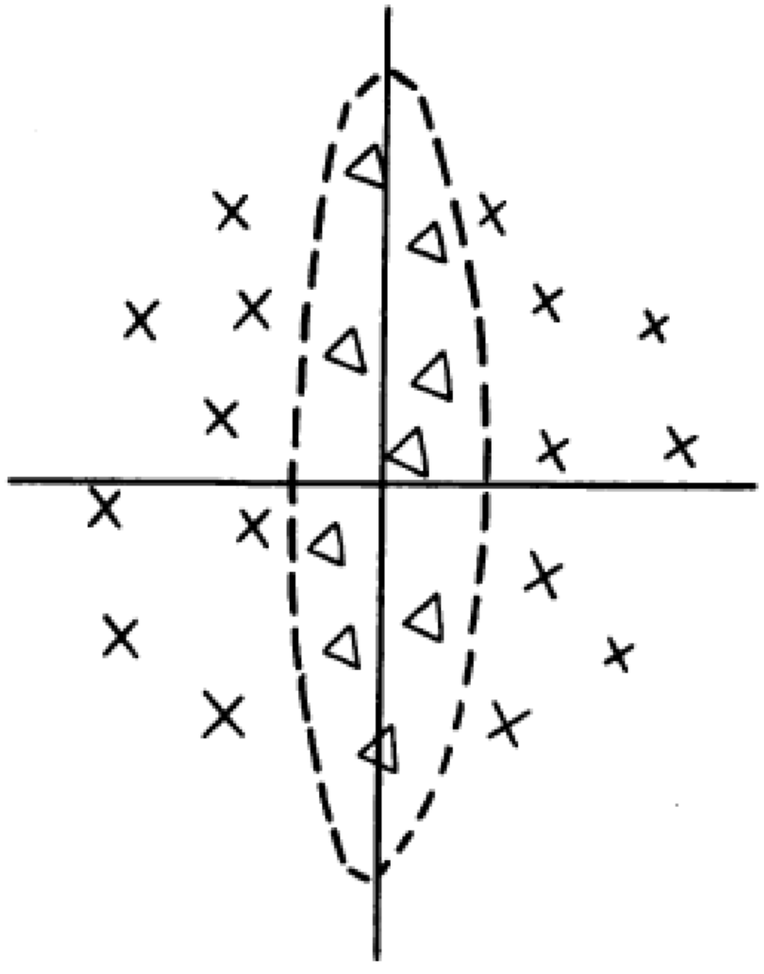
From M. Lewicky

J. Physiol. (1959) 148, 574–591

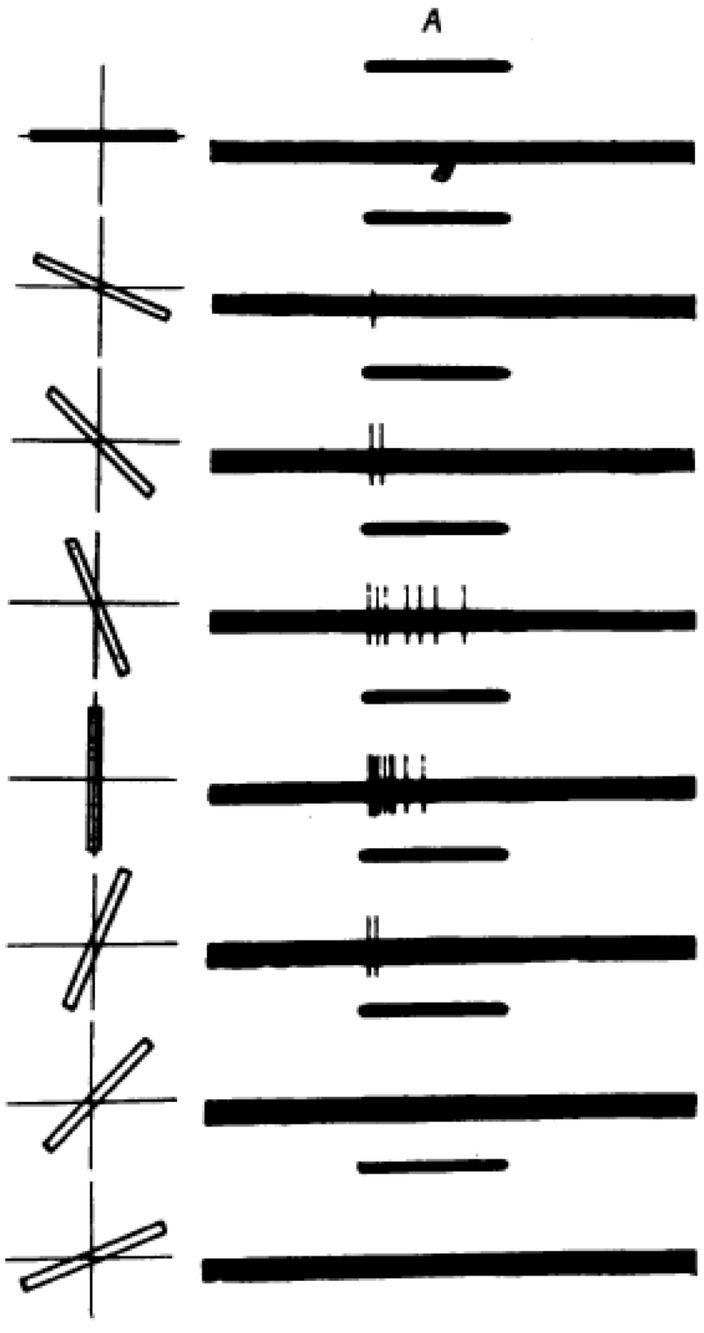
RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

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

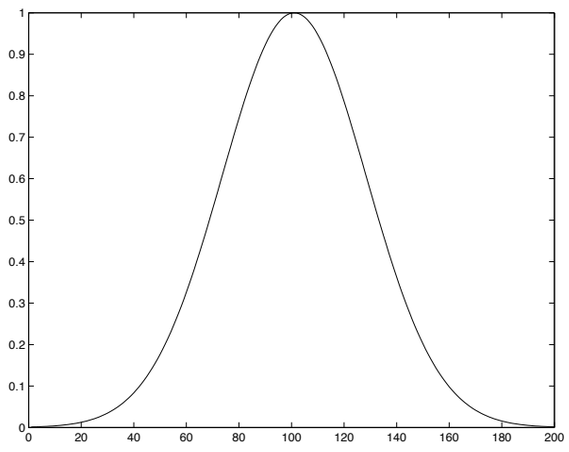


Responses to an oriented bar

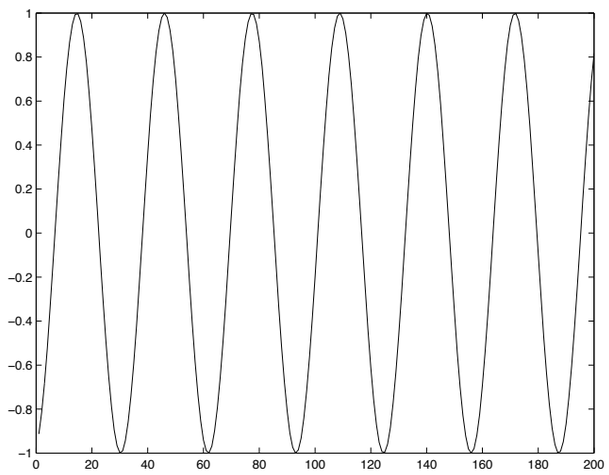
Other filters: gabors

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

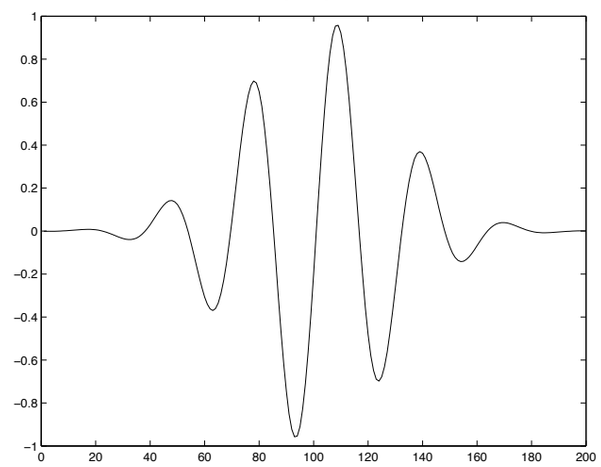
$$w(x) = \cos(2\pi f x)$$



x



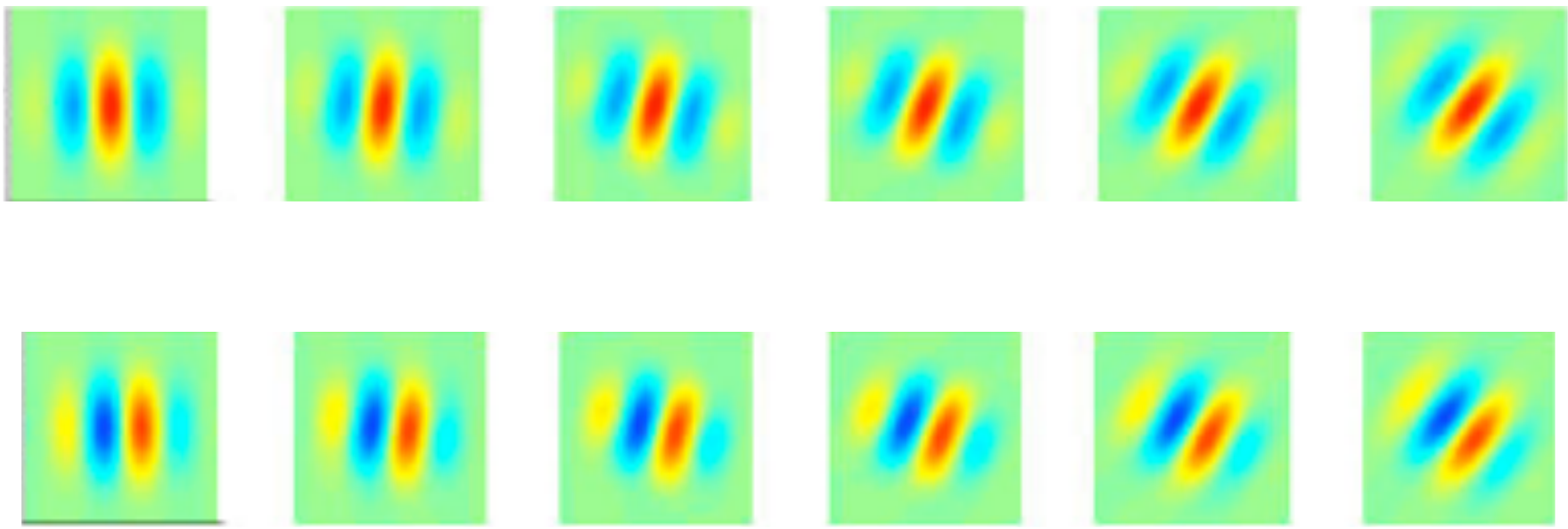
=



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

$\lambda = \frac{1}{f}$ 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; in 2 lectures)

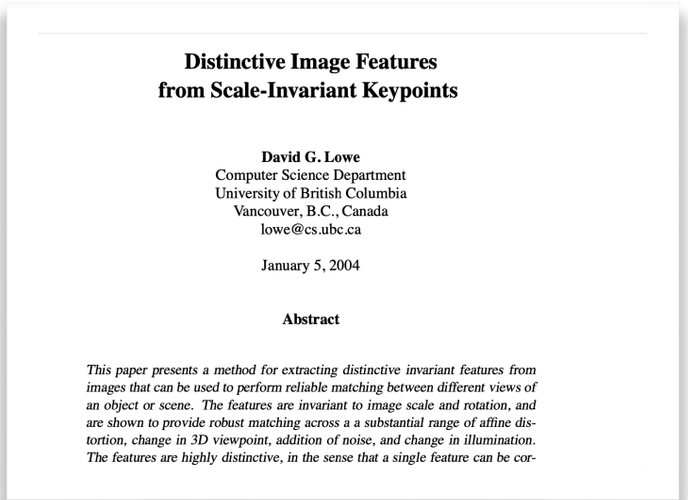
http://en.wikipedia.org/wiki/Gabor_filter

Outline

- Convolution
  - Linear Shift Invariance (LSI)
  - Convolution Properties (commutative, associative, distributive)
  - Normalize Cross-Correlation
- Edges
  - Canny edges (hysterisis)
  - derivative-of-gaussians (DoG)
  - laplacian-of-Gaussians (LoG)
  - filter banks
- **[Interlude] Histograms of oriented gradientas (HOG)**
- Efficiency
  - pyramids
  - linear approximiations (SVD, separability)
  - steerability

Background: SIFT image patch descriptor

73K citations.
Single-handedly raised the impact factor of the International Journal of Computer Vision (IJCV)



Distinctive Image Features from Scale-Invariant Keypoints

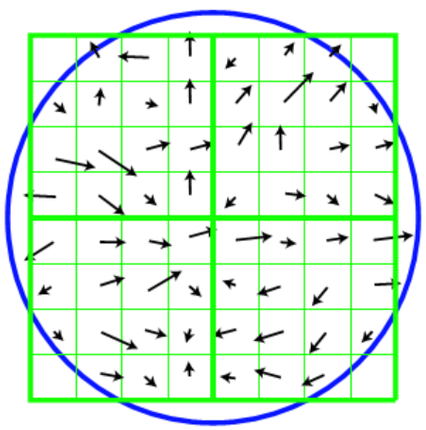
David G. Lowe
Computer Science Department
University of British Columbia
Vancouver, B.C., Canada
lowe@cs.ubc.ca

January 5, 2004

Abstract

This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be cor-

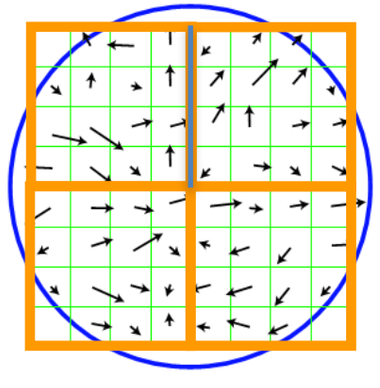
Part 1: finding interesting patches
(we'll ignore for now, but revisit in Correspondence lectures)

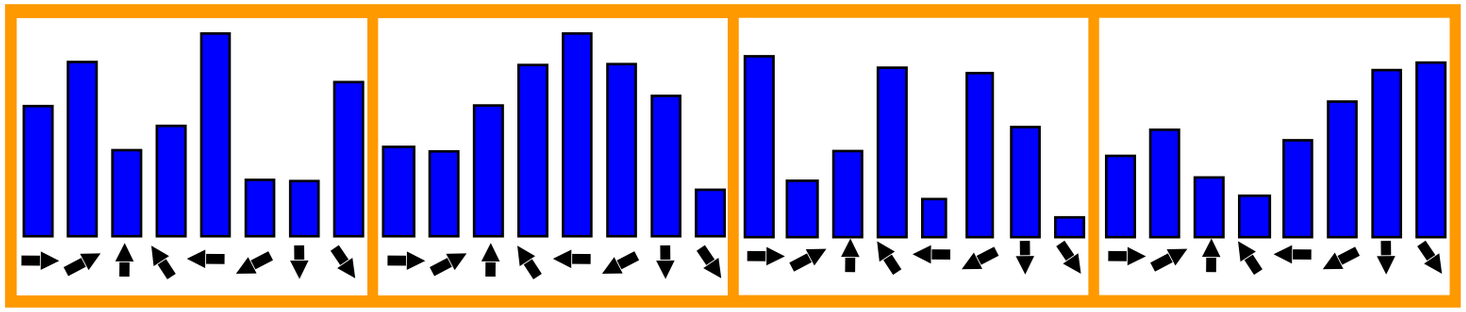


Part 2: constructing a descriptor for an image patch
(that is more invariant than raw pixels)

Background: SIFT image patch descriptor

Histograms of gradient directions over spatial cells





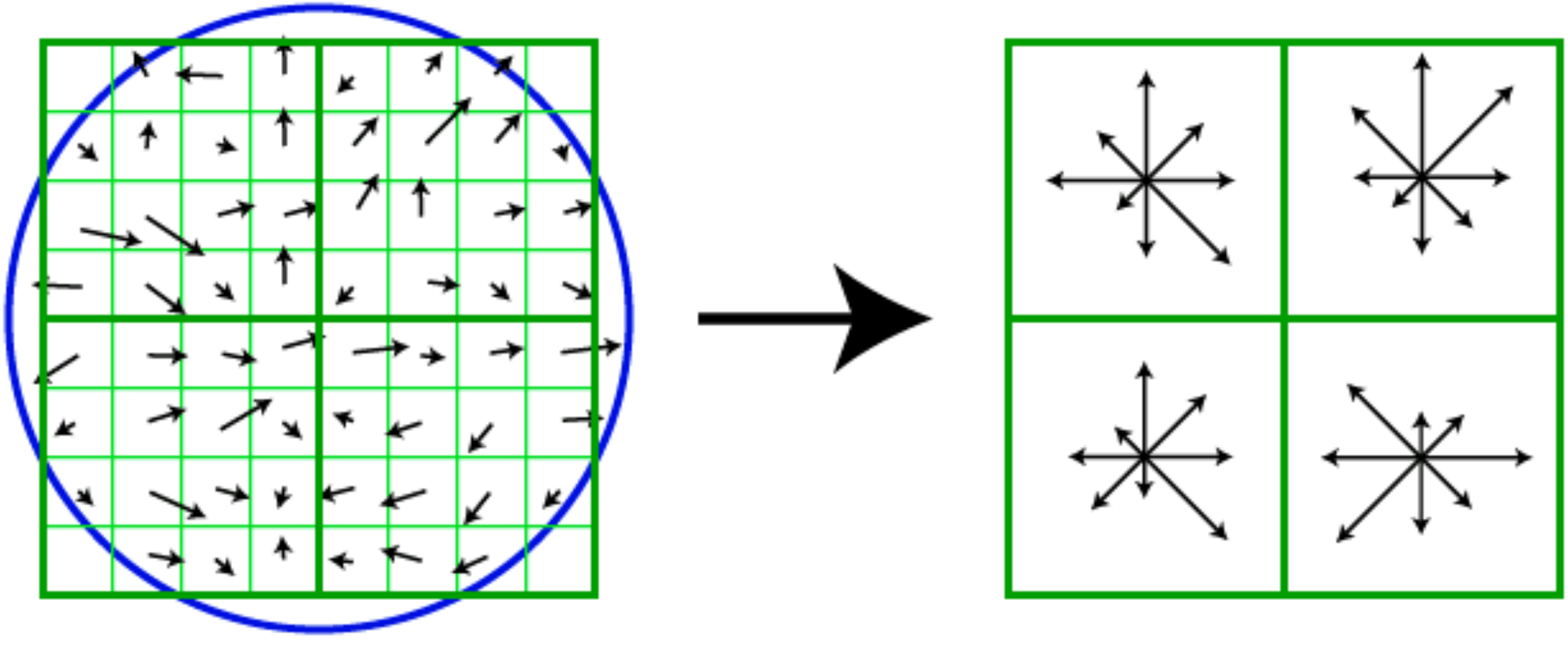
1. Compute gradient vector at each pixel (with [-1 0 1] and $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ filters)
2. Snap gradient vector to one of 8 orientations (North, West, Northwest,…)
3. Split up patch into 2x2 quadrants and count the number of each (North, West, Northwest) orientation

SIFT Descriptor - Details

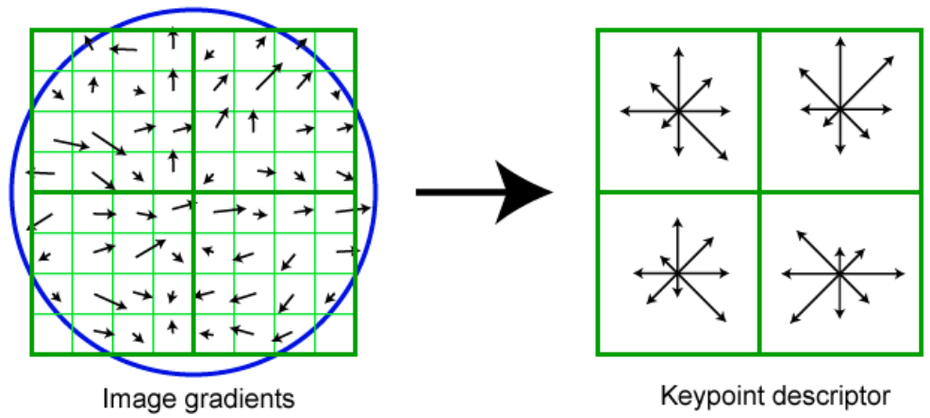
Assume that we bin “o” orientations over “kxk” cells, where each cell is “sxs” pixels

What’s final dimensionality of descriptor?

Common visualizations: o=8, k=2, s=4 => 64 dim
Common implementations: o=8, k=4, s=4 => 128 dim
you’ll experiment with this!



Post-processing



1. Rescale 128-dim vector to have unit norm

$$x = \frac{x}{||x||}, \quad x \in R^{128}$$

“invariant to linear scalings of intensity”

2. Clip high values

$$x := \min(x, .2)$$

$$x := \frac{x}{||x||}$$

approximate binarization allows for for flat patches with small gradients to remain stable

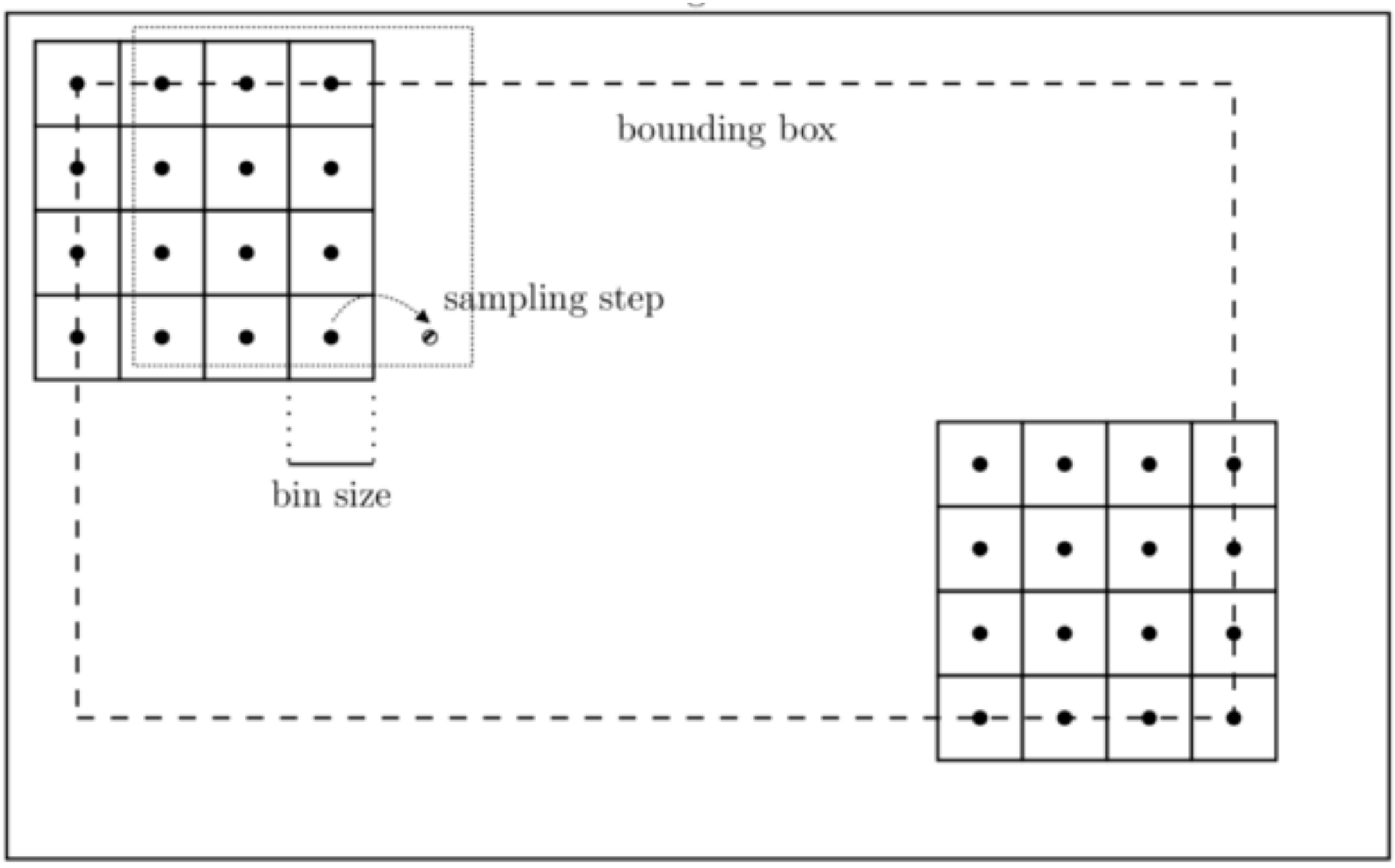
Histograms of Oriented Gradients (45K citations)

Compute 128-dim SIFT descriptors *blocks* densely across image

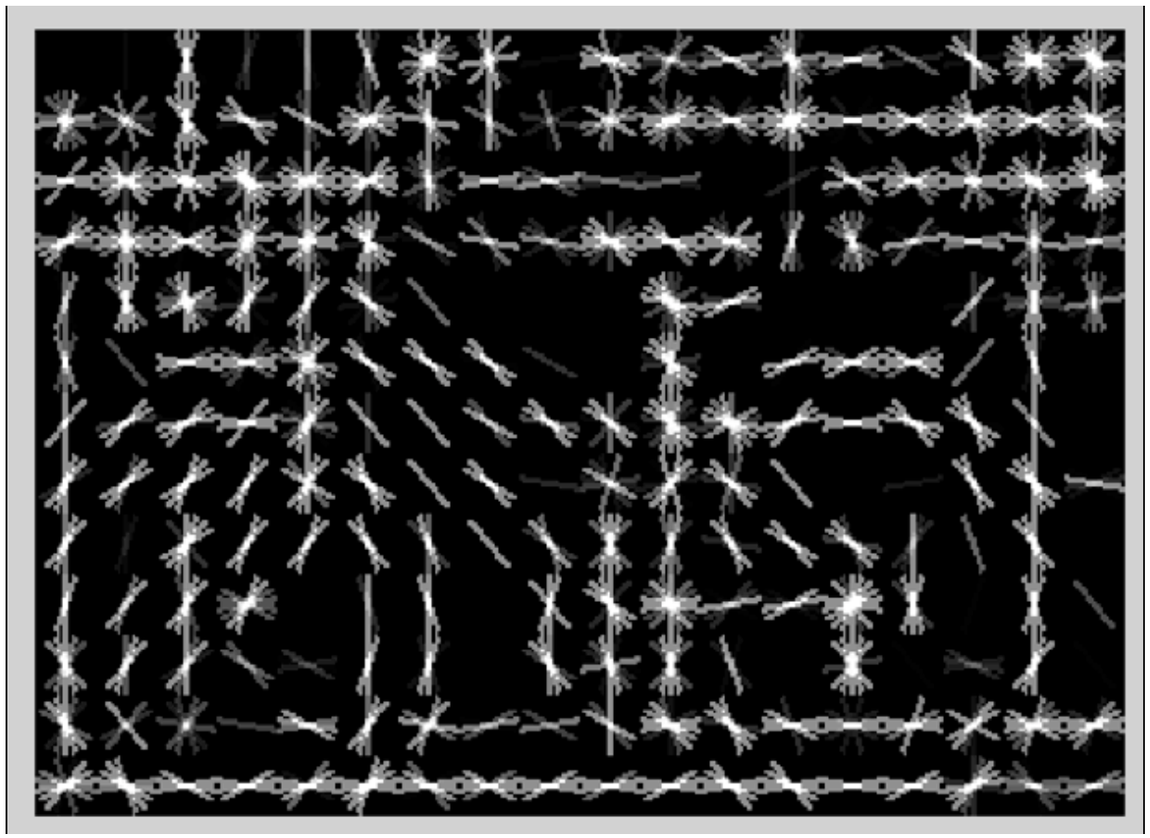
Naive implementation: loop over all possible matches and compute for each
Insight: Compute descriptor blocks at strides of a cell (e.g., 4-pixel shifts).
This allows you to compute gradients *once* on whole image and compute histograms of gradient orientations *once*.

What is output size of compute_gradients() and bin_gradients()?

[My own implementation still iterators over all block positions. Maybe you can find a way to avoid a double-for loop!]



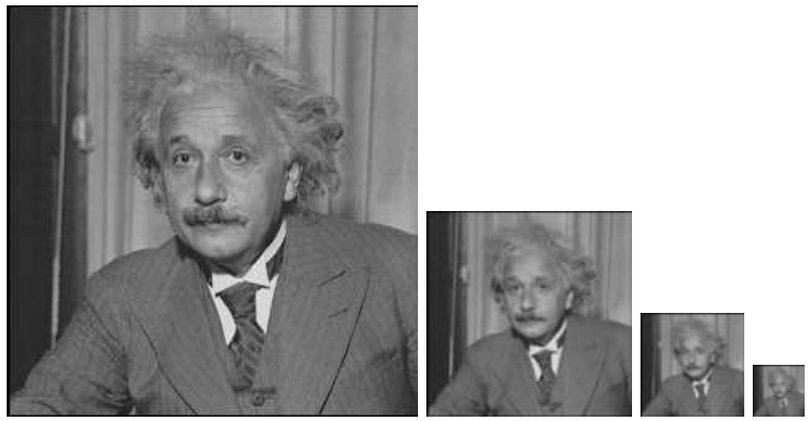
Slick visualization of bin_gradients() output



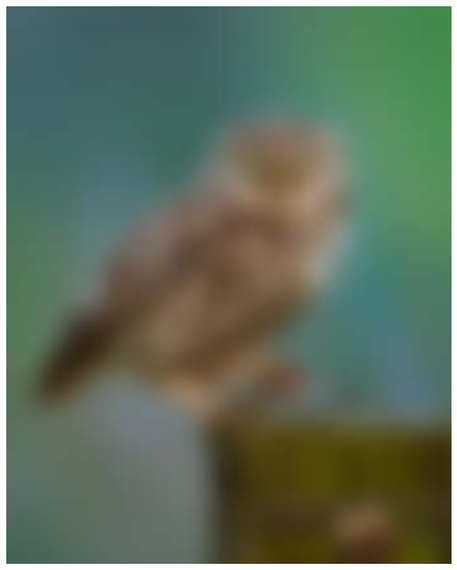
Outline

- Convolution
  - Linear Shift Invariance (LSI)
  - Convolution Properties (commutative, associative, distributive)
  - Normalize Cross-Correlation
- Edges
  - Canny edges (hysterisis)
  - derivative-of-gaussians (DoG)
  - laplacian-of-Gaussians (LoG)
  - filter banks
- **Efficiency**
  - pyramids
  - linear approximiations (SVD, separability)
  - steerability

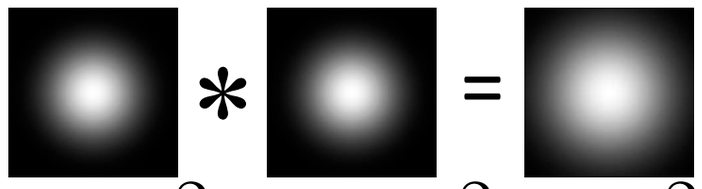
Pyramids



- 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

The Laplacian Pyramid as a Compact Image Code

PETER J. BURT, MEMBER, IEEE, AND EDWARD H. ADELSON

Abstract—**We describe a technique for image encoding in which local operators of many scales but identical shape serve as the basis functions. The representation differs from established techniques in that the code elements are localized in spatial frequency as well as in space.**

Pixel-to-pixel correlations are first removed by subtracting a low-pass filtered copy of the image from the image itself. The result is a net data compression since the difference, or error, image has low variance and entropy, and the low-pass filtered image may represented at reduced sample density. Further data compression is achieved by quantizing the difference image. These steps are then repeated to compress the low-pass image. Iteration of the process at appropriately expanded scales generates a pyramid data structure.

The encoding process is equivalent to sampling the image with Laplacian operators of many scales. Thus, the code tends to enhance salient image features. A further advantage of the present code is that it is well suited for many image analysis tasks as well as for image compression. Fast algorithms are described for coding and decoding.

does not permit simple sequential coding. Noncausal approaches to image coding typically involve image transforms, or the solution to large sets of simultaneous equations. Rather than encoding pixels sequentially, such techniques encode them all at once, or by blocks.

Both predictive and transform techniques have advantages. The former is relatively simple to implement and is readily adapted to local image characteristics. The latter generally provides greater data compression, but at the expense of considerably greater computation.

Here we shall describe a new technique for removing image correlation which combines features of predictive and transform methods. The technique is noncausal, yet computations are relatively simple and local.

The predicted value for each pixel is computed as a local

Peter J. Burt (M'80) received the B.A degeree i physics from Harvard University, Cambridge, MA in 1968, and the M.S. and Ph.D. degrees i computer science from the University of Massa chusetts, Amherst, in 1974 and 1976, respec-tively

From 1968 to 1972 he conducted research i sonar, particularly in acoustic imaging devices a the USN Underwater Sound Laboratory, Nev London, CT and in London, England. As Postdoctoral Fellow, he has studied both natural vision and computer imag understanding at New York University, New York, NY (1976-1978), Be Laboratories (1978-1979), and the University of Maryland, College Par (1979-1989). He has been a member of the faculty at Rensselae Polytechnic Institute, Troy, NY, since 1980.

Edward H. Adelson received B.A. degree in physics and philosophy from Yale University, New Haven, CT, in 1974, and the Ph.D degree in experimental psychology from the University of Michigan, Ann Arbor, in 1979.

From 1979 to 1981 he was Postdoctoral Fellow at New York University, New York, NY. Since 1981, he has been at RCA David Sarnoff Research Center, Princeton, NJ, as a member of the Technical Staff in the Image Quality and Human Perception Research Group. His research interests center on visual processes in both human and machine visual systems, and include psychophysics, image processing, and artificial intelligence.

Dr. Adelson is a member of the Optical Society of America, the Association for Research in Vision and Opthalmology, and Phi Beta Kappa.

Key idea: rather than convolving with big gaussian

Convolve with small gaussian, subsample, convolve with small gaussian, subample,….

Can we make even more efficient (since we know we will be subsampling every-other-row-and-column the output)?

yes - perform *strided* convolution that shifts filter by 2 pixels instead of 1

Burt & Adelson, 83

Implementation in 1D

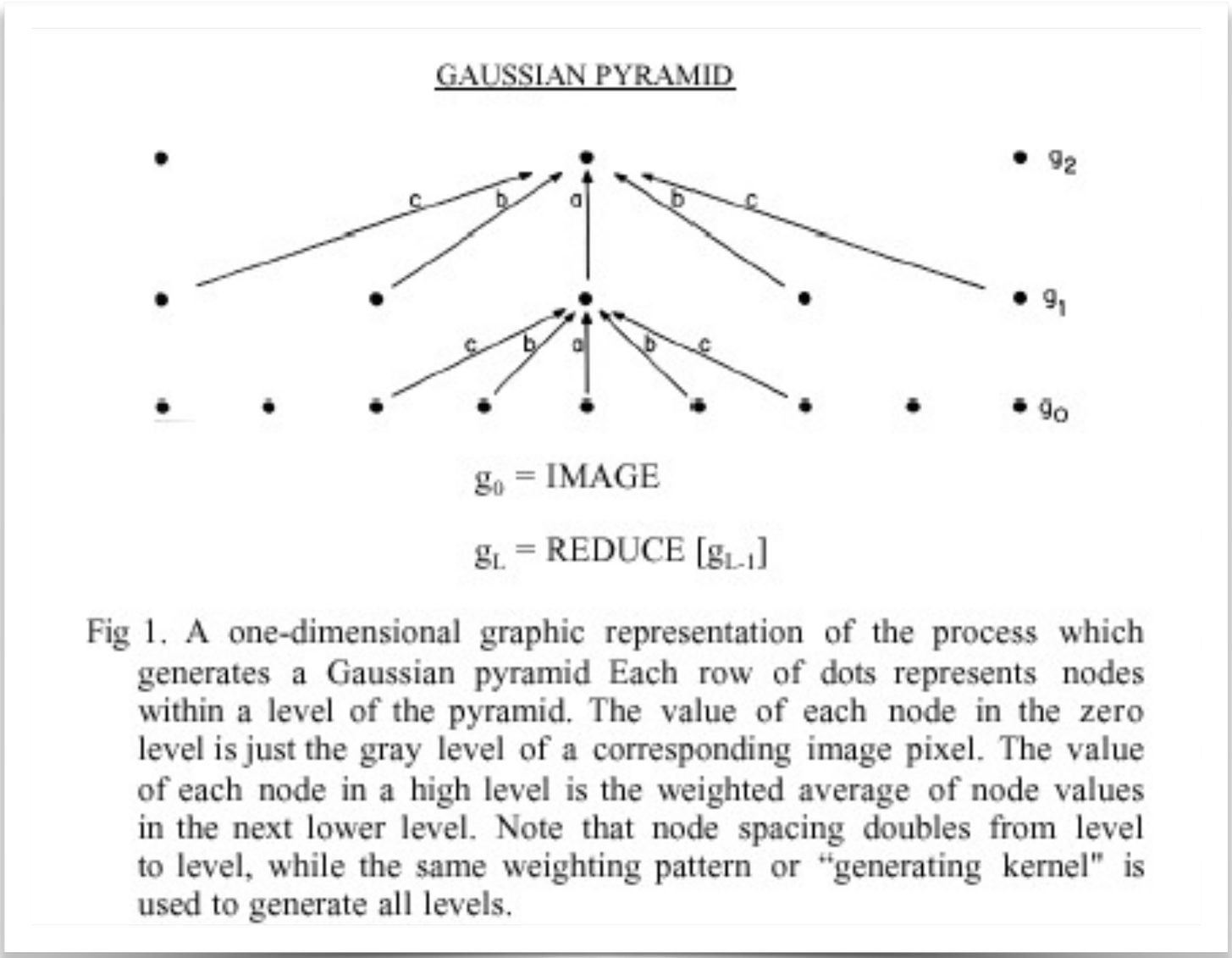
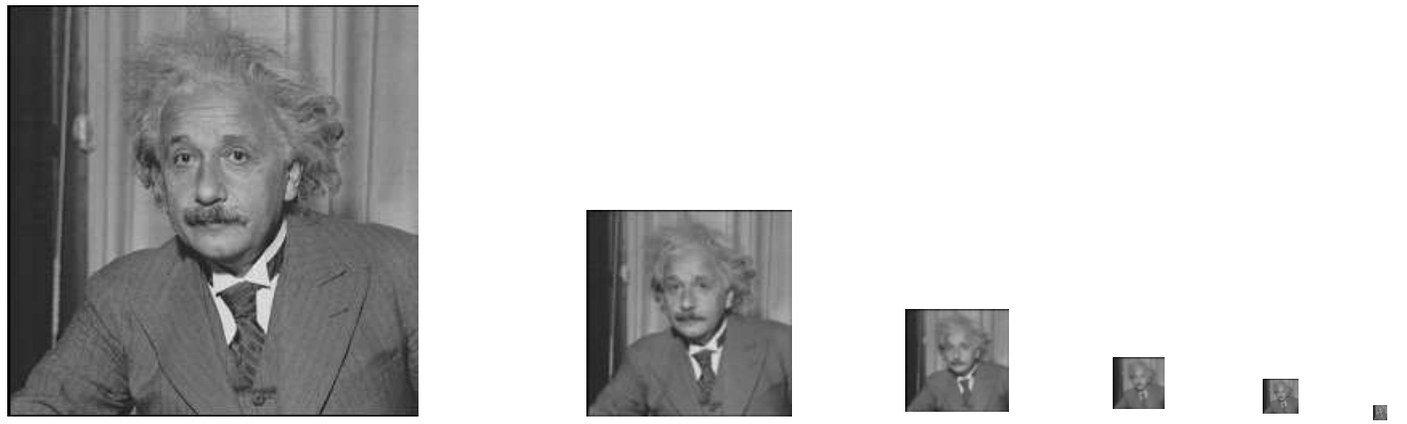


Fig 1. A one-dimensional graphic representation of the process which generates a Gaussian pyramid Each row of dots represents nodes within a level of the pyramid. The value of each node in the zero level is just the gray level of a corresponding image pixel. The value of each node in a high level is 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.

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



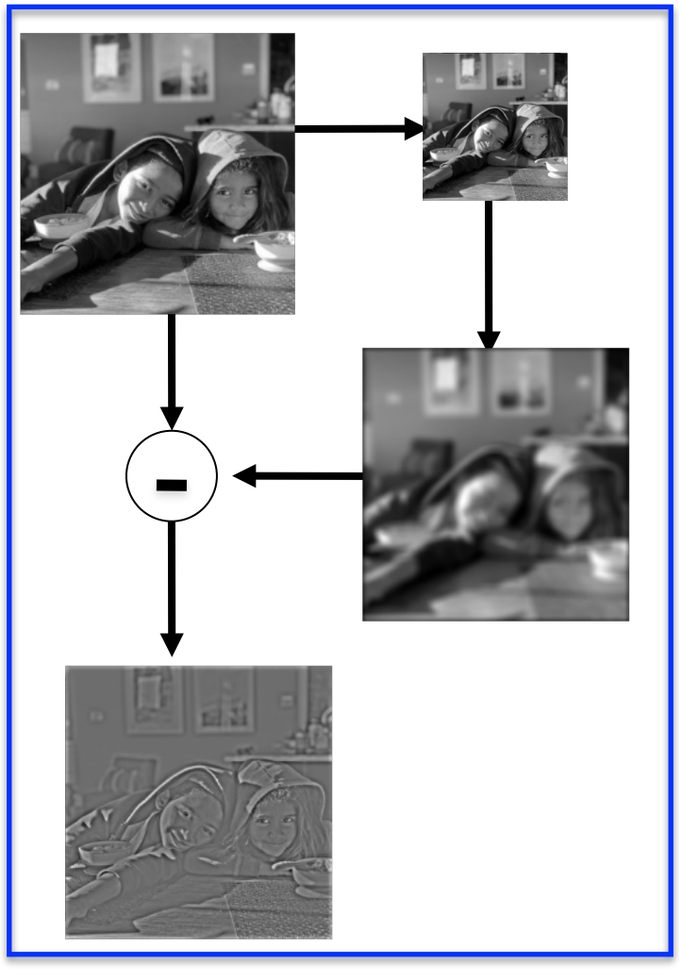
First 6 levels of a Gaussian Pyramid. Original image is 257x257; level 5 is 9x9

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

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

Laplacian pyramid

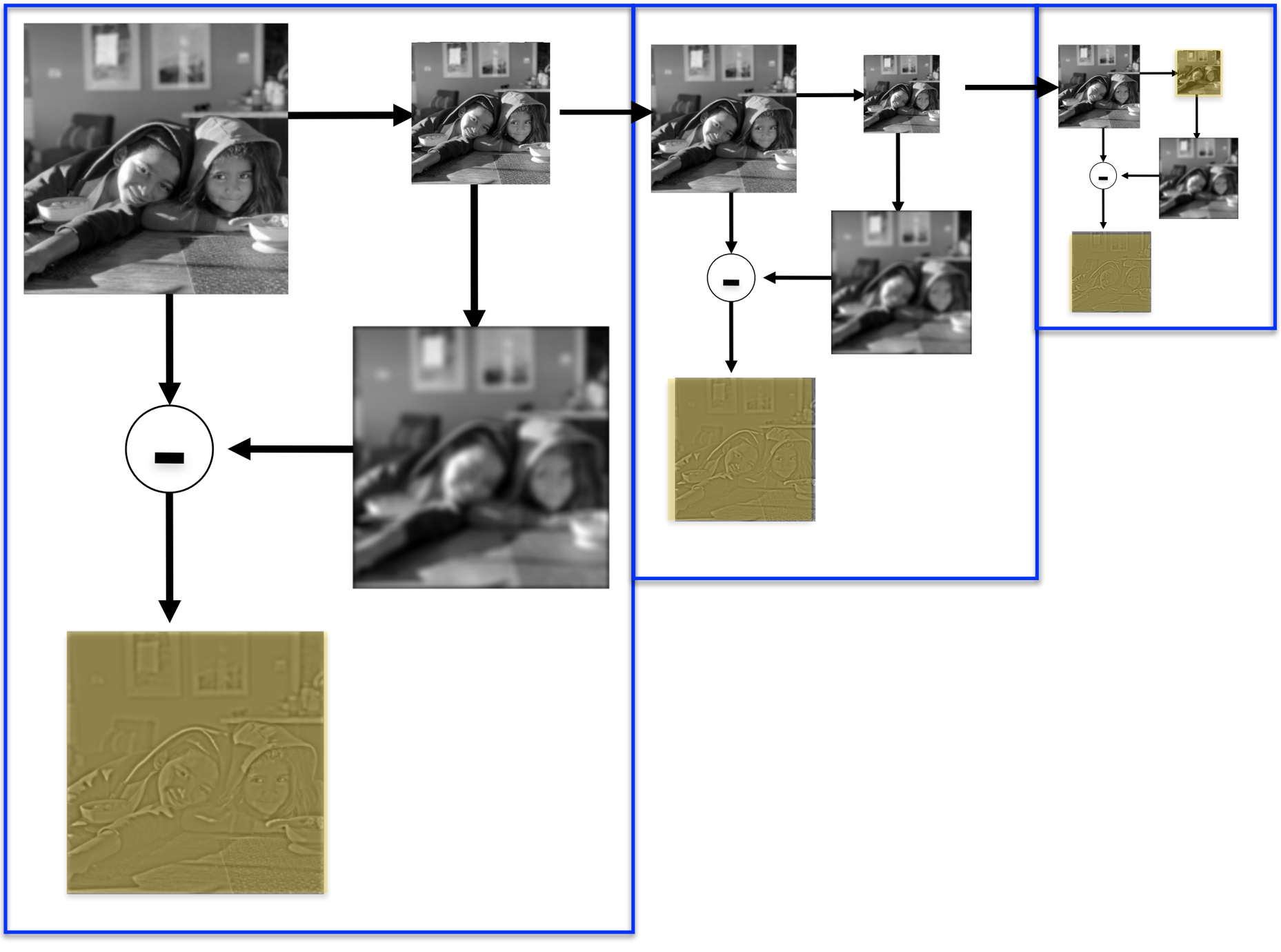
Store *difference* between upsampled image and original image



Upsampling: insert zeros between pixels and apply renormalized Gaussian filter: $H[u] = \frac{1}{16}[1 \quad 4 \quad 6 \quad 4 \quad 1]$

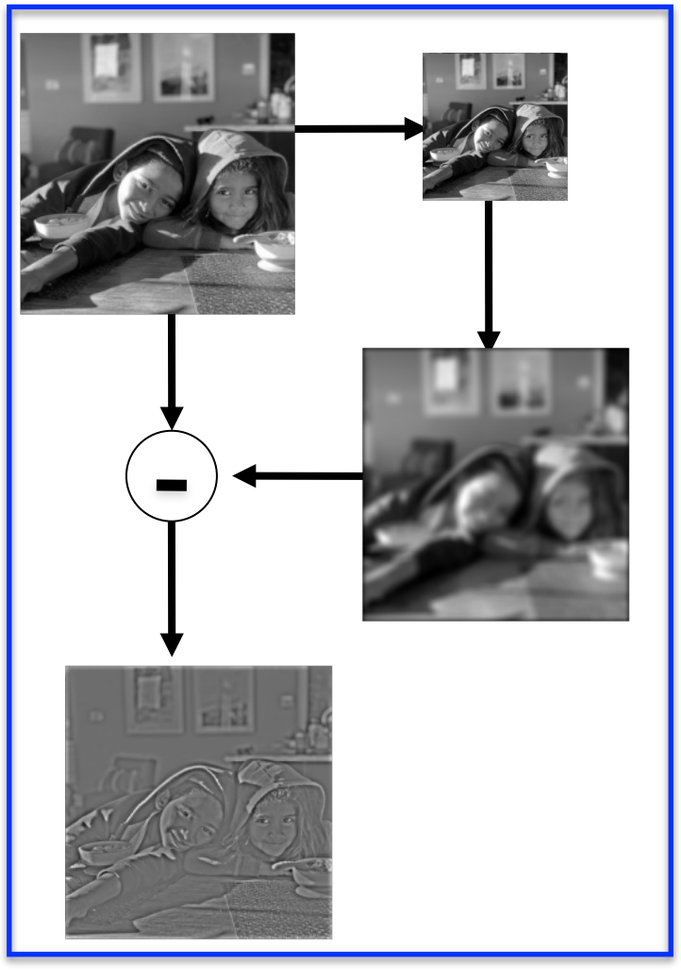
$$EXPAND(F)[i] = 2\sum_{u=-2}^{2} H[u]F[(i+u)/2]$$

Laplacian pyramid



Only need to store tiny-res image + sparse delta-images

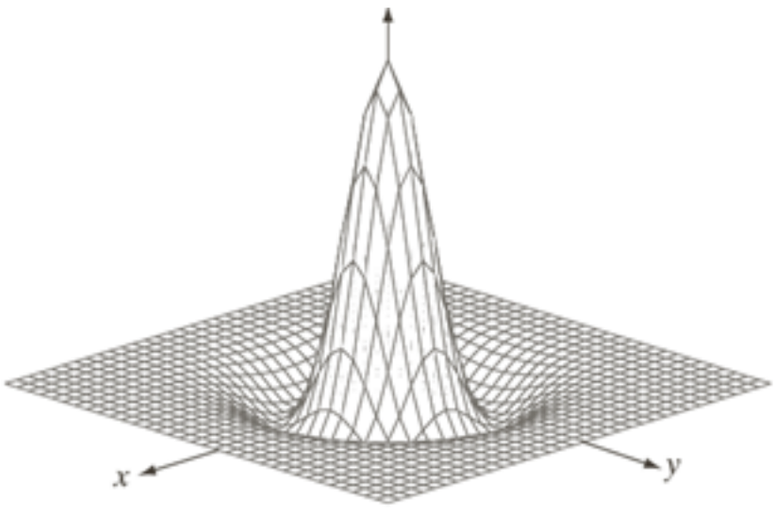
Laplacian pyramid

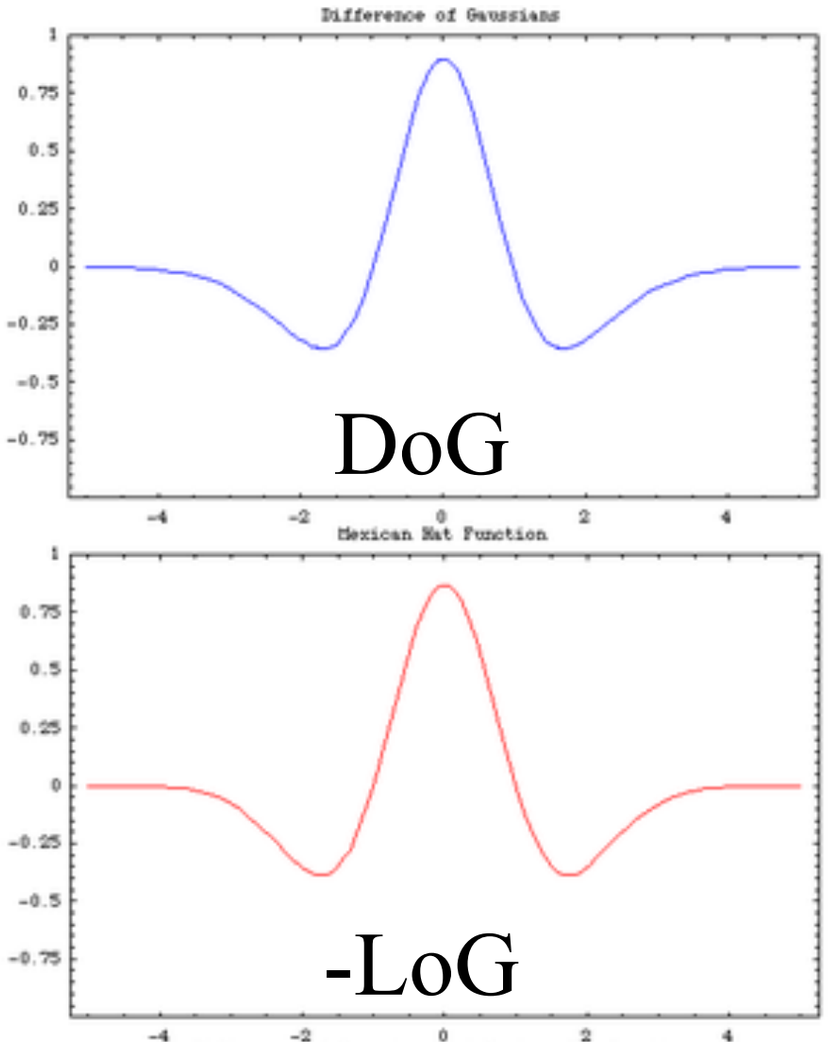


Can we directly produce the difference image with a linear filter?

Difference of Gaussian (DoG) filter

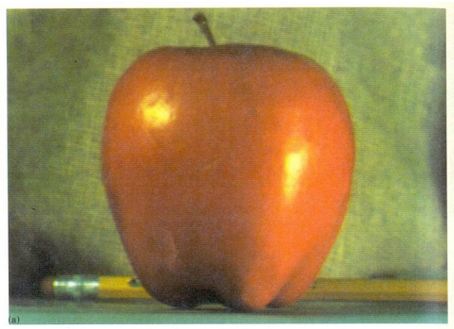
Looks very similar to a laplacian of Gaussian filter, but a different derivation!



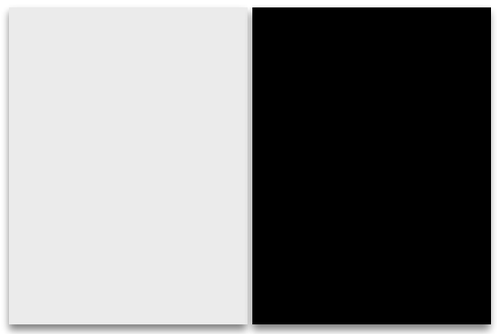


Multiresolution blending

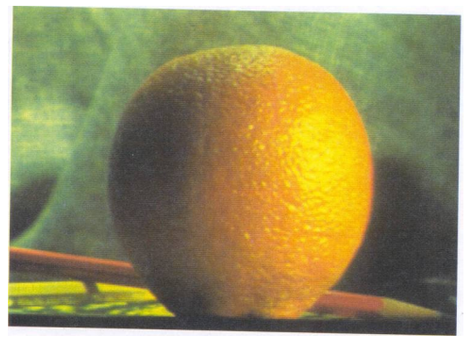
Left image

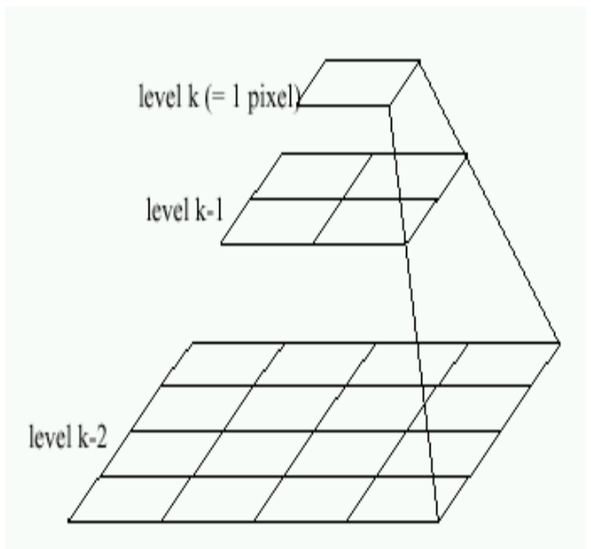


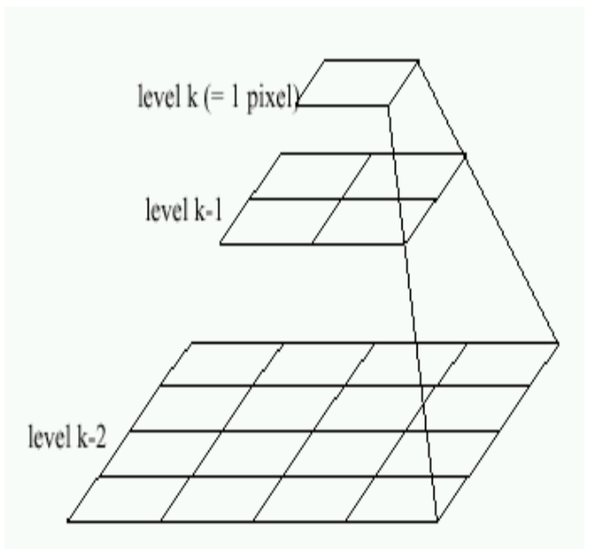
Blend mask

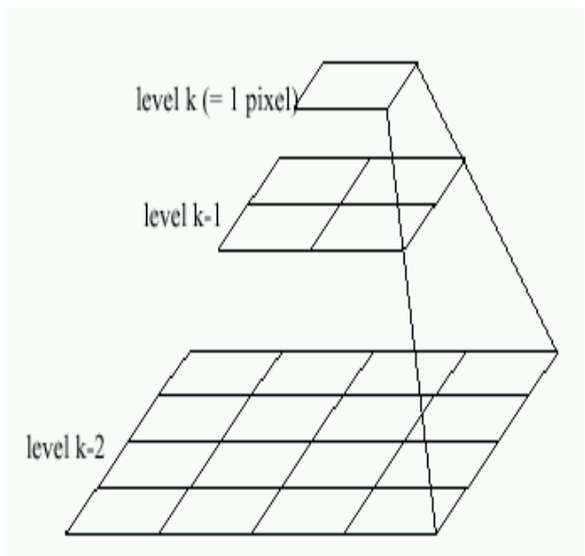


Right image





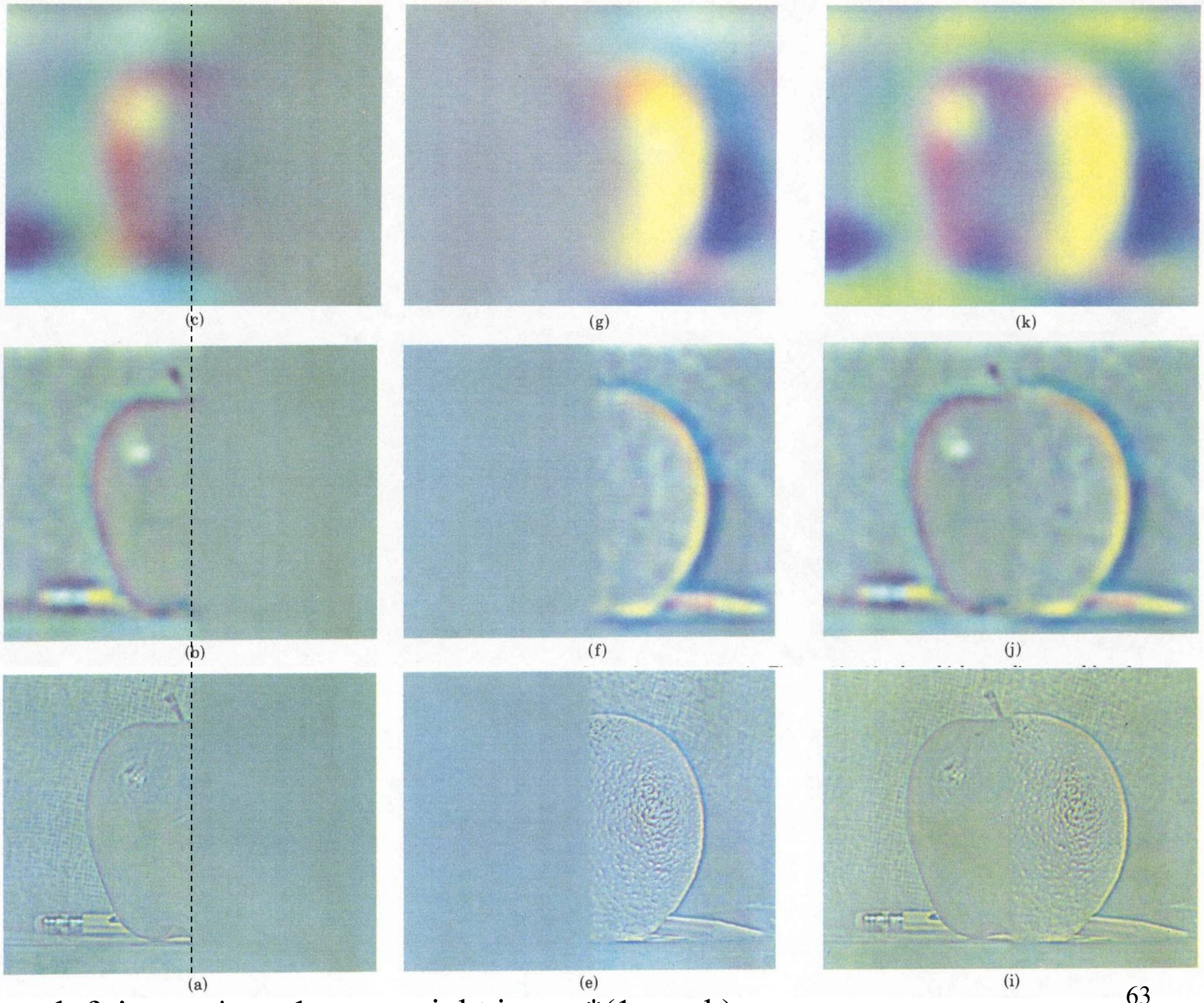




laplacian level 4

laplacian level 2

laplacian level 0

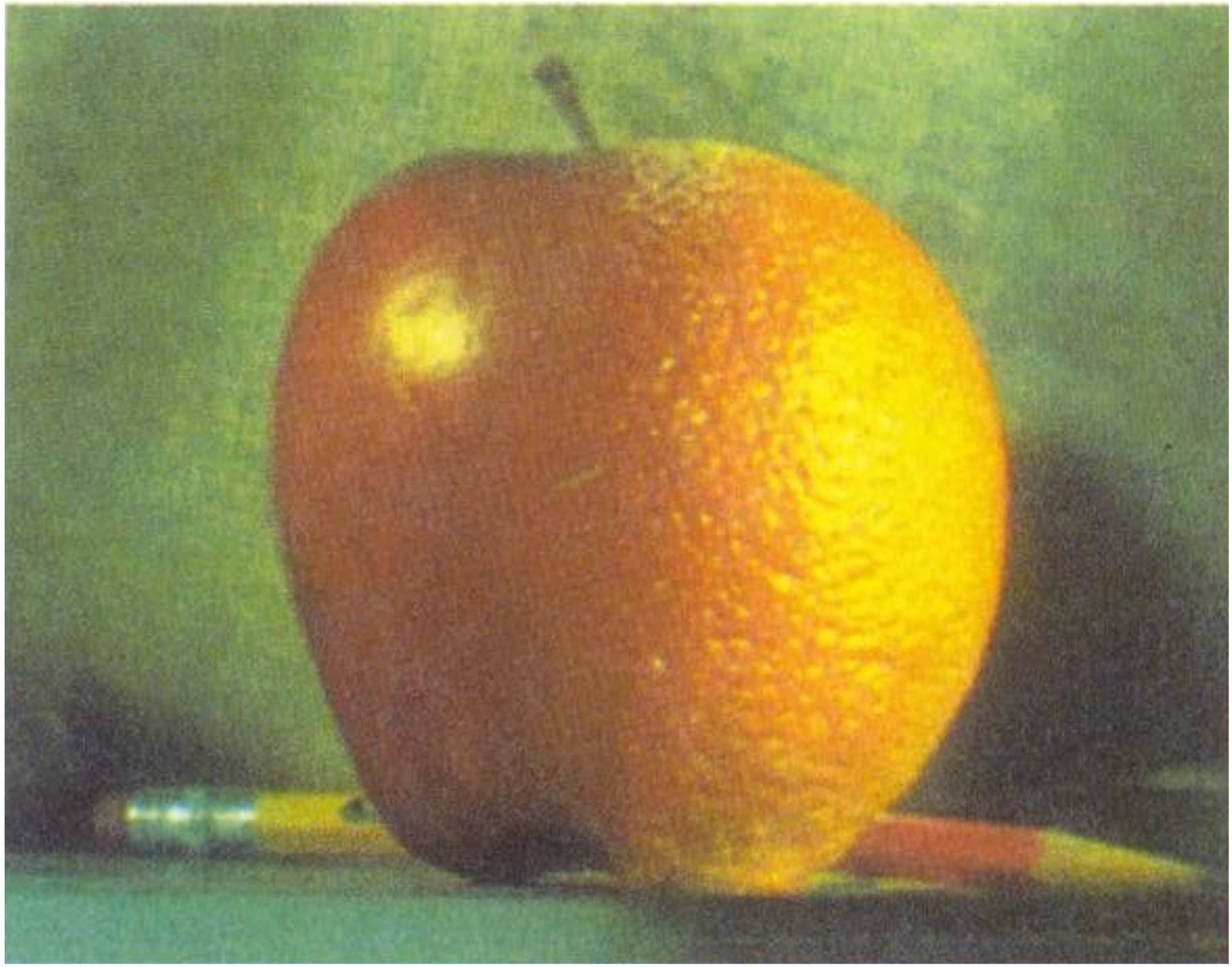


left image*mask

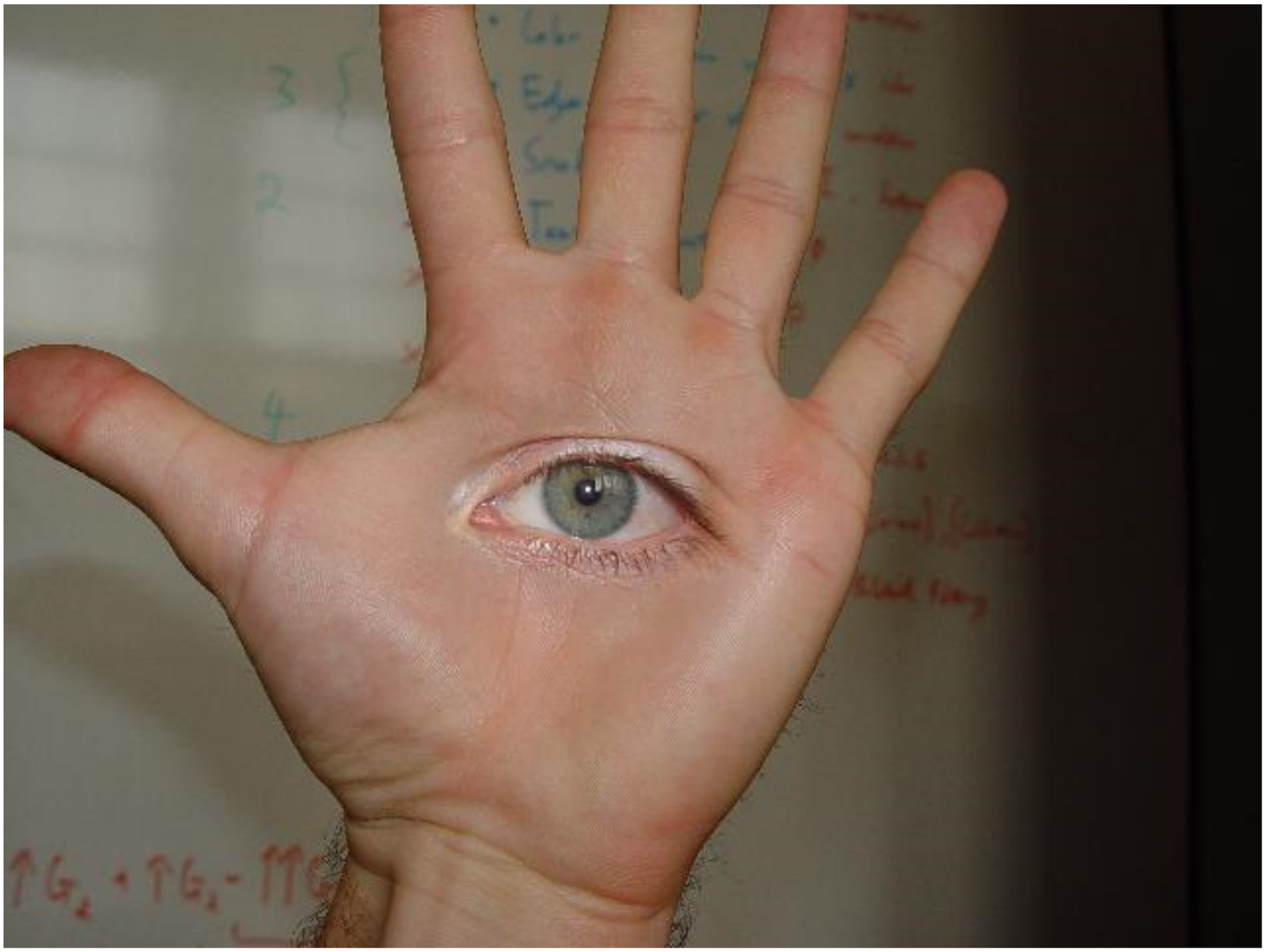
right image*(1-mask)

sum

Pyramid Blending



Fun example!

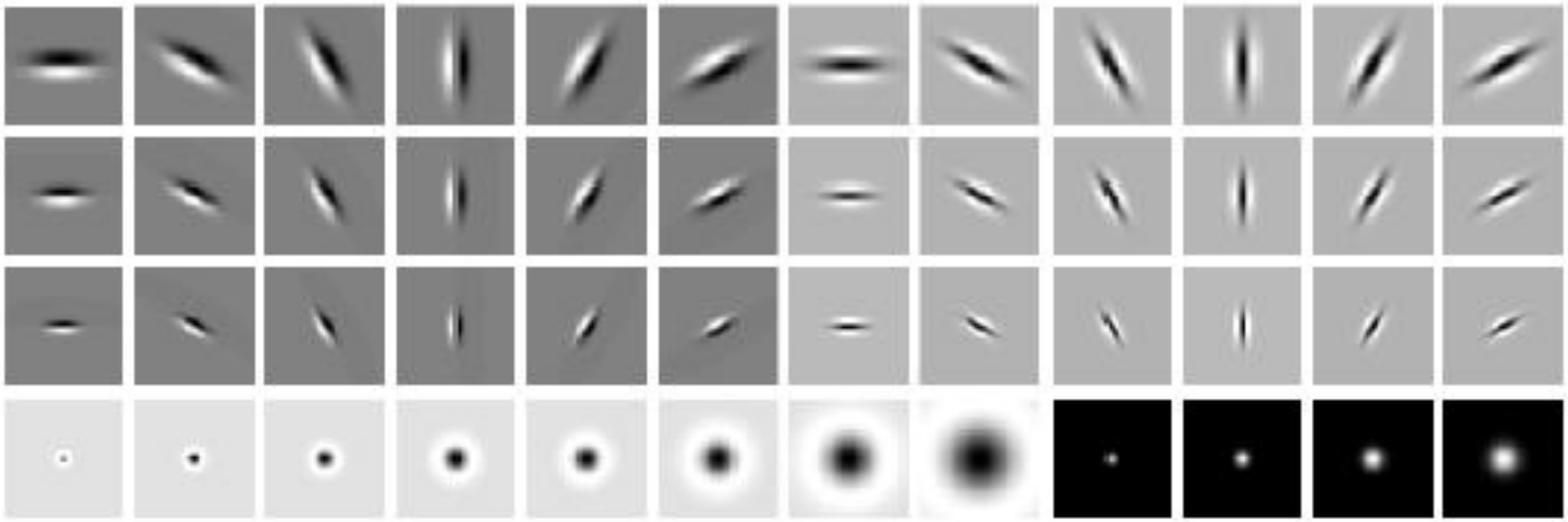


© david dmartin (Boston College)

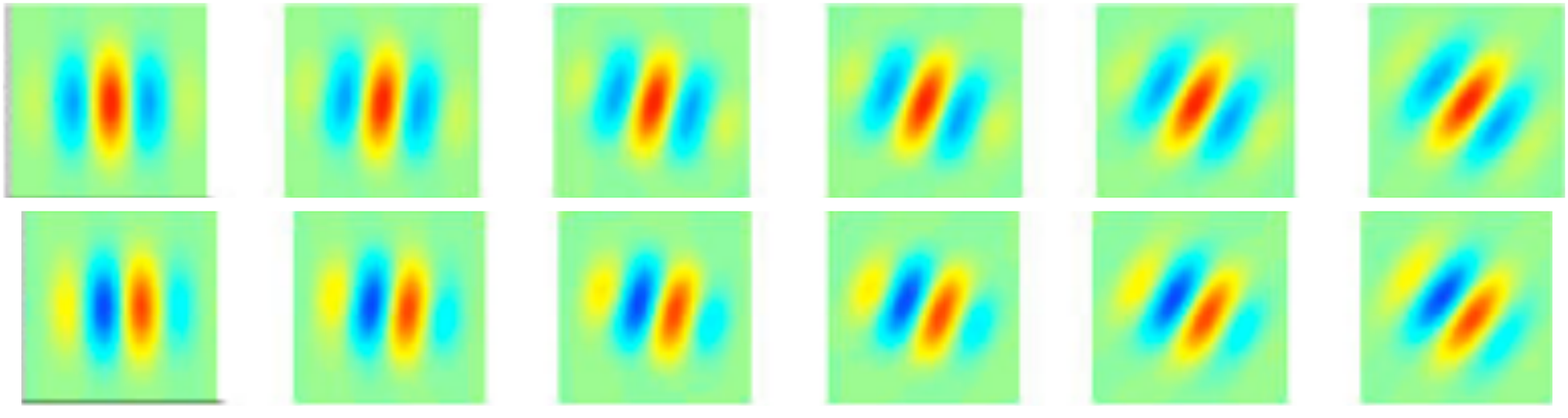
Outline

- Convolution
  - Linear Shift Invariance (LSI)
  - Convolution Properties (commutative, associative, distributive)
  - Normalize Cross-Correlation
- Edges
  - Canny edges (hysterisis)
  - derivative-of-gaussians (DoG)
  - laplacian-of-Gaussians (LoG)
  - filter banks
- Efficiency
  - pyramids
  - **steerability**
  - linear approximiations (SVD, separability)

Recall:
oriented filter banks



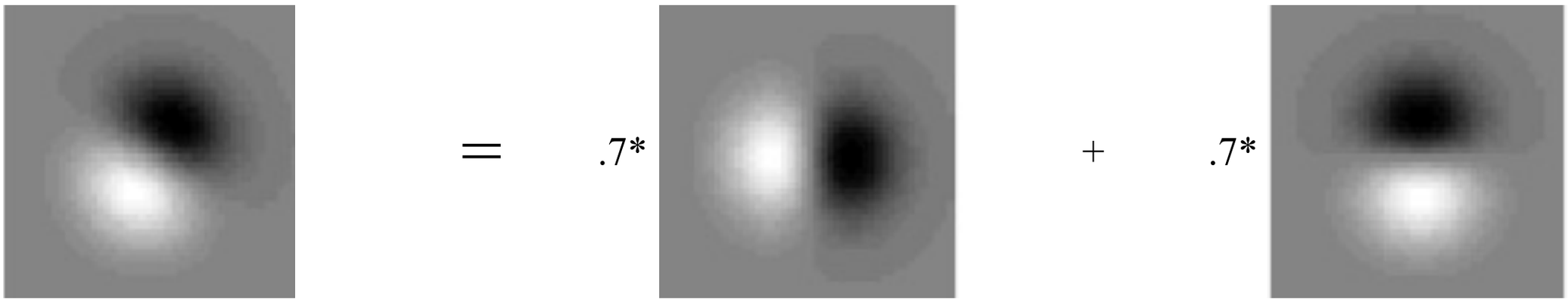
Leung Malik



Gabor filter bank

Remarkable fact

Some oriented filters can be written as linear combinations of a few “basis” filters



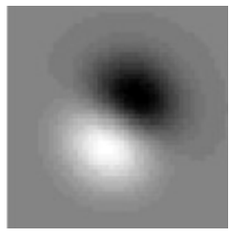
We’ll call such filters *steerable*

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 13, NO. 9, SEPTEMBER 1991 891

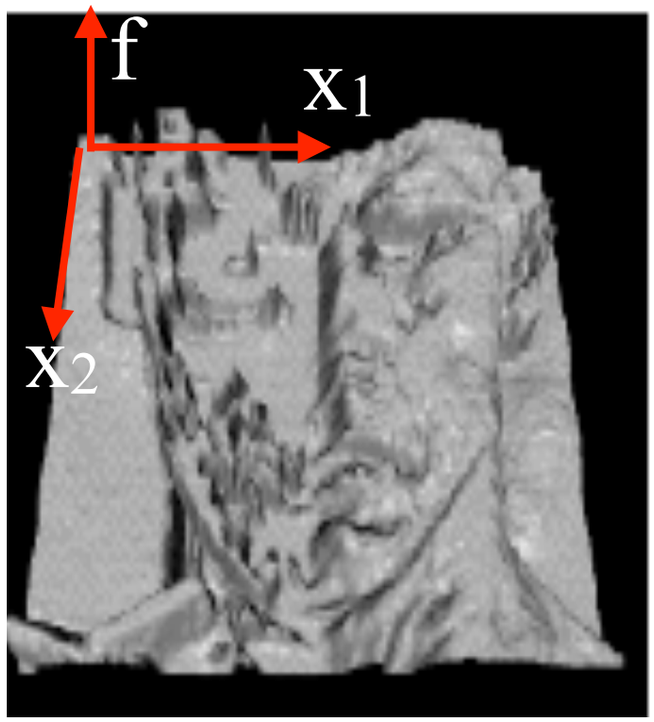
The Design and Use of Steerable Filters

William T. Freeman and Edward H. Adelson

Deriving steerability for derivative-of-guassian filters with a *directional* derivative



https://en.wikipedia.org/wiki/Directional_derivative

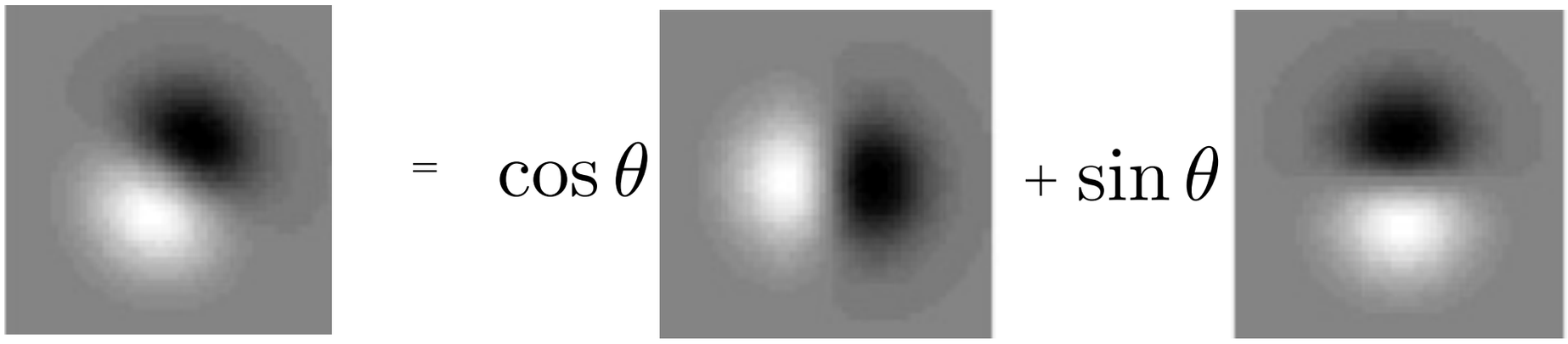


$$f(\mathbf{x}) \quad \text{where} \quad \mathbf{x} = (x_1, x_2), \mathbf{v} = (v_1, v_2)$$

$$\nabla_v f(\mathbf{x}) = \lim_{a \to 0} \frac{f(\mathbf{x} + a\mathbf{v}) - f(\mathbf{x})}{a} = \nabla f(\mathbf{x}) \cdot \mathbf{v}$$

Steering derivative-of-Gaussians

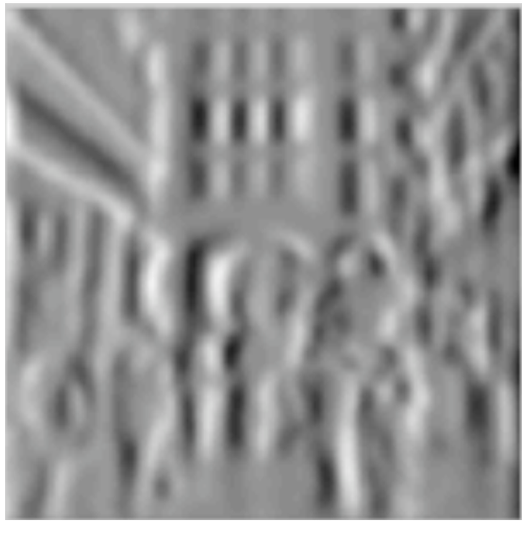
$$\nabla_\theta G_\sigma(x,y) = \cos\theta \frac{\partial G_\sigma}{\partial x} + \sin\theta \frac{\partial G_\sigma}{\partial y}$$



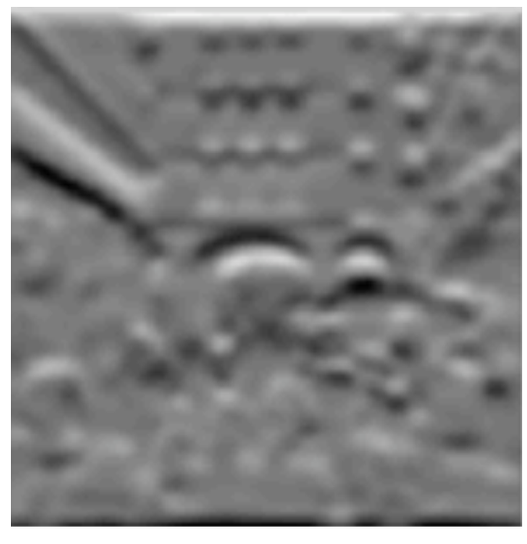
$$= \cos\theta$$

$$+ \sin\theta$$

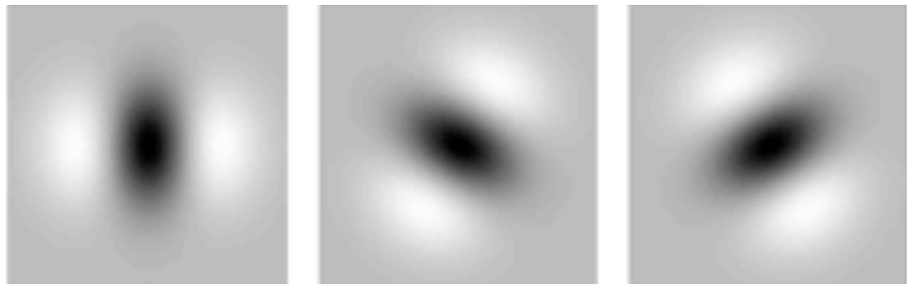
$$= \cos\theta$$



$$+ \sin\theta$$



Second case: Second-derivatives of Gaussians



$$G_\theta = k_a(\theta)G_a + k_b(\theta)G_b + k_c(\theta)G_c$$

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}	=	$0.9213(2y^2 - 1)e^{-(x^2+y^2)}$	$k_c(\theta)$	=	$\sin^2(\theta)$

When is this possible? Filters must be “smooth” in orientation space

Separability

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

Complexity of filtering? (let's stick with correlation for now)

$O(N^2M^2)$

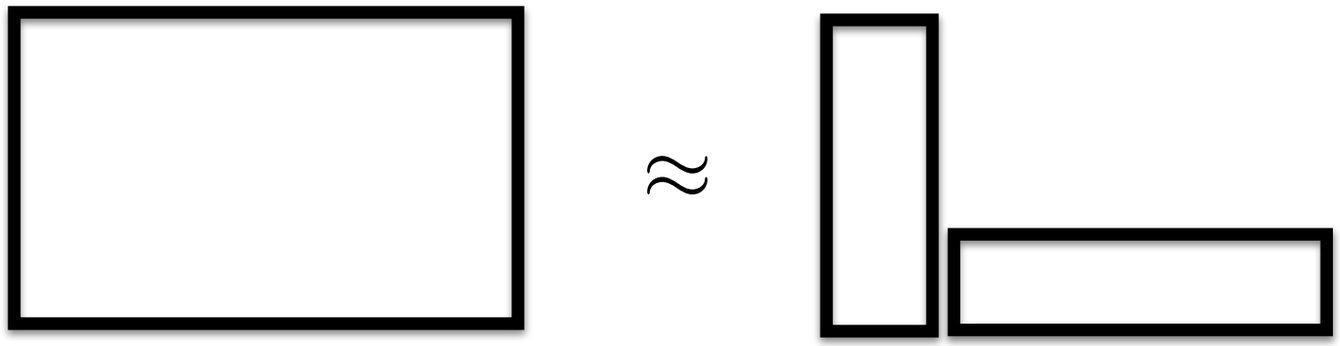
$$H[u,v] = H_x[u]H_y[v]$$

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

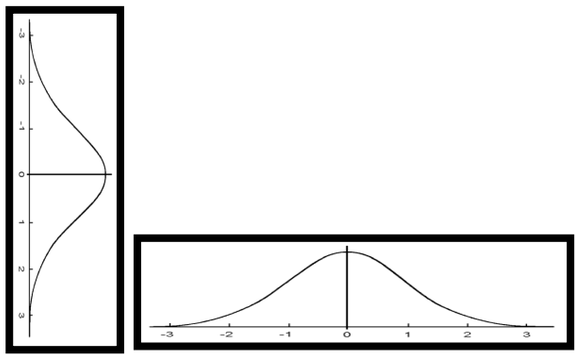
Separability

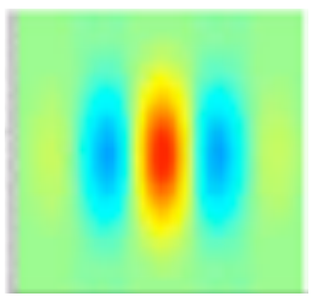
Given a filter, how can we come up with a good separable approximation?

$$H[u, v] \approx H_x[u] H_y[v]$$

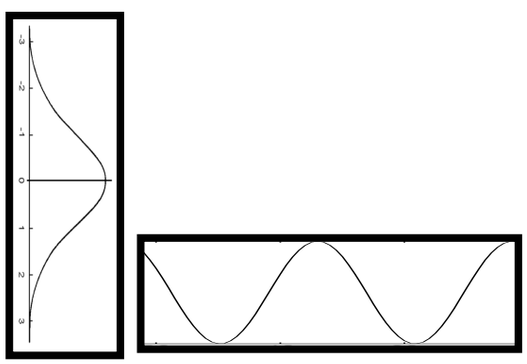


=



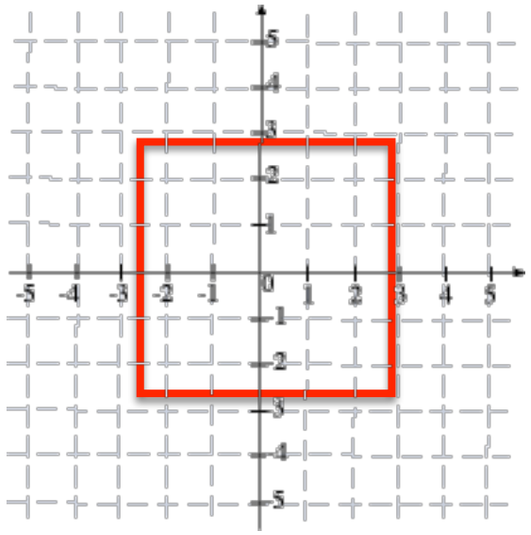


=

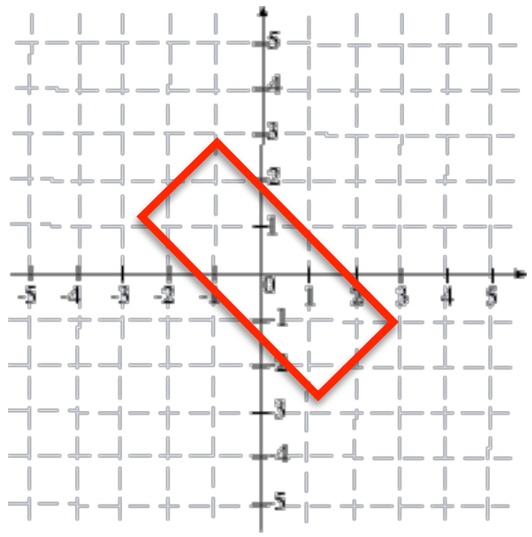


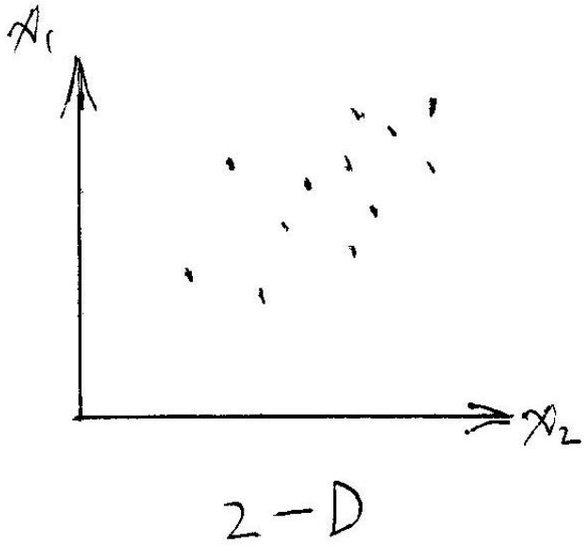
Linear algebra digression

Any matrix can be thought of as a *transformation*



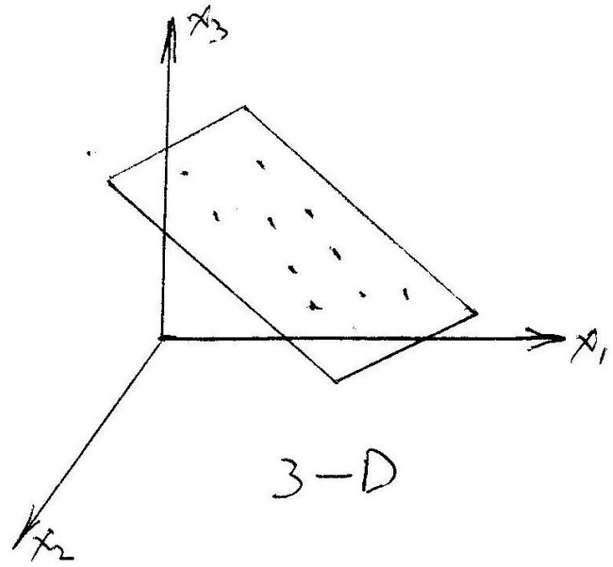
$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$





2-D

$$A = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}$$

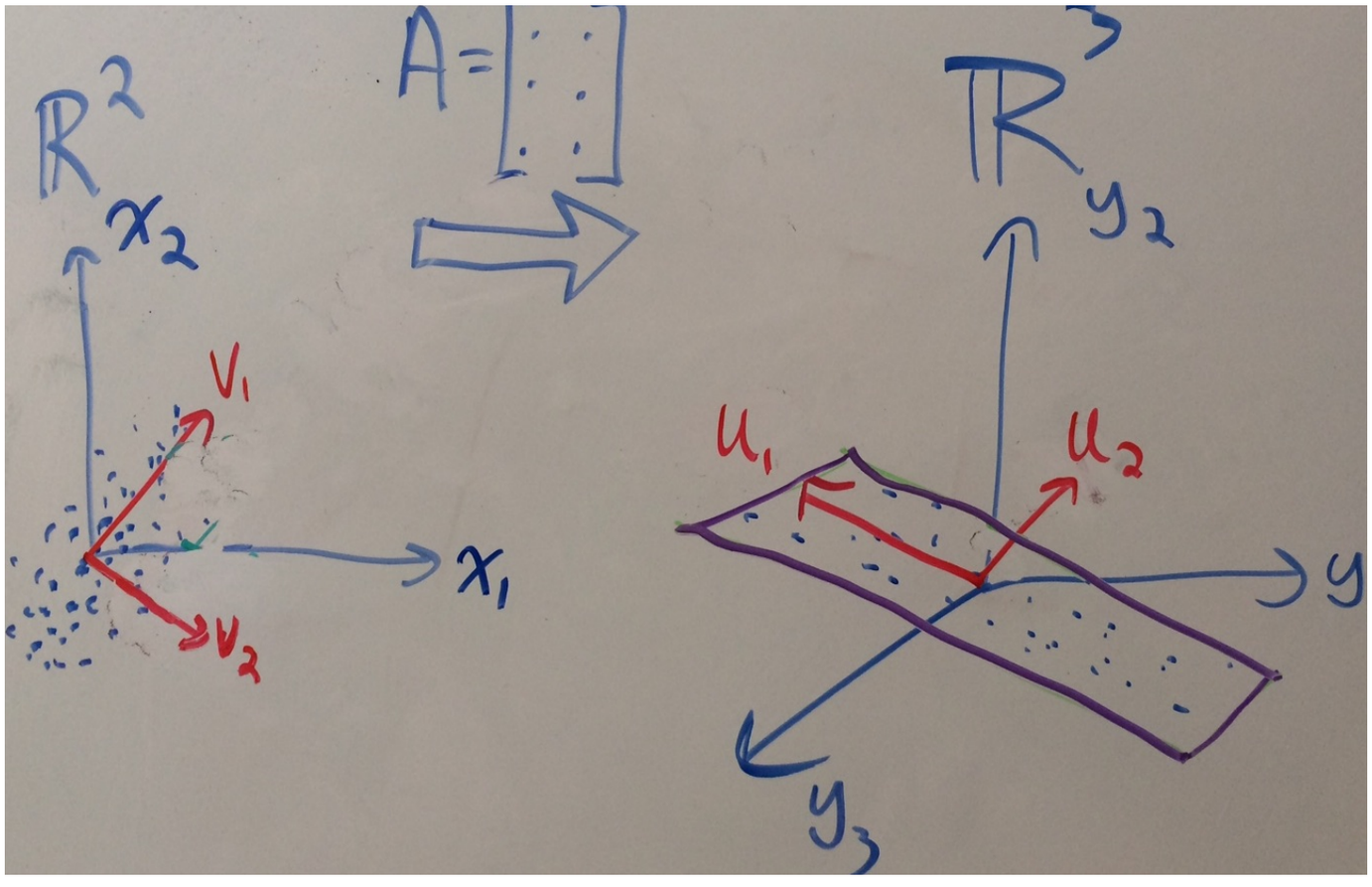


3-D

Change of basis

See handout

Recall: SVD

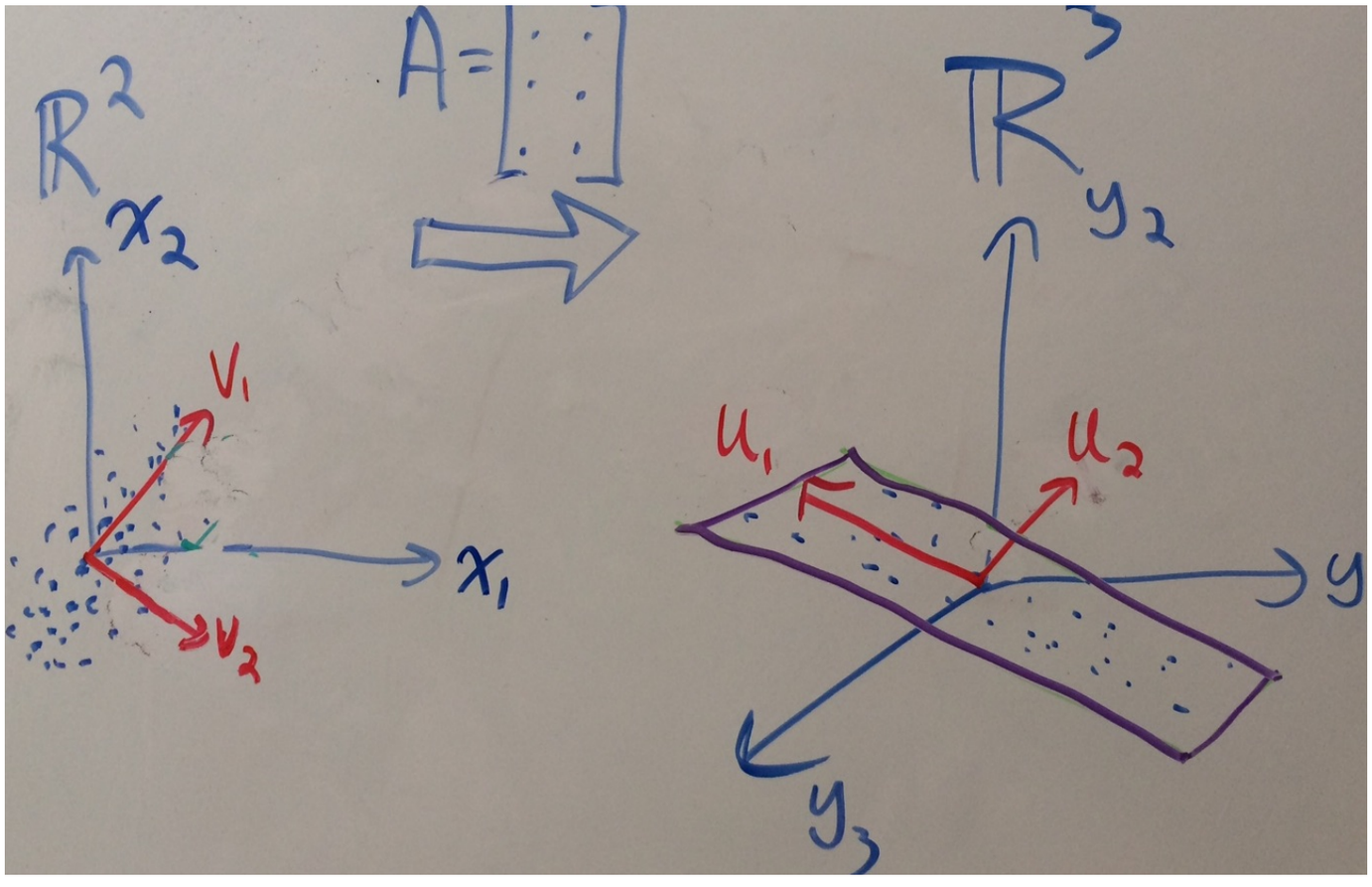


$$y = Ax$$

$$y = U\Sigma V^T x$$

Recall: SVD

$$y = U\Sigma V^T x$$



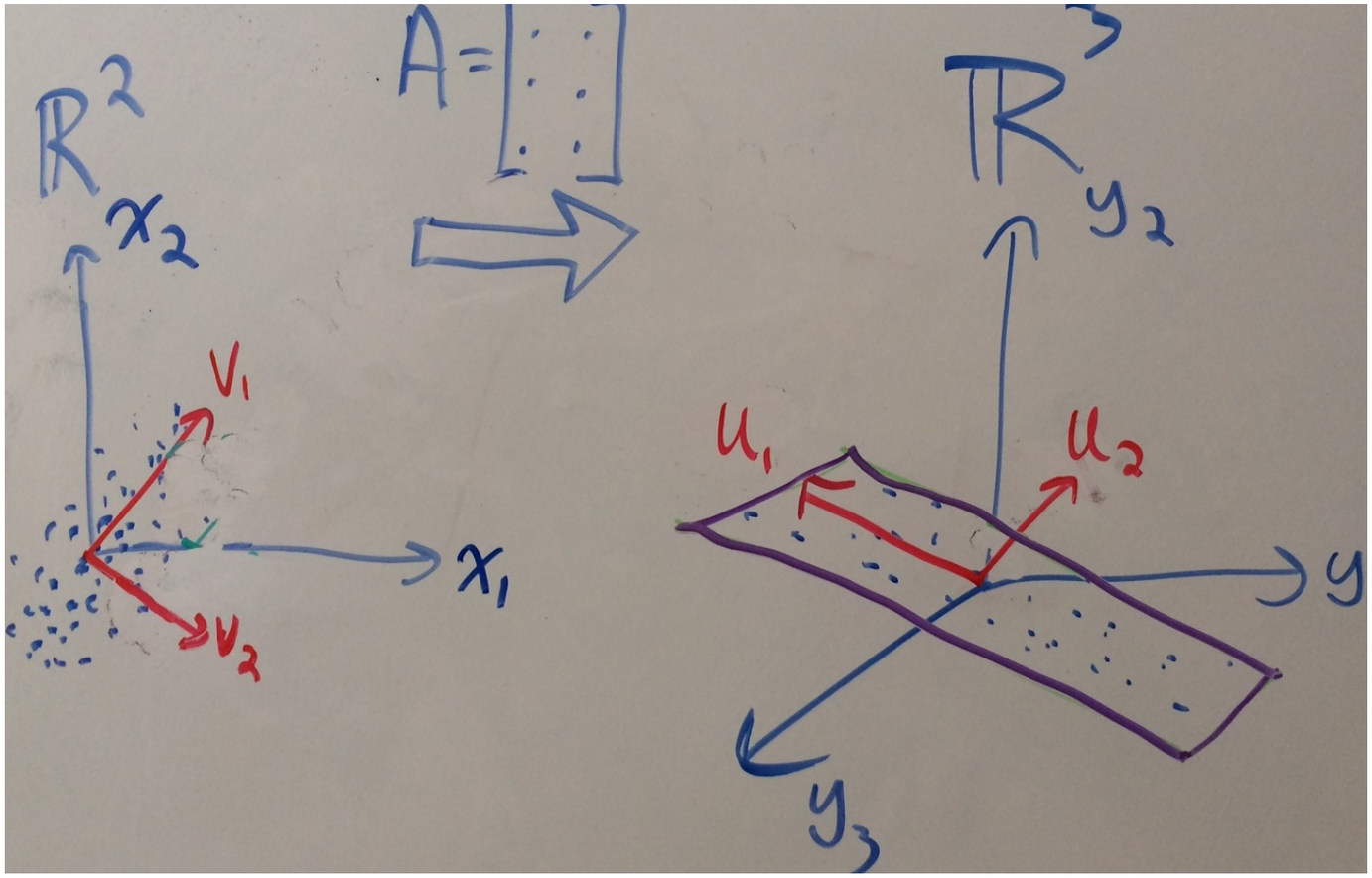
$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0\dots \\ 0 & \sigma_2 & 0\dots \\ 0 & 0 & \sigma_3\dots \\ \vdots & \vdots & \vdots \end{bmatrix}$$

$$U = \begin{bmatrix} u_1 & u_2 \dots u_m \end{bmatrix}, U^T U = I$$

$$V = \begin{bmatrix} v_1 & v_2 \dots v_n \end{bmatrix}, V^T V = I$$

Recall: SVD

$$y = U\Sigma V^T x$$



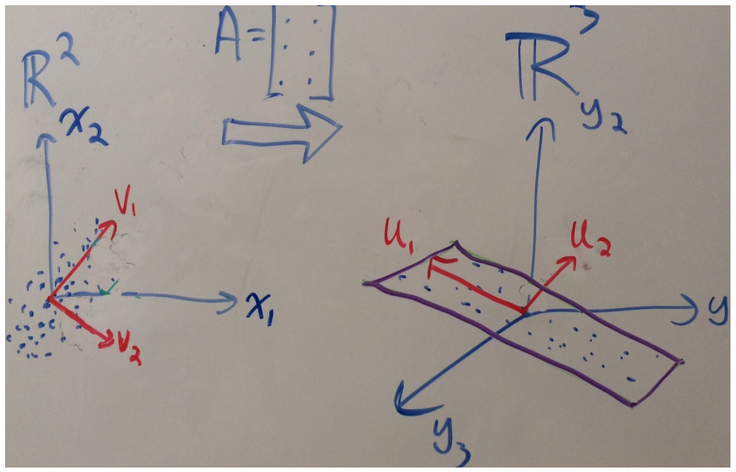
Notation: u_i = left singular vector, $sigma_i$ = singular values, v_i = right singular vectors

Any linear operator can be thought of as mapping from R^m to R^n

1. projection (with right singular vectors)
2. scaling (with singular values),
3. reconstruction (with left singular vectors)

Recall: SVD

Immediate consequences (by appealing to geometric intuition)



$$||A||_F = A(:)^T A(:)$$

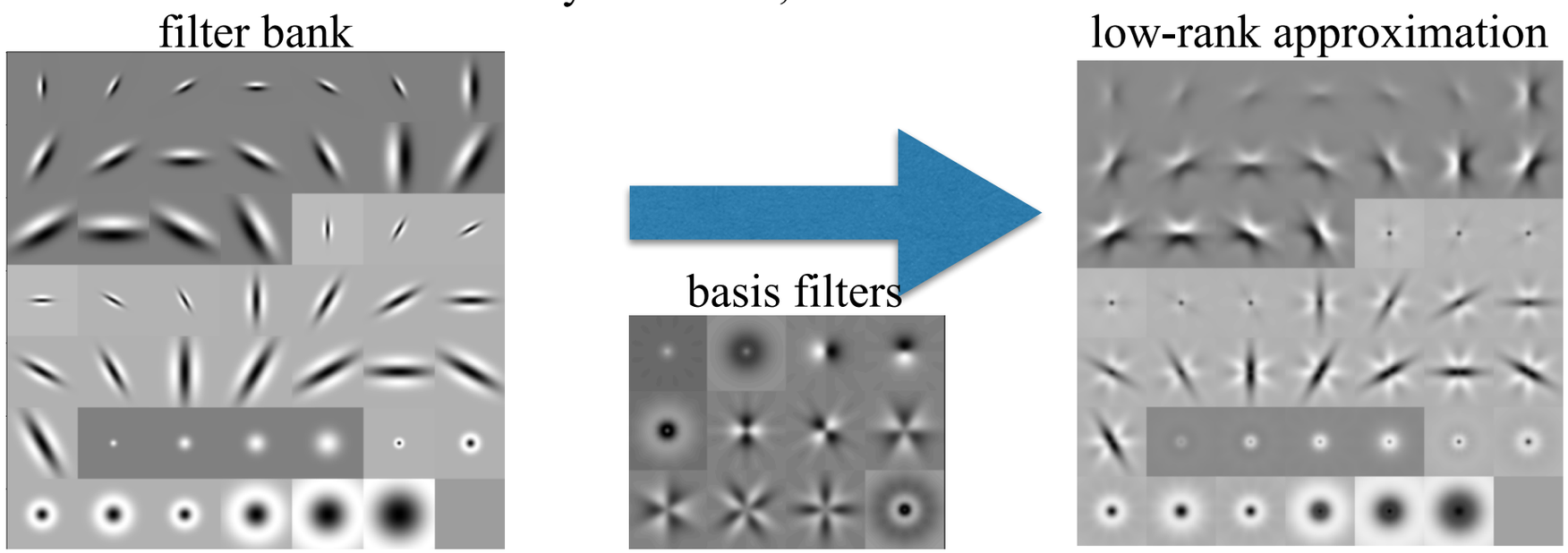
Low rank:
$$\min_{A':rank(A')<=k} ||A - A'||_F = U(:, 1:k)\Sigma(1:k, 1:k)V(:, 1:k)^T$$

Homogenous least-squares:
$$\min_{h:h^T h=1} ||Ah||^2 = V(:, end)$$

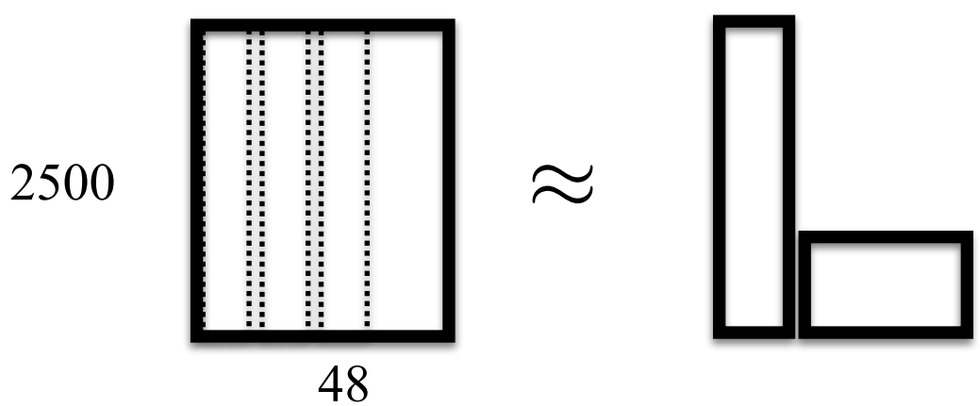
Pseudoinverse:
$$A^+ = \underset{A^+}{\operatorname{argmin}} ||A^+ A - I||_F = V \begin{bmatrix} \frac{1}{\sigma_1} & 0 & 0 \dots \\ 0 & \frac{1}{\sigma_2} & 0 \dots \\ 0 & 0 & \frac{1}{\sigma_3} \dots \\ \vdots & \vdots & \vdots \end{bmatrix}^T U^T$$

Least-squares method of steerability

Shy & Perona, CVPR94



- Column-vectorize each of 48 50x50 filters and stack'em into a 2500x48 matrix
- Apply SVD to generate low-rank approximation of 48 filters as linear combination of 12 *basis* filters



Least-squares method of steerability

Shy & Perona, CVPR94

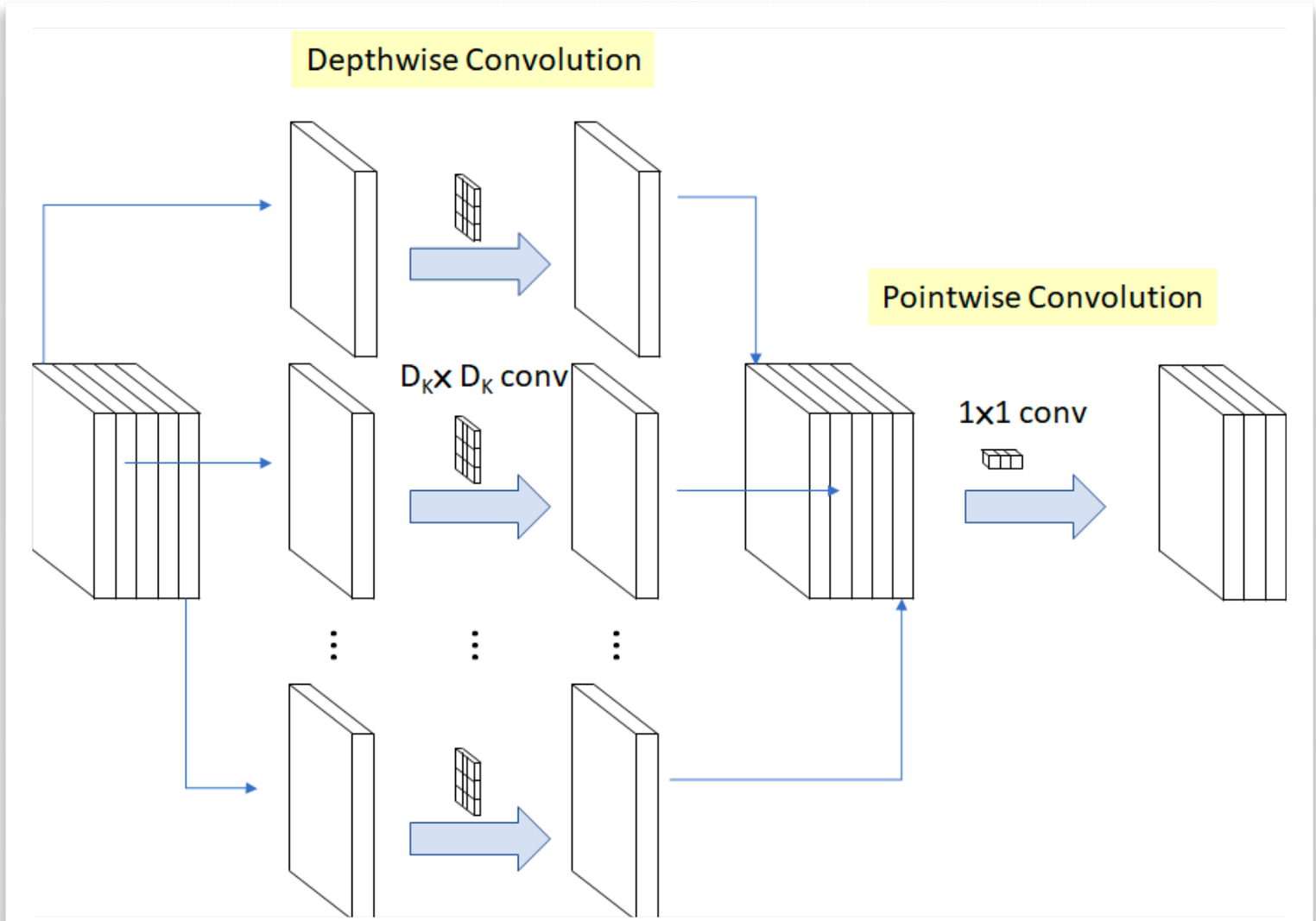
Rank 1 approximation

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

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

Low-rank separability is ubiquitous in neural networks

Depth-separable convolution in Google's MobileNet



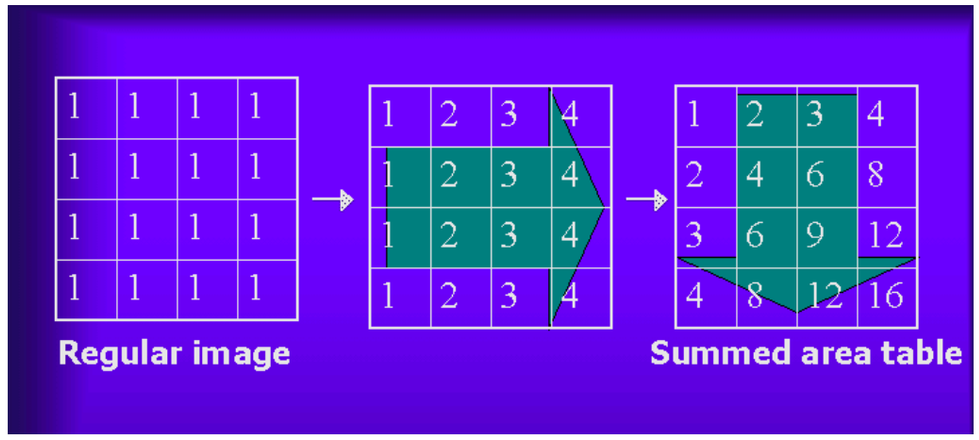
Final trick for efficient *box* filters

Consider a filter of all “1”’s (which sums up values in a neighborhood)

Turns out there is a way to implement filter that is *independant* of filter size

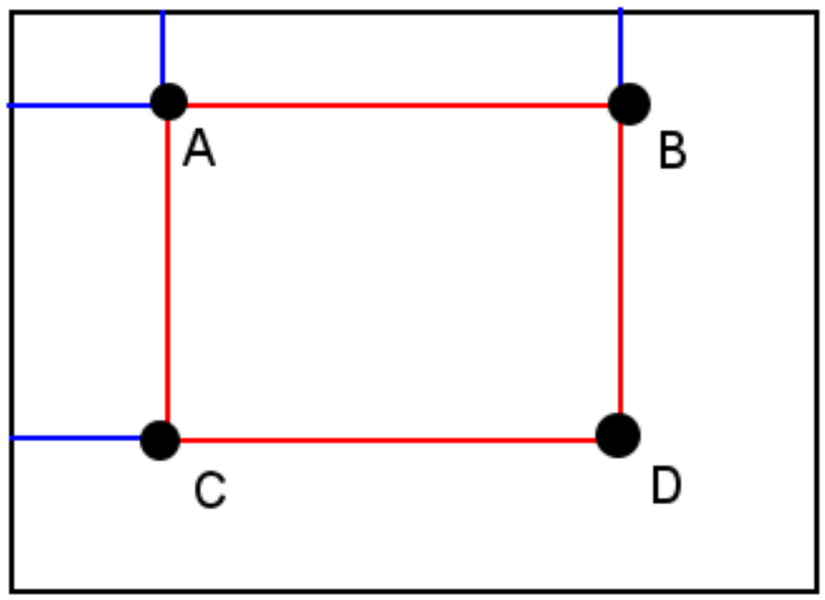
Box filtering with integral images (or summed area tables)

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

Compute filter response for any size window with 4 table lookups (in integral image)!

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

Efficient detection

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

A look back

- Convolution (linear-shift-invariant)
- Edges (Canny, hysteresis, LoG)
- Efficiency (pyramids, separability, steerability)
- Next class: bag-of-words, texture, frequencies