Image and Vision Computing

Image Filtering

Instructor: Chen Yisong

HCI & Multimedia Lab, Peking University

What is an image?

- Binary
- Gray Scale
- Color

Image as a matrix: Binary Images

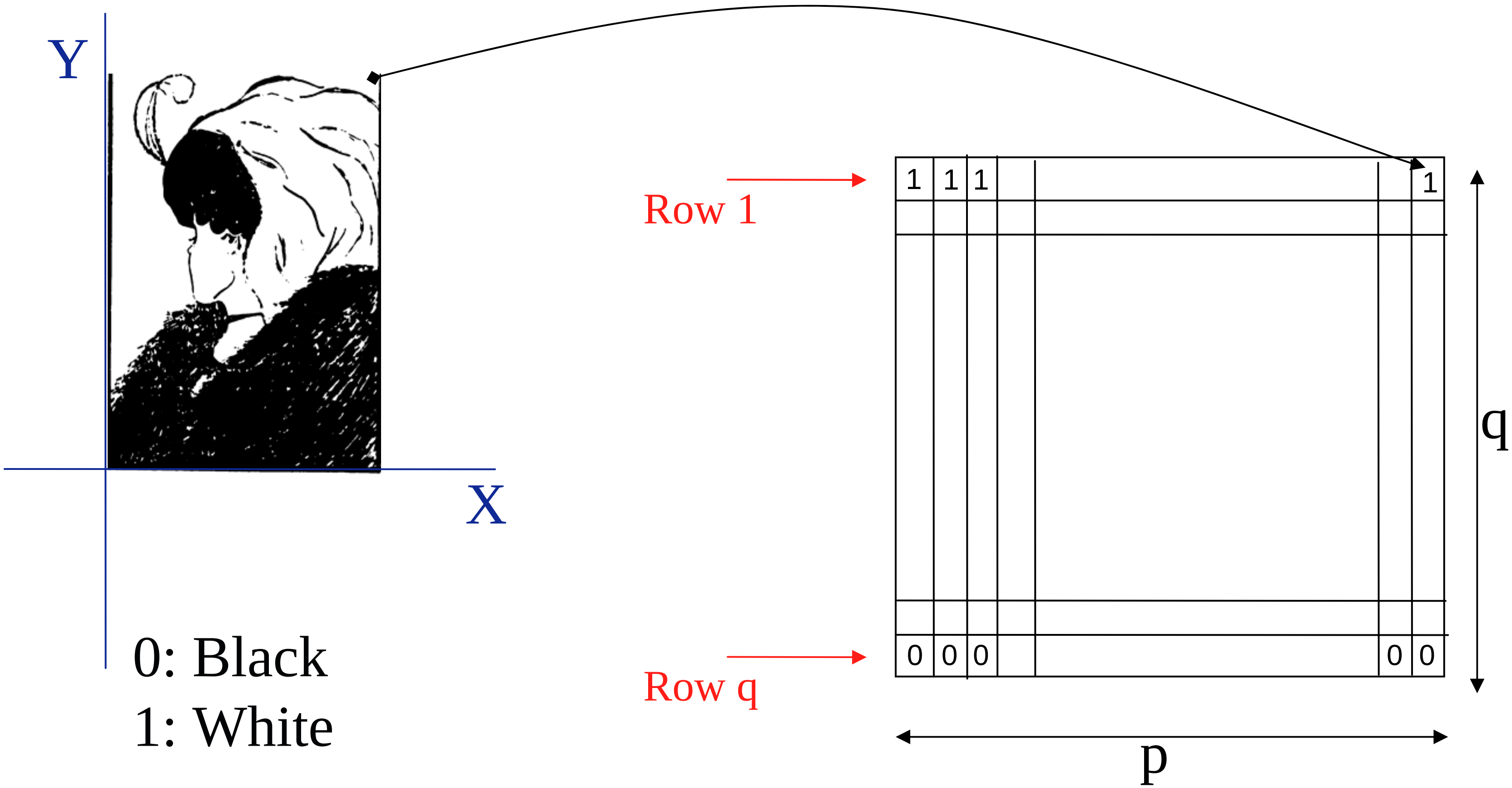


Image as a matrix:
Gray Level Image

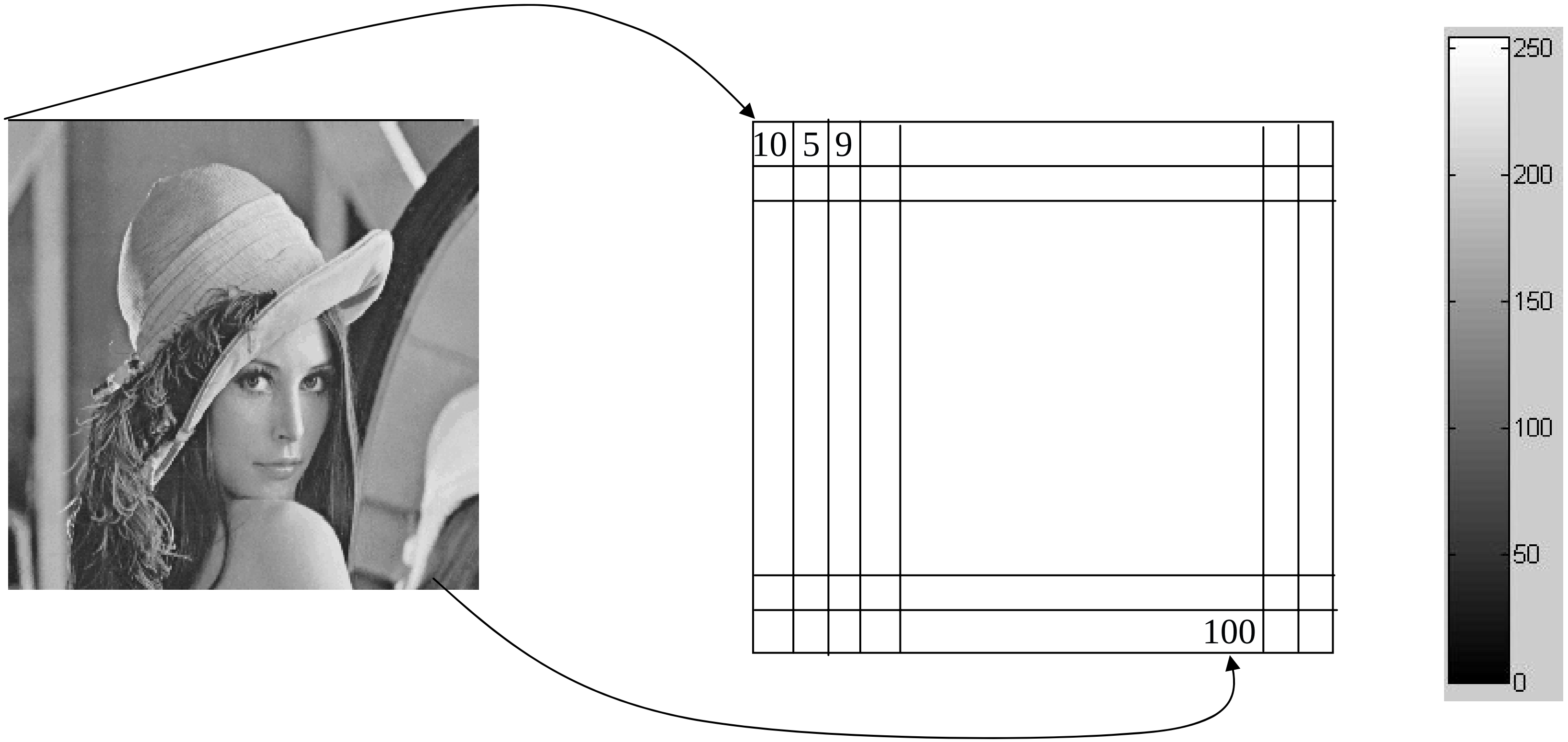


Image as multiple matrices: Color Image

What can we do with an image?

Object Detection/Recognition

Curve Detection/Fitting

Line Detection/Fitting

Key Feature Estimation

Scene Editing/Augmentation

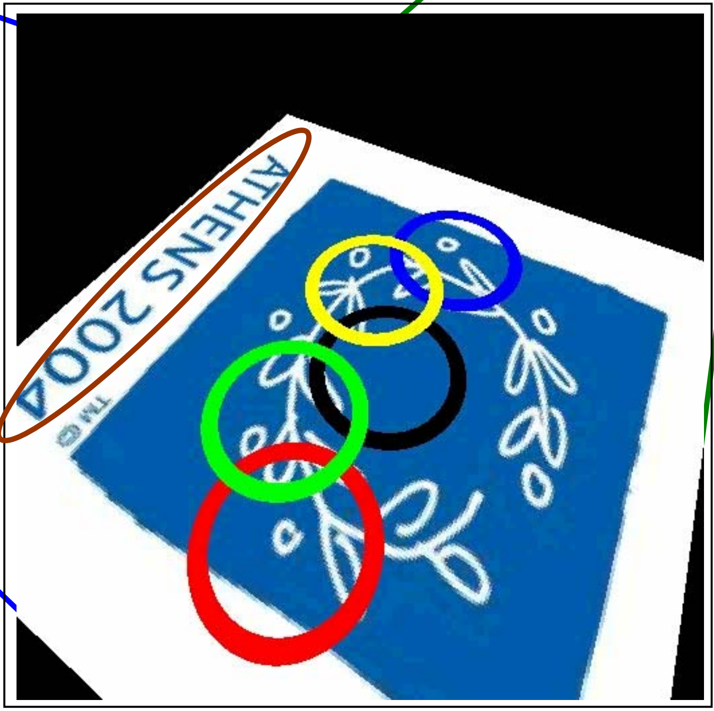
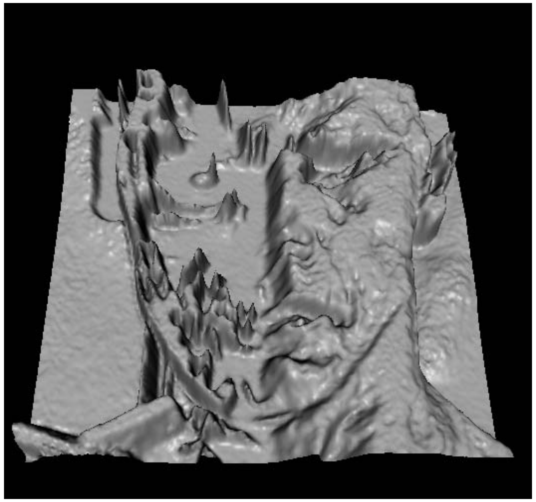


Image as a Function

- We can think of an **image** as a function, *f*, from R^2 to R:
  - $f(x, y)$ gives the **intensity** at position (x, y)
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    - f: $[a,b]$x$[c,d] \rightarrow [0,1]$
- A color image is just three functions pasted together. We can write this as a "vector-valued" function:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

Image as a Function



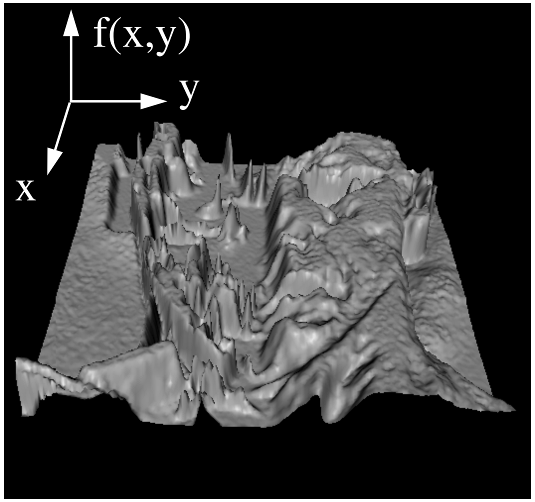


Image Processing

- Define a new image *g* in terms of an existing image *f*
  - We can transform either the domain or the range of *f*
- Range transformation:

$$g(x, y) = t(f(x, y))$$

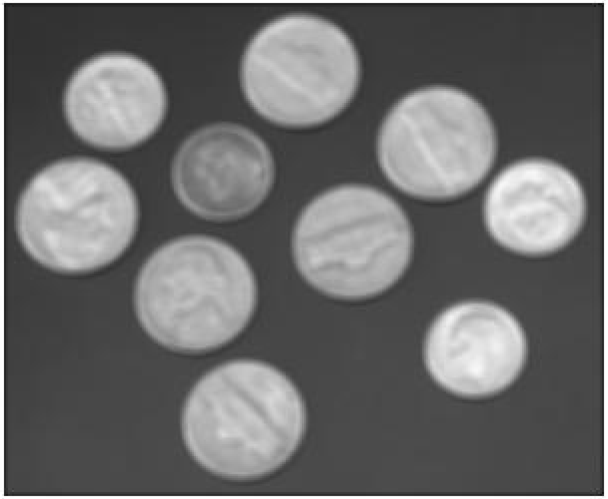
What kinds of operations can this perform?

Smoothing, Enhancing, Denoising, Binarizing……

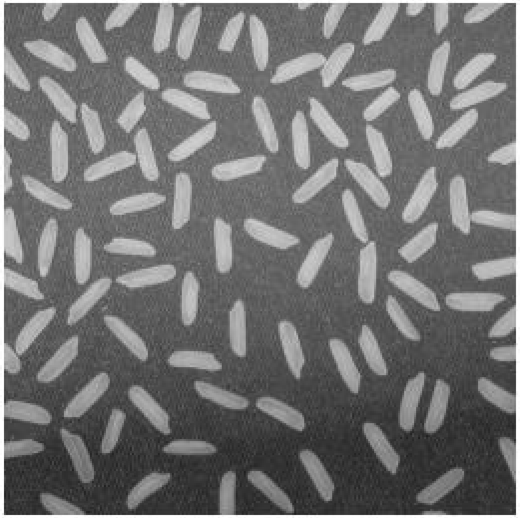
Blur

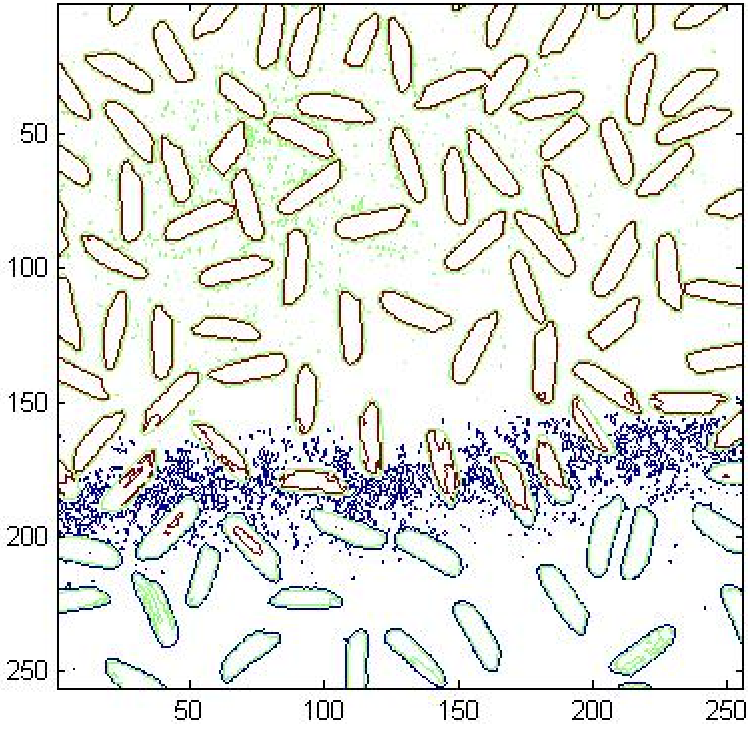
Original Image

Filtered Image

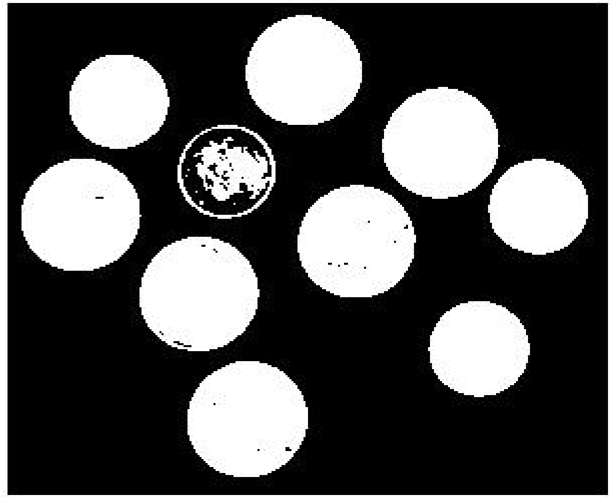


Contour





More examples



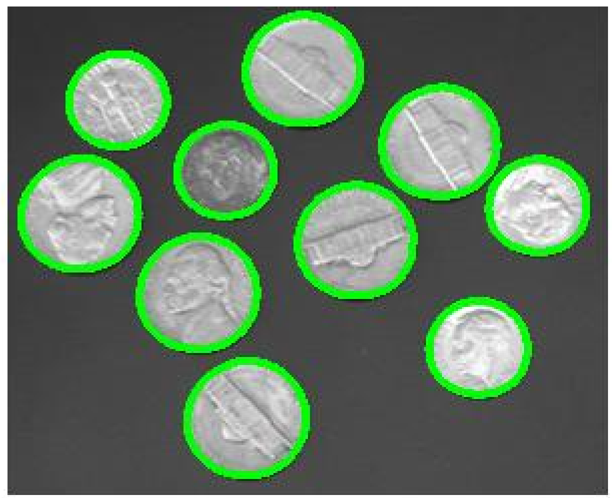


Image Processing

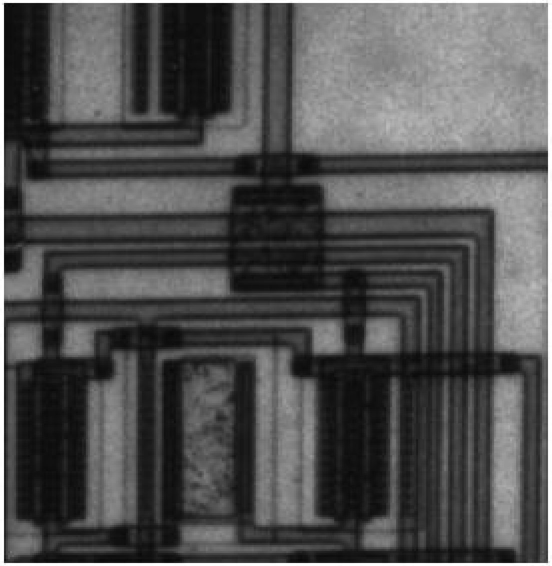
- Some operations preserve the range but change the domain of f :

$$g(x,y) = f(t_x(x,y), t_y(x,y))$$

- What kinds of operations can this perform?

Translation, Rotation, Scaling……

Image Scaling



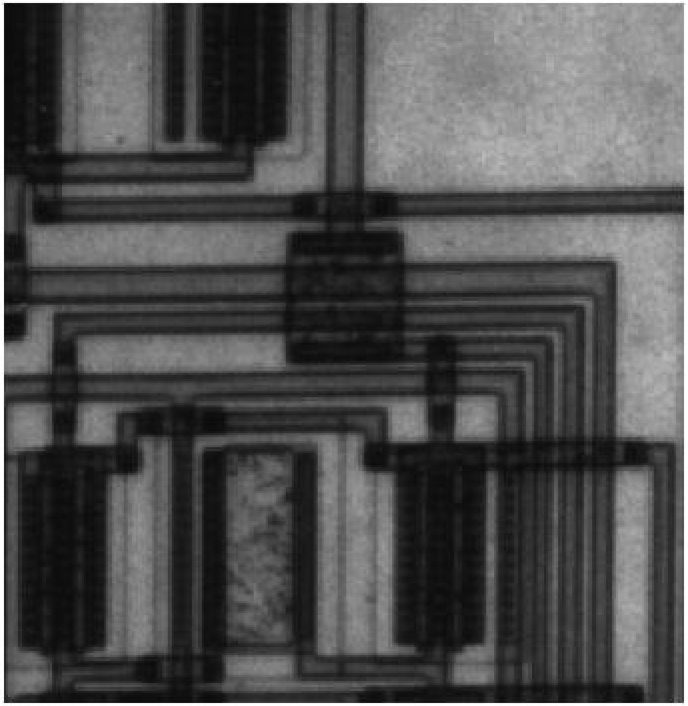
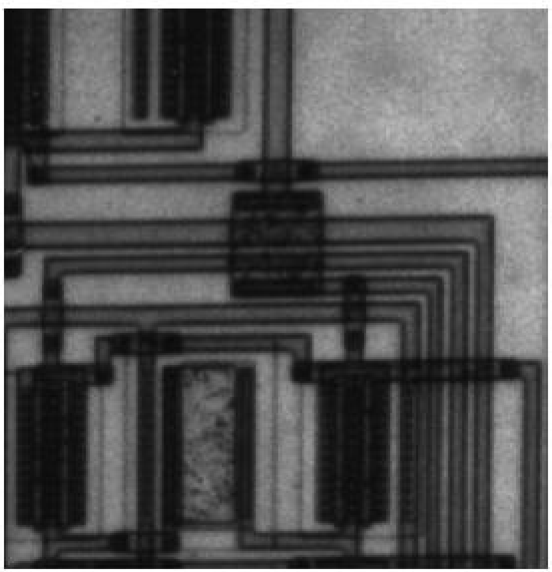


Image Rotation



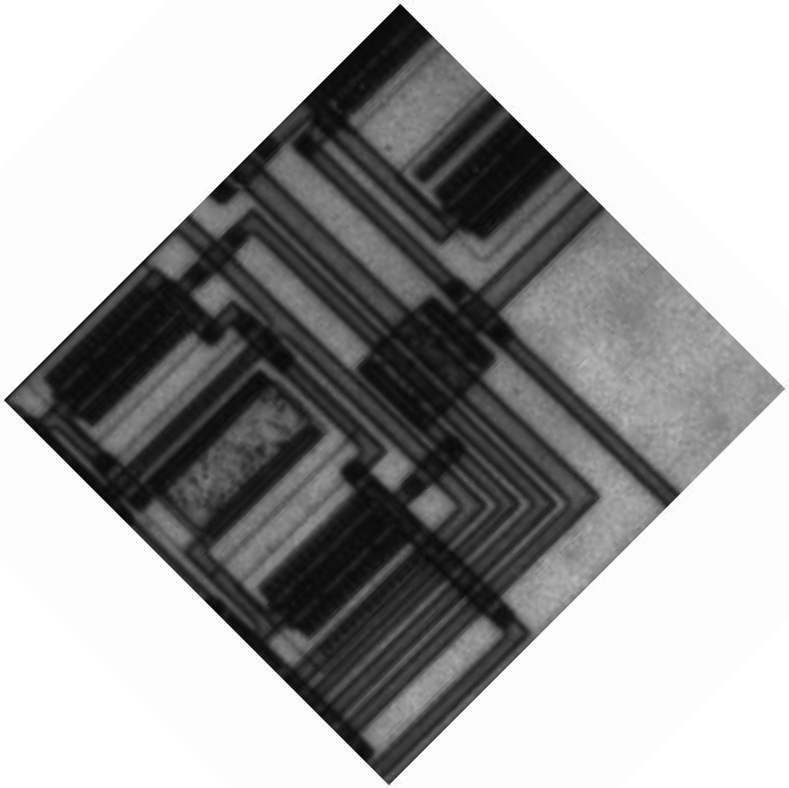


Image Processing

- Still other operations operate on both the domain *and* the range of f .

$$g(x, y) = s(f(t_x(x, y), t_y(x, y)))$$

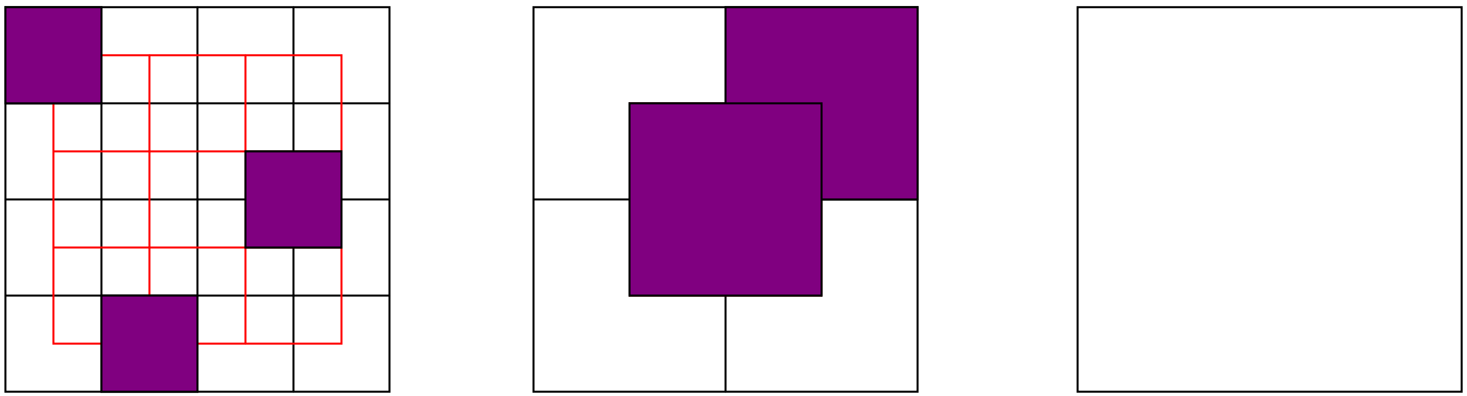
- What kinds of operations can this perform?

Fractal Image coding, Wavelet Image coding……

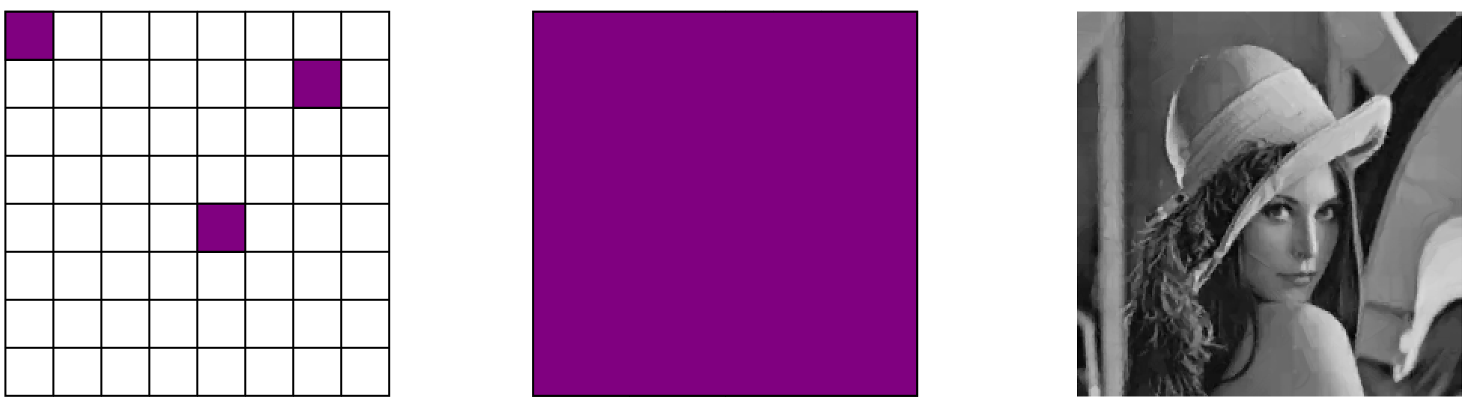
Fractal image coding

$$\omega_i\left(\begin{bmatrix} x_D \\ y_D \\ z_D \end{bmatrix}\right) = \begin{bmatrix} x_R \\ y_R \\ z_R \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix}\begin{bmatrix} x_D \\ y_D \\ z_D \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix}$$

- 迭代函数系统IFS
  - 定义域块



  - 值域块



Wavelet Image Coding

original image

one step decomposition

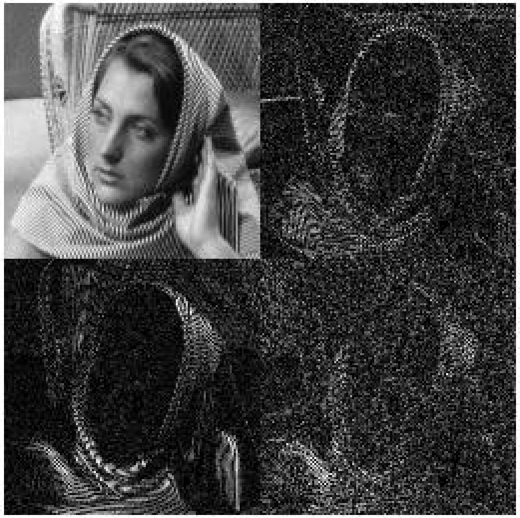


Image Scaling

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?

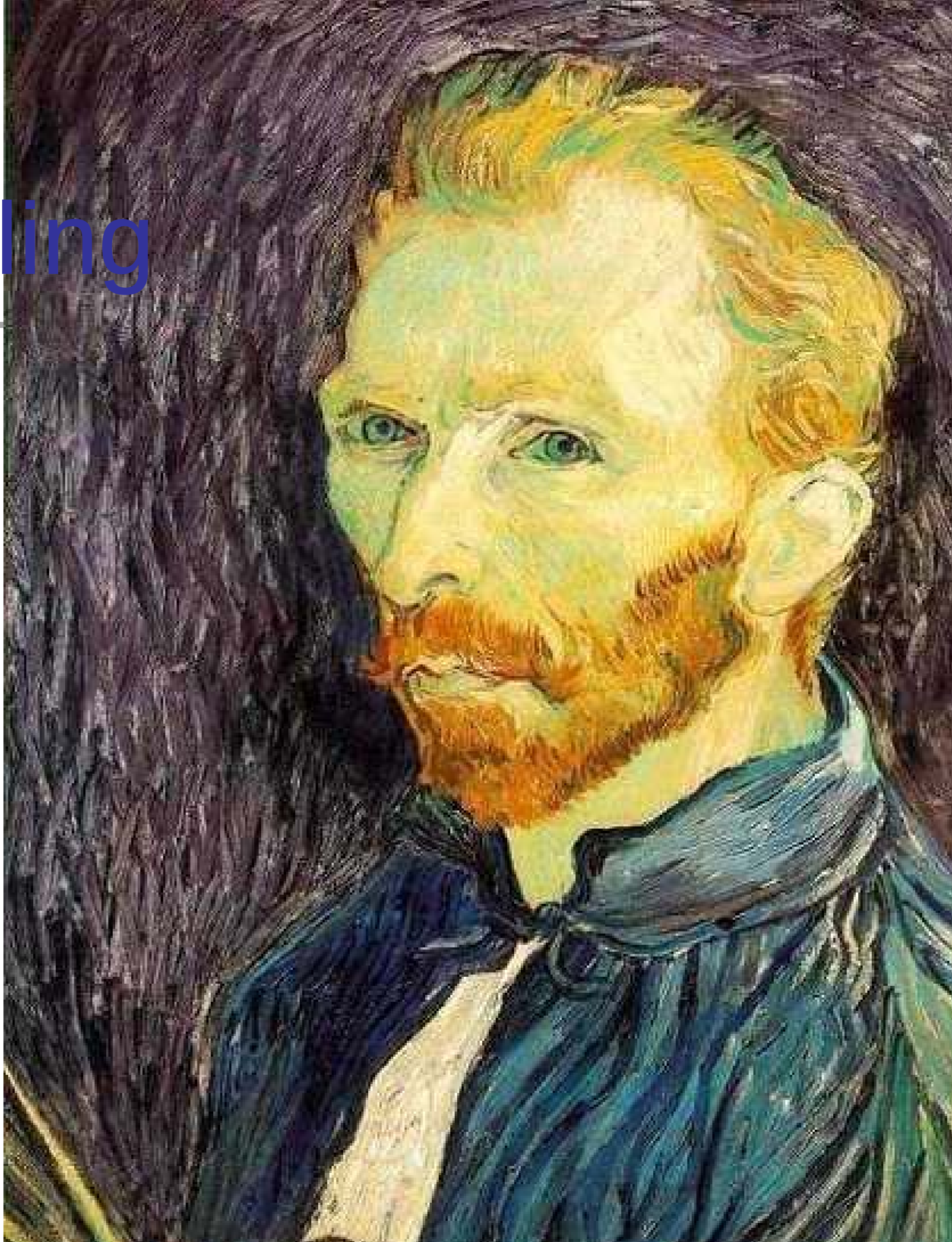
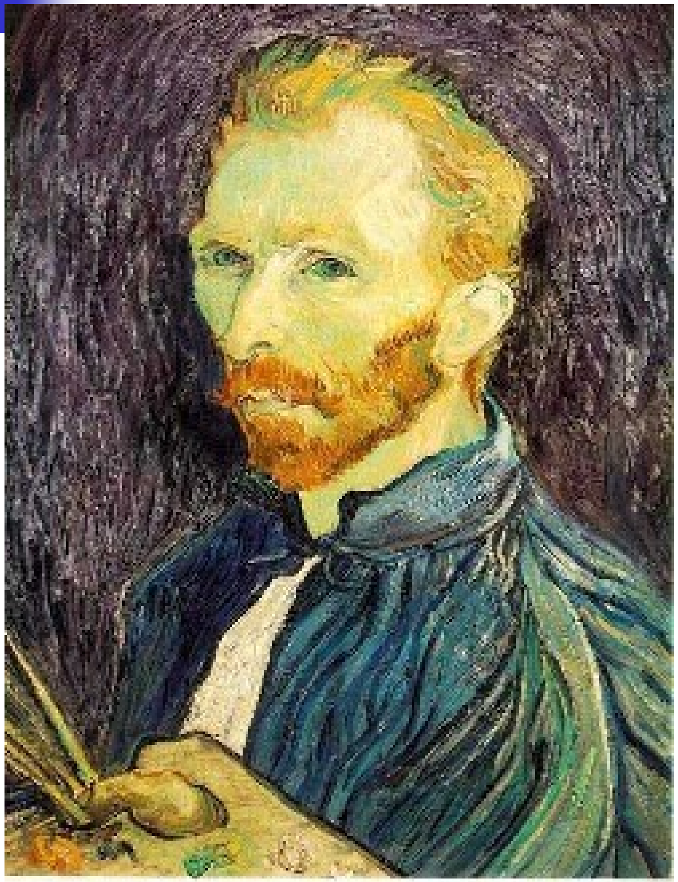
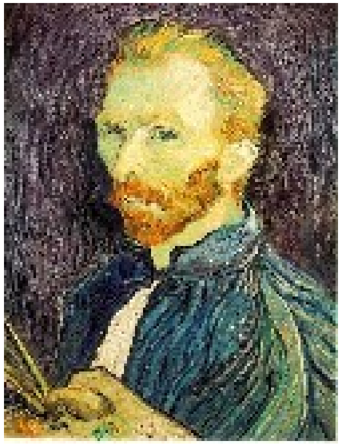


Image Sub-Sampling



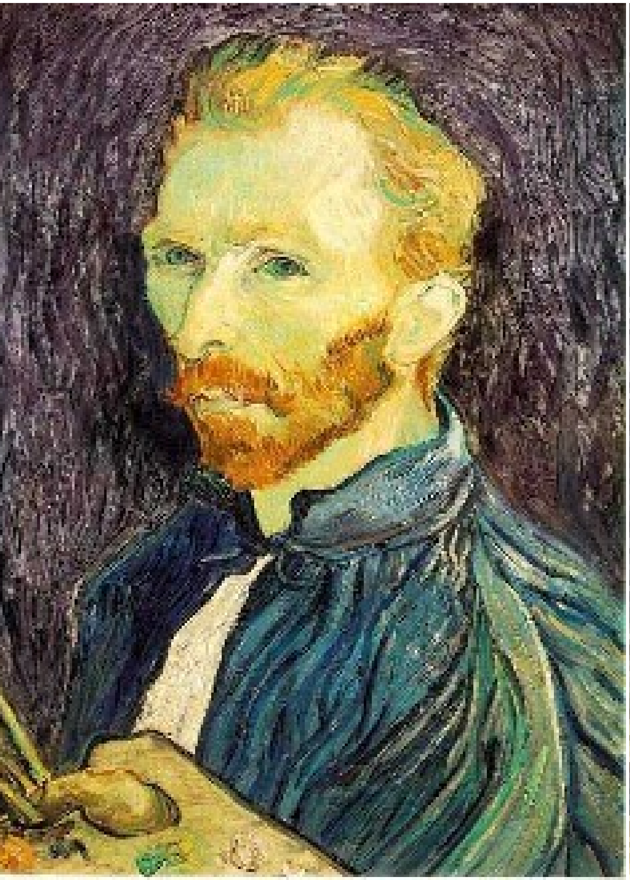


1/4

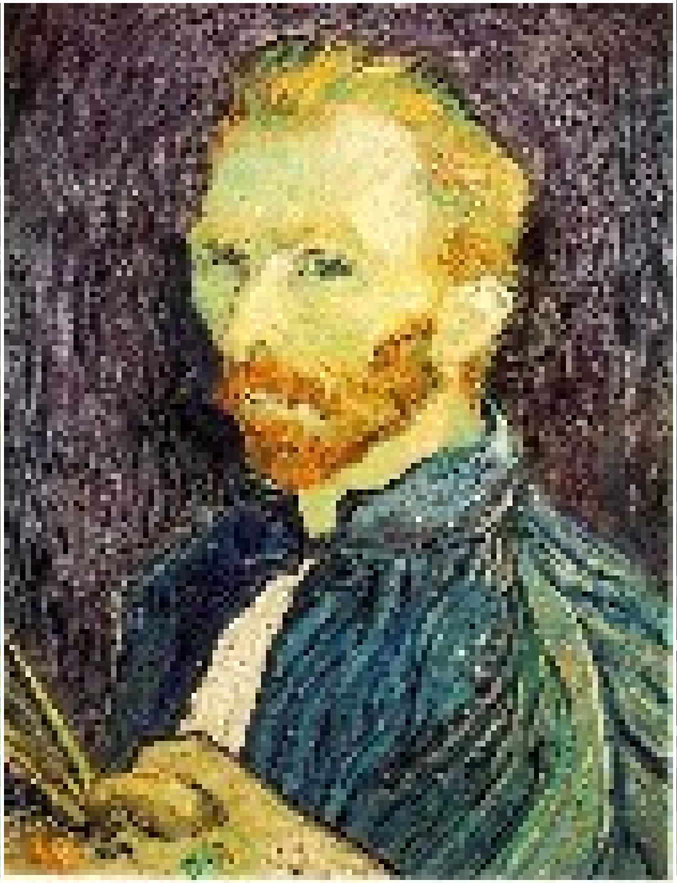
1/8

A image of size 1/2 size image is created by throwing away every other row and column - called *image sub-sampling*

Image Sub-Sampling



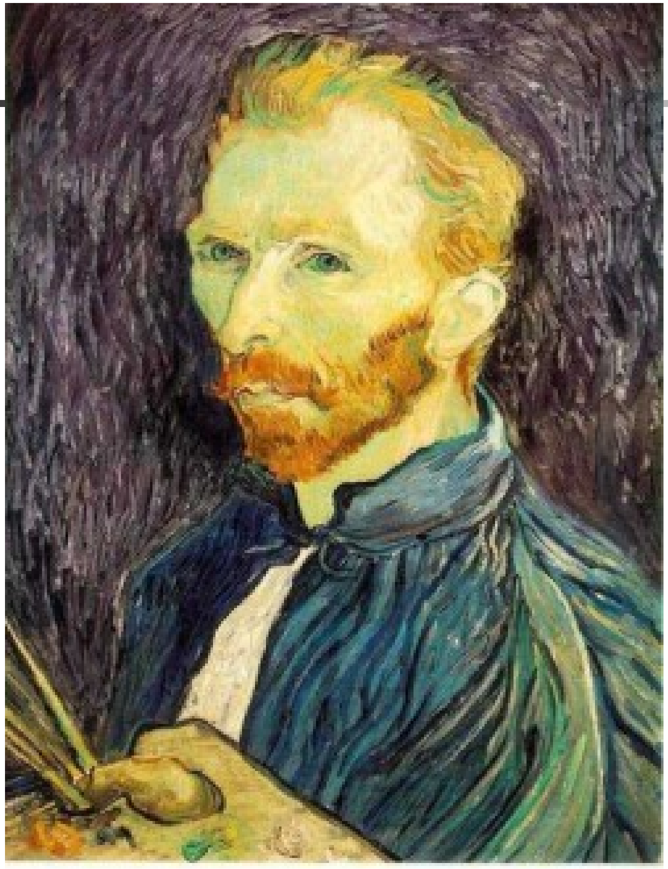
1/2



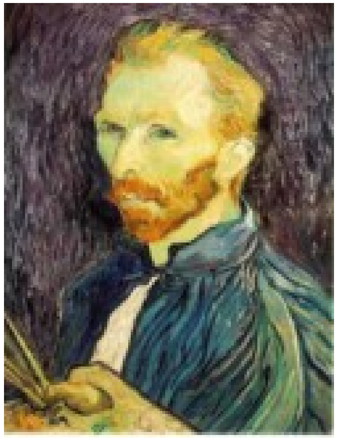
1/4 (2x zoom)

1/8 (4x zoom)

Sub-Sampling with Gaussian Pre-Filtering



Gaussian 1/2

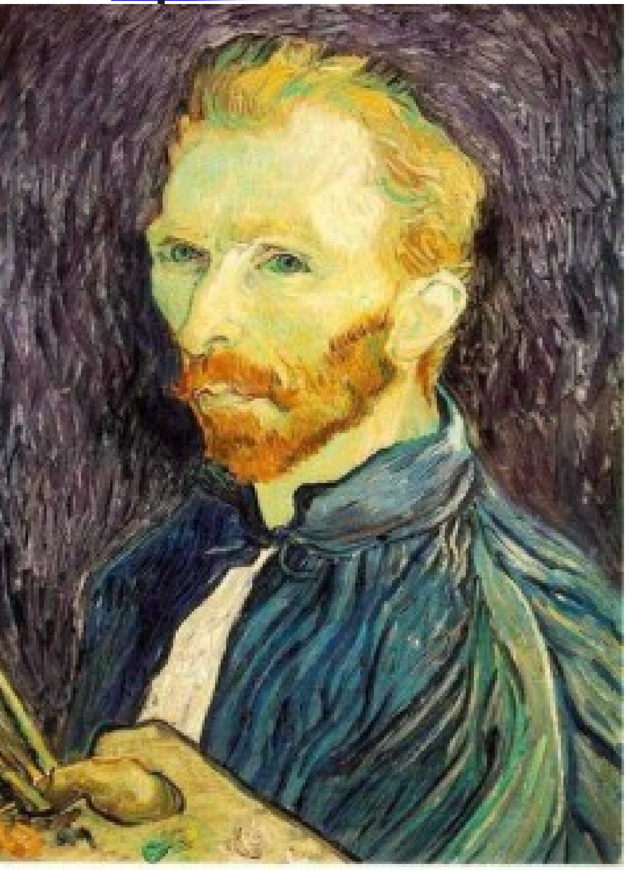


G 1/4

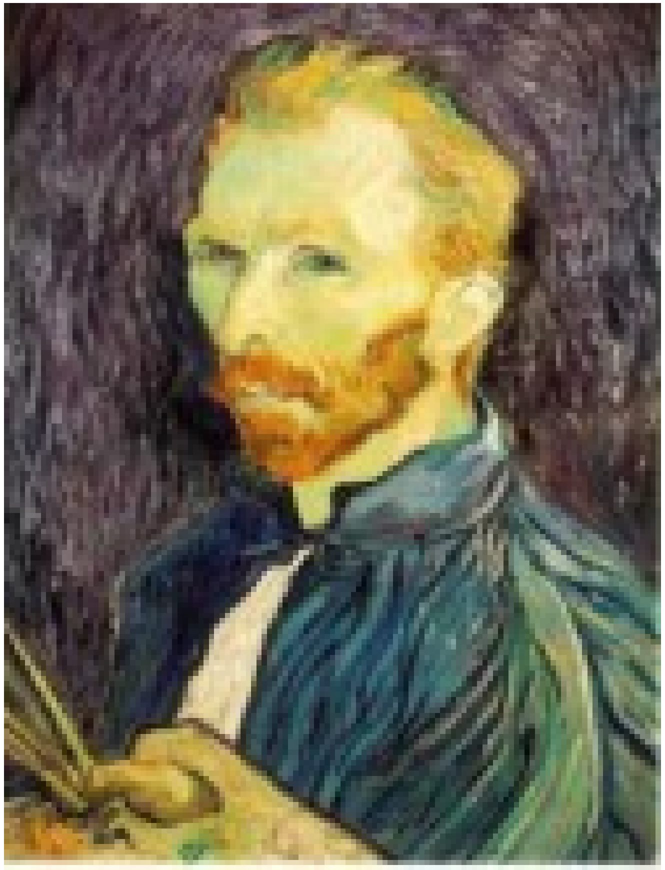
G 1/8

- Solution: filter the image, *then* subsample
  - Filter size should double for each ½ size reduction. Why?

Sub-Sampling with Gaussian Pre-Filtering



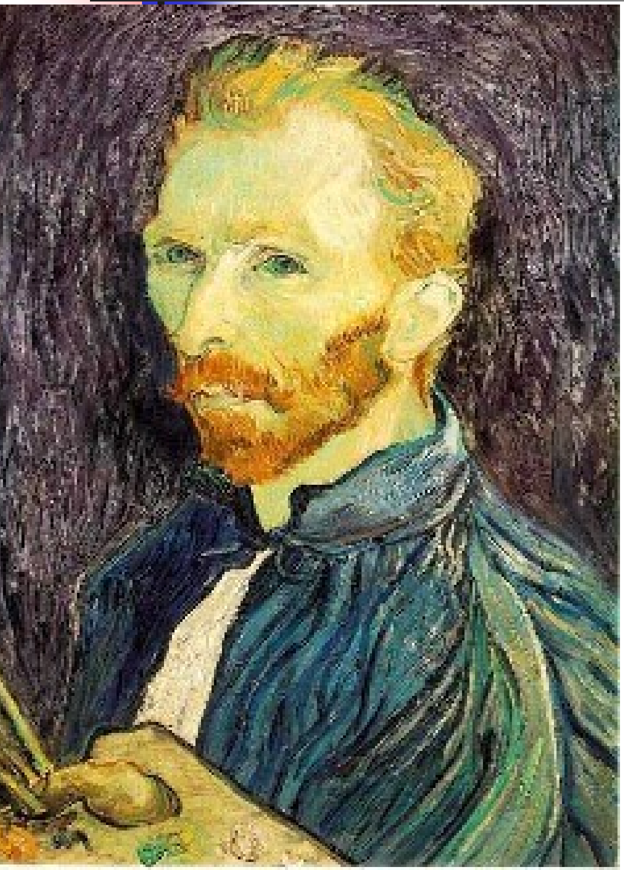
Gaussian 1/2



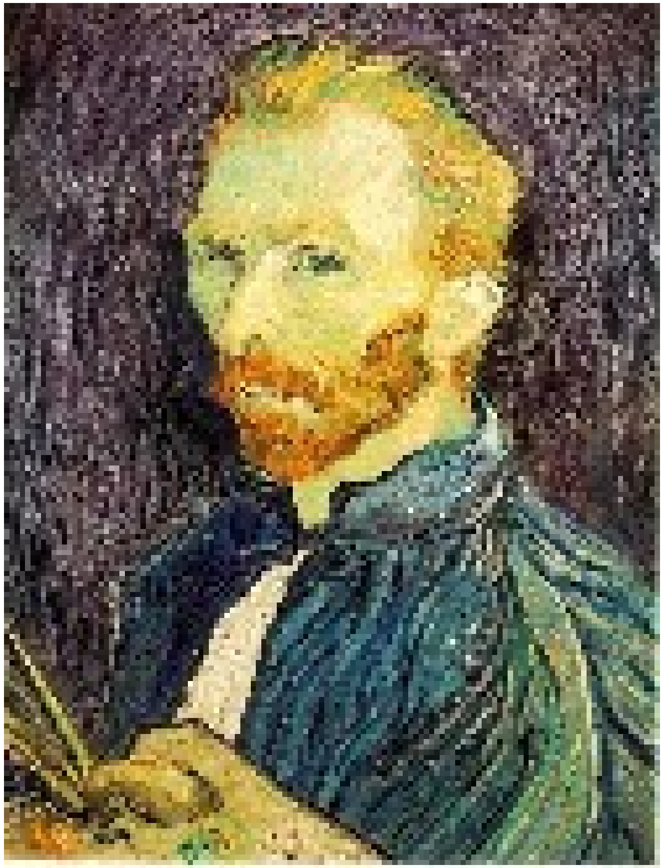
G 1/4

G 1/8

Compare with...



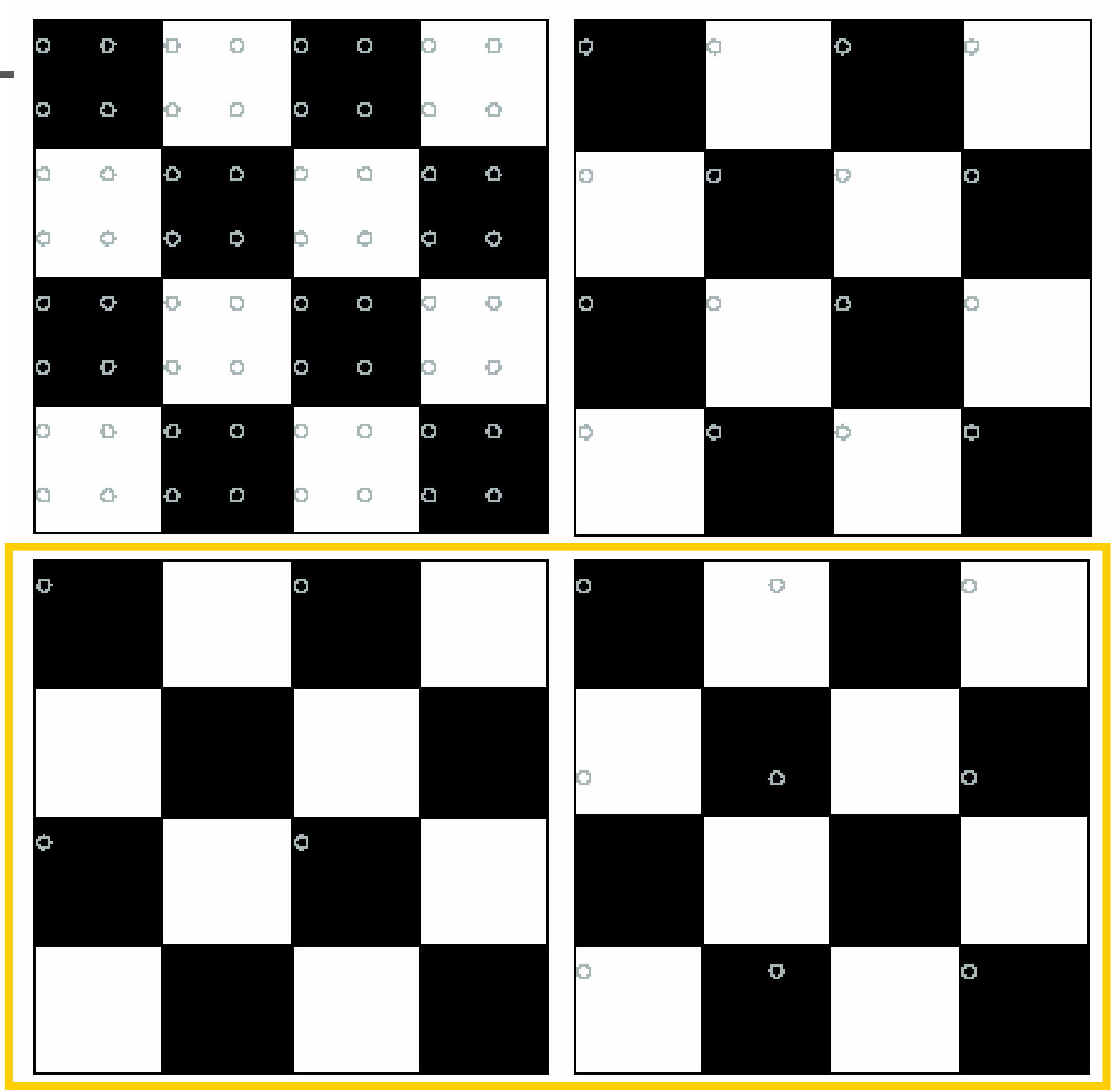
1/2



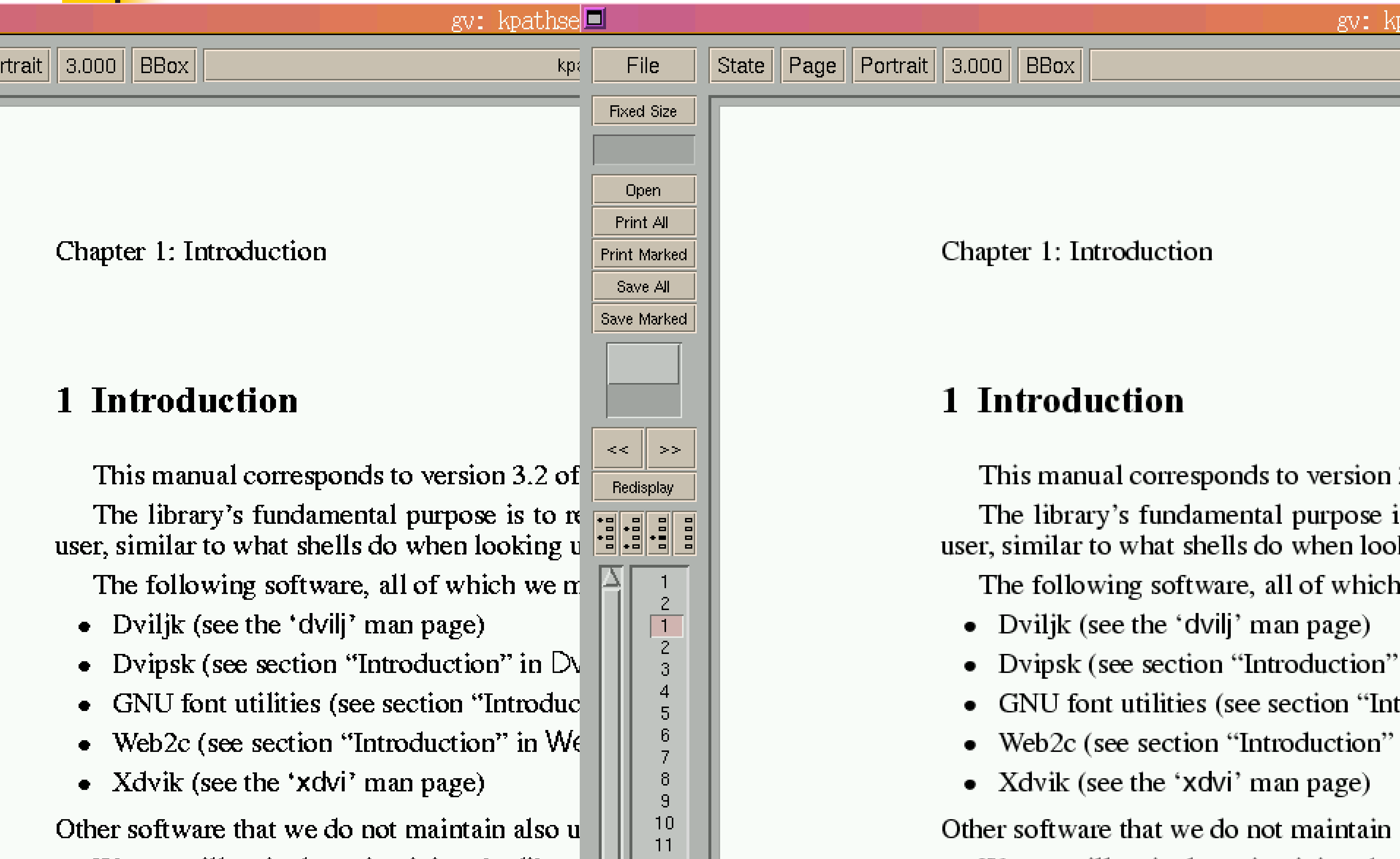
1/4 (2x zoom)

1/8 (4x zoom)

Aliasing(混叠/走样)



Aliasing


gv: kpathse
rtrait
3.000
BBox
File
State
Page
Portrait
3.000
BBox
Fixed Size
Open
Print All
Print Marked
Save All
Save Marked
Redisplay
Chapter 1: Introduction
1 Introduction
This manual corresponds to version 3.2 of
The library's fundamental purpose is to re
user, similar to what shells do when looking u
The following software, all of which we m
• Dviljk (see the 'dvilj' man page)
• Dvipsk (see section "Introduction" in Dv
• GNU font utilities (see section "Introduc
• Web2c (see section "Introduction" in We
• Xdvik (see the 'xdvi' man page)
Other software that we do not maintain also u
Chapter 1: Introduction
1 Introduction
This manual corresponds to version
The library's fundamental purpose i
user, similar to what shells do when loo
The following software, all of which
• Dviljk (see the 'dvilj' man page)
• Dvipsk (see section "Introduction"
• GNU font utilities (see section "Int
• Web2c (see section "Introduction"
• Xdvik (see the 'xdvi' man page)
Other software that we do not maintain

Canon D60 (w/ anti-alias filter)

Sigma SD9 (w/o anti-alias filter)

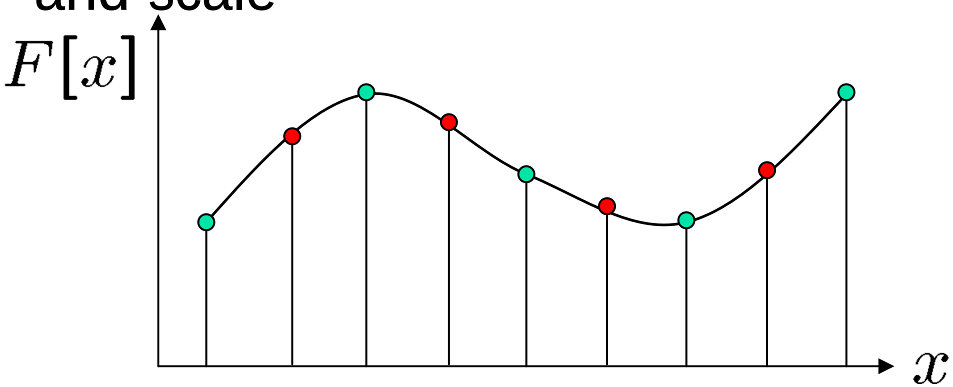
From Rick Matthews website, images by Dave Etchells

Image Resampling

- So far, we considered only power-of-two subsampling
  - What about arbitrary scale reduction?
  - How can we increase the size of the image?
- Recall how a digital image is formed

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

  - It is a discrete point-sampling of a continuous function
  - If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

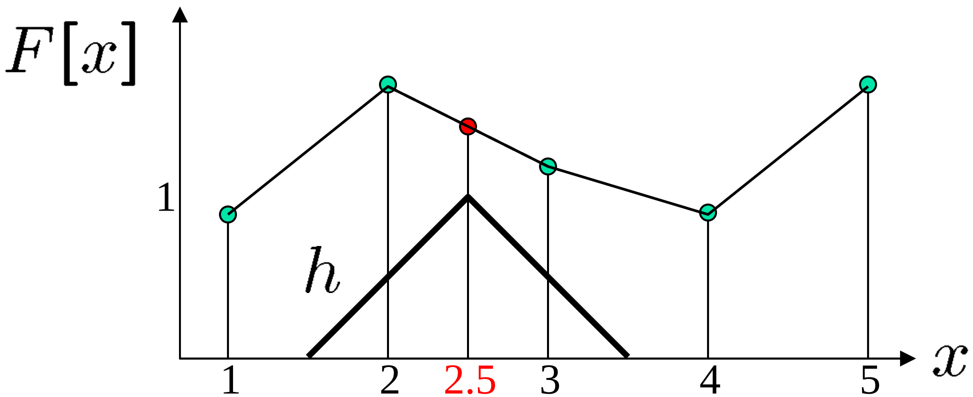


d = 1 in this example

Now d = 0.5

Image Resampling

- So what to do if we don't know f
  - Answer: guess an approximation $\tilde{f}$
  - Can be done in a principled way: filtering



d = 1 in this example

- Image reconstruction
  - Convert F to a continuous function

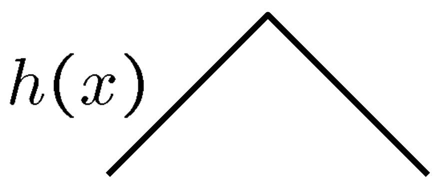
$$f_F(x) = F(\tfrac{x}{d}) \text{ when } \tfrac{x}{d} \text{ is an integer, 0 otherwise}$$

  - Reconstruct by cross-correlation:

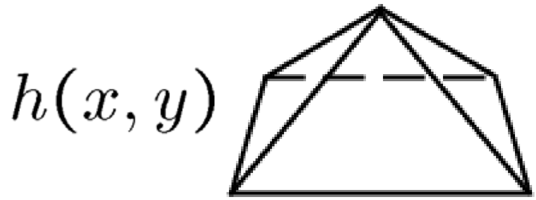
$$\tilde{f} = h \otimes f_F$$

Resampling Filters

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



performs
linear interpolation

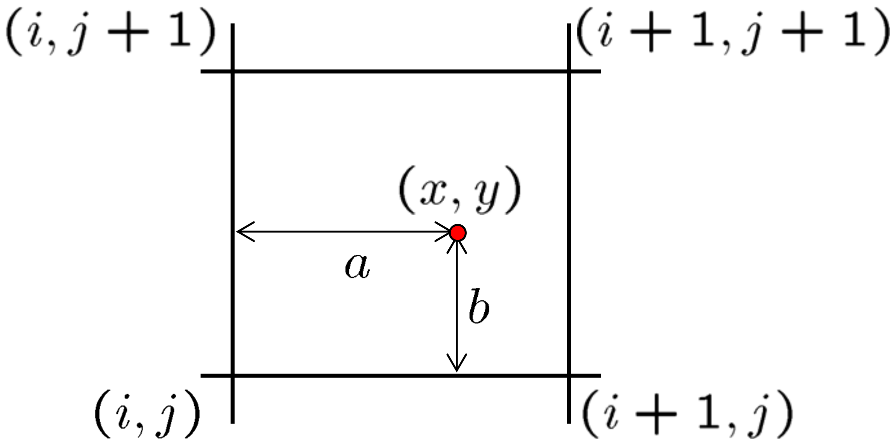


(tent function) performs
bilinear interpolation

- Better filters give better resampled images
  - Bicubic is common choice
    - fit 3rd degree polynomial surface to pixels in neighborhood
    - can also be implemented by a convolution

Bilinear Interpolation(双线性插值)

- A simple method for resampling images



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

Correlation(相关)

template

How do we locate the template in the image?

Minimize

$$E(i,j)=\sum_m\sum_n\left[f(m,n)-t(m-i,n-j)\right]^2$$

$$=\sum_m\sum_n\left[f^2(m,n)+t^2(m-i,n-j)-2f(m,n)t(m-i,n-j)\right]$$

Maximize

$$R_{tf}(i,j)=\sum_m\sum_n t(m-i,n-j)f(m,n)$$

Cross-correlation

Correlation (相关)

- Cauchy inequality (柯西不等式)

$$a^2 + b^2 + c^2 \geq ab + bc + ca$$

R{(a,b,c), (a,b,c)}> R{(a,b,c), (b,c,a)}

$$4a^2 + 4b^2 + 4c^2 \geq a^2 + b^2 + c^2$$

R{(a,b,c), (4a,4b,4c)}> R{(a,b,c), (a,b,c)} ?

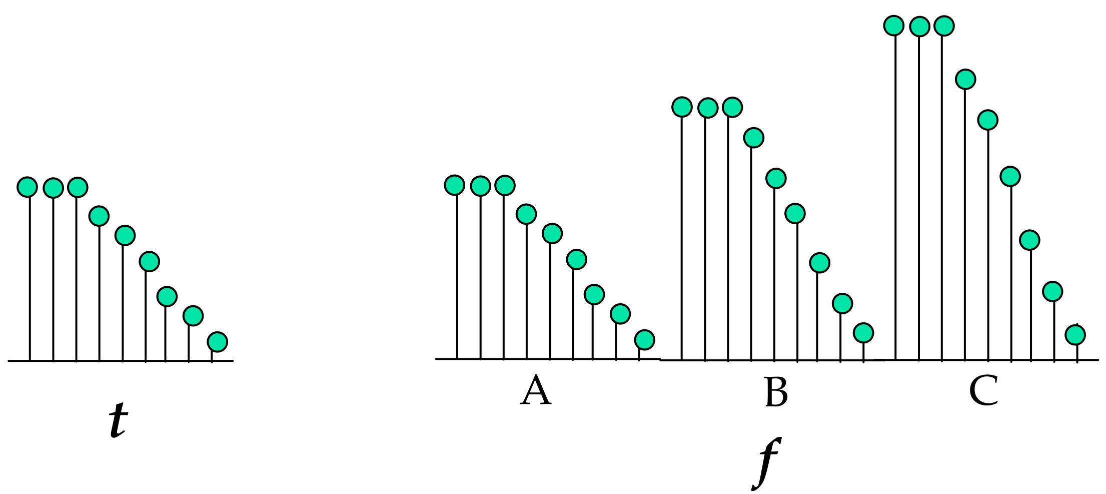
Cross-correlation (互相关)

$$R_{tf}(i,j)=\sum_m\sum_n t(m-i,n-j)f(m,n) \qquad R_{tf}=t\otimes f$$

Note: $t\otimes f \neq f\otimes t$

$$R_{ff}=f\otimes f$$ Auto-correlation

Problem:



$R_{tf}(C)>R_{tf}(B)>R_{tf}(A)$ We need $R_{tf}(A)$ to be the maximum!

Correlation

- Cauchy inequality (柯西不等式)

Corr(A,B)=dot(A,B)/sqrt(|A||B|)

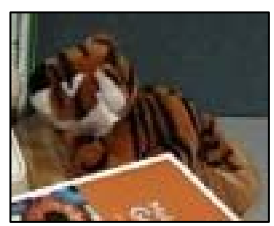
Corr{(a,b,c), (4a,4b,4c)}= Corr{(a,b,c), (a,b,c)}=1.0

Normalized Correlation

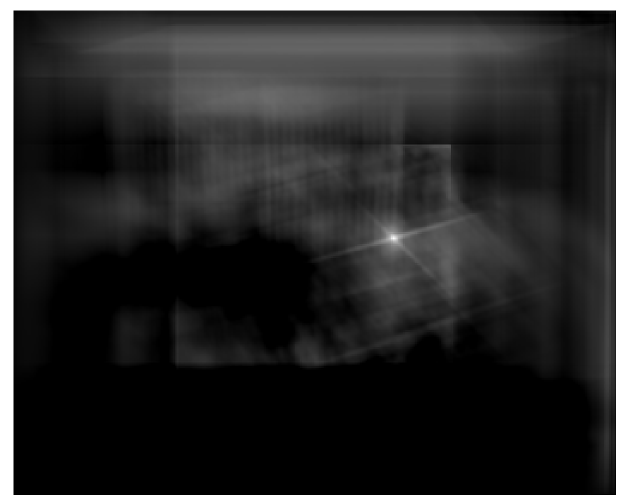
Account for energy differences

$$N_{tf}(i,j)=\frac{\sum_m\sum_n t(m-i,n-j)f(m,n)}{\left[\sum_m\sum_n t^2(m-i,n-i)\right]^{1/2}\left[\sum_m\sum_n f^2(m,n)\right]^{1/2}}$$

$\otimes$



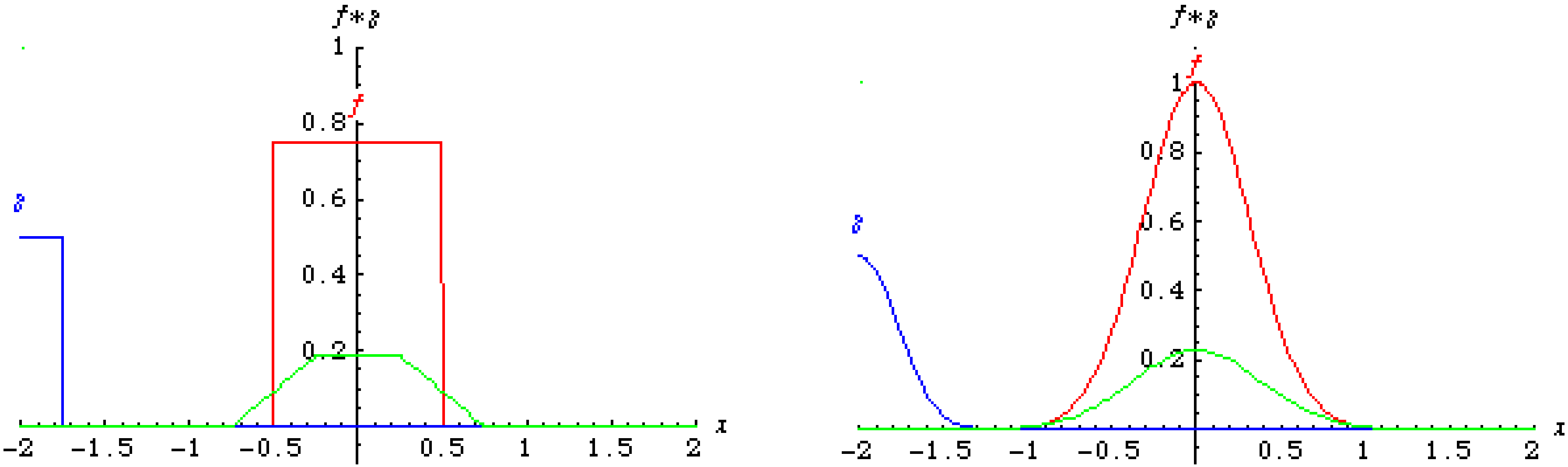
$=$



Normalized Correlation

```
onion = imread('onion.png');
peppers = imread('peppers.png');
imshow(onion);
figure, imshow(peppers);
rect_onion = [111 33 65 58];
rect_peppers = [163 47 143 151];
sub_onion = imcrop(onion,rect_onion);
sub_peppers = imcrop(peppers,rect_peppers);
c = normxcorr2(sub_onion(:,:,1),sub_peppers(:,:,1));
[max_c, imax] = max(abs(c(:)));
[ypeak, xpeak] = ind2sub(size(c),imax(1));
corr_offset = [(xpeak-size(sub_onion,2))；(ypeak-size(sub_onion,1))];
rect_offset = [(rect_peppers(1)-rect_onion(1))；(rect_peppers(2)-rect_onion(2))];
offset = corr_offset + rect_offset;
xoffset = offset(1);
yoffset = offset(2);
xbegin = round(xoffset+1);
xend   = round(xoffset+ size(onion,2));
ybegin = round(yoffset+1);
yend   = round(yoffset+size(onion,1));
extracted_onion = peppers(ybegin:yend,xbegin:xend,:);
recovered_onion = uint8(zeros(size(peppers)));
recovered_onion(ybegin:yend,xbegin:xend,:) = onion;
[m,n,p] = size(peppers);
mask = ones(m,n);
i = find(recovered_onion(:,:,1)==0);
mask(i) = .2;
figure, imshow(peppers(:,:,1))；
hold on；
h = imshow(recovered_onion);
set(h,'AlphaData',mask);
```

Convolution(卷积)



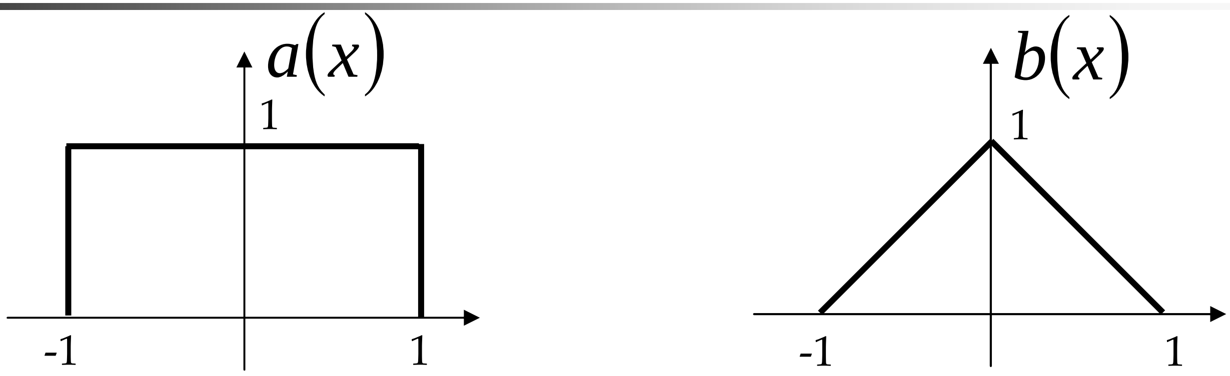
f

g

$f * g$

Eric Weisstein's Math World

Example of Convolution



$$c = a * b$$

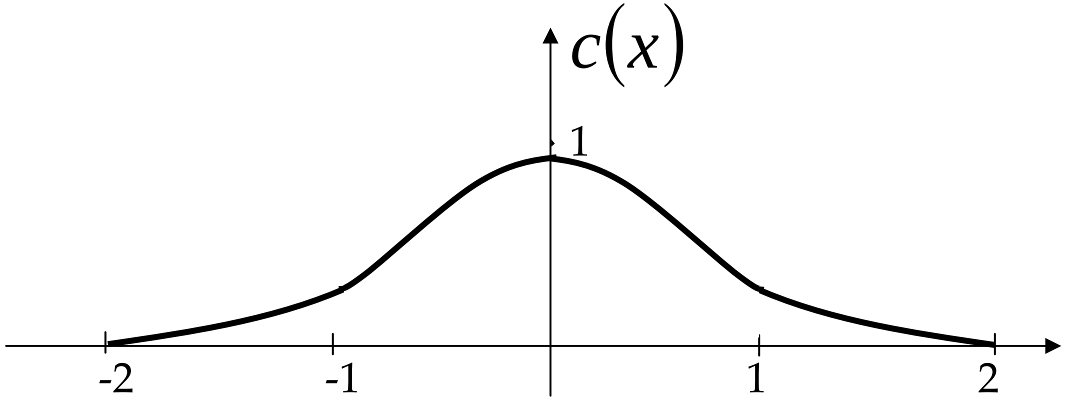
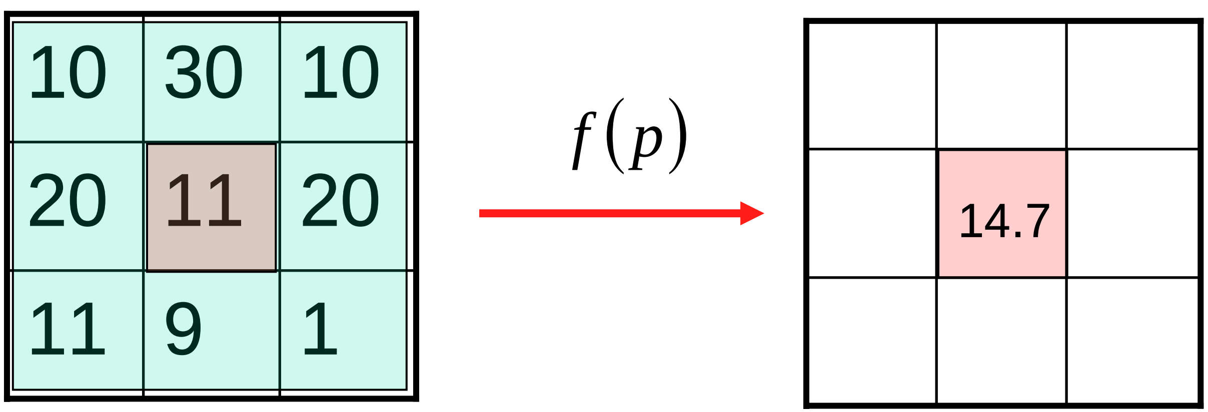


Image Filtering (Discrete)

Image Filtering

- Modify pixels based on the neighborhood



Linear Filtering

- The output is the linear combination of the neighborhood pixels
- Weighted Sum(加权和)

1	3	0
2	10	2
4	1	1

Image

$\otimes$

1	0	-1
2	0	-2
1	0	-1

Kernel

=

1	0	0
4	4	-4
4	0	-1

Filter Output

convolution

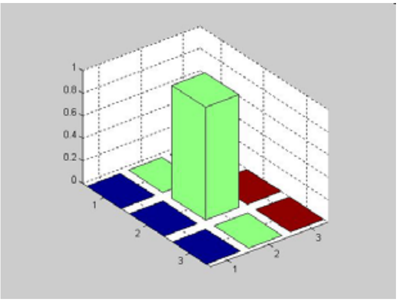
Average Filter(平均濾波器)

- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.

F

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Filtering Examples

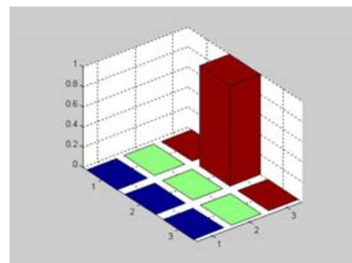


*

0	0	0
0	1	0
0	0	0

=

Filtering Examples

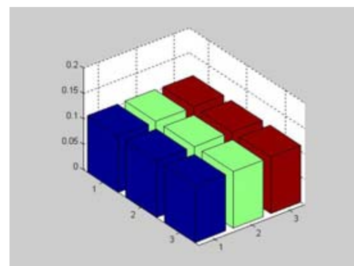


*

0	0	0
0	0	1
0	0	0

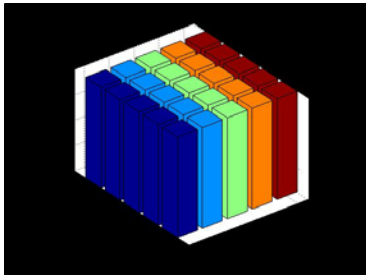
=

Filtering Examples



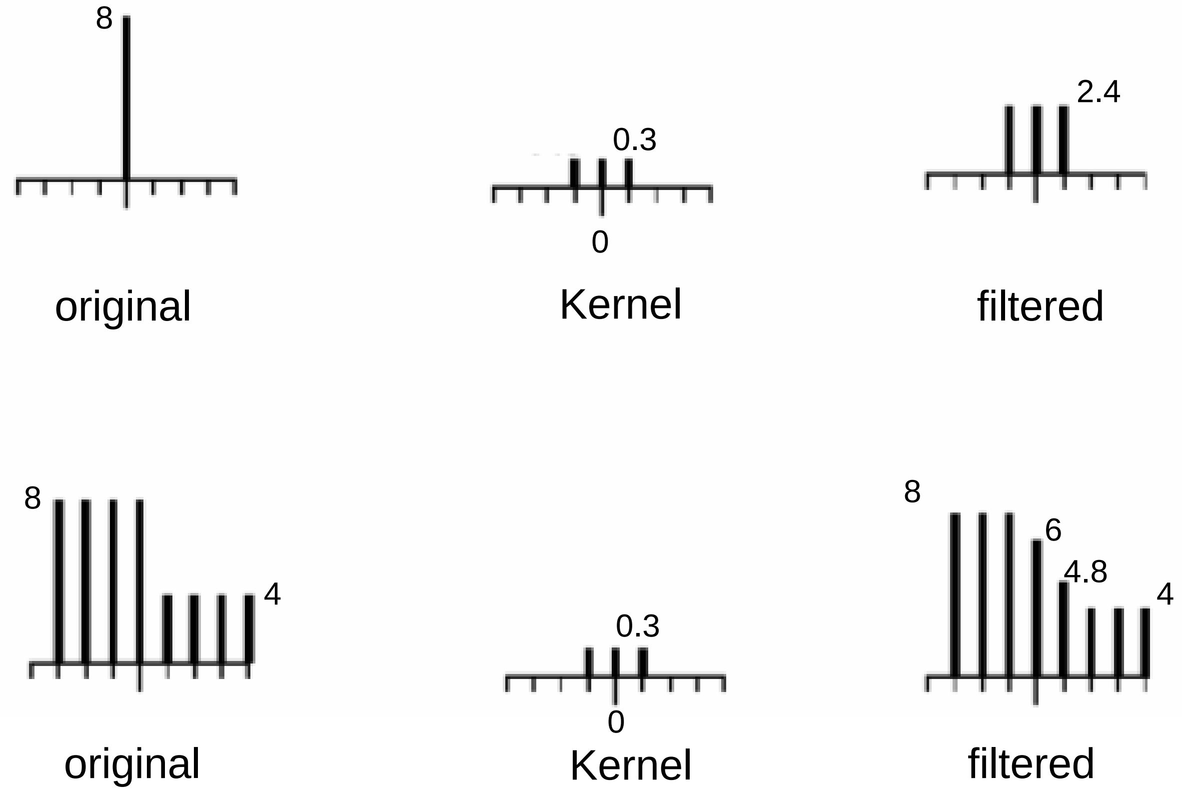
$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Filtering Examples

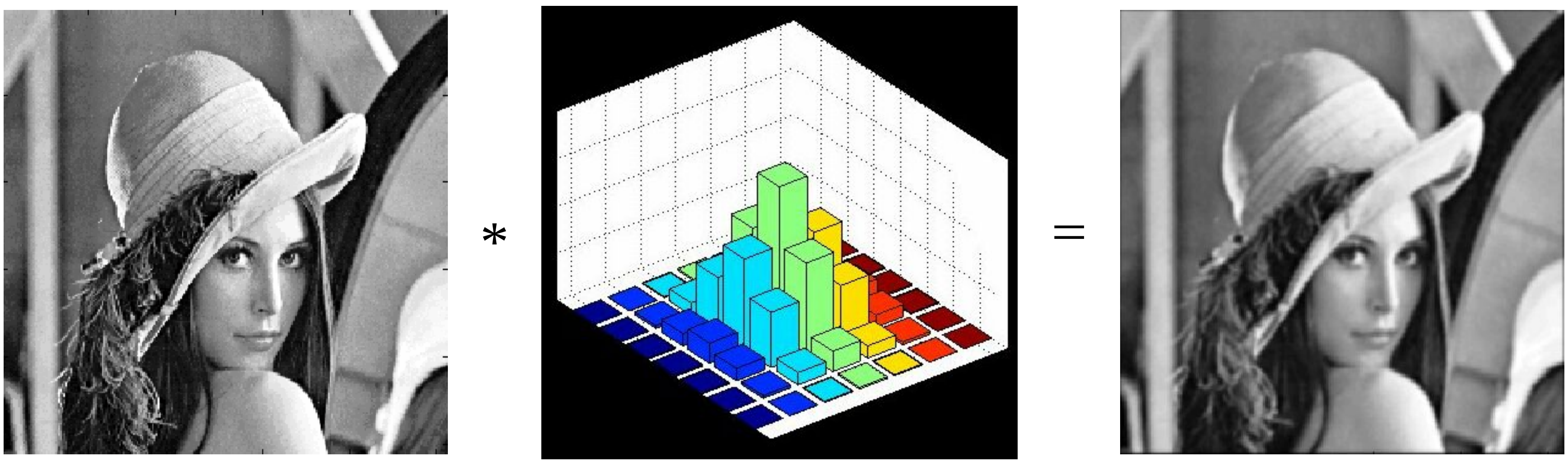


$$* \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

Blurring Examples



Gaussian Filter(高斯滤波器)



Gaussian vs. Averaging

Gaussian Smoothing

Smoothing by Averaging

Noise Filtering(噪声过滤)

Gaussian Noise

After Averaging

After Gaussian Smoothing

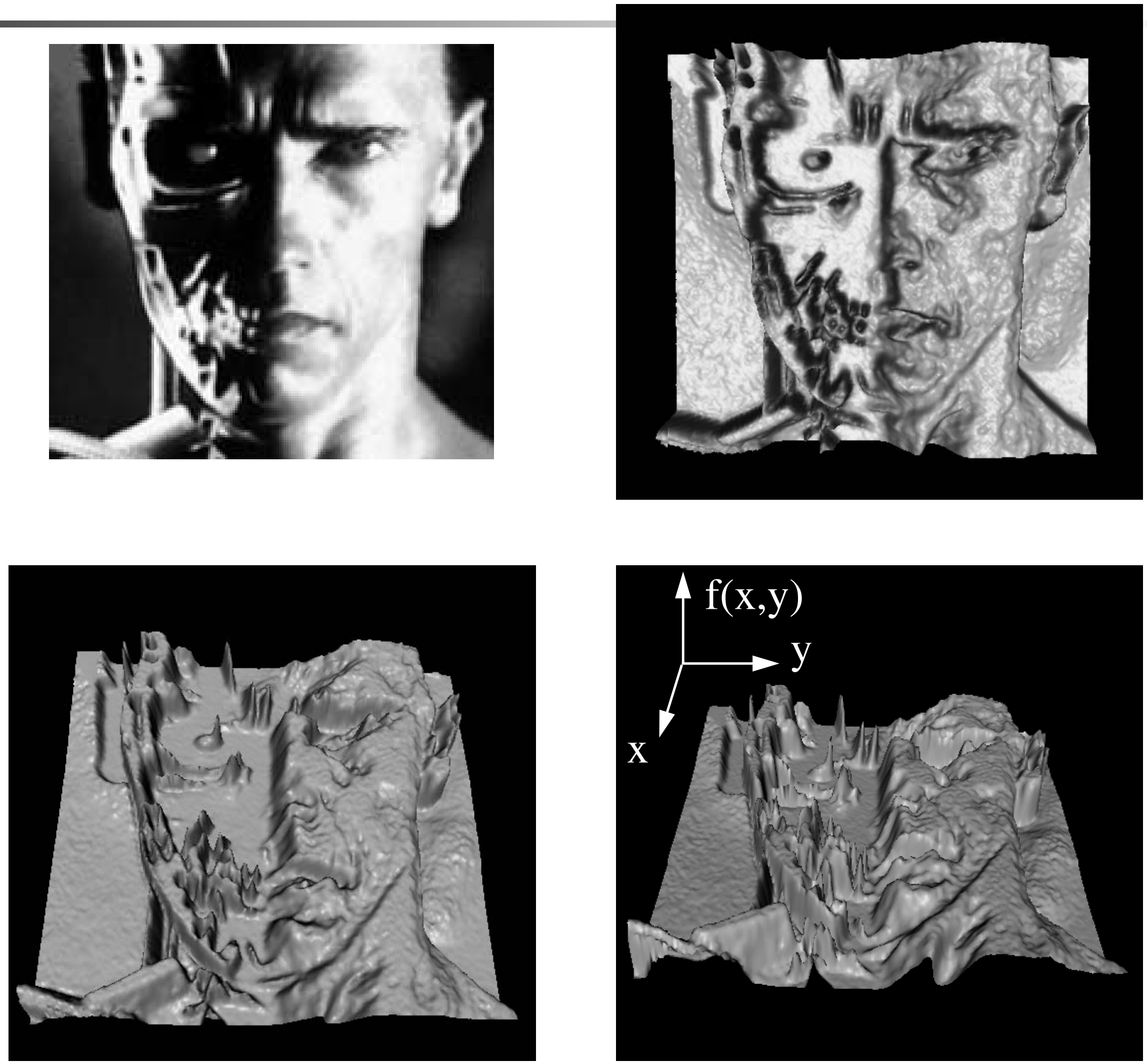
Noise Filtering

Salt & Pepper Noise

After Averaging

After Gaussian Smoothing

Image as a Function



Digital Images

The scene is

- **projected** on a 2D plane,
- **sampled** on a regular grid, and each sample is
- **quantized** (rounded to the nearest integer)

$$f(i, j) = \text{Quantize}\{f(i\Delta, j\Delta)\}$$

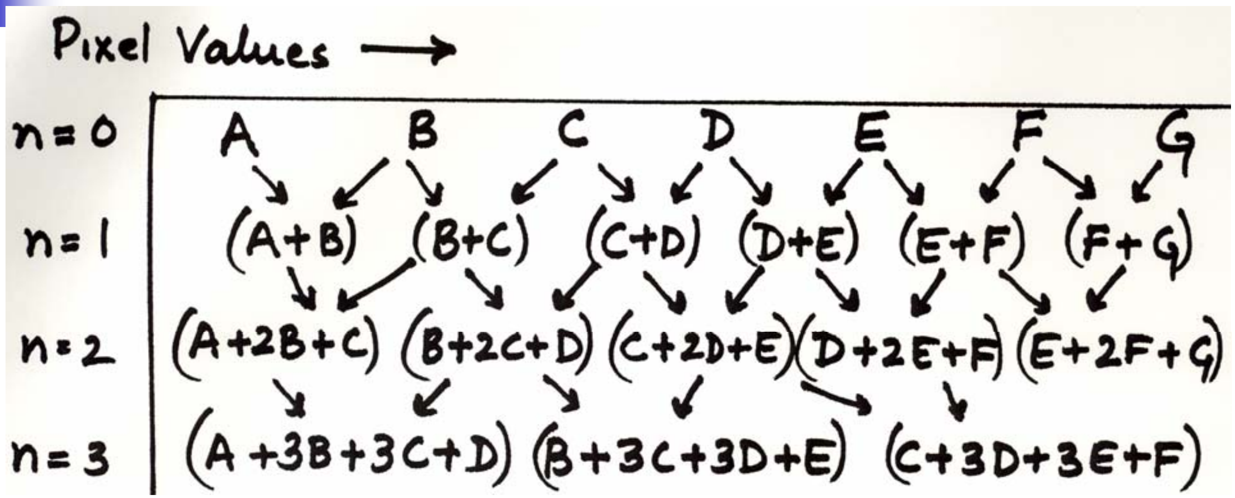
Image as a matrix

j

i

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Averaging



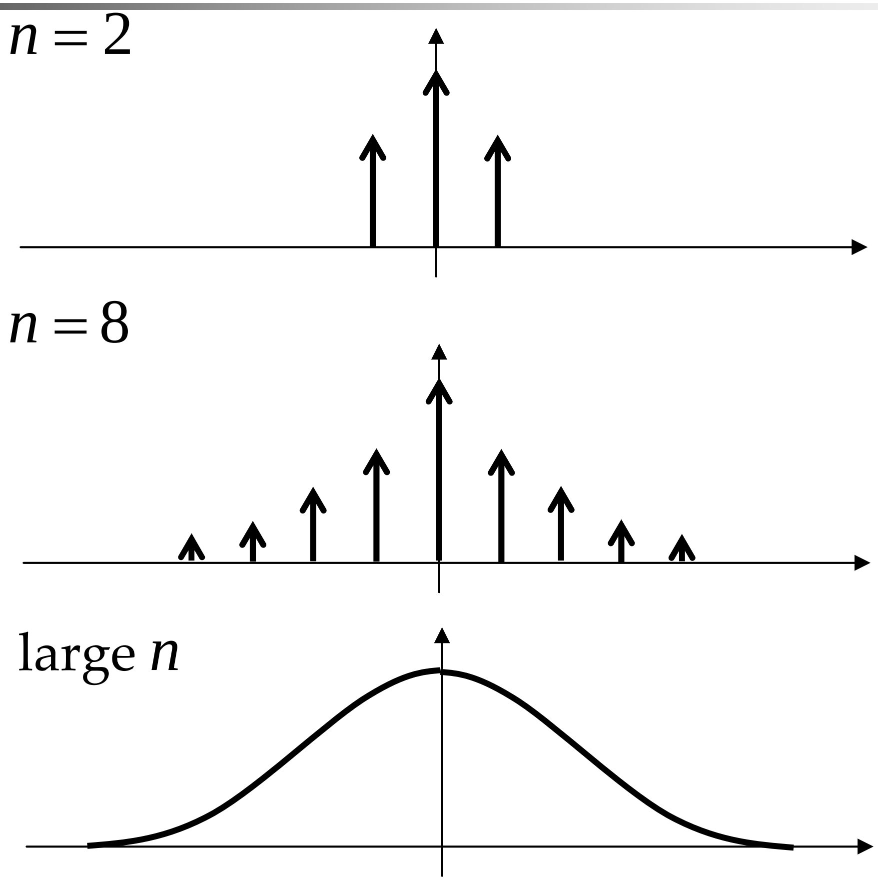
For *n*=2, convolve pixel values with $\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline\end{array}$

2D images:

(a) use $\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline\end{array}$ then $\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline\end{array}$ or (b) use $\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline\end{array} * \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$

Which is faster? $(a)\, O(2(n+1))$ $(b)\, O((n+1)^2)$

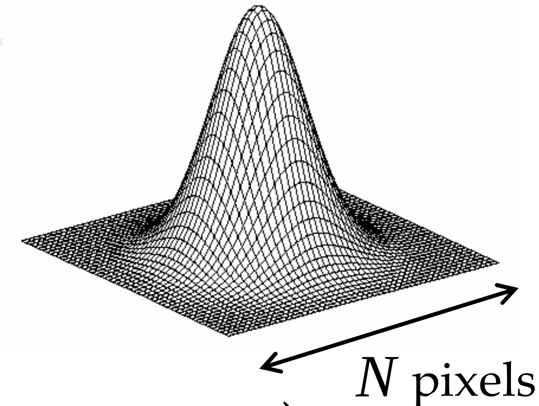
Averaging By Convolution



Repeated averaging $\approx$ Gaussian smoothing

Gaussian Smoothing(高斯平滑)

Gaussian kernel $h(i,j)=\frac{1}{2\pi\sigma^2}e^{-\frac{1}{2}\left(\frac{i^2+j^2}{\sigma^2}\right)}$



Filter size $N \propto \sigma$ …can be very large

(truncate, if necessary)

$$g(i,j)=\frac{1}{2\pi\sigma^2}\sum_{m=1}\sum_{n=1}e^{-\frac{1}{2}\left(\frac{m^2+n^2}{\sigma^2}\right)}f(i-m,j-n)$$

2D Gaussian is separable!

$$g(i,j)=\frac{1}{2\pi\sigma^2}\sum_{m=1}e^{-\frac{1}{2}\frac{m^2}{\sigma^2}}\sum_{n=1}e^{-\frac{1}{2}\frac{n^2}{\sigma^2}}f(i-m,j-n)$$

Use 2 1D Gaussian filters

Image Border

- Ignore
  - Output image will be smaller than original
- Pad with constant values
  - Can introduce substantial 1st order derivative values
- Pad with reflection
  - Can introduce substantial 2nd order derivative values

Gaussian Smoothing

original

$\sigma = 2$

$\sigma = 2.8$

$\sigma = 4$

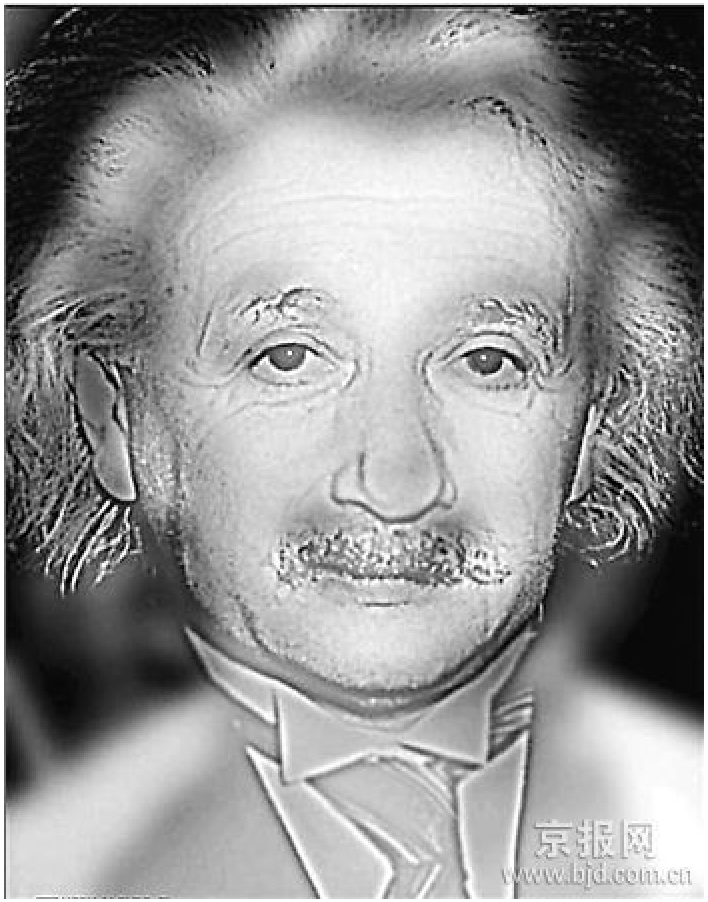
Gaussian Smoothing

by Charles Allen Gillbert

by Harmon & Julesz

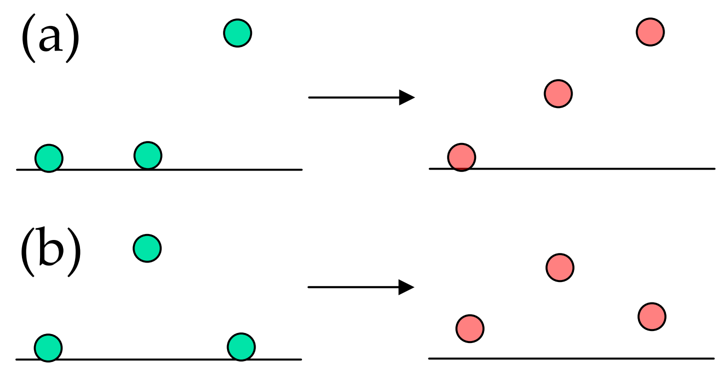
http://www.michaelbach.de/ot/cog_blureffects/index.html

More interesting examples

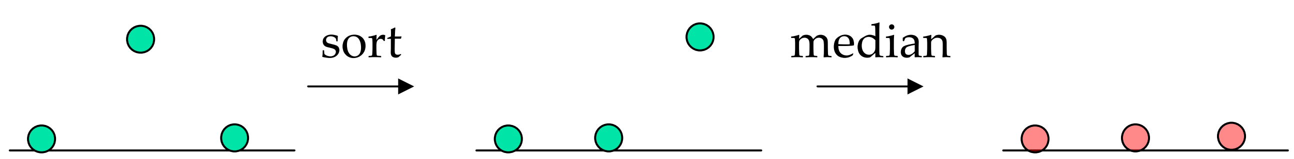


Median Filter(中值滤波)

- Smoothing is averaging
  - (a) Blurs edges
  - (b) Sensitive to outliers



- Median filtering
  - Sort N^2-1 values around the pixel
  - Select middle value (median)



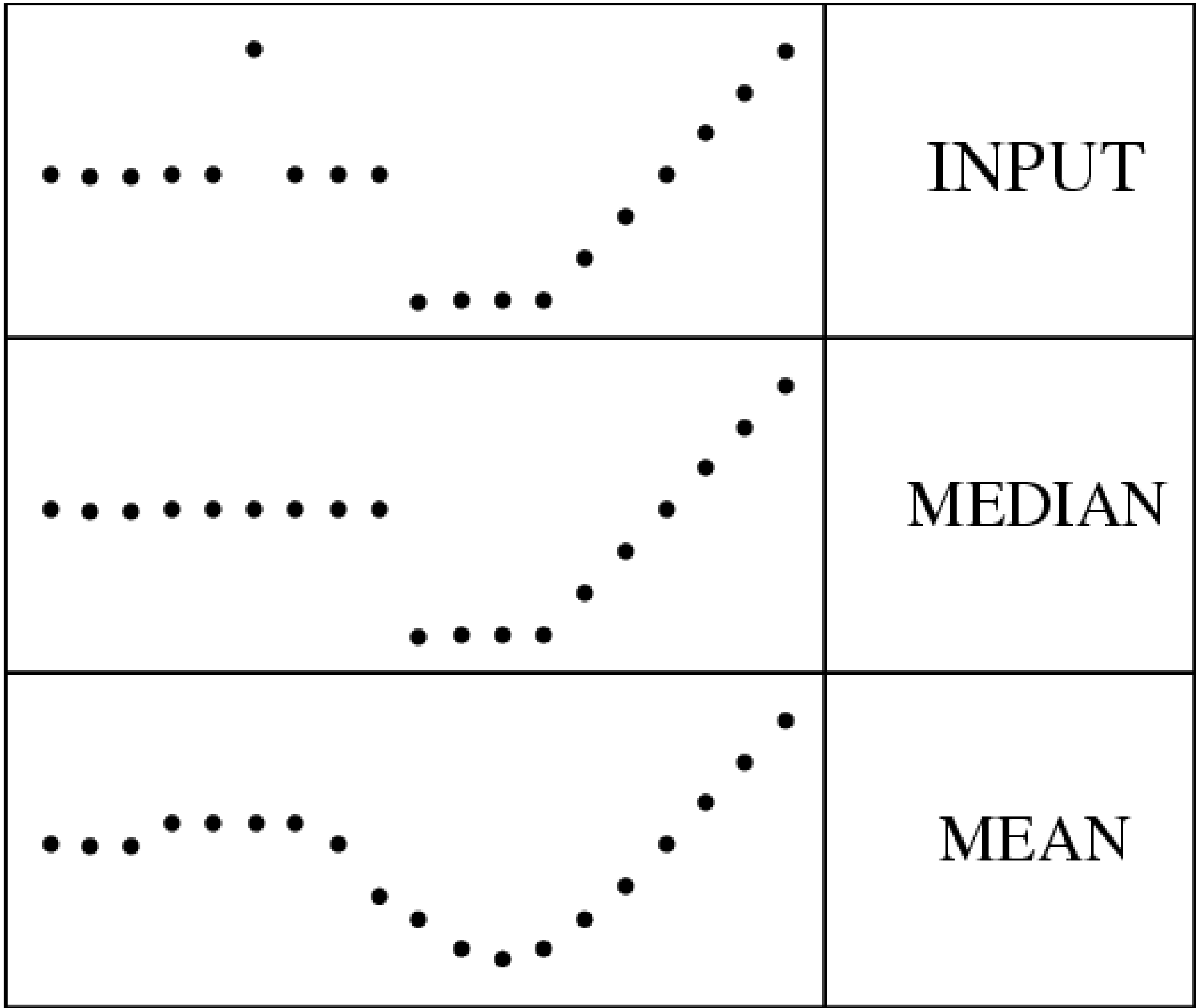
  - Non-linear (Cannot be implemented with convolution)

Median filters : principle

- non-linear filter
- method :
  - rank-order neighbourhood intensities
  - take middle value
- no new grey levels emerge...

Median filters : example

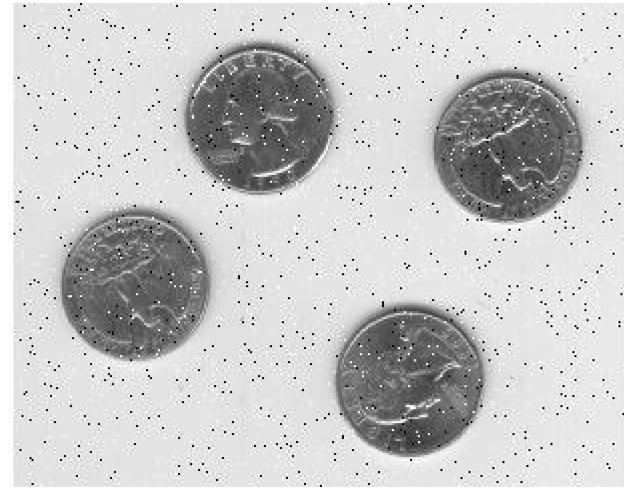
filters of width 5 :



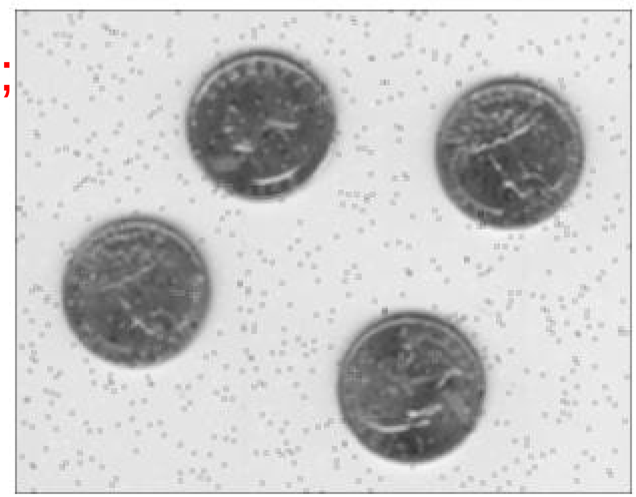
Median filters : discussion

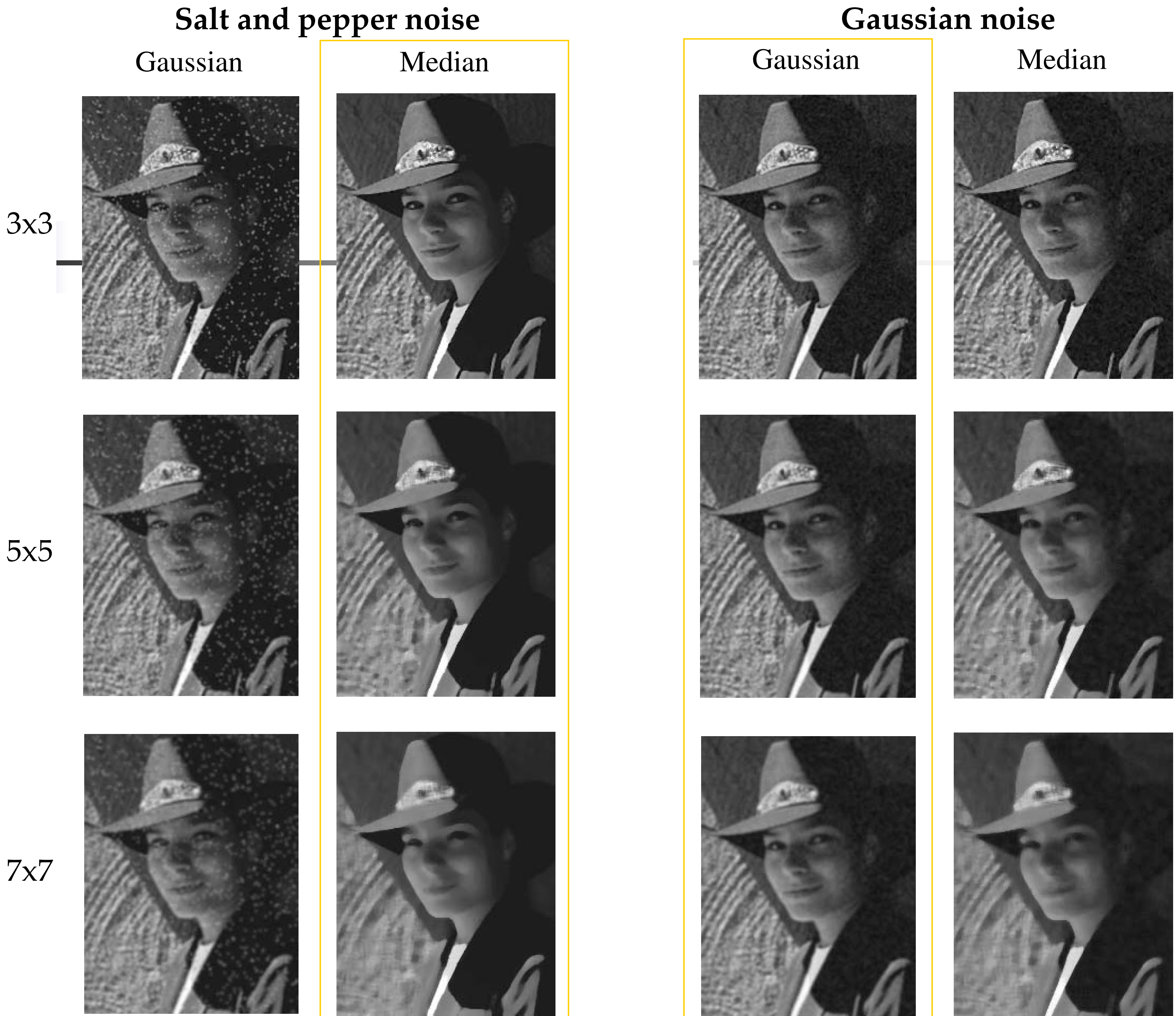
- median completely discards the spike,
- linear filter always responds to all aspects
- median filter preserves discontinuities,
- linear filter produces rounding-off effects
- DON'T become all too optimistic

Median filters



```
I = imread('eight.tif');
imshow(I);
J = imnoise(I,'salt & pepper',0.02);
figure, imshow(J);
K = filter2(fspecial('average',3),J)/255;
figure, imshow(K);
L = medfilt2(J,[3 3]);
figure, imshow(L);
```




Salt and pepper noise
Gaussian
Median
Gaussian noise
Gaussian
Median
3x3
5x5
7x7

Convolution(卷积)

- A **convolution** operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

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

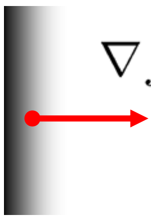
- It is written:

$$G = H \star F$$

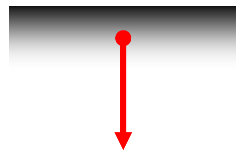
- Suppose H is a Gaussian or mean kernel. How does convolution differ from cross-correlation?

Image gradient(梯度)

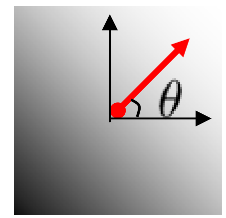
- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- The gradient points in the direction of most rapid change in intensity



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$



$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$



$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

- The gradient direction is given by:

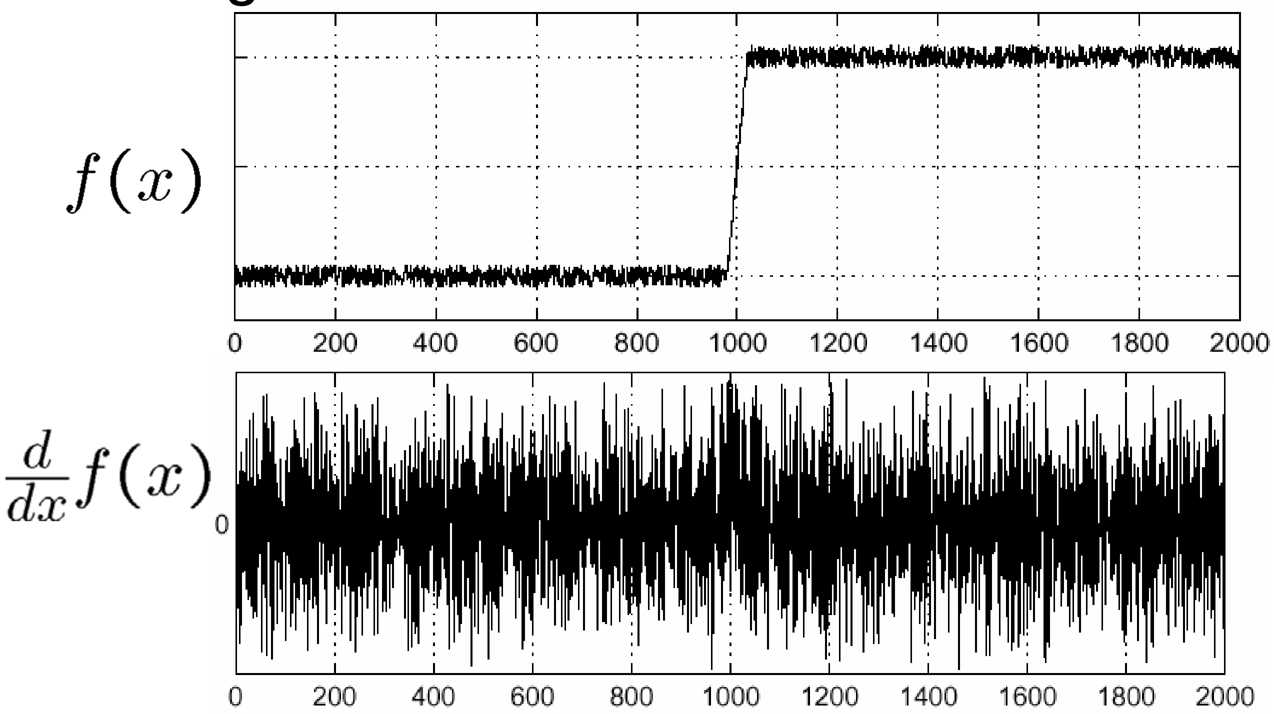
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

  - how does this relate to the direction of the edge?
- The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Effects of noise(噪声的影响)

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



- Where is the edge?

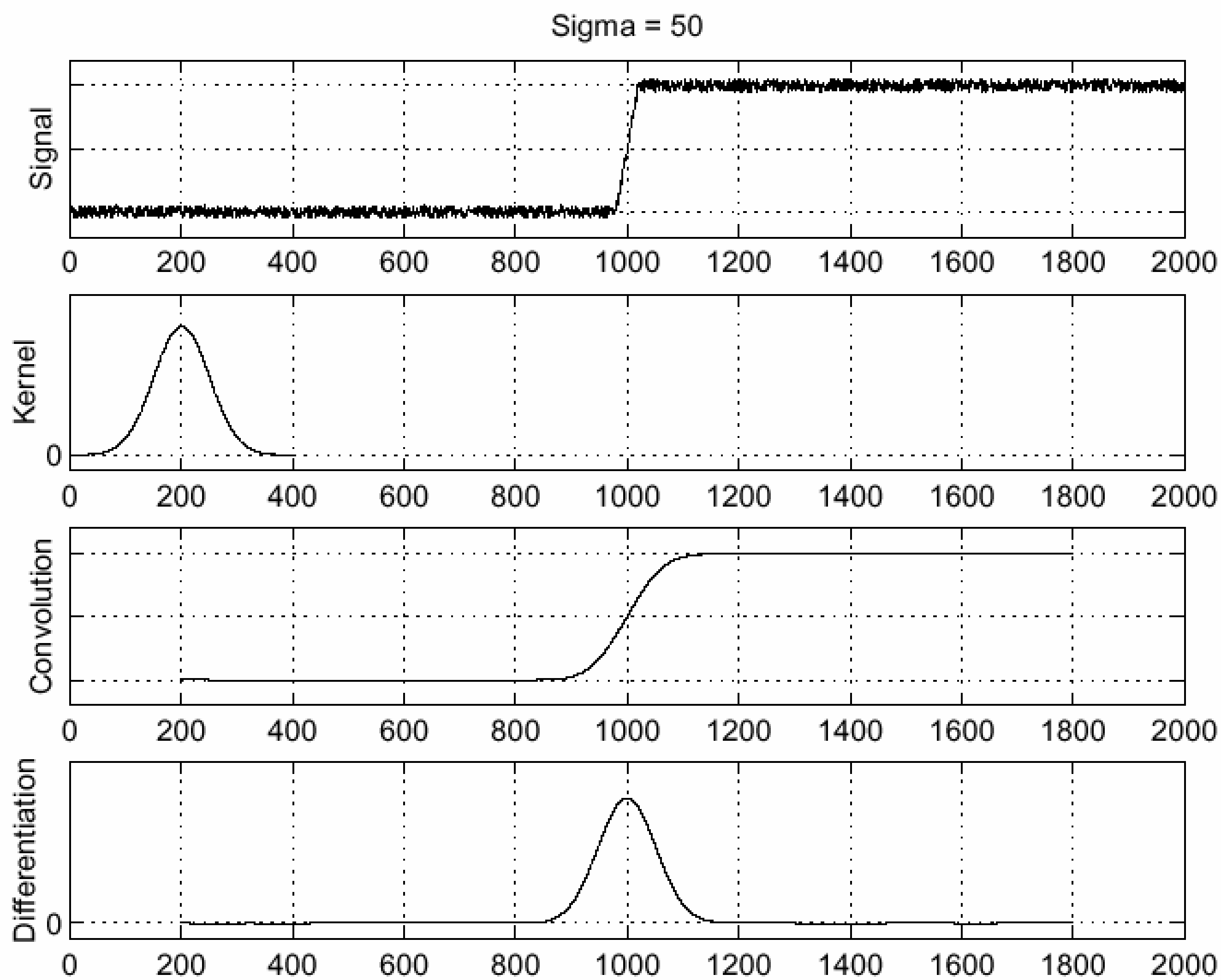
Solution: smooth first

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

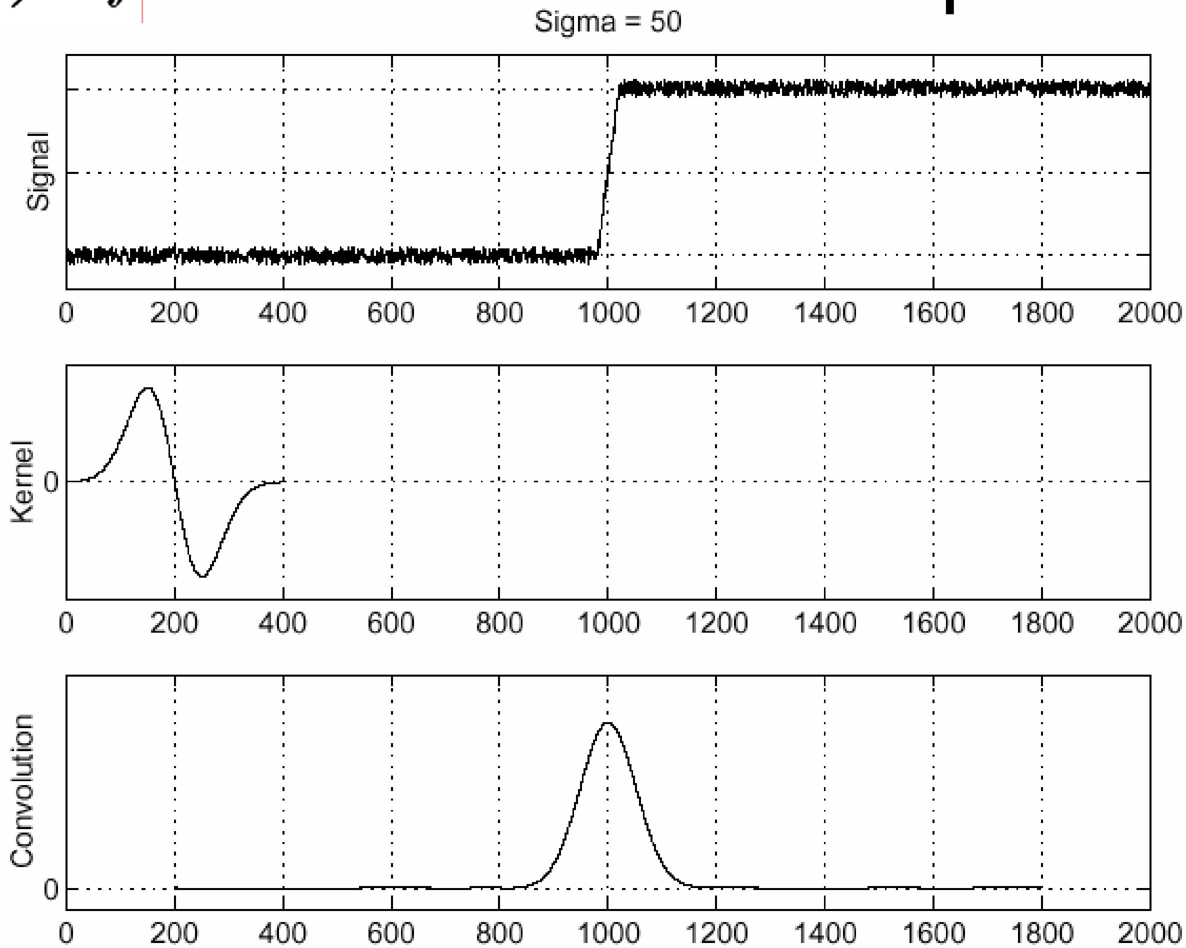
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

This saves us one operation:

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$

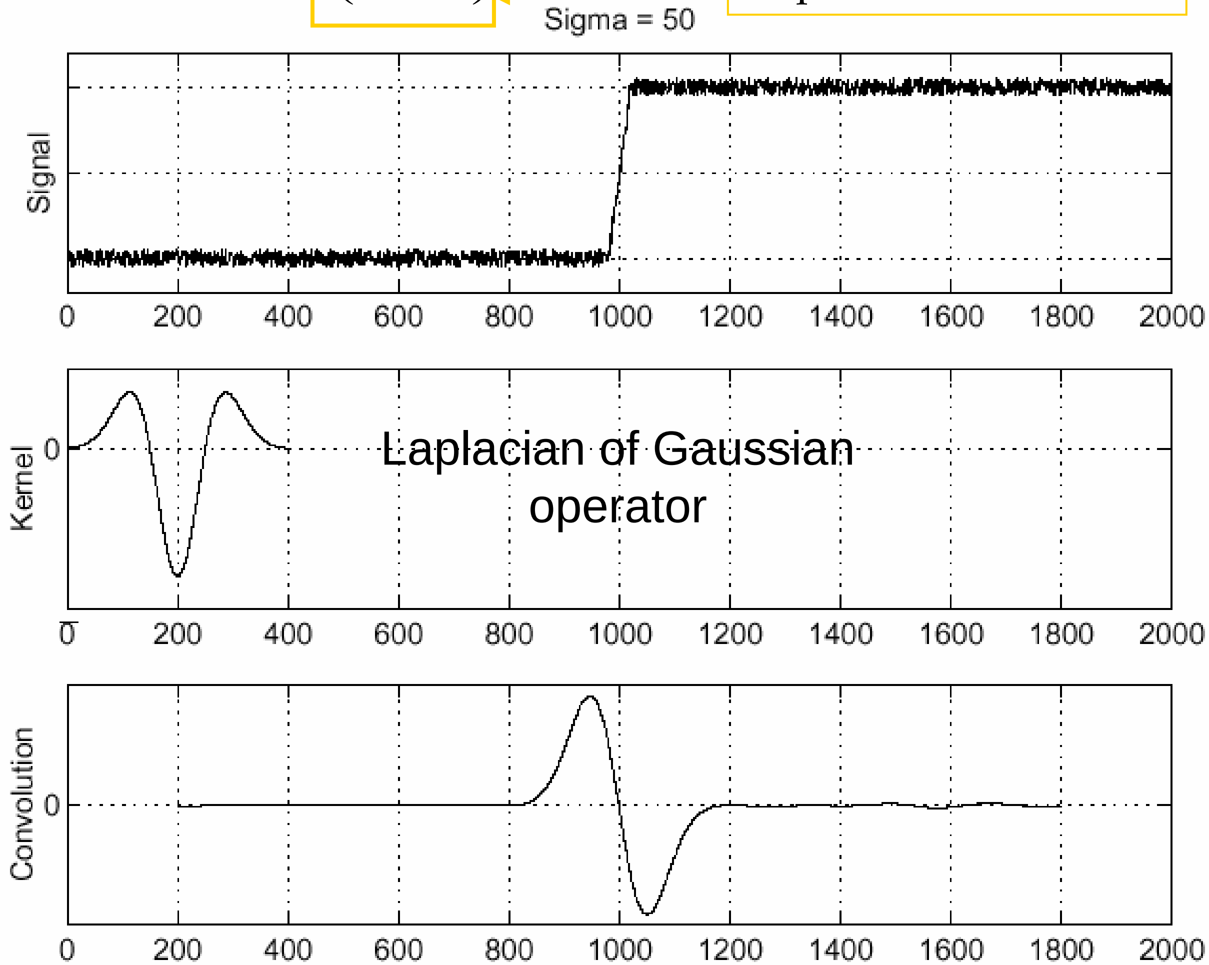


Laplacian of Gaussian

$$\frac{\partial^2}{\partial x^2}(h * f) = \boxed{\left(\frac{\partial^2}{\partial x^2}h\right)} * f$$

Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$



f

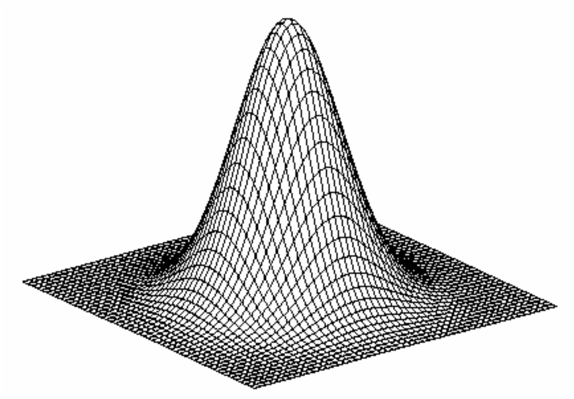
$\frac{\partial^2}{\partial x^2}h$

$(\frac{\partial^2}{\partial x^2}h) \star f$

Where is the edge?

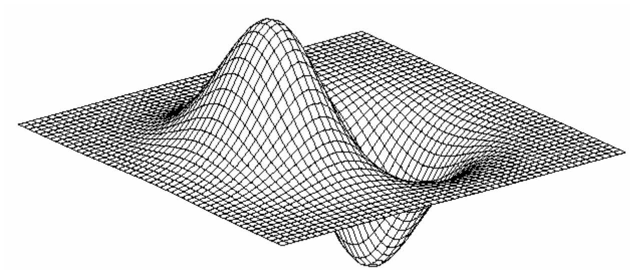
Zero-crossings of bottom graph

2D edge detection filters



Gaussian

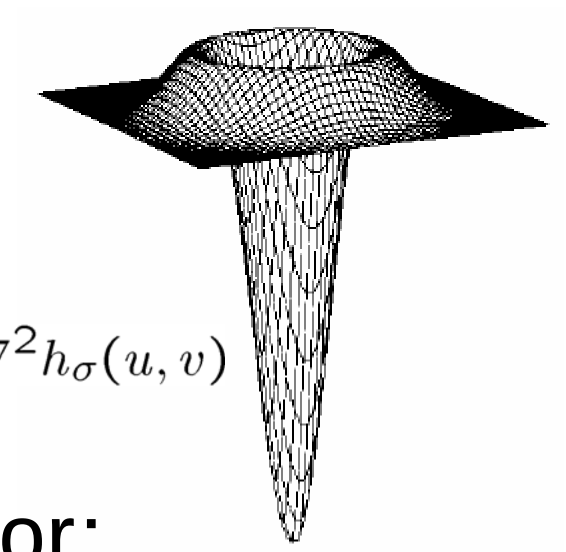
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian



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

- ∇^2 is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Edge detection by subtraction

original

Edge detection by subtraction

smoothed (5x5 Gaussian)

Edge detection by subtraction

smoothed – original
(scaled by 4, offset +128)

Why does this work?

filter demo

original

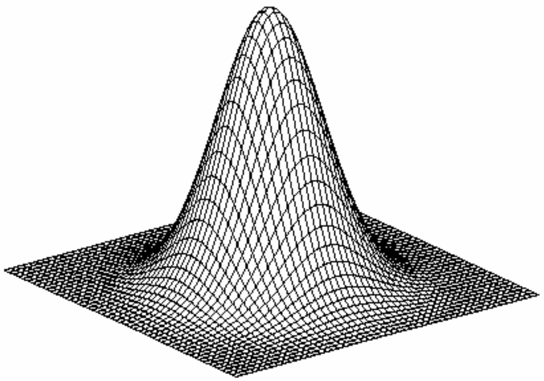
smoothed (5x5 Gaussian)

smoothed – original

(scaled by 4, offset +128)

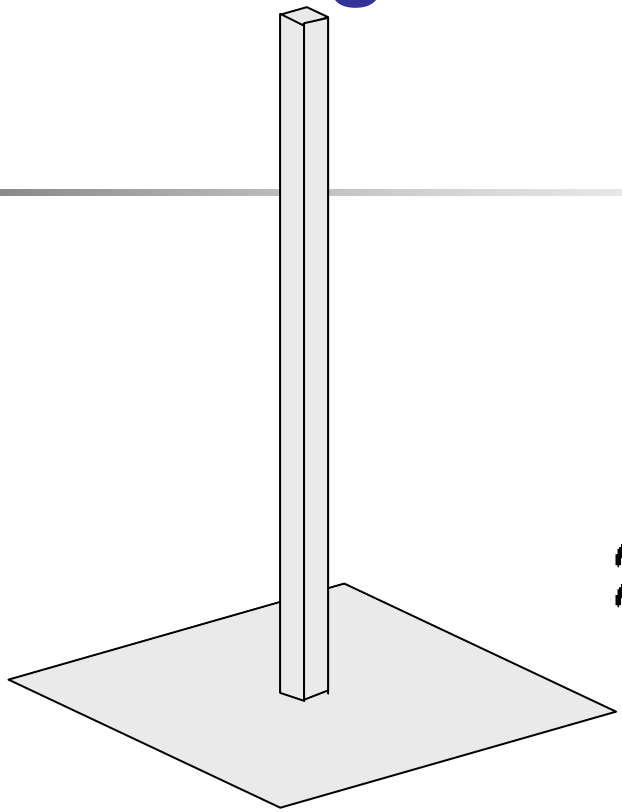
why does
this work?

Gaussian - image filter



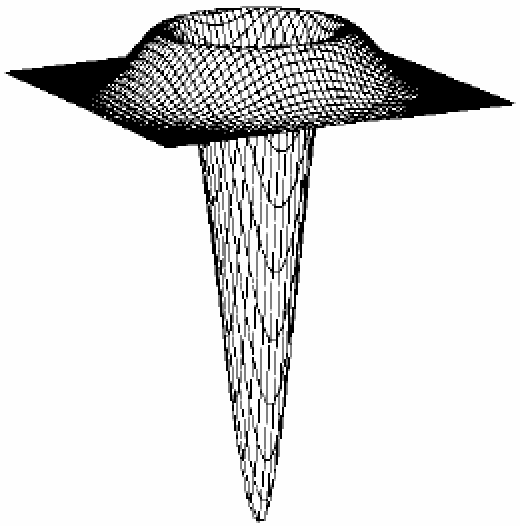
Gaussian

$-$



delta function

$\approx$



Laplacian of Gaussian

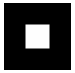
Average Filter

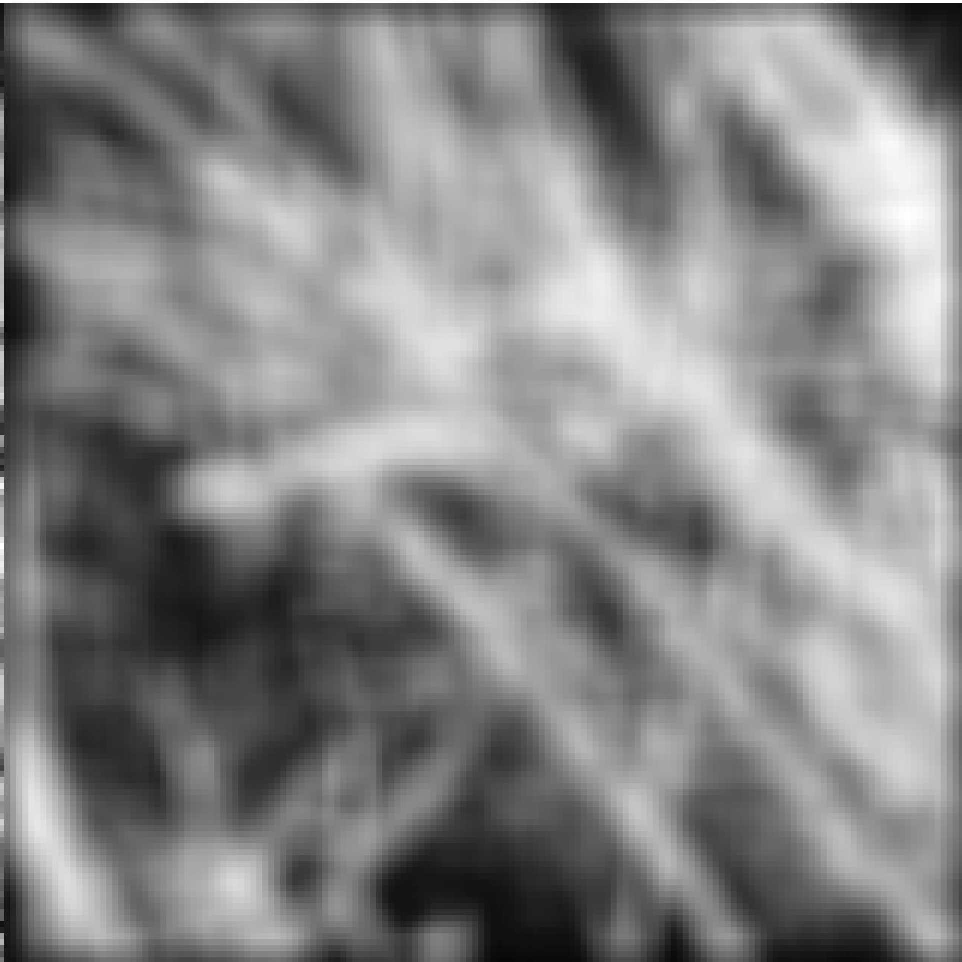
- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.

F

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

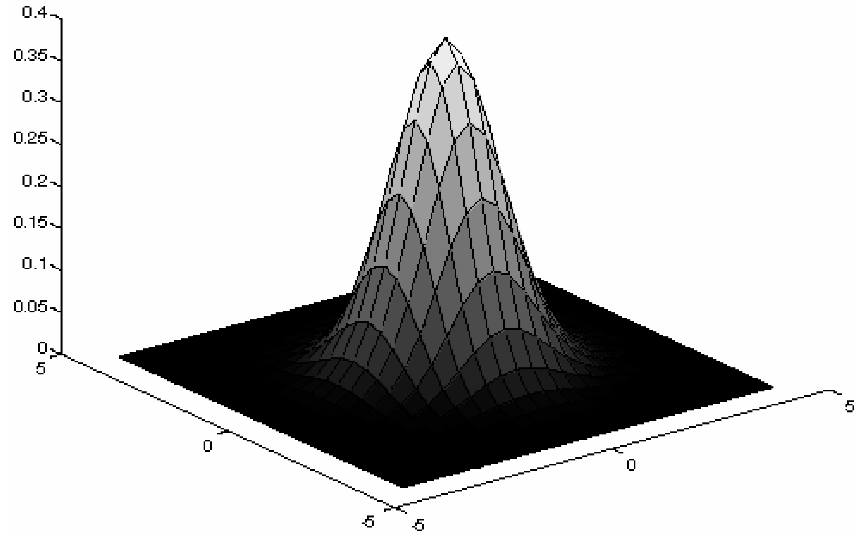
Example: Smoothing by Averaging





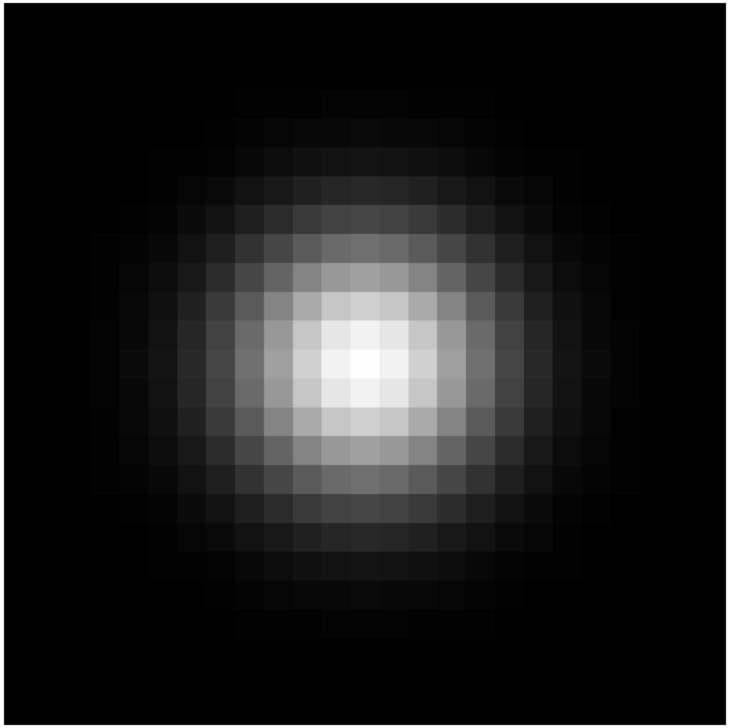
Gaussian Averaging(高斯平均)

- Rotationally symmetric.
- Weights nearby pixels more than distant ones.
  - This makes sense as probabalistic inference.



- A Gaussian gives a good model of a fuzzy blob

An Isotropic Gaussian

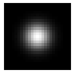


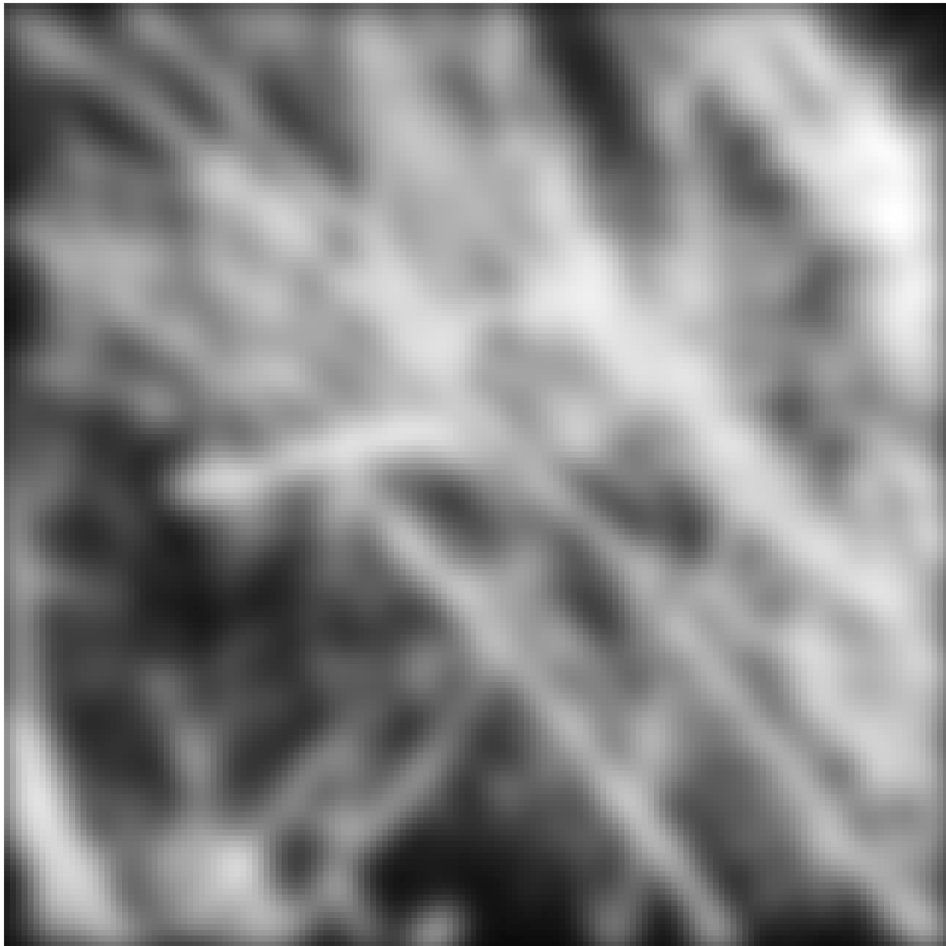
- The picture shows a smoothing kernel proportional to

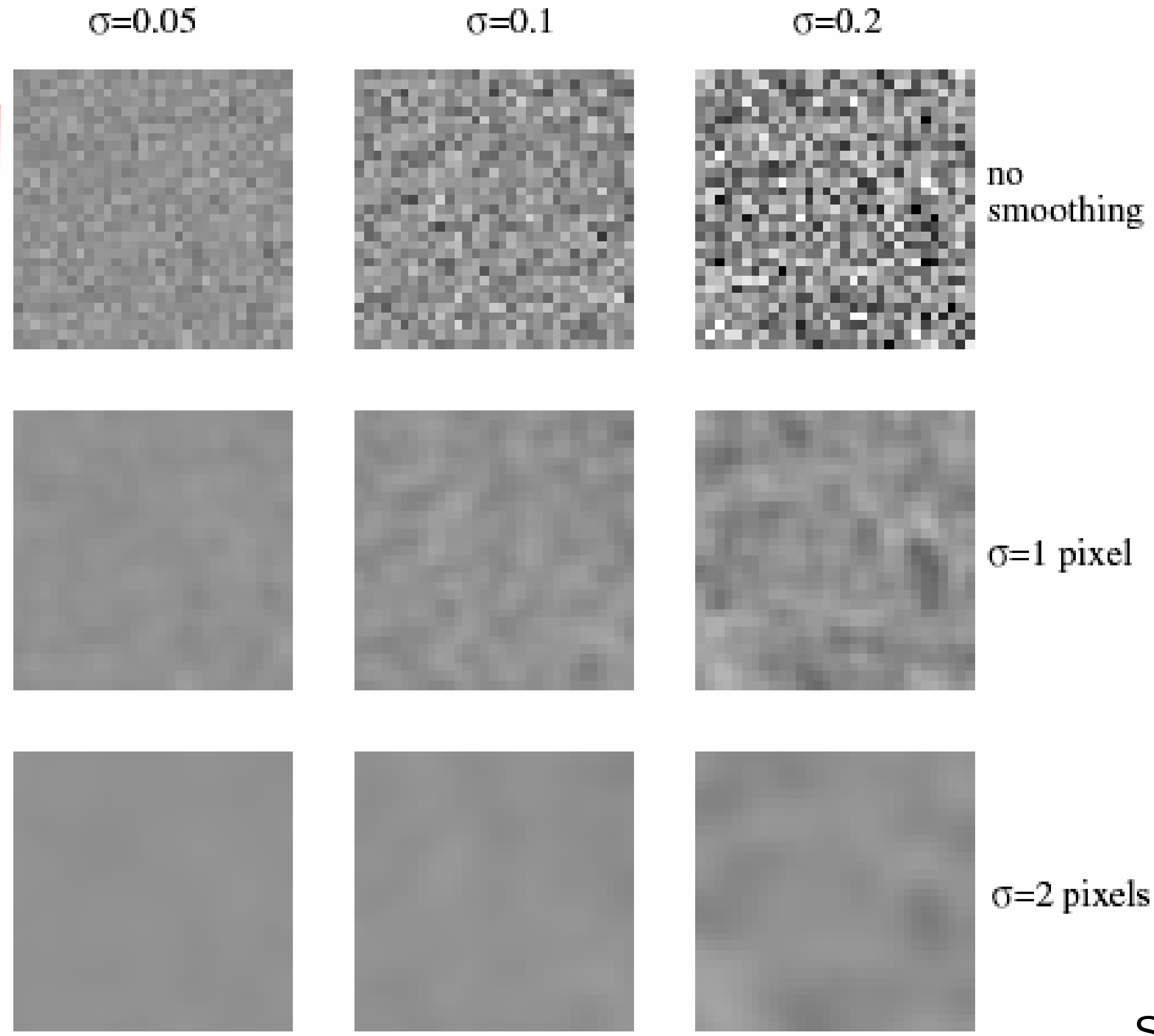
$$\exp\left(-\left(\frac{x^2+y^2}{2\sigma^2}\right)\right)$$

(which is a reasonable model of a circularly symmetric fuzzy blob)

Smoothing with a Gaussian







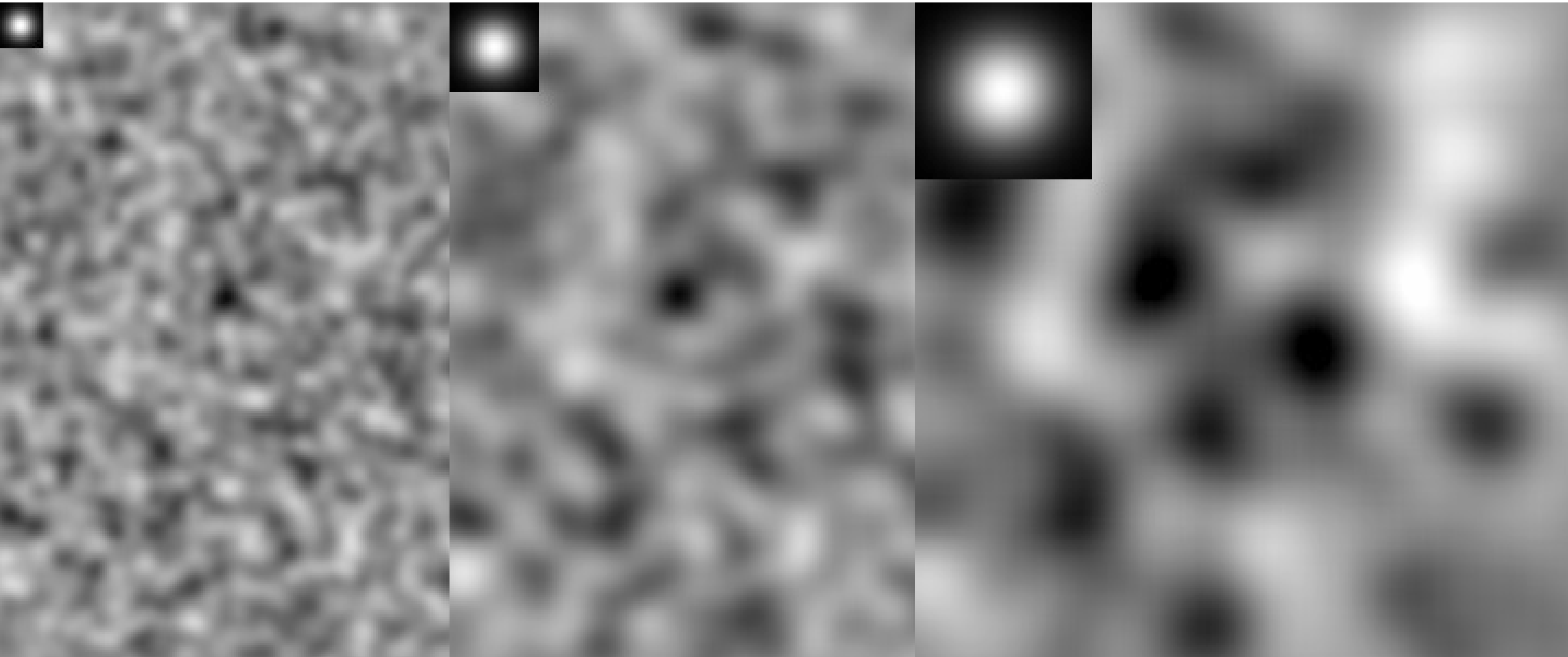
Noise

The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.

Slide credit: Marc Pollefeys

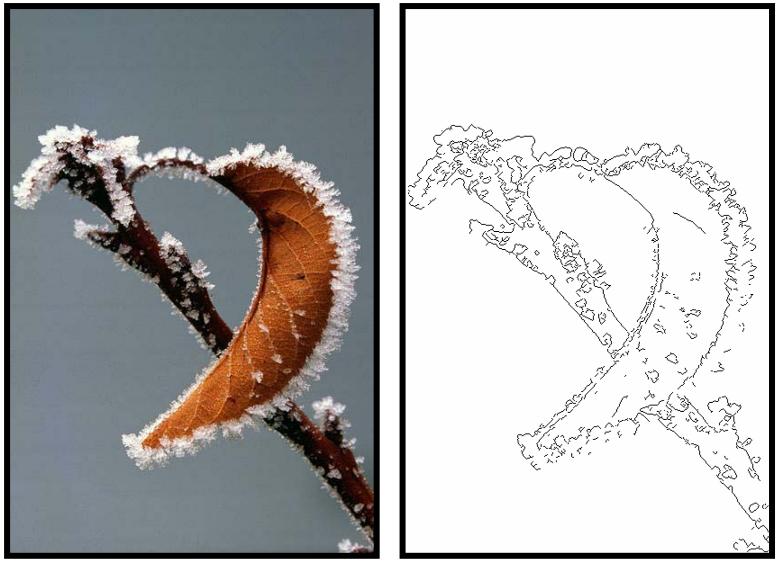
Filter responses are correlated

- Filtered noise is sometimes useful
  - looks like some natural textures, can be used to simulate fire, etc.



Edge Detection (Break)

Example

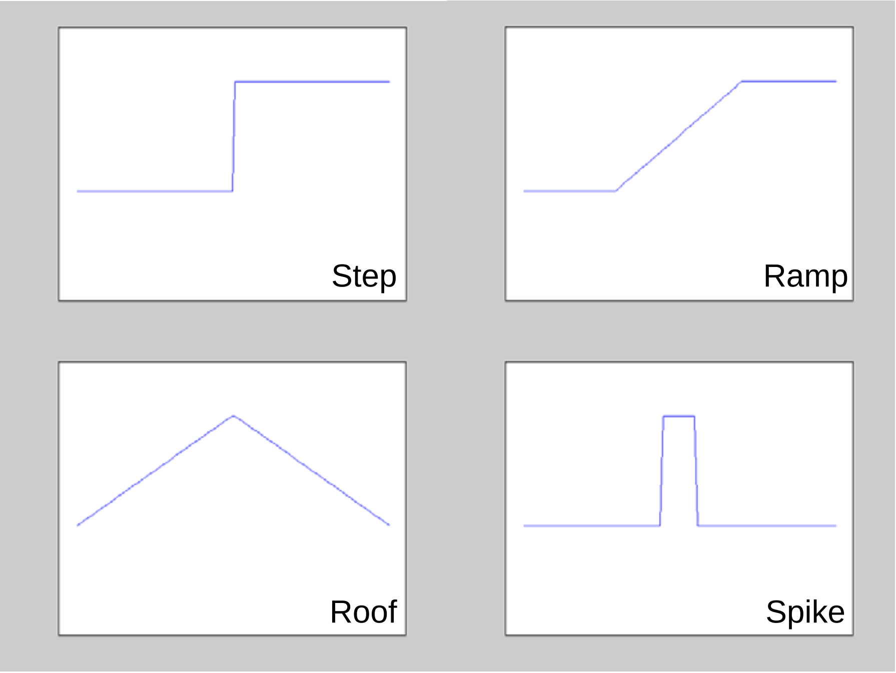


Edge Detection(边缘检测)

- What is an Edge?
- How can we find it?

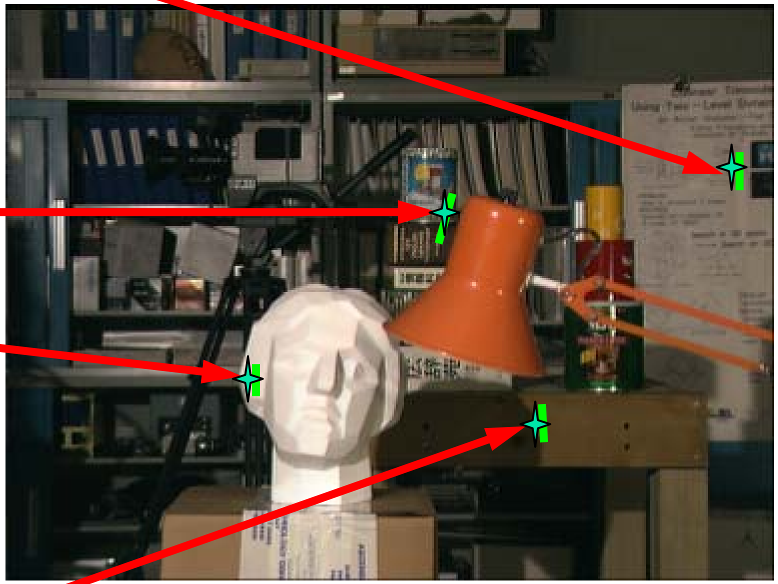
What is an Edge?

- Discontinuity of intensities in the image
- Edge models
  - Step
  - Roof
  - Ramp
  - Spike

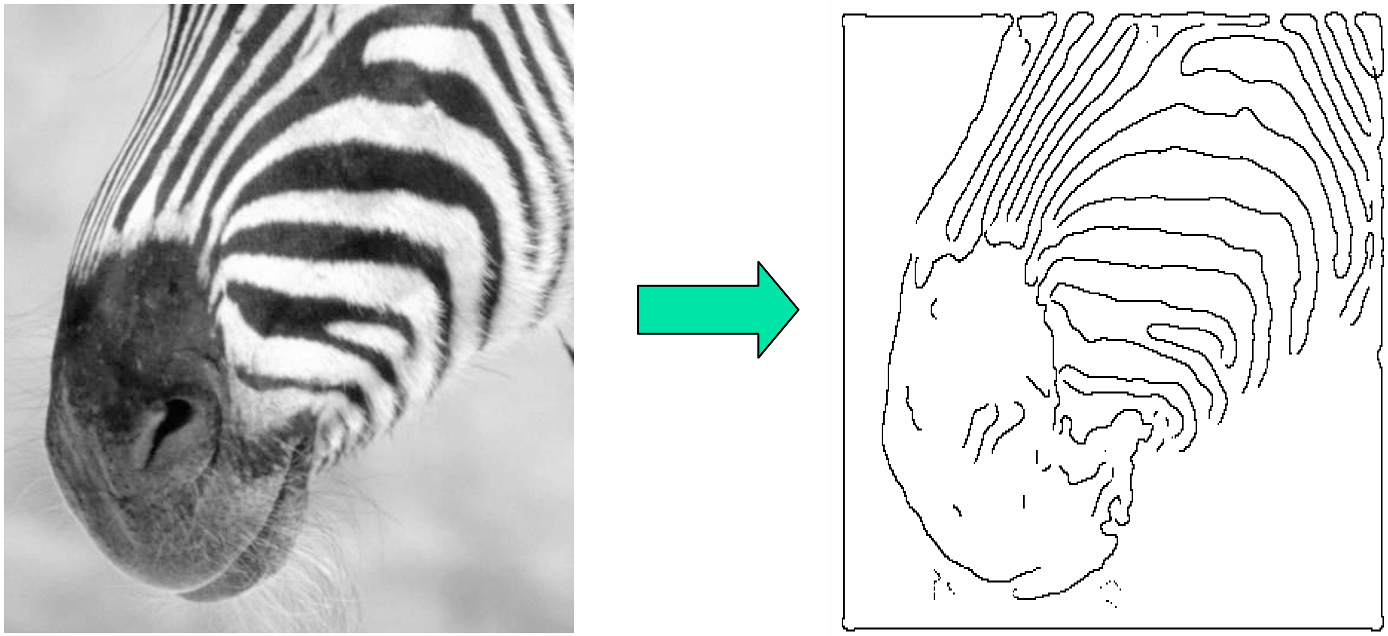


What Causes an Edge?

- Reflectance discontinuity (i.e., change in surface material properties)
- Depth discontinuity
- Surface orientation discontinuity
- Illumination discontinuity (e.g., shadow)



Quiz: How Can We Find Edges?



Detecting Discontinuities

- Image derivatives

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0} \left(\frac{f(x+\varepsilon) - f(x)}{\varepsilon} \right) \Rightarrow \frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}) - f(x)}{\Delta x}$$

- Convolve image with derivative filters

Backward difference	[-1 1]
Forward difference	[1 -1]
Central difference	[-1 0 1]

Derivative in Two-Dimensions

- Definition

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}\right) \qquad \frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0}\left(\frac{f(x, y+\varepsilon) - f(x,y)}{\varepsilon}\right)$$

- Approximation

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x_{n+1}, y_m) - f(x_n, y_m)}{\Delta x} \qquad \frac{\partial f(x,y)}{\partial y} \approx \frac{f(x_n, y_{m+1}) - f(x_n, y_m)}{\Delta x}$$

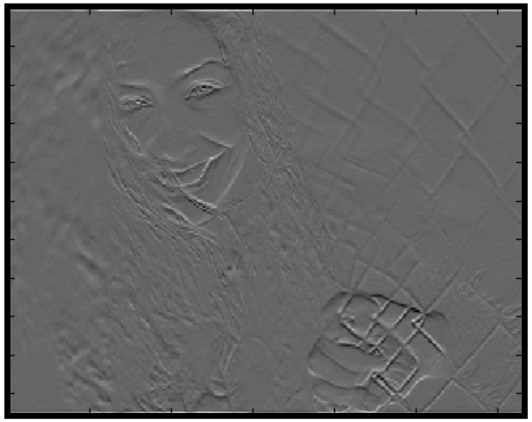
- Convolution kernels

$$f_x = \begin{bmatrix} 1 & -1 \end{bmatrix} \qquad f_y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Image Derivatives(导数)

Image I

$I_x = I * \begin{bmatrix} 1 & -1 \end{bmatrix}$



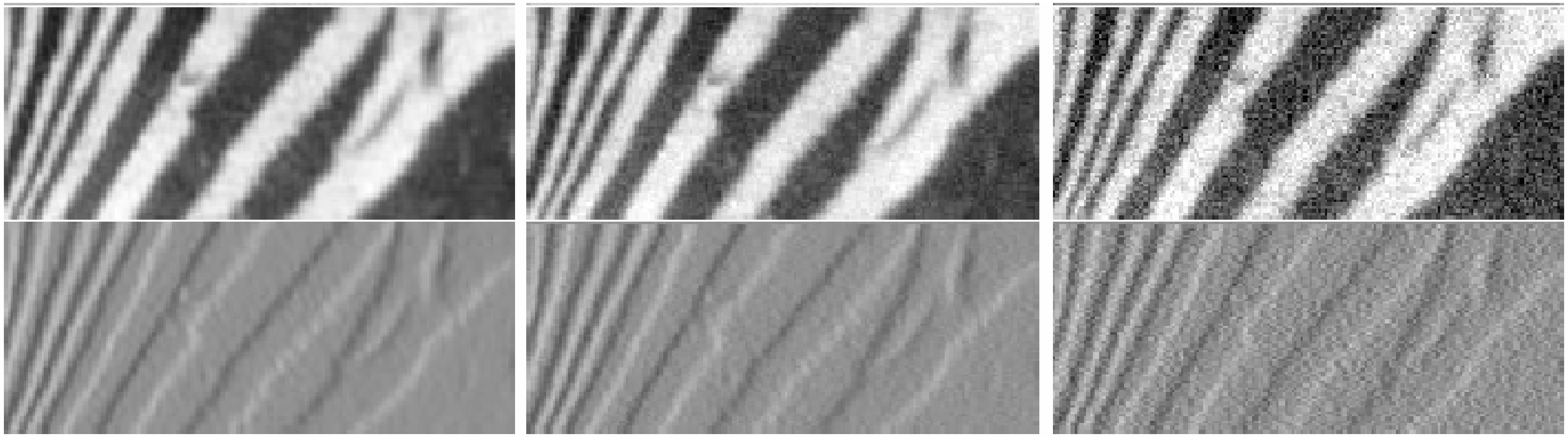
$I_y = I * \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

Derivatives and Noise

- Strongly affected by noise
  - obvious reason: image noise results in pixels that look very different from their neighbors
- The larger the noise is the stronger the response

- What is to be done?
  - Neighboring pixels look alike
  - Pixel along an edge look alike
  - Image smoothing should help
    - Force pixels different to their neighbors (possibly noise) to look like neighbors

Derivatives and Noise



Increasing noise →

Zero mean additive gaussian noise

Image Smoothing(图像平滑)

- Expect pixels to “**be like**” their neighbors
  - Relatively few reflectance changes
- Generally expect noise to be independent from pixel to pixel
  - Smoothing suppresses noise

Gaussian Smoothing(高斯平滑)

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

- Scale of Gaussian σ
  - As σ increases, more pixels are involved in average
  - As σ increases, image is more blurred
  - As σ increases, noise is more effectively suppressed

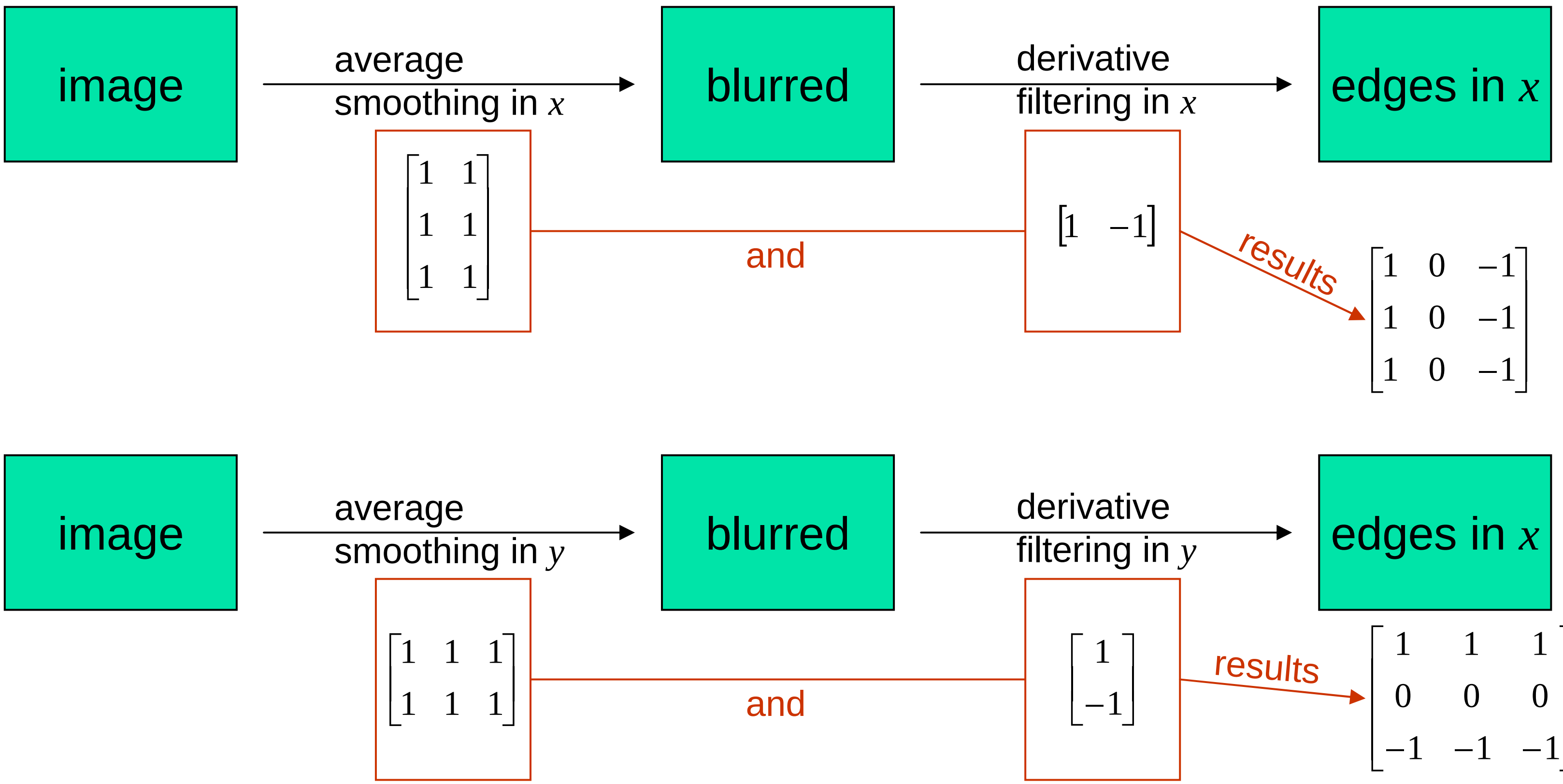
Edge Detectors(边缘检测子)

- Gradient operators(梯度算子)
  - Prewit
  - Sobel
- Laplacian of Gaussian (Marr-Hildreth)
- Gradient of Gaussian (Canny)
- Facet Model Based Edge Detector (Haralick)

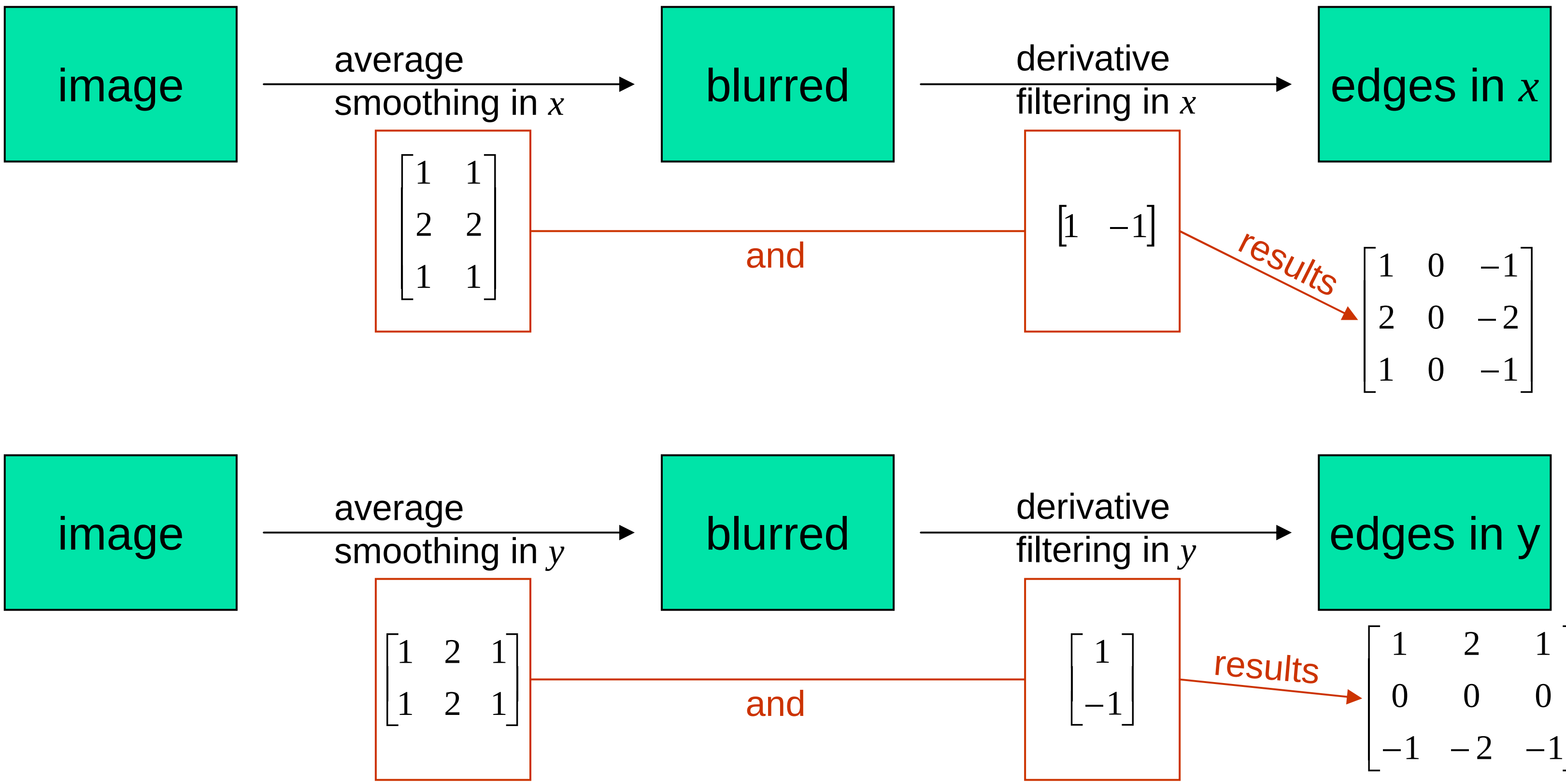
Prewitt and Sobel Edge Detector

- Compute derivatives
  - In x and y directions
- Find gradient magnitude
- Threshold gradient magnitude

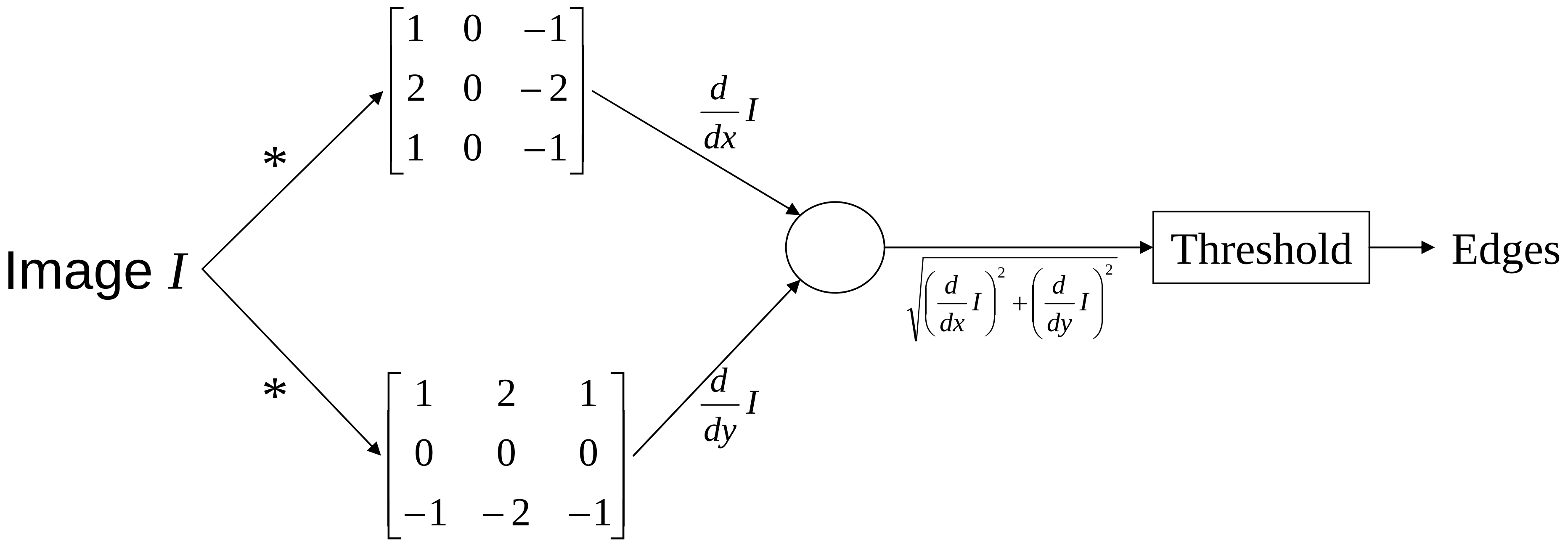
Prewitt Edge Detector



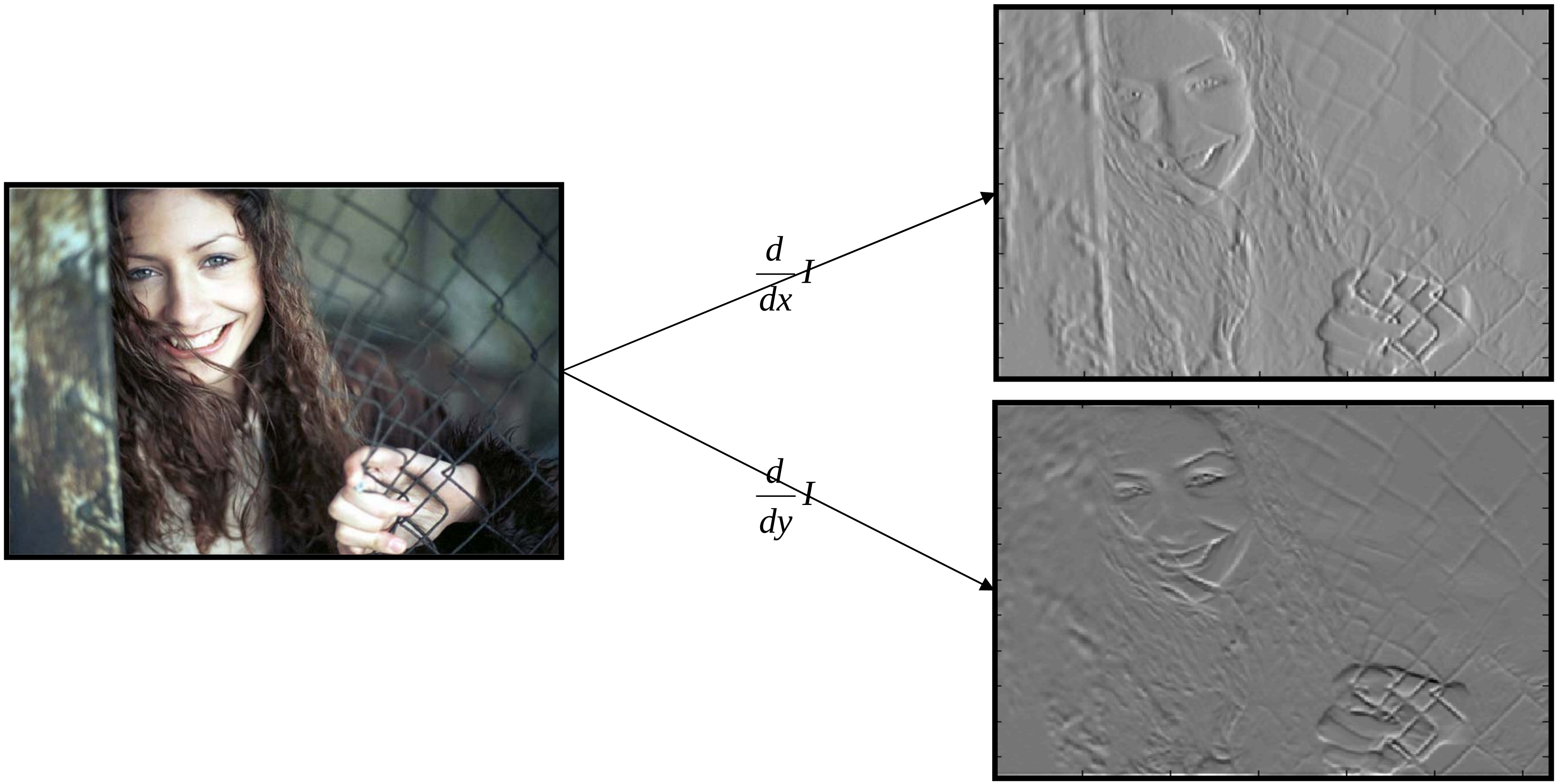
Sobel Edge Detector



Sobel Edge Detector



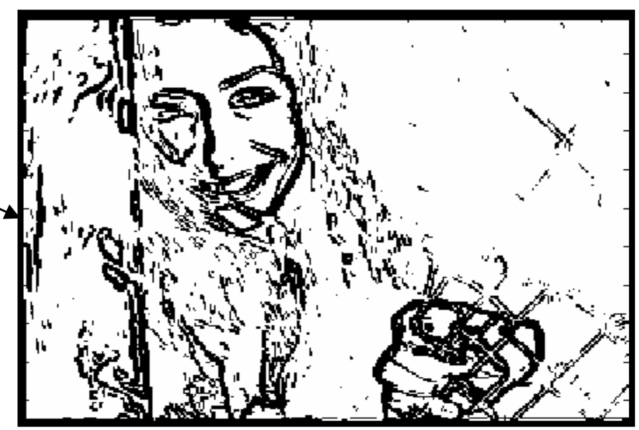
Sobel Edge Detector



Sobel Edge Detector

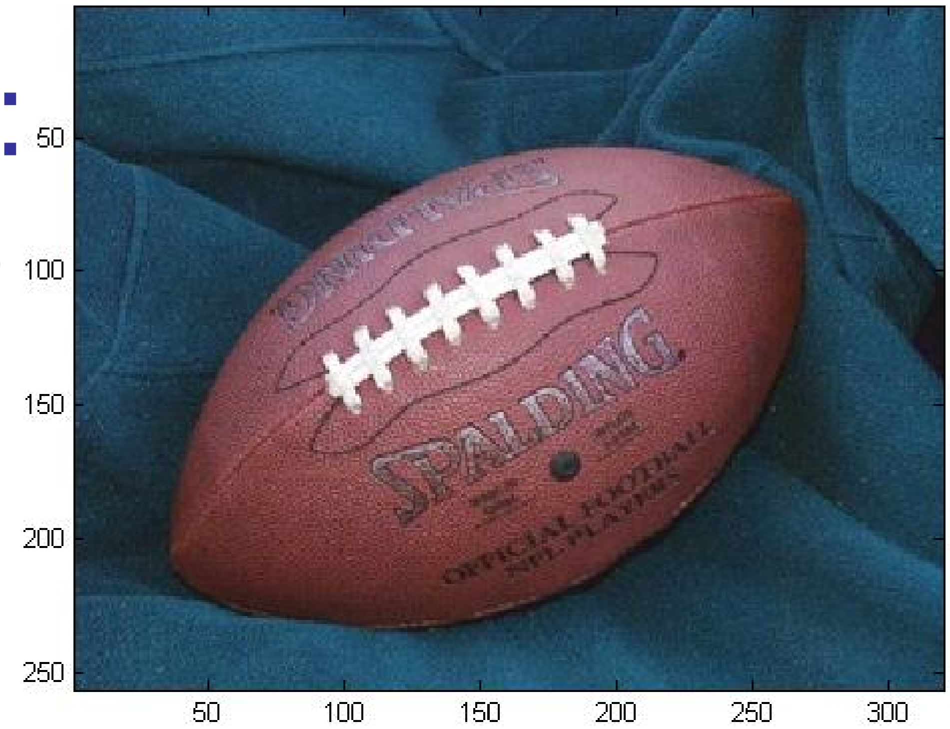
$$\Delta = \sqrt{\left(\frac{d}{dx}I\right)^2 + \left(\frac{d}{dy}I\right)^2}$$

$$\Delta \geq Threshold = 100$$



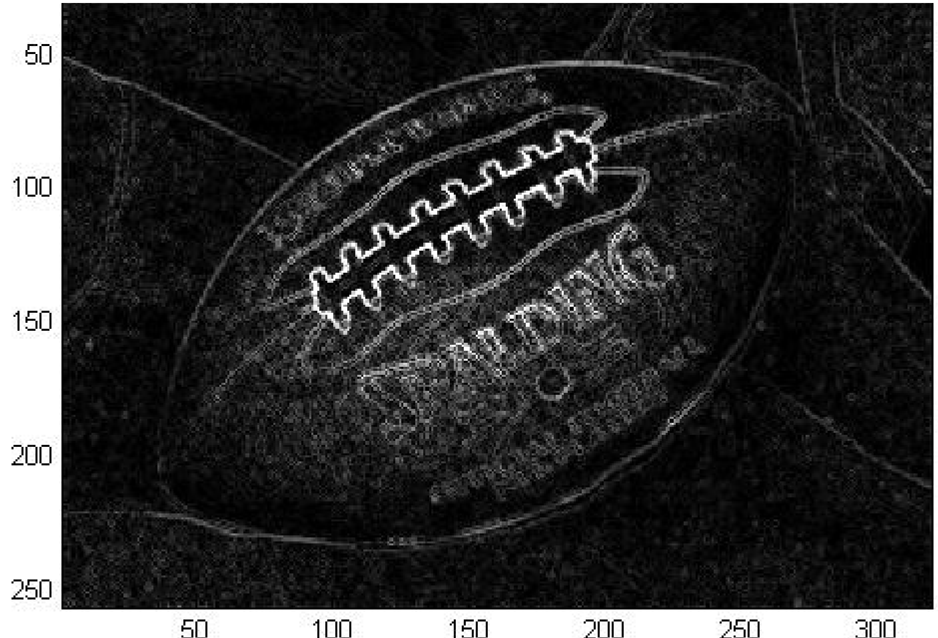
Matlab Demos of Edge detection

Edge Finding: Matlab Demo




```
im = imread('football.jpg');
image(im);
figure(2);
bw = double(rgb2gray(im));

[dx,dy] = gradient(bw);
gradmag = sqrt(dx.^2 + dy.^2);
image(gradmag);
```



What is Image Filtering?

Modify the pixels in an image based on some function of a local neighborhood of the pixels

10	5	3
4	5	1
1	1	7

Some function →

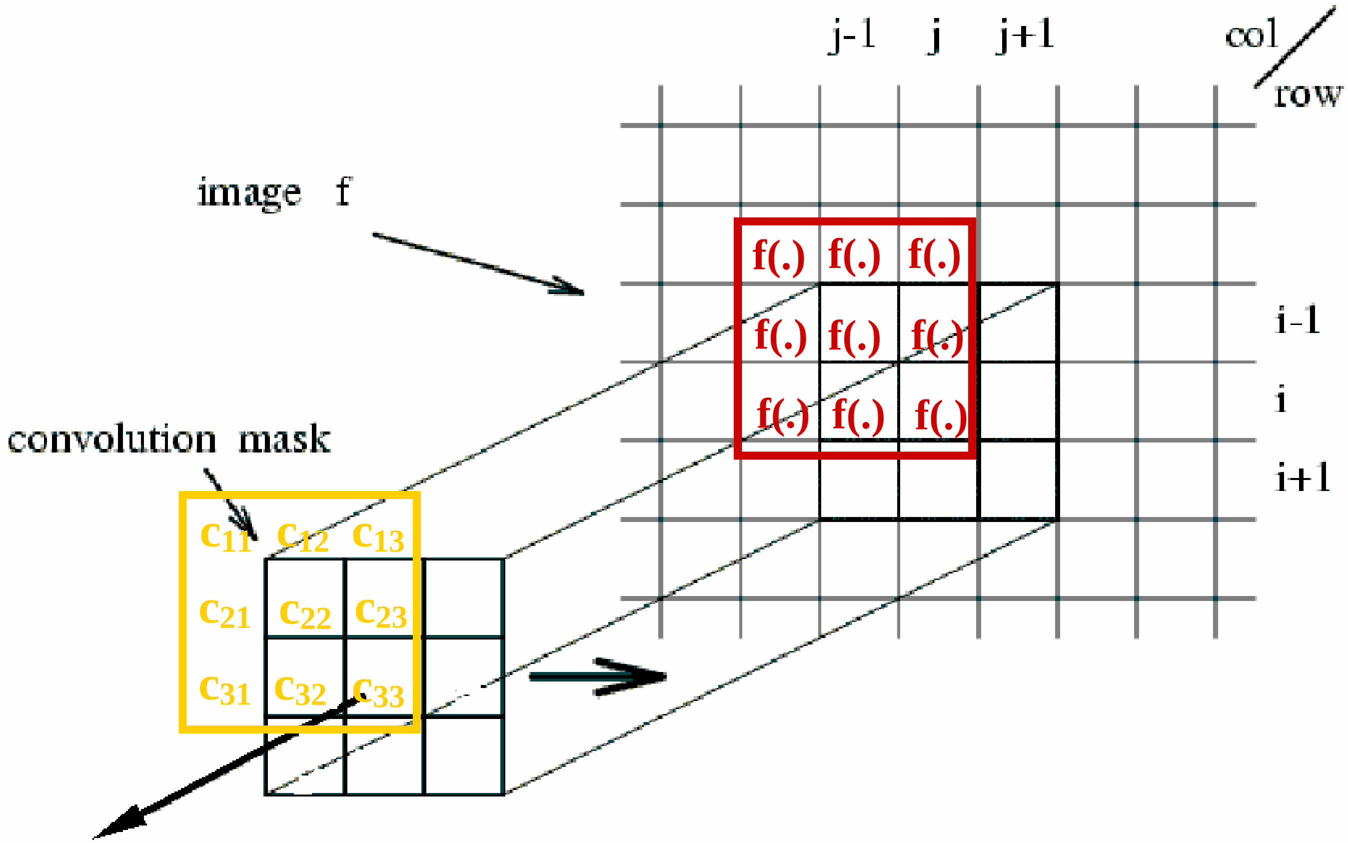
	7	

Linear Filtering

- Linear case is simplest and most useful
  - Replace each pixel with a linear combination of its neighbors.
- The prescription for the linear combination is called the convolution kernel.

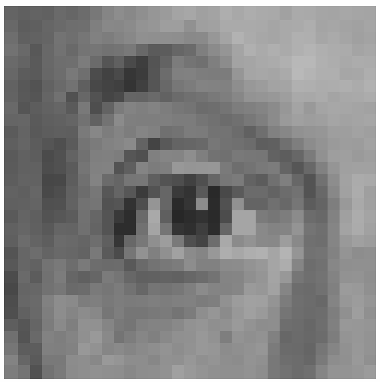
$$
\begin{array}{|c|c|c|}\hline 10 & 5 & 3 \\ \hline 4 & 5 & 1 \\ \hline 1 & 1 & 7 \\ \hline \end{array}
\otimes
\underbrace{\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0.5 & 0 \\ \hline 0 & 1.0 & 0.5 \\ \hline \end{array}}_{\text{kernel}}
=
\begin{array}{|c|c|c|}\hline & & \\ \hline & 7 & \\ \hline & & \\ \hline \end{array}
$$

Linear Filter = Convolution

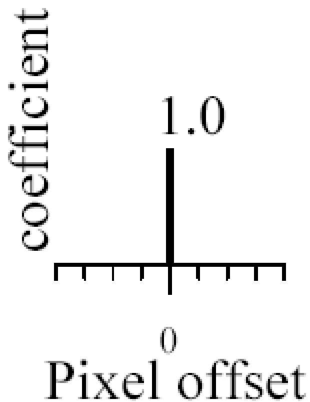


$$
\begin{aligned}
o(i,j) = \; & c_{11}\, f(i-1,j-1) + c_{12}\, f(i-1,j) + c_{13}\, f(i-1,j+1) + \\
& c_{21}\, f(i,j-1) + c_{22}\, f(i,j) + c_{23}\, f(i,j+1) + \\
& c_{31}\, f(i+1,j-1) + c_{32}\, f(i+1,j) + c_{33}\, f(i+1,j+1)
\end{aligned}
$$

Filtering Examples

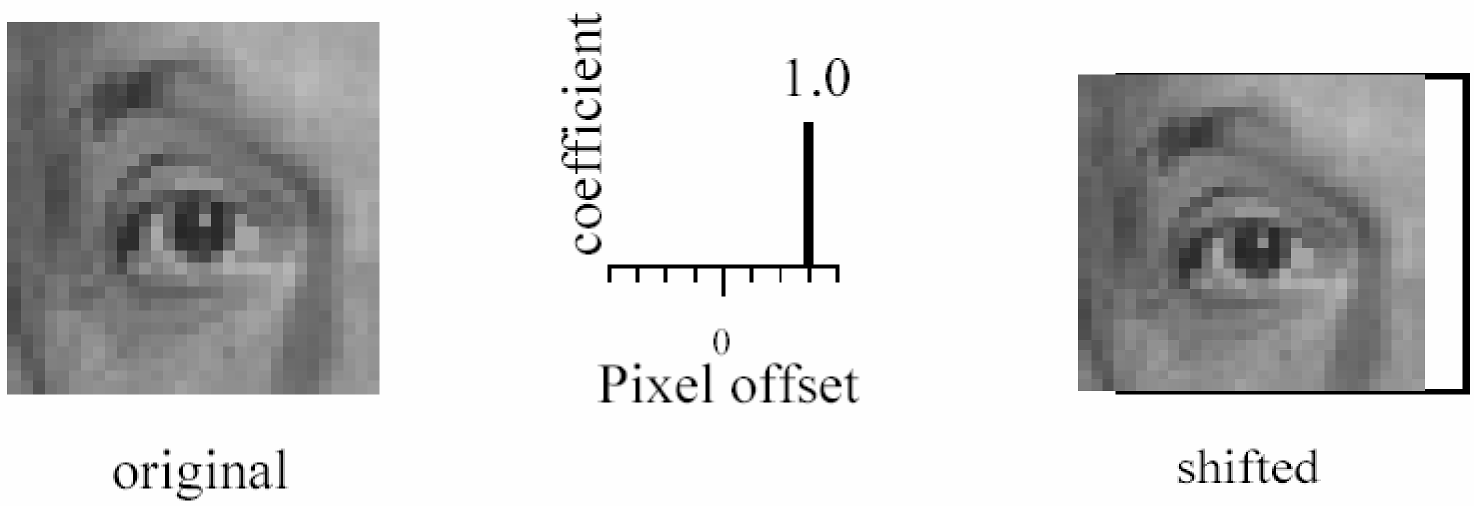


original

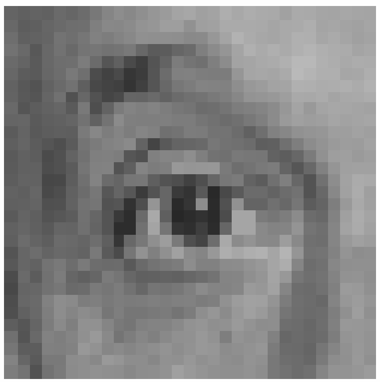


Filtered
(no change)

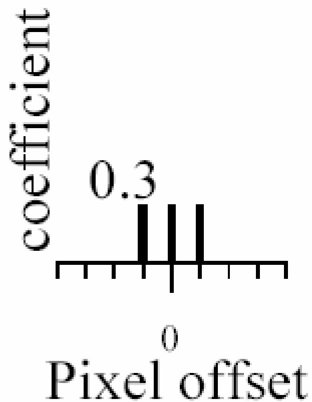
Filtering Examples

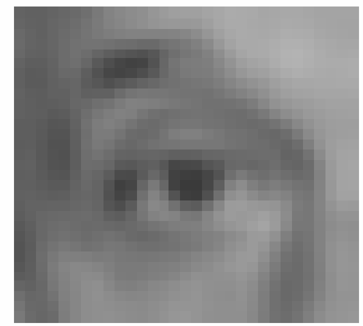


Filtering Examples



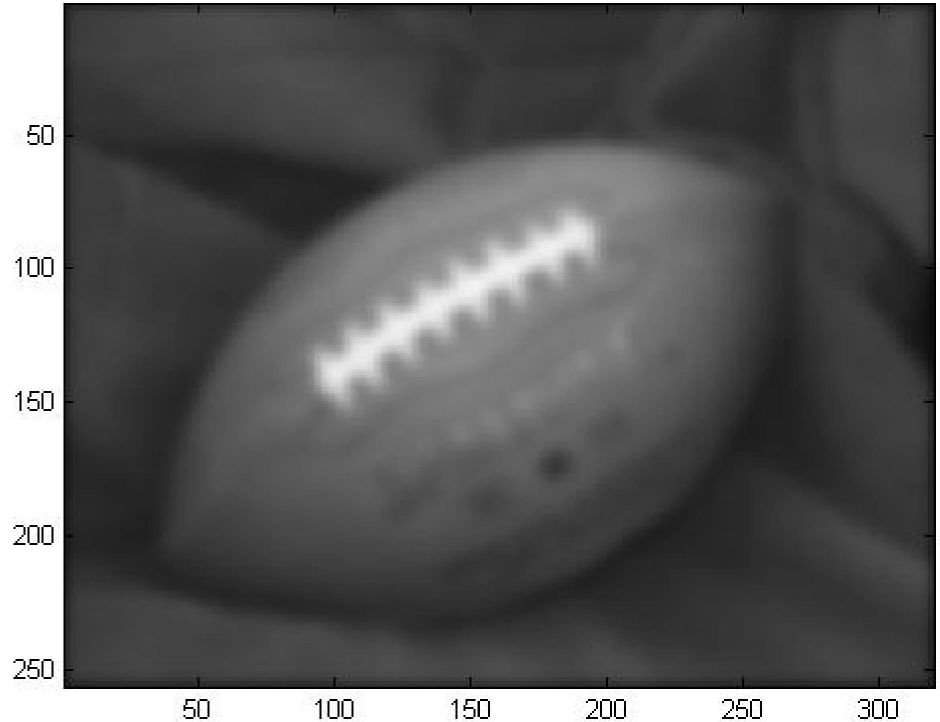
original





Blurred (filter applied in both dimensions).

Image Smoothing With Gaussian

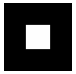


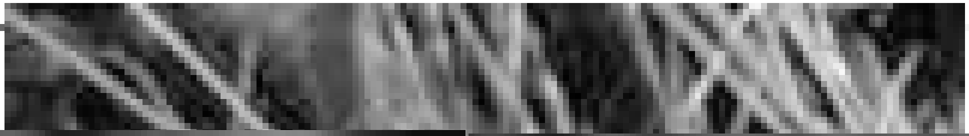

```
figure(3);
sigma = 3;
width = 3 * sigma;
support = -width : width;
gauss2D = exp( - (support / sigma).^2 / 2);
gauss2D = gauss2D / sum(gauss2D);
smooth = conv2(conv2(bw, gauss2D, 'same'), gauss2D', 'same');
image(smooth);
colormap(gray(255));
gauss3D = gauss2D' * gauss2D;
tic ; smooth = conv2(bw,gauss3D, 'same'); toc
```

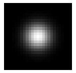
Slide credit: Marc Pollefeys

Smoothing With Gaussian

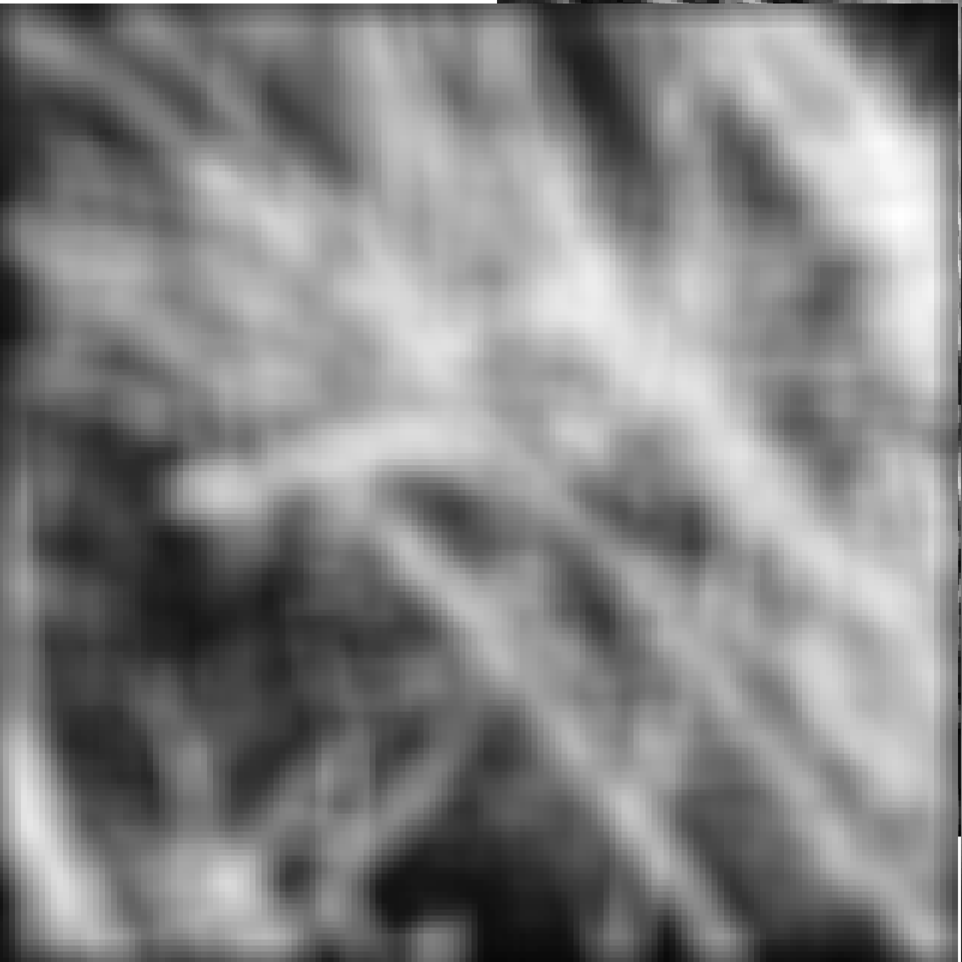
Averaging

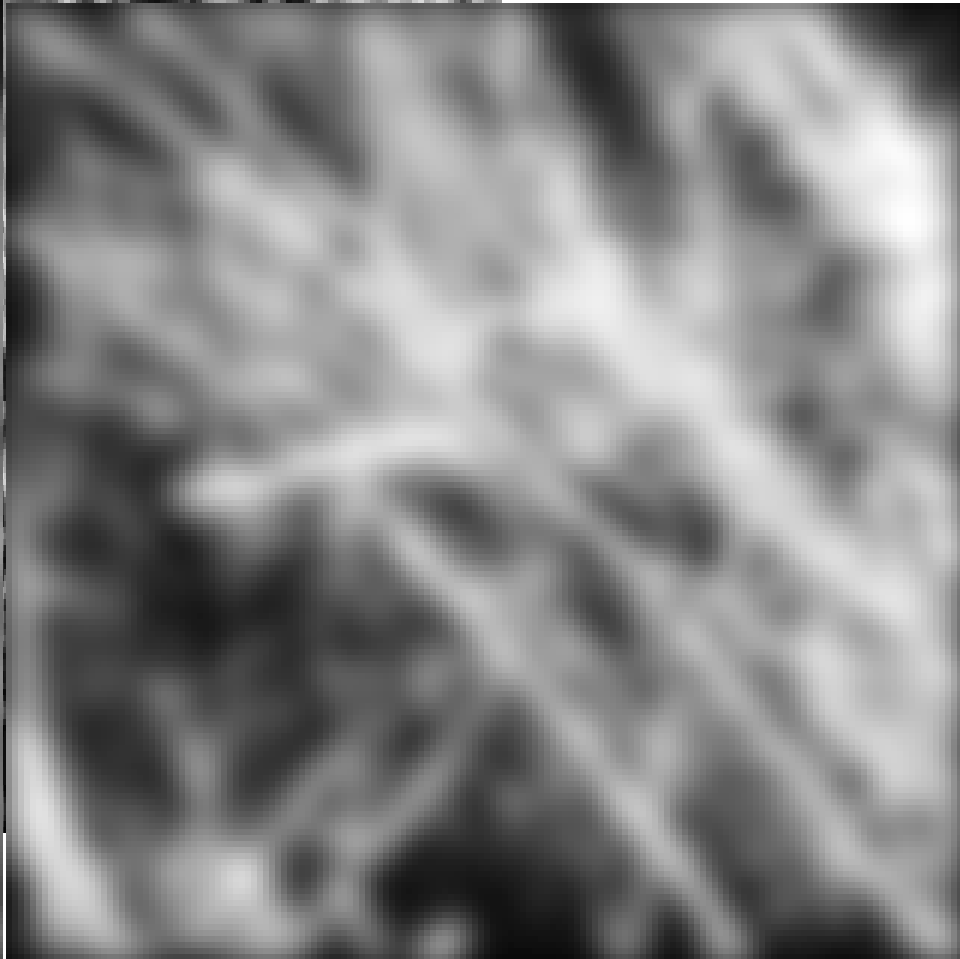






Gaussian



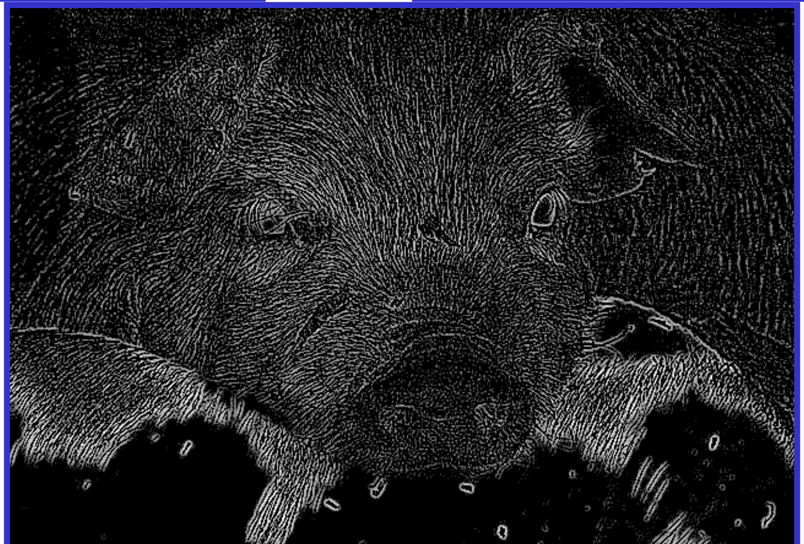


Example of Blurring

-

=

Image

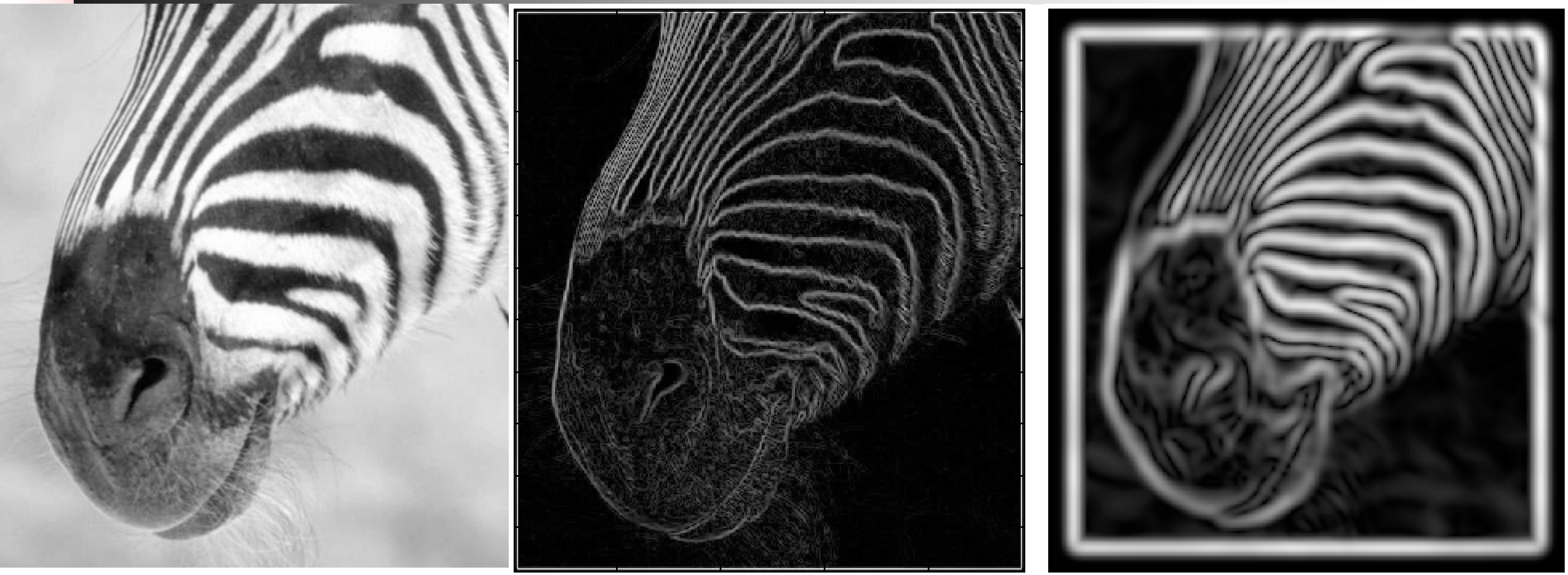


Blurred Image

Edge Detection With Smoothed Images

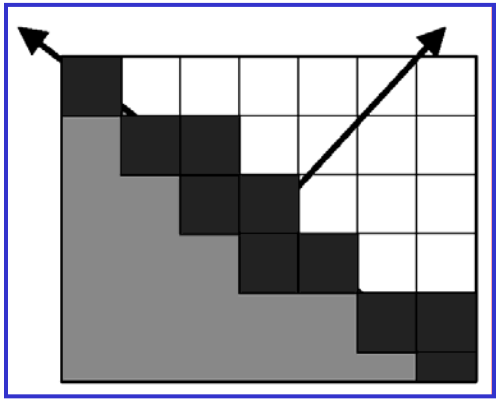
```
figure(4);
[dx,dy] = gradient(smooth);
gradmag = sqrt(dx.^2 + dy.^
gmax = max(max(gradmag));
imshow(gradmag);
colormap(gray(gmax));
```

Increased smoothing



- Eliminates noise edges.
- Makes edges smoother and thicker.
- Removes fine detail.

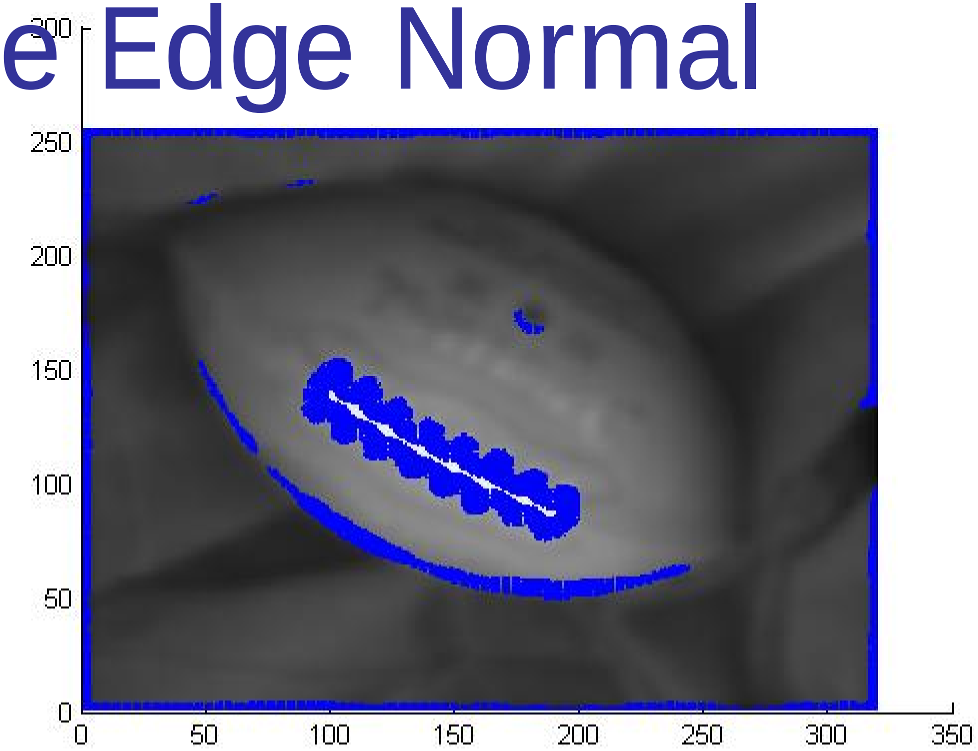
The Edge Normal



$$S = \sqrt{dx^2 + dy^2}$$

$$\alpha = \arctan \frac{dy}{dx}$$

Displaying the Edge Normal

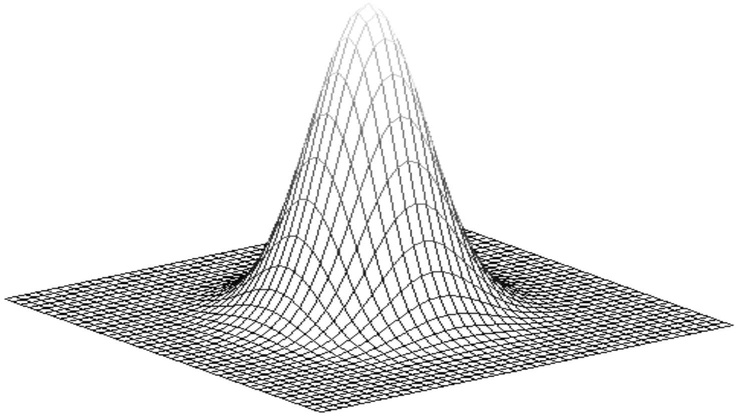


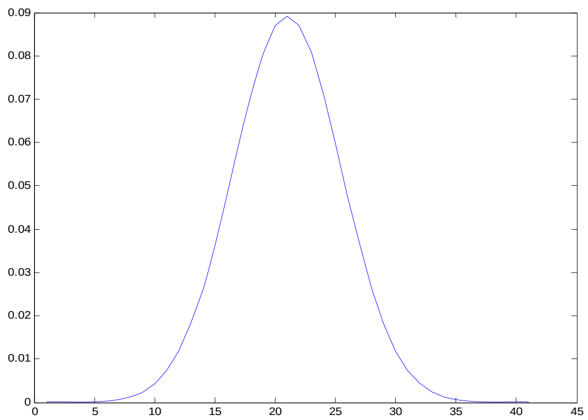

```
figure(5);
hold on;
image(smooth);
colormap(gray(255));
[m,n] = size(gradmag);

edges = (gradmag > 0.3 * gmax);
inds = find(edges);
[posx,posy] = meshgrid(1:n,1:m); posx2=posx(inds); posy2=posy(inds);
gm2= gradmag(inds);
sintheta = dx(inds) ./ gm2;
costheta = - dy(inds) ./ gm2;
quiver(posx2,posy2, gm2 .* sintheta / 10, -gm2 .* costheta / 10,0);
hold off;
```

Separable Kernels

$$f[m,n] = I \otimes g = I \otimes g_X \otimes g_Y$$

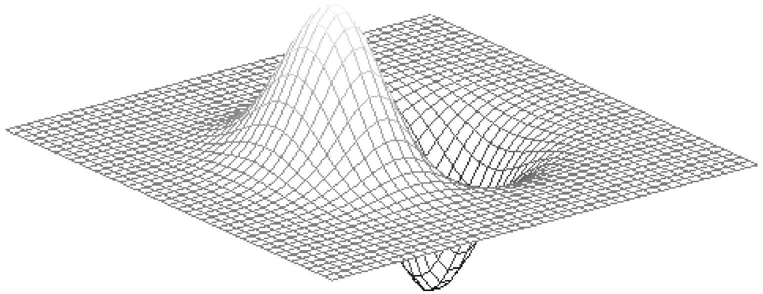




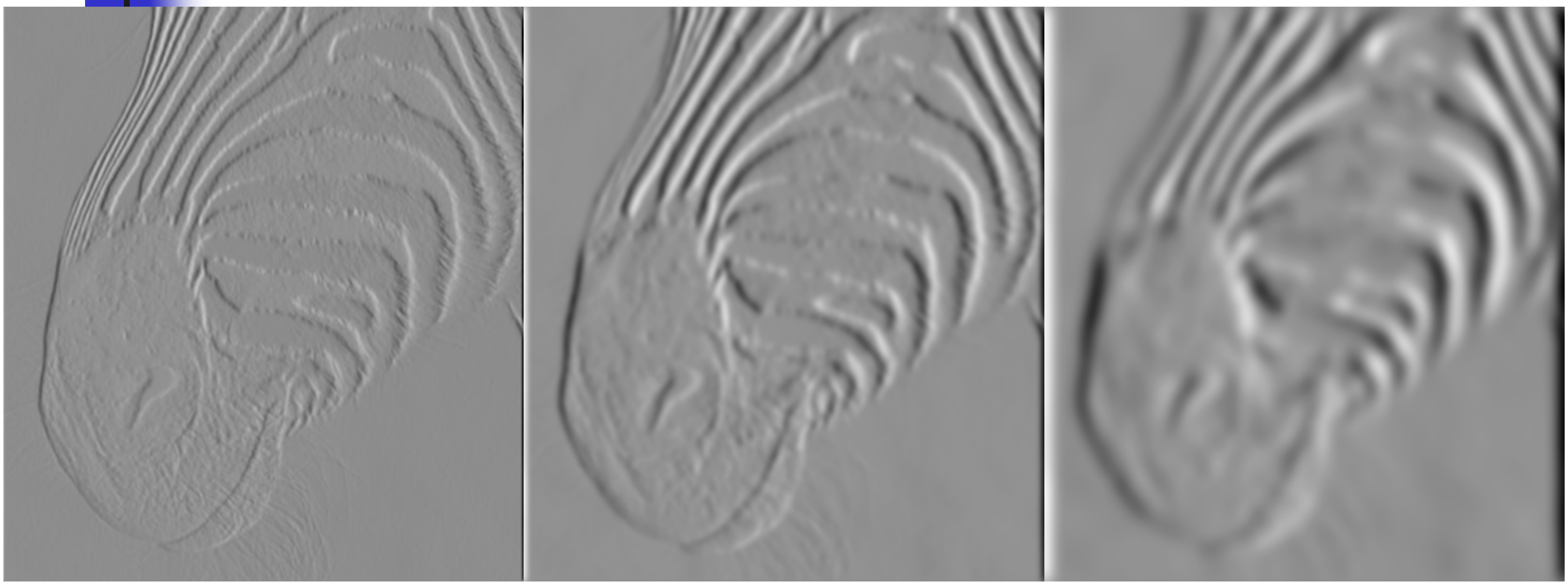
Combining Kernels / Convolutions

$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$



Effect of Smoothing Radius



1 pixel 3 pixels 7 pixels

Robert's Cross Operator

$$\left| \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array} \right| + \left| \begin{array}{|c|c|} \hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array} \right|$$

$$S = \sqrt{[\, I(x, y) - I(x+1, y+1)\,]^2 + [\, I(x, y+1) - I(x+1, y)\,]^2}$$

or

$$S = |\, I(x, y) - I(x+1, y+1)\, | + |\, I(x, y+1) - I(x+1, y)\, |$$

Sobel Operator

$$S_1 = \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \qquad S_2 = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$\text{Edge Magnitude} = \sqrt{S_1^2 + S_1^2}$$

$$\text{Edge Direction} = \tan^{-1}\left(\frac{S_1}{S_2}\right)$$

The Sobel Kernel, Explained

-1	0	1
-2	0	2
-1	0	1

$$= 1/4 * [\ 1\ \ 0\ \ -1] \circledast \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

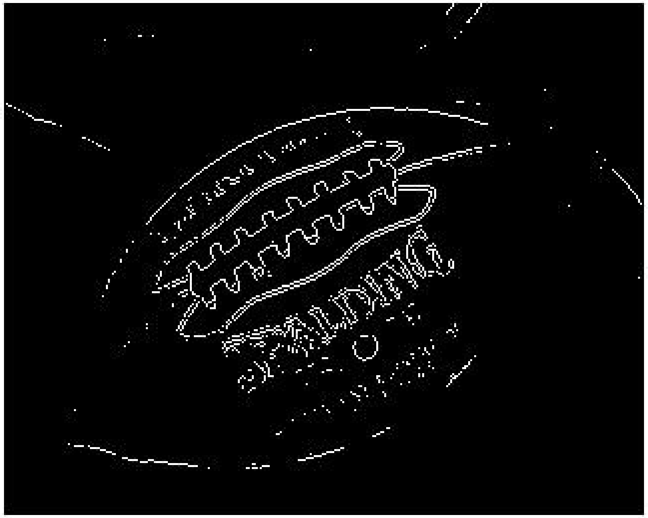
Sobel kernel is separable!

1	2	1
0	0	0
-1	-2	-1

$$= 1/4 * [\ 1\ \ 2\ \ 1] \circledast \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Sobel Edge Detector

```
figure(6)
edge(bw, 'sobel')
```



Robinson Compass Masks

-1	0	1
-2	0	2
-1	0	1

→

0	1	2
-1	0	1
-2	-1	0

↗

1	2	1
0	0	0
-1	-2	-1

↑

2	1	0
1	0	-1
0	-1	-2

↖

1	0	-1
2	0	-2
1	0	-1

←

0	-1	-2
1	0	-1
2	1	0

↙

-1	-2	-1
0	0	0
1	2	1

↓

-2	-1	0
-1	0	1
0	1	2

↘

Claim Your Own Kernel!

1	1	1
1	-2	1
-1	-1	-1

Prewitt 1

5	5	5
-3	0	-3
-3	-3	-3

Kirsch

-1	$-\sqrt{2}$	-1
0	0	0
1	$\sqrt{2}$	1

Frei & Chen

1	1	1
0	0	0
-1	-1	-1

Prewitt 2

1	2	1
0	0	0
-1	-2	-1

Sobel

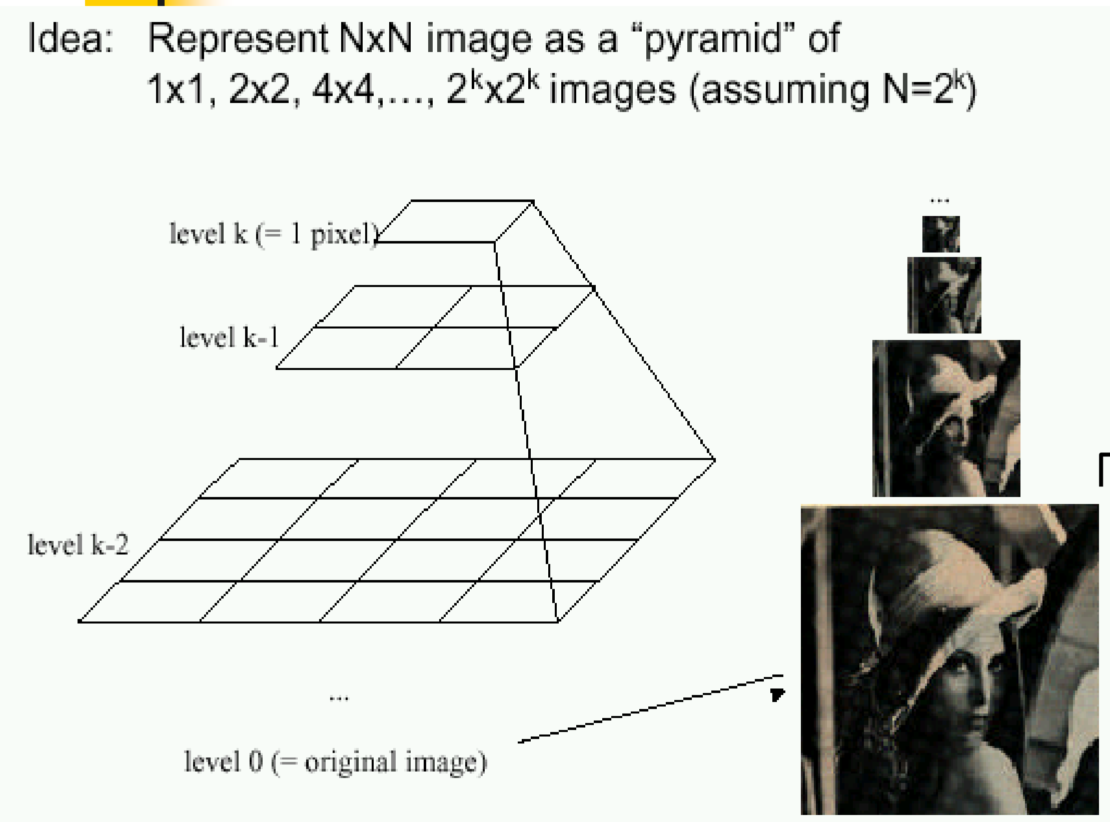
Yours?

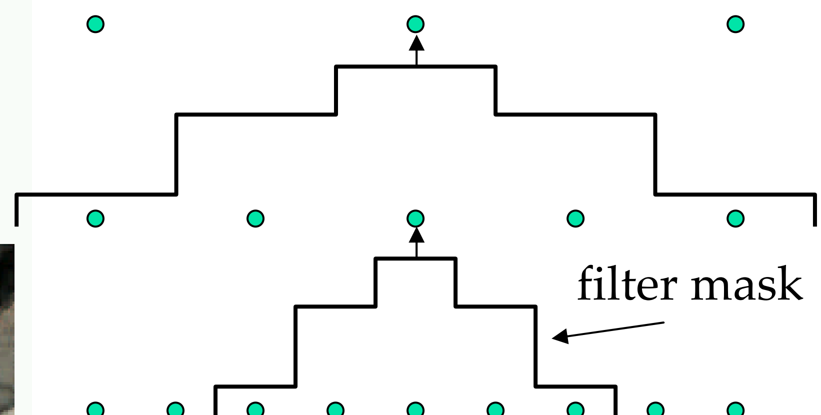
Comparison (by Allan Hanson)

- Analysis based on a step edge inclined at an angle q (relative to y-axis) through center of window.
- Robinson/Sobel: true edge contrast less than 1.6% different from that computed by the operator.
- Error in edge direction
  - Robinson/Sobel: less than 1.5 degrees error
  - Prewitt: less than 7.5 degrees error
- Summary
  - Typically, 3 x 3 gradient operators perform better than 2 x 2.
  - Prewitt2 and Sobel perform better than any of the other 3x3 gradient estimation operators.
  - In low signal to noise ratio situations, gradient estimation operators of size larger than 3 x 3 have improved performance.
  - In large masks, weighting by distance from the central pixel is beneficial.

Image Pyramids(图像金字塔)

Sometimes We want Many Resolutions

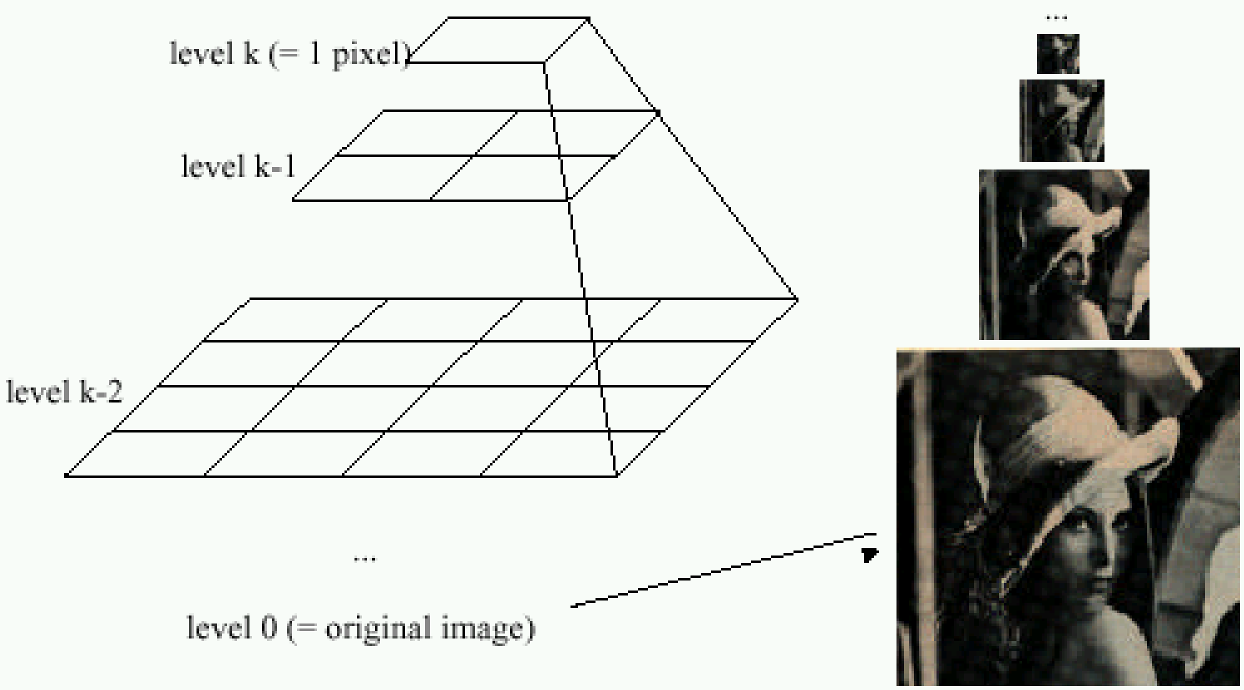




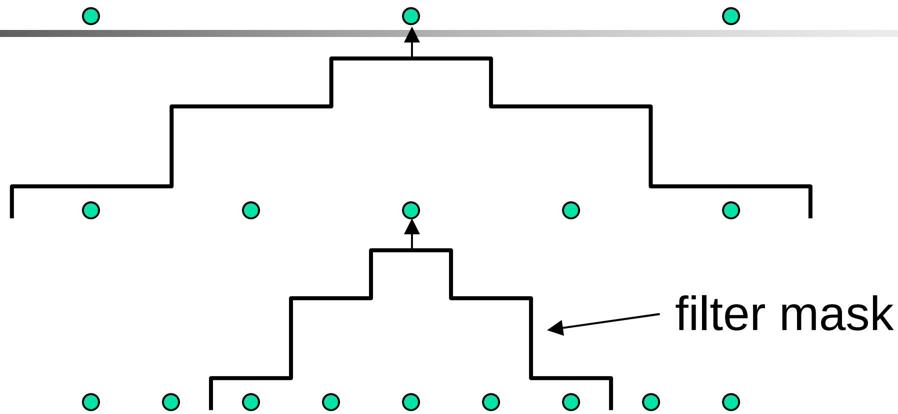
- Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]
  - In computer graphics, a *mip map* [Williams, 1983]
  - A precursor to *wavelet transform*
- Gaussian Pyramids have all sorts of applications in computer vision
  - We'll talk about these later in the course

Image Pyramids

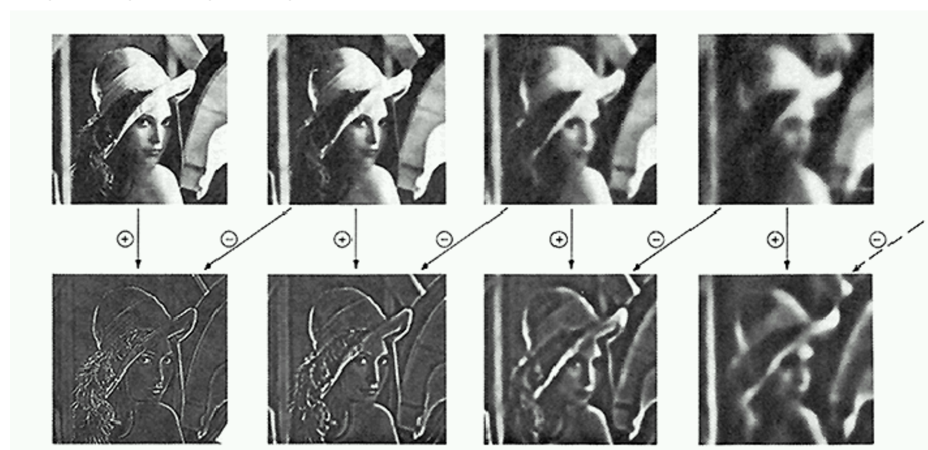
Idea: Represent NxN image as a "pyramid" of 1x1, 2x2, 4x4,..., 2^kx2^k images (assuming N=2^k)



Pyramid Creation



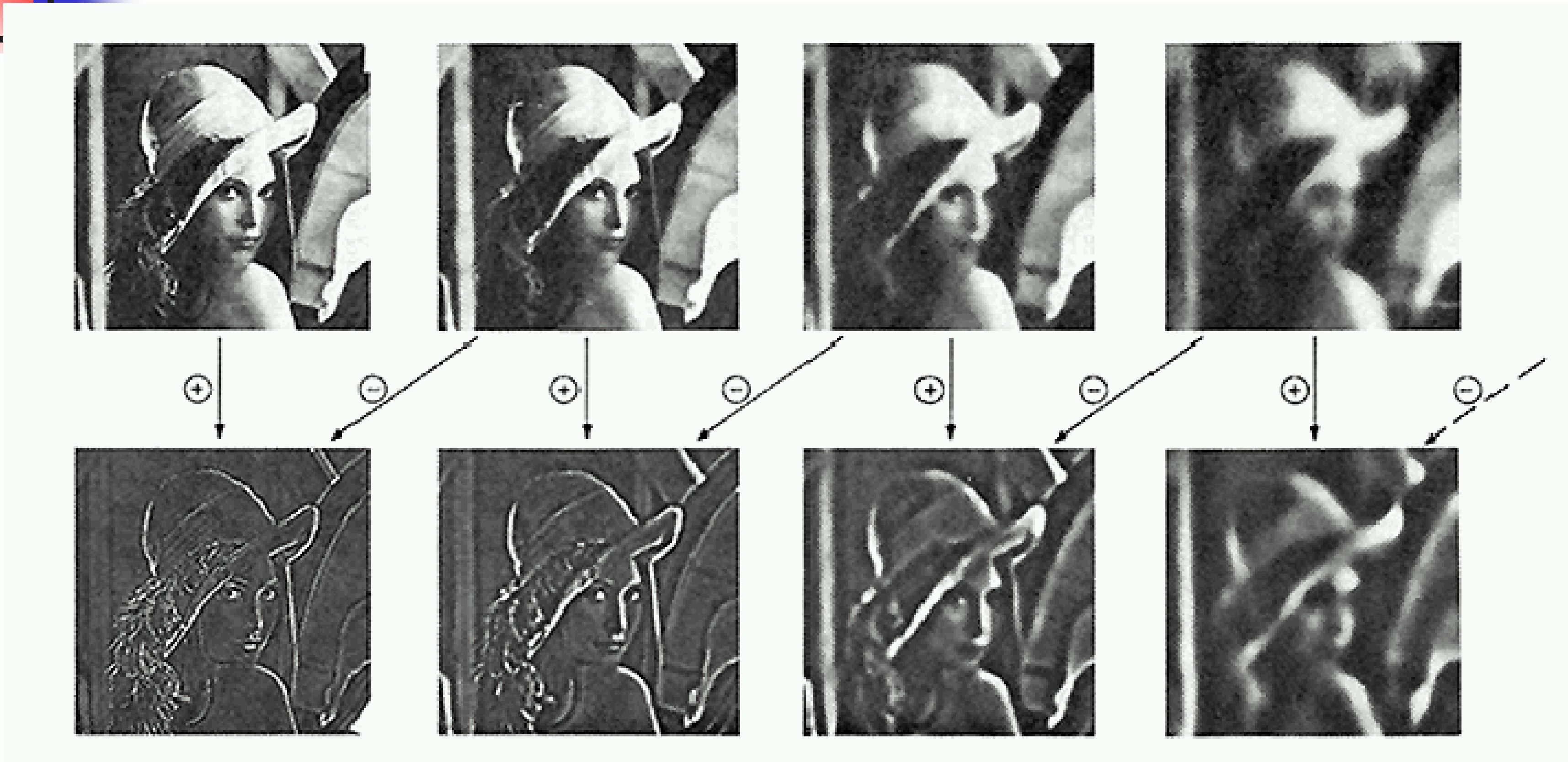
"Gaussian" Pyramid



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

Octaves in the Spatial Domain

Lowpass Images

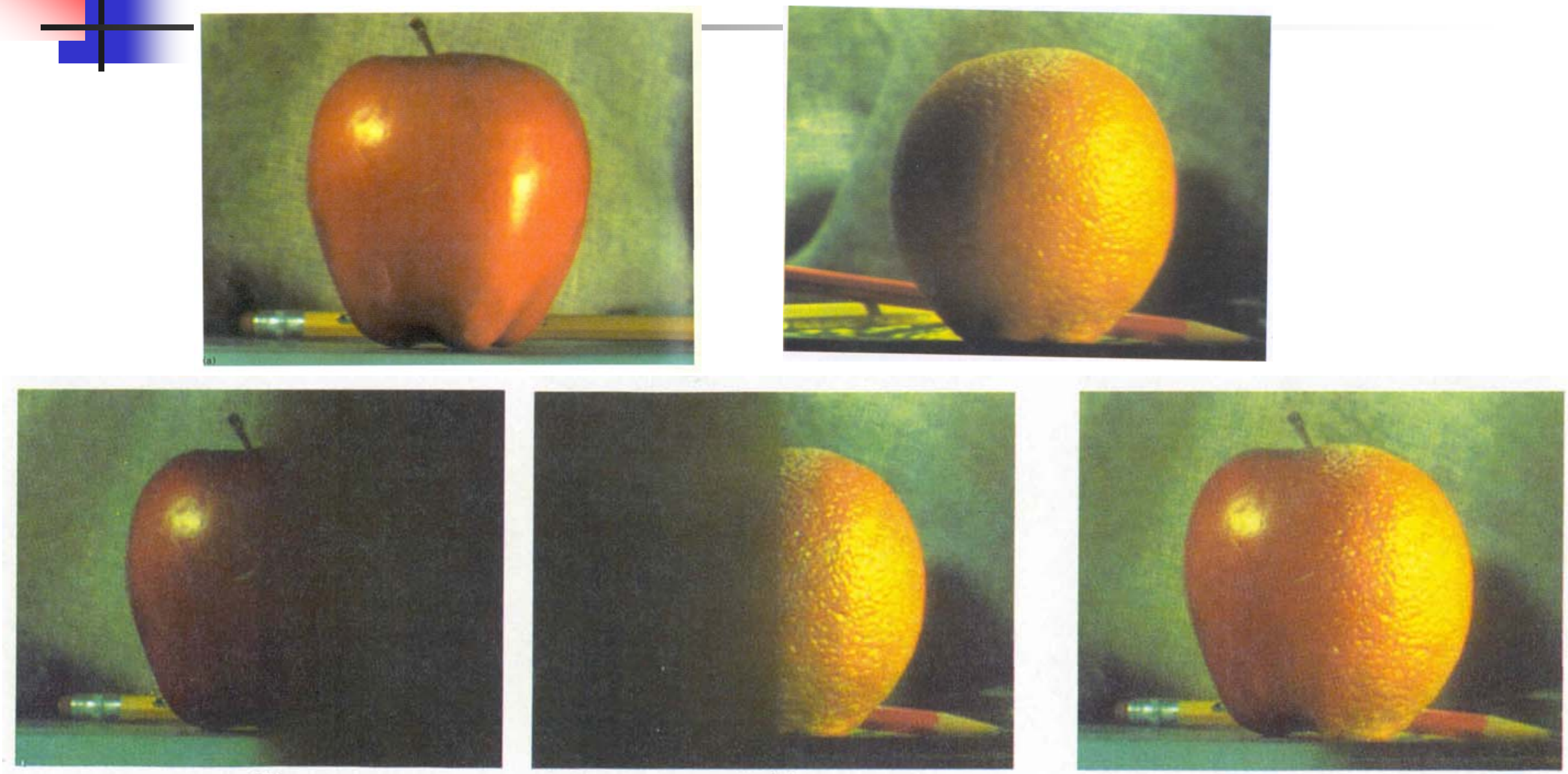


- Bandpass Images

Pyramids

- Many applications
  - small images faster to process
  - good for multiresolution processing
  - compression
  - progressive transmission
- Known as “MIP-maps” in graphics community
- Precursor to wavelets
  - Wavelets also have these advantages

Pyramid Blending



(d) (h) (l)

- Burt, P. J. and Adelson, E. H., A multiresolution spline with applications to image mosaics, ACM Transactions on Graphics, 42(4), October 1983, 217-236.

Questionnaire

- I want simpler and more intuitive explanations
- I want more technique details
- The current method is good to me
- Other comments on the lectures or the assignments?