Images and Filters

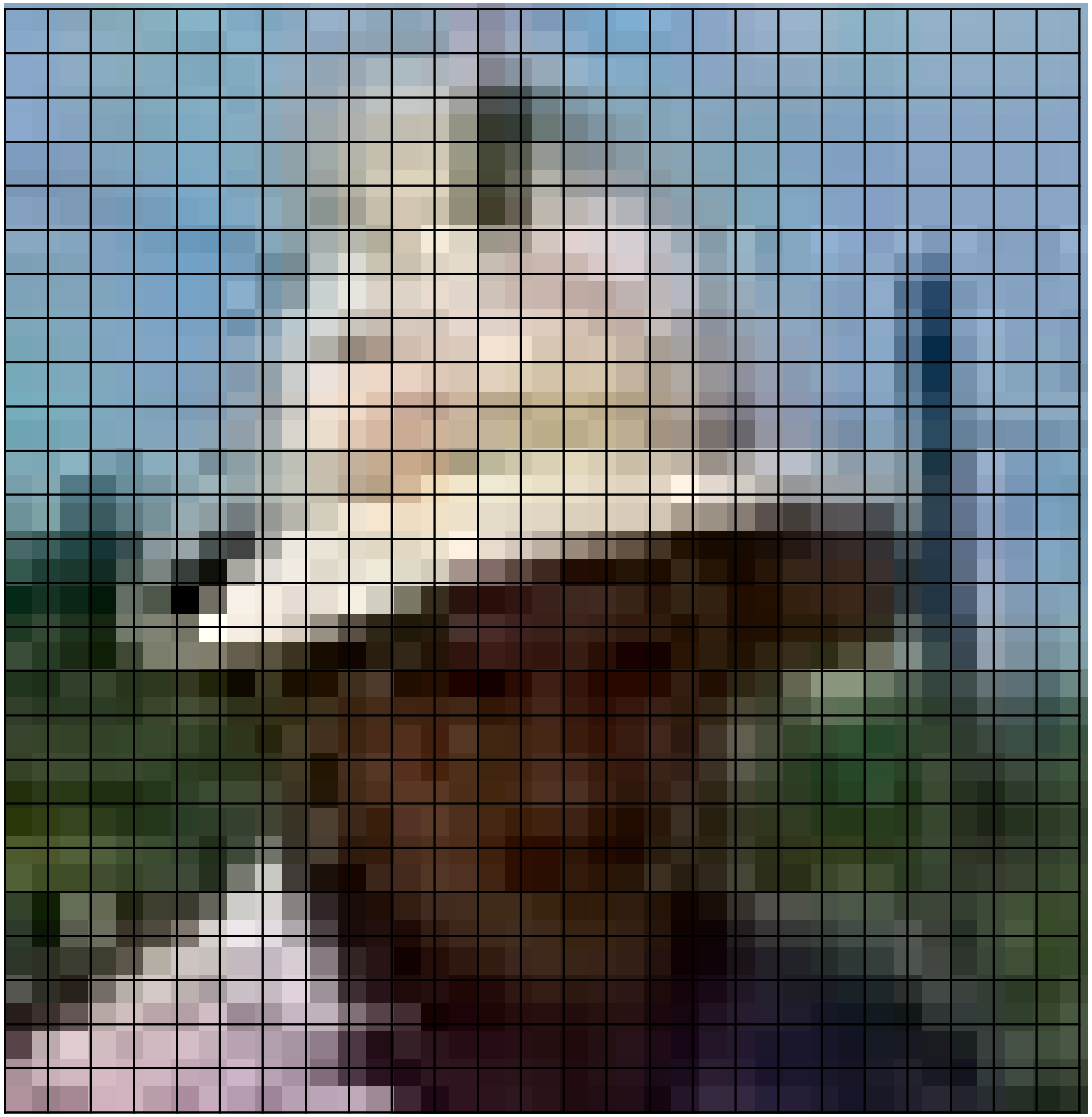
EE/CSE 576

Linda Shapiro

What is an image?

$$P = f(x, y)$$

$$f : R^2 \Rightarrow R$$



$$P = f(x,y)$$

$$f : R^2 \Rightarrow R$$

1. We **sample** the image to get a discrete set of pixels with **quantized** values.
2. For a gray tone image there is one **band** F(r,c), with values usually between 0 and 255.
3. For a color image there are 3 bands R(r,c), G(r,c), B(r,c)

Image Operations

(functions of functions)

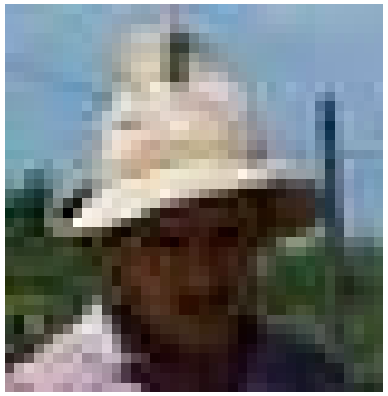
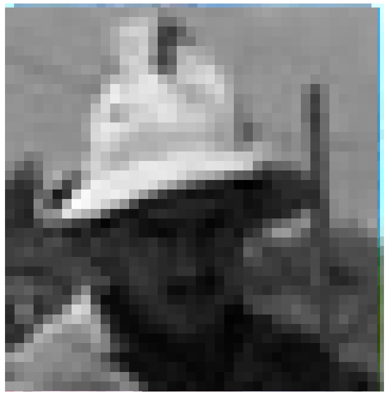
F() = 

Image Operations
(functions of functions)

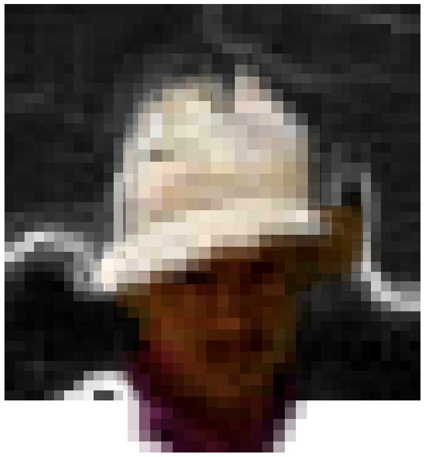
F( ) = 

Image Operations

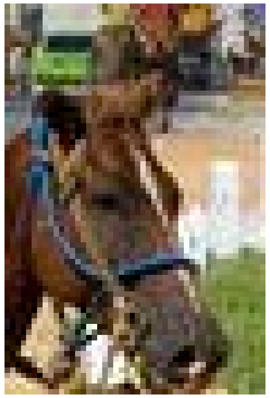
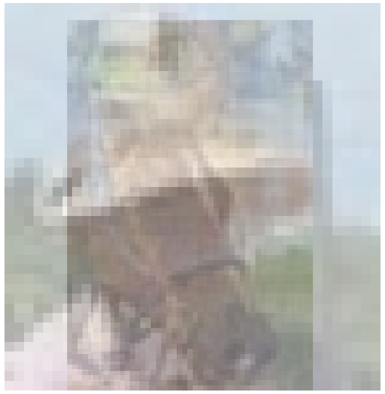
(functions of functions)

F( ) =

0.1
0
0.8
0.9
0.9
0.9
0.2
0.4
0.3
0.6
0
0
0.1
0.5
0.9
0.9
0.2
0.4
0.3
0.6
0
0
0.1
0.9
0.9
0.2
0.4
0.3
0.6
0
0
0.1
0.5

Image Operations

(functions of functions)

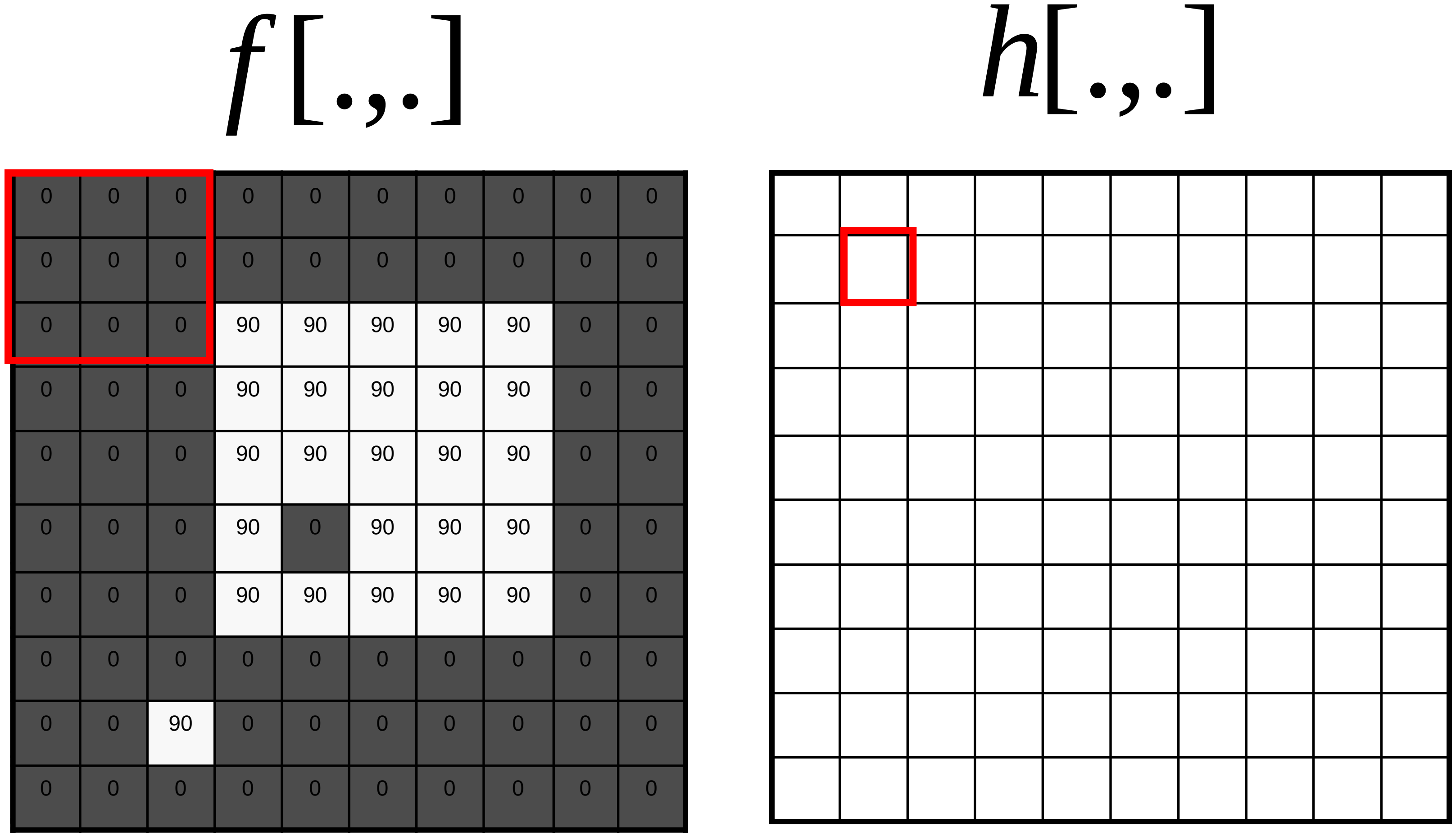
F(, ) = 

Local image functions

F( ) =

Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10							

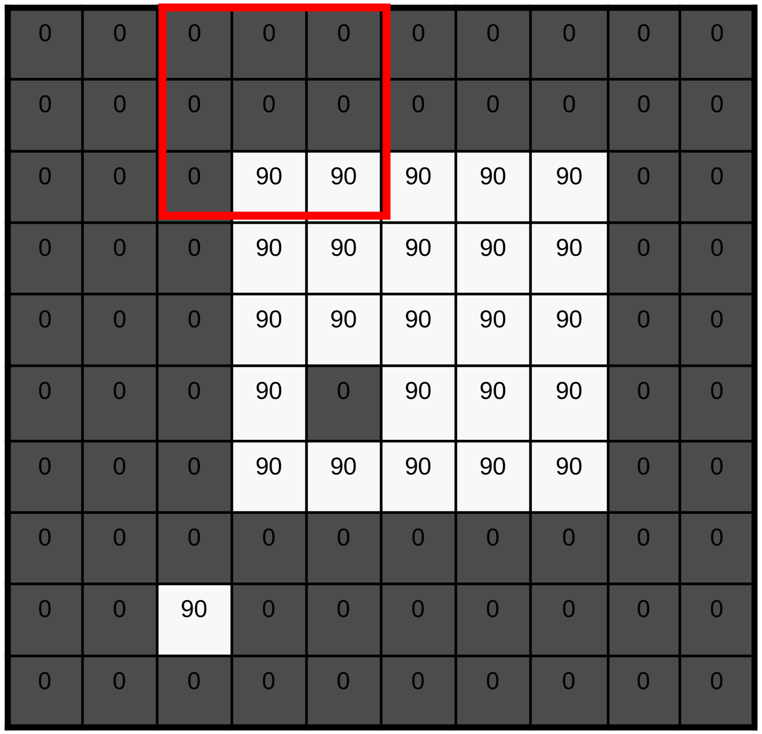
$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

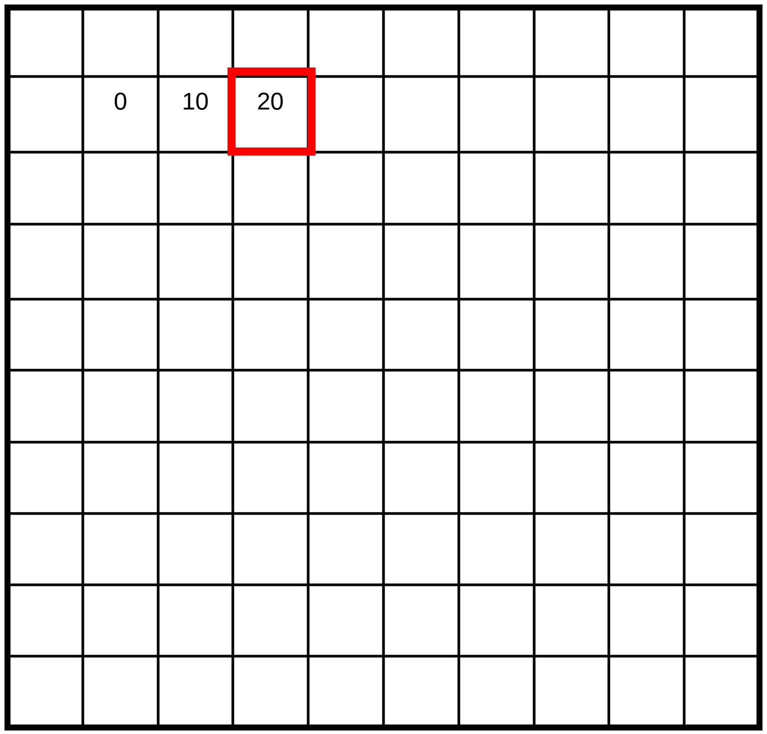
$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$f[.,.]$$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[.,.]$$



	0	10	20						

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$$f[.,.]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[.,.]$$

	0	10	20	30	30				

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
				?					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
						?			
				50					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot\,,\cdot\,] \quad \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

$$f[.,.]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[.,.]$$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\; f[m+k, n+l]$$

Credit: S. Seitz

Box Filter

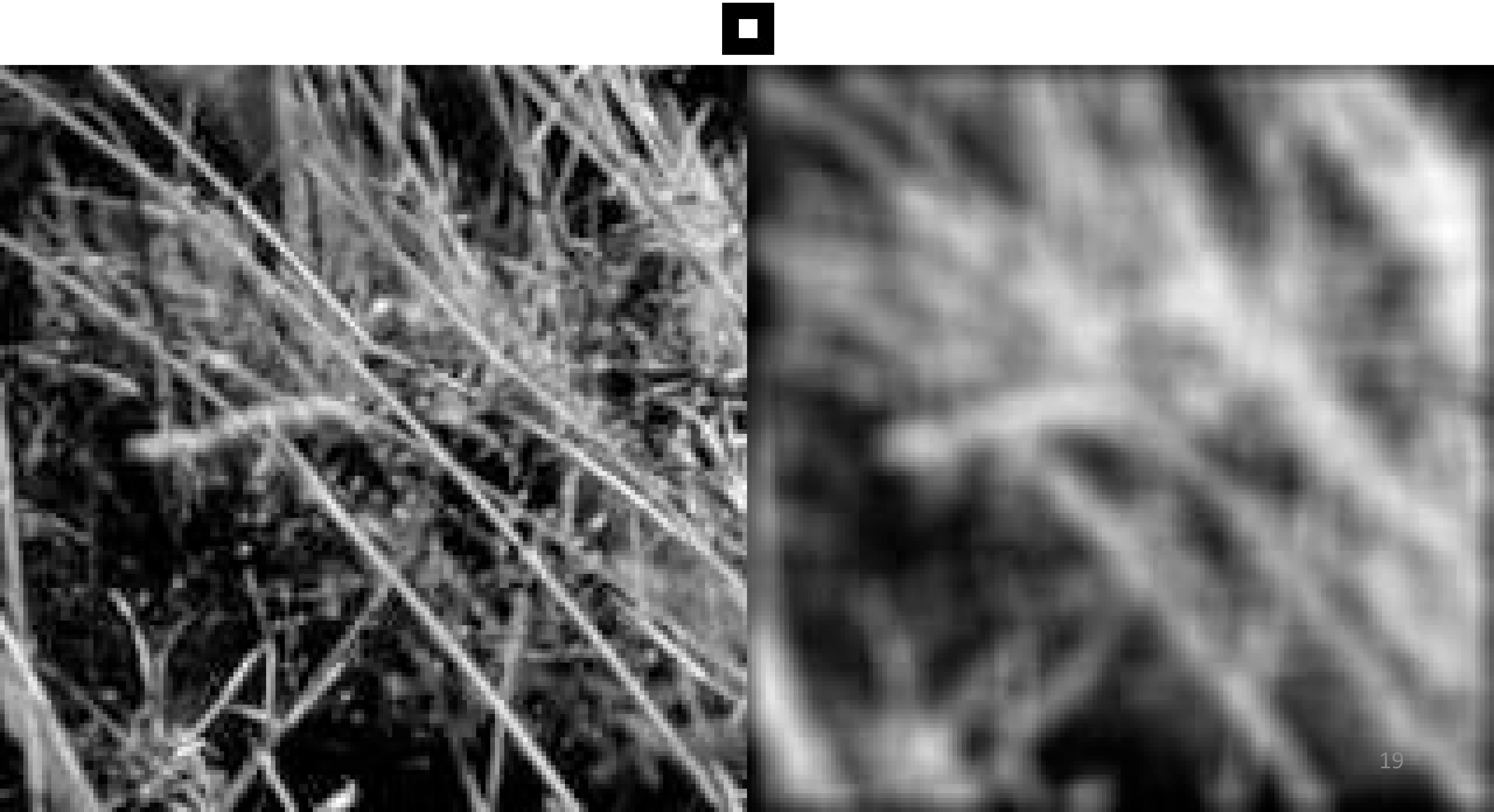
What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

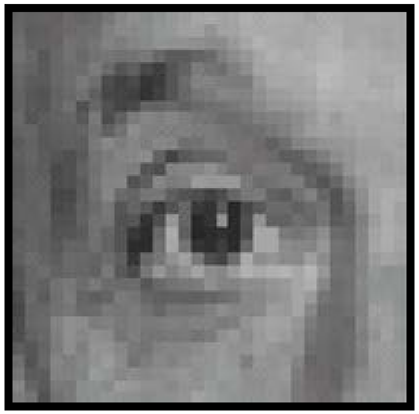
$$g[\cdot\,,\cdot\,] = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Slide credit: David Lowe (UBC)

Smoothing with box filter



Practice with linear filters



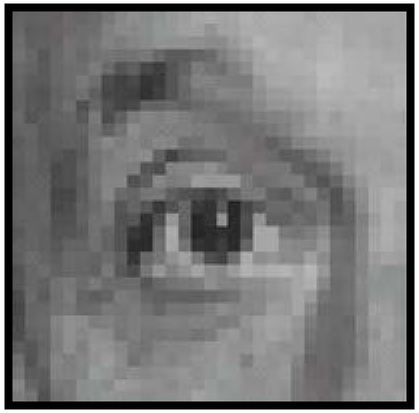
Original

0	0	0
0	1	0
0	0	0

?

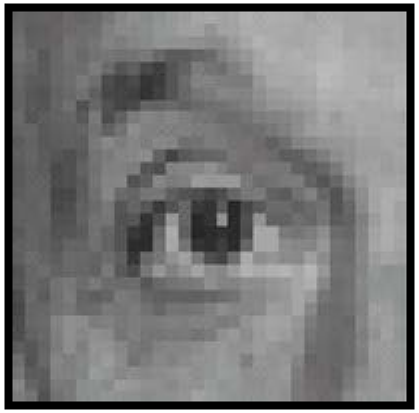
Source: D. Lowe

Practice with linear filters



Original

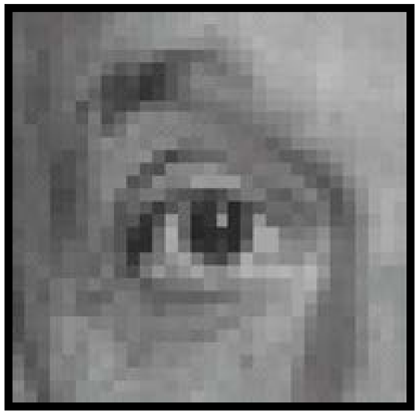
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

Practice with linear filters

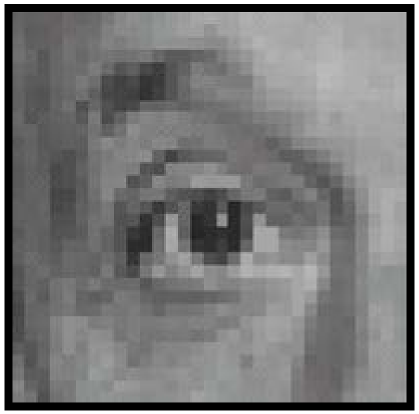

Original

0	0	0
0	0	1
0	0	0

?

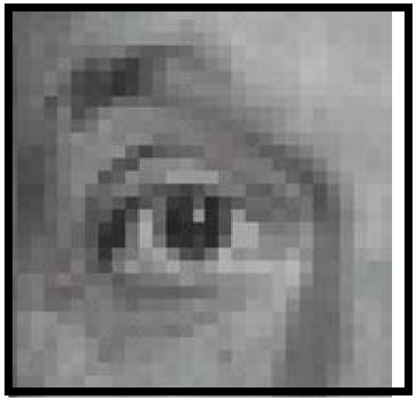
Source: D. Lowe

Practice with linear filters



Original

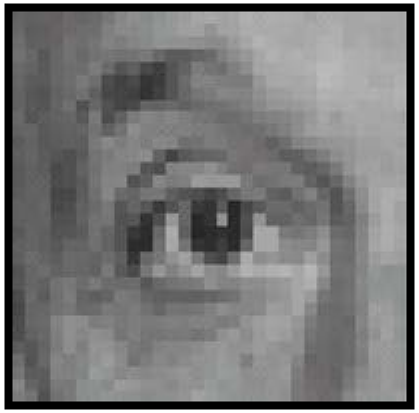
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$



Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters



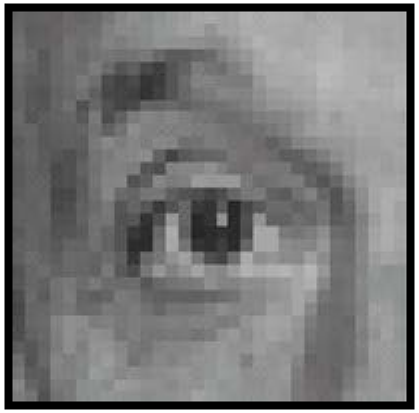
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

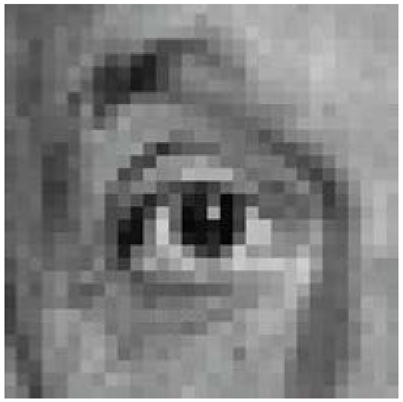
?

Source: D. Lowe

Practice with linear filters



$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$



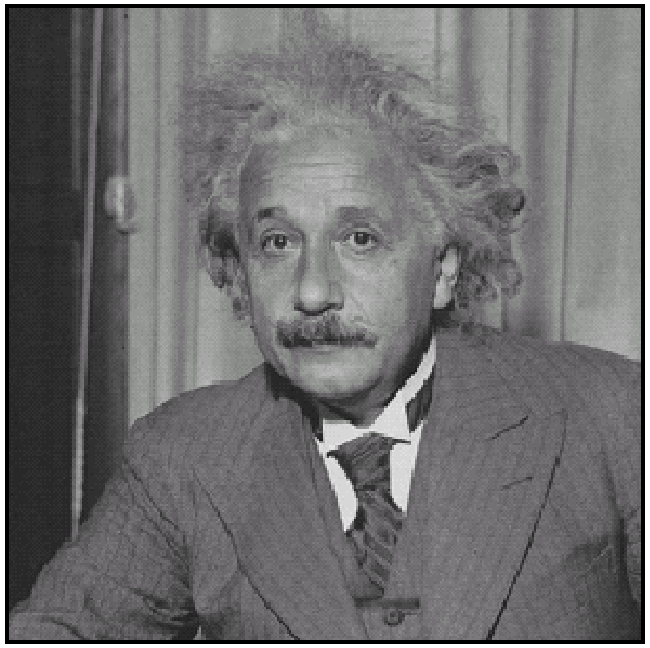
Original

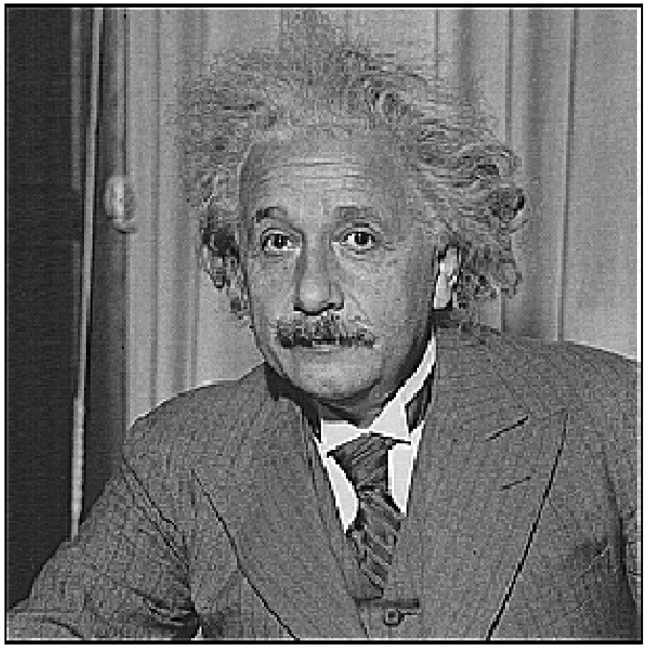
Sharpening filter

- Accentuates differences with local average

Source: D. Lowe

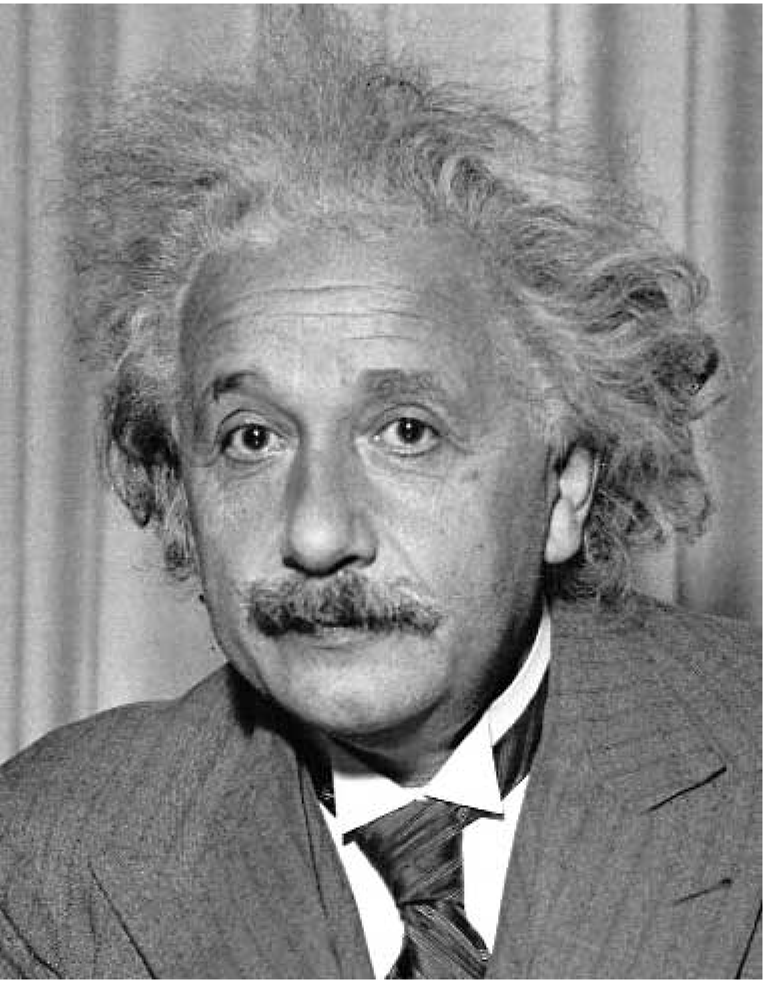
Sharpening


before


after

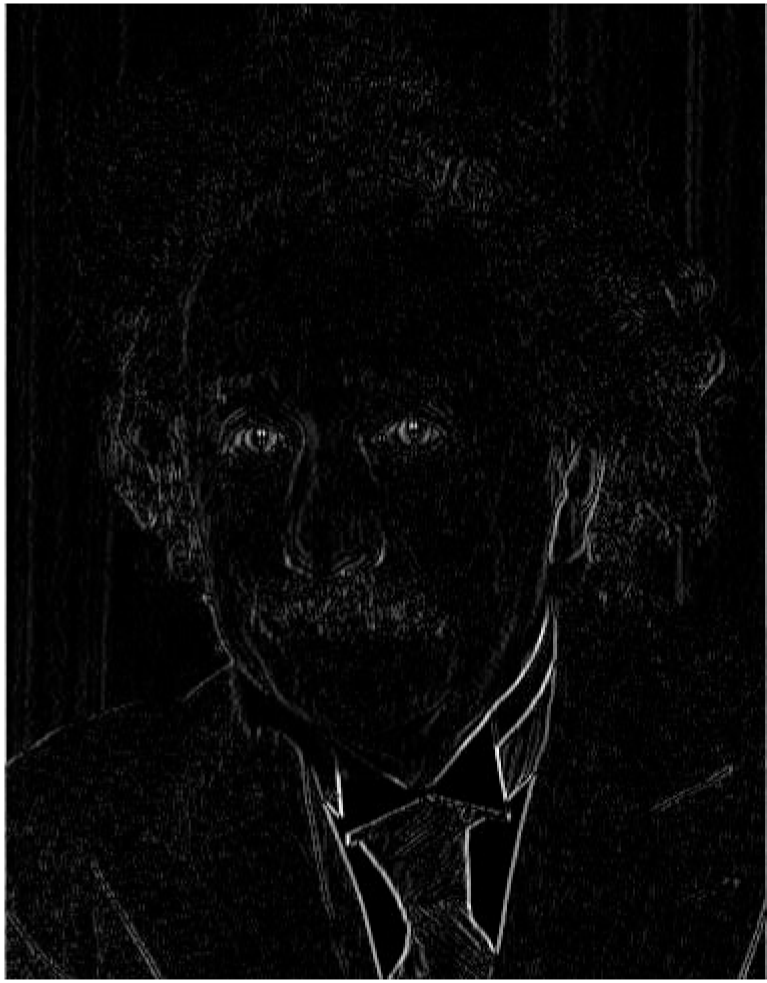
Source: D. Lowe

Other filters



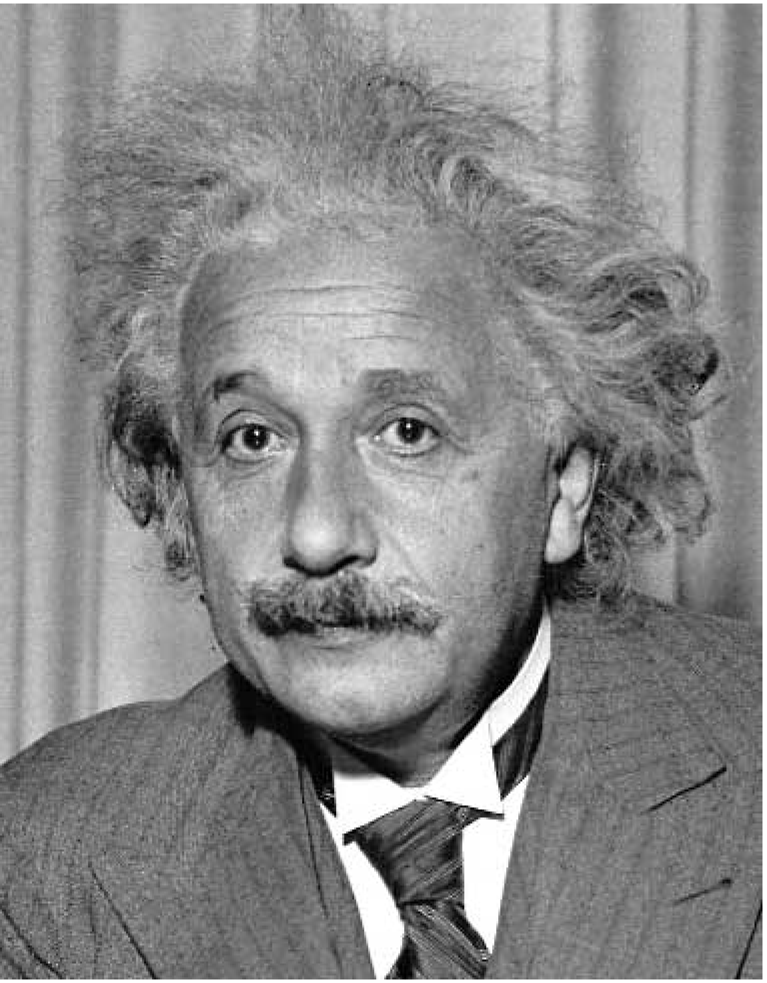
1	0	-1
2	0	-2
1	0	-1

Sobel



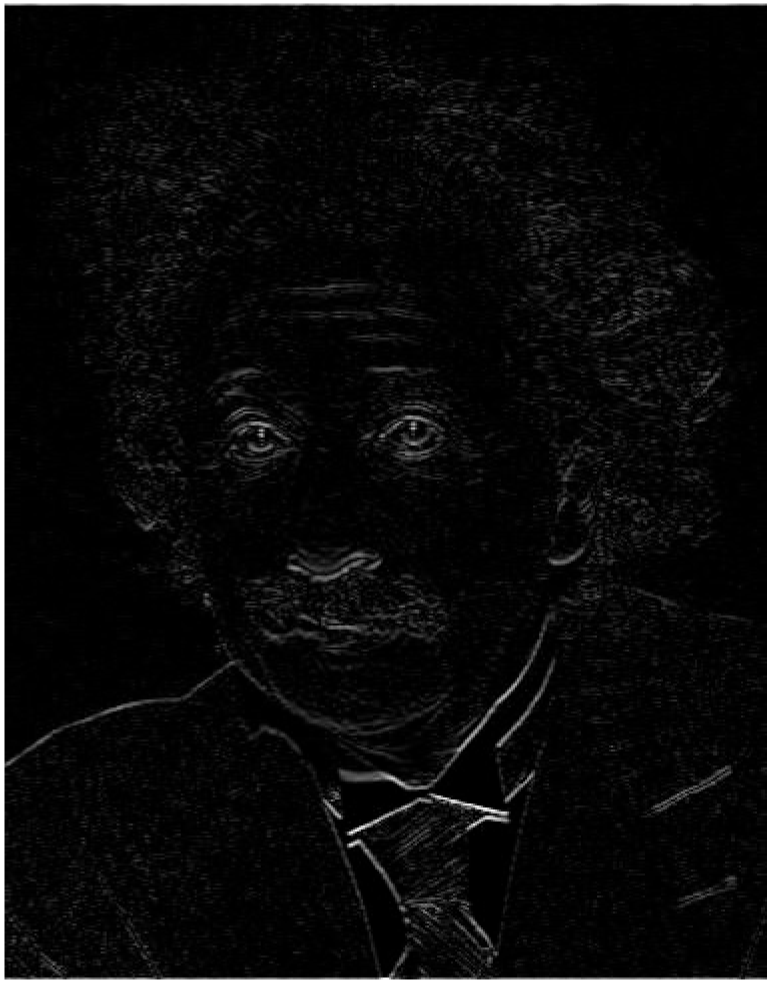
Vertical Edge
(absolute value)

Other filters



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

Sobel



Horizontal Edge
(absolute value)

Basic gradient filters

Horizontal Gradient

0	0	0
-1	0	1
0	0	0

or

-1	0	1

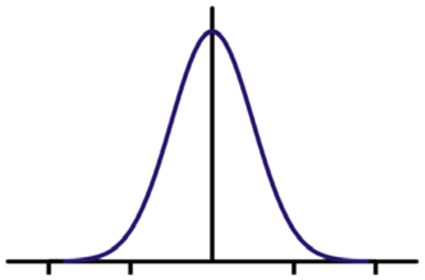
Vertical Gradient

0	1	0
0	0	0
0	-1	0

or

-1
0
1

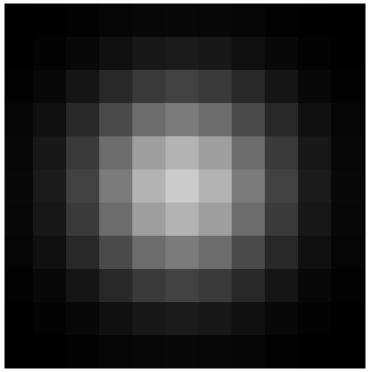
Gaussian filter



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

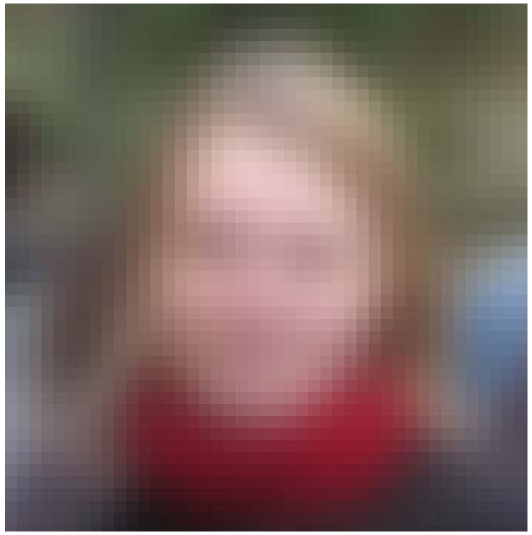
Input image f

$*$



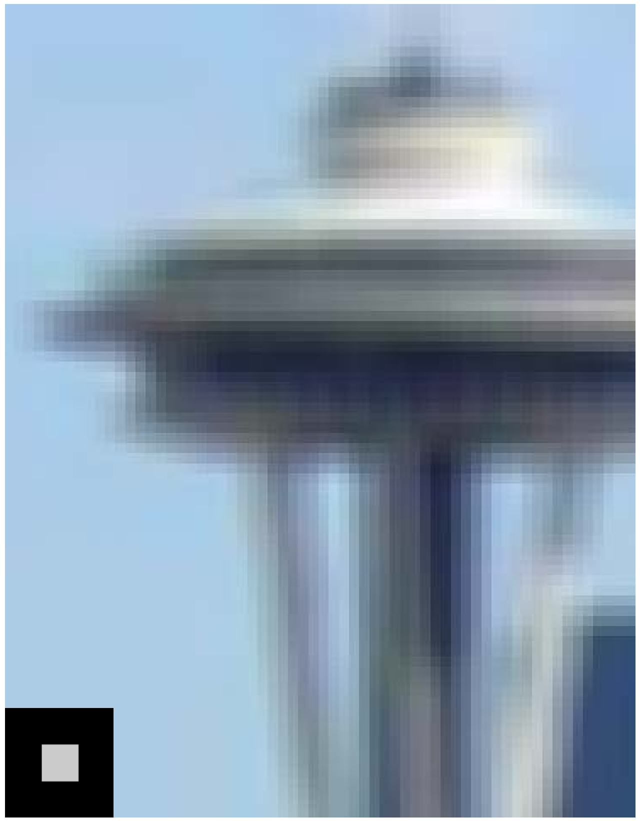
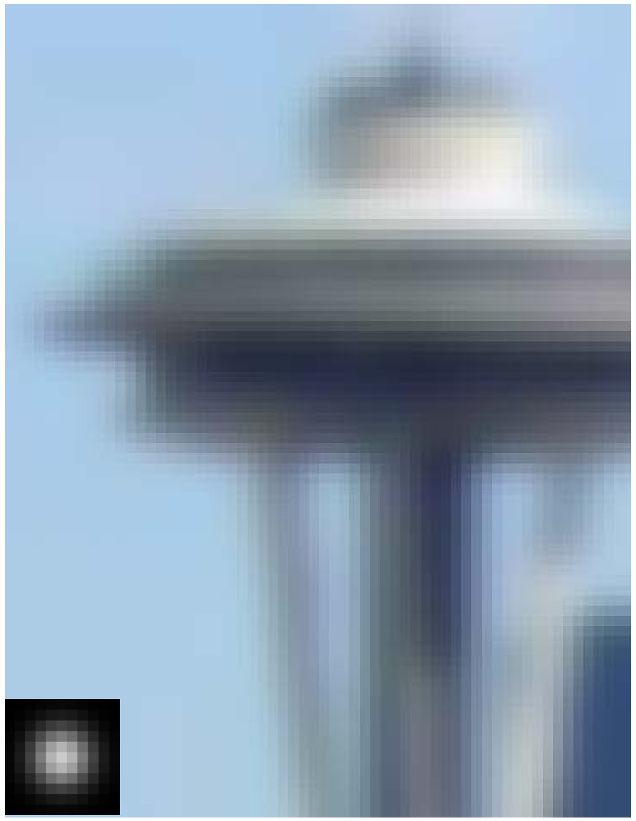
Filter h

$=$



Output image g

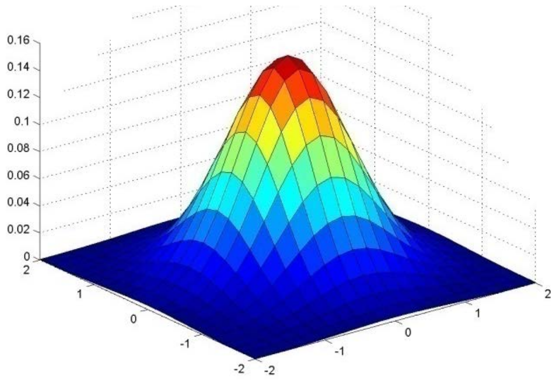
Gaussian vs. mean filters

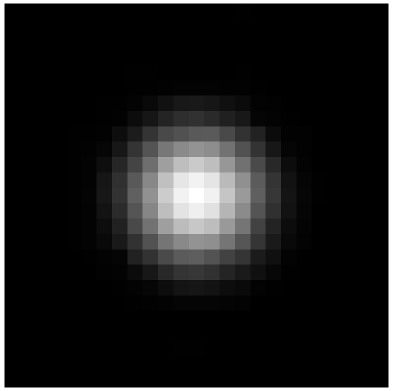



What does real blur look like?

Important filter: Gaussian

- Spatially-weighted average





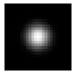
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

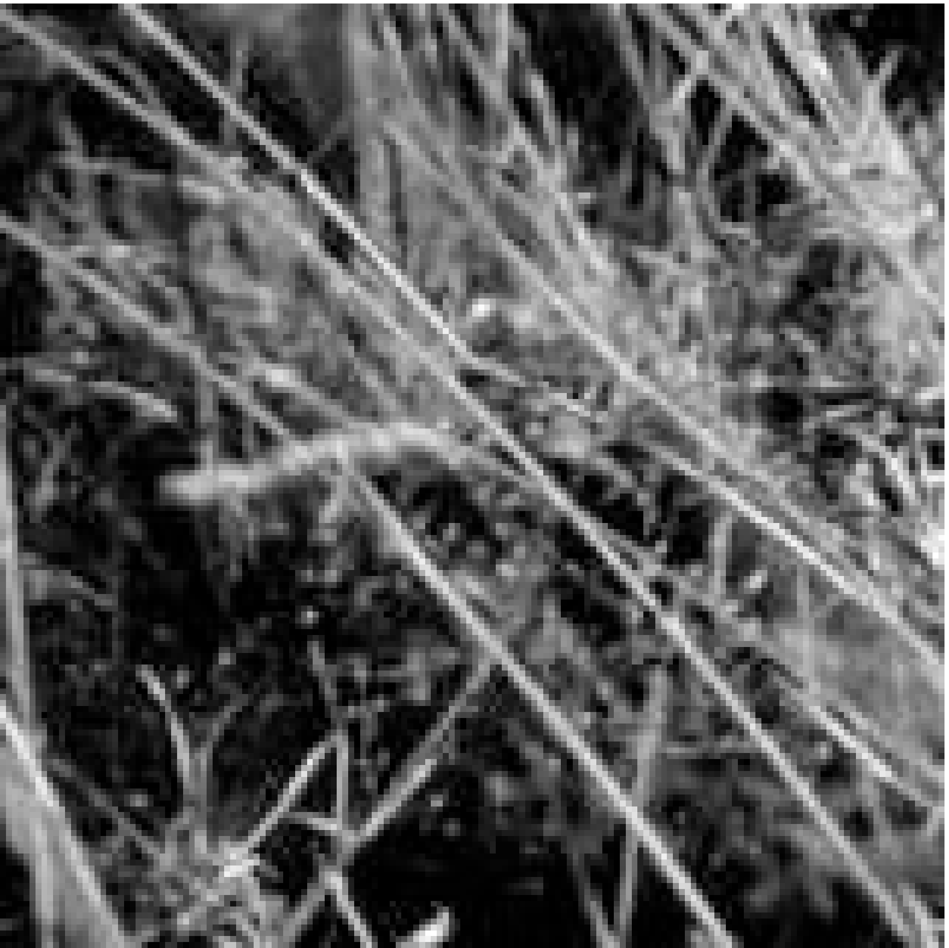
5 x 5, σ = 1

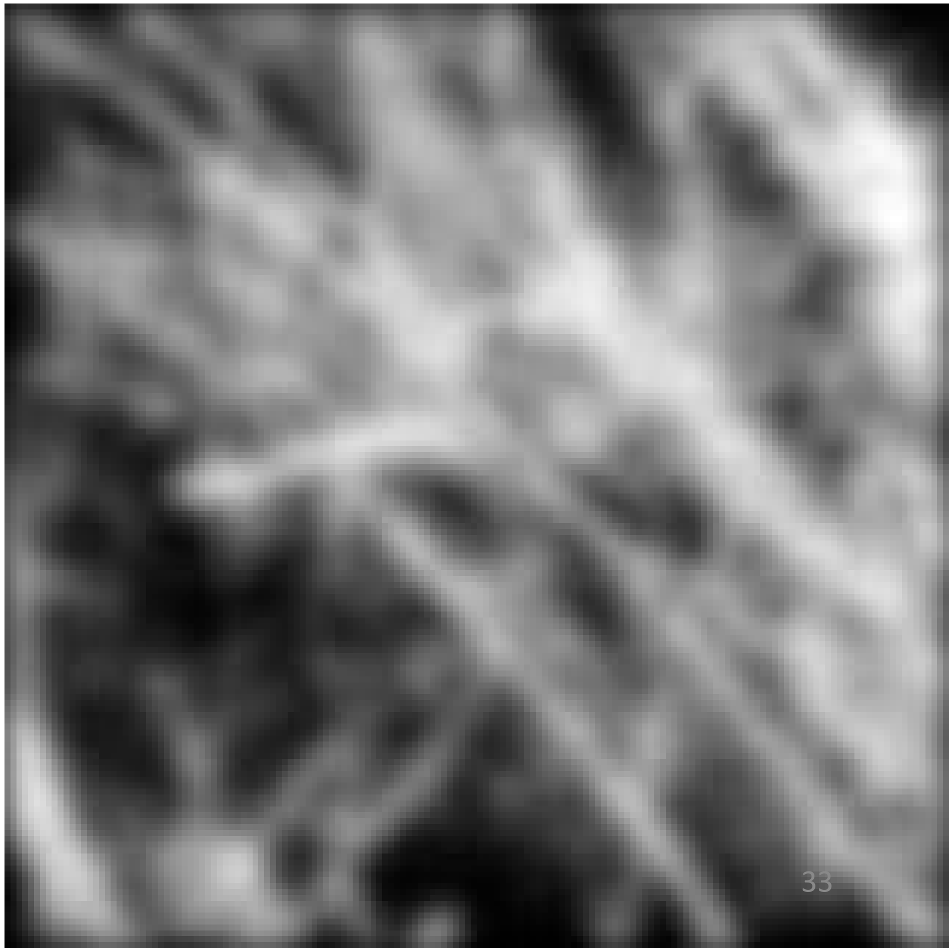
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

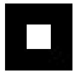
Smoothing with Gaussian filter

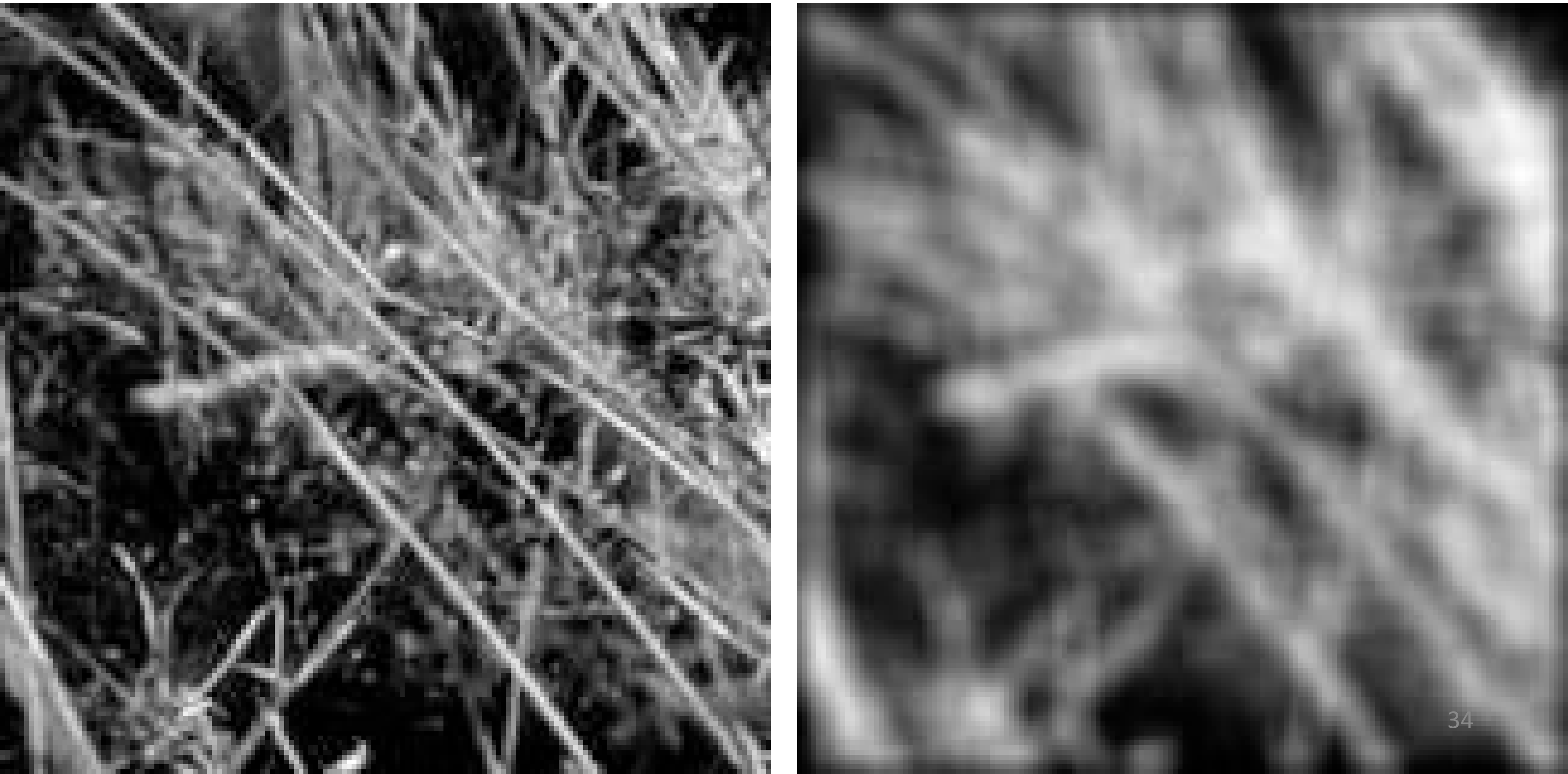






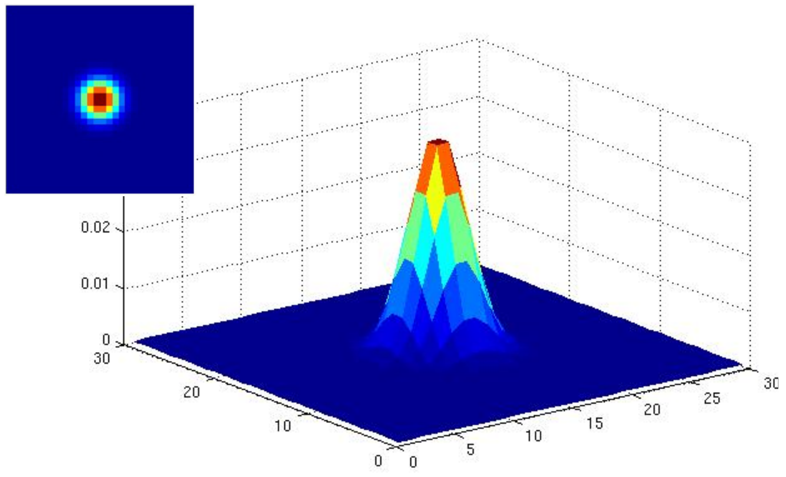
Smoothing with box filter

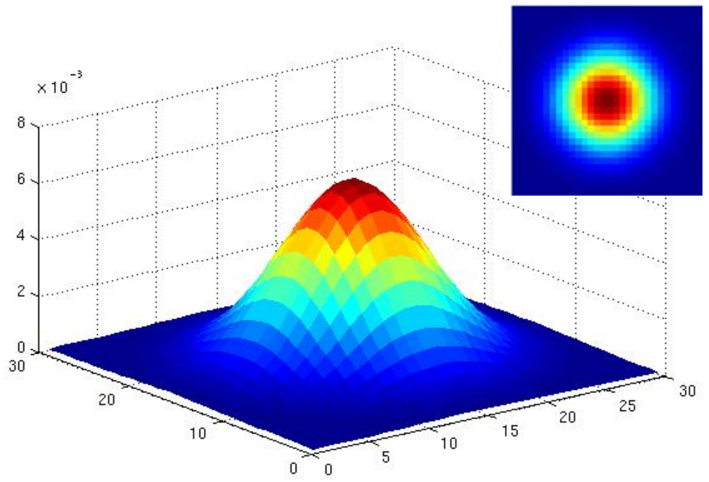




Gaussian filters

- What parameters matter here?
- **Variance** of Gaussian: determines extent of smoothing

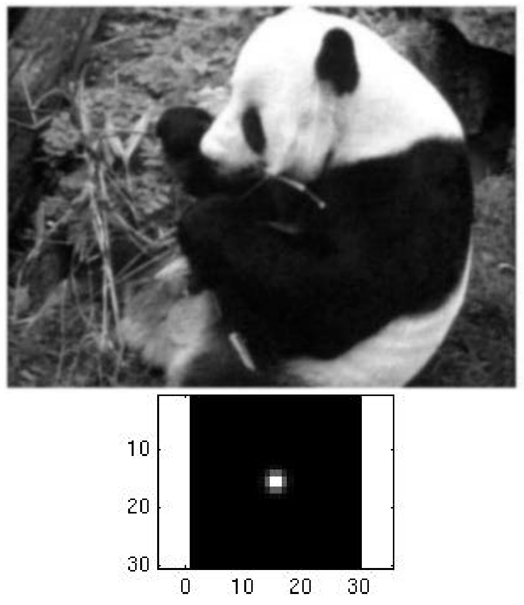


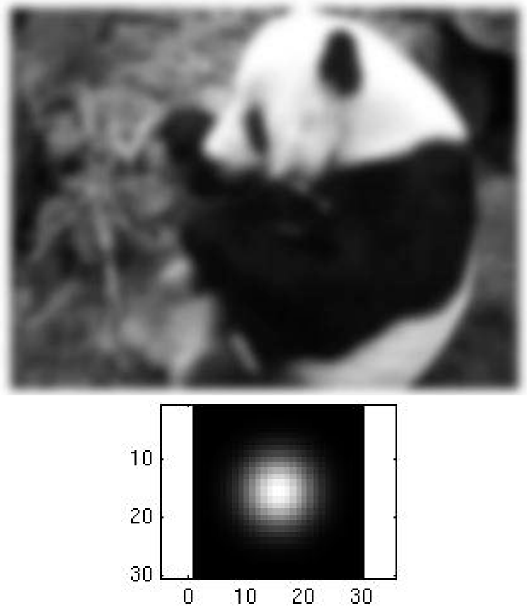


Source: K. Grauman

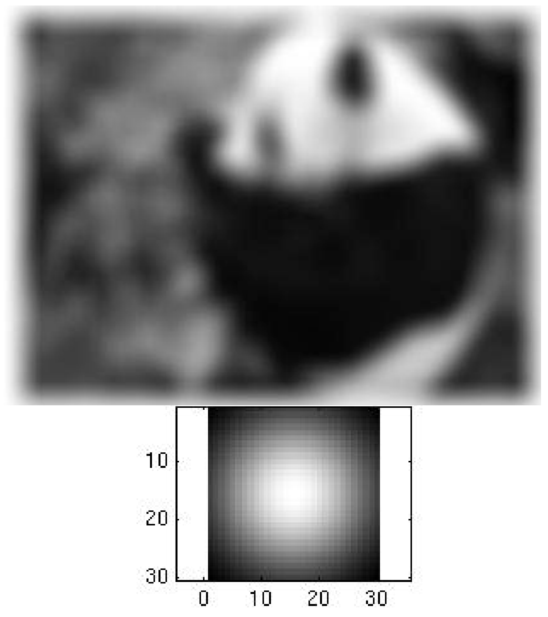
Smoothing with a Gaussian

Parameter σ is the "scale" / "width" / "spread" of the Gaussian kernel, and controls the amount of smoothing.



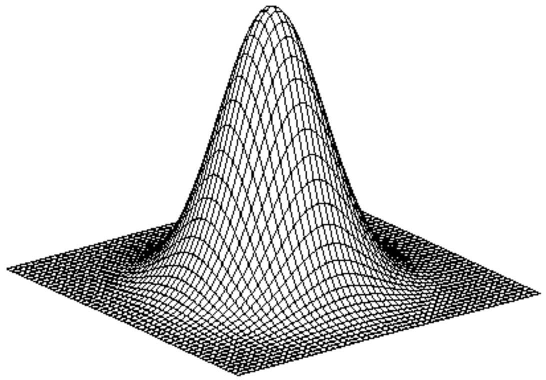


…



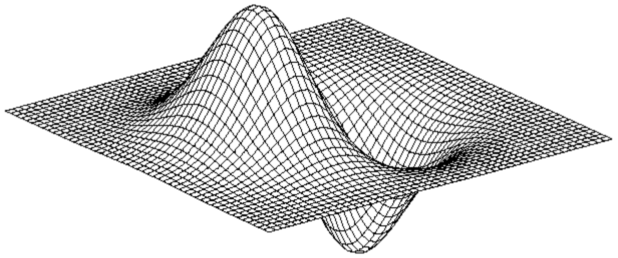
Source: K. Grauman

2D edge detection filters



Gaussian

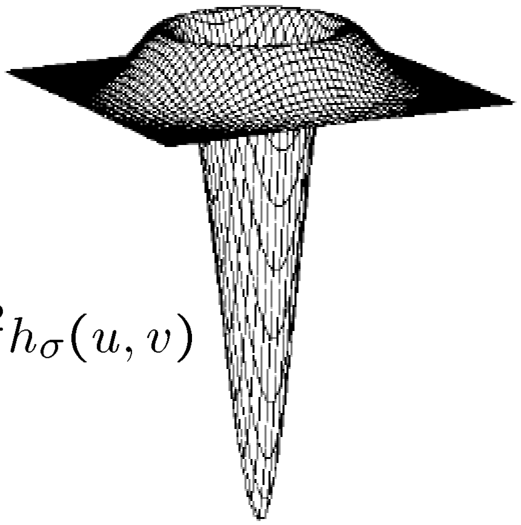
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



x derivative of Gaussian

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

Laplacian of Gaussian
or LoG filter



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

∇^2 is the **Laplacian** operator (sum of 2nd derivatives):

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

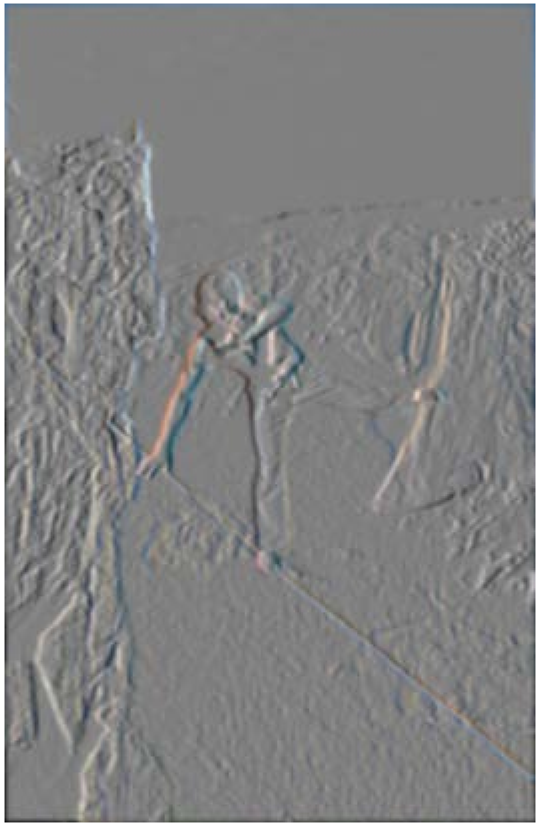
Often approximated by

0	1	0
1	-4	1
0	1	0

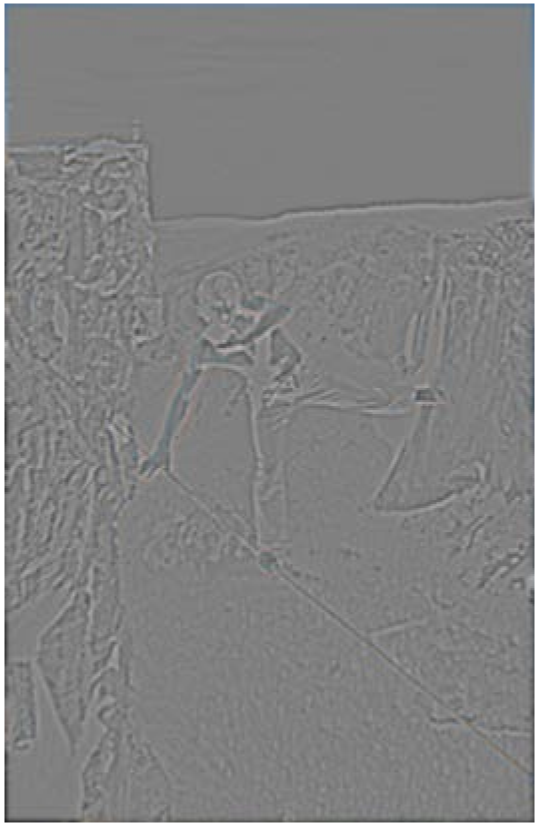
First and second derivatives

What are these good for?

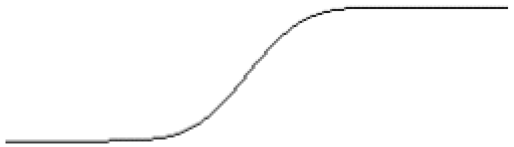
Original

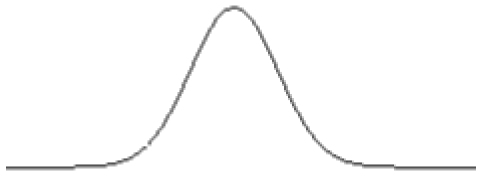


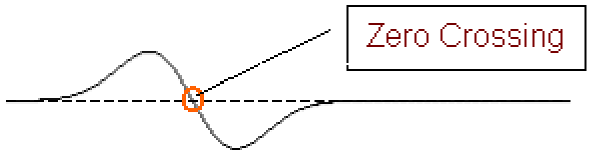
First Derivative x



Second Derivative x, y

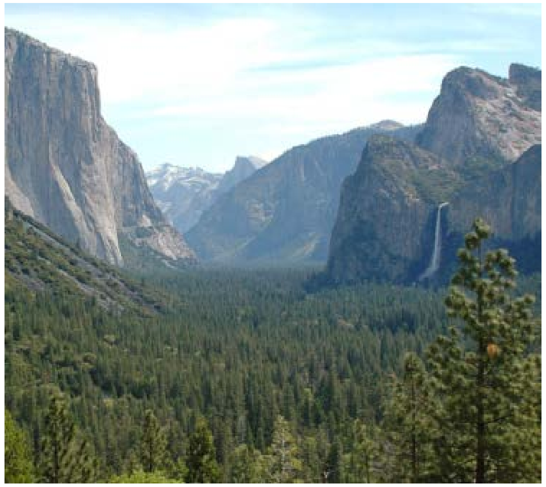


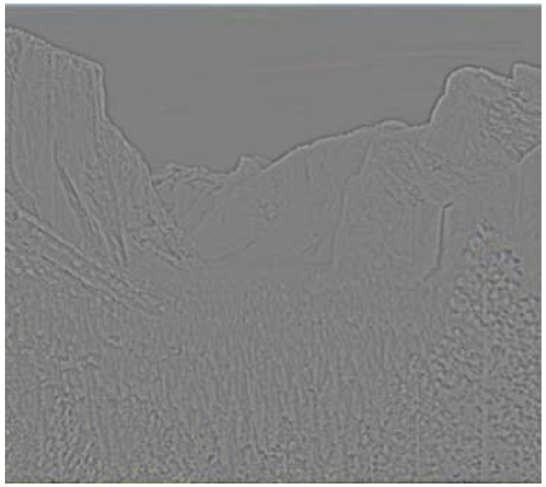


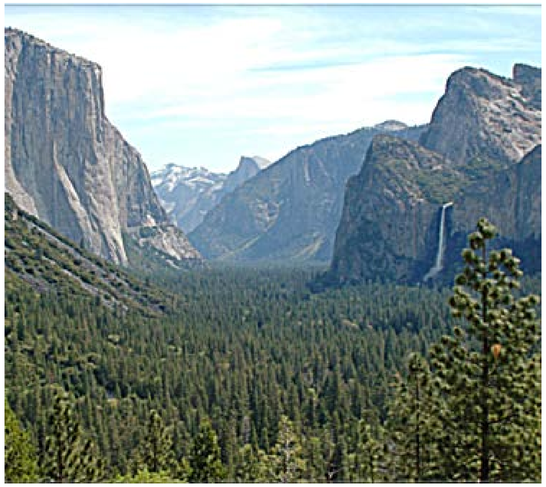


Subtracting filters

$$Sharpen(x,y) = f(x,y) - \alpha(f * \nabla^2 \mathcal{G}_\sigma(x,y))$$


Original


Second Derivative

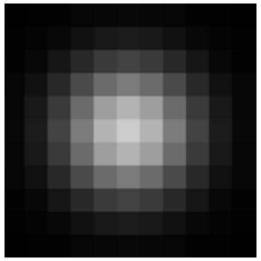

Sharpened

Combining filters

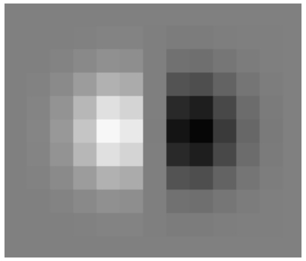
$$f * g * g' = f * h \text{ for some } h$$

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

$*$

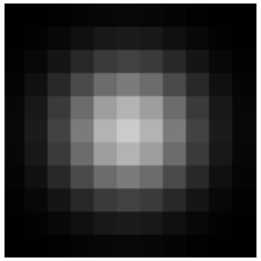


$=$

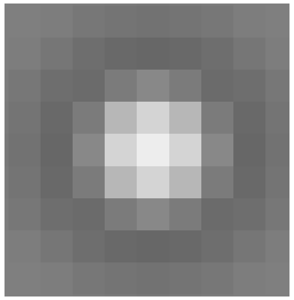


0	0	0	0	0
0	0	-1	0	0
0	-1	4	-1	0
0	0	-1	0	0
0	0	0	0	0

$*$



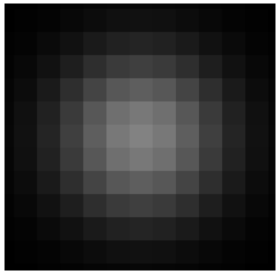
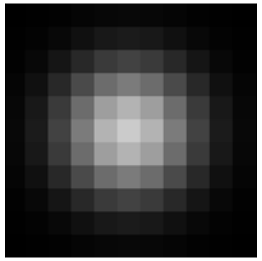
$=$



It's also true: $f * (g * h) = (f * g) * h$

$$f * g = g * f$$

Combining Gaussian filters

 *  = ?

$$f * \mathcal{G}_\sigma * \mathcal{G}_{\sigma'} = f * \mathcal{G}_{\sigma''}$$

$$\sigma'' = \sqrt{\sigma^2 + \sigma'^2}$$

More blur than either individually (but less than $\sigma'' = \sigma + \sigma'$)

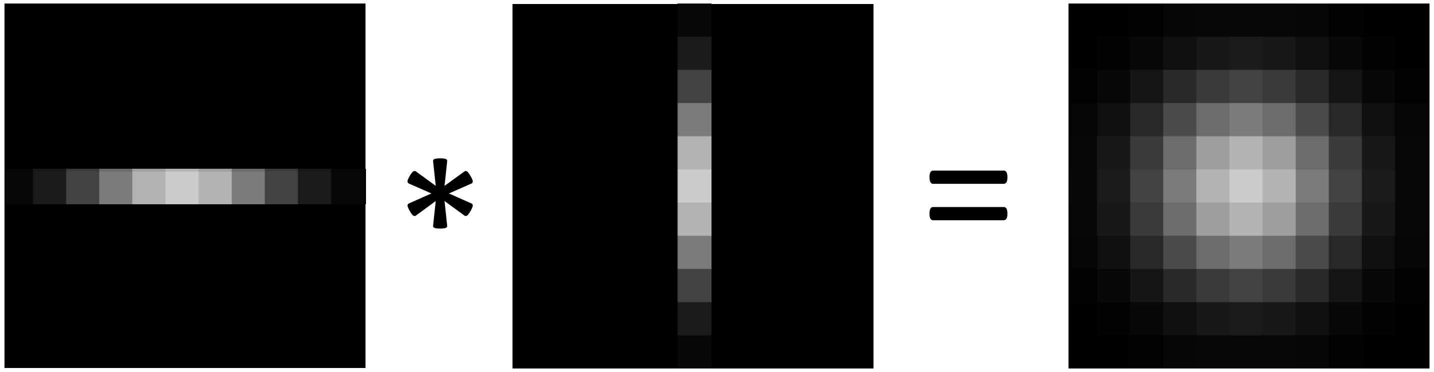
Separable filters

$$\mathcal{G}_\sigma = \mathcal{G}^x_\sigma * \mathcal{G}^y_\sigma$$

$$\mathcal{G}^x_\sigma(x,y) = \frac{1}{Z} e^{\frac{-(x^2)}{2\sigma^2}}$$

$$\mathcal{G}^y_\sigma(x,y) = \frac{1}{Z} e^{\frac{-(y^2)}{2\sigma^2}}$$

Compute Gaussian in horizontal direction, followed by the vertical direction. **Much faster!**



Not all filters are separable.

Freeman and Adelson, 1991

Sums of rectangular regions

If an image will be repeatedly convolved with different box filters, we can precompute a *summed area table*

$$s(i,j) = \sum_{k=0}^{i}\sum_{l=0}^{j} f(k,l)$$

How do we compute the sum of the pixels in the red box?

After some pre-computation, this can be done in constant time for any box.

This "trick" is commonly used for computing Haar wavelets (a fundemental building block of many object recognition approaches.)

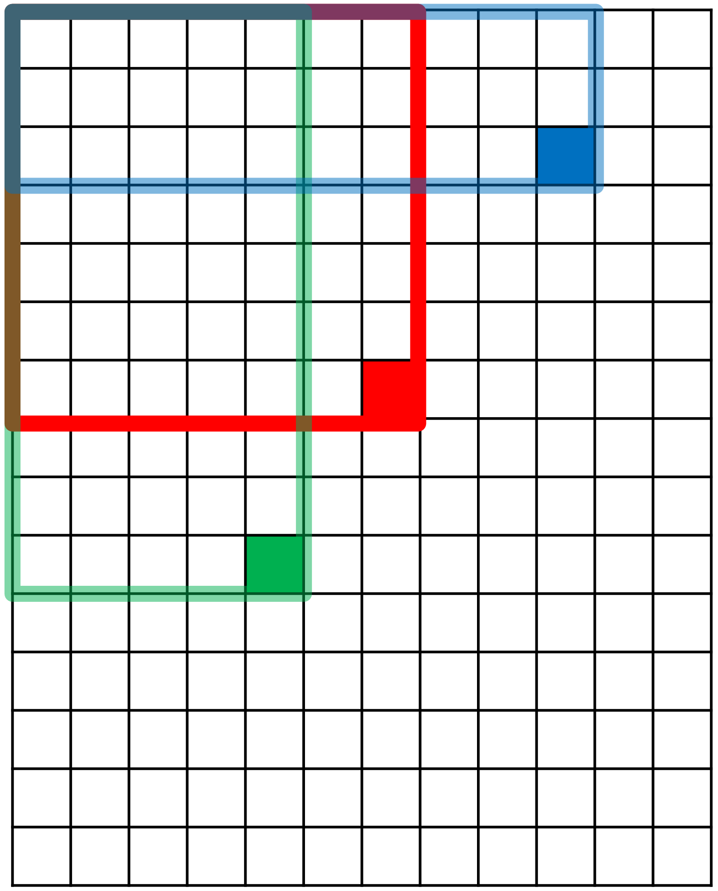
243	239	240	225	206	185	188	218	211	206	216	225
242	239	218	110	67	31	34	152	213	206	208	221
243	242	123	58	94	82	132	77	108	208	208	215
235	217	115	212	243	236	247	139	91	209	208	211
233	208	131	222	219	226	196	114	74	208	213	214
232	217	131	116	77	150	69	56	52	201	228	223
232	232	182	186	184	179	159	123	93	232	235	235
232	236	201	154	216	133	129	81	175	252	241	240
235	238	230	128	172	138	65	63	234	249	241	245
237	236	247	143	59	78	10	94	255	248	247	251
234	237	245	193	55	33	115	144	213	255	253	251
248	245	161	128	149	109	138	65	47	156	239	255
190	107	39	102	94	73	114	58	17	7	51	137
23	32	33	148	168	203	179	43	27	17	12	8
17	26	12	160	255	255	109	22	26	19	35	24

Sums of rectangular regions

The trick is to compute an "integral image." Every pixel is the sum of itself and its neighbors to the upper left.

Sequentially compute using:

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



Sums of rectangular regions

The trick is to compute an “integral image.” Every pixel is the sum of itself and its (modified) N and W neighbors minus its (modified) NW neighbor.

1	2	3
4	5	6
7	8	9

1

1	3

1	3	6

Compute sequentially using:

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

1	3	6
5		

1	3	6
5	?	

5+5+3-1=12

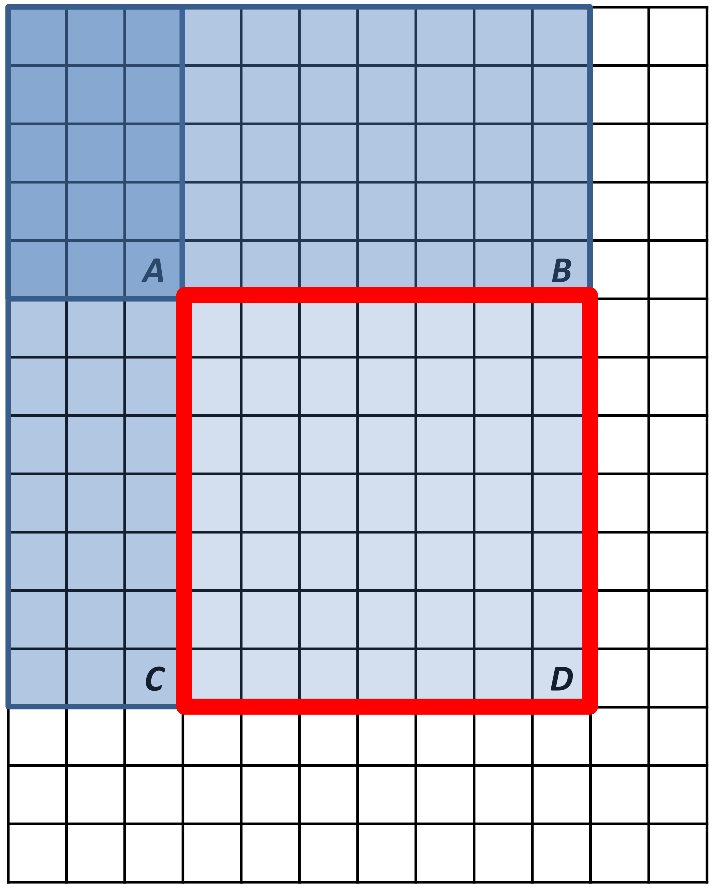
Result:

1	3	6
5	12	21
12	27	45

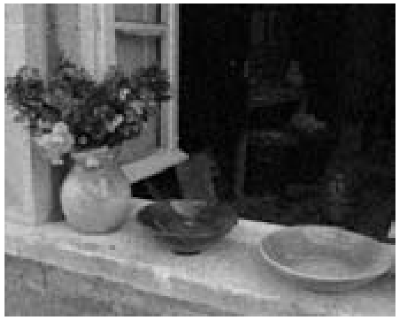
Sums of rectangular regions

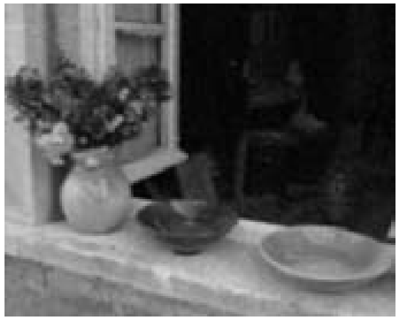
Area of red rectangle is found using:

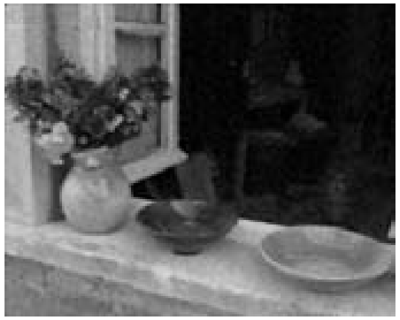
$$A + D - B - C$$

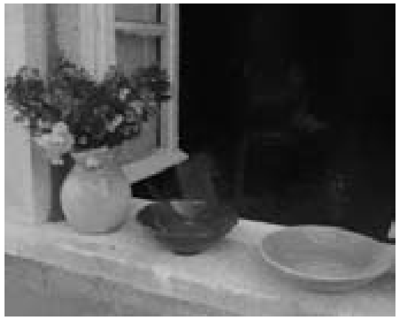


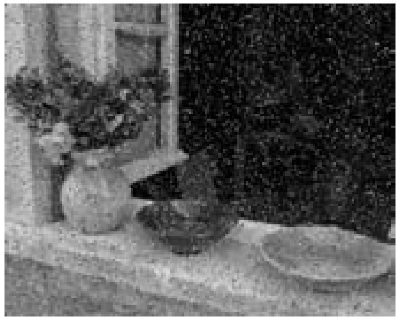
Linear vs. Non-Linear Filters

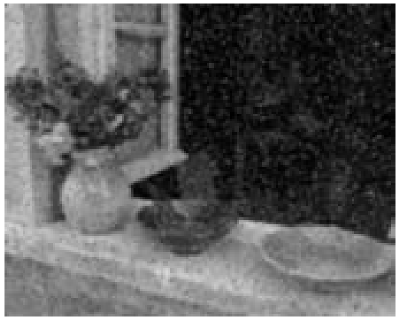

(a)

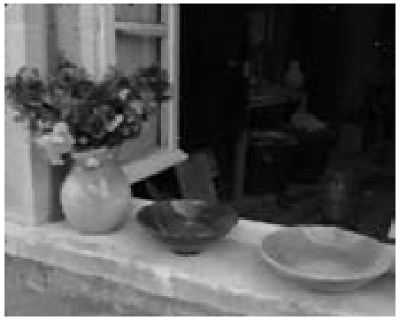

(b)

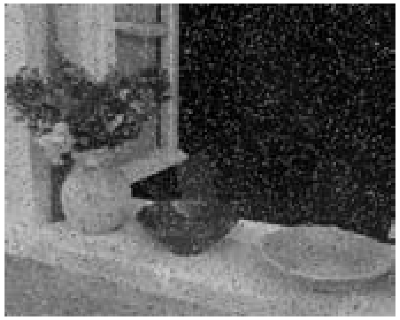

(c)


(d)


(e)


(f)


(g)


(h)

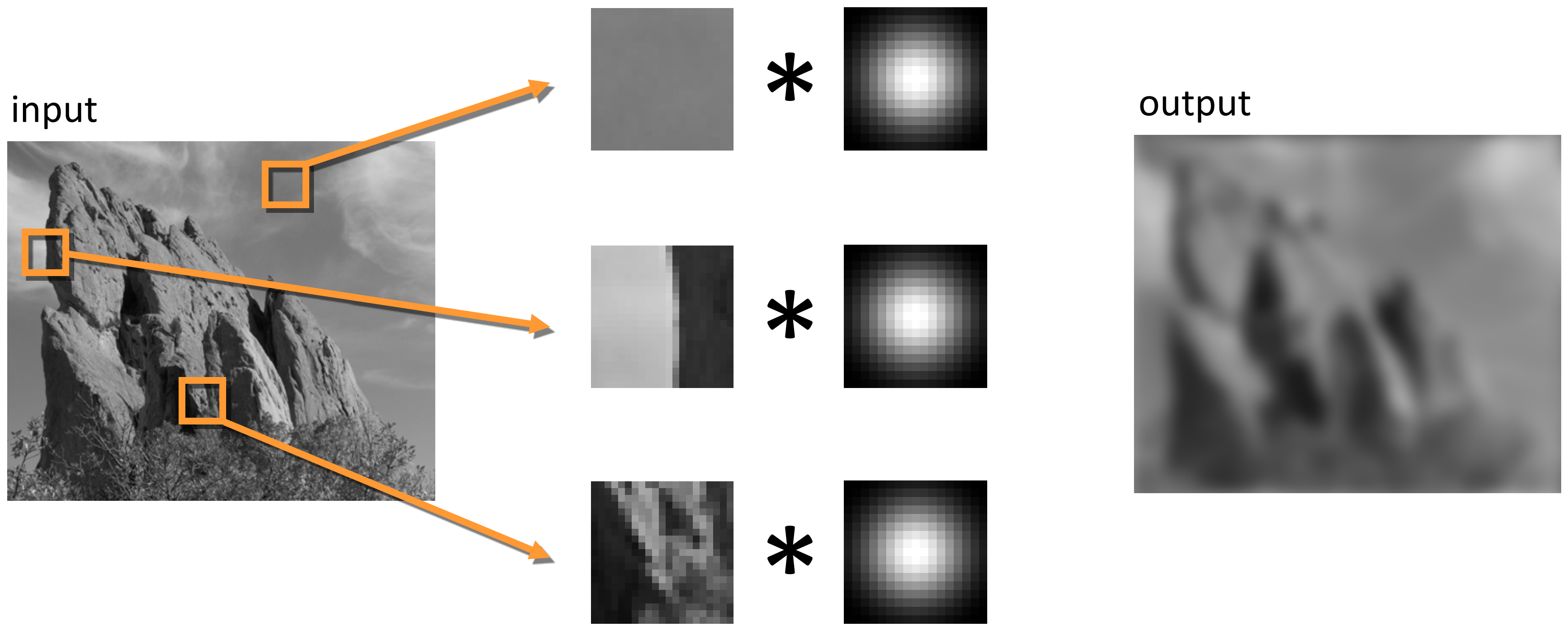
a. original image with Gaussian noise, b. Gaussian filtered, c. median filtered, d. bilateral filtered
e. original image with shot noise, f. Gaussian filtered, g. median filtered, h. bilateral filtered

Spatially varying filters

- Some filters vary spatially.
- The bilateral filter is the product of a domain kernel (Gaussian) and a data dependent range kernel.
- $d(i,j,k,l) = \exp[(-(i-k)^2+(j-l)^2)/2\sigma_d^2]$ is the domain kernel
- $r(i,j,k,l) = \exp[-||f(i,j)-f(k,l)||^2/2\sigma_r^2]$ is the range kernel
- $w(i,j,k,l) = d(i,j,k,l) * r(i,j,k,l)$ is their product
- $g(i,j) = \Sigma_{k,l}\, f(k,l)\, w(i,j,k,l) \,/\, \Sigma_{k,l}\, w(i,j,k,l)$ is the bilateral filter

from Szeliski text

Constant blur: same kernel everywhere

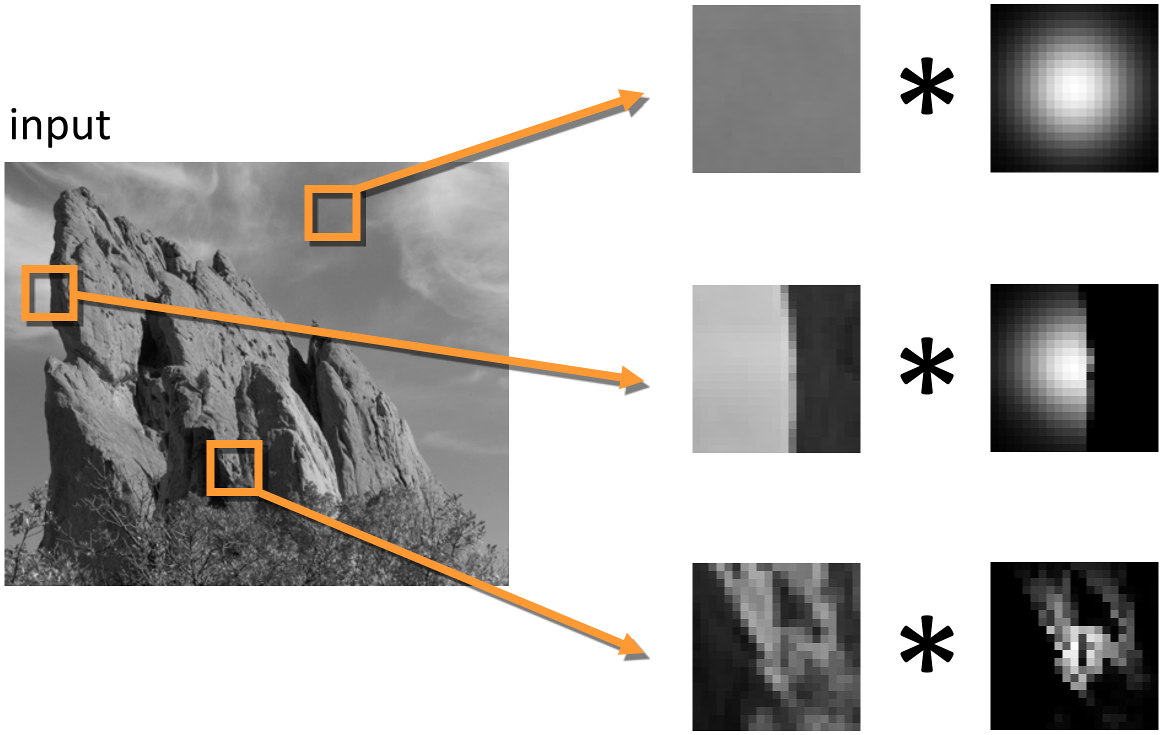


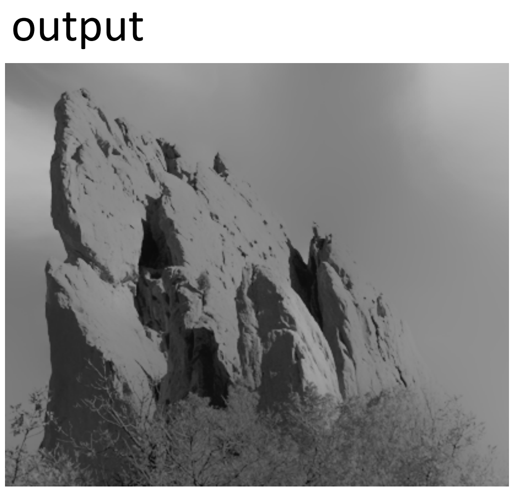
Same Gaussian kernel everywhere.

Slides courtesy of Sylvian Paris

Bilateral filter: kernel depends on intensity

Maintains edges when blurring!



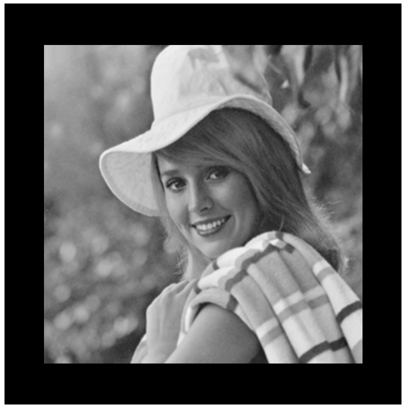


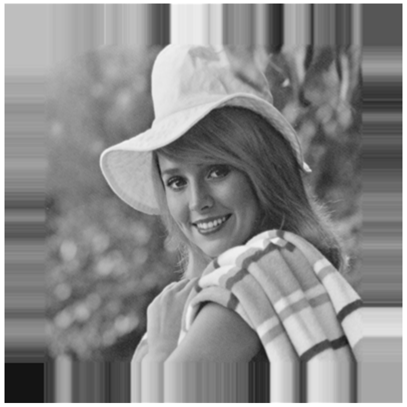
The kernel shape depends on the image content.

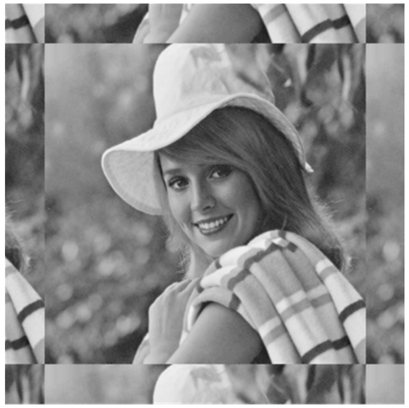
Slides courtesy of Sylvian Paris

Borders

What to do about image borders:


black


fixed


periodic

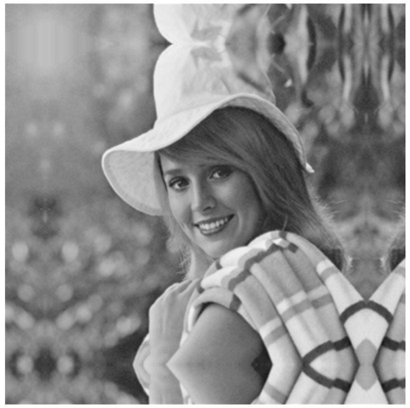
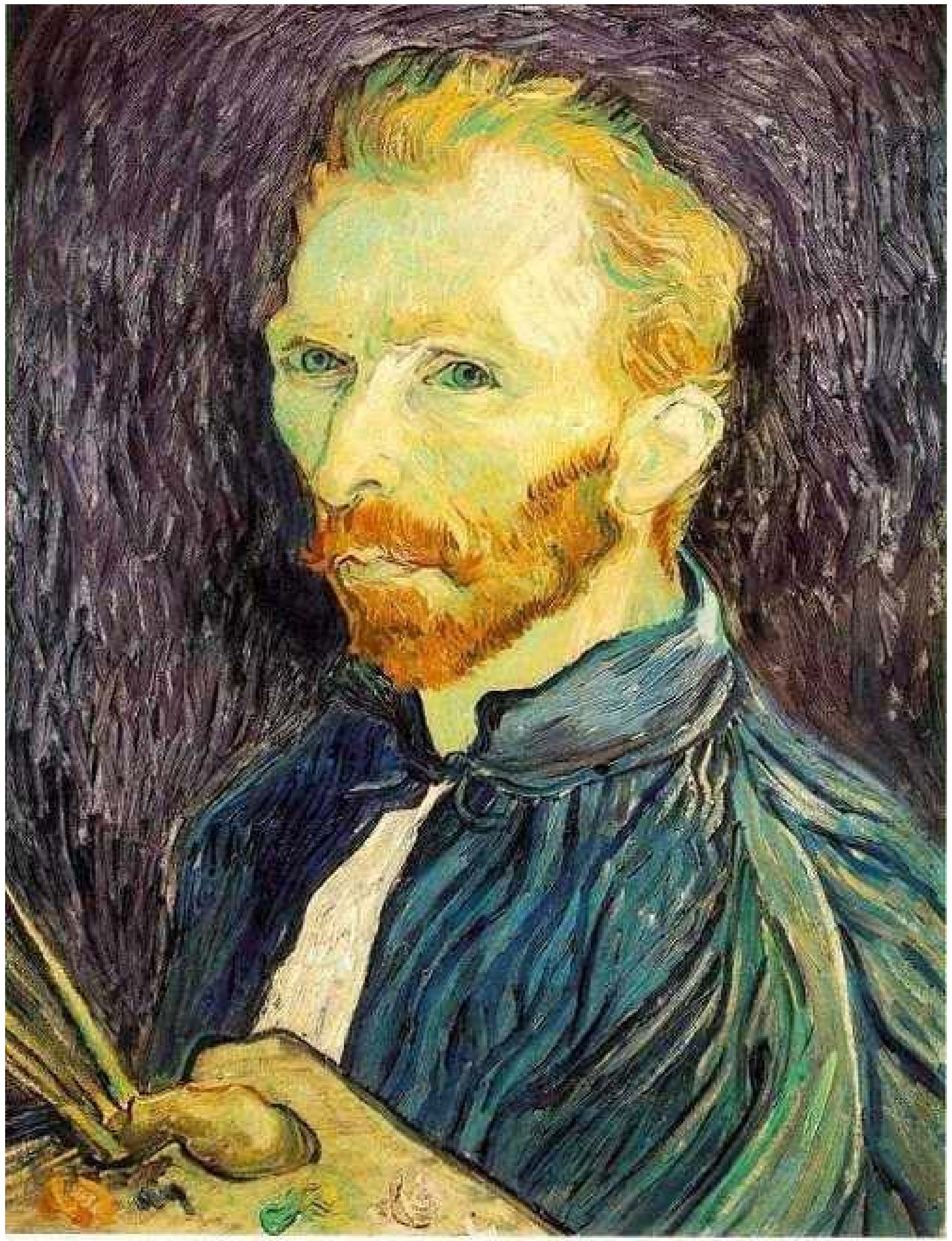

reflected

Image Sampling

F( ) =

F( ) =

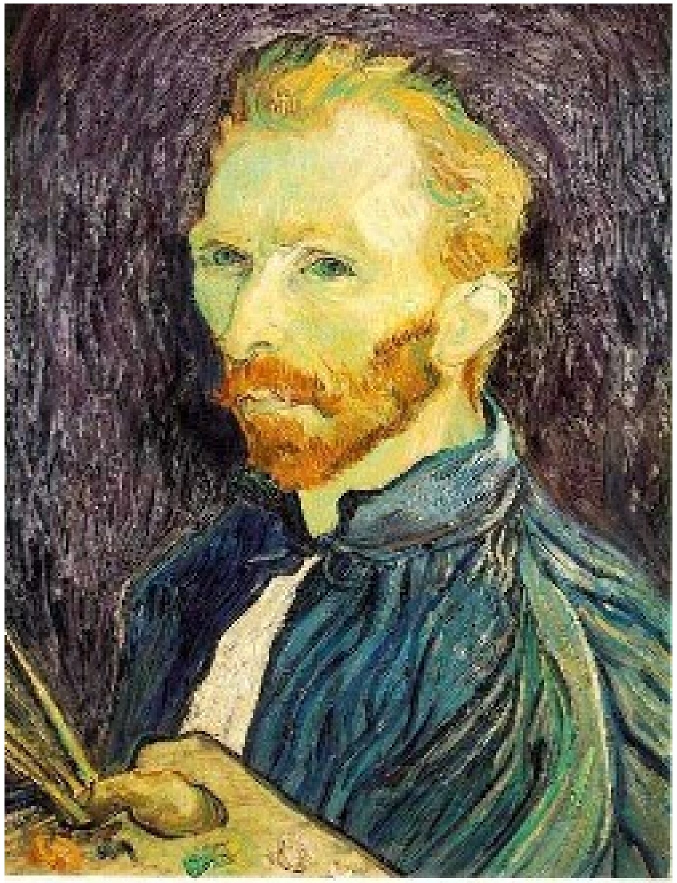
Image Scaling

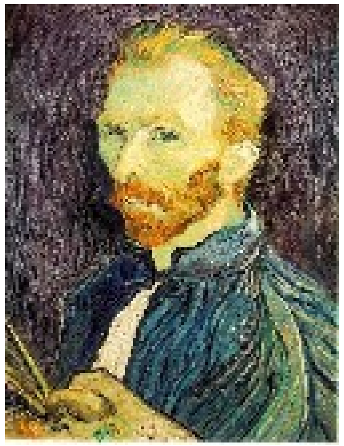


This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?

Image sub-sampling



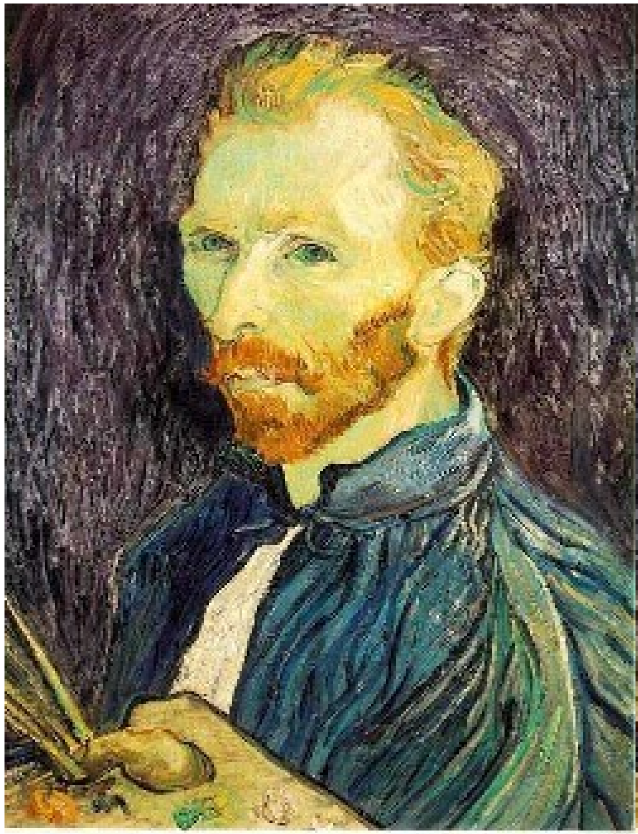


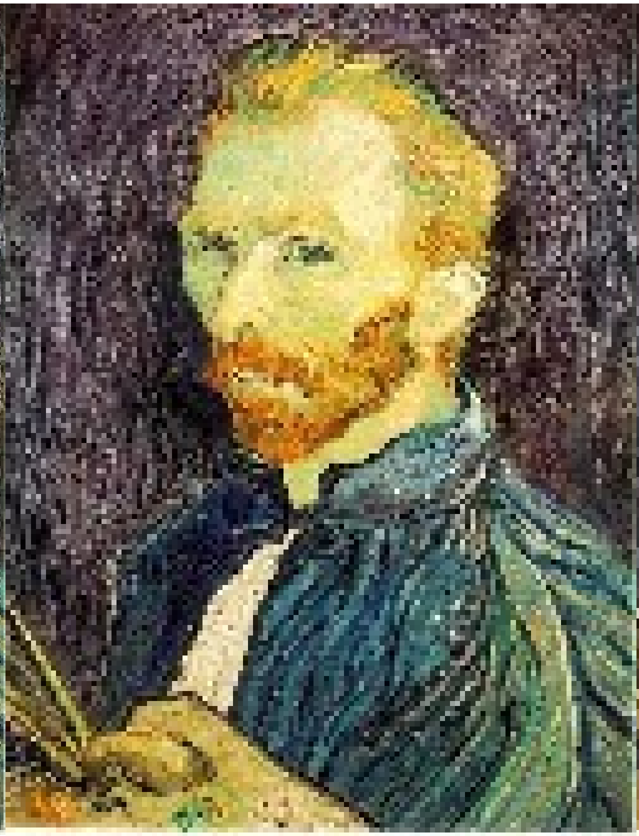
1/4

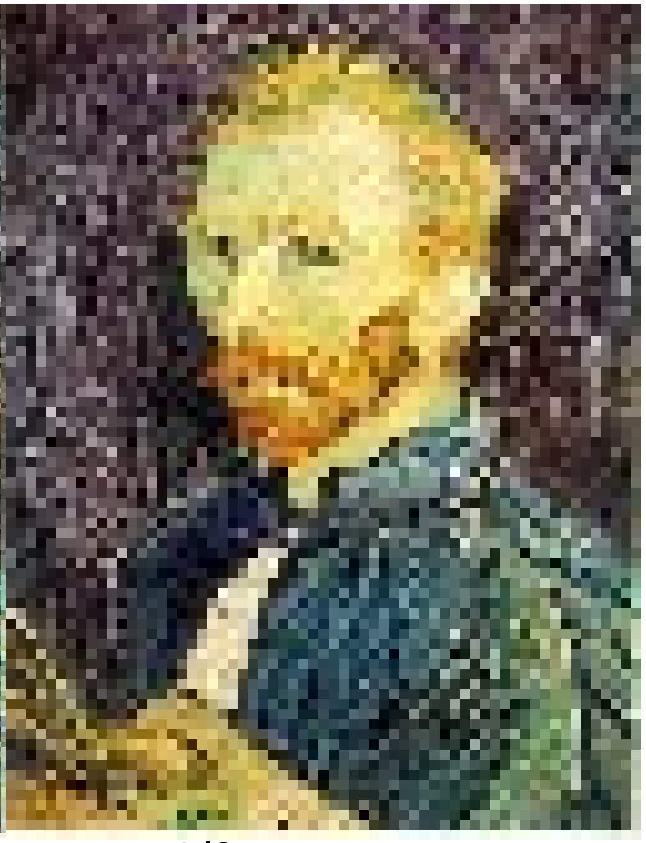
1/8

Throw away every other row and column to create a 1/2 size image
- called *image sub-sampling*

Image sub-sampling

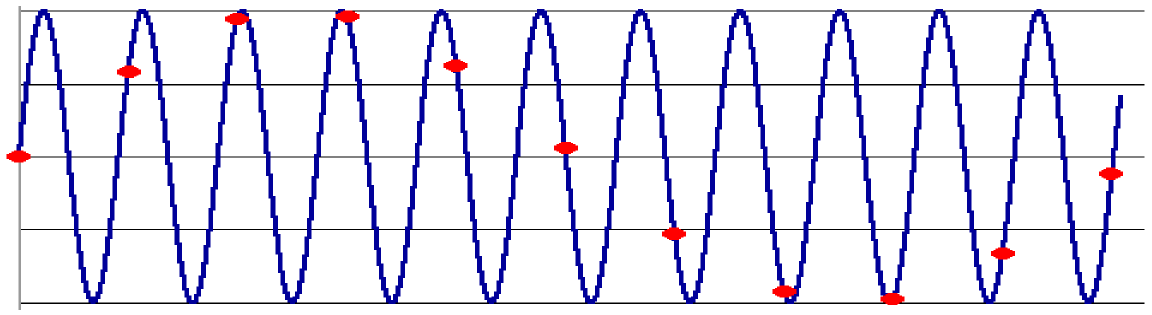

1/2


1/4 (2x zoom)


1/8 (4x zoom)

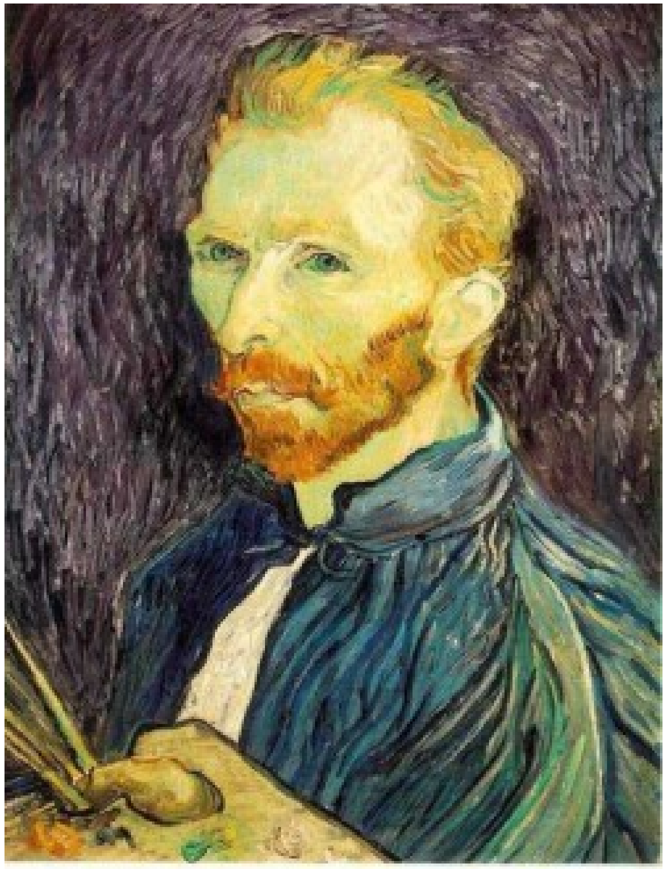
Why does this look so bad?

Down-sampling

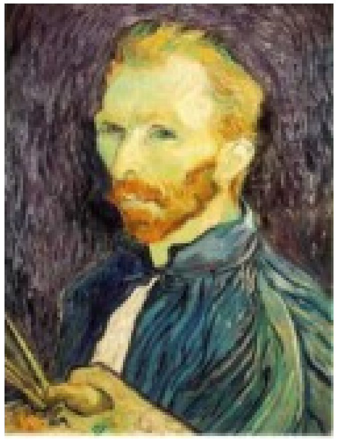


- **Aliasing** can arise when you sample a continuous signal or image
  - occurs when your sampling rate is not high enough to capture the amount of detail in your image
  - Can give you the wrong signal/image—an *alias*
  - formally, the image contains structure at different scales
    - called "frequencies" in the Fourier domain
  - the sampling rate must be high enough to capture the highest frequency in the image

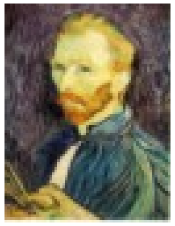
Subsampling with Gaussian pre-filtering



Gaussian 1/2



G 1/4



G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction.

Finale

- Filtering is just applying a mask to an image.
- Computer vision people call the linear form of these operations "convolutions". They are actually "correlations," since the true convolution inverts the mask.
- There are many nonlinear filters, too, such as median filters and morphological filters.
- Filtering is the lowest level of image analysis and is taught heavily in image processing courses.