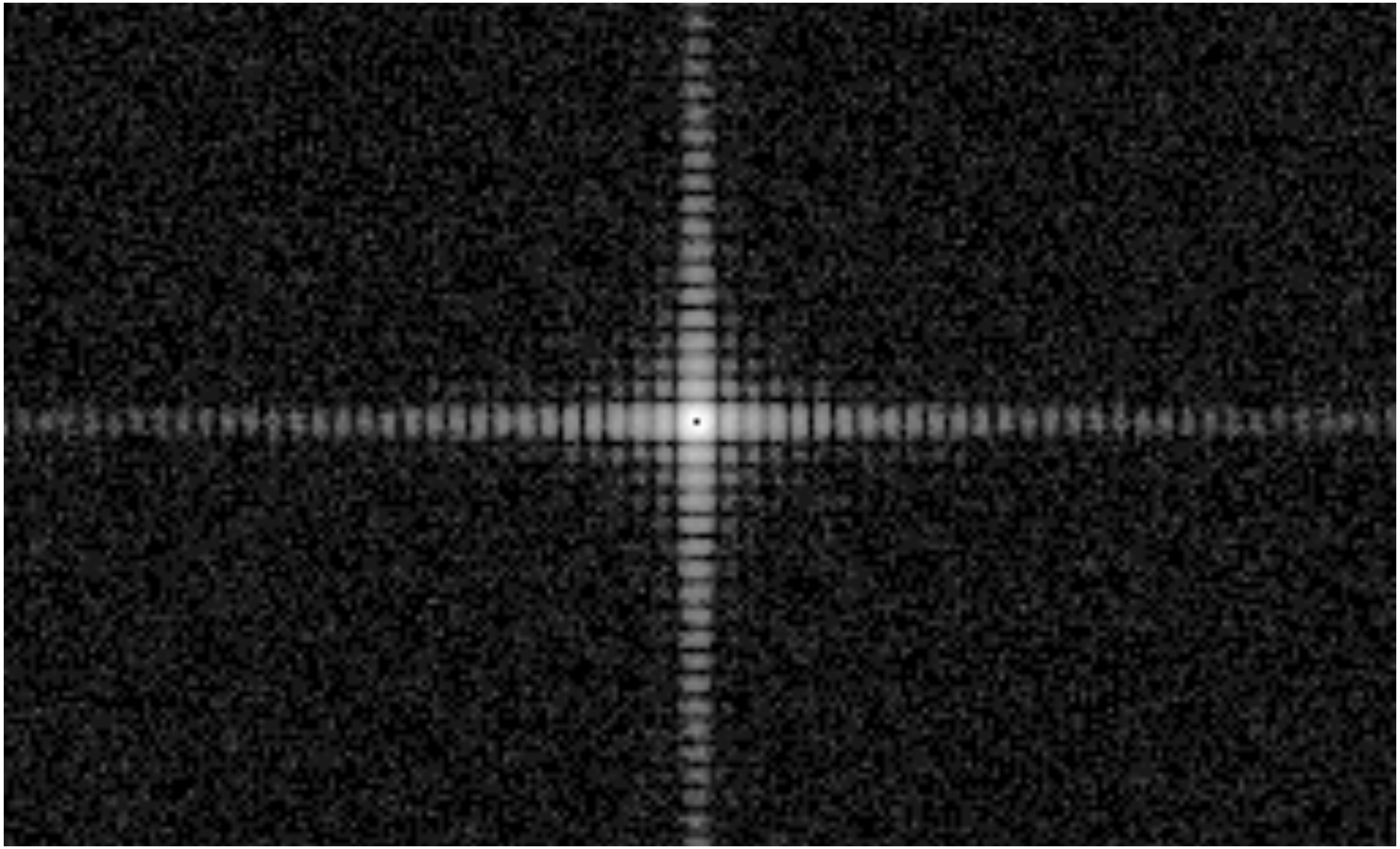
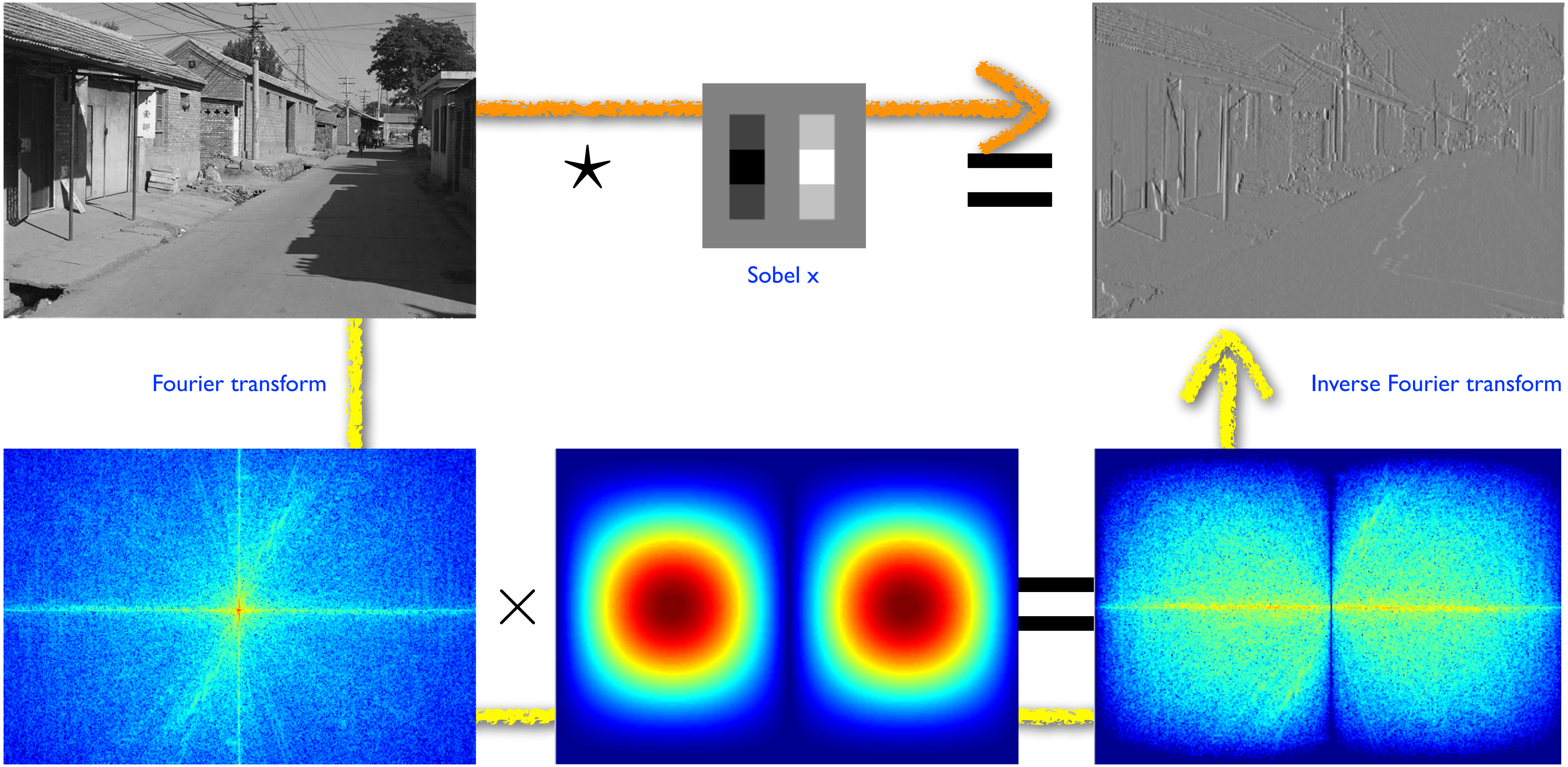


Frequency Domain Filtering

Spatial domain filtering



Sobel x

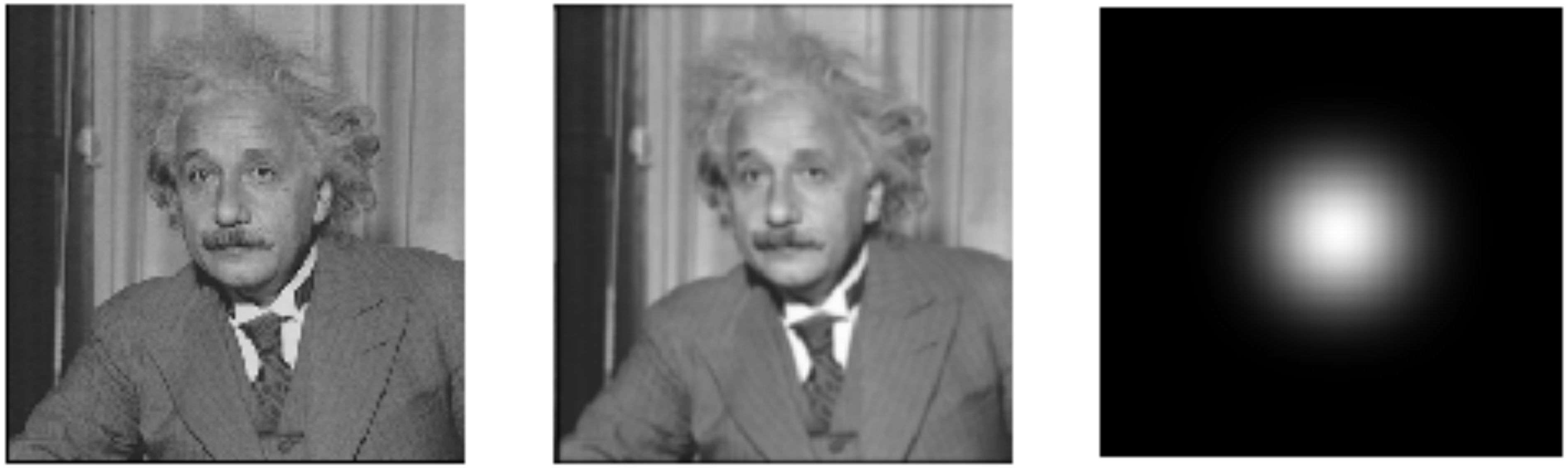
Fourier transform

Inverse Fourier transform

Frequency domain filtering

Frequency Domain Filtering

low-pass

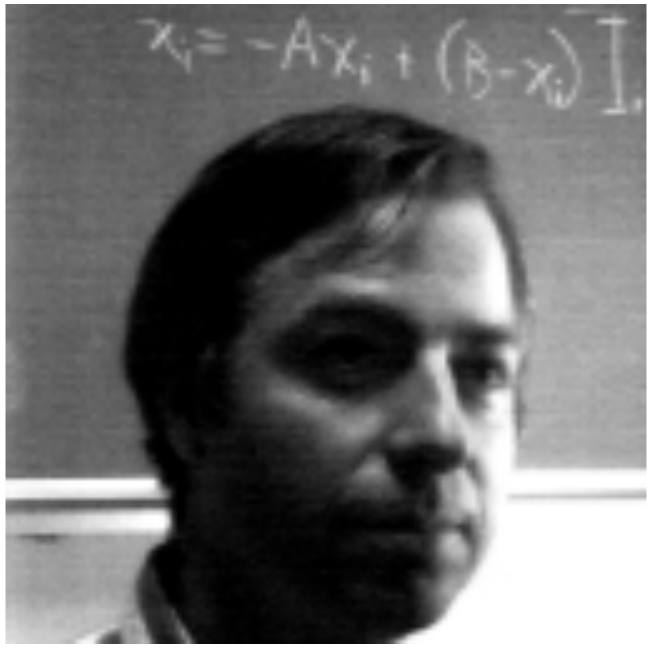


band-pass

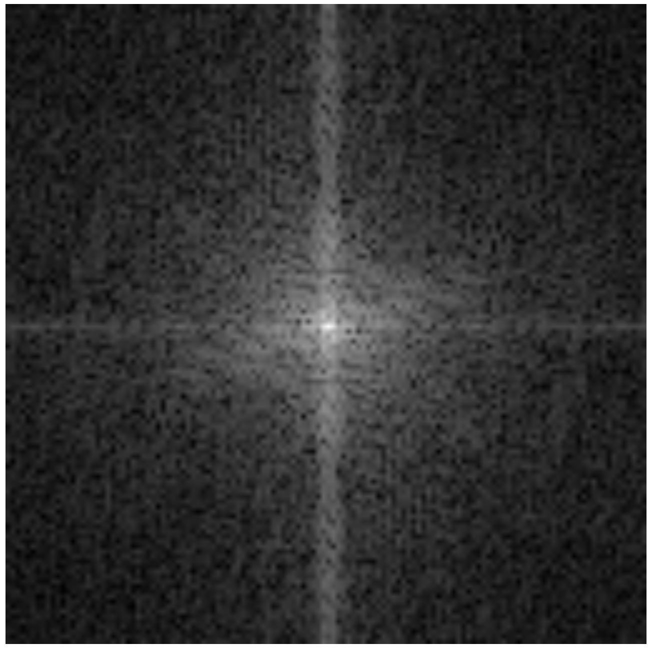
Frequency Domain Filtering

high-pass

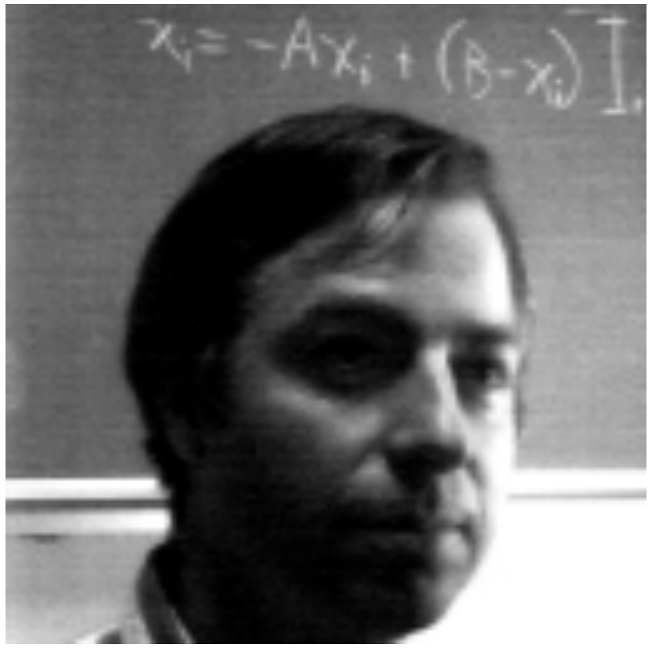
Original image



Frequency magnitude

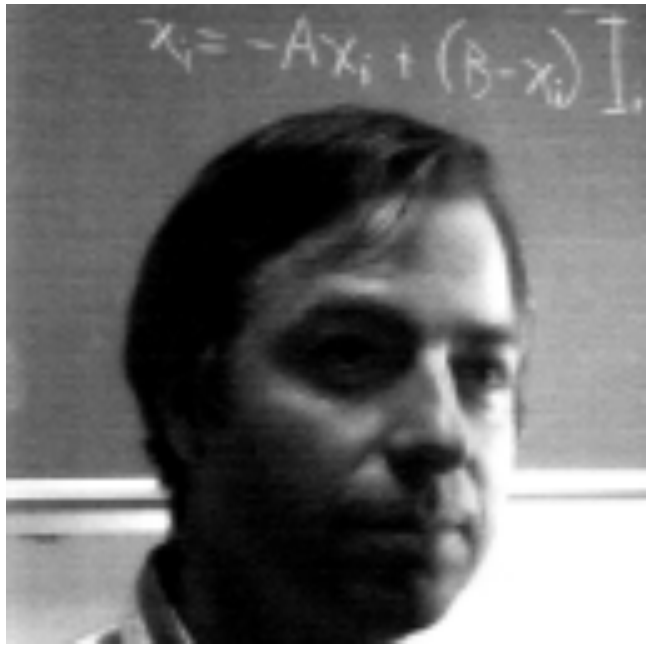


Inverse Fourier transform

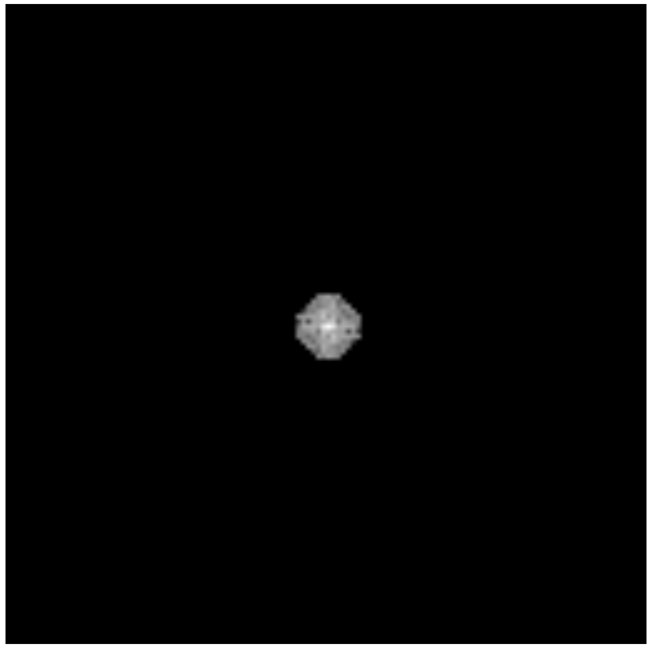


http://cns-alumni.bu.edu/~slehar/fourier/fourier.html

Original image



Low-pass filter

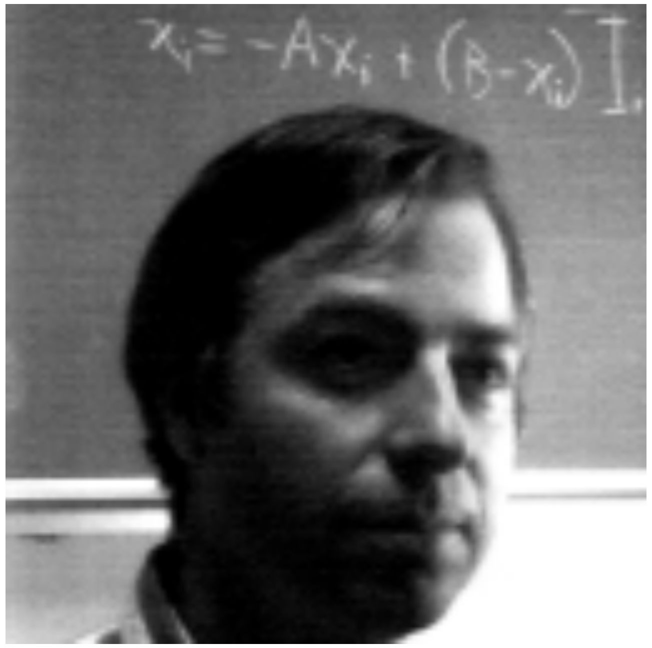


Inverse Fourier transform

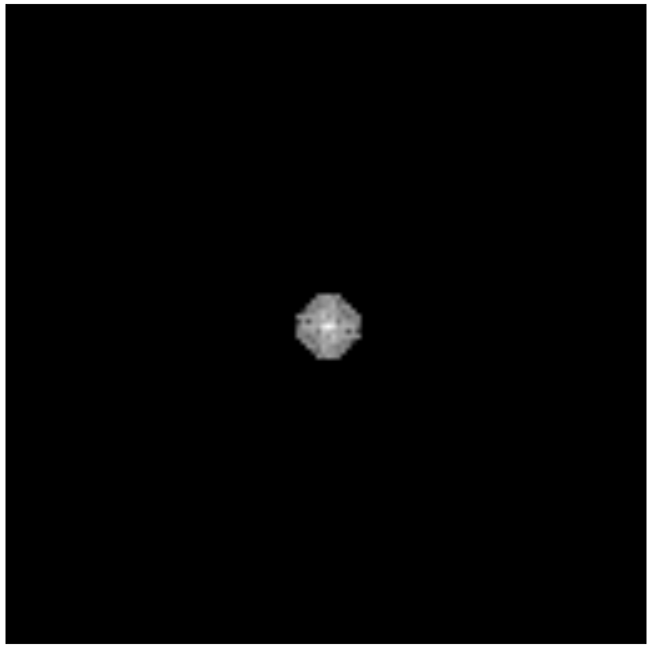
?

http://cns-alumni.bu.edu/~slehar/fourier/fourier.html

Original image



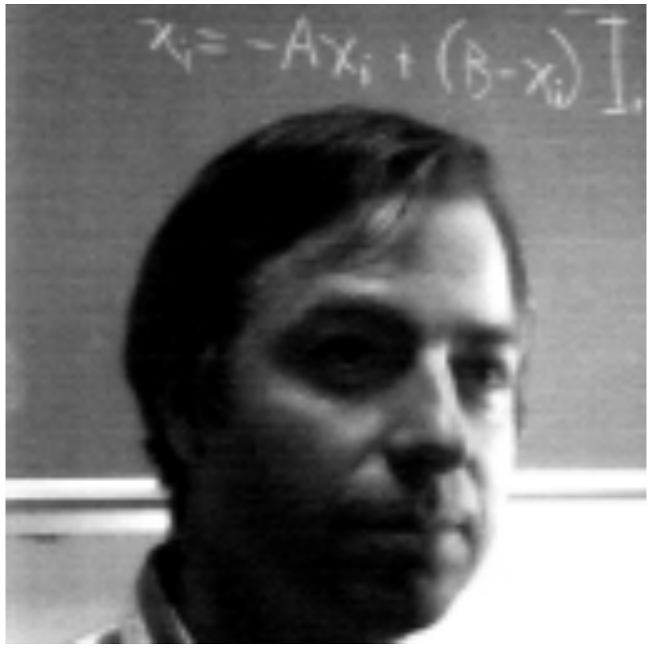
Low-pass filter



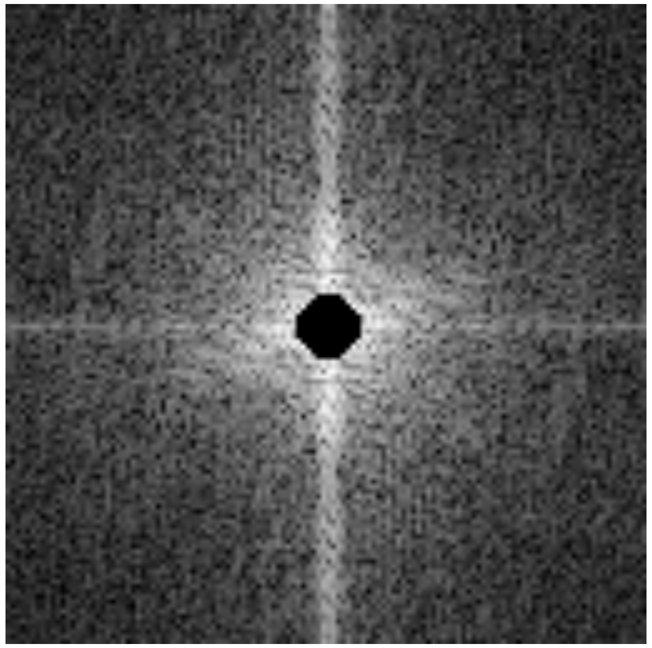
Inverse Fourier transform

http://cns-alumni.bu.edu/~slehar/fourier/fourier.html

Original image



High-pass filter

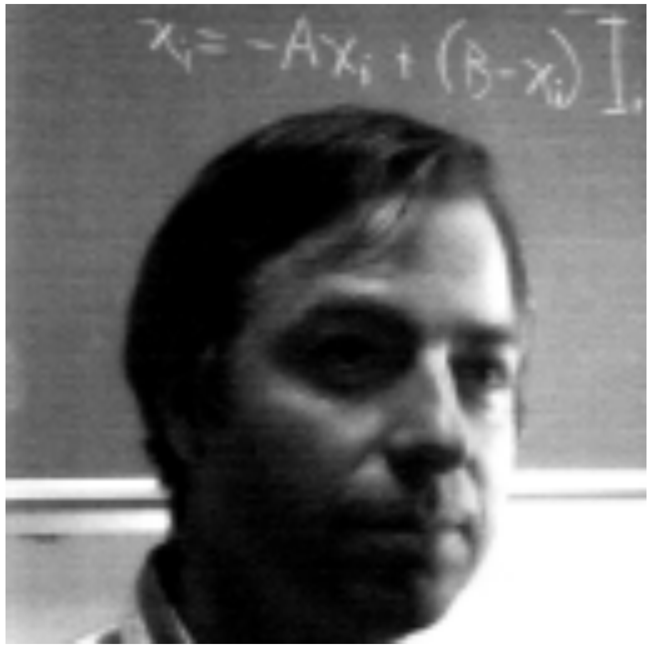


Inverse Fourier transform

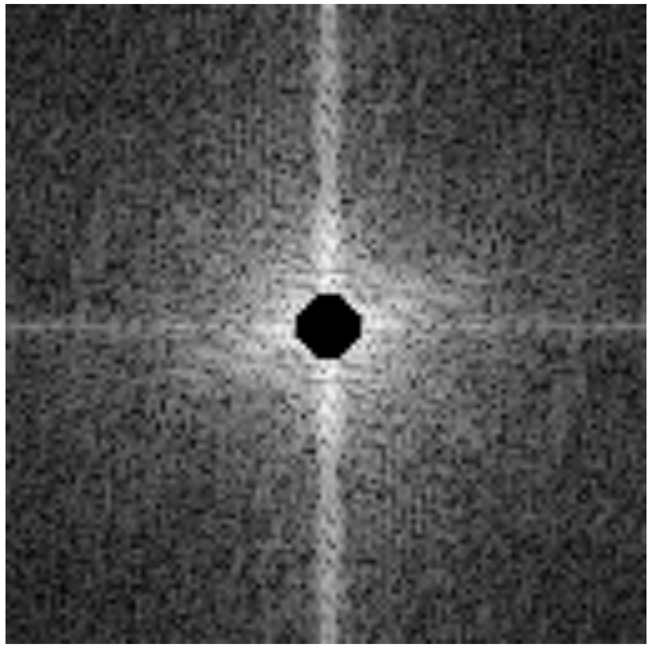
?

http://cns-alumni.bu.edu/~slehar/fourier/fourier.html

Original image

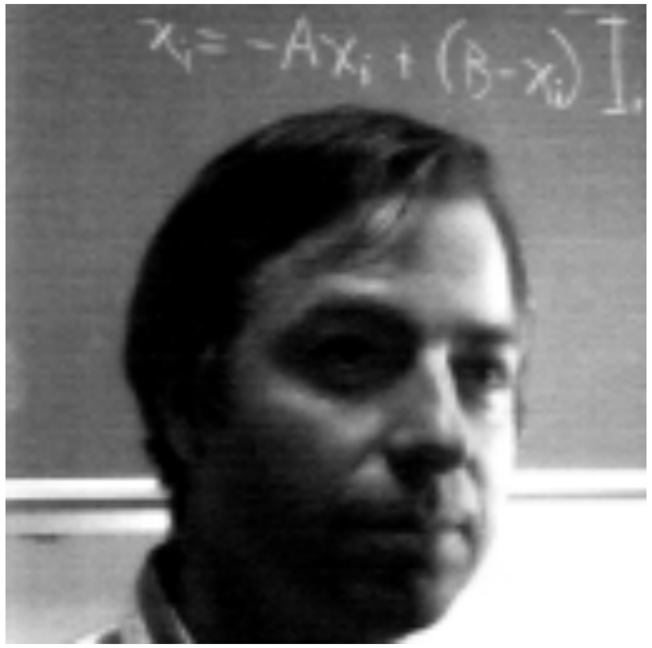


High-pass filter

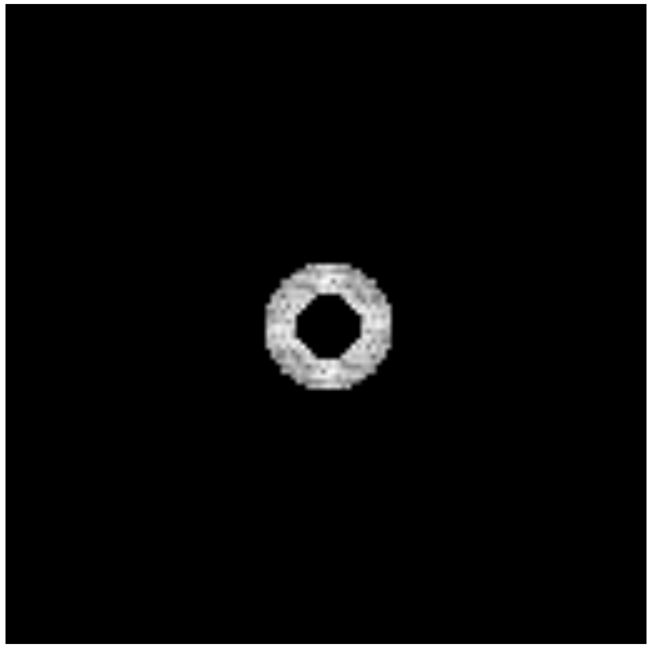


Inverse Fourier transform

http://cns-alumni.bu.edu/~slehar/fourier/fourier.html



Original image

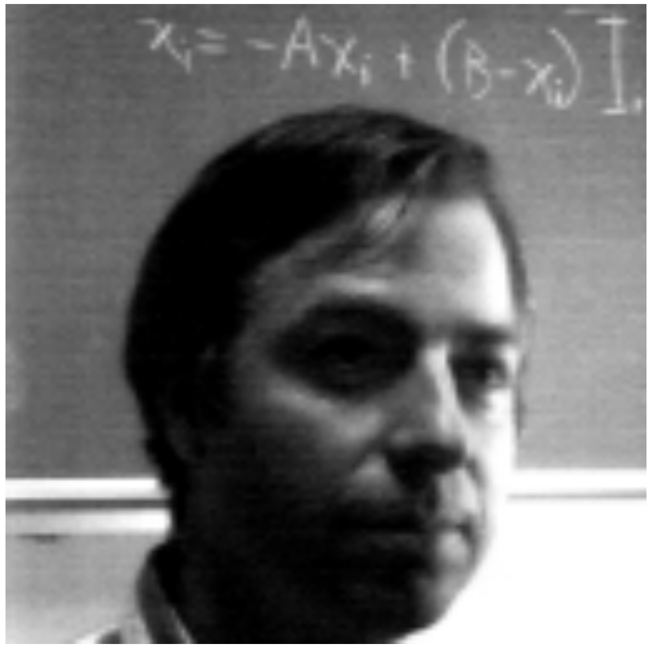


Band-pass filter

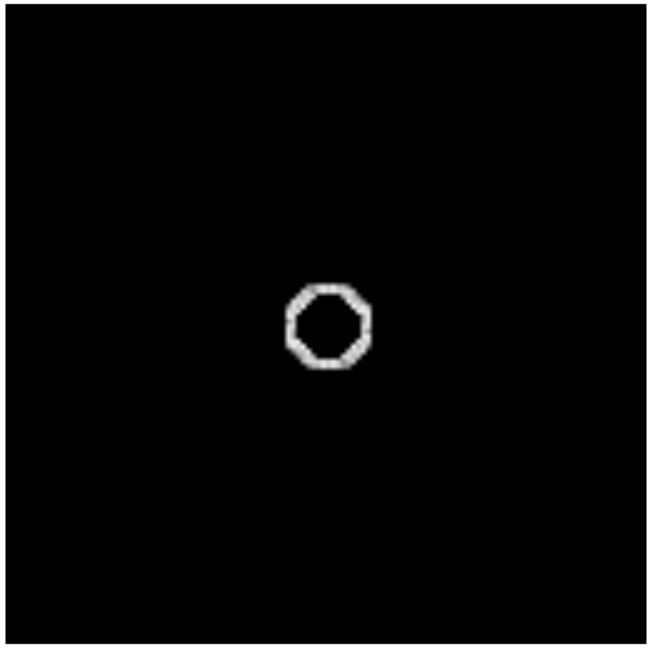
Inverse Fourier transform

http://cns-alumni.bu.edu/~slehar/fourier/fourier.html

Original image



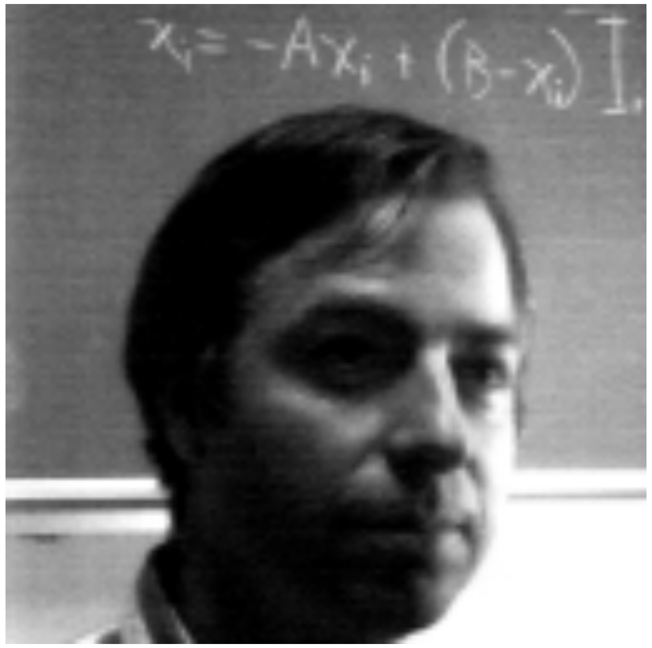
Band-pass filter



Inverse Fourier transform

http://cns-alumni.bu.edu/~slehar/fourier/fourier.html

Original image

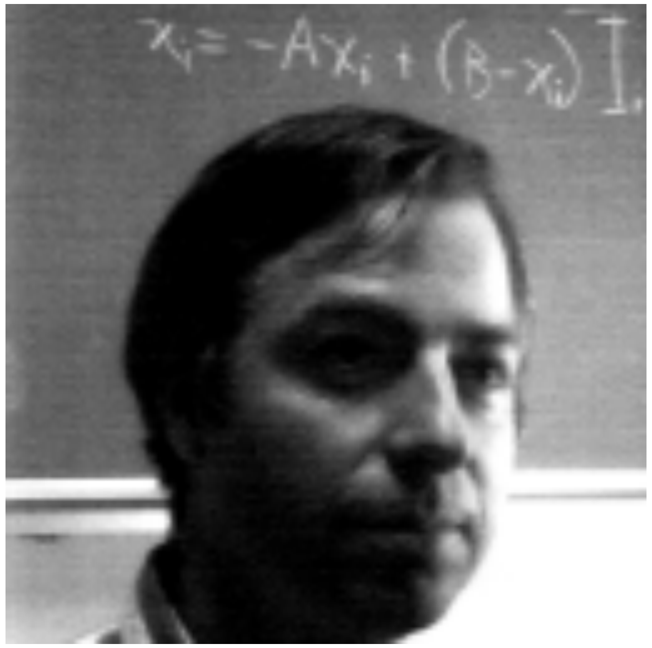


Band-pass filter

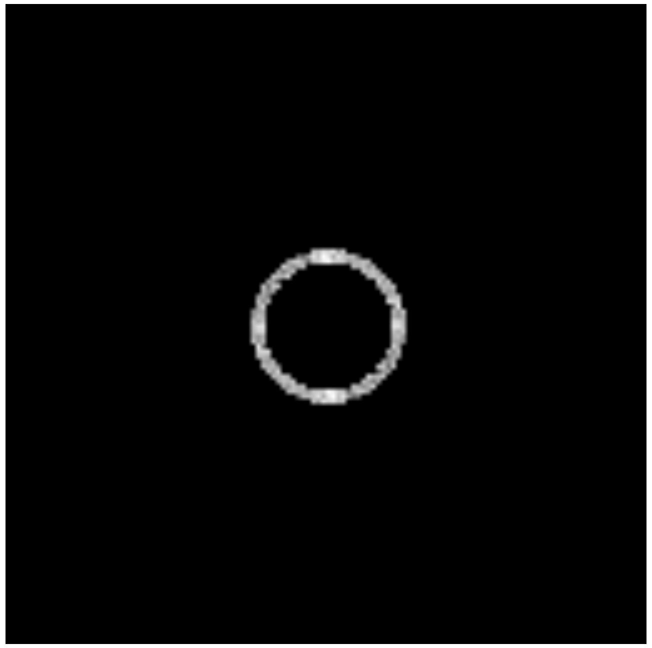
Inverse Fourier transform

http://cns-alumni.bu.edu/~slehar/fourier/fourier.html

Original image



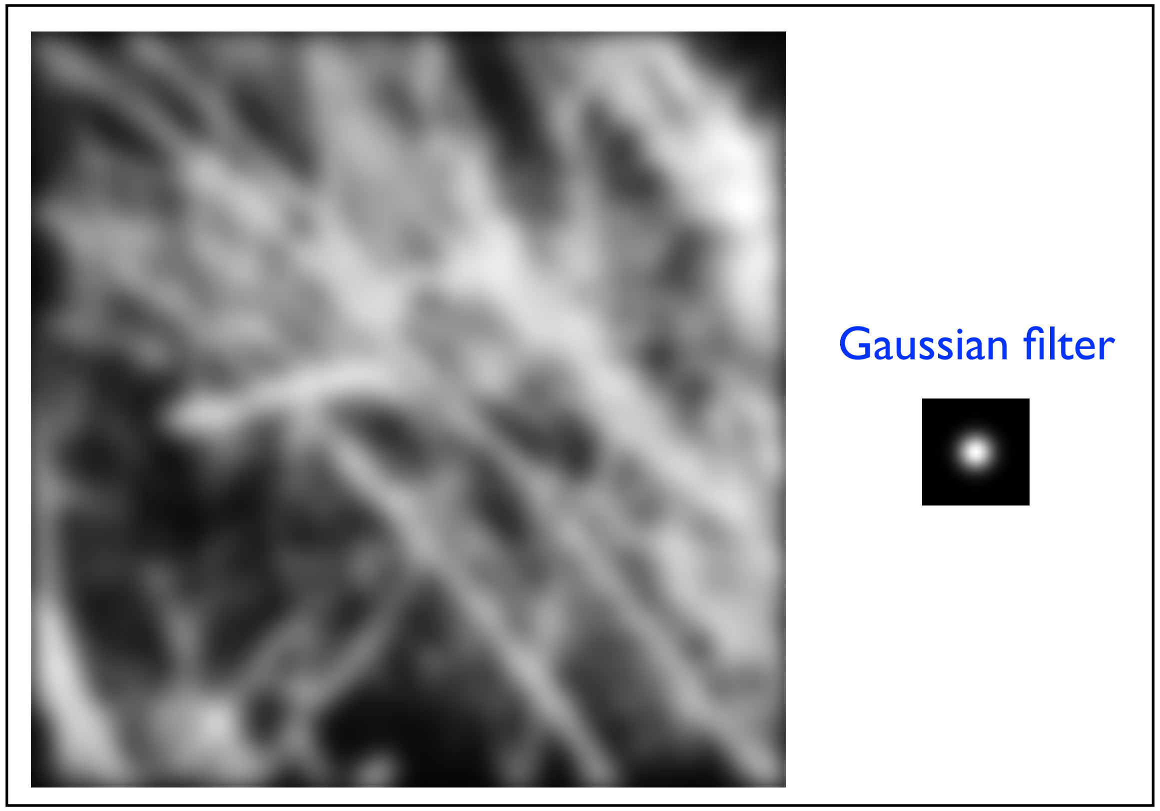
Band-pass filter

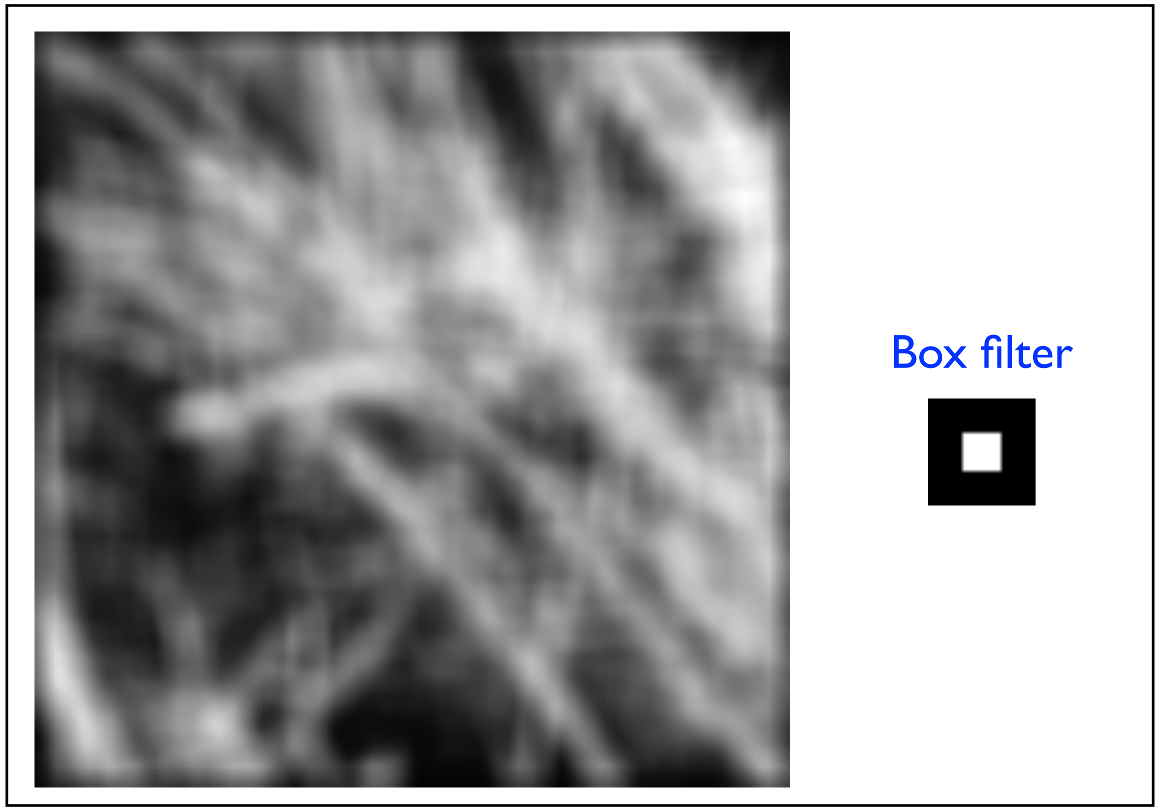


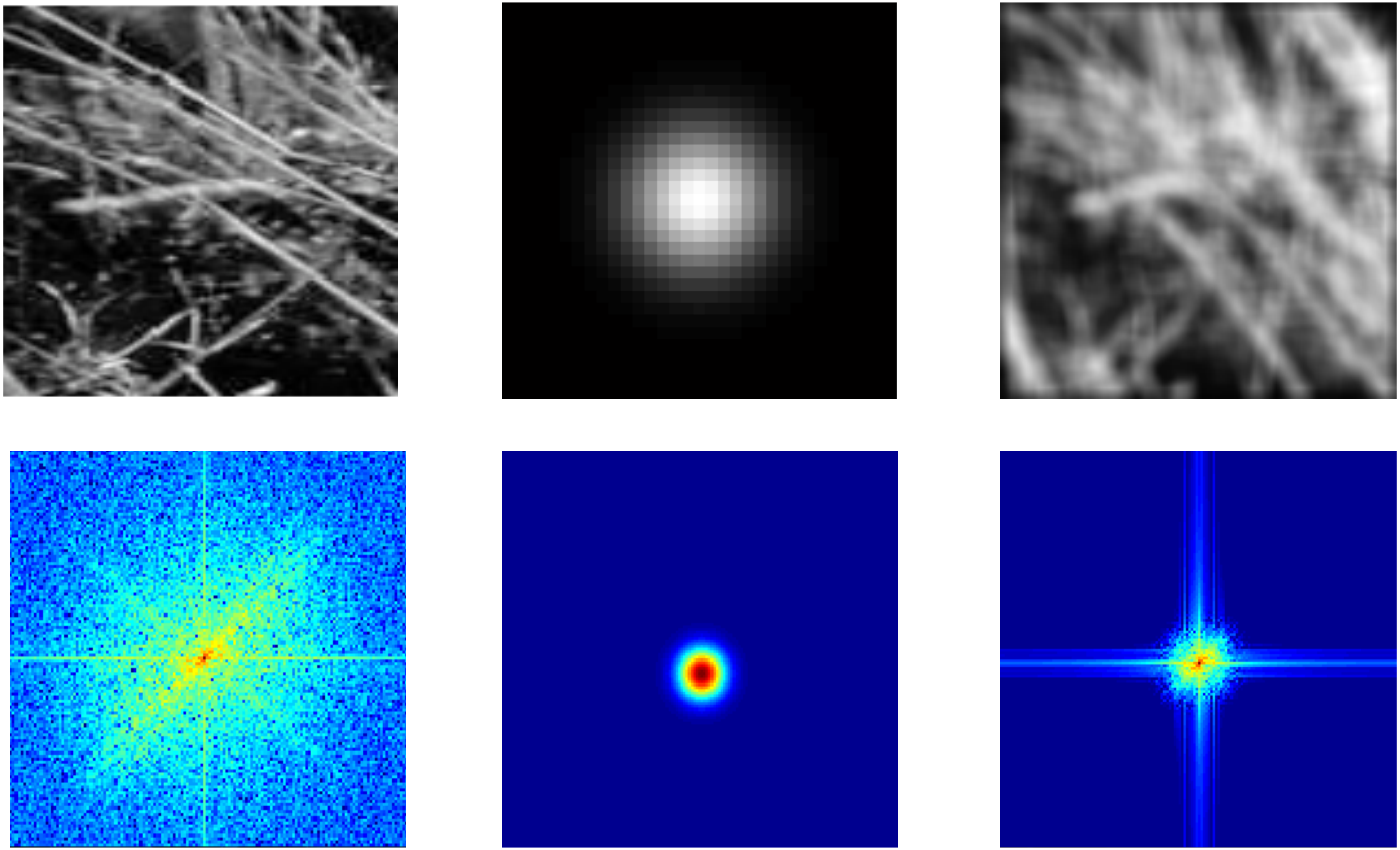
Inverse Fourier transform

http://cns-alumni.bu.edu/~slehar/fourier/fourier.html

Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?



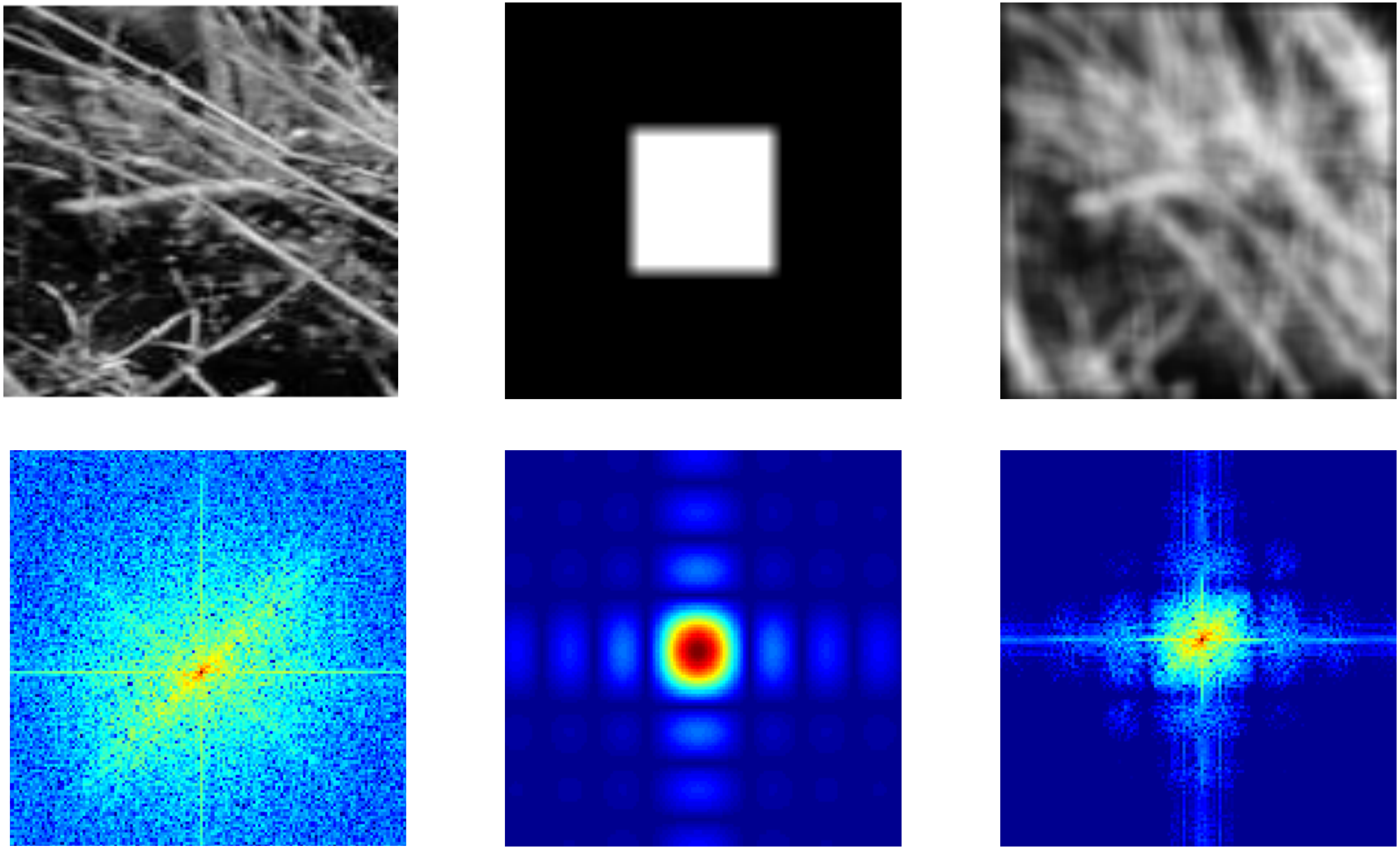




original

filter

result



original

filter

result

Match the image to the Fourier magnitude image:

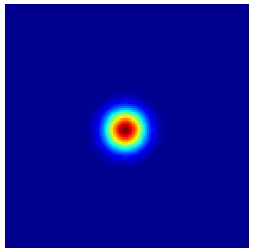
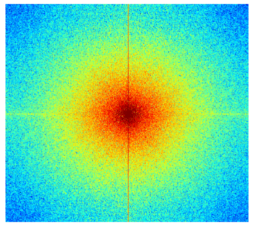
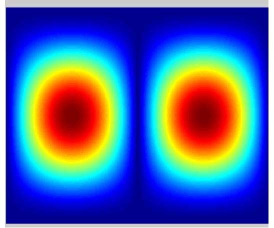
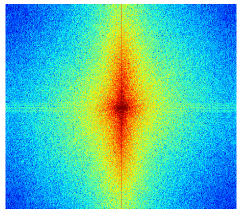
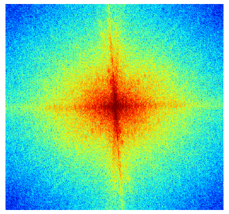
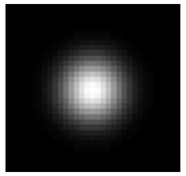
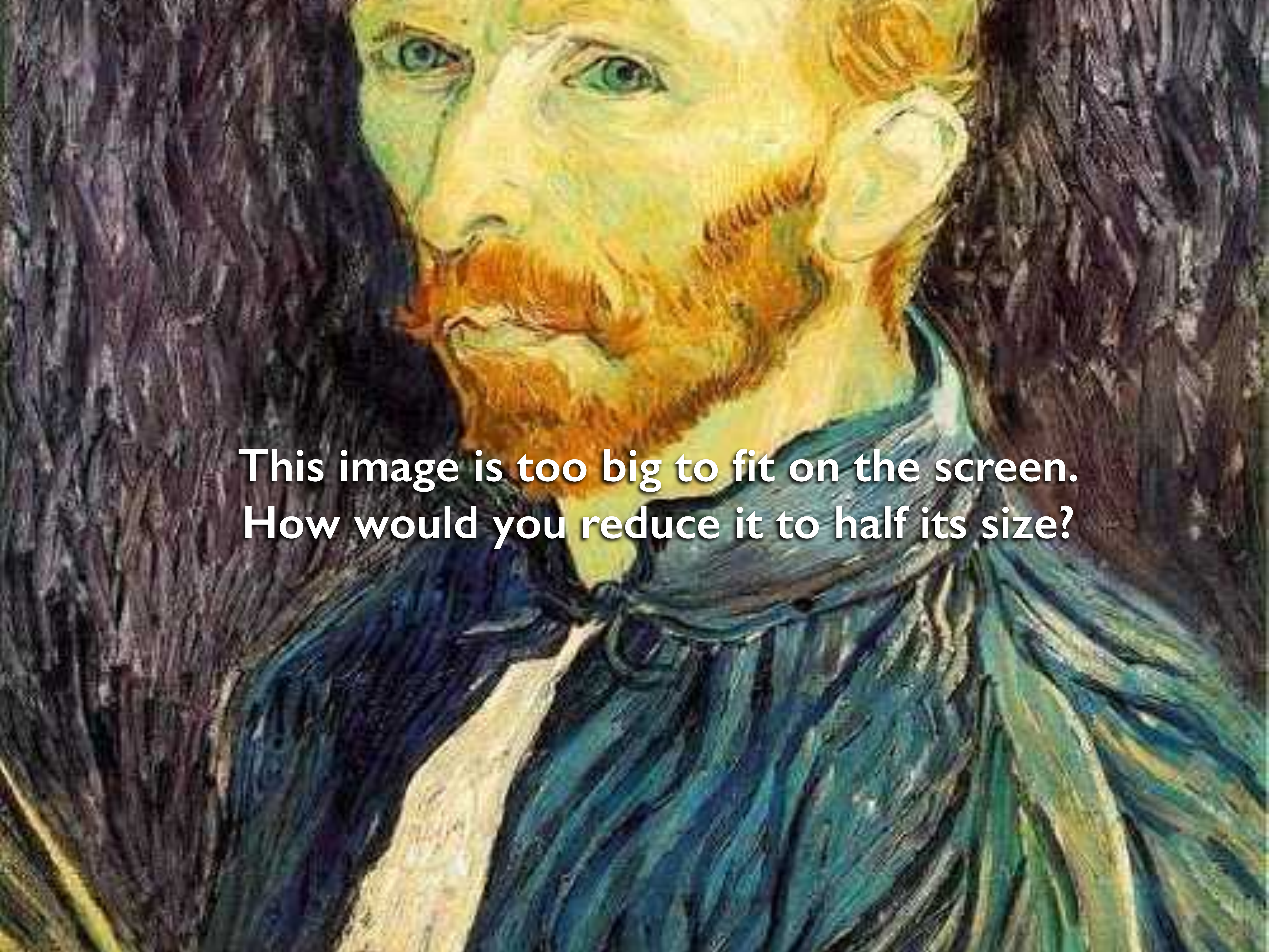
1 	2 	3 	4 	5 
a 	b 	c 	d 	e 

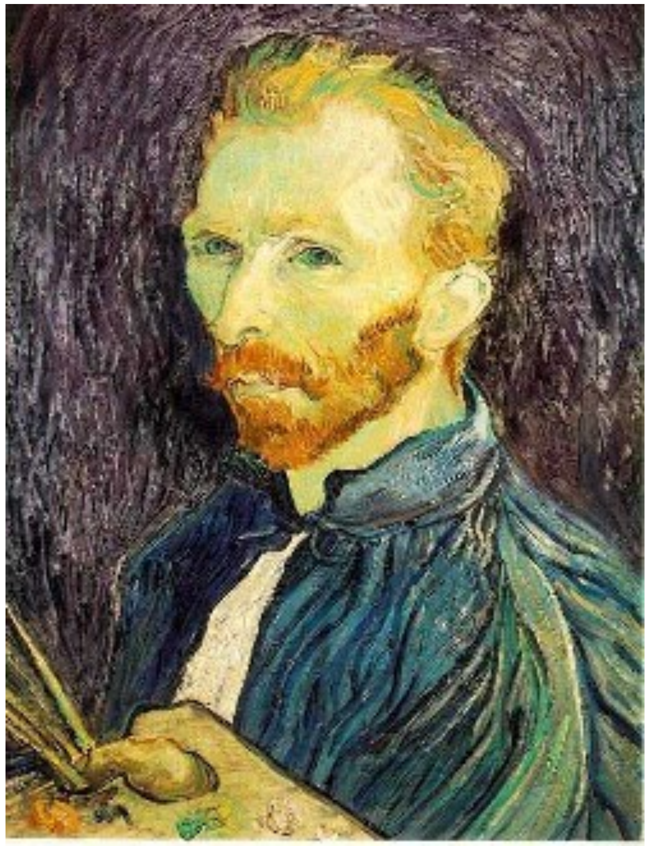
Image Subsampling



This image is too big to fit on the screen.
How would you reduce it to half its size?

Naive image sub-sampling

'throw away even rows and columns'



1/2

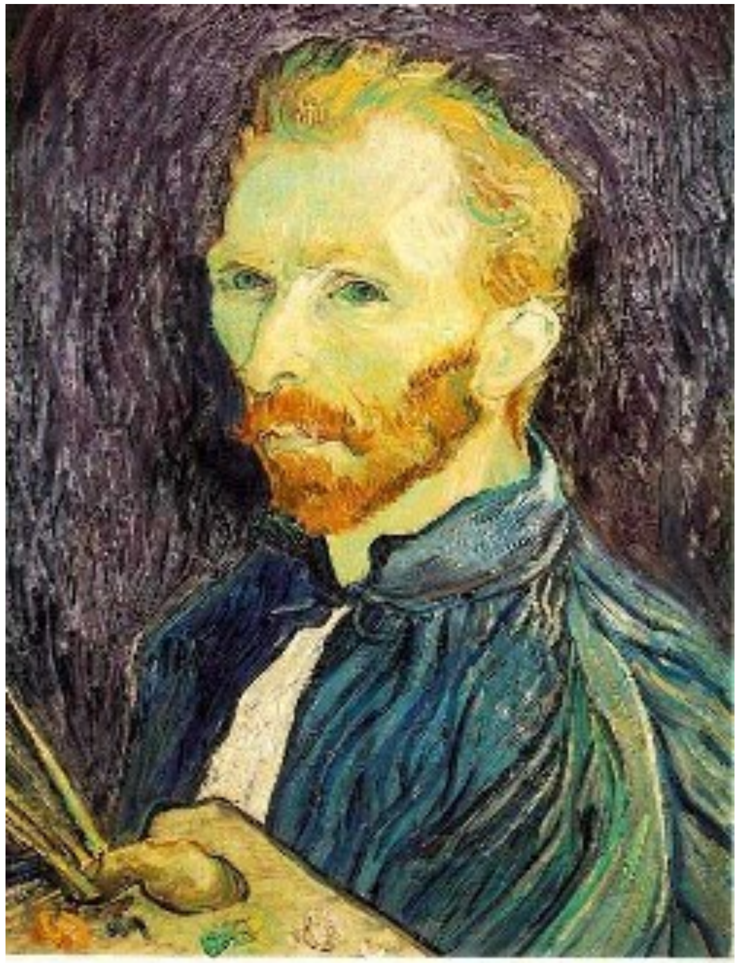
delete even rows
delete even columns

1/4

delete even rows
delete even columns

1/8

What are the problems with this approach?



1/2

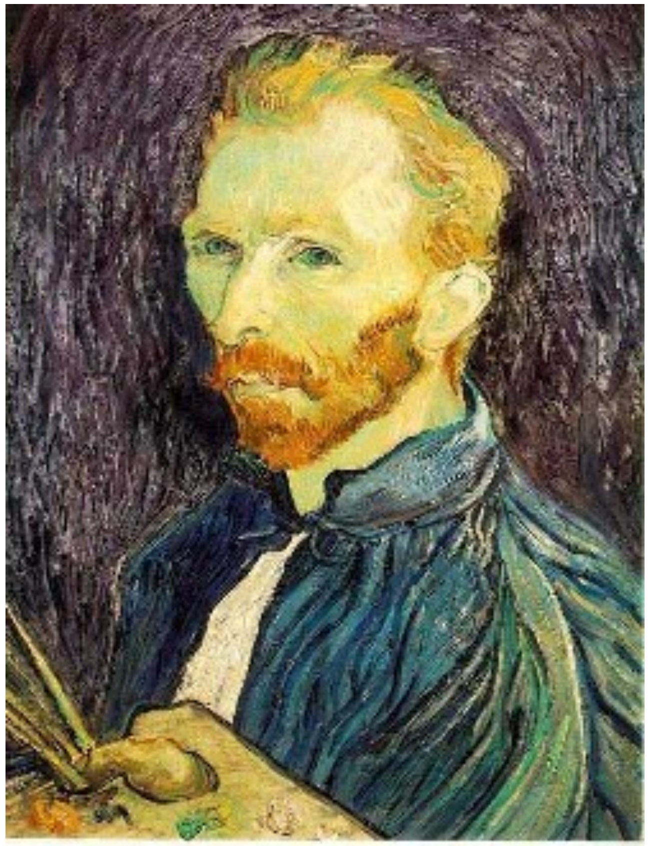
1/4 scaled by 2

1/4 scaled by 4

Why is the 1/4 image so blocky (pixelated, aliased)?

How can we fix this?

Add Gaussian (lowpass) pre-filtering



1/2

Gaussian filtering
delete even rows
delete even columns

1/4

Gaussian filtering
delete even rows
delete even columns

1/8

What will the images look like scale to the same size?

Gaussian pre-filtering

Naive subsampling

This sequence of subsampled images is called the...

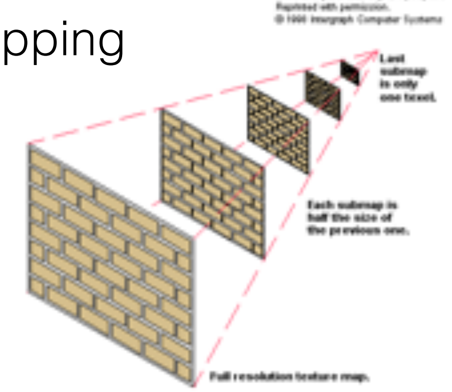
Gaussian image pyramid

Image Pyramids

What are image pyramids used for?

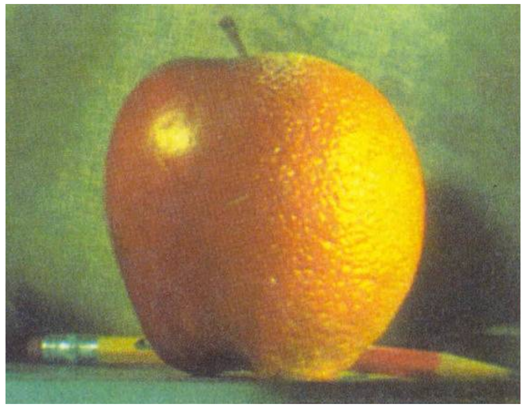
Image compression

Multi-scale
texture mapping

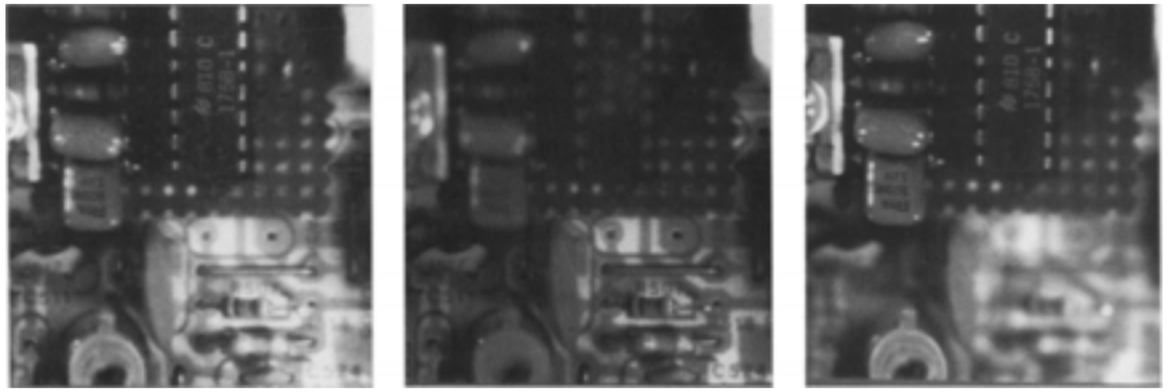


From Computer Desktop Encyclopedia
Reprinted with permission.
© 1998 Intergraph Computer Systems

Image blending

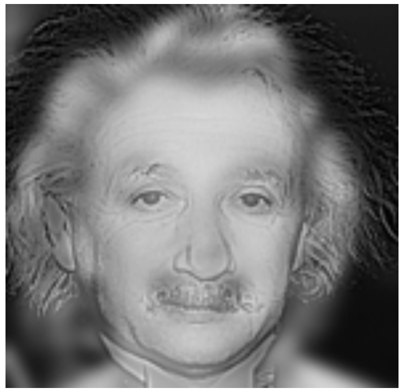


Multi-focus composites



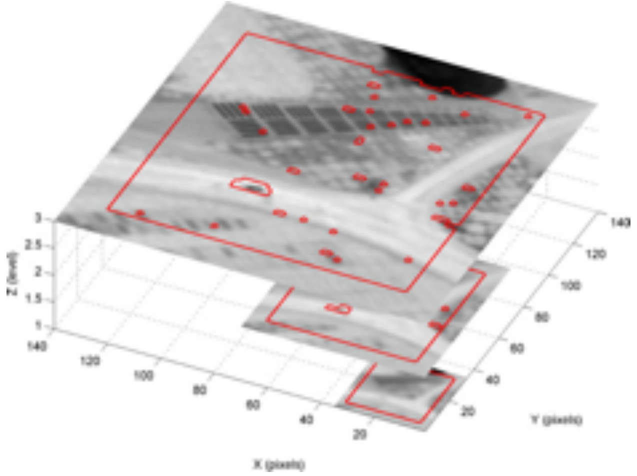
Noise removal

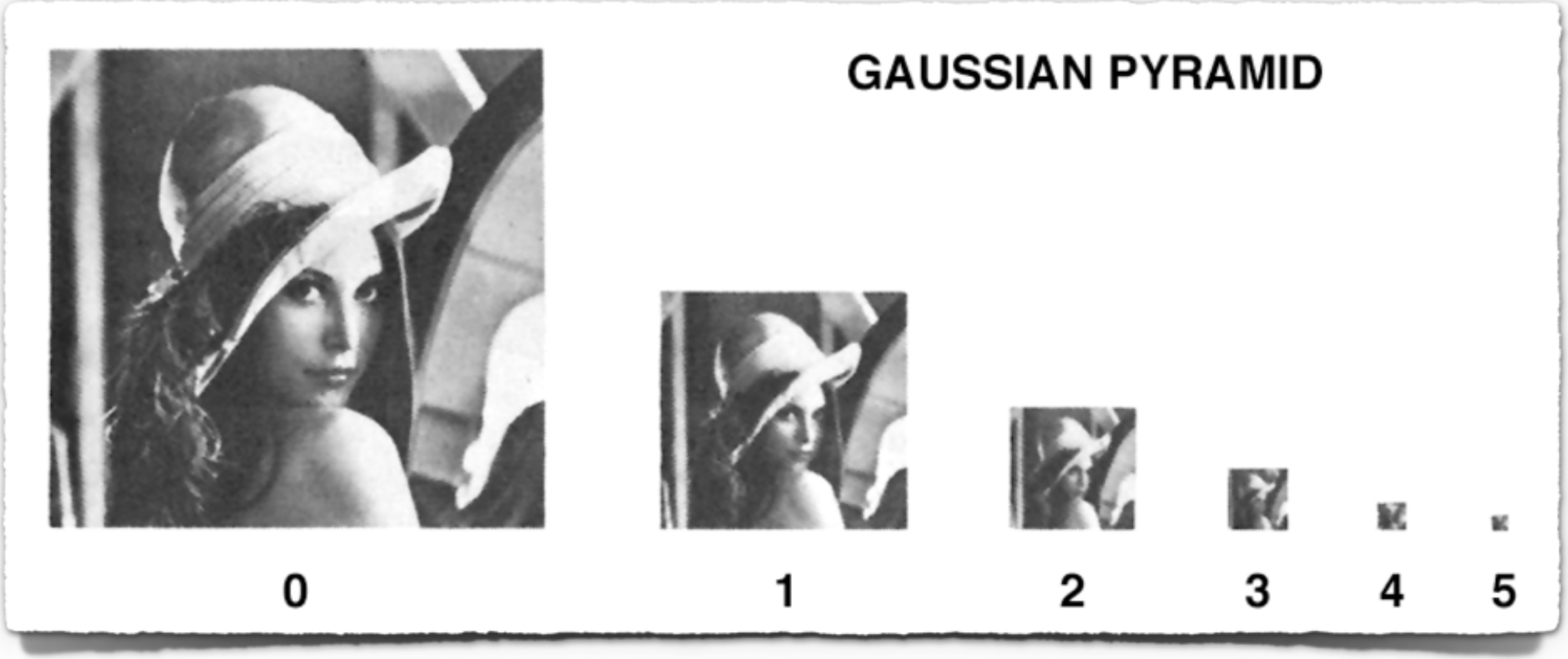
Hybrid images



Multi-scale detection

Multi-scale registration



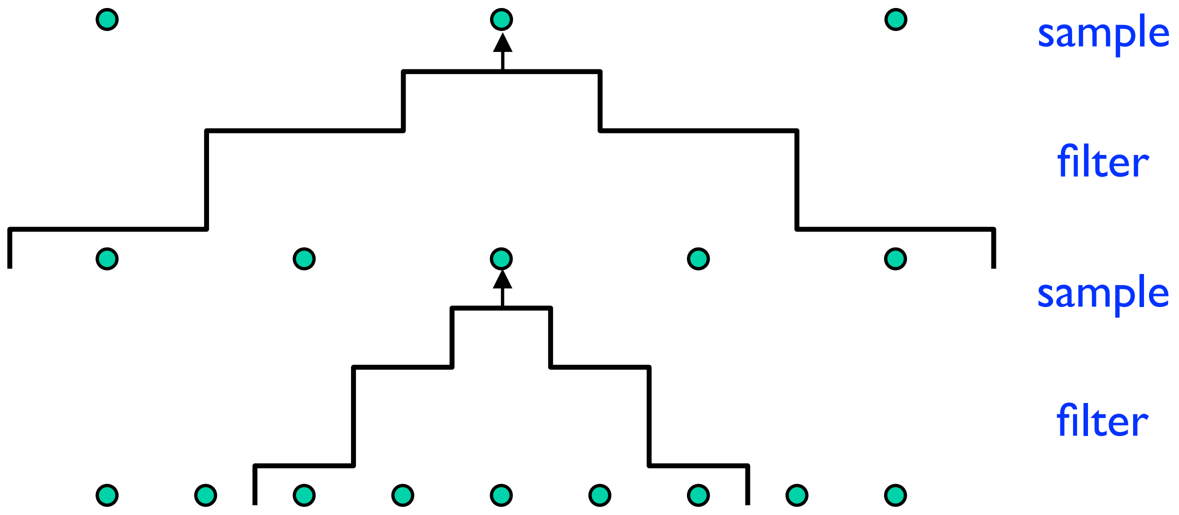


The Laplacian Pyramid as a Compact Image Code (1983)

Peter J. Burt and Edward H. Adelson

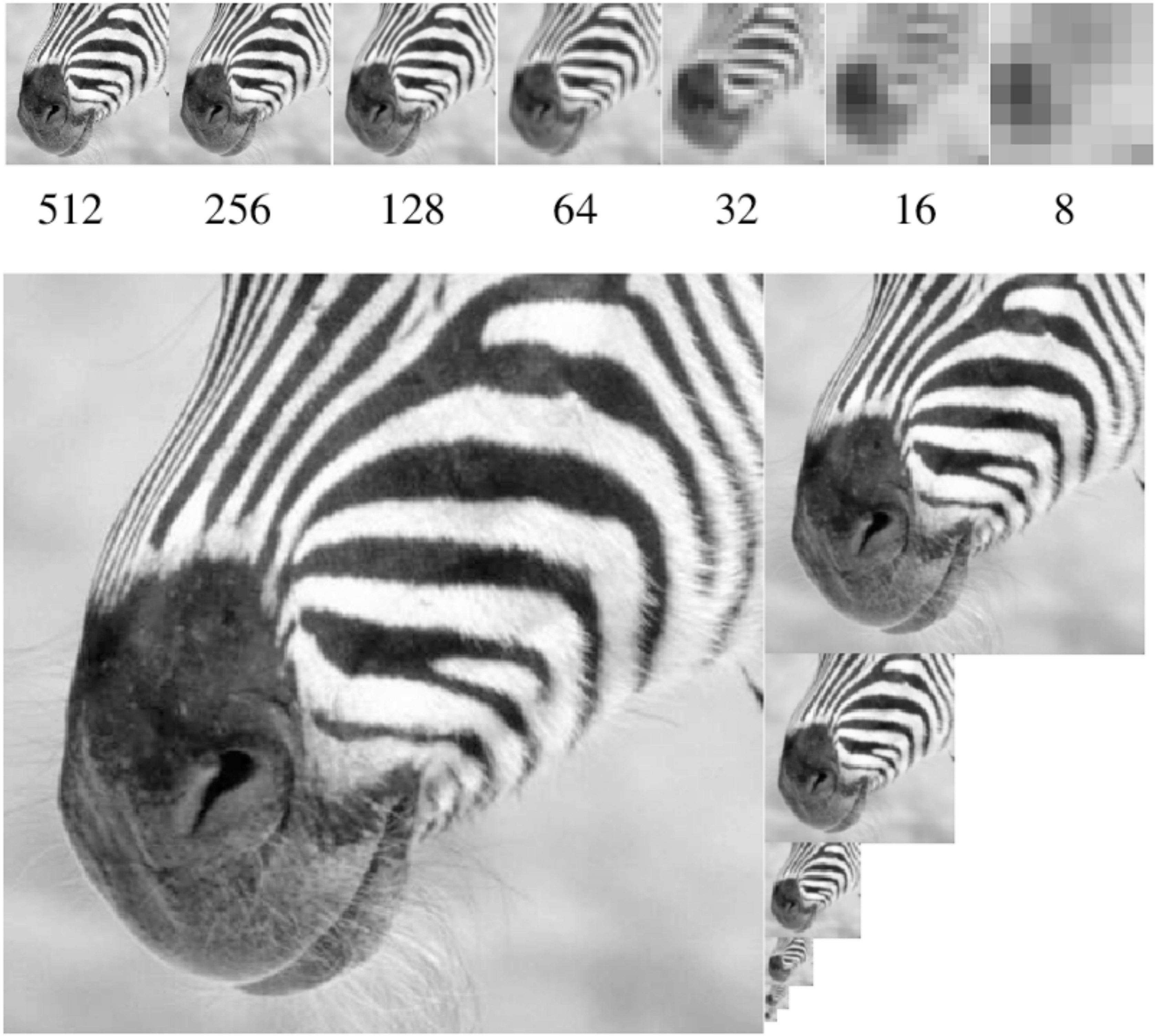
Constructing a Gaussian Pyramid

```
repeat
    filter
    subsample
until min resolution reached
```



Whole pyramid is only 4/3 the size of the original image!

Gaussian pyramid

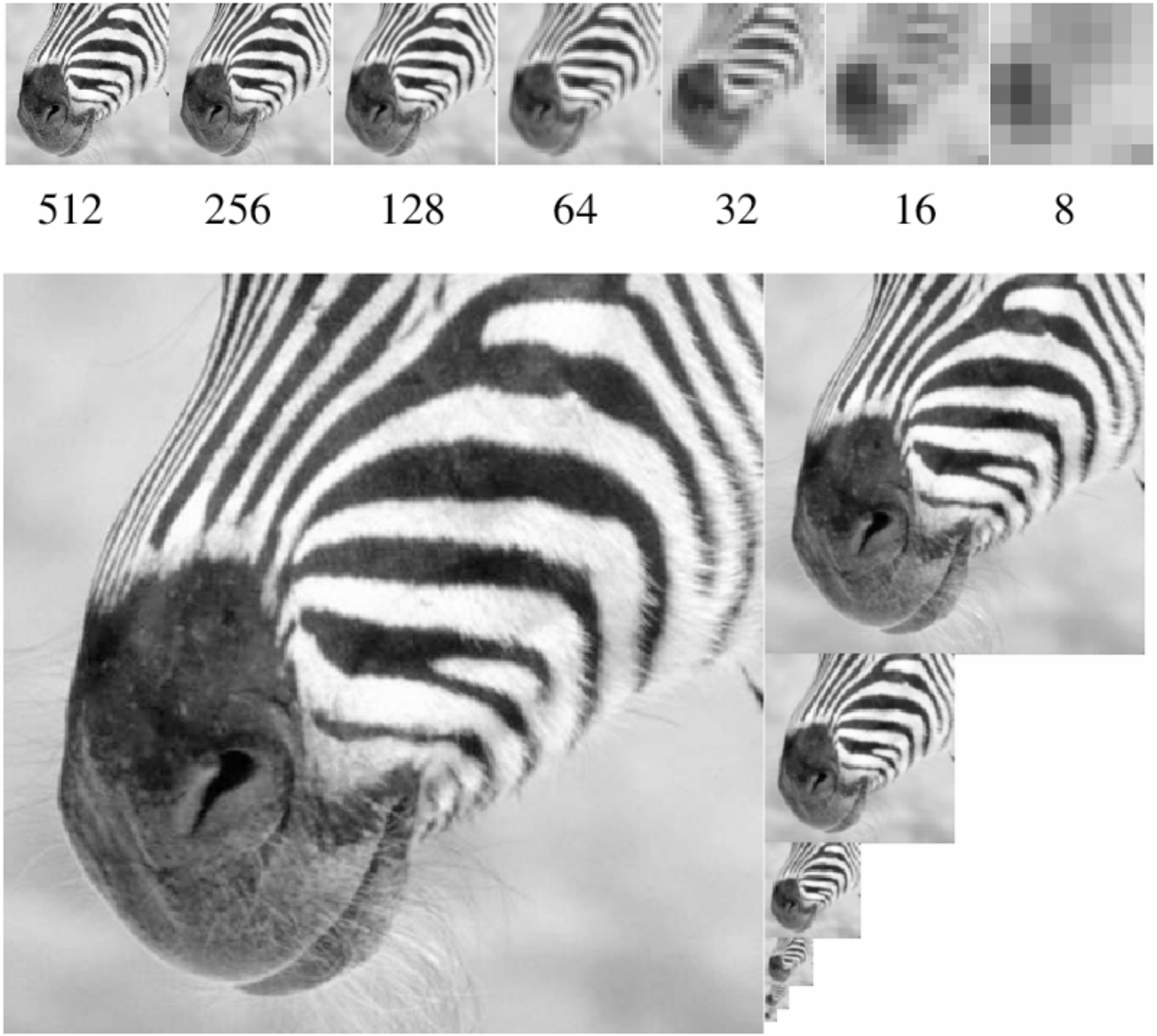


What happens to the details of the image?

What is preserved at the higher scales?

How would you reconstruct the original image using the upper pyramid?

Gaussian pyramid



What happens to the details of the image?

What is preserved at the higher scales?

Not possible

Level 0

Level 1

What is lost between levels?

What does blurring take away?

Level 0

-

Level 1

=

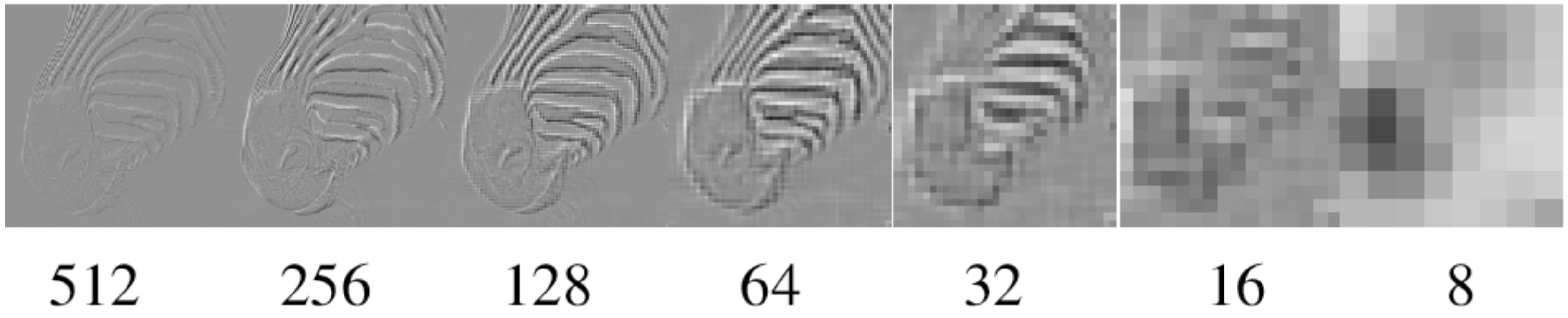
Residual

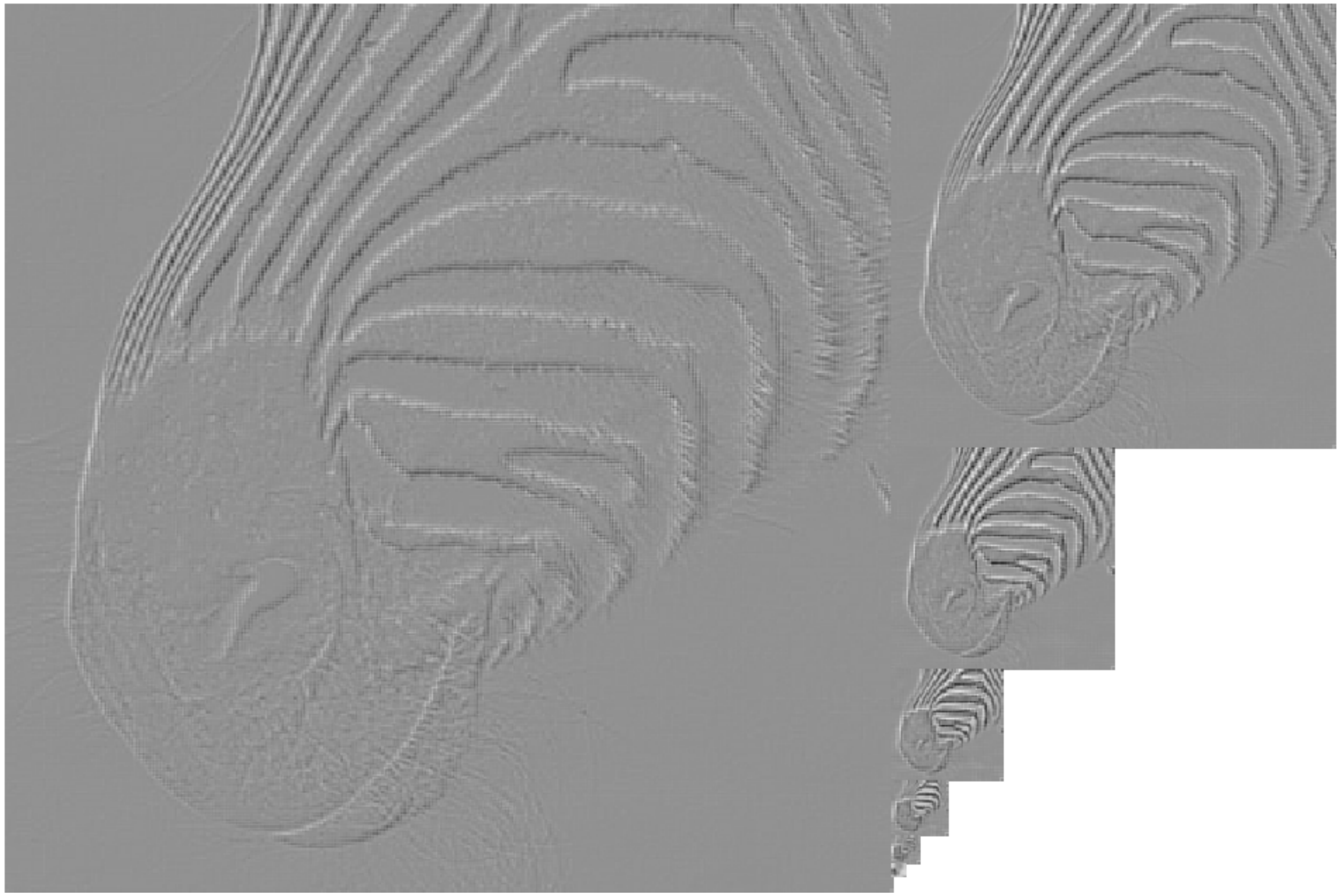
(thrown away by blurring)

(band-pass filter)

We can retain the residuals with a ...

Laplacian pyramid





Retains the residuals (details) between pyramid levels

Can you reconstruct the original image using the upper pyramid?

What exactly do you need to reconstruct the original image?

Partial answer:

Level 0

=

Level 1
(resized)

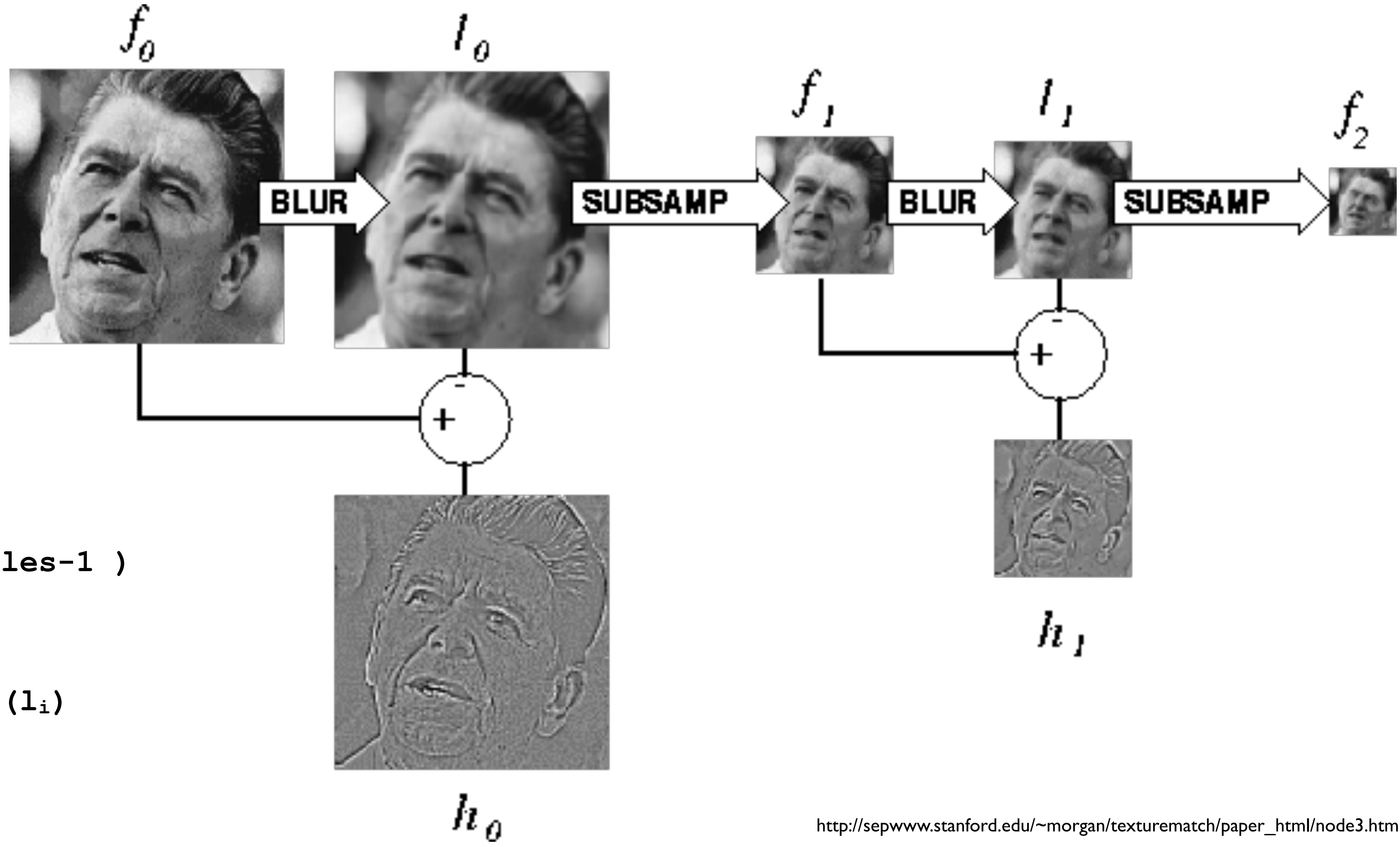
+

Level 0

Low frequency component

High frequency component

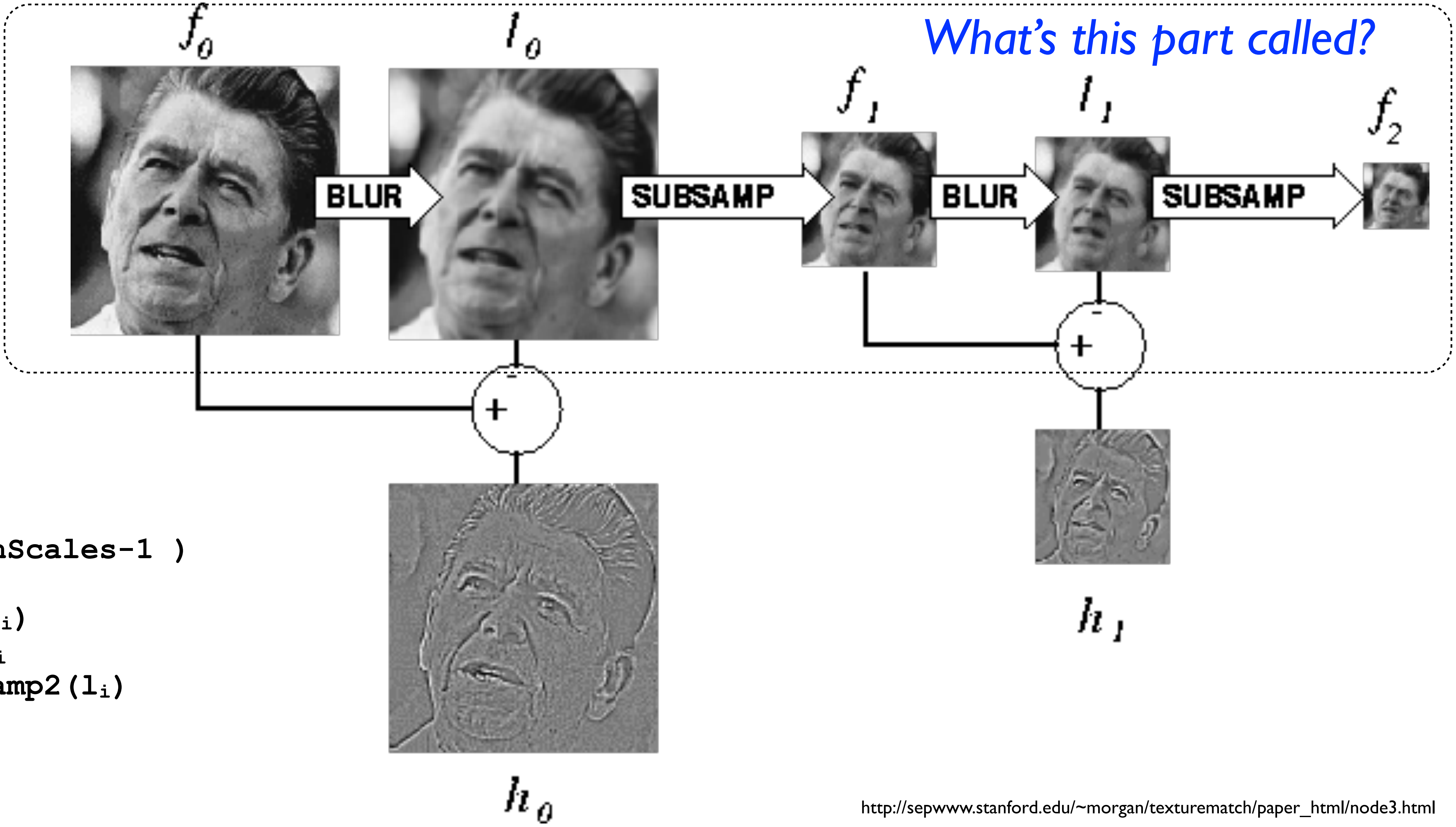
Constructing the Laplacian Pyramid



```
do( i = 0 : nScales-1 )
{
  $l_i$ = blur($f_i$)
  $h_i$ = $l_i$ - $f_i$
  $f_{i+1}$ = subSamp2($l_i$)
}
```

http://sepwww.stanford.edu/~morgan/texturematch/paper_html/node3.html

Constructing the Laplacian Pyramid

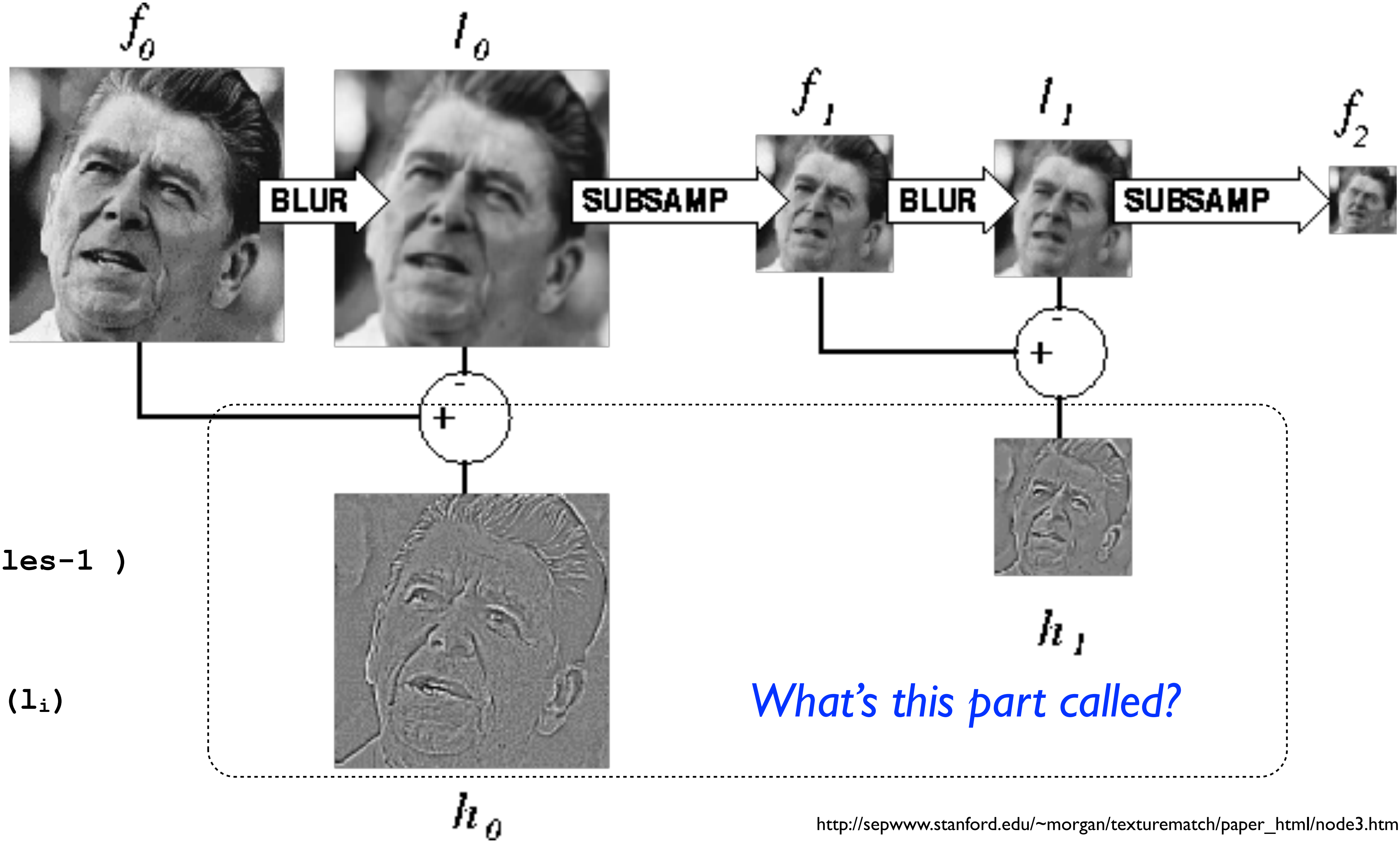


What's this part called?

```
do( i = 0 : nScales-1 )
{
  $l_i$ = blur($f_i$)
  $h_i$ = $l_i$ - $f_i$
  $f_{i+1}$ = subSamp2($l_i$)
}
```

http://sepwww.stanford.edu/~morgan/texturematch/paper_html/node3.html

Constructing the Laplacian Pyramid

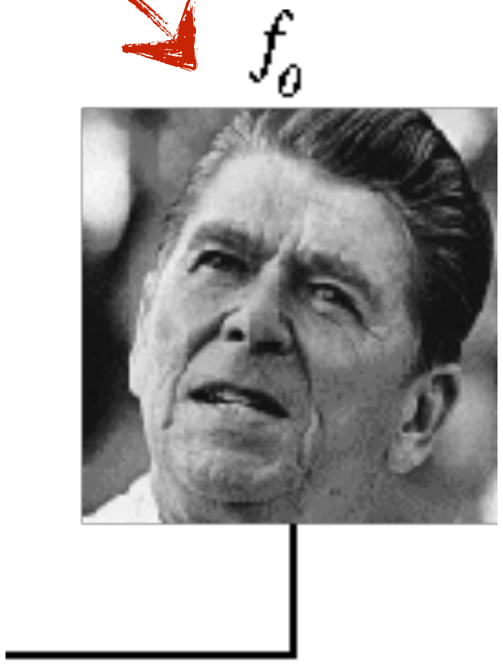


What's this part called?

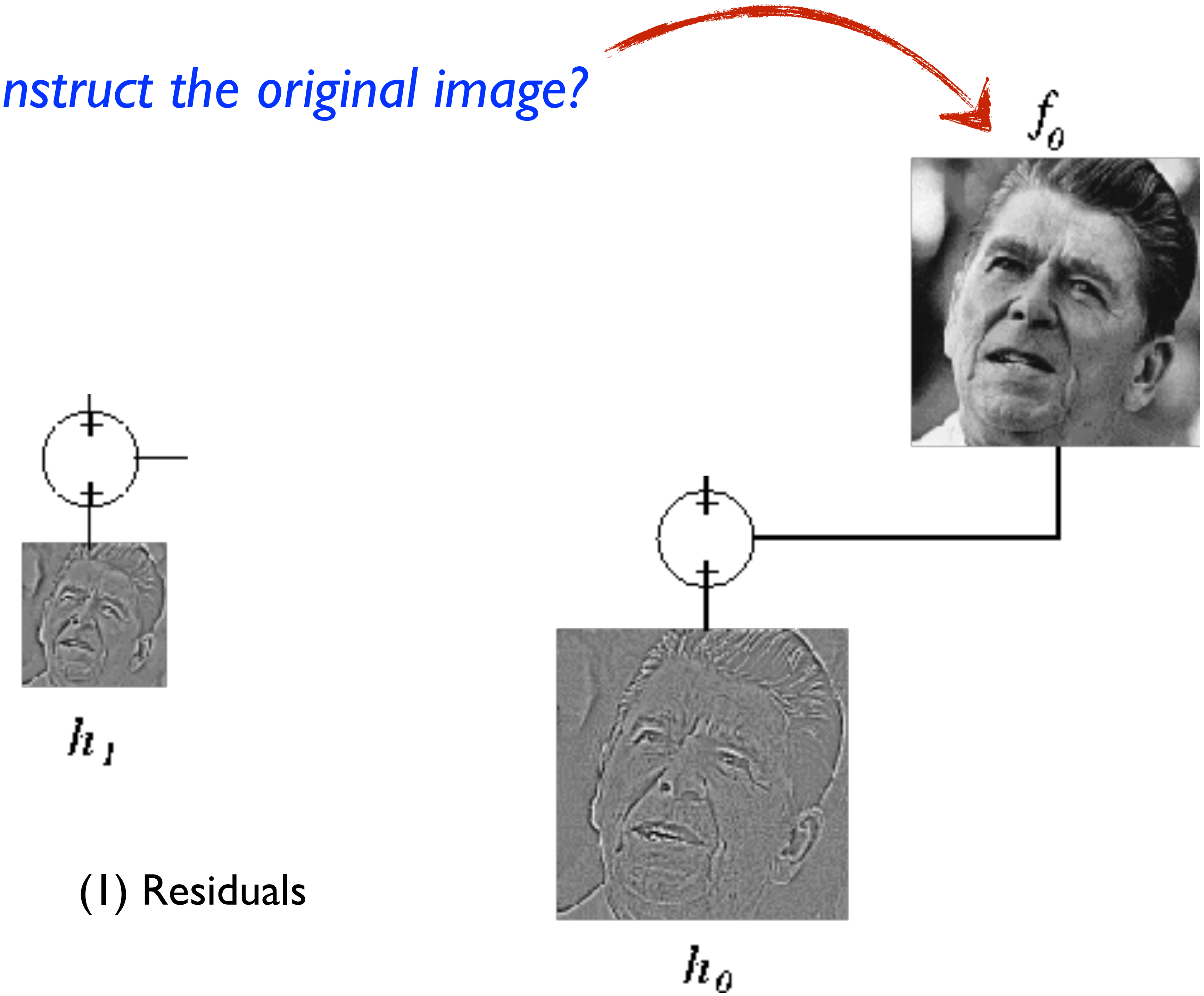
```
do( i = 0 : nScales-1 )
{
  $l_i$ = blur($f_i$)
  $h_i$ = $l_i$ - $f_i$
  $f_{i+1}$ = subSamp2($l_i$)
}
```

http://sepwww.stanford.edu/~morgan/texturematch/paper_html/node3.html

What do you need to construct the original image?

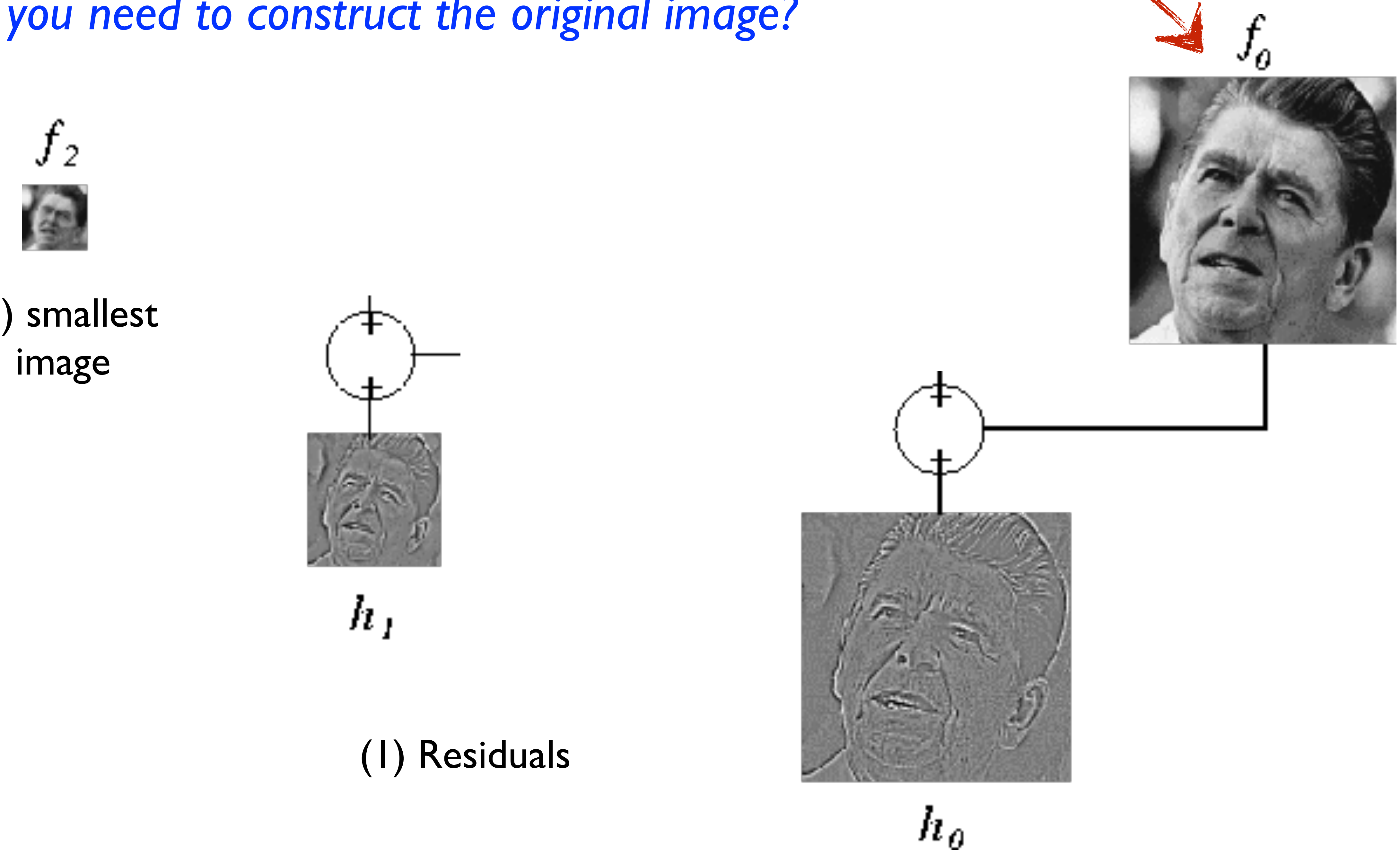


What do you need to construct the original image?



(1) Residuals

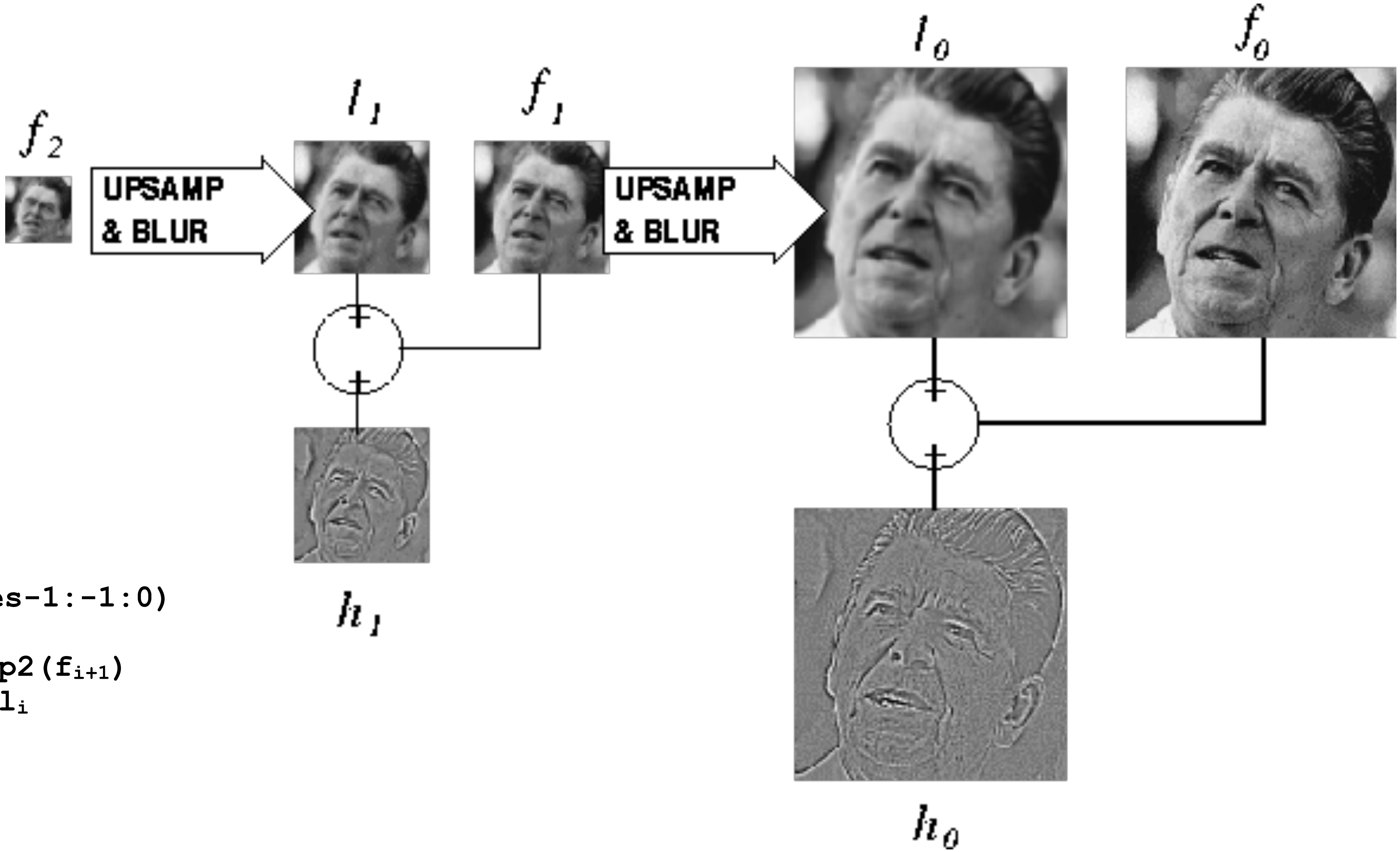
What do you need to construct the original image?



(2) smallest image

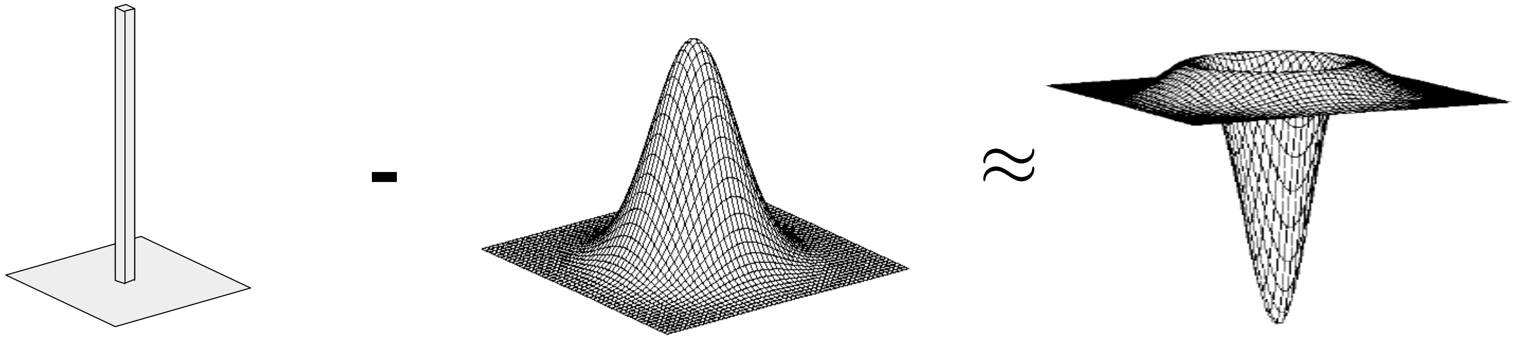
(1) Residuals

Reconstructing the original image



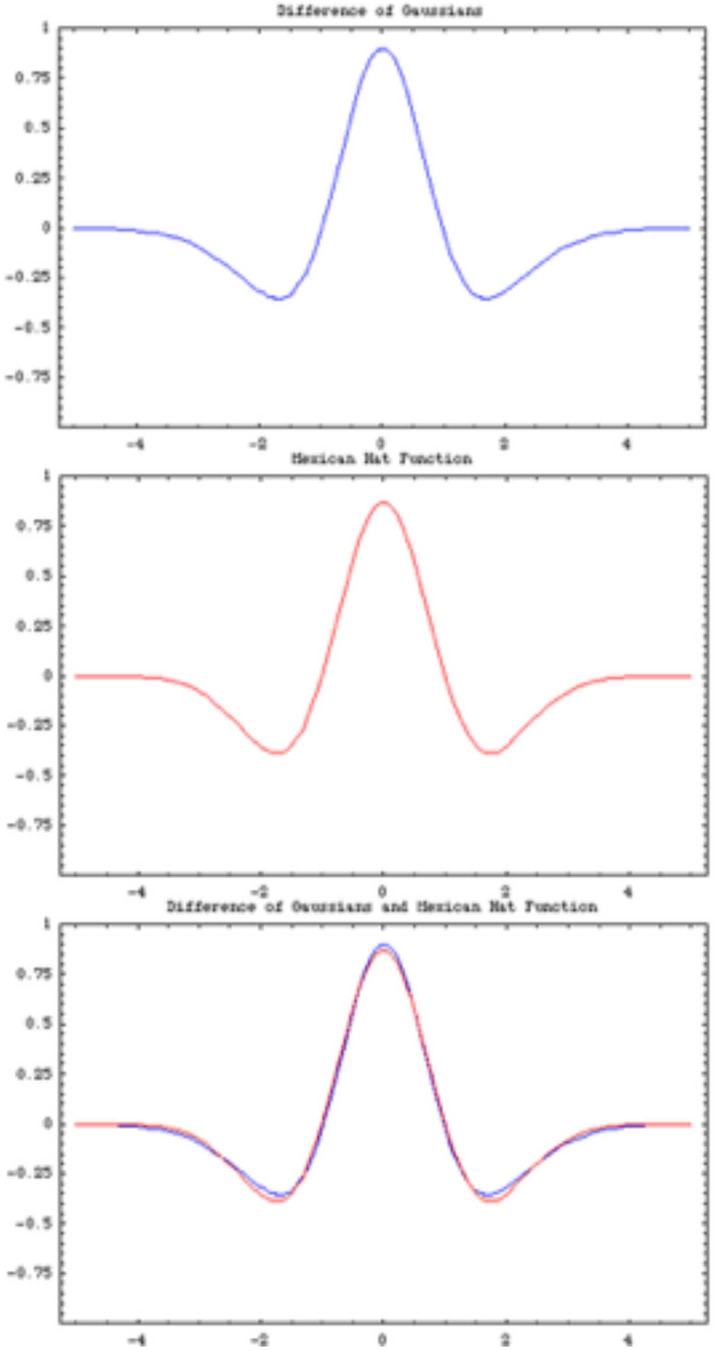
```
do( i = nScales-1:-1:0)
{
    $l_i$ = upSamp2($f_{i+1}$)
    fi = $h_i$ + $l_i$
}

output: $f_0$
```

Why is it called the Laplacian Pyramid?



unit Gaussian Laplacian

Difference of Gaussians approximates the Laplacian



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

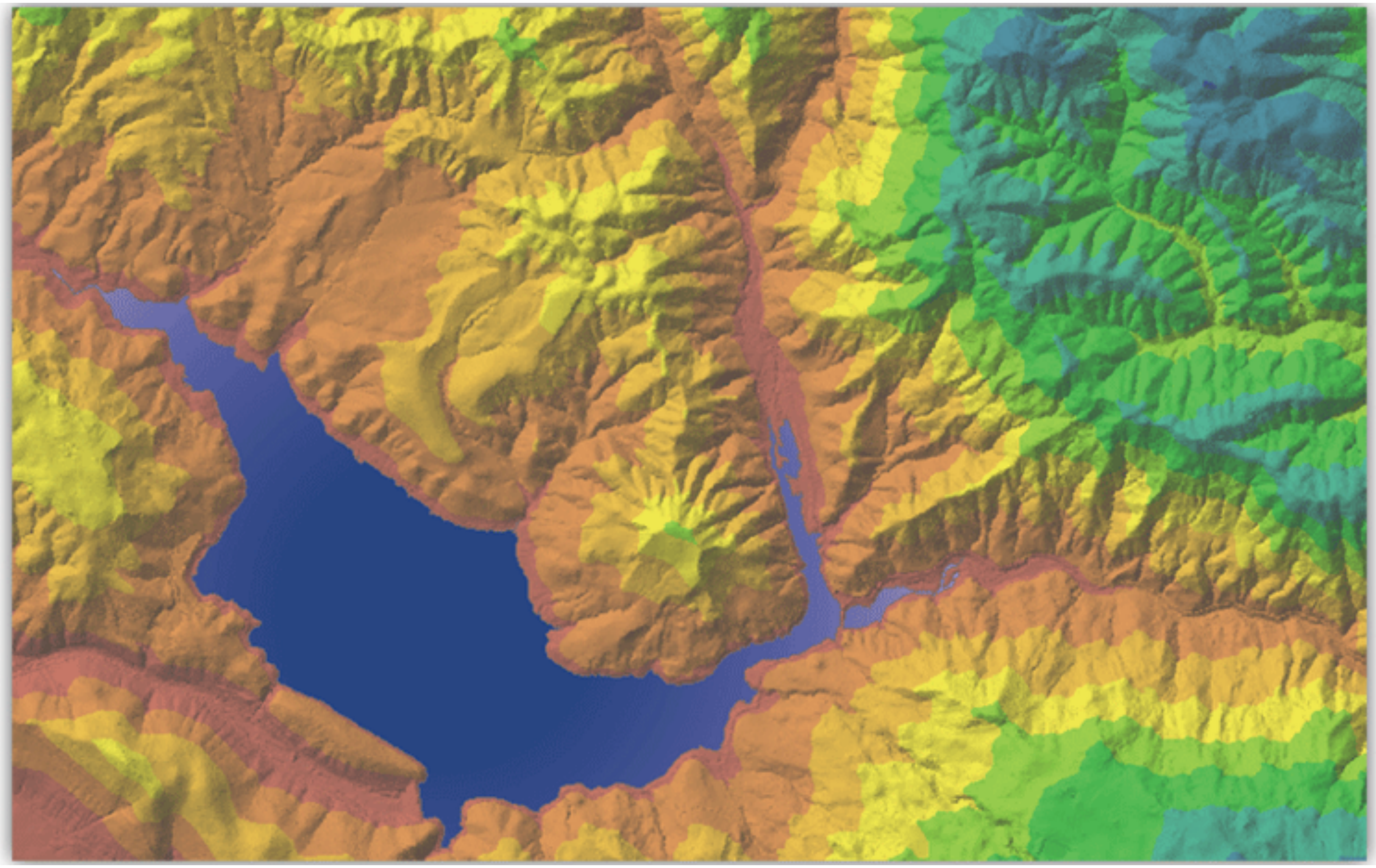
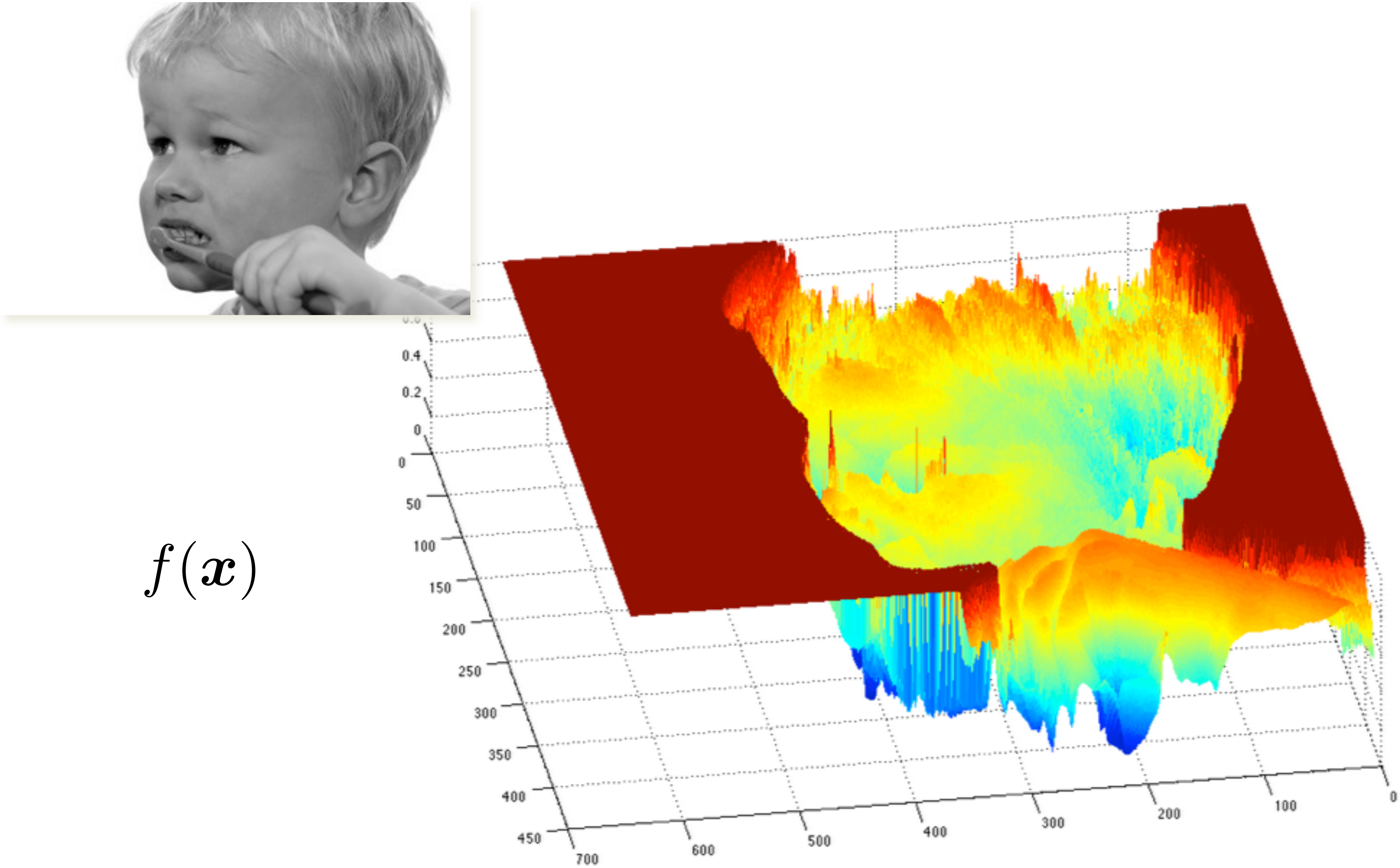


Image Gradients and Gradient Filtering

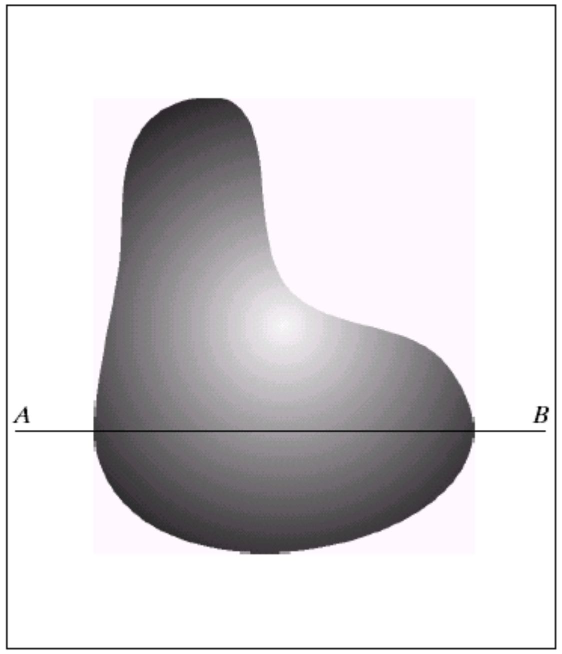
16-385 Computer Vision

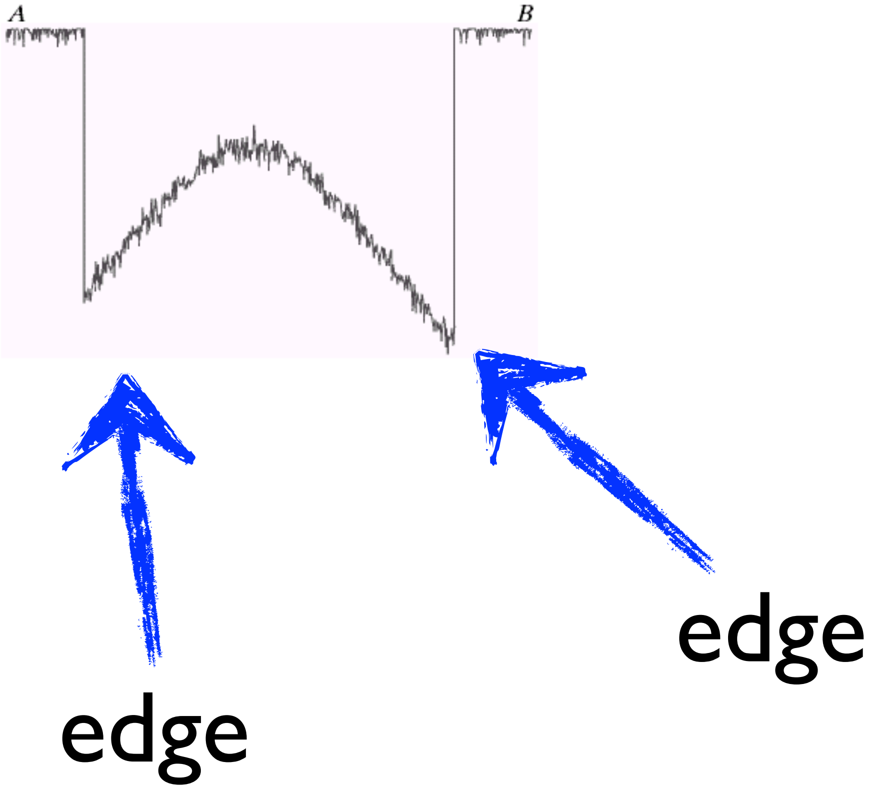
What is an image edge?

Recall that an image is a 2D function



$$f(\boldsymbol{x})$$





edge

edge

How would you detect an edge?

What kinds of filter would you use?

The ‘Sobel’ filter

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

a derivative filter
(with some smoothing)

Filter returns large response on vertical or horizontal lines?

The ‘Sobel’ filter

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

a derivative filter
(with some smoothing)

Filter returns large response on vertical or horizontal lines?

Is the output always positive?

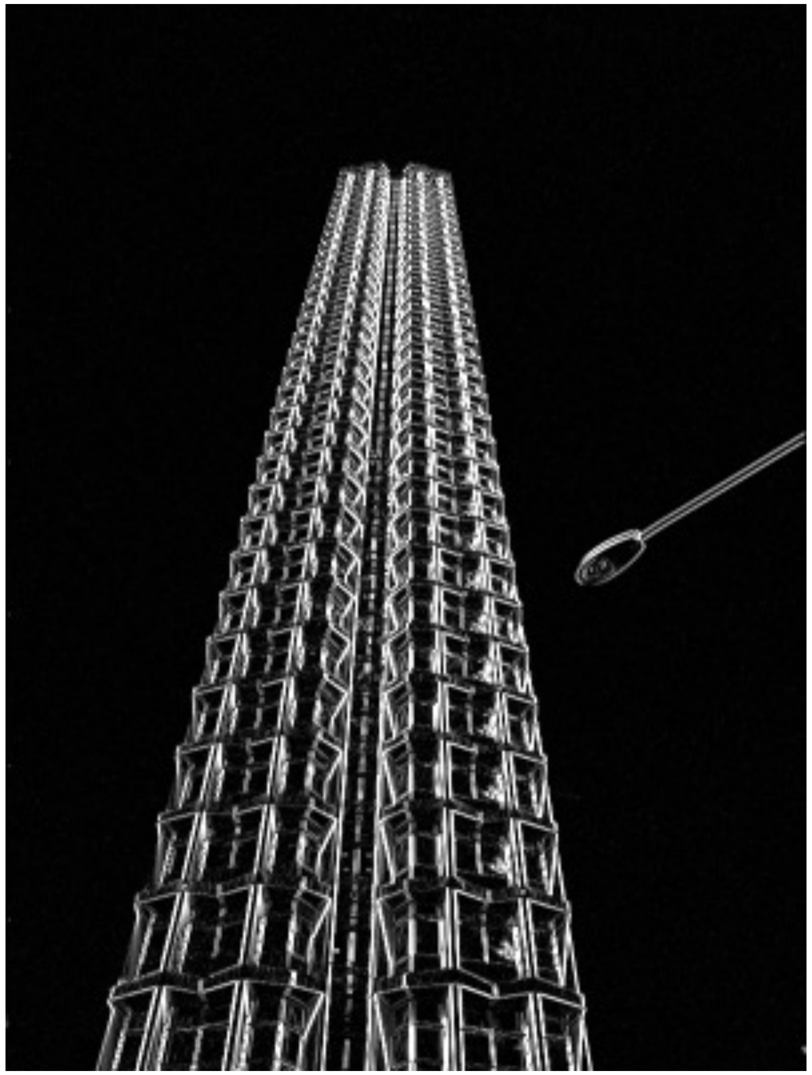
The ‘Sobel’ filter

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

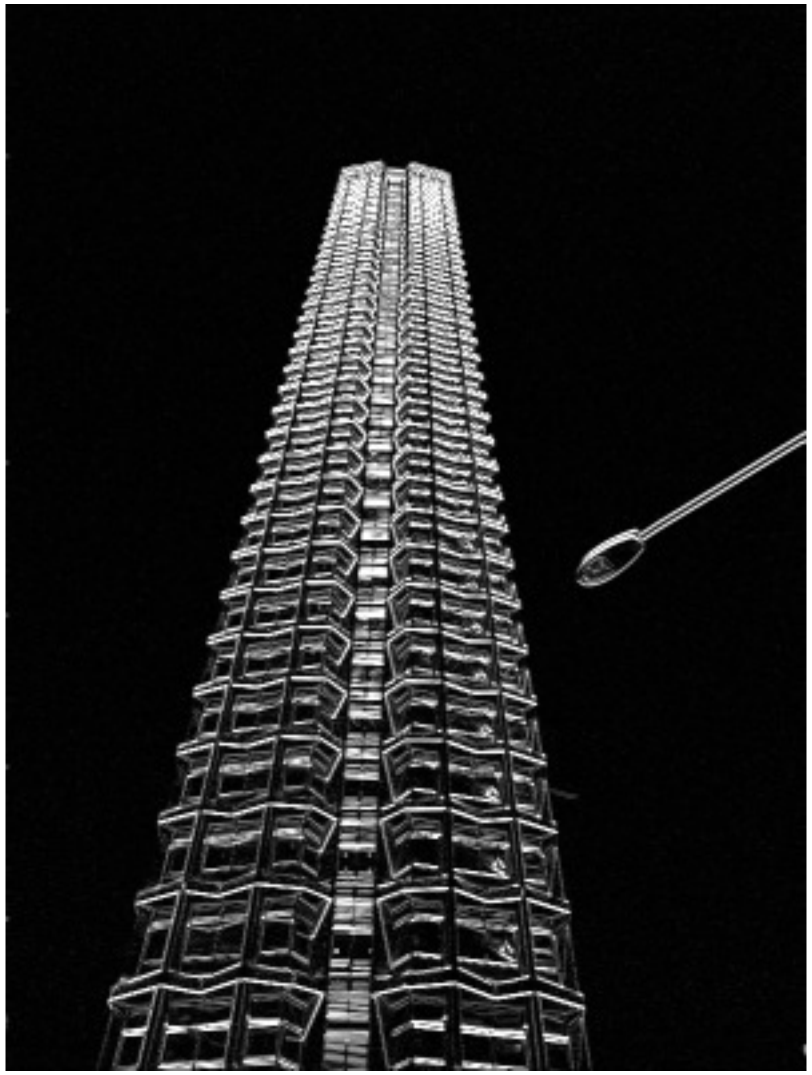
a derivative filter
(with some smoothing)

Responds to horizontal lines

Output can be positive or negative

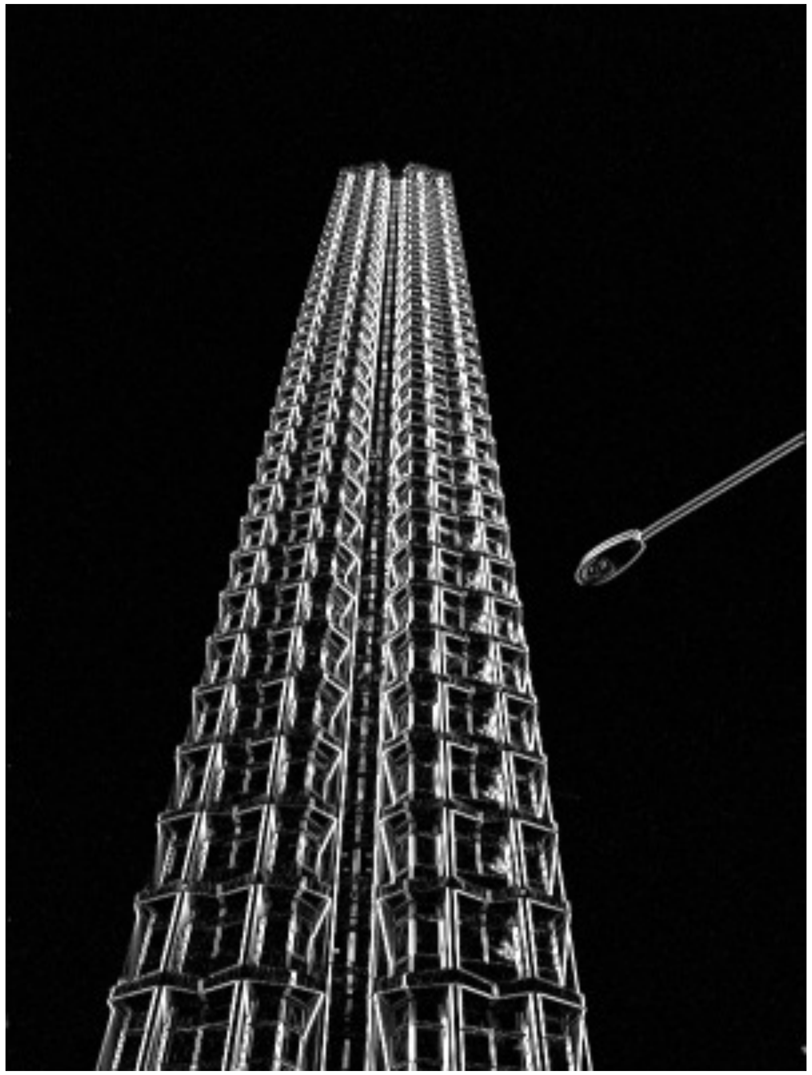


Output of which Sobel filter?

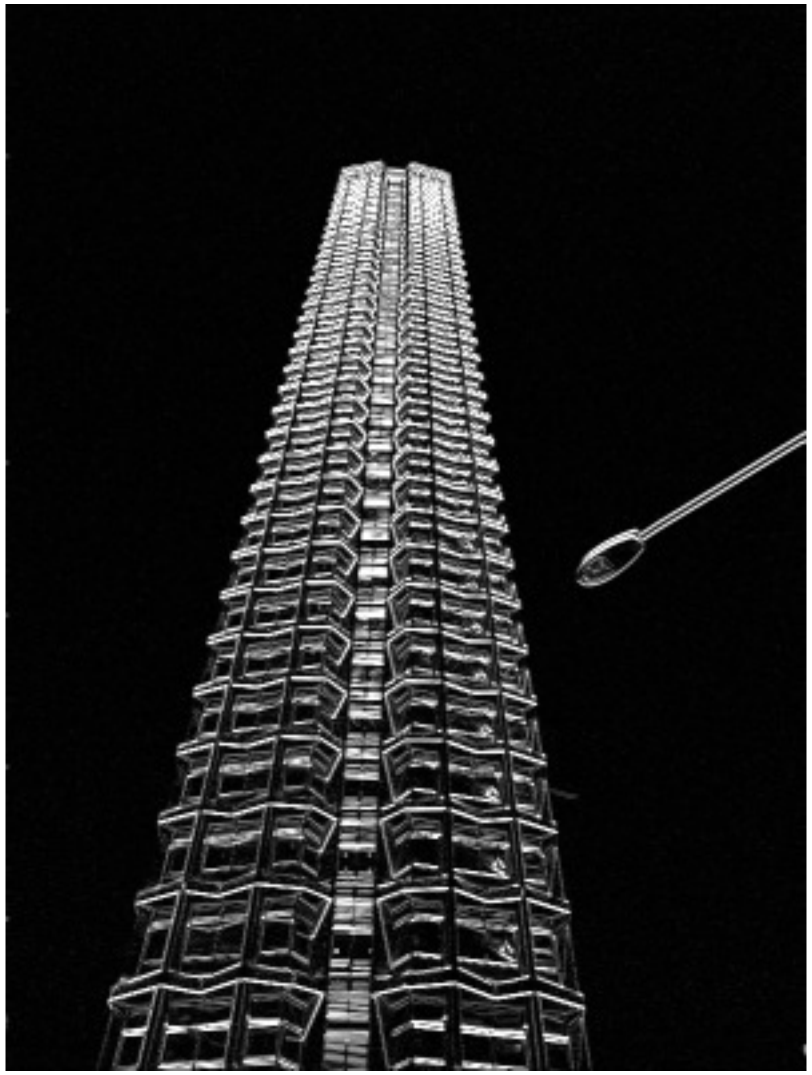


Output of which Sobel filter?

How do you visualize negative derivatives/gradients?



Derivative in X direction



Derivative in Y direction

Visualize with scaled absolute value

The ‘Sobel’ filter

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

Where does this filter come?

Do you remember this from high school?

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Do you remember this from high school?

The derivative of a function f at a point x is defined by the limit

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

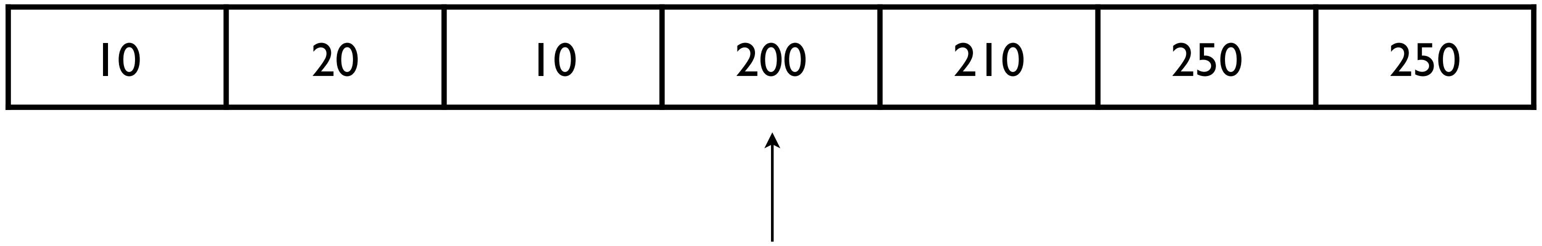
Approximation of the derivative when h is small

This definition is based on the ‘forward difference’ but ...

it turns out that using the ‘central difference’ is more accurate

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

How do we compute the derivative of a discrete signal?



10	20	10	200	210	250	250

it turns out that using the ‘central difference’ is more accurate

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

How do we compute the derivative of a discrete signal?

10	20	10	200	210	250	250

$$f'(x) = \frac{f(x+1) - f(x-1)}{2} = \frac{210 - 10}{2} = 100$$

-1	0	1

1D derivative filter

Decomposing the Sobel filter

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

Sobel

What this?

Decomposing the Sobel filter

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

Sobel

weighted average
and scaling

Decomposing the Sobel filter

$$\underset{\text{Sobel}}{\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}} = \underset{\substack{\text{weighted average}\\ \text{and scaling}}}{\begin{bmatrix}1\\ 2\\ 1\end{bmatrix}} \overset{\textit{What this?}}{\begin{bmatrix}1 & 0 & -1\end{bmatrix}}$$

Decomposing the Sobel filter

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

Sobel

weighted average
and scaling

What this?

x-derivative

The Sobel filter only returns the x and y edge responses.

How can you compute the **image gradient**?

How do you compute the image gradient?

Choose a derivative filter

$$\boldsymbol{S}_x = \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array}$$

$$\boldsymbol{S}_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

What is this filter called?

Run filter over image

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f}$$

$$\frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

What are the dimensions?

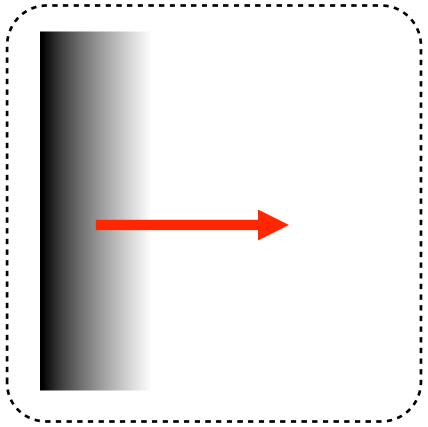
Image gradient

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

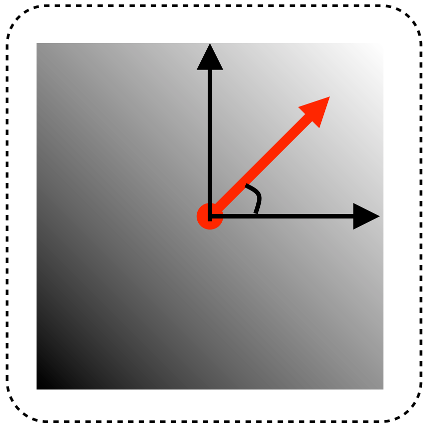
What are the dimensions?

Matching that Gradient !

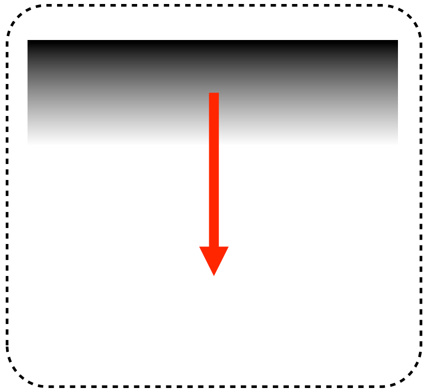
(a)



(b)



(c)



(1)

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

(2)

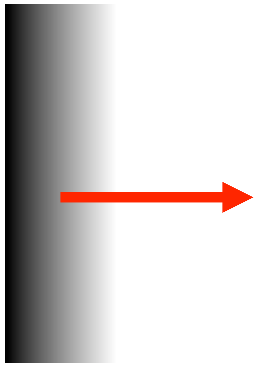
$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

(3)

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

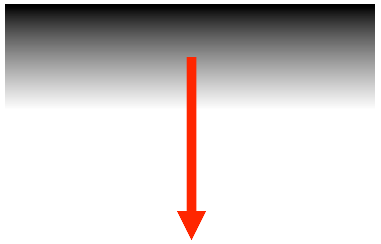
Image Gradient

Gradient in x only



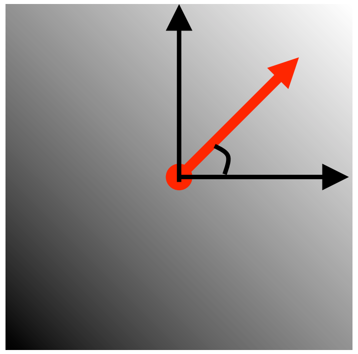
$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

Gradient in y only



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

Gradient in both x and y



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

Gradient direction

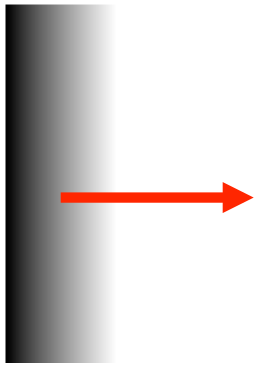
?

Gradient magnitude

?

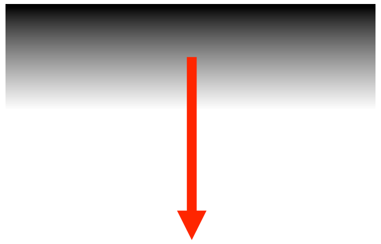
Image Gradient

Gradient in x only



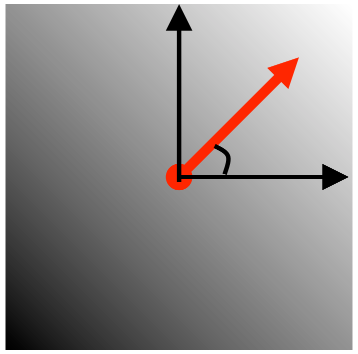
$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

Gradient in y only



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

Gradient in both x and y



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

Gradient direction

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

How does the gradient direction relate to the edge?

Gradient magnitude

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

What does a large magnitude look like in the image?

Common ‘derivative’ filters

Sobel

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

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

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

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

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

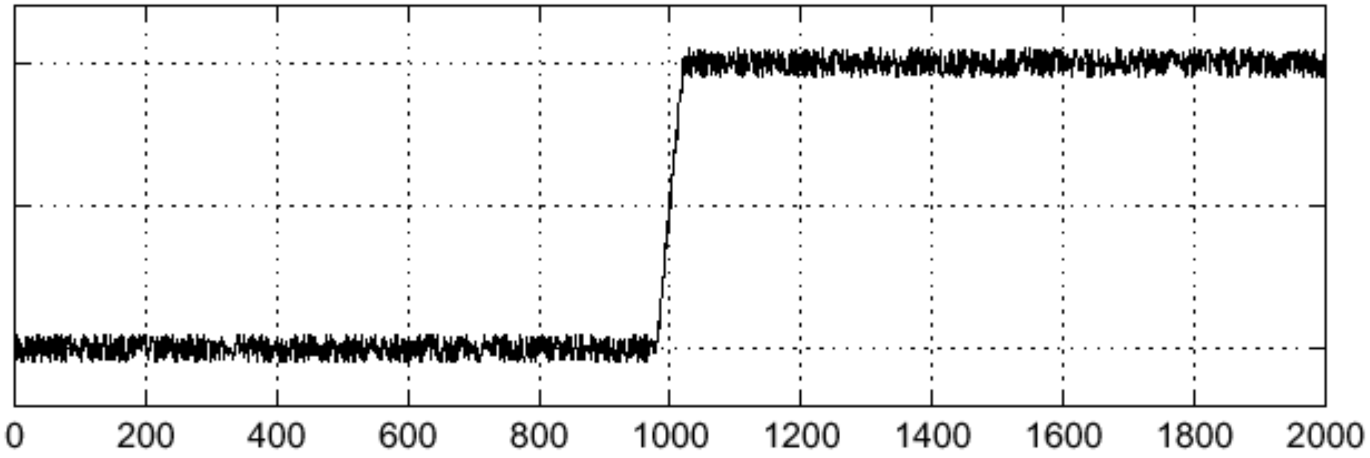
Roberts

0	1
-1	0

1	0
0	-1

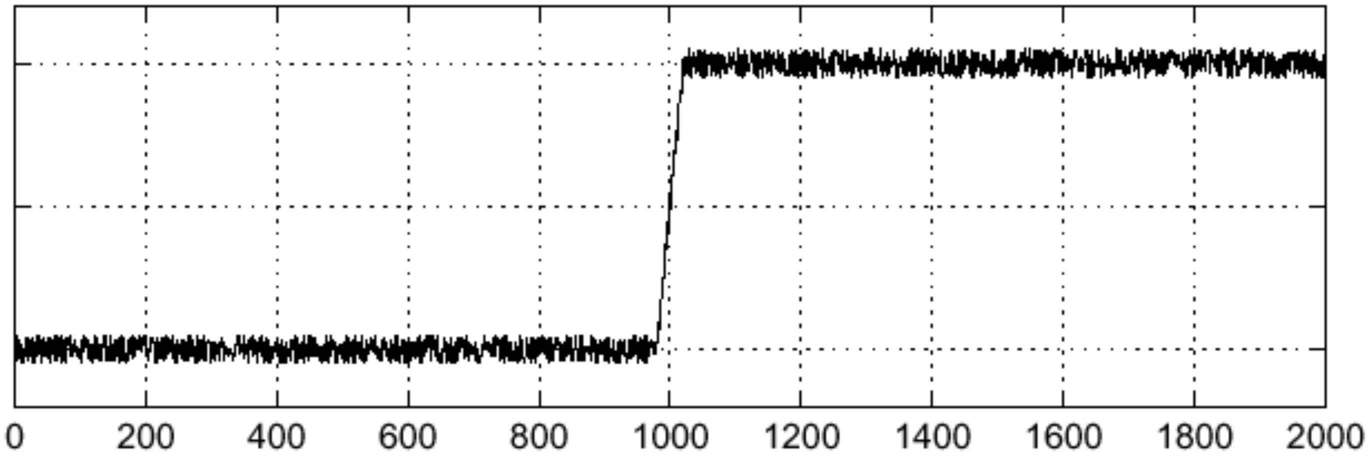
How do you find the edge from this signal?

Intensity plot



How do you find the edge from this signal?

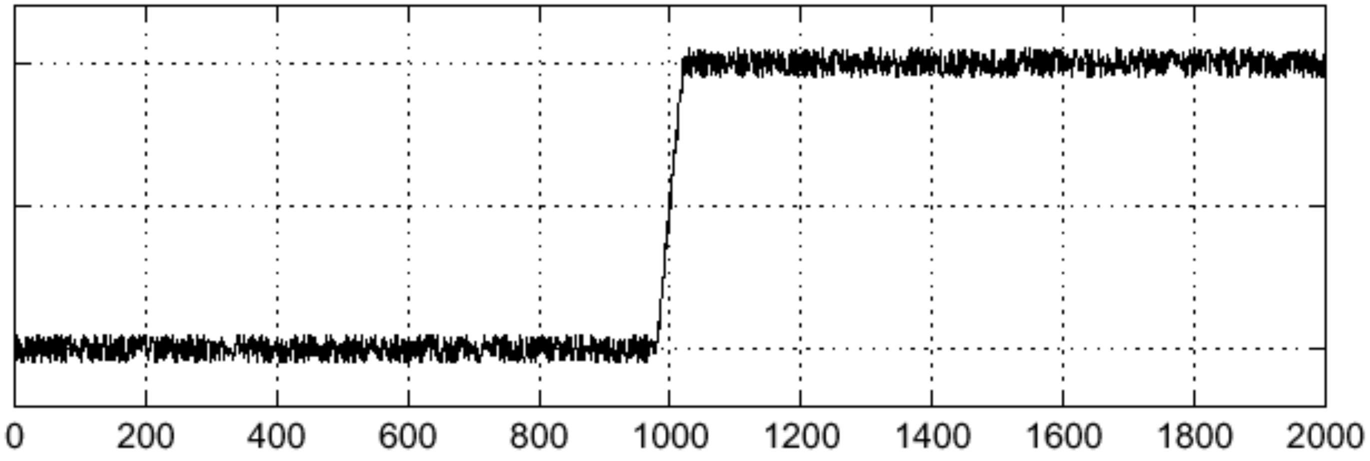
Intensity plot



Use a derivative filter!

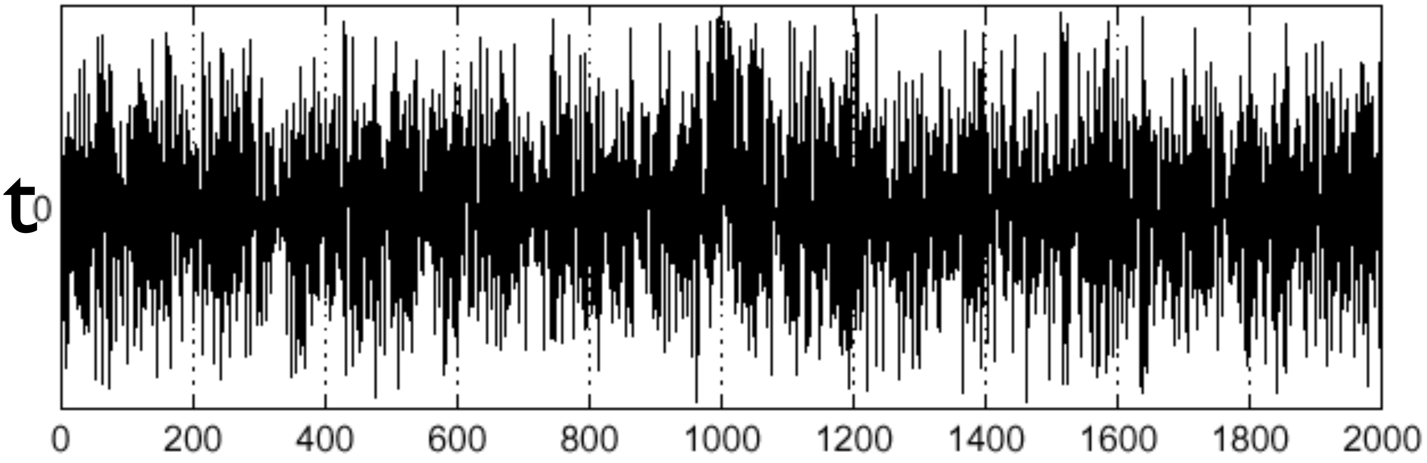
How do you find the edge from this signal?

Intensity plot



Use a derivative filter!

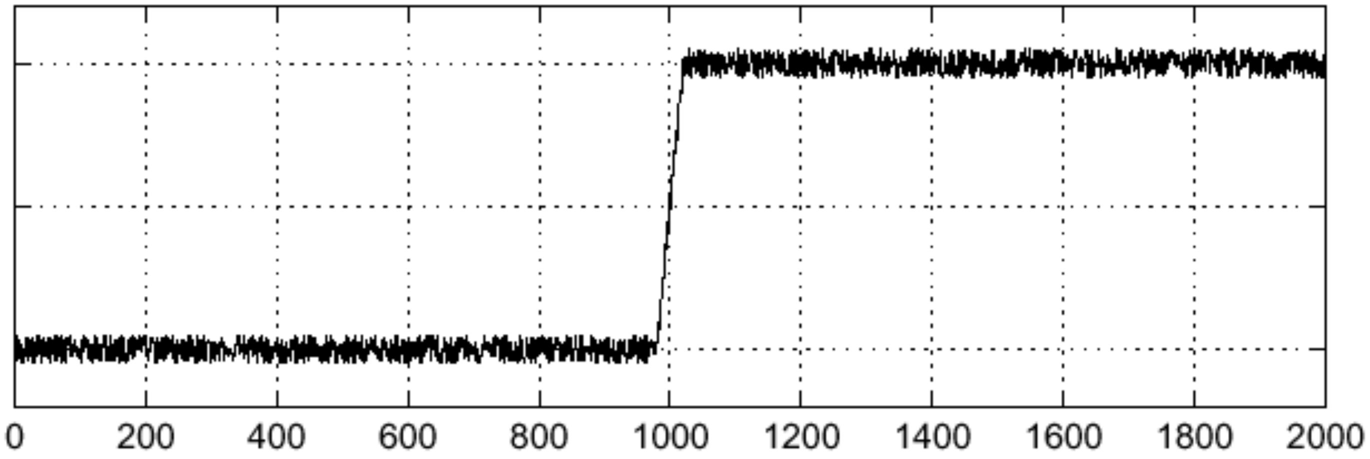
Derivative plot



What happened?

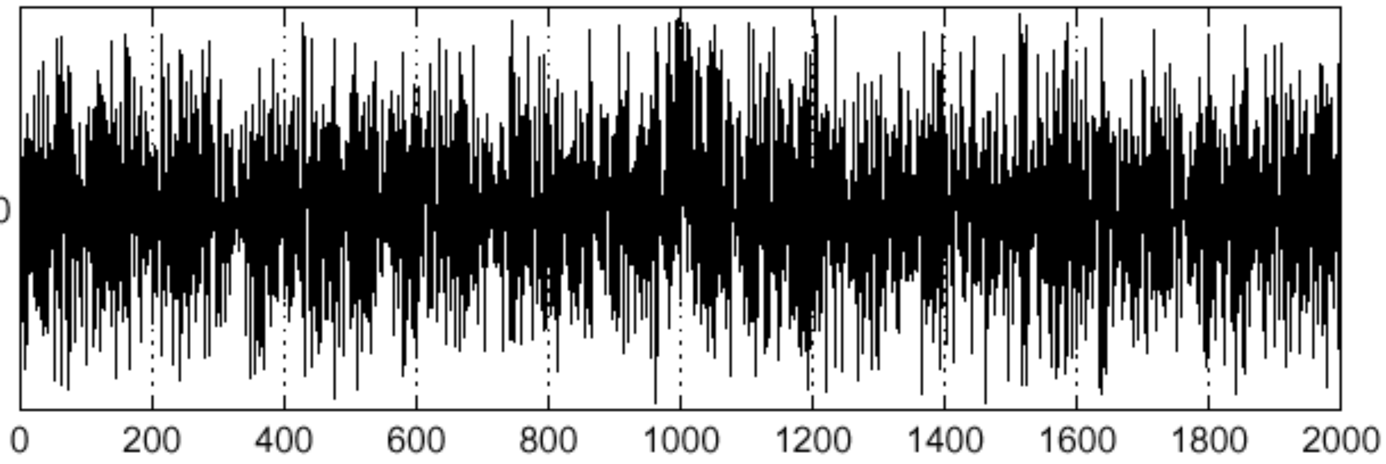
How do you find the edge from this signal?

Intensity plot



Use a derivative filter!

Derivative plot



Derivative filters are sensitive to noise

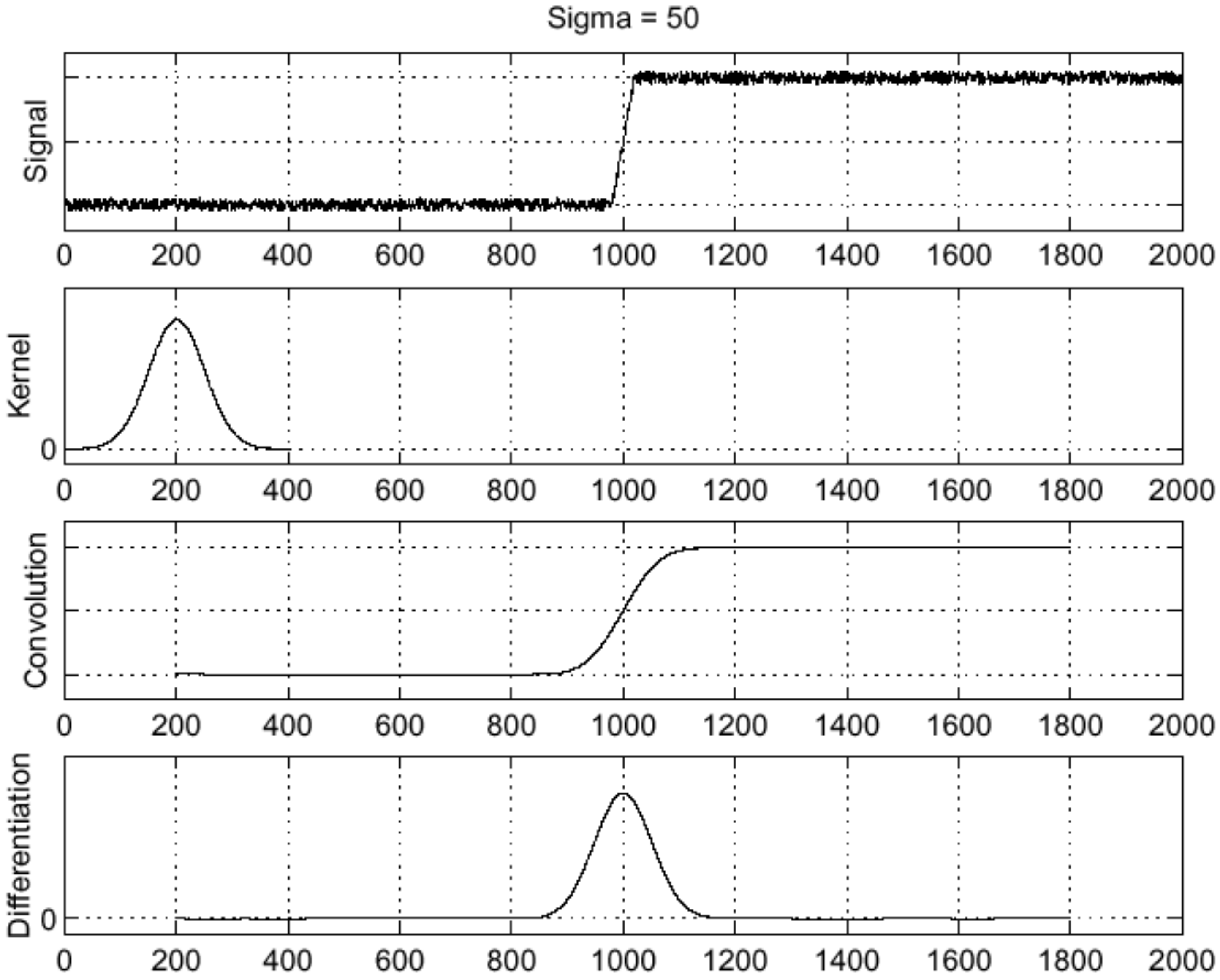
Input

Gaussian

Smoothed input

Derivative

Output



Don't forget to smooth before running derivative filters!

Laplace filter

A.K.A. Laplacian, Laplacian of Gaussian (LoG), Marr filter, Mexican Hat Function

Laplace filter

A.K.A. Laplacian, Laplacian of Gaussian (LoG), Marr filter, Mexican Hat Function

Laplace filter

A.K.A. Laplacian, Laplacian of Gaussian (LoG), Marr filter, Mexican Hat Function

first-order
finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

derivative filter

1	0	-1

second-order
finite difference

$$f''(x) \approx \frac{\delta_h^2[f](x)}{h^2} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

Laplace filter

?	?	?

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

derivative filter

1	0	-1

second-order finite difference

$$f''(x) \approx \frac{\delta_h^2[f](x)}{h^2} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

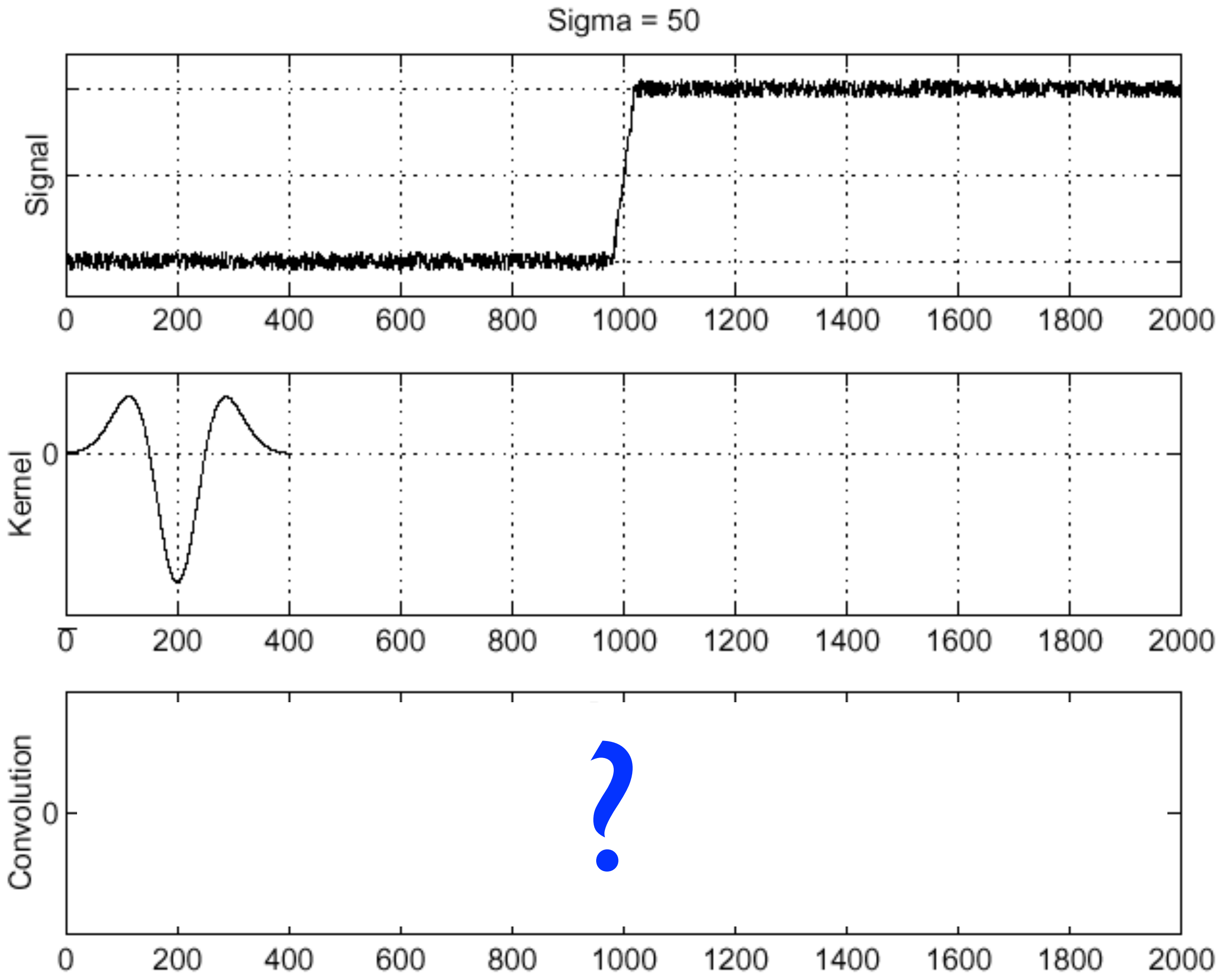
Laplace filter

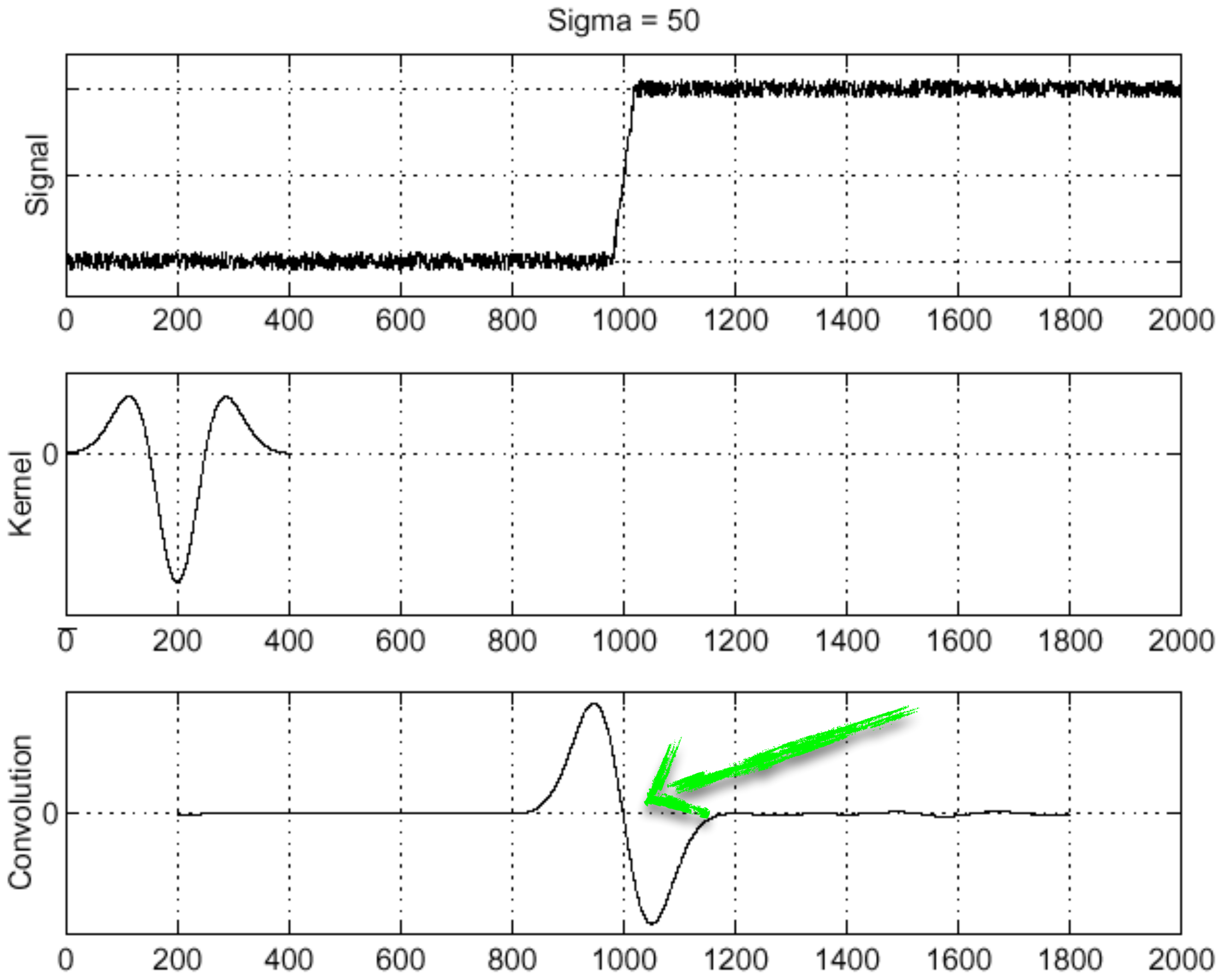
1	-2	1

Input

Laplacian

Output





Input

Laplacian

Output

Zero crossings are more accurate at localizing edges

Second derivative is noisy

2D Laplace filter

1	-2	1

1D Laplace filter

?	?	?
?	?	?
?	?	?

2D Laplace filter

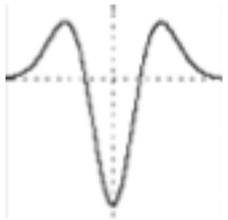
2D Laplace filter

1	-2	1

1D Laplace filter

?	?	?
?	?	?
?	?	?

2D Laplace filter



hint

2D Laplace filter

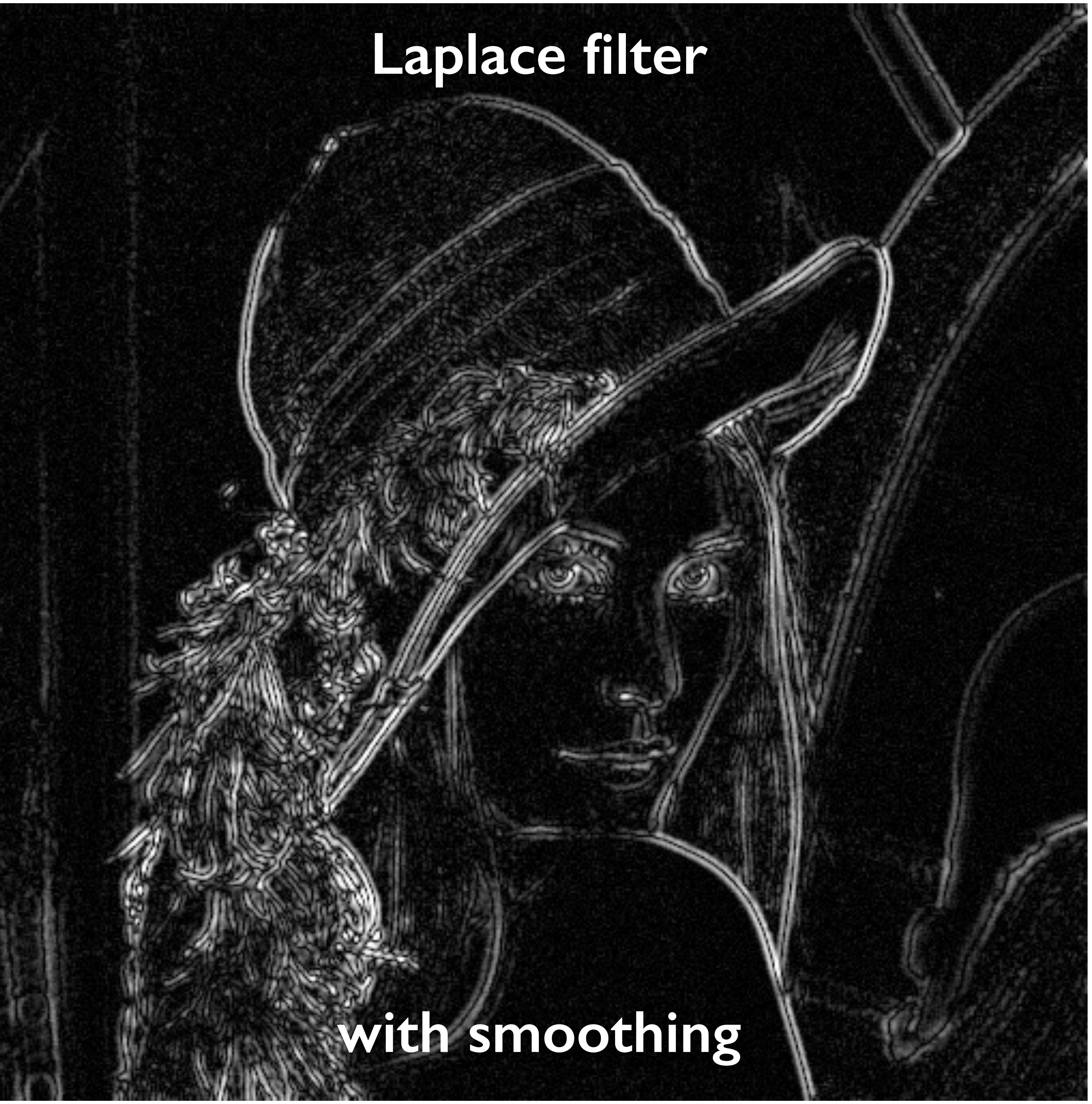
1	-2	1

1D Laplace filter

0	1	0
1	-4	1
0	1	0

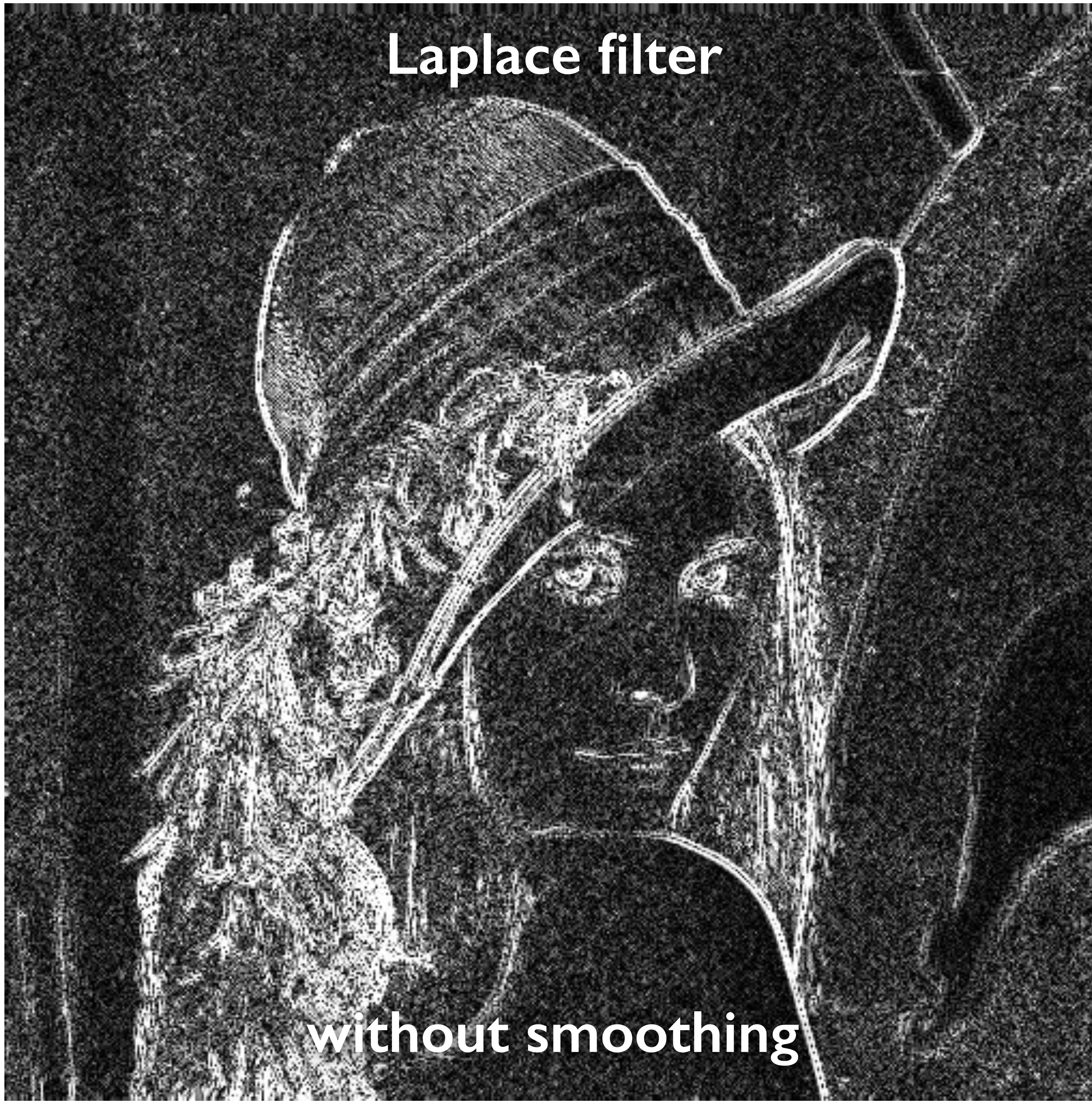
2D Laplace filter

If the Sobel filter approximates the first derivative, the Laplace filter approximates ....?



Laplace filter

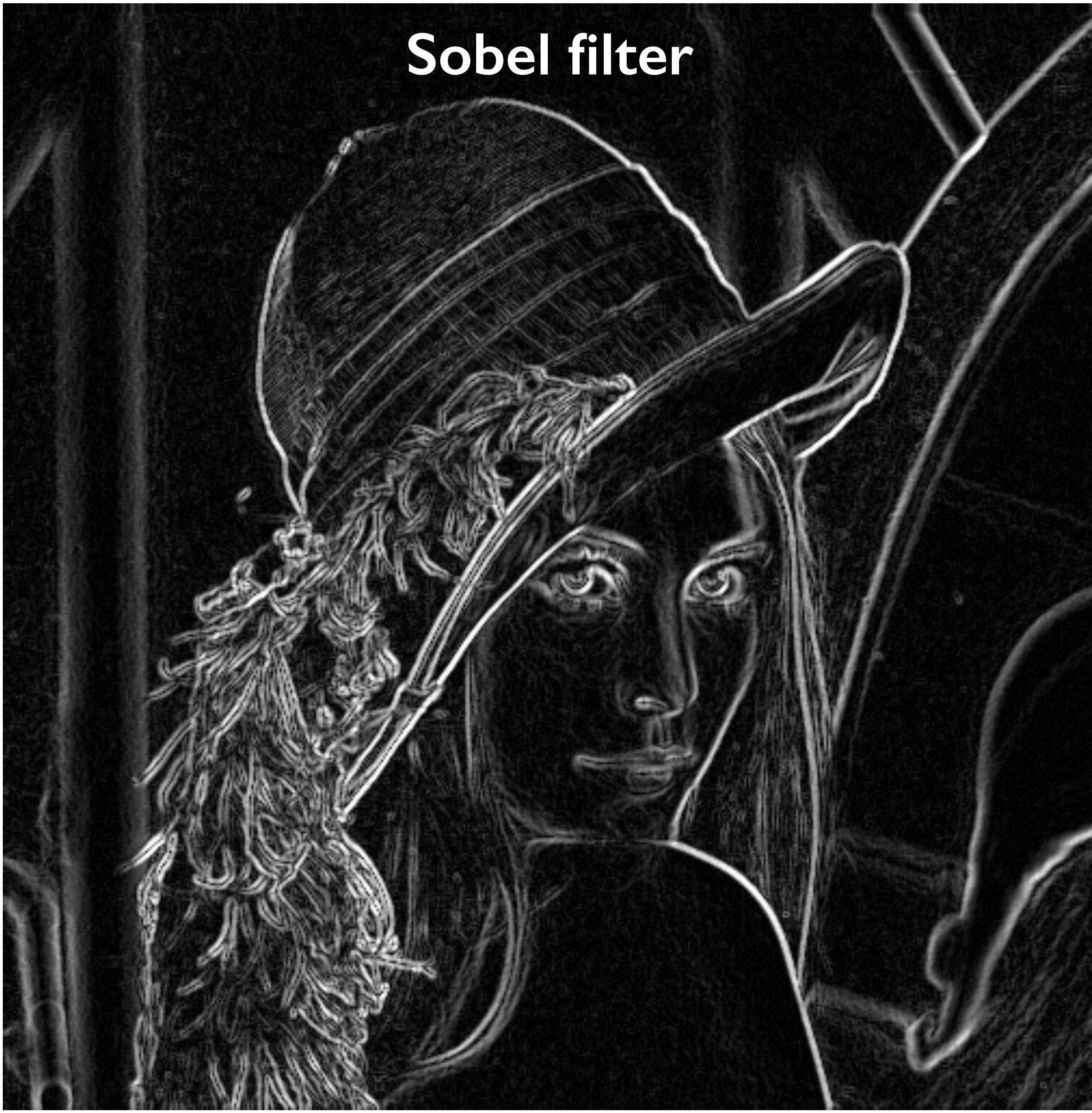
with smoothing



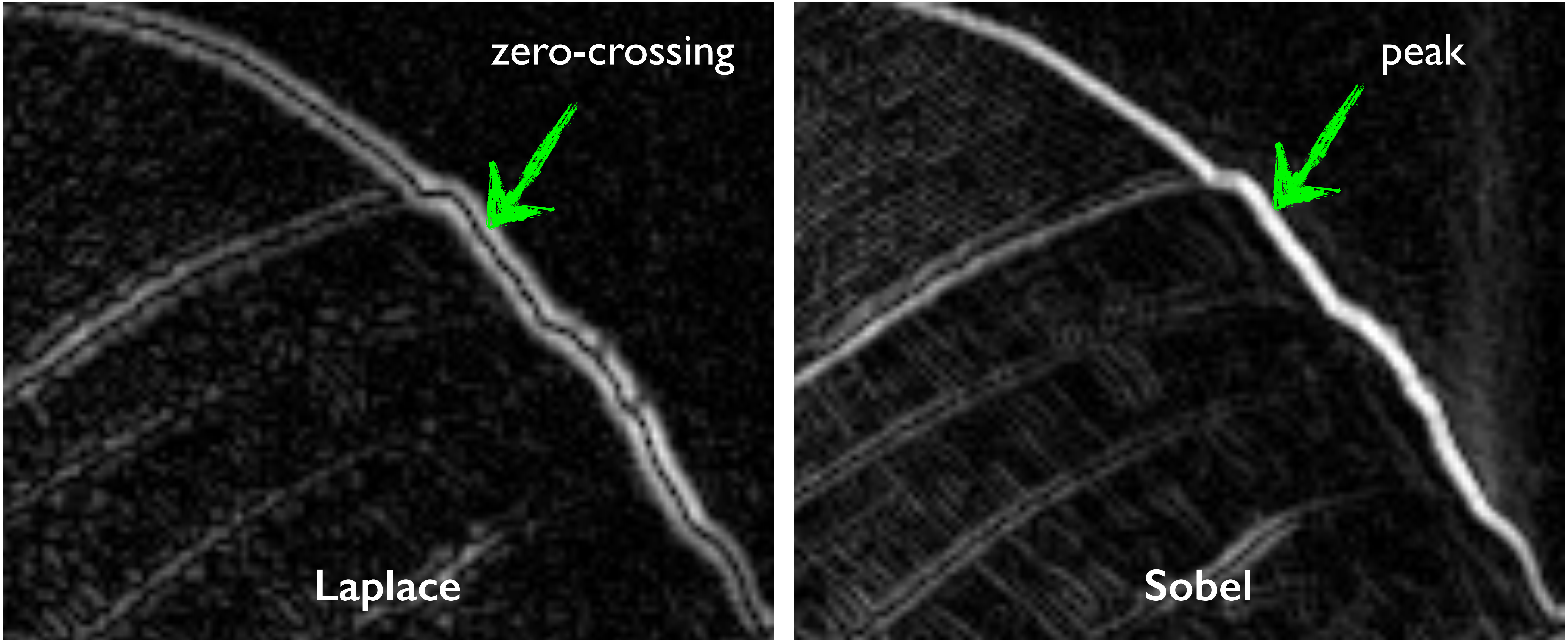
Laplace filter

without smoothing


Laplace filter


Sobel filter

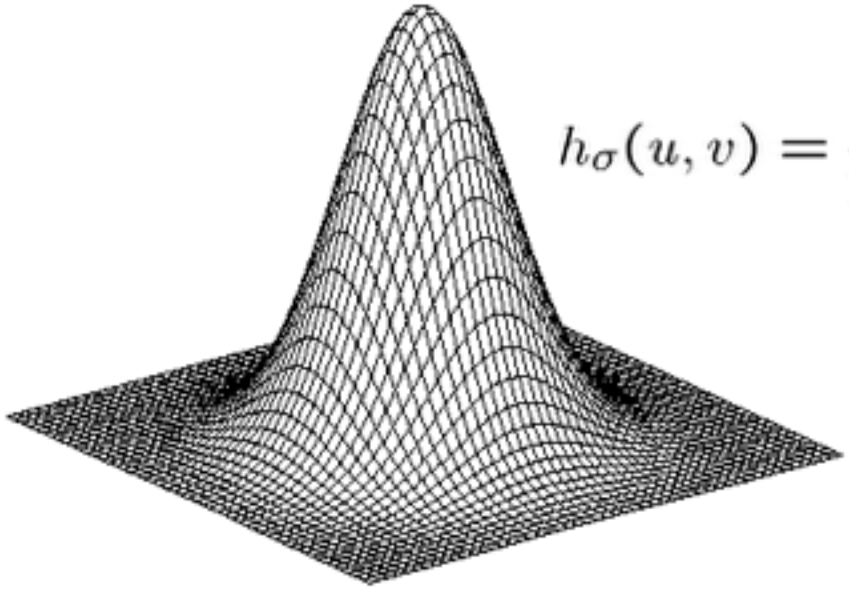
What's different between the two results?



Zero crossings are more accurate at localizing edges

(but not very convenient)

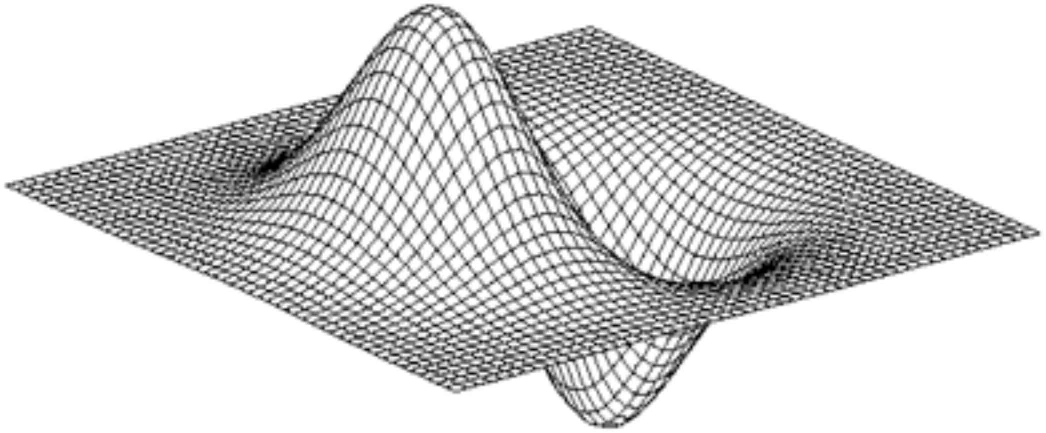
2D Gaussian Filters



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

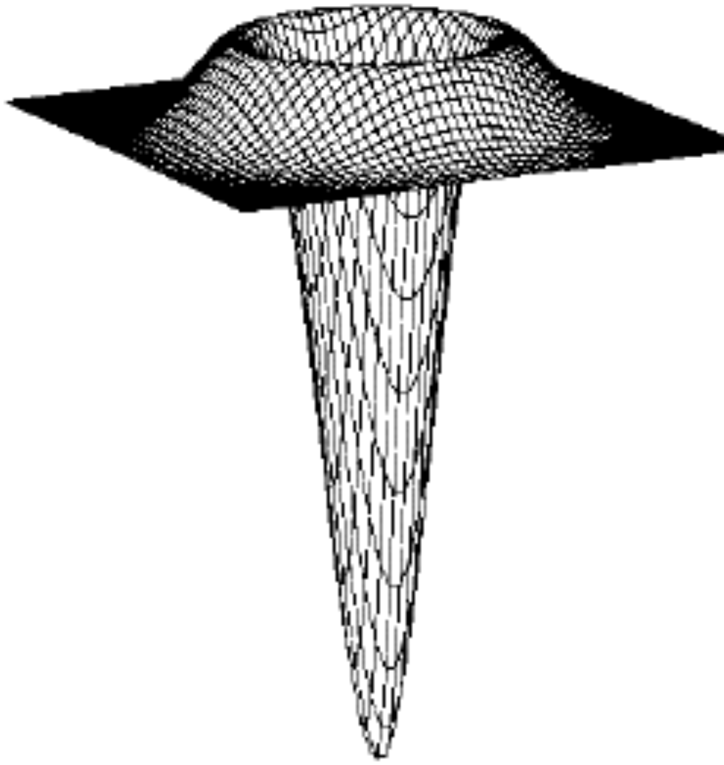
Gaussian

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



Derivative of Gaussian

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



Laplacian of Gaussian

Filtering vs Convolution

16-385 Computer Vision

Filters we have learned so far …

The 'Box' filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Gaussian filter

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

Sobel filter

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

Laplace filter

0	1	0
1	-4	1
0	1	0

Filtering vs Convolution

output filter image

filtering
(cross-correlation)

$h = g \otimes f$

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

What's the difference?

convolution

$h = g \star f$

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m-k, n-1]$$

Credit: Steve Seitz

Filtering vs Convolution

filtering

(cross-correlation)

$h = g \otimes f$

$$\overset{\text{output}}{h[m,n]} = \sum_{k,l} \overset{\text{filter}}{g[k,l]}\, \overset{\text{image}}{f[m+k, n+l]}$$

↓ filter flipped vertically and horizontally

convolution

$h = g \star f$

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m-k, n-1]$$

Filtering vs Convolution

filtering
(cross-correlation)

$h = g \otimes f$

$$\overset{\text{output}}{h[m,n]} = \sum_{k,l} \overset{\text{filter}}{g[k,l]}\, \overset{\text{image}}{f[m+k,n+l]}$$

↓ filter flipped vertically and horizontally

convolution

$h = g \star f$

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m-k,n-1]$$

Suppose g is a Gaussian filter.
How does convolution differ from filtering?

Recall...

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array}$$

Commutative

$$a \star b = b \star a \ .$$

Associative

$$(((a \star b_1) \star b_2) \star b_3) = a \star (b_1 \star b_2 \star b_3)$$

Distributes over addition

$$a \star (b + c) = (a \star b) + (a \star c)$$

Scalars factor out

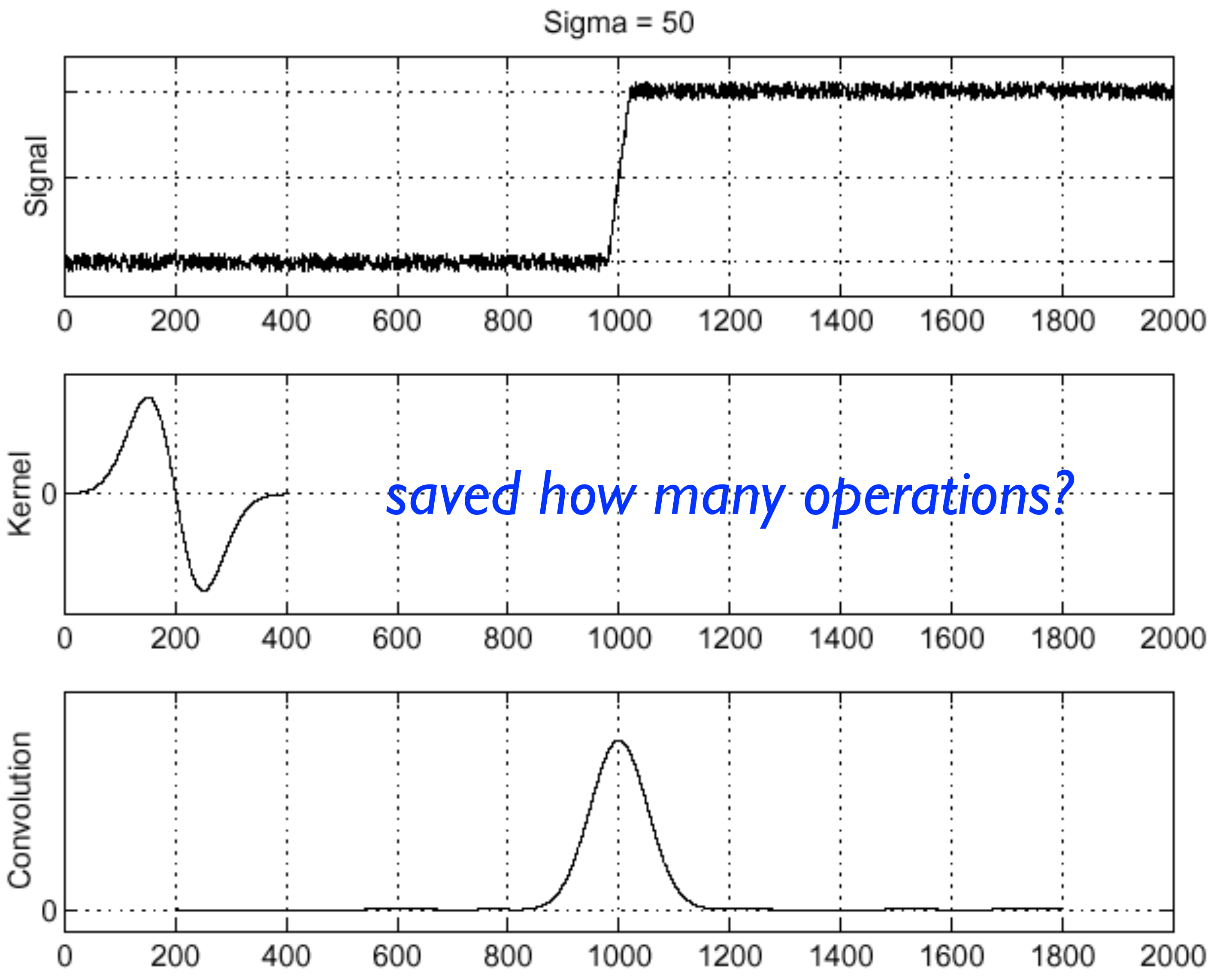
$$\lambda a \star b = a \star \lambda b = \lambda (a \star b)$$

Derivative Theorem of Convolution $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x} h) \star f$

can precompute this

Derivative Theorem of Convolution

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



saved how many operations?

Input

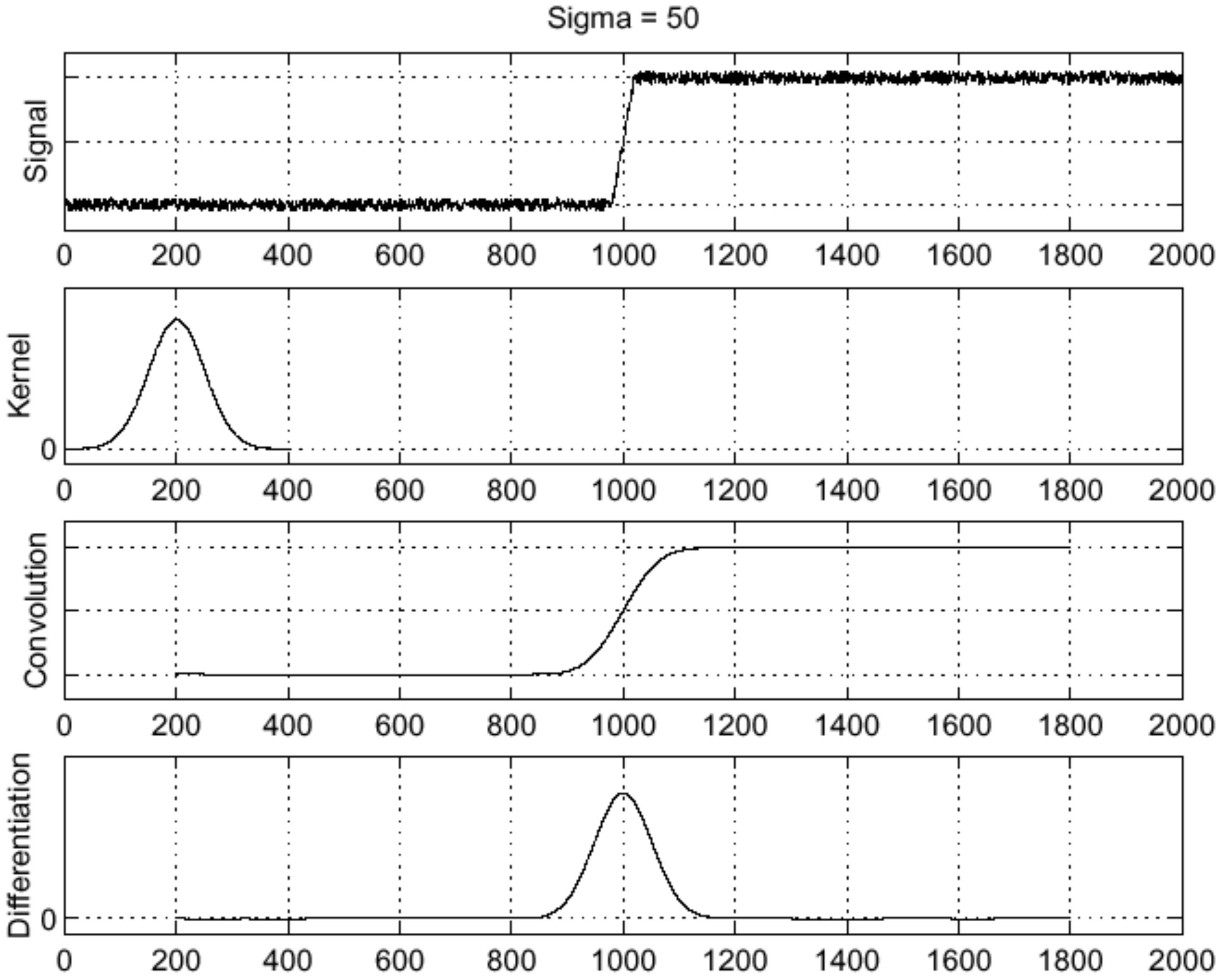
Derivative of Gaussian

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

Output

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

Recall ...



Input

Gaussian

Smoothed input

Derivative

Output