LUND UNIVERSITY

Image Analysis (FMAN20)
Lecture 3, 2019

MAGNUS OSKARSSON

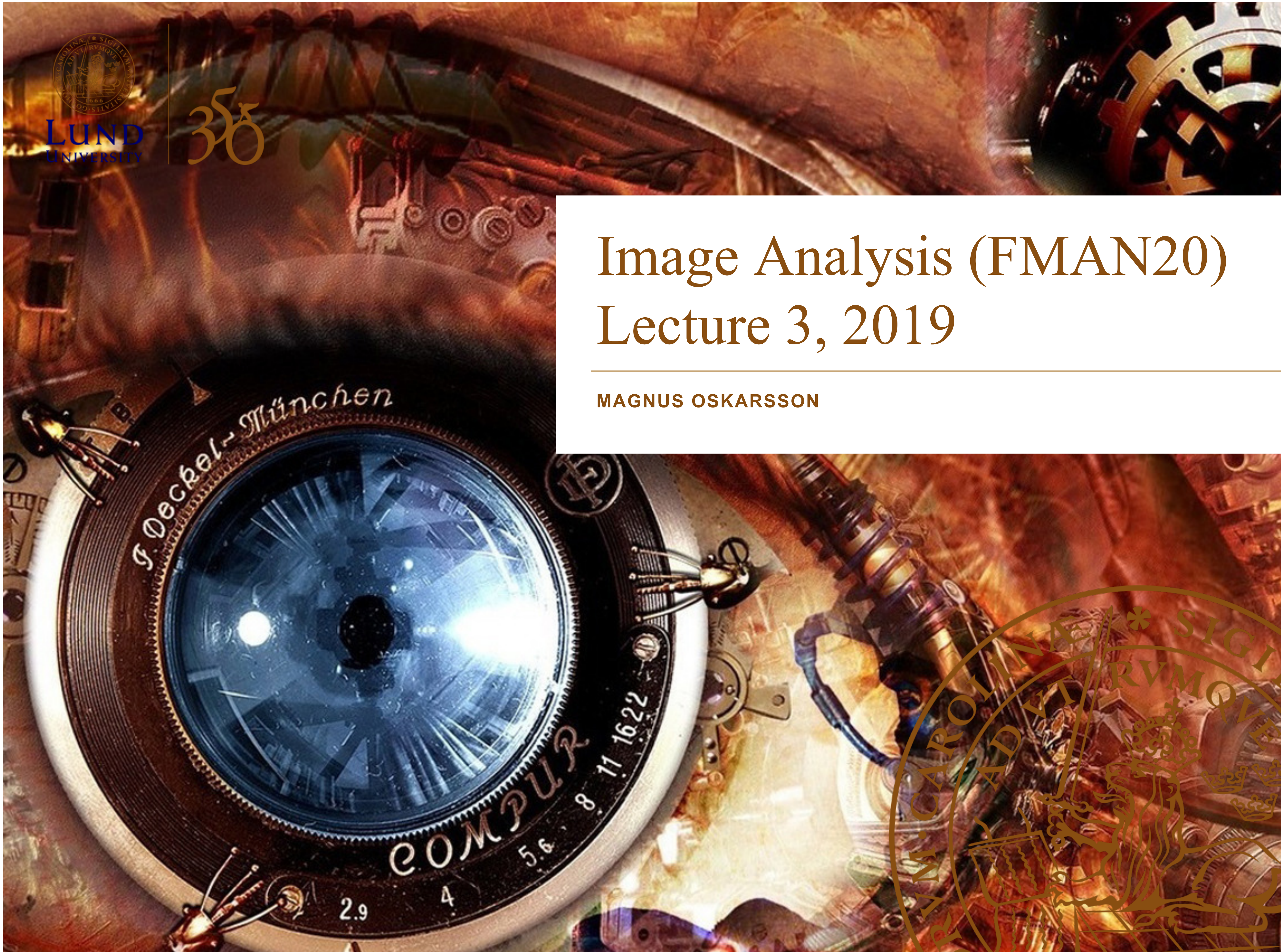


Image filtering - Motivation

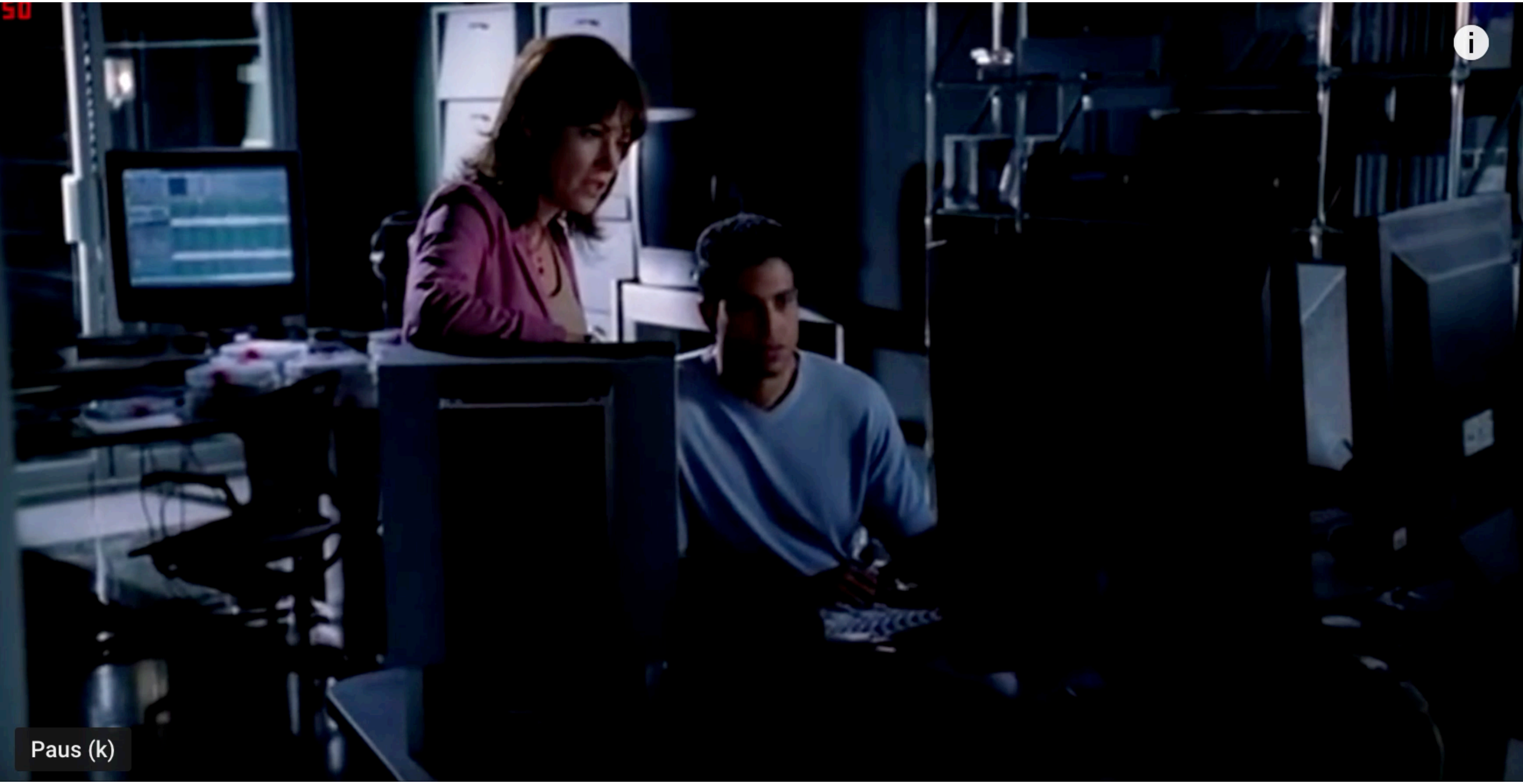
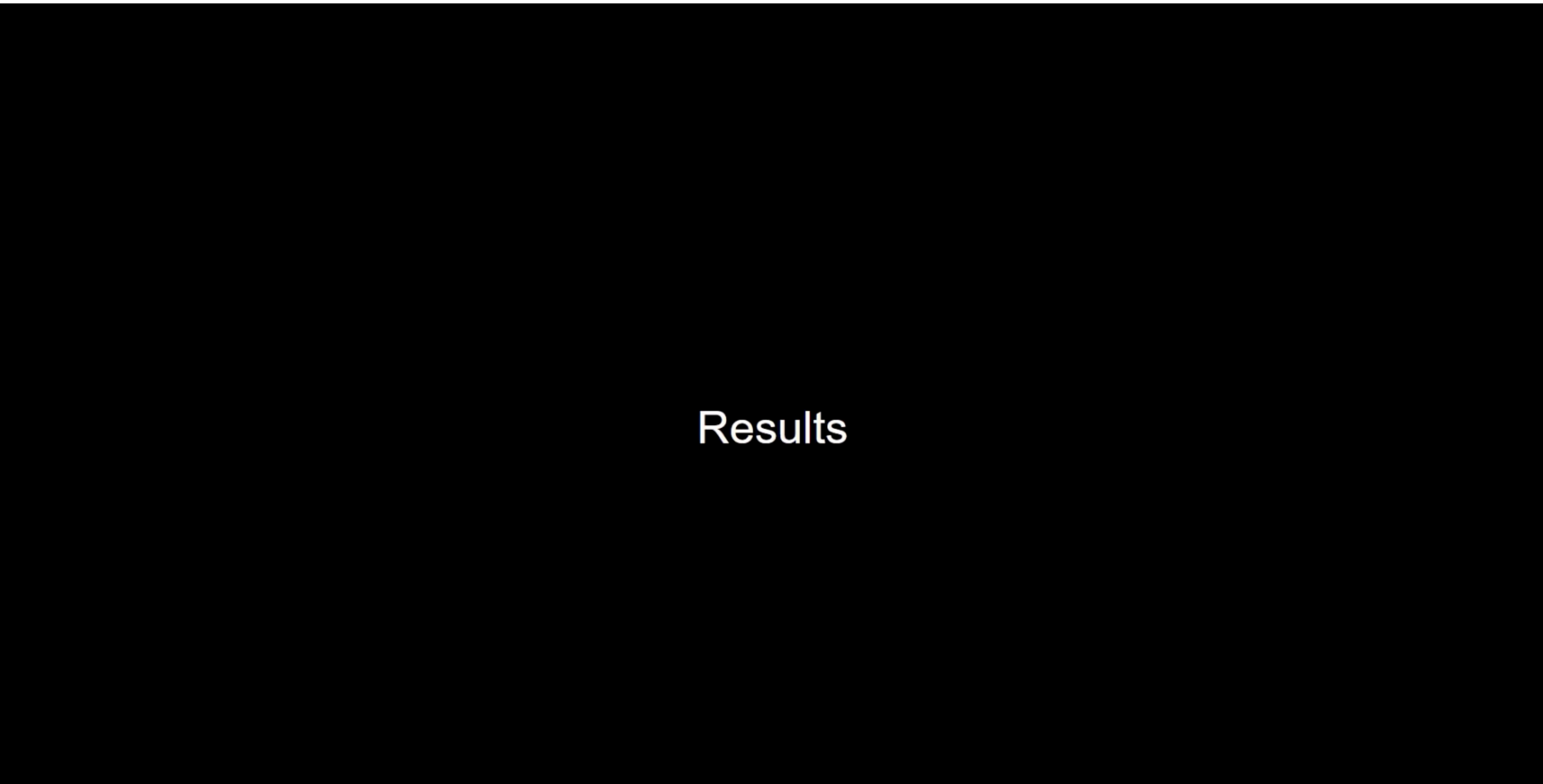
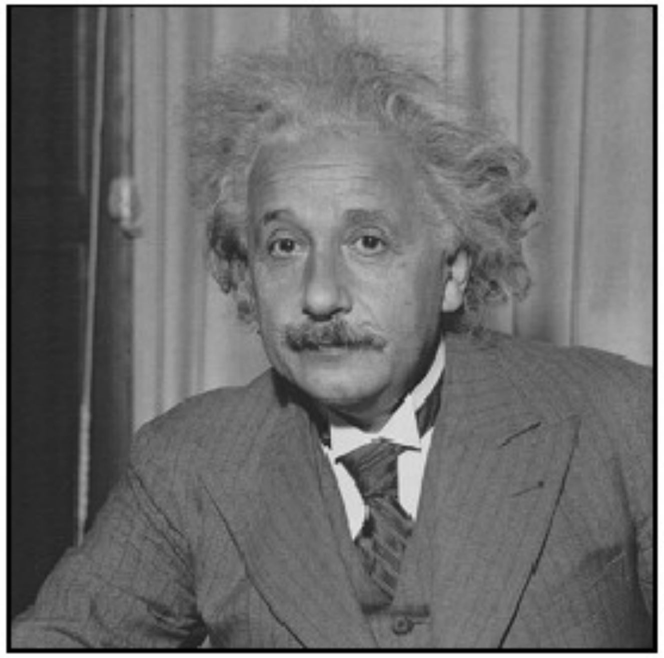
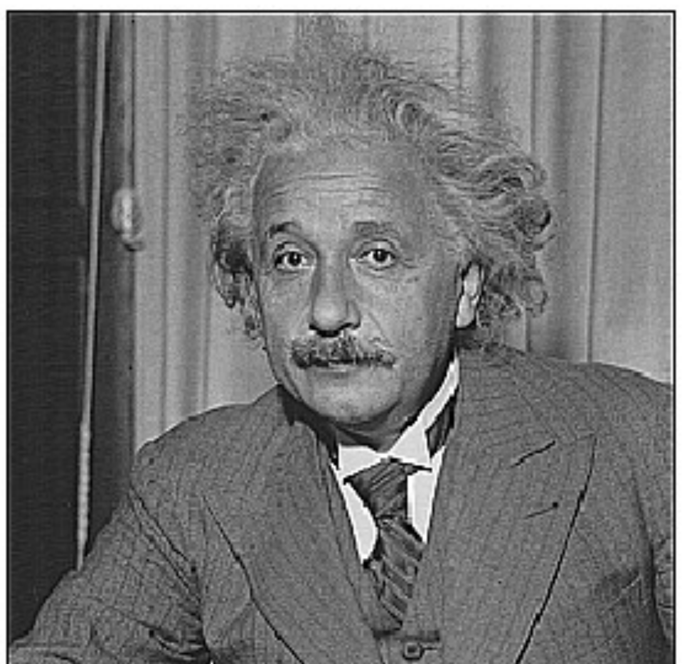


Image filtering - Motivation



Today: Image Filters


Smooth/Sharpen Images... *Find edges...* *Find waldo…*

Overview – Convolutions

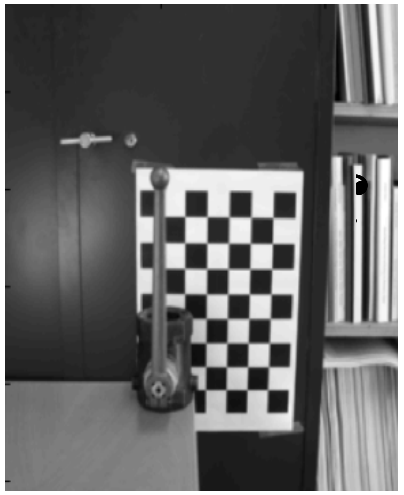
1. Convolution
   1. Definition, properties
   2. Convolution vs Cross-corellation
   3. Convolution and translation invariant linear systems
   4. Motivation using sliding means (1D and 2D)
   5. Interpretation as ‘sliding’ scalar product.
   6. Median Filter (not a convolution)
   7. Gaussian smoothing
   8. Derivatives + Smoothing
2. Convolution theorem
3. Connecting linear algebra, Fourier transform and convolutions

Convolution Operator

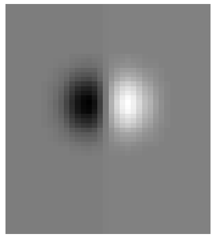
$$g = f * h$$

$$g(i,j) = \sum_u \sum_v f(i-u, j-v)h(u,v)$$

$=$



$*$



Cross-Correlation
Sliding scalar product

$$g(i,j) = \sum_y \sum_x f(i+y, j+x)\check{h}(y,x)$$

Compare with convolution

$$\boxed{\begin{aligned} g(i,j) &= \sum_u \sum_v f(i-u, j-v)h(u,v) \\ \check{h}(u,v) &= h(-u,-v) \end{aligned}}$$

Why use convolution?

Cross-correlation seems much simpler.

One motivation: Convolution has simpler calculation rules

$$f * h = h * f,$$

$$f * (g * h) = (f * g) * h,$$

$$f * (g + h) = f * g + f * h,$$

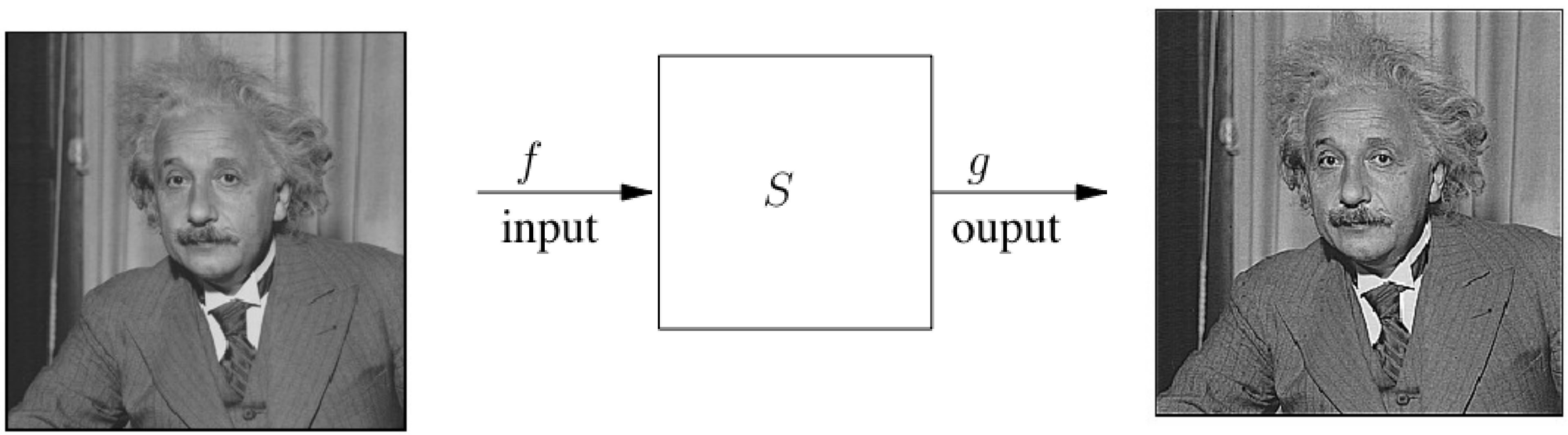
$$a(f * g) = (af) * g,$$

$$\delta * f = f,$$

$$\partial(f * g) = (\partial f) * g,$$

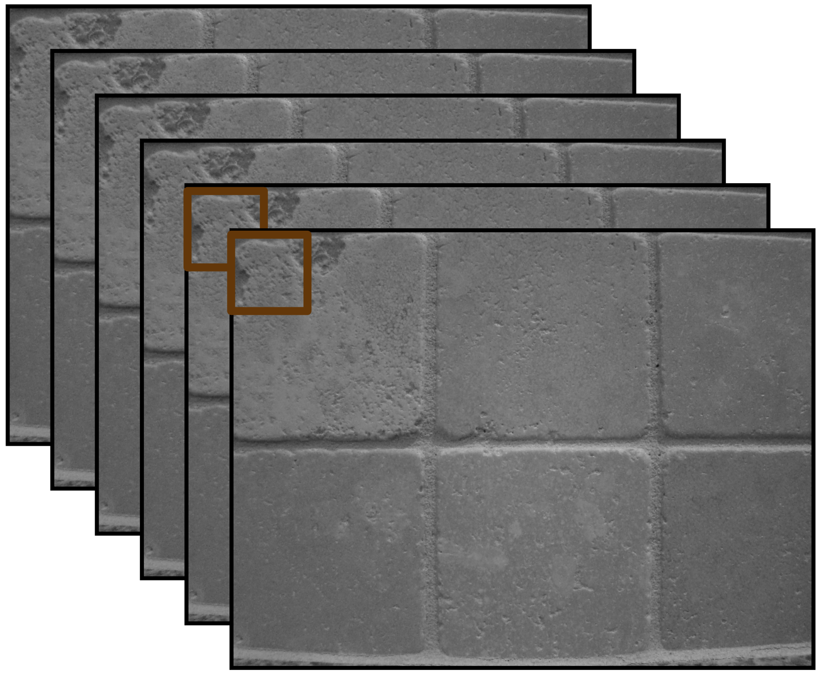
LUND UNIVERSITY

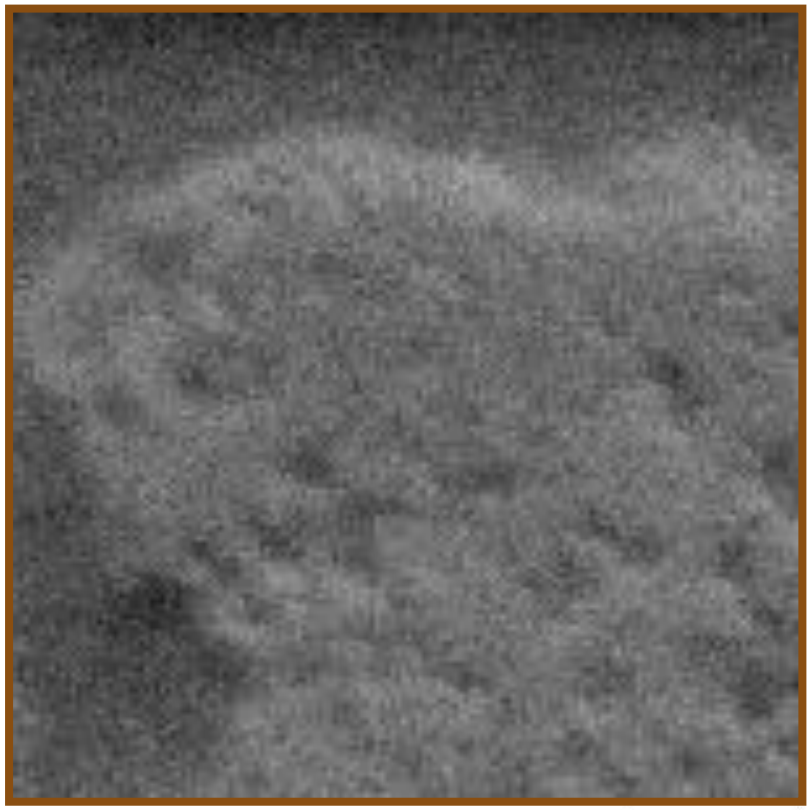
Convolutions and linear systems

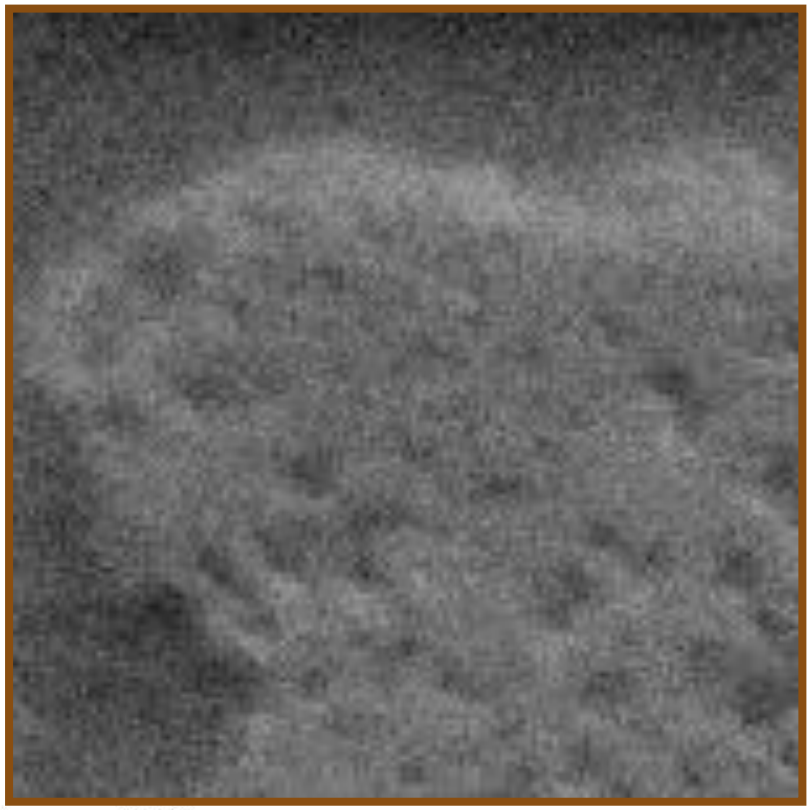


Any linear and translation invariant system can be represented as a convolution.

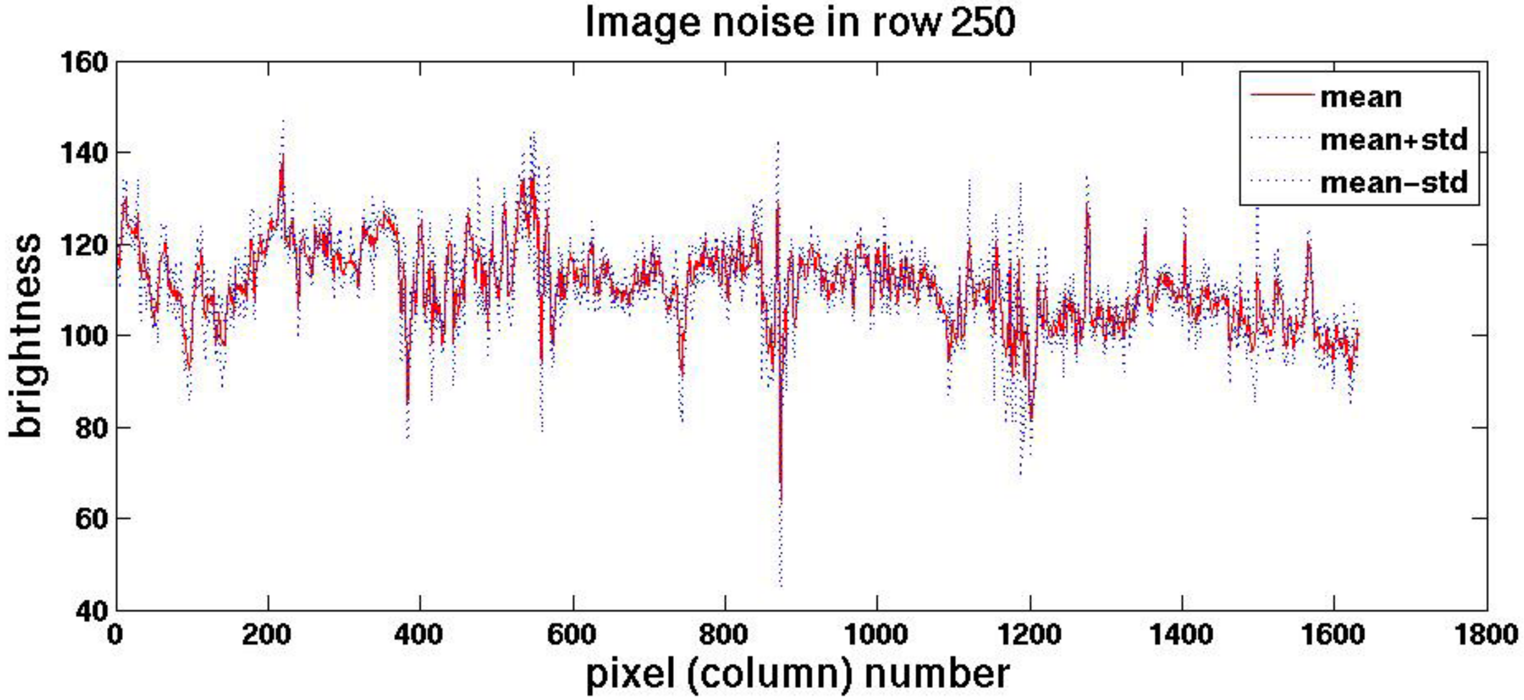
Motivation: noise reduction



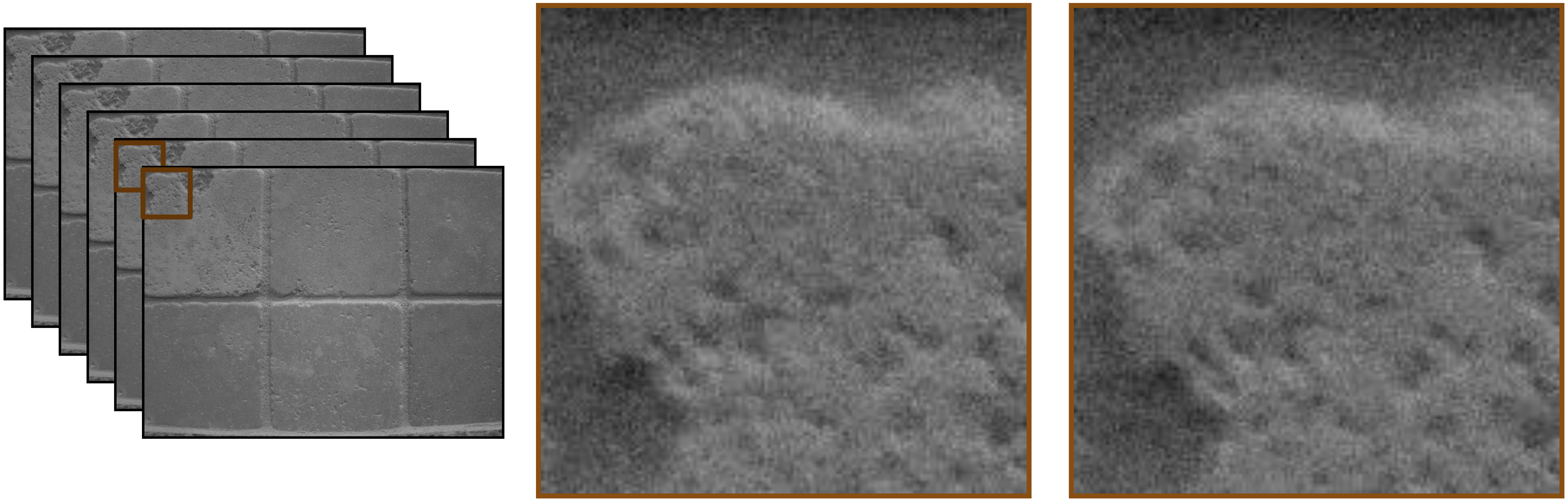




- We can measure **noise** in multiple images of the same static scene.
- How could we reduce the noise, i.e., give an estimate of the true intensities?



Motivation: noise reduction



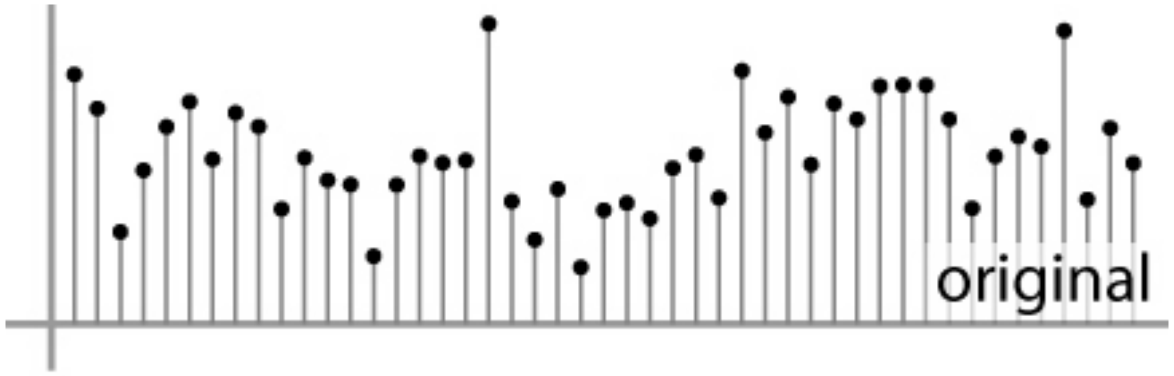
- How could we reduce the noise, i.e., give an estimate of the true intensities?
- **What if there's only one image?**

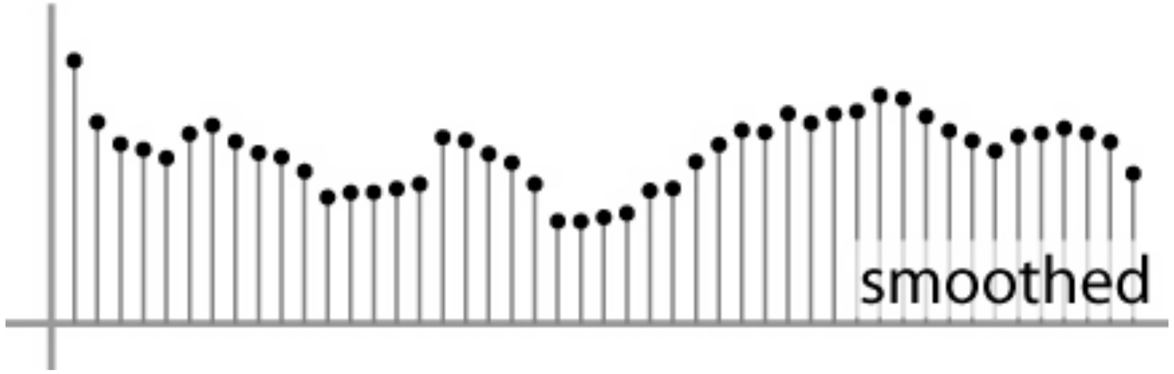
First attempt at a solution

- Let’s replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:

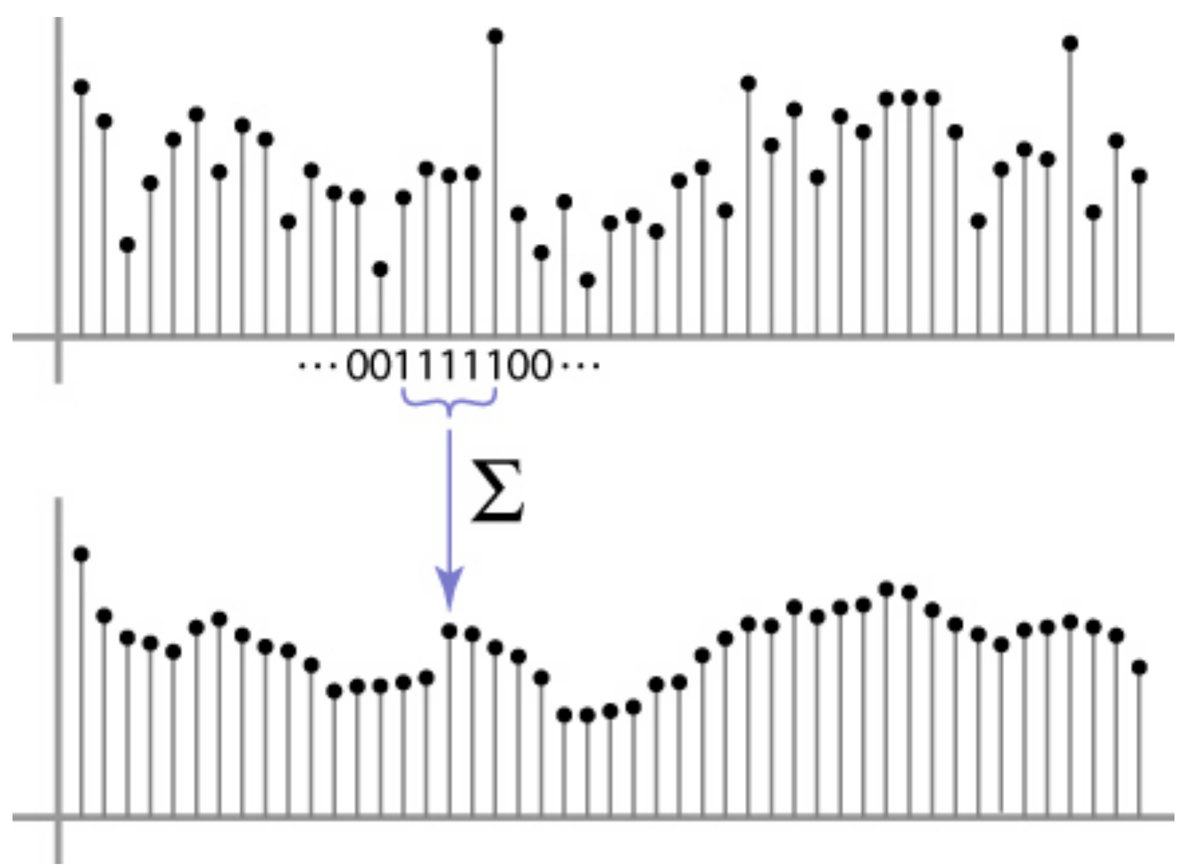




Source: S. Marschner

Weighted Moving Average

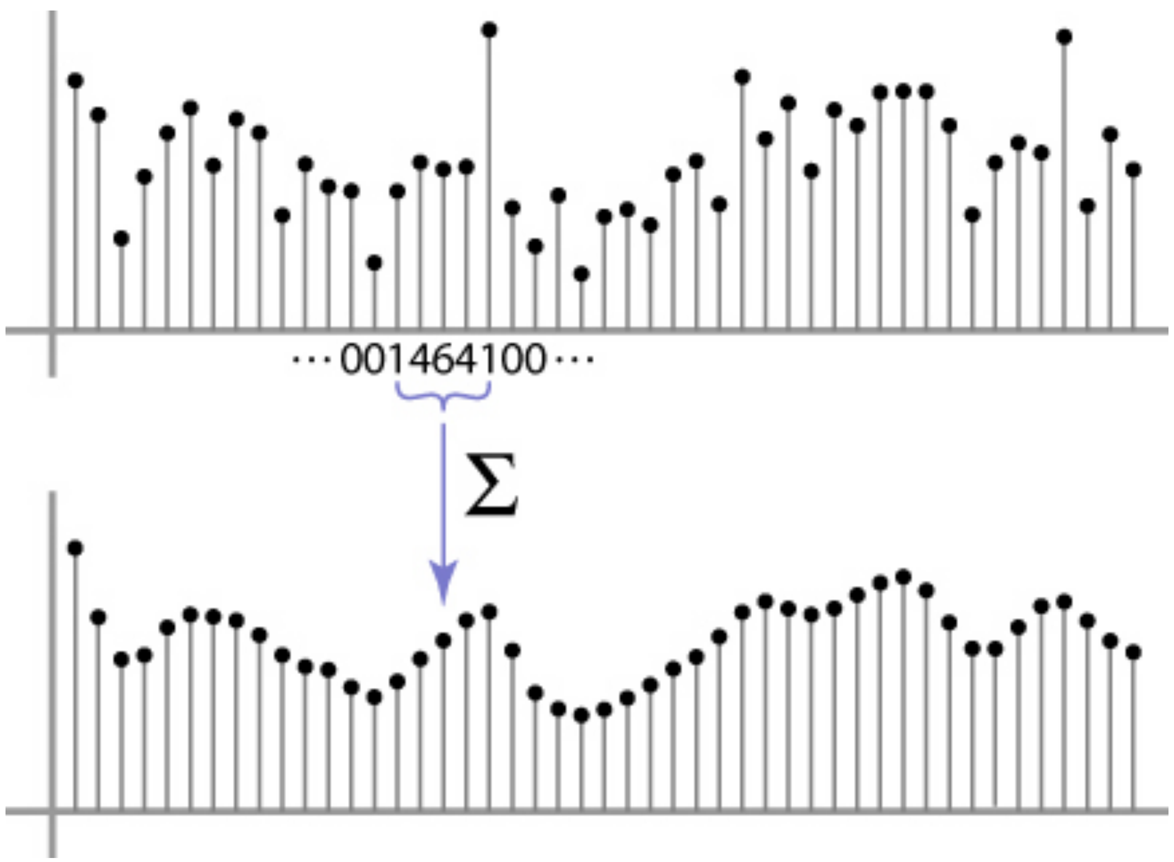
- Can add weights to our moving average
- *Weights* [1, 1, 1, 1, 1] / 5



Source: S. Marschner

Weighted Moving Average

- Non-uniform weights [1, 4, 6, 4, 1] / 16



Source: S. Marschner

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0								

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10							

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20						

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30				

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

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

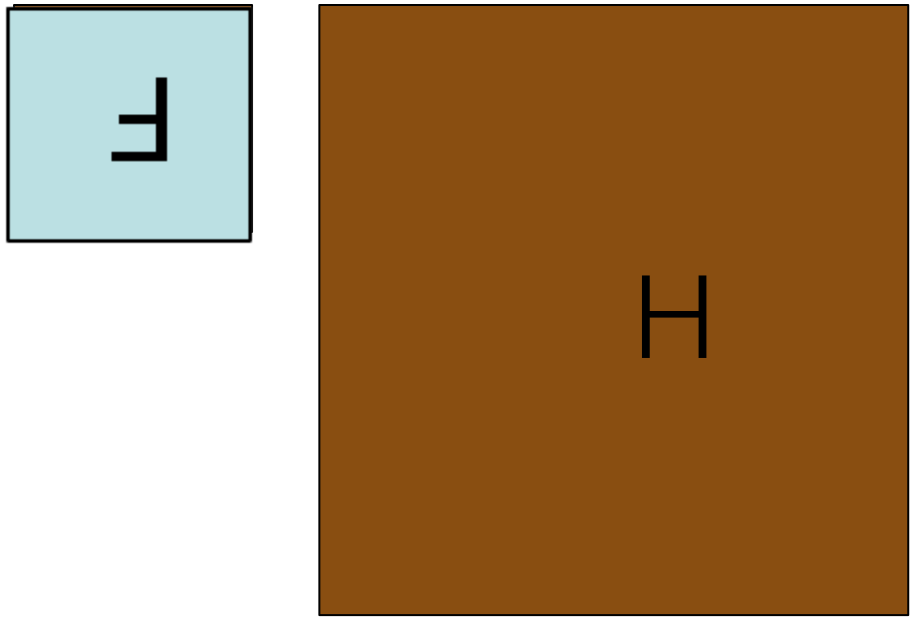
$G[x, y]$

	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	

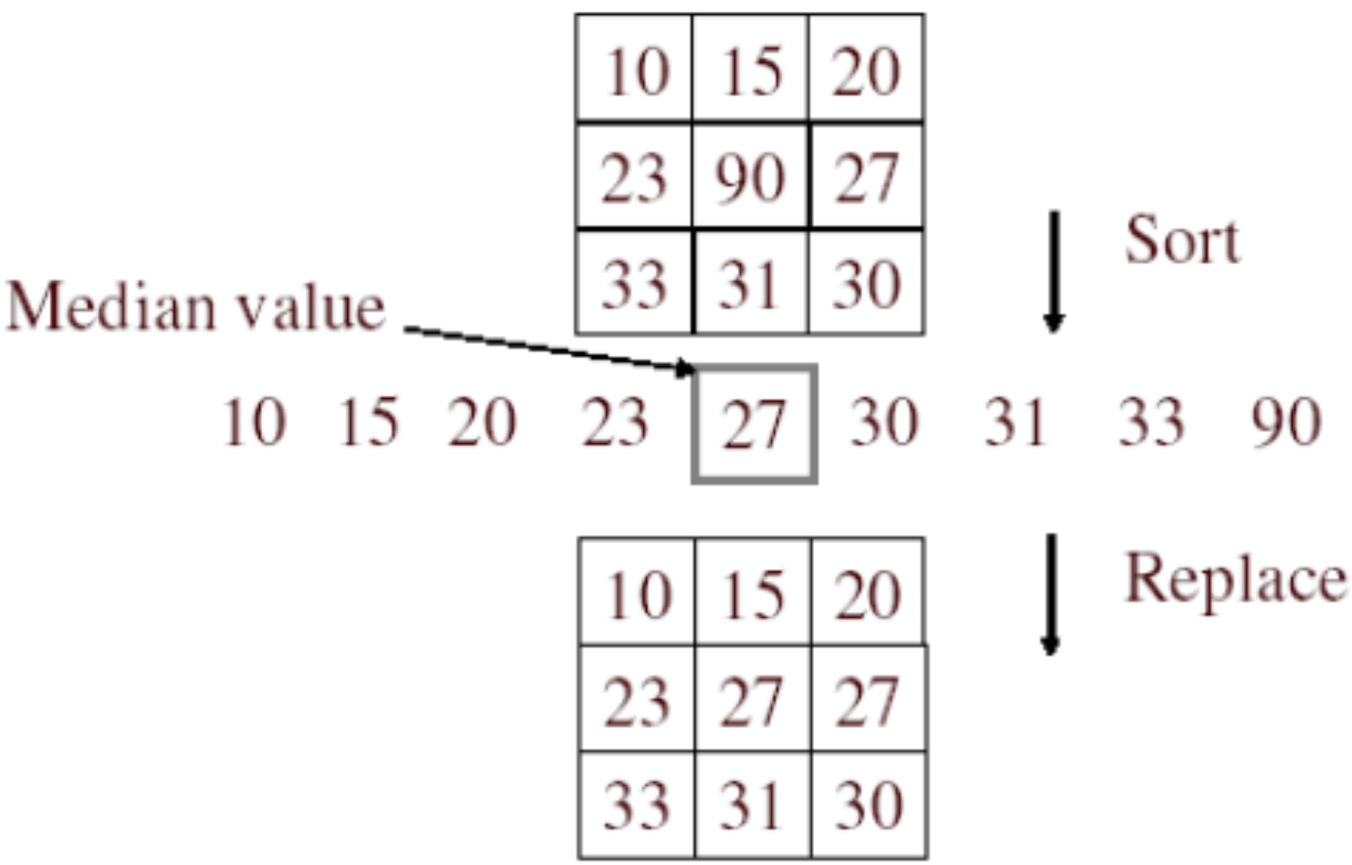
Source: S. Seitz

Convolution - repetition

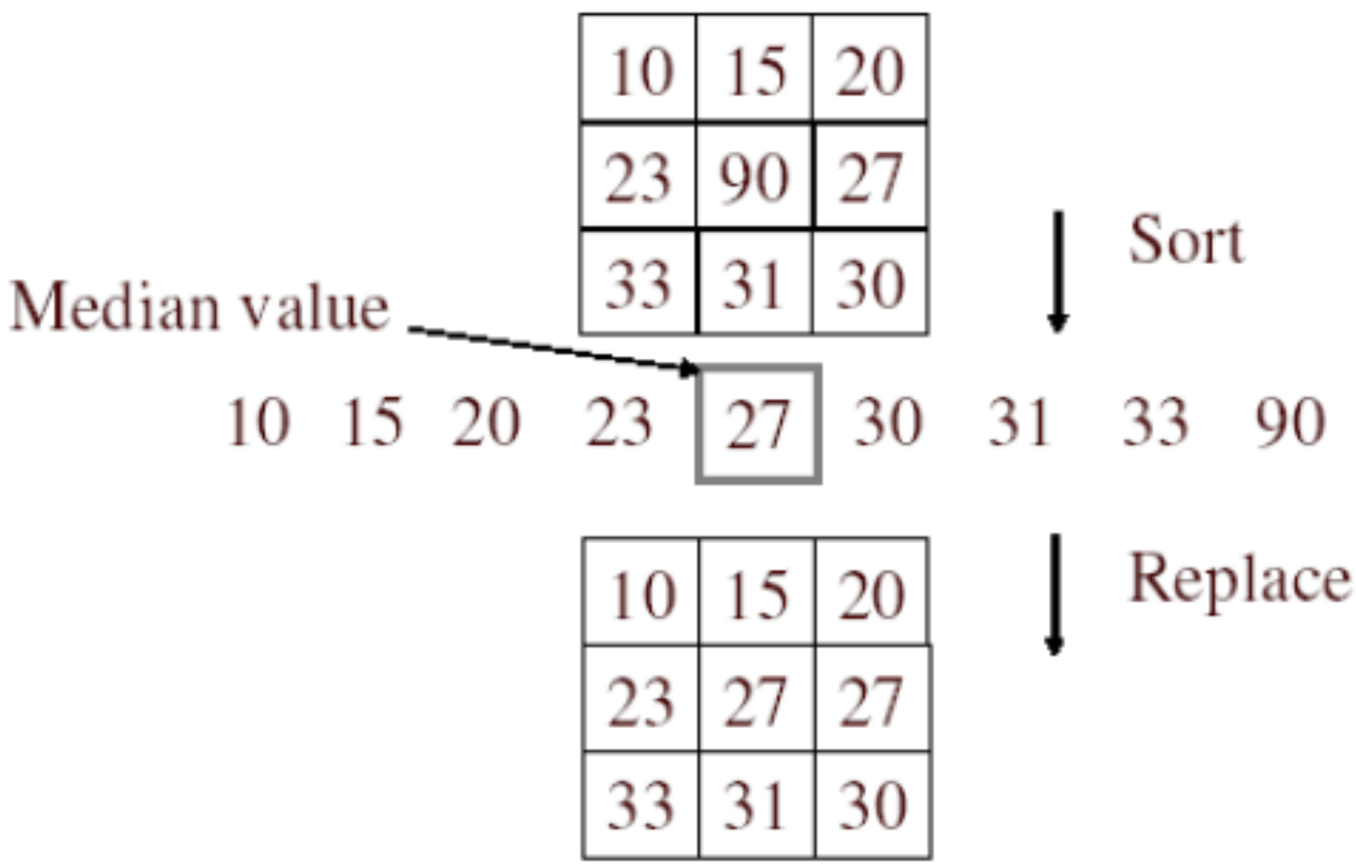
- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation
  - Produces scalar product of flipped filter at every position!



Not all filters can be written using convolutions!



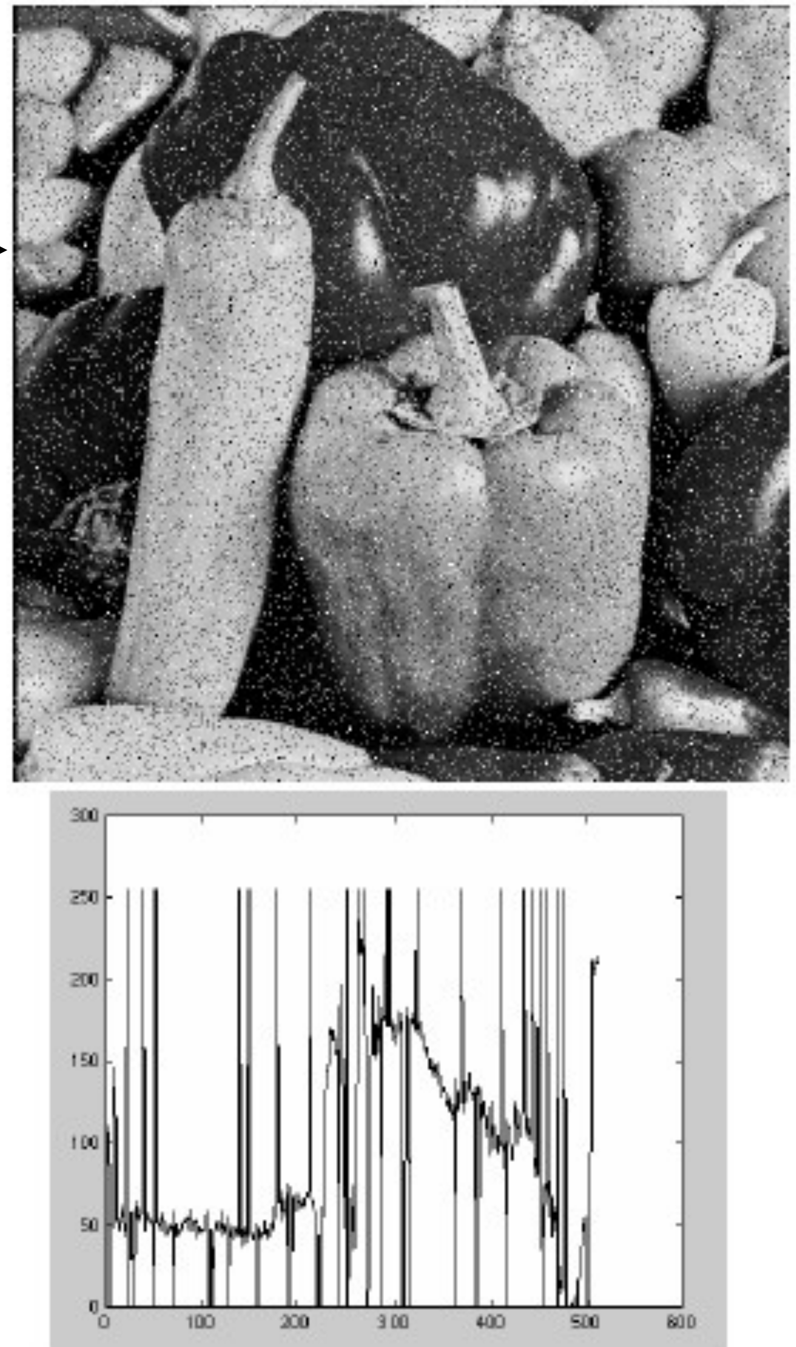
Median filter – an example of a non-linear sliding window smoother (Not a convolution)



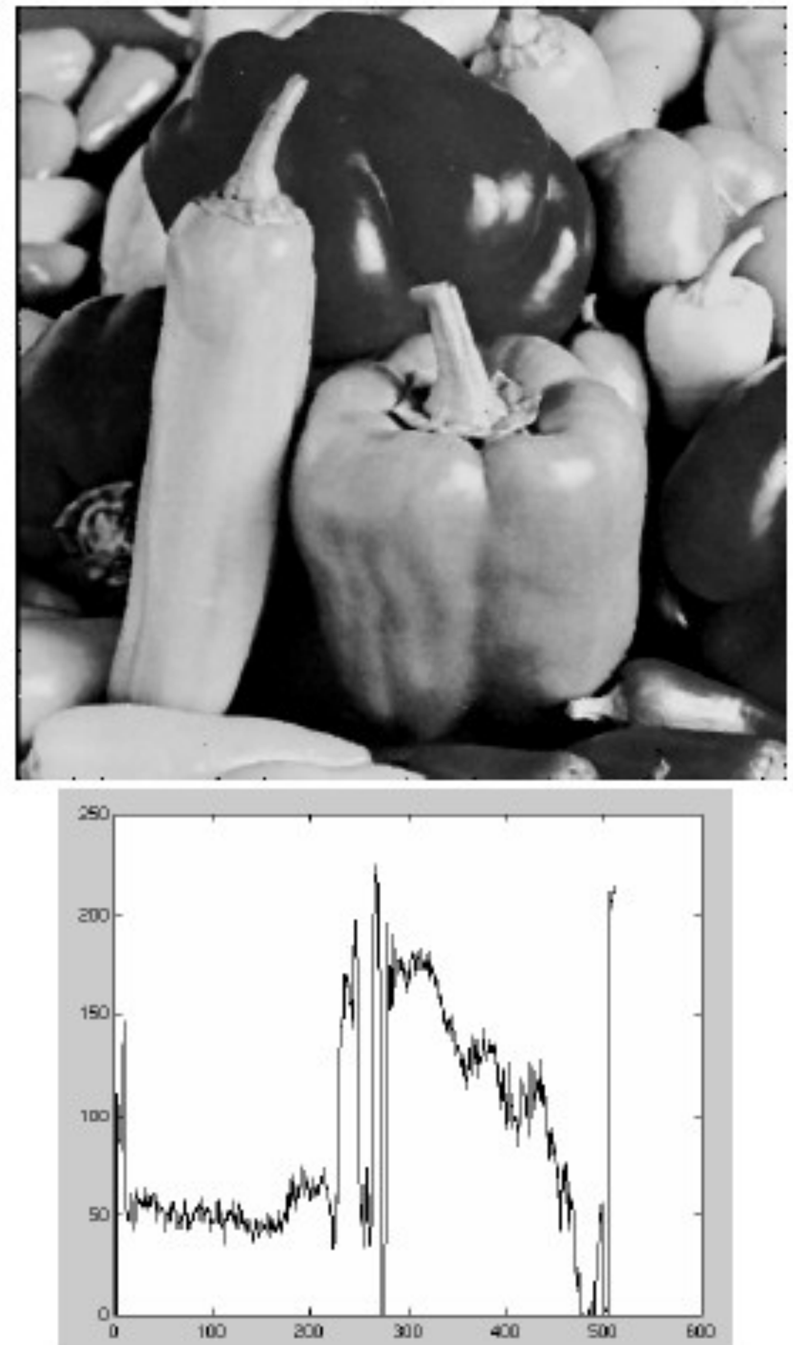
- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Not linear
- Not a convolution

Median filter

Salt and pepper noise



Median filtered

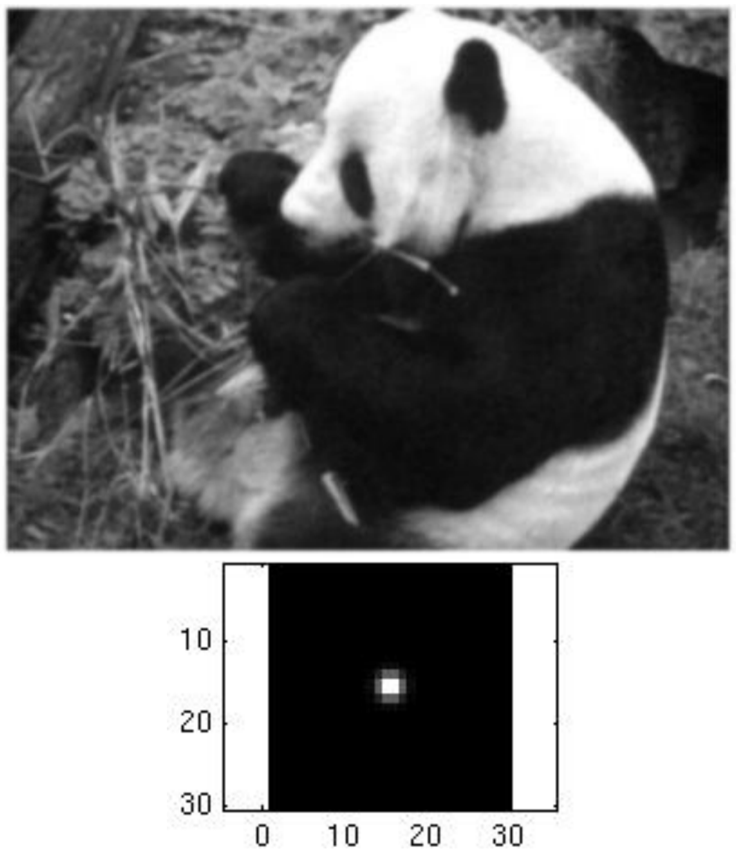


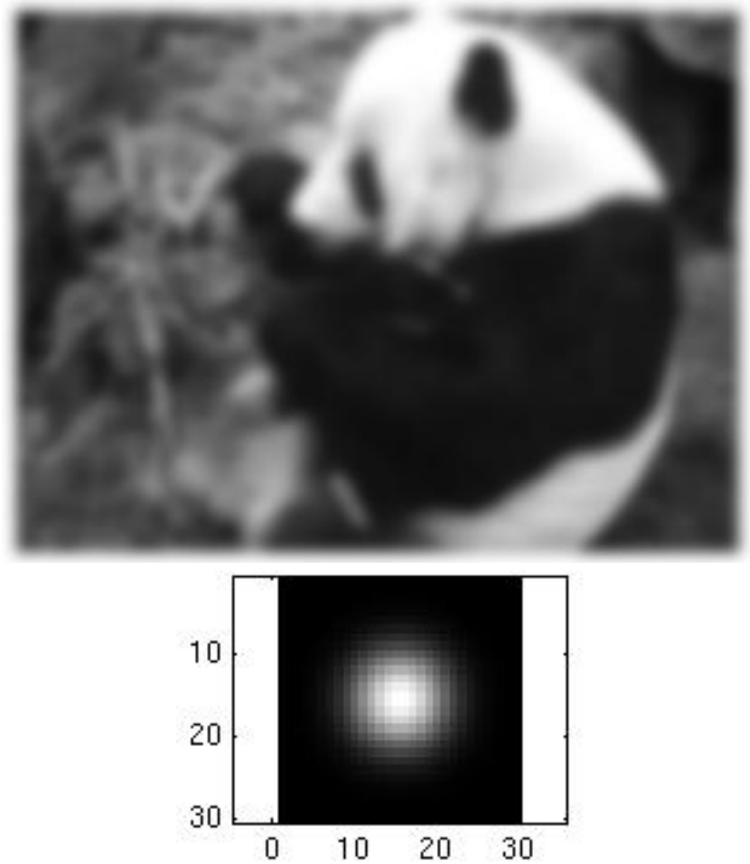
Plots of a row of the image

Source: M

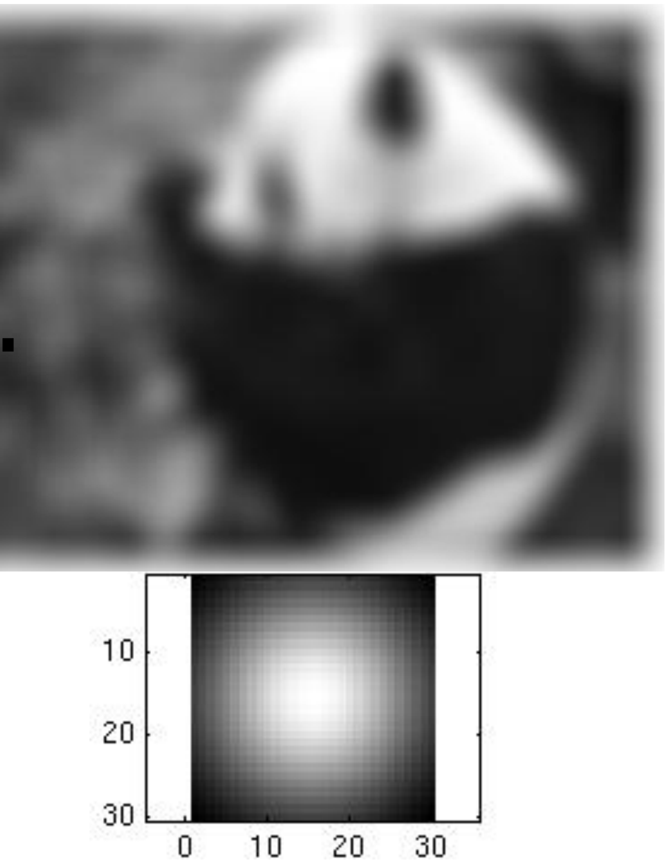
Smoothing with a Gaussian

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





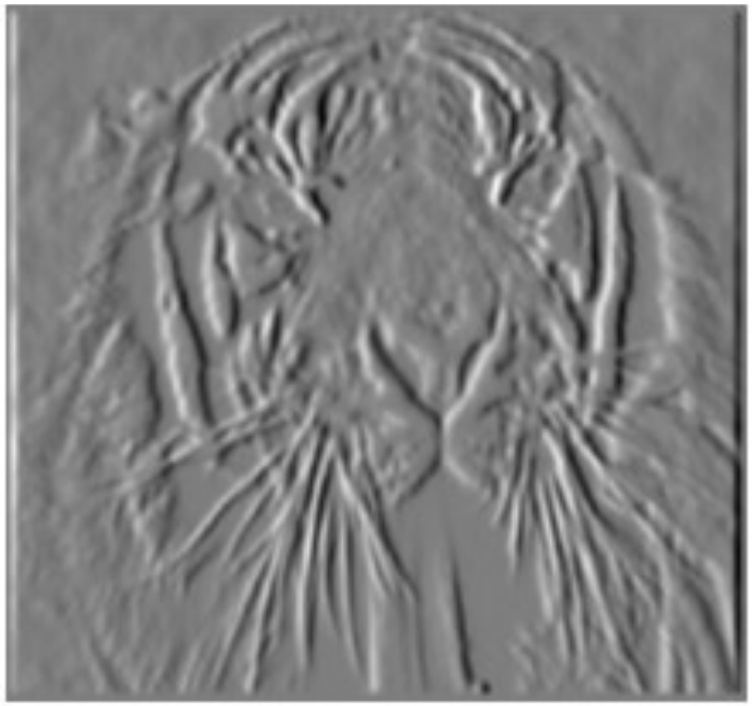
. . .

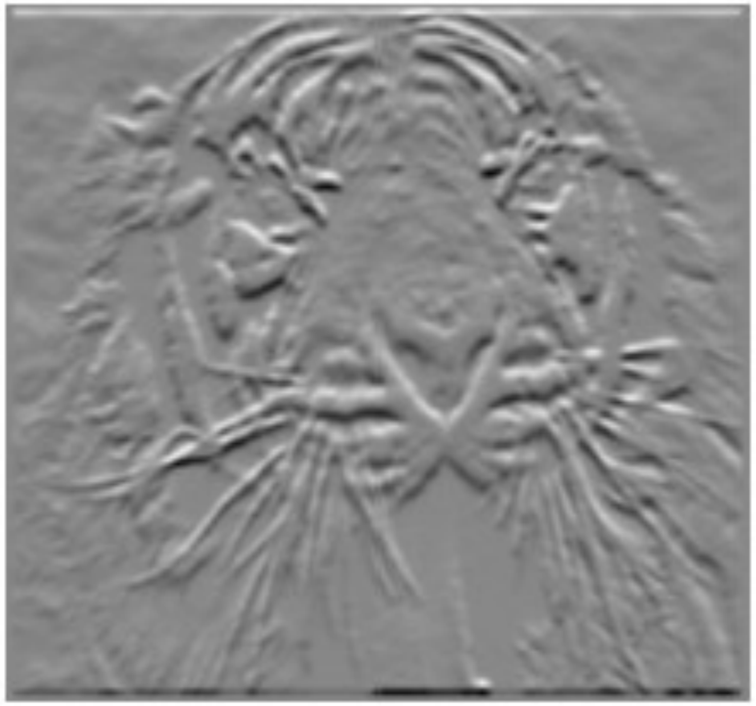



```
for sigma=1:3:10
  h = fspecial('gaussian`, fsize, sigma);
  out = imfilter(im, h);
  imshow(out);
  pause;
end
```

Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$





$$\frac{\partial f(x,y)}{\partial y}$$

-1	1

-1
1

? or

1
-1

Which shows changes with respect to x?

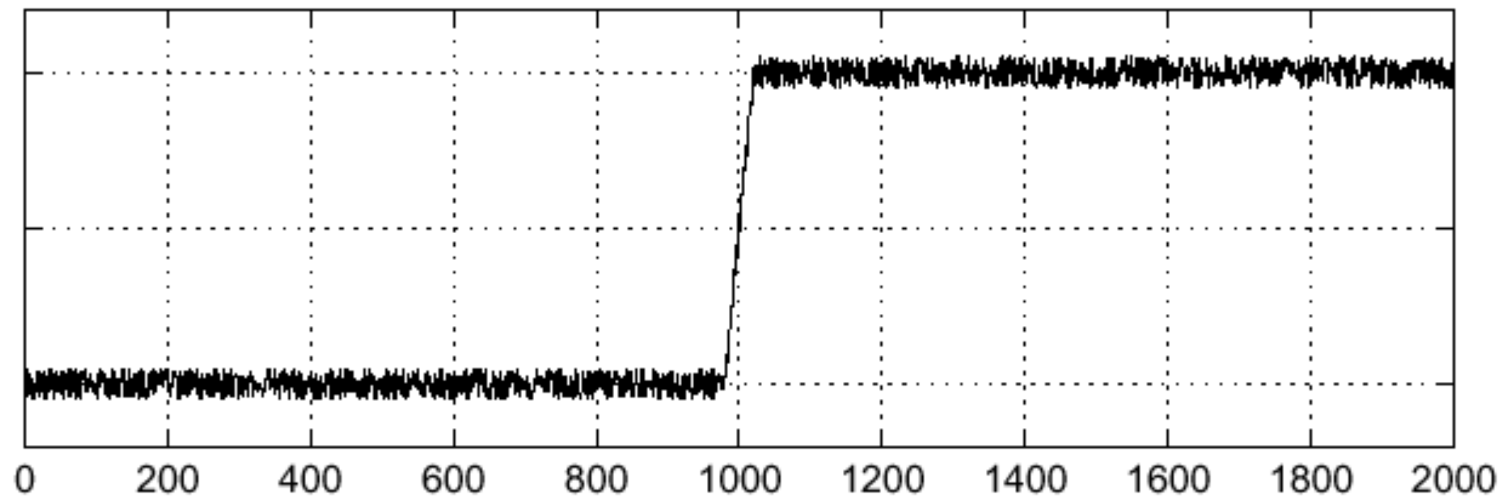
(showing flipped filters)

Effects of noise

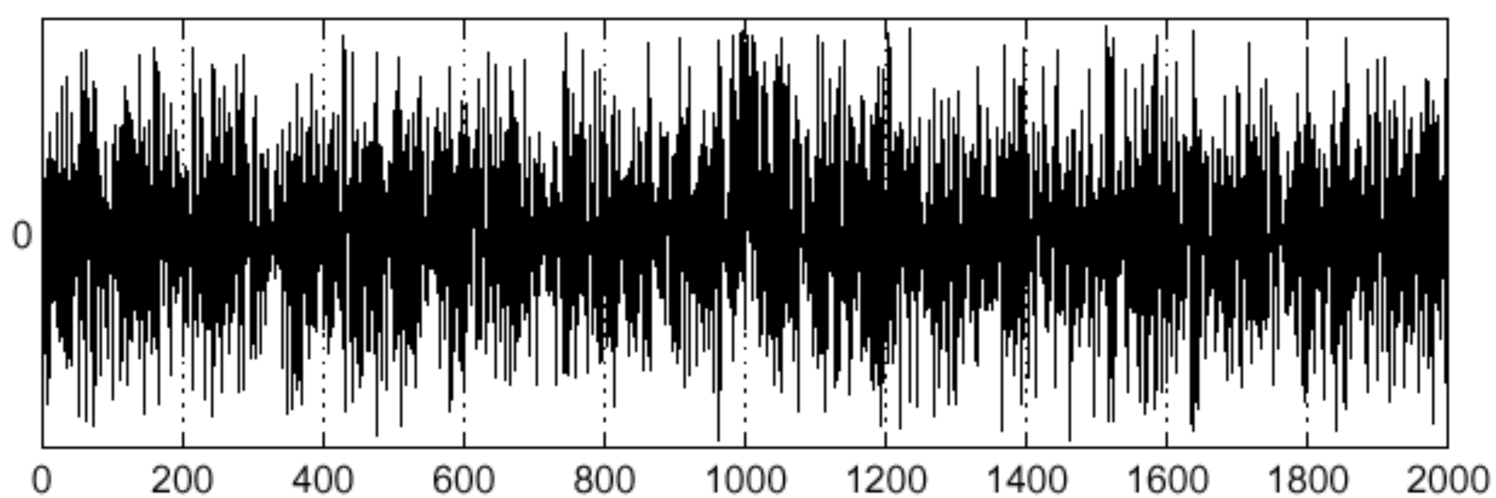
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$f(x)$



$\frac{d}{dx}f(x)$



Where is the edge?

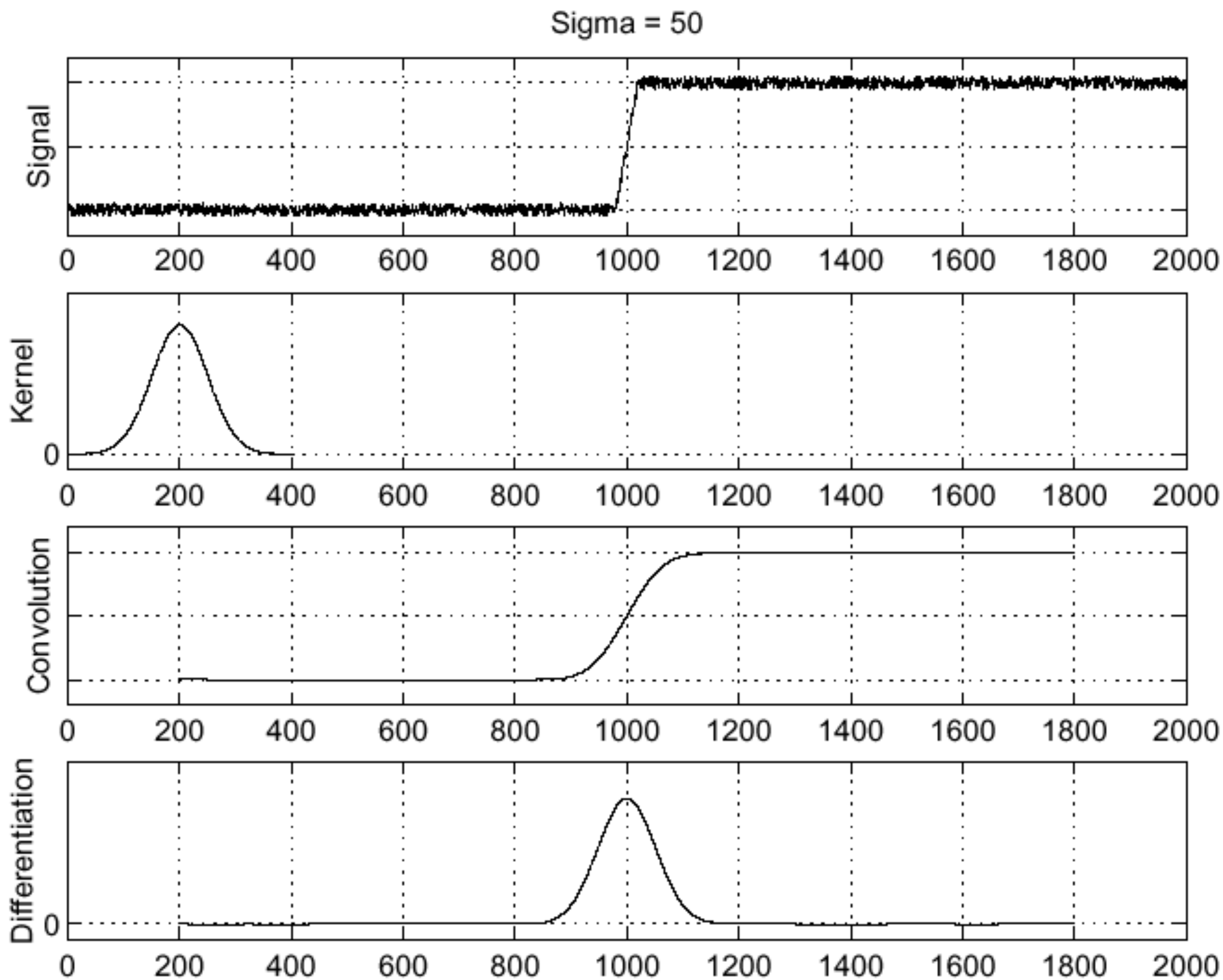
Solution: smooth first

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 property of convolution

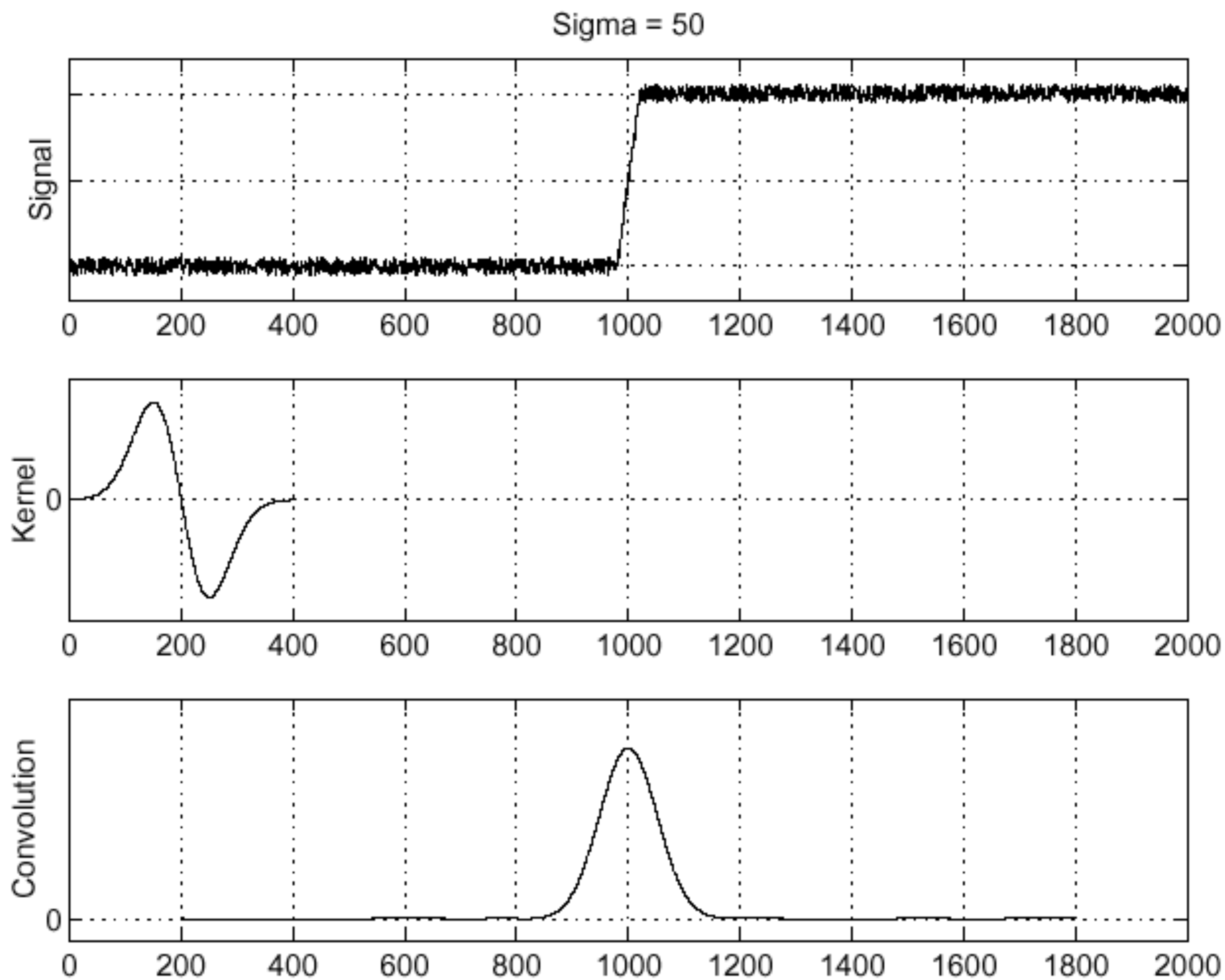
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution.

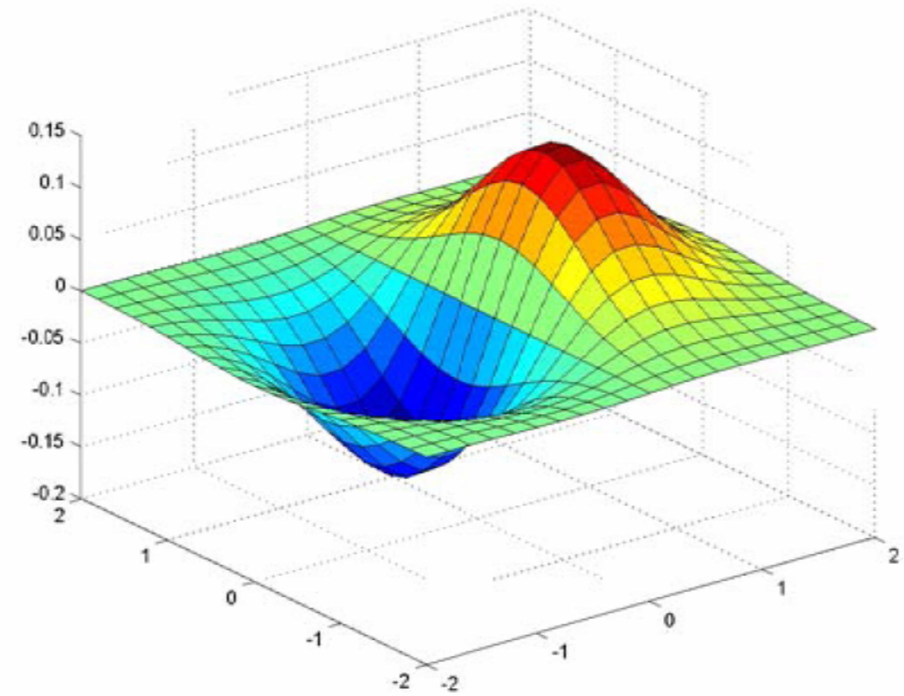
$$f$$

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

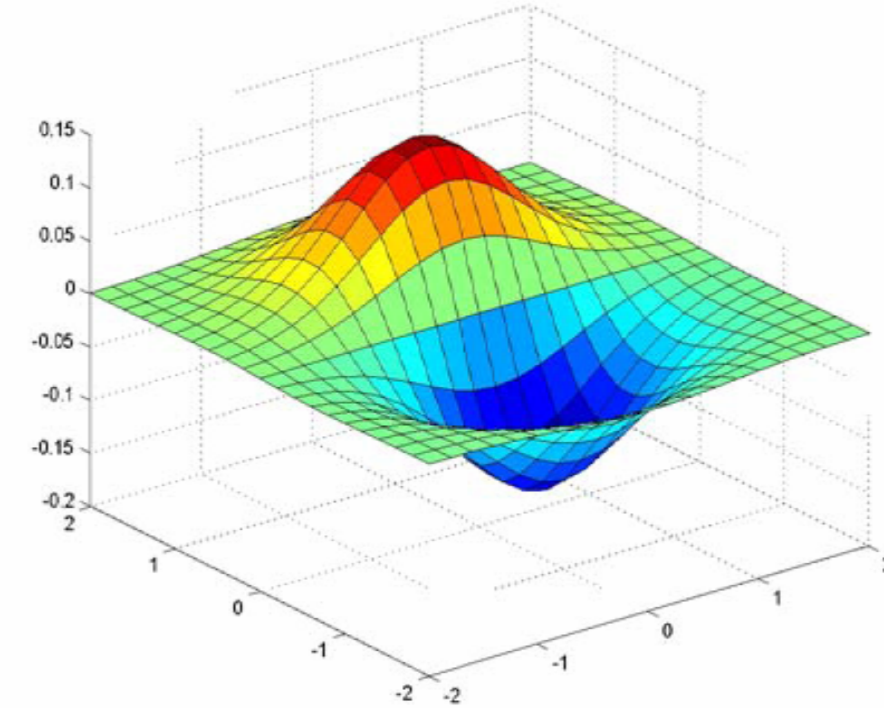
$$(\frac{\partial}{\partial x}h) \star f$$



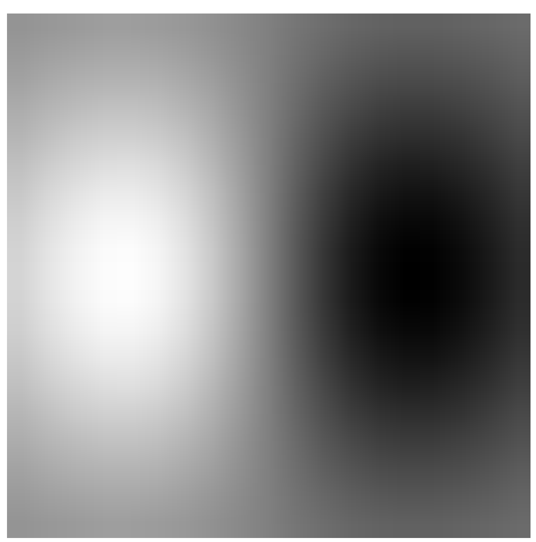
Derivative of Gaussian filters

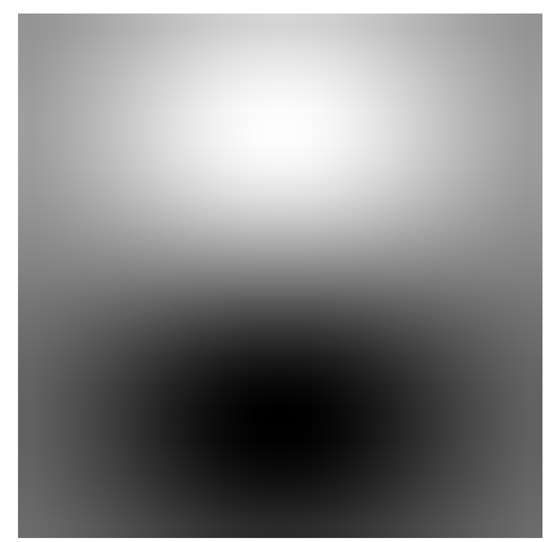


x-direction



y-direction





Source: L. Laz

Effect of σ on derivatives

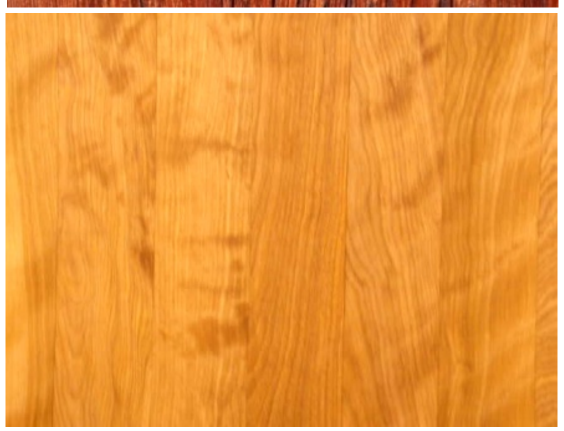
σ = 1 pixel σ = 3 pixels

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

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

So, what scale to choose?

It depends what we're looking for.



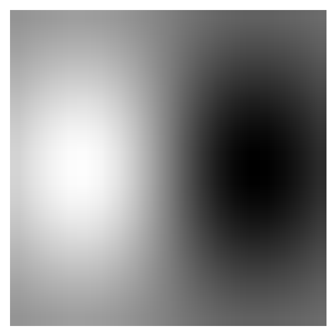
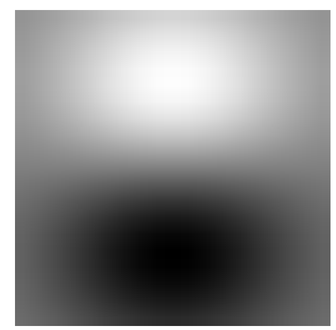
Too fine of a scale…can't see the forest for the trees.

Too coarse of a scale…can't tell the maple grain from the cherr

Template matching

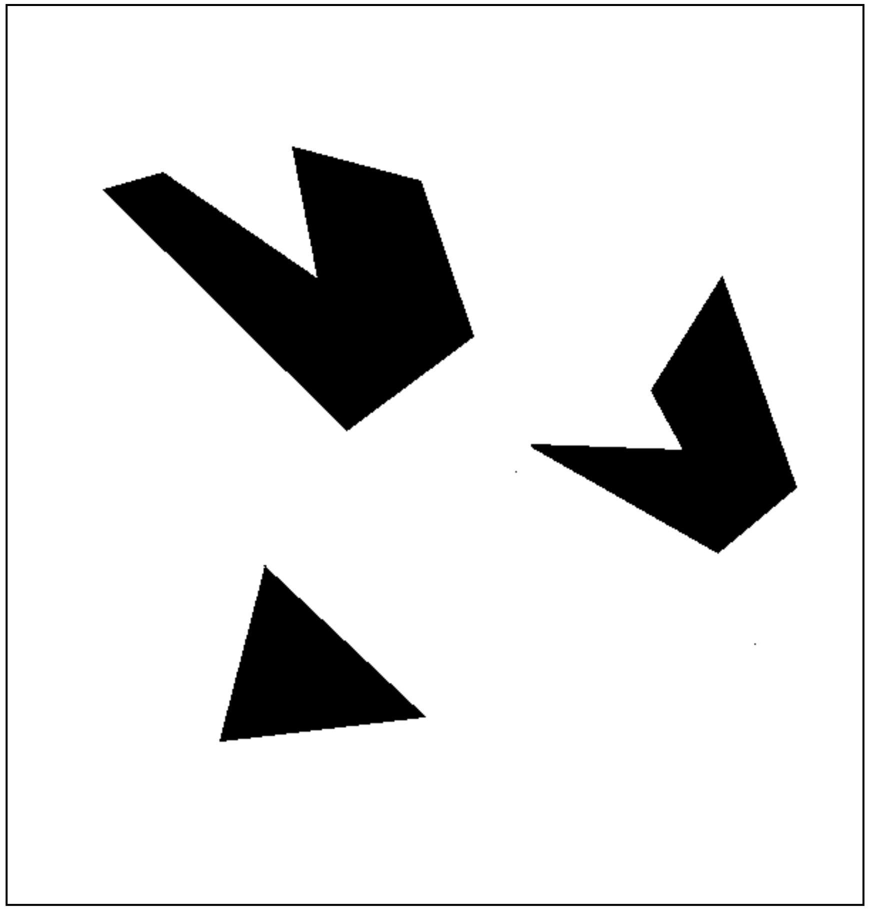
- Filters as **templates**:

  Note that filters look like the effects they are intended to find --- “matched filters”

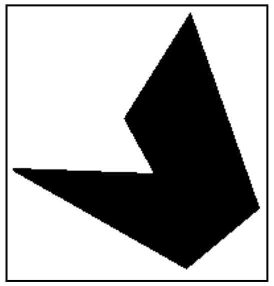



- Use normalized cross-correlation score to find a given pattern (template) in the image.
  - Szeliski Eq. 8.11
- Normalization needed to control for relative brightnesses.

Template matching



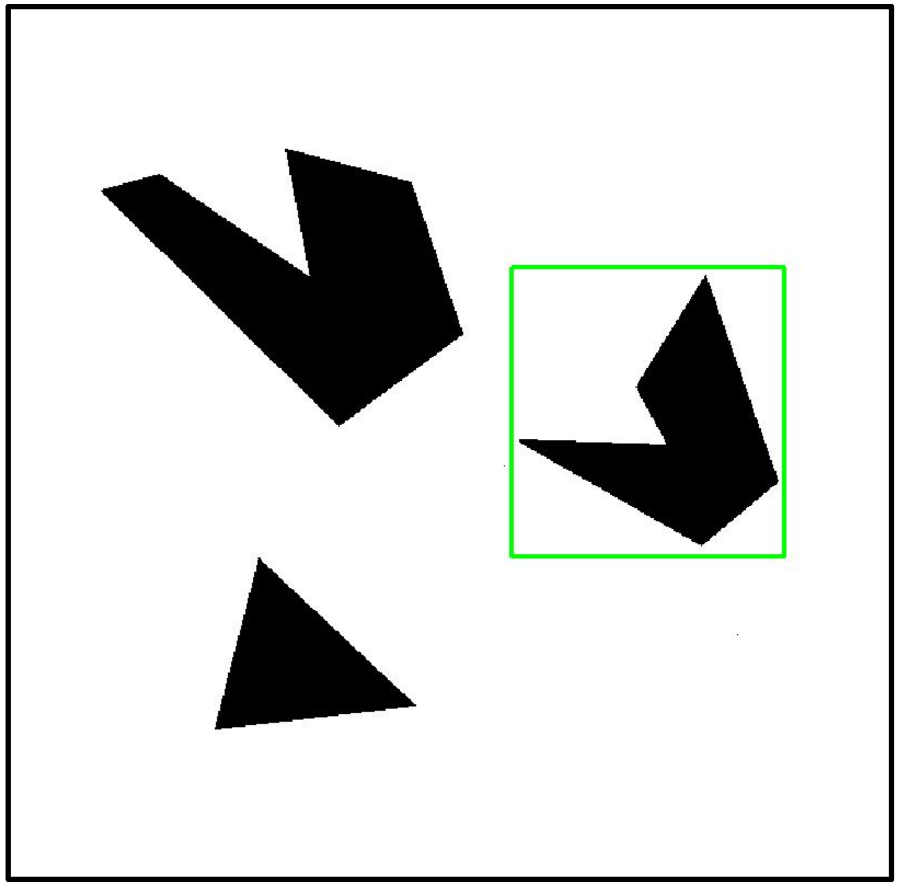
Scene



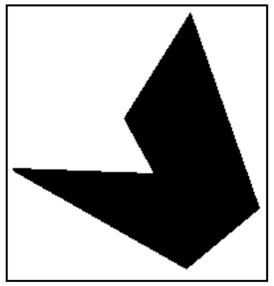
Template (mask)

A toy example

Template matching

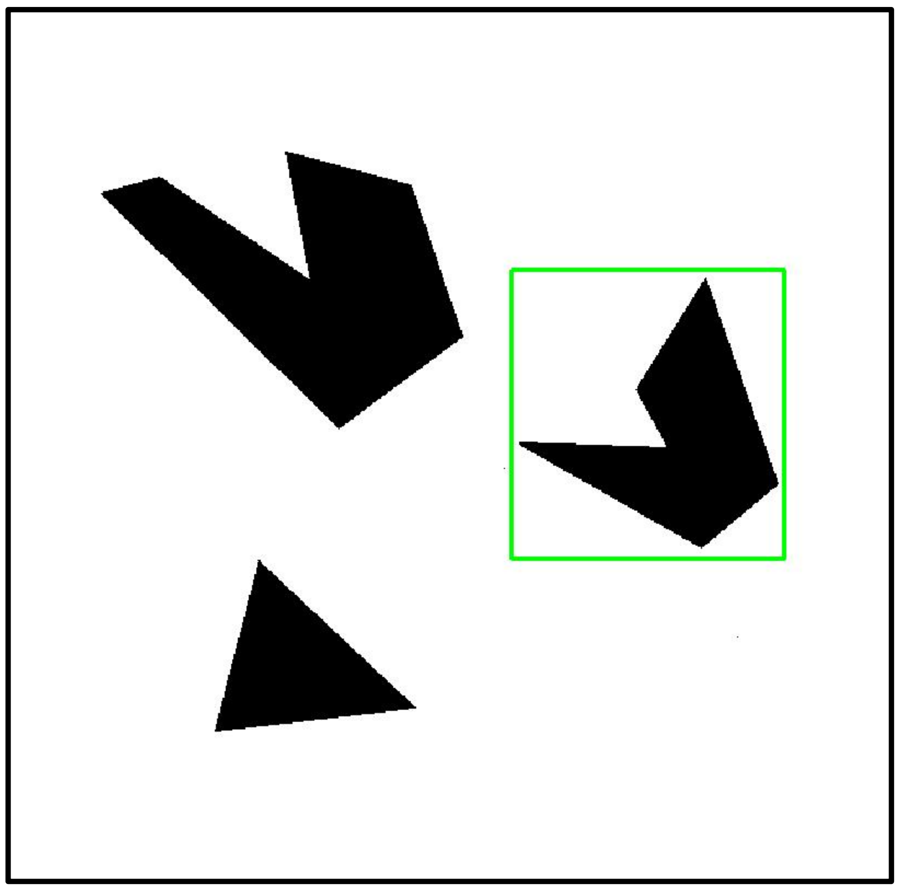


Detected template

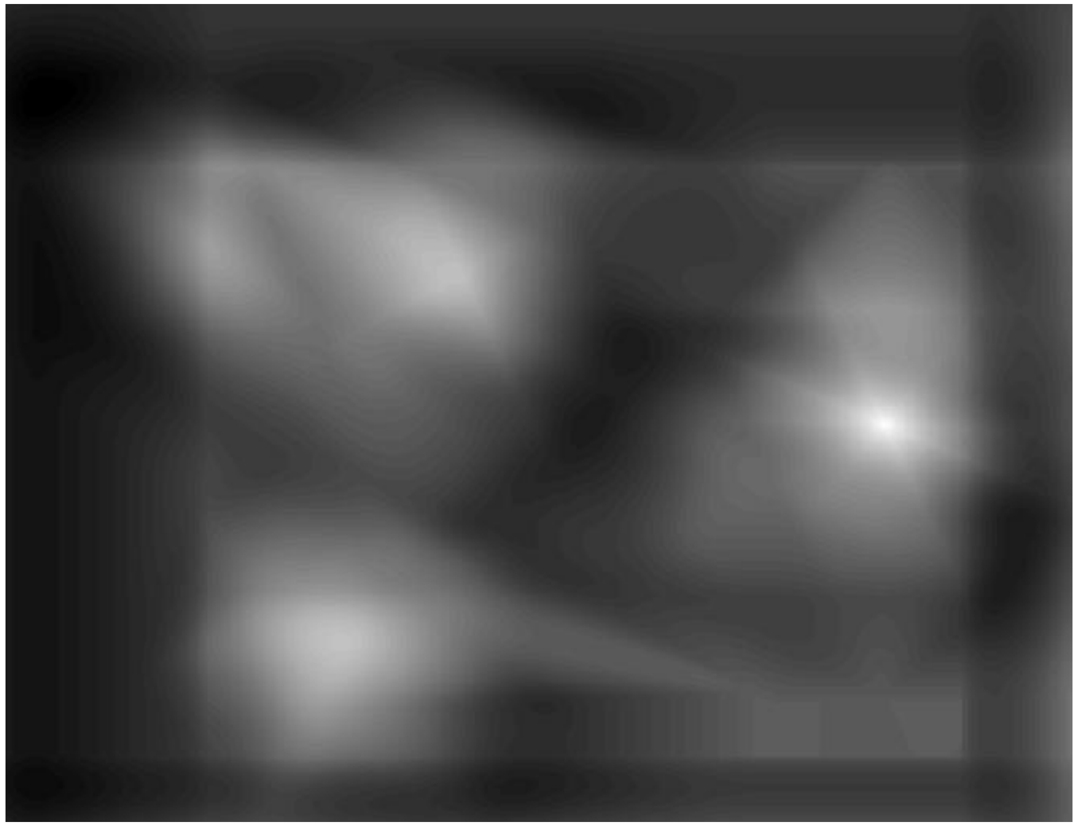


Template

Template matching



Detected template



Correlation map

Template matching

```
h =
        0       1       2
        1       4       1
        0       1       0
f =
        5       3       1       5       5
        5       2       0       1       2
        1       2       1       4       1
        1       5       0       1       0
        1       4       5       4       2
```

Template (mask)

Scene

f_{cut}

1	5	5
0	1	2
1	4	1

$$r = ||f_{cut} - h||^2$$

Template matching

$$r = ||f_{cut} - h||^2$$

$$r = (f_{cut} - h) \cdot (f_{cut} - h) = f_{cut} \cdot f_{cut} - 2f_{cut} \cdot h + h \cdot h$$

$$f_{cut}^2 \cdot e$$

$$(f.^2) * e - 2f * \hat{h} + h \cdot h$$

Template matching

$$(f.^2) * e - 2f * \hat{h} + h \cdot h$$

```
h =
     0     1     2
     1     4     1
     0     1     0
f =
     5     3     1     5     5
     5     2     0     1     2
     1     2     1     4     1
     1     5     0     1     0
     1     4     5     4     2
```

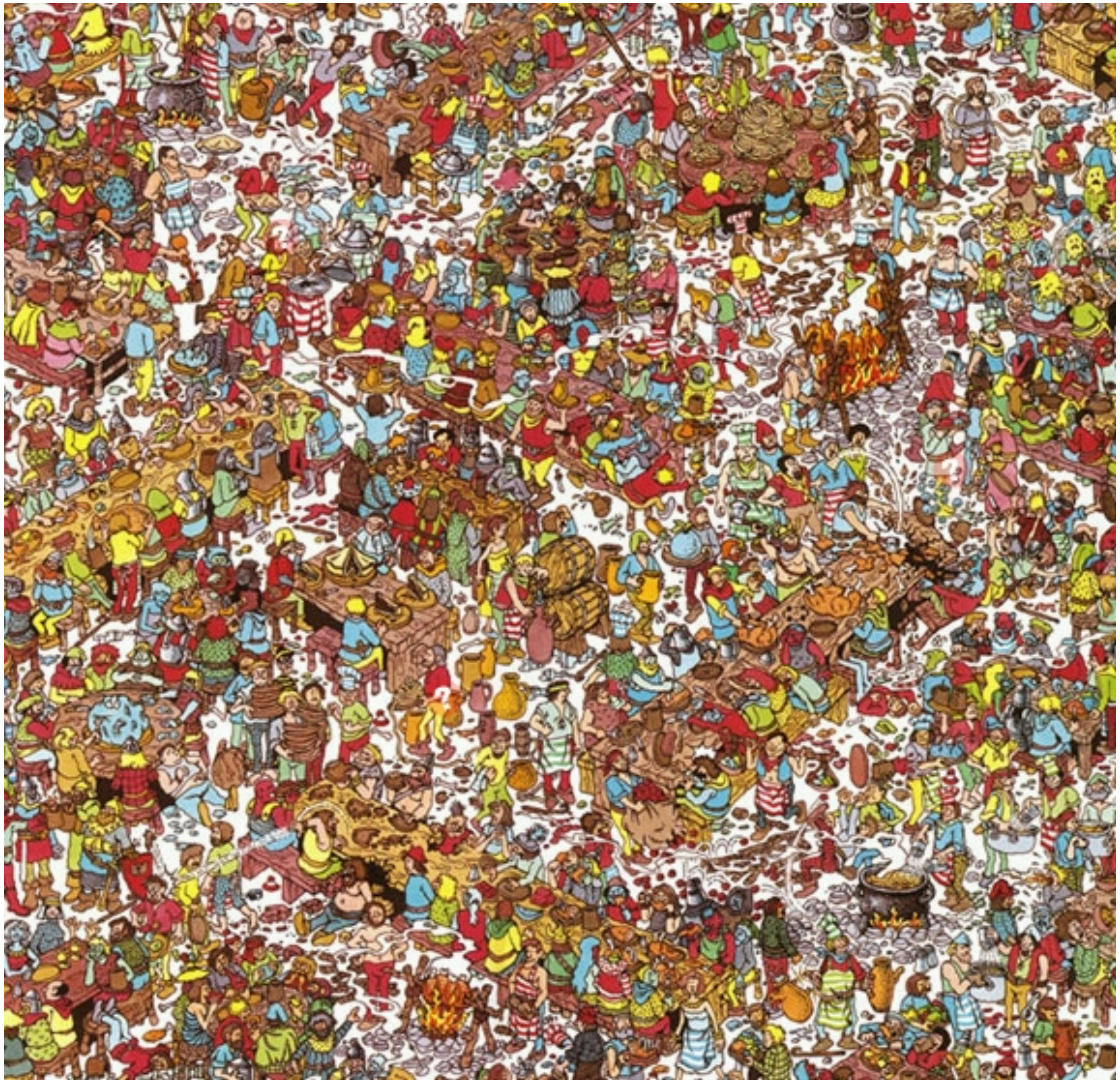
```
e = ones(size(h))
hnorm2 = norm(h,'fro')^2;
hhat = flipud(fliplr(h))
```

```
e =
     1     1     1
     1     1     1
     1     1     1
hhat =
     0     1     0
     1     4     1
     2     1     0
```

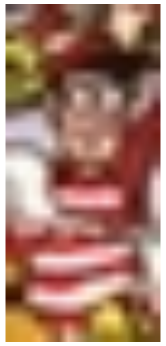
```
r = conv2(f.^2,e,'same') - 2*conv2(f,hhat,'same') + hnorm2
```

```
r =
   31.0000   48.0000   40.0000   26.0000   25.0000
   24.0000   54.0000   55.0000   48.0000   66.0000
   52.0000   51.0000   52.0000   -0.0000   27.0000
   42.0000   40.0000   88.0000   60.0000   54.0000
   29.0000   38.0000   47.0000   22.0000   21.0000
```

Where's Waldo?

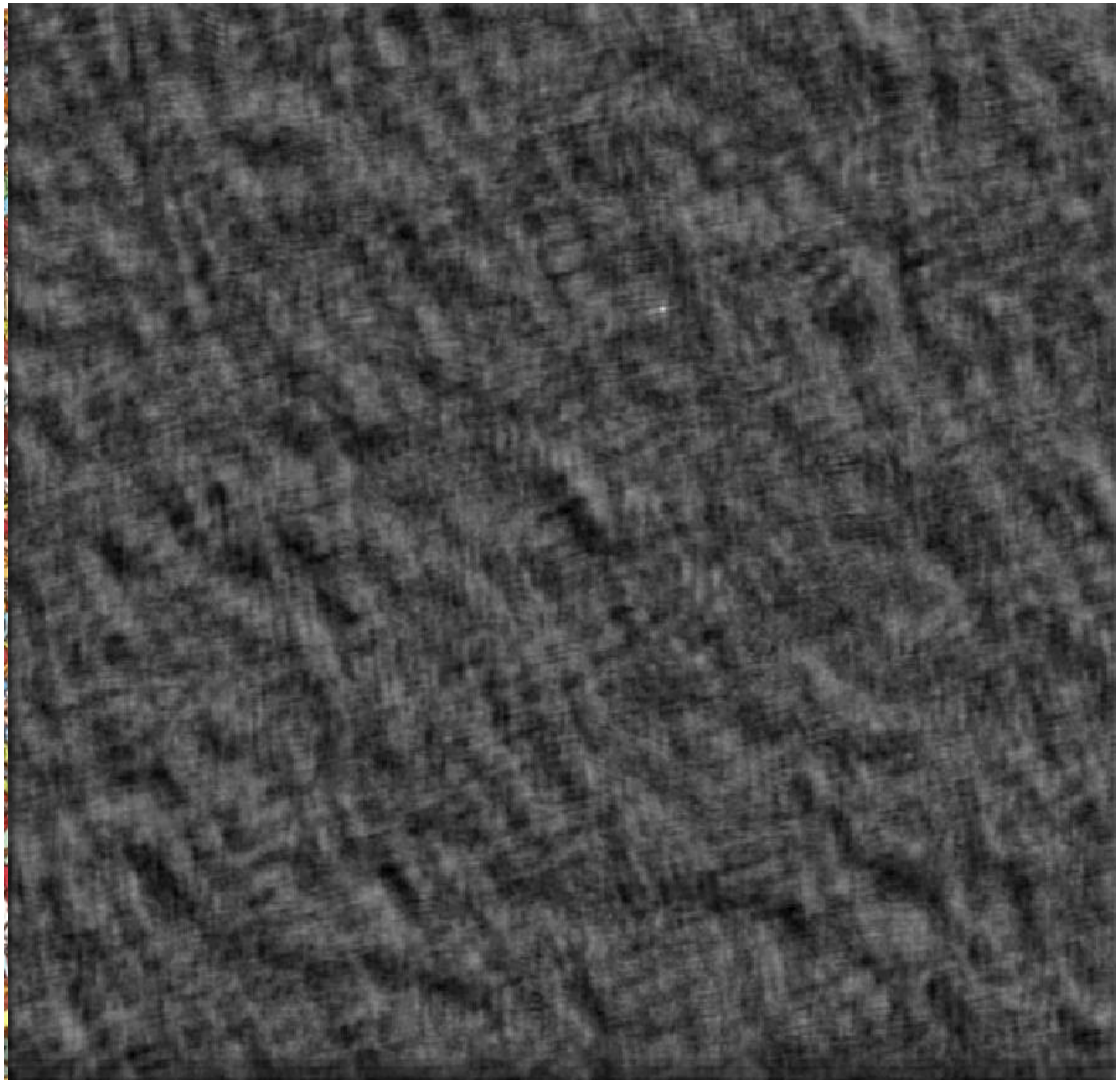


Scene

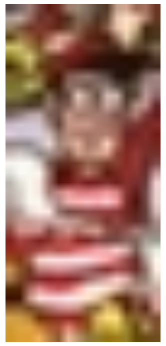


Template

Where's Waldo?



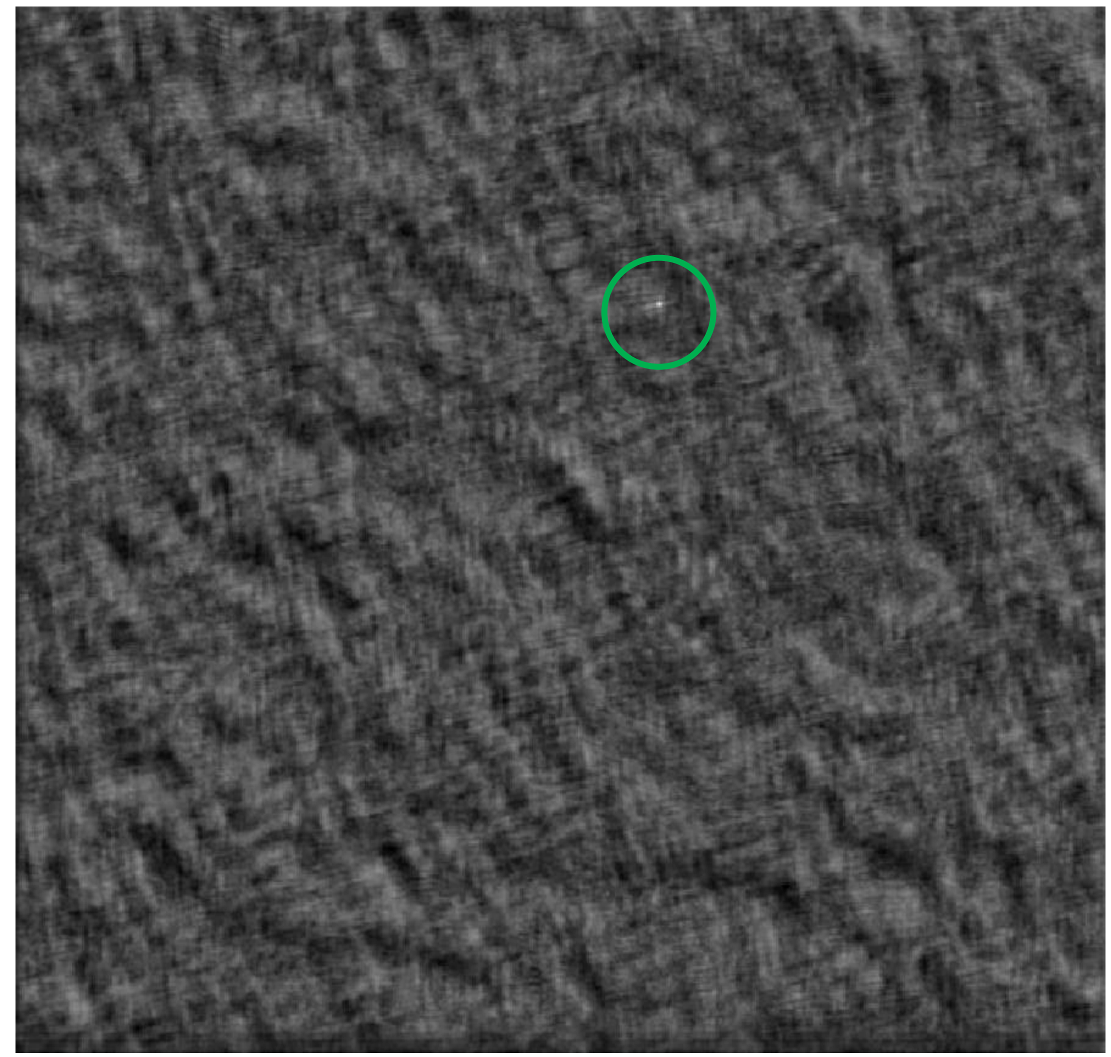
Scene



Template

Where's Waldo?

Detected template



Correlation map

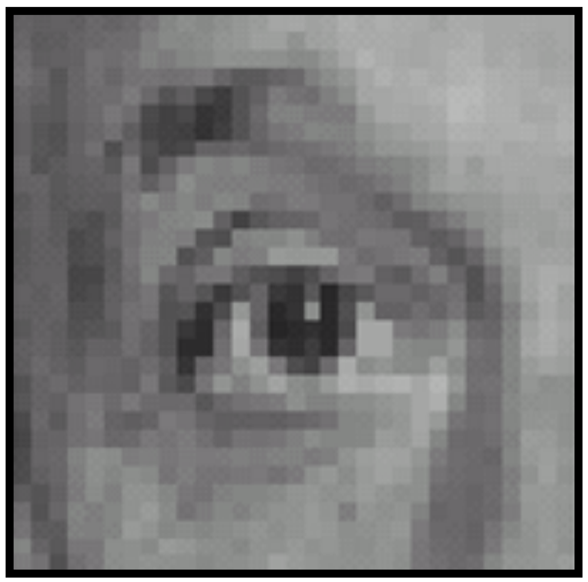
Template matching

Scene

Template

What if the template is not identical to some subimage in the scene?

Practice with linear filters



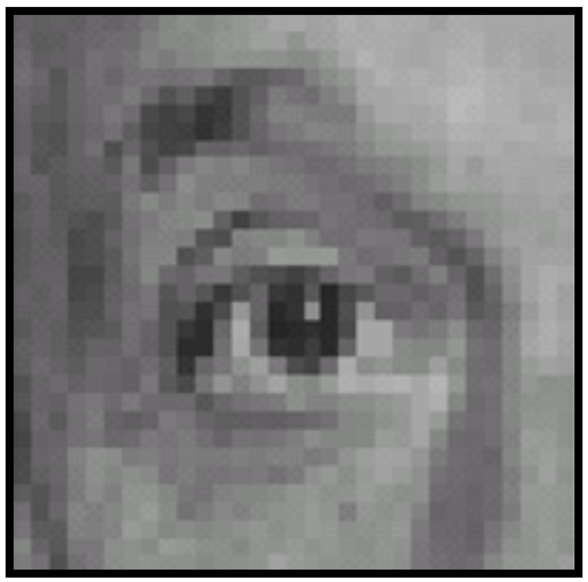
Original

0	0	0
0	1	0
0	0	0

?

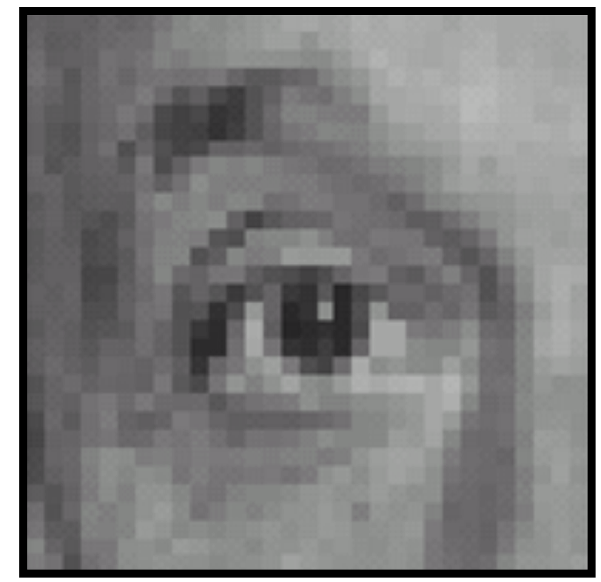
Source: D. Lowe

Practice with linear filters



Original

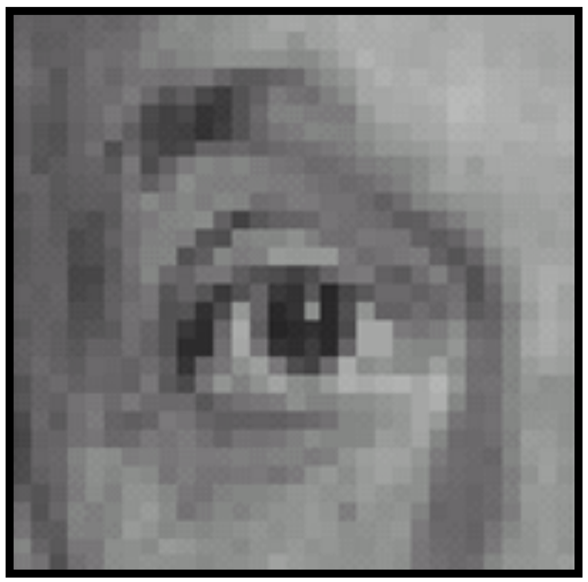
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

Practice with linear filters



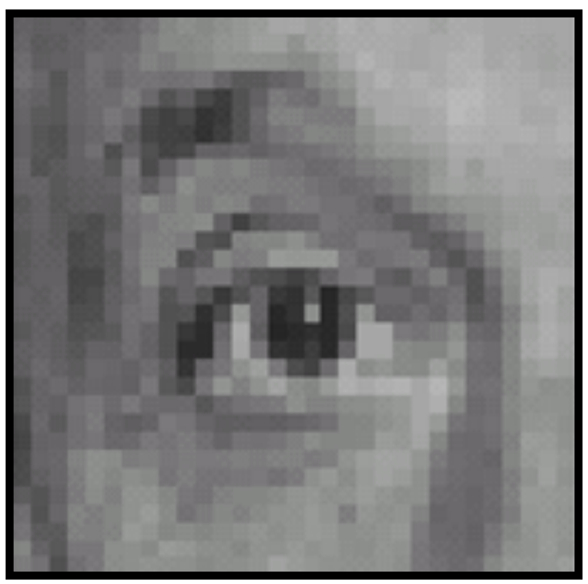
Original

0	0	0
0	0	1
0	0	0

?

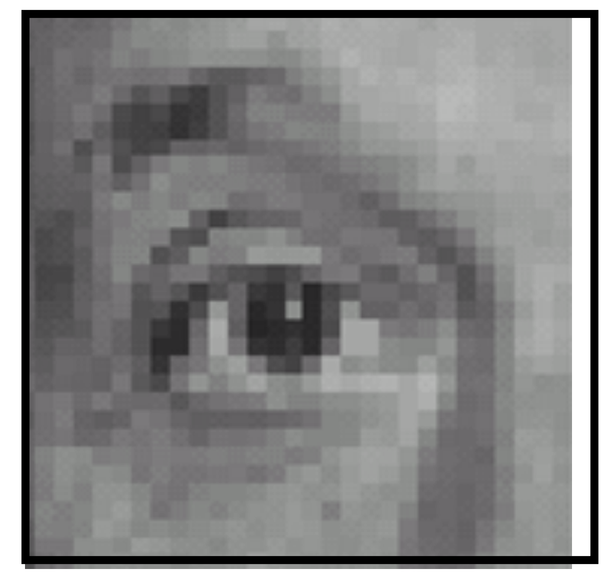
Source: D. Lowe

Practice with linear filters



Original

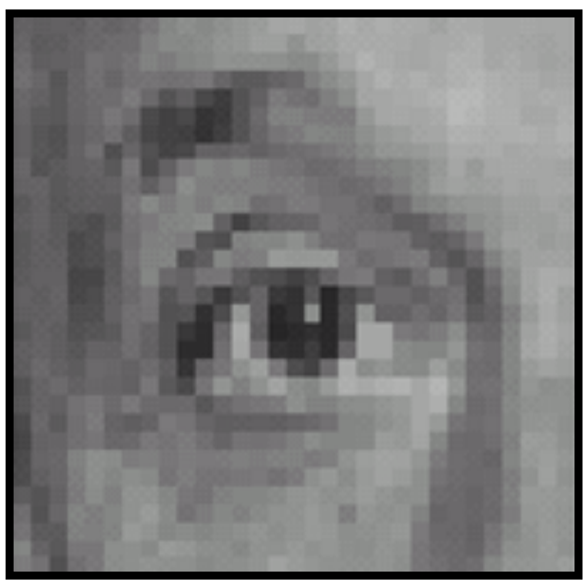
0	0	0
0	0	1
0	0	0



Shifted left
by 1 pixel with
correlation

Source: D. Lowe

Practice with linear filters



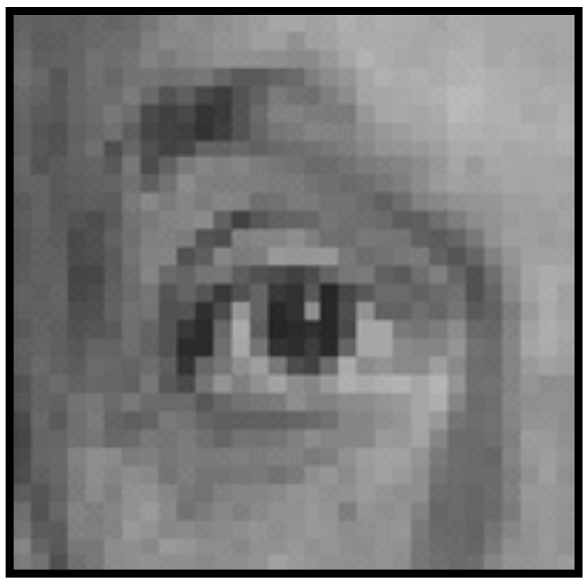
Original

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

?

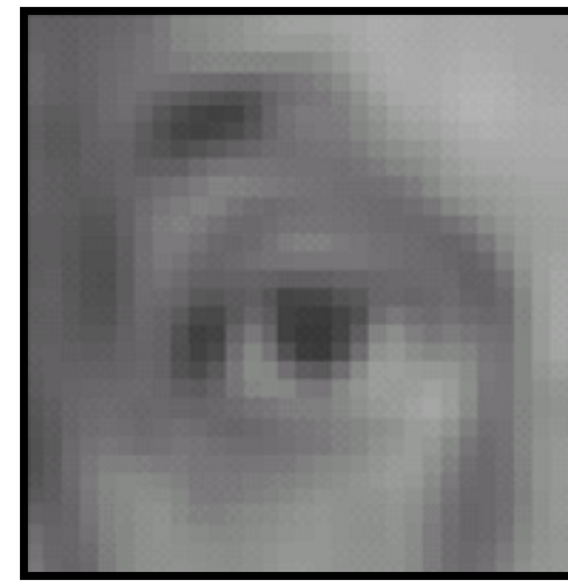
Source: D. Lowe

Practice with linear filters



Original

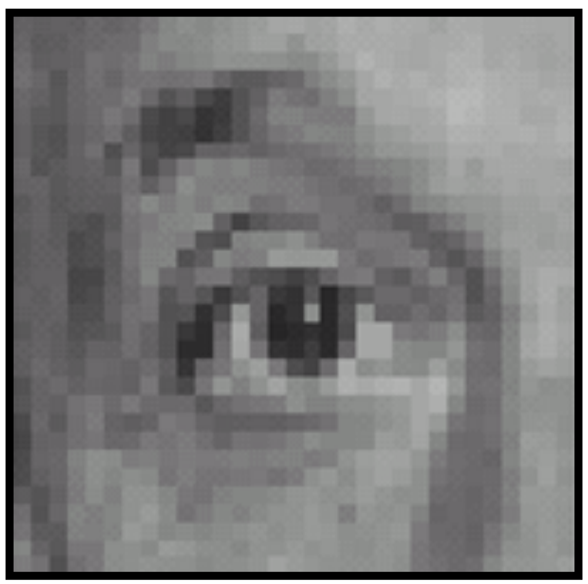
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters



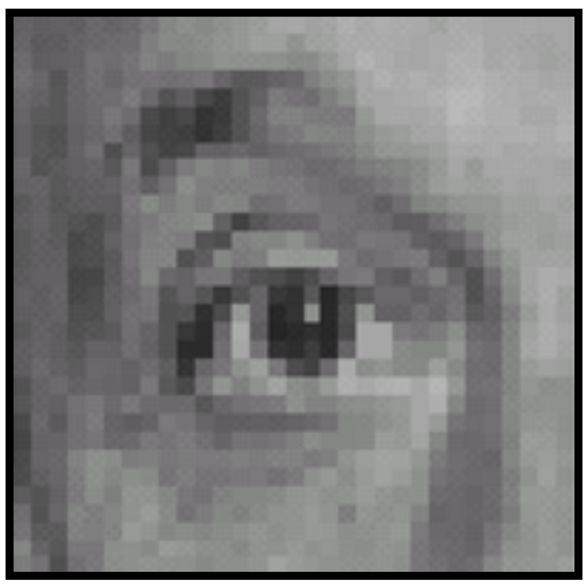
Original

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

?

Source: D. Lowe

Practice with linear filters



Original

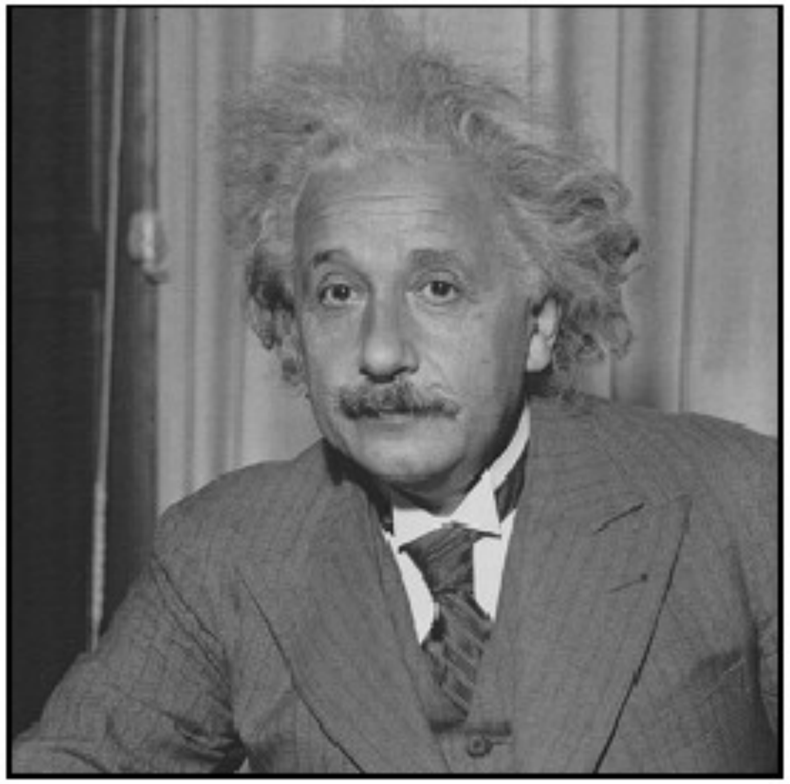
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

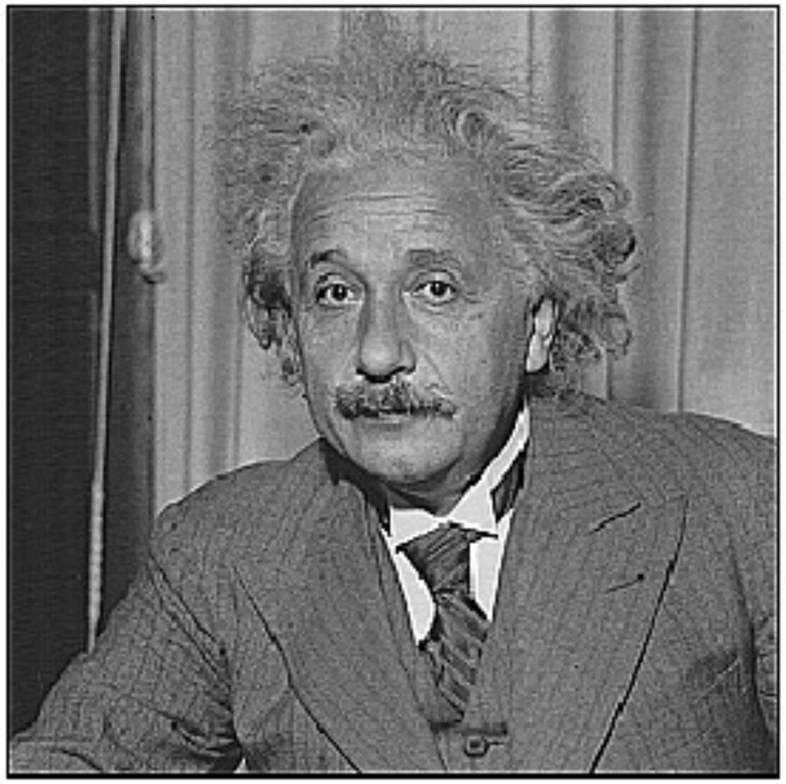
- Accentuates differences with local average

Source: D. Lowe

Filtering examples: sharpening



before



after

Convolutions and the Fourier Transform

- What to do near the edge of the image?
- Understanding the Fourier Transform
- The convolution Theorem
- Understanding Convolutions using the Fourier Transform

Lund University

What to do near the edge of the image?

In practice we do not have infinite images.
How should we treat the edges of the image? What values should one assume 'outside' the image.
Some common choices are

1. Only calculate the result where we can be certain. The result is a smaller image.
2. Assume that there are zeros outside the image. This often means that we introduce artificial sharp edges at the border.
3. Make a periodic expansion of the image, i.e. assume that the image is periodic. This fits well with the theory for discrete fourier transform.

What to do near the edge of the image?

Assume that one would like to convolute the image

$$f = \begin{bmatrix} \underline{1} & 2 & 3 & 5 \\ 1 & 3 & 2 & 1 \\ 2 & 2 & 2 & 2 \end{bmatrix}$$

with the smoothing filter

$$h = \begin{bmatrix} \underline{1} & 1 \\ 1 & 1 \end{bmatrix}$$

LUND UNIVERSITY

What to do near the edge of the image?

(1) Don't let h extend outside f

$$\begin{bmatrix} 7 & 10 & 11 \\ 8 & 9 & 7 \end{bmatrix}$$

(2) Extend with zeros $\Rightarrow$ equal or larger resulting $h * f$-image

$$\begin{bmatrix} \underline{1} & 3 & 5 & 8 & 5 \\ 2 & 7 & 10 & 11 & 6 \\ 3 & 8 & 9 & 7 & 3 \\ 2 & 4 & 4 & 4 & 2 \end{bmatrix}$$

LUND UNIVERSITY

What to do near the edge of the image?

(3) Extend f and h to periodic functions with the same period: f_p, $h_p \Rightarrow$ periodic $h_p * f_p$ result with same period

$$\begin{bmatrix} \underline{10} & 7 & 9 & 12 \\ 8 & 7 & 10 & 11 \\ 6 & 8 & 9 & 7 \end{bmatrix}$$

Here we have also made a periodic function of h:

$$h = \begin{bmatrix} \underline{1} & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad .$$

LUND UNIVERSITY

Discrete Fourier Transform - 2D

$$F(u,v) = \sum_{x=1}^{M} \sum_{y=1}^{N} f(x,y)\, e^{-i2\pi((u-1)(x-1)/M+(v-1)(y-1)/N)}$$

$$f(x,y) = \frac{1}{MN} \sum_{u=1}^{M} \sum_{v=1}^{N} F(u,v)\, e^{i2\pi((u-1)(x-1)/M+(v-1)(y-1)/N)}$$

Fourier transform

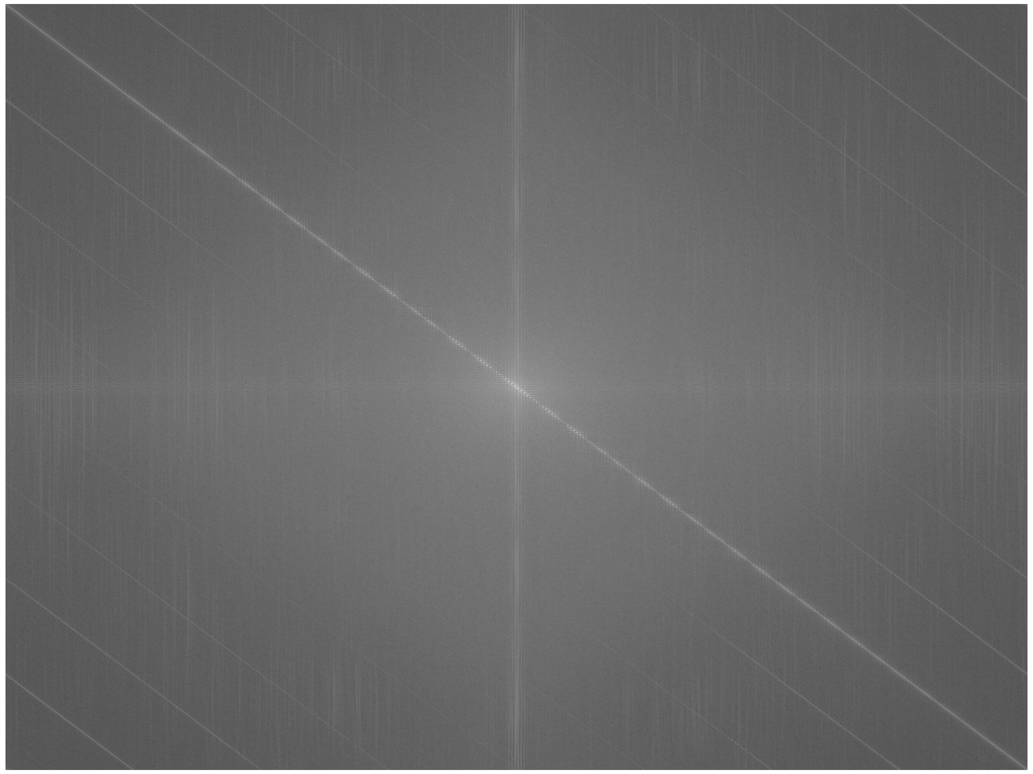
- Image

Fourier transform

• Image

• abs(fft2(I))

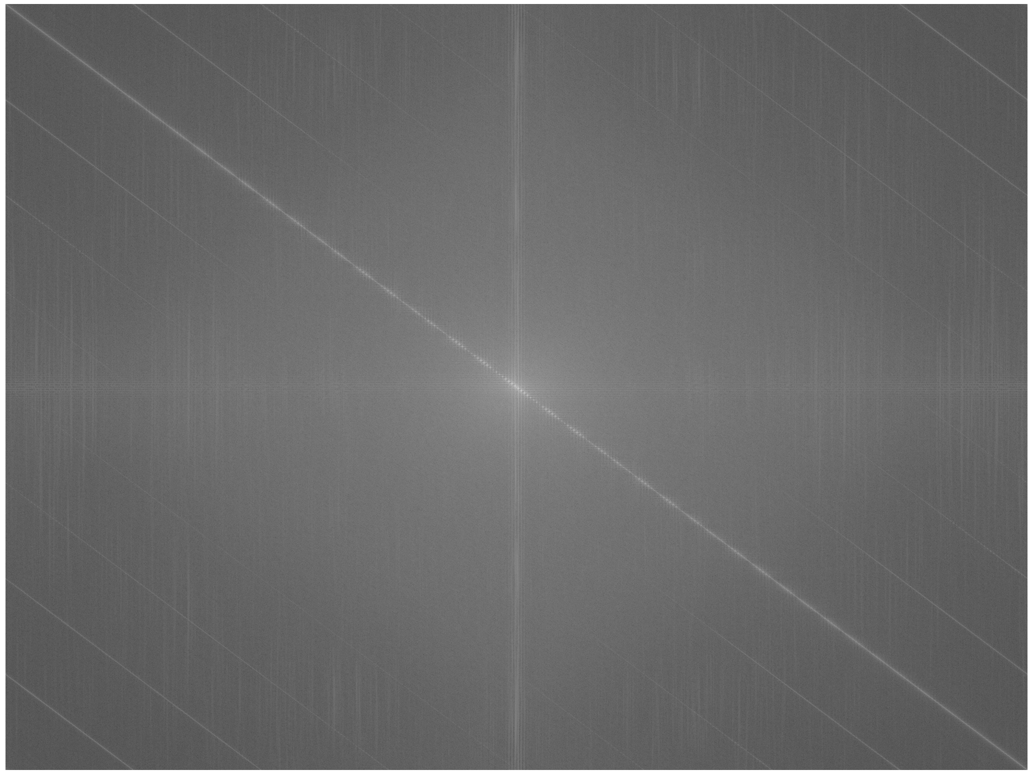
Fourier transform



- Image
- log(abs(fft2(I)))

Edge effects

- Image

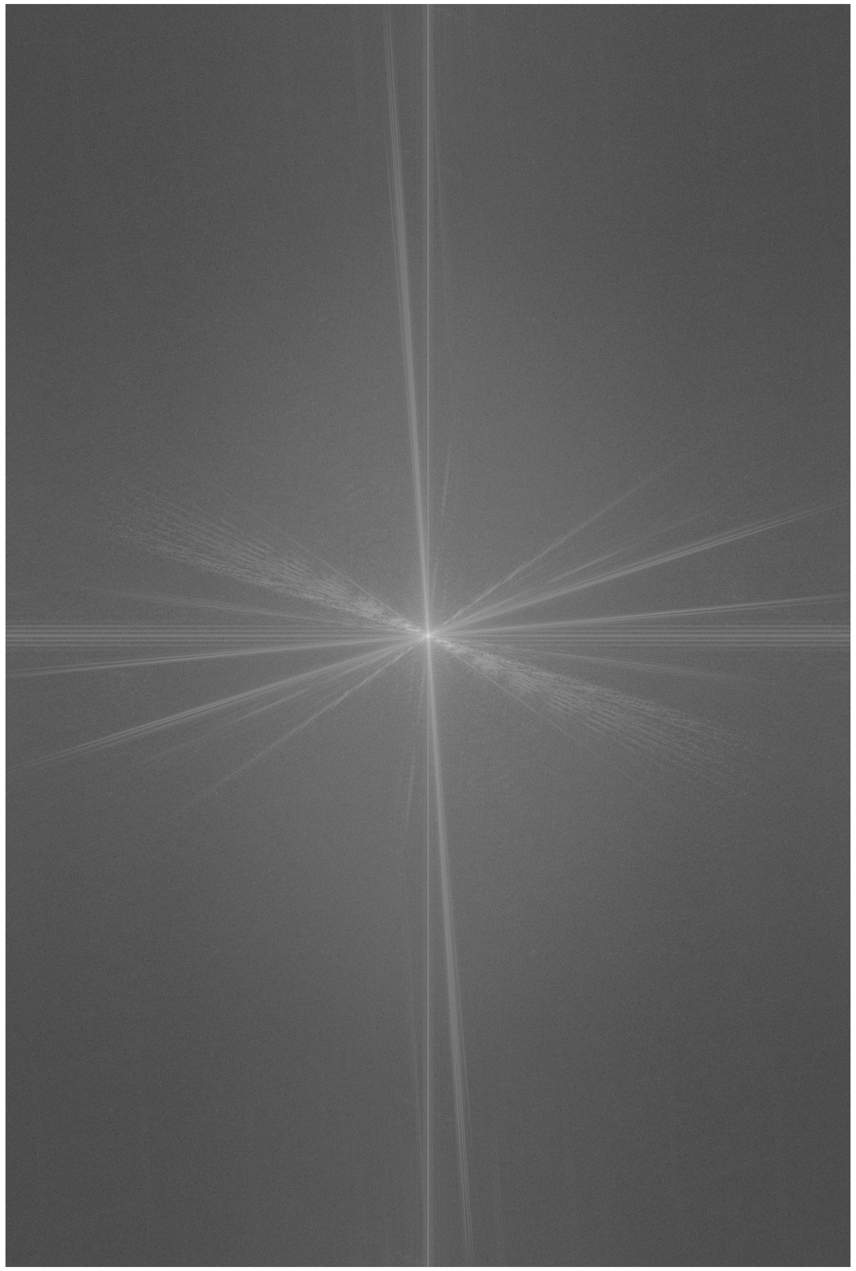


- log(abs(fft2(I)))

Fourier transform

- Image

Fourier transform



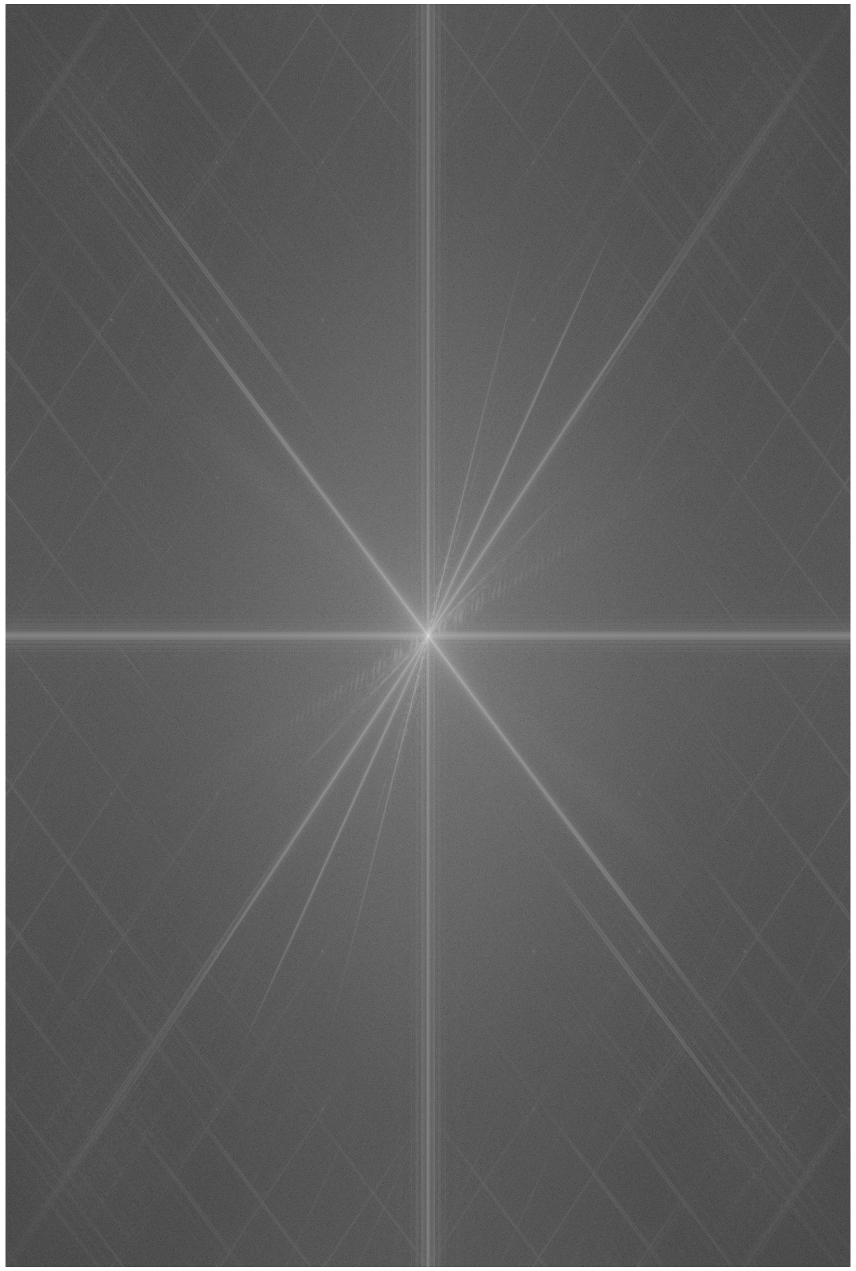
•Image

•Fourier transform

Fourier transform

- Image

Fourier transform



- Image
- Fourier transform

Using FFT for convolutions

1. $f \quad \rightarrow \quad FFT \quad \rightarrow F$
2. $h \quad \rightarrow \quad FFT \quad \rightarrow \quad H$
3. $H, F \quad \rightarrow \quad \times \quad \rightarrow \quad H \cdot F$
4. $H \cdot F \quad \rightarrow \quad IFFT \quad \rightarrow \quad h * f$

The computational complexity of using FFT for a convolution is:

$$2\frac{N \log N}{2} + N + \frac{N \log N}{2} \sim \frac{3}{2} N \log N$$

Calculation based on the definition gives complexity N^2

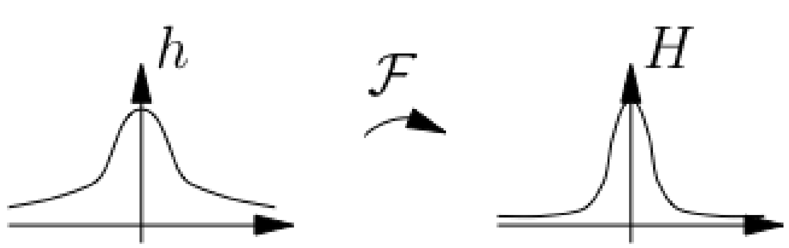
Frequency function

$$g(x) = h * f = \int h(x-y)f(y)dy$$

$\mathcal{F}g = G$, $\mathcal{F}h = H$, $\mathcal{F}f = F$

$$G(u,v) = H(u,v)F(u,v) \quad .$$

Definition
$H = \mathcal{F}(h)$ is called the **frequency function** of h.



Filter for image enhancement

signal plane	frequency plane
smoothing	low pass
sharpening	high pass

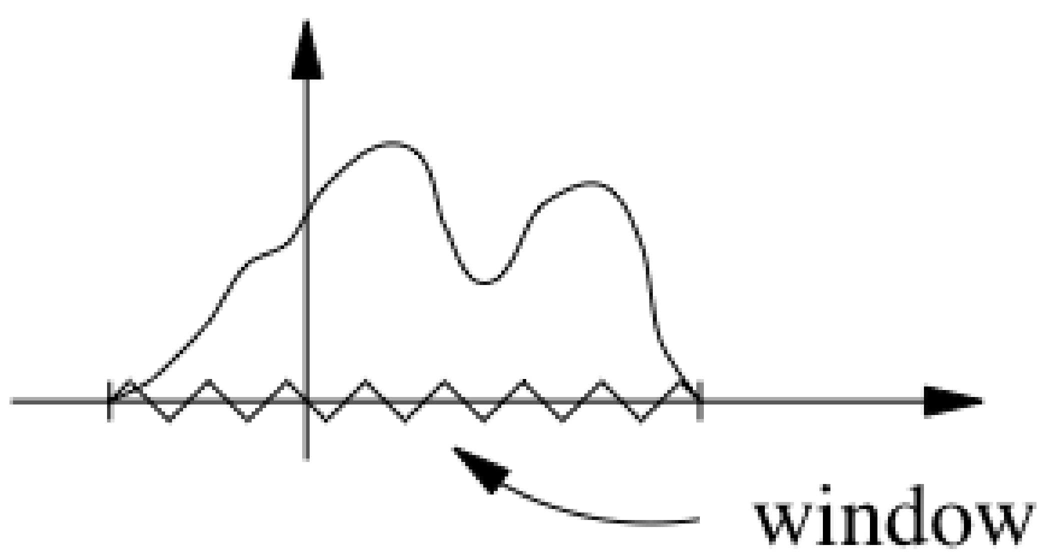
For discrete functions: $DFT(h * f)(u, v) = H(u, v)F(u, v)$.

LUND UNIVERSITY

Let the output, g be give by the convolution

$$g(x) = S(f)(x) = \int h(x-y)f(y)dy \ ,$$

where f represents the input and h the impulse response
If $g(x)$ only depends on f:s values in a surrounding (=a small window) of x then S is called a **window operator**.
The window is given by $\{\, x \mid h(x) \neq 0 \,\}$.



LUND UNIVERSITY

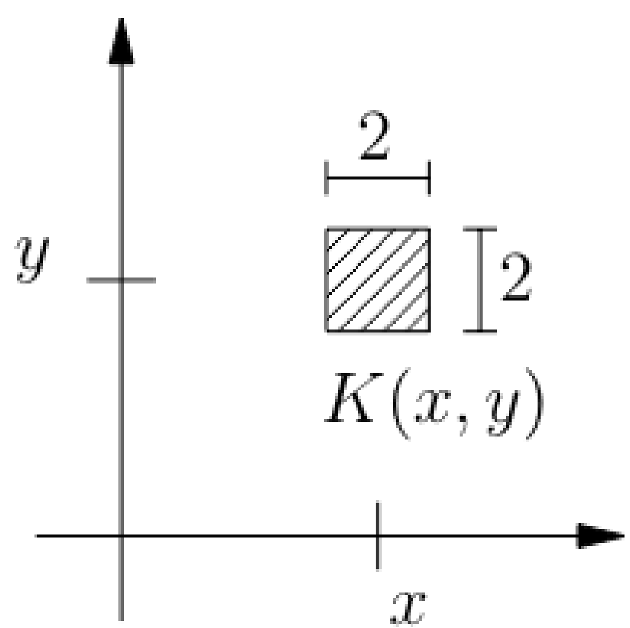
Assume that $f(x, y)$ represents a continuous image. Let

$$h(x, y) = \mathrm{rect}(x)\,\mathrm{rect}(y) \quad .$$

Then

$$S(f) = h * f = \int_{K(x,y)} f(s, t)dsdt \quad ,$$

where the region of integration $K(x, y)$ is a unit square with centre at (x, y).

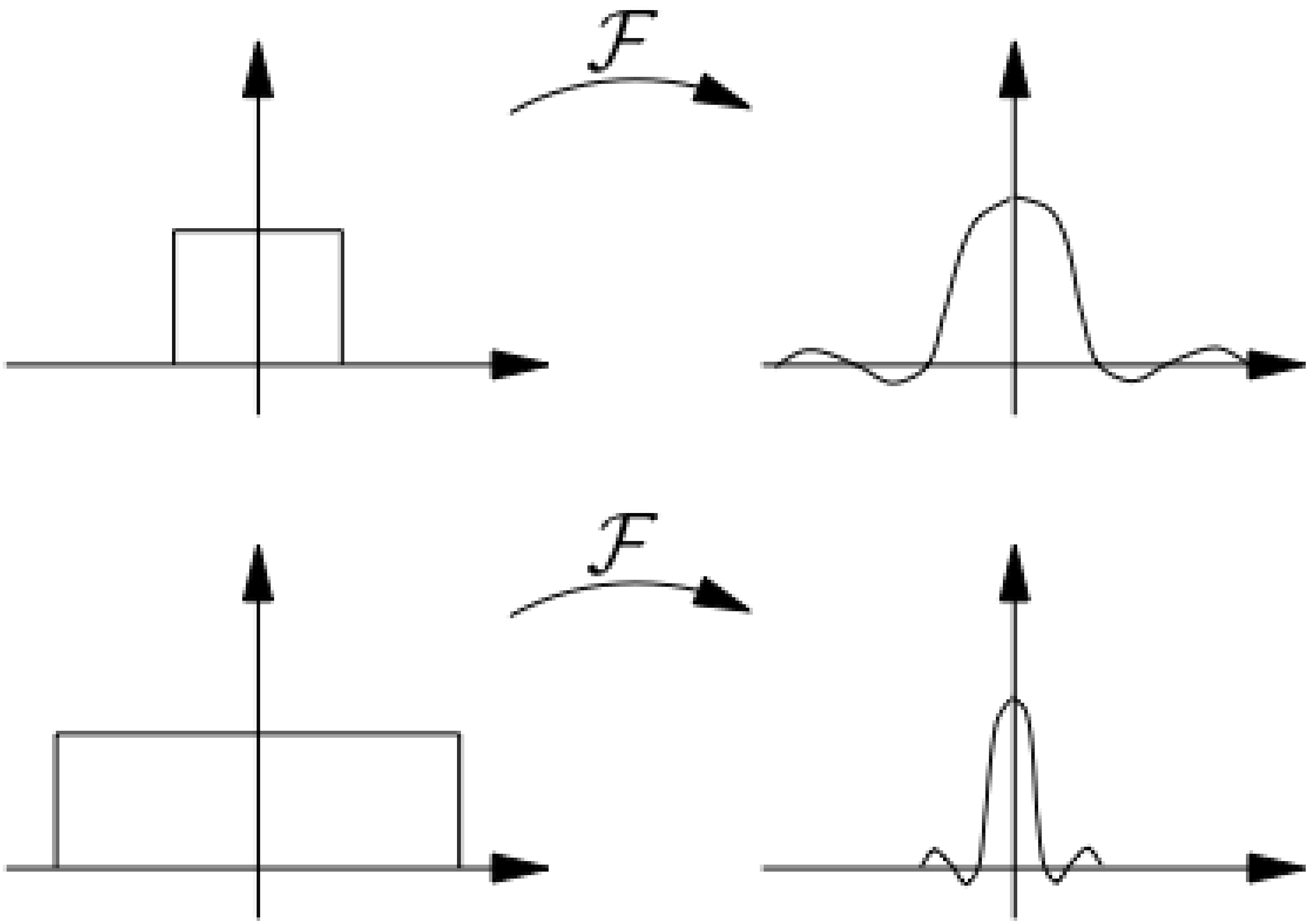


S is called a **mean value operator**.
The fourier transform gives

$$H(u) = 4\,\mathrm{sinc}(2\pi u)\,\mathrm{sinc}(2\pi v) \quad .$$

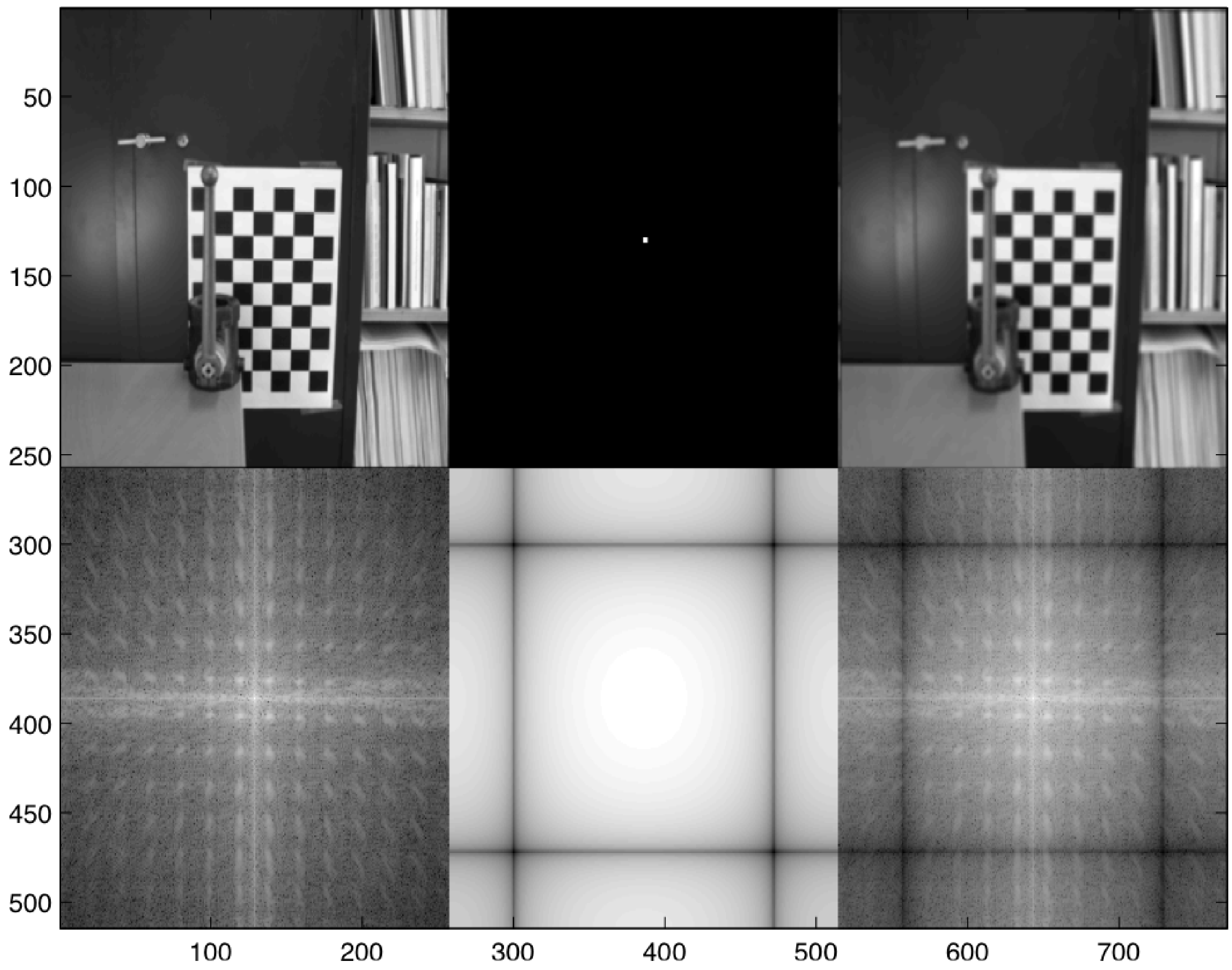
The scaling rule (page 148 in Forsythe-Ponce)

$$f(\lambda x) \rightarrow \frac{1}{\lambda} F(\frac{u}{\lambda}) \quad .$$



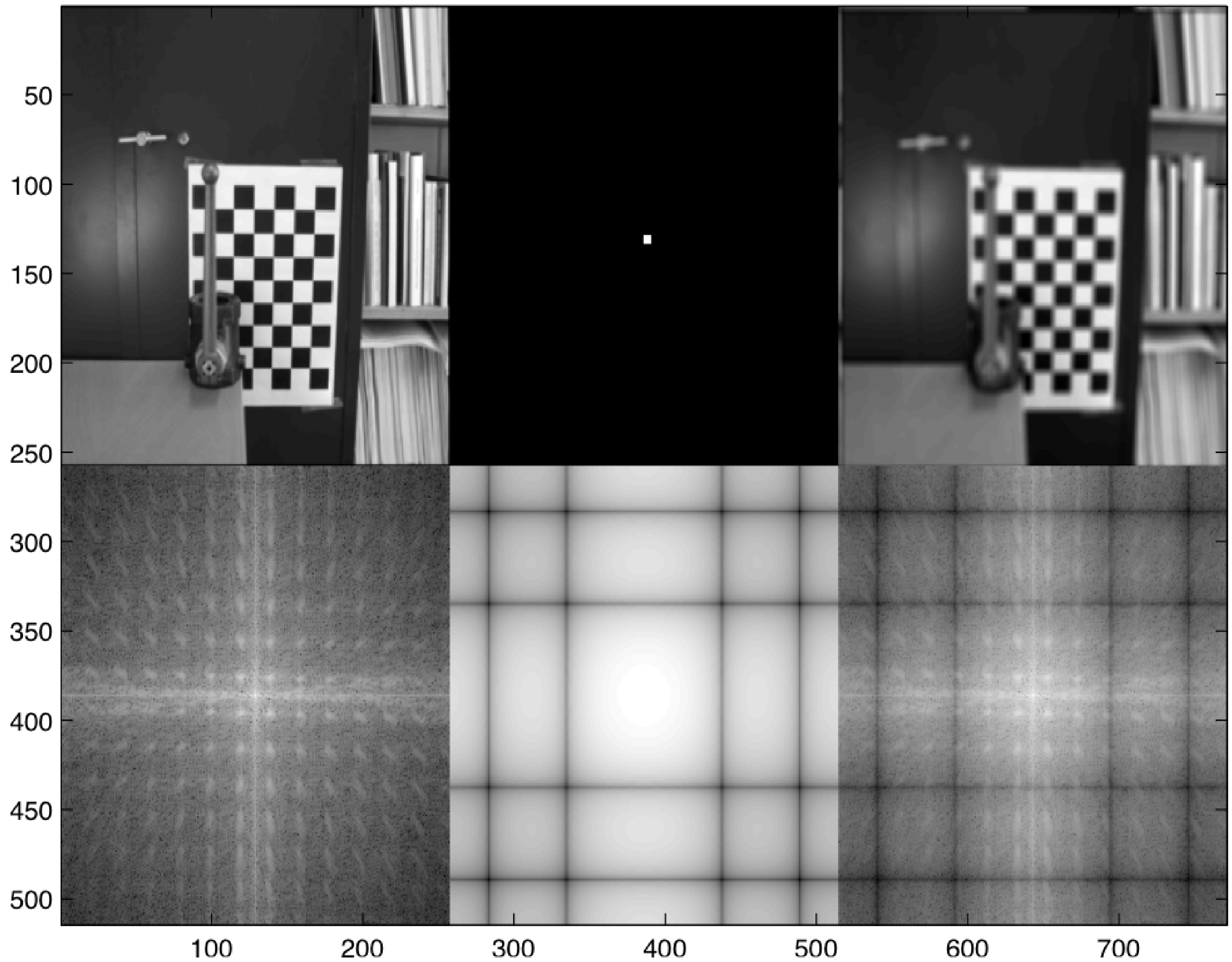
$\mathcal{F}$
$\mathcal{F}$

signal space: image, filter, result



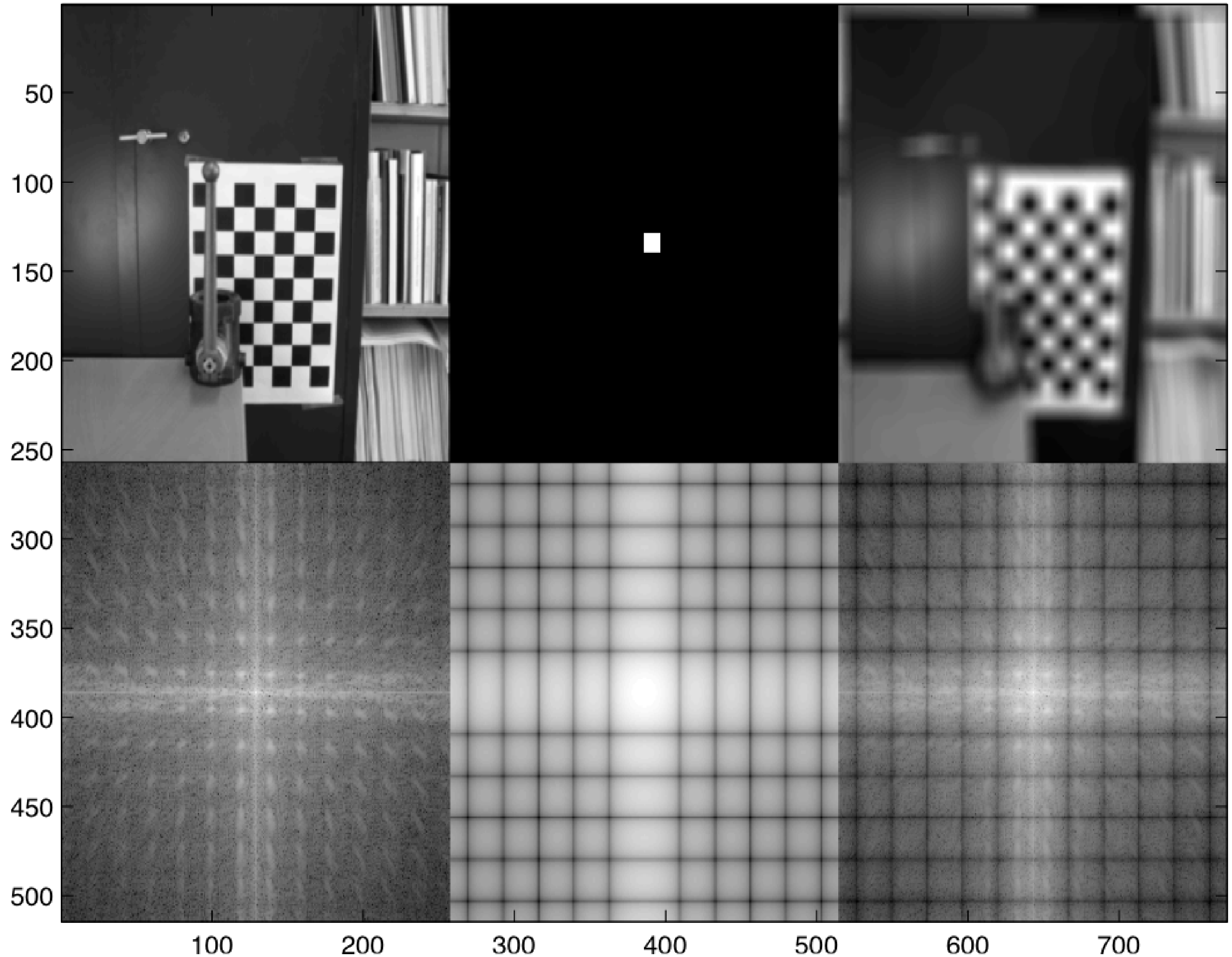
frequency space: image, filter result

signal space: image, filter, result



frequency space: image, filter result

signal space: image, filter, result



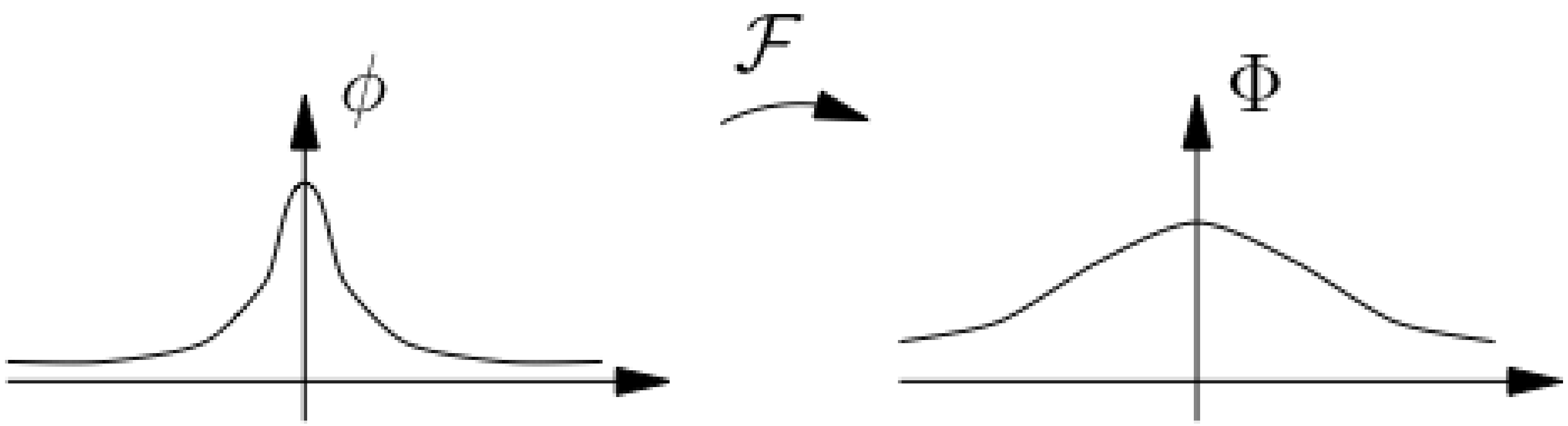
frequency space: image, filter result

LUND UNIVERSITY

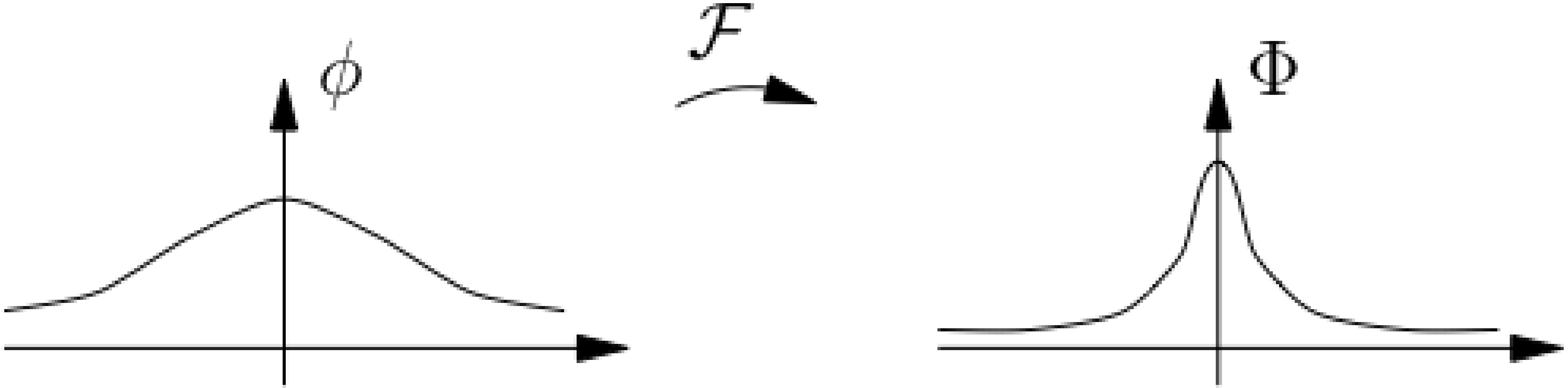
Example

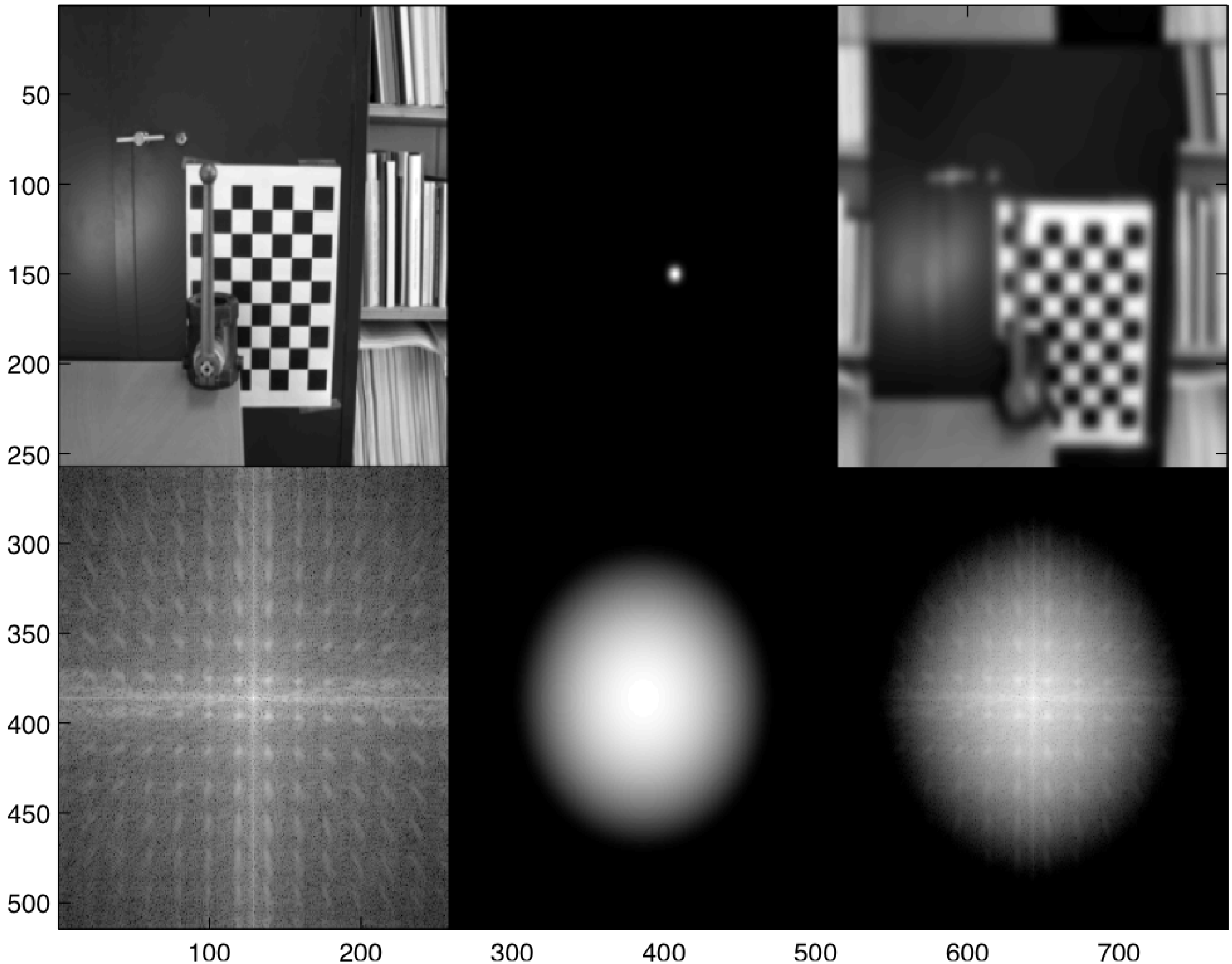
Notice that

$$\phi(x) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-x^2/(2\sigma^2)} \quad \rightarrow \quad \Phi(u) = e^{-2(\sigma\pi u)^2}.$$



Larger σ gives



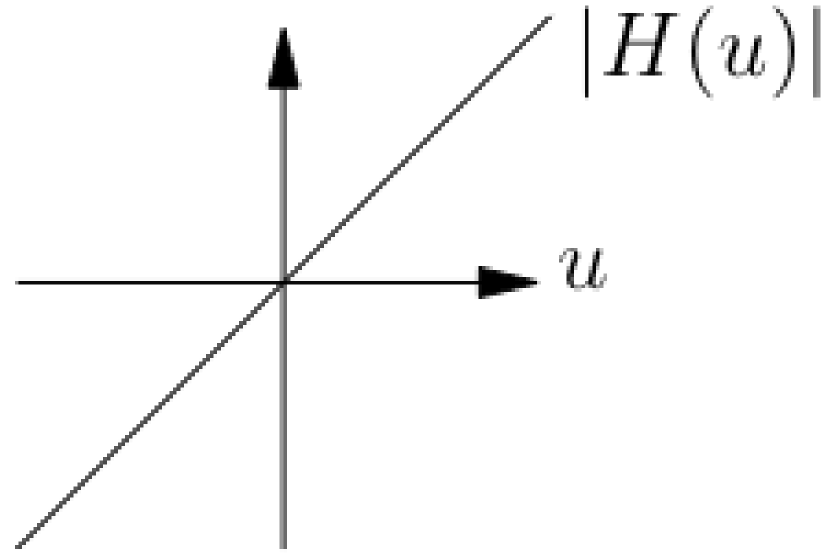

50
100
150
200
250
300
350
400
450
500
100
200
300
400
500
600
700

Example

Differentiation

$$\frac{\partial f}{\partial x} \quad \rightarrow \quad 2\pi i u F(u)$$

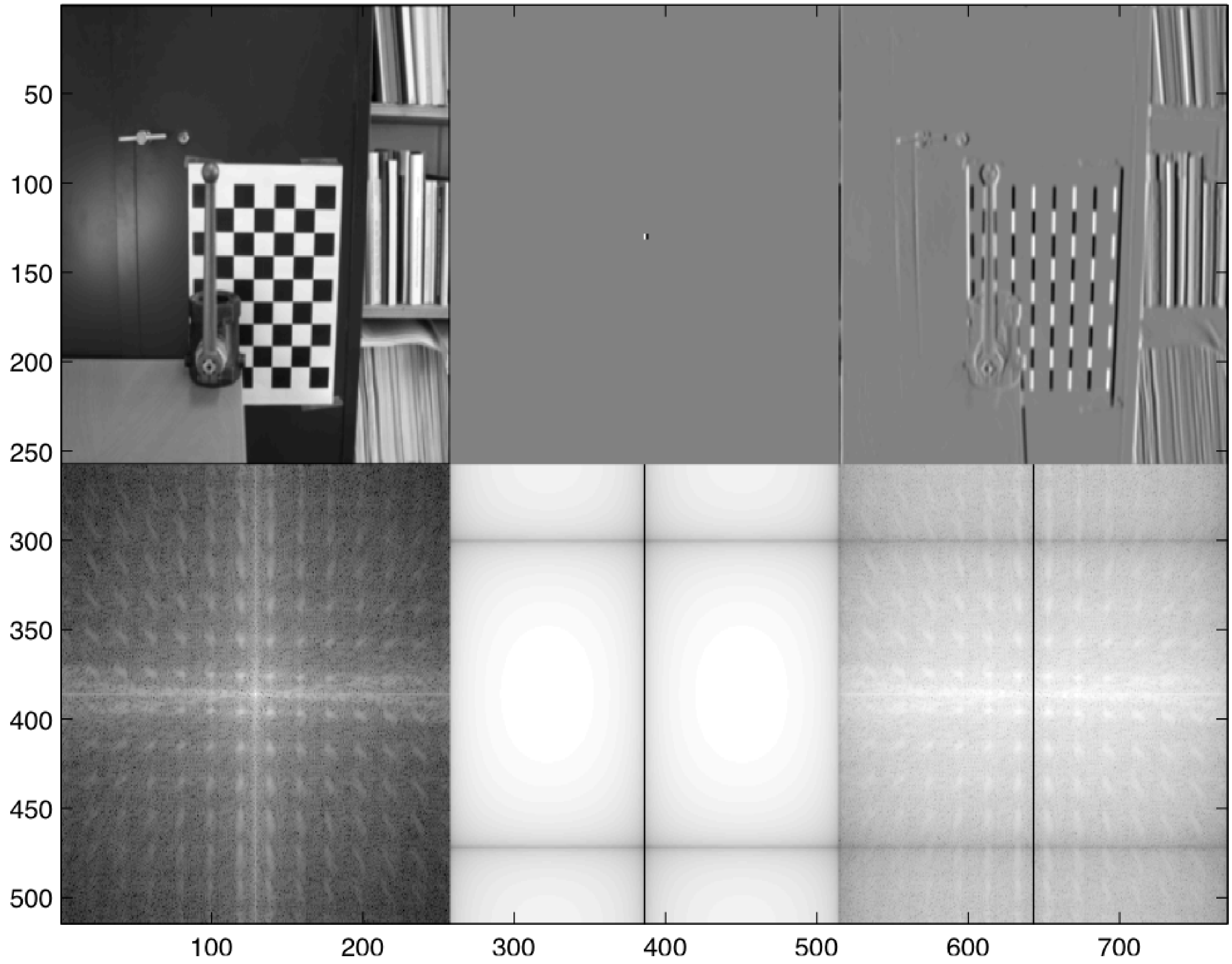
$$H(u) = 2\pi i u$$



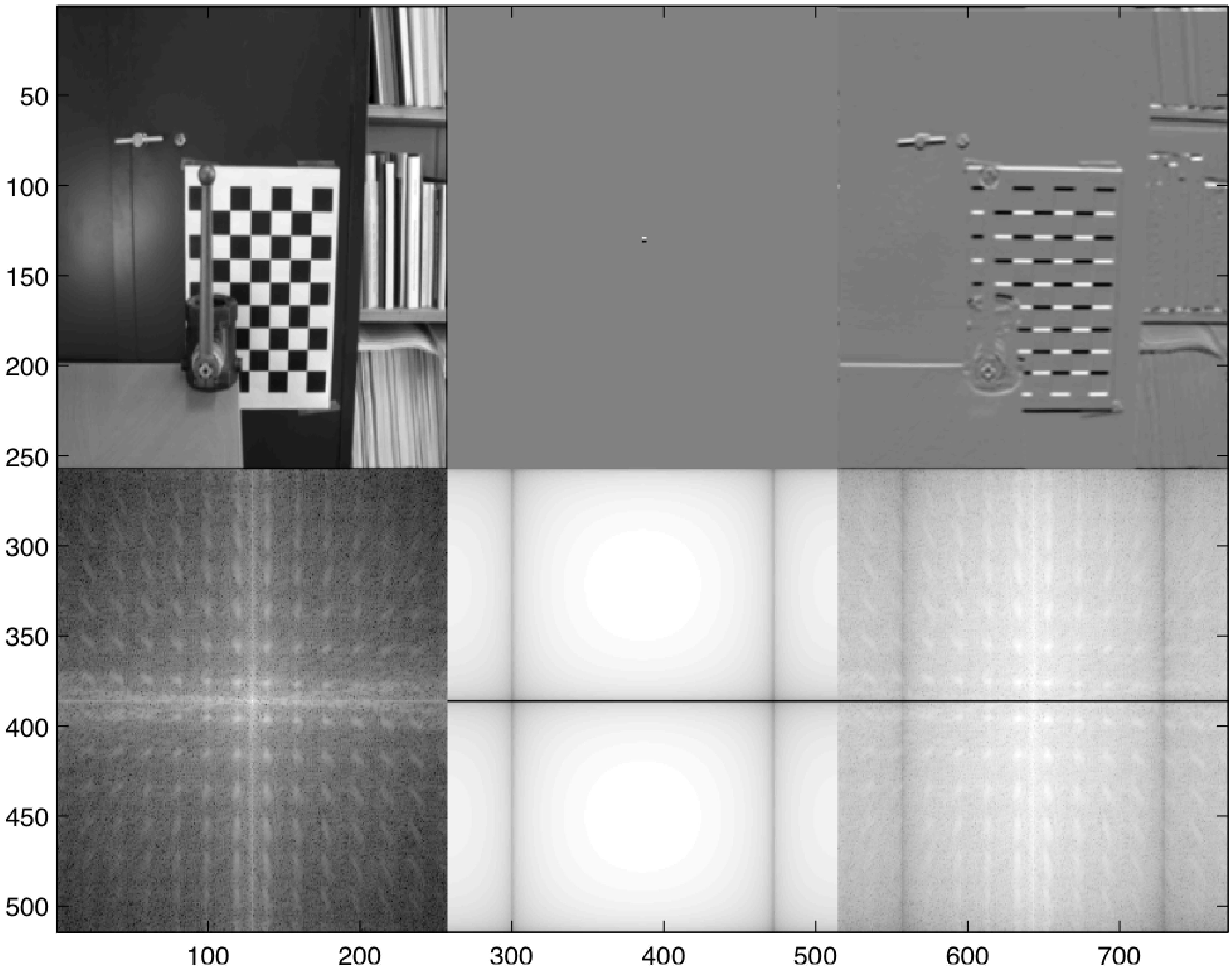
Sensitive to noise.
Combine with smoothing:

$$f \to \phi * f \to \frac{\partial}{\partial x}\phi * f$$

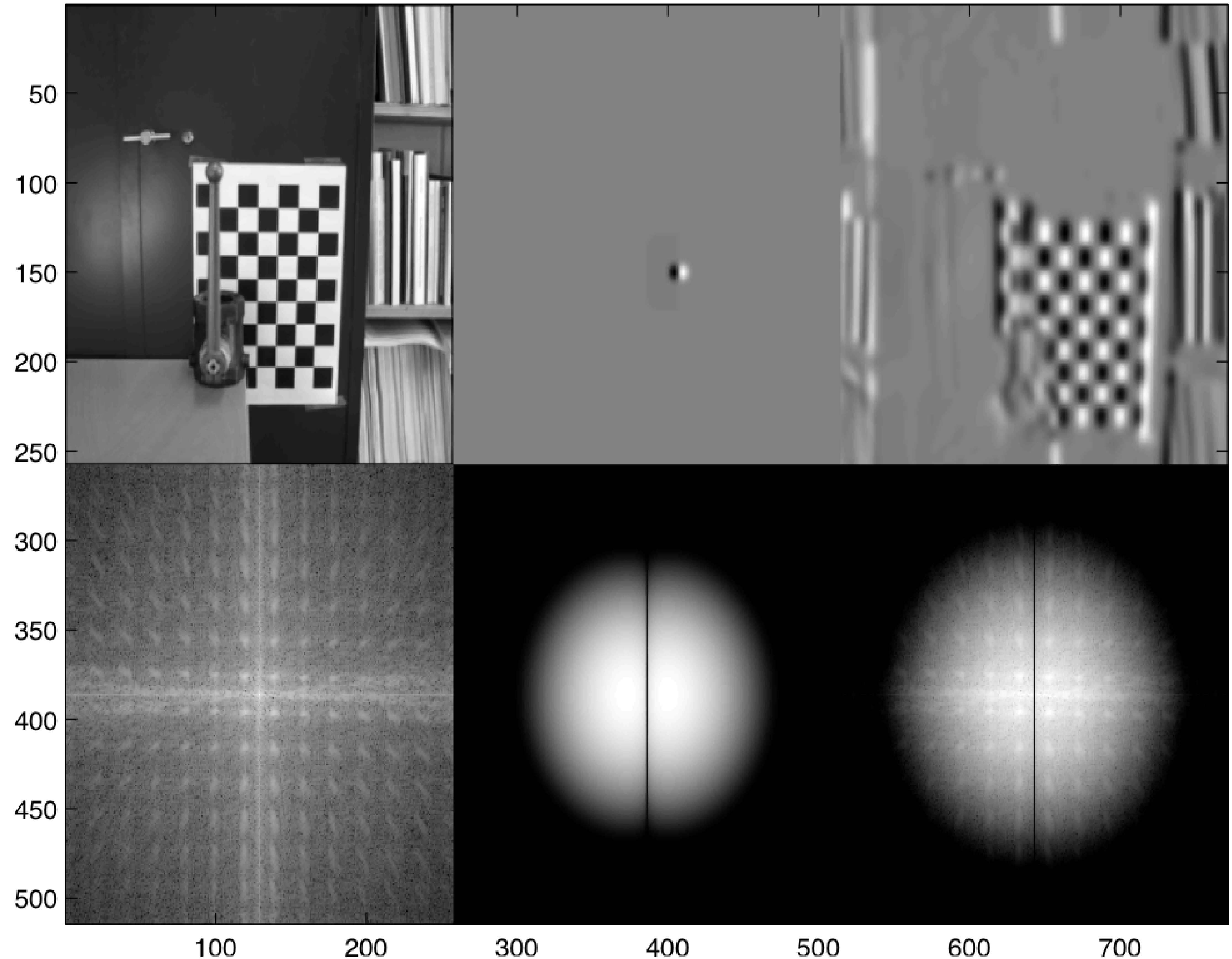
$$\frac{\partial}{\partial x}\phi = -\frac{x}{\sqrt{2\sigma^6\pi}}e^{-x^2/(2\sigma^2)}$$

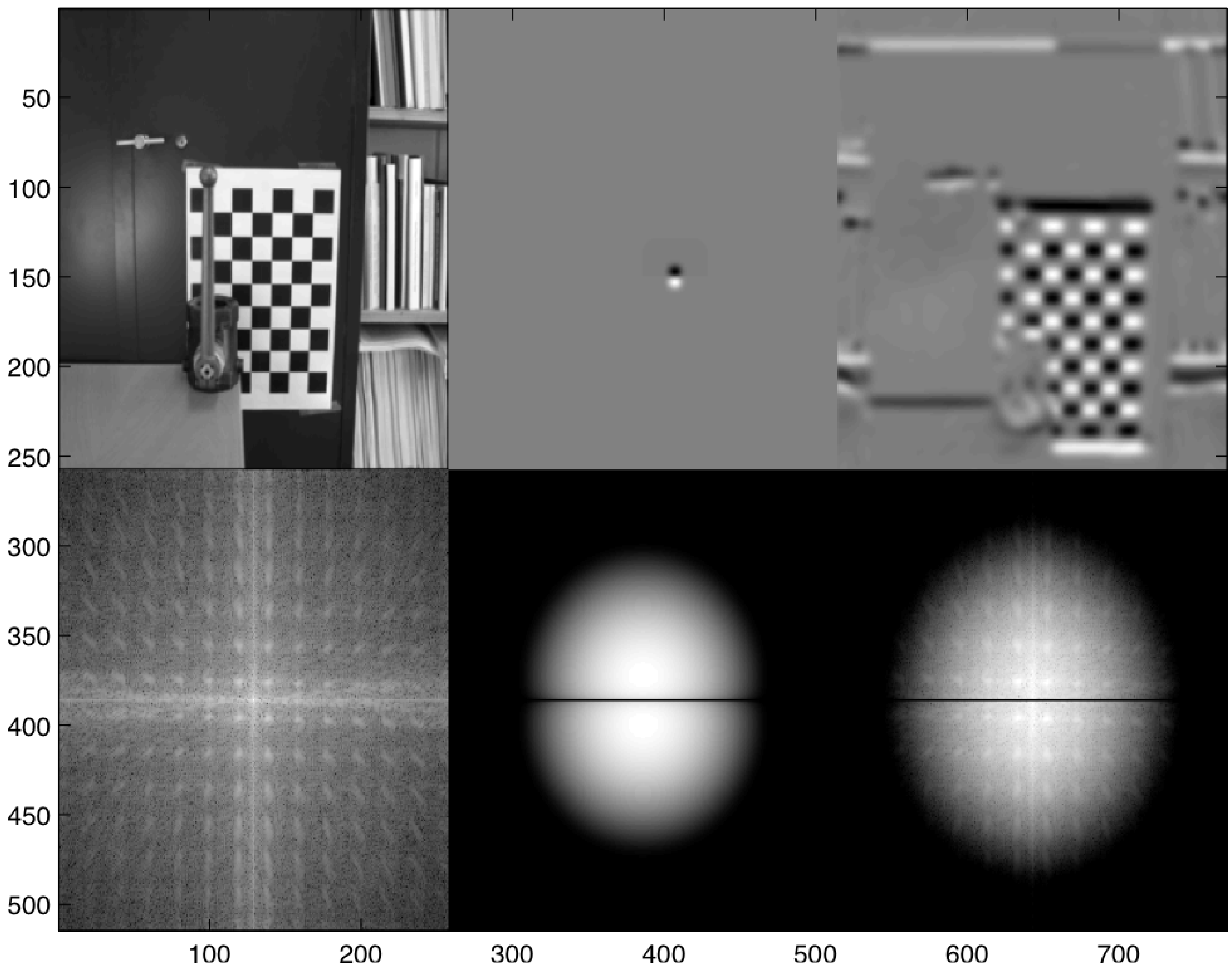


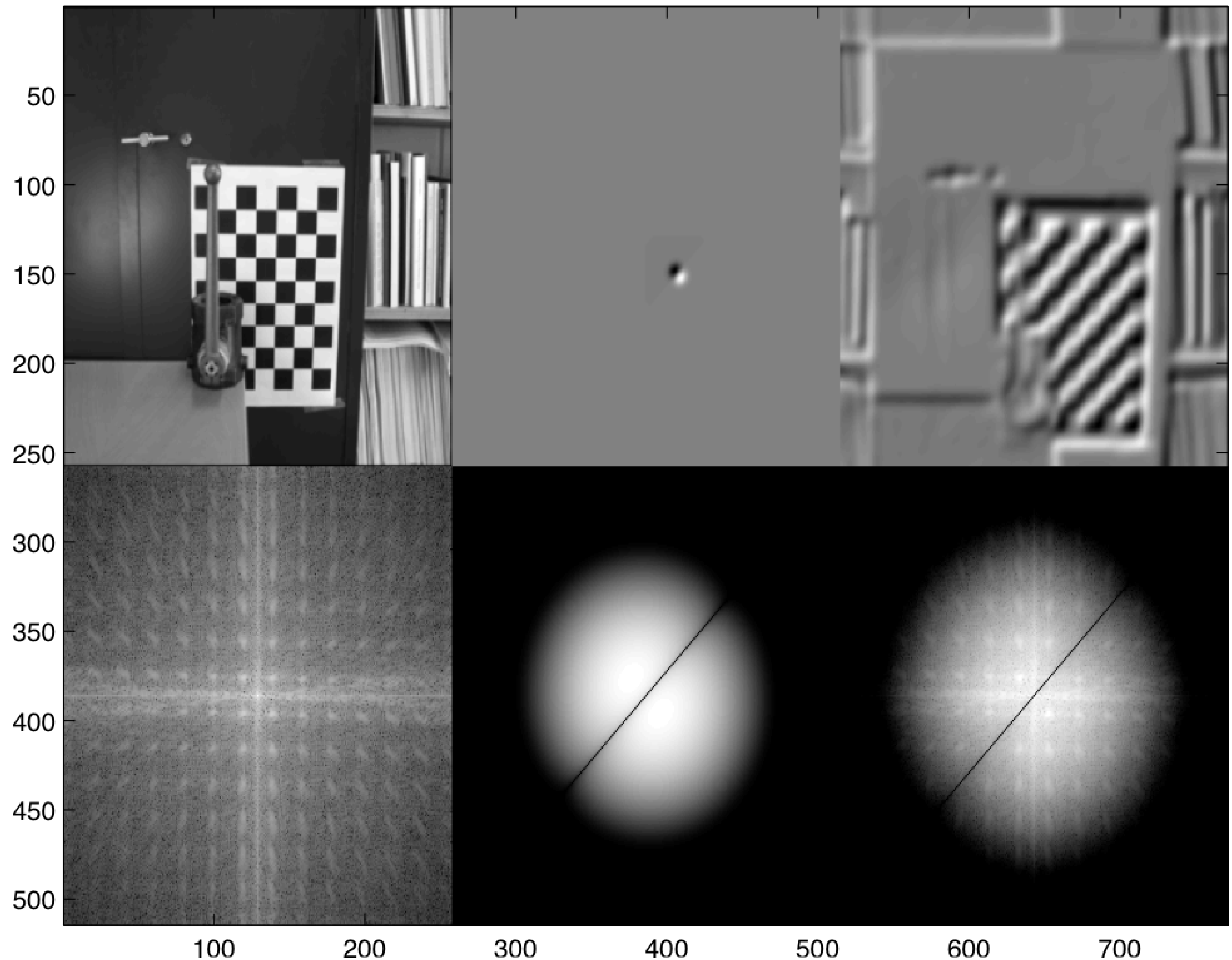
50
100
150
200
250
300
350
400
450
500
100
200
300
400
500
600
700

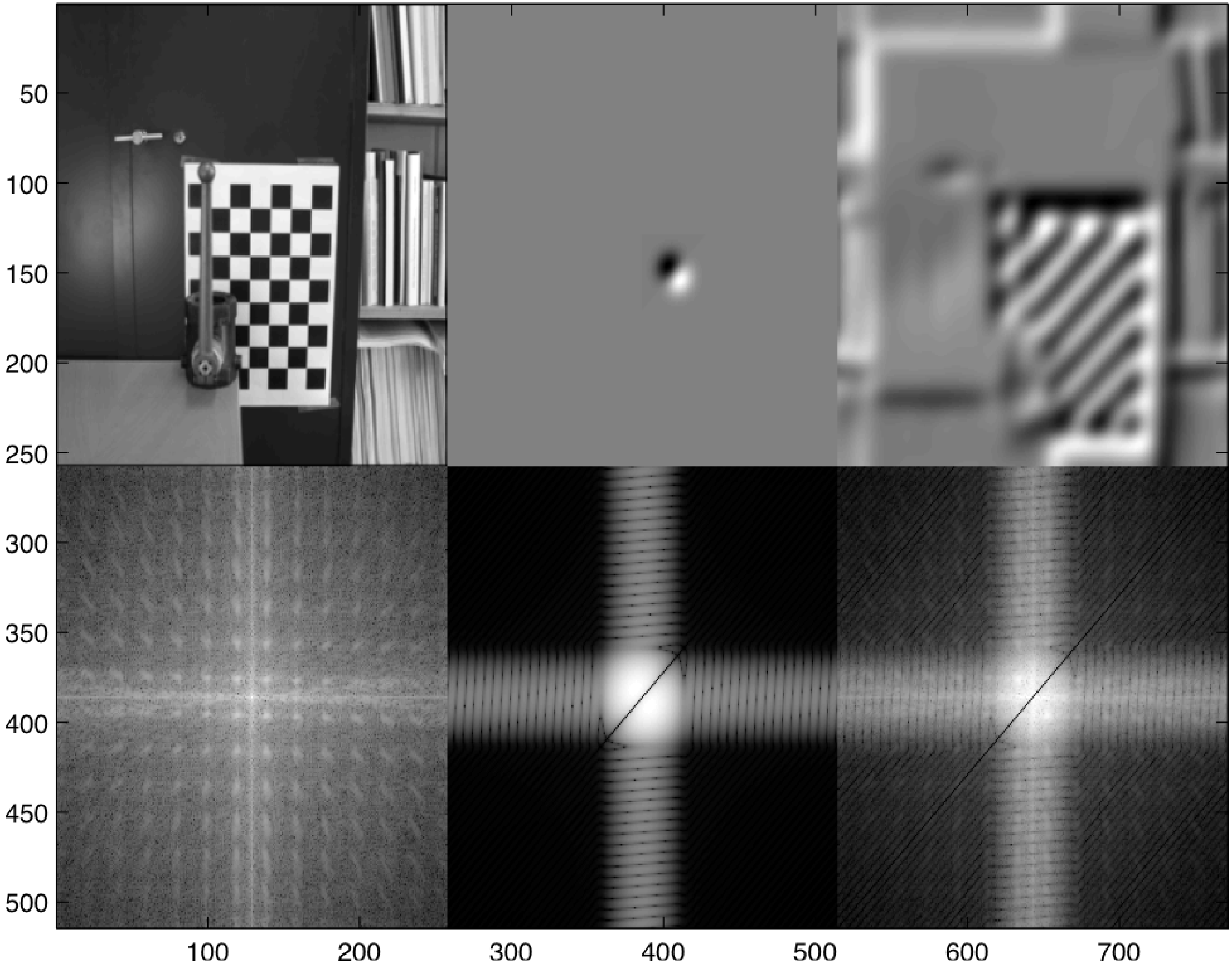


50
100
150
200
250
300
350
400
450
500
100
200
300
400
500
600
700


50
100
150
200
250
300
350
400
450
500
100
200
300
400
500
600
700


50
100
150
200
250
300
350
400
450
500
100
200
300
400
500
600
700


50
100
150
200
250
300
350
400
450
500
100
200
300
400
500
600
700


50
100
150
200
250
300
350
400
450
500
100
200
300
400
500
600
700

Review

- Convolution (with flip) and cross-correlation (without flip)
  - Properties
  - Examples
- Convolution theorem
- Interpreting convolutions through the Fourier transform

- Read lecture notes
- Experiment with matlab demo scripts
- Finish assignment 1

LUND UNIVERSITY