Image and Video Understanding

2VO 710.095 WS

Christoph Feichtenhofer, Axel Pinz

Further reading: Szeliski, Richard. *Computer Vision: Algorithms and Applications*. Springer, 2010, Chapter 3, Section 3.1-3.5

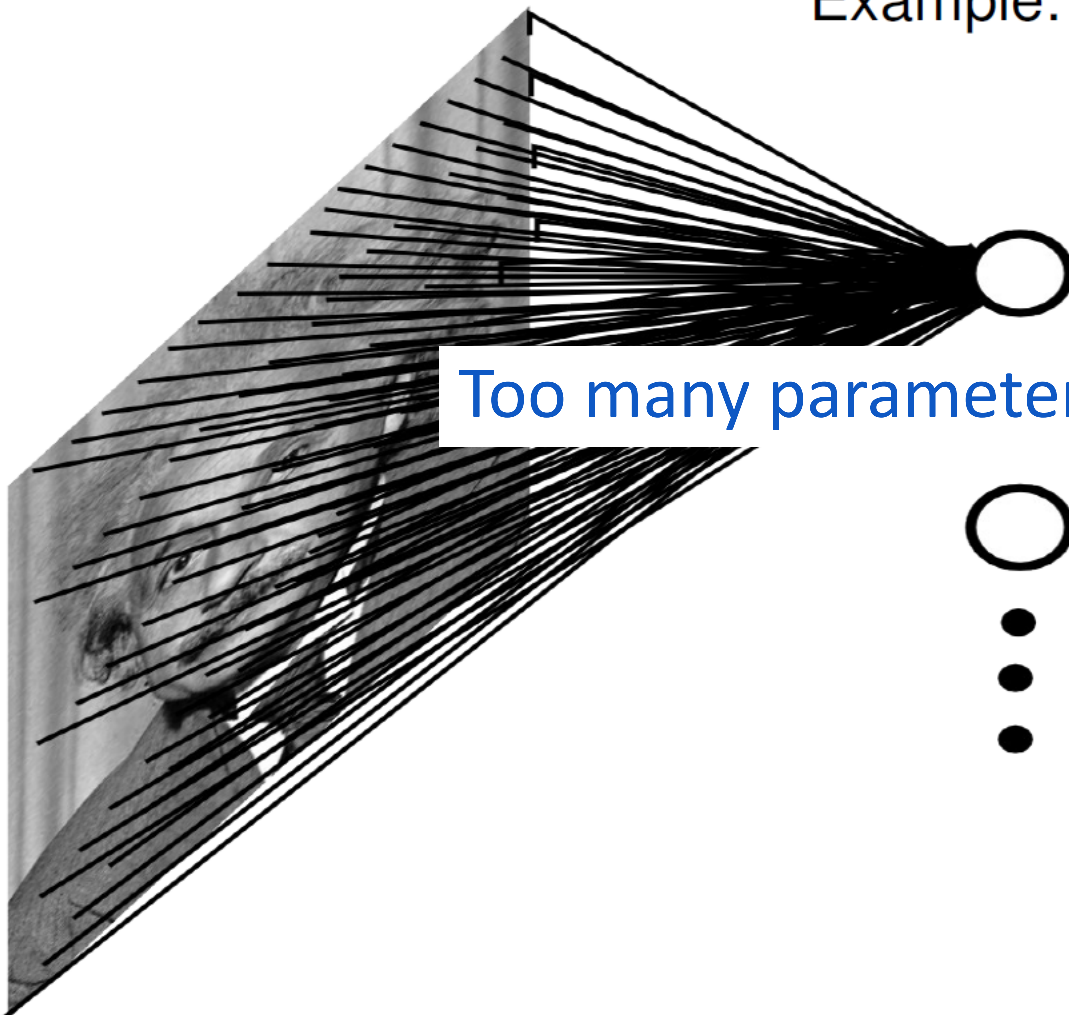
Slide credits:
Many thanks to all the great computer vision researchers on which this presentation relies on.

Outline

- Linear filtering and the importance of **convolution**
  - Apply a filtermask to the local neighborhood at each pixel in the image
  - The filtermask defines how to combine values from neighbors.
  - Can be used for
    - Extract **intermediate representations** to abstract images by higher-level "**features**", for further processing (i.e., preserve the useful information only and discard redundancy)
    - Image **modification,** e.g., to reduce noise, resize, increase contrast, etc.
    - Match **template** images (e.g. by correlating two image patches)
- Image filtering in the frequency domain
  - Provides a nice way to illustrate the effect of linear filtering
    - Filtering is a way to modify the frequencies of images
  - Efficient signal filtering is possible in that domain
  - The frequency domain offers an alternative way to understanding and manipulating the image.

Motivation: Images as a composition of local parts
“Pixel-based” representation

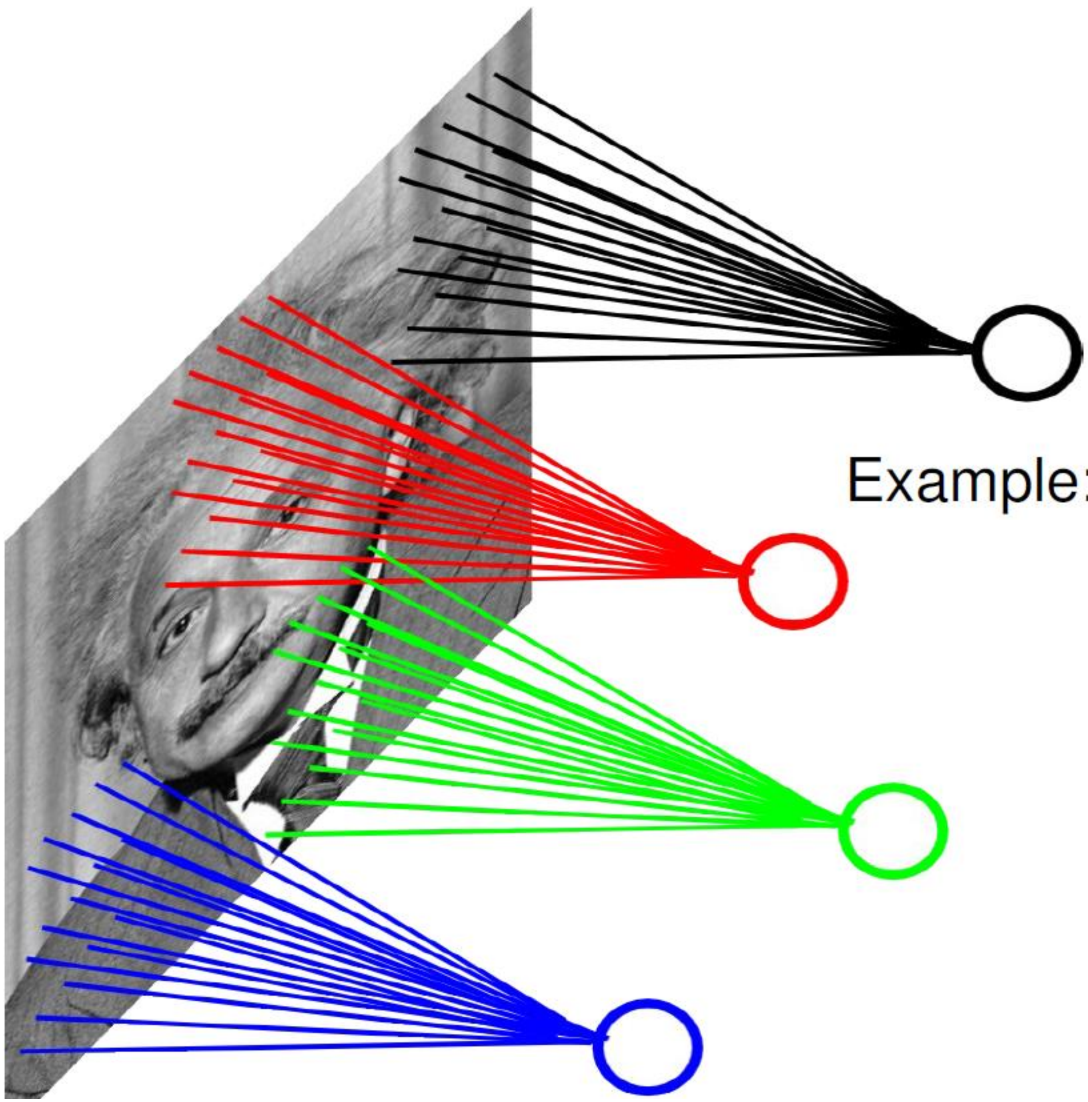
Example: 1000x1000 image



Credit: M. A. Ranzato

Motivation: Images as a composition of local parts

"Patch-based" representation



Example: 1000x1000 image

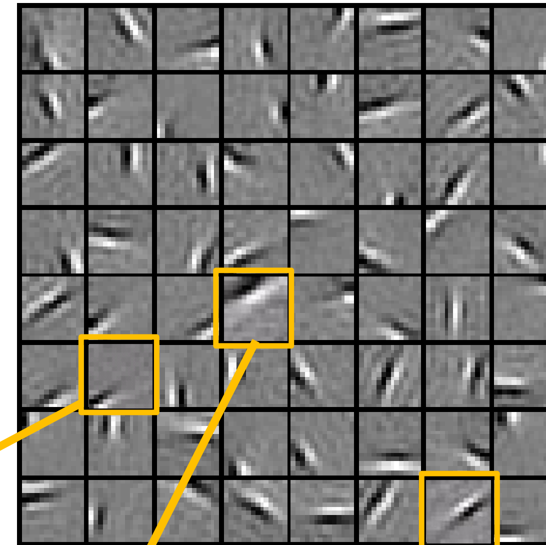
Credit: M. A. Ranzato

Motivation: Images as a composition of local parts

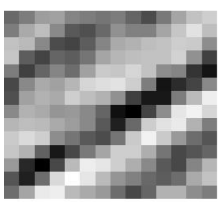
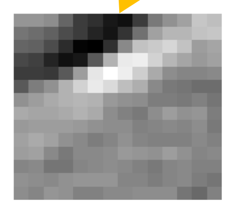
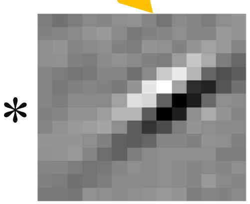
Sparse coding example

Natural Images

Learned bases ($\phi_1, \ldots, \phi_{64}$): "Edges"



Test example

 $\approx 0.8 *$  $+ \; 0.3 *$  $+ \; 0.5 *$ 

$$x \approx 0.8 * \phi_{36} + 0.3 * \phi_{42} + 0.5 * \phi_{63}$$

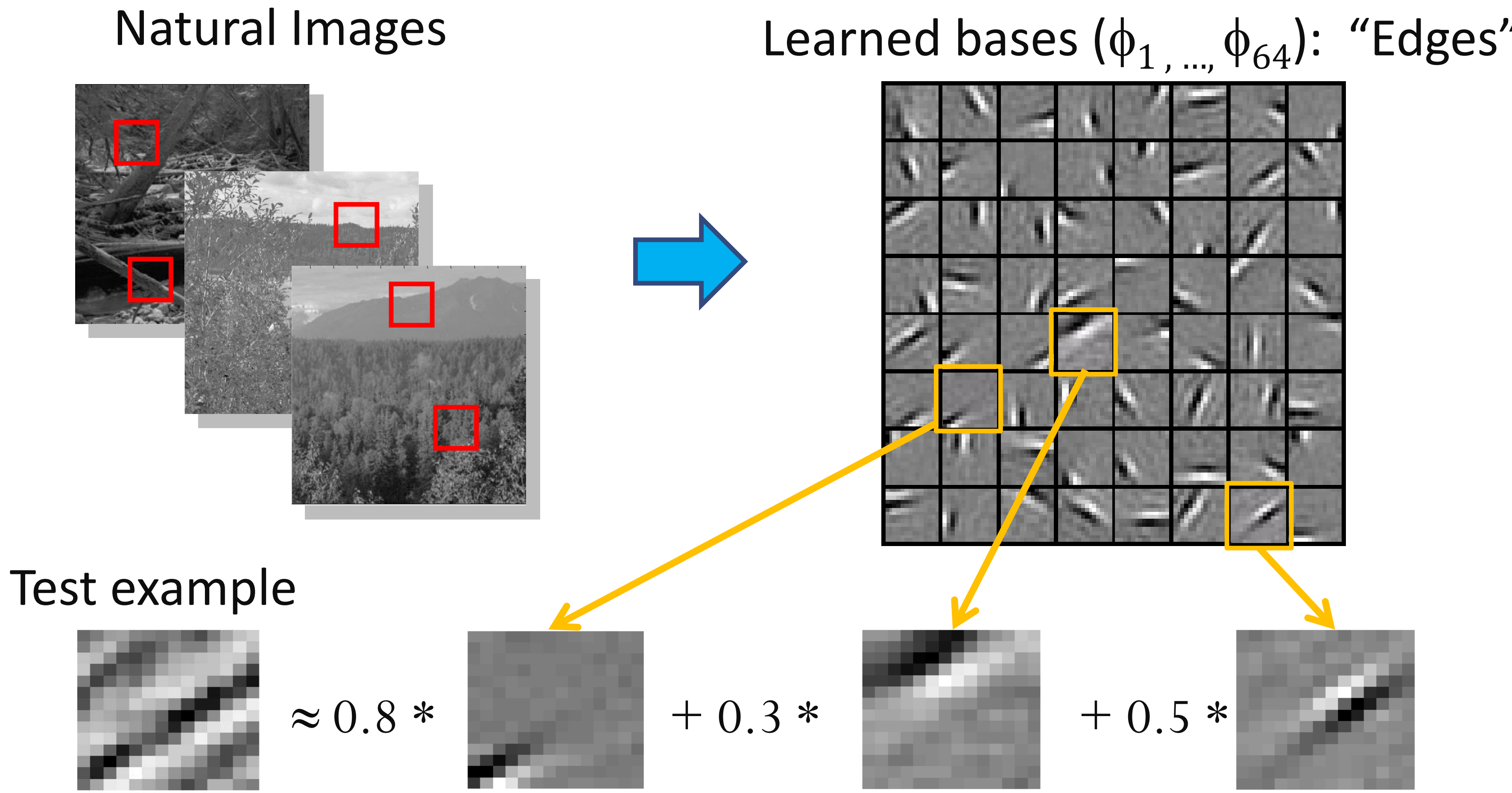
$[a_1, \ldots, a_{64}]$ = [0, 0, …, 0, **0.8**, 0, …, 0, **0.3**, 0, …, 0, **0.5**, 0]

(feature representation)

Credit: A. Coates

Motivation: Images as a composition of local parts

Sparse coding example

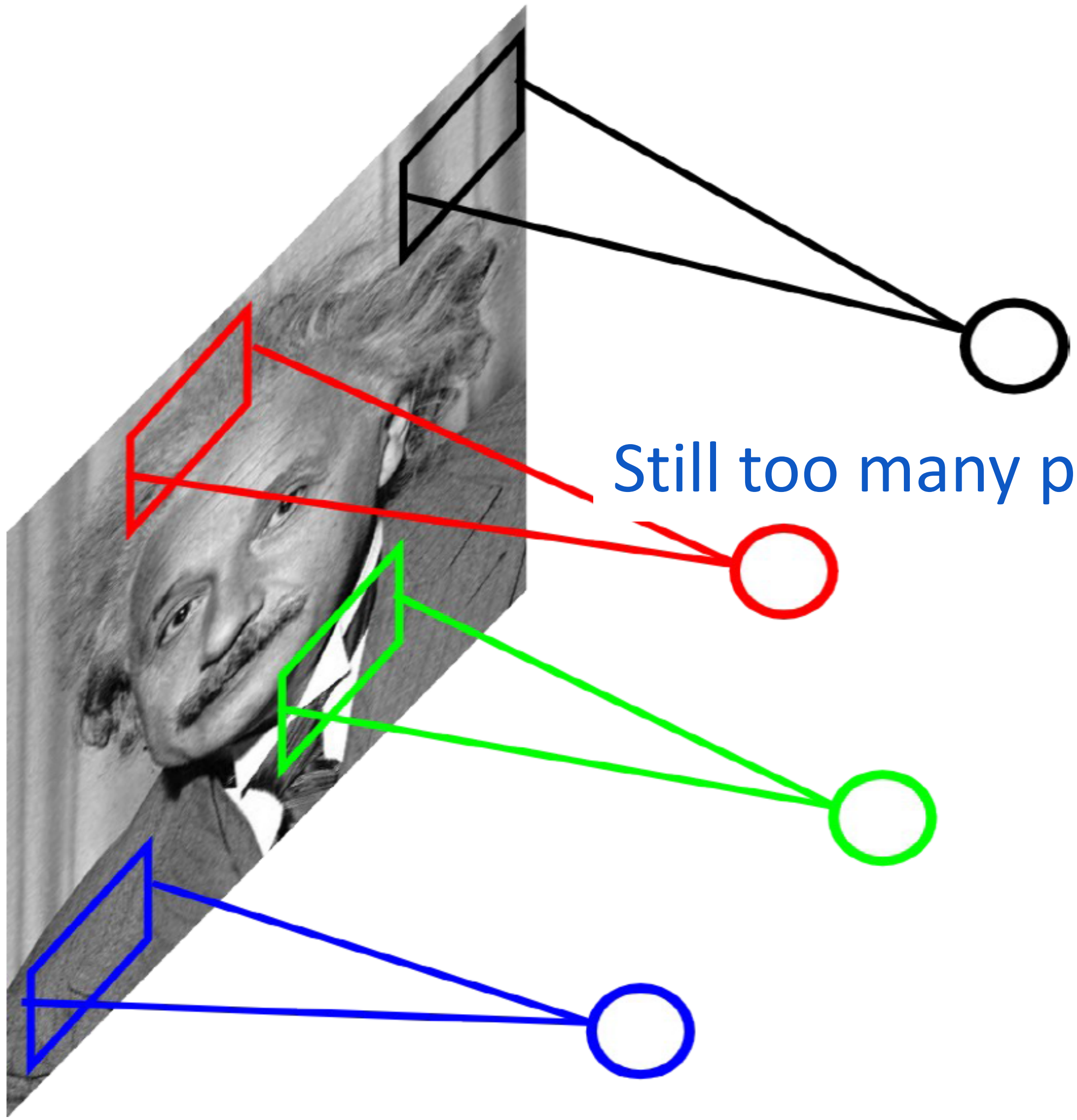


- Method "invents" edge detection
- Automatically learns to represent an image in terms of the edges that appear in it
- Gives a more succinct, higher-level representation than the raw pixels
- Quantitatively similar to primary visual cortex (area V1) in brain

Credit: A. Coates

Motivation: Images as a composition of local parts

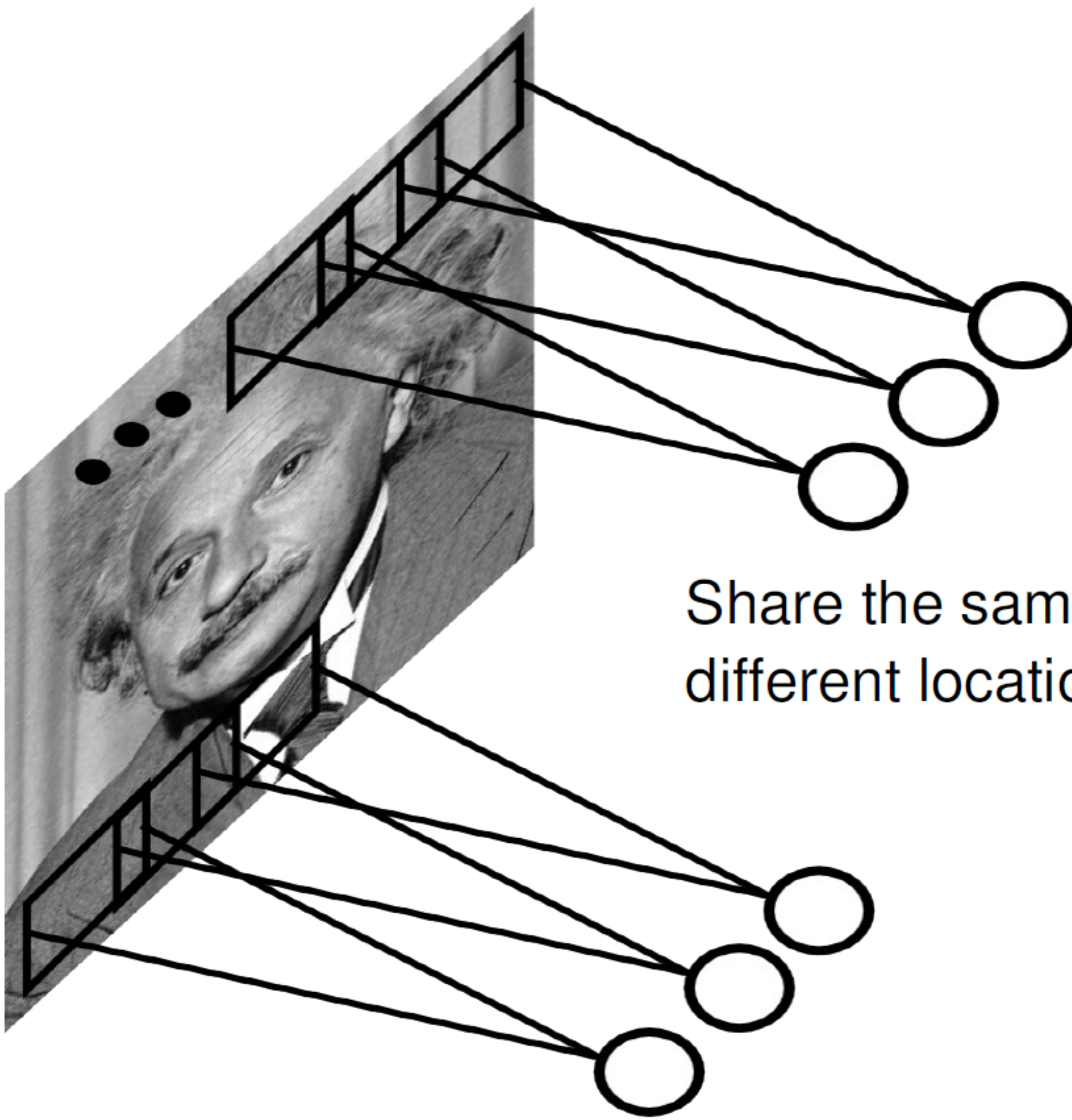
"Patch-based" representation



Still too many parameters!

Credit: M. A. Ranzato

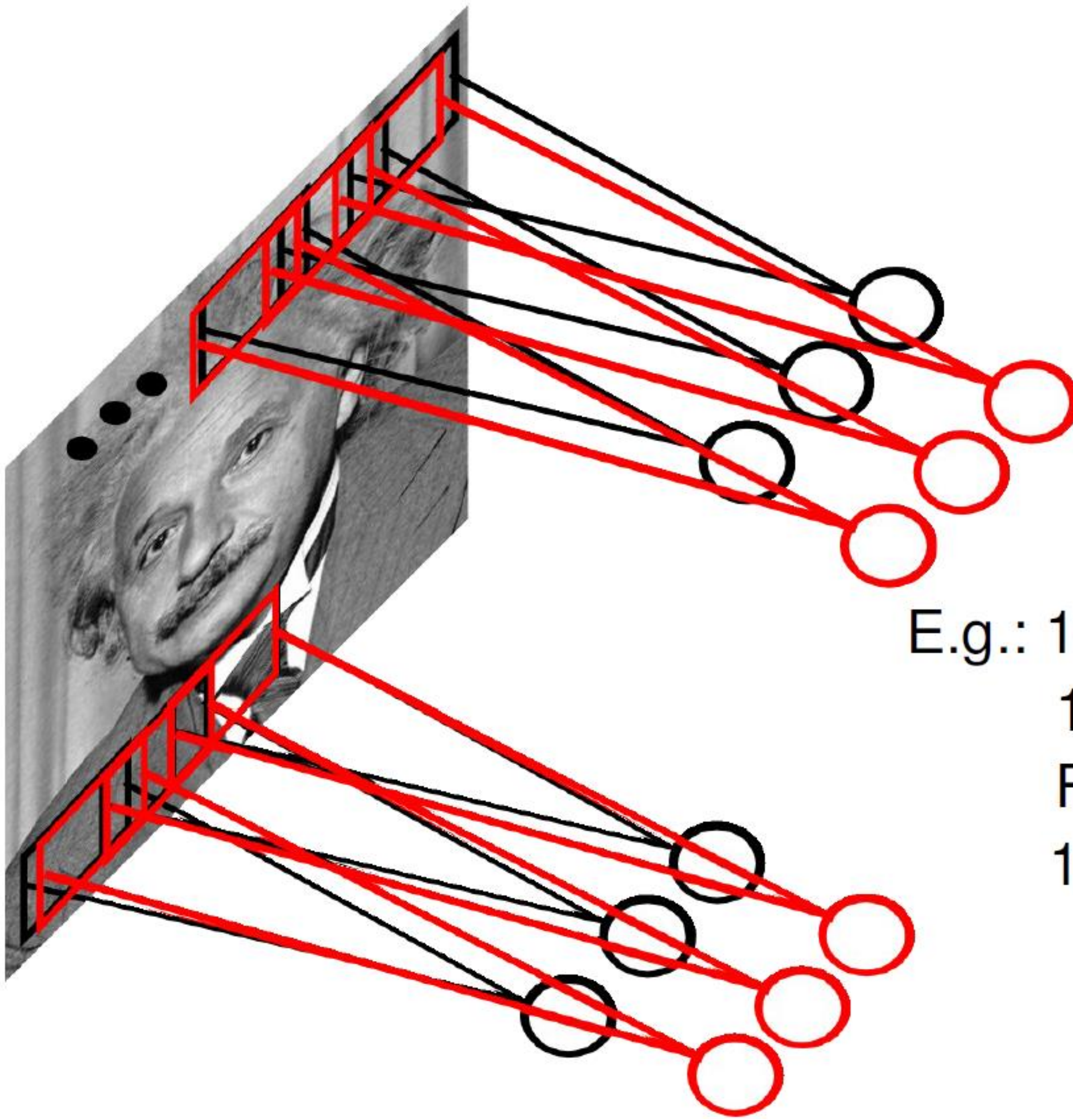
Motivation: Images as a composition of local parts
Convolution example



Share the same parameters across different locations:

Credit: M. A. Ranzato

Motivation: Images as a composition of local parts
Convolution example

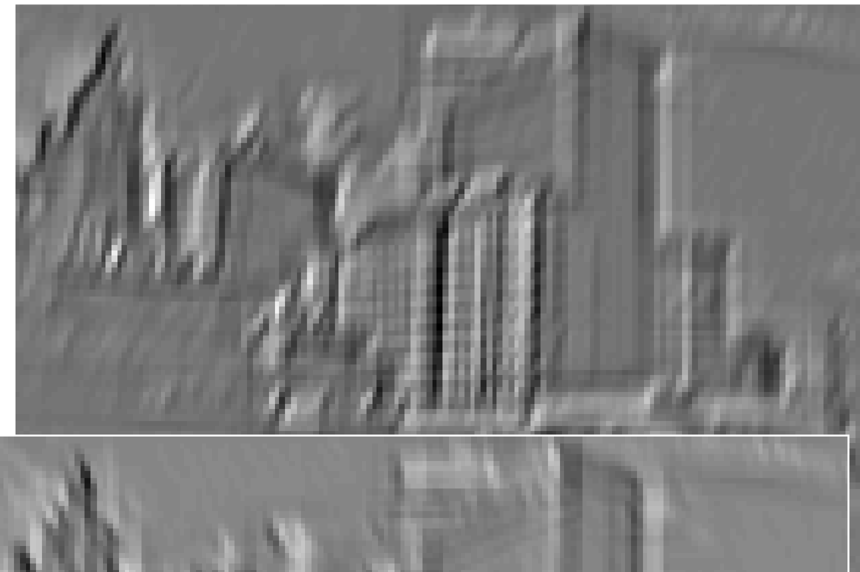


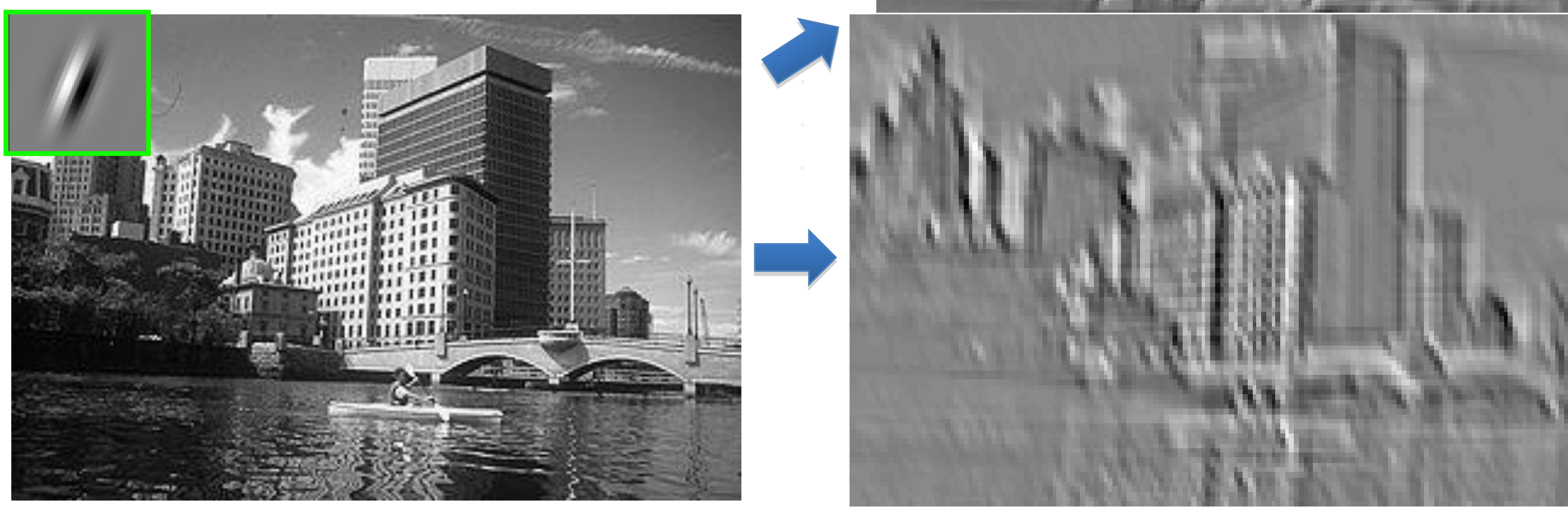
E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

Credit: M. A. Ranzato

Motivation: Images as a composition of local parts
Convolution example

- Why convolution?
  - Statistics of images look similar at different locations
  - Dependencies are very local
  - Filtering is an opteration with translation *equivariance*

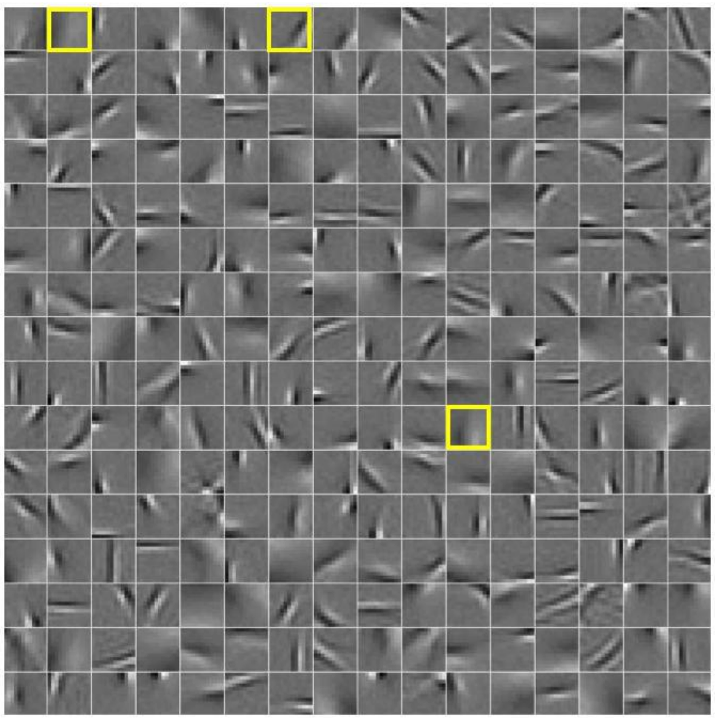




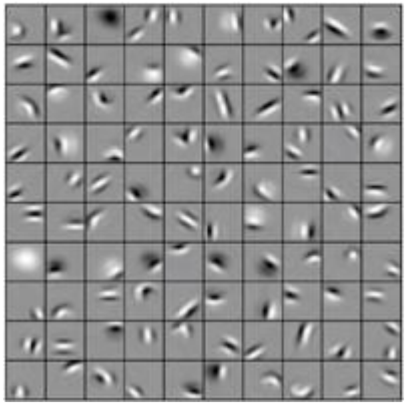
Credit: R. Fergus

Motivation: Images as a composition of local parts
Filtering example

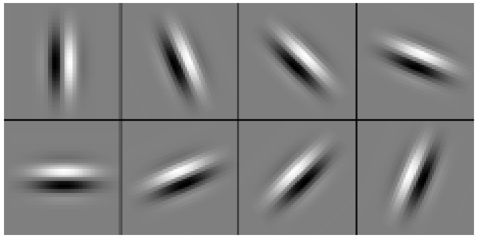
- Why translation *equivariance*?
  - Input translation leads to a translation of features
    - Fewer filters needed: no translated replications
    - But still need to cover orientation/frequency



Patch-based



Patch-based



Convolutional

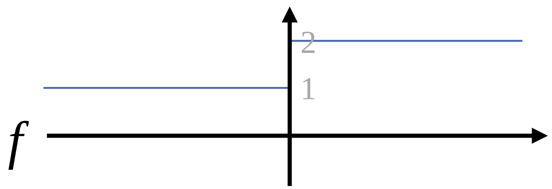
Credit: R. Fergus

Linear filtering

Further reading: Szeliski, Richard. *Computer Vision: Algorithms and Applications*. Springer, 2010, Chapter 3, Section 3.2

Basics: Smoothing via local averaging

Graphic depiction



Formalization

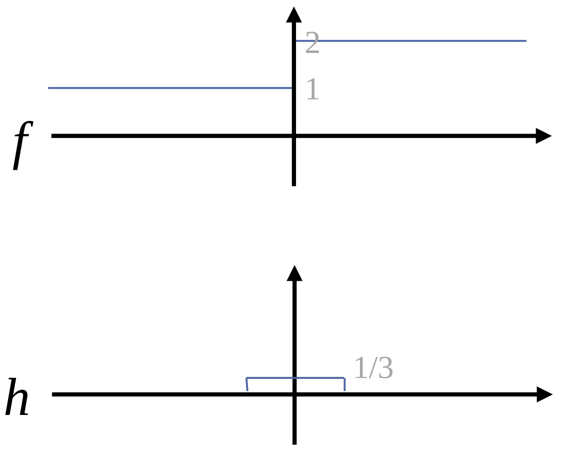
- We begin by considering a function:

$$f(x)$$

Credit: R. Wildes

Basics: Smoothing via local averaging

Graphic depiction



Formalization

- We begin by considering a function:

$$f(x)$$

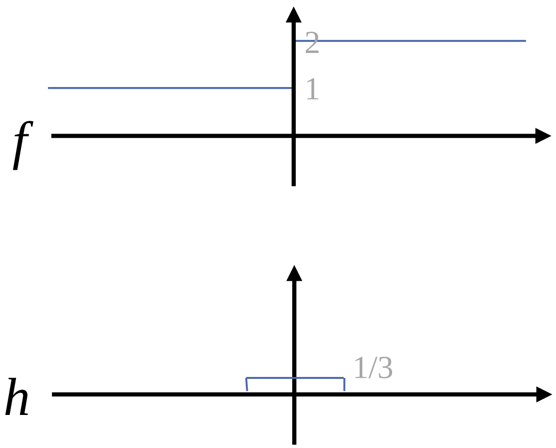
- And we multiply it with values of another function:

$$f(x)h$$

Credit: R. Wildes

Basics: Smoothing via local averaging

Graphic depiction



Formalization

- We begin by considering a function:

$$f(x)$$

- And we multiply it with values of another function:

$$f(x)h$$

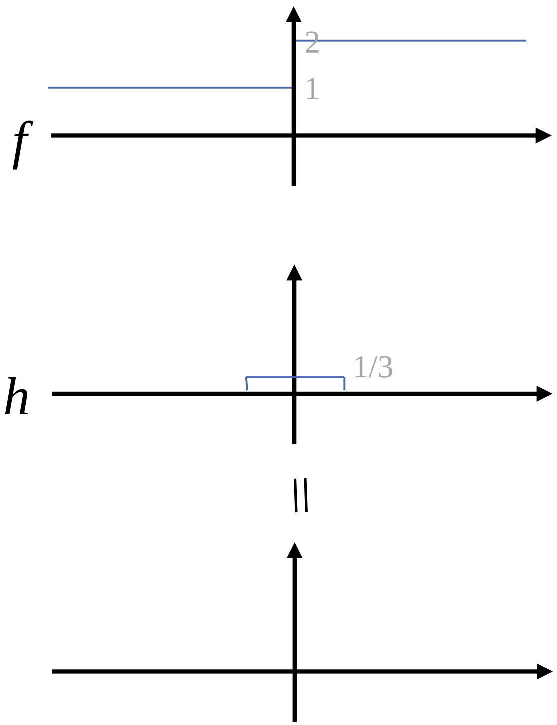
- But we do this at various offsets:

$$f(x-s)h(s)$$

Credit: R. Wildes

Basics: Smoothing via local averaging

Graphic depiction



Formalization

- We begin by considering a function:

$$f(x)$$

- And we multiply it with values of another function:

$$f(x)h$$

- But we do this at various offsets:

$$f(x-s)h(s)$$

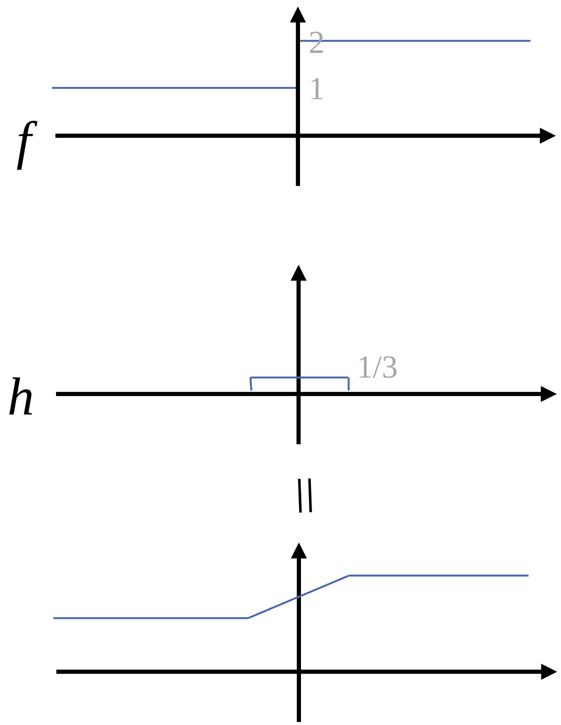
- and multiply by infinitesimal support elements:

$$f(x-s)h(s)ds$$

Credit: R. Wildes

Basics: Smoothing via local averaging

Graphic depiction



Formalization

- We begin by considering a function:

$$f(x)$$

- And we multiply it with values of another function:

$$f(x)h$$

- But we do this at various offsets:

$$f(x-s)h(s)$$

- and multiply by infinitesimal support elements:

$$f(x-s)h(s)ds$$

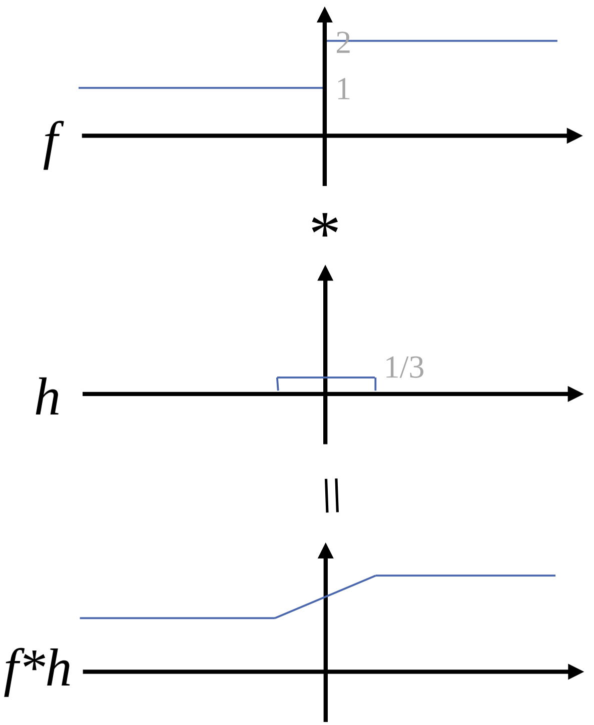
- Finally, we sum up (integrate):

$$\int f(x-s)h(s)ds$$

Credit: R. Wildes

Basics: Smoothing via local averaging

Graphic depiction



- We call this operation * a *convolution*

$$f*h = \int f(x-s)h(s)ds$$

Formalization

- We begin by considering a function:

$$f(x)$$

- And we multiply it with values of another function:

$$f(x)h$$

- But we do this at various offsets:

$$f(x-s)h(s)$$

- and multiply by infinitesimal support elements:

$$f(x-s)h(s)ds$$

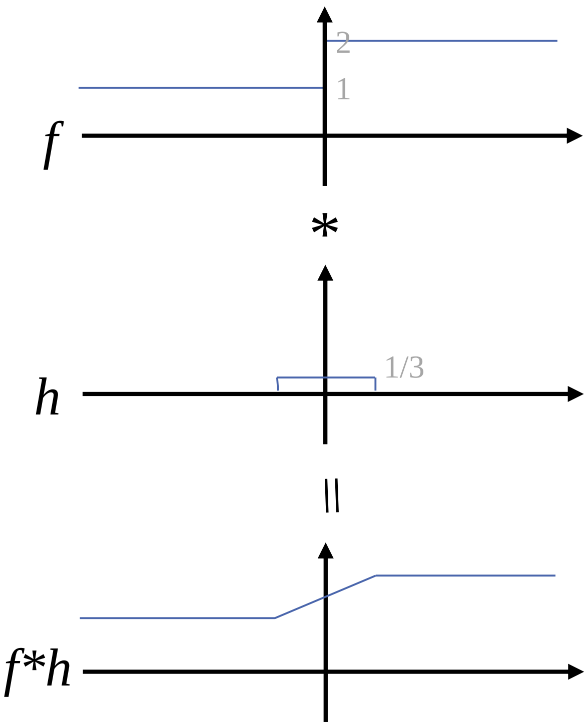
- Finally, we sum up (integrate):

$$\int f(x-s)h(s)ds$$

Credit: R. Wildes

Basics: Smoothing via local averaging

Graphic depiction



Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1		
	0		
	1		
-1	-1		
	0		
	1		
0	-1		
	0		
	1		
1	-1		
	0		
	1		
2	-1		
	0		
	1		

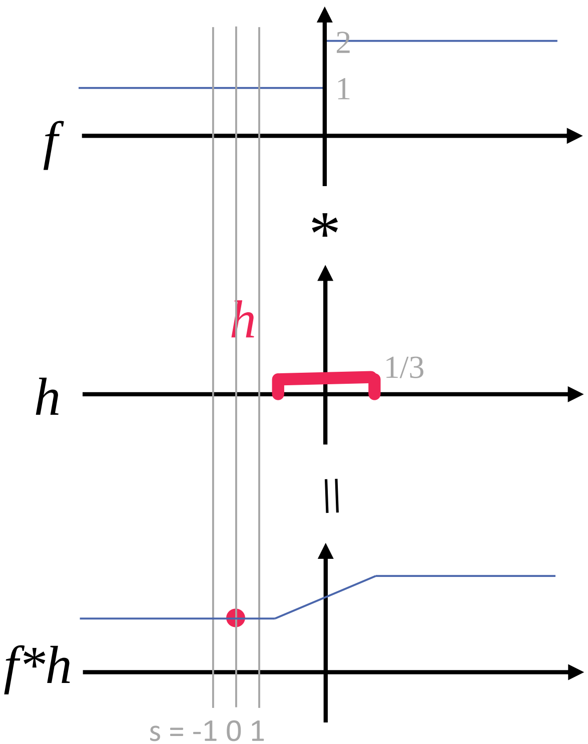
Formalization

$$\int f(x-s)h(s)ds$$

Credit: R. Wildes

Basics: Smoothing via local averaging

Graphic depiction



Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1/3)	
	0	(1)(1/3)	
	1	(1)(1/3)	1
-1	-1		
	0		
	1		
0	-1		
	0		
	1		
1	-1		
	0		
	1		
2	-1		
	0		
	1		

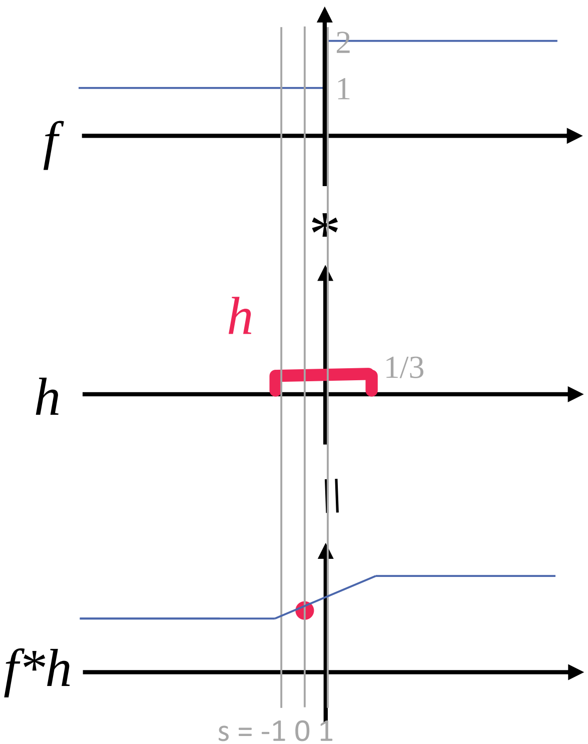
Formalization

$$\int f(x-s)h(s)ds$$

Credit: R. Wildes

Basics: Smoothing via local averaging

Graphic depiction



Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1/3)	
	0	(1)(1/3)	
	1	(1)(1/3)	1
-1	-1	(3/2)(1/3)	
	0	(1)(1/3)	
	1	(1)(1/3)	7.0/6.0
0	-1		
	0		
	1		
1	-1		
	0		
	1		
2	-1		
	0		
	1		

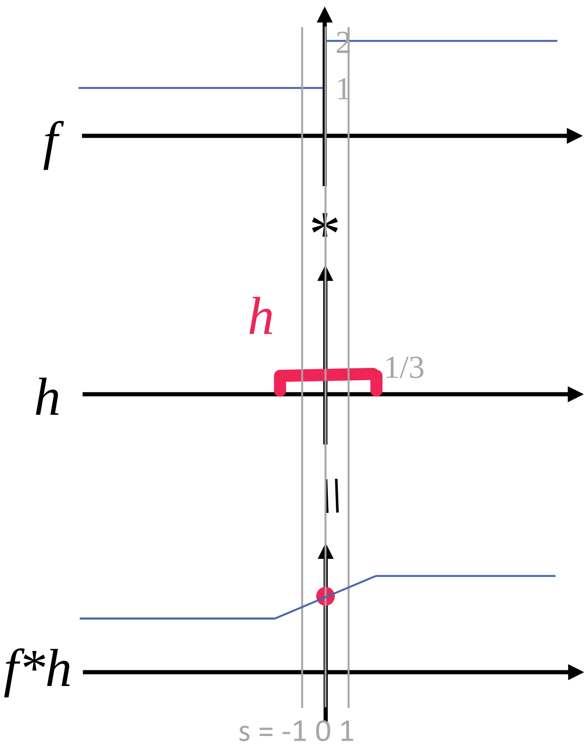
Formalization

$$\int f(x-s)h(s)ds$$

Credit: R. Wildes

Basics: Smoothing via local averaging

Graphic depiction



Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1/3)	
	0	(1)(1/3)	
	1	(1)(1/3)	1
-1	-1	(3/2)(1/3)	
	0	(1)(1/3)	
	1	(1)(1/3)	7.0/6.0
0	-1	(2)(1/3)	
	0	(3/2)(1/3)	
	1	(1)(1/3)	3.0/2.0
1	-1		
	0		
	1		
2	-1		
	0		
	1		

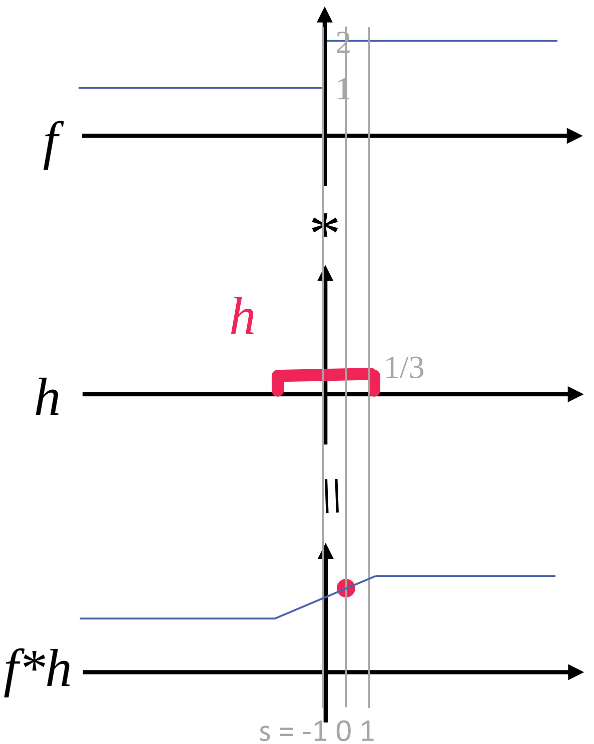
Formalization

$$\int f(x-s)h(s)ds$$

Credit: R. Wildes

Basics: Smoothing via local averaging

Graphic depiction



Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1/3)	
	0	(1)(1/3)	
	1	(1)(1/3)	1
-1	-1	(3/2)(1/3)	
	0	(1)(1/3)	
	1	(1)(1/3)	7.0/6.0
0	-1	(2)(1/3)	
	0	(3/2)(1/3)	
	1	(1)(1/3)	3.0/2.0
1	-1	(2)(1/3)	
	0	(2)(1/3)	
	1	(3/2)(1/3)	11.0/6.0
2	-1		
	0		
	1		

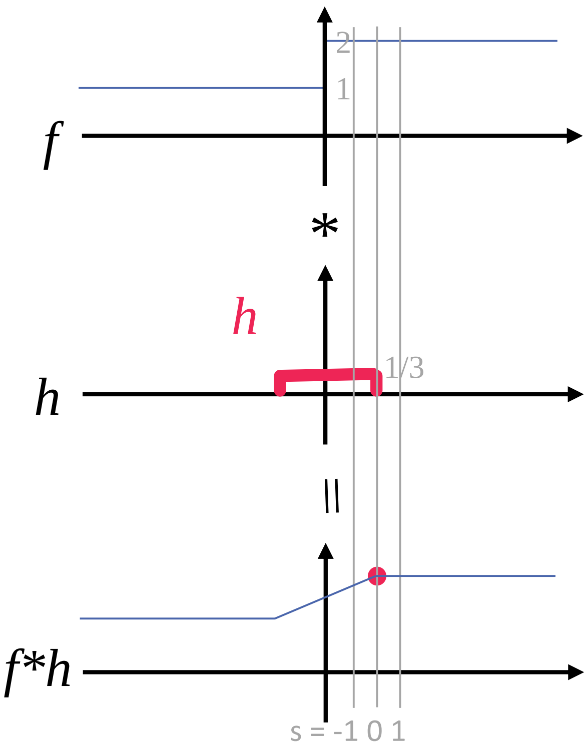
Formalization

$$\int f(x-s)h(s)ds$$

Credit: R. Wildes

Basics: Smoothing via local averaging

Graphic depiction



Formalization

$$\int f(x-s)h(s)ds$$

Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1/3)	
	0	(1)(1/3)	
	1	(1)(1/3)	1
-1	-1	(3/2)(1/3)	
	0	(1)(1/3)	
	1	(1)(1/3)	7.0/6.0
0	-1	(2)(1/3)	
	0	(3/2)(1/3)	
	1	(1)(1/3)	3.0/2.0
1	-1	(2)(1/3)	
	0	(2)(1/3)	
	1	(3/2)(1/3)	11.0/6.0
2	-1	(2)(1/3)	
	0	(2)(1/3)	
	1	(2)(1/3)	2

Credit: R. Wildes

Basics: 2D Convolution

Definition

- Consider a system that, given $f(x,y)$ as input, produces output

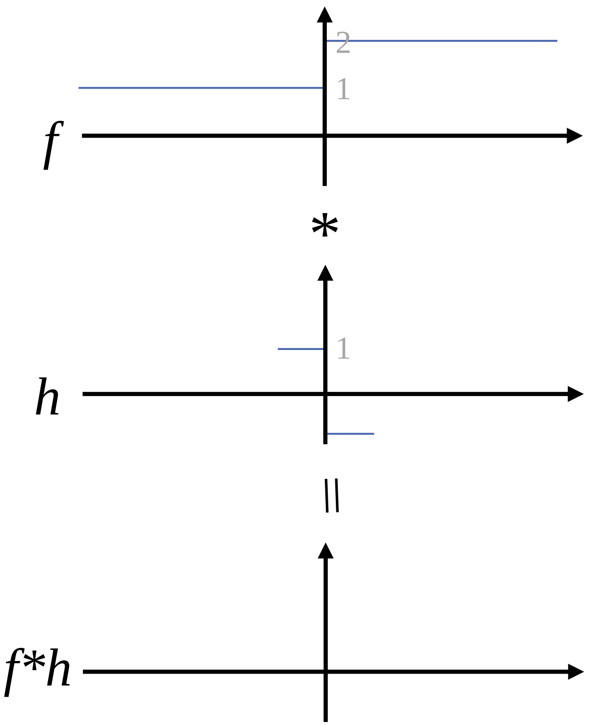
$$g(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-\xi, y-\eta)h(\xi,\eta)d\xi d\eta$$

- We say that g is the convolution of f and h, written as $g=f*h$.

Credit: R. Wildes

Basics: Another convolution example

Graphic depiction



Formalization

$$\int_{-\infty}^{\infty} f(x-s)h(s)ds$$

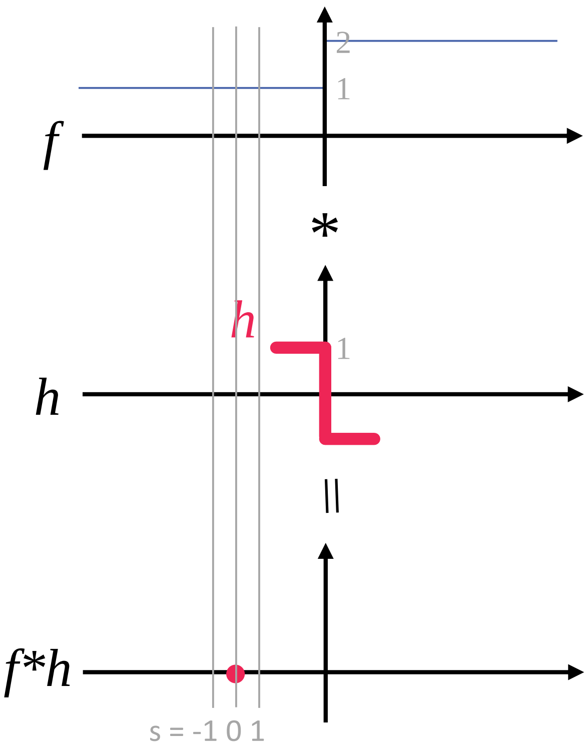
Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1		
	0		
	1		
-1	-1		
	0		
	1		
0	-1		
	0		
	1		
1	-1		
	0		
	1		
2	-1		
	0		
	1		

Credit: R. Wildes

Basics: Another convolution example

Graphic depiction



Formalization

$$\int_{-\infty}^{\infty} f(x-s)h(s)ds$$

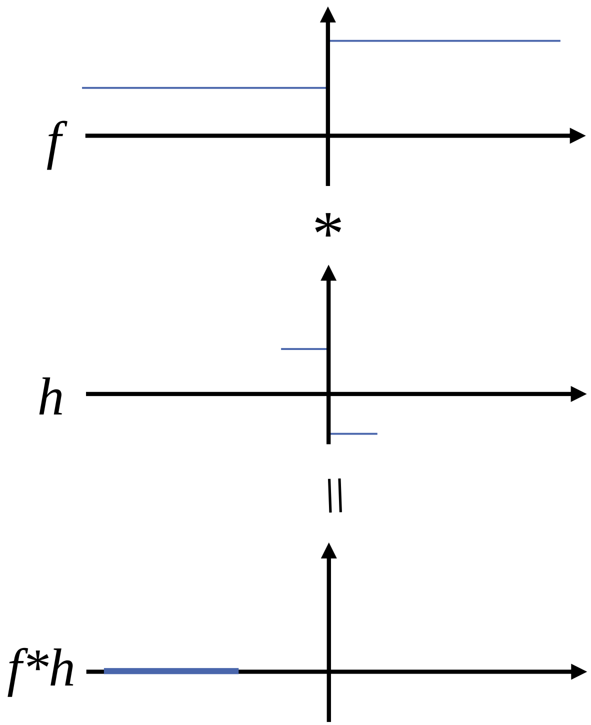
Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1)	
	0	(1)(0)	
	1	(1)(-1)	0
-1	-1		
	0		
	1		
0	-1		
	0		
	1		
1	-1		
	0		
	1		
2	-1		
	0		
	1		

Credit: R. Wildes

Basics: Another convolution example

Graphic depiction



Formalization

$$\int_{-\infty}^{\infty} f(x-s)h(s)ds$$

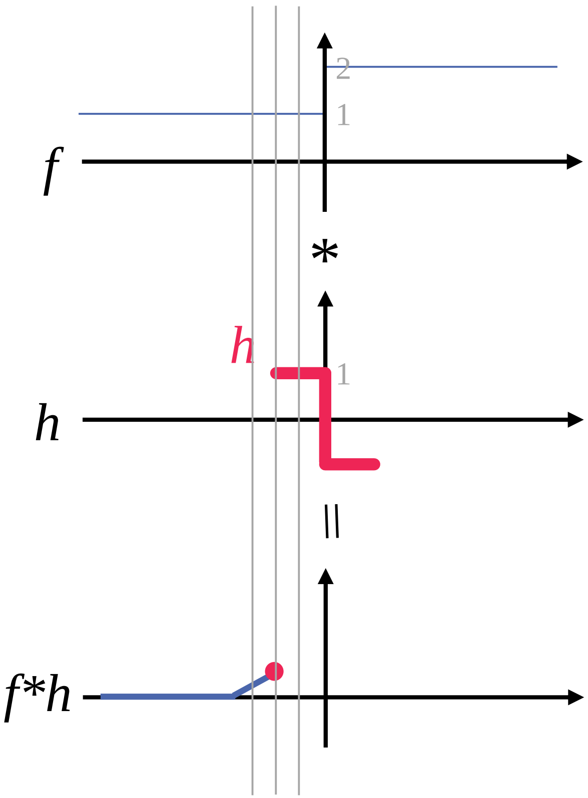
Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1)	
	0	(1)(0)	
	1	(1)(-1)	0
-1	-1		
	0		
	1		
0	-1		
	0		
	1		
1	-1		
	0		
	1		
2	-1		
	0		
	1		

Credit: R. Wildes

Basics: Another convolution example

Graphic depiction



Formalization

$$\int_{-\infty}^{\infty} f(x-s)h(s)ds$$

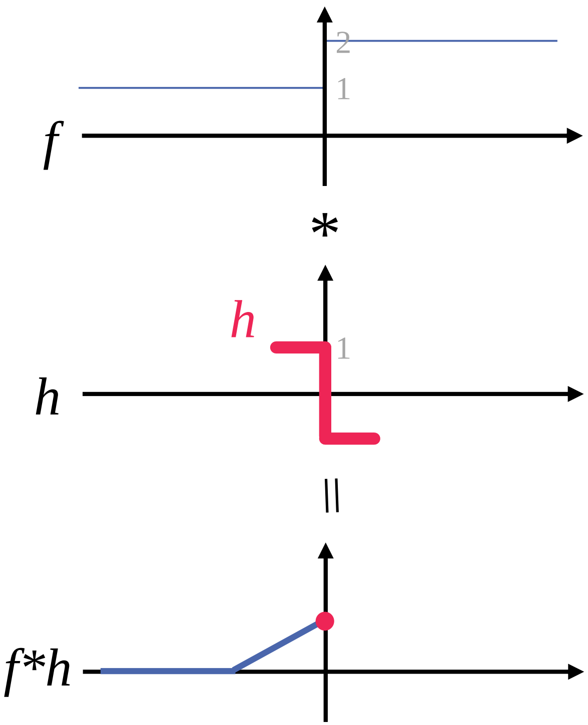
Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1)	
	0	(1)(0)	
	1	(1)(-1)	0
-1	-1	(3/2)(1)	
	0	(1)(0)	
	1	(1)(-1)	1/2
0	-1		
	0		
	1		
1	-1		
	0		
	1		
2	-1		
	0		
	1		

Credit: R. Wildes

Basics: Another convolution example

Graphic depiction



Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1)	
	0	(1)(0)	
	1	(1)(-1)	0
-1	-1	(3/2)(1)	
	0	(1)(0)	
	1	(1)(-1)	1/2
0	-1	(2)(1)	
	0	(3/2)(0)	
	1	(1)(-1)	1
1	-1		
	0		
	1		
2	-1		
	0		
	1		

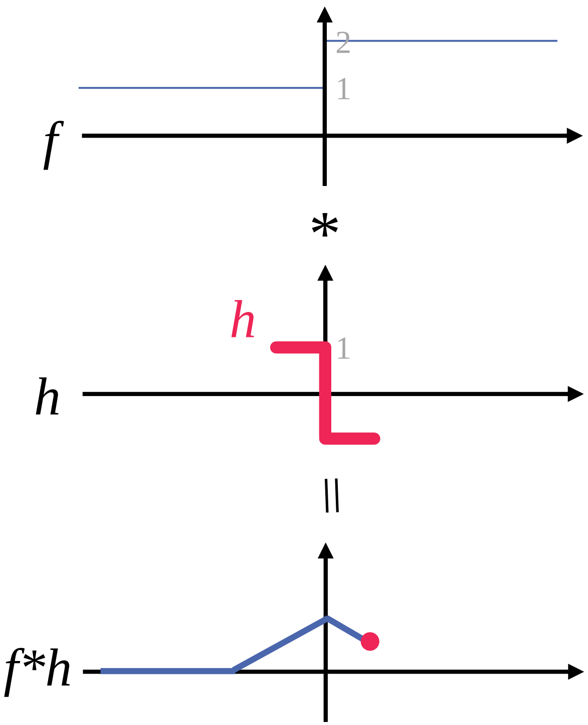
Formalization

$$\int_{-\infty}^{\infty} f(x-s)h(s)ds$$

Credit: R. Wildes

Basics: Another convolution example

Graphic depiction



Formalization

$$\int_{-\infty}^{\infty} f(x-s)h(s)ds$$

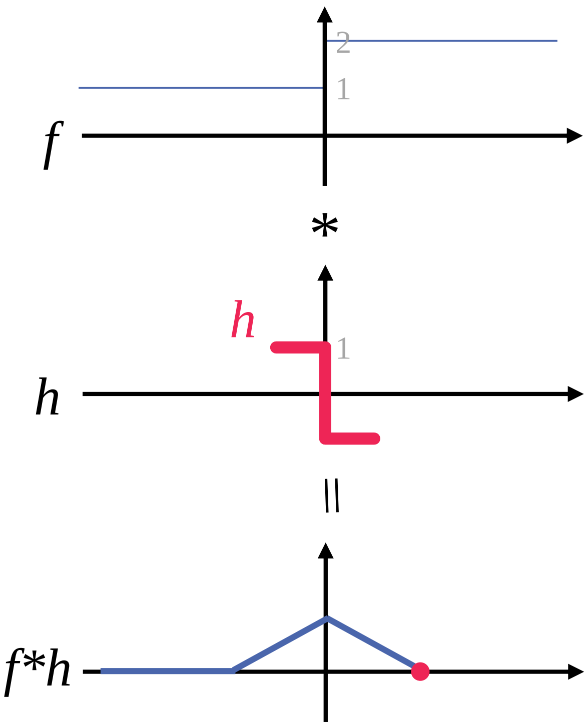
Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1)	
	0	(1)(0)	
	1	(1)(-1)	0
-1	-1	(3/2)(1)	
	0	(1)(0)	
	1	(1)(-1)	1/2
0	-1	(2)(1)	
	0	(3/2)(0)	
	1	(1)(-1)	1
1	-1	(2)(1)	
	0	(2)(0)	
	1	(3/2)(-1)	1/2
2	-1		
	0		
	1		

Credit: R. Wildes

Basics: Another convolution example

Graphic depiction



Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1)	
	0	(1)(0)	
	1	(1)(-1)	0
-1	-1	(3/2)(1)	
	0	(1)(0)	
	1	(1)(-1)	1/2
0	-1	(2)(1)	
	0	(3/2)(0)	
	1	(1)(-1)	1
1	-1	(2)(1)	
	0	(2)(0)	
	1	(3/2)(-1)	1/2
2	-1	(2)(1)	
	0	(2)(0)	
	1	(2)(-1)	0

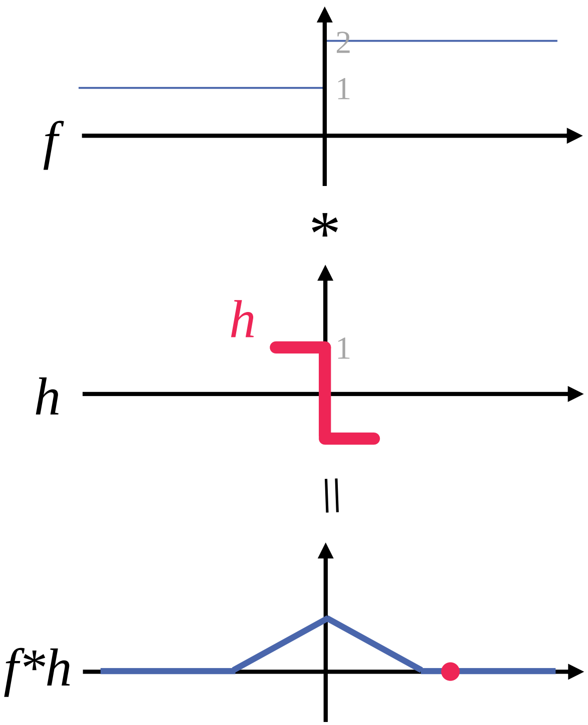
Formalization

$$\int_{-\infty}^{\infty} f(x-s)h(s)ds$$

Credit: R. Wildes

Basics: Another convolution example

Graphic depiction



Formalization

$$\int_{-\infty}^{\infty} f(x-s)h(s)ds$$

Numerical calculation

x	s	f(x-s)h(s)	+
-2	-1	(1)(1)	
	0	(1)(0)	
	1	(1)(-1)	0
-1	-1	(3/2)(1)	
	0	(1)(0)	
	1	(1)(-1)	1/2
0	-1	(2)(1)	
	0	(3/2)(0)	
	1	(1)(-1)	1
1	-1	(2)(1)	
	0	(2)(0)	
	1	(3/2)(-1)	1/2
2	-1	(2)(1)	
	0	(2)(0)	
	1	(2)(-1)	0

Credit: R. Wildes

Basics: Convolution

Definition

- Consider a system that, given $f(x,y)$ as input, produces output

$$g(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-\xi, y-\eta)h(\xi,\eta)d\xi d\eta$$

- We say that g is the convolution of f and h, written as $g=f*h$.

Convolution is linear

- Applying the system to $(a\ f1(x,y)\ +\ b\ f2(x,y))$ yields $(a\ g1(x,y)\ +\ b\ g2(x,y))$.
- Follows from rule for integrating the product of a constant and a function
- and the rule for integrating the sum of two functions.

$$\int\left[a\alpha(\xi)+b\beta(\xi)\right]d\xi = a\int\alpha(\xi)d\xi + b\int\beta(\xi)d\xi$$

Credit: R. Wildes

Basics: Convolution

Definition

- Consider a system that, given $f(x,y)$ as input, produces output

$$g(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-\xi, y-\eta)h(\xi,\eta)d\xi d\eta$$

- We say that g is the convolution of f and h, written as $g=f*h$.

Convolution is linear

- Applying the system to $(a\, f1(x,y) + b\, f2(x,y))$ yields $(a\, g1(x,y) + b\, g2(x,y))$.
- Follows from rule for integrating the product of a constant and a function
- and the rule for integrating the sum of two functions.

Convolution is shift invariant

- Applying the system to $f(x\text{-}a,y\text{-}b)$ yields $g(x\text{-}a,y\text{-}b)$.
- Follows from the convolution integral being independent of (x,y)
- So a change of variables $(x,y) \rightarrow (x\text{-}a,y\text{-}b)=(x',y')$ just shifts the result.
- Another way to think of shift invariance is that the operation (e.g. *) "behaves the same everywhere"
- For images: The value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood

Credit: R. Wildes

Images as functions

- We can think of an image as a function, *f*, from R^2 to R:
  - $f(x, y)$ gives the intensity at position (x, y)
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    - f: $[a,b]$ x $[c,d]$ → $[0, 1.0]$

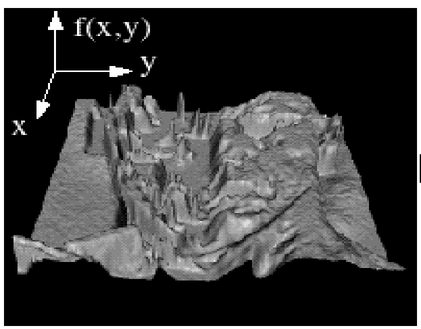
- A color image is just three functions pasted together. We can write this as a "vector-valued" function:

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

Credit: S. Seitz

Digital images

- In computer vision we operate on **digital** (**discrete**) images:
  - **Sample** the 2D space on a regular grid
  - **Quantize** each sample (round to nearest integer)
- Image thus represented as a matrix of integer values.

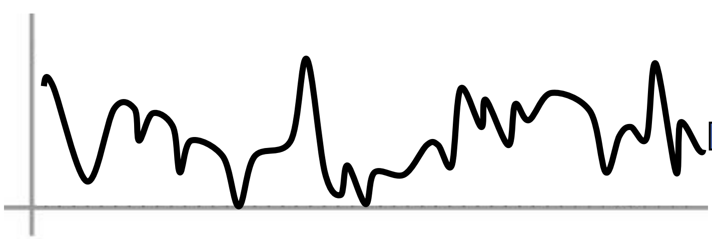


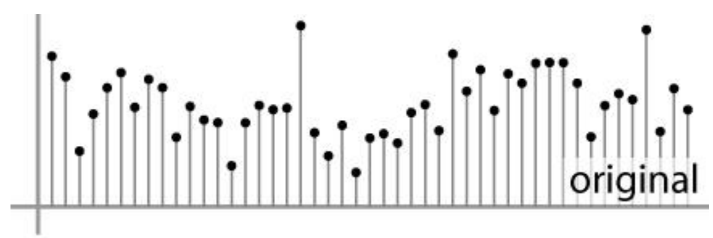
j

i

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

2D





1D

Credit: S. Seitz

Image filtering

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

10	5	3
4	5	1
1	1	7

Local image data

Some function $\rightarrow$

	7	

Modified image data

Credit: L. Zhang

Linear filtering

- One simple version: linear filtering
  (cross-correlation, convolution)
  - Replace each pixel by a linear combination (a weighted sum) of its neighbors
- The prescription for the linear combination is called the “kernel” (or “mask”, “filter”)

10	5	3
4	6	1
1	1	8

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	8	

Modified image data

Credit: L. Zhang

Convolution: The 2D case

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

Numerical calculation: Smoothing via local averaging

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

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

$g =$

	0								

Credit: S. Seitz

Convolution: The 2D case

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

Numerical calculation: Smoothing via local averaging

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$$f = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 0 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$g =$

	0								

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

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

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

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

$g =$

	0	10							

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

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

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

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

$g =$

	0	10	20						

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

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

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

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

$g =$

	0	10	20	30					

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

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

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$$f = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\ \hline 0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\ \hline 0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\ \hline 0 & 0 & 0 & 90 & 0 & 90 & 90 & 90 & 0 & 0 \\ \hline 0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

$$g = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline \; & \; & \; & \; & \; & \; & \; & \; & \; & \; \\ \hline & 0 & 10 & 20 & 30 & \boxed{30} & & & & \\ \hline & & & & & & & & & \\ \hline & & & & & & & & & \\ \hline & & & & & & & & & \\ \hline & & & & & & & & & \\ \hline & & & & & & & & & \\ \hline & & & & & & & & & \\ \hline & & & & & & & & & \\ \hline & & & & & & & & & \\ \hline \end{array}$$

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

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

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

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

$g =$

	0	10	20	30	30	30			

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

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

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

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

$g =$

	0	10	20	30	30	30			
				?					

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

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

$$g(x, y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

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

$g =$

	0	10	20	30	30	30			
				50					

Credit: S. Seitz

Convolution: The 2D case

Numerical calculation: Smoothing via local averaging

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

$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l)$$

$f =$

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

$g =$

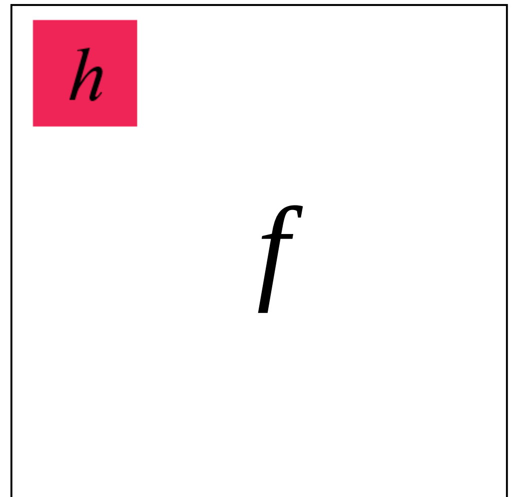
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Credit: S. Seitz

Convolution vs. correlation

Let f be the image and g be the kernel, the cross-correlation operation $\otimes$ is defined as

$$g(x,y)=\sum_{k,l} f(x+k,y+l)h(k,l)$$

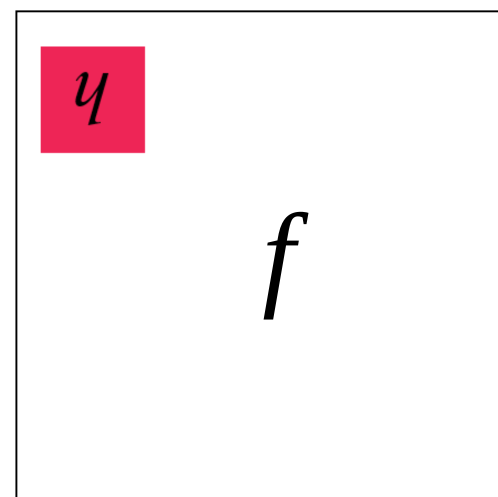


Note: We have defined convolution as $g=f*h$

$$g(x,y)=\sum_{k,l} f(x-k,y-l)h(k,l)$$

Filter kernel is “flipped”in both dimensions (bottom to top, right to left)

Then cross-correlation is applied

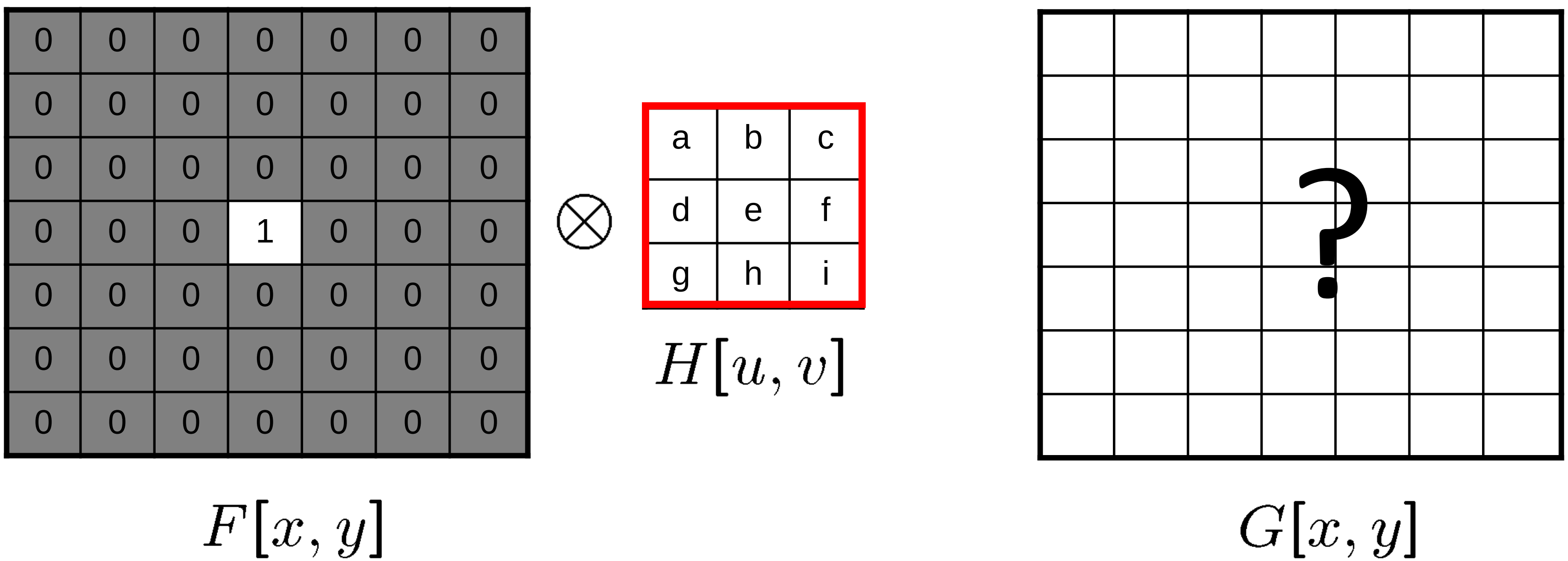


For a symmetric kernel, how will the outputs differ?

If the input is an impulse signal, how will the outputs differ?

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?



Credit: K. Grauman

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$F[x, y]$

$\otimes$

a	b	c
d	e	f
g	h	i

$H[u, v]$

0	0	0	0	0	0	0
0	0	0	0	0		

$G[x, y]$

$$G = H \otimes F$$

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$F[x, y]$

$\otimes$

a	b	c
d	e	f
g	h	i

$H[u, v]$

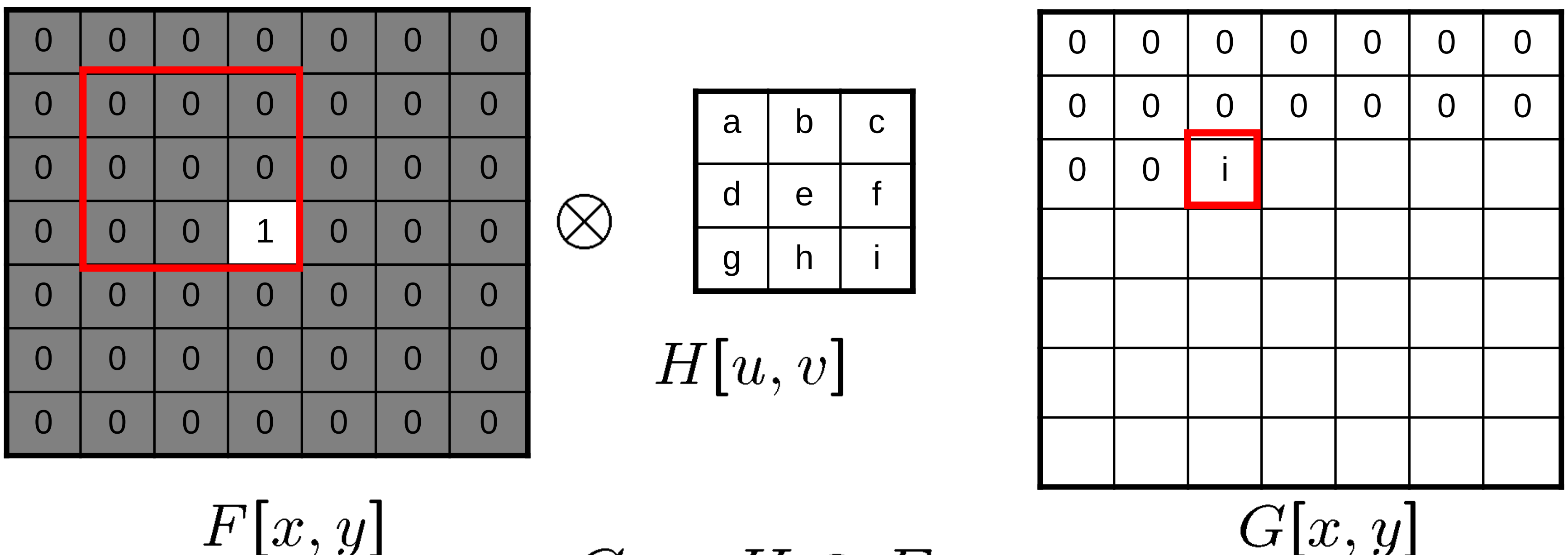
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0					

$G[x, y]$

$$G = H \otimes F$$

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?



$$G = H \otimes F$$

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$$F[x, y]$$

$$\otimes$$

a	b	c
d	e	f
g	h	i

$$H[u, v]$$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	i	$\boxed{h}$			

$$G[x, y]$$

$$G = H \otimes F$$

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$F[x, y]$

$\otimes$

a	b	c
d	e	f
g	h	i

$H[u, v]$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	i	h	g	0	0
0	0	f	e			

$G[x, y]$

$$G = H \otimes F$$

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$$F[x, y]$$

$$\otimes$$

a	b	c
d	e	f
g	h	i

$$H[u, v]$$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	i	h	g	0	0
0	0	f	e	d	0	0
0	0	c	b	a		

$$G[x, y]$$

$$G = H \otimes F$$

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$F[x, y]$

$\otimes$

a	b	c
d	e	f
g	h	i

$H[u, v]$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	i	h	g	0	0
0	0	f	e	d	0	0
0	0	c	b	a	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$G[x, y]$

$$G = H \otimes F$$

Filtering an impulse signal

What is the result of filtering the impulse signal (image) *F* with the arbitrary kernel *H*?

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$F[x,y]$

$\otimes$

a	b	c
d	e	f
g	h	i

$H[u,v]$

		i	h	g		
		f	e	d		
		c	b	a		

$G[x,y]$

$$G = H \otimes F$$

Filter output is reversed.

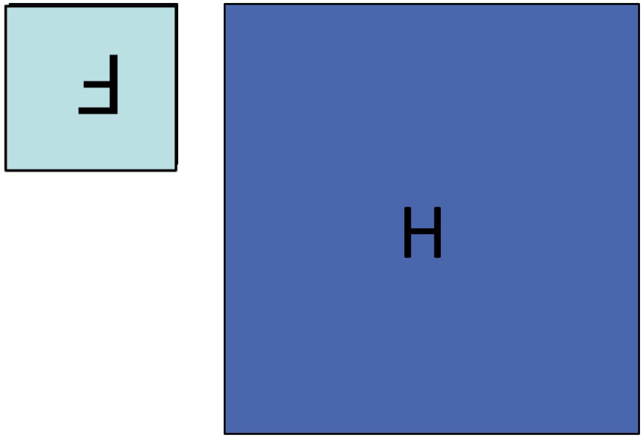
Convolution

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

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

$$G = H \star F$$

Notation for convolution operator



Credit: K. Grauman

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Credit: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Credit: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

?

Credit: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

Shifted left
By 1 pixel

Credit: D. Lowe

Practice with linear filters

Original

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

?

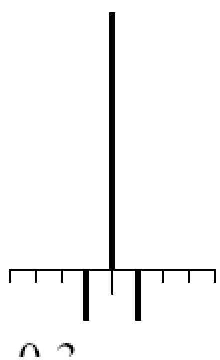
(Note that filter sums to 1)

Credit: D. Lowe

Practice with linear filters

Original

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



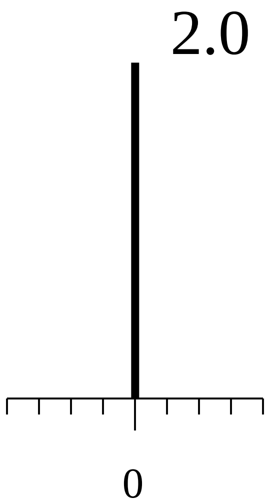
Sharpening filter

- Accentuates differences with local average

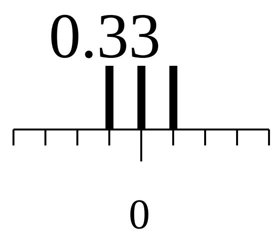
Credit: K. Grauman

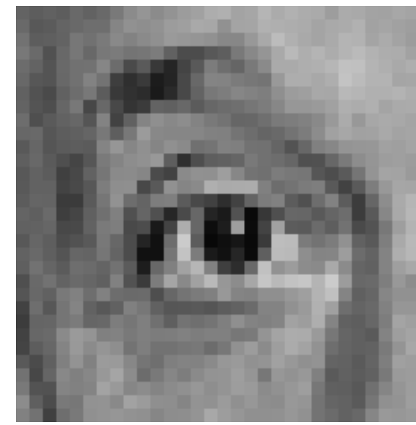
Sharpening

original



−

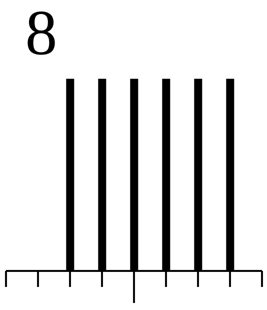


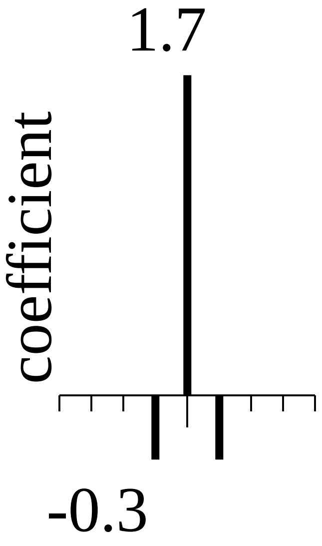


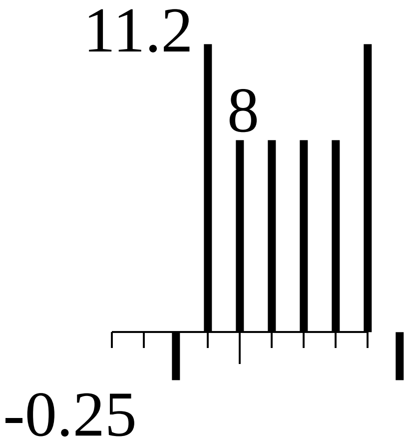
Sharpened original

Slide credit: Bill Freeman

Sharpening example



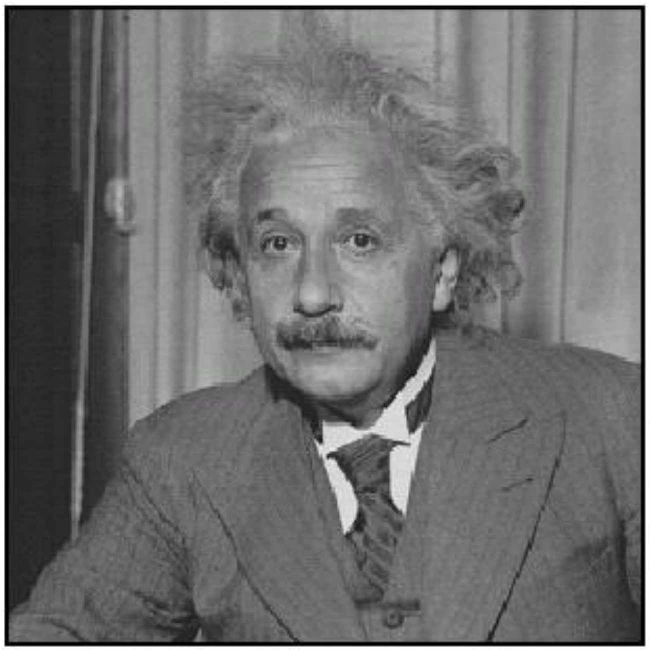




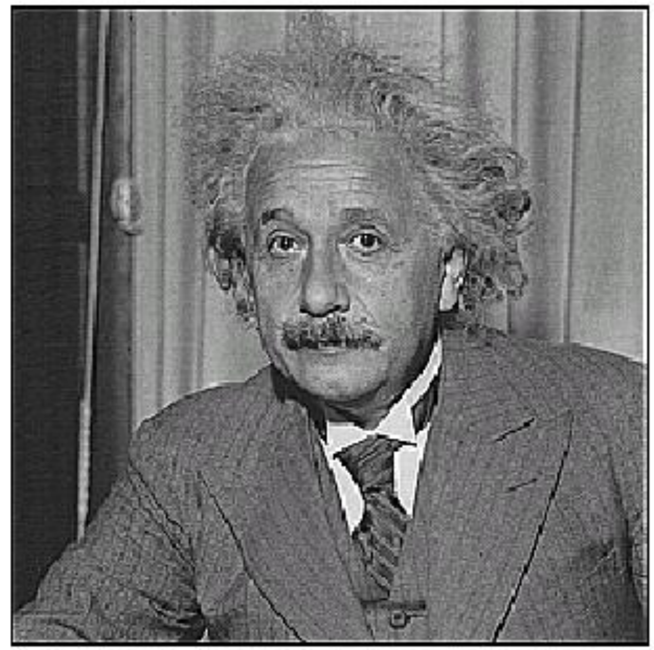
Sharpened
(differences are accentuated; constant areas are left untouched).

Slide credit: Bill Freeman

Sharpening



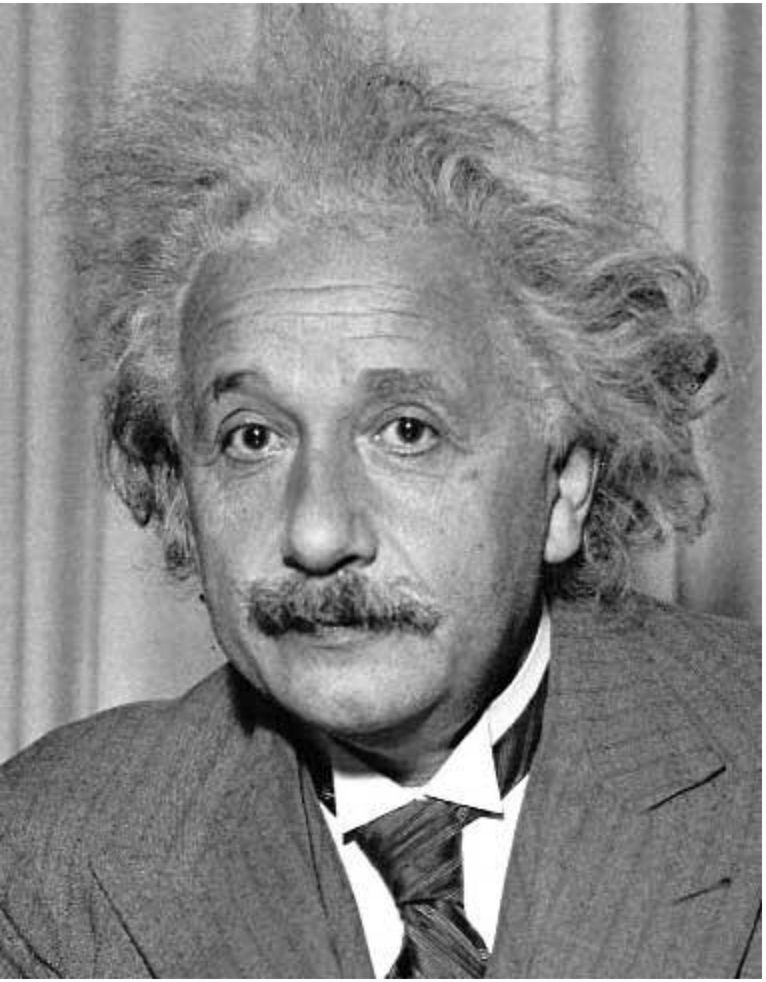
before



after

Credit: D. Lowe

Other filters



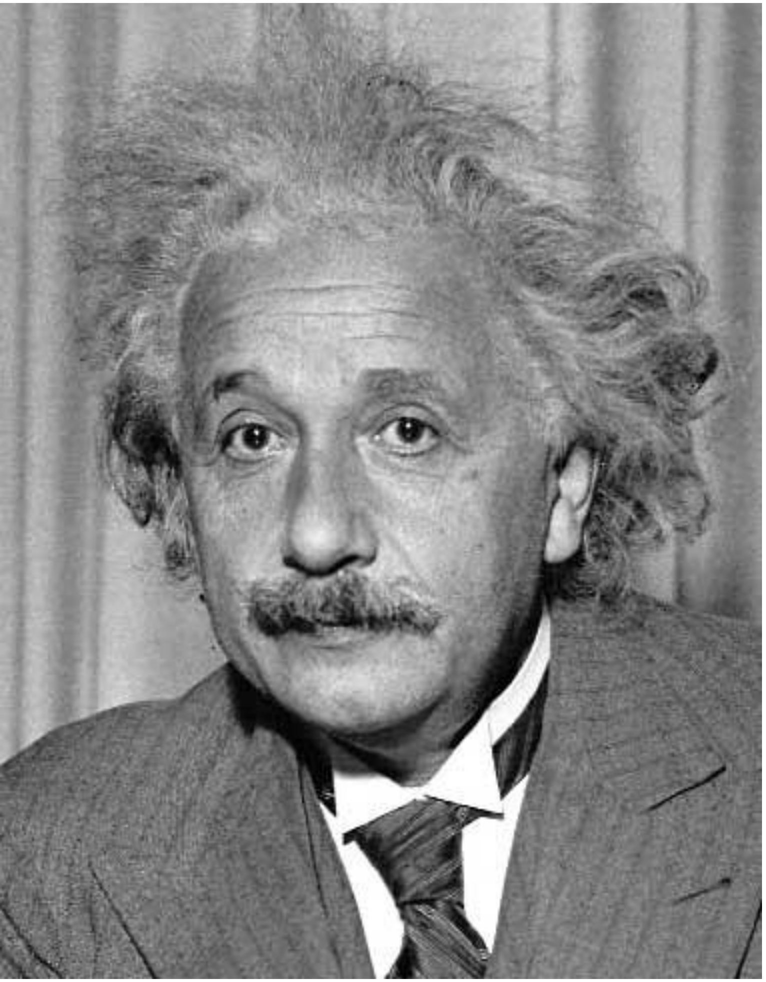
1	0	-1
2	0	-2
1	0	-1

Sobel

Vertical Edge
(absolute value)

Credit: D. Lowe

Other filters



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

Sobel

Horizontal Edge
(absolute value)

Credit: D. Lowe

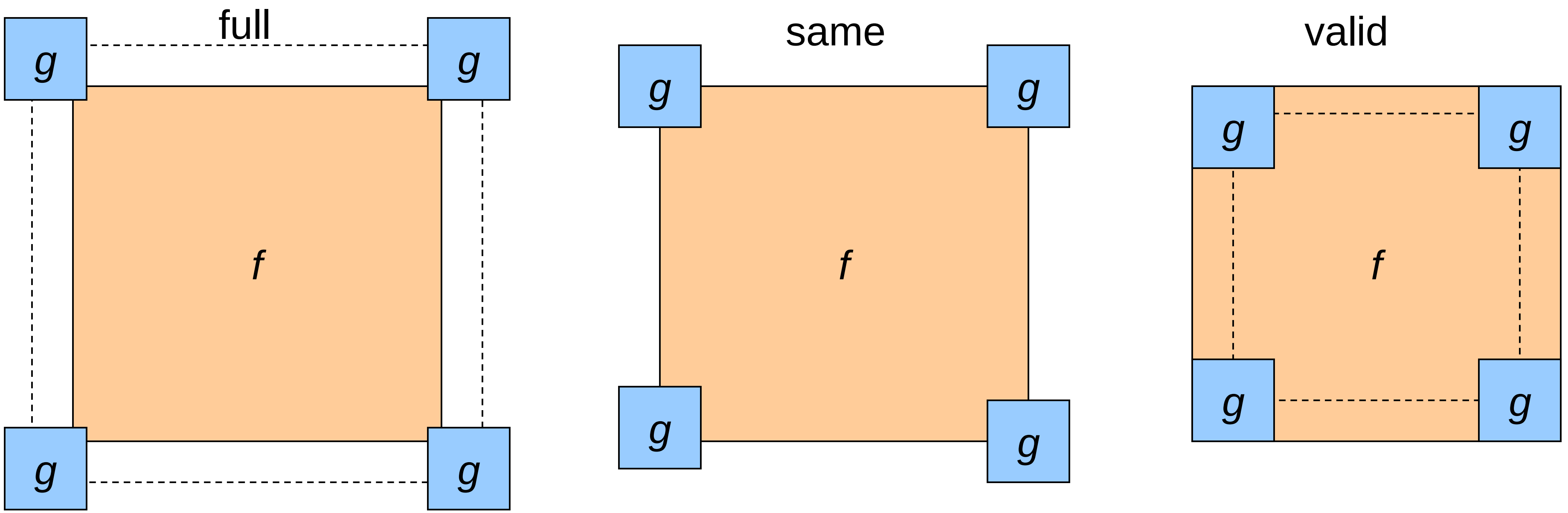
Border treatment

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Credit: S. Marschner

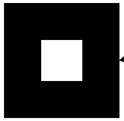
Border treatment

- What is the size of the output?
  - *shape* = ‘full’: output size is sum of sizes of f and g
  - *shape* = ‘same’: output size is same as f
  - *shape* = ‘valid’: output size is difference of sizes of f and g



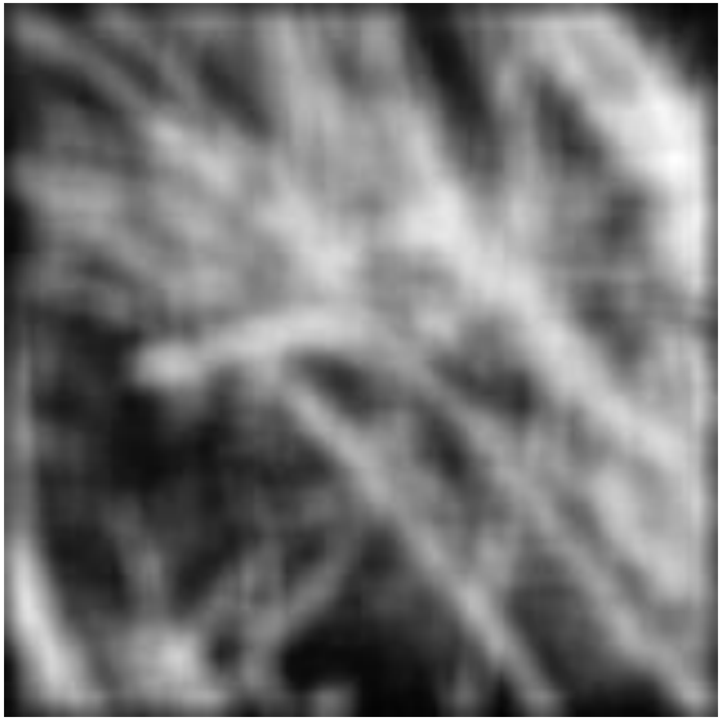
Credit: S. Lazebnik

Smoothing by averaging



depicts box filter:
white = high value, black = low value

original



filtered

Credit: D. Forsyth

Important filter: Gaussian

- What if we want nearest neighboring pixels to have the most influence on the output?

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

$F[x, y]$

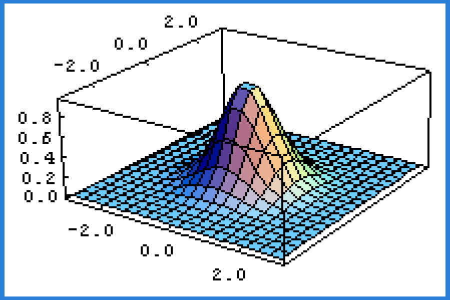
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

This kernel is an approximation of a Gaussian function:

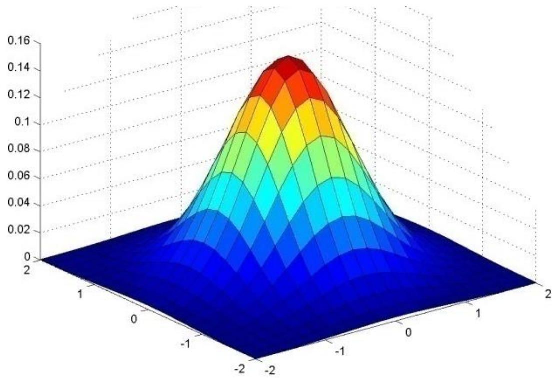
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

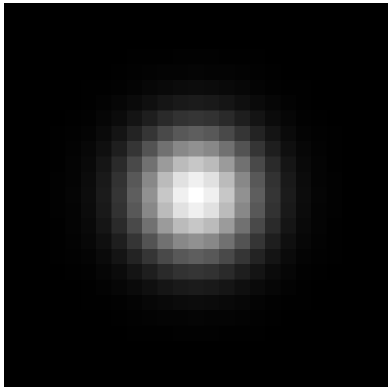


Credit: S. Seitz

Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness





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

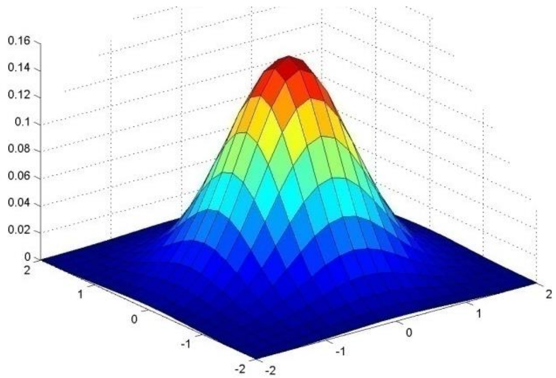
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

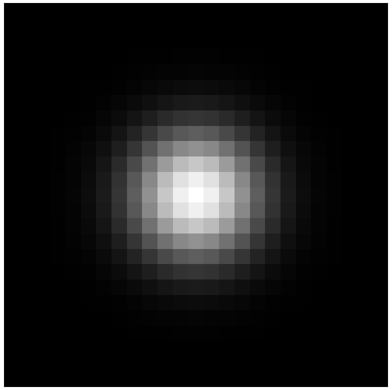
Credit: C. Rasmussen

Important filter: Gaussian

- **Gaussian filters have some interesting properties**

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





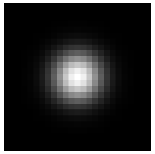
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

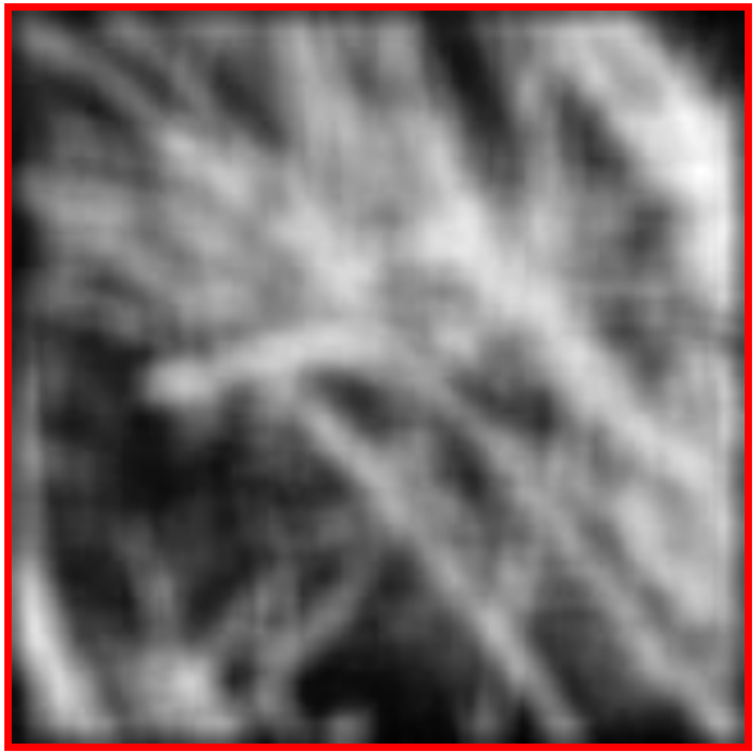
5 x 5, σ = 1

- Common in many natural models
- Smooth and symmetric function $\Rightarrow$ it has an infinite number of derivatives
- Fourier Transform of Gaussian is Gaussian (see later)
- Convolution of a Gaussian with itself is a Gaussian
- Gaussian is separable (e.g. 2D convolution can be performed by two 1-D convolutions)
- There is evidence that the human visual system performs Gaussian filtering

Smoothing with a Gaussian

- Remove “high-frequency” components from the image (low-pass filter)
  Images become more smooth

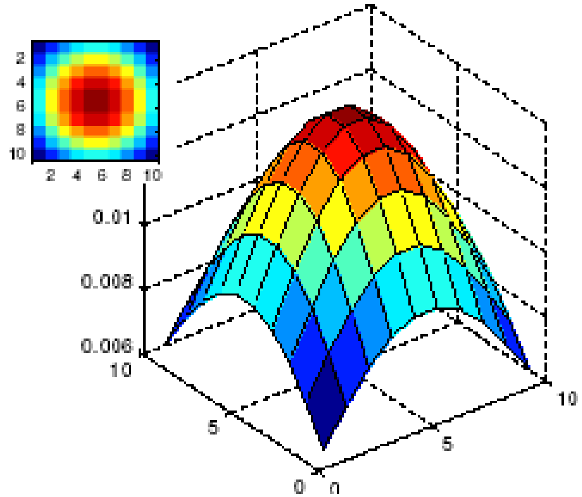




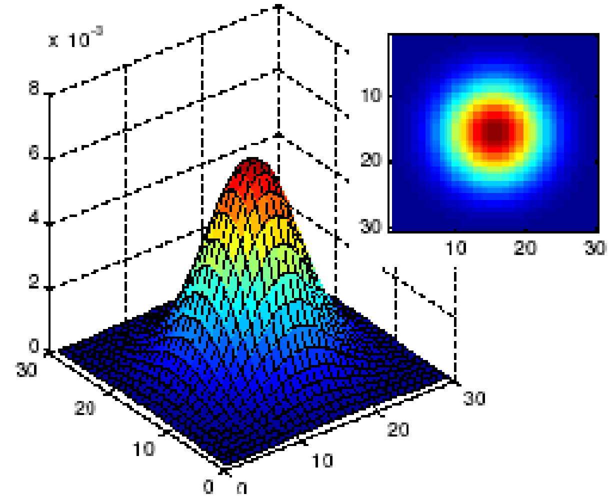
Credit: D. Forsyth

Gaussian filters

- What parameters matter here?
- **Size** of kernel or mask
  - Note, Gaussian function has infinite support, but discrete filters use finite kernels



σ = 5 with 10 x 10 kernel

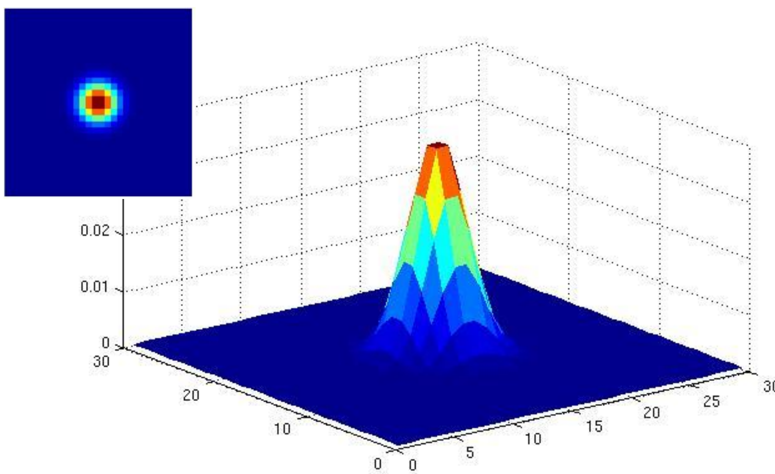


σ = 5 with 30 x 30 kernel

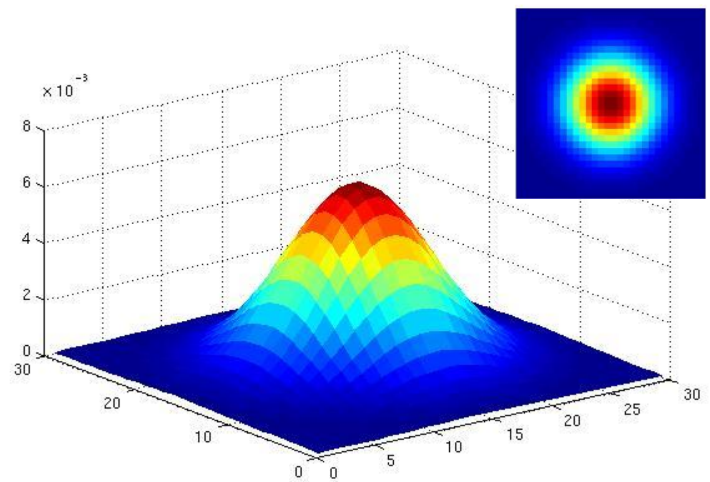
Credit: K. Grauman

Gaussian filters

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



σ = 2 with 30 x 30 kernel



σ = 5 with 30 x 30 kernel

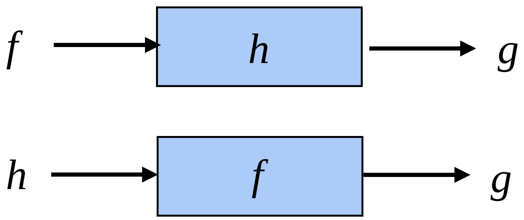
- Rule of thumb: filter width of about 6σ

Credit: K. Grauman

Basics: More facts about convolution

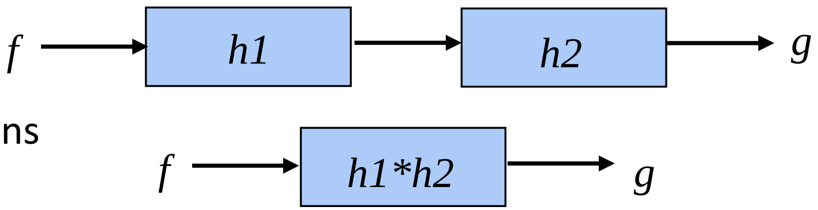
Convolution is commutative

- That is $f*h = h*f$
- Interchange of h and f possible
- Order does not care



Convolution is associative

- That is $(f*h1)*h2 = f*(h1*h2)$
- Can be exploited for efficient implementations

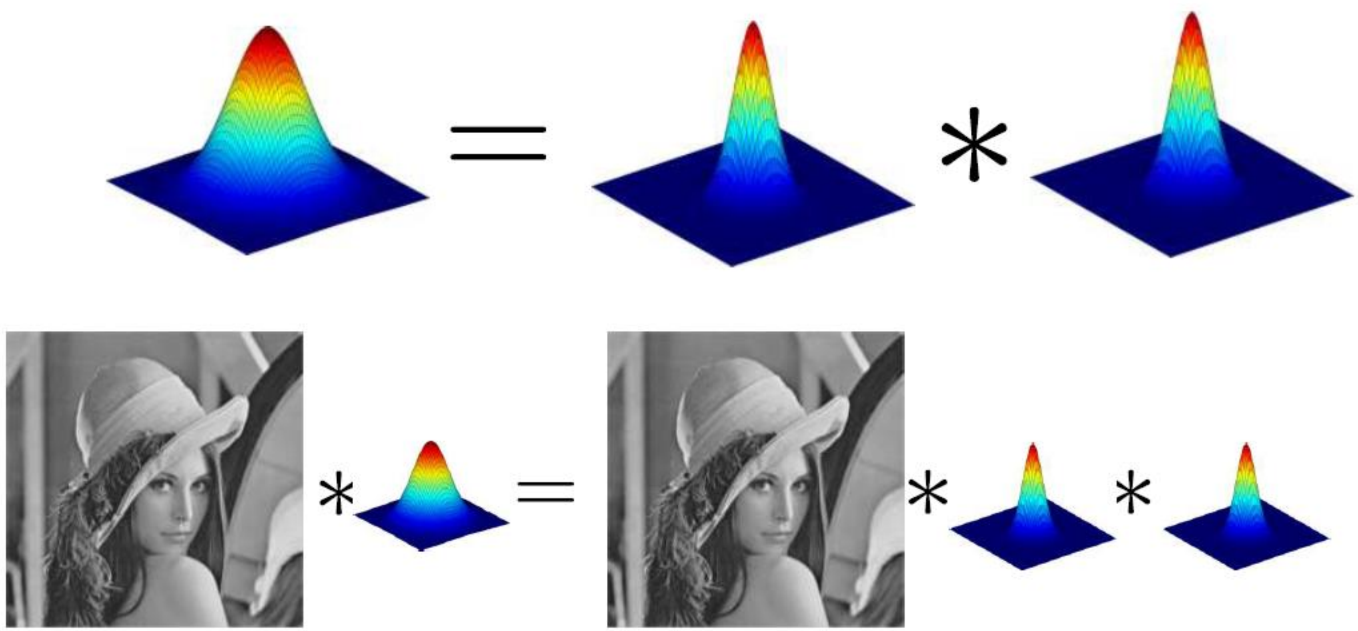


Credit: R. Wildes

Gaussian filters

Note: Convolution is associative: $(f*g)*h = f*(g*h)$

- Can be exploited for multi-scale processing and efficiency:
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
  - Efficiency: multiple smoothing with small-width kernel delivers same result as larger-width kernel

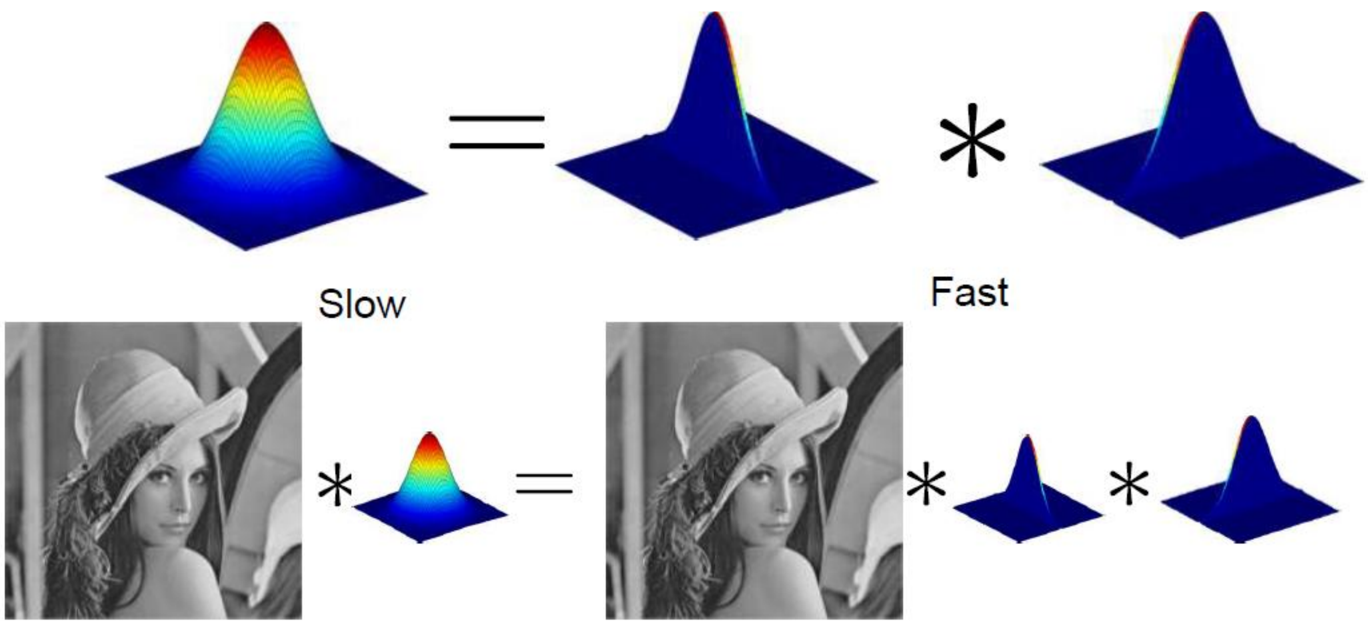


Credit: I. Kokkinos

Gaussian filters: Separability

Note: Convolution is associative: $(f*g)*h = f*(g*h)$

- Can be further exploited for efficiency:
  - Separable kernel factors into product of two 1D Gaussians
  - Efficiency: multiple smoothing with 1D filter delivers same result as with high dimensional filter



Credit: I. Kokkinos

Separability of the Gaussian filter for 2D

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

$$= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

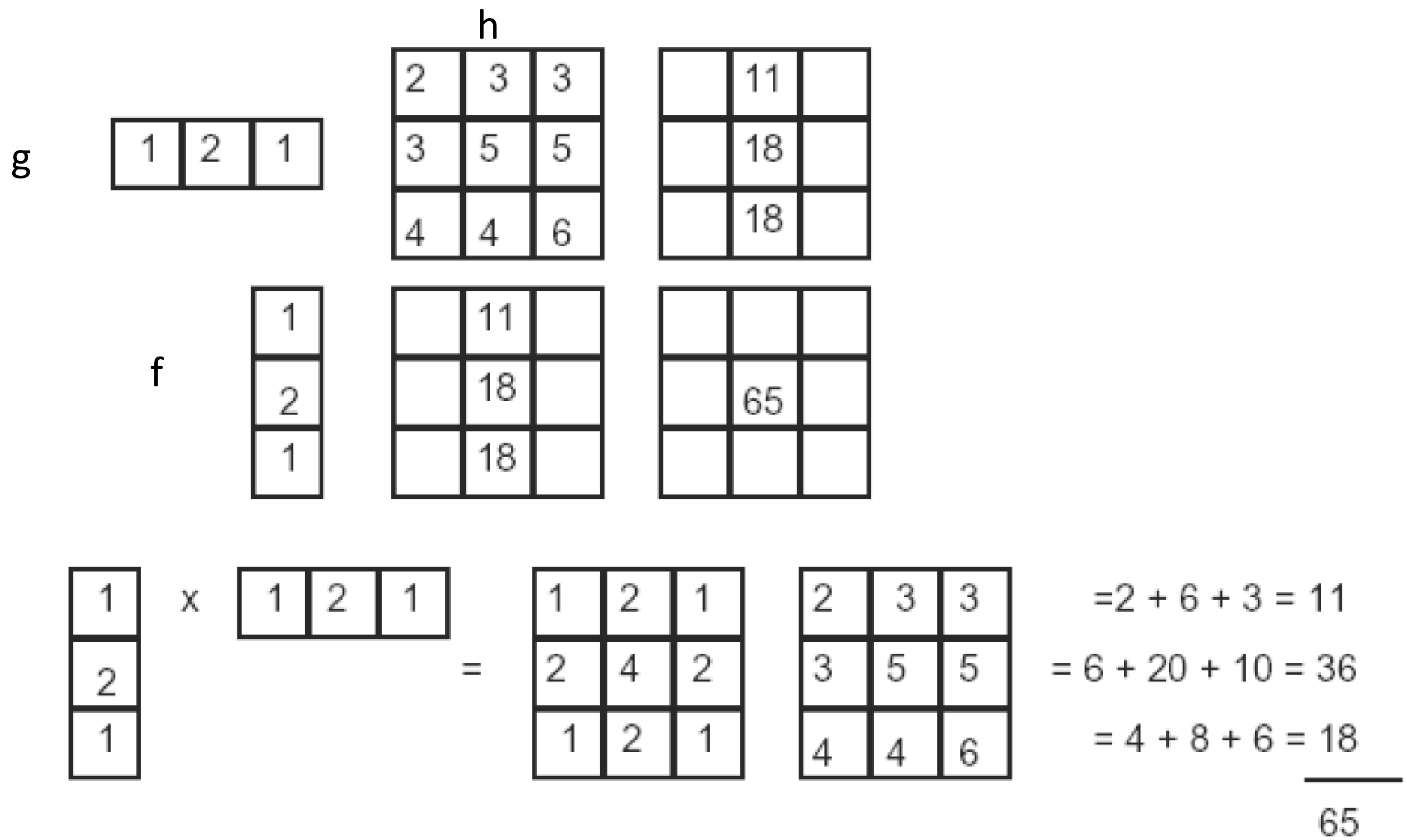
In this case, the two functions are the (identical) 1D Gaussian

Credit: D. Lowe

Separability: Numeric example for 2D

Note: Convolution is associative: $(f*g)*h = f*(g*h)$

- 2D filters are *separable* if they can be expressed as the outer product of two vectors. For example:

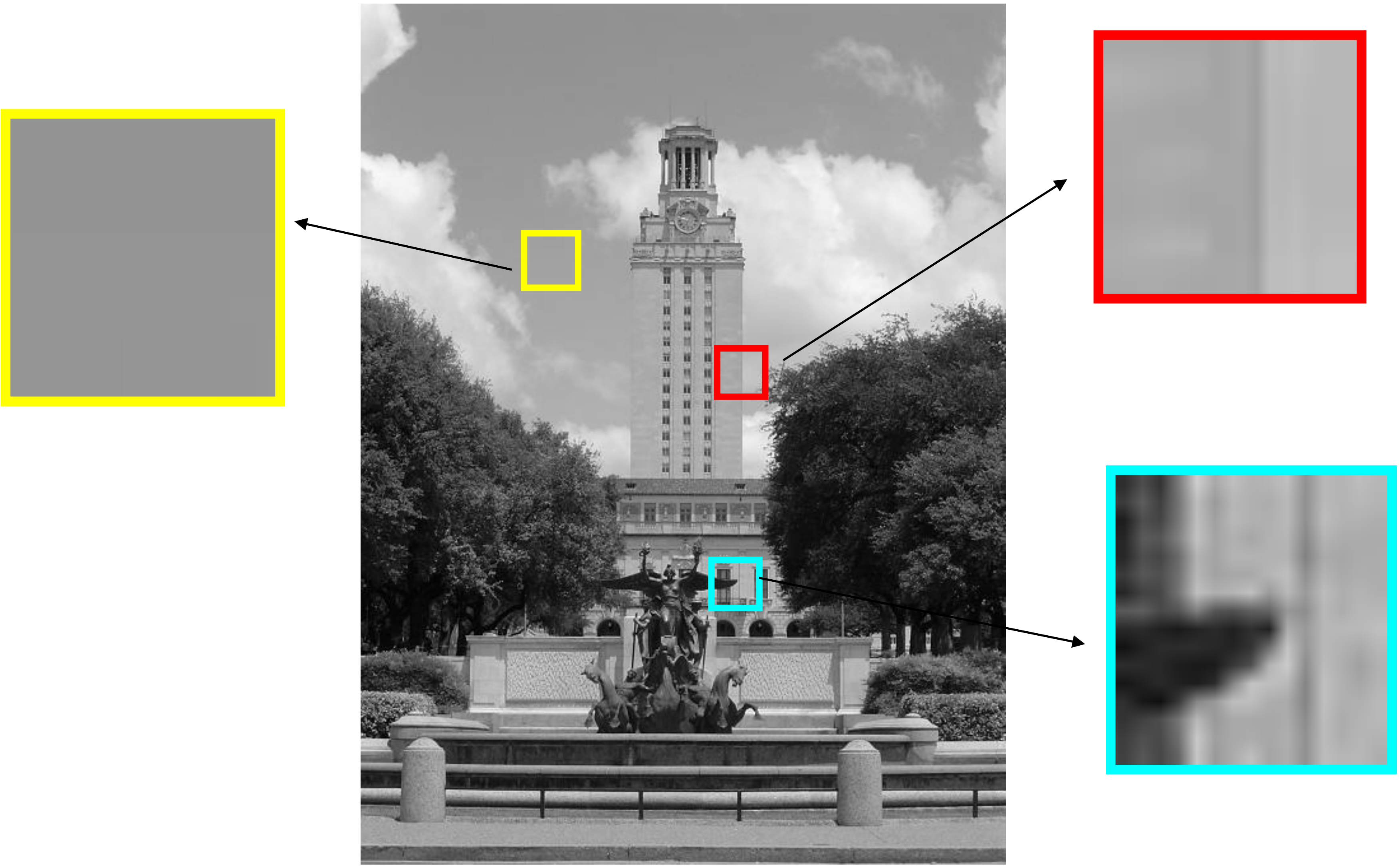


For MN image, PQ filter: 2D takes MNPQ add/times, while 1D takes MN(P + Q)

Credit: K. Grauman

More important filters: Edges/gradients and invariance

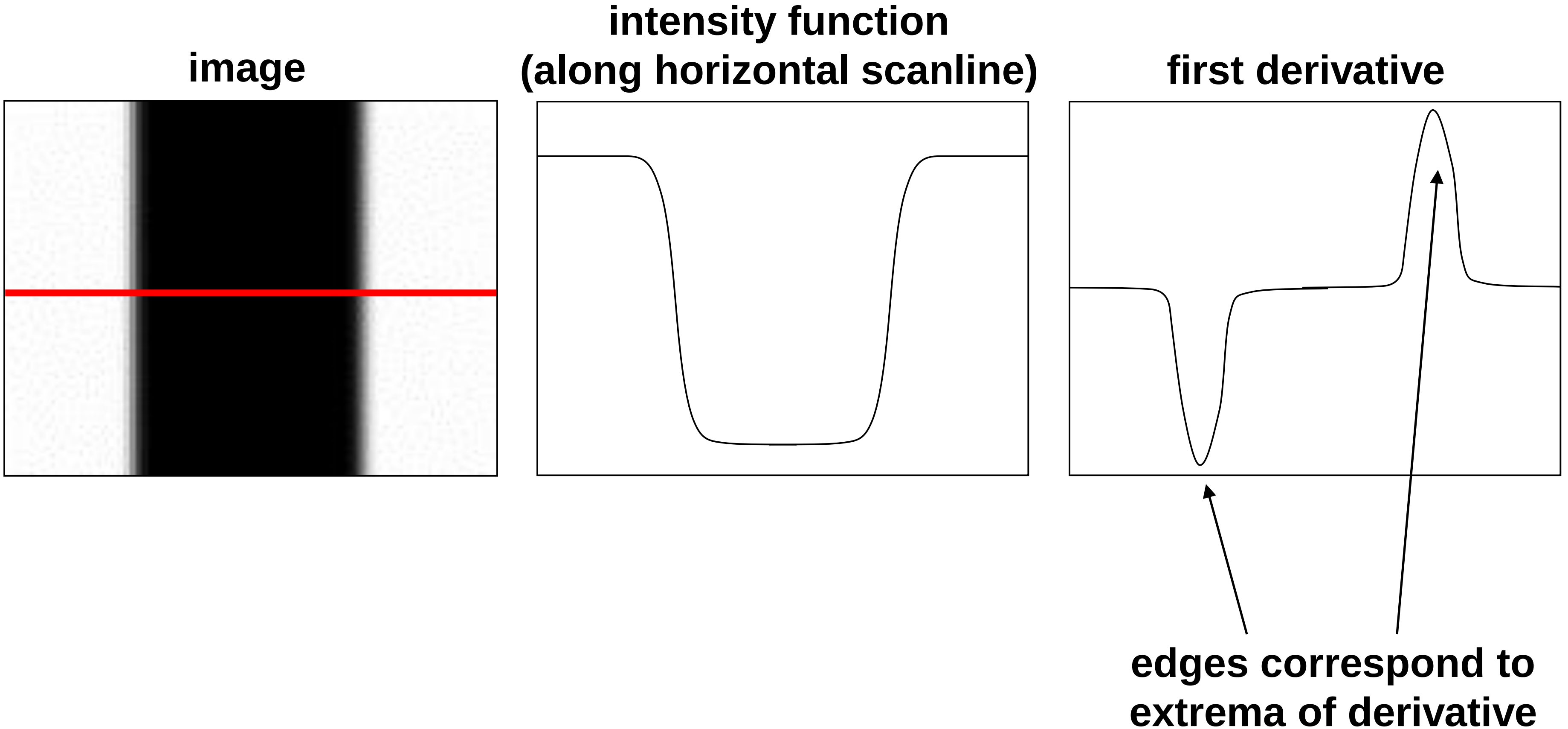
- Motivation: We want to represent distinctive parts of an image



Credit: K. Grauman

Derivatives and edges

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

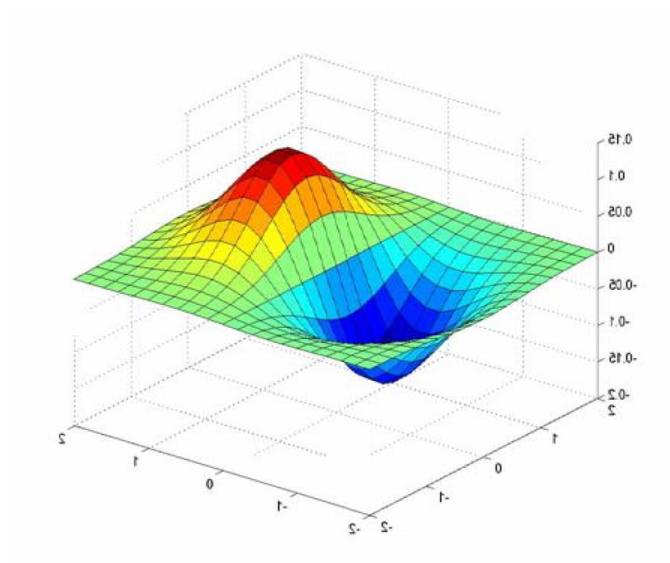


Credit: L. Lazebnik

More important filters: Derivative of Gaussian

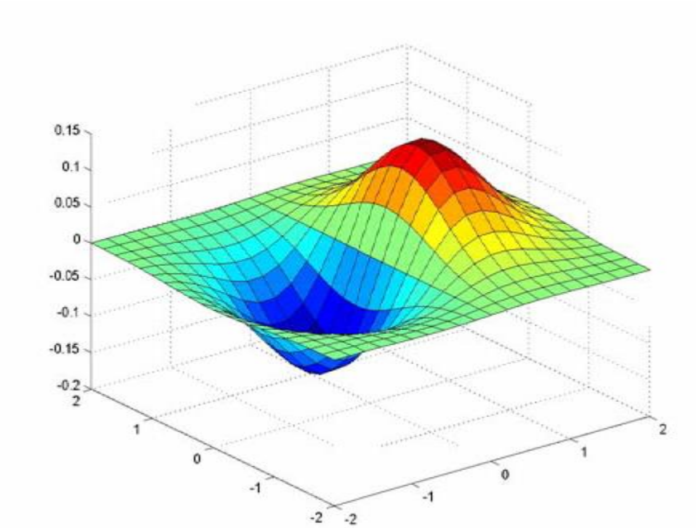
$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

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

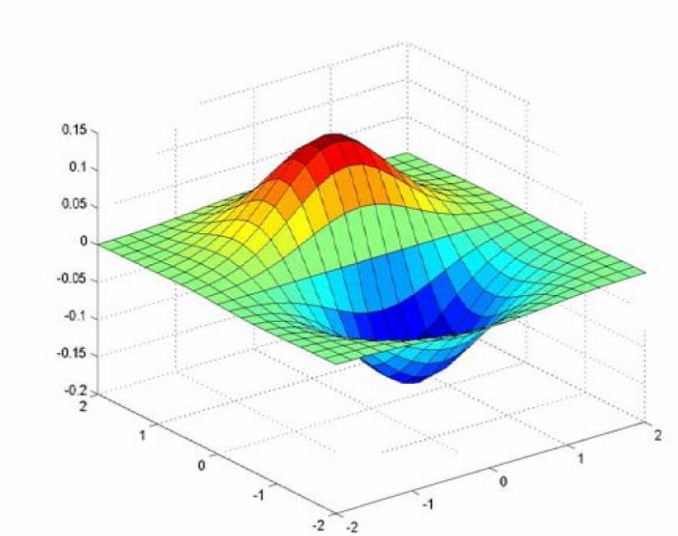


Credit: K. Grauman

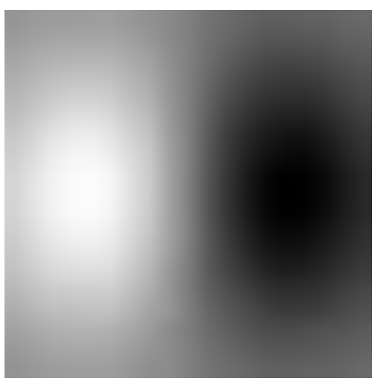
More important filters: Derivative of Gaussian

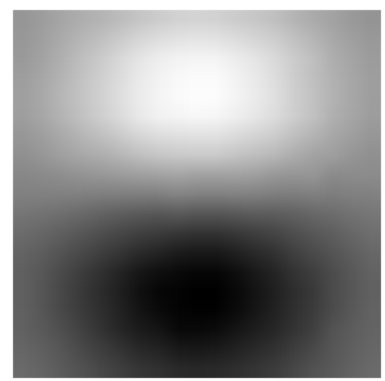


x-direction



y-direction



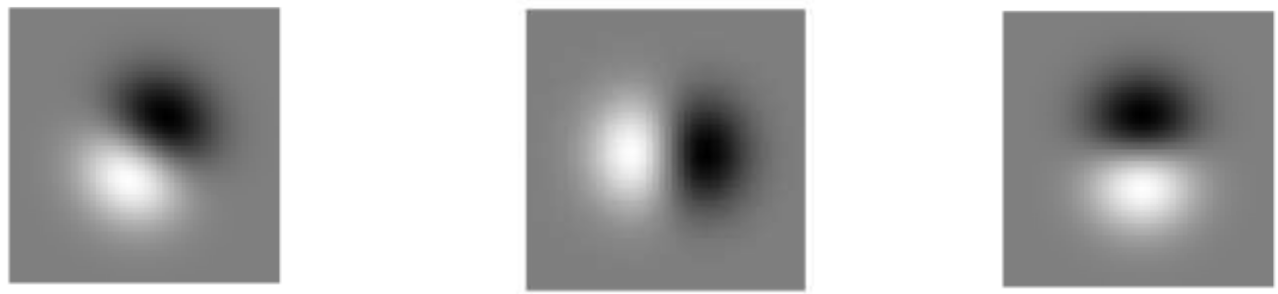


Credit: L. Lazebnik

Gaussian filters: Steerability

Distributive property: $$f * (g + h) = f * g + f * h$$

Steerable filter: $$g_\theta(x, y) = \cos(\theta) g_0(x, y) + \sin(\theta) g_{\pi/2}(x, y)$$

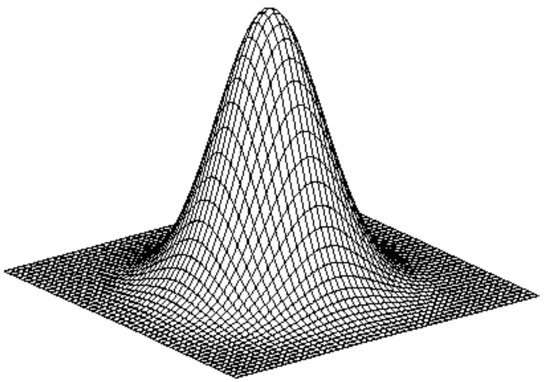


$$I \qquad I * g_\theta = \cos(\theta)(I * g_0) + \sin(\theta)(I * g_{\pi/2})$$

W. Freeman and E. Adelson, 'The design and use of steerable filters', PAMI, 1991

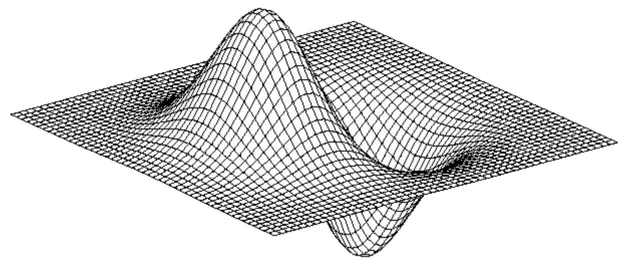
Credit: I. Kokkinos

More important filters: Derivative(s) of Gaussian



Gaussian

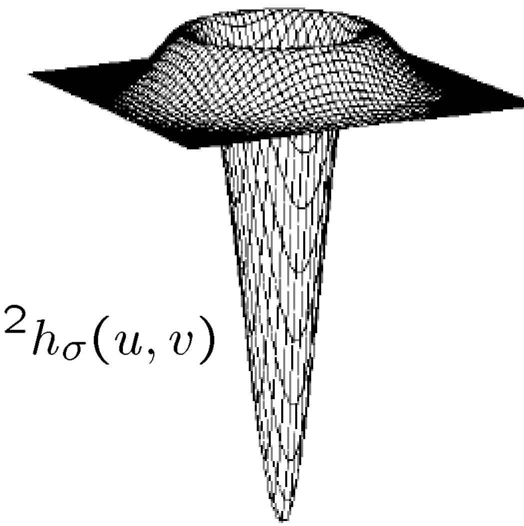
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



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

- ∇^2 is the Laplacian operator:

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

The Fourier transform

Further reading: Szeliski, Richard. Computer Vision: Algorithms and Applications. Springer, 2010, Chapter 3, Section 3.4

Linear image transformations

- In analyzing images, it's often useful to make a change of basis.

transformed image

$$\vec{F} = U\vec{f}$$

Vectorized image

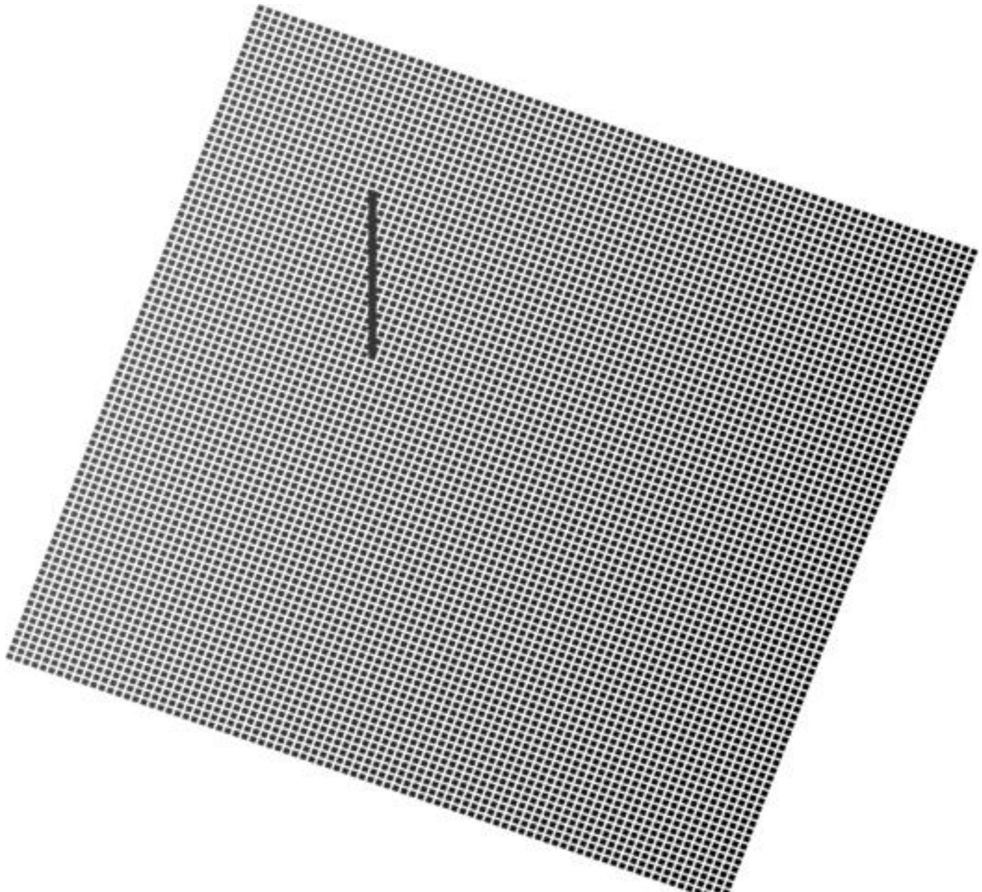
Fourier transform, or
Wavelet transform, or
Steerable pyramid transform

Credit: B. Freeman

Canonical basis for 2D signals

Kronecker delta

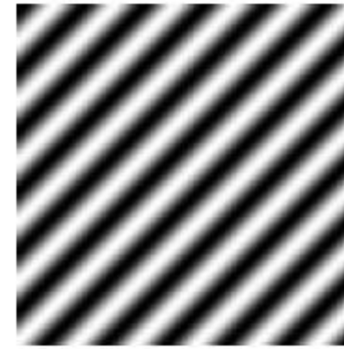
$$d_k[n] = \begin{cases} 1, & n = k \\ 0, & \text{otherwise} \end{cases}$$

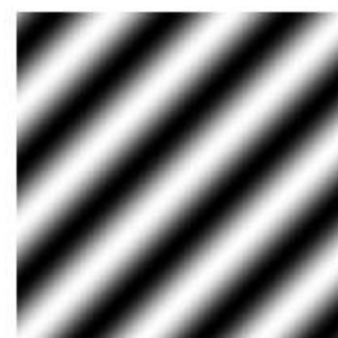


Credit: I. Kokkinos

Fourier Transform = Change of Basis

$$= \quad F(\omega_1) \cdot e^{j\omega_1 x} + F(\omega_2) \cdot e^{j\omega_2 x} + F(\omega_K) \cdot e^{j\omega_K x}$$





Credit: I. Kokkinos

Self-inverting transforms

Same basis functions are used for the inverse transform

$$\vec{f} = U^{-1}\vec{F}$$

$$= U^{+}\vec{F}$$

U transpose and complex conjugate

Credit: B. Freeman

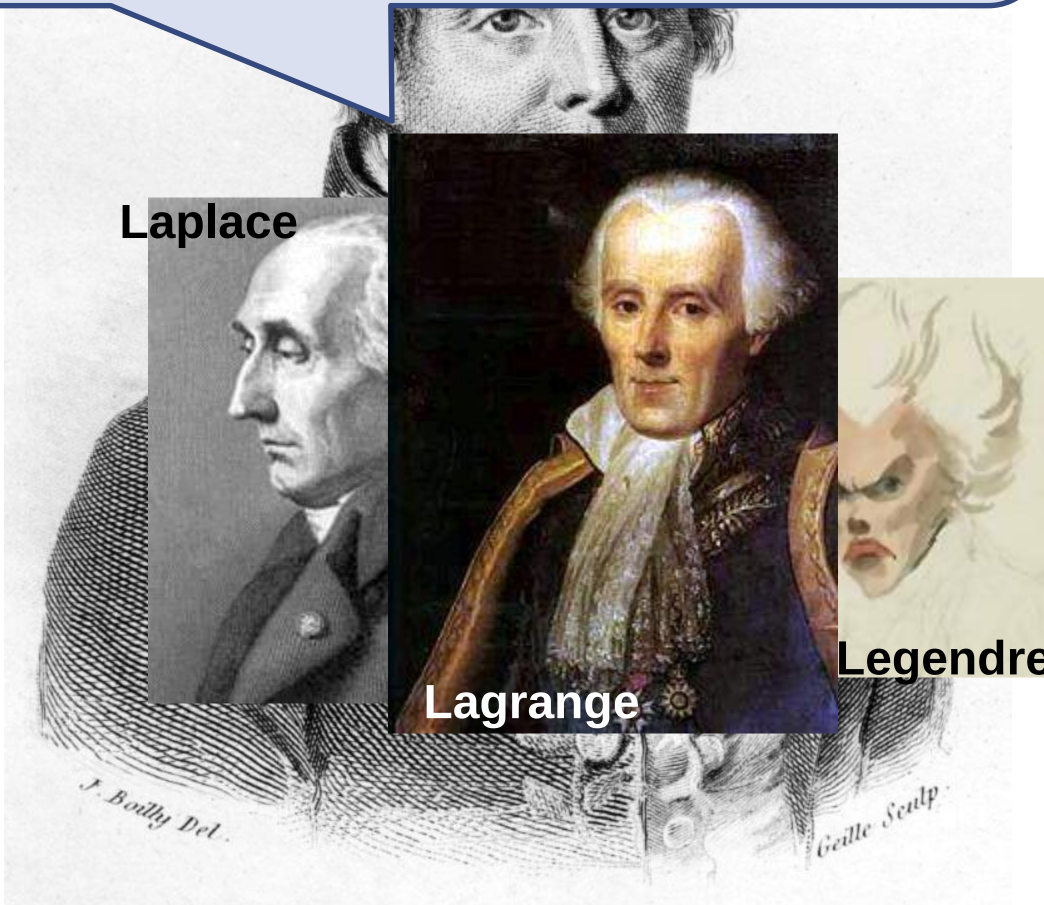
Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

> ***Any** univariate function can rewritten as a weighted sum sines and cosines of different frequencies.*

...the manner in which the author arrives at these equations is not exempt of difficulties and...his analysis to integrate them still leaves something to be desired on the score of generality and even rigour.

- Don't believe it?
  - Neither did Lagrange, Laplace, Poisson and others
  - Not translated into English until 1878!
- But it's true!
  - called Fourier Series



Credit: J. Hays

Fourier Transform

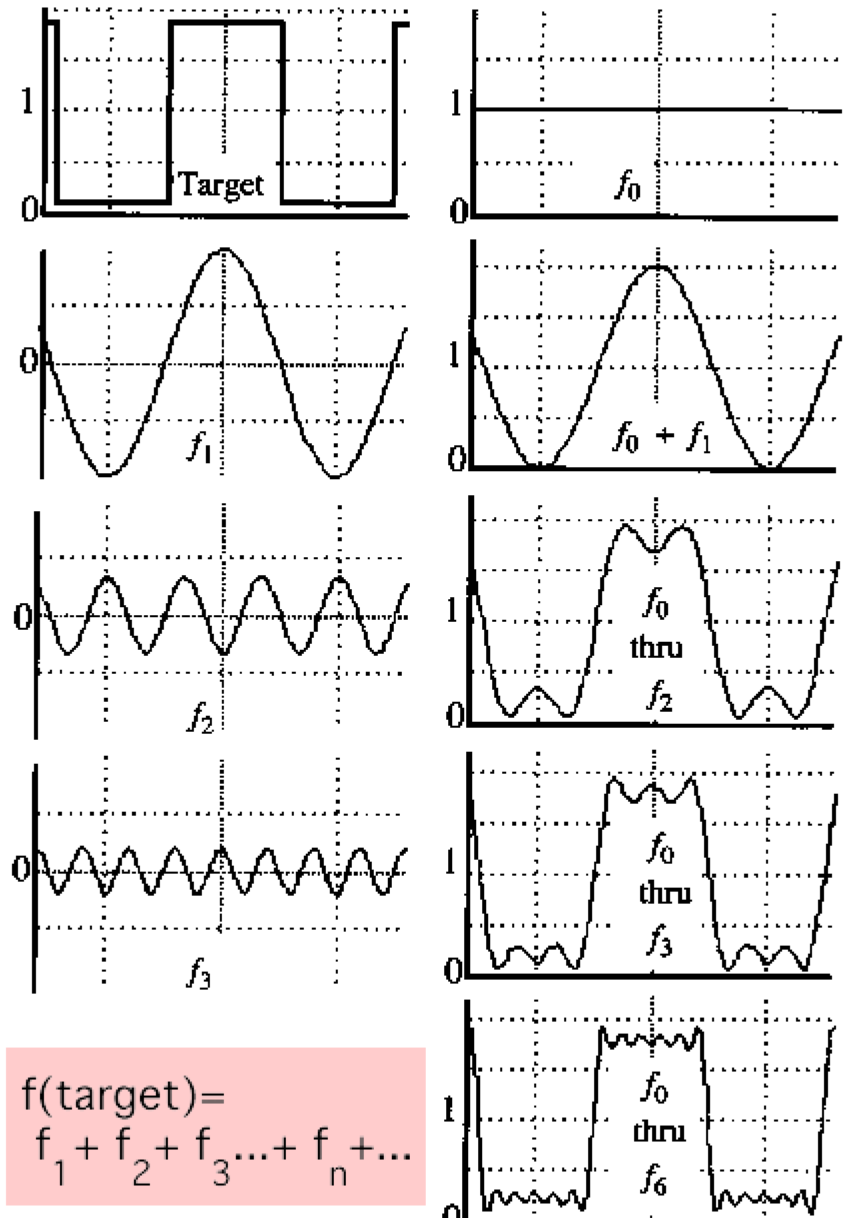
Our building block:

$$A\sin(\omega x + \phi)$$

Add enough of them to get any signal *g(x)* you want!

The Fourier transform ***F(ω)*** stores the magnitude and phase at each frequency

- Magnitude A encodes how much signal there is at a particular frequency ω
- Phase ϕ encodes spatial information (indirectly)



f(target)= $f_1 + f_2 + f_3 ... + f_n + ...$

Credit: J. Hays

Fourier Transform

- We want to understand the frequency ω of our signal. So, let's reparametrize the signal by ω instead of *x*:

f(x) ⟶ Fourier Transform ⟶ ***F(ω)***

For every ω from 0 to inf, ***F(ω)*** holds the amplitude *A* and phase ϕ of the corresponding sine $A\sin(\omega x + \phi)$

- How can *F* hold both? Complex number trick!

$$F(\omega) = R(\omega) + iI(\omega)$$

$$A = \pm\sqrt{R(\omega)^2 + I(\omega)^2} \qquad \phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$$

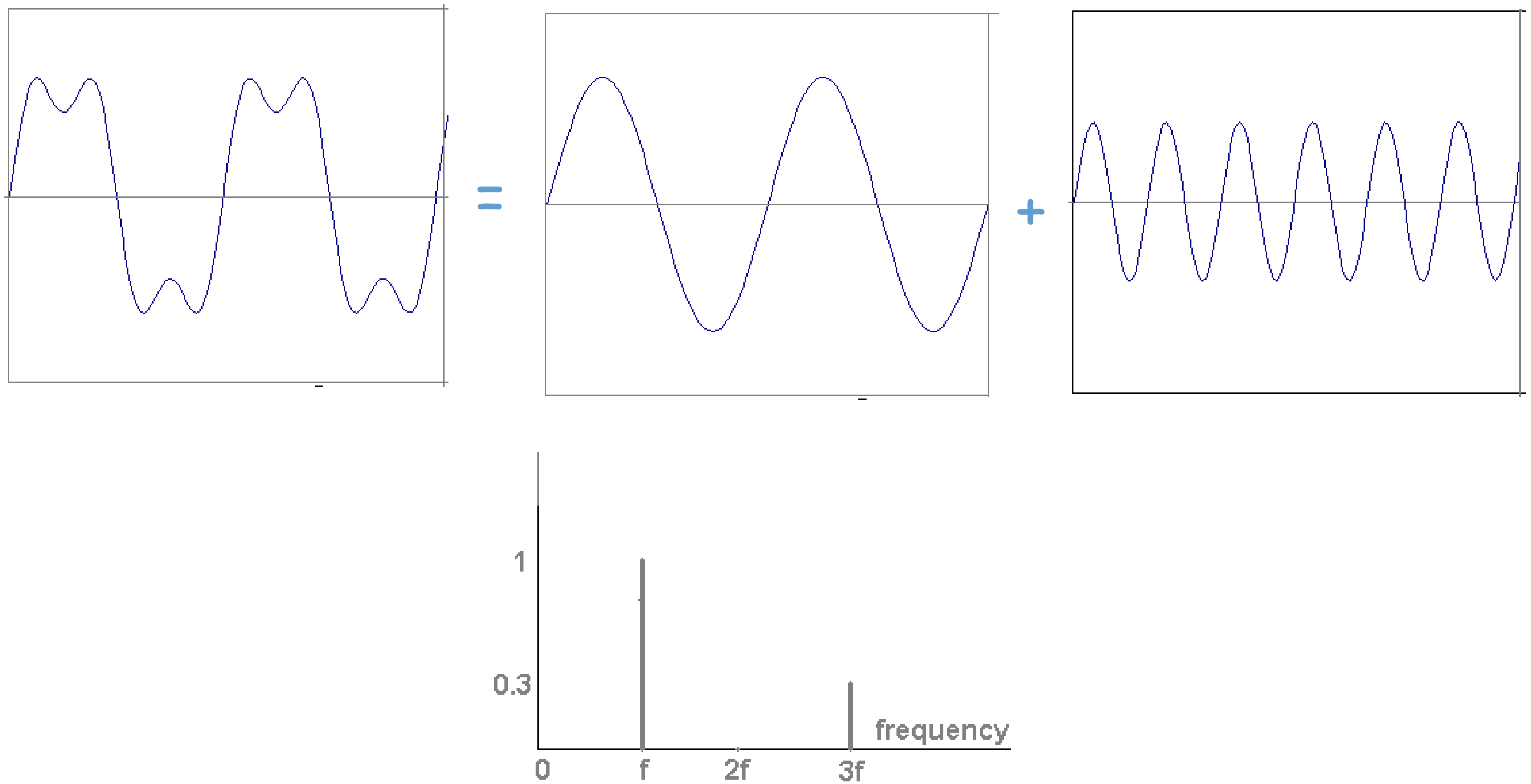
We can always go back:

F(ω) ⟶ Inverse Fourier Transform ⟶ ***f(x)***

Credit: A. Efros

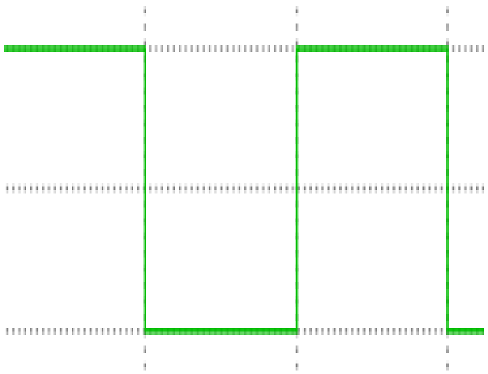
Frequency Spectra

- example : $g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi(3f)\, t)$



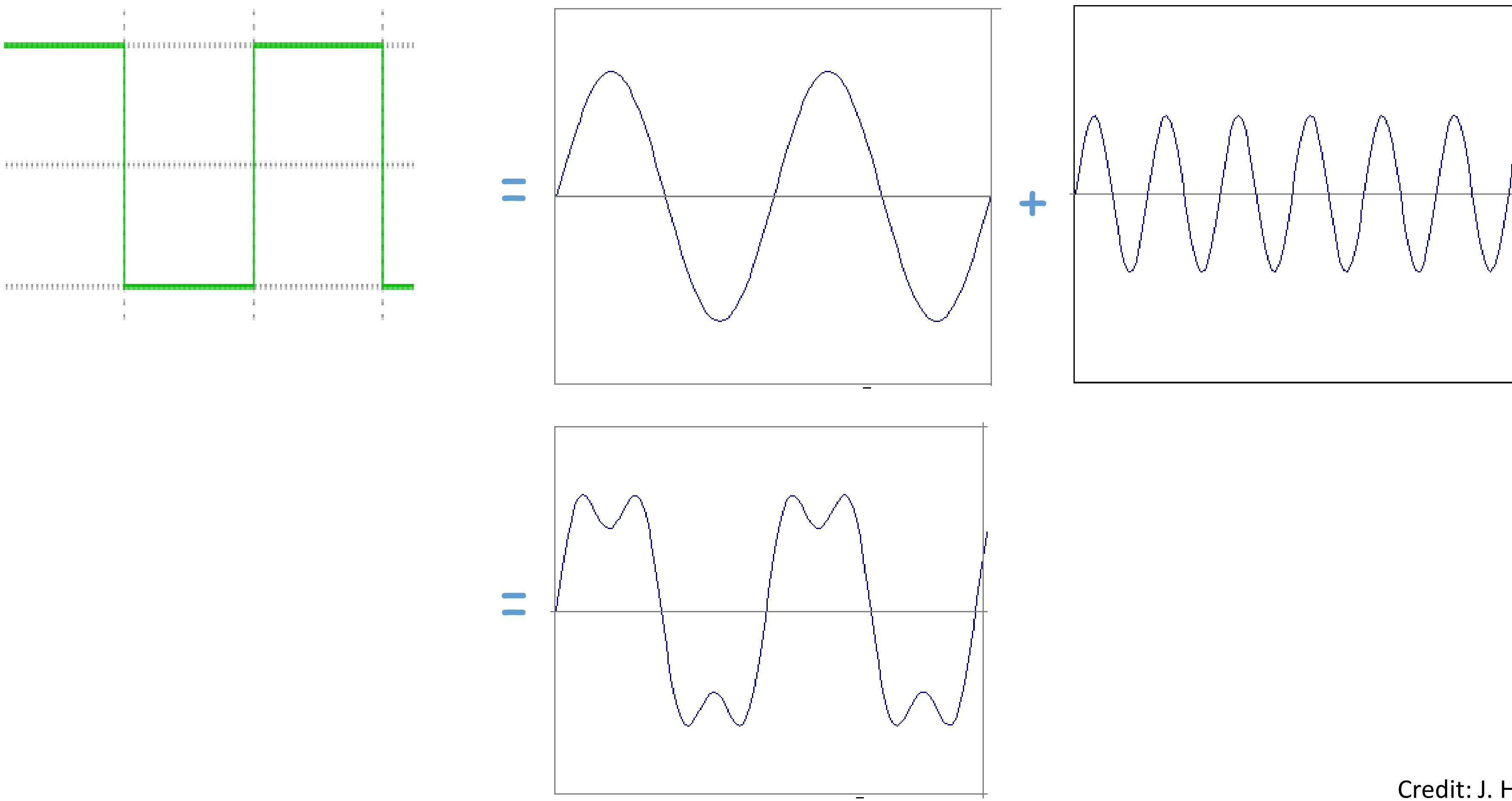
Credit: A. Efros

Frequency Spectra



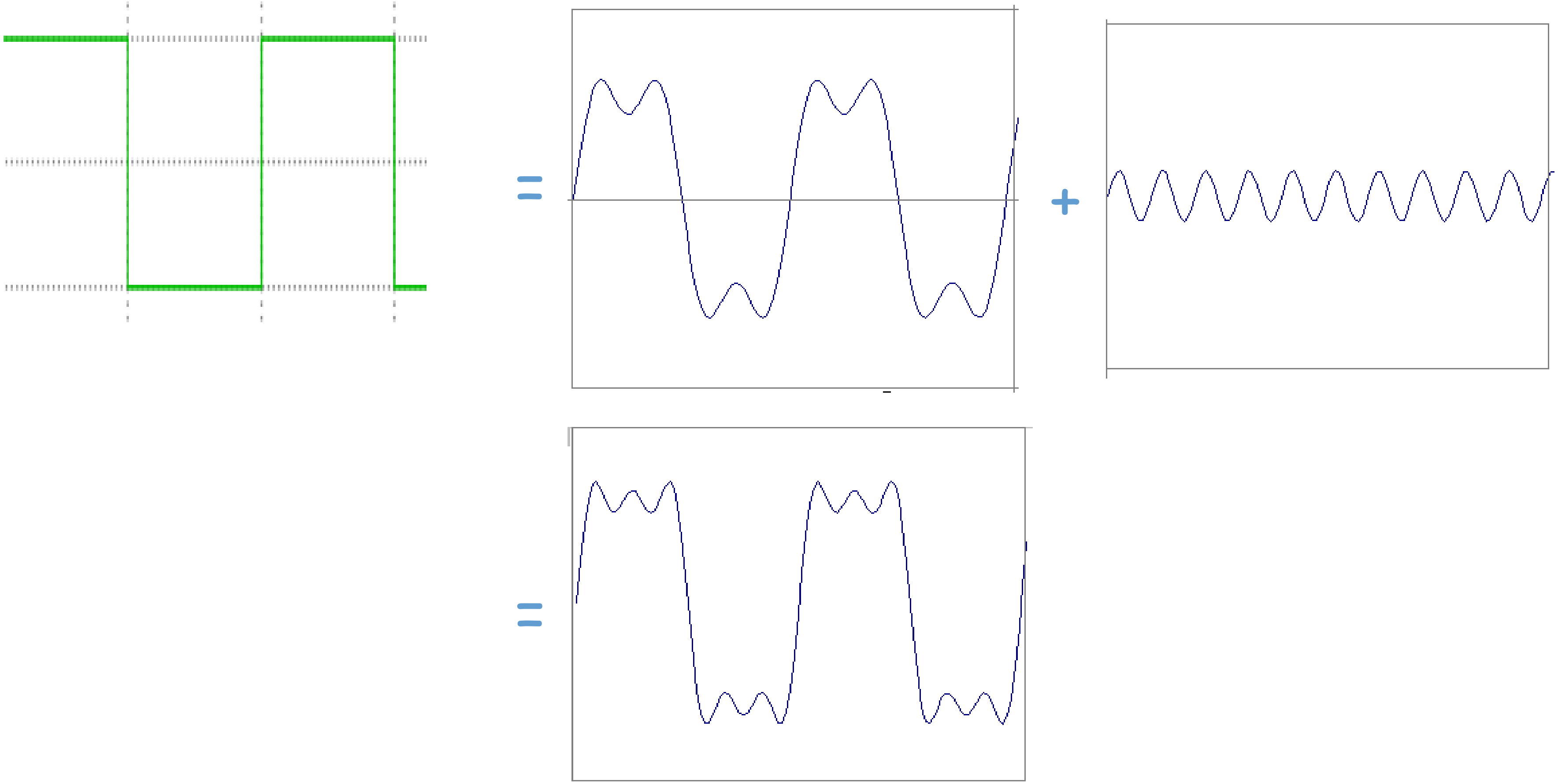
Credit: J. Hays

Frequency Spectra



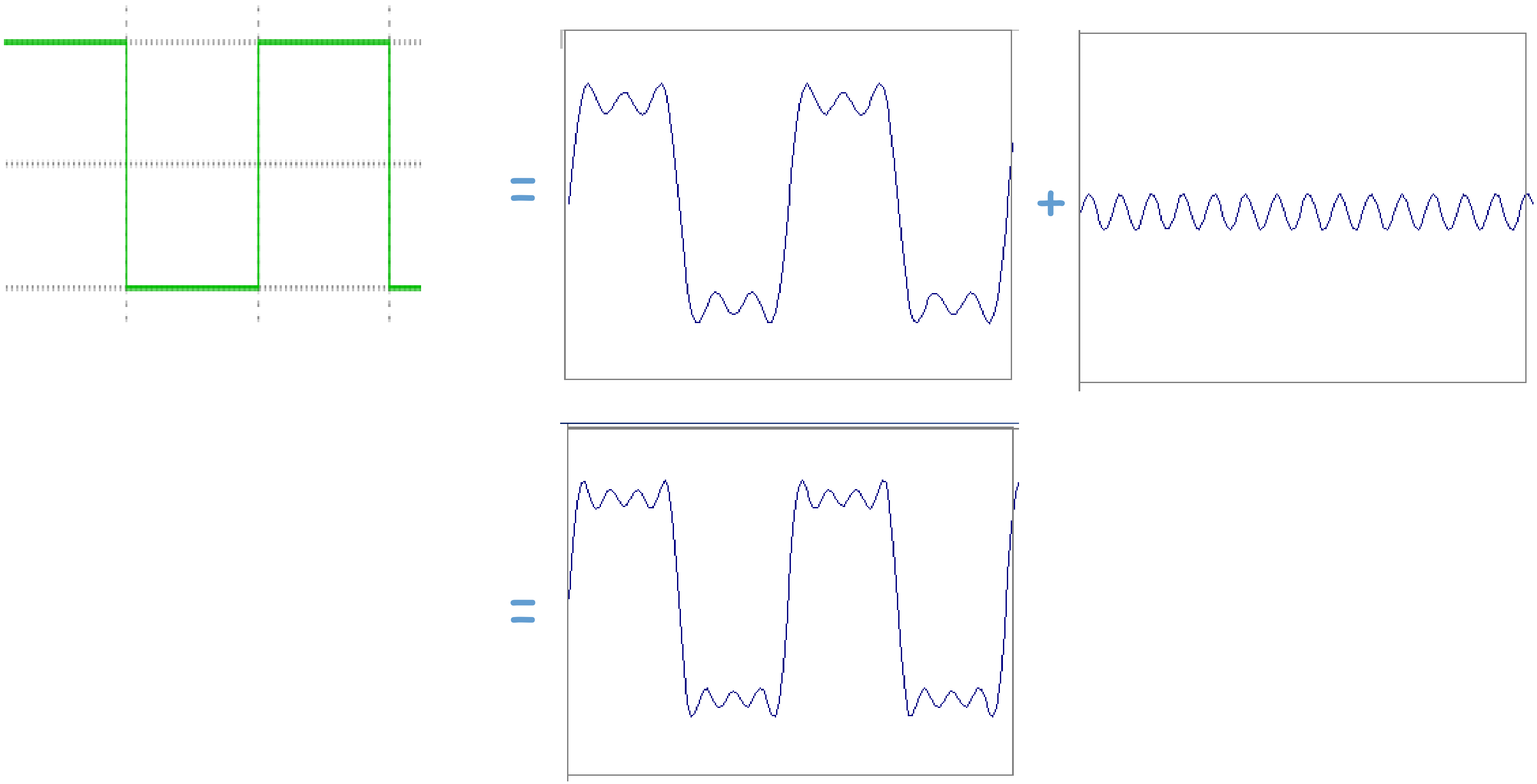
Credit: J. Hays

Frequency Spectra



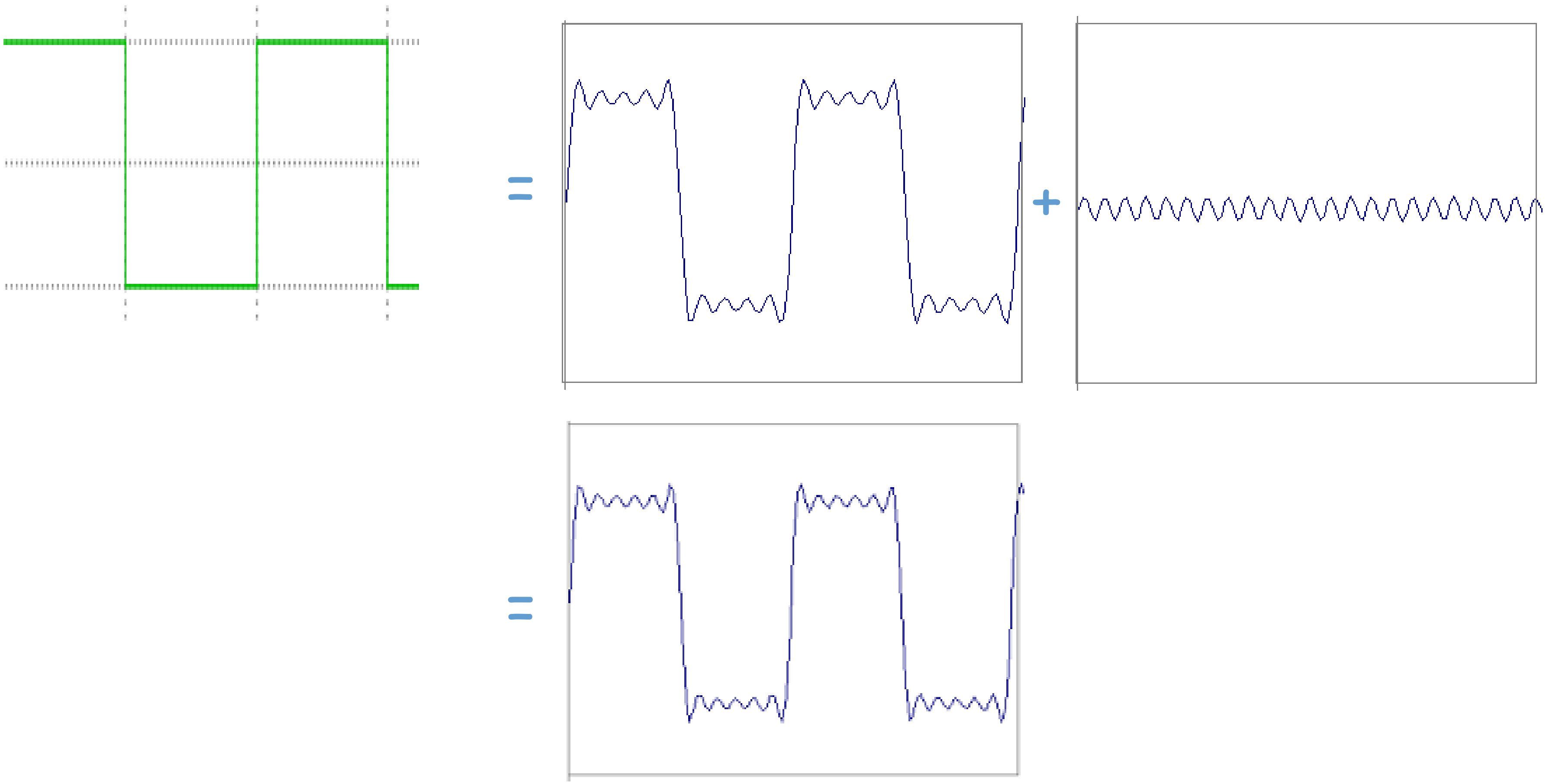
Credit: J. Hays

Frequency Spectra



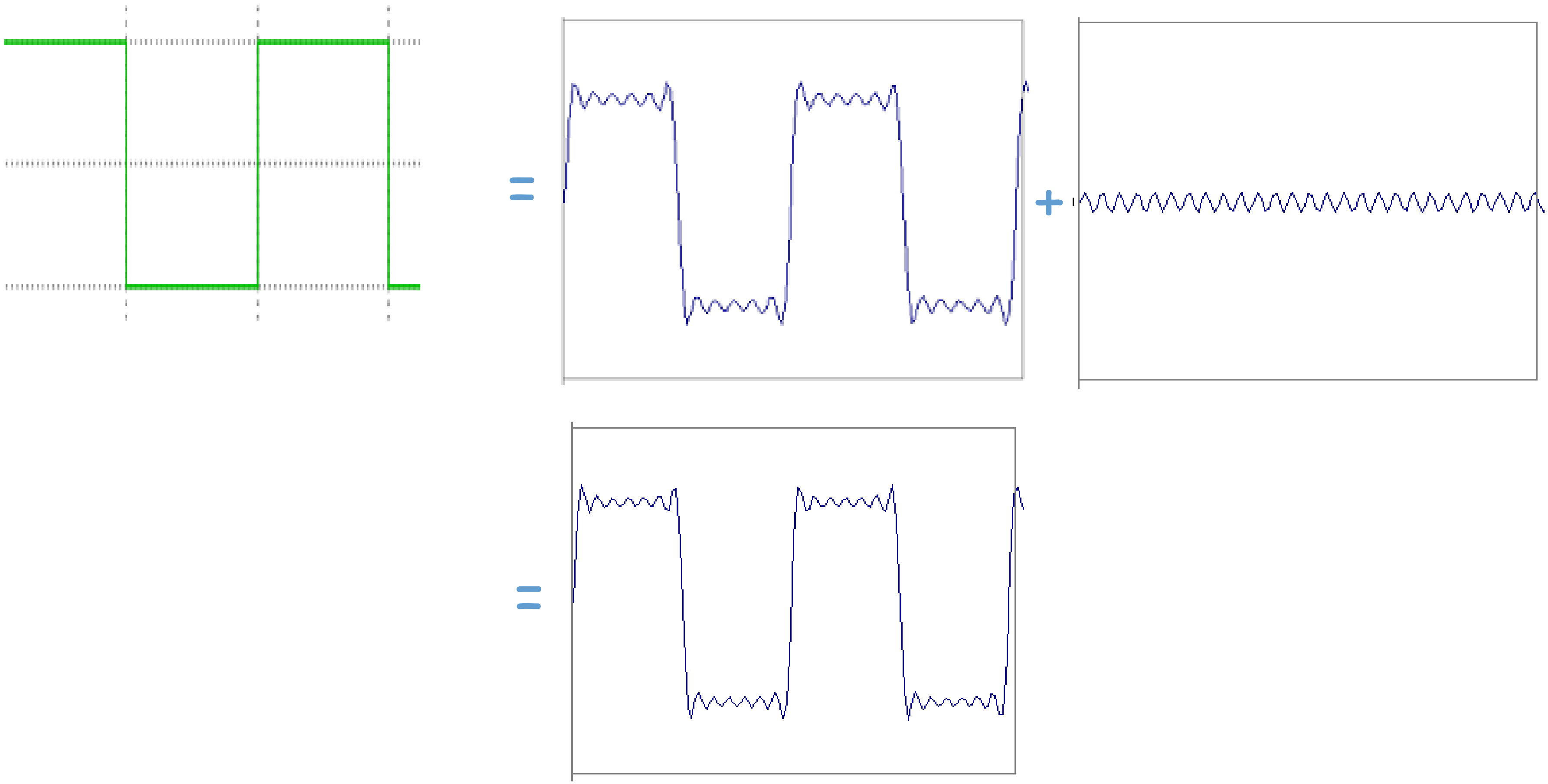
Credit: J. Hays

Frequency Spectra



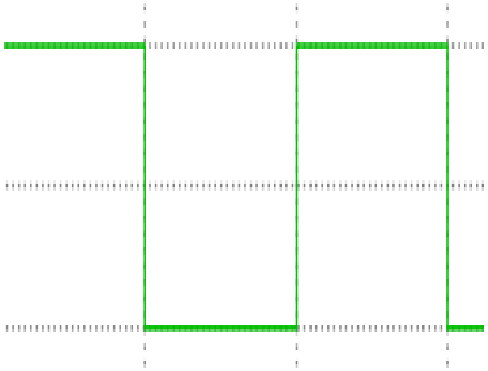
Credit: J. Hays

Frequency Spectra



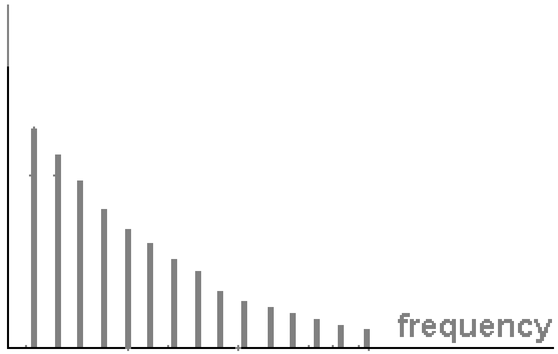
Credit: J. Hays

Frequency Spectra



=

$$\boxed{A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)}$$



Credit: J. Hays

Basics: The Fourier transform

Eigenfunctions

- An eigenfunction of a system is one that is simply multiplied by another factor in the output.

$$f(w) \longrightarrow \boxed{\qquad\qquad} \longrightarrow A(w)\ \mathrm{f}(w)$$

- We think of this as analogous to the case of eigenvectors from linear algebra.

Credit: R. Wildes

Basics: The Fourier transform

Eigenfunctions

- An eigenfunction of a system is one that is simply multiplied by another factor in the output.

$$f(w) \longrightarrow \boxed{\quad\quad} \longrightarrow A(w)\ \mathrm{f}(w)$$

- We think of this as analogous to the case of eigenvectors from linear algebra.

Remark

- Notation

$$e^{iwt} = \exp(iwt)$$

with the imaginary number

$$i = \sqrt{-1}$$

Credit: R. Wildes

Basics: The Fourier transform

Eigenfunctions

- An eigenfunction of a system is one that is simply multiplied by another factor in the output.

$$f(w) \longrightarrow \boxed{\quad\quad} \longrightarrow A(w)\, \mathrm{f}(w)$$

- We think of this as analogous to the case of eigenvectors from linear algebra.
- For the case of 1D Linear Shift Invariant (LSI) systems we find that $\exp(iwt)$ is an eigenfunction of convolution.

$$\exp(iwt) \longrightarrow \boxed{\quad\quad} \longrightarrow A(w) \exp(iwt)$$

- Here $A(w)$ is the (possibly complex) factor by which the input signal is multiplied.
- So, from the input exponential we obtain another exponential; but, scaled and shifted in phase.

Credit: R. Wildes

Basics: The Fourier transform

Eigenfunctions

- An eigenfunction of a system is one that is simply multiplied by another factor in the output.

$$f(w) \longrightarrow \boxed{} \longrightarrow A(w)\,\mathrm{f}(w)$$

- We think of this as analogous to the case of eigenvectors from linear algebra.
- For the case of 1D LSI systems we find that $\exp(iwt)$ is an eigenfunction of convolution.

$$\exp(iwt) \longrightarrow \boxed{} \longrightarrow A(w)\exp(iwt)$$

- Here $A(w)$ is the (possibly complex) factor by which the input signal is multiplied.
- So, from the input exponential we obtain another exponential; but, scaled and shifted in phase.

Frequency

- We call w the frequency (or wave number) of the eigenfunction.
- In practice, we use real waveforms, like $\cos(wt)$ and $\sin(wt)$, with the relationship

$$\exp(iwt) = \cos(wt) + i\sin(wt)$$

  which is known as Euler's relation.
- The complex exponential is used in derivations simply because it provides a compact notation.

Credit: R. Wildes

Basics: The Fourier transform

1D frequency

- We consider functions of the form

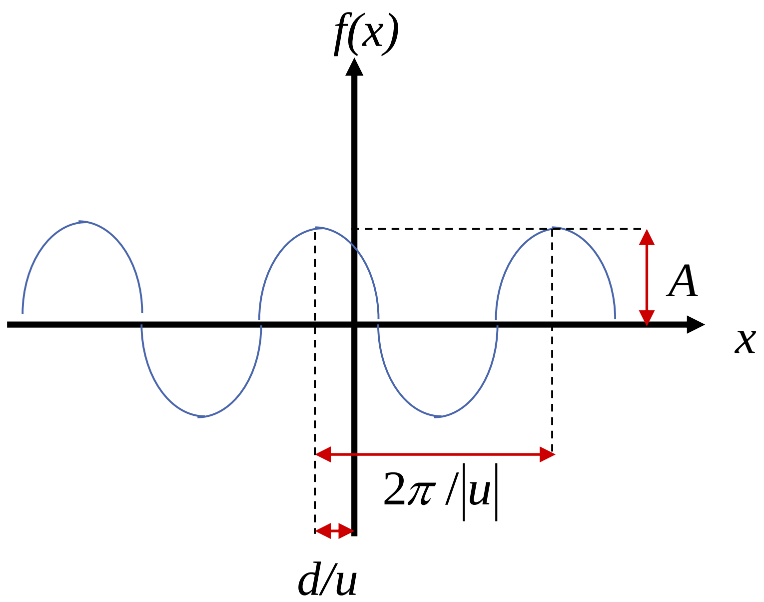
  $$f(x) = A\cos(ux + d)$$

  where

  A is the amplitude

  u is the (angular) frequency

  d is the phase constant.
- Notice that the function repeats its value when $ux + d$ increases by 2π
- For example, when $d = 0$, the maxima and minima occur when $ux = k\pi$, for k an integer.



Credit: R. Wildes

Fourier transform: 1D case

Consider

- A graphical interpretation

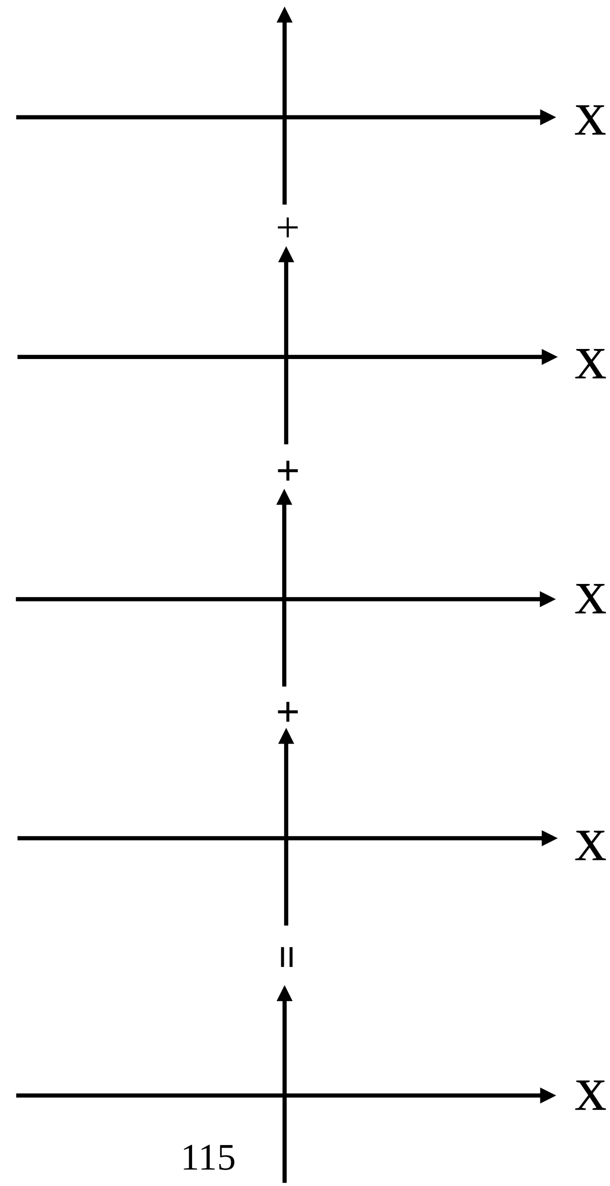
$$1$$

$$+\frac{1}{2}\cos 2\pi x$$

$$+\cos 4\pi x$$

$$+\frac{2}{3}\cos 6\pi x$$

$$= f(x)$$



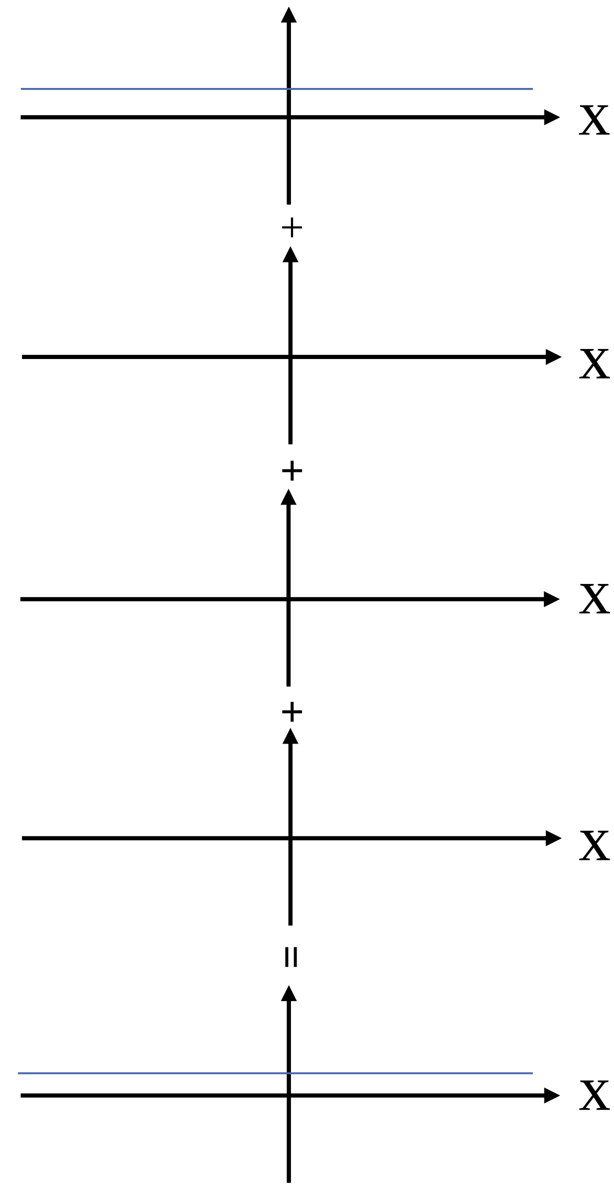
Credit: R. Wildes

Fourier transform: 1D case

Consider

- A graphical interpretation

$$
\begin{aligned}
&1 \\
&+\frac{1}{2}\cos 2\pi x \\
&+\cos 4\pi x \\
&+\frac{2}{3}\cos 6\pi x \\
&= f(x)
\end{aligned}
$$



Credit: R. Wildes

Fourier transform: 1D case

Consider

- A graphical interpretation

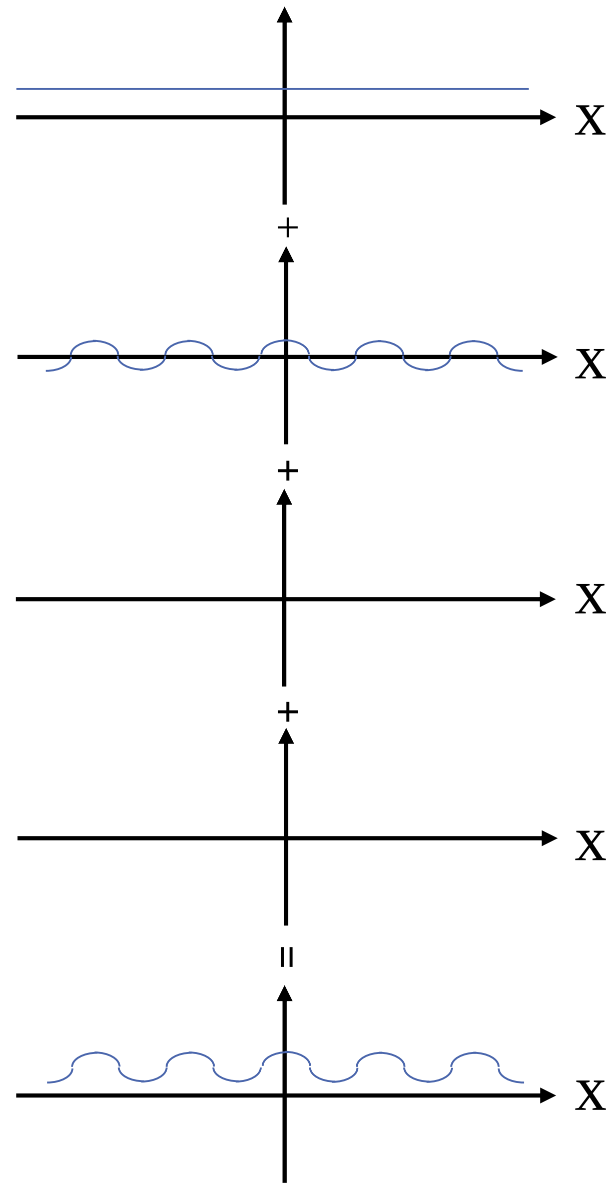
$$1$$

$$+\frac{1}{2}\cos 2\pi x$$

$$+\cos 4\pi x$$

$$+\frac{2}{3}\cos 6\pi x$$

$$= f(x)$$



Credit: R. Wildes

Fourier transform: 1D case

Consider

- A graphical interpretation

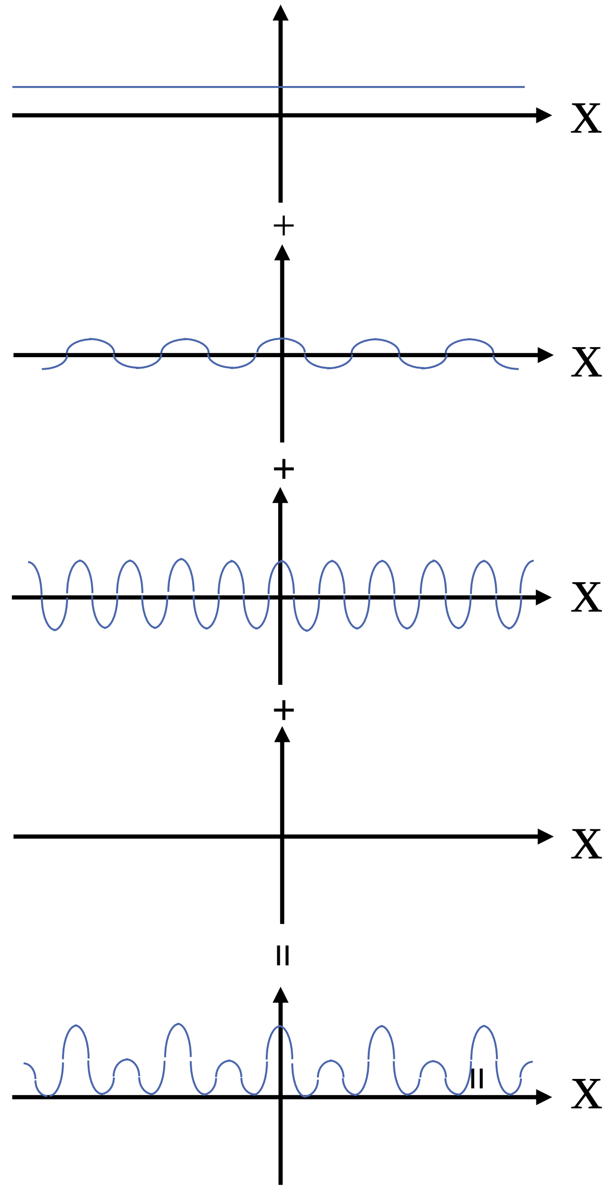
$$1$$

$$+\frac{1}{2}\cos 2\pi x$$

$$+\cos 4\pi x$$

$$+\frac{2}{3}\cos 6\pi x$$

$$= f(x)$$



Credit: R. Wildes

Fourier transform: 1D case

Consider

- A graphical interpretation

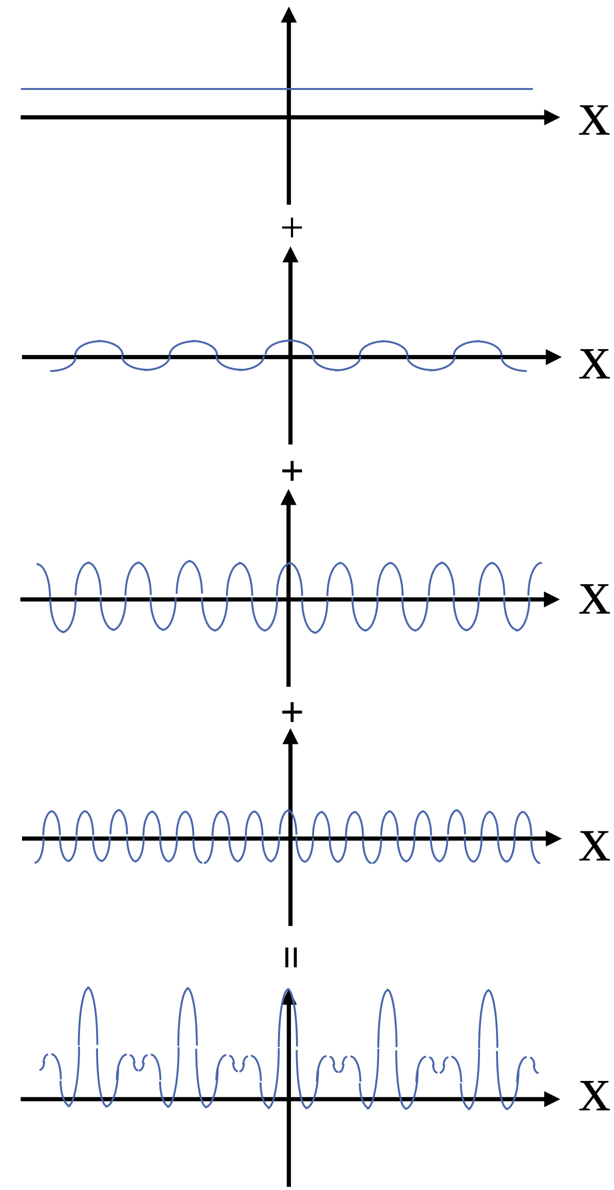
$$1$$

$$+\frac{1}{2}\cos 2\pi x$$

$$+\cos 4\pi x$$

$$+\frac{2}{3}\cos 6\pi x$$

$$= f(x)$$

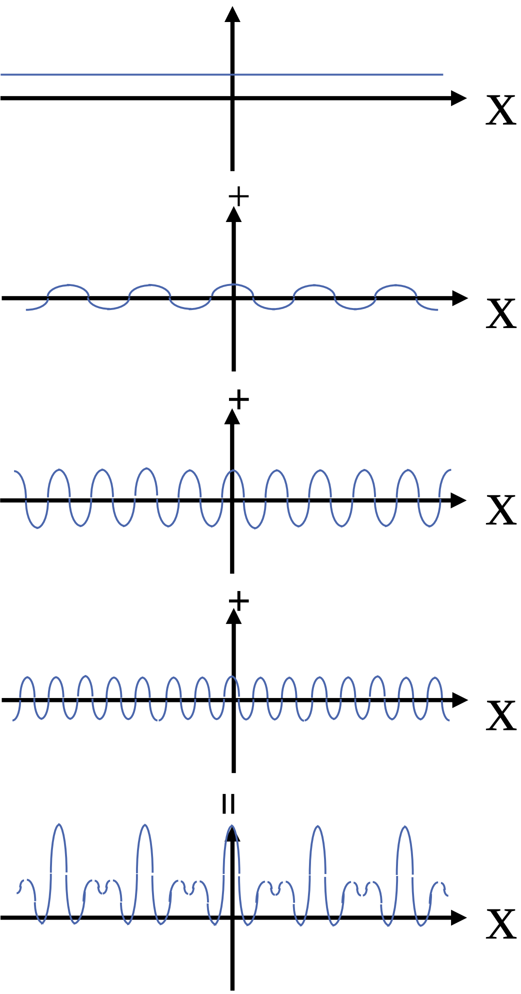


Observation

- Complicated signals can be represented as the sum of simple components.

Credit: R. Wildes

Fourier transform: 1D case



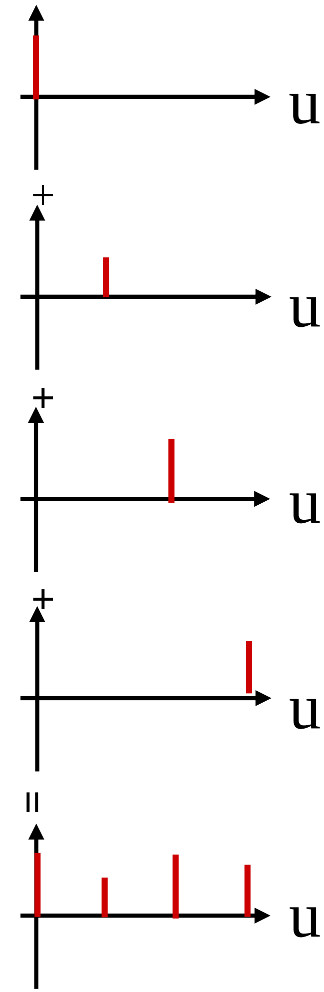
f(x)

$$1\cos 0\pi x \rightarrow 1\cos ux; u = 0$$

$$+\frac{1}{2}\cos 2\pi x \rightarrow +\frac{1}{2}\cos ux; u = 2\pi$$

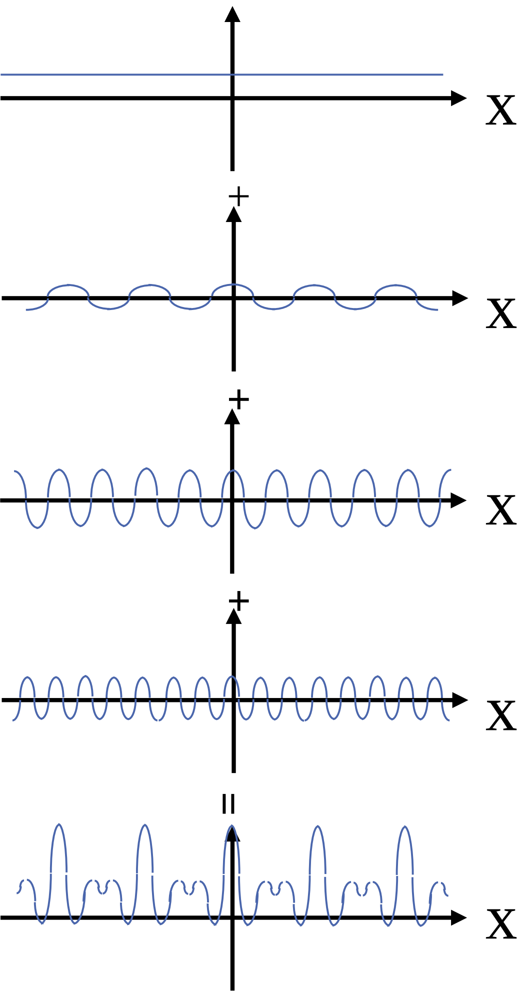
$$+\cos 4\pi x \rightarrow +1\cos ux; u = 4\pi$$

$$+\frac{2}{3}\cos 6\pi x \rightarrow +\frac{2}{3}\cos ux; u = 6\pi$$



Credit: R. Wildes

Fourier transform: 1D case



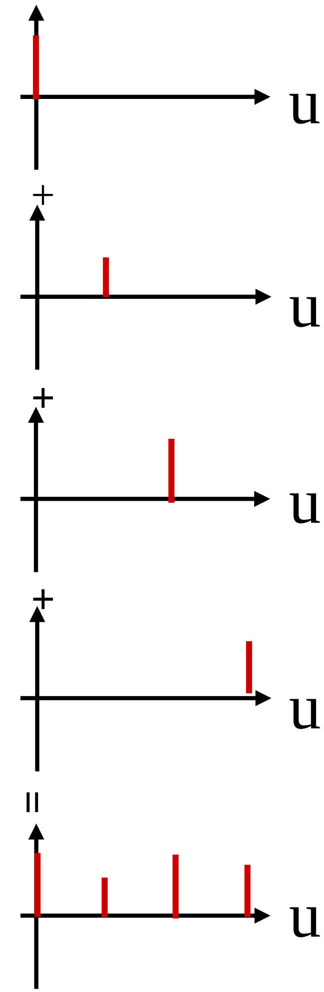
$f(x)$

$$1\cos 0\pi x \rightarrow 1\cos ux; u = 0$$

$$+\frac{1}{2}\cos 2\pi x \rightarrow +\frac{1}{2}\cos ux; u = 2\pi$$

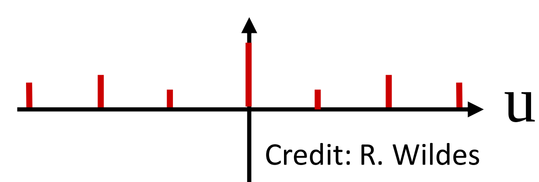
$$+\cos 4\pi x \rightarrow +1\cos ux; u = 4\pi$$

$$+\frac{2}{3}\cos 6\pi x \rightarrow +\frac{2}{3}\cos ux; u = 6\pi$$



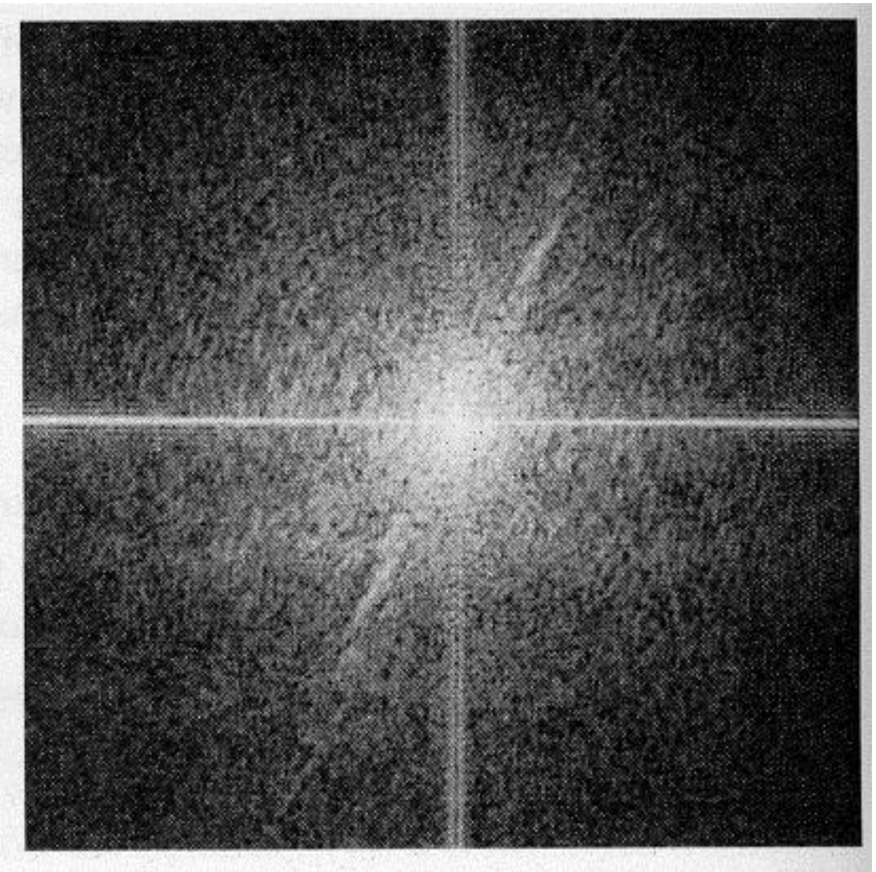
Note

- By symmetry, we may choose to represent this as



Credit: R. Wildes

The Fourier transform of an image



Source image (J. Fourier)

Fourier power spectrum

Credit: R. Wildes

Basics: The 2D Fourier transform

2D Eigenfunctions

- For the case of 1D LSI systems we found that $\exp(iwt)$ is an eigenfunction of convolution.

$$\exp(iwt) \longrightarrow \boxed{\qquad\qquad} \longrightarrow A(w)\exp(iwt)$$

- One can show that that $\exp[i(ux+vy)]$ is an eigenfunction in 2D.

$$\exp[i(ux+vy)] \longrightarrow \boxed{\qquad\qquad} \longrightarrow A(u,v)\exp[i(ux+vy)]$$

- The 2D Fourier transform $F(u,v)$ of $f(x,y)$ is given by

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\exp[-i(ux+vy)]dxdy$$

Credit: R. Wildes

The 2D discrete Fourier transform

$$F[u,v] = \left|F[u,v]\right| e^{i\phi[u,v]}$$

Discrete domain, Image of height M and width N

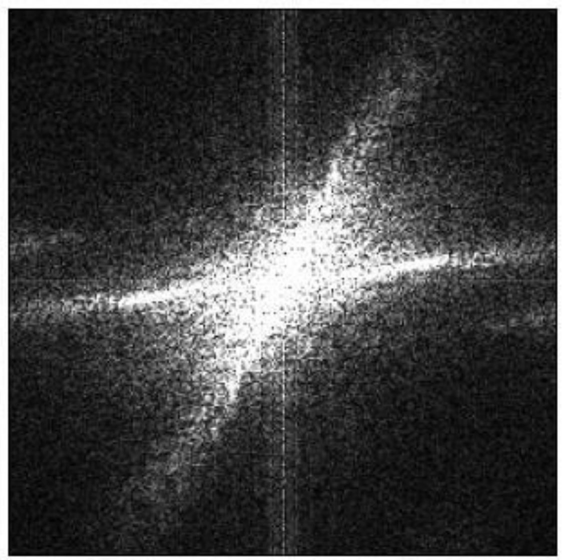
Forward transform

$$F[u,v] = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f[x,y] e^{-\pi i\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

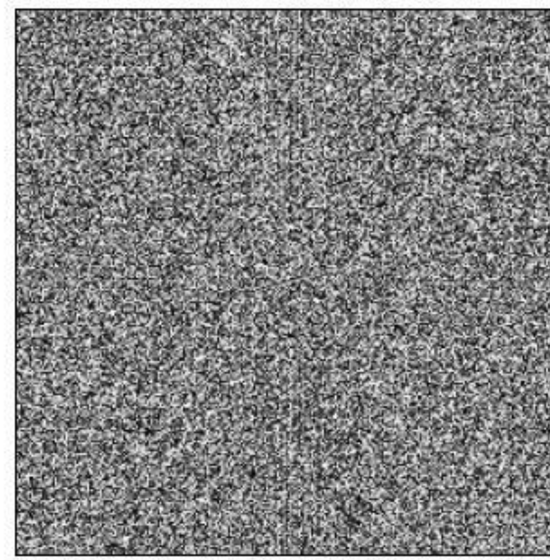
Inverse transform

$$f[x,y] = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F[u,v] e^{+\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)}$$

Source image $f[x,y]$



Fourier spectrum $\left|F[u,v]\right|$

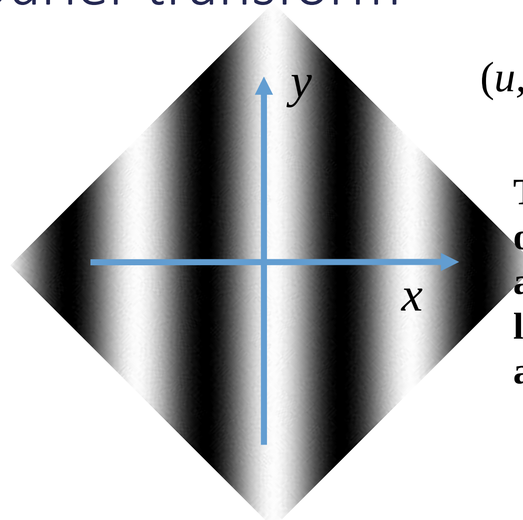


Fourier phase $\phi[u,v]$

Basics: The Fourier transform

2D frequency

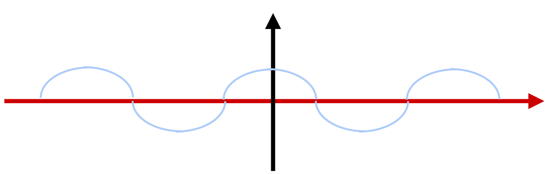
- For two spatial dimensions, we see that there are two corresponding frequency components, u and v.
- We refer to the uv-plane as the frequency domain.
- We refer to the xy-plane as the spatial domain.
- The real waveforms $cos(ux+vy)$ and $sin(ux+vy)$ correspond to waves in 2D.



$f[x, y]$

$(u,v)=(a,0)$

The maxima and minima of the cosinusoids lie along parallel equidistant lines $ux+vy=k\pi$ for k an integer.



Cross sections orthogonal to the ridges show a sinusoidal profile

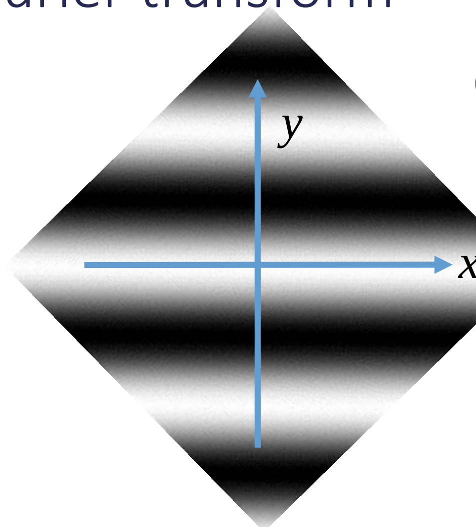
Credit: R. Wildes

Basics: The Fourier transform

2D frequency

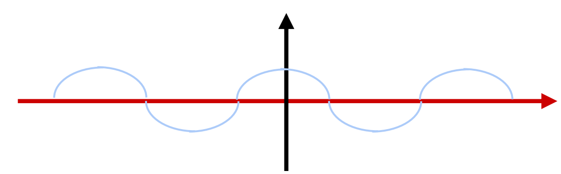
- For two spatial dimensions, we see that there are two corresponding frequency components, u and v.
- We refer to the uv-plane as the frequency domain.
- We refer to the xy-plane as the spatial domain.
- The real waveforms $cos(ux+vy)$ and $sin(ux+vy)$ correspond to waves in 2D.

$(u,v)=(0,a)$



$f[x,y]$

The maxima and minima of the cosinusoids lie along parallel equidistant lines $ux+vy=k\pi$ **for** k **an integer.**



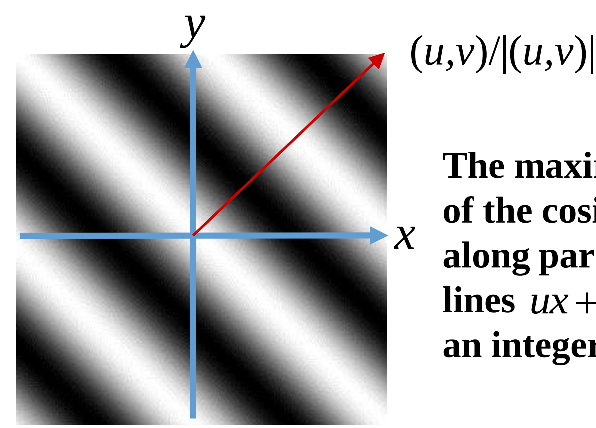
Cross sections orthogonal to the ridges show a sinusoidal profile

Credit: R. Wildes

Basics: The Fourier transform

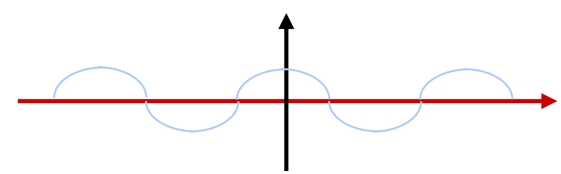
2D frequency

- For two spatial dimensions, we see that there are two corresponding frequency components, u and v.
- We refer to the uv-plane as the frequency domain.
- We refer to the xy-plane as the spatial domain.
- The real waveforms $\cos(ux+vy)$ and $\sin(ux+vy)$ correspond to waves in 2D.



$f[x, y]$

The maxima and minima of the cosinusoids lie along parallel equidistant lines $ux + vy = k\pi$ for k an integer.



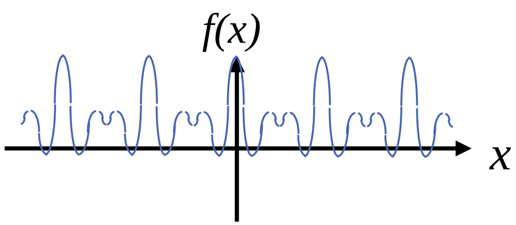
Cross sections orthogonal to the ridges show a sinusoidal profile

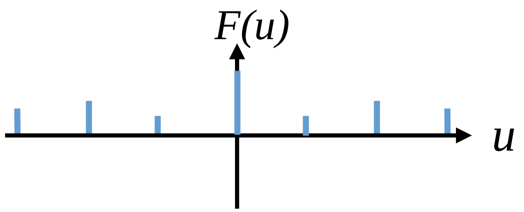
Credit: R. Wildes

The Fourier transform: 2D case

A 2D example

- Recall the 1D example





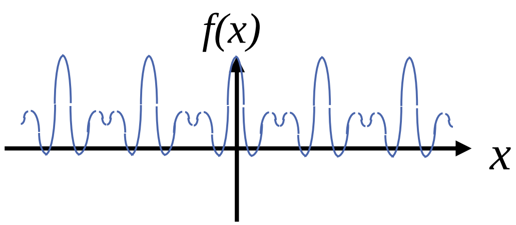
for the cosine component

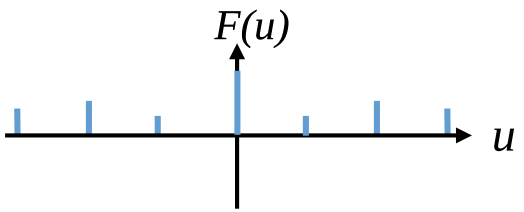
Credit: R. Wildes

The Fourier transform: 2D case

A 2D example

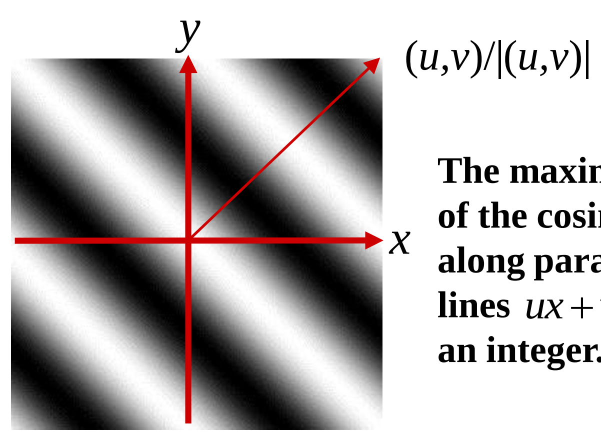
- Recall the 1D example





for the cosine component

- And the interpretation of 2D spatial frequency



$f[x, y]$

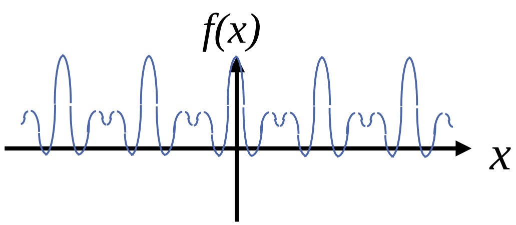
The maxima and minima of the cosinusoids lie along parallel equidistant lines $ux + vy = k\pi$ **for** $\boldsymbol{k}$ **an integer.**

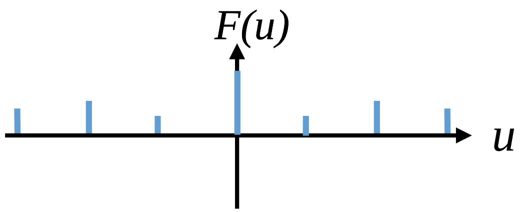
Credit: R. Wildes

The Fourier transform: 2D case

A 2D example

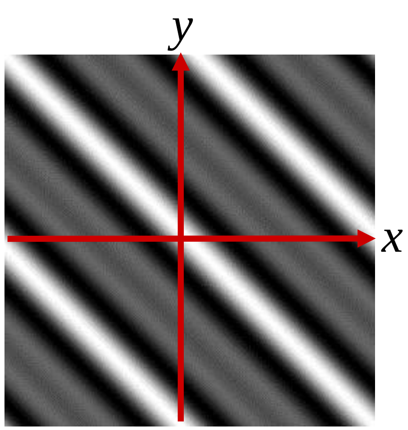
- Recall the 1D example





for the cosine component

- Then a 2D analogue could be



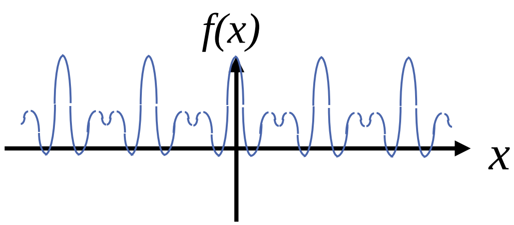
$f[x, y]$

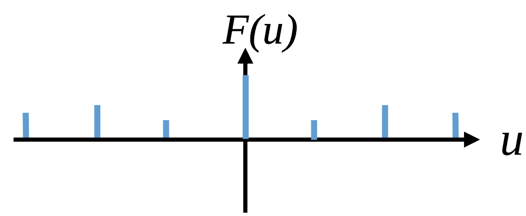
Credit: R. Wildes

The Fourier transform: 2D case

A 2D example

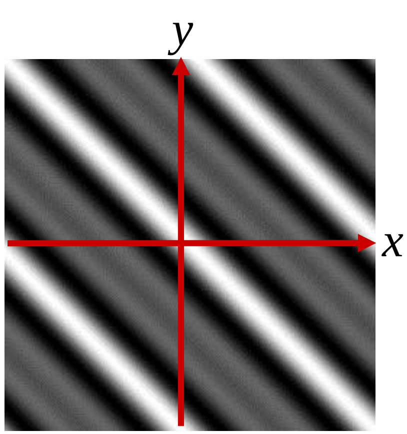
- Recall the 1D example



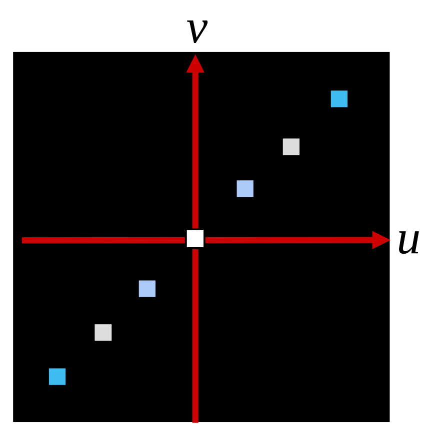


for the cosine component

- Then a 2D analogue could be



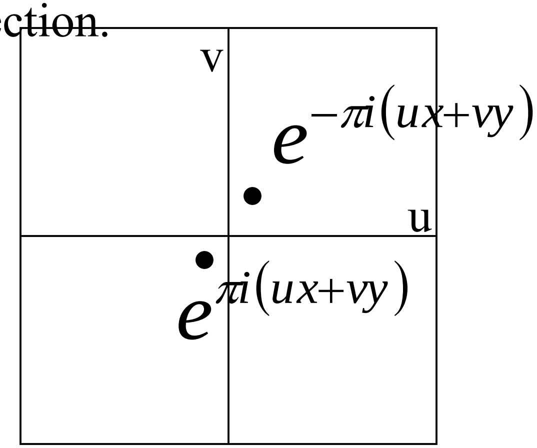
$f[x,y]$



$|F[u,v]|$

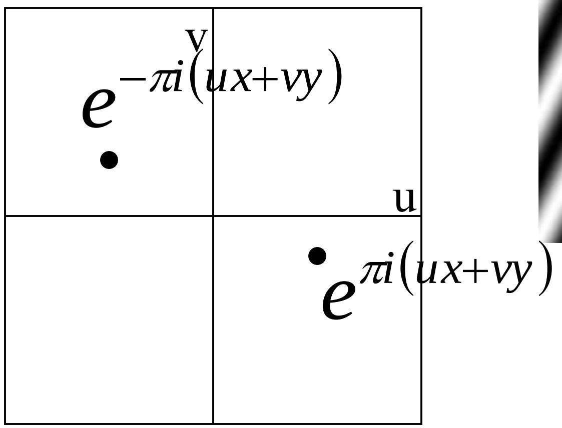
Credit: R. Wildes

To get some sense of what basis elements look like, we plot a basis element --- or rather, its real part --- as a function of x,y for some fixed u, v. We get a function that is constant when (ux+vy) is constant. The magnitude of the vector (u, v) gives a frequency, and its direction gives an orientation. The function is a sinusoid with this frequency along the direction, and constant perpendicular to the direction.



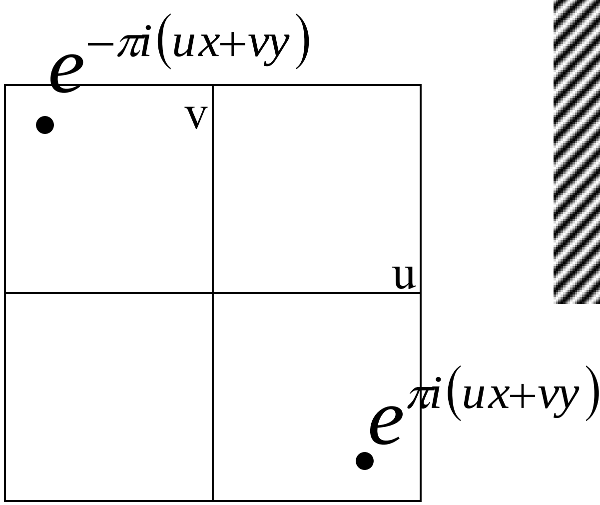
Credit: B. Freeman

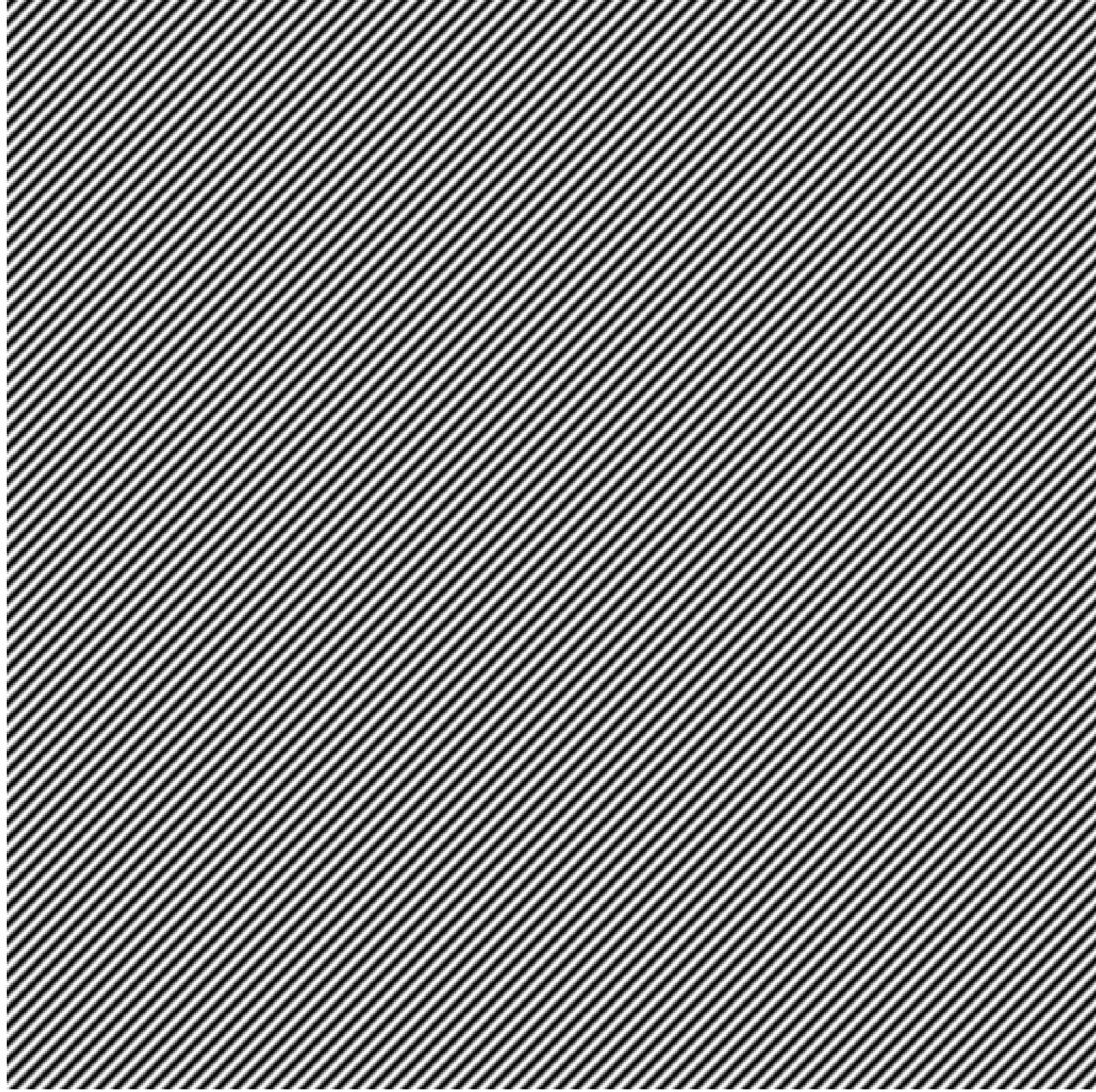
Here u and v are larger than in the previous slide.



Credit: B. Freeman

And larger still...



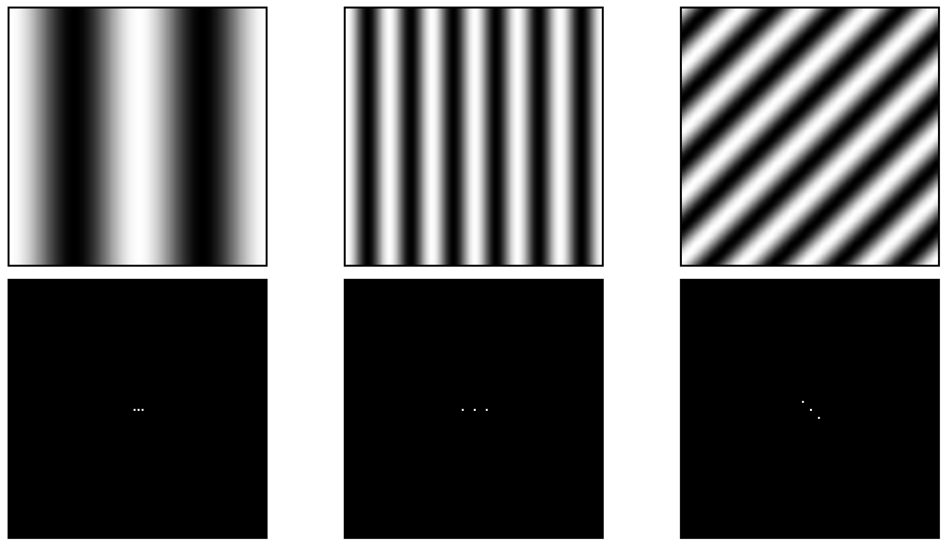


Credit: B. Freeman

The Fourier transform: Filtering

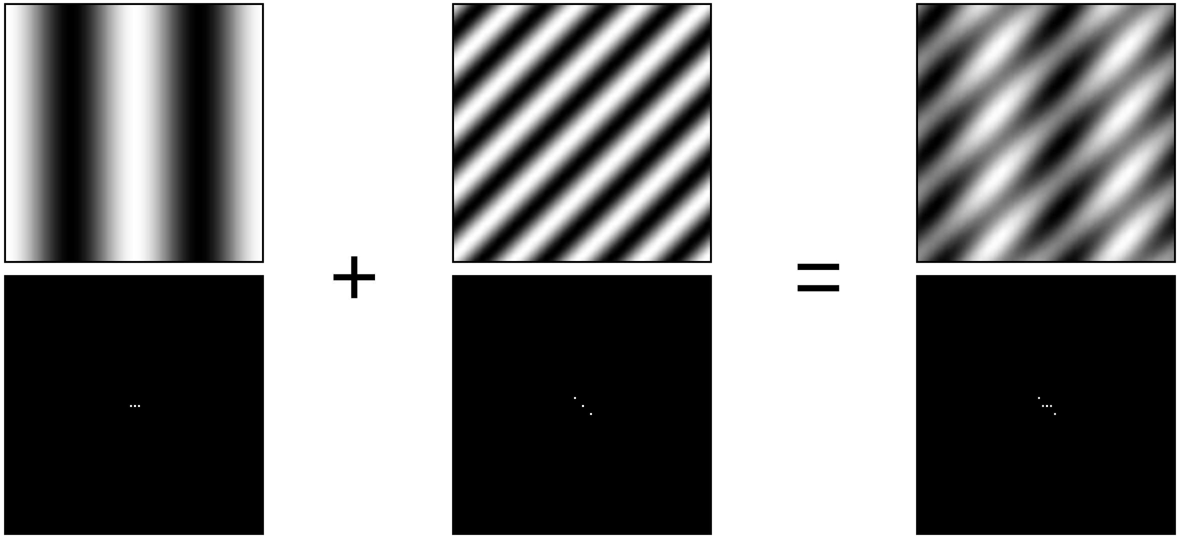
Intensity Image

Fourier Image



http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

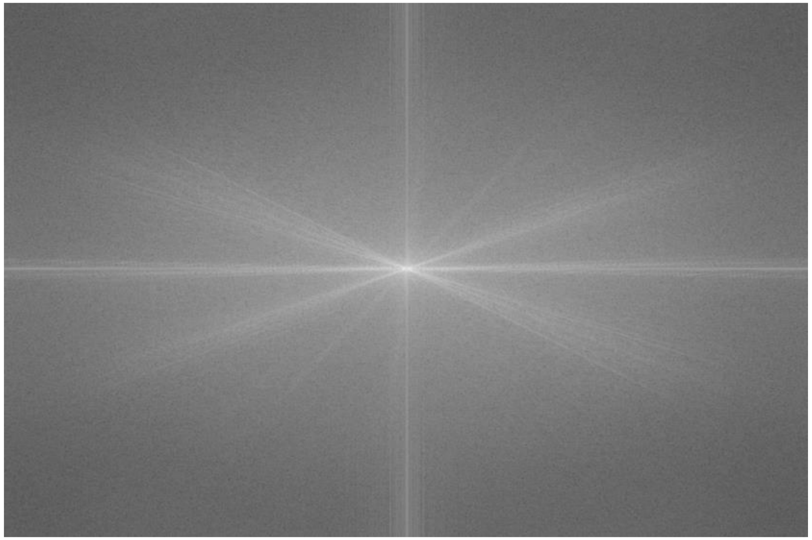
The Fourier transform: Filtering



http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering
More: http://www.cs.unm.edu/~brayer/vision/fourier.html

Understanding the Fourier transform of an image

Source image (J. Fourier)



Fourier power spectrum

Credit: K. Derpanis

Phase and Magnitude

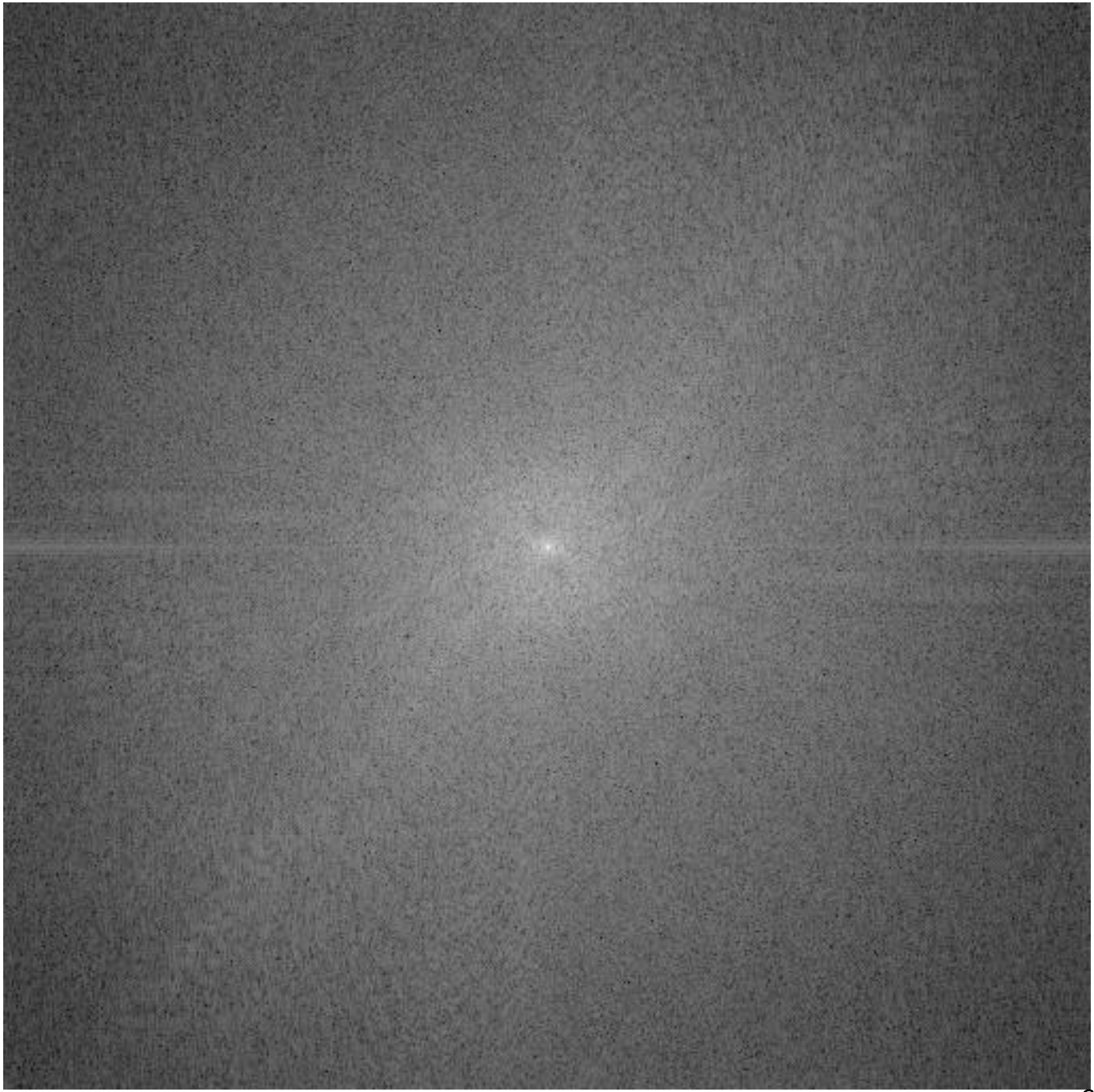
- Fourier transform of a real function is complex
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
- Magnitude is the magnitude of the complex transform

- Curious fact
  - all natural images have about the same magnitude transform
  - hence, phase seems to matter, but magnitude largely doesn’t
- Demonstration
  - Take two pictures, swap the phase transforms, compute the inverse - what does the result look like?

Credit: B. Freeman

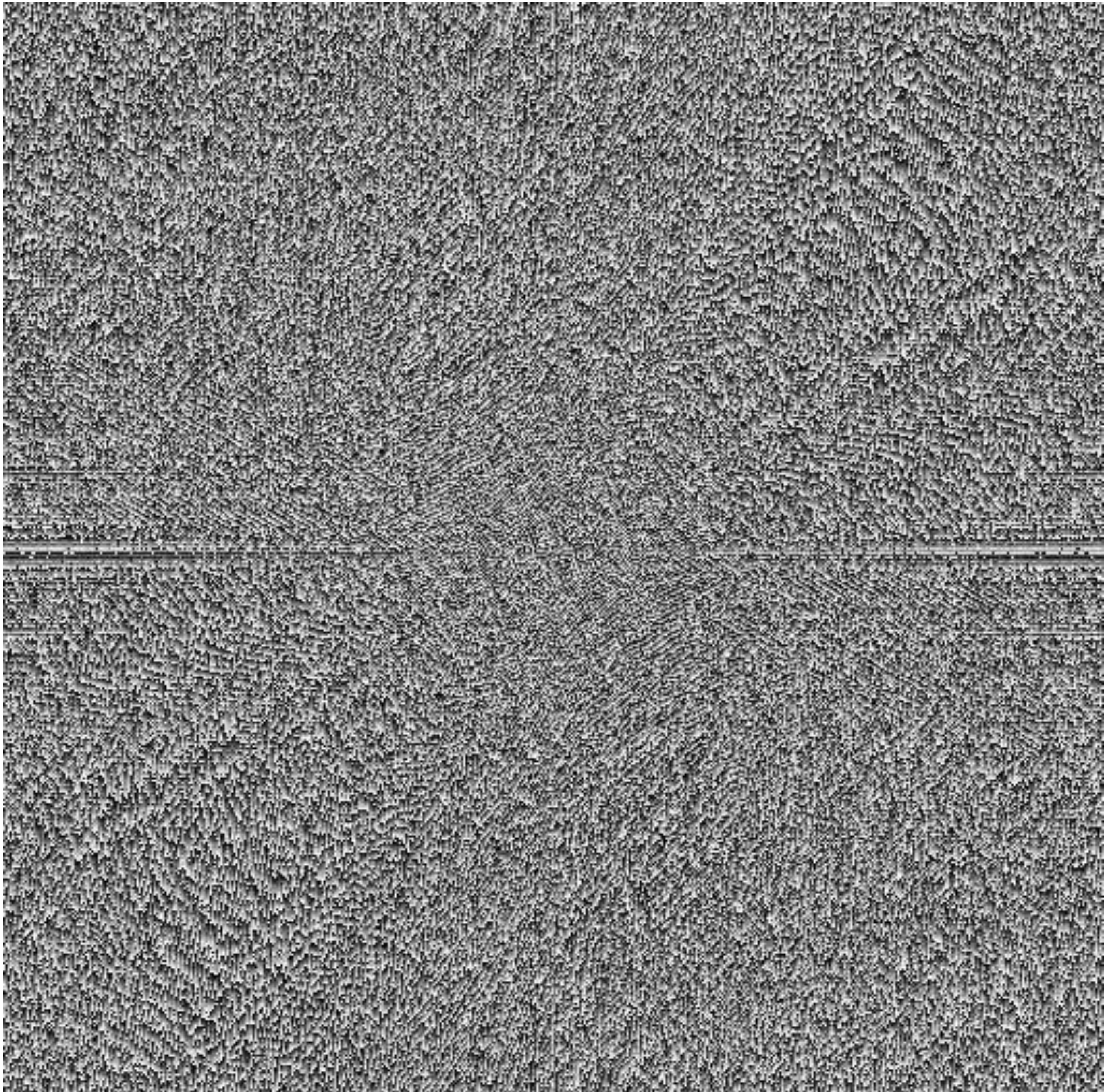
Credit: B. Freeman

This is the magnitude transform of the cheetah pic



Credit: B. Freeman

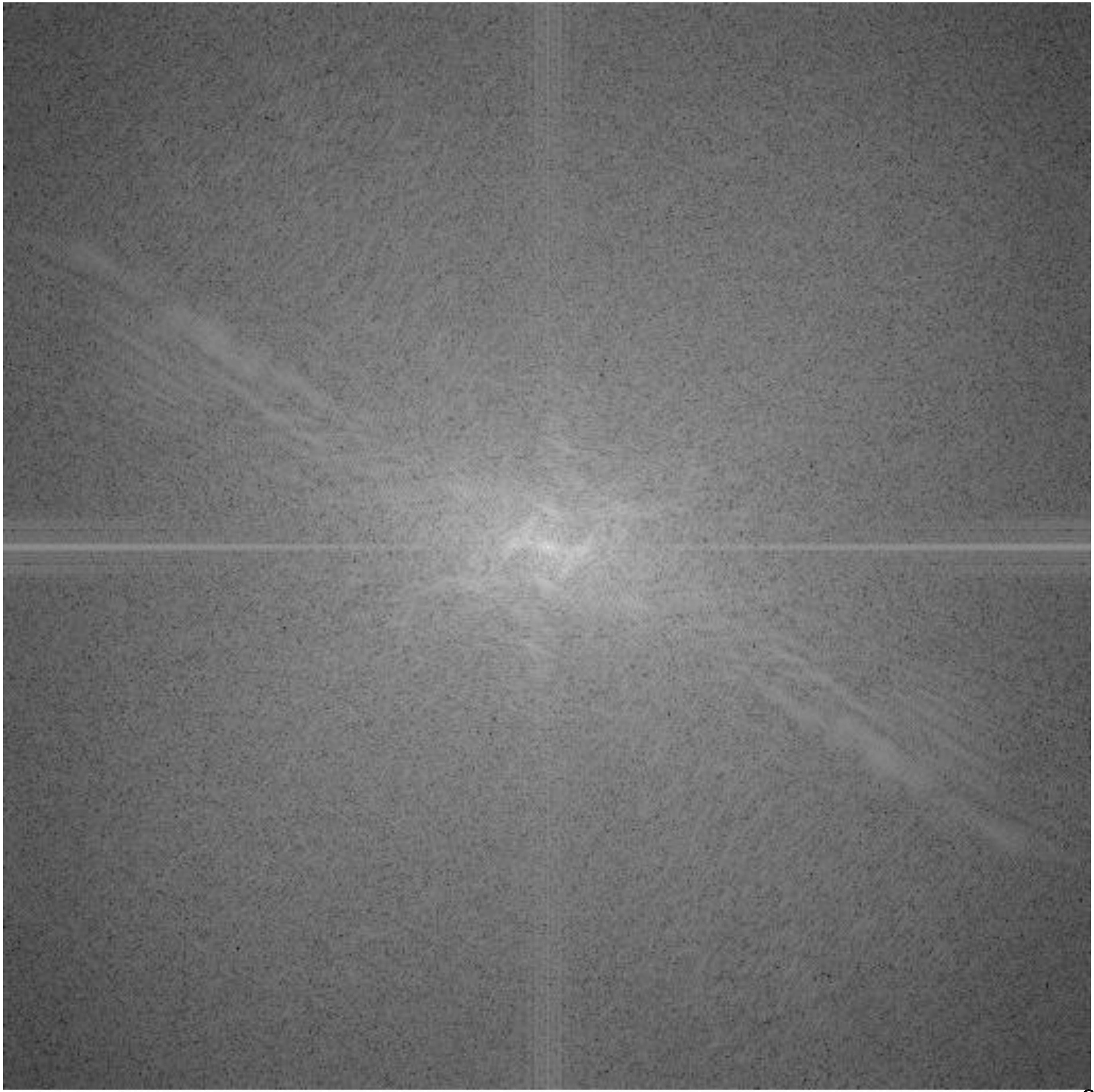
This is the phase transform of the cheetah pic



Credit: B. Freeman

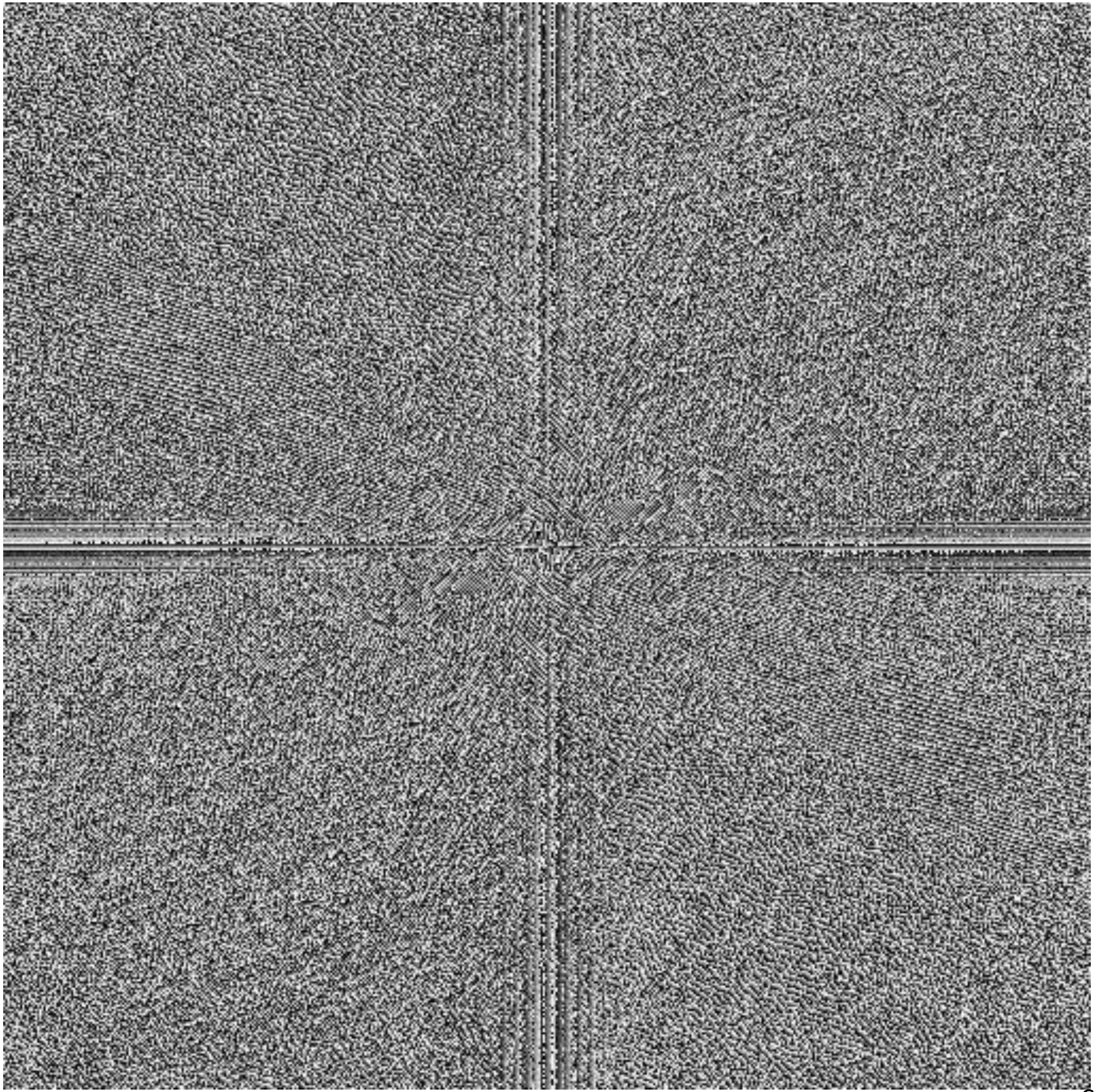

Credit: B. Freeman

This is the magnitude transform of the zebra pic



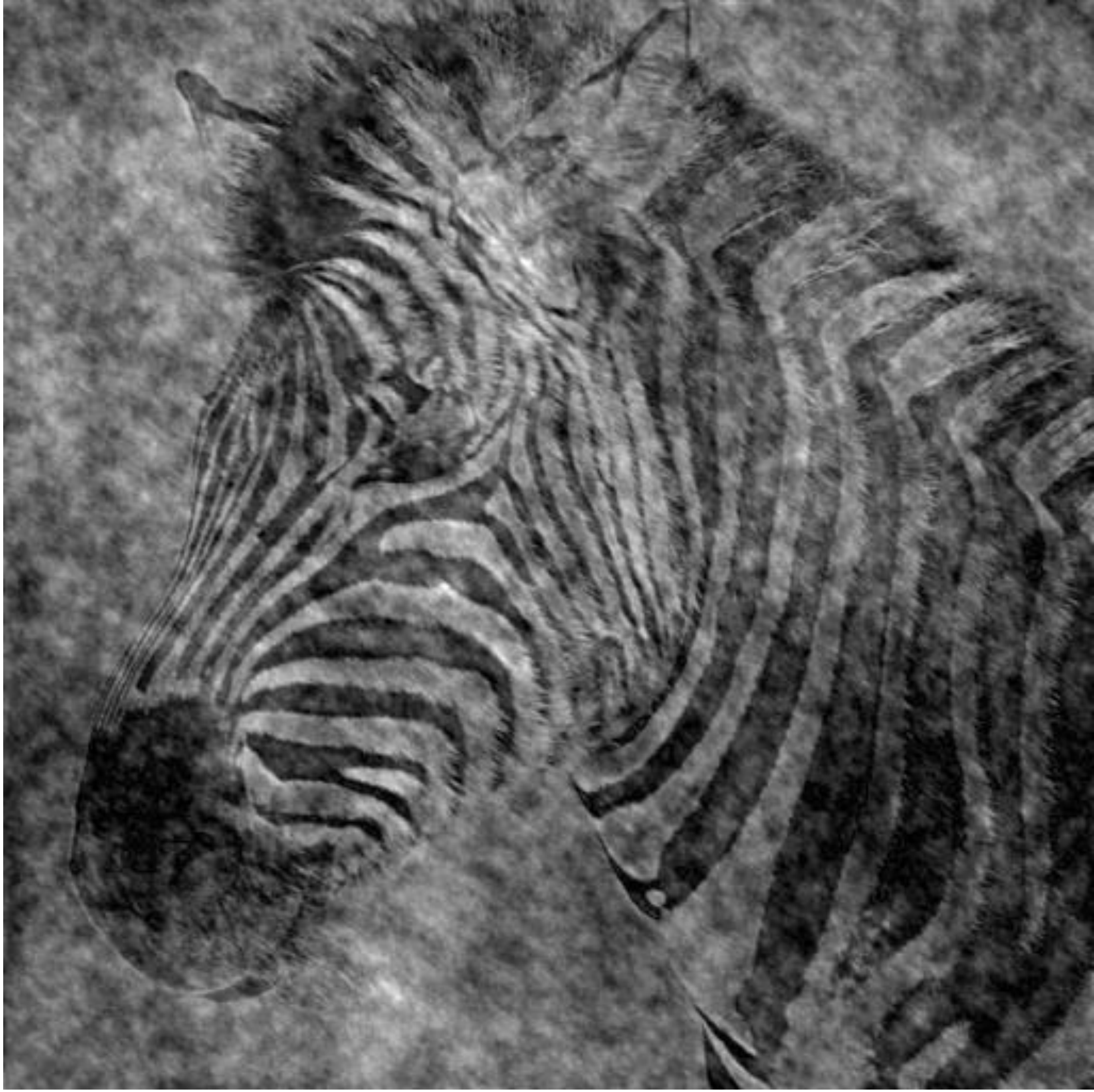
Credit: B. Freeman

This is the phase transform of the zebra pic



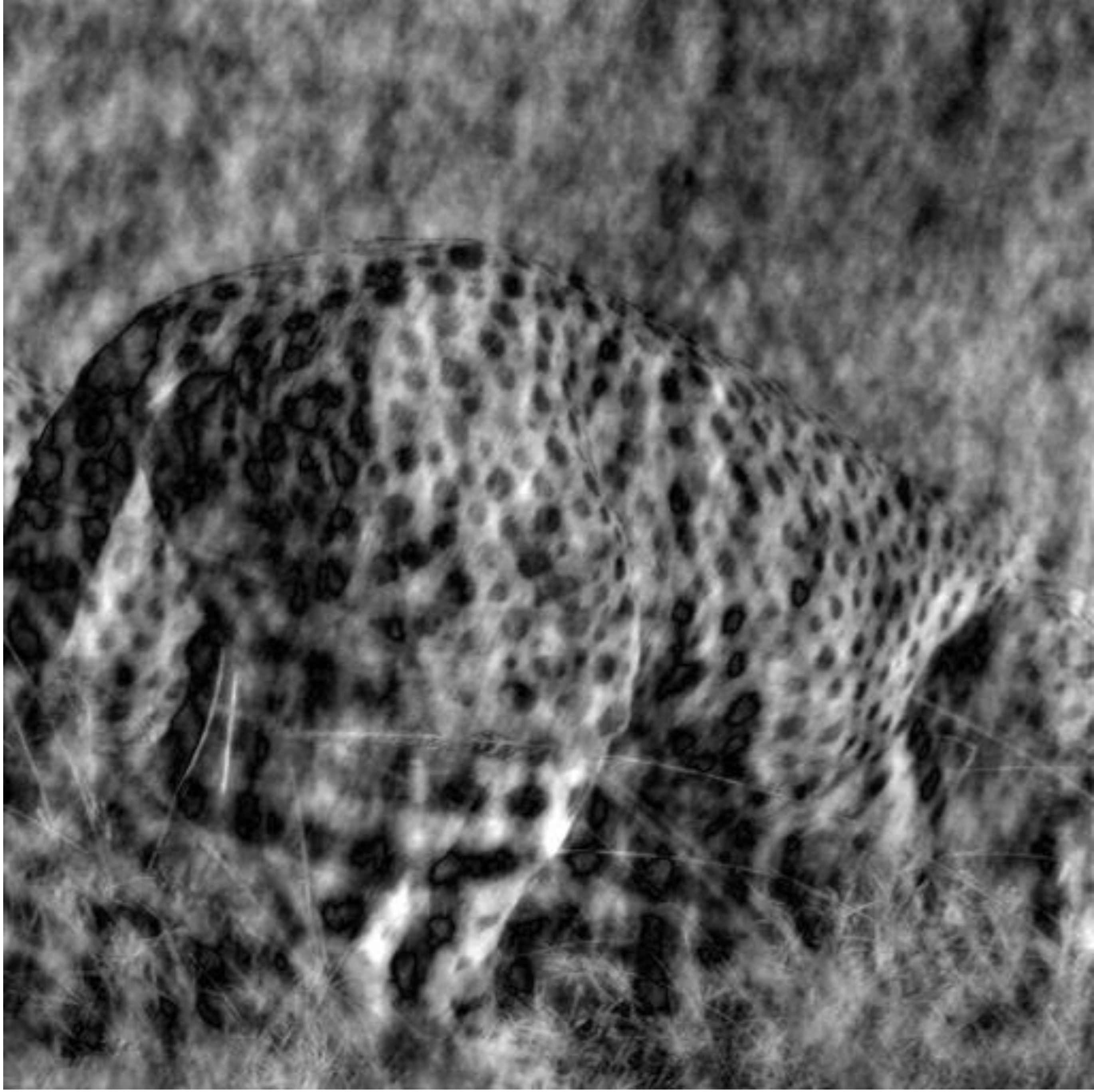
Credit: B. Freeman

Reconstruction with zebra phase, cheetah magnitude



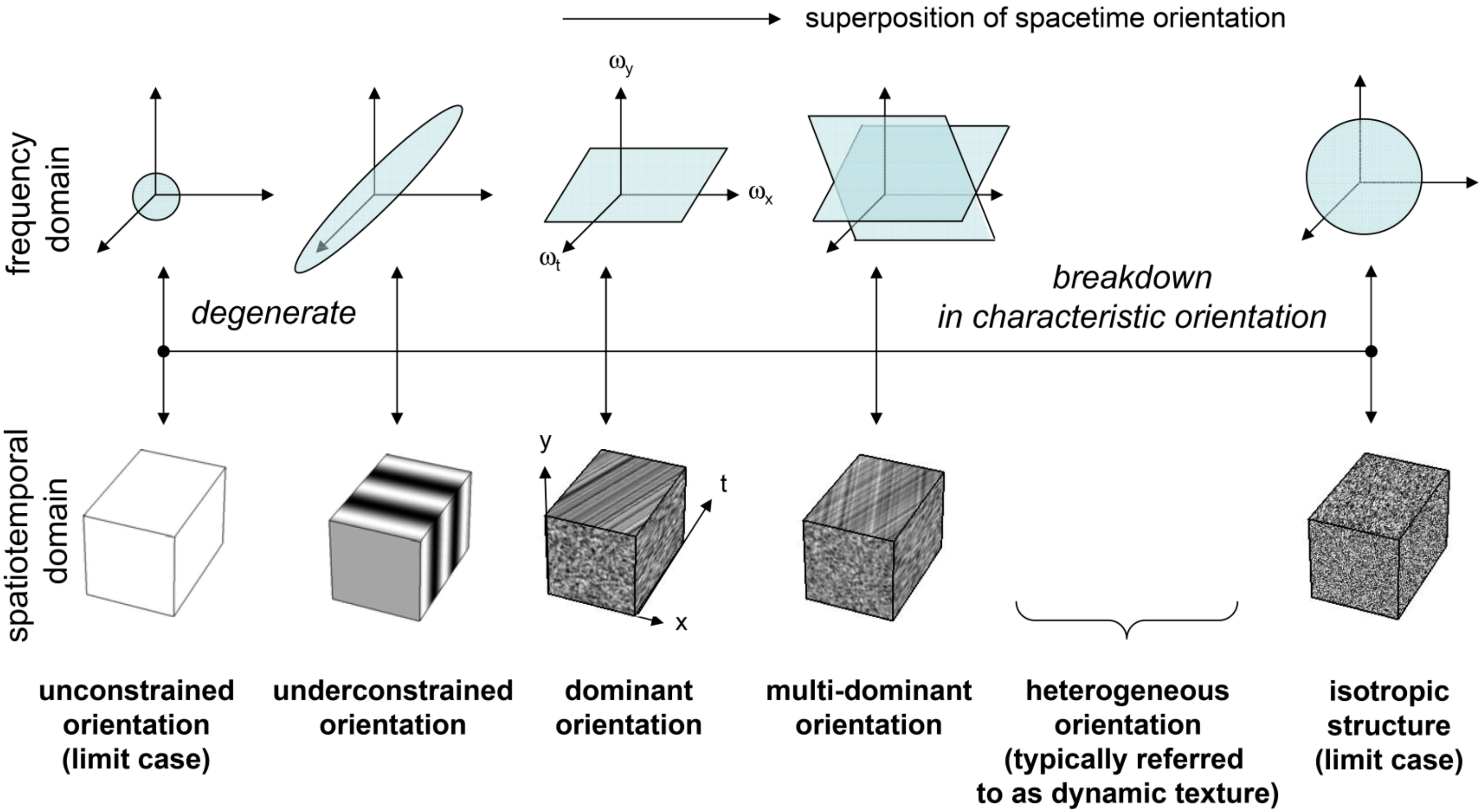
Credit: B. Freeman

Reconstruction with cheetah phase, zebra magnitude



Credit: B. Freeman

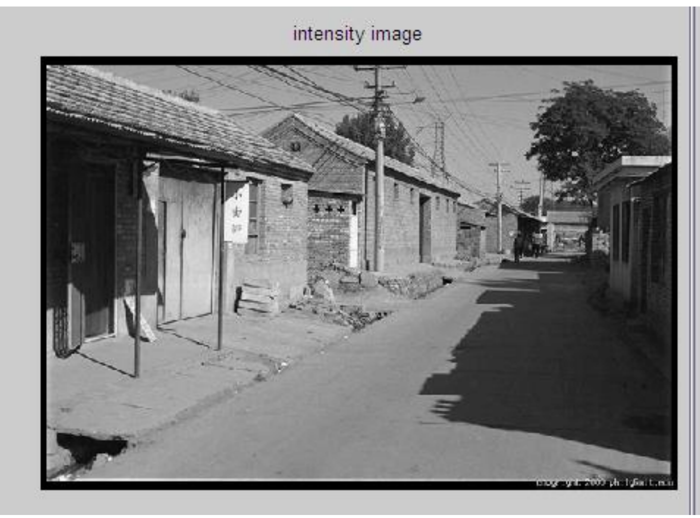
Extension to 3D



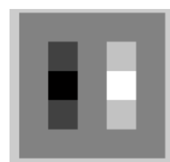
Credit: K. Derpanis

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

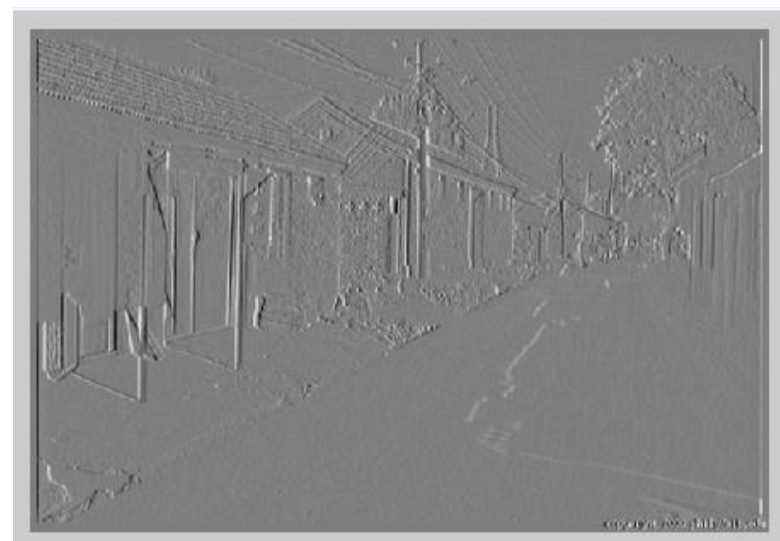
Filtering in spatial domain



*



=



Credit: J. Hays

The Convolution Theorem

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

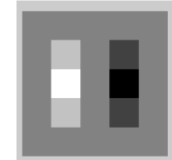
$$\mathrm{F}[g * h] = \mathrm{F}[g]\,\mathrm{F}[h]$$

- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

$$g * h = \mathrm{F}^{-1}[\mathrm{F}[g]\,\mathrm{F}[h]]$$

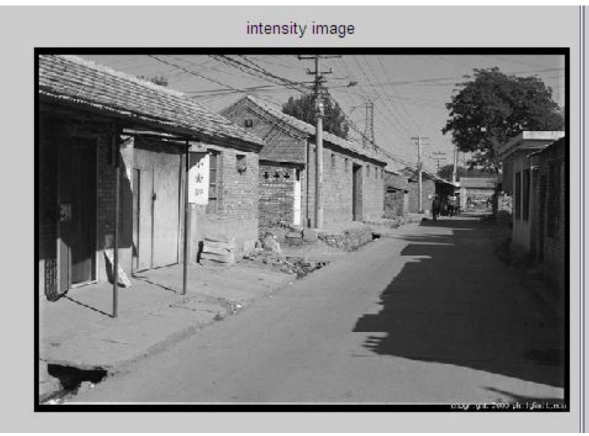
Credit: J. Hays

Filtering in frequency domain

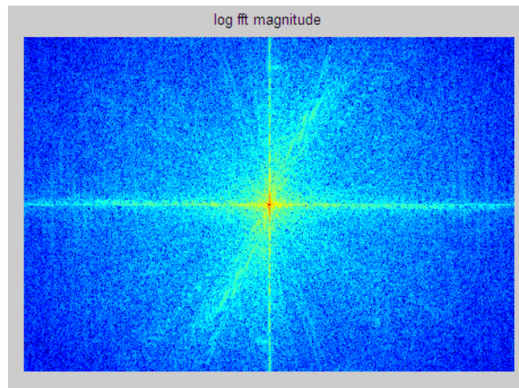


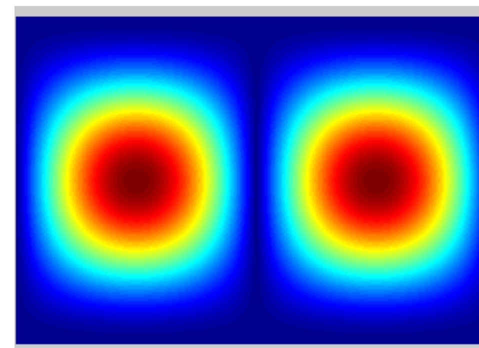
FFT

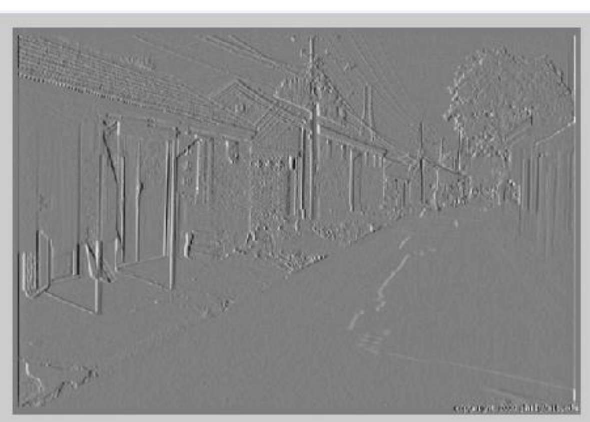
Fast Fourier Transform (FFT) = fast implementation



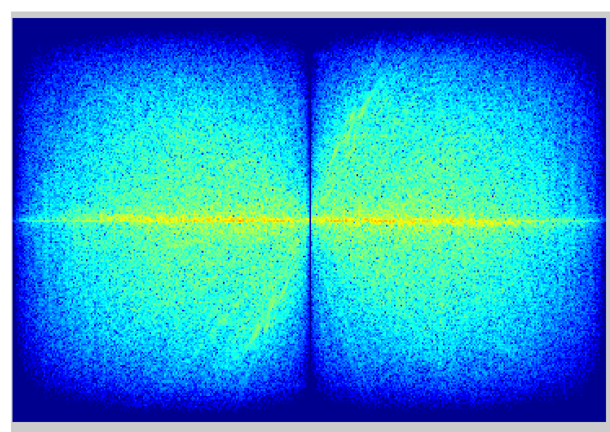
FFT







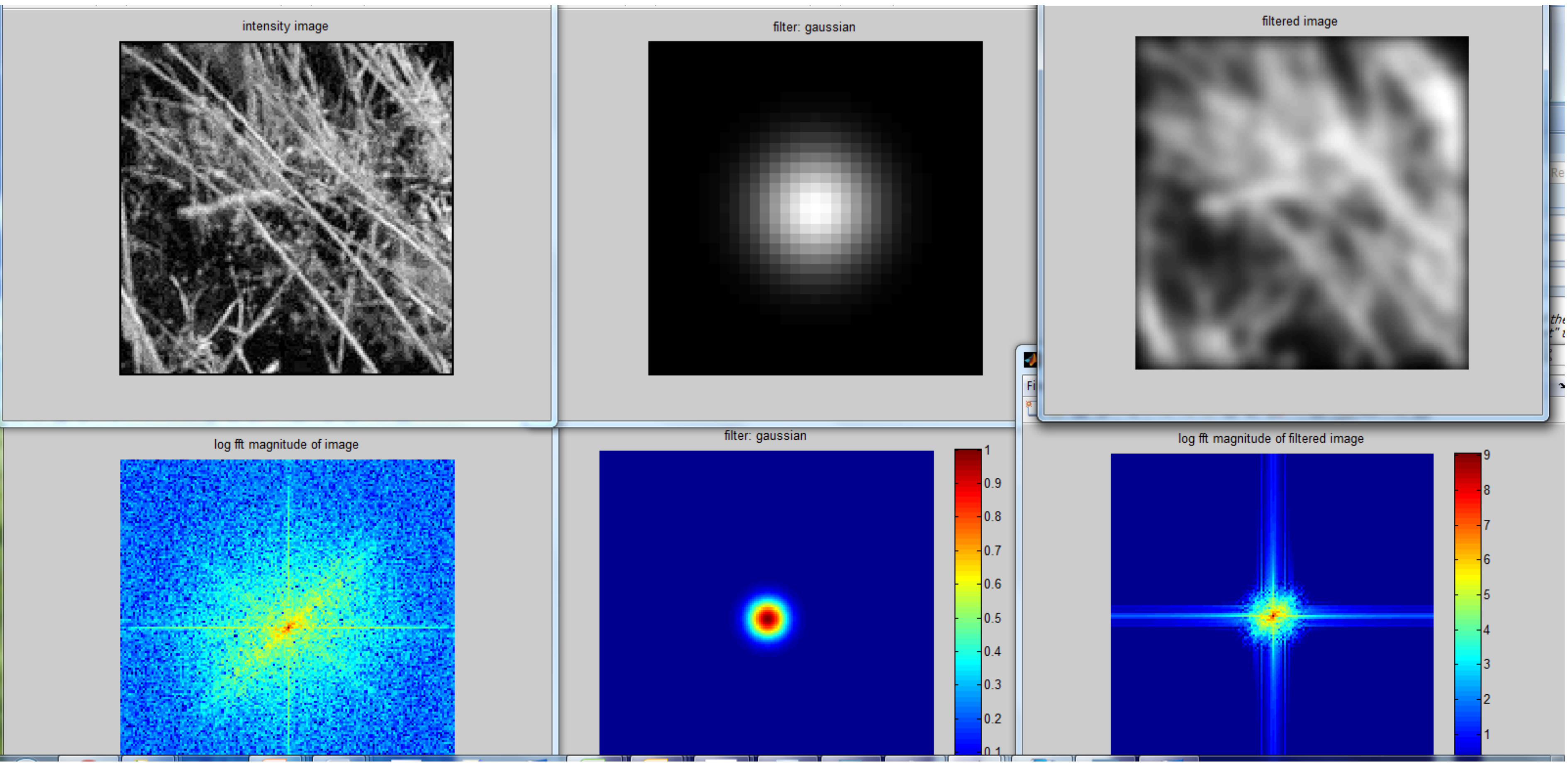
Inverse FFT



Credit: D. Hoiem

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

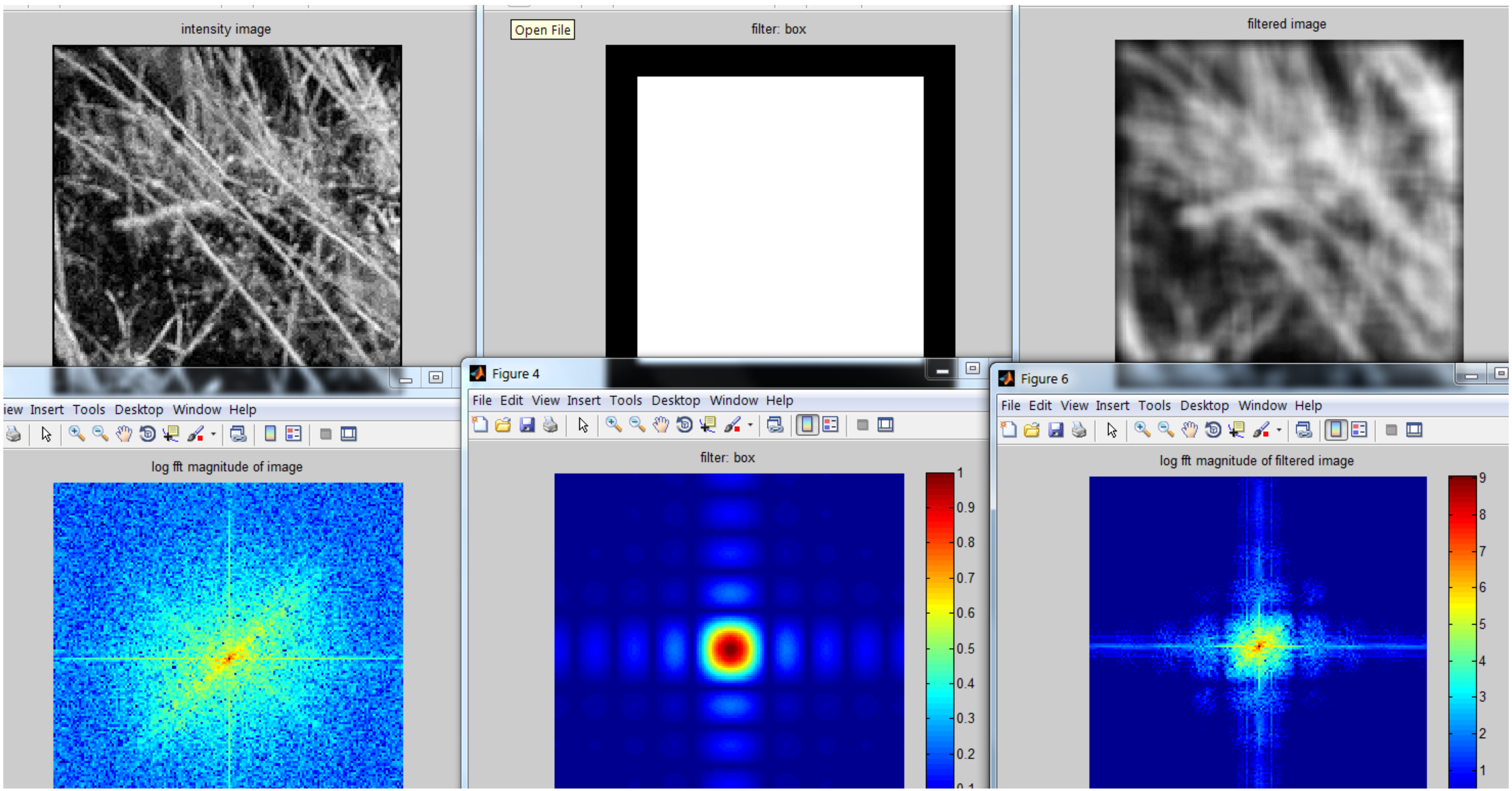
Gaussian



Credit: D. Hoiem

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

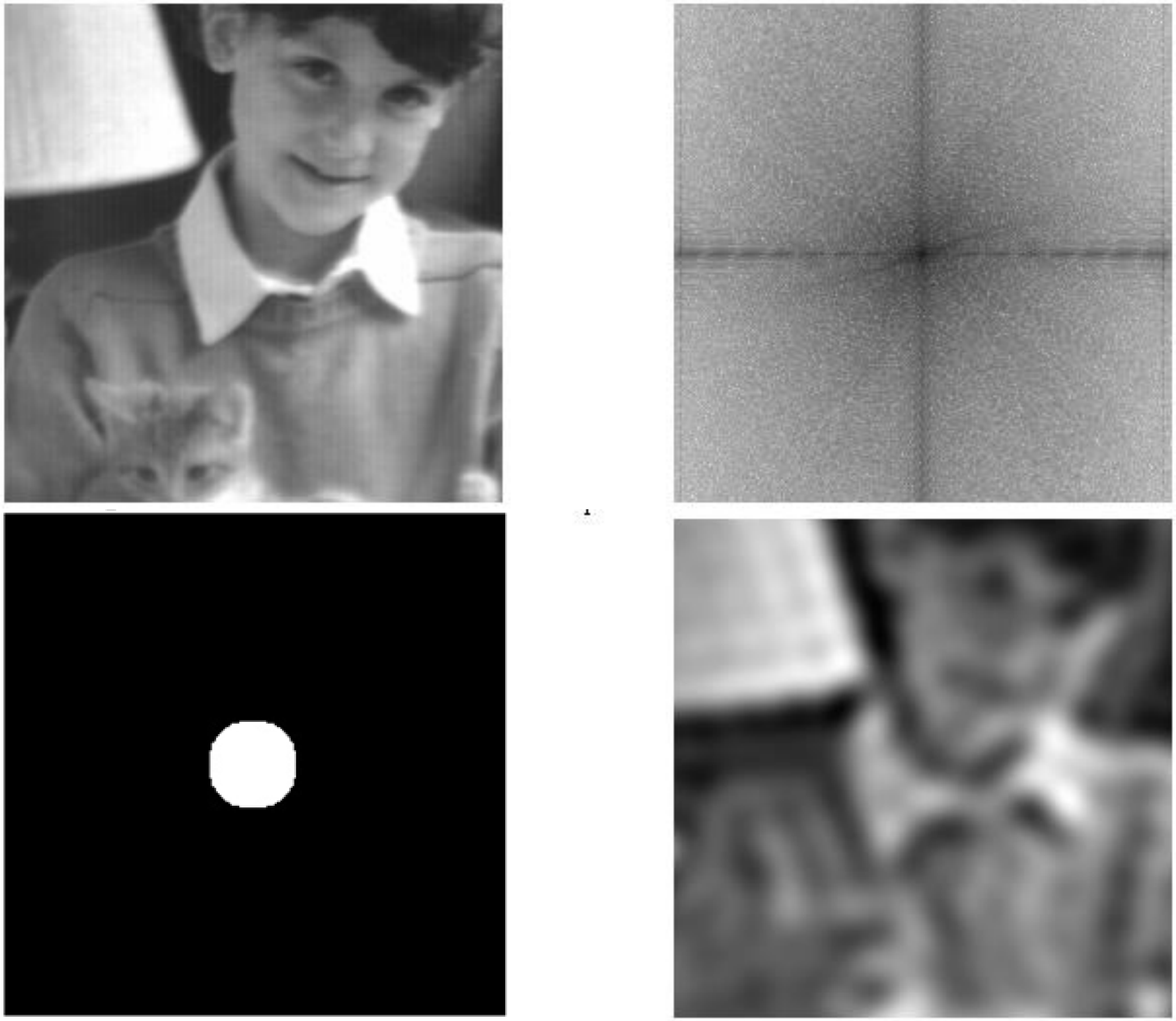
Box Filter



Credit: D. Hoiem

Low pass filtering

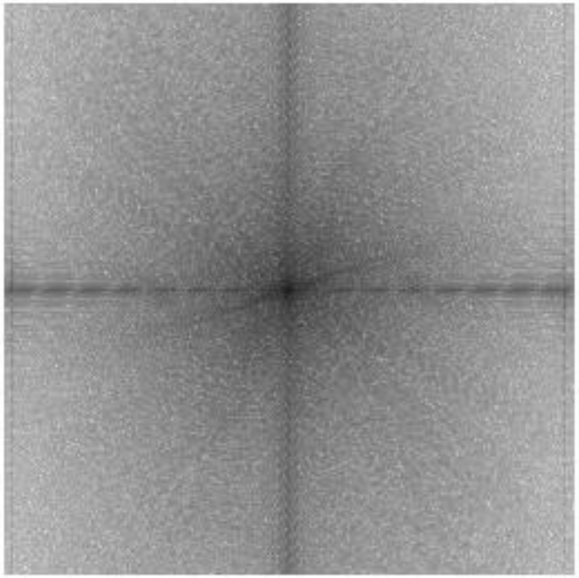
http://www.reindeergraphics.com

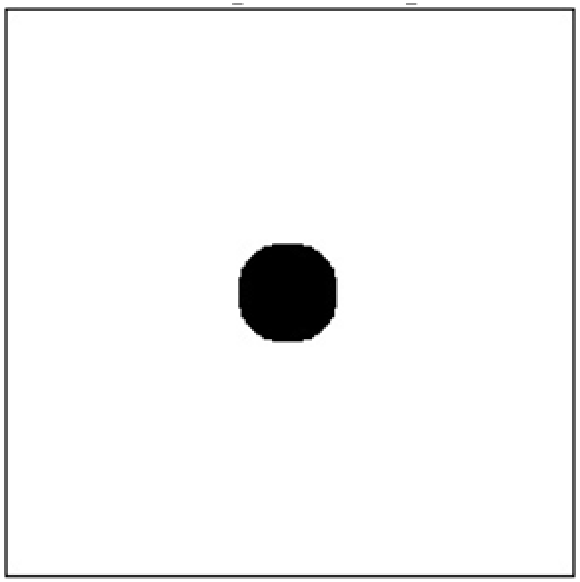


Credit: R. Fergus

High pass filtering

http://www.reindeergraphics.com

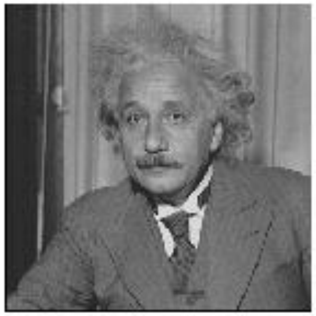
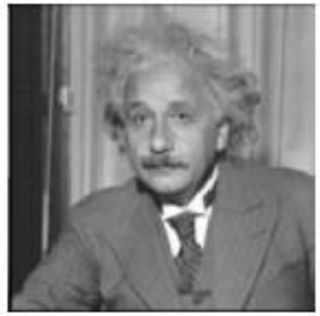
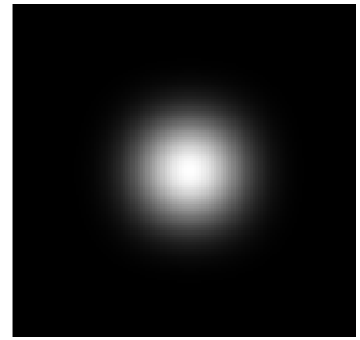




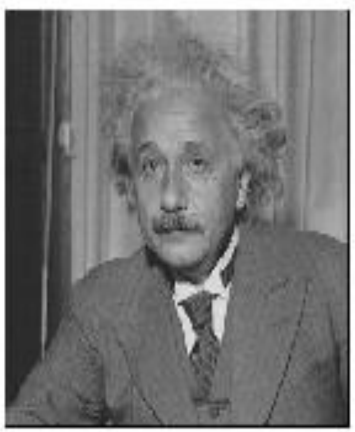
Credit: R. Fergus

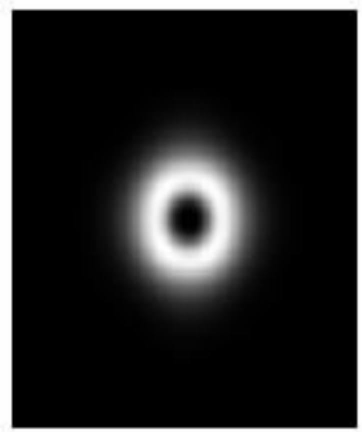
Low-pass, Band-pass, High-pass filters

low-pass:

High-pass / band-pass:

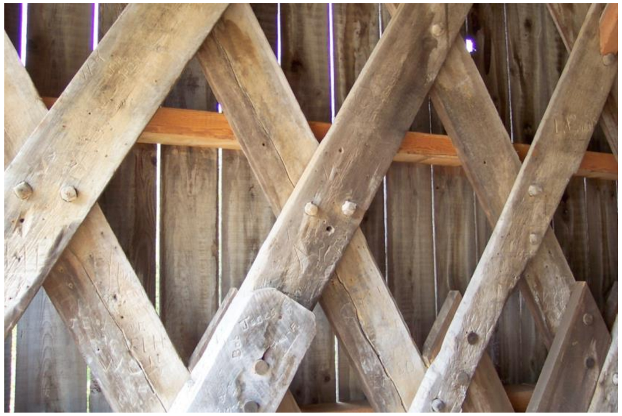


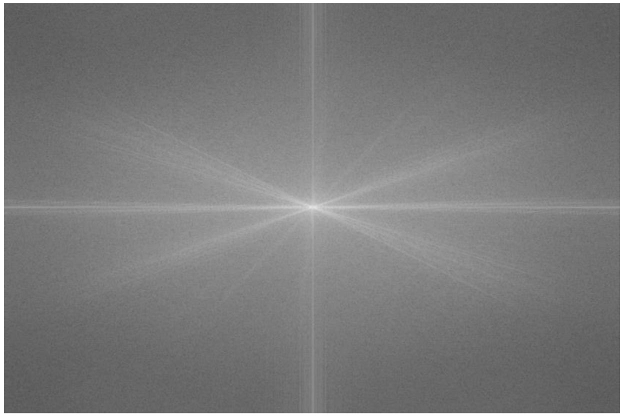


Credit: A. Efros

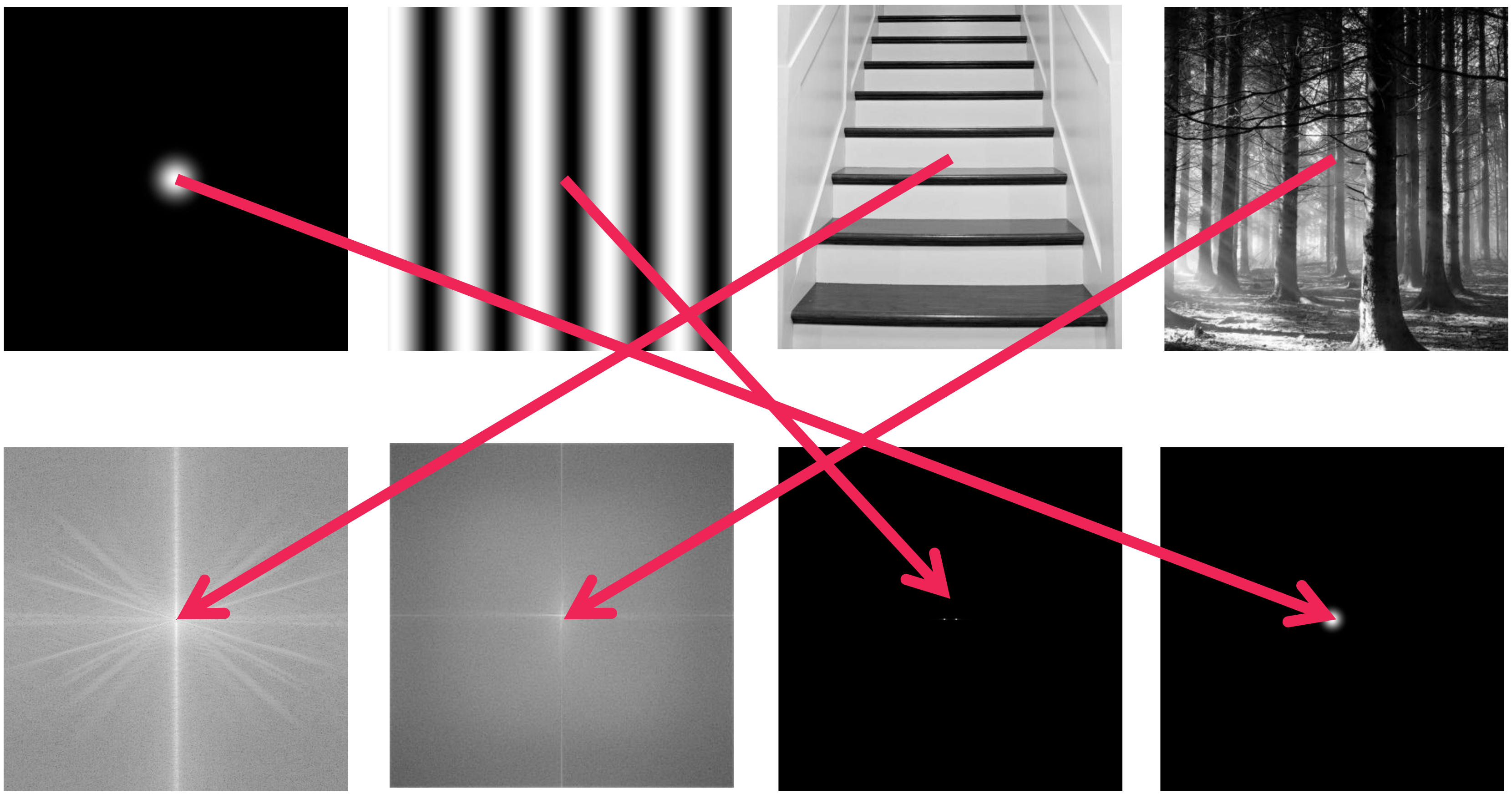
Why is the Frequency domain useful for us?

- The linear **convolution** operation can be understood from a different angle
- It can be performed very efficient using a clever implementation (FFT)
- The Frequency domain provides an alternative way to understand and manipulate the content of images





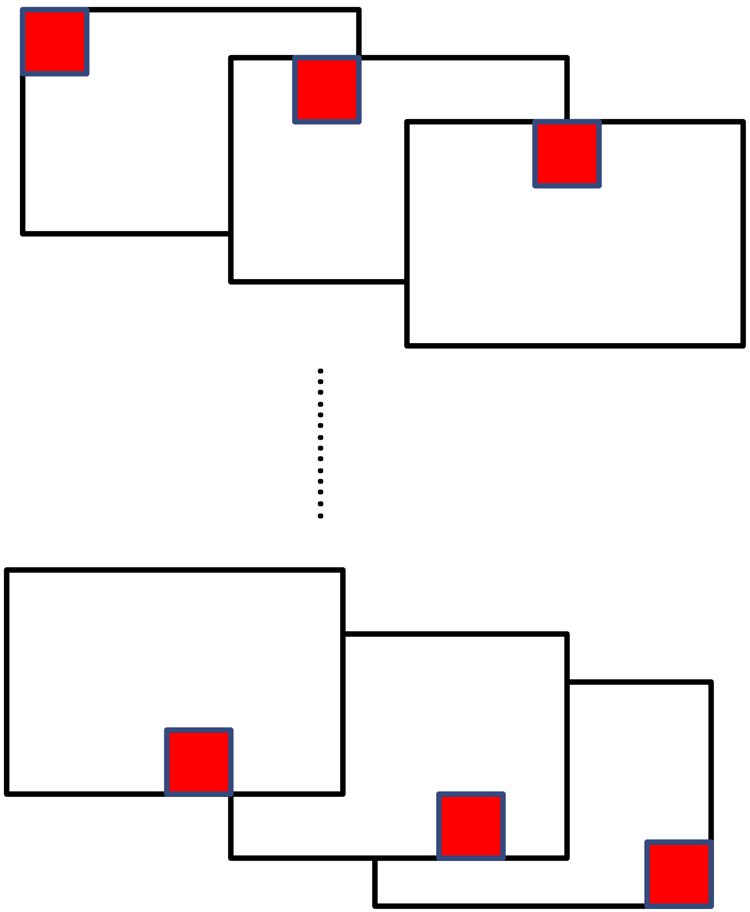
Match the spatial domain image to the Fourier magnitude image



Credit: K. Derpanis

Spatial Domain

Basis functions:



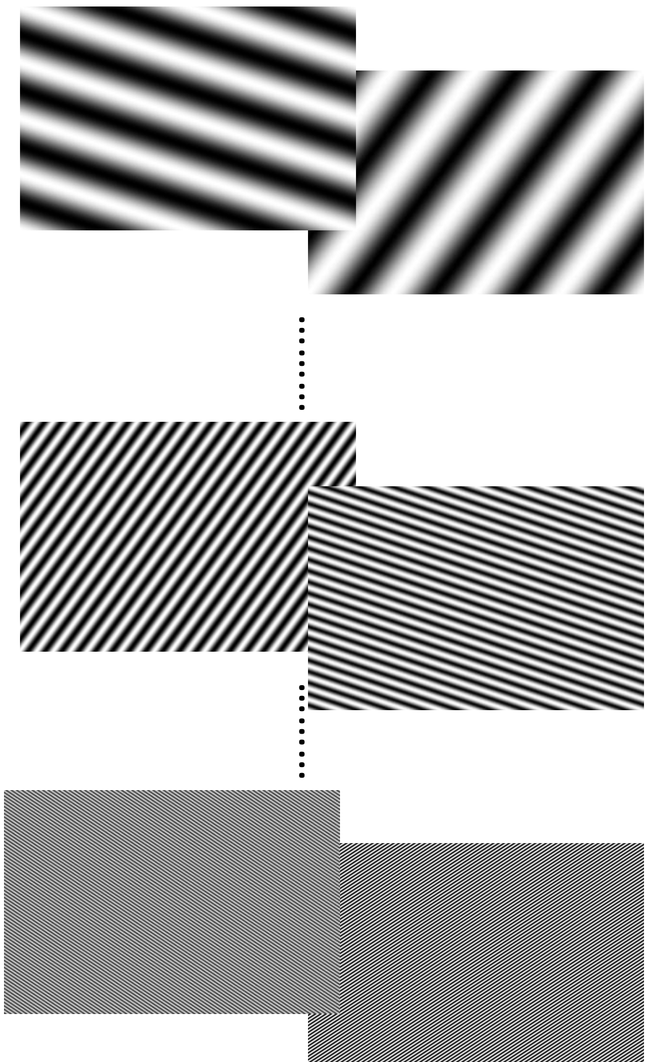
Tells you *where* things are….

… but no concept of *what* it is

Credit: R. Fergus

Fourier domain

Basis functions:



Tells you *what* is in the image....

... but not *where* it is

Credit: R. Fergus

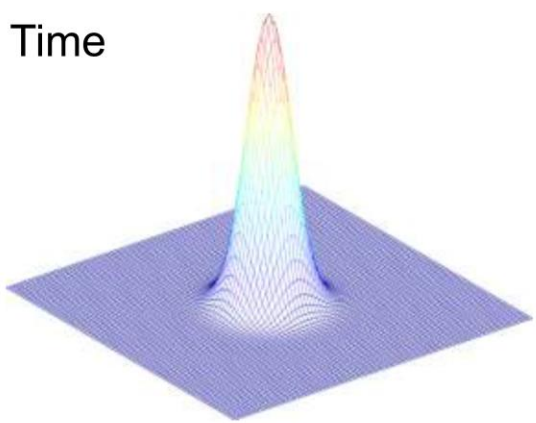
Modulation property and Gabor filters

Modulation property: $$f(x) \leftrightarrow F(\omega)$$

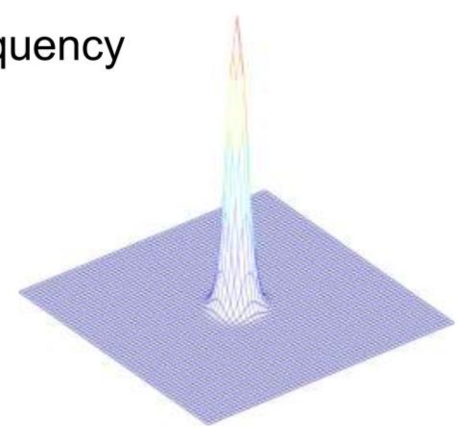
$$f(x)e^{j\omega_c x} \leftrightarrow F(\omega - \omega_c)$$

Gaussian: $$\frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \leftrightarrow e^{-\frac{(\omega_x^2+\omega_y^2)\sigma^2}{2}}$$

Time



Frequency



Credit: I. Kokkinos

Modulation property and Gabor filters

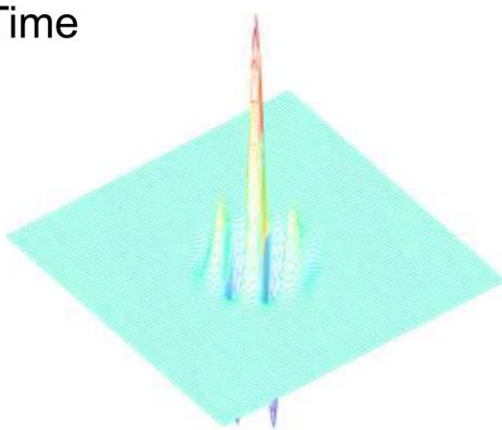
Modulation property: $$f(x) \leftrightarrow F(\omega)$$

$$f(x)e^{j\omega_c x} \leftrightarrow F(\omega - \omega_c)$$

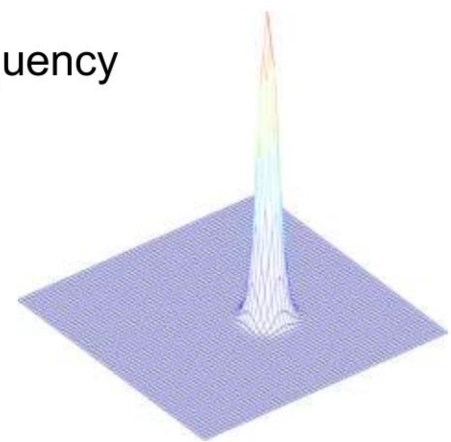
Gaussian: $$\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}} \leftrightarrow e^{-\frac{(\omega_x^2+\omega_y^2)\sigma^2}{2}}$$

Gabor: $$\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}e^{j(\omega_x^c x+\omega_y^c y)} \leftrightarrow e^{-j\frac{\sigma^2((\omega_x-\omega_x^c)^2+(\omega_y-\omega_y^c)^2)}{2}}$$

Time



Frequency

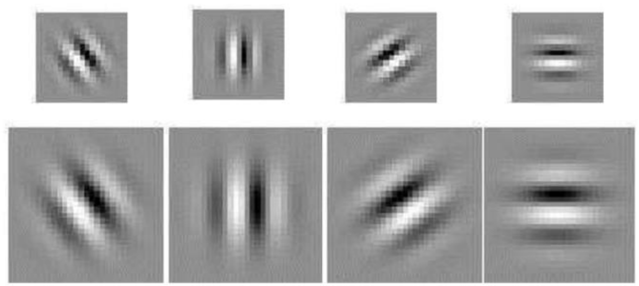


Credit: I. Kokkinos

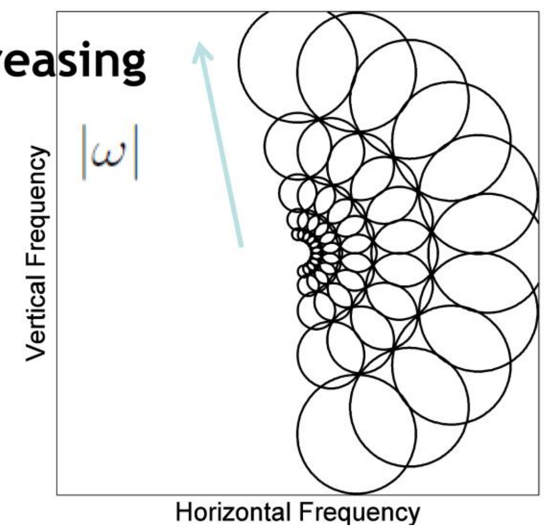
2D Gabor filterbank and texture analysis

Consider many combinations of $|\omega|$ and $\angle\omega$

Increasing $|\omega|$



Increasing $|\omega|$



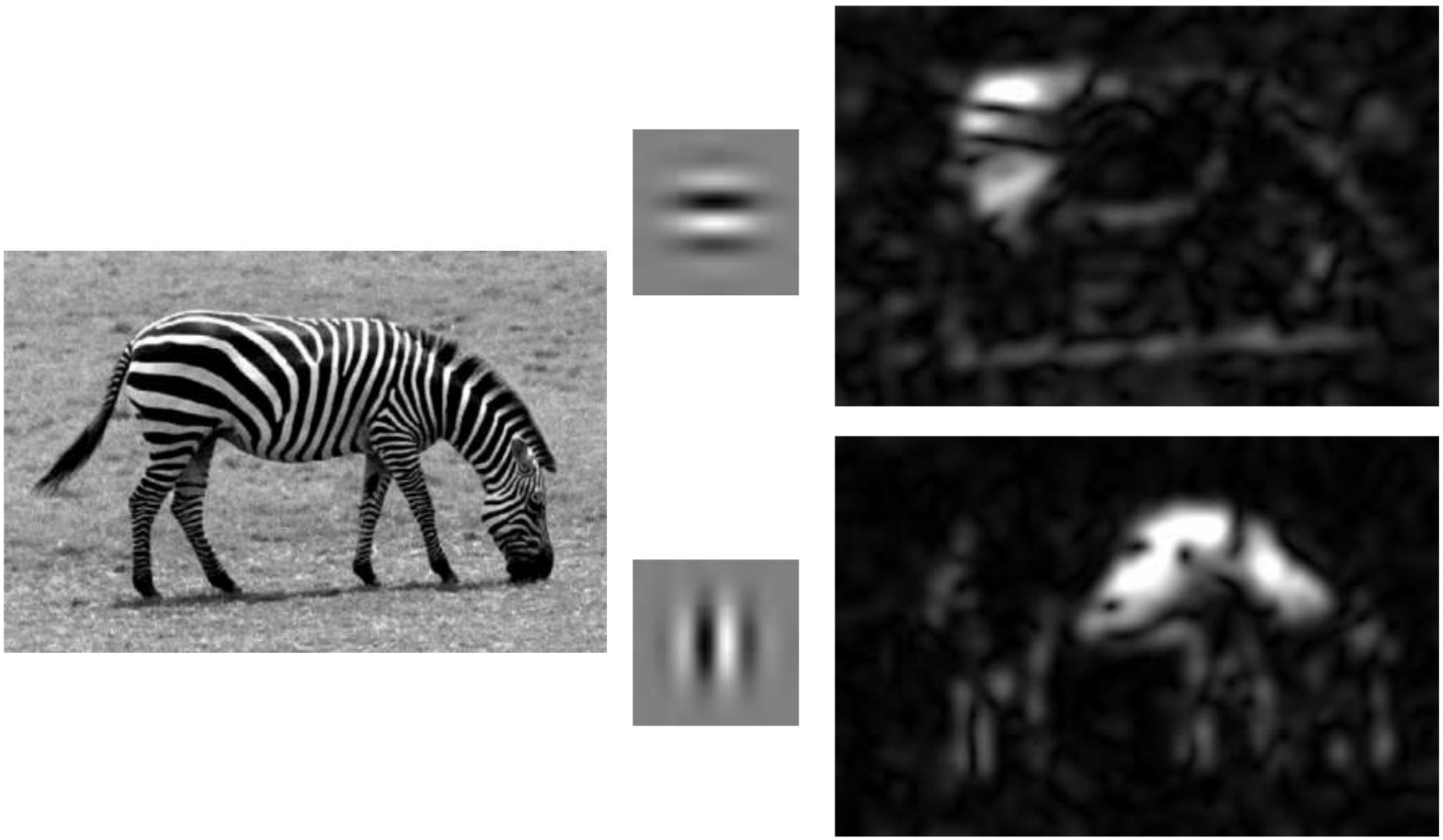
Frequency responce isocurves

Credit: I. Kokkinos

2D Gabor filterbank and texture analysis

Credit: I. Kokkinos

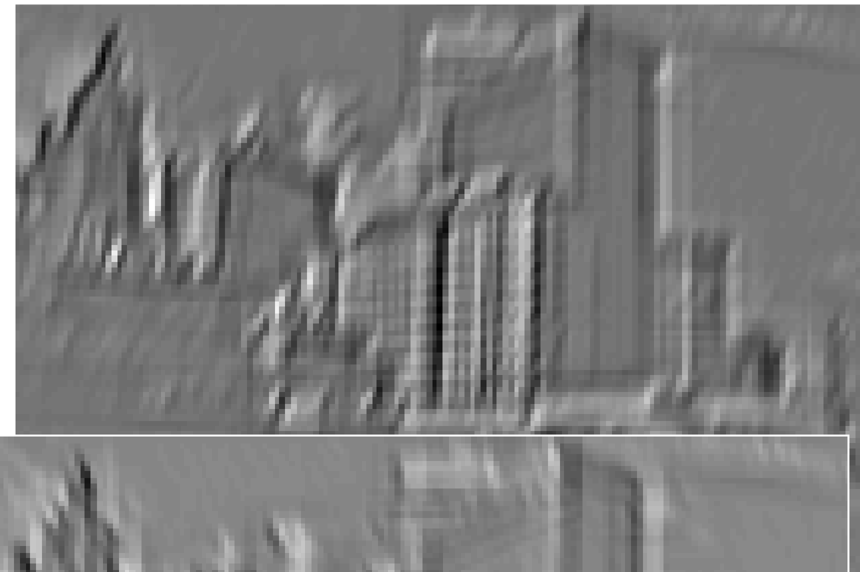
2D Gabor filterbank and texture analysis



Credit: I. Kokkinos

Summary: Images as a composition of local parts
Filtering example

- Why filtering?
  - Statistics of images look similar at different locations
  - Dependencies are very local
  - Filtering is an opteration with translation *equivariance*

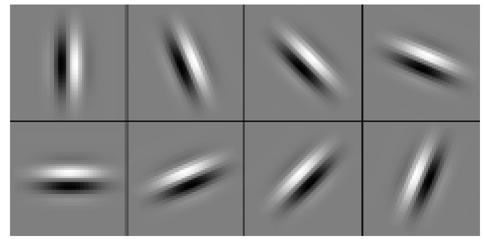


Credit: R. Fergus

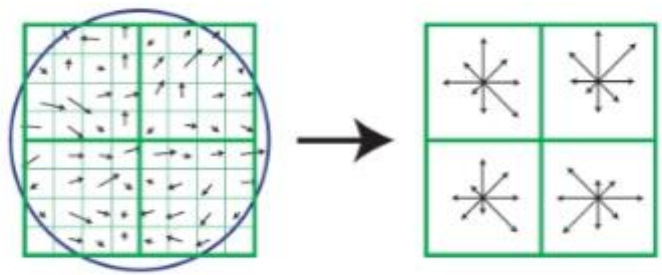
Summary: Compare: SIFT Descriptor

Image Pixels

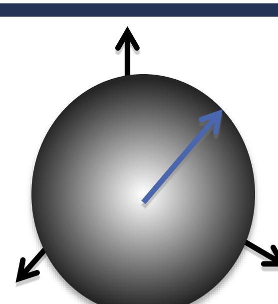
Apply
Gabor filters



Spatial pool
(Sum)



Normalize to unit
length



Feature Vector

Credit: R. Fergus

Summary

- Linear filtering and the importance of **convolution**
  - Apply a filtermask to the local neighborhood at each pixel in the image
  - The filtermask defines how to combine values from neighbors.
  - Can be used for
    - Extract **intermediate representations** to abstract images by higher-level "**features**", for further processing (i.e., preserve the useful information only and discard redundancy)
    - Image **modification,** e.g., to reduce noise, resize, increase contrast, etc.
    - Match **template** images (e.g. by correlating two image patches)
- Image filtering in the frequency domain
  - Provides a nice way to illustrate the effect of linear filtering
    - Filtering is a way to modify the frequencies of images
  - Efficient signal filtering is possible in that domain
  - The frequency domain offers an alternative way to understanding and manipulating the image.