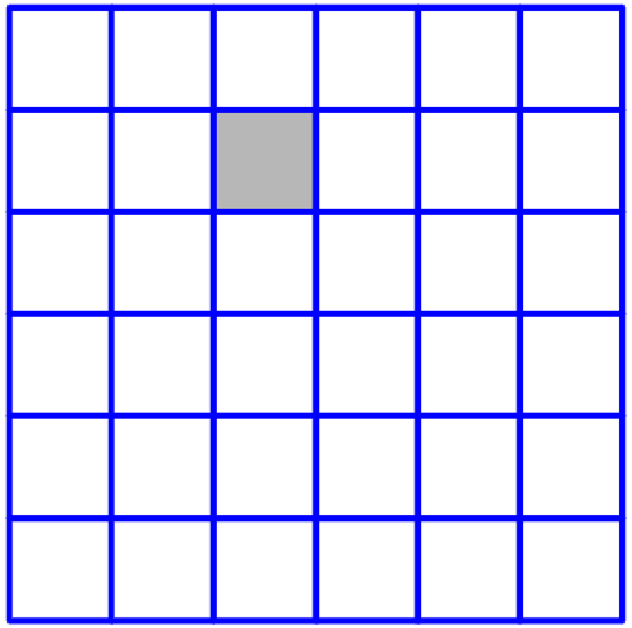
CMSC 491/691

Lecture 3

Image Filtering

An image is a matrix of pixels

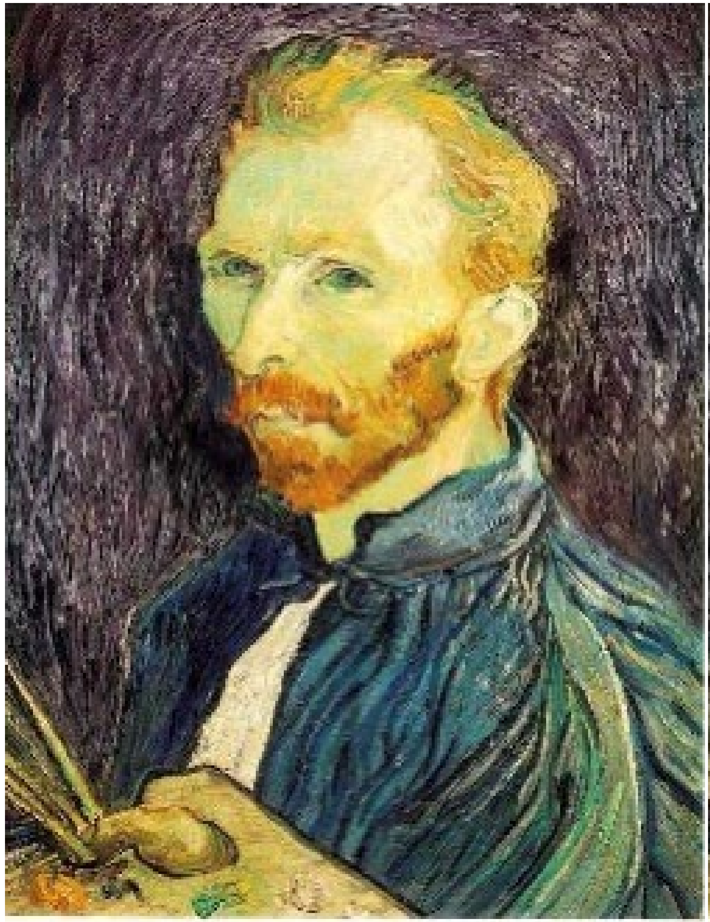


F[x,y]

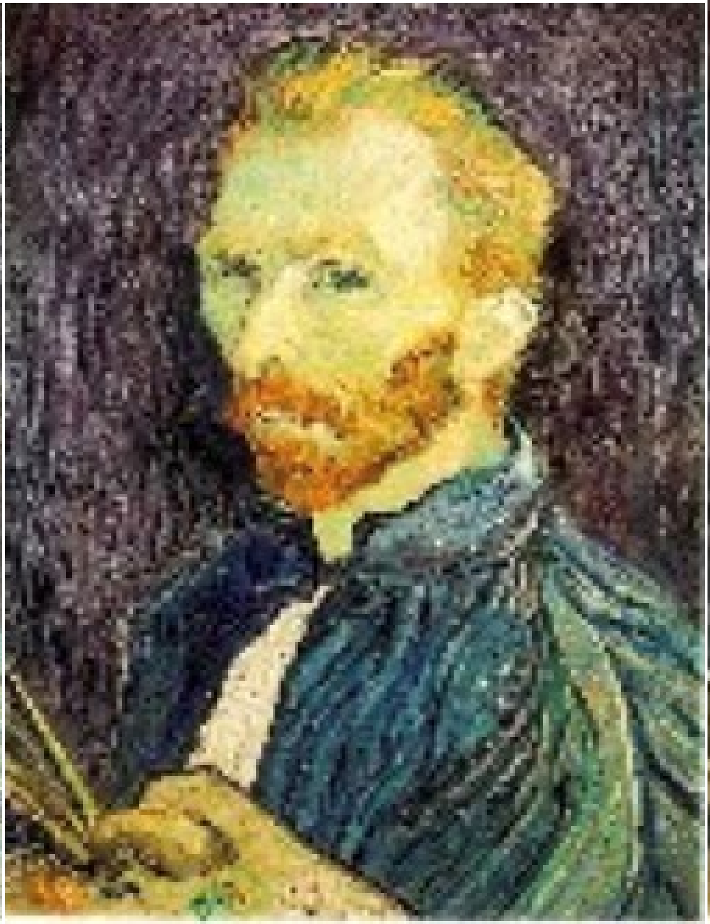
An array of numbers ("pixels")

x,y are integer column/row indices

Subsampling

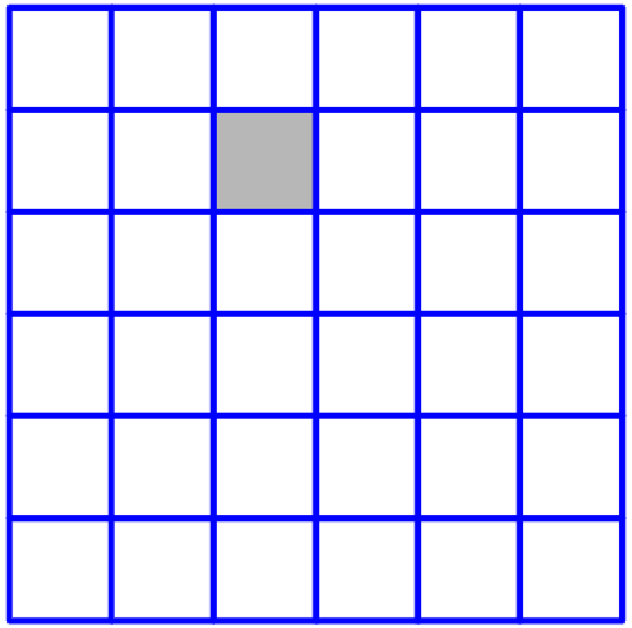


1/2



1/4 (2x zoom)

1/8 (4x zoom)

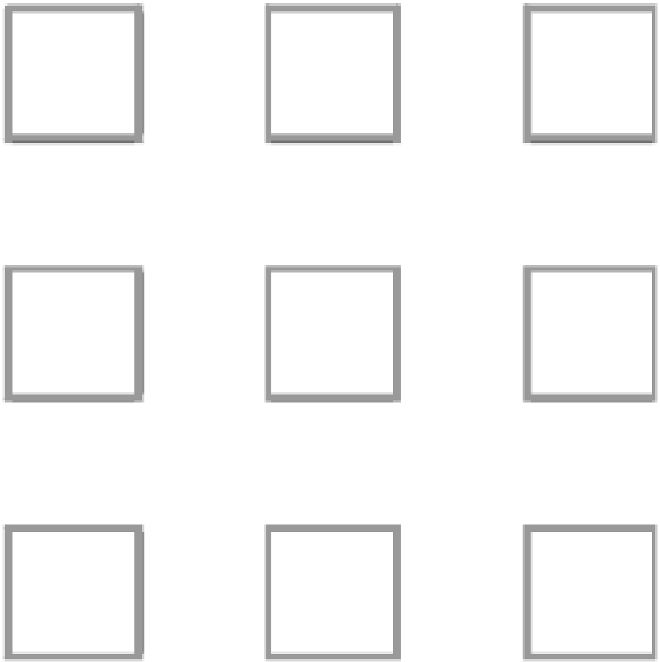


F[x,y]

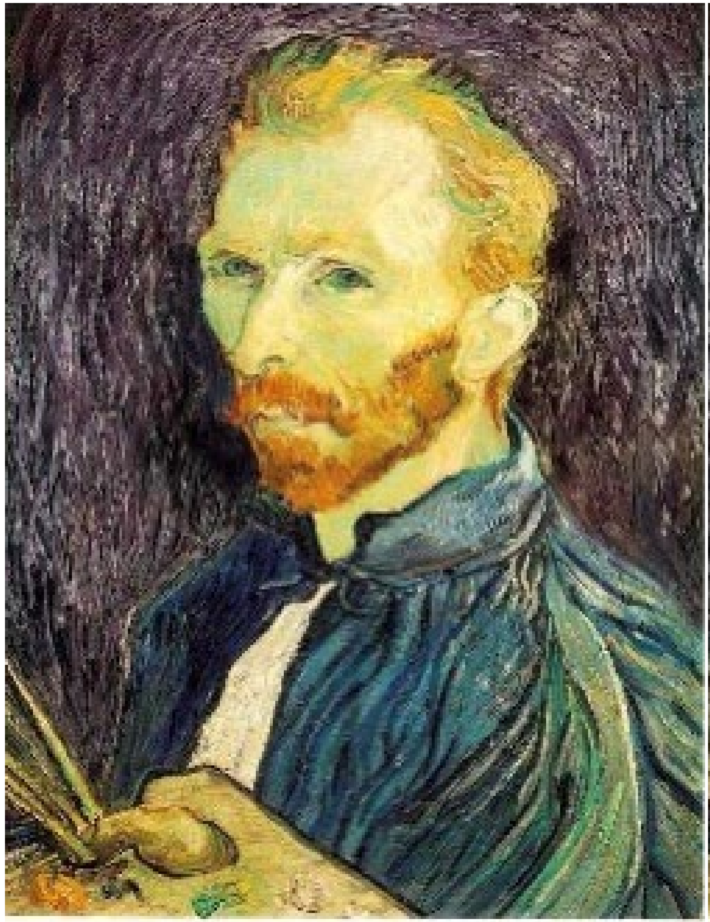
An array of numbers (“pixels”)

x,y are integer column/row indices

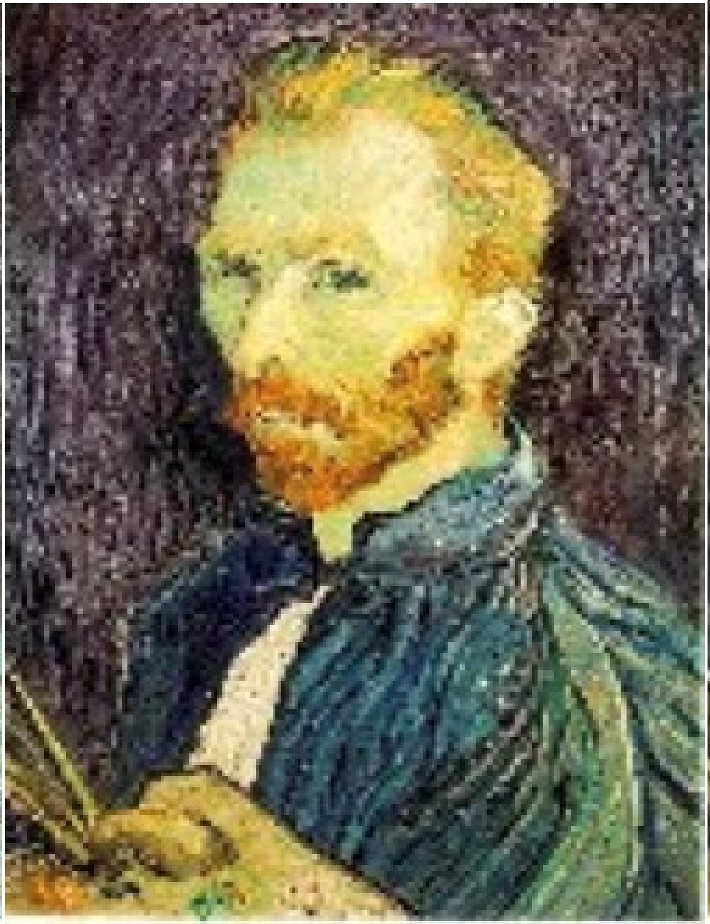
Throw away even rows and columns



Subsampling



1/2

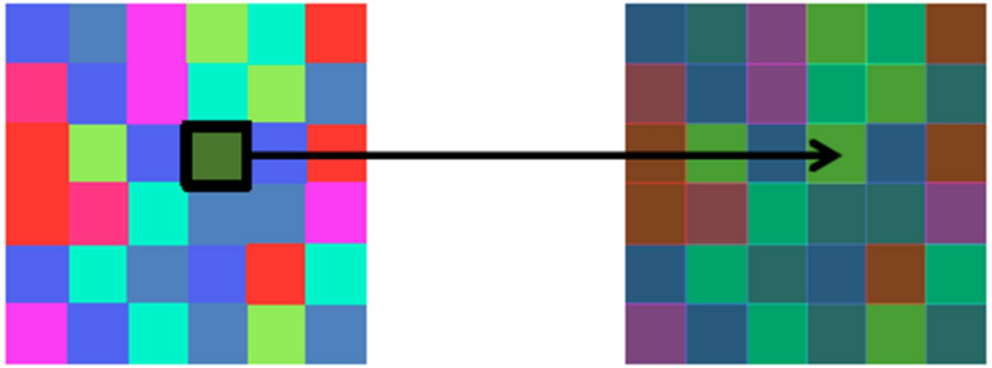


1/4 (2x zoom)

1/8 (4x zoom)

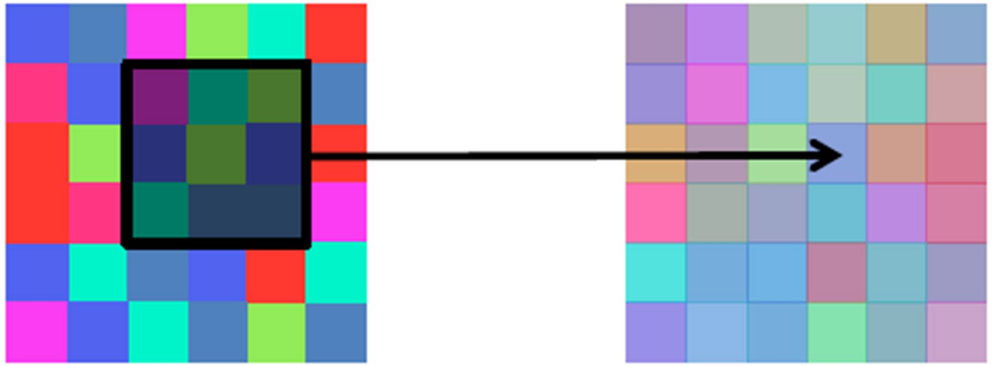
Point Processing vs Image Filtering

Point Operation



point processing

Neighborhood Operation

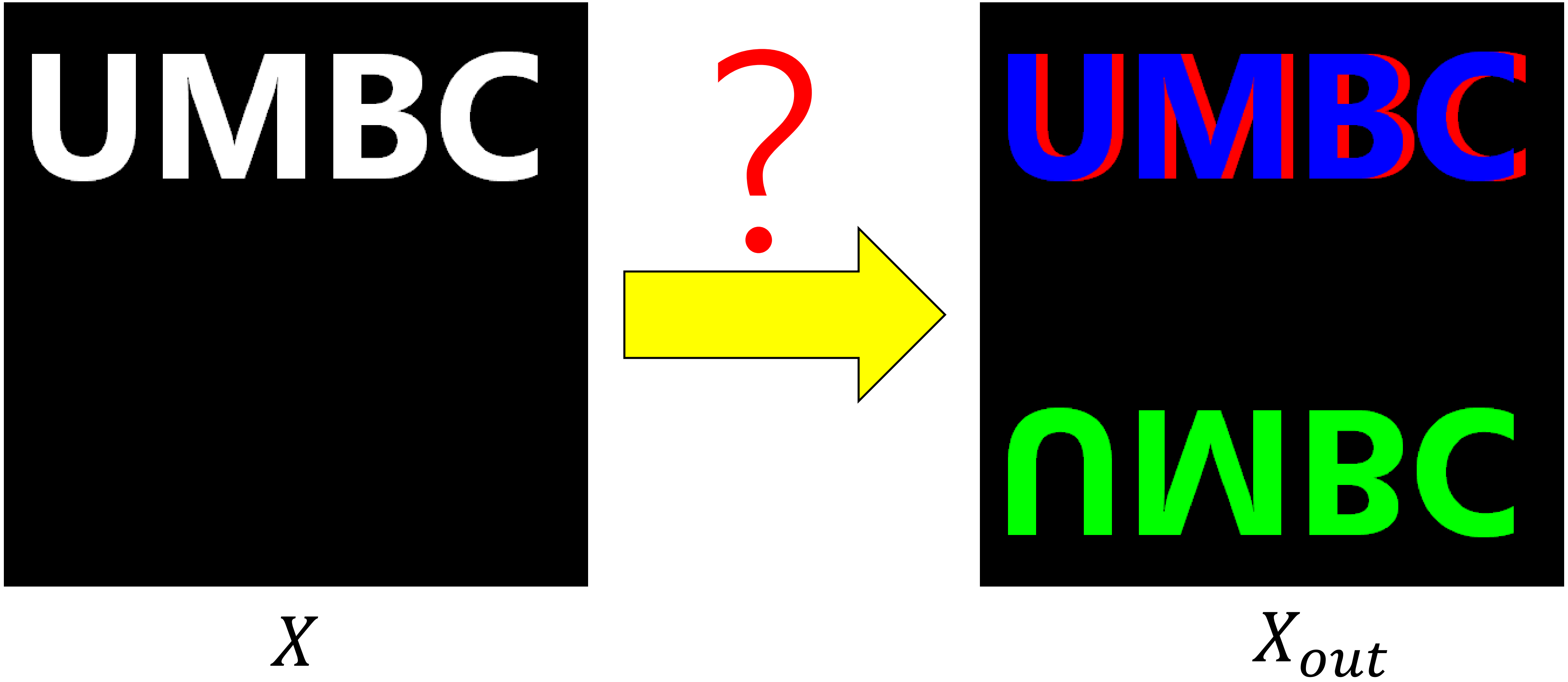


"filtering"

Recap

Quiz!

Write an Equation to generate X_{out}
using X , appropriate filters, point operators, etc.



```
import numpy as np
import imageio as io
import scipy.ndimage

x = io.imread("x.png", as_gray=False, pilmode="RGB")/255.

x_red = np.matmul(x, [[1, 0, 0], [0, 0, 0], [0, 0, 0]])
x_green = np.matmul(x, [[0, 0, 0], [0, 1, 0], [0, 0, 0]])
x_blue = np.matmul(x, [[0, 0, 0], [0, 0, 0], [0, 0, 1]])

x_green_flipped = np.flip(x_green, axis=0)

io.imwrite("x_red.png", x_red)
io.imwrite("x_blue.png", x_blue)
io.imwrite("x_green.png", x_green)
io.imwrite("x_green_flipped.png", x_green_flipped)

x_1 = x_blue + x_green_flipped
io.imwrite("x_1.png", x_1)

x_rightshift = scipy.ndimage.shift(x, [0, 10, 0])
io.imwrite("x_rightshift.png", x_rightshift)

x_underlay =  x_rightshift - x*x_rightshift
io.imwrite("x_underlay.png", x_underlay)

x_red_underlay = np.matmul(x_underlay, [[1, 0, 0], [0, 0, 0], [0, 0, 0]])
io.imwrite("x_red_underlay.png", x_red_underlay)

x_2 = x_1 + x_red_underlay
io.imwrite("x_2.png", x_2)
```

Example: box filter

$$g[\cdot\,,\cdot\,]$$

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

Slide credit: David Lowe (UBC)

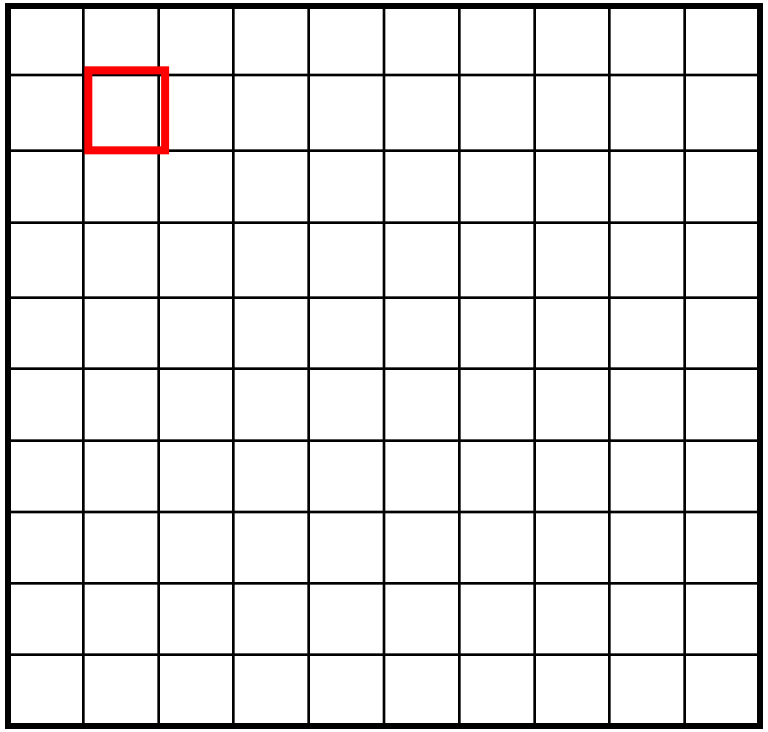
Image filtering

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

$f[.,.]$

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

$h[.,.]$



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

Image filtering

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

$f[.,.]$

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

$h[.,.]$

	0	10							

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10	20						

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

Image filtering

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

$f[.,.]$

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

$h[.,.]$

	0	10	20	30					

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

Image filtering

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

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				

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

Image filtering

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

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
				?					

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

Image filtering

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

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
						?			
				50					

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

Image filtering

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

$f[.,.]$

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

$h[.,.]$

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

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

Box Filter

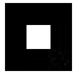
What does it do?

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

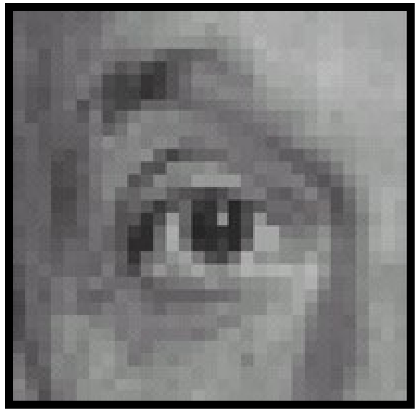
$$g[\cdot\,,\cdot\,] = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Slide credit: David Lowe (UBC)

Smoothing with box filter



Practice with linear filters



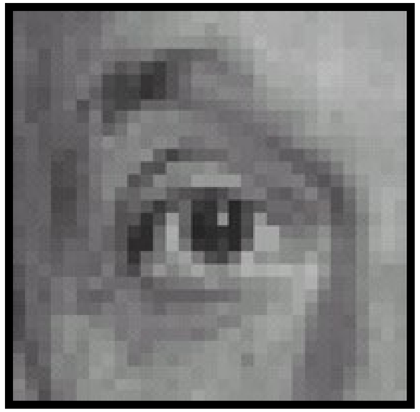
Original

0	0	0
0	1	0
0	0	0

?

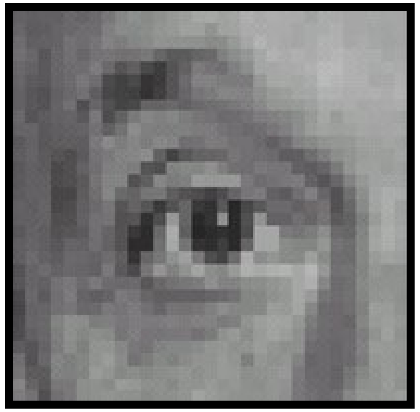
Source: D. Lowe

Practice with linear filters



Original

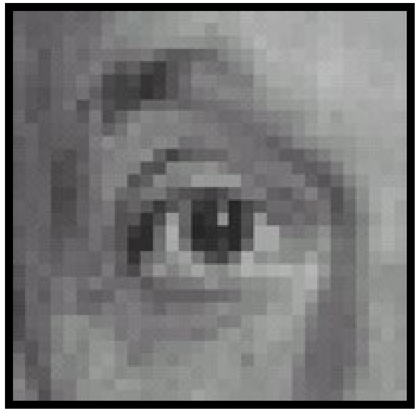
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

Practice with linear filters



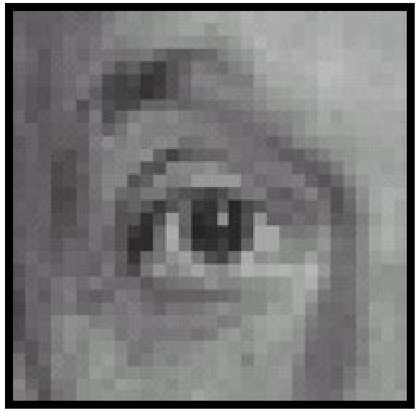
Original

0	0	0
0	0	1
0	0	0

?

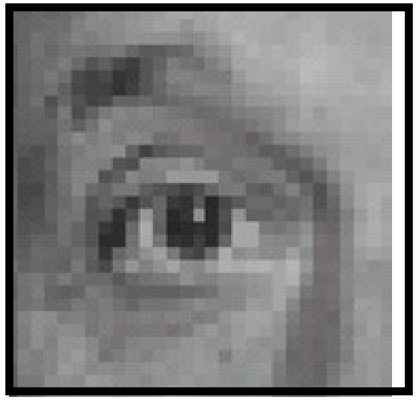
Source: D. Lowe

Practice with linear filters



Original

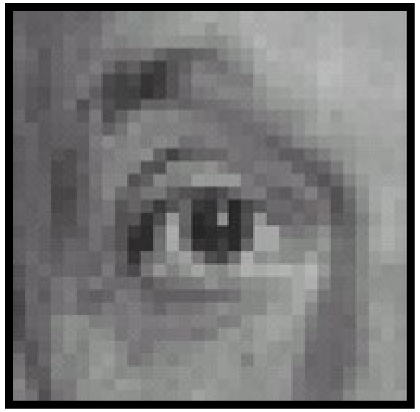
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters



Original

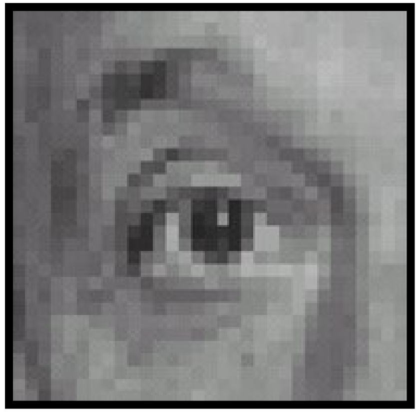
$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

?

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters



Original

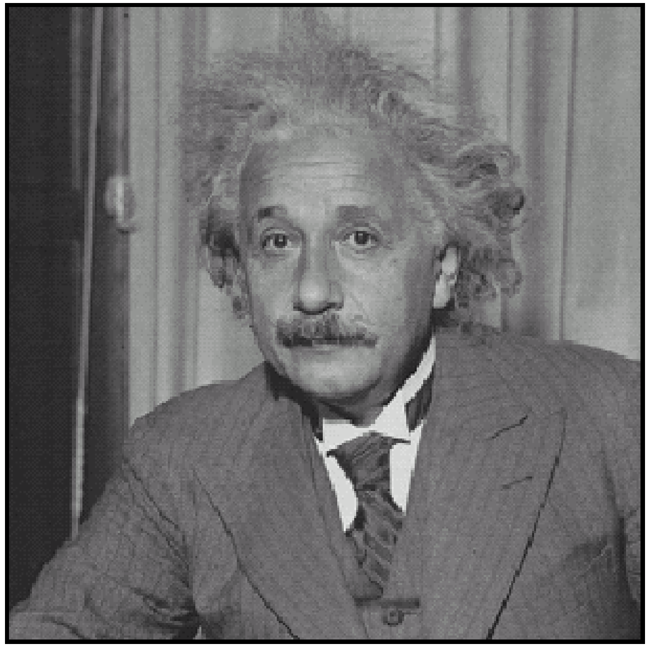
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

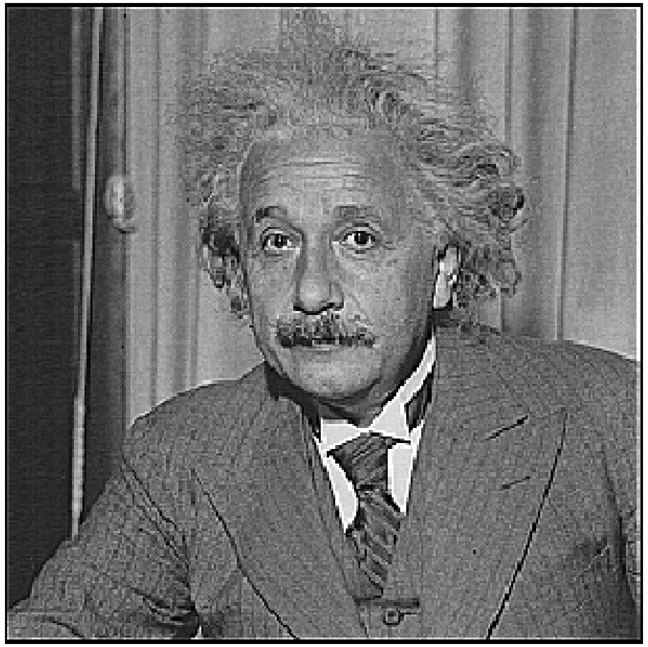
- Accentuates differences with local average

Source: D. Lowe

Sharpening



before

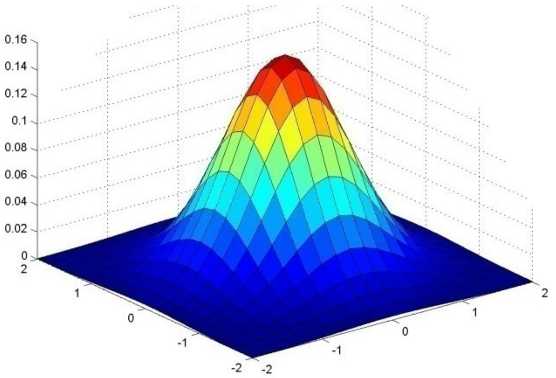


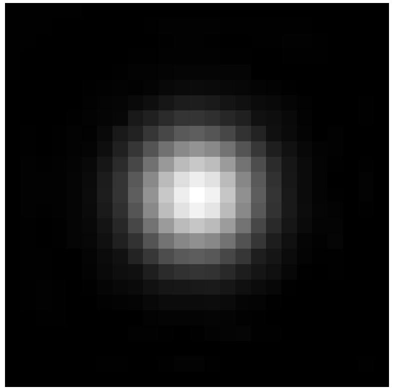
after

Source: D. Lowe

Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness





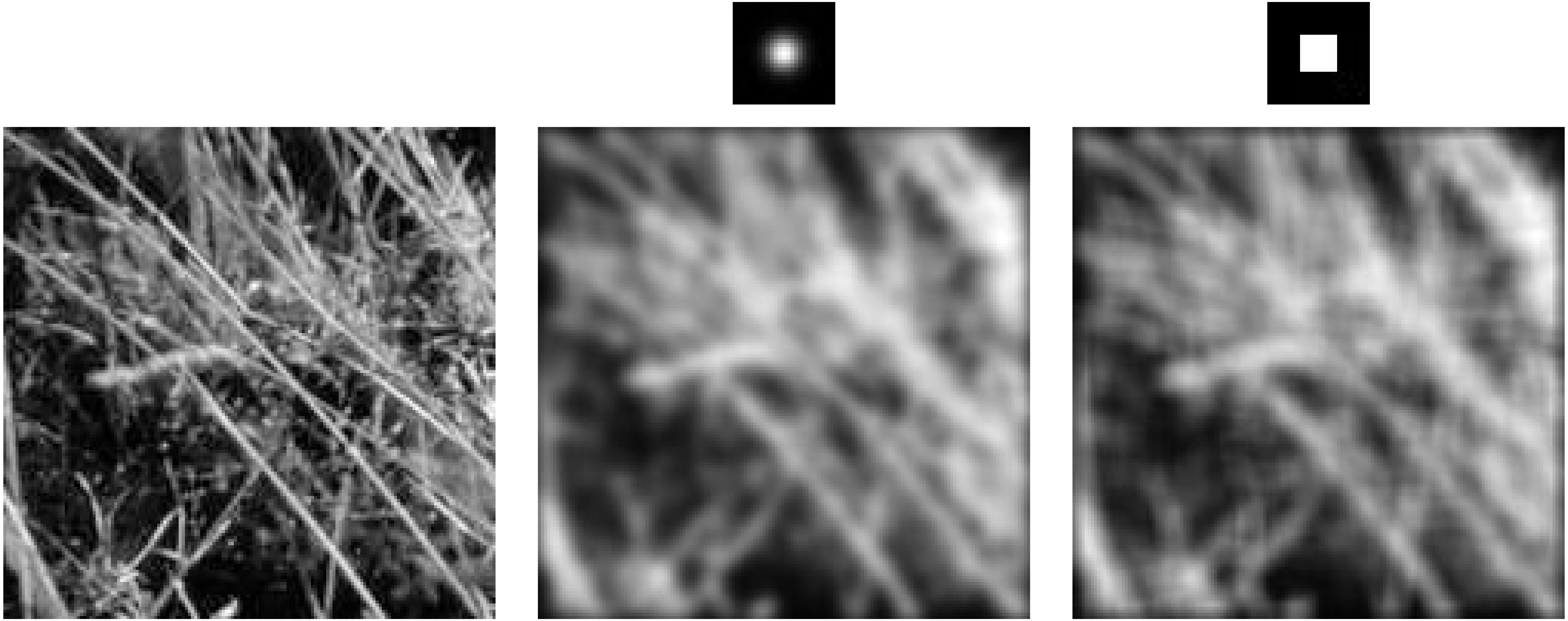
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

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

Slide credit: Christopher Rasmussen

Smoothing with Gaussian vs Box filter



Edge Detection in Images

Edges are useful image features

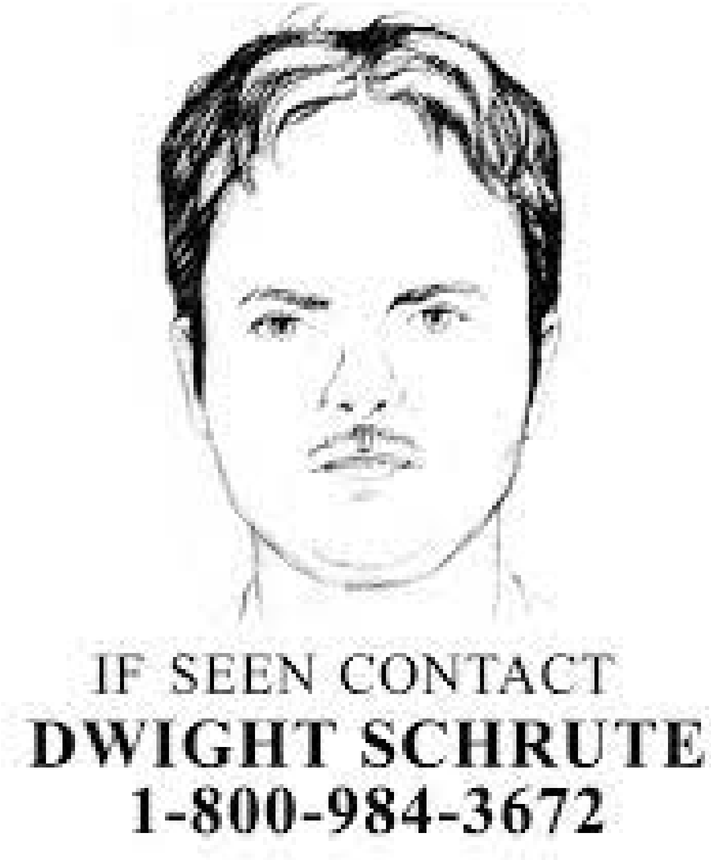
People saying I can't make memes using edge detection

Me:

What are Image Edges?

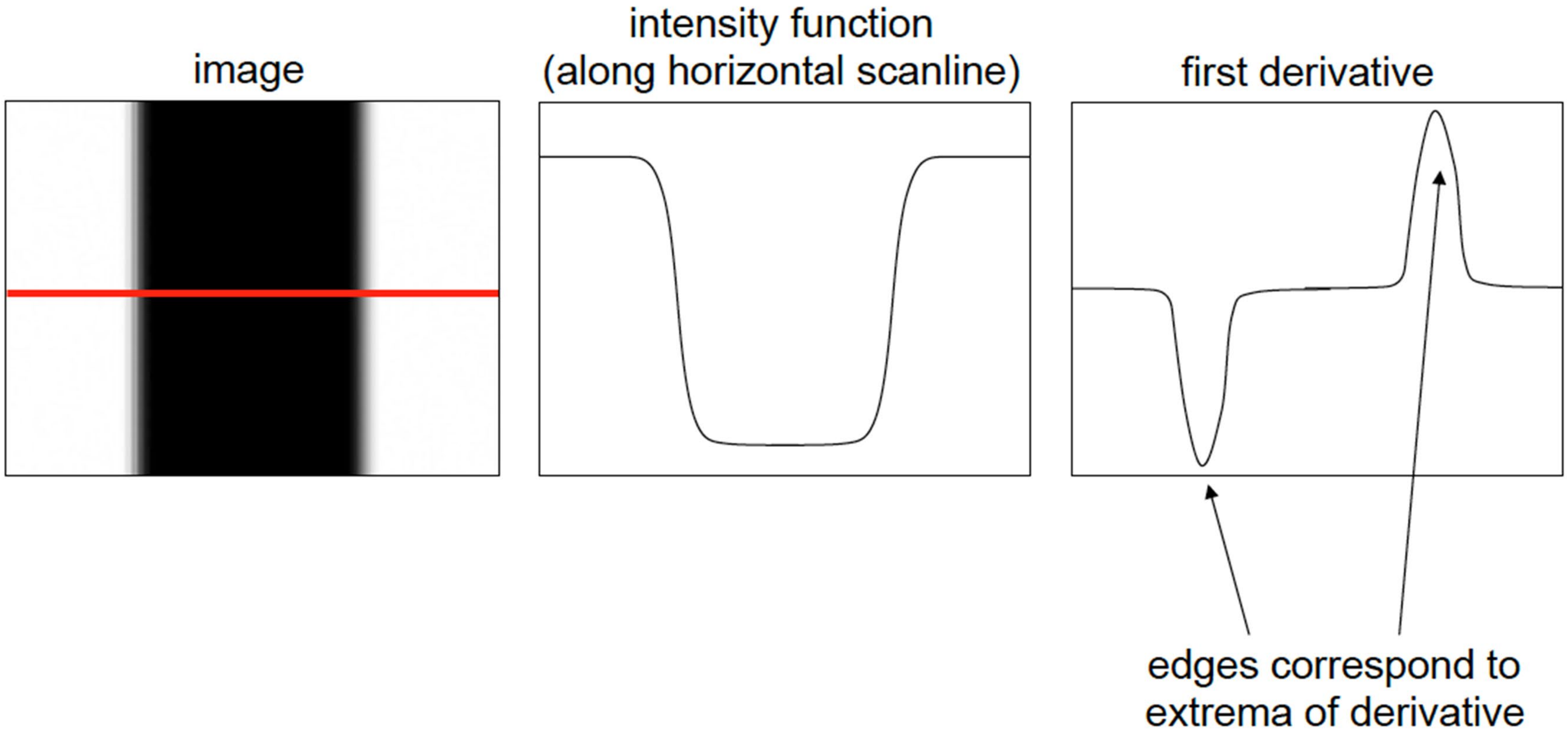
- Sudden changes in image intensity
  - Most shape information is in edges
  - Shape leads to semantic information (cats vs humans vs chairs vs apples)
  - Edges are more compact than pixels (number of bits …)

- E.g. Sketches / line drawings
  - Artists use it all the time



What are Image Edges?

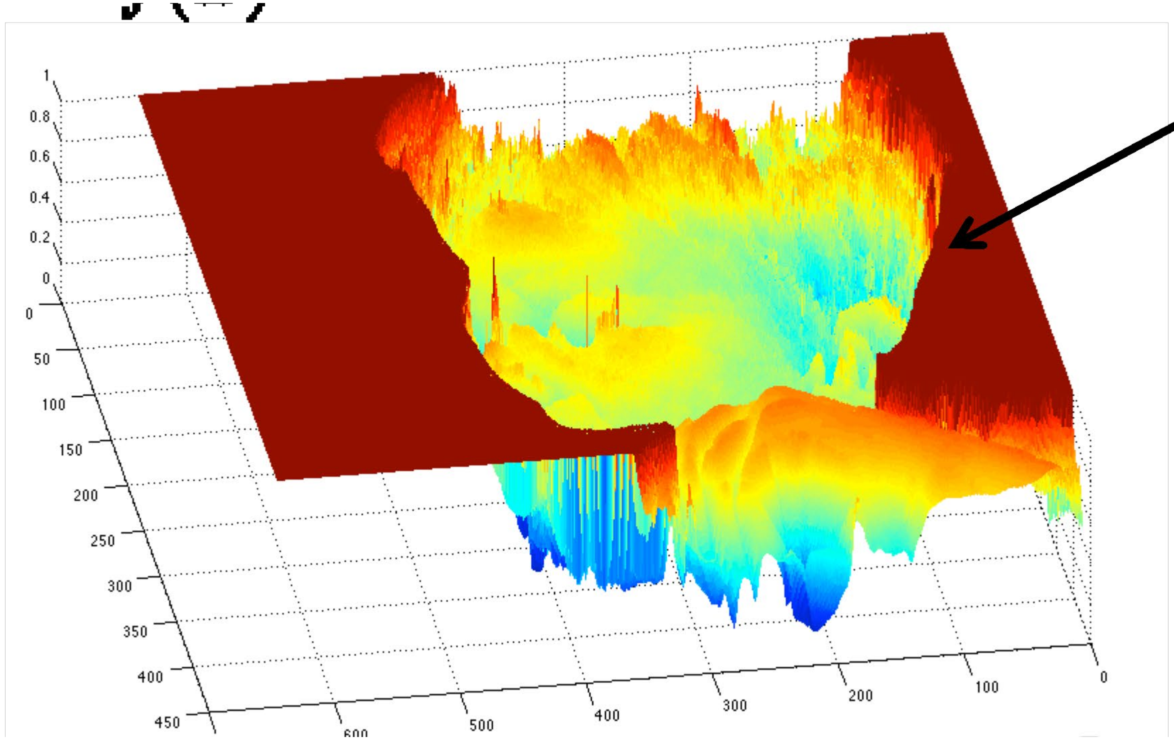
Edges = locations of rapid change in signal (image) intensity



Gradients!

- To locate rapid changes, compute gradients (first derivative)

grayscale image



Very sharp discontinuities in intensity.

Gradients in Practice: Finite Difference

- Forward difference

$$\frac{\partial f(x)}{\partial x} = \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

- Backward difference

$$\frac{\partial f(x)}{\partial x} = \frac{f(x) - f(x-\Delta x)}{\Delta x}$$

- Central difference

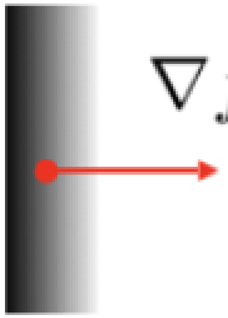
$$\frac{\partial f(x)}{\partial x} = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}$$

Image gradient

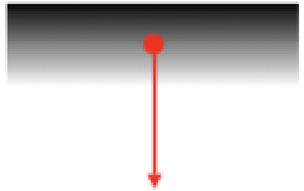
The gradient of an image:

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

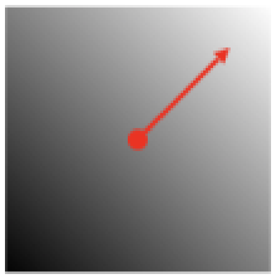
The gradient points in the direction of most rapid change in intensity



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



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



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

Edge Detection using Finite Difference

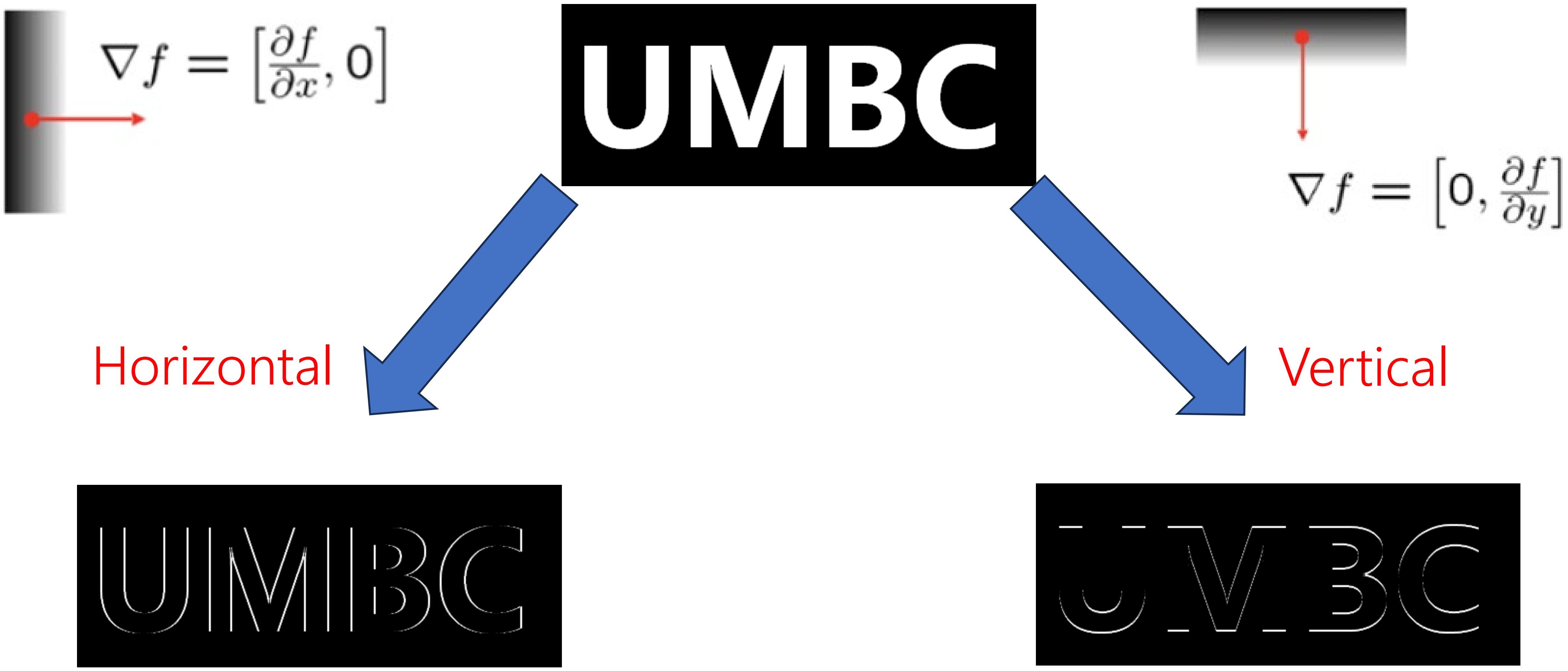


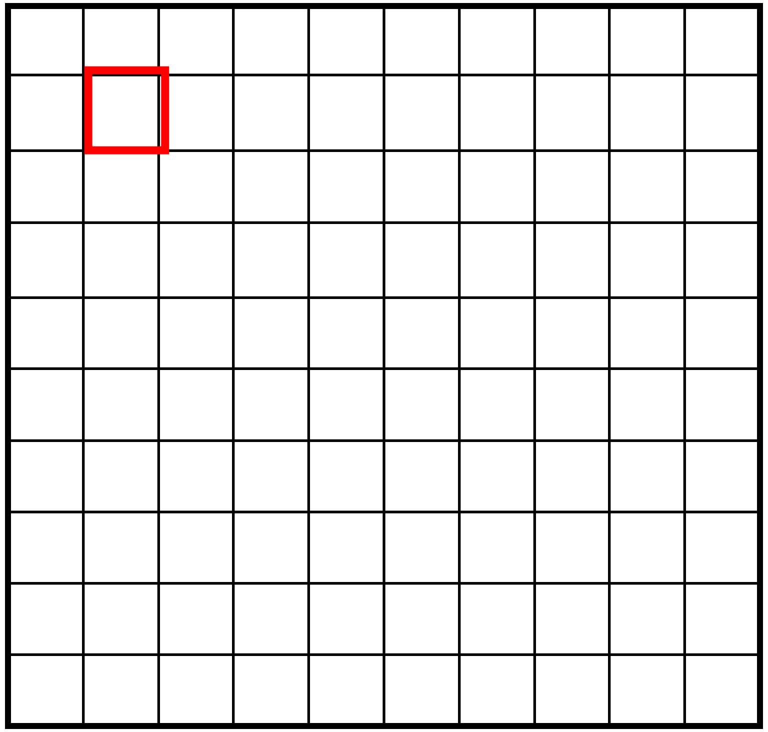
Image filtering

$g[\cdot,\cdot]$

$f[.,.]$

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

$h[.,.]$



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

Repeat with a different filter

$g[\cdot\,,\cdot\,]$

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

Image filtering

$g[\cdot,\cdot]$

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

$f[.,.]$

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

$h[.,.]$

	0								

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

Image filtering

$g[\cdot,\cdot]$

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

$f[.,.]$

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

$h[.,.]$

	0	-180							

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

Image filtering

g[· ,·]

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

$f[.,.]$

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

$h[.,.]$

	0	-180	-180						

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

Image filtering

$g[\cdot,\cdot]$

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

$f[.,.]$

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

$h[.,.]$

	0	-180	-180	0					

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

Image filtering

$g[\cdot,\cdot]$

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

$f[.,.]$

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

$h[.,.]$

	0	-180	-180	0	0				

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

Image filtering

$g[\cdot,\cdot]$

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

$f[.,.]$

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

$h[.,.]$

	0	-180	-180	0	0	0			

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

Image filtering

$g[\cdot,\cdot]$

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

$f[.,.]$

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

$h[.,.]$

	0	-180	-180	0	0	0	180	180	

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

Image filtering

$g[\cdot,\cdot]$

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

$f[.,.]$

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

$h[.,.]$

	0	-180	-180	0	0	0	180	180	
	0	-270	-270	0	0	0	270	270	
	0	-270	-270	0	0	0	270	270	
	0	-270	-270	0	0	0	270	270	
	0	-180	-180	0	0	0	180	180	

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

Edge Detection via Convolution

Sobel Filter

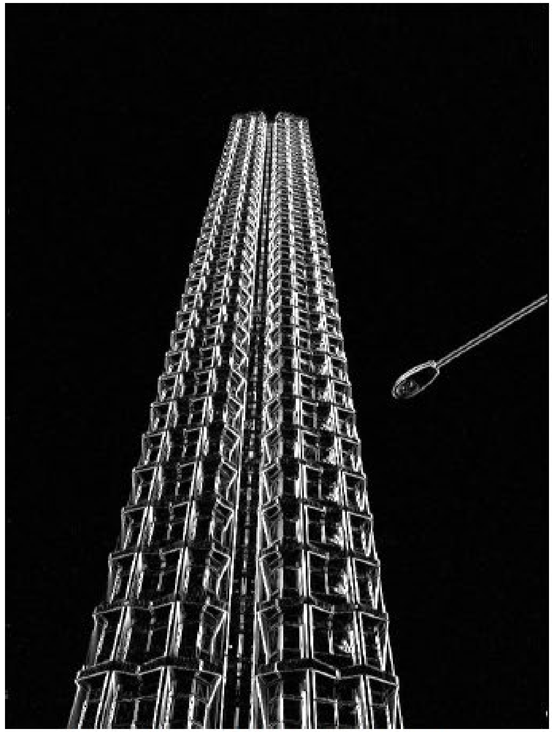
Horizontal Sober filter:

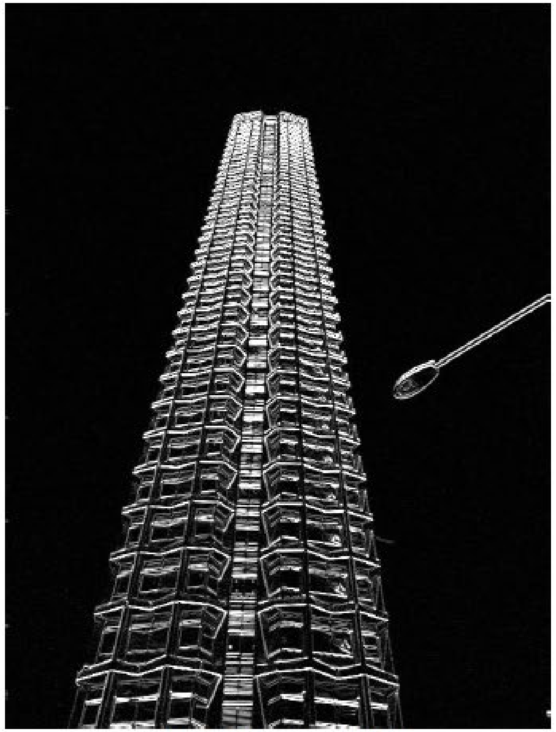
1	0	-1
2	0	-2
1	0	-1

Vertical Sobel filter:

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


original


horizontal Sobel filter


vertical Sobel filter

Edge Detection via Convolution

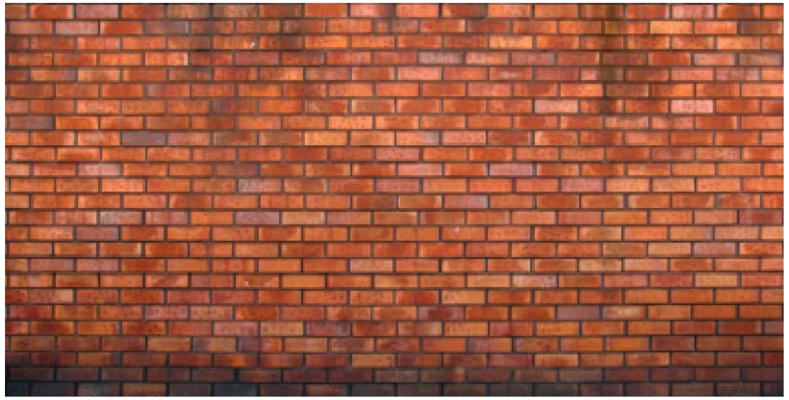
Sobel Filter

Horizontal Sober filter:

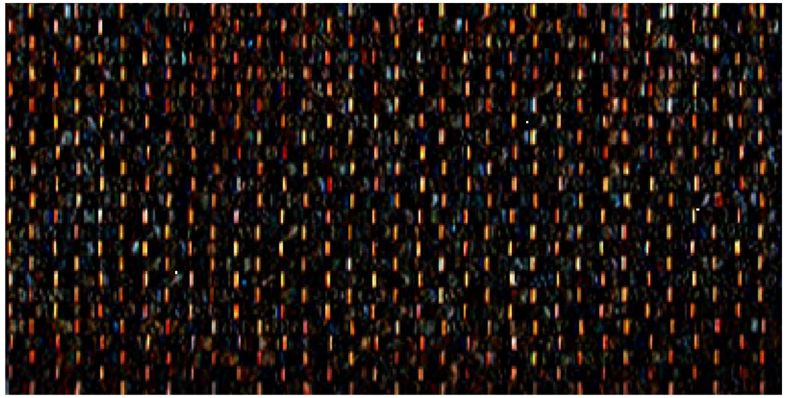
1	0	-1
2	0	-2
1	0	-1

Vertical Sobel filter:

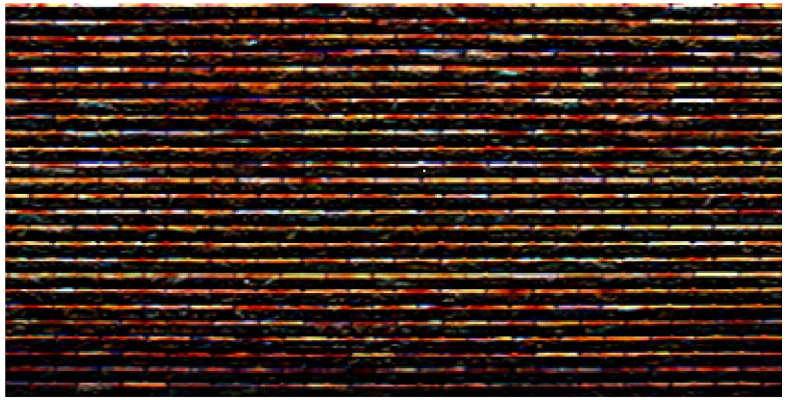
1	2	1
0	0	0
-1	-2	-1



original



horizontal Sobel filter



vertical Sobel filter

Many Types of Edge Filters

Sobel

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

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

Scharr

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

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

Prewitt

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

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

Roberts

0	1
-1	0

1	0
0	-1

Computing image gradients

1. Select your favorite derivative filters.

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

2. Convolve with the image to compute derivatives.

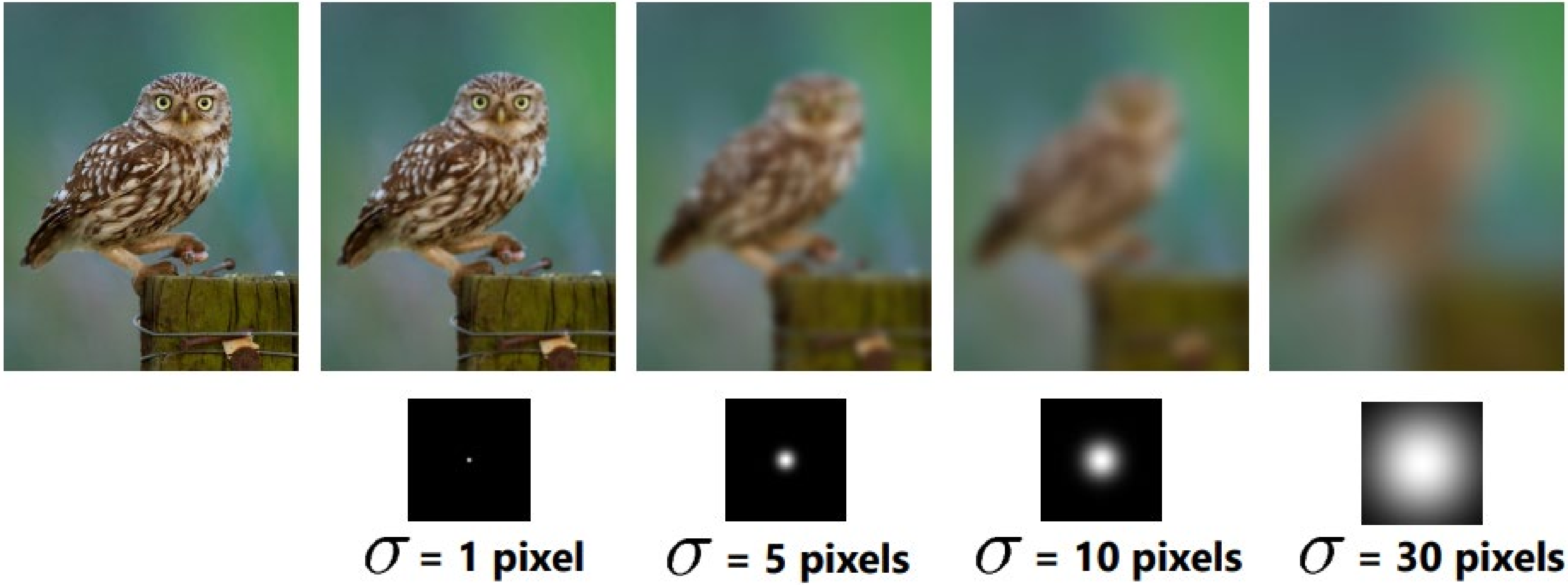
$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

3. Form the image gradient, and compute its direction and amplitude.

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right) \qquad ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

gradient — direction — amplitude

Gaussian Filter Revisited (Smoothing / Blurring)



Sharpening Filter

- What does blurring take away?

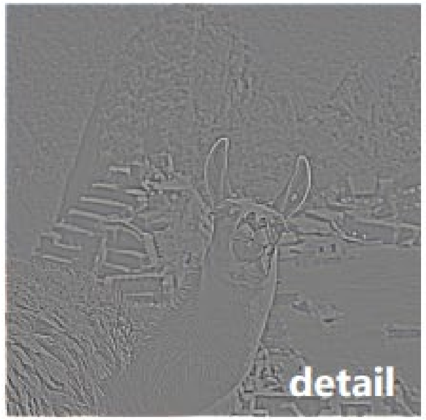
−

=

(This "detail extraction" operation is also called a ***high-pass filter***)

Let's add it back:

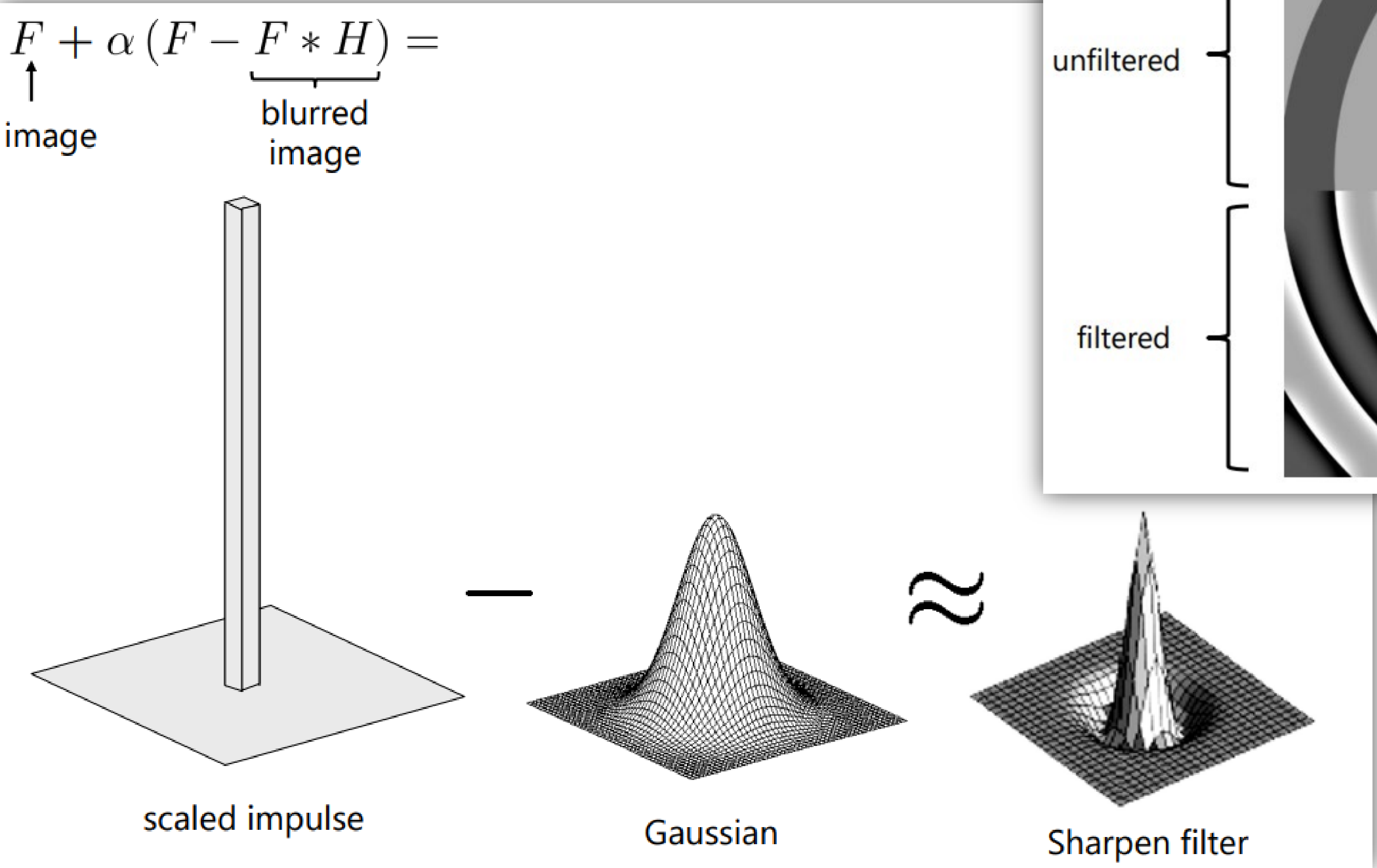
+ α



=

Photo credit: https://www.flickr.com/photos/geezaweezer/16089096376/

Sharpening Filter



Thresholding

$$g(m,n) = \begin{cases} 255, & f(m,n) > A \\ 0 & otherwise \end{cases}$$

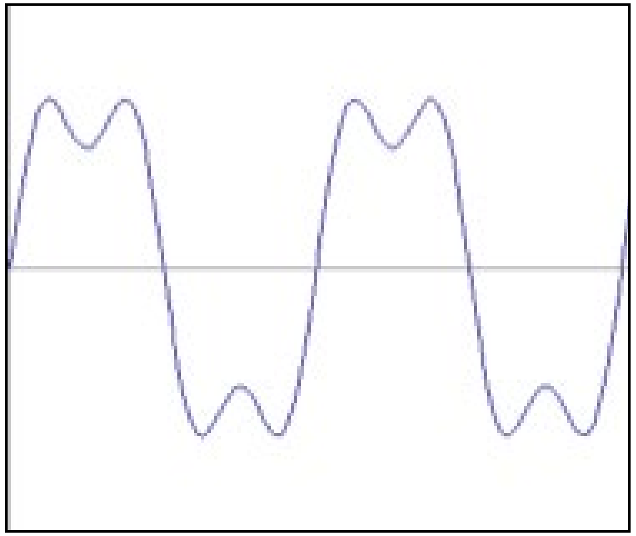
Question:

Does thresholding make storage efficient?

By how much (in this example)?

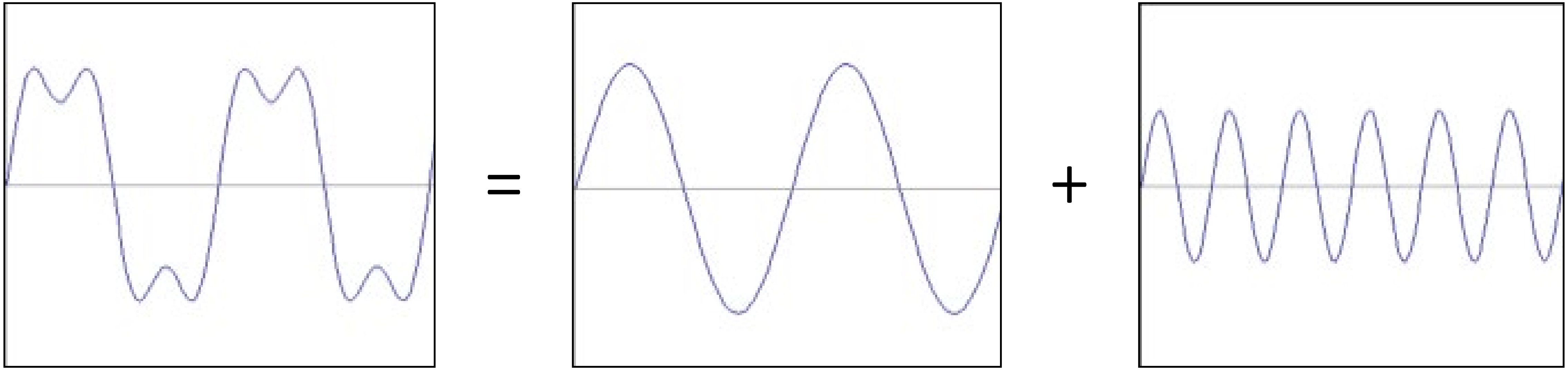
Fourier Domain Filtering

How would you generate this function?



= ? + ?

How would you generate this function?



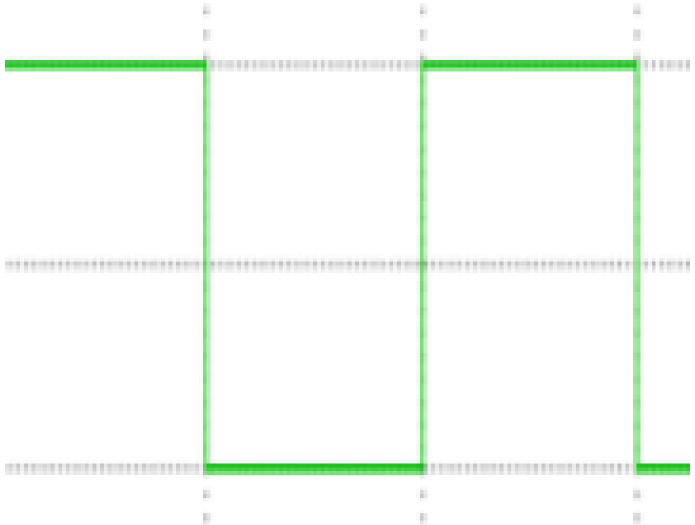
$$f(x) = \sin(2\pi x) + \frac{1}{3}\sin(2\pi 3x)$$

$$\sin(2\pi x)$$

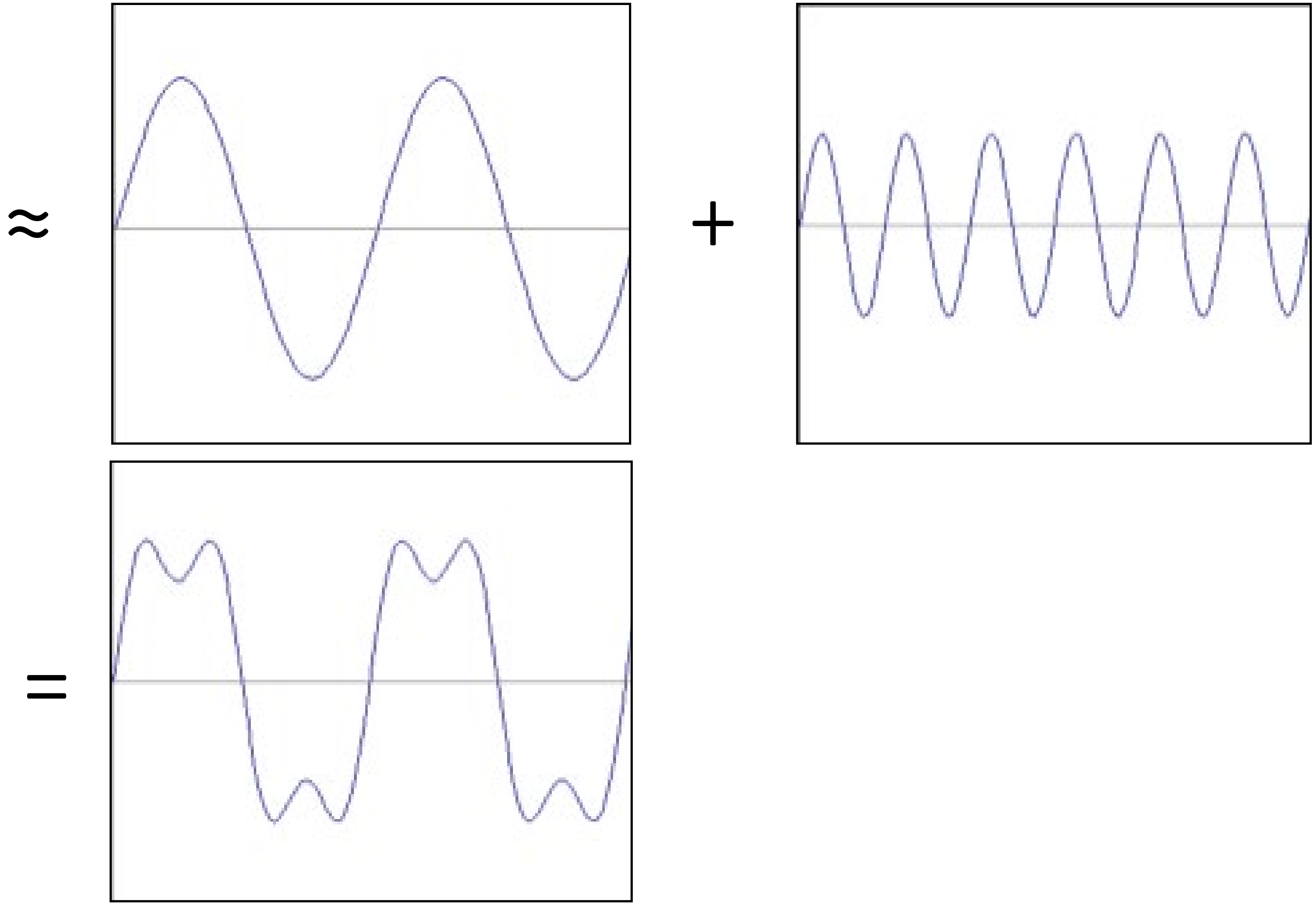
$$\frac{1}{3}\sin(2\pi 3x)$$

Examples

How would you generate this function?

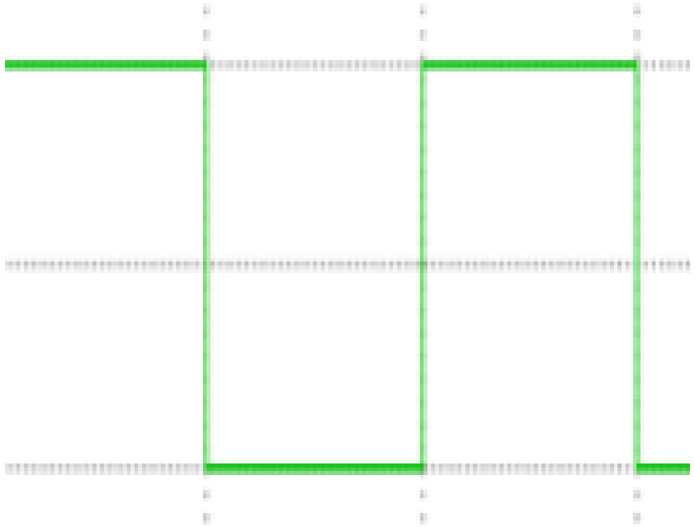


square wave



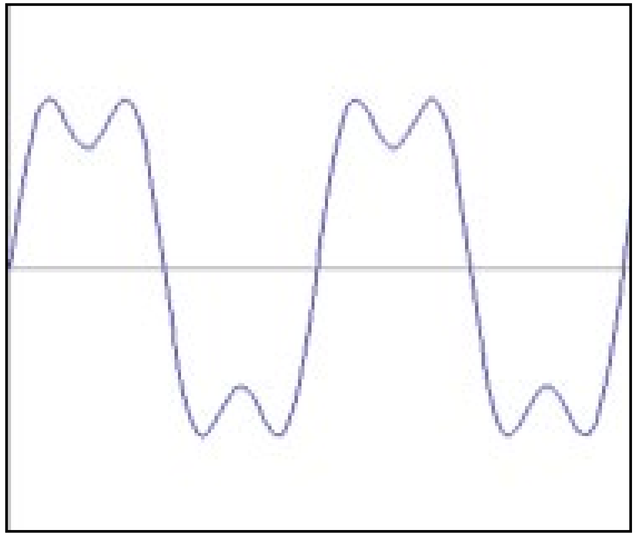
Examples

How would you generate this function?

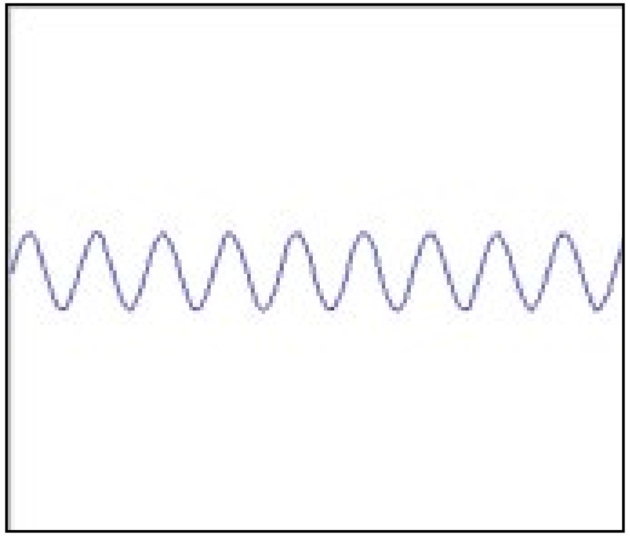


square wave

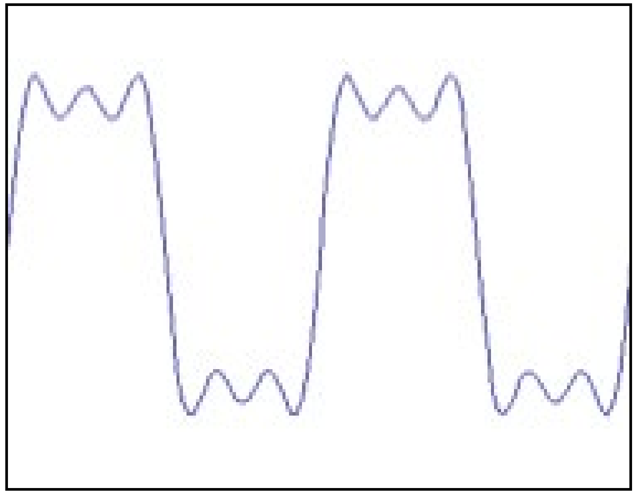
≈



+

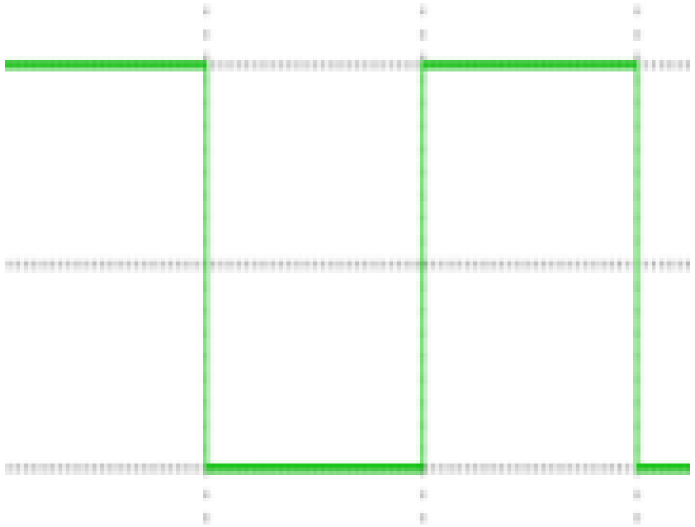


=



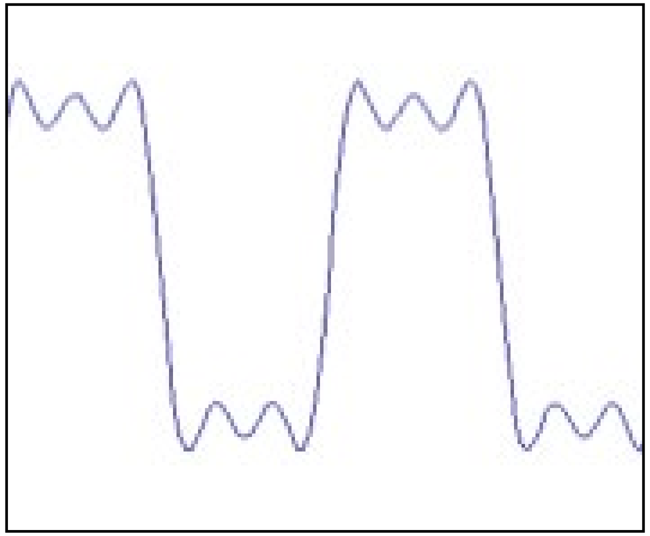
Examples

How would you generate this function?

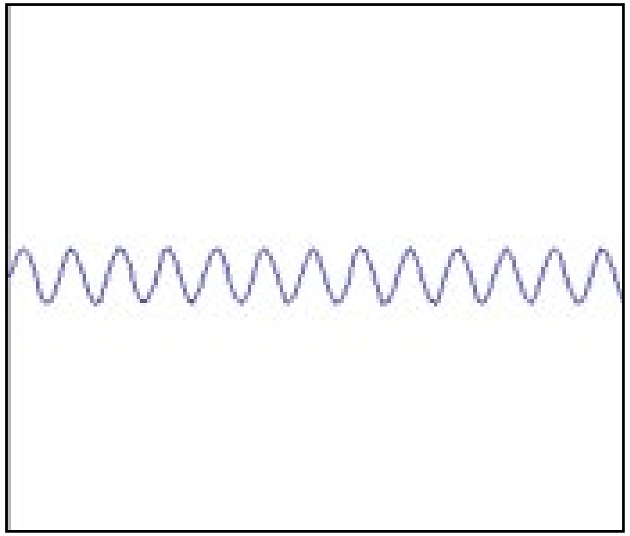


square wave

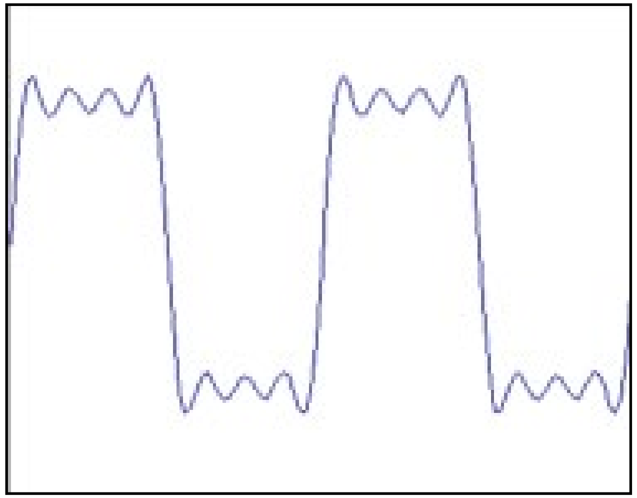
≈



+

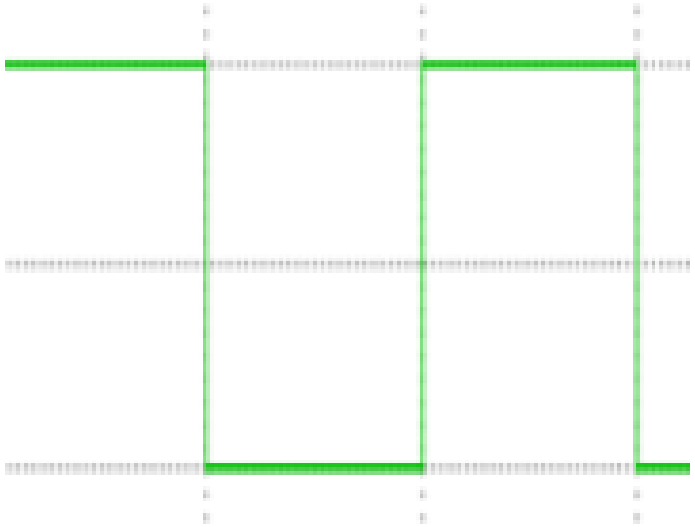


=



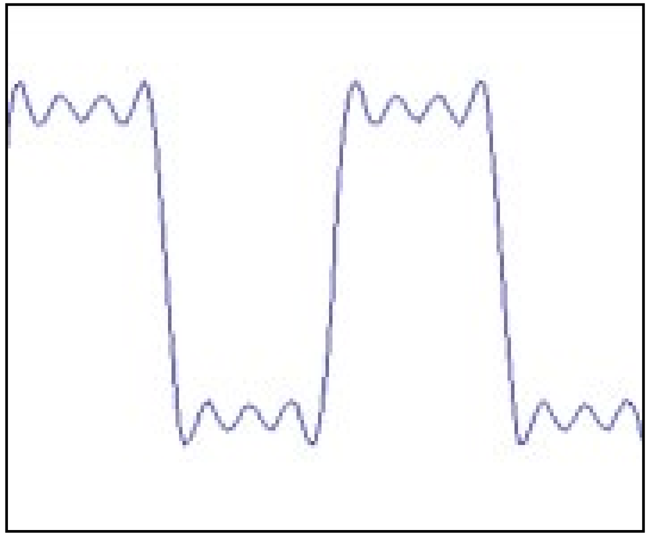
Examples

How would you generate this function?

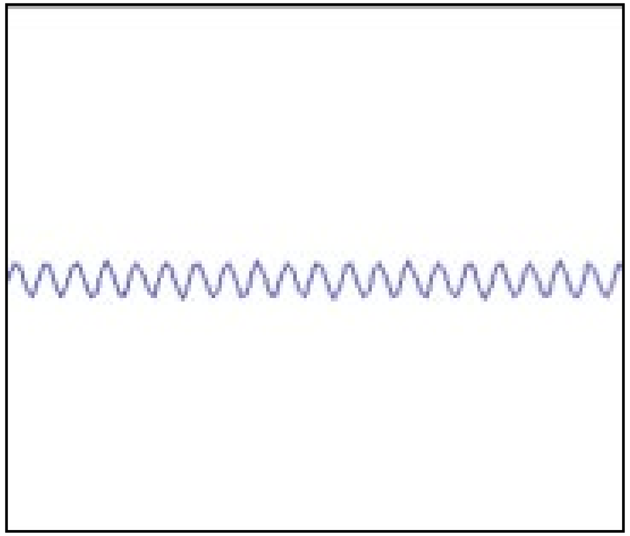


square wave

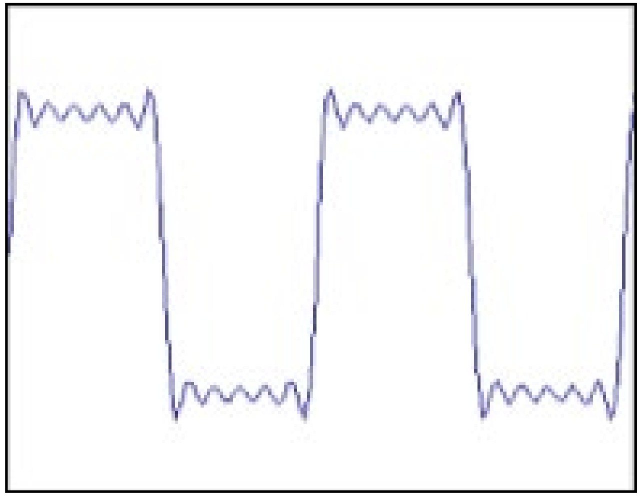
≈



+

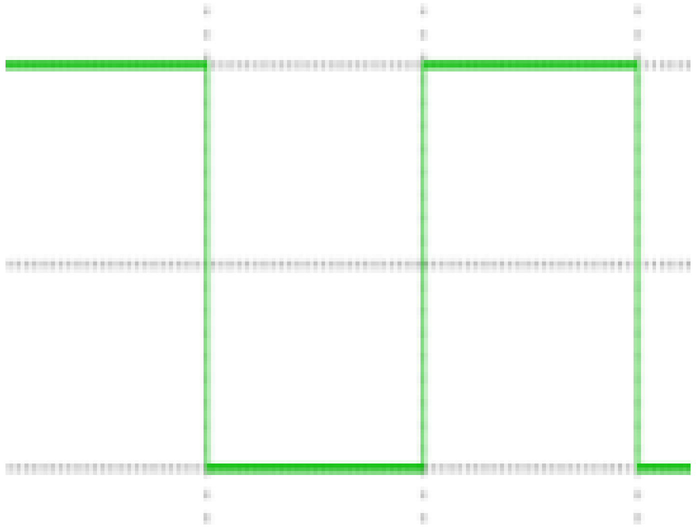


=



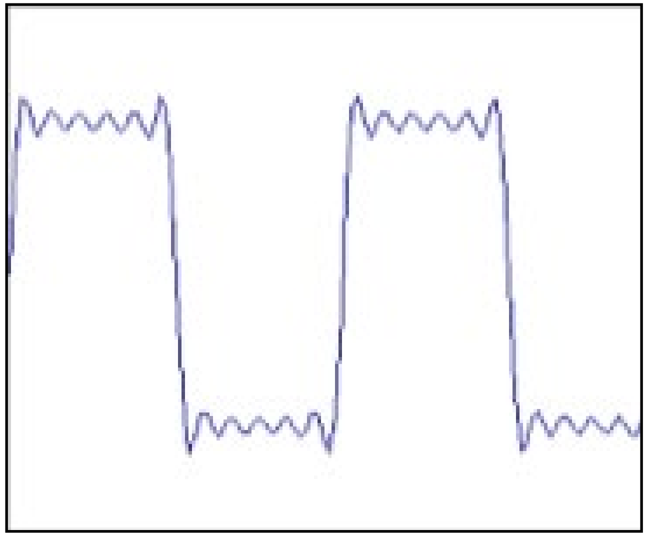
Examples

How would you generate this function?

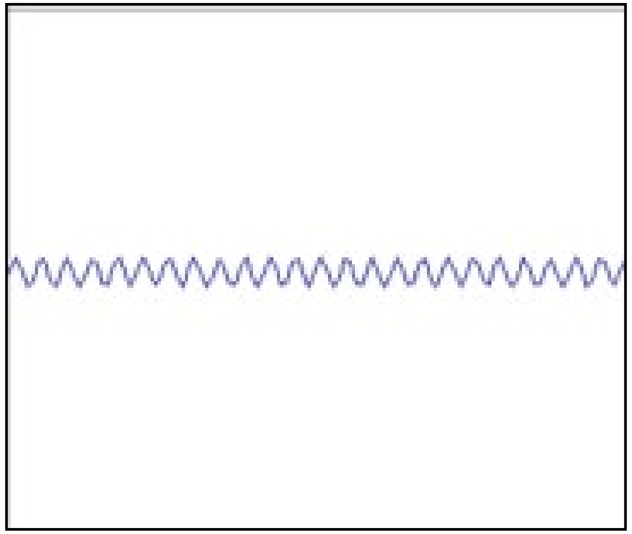


square wave

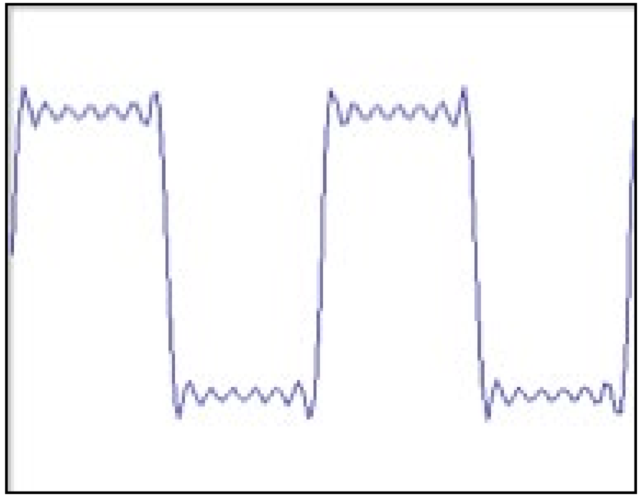
≈



+

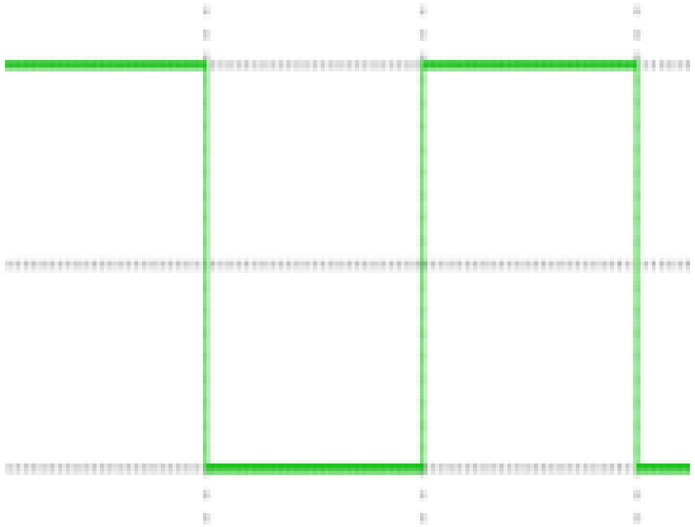


=



How would you express this mathematically?

Examples



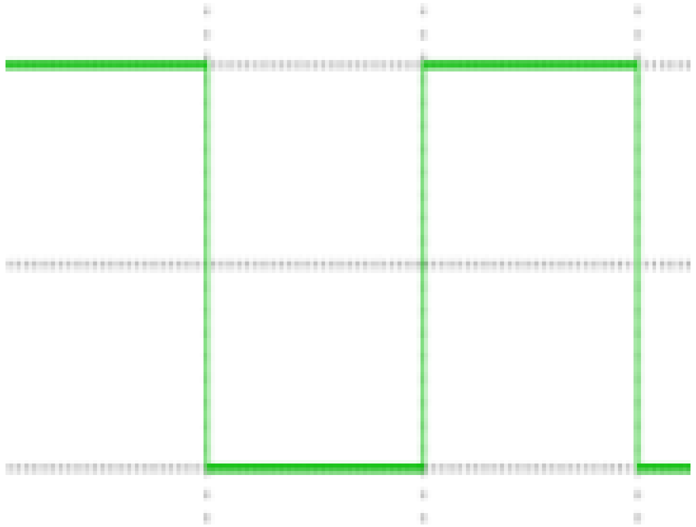
$$= \quad A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kx)$$

square wave

infinite sum of sine waves

How would could you visualize this in the frequency domain?

Examples

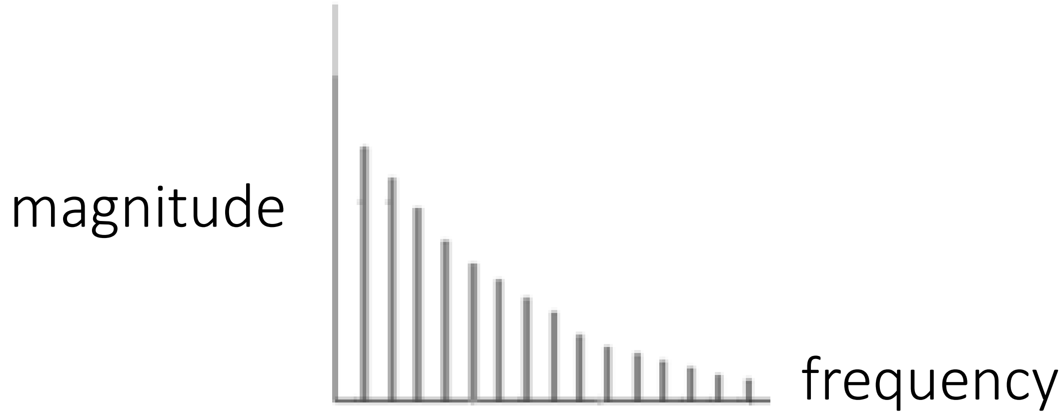


square wave

$=$

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

infinite sum of sine waves

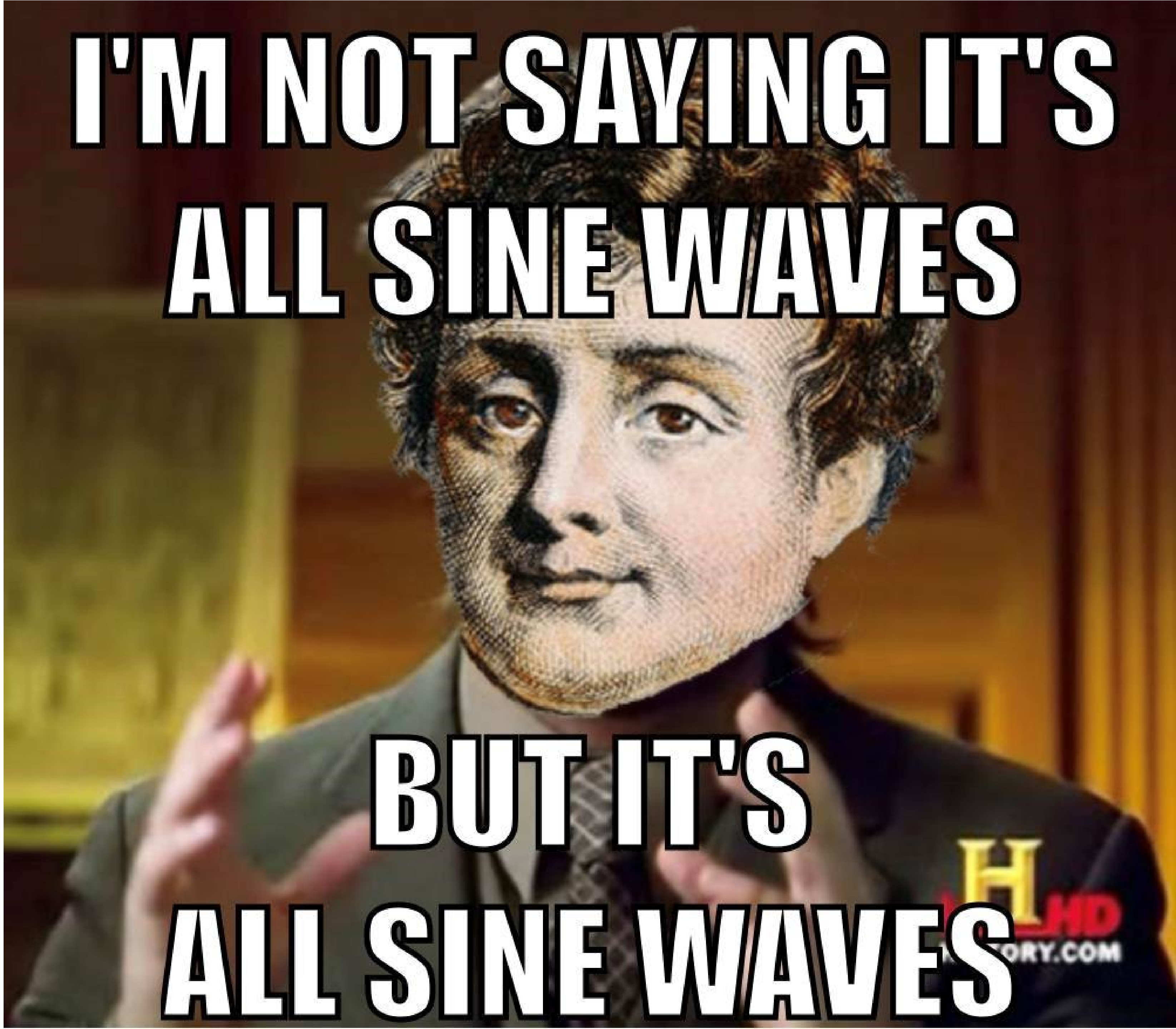


Fourier Series

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

- amplitude: A
- sinusoid: $\sin$
- angular frequency: ω
- variable: x
- phase: ϕ

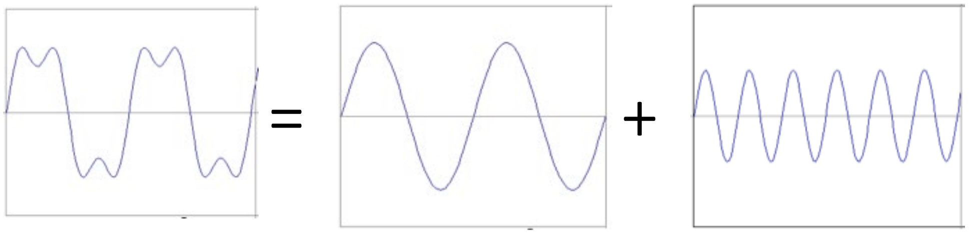
Fourier's claim:
Add enough of these to get <u>any</u> *periodic* signal you want!

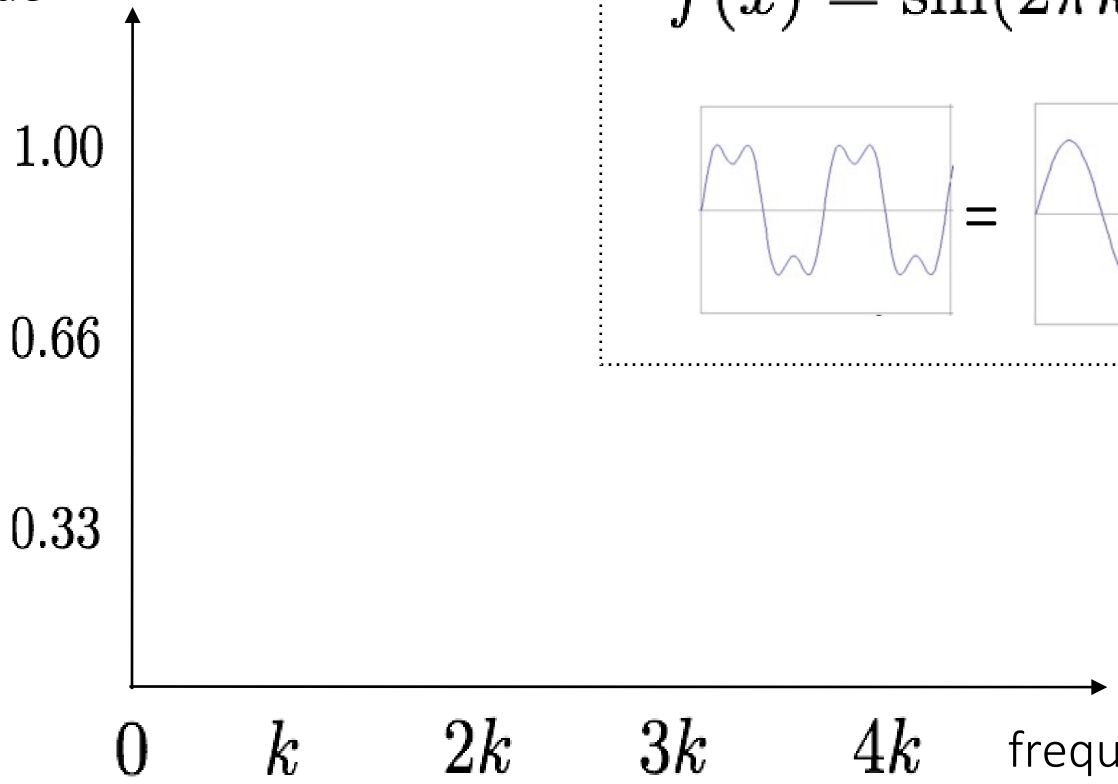


Visualizing the frequency spectrum

Recall the temporal domain visualization

$$f(x) = \sin(2\pi kx) + \frac{1}{3}\sin(2\pi 3kx)$$

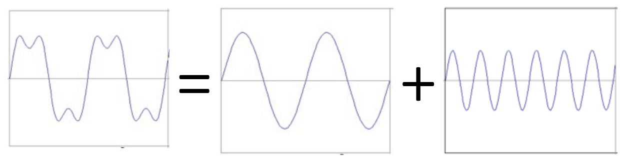


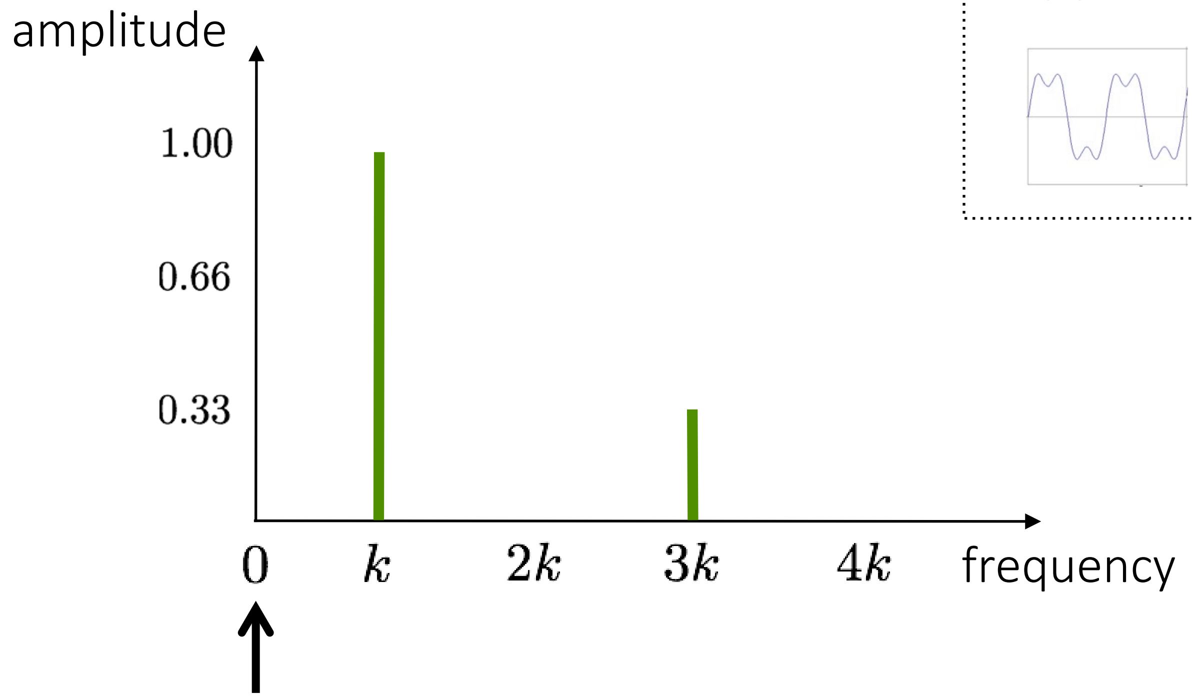


Visualizing the frequency spectrum

Recall the temporal domain visualization

$$f(x) = \sin(2\pi kx) + \frac{1}{3}\sin(2\pi 3kx)$$





not visualizing the symmetric negative part

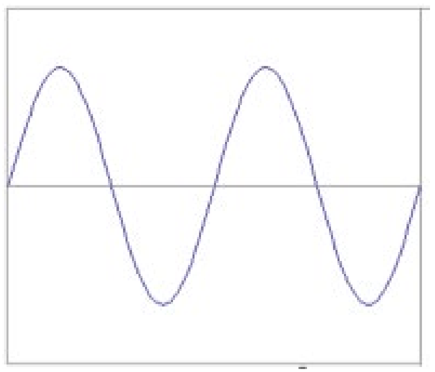
signal average (zero for a sine wave with no offset)

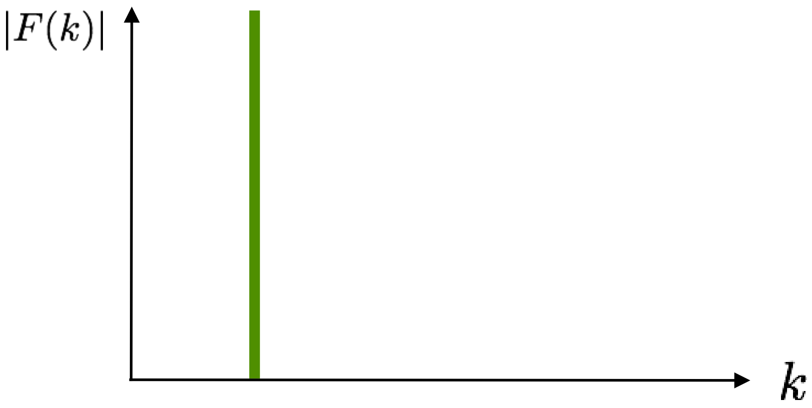
Need to understand this to understand the 2D version!

Examples

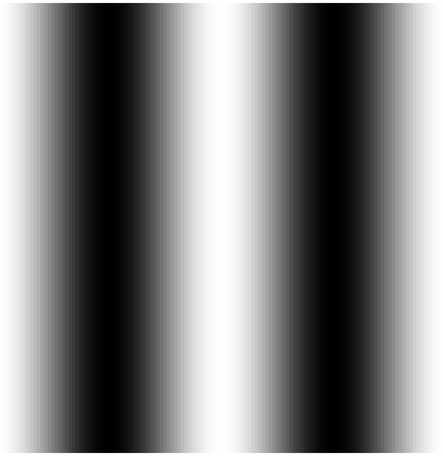
Spatial domain visualization | Frequency domain visualization

1D





2D



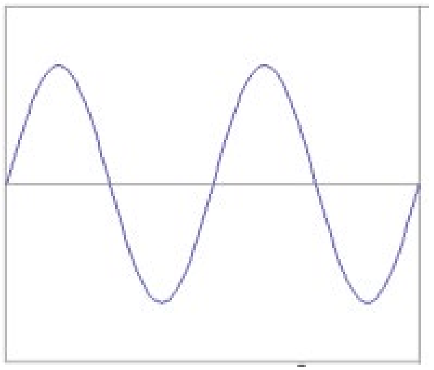
?

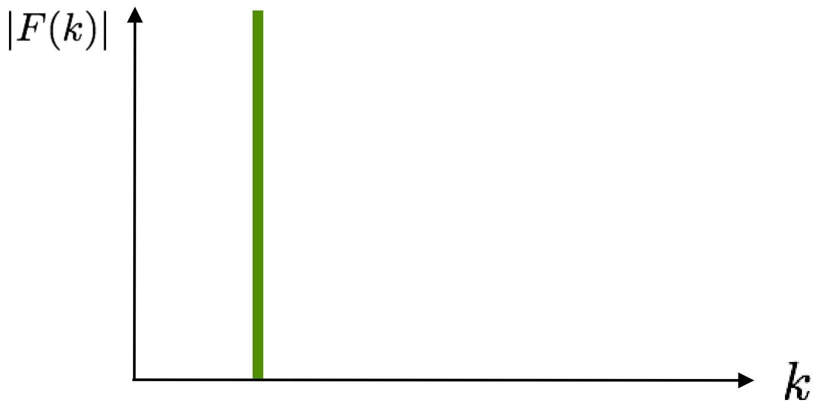
Examples

Spatial domain visualization

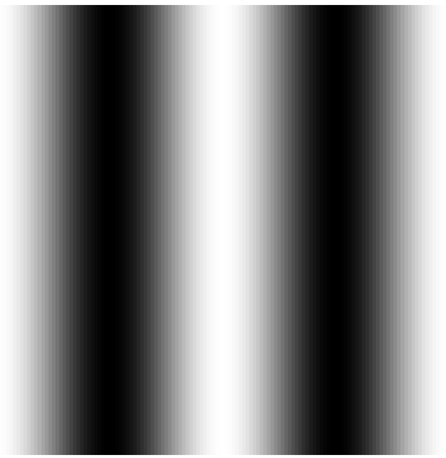
Frequency domain visualization

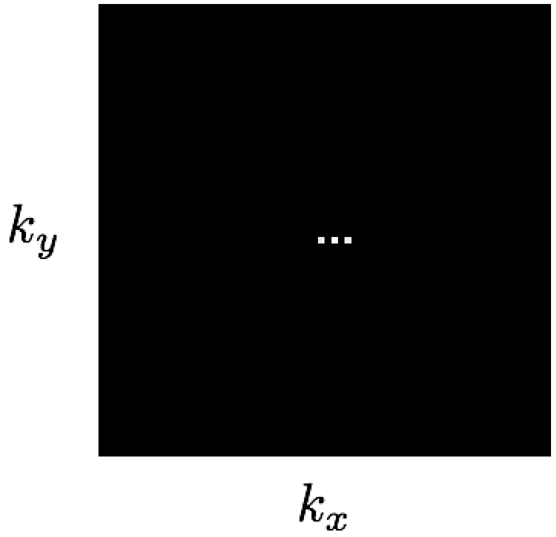
1D





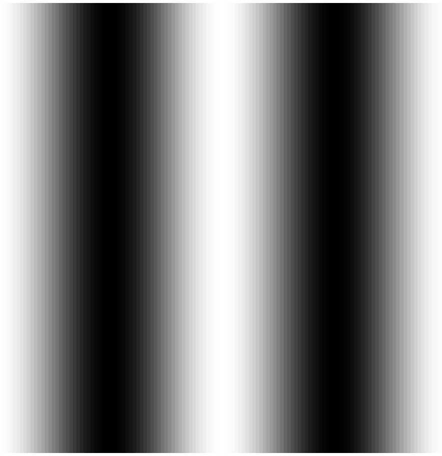
2D



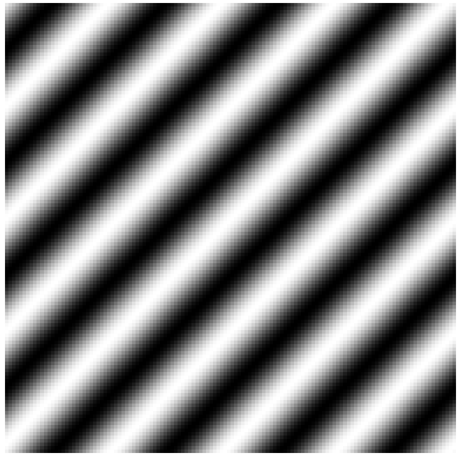


What do the three dots correspond to?

Examples

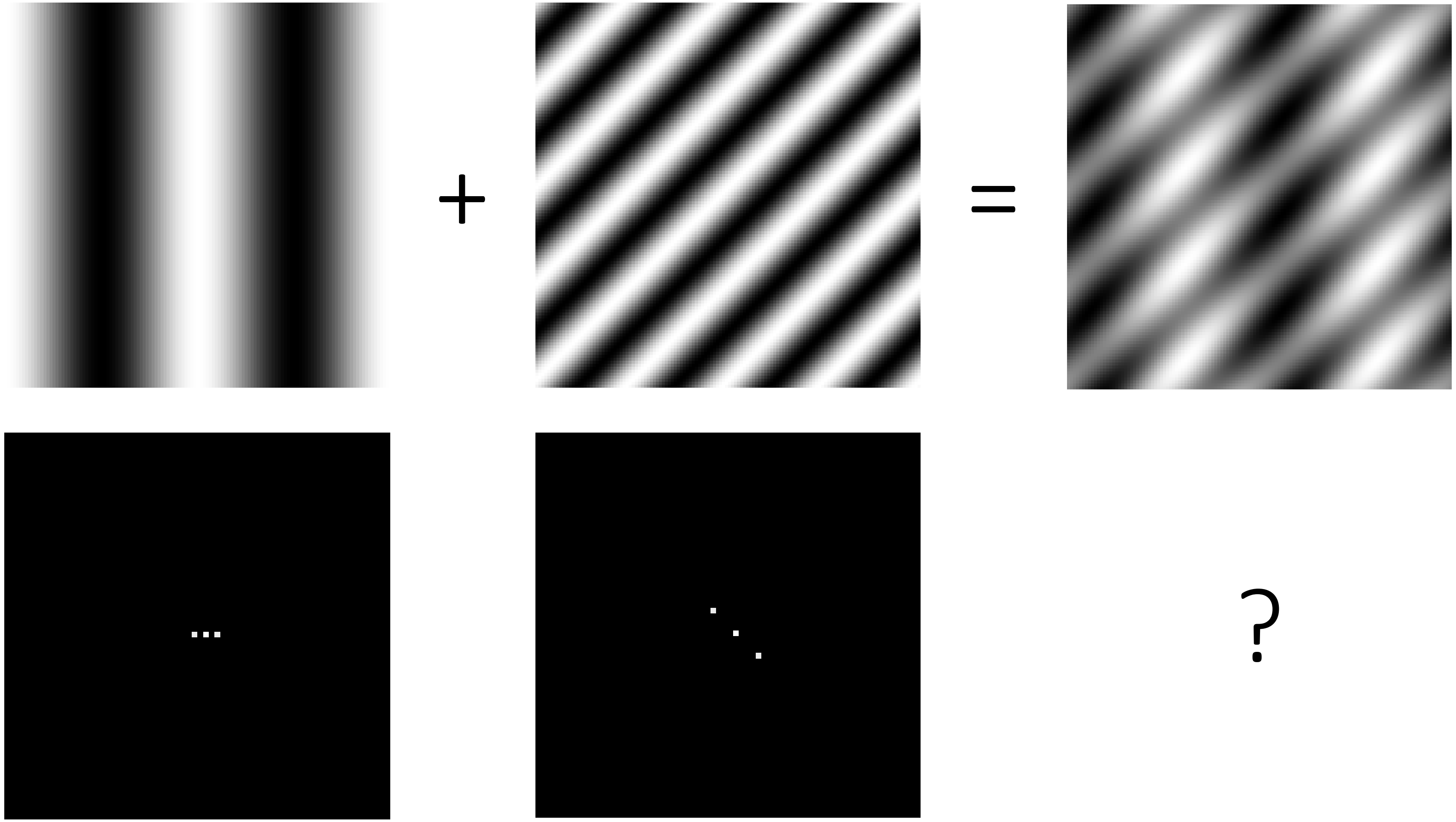


+

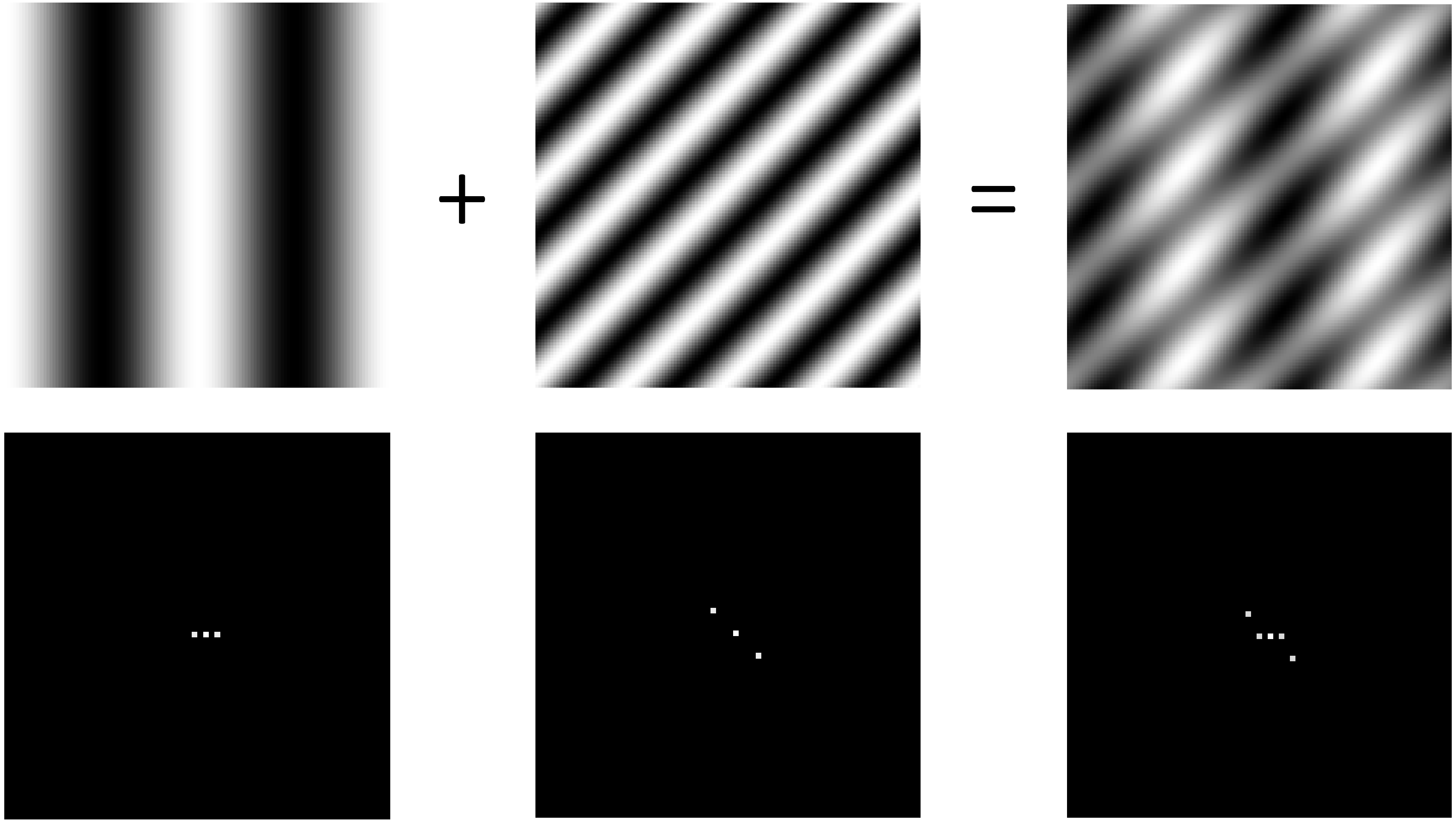


= ?

Examples



Examples



Fourier transform

	Fourier transform	inverse Fourier transform
continuous	$F(k) = \int_{\infty}^{-\infty} f(x) e^{-j2\pi kx} dx$	$f(x) = \int_{\infty}^{-\infty} F(k) e^{j2\pi kx} dk$
discrete	$F(k) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-j2\pi kx/N}$ $k = 0, 1, 2, \ldots, N-1$	$f(x) = \sum_{k=0}^{N-1} F(k) e^{j2\pi kx/N}$ $x = 0, 1, 2, \ldots, N-1$

Fourier transform

Where is the connection to the 'summation of sine waves' idea?

$$f(x) = \sum_{k=0}^{N-1} F(k) e^{j2\pi kx/N}$$

Euler's formula

$$e^{j\theta} = \cos\theta + j\sin\theta$$

sum over frequencies

$$f(x) = \sum_{k=0}^{N-1} F(k) \left\{ \cos(2\pi kx) + j\sin(2\pi kx) \right\}$$

scaling parameter

wave components

2D Fourier Transform

Definition

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}\,dx\,dy,$$
$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}\,du\,dv$$

where u and v are spatial frequencies.

Also will write FT pairs as $f(x,y) \Leftrightarrow F(u,v)$.

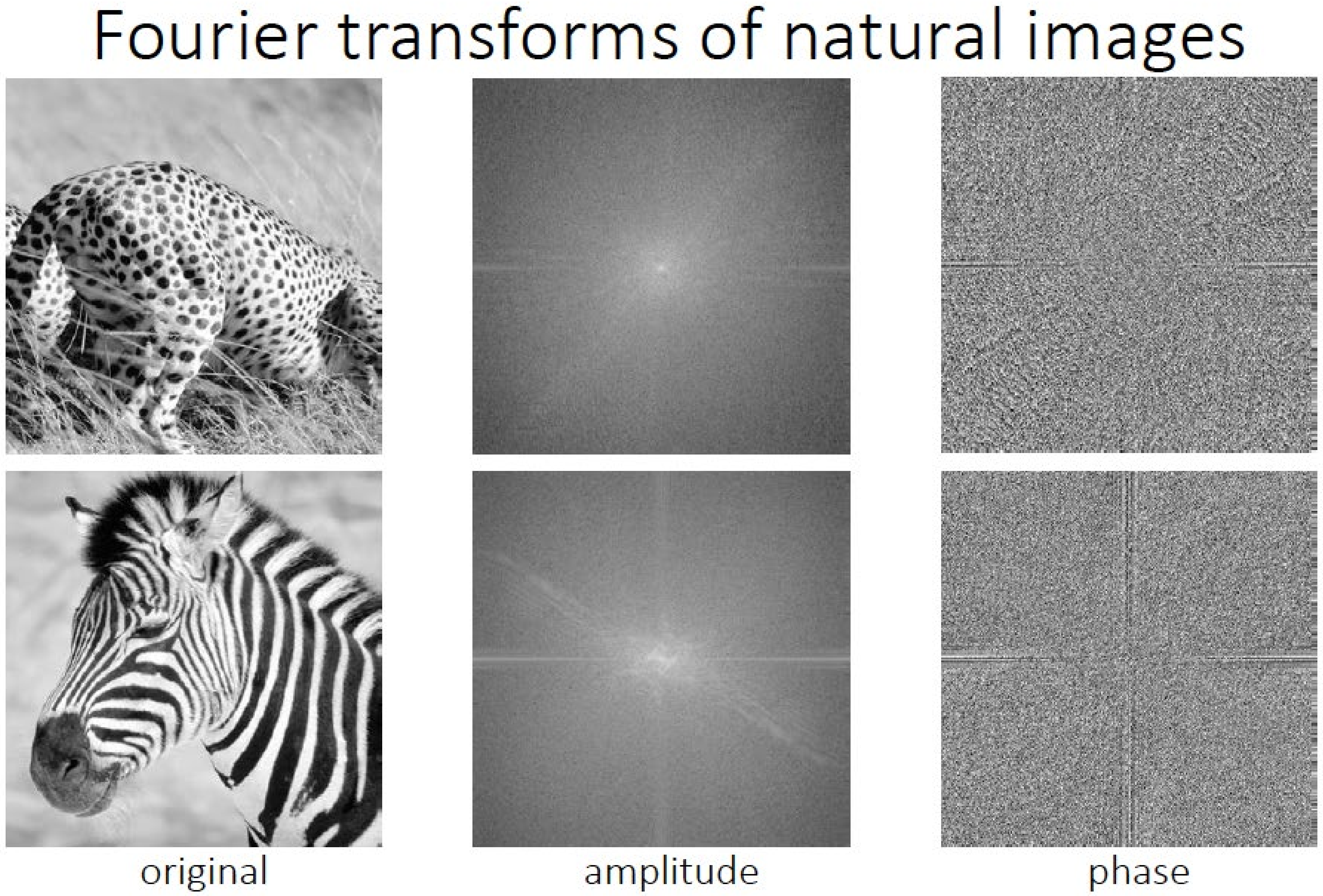
- $F(u,v)$ is complex in general,

$$F(u,v) = F_{\mathrm{R}}(u,v) + jF_{\mathrm{I}}(u,v)$$

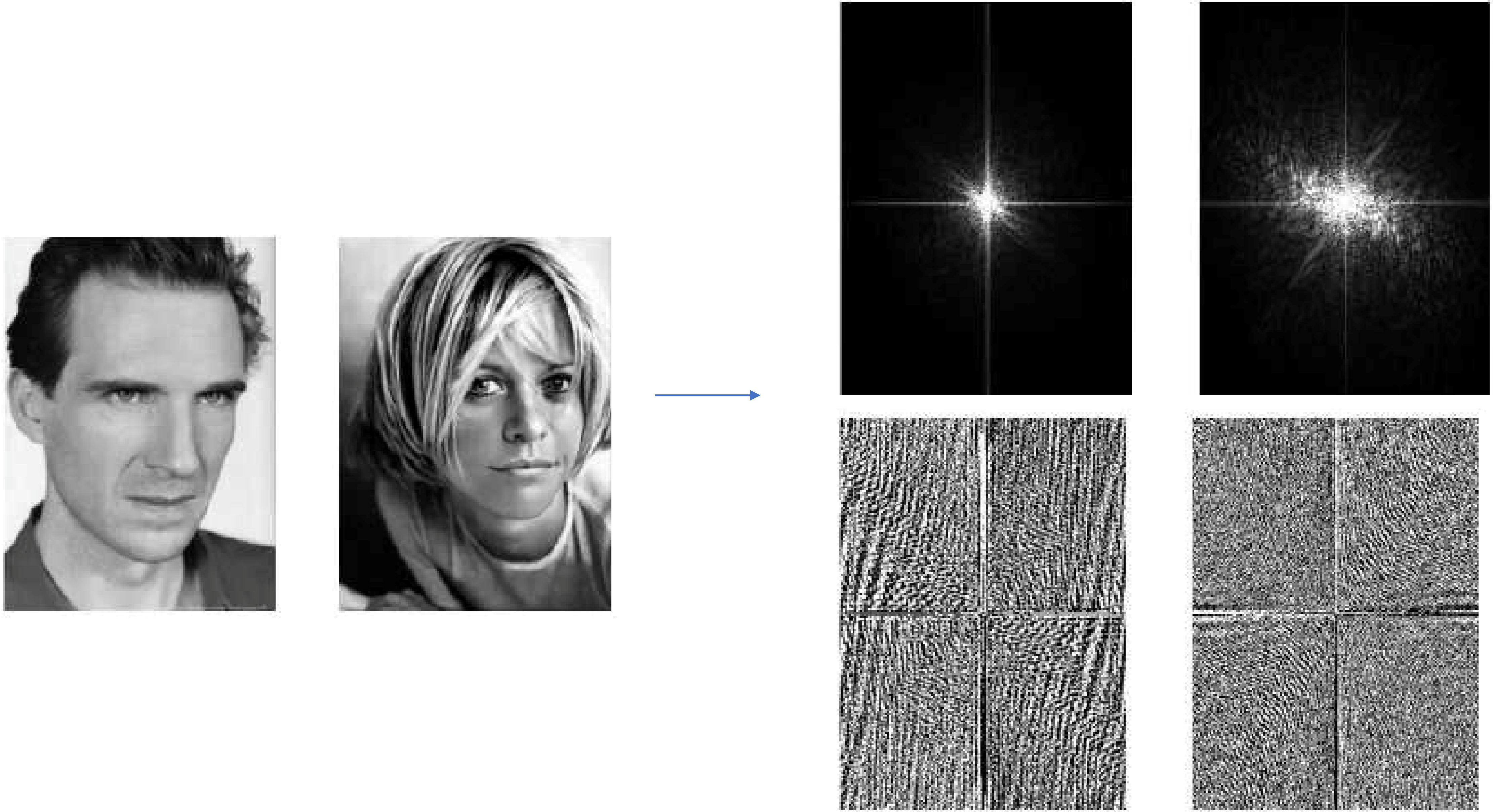
- $|F(u,v)|$ is the magnitude spectrum
- $\arctan(F_{\mathrm{I}}(u,v)/F_{\mathrm{R}}(u,v))$ is the phase angle spectrum.

Slides courtesy of A. Zissserman

Frequency Domain of the Image



More examples

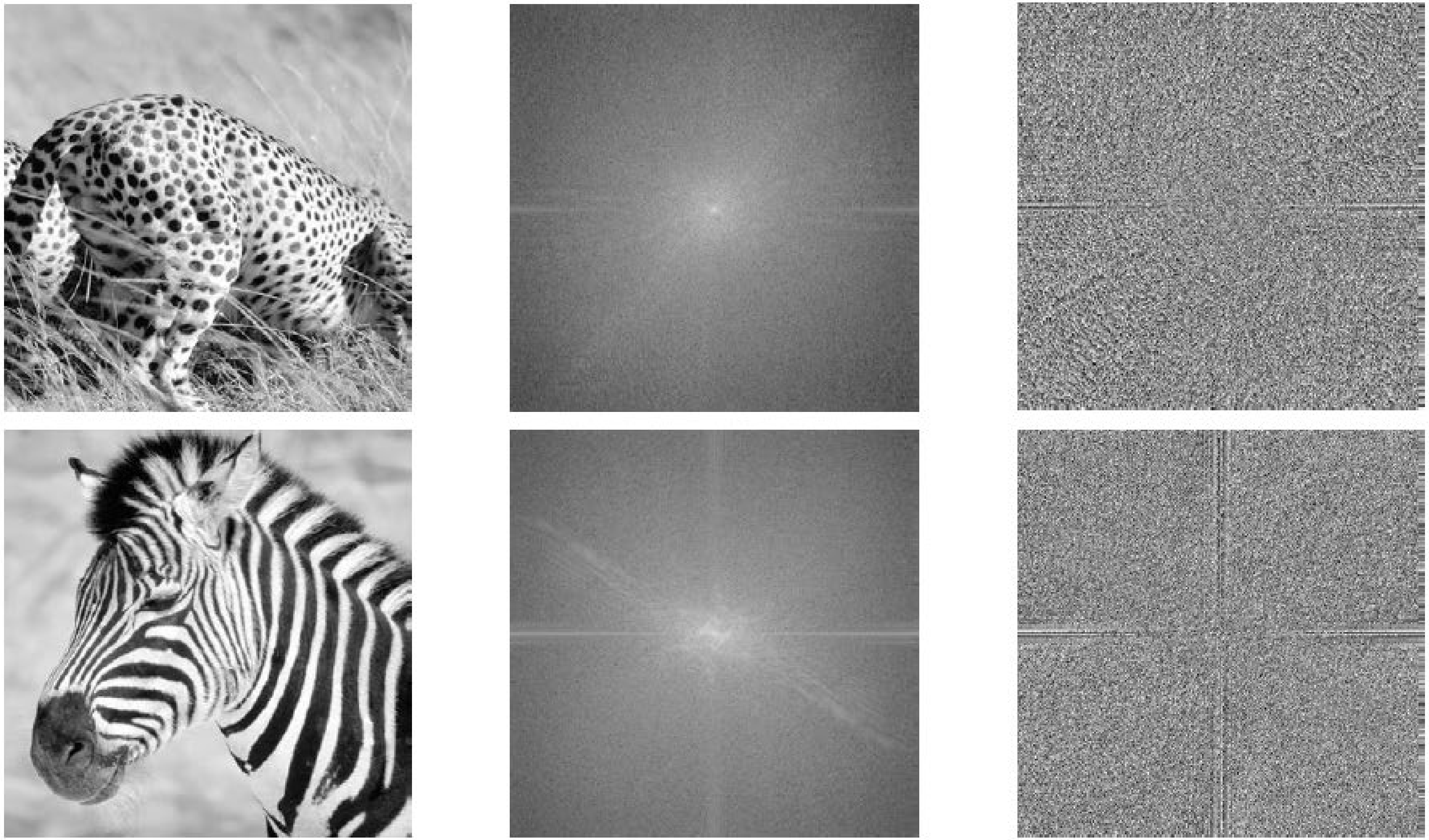


mag

phase

Phase

Fourier transforms of natural images

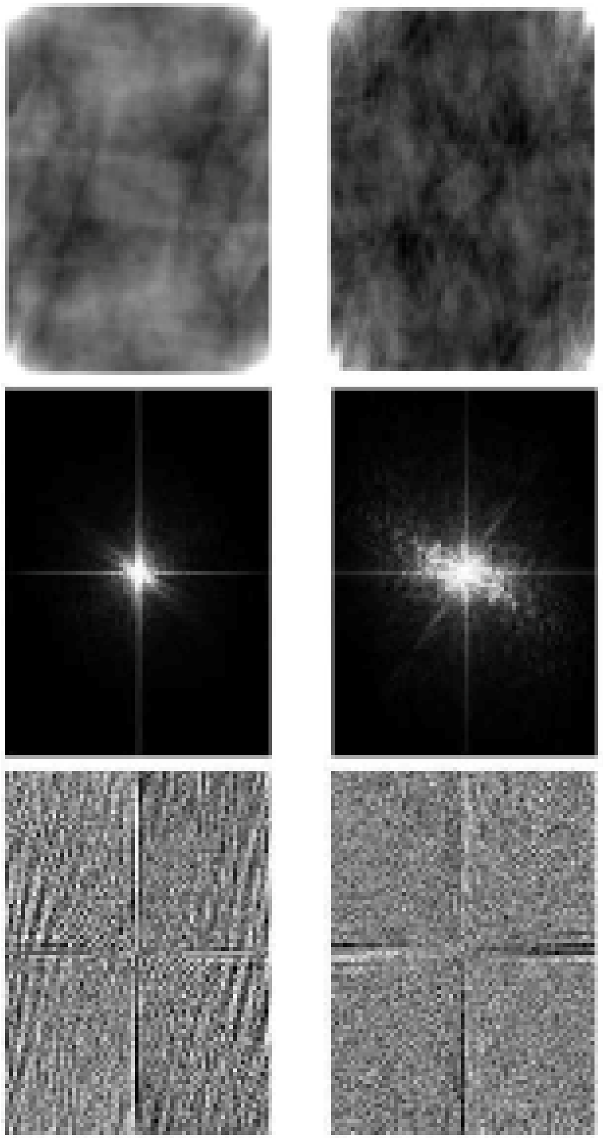


original amplitude phase

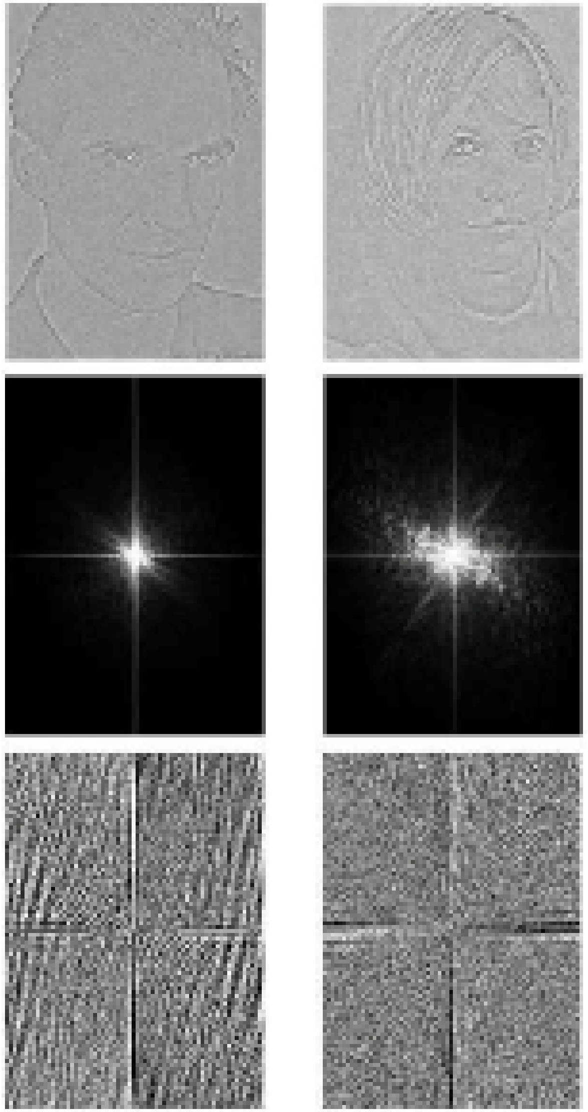
“We generally do not display phase images because most people who see them shortly thereafter succumb to hallucinogenics or end up in a Tibetan monastery” – John Brayer

Magnitude only and phase only reconstructions

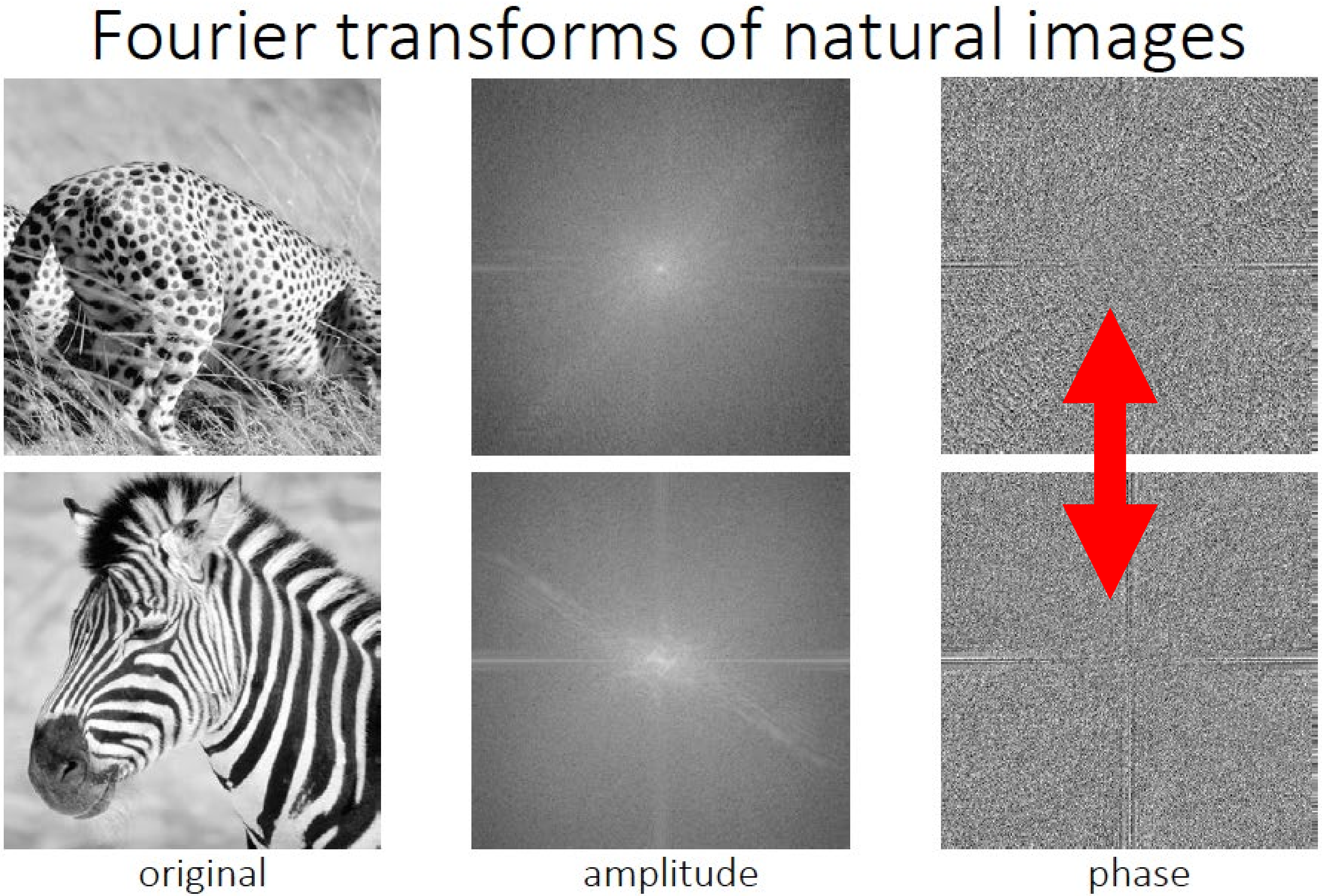
Reconstruction using
magnitude only
Top Left Photo: Ralph's magnitude is the same, Phase = 0
Top Right Photo: Meg's magnitude is the same, Phase = 0



Reconstruction using
phase only
Top Left Photo: Ralph's magnitude normalized to one, Phase is the same
Top Right Photo: Meg's magnitude normalized to one, Phase is the same



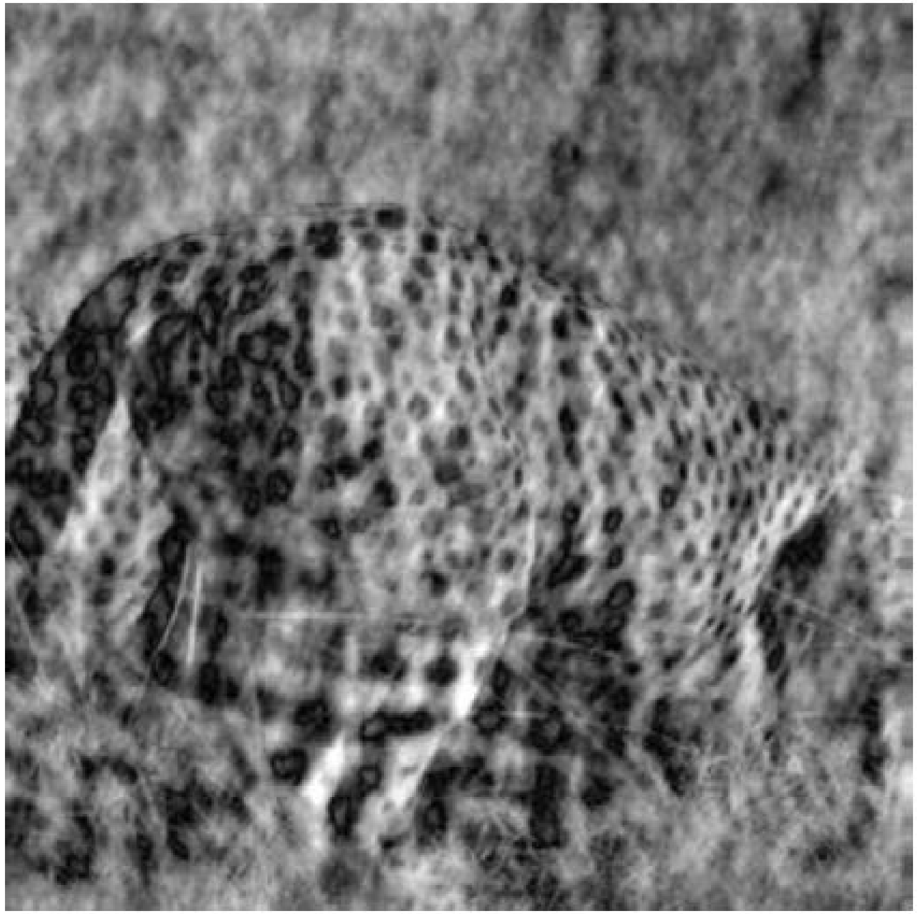
Phase swapping



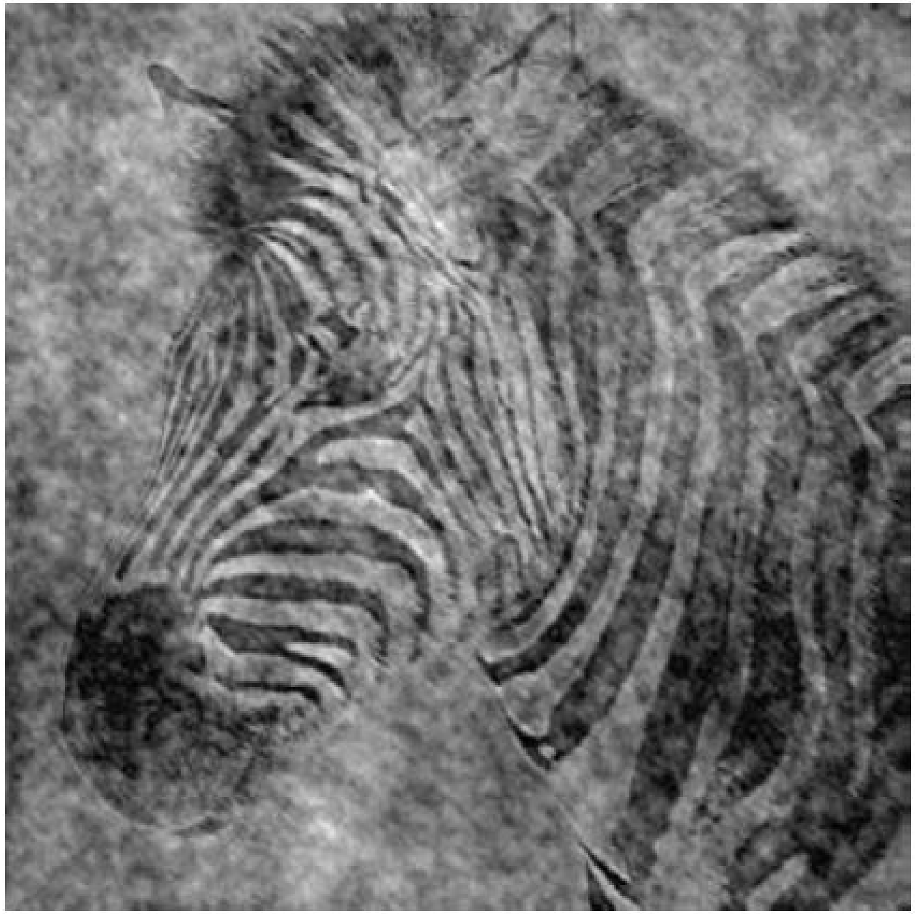
What if we took the phase of each image, swapped it, and did the inverse Fourier transform?

Phase Swapping

Image phase matters!



cheetah phase with zebra amplitude



zebra phase with cheetah amplitude

The convolution theorem

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

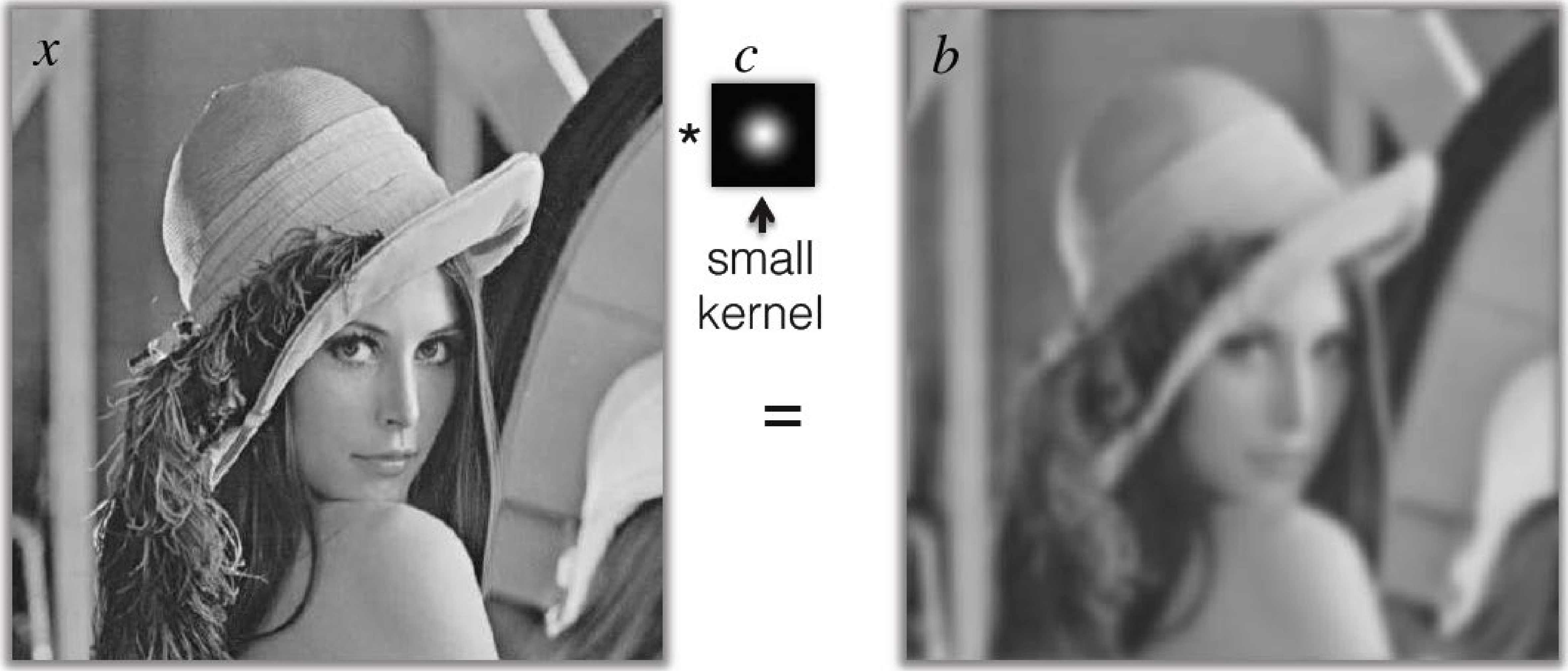
$$\mathcal{F}\{g * h\} = \mathcal{F}\{g\}\mathcal{F}\{h\}$$

The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms:

$$\mathcal{F}^{-1}\{gh\} = \mathcal{F}^{-1}\{g\} * \mathcal{F}^{-1}\{h\}$$

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

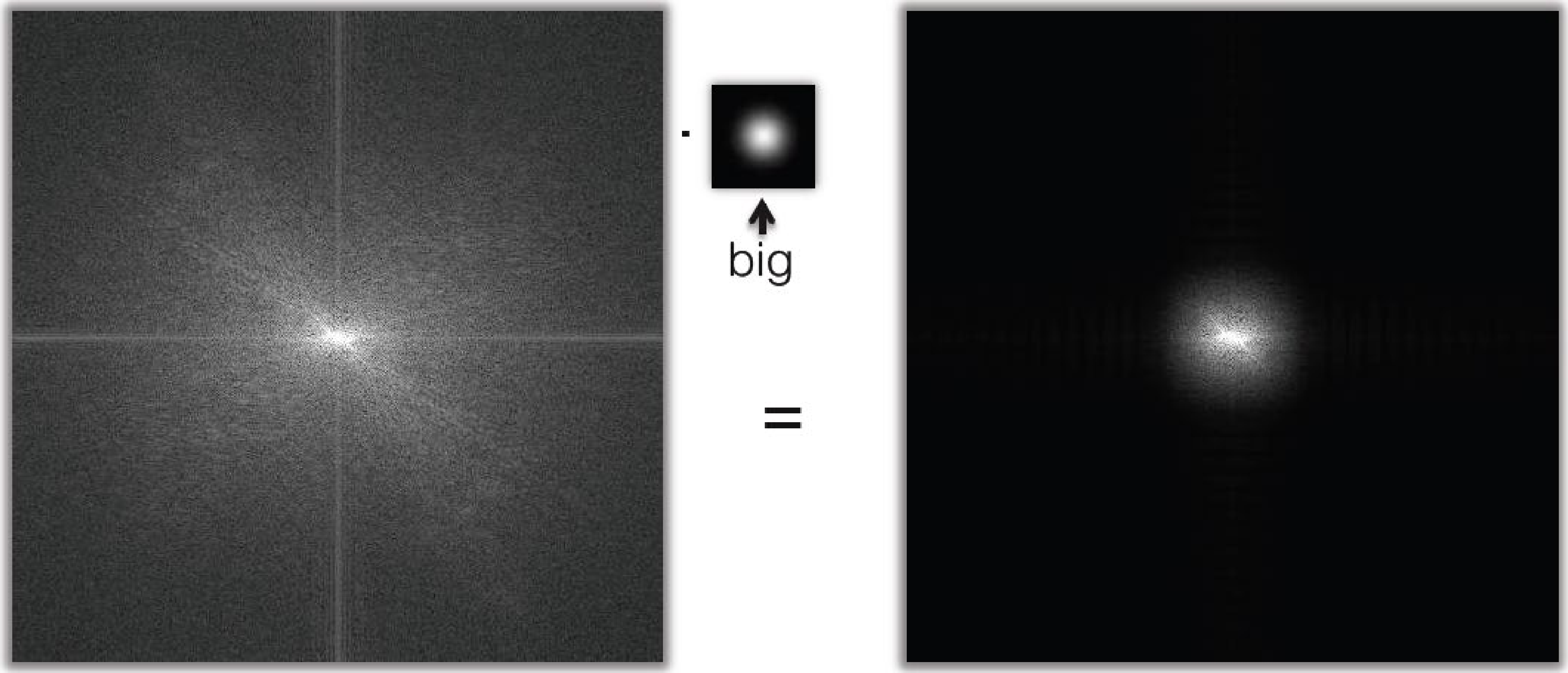
- low-pass filter: convolution in primal domain $b = x * c$



Slides courtesy of G. Wetzstein

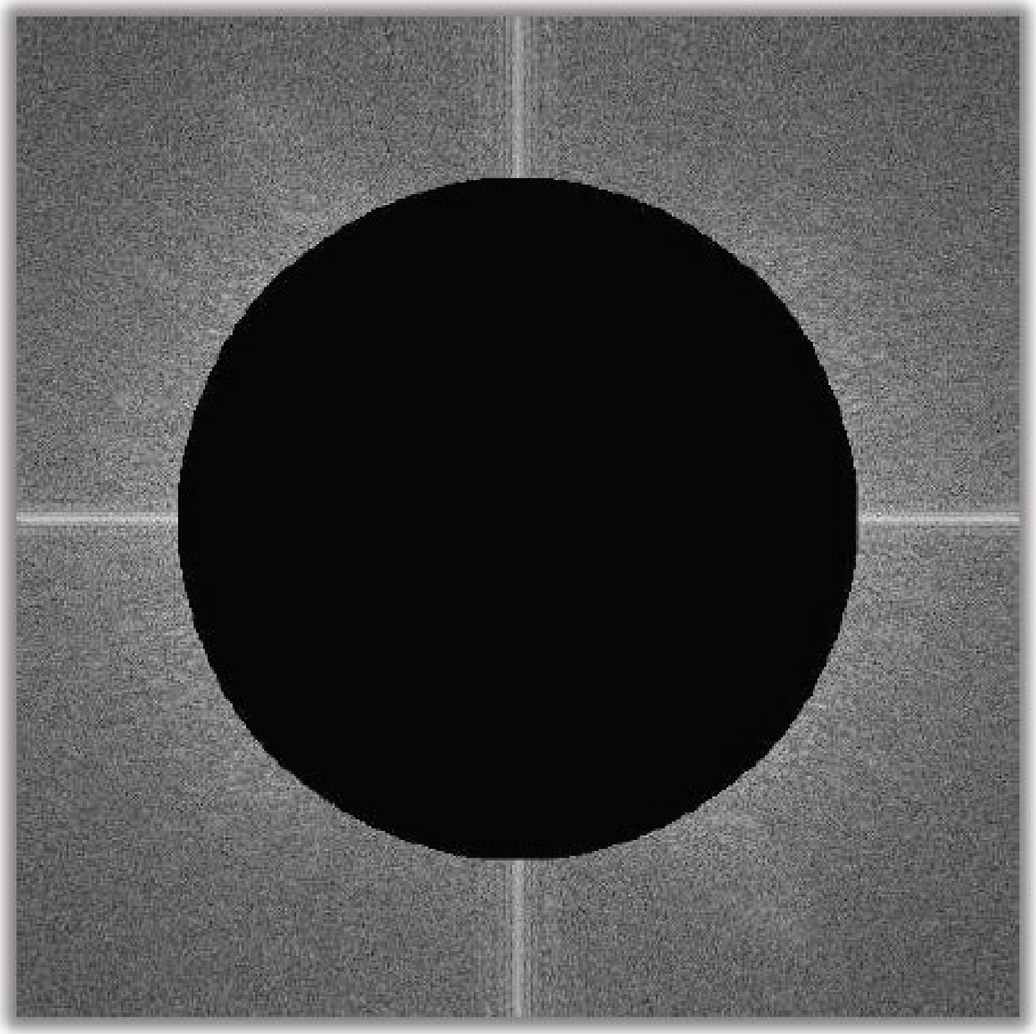
$$F\{b\} = F\{x\} \cdot F\{c\}$$

- low-pass filter: multiplication in frequency domain



Slides courtesy of G. Wetzstein

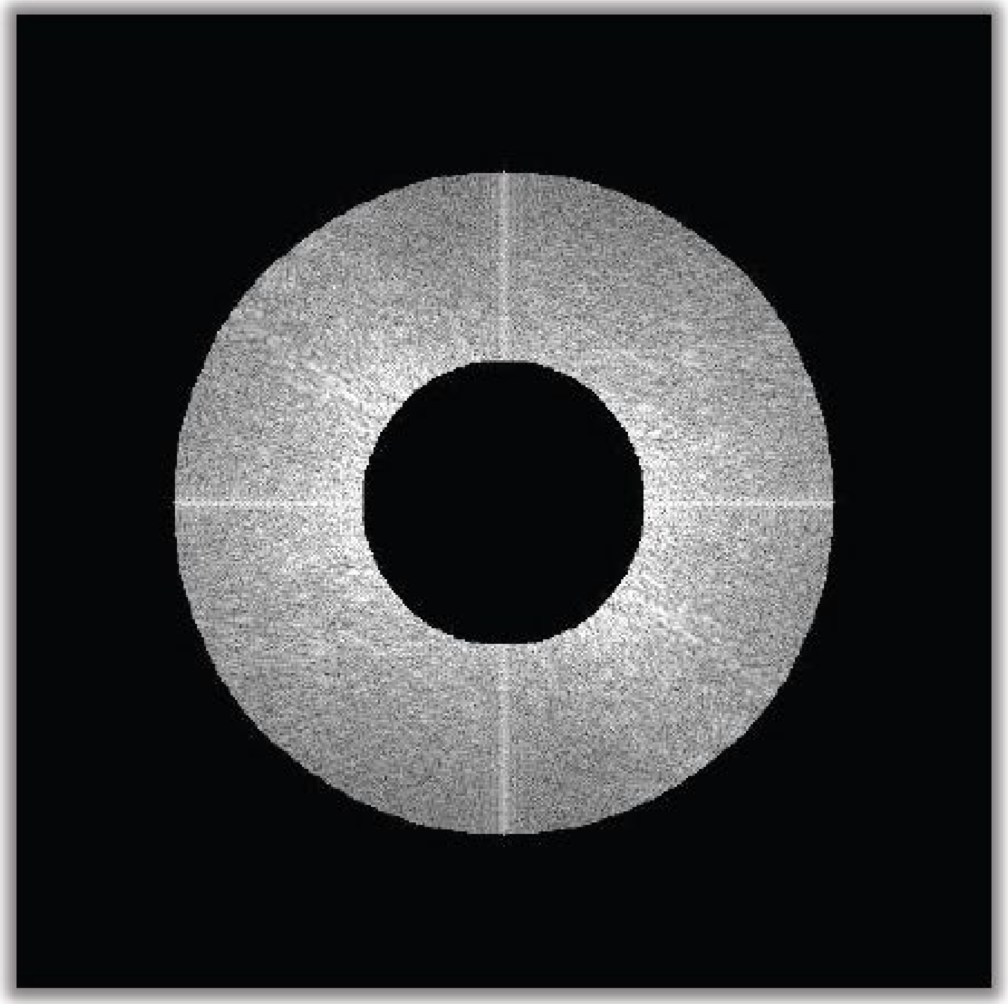
High Pass Filter



Slides courtesy of G. Wetzstein

Bandpass filtering

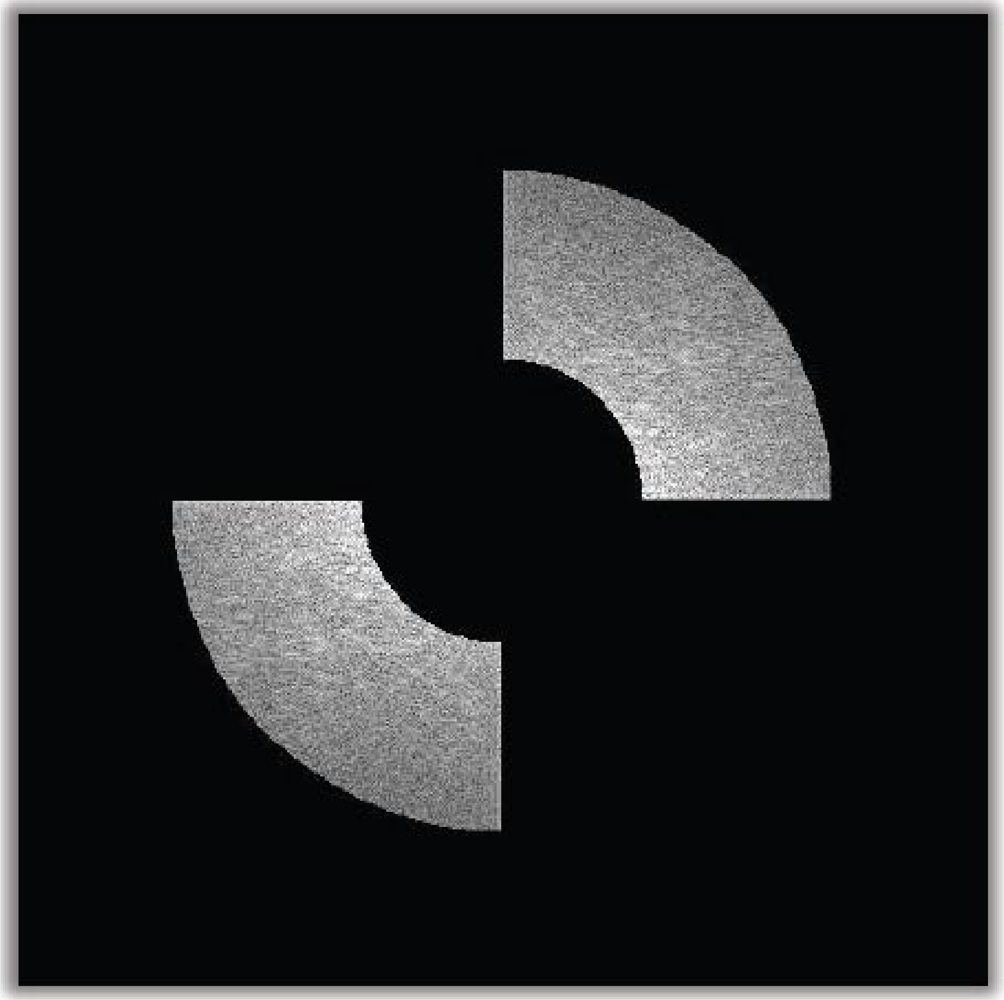
←



Slides courtesy of G. Wetzstein

- edges with specific orientation (e.g., hat) are gone!

←



Slides courtesy of G. Wetzstein