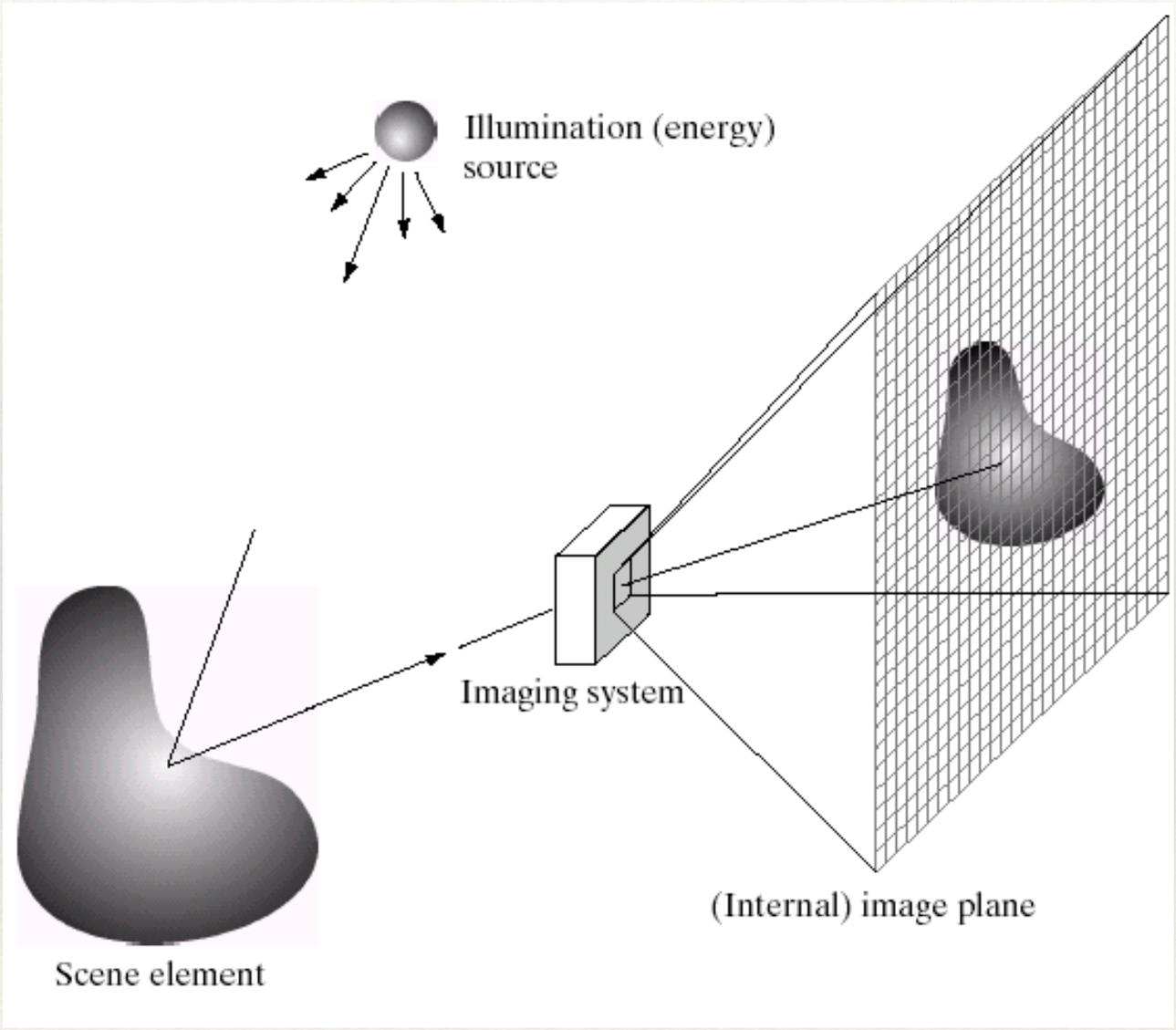
Xu Zhao @ Shanghai Jiao Tong university

Lecture 3-1: Image Filtering

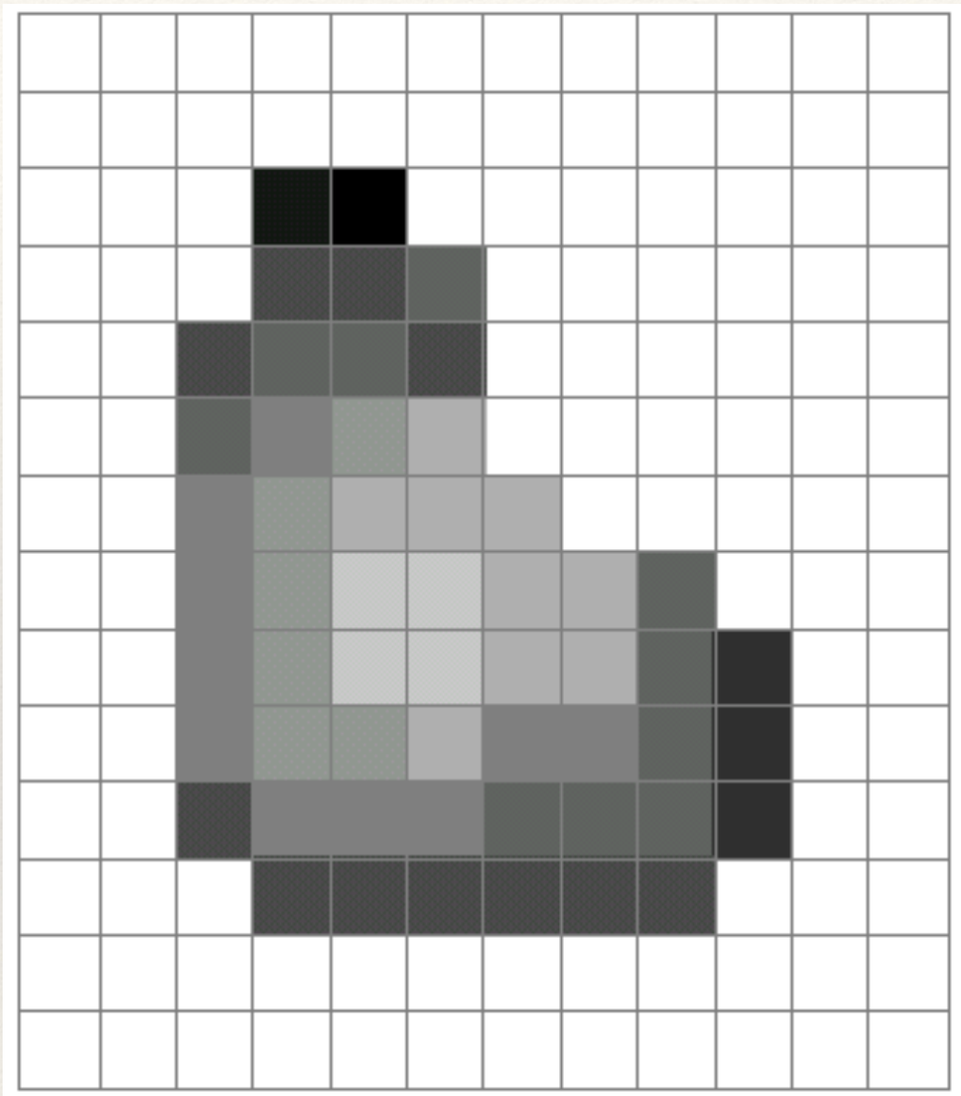
Contents

- **Spatial filtering**
  - Linear filter
  - Non-linear filter
- **Frequency perspective**
  - Spatial frequency
  - Fourier transform

Image formation



Camera Sensor

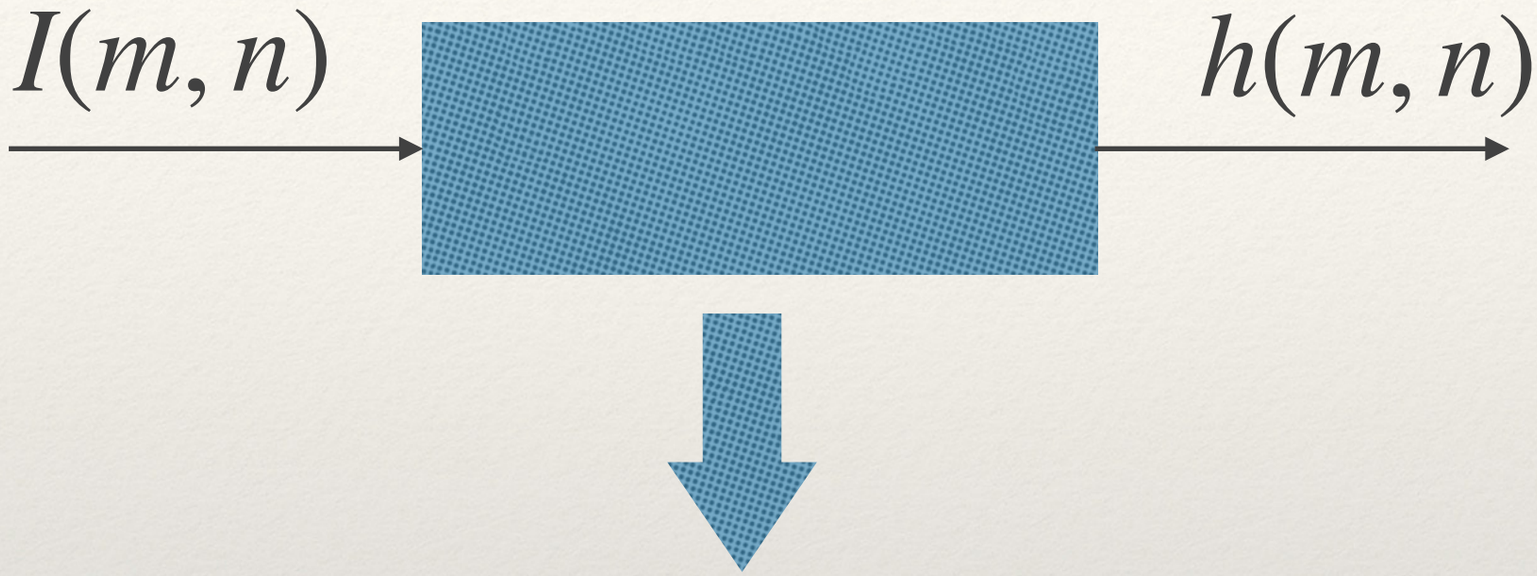


Output Image

Image filtering, why and how?

- Motivation
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)
- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

Linear filtering



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

For a linear system, each output is a linear combination of all the input values

Example - Box filter

$f[\cdot,\cdot]$

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

Slide credit: David Lowe (UBC)

Example - Box filter

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

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

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

	0								

m =1, n = 1

k,l = [-1,0,1]

Example - Box filter

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

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

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

	0	10							

m =2, n = 1

k,l = [-1,0,1]

Example - Box filter

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

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

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

	0	10	20						

m =3, n = 1

k,l = [-1,0,1]

Example - Box filter

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

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

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

	0	10	20	30					

m =4, n = 1

k,l = [-1,0,1]

Example - Box filter

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

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

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

	0	10	20	30	30				

m =5, n = 1

k,l = [-1,0,1]

Example - Box filter

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

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

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

	0	10	20	30	30				
				?					

m =4, n = 6

k,l = [-1,0,1]

Example - Box filter

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

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

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

	0	10	20	30	30				
						?			
				50					

m =6, n = 4

k,l = [-1,0,1]

Example - Box filter

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

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

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

	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	

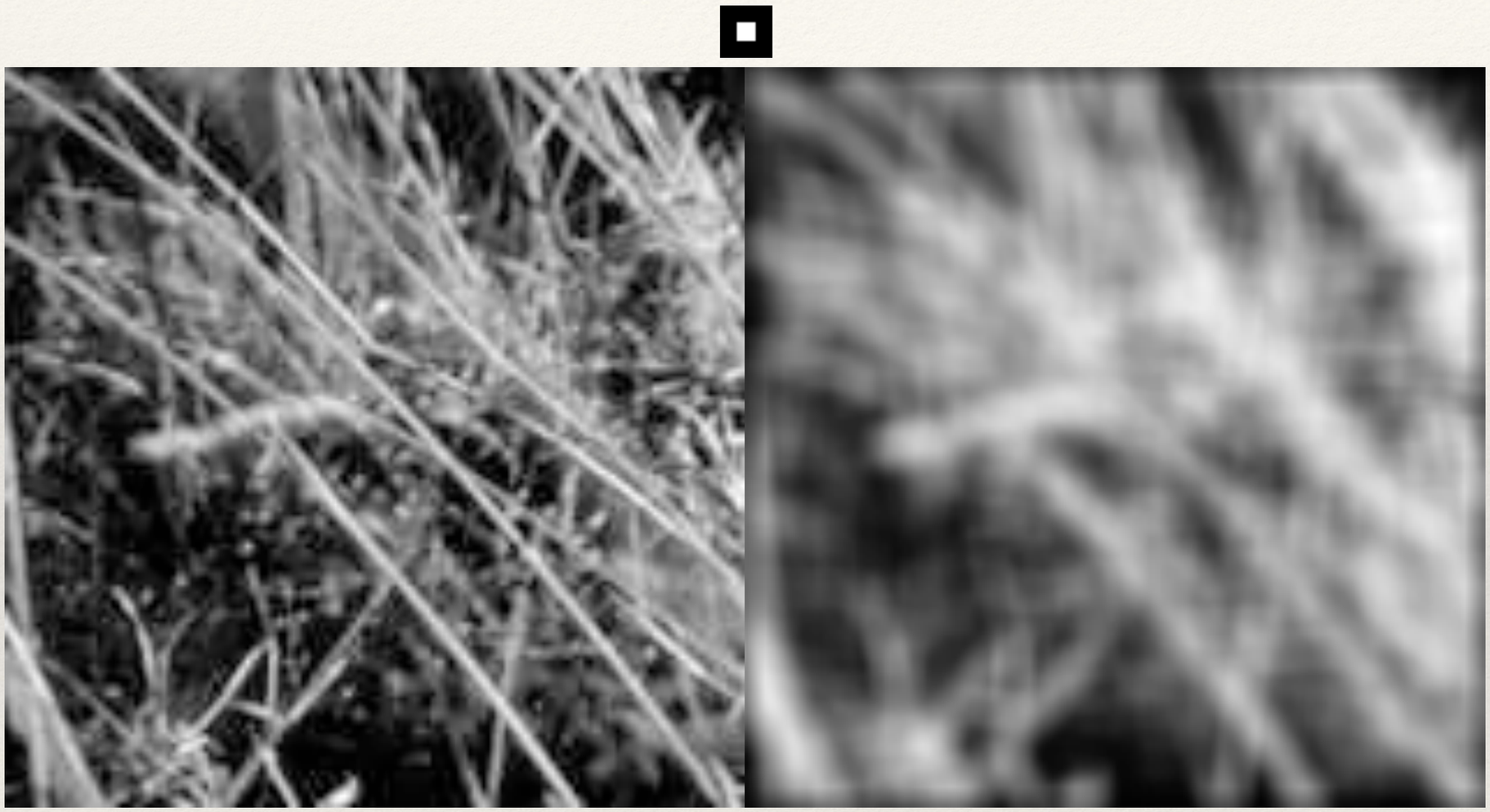
Box filter

- What does it do?
  - Replaces each pixel with an average of its neighborhood
  - Achieve smoothing effect (remove sharp features)

$$f[\cdot,\cdot]$$

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

Smoothing with box filter



$f[\cdot,\cdot]$

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

Integral image

- Don’t forget to use integral image to improve computational efficiency for utilization of bunch of box filters.

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

(a) S = 24

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

(b) s = 28

3	5	12	*14*	17
4	11	19	24	31
9	17	28	38	46
13	24	37	**48**	62
15	30	44	59	81

(c) S = 24

Correlation and Convolution

- 2D correlation:

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

- 2D convolution:

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

- **Convolution is the same as correlation with a 180° rotated filter kernel.**
- **Correlation and convolution are identical when the filter kernel is symmetric.**

Key properties of linear filters

- **Linearity**:

  `imfilter(I, f1+f2)=imfilter(I,f1)+imfilter(I,f2)`

- **Shift invariance**: same behavior given intensities regardless of pixel location m,n

  `imfilter(I,shift(f)) = shift(imfilter(I,f))`

- Any linear, shift-invariant operator can be represented as a convolution.

Convolution properties

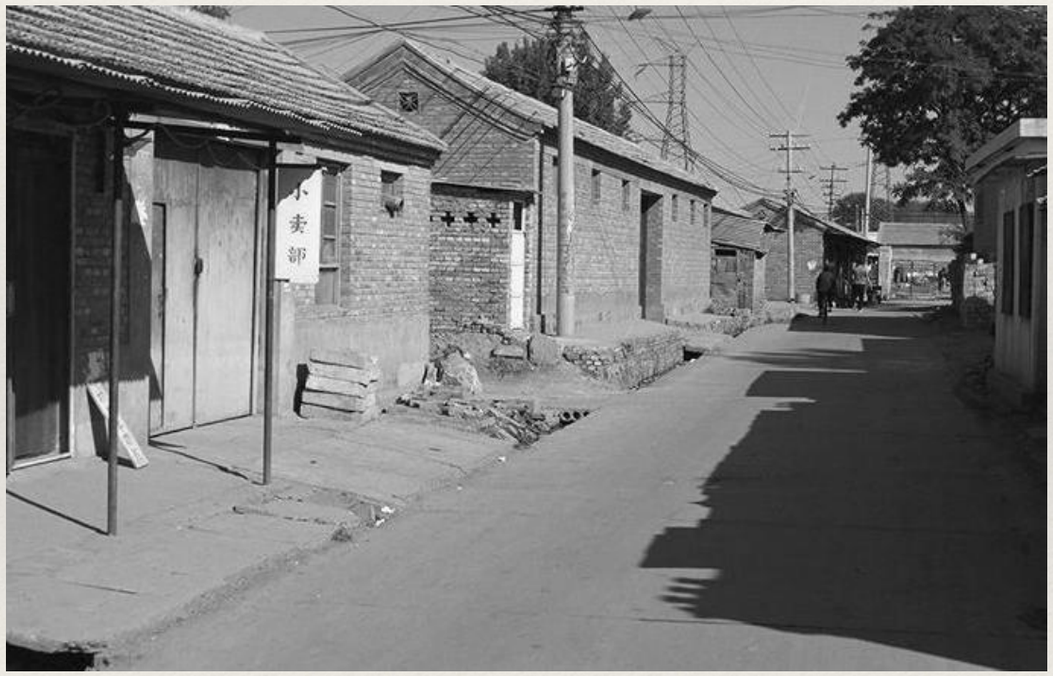
- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
  - Correlation is _not_ associative (rotation effect)
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k (a * b)$
- Identity: unit impulse $e = [0, 0, 1, 0, 0]$, $a * e = a$

Source: S. Lazebnik

Correlation for template matching

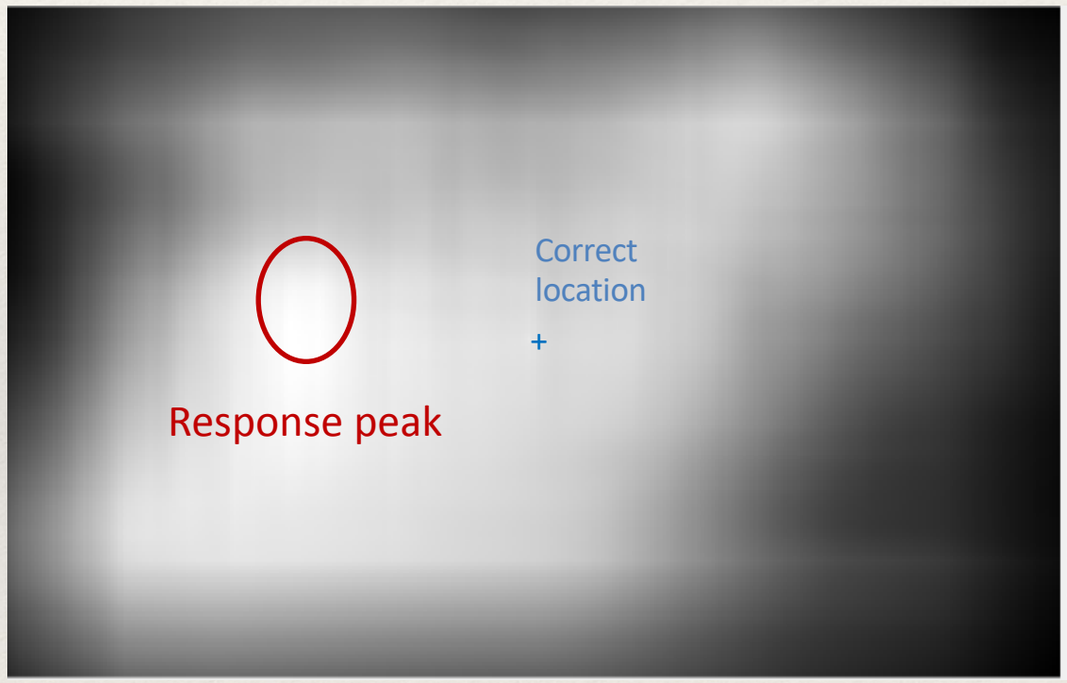
Let's see if we can use correlation to 'find' the parts of the image that look like the filter.

D



f

I = correlate2d(D, f, 'same')



It does't work. Why?

[Thanks to Robert Collins @ Penn State]

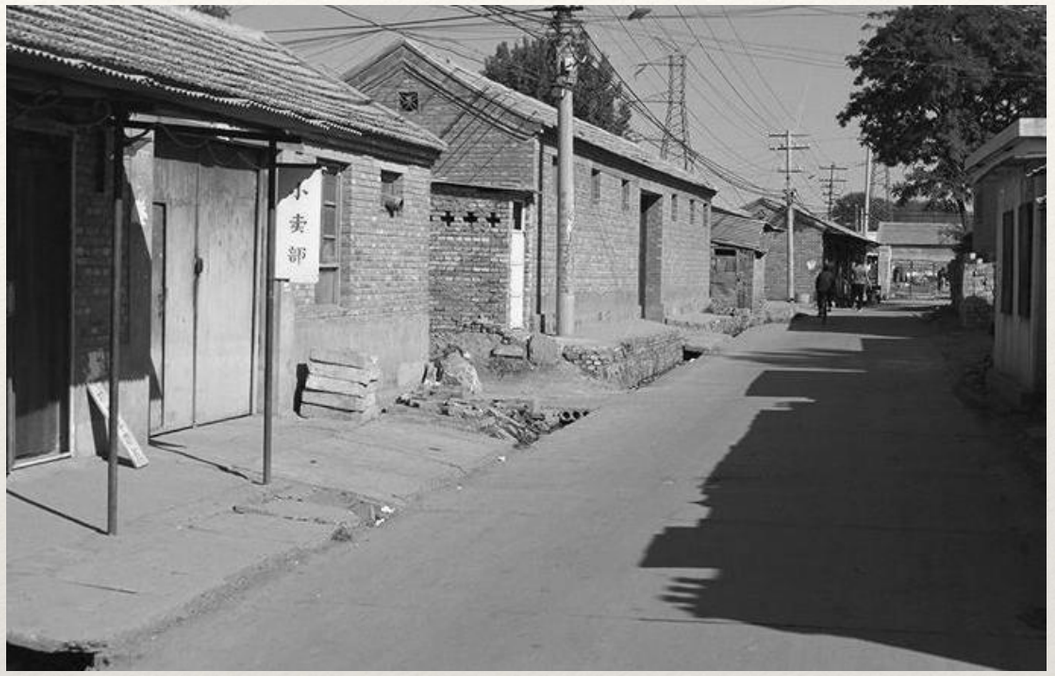
Correlation for template matching

- It does't work. Why?

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

- As brightness in *I* increases, the response in *h* will increase, as long as *f* is positive.
- Overall brighter regions will give higher correlation response.

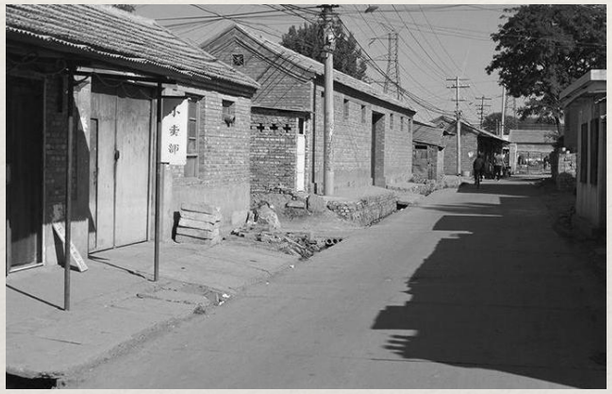
D

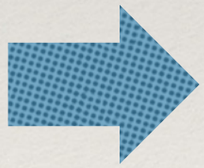


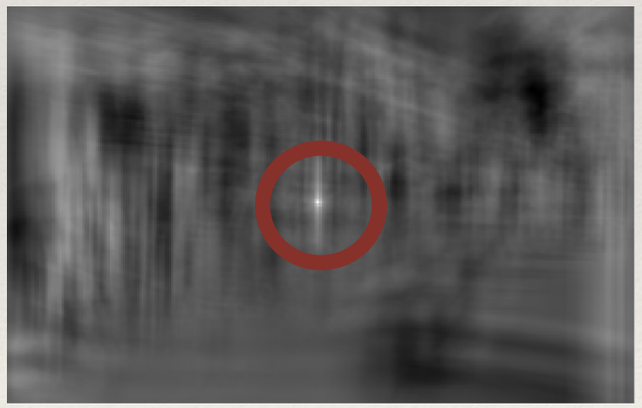
f

Correlation for template matching

- **Normalization correlation**. Subtract the mean so zero centered. Score is higher only when dark parts match and when light parts match.
- This is a great way to build simple, fast pattern detectors that require little computation.







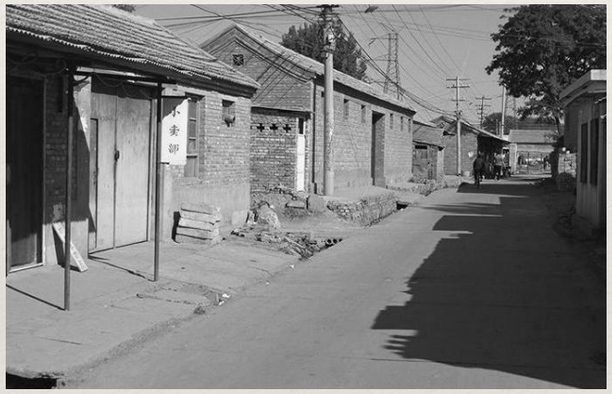
D2 = D – np.mean(D)

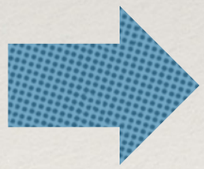
f2 = f – np.mean(f)

I2 = correlate2d(D2, f2, ‘same’)

Correlation for template matching

- What happens with convolution?





D2 = D – np.mean(D)

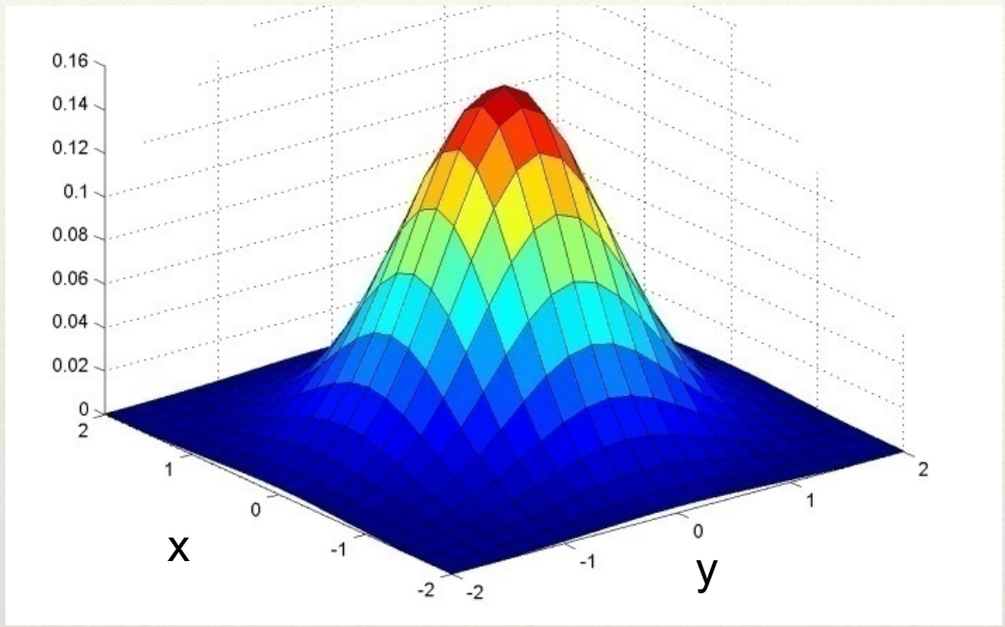
f2 = f – np.mean(f)

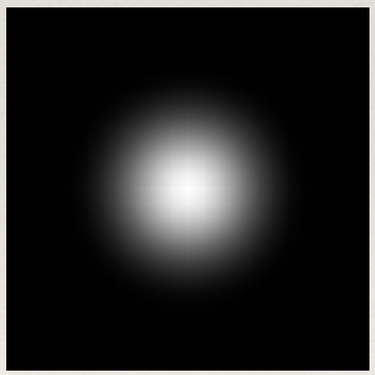
I2 = convolve2d(D2, f2, ‘same’)

Correlation for template matching

FIGURE 4.16: Examples of Freeman *et al.*'s system controlling a television set. Each state is illustrated with what the television sees on the **left** and what the user sees on the **right**. In **(a)**, the television is asleep, but a process is watching the user. An open hand causes the television to come on and show its user interface panel **(b)**. Focus on the panel tracks the movement of the user's open hand in **(c)**, and the user can change channels by using this tracking to move an icon on the screen in **(d)**. Finally, the user displays a closed hand in **(e)** to turn off the set. *This figure was originally published as Figure 12 of "Computer Vision for Interactive Computer Graphics," W. Freeman* et al., *IEEE Computer Graphics and Applications, 1998* © *IEEE, 1998.*

Gaussian filters





x

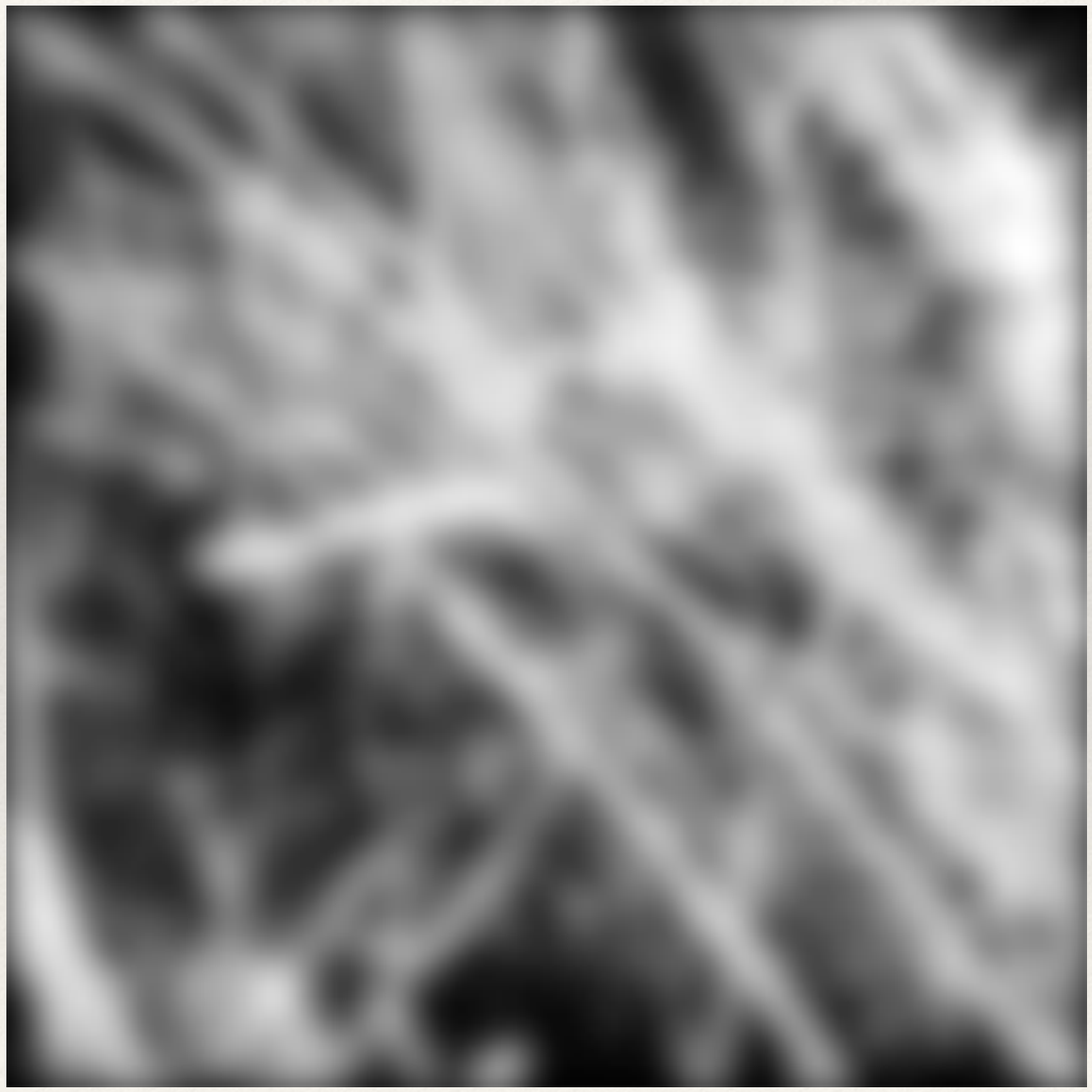
y					
	0.003	0.013	0.022	0.013	0.003
	0.013	0.059	0.097	0.059	0.013
	0.022	0.097	0.159	0.097	0.022
	0.013	0.059	0.097	0.059	0.013
	0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

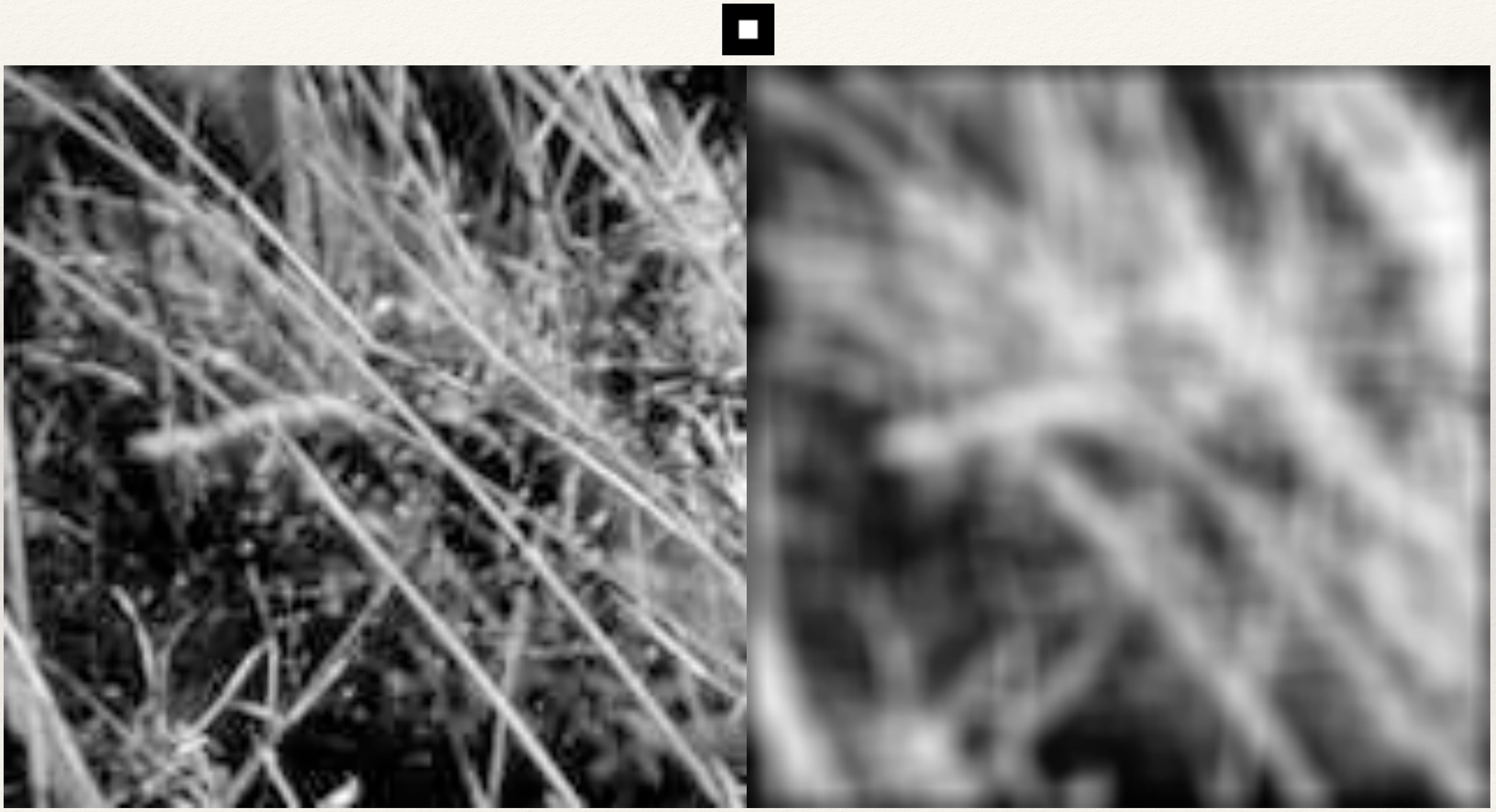
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Weight contributions of neighboring pixels by nearness

Smoothing with Gaussian filter



Smoothing with box filter



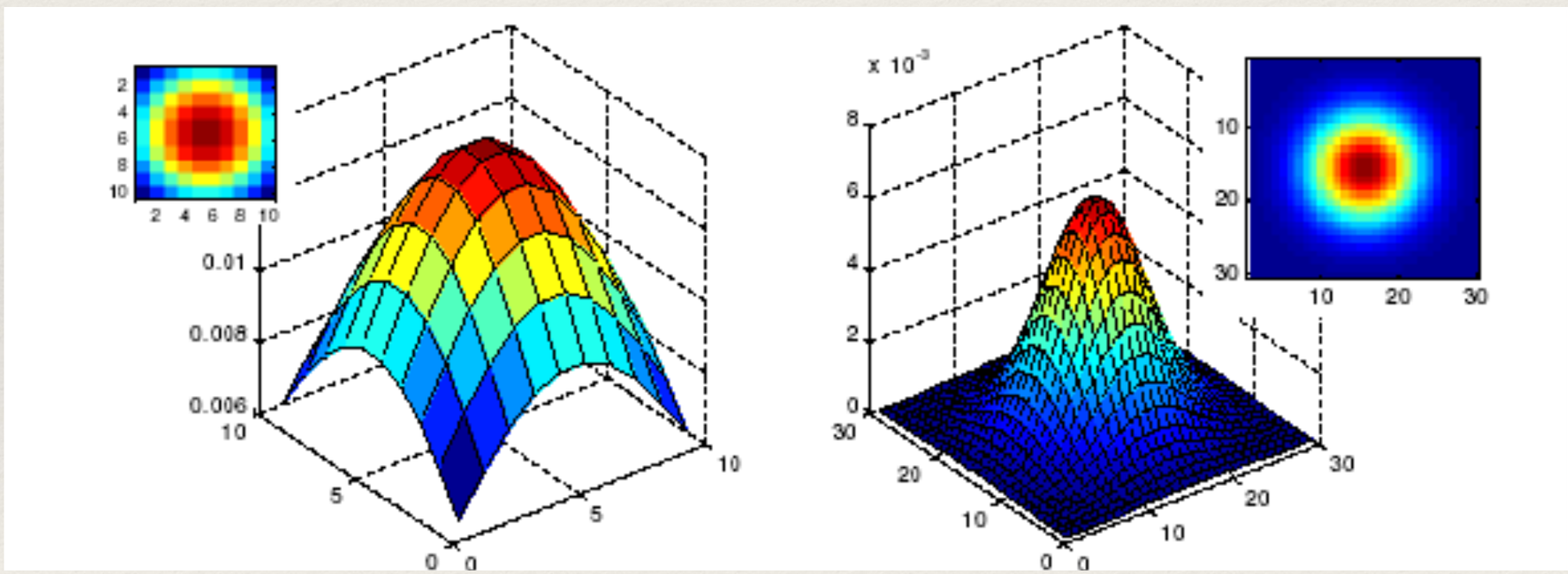
Gaussian filters

- Remove “high-frequency” components from the image (low-pass filter)
  - Images become more smooth
- Gaussian convolved with Gaussian is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  - Convolving twice with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- Separable kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

Gaussian filters

- What parameters matter here?
  - **Size** of kernel or mask

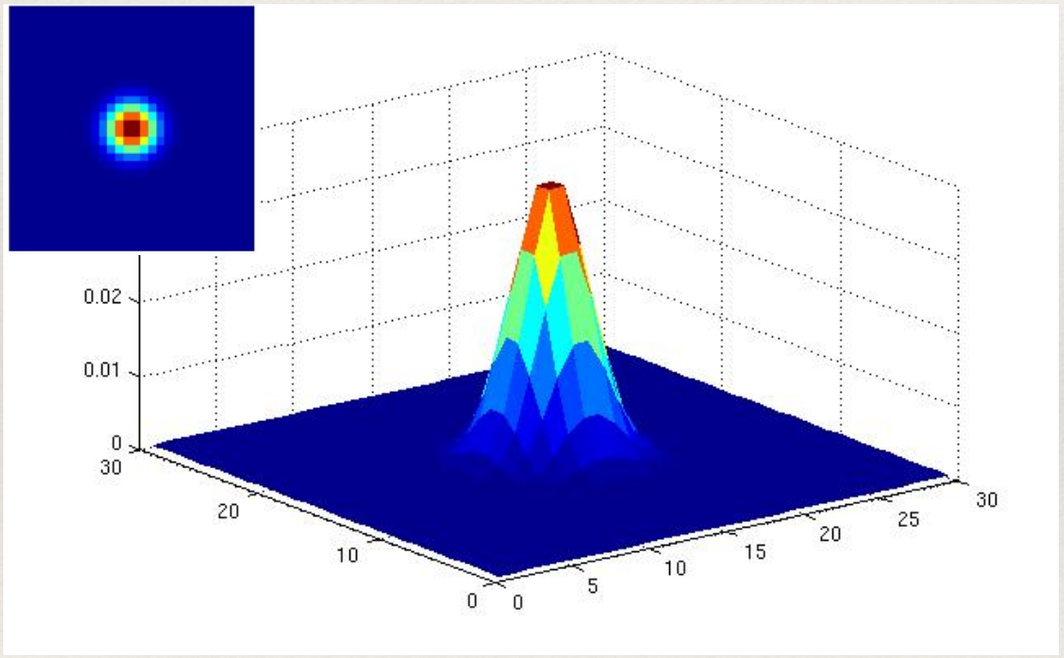


σ = 5 with 10 x 10 kernel

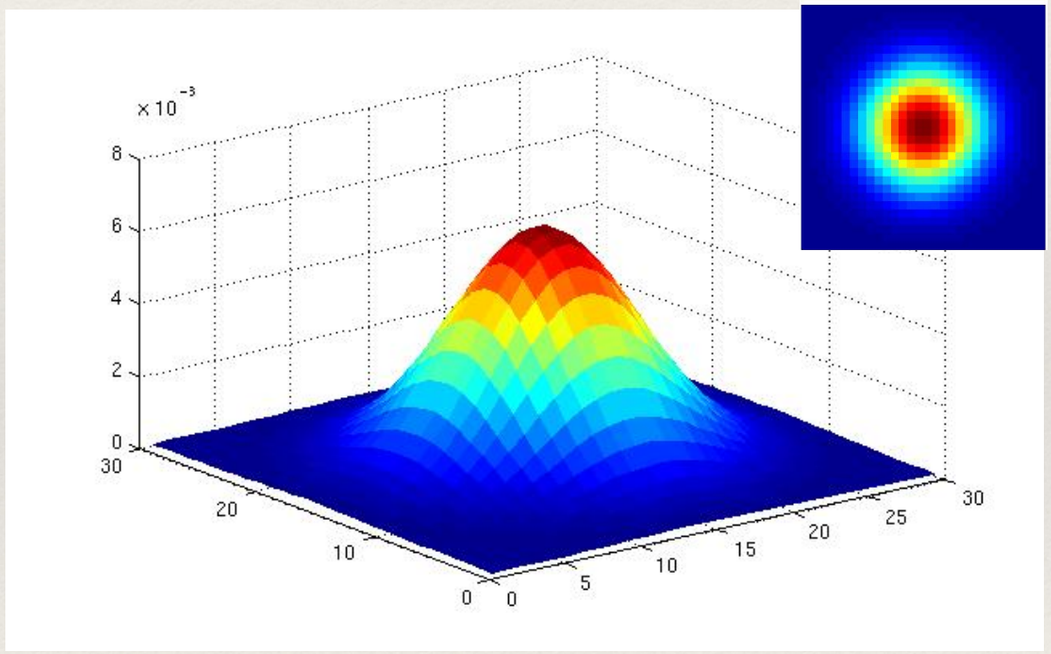
σ = 5 with 30 x 30 kernel

Gaussian filters

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



σ = 5 with 10 x 10 kernel



σ = 5 with 30 x 30 kernel

Source: K. Grauman

Gaussian filters: Separability

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

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

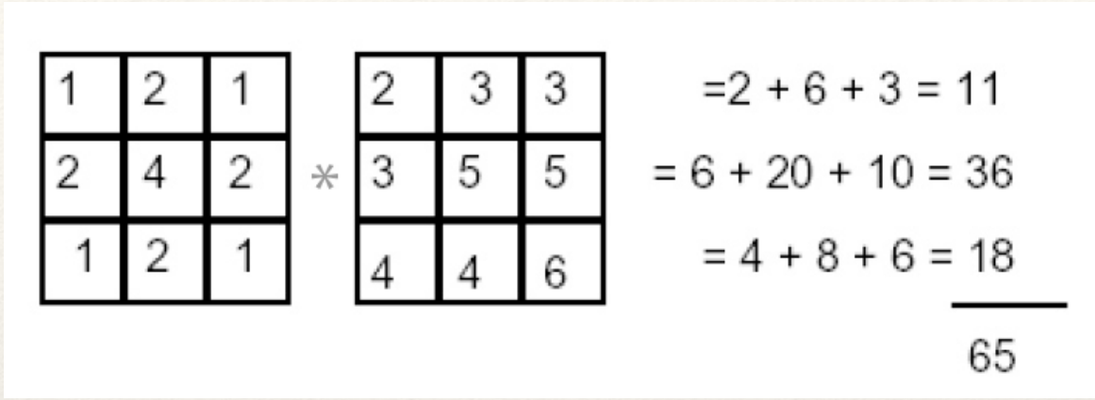
The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

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

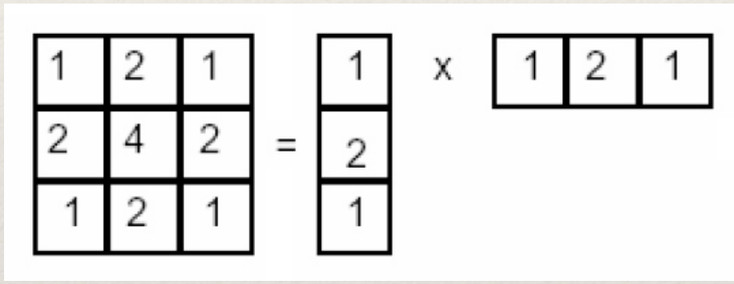
Source: D. Lowe

Separability

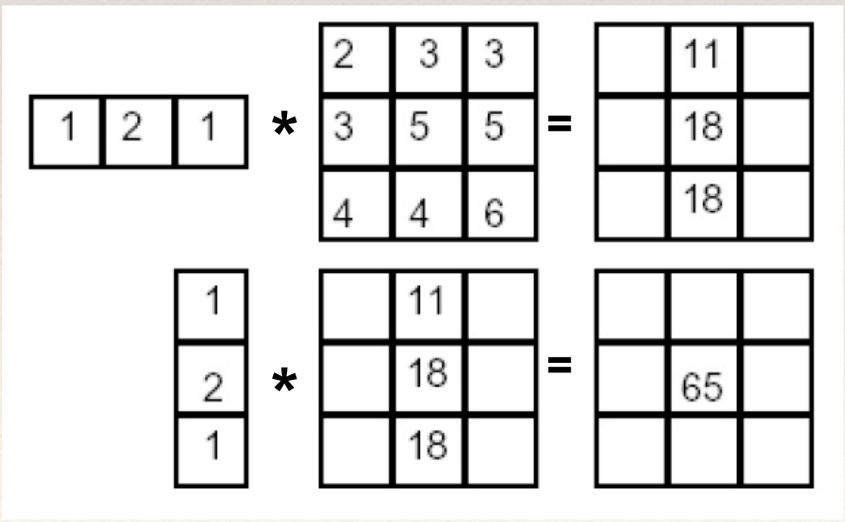
2D convolution
(center location only)



The filter factors
into a product of 1D
filters:



Perform convolution
along rows, followed by
convolution
along the remaining
column:



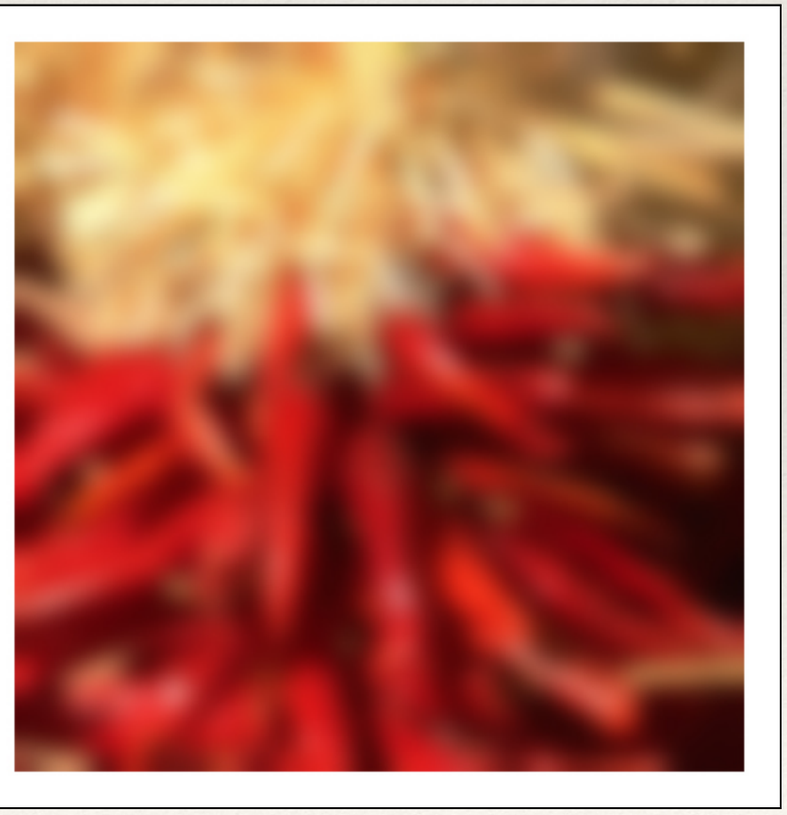
Source: K. Grauman

Separability

- Why is separability useful in practice?
  - MxN image, PxQ filter
    - 2D convolution: ~MNPQ multiply-adds
    - Separable 2D: ~MN(P+Q) multiply-adds
- Speed up = PQ/(P+Q)
  - 9x9 filter = ~4.5x faster

Practical matters

- 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



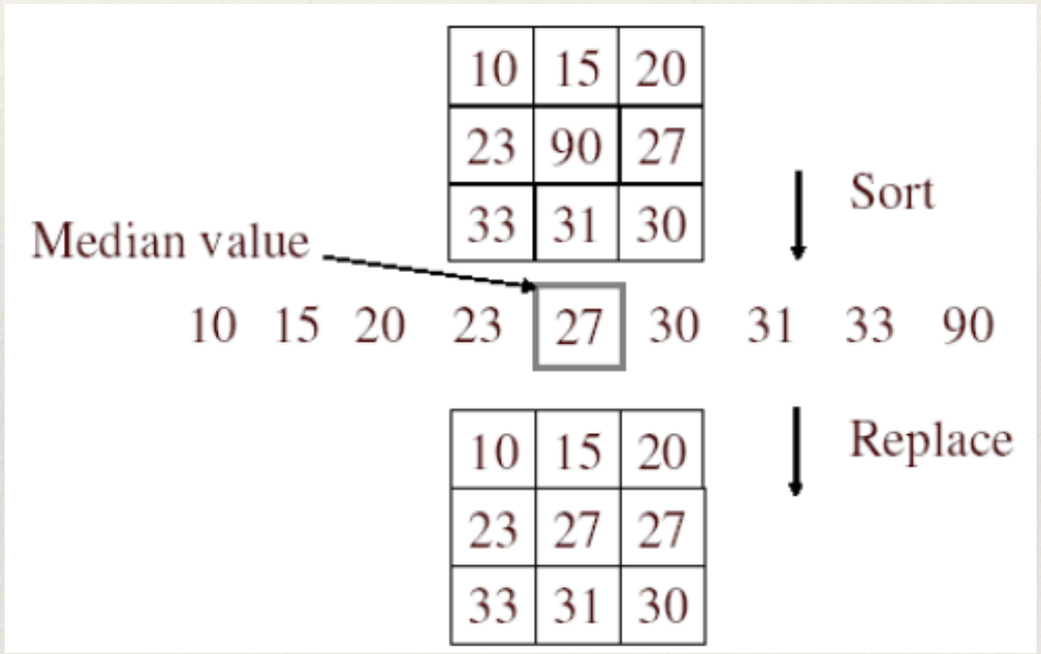
Source: S. Marschner

Convolution in Convolutional Neural Networks

- Convolution is the basic operation in CNNs
- Learning convolution kernels allows us to learn which `features' provide useful information in images

Non-linear filters

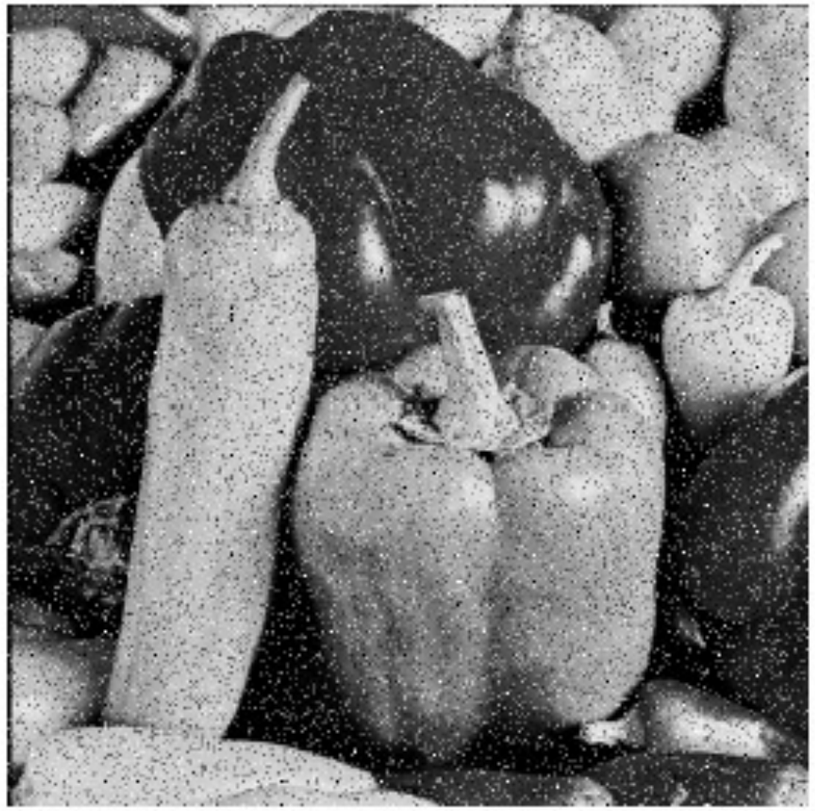
- Median filter
  - Operates over a window by selecting the median intensity in the window.
  - ‘Rank’ filter as based on ordering of gray levels
    - E.G., min, max, range filters



Non-linear filters

- Median filter
  - Operates over a window by selecting the median intensity in the window.
  - ‘Rank’ filter as based on ordering of gray levels
    - E.G., min, max, range filter
  - Very efficient to remove salt-and–pepper noise
  - Variants: α-trimmed mean, weighted median

Salt and pepper noise



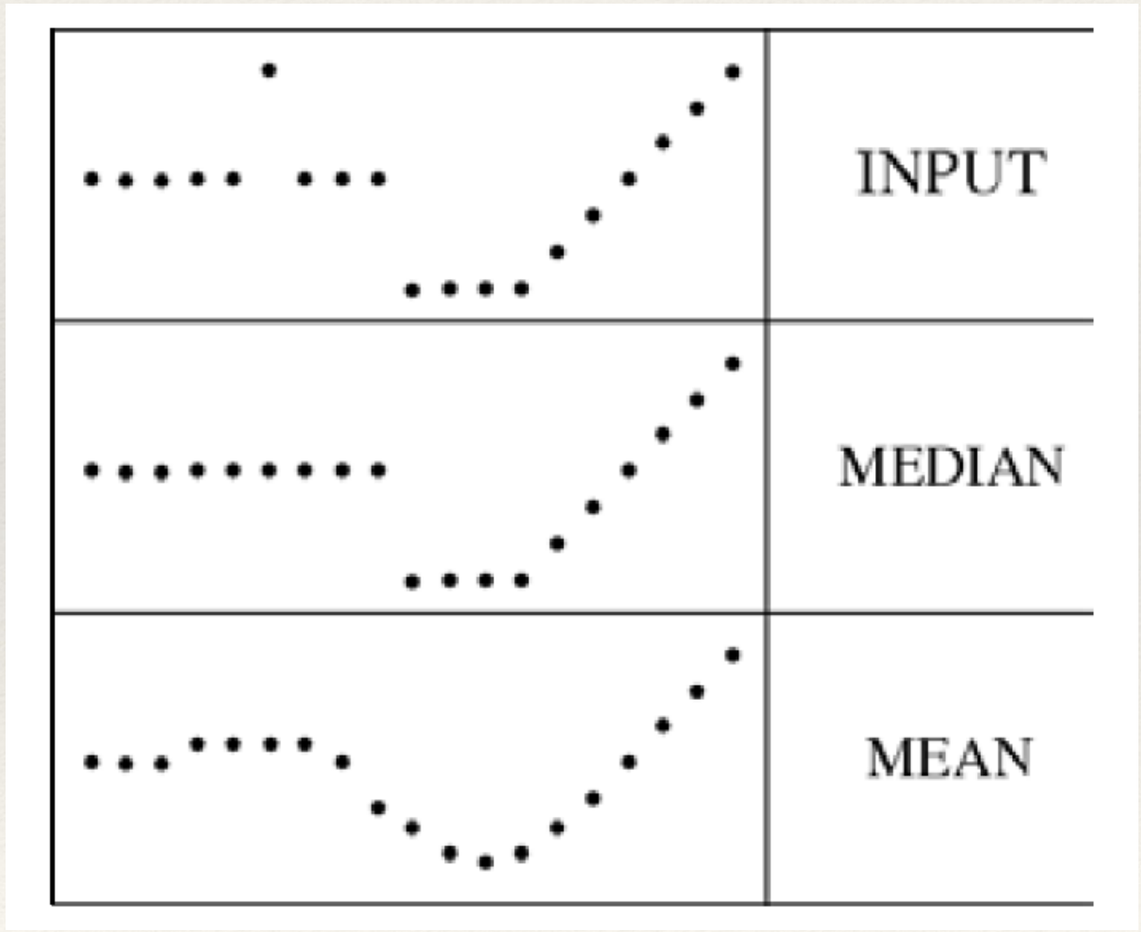
Median filtered

Plots of a row of the image

Source: M. Hebert

Non-linear filters

- Median filter
  - Median filter is edge preserving



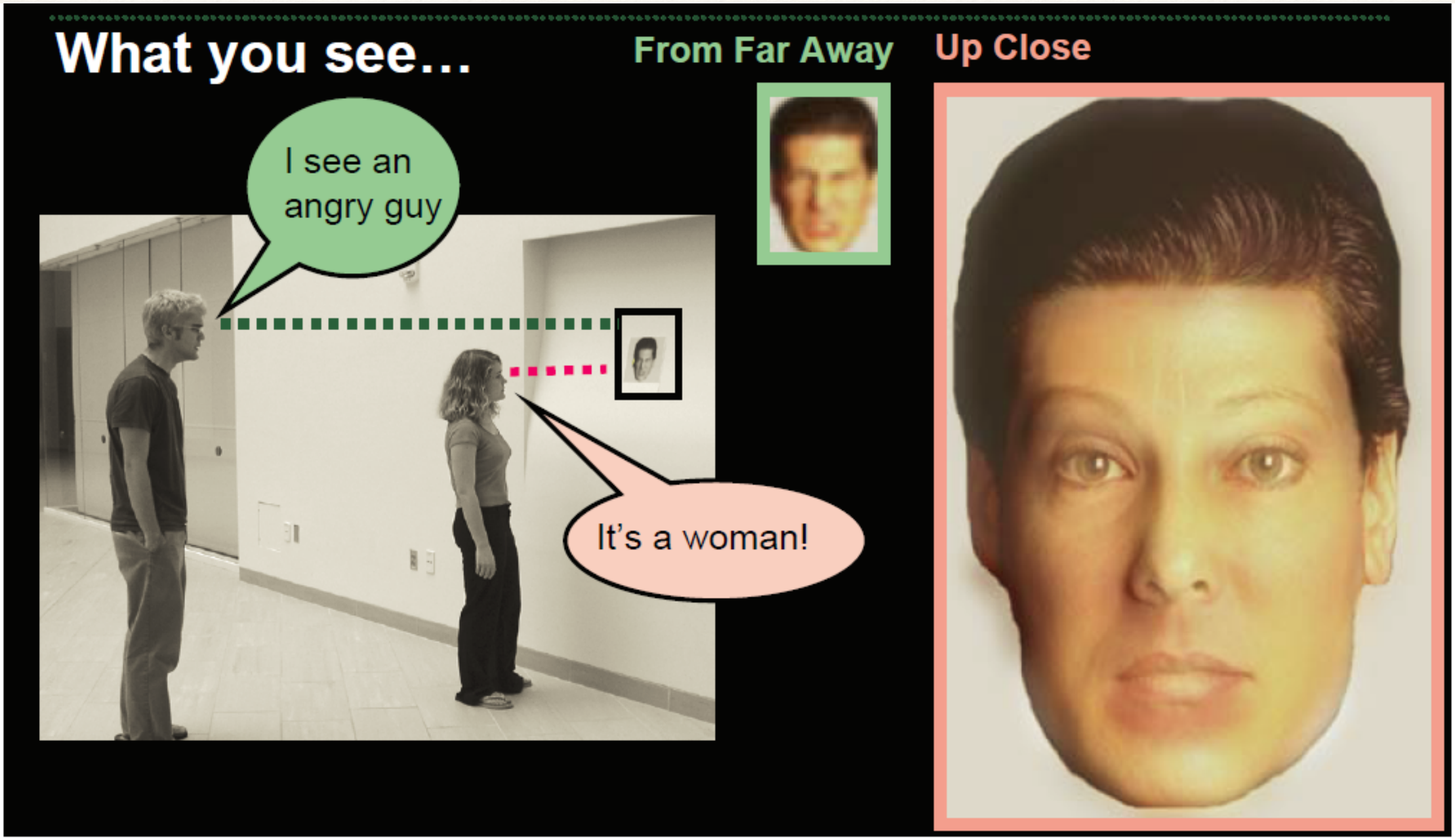
Source: K. Grauman

Frequency perspective

What can you see?

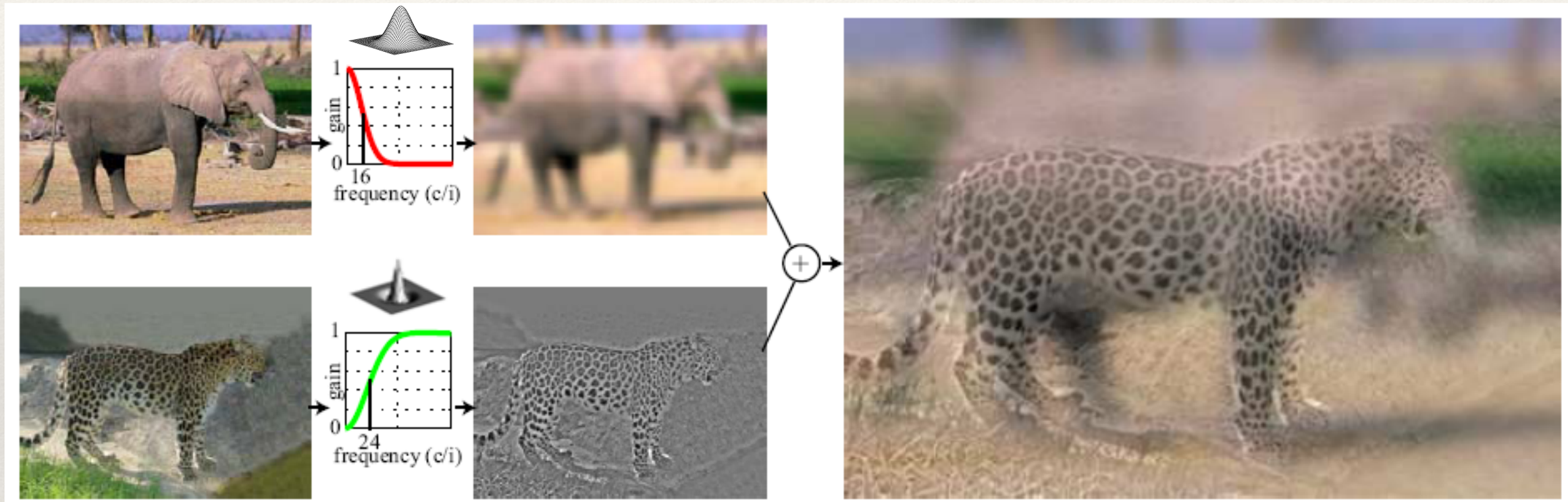
Salvador Dali (萨尔瓦多·达利),1976

Hybrid Images



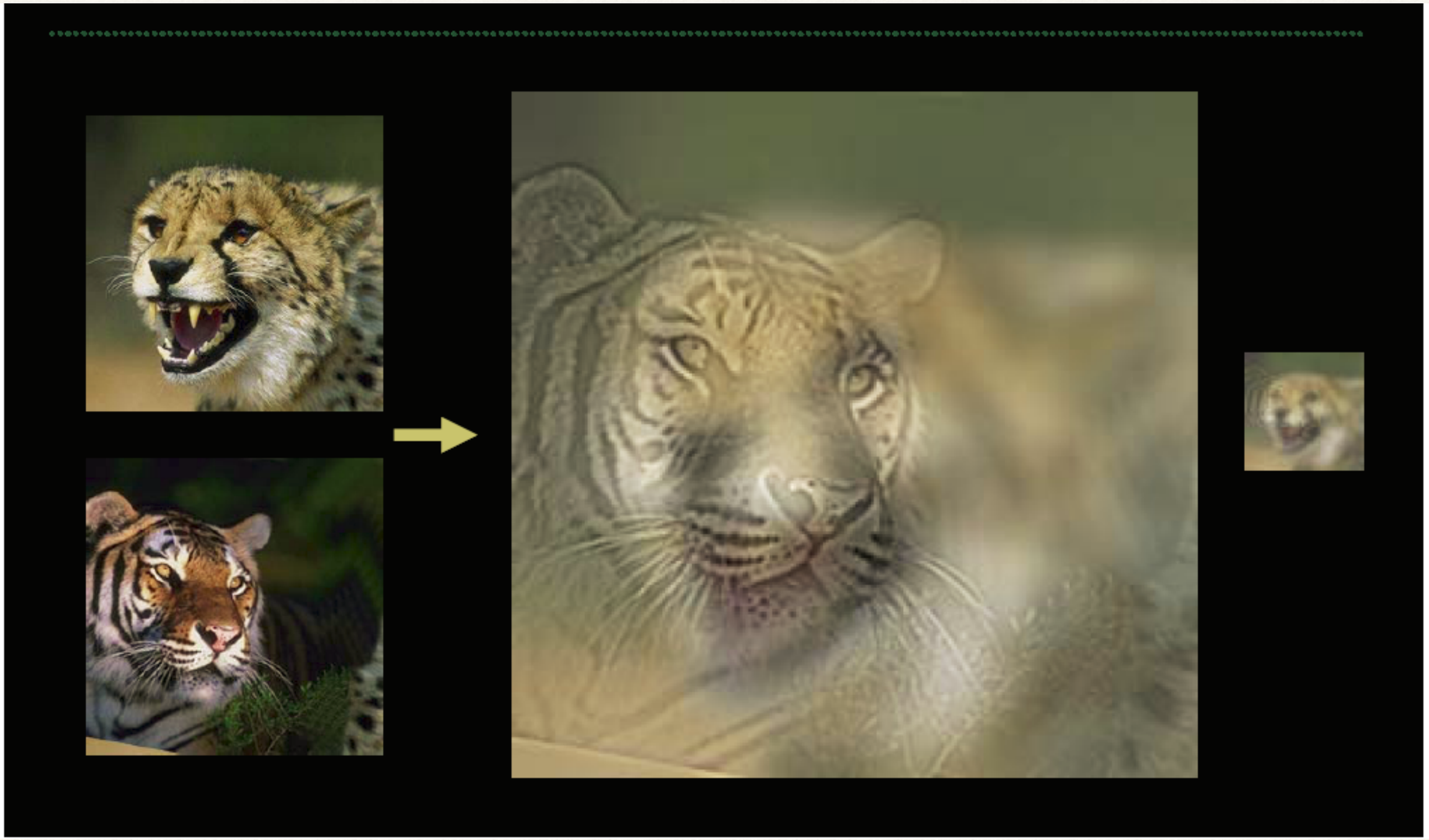
Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Hybrid Images



Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Hybrid Images

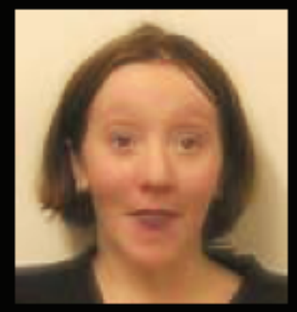


Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Changing expression

SIGGRAPH2006

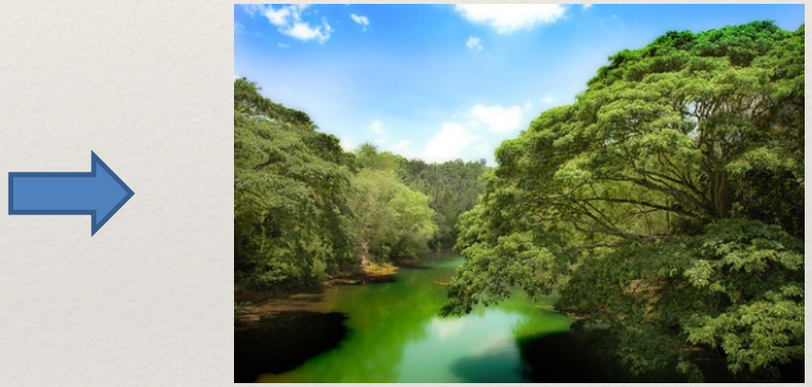
Sad ⟷ Surprised



Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Sampling

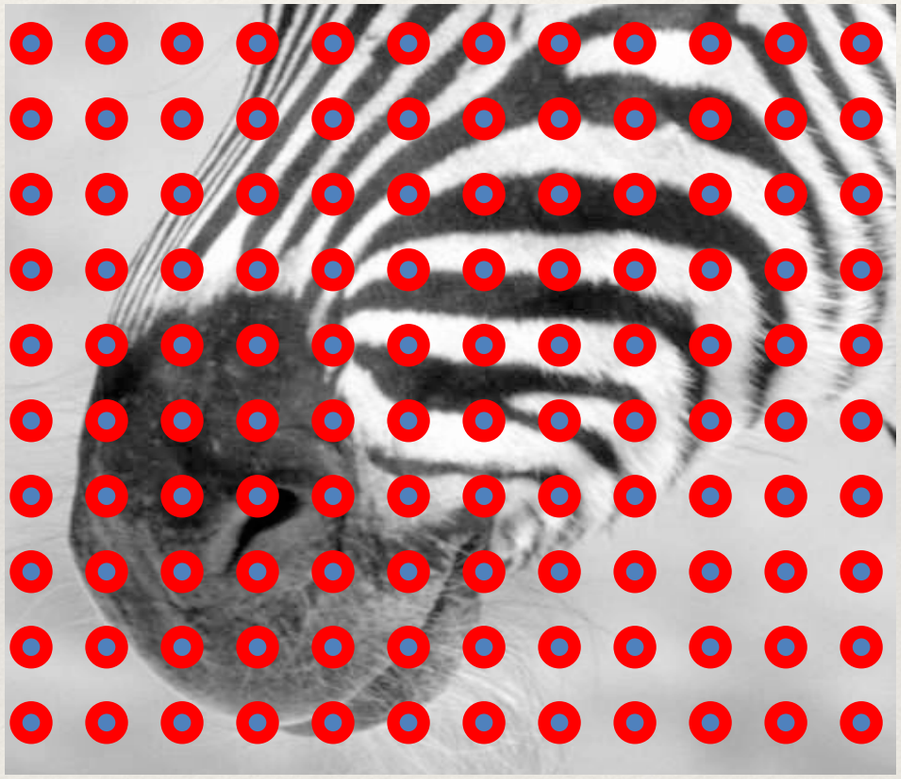
- Why does a lower resolution image still make sense to us? What do we lose?



Source: James Tompkin

Subsampling

- Subsampling by a factor of 2



Throw away every other row and column to create a 1/2 size image

Source: James Tompkin

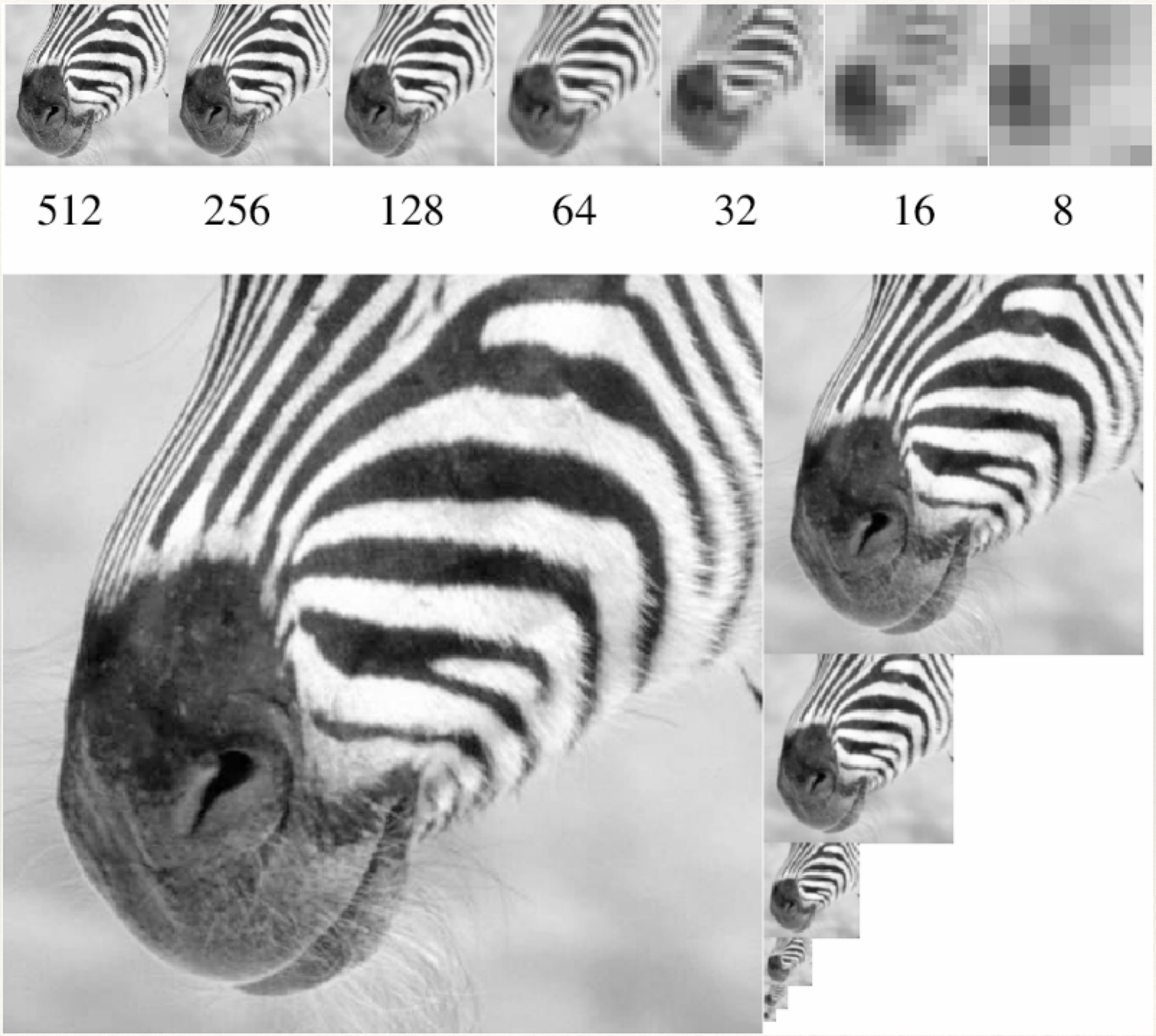
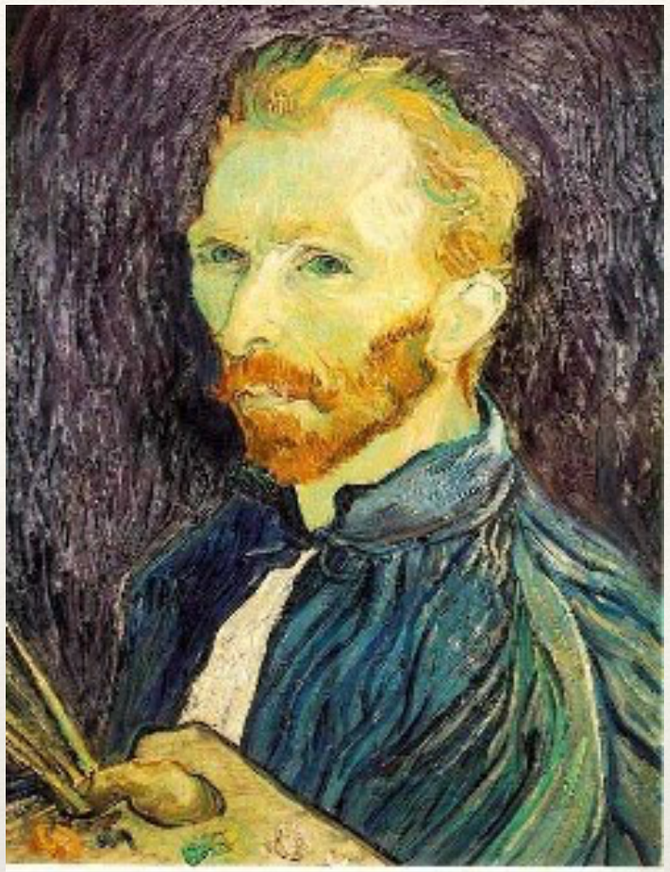


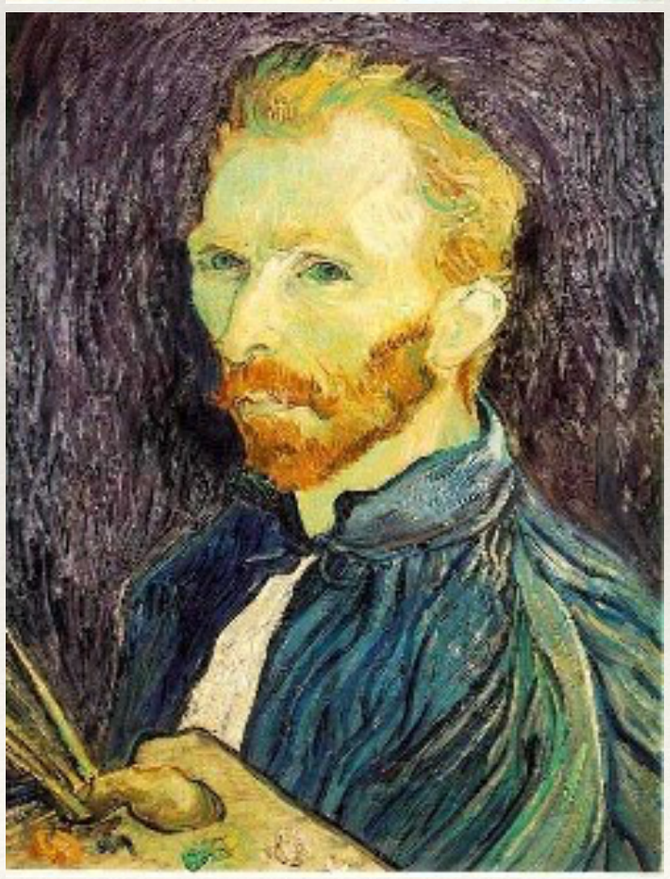
Figure from David Forsyth

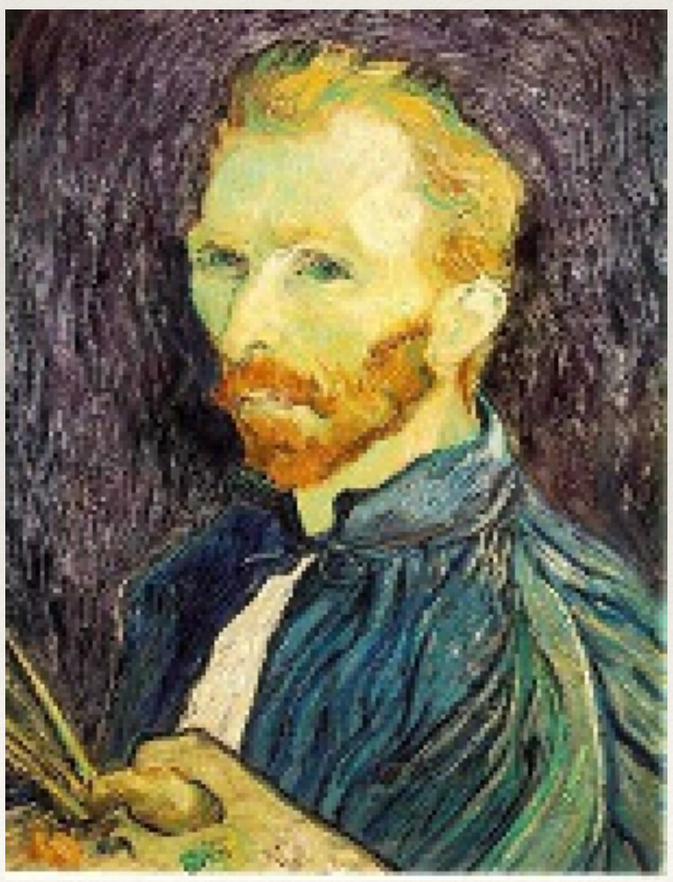
1/2

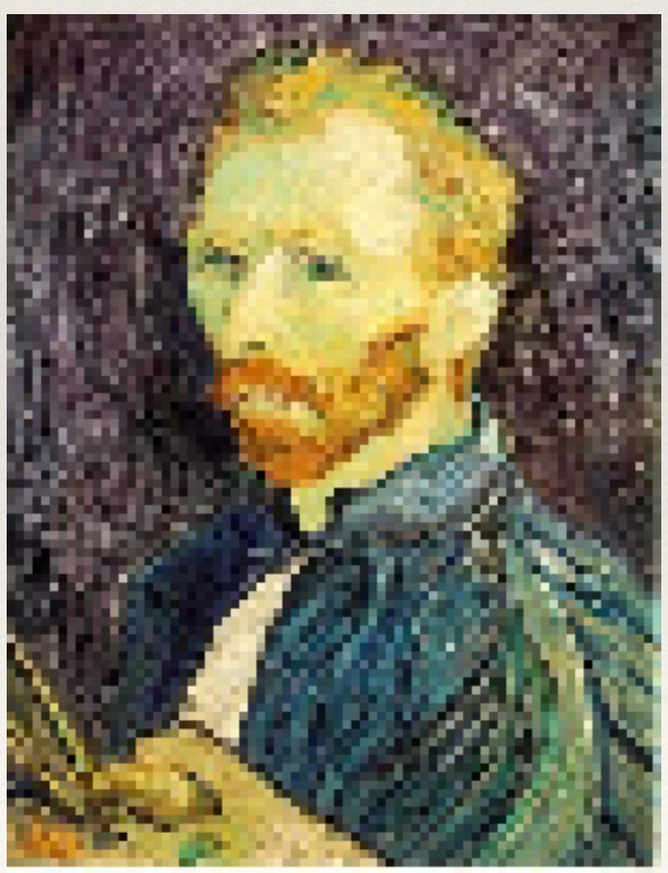


1/4 2x subsample

1/8 4x subsample

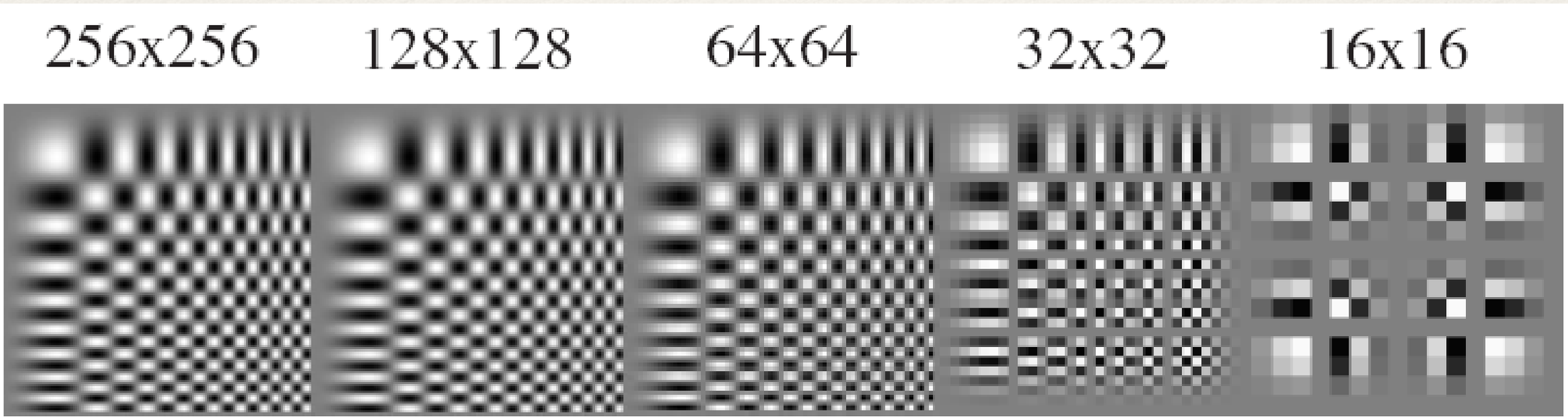






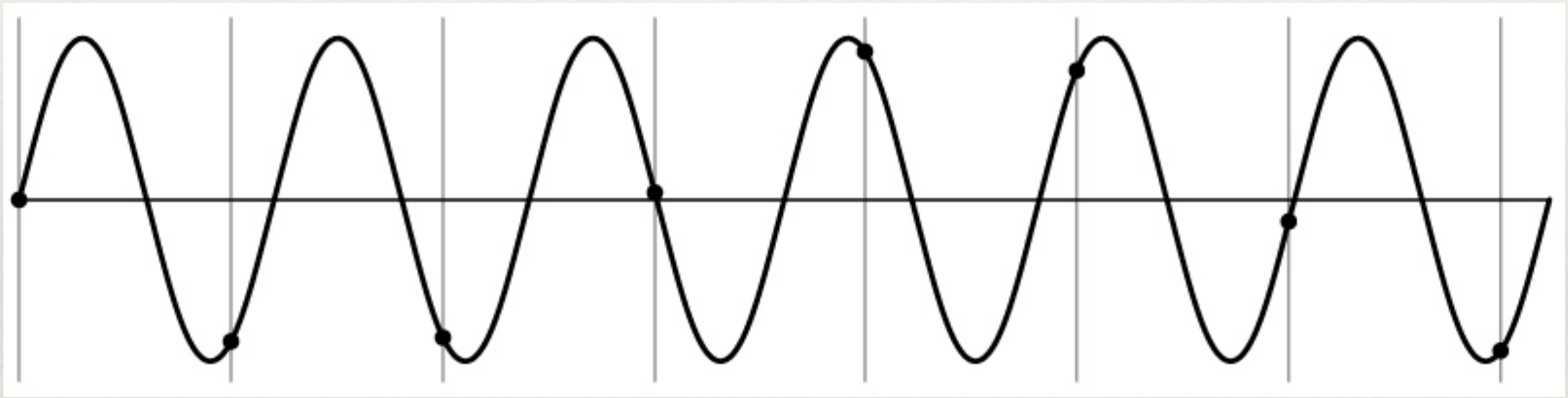
Steve Seitz

Sampling and aliasing



Sampling and aliasing

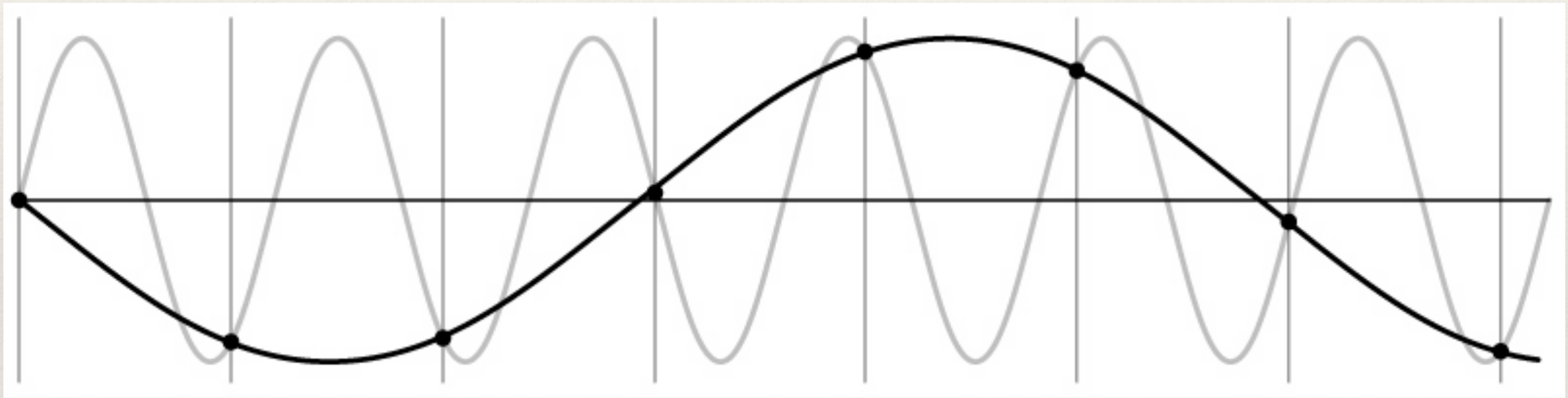
1D example (sinewave):



Source: S. Marschner

Sampling and aliasing

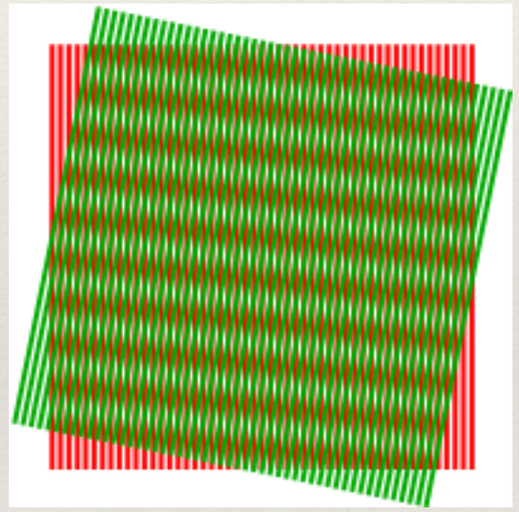
1D example (sinewave):



Source: S. Marschner

Aliasing problem

- Sub-sampling may be dangerous….
- Characteristic errors may appear:
  - "car wheels rolling the wrong way in movies
  - "checkerboards disintegrate in ray tracing"
  - "striped shirts look funny on color television"
    - Moiré patterns



Source: James Tompkin

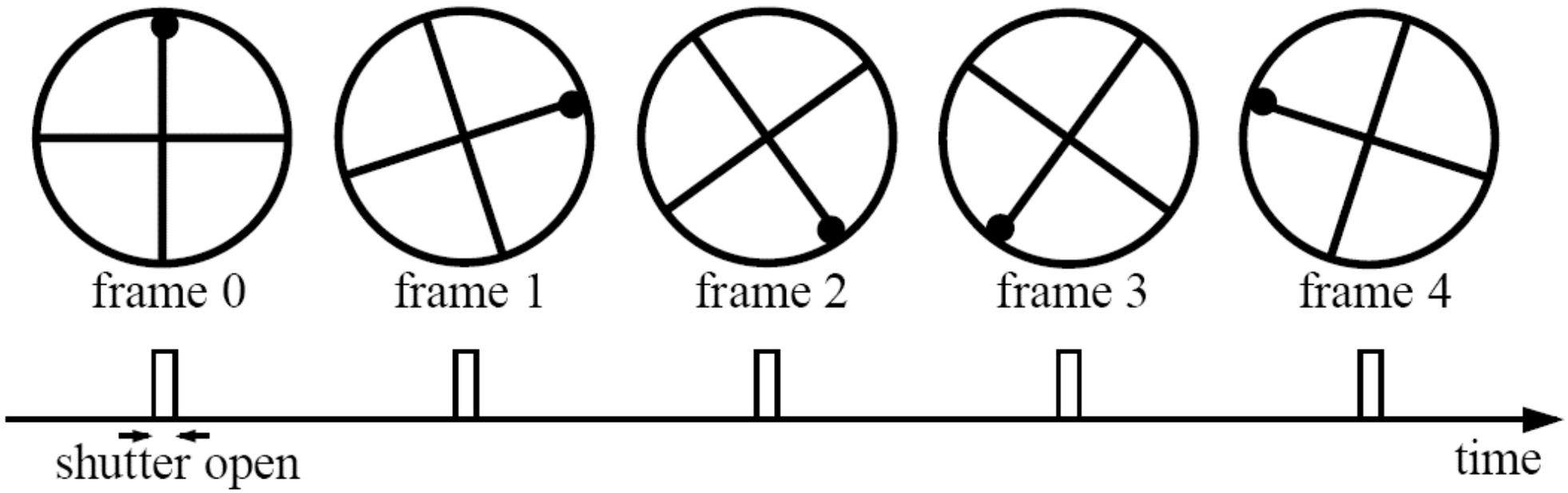
Aliasing and Moiré patterns

- Moire pattern, known as the moire effect
- Useful in metrology, strain analysis or even document authentication and anti-counterfeiting. In other situations moire patterns may be unwanted.
- This visual perception occurs when a fine pattern on your subject meshes with the pattern on the imaging chip of your camera. Difference in frequency.

Aliasing in video

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):

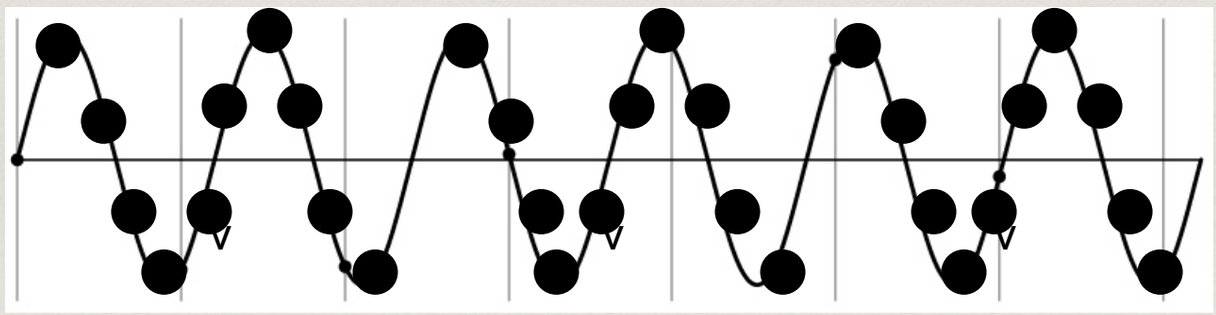


Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

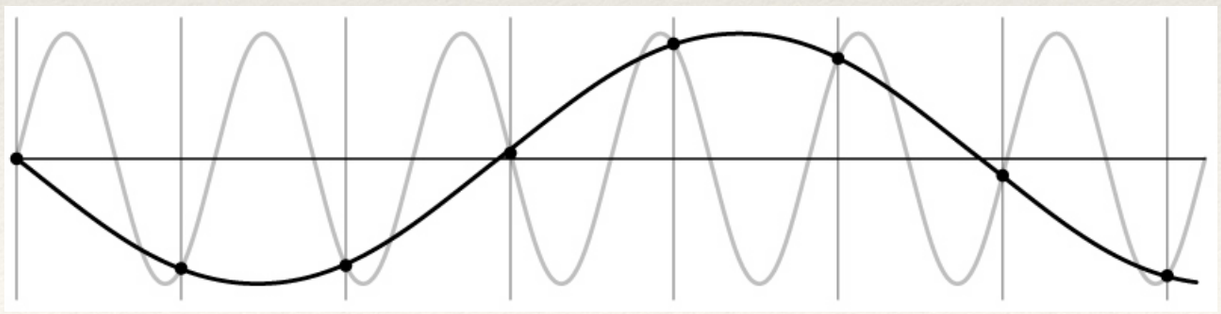
Slide by Steve Seitz

Nyquist-Shannon Sampling Theorem

- When sampling a signal at discrete intervals, the sampling frequency must be $\geq 2 \times f_{max}$
- f_{max} = max frequency of the input signal
- This will allows to reconstruct the original perfectly from the sampled version



good

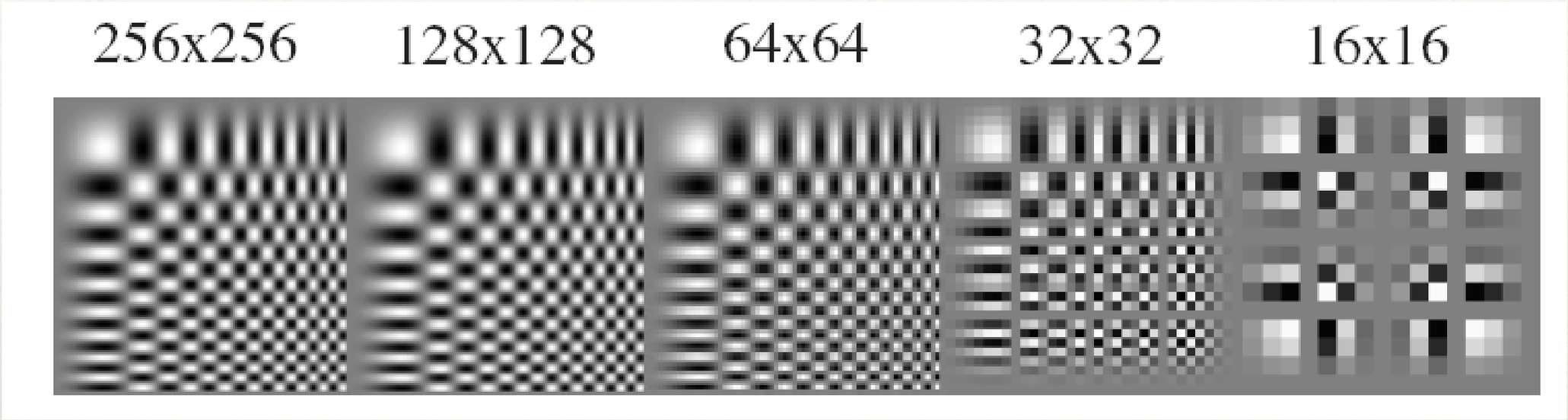


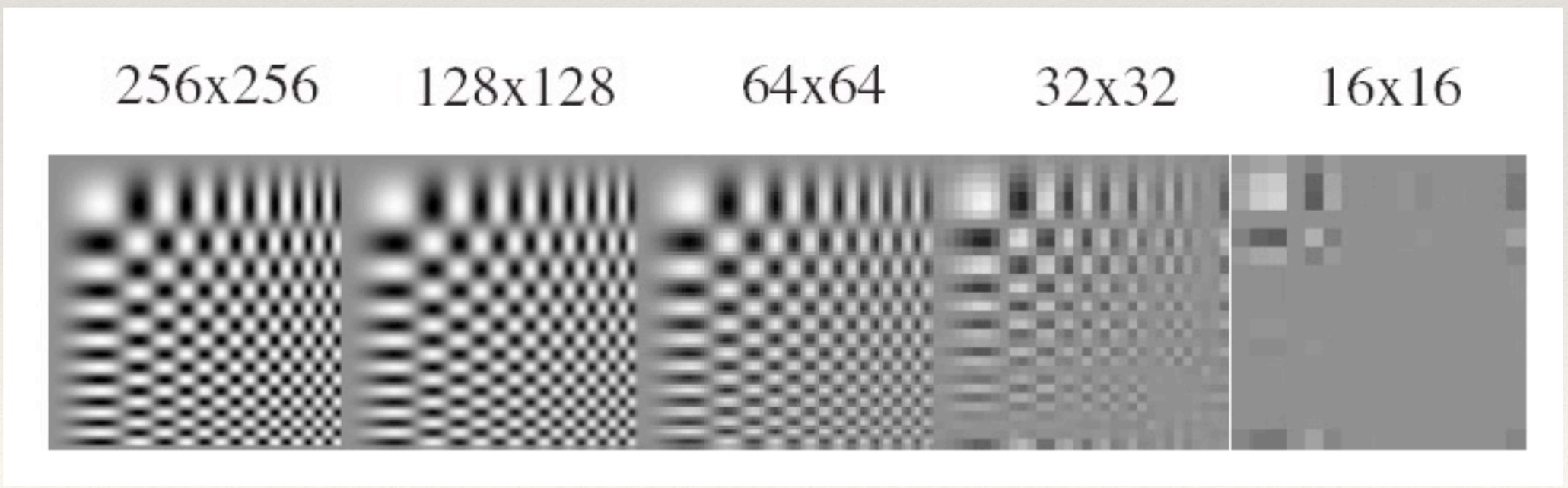
bad

How to fix aliasing?

- Sample more often (better sensor)
- Get rid of all frequencies that are greater than half the new sampling frequency
- Will lose information
- But it's better than aliasing
- Apply a smoothing (*low pass*) filter

Anti-aliasing





Forsyth and Ponce 2002

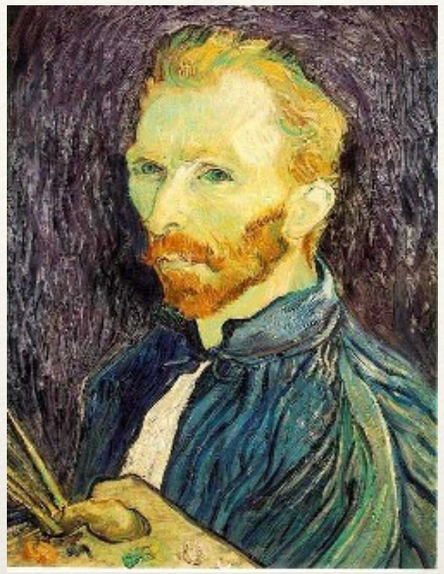
Algorithm for downsampling by factor of 2

- Start with image(h, w)
- Apply low-pass filter:

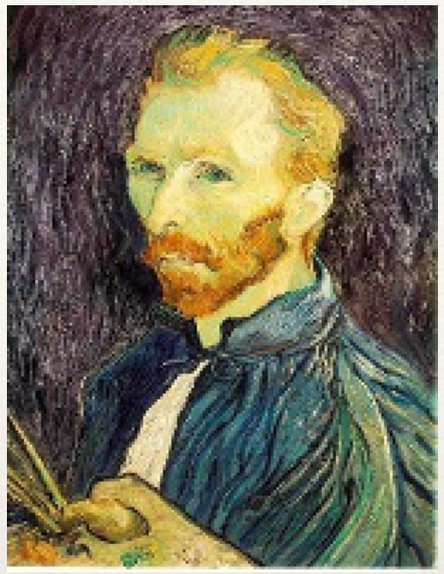
  im_blur = imfilter(image, fspecial('gaussian', 7, 1))

- Sample every other pixel

  im_small = im_blur(1:2:end, 1:2:end)

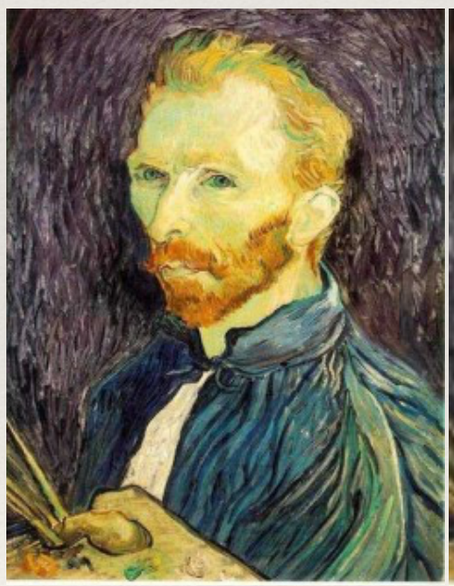


1/2



1/4 (2x subsample)

1/8 (4x subsample)



Gaussian 1/2

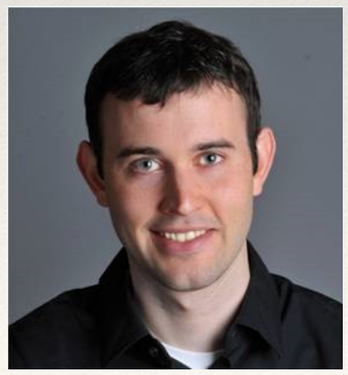
G 1/4

G 1/8

Subsampling with Gaussian pre-filtering

Human spatial frequency sensitivity

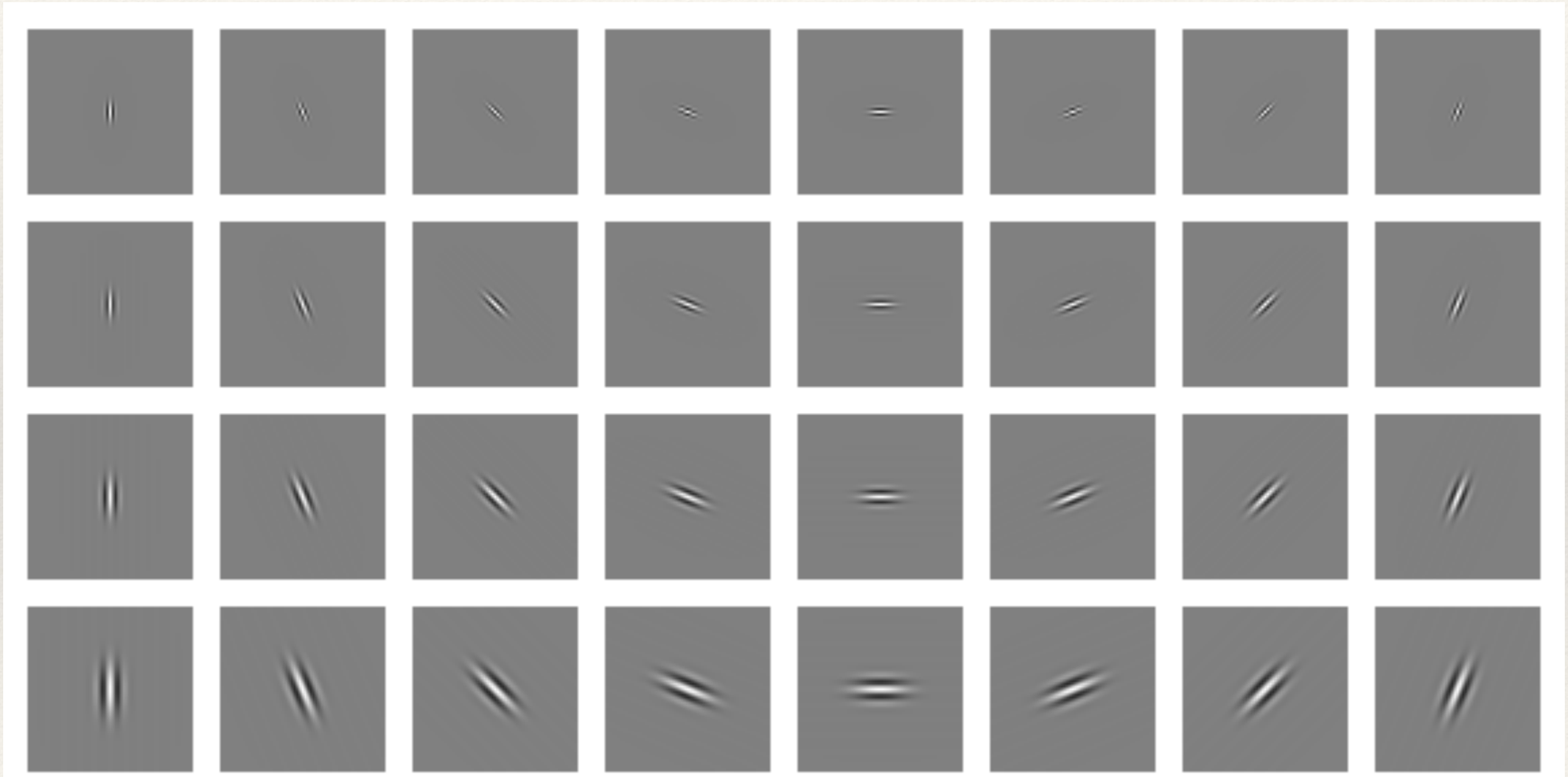
Why do we get different, distance-dependent interpretations of hybrid images?



Hays

Clues from Human Perception

Early processing in humans filters for orientations and scales of frequency.

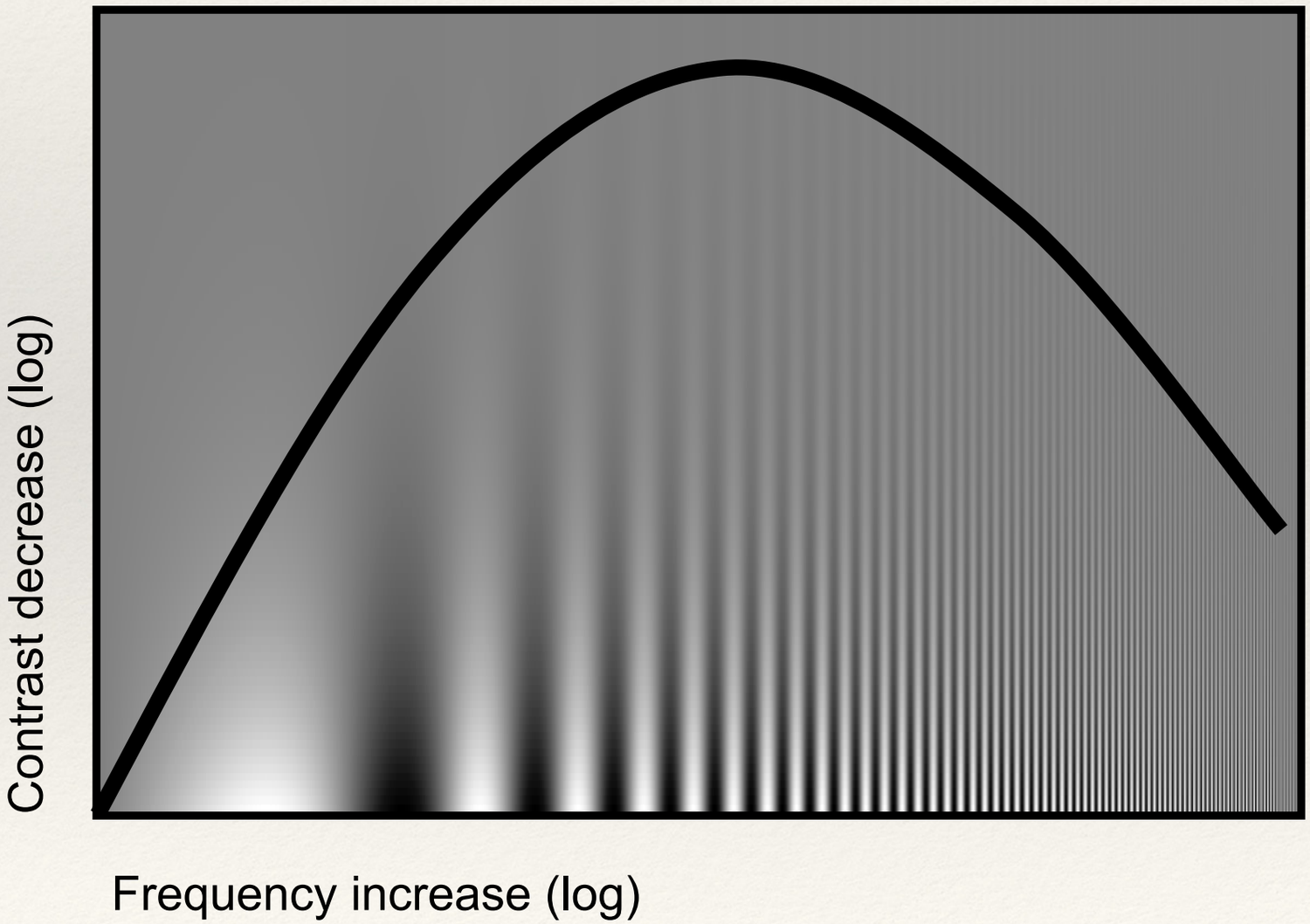


Early Visual Processing: Multi-scale edge and blob filters

Source: James Tompkin

Campbell-Robson contrast sensitivity curve[v]

Perceptual cues in the mid-high frequencies dominate perception.



Source: James Tompkin

Salvador Dali

"Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln", 1976

Fourier Transforms

- The tool manipulating frequency. Sampling, De-noise, filtering…
- Linear, space invariant operations are just diagonal operations in the frequency domain
- Linear convolution is multiplication in the frequency domain

Fourier series

- A bold idea (1807):
  - **Any** univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.

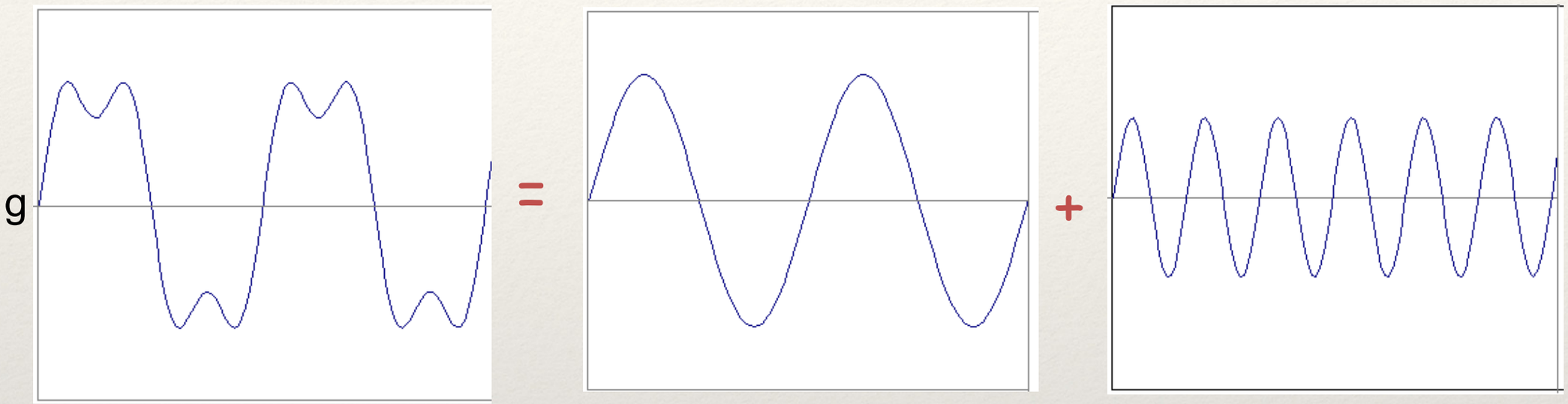
$$\sum_{n=1}^{\infty} (a_n \cos(nt) + b_n \sin(nt))$$

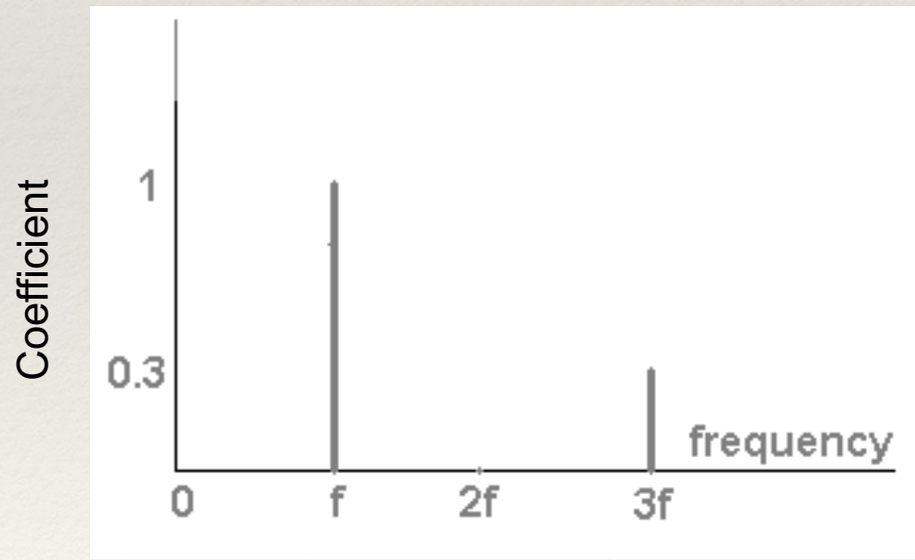
- Add enough of them to get any signal *g(t)* you want!

Jean Baptiste Joseph Fourier (1768-1830)

$$t = [0,2], f = 1$$

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

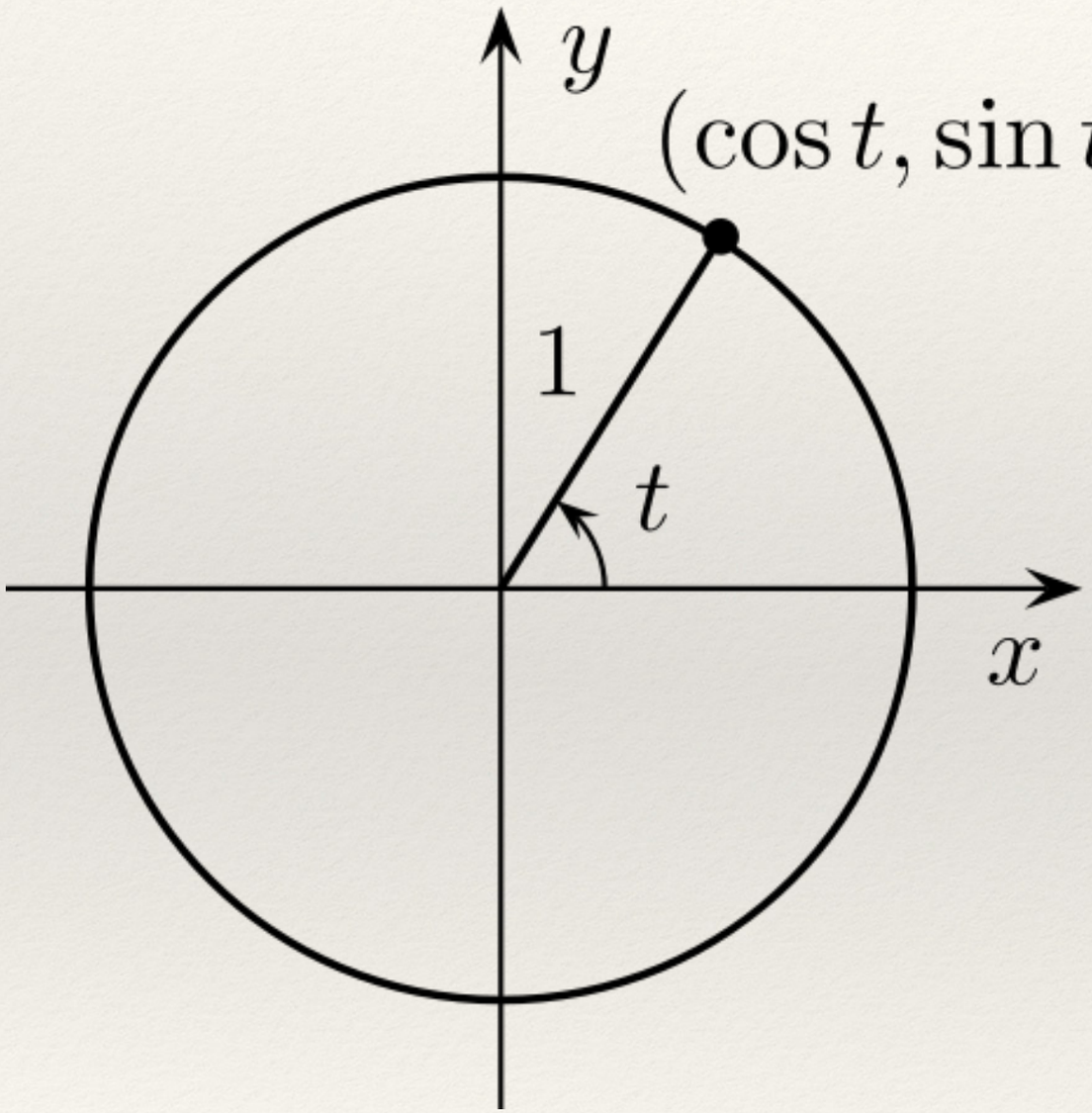


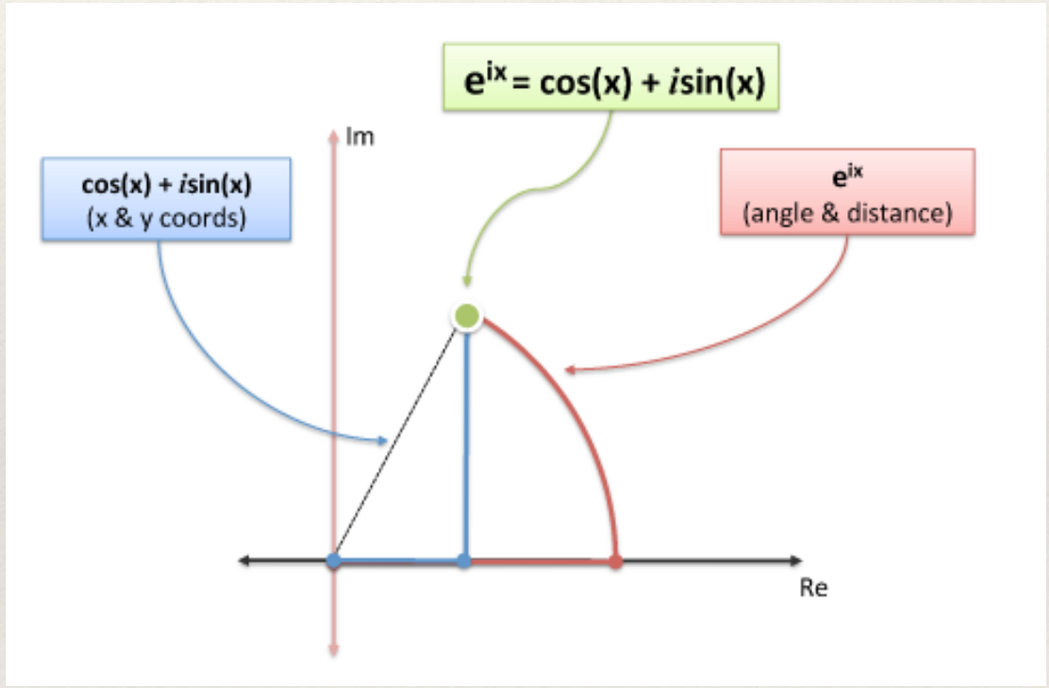


Slides: Efros

Wikipedia-Fourier transform

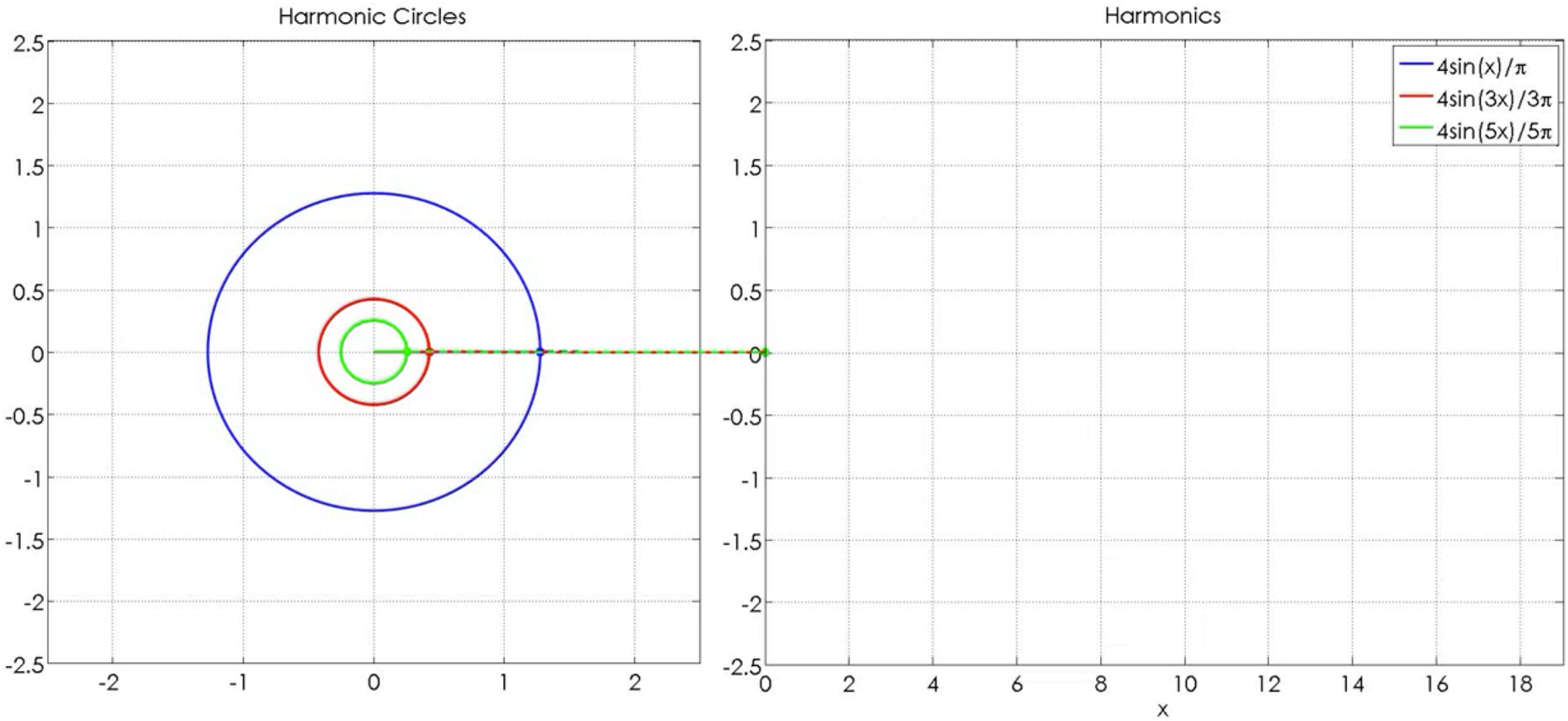
Sine/cosine and circle





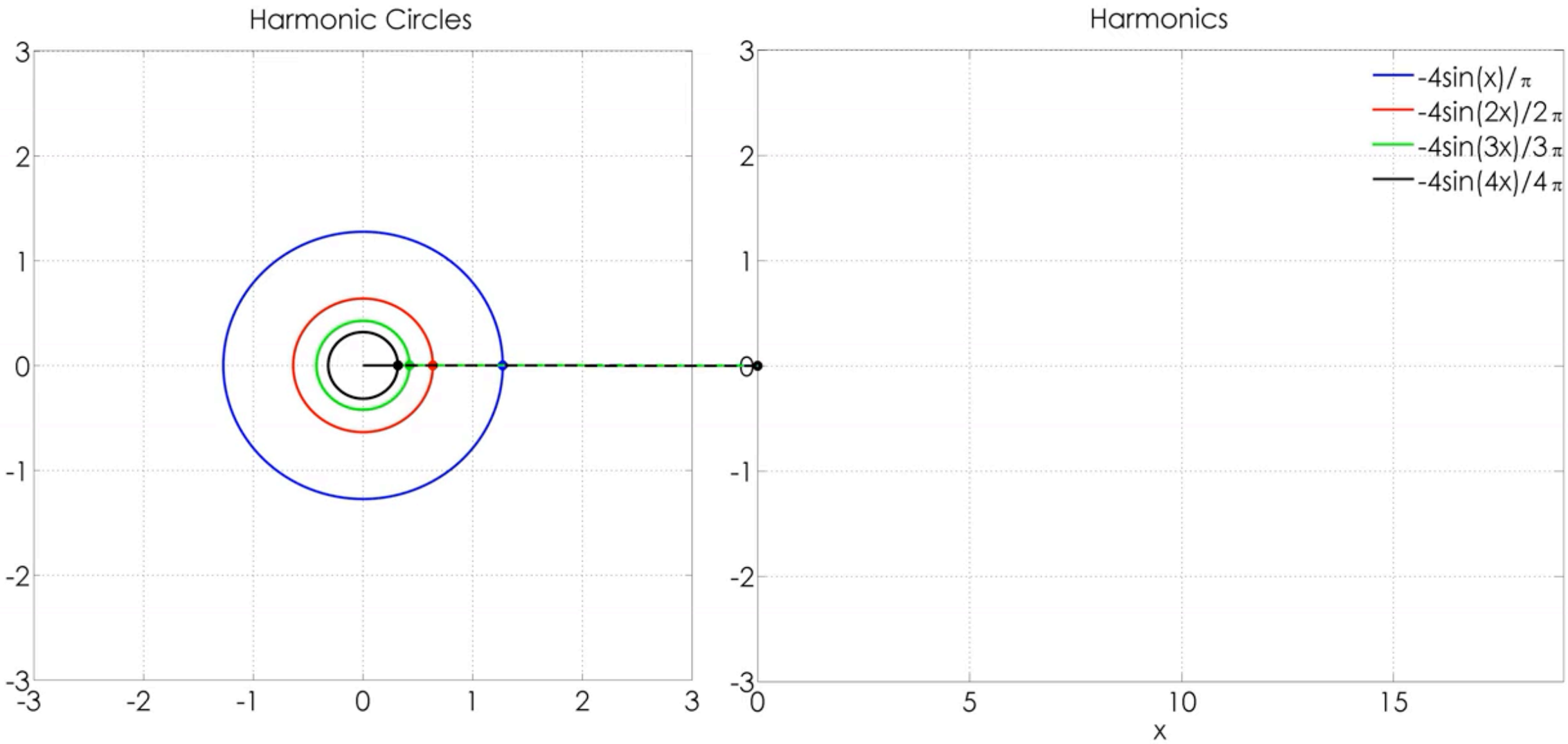
Wikipedia - Unit circle

Square wave (approx.)



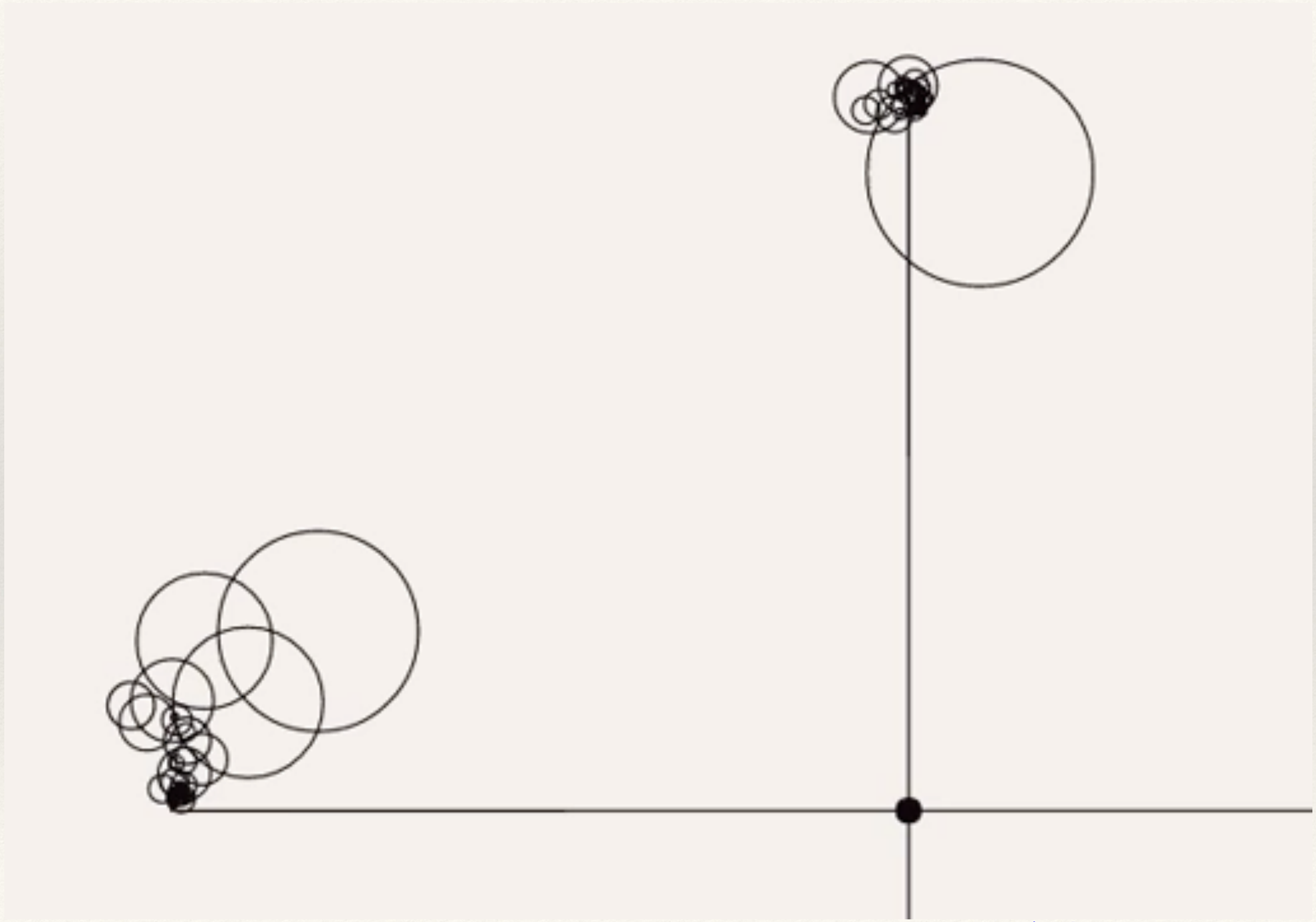
Mehmet E. Yavuz

Square wave (approx.)



Mehmet E. Yavuz

One series in each of x and y



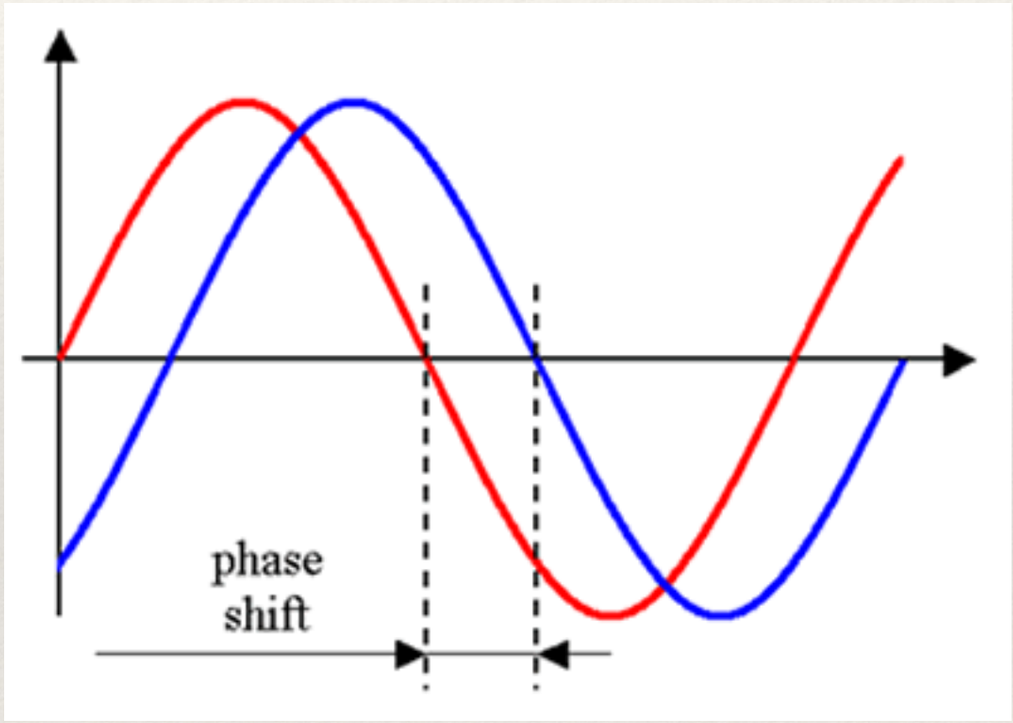
Generative Art, じゃがりきん, Video, 2018

Amplitude-phase form

- Add phase term to shift cos values into sin values

$$\sum_{n=1}^{N} (a_n \sin(nx + \emptyset_n))$$

Phase



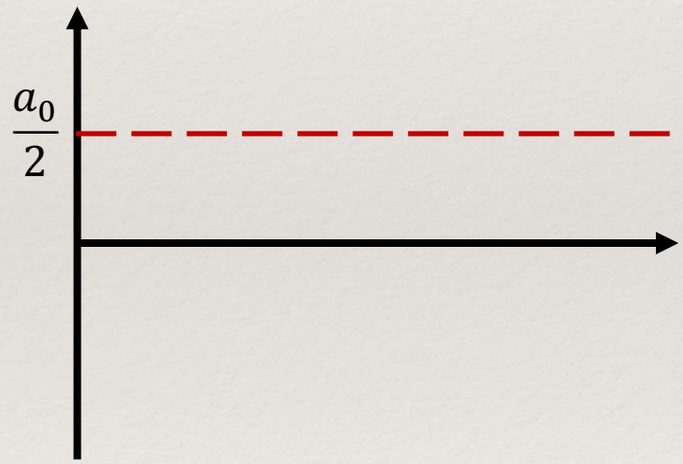
[Peppergrower; Wikipedia]

Amplitude-phase form

- Add component of infinite frequency
  - = mean of signal over period
  - = value around which signal fluctuates

$$\frac{a_0}{2} + \sum_{n=1}^{N} (a_n \sin(nx + \emptyset_n))$$

Average of signal over period

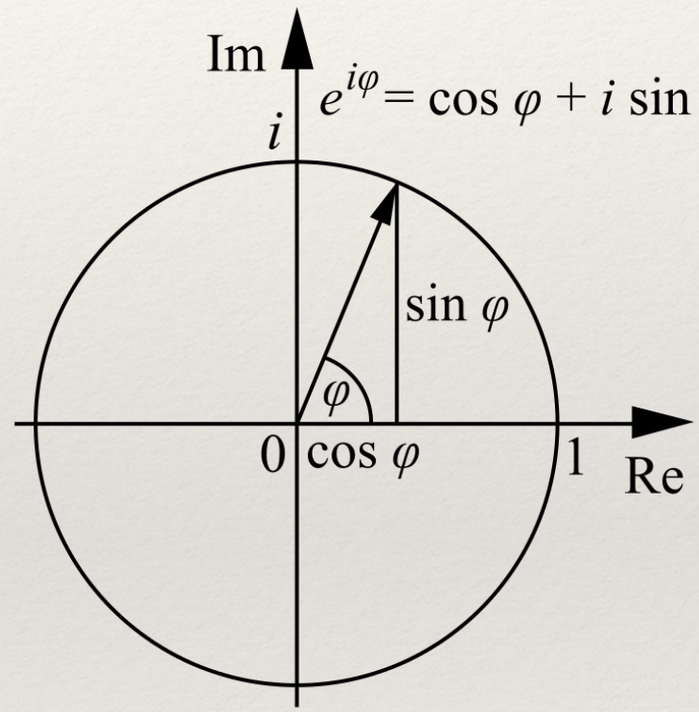


Amplitude-phase form

$$s_N(x) \triangleq \frac{A_0}{2} + \sum_{n=1}^{N} A_n \cdot \sin\left(\frac{2\pi n x}{P} + \phi_n\right), \quad \text{for integer } N \geq 1. \quad \textit{(Eq.1)}$$

$$s_N(x) = \overbrace{a_0}^{A_0}/2 + \sum_{n=1}^{N} \left(\overbrace{a_n}^{A_n \sin(\phi_n)} \cos\left(\frac{2\pi n x}{P}\right) + \overbrace{b_n}^{A_n \cos(\phi_n)} \sin\left(\frac{2\pi n x}{P}\right) \right), \quad \textit{(Eq.2)}$$

$$s_N(x) = \sum_{n=-N}^{N} c_n \cdot e^{i\frac{2\pi n x}{P}}, \quad \textit{(Eq.3)}$$



[Peppergrower; Wikipedia]

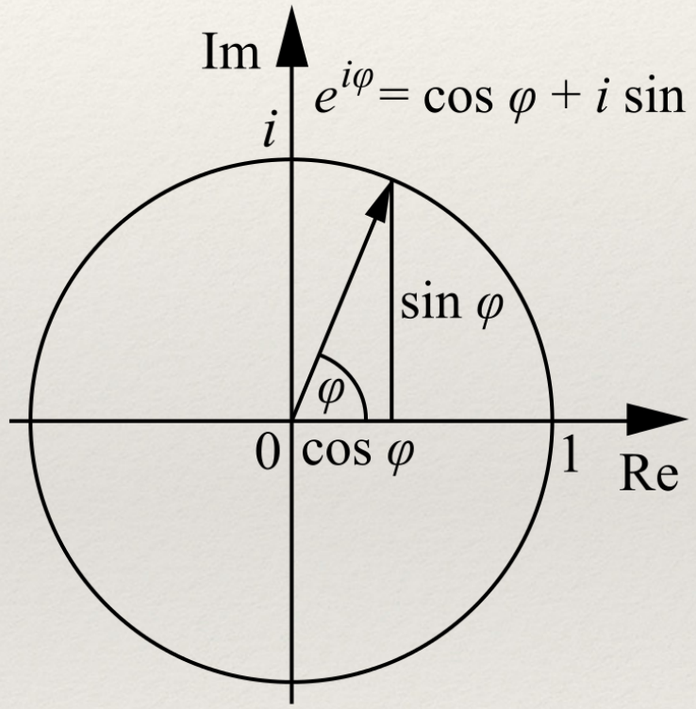
Fourier Transform

- Stores the amplitude and phase at each frequency:
  - For mathematical convenience, this is often notated in terms of real and complex numbers
  - Related by Euler's formula
- Amplitude encodes how much signal there is at a particular frequency

$$A = \pm\sqrt{\mathrm{Re}(\varphi)^2 + \mathrm{Im}(\varphi)^2}$$

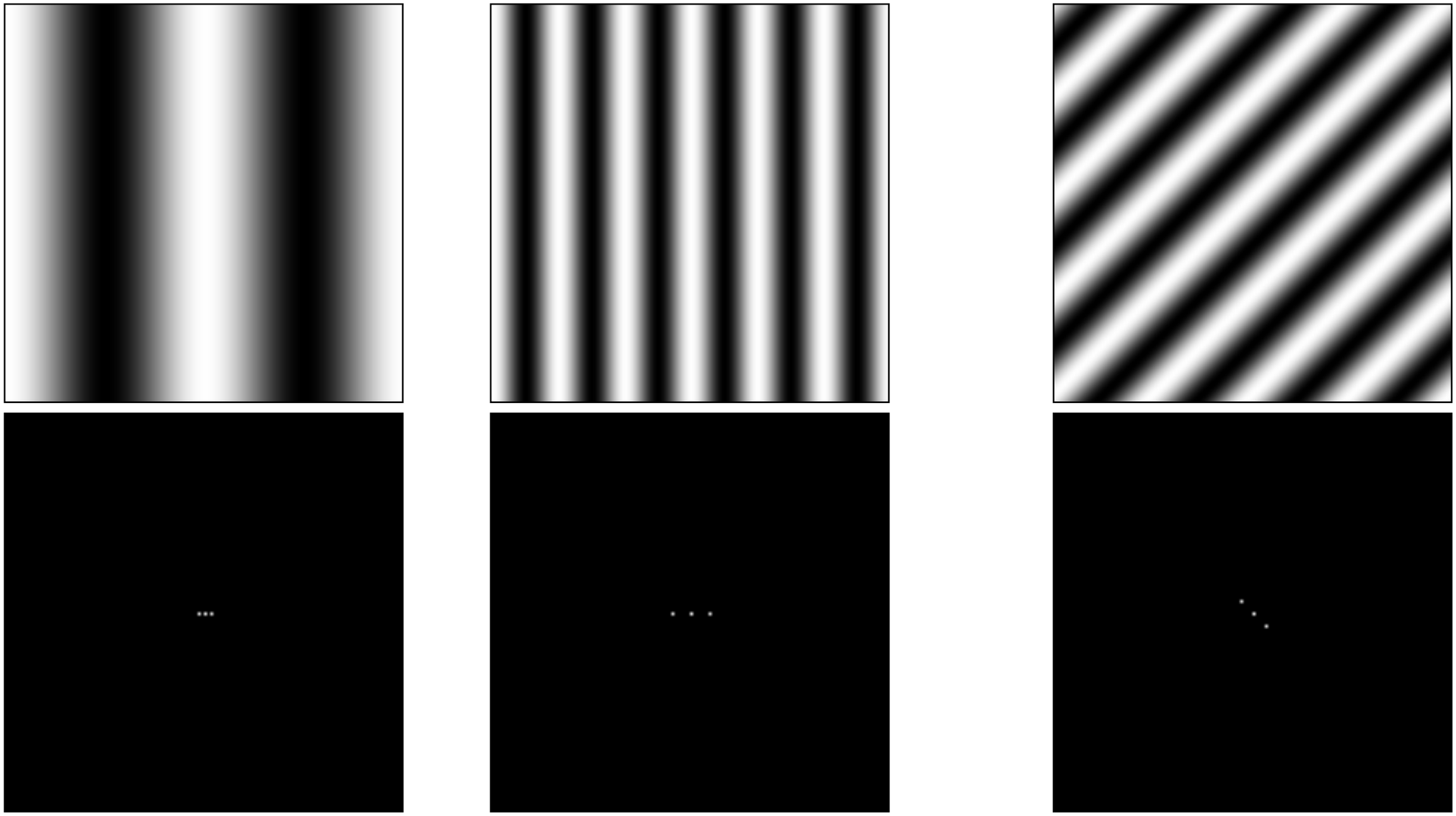
- Phase encodes spatial information (indirectly)

$$\phi = \tan^{-1}\frac{\mathrm{Im}(\varphi)}{\mathrm{Re}(\varphi)}$$



Fourier analysis in images

Spatial domain images

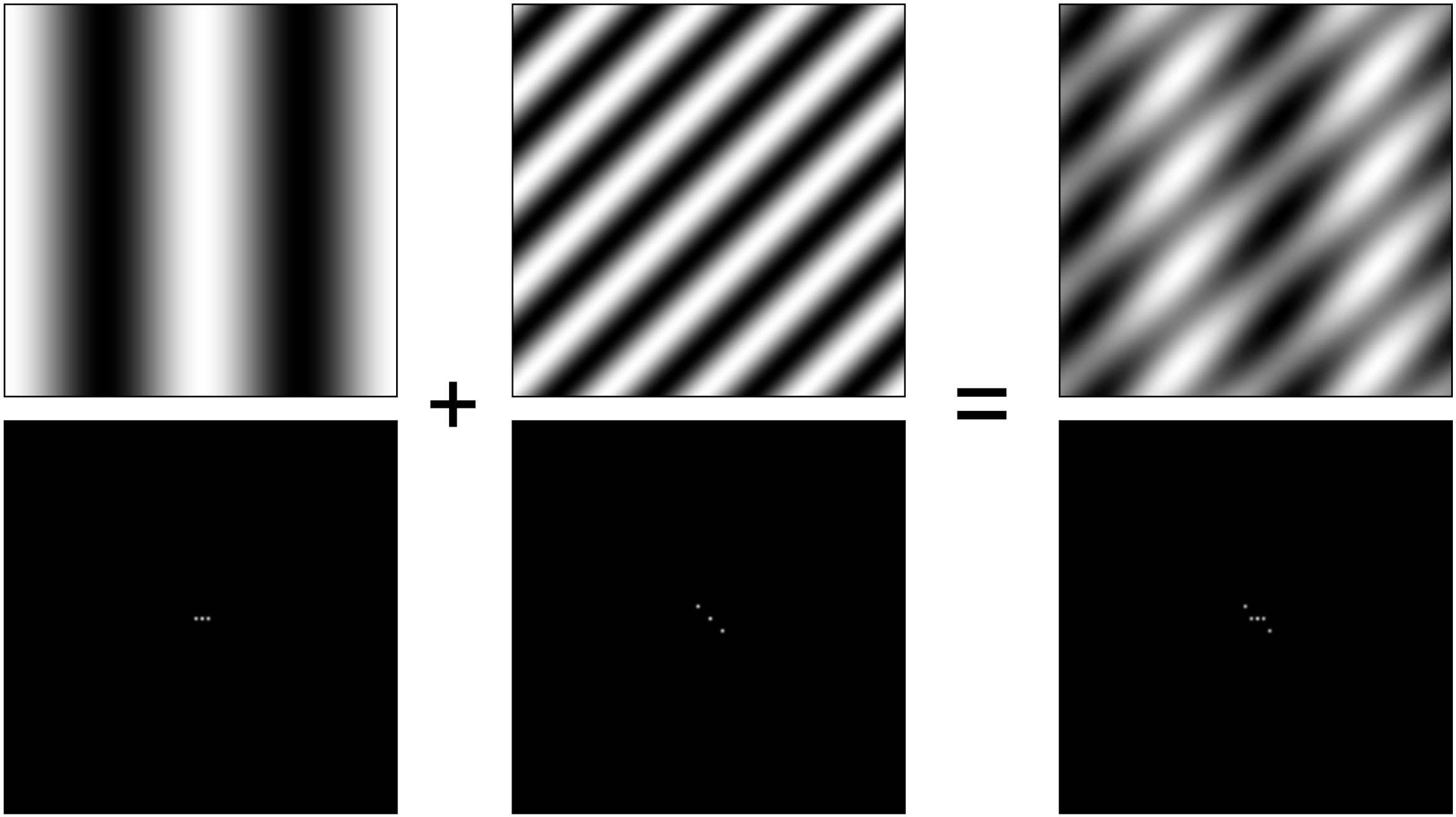


Fourier decomposition amplitude images

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

Signals can be composed

Spatial domain images



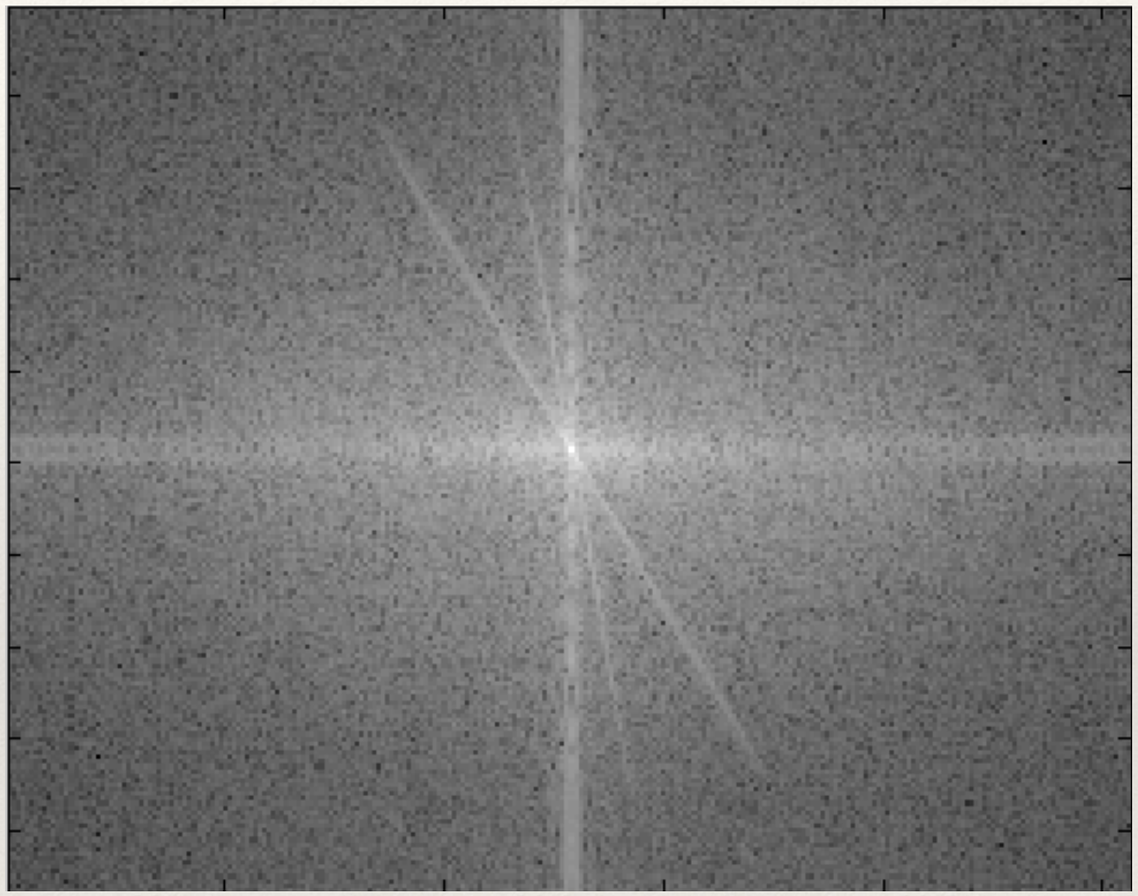
Fourier decomposition amplitude images

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

Natural images

Natural image

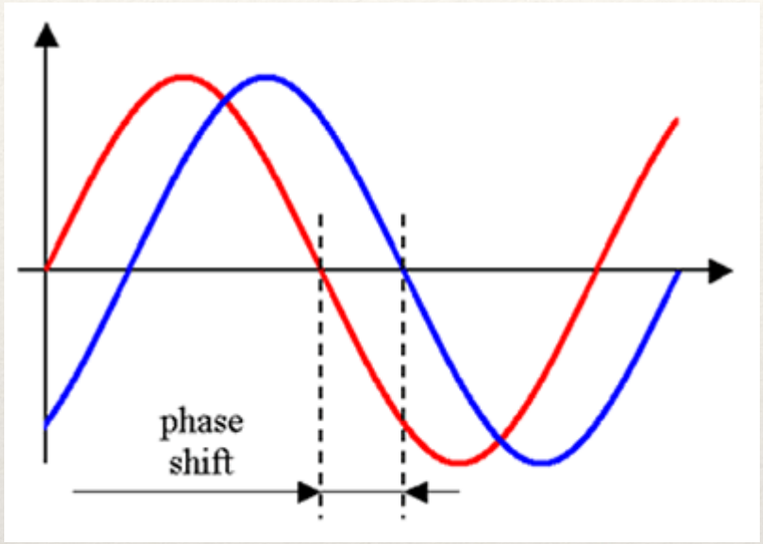
Fourier decomposition amplitude image



① What does it mean to be at pixel x,y?
② What does it mean to be more or less bright in the Fourier decomposition image?

Source: James Tompkin

Amplitude / Phase



- Amplitude tells you “how much”
- Phase tells you “where”
- Translate the image?
  - Amplitude unchanged
  - Adds a constant to the phase.

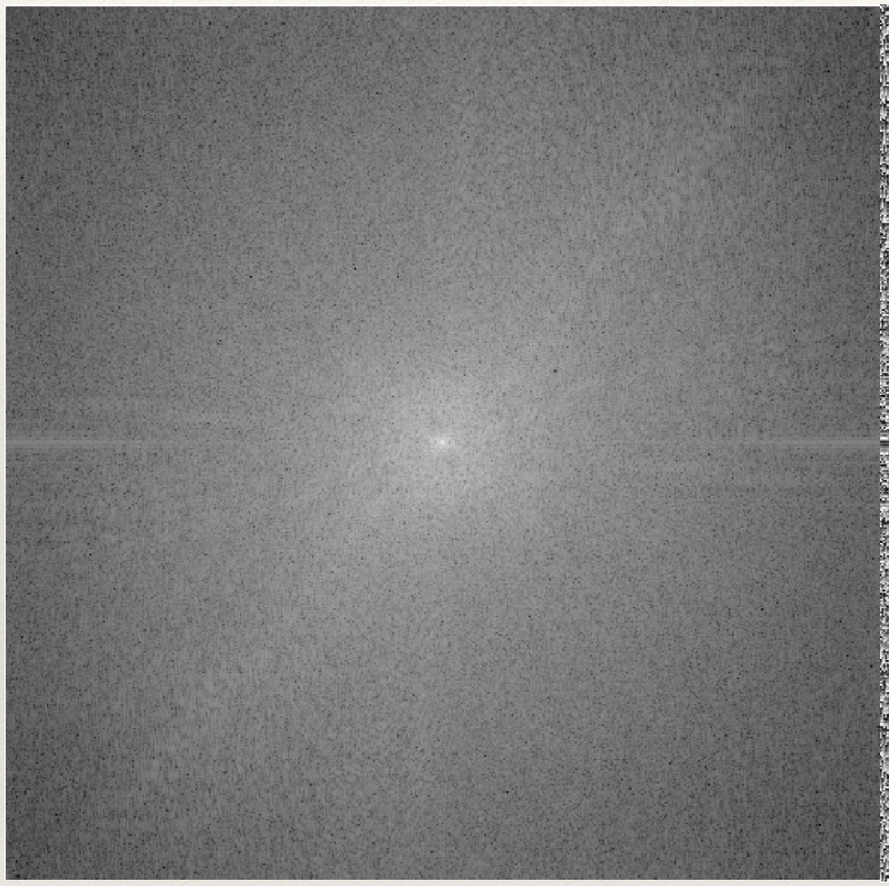
[Peppergrower; Wikipedia]

Amplitude / Phase

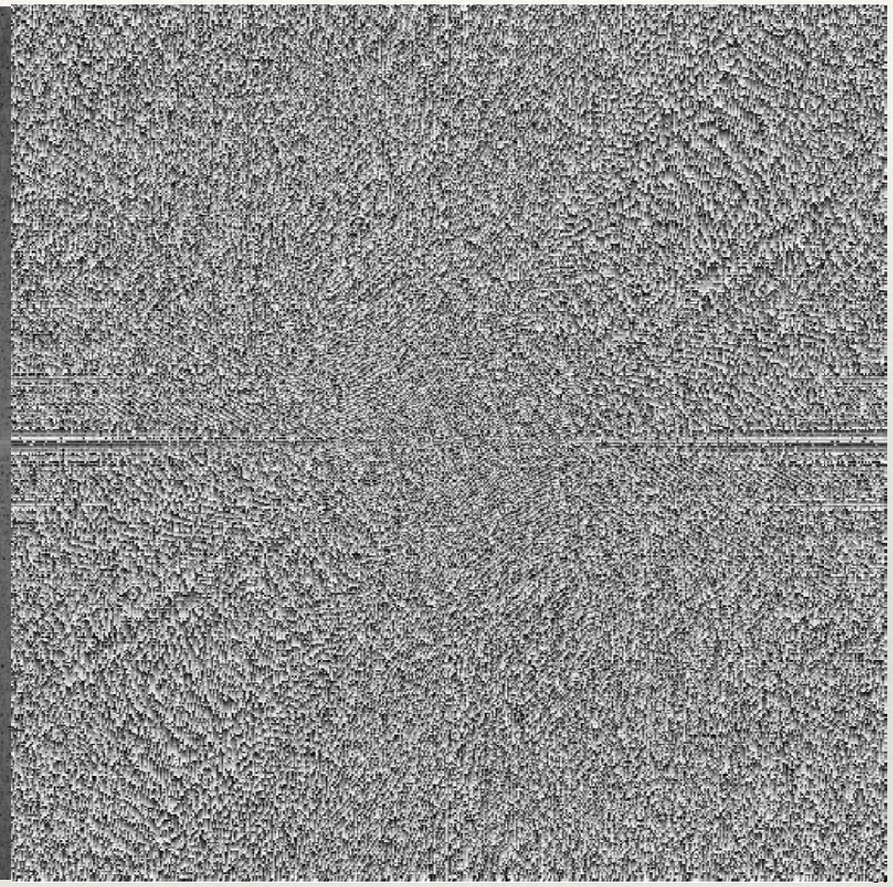
Efros

Amplitude / Phase

Amplitude



Phase



Efros

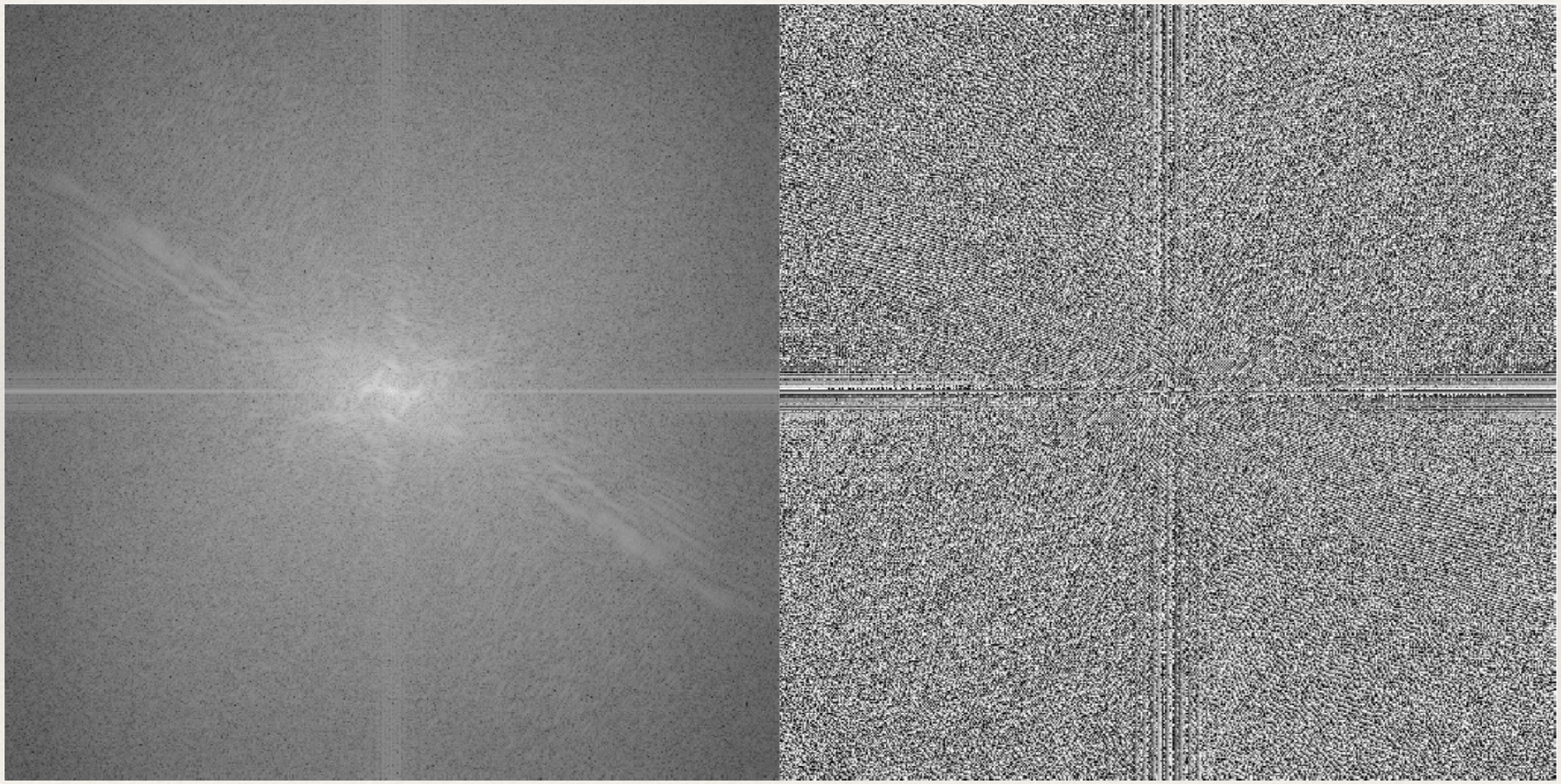
Amplitude / Phase

Efros

Amplitude / Phase

Amplitude

Phase

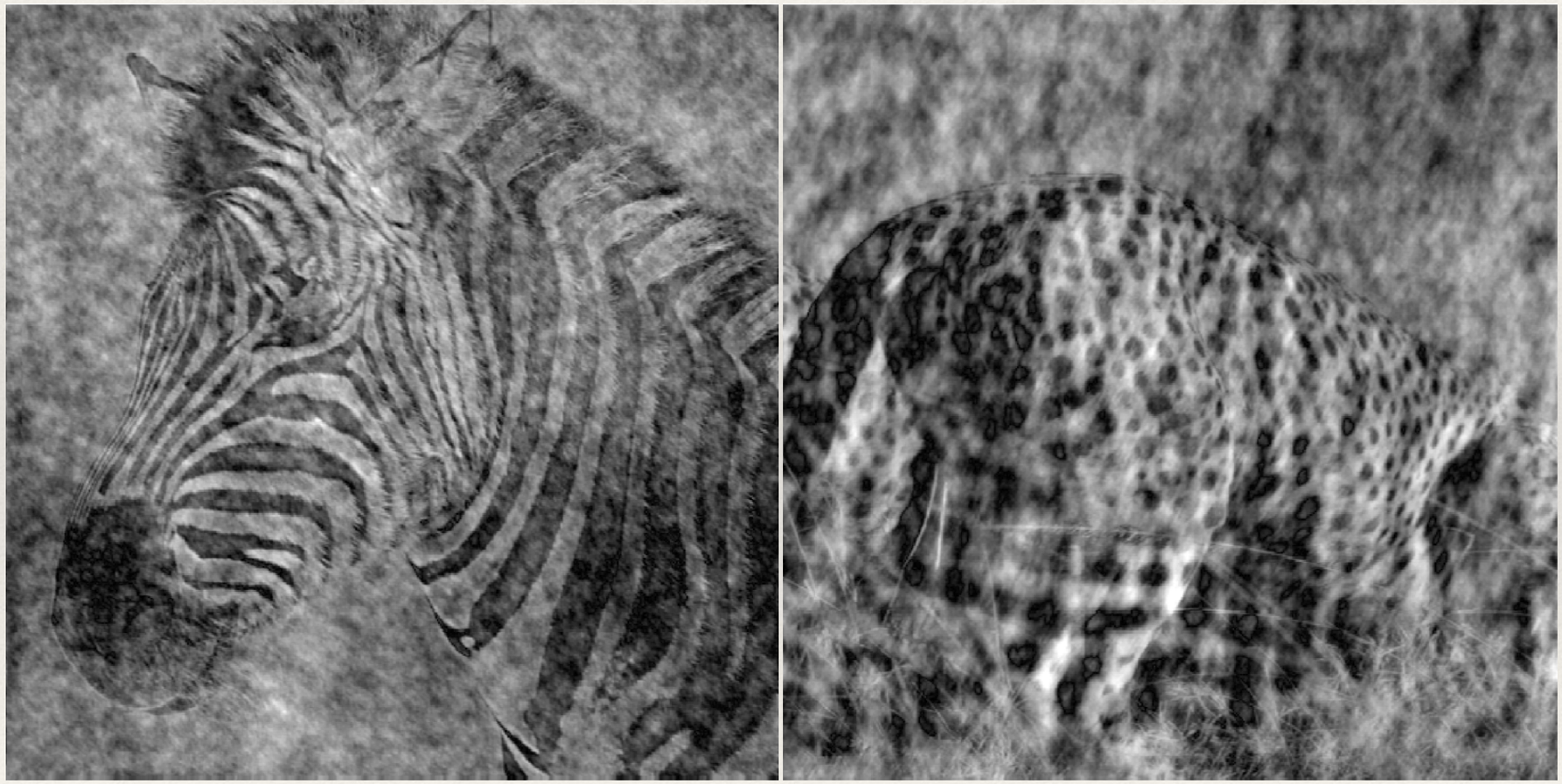


Amplitude / Phase

Cheebra

Zebra phase, cheetah amplitude

Cheetah phase, zebra amplitude



Efros

Amplitude / Phase

- The frequency amplitude of natural images are quite similar
  - Heavy in low frequencies, falling off in high frequencies
    - Will *any* image be like that, or is it a property of the world we live in?
- Most information in the image is carried in the phase, not the amplitude
  - Not quite clear why

Efros

Properties of Fourier Transforms

- Linearity

$$\mathrm{F}[ax(t)+by(t)] = a\,\mathrm{F}[x(t)]+b\,\mathrm{F}[y(t)]$$

- Fourier transform of a real signal is symmetric about the origin
- The energy of the signal is the same as the energy of its Fourier transform

The Convolution Theorem

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

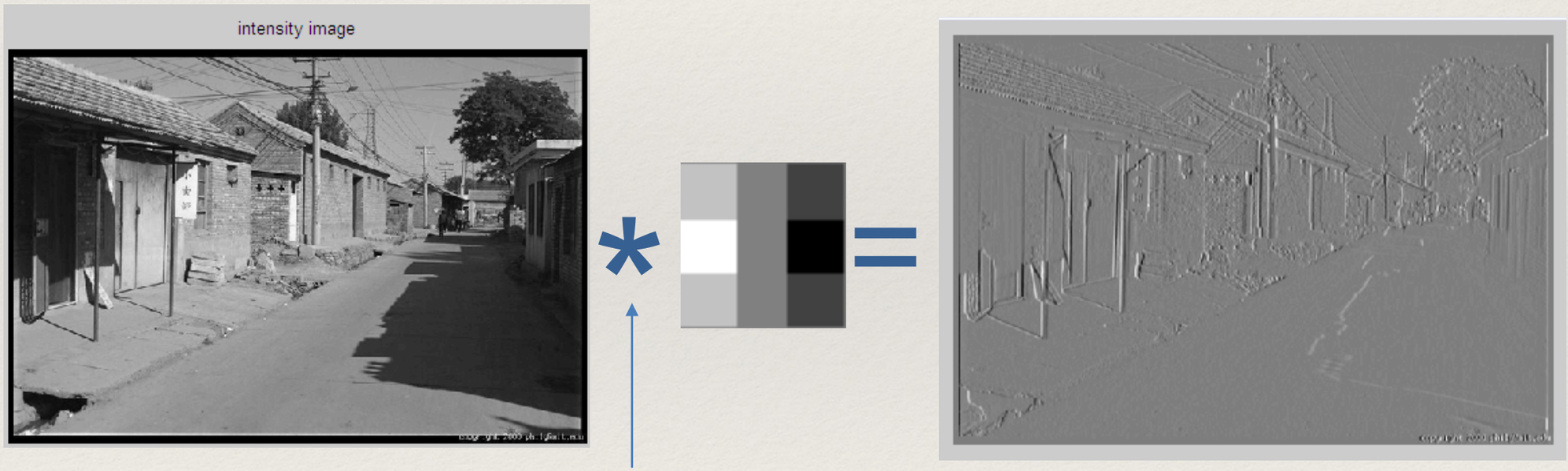
$$\mathrm{F}[g * h] = \mathrm{F}[g]\mathrm{F}[h]$$

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

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

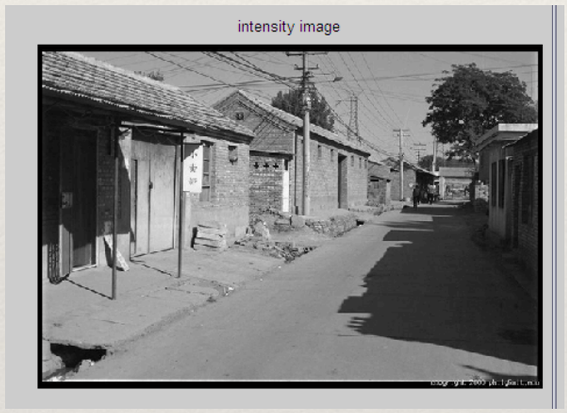
Filtering in spacial domain

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



Hays

Filtering in frequency domain

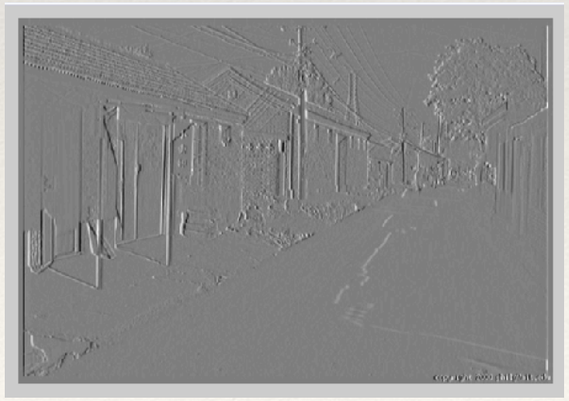


Fourier transform

Element-wise product

Fourier transform

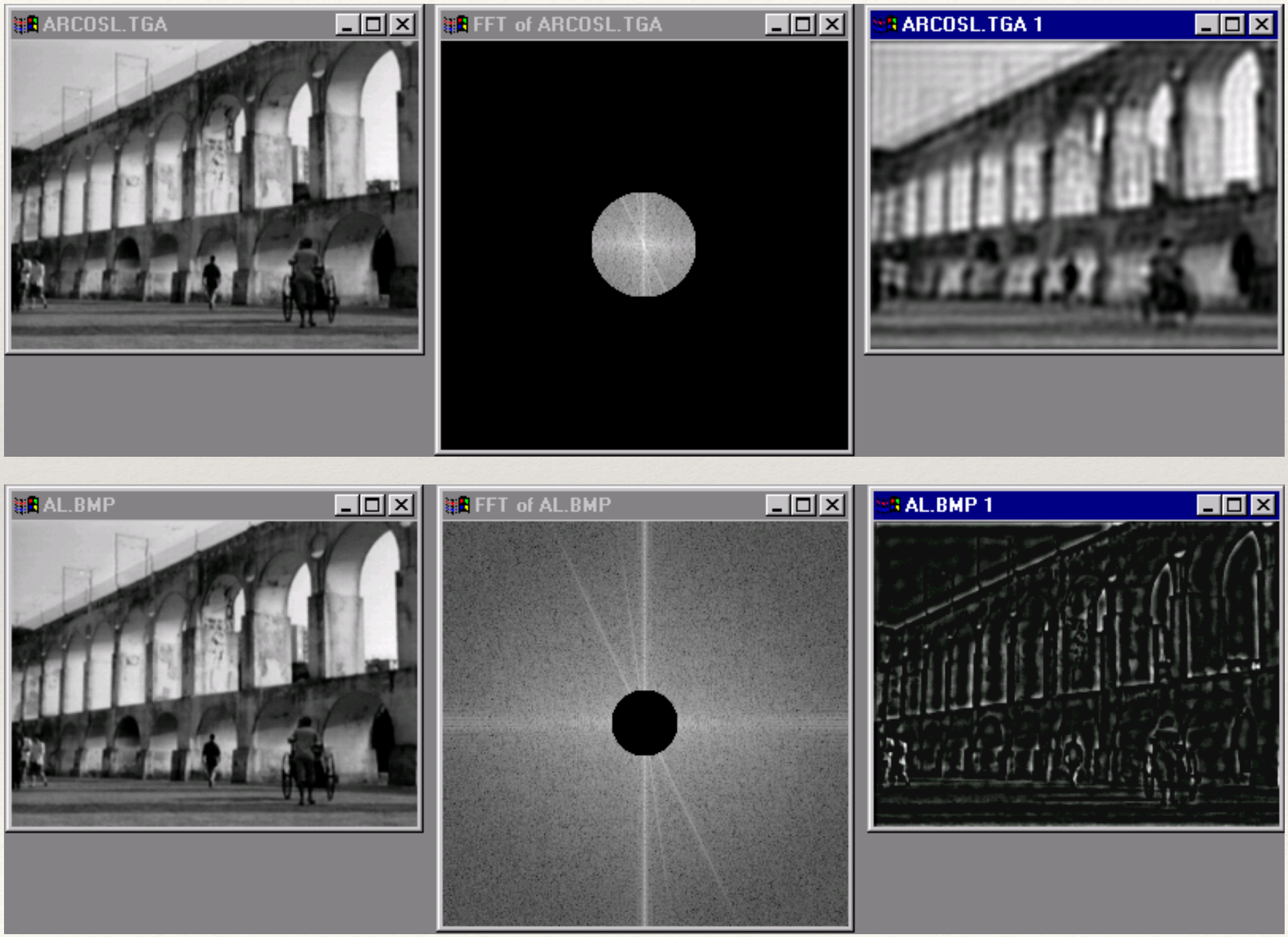
Inverse Fourier transform



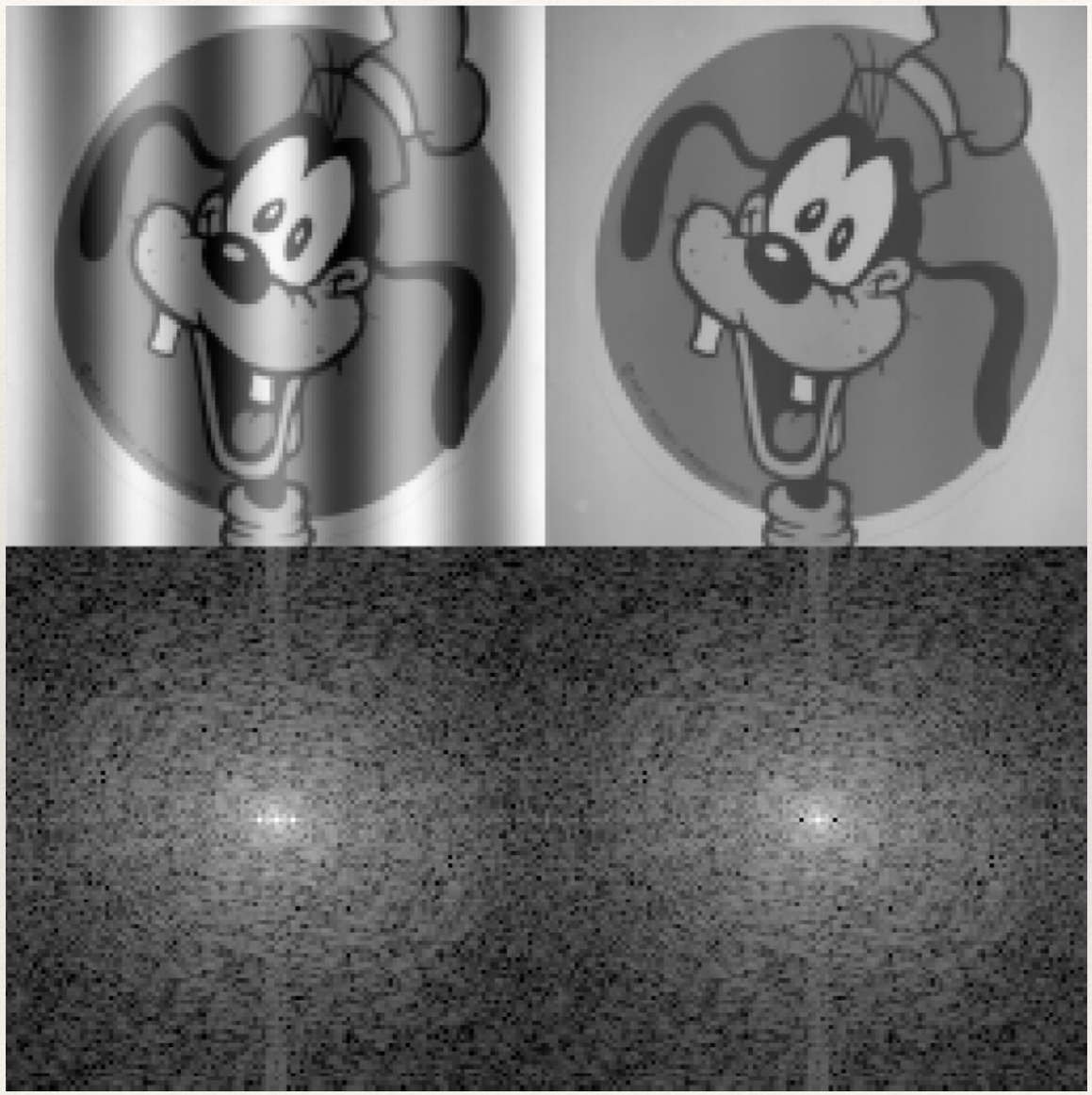
Slide: Hoiem

Frequency manipulation

Low and high pass filtering



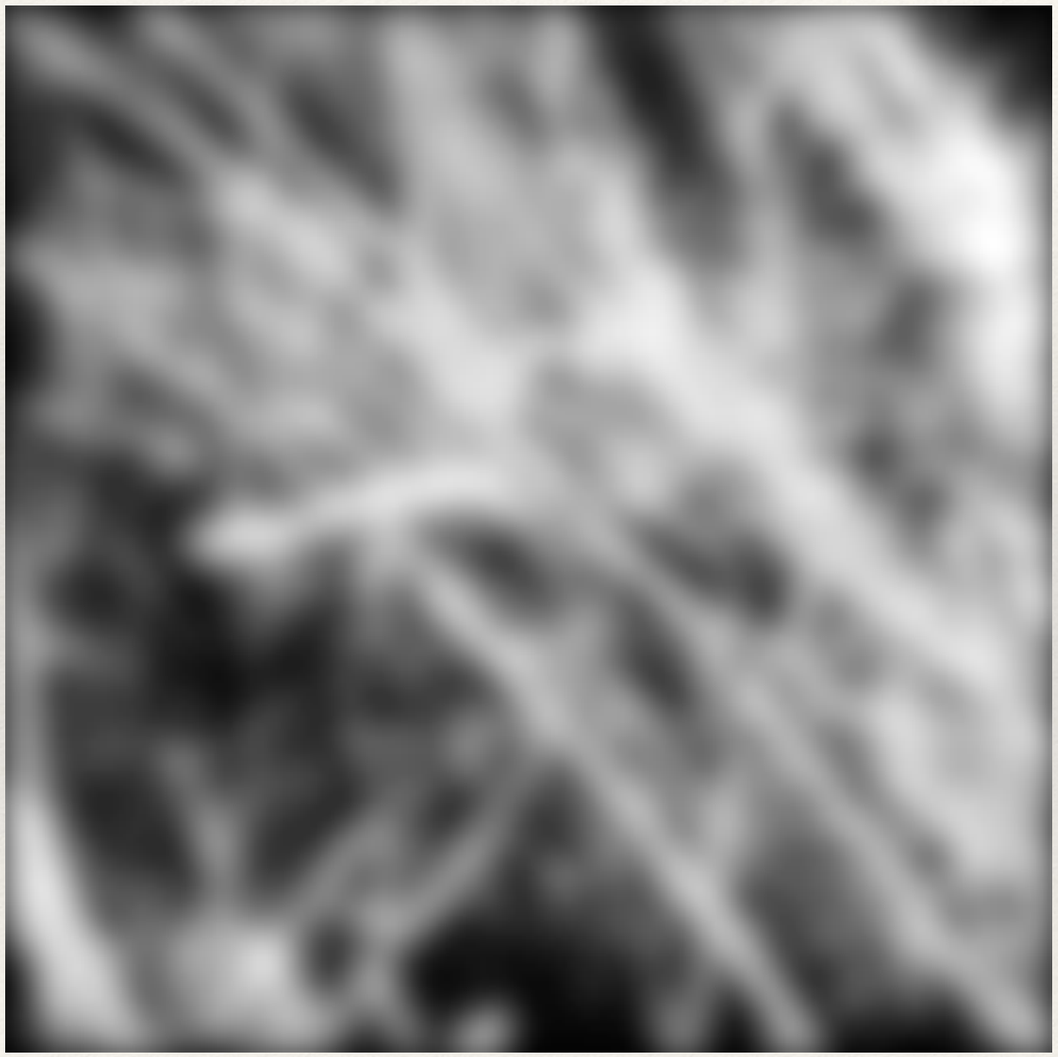
Removing frequency bands



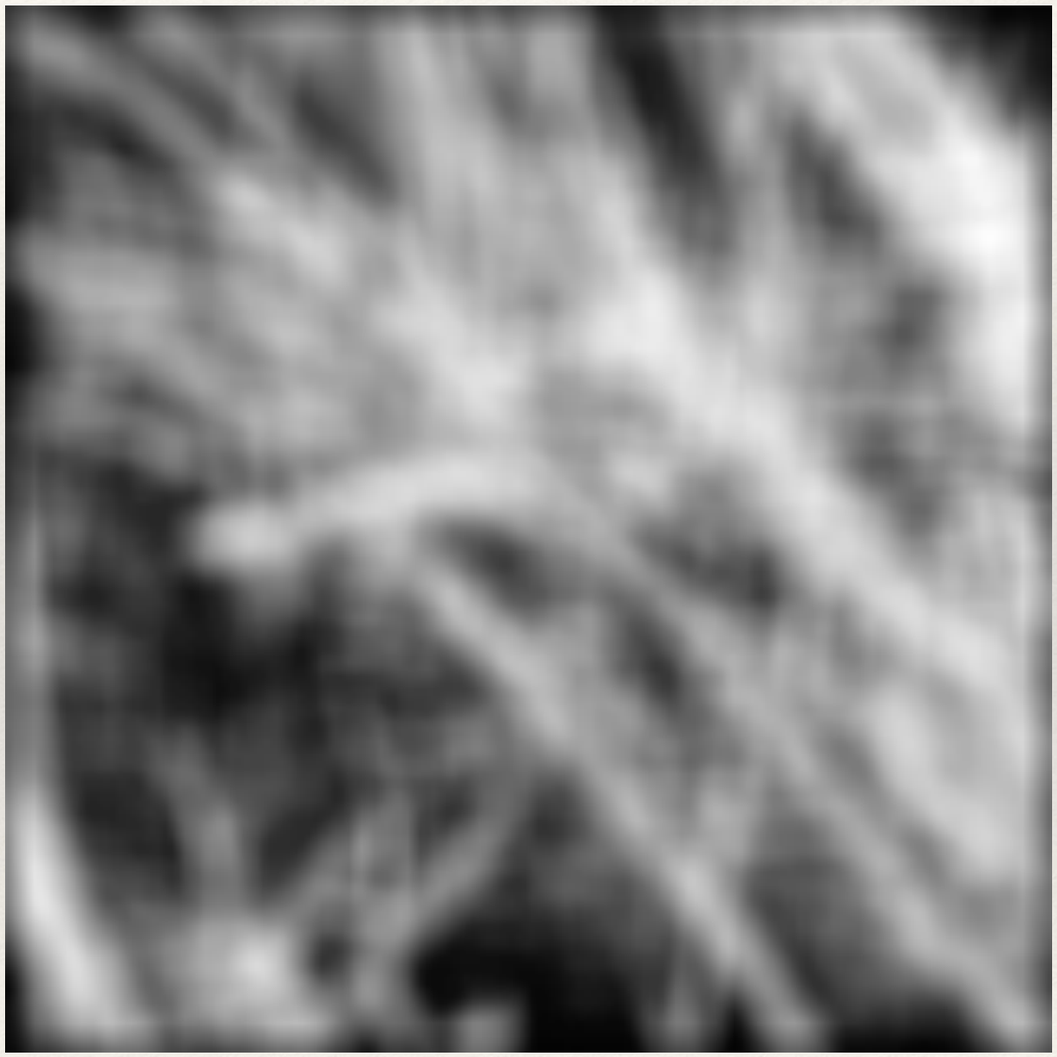
Brayer

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

Gaussian 

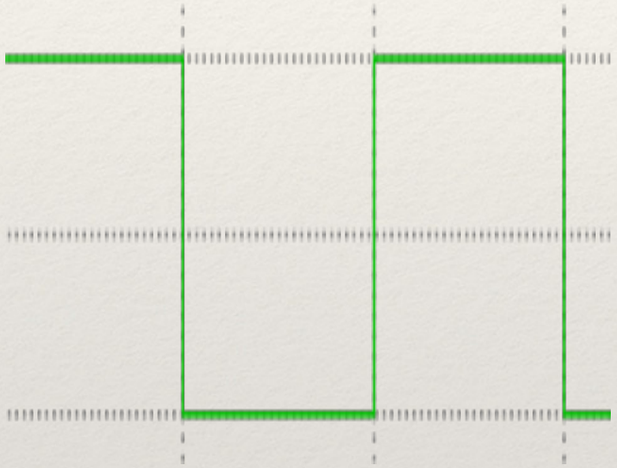


Box filter 



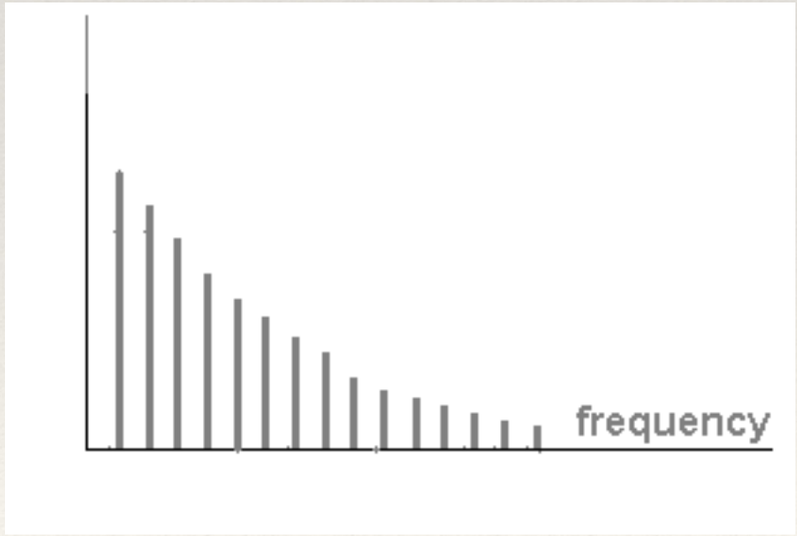
Why do we have those lines in the image?

- Sharp edges in the image need all frequencies to represent them.



=

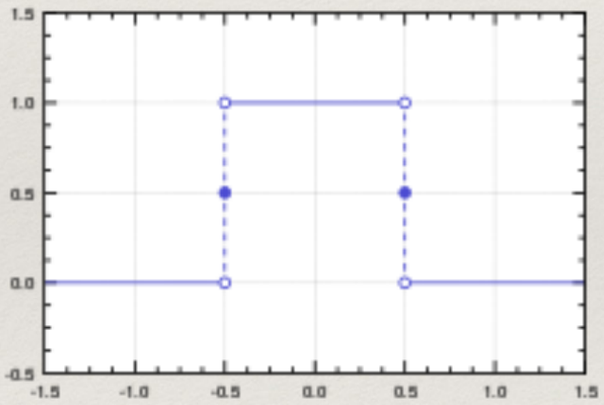
$$A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)$$



Box filter / sinc filter duality

- What is the spatial representation of the hard cutoff (box) in the frequency domain?
- http://madebyevan.com/dft/

Box filter

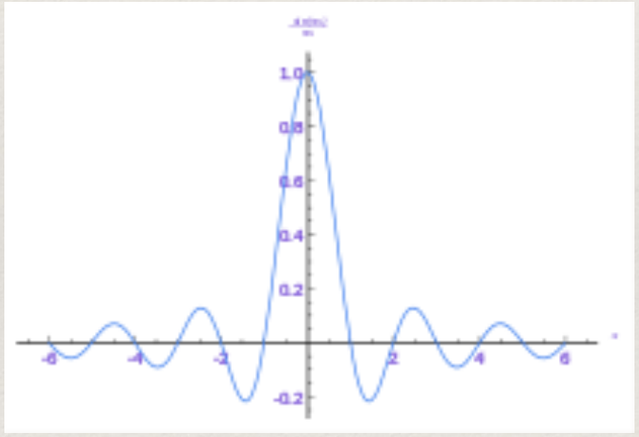


Spatial Domain

Frequency Domain

Sinc filter

sinc(x) = sin(x) / x

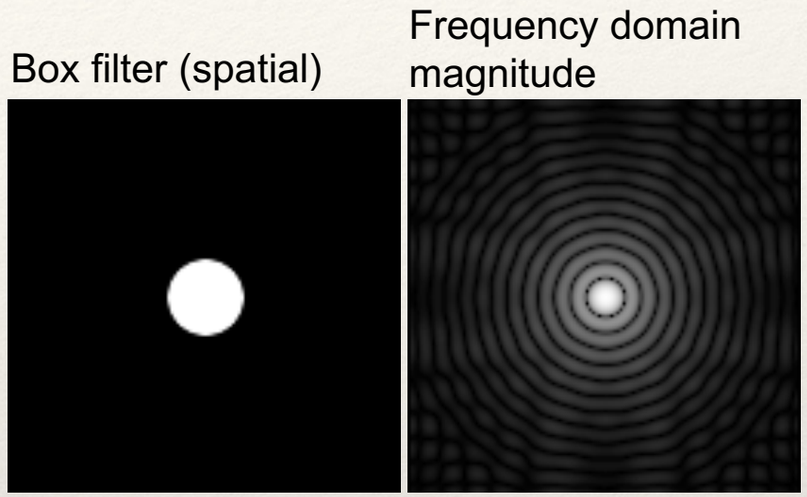


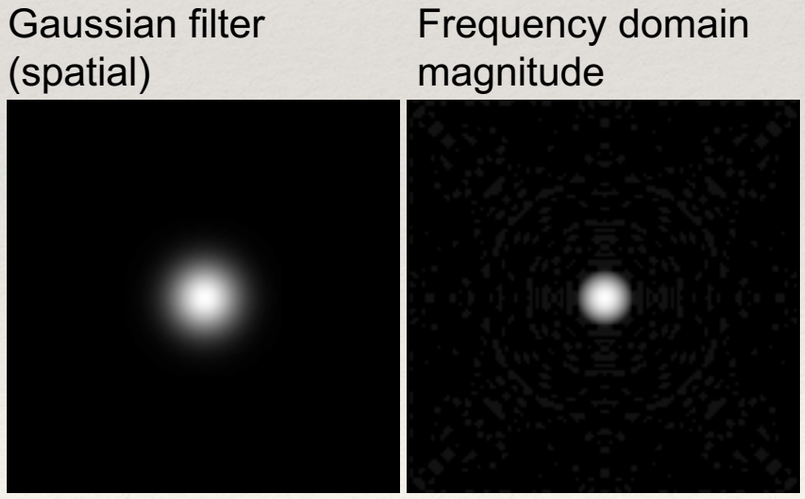
Frequency Domain

Spatial Domain

Gaussian filter duality

- Fourier transform of one Gaussian is another Gaussian (with inverse variance).
- Why is this useful?
  - Smooth degradation in frequency components
  - No sharp cut-off
  - No negative values
  - Never zero (infinite extent)



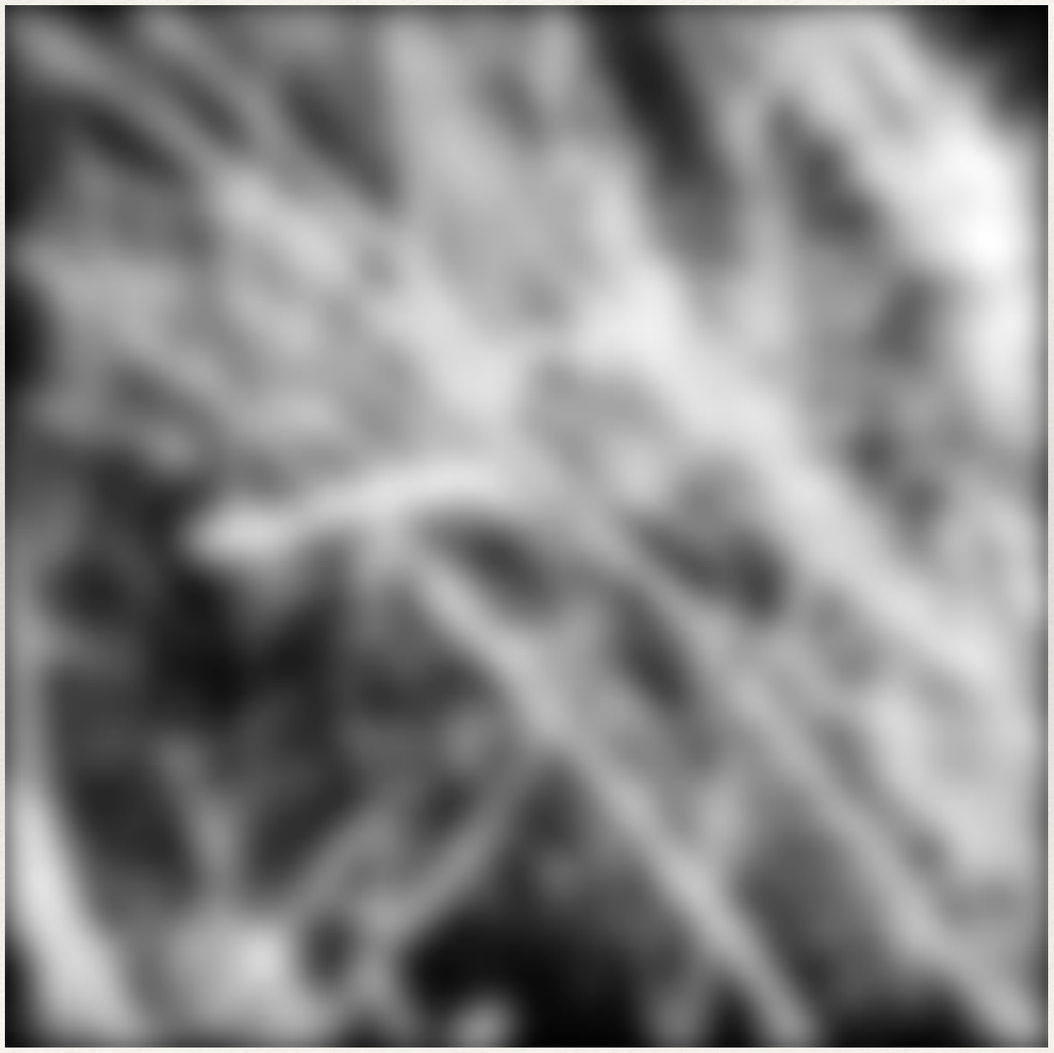


Source: James Tompkin

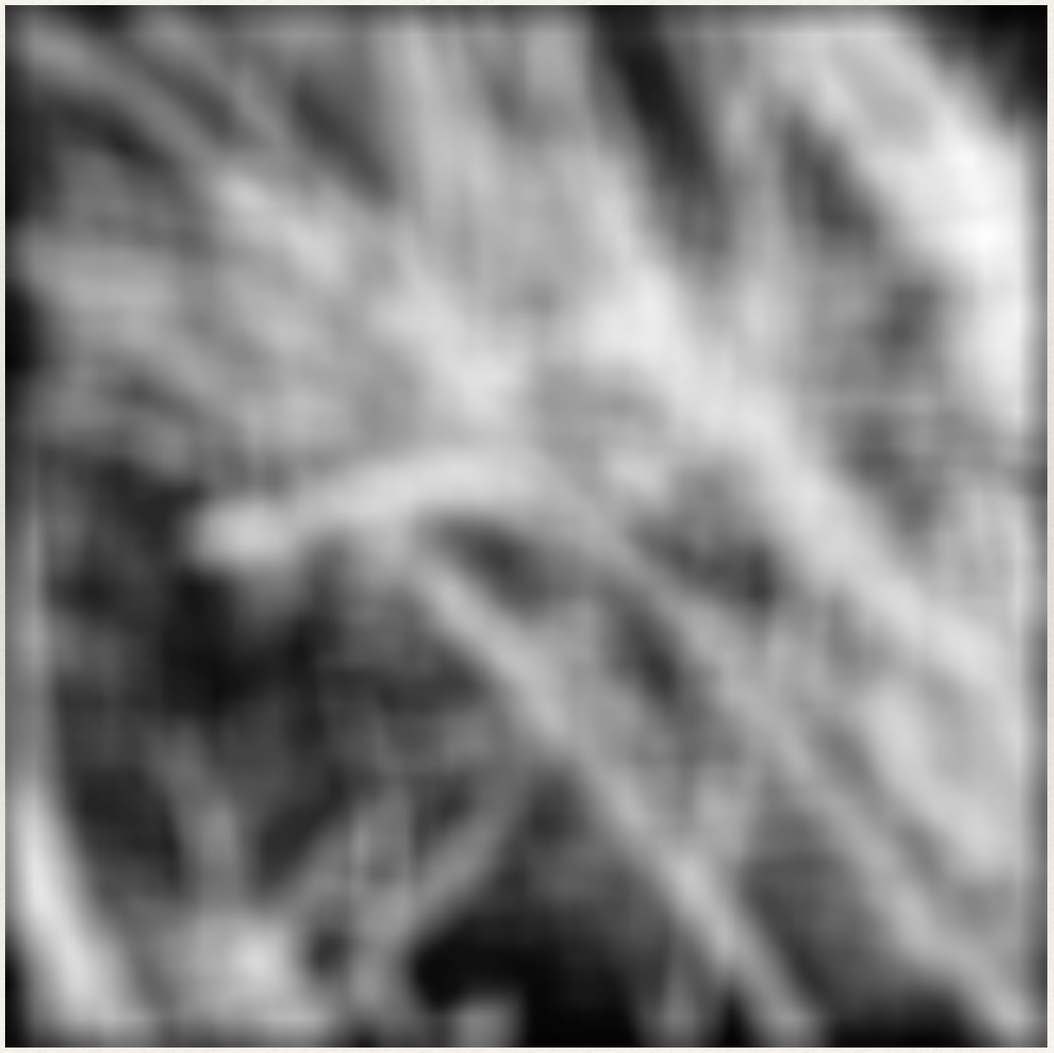
Ringing artifacts -> ‘Gibbs effect’

Where *infinite* series can never be reached

Gaussian 



Box filter 

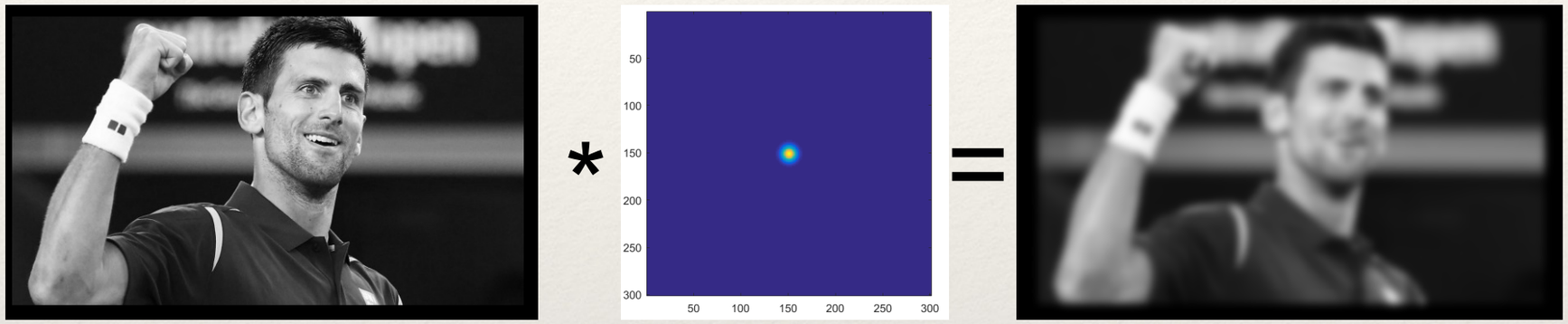


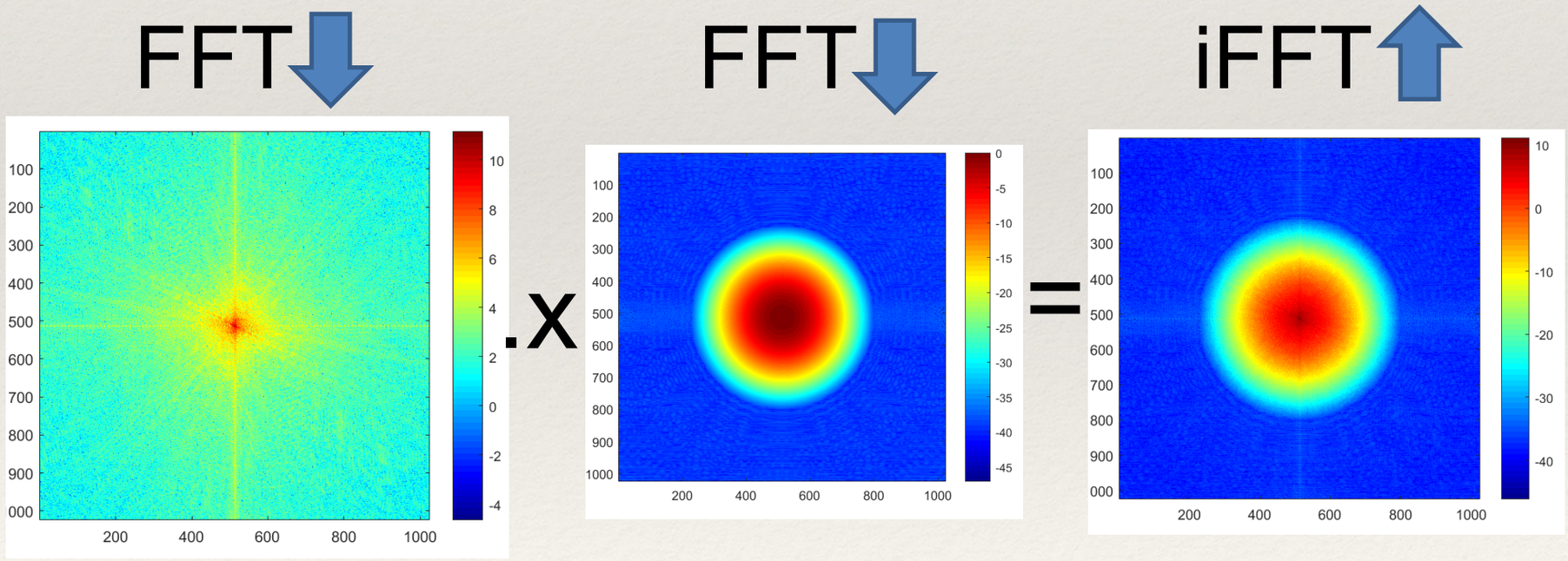
Source: James Tompkin

Is convolution invertible?

- If convolution is just multiplication in the Fourier domain, isn't deconvolution just division?
- Sometimes, it clearly is invertible (e.g. a convolution with an identity filter)
- In one case, it clearly isn't invertible (e.g. convolution with an all zero filter)
- What about for common filters like a Gaussian?

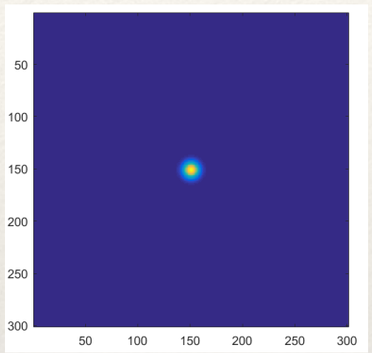
Convolution





Hays

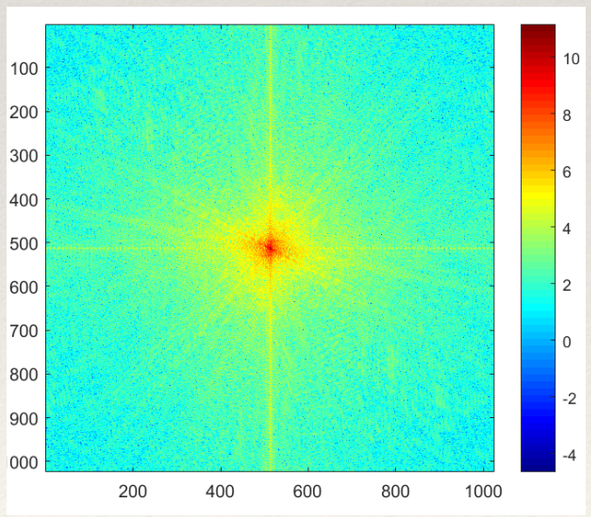
Deconvolution



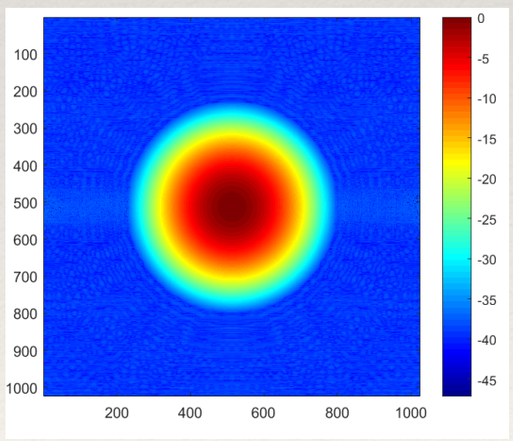
iFFT

FFT

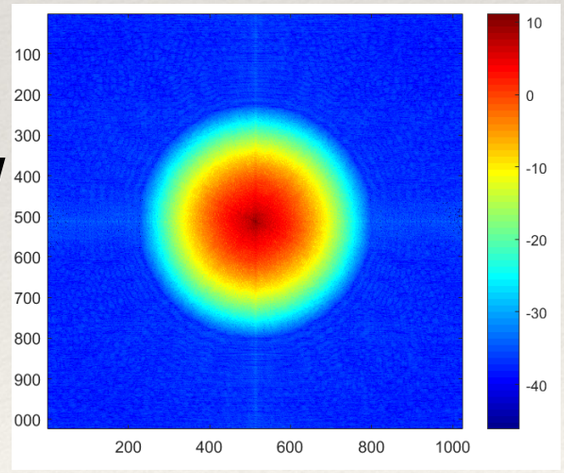
FFT



=

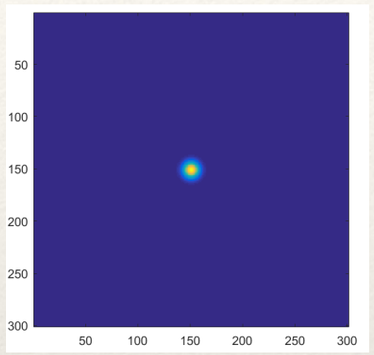


./



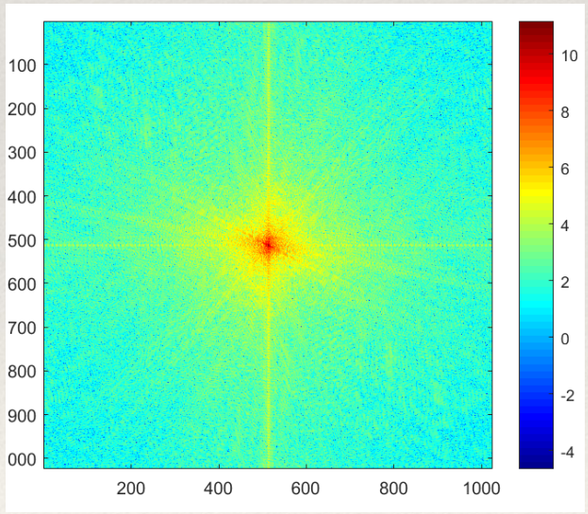
Hays

But under more realistic conditions



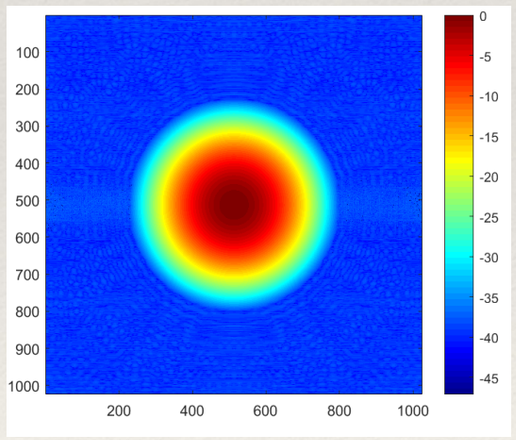
Random noise, .000001 magnitude

iFFT



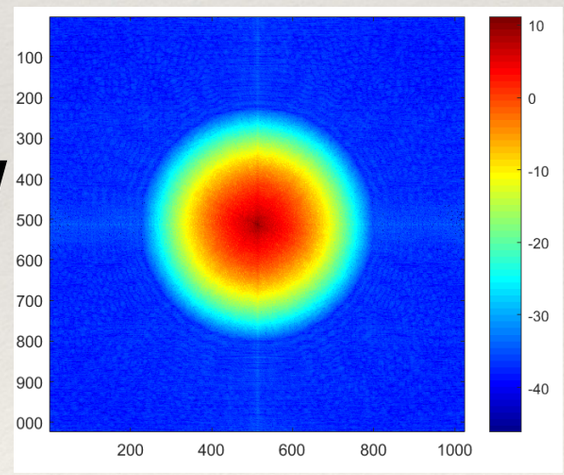
=

FFT



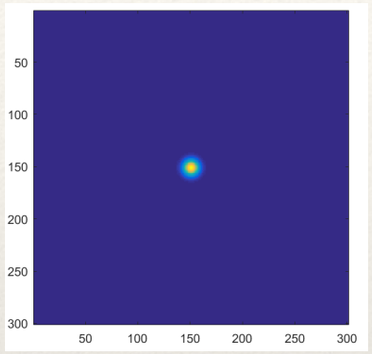
./

FFT



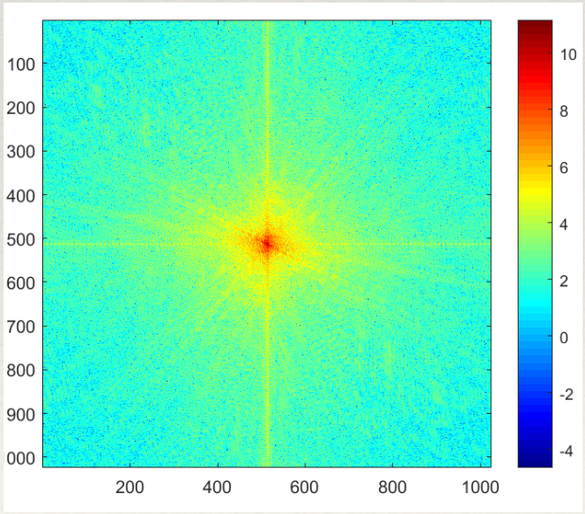
Hays

But under more realistic conditions



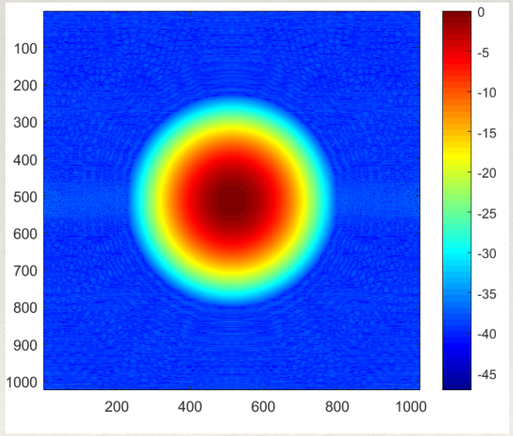
Random noise, .0001 magnitude

iFFT



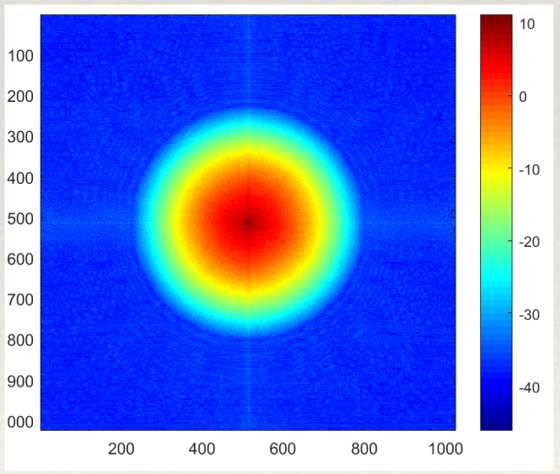
=

FFT



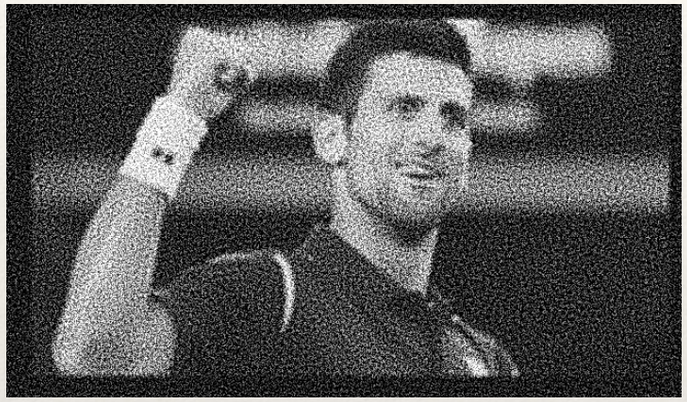
./

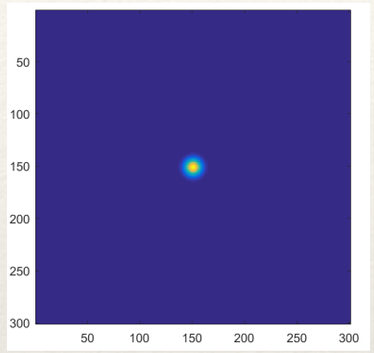
FFT



Hays

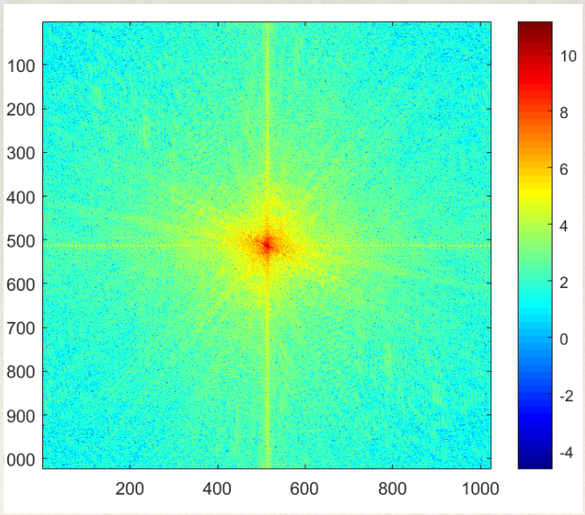
But under more realistic conditions





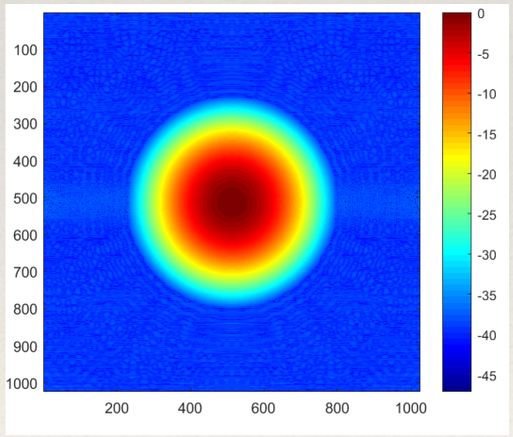
Random noise, .001 magnitude

iFFT



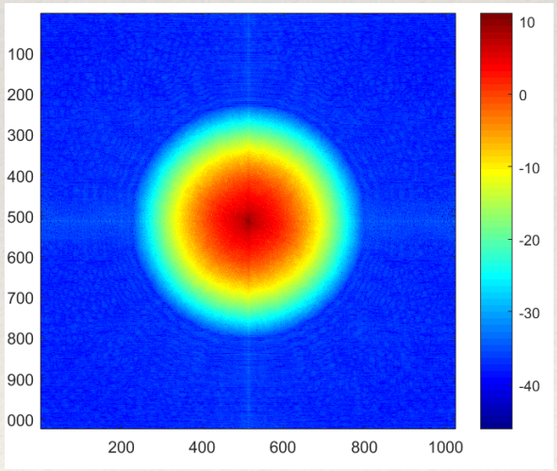
=

FFT



./

FFT



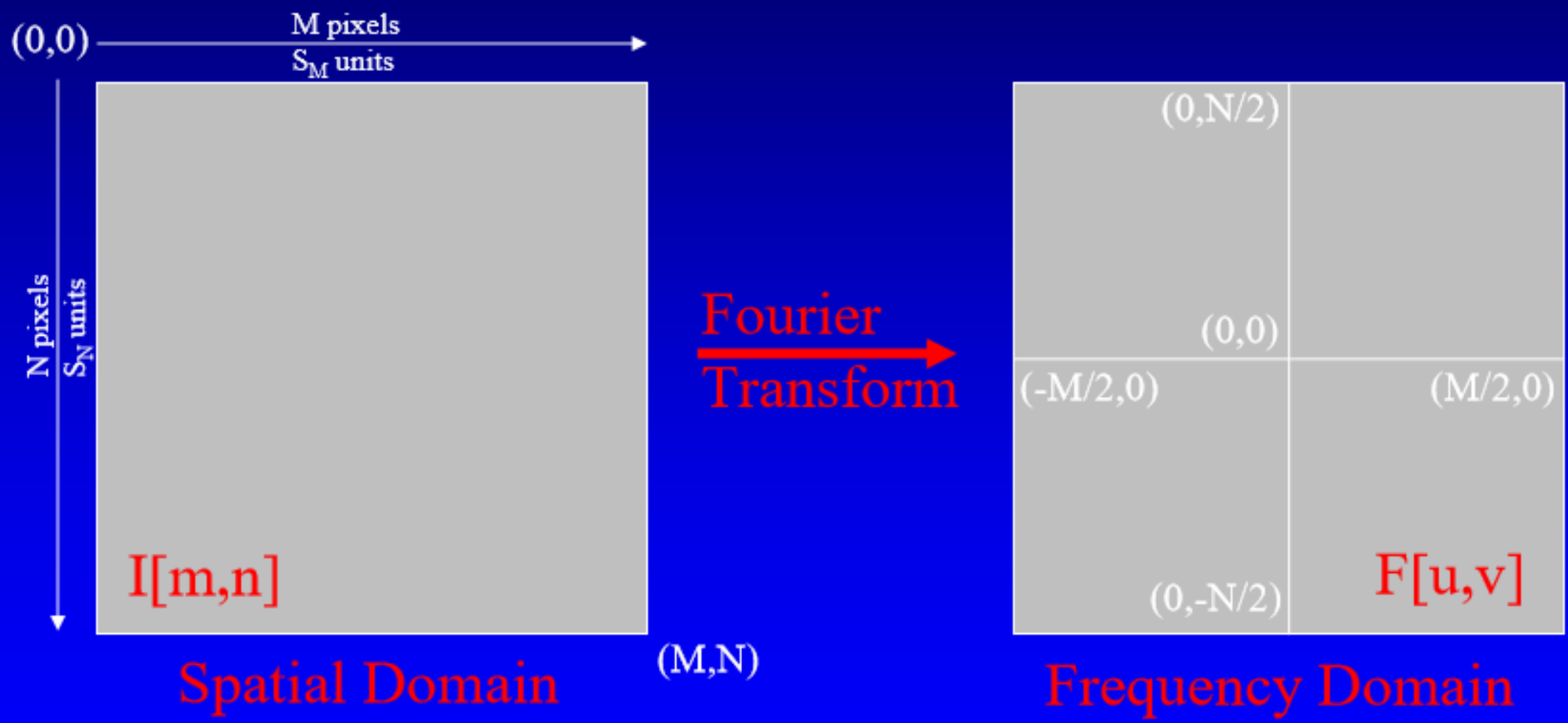
Hays

Deconvolution is hard.

- Active research area.
- Even if you know the filter (non-blind deconvolution), it is still hard and requires strong *regularization* to counteract noise.
- If you don’t know the filter (blind deconvolution), then it is harder still.

2D Discrete Fourier Transform

$$F[u,v] = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I[m,n] \cdot e^{-i2p\left(\frac{um}{M} + \frac{vn}{N}\right)}$$

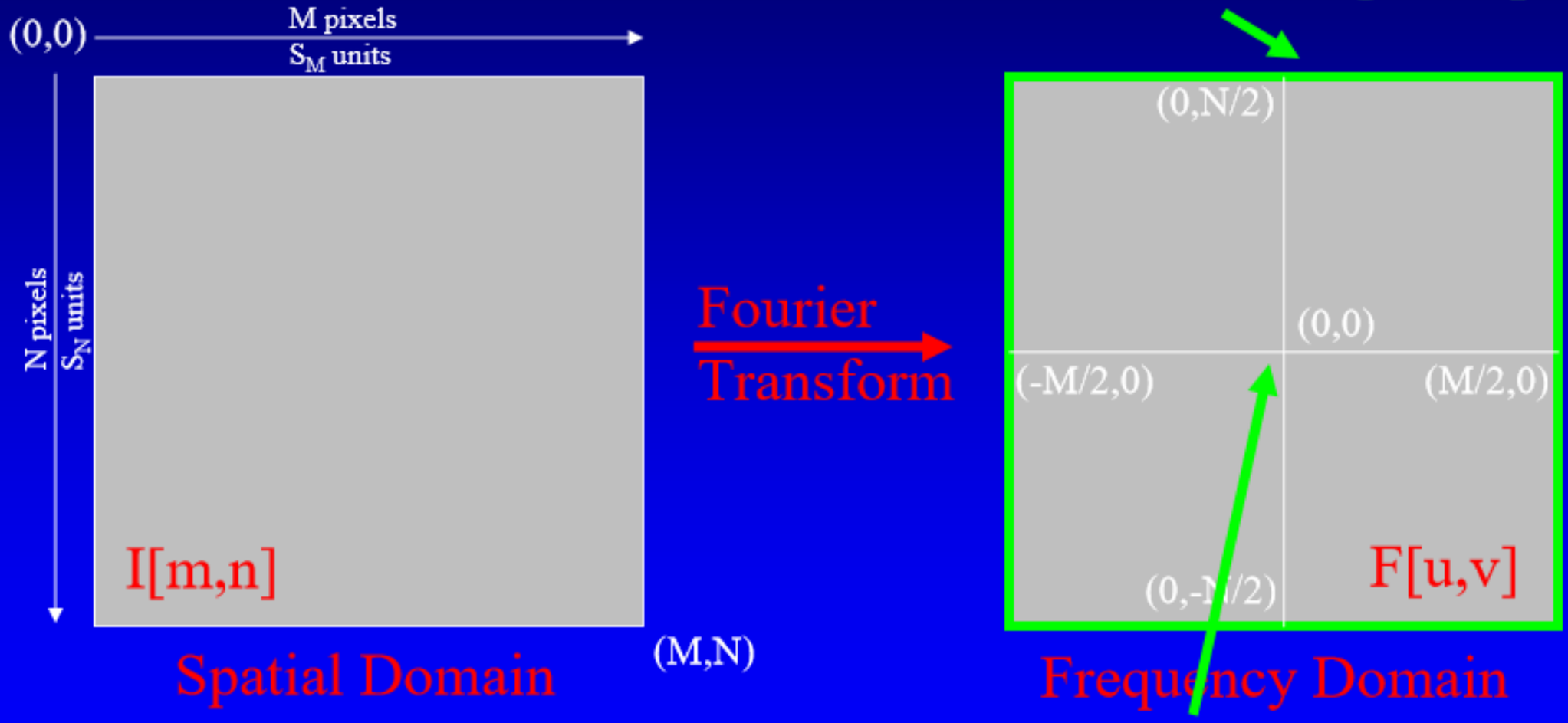


Source: Seul et al, *Practical Algorithms for Image Analysis*, 2000, p. 249, 262.

2D FFT can be computed as two discrete Fourier transforms in 1 dimension

2D Discrete Fourier Transform

Edge represents highest frequency, smallest resolvable length (2 pixels)



Center represents lowest frequency, which represents average pixel value

Earl F. Glynn