CSCE 448/748 – Computational Photography

Sampling, Frequency, and Filtering

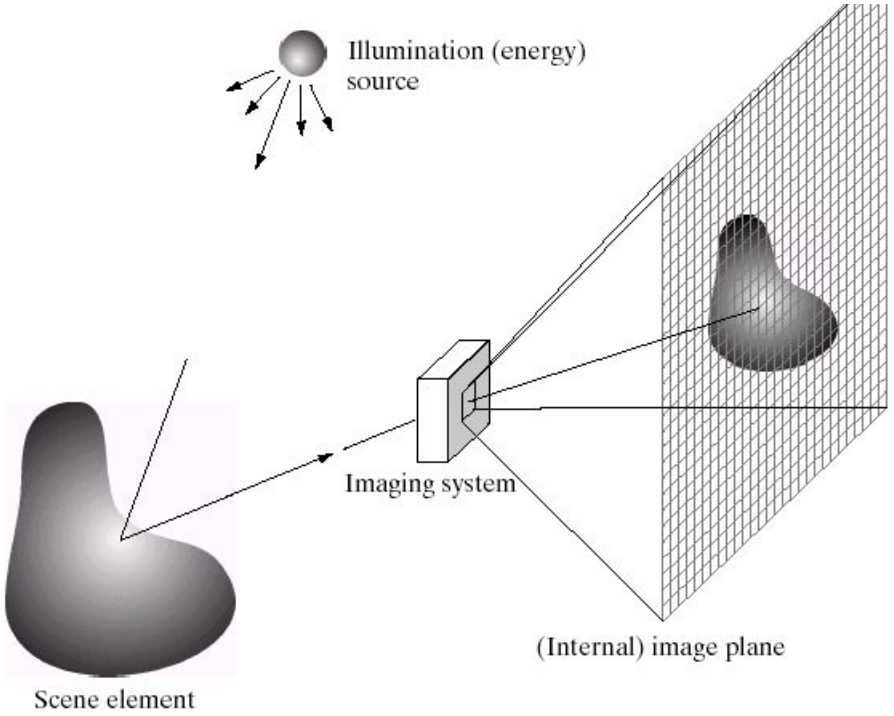
Nima Kalantari

Many slides from Alexei A. Efros, James Hayes, Rob Fergus

Outline

- Sampling and aliasing
- Frequency domain
- Smoothing filters
- Antialiasing
- Other filters

Image formation involves sampling



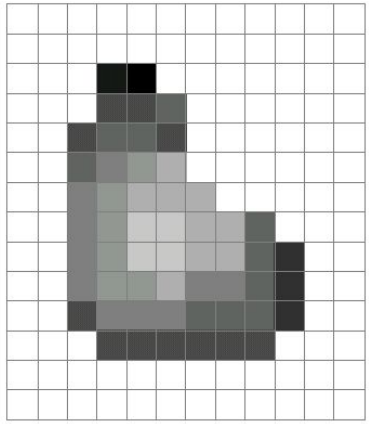


Image formation involves sampling

Scene

Digital image

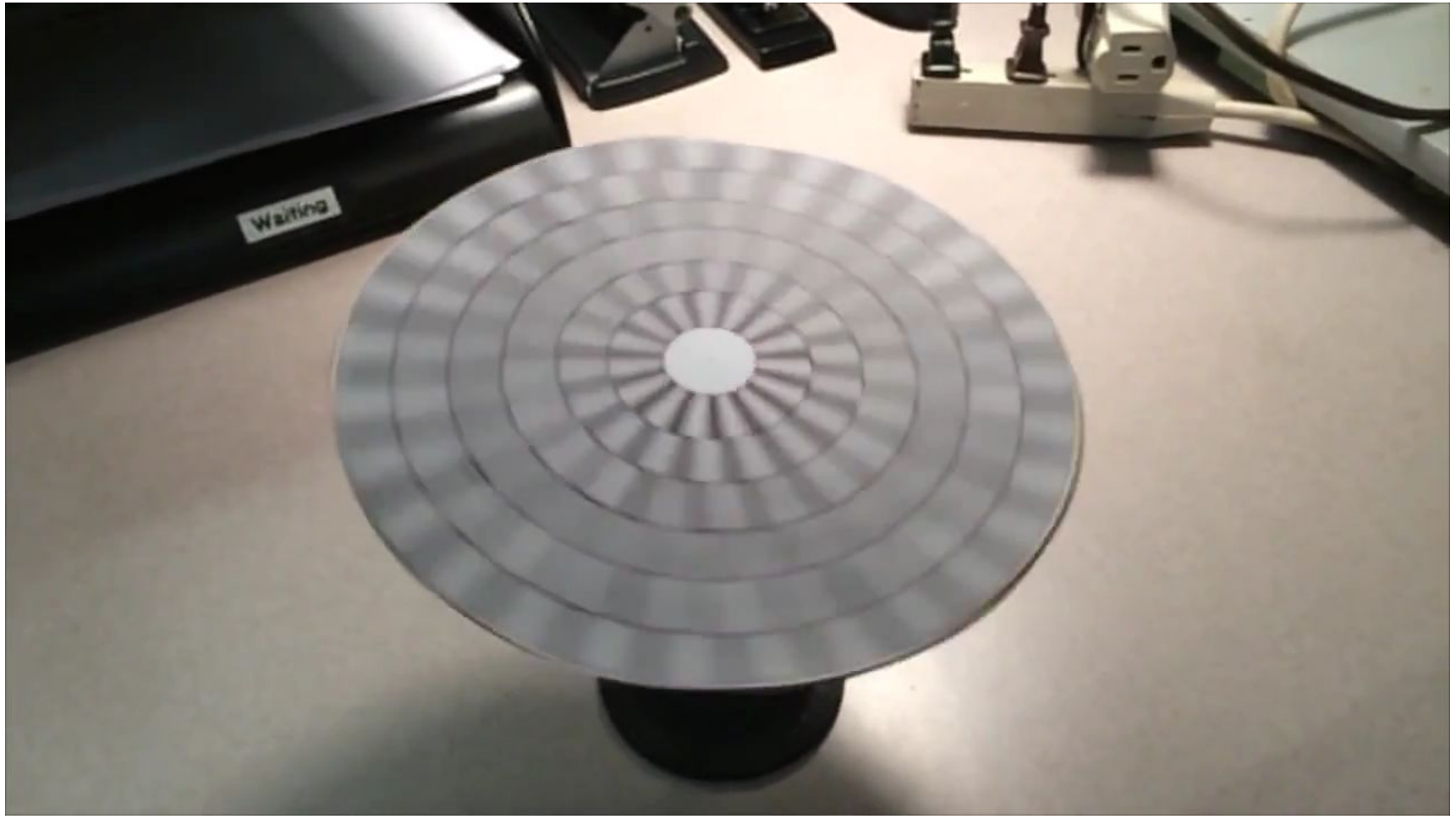
Aliasing in images (Moire pattern)

Original

Aliased

Credit: Scientific Volume Imaging

Aliasing in video (wagon-wheel effect)

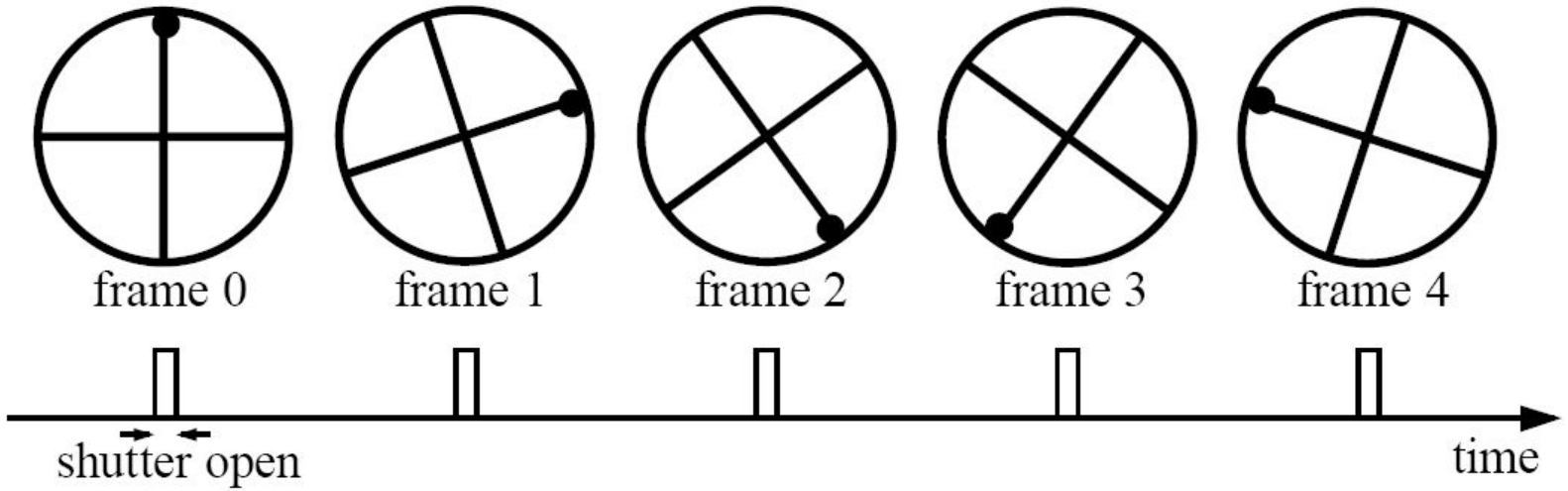


Created by Jesse Mason, https://www.youtube.com/watch?v=QOwzkND_ooU

Aliasing in video (wagon wheel effect)

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

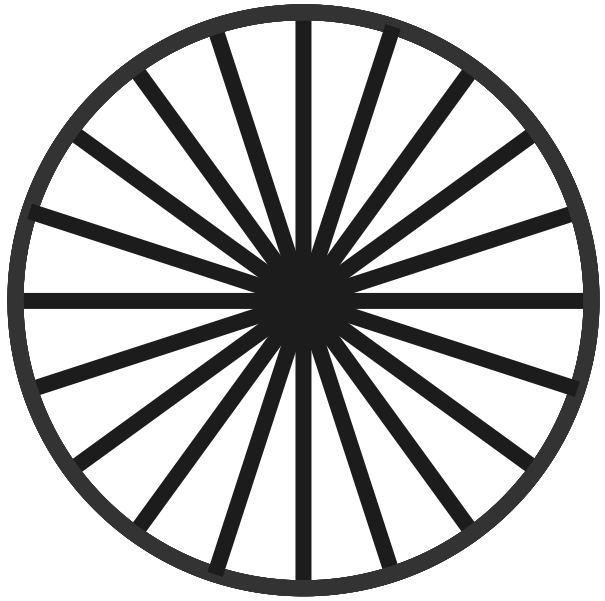
If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



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

Slide by Steve Seitz

Aliasing in video (wagon wheel effect)



Sampling Artifacts in Computer Graphics

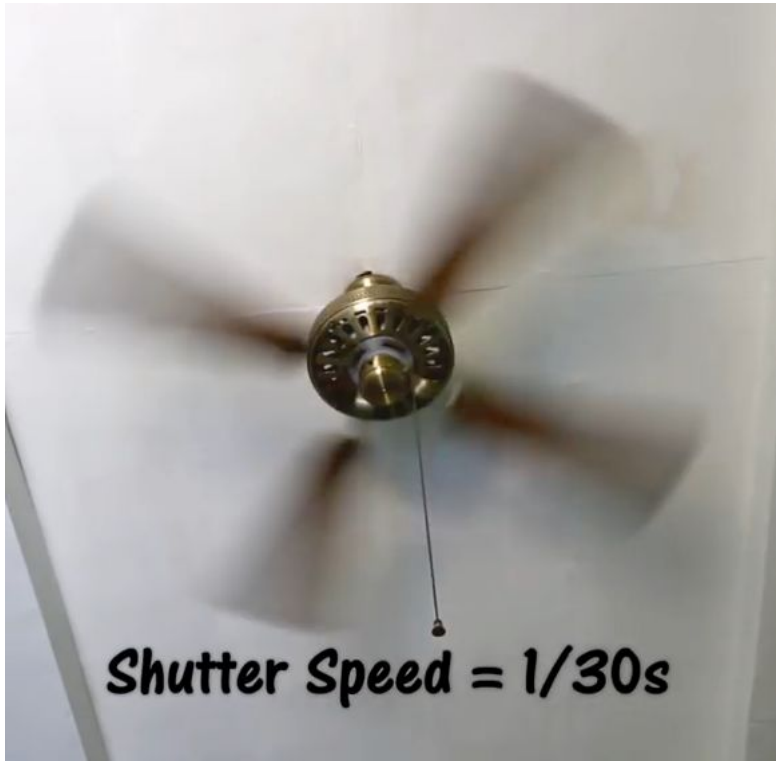
- Artifacts due to sampling - “Aliasing”
  - Jaggies – sampling in space
  - Moire – undersampling images (and textures)
  - Wagon wheel effect – sampling in time
  - [Many more] …

- Aliasing
  - Fast-changing signals (high frequency) sampled too slowly

Antialiasing

Video: Point vs Antialiased Sampling

Point in Time



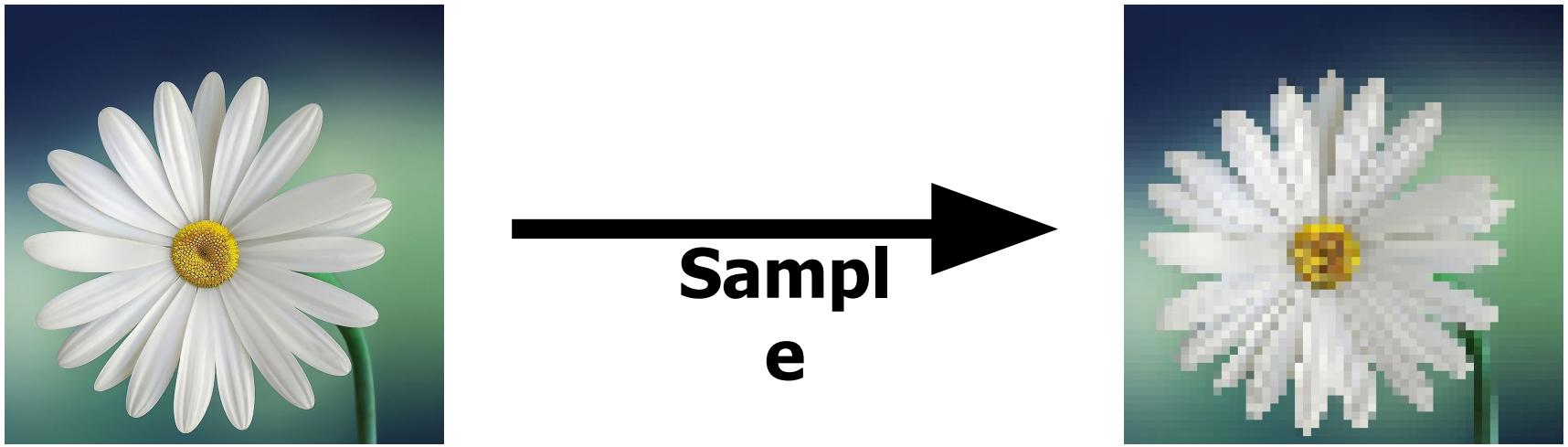
Motion Blurred

Video: Point Sampling in Time

Credit: Aris & cams youtube, https://youtu.be/NoWwxTktoFs

30 fps video. 1/800 second exposure is sharp in time, causes time aliasing.

Imaging: sampling in space



Note jagged edges of the flower petals

Imaging: antialiased sampling

Pre-Filter **Sample**

Note antialiased edges

Aliasing and Antialiasing

- Why undersampling results in aliasing?
- Why pre-filtering reduces aliasing?

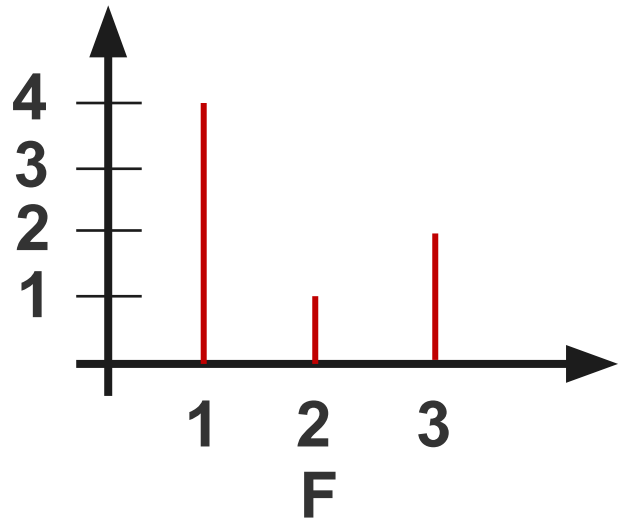
Outline

- Sampling and aliasing
- Frequency domain
- Smoothing filters
- Antialiasing
- Other filters

Image representation with basis functions

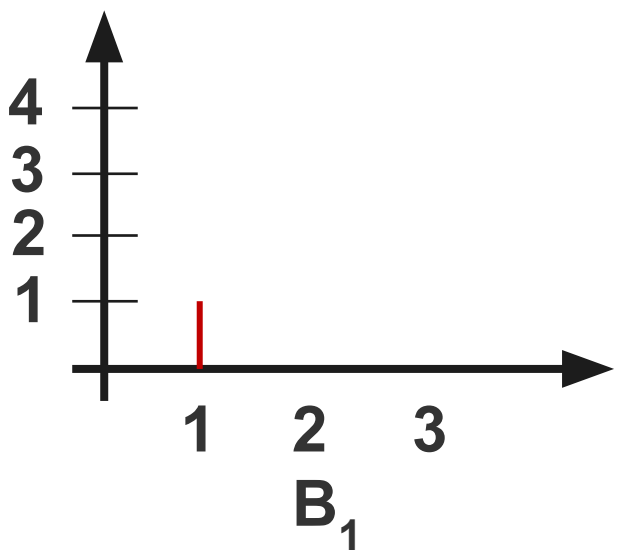
Spatial basis (1D example)

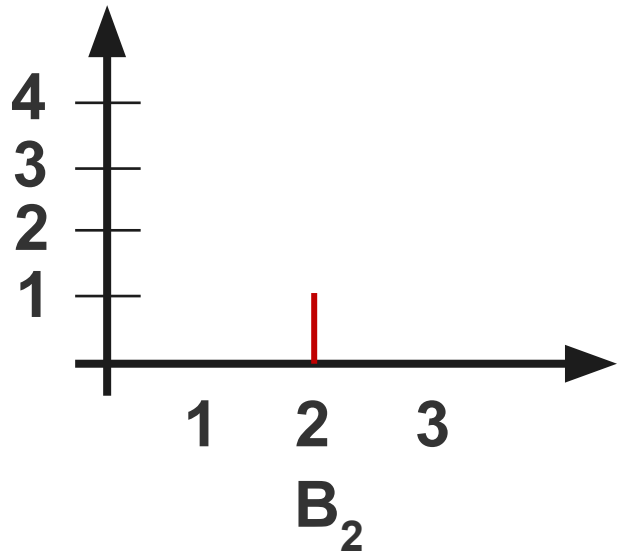
- **Signal**

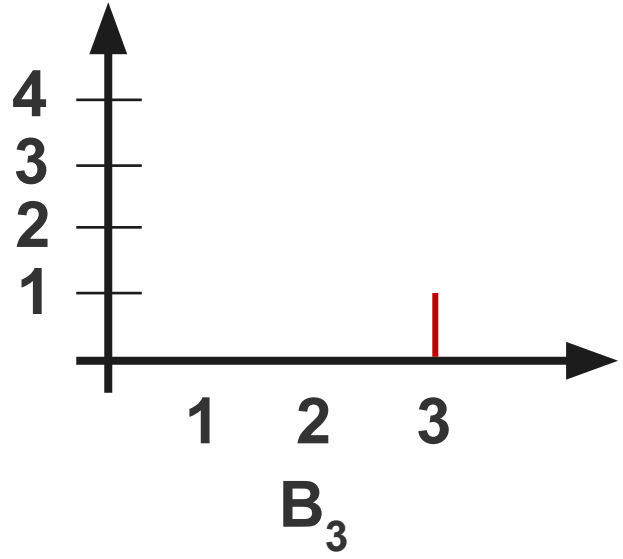


$$F = 4\,B_1 + 1\,B_2 + 2\,B_3$$

- **Basis**







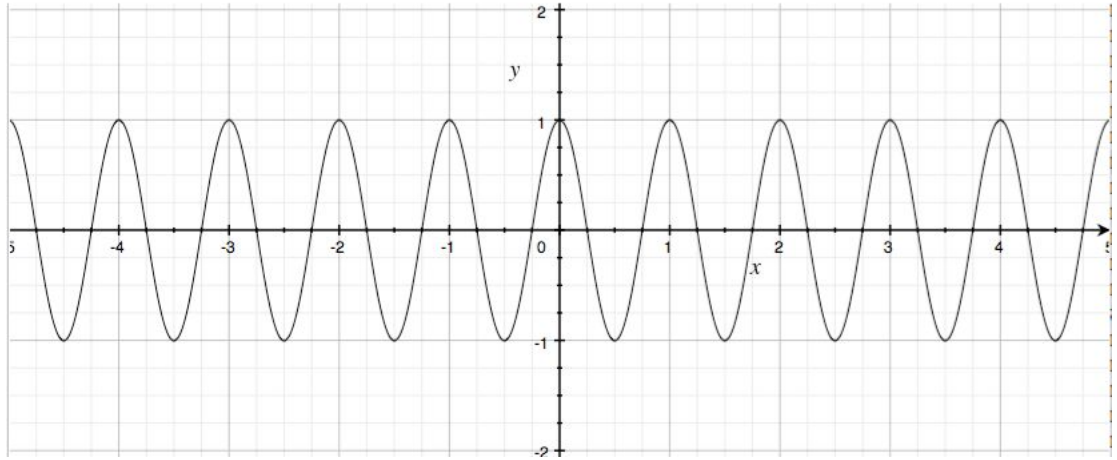
Spatial basis

- In 2D, the basis functions will be 2 dimensional
- Each basis will have a value of 1 at each pixel
- The image can be described as a weighted combination of the basis functions

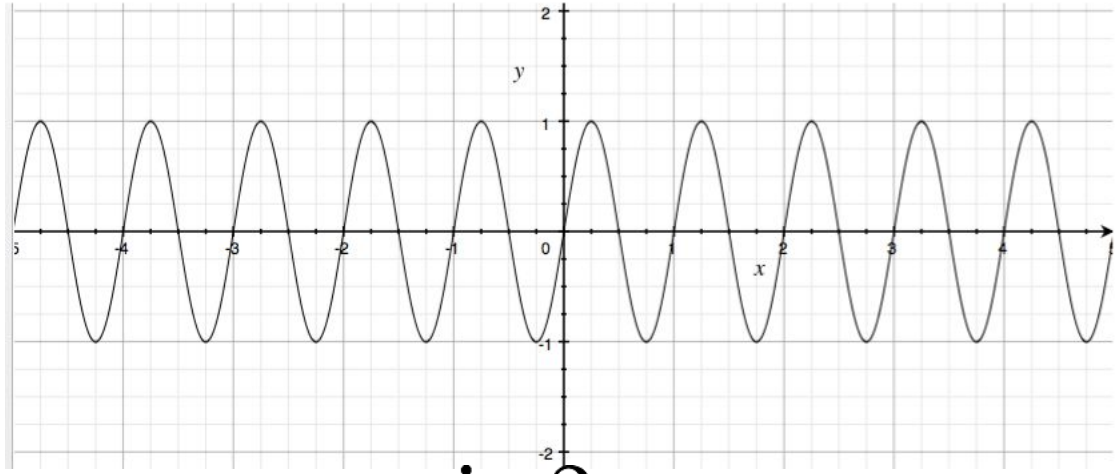
Frequency domain

- Representing a signal using frequency basis functions
- The signal is represented as a weighted combination of a series of basis functions at different frequencies

Sines and Cosines



$$\cos 2\pi x$$

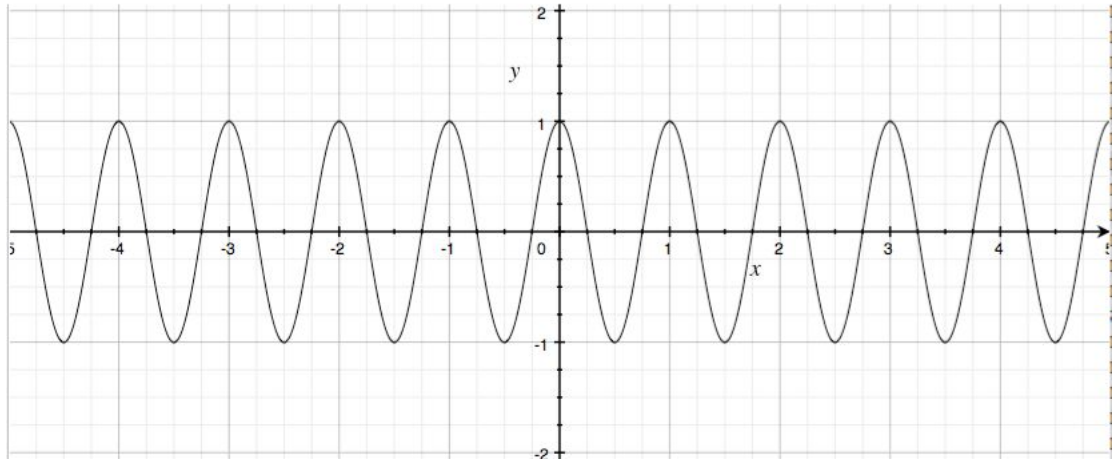


$$\sin 2\pi x$$

Frequencies

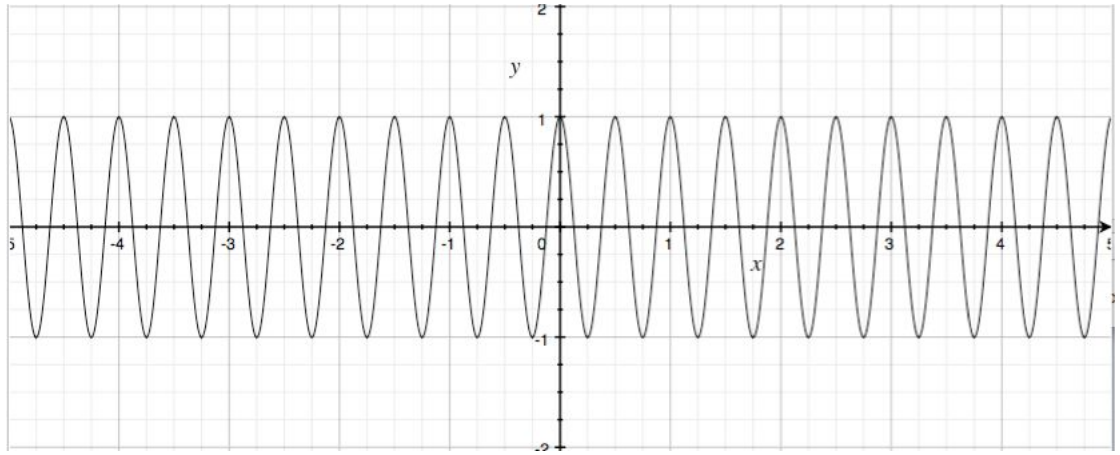
$$\cos 2\pi\omega x$$

$$\omega = \frac{1}{T}$$



$$\omega = 1$$

$$\cos 2\pi x$$



$$\omega = 2$$

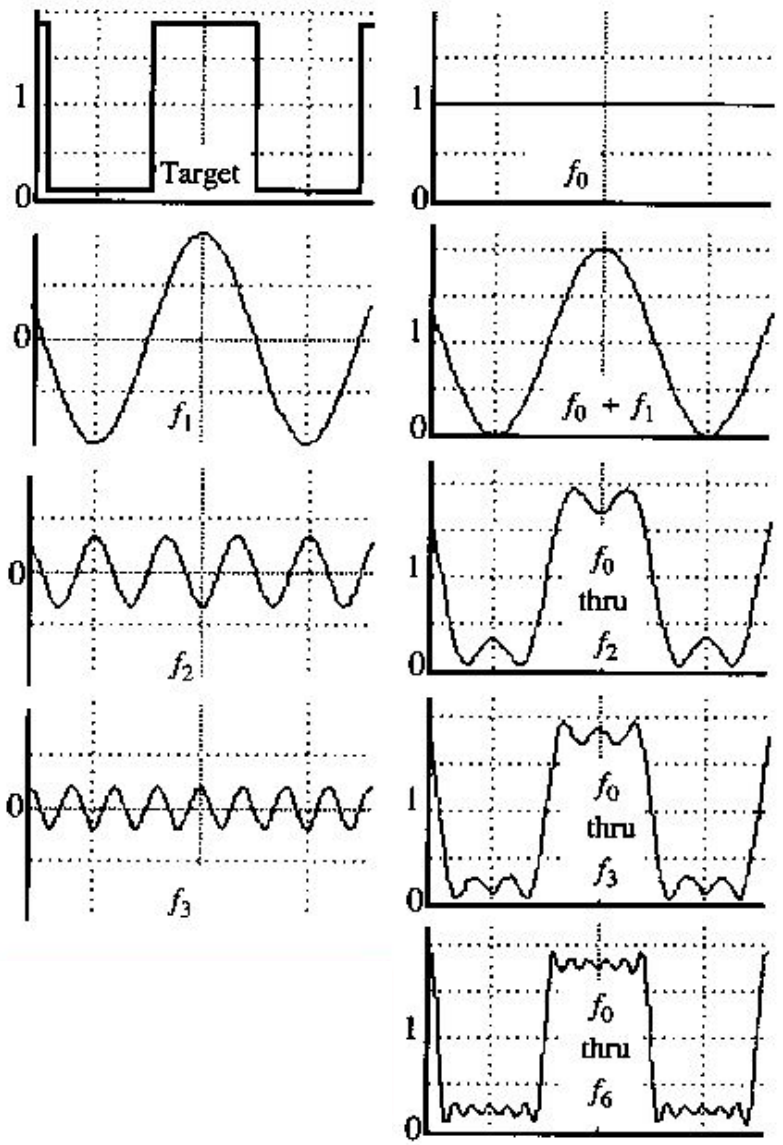
$$\cos 4\pi x$$

Fourier Transform

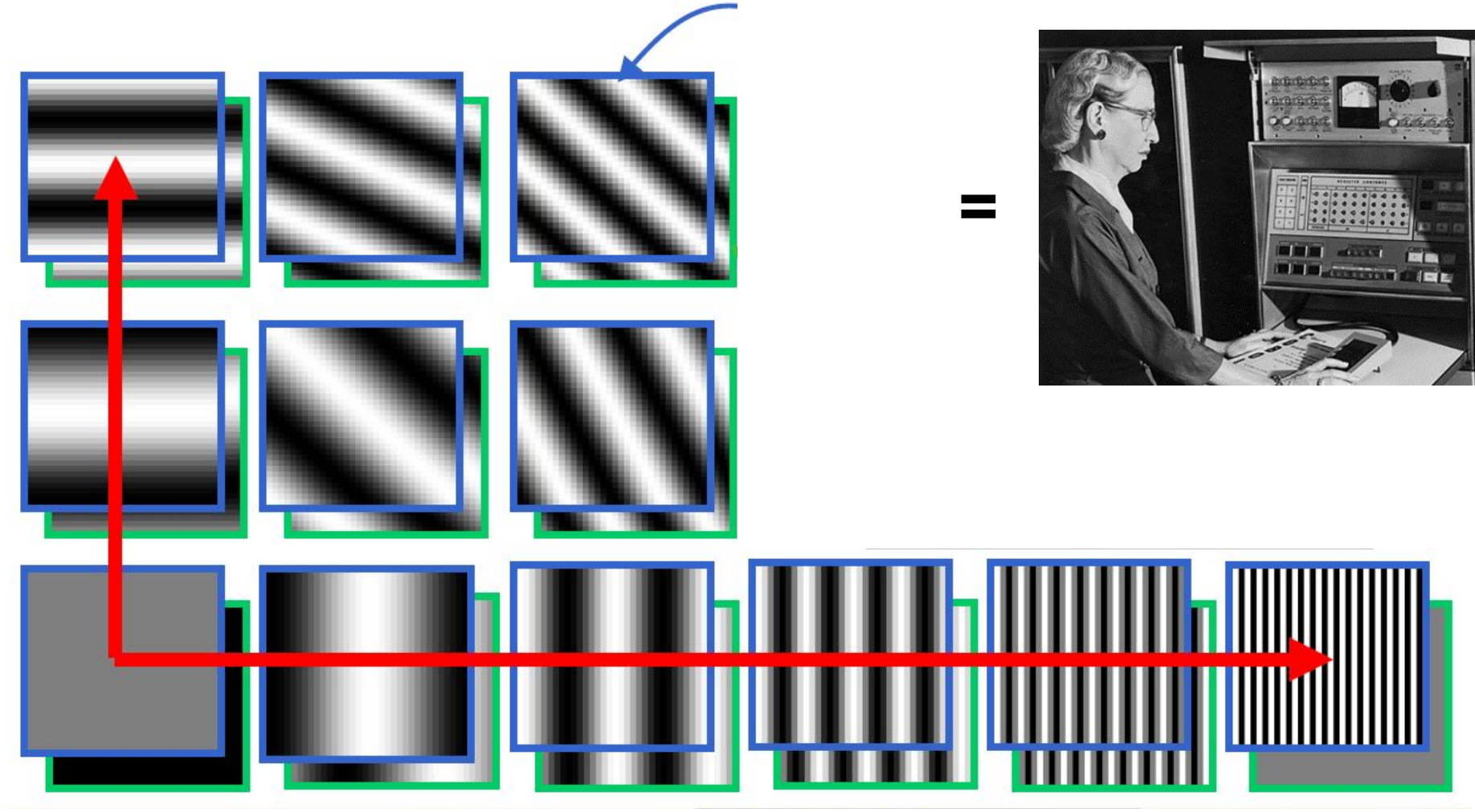
- Represent a function as a weighted sum of sines and cosines

Joseph Fourier 1768 - 1830

$$f(\text{target}) = A_0 f_0 + A_1 f_1 + A_2 f_2 + \ldots$$



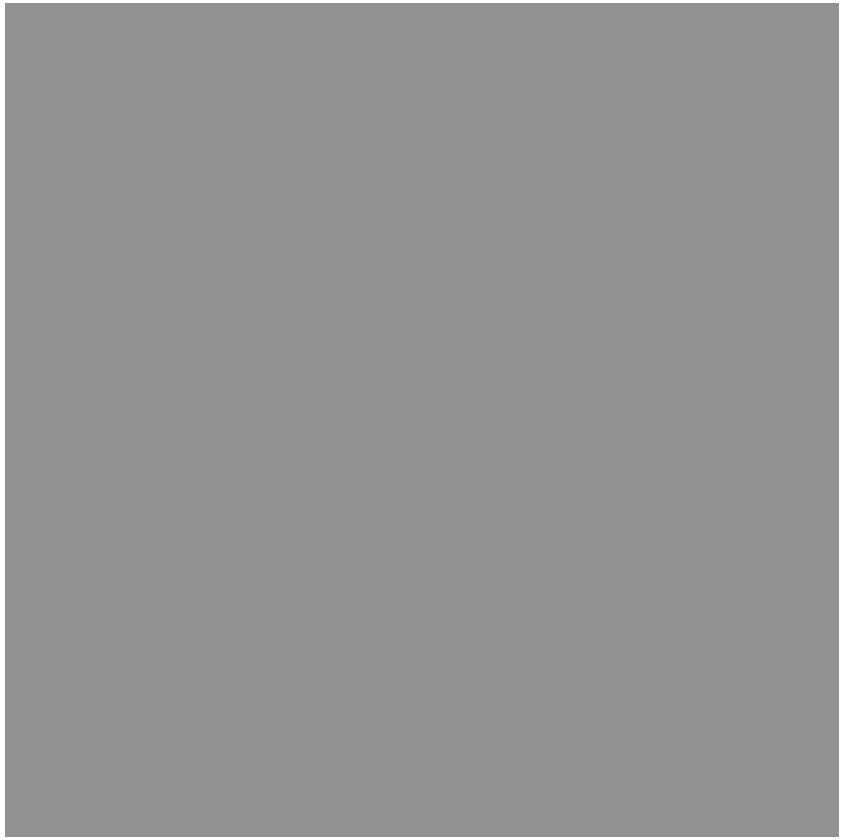
Extension to 2D



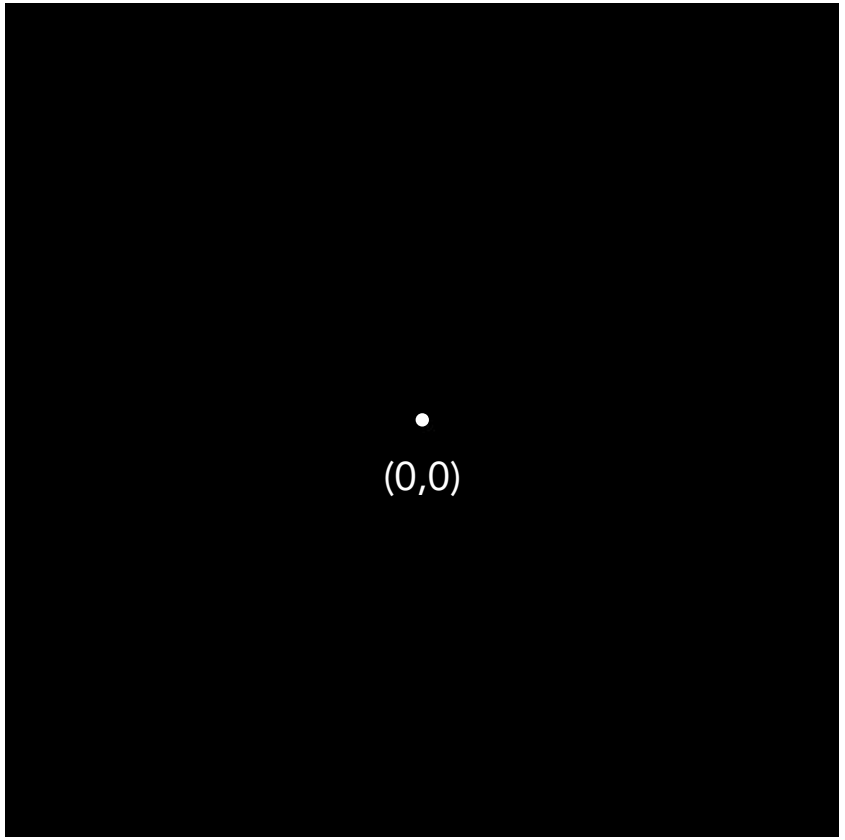
=

Image as a sum of basis images

Constant

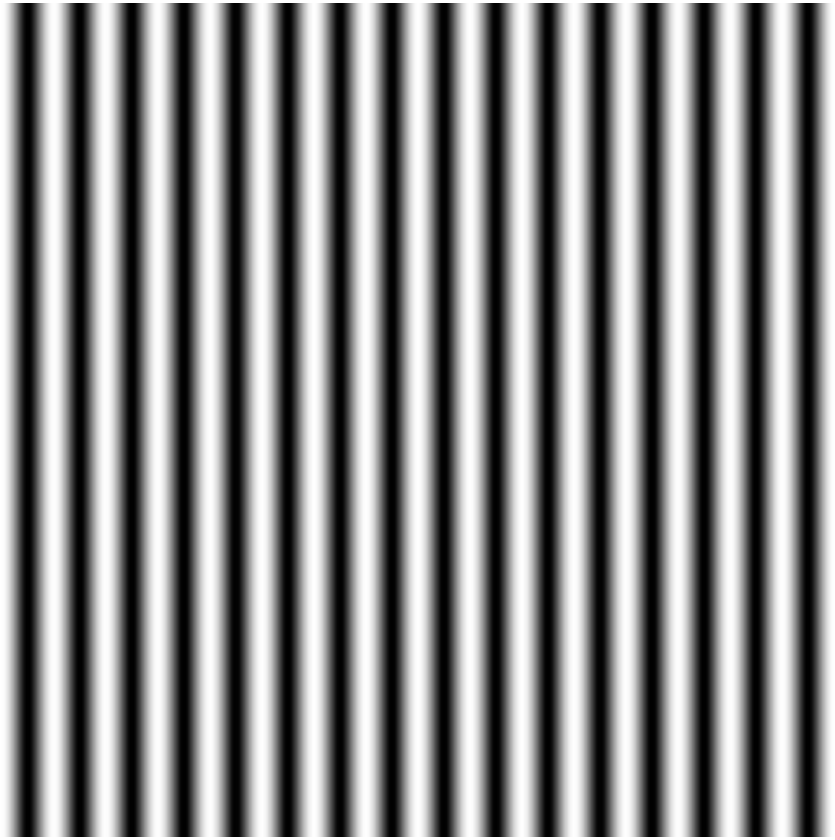


Spatial Domain

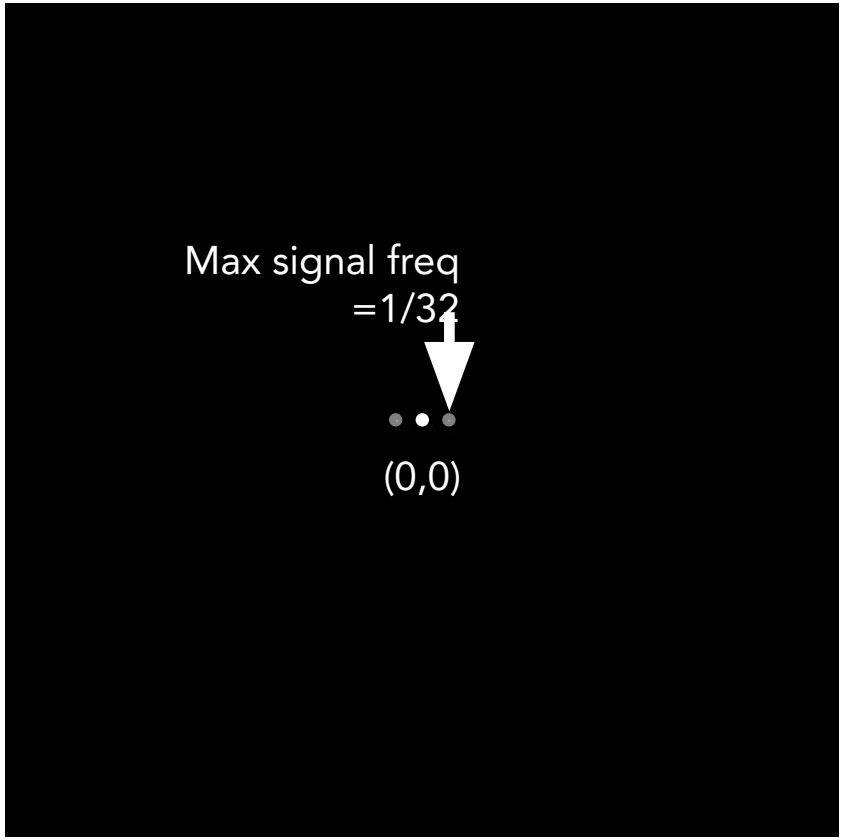


Frequency Domain

$\sin(2\pi/32)x$— frequency 1/32; 32 pixels per cycle

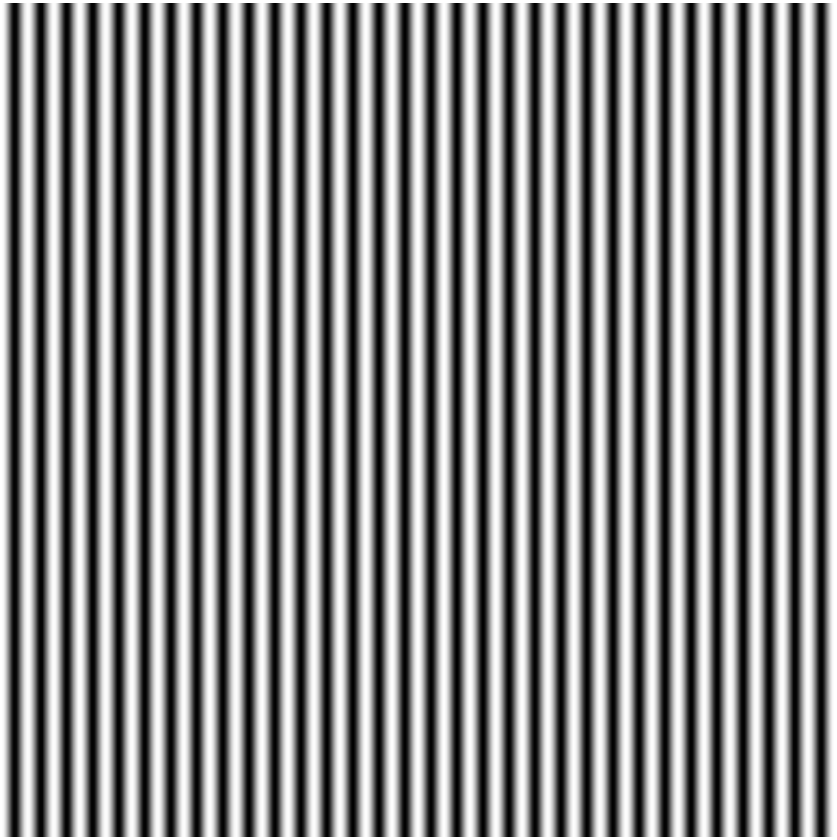


Spatial Domain

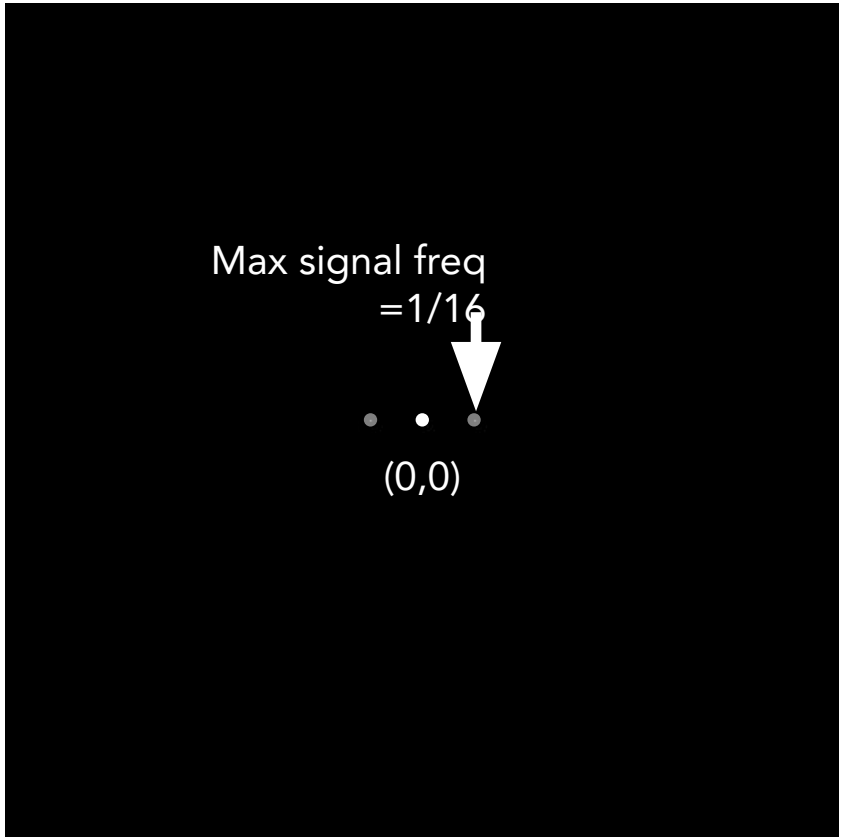


Frequency Domain

$\sin(2\pi/16)x$ — frequency 1/16; 16 pixels per cycle

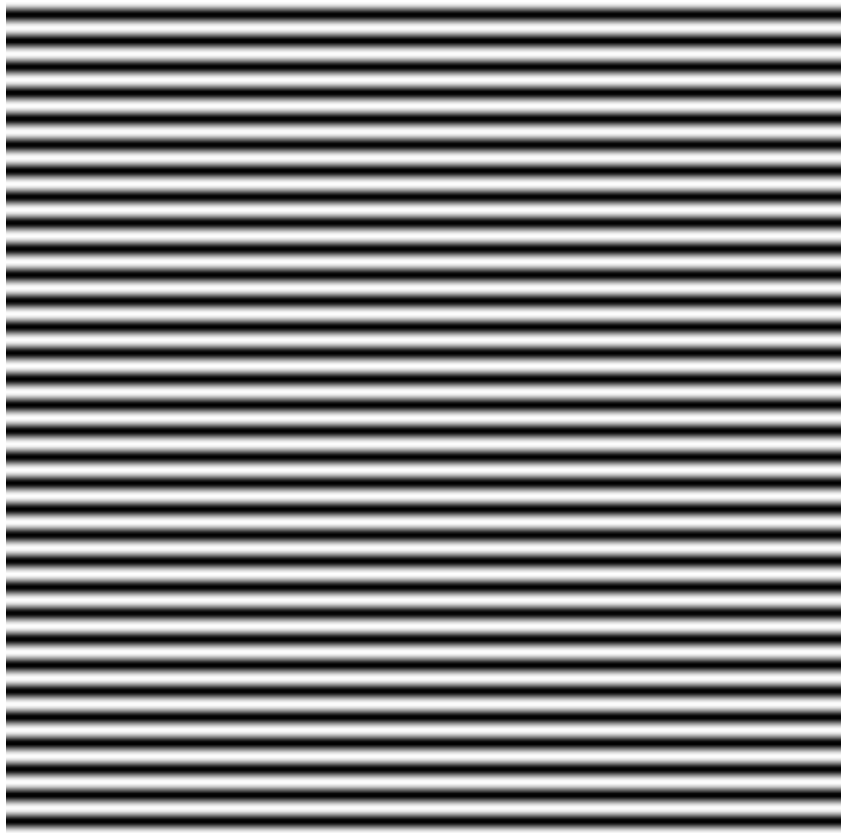


Spatial Domain

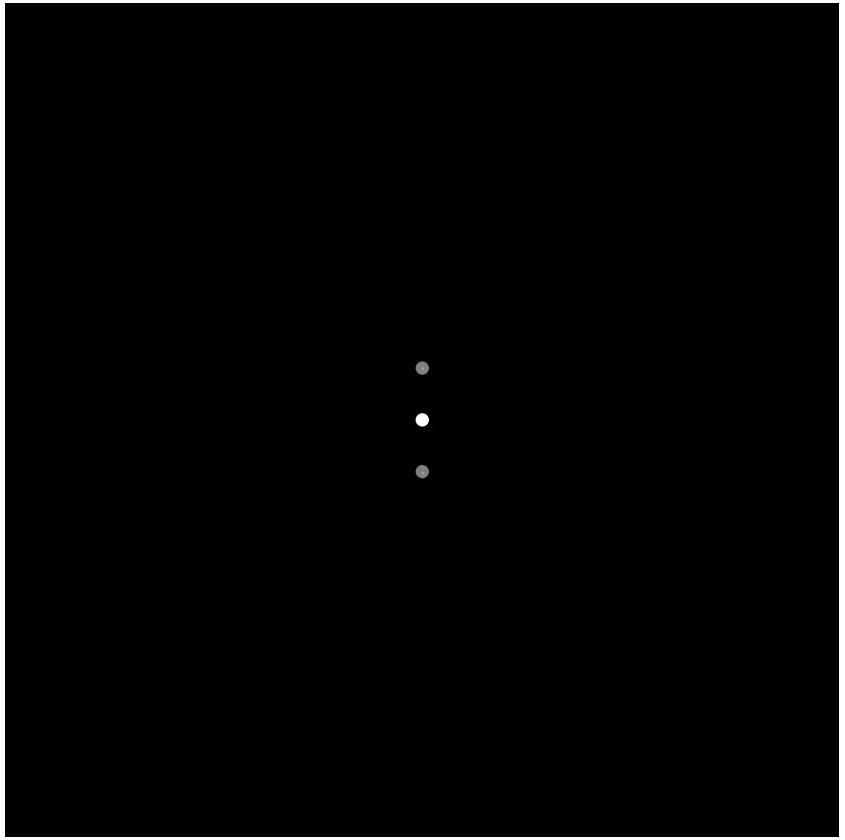


Frequency Domain

$$\sin(2\pi/16)y$$



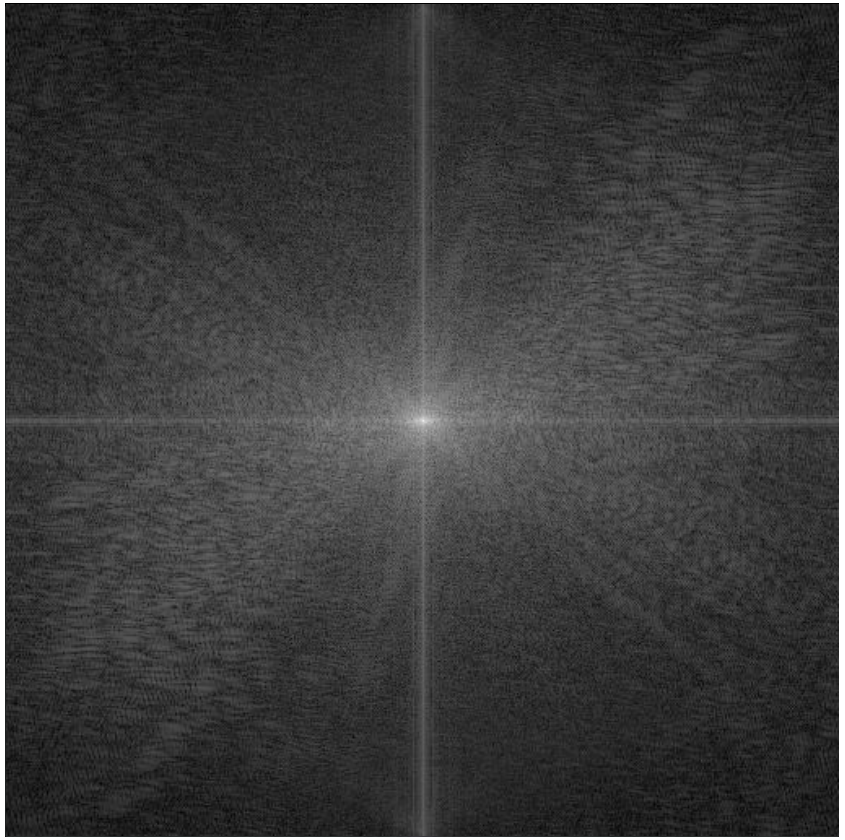
Spatial Domain



Frequency Domain

2D Frequency Space

Spatial Domain

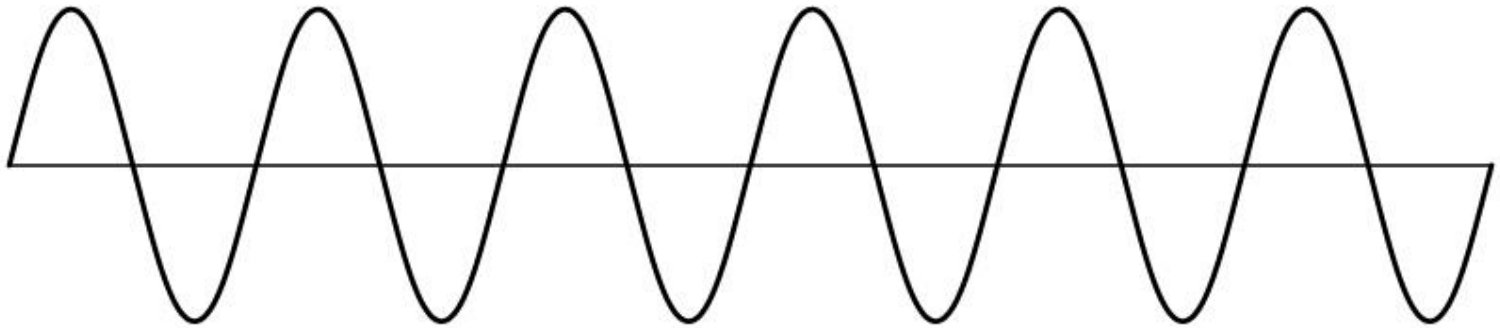


Frequency Domain

Note: Frequency domain also known as frequency space, Fourier domain, spectrum, …

Sampling

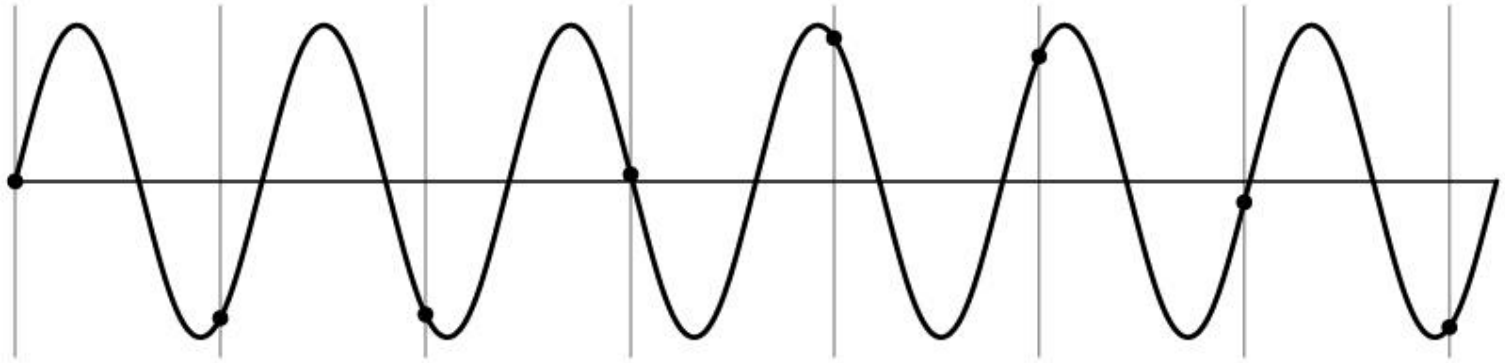
- Simple example: a sine wave



© 2006 Steve Marschner

Undersampling

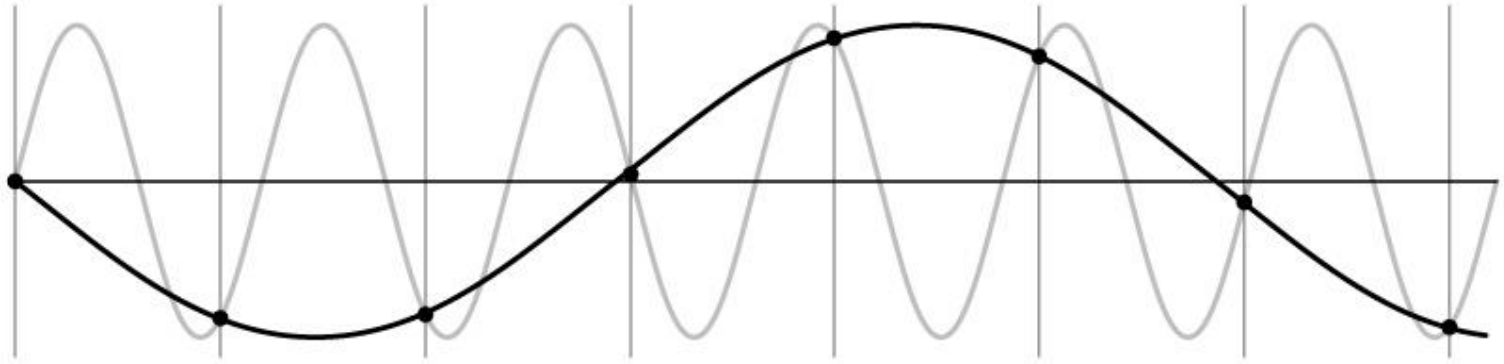
- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - Unsurprising result: information is lost



© 2006 Steve Marschner

Undersampling

- What if we "missed" things between the samples?
- Simple example: undersampling a sine wave
  - Unsurprising result: information is lost
  - Surprising result: indistinguishable from lower frequency
  - Aliasing: high frequencies traveling in disguise as low frequencies



© 2006 Steve Marschner

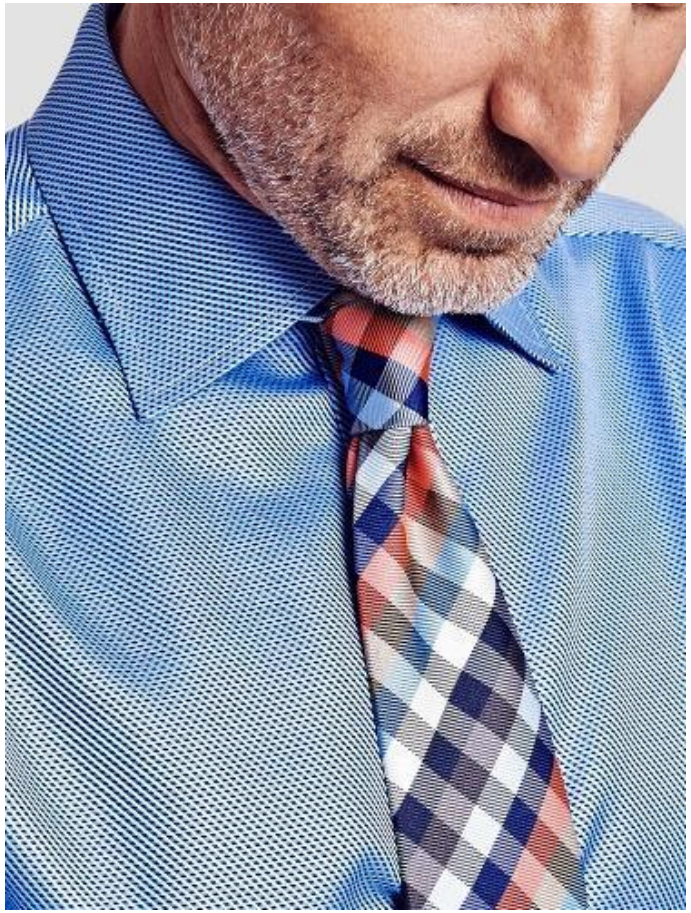
Aliasing in images

Original

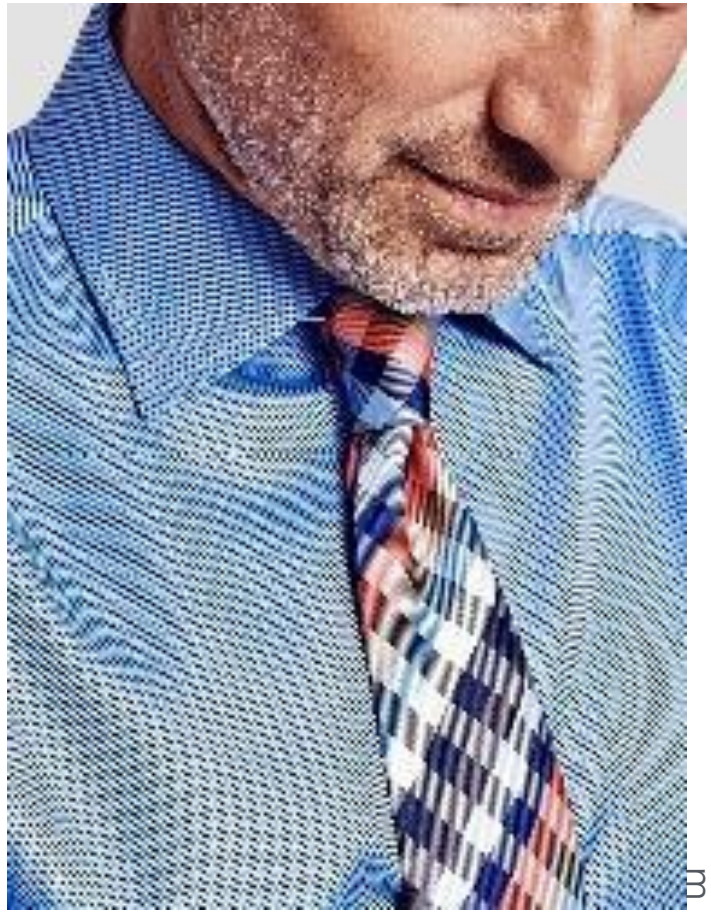
Aliased

Credit: Scientific Volume Imaging

Moiré Patterns in Imaging



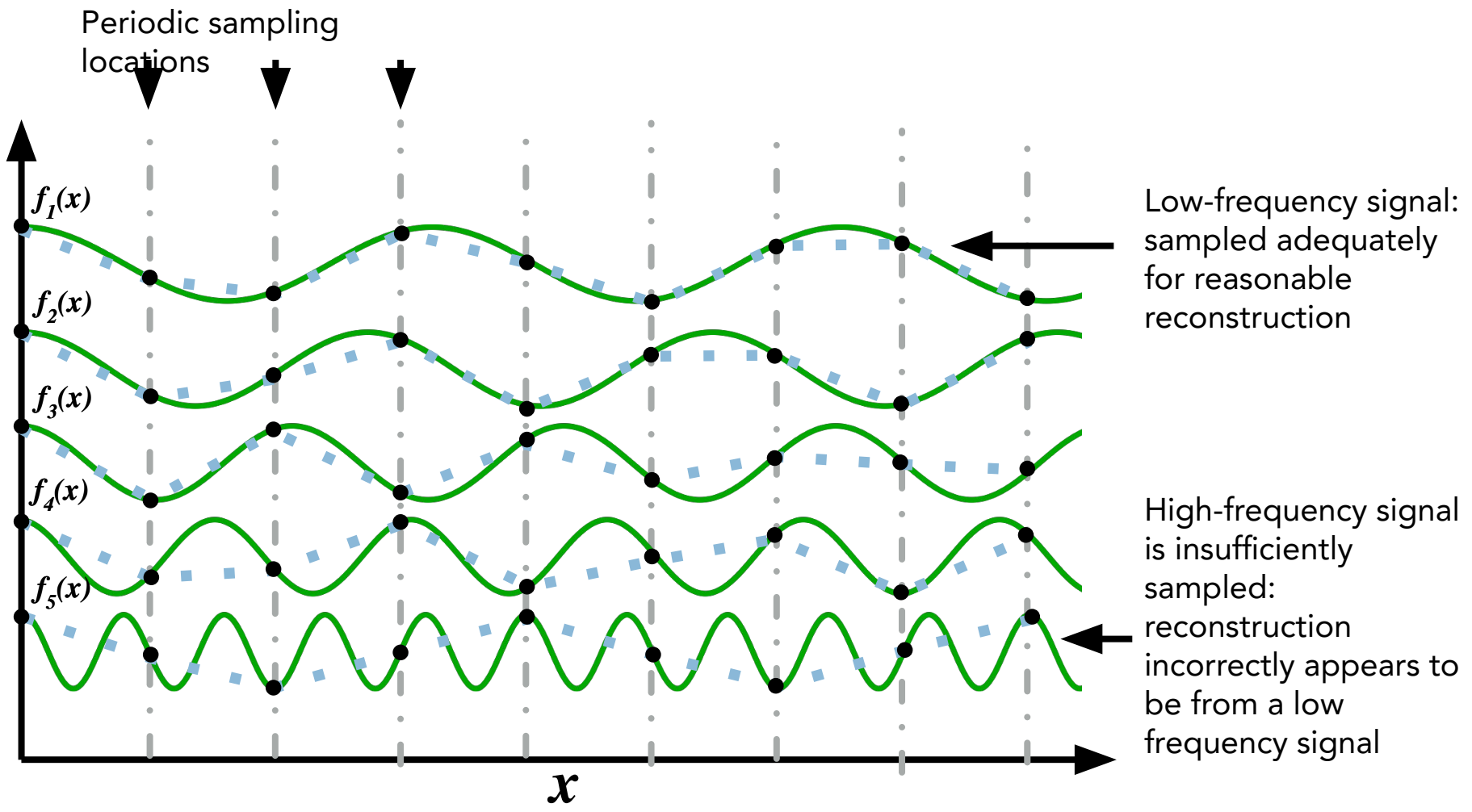
Read every sensor pixel



Skip odd rows and columns

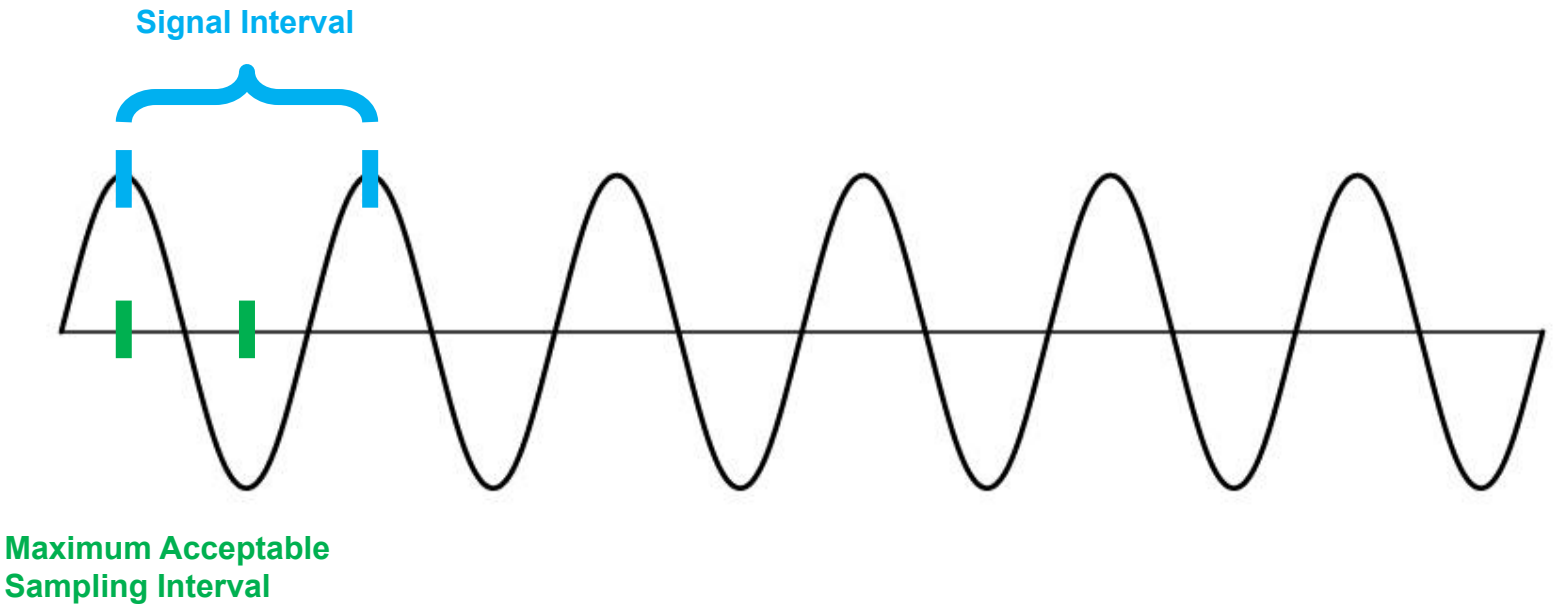
lystit.com

Higher Frequencies Need Faster Sampling



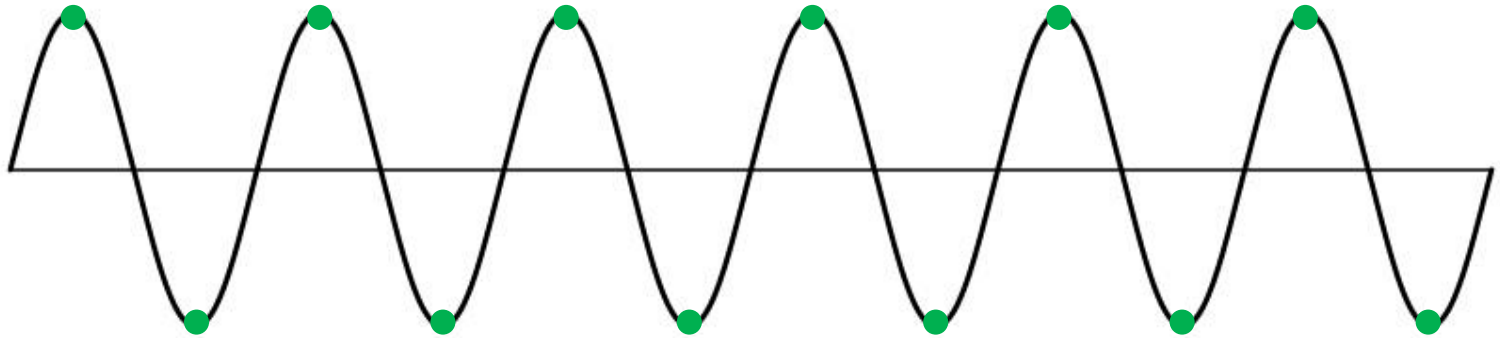
Nyquist Theorem

- Sampling rate should be equal to or greater than twice the highest frequency in the analog signal



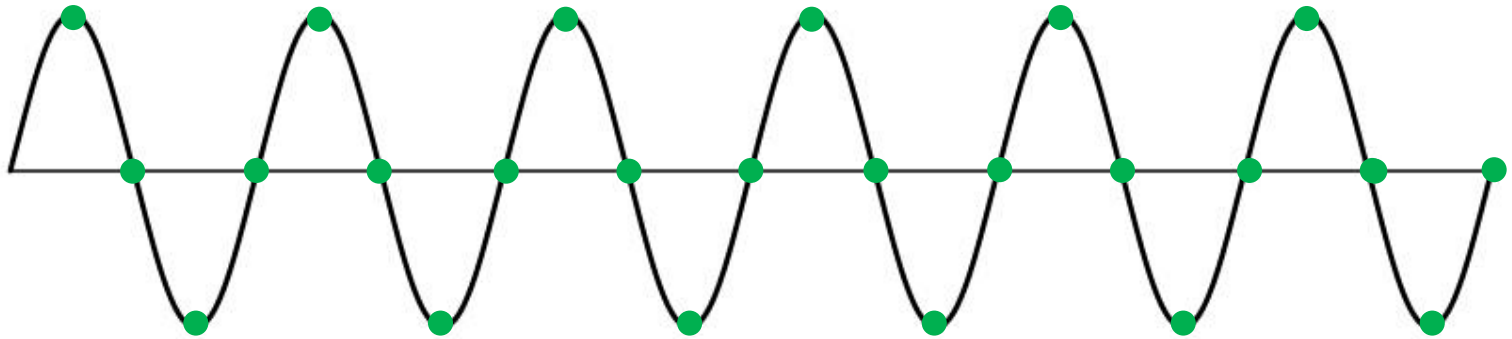
Nyquist Theorem

- Sampling rate should be equal to or greater than twice the highest frequency in the analog signal



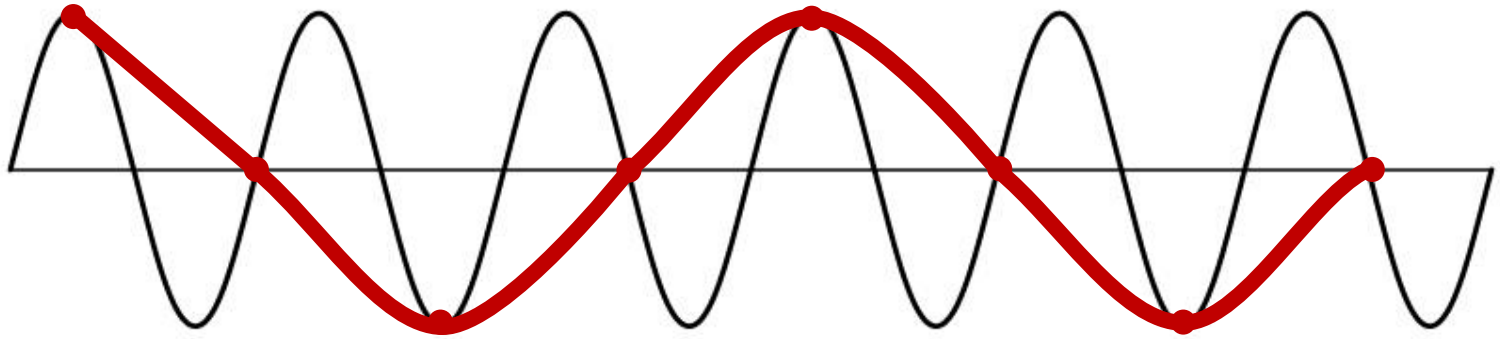
Nyquist Theorem

- Sampling rate should be equal to or greater than twice the highest frequency in the analog signal



Nyquist Theorem

- Sampling rate should be equal to or greater than twice the highest frequency in the analog signal



Antialiasing

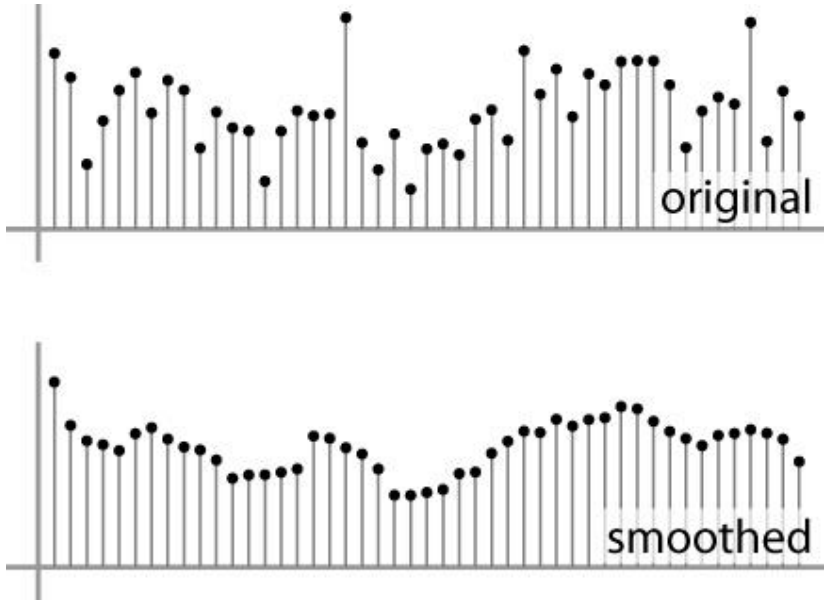
- Fast-changing signals sampled too slowly
- What can we do about aliasing?
  - Sample more often
    - Join the Megapixel craze of the photo industry
    - Only shifts the problem to higher frequencies
  - Make the signal less “wiggly” (filtering)
    - Get rid of some high frequencies
    - Will lose information
    - But it’s better than aliasing

Outline

- Sampling and aliasing
- Frequency domain
- Smoothing filters
- Antialiasing
- Other filters

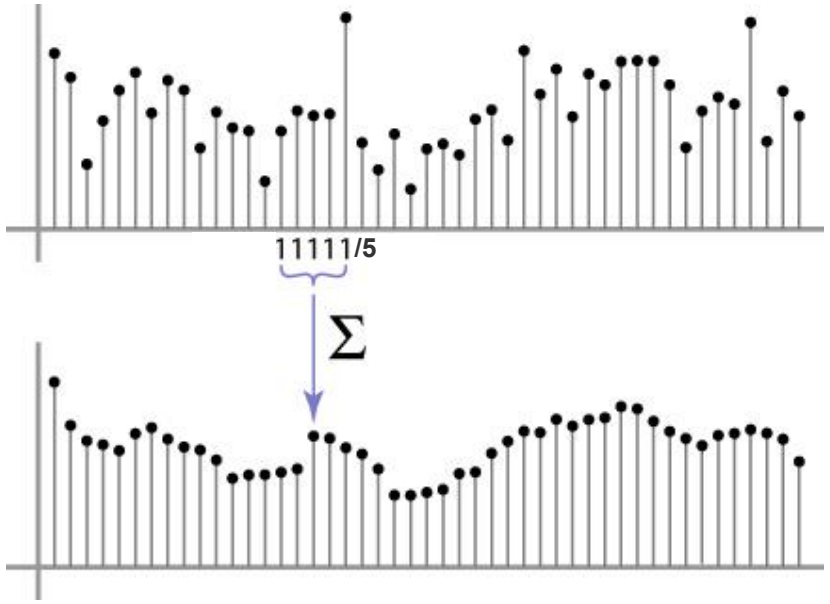
Moving Average

- **Basic idea: define a new function by averaging over a sliding window**



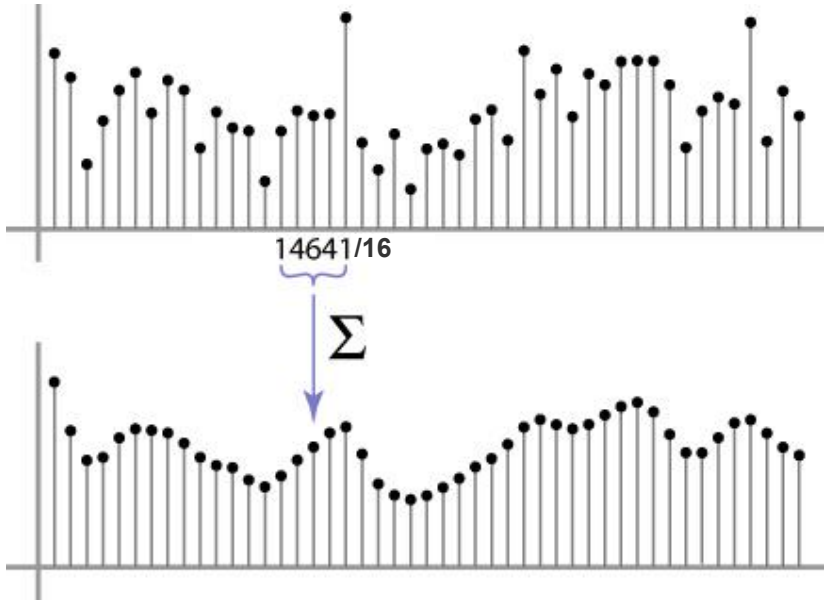
Weighted Moving Average

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



Weighted Moving Average

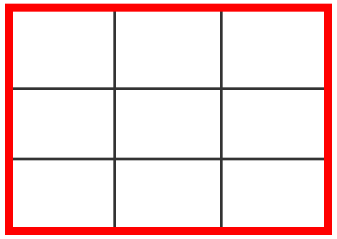
- Bell curve (gaussian-like) weights [1, 4, 6, 4, 1]/16



Weighted Moving Average In 2D

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

$F[x, y]$



$H[u, v]$

Slide by Steve Seitz

Cross-correlation

- Let F be the image, H be the kernel (of size 2*k*+1 x 2*k*+1), and G be the output image

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

- H is called “filter”, “kernel”, or “mask”
- This is called a cross-correlation operation:

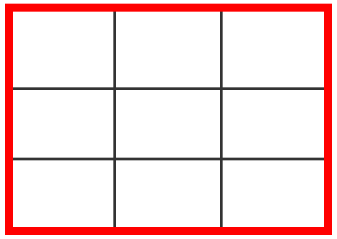
$$G = H \otimes F$$

Moving Average In 2D

- What are the weights 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

$F[x, y]$



$H[u, v]$

Image filtering

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

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

$g[.,.]$

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

Credit: S. Seitz

Image filtering

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

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

$g[.,.]$

	0	10							

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

Credit: S. Seitz

Image filtering

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

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

$g[.,.]$

	0	10	20						

Credit: S. Seitz

Image filtering

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

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

$g[.,.]$

	0	10	20	30					

Credit: S. Seitz

Image filtering

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

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

$g[.,.]$

	0	10	20	30	30				

Credit: S. Seitz

Image filtering

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

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

$g[.,.]$

	0	10	20	30	30				
				50					

Credit: S. Seitz

Image filtering

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

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

$g[.,.]$

	0	10	20	30	30				
						90			
				50					

Credit: S. Seitz

Image filtering

$h[\cdot,\cdot]$

$\frac{1}{9}$

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

$g[.,.]$

	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	

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

Credit: S. Seitz

Box filter

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

$$h[\cdot,\cdot]$$

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

Slide credit: David Lowe (UBC)

Cross-correlation vs. Convolution

- **Cross-correlation:**

$$G = H \otimes F$$

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

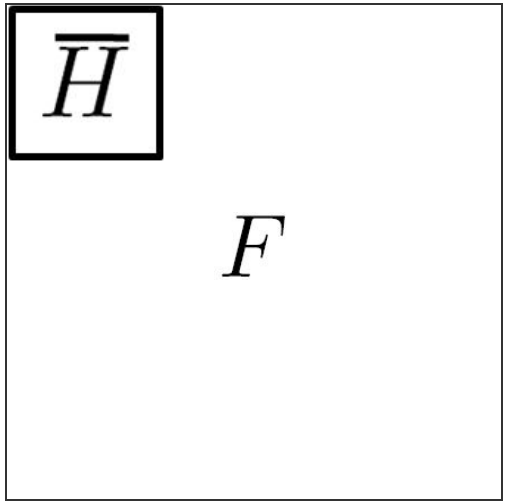
- **Convolution = cross-correlation with a horizontally and vertically flipped filter**

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

$$G = H \star F$$

Slide by Steve Seitz

Convolution



Adapted from F. Durand

Convolution is nice!

- Convolution is a multiplication-like operation
  - Commutative $a \star b = b \star a$
  - Associative $a \star (b \star c) = (a \star b) \star c$
  - Distributes over addition $a \star (b + c) = a \star b + a \star c$
  - Scalars factor out $\alpha a \star b = a \star \alpha b = \alpha(a \star b)$
  - Identity: unit impulse *e* = [..., 0, 0, 1, 0, 0, ...] $a \star e = a$
- Conceptually no distinction between filter and signal

Convolution is nice!

- Usefulness of associativity
  - often apply several filters one after another
    - $(((a * b_1) * b_2) * b_3) = a * (b_1 * b_2 * b_3)$

Gaussian filtering

- A Gaussian kernel gives less weight to pixels further from the center of the window

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

$F[x, y]$

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

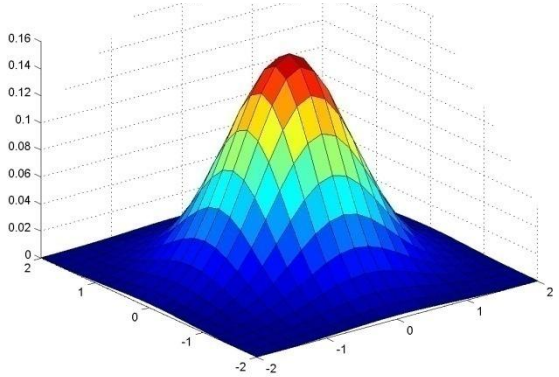
$H[u, v]$

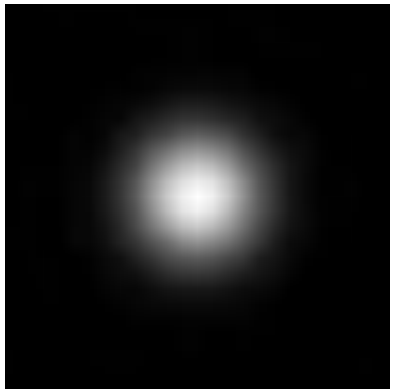
Slide by Steve Seitz

Gaussian filter

- This kernel is an approximation of a Gaussian function:

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





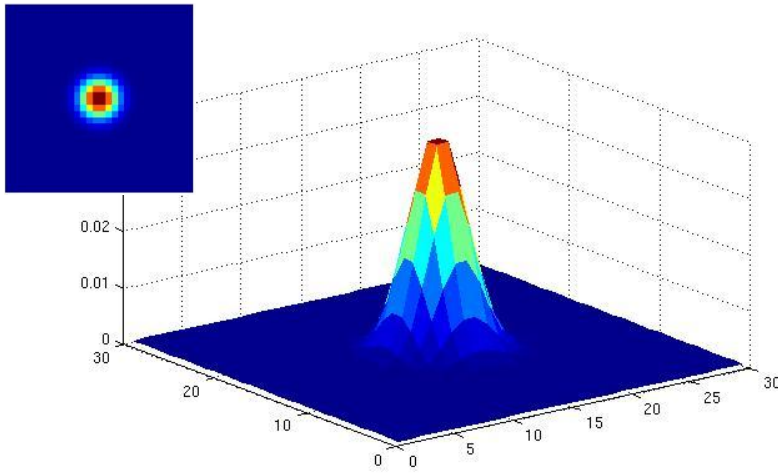
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, σ = 1

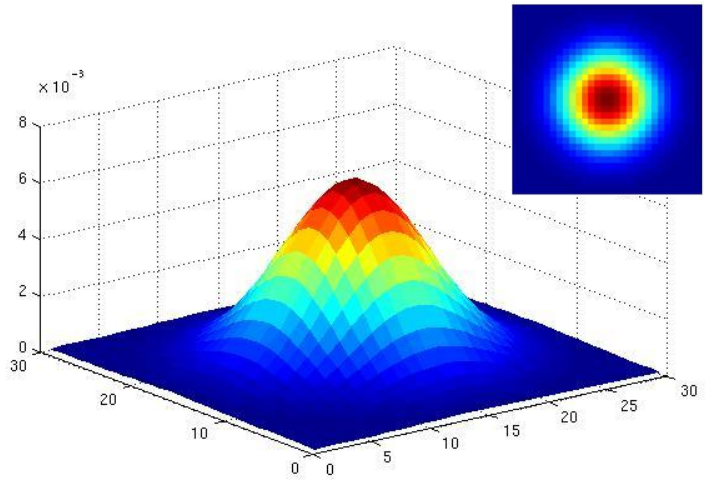
Slide credit: Christopher Rasmussen

Gaussian Kernel

- **Standard deviation σ: determines extent of smoothing**



σ = 2 with 30 x 30 kernel

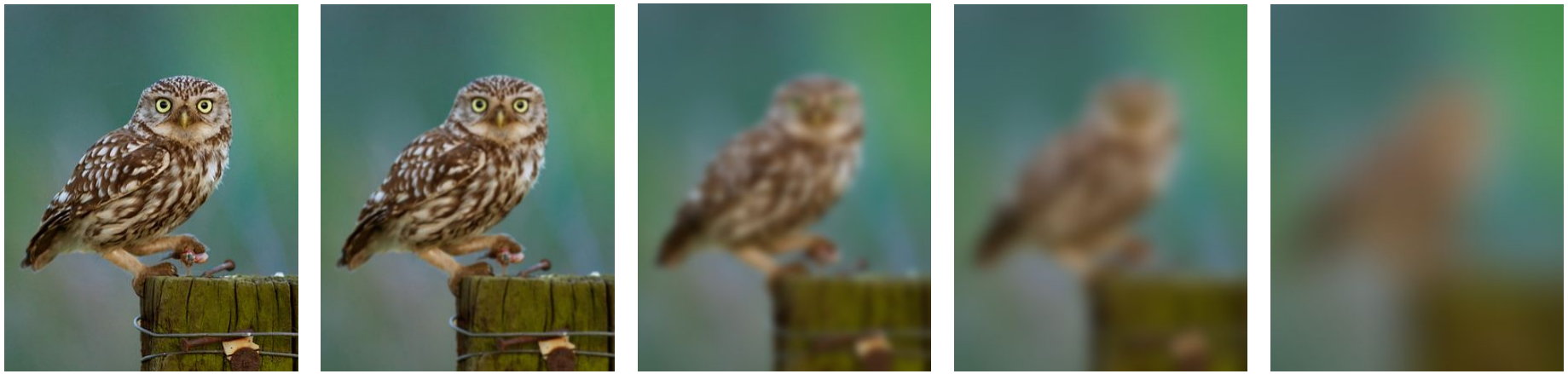


σ = 5 with 30 x 30 kernel

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

Source: K. Grauman

Gaussian filters



σ = 1 pixel

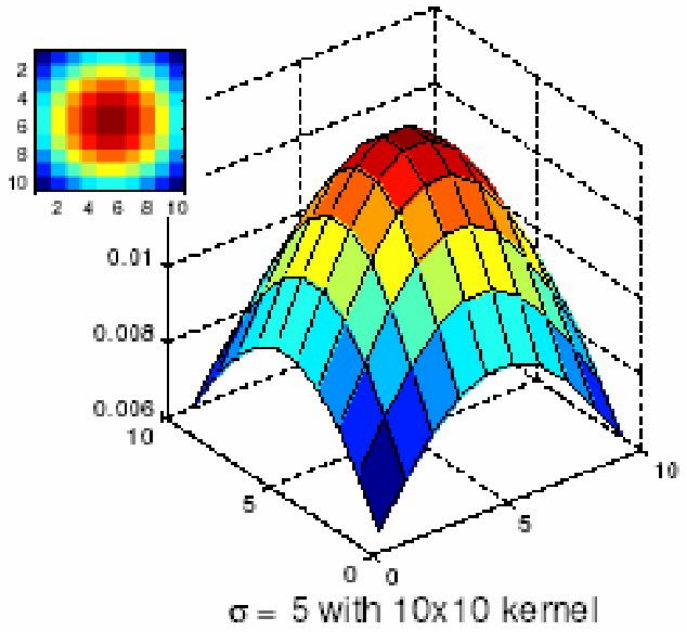
σ = 5 pixels

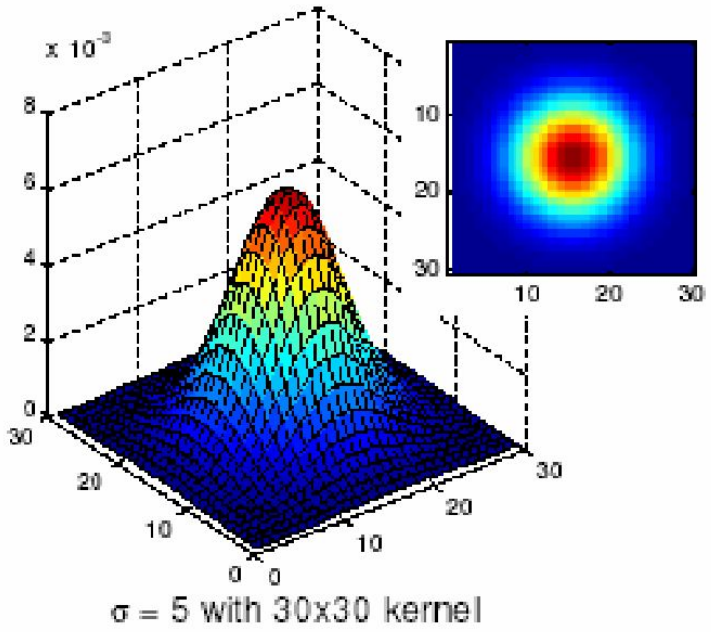
σ = 10 pixels

σ = 30 pixels

Choosing kernel width

- **The Gaussian function has infinite support, but discrete filters use finite kernels**

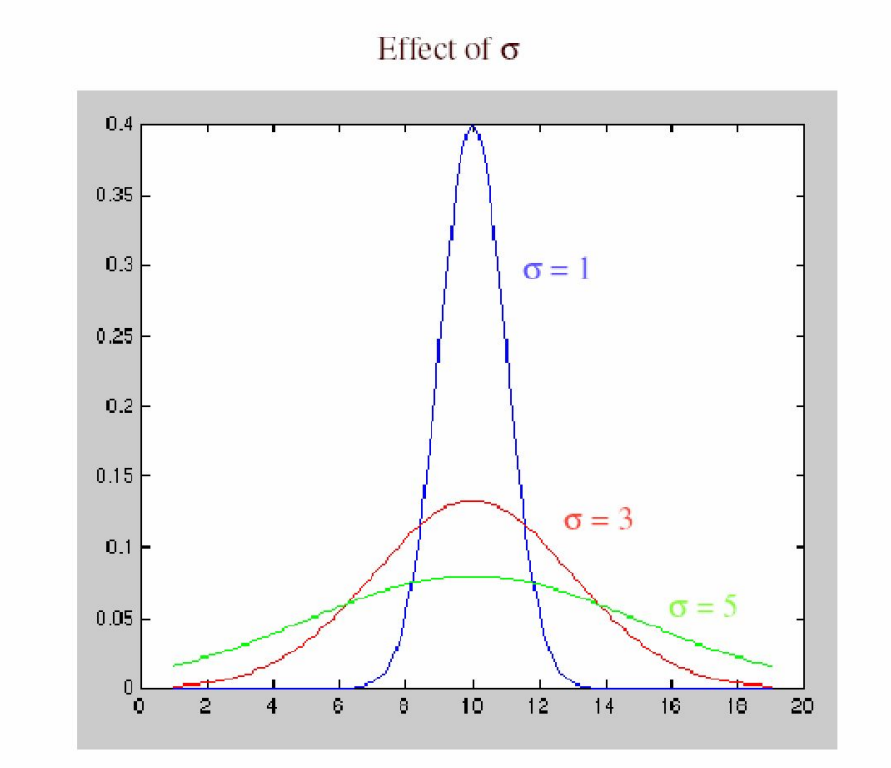




Source: K. Grauman

Practical matters

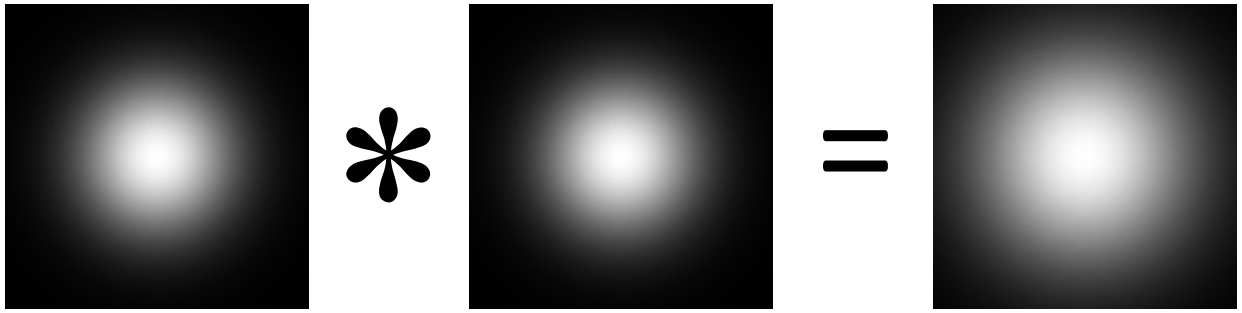
- How big should the filter be?
- Values at edges should be near zero
- Rule of thumb for Gaussian: set filter half-width to about 3 σ



Side by Derek Hoiem

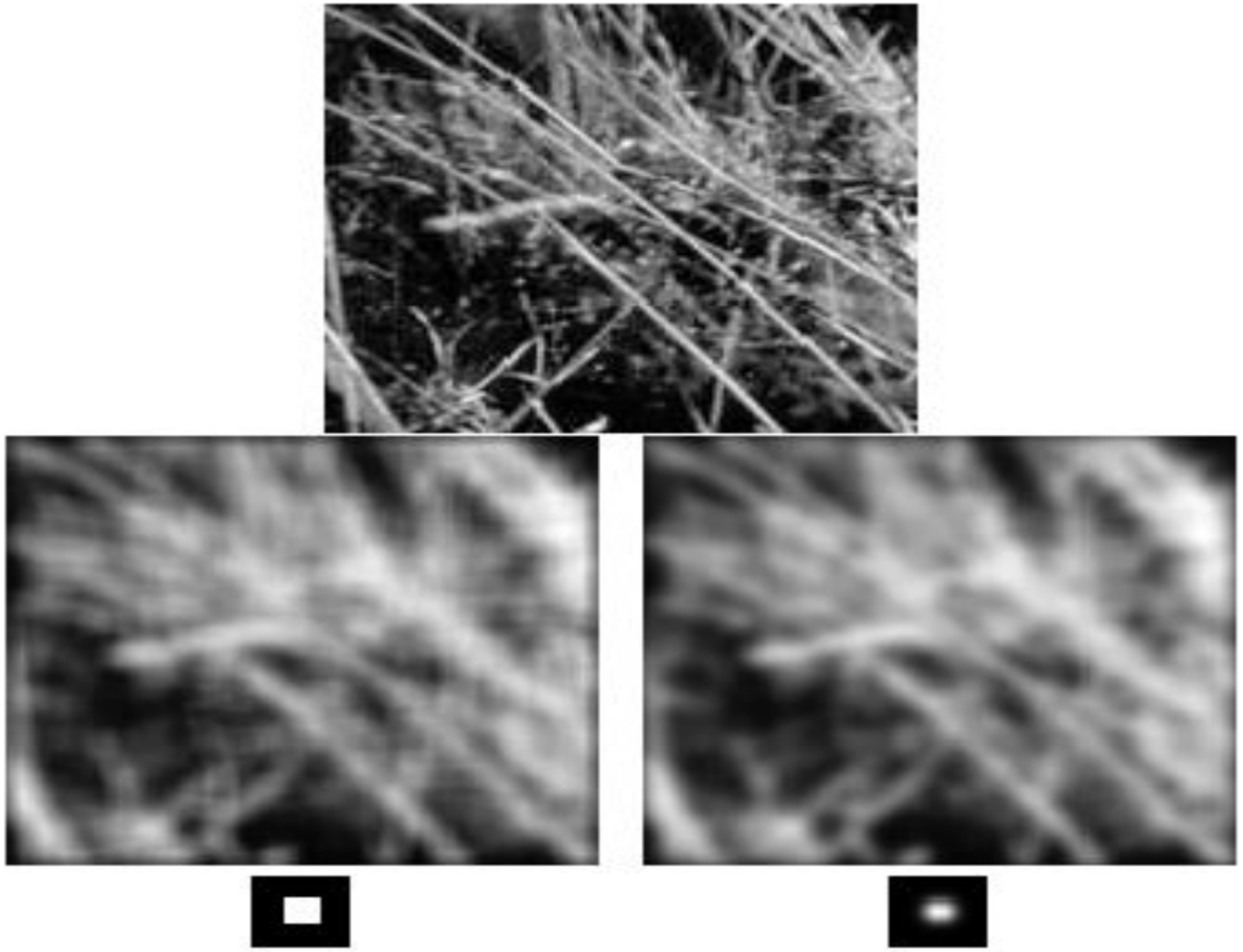
Gaussian and convolution

- Removes “high-frequency” components from the image (low-pass filter)
- Convolution with self is another Gaussian
  - Convolving twice with Gaussian kernel of std. σ is equal to convolving once with kernel of std. $\sigma\sqrt{2}$



Source: K. Grauman

Box vs. Gaussian filtering

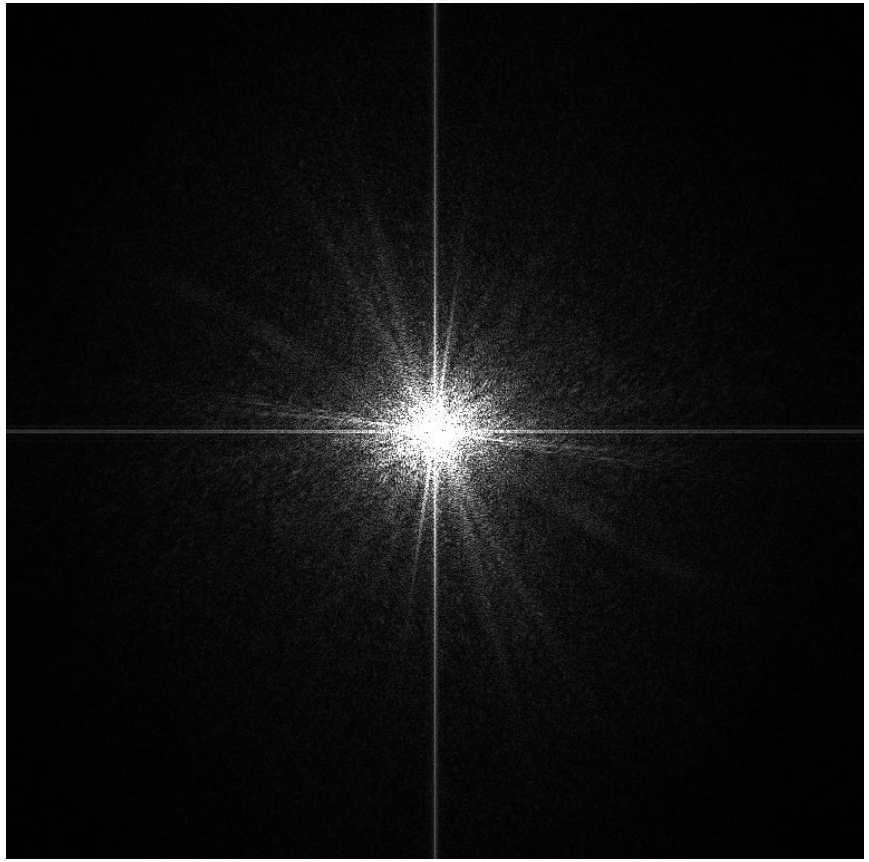


Slide by Steve Seitz

Filtering in Frequency Domain

Visualizing Image Frequency Content

Spatial Domain

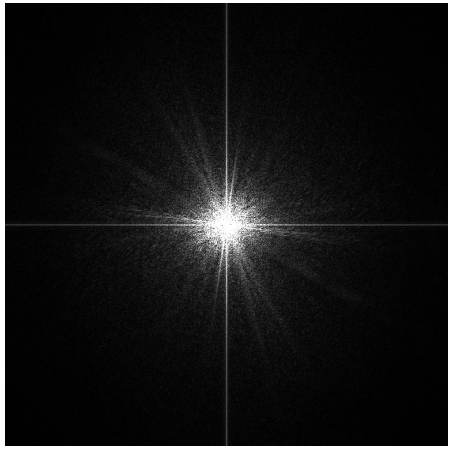


Frequency Domain

Filtering in the Frequency Domain

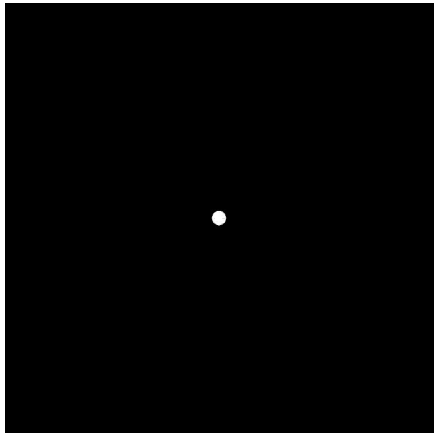
Spatial Domain

Fourier



Frequency Domain

Process



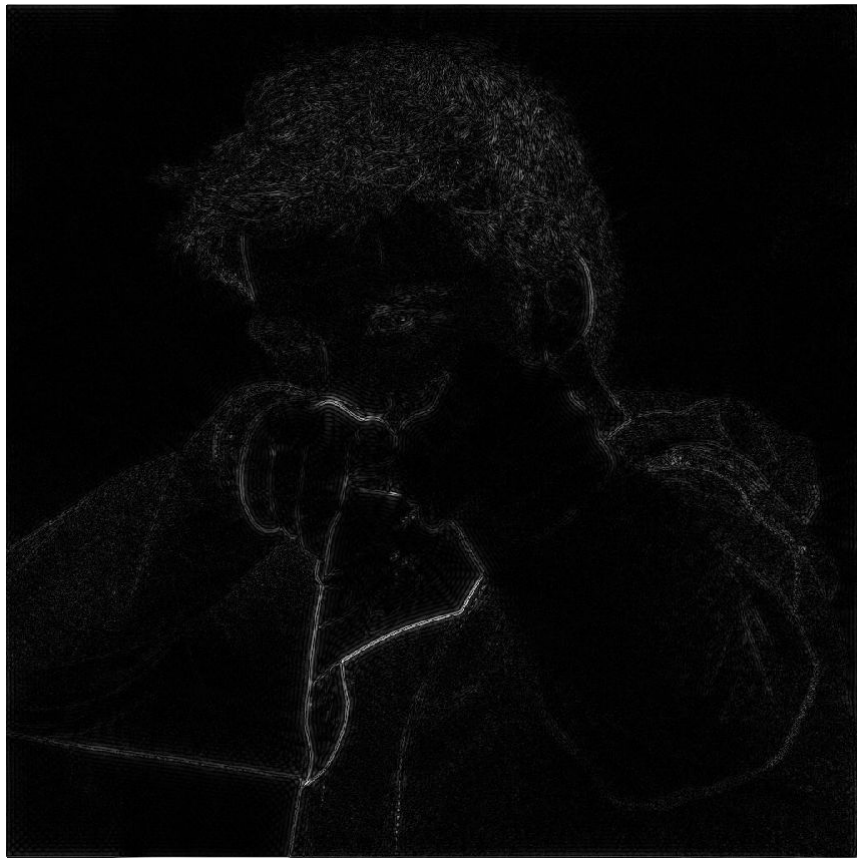
Frequency Domain

Inverse

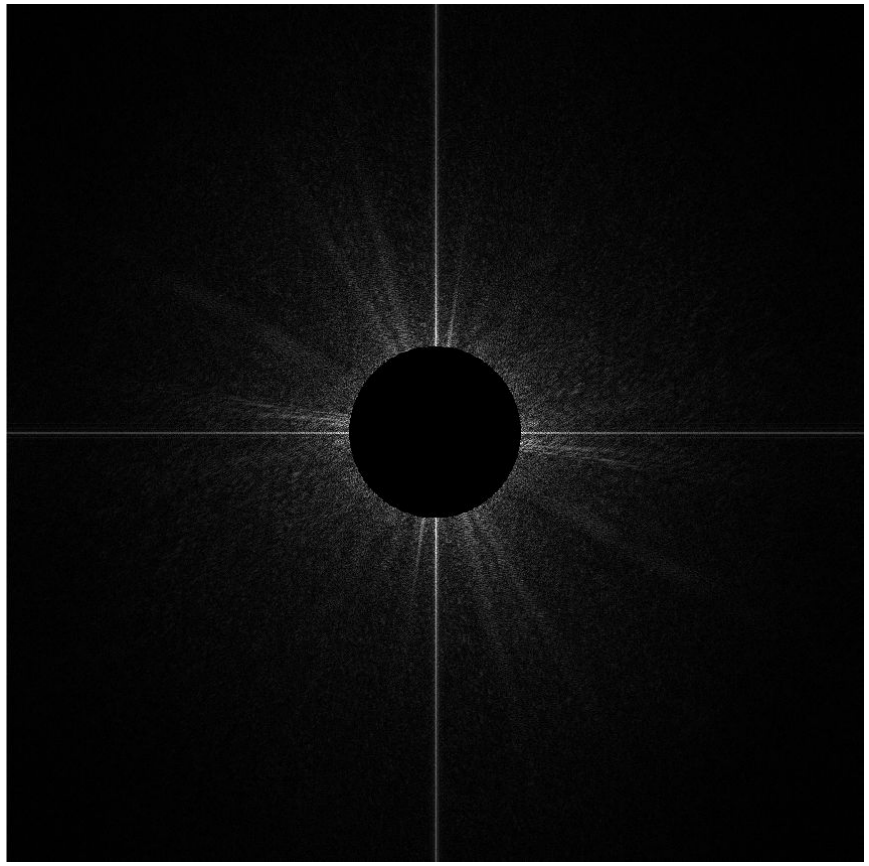
Fourier

Spatial Domain

Filter Out Low Frequencies Only (Edges)



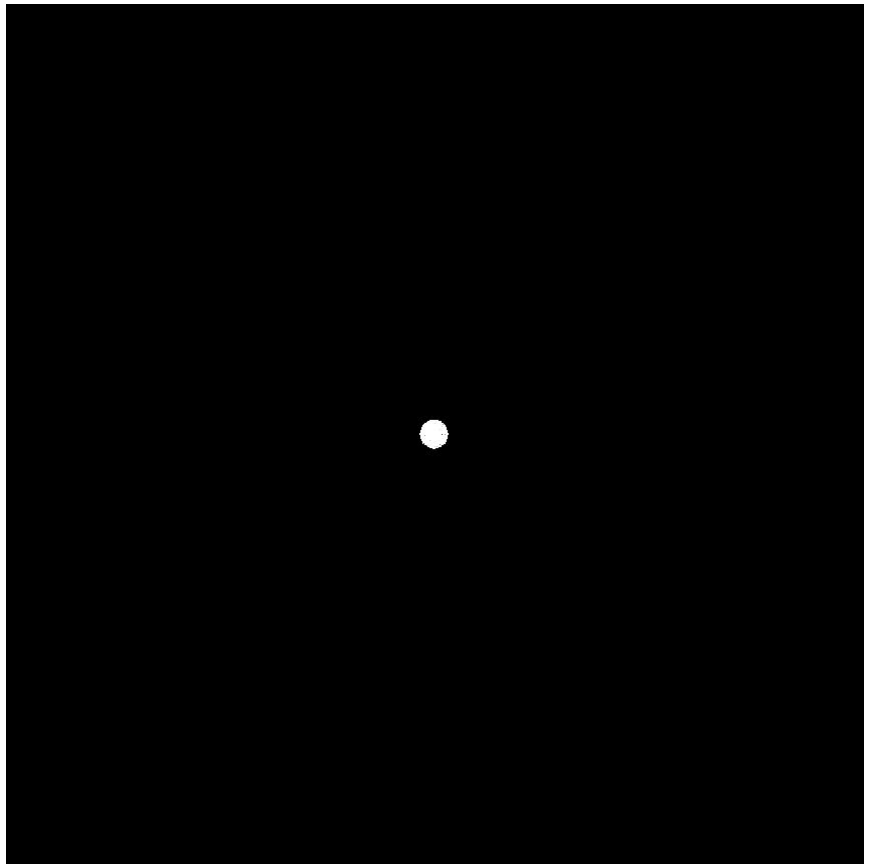
Spatial Domain



Frequency Domain

Filter Out High Frequencies (Blur)

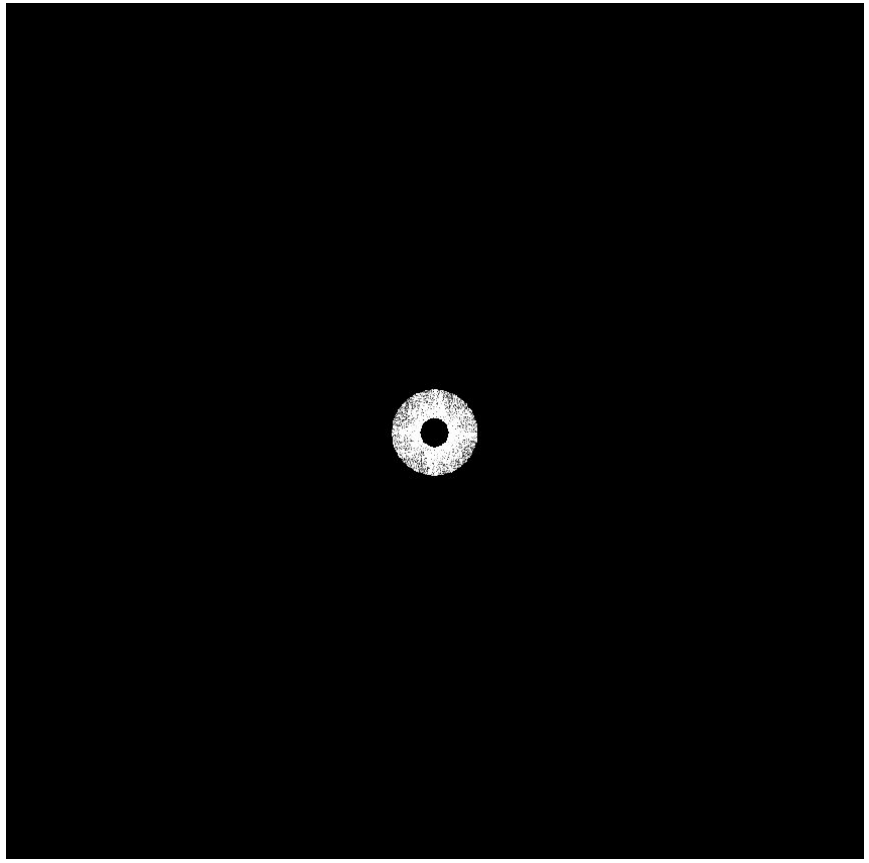
Spatial Domain



Frequency Domain

Filter Out Low and High Frequencies

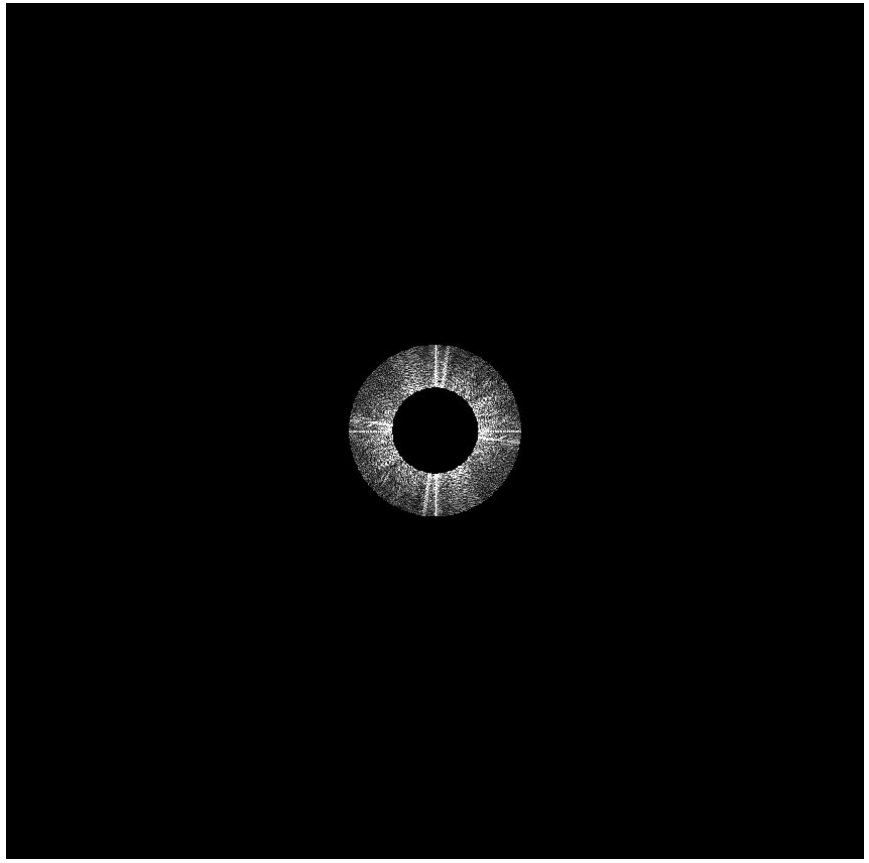
Spatial Domain



Frequency Domain

Filter Out Low and High Frequencies

Spatial Domain



Frequency Domain

Relationship between the two

The Convolution Theorem

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

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

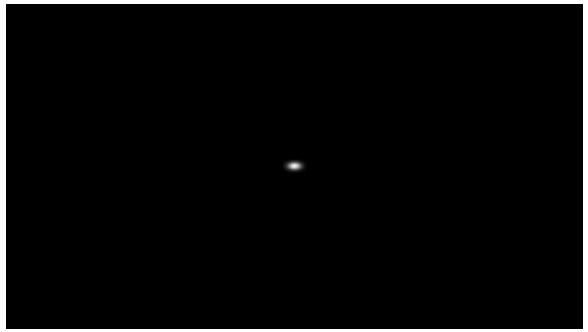
- Convolution in spatial domain is equivalent to multiplication in frequency domain!

2D convolution theorem example

$f(x,y)$

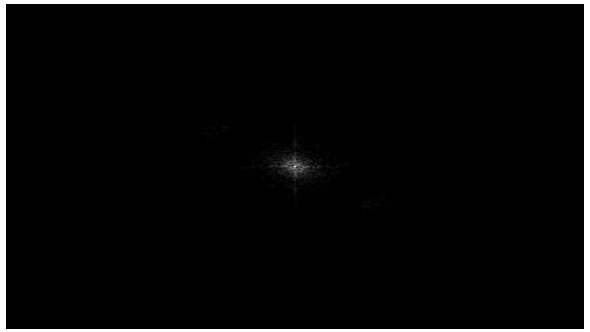
$*$

$h(x,y)$

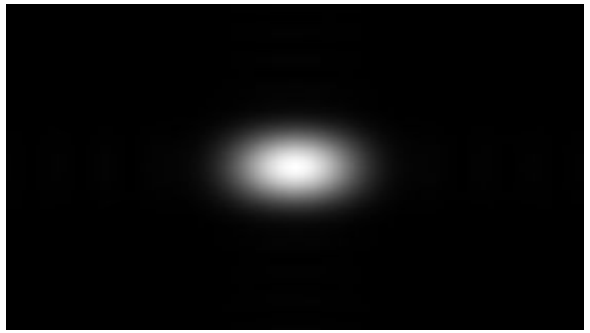


$\Downarrow$

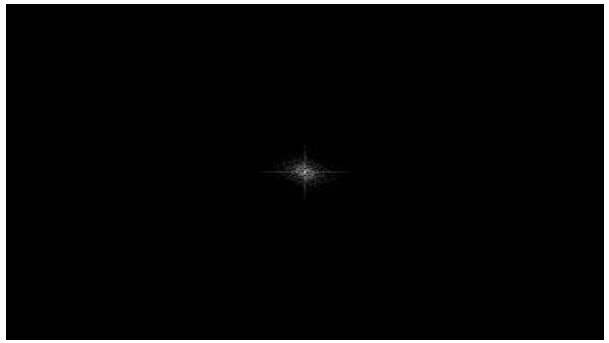
$g(x,y)$



$\times$



$\Downarrow$



$|F(s_x,s_y)|$

$|H(s_x,s_y)|$

$|G(s_x,s_y)|$

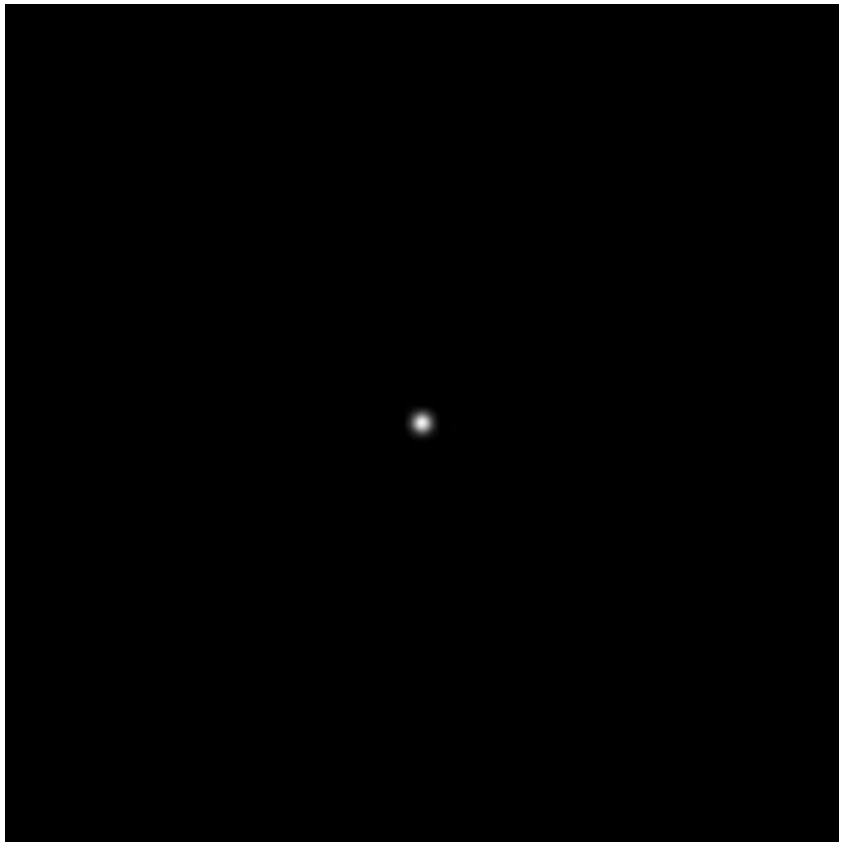
Box vs. Gaussian filtering

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

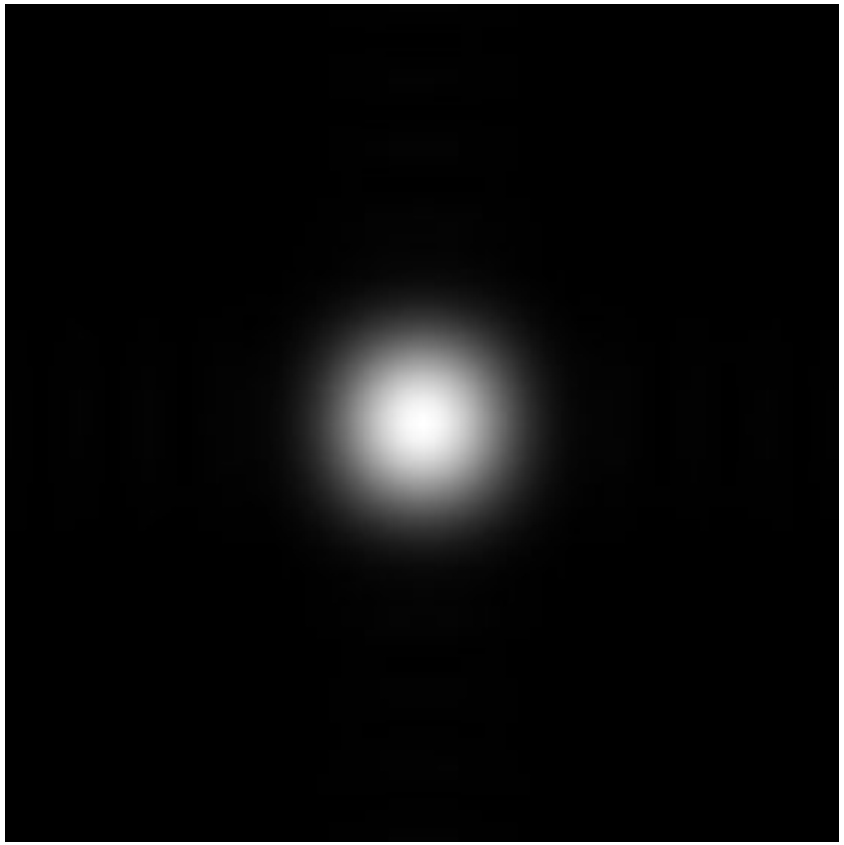
Gaussian

Box filter

Gaussian filter

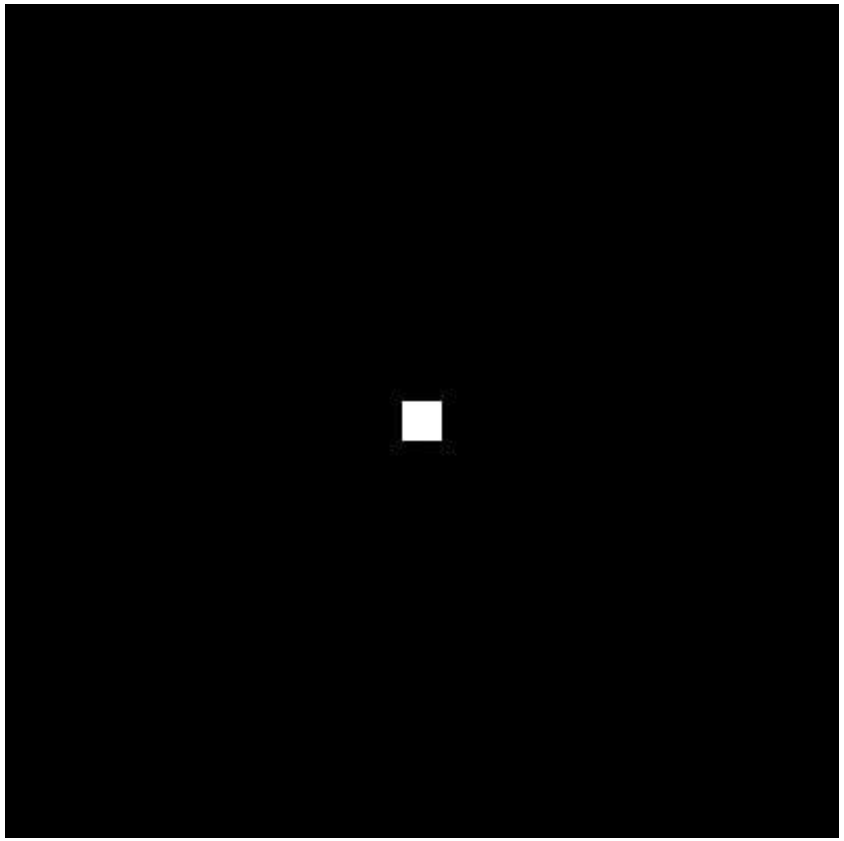


Spatial
Domain

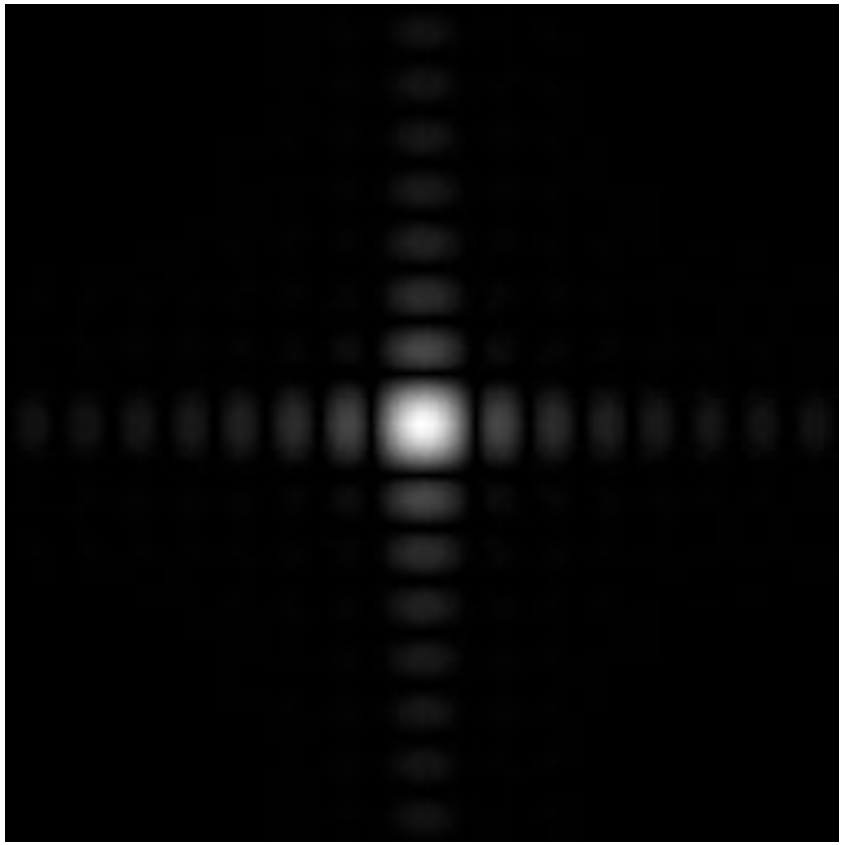


Frequency
Domain

Box filter



Spatial
Domain

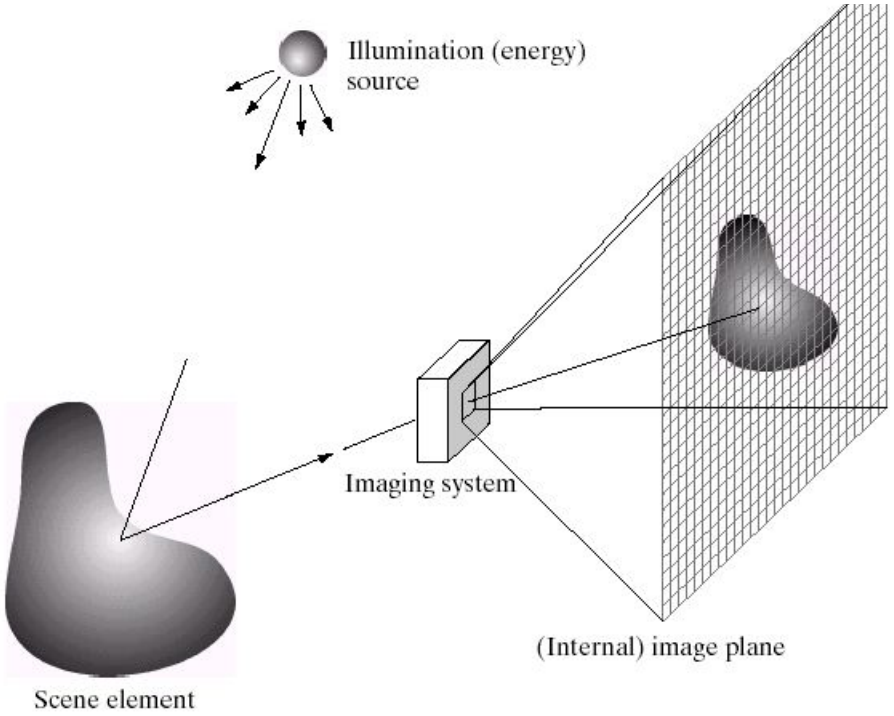


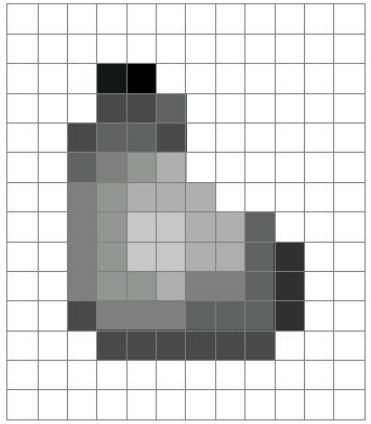
Frequency
Domain

Outline

- Sampling and aliasing
- Frequency domain
- Smoothing filters
- Antialiasing
- Other filters

Image formation involves sampling





Antialiasing

- Option 1
  - Filter the scene
  - Sample the filtered scene at each pixel
- Option 2
  - Average the scene under each pixel

Digital sensors

- Fill factor: ratio of light sensitive area to total pixel area

- Pixel area
- Light sensitive area

Fill factor effect

4%

100%

Outline

- Sampling and aliasing
- Frequency domain
- Smoothing filters
- Antialiasing
- Other filters

Linear filters: examples

Original

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

Blur
(with a mean filter)

Source: D. Lowe

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

Linear filters: examples

Original

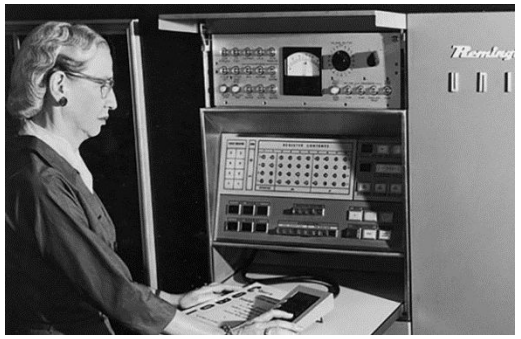
$$\left(\begin{matrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{matrix} - \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \right) =$$

Sharpening filter
(accentuates edges)

Source: D. Lowe

Filtering – sharpening

Image

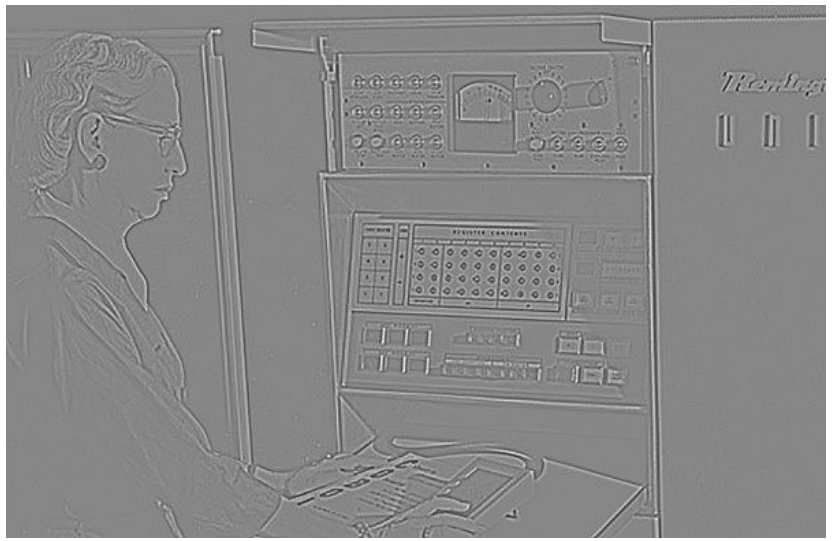


-

Smoothed

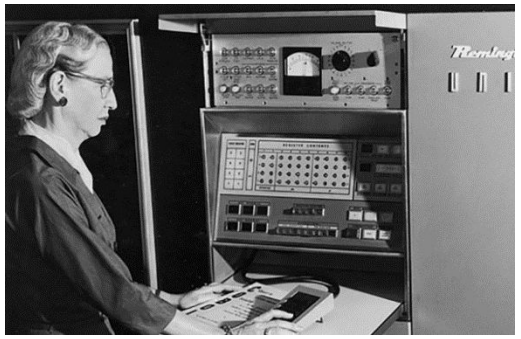
=

Details



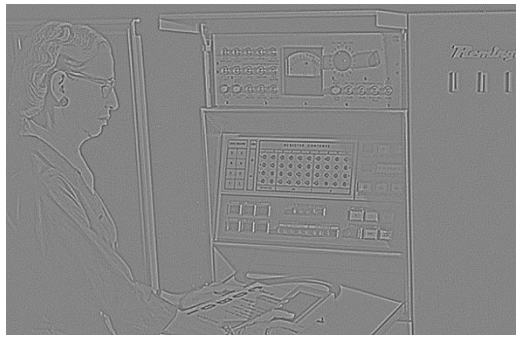
Filtering – Sharpening

Image



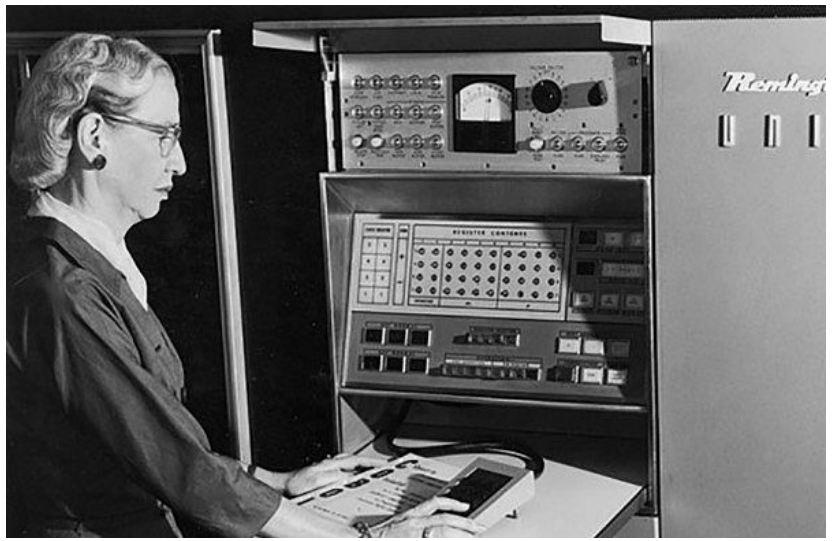
+α

Details



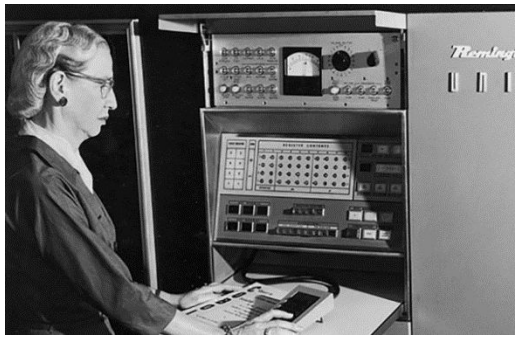
“Sharpened” α=1

=



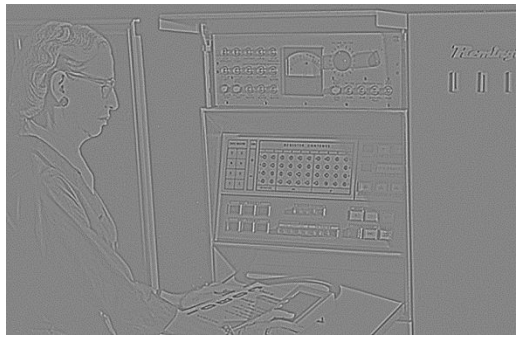
Filtering – Sharpening

Image



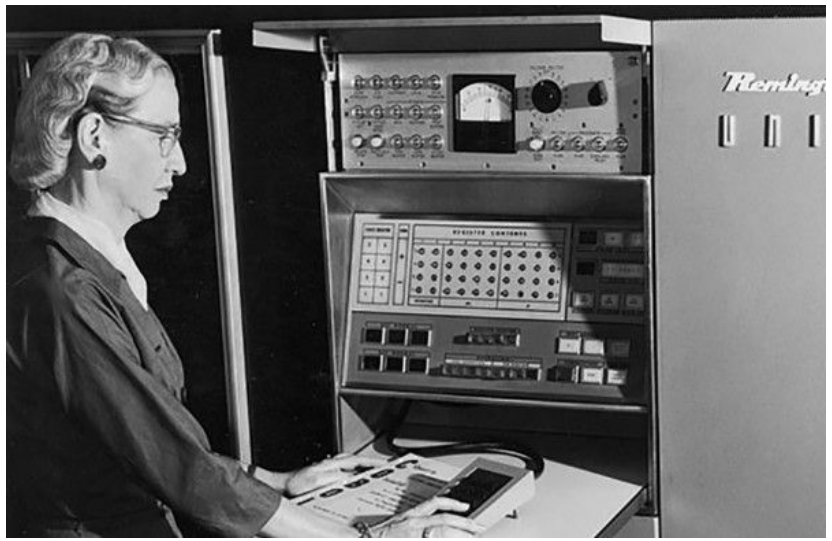
+α

Details



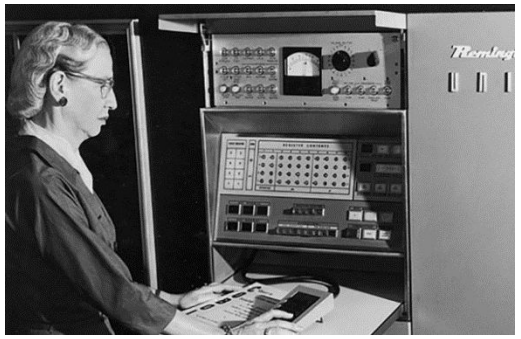
“Sharpened” α=0

=



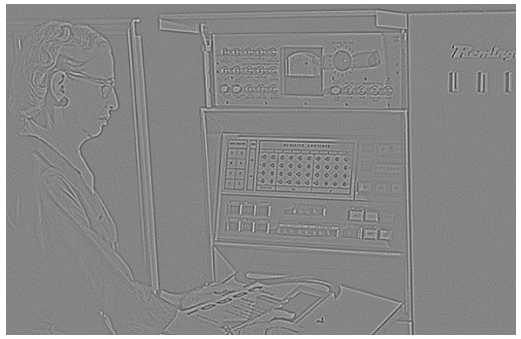
Filtering – Sharpening

Image



+α

Details

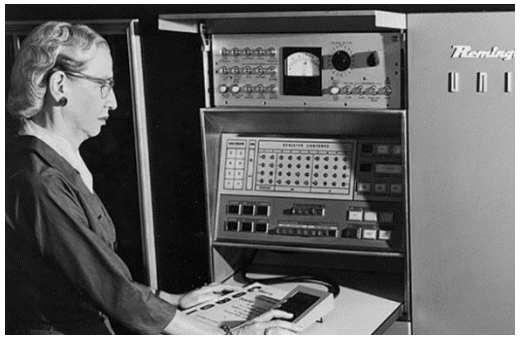


“Sharpened” α=2

=

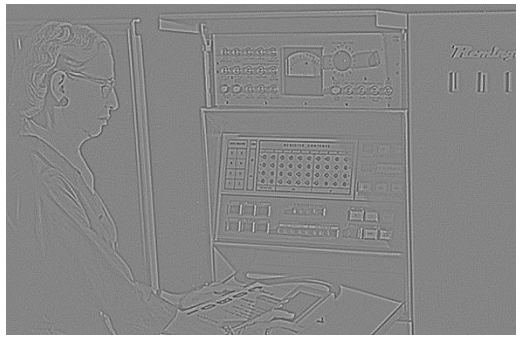
Filtering – Sharpening

Image



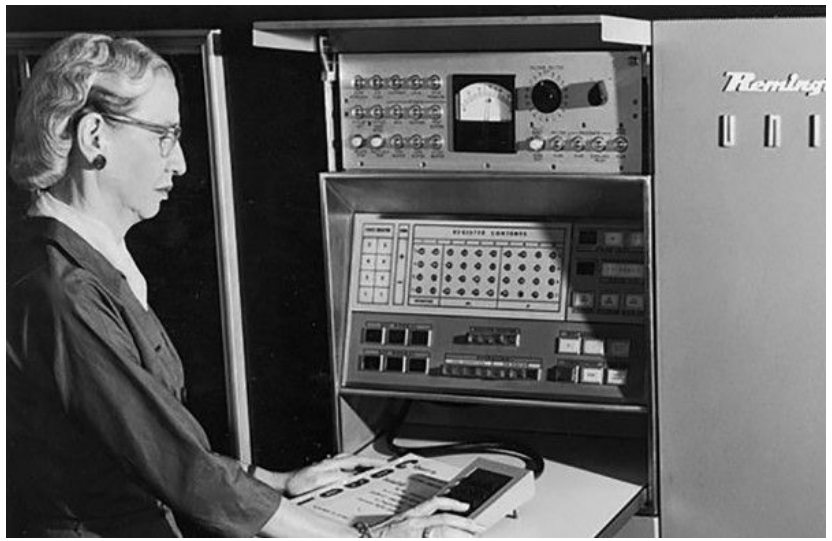
+α

Details



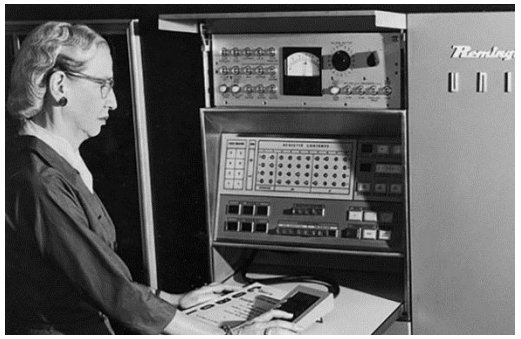
“Sharpened” α=0

=



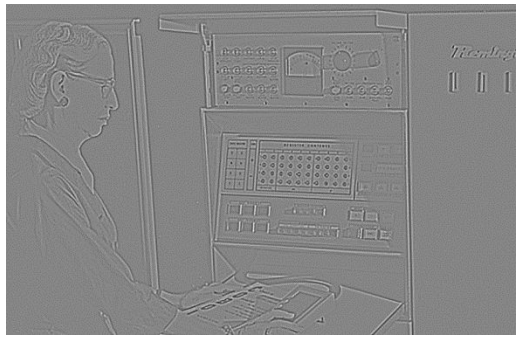
Filtering – Extreme Sharpening

Image



+α

Details



“Sharpened” α=10

=

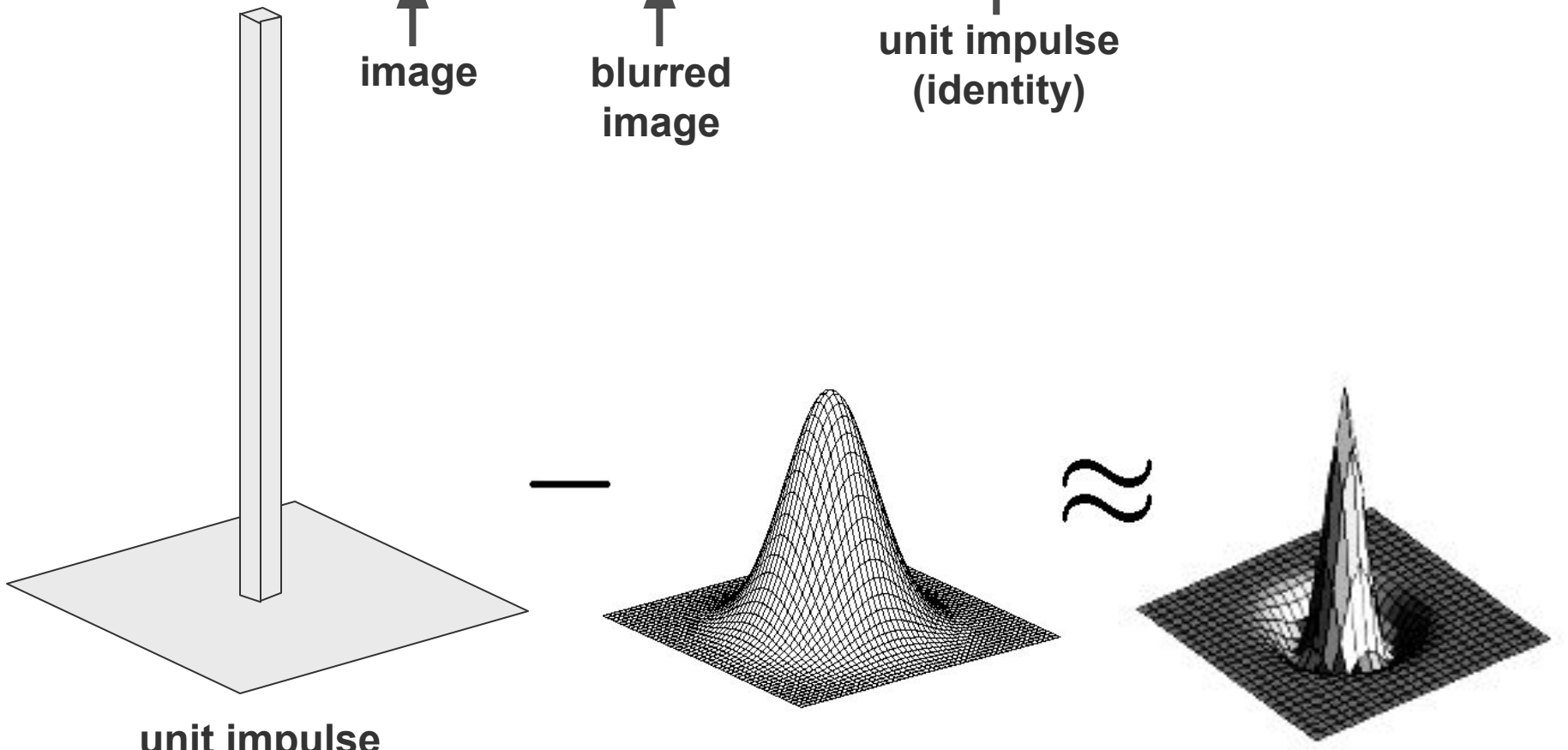
Unsharp mask filter

$$f + \alpha(f - f * g) = f + \alpha f * (e - g)$$

image

blurred image

unit impulse (identity)



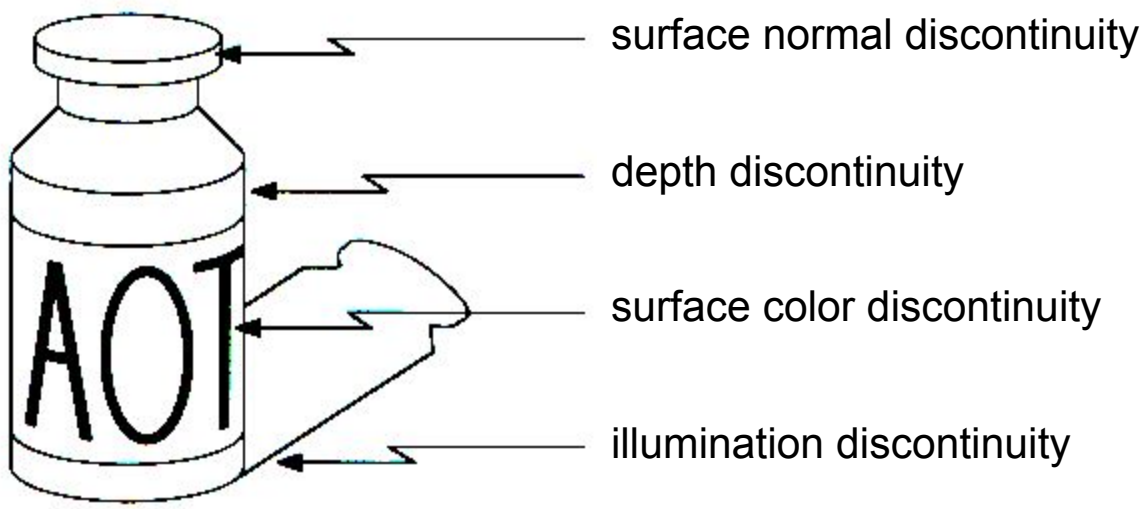
unit impulse

Gaussian

Laplacian of Gaussian

Edges

Origin of edges

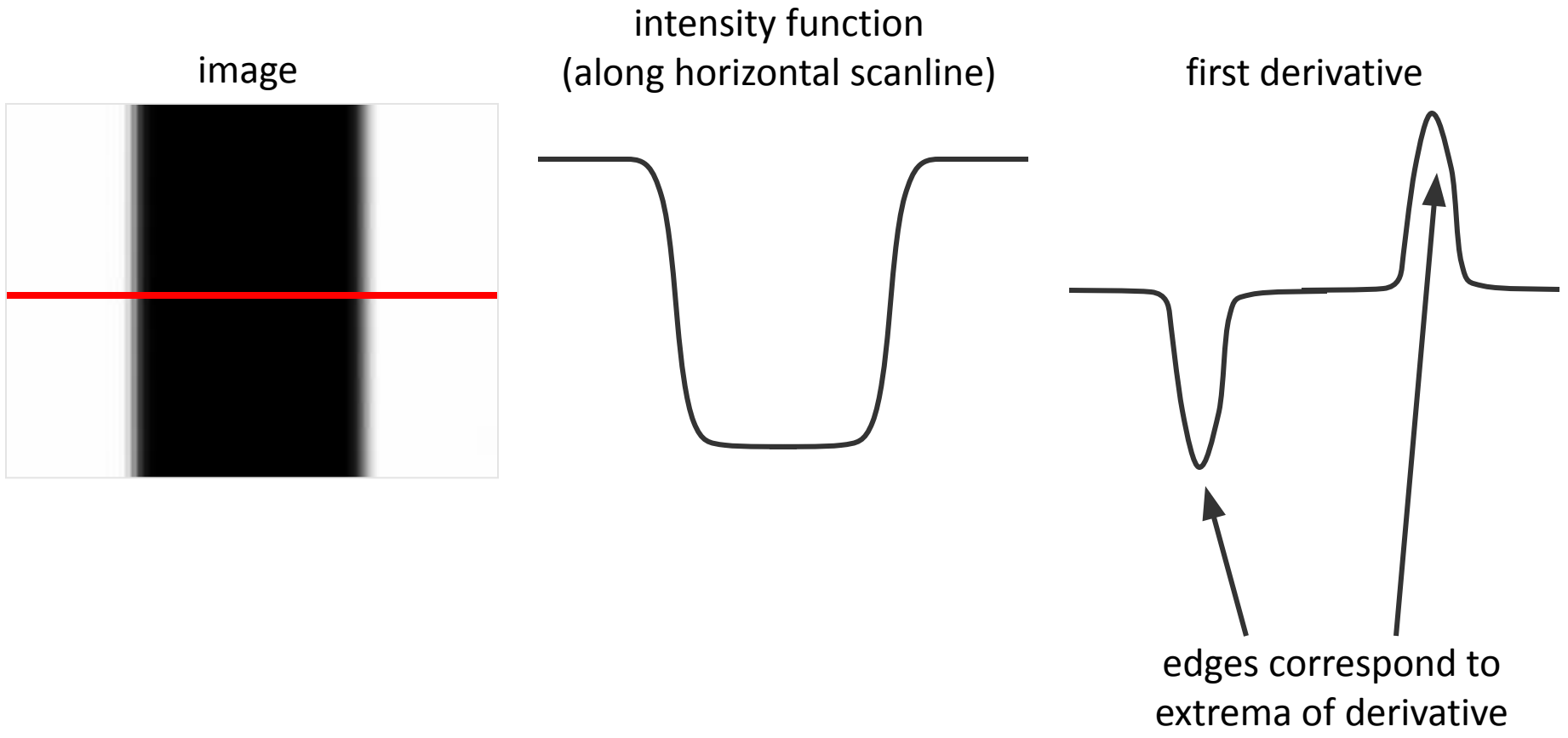


Edges are caused by a variety of factors

Credit: Ali Farhadi

Characterizing edges

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



Credit: Ali Farhadi

Partial derivatives with convolution

- For 2D function f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

- For discrete data, we can approximate using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$

- To implement above as convolution, what would be the associated filter?

-1	1

x

-1
1

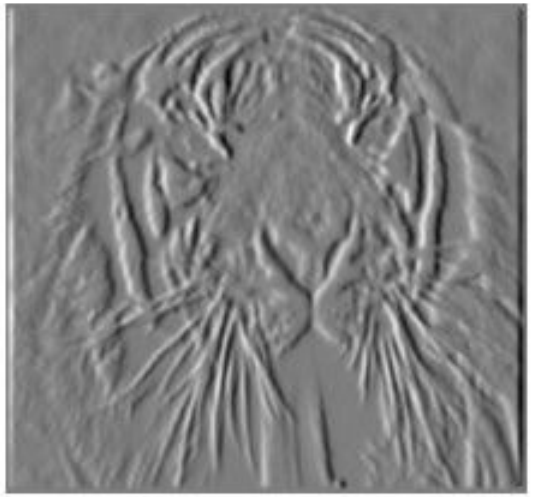
y

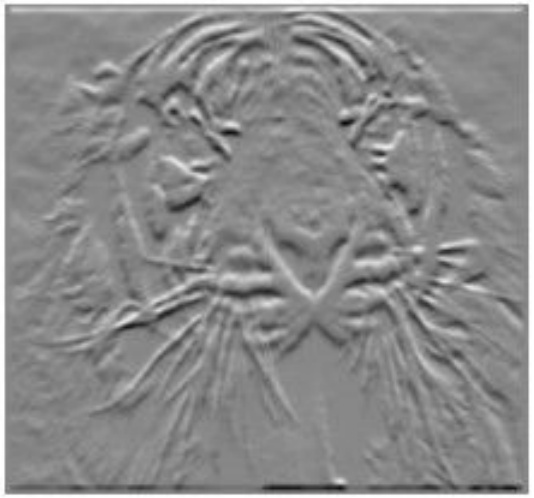
Source: K. Grauman

Partial derivatives of an image

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

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





-1	1

-1
1

Which shows changes with respect to x?

Finite difference filters

- **Other approximations of derivative filters exist:**

Prewitt

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

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

Sobel

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

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

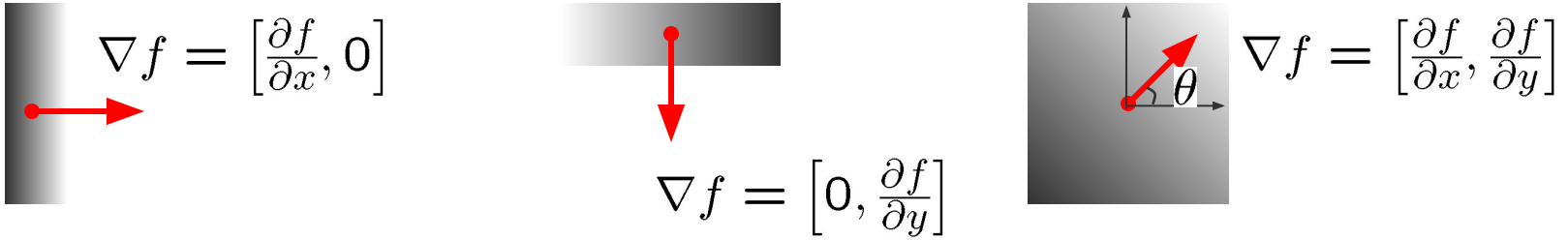
Roberts

0	1
-1	0

1	0
0	-1

Image gradient

- The gradient of an image: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

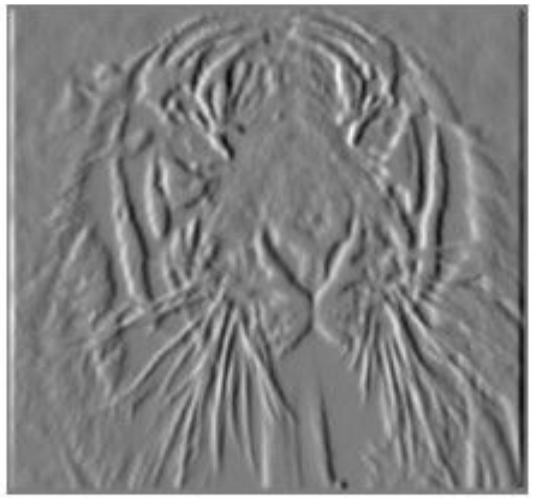


- Gradient points in the direction of most rapid increase in intensity
  - How does this direction relate to the direction of the edge?
- The gradient direction is given by $$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$
- The edge strength is given by the gradient magnitude

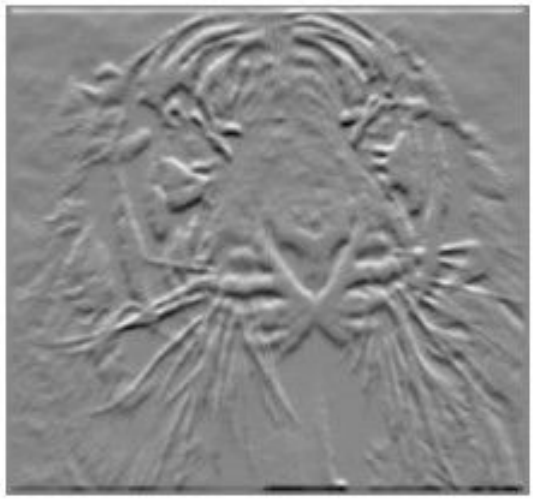
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

Image Gradient



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

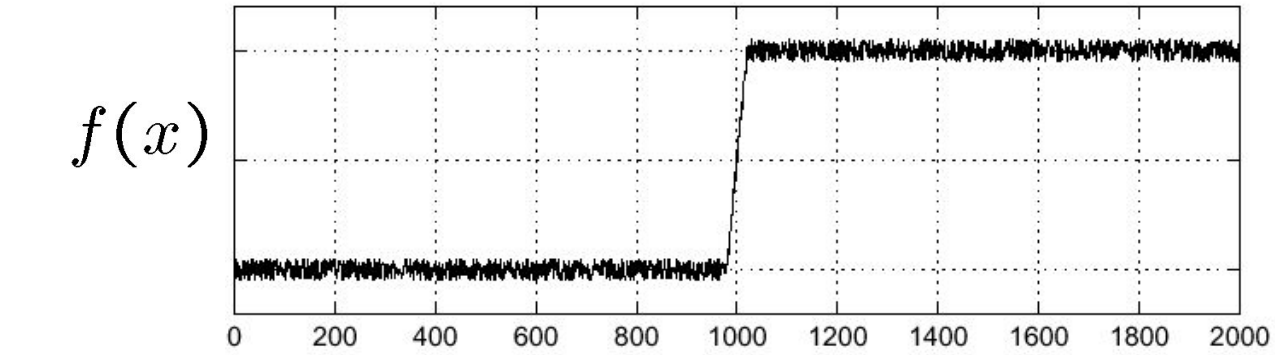


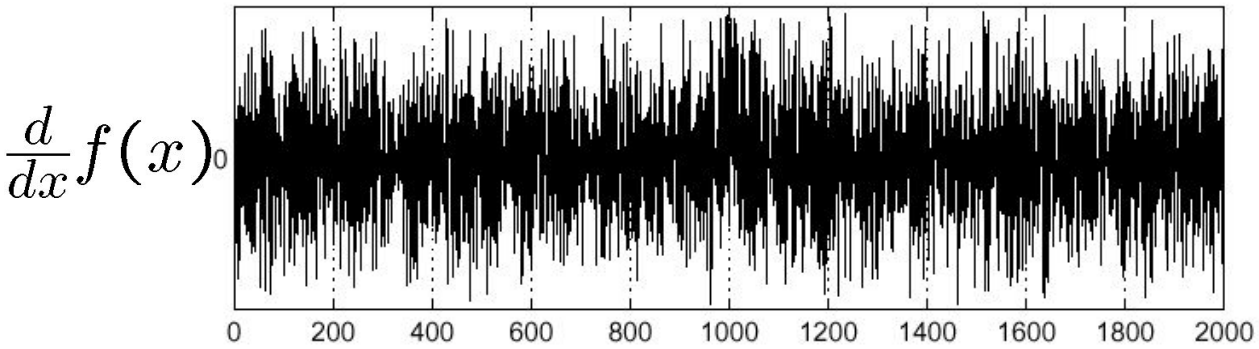
$$\frac{\partial f(x,y)}{\partial y}$$

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

Effects of noise

- Consider a single row the image (shown below)
- How can we find the edge?

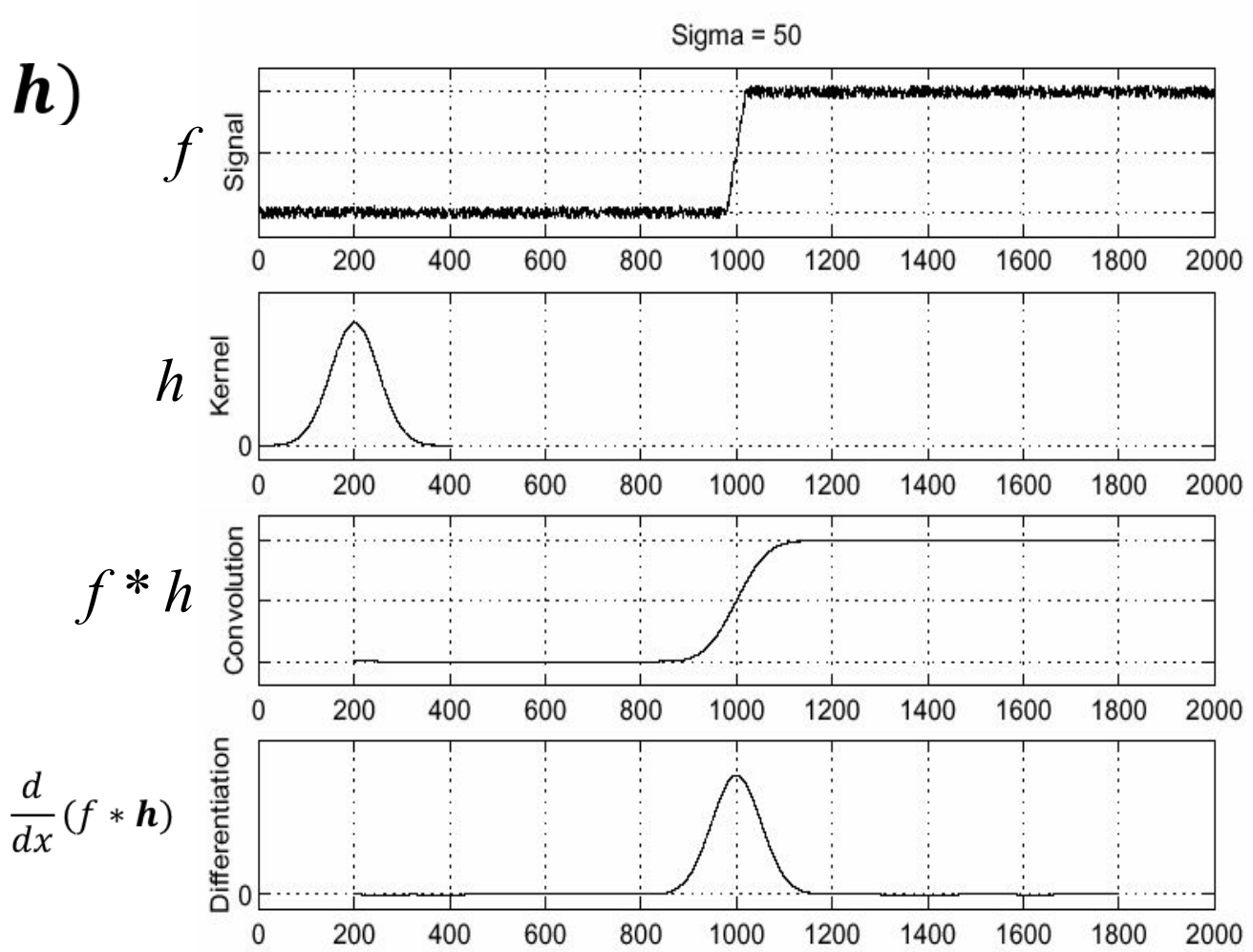




Source: S. Seitz

Solution: smooth first

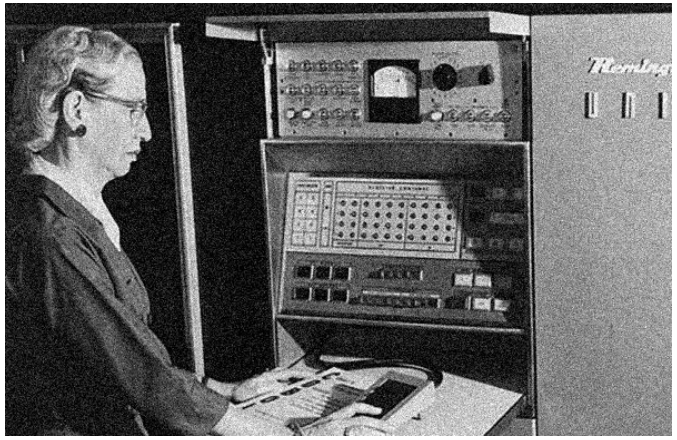
- Look for peak in $\frac{d}{dx}(f * \boldsymbol{h})$ to find the edge

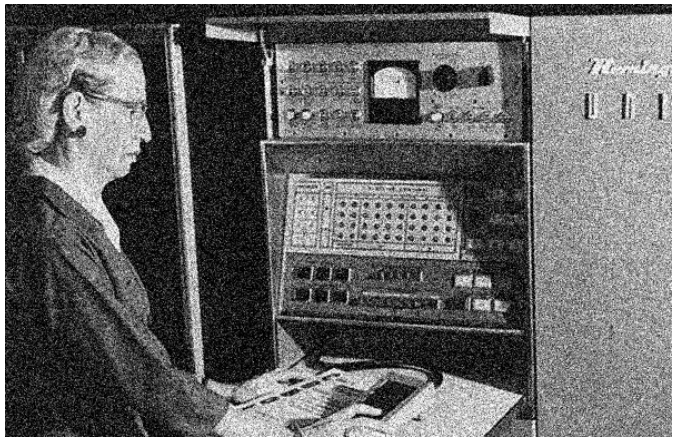


Source: S. Seitz

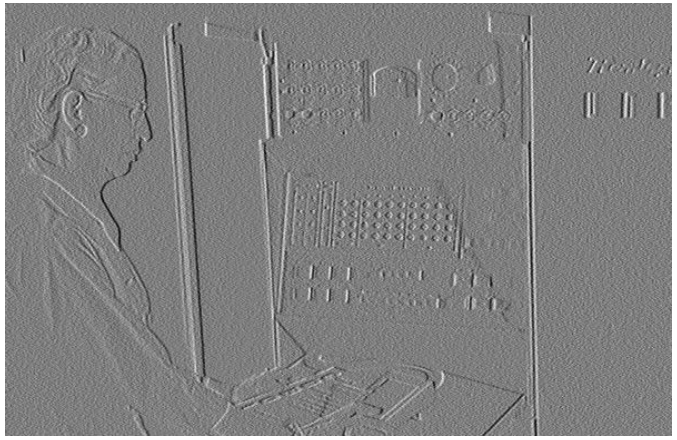
Noise in 2D

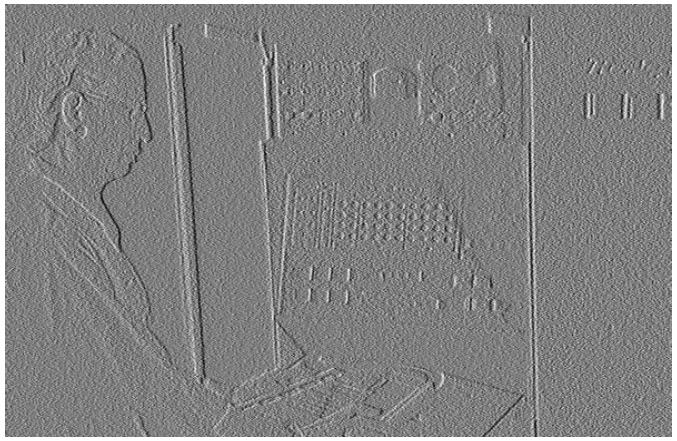
Noisy Input



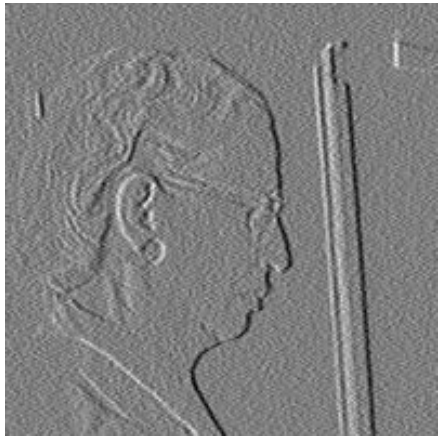


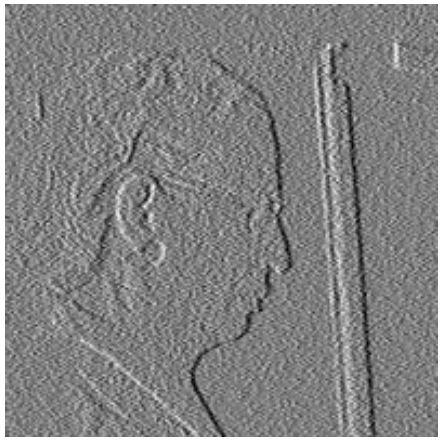
dx via [-1,1]





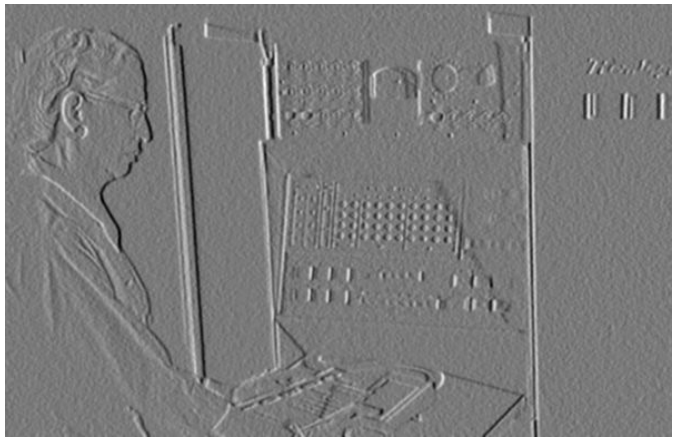
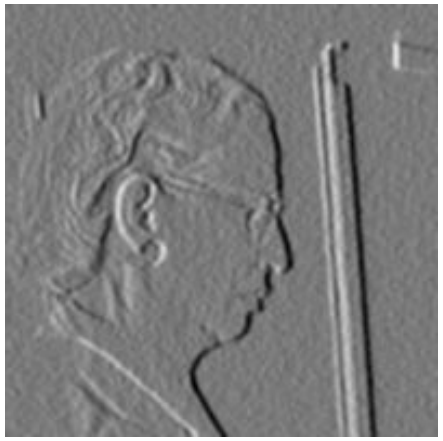
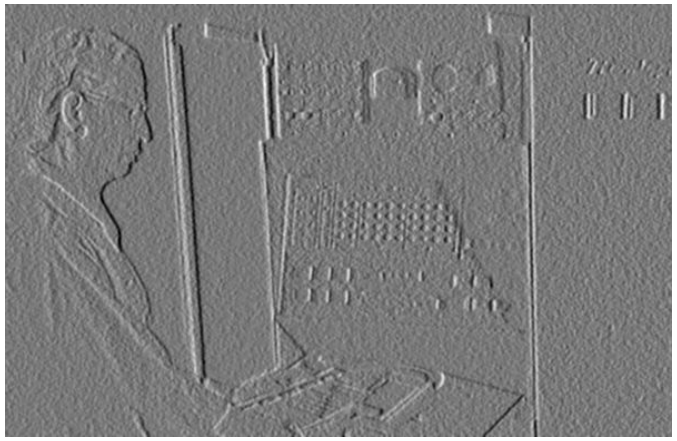
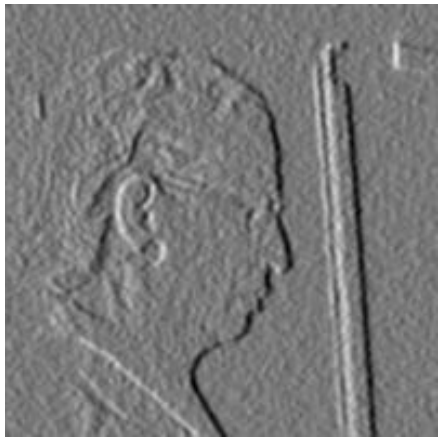
Zoom





Source: D. Fouhey

Noise + Smoothing

Smoothed Input	dx via [-1,1]	Zoom
		
		

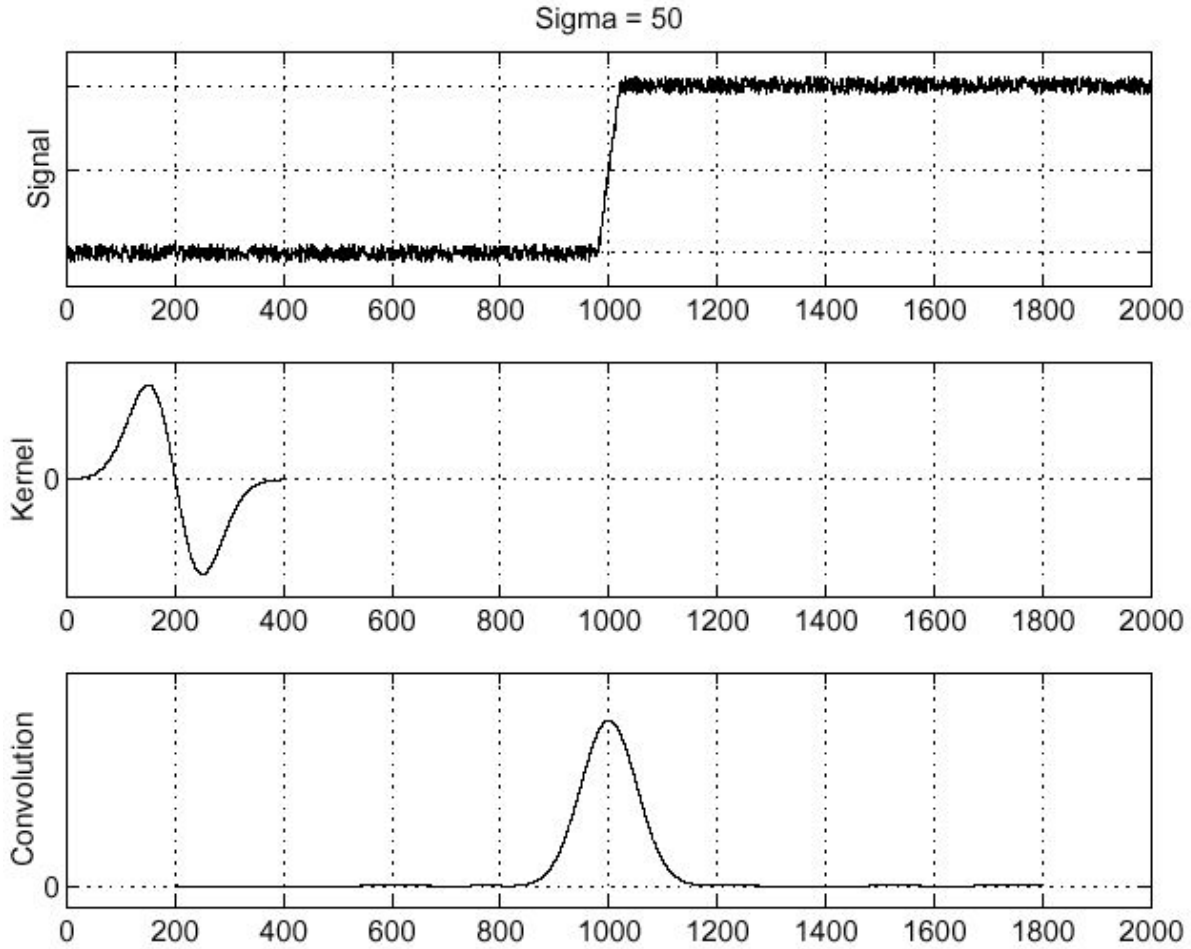
Source: D. Fouhey

Derivative theorem of convolution

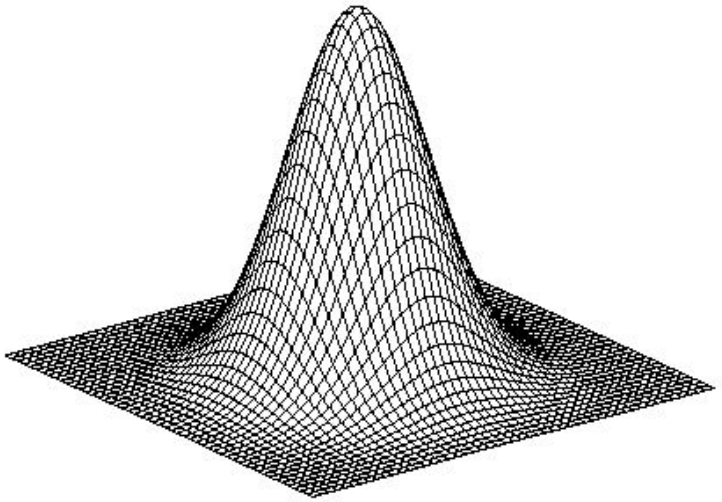
- **This saves us one operation:** $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

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

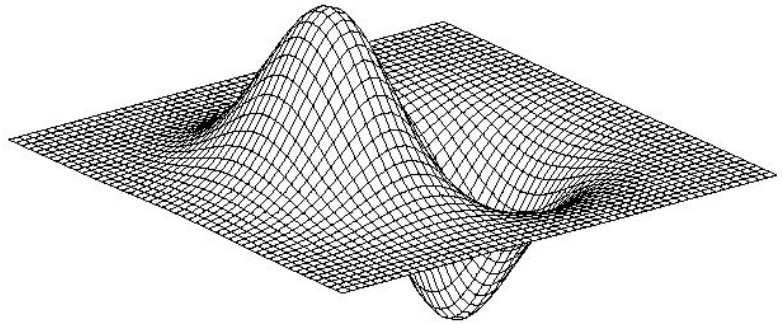
$(\frac{\partial}{\partial x}h) \star f$



Derivative of Gaussian filter

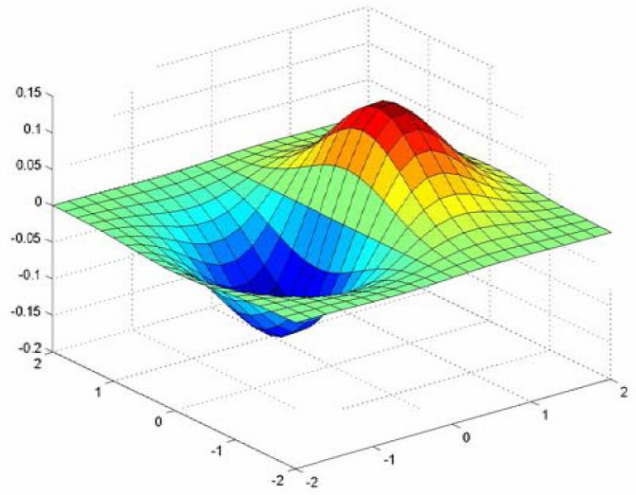


* [1 -1] =

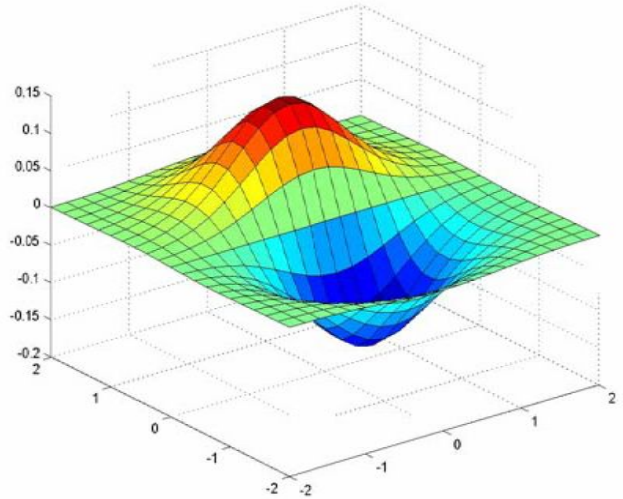


Derivative of Gaussian filter

- **Which one finds horizontal/vertical edges?**

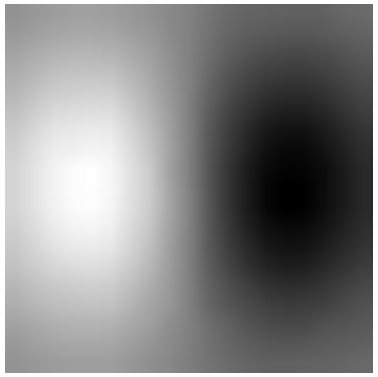


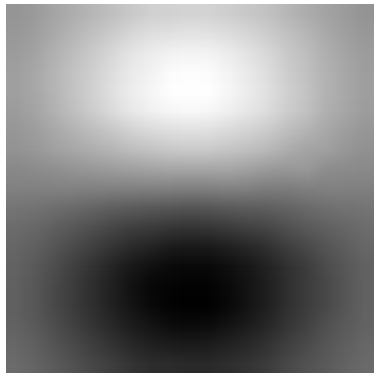
x-direction



y-direction

Vertical

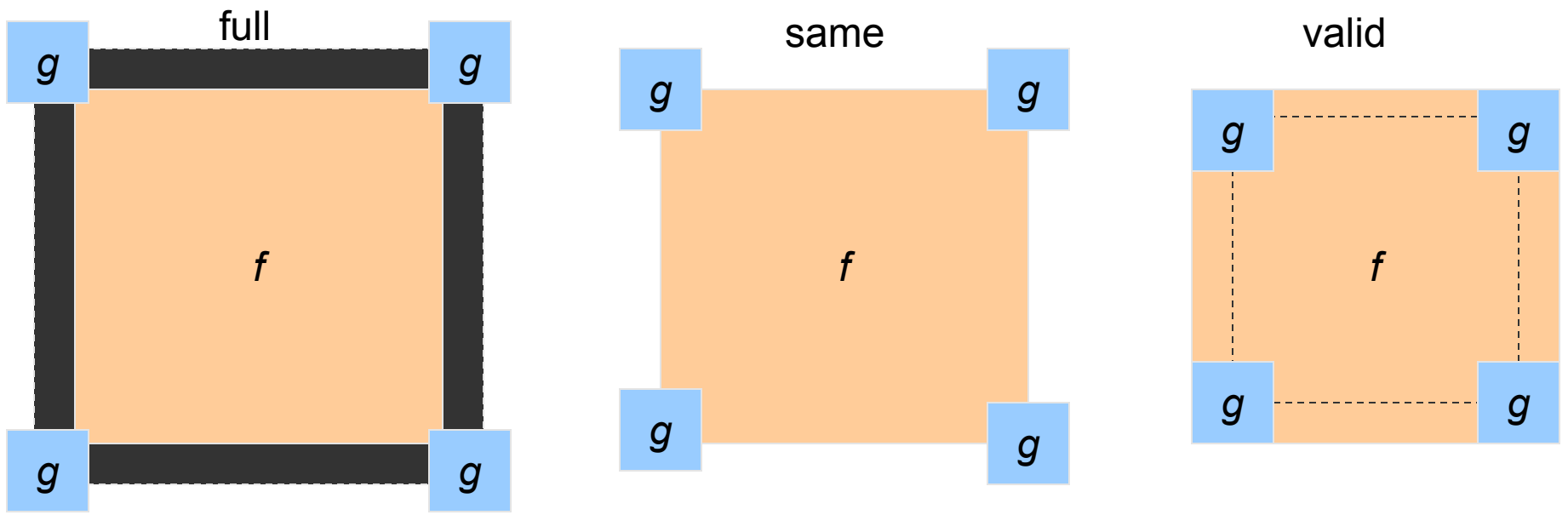




Horizontal

Practical matters

- What is the size of the output?
- Python: convolve2d(f, g, *shape*)
  - *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



Source: S. Lazebnik

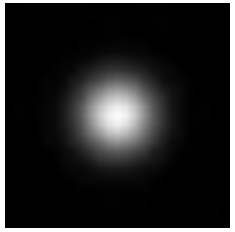
Practical matters

- **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**

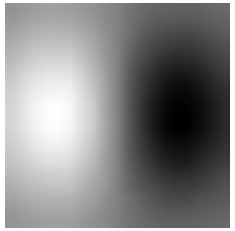
Source: S. Marschner

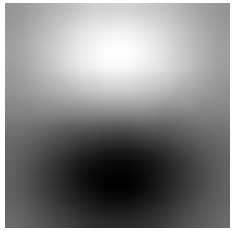
Summary

- **Smoothing filters**
  - Gaussian: remove “high-frequency” components; “low-pass” filter
  - Can the values of a smoothing filter be negative?
  - What should the values sum to?
    - One: constant regions are not affected by the filter



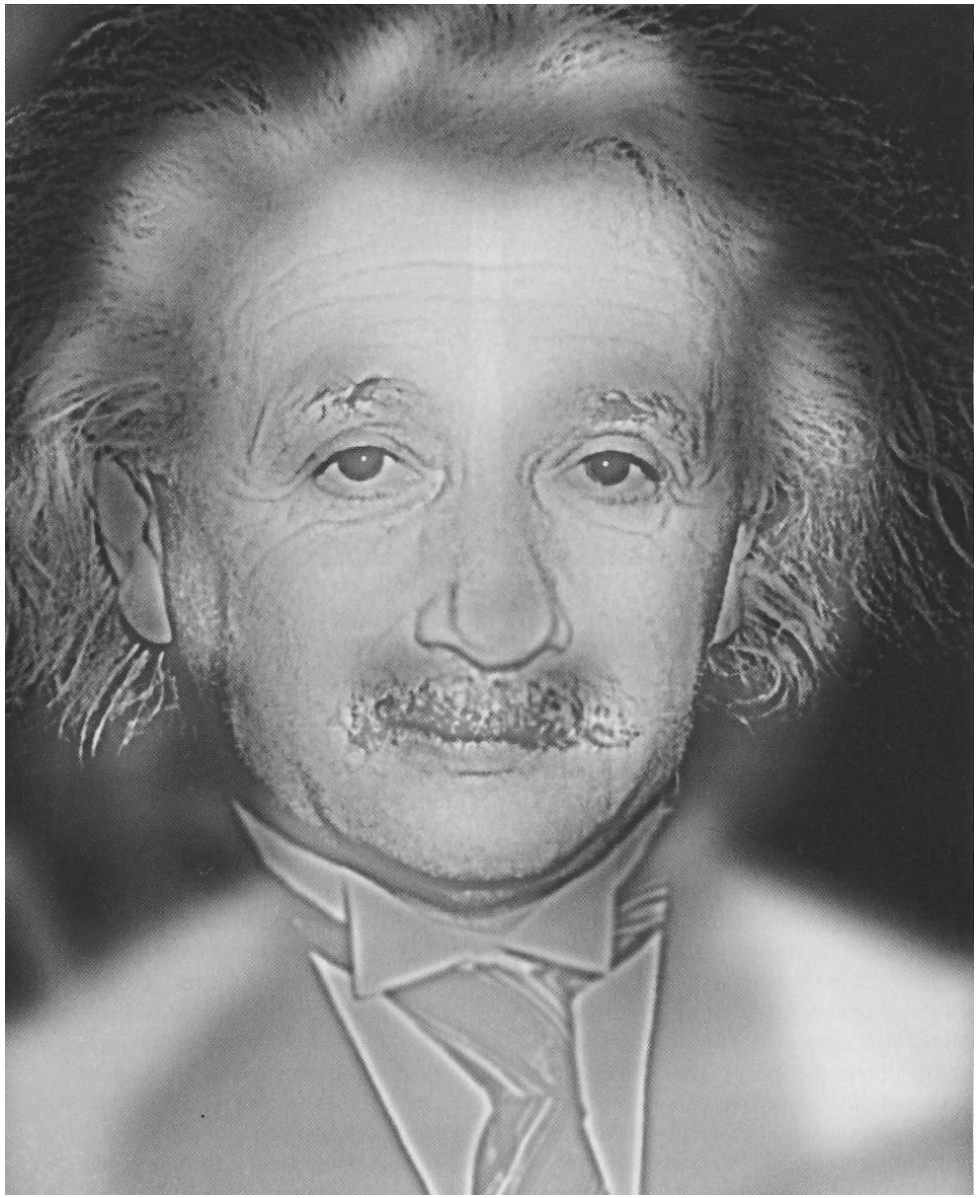
- **Derivative filters**
  - Derivatives of Gaussian
  - Can the values of a derivative filter be negative?
  - What should the values sum to?
    - Zero: no response in constant regions





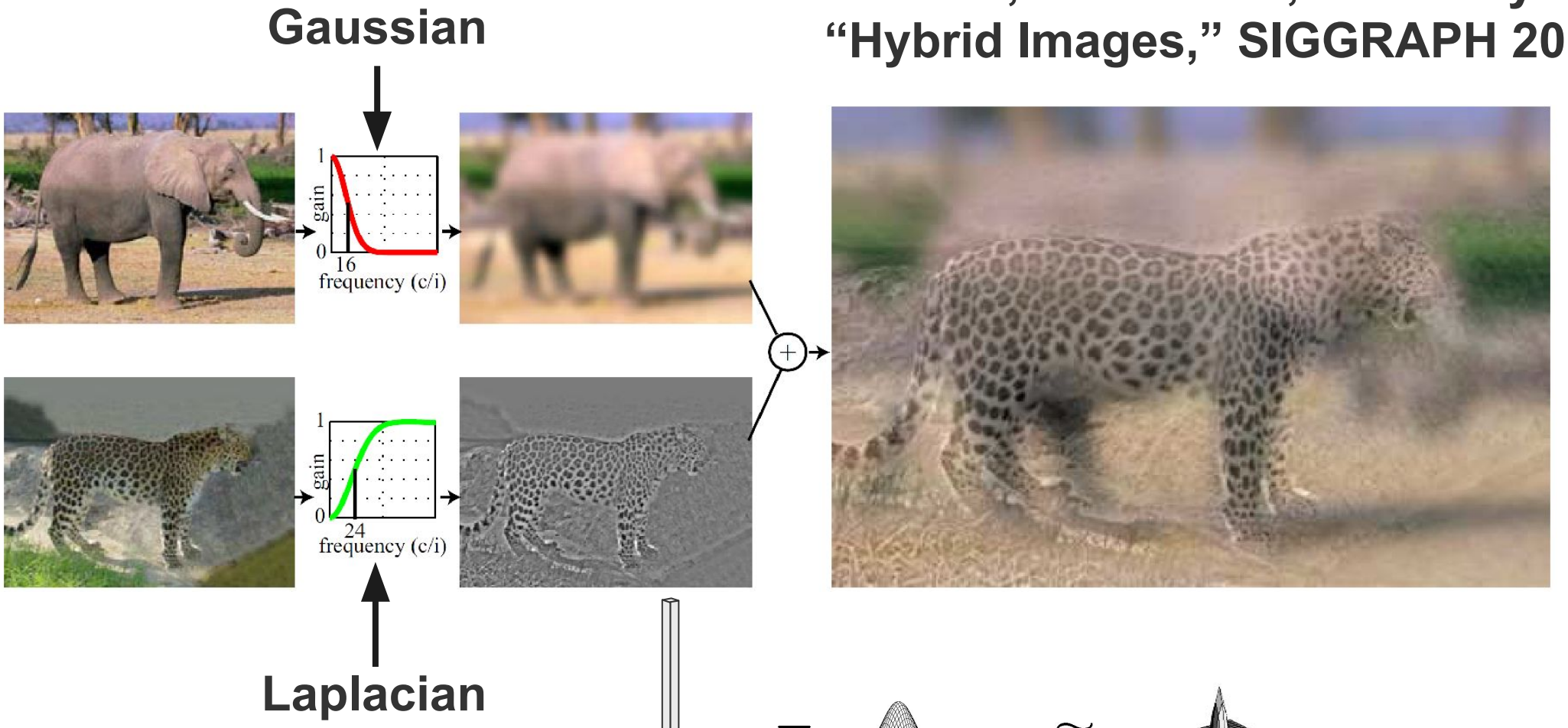
A cool application

Hybrid images

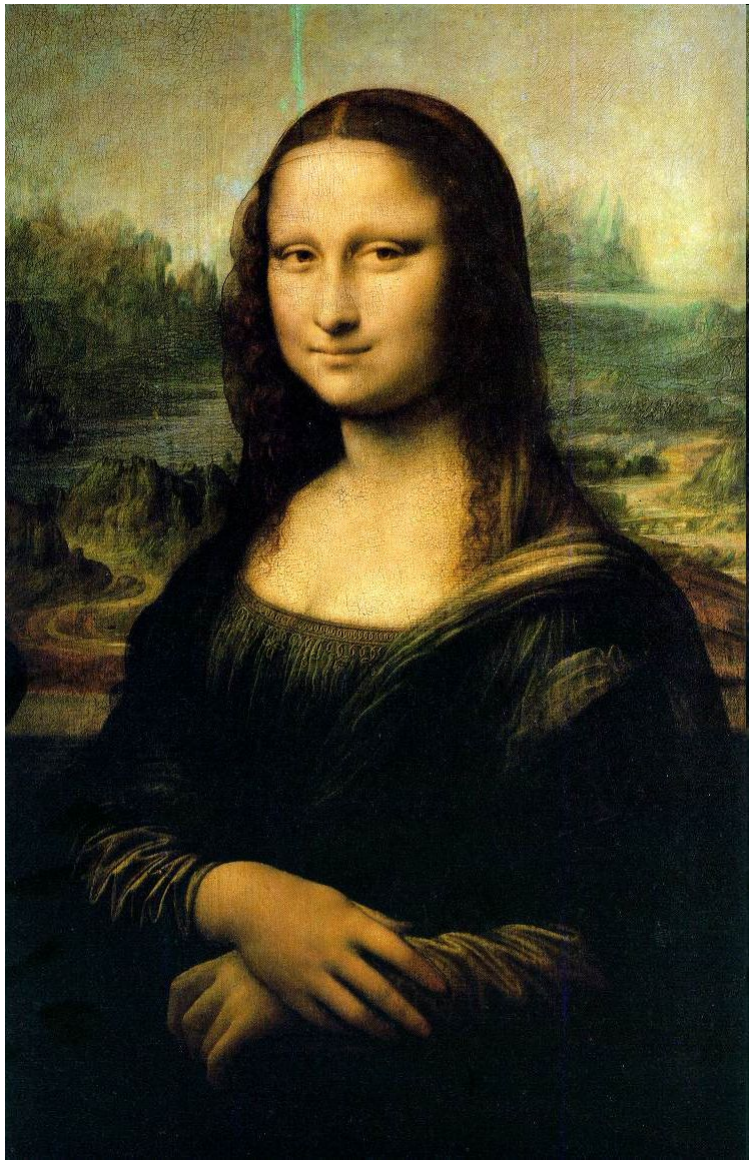


Hybrid Images

A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006



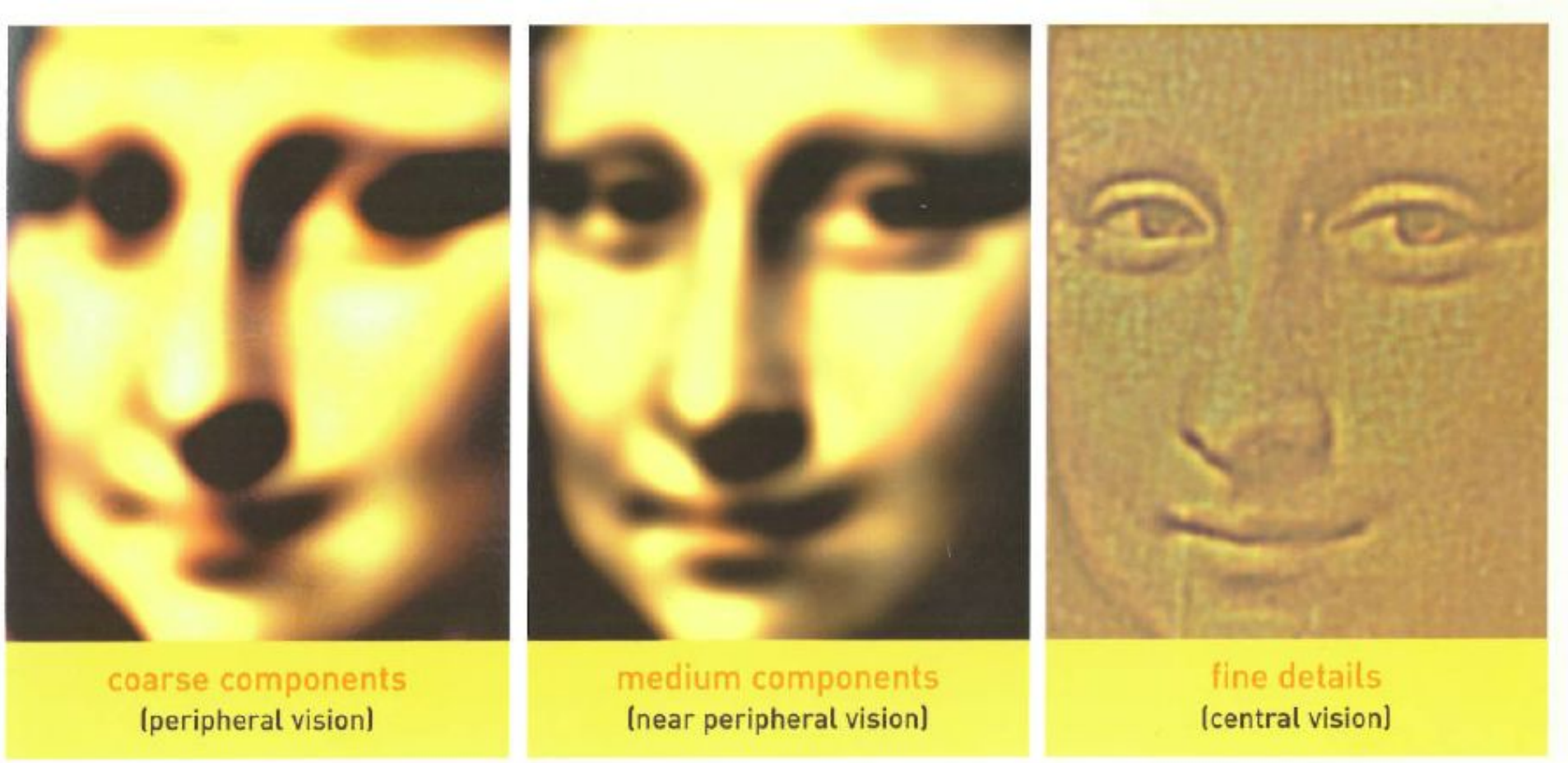
Da Vinci and Peripheral Vision



https://en.wikipedia.org/wiki/Speculations_about_Mona_Lisa#Smile

Da Vinci and Peripheral Vision

- **Leonardo playing with peripheral vision**



Livingstone, Vision and Art: The Biology of Seeing