

Convolution

Fredo Durand
MIT EECS 6.8370/6.871

Do the flip earlier.

Administrivia

- PSet 0 has weight 0 (still do it!!)
- PSet 1 due next Wednesday
- Pset system miki3
 - Contact us if you don't have an account
- How is it going?



Blur, sharpen



Image processing motivation

- Sharpen images
- Downsample images
- Fake depth of field
- Smooth out noise, skin blemishes
- ...
- We must understand convolution!

Sharpening



Blur in optics

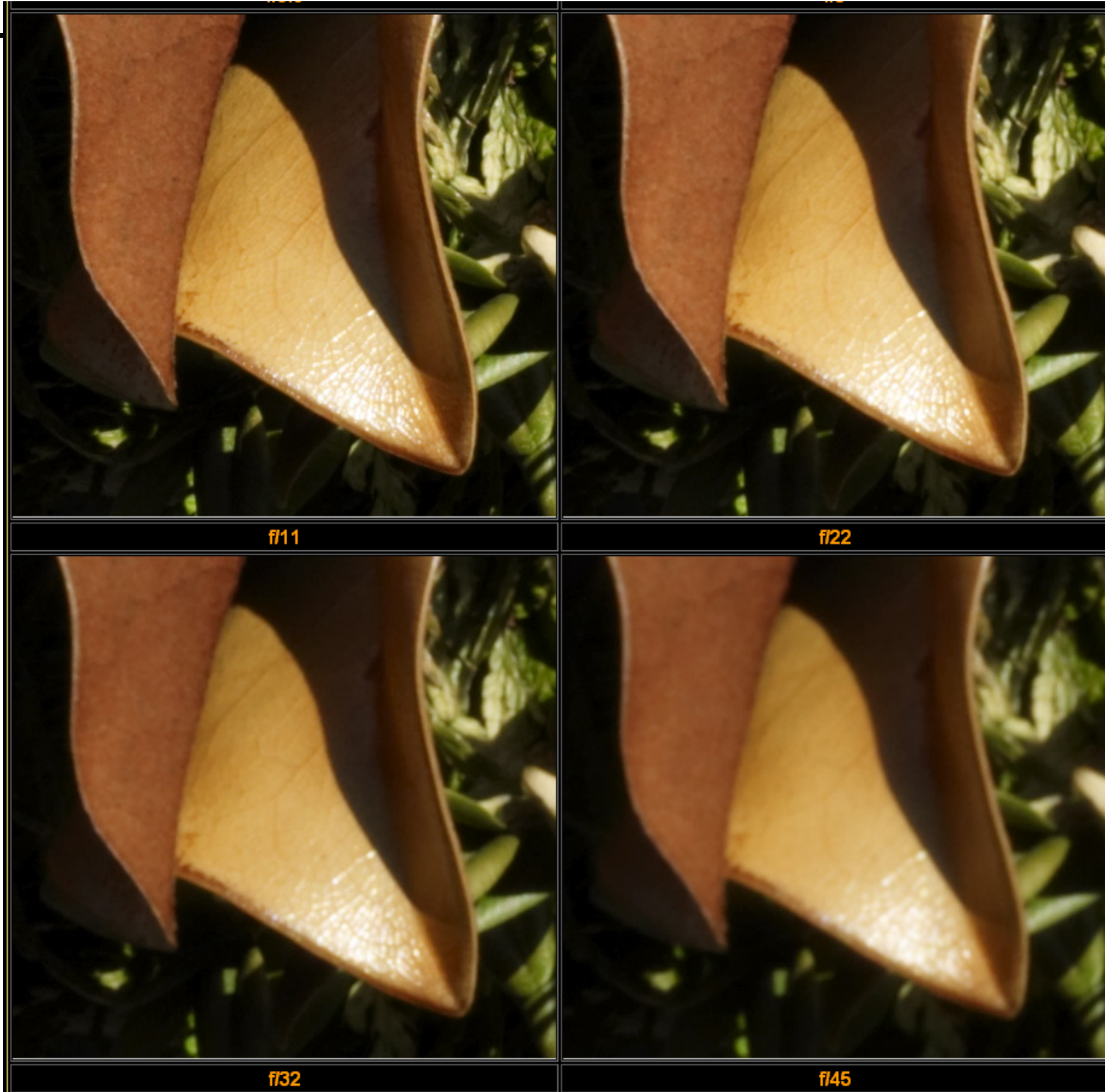
- Diffraction
- Lens aberrations
- Object movement
- Camera shake
- Can we remove blur computationally?
 - Inverse the blur equation
 - deconvolution
 - later!

Lens diffraction

- <http://luminous-landscape.com/tutorials/understanding-series/u-diffraction.shtml>

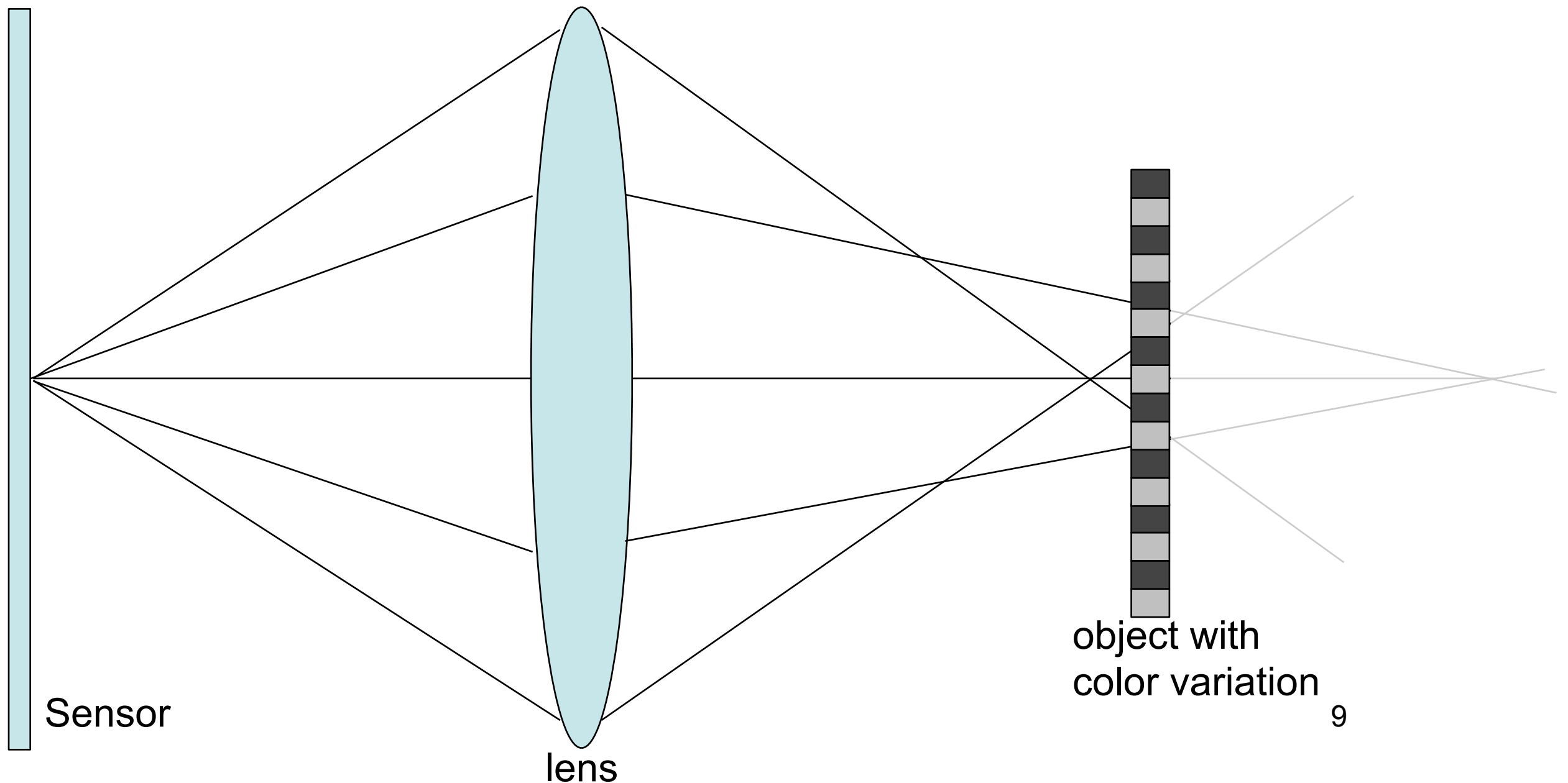
(heavily cropped)

- See also <http://www.cambridgeincolor.com/tutorials/diffraction-photography.htm>



Blur example: spherical aberration

- Pixel value: weighted average of local color



Remove optical artifacts

- Calibrate lenses and remove blur
- e.g. DXO



C in CNNs

Questions?

Convolution

101

Blur as convolution

- Replace each pixel by a linear combination of its neighbors.
 - only depends on relative position of neighbors
- The prescription for the linear combination is called the “convolution kernel”.

10	5	3
4	5	1
1	1	7

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	7	

Modified image data
(shown at one pixel)

Linear **shift-invariant** filtering

- Replace each pixel by a linear combination of its neighbors.
 - only depends on relative position of neighbors
- The prescription for the linear combination is called the “convolution kernel”.
 - same kernel for all pixels**

10	5	3
4	5	1
1	1	7

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	7	

Modified image data
(shown at one pixel)

Example of linear NON-shift invariant transformation?

- e.g. neutral-density graduated filter (darken high y , preserve small y)
 $J(x,y)=I(x,y)*(1-y/y_{\max})$
- Formally, what does linear mean?
 - For two scalars a & b and two inputs x & y :
 $F(ax+by)=aF(x)+bF(y)$
- What does shift invariant mean?
 - For a translation T :
 $F(T(x))=T(F(x))$
 - If I blur a translated image,
I get a translated blurred image

Questions?



Why linearity matters

- If blur is a linear operation
- We can try to deblur by inverting a linear system
- We have tools to tell us how doable it may be
 - Spoiler alert: not always easy
- Side note: makes it important to know if your image is gamma-encoded

Convolution algorithm

- Set output image to zero
- For x, y in output image
 - for x', y' in kernel
 - $\text{out}[x, y] += \text{input}[x+x', y+y'] * \text{kernel}[x', y']$
- (this assumes the kernel coordinates are centered)

*Yes, there is a
"flip" missing*

10	5	3
4	5	1
1	1	7

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	7	

Modified image data
(shown at one pixel)

Questions?

- Set output image to zero
- For x, y in output image
 - for x', y' in kernel
 - $\text{out}[x, y] += \text{input}[x+x', y+y'] * \text{kernel}[x', y']$
 - (this assumes the kernel coordinates are centered)

10	5	3
4	5	1
1	1	7

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

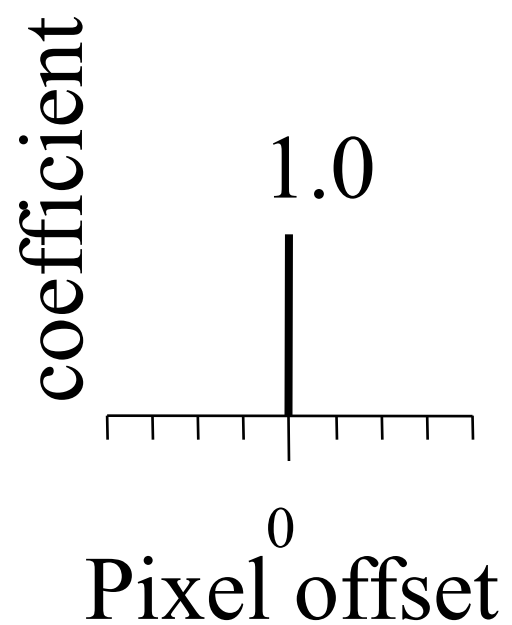
	7	

Modified image data
(shown at one pixel)

Convolution (warm-up slide)



original

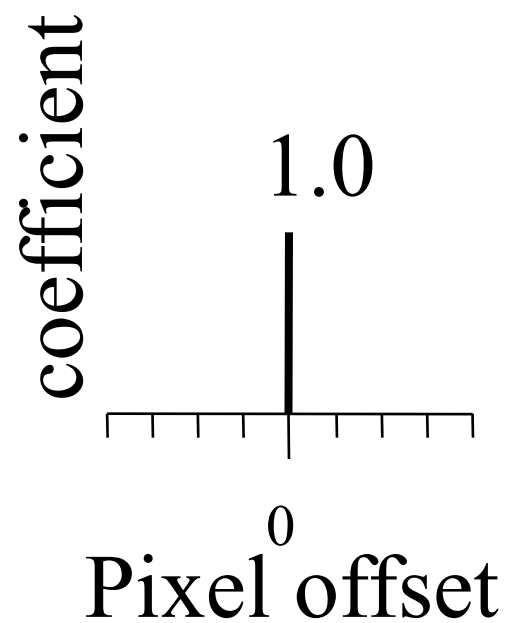


?

Convolution (warm-up slide)



original

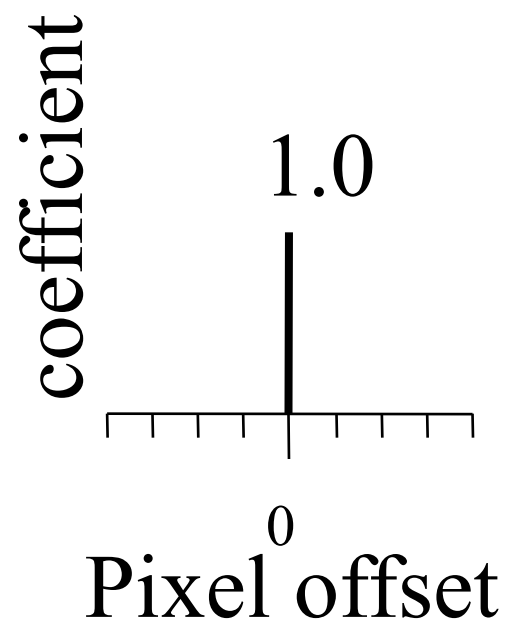


Filtered
(no change)

Convolution (warm-up slide)



original



Filtered
(no change)

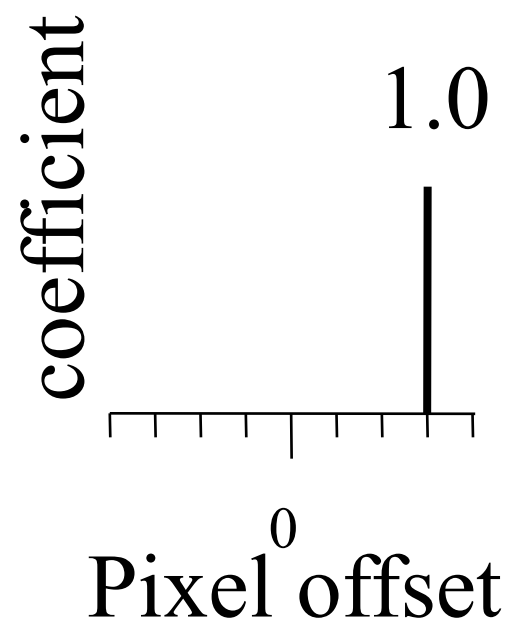
δ

$$f = f \otimes \delta$$

Convolution

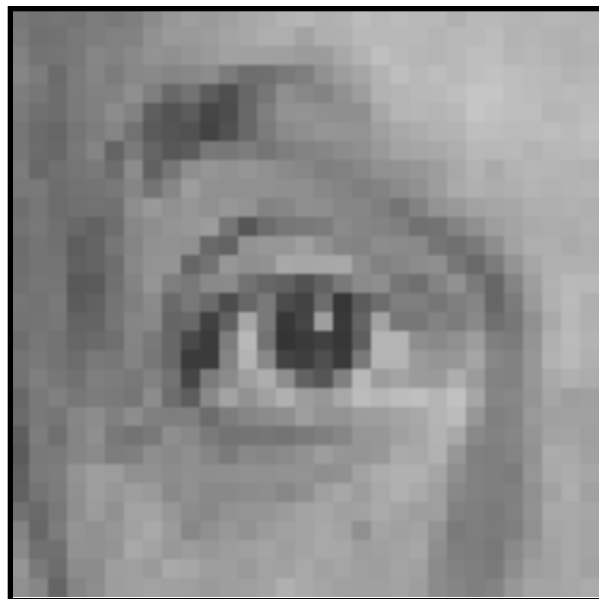


original

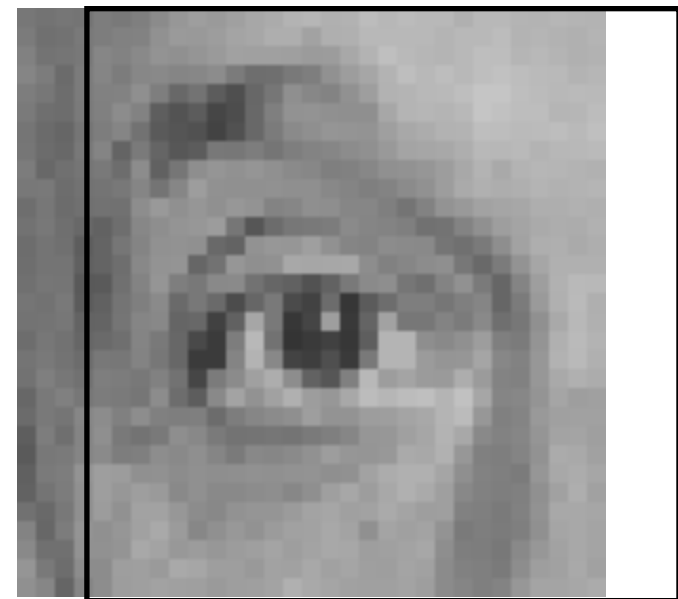
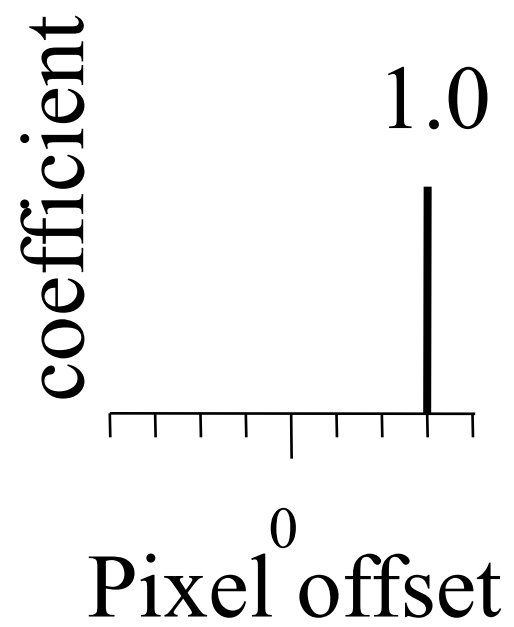


?

shift



original

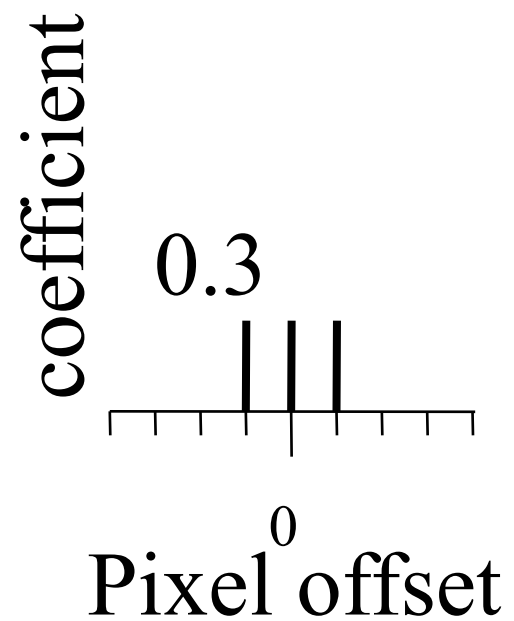


shifted

Convolution



original

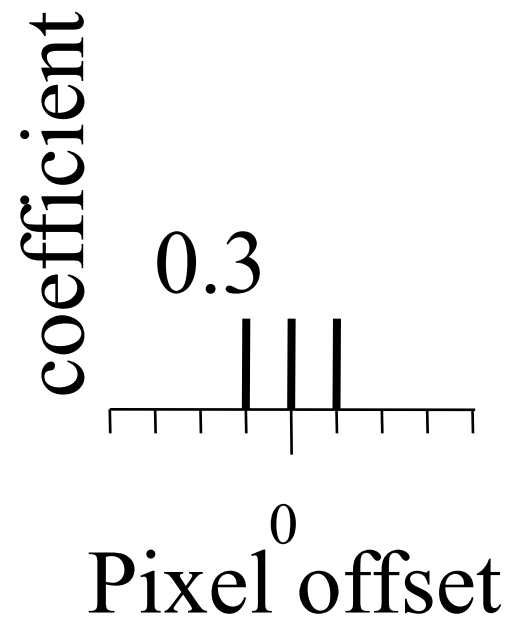


?

Blurring

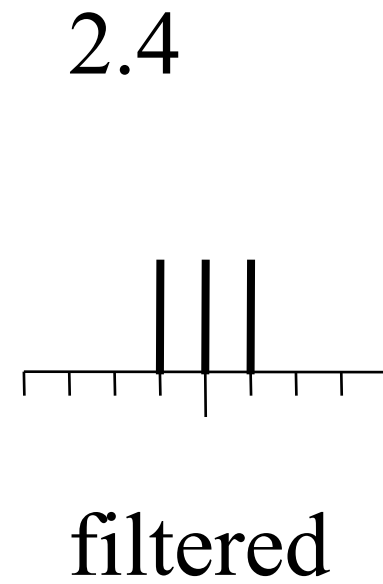
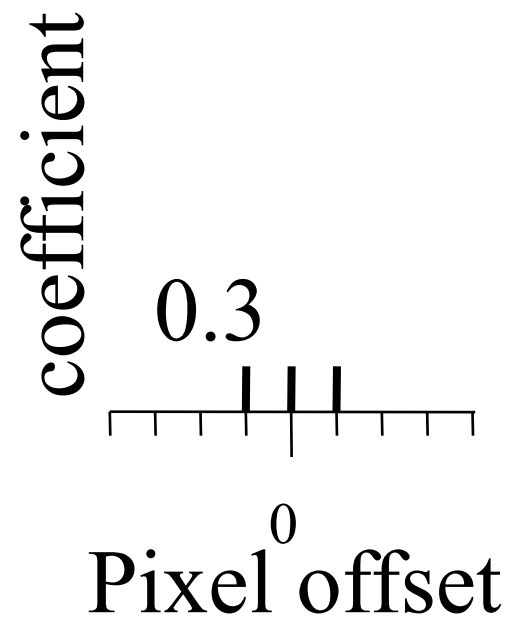
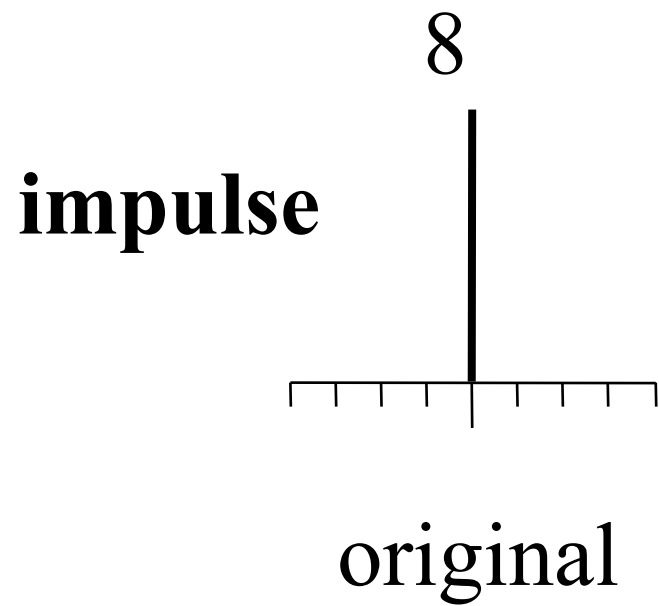


original



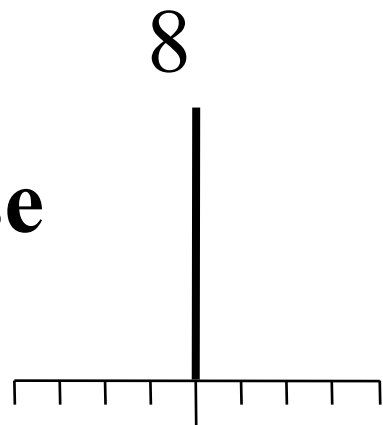
Blurred (filter applied in both dimensions).

Blur examples



Blur examples

impulse



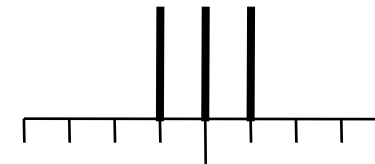
original

coefficient

0.3

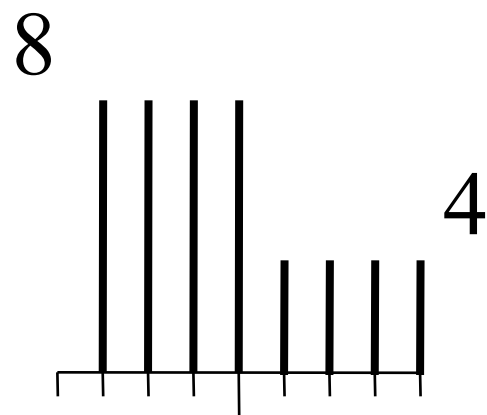
Pixel⁰ offset

2.4



filtered

edge



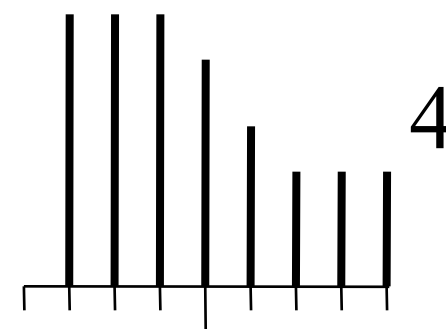
original

coefficient

0.3

Pixel⁰ offset

8



filtered

Questions?

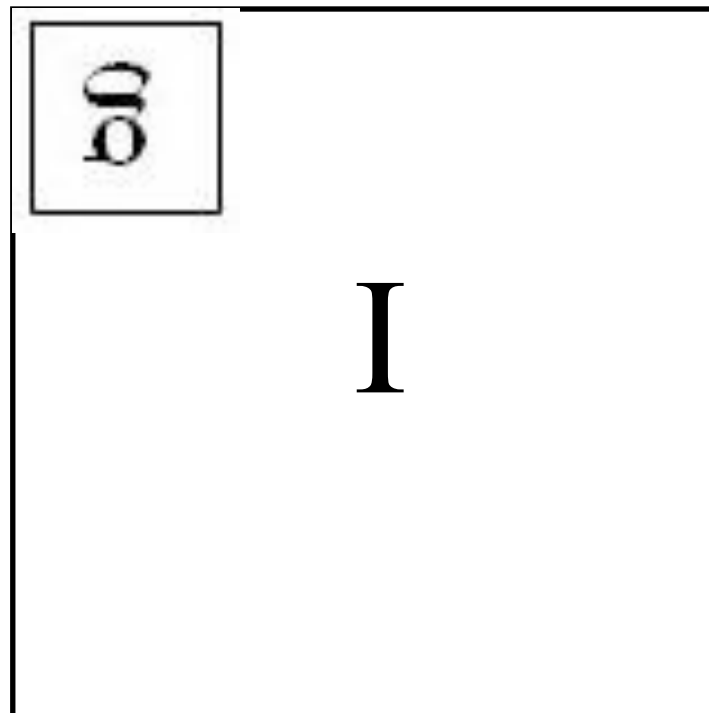


Formally



More formally: Convolution

$$(I \otimes g)(x) = \int_{x'} I(x') g(x - x') dx'$$

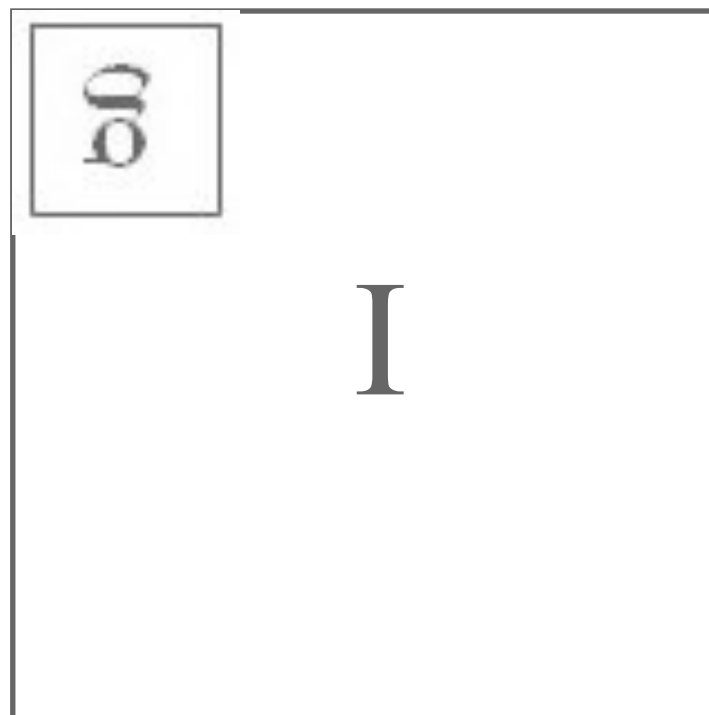


Demo

- <http://math.mit.edu/daimp/ConvFlipDrag.html>

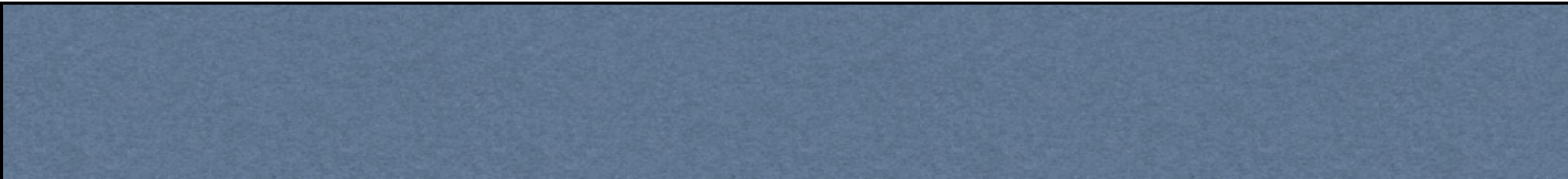
Questions?

$$(I \otimes g)(x) = \int_{x'} I(x')' g(x - x') dx'$$





**What's up
with flipping?**



Simpler flip

- $f(x-1)$
 - does it translate to the left or to the right?

Simpler flip

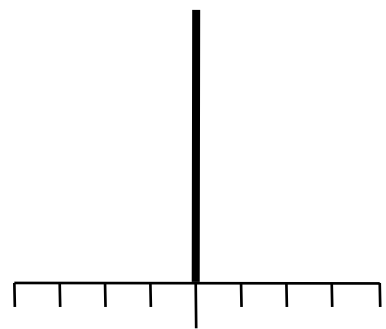
- $f(x-1)$
 - translates to the right!
 - Because it's where we're reading values from, not where values are going to.
 - Same phenomenon happens with convolutions.

$$(I \otimes g)(x) = \int_{x'} I(x') g(x - x') dx'$$

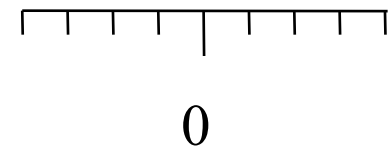
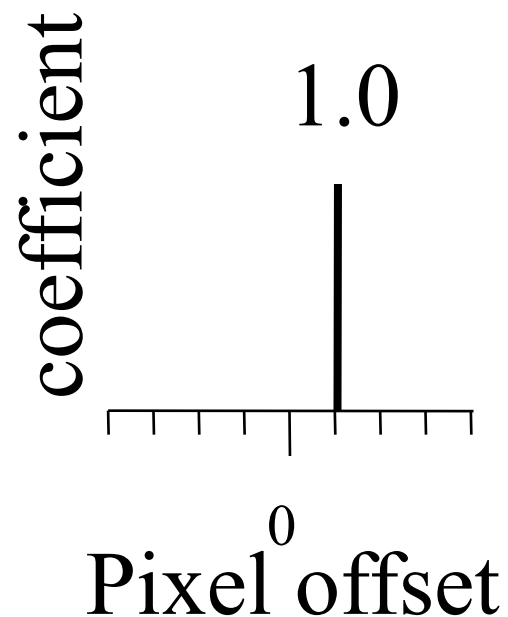
Point Spread Function

- Run convolution on Impulse

$$(I \otimes g)(x) = \int_{x'} I(x')g(x - x')dx'$$



original

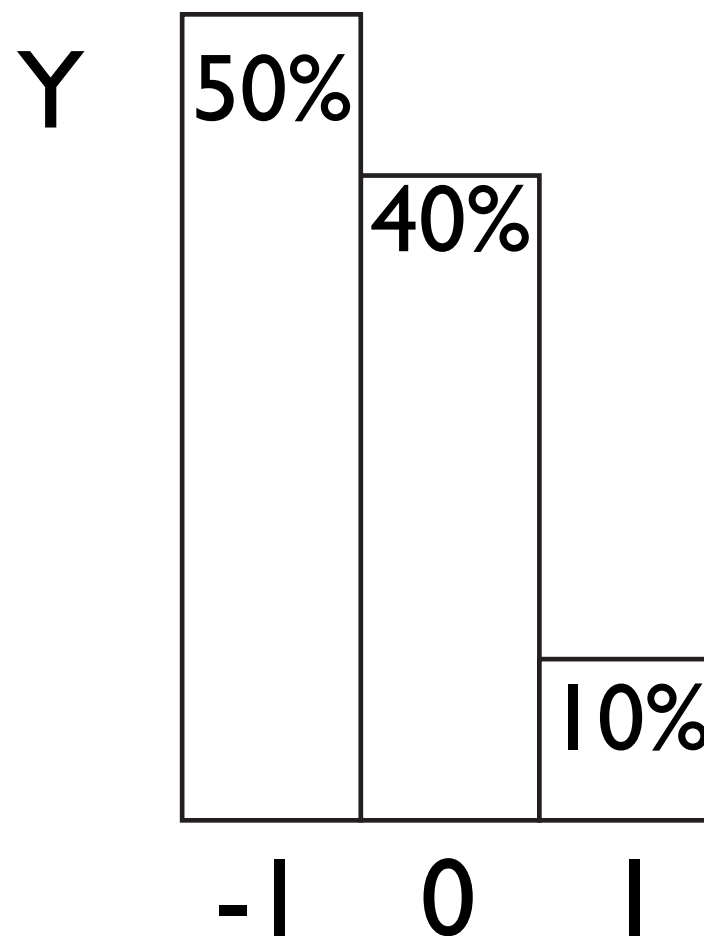


Convolution & probability

- Convolution was first used by Laplace to study the probability of the sum of two random variables



Random variables



Random variables

- How can $X+Y=0$?

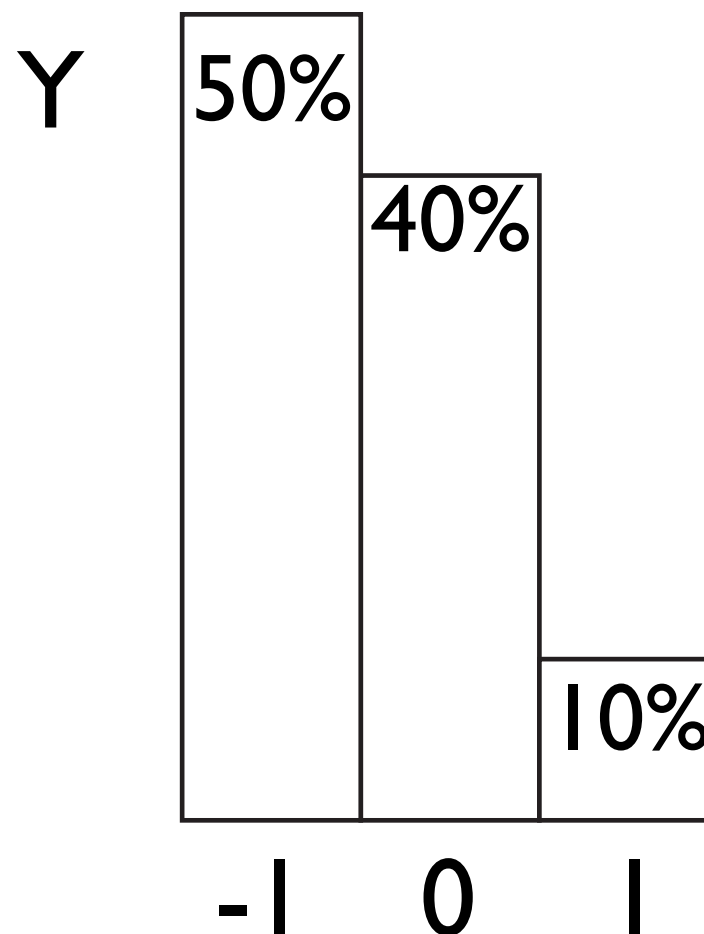
- $X=-1, Y=1$
- $X=0, Y=0$
- $X=1, Y=-1$

Probability?

$$P(X=-1)*P(Y=1)$$

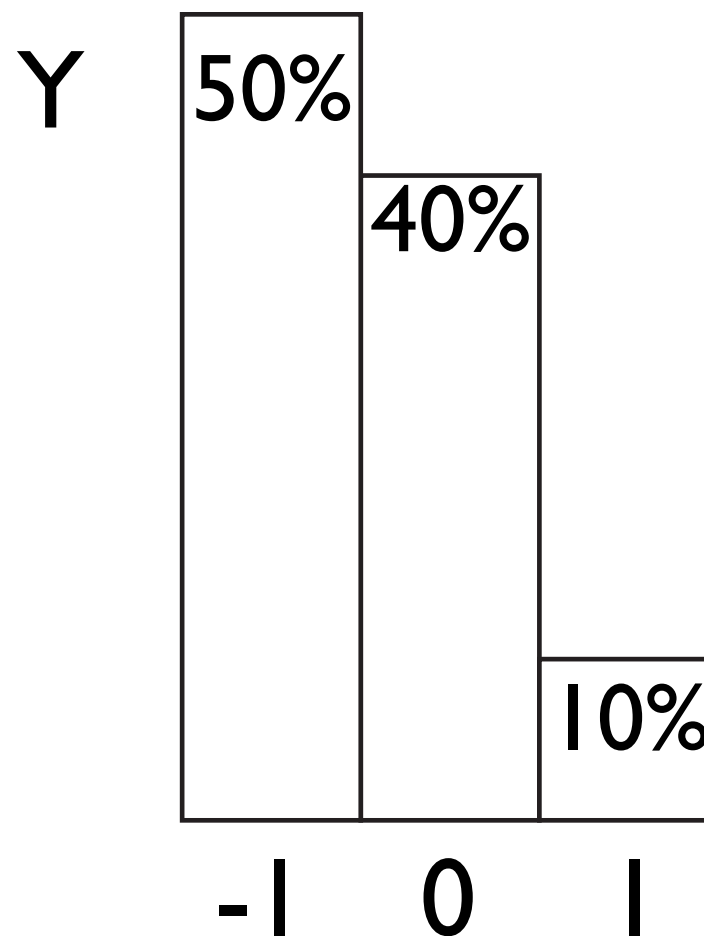
$$P(X=0)*P(Y=0)$$

$$P(X=1)*P(Y=-1)$$



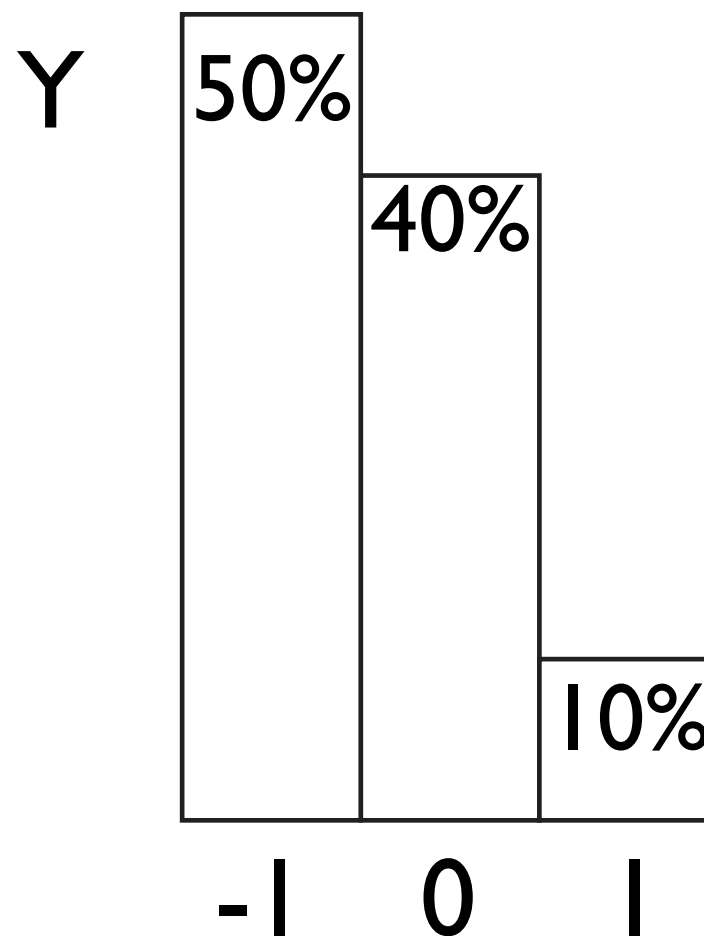
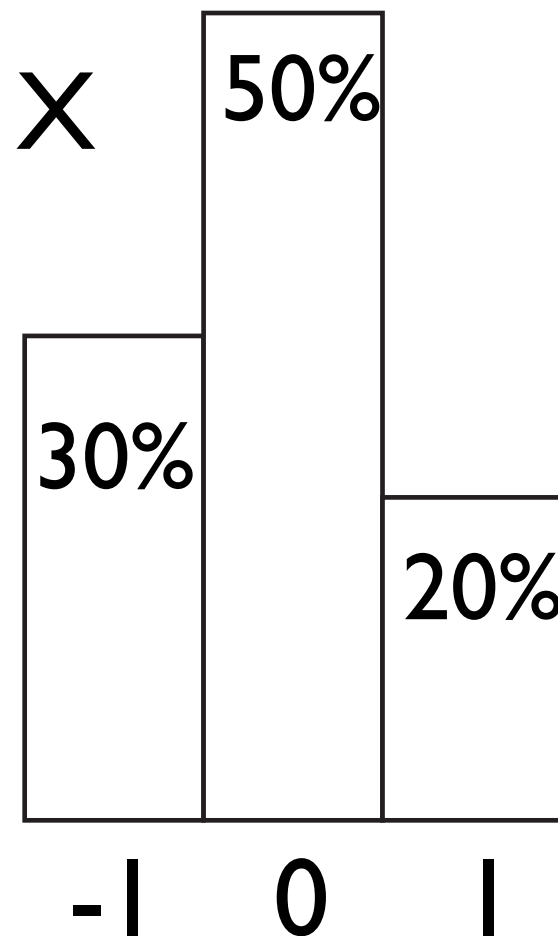
Sum of random variables

$$P(X + Y = k) = \sum_{k'} P(X = k')P(Y = k - k')$$



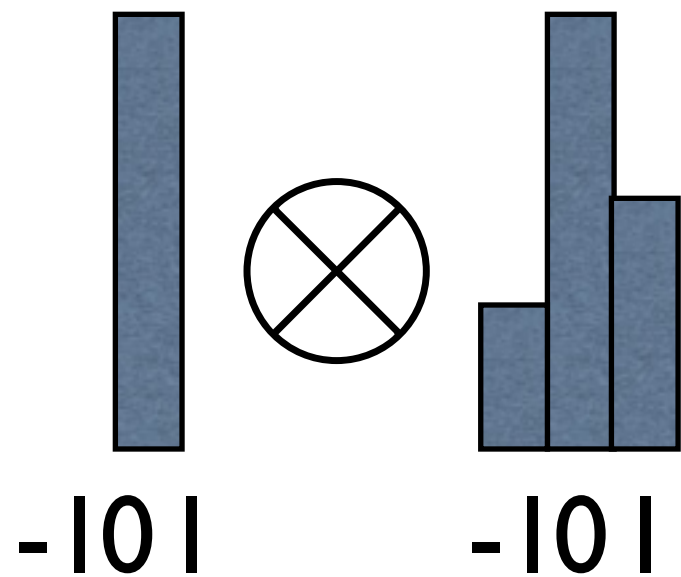
Questions?

$$P(X + Y = k) = \sum_{k'} P(X = k') P(Y = k - k')$$



Point Spread Function & flip

$$(I \otimes g)(x) = \int_{x'} I(x') g(x - x') dx'$$



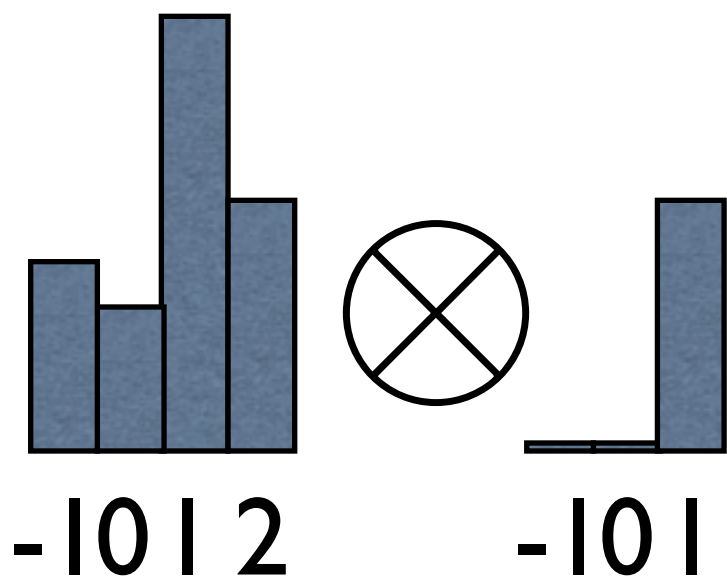
$$(I \otimes g)(-1) = I(-1)g(-1 - -1) + I(0)g(-1 - 0) + I(1)g(-1 - 1)$$

$$(I \otimes g)(0) = I(-1)g(0 - -1) + I(0)g(0 - 0) + I(1)g(0 - 1)$$

$$(I \otimes g)(1) = I(-1)g(1 - -1) + I(0)g(1 - 0) + I(1)g(1 - 1)$$

Point Spread Function & flip

$$(I \otimes g)(x) = \int_{x'} I(x') g(x - x') dx'$$



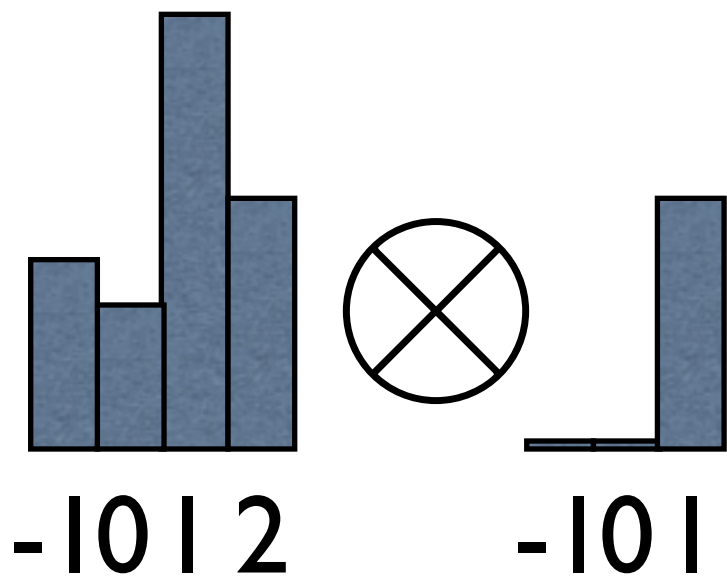
$$(I \otimes g)(-1) = I(-1)g(-1 - -1) + I(0)g(-1 - 0) + I(1)g(-1 - 1) \dots$$

$$(I \otimes g)(0) = I(-1)g(0 - -1) + I(0)g(0 - 0) + I(1)g(0 - 1) \dots$$

$$(I \otimes g)(1) = I(-1)g(1 - -1) + I(0)g(1 - 0) + I(1)g(1 - 1) \dots$$

Point Spread Function & flip

$$(I \otimes g)(x) = \int_{x'} I(x') g(x - x') dx'$$

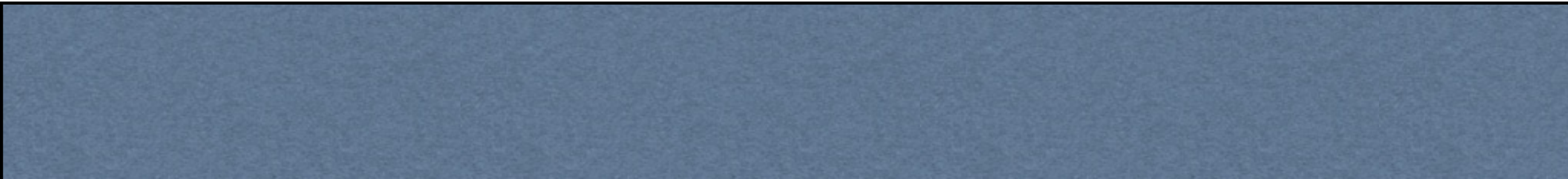


- Remember: it's always the inverse transform
- Forward model: light goes from x to $x+x'$
- Backward model: light at x comes from $x-x'$

Questions?



Symmetry



Commutativity

$$I \otimes g = g \otimes I$$

Commutativity

$$I \otimes g = g \otimes I$$

$$(I \otimes g)(x) = \int_{x'} I(x') g(x - x') dx'$$

use $y=x-x'$

$$\int_y \left| \frac{dy}{dx'} \right| = 1 \mid I(x' = -y + x) g(y) dy$$

It doesn't matter which one you flip!

Similar to probability situation, as long as the sum is x



Nitty-gritty

stuff



Our images aren't infinite

Our images aren't infinite

- Zero (black) padding
- Edge padding
- Mirror padding

Edge Padding

```
def clipX(im, x): return min(width(im)-1, max(x, 0))
def clipY(im, y): return min(height(im)-1, max(y, 0))
def getSafePix(im, y, x):
    return im[clipY(im, y), clipX(im, x)]
```



Centering the kernel

- Our images are defined with 0,0 in the upper left corner
- Kernels are usually assumed to be centered in the center

Normalization

- As a rule of thumb, you want kernels to be normalized when you want the output to preserve the overall brightness of the image.
- But not always (e.g. gradient)

Best input to debug convolution

- Impulse



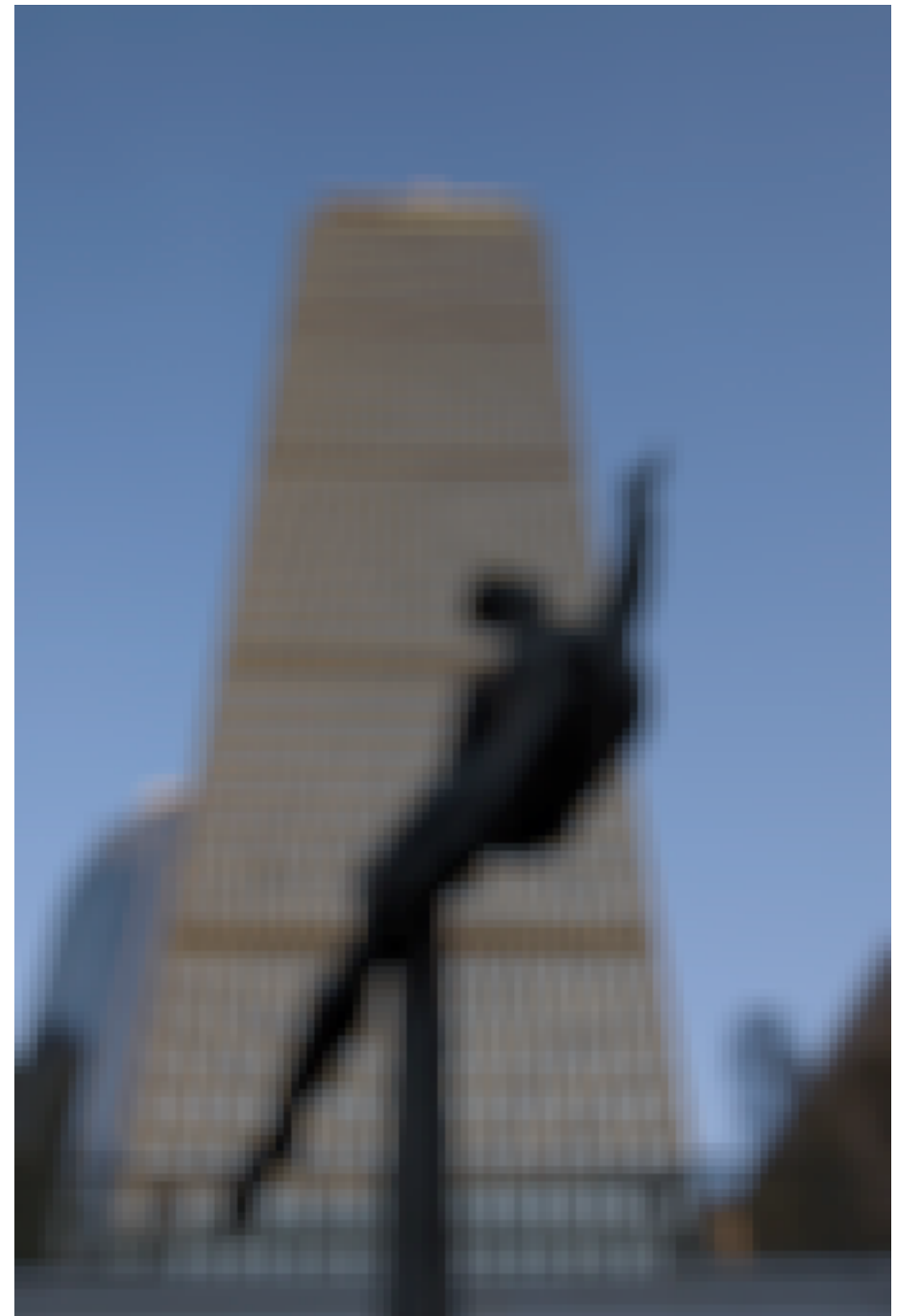
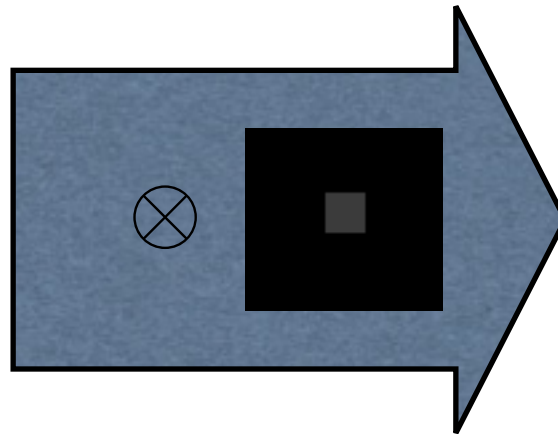
Blur zoo



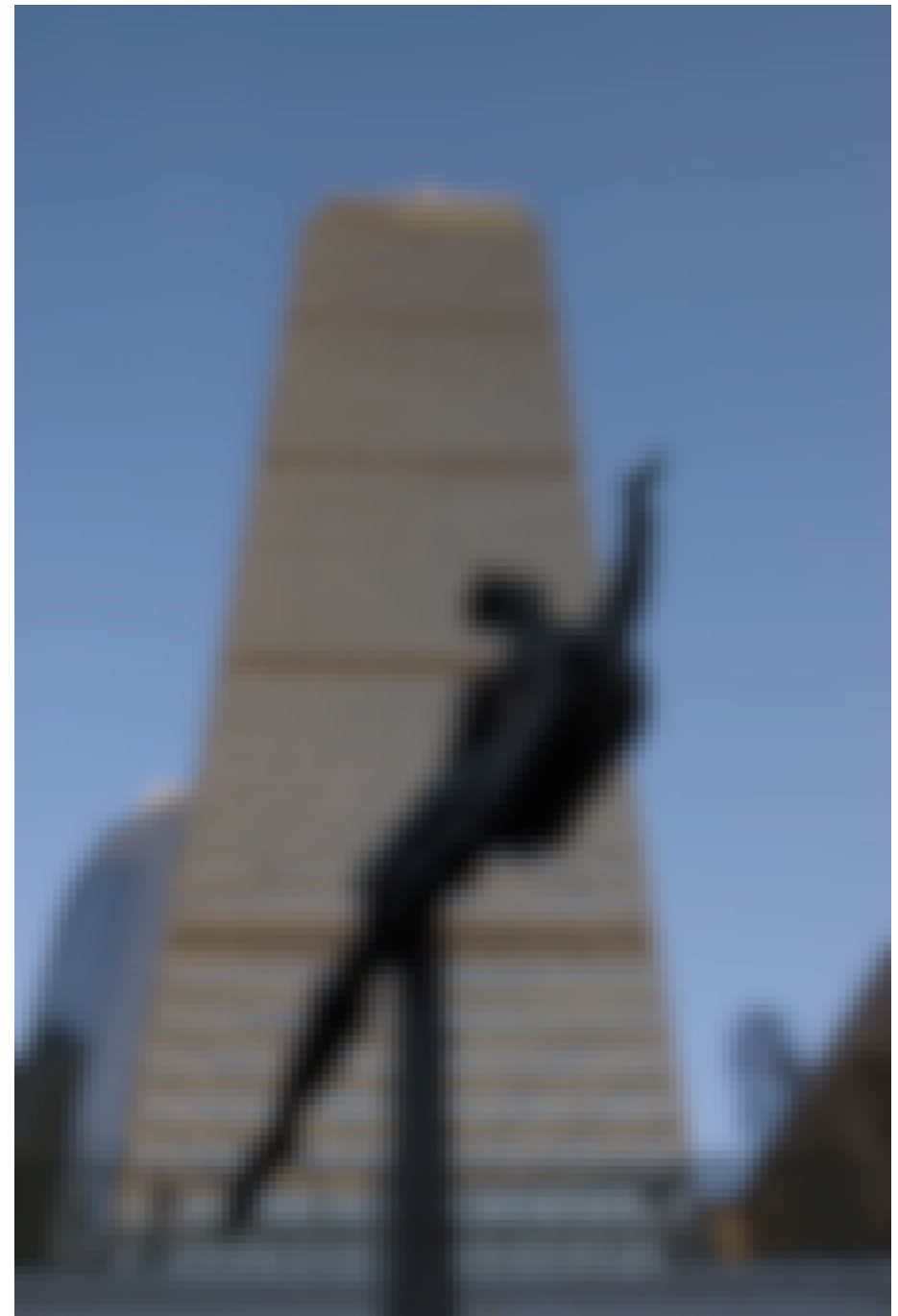
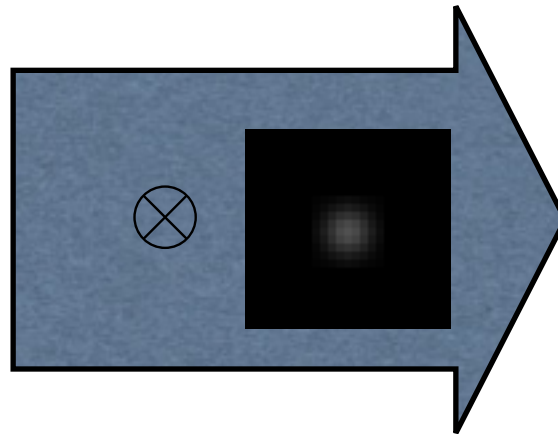
Demo applet

- <http://graphics.stanford.edu/courses/cs178/applets/convolution.html>

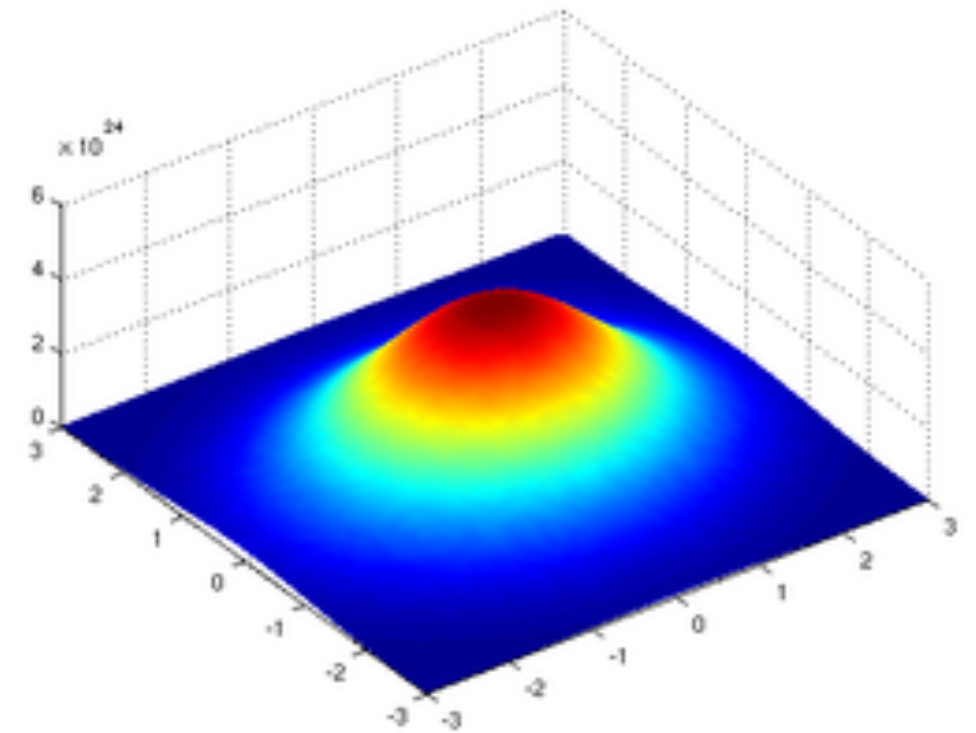
Box filter



Nice and smooth: Gaussian



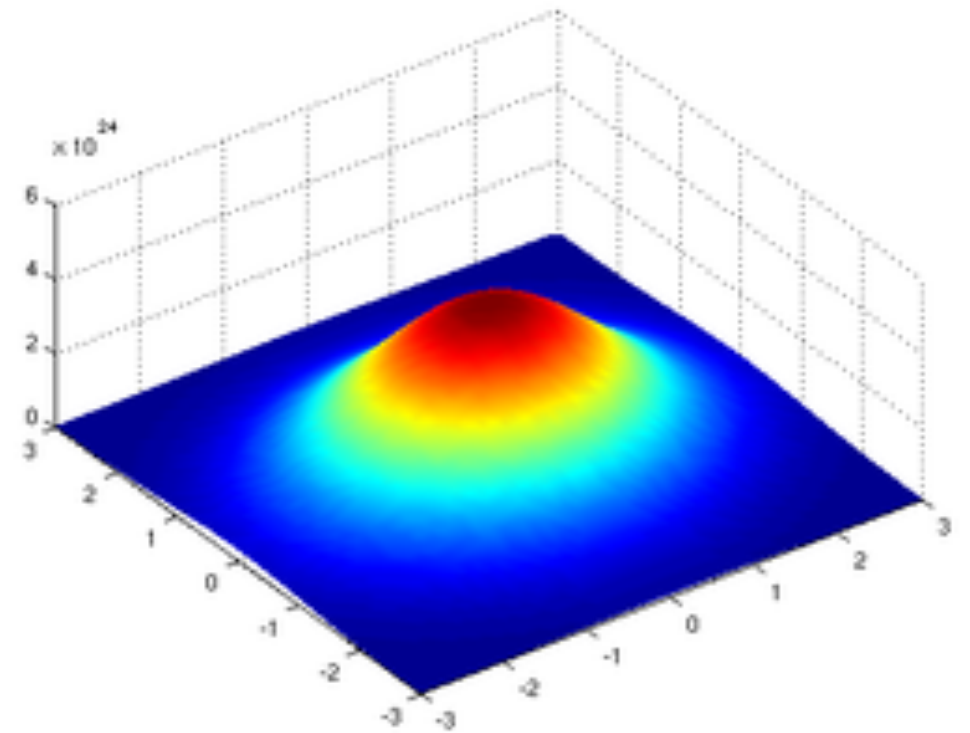
Gaussian formula



http://en.wikipedia.org/wiki/Gaussian_function

Gaussian formula

$$ae^{-\frac{r^2}{2\sigma^2}}$$

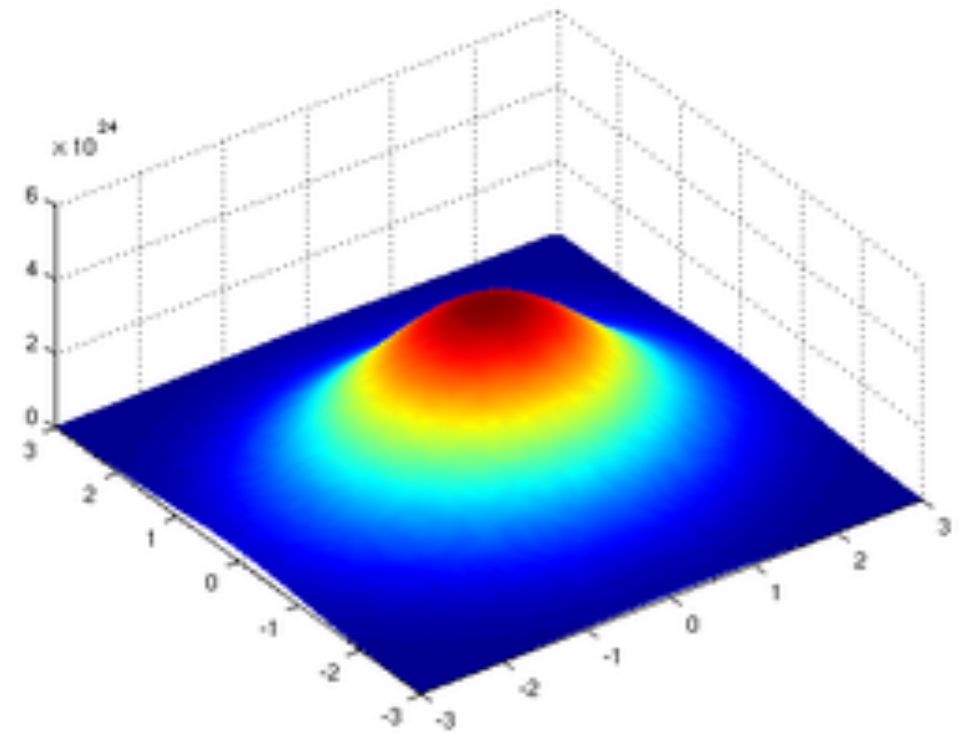


http://en.wikipedia.org/wiki/Gaussian_function

- **r** is the distance to the center
- **a** is a normalization constant
 - I usually just normalize my kernels after the fact
- **σ** is the standard deviation and controls the

Gaussian formula

$$ae^{-\frac{r^2}{2\sigma^2}}$$



http://en.wikipedia.org/wiki/Gaussian_function

- Gaussians have an infinite support
 - >0 everywhere
- But are often truncated
 - I recommend 3σ

Truncation

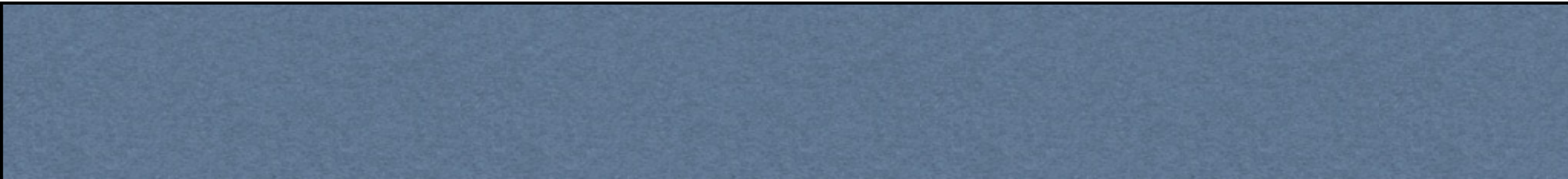
- For computational tractability/efficiency
- consider Gaussian is zero beyond 3sigma

Questions?





Sharpening



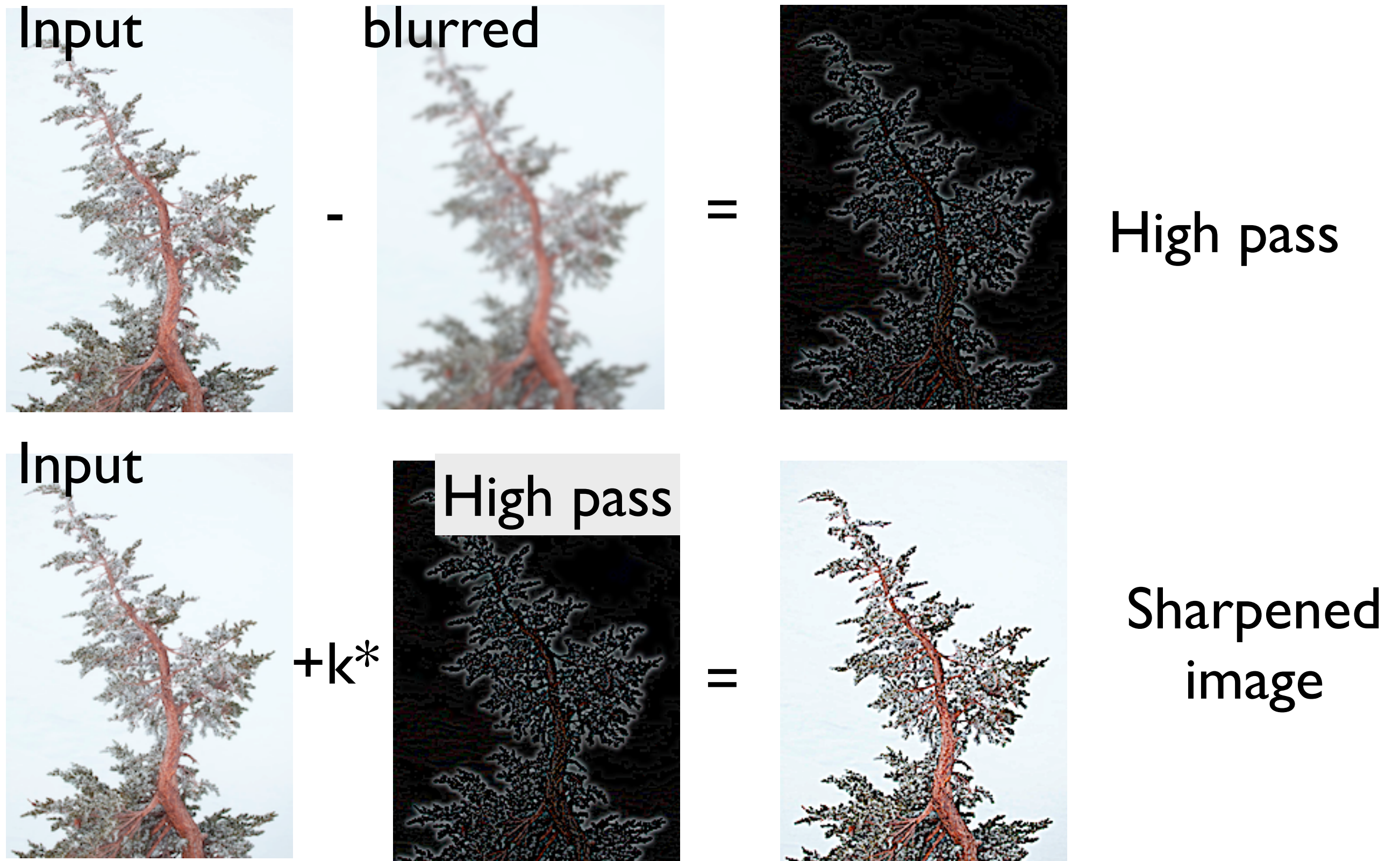
How can we sharpen?

- Blurring was easy
- Sharpening is not as obvious

How can we sharpen?

- Blurring was easy
- Sharpening is not as obvious
- Idea: amplify the non-blurring part of an image
- $\text{out} = \text{input} + k * (\text{input} - \text{blur}(\text{input}))$

Sharpening



Sharpening: kernel view

- Recall

$$f' = f + k * (f - f \otimes g)$$

- f is the input
- f' is a sharpened image,
- g is a blurring kernel
- k is a scalar controlling the strength of sharpening

Sharpening: kernel view

- Recall

$$f' = f + k * (f - f \otimes g)$$

- Denote δ the Dirac kernel (pure impulse)

$$f = f \otimes \delta$$

Sharpening: kernel view

- Recall

$$f' = f + k * (f - f \otimes g)$$

$$f' = f \otimes \delta + k * (f \otimes \delta - f \otimes g)$$

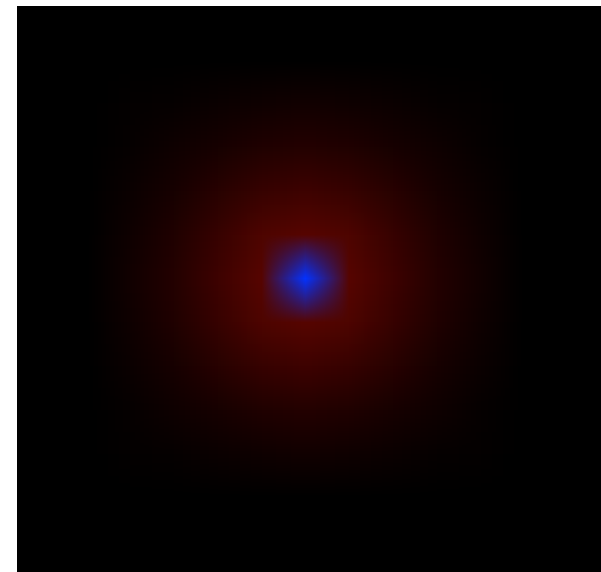
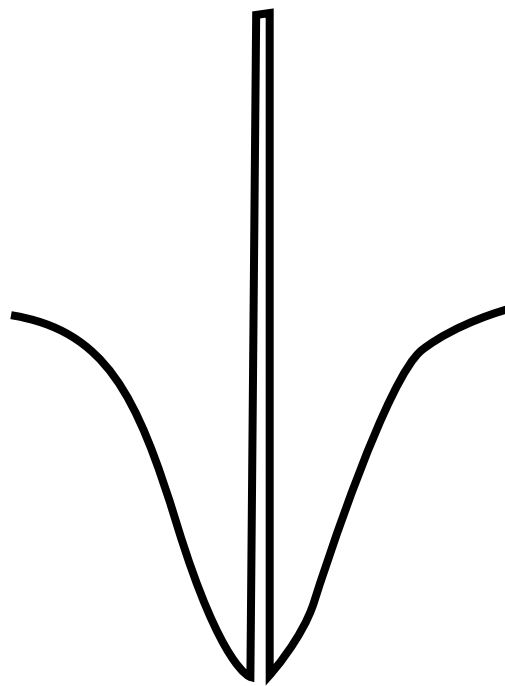
$$f' = f \otimes ((k + 1)\delta - kg)$$

- Sharpening is also a convolution

Sharpening kernel

- Note: many other kernels exist
(just like we saw multiple blurring kernels)

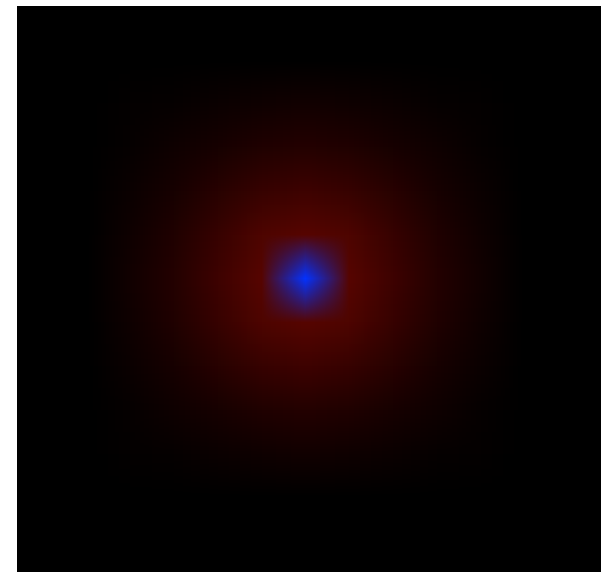
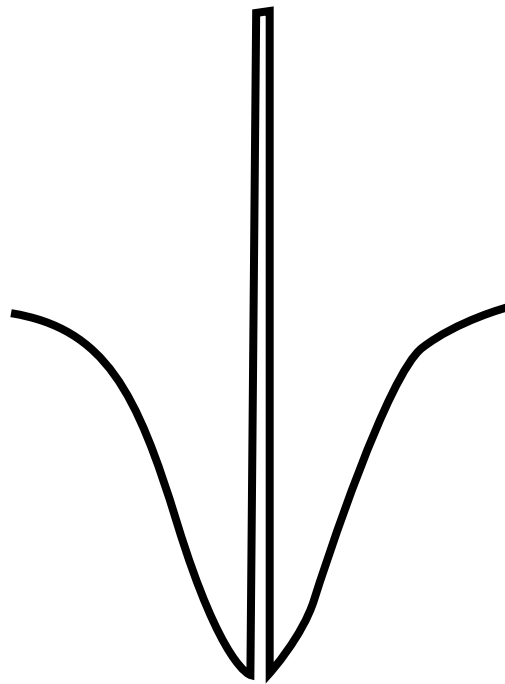
$$f' = f \otimes ((k + 1)\delta - g)$$



blue: positive
red: negative

Intuition

- Amplify the difference between a pixel and its neighbors



blue: positive
red: negative

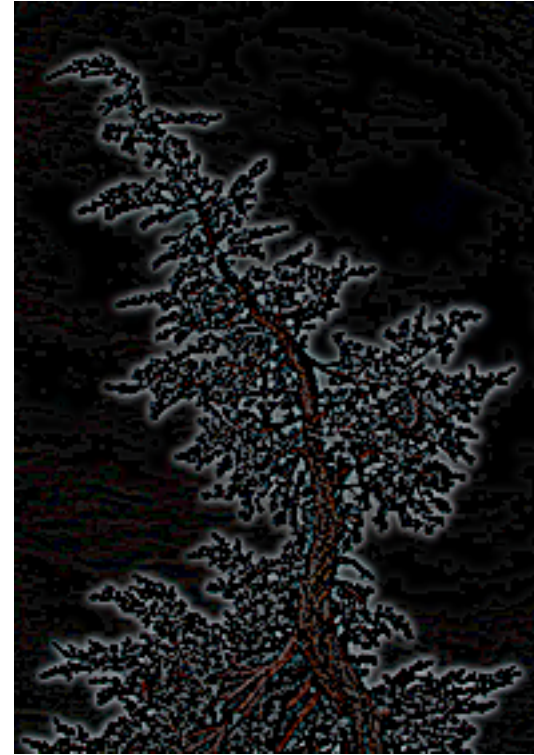
Questions?



-



=



+k*



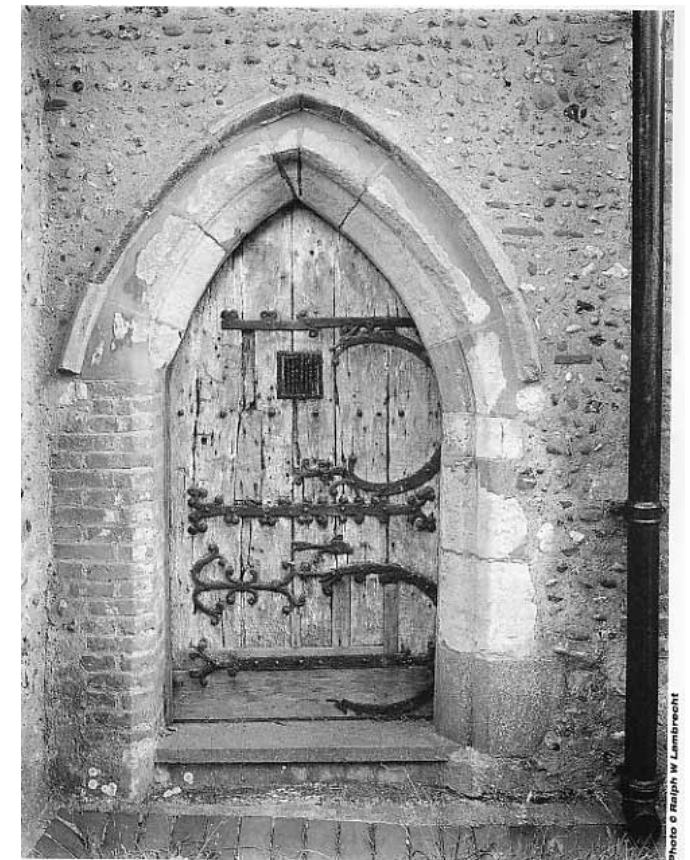
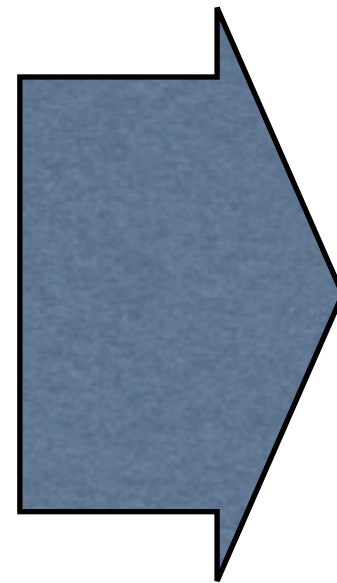
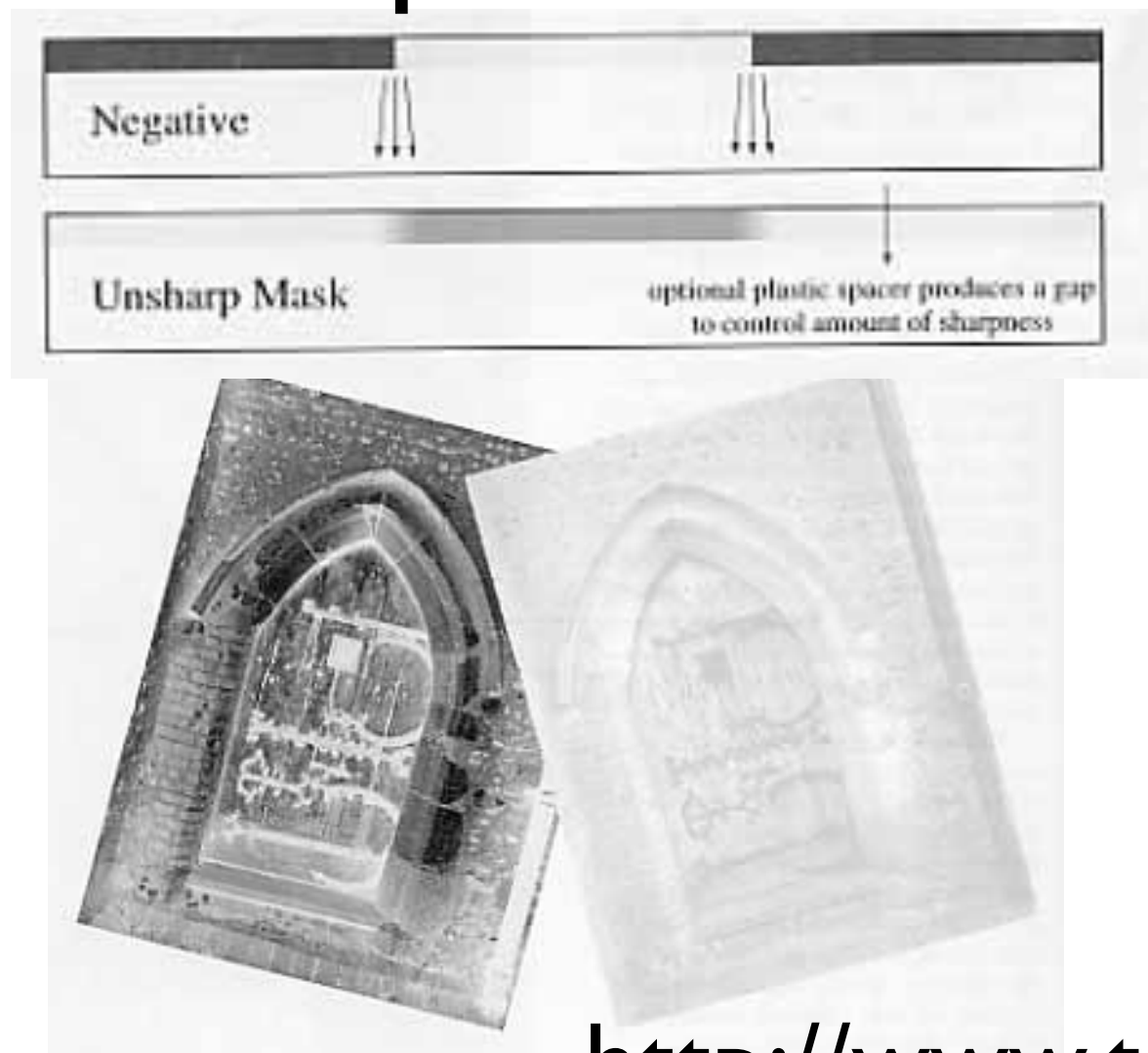
=



**Unsharp
Mask**

Unsharp mask

- Sharpening is often called “unsharp mask” because photographers used to sandwich a negative with a blurry positive film in order to sharpen

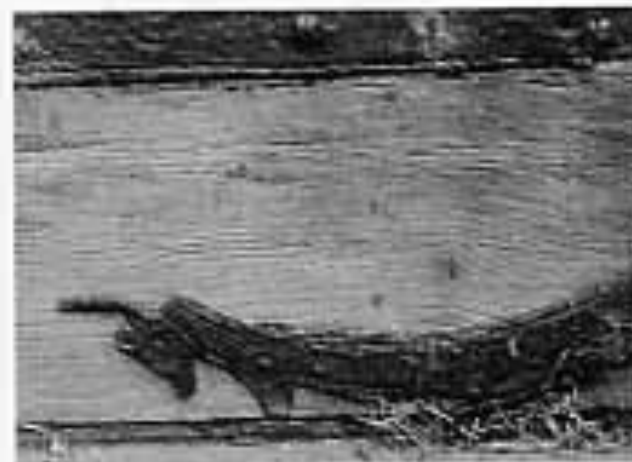


<http://www.tech-diy.com/UnsharpMasks.htm>

Fig.4: The two examples here show a detail of the brick-work to the left of the church door. The one on the left was printed with the negative alone – the one on the right was printed with both negative and mask as a sandwich. The increase in local contrast and edge sharpness is minute, but clearly visible. Grade 2.5 was used for the straight print but increased to 4.5 for the sandwiched image to compensate for the reduced contrast.



Fig.5: These two examples show a detail of the lower right hand side of the church door. Here the difference in sharpness is clearly visible between the (left) negative and (right) sandwich prints.



All photos © Ralph W Lambrecht

Unsharp mask

- http://en.wikipedia.org/wiki/Unsharp_masking
- <http://www.largeformatphotography.info/unsharp/>
- <http://www.tech-diy.com/UnsharpMasks.htm>
- <http://www.cambridgeincolour.com/tutorials/unsharp-mask.htm>



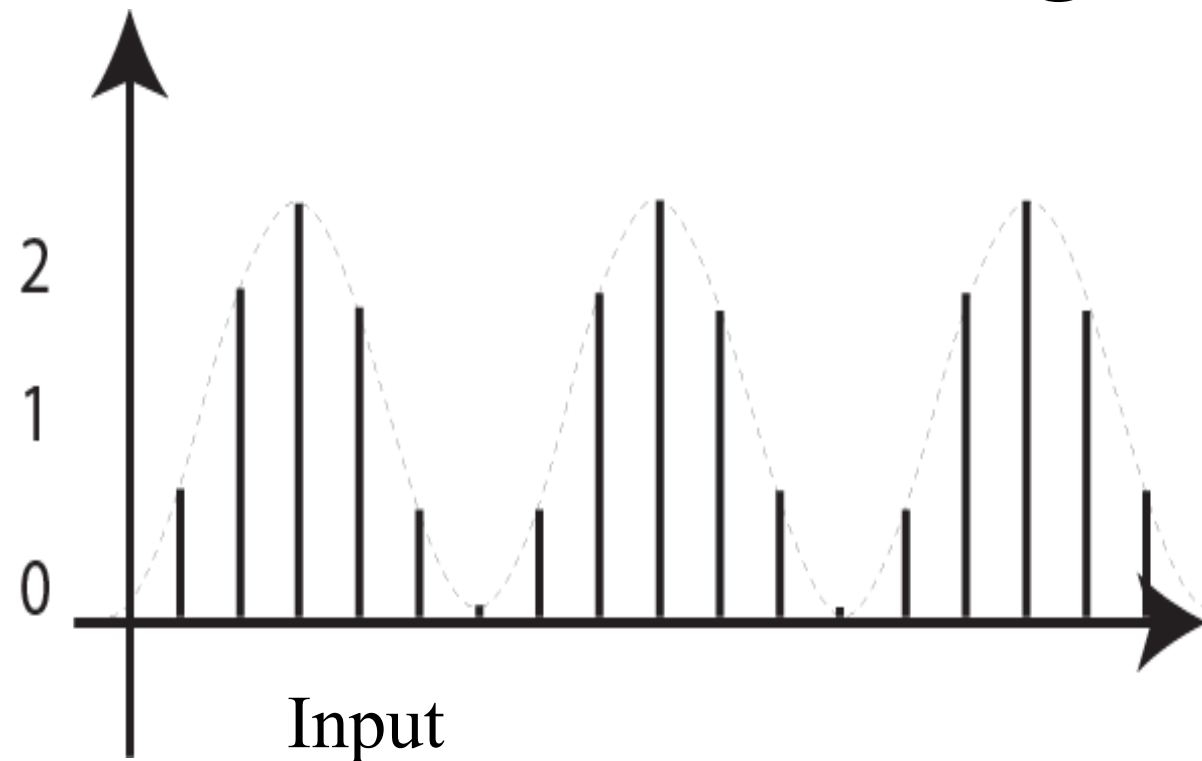
Fourier motivation



Studying convolutions

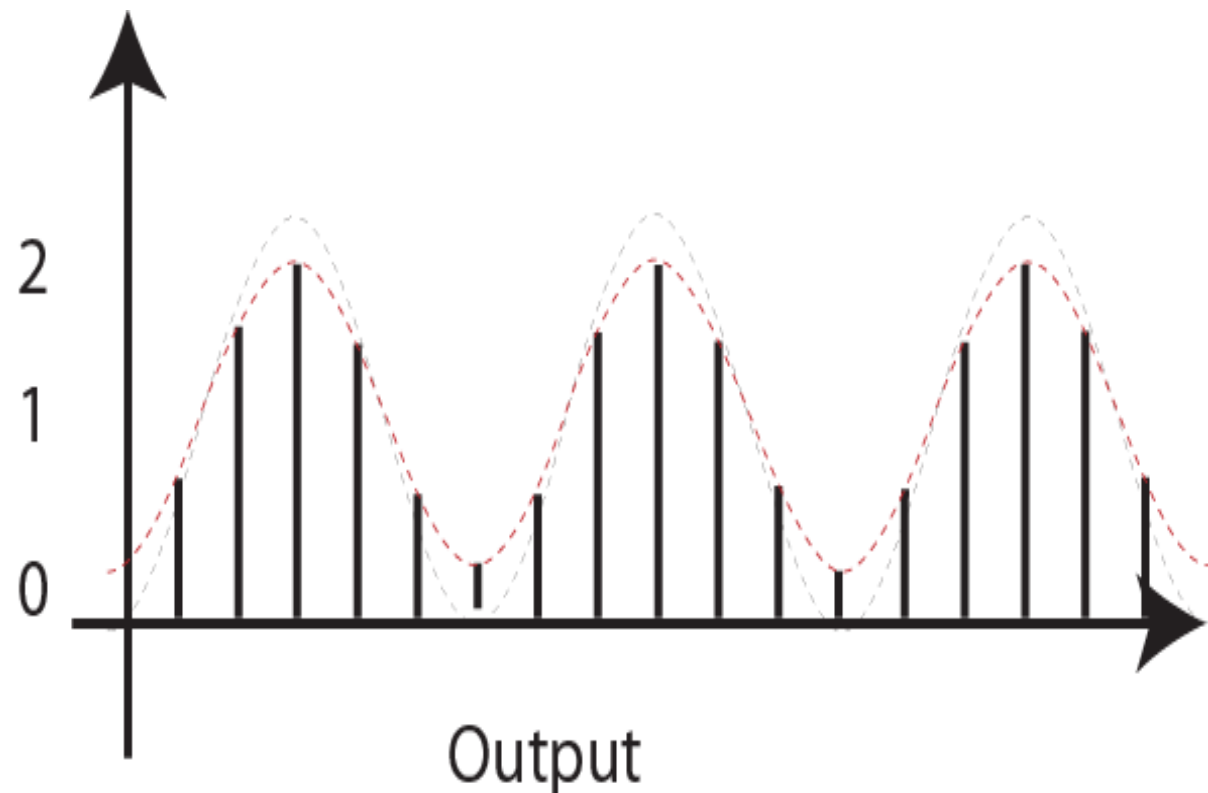
- Convolution is complicated
 - and hard to inverse
- But at least it's linear
$$(f+kg) h = fh +k (gh)$$
- We want to find a better expression
 - Let's study functions whose behavior is simple under convolution

Blurring: convolution




Convolution
sign


Kernel

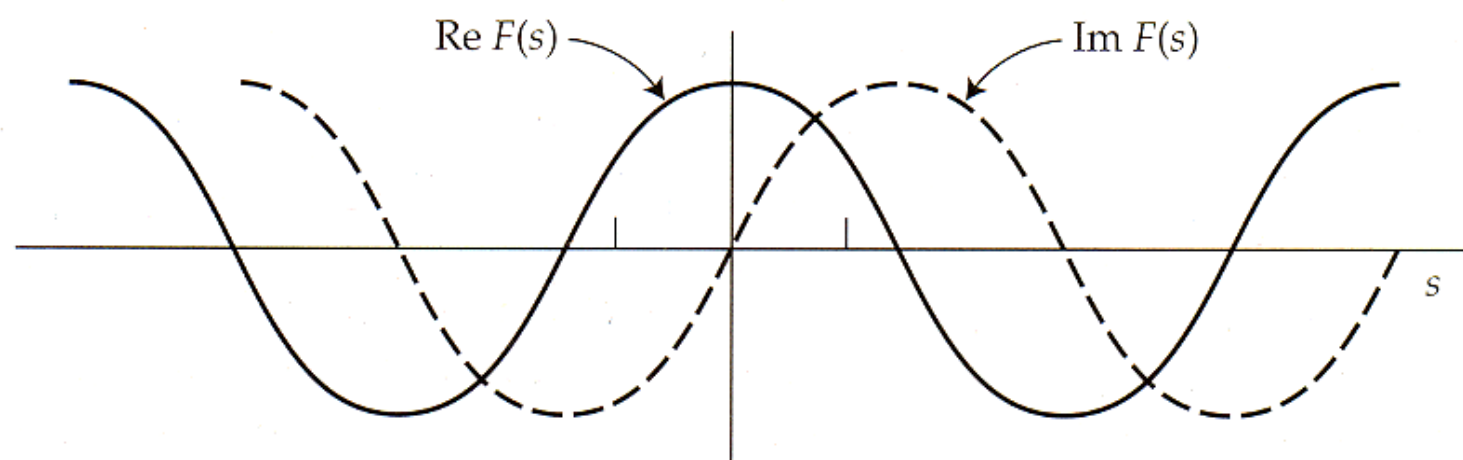
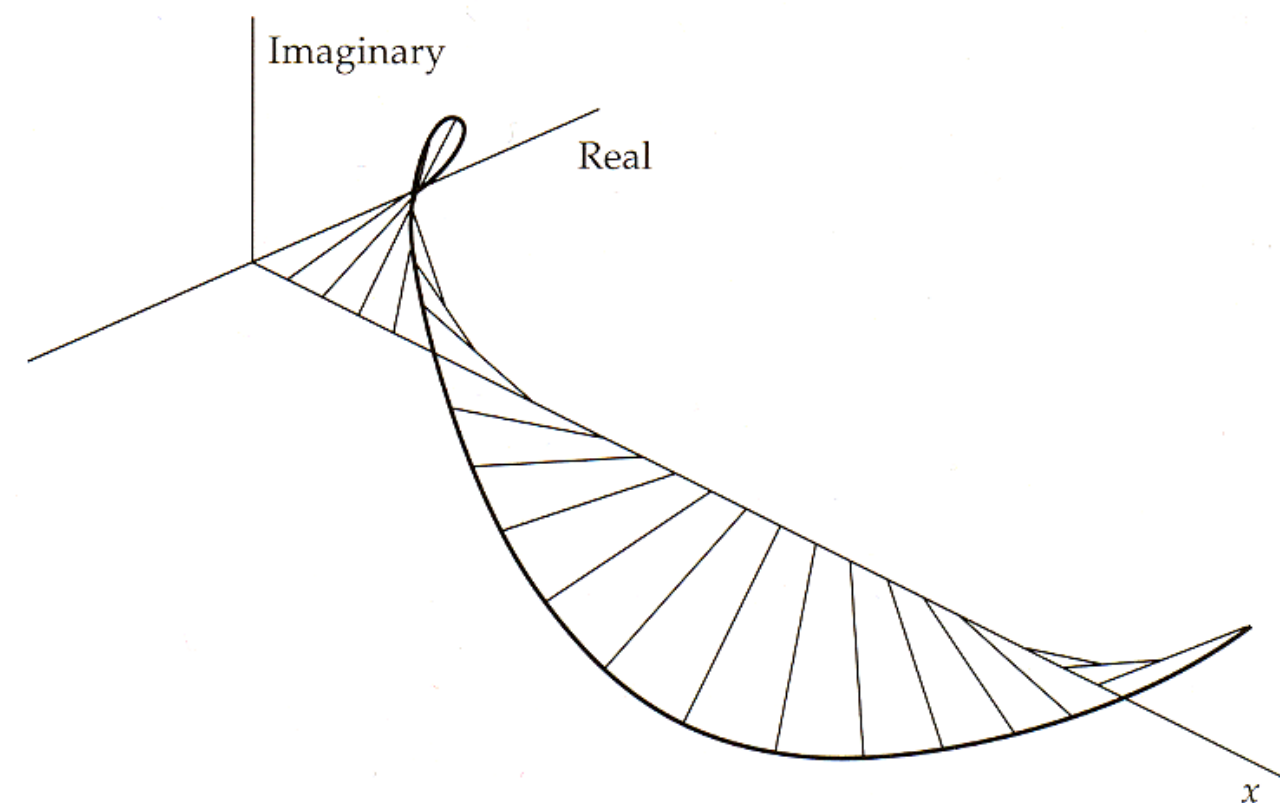


Same shape, just reduced contrast!!!

This is an eigenvector
(output is the input multiplied by a
constant)

Eigenvectors

- More precisely, the eigenvectors are $e^{i\omega x}$
- aka $\cos \omega x + i \sin \omega x$



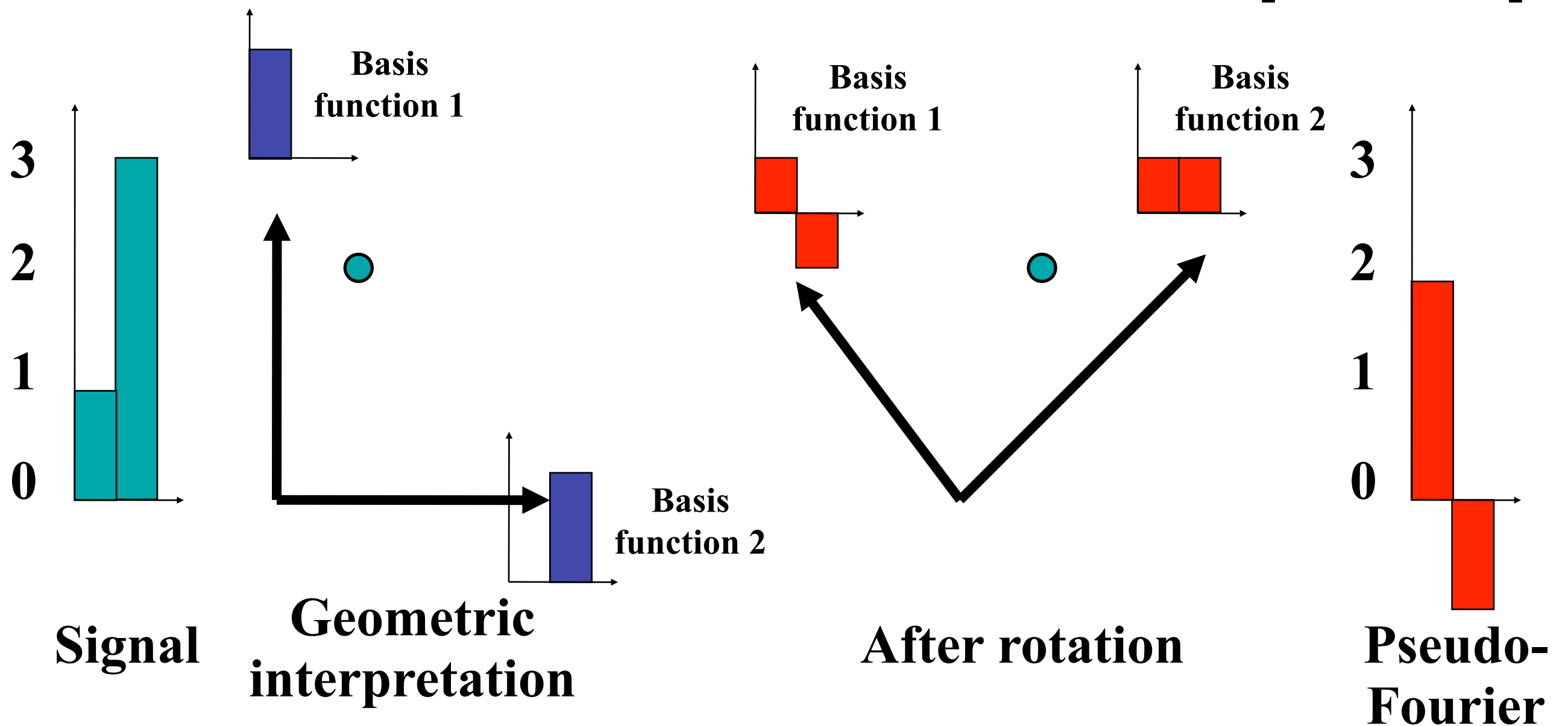
Big Motivation for Fourier analysis

- Sine waves are eigenvectors of the convolution operator

Fourier as change of basis

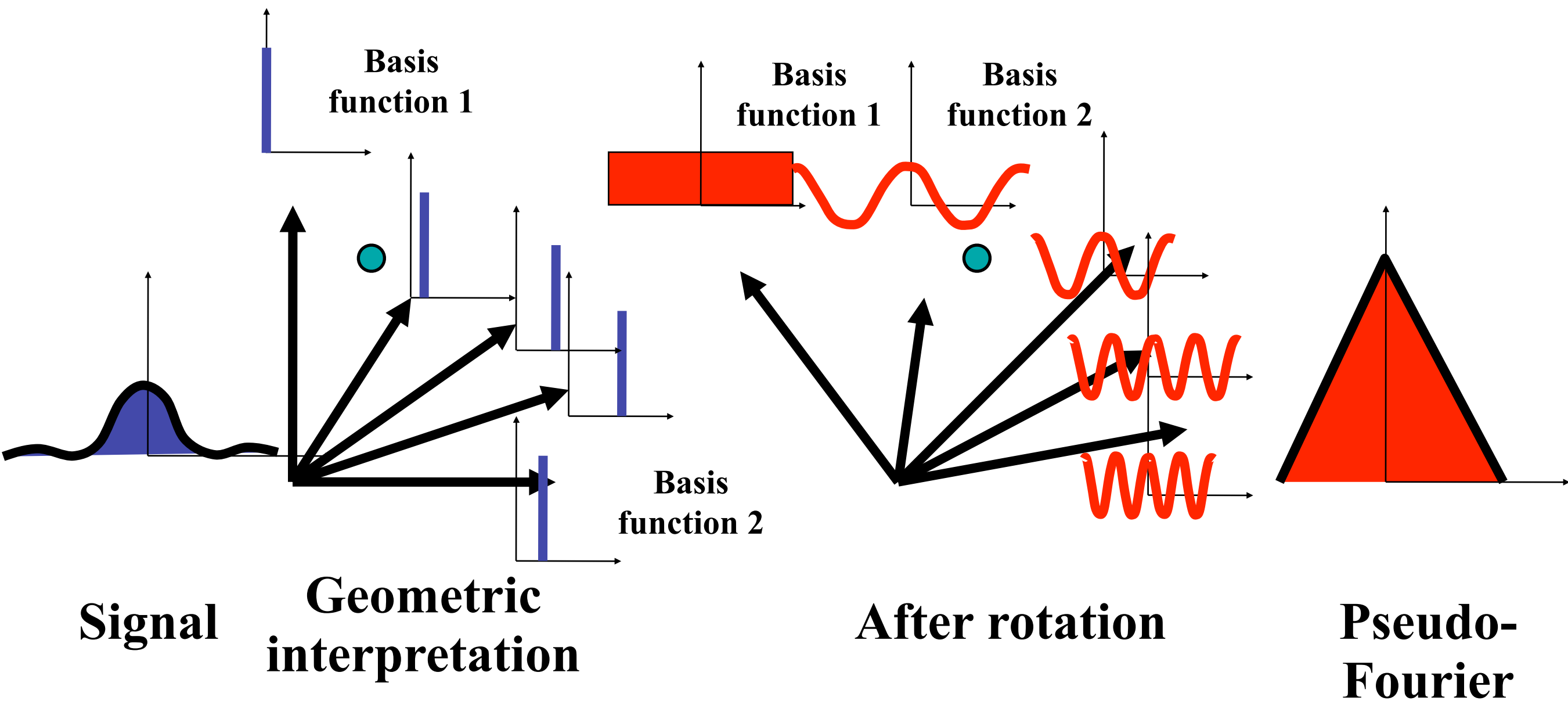
- Shuffle the data to reveal other information
- E.g., take average & difference: matrix

$$\begin{bmatrix} 0.5 & 1 \\ 0.5 & -1 \end{bmatrix}$$

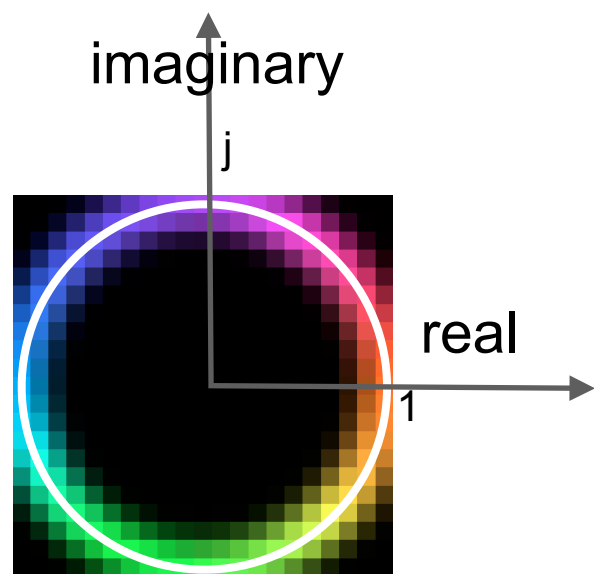


Fourier as change of basis

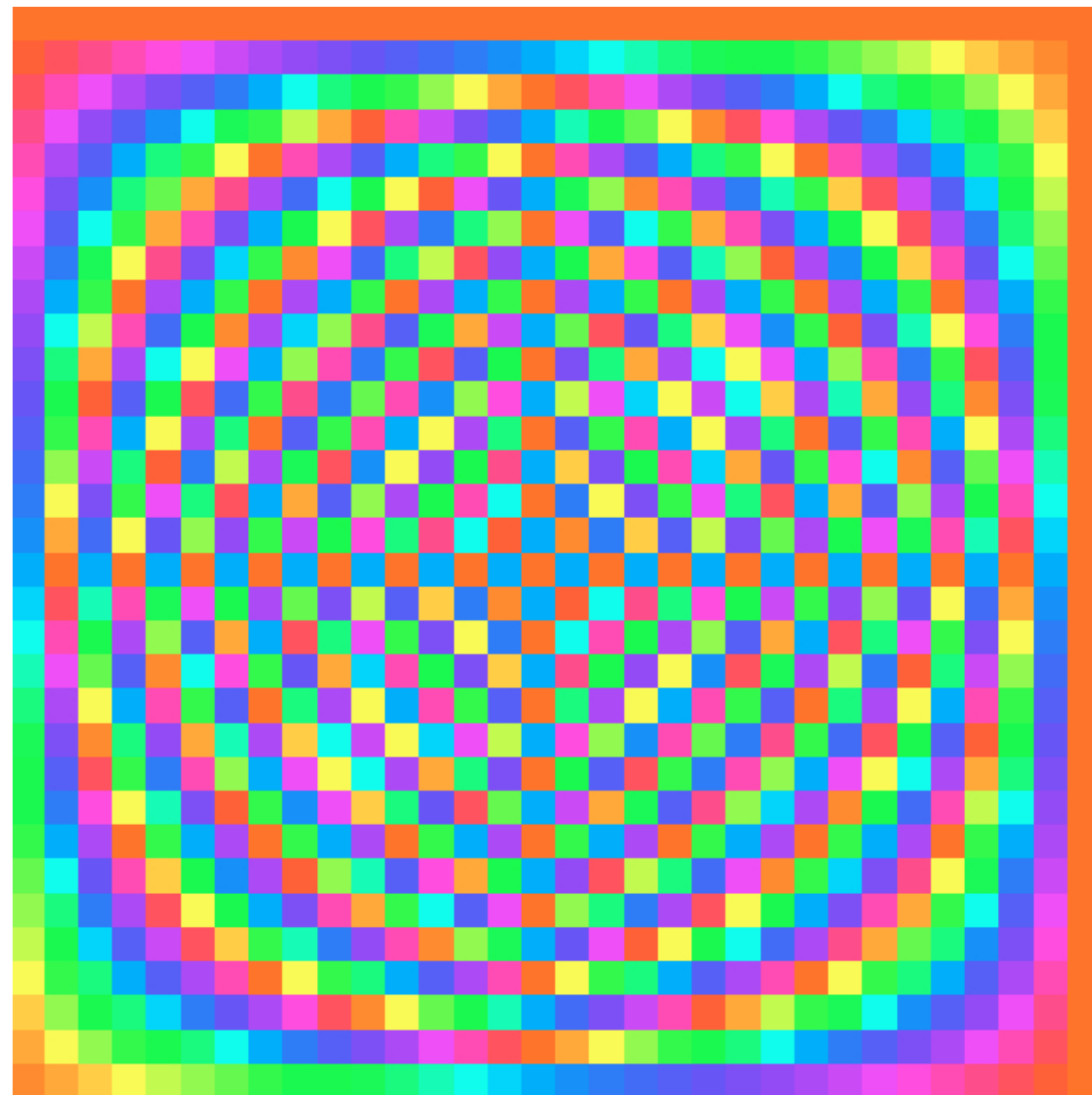
- Same thing with infinite-dimensional vectors



Fourier transform visualization



color key



Fourier transform matrix



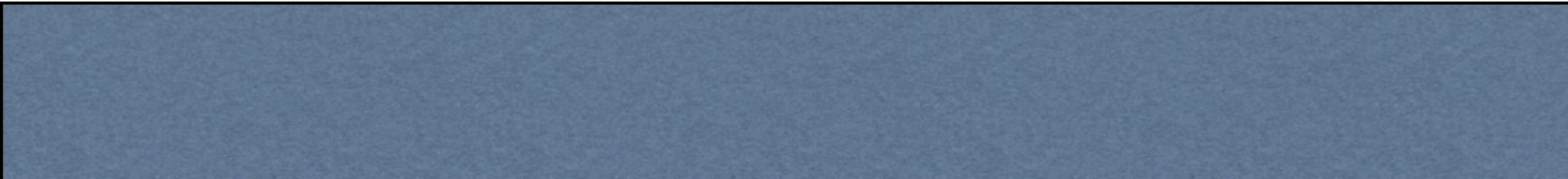
input signal

$$F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k,l] e^{-\pi i \left(\frac{km}{M} + \frac{ln}{N} \right)}$$

Question?

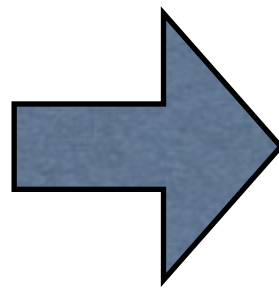


Sharpening++

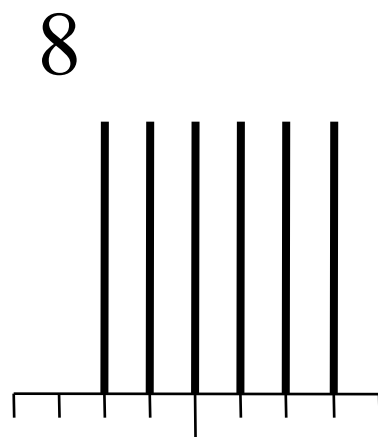


Problem with excess

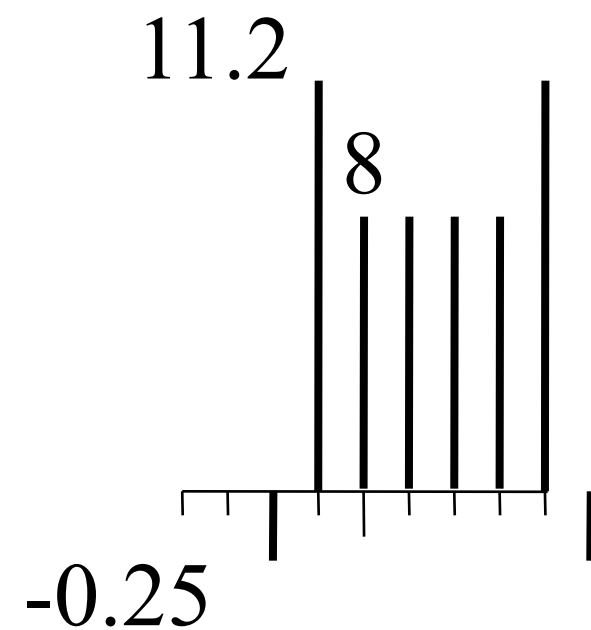
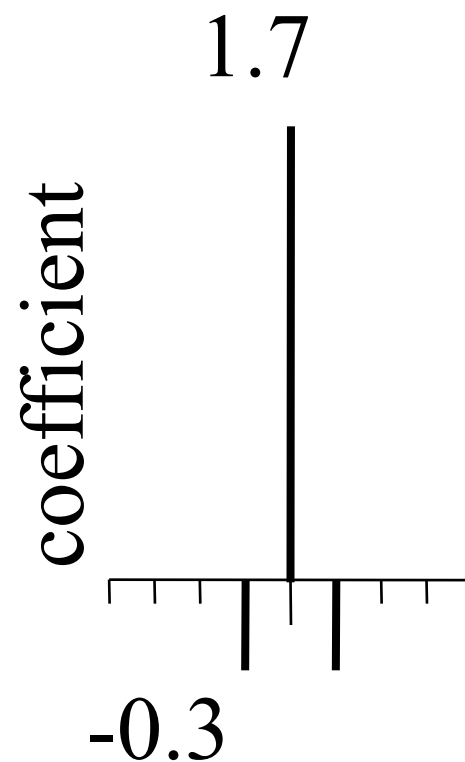
- Haloes around strong edges



oversharpening



original



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

Bells and whistles

- Apply mostly on luminance
- Old Clarity in Adobe Camera Raw
 - As far as I understand, apply only for mid-tones
 - Avoids haloes around black and white points
- Only apply at edges
 - To avoid the amplification of noise
- Sharpening chrominance as well
 - But with very large blur

Lightroom demo



Cost



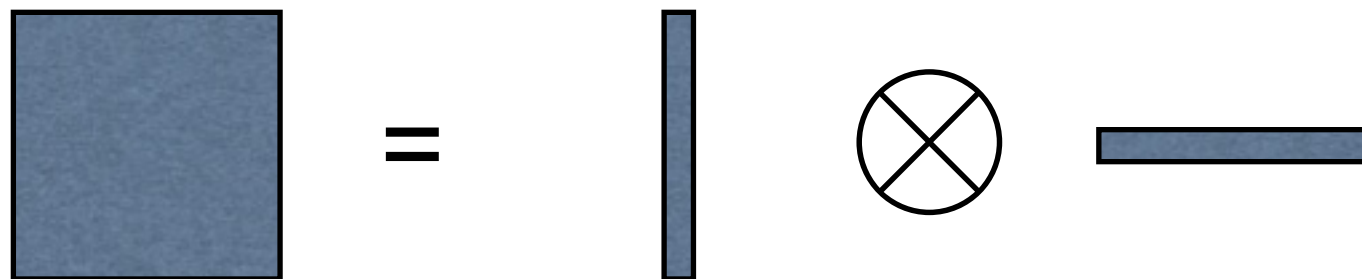
Convolution cost?

- Set output image to zero
- For y, x in output image
 - for y', x' in kernel
 - $\text{out}[y, x] += \text{input}[y+y', x+x'] * \text{kernel}[y', x']$
- Cost?
 - - $\text{input.width} * \text{input.height} * \text{kernel.width} * \text{kernel.height}$

Separable filters

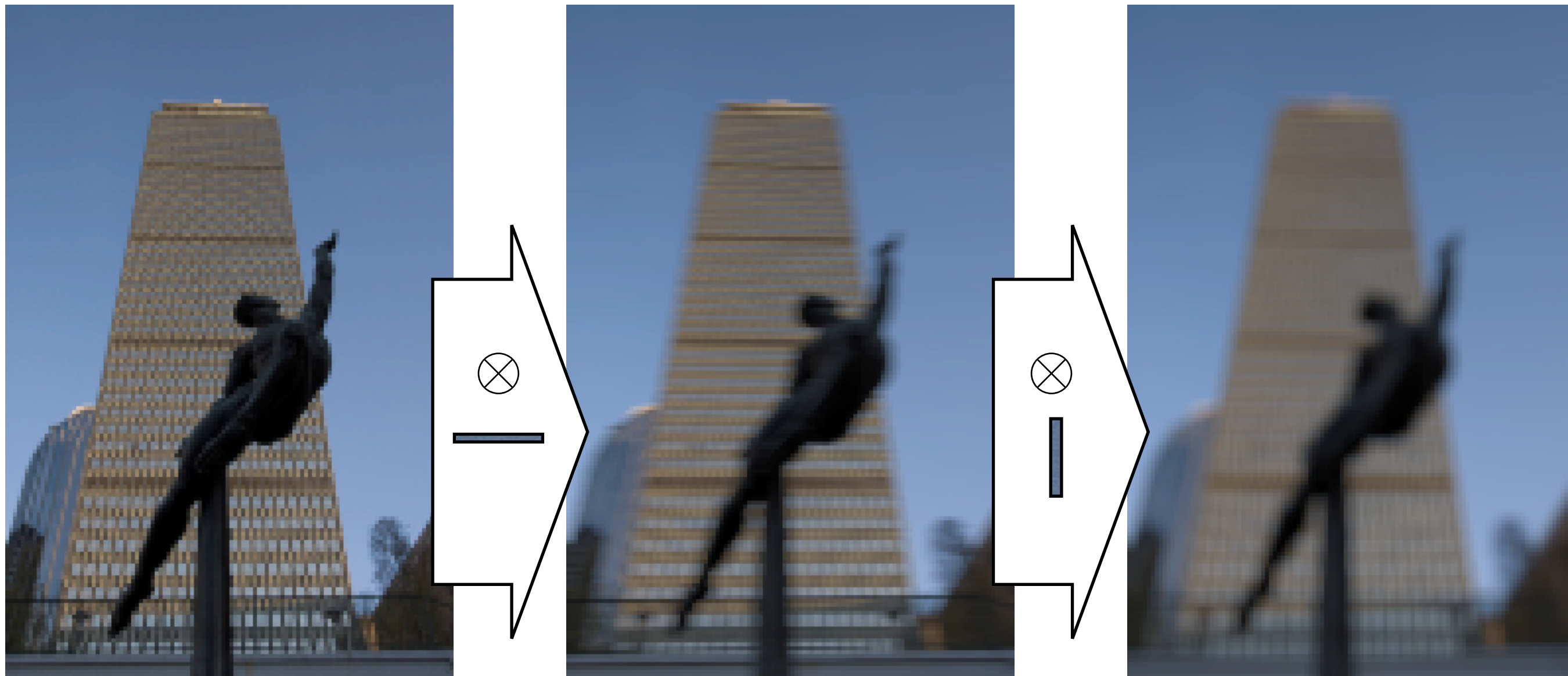
Separability

- The kernel can be decomposed into the convolution of a horizontal and a vertical filter.
- Example: box
 - $g'(x) = 1$ if $(-k < x < k)$, 0 otherwise
 - $g(y, x) = g(y) \otimes g(x)$
 - (separability doesn't require the two 1D kernels to be the same, but it's the case here)



Separable blur

- First blur horizontally using $g(x)$
- Then blur vertically using $g(y)$



Separable Convolution cost?

- For y, x in output image
 - for y', x' in kernel
 - $\text{out}[y, x] += \text{input}[y+y', x+x'] * \text{kernel}[y', x']$
- Horizontal cost?
 - $\text{input.width} * \text{input.height} * \text{kernel.width}$
- Vertical cost?
 - $\text{input.width} * \text{input.height} * \text{kernel.height}$
- Total
 - $\text{input.width} * \text{input.height} * (\text{kernel.height} + \text{kernel.width})$
 - instead of $\text{input.width} * \text{input.height} * (\text{kernel.height} * \text{kernel.width})$

Good news

- Gaussians are separable too
- See Pset 3!

Oriented filters

Gradient: finite difference

- horizontal gradient $[-1, 1]$
- vertical gradient: $[-1], [1]$

Gradient: finite difference

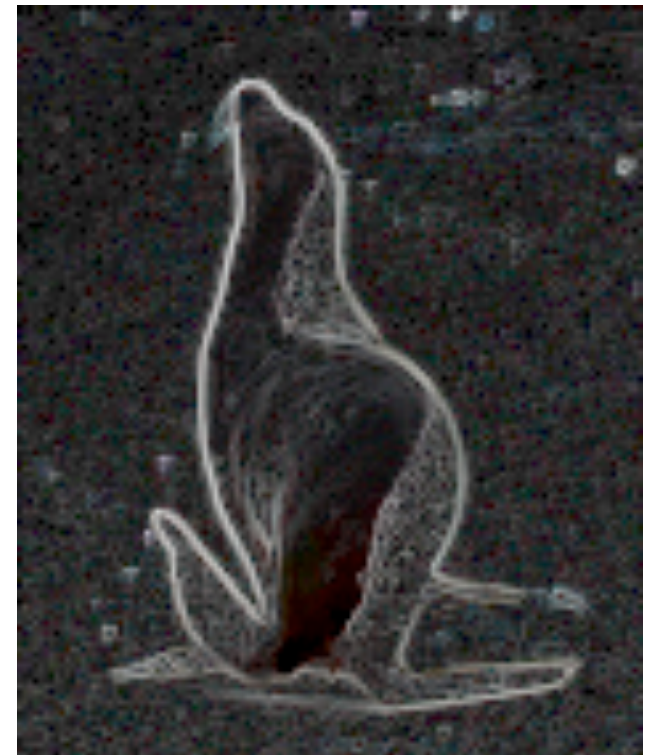
- horizontal gradient $[-1, 1]$
- vertical gradient: $[-1, 1]$



Horizontal
gradient
(absolute
value)



Vertical
gradient
(absolute
value)



Gradient
magnitude

Gradient

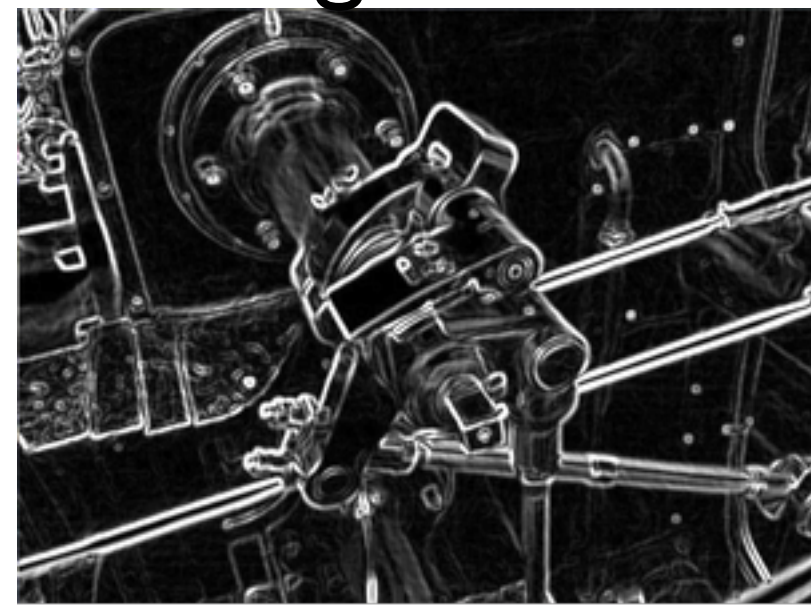
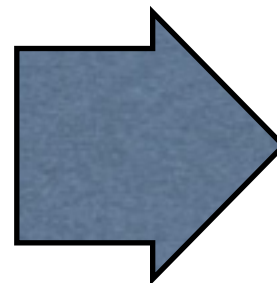
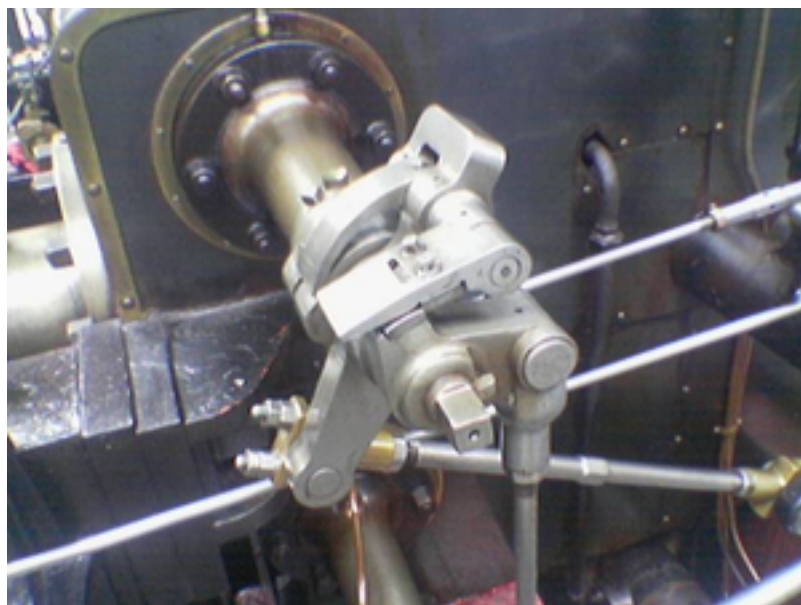
- e.g. Sobel
 - http://en.wikipedia.org/wiki/Sobel_operator

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

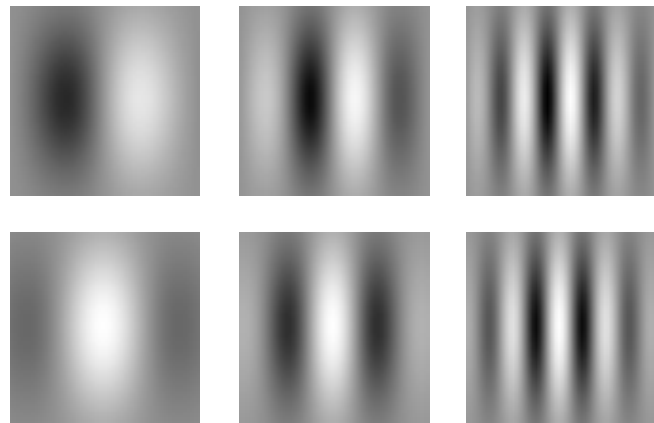
Horizontal gradient

Vertical gradient

Magnitude

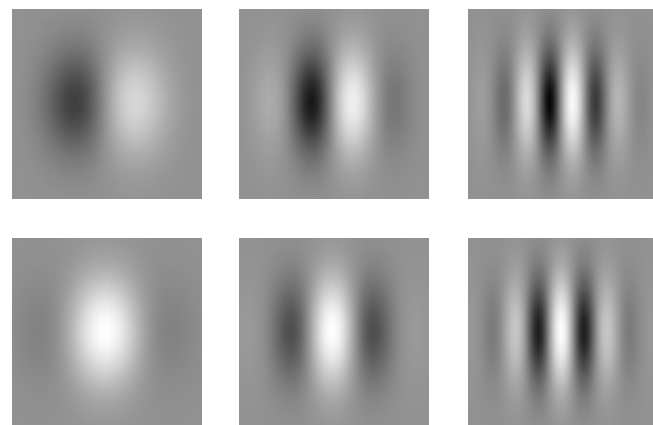


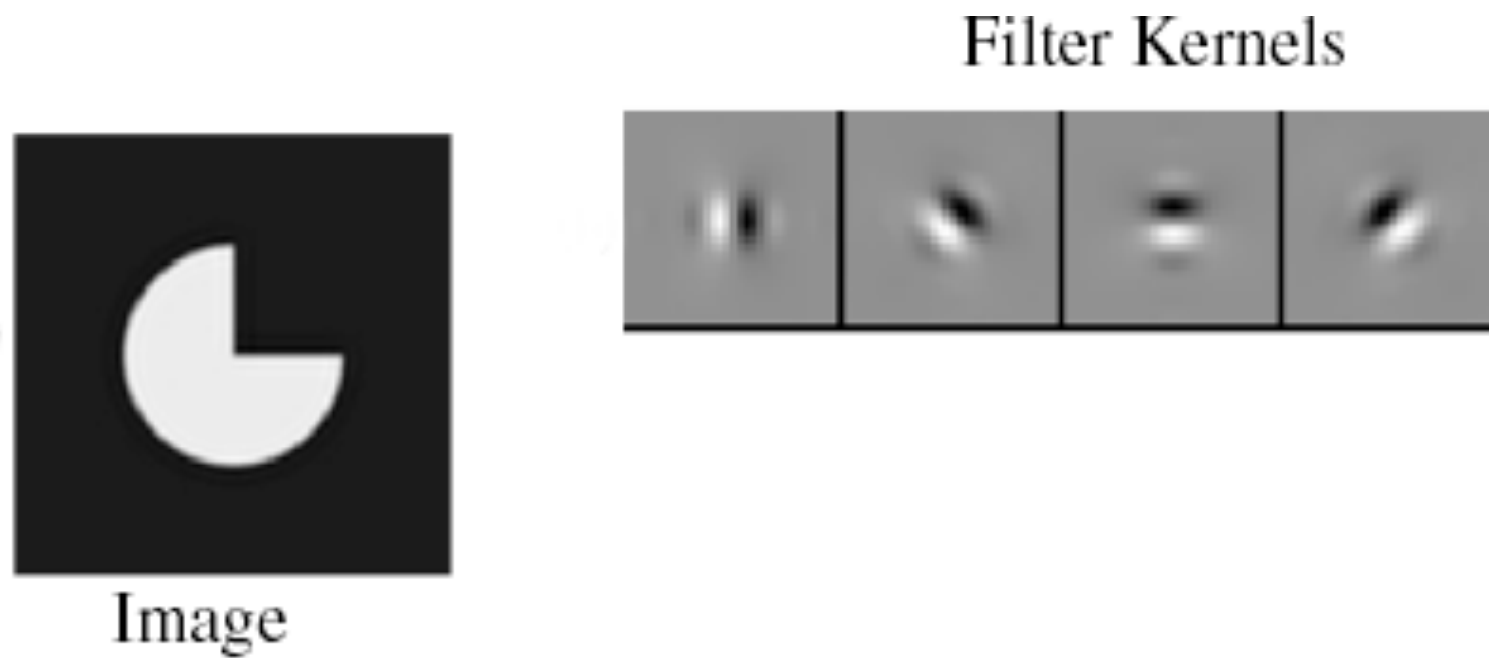
Oriented filters



Gabor filters at different
scales and spatial frequencies

top row shows anti-symmetric
(or odd) filters, bottom row the
symmetric (or even) filters.





Filtered images



Reprinted from “Shiftable MultiScale Transforms,” by Simoncelli et al., IEEE Transactions on Information Theory, 1992, copyright 1992, IEEE

References

- <http://www.youtube.com/watch?v=IW4Reburjpc>
- <http://en.wikipedia.org/wiki/Convolution>
- <http://math.mit.edu/daimp/ConvFlipDrag.html>
- <http://www.jhu.edu/signals/lecture1/frames.html>
- <http://graphics.stanford.edu/courses/cs178/applets/convolution.html>