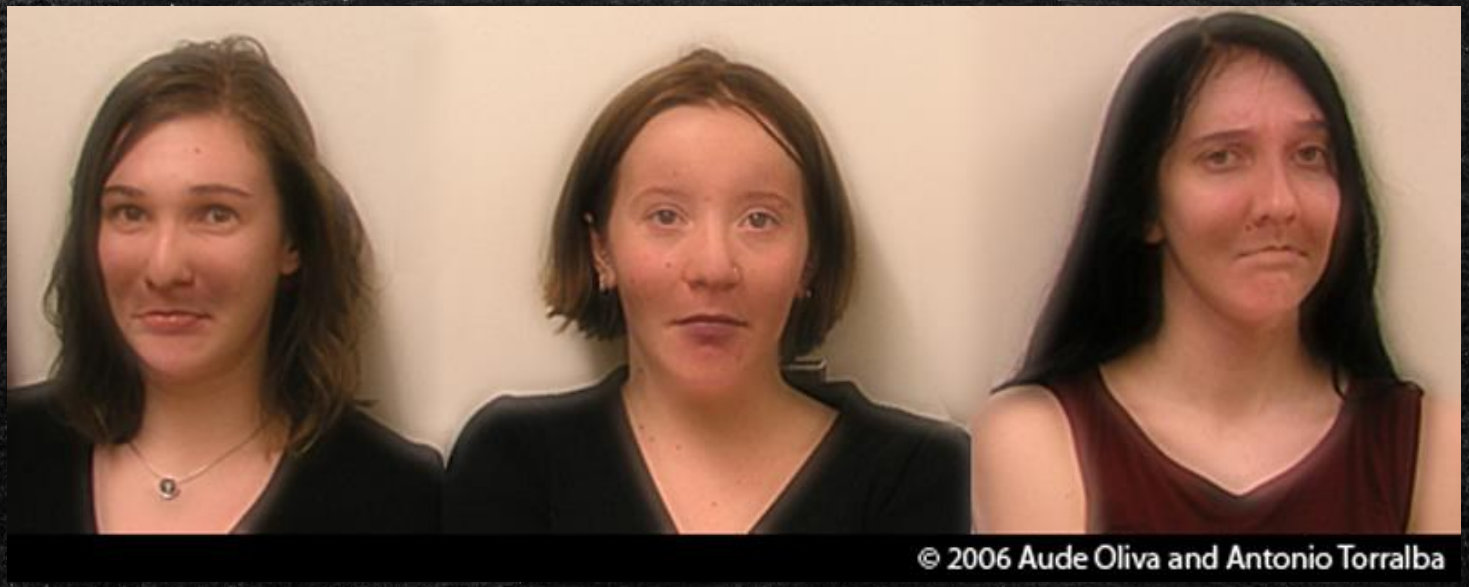
The Power of Image Filtering…

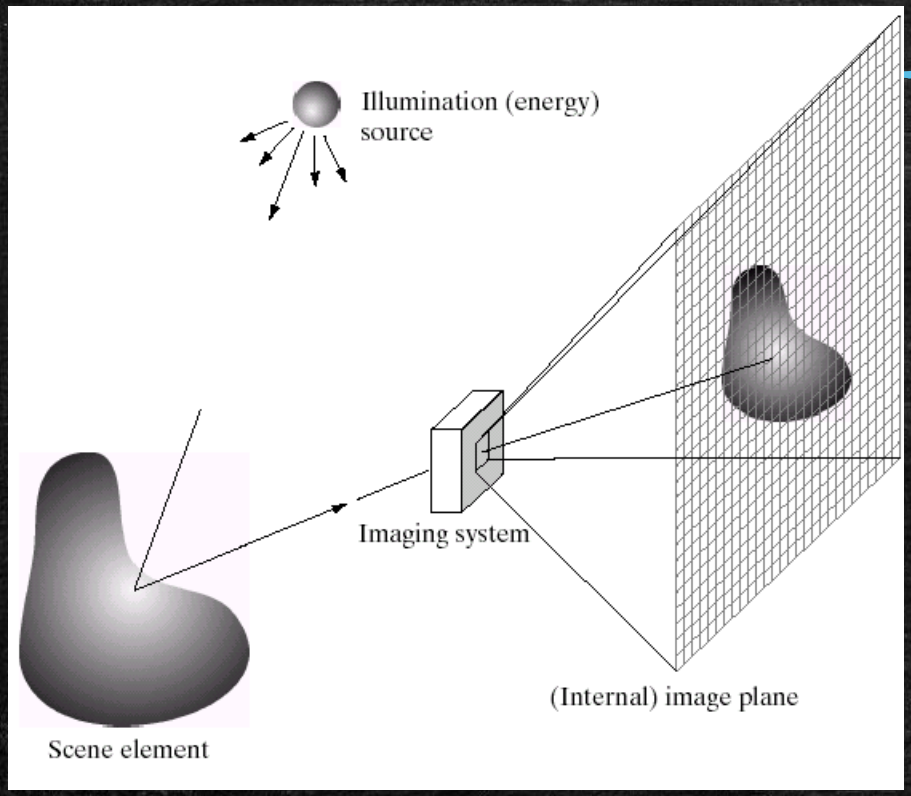
Hybrid Imagery Example

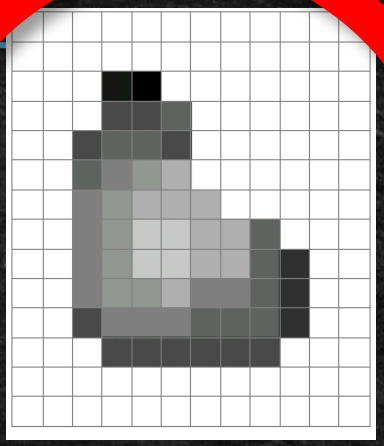


Hybrid Images, Oliva et al., http://cvcl.mit.edu/publications/OlivaTorralb_Hybrid_Siggraph06.pdf

What is an image?

What is an image?

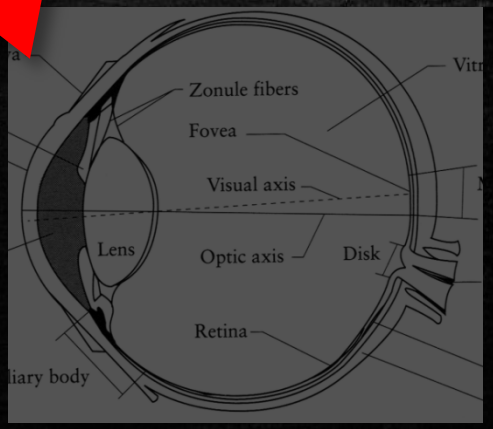




Digital Camera

We'll focus on these in this class

(More on this process later)

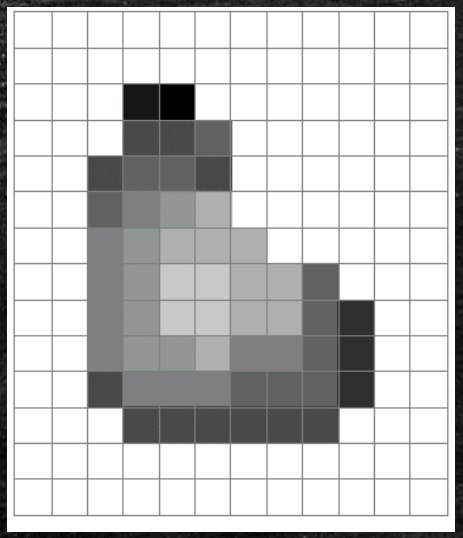


The Eye

Source: A. Efros

What is an image?

- A grid (matrix) of intensity values



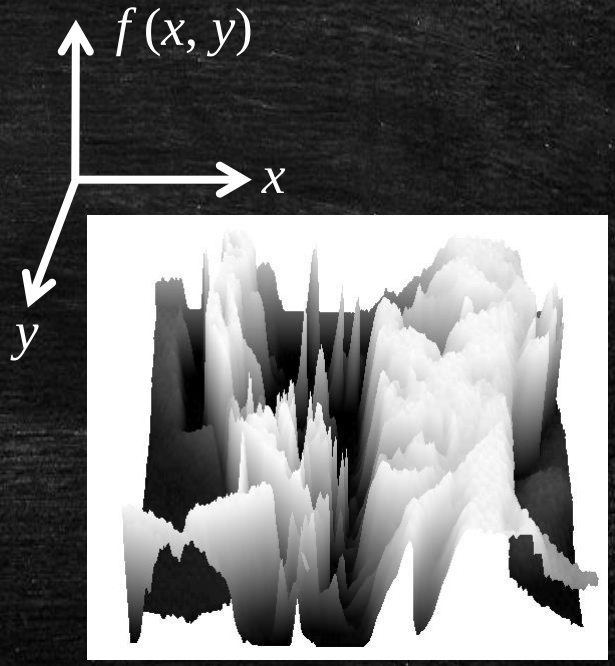
=

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(common to use one byte per value: 0 = black, 255 = white)

What is an image?

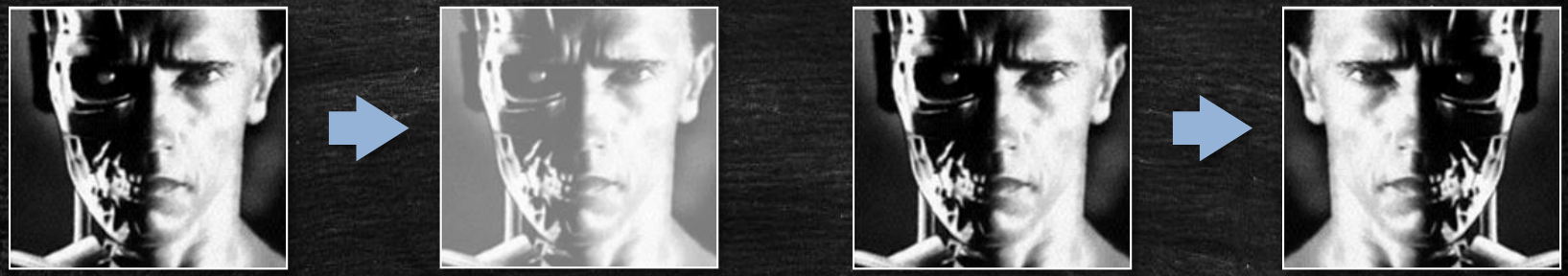
- We can think of a (grayscale) image as a **function**, f, from $\mathbb{R}^2$ to $\mathbb{R}$ (or a 2D *signal*):
  - $f(x,y)$ gives the **intensity** at position (x,y)



  - A **digital** image is a discrete (**sampled, quantized**) version of this function

Image transformations

- As with any function, we can apply operators to an image



$g(x,y) = f(x,y) + 20$

$g(x,y) = f(-x,y)$

- We'll talk about a special kind of operator, *convolution* (linear filtering)

Question: Noise reduction

- Given a camera and a still scene, how can you reduce noise?

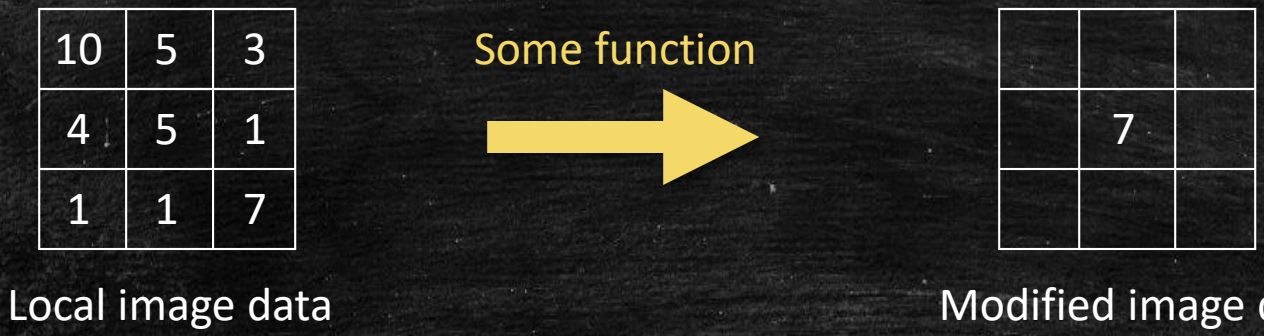
Take lots of images and average them!

What's the next best thing?

Source: S. Seitz

Image filtering

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



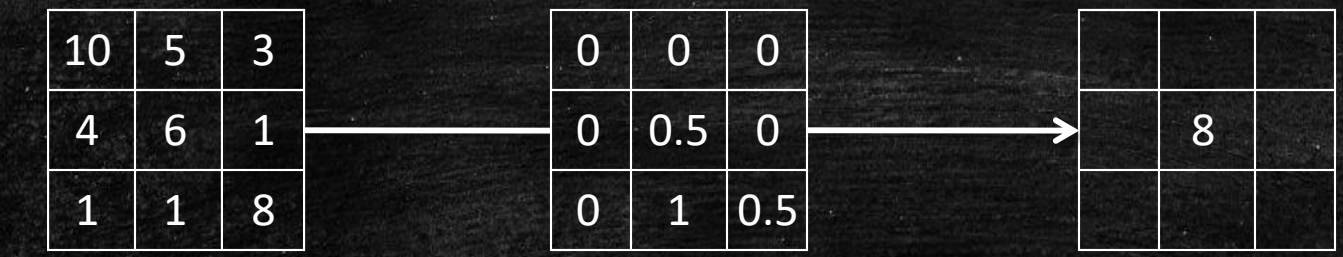
Local image data

Modified image data

Source: L. Zhang

Linear filtering

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



Source: L. Zhang

Convolution

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

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

- Convolution is **commutative** and **associative**

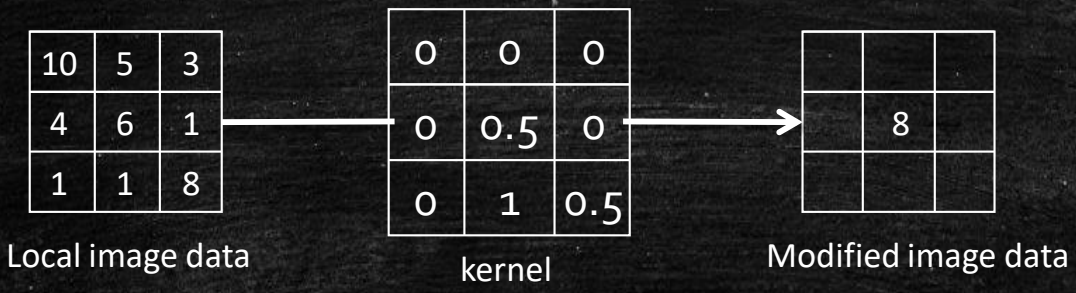
This is called a **convolution** operation:

$$G = H * F$$

Kernel Transformations

Replace each pixel by a linear combination of its neighbors

- The prescription for the linear combination is called the "kernel" (or "mask", "filter")
- Involves sliding a kernel (filter) across an image.
- A mask should always be in odd number, because other wise you cannot find the mid of the mask.



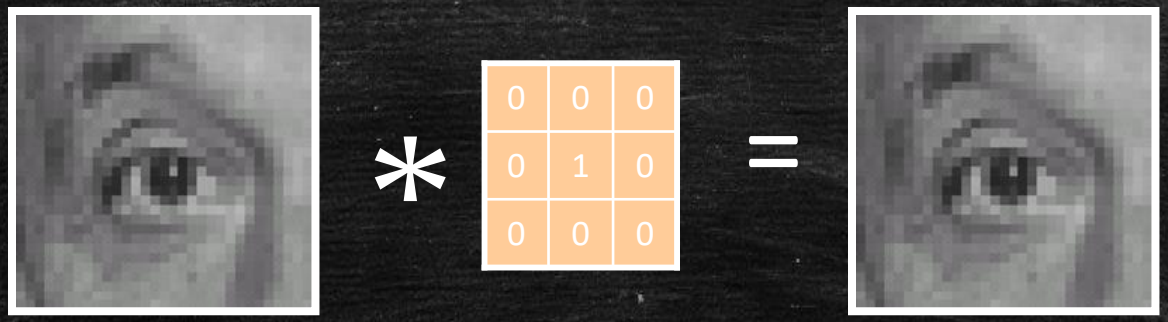
Source: L. Zhang

How to do it

In order to perform a filter on an image, following steps should be taken.

1) Slide the mask onto the image.
2) Multiply the corresponding elements and then add them
3) Repeat this procedure until all values of the image have been calculated.

Linear filters: examples



Original

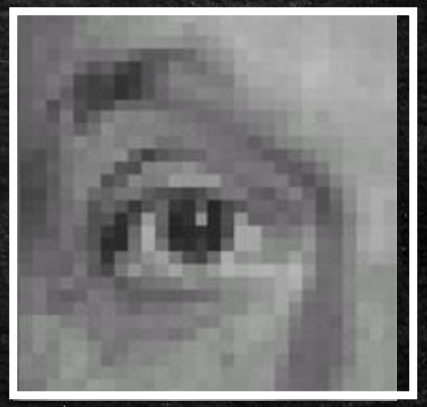
Identical image

Source: D. Lowe

Linear filters: examples

Original

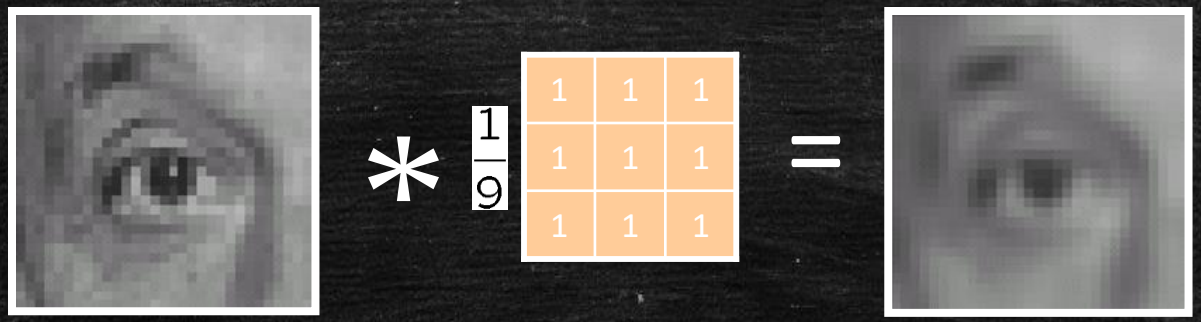
$$*\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$



Shifted left
By 1 pixel

Source: D. Lowe

Linear filters: examples



Original

Blur (with a mean filter)

Source: D. Lowe

2D Average filtering example using a 3 x 3 sampling window:

Keeping border values unchanged

Average = round(1+4+0+2+2+4+1+0+1)/9 = 2

Input

1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

Output

1	4	0	1	3	1
2	2	2	2	1	3
1	2	1	1	1	0
1	2	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

We just did blurring with a 3x3 mask.

Consider a 5x5 mask:

1) What would the effect on blurring be?
2) What would the values of the mask need to be?

It is also possible to have masks that are 7x7, 9x9, etc… Try it out!

Source: D. Lowe

Octave Implementation

```
rgbmonkey = imread("monkey.jpeg");

gs = mat2gray(rgb2gray(rgbmonkey)); %mat2gray for normalisation

kernel_identity = [0 0 0;0 1 0;0 0 0];

resultant = filter2 (kernel_identity,gs);
```

Try implementing different levels of blurring!

Octave built-in blur function

resultant = imsmooth(img, type,options)

Where type can be:

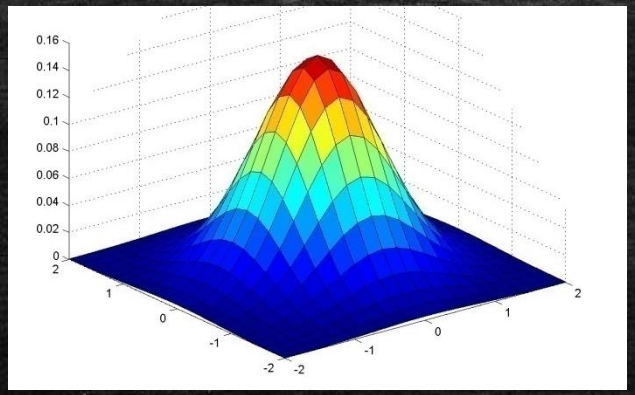
"**Gaussian**" - This is the default.

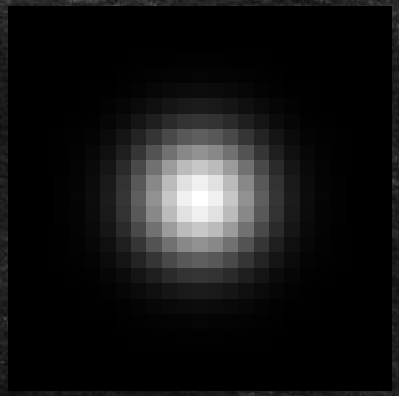
"**Average**" - Smoothing using a rectangular averaging (mean) filter.

"**Disk**" - Smoothing using a circular averaging (mean) filter.

....

Gaussian Kernel



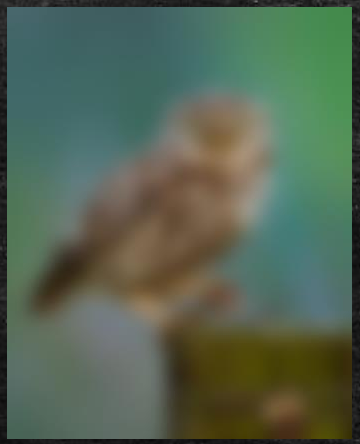
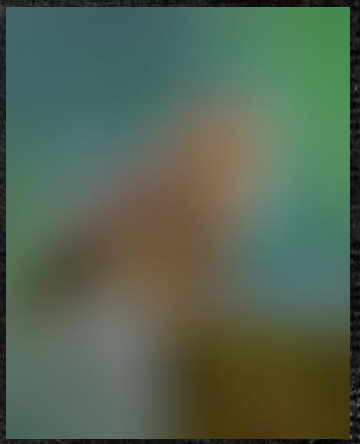


$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

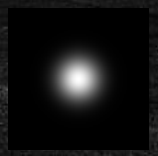
Source: C. Rasmussen

Gaussian filters



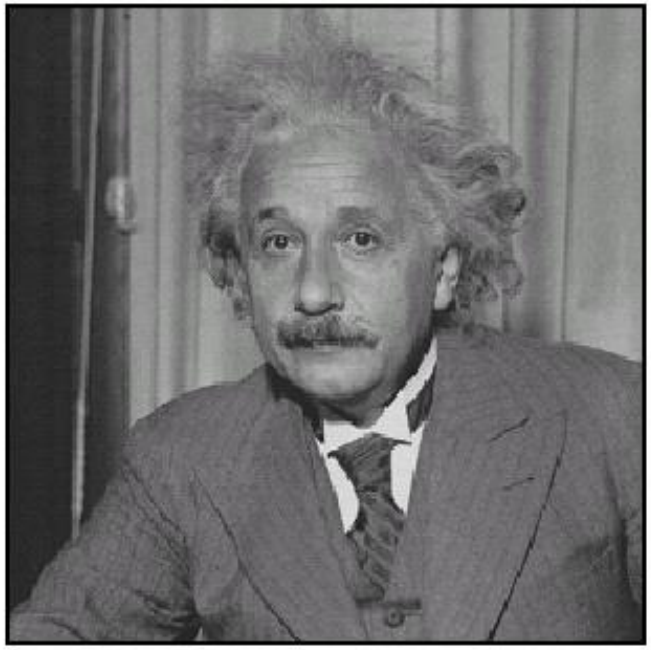






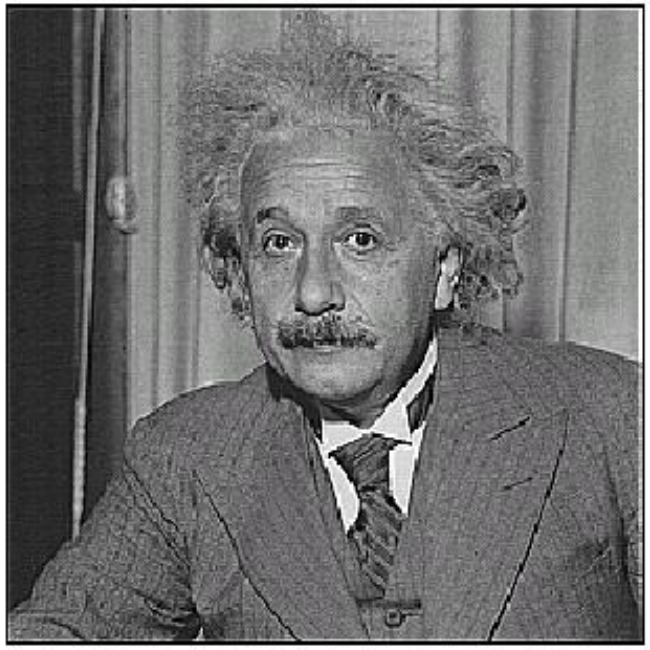



σ = 1 pixel σ = 5 pixels σ = 10 pixels σ = 30 pixels

High-pass filters: Sharpening



before



after

Source: D. Lowe

Sharpening

- What does blurring take away?

Let's add it back:

Source: S. Lazebnik

Linear filters: examples

Original

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

Sharpening filter
(accentuates edges)

Source: D. Lowe

Sharpening revisited

- What does blurring take away?

Let's add it back:

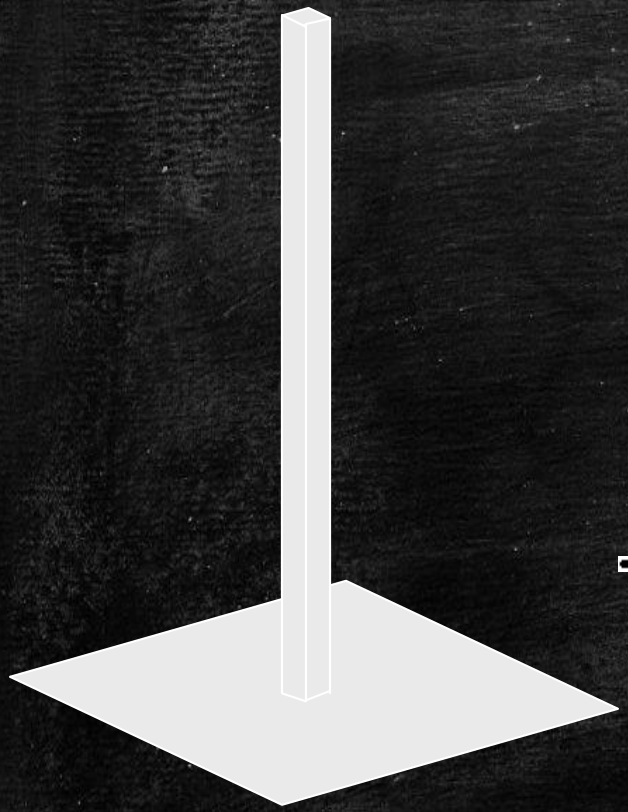
Source: S. Lazebnik

Sharpen filter

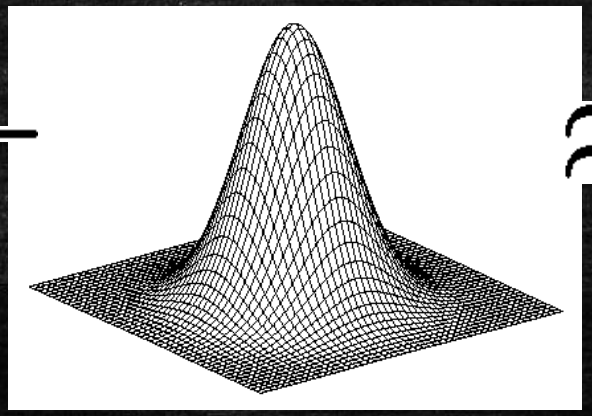
$$F + \alpha\,(F - F * H)$$

image

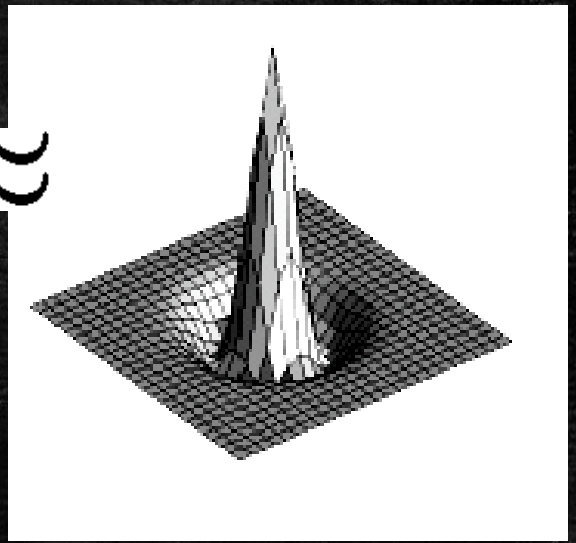
blurred image



scaled impulse



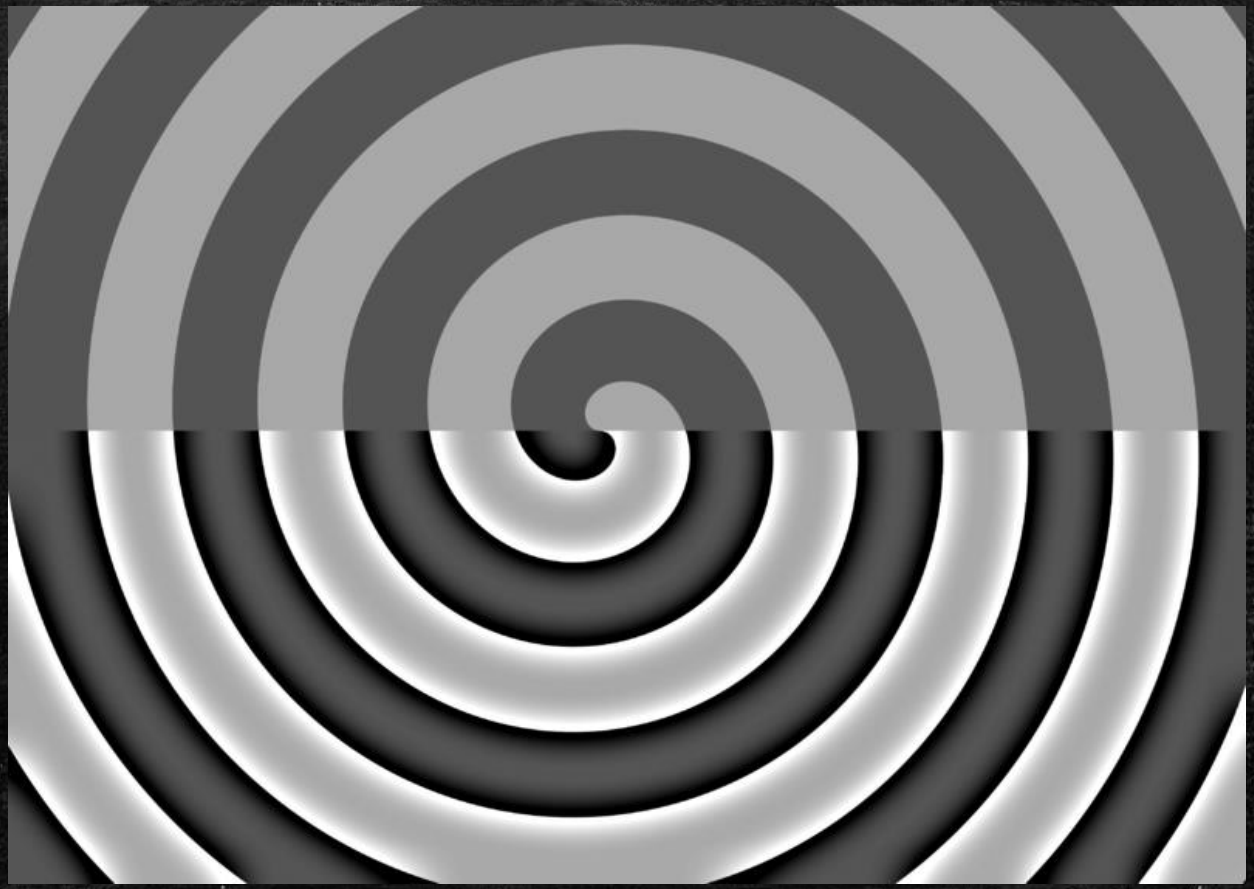
Gaussian



Sharpen filter

unfiltered

filtered



Convolution in the real world

Camera shake

Source: Fergus, *et al.* "Removing Camera Shake from a Single Photograph", SIGGRAPH 2006

Bokeh: Blur in out-of-focus regions of an image.

Source: http://lullaby.homepage.dk/diy-camera/bokeh.html