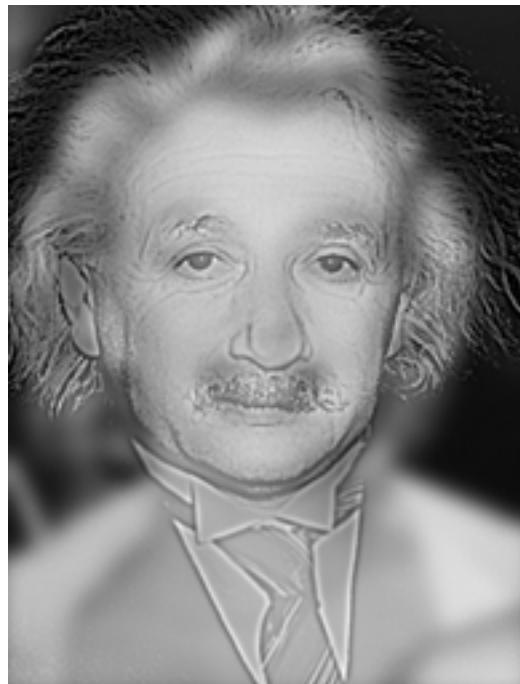


# CSC 589 Introduction to Computer Vision

## Lecture 2 linear filtering



Instructor: Bei Xiao

Wednesday , Jan 23, 2019

# Announcement

- First homework out: Image Processing with Numpy/Scipy. Due, Wed, January 30th.
- Course GITHUB (serve as our main portal):  
<https://github.com/fruittree/CSC476ComputerVision>
- Official office hour: Tuesday, Friday, 2:30pm-3:30pm  
DMTI 204

# Take-home reading

- Chapter 3. Szeliski. Page 89-104.
- Quiz next week for image convolution.

# Today's class

- What is an image?
- Point Process
- Neighborhood operation using scipy
- Convolution
- In-class Exercises

# Choices of Python Image libraries

- Level 1 (basic): Numpy, treating image as matrix
- Level 2 (Scipy): an image I/O (`scipy.misc.imread`),  
[scipy.ndimage](#) package that has convolution, filters,  
etc.
- Level 3 ([sckit-image](#)): equivalent to MATLAB image  
processing toolbox, but better. Many built-in stuff, so  
not suitable for conceptual learning in the beginning.
- High-level (OpenCV): it is not suitable for teaching but  
suitable for development. Very different from actual  
Python. We might use it for some projects later in the  
course.

# Choices of Python libraries

- Level 1 (basic): Numpy, treating image as matrix
- Level 2 (Scipy): an image I/O (`scipy.misc.imread`),  
[scipy.ndimage](#) package
- Level 3 ([sckit-image](#)): equivalent to MATLAB image processing toolbox, but better. We will use it when we need to.
- High-level (OpenCV): it is not suitable for teaching but suitable for development. We might use it for some projects later in the course.

In the first 3 weeks, we use Level 1 and 2.

The higher level library you use, the less control you have!

# To start, we will use the basic level libraries!

- Level 1 (basic): Numpy, basic numerical Python, treating image as matrix
- Level 2 (Scipy): an image I/O (`scipy.misc.imread`), `scipy.ndimage` package
- We will write our own functions!

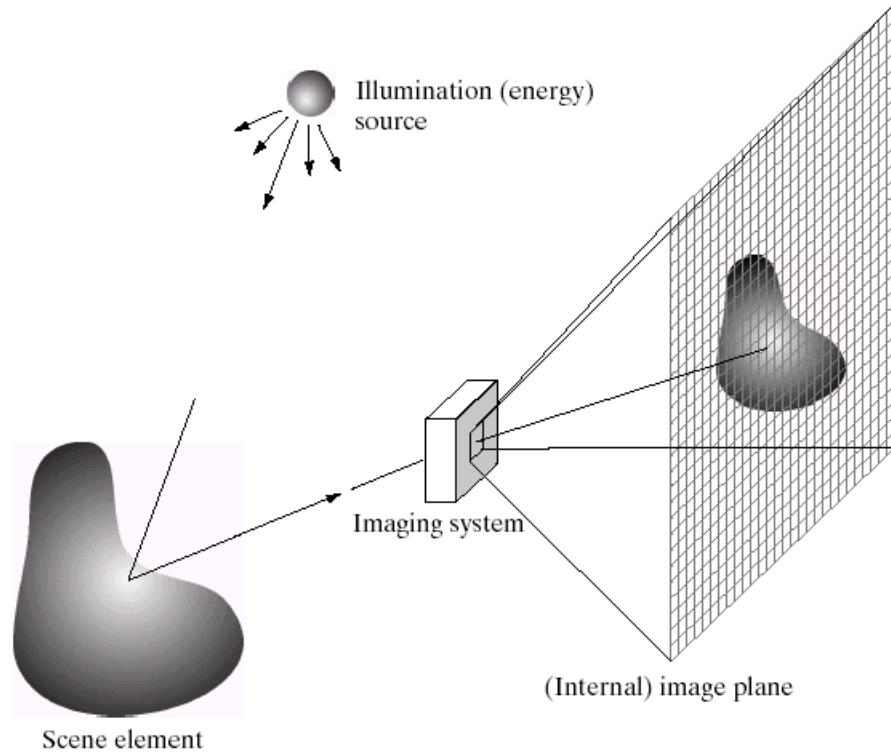
# What is an image?



# A useful tutorial

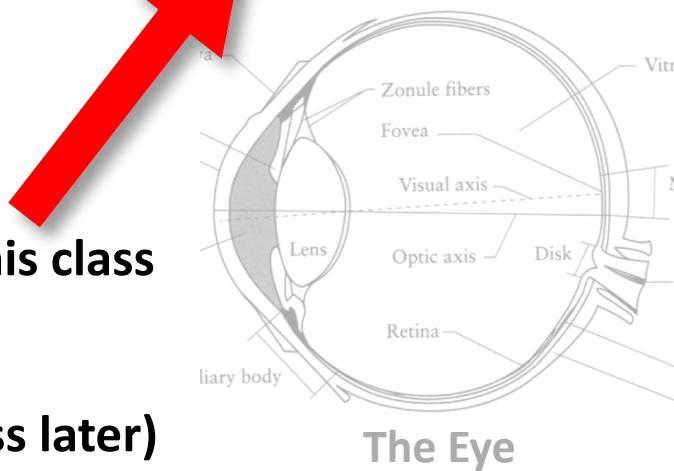
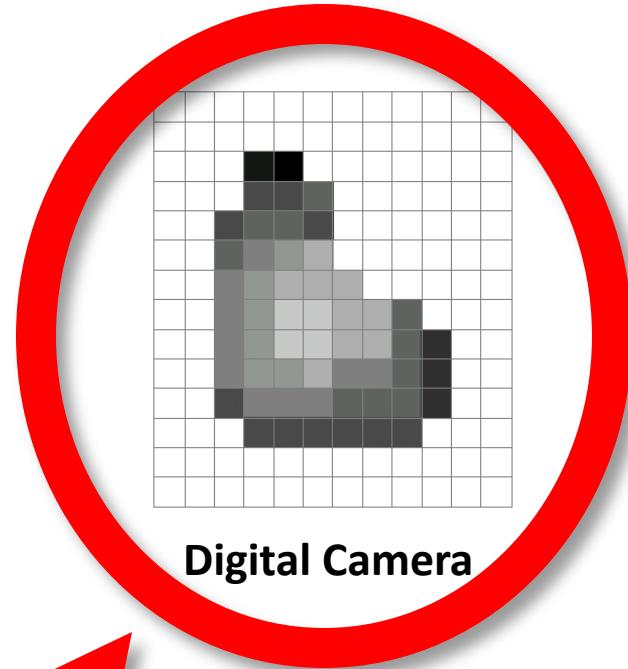
Computer vision for dummies

# What is an image?



We'll focus on these in this class

(More on this process later)

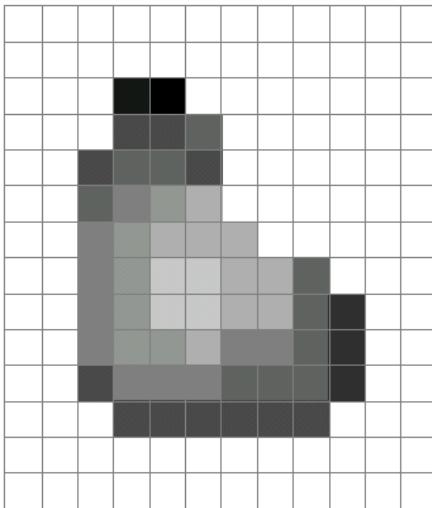


The Eye

Source: A. Efros

# What is an image?

- A grid (matrix) of intensity values

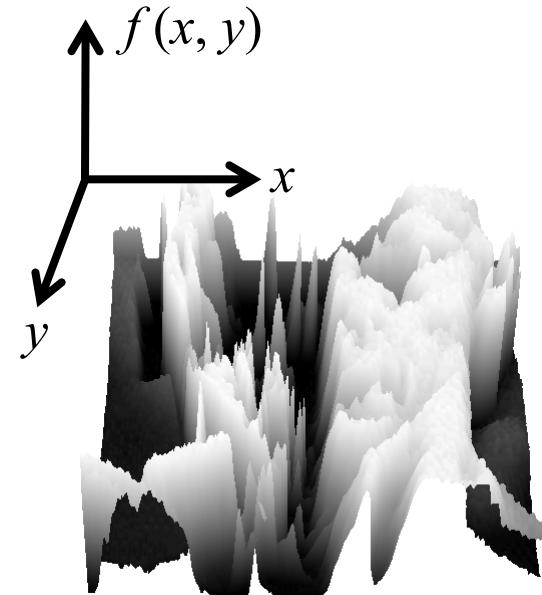


255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	20	0	255	255	255	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255	255	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255	255	255	255	255
255	255	127	145	145	175	127	127	95	47	255	255	255	255	255	255
255	255	74	127	127	127	127	95	95	95	47	255	255	255	255	255
255	255	255	74	255	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	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  $\mathbb{R}$ :
  - $f(x,y)$  gives the **intensity** at position  $(x,y)$



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

Slide credit: James Hays

# Image Processing

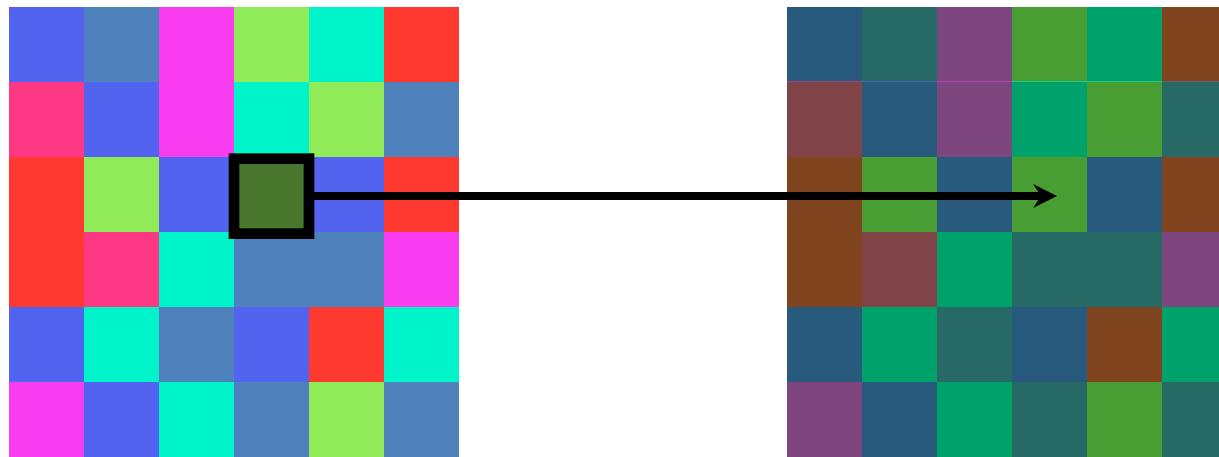
---

- Define a new image  $g$  in terms of an existing image  $f$ 
  - We can transform either the domain or the range of  $f$
- Range transformation:

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

What kinds of operations can this perform?

# Point Operations



# Point Processing

Original



Darken



Lower Contrast



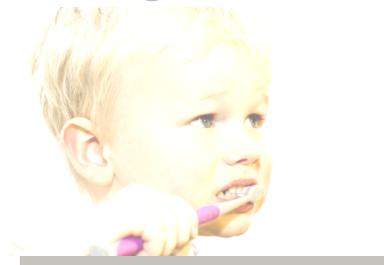
Nonlinear Lower Contrast



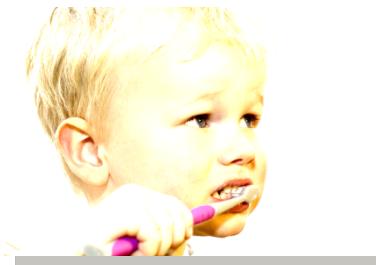
Invert



Lighten



Raise Contrast



Nonlinear Raise Contrast



# Point Processing

Original



x

Darken



x - 128

Lower Contrast



x / 2

Nonlinear Lower Contrast



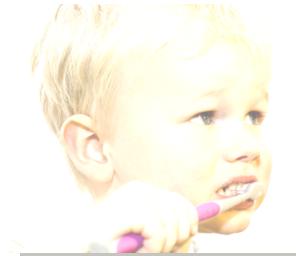
((x / 255.0) ^ 0.33) \* 255.0

Invert



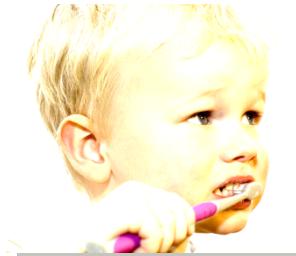
255 - x

Lighten



x + 128

Raise Contrast



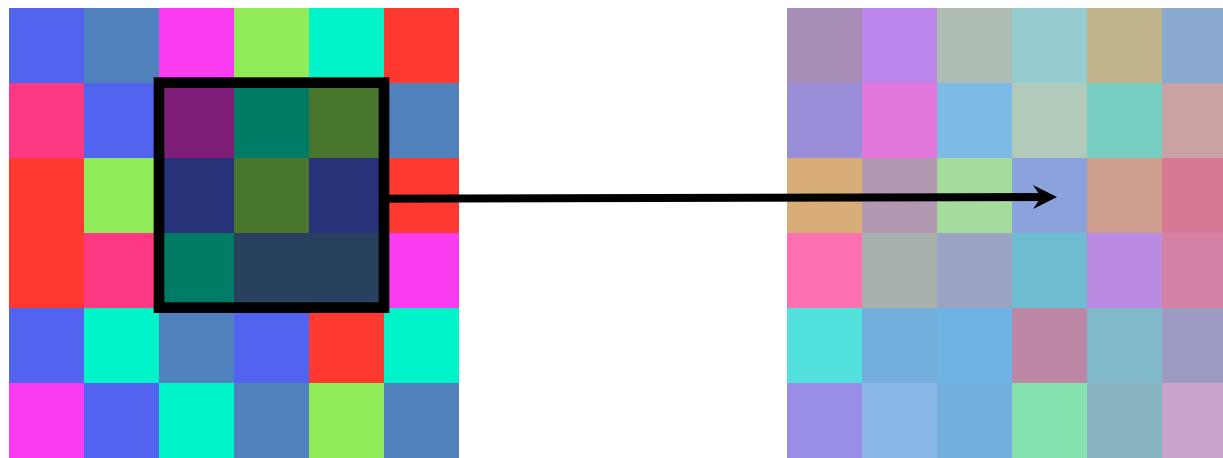
x \* 2

Nonlinear Raise Contrast



((x / 255.0) ^ 2) \* 255.0

# Neighborhood Operations



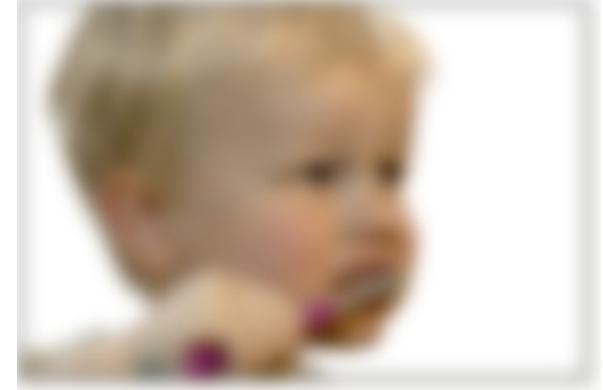
# Neighborhood operations



Image



Edge detection



Blur

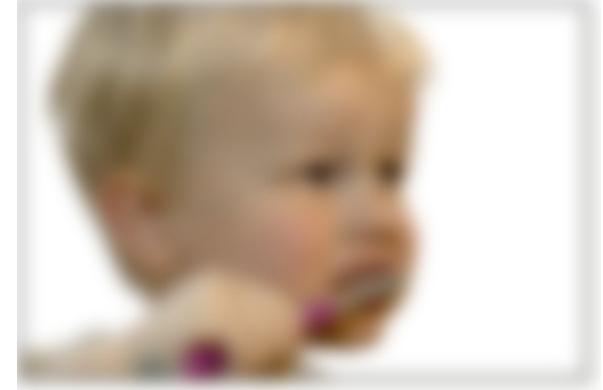
# Neighborhood operations



Image



Edge detection



Blur

# 3×3 Neighborhood



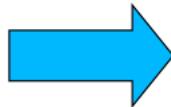
# 5×5 Neighborhood



# 7x7 Neighborhood



# Let's represent this more generally



0	3	0	0
0	6	1	16
0	0	2	46
0	0	2	43

Slide source: Matthew Tappen

# Let's represent this more generally

Normalized box filter (3×3)

0	3	0	0
0	6	1	16
0	0	2	46
0	0	2	43

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Slide source: Matthew Tappen

# Let's represent this more generally

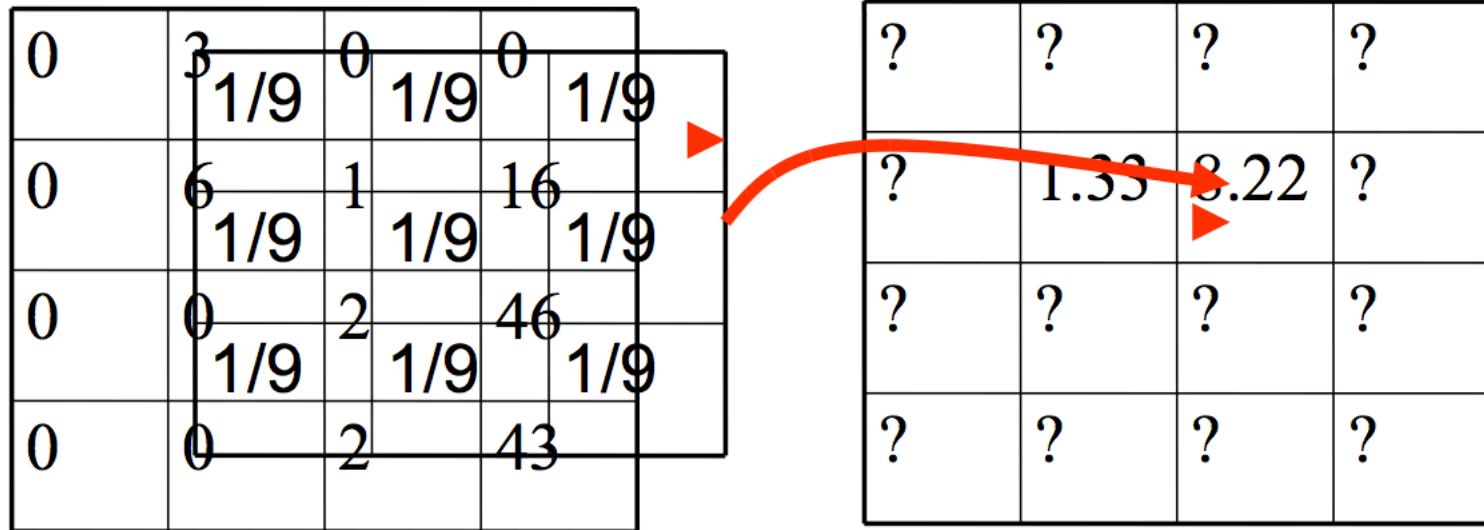
0	3	0	1/9	0
1/9	1/9	1/9	1/9	1/9
0	6	1	16	1/9
1/9	1/9	1/9	1/9	1/9
0	0	2	46	1/9
1/9	1/9	1/9	1/9	1/9
0	0	2	43	

?	?	?	?
?	1.33	?	?
?	?	?	?
?	?	?	?

- Multiply corresponding numbers and add

Slide source: Matthew Tappen

# Let's represent this more generally



- Multiply corresponding numbers and add
- Template moves across the image
- Think of it as a sliding window

Slide source: Matthew Tappen

# Image Filtering

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

10	5	3
4	5	1
1	1	7

Some function

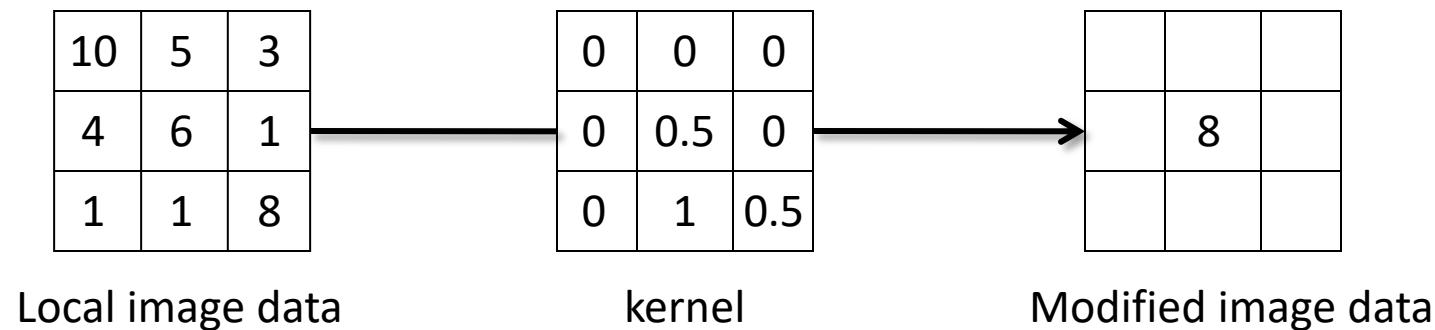


		7

Local image data

# Linear filtering

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



# This is called convolution

- Mathematically expressed as

$$R(i, j) = \sum_{m=-N}^N \sum_{n=-N}^N I(m, n)K(i - m, j - n)$$

Resulting Image      Input Image      Convolution Kernel

<https://en.wikipedia.org/wiki/Convolution>

<https://dsp.stackexchange.com/questions/2654/what-is-the-difference-between-convolution-and-cross-correlation>

Slide source: Matthew Tappen

# Notation

- Also denoted as
- $R = I * K$
- We “convolve”  $I$  with  $K$ 
  - Not convolute!

$$R(i, j) = \sum_{m=-N}^N \sum_{n=-N}^N I(m, n)K(i - m, j - n)$$

The diagram shows the convolution formula  $R(i, j) = \sum_{m=-N}^N \sum_{n=-N}^N I(m, n)K(i - m, j - n)$ . Three blue arrows point to the variables: one points to  $R(i, j)$  with the label "Resulting Image", another points to  $I(m, n)$  with the label "Input Image", and a third points to  $K(i - m, j - n)$  with the label "Convolution Kernel".

# Filtering vs. Convolution

- 2d filtering
  - `h=filter2(g,f);` or  
`h=imfilter(f,g);`

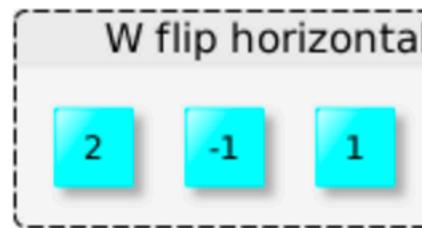
$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

- 2d convolution (must flip kernel)
  - `h=conv2(g,f);`

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

We use Filter and Convolution interchangeable when the image is SYMMETRIC.

# Short tutorial on convolution



# Doing by hand

1)

0

1

2

3

4

$$(1 \times 0) = 0$$



2)

0

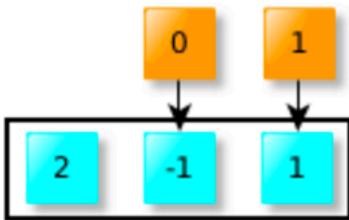
1

2

3

4

$$(1 \times 1) + (-1 \times 0) = 1$$



3)

0

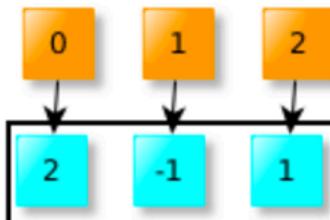
1

2

3

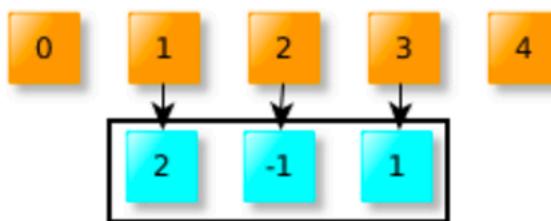
4

$$(1 \times 2) + (-1 \times 1) + (2 \times 0) = 1$$



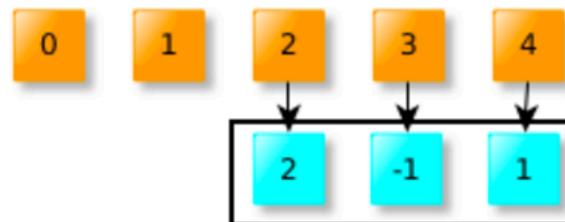
# Doing by hand

4)



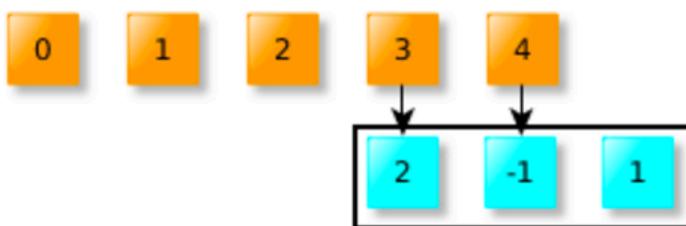
$$(1 \times 3) + (-1 \times 2) + (2 \times 1) = 3$$

5)



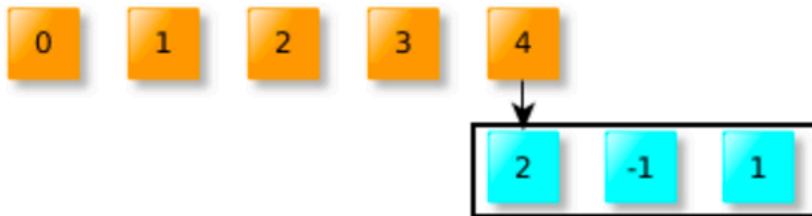
$$(1 \times 4) + (-1 \times 3) + (2 \times 2) = 5$$

6)

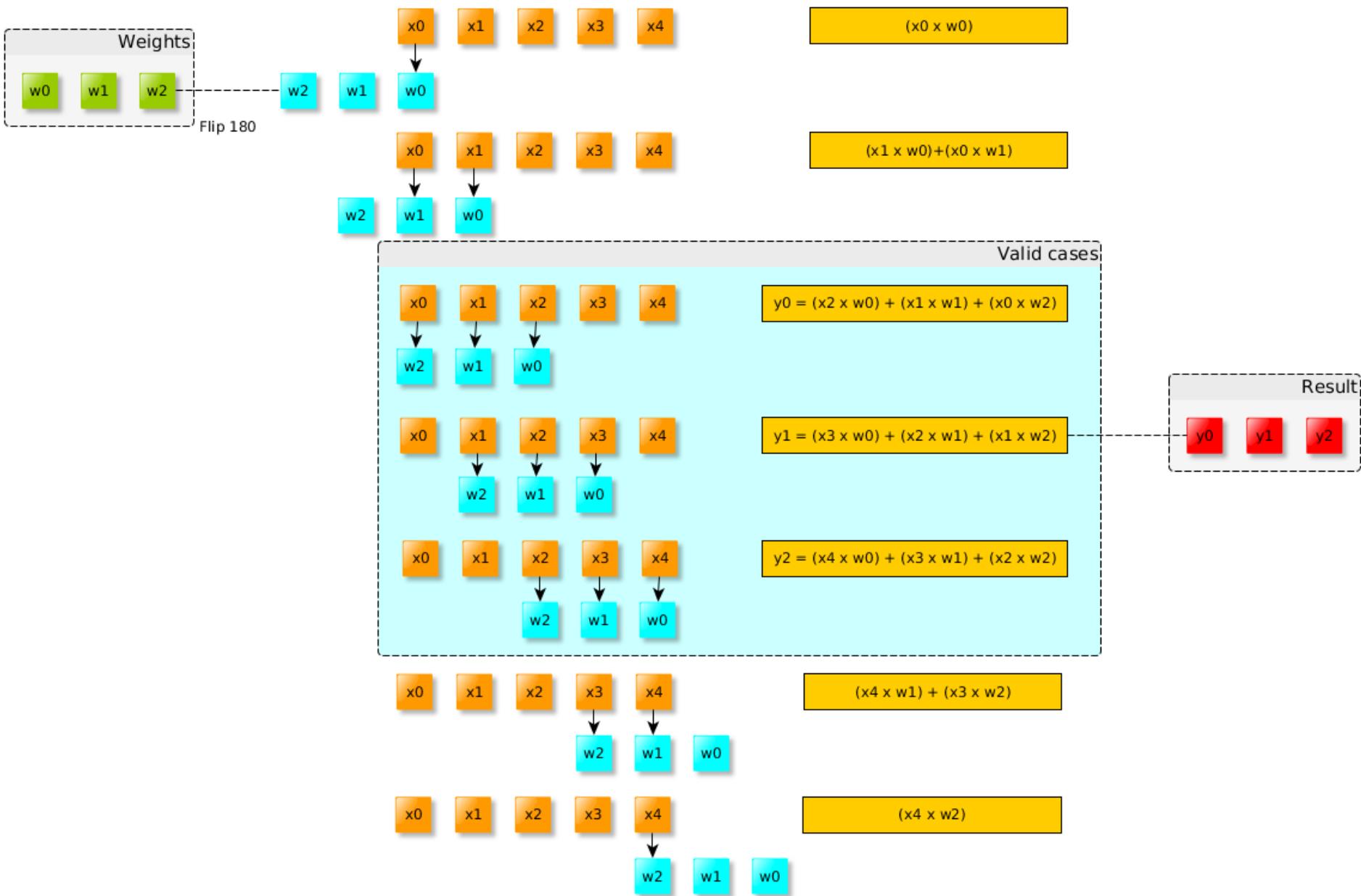


$$(2 \times 3) + (-1 \times 4) = 2$$

7)



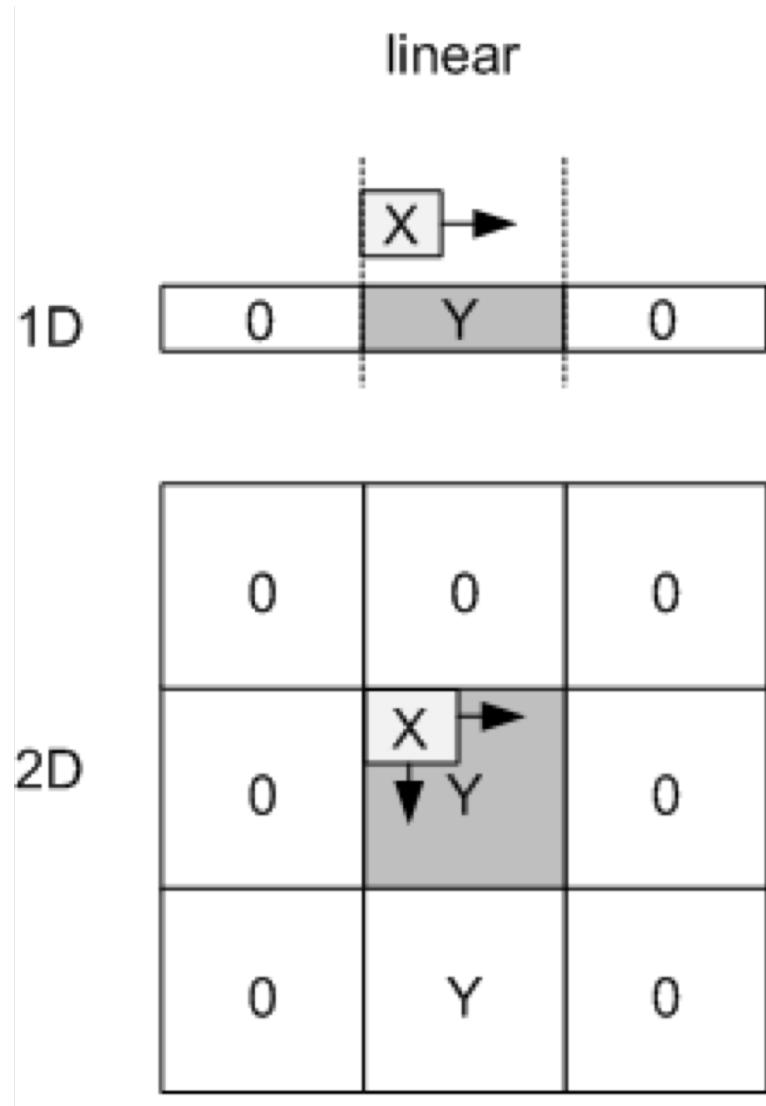
$$(2 \times 4) = 8$$



# in Python

```
import numpy as np  
x = np.array([0,1,2,3,4])  
w = np.array([1,-1,2])  
  
res = np.convolve(x,w, 'valid')  
print res
```

# Input Padding



# 2D convolution



\*

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



# Let's take out paper and pen

- What is the result of the following convolution?

1	2	3
4	5	6
7	8	9

Input

$m$	-1	0	1
-1	-1	-2	-1
0	0	0	0
1	1	2	1

Kernel

# Let's take out a paper and a pen

- What is the result of the following convolution?

1	2	1
0	0	0
-1	-2	-1
7	8	9

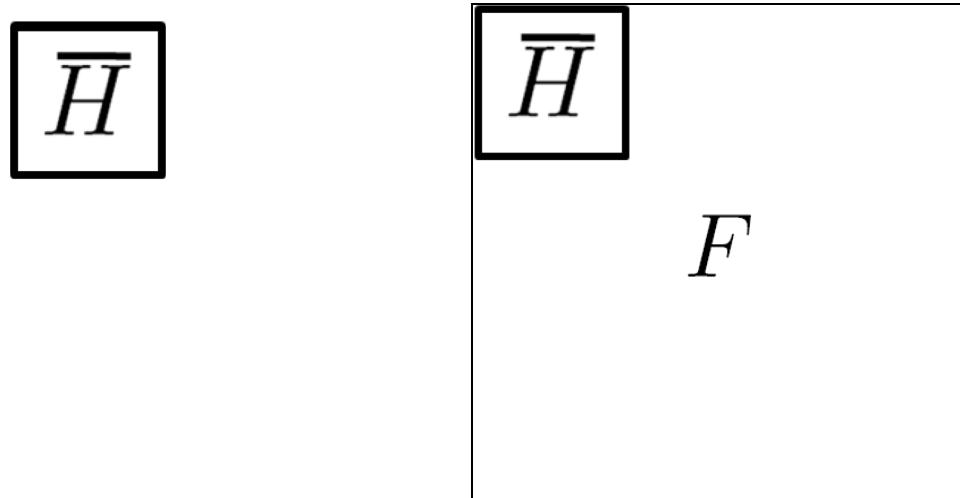
$$\begin{aligned}1*0 + 2*0 + 1*0 + \\0*0 + 1*0 + 2*0 + \\(-1)*0 + (-2)*4 + (-1)*5 = -13\end{aligned}$$

# Let's take out a paper and a pen

- What is the result of the following convolution?

1	2	1
0	0	0
1	2	3
-1	-2	-1
4	5	6
7	8	9

# Convolution



# Mean filtering




*H*

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

*F*

=

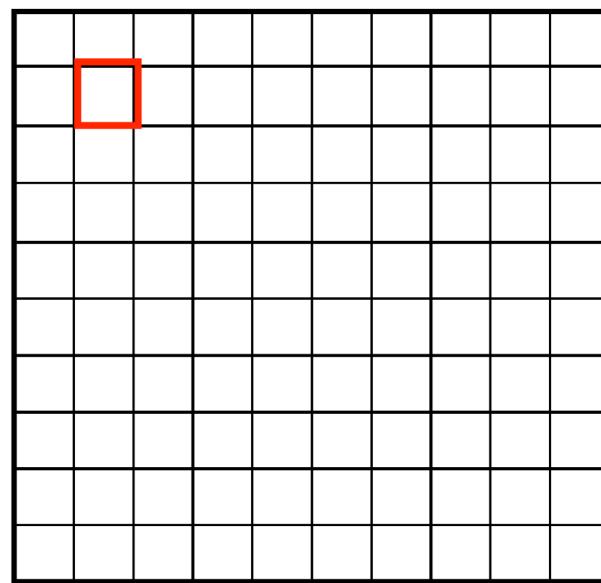
	0	10	20	30	30	30	20	10		
0	20	40	60	60	60	60	40	20		
0	30	60	90	90	90	90	60	30		
0	30	50	80	80	90	60	30			
0	30	50	80	80	90	60	30			
0	20	30	50	50	60	40	20			
10	20	30	30	30	30	20	10			
10	10	10	0	0	0	0	0	0		

*G*

# Mean filtering/Moving average

$$F[x, y]$$

$$G[x, y]$$



# Mean filtering/Moving average

$$F[x, y]$$

$$G[x, y]$$

# Let's take out a paper and a pen

A diagram illustrating a convolution operation. On the left is a 3x3 **input** matrix:

1	3	1
0	-1	1
2	2	-1

In the center is a **\*** symbol followed by a 2x2 **kernel**:

1	2
0	-1

To the right is a **flip(kernel)** matrix:

-1	0
2	1

A curved arrow points from the **kernel** to the **flip(kernel)** matrix.

# Let's take out a paper and a pen

Multiply the window element by element with flip(kernel), then sum the results

1	3	1
0	-1	1
2	2	-1

$$\Rightarrow 1(-1) + 3(0) + 0(2) - 1(1) = -2$$

1	3	1
0	-1	1
2	2	-1

$$\Rightarrow 3(-1) + 1(0) - 1(2) + 1(1) = -4$$

1	3	1
0	-1	1
2	2	-1

$$\Rightarrow 0(-1) - 1(0) + 2(2) + 2(1) = 6$$

result(valid)

-2	-4
6	4

1	3	1
0	-1	1
2	2	-1

$$\Rightarrow -1(-1) + 1(0) + 2(2) - 1(1) = 4$$

# Check it with Python?

```
import numpy as np  
from scipy import signal
```

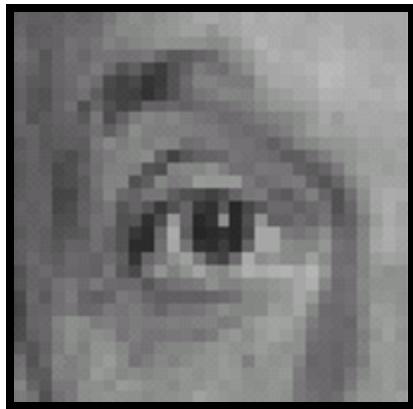
```
# use np.array
```

```
# use signal.convolve2d
```

# Rules of a image filter

- It's size has to be uneven, so that it has a center, for example,  $3\times 3$ ,  $5\times 5$ ,  $7\times 7$
- It doesn't have to, but the sum of all elements of the filter should be 1 if you want the result image to have the **same brightness** as the original
- If the sum of the element of is larger than 1, the result will be a brighter image, if it is smaller than 1, the resulting image will be darker.

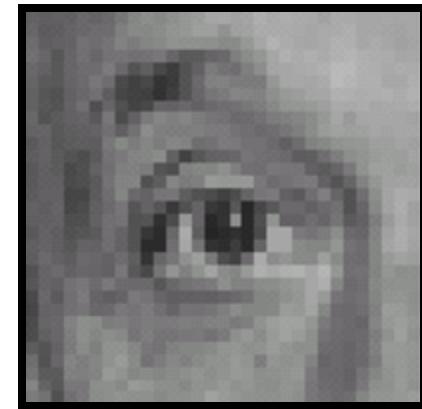
# Practice with linear filters



\*

0	0	0
0	1	0
0	0	0

=



Original

Filtered  
(no change)

# Linear filters: examples



\*

0	0	0
1	0	0
0	0	0

=



Original

Shifted left  
By 1 pixel

# Linear filters: examples

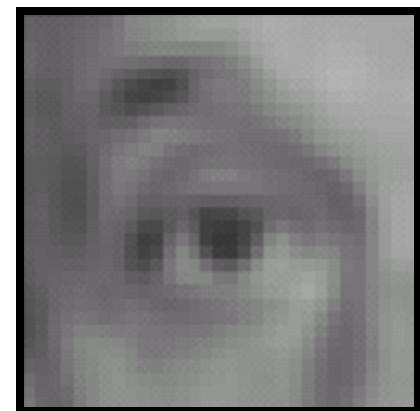


\*

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

=



Original

Blur (with a mean filter)

# Linear filters: examples

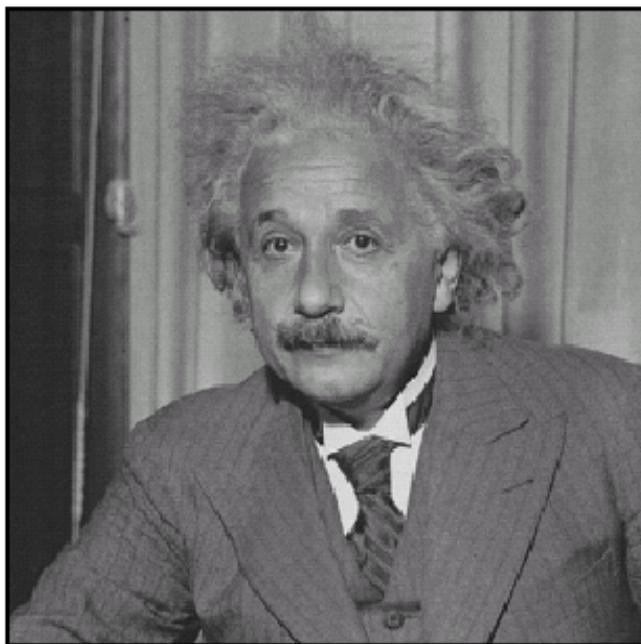


$$\text{Original} \quad * \left( \begin{array}{ccc} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{array} \right) - \frac{1}{9} \left( \begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} \right) = \text{Sharpened Image}$$

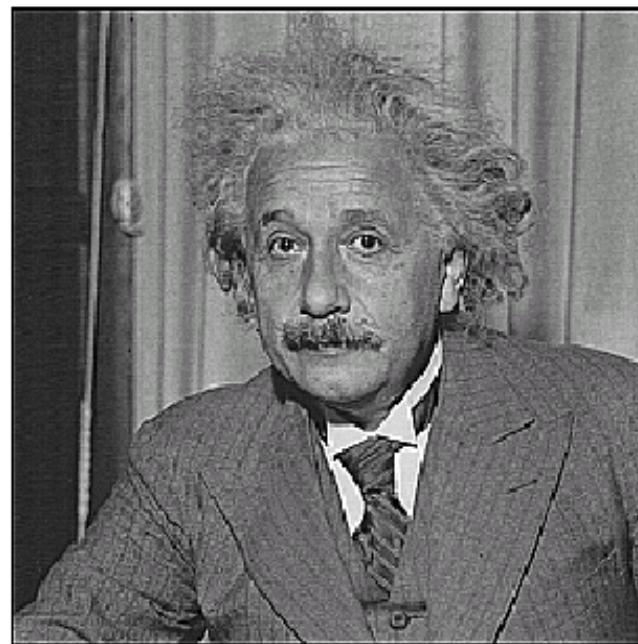


**Sharpening filter**  
(accentuates edges)

# Sharpening

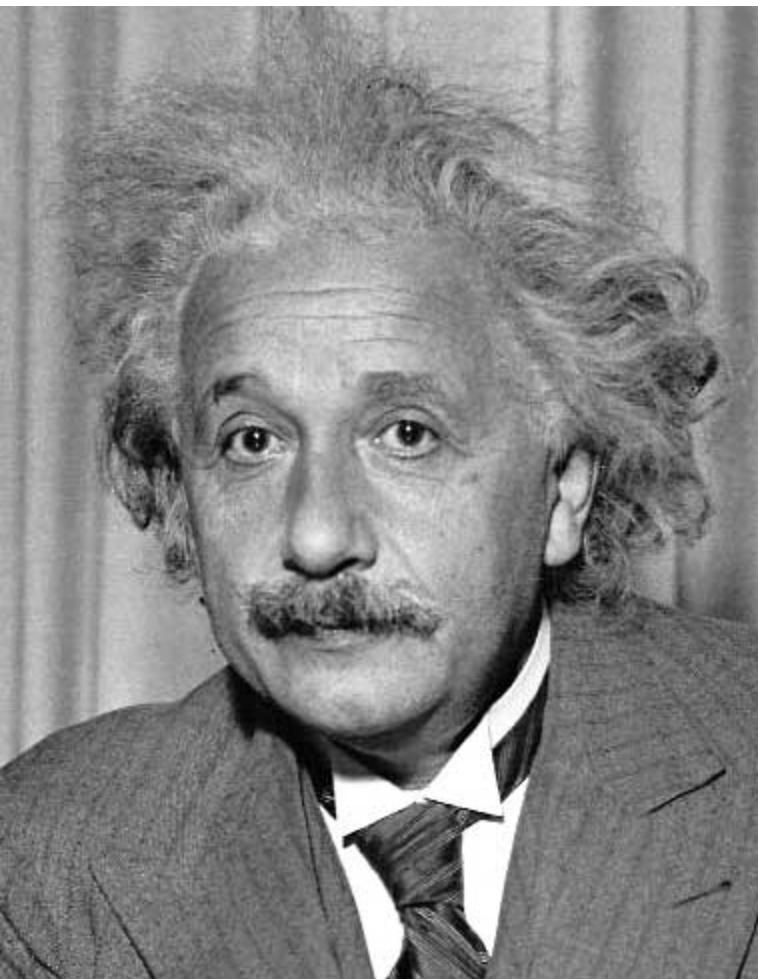


**before**



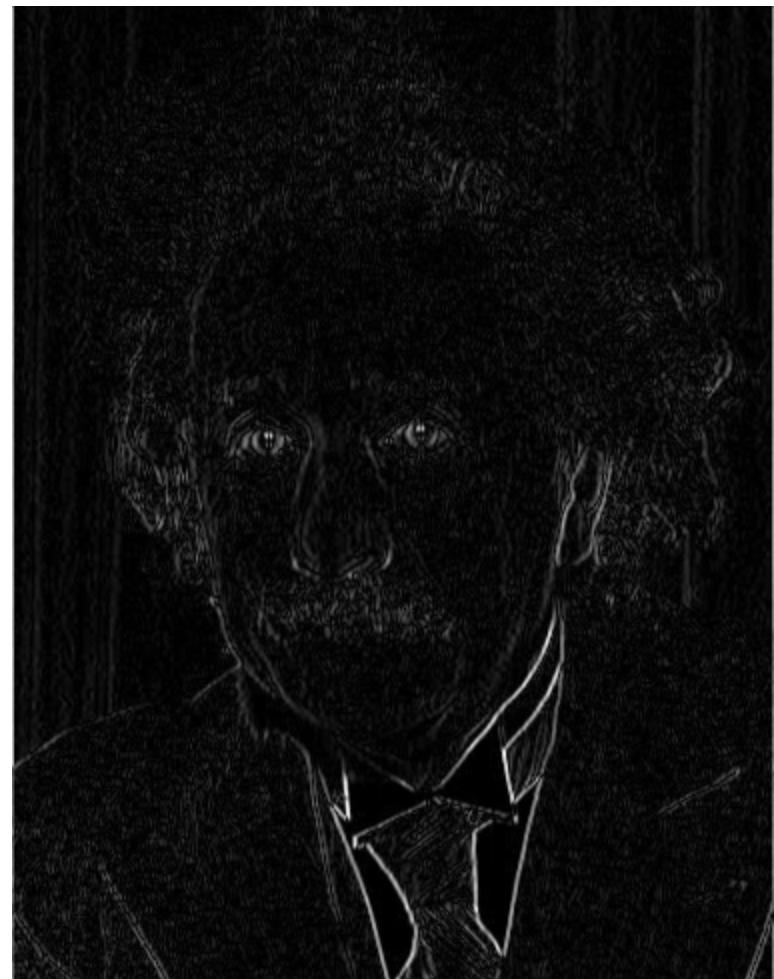
**after**

# Other filters



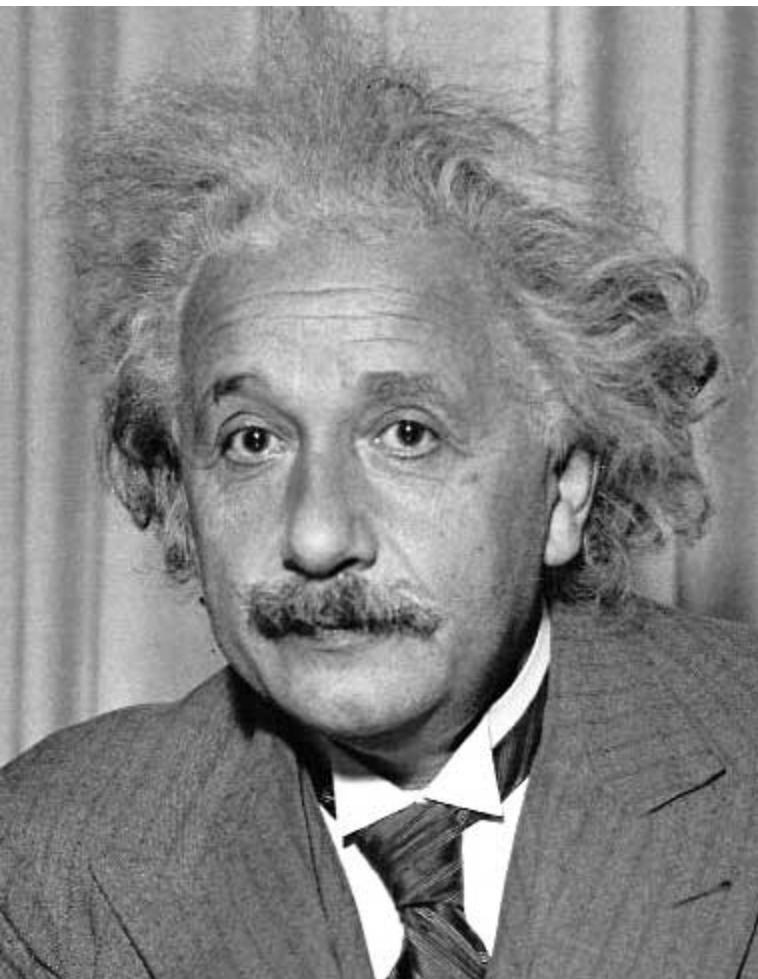
1	0	-1
2	0	-2
1	0	-1

Sobel



Vertical Edge  
(absolute value)

# Other filters



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

Sobel



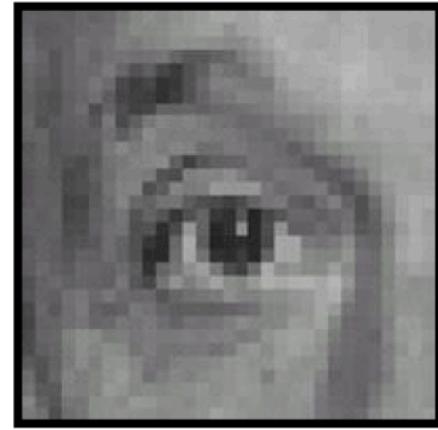
Horizontal Edge  
(absolute value)

# Practice with linear filters



Original

0	0	0
0	1	0
0	0	0



Filtered  
(no change)

# Practice with linear filters



Original

0	0	0
0	0	1
0	0	0

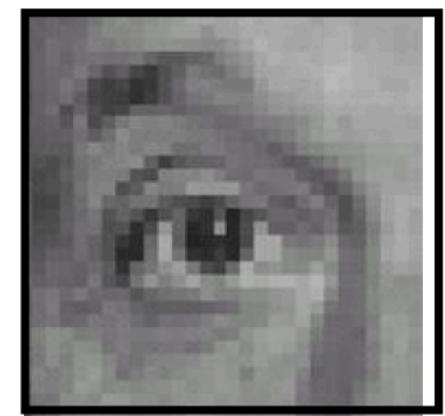
?

# Practice with linear filters



Original

0	0	0
0	0	1
0	0	0



Shifted left  
By 1 pixel

# Practice with linear filters



$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} ?$$

(Note that filter sums to 1)

Original

# Practice with linear filters



Original

0	0	0
0	2	0
0	0	0

-

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1



## Sharpening filter

- Accentuates differences with local average

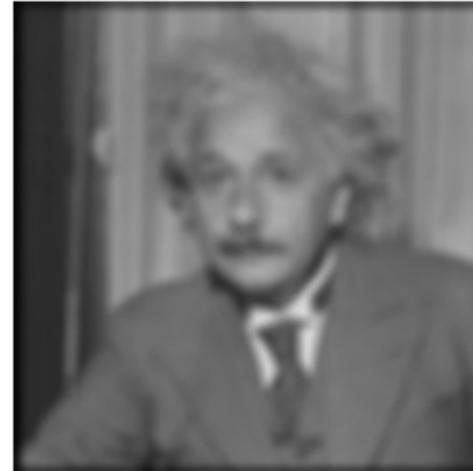
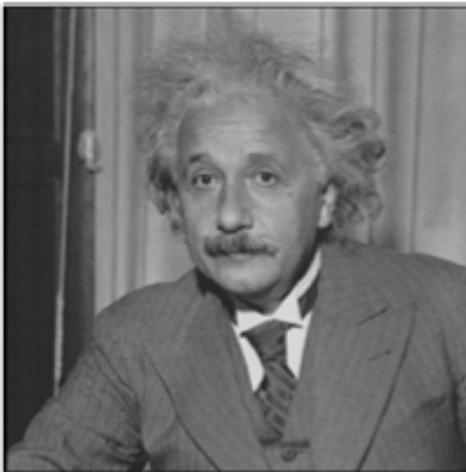
# Exercise

- Find out why the sharper filter works by manually convolve a 5x5 matrix with the kernel.

Using Python to test all the linear filter operations on images.

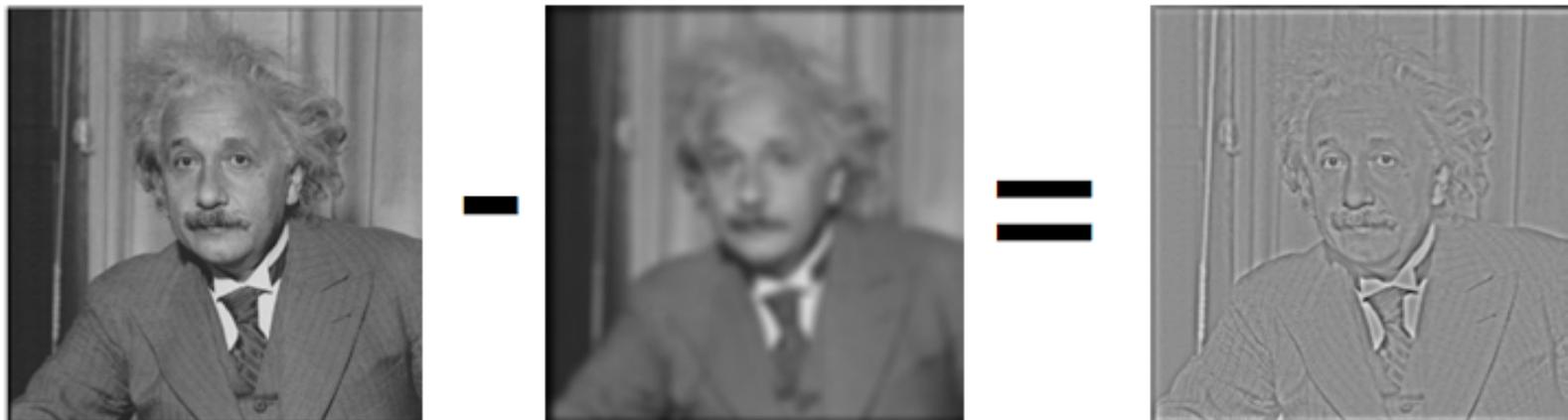
# Image Sharpening

- Take this image and blur it



# Image Sharpening

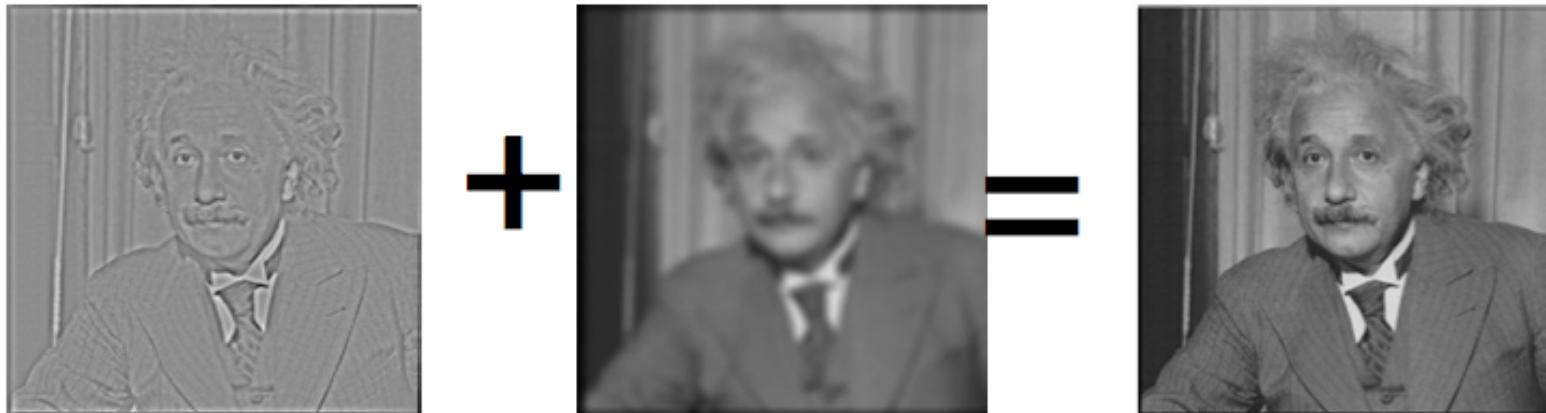
- What do we get if we subtract this two?



This is the left-over sharp stuff!

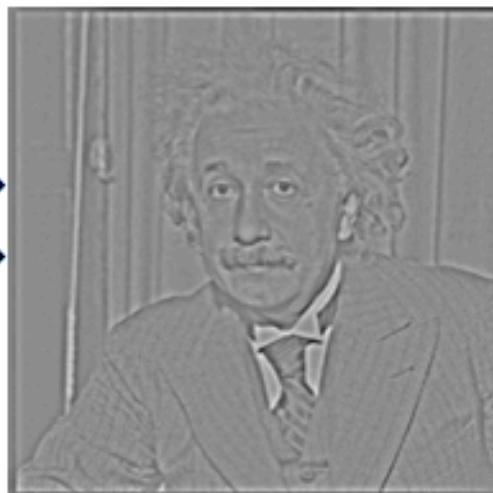
# Let's make the image sharper?

- We know: “sharp stuff + blurred = original”

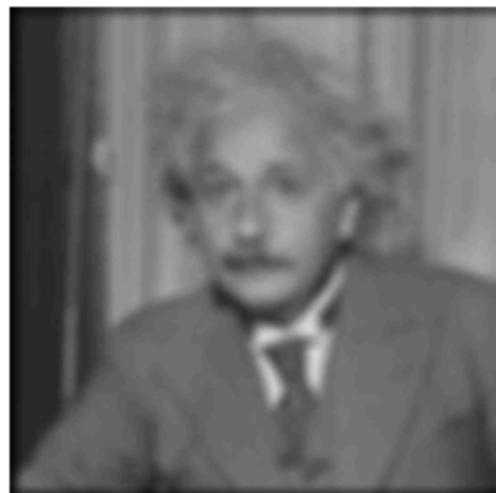


# Let's boost the sharp stuff a little

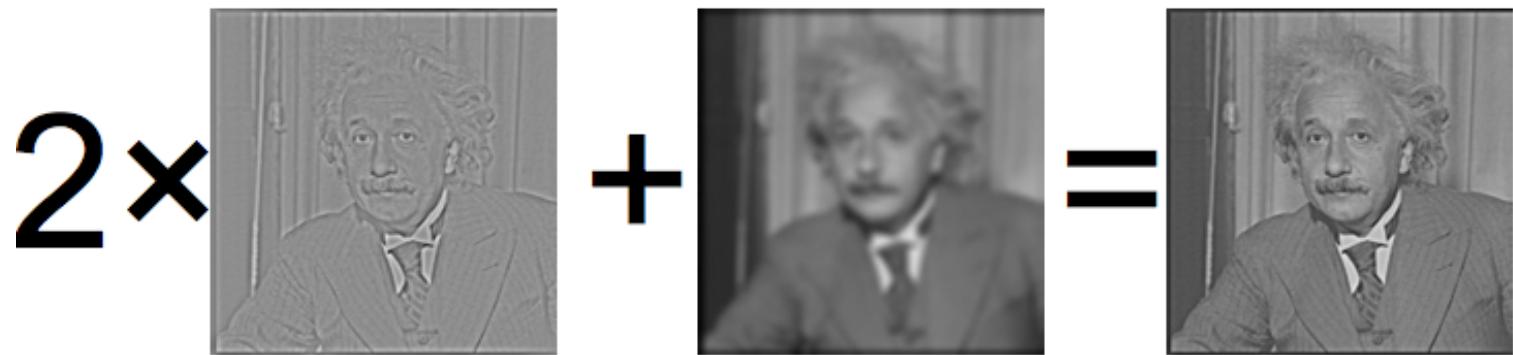
**2×**



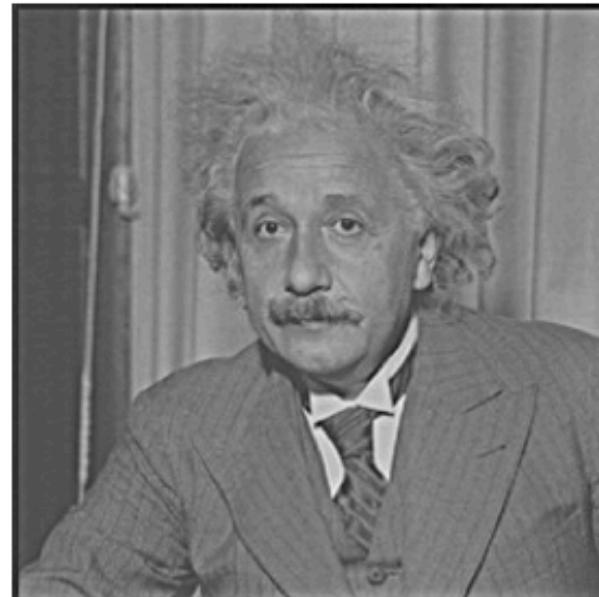
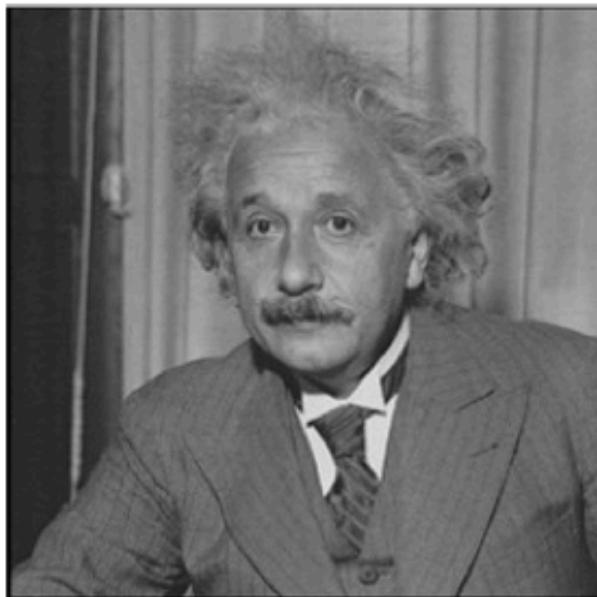
**+**



# Let's boost the sharp stuff a little



# Side by Side



# With Python

```
baby = misc.imread('Kaila.jpg',flatten=1)

## blur
blurred = ndimage.gaussian_filter(baby, 3)
## this is actually the original
filter_blurred= ndimage.gaussian_filter(blurred,1)
alpha = 30
## sharpen
sharpened = blurred + alpha * (blurred - filter_blurred)
```

# Now look at the computation

- Operations
  - 1 convolution
  - 1 subtraction over the whole image
- As an equation:

$$\mathcal{I} * f + 2(\mathcal{I} - \mathcal{I} * f)$$

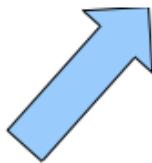
Blur: Convolve with mean filter

Sharpen: original - Convolve with mean filter

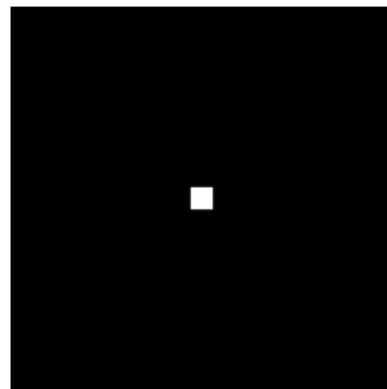
# Rewrite this

$$\mathcal{I} * f + 2(\mathcal{I} - \mathcal{I} * f)$$

$$\mathcal{I} * f + 2(\mathcal{I} * \delta - \mathcal{I} * f)$$



This is an identity filter or unit impulse



Rewrite this

$$\mathcal{I} * f + 2(\mathcal{I} - \mathcal{I} * f)$$

$$\mathcal{I} * f + 2(\mathcal{I} * \delta - \mathcal{I} * f)$$

$$\mathcal{I} * (f + 2\delta - 2f)$$

# Now look at the computation

$$\mathcal{I} * (f + 2\delta - 2f)$$

- Can pre-compute new filter
- Operations

1 convolution

# Photoshop: Unsharp Masking

- [http://en.wikipedia.org/wiki/Unsharp\\_masking](http://en.wikipedia.org/wiki/Unsharp_masking)



Source image (left) and the sharpened image (right).

# Unsharp Masking (Scikit)

```
from skimage import filter
from skimage import img_as_float
import matplotlib.pyplot as plt

unsharp_strength = 0.8
blur_size = 8 # Standard deviation in pixels.

# Convert to float so that negatives don't cause problems
image = img_as_float(data.camera())
blurred = filter.gaussian_filter(image, blur_size)
highpass = image - unsharp_strength * blurred
sharp = image + highpass

fig, axes = plt.subplots(ncols=2)
axes[0].imshow(image, vmin=0, vmax=1)
axes[1].imshow(sharp, vmin=0, vmax=1)
```

# Your Convolution filter toolbox

- 90% of the filtering that you will do will be either
- Smoothing (or Blurring)
- High-Pass Filtering (will explain later)
- Most common filters:
- Smoothing: Gaussian
- High Pass Filtering: Derivative of Gaussian

# Python image convolution demo

[scipy.ndimage.filters.convolve](#), specifically  
for images

[scipy.signal.convolve2d](#), general 2D convolution

# Lecture 2 Lab

- Please download Lecture2\_inclass exercise and finish the problem sets.
- The problem set is due next class. No submission is required but the instructor will check your solutions and code.

# Next class (Next Wed)

- More on image processing, contrast enhancement, image histograms, Gaussian Filter.
- Take-home reading: Szeliski, Chapter 3.1-3.2