CSC 589 Introduction to Computer Vision

Lecture 2 linear filtering

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

Instructor: Bei Xiao

Thursday, January 15th

Take-home work (very important)

Reading

Read Szelisky : Chapter 3.1, 3.2. You can skip color if you want.

A tutorial on Image convolution: http://lodev.org/cgtutor/filtering.html

Software

Install Numpy, Scipy
The easiest thing to do is to install Spyder, which has Numpy Scipy
https://bitbucket.org/spyder-ide/spyderlib/downloads

But if you can figure out using command line + sublime text, it will be great because eventually we will do that.

We will use Scipy.ndimages
http://docs.scipy.org/doc/scipy/reference/ndimage.html#scipy.ndimage

Scipy image tutorial: http://scipy-lectures.github.io/advanced/image_processing/

Today's class

- What is an image?
- Point Process
- Neighborhood operation
- Convolution

Coding environment

- MATLAB
  - Pro: Well-established packages. Many tutorials and examples online. Great for numerical stuff.
  - I have many years of experience with it.
- Cons:
  - Expensive! Talk to me about getting access.
  - Not a general programming language.

Coding environment

- Numerical Python
- Pro: All the capabilities of MATLAB
  - Free!
  - Real programming Language
  - Used for lots of stuff besides numerical programming
  - By using it, we are contributing to the community of Python users!
- Cons:
  - Needs set up (install packages, import Libraries)
  - documentation is a bit sparse, lack of good tutorials

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.
- Python Image Library (Pillow), kind of limited and not many people use it.

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.

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

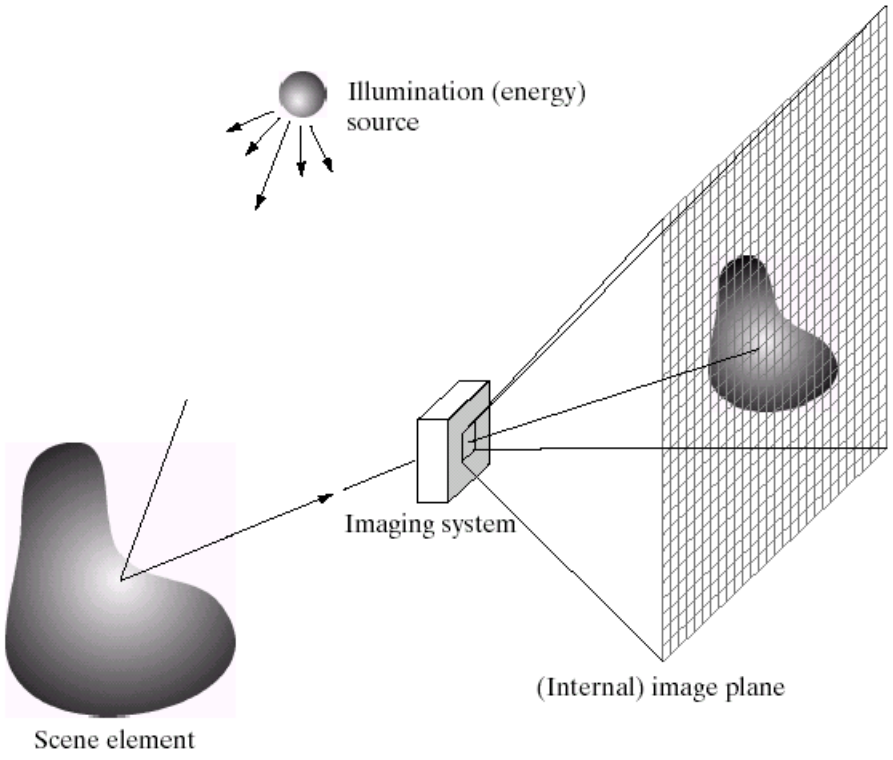
- We will write our own functions!

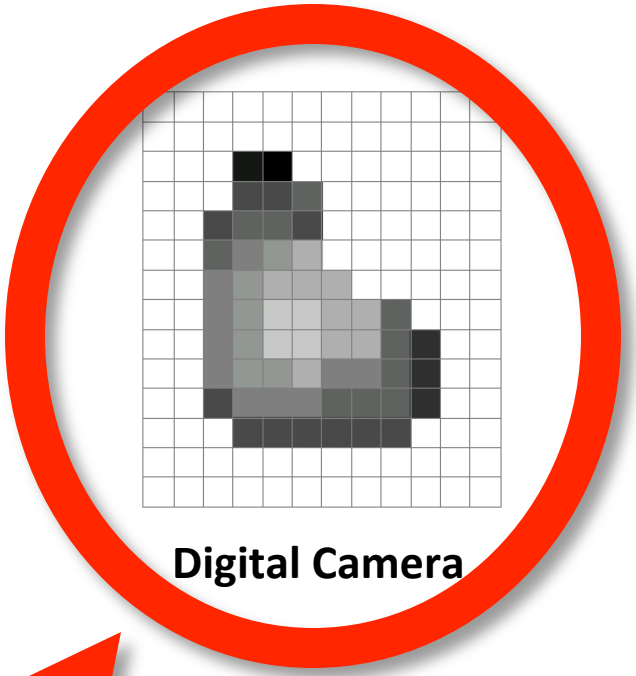
What is an image?

A useful tutorial

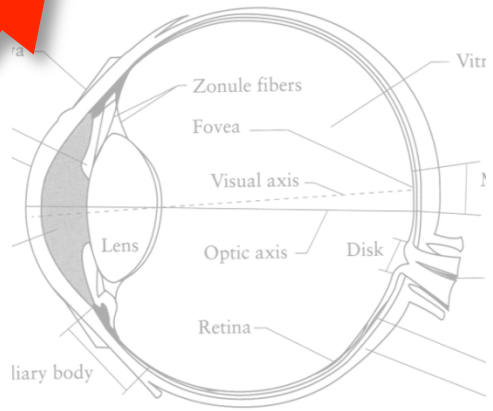
Computer vision for dummies

What is an image?





We'll focus on these in this class

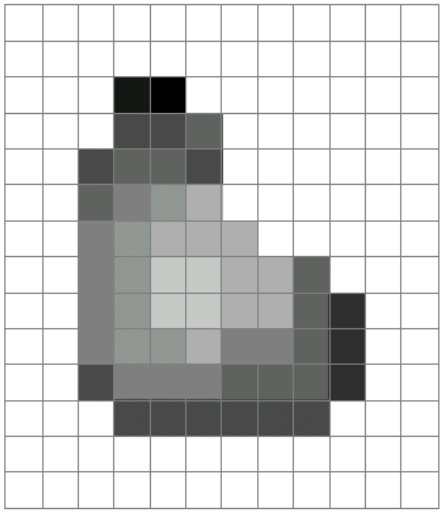


(More on this process later)

Source: A. Efros

What is an image?

- A grid (matrix) of intensity values



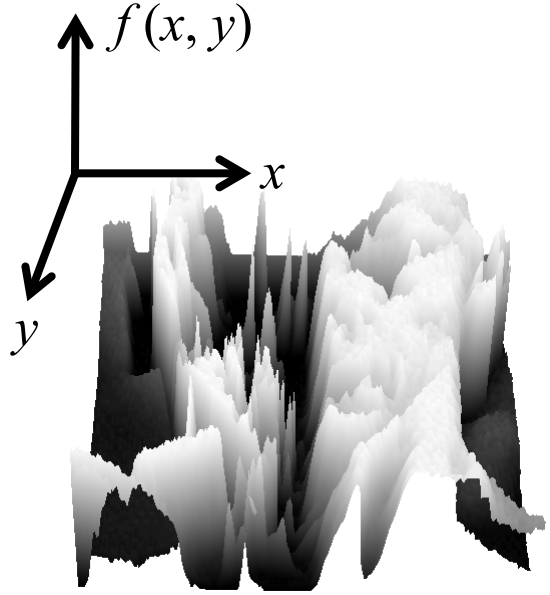
=

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 R^2 to 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

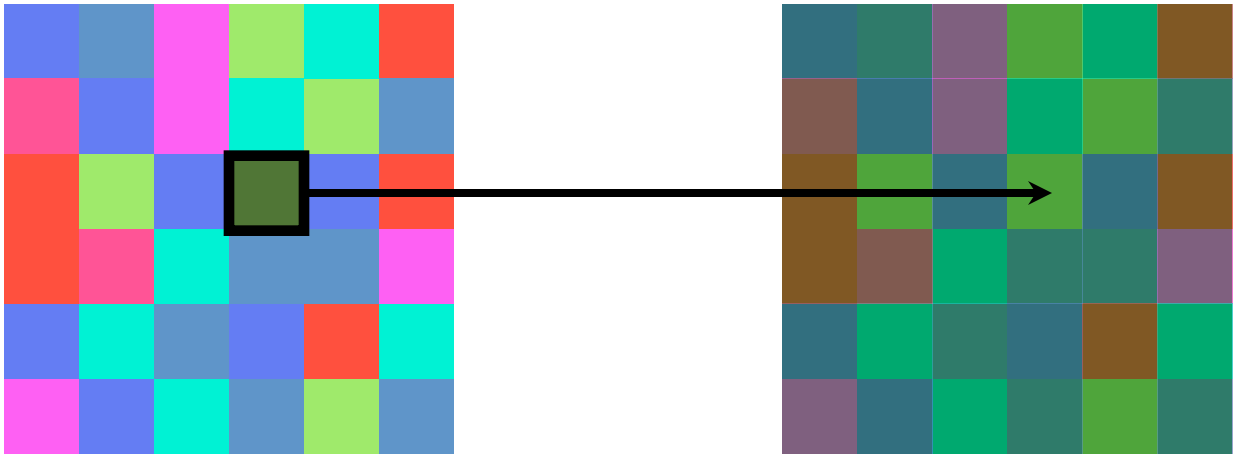
Image Processing

- 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

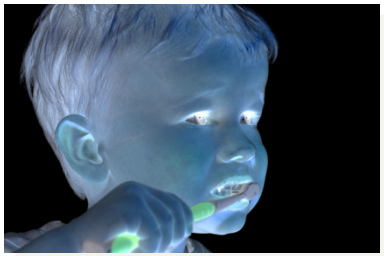
Original

Darken

Lower Contrast

Nonlinear Lower Contrast

Invert



Lighten

Raise Contrast

Nonlinear Raise Contrast

Slide source: S. Narasimhan

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

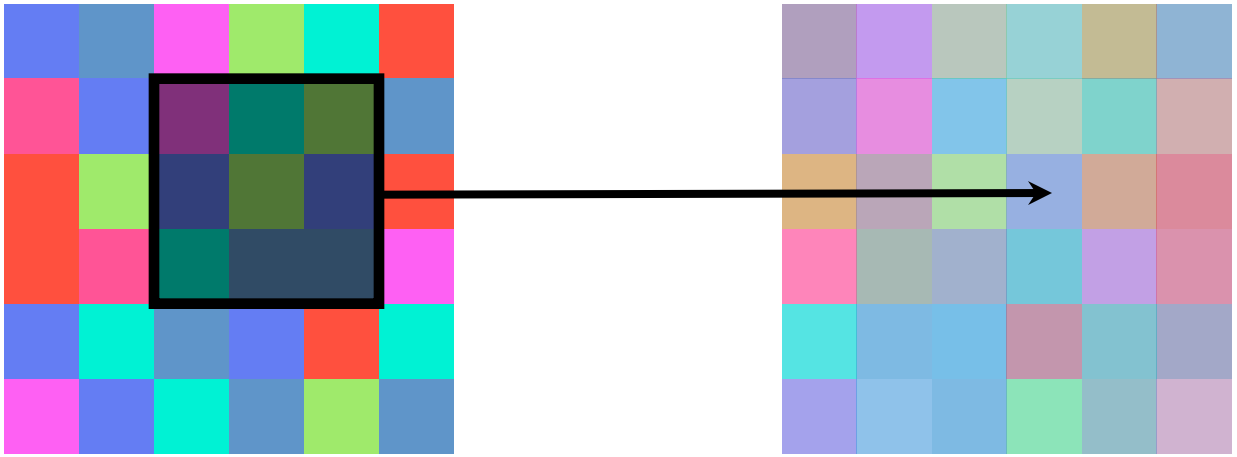
`x * 2`

Nonlinear Raise Contrast

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

Slide source: S. Narasimhan

Neighborhood Operations



Slide source: S. Narasimhan

Neighborhood operations

Image

Edge detection

Blur

Slide source: S. Narasimhan

Neighborhood operations

Image

Edge detection

Blur

Slide source: S. Narasimhan

3×3 Neighborhood

5×5 Neighborhood

7×7 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

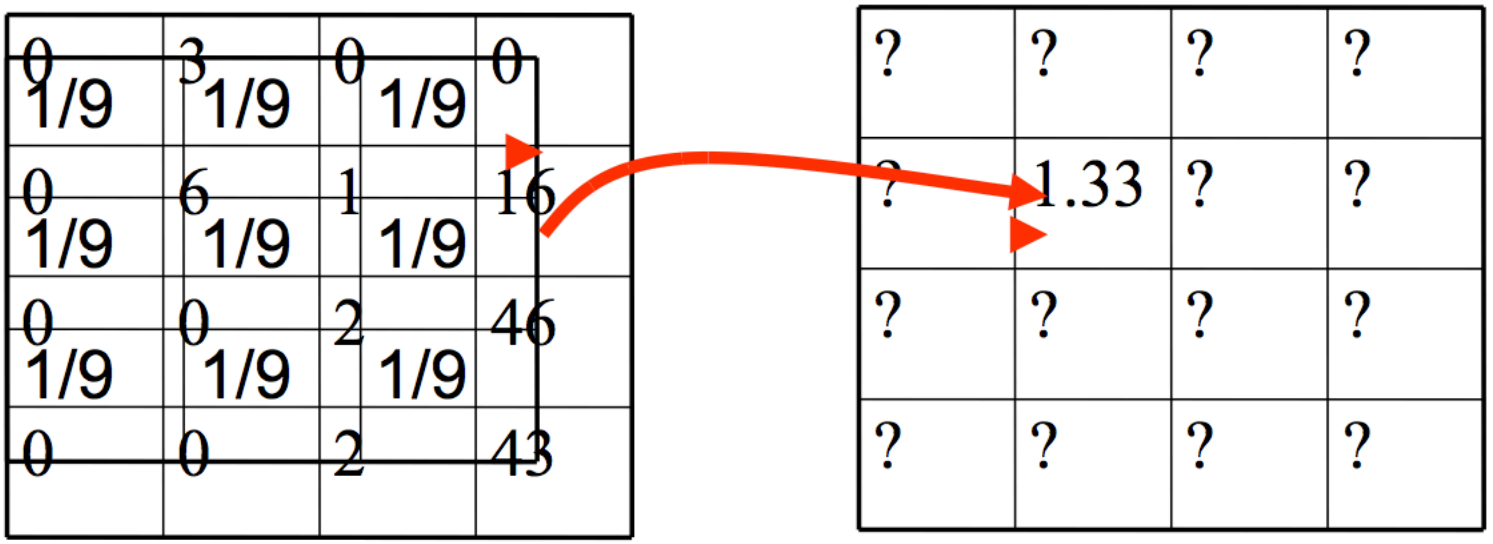
Normalized box filter (3×3)

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

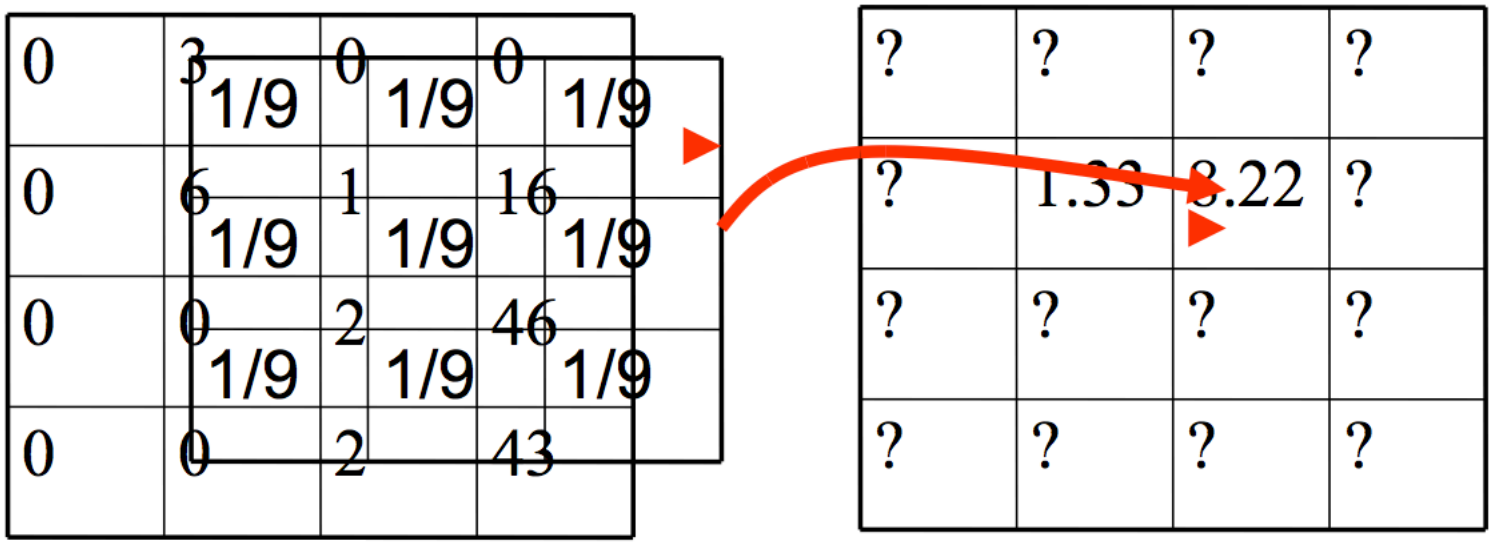
Let's represent this more generally



- 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

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

Slide source: Matthew Tappen

Notation

- Also denoted as
- R = I * K
- We “convolve” I with K

  -Not convolute!

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

Resulting Image

Input Image

Convolution Kernel

Slide source: Matthew Tappen

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
  - `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

Sliding Template View

- Take the template K

1	2	3
4	5	6
7	8	9

- Flip it

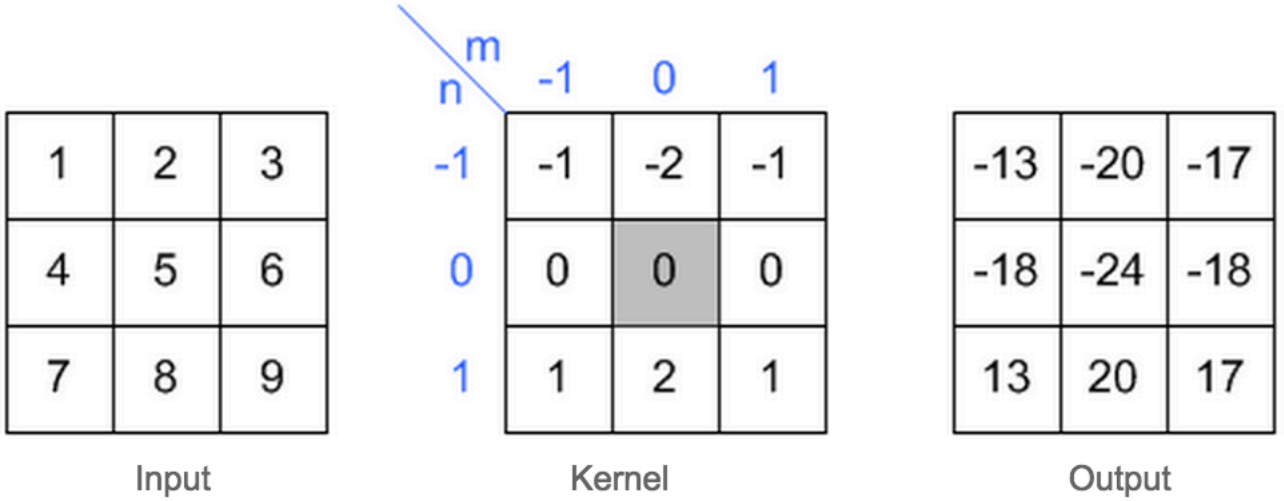
9	8	7
6	5	4
3	2	1

- Slide across image

Slide source: Matthew Tappen

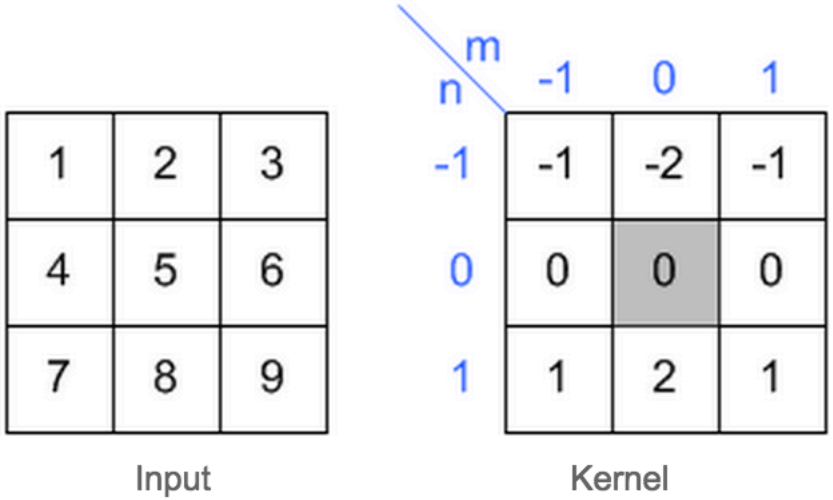
Let's take out paper and pen

- What is the result of the following convolution?



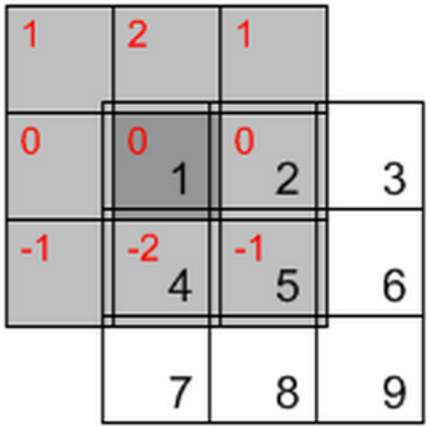
Let’s take out paper and pen

- What is the result of the following convolution?



Let’s take out a paper and a pen

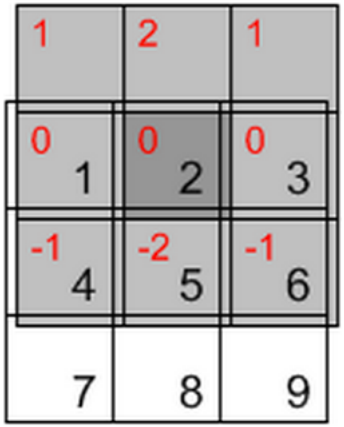
- What is the result of the following convolution?



1*0 + 2*0 + 1*0+
0*0 + 1*0 + 2*0+
(-1)*0 + (-2)*4 +(-1)*5 = -13

Let’s take out a paper and a pen

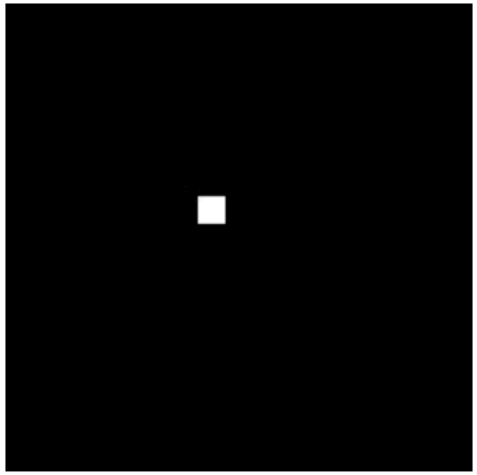
- What is the result of the following convolution?



Rules of a image filter

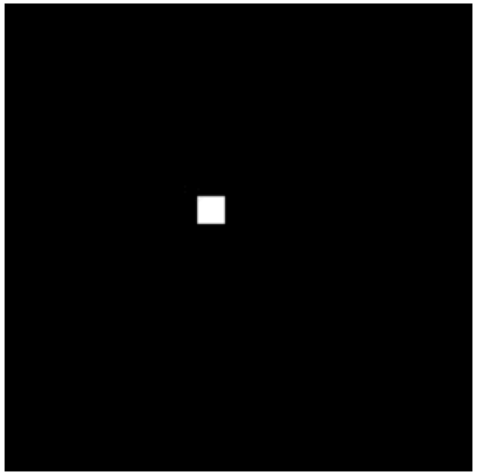
- It's size has to be uneven, so that it has a center, for example, 3×3, 5×5, 7×7
- It doesn't have to, but the sum of all elements of the filter should be 1 if you want the result image to have the same brightness as the original
- If the sum of the element of is larger than 1, the result will be a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

Predict the image



Slide source: Matthew Tappen

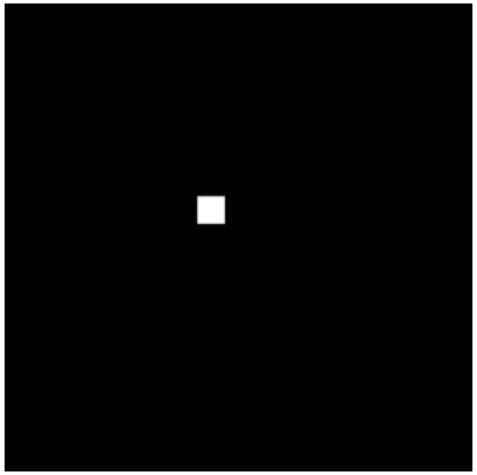
Predict the image



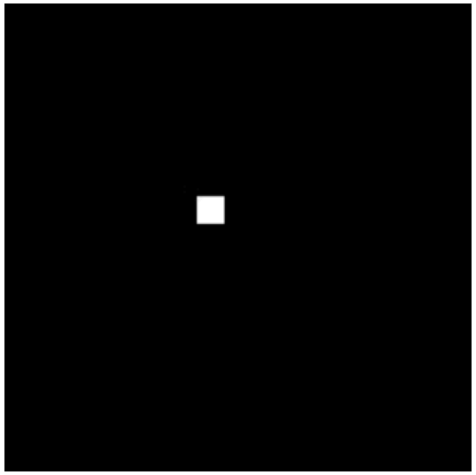
0	0	0
0	1	0
0	0	0

Slide source: Matthew Tappen

Predict the image

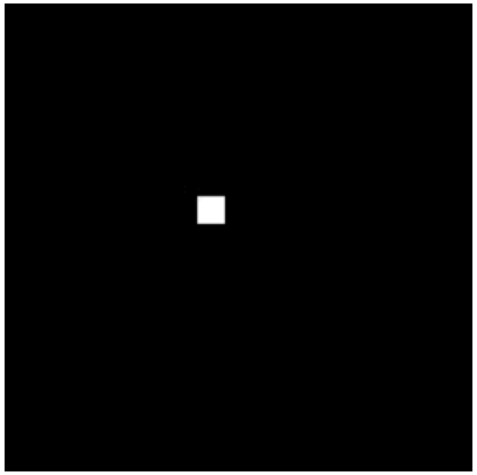


0	0	0
0	1	0
0	0	0



Slide source: Matthew Tappen

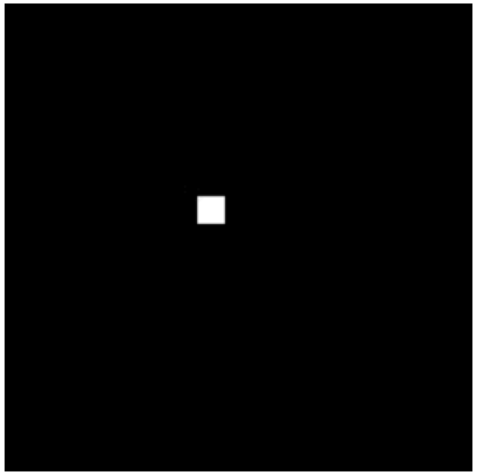
Predict the image



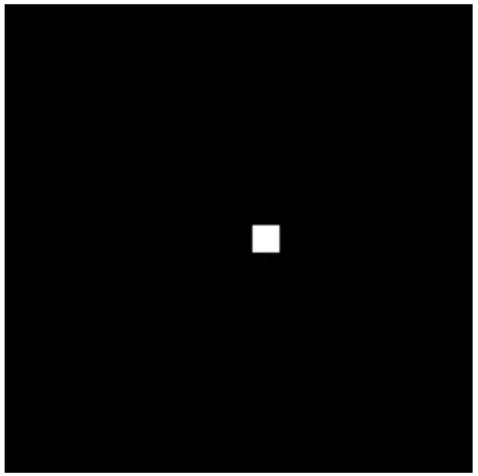
0	0	0
0	0	1
0	0	0

Slide source: Matthew Tappen

Predict the image

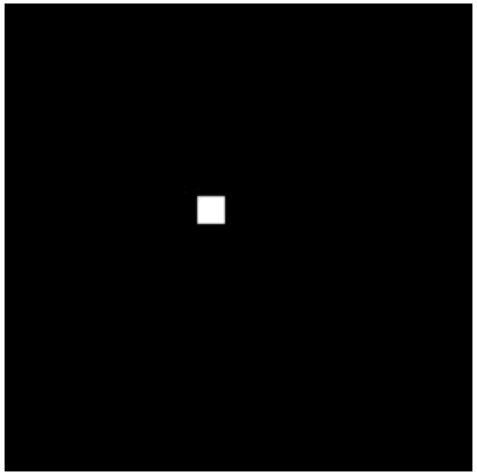


0	0	0
0	0	1
0	0	0



Slide source: Matthew Tappen

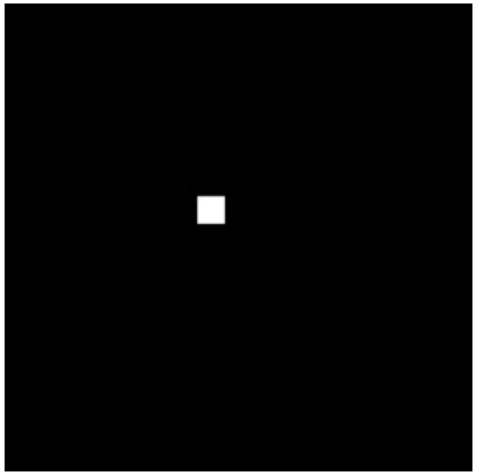
Predict the image



0	1	0
1	2	1
0	1	0

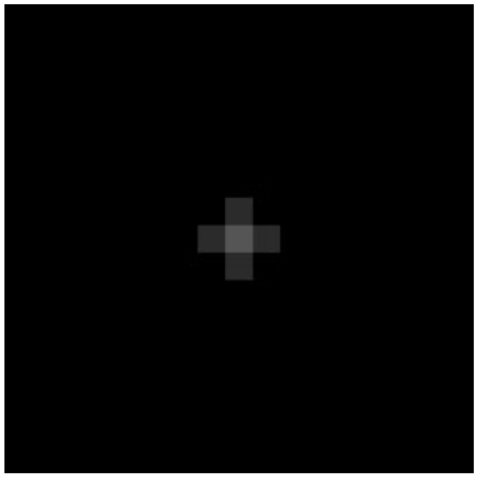
Slide source: Matthew Tappen

Predict the image



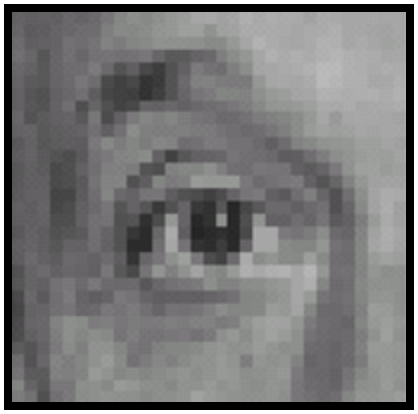
0	1	0
1	2	1
0	1	0

Should be brighter!



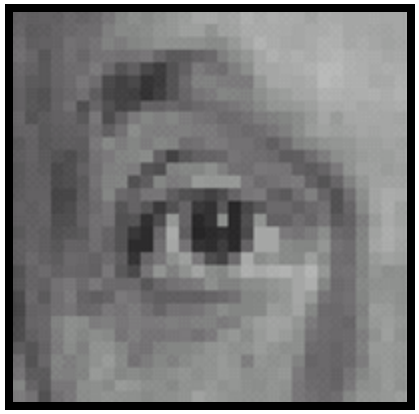
Slide source: Matthew Tappen

Practice with linear filters



Original

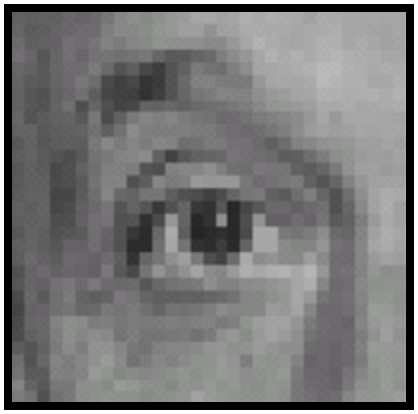
$$*\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$



Filtered
(no change)

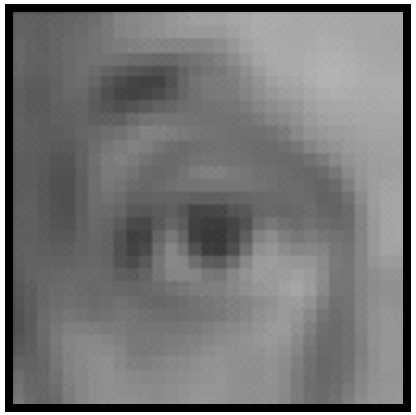
Source: D. Lowe

Linear filters: examples



Original

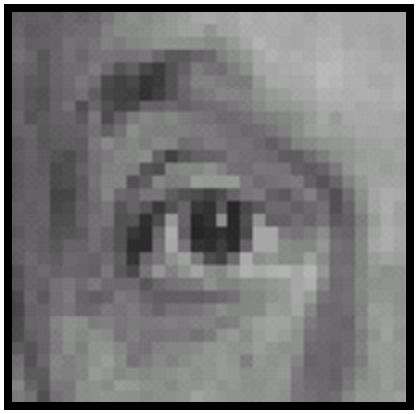
$$* \quad \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$



Blur (with a mean filter)

Source: D. Lowe

Practice with linear filters



Original

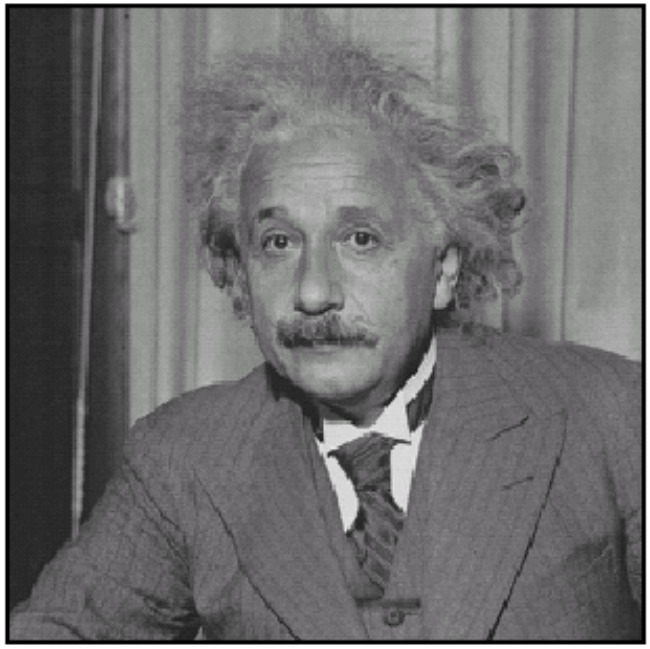
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

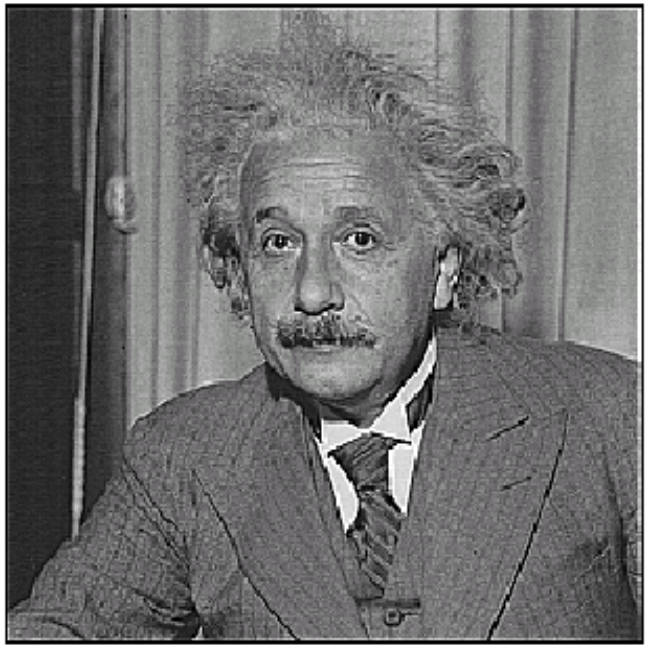
- Accentuates differences with local average

Source: D. Lowe

Sharpening



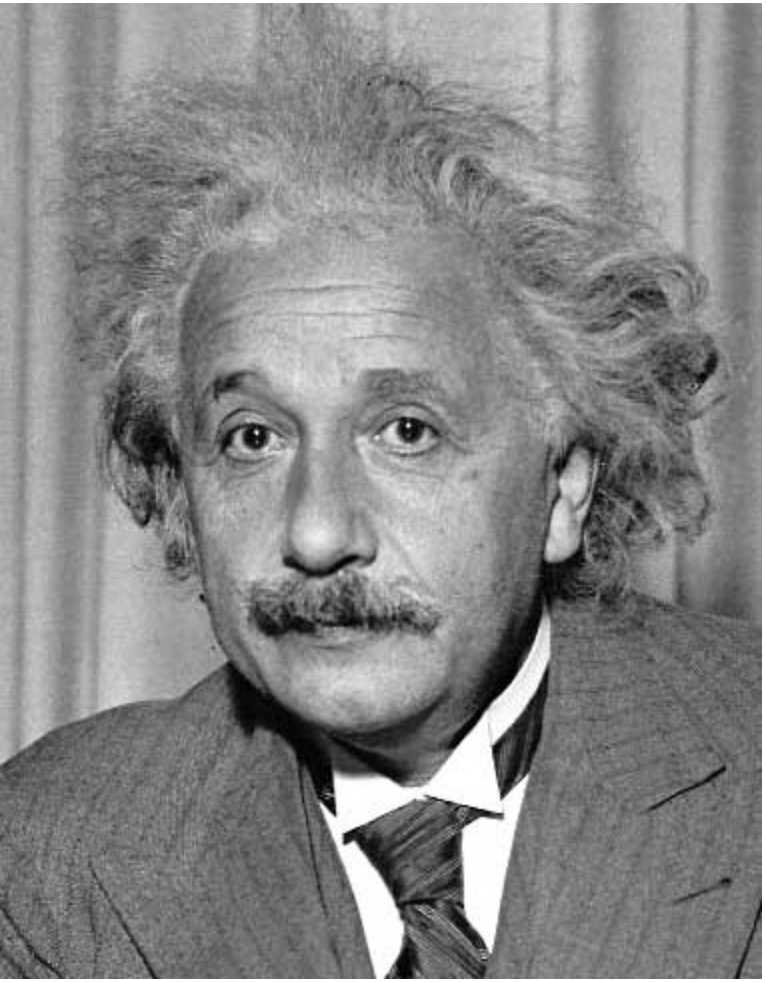
before



after

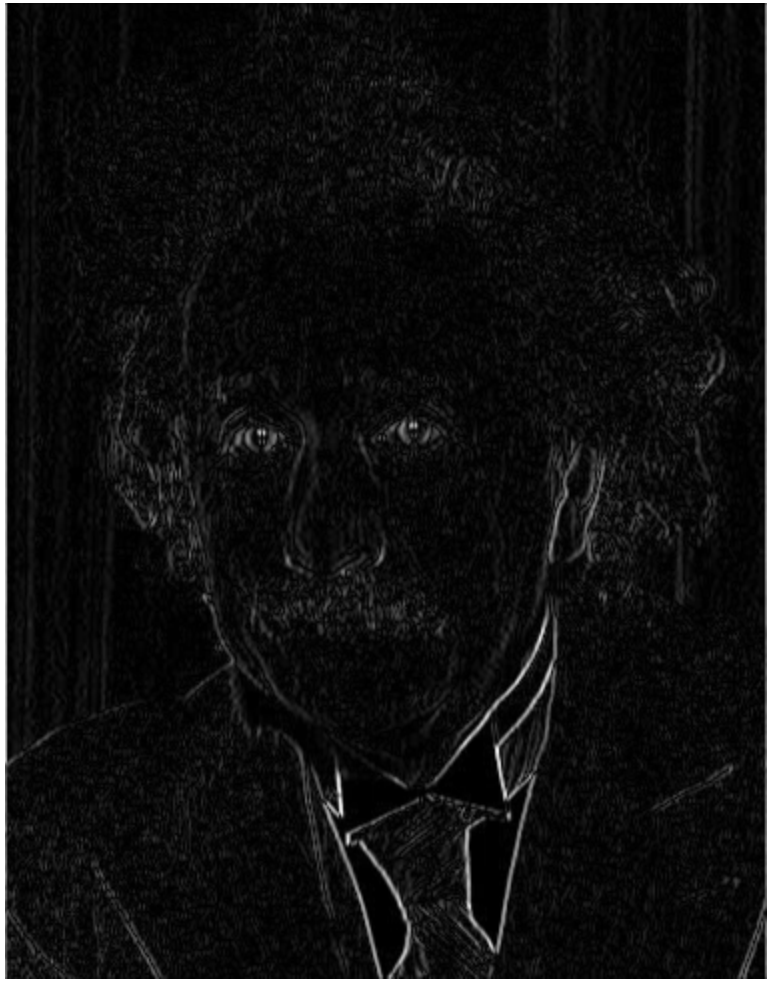
Source: D. Lowe

Other filters



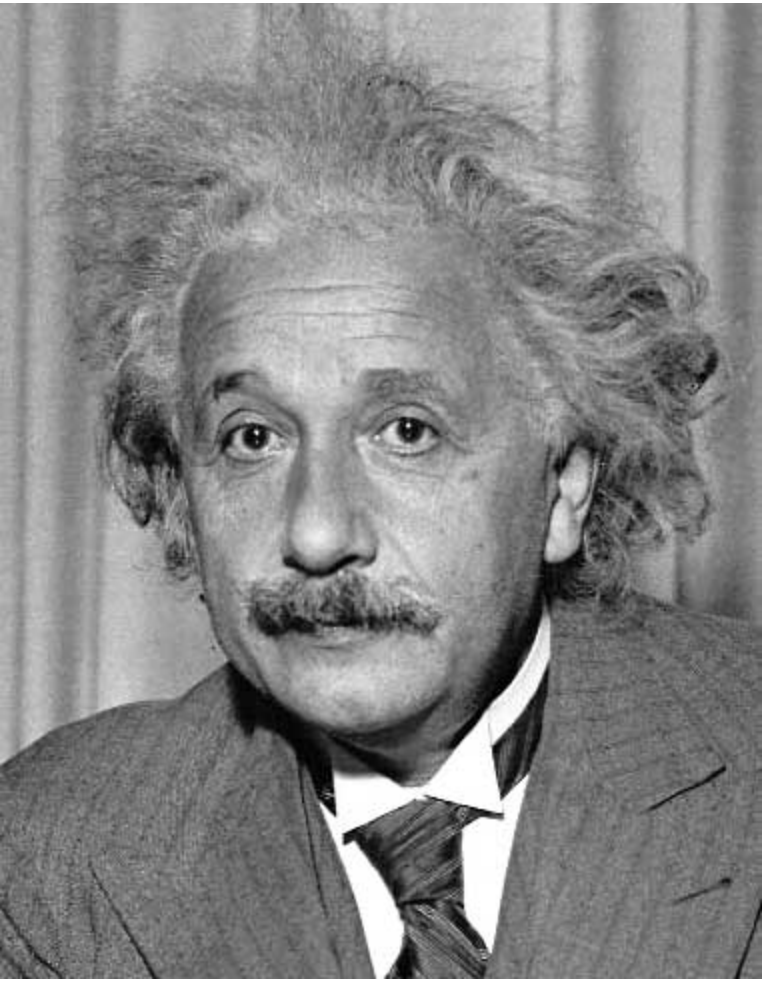
1	0	-1
2	0	-2
1	0	-1

Sobel



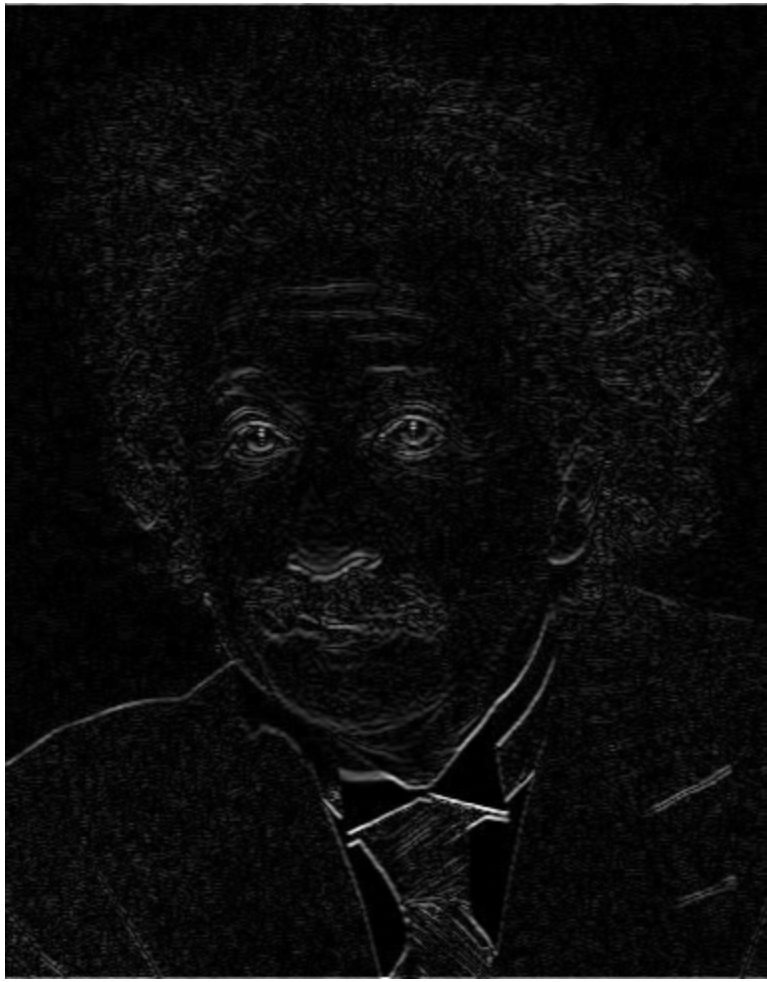
Vertical Edge
(absolute value)

Other filters



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

Sobel



Horizontal Edge
(absolute value)

Next class (Next Thursday)

- Tutorial on image processing with Python (Numpy, Scipy, Matplotlib)
- More on image processing, contrast enhancement, image histograms, Gaussian Filter