Image Representation and Filtering

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE GRANADA

DaSCI

Instituto Andaluz de Investigación en **Data Science** and **Computational Intelligence**

Readings

- ***Image Formation & Sampling***
  - Szeliski (2022), Chapter 2
- ***Image Filtering & Edge Detection***
  - Szeliski (2022), Chapter 3 & 7.2
  - Forsyth and Ponce (2012), Chapter 4, 5.1 & 5.2

- Slides credit: many slides have been adapted from those by Noah Snavely, Derek Hoiem and Stan Birchfield, among others.

What is an image?

"Computer vision may count the trees, estimate the distance to the islands, but it cannot detect the fantasies the people might have had who visited this bay" (Klette, 2014)

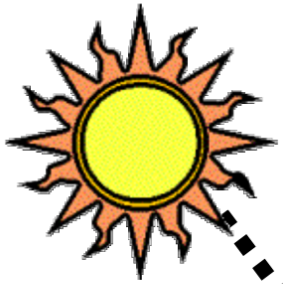
Creator:
Paulo Feal Moreno

Very **complex data** encoding **multiples cues** at **different scales and abstraction levels**.

What is an image?

Source emits photons

Illumination



Photons travel in a straight line

Surface Reflection

They hit an object. Some are absorbed, some bounce off in a new direction.

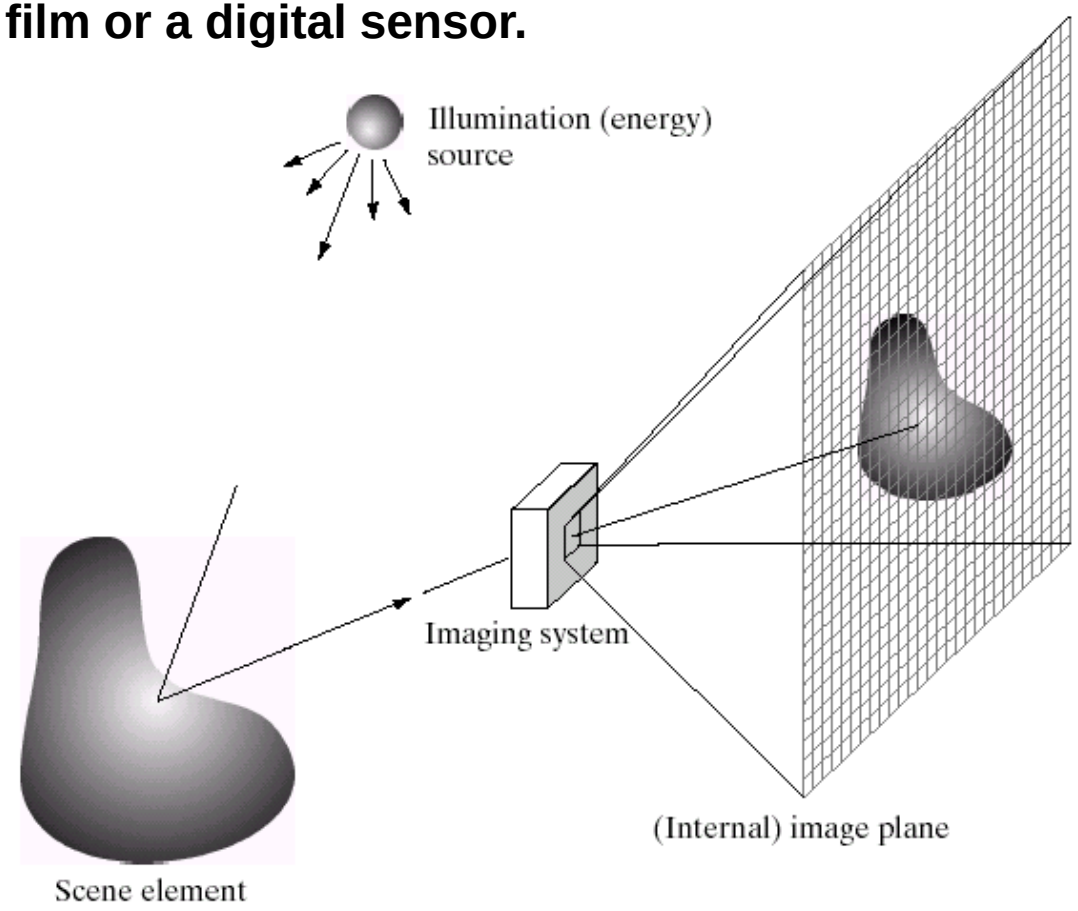
And then some reach an eye/camera and are measured.

Sensor Response

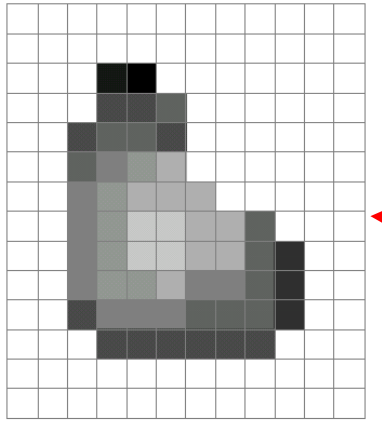
Robert Collins
CSE486, Penn State

What is an image?

A record of a set of light rays that bounce through space and hit a piece of light-sensitive film or a digital sensor.

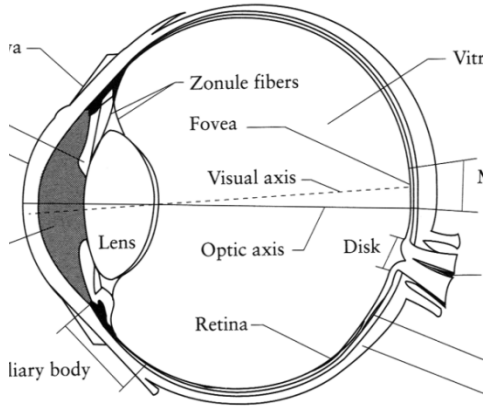


A grid of pixel values.



We focus on this! Digital Images.

A response of our retinas to a light stimulus

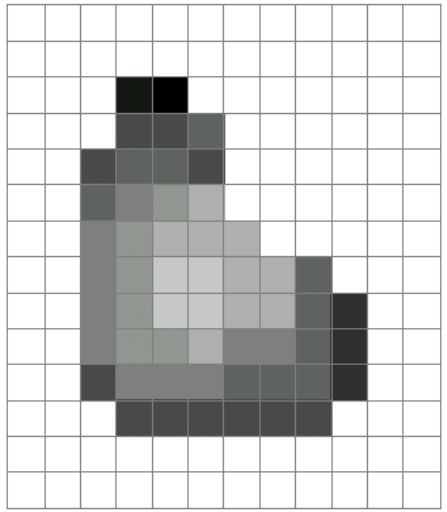


We receive two “images” (one for each eye) and our brain assembles the 3D scene.

Source: A. Efros

What is an image?

- A grid (matrix) of intensity (integer) values



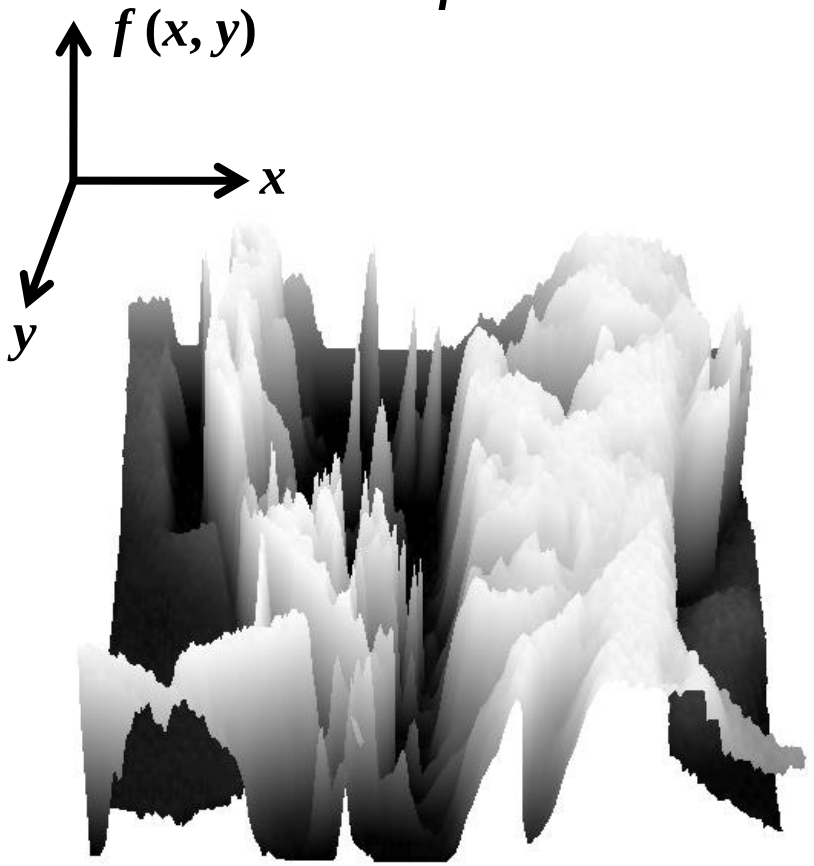
=

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?

- 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)

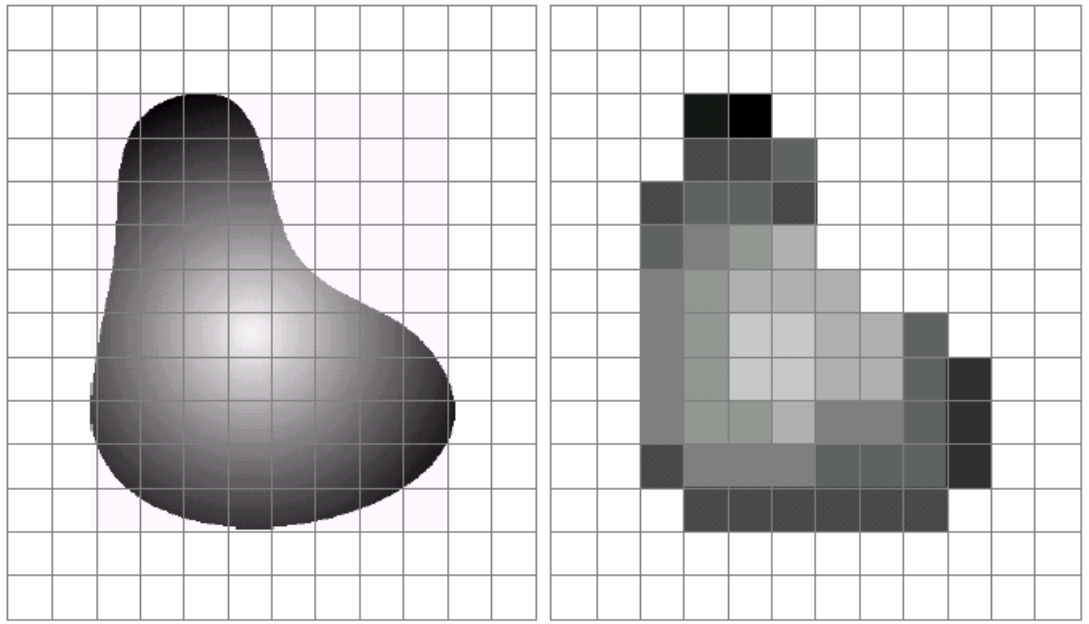


  - A **digital** image is a **discrete** (**sampled**, **quantized**) version of this function

We digitize the coordinate values **We digitize the amplitudes**

What is an image?

– A **digital** image is a **discrete** (**sampled**, **quantized**) version of this intensity function
  - **Sample** the 2D space on a regular grid
  - **Quantize** each sample (round to nearest integer)



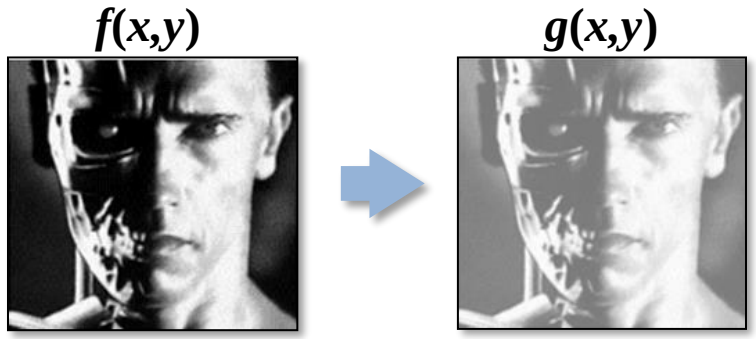
a b

FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

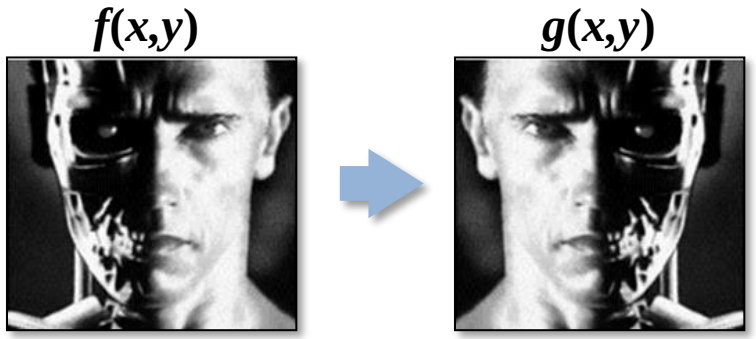
We deal with a **noisy representation of reality** (inaccurate in terms of intensity and space)

Image transformations

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

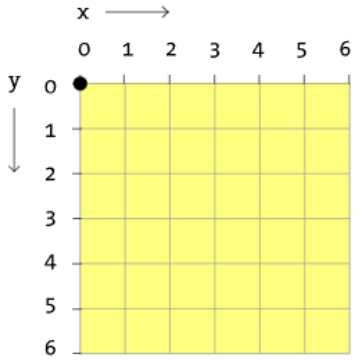


$g(x,y) = f(x,y) + 20$



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

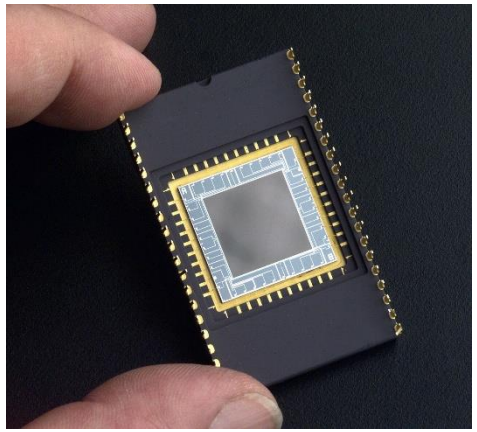
- We'll talk about a special kind of operator: ***convolution*** (local transformation)



In this example, we assume (0,0) is the center of the image.

Note: generally, (0,0) can be found at the top left with the positive direction to the right horizontally and down vertically.

Digital camera



A digital camera replaces film with a sensor array (charge-coupled device (CCD))

- Each cell in the array is a light-sensitive diode that converts photons to electrons.
- The camera collects light that bounces off objects.
- http://electronics.howstuffworks.com/digital-camera.htm

Digital color images

Color images, RGB color space

R

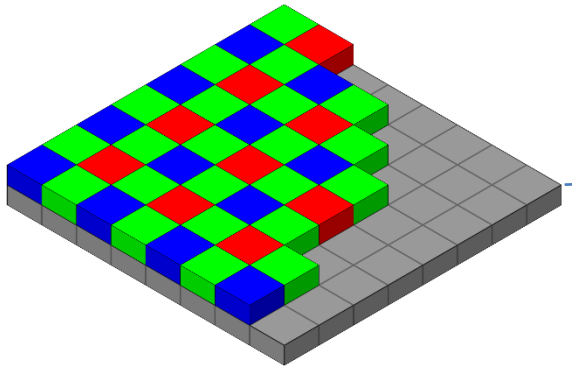
G

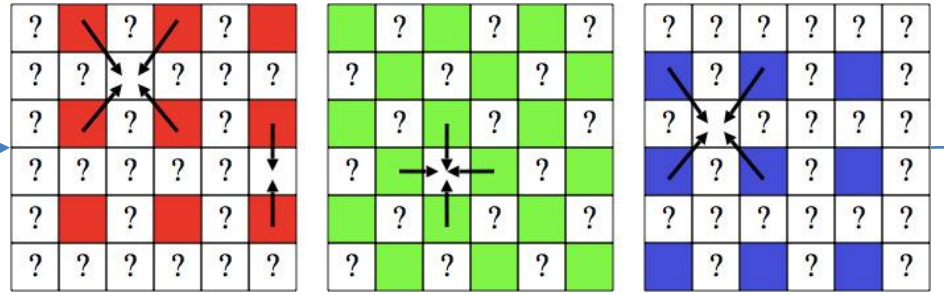
B

Three gray level images

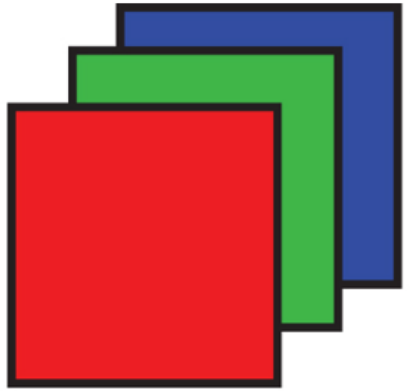
Digital color images

Typically, each 2x2 block is encoded as RGGB





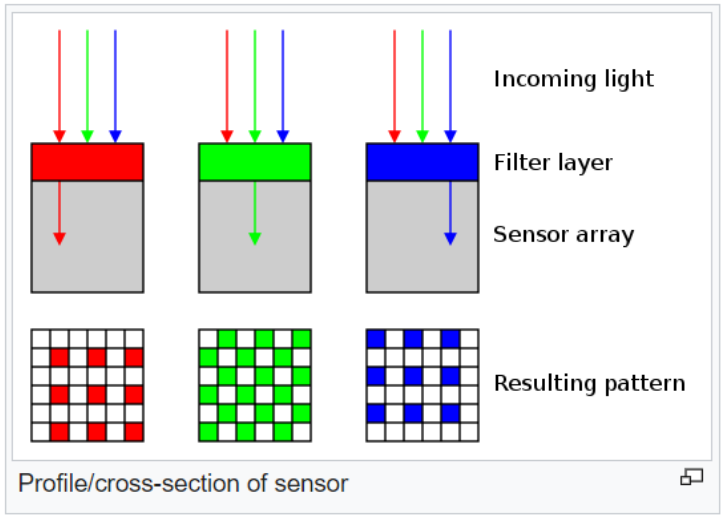
RGB image is computed by interpolation



The **Bayer Filter** is one color filter array (patented in 1976).

Green: 50%, Red:25%, and Blue:25%

Mimic the physiology of the human eye (M and L cone cells are most sensitive to green light).



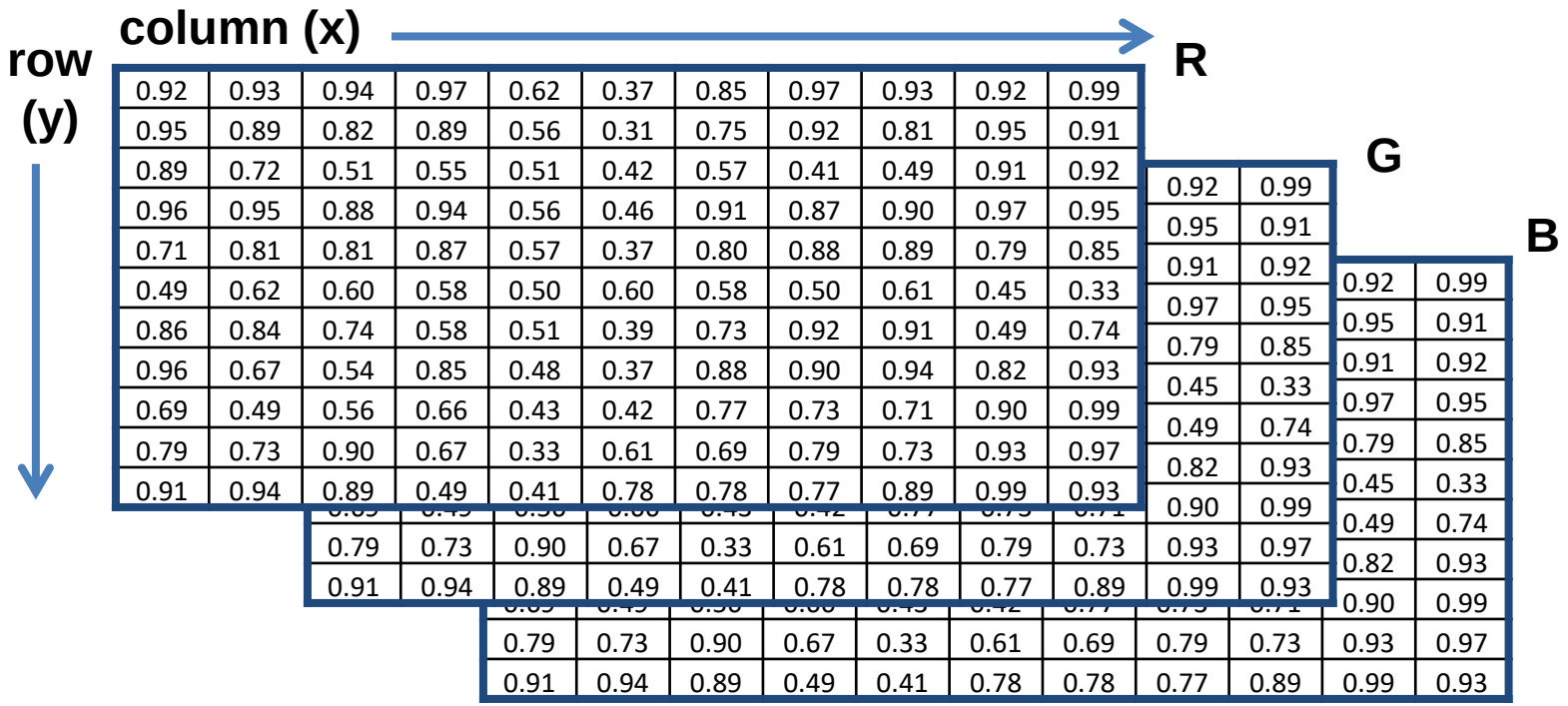
https://en.wikipedia.org/wiki/Bayer_filter

This strategy is much less expensive than a three-CCD camera (https://en.wikipedia.org/wiki/Three-CCD_camera)

Images in Python

```
im = cv2.imread(filename)                    # read image
im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) # order channels as RGB
im = im / 255                                # values range from 0 to 1
```

- RGB image `im` is a H x W x 3 matrix (numpy.ndarray)
- `im[0,0,0]` = top-left pixel value in R-channel
- `im[y, x, c]` = y+1 pixels down, x+1 pixels to right in the c^{th} channel
- `im[H-1, W-1, 2]` = bottom-right pixel in B-channel



Slide credit: Derek Hoiem

Filters

- Filtering
  - **Form a new image whose pixel values are a combination of the original pixel values**
- Why?
  - To get useful information from images
    - E.g., extract edges or contours (to understand shape)
  - To enhance images
    - E.g., to remove noise
    - E.g., to sharpen details
  - To detect patterns
    - Template matching
  - A key operator in Convolutional Neural Networks

Image filtering

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

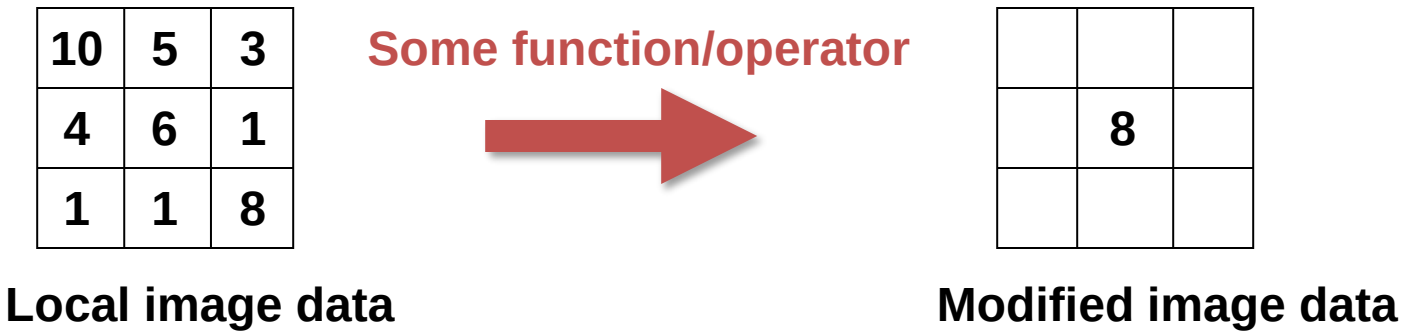


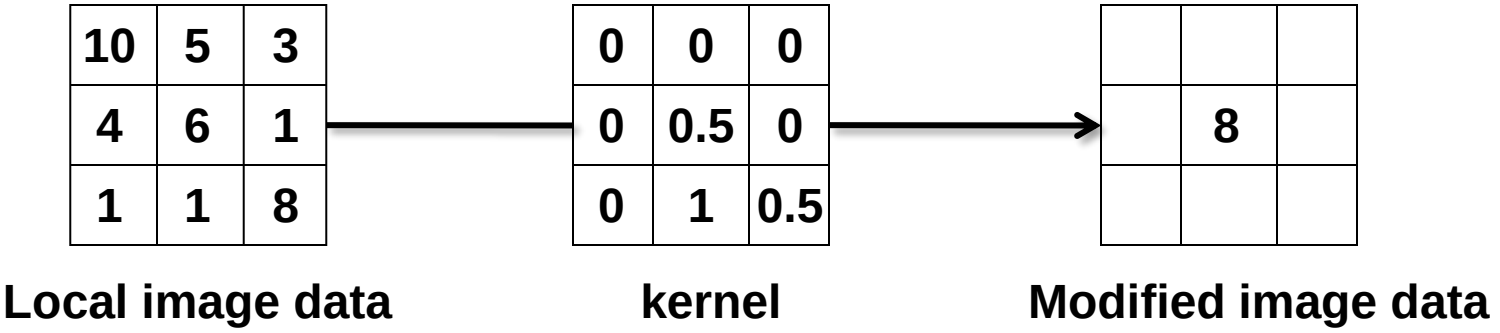
Image filtering

- Modify the pixels in an image based on some function of a **local neighborhood** of each pixel
  - Wait… why a local neighborhood?? Why not global or individual??? 🤔
    - Nearby pixels show dependence between their values
    - No rule for distant pixels/regions (high spatial variability)
    - The isolated and decontextualized analysis of the pixels is worthless. Local information is essential.

Pixels, by themselves, are not relevant. What is relevant is the relationship among pixels.

Linear filtering

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



Common types of noise

- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution
- Different types of noise need different transformations

Original

Salt and pepper noise

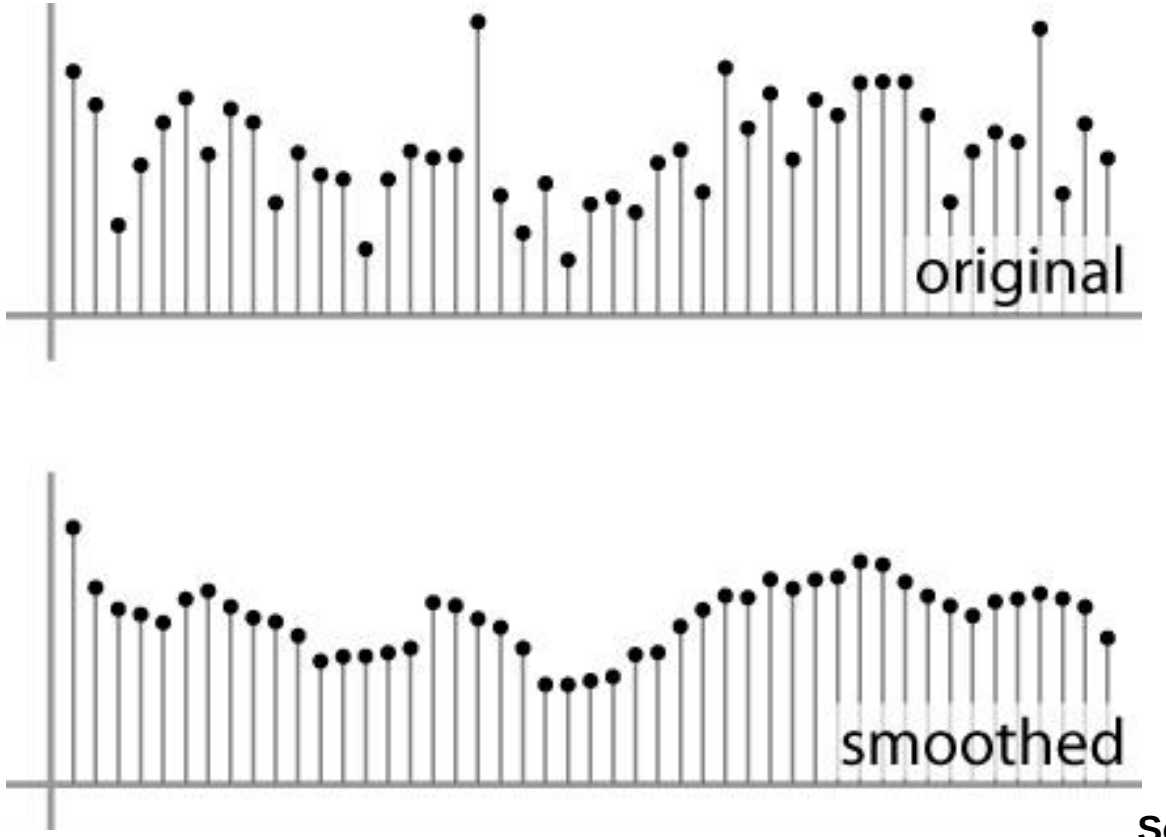
Impulse noise

Gaussian noise

Source: S. Seitz

Smoothing for noise reduction

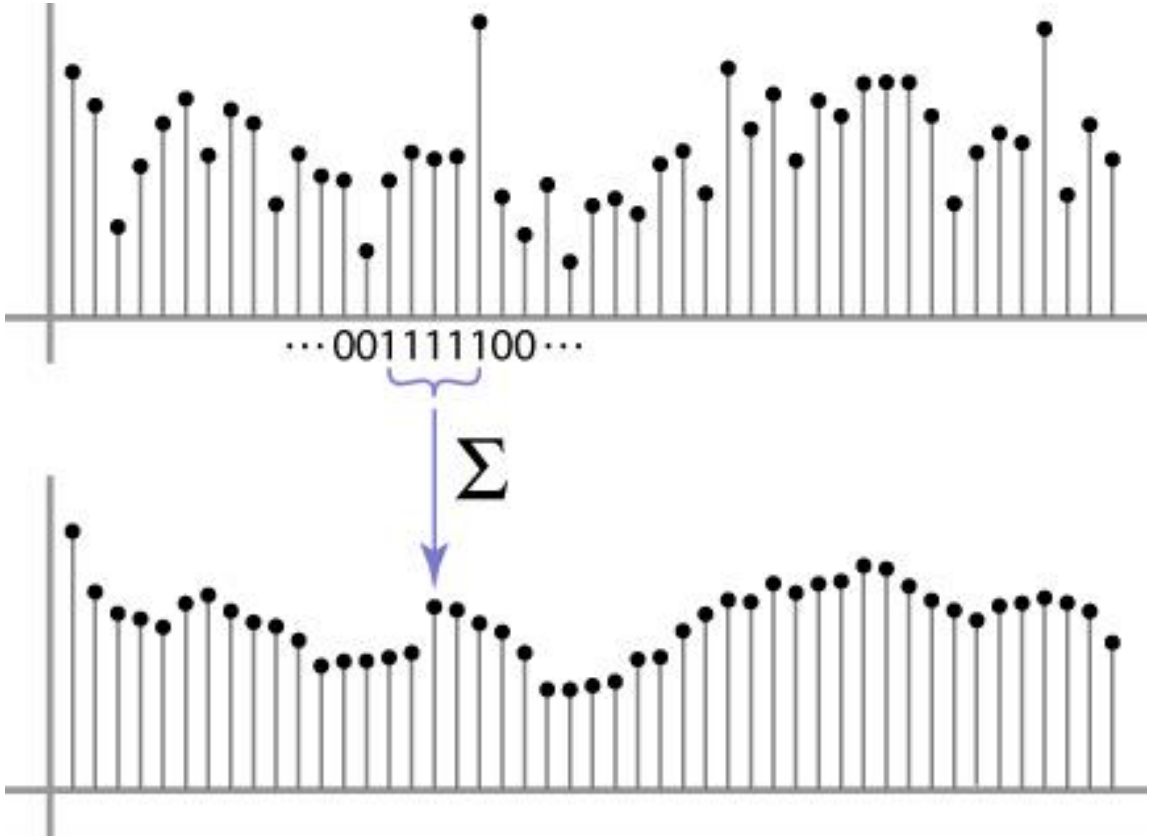
- Let’s replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:

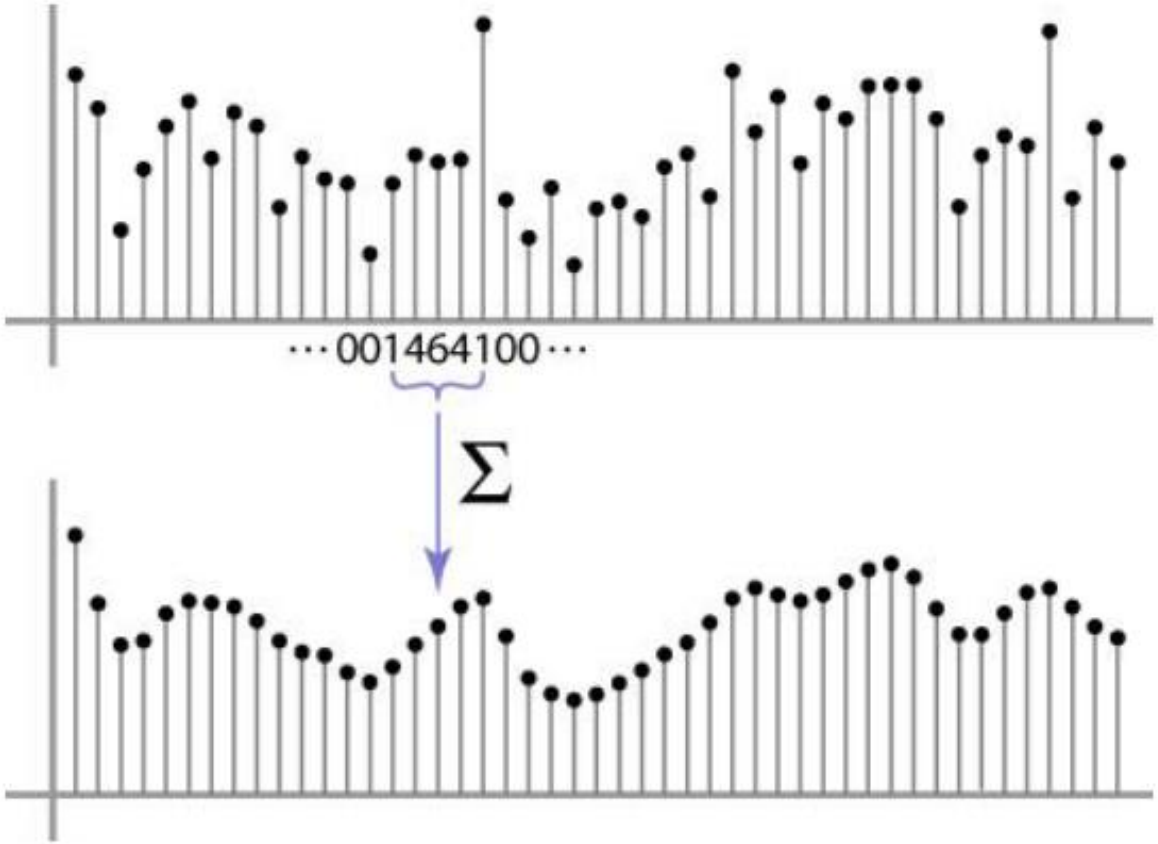


Source: S. Marschner

Smoothing for noise reduction

- Can add weights to our moving average. For instance, non-uniform weights [1, 4, 6, 4, 1] / 16





Cross-correlation

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

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

This is called a cross-correlation operation: $G = H \otimes F$

- Can think of as a "dot product" between local neighborhood and kernel for each pixel

Convolution

- Same as cross-correlation, except that the kernel is “flipped”
  (horizontally & vertically = bottom to top & right to left = rotate it 180˚)

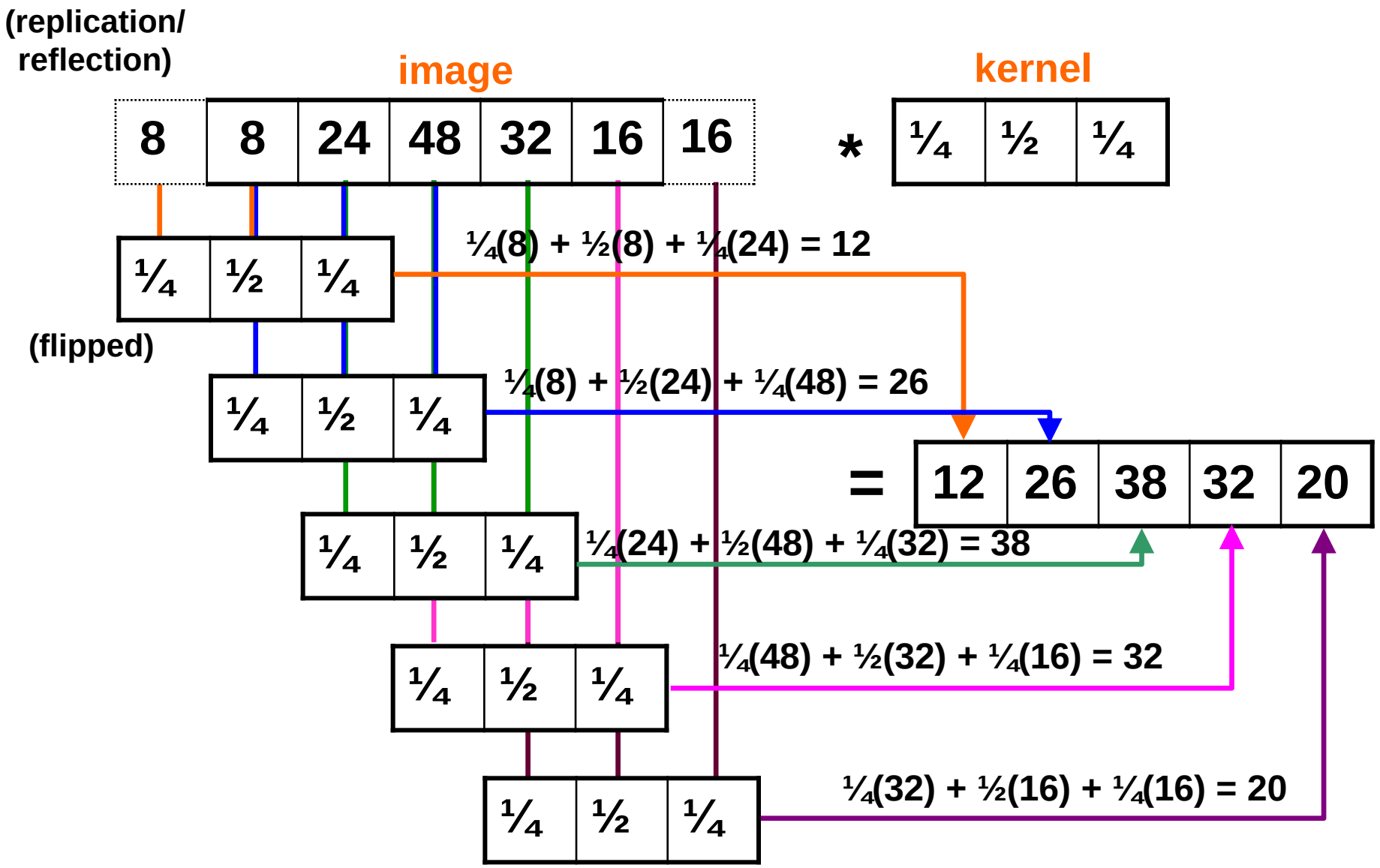
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

This is called a convolution operation:

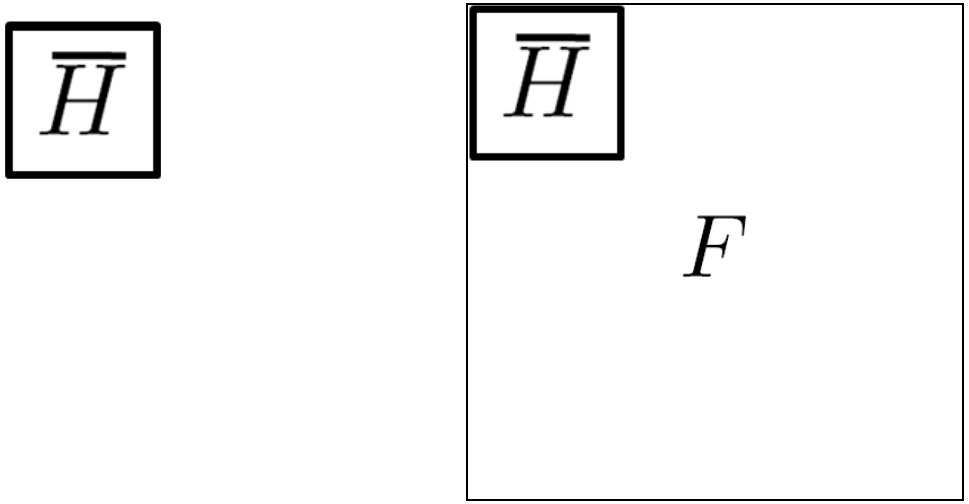
$$G = H * F$$

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

1D convolution example

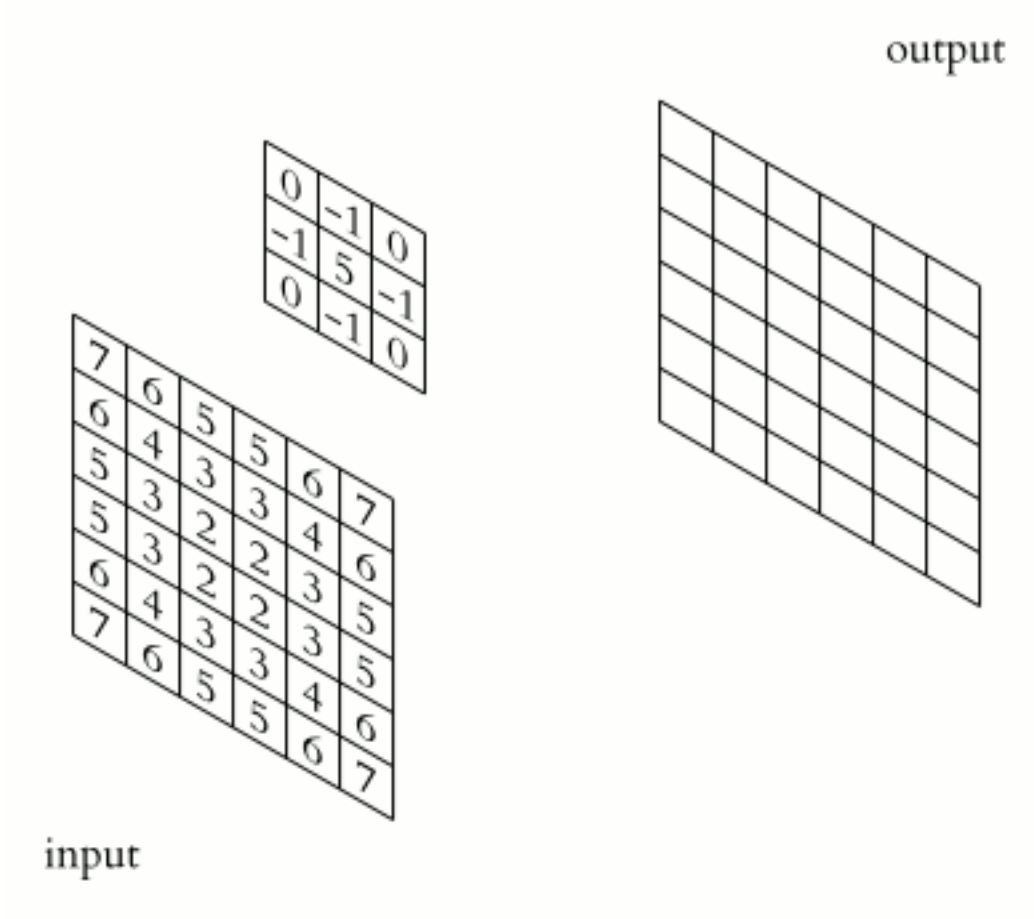


Convolution



Adapted from F. Durand

Convolution

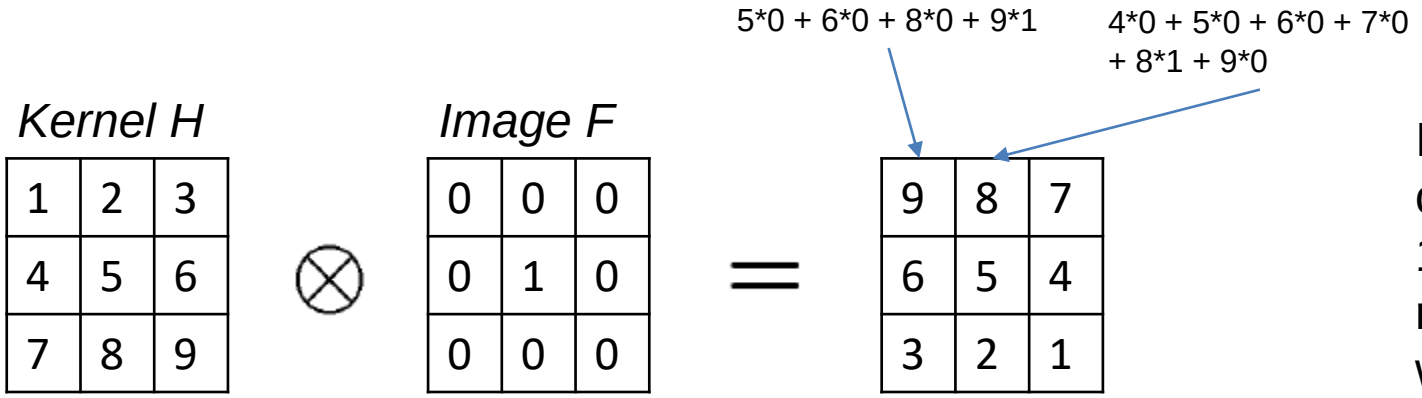


https://upload.wikimedia.org/wikipedia/commons/1/19/2D_Convolution_Animation.gif

(using replicated/
reflected boundaries)

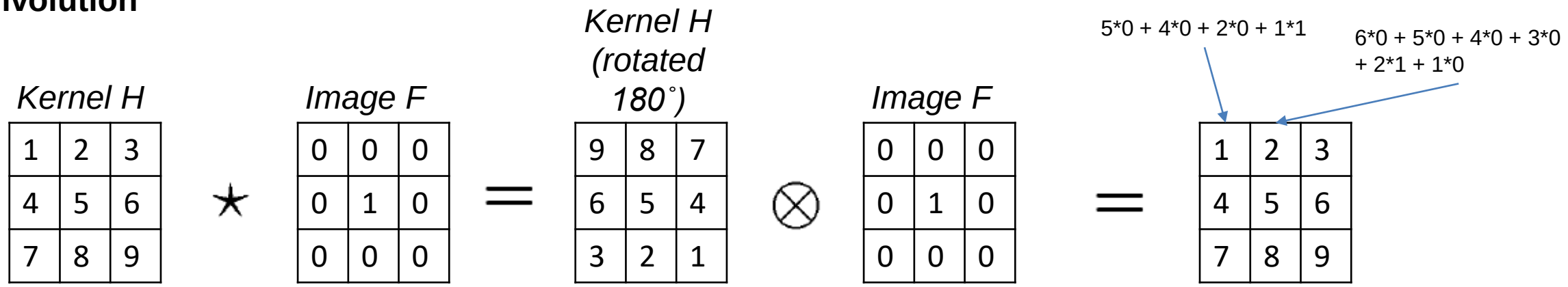
Cross-correlation vs Convolution

Cross-correlation



If the input is an impulse signal, cross-correlation produces the filter rotated 180º. **Convolution gives a more natural output**: if the mask is convolved with an impulse signal, it provides exactly the same mask as output.

Convolution



Cross-correlation vs Convolution (technical note)

- Functions like `cv2.filter2D` actually apply cross-correlation and not convolution!!!

- Other functions, like `scipy.signal.convolve2d`, do flip the kernel.
  - `cv2.filter2D(im, -1, cv2.flip(fil,-1))` same as `signal.convolve2d(im,fil,mode='same')`

- So, please, check the documentation before using the functions!!!

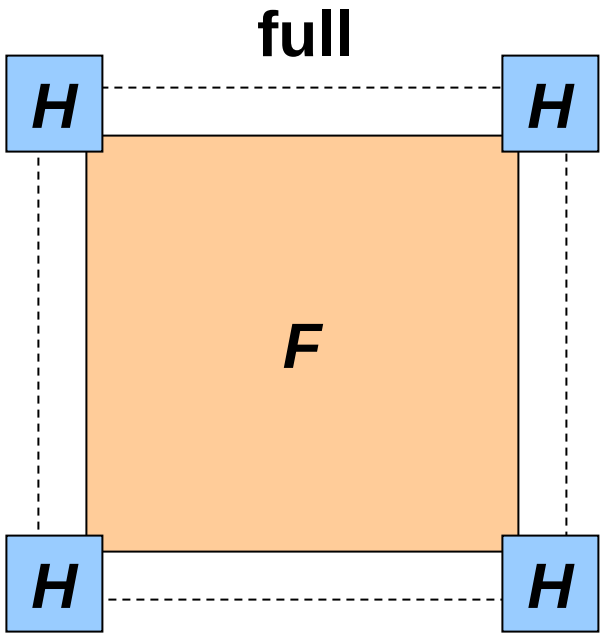
Cross-correlation vs Convolution (technical note)

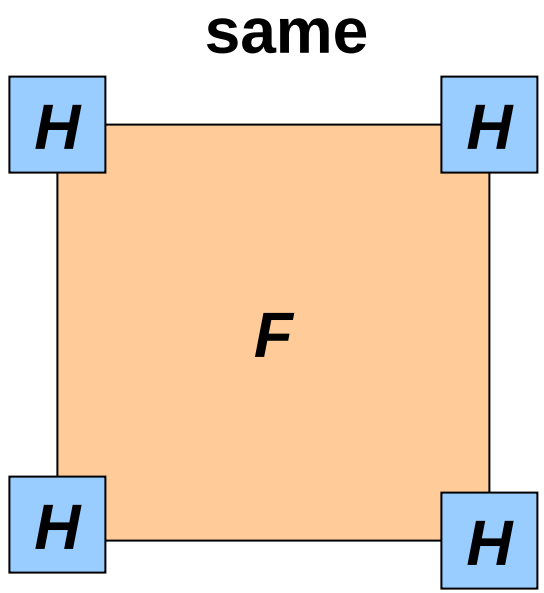
```
import scipy.signal as sp
import numpy as np
import cv2

image = np.array([[0, 0, 0], [0,1,0],[0,0,0]], dtype=np.uint8)
filter = np.array([[1, 2, 3], [4,5,6],[7,8,9]], dtype=np.uint8)

print('Result using sp.convolve2d')
print(sp.convolve2d(image, filter, mode='same'))

print('Result using cv2.filter2D')
print(cv2.filter2D(image, -1, filter, borderType=cv2.BORDER_CONSTANT))

print('Result using cv2.filter2D and cv2.flip')
print(cv2.filter2D(image, -1, cv2.flip(filter,-1), borderType=cv2.BORDER_CONSTANT))
```

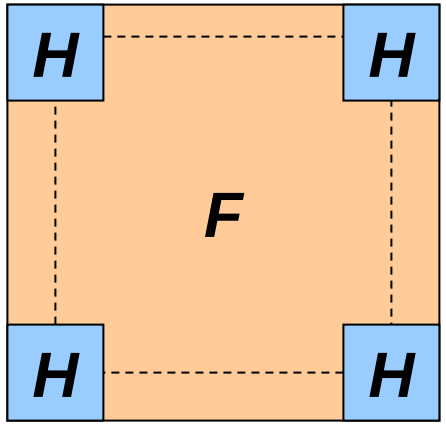
```
Result using sp.convolve2d
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Result using cv2.filter2D
[[9 8 7]
 [6 5 4]
 [3 2 1]]
Result using cv2.filter2D and cv2.flip
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Boundary issues

- What is the size of the output?
- Output size / “shape” options







Kernel H

1	2	3
4	5	6
7	8	9

★

Image F

0	0	0
0	1	0
0	0	0

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

1	2	3
4	5	6
7	8	9

5

Boundary issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black) (a.k.a. zero-padding)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

Boundary issues

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - Types (OpenCV):

    * *BORDER_REPLICATE: aaaaaa | abcdefgh | hhhhhhh*
    * *BORDER_REFLECT: fedcba | abcdefgh | hgfedcb*
    * *BORDER_REFLECT_101: gfedcb | abcdefgh| gfedcba*
    * *BORDER_WRAP: cdefgh | abcdefgh | abcdefg*
    * *BORDER_CONSTANT: iiiiii | abcdefgh | iiiiiii*

Cross-correlation vs Convolution

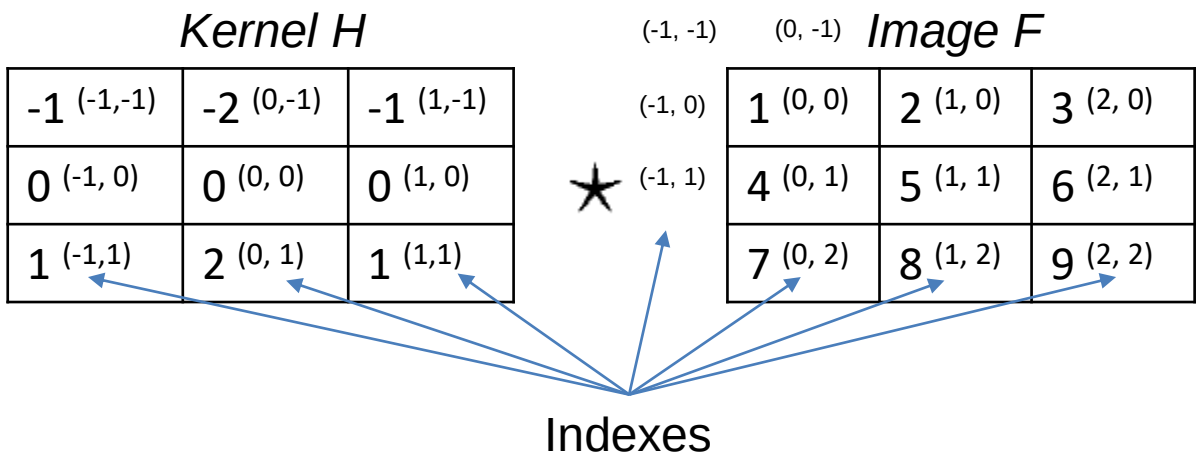
Cross-correlation

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

Convolution

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

How these indexes exactly work? Let's see an example using cross-correlation (convolution is the same: just flip the filter in both dimensions, then apply cross-correlation)



$$G[0,0] = H[-1,-1]\cdot F[-1,-1] + H[-1,0]\cdot F[-1,0] + H[-1,1]\cdot F[-1,1] + H[0,-1]\cdot F[0,-1] + H[0,0]\cdot F[0,0] + H[0,1]\cdot F[0,1] + H[1,-1]\cdot F[1,-1] + H[1,0]\cdot F[1,0] + H[1,1]\cdot F[1,1]$$
$$= -1\cdot 0 + 0\cdot 0 + 1\cdot 0 + (-2)\cdot 0 + 0\cdot 1 + 2\cdot 4 + (-1)\cdot 0 + 0\cdot 2 + 1\cdot 5$$
$$= 13$$

$$G[1,1] = H[-1,-1]\cdot F[0,0] + H[-1,0]\cdot F[0,1] + H[-1,1]\cdot F[0,2] + H[0,-1]\cdot F[1,0] + H[0,0]\cdot F[1,1] + H[0,1]\cdot F[1,2] + H[1,-1]\cdot F[2,0] + H[1,0]\cdot F[2,1] + H[1,1]\cdot F[2,2]$$
$$= -1\cdot 1 + 0\cdot 4 + 1\cdot 7 + (-2)\cdot 2 + 0\cdot 5 + 2\cdot 8 + (-1)\cdot 3 + 0\cdot 6 + 1\cdot 9$$
$$= 24$$

Note: The important thing is that you perform the dot product of an image and a filter (in the case of convolution, rotated 180˚), and replace the central value.

Nice reference on convolution:
https://www.songho.ca/dsp/convolution/convolution.html

Key properties of linear filters

Linearity: if `f` and `g` are images, `a` and `b` scalars, and `L` a linear filter/operator:

$$L(af+bg)=aL(f)+bL(g)$$

homogeneity (or scaling): $L(af) = aL(f)$

+

superposition (or additivity): $L(f+g) = L(f)+L(g)$

Shift invariance: same behavior regardless of pixel location. Operator behaves the same everywhere, i.e. the value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood.

$$L(shift(f)) = shift(L(f))$$

More properties (of convolution)

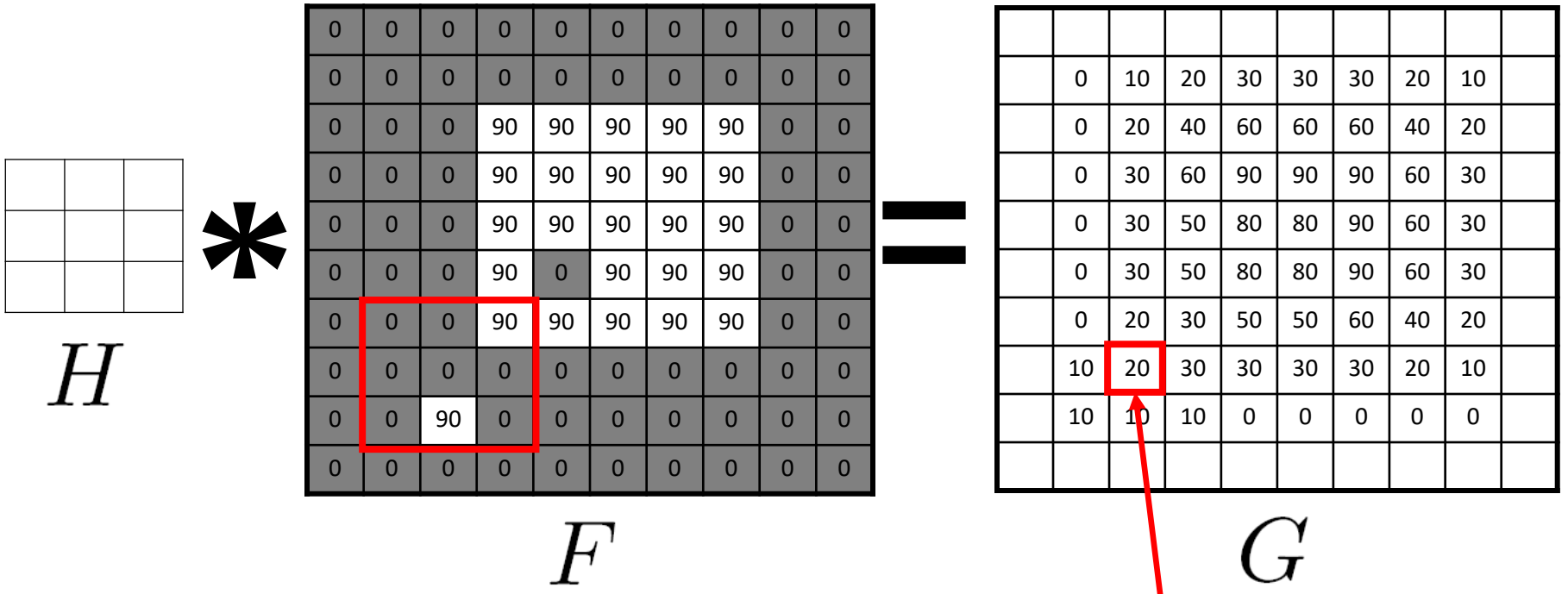
- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal (image)
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k (a * b)$
- Identity: unit impulse $e = [0, 0, 1, 0, 0]$,
  $a * e = a$

Source: S. Lazebnik

Cross-correlation vs Convolution

Cross-correlation	Convolution
Measurement of the **similarity** between two signals/sequences.	Measurement of **effect** of one signal on the other signal.
If the filter is symmetric/isotropic, like a Gaussian or a Laplacian, **it makes no difference**. When the filter is not symmetric, like a derivative, it makes a difference.	
Not associative.	**Associative**
Not commutative.	**Commutative**.
	Associativity allows you to "pre-convolve" filters (i.e. **build complex filters from simple ones**), so you convolve the image with a single filter. • If we have an image f, and we want to convolve it with g and then with h. Knowing that $f*g*h=f*(g*h)$, we can convolve g and h, create a single filter, and then convolve f with it.
If you are doing **template matching**, i. e. looking for a single template, correlation is sufficient.	If you need to **use multiple filters** in succession, and you need to perform this operation on multiple images, it makes sense to convolve the multiple filters into a single filter ahead of time.

Mean filtering



$$\frac{1 \cdot 90}{9} + \frac{1 \cdot 90}{9} = 10 + 10 = 20$$

Mean filtering/Moving average

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

1	1	1
1	1	1
1	1	1

H

$F[x, y]$

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

$G[x, y]$

Mean filtering/Moving average

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

H

$F[x, y]$

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

$G[x, y]$

	0	10							

Mean filtering/Moving average

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

H

$F[x, y]$

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

$G[x, y]$

	0	10	20						

Mean filtering/Moving average

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

H

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

Mean filtering/Moving average

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

H

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30				

Mean filtering/Moving average

What happens if we don't divide by 9?? 🤔

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

H

$F[x, y]$

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

$G[x, y]$

	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	

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

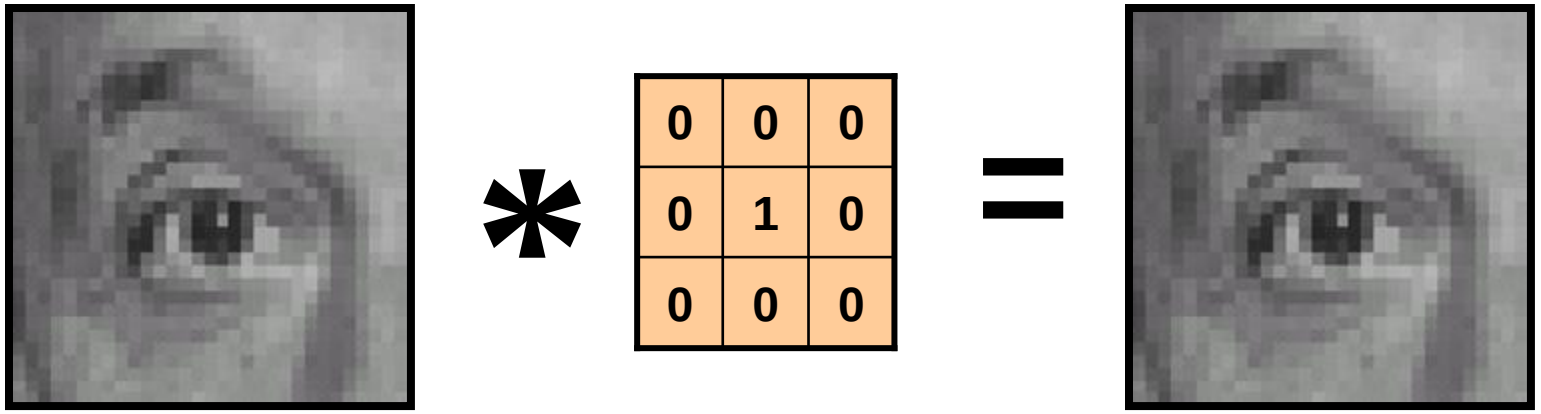
Linear filters: examples

Original

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

Source: D. Lowe

Linear filters: examples



Original

Identical image

Source: D. Lowe

Linear filters: examples

Original

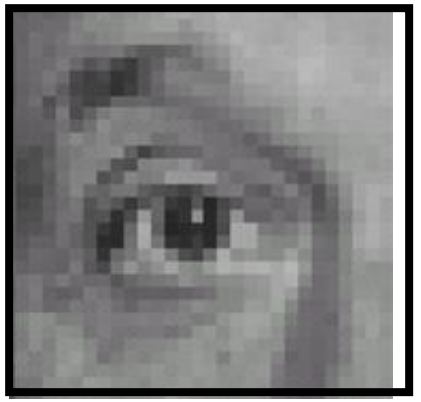
$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = ?$$

Source: D. Lowe

Linear filters: examples

Original

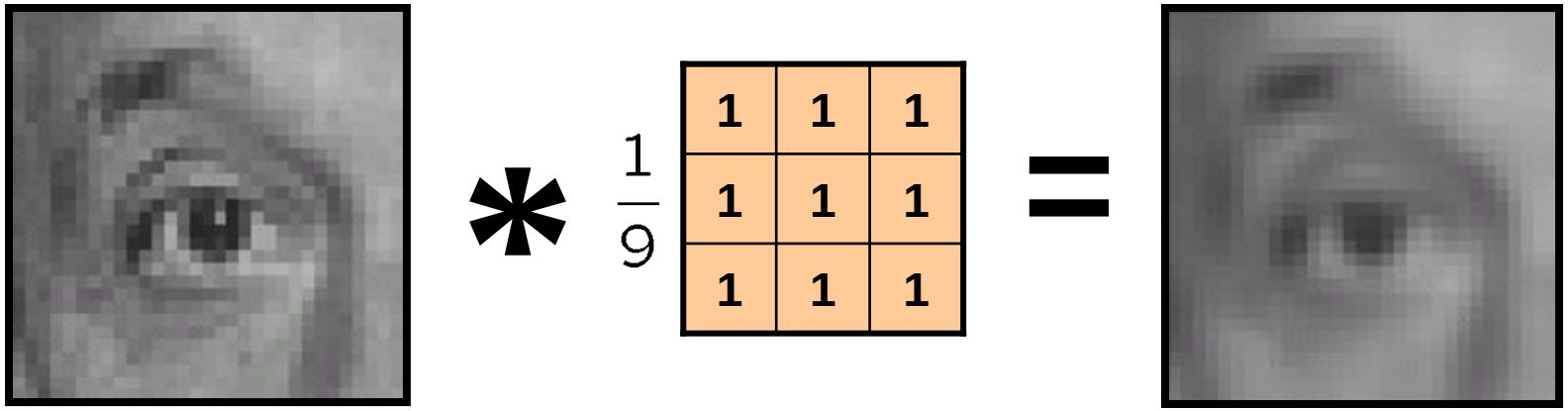
$$*\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} =$$



Shifted left by 1 pixel

Source: D. Lowe

Linear filters: examples



Original

Blur (with a mean filter)

"box filter" (all mask values are equal): linear filter in which each pixel in the resulting image has a value equal to the average value of its neighboring pixels in the input image

Source: D. Lowe

Linear filters: examples

Original

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

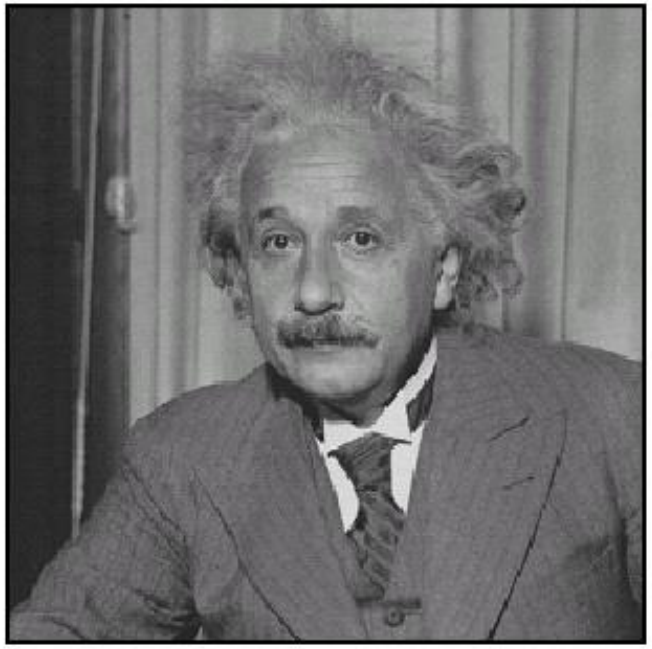
Sharpening filter (accentuates edges)

We enhance the intensity of each pixel

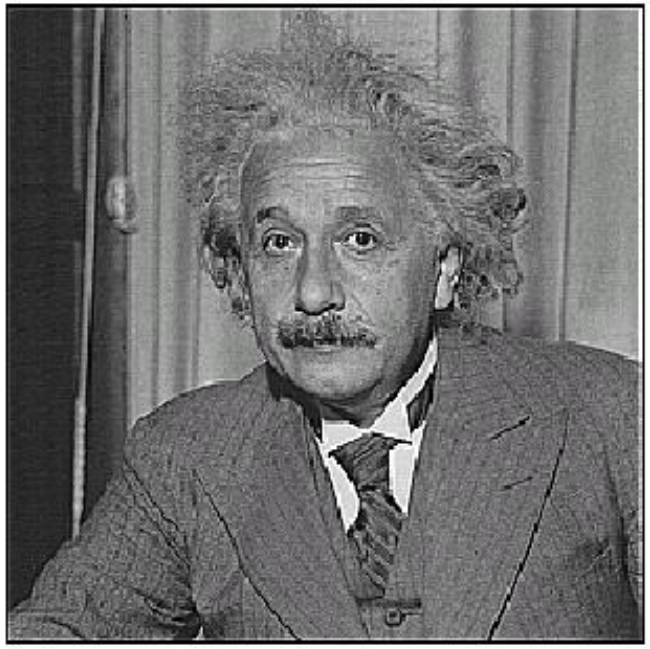
We smooth the image (remove high frequencies)

Source: D. Lowe

Sharpening



before



after

Source image

Sharpened Image

Highly sharpened image

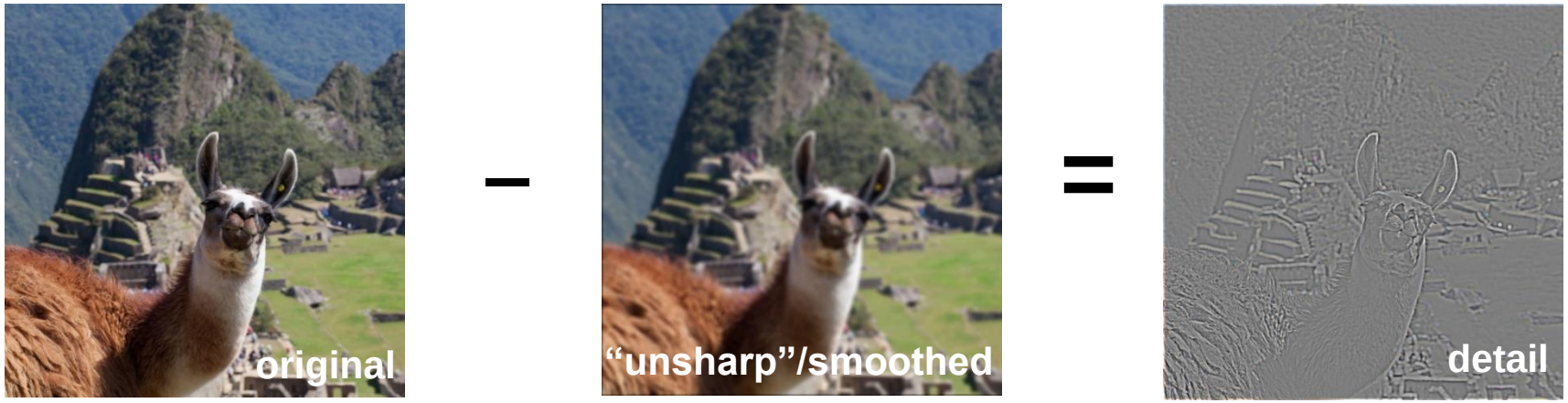
https://en.m.wikipedia.org/wiki/Unsharp_masking#/media/File%3AUnsharped_eye.jpg

Sharpening amplifies the high-frequency components of the image.

Unsharp Masking

Unsharp masking is an image sharpening technique

- What does blurring take away?



(This “detail extraction” operation is also called a *high-pass filter*)

Let’s add it back:

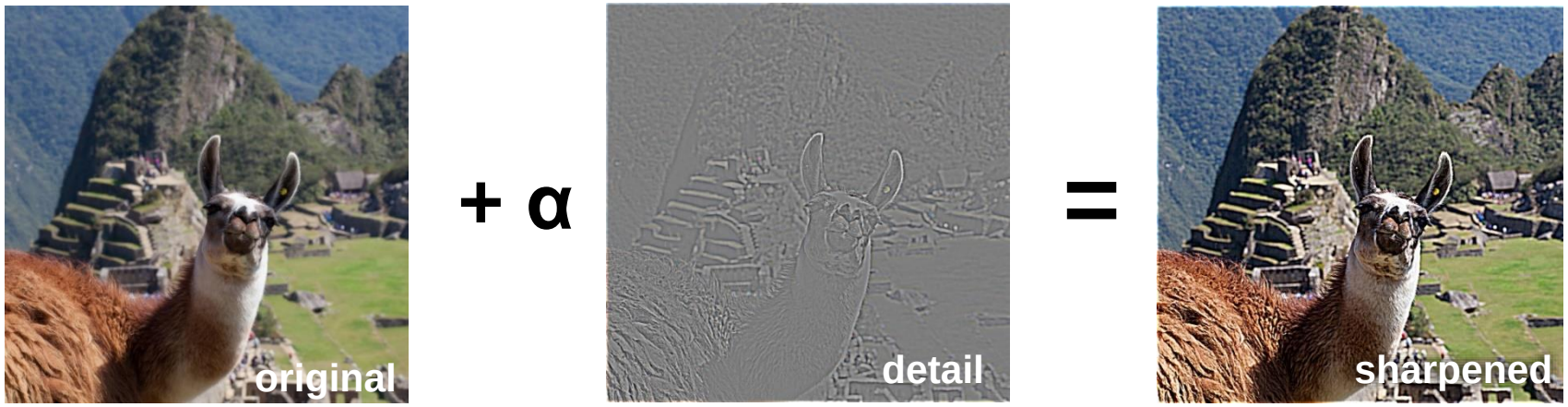


Photo credit: https://www.flickr.com/photos/geezaweezer/16089096376/

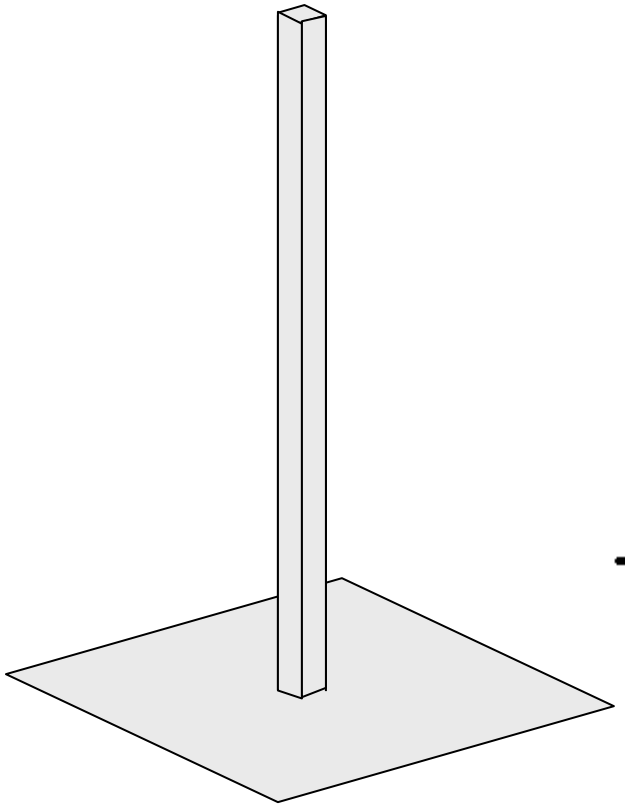
Unsharp Masking

$$F + \alpha\,(F - \underbrace{F * H}) = (1+\alpha)\,F - \alpha\,(F * H) = F * ([1+\alpha]\,e - \alpha H)$$

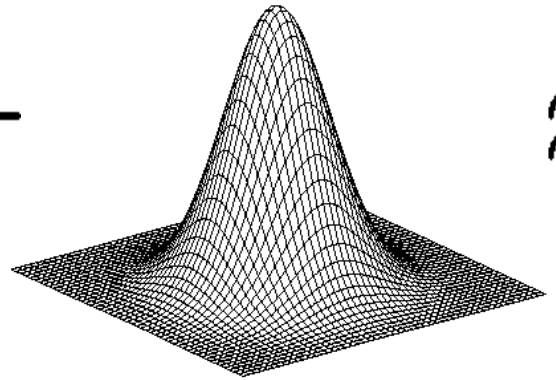
image

unsharp / blurred image

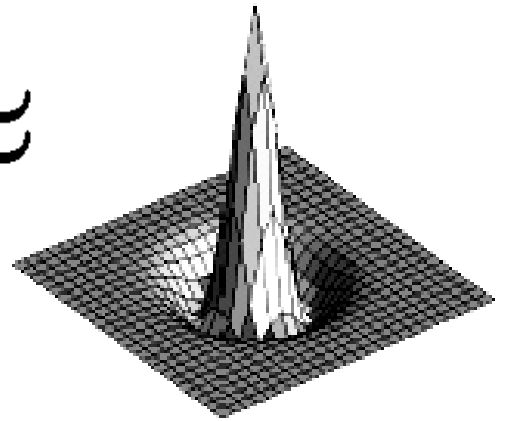
unit impulse (identity kernel with single 1 in center, zeros elsewhere)



scaled impulse



Gaussian



Sharpen filter

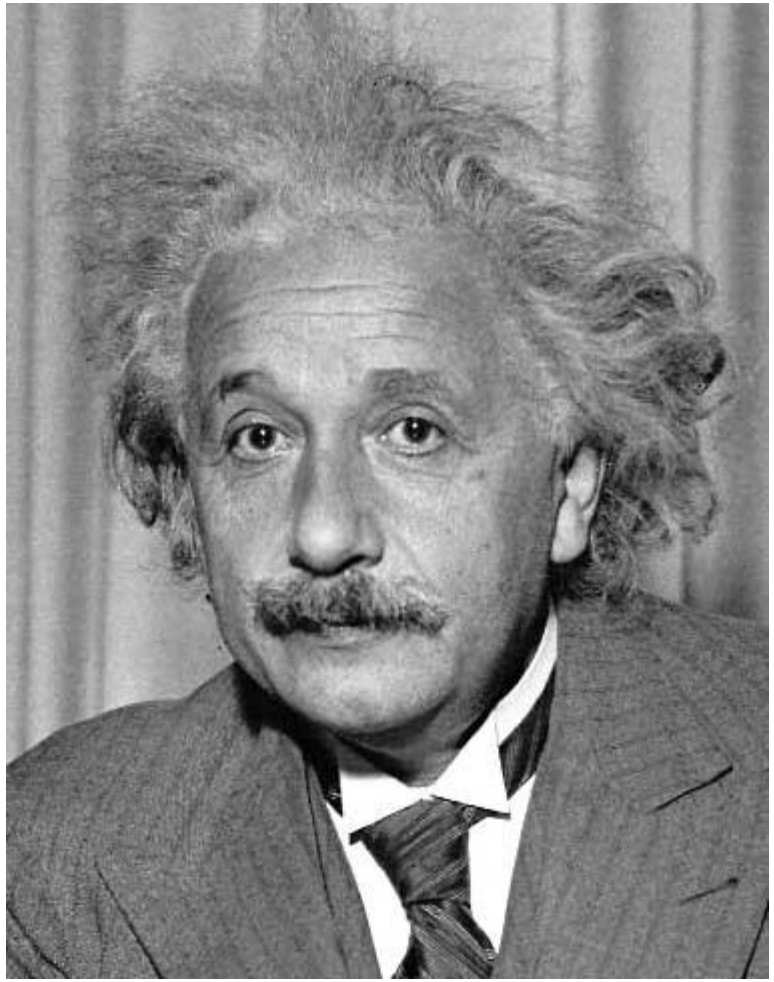
Take-home message:
sharpening is taking an image F:

1) blurring it,
2) subtracting the blurred image from F to get high-frequency edges,
3) then scaling and adding those edges back to F to yield an image with emphasized edges

Unsharp Masking

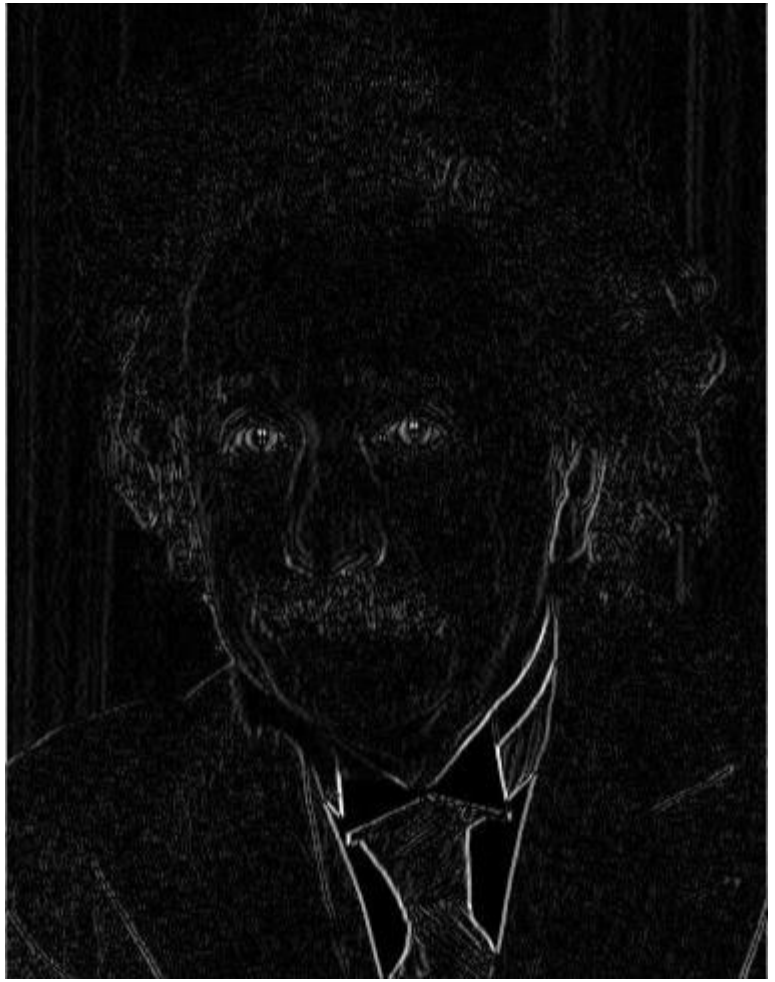
Edges are transformed so that the image gets darker on one side and brighter on the other – in this way, the edge is emphasized. **The sharpened image “anticipates” the edges**

Other filters



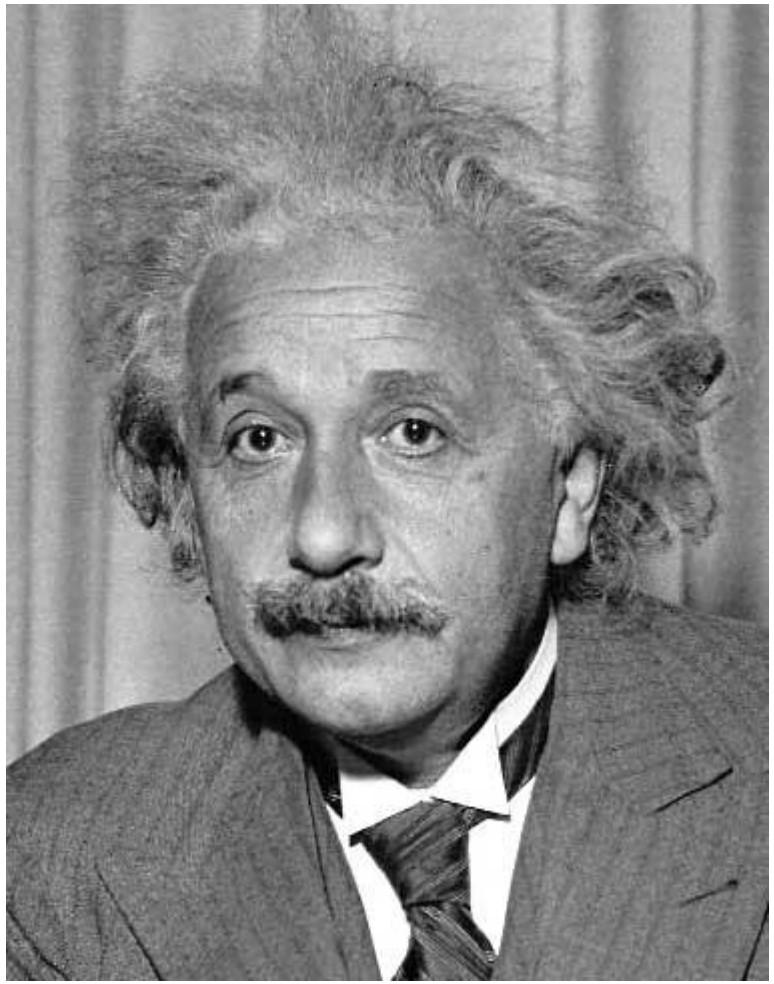
1	0	-1
2	0	-2
1	0	-1

Sobel



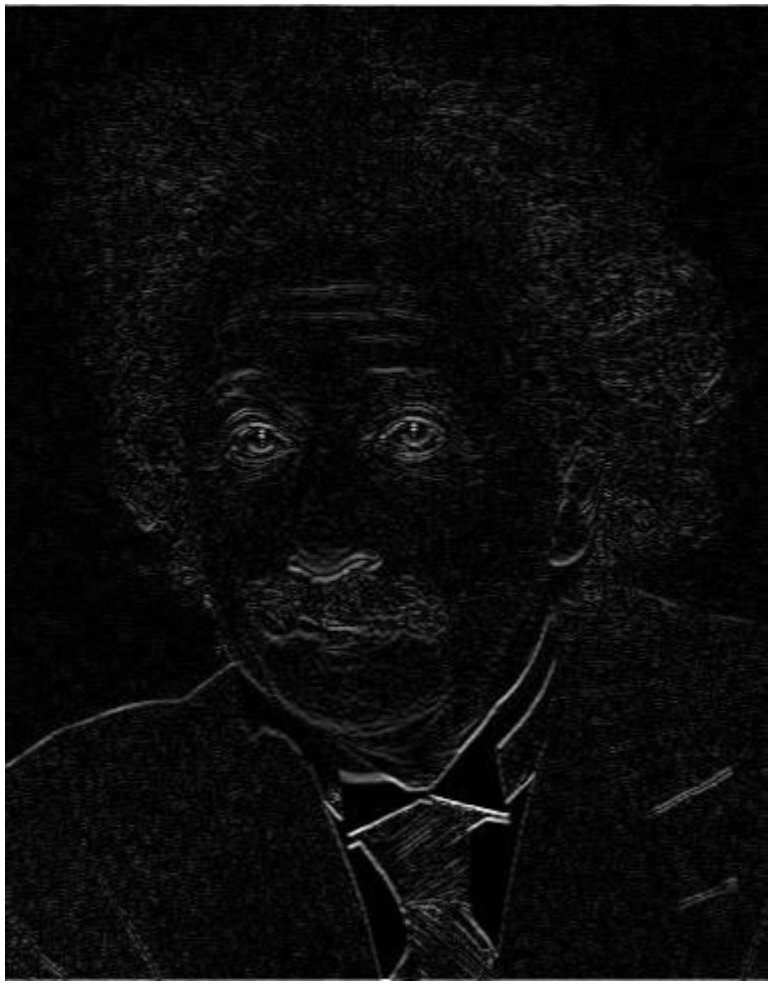
Vertical Edge (absolute value)

Other filters



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

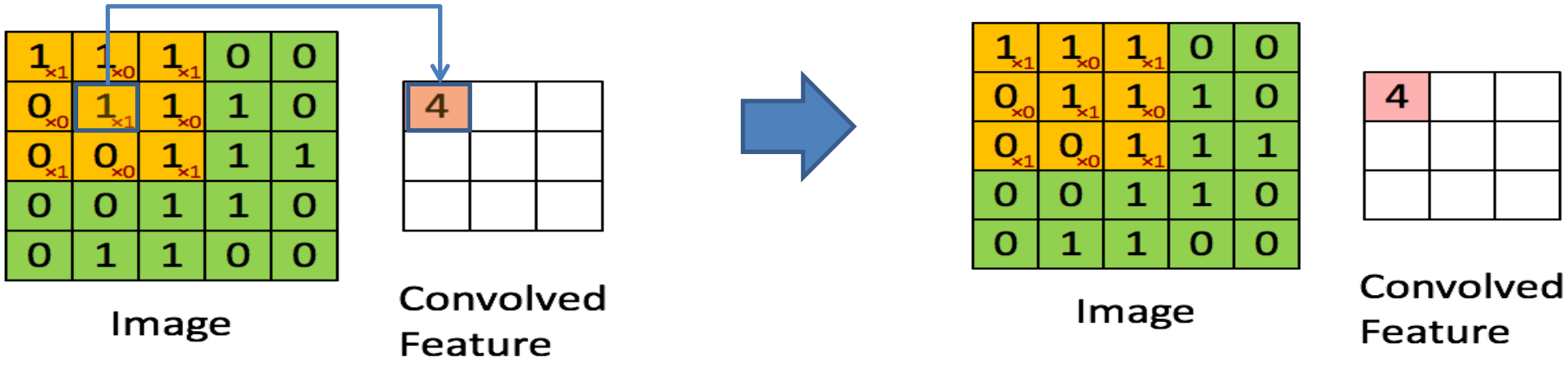
Sobel



Horizontal Edge (absolute value)

Image Filtering (key ideas so far)

- Convolution is a **local and linear operation**
  - Red numbers below represent the coefficients of the mask.
  - The filter is multiplied element-by-element with the image, the products are added, and the central position of the filter is replaced in the image.



https://glassboxmedicine.com/2019/07/26/convolution-vs-cross-correlation/

Image Filtering (key ideas so far)

- Convolution is a **local and linear operation**
  - The coefficients of the mask determine the result

Gaussian filter (removes high frequencies → image smoothing)

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Sobel filter (removes low frequencies → enhance edges/boundaries)

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

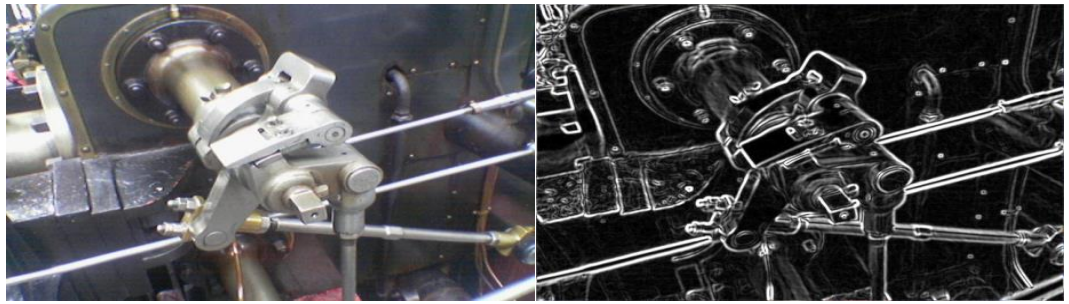
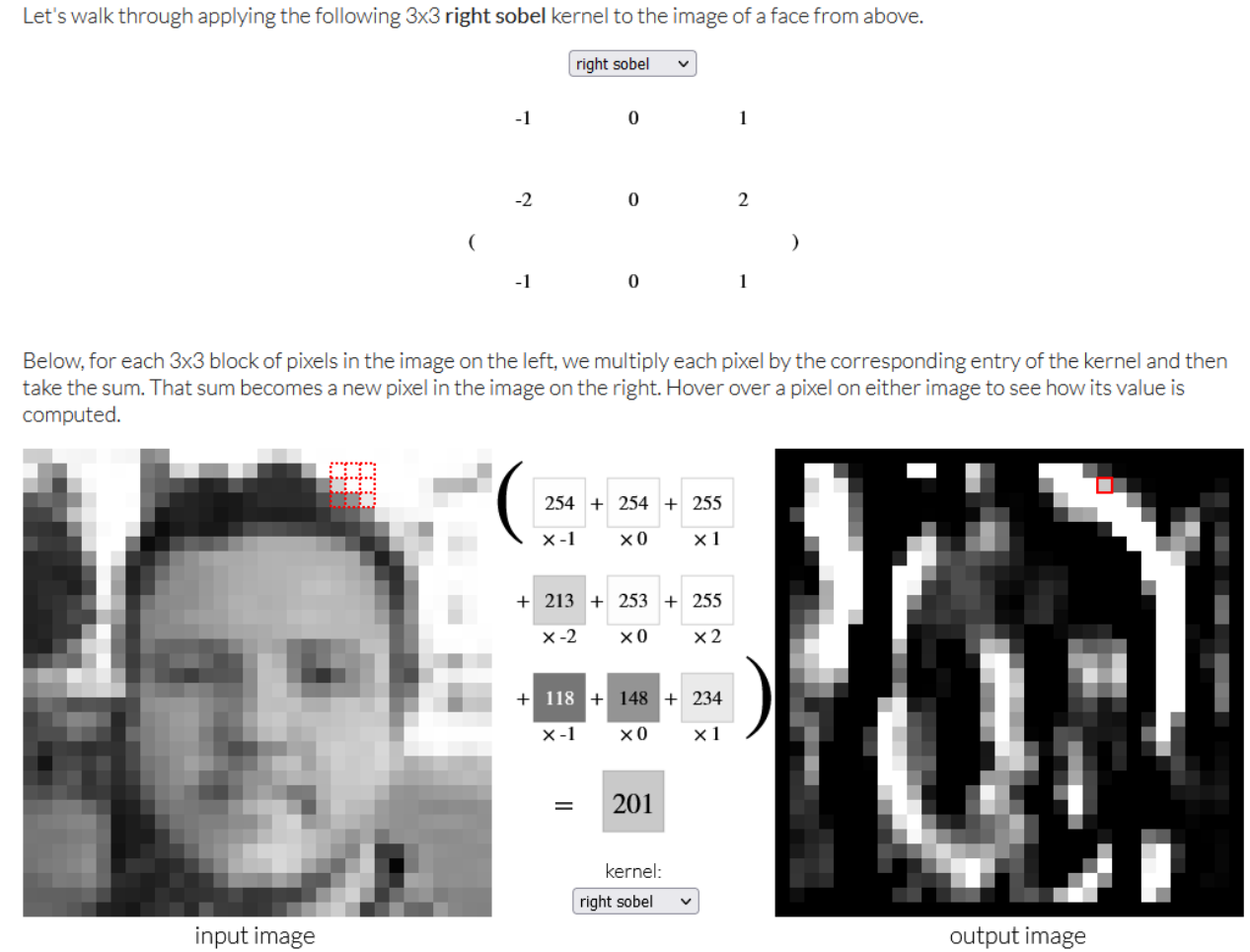


Image Filtering

- Image Kernels visually explained
  - https://setosa.io/ev/image-kernels/



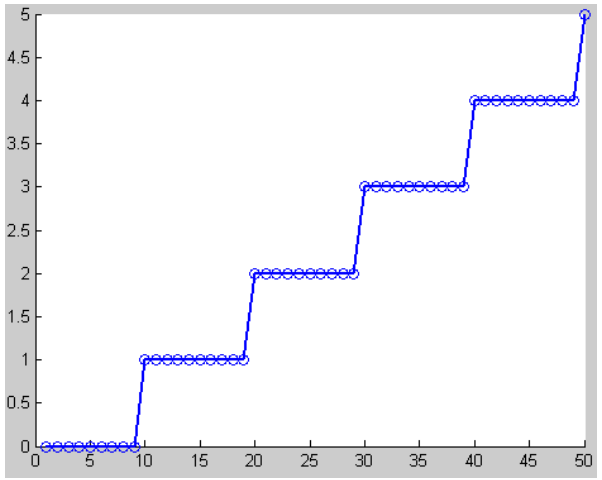
Nonlinear filters

Median filter: Replace pixel with median of surrounding *n x n* region (no new pixel values introduced)

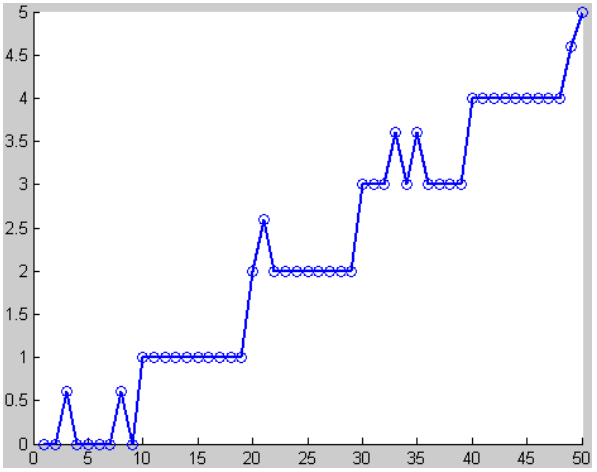
Good for impulse noise (salt-and-pepper noise)

What is mean filter good for?

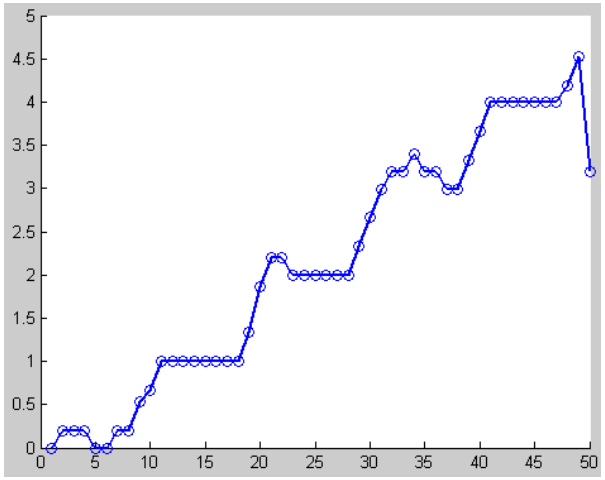
Additive Gaussian noise



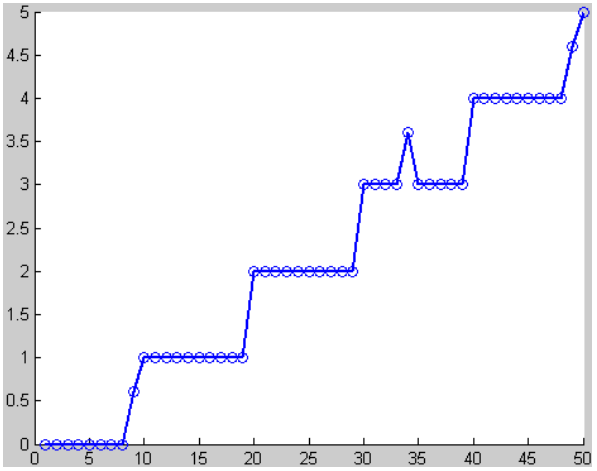
image



impulse noise



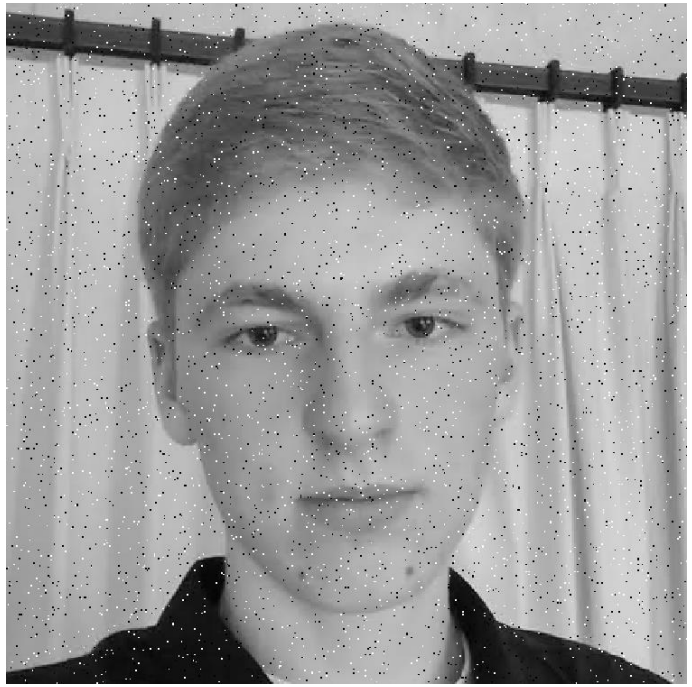
mean filtering



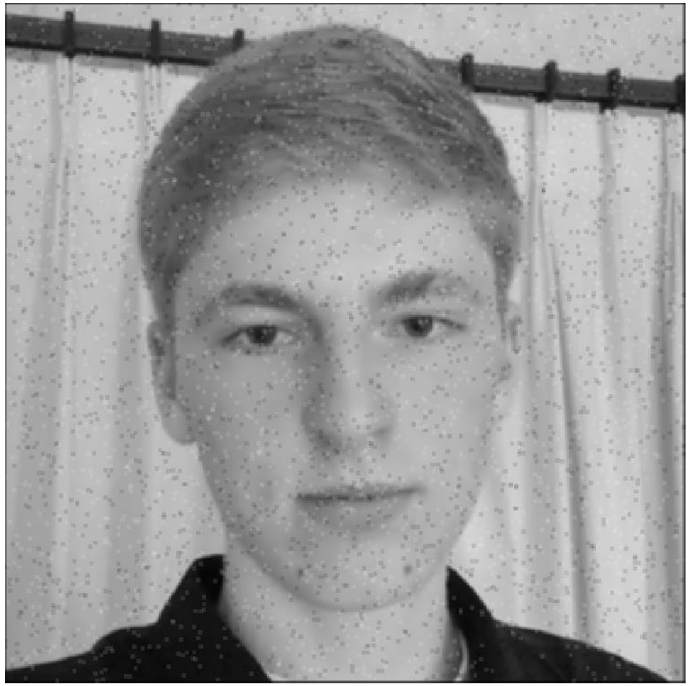
median filtering

Nonlinear filters

Image with salt-and-pepper noise



mean filtering (3x3)

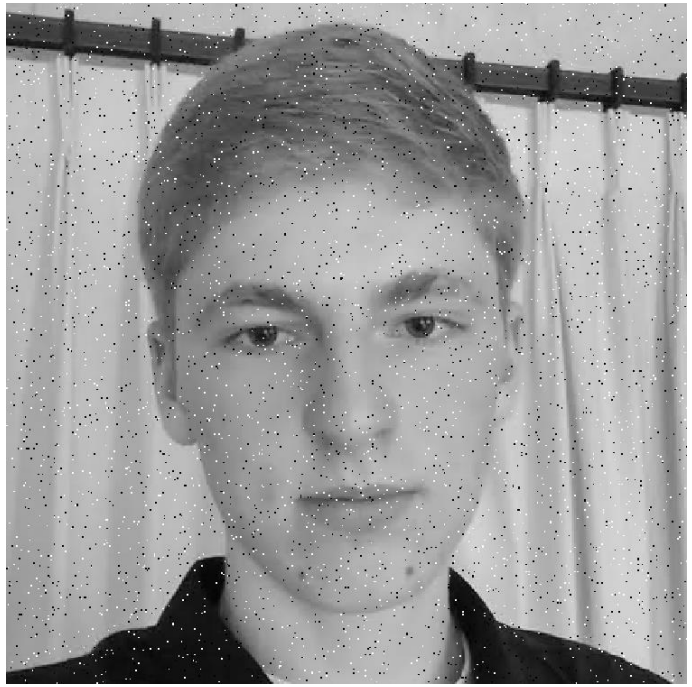


mean filtering (11x11)

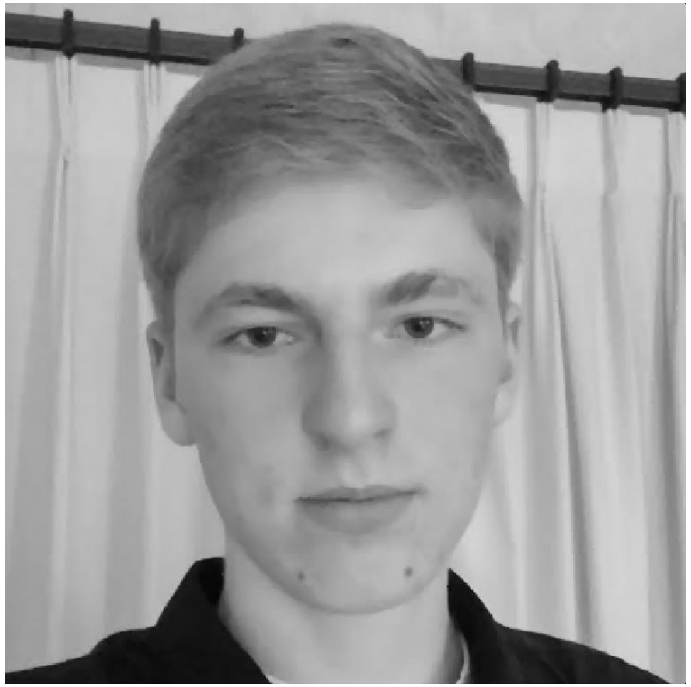
https://cs.brown.edu/courses/cs143/2017_Spring/lectures_Spring2017/04_Spring2017_ThinkingInFrequencyPart1.pptx

Nonlinear filters

Image with salt-and-pepper noise



median filtering (3x3)



median filtering (11x11)

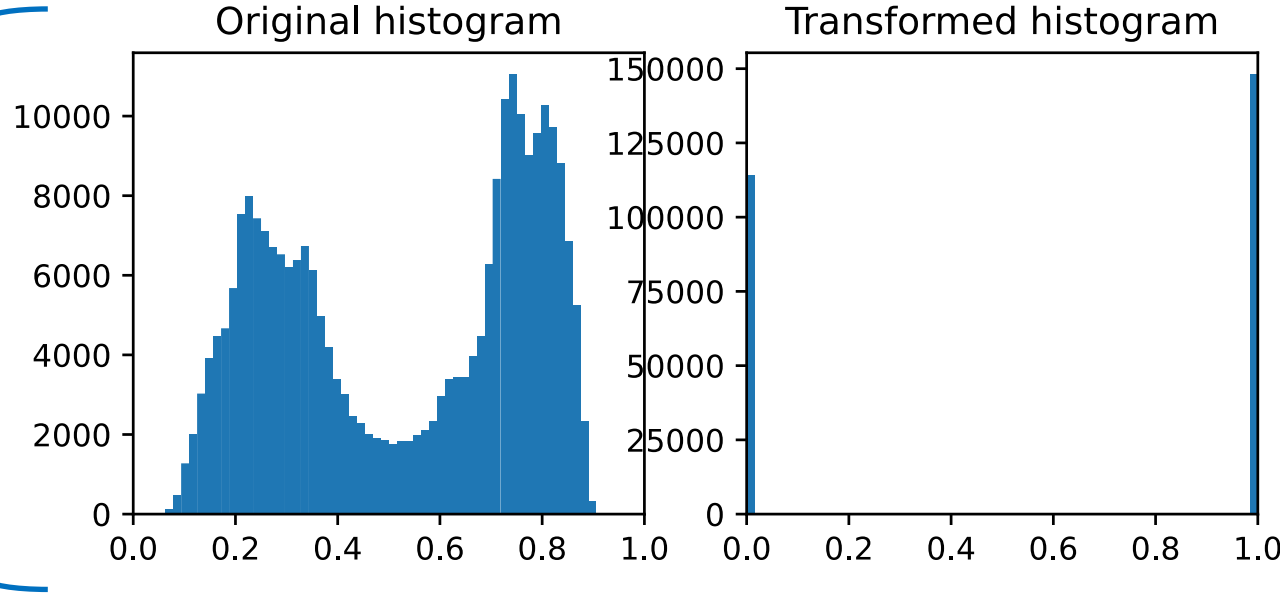
https://cs.brown.edu/courses/cs143/2017_Spring/lectures_Spring2017/04_Spring2017_ThinkingInFrequencyPart1.pptx

Thresholding

$$g(m,n) = \begin{cases} 255, & f(m,n) > A \\ 0 & otherwise \end{cases}$$

Thresholding

Image histogram (number of pixels with a particular gray-level value)



https://vincmazet.github.io/bip/histogram/transformations.html

How to select this threshold? We can apply techniques like Otsu's method (Otsu, Nobuyuki. "A threshold selection method from gray-level histograms." *IEEE transactions on systems, man, and cybernetics* 9.1 (1979): 62-66)

Otsu's Thresholding Method

We **iterate through all possible threshold values** and **measure the spread/variance** of background and foreground pixels. Then, **we select the threshold with the smallest spread/variance**. i.e. we want to minimize the within-class variance: $\sigma^2(t) = \omega_{bg}(t)\sigma^2_{bg}(t) + \omega_{fg}(t)\sigma^2_{fg}(t)$

Let's try with this image and threshold (t=100):

Foreground Pixels **Background Pixels**

$$P_{all} = 16$$
$$P_{BG} = 7$$
$$P_{FG} = 9$$

$$\omega_{bg}(t) = \frac{P_{BG}(t)}{P_{all}} = 7/16 = 0.44$$
$$\omega_{fg}(t) = \frac{P_{FG}(t)}{P_{all}} = 9/16 = 0.56$$

$$\overline{x_{bg}} = \frac{21+22+25+26+27+23+24}{7} = 24$$
$$\overline{x_{fg}} = \frac{120+120+160+180+190+123+145+165+175}{9} = 153.1$$

$$\sigma^2_{bg}(t=100) = \frac{(21-24)^2+(22-24)^2\ldots\ldots(24-24)^2}{7} = 4.0$$
$$\sigma^2_{fg}(t=100) = \frac{(120-153.1)^2+(120-153.1)^2\ldots\ldots(175-153.1)^2}{9} = 657.43$$

$$\sigma^2(t=100) = 0.44 * 4.0 + 0.56 * 657.43 = 369.9208$$

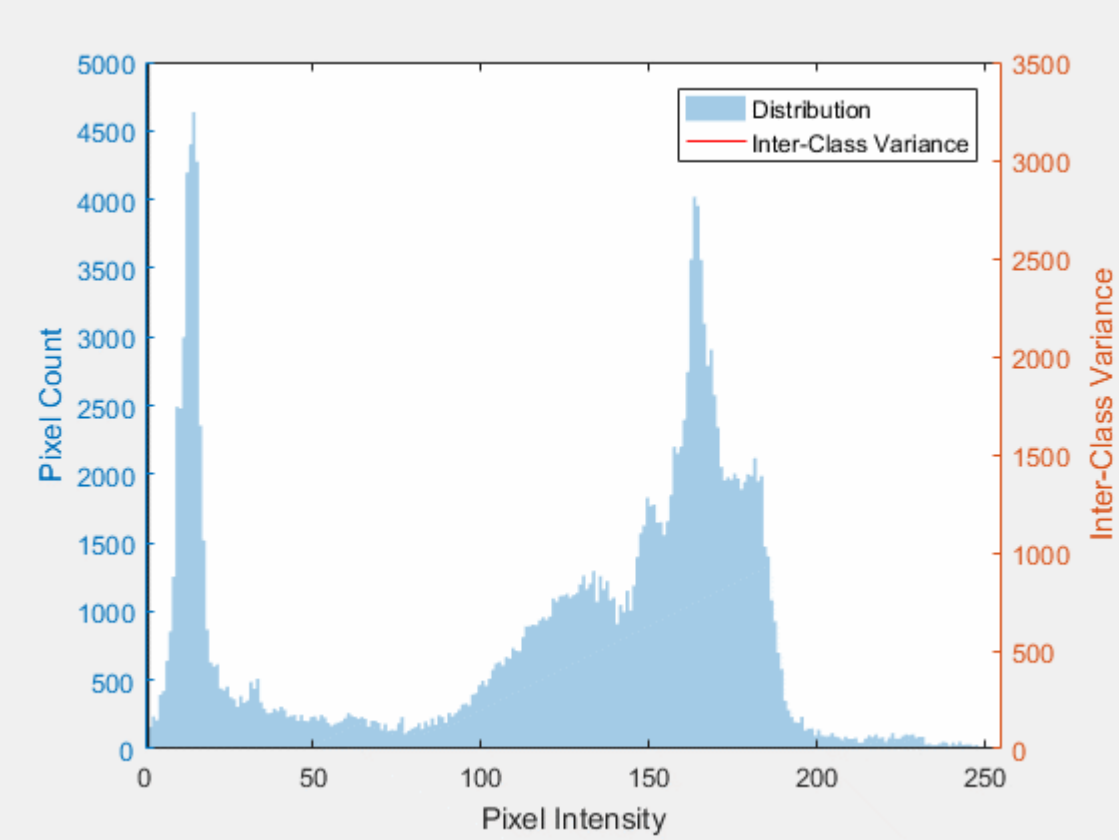
We keep the threshold corresponding to the smallest σ²

Otsu, Nobuyuki. "A threshold selection method from gray-level histograms." *IEEE transactions on systems, man, and cybernetics* 9.1 (1979): 62-66

https://muthu.co/otsus-method-for-image-thresholding-explained-and-implemented/

Otsu’s Thresholding Method

Minimizing the intra-class variance is equivalent to maximizing inter-class variance



https://en.wikipedia.org/wiki/Otsu%27s_method#/media/File:Otsu's_Method_Visualization.gif

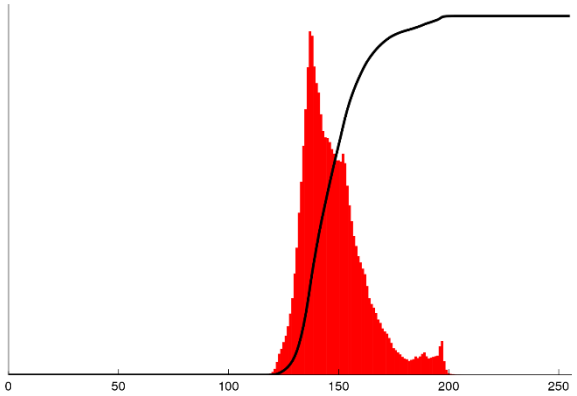
Otsu, Nobuyuki. "A threshold selection method from gray-level histograms."
IEEE transactions on systems, man, and cybernetics 9.1 (1979): 62-66

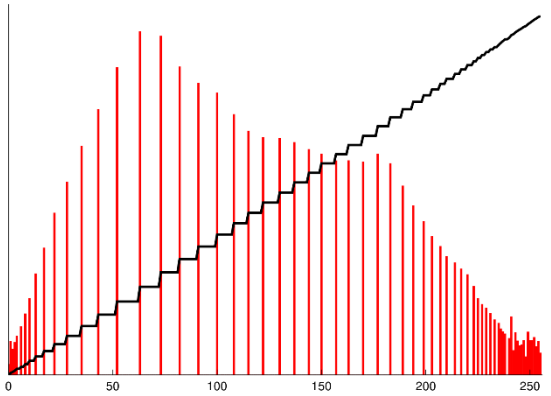
Histogram Equalization

We **increase the global contrast**, especially when the image is represented by a narrow range of intensity values.

Before

After





Histogram (red) and cumulative histogram (black)

Linear filters

- Can thresholding be implemented with a linear filter?
- Which one of the following 3x3 filters returns a positive value if the average value of the 4-adjacent neighbors is less than the center, and a negative value (or zero) otherwise?

(a)

0	¼	0
¼	-1	¼
0	¼	0

(b)

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

(c)

0	-1	0
-1	4	-1
0	-1	0

Image Representation and Filtering

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE GRANADA

DaSCI

Instituto Andaluz de Investigación en **Data Science** and **Computational Intelligence**