CAP 4453
Robot Vision

Dr. Gonzalo Vaca-Castaño

Gonzalo.vacacastano@ucf.edu

Administrative details

- Homework 1 issues ?

Outline

- ~~Image as a function~~
  - ~~Linear algebra~~
- Extracting useful information from Images
  - ~~Histogram~~
  - Noise
  - Filtering (linear)
  - Smoothing/Removing noise
  - Convolution/Correlation
  - Image Derivatives/Gradient
  - Edges
- Colab Notes/ homeworks
- Read Szeliski, Chapter 3.
- Read/Program CV with Python, Chapter 1.

Image noise

- Light Variations

- Camera Electronics

- Surface Reflectance

- Lens

- Noise is random,
  - it occurs with some probability
  - It has a distribution

Additive Noise

$$I_{observed}(x, y) = I_{original}(x, y) + n(x, y)$$

True pixel value at x,y

Noise at x,y

Multiplicative Noise

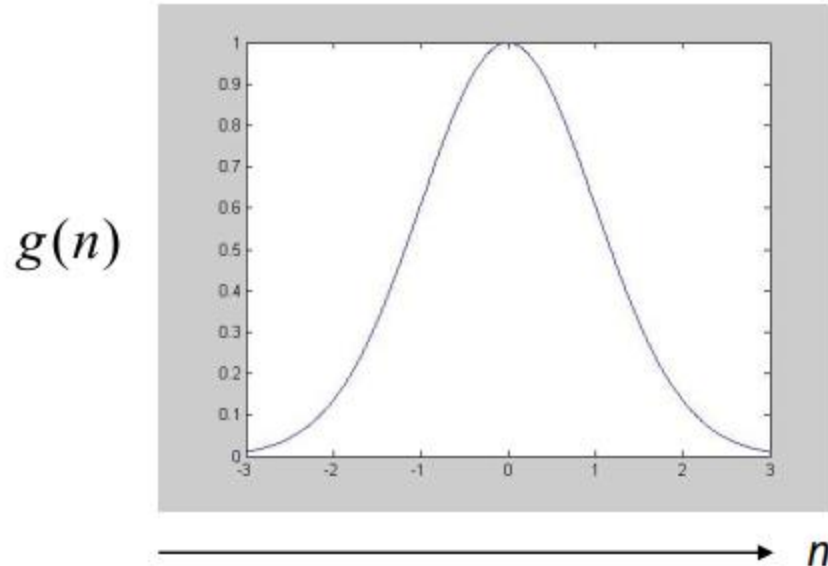$$I_{observed}(x, y) = I_{original}(x, y) \times n(x, y)$$

True pixel value at x,y

Noise at x,y

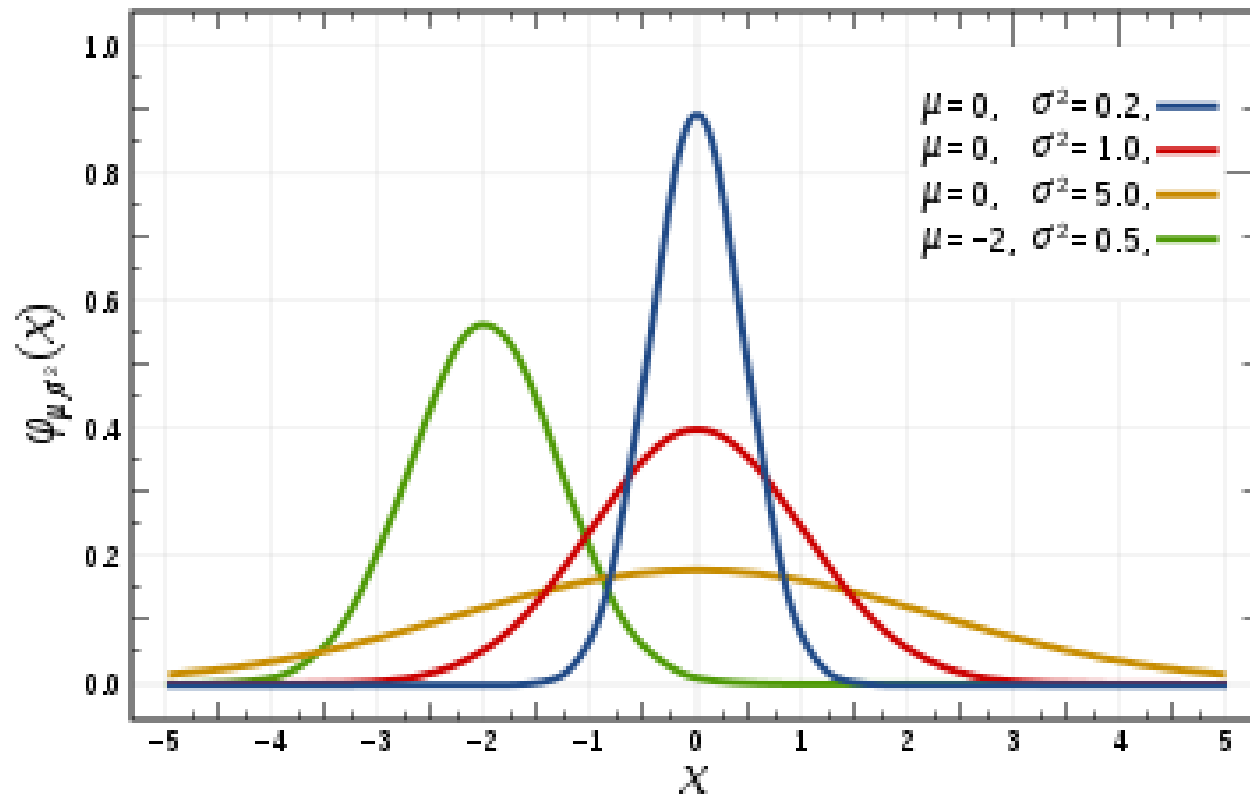# Gaussian Noise

$$n(x,y) \approx g(n) = e^{\frac{-n^2}{2\sigma^2}}$$

$g(n)$

Probability Distribution
n is a random variable

Gaussian function

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2} \right).$$
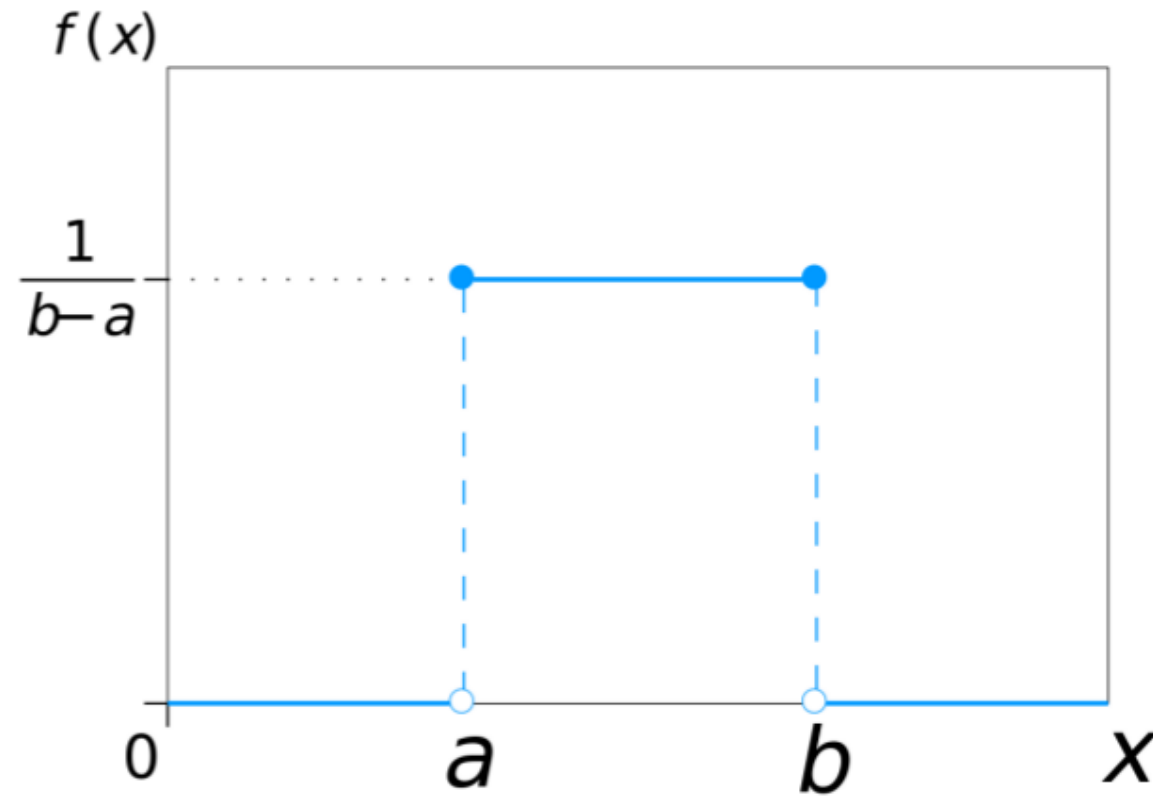
Salt and pepper noise

- Each pixel is randomly made black or white with a uniform probability distribution

Salt-pepper

Uniform distribution

Noise implementation

```python
#Parameters
#----------
#image : ndarray
#    Input image data. Will be converted to float.
#mode : str
#    One of the following strings, selecting the type of noise to add:

#    'gauss'     Gaussian-distributed additive noise.
#    'poisson'   Poisson-distributed noise generated from the data.
#    's&p'       Replaces random pixels with 0 or 1.
#    'speckle'   Multiplicative noise using out = image + n*image,where
#                n,is uniform noise with specified mean & variance.
import numpy as np
import os
import cv2

def noisy(noise_typ,image):
    if noise_typ == "gauss":
        row,col,ch= image.shape
        mean = 0
        var = 1
        sigma = var**0.5
        gauss = np.random.normal(mean,sigma,(row,col,ch))
        gauss = gauss.reshape(row,col,ch)
        noisy = image + gauss
        return noisy
    elif noise_typ == "s&p":
        row,col,ch = image.shape
        s_vs_p = 0.5
        amount = 0.004
        out = image
        # Salt mode
        num_salt = np.ceil(amount * image.size * s_vs_p)
        coords = [np.random.randint(0, i - 1, int(num_salt))
                    for i in image.shape]
        out[coords] = 1

        # Pepper mode
        num_pepper = np.ceil(amount* image.size * (1. - s_vs_p))
        coords = [np.random.randint(0, i - 1, int(num_pepper))
                    for i in image.shape]
        out[coords] = 0
        return out
    elif noise_typ == "poisson":
        vals = len(np.unique(image))
        vals = 2 ** np.ceil(np.log2(vals))
        noisy = np.random.poisson(image * vals) / float(vals)
        return noisy
    elif noise_typ =="speckle":
        row,col,ch = image.shape
        gauss = np.random.randn(row,col,ch)
        gauss = gauss.reshape(row,col,ch)
        noisy = image + image * gauss
        return noisy
```

Outline

- ~~Image as a function~~
- Extracting useful information from Images
  - ~~Histogram~~
  - ~~Noise~~
  - **Filtering (linear)**
  - Smoothing/Removing noise
  - Convolution/Correlation
  - Image Derivatives/Gradient
  - Edges
- Colab Notes/ homeworks
- Read Szeliski, Chapter 3.
- Read/Program CV with Python, Chapter 1.

Filters

- Filtering
  - Form a new image whose pixels are a combination of the original pixels
- Why?
  - To get useful information from images
    - E.g., extract edges or contours (to understand shape)
  - To enhance the image
    - E.g., to remove noise
    - E.g., to sharpen and "enhance image" a la CSI
  - A key operator in Convolutional Neural Networks

Linear shift-invariant image filtering

- Replace each pixel by a *linear* combination of its neighbors (and possibly itself).

- The combination is determined by the filter's *kernel*.

- The same kernel is *shifted* to all pixel locations so that all pixels use the same linear combination of their neighbors.

Filtering

- Modify pixels based on some function of neighborhood
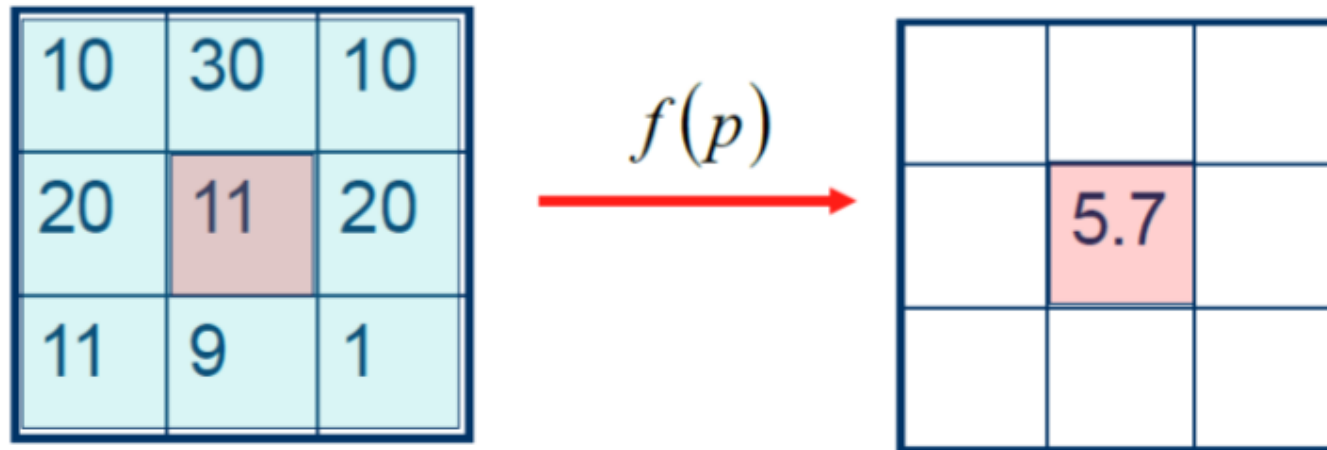
10	30	10
20	11	20
11	9	1

$f(p)$

→

	5.7	

Image filtering

- Image filtering: compute function of local neighborhood at each position

(kernel)

h=output   f=filter   I=image

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

2d coords=k,l   2d coords=m,n

[   ]   [ ]   [   ]

Image filtering

- Image filtering: compute function of local neighborhood at each position

- Enhance images
  - Denoise, resize, increase contrast, etc.
- Extract information from images
  - Texture, edges, distinctive points, etc.
- Detect patterns
  - Template matching

Let's run the box filter

Box filter

$g[\cdot,\cdot]$

kernel

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

note that we assume that the kernel coordinates are centered

image $f[\cdot,\cdot]$

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	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	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

output $h[\cdot,\cdot]$

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

output   filter   image (signal)

Let's run the box filter

$$g[\cdot,\cdot]$$

kernel

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

image $f[\cdot,\cdot]$

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	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	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

output $h[\cdot,\cdot]$

0

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

output   filter   image (signal)

Let's run the box filter

$g[\cdot,\cdot]$

kernel

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

image $f[\cdot,\cdot]$

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	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	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

output $h[\cdot,\cdot]$

		0							

shift-invariant: as the pixel shifts, so does the kernel

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

output   filter   image (signal)

Let's run the box filter

$g[\cdot,\cdot]$

kernel

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

image $f[\cdot,\cdot]$

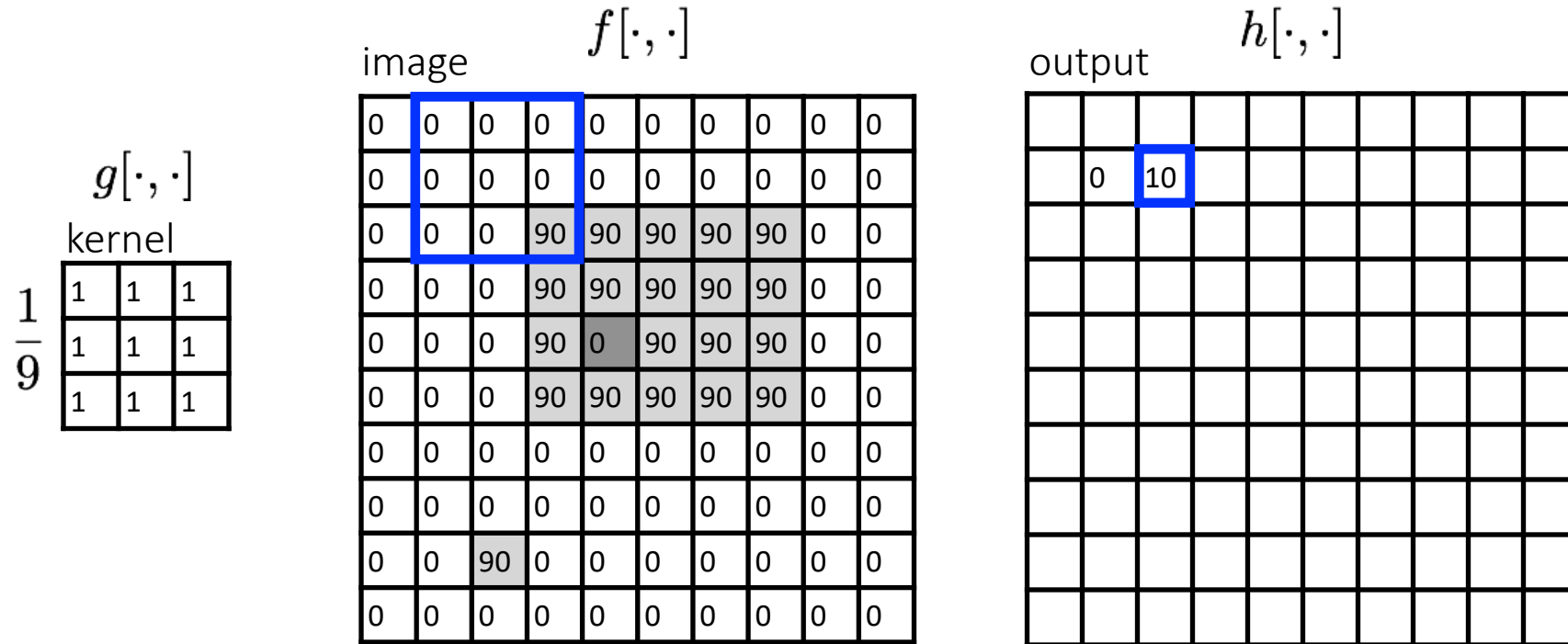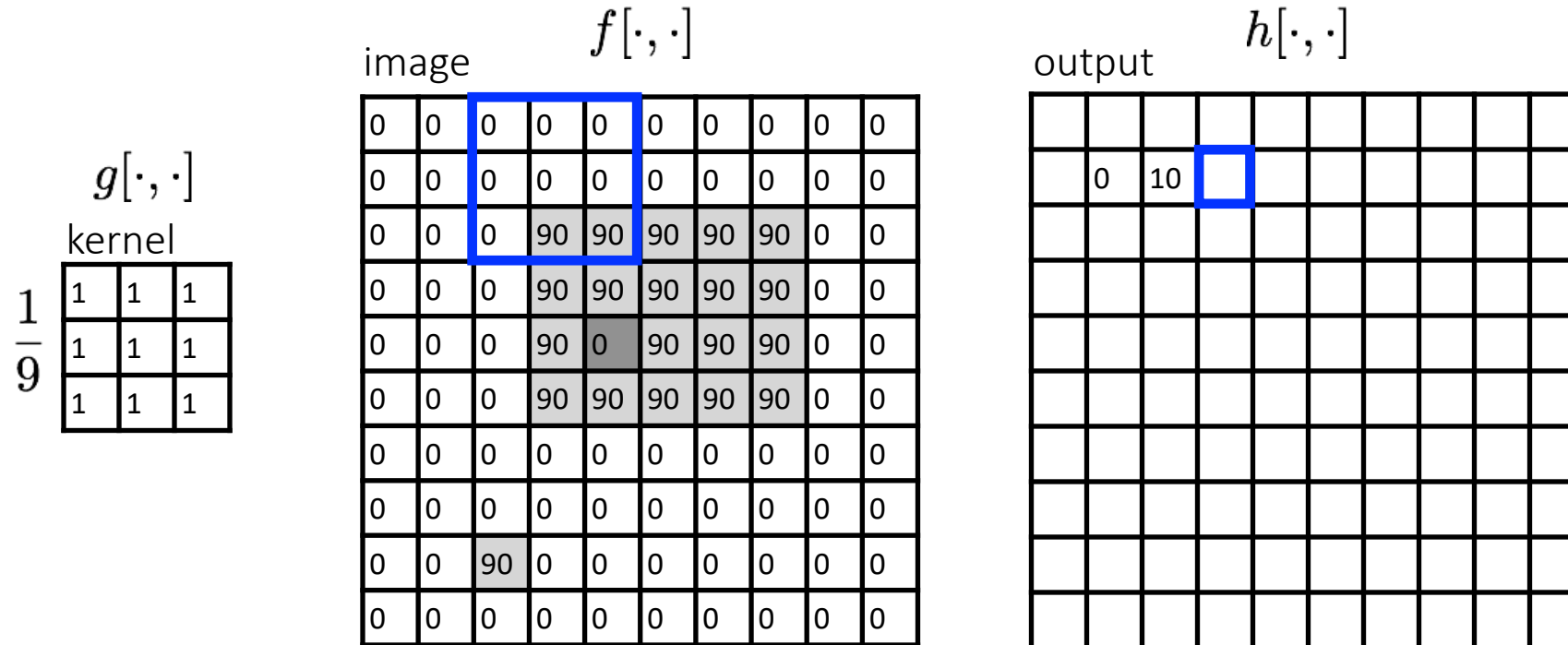| 0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $h[\cdot,\cdot]$

	0	10							

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

output   filter   image (signal)

Let's run the box filter

$g[\cdot,\cdot]$

kernel

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

image $f[\cdot,\cdot]$

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	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	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

output $h[\cdot,\cdot]$

		0	10						

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

output   filter   image (signal)

Let's run the box filter

$g[\cdot,\cdot]$

kernel

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

image $f[\cdot,\cdot]$

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	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	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

output $h[\cdot,\cdot]$

	0	10	20						

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

output   filter   image (signal)

Let's run the box filter

$f[\cdot,\cdot]$

image

$h[\cdot,\cdot]$

output

$g[\cdot,\cdot]$

kernel

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

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	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	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

Output: 0, 10, 20

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

output   filter   image (signal)

Let's run the box filter

$g[\cdot,\cdot]$

kernel

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

image $f[\cdot,\cdot]$

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	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	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

output $h[\cdot,\cdot]$

	0	10	20	30					

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

output   filter   image (signal)

Let's run the box filter

$g[\cdot,\cdot]$

kernel

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

image $f[\cdot,\cdot]$

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	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	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

output $h[\cdot,\cdot]$

	0	10	20	30	30				

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

output $\quad$ filter $\quad$ image (signal)

Let's run the box filter

$$g[\cdot,\cdot]$$

kernel

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

image $$f[\cdot,\cdot]$$

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	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	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

output $$h[\cdot,\cdot]$$

	0	10	20	30	30	30			

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

output   filter   image (signal)

Let's run the box filter

$g[\cdot,\cdot]$

kernel

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

image $f[\cdot,\cdot]$

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	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	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

output $h[\cdot,\cdot]$

	0	10	20	30	30	30	20		

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

output   filter   image (signal)

Let's run the box filter

$$f[\cdot,\cdot]$$

image

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	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	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

$$h[\cdot,\cdot]$$

output

	0	10	20	30	30	30	20	10	

$$g[\cdot,\cdot]$$

kernel

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

1	1	1
1	1	1
1	1	1

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

output   filter   image (signal)

Let's run the box filter

$f[\cdot,\cdot]$

image

$h[\cdot,\cdot]$

output

$g[\cdot,\cdot]$

kernel

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

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	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	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

		0	10	20	30	30	30	20	10	
		0								

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

output   filter   image (signal)

Let's run the box filter

$g[\cdot,\cdot]$

kernel

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

image

$f[\cdot,\cdot]$

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	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	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

output

$h[\cdot,\cdot]$

	0	10	20	30	30	30	20	10	
	0	20							

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

output   filter   image (signal)

Let's run the box filter

$g[\cdot,\cdot]$

kernel

$\dfrac{1}{9}$

1	1	1
1	1	1
1	1	1

image $f[\cdot,\cdot]$

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	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	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

output $h[\cdot,\cdot]$

	0	10	20	30	30	30	20	10	
	0	20	40						

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

output   filter   image (signal)

Let's run the box filter

$f[\cdot,\cdot]$

image

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	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	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

$h[\cdot,\cdot]$

output

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0								

$g[\cdot,\cdot]$

kernel

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

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

output   filter   image (signal)

Let's run the box filter

$f[\cdot,\cdot]$

image

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	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	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

$g[\cdot,\cdot]$

kernel

$\dfrac{1}{9}$

1	1	1
1	1	1
1	1	1

$h[\cdot,\cdot]$

output

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30							

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

output   filter   image (signal)

Let's run the box filter

$g[\cdot,\cdot]$

kernel

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

1	1	1
1	1	1
1	1	1

image $f[\cdot,\cdot]$

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	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	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

output $h[\cdot,\cdot]$

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

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

output   filter   image (signal)

Let's run the box filter

$g[\cdot,\cdot]$

kernel

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

$f[\cdot,\cdot]$

image

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	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	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

$h[\cdot,\cdot]$

output

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

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

output   filter   image (signal)

… and the result is

image $f[\cdot,\cdot]$ output $h[\cdot,\cdot]$

$g[\cdot,\cdot]$

kernel

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

1	1	1
1	1	1
1	1	1

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	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	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

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

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

output   filter   image (signal)

Correlation (linear relationship)

$$f \otimes h = \sum_k \sum_l f(k,l) h(k,l)$$

f = Image

h = Kernel

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$$f \otimes h = f_1 h_1 + f_2 h_2 + f_3 h_3$$
$$+ f_4 h_4 + f_5 h_5 + f_6 h_6$$
$$+ f_7 h_7 + f_8 h_8 + f_9 h_9$$

Convolution

$$f * h = \sum_{k}\sum_{l} f(k,l) h(-k,-l)$$

f = Image

h = Kernel

h

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$X - flip$

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$Y - flip$

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

*

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

$$f * h = f_1 h_9 + f_2 h_8 + f_3 h_7$$
$$+ f_4 h_6 + f_5 h_5 + f_6 h_4$$
$$+ f_7 h_3 + f_8 h_2 + f_9 h_1$$
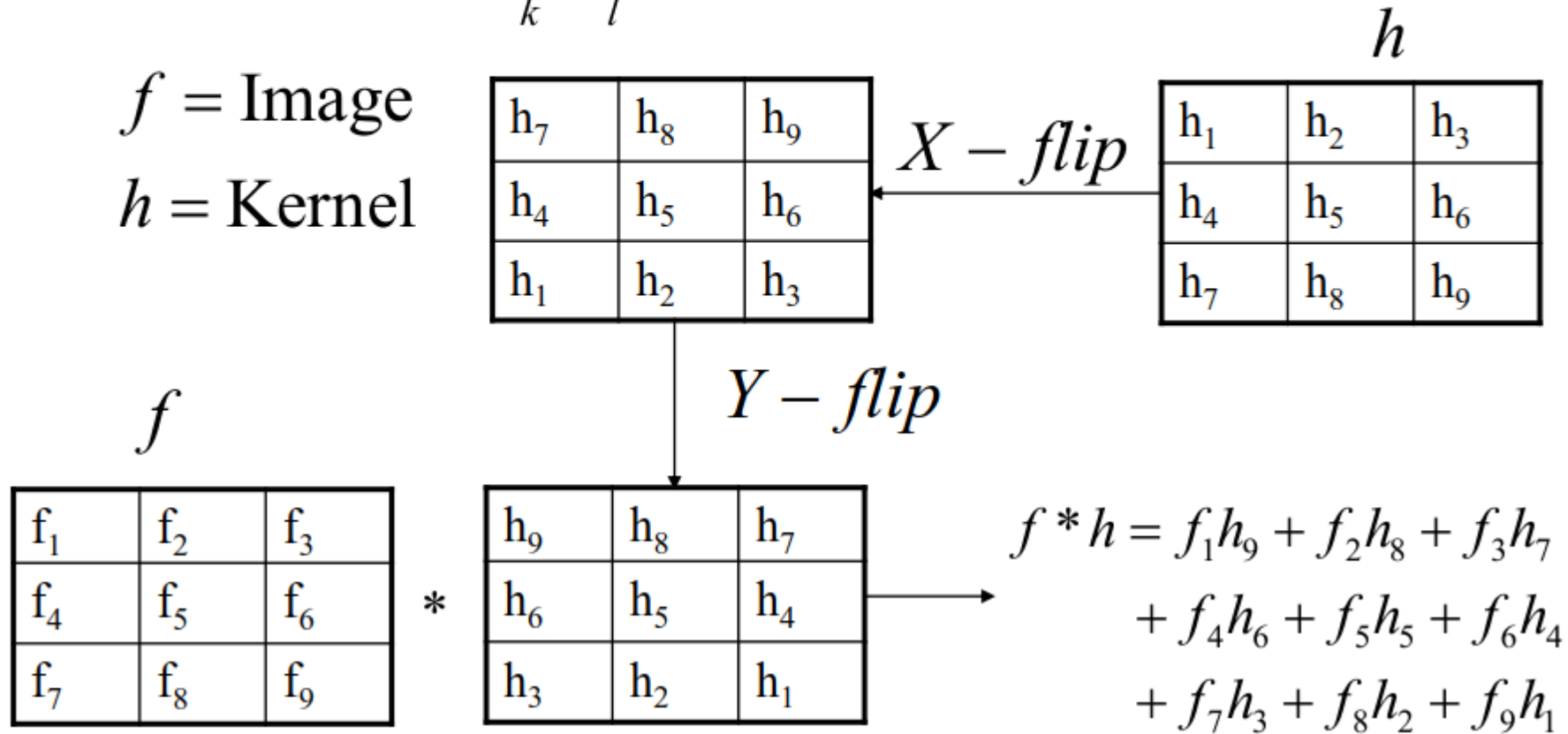
Correlation and Convolution

- **Convolution** is a filtering operation
  - expresses the amount of overlap of one function as it is shifted over another function

- **Correlation** compares the similarity of two sets of data
  - relatedness of the signals!

Key properties of linear filters

Linearity:

$$\text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2)$$

Shift invariance: same behavior regardless of pixel location

$$\text{filter}(\text{shift}(f)) = \text{shift}(\text{filter}(f))$$

Any linear, shift-invariant operator can be represented as a convolution

Source: S. Lazebnik

More properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality

- 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 $e = [0, 0, 1, 0, 0]$,
  $a * e = a$

Filtering Examples - 1
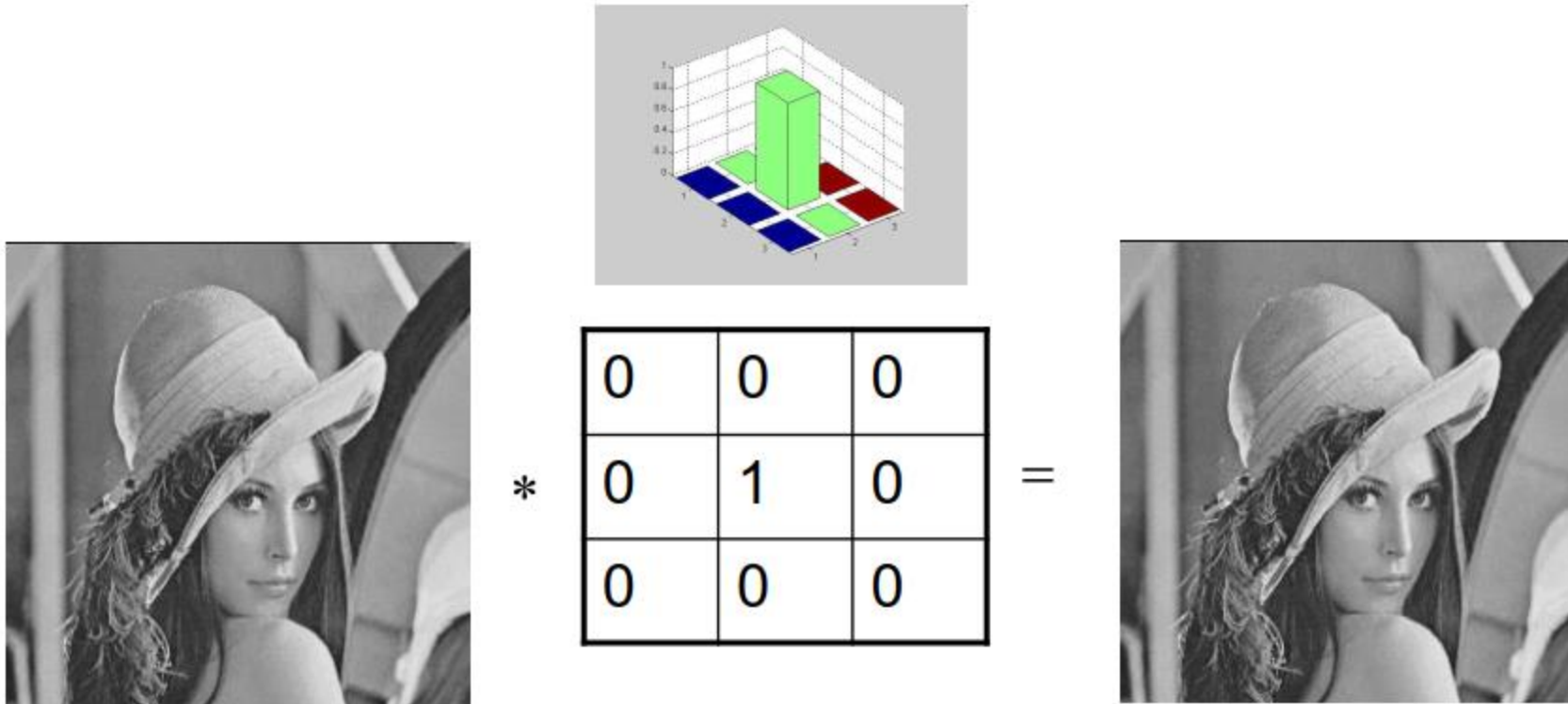
Filtering Examples - 2
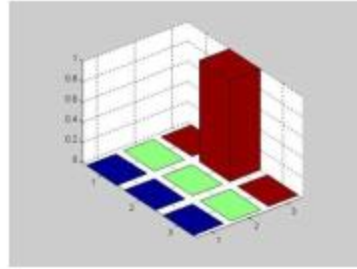
0	0	0
1	0	0
0	0	0

* ... =

Filtering Examples - 2

$$* \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} =$$

Example: box filter

Average: mean
• Dividing the sum of N values by N

What does it do?

- Replaces each pixel with an average of its neighborhood

$$g[\cdot,\cdot]$$

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

Filtering Examples - 3

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

Filtering Examples - 3

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

1	1	1
1	1	1
1	1	1

$=$
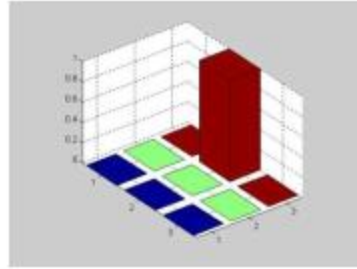
Example: box filter

What does it do?

- Replaces each pixel with an average of its neighborhood

- Achieve smoothing effect (remove sharp features)

$g[\cdot,\cdot]$

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

1	1	1
1	1	1
1	1	1

Filtering Examples - 4
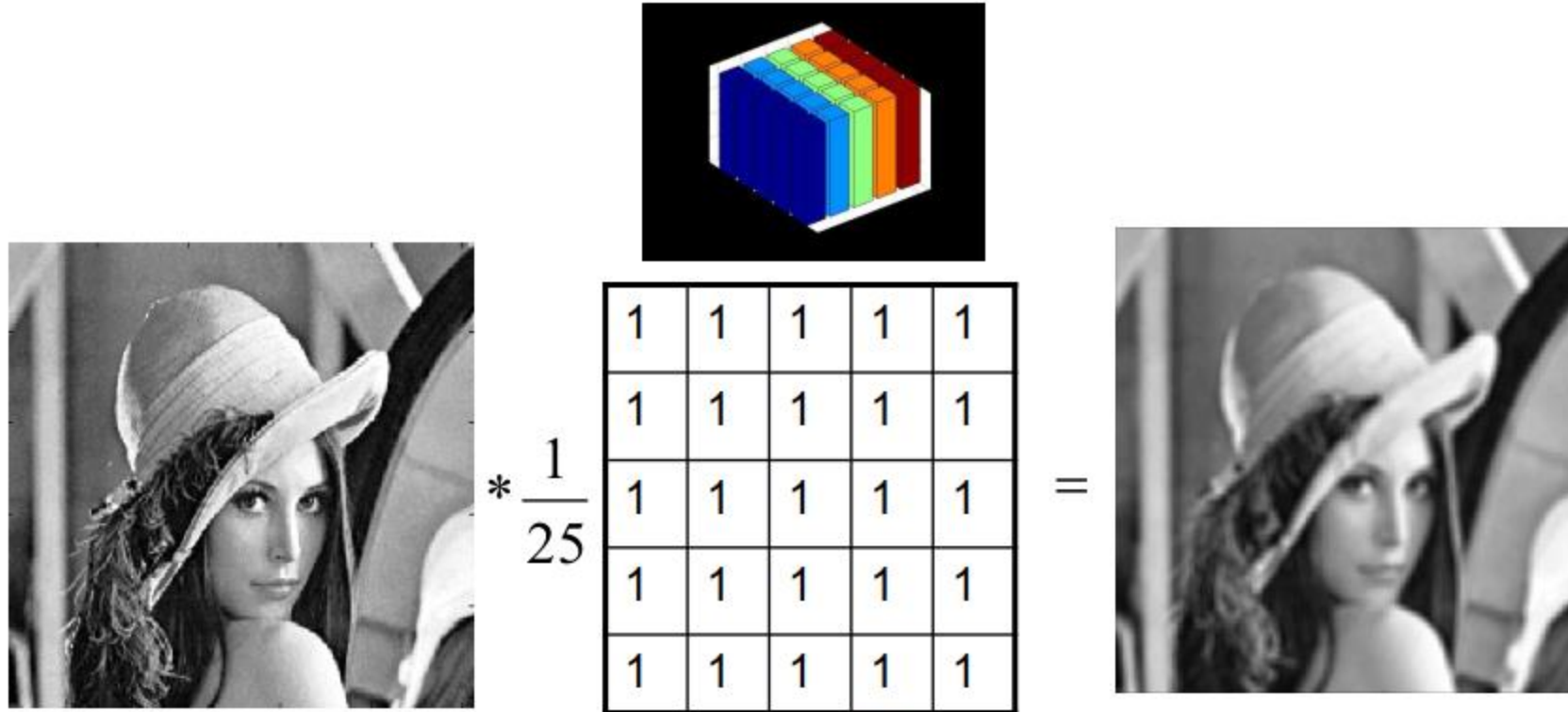
$$* \frac{1}{25} \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline \end{array} =$$

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example:
box filter

$$\begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline \end{array}$$

column   row

What is the rank of this filter matrix?

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example:
box filter

1	1	1
1	1	1
1	1	1

=

1
1
1

column

*

1	1	1

row
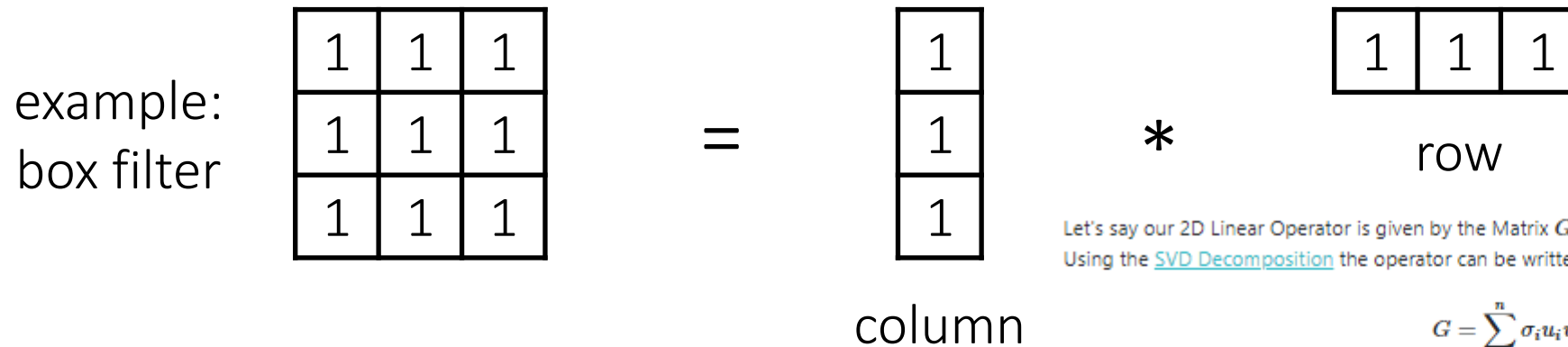
Let's say our 2D Linear Operator is given by the Matrix $G \in \mathbb{R}^{n \times n}$.
Using the SVD Decomposition the operator can be written as:

$$G = \sum_{i=1}^{n} \sigma_i u_i v_i^T$$

Separable Linear 2D Operator is defined as operator which can be composed by Outer Product of 2 vectors.
Looking at the SVD Decomposition of G we can conclude that G is separable operator if and only if $\forall i > 1 \ \sigma_i = 0$ and it is given by:

$$G = \sigma_1 u_1 v_1^T$$

Usually LPF 2D Linear Operators, such as the Gaussian Filter, in the Image Processing world are normalized to have sum of 1 (Keep DC) which suggests $\sigma_1 = 1$ moreover, they are also symmetric and hence $u_1 = v_1$ (If you want, in those cases, it means you can use the Eigen Value Decomposition instead of the SVD).

So basically, to prove that a Linear 2D Operator is Separable you must show that it has only 1 non vanishing singular value.

What is the rank of this filter matrix?

Matrix rank is 1 for separable filters

s = svd(G);
sum(s > eps('single'))

image processing - How to Prove a 2D Filter Is Separable? - Signal Processing Stack Exchange

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example:
box filter

1	1	1
1	1	1
1	1	1

=

1
1
1

column

*

1	1	1

row

Why is this important?

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example:
box filter

1	1	1
1	1	1
1	1	1

=

1
1
1

column

*

1	1	1

row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the "column" and "row" filters).

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example:
box filter

$$
\begin{array}{|c|c|c|}
\hline
1 & 1 & 1 \\
\hline
1 & 1 & 1 \\
\hline
1 & 1 & 1 \\
\hline
\end{array}
\;=\;
\begin{array}{|c|}
\hline
1 \\
\hline
1 \\
\hline
1 \\
\hline
\end{array}
\;*\;
\begin{array}{|c|c|c|}
\hline
1 & 1 & 1 \\
\hline
\end{array}
$$

column   row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the "column" and "row" filters).

If the image has M x M pixels and the filter kernel has size N x N:
• What is the cost of convolution with a non-separable filter?

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example:
box filter

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

row

column

2D convolution with a separable filter is equivalent to two 1D convolutions (with the "column" and "row" filters).

If the image has M x M pixels and the filter kernel has size N x N:
- What is the cost of convolution with a non-separable filter? $\longrightarrow$ $M^2 \times N^2$
- What is the cost of convolution with a separable filter?

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example:
box filter

1	1	1
1	1	1
1	1	1

=

1
1
1

column

*

1	1	1

row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the "column" and "row" filters).

If the image has M x M pixels and the filter kernel has size N x N:
- What is the cost of convolution with a non-separable filter? ⟶ $M^2 \times N^2$
- What is the cost of convolution with a separable filter? ⟶ $2 \times N \times M^2$

The Gaussian filter

- named (like many other things) after Carl Friedrich Gauss

- kernel values sampled from the 2D Gaussian function:

$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2 + j^2}{2\sigma^2}}$$

- weight falls off with distance from center pixel

- theoretically infinite, in practice truncated to some maximum distance

Any heuristics for selecting where to truncate?
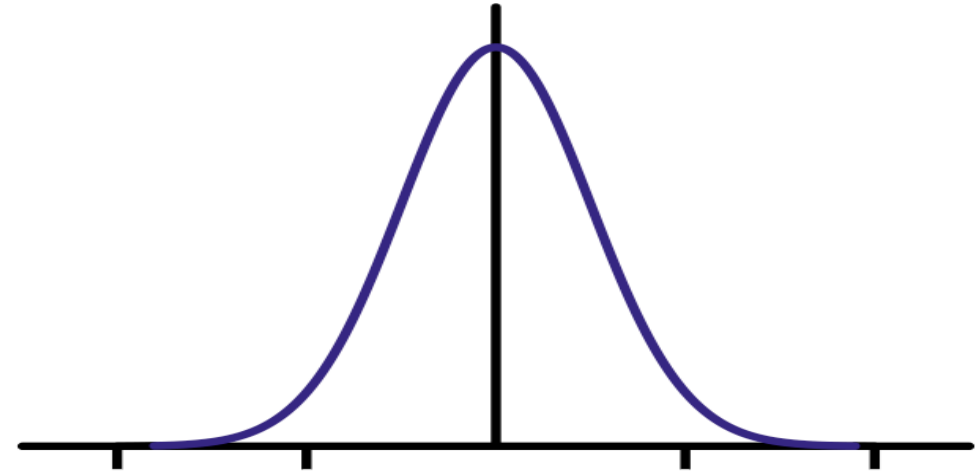
Standard deviation σ
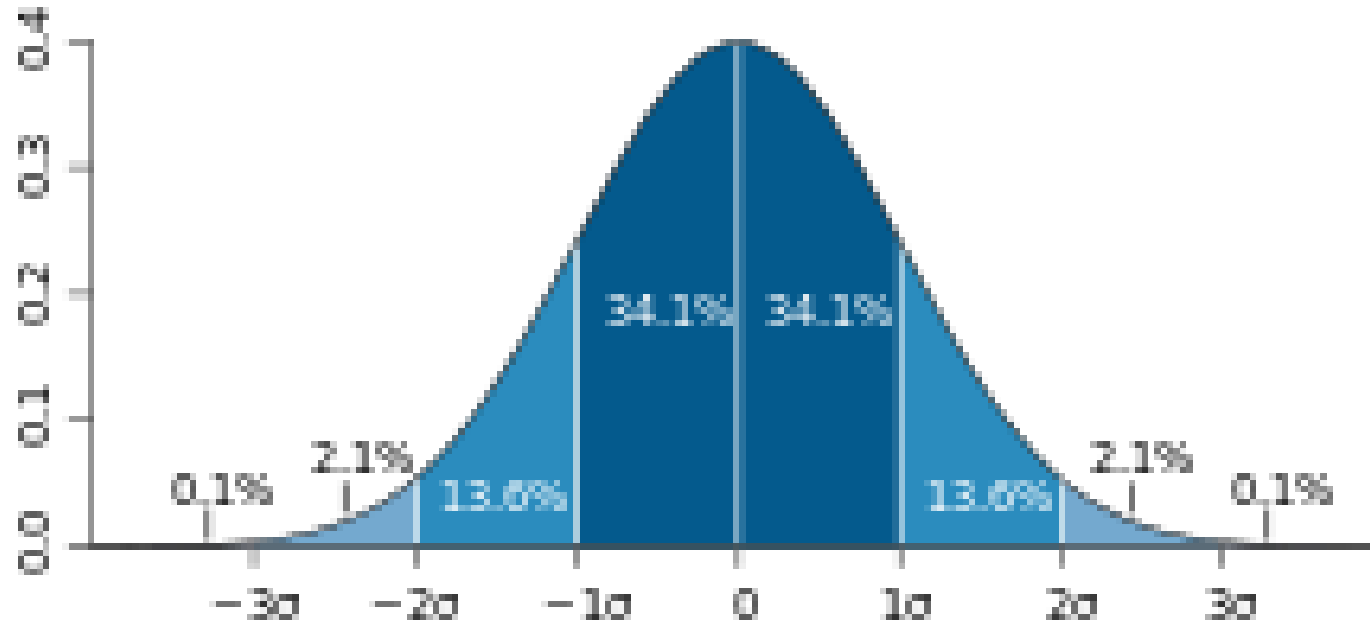
The Gaussian filter

- named (like many other things) after Carl Friedrich Gauss

- kernel values sampled from the 2D Gaussian function:

$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

- weight falls off with distance from center pixel

- theoretically infinite, in practice truncated to some maximum distance

Any heuristics for selecting where to truncate?
- usually at 2-3σ
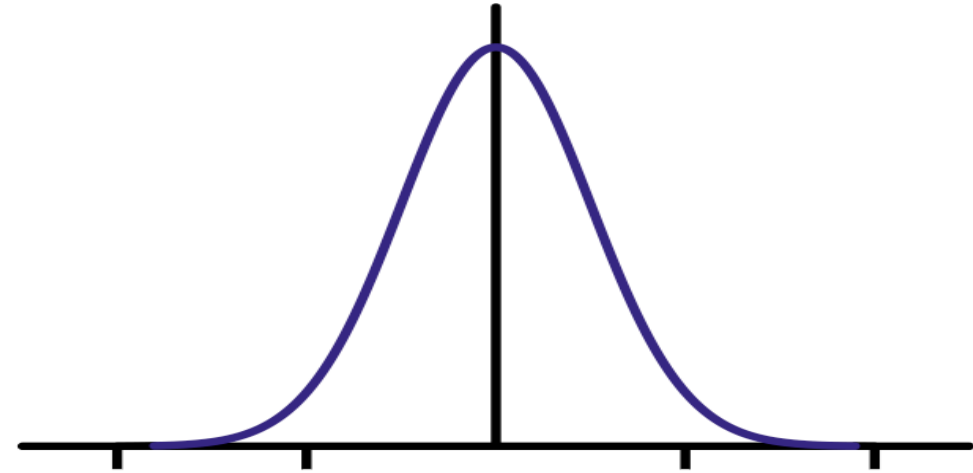
Is this a separable filter?

kernel $\frac{1}{16}$

1	2	1
2	4	2
1	2	1

The Gaussian filter

- named (like many other things) after Carl Friedrich Gauss

- kernel values sampled from the 2D Gaussian function:

$$f(i, j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2 + j^2}{2\sigma^2}}$$

- weight falls off with distance from center pixel

- theoretically infinite, in practice truncated to some maximum distance

Any heuristics for selecting where to truncate?
- usually at 2-3σ
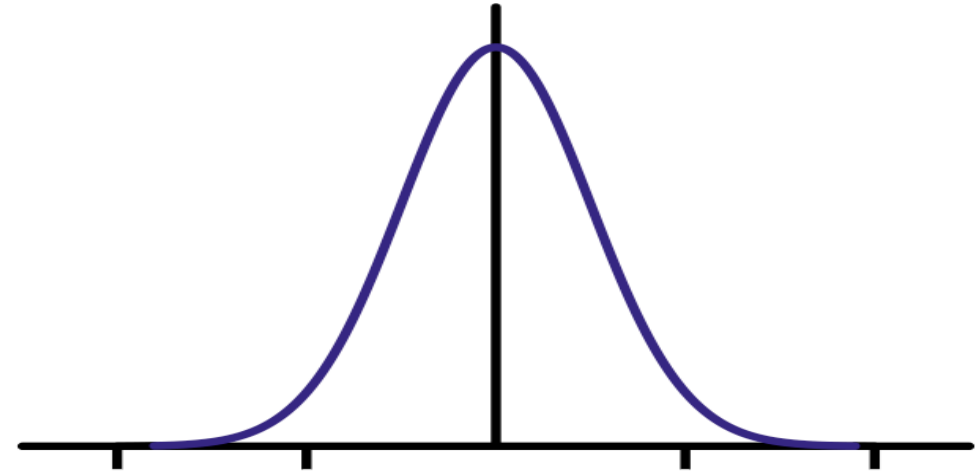
Is this a separable filter?   Yes!

kernel $\frac{1}{16}$

1	2	1
2	4	2
1	2	1

The Gaussian Filter
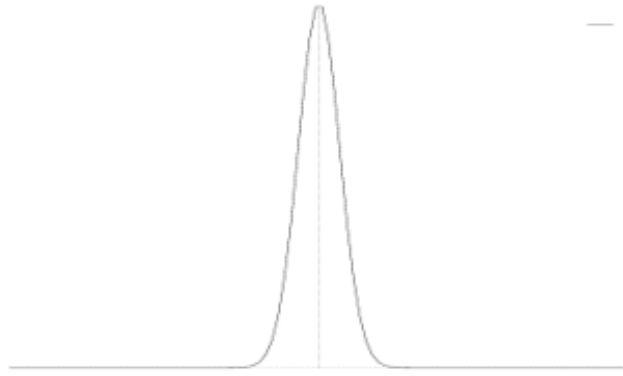
$$g(x) = e^{\frac{-x^2}{2o^2}}$$
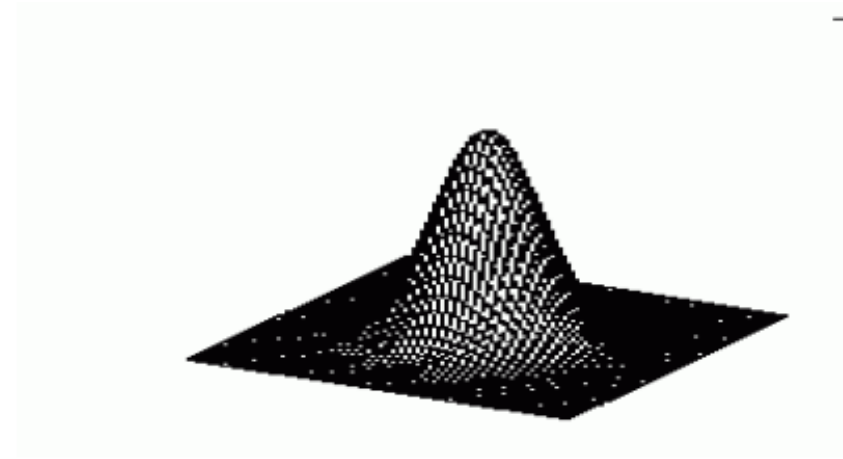
$$g(x,y) = e^{\frac{-(x^2+y^2)}{2o^2}}$$

$$g(x) = [.011 \quad .13 \quad .6 \quad 1 \quad .6 \quad .13 \quad .011]$$

$$\sigma = 1$$

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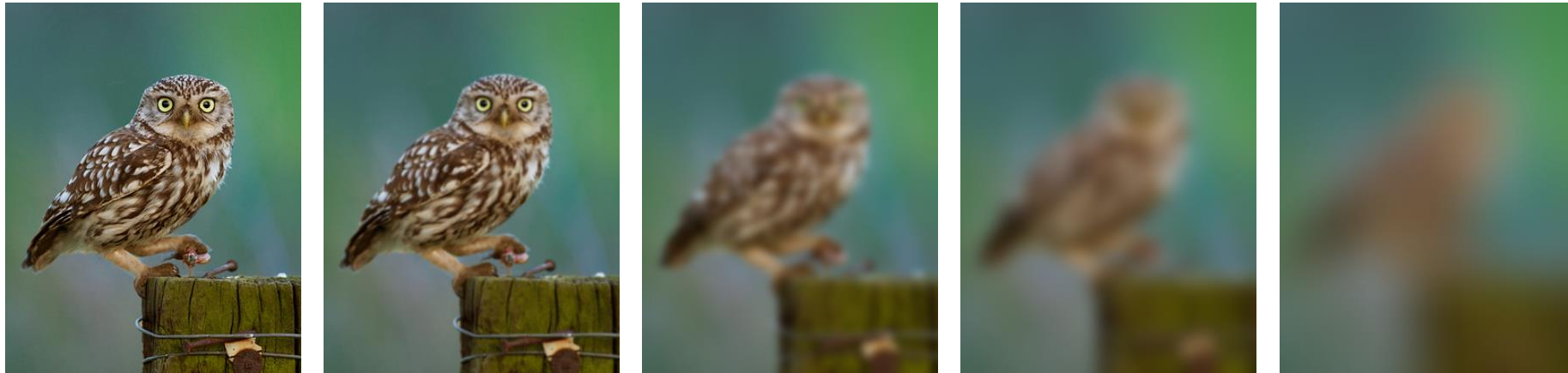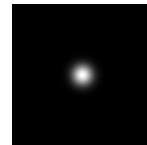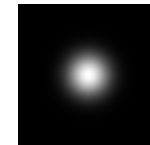
σ = 1 pixel σ = 5 pixels σ = 10 pixels σ = 30 pixels

Filtering Examples - 5

$*$ $=$
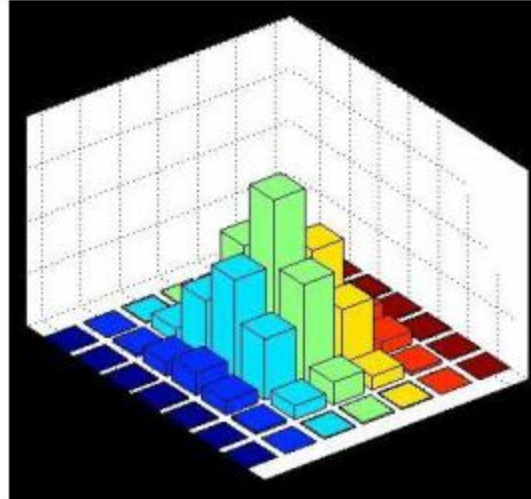
Filtering Examples - 5
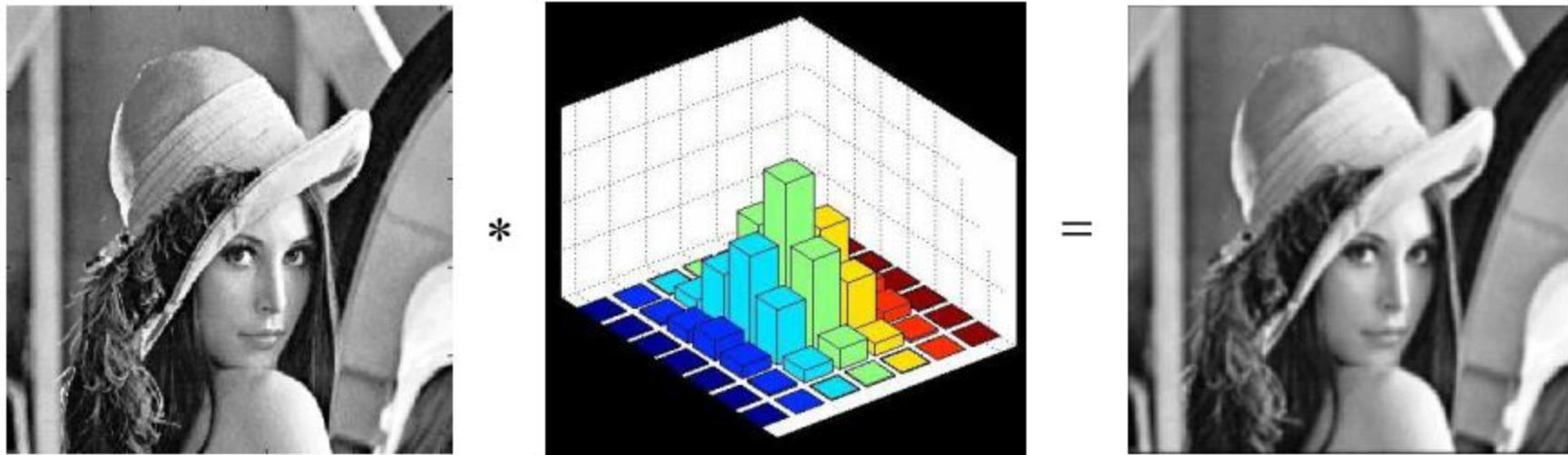
Gaussian Smoothing

Filtering Examples - 6

Gaussian Smoothing

Smoothing by Averaging

Filtering Examples - 7

After additive
Gaussian Noise

After Averaging

After Gaussian Smoothing

Filtering Examples – 8
Sharpening

input   filter   output

$$0 \quad 0 \quad 0$$
$$0 \quad 2 \quad 0 \qquad - \frac{1}{9} \qquad 1 \quad 1 \quad 1$$
$$0 \quad 0 \quad 0 \qquad\qquad 1 \quad 1 \quad 1$$

(Note that filter sums to 1)

- do nothing for flat areas
- stress intensity peaks

Filtering Examples – 8
Sharpening

Accentuates differences with local average

input   filter   output

0	0	0
0	2	0
0	0	0

$- \dfrac{1}{9}$

1	1	1
1	1	1
1	1	1
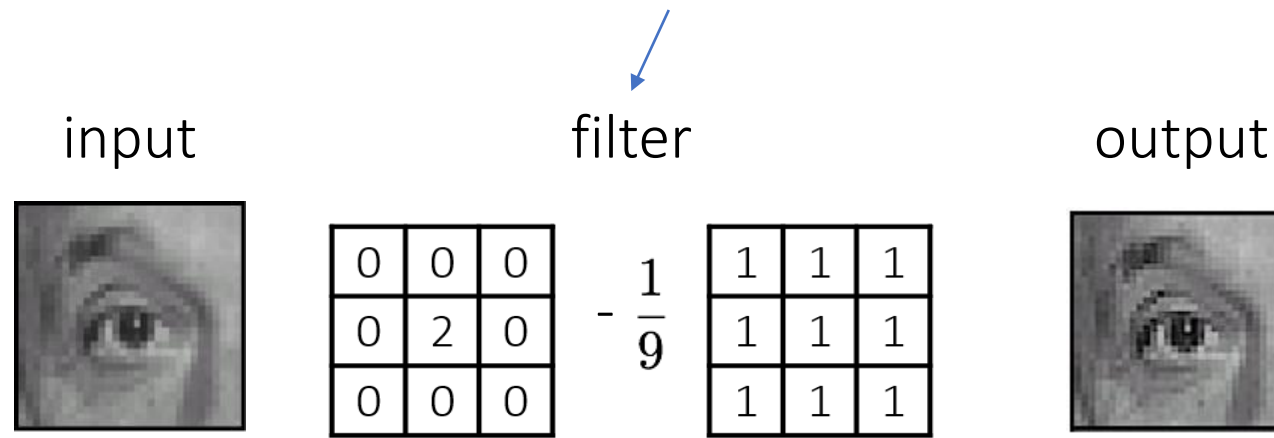
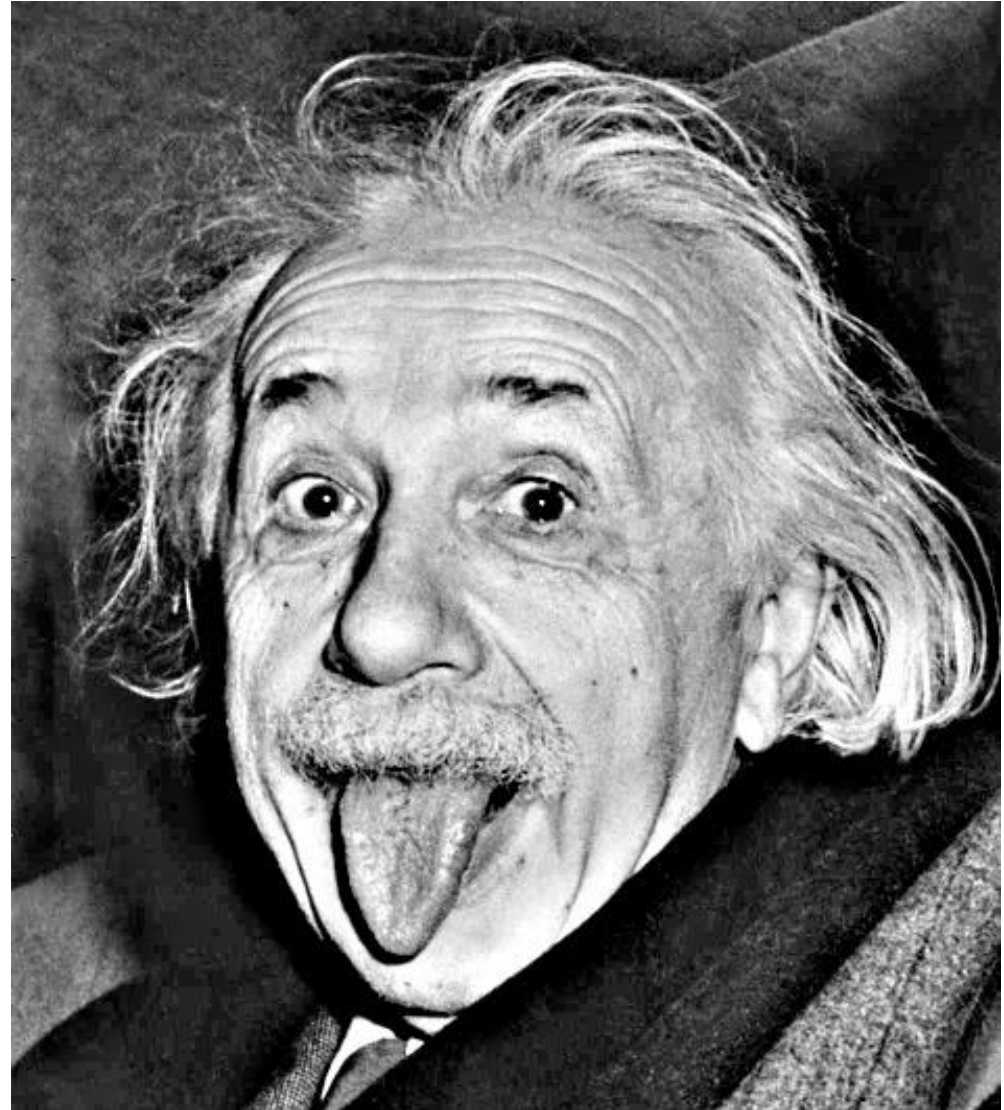(Note that filter sums to 1)

- do nothing for flat areas
- stress intensity peaks

Sharpening

- What does blurring take away?
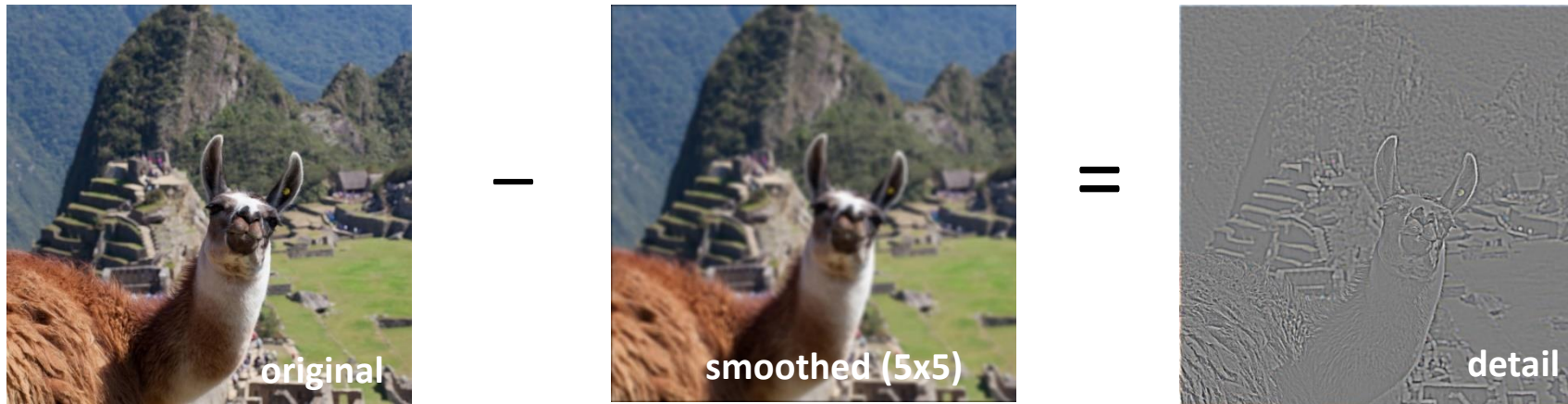
(This "detail extraction" operation is also called a **high-pass filter**)

Let's add it back:
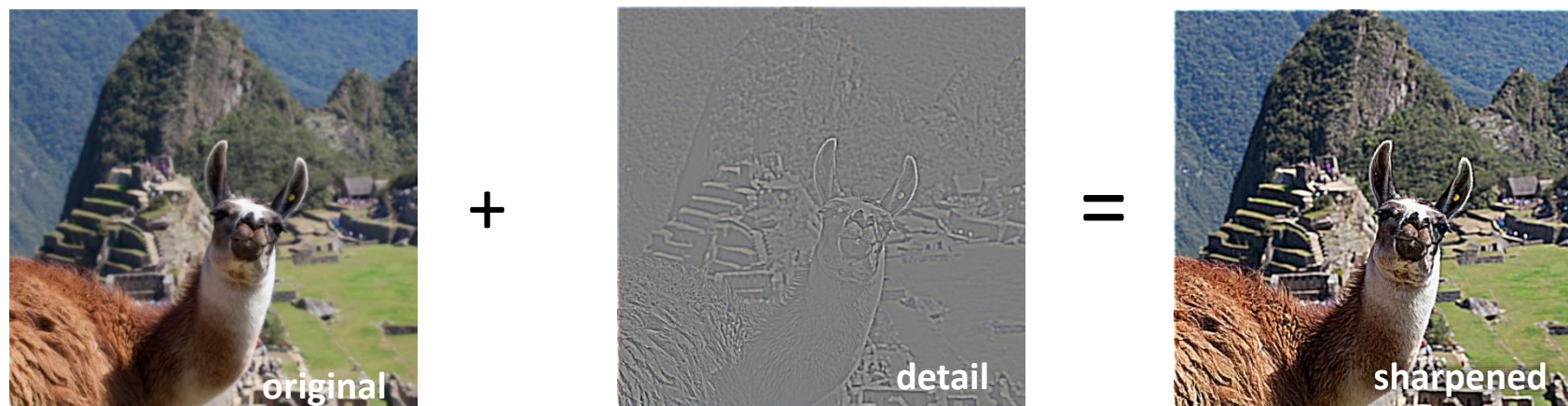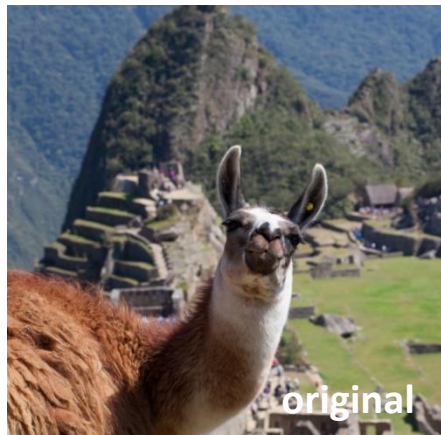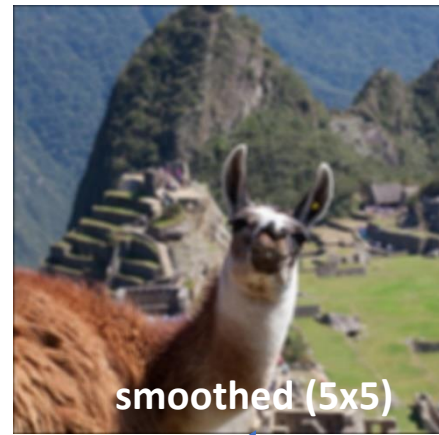
Photo credit: https://www.flickr.com/photos/geezaweezer/16089096376/

Sharpening

- What does blurring take away?

original − smoothed (5x5)

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

original + detail = sharpened

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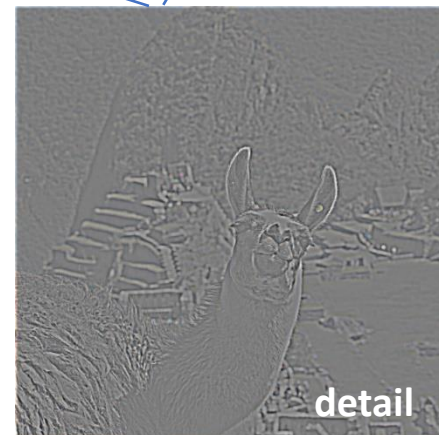
Sharpening

- ## What does blurring take away?

2 times original

$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

−

Smoothed

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

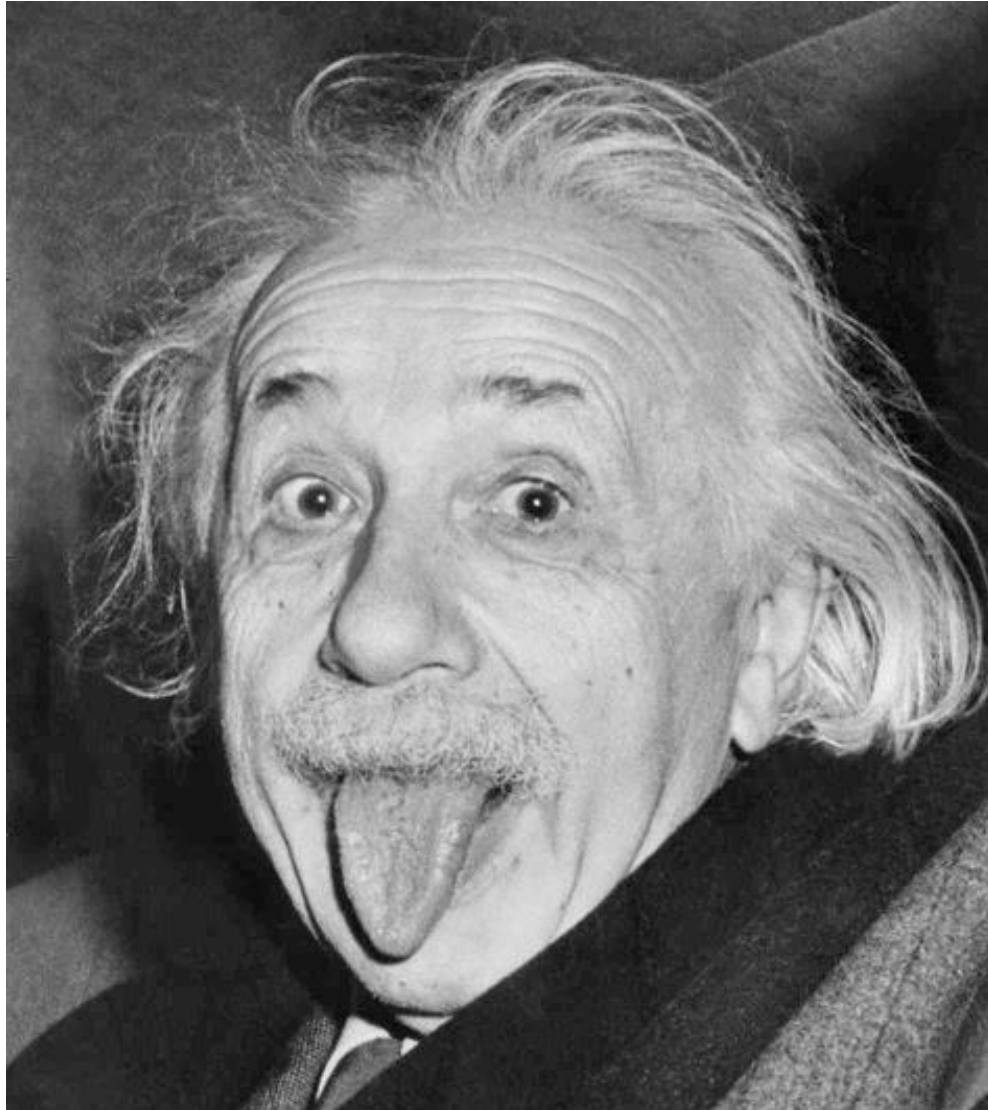(This "detail extraction" operation is also called a **high-pass filter**)
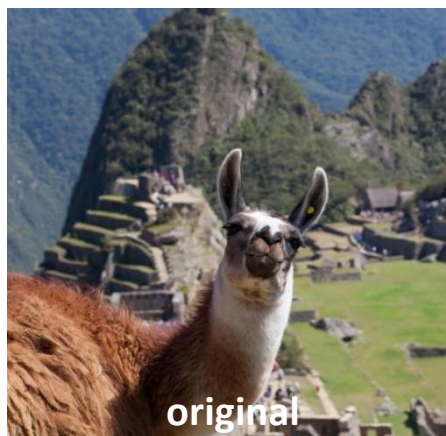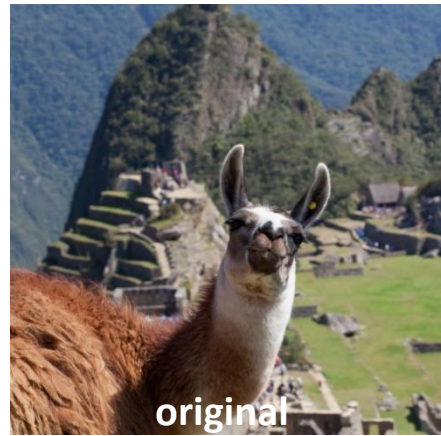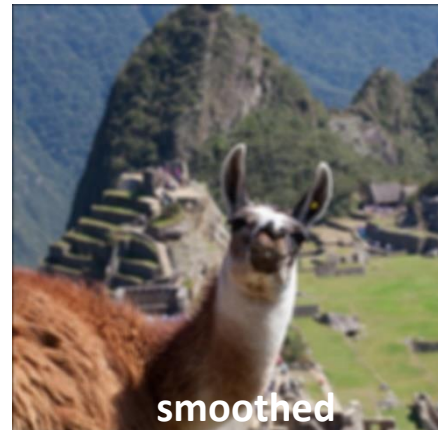
original + original − smoothed = sharpened

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

Sharpening

- What does blurring take away?

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

original + original − smoothed (5x5) = sharpened

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

Sharpening examples

Median Filter

- A **Median Filter** operates over a window by selecting the median intensity in the window.

Image filtering - median

$f[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
			?						

Image filtering - median

$$f[.,.]$$

$$h[.,.]$$

	0	10	20	30	30				
				90					

Median of {0,0,0,0, 90, 90,90,90,90}

Median Filter

- A **Median Filter** operates over a window by selecting the median intensity in the window.

- Great to deal with salt and pepper noise !

Median Filter

Mean   Gaussian   Median

3x3

5x5

7x7
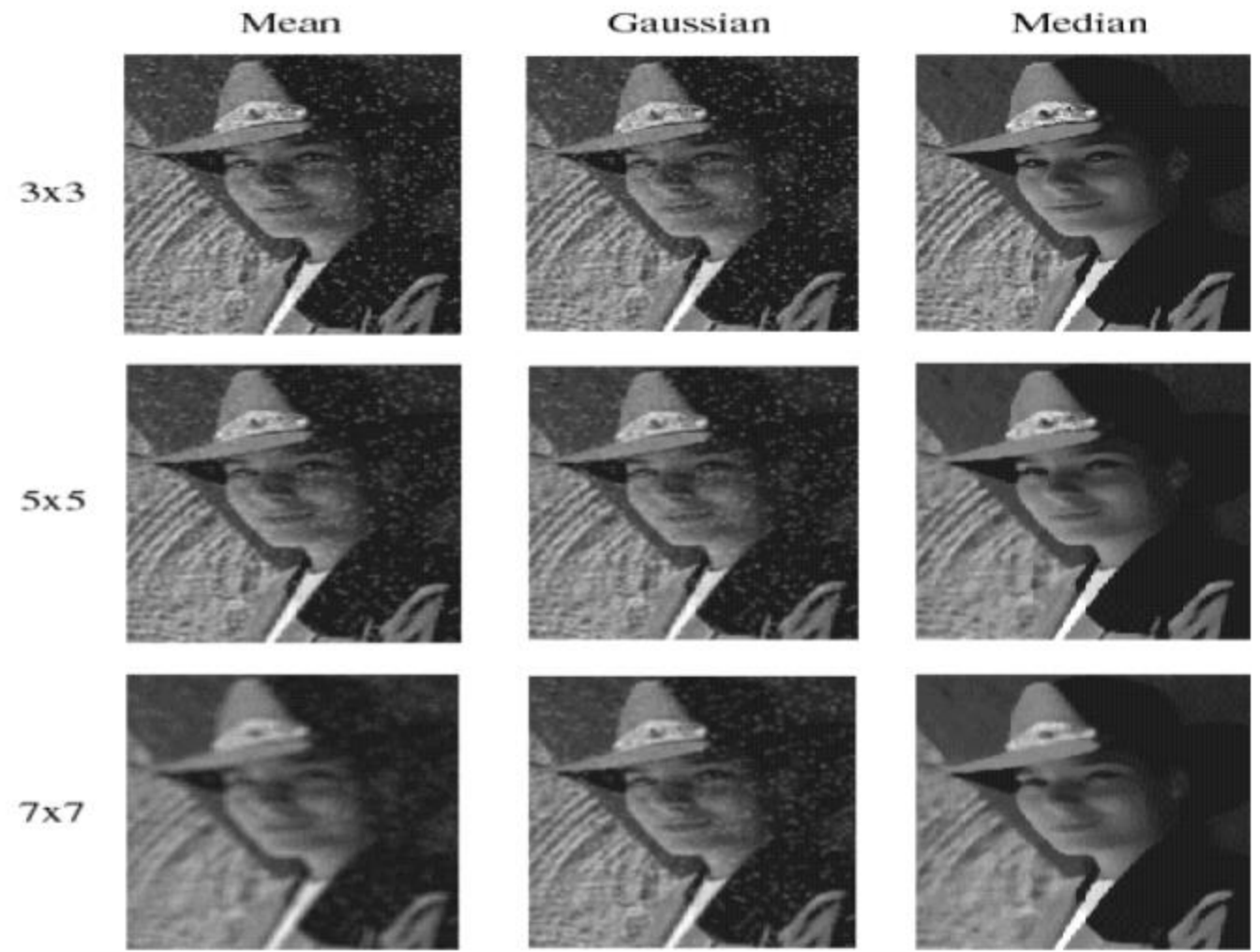
Image Boundary Effect
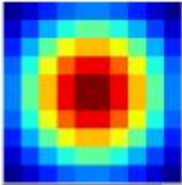
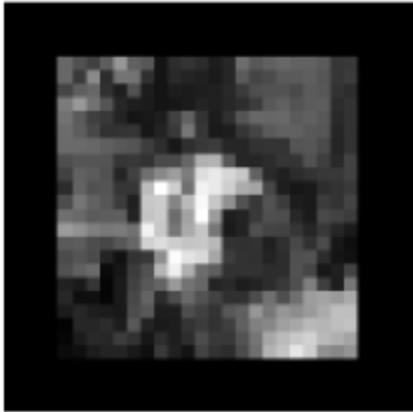The filter window falls off at the edge of image.

Practical matters

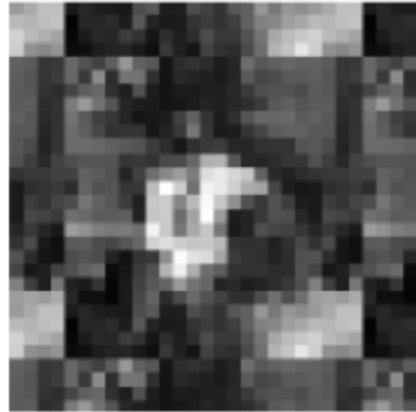## What about near the edge?

- The filter window falls off the edge of the image
- Need to extrapolate
- methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge

Source: S. Marschner

zero   wrap   clamp   mirror

Copy edge   Reflect across edge

Questions?