Image Filtering

Computer Vision

Albert Chung
Department of Computer Science
University of Exeter

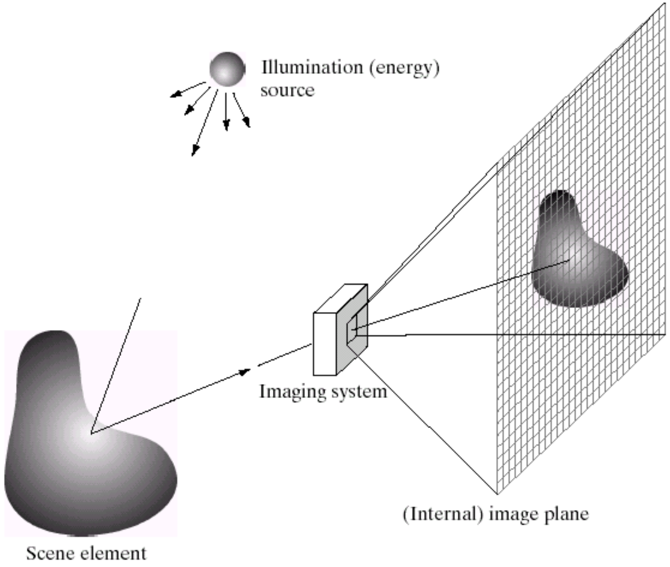
Spring Term 2023

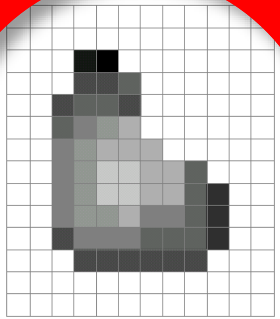
Main Reference

- R. Szeliski, “Computer Vision: Algorithms and Applications”:
  - Linear filtering: Chapter 3.2
  - Non-linear filtering: Chapter 3.3.1

What is an image?

What is an image?

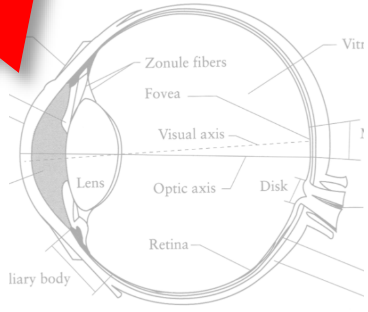




Digital Camera

We’ll focus on these in this class

(More on this process later)

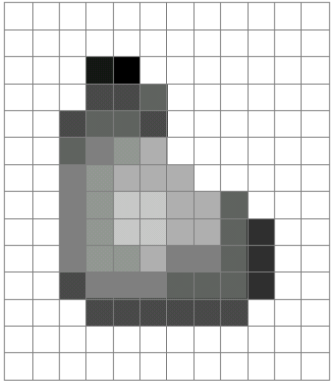


The Eye

Source: A. Efros

What is an image?

- A grid (matrix) of intensity values



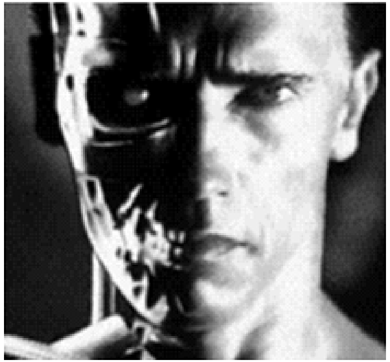
=

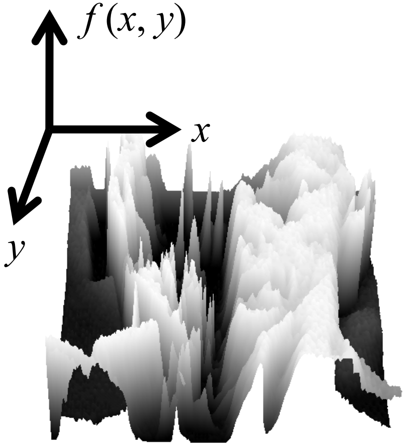
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175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

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

Image as a function

- 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)





3D view

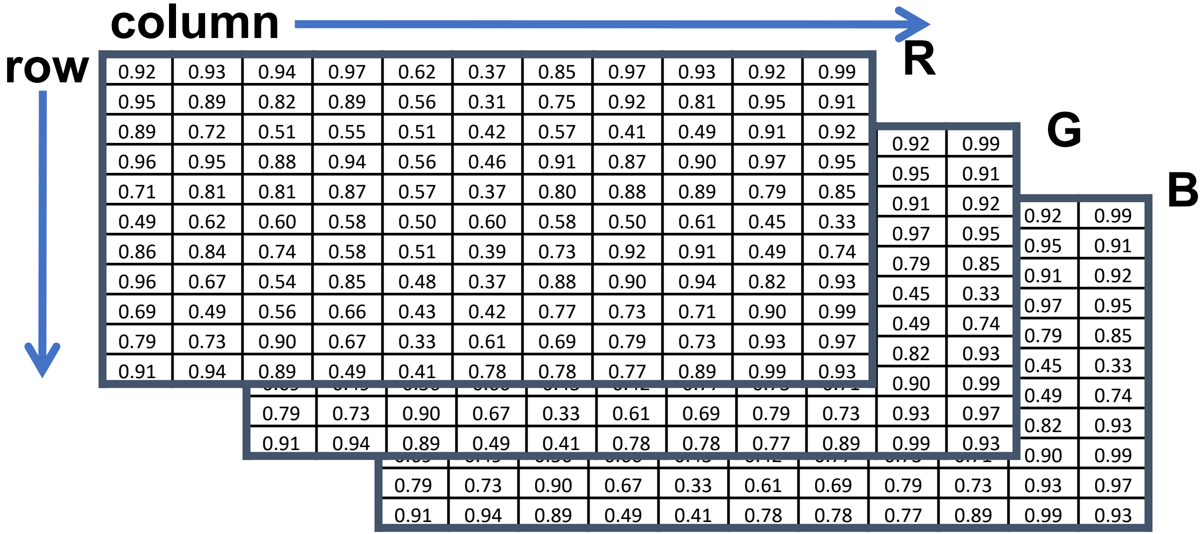
- A digital image is a discrete (sampled, quantized) version of this function

Colour Images

One (grayscale) image per RGB channel or one 3D vector per point (x, y) or (i, j)

Images in Python

- Images represented as a matrix
- Suppose we have a NxM RGB image called "im"
  - im[0, 0, 0] = top-left pixel value in R-channel
  - im[y, x, b] = y pixels down, x pixels to right in the $(b-1)^{th}$ channel
  - im[N-1, M-1, 2] = bottom-right pixel in B-channel
- cv2.imread(filename) returns a uint8 image (values 0 to 255)
  - Convert to double format (values 0 to 1) with cv2.normalize()

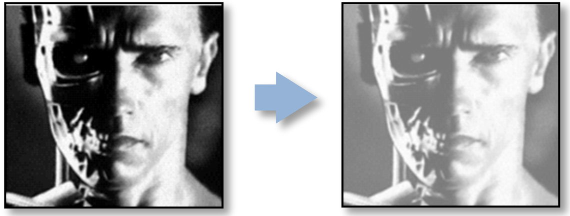


0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99
0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91
0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92
0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95
0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85
0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33
0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74
0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93
0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

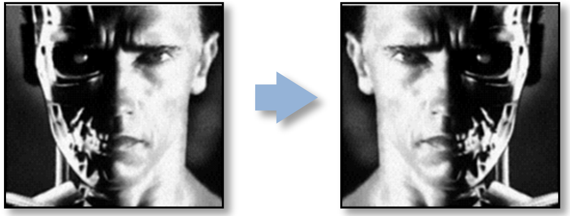
James Hays

Image transformations

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



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



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

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

Filters

- Image filtering
  - Form a new image whose pixels are certain statistics of the original pixels
  - Modify the pixels in an image based on some function of a local neighborhood of each pixel

10	5	3
4	5	1
1	1	7

Local image data

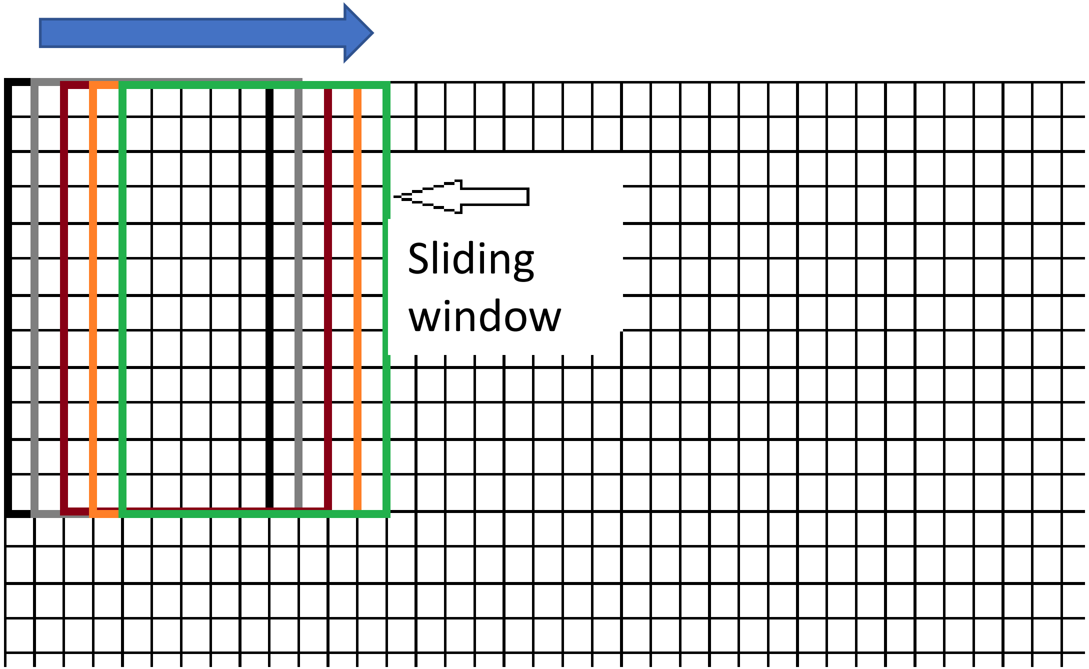
Some function

	8	

Modified image data

- Reasons
  - To get useful information from images
    - Extract edges or contours (to understand shape)
    - Preprocess image to detect corners or features
  - To enhance the image
    - Remove or reduce noise
    - Sharpen and “enhance image”
  - A key operator in Convolutional Neural Networks

Sliding window



Cross-correlation

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

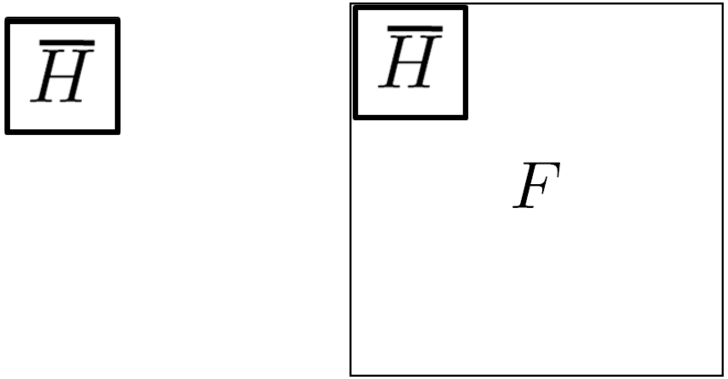
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

- This is called cross-correlation operation:

$$G = H \otimes F$$

- It can be thought of as a “dot product” between local neighborhood and kernel for each pixel

Cross-correlation



Convolution

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

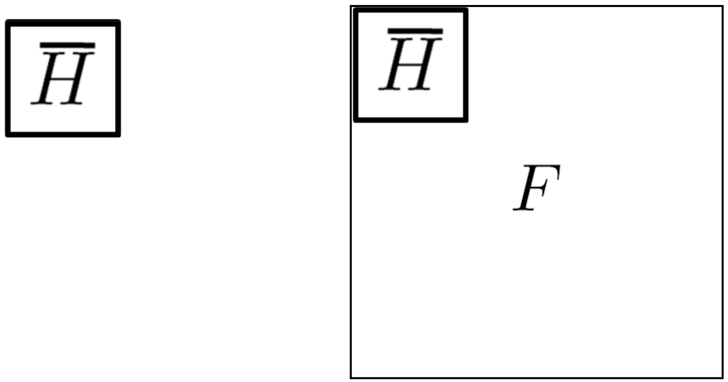
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

- This is called a convolution operation:

$$G = H * F$$

- Convolution is commutative and associative
- Modern deep learning libraries usually perform cross-correlation and call it convolution

Convolution



Linear filtering

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

10	5	3
4	6	1
1	1	8

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	8	

Modified image data

Mean filtering/Moving average

H

$*$

F

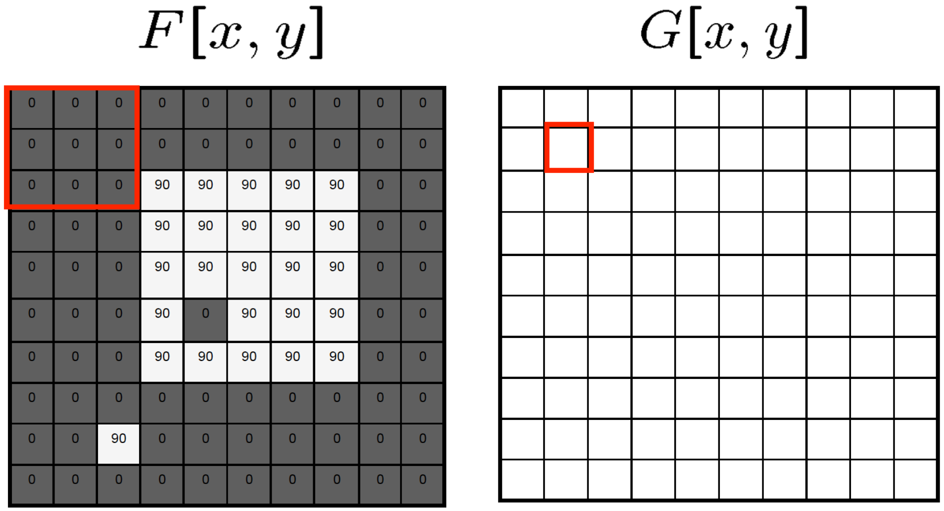
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$=$

G

	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	

Mean filtering/Moving average



Mean filtering/Moving average

$$F[x, y]$$

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

$$F[x, y]$$

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

$F[x, y]$

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

$F[x, y]$

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

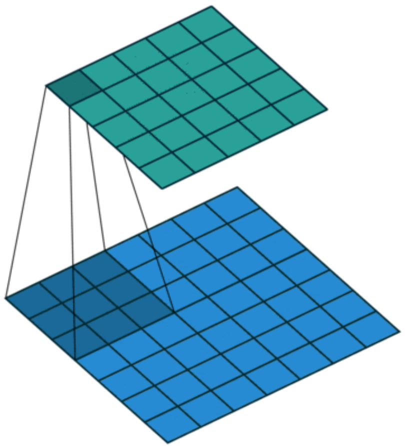
$F[x, y]$

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

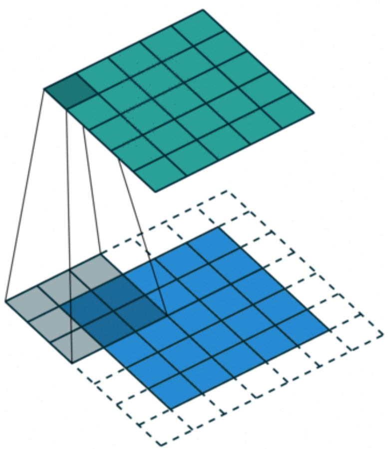
$G[x, y]$

	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	

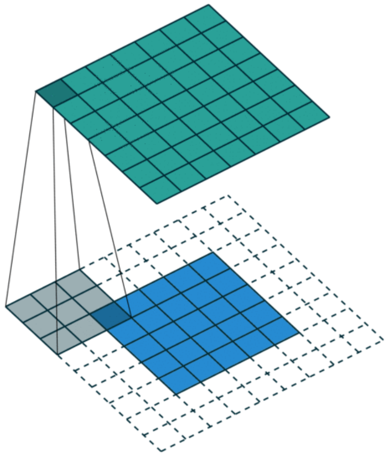
Different settings of convolution



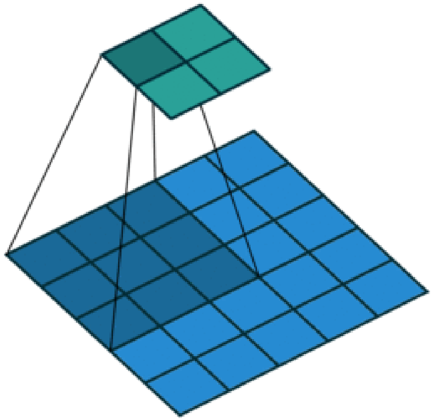
No padding, stride = 1



Half padding, stride = 1



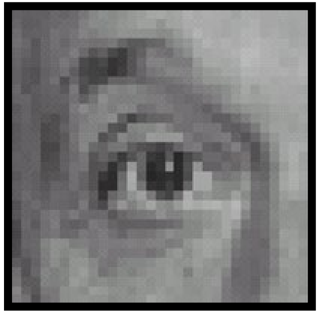
Full padding, stride = 1



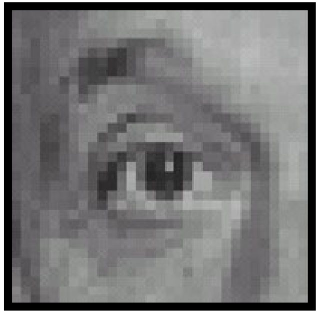
No padding, stride = 2

More animations: https://github.com/vdumoulin/conv_arithmetic

Linear filters: example



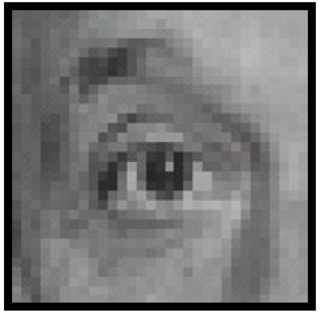
$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$



Original

Identical image

Linear filters: example

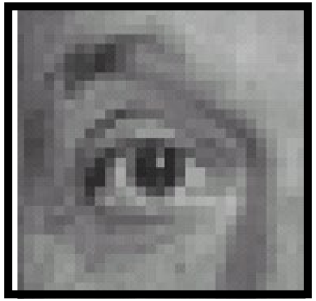


Original

$$*$$

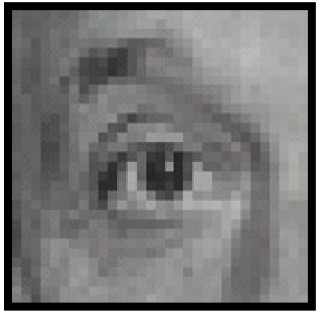
0	0	0
1	0	0
0	0	0

$$=$$



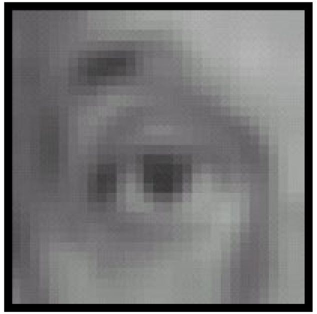
Shifted right by 1 pixel

Linear filters: example



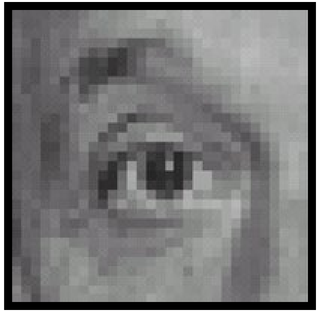
Original

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



Blur (with a mean filter)

Linear filters: sharpening

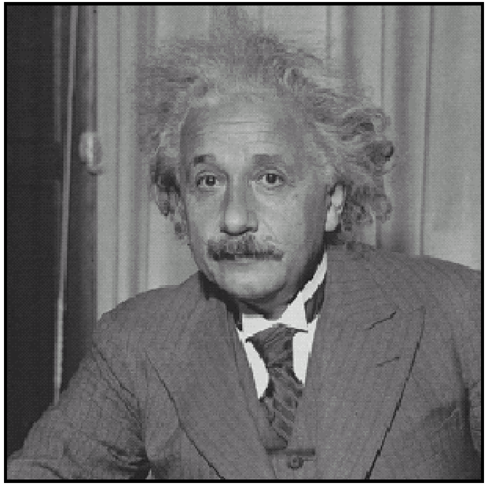


Original

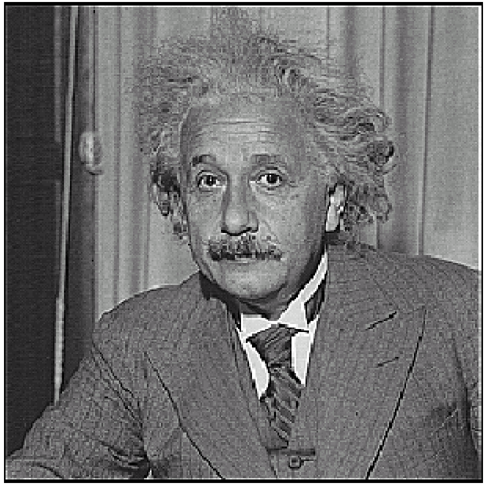
$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

Sharpening filter
(accentuates edges)

Sharpening



before



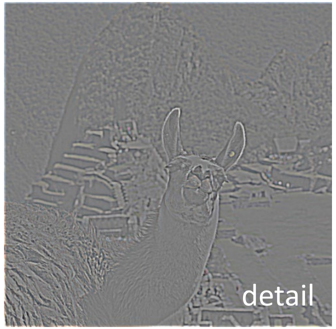
after

Sharpening

- What does blurring take away?

–

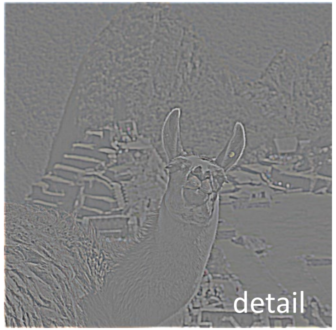
=



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

Let’s add it back:

+ α

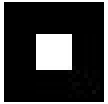


=

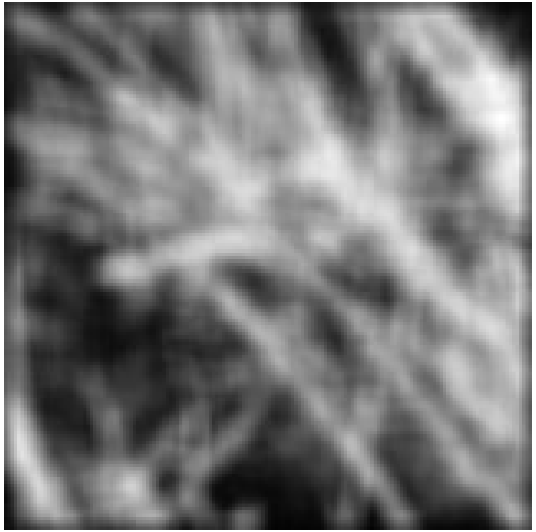
Photo credit: https://www.flickr.com/photos/geezaweezer/16089096376/

Ringing artifacts

- Box filter creates “ringing” artifacts.



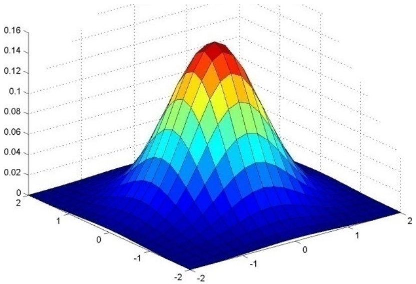
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

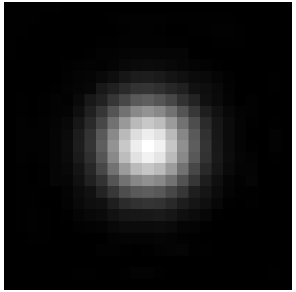


Source: D. Forsyth

Gaussian Kernel

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

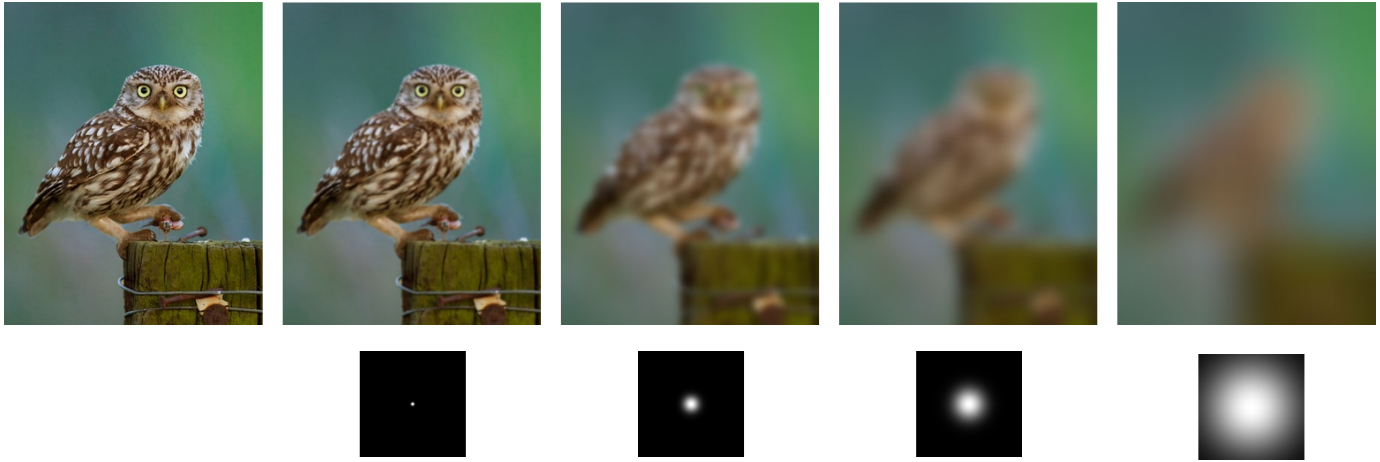




0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, σ = 1

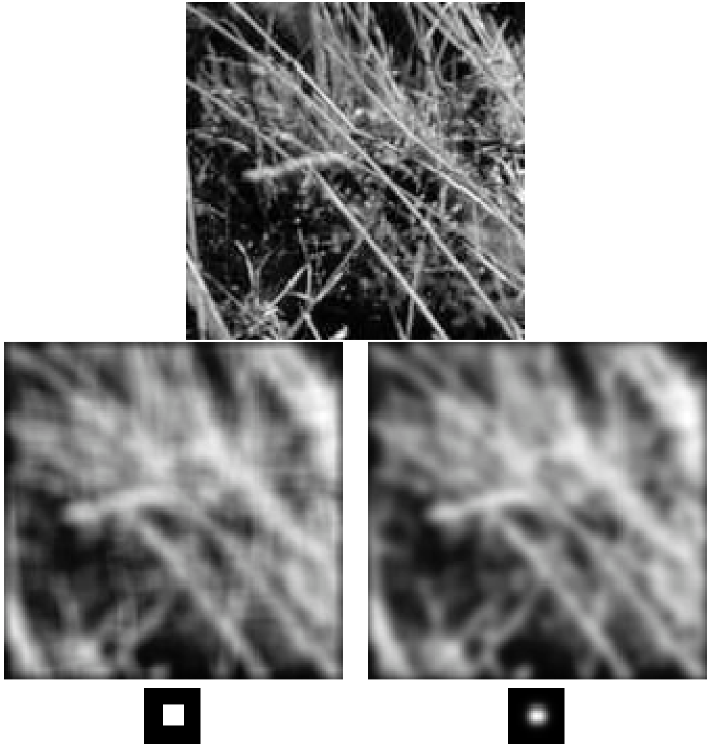
Gaussian filters



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

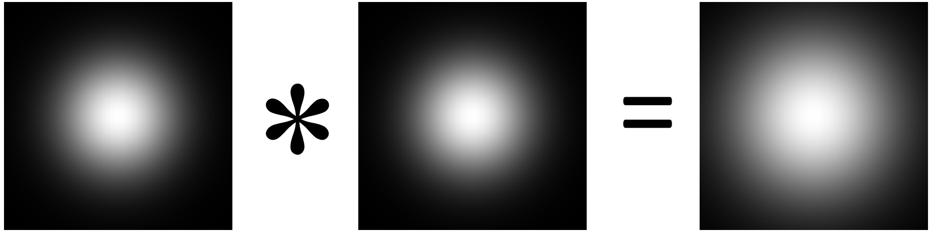
Mean vs. Gaussian filtering

- Gaussian filter is always preferred to box/mean filter when an image with lot of sharp edges is considered.



Gaussian filter

- Removes “high-frequency” components from the image (low-pass filter)
- Convolution with self is another Gaussian



  - Convolving twice with Gaussian kernel of width σ = convolving once with kernel of width $\sigma\sqrt{2}$

Source: K. Grauman

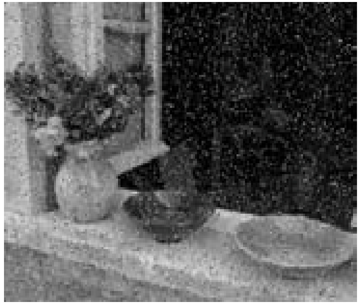
Non-linear Filtering

- Gaussian filter fails to remove occasionally high noise.

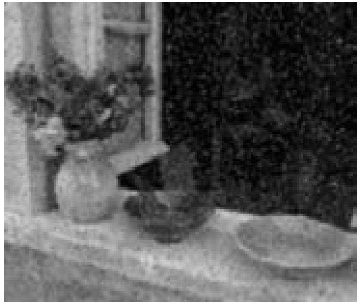
Gaussian noise

Gaussian filtered

median filtered



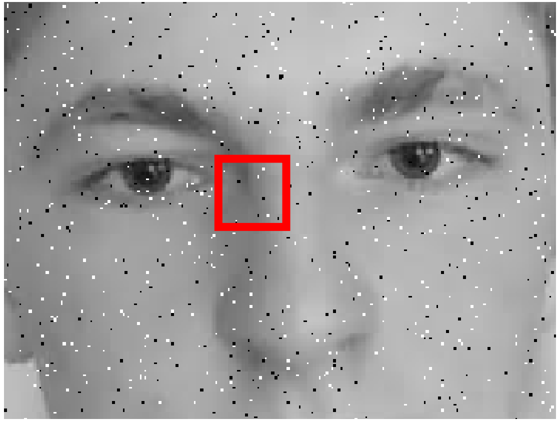
shot noise



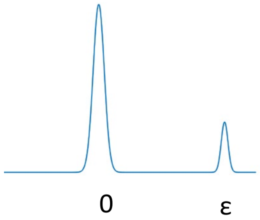
Gaussian filtered

median filtered

Median Filter



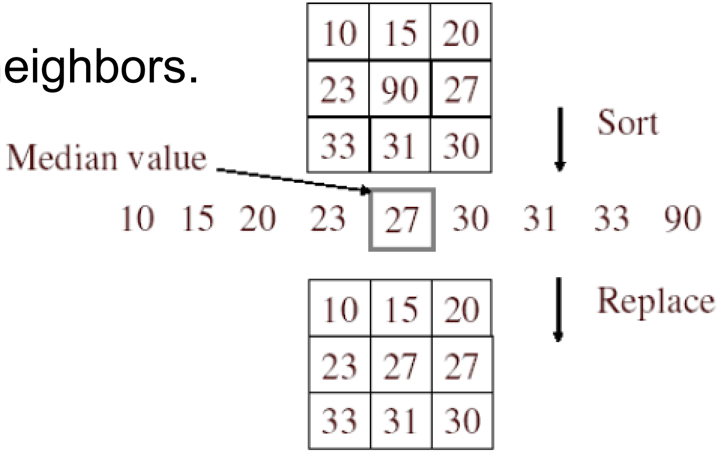
$$f(x,y) = \overbrace{\hat{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$



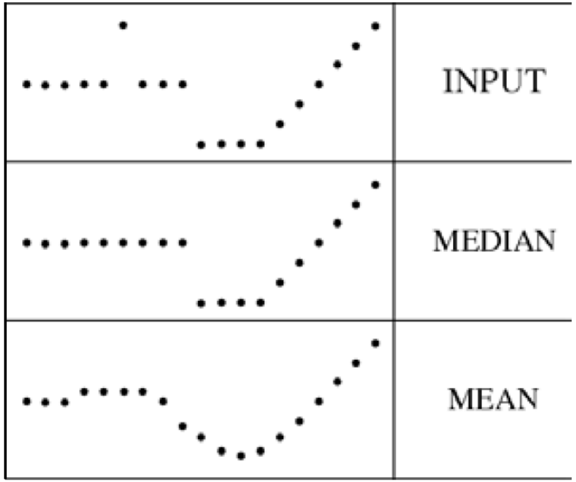
- robust statistics for outlier rejection
- median of a distribution is a robust measure (as opposed to the mean)

Median Filter

- Replace each pixel by the median of its neighbors.



- Comparison with mean:



Example: Mean Filter result?

$h[\cdot,\cdot]$

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

$f[.,.]$

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

$g[.,.]$

	0	10	20	30	30				
				?					

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

Example: Median Filter result?

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

				?					

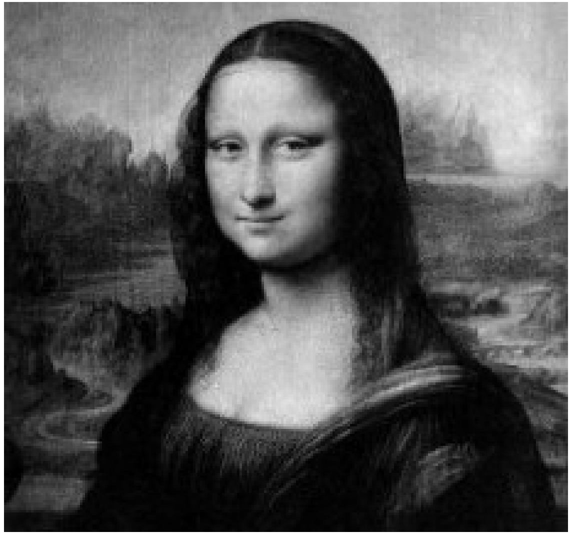
Median vs. Gaussian Filtering

3x3 5x5 7x7

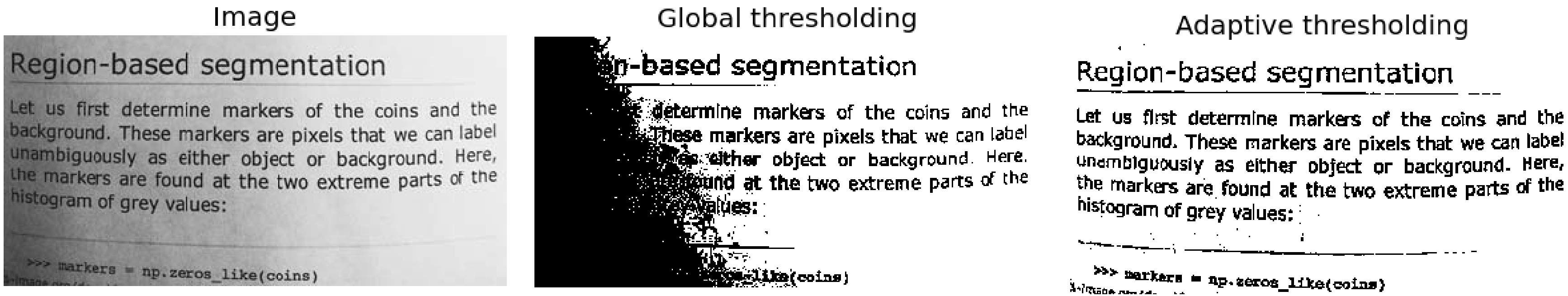
Gaussian

Median

Non-linear Filter: Thresholding



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



Summary

- Image: definition and representation
- Sliding window: cross-correlation and convolution
- Linear filtering: box and Gaussian filter, sharpening
- Non-linear filtering: median filtering, thresholding