THE UNIVERSITY OF TEXAS AT DALLAS

Image Processing: Filtering I

CS 4391 Introduction to Computer Vision

Professor Yapeng Tian

Department of Computer Science

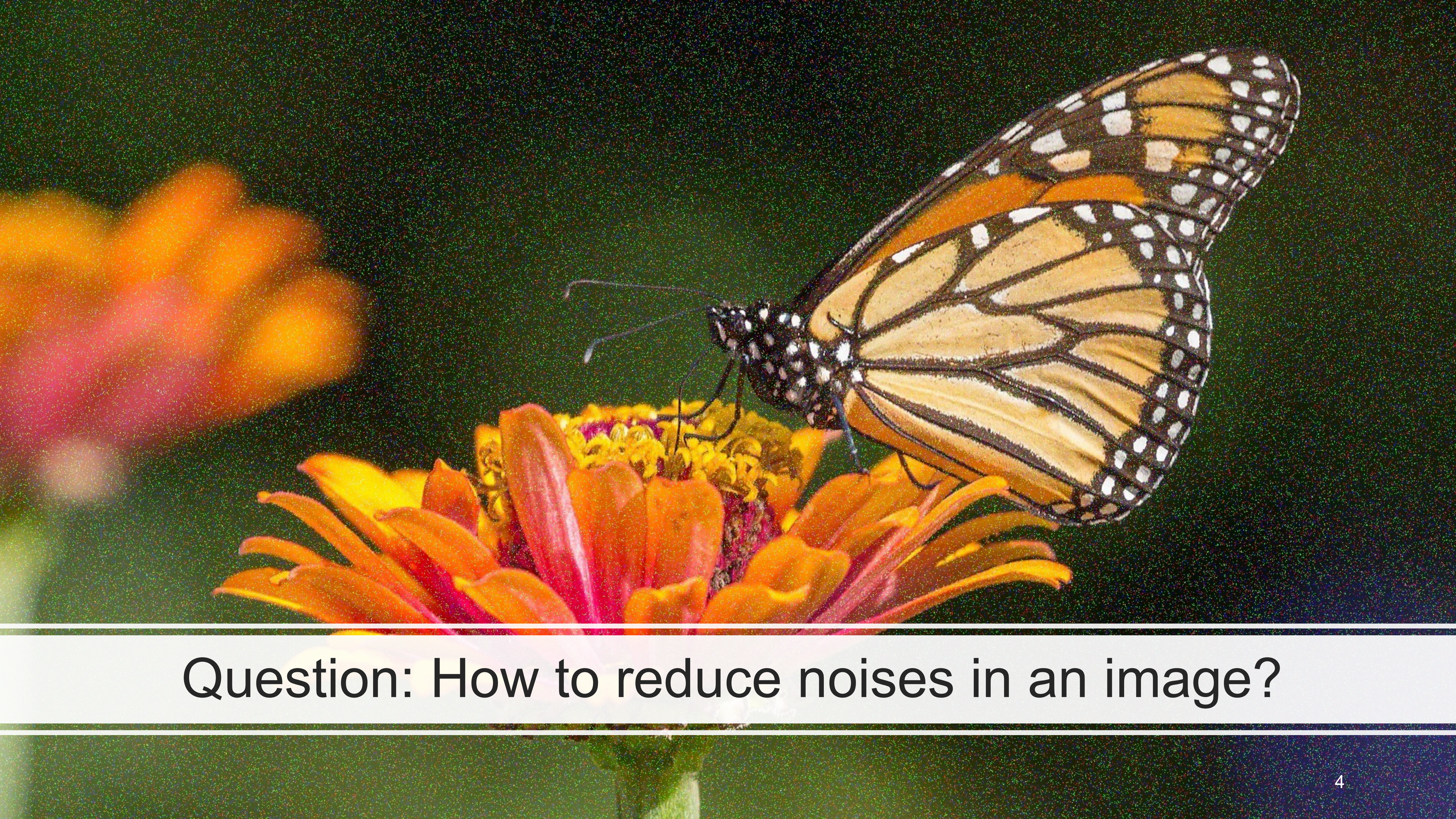

Question: How to reduce noises in an image?

Image Filtering

- Goal: generate a new image G whose pixel values are a combination of the original pixel values F
  - Enhance image quality (e.g., denoising, sharpening)
  - Extract visual features (e.g., edges, contours)
  - Basic computation unit in convolutional neural networks

F

G

Noise Reduction as An Example

How was the noisy image generated?

$$F[i, j, c] = I[i, j, c] + n[i, j, c]$$

i : row, j:column, c:color, n: additive noise

How to remove the noise n from the noisy image ?

Characteristics of Noises and Natural Images

Image noises:

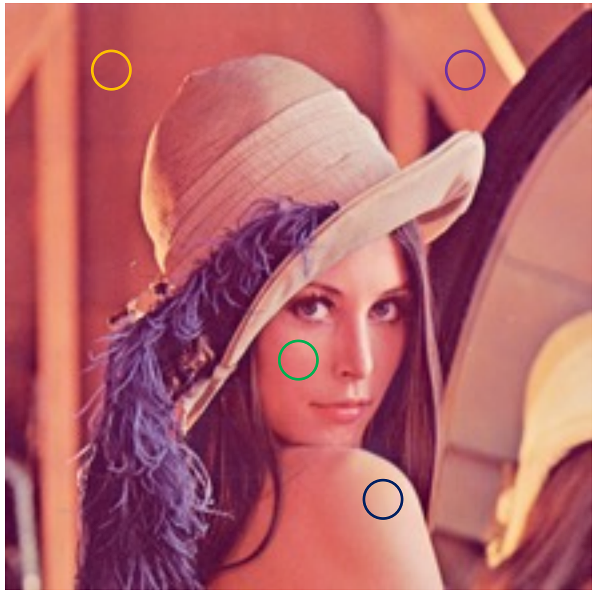
- Random and characterized by high frequency components
- Fewer details or finer textures

Natural images:

- Both low and high frequencies that are more evenly distributed
- More textures, patterns, and shapes with gradual changes in intensity or color

Image Prior: Local Smoothness

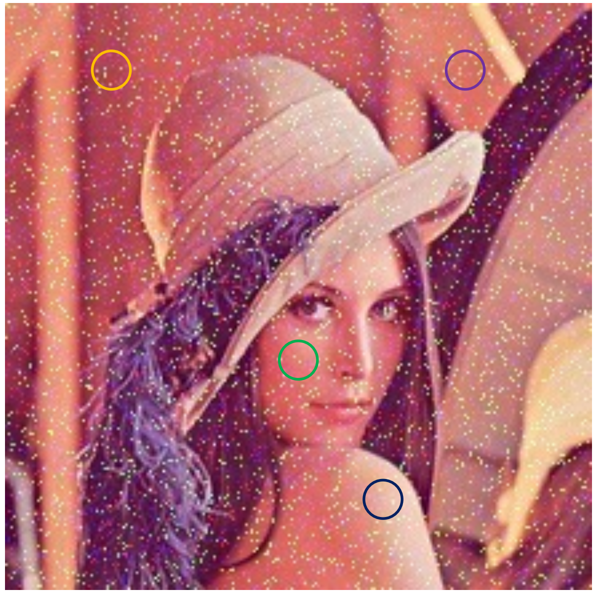
- Local natural image regions are typically smooth or uniform
- The overall structures or texture of a natural image often has a more subtle and gradual variation than image noise



- Image pixels in a small window (e.g., 5x5) usually are similar
- Noise values are dramatically changing at arbitrary directions

Image Prior: Local Smoothness

- Local natural image regions are typically smooth or uniform
- The overall structures or texture of a natural image often has a more subtle and gradual variation than image noise



- Image pixels in a small window (e.g., 5x5) usually are similar
- Noise values are dramatically changing at arbitrary directions
- Due to noises, a noisy image have higher local variations than the clean image

Image Filtering for Noise Reduction

Reduce noises by enforcing local smoothness prior

- Make each pixel in a noisy image to be similar to its local neighborhoods
- How? There are many local neighborhoods (e.g., 9 in a 3x3 window)
    - A naïve method: replace each pixel value with the mean value of its local neighborhoods

2	5	3
4	15	1
4	3	8

Local image data

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

Filter/kernel

	5	

Modified image data

Image Filtering Process

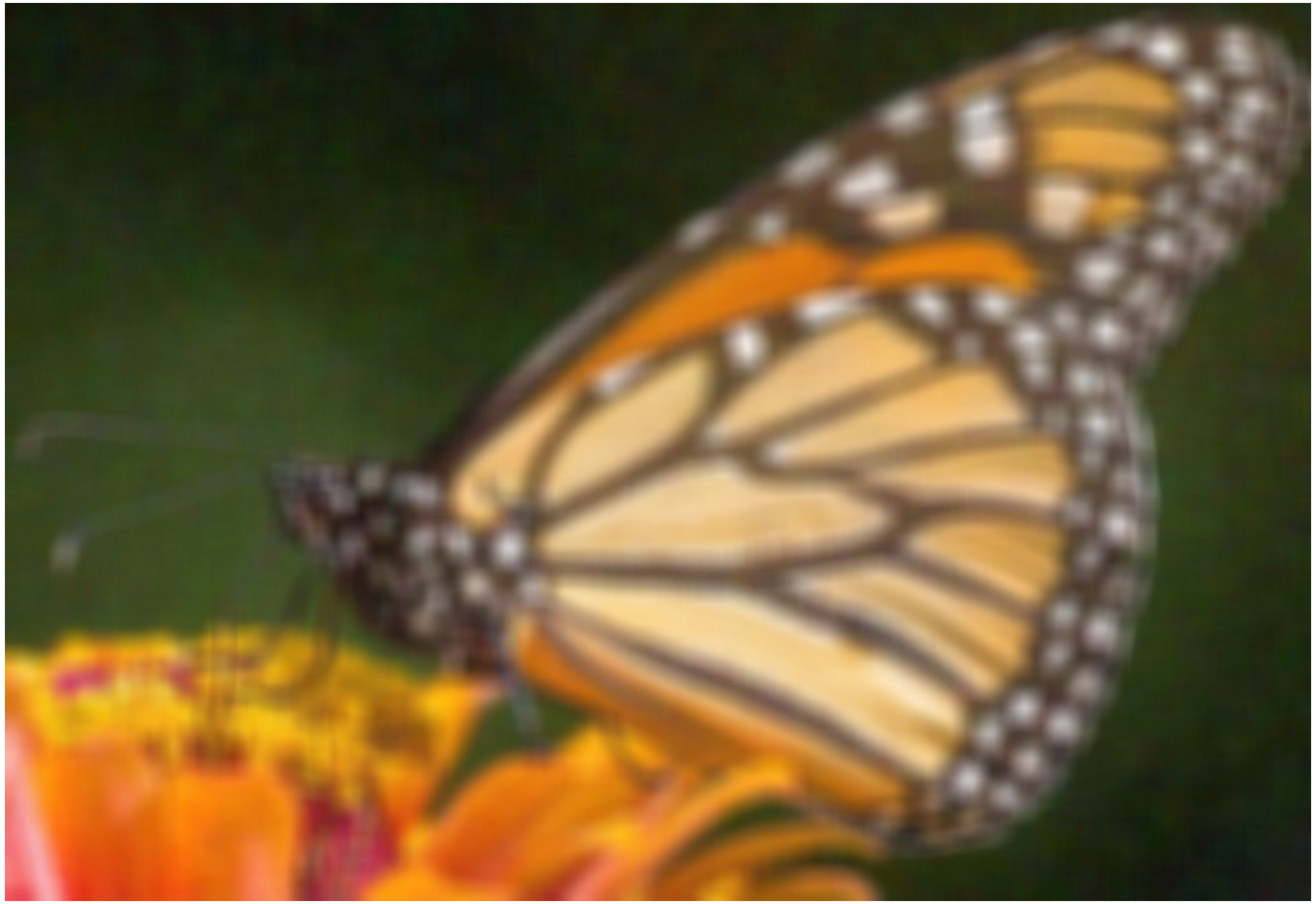
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Noisy Image

Apply the filter to every pixel

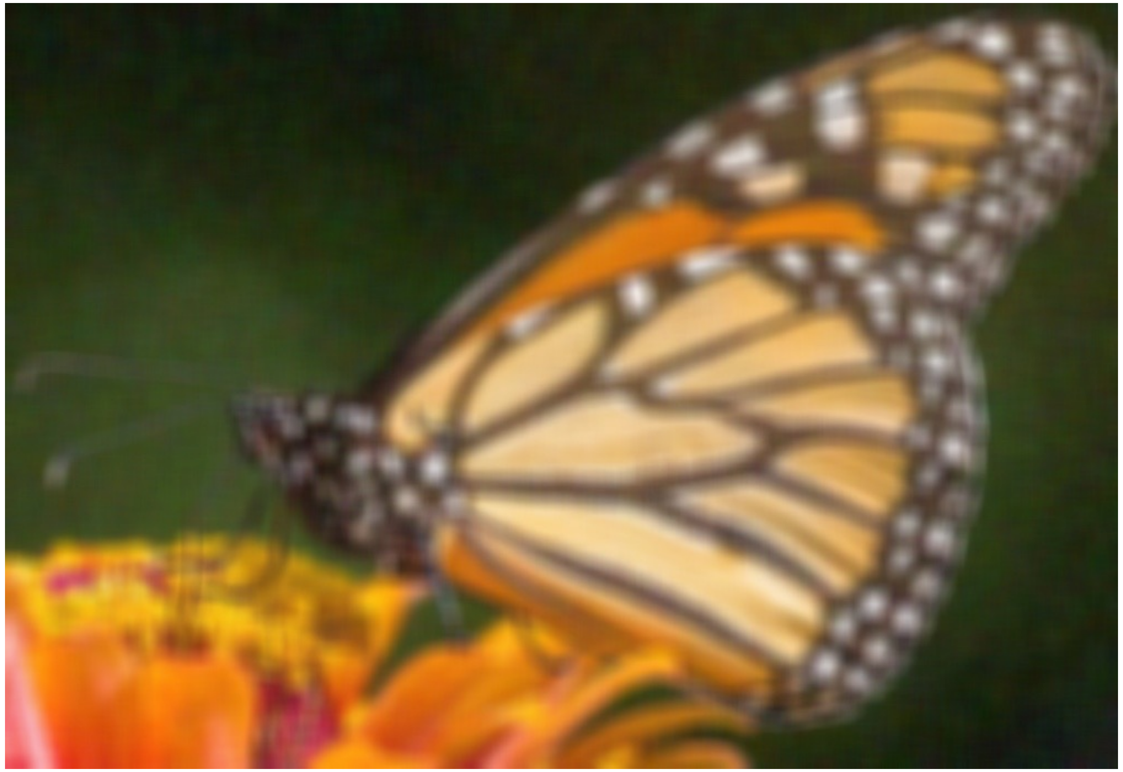
Image Filtering Process

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



Filtered Image

Apply the filter to every pixel



Filtered Image

Noisy Image

Image Filtering

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

10	5	3
4	5	1
1	1	7

Local image data

Some function →

	7	

Modified image data

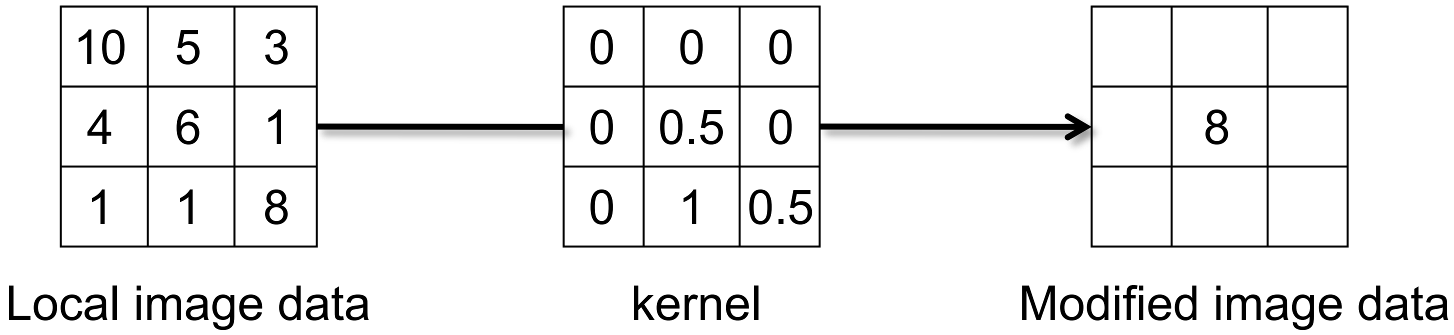
Slide credit: N. Snavely

Linear filtering

A simple 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”)



Slide credit: N. Snavely

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 a **cross-correlation** operation: $G = H \otimes F$

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

Slide credit: N. Snavely

Convolution

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

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

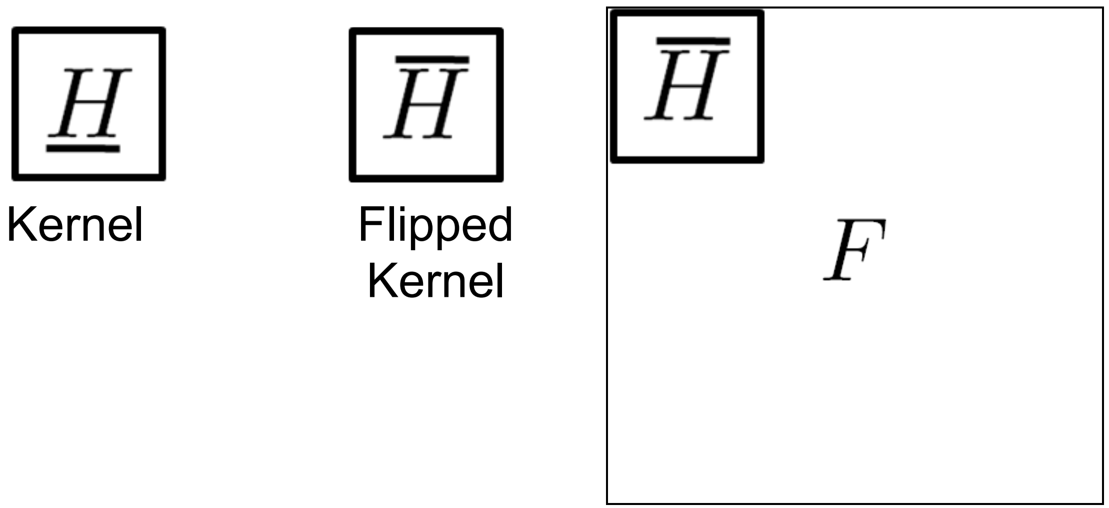
This is called a **convolution** operation:

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

$$G = H * F$$

Slide credit: N. Snavely

Convolution



Adapted from F. Durand

Mean filtering

H $*$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	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

F

=

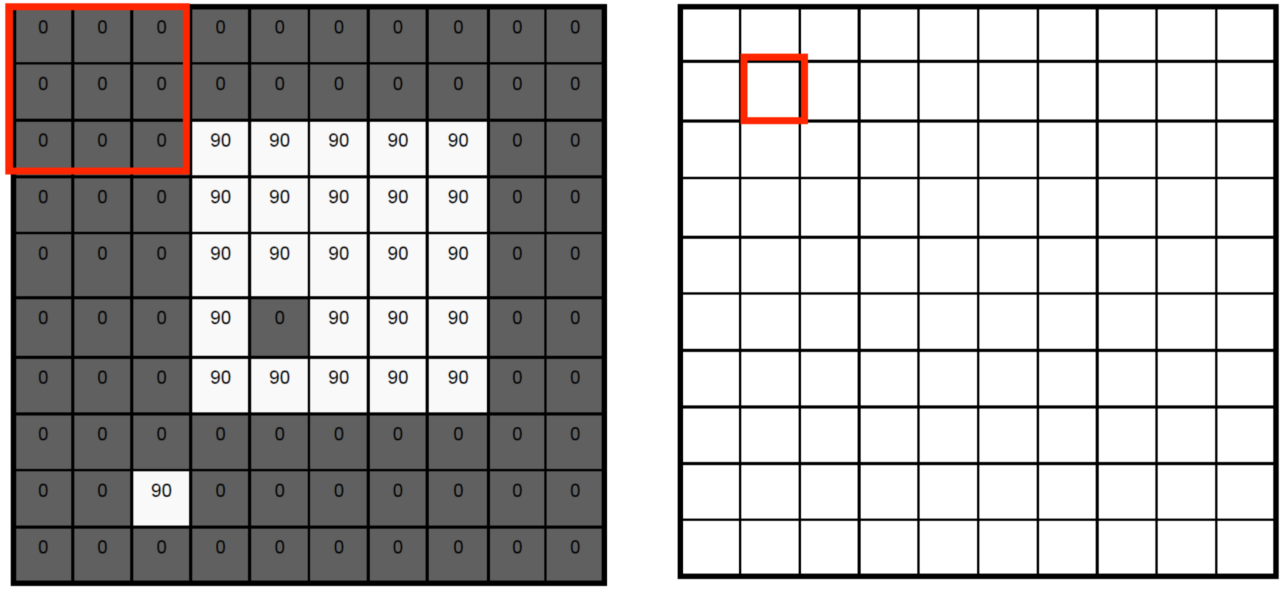
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

G

Slide credit: N. Snavely

Mean filtering/Moving average

$F[x, y]$ $G[x, y]$



Slide credit: N. Snavely

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							

Slide credit: N. Snavely

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						

Slide credit: N. Snavely

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					

Slide credit: N. Snavely

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				

Slide credit: N. Snavely

Mean filtering/Moving average

$F[x, y]$

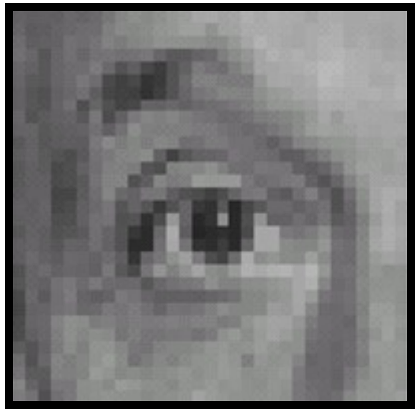
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	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	

Slide credit: N. Snavely

Linear filters: examples



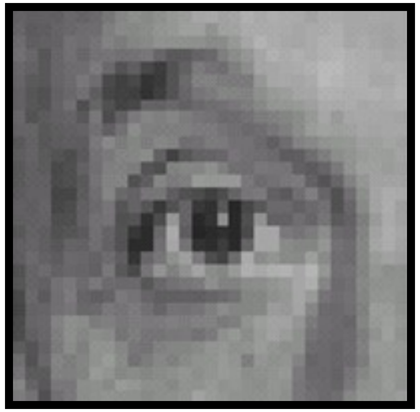
Original

$*$

0	0	0
0	1	0
0	0	0

Source: D. Lowe

Linear filters: examples

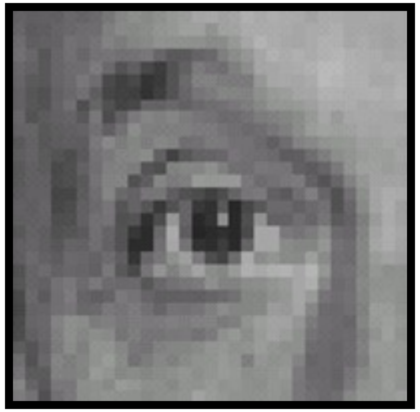


Original

$*$

0	0	0
0	1	0
0	0	0

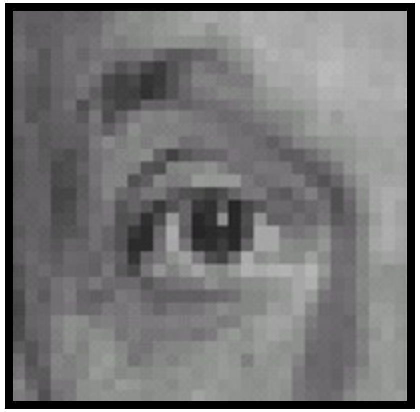
$=$



Identical image

Source: D. Lowe

Linear filters: examples



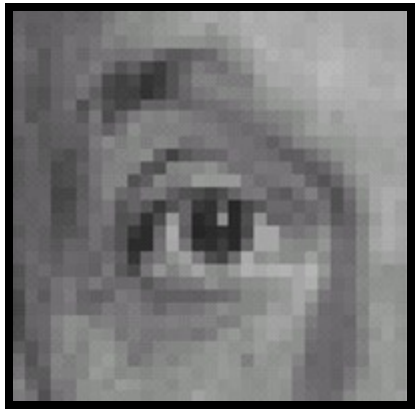
Original

*

0	0	0
1	0	0
0	0	0

Source: D. Lowe

Linear filters: examples

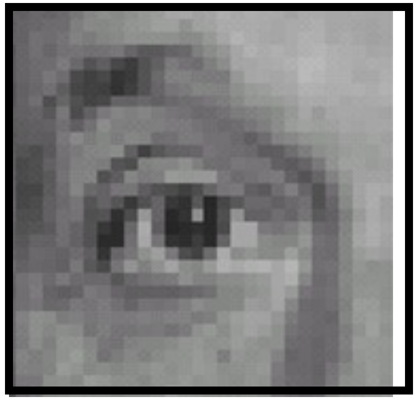


Original

*

0	0	0
1	0	0
0	0	0

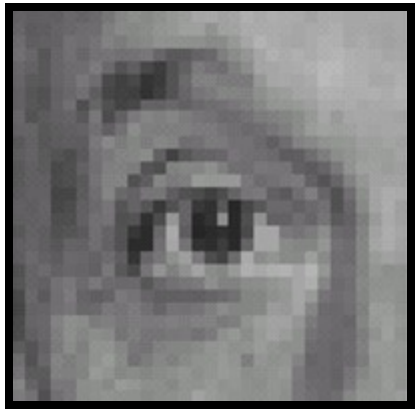
=



Shifted left by 1 pixel

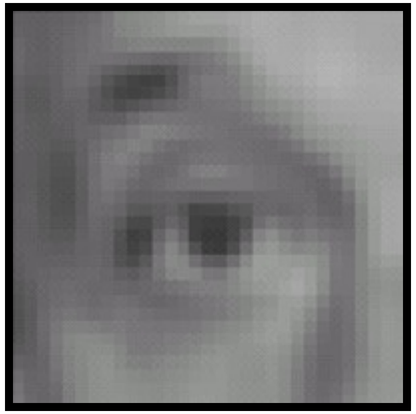
Source: D. Lowe

Linear filters: examples



Original

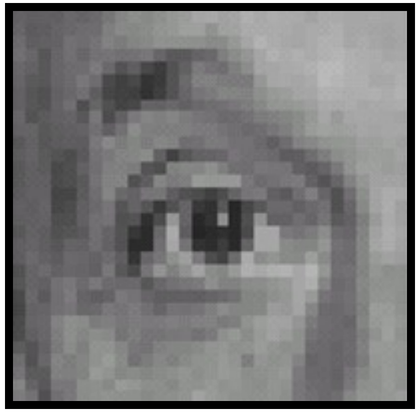
$$* \ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} =$$



Blur (with a mean/box filter)

Source: D. Lowe

Linear filters: examples



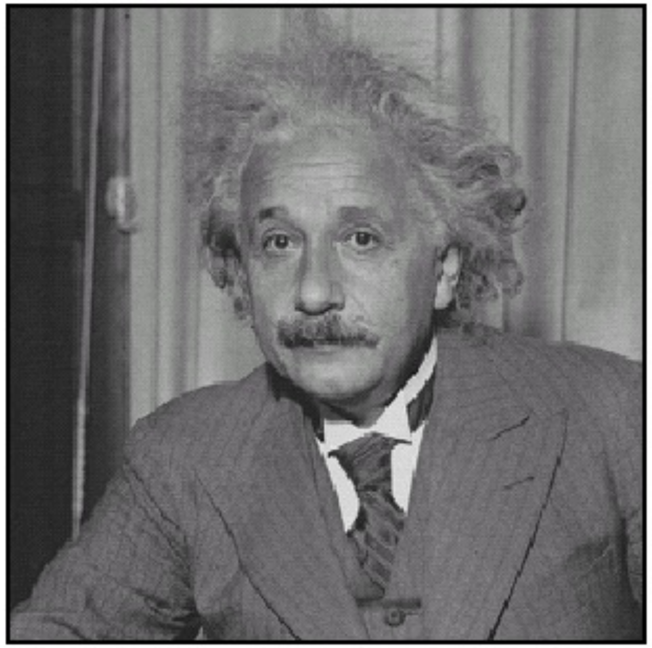
Original

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

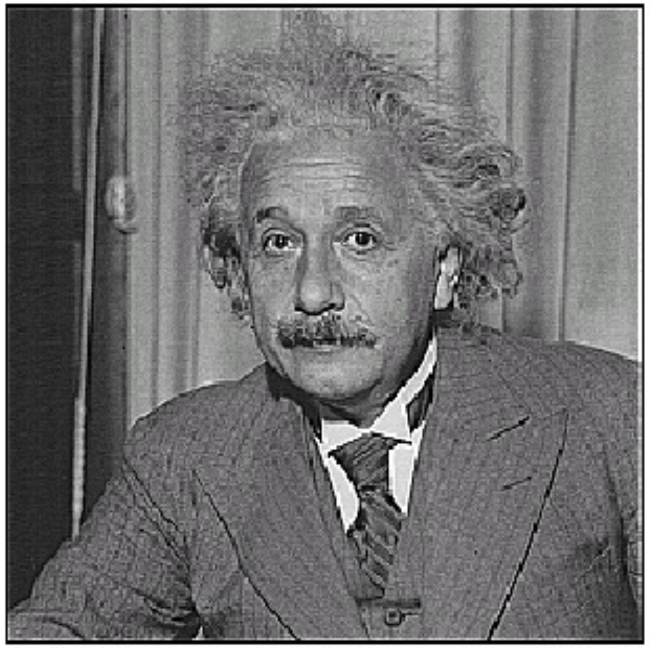
Sharpening filter

Source: D. Lowe

Sharpening



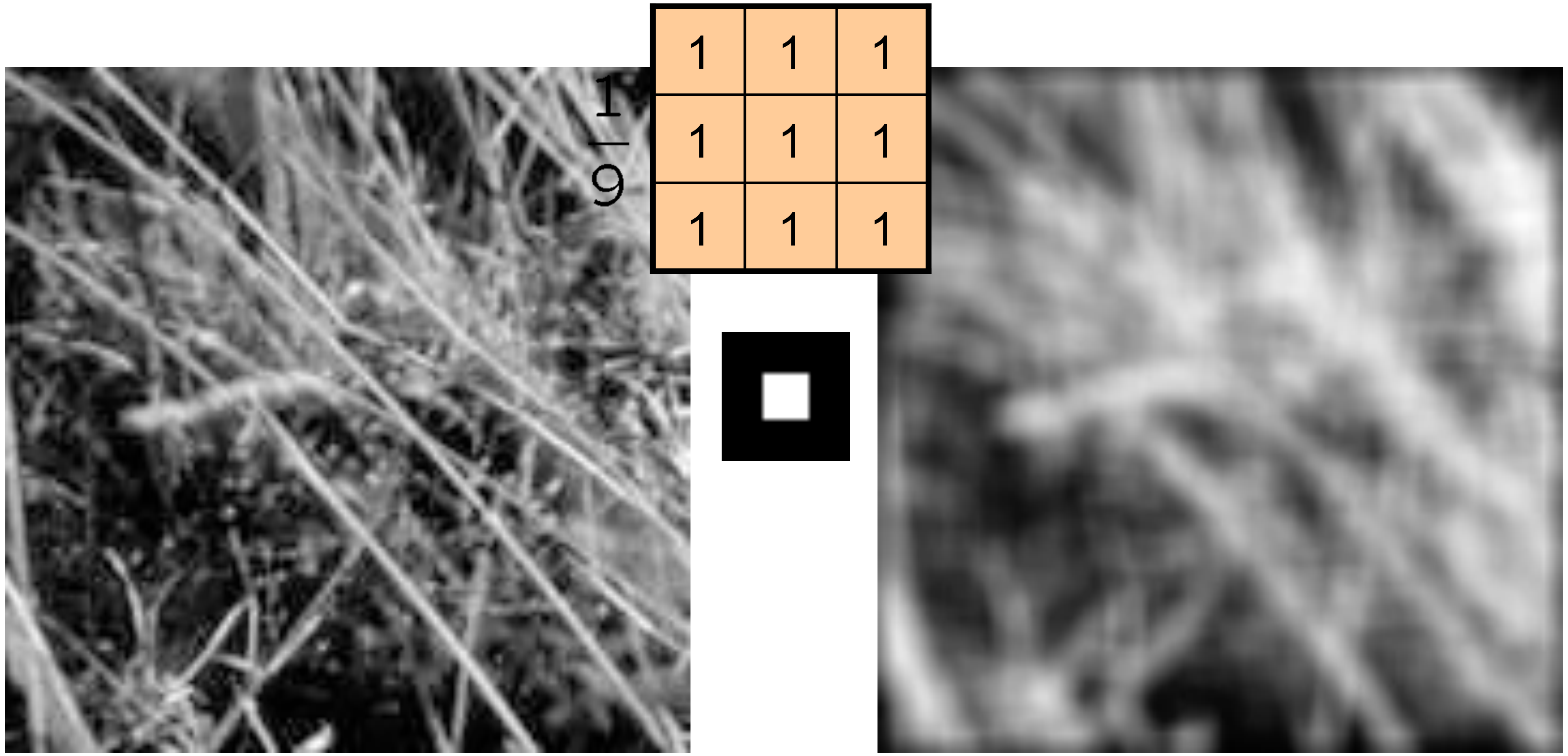
before



after

Source: D. Lowe

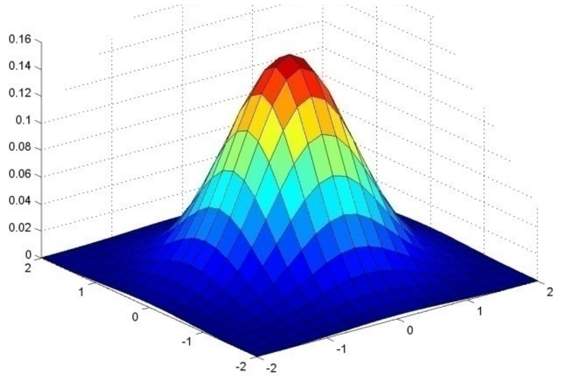
Smoothing with mean filter revisited

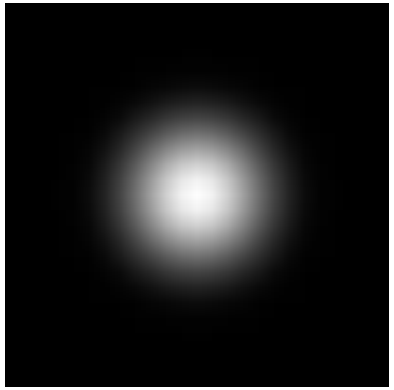


Block artifacts appear in the outputted image because non-relevant pixels are assigned the same weights during filtering

Source: D. Forsyth

Gaussian kernel



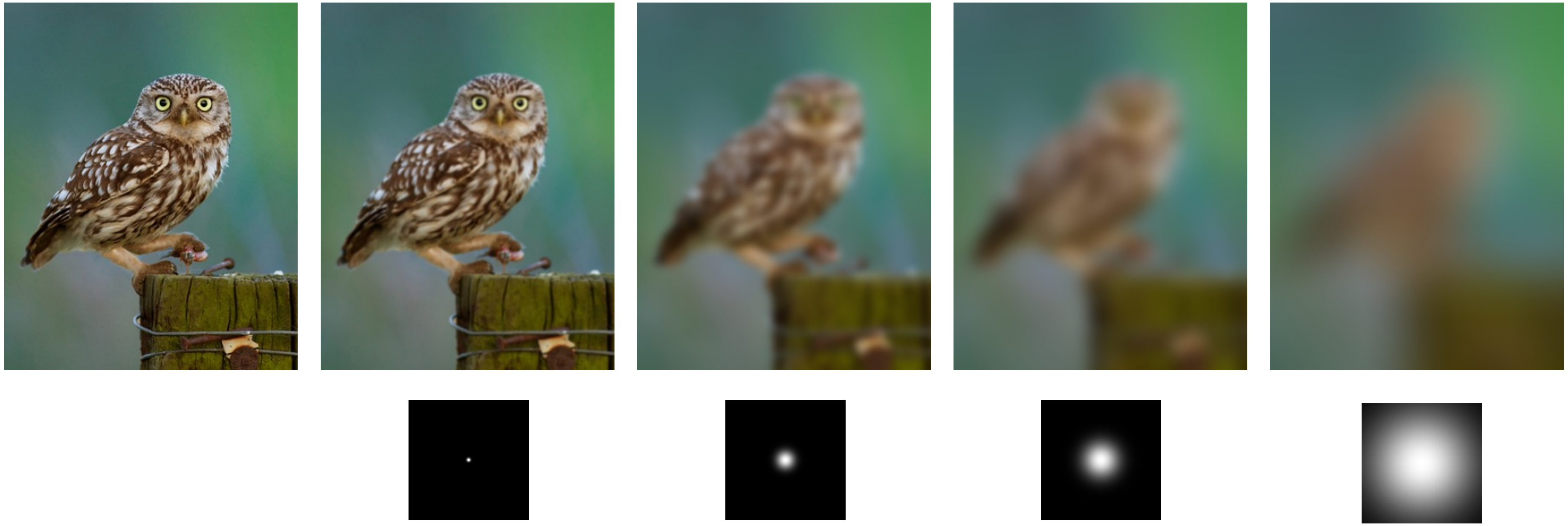


$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- If a neighboring pixel is closer to the current pixel, it will be assigned a larger weight
- The σ controls the width of the kernel

Source: C. Rasmussen

Gaussian filters

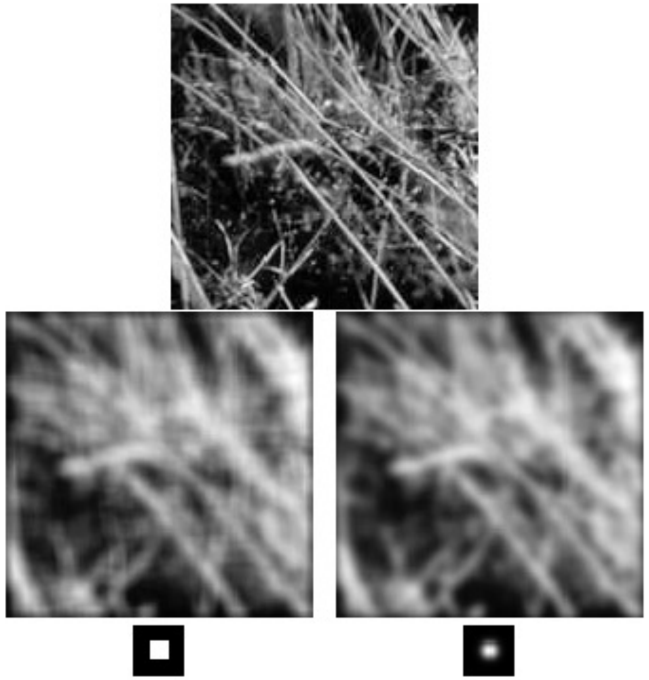


σ = 1 pixel σ = 5 pixels σ = 10 pixels σ = 30 pixels

A Gaussian filter with a larger σ will produce a more blurred image

Slide credit: N. Snavely

Mean vs. Gaussian filtering



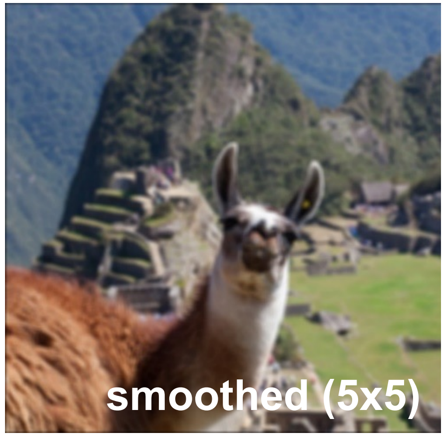
Both mean and Gaussian utilize local smoothness prior

- Mean filter assumes all pixels in a local window are equally important
- Gaussian filter assumes pixels that are closer to the target pixel are more important

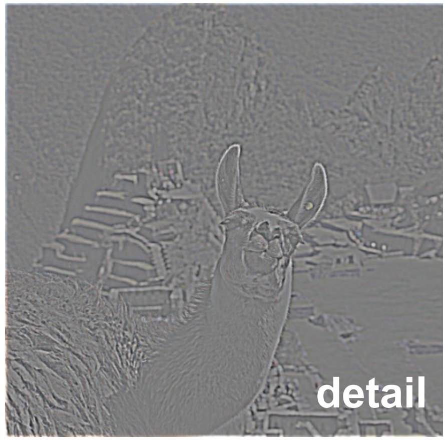
Source: N. Snavely

Sharpening revisited: What does blurring take away?

–



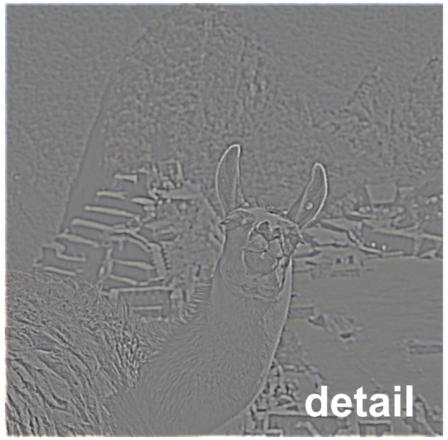
=



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

Let's add it back:

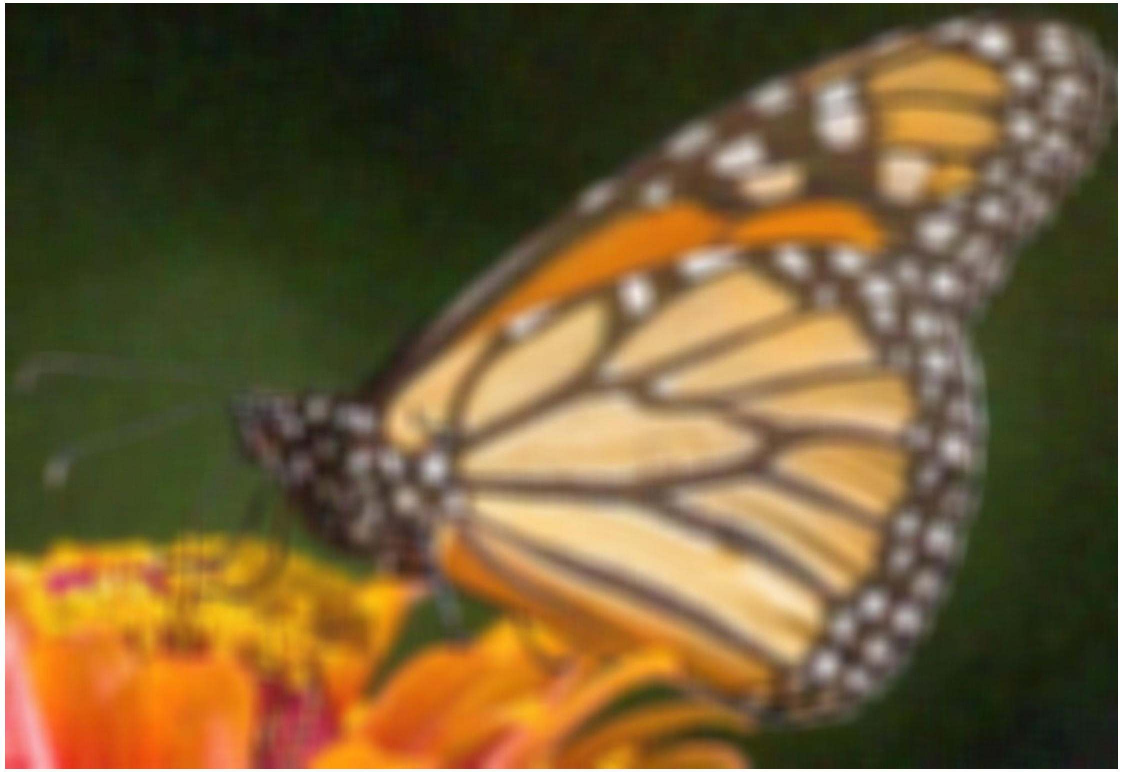
+ α



=

Slide credit: N. Snavely

Filtered Image



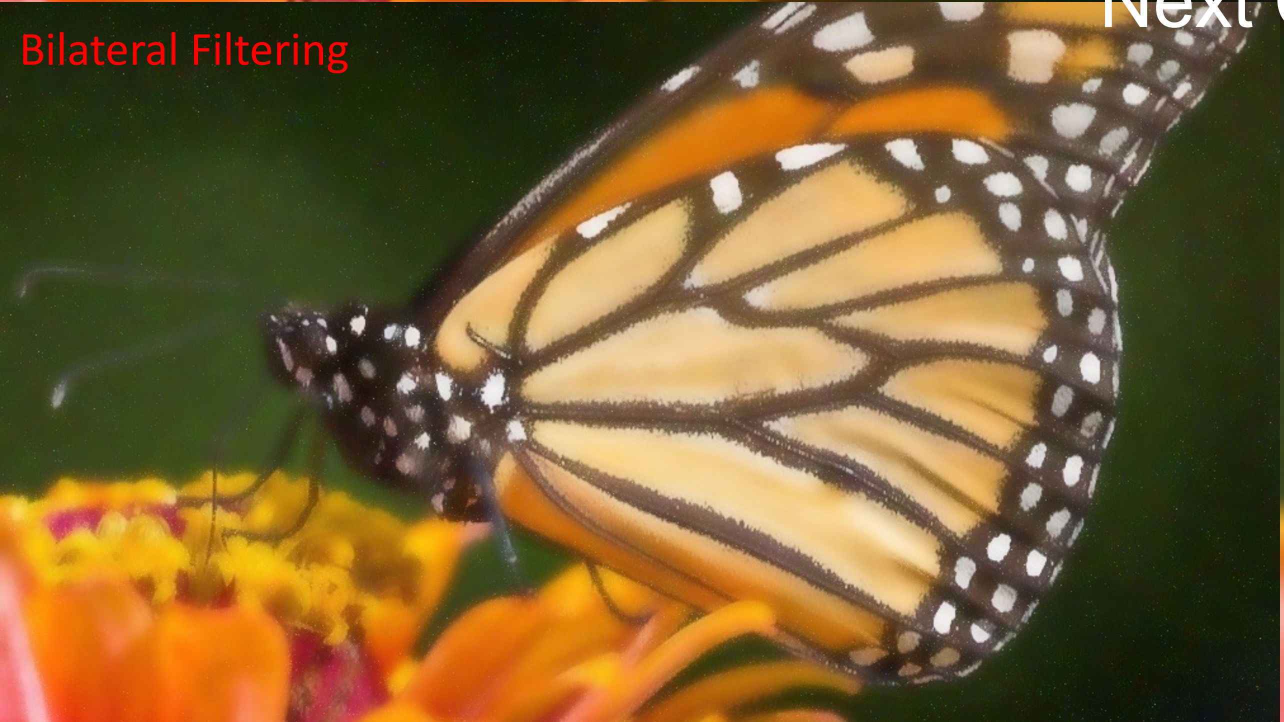
Noisy Image

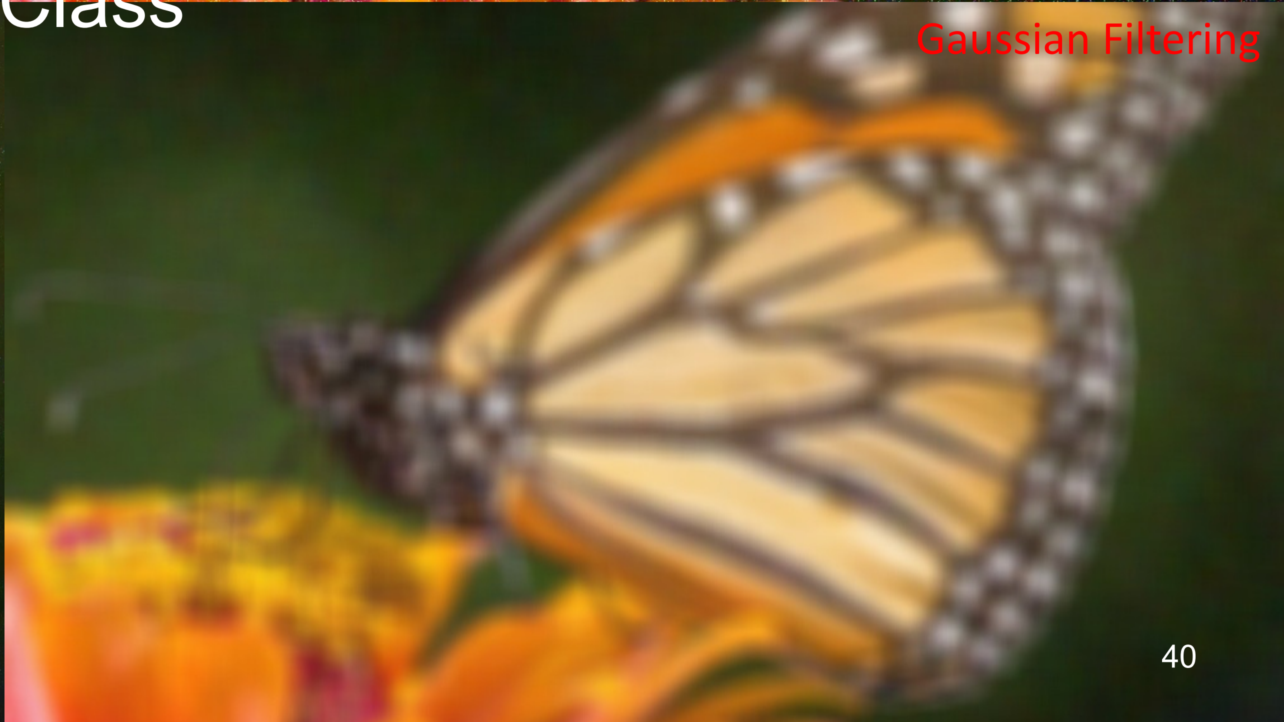
Question: How to handle blurry artifacts and preserve high-frequency details in the filtered image?

Next Class


Nonlocal Means Filtering


Noisy Image


Bilateral Filtering


Gaussian Filtering

Further Reading

Chapters 3.1 and 3.2, Computer Vision: Algorithms and Applications, Richard Szeliski