Computer Vision

Image Filtering

Dr. A U G Sankararao

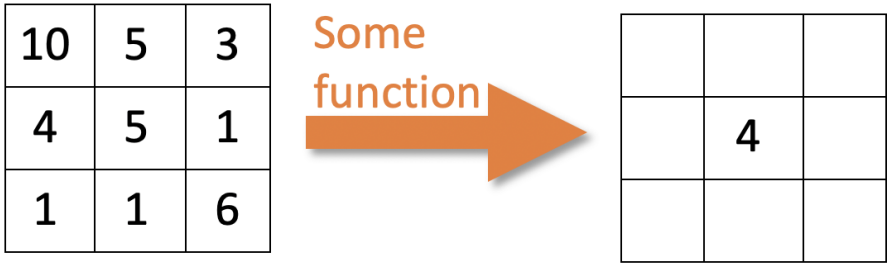
Indian Institute of Information Technology
Sri City, Chittoor

Today's Agenda

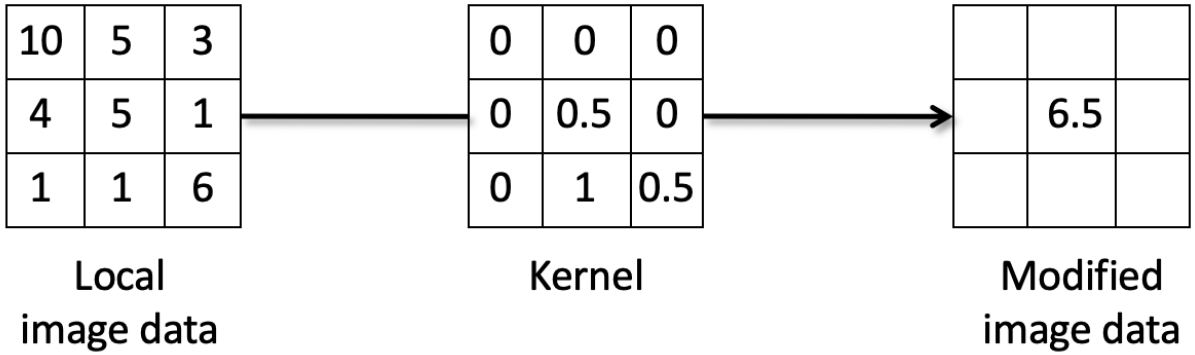
- Image Filtering
  - Smoothing
  - Intro to Edge Filtering
  - Sharpening

Linear Filter

- **Image Filter**: Modify image pixels based on some function of a local neighbourhood of each pixel



- **Linear Filter:** Replace each pixel by linear combination (a weighted sum) of Neighbours
- Linear combination called kernel, mask or filter



Linear Filter: Cross Correlation

Given a kernel of size $(2k+1) \times (2k+1)$:

- **Correlation** defined as:

$$G(i,j) = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u, j+v)}_{\text{Loop over pixels in considered neighbourhood around } I(i,j)}$$

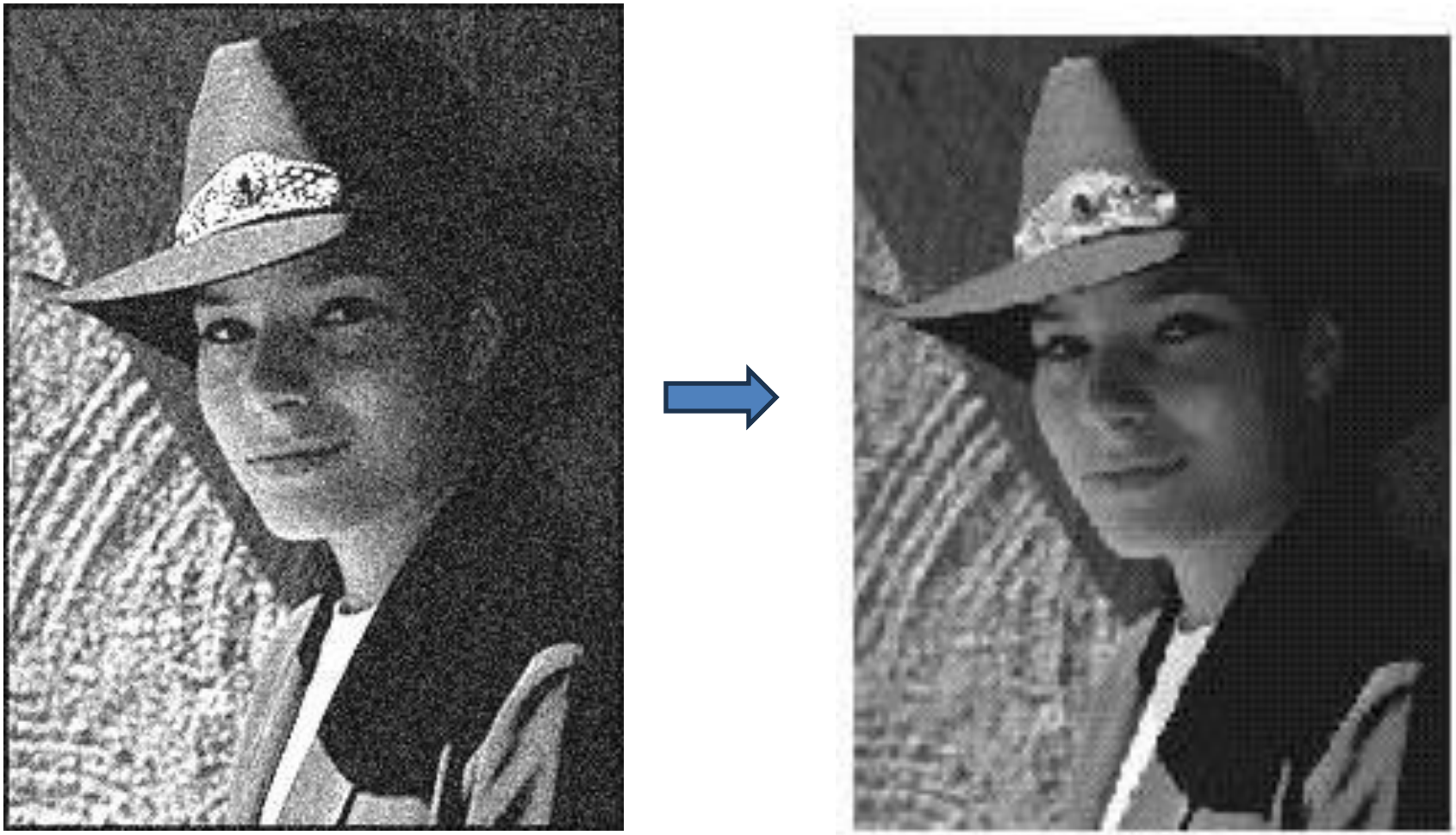
- **Cross-correlation** defined as:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H(u,v)}_{\text{Non-uniform weights}} I(i+u, j+v)$$

- Cross-correlation denoted by $G = H \otimes I$
- Can be viewed as "dot product" between local neighbourhood and kernel for each pixel
- Entries of kernel or mask $H(u,v)$ called filter co-efficients

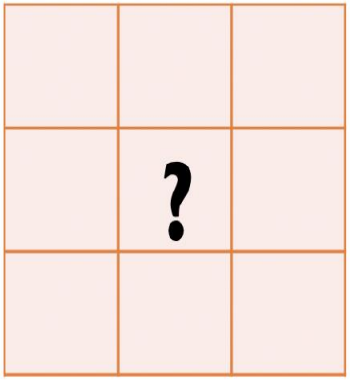
Image denoising

- How can we reduce noise in a photograph?



Moving average

- Let's replace each pixel with a *weighted average* of its neighborhood
- What are the weights for the average of a 3x3 neighborhood?



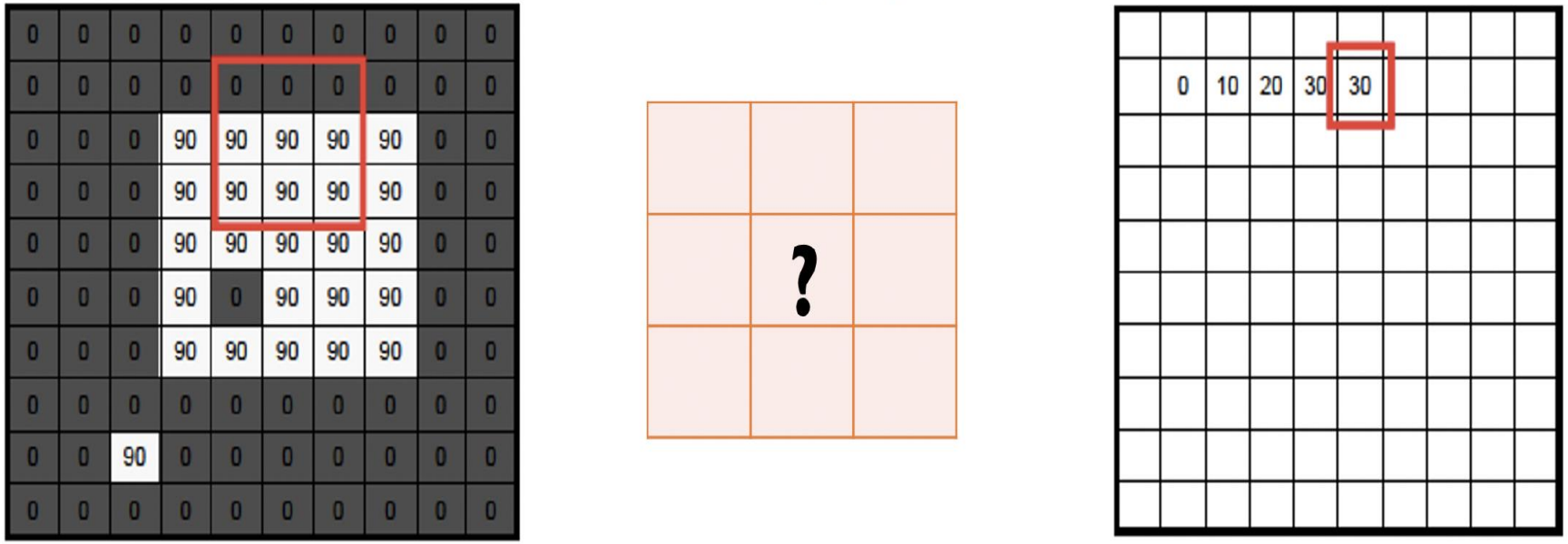
"box filter"

Source: D. Lowe

Linear Filter: Moving Average

What values belong in the kernel H for the moving average example we saw earlier?

$I(i,j)$ $\otimes$ $H(u,v)$ $=$ $G(i,j)$

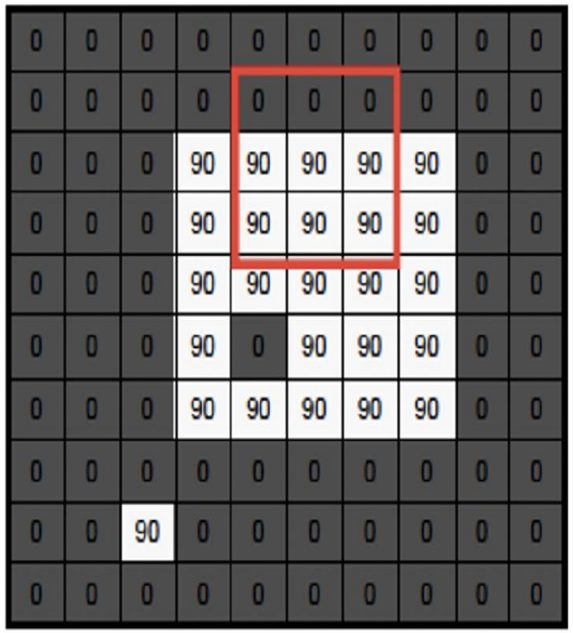


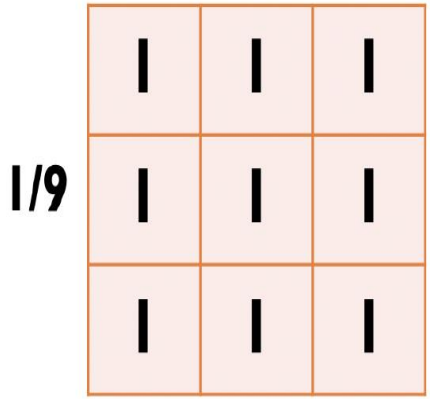
Credit: K Grauman, Univ of Texas Austin

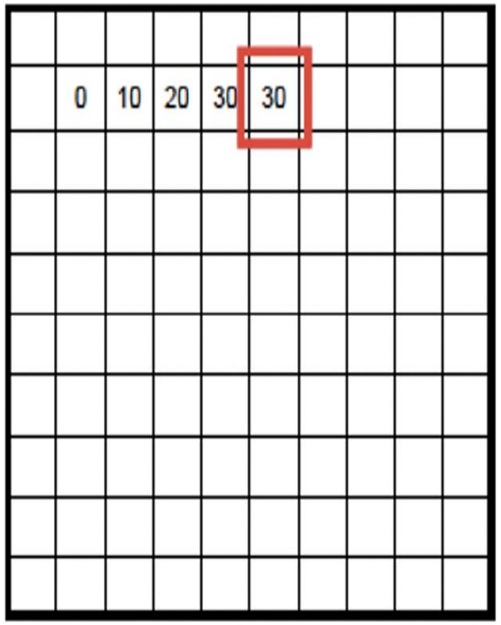
Linear Filter: Moving Average

What values belong in the kernel H for the moving average example we saw earlier?

$$I(i,j) \quad \otimes \quad H(u,v) \quad = \quad G(i,j)$$







Credit: K Grauman, Univ of Texas Austin

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

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

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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							

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

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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						

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

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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					

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

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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				

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

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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				
				?					

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

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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				
						?			
				50					

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

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

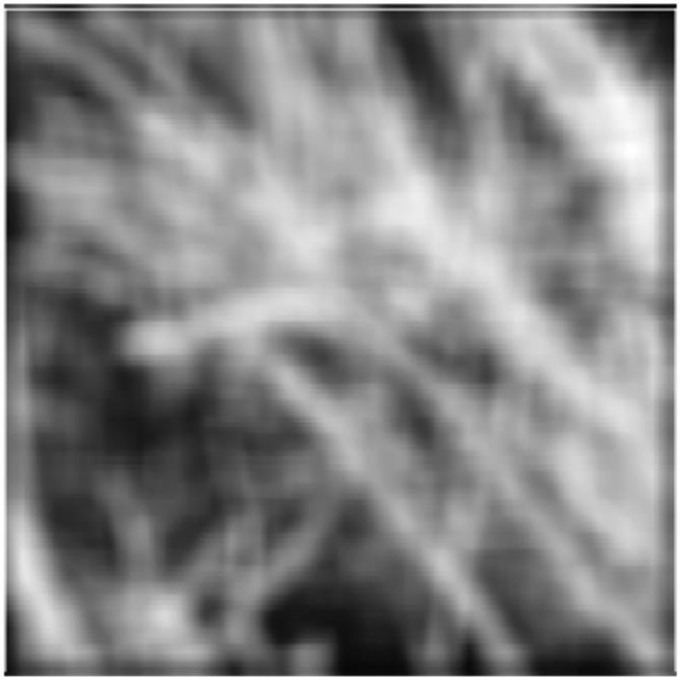
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

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

Credit: S. Seitz

Moving Average Filter

- Effect of moving average filter (also known as box filter):



- What's wrong with this picture?
- What's the solution?

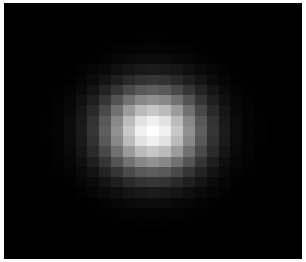
Credit: K Grauman, Univ of Texas Austin

Gaussian Average Filter

- What if we want nearest neighbouring pixels to have the most influence on the output?

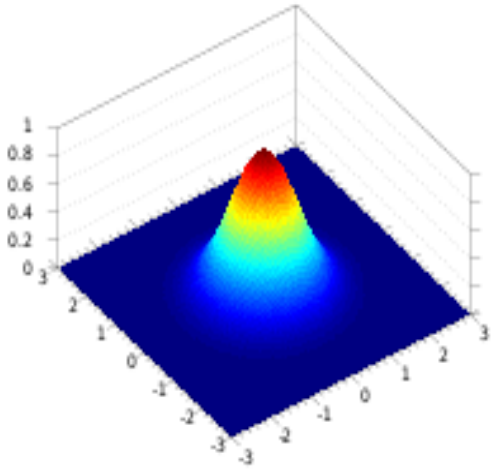
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



This kernel is an approximation of a 2D Gaussian function:

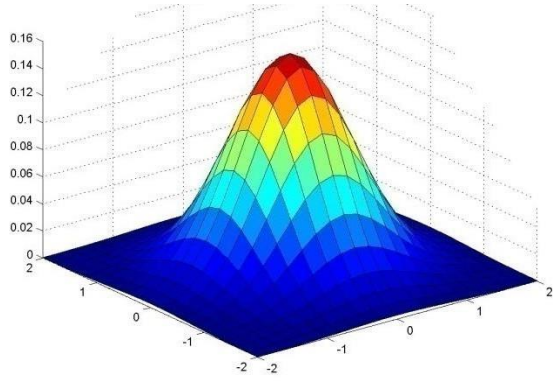
$$h(u,v) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{u^2+v^2}{\sigma^2}}$$

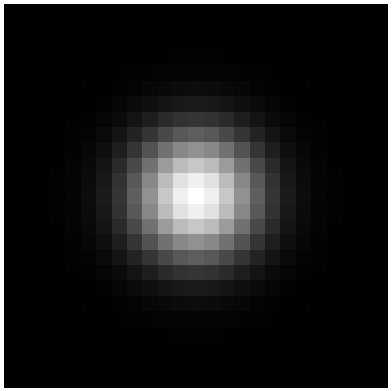


Credit: K Grauman, Univ of Texas Austin

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

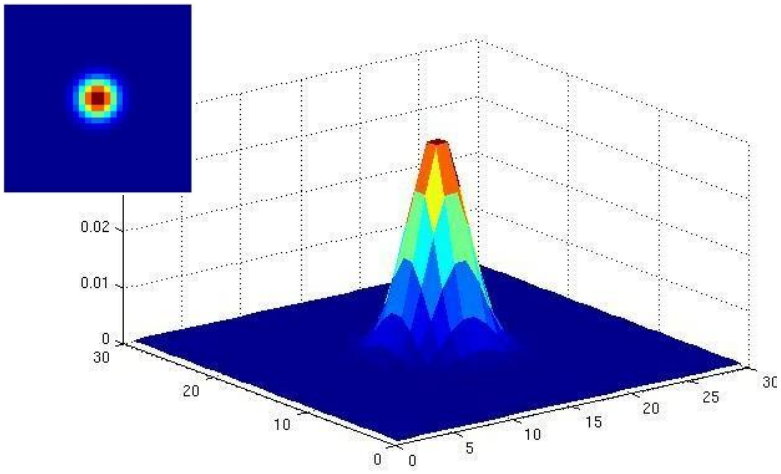
5 x 5, $\sigma = 1$

- Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should renormalize weights to sum to 1 in any case)

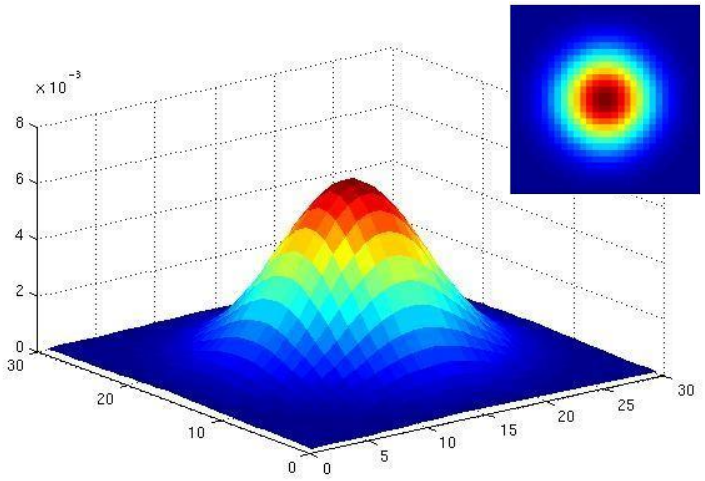
Source: C. Rasmussen

Gaussian Kernel

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



σ = 2 with 30 x 30 kernel



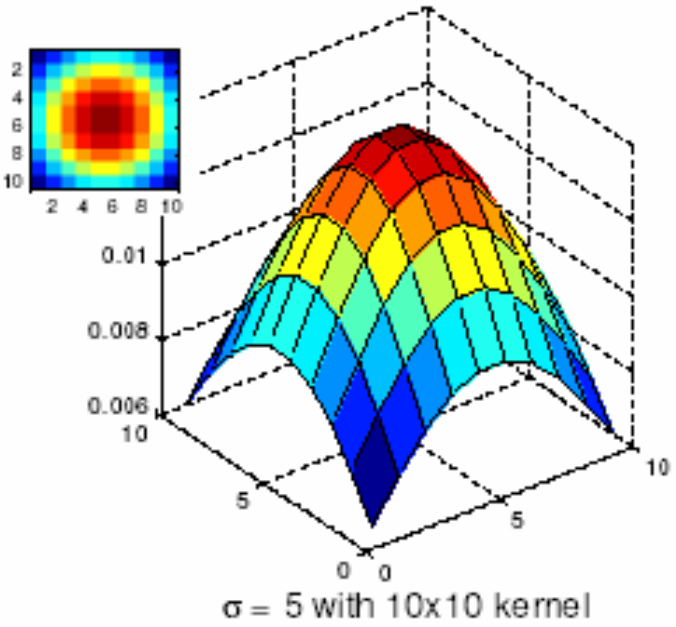
σ = 5 with 30 x 30 kernel

- Standard deviation σ: determines extent of smoothing

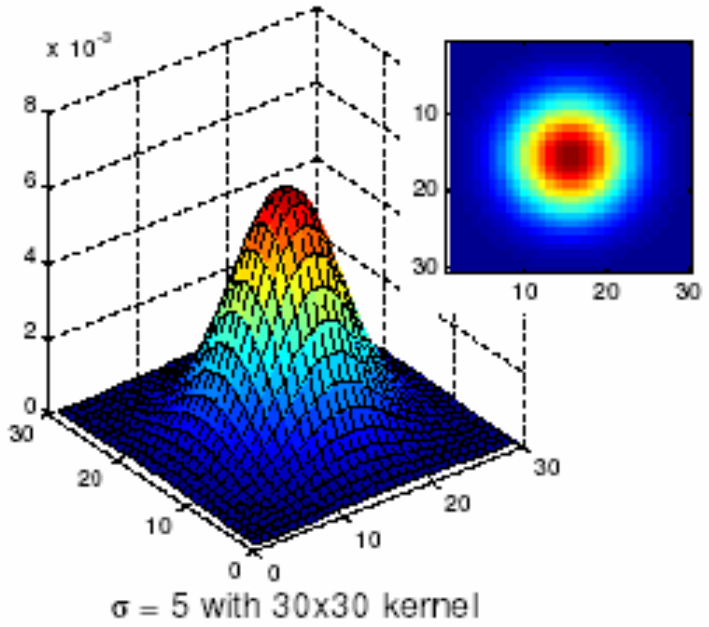
Source: K. Grauman

Choosing kernel width

- The Gaussian function has infinite support, but discrete filters use finite kernels



σ = 5 with 10x10 kernel

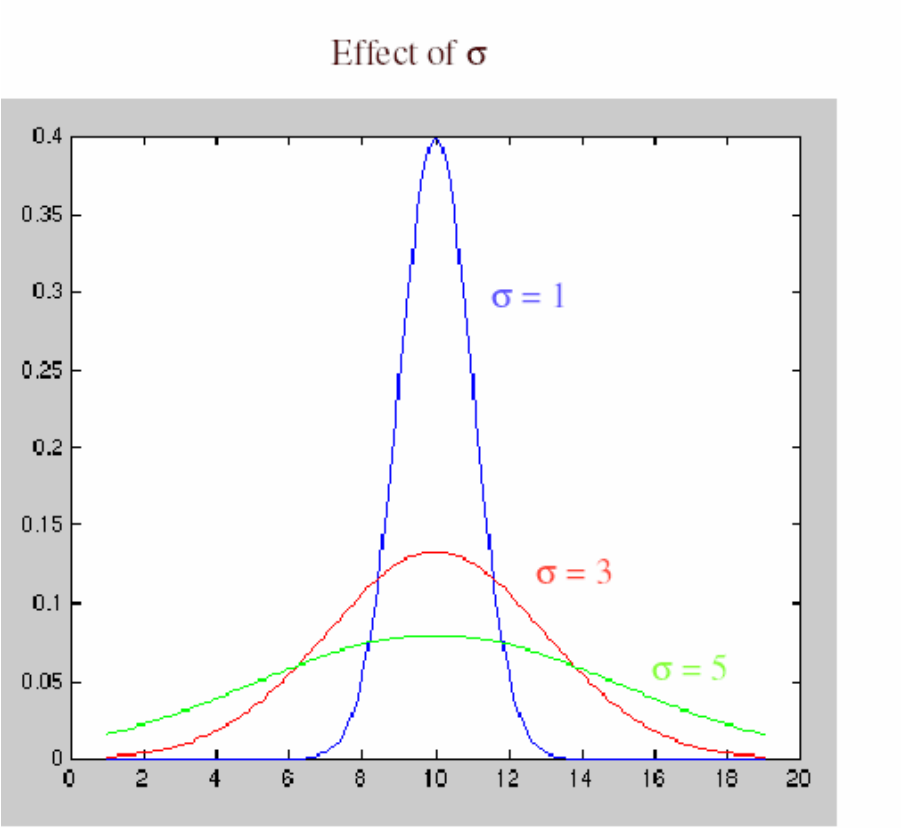


σ = 5 with 30x30 kernel

Source: K. Grauman

Choosing kernel width

- Rule of thumb: set filter half-width to about 3σ



Gaussian filters

- Remove high-frequency components from the image (*low-pass filter*)
- Convolution with self is another Gaussian
  - So can smooth with small-σ kernel, repeat, and get same result as larger-σ kernel would have
  - Convolving two times with Gaussian kernel with std. dev. σ is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians
  - Discrete example:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Source: K. Grauman

Separability of the Gaussian filter

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

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

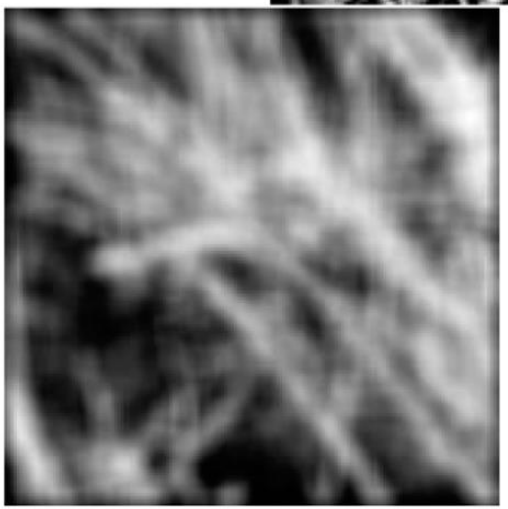
Source: D. Lowe

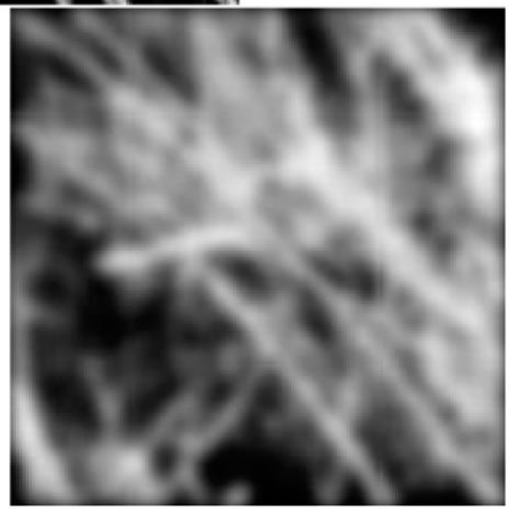
Why is separability useful?

- A 2D convolution can be reduced to two 1D convolutions (one along rows and one along columns)
- What is the complexity of filtering an $n \times n$ image with an $k \times k$ kernel?
  - $O(n^2 k^2)$
- What if the kernel is separable?
  - $O(n^2 k)$

Average Filters: A Comparison

Box filter





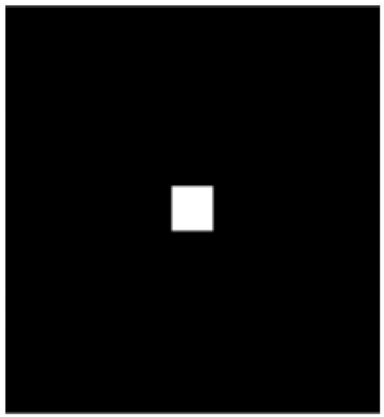
Gaussian filter

Credit: K Grauman, Univ of Texas Austin

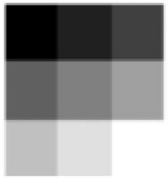
Beyond Correlation: Impulse Signal

- *What is the result of filtering the impulse signal (image) I with the arbitrary kernel H?*

$I(i,j)$



$\otimes \quad H(u,v)$



a	b	c
d	e	f
g	h	i

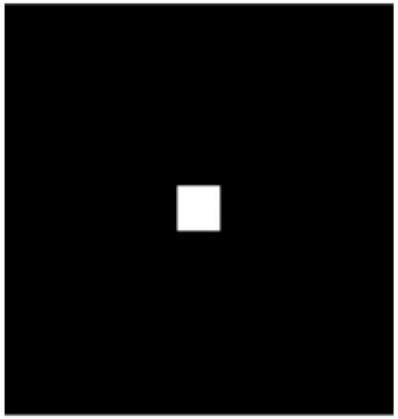
$G(i,j)$

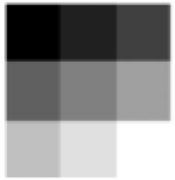
Credit: KivwWorker

Beyond Correlation: Impulse Signal

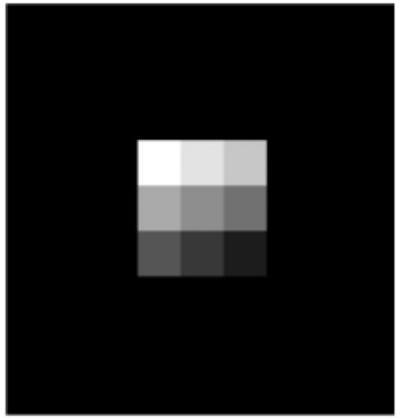
- *What is the result of filtering the impulse signal (image) I with the arbitrary kernel H?*

$I(i,j)$ $\otimes$ $H(u,v)$ $G(i,j)$





a	b	c
d	e	f
g	h	i



!	ч	9
ɟ	ə	d
ɔ	q	ɐ

- **Cross-correlation** defined as:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H(u,v)}_{\text{Non-uniform weights}} I(i+u, j+v)$$

Credit: KivwWorker

Practice with linear filters

- *What is the size of the output image ?*

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

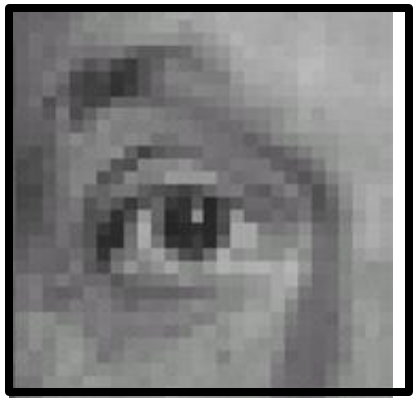
?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0



Shifted *left*
By 1 pixel

Source: D. Lowe

Practice with linear filters

Original

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

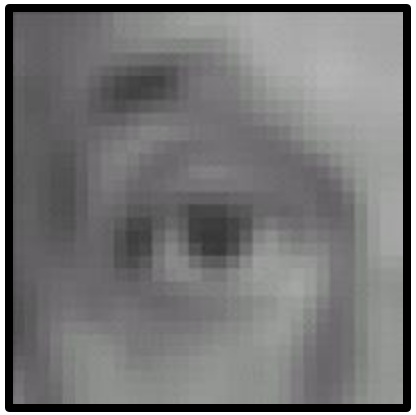
?

Source: D. Lowe

Practice with linear filters

Original

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



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters

Original

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

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters

Original

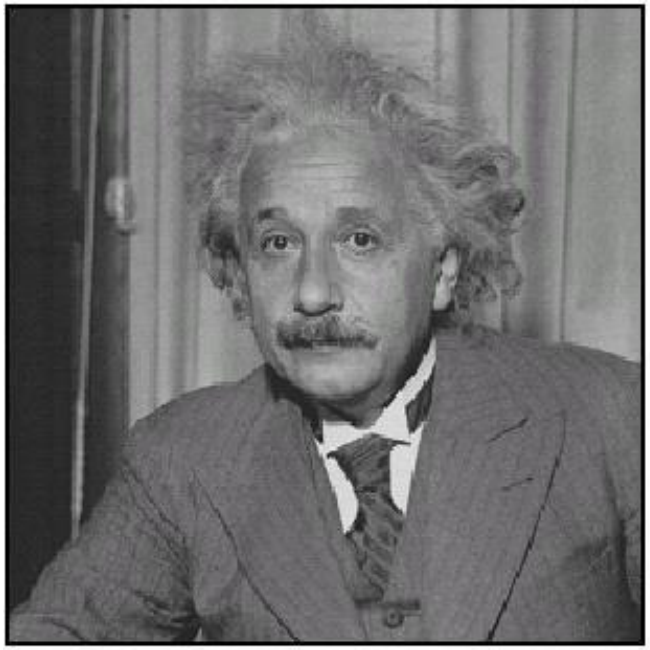
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

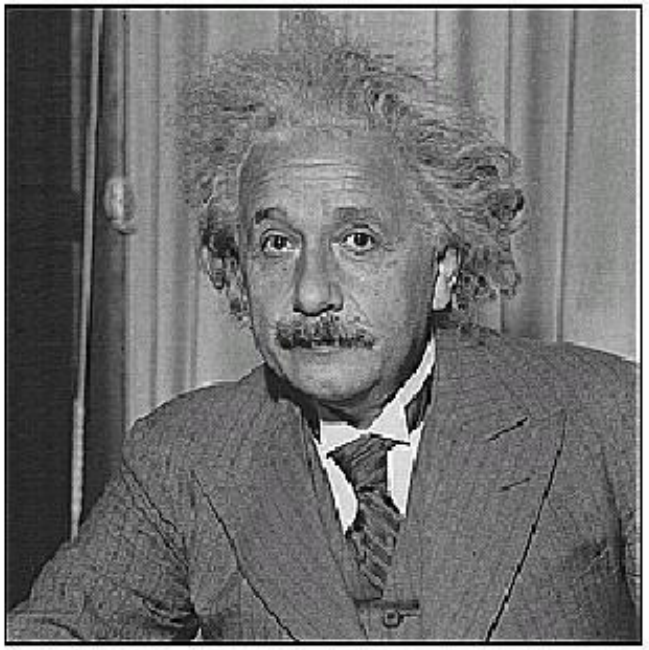
Source: D. Lowe

Sharpening

- Sharpening enhance the clarity and definition of edges and fine details in an image



before



after

Source: D. Lowe

Sharpening

- What does blurring take away?

–

=

Let's add it back:

+

=

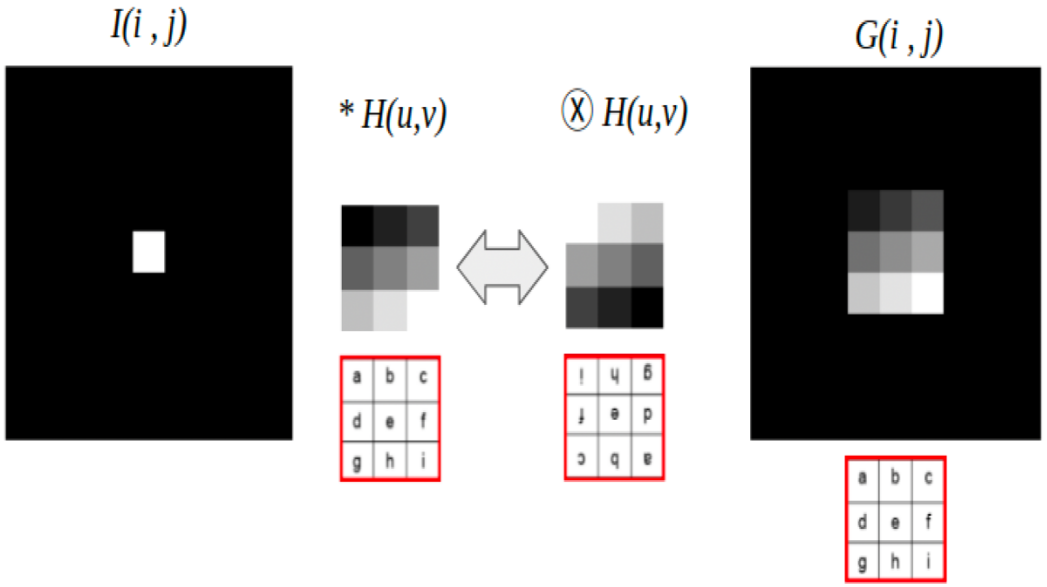
Convolution

Given a kernel of size $(2k+1) \times (2k+1)$:

- **Convolution** defined as:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v) I(i-u, j-v)$$

- Equivalent to flip the filter in both directions (bottom to top, right to left) and apply cross-correlation
- Denoted by $G = H * I$

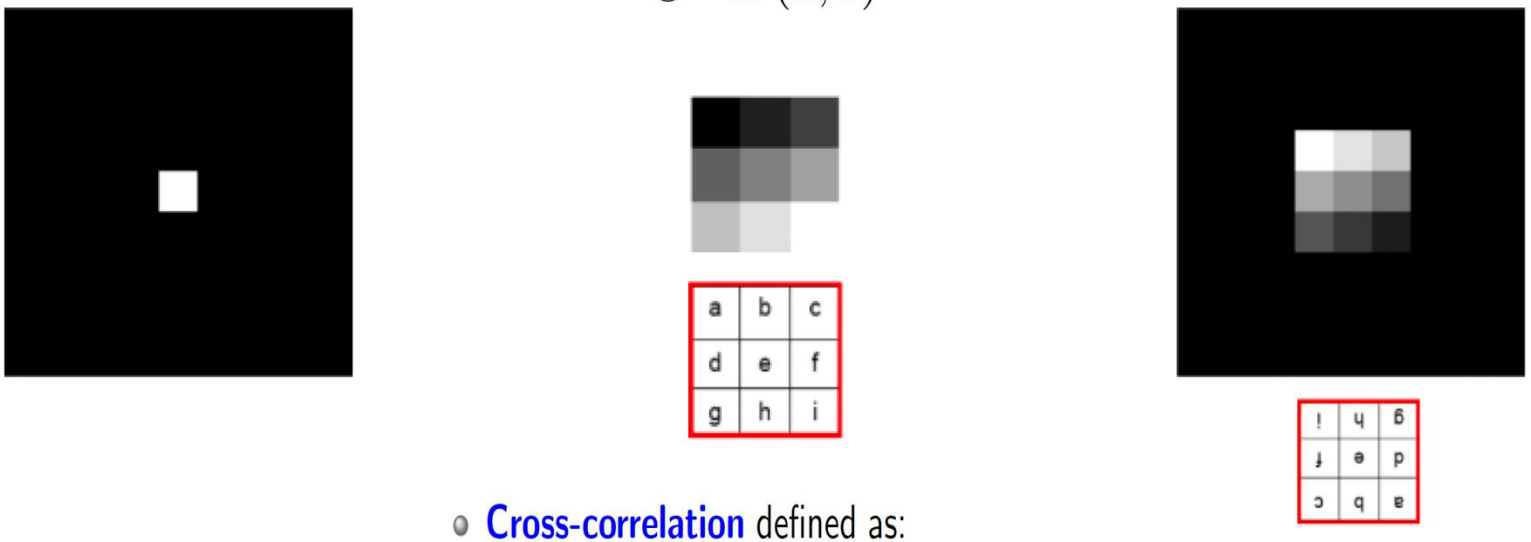


Credit: Vineet N

Correlation

- *What is the result of filtering the impulse signal (image) I with the arbitrary kernel H?*

$I(i,j)$ $\otimes$ $H(u,v)$ $G(i,j)$



- **Cross-correlation** defined as:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H(u,v)}_{\text{Non-uniform weights}} I(i+u, j+v)$$

Credit: KivwWorker

Linear Shift-Invariant Operators

- Both correlation and convolution are Linear Shift-Invariant operators, which obey:
  - Linearity (or Superposition principle):

    $I \circ (h_0 + h_1) = I \circ h_0 + I \circ h_1$
  - Shift-Invariance: shifting (or translating) a signal commutes with applying the operator

    $g(i ; j) = h(i + k; j + l) \Leftrightarrow (f \circ g)(i, j) = (f \circ h)(i + k; j + l)$

    filter(shift(*f*)) = shift(filter(*f*))

- Equivalent to saying that the effect of the operator is the same everywhere. Why do we need this in computer vision?

- Theoretical result: any linear shift-invariant operator can be represented as a convolution

Properties in more detail

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
- 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 cumulative filter: $a * (b_1 * b_2 * b_3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$

  -- We can combine the responses of a signal over two or more filters by combining the filters
- Scalars factor out: $ka * b = a * kb = k(a * b)$
- Identity: unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Separability

- Convolution operator requires k^2 operations per pixel, where k is the width (and height) of a convolution kernel.
- Can be costly. How can we reduce cost?
- For certain filters, can be sped up by performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring $2k$ operations $\implies$ convolution kernel is **separable**.
- $$K = vh^T$$

  where v, h are 1D kernels, and K is the 2D kernel

Example 1:

$$\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} \implies v = h = \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix}$$

Example 2:

$$\frac{1}{8}\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix} \implies v = \frac{1}{4}\begin{bmatrix}-1\\-2\\-1\end{bmatrix} \;\&\; h = \frac{1}{2}\begin{bmatrix}1\\0\\1\end{bmatrix}$$

Separable Convolution

How can we tell if a given kernel K is separable?

- Visual inspection
- Analytically, look at the Singular Value Decomposition (SVD), and if only one singular value is non-zero, then it is separable.

$$K = U\Sigma V^T = \sum_i \sigma_i u_i v_i^T$$

where $\Sigma = \text{diag}(\sigma_i)$

$\sqrt{\sigma_1}u_1$ and $\sqrt{\sigma_1}v_1$ are the vertical and horizontal kernels

Source: Raquel Urtasun, Univ of Toronto

Practical Issues

Ideal size for the filter?

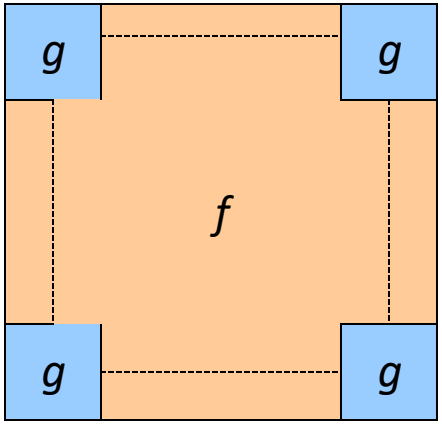
The bigger the mask:

- more neighbours contribute
- smaller noise variance of output
- bigger noise spread
- more blurring
- more expensive to compute

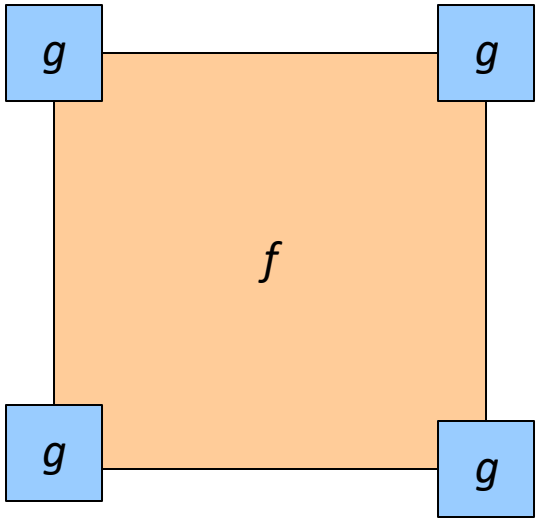
Dealing with edges

- If we convolve image *f* with filter *g*, what is the size of the output?

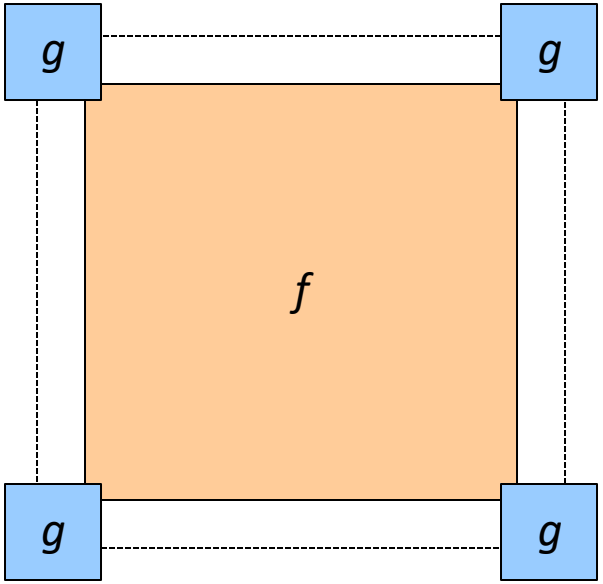
Output is smaller than input



Output is same size as input



Output is larger than input



Dealing with edges

- If the filter window falls off the edge of the image, we need to pad the image
  - Zero pad (or clip filter)
  - Wrap around
  - Copy edge
  - Reflect across edge

Source: S. Marschner

Dealing with edges

- If the filter window falls off the edge of the image, we need to pad the image
  - **Zero pad (or clip filter)**
  - Wrap around
  - Copy edge
  - Reflect across edge

Source: S. Marschner

Dealing with edges

- If the filter window falls off the edge of the image, we need to pad the image
  - Zero pad (or clip filter)
  - **Wrap around**
  - Copy edge
  - Reflect across edge

Source: S. Marschner

Dealing with edges

- If the filter window falls off the edge of the image, we need to pad the image
  - Zero pad (or clip filter)
  - Wrap around
  - **Copy edge**
  - Reflect across edge

Source: S. Marschner

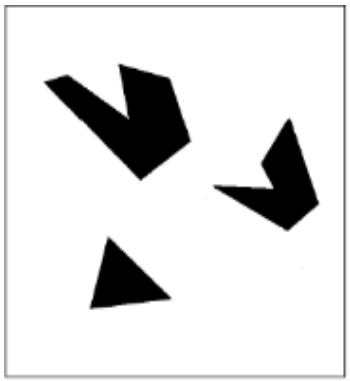
Dealing with edges

- If the filter window falls off the edge of the image, we need to pad the image
  - Zero pad (or clip filter)
  - Wrap around
  - Copy edge
  - **Reflect across edge**

Source: S. Marschner

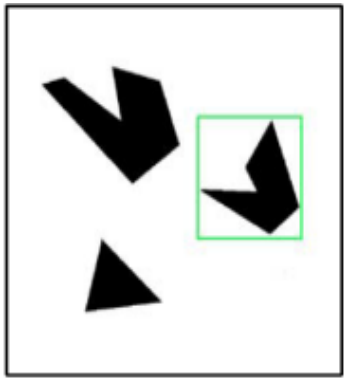
Do we need cross-correlation at all ?

- Can be used for Template Matching
- Filters look for like objects they are intended to find => use normalized cross-correlation (to control relative brightness) score to find a given pattern in an image

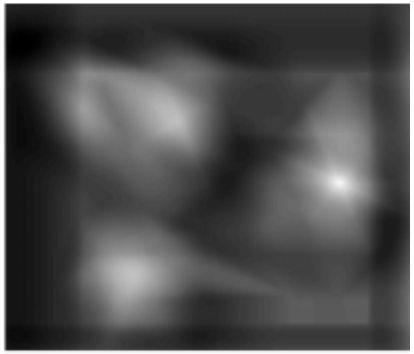


Scene

Template (mask)



Detected template



Correlation map

Credit: K Grauman, Univ of Texas Austin

Source: S. Marschner

Do we need cross-correlation at all ?

- Even if the template is not identical to some sub-image in the scene, match can be meaningful, if scale, orientation and general appearance is right.

Scene

Template

Detected template

Credit: K Grauman, Univ of Texas Austin

Source: S. Marschner

Noise

Original

Salt and pepper noise

Impulse noise

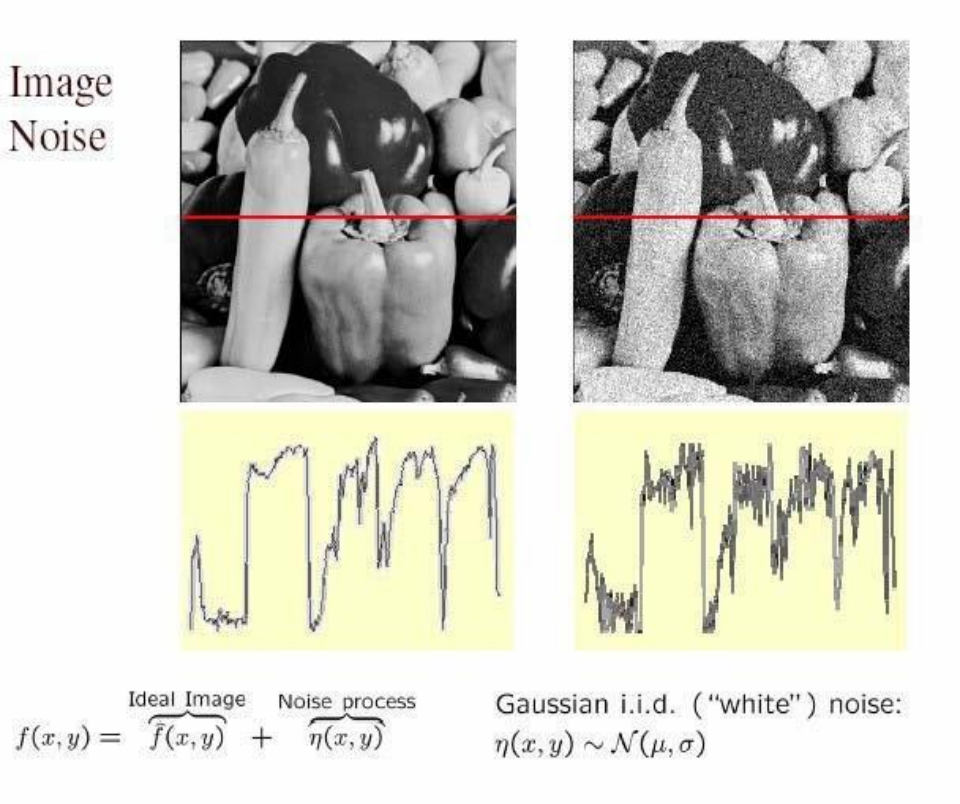
Gaussian noise

- **Salt and pepper noise**: contains random occurrences of black and white pixels
- **Impulse noise:** contains random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Source: S. Seitz

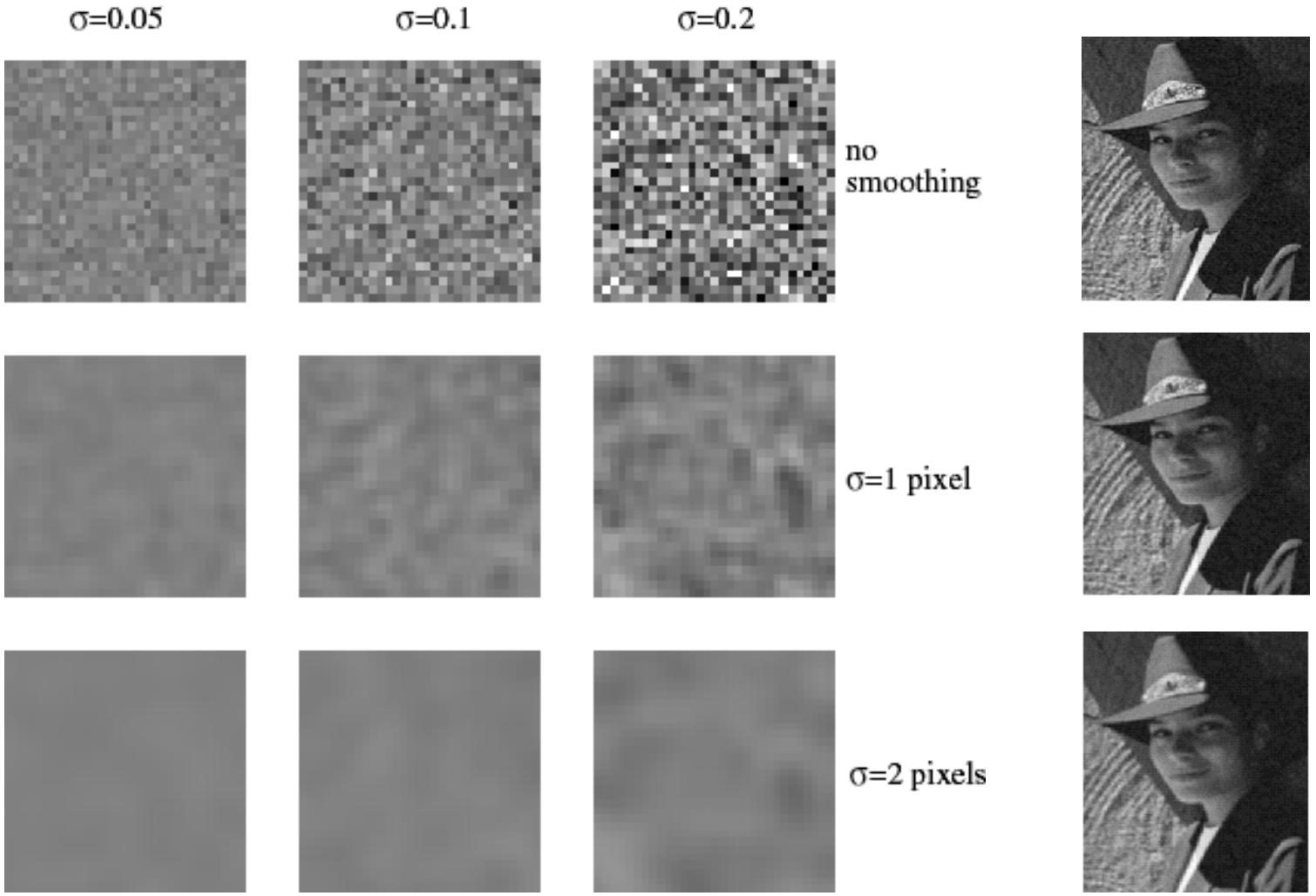
Gaussian noise

- Mathematical model: sum of many independent factors
- Good for small standard deviations
- Assumption: independent, zero-mean noise



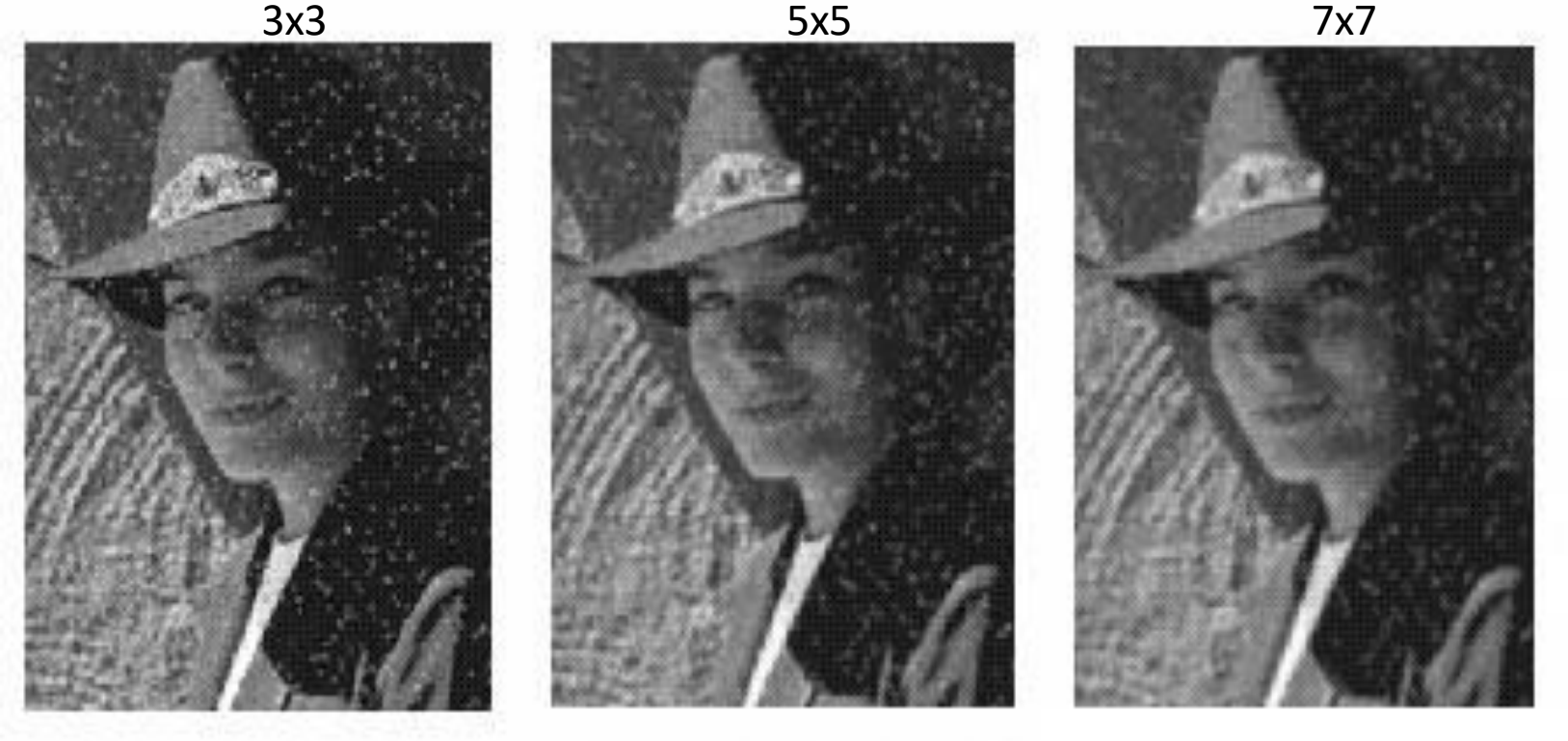
Source: M. Hebert

Reducing Gaussian noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

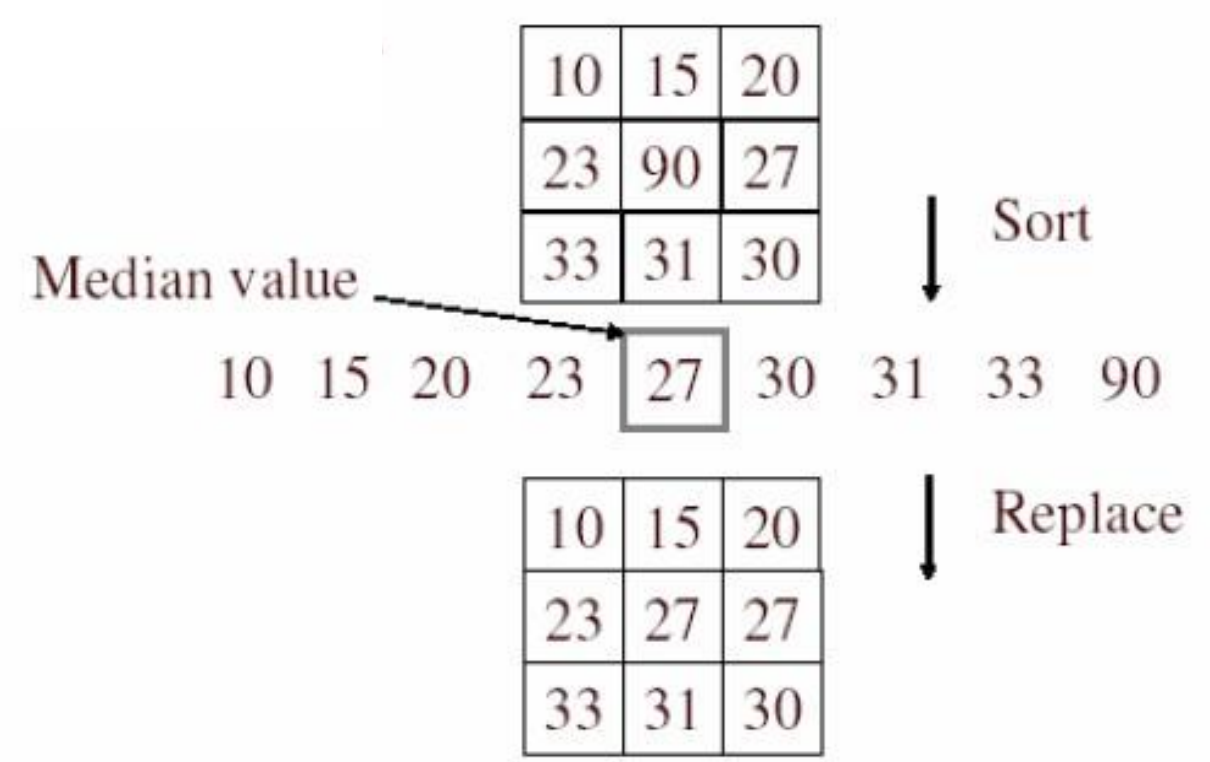
Reducing salt-and-pepper noise



- Reducing Salt-and-Pepper noise using Gaussian filters
- See the problem? What do we do?

Alternative idea: Median filtering

- A **median filter** operates over a window by selecting the median intensity in the window



- Is median filtering linear?

Source: K. Grauman

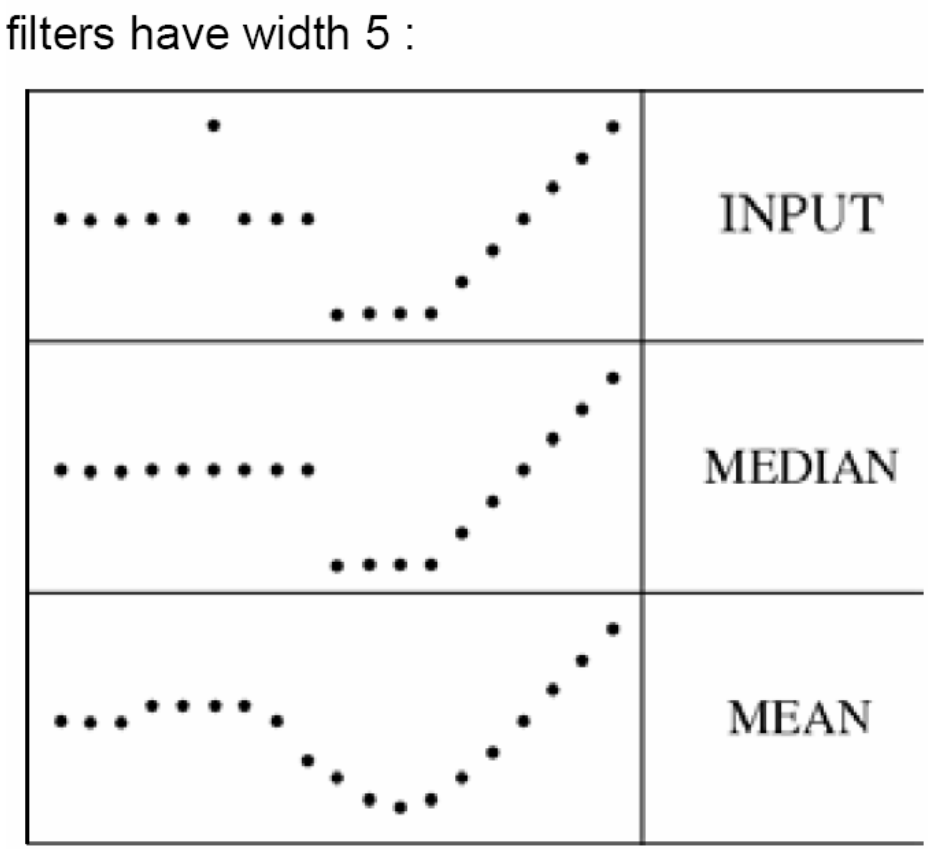
Median filter

Properties:

- Non-linear
- Does not spread noise
- Can remove spike noise
- Robust to outliers
- Not good for Gaussian noise

Median filter

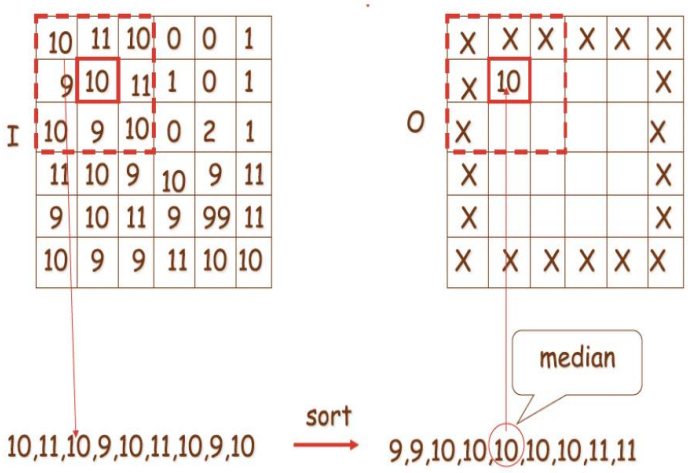
- What advantage does median filtering have over Gaussian filtering?
  - Robustness to outliers

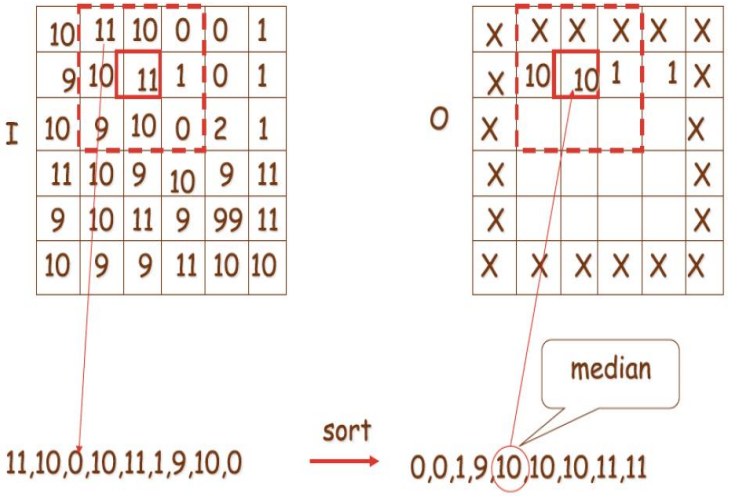


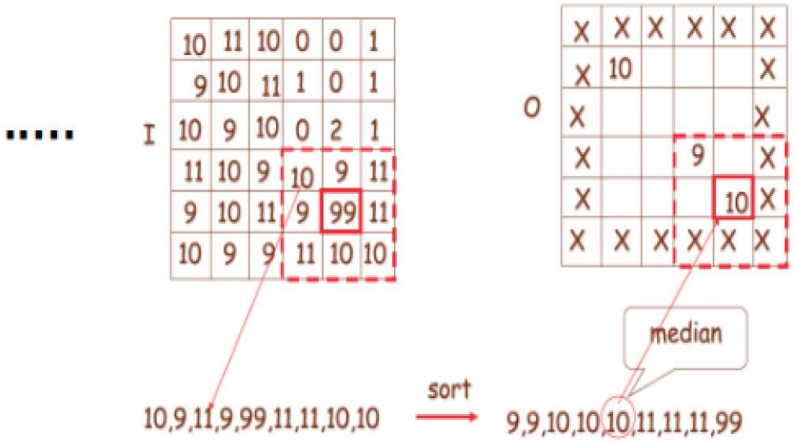
Source: K. Grauman

Median filter : Example

- Is median filtering linear?



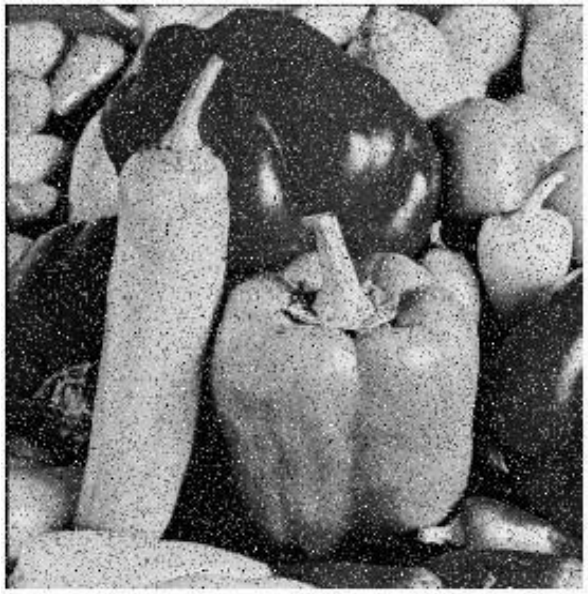


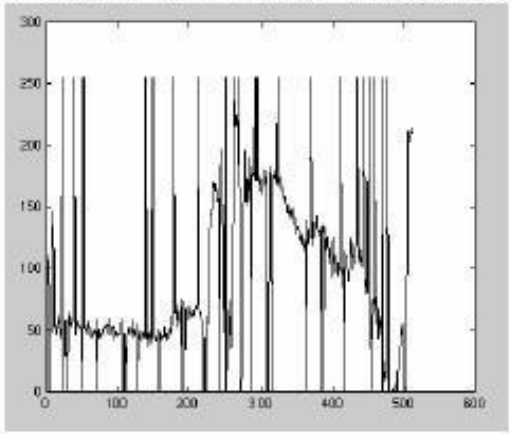


Notice how the outlier pixel value (99) got filtered out

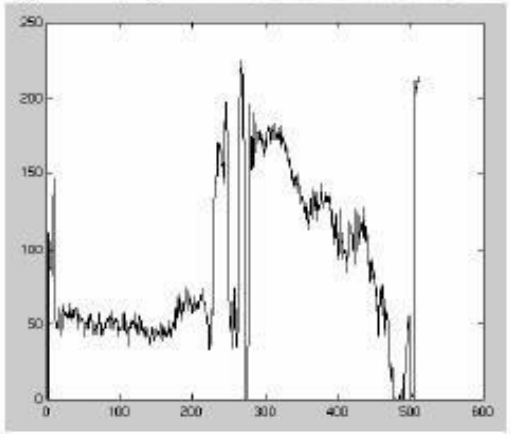
Median filter

Salt-and-pepper noise



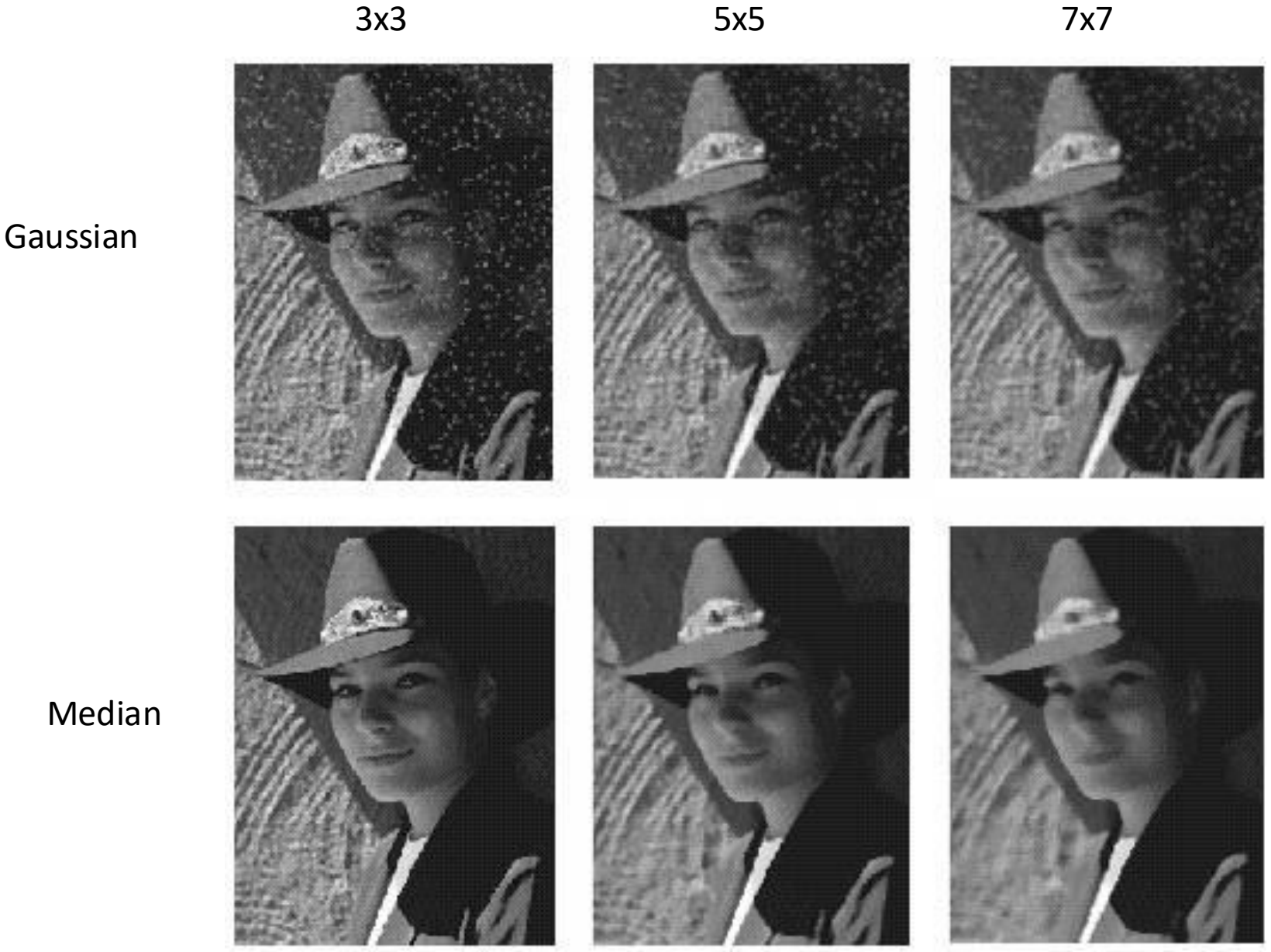


Median filtered

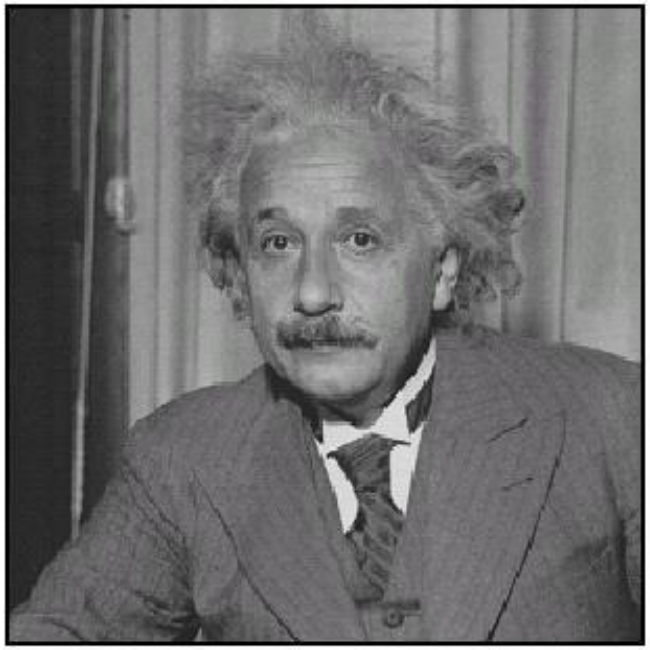


Source: M. Hebert

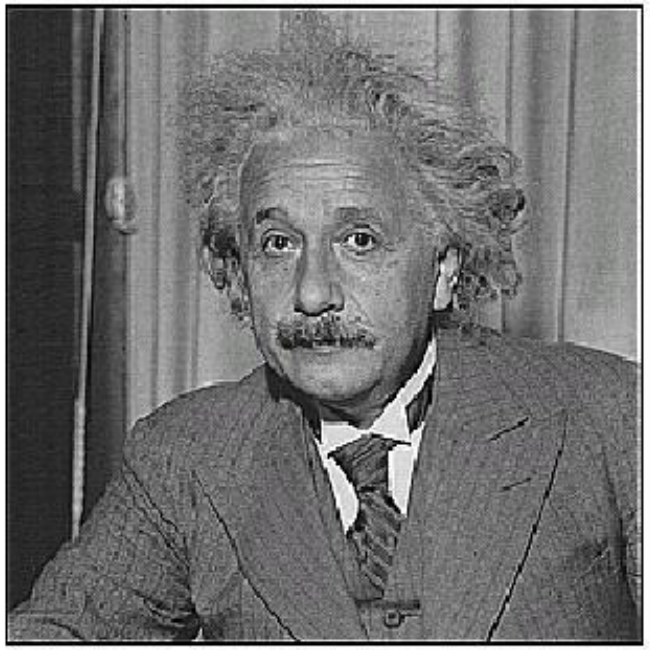
Gaussian vs. median filtering



Sharpening revisited



before



after

Source: D. Lowe

Sharpening revisited

- What does blurring take away?

–

=

Let's add it back:

+ α

=

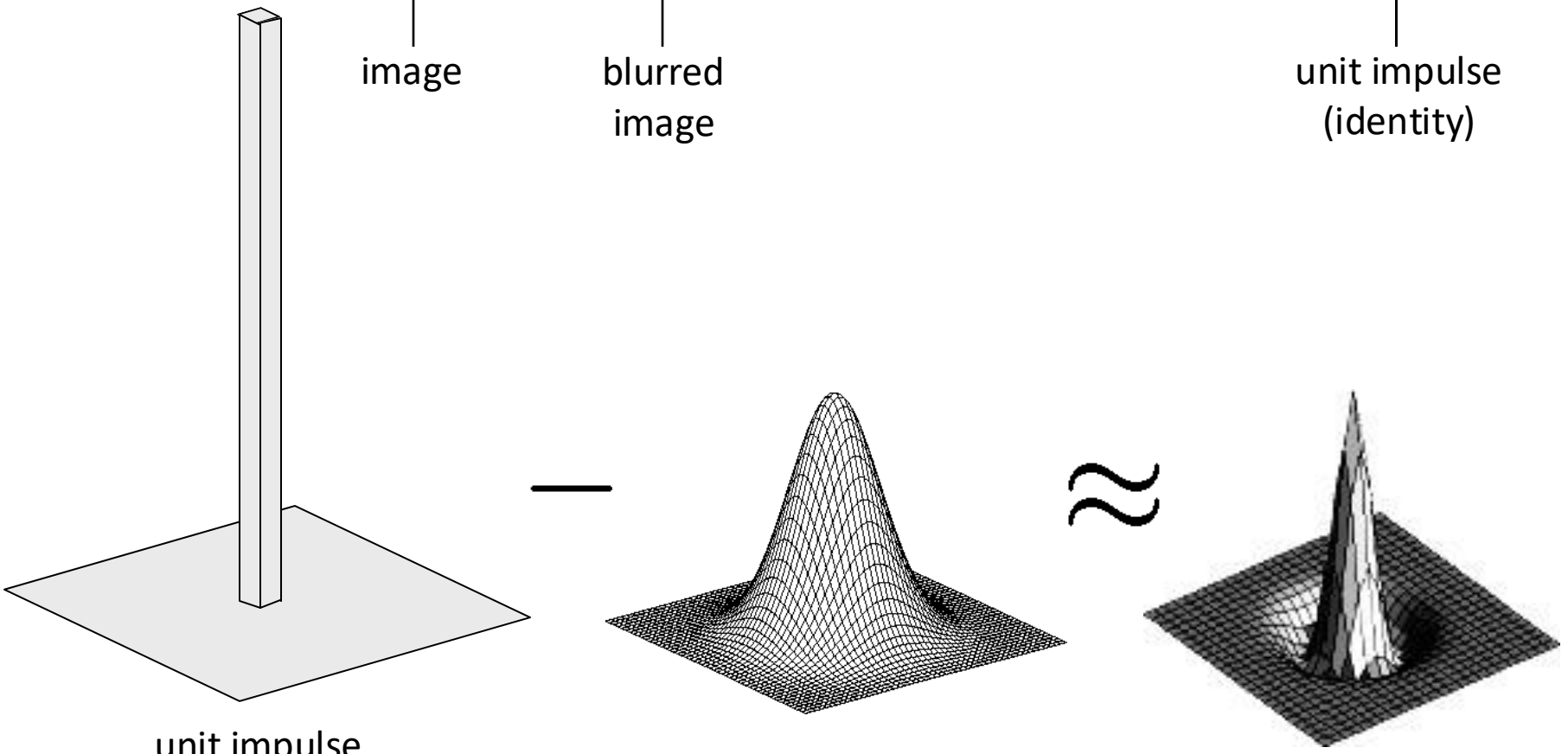
Sharp mask filter

$$f + \alpha(f - f * g) = (1+\alpha)f - \alpha\, f * g = f * ((1+\alpha)e - g)$$

image

blurred image

unit impulse (identity)



unit impulse

Gaussian

Laplacian of Gaussian

Eg: Sharpening filter

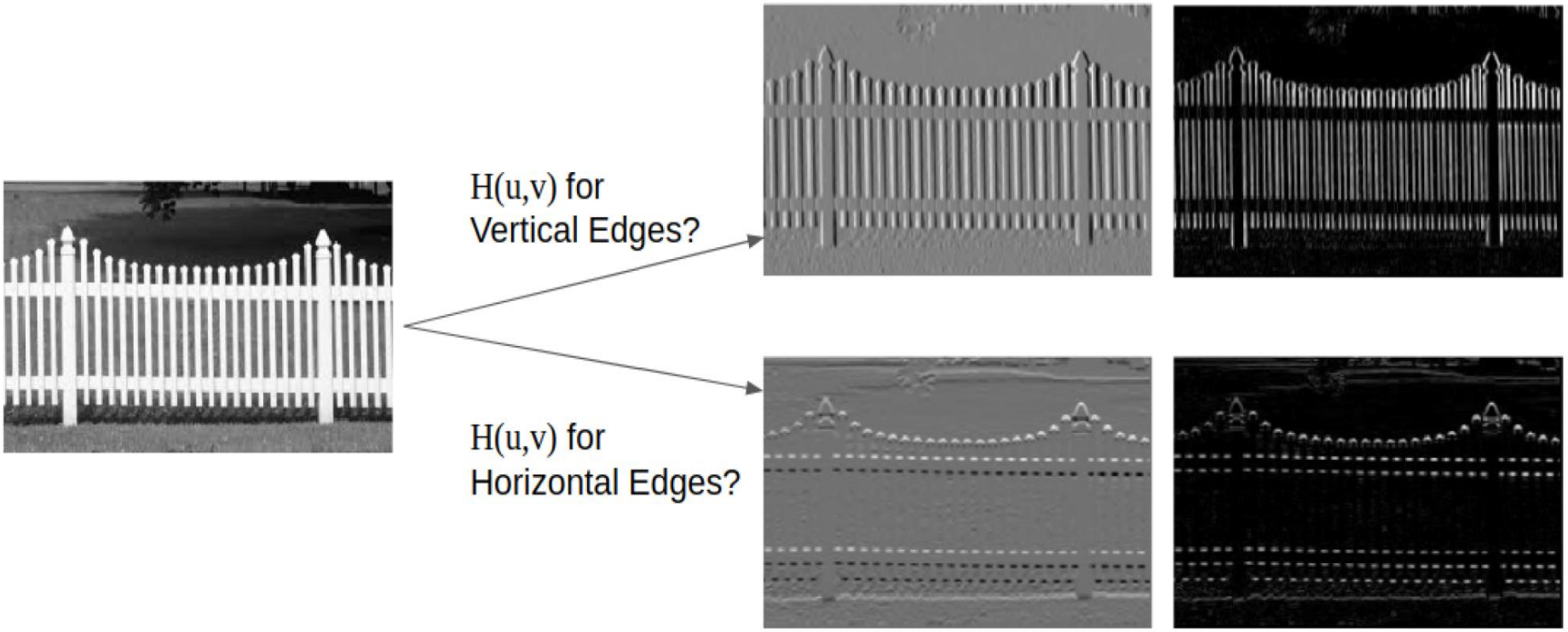
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Laplacian filters

Other Filters: Edge filter

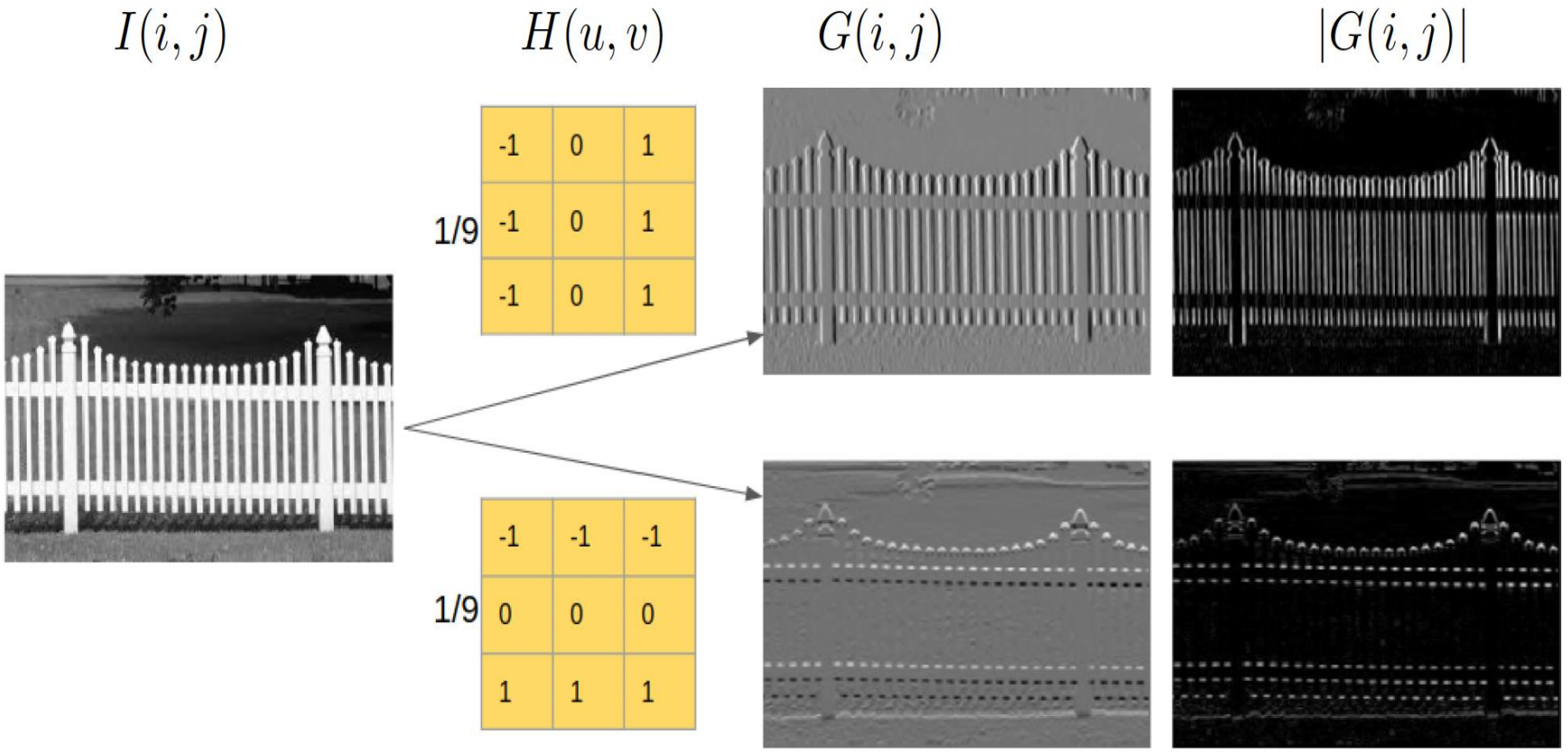
- *What should H look like to find the edges in a given image?*

$I(i,j)$ $H(u,v)$ $G(i,j)$ $|G(i,j)|$



Credit: KiwiWorker

Other Filters: Edge filter



- *We will discuss in detail about edge detection*

Credit: KivwWorker

Thank you…