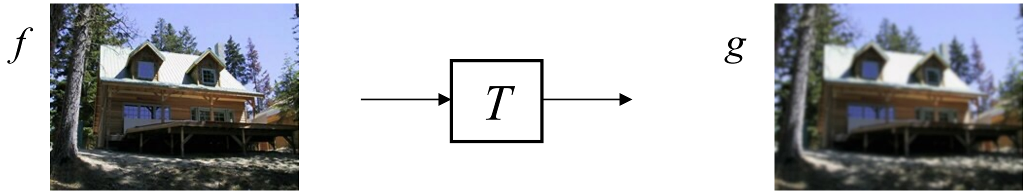
Image filtering

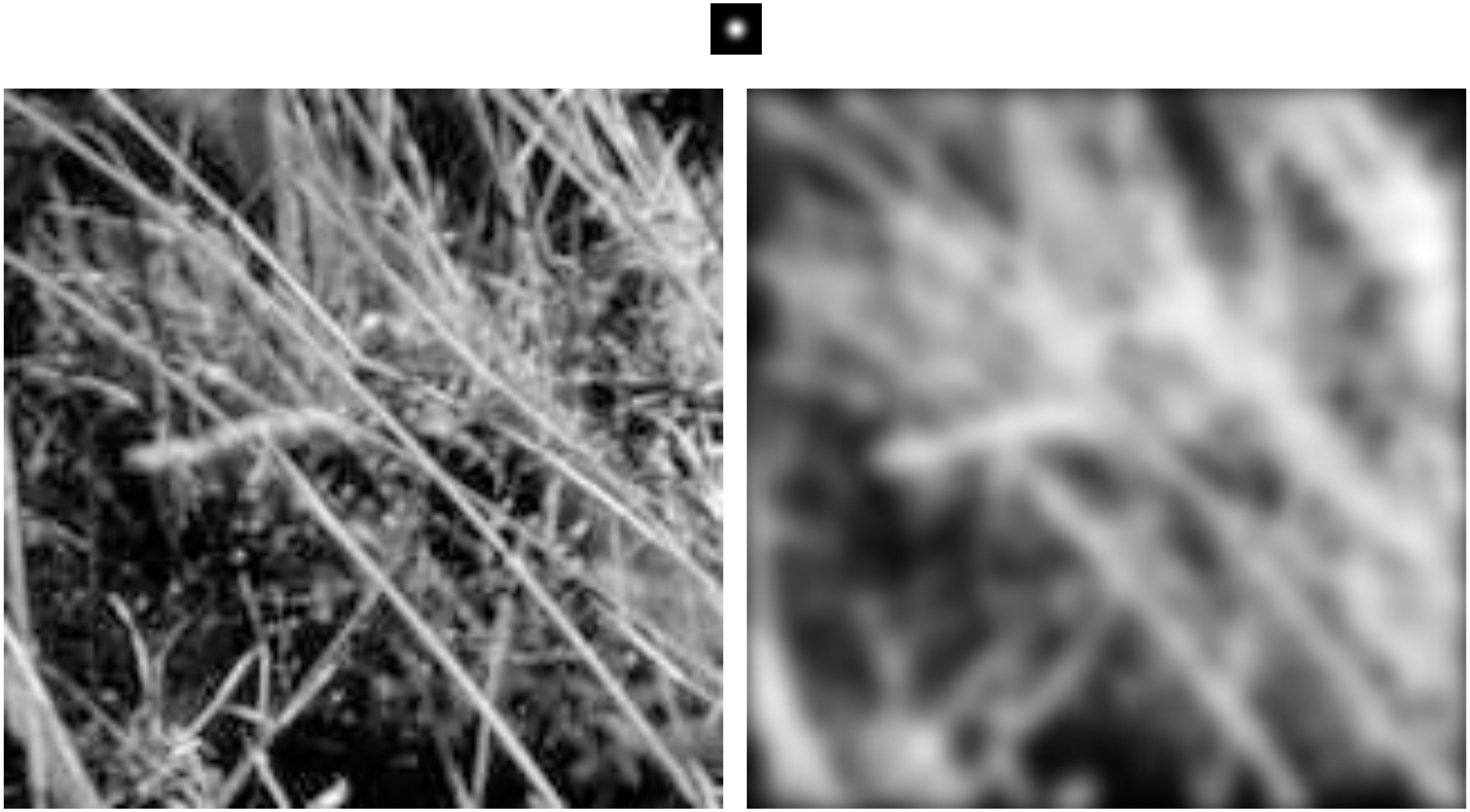
- Roughly speaking, replace image value at x with some function of values in its spatial neighborhood $N(x)$:

$$g(x) = T(f(N(x)))$$



- Examples: smoothing, sharpening, edge detection, etc.

Image filtering



Recall: Image transformations

- What are different kinds of image transformations?
  - Range transformations or point processing
  - Image warping
  - Image filtering

Image filtering: Outline

- Linear filtering and its properties
- Gaussian filters and their properties
- Nonlinear filtering: Median filtering
- Fun filtering application: Hybrid images

Sliding window operations

- Let’s slide a fixed-size window over the image and perform the same simple computation at each window location
- Example use case: how do we reduce image noise?
  - Let’s take the *average* of pixel values in each window
  - More generally, we can take a *weighted sum* where the weights are given by a *filter kernel*

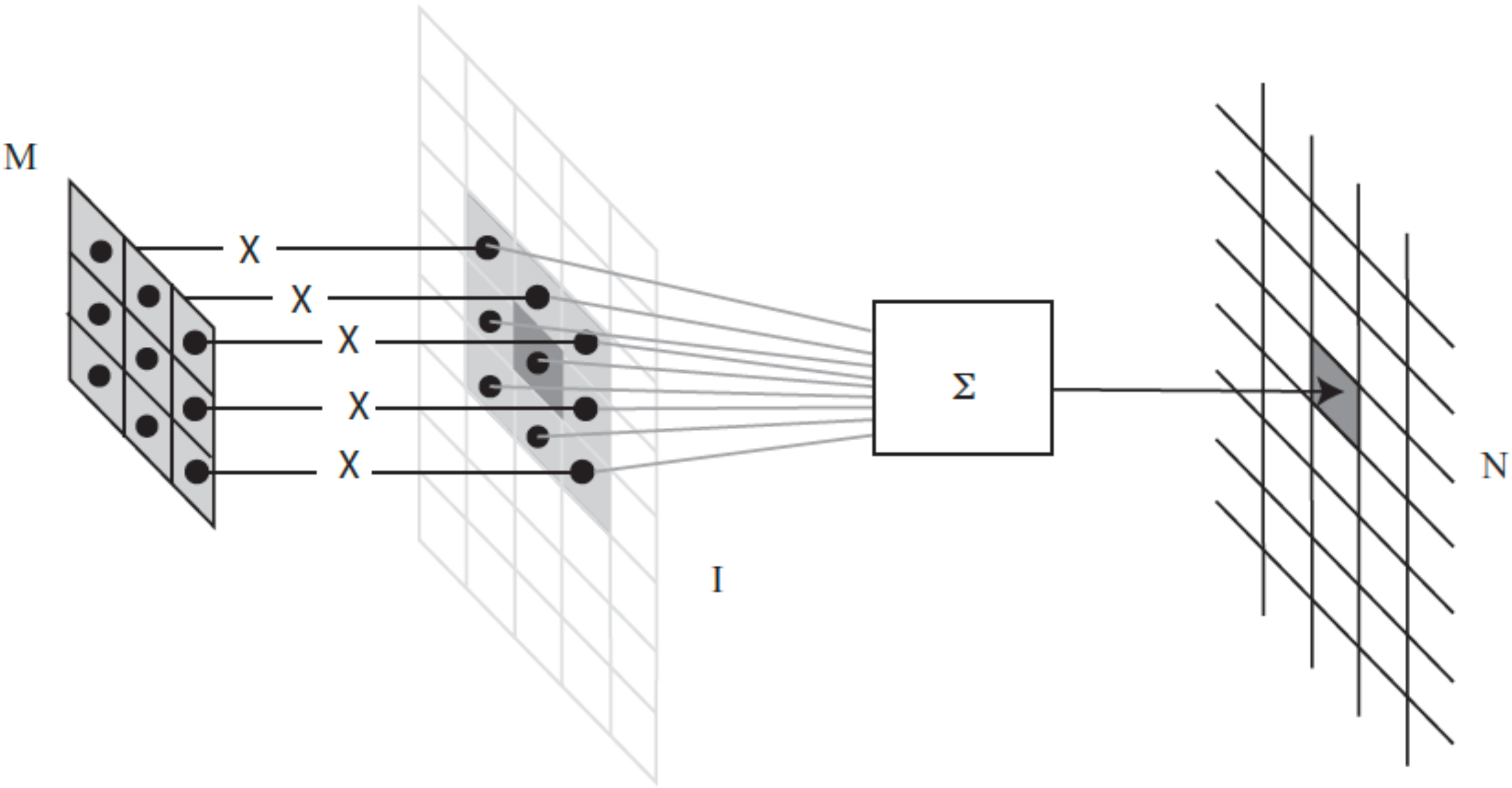


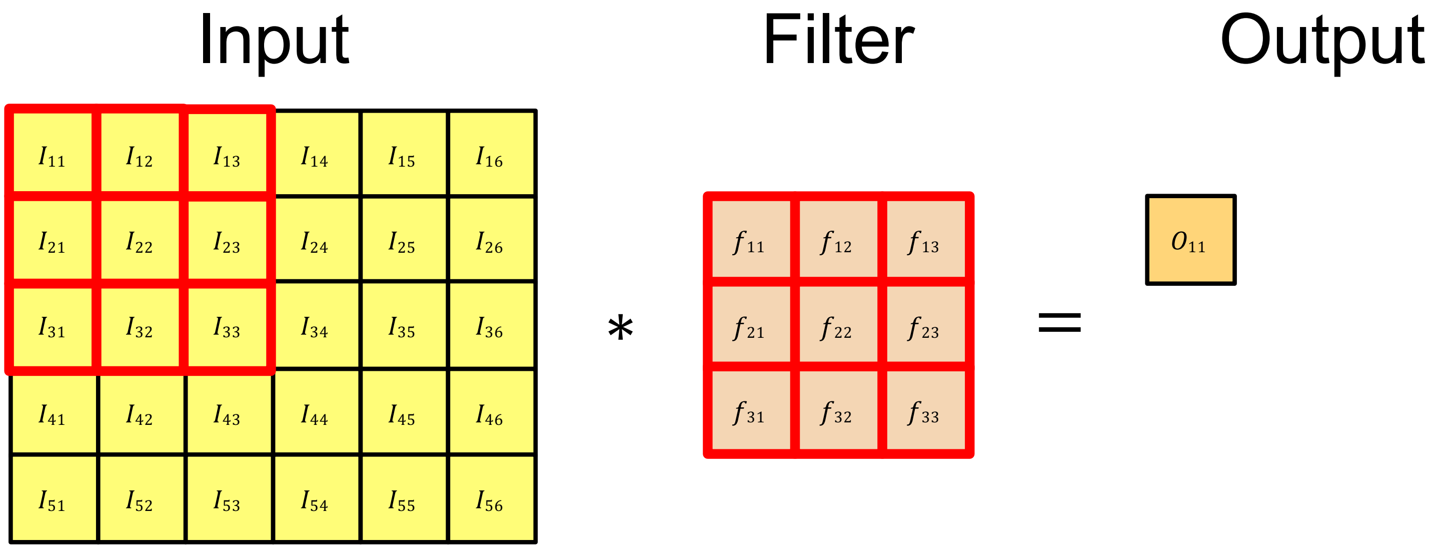
FIGURE 3.1: *To compute the value of $\mathcal{N}$ at some location, you shift a copy of $\mathcal{M}$ (the flipped version of $\mathcal{W}$) to lie over that location in $\mathcal{I}$; you multiply together the non-zero elements of $\mathcal{M}$ and $\mathcal{I}$ that lie on top of one another; and you sum the results.*

Applying a linear filter

$$\begin{array}{ccccc} \text{Input} & & \text{Filter} & & \text{Output} \\ \begin{bmatrix} I_{11} & I_{12} & I_{13} & I_{14} & I_{15} & I_{16} \\ I_{21} & I_{22} & I_{23} & I_{24} & I_{25} & I_{26} \\ I_{31} & I_{32} & I_{33} & I_{34} & I_{35} & I_{36} \\ I_{41} & I_{42} & I_{43} & I_{44} & I_{45} & I_{46} \\ I_{51} & I_{52} & I_{53} & I_{54} & I_{55} & I_{56} \end{bmatrix} & * & \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} & = & \end{array}$$

Adapted from D. Fouhey and J. Johnson

Applying a linear filter



Input

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}

$*$

Filter

f_{11}	f_{12}	f_{13}
f_{21}	f_{22}	f_{23}
f_{31}	f_{32}	f_{33}

$=$

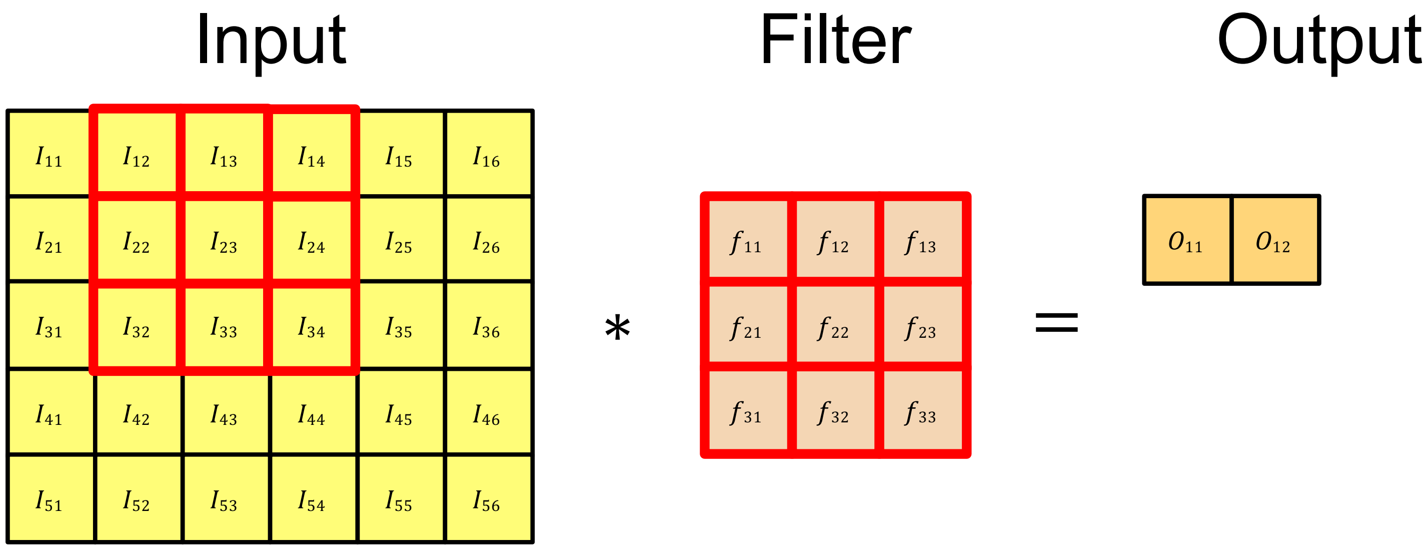
Output

O_{11}

$$O_{11} = I_{11} \cdot f_{11} + I_{12} \cdot f_{12} + I_{13} \cdot f_{13} + \ldots + I_{33} \cdot f_{33}$$

Adapted from D. Fouhey and J. Johnson

Applying a linear filter



Input

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}

$*$

Filter

f_{11}	f_{12}	f_{13}
f_{21}	f_{22}	f_{23}
f_{31}	f_{32}	f_{33}

$=$

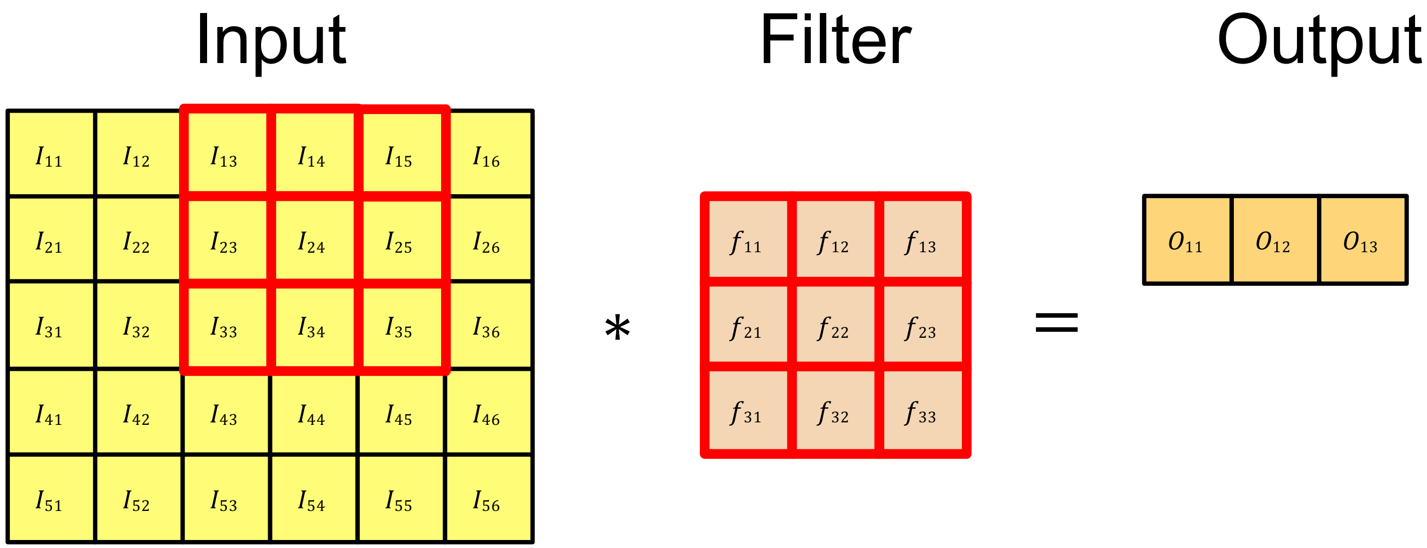
Output

O_{11}	O_{12}

$$O_{12} = I_{12} \cdot f_{11} + I_{13} \cdot f_{12} + I_{14} \cdot f_{13} + \dots + I_{34} \cdot f_{33}$$

Adapted from D. Fouhey and J. Johnson

Applying a linear filter



Input

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}

$*$

Filter

f_{11}	f_{12}	f_{13}
f_{21}	f_{22}	f_{23}
f_{31}	f_{32}	f_{33}

$=$

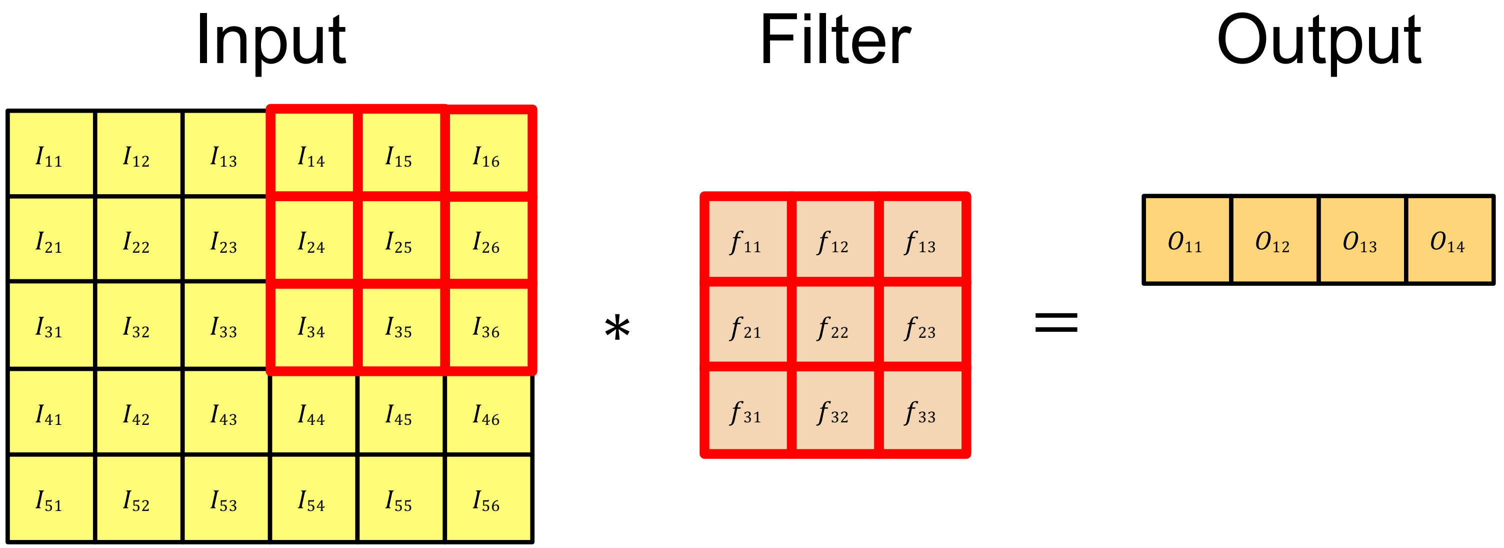
Output

O_{11}	O_{12}	O_{13}

$$O_{13} = I_{13} \cdot f_{11} + I_{14} \cdot f_{12} + I_{15} \cdot f_{13} + \ldots + I_{35} \cdot f_{33}$$

Adapted from D. Fouhey and J. Johnson

Applying a linear filter



Input

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}

$*$

Filter

f_{11}	f_{12}	f_{13}
f_{21}	f_{22}	f_{23}
f_{31}	f_{32}	f_{33}

$=$

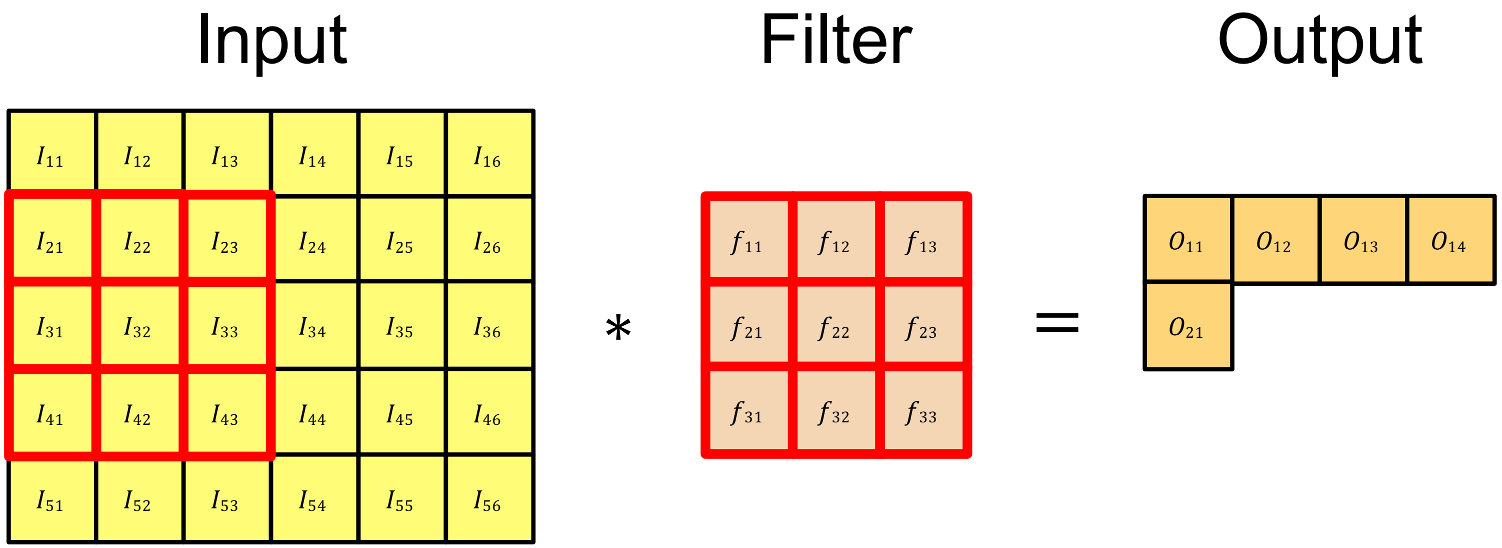
Output

O_{11}	O_{12}	O_{13}	O_{14}

$$O_{14} = I_{14} \cdot f_{11} + I_{15} \cdot f_{12} + I_{16} \cdot f_{13} + \ldots + I_{36} \cdot f_{33}$$

Adapted from D. Fouhey and J. Johnson

Applying a linear filter



Input

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}

*

Filter

f_{11}	f_{12}	f_{13}
f_{21}	f_{22}	f_{23}
f_{31}	f_{32}	f_{33}

=

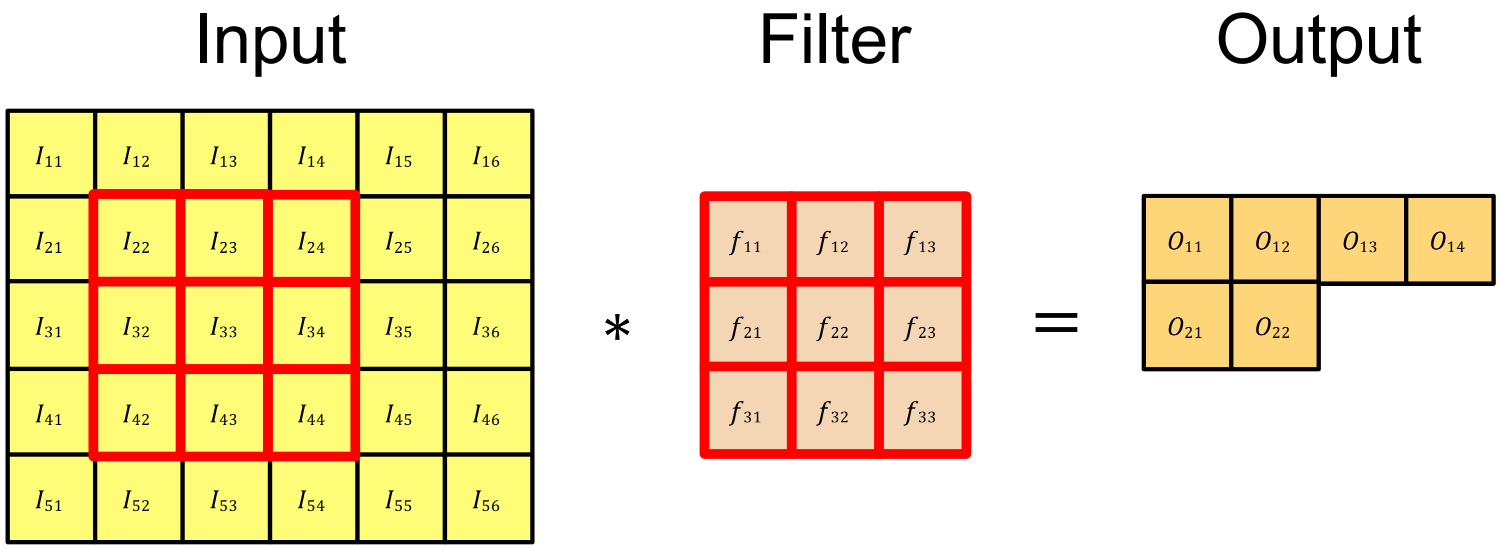
Output

O_{11}	O_{12}	O_{13}	O_{14}
O_{21}			

$$O_{21} = I_{21} \cdot f_{11} + I_{22} \cdot f_{12} + I_{23} \cdot f_{13} + \dots + I_{43} \cdot f_{33}$$

Adapted from D. Fouhey and J. Johnson

Applying a linear filter



Input

$$\begin{array}{cccccc} I_{11} & I_{12} & I_{13} & I_{14} & I_{15} & I_{16} \\ I_{21} & I_{22} & I_{23} & I_{24} & I_{25} & I_{26} \\ I_{31} & I_{32} & I_{33} & I_{34} & I_{35} & I_{36} \\ I_{41} & I_{42} & I_{43} & I_{44} & I_{45} & I_{46} \\ I_{51} & I_{52} & I_{53} & I_{54} & I_{55} & I_{56} \end{array}$$

$*$

Filter

$$\begin{array}{ccc} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{array}$$

$=$

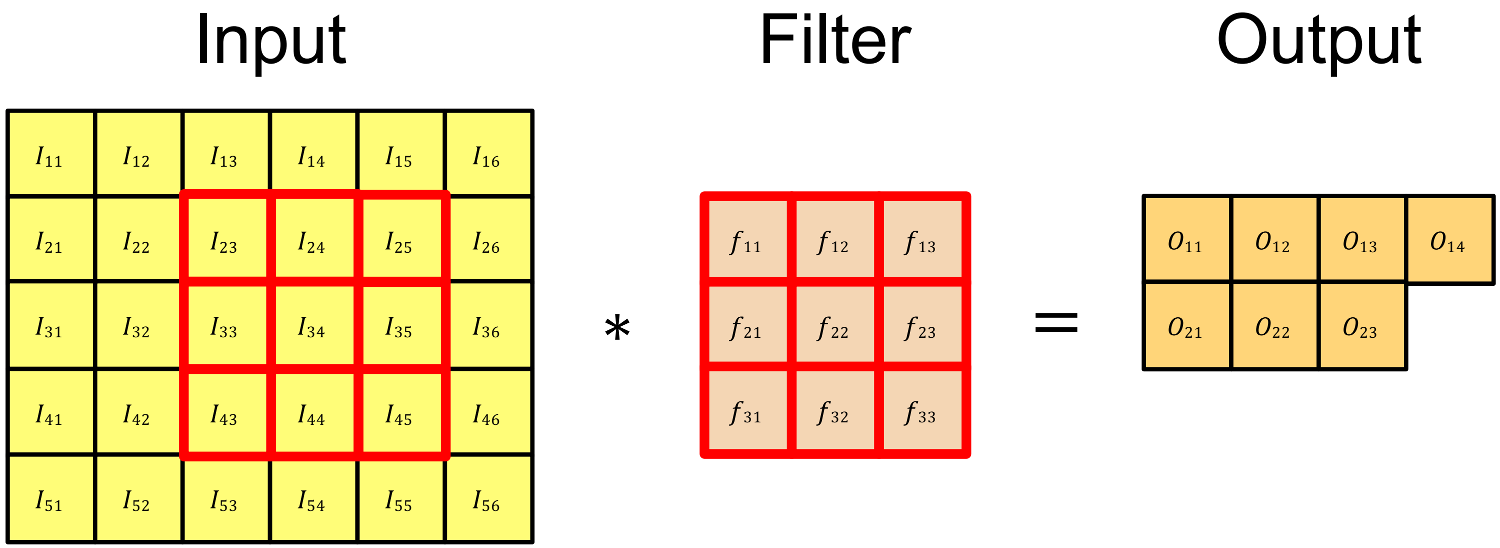
Output

$$\begin{array}{cccc} O_{11} & O_{12} & O_{13} & O_{14} \\ O_{21} & O_{22} & & \end{array}$$

$$O_{22} = I_{22} \cdot f_{11} + I_{23} \cdot f_{12} + I_{24} \cdot f_{13} + \ldots + I_{44} \cdot f_{33}$$

Adapted from D. Fouhey and J. Johnson

Applying a linear filter



Input

Filter

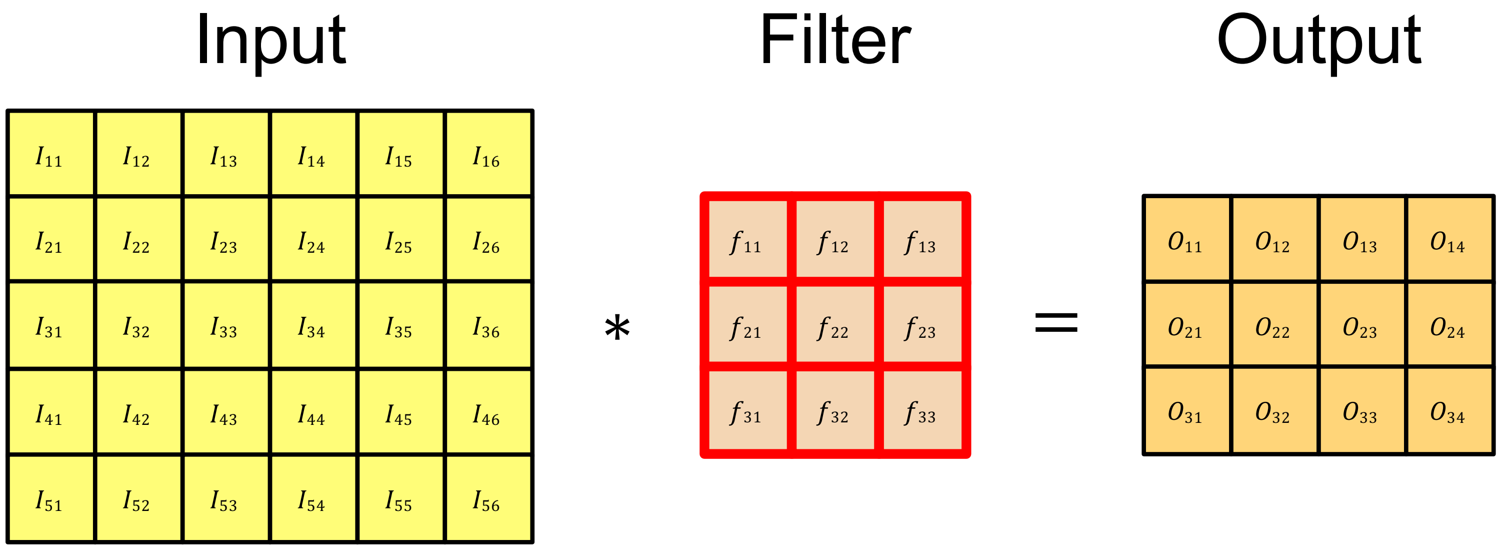
Output

$$\begin{array}{|c|c|c|c|c|c|}\hline I_{11} & I_{12} & I_{13} & I_{14} & I_{15} & I_{16} \\ \hline I_{21} & I_{22} & I_{23} & I_{24} & I_{25} & I_{26} \\ \hline I_{31} & I_{32} & I_{33} & I_{34} & I_{35} & I_{36} \\ \hline I_{41} & I_{42} & I_{43} & I_{44} & I_{45} & I_{46} \\ \hline I_{51} & I_{52} & I_{53} & I_{54} & I_{55} & I_{56} \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline f_{11} & f_{12} & f_{13} \\ \hline f_{21} & f_{22} & f_{23} \\ \hline f_{31} & f_{32} & f_{33} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline O_{11} & O_{12} & O_{13} & O_{14} \\ \hline O_{21} & O_{22} & O_{23} & \\ \hline \end{array}$$

$$O_{23} = I_{23} \cdot f_{11} + I_{24} \cdot f_{12} + I_{25} \cdot f_{13} + \ldots + I_{45} \cdot f_{33}$$

Adapted from D. Fouhey and J. Johnson

Applying a linear filter



Input

Filter

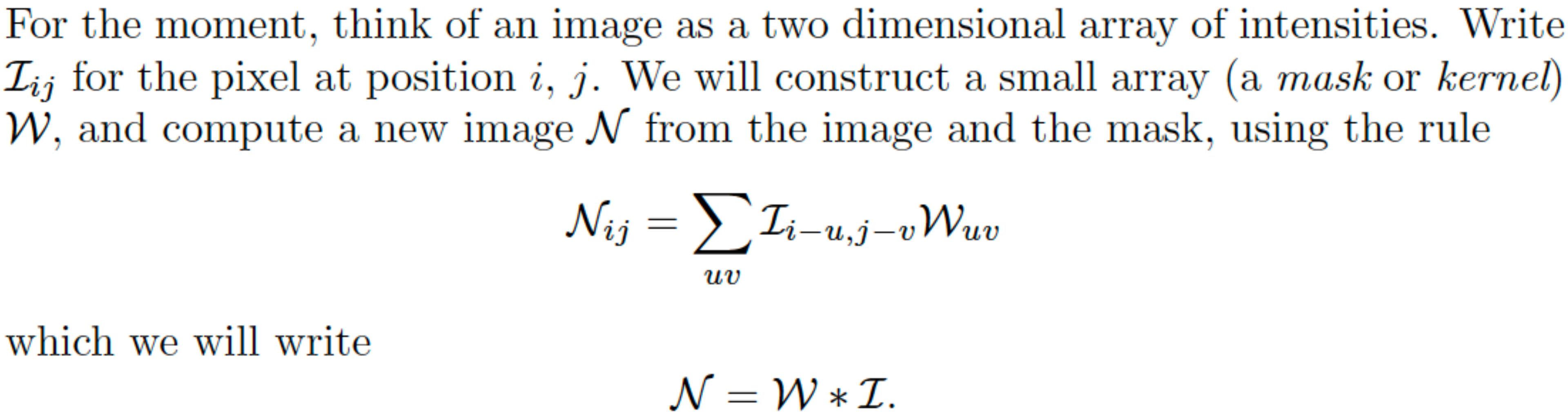
Output

$$\begin{bmatrix} I_{11} & I_{12} & I_{13} & I_{14} & I_{15} & I_{16} \\ I_{21} & I_{22} & I_{23} & I_{24} & I_{25} & I_{26} \\ I_{31} & I_{32} & I_{33} & I_{34} & I_{35} & I_{36} \\ I_{41} & I_{42} & I_{43} & I_{44} & I_{45} & I_{46} \\ I_{51} & I_{52} & I_{53} & I_{54} & I_{55} & I_{56} \end{bmatrix} * \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} = \begin{bmatrix} O_{11} & O_{12} & O_{13} & O_{14} \\ O_{21} & O_{22} & O_{23} & O_{24} \\ O_{31} & O_{32} & O_{33} & O_{34} \end{bmatrix}$$

What filter values should we use to find the average in a 3×3 window?

Adapted from D. Fouhey and J. Johnson

Convolution

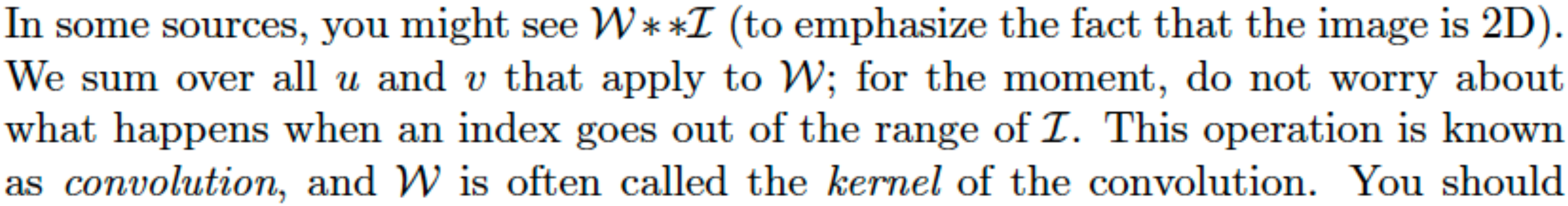


For the moment, think of an image as a two dimensional array of intensities. Write $\mathcal{I}_{ij}$ for the pixel at position i, j. We will construct a small array (a *mask* or *kernel*) $\mathcal{W}$, and compute a new image $\mathcal{N}$ from the image and the mask, using the rule

$$\mathcal{N}_{ij} = \sum_{uv} \mathcal{I}_{i-u,j-v} \mathcal{W}_{uv}$$

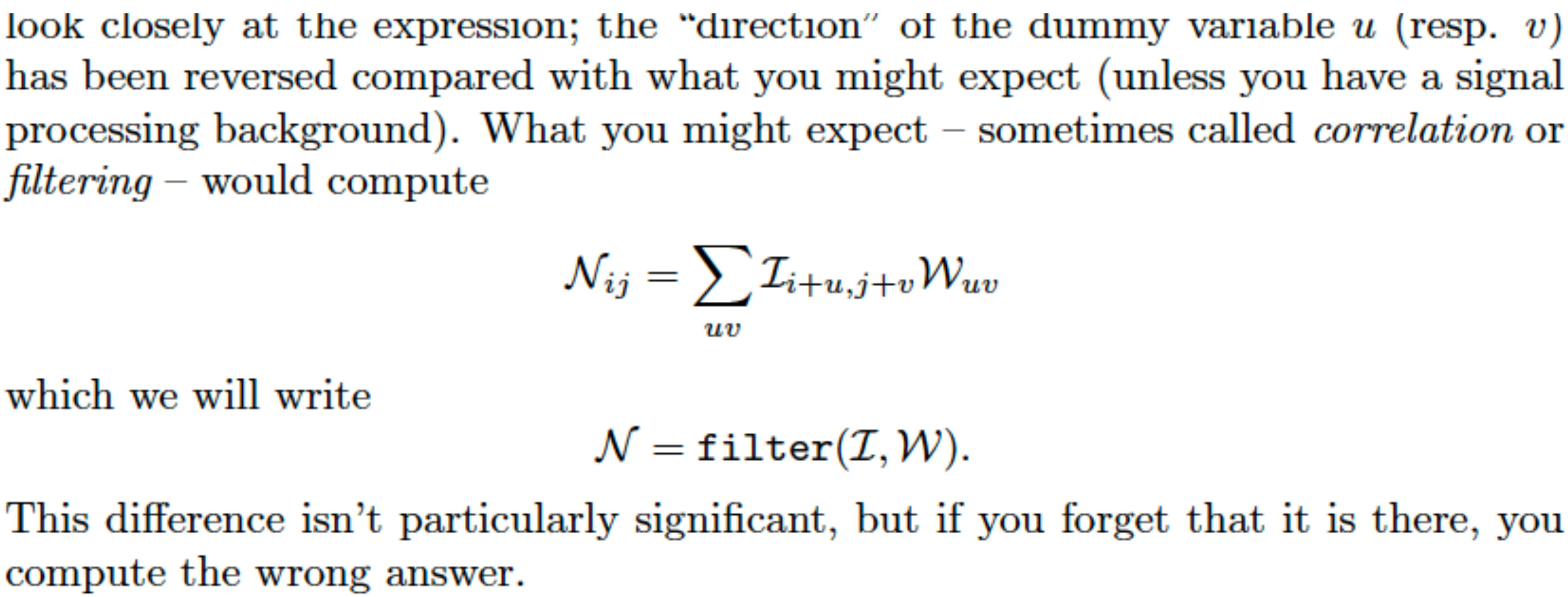
which we will write

$$\mathcal{N} = \mathcal{W} * \mathcal{I}.$$



In some sources, you might see $\mathcal{W} * * \mathcal{I}$ (to emphasize the fact that the image is 2D). We sum over all u and v that apply to $\mathcal{W}$; for the moment, do not worry about what happens when an index goes out of the range of $\mathcal{I}$. This operation is known as *convolution*, and $\mathcal{W}$ is often called the *kernel* of the convolution. You should

Filtering



look closely at the expression; the “direction” of the dummy variable u (resp. v) has been reversed compared with what you might expect (unless you have a signal processing background). What you might expect – sometimes called *correlation* or *filtering* – would compute

$$\mathcal{N}_{ij} = \sum_{uv} \mathcal{I}_{i+u,j+v} \mathcal{W}_{uv}$$

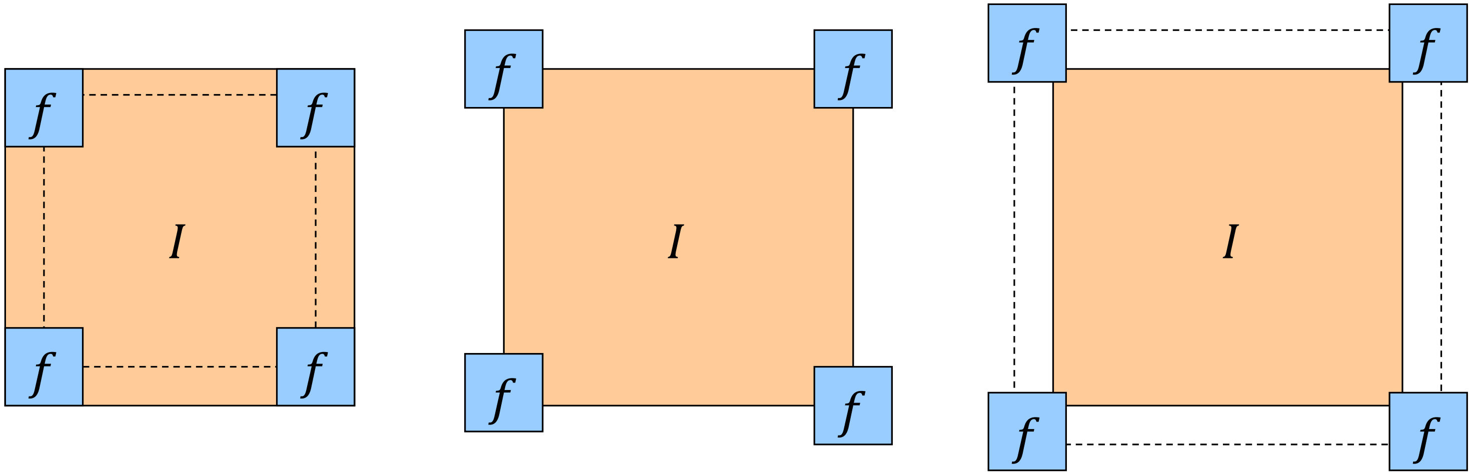
which we will write

$$\mathcal{N} = \texttt{filter}(\mathcal{I}, \mathcal{W}).$$

This difference isn’t particularly significant, but if you forget that it is there, you compute the wrong answer.

Practical details: Dealing with edges

- To control the size of the output, we need to use *padding*



Output is smaller than input

Output is same size as input

Output is larger than input

Practical details: Dealing with edges

- To control the size of the output, we need to use *padding*
- What values should we pad the image with?

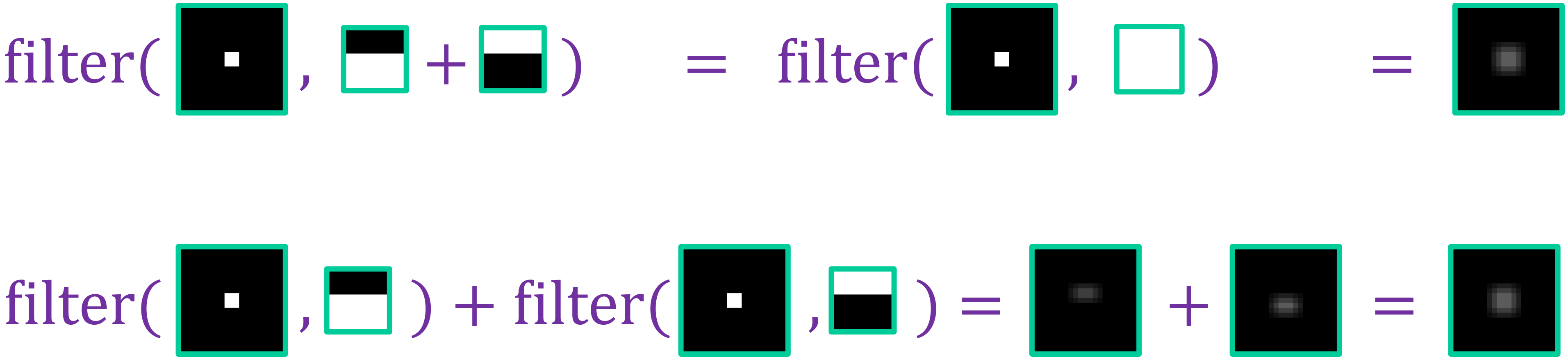
Practical details: Dealing with edges

- To control the size of the output, we need to use *padding*
- What values should we pad the image with?
  - Zero pad (or clip filter)
  - Wrap around
  - Copy edge
  - Reflect across edge

Source: S. Marschner

Properties: Linearity

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



filter(, +) = filter(,) =

filter(,) + filter(,) = + =

Adapted from D. Fouhey and J. Johnson

Properties: Linearity

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

Also:

$$\text{filter}(I_1 + I_2, f) = \; \text{filter}(I_1, f) + \; \text{filter}(I_2, f)$$

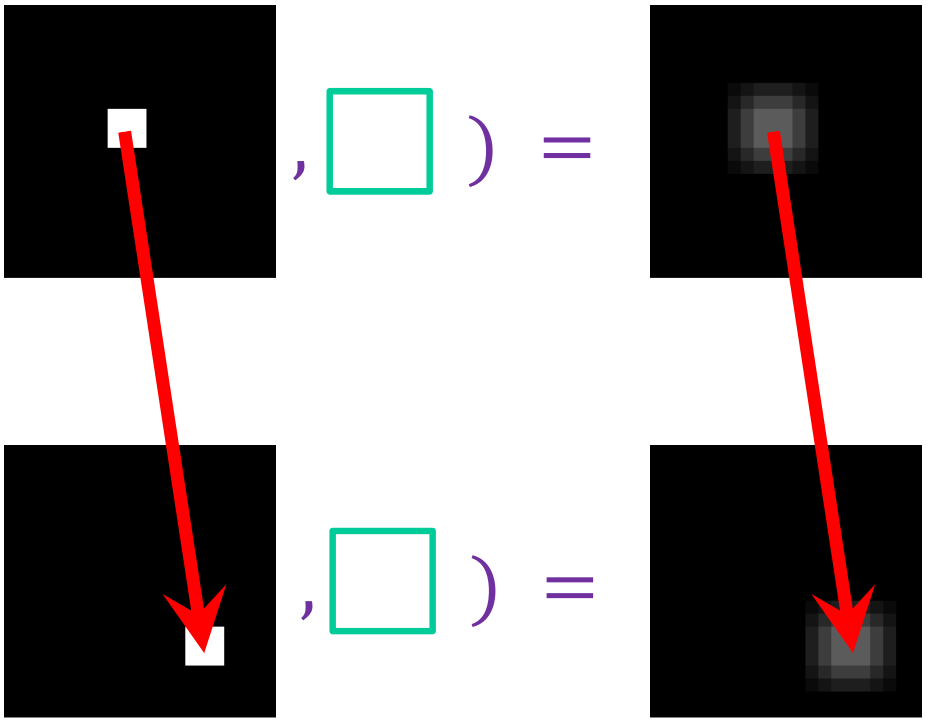
$$\text{filter}(kI, f) = k \; \text{filter}(I, f)$$

$$\text{filter}(I, kf) = k \; \text{filter}(I, f)$$

Adapted from D. Fouhey and J. Johnson

Properties: Shift-invariance

$$\text{filter}(\text{shift}(I), f) \;=\; \text{shift}(\text{filter}(I, f))$$



filter(,) = $\text{filter}(I, f)$

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

Adapted from D. Fouhey and J. Johnson

More linear filtering properties

- Commutativity: $f * g = g * f$
  - For infinite signals, no difference between filter and signal
- Associativity: $f * (g * h) = (f * g) * h$
  - Convolving several filters one after another is equivalent to convolving with one combined filter:

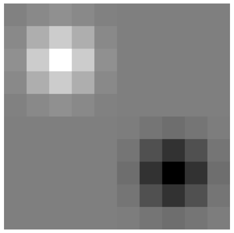
$$\left(\left((g * f_1) * f_2\right) * f_3\right) = g * (f_1 * f_2 * f_3)$$

- Identity: for *unit impulse* e, $f * e = f$

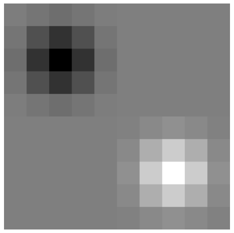
Note: Filtering vs. “convolution”

- In classical signal processing terminology, convolution is filtering with a *flipped* kernel, and filtering with an upright kernel is known as *cross-correlation*
  - Check convention of filtering function you plan to use!

Filtering or “cross-correlation”
(Kernel in original orientation)

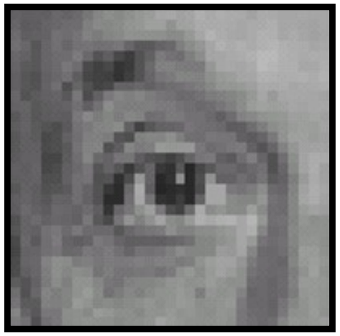


“Convolution”
(Kernel flipped in x and y)



Adapted from D. Fouhey and J. Johnson

Practice with linear filters



Original

0	0	0
0	1	0
0	0	0

?

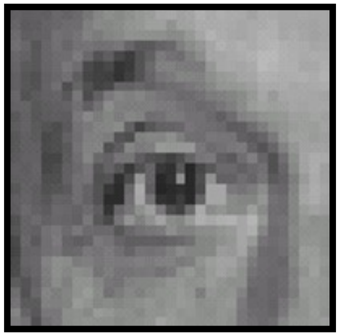
Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

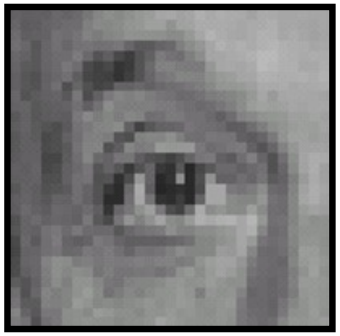
One surrounded by zeros is the *identity filter*



Filtered
(no change)

Source: D. Lowe

Practice with linear filters



Original

0	0	0
0	0	1
0	0	0

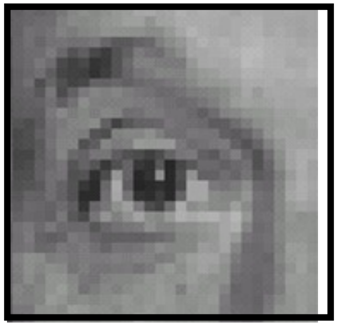
?

Source: D. Lowe

Practice with linear filters

Original

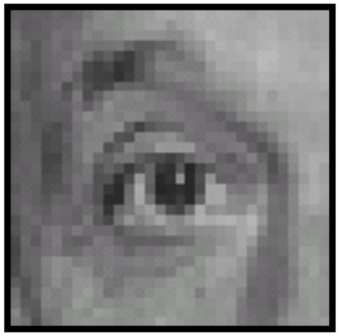
0	0	0
0	0	1
0	0	0



Shifted *left*
By one pixel

Source: D. Lowe

Practice with linear filters



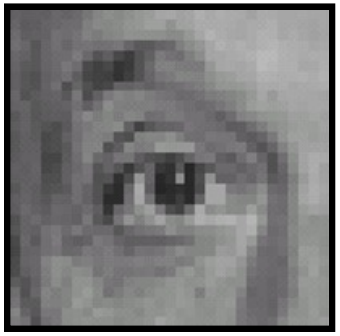
Original

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

?

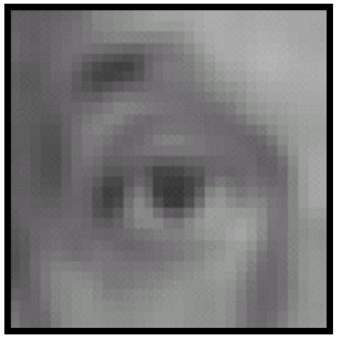
Source: D. Lowe

Practice with linear filters



Original

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



Blur (with a
box filter)

Source: D. Lowe

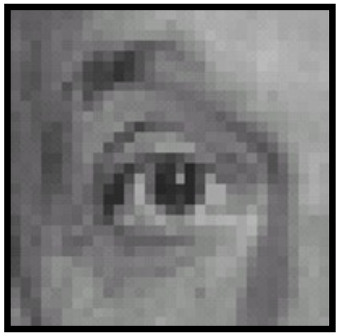
Practice with linear filters

Original

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

Source: D. Lowe

Practice with linear filters

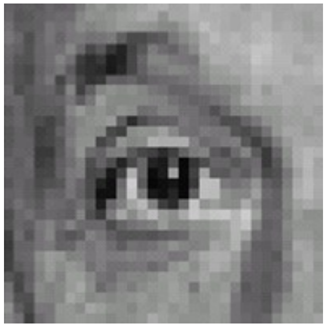


Original

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

Sharpening filter:
Accentuates differences
with local average

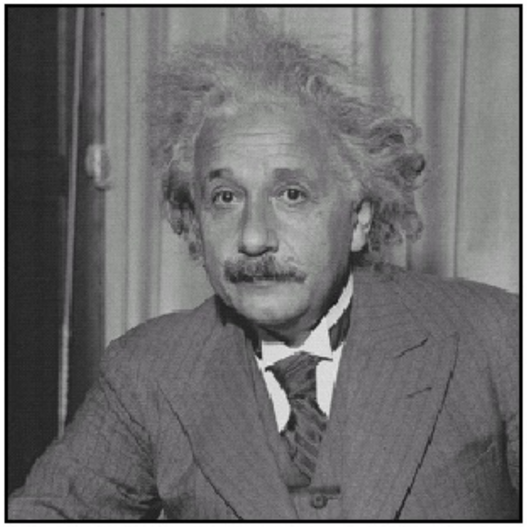
(Note that filter sums to 1)



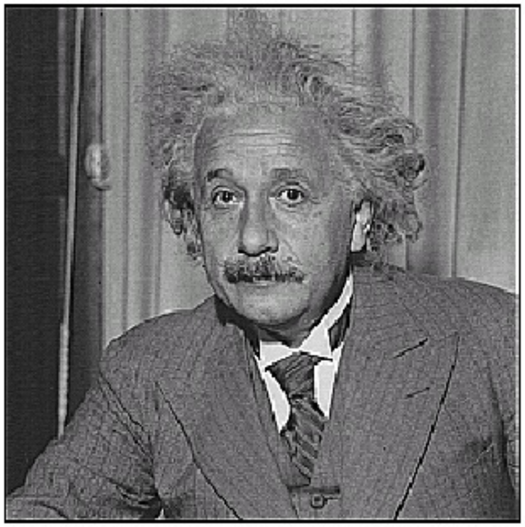
Sharpened

Source: D. Lowe

Sharpening



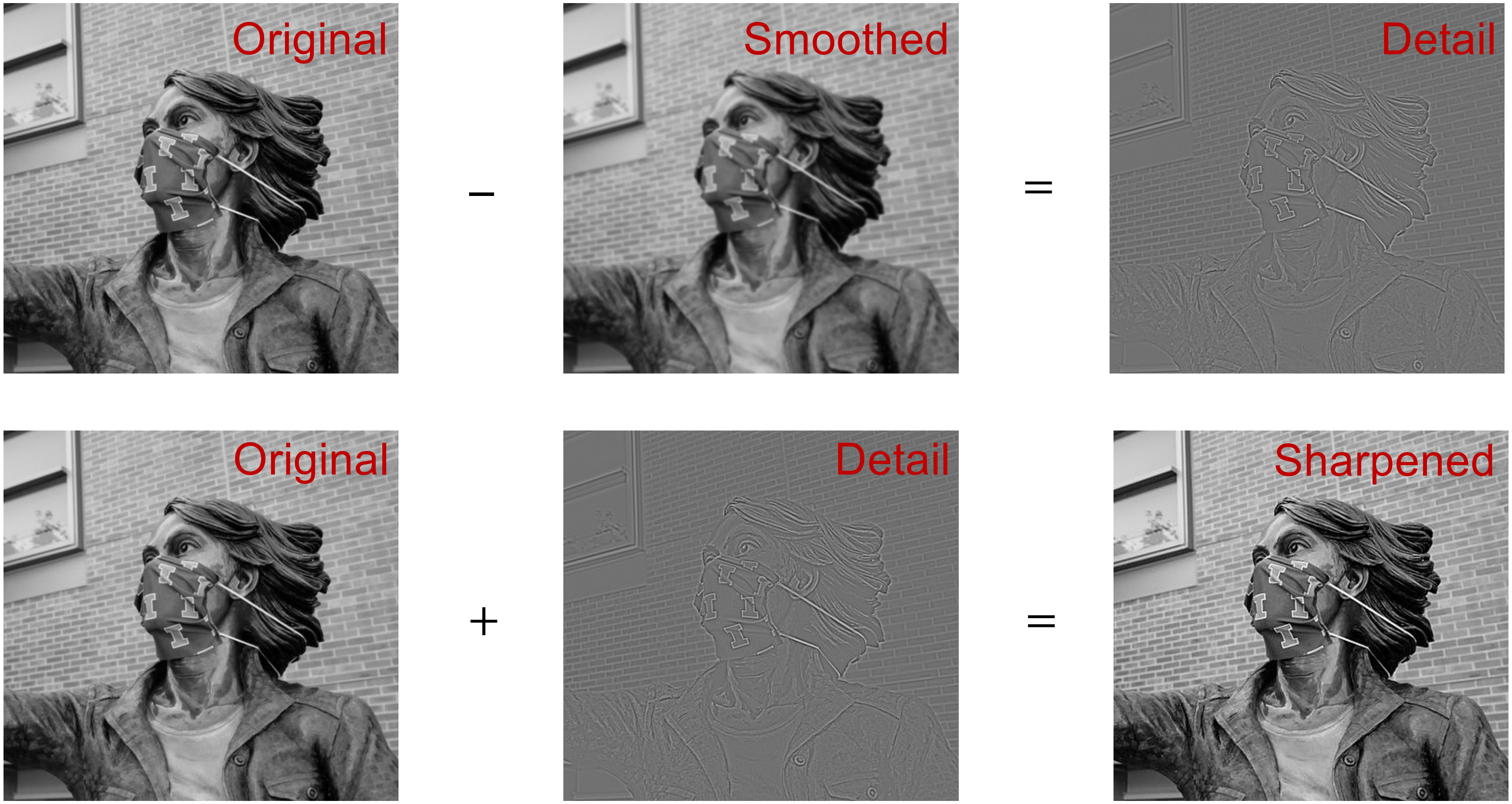
before



after

Source: D. Lowe

Sharpening



Source:
S. Gupta

Filters are dot products

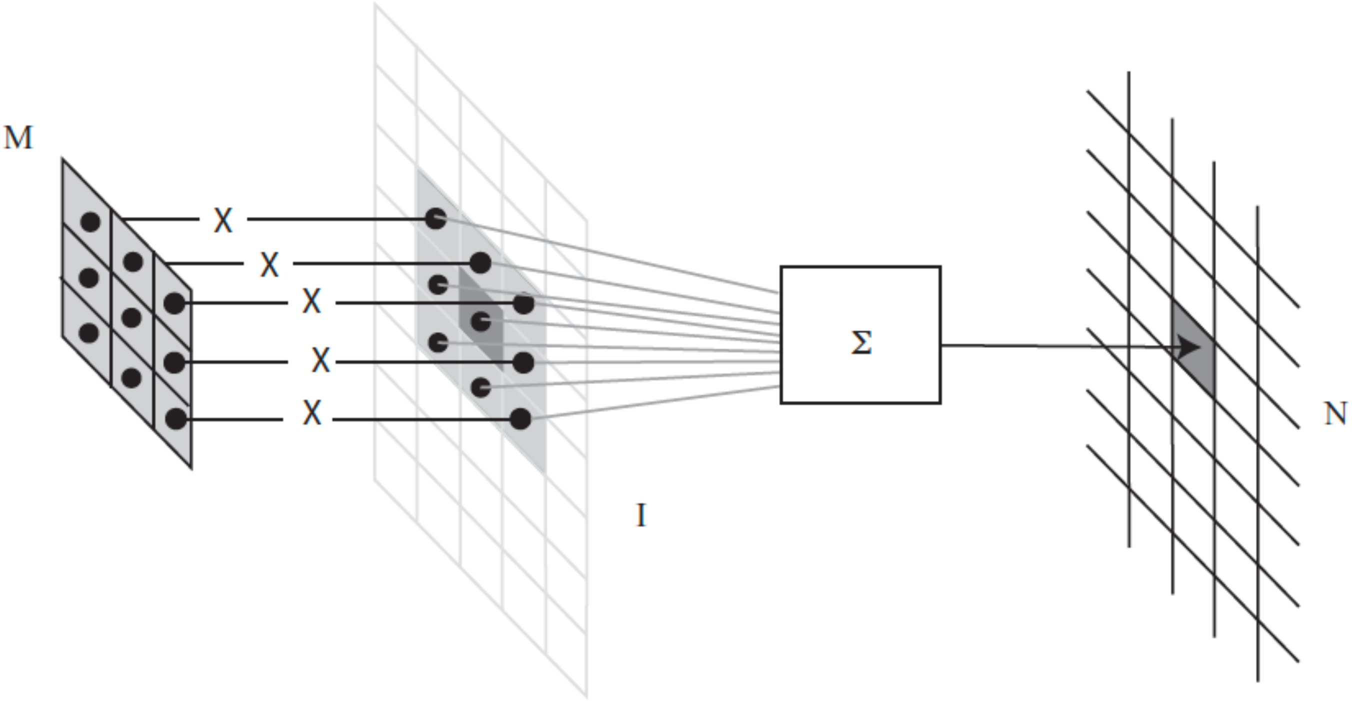


FIGURE 3.1: *To compute the value of $\mathcal{N}$ at some location, you shift a copy of $\mathcal{M}$ (the flipped version of $\mathcal{W}$) to lie over that location in $\mathcal{I}$; you multiply together the non-zero elements of $\mathcal{M}$ and $\mathcal{I}$ that lie on top of one another; and you sum the results.*

Filters are dot products

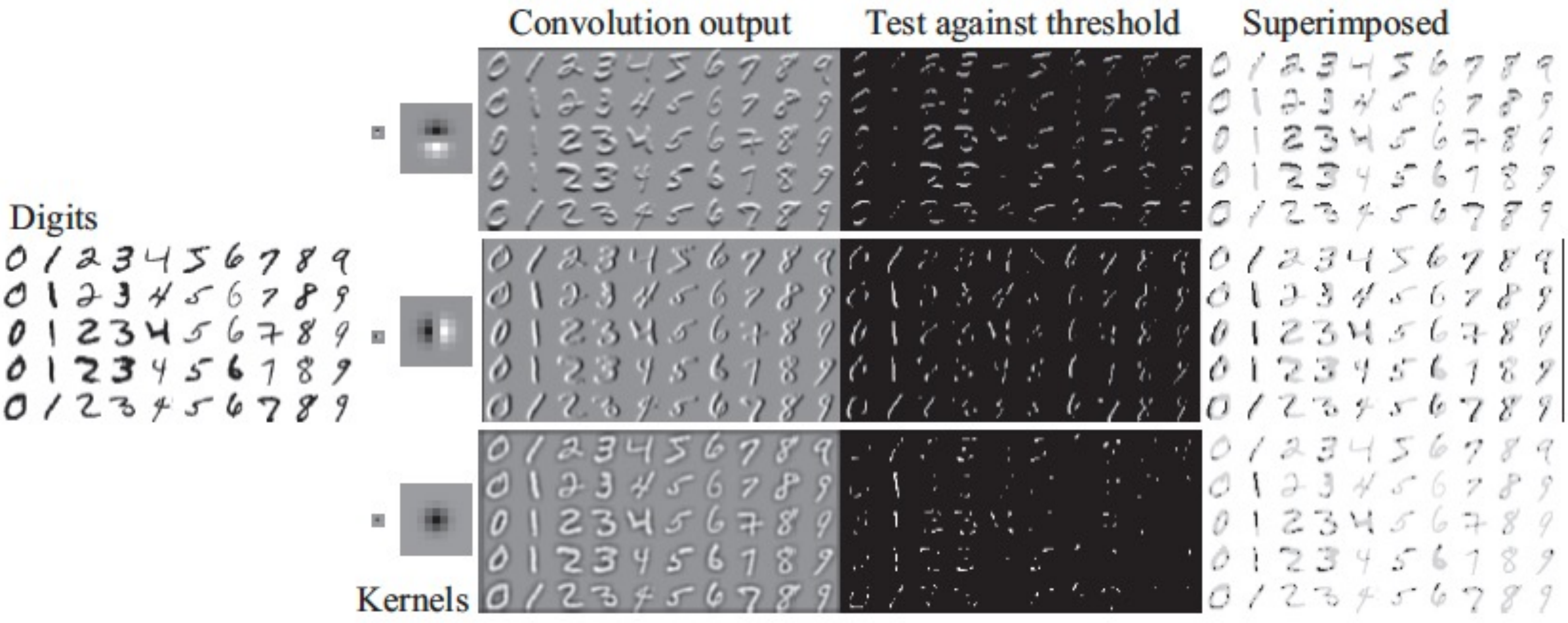
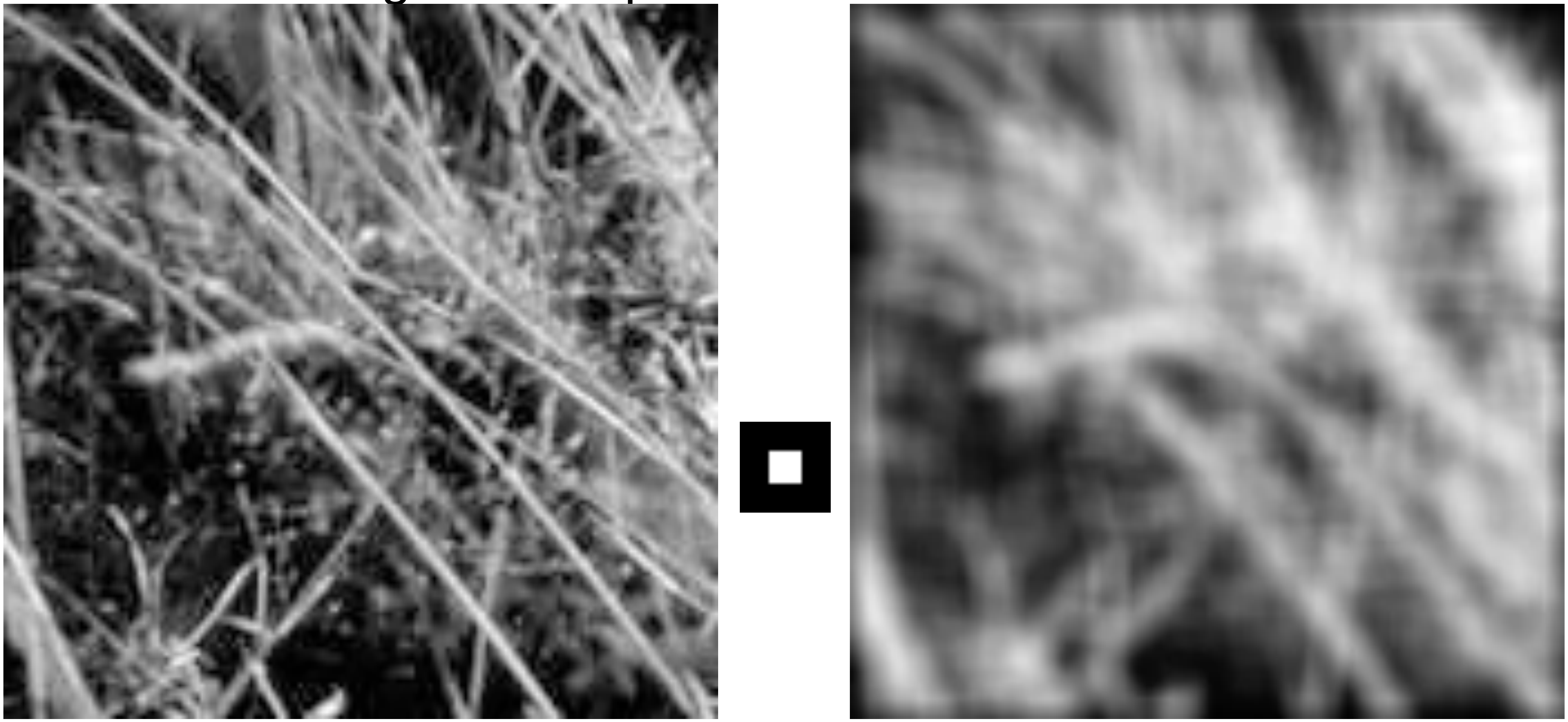


Image filtering: Outline

- Linear filtering and its properties
- Gaussian filters and their properties

Smoothing with box filter revisited

- What’s wrong with this picture?



Source: D. Forsyth

Smoothing with box filter revisited

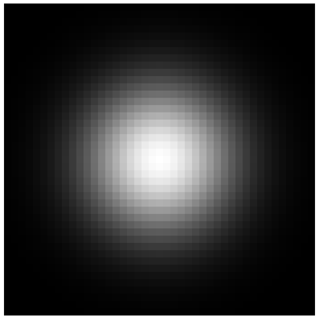
- What’s wrong with this picture?
- What’s the solution?
  - To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center

“proportional to”
(renormalize values to sum to 1)

$$G(x,y) \propto \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

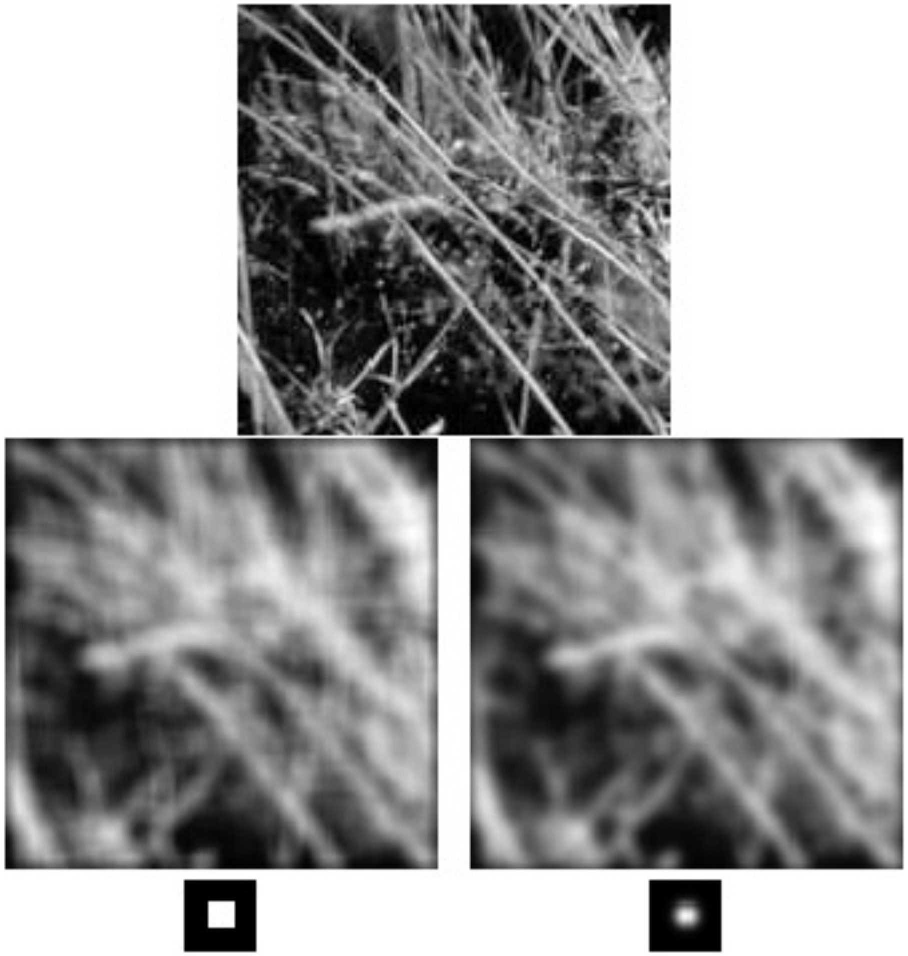
standard deviation
(determines size of “blob”)

Gaussian filter



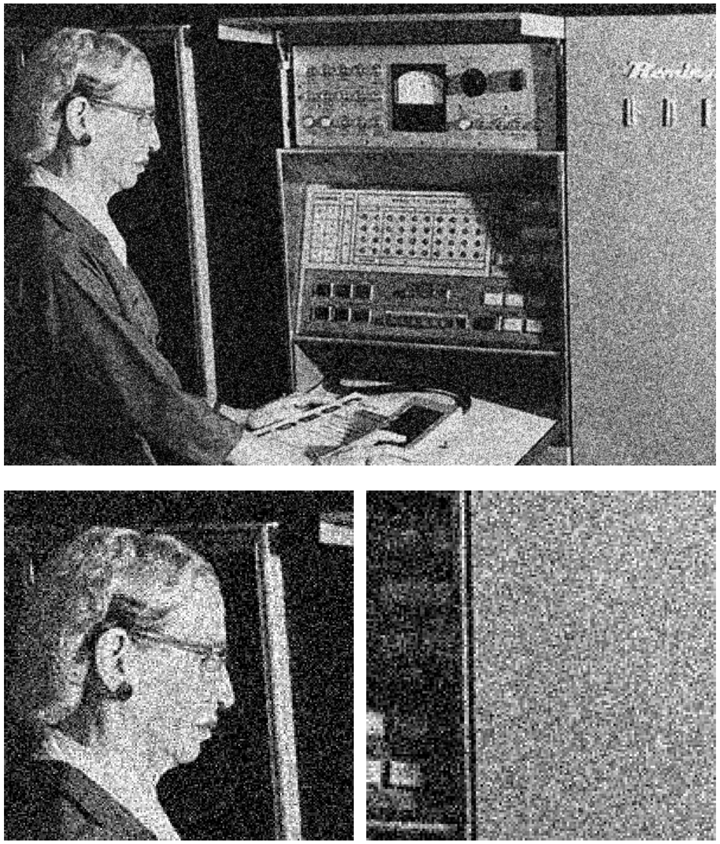
Adapted from D. Fouhey and J. Johnson

Gaussian vs. box filtering



Applying Gaussian filters

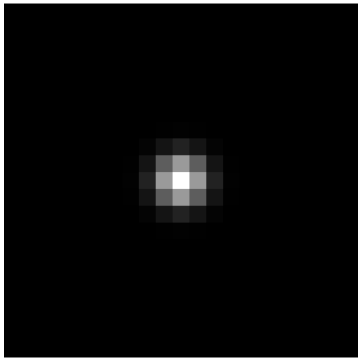
Input image
(no filter)

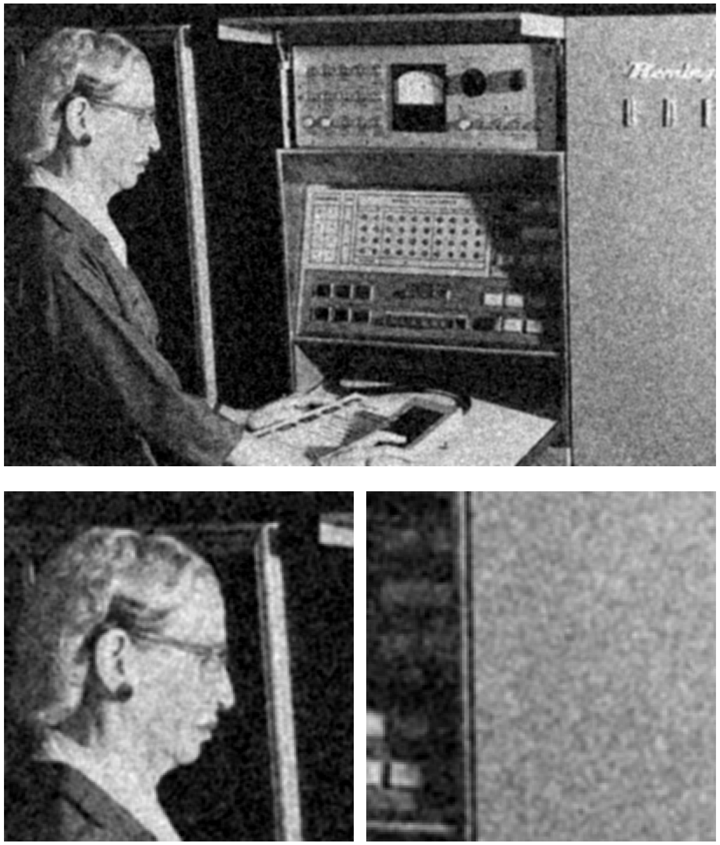


Source: D. Fouhey and J. Johnson

Applying Gaussian filters

$\sigma = 1$

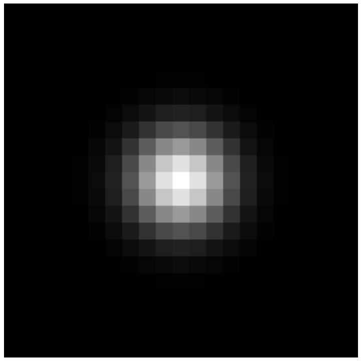


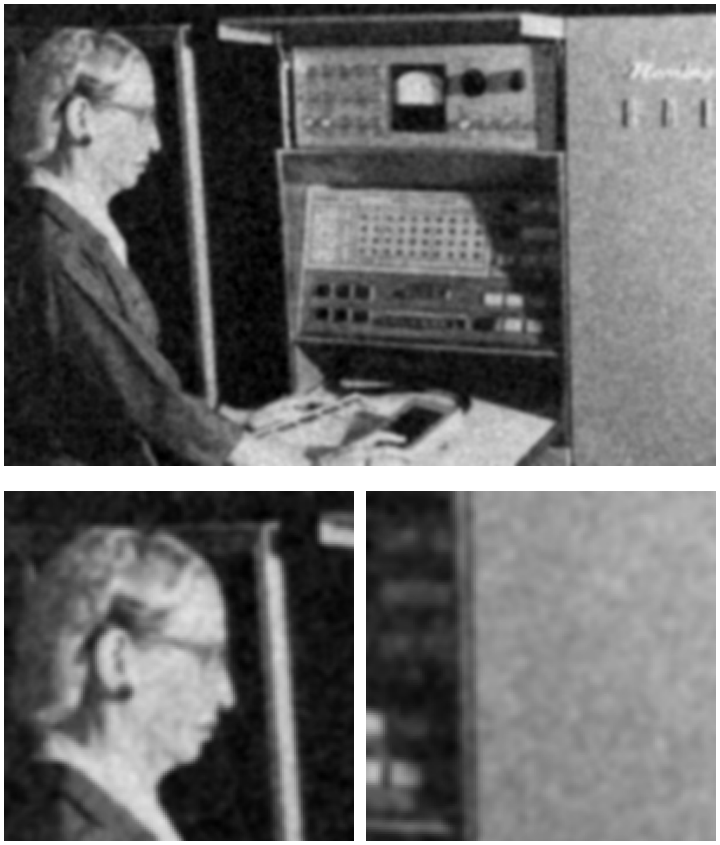


Source: D. Fouhey and J. Johnson

Applying Gaussian filters

$\sigma = 2$

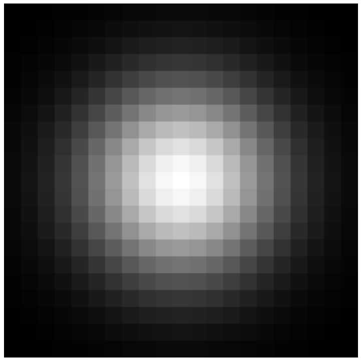


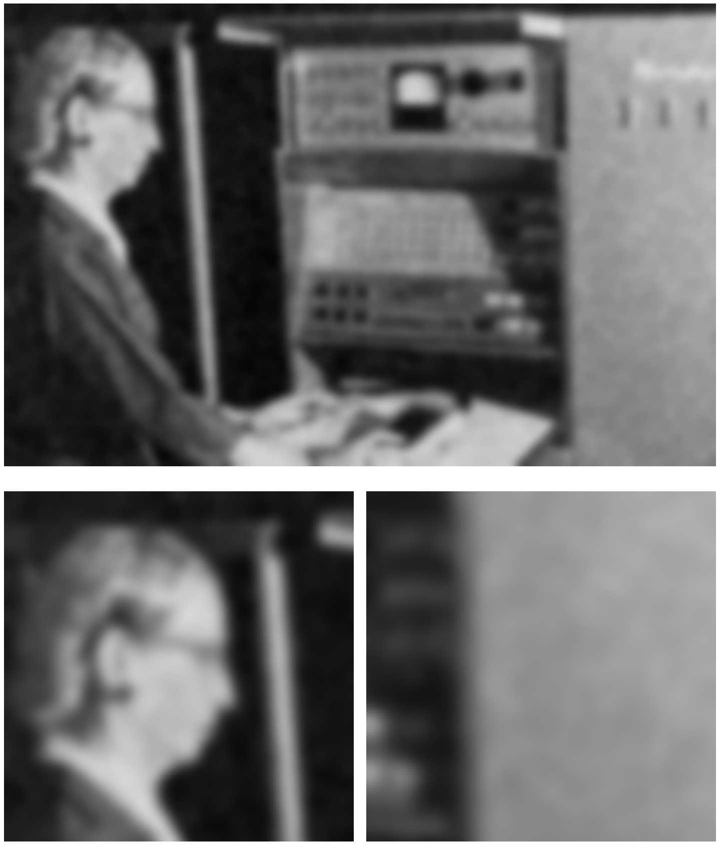


Source: D. Fouhey and J. Johnson

Applying Gaussian filters

$\sigma = 4$

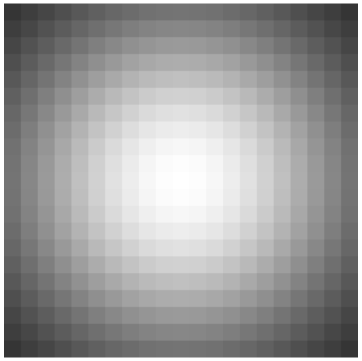


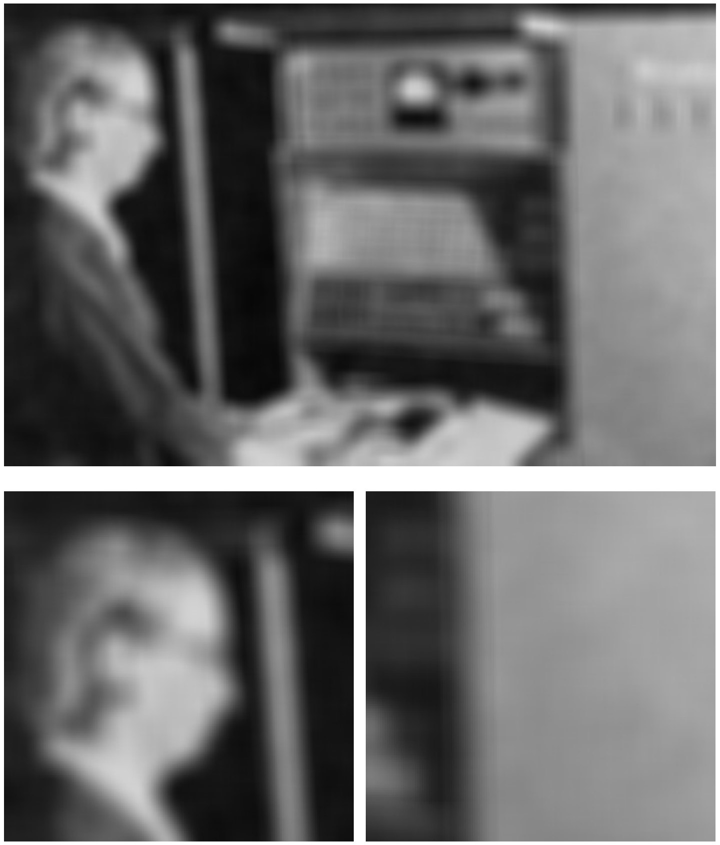


Source: D. Fouhey and J. Johnson

Applying Gaussian filters

$\sigma = 8$





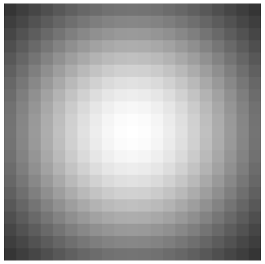
Source: D. Fouhey and J. Johnson

Choosing filter size

- Rule of thumb: set filter width to about 6σ (captures 99.7% of the energy)

$\sigma = 8$

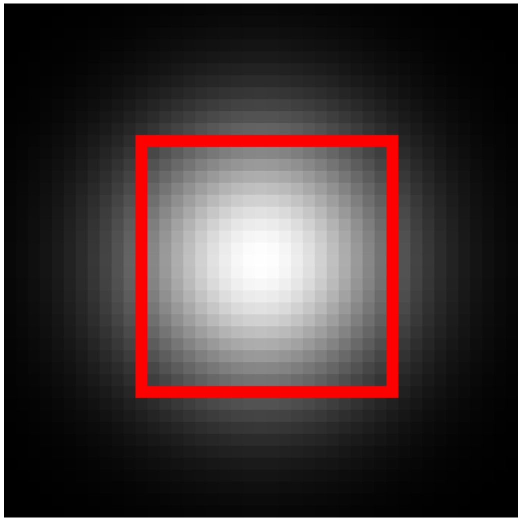
$\text{Width} = 21$



Too small!

$\sigma = 8$

$\text{Width} = 43$



A bit small (might be OK)

Adapted from D. Fouhey and J. Johnson

Gaussian filters: Properties

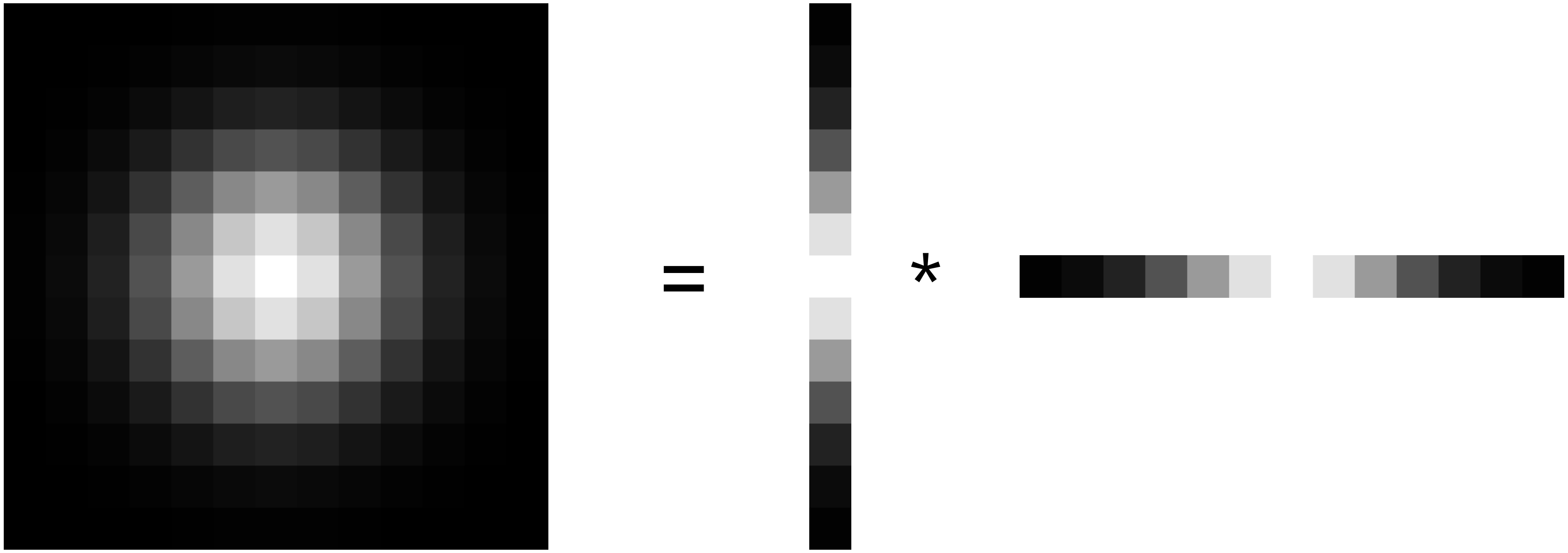
- Gaussian is a *low-pass filter*: it removes high-frequency components from the image (more on this soon)
- Convolution with self is another Gaussian
  - So we can smooth with small-σ kernel, repeat, and get same result as larger-σ kernel would have
  - Convolving *two times* with Gaussian kernel with std. dev. σ is the same as convolving *once* with kernel with std. dev. $\sigma\sqrt{2}$
- Gaussian kernel is *separable*: it factors into product of two 1D Gaussians

Source: K. Grauman

Separability of the Gaussian filter

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

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



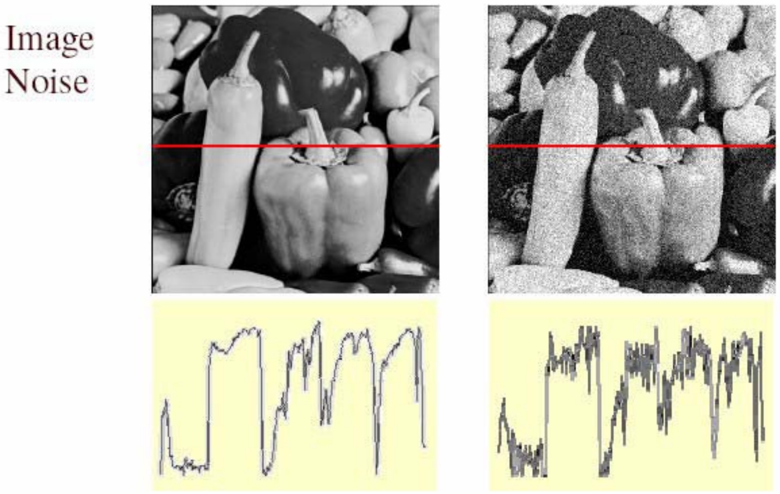
Adapted from D. Fouhey and J. Johnson

Why is separability useful?

- Separability means that 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 $m \times m$ kernel?
  - $O(n^2 m^2)$
- What if the kernel is separable?
  - $O(n^2 m)$

Gaussian noise

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



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

Gaussian i.i.d. (“white”) noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

Source: M. Hebert

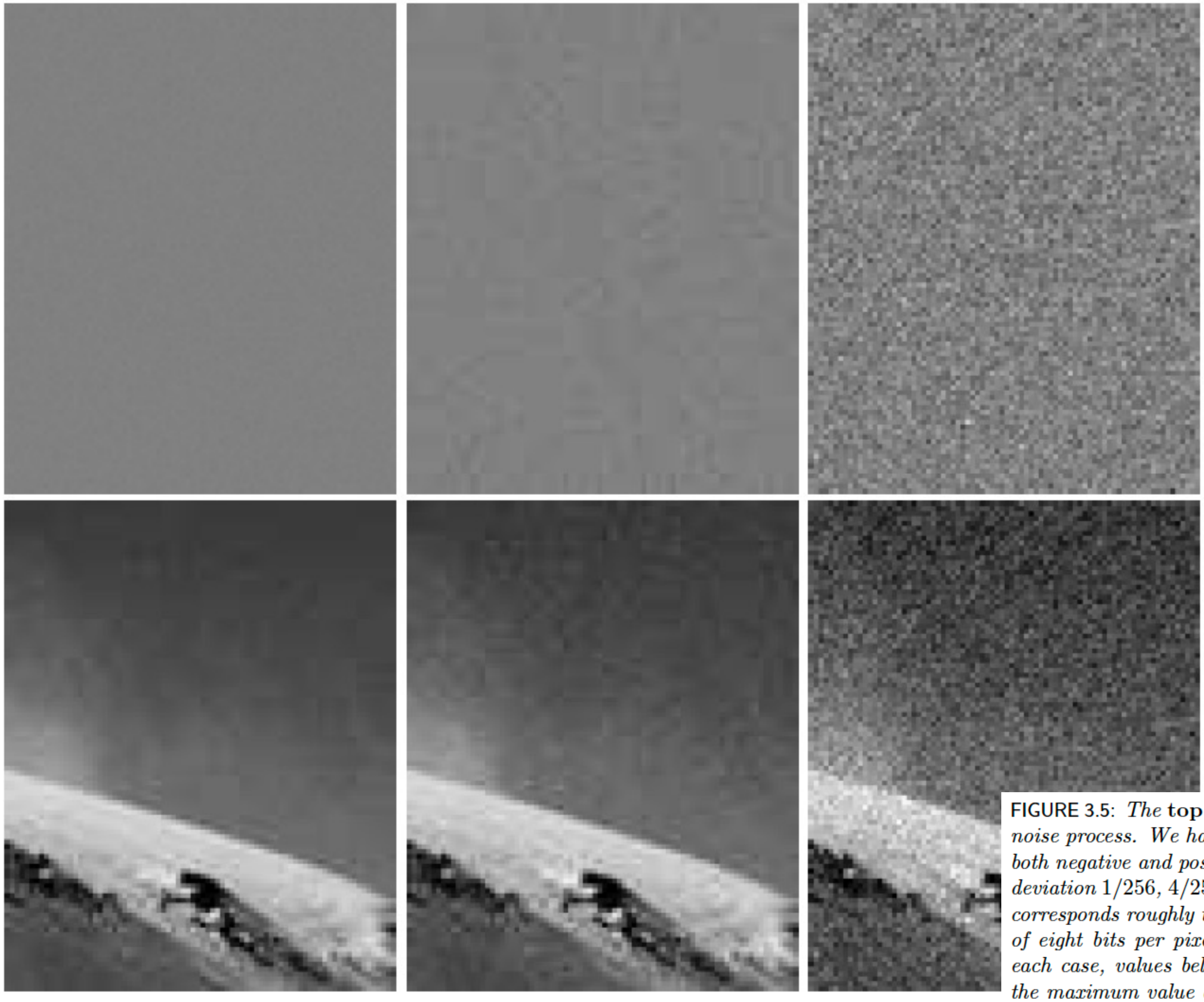


FIGURE 3.5: *The* **top row** *shows three realizations of a stationary additive Gaussian noise process. We have added half the range of brightnesses to these images to show both negative and positive values of noise. From left to right, the noise has standard deviation* $1/256$, $4/256$, *and* $16/256$ *of the full range of brightness, respectively. This corresponds roughly to bits zero, two, and five of a camera that has an output range of eight bits per pixel. The* **lower row** *shows this noise added to an image. In each case, values below zero or above the full range have been adjusted to zero or the maximum value accordingly.*

The effects of smoothing on Gaussian noise

Assume each image pixel is independent identically distributed Gaussian noise

I_ij ~ N(0, s^2)

Filter(I, F) ?

Mean: 0

Variance: sum_uv F_uv^2 s^2 --- good choice of F helps!

BUT pixels are no longer independent!

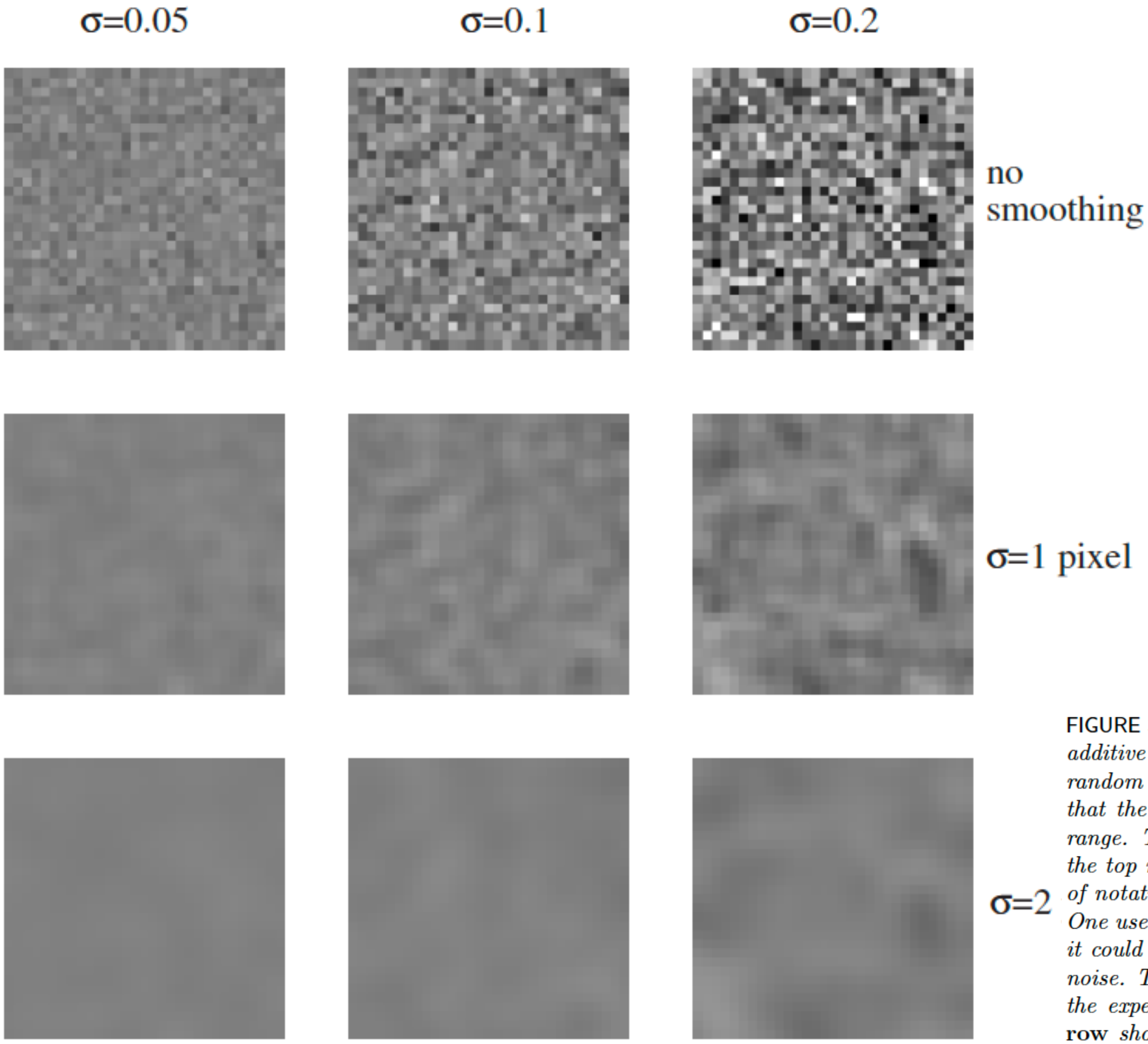


FIGURE 3.8: *The* **top row** *shows images of a constant mid-gray level corrupted by additive Gaussian noise. In this noise model, each pixel has a zero-mean normal random variable added to it. The range of pixel values is from zero to one, so that the standard deviation of the noise in the first column is about 1/20 of full range. The* **center row** *shows the effect of smoothing the corresponding image in the top row with a Gaussian filter of σ one pixel. Notice the annoying overloading of notation here; there is Gaussian noise and Gaussian filters, and both have σ's. One uses context to keep these two straight, although this is not always as helpful as it could be, because Gaussian filters are particularly good at suppressing Gaussian noise. This is because the noise values at each pixel are independent, meaning that the expected value of their average is going to be the noise mean. The* **bottom row** *shows the effect of smoothing the corresponding image in the top row with a Gaussian filter of σ two pixels.*