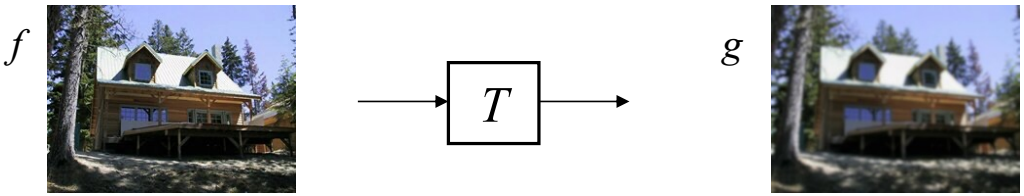
Image filtering

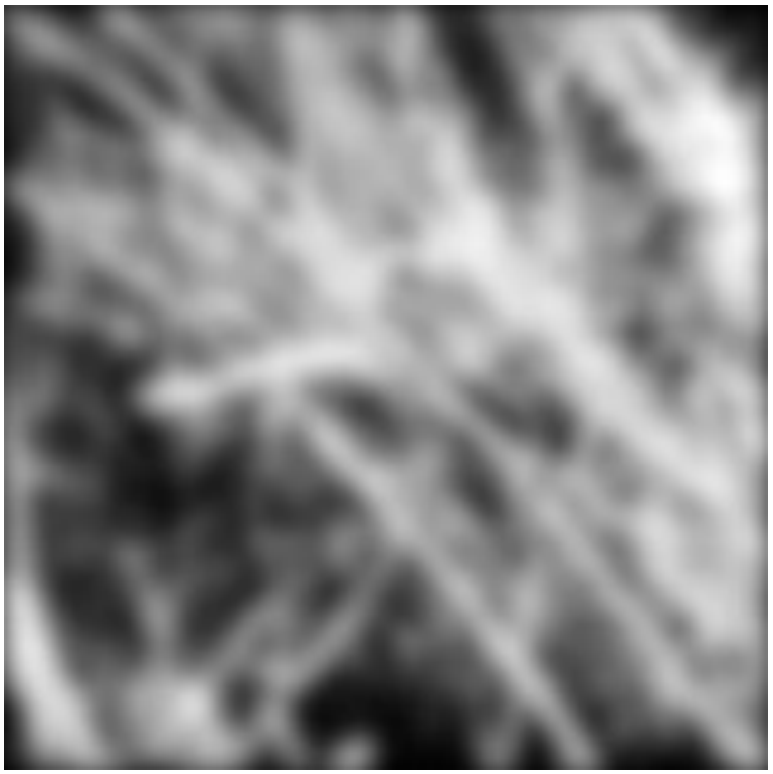
- Roughly speaking, replace image value at x with some function of values in its spatial neighborhood $N(x)$:

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



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

Image filtering



Topics:

- Movie I (this)
  - Linear filtering and convolution
  - Some properties, examples
- Movie II:
  - Denoising using linear filtering
  - Denoising with a non-linear filter
- Movie III:
  - Filtering as pattern detection
  - Finding derivatives with filters

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*

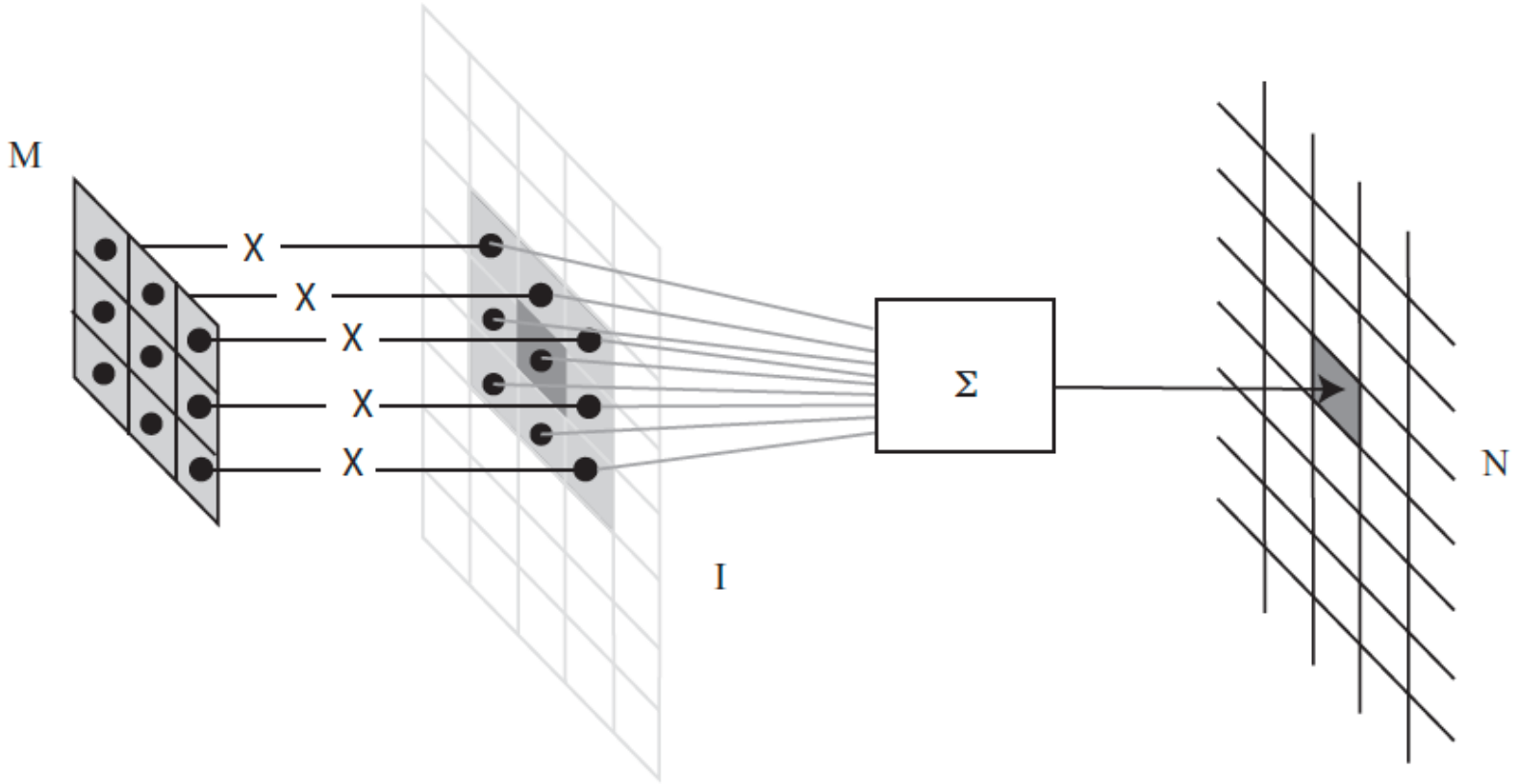
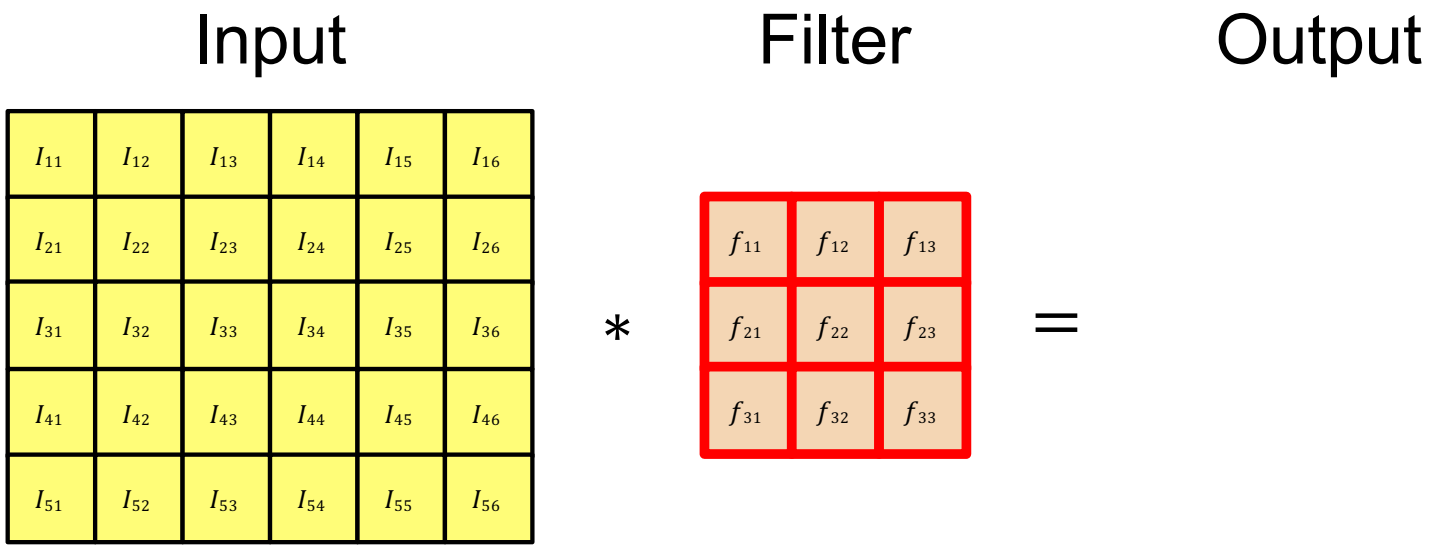


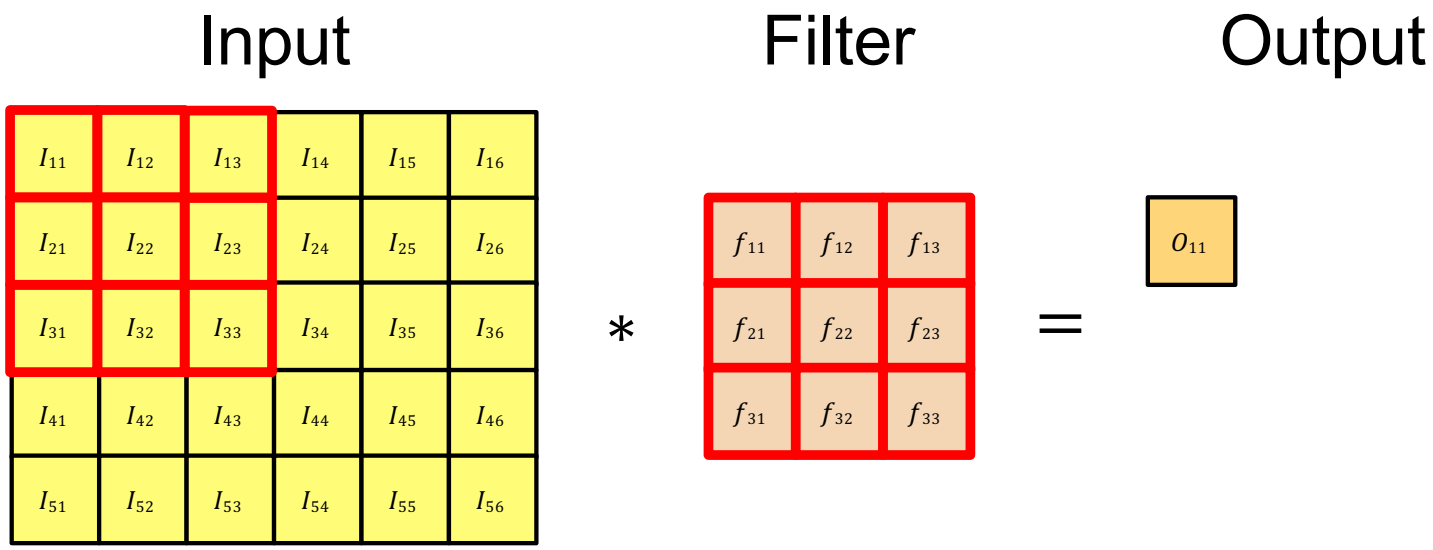
FIGURE 3.1: *To compute the value of $\mathcal{N}$ at some location, you shift a copy of $\mathcal{M}$ (the flipped version of $\mathcal{W}$) to lie over that location in $\mathcal{I}$; you multiply together the non-zero elements of $\mathcal{M}$ and $\mathcal{I}$ that lie on top of one another; and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

Applying a linear filter



Adapted from D. Fouhey and J. Johnson

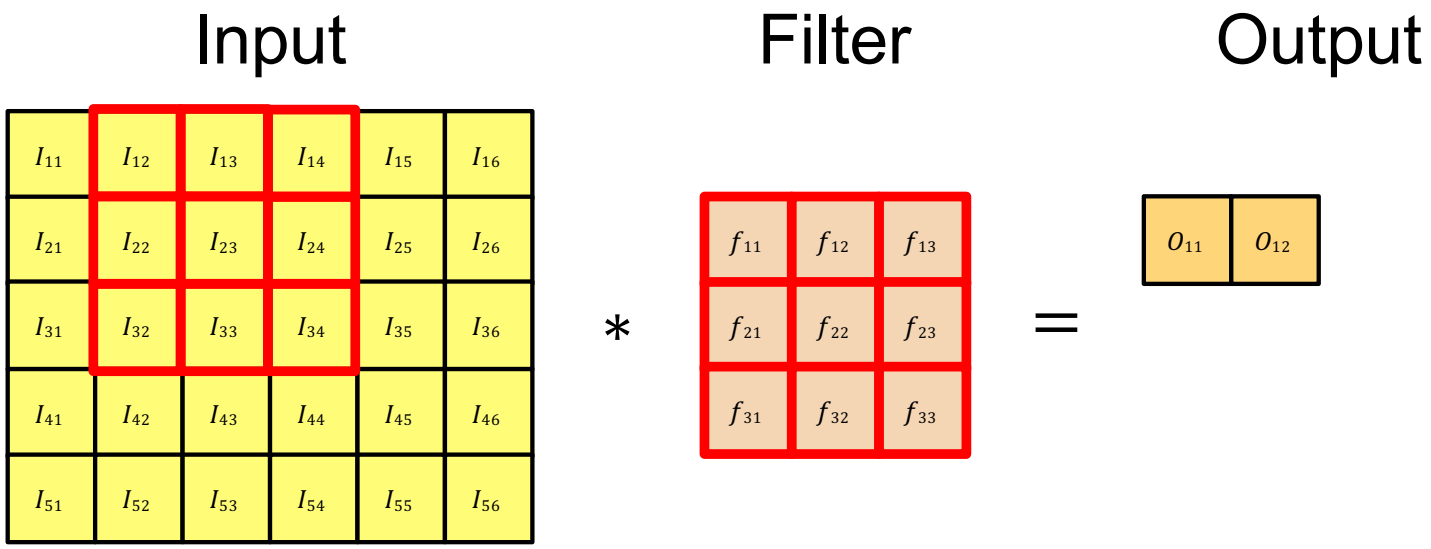
Applying a linear filter



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

Adapted from D. Fouhey and J. Johnson

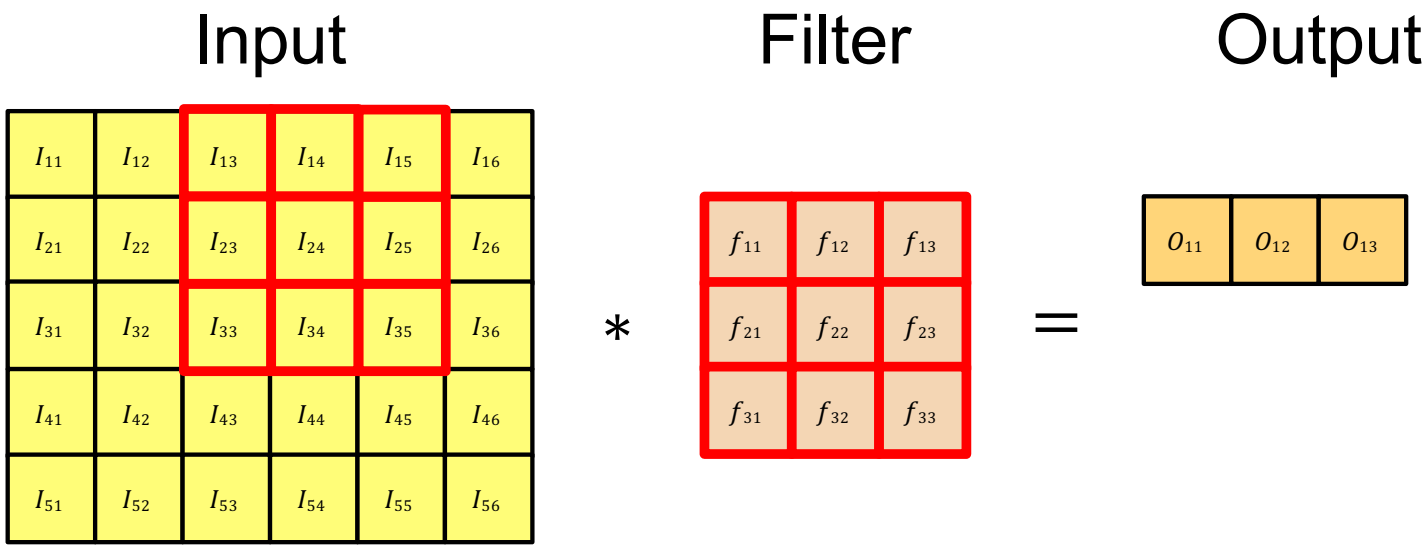
Applying a linear filter



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

Adapted from D. Fouhey and J. Johnson

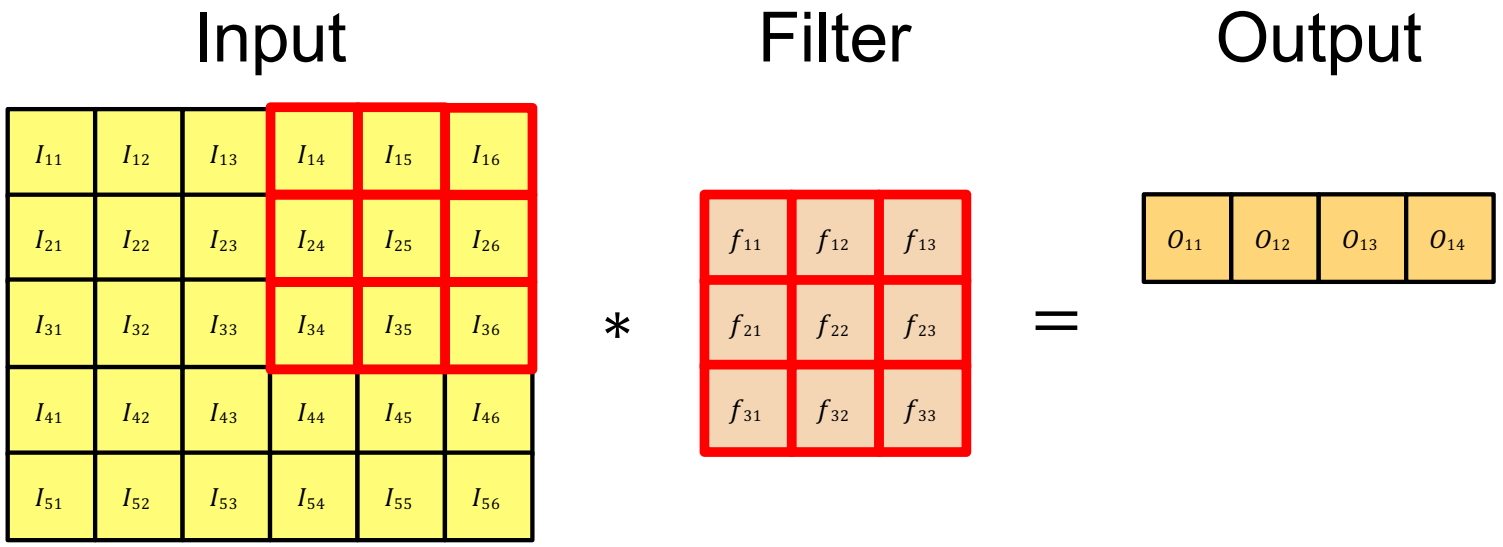
Applying a linear filter



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

Adapted from D. Fouhey and J. Johnson

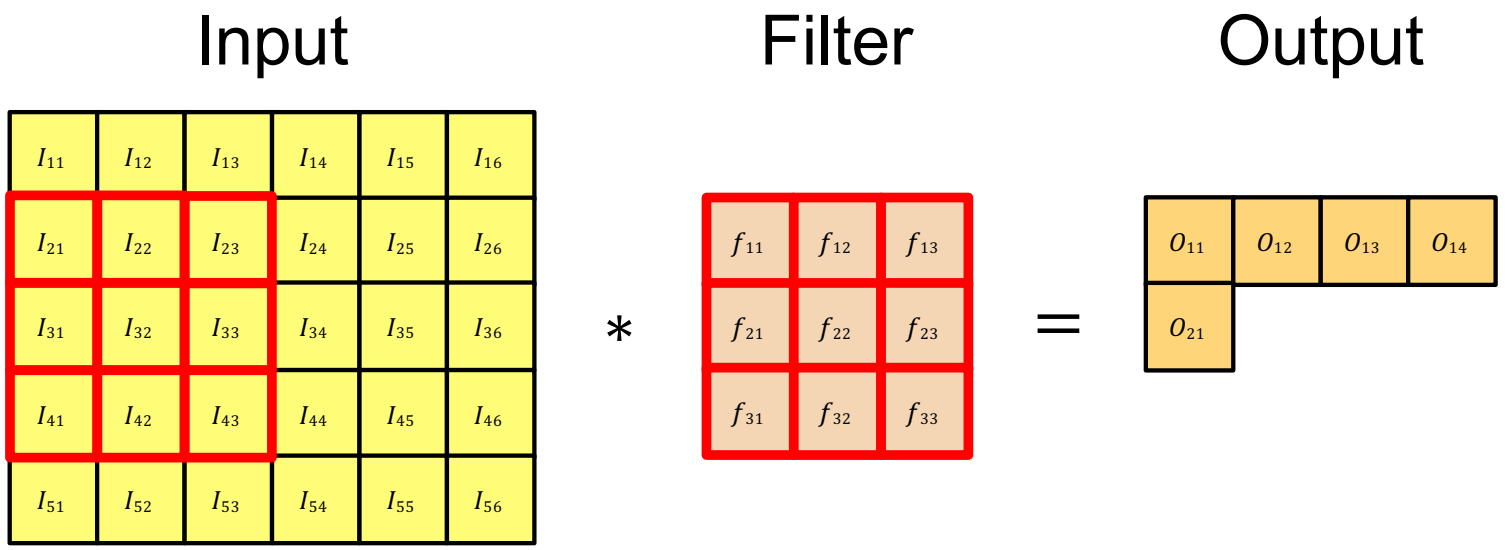
Applying a linear filter



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

Adapted from D. Fouhey and J. Johnson

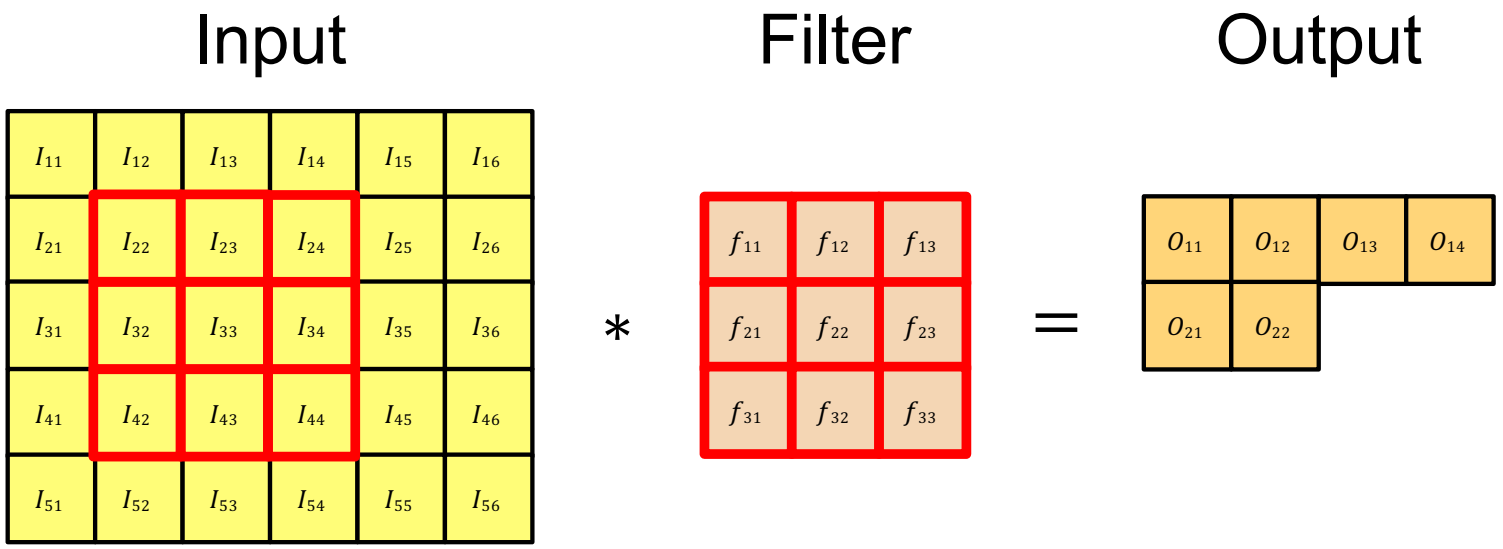
Applying a linear filter



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

Adapted from D. Fouhey and J. Johnson

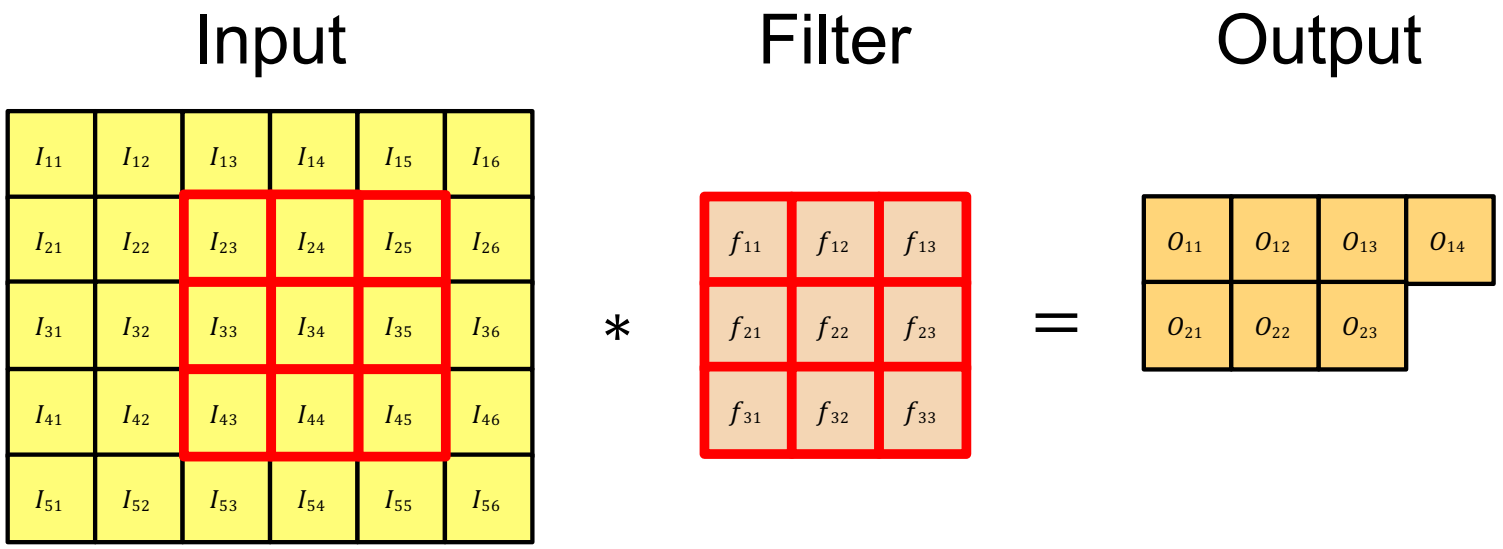
Applying a linear filter



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

Adapted from D. Fouhey and J. Johnson

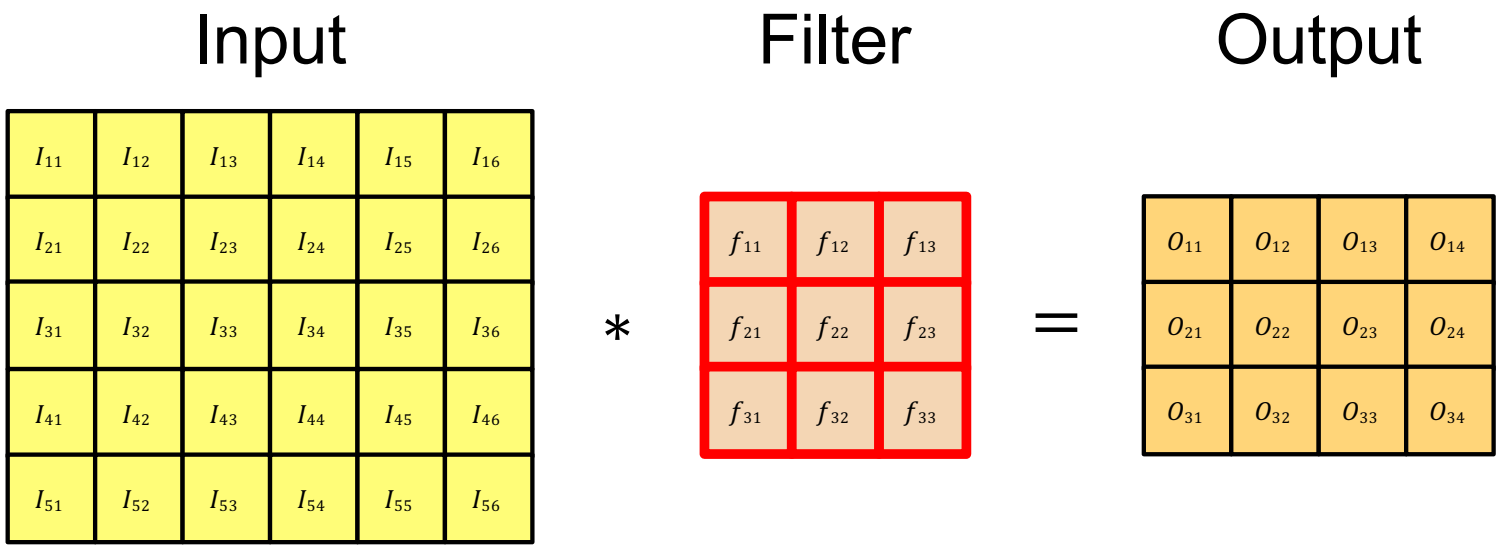
Applying a linear filter



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

Adapted from D. Fouhey and J. Johnson

Applying a linear filter



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

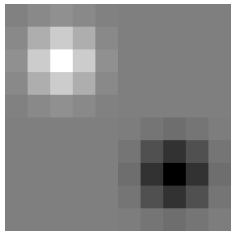
$$\mathcal{N} = \texttt{filter}(\mathcal{I}, \mathcal{W}).$$

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

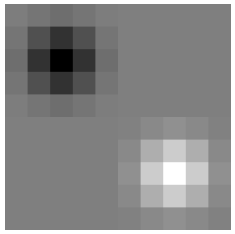
Note: Filtering vs. “convolution”

- 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

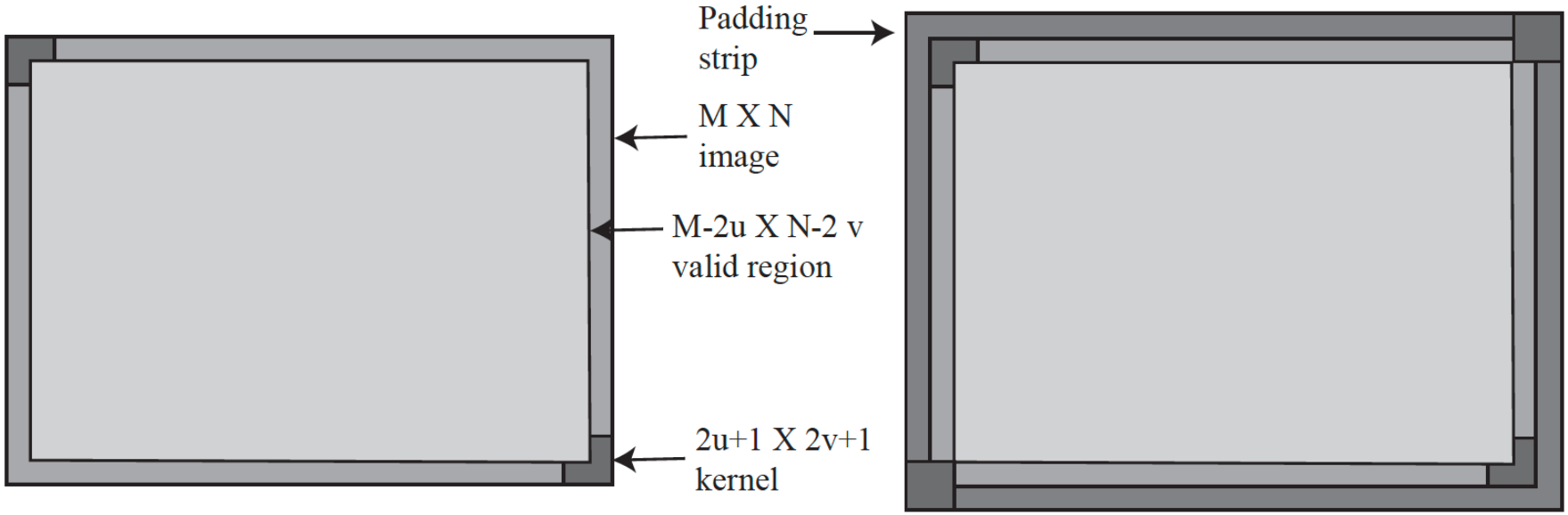


FIGURE 4.1: *The mid-gray box represents an $M \times N$ image, and the darker gray box a $2u + 1 \times 2v + 1$ kernel. The valid region is lighter gray. It can be constructed by placing the kernel at the top left and bottom right corners of the image, then constructing the box that joins their centers (***left***). A version of this construction reveals how the image should be padded to produce an $M \times N$ result. Place the center of the kernel at the bottom left and top right of the image, and construct the box that joins* their *outer corners (***right***).*

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

important property of convolution is that the result depends on the local pattern around a pixel, but not where the pixel is. Define the operation $\texttt{shift}(\mathcal{I}, m, n)$ which shifts an image so that the i, j'th pixel is moved to the $i-m$, $j-n$'th pixel, so

$$\texttt{shift}(\mathcal{I}, m, n)_{ij} = \mathcal{I}_{i-m,j-n}.$$

Ignore the question of the range, as `shift` just relabels pixel locations. Check that:

- Convolution is linear in the image, so

$$\begin{aligned} \mathcal{W} * (k\mathcal{I}) &= k(\mathcal{W} * \mathcal{I}) \\ \mathcal{W} * (\mathcal{I} + \mathcal{J}) &= \mathcal{W} * \mathcal{I} + \mathcal{W} * \mathcal{J}. \end{aligned}$$

- Convolution is linear in the mask, so

$$\begin{aligned} (k\mathcal{W}) * \mathcal{I}) &= k(\mathcal{W} * \mathcal{I}) \\ (\mathcal{W} + \mathcal{V}) * \mathcal{I}) &= \mathcal{W} * \mathcal{I} + \mathcal{V} * \mathcal{I} \end{aligned}$$

- Convolution is associative, so

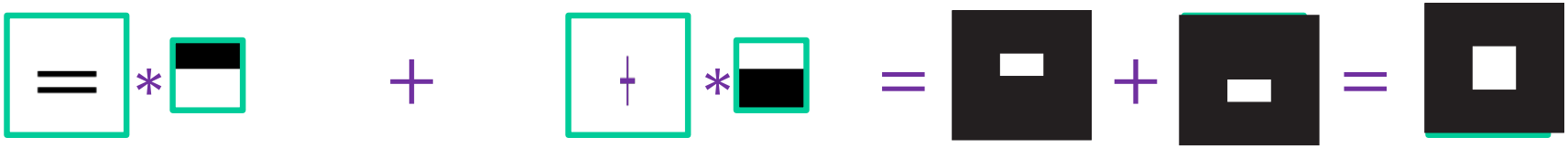
$$\mathcal{W} * (\mathcal{V} * \mathcal{I}) = (\mathcal{W} * \mathcal{V}) * \mathcal{I}.$$

- Convolution is shift-invariant, so

$$\mathcal{W} * (\texttt{shift}(\mathcal{I}, m, n)) = \texttt{shift}(\mathcal{W} * \mathcal{I}, m, n).$$

Properties: Linearity

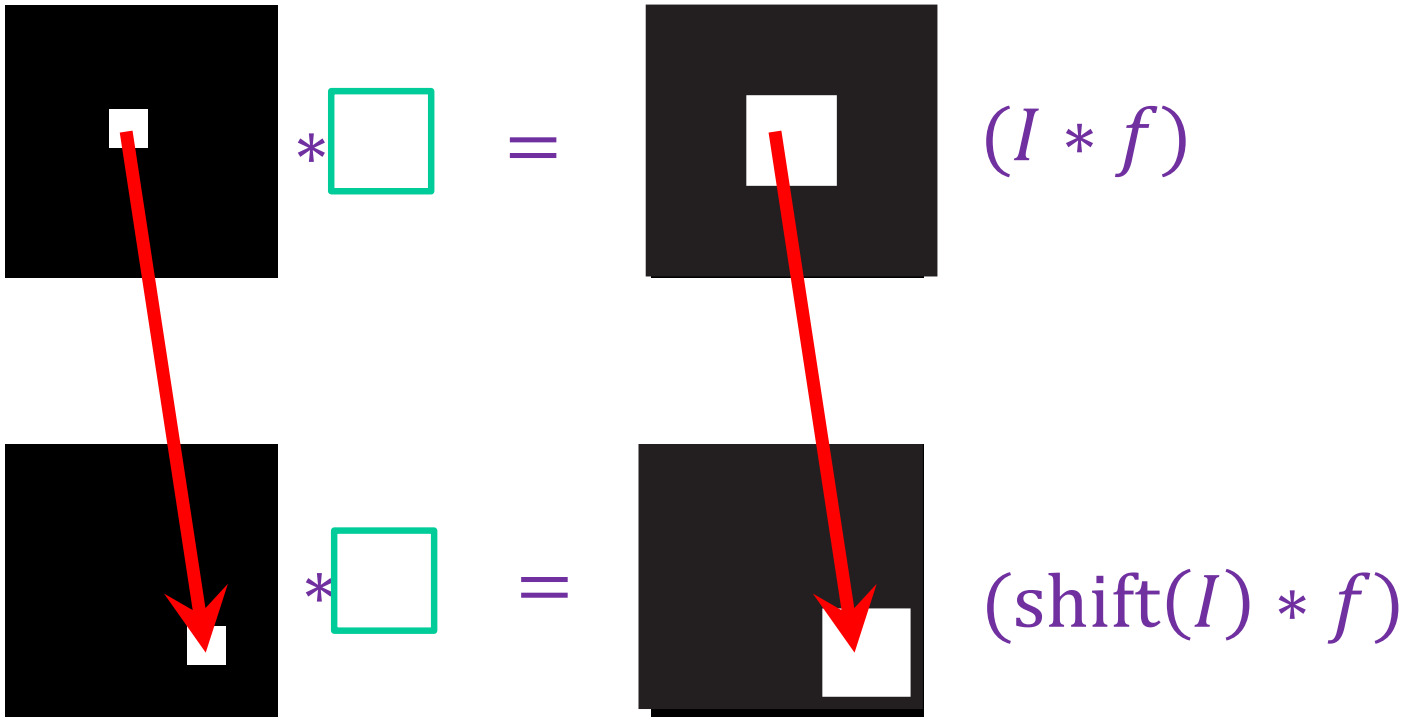
$$I * (f_1 + f_2) = I * f_1 + I * f_2$$



Adapted from D. Fouhey and J. Johnson

Properties: Shift-invariance

$$(\text{shift}(I) * f) = \text{shift}((I(* f))$$



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:

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

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

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

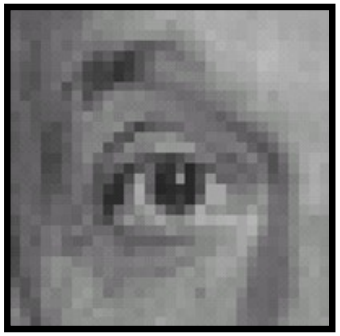
Source: D. Lowe

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

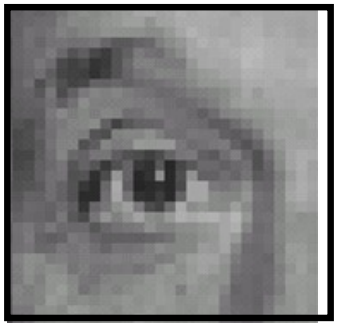
?

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

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

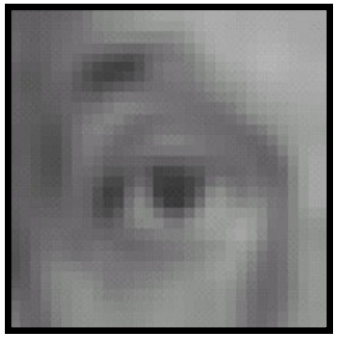
?

Source: D. Lowe

Practice with linear filters

Original

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



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters

Original

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

Source: D. Lowe

Practice with linear filters

Original

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

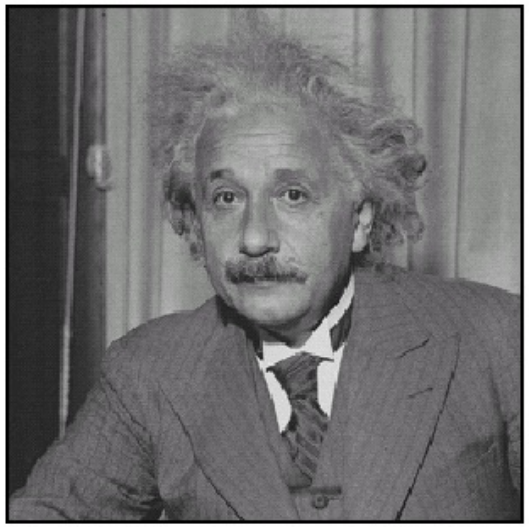
Sharpening filter:
Accentuates differences with local average

(Note that filter sums to 1)

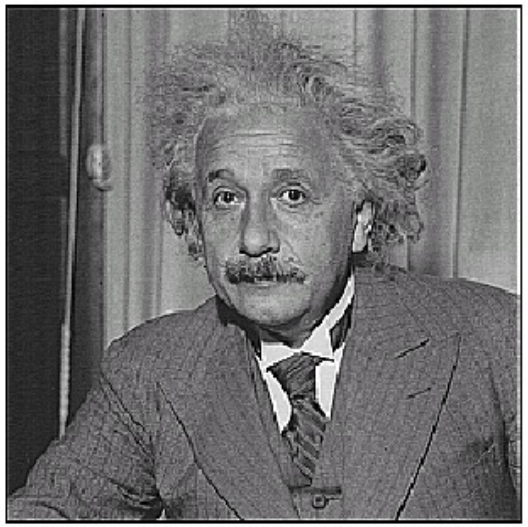
Sharpened

Source: D. Lowe

Sharpening



before



after

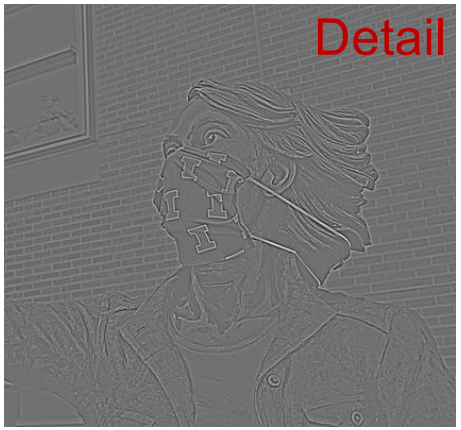
Source: D. Lowe

Sharpening

−

=

+



=

Source:
S. Gupta