CS4495/6495
Introduction to Computer Vision

2A-L3 *Linearity and convolution*

Linearity property

- In this lesson we’re going to finish up the basics of filtering so that next time we can apply it to some computer vision operations such as edge detection.
- We begin by developing some *linear* intuition.
- The reason that linearity is important will become clear in a just a little bit.

And now some linear intuition…

- An operator H (or system) is linear if two properties hold ($f1$ and $f2$ are some functions, a is a constant):

- Additivity (things sum):
  - $H(f1 + f2) = H(f1) + H(f2)$ (like distributive law)

- Multiplicative scaling (Homogeneity of degree 1):
  - $H(a \cdot f1) = a \cdot H(f1)$ (constant scales)

And now some linear intuition…

- An operator H (or system) is linear if two properties hold ($f1$ and $f2$ are some functions, a is a constant):

- Additivity (things sum):
  - $H(f1 + f2) = H(f1) + H(f2)$ (like distributive law)

- Multiplicative scaling (Homogeneity of degree 1):
  - $H(a \cdot f1) = a \cdot H(f1)$ (constant scales)

Because it is sums and multiplies, the "filtering" operation we were doing is linear.

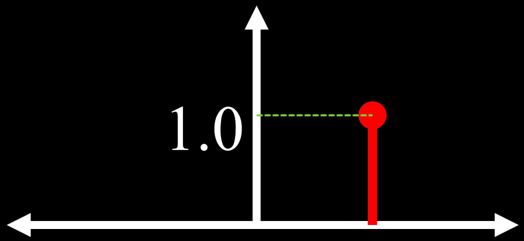
Quiz

Which of these operators are not linear:

a) Sum
b) Max
c) Average
d) Square root
e) (b) and (d)

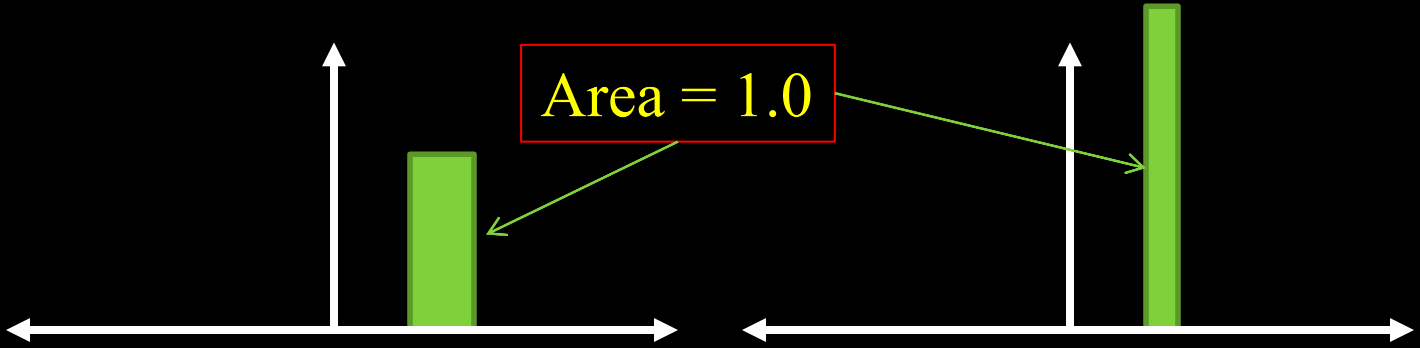
An impulse function…

- In the discrete world, an *impulse* is a very easy signal to understand: it's just a value of 1 at a single location.



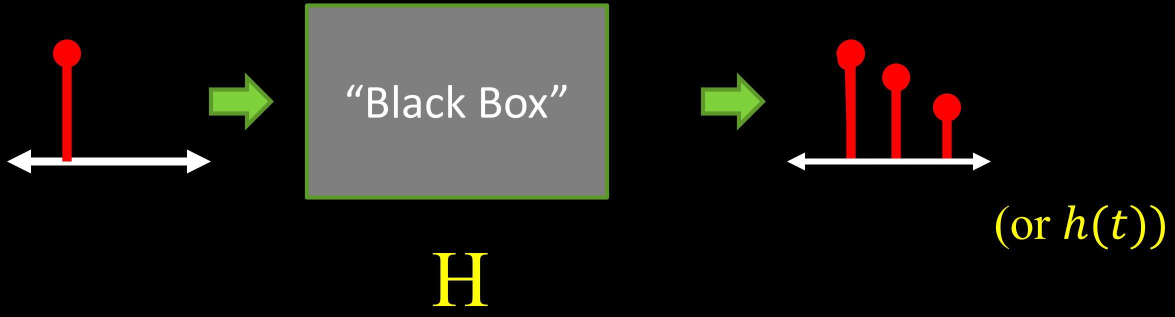
An impulse function…

- In the continuous world, an *impulse* is an idealized function that is very narrow and very tall so that it has a unit area.   In the limit:



An impulse response

- If I have an unknown system and I "put in" an impulse, the response is called the impulse response. (Duh?)



- So if the black box is linear you can describe H by $\boldsymbol{h(x)}$
- Why?

Filtering an impulse signal

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

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$\otimes$

a	b	c
d	e	f
g	h	i

=

?

F(x,y) ***H(u,v)*** ***G(x,y)***

Filtering an impulse signal

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

F(x,y)

$\otimes$

a	b	c
d	e	f
g	h	i

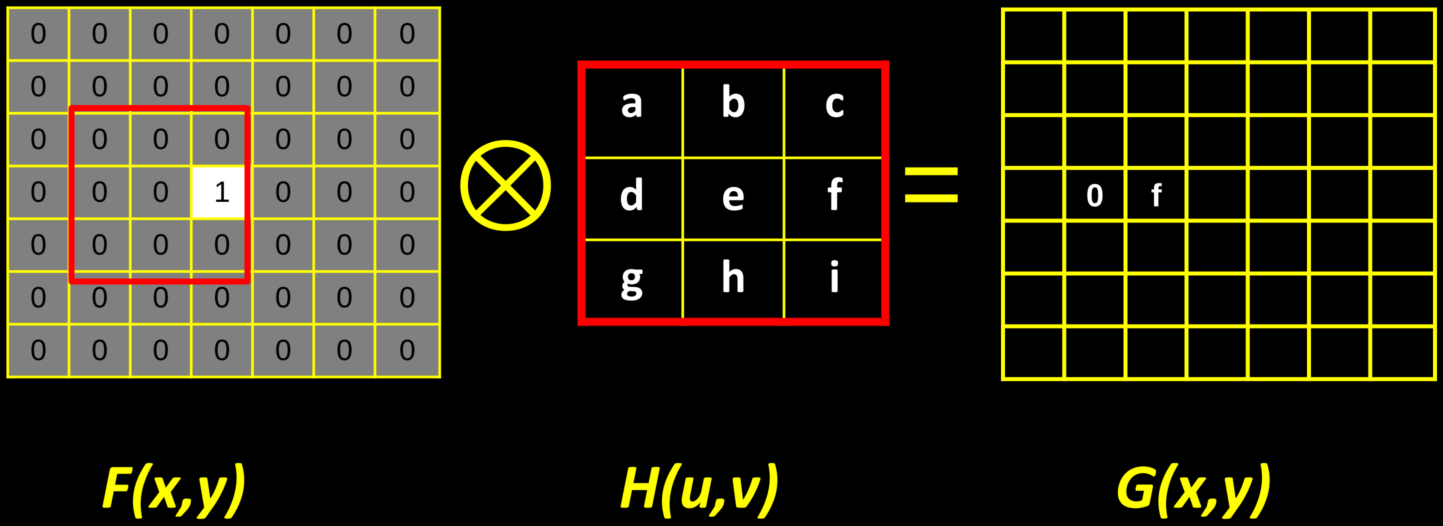
H(u,v)

$=$

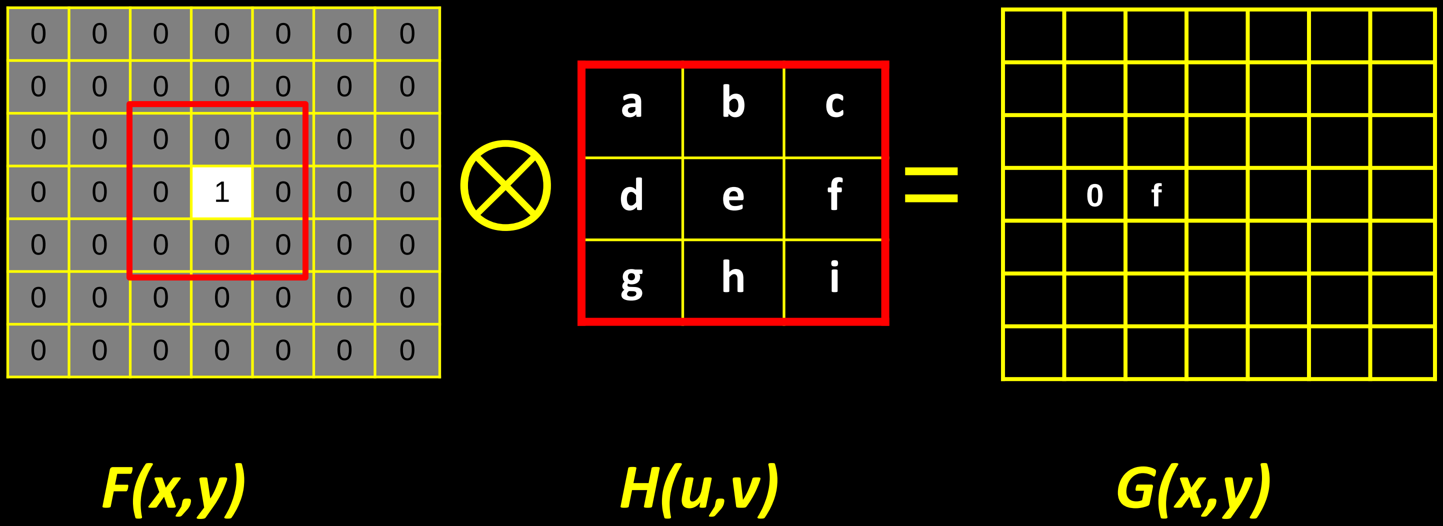
	0					

G(x,y)

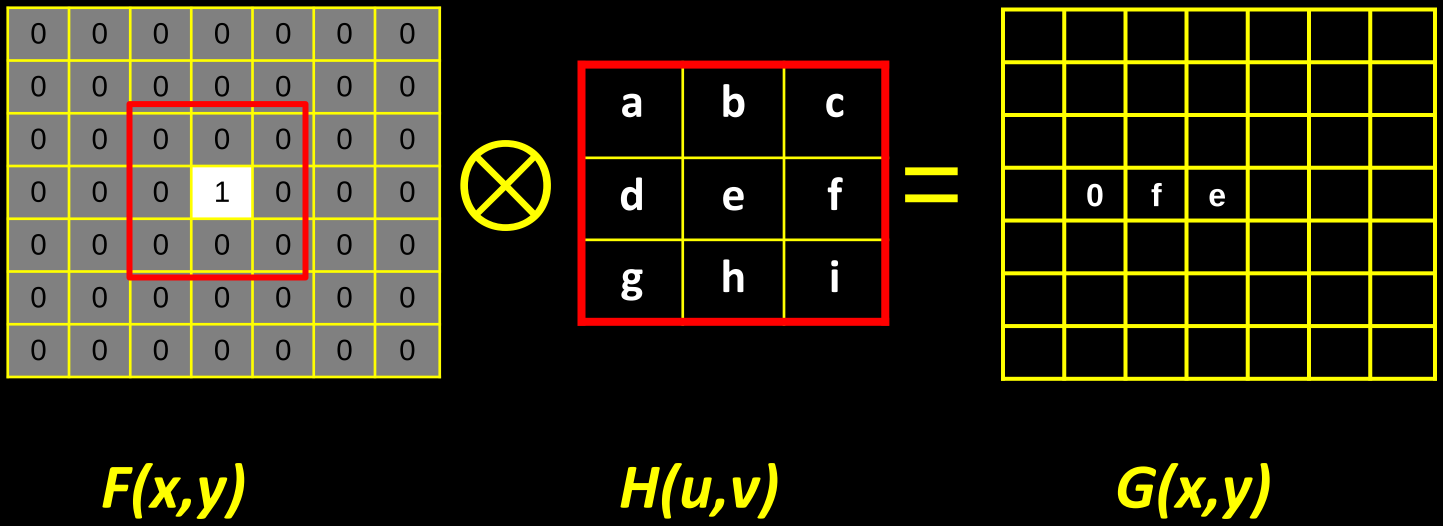
Filtering an impulse signal



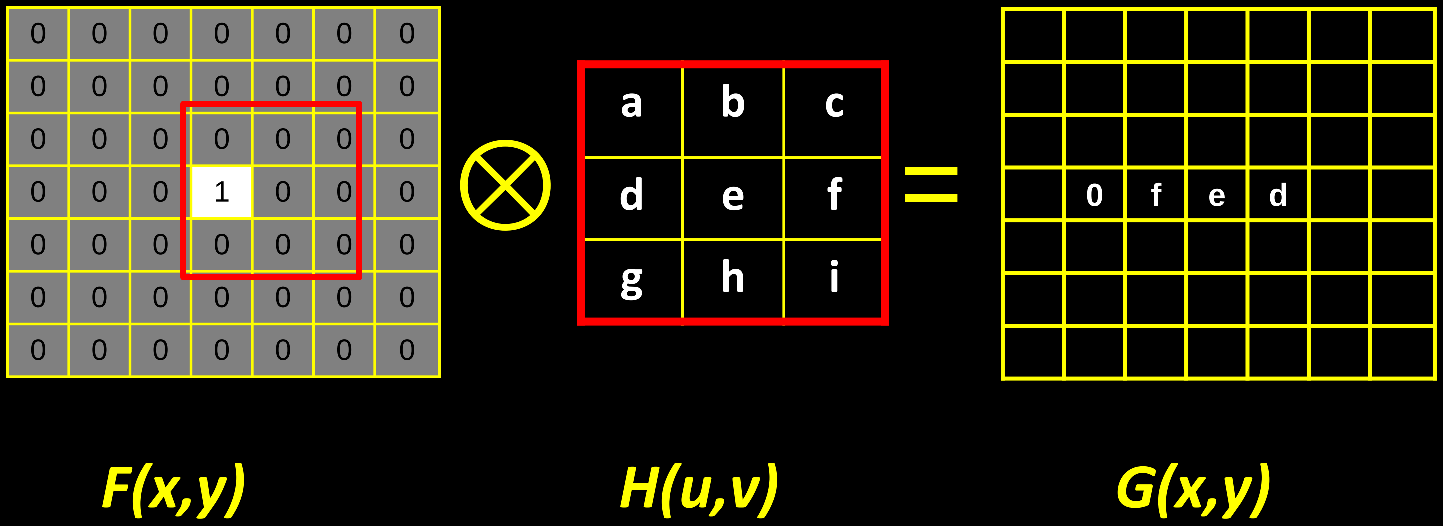
Filtering an impulse signal



Filtering an impulse signal



Filtering an impulse signal



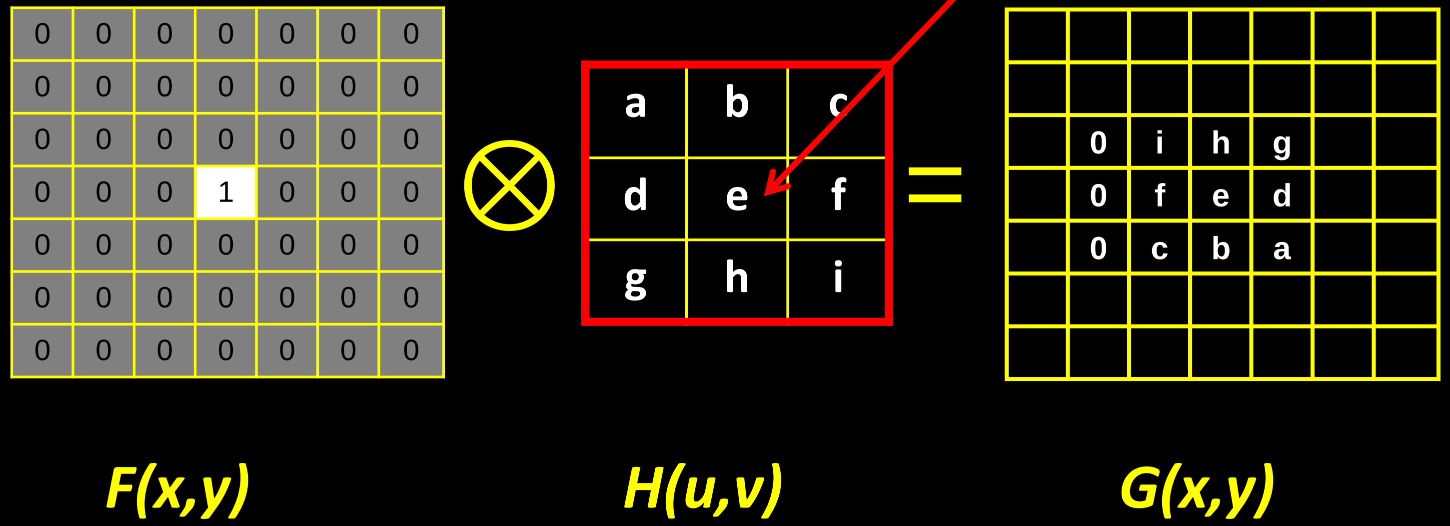
Filtering an impulse signal

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\otimes
\begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & i
\end{bmatrix}
=
\begin{bmatrix}
 & & & & & & \\
 & & & & & & \\
 & 0 & i & h & g & & \\
 & 0 & f & e & d & & \\
 & 0 & c & b & a & & \\
 & & & & & & \\
 & & & & & &
\end{bmatrix}
$$

F(x,y) ***H(u,v)*** ***G(x,y)***

Filtering an impulse signal

Assuming center coordinate is "reference point".



Quiz

Suppose our kernel was size MxM and our image was NxN. How many multiplies would it take to filter the who image with the filter?

a) M*N*2
b) M*M*N*2
c) M*N*N
d) M*M*N*N

Correlation vs Convolution

Cross-correlation

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

$$G = H \otimes F$$

Flip in both dimensions
(bottom to top, right to left)

Convolution

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

$$G = H \star F$$

For a Gaussian or box filter, how will the outputs differ?

K. Grauman

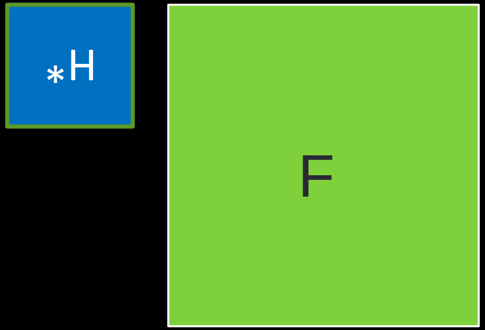
Convolution

Centered at zero!

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

$$G = H * F$$

Notation for convolution operator



K. Grauman

Quiz

When convolving a filter with an impulse image, we get the filter back as a result.

So if we convolve an image with an impulse we get:

a) A blurred version of the image
b) The original image
c) A shifted version of the original image.
d) No idea

One more thing…

Shift invariant:

- Operator behaves the same everywhere, i.e. the value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood.

Properties of convolution

- Linear & shift invariant
- Commutative:

  f * g = g * f

- Associative

  (f * g) * h = f * (g * h)

- Identity:

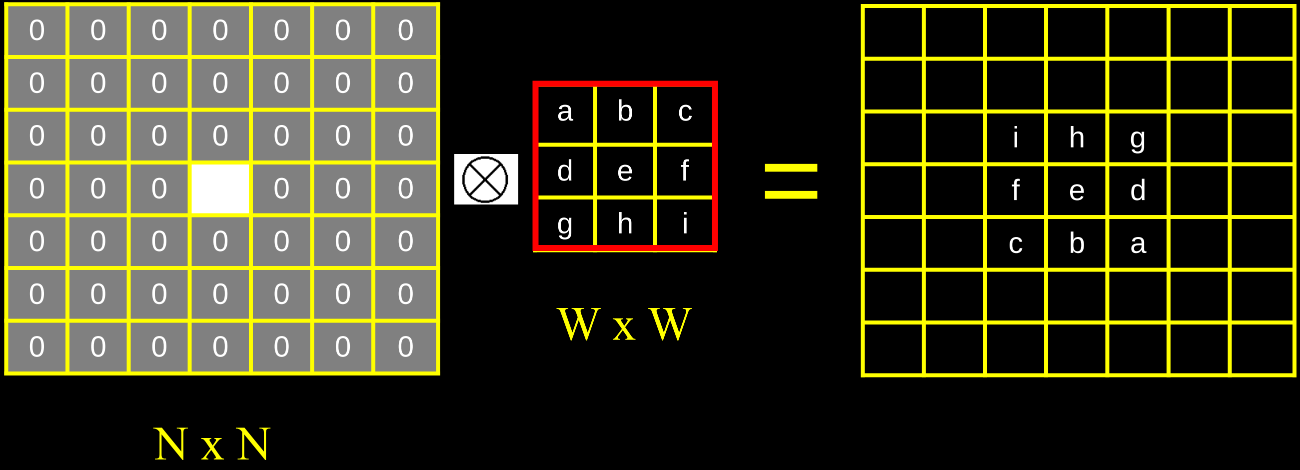
  unit impulse e = [..., 0, 0, 1, 0, 0, ...]. f * e = f

- Differentiation: $$\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$$

We'll use this later!

Computational Complexity

- If an image is *NxN* and a kernel (filter) is WxW, how many multiplies do you need to compute a convolution?



- You need N*N*W*W = N^2W^2
  - which can get big (ish)

Separability

- In some cases, filter is separable, meaning you can get the square kernel H by convolving a single column vector by some row vector:

$$c\;\begin{bmatrix}1\\2\\1\end{bmatrix} \times \overset{r}{\begin{bmatrix}1&2&1\end{bmatrix}} = \begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\;H$$

Separability

$$\mathbf{c}\;\begin{bmatrix}1\\2\\1\end{bmatrix} \times \overset{\mathbf{r}}{\begin{bmatrix}1&2&1\end{bmatrix}} = \begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\;\mathbf{H}$$

$$G = H * F = (C * R) * F = C * (R * F)$$

- So we do two convolutions but each is W*N*N. So this is useful if W is big enough such that $2 \cdot W \cdot N^2 << W^2 \cdot N^2$
- Used to be ***very*** important. Still, if W=31, save a factor of 15.

Quiz

True or false: Division is a linear operation.

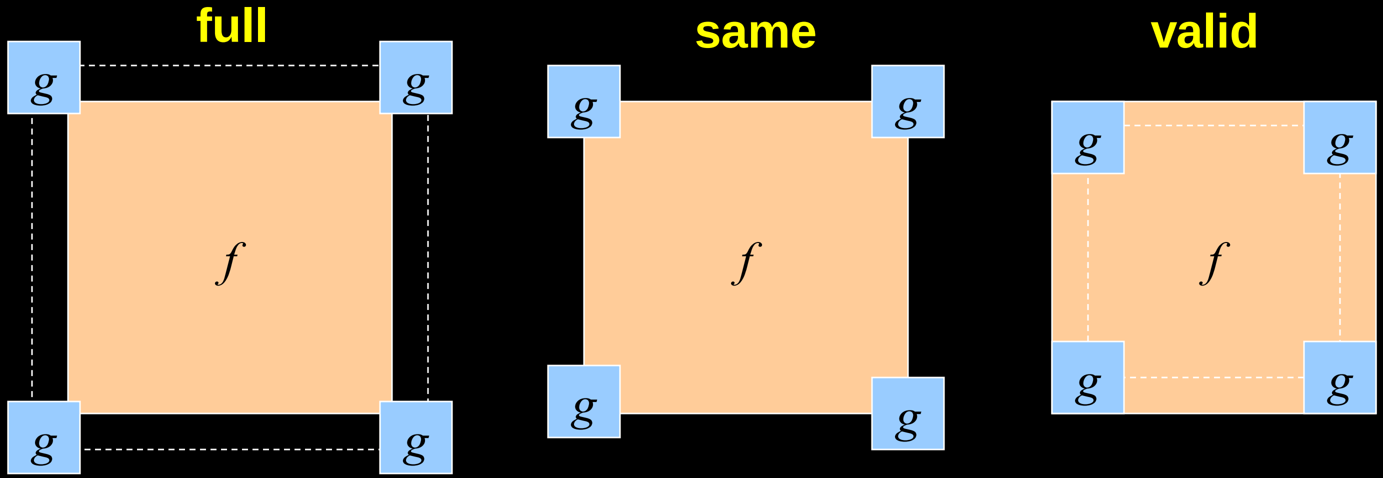
a) False because $X/(Y+Z) \neq X/Y + X/Z$

b) True because $(X+Y)/Z = X/Z + Y/Z$

c) I have no idea

Boundary issues

- What is the size of the output?
- Using old Matlab nomenclature we have three choices:



Source: S. Lazebnik

Boundary issues

What about near the edge?

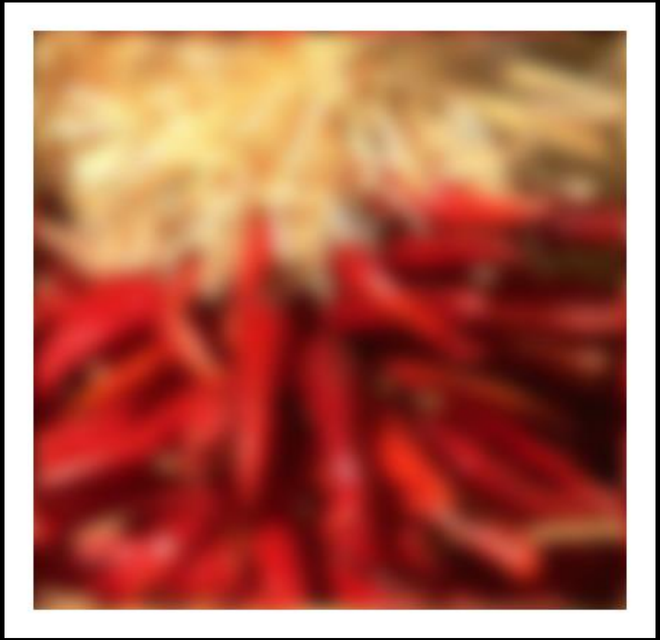
- methods:
  - clip filter (black)

Source: S. Marschner

Boundary issues

What about near the edge?

- methods:
  - clip filter (black)
  - wrap around

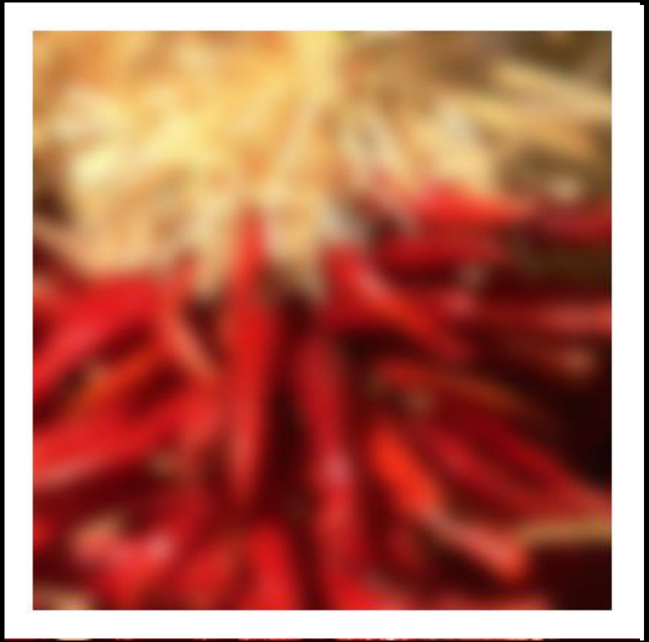


Source: S. Marschner

Boundary issues

What about near the edge?

- methods:
  - clip filter (black)
  - wrap around
  - copy edge



Source: S. Marschner

Boundary issues

What about near the edge?

- methods:
  - clip filter (black)
  - wrap around
  - copy edge
  - reflect across edge

Source: S. Marschner

Boundary issues

What about near the edge?

- methods (new MATLAB):
  - clip filter (black): imfilter(f, g, 0)
  - wrap around: imfilter(f, g, ‘circular’)
  - copy edge: imfilter(f, g, ‘replicate’)
  - reflect across edge: imfilter(f, g, ‘symmetric’)

Quiz

The reflection method of handling boundary conditions in filtering is good because:

a) The created imagery has the same statistics as the original image
b) The computation is the least expensive of the possibilities.
c) Setting pixels to zero is fast.
d) None of the above.

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

Filtered
(no change)

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

?

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

Center coordinate is 0,0!

Shifted left by 1 pixel with correlation

Practice with linear filters

Original

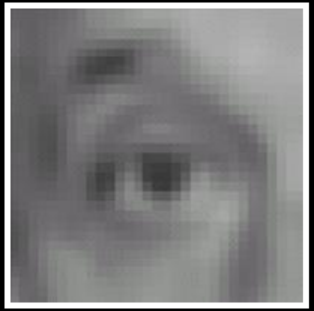
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

?

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)

Practice with linear filters

Original

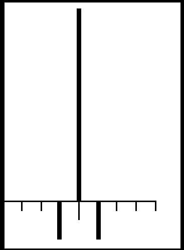
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

?

Practice with linear filters

Original

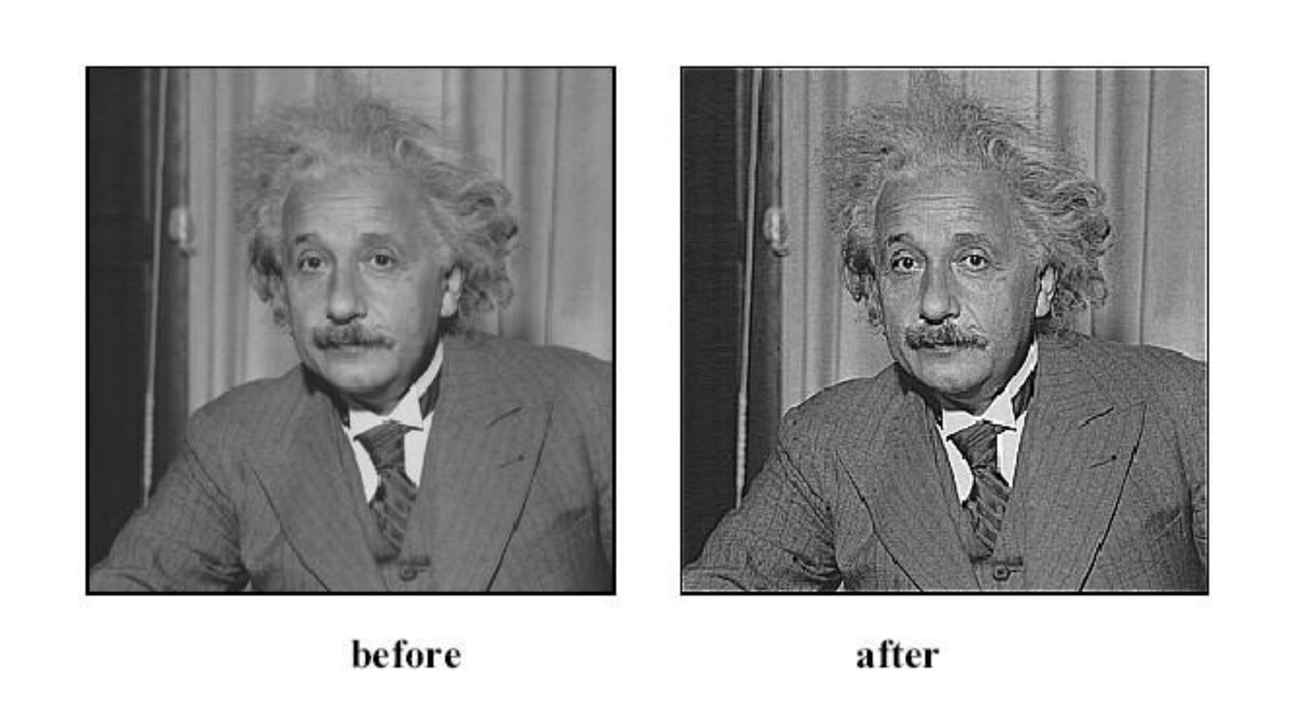
$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$



Sharpening filter

- Accentuates differences with local average

Filtering examples: sharpening



K. Grauman

Quiz

If a filter's coefficients don't add to 1.0 they can be corrected by multiplying by the necessary scale value. Or the resulting image can be multiplied by the square root of that number after the operation to compensate for the horizontal and vertical application of the filter.

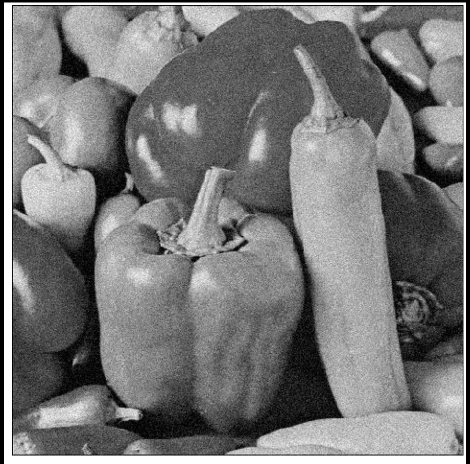
a) True

b) False

Different kinds of noise

- We said that Gaussian averaging was a reasonable thing to do if the noise was independent at each pixel and centered about zero such as if created by a Gaussian or normal noise process.
- But there are other kinds of noise.

Different kinds of noise



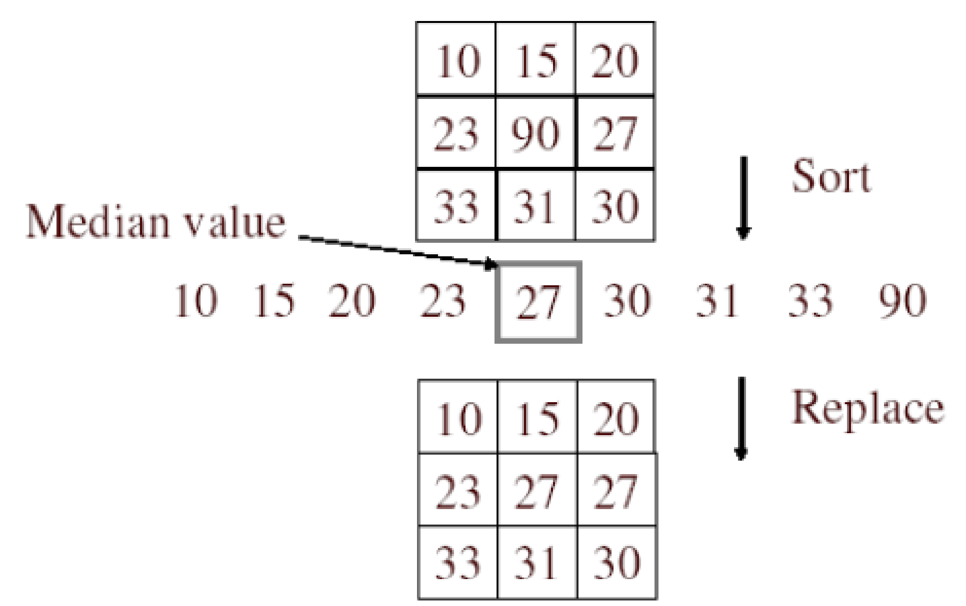
Additive Gaussian noise

Salt and pepper noise

Recall: Assumptions for removing noise

- The way to approach this is to remember our other assumption about images: that the underlying image changes smoothly around a pixel. So if you were to inject some arbitrary value in there the question is how to find the original pixel.
- Remember that when we are the blurring, we ***replace*** the pixel value by the local average. That's fine when the noise is modest and tends to add to zero over a neighborhood. But if there are a few totally random values thrown in, we need another approach.
- As many of you know the way to deal with such ugly perturbation is to use what's called a ***median*** filter.

Median filter

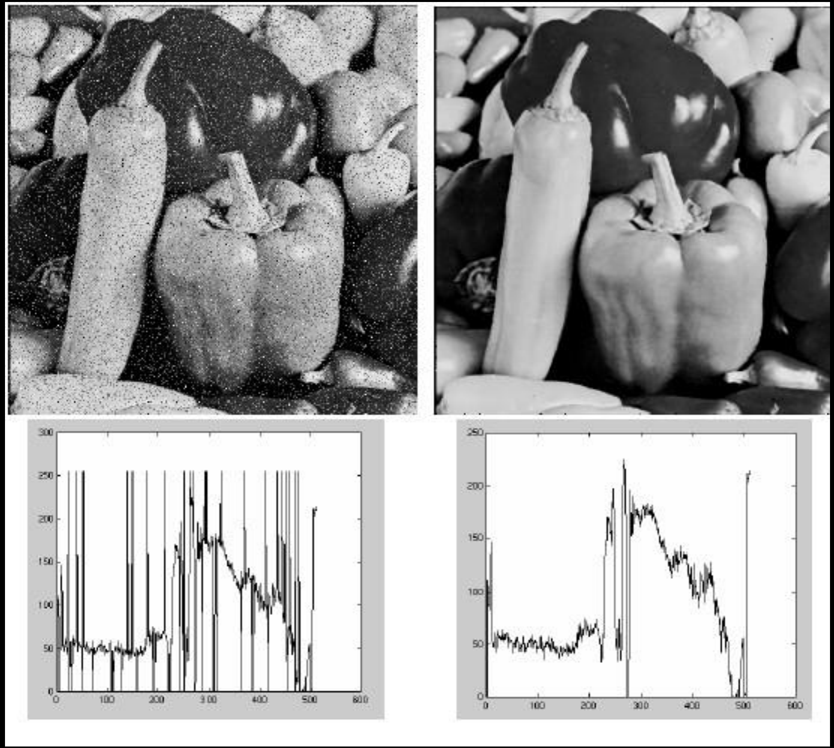


- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Linear?

K. Grauman

Salt and pepper noise

Median filtered

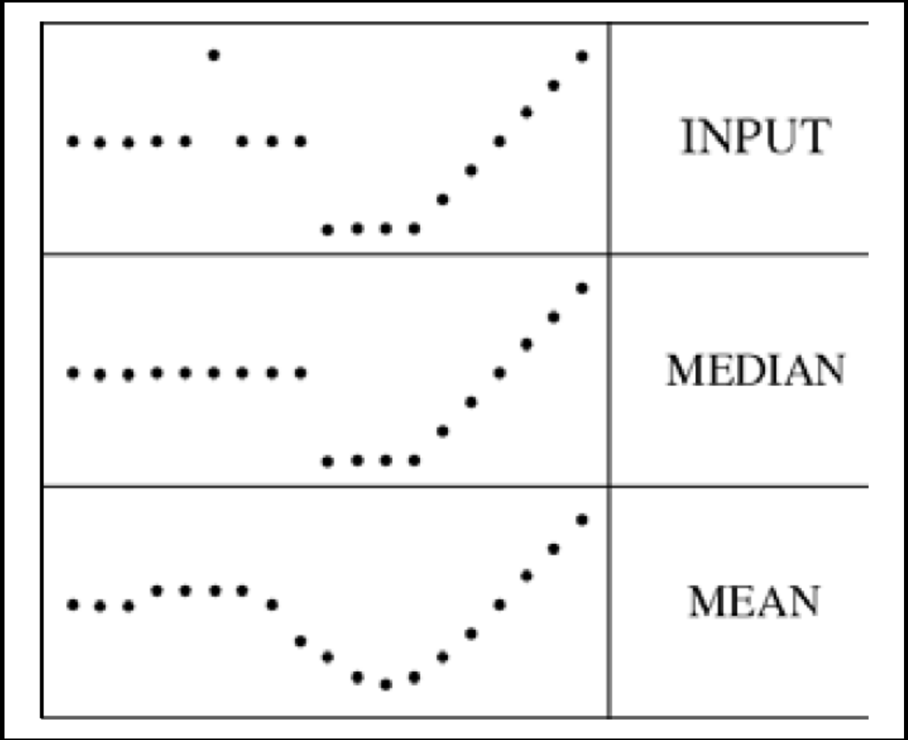


Plots of a row of the image

Source: M. Hebert

Median filter

Median filter is edge preserving



Summary

- In this lesson we learned about correlation filtering and its flipped version called convolution.
- Most of the time if we’re using symmetric filters it won’t matter which is use.
- But if we’re taking derivatives or some other operation that has a specific direction, it will. You’ll get used to it.
- More importantly filtering will be an important tool in your image processing toolbox.