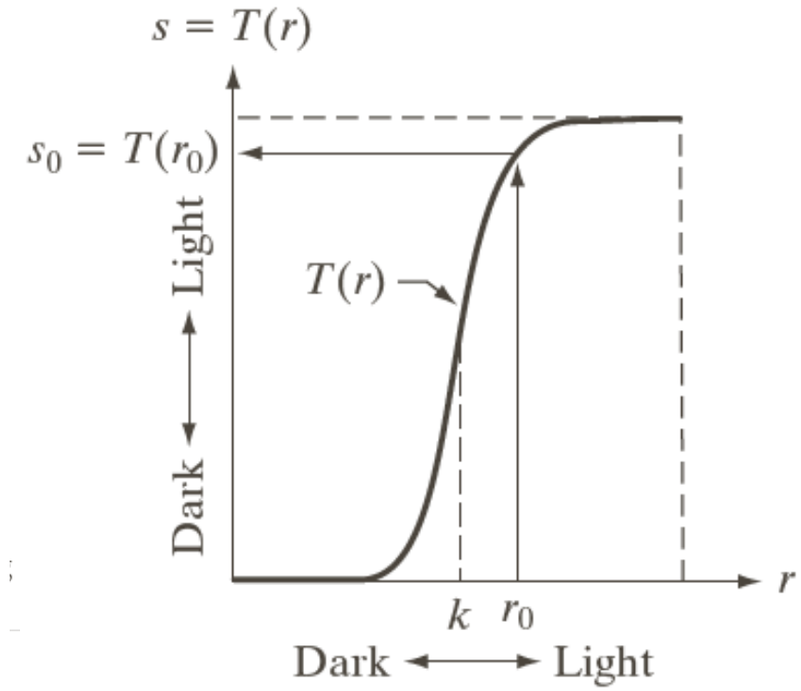
Spatial **Neighborhood** Filtering

for Smoothing and Sharpening

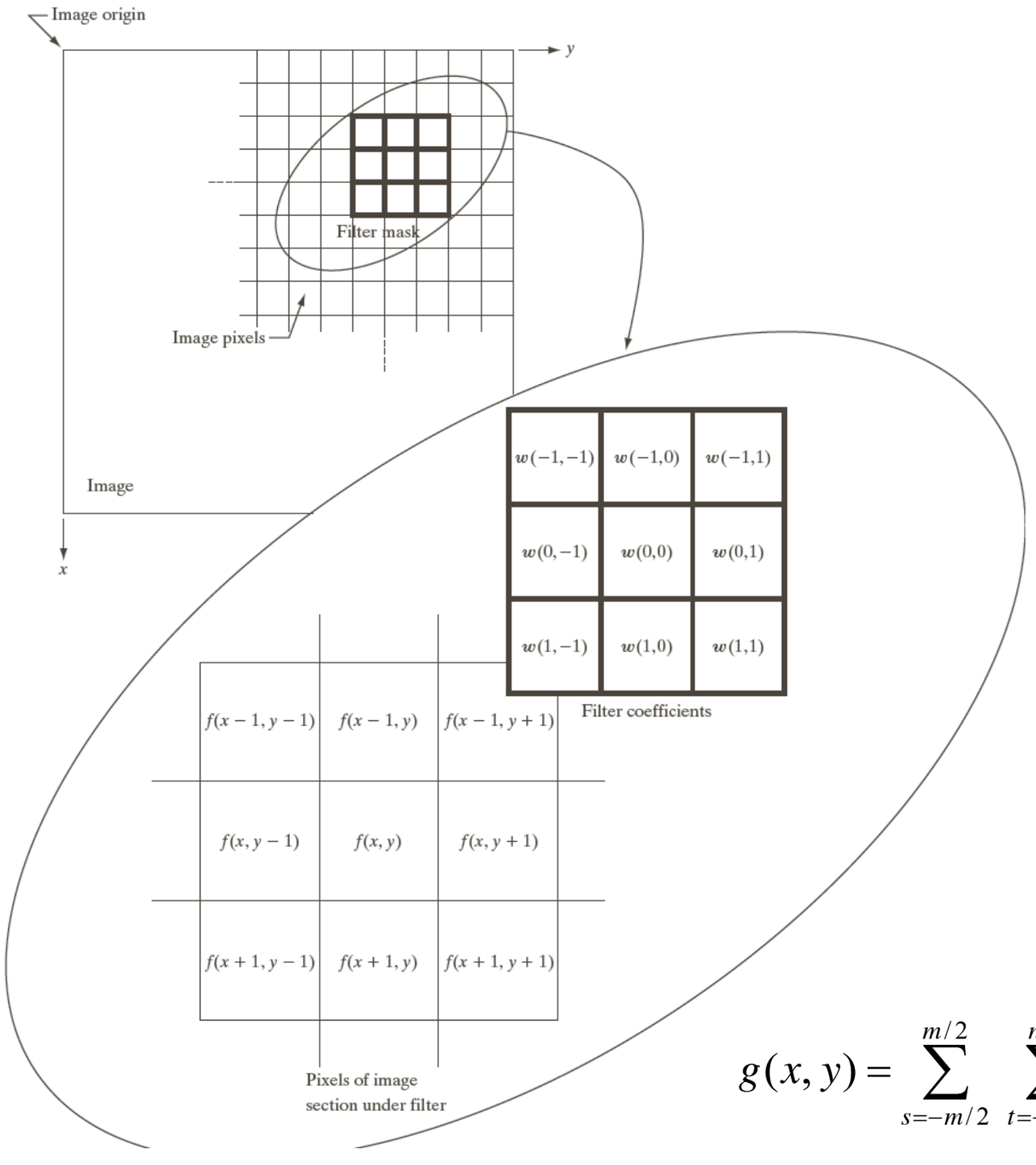
Previously..

- Intensity transformations (point processing)



- Histogram-based methods (histogram equalization)

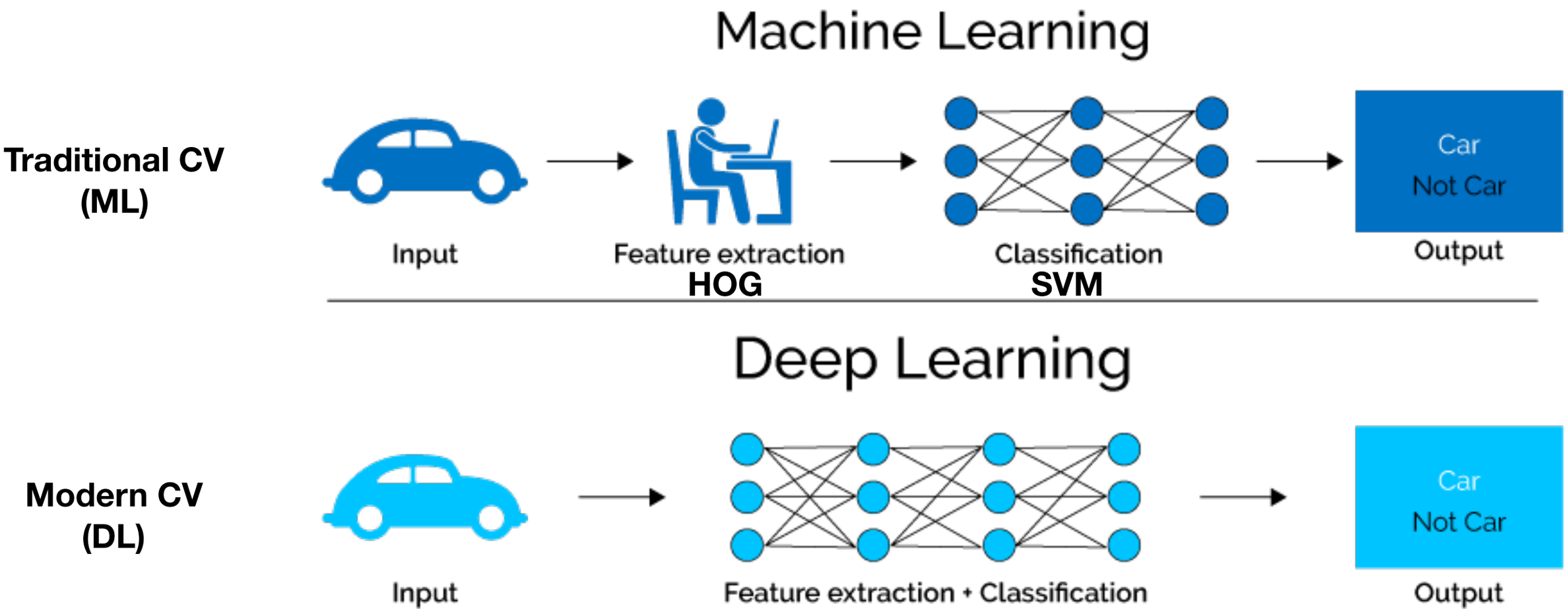
Today..



- Neighborhood processing /filtering:
  - Smoothing
  - Sharpening

$$g(x,y) = \sum_{s=-m/2}^{m/2} \sum_{t=-n/2}^{n/2} w(s,t)\, f(x+s, y+t)$$

Old vs. New approach..



Smoothing

Image Enhancement in the Spatial Domain

Gaussian noise

```
noise =
    randn(size(im)).*sigma;
output = im + noise;
```

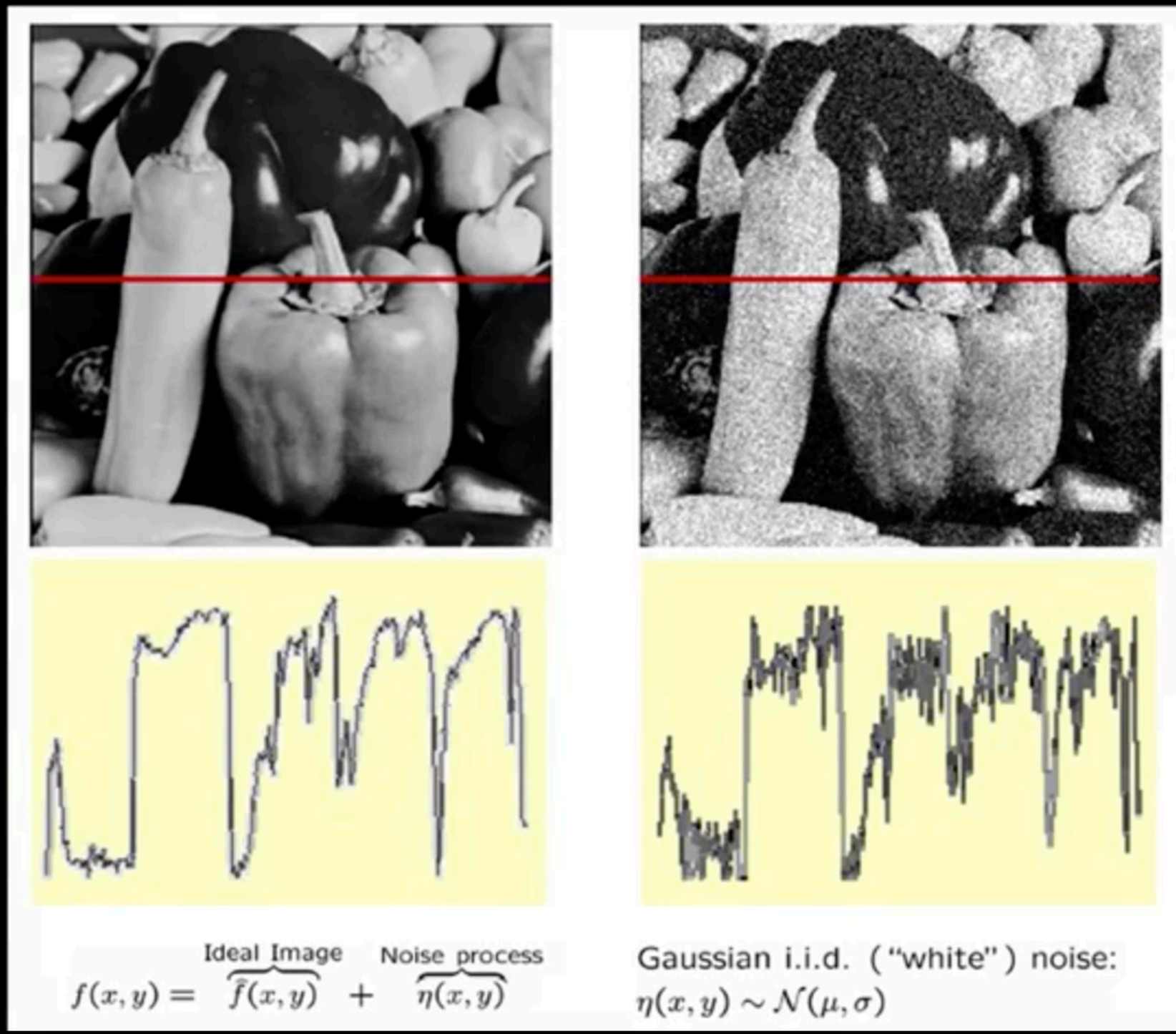
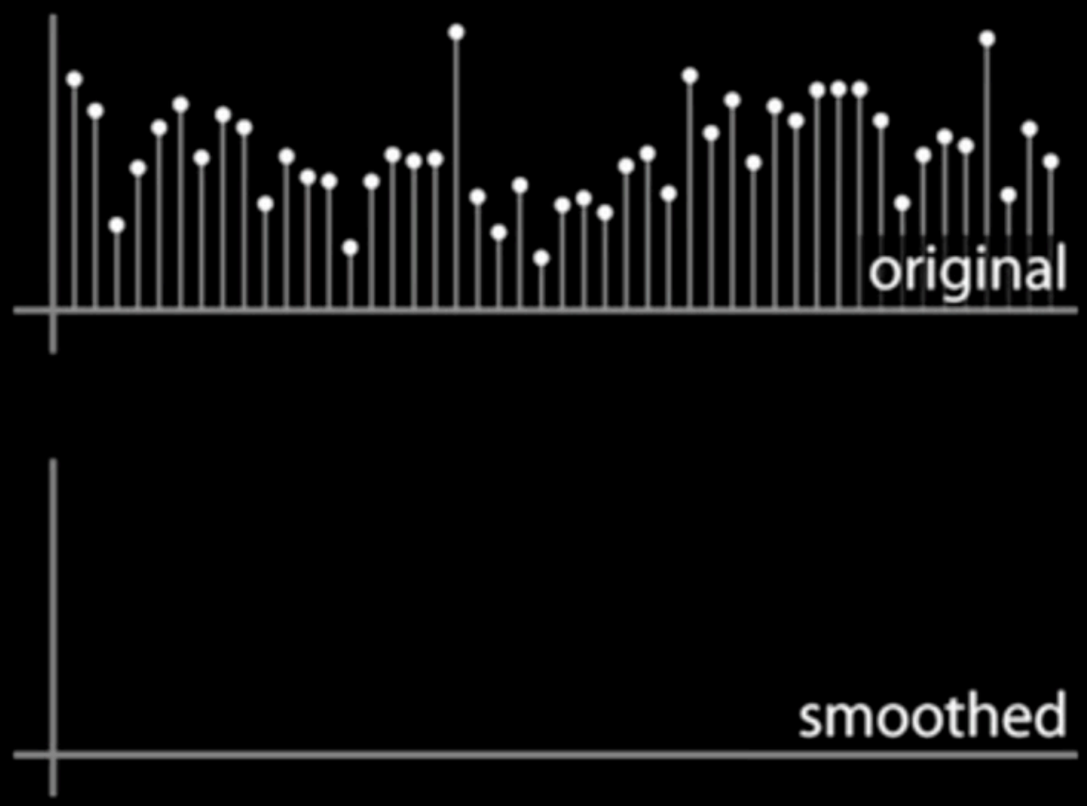


Fig: M. Hebert

First attempt at a solution – 1D

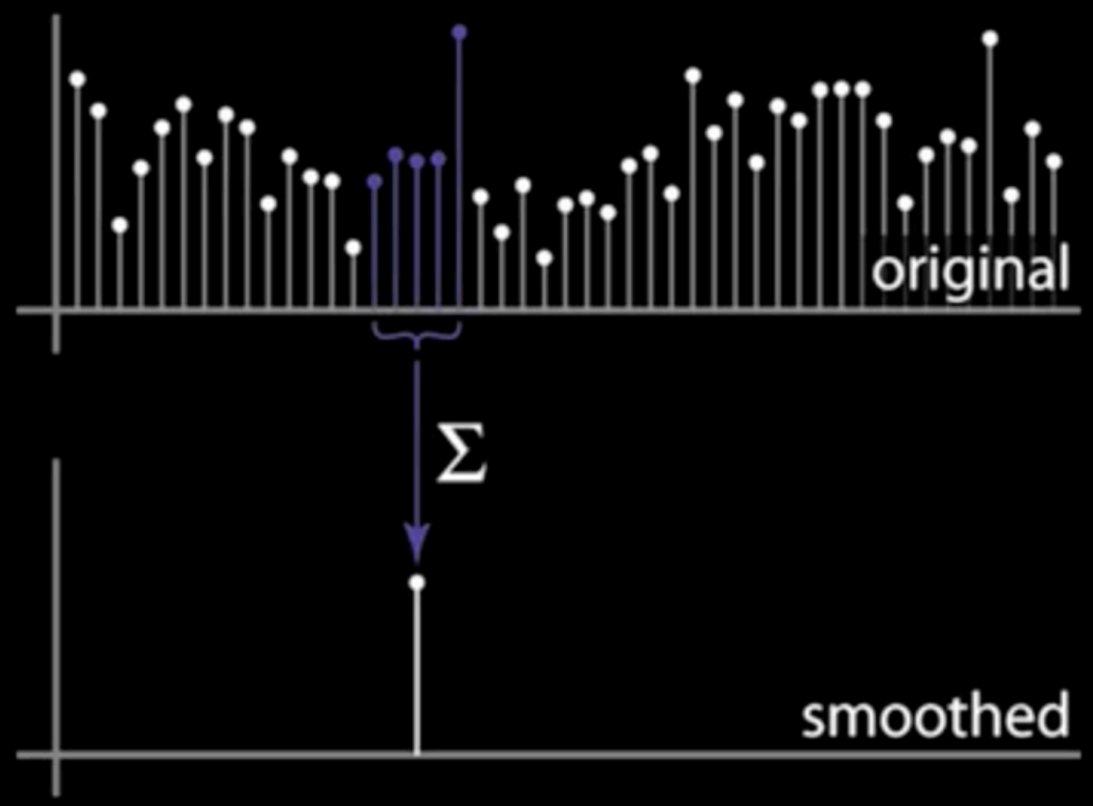
Replace each pixel with an average of all the values in its neighborhood – a *moving average:*



Source: S. Marschner

First attempt at a solution – 1D

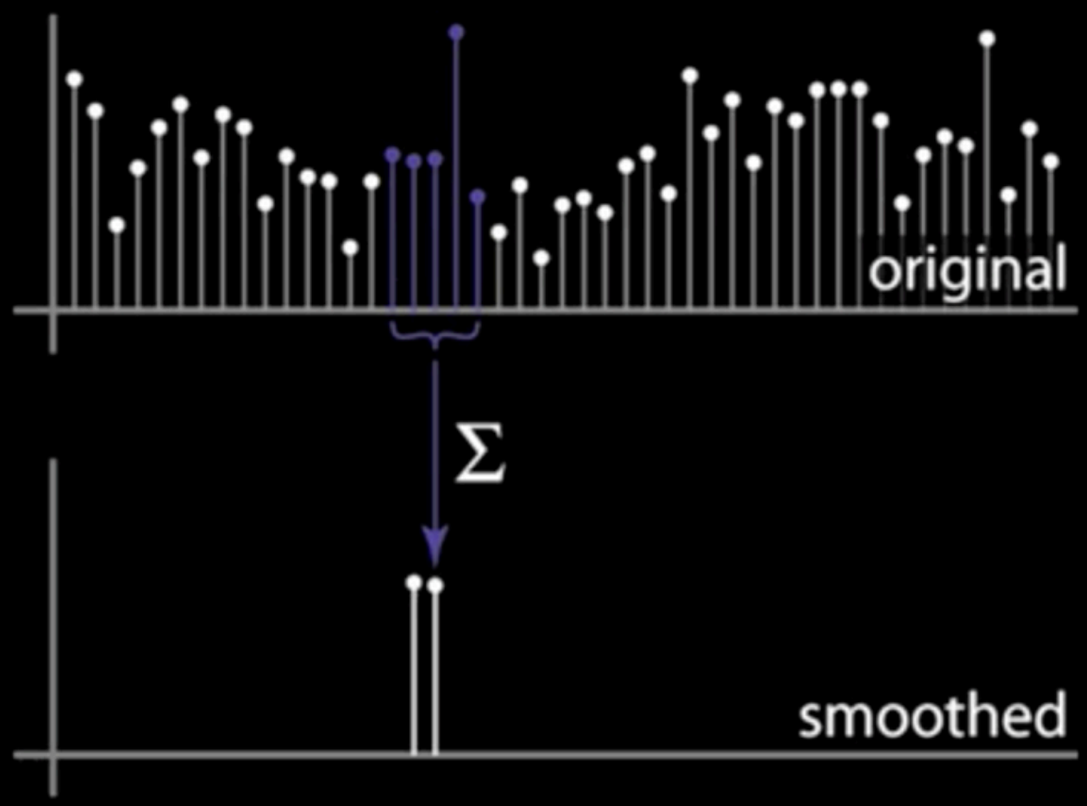
Replace each pixel with an average of all the values in its neighborhood – a *moving average:*



Source: S. Marschner

First attempt at a solution – 1D

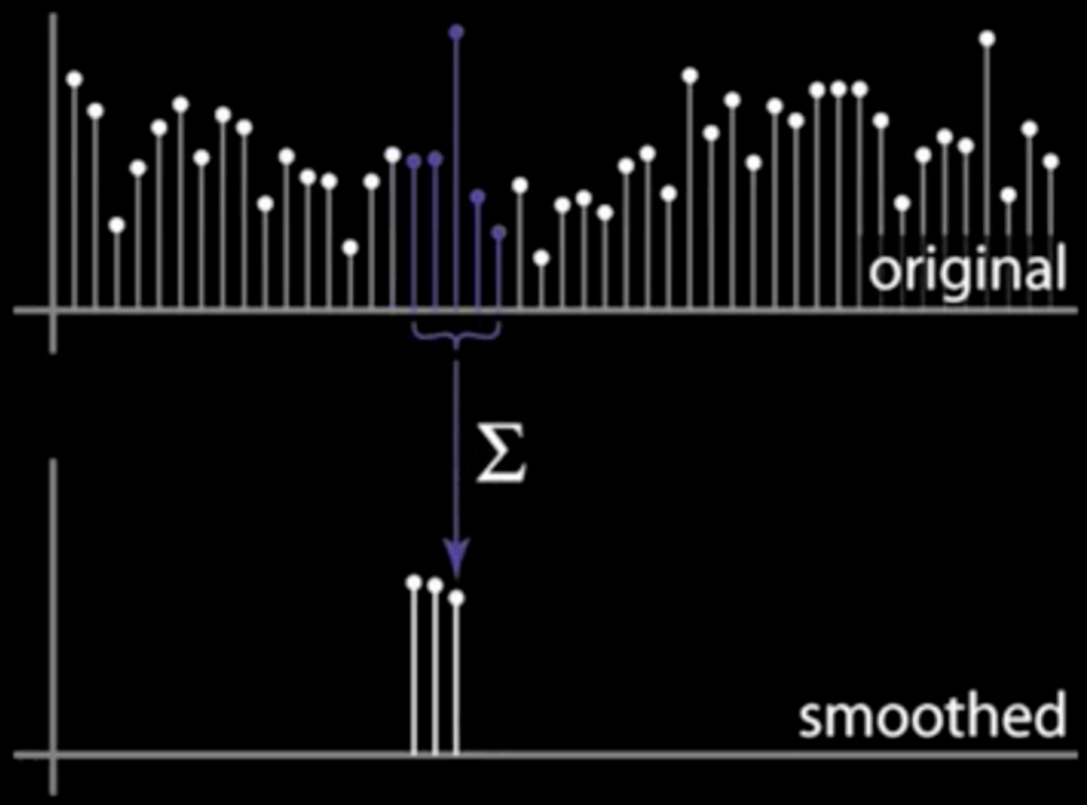
Replace each pixel with an average of all the values in its neighborhood – a *moving average:*



Source: S. Marschner

First attempt at a solution – 1D

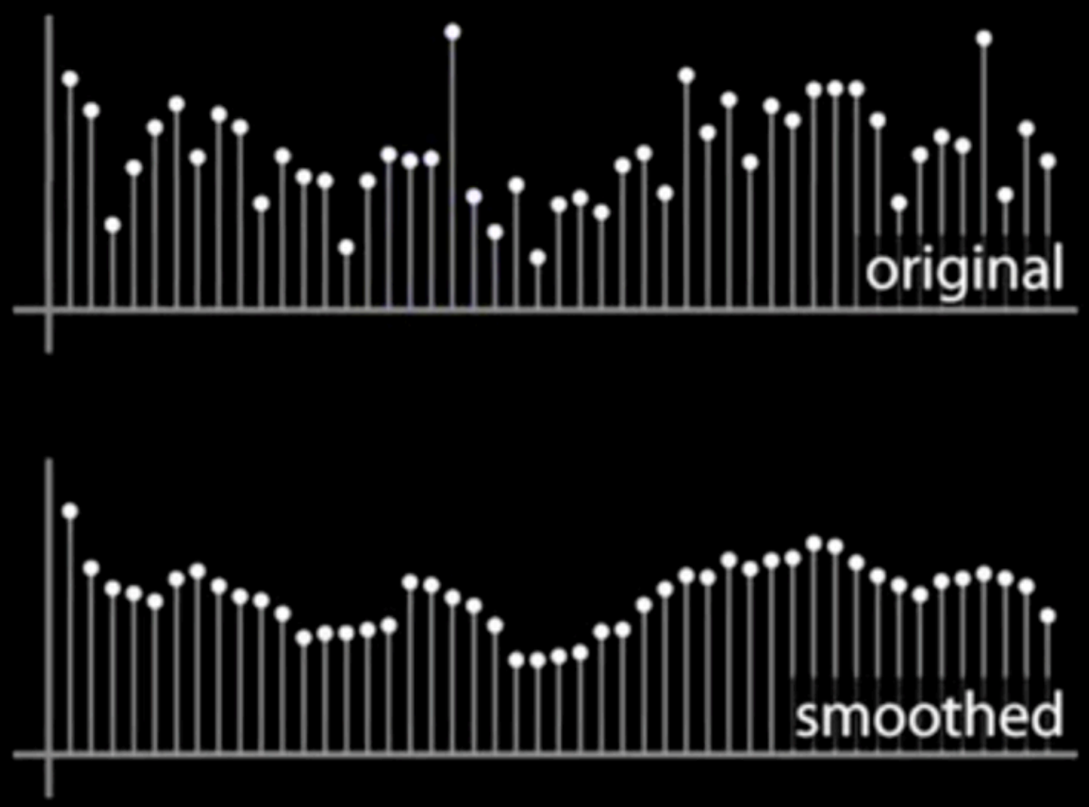
Replace each pixel with an average of all the values in its neighborhood – a *moving average:*



Source: S. Marschner

First attempt at a solution – 1D

Replace each pixel with an average of all the values in its neighborhood – a *moving average:*



Source: S. Marschner

Averaging assumptions

1. The “true” value of pixels are similar to the true value of pixels nearby.

2. The noise added to each pixel is done independently.

Q&A 01

Quiz

If noise is just a function added to an images, we could remove the noise by subtracting the noise again - that is the operation is reversible:

a) True
b) True but we don't know the noise function so we can't actually do the subtraction.
c) False. Additive noise destroys information in the image and there is no way to recover it.

Quiz

If noise is just a function added to an images, we could remove the noise by subtracting the noise again - that is the operation is reversible:

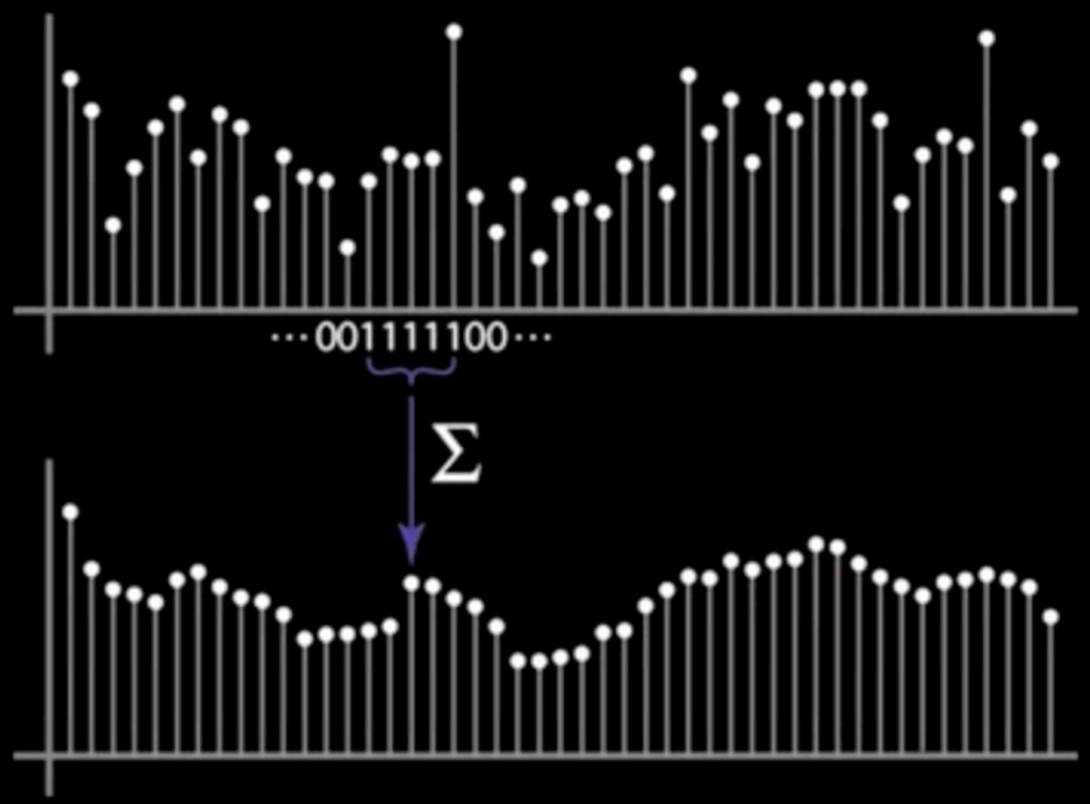
a) True

b) True but we don't know the noise function so we can't actually do the subtraction.

c) False. Additive noise destroys information in the image and there is no way to recover it.

Weighted Moving Average

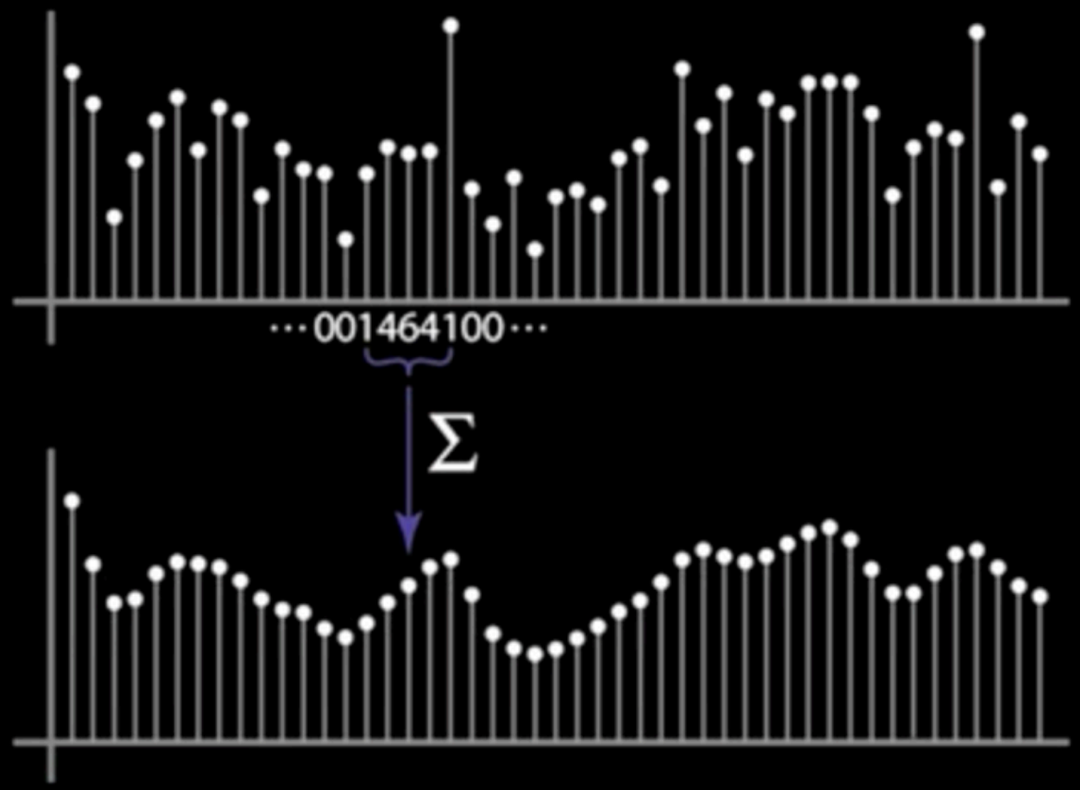
- Can add weights to our moving average
- *Weights* [1, 1, 1, 1, 1] / 5



Source: S. Marschner

Weighted Moving Average

- Non-uniform weights [1, 4, 6, 4, 1] / 16



Source: S. Marschner

Q&A 02

Quiz

To do the moving average computation the number of weights should be

a) Odd – makes it easier to have a middle pixel
b) Even – that way the filter can be exactly symmetric around a pixel.
c) Either even or odd.

Quiz

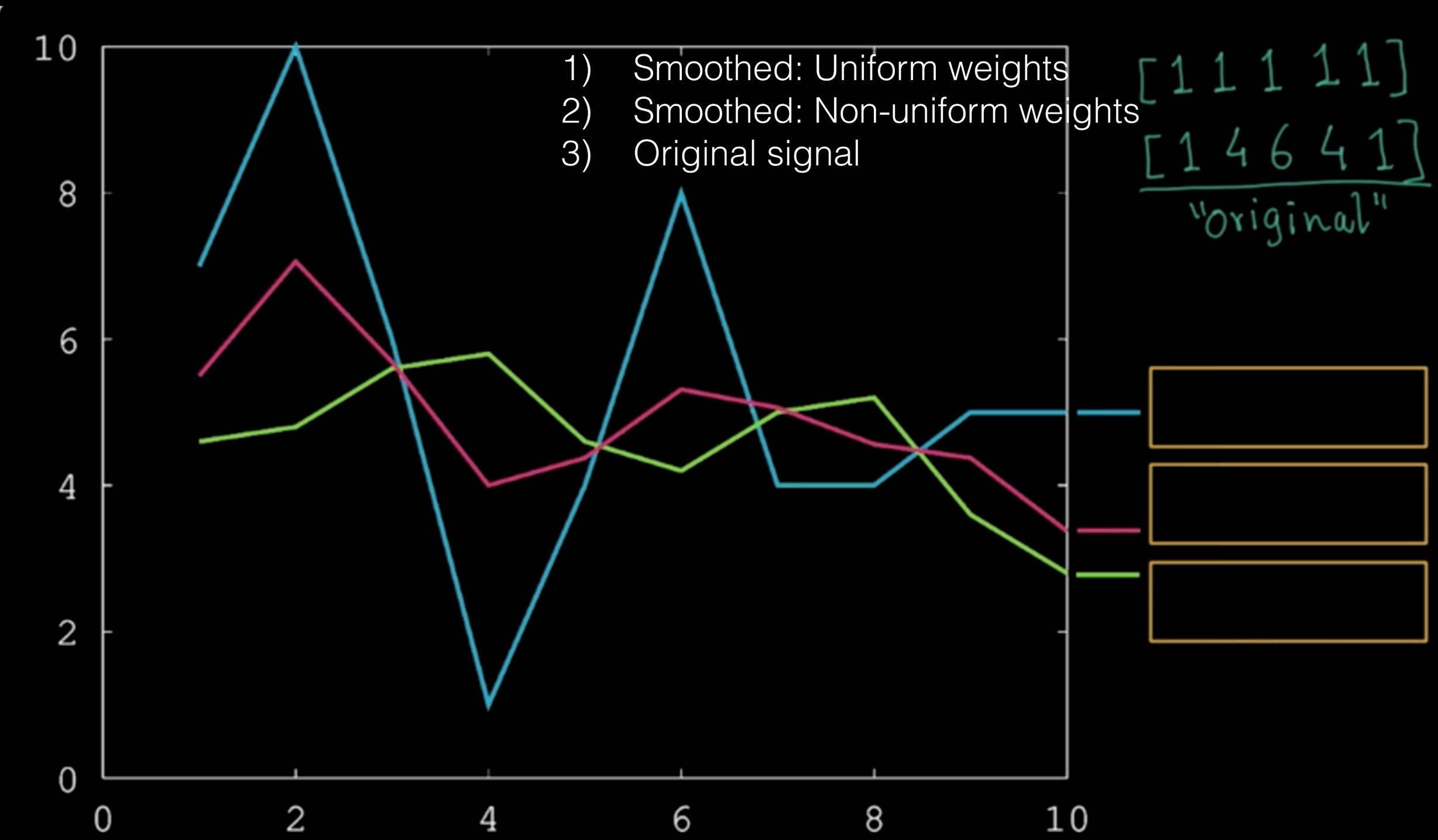
To do the moving average computation the number of weights should be

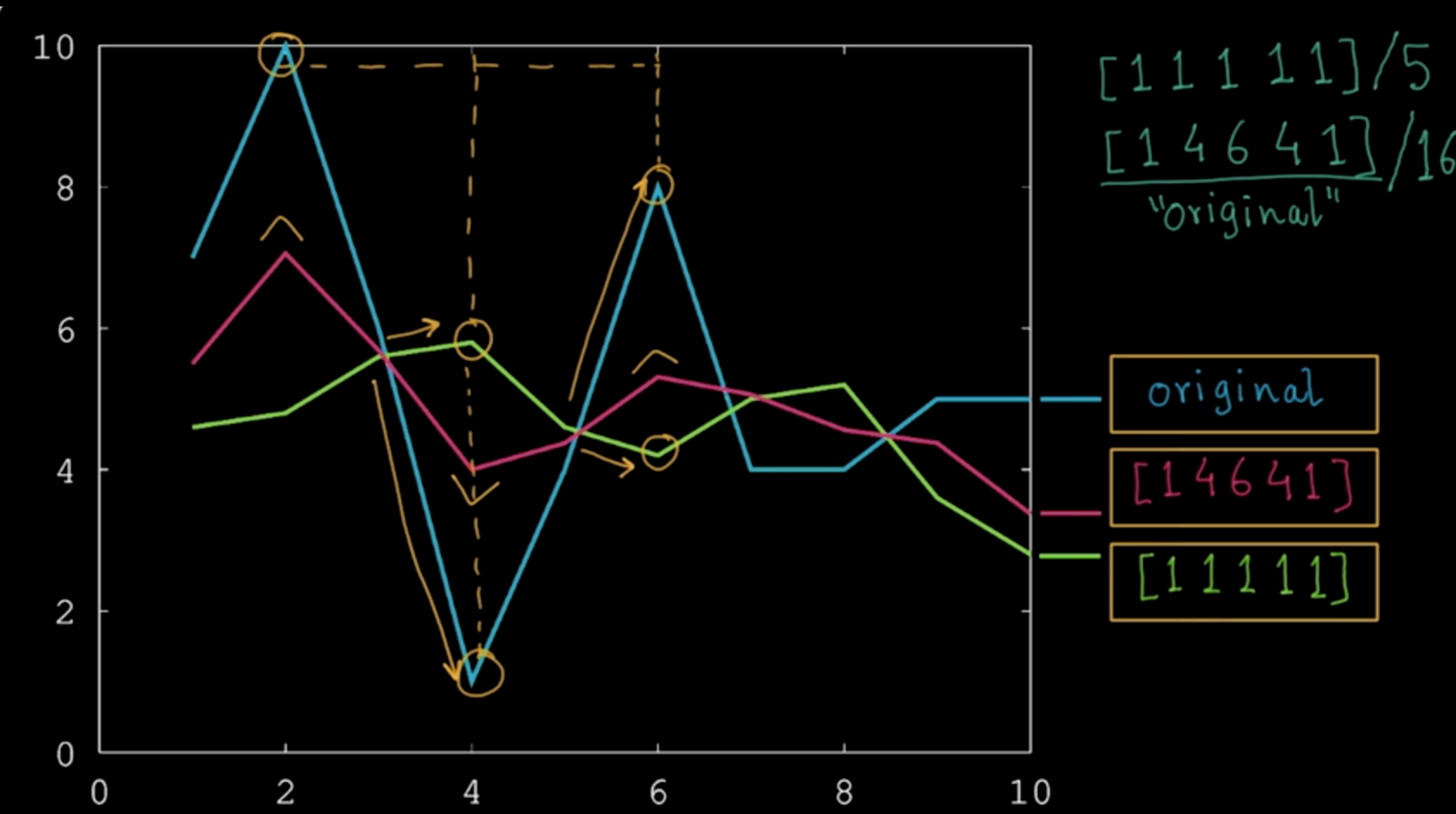
a) Odd – makes it easier to have a middle pixel

b) Even – that way the filter can be exactly symmetric around a pixel.

c) Either even or odd.

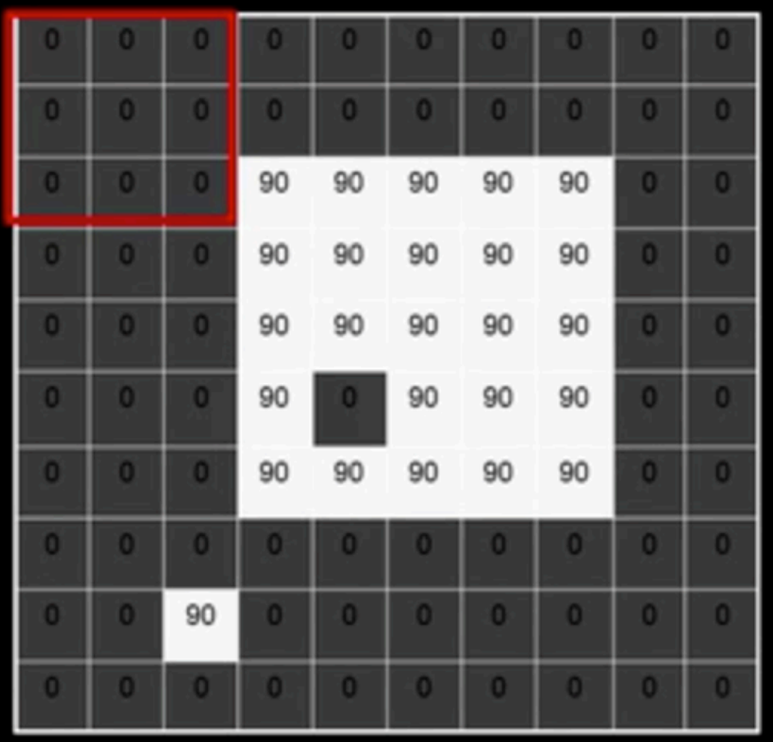
Q&A 03


1) Smoothed: Uniform weights
2) Smoothed: Non-uniform weights
3) Original signal
[1 1 1 1 1]
[1 4 6 4 1]
"original"
10
8
6
4
2
0
0
2
4
6
8
10

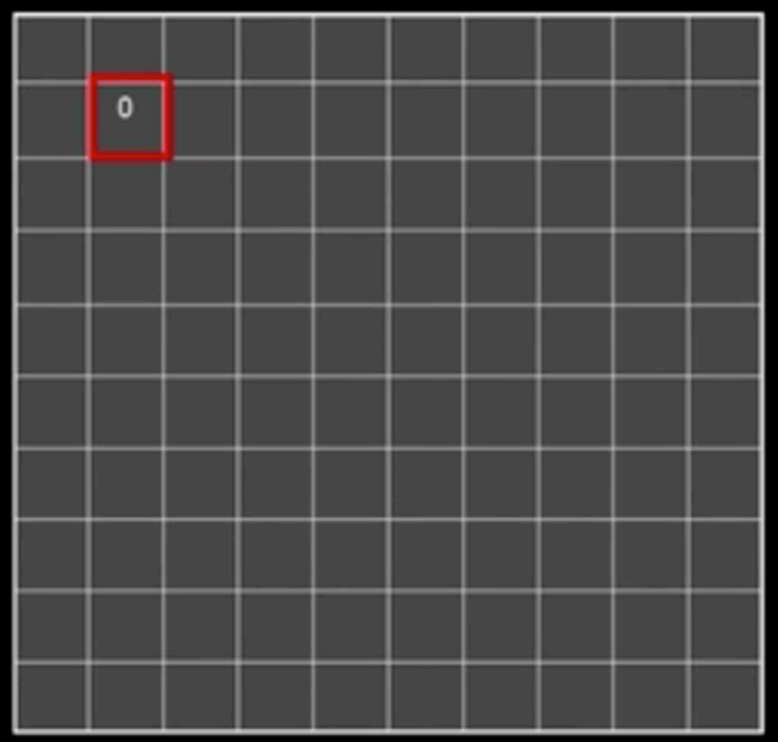

10
8
6
4
2
0
0
2
4
6
8
10
[1 1 1 1 1]/5
[1 4 6 4 1]/16
"original"
original
[14641]
[11111]

Moving Average In 2D

$F(x, y)$



$G(x, y)$

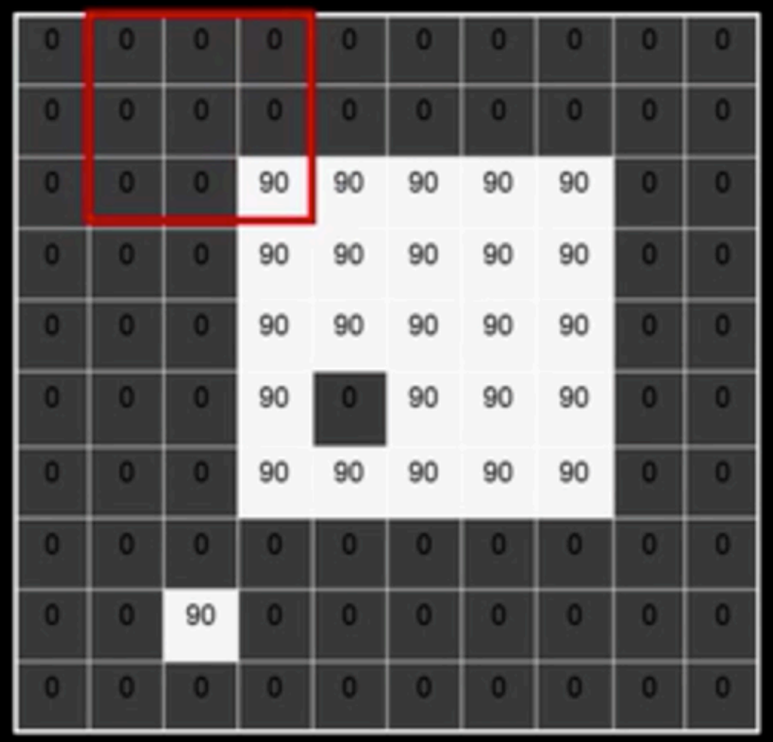


NB: The filter / mask / kernel is a 3x3 matrix with «1 divided by 9» wights

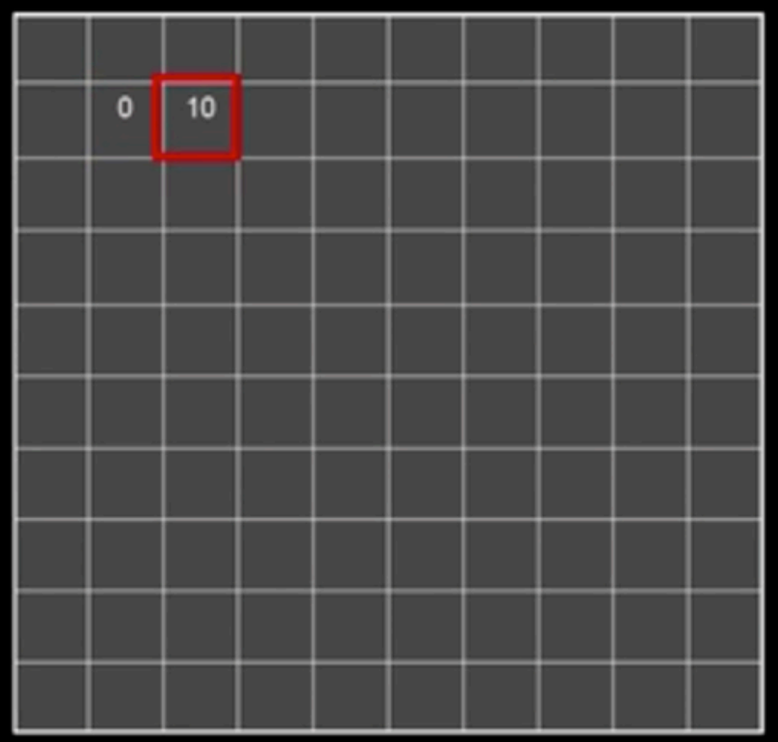
Source: S. Seitz

Moving Average In 2D

$F(x, y)$

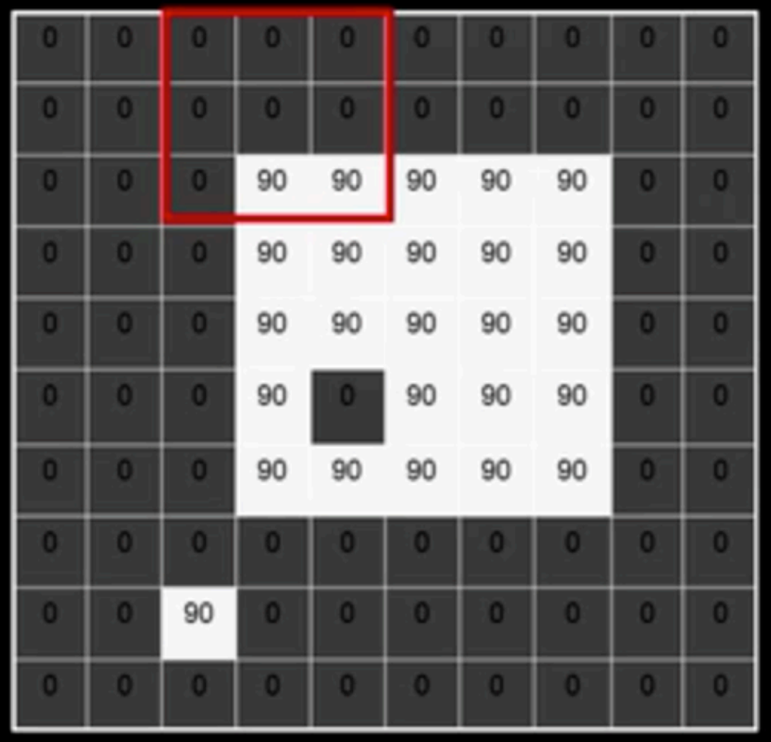


$G(x, y)$

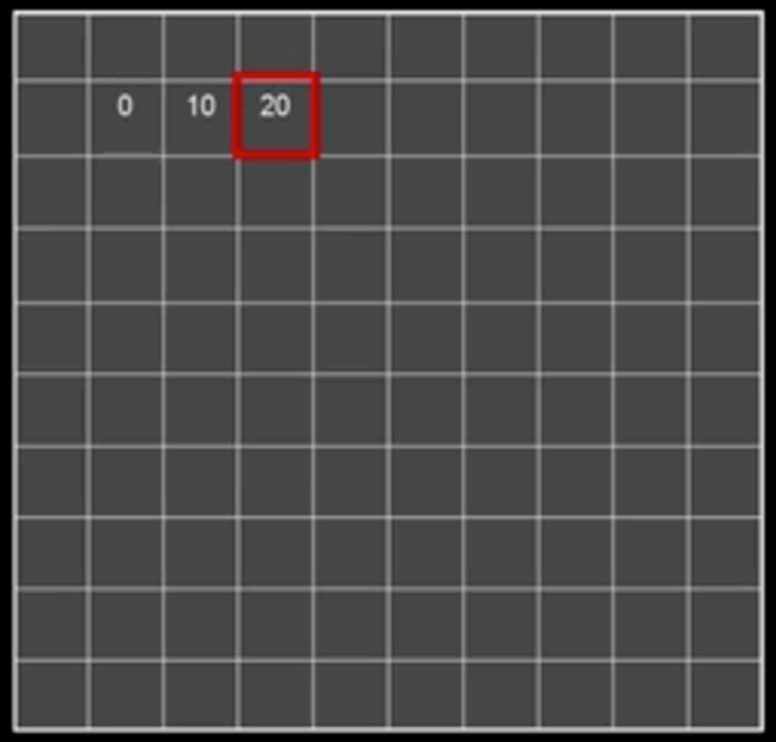


Moving Average In 2D

$F(x, y)$

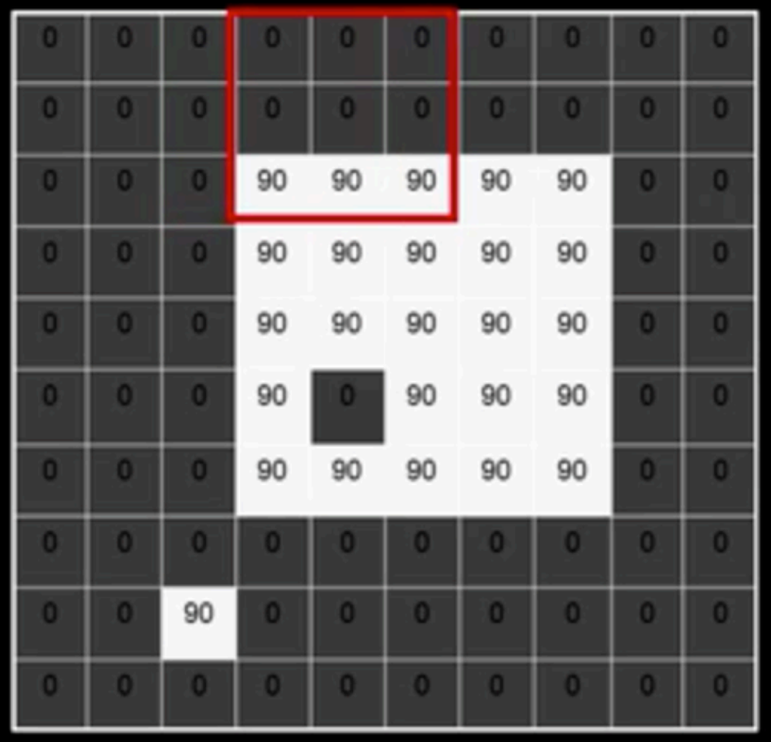


$G(x, y)$

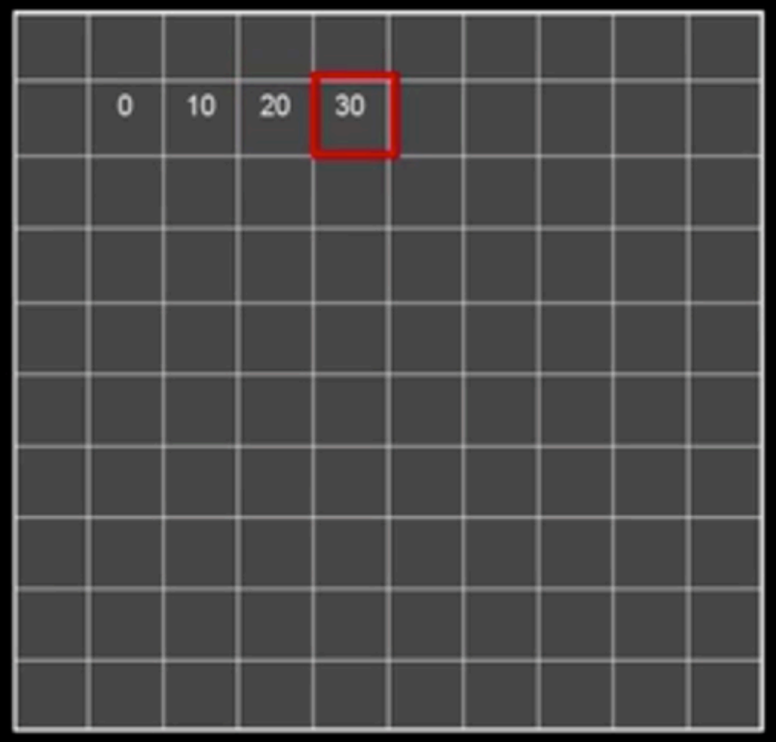


Moving Average In 2D

$F(x, y)$

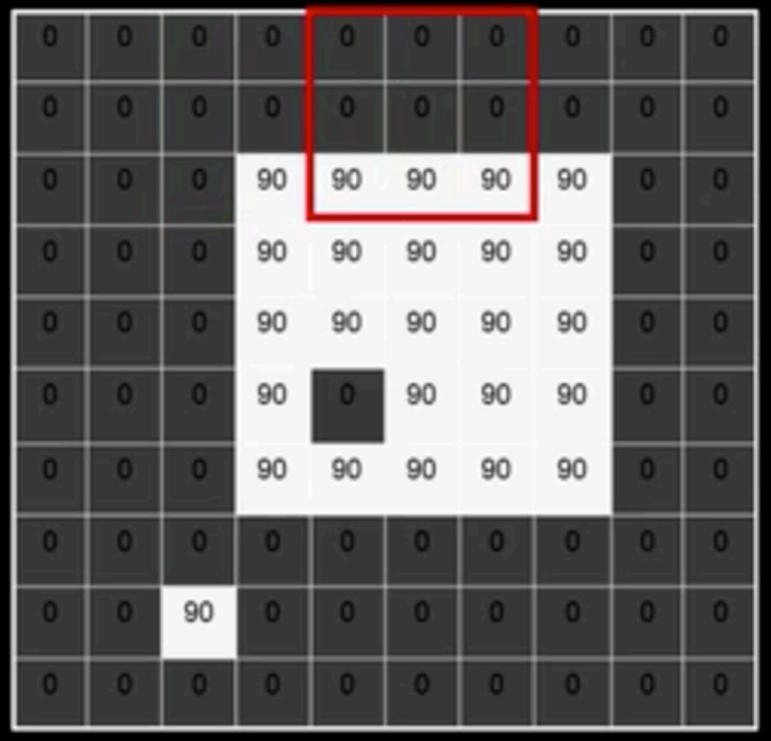


$G(x, y)$

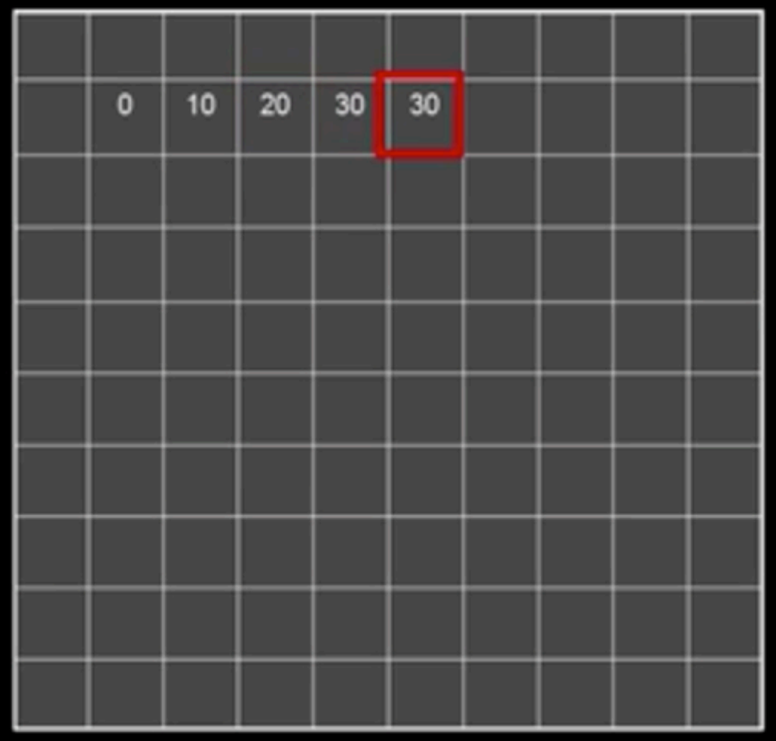


Moving Average In 2D

$F(x, y)$



$G(x, y)$



Q&A 04

Moving Average In 2D

$F(x, y)$

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	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	

Non-uniform weights [1, 4, 6, 4, 1] / 16

Moving Average In 2D

$F(x, y)$

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	

Moving Average In 2D

$F(x, y)$

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	

Correlation filtering – uniform weights

Say the averaging window size is 2k+1 x 2k+1:

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

Correlation filtering – uniform weights

Say the averaging window size is 2k+1 x 2k+1:

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

Uniform weight for each pixel

Loop over all pixels in neighborhood around image pixel F[i,j]

Correlation filtering – nonuniform weights

Now generalize to allow **different weights** depending on neighboring pixel's relative position:

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

Non-uniform weights

This is called ***cross-correlation***, denoted $G = H \otimes F$

k=1

Indices for H:

-1,-1	–1,0	–1,1
1,-1		1,1

Indices for F:

i-1, j-1	i-1,j	i-1, j+1		
	i,j			
i+1, j-1		i+1,j+1		

Averaging filter

$$F(x,y) \otimes H(u,v) = G(x,y)$$

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

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

"box filter"

	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	

$$G = F \otimes H$$

Smoothing by box averaging

original

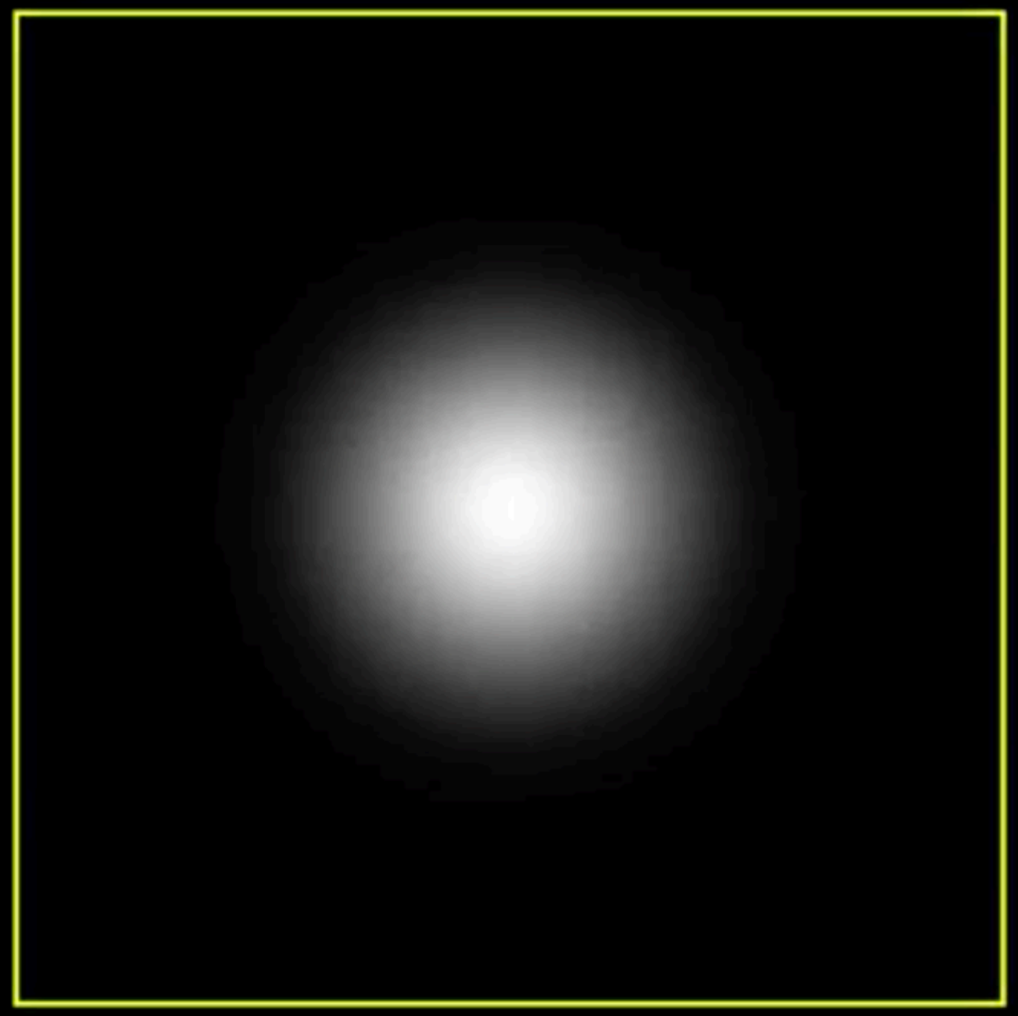
?

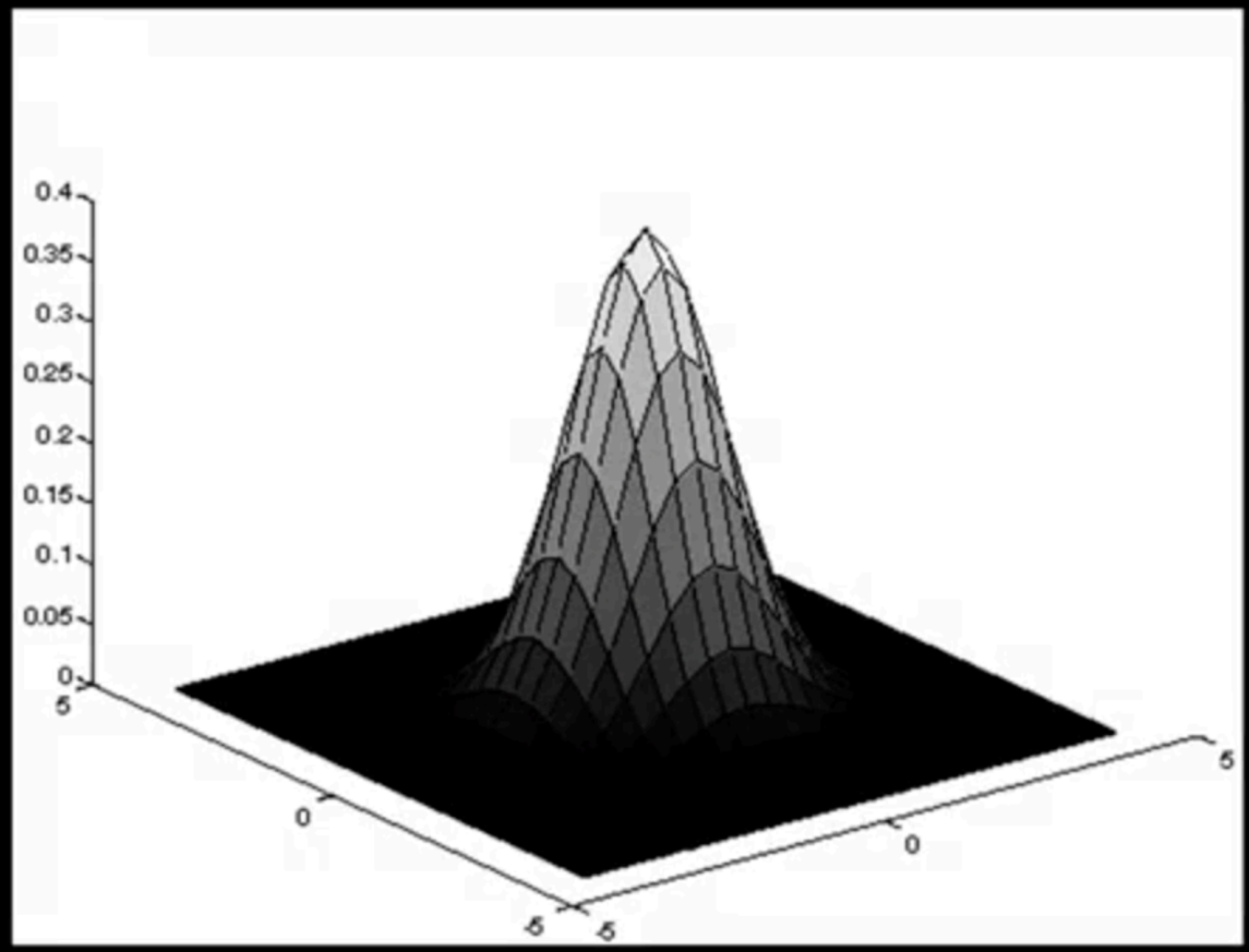
Smoothing by box averaging

original

filtered

Blurry spot as a function





D. Forsyth

Q&A 05

Quiz

To blur a single pixel into a "blurry" spot, we would need to filter the spot with a

a) 3x3 square of uniform weights

b) 11x11 square of uniform weights would be better since it's bigger

c) Something that looks like a blurry spot – higher values in the middle, falling off to the edges.

Quiz

To blur a single pixel into a "blurry" spot, we would need to filter the spot with a

a) 3x3 square of uniform weights

b) 11x11 square of uniform weights would be better since it's bigger

c) Something that looks like a blurry spot – higher values in the middle, falling off to the edges.

Gaussian filter

- Nearest neighboring pixels have the most influence

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

$F(x, y)$

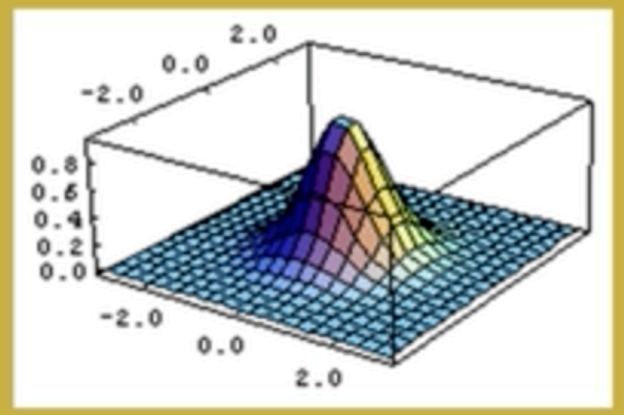
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H(u, v)$

This kernel is an approximation of a Gaussian function:

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

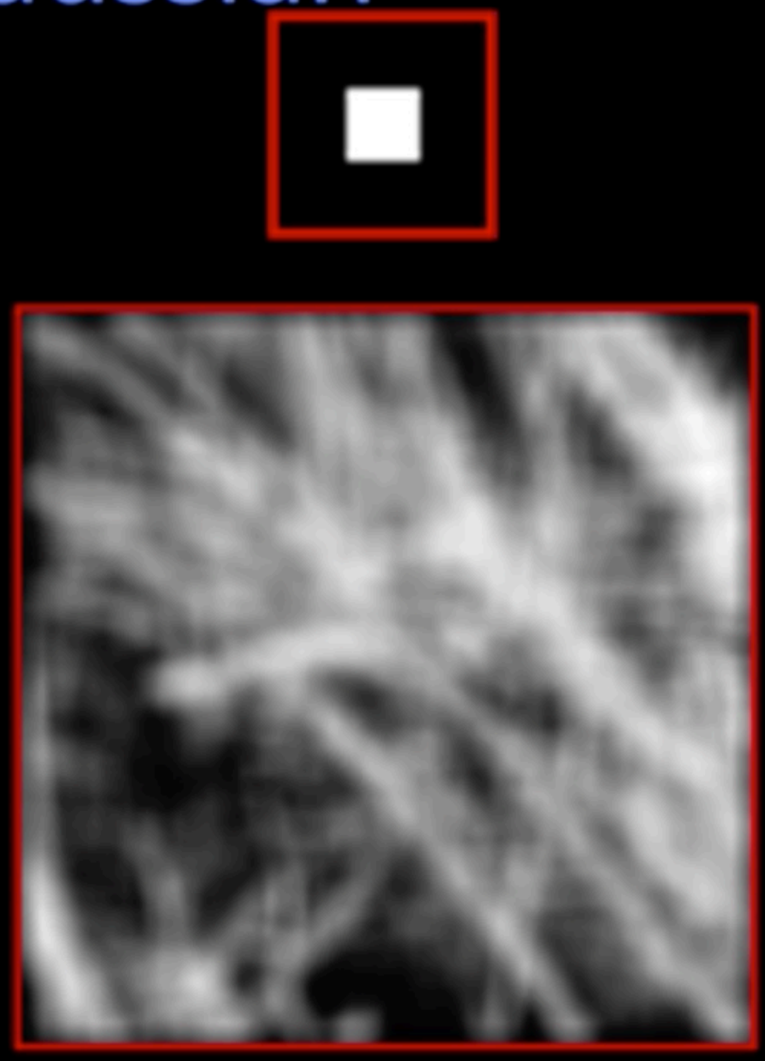


Source: S. Seitz

Smoothing with a Gaussian

Smoothing with a Gaussian

Smoothing with not a Gaussian



Q&A 06

Normalization constant

Quiz

Gaussian filters are referred to as exponentials. The complete formula is:

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

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

c) Or even: $$h(u,v) = \frac{1}{\sqrt{2\pi\sigma}}\exp(-\frac{x^2+y^2}{2\sigma^2})$$

Quiz

Gaussian filters are referred to as exponentials. The complete formula is:

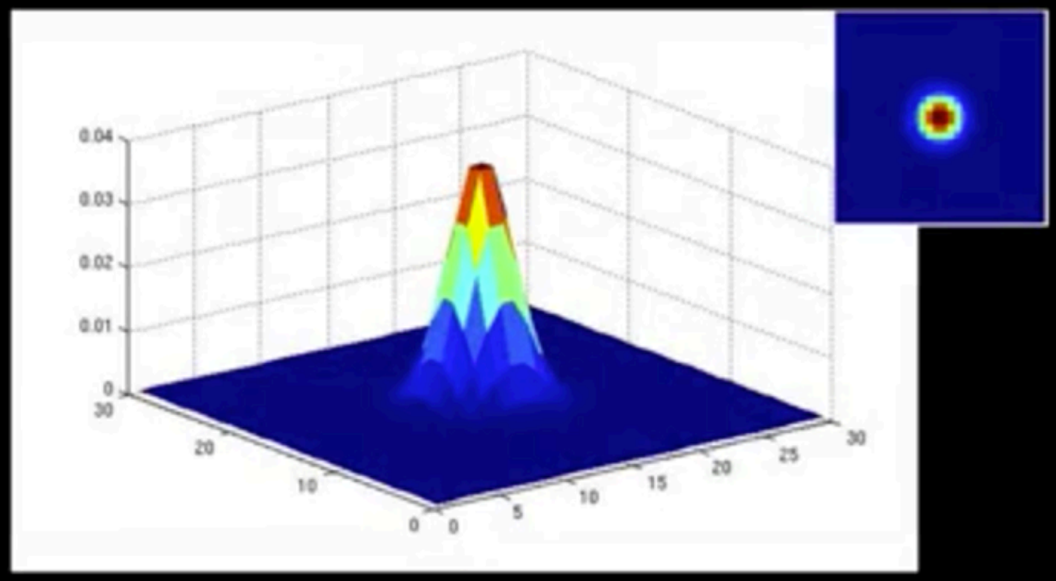
a) is $$h(u,v) = \frac{1}{2\pi\sigma^2}\exp(-\frac{x^2+y^2}{2\sigma^2})$$

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

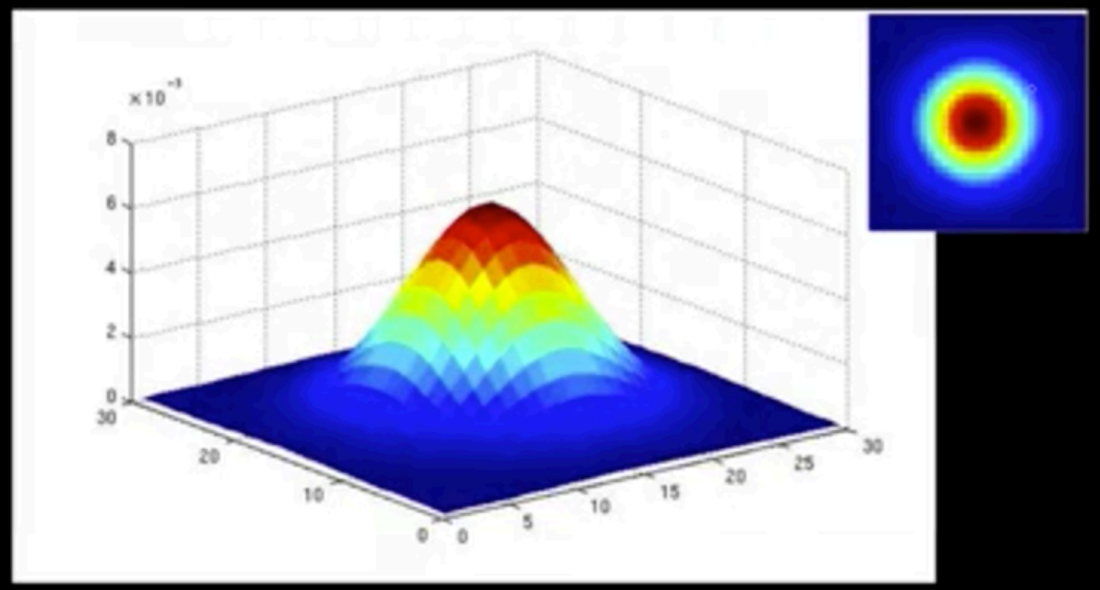
c) Or even: $$h(u,v) = \frac{1}{\sqrt{2\pi\sigma}}\exp(-\frac{x^2+y^2}{2\sigma^2})$$

Gaussian filters

Variance (σ^2) or **standard deviation** (σ) – determines extent of smoothing



σ = 2 with 30 x 30 kernel

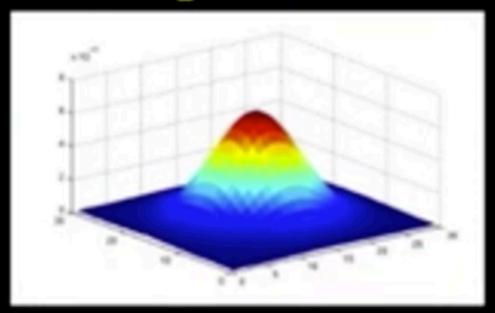


σ = 5 with 30 x 30 kernel

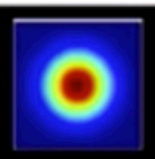
K. Grauman

Matlab

```
>> hsize = 31;
>> sigma = 5;
>> h = fspecial('gaussian', hsize, sigma);
```



```
>> surf(h);
```



```
>> imagesc(h);
```

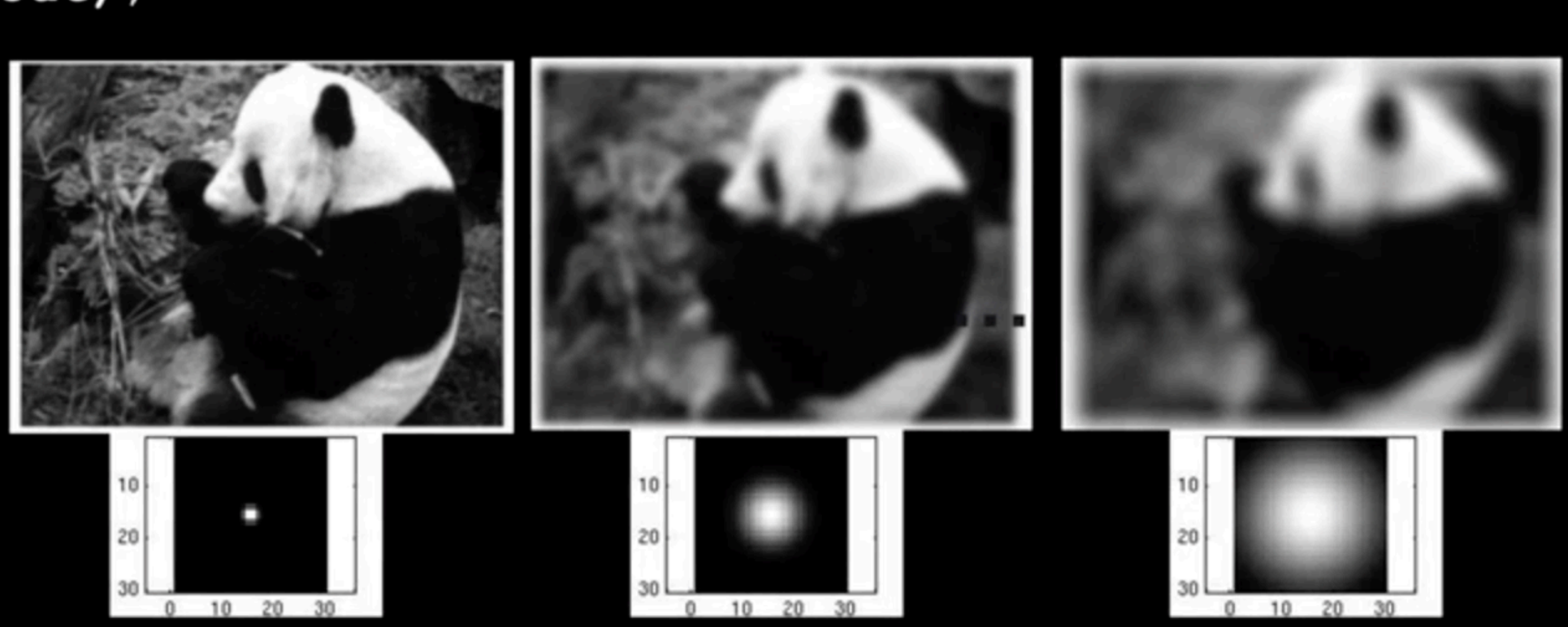
im

```
>> outim = imfilter(im, h);
>> imshow(outim);
```

K. Grauman

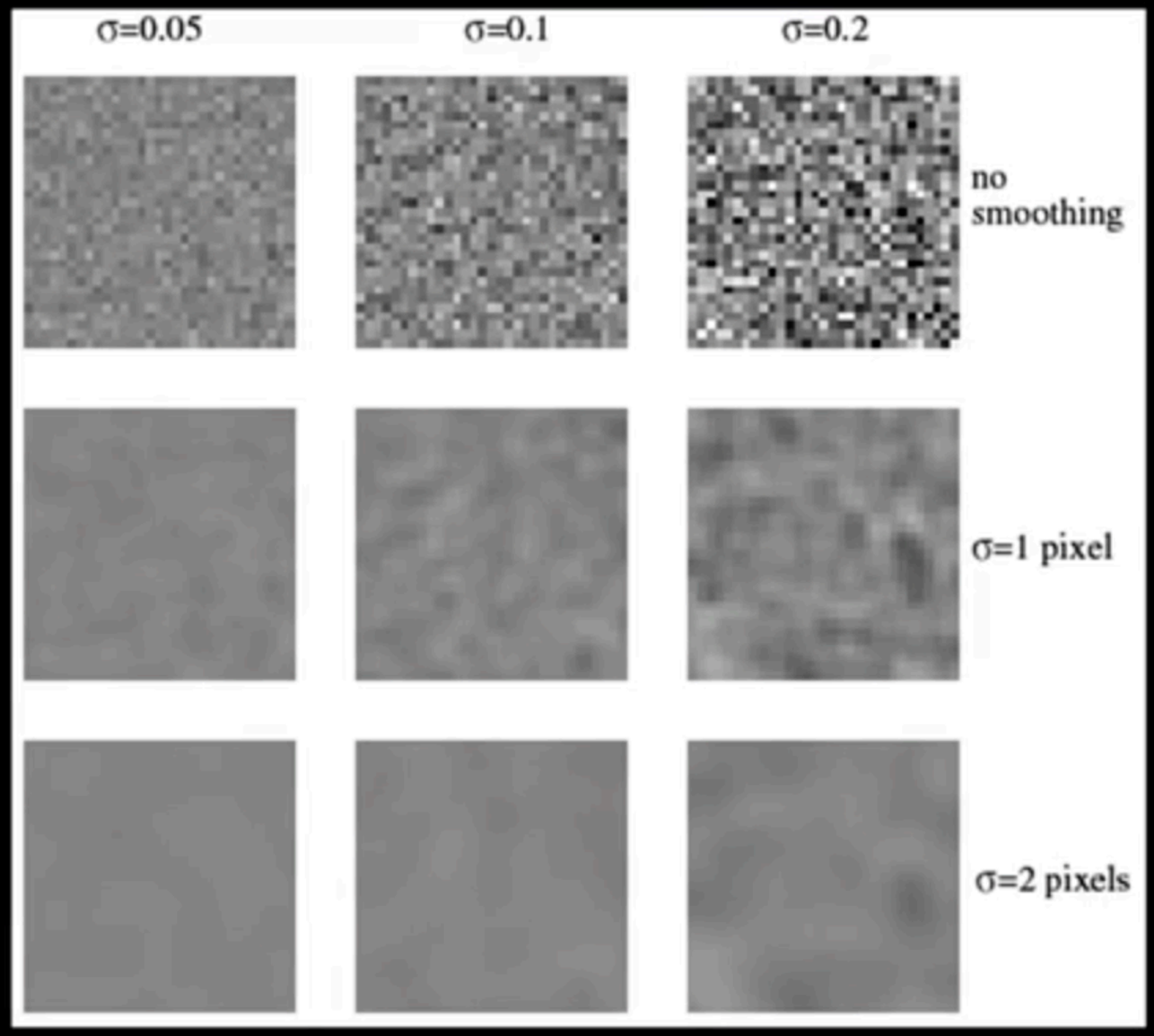
Smoothing with a Gaussian

```
for sigma=1:3:10
 h = fspecial('gaussian',
 fsize, sigma);
 out = imfilter(im, h);
 imshow(out);
 pause;
end
```



Keeping the two Gaussians straight…

More Gaussian noise (like earlier) σ →



Wider Gaussian smoothing kernel σ→

Q&A 07

Gaussian:
effect of sigma, size and normalization coefficient

Quiz

When filtering with a Gaussian, which is true:

a) The sigma is most important – it defines the blur kernel's scale with respect to the image.

b) The kernal size is most important – because it defines the scale.

c) Altering the normalization coefficient does not effect the blur, only the brightness.

d) A and C.

Quiz

When filtering with a Gaussian, which is true:

a) The sigma is most important – it defines the blur kernel's scale with respect to the image.

b) The kernal size is most important – because it defines the scale.

c) Altering the normalization coefficient does not effect the blur, only the brightness.

d) A and C.

Linearity and convolution

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):
  - $H(f1 + f2) = H(f1) + H(f2)$ (looks like distributive law)
- Multiplicative scaling (Homogeneity of degree 1) (constant scales):
  - $H(a \cdot f1) = a \cdot H(f1)$

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):
  - $H(f1 + f2) = H(f1) + H(f2)$ (looks like distributive law)
- Multiplicative scaling (Homogeneity of degree 1) (constant scales):
  - $H(a \cdot f1) = a \cdot H(f1)$

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

Q&A 08

Quiz

Which of these operators are not linear:

a) Sum

b) Max

c) Average

d) Square root

e) (b) and (d)

Quiz

Which of these operators are not linear:

a) Sum

b) Max

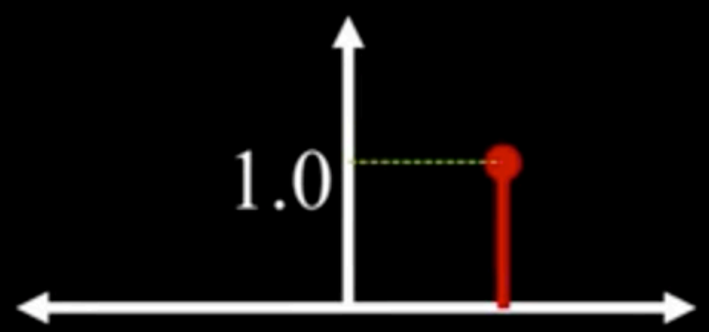
c) Average

d) Square root

e) (b) and (d)

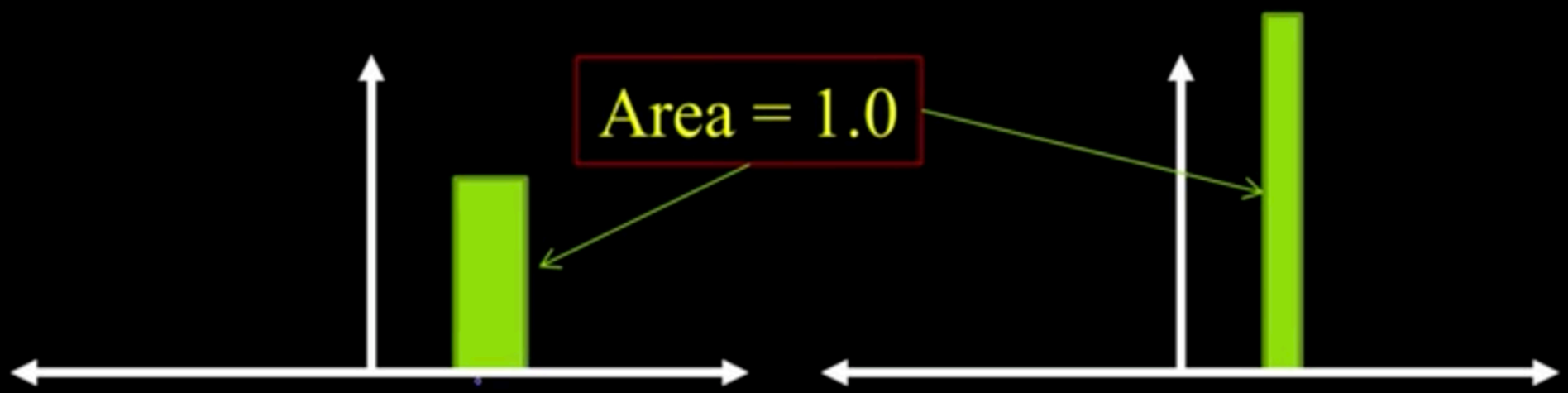
An impulse function…

- 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:



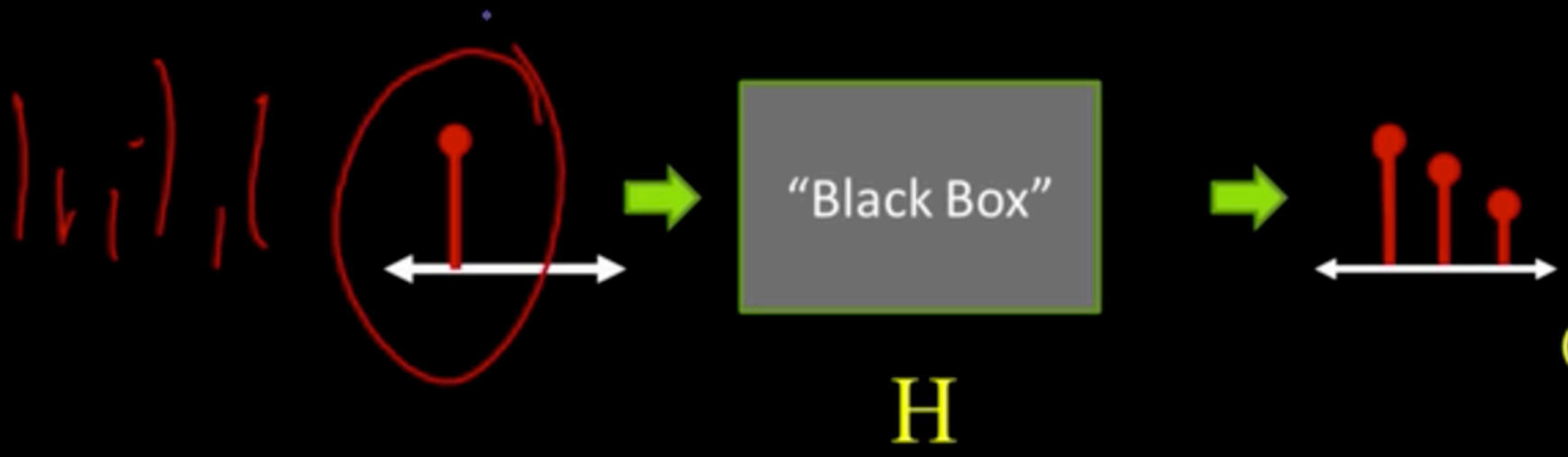
An impulse response

- 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?

An impulse response

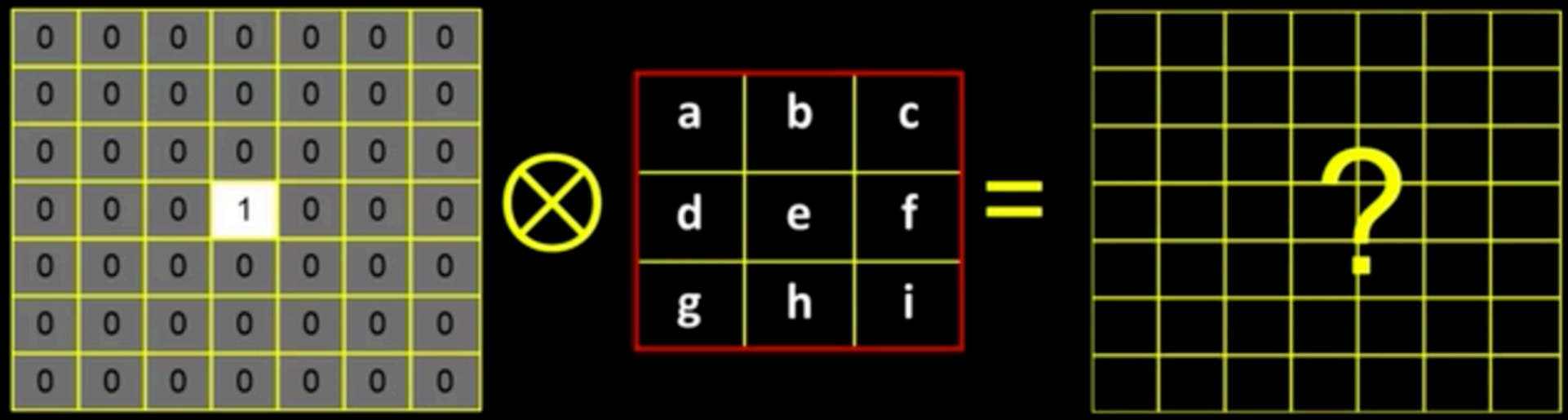
- If I have an unknown system and I “put in” an impulse, the response is called the impulse response. (Duh?)



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

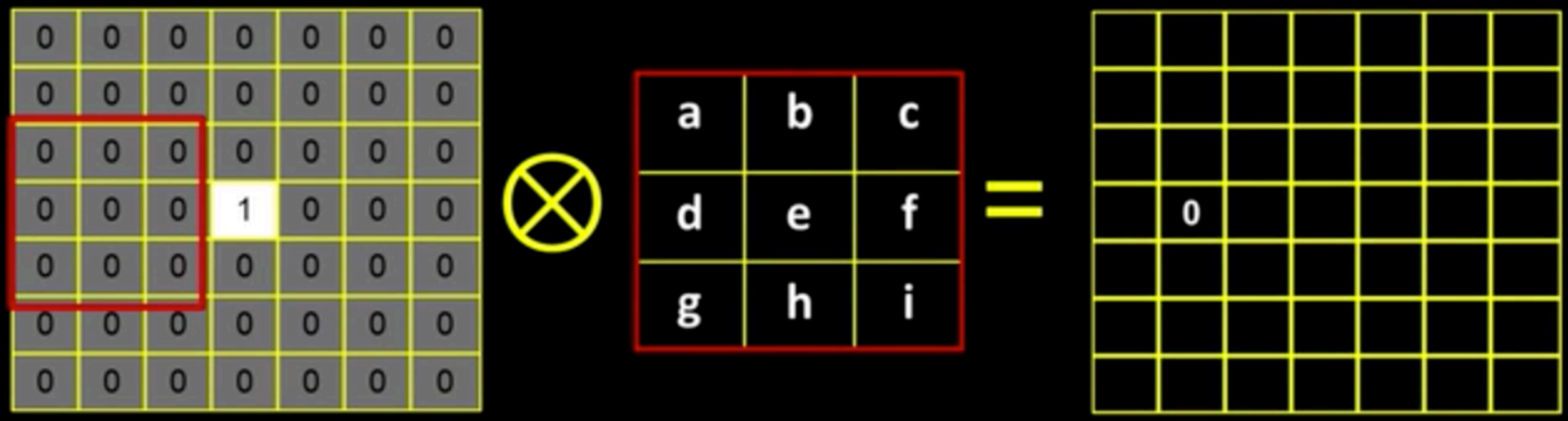
Filtering an impulse signal

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



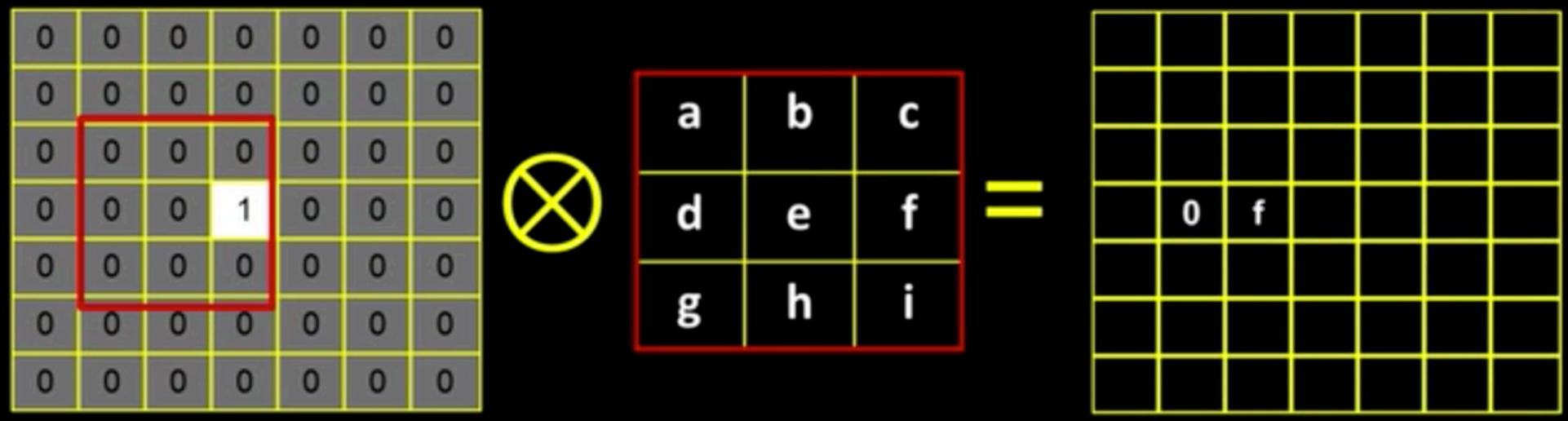
F(x,y) ***H(u,v)*** ***G(x,y)***

Filtering an impulse signal



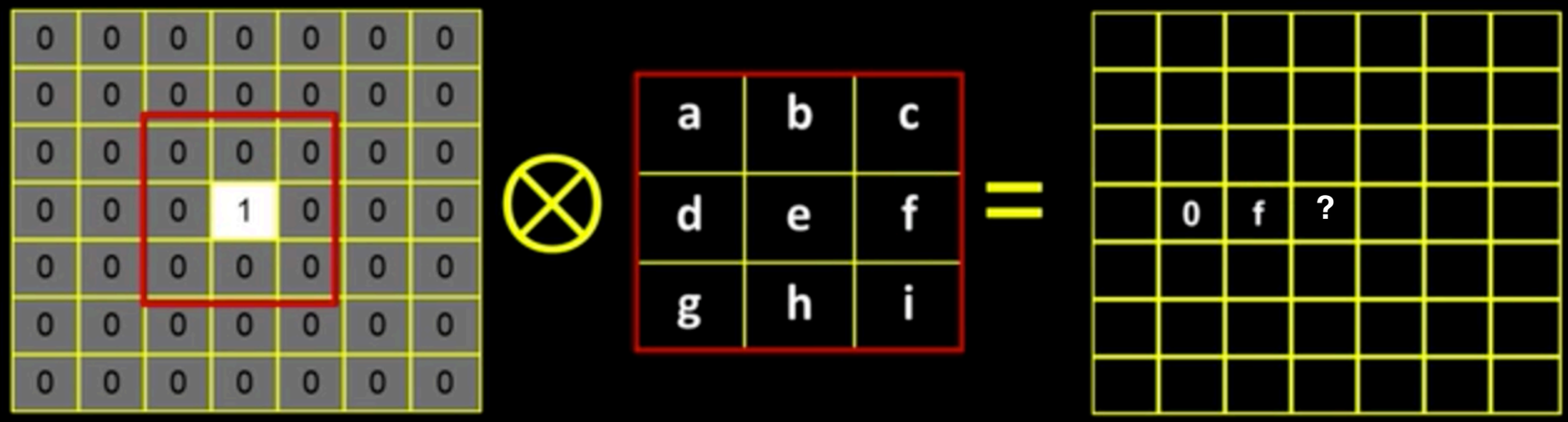
F(x,y) *H(u,v)* *G(x,y)*

Filtering an impulse signal



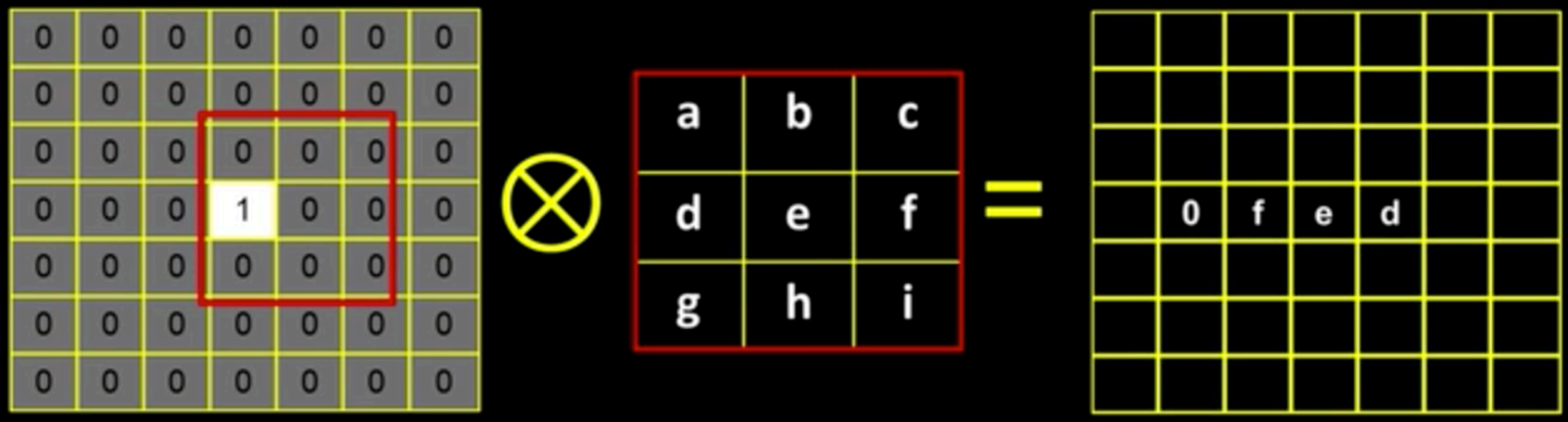
F(x,y) *H(u,v)* *G(x,y)*

Filtering an impulse signal



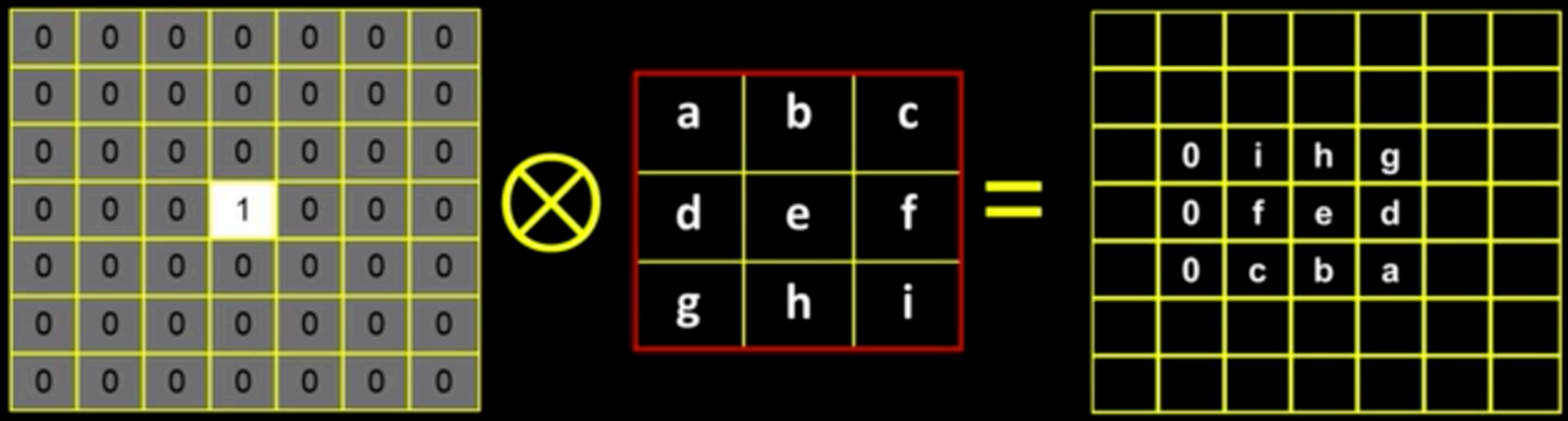
F(x,y) *H(u,v)* *G(x,y)*

Filtering an impulse signal



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

Filtering an impulse signal



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

Filtering an impulse signal

Assuming center coordinate is “reference point”.

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	i	h	g		
	0	f	e	d		
	0	c	b	a		

$G(x,y)$

Q&A 09

Quiz

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

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

Quiz

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

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

$M^2 N^2$.

Correlation vs Convolution

Cross-correlation

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

$$G = H \otimes F$$

K. Grauman

Correlation vs Convolution

Cross-correlation

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

$$G = H \otimes F$$

Convolution

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

$$G = H \star F$$

K. Grauman

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

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

$$G = H \star F$$

K. Grauman

Convolution

Centered at zero!

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

$$G = H * F$$

Notation for convolution operator

Correlation:
Indices in F corresponding to the cells in H above

i-1, j-1	i-1,j	i-1, j+1		
	i,j			
i+1, j-1		i+1,j+1		

k=1
Indices for H:

-1,-1	–1,0	–1,1
	0,0	
1,-1		1,1

Convolution:
Indices in F corresponding to the cells in H above

i+1,j+1	i+1,j	i+1, j-1		
	i,j			
i-1, j+1		i-1, j-1		

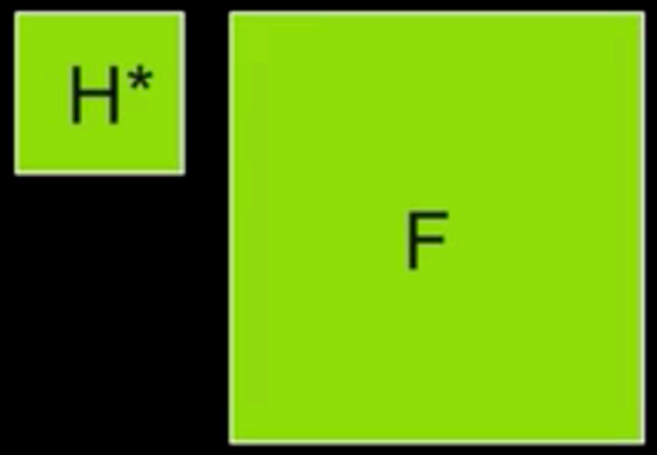
Convolution

Centered at zero!

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

$$G = H * F$$

Notation for convolution operator



K. Grauman

Convolution

Centered at zero!

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

$$G = H * F$$

Notation for convolution operator

K. Grauman

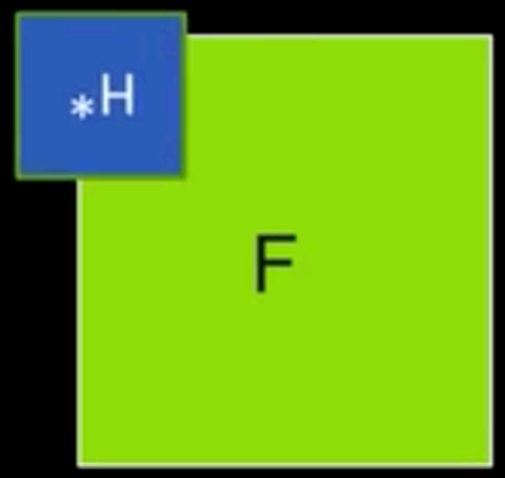
Convolution

Centered at zero!

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

$$G = H * F$$

Notation for convolution operator



K. Grauman

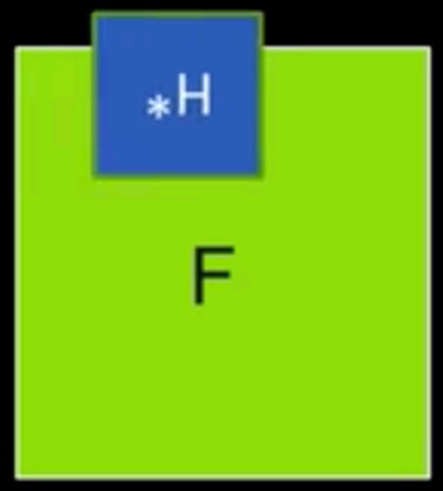
Convolution

Centered at zero!

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

$$G = H * F$$

Notation for convolution operator



K. Grauman

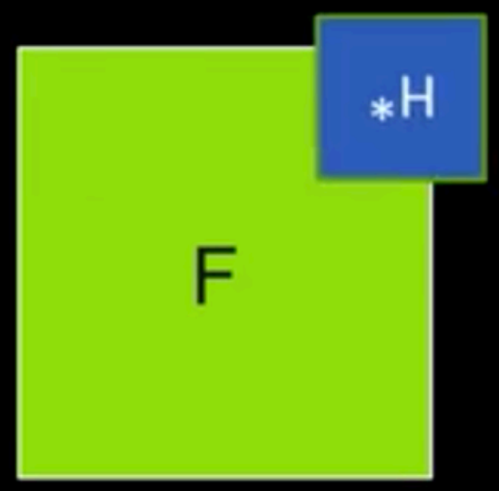
Convolution

Centered at zero!

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

$$G = H * F$$

Notation for convolution operator



K. Grauman

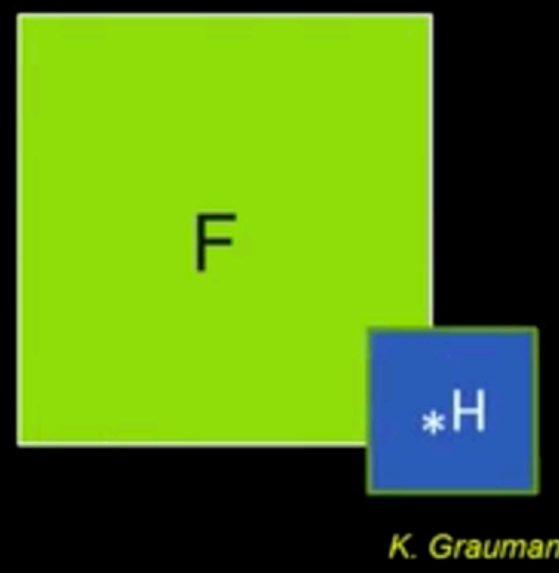
Convolution

Centered at zero!

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

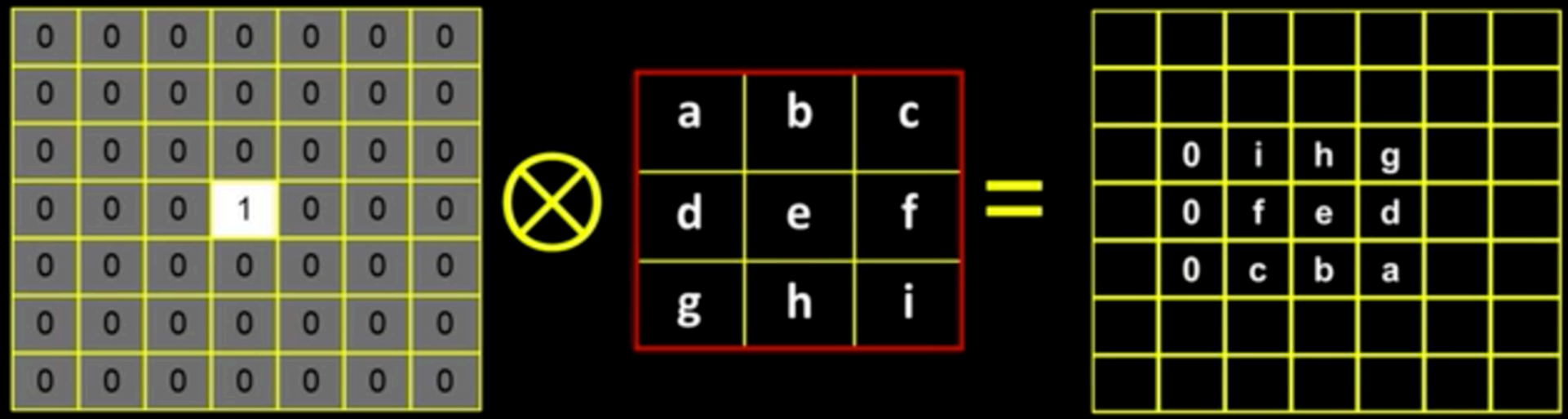
$$G = H * F$$

Notation for convolution operator



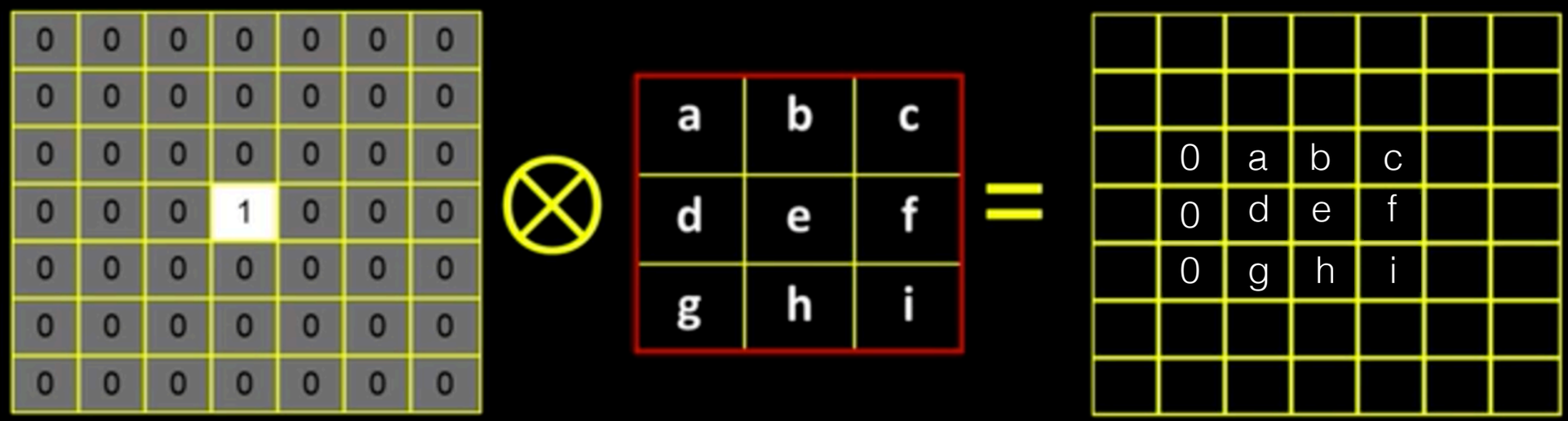
K. Grauman

Correlation:



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

Convolution:



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

Q&A 10

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

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

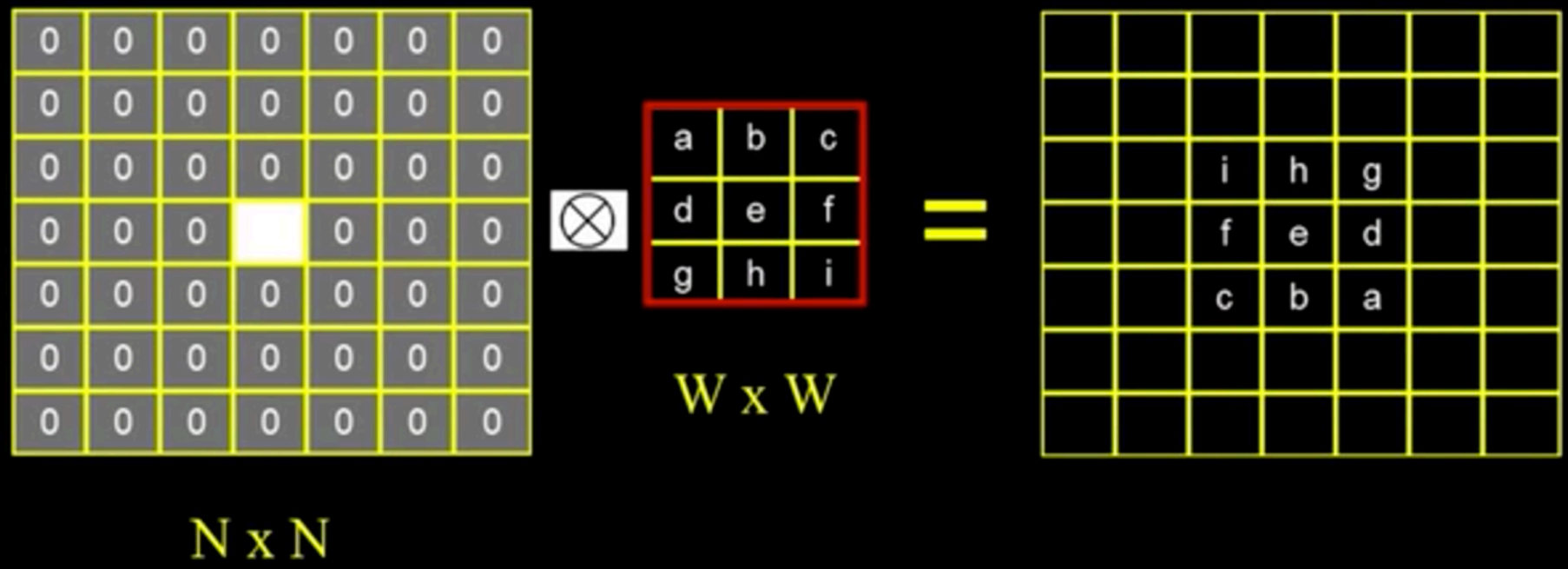
  (f * g) * h = f * (g * h)
- Identity:

  unit impulse e = [..., 0, 0, 1, 0, 0, ...]. f * e = f
- Differentiation: $\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$

Separability

Computational Complexity

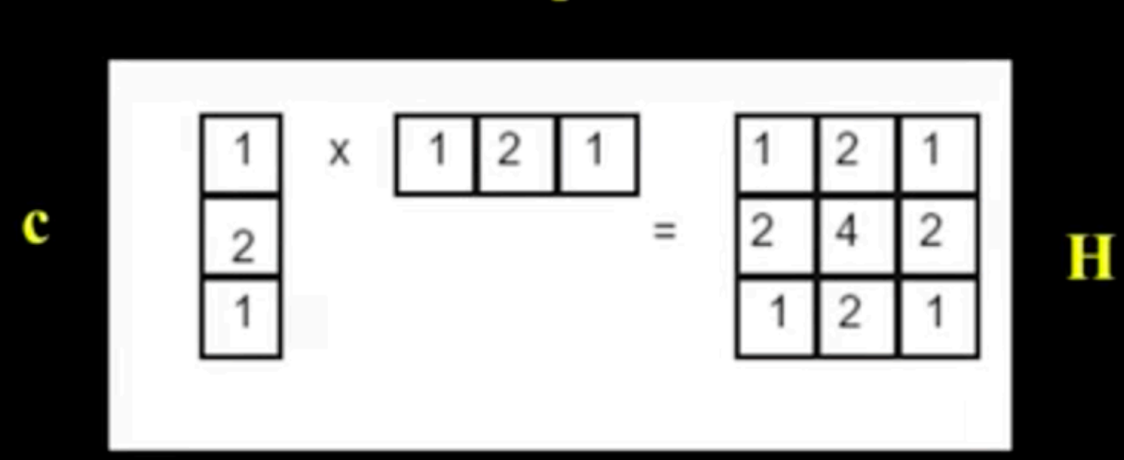
- If an image is *NxN* and a kernel (filter) is WxW, how many multiplies do you need to compute a convolution?



- 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:



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:

r

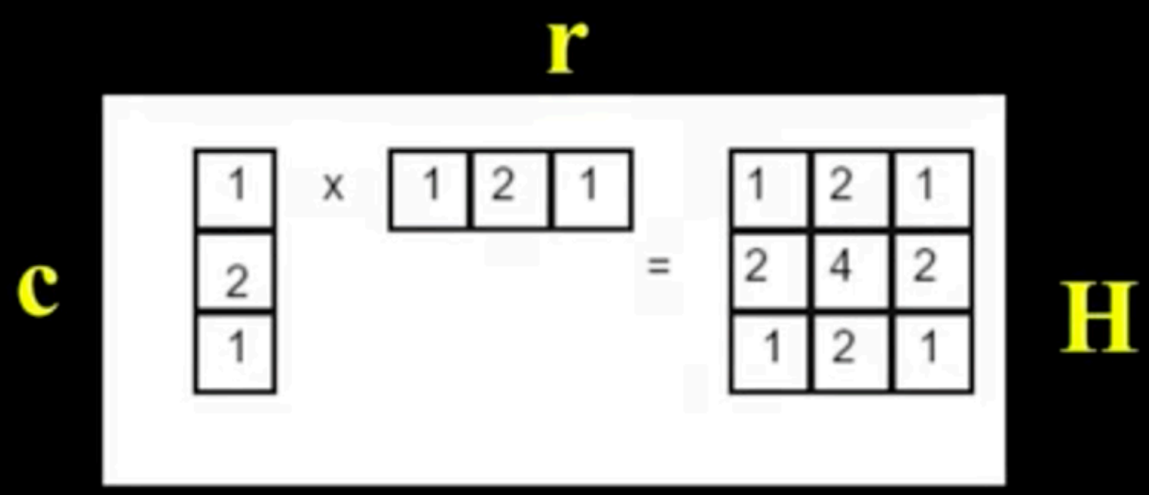
c

1	x	1	2	1	=	1	2	1
2						2	4	2
1						1	2	1

H

c * r = H

Separability



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

- 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.

Q&A 11

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

Quiz?

True or false: Division is a linear operation.

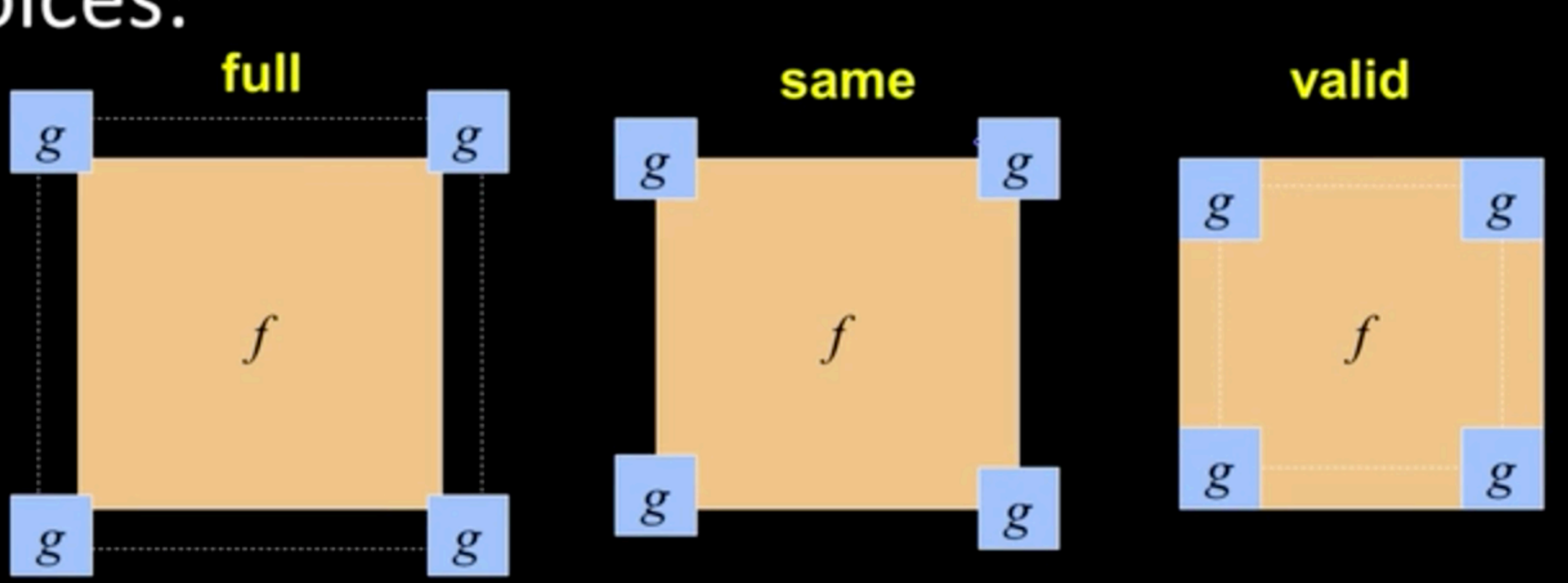
a) False because $X/(Y+Z) \neq X/Y + X/Z$

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

c) I have no idea

Boundary issues

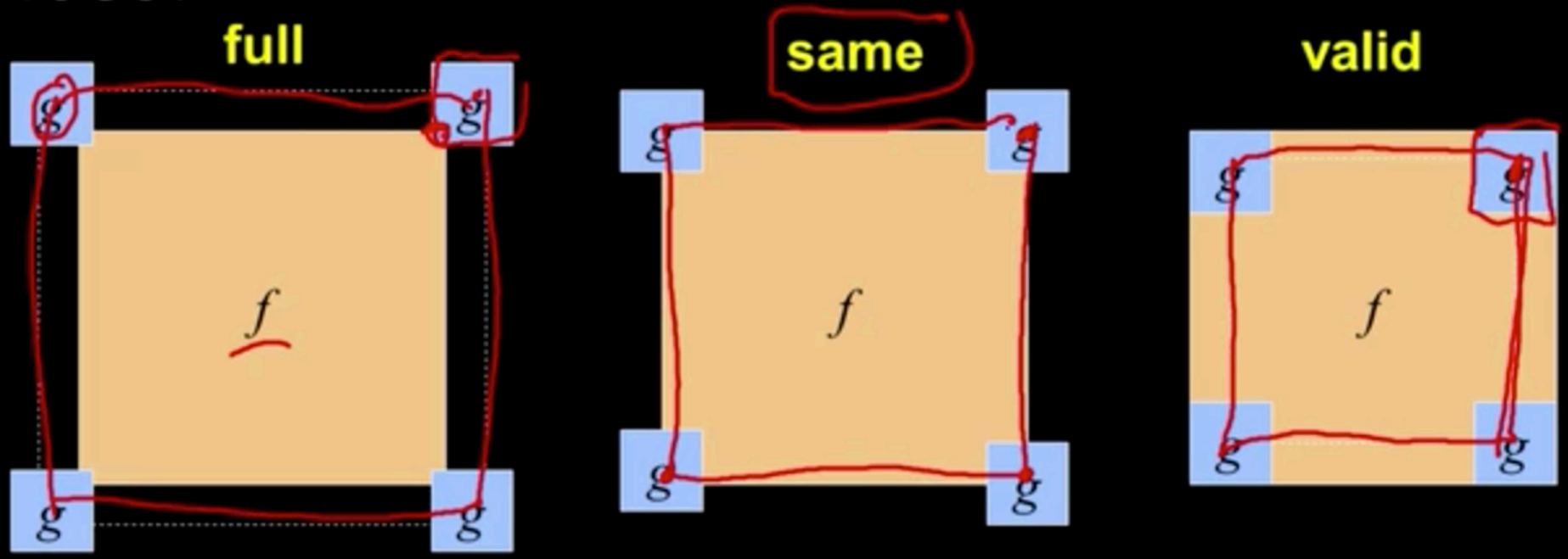
- What is the size of the output?
- Using old Matlab nomenclature we have three choices:



Source: S. Lazebnik

Boundary issues

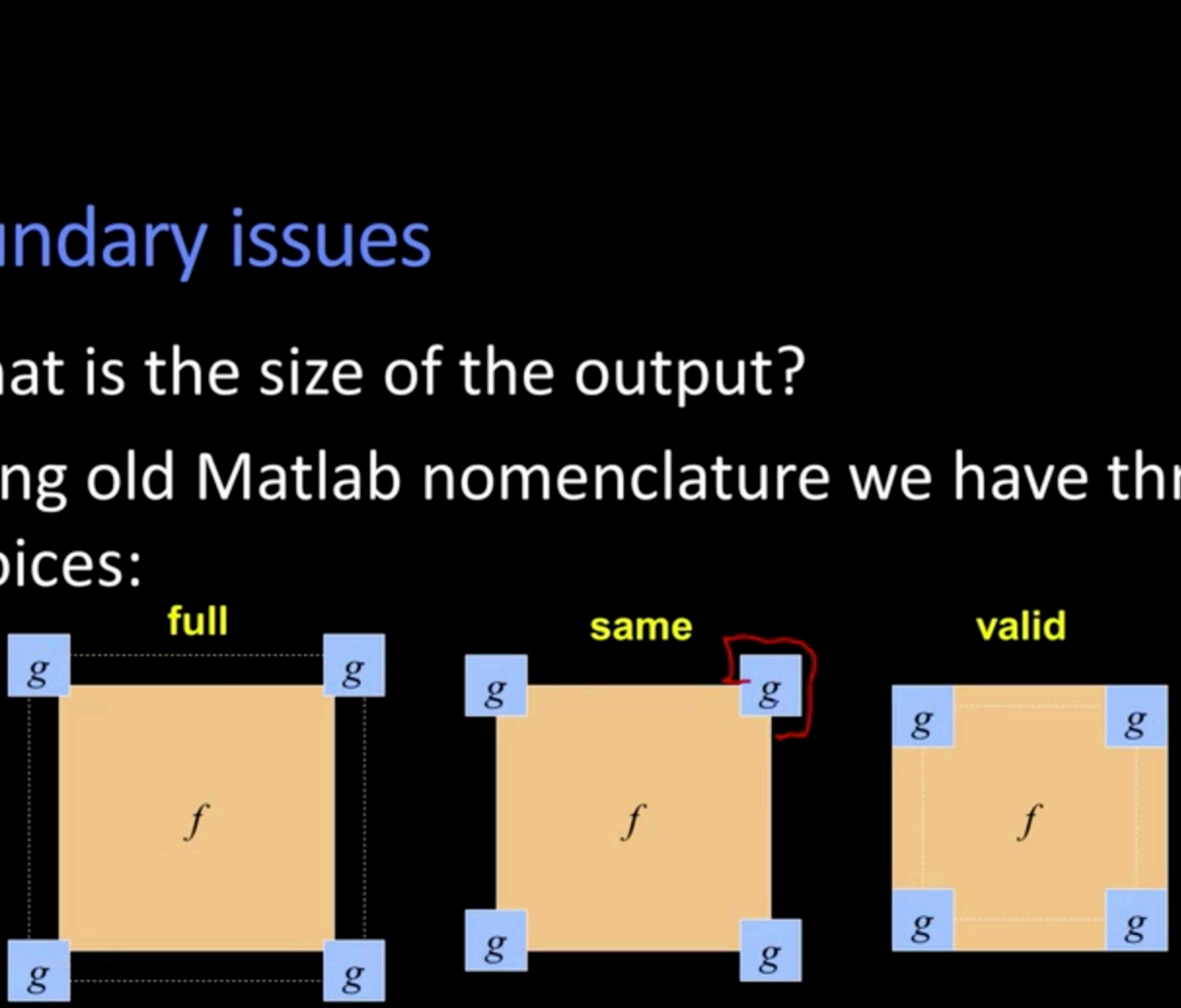
- What is the size of the output?
- Using old Matlab nomenclature we have three choices:



Source: S. Lazebnik

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?

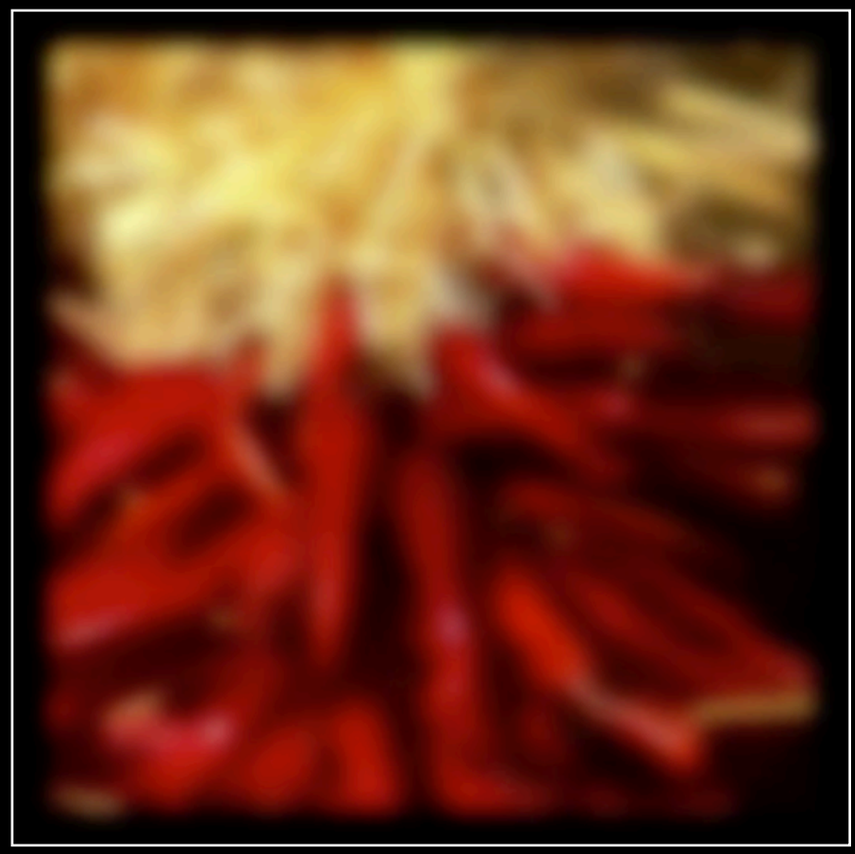
- methods:
  - clip filter (black)

Source: S. Marschner

Boundary issues

What about near the edge?

- methods:
  - clip filter (black)

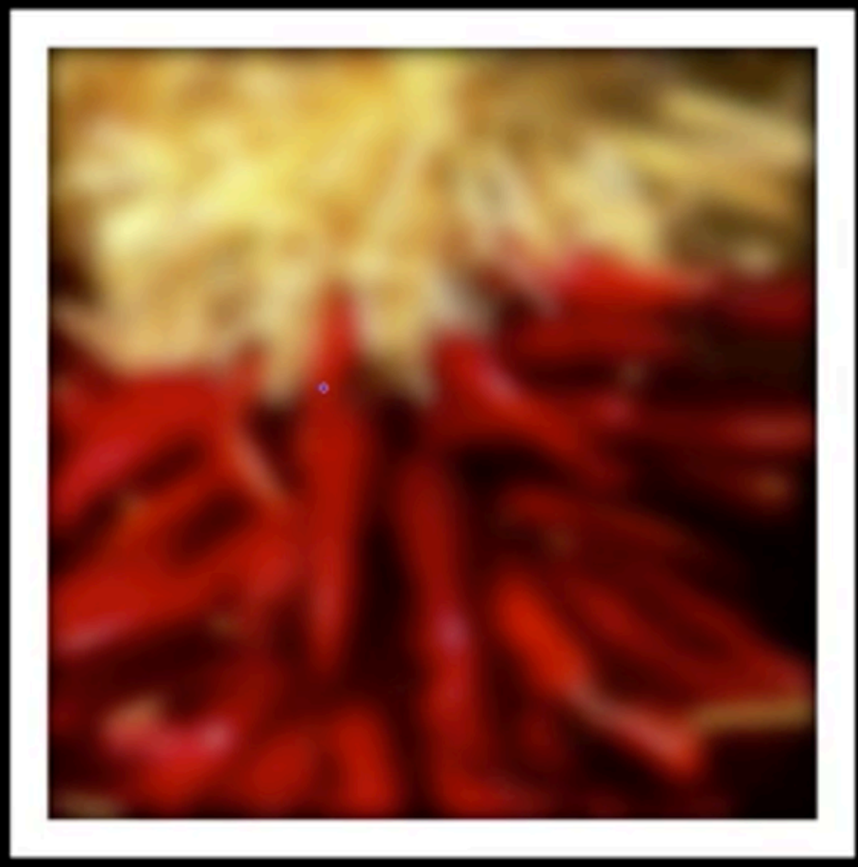


Source: S. Marschner

Boundary issues

What about near the edge?

- methods:
  - clip filter (black)



Source: S. Marschner

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

Source: S. Marschner

Boundary issues

What about near the edge?

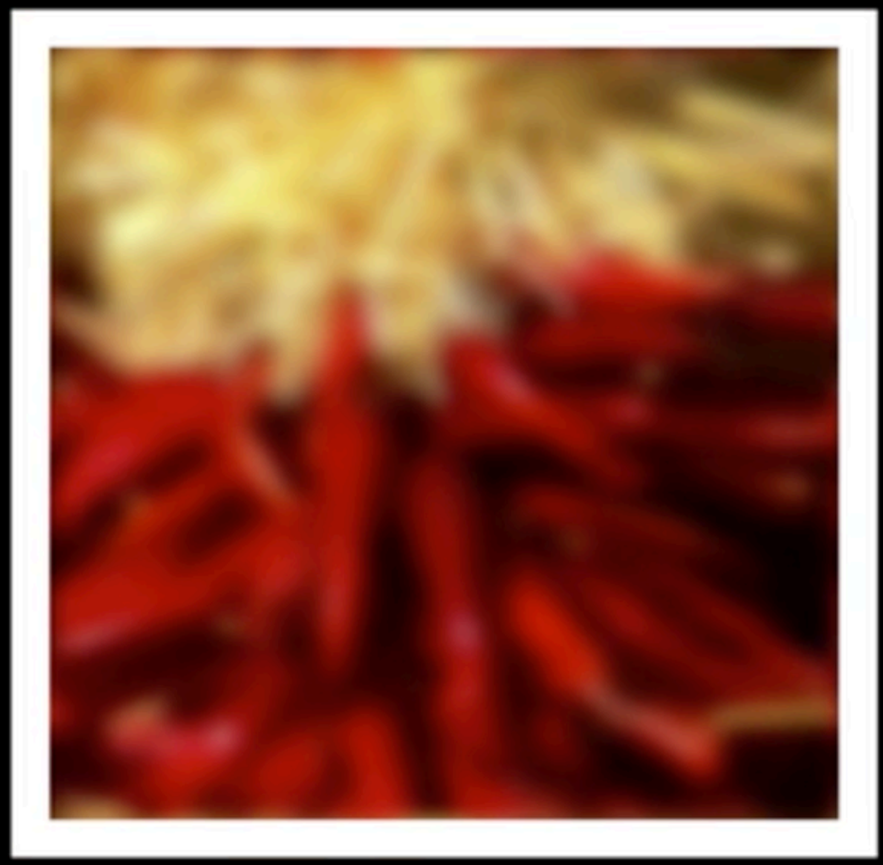
- methods:
  - clip filter (black)
  - wrap around

Source: S. Marschner

Boundary issues

What about near the edge?

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

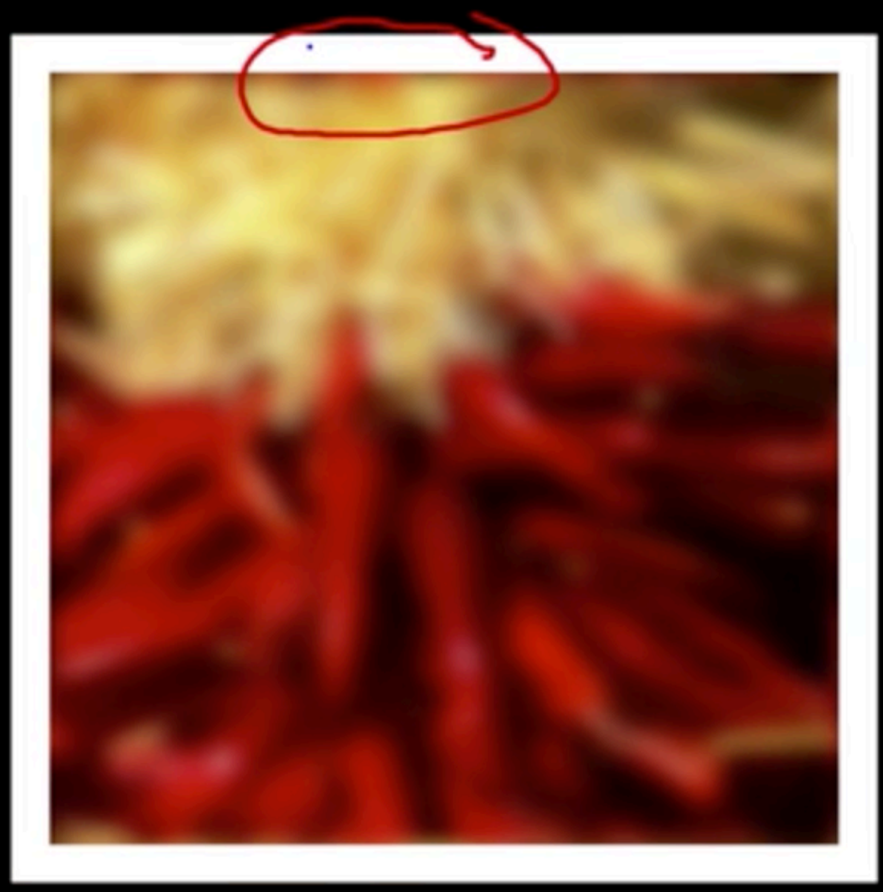


Source: S. Marschner

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?

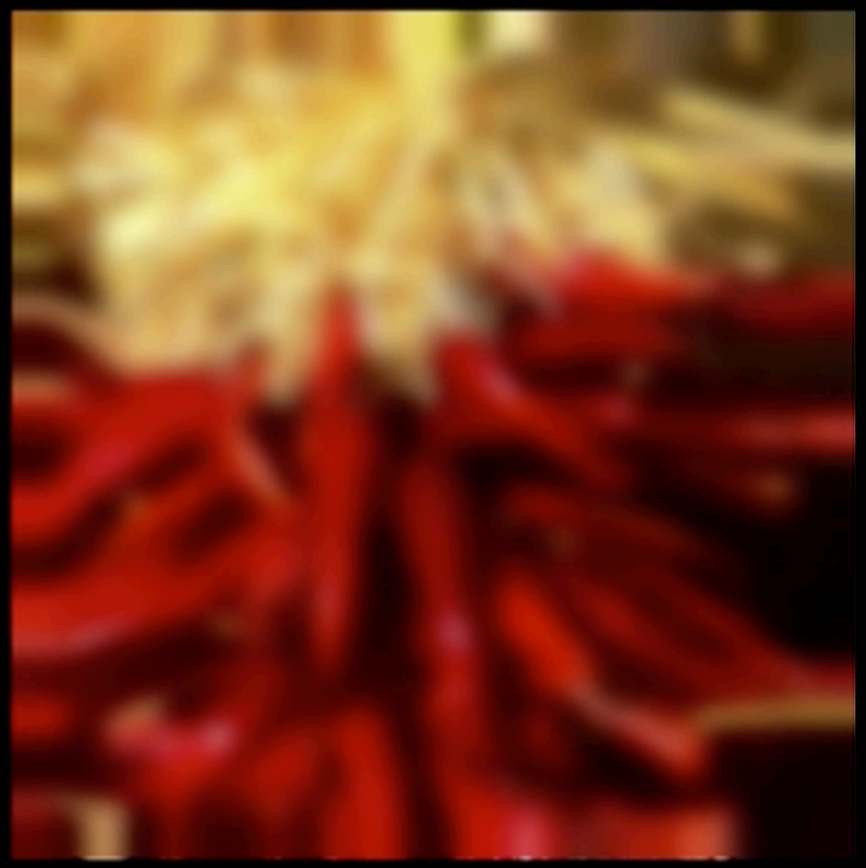
- methods:
  - clip filter (black)
  - wrap around
  - copy edge

Source: S. Mars

Boundary issues

What about near the edge?

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

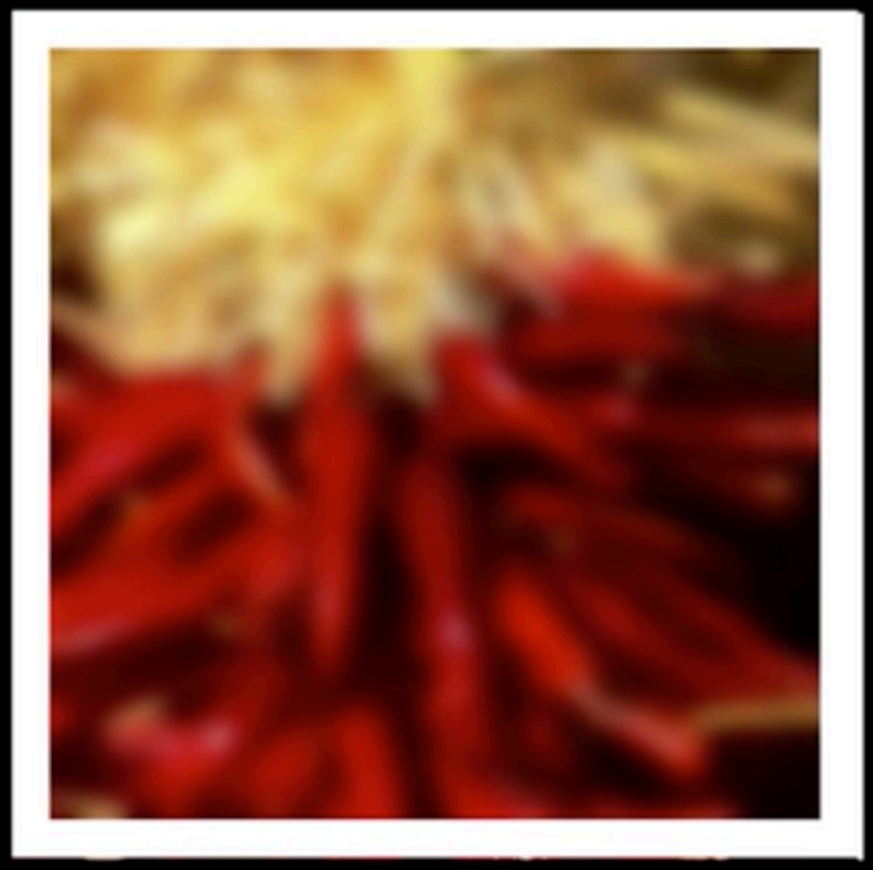


Source: S. Mars

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?

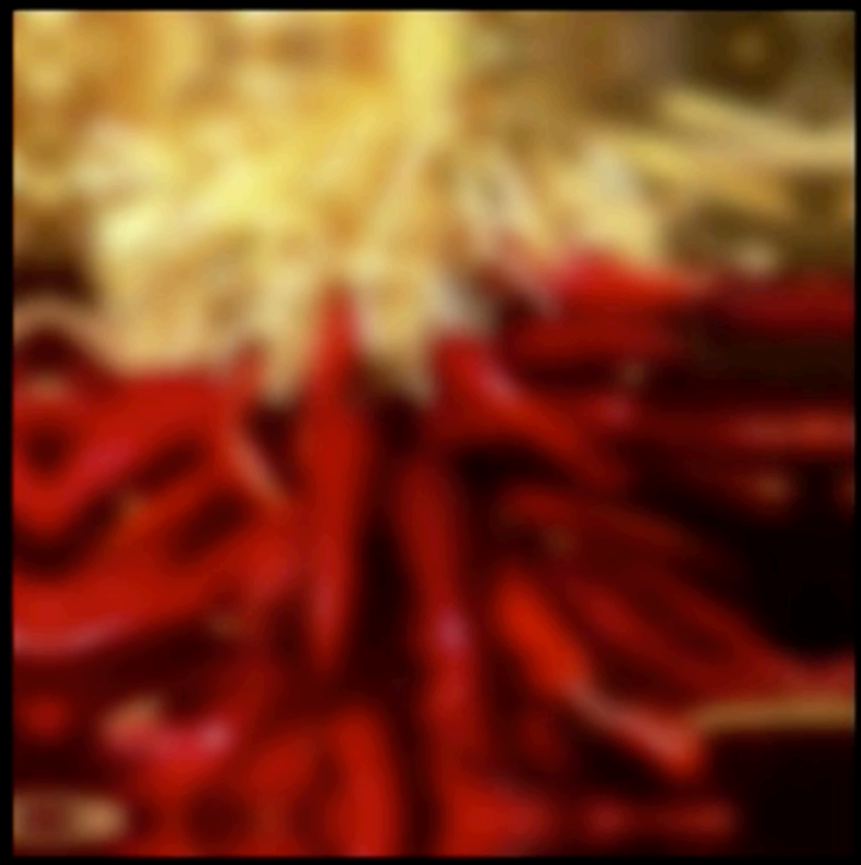
- methods:
  - clip filter (black)
  - wrap around
  - copy edge
  - reflect across edge

Source: S. Marschner

Boundary issues

What about near the edge?

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

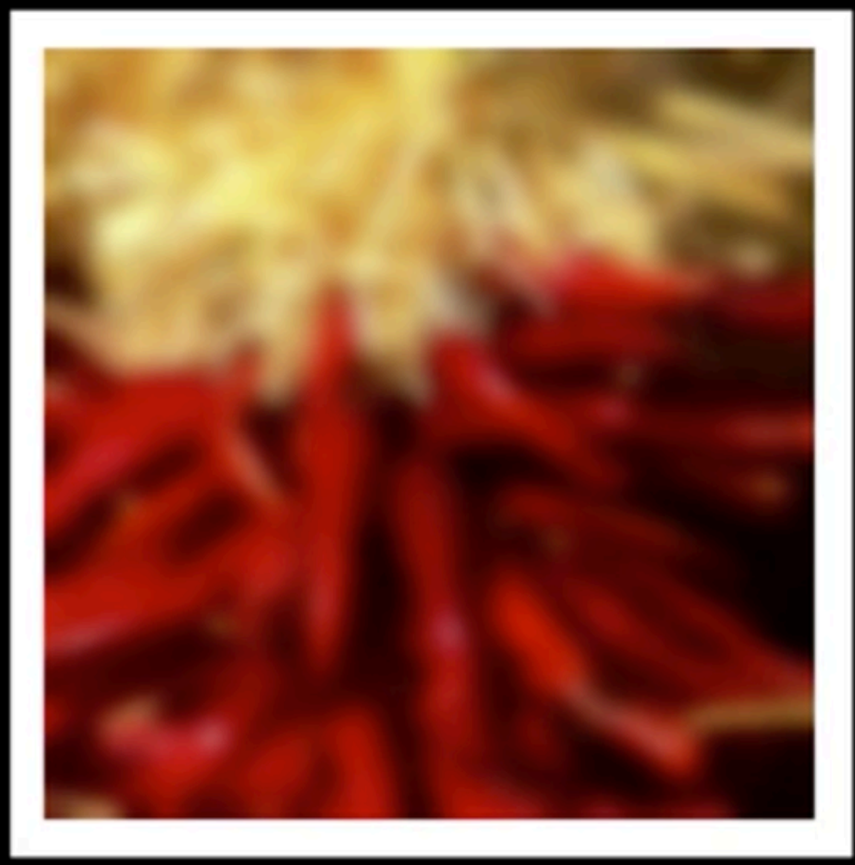


Source: S. Marschner

Boundary issues

What about near the edge?

- methods:
  - 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’)

Q&A 12

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.

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)

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 with
correlation

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

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

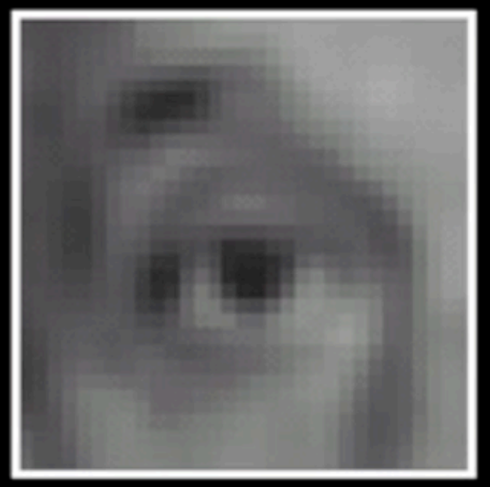
?

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)

Practice with linear filters

Original

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

?

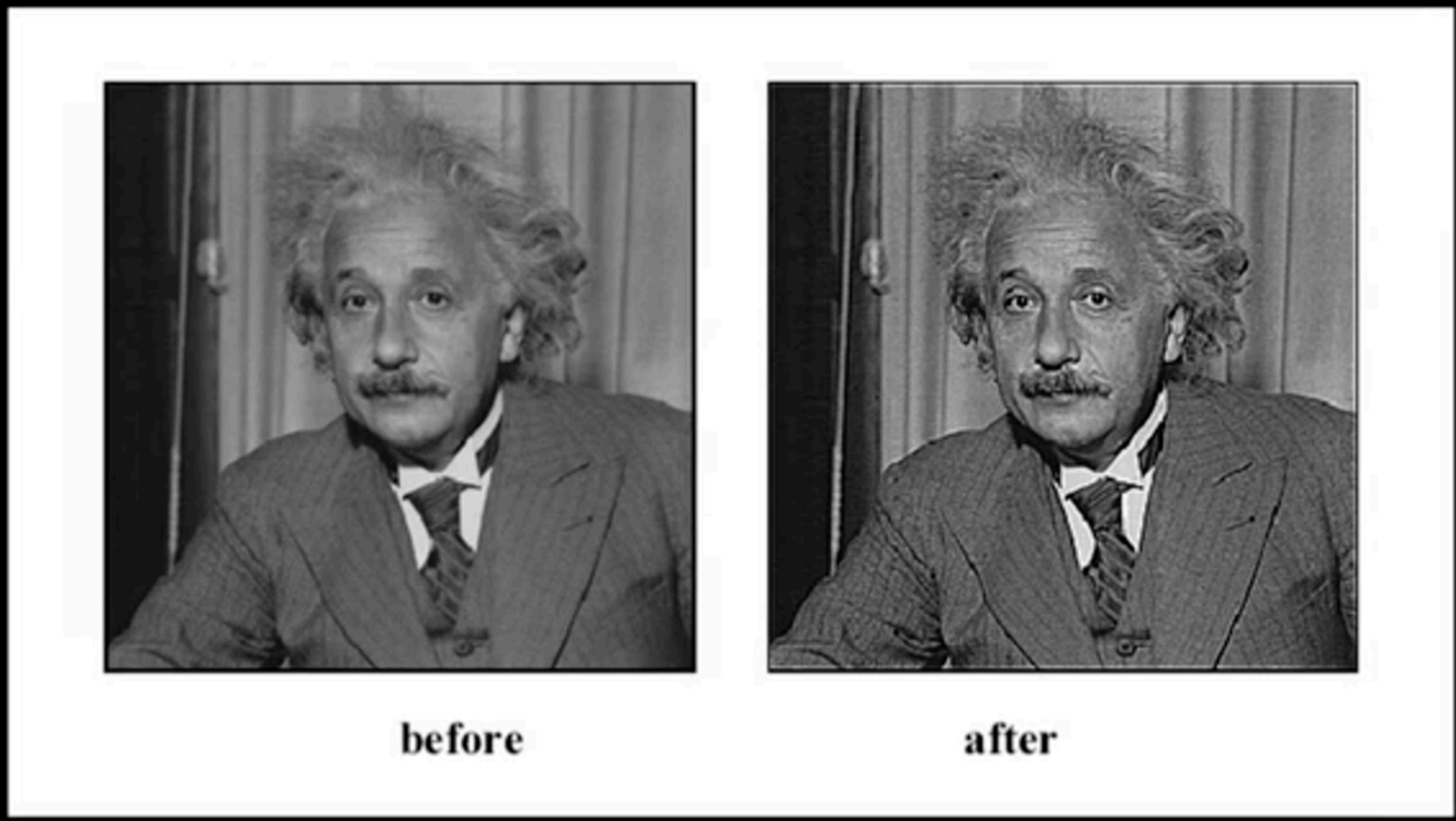
Practice with linear filters

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

Original

Sharpening Filter:
Accentuates differences with local average

Filtering examples: sharpening



before

after

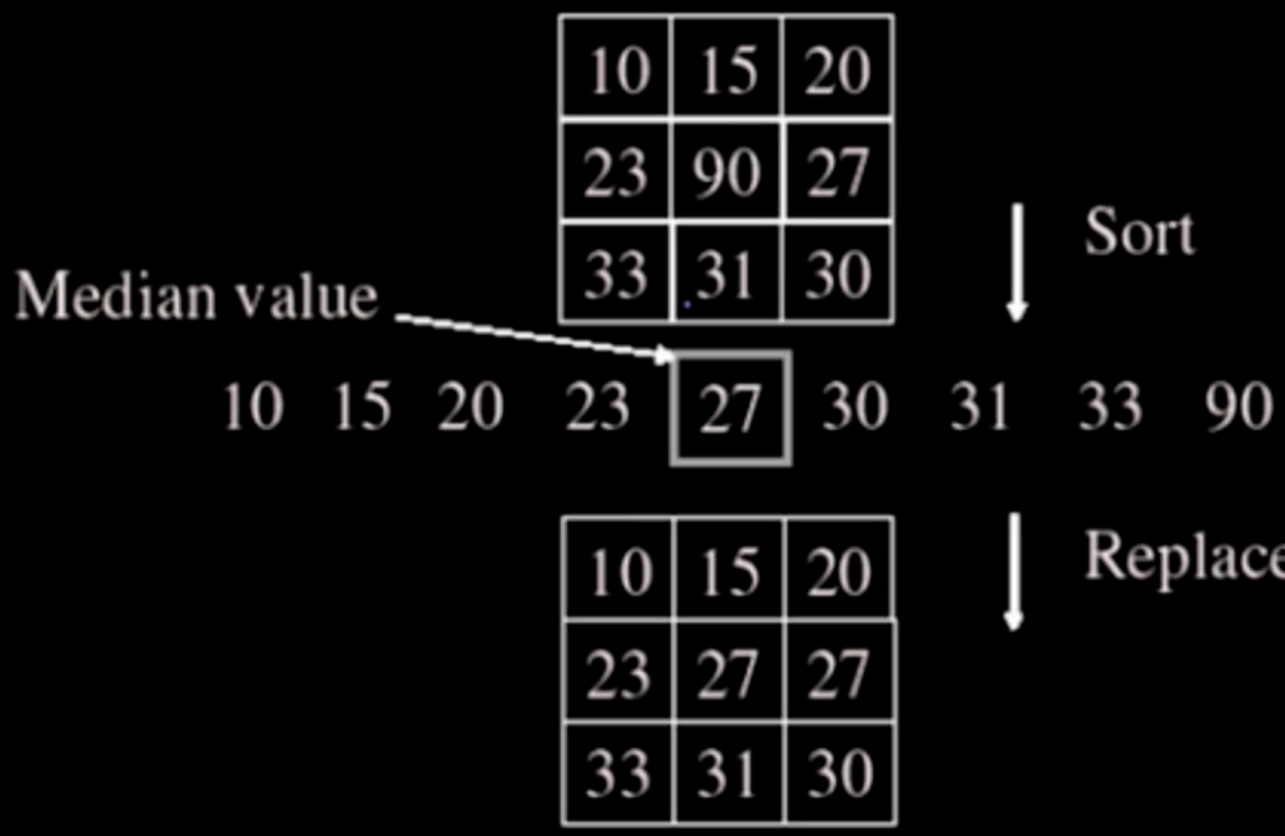
K. Grauman

Different kinds of noise

Additive Gaussian noise

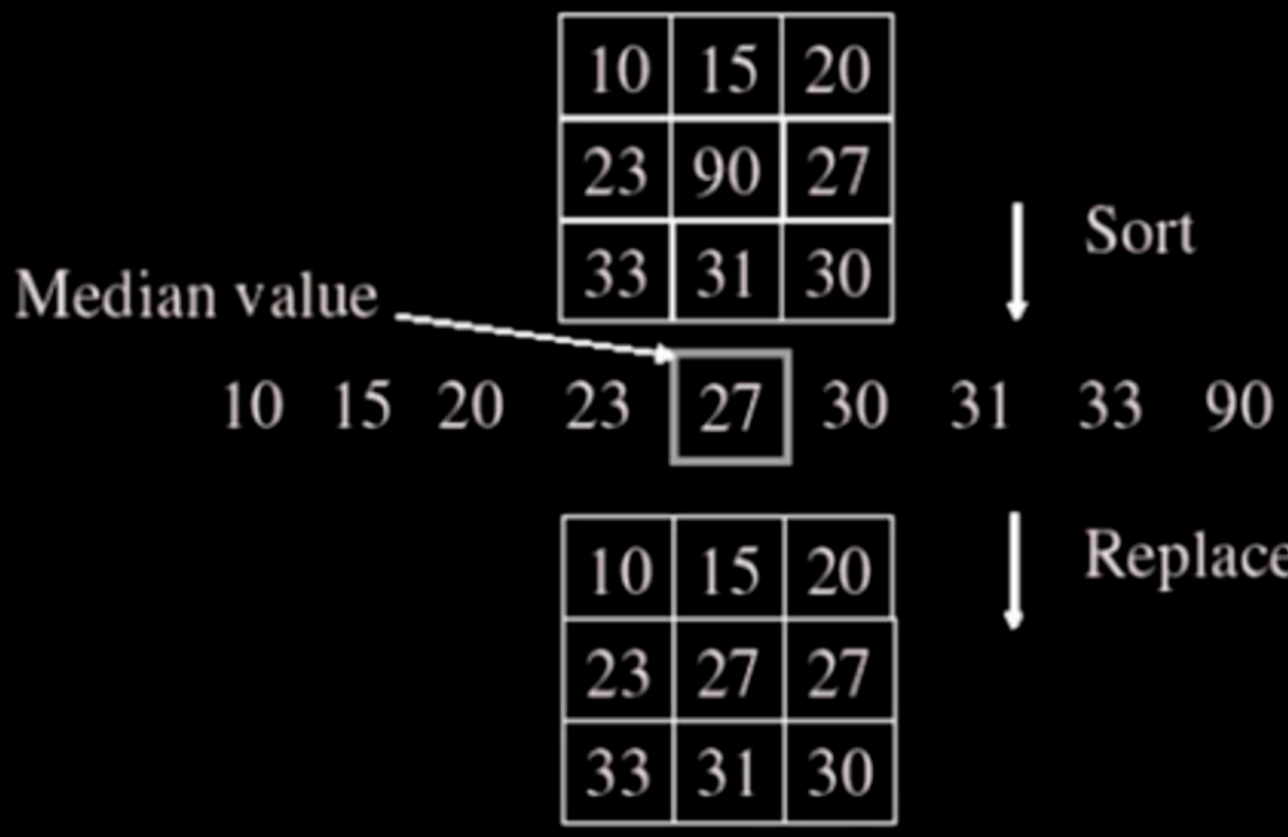
Salt and pepper noise

Median filter



K. Grauman

Median filter

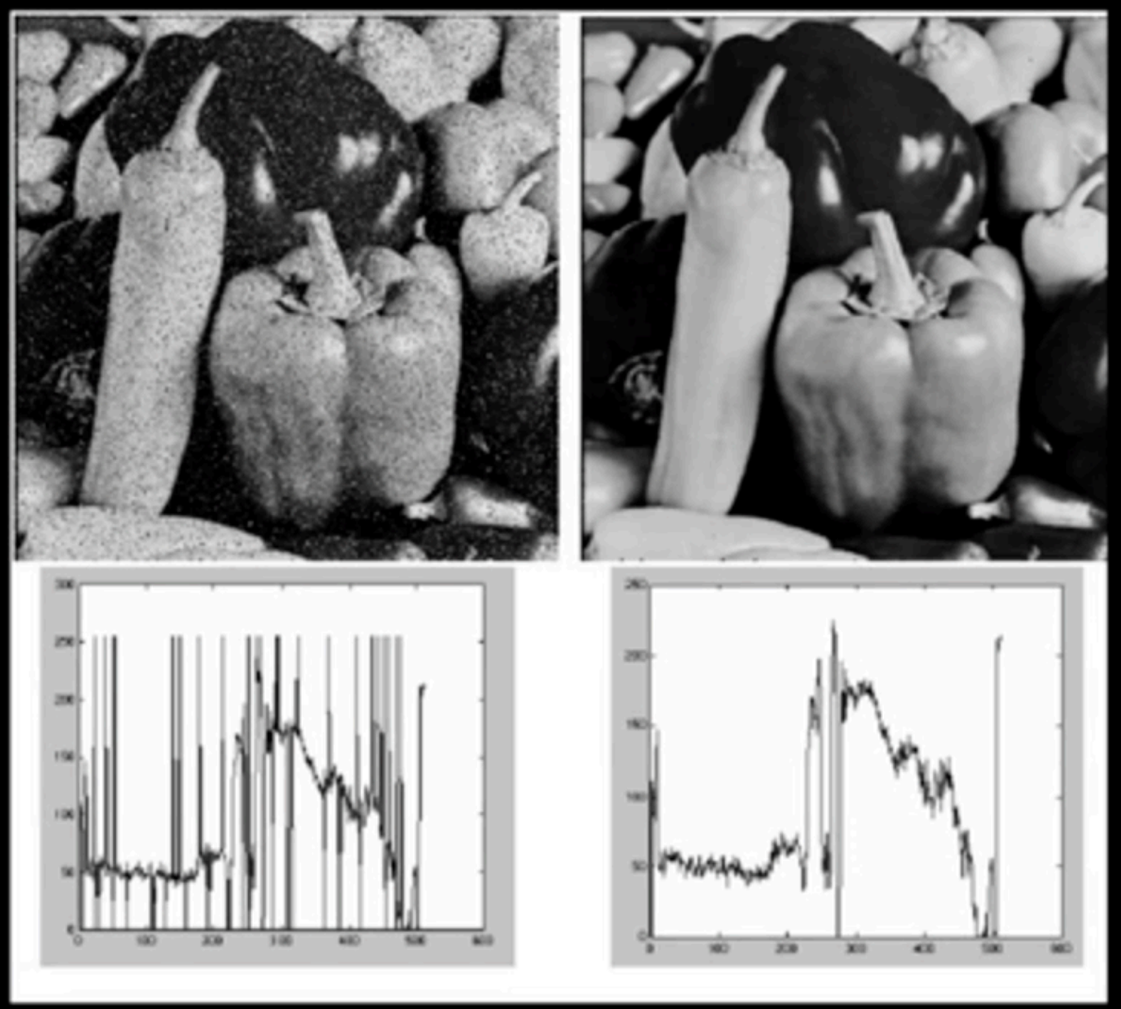


- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Linear?

K. Grauman

Median filter

Salt and pepper noise →



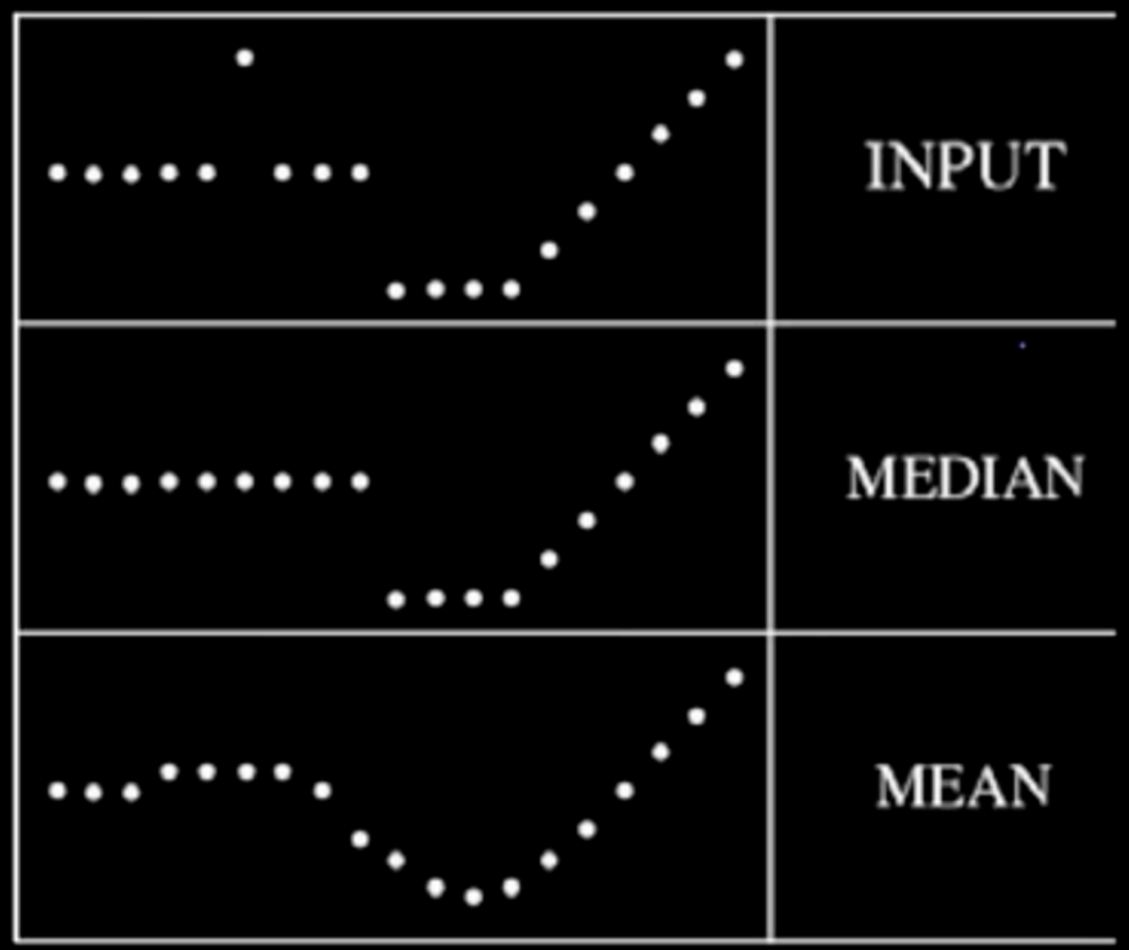
← Median filtered

Plots of a row of the image

Source: M. Hebert

Median filter

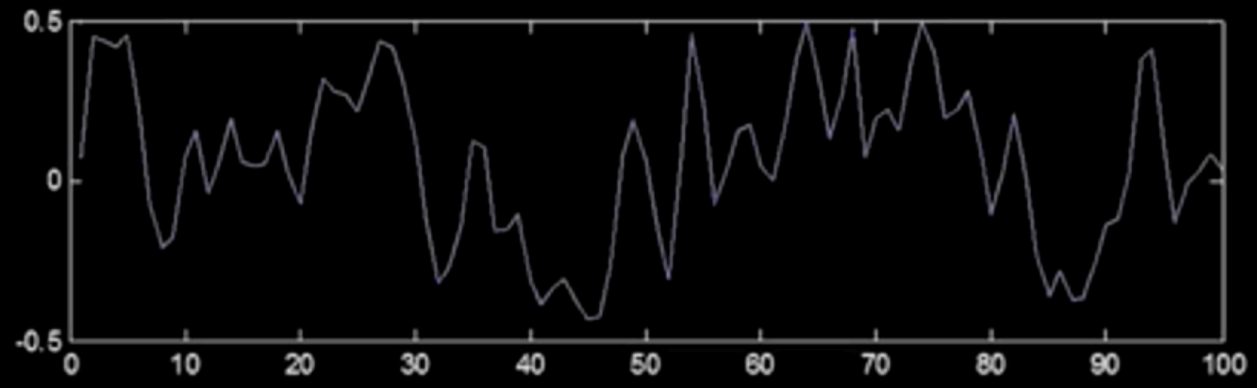
Median filter is edge preserving



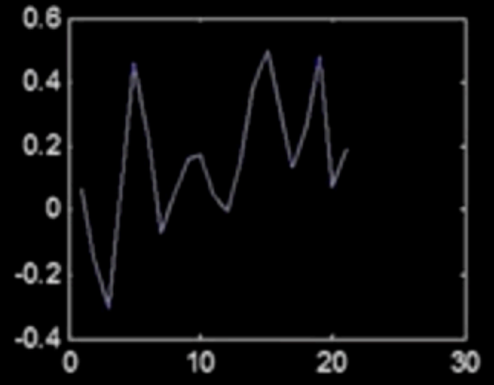
Filters as templates

1D (nx)correlation

Signal

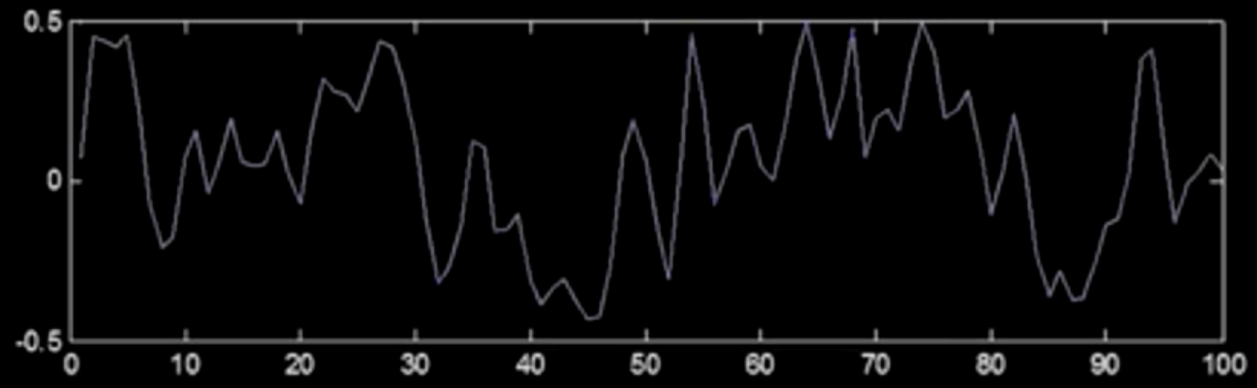


Filter

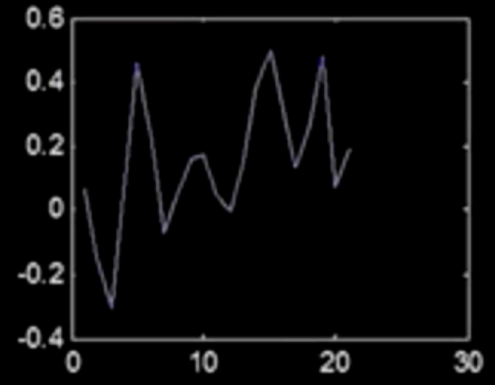


1D (nx)correlation

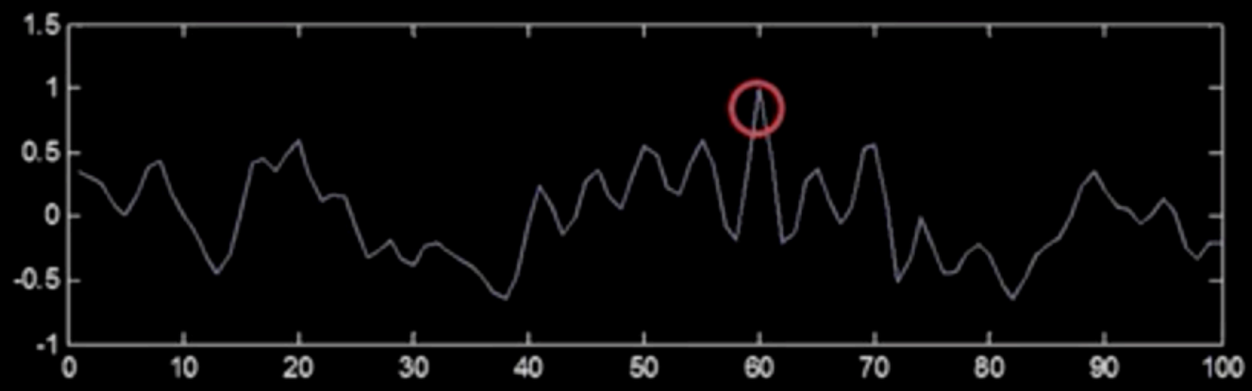
Signal



Filter

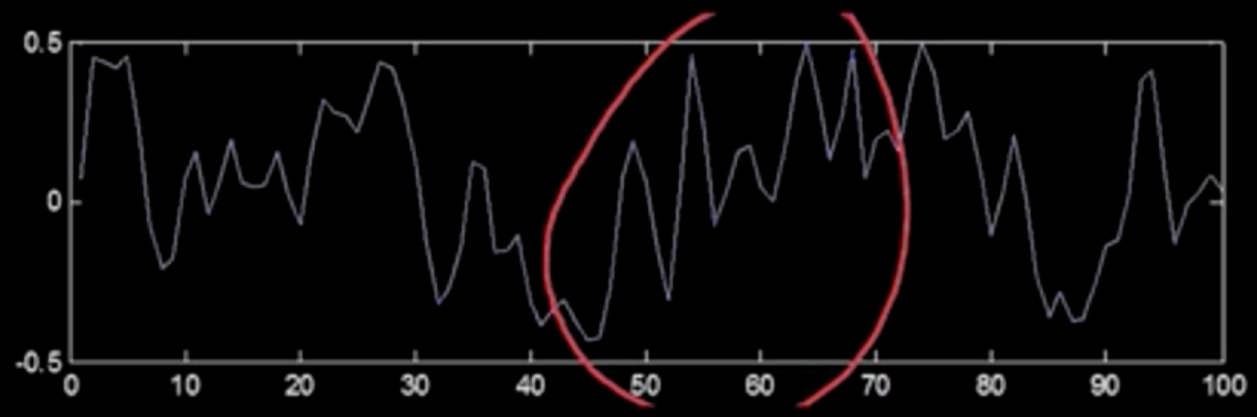


Normalized cross-correlation

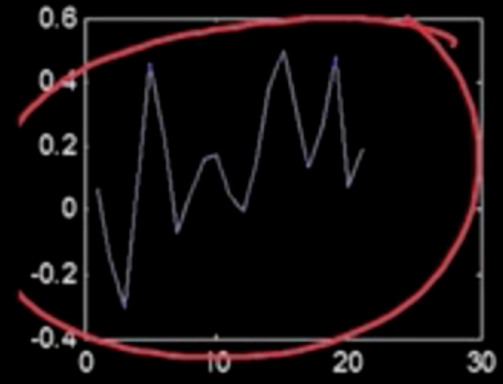


1D (nx)correlation

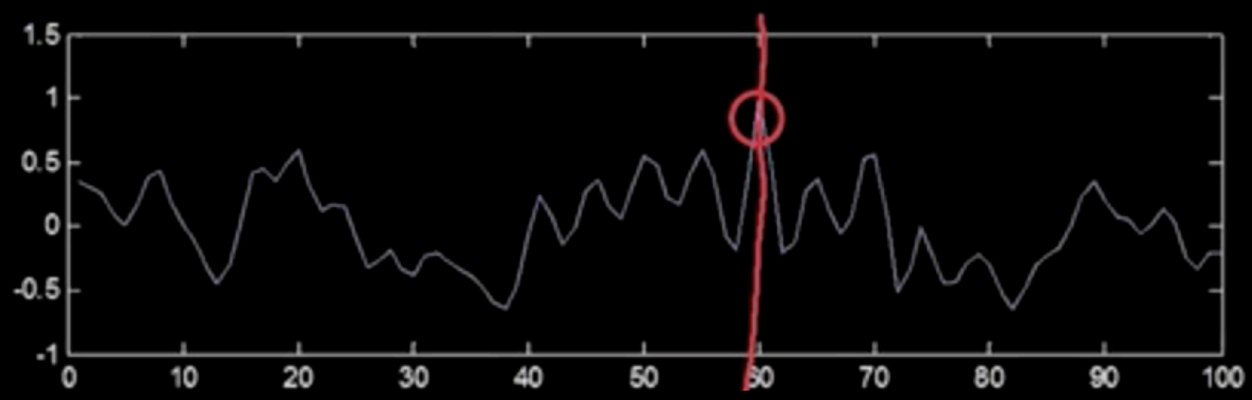
Signal



Filter



Normalized cross-correlation

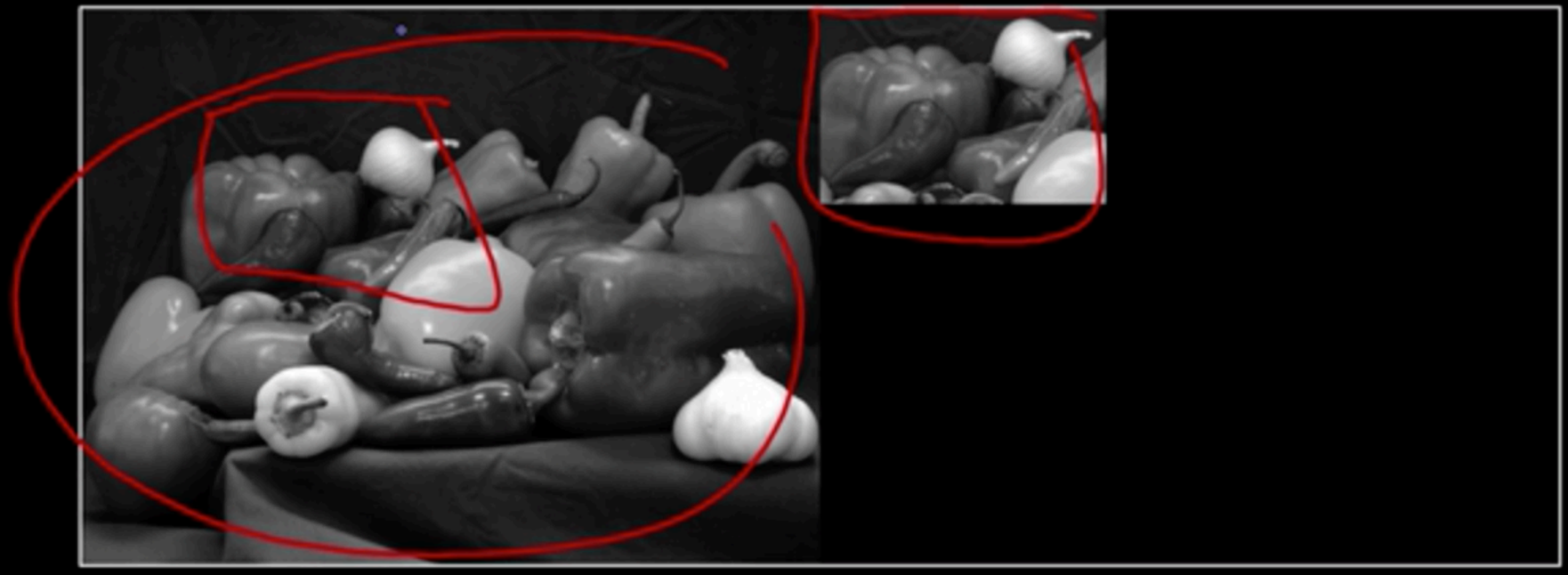


Matlab cross-correlation doc

```
onion   = rgb2gray(imread('onion.png'));
peppers = rgb2gray(imread('peppers.png'));
imshowpair(peppers,onion,'montage')
```

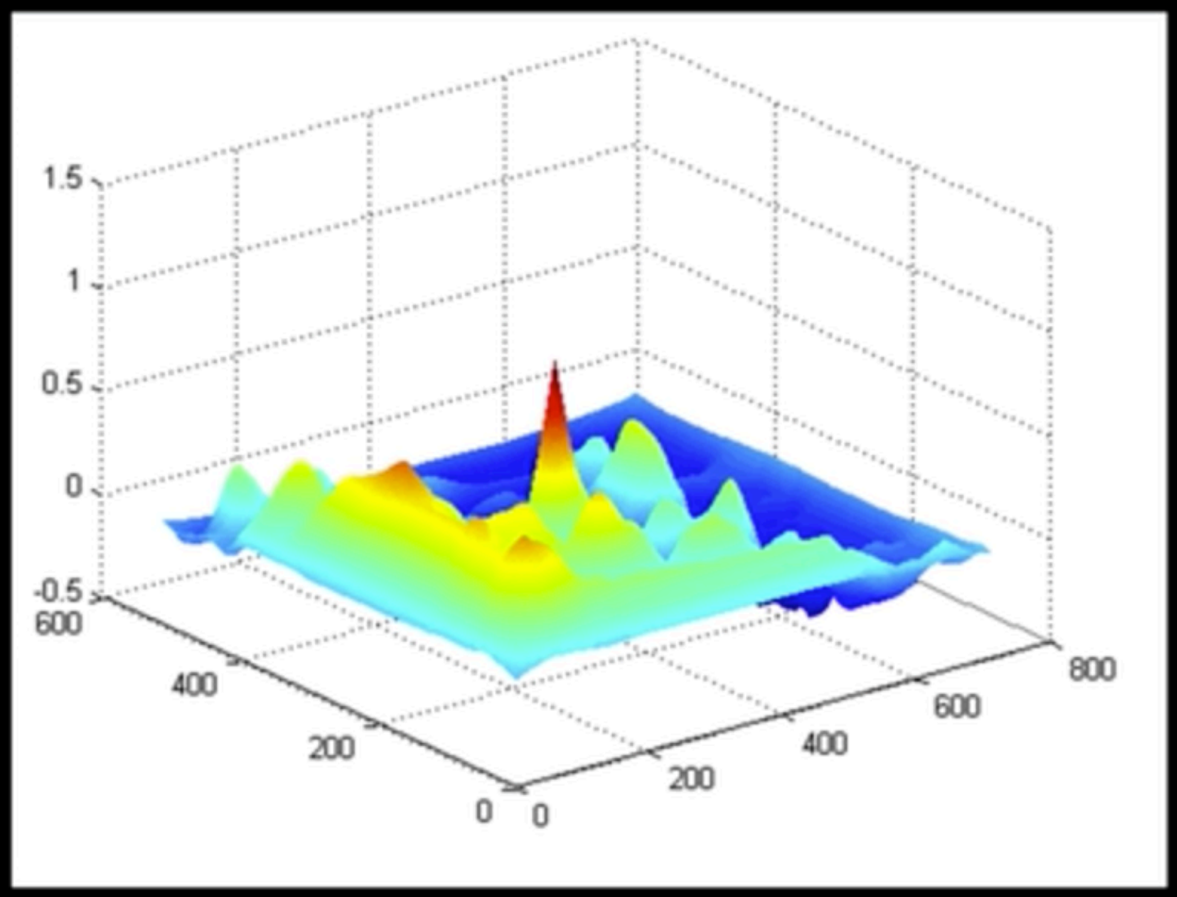
Matlab cross-correlation doc

```
onion   = rgb2gray(imread('onion.png'));
peppers = rgb2gray(imread('peppers.png'));
imshowpair(peppers,onion,'montage')
```



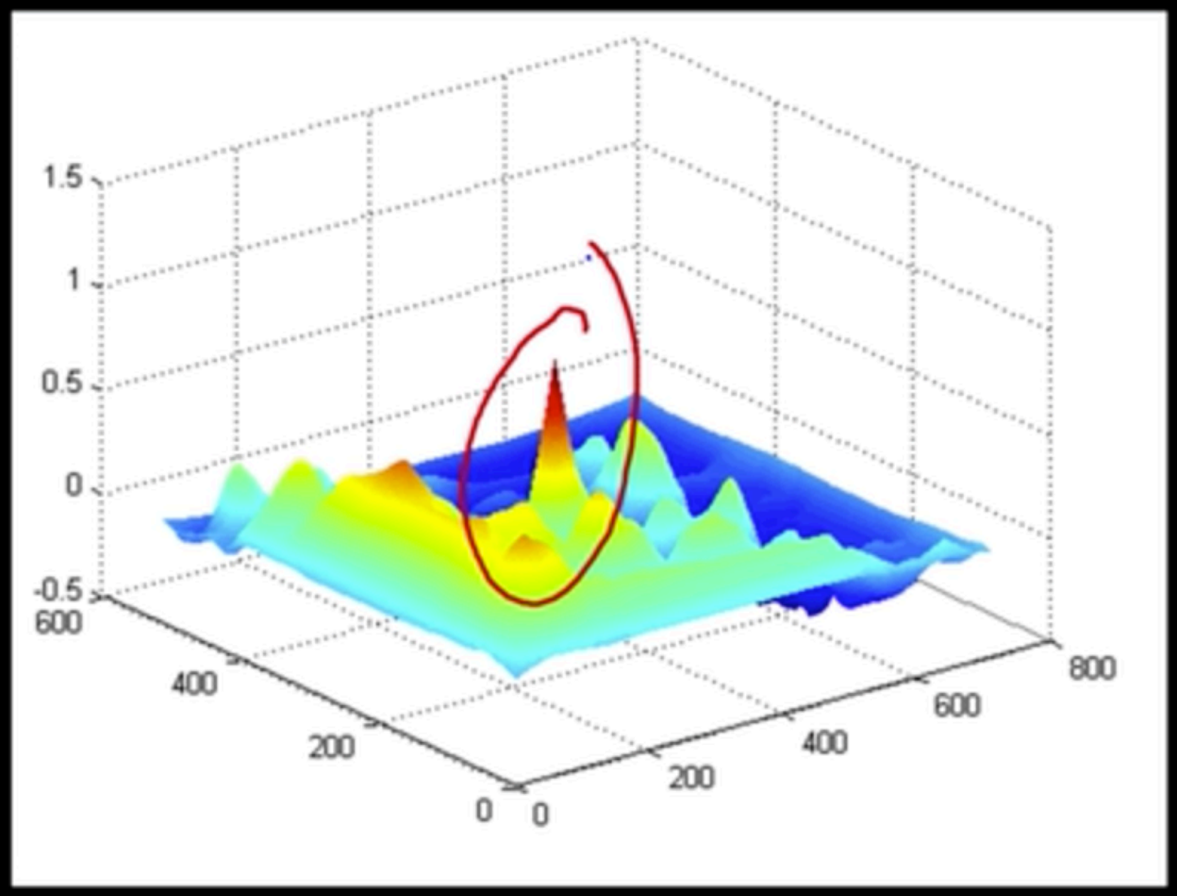
Matlab cross-correlation doc

```
c = normxcorr2(onion,peppers);
figure, surf(c), shading flat;
```

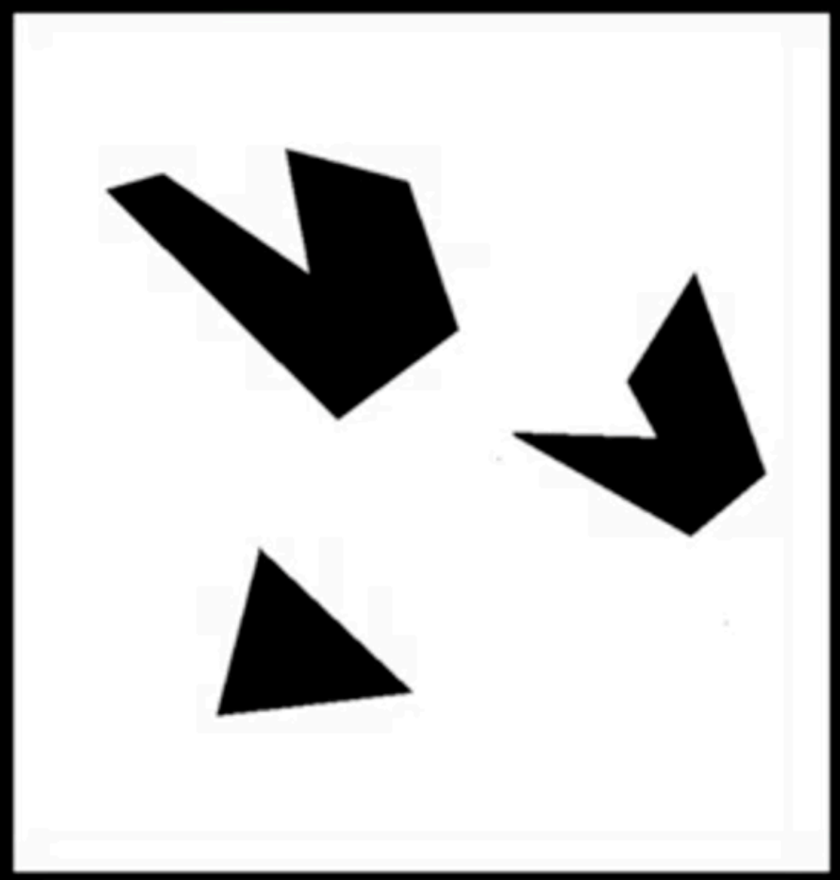


Matlab cross-correlation doc

```
c = normxcorr2(onion,peppers);
figure, surf(c), shading flat;
```



Template matching



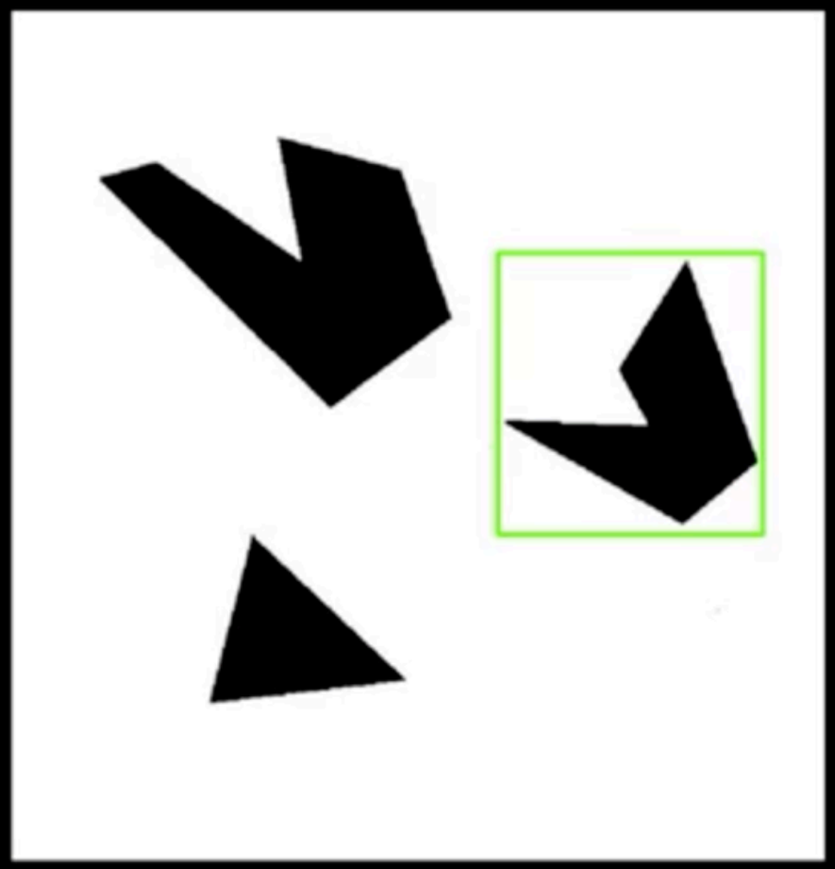
Scene

A toy example

Template (mask)

K. Grauman

Template matching



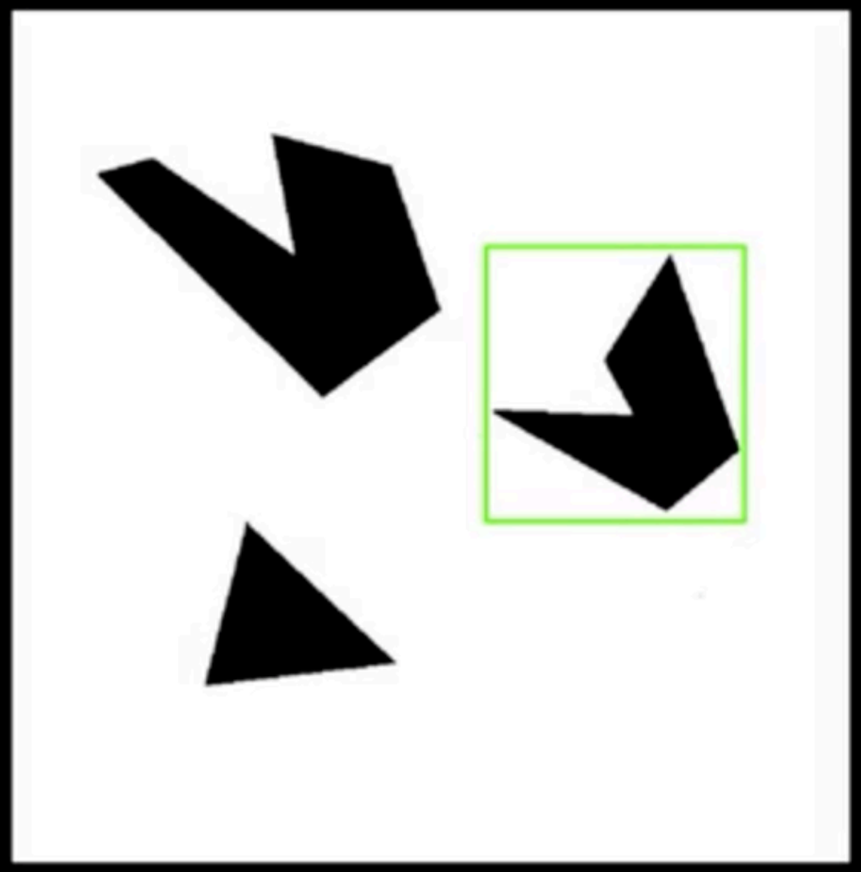
Detected template

A toy example

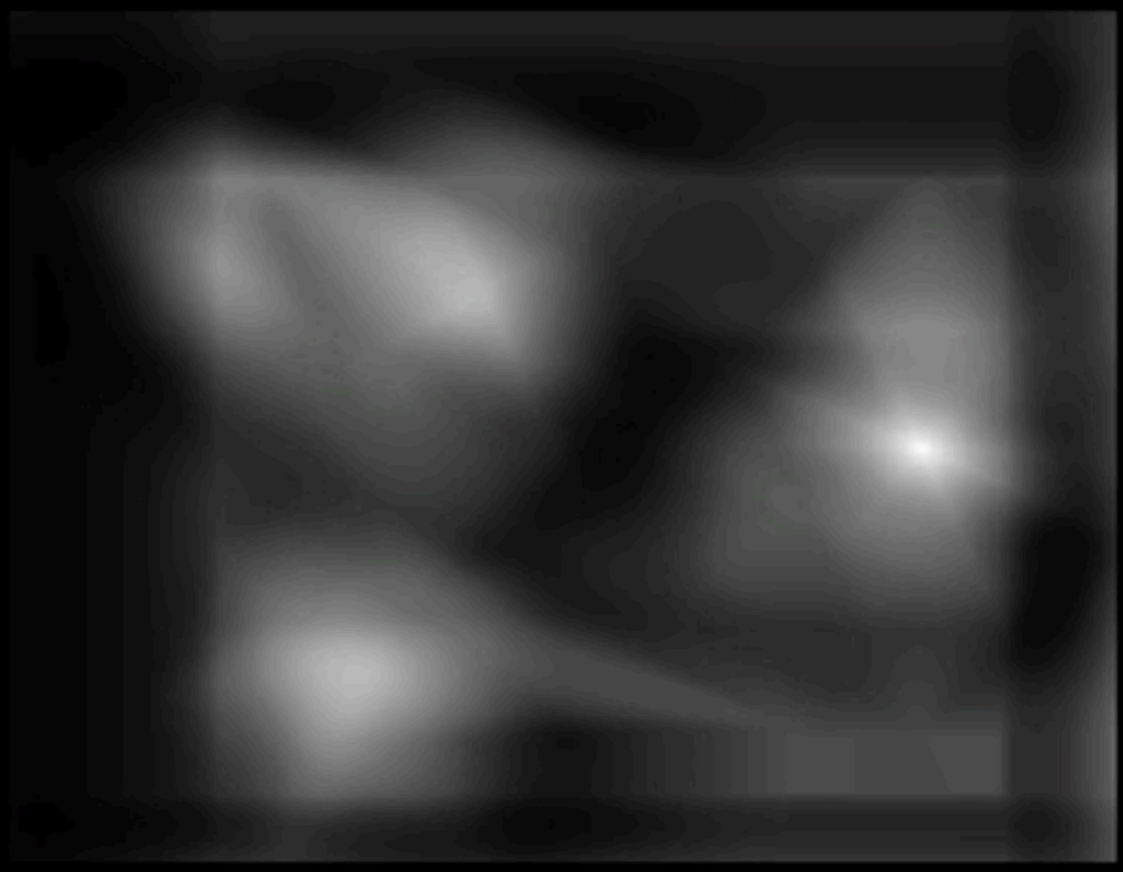
Template (mask)

K. Grauman

Template matching

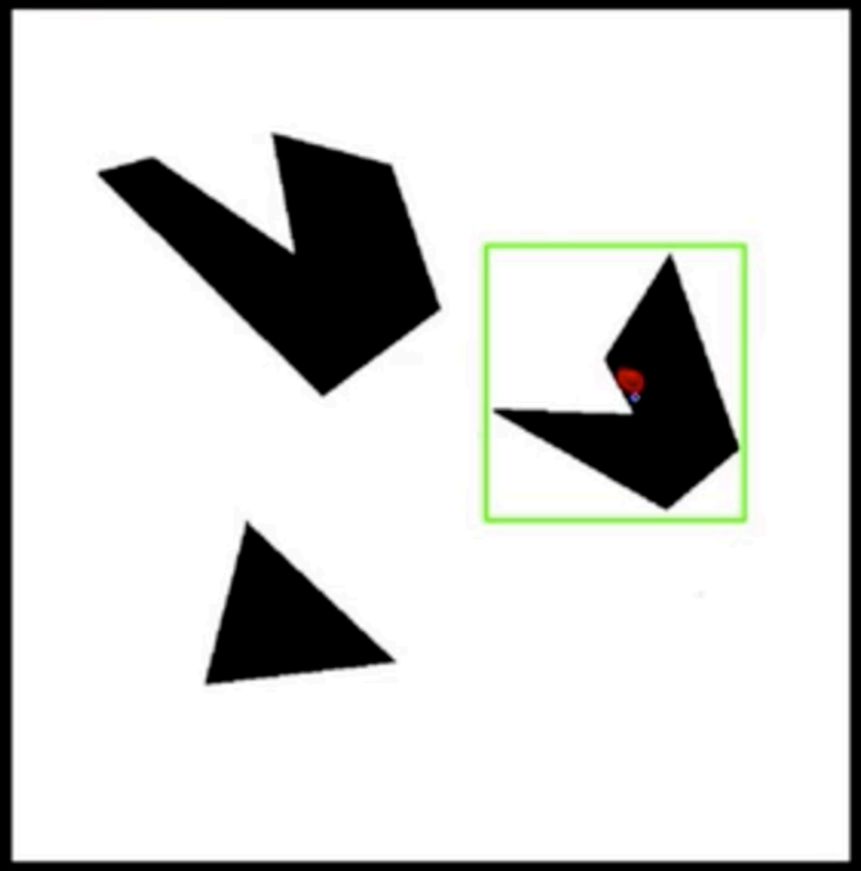


Detected template

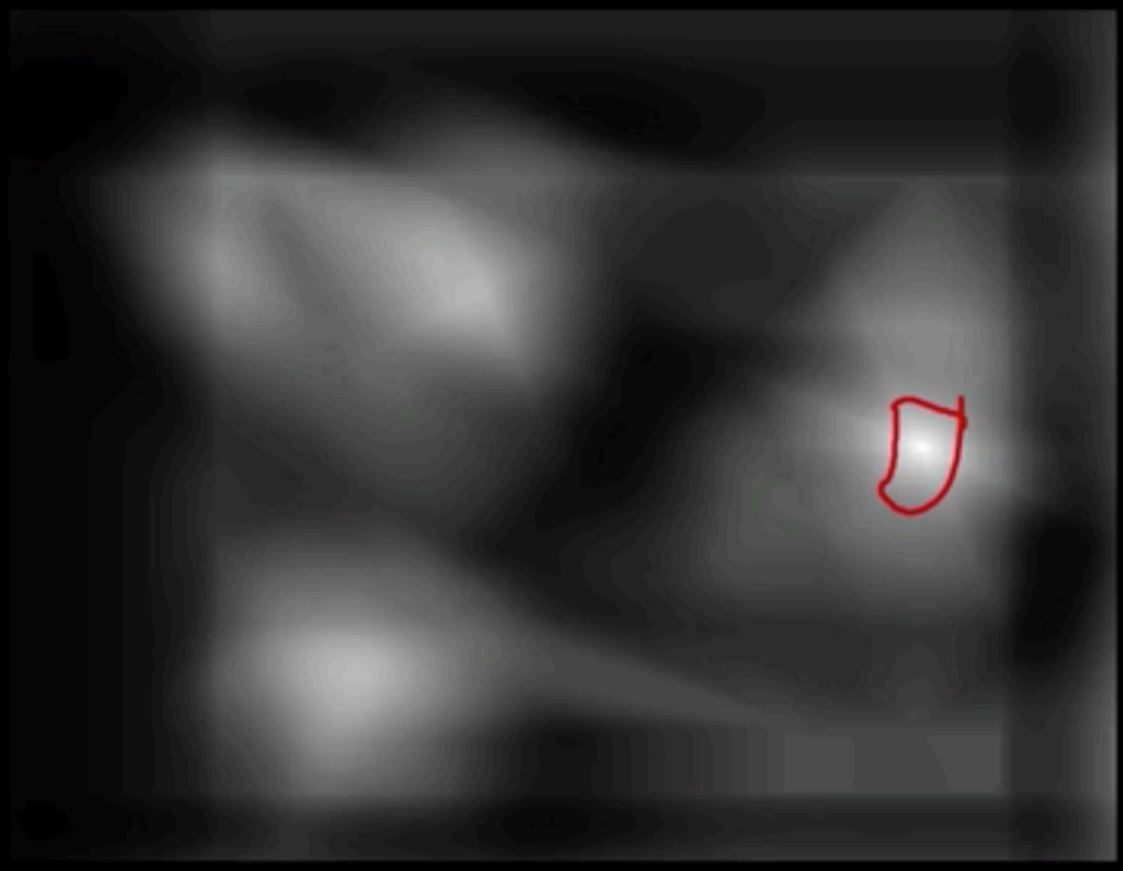


Correlation map

Template matching



Detected template



Correlation map

Template matching

What if the template is not identical to some sub-image in the scene?

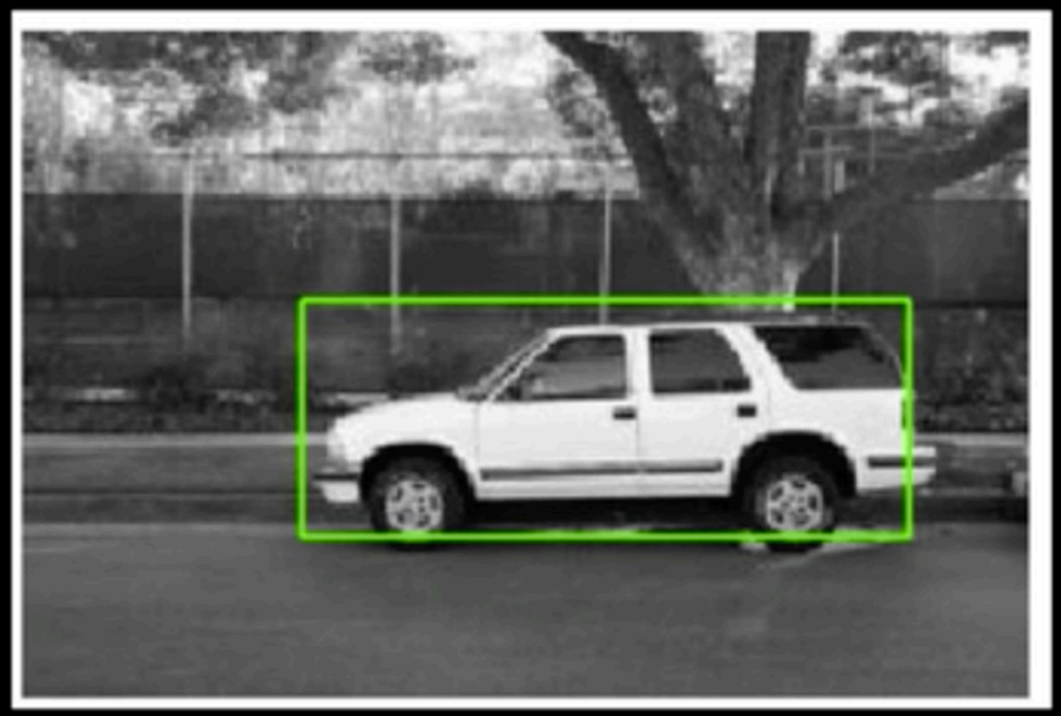
Scene

Template

K. Grauman

Template matching

Match can be meaningful, if scale, orientation, and general appearance is right.



Detected template

Template