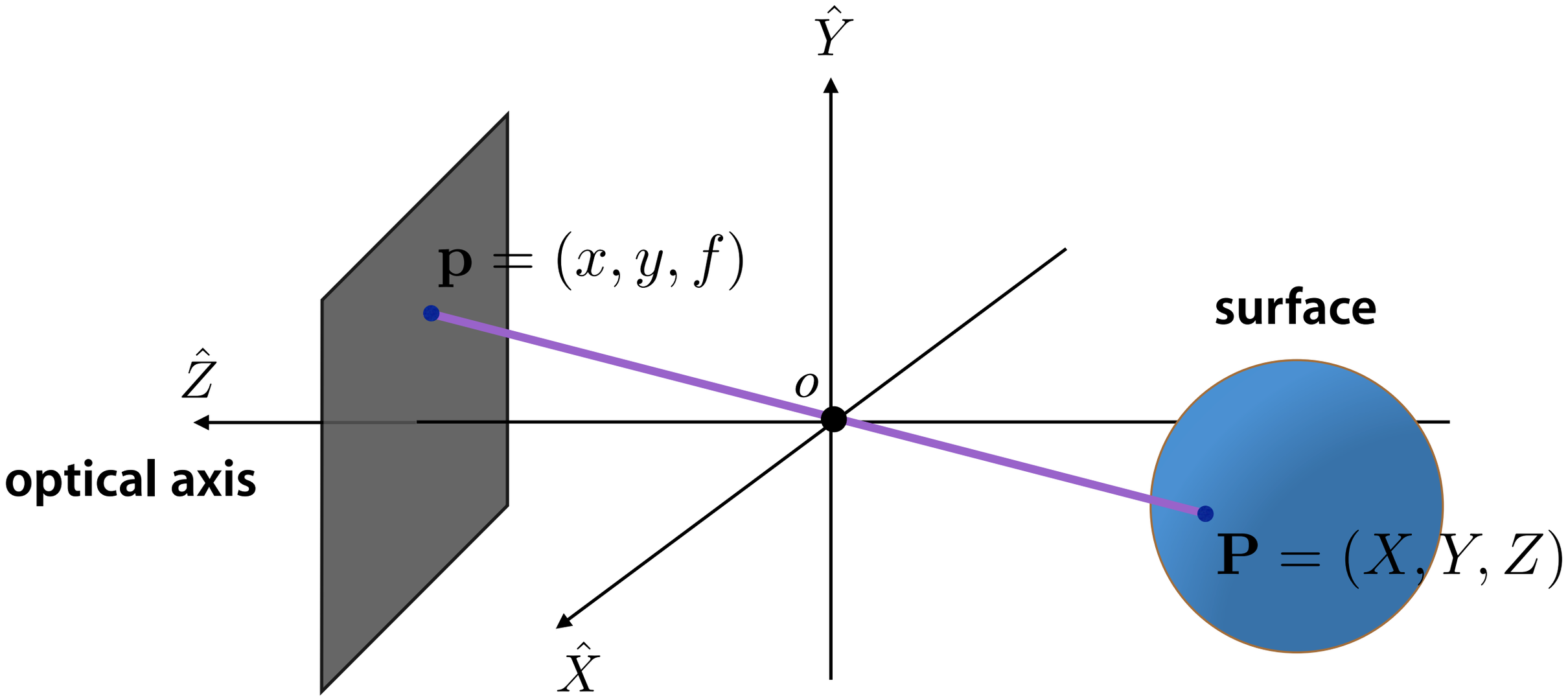
Intro to

Computer Vision

with Prof. Kosta Derpanis

Image filtering


$\hat{Y}$
$\mathbf{p} = (x, y, f)$
$\hat{Z}$
Z
o
optical axis
surface
$\mathbf{P} = (X, Y, Z)$
$\hat{X}$

images as functions

$$f : \mathbb{R}^2 \to \mathbb{R}$$

images as functions

$$f : \mathbb{R}^2 \to \mathbb{R}^3$$

images as functions

$$f : \mathbb{R}^2 \to \mathbb{R}^3$$

$$\begin{pmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{pmatrix}$$

filter

I

image

ϕ

$\phi(I)$

image

Image

NOISE

original

Gaussian noise

$$I(x,y) = I_{\text{ideal}}(x,y) + \eta(x,y)$$

where

$$\eta(x,y) \sim \mathcal{N}(\mu = 0, \sigma)$$

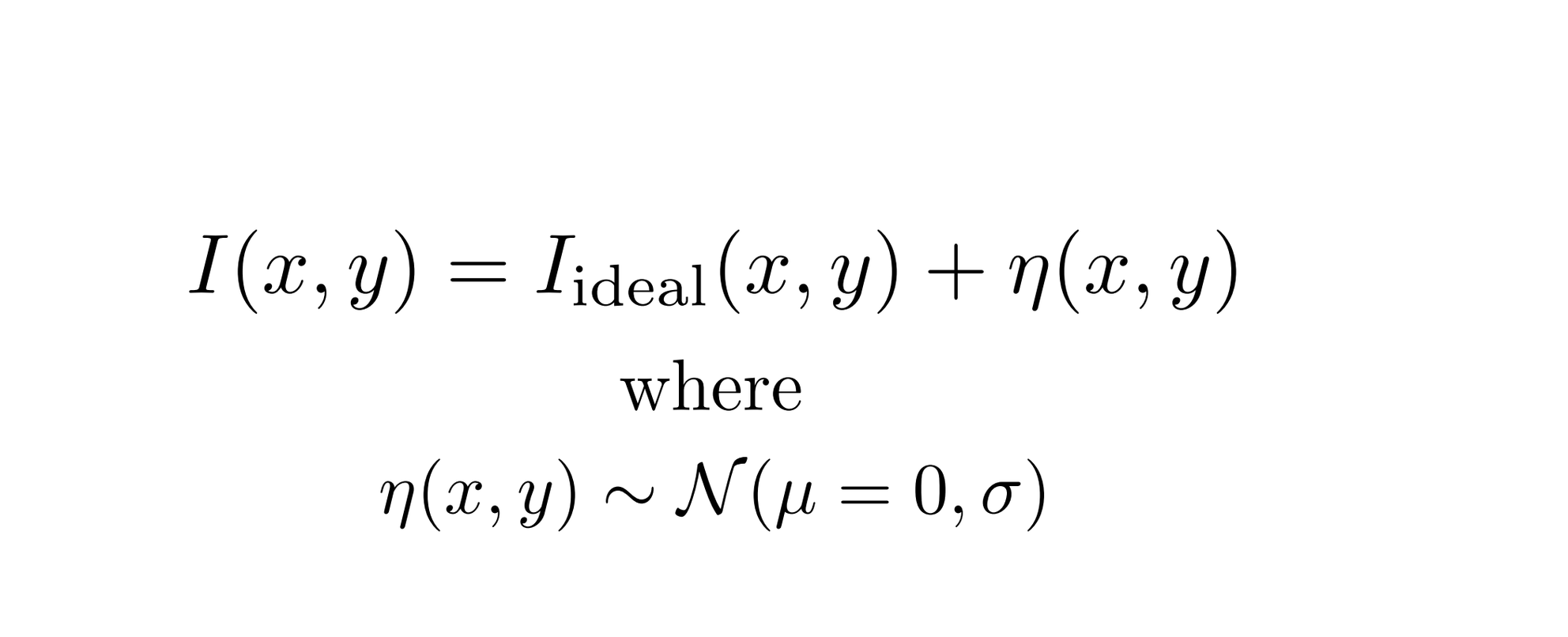
assumed independent and identically distributed

I.I.D.

$$I(x,y) = I_{\text{ideal}}(x,y) + \eta(x,y)$$

where

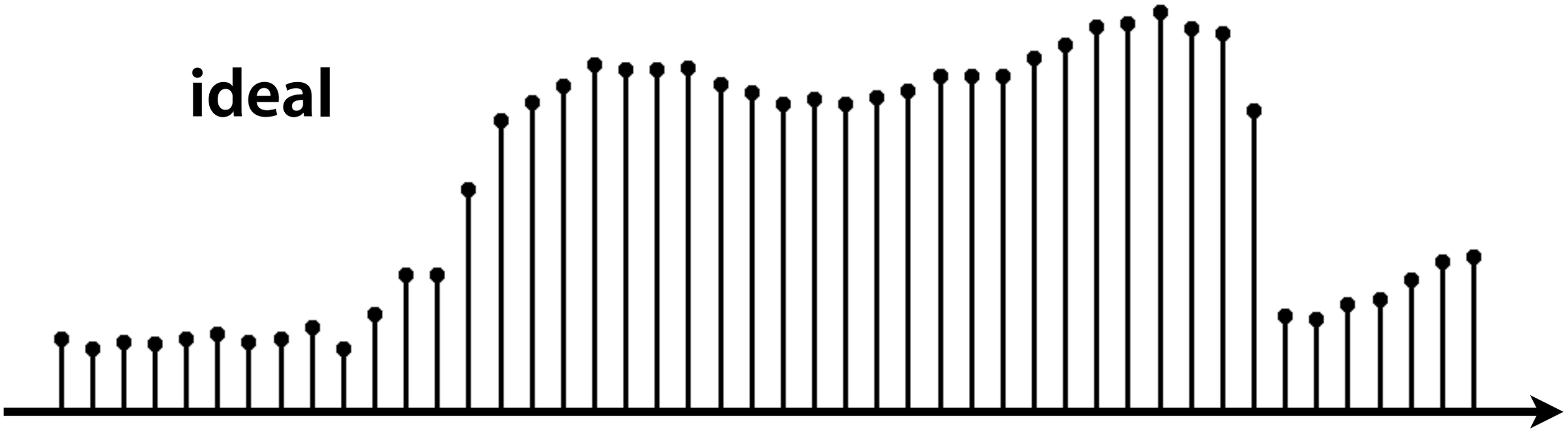
$$\eta(x,y) \sim \mathcal{N}(\mu = 0, \sigma)$$



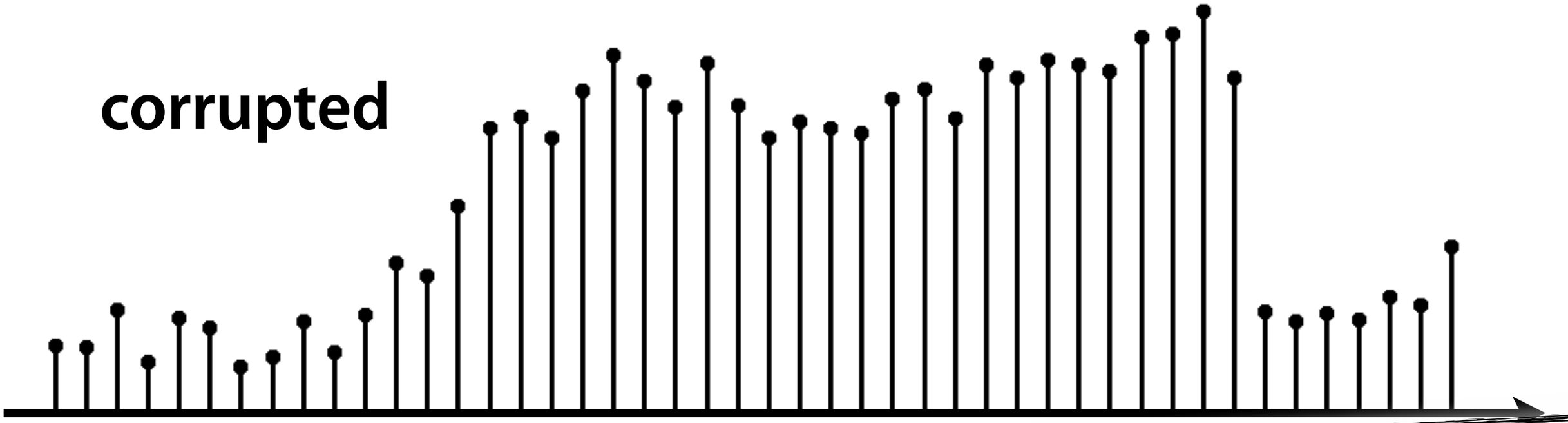
$$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

Gaussian noise

ideal

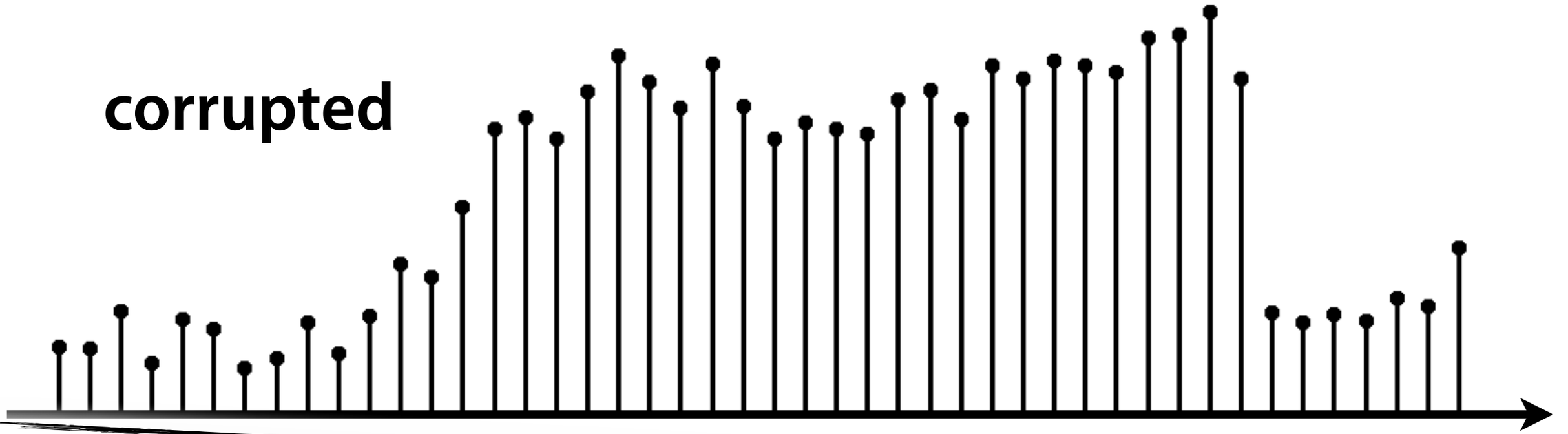


corrupted

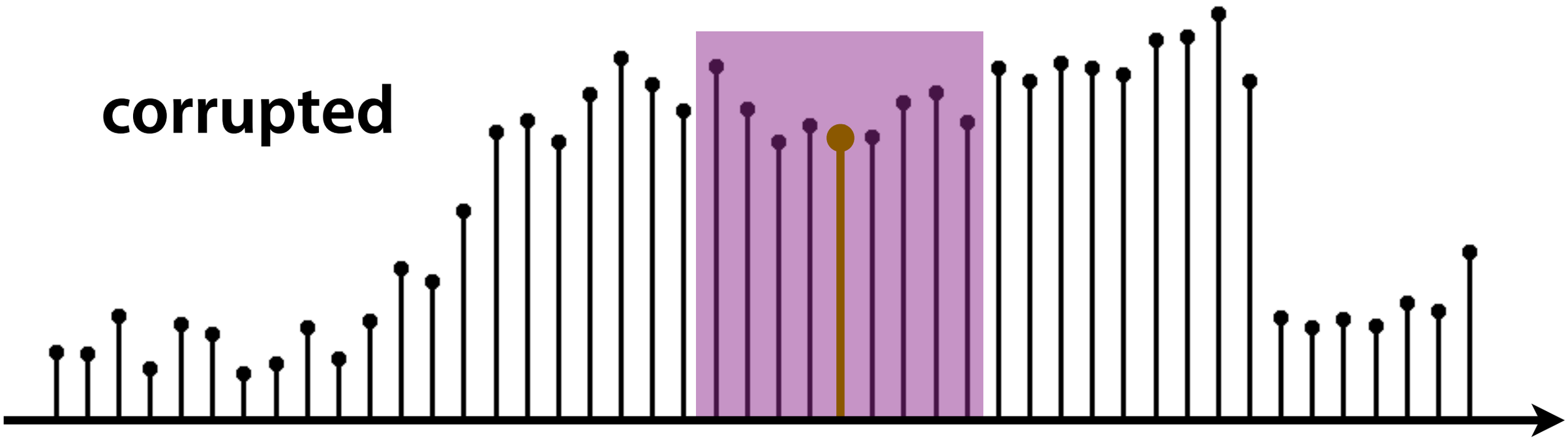


How can we remove the noise?

corrupted

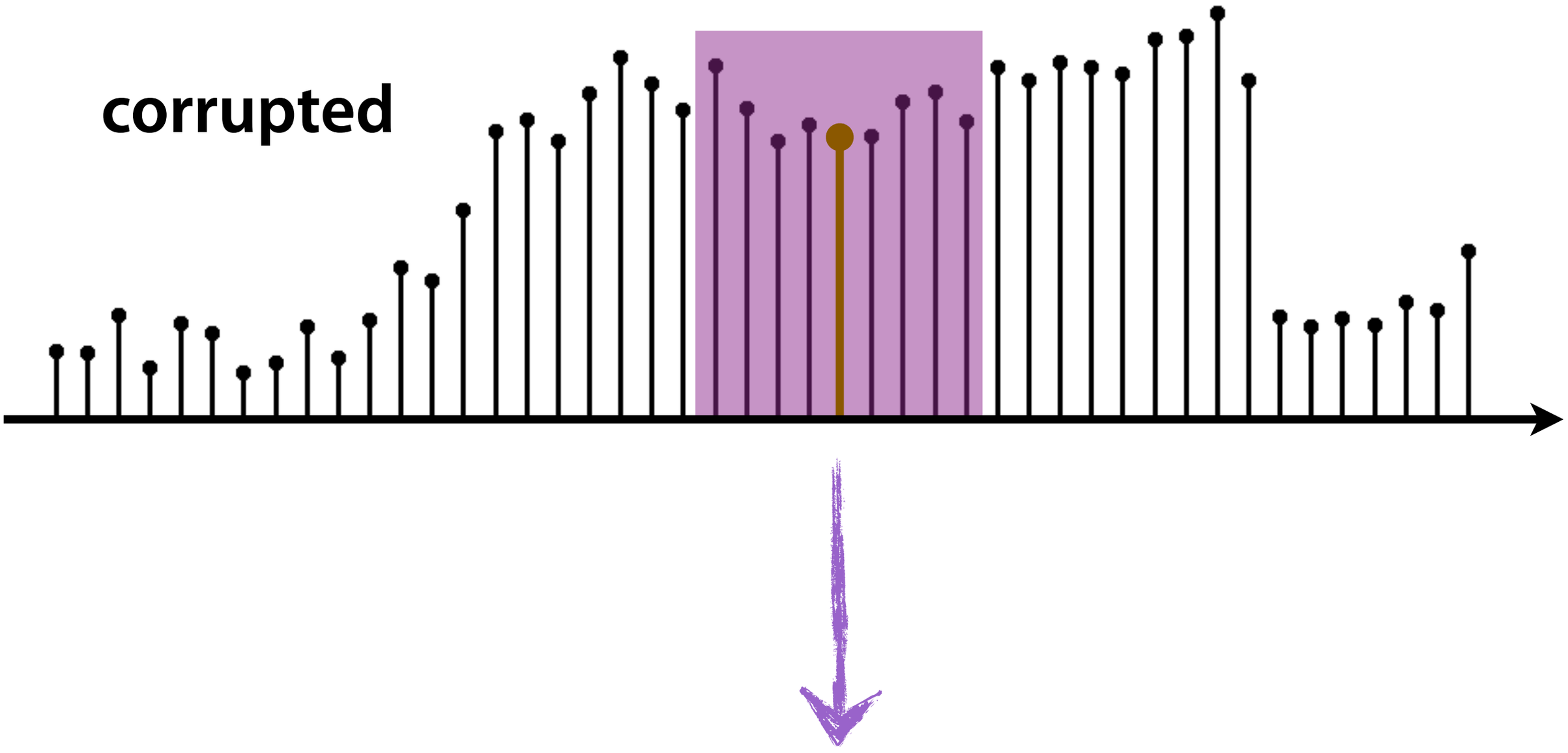


assume noise is I.I.D. and pixel neighbours are similar

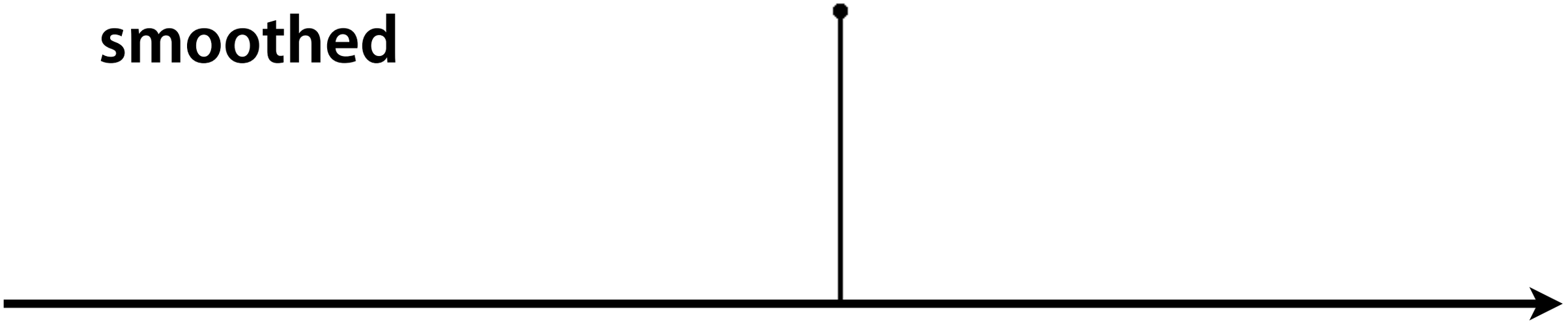


replace each pixel with an average of its neighours

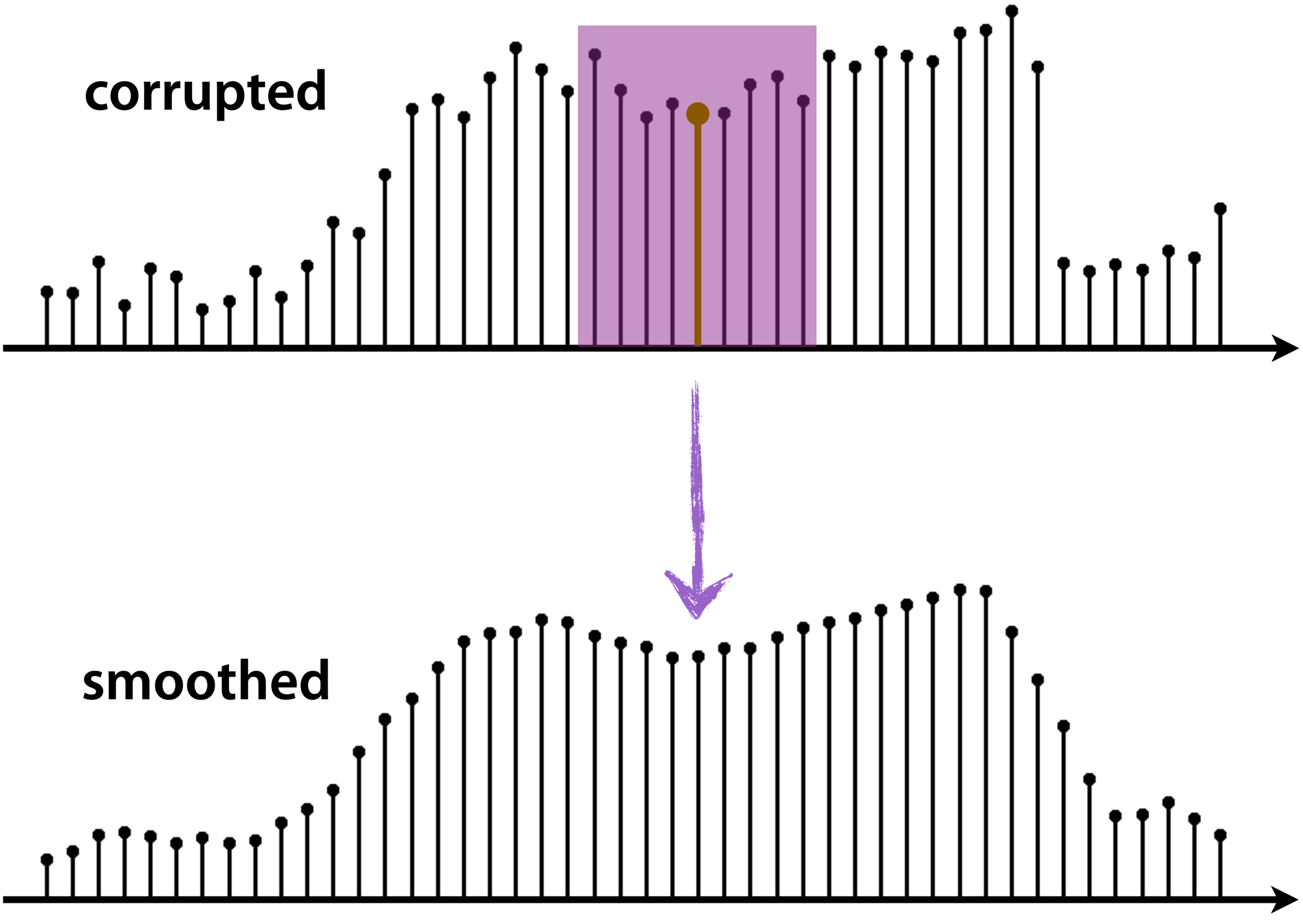
corrupted



smoothed

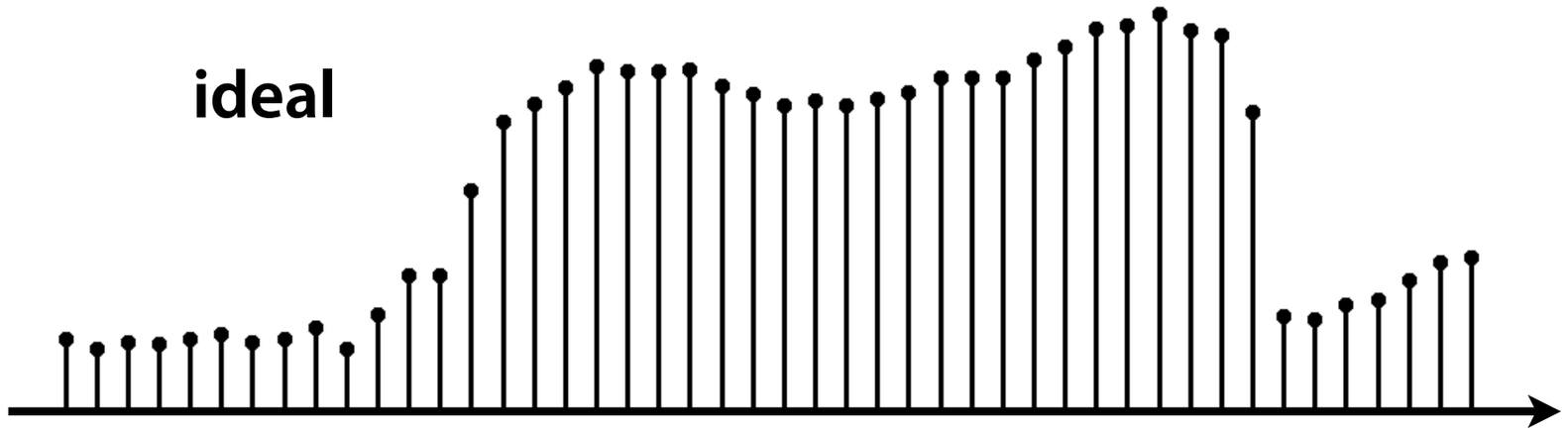


corrupted

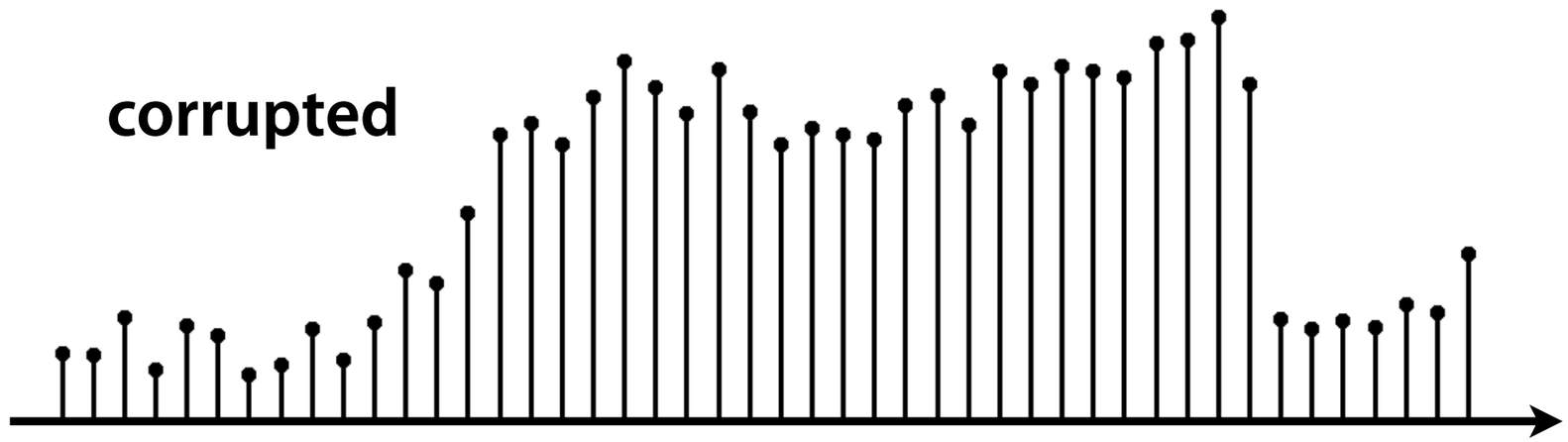


smoothed

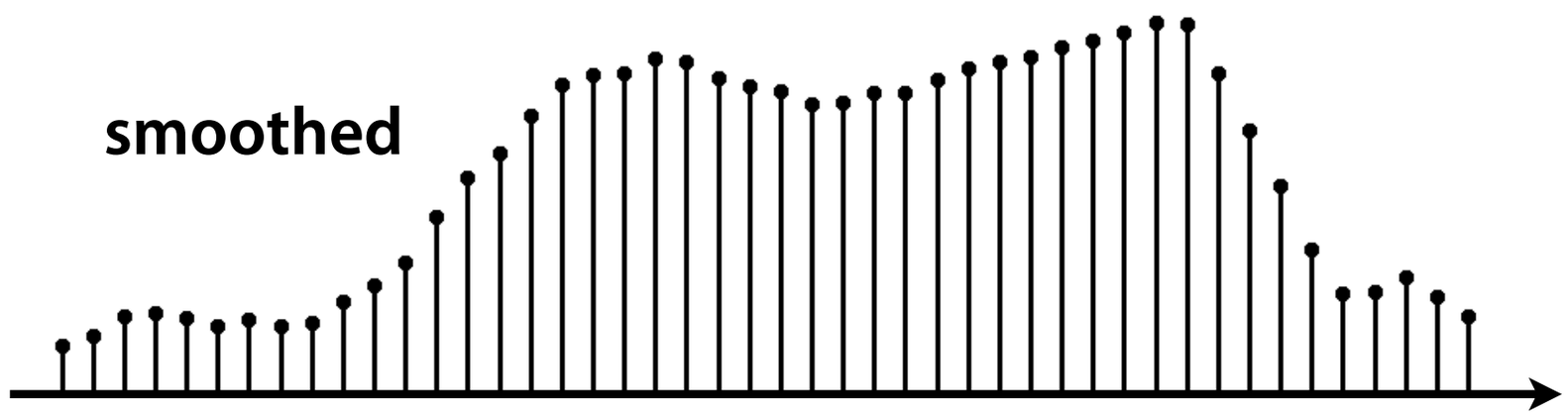
ideal



corrupted



smoothed



0	0	0	9	18
0	90	0	27	0
0	0	0	9	0
0	0	0	9	0
0	0	0	0	18

input

	10			

output

0	0	0	9	18
0	90	0	27	0
0	0	0	9	0
0	0	0	9	0
0	0	0	0	18

input

	10	15		

output

0	0	0	9	18
0	90	0	27	0
0	0	0	9	0
0	0	0	9	0
0	0	0	0	18

input

	10	15	7	

output

0	0	0	9	18
0	90	0	27	0
0	0	0	9	0
0	0	0	9	0
0	0	0	0	18

input

	10	15	7	
	10	15	5	
	0	2	4	

output

$$(2K+1) \times (2K+1)$$

averaging window size

$$H[x,y] = \frac{1}{(2K+1)^2} \sum_{u=-K}^{K} \sum_{v=-K}^{K} F[x+u, y+v]$$

output

$$H[x,y] = \frac{1}{(2K+1)^2} \sum_{u=-K}^{K} \sum_{v=-K}^{K} F[x+u, y+v]$$

input

$$H[x,y] = \frac{1}{(2K+1)^2} \sum_{u=-K}^{K} \sum_{v=-K}^{K} F[x+u, y+v]$$

loop over pixels in neighbourhood

$$H[x,y] = \frac{1}{(2K+1)^2} \sum_{u=-K}^{K} \sum_{v=-K}^{K} F[x+u, y+v]$$

weight

$$H[x,y] = \frac{1}{(2K+1)^2} \sum_{u=-K}^{K} \sum_{v=-K}^{K} F[x+u, y+v]$$

How can the weights vary per neighbourhood position?

mask, kernel or filter

$$H[x,y] = \frac{1}{(2K+1)^2} \sum_{u=-K}^{K} \sum_{v=-K}^{K} F[x+u, y+v] G[u,v]$$

correlation

$$H[x,y] = \frac{1}{(2K+1)^2} \sum_{u=-K}^{K} \sum_{v=-K}^{K} F[x+u, y+v] G[u,v]$$

replace each pixel with a linear combination of neighbours

$$H = G \otimes F$$

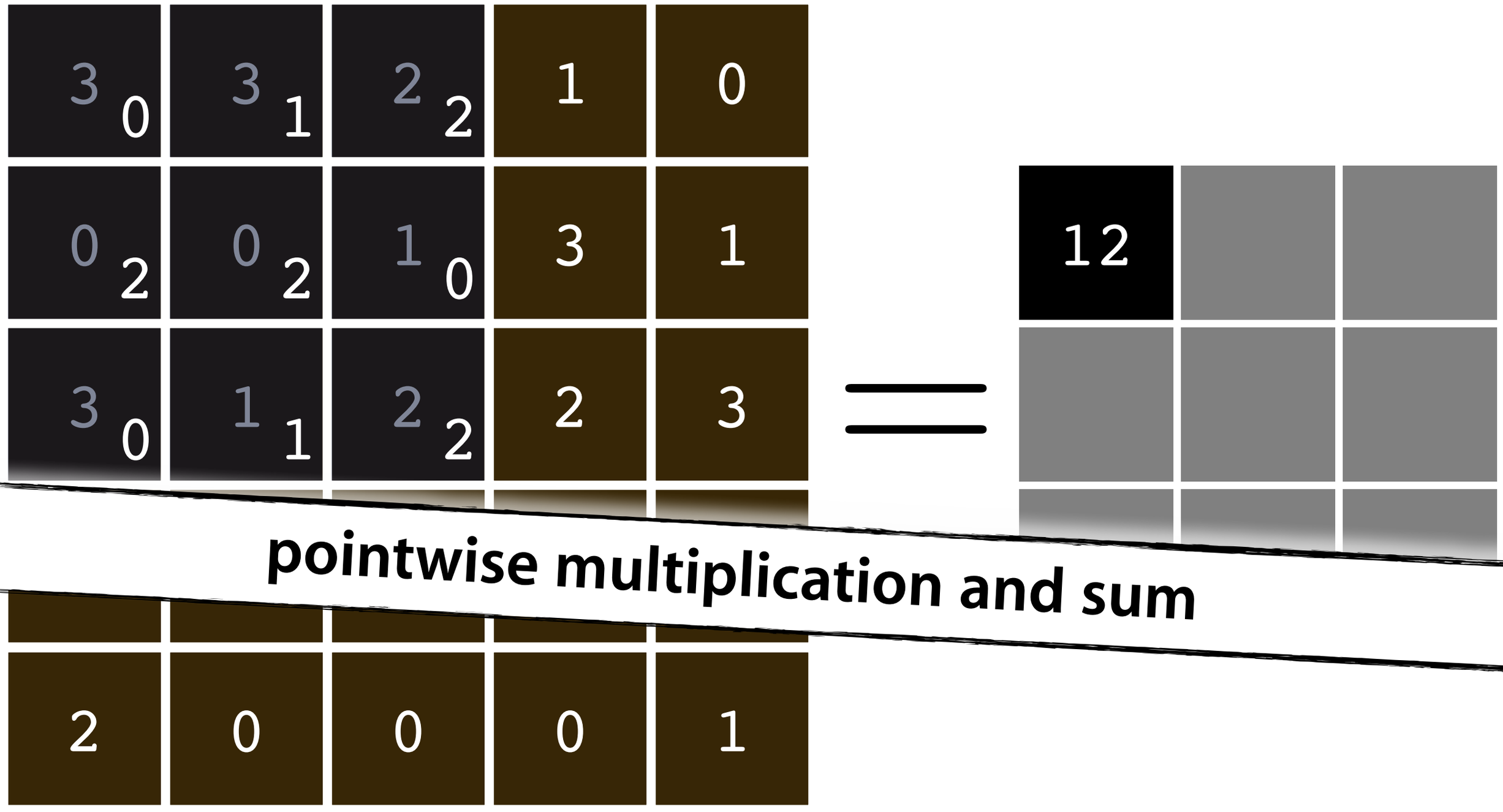
cross correlation

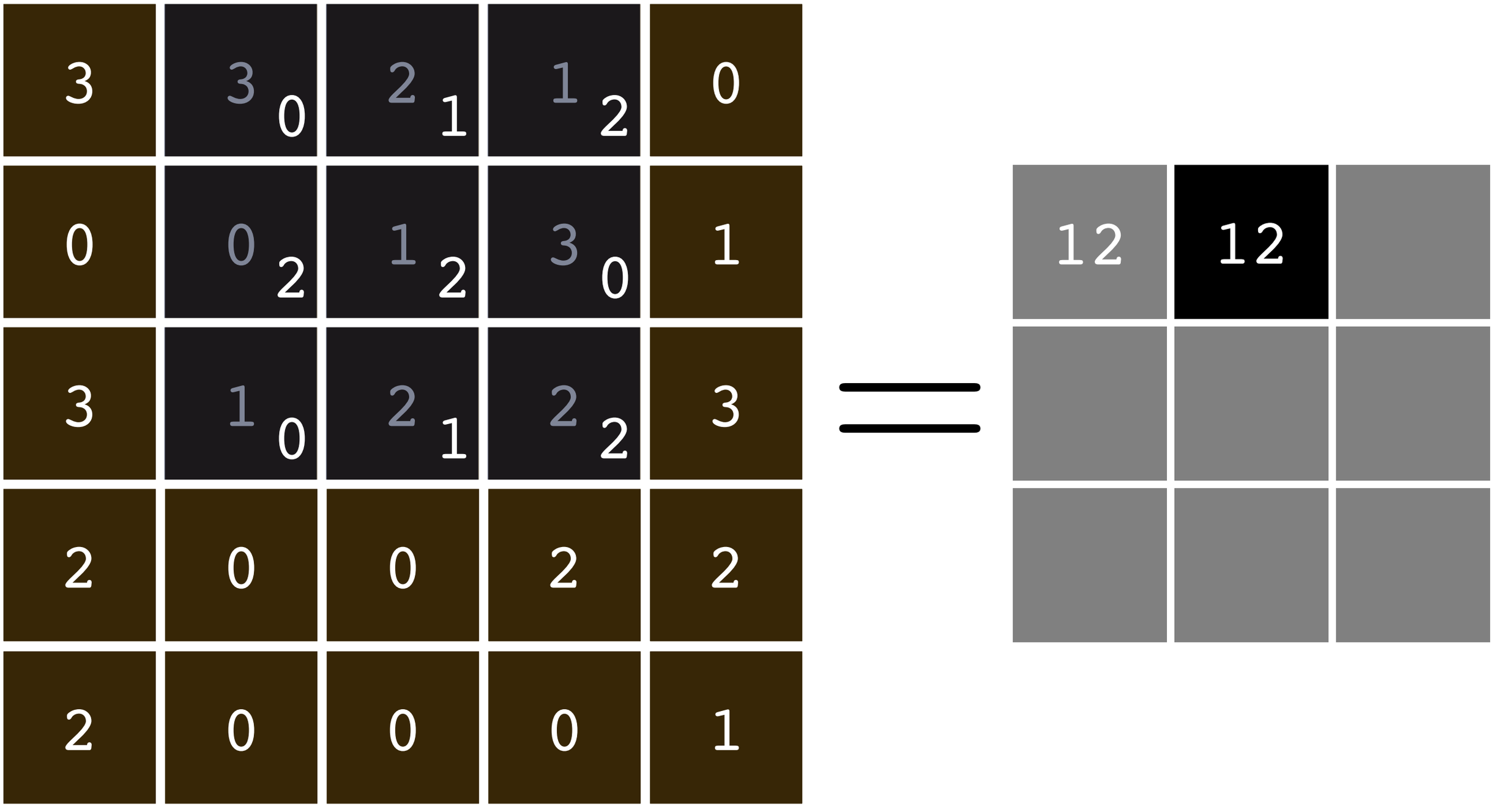
correlation

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

*

0	1	2
2	2	0
0	1	2

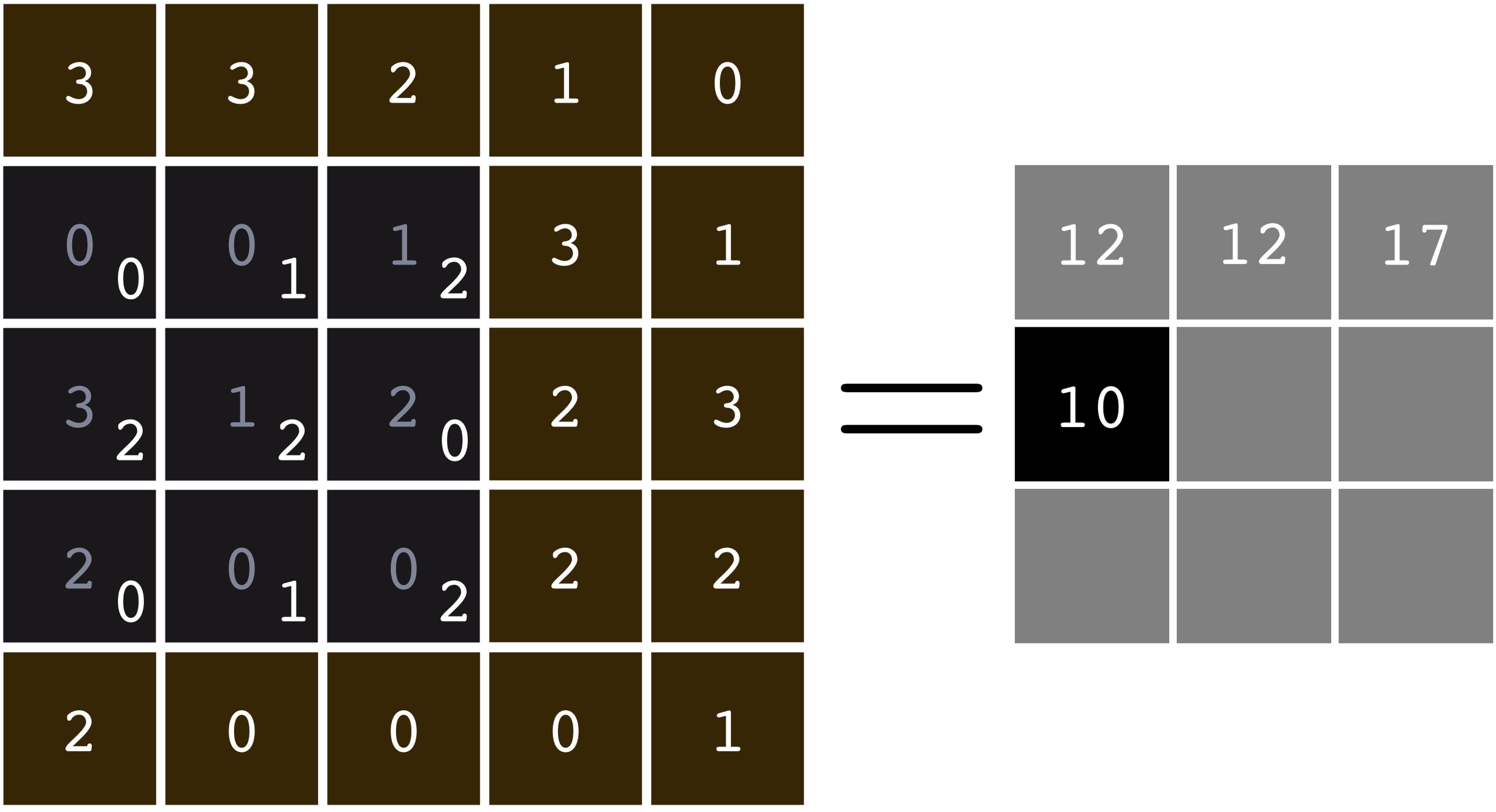

3 0
3 1
2 2
1
0
0 2
0 2
1 0
3
1
3 0
1 1
2 2
2
3
2
0
0
0
1
12
pointwise multiplication and sum


3
3 0
2 1
1 2
0
0
0 2
1 2
3 0
1
3
1 0
2 1
2 2
3
2
0
0
2
2
2
0
0
0
1
12
12

3	3	2 0	1 1	0 2
0	0	1 2	3 2	1 0
3	1	2 0	2 1	3 2
2	0	0	2	2
2	0	0	0	1

=

12	12	17


3 3 2 1 0
0 0 0 1 1 2 3 1
3 2 1 2 2 0 2 3
2 0 0 1 0 2 2 2
2 0 0 0 1
12 12 17
10

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

12	12	17
10	17	19
9	6	

correlation

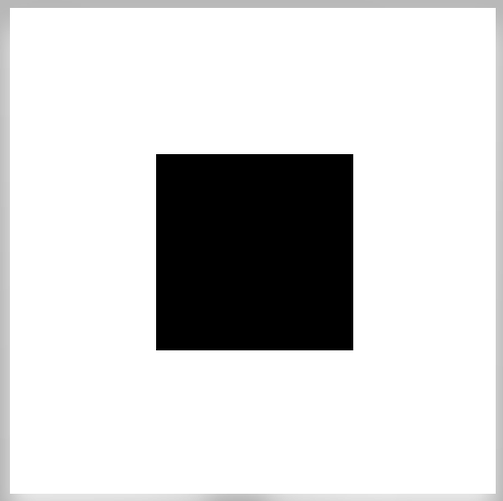
3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

$$\begin{matrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{matrix}$$

=

12	12	17
10	17	19
9	6	14


original



box filter

smoothing by averaging

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$G[x, y]$

$\otimes$

0	0	0	9	18
0	90	0	27	0
0	0	0	9	0
0	0	0	9	0
0	0	0	0	18

$F[x, y]$

$$\frac{1}{16}$$

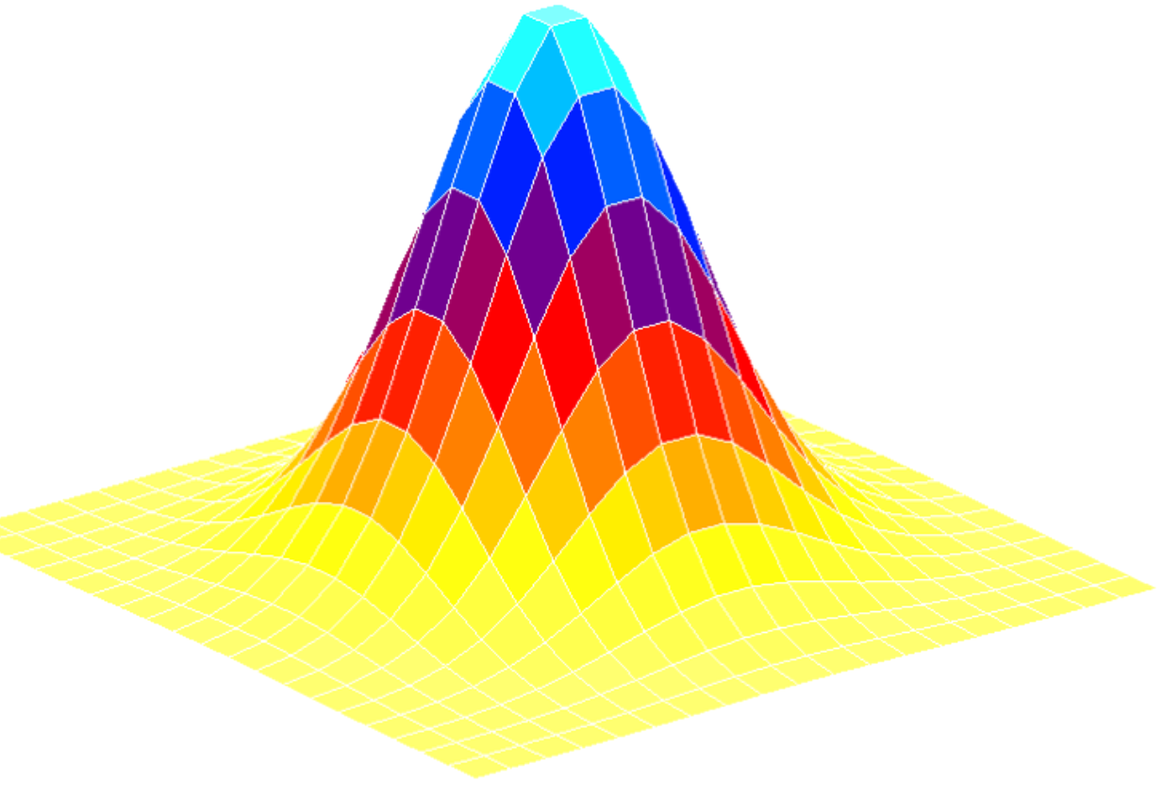
1	2	1
2	4	2
1	2	1

$G[x, y]$

$\otimes$

0	0	0	9	18
0	90	0	27	0
0	0	0	9	0
0	0	0	9	0
0	0	0	0	18

$F[x, y]$



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

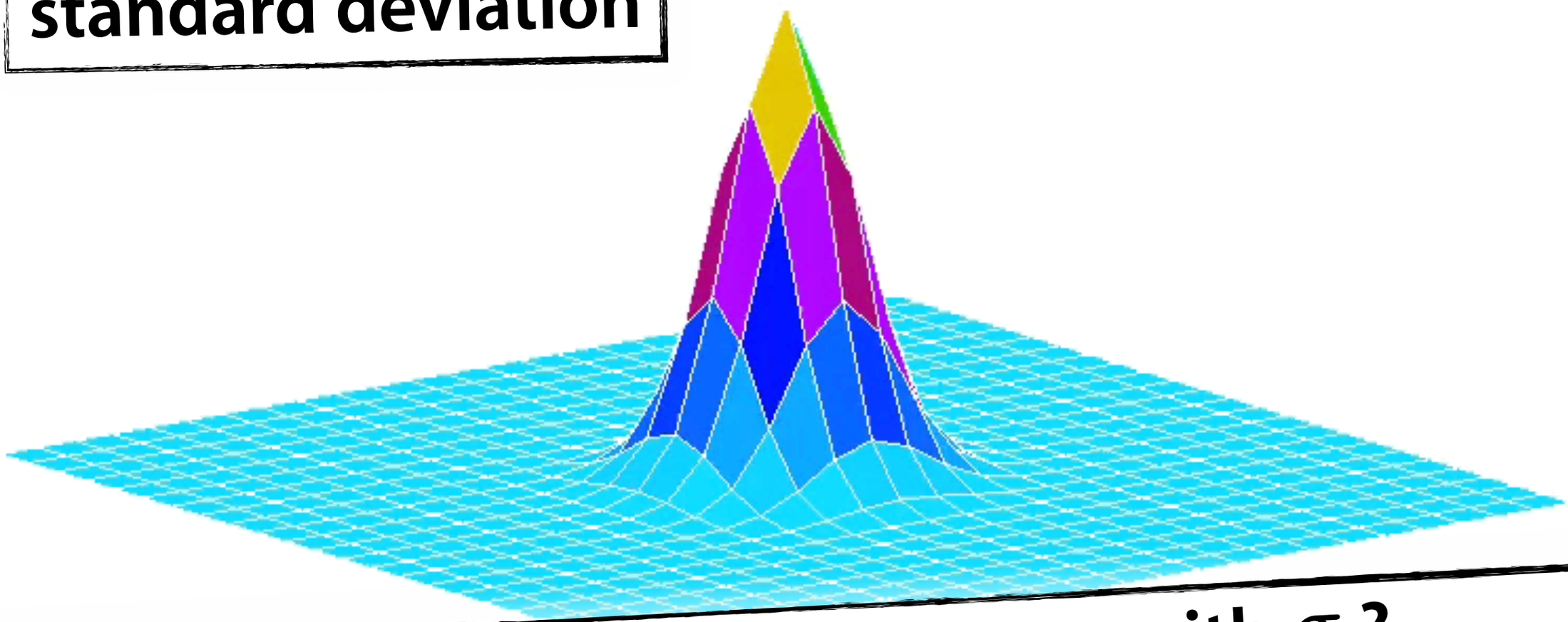
$$\approx$$

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

$$G[x,y]$$

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

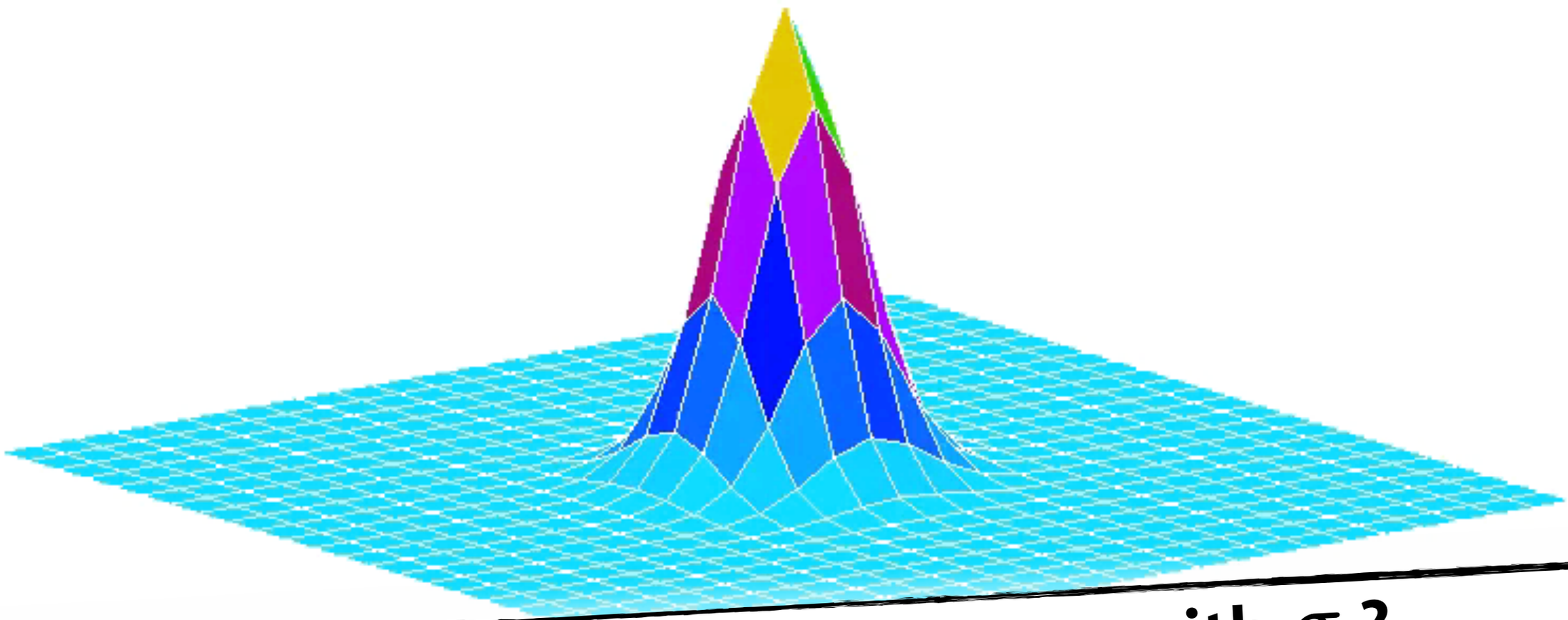
standard deviation



How does the Gaussian vary with σ ?

normalization

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

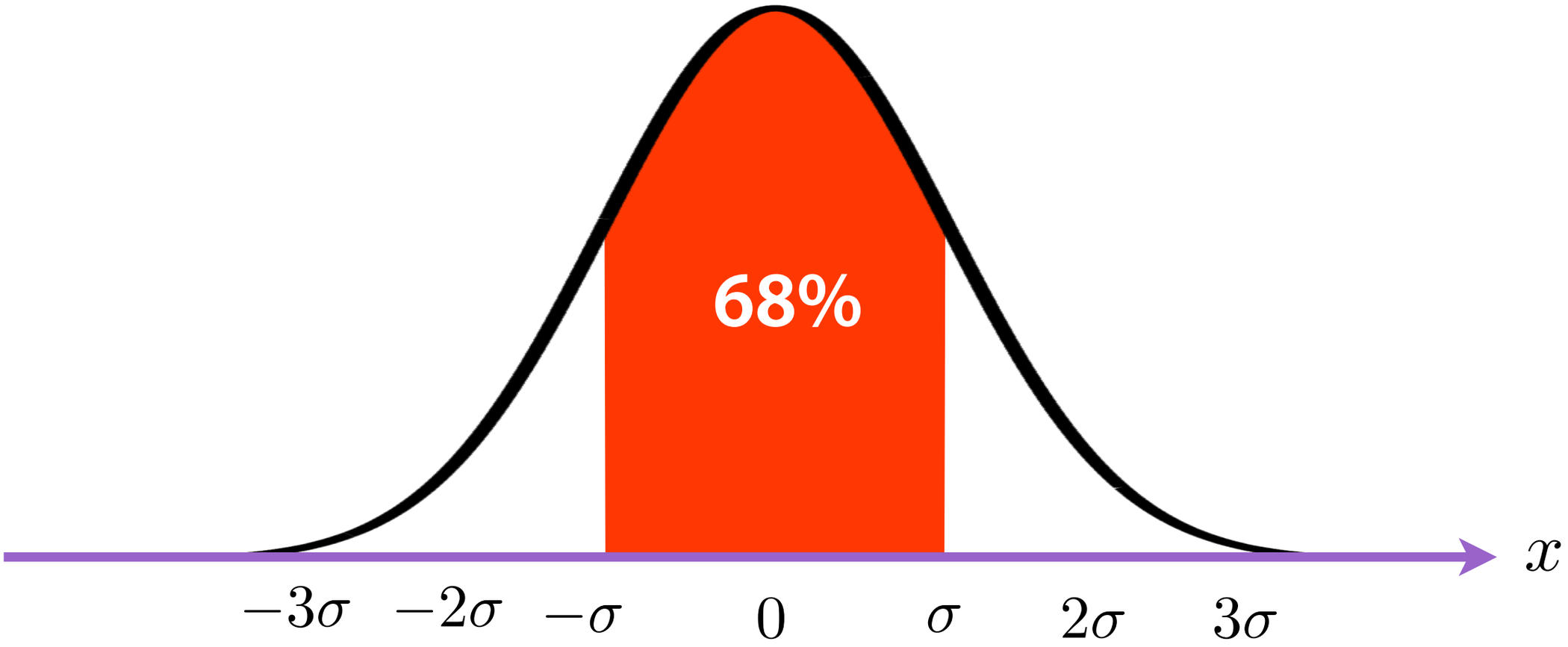


How does the Gaussian vary with σ ?

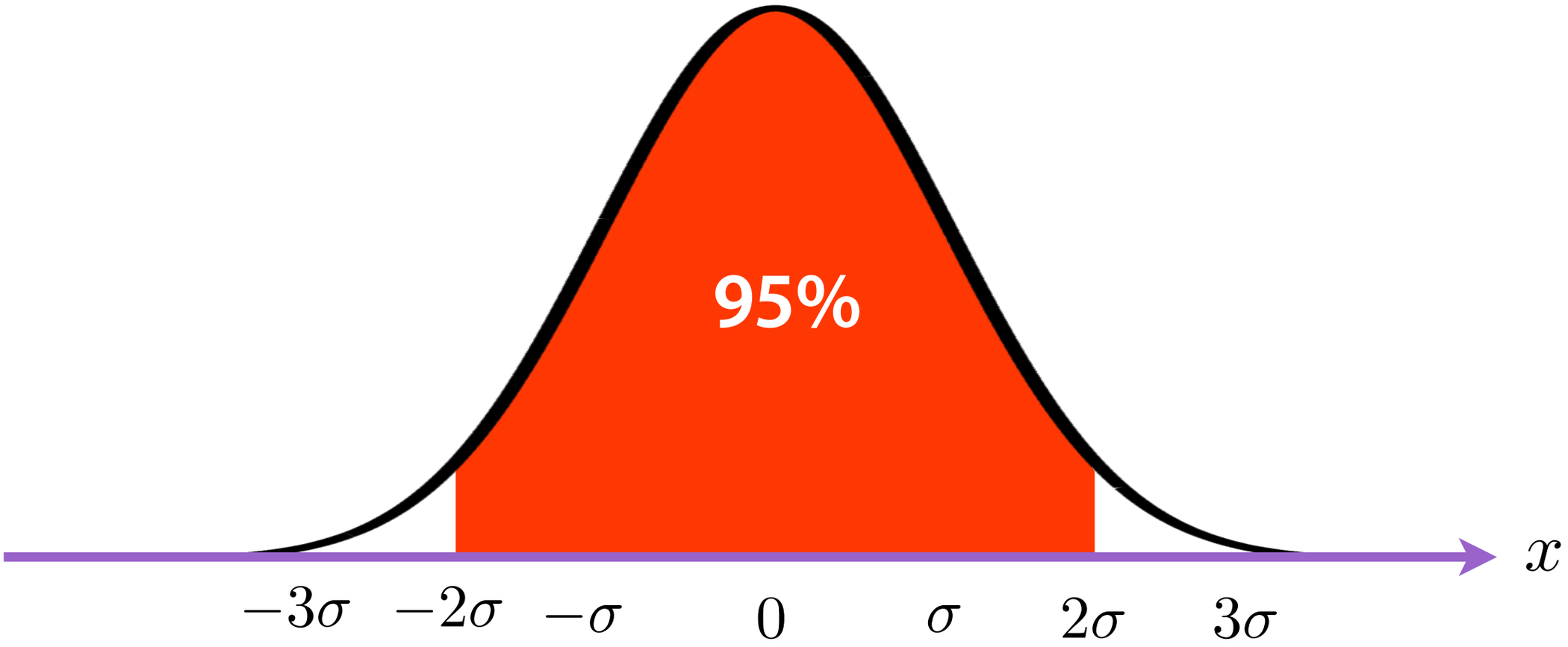
Gaussian filters have infinite support

Discrete filters use finite kernels

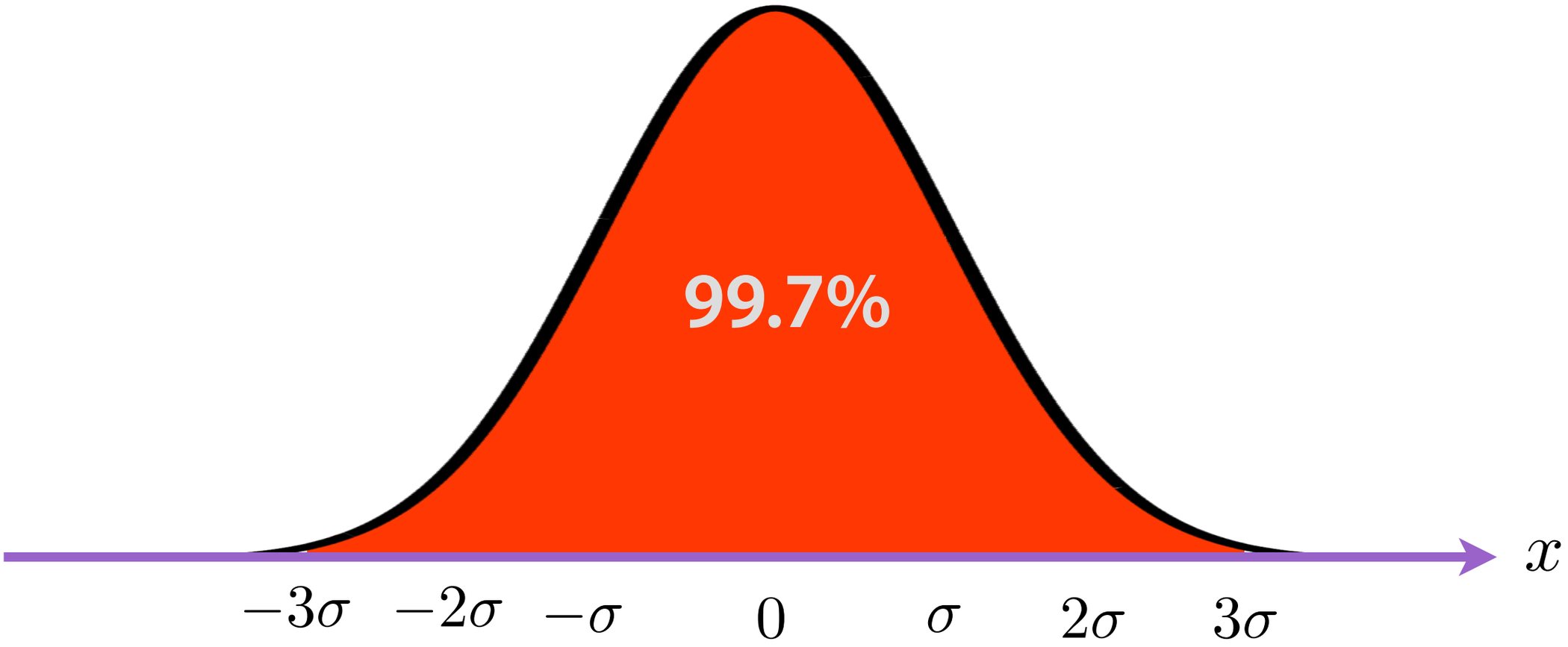
three-sigma rule

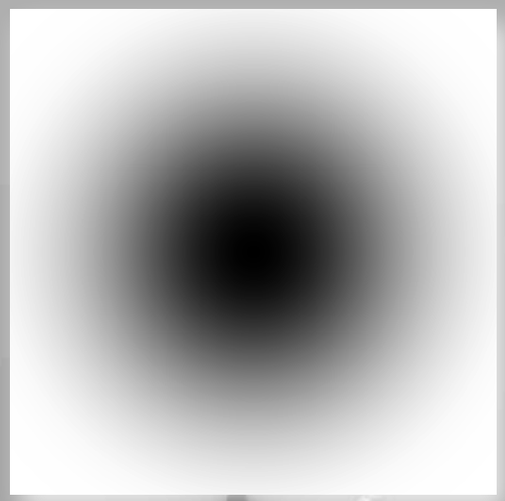


three-sigma rule



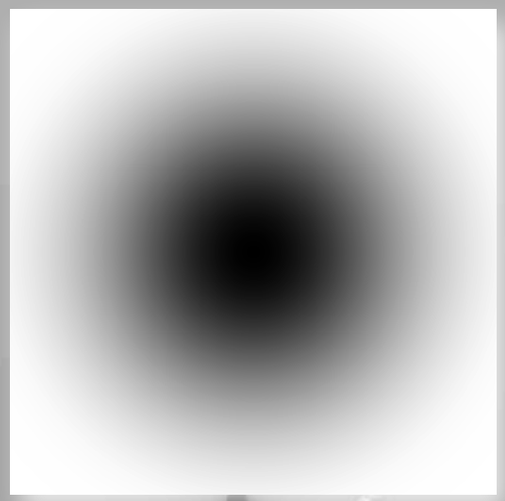
three-sigma rule





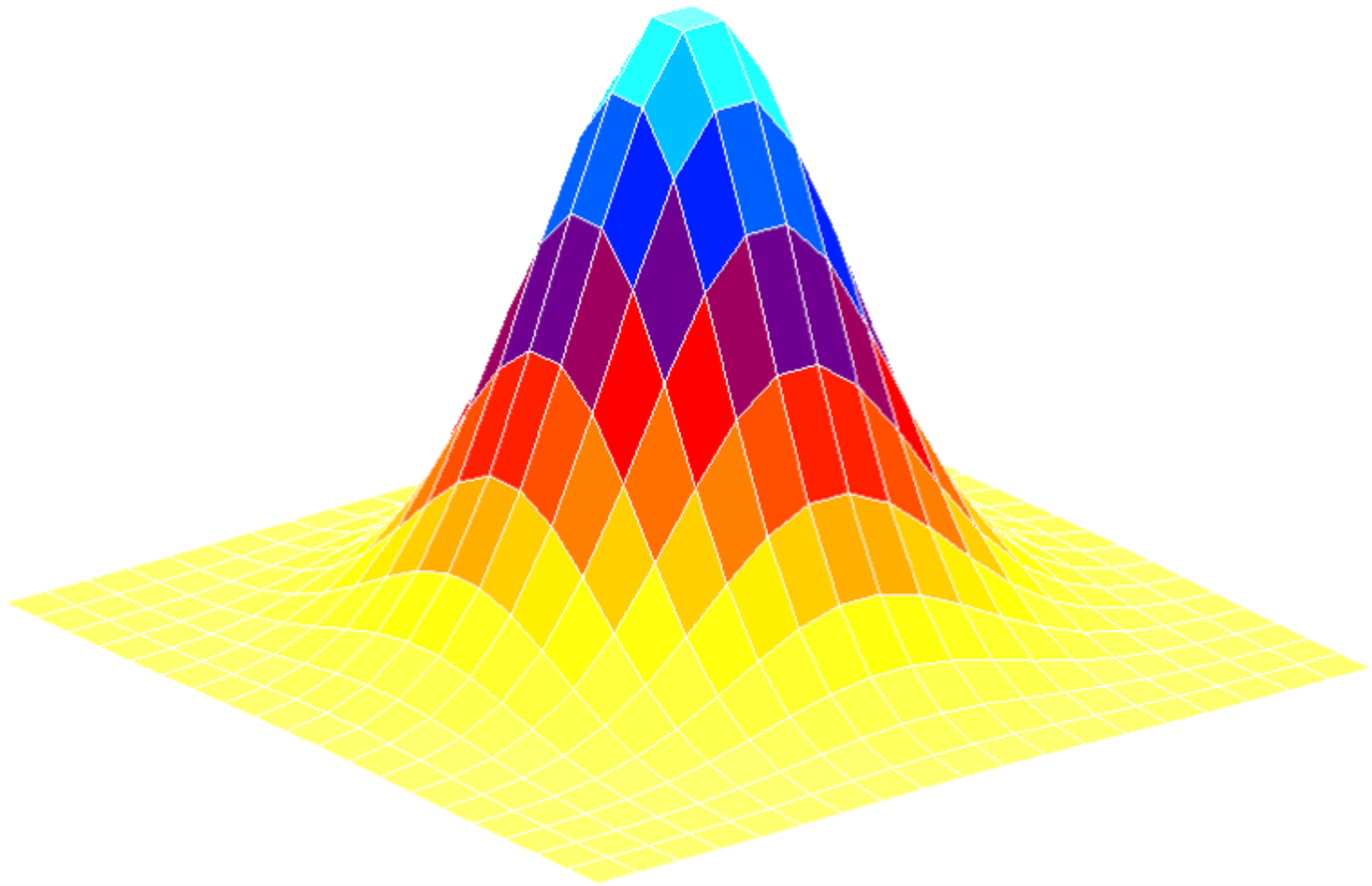
Gaussian filter

original

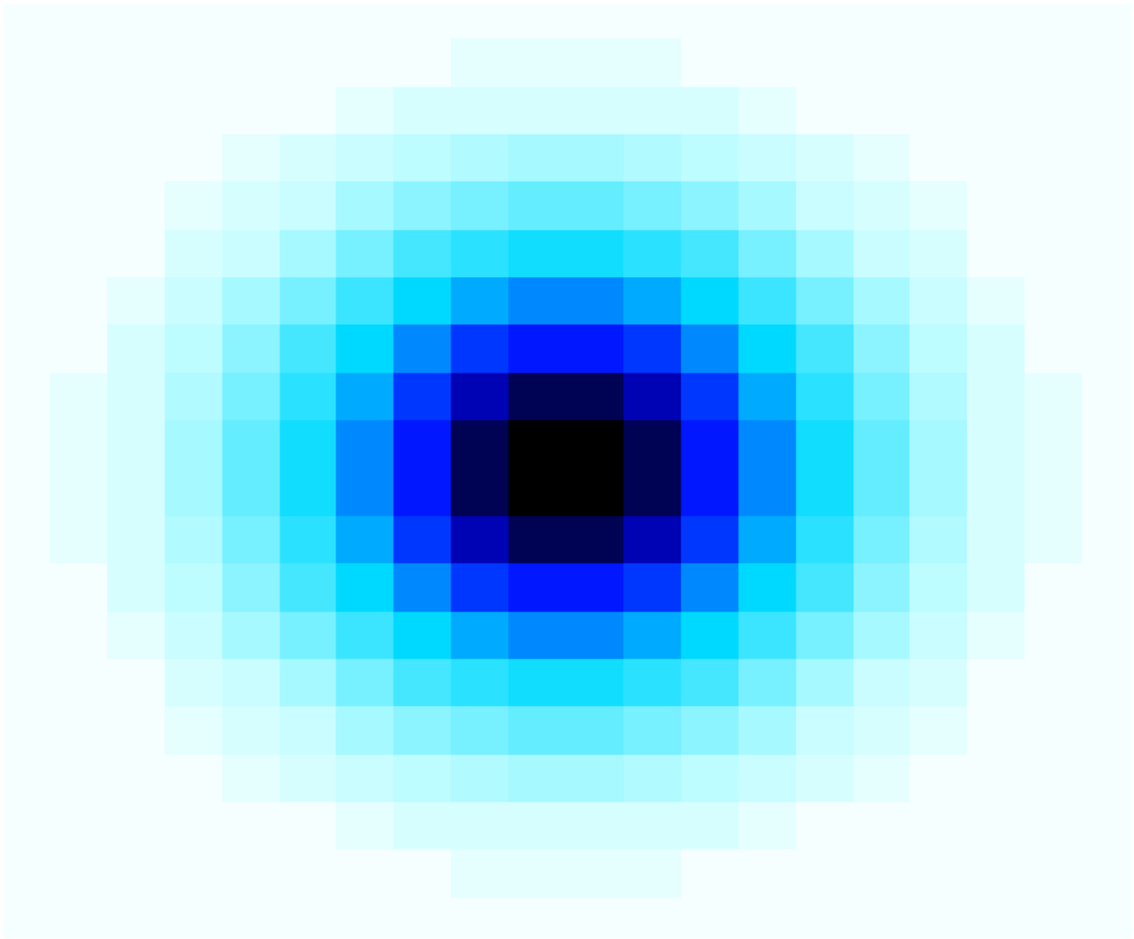


Gaussian filter

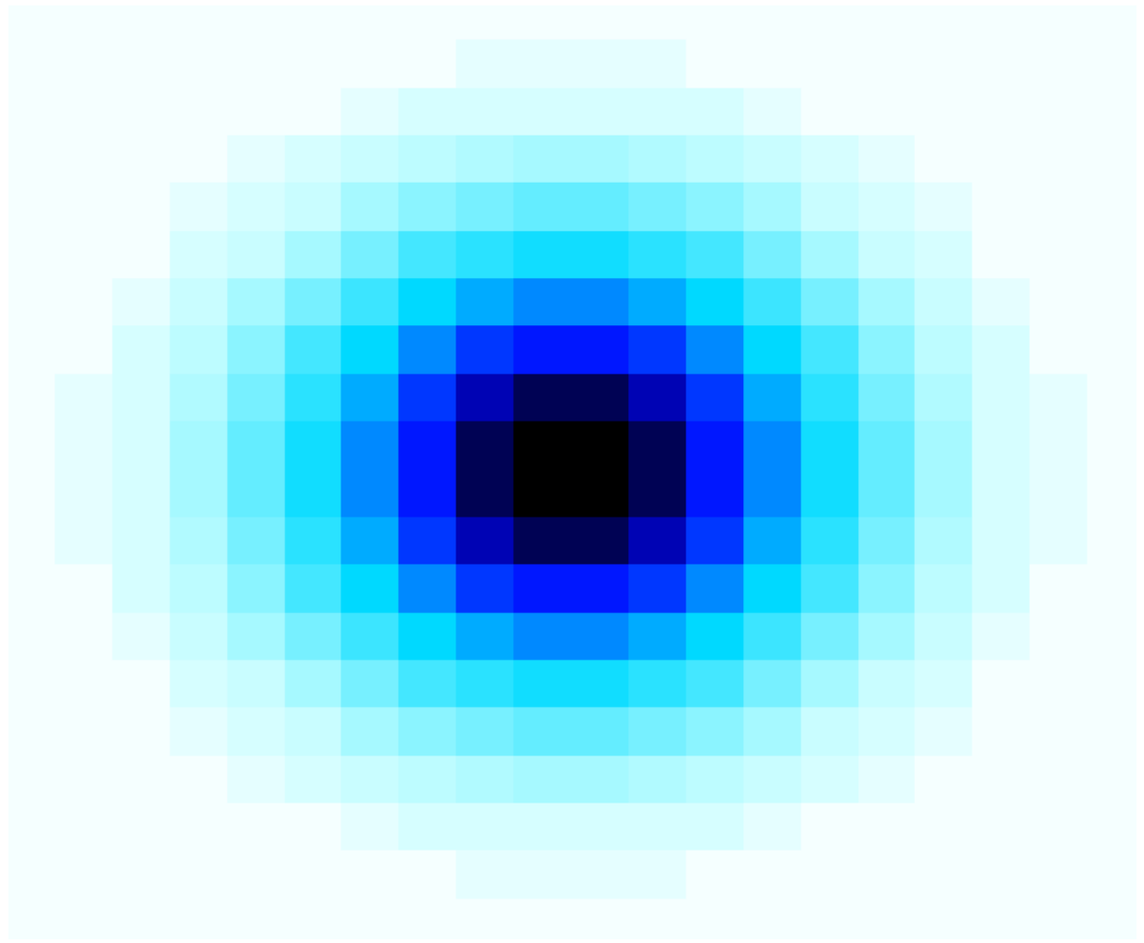
Gaussian blur



```
>> width = 19;
>> sigma = 3;
>> h = fspecial('gaussian', width, sigma);
>> surf(h)
>> set(gca, 'Color', [0 0 0]);  % optional
```

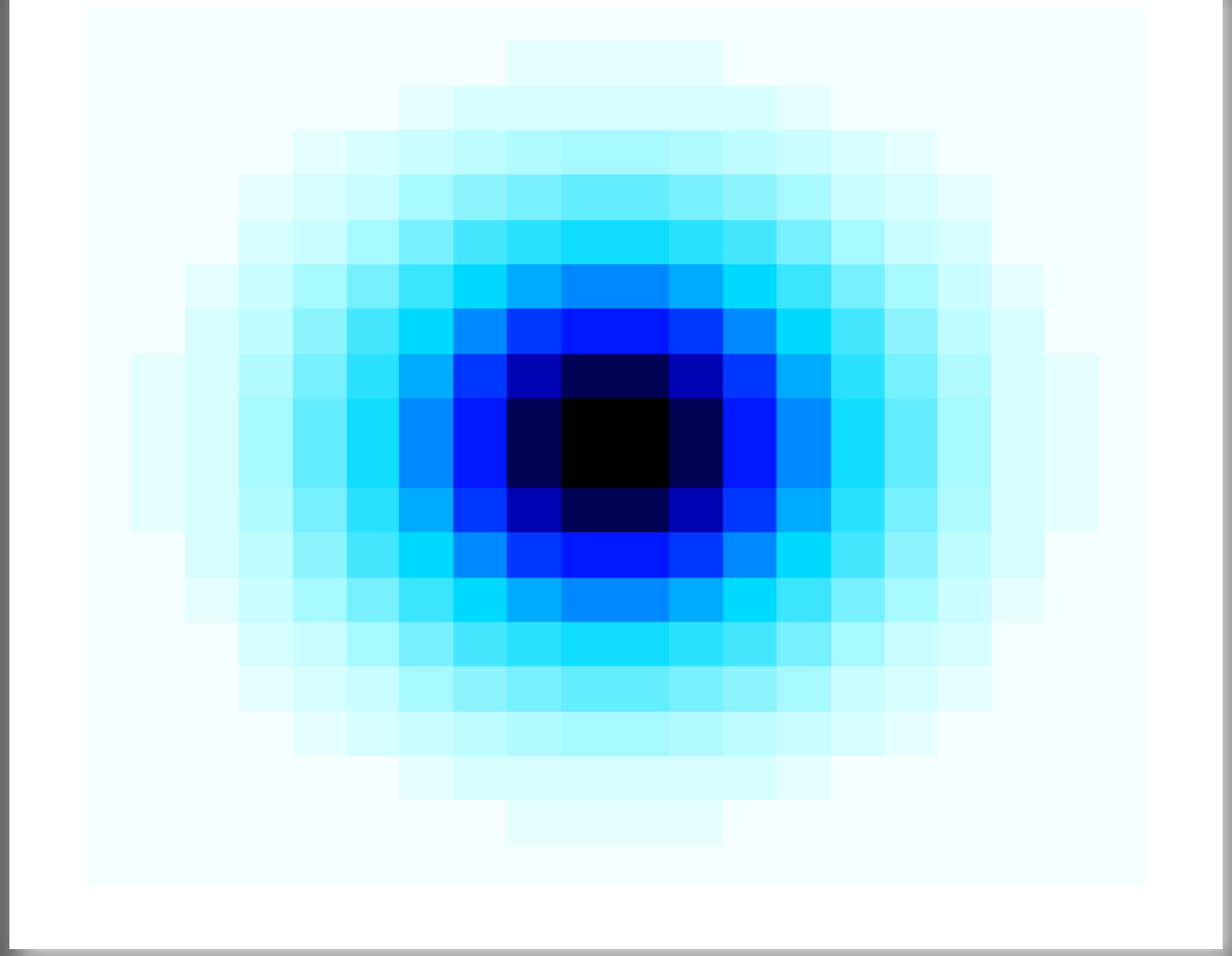


```
>> width = 19;
>> sigma = 3;
>> h = fspecial('gaussian', width, sigma);
>> imagesc(h);
>> colormap hot   % optional
```



```
>> width = 19;
>> sigma = 3;
>> h = fspecial('gaussian', width, sigma);
>> imagesc(h);
```

display image with scaled colours



```
>> width = 19;
>> sigma = 3;
>> h = fspecial('gaussian', width, sigma);
>> imagesc(h);
>> colormap hot    % optional
```

```
>> width = 21;
>> sigma = 3;
>> h = fspecial('gaussian', width, sigma);
>> I = imfilter(im, h, 'symmetric', 'same');
>> imshow(I, []);
```

```
>> width = 21;
>> sigma = 3;
>> h = fspecial('gaussian', width, sigma);
>> I = imfilter(im, h, 'symmetric', 'same');
>> imshow(I, []);
```

```
>> width = 21;
>> sigma = 3;
>> h = fspecial('gaussian', width, sigma);
>> I = imfilter(im, h, 'symmetric', 'same');
>> imshow(I, []);
```

```
>> sigma = 16;
>> noise = randn(size(im))*sigma;
>> I = im + noise;
>> imshow(I, []);
```

```
>> sigma = 16;
>> noise = randn(size(im))*sigma;
>> I = im + noise;
>> imshow(I, []);
```

```
>> sigma = 16;
>> noise = randn(size(im))*sigma;
>> I = im + noise;
>> imshow(I, []);
```

```
>> sigma = 16;
>> noise = randn(size(im))*sigma;
>> I = im + noise;
>> imshow(I, []);
```

```
>> sigma = 16;
>> noise = randn(size(im))*sigma;
>> I = im + noise;
>> imshow(I, []);
```

```
>> sigma = 16;
>> noise = randn(size(im))*sigma;
>> I = im + noise;
>> imshow(I, []);
```

What about near the image edges?

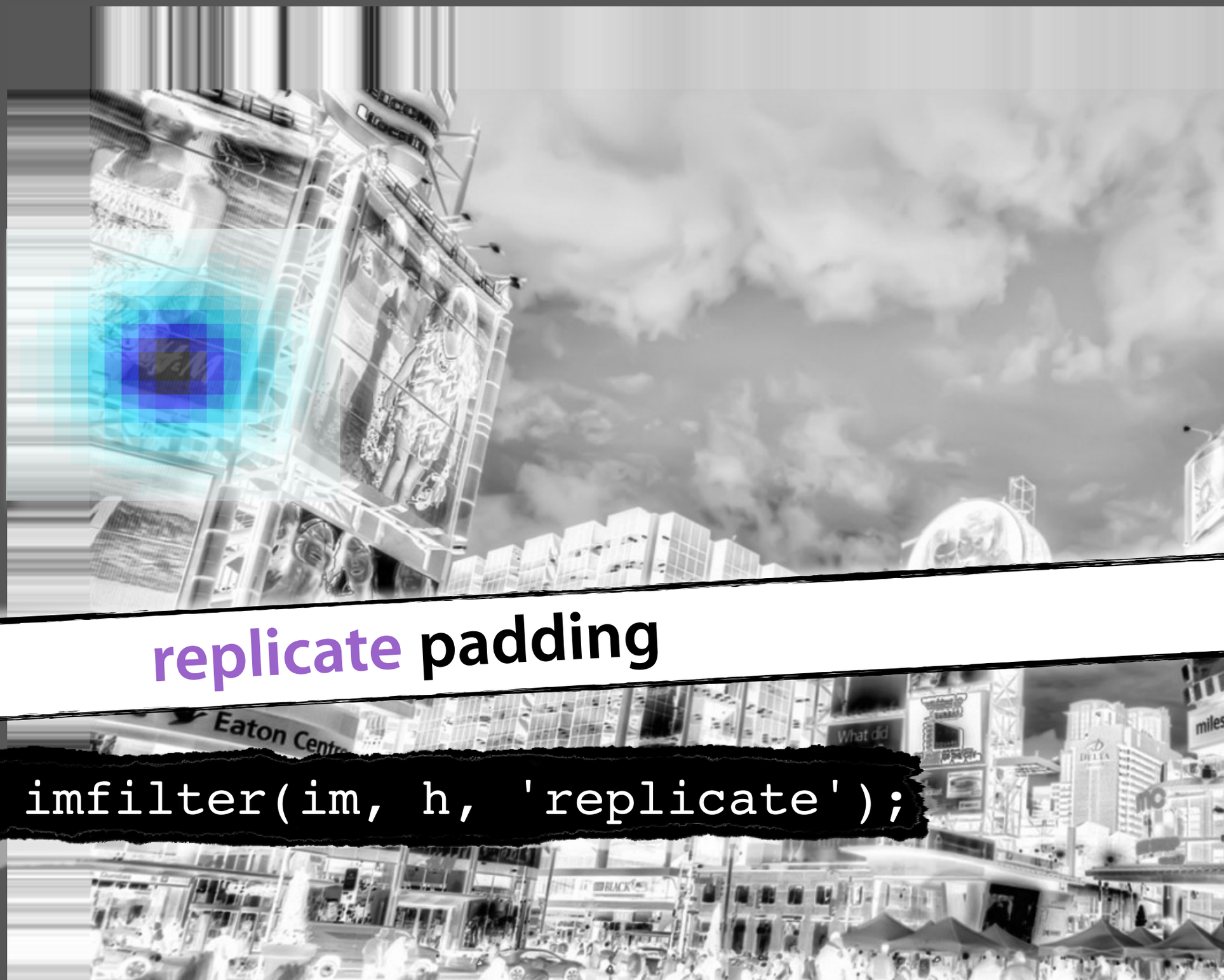

Eaton Centre
What did you get?
IT'S BETTER TO BE A MEMBER.
BREAK OUT OF THE CITY

clip padding

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3	1	0
0	3	1	2	2	3	0
0	2	0	0	2	2	0
0	2	0	0	0	0	0
0	0	0	0	0	0	0

symmetric padding

```
I = imfilter(im, h, 'symmetric');
```



replicate padding

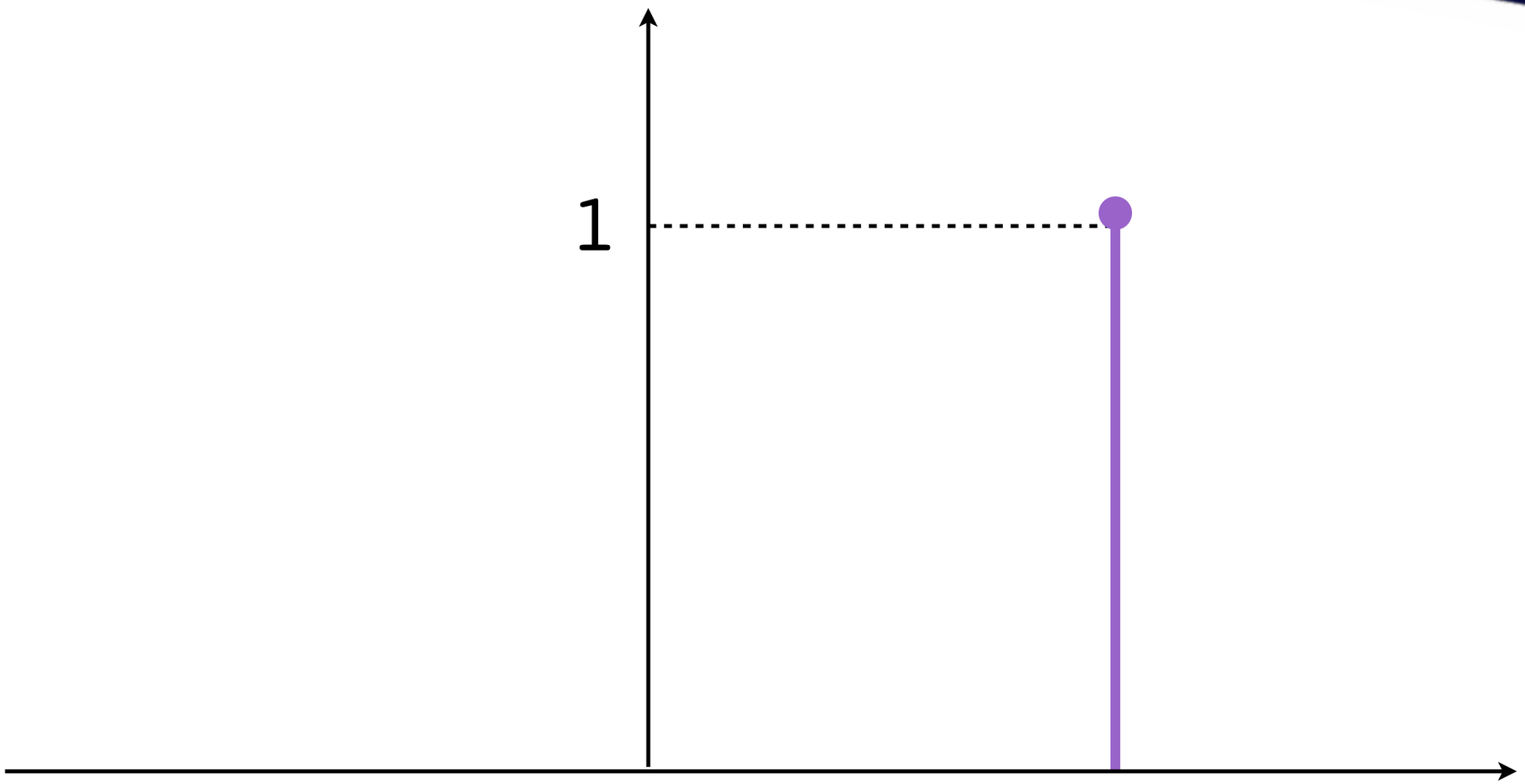
```
I = imfilter(im, h, 'replicate');
```

circular padding

```
I = imfilter(im, h, 'circular');
```

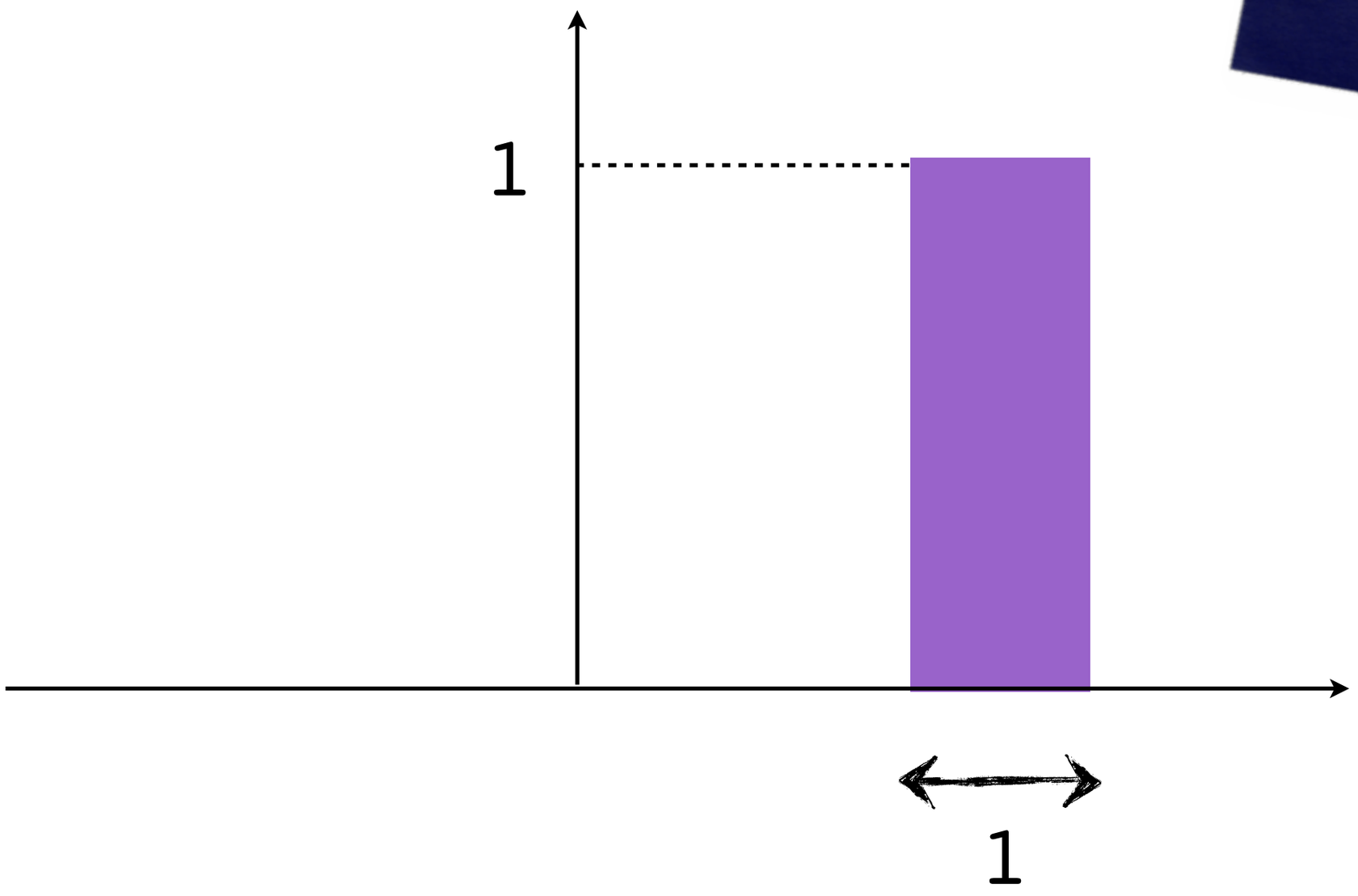
Convolution

discrete impulse



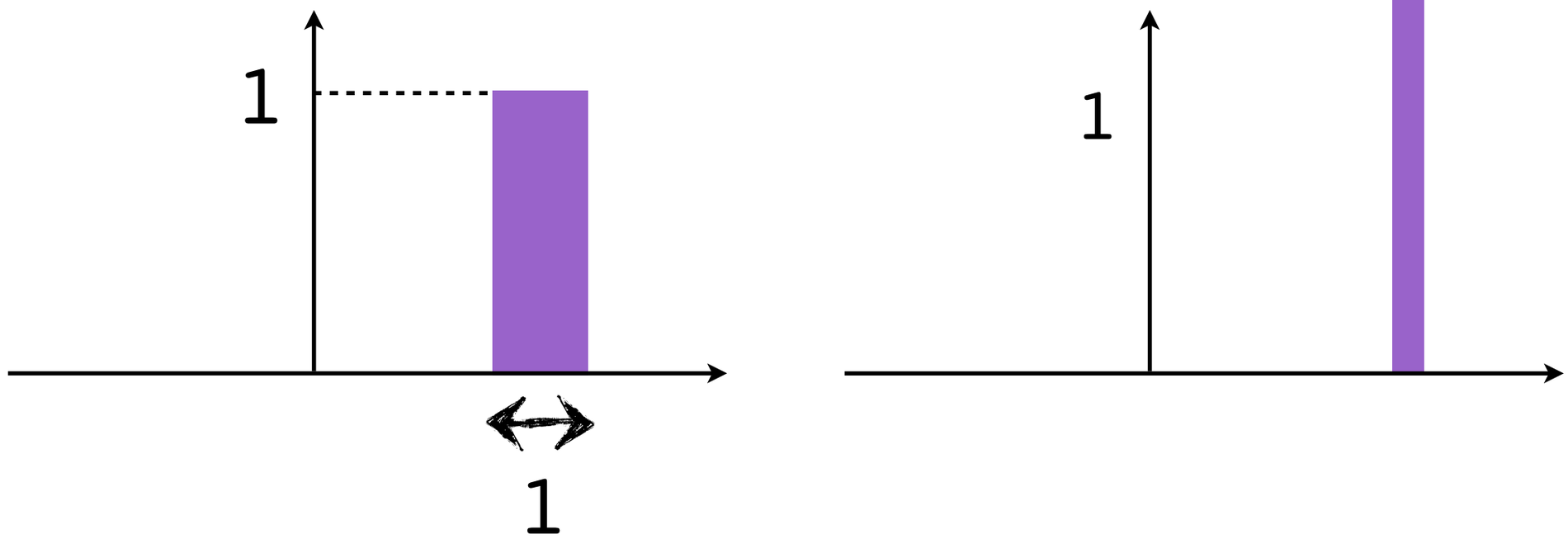
A function with a value of 1 at a single location

continuous impulse



A function that is very narrow and very tall such that at the limit it has a unit area

continuous impulse

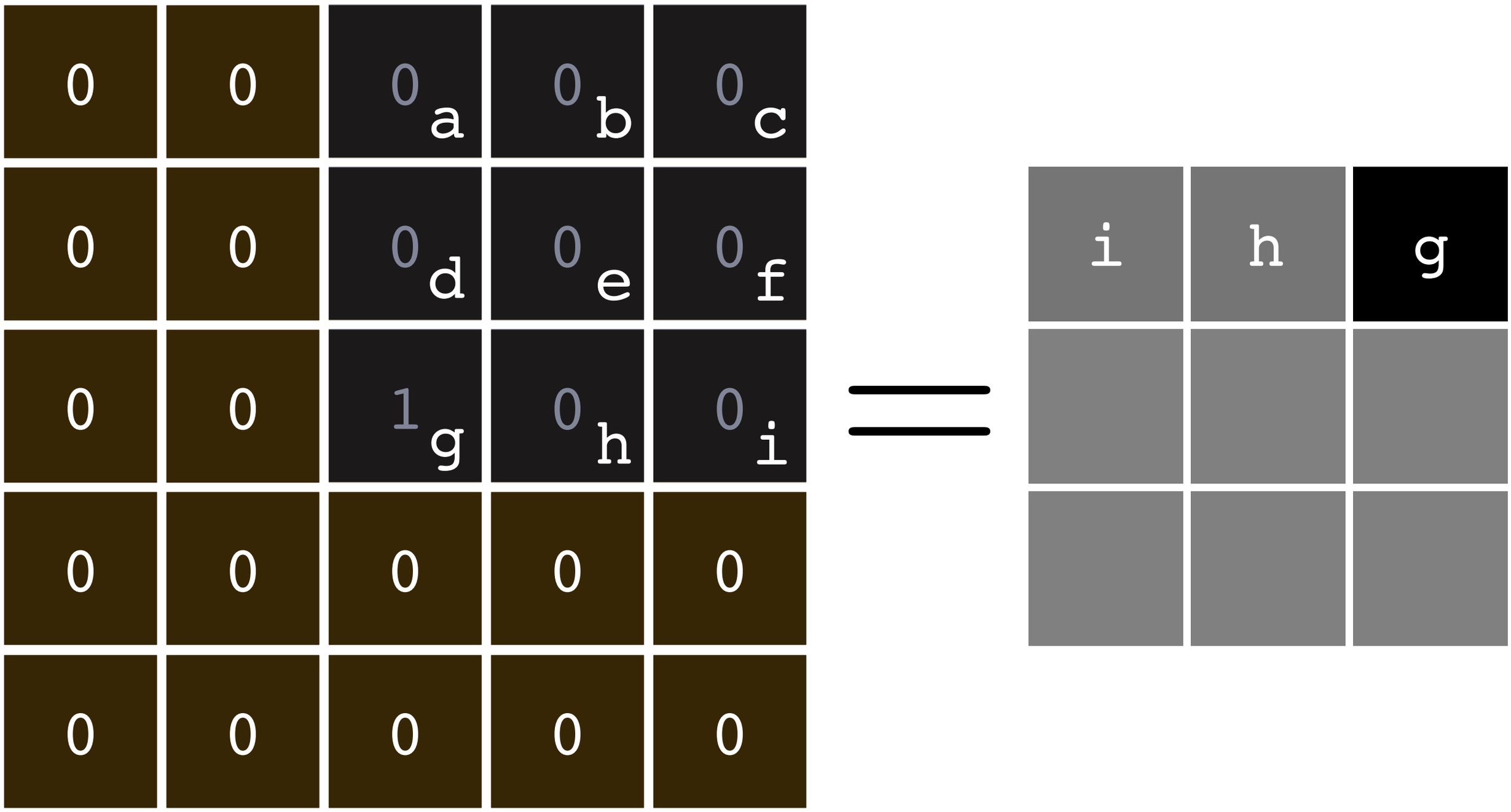


A function that is very narrow and very tall such that at the limit it has a unit area

0	0_{a}	0_{b}	0_{c}	0
0	0_{d}	0_{e}	0_{f}	0
0	0_{g}	1_{h}	0_{i}	0
0	0	0	0	0
0	0	0	0	0

$$=$$

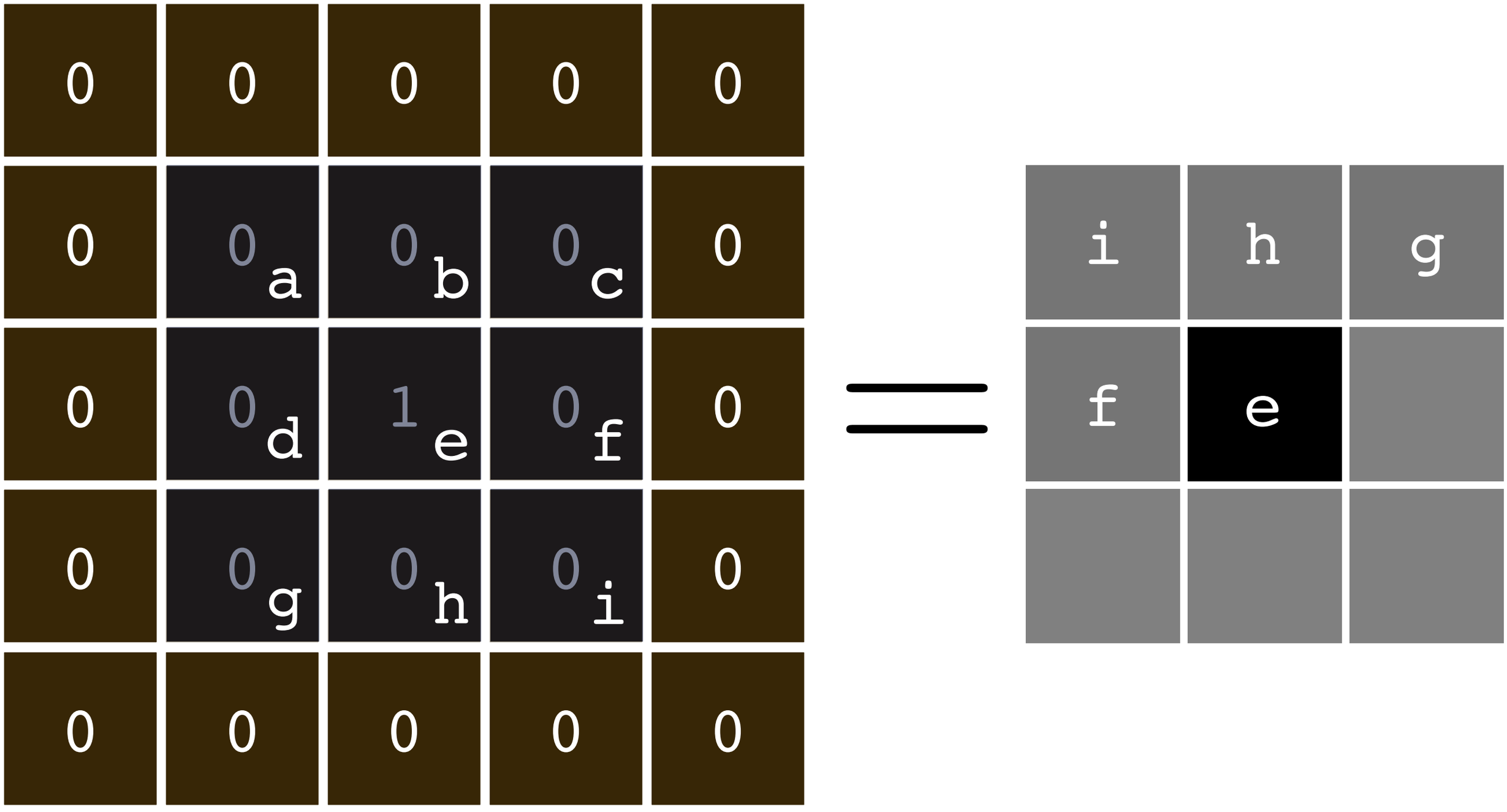
i	h	


0 0 0a 0b 0c
0 0 0d 0e 0f
0 0 1g 0h 0i
0 0 0 0 0
0 0 0 0 0
i h g

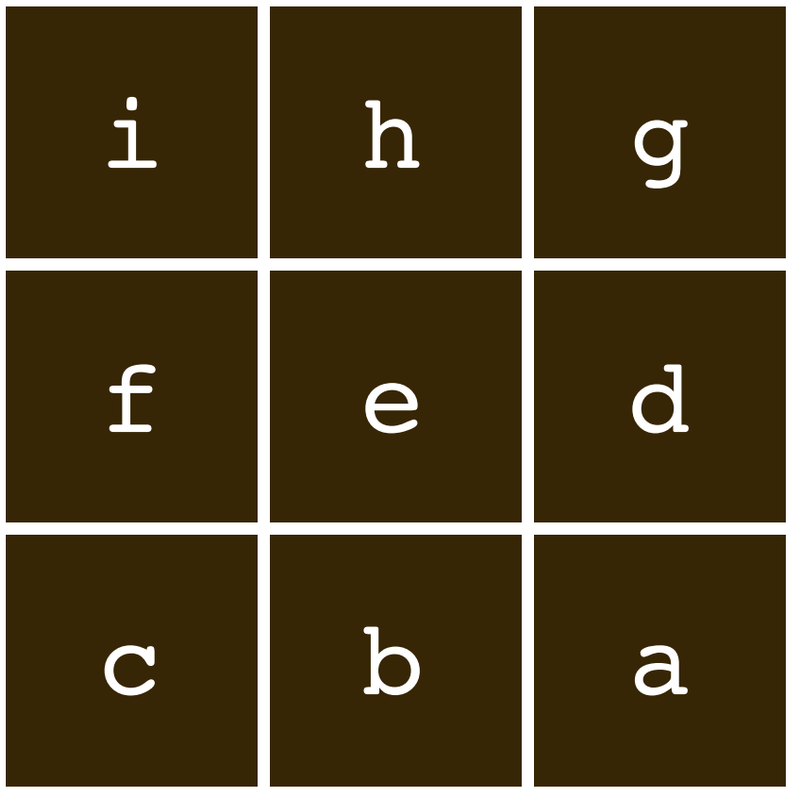
0	0	0	0	0
0_{a}	0_{b}	0_{c}	0	0
0_{d}	0_{e}	1_{f}	0	0
0_{g}	0_{h}	0_{i}	0	0
0	0	0	0	0

=

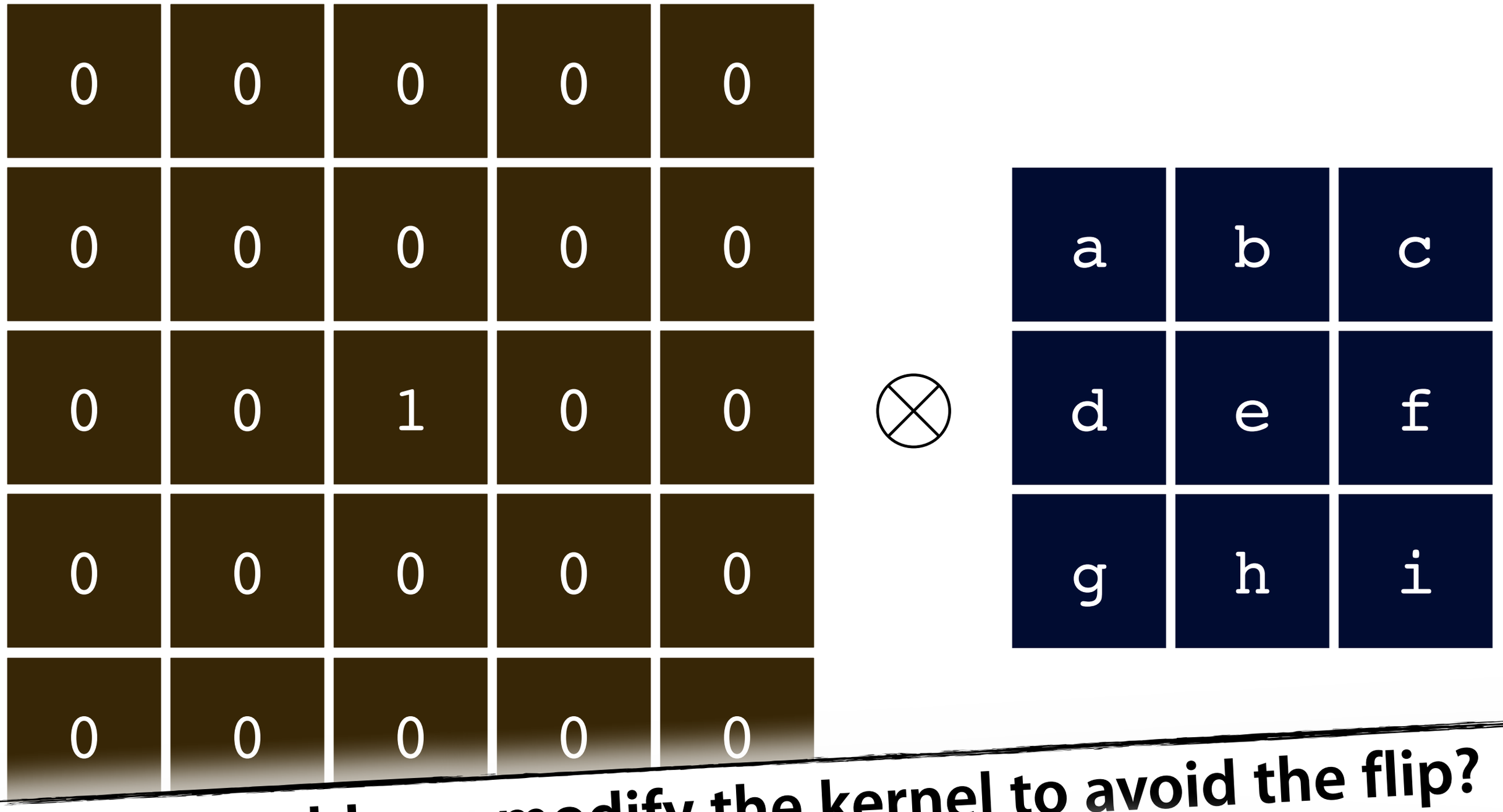
i	h	g
f		


0 0 0 0 0
0 0a 0b 0c 0
0 0d 1e 0f 0
0 0g 0h 0i 0
0 0 0 0 0
i h g
f e

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

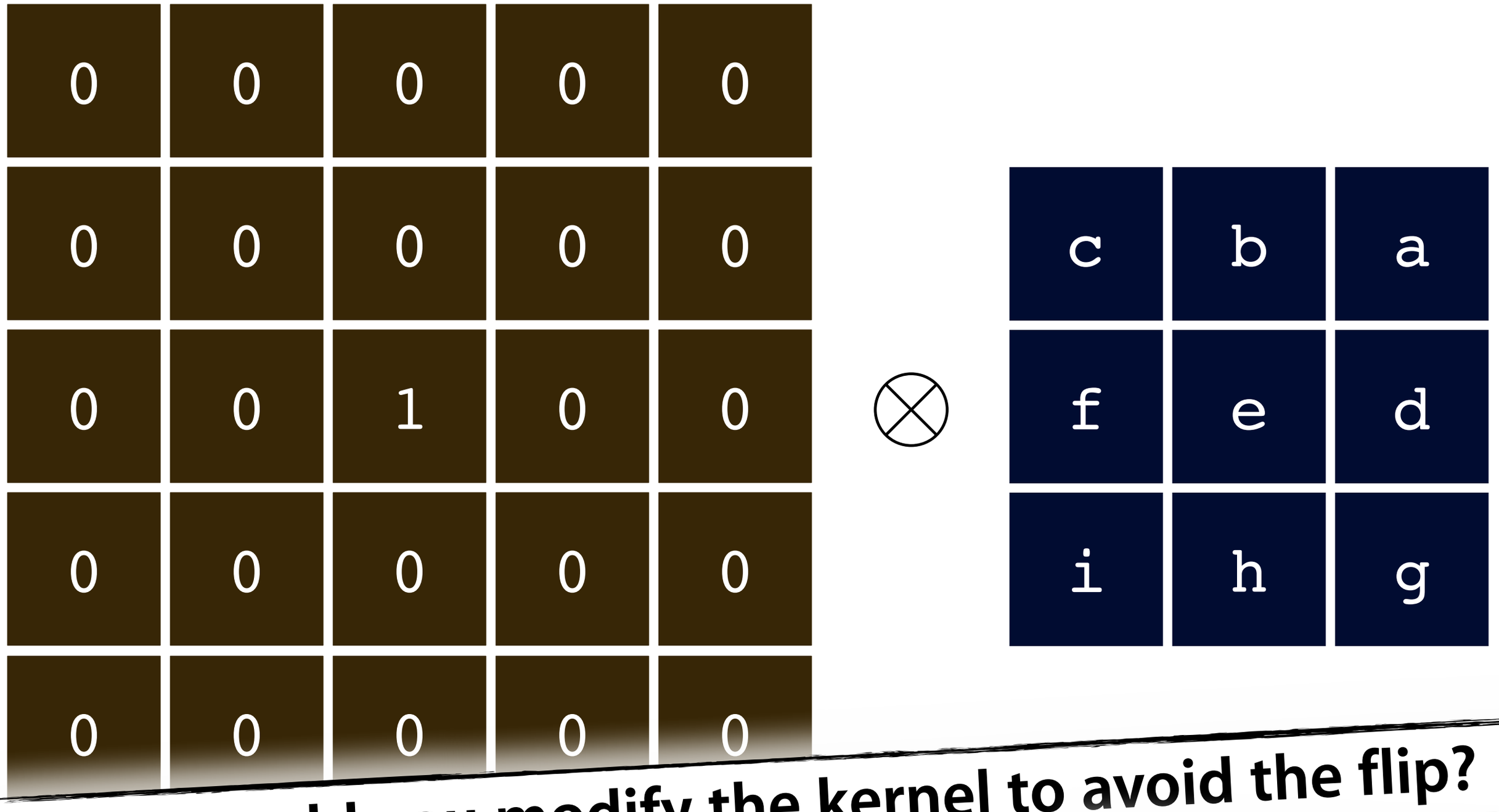


Filter output is REVERSED



How would you modify the kernel to avoid the flip?

Flip the kernel



How would you modify the kernel to avoid the flip?

Flip the kernel

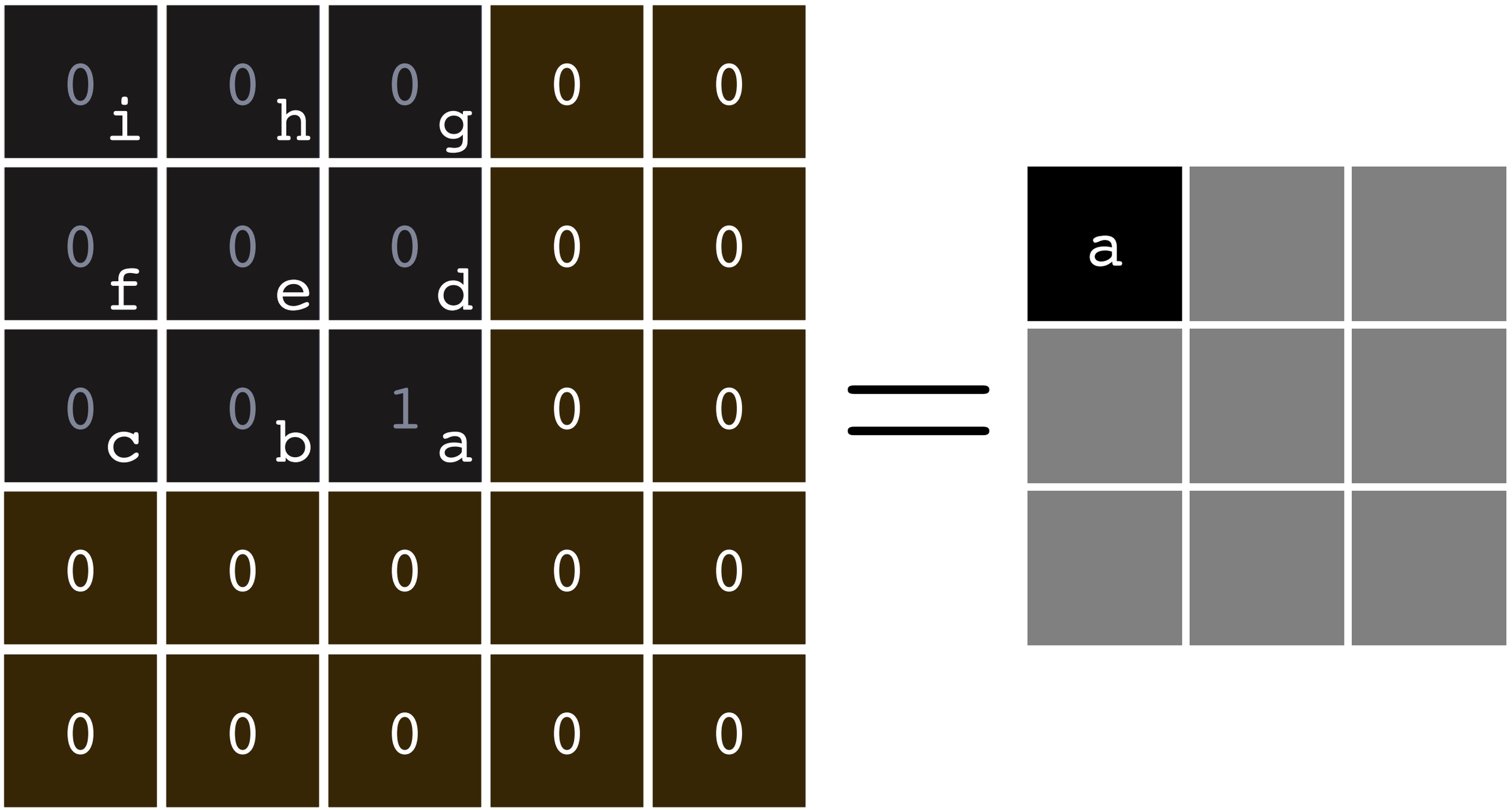
0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

⊗

i	h	g
f	e	d
c	b	a

How would you modify the kernel to avoid the flip?

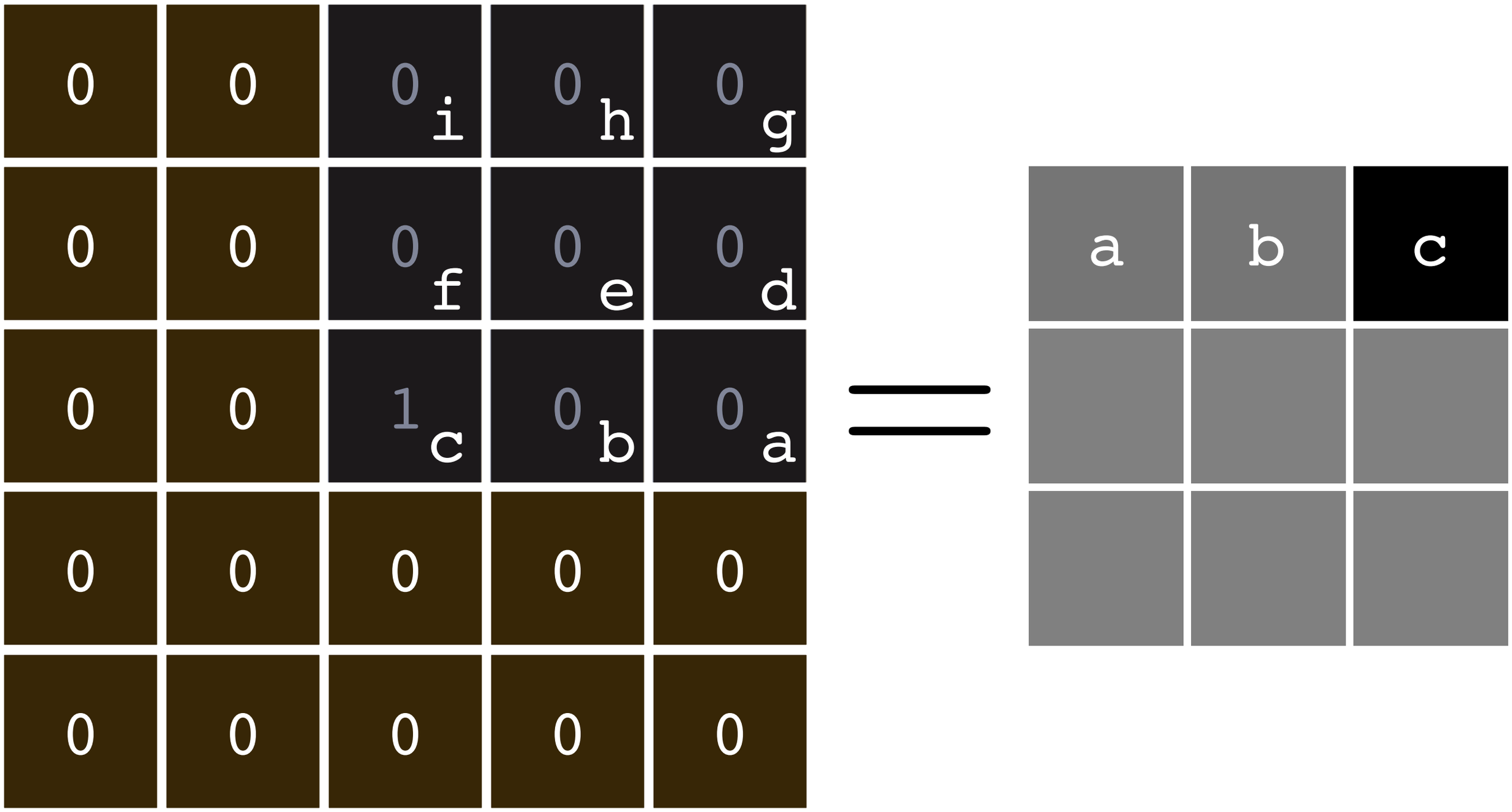
Flip the kernel


0 i
0 h
0 g
0
0
0 f
0 e
0 d
0
0
0 c
0 b
1 a
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
a

0	0 i	0 h	0 g	0
0	0 f	0 e	0 d	0
0	0 c	1 b	0 a	0
0	0	0	0	0
0	0	0	0	0

$$\equiv$$

a	b	


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
i h g
f e d
c b a
a b c

0	0	0	0	0
0	0	0	0	0
0	0	1 i	0 h	0 g
0	0	0 f	0 e	0 d
0	0	0 c	0 b	0 a

=

a	b	c
d	e	f
g	h	i

$$(2K+1) \times (2K+1)$$

filter dimensions

convolution

$$H[x,y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u,v]F[x-u,y-v]$$

correlation

$$H[x,y] = \frac{1}{(2K+1)^2} \sum_{u=-K}^{K} \sum_{v=-K}^{K} F[x+u, y+v]\, G[u,v]$$

convolution

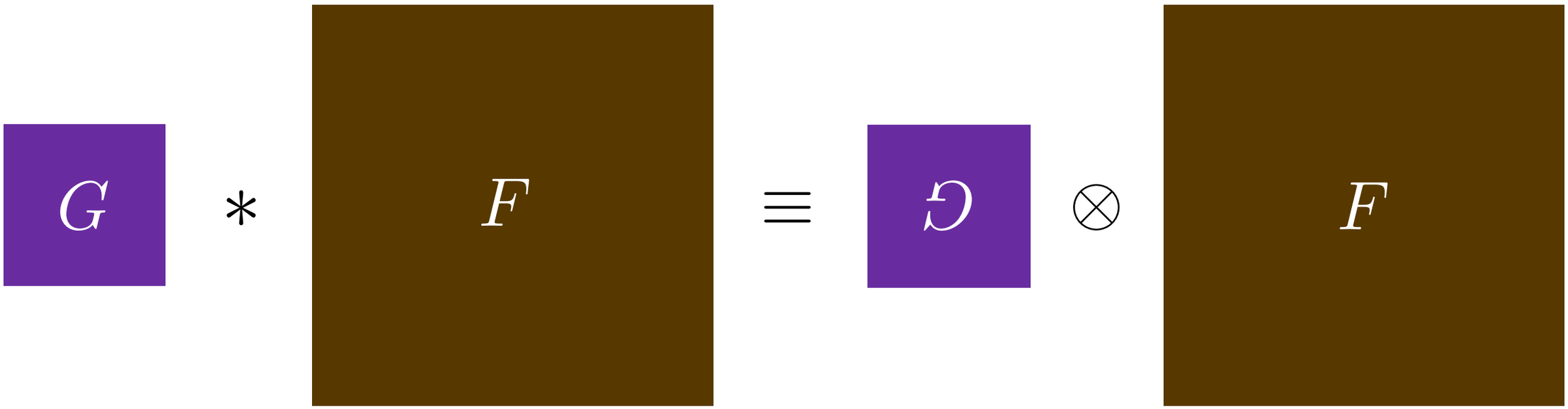
$$H[x,y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u,v]F[x-u,y-v]$$

$$H = G * F$$

convolution

convolution

$$H[x,y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u,v]F[x-u,y-v]$$



Convolution Properties

$$F[x,y] * \delta[x,y] = F[x,y]$$

$$E[x,y] * (F[x,y] + G[x,y])$$

$$= (E[x,y] * F[x,y]) + (E[x,y] * G[x,y])$$

$$G * (\alpha F_1[x, y] + \beta F_2[x, y])$$
$$= \alpha H_1[x, y] + \beta H_2[x, y]$$

shift
invariant

$$G[x, y] * F[x - \alpha, y - \beta]$$
$$= H[x - \alpha, y - \beta]$$

associative

$$(E[x,y] * F[x,y]) * G[x,y]$$

$$= E[x,y] * (F[x,y] * G[x,y])$$

sequential filtering is equivalent to applying one filter

$$G(\mathbf{x}, \sqrt{\sigma_1^2 + \sigma_2^2}) = G(\mathbf{x}, \sigma_1) * G(\mathbf{x}, \sigma_2)$$

convolution of Gaussians is a Gaussian

associative

$$(E[x,y] * F[x,y]) * G[x,y]$$

$$= E[x,y] * (F[x,y] * G[x,y])$$

sequential filtering is equivalent to applying one filter

correlation

$$\left(\begin{array}{|c|c|} \hline -1 & 1 \\ \hline \end{array} \otimes \begin{array}{|c|c|} \hline 1 & -1 \\ \hline \end{array} \right) \otimes \begin{array}{|c|c|} \hline -1 & 1 \\ \hline \end{array}$$

correlation

$$\left(\begin{array}{|c|c|}\hline 1 & 1 \\ \hline -1 & 1 \\ \hline \end{array}\right) \otimes \begin{array}{|c|c|}\hline -1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline -1 & 3 & -3 & 1 \\ \hline \end{array}$$

correlation

$$\left(\begin{bmatrix} -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} -1 & 1 \end{bmatrix}$$

correlation

$$\begin{bmatrix} -1 & 1 \end{bmatrix} \otimes \left(\begin{bmatrix} 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} -1 & 1 \end{bmatrix} \right)$$

correlation

$$\boxed{1}\boxed{-3}\boxed{3}\boxed{-1} = \boxed{-1}\boxed{1} \otimes \left(\boxed{1} \boxed{-} \right.$$

1	−3	3	−1	≠	−1	3	−3	1

1	−3	3	−1	≠	−1	3	−3	1

correlation is not associative

$$F[x,y] * G[x,y] = G[x,y] * F[x,y]$$

commutativity proof

By definition

$$H * F = \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v] F[x - u, y - v]$$

Let $p = x - u$ **and** $q = y - v$

substitute

$$H * F = \sum_{q=-\infty}^{\infty} \sum_{p=-\infty}^{\infty} H[x - p, y - q] F[p, q]$$

rearrange

$$= \sum_{q=-\infty}^{\infty} \sum_{p=-\infty}^{\infty} F[p, q] H[x - p, y - q]$$

$$= F * H$$


motion
blur

motion
blur

blurred image

motion blur

=

blurred image

ideal image

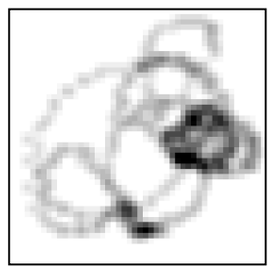
motion blur

blurred image

=

ideal image

*



kernel

$$H[x,y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u,v]F[x-u,y-v]$$

How many operations does convolution require per pixel?

$(2K+1)^2$

separable filters

$$F[x, y] = U[x]V[y]$$

Assume the filter can be written as

$$F[x, y] = U[x]V[y]$$

$$H * F = \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v]F[x - u, y - v]$$

insert filter definition

Assume the filter can be written as

$$F[x, y] = U[x]V[y]$$

$$H * F = \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v]F[x - u, y - v]$$

insert filter definition

$$= \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v]U[x - u]V[y - v]$$

factor out

$$= \sum_{v=-\infty}^{\infty} V[y - v]\left(\sum_{u=-\infty}^{\infty} H[u, v]U[x - u] \right)$$

Assume the filter can be written as

$$F[x, y] = U[x]V[y]$$

$$H * F = \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v]F[x - u, y - v]$$

insert filter definition

$$= \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v]U[x - u]V[y - v]$$

factor out

$$= \sum_{v=-\infty}^{\infty} V[y - v]\left(\sum_{u=-\infty}^{\infty} H[u, v]U[x - u] \right)$$

1D horizontal convolution

Assume the filter can be written as

$$F[x, y] = U[x]V[y]$$

$$H * F = \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v]F[x - u, y - v]$$

insert filter definition

$$= \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v]U[x - u]V[y - v]$$

factor out

$$= \sum_{v=-\infty}^{\infty} V[y - v]\left(\sum_{u=-\infty}^{\infty} H[u, v]U[x - u] \right)$$

1D vertical convolution

Assume the filter can be written as

$$F[x, y] = U[x]V[y]$$

$$H * F = \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v]F[x - u, y - v]$$

insert filter definition

$$= \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v]U[x - u]V[y - v]$$

factor out

$$= \sum_{v=-\infty}^{\infty} V[y - v]\left(\sum_{u=-\infty}^{\infty} H[u, v]U[x - u] \right)$$

Filter horizontally then vertically

Assume the filter can be written as

$$F[x, y] = U[x]V[y]$$

$$H * F = \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v]F[x - u, y - v]$$

insert filter definition

$$= \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u, v]U[x - u]V[y - v]$$

factor out

$$= \sum_{v=-\infty}^{\infty} V[y - v]\left(\sum_{u=-\infty}^{\infty} H[u, v]U[x - u] \right)$$

How many operations does convolution require per pixel?

$$2(2K + 1)$$

convolution

$$H[x,y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u,v]F[x-u,y-v]$$

How many operations does convolution require per pixel?

$$(2K+1)^2$$

separable filtering

Assume the filter can be written as:

$$F[x,y] = U[x]V[y]$$

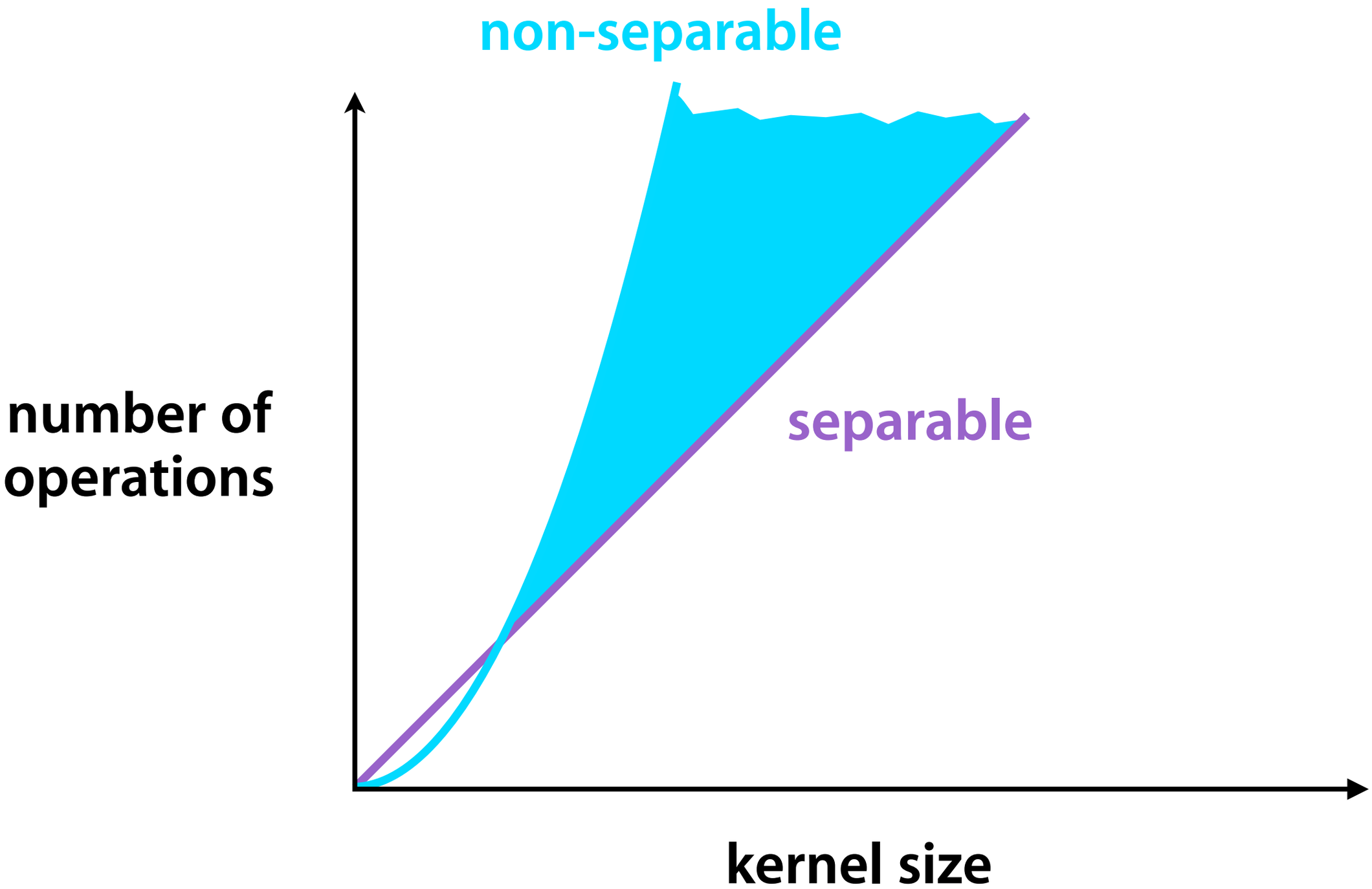
$$H * F = \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u,v]F[x-u,y-v]$$

$$= \sum_{v=-\infty}^{\infty} \sum_{u=-\infty}^{\infty} H[u,v]U[x-u]V[y-v]$$

$$= \sum_{v=-\infty}^{\infty} V[y-v]\left(\sum_{u=-\infty}^{\infty} H[u,v]U[x-u]\right)$$

How many operations does convolution require per pixel?

$2(2K+1)$


non-separable
separable
number of operations
kernel size

Show that the 2D Gaussian filter is separable.

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

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

Show that the 2D Gaussian filter is separable.

$$
\begin{aligned}
G(x,y) &= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \\
&= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2}{2\sigma^2}\right)\exp\left(-\frac{y^2}{2\sigma^2}\right) \\
&= \left[\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{x^2}{2\sigma^2}\right)\right]\left[\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{y^2}{2\sigma^2}\right)\right] \\
&= G_1(x)G_1(y)
\end{aligned}
$$

For a symmetric filter how will the outputs differ?

no difference between outputs

For an impulse signal, how will the outputs differ?

output of convolution is flipped

For an impulse filter, how will the outputs differ?

no difference between outputs

image sharpening

input

blurred

“sharp-stuff”

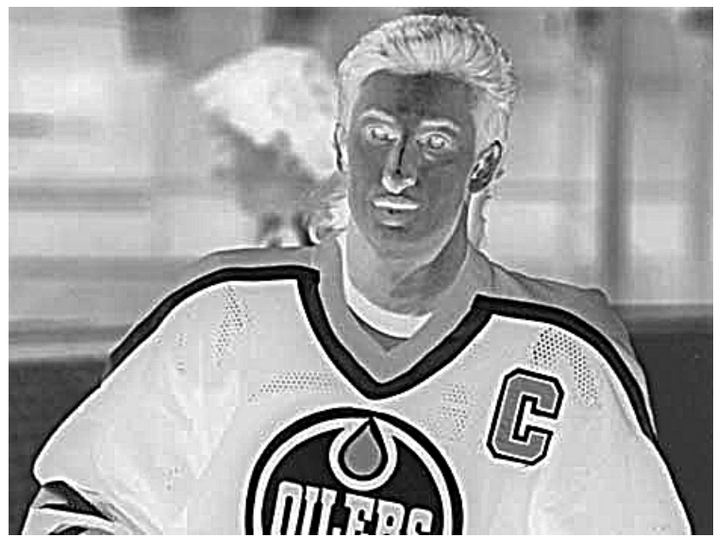
image sharpening

blurred

+

“sharp-stuff”

=

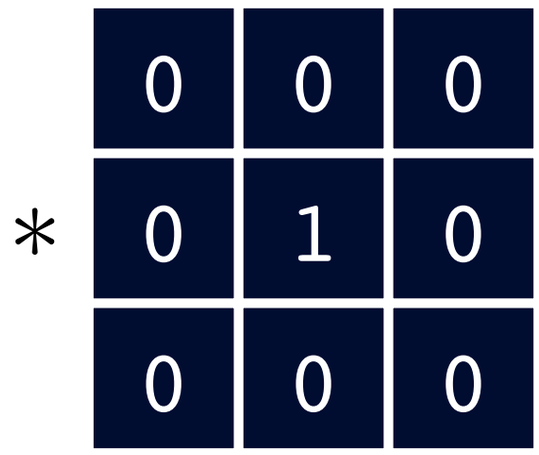


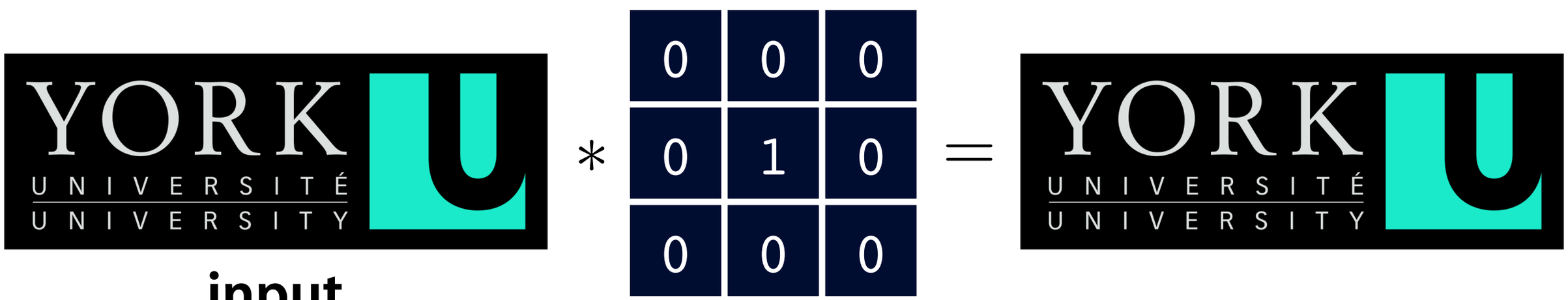
sharpened

−1	−1	−1
−1	9	−1
−1	−1	−1

Sharpening Filter

input

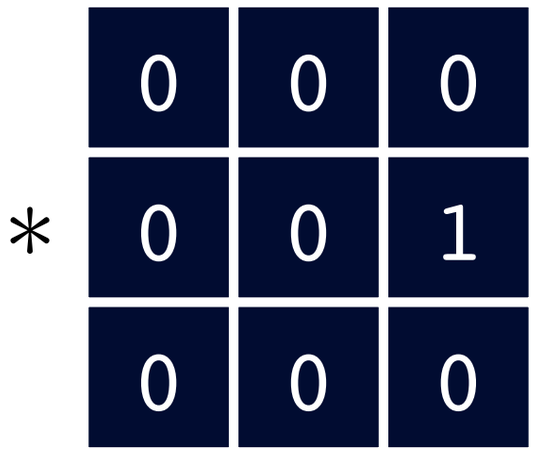




input

no change to input image

input



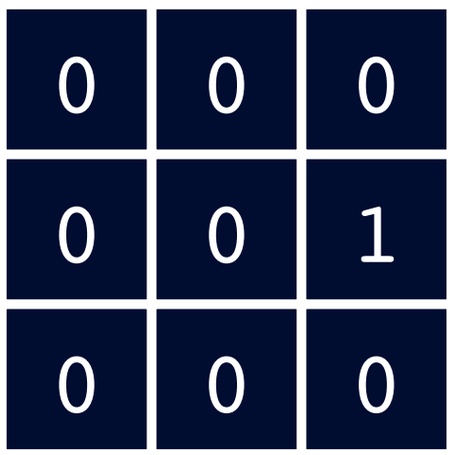
input

$*$

0	0	0
0	0	1
0	0	0

$=$

t

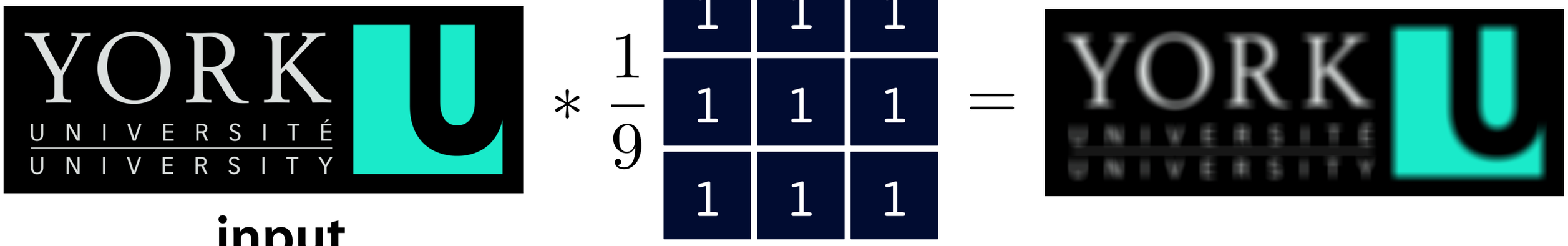


=

shift input image one pixel to the right

input

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

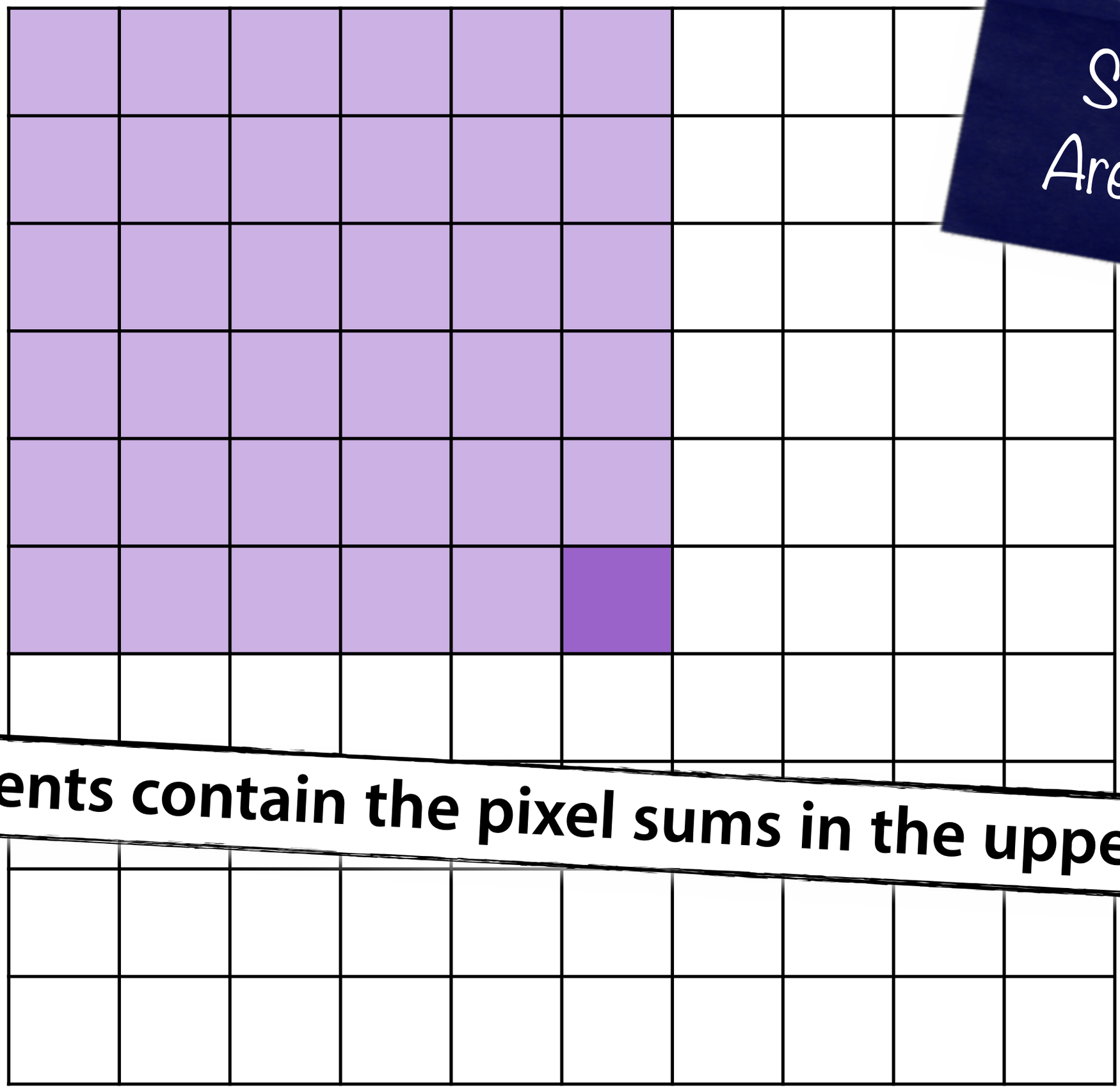


input

input image blurred by box filtering

108	76	98	14	246	146	56	236	14	12
20	44	180	62	244	8	20	156	150	122
224	144	230	80	108	76	140	114	44	144
174	234	26	56	34	234	228	40	100	34
152	194	90	30	202	22	200	226	130	50
164	112	40	52	200	0	8	44	252	102
186	34	150	212	102	236	48	72	228	54
118	200	196	28	94	34	32	52	104	160
52	34	18	90	202	52	110	218	252	74

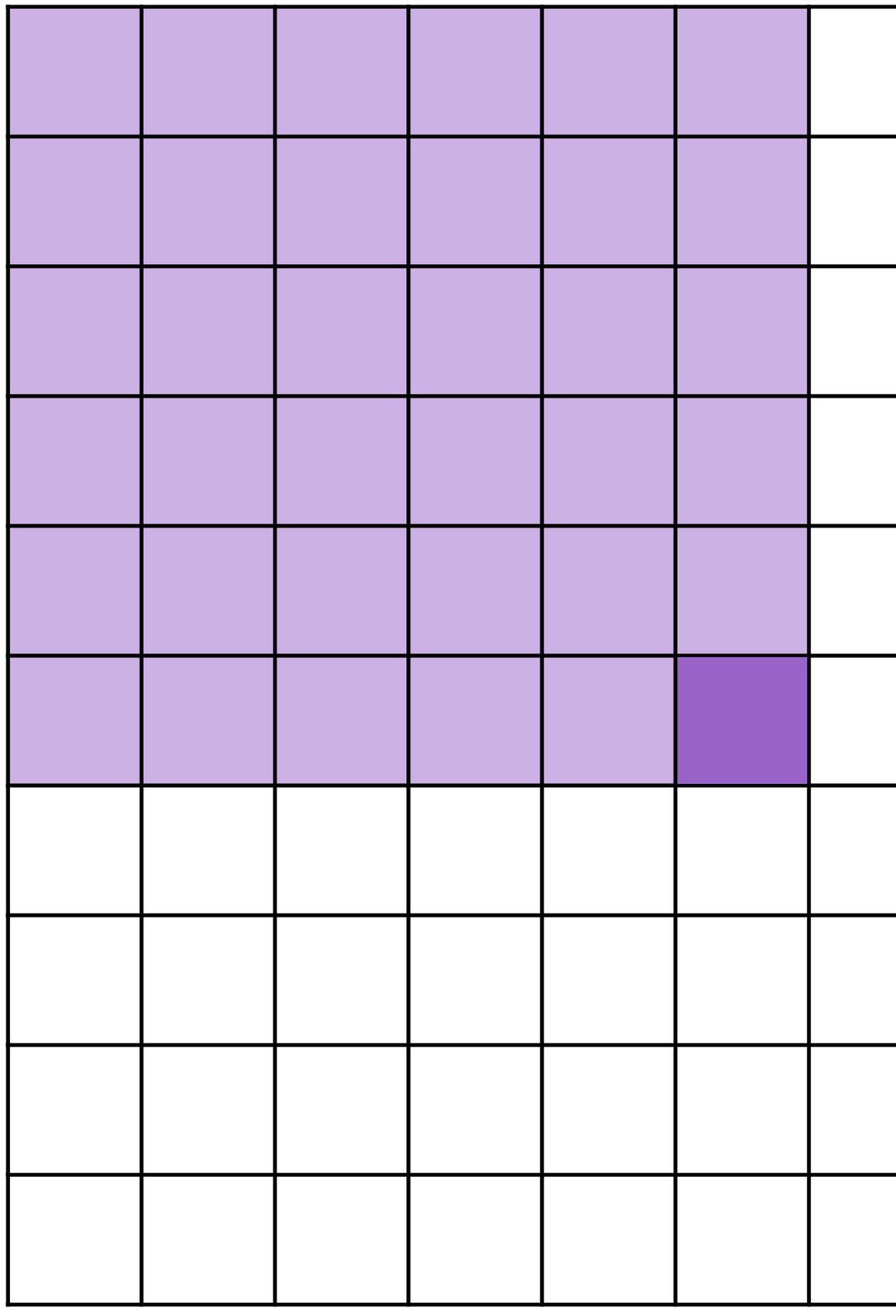
How can we compute the sum within the box?

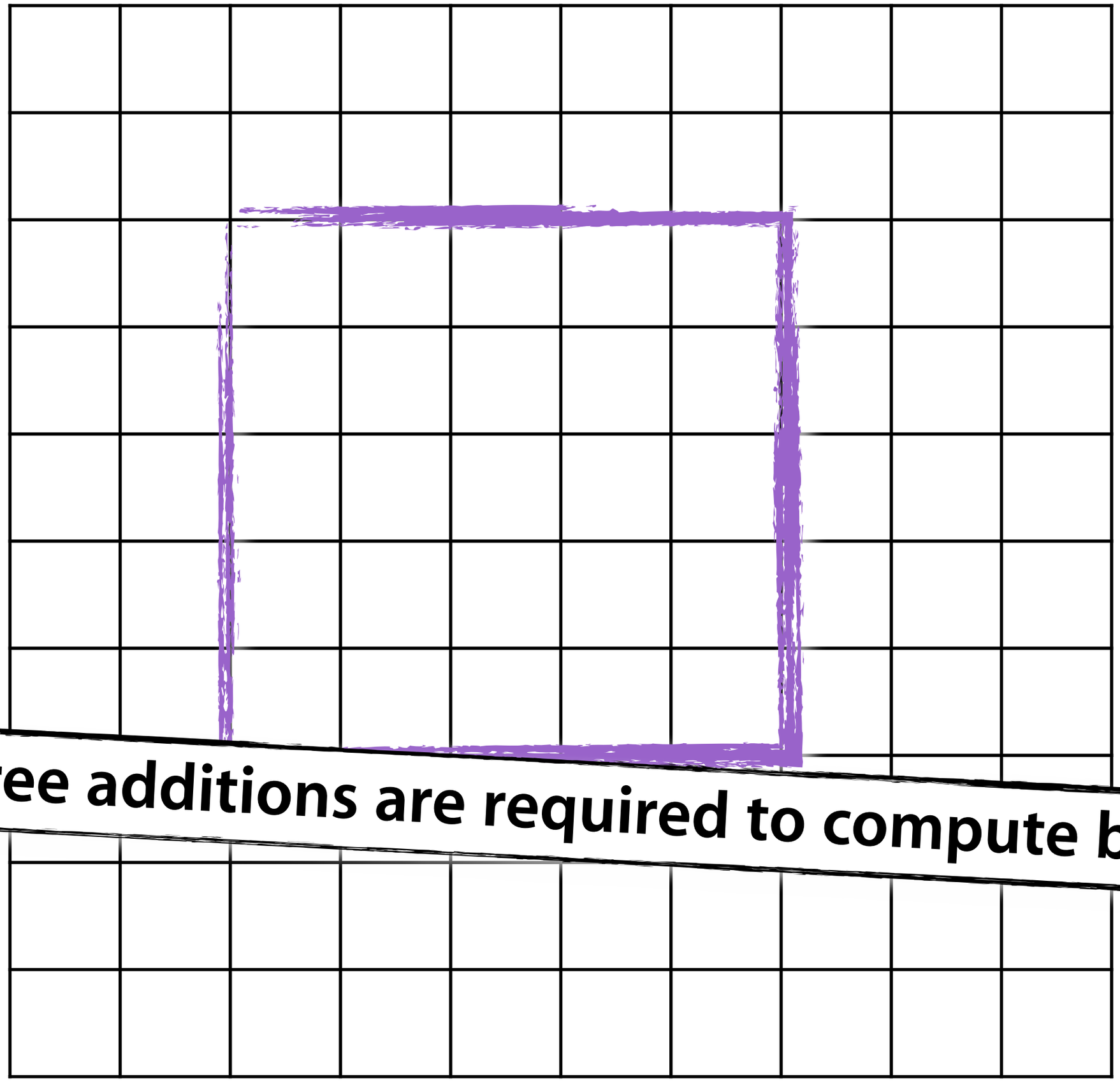


Summed Area Table

elements contain the pixel sums in the upper-left

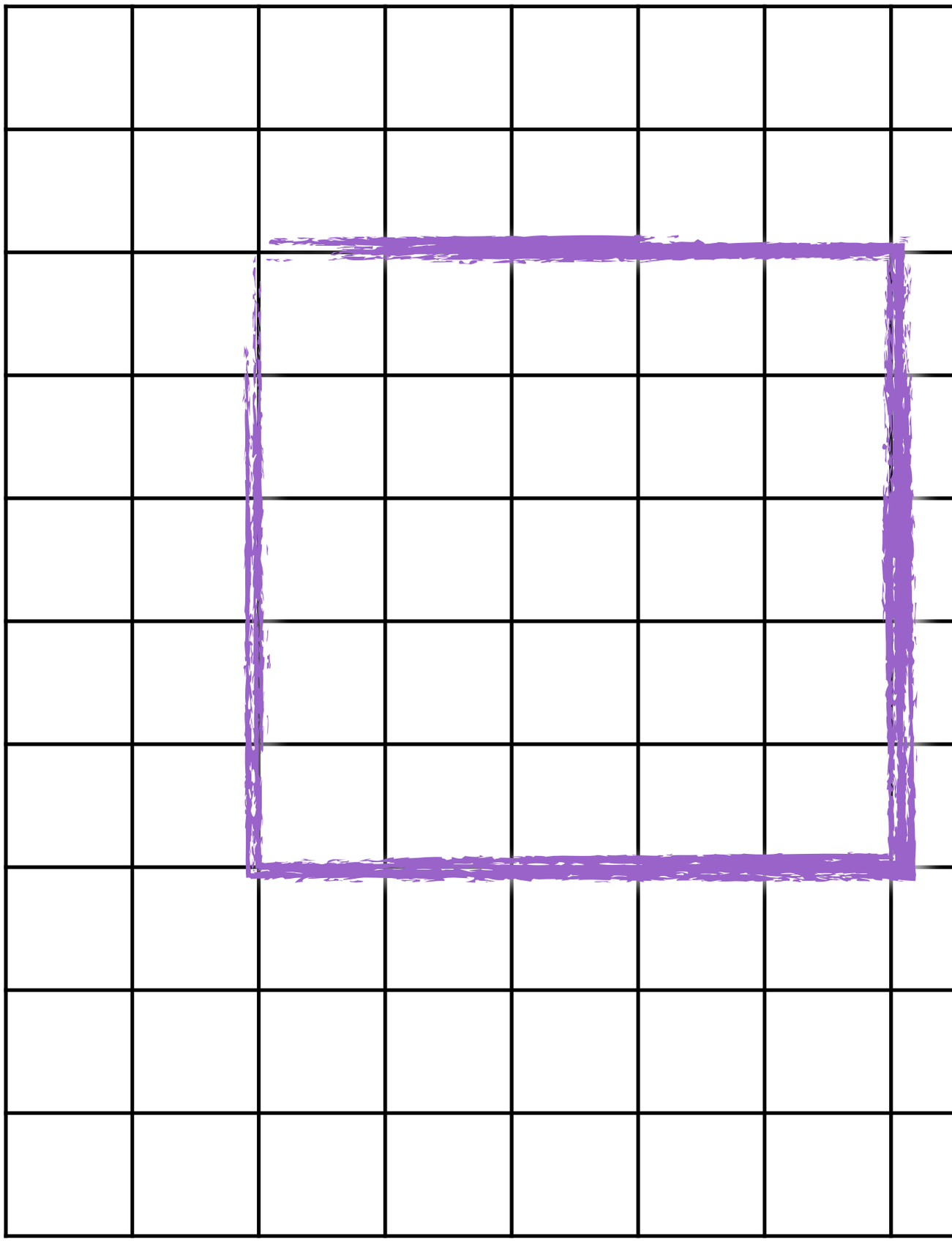
$$
\begin{aligned}
I_{\text{S.A.T}}[x,y] = \\
I[x,y] + \\
I_{\text{S.A.T.}}[x-1,y] + \\
I_{\text{S.A.T.}}[x,y-1] - \\
I_{\text{S.A.T.}}[x-1,y-1]
\end{aligned}
$$



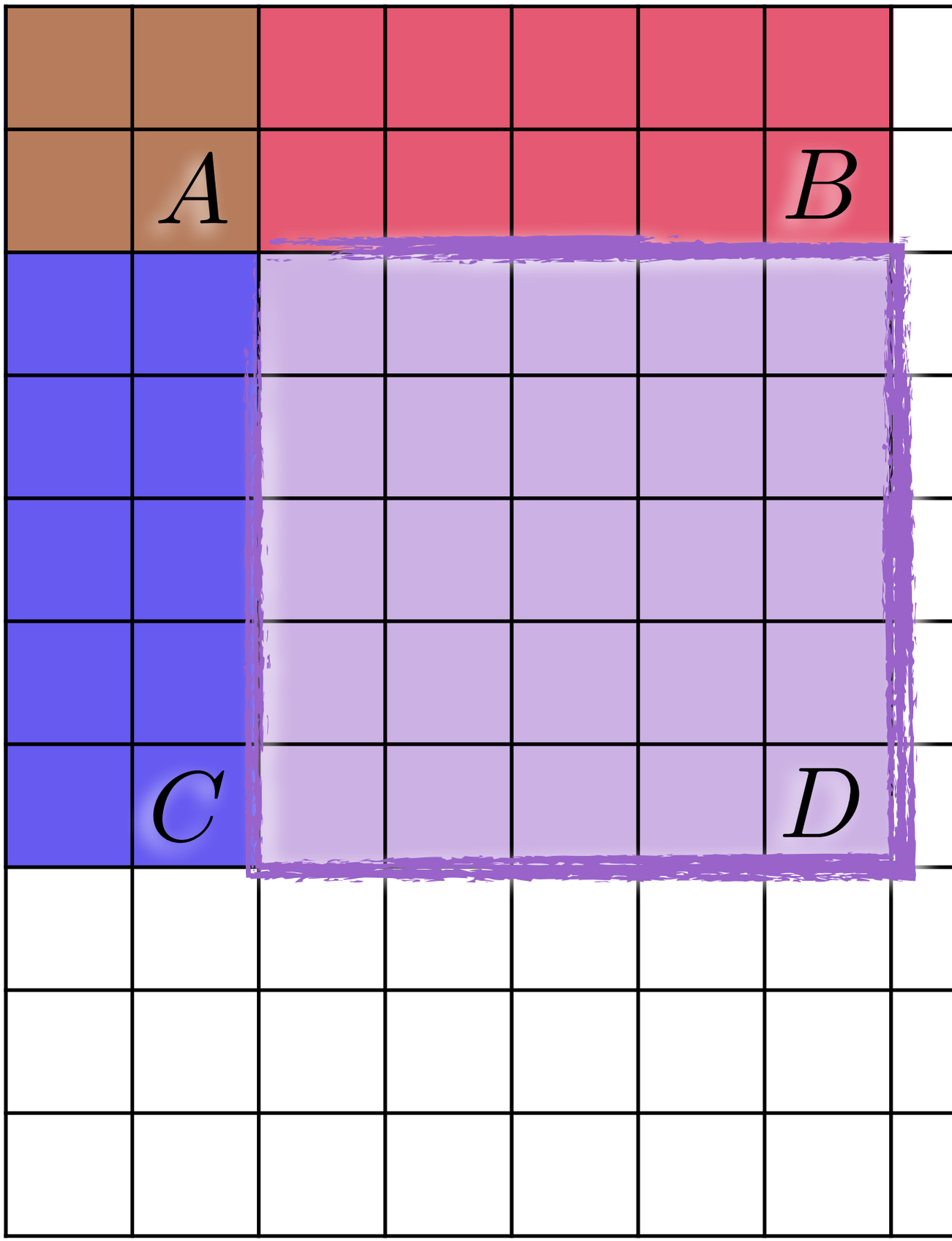


only three additions are required to compute box sum

$$D - B - C + A$$



$$D - B - C + A$$



Non-linear Filtering


salt and pepper noise

10	15	20
23	90	27
33	31	30

sort pixel values

10 15 20 23 **27** 30 31 33 90

replace pixel with median value

median filtered output

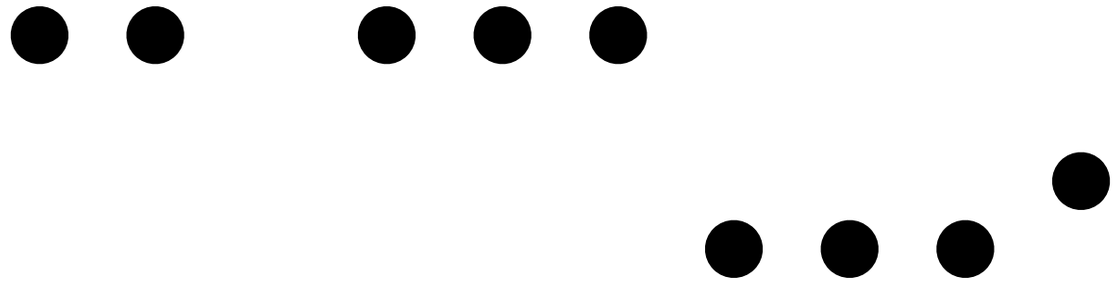
large kernel sizes can remove image details

median filtered output

median filtering

outlier

input



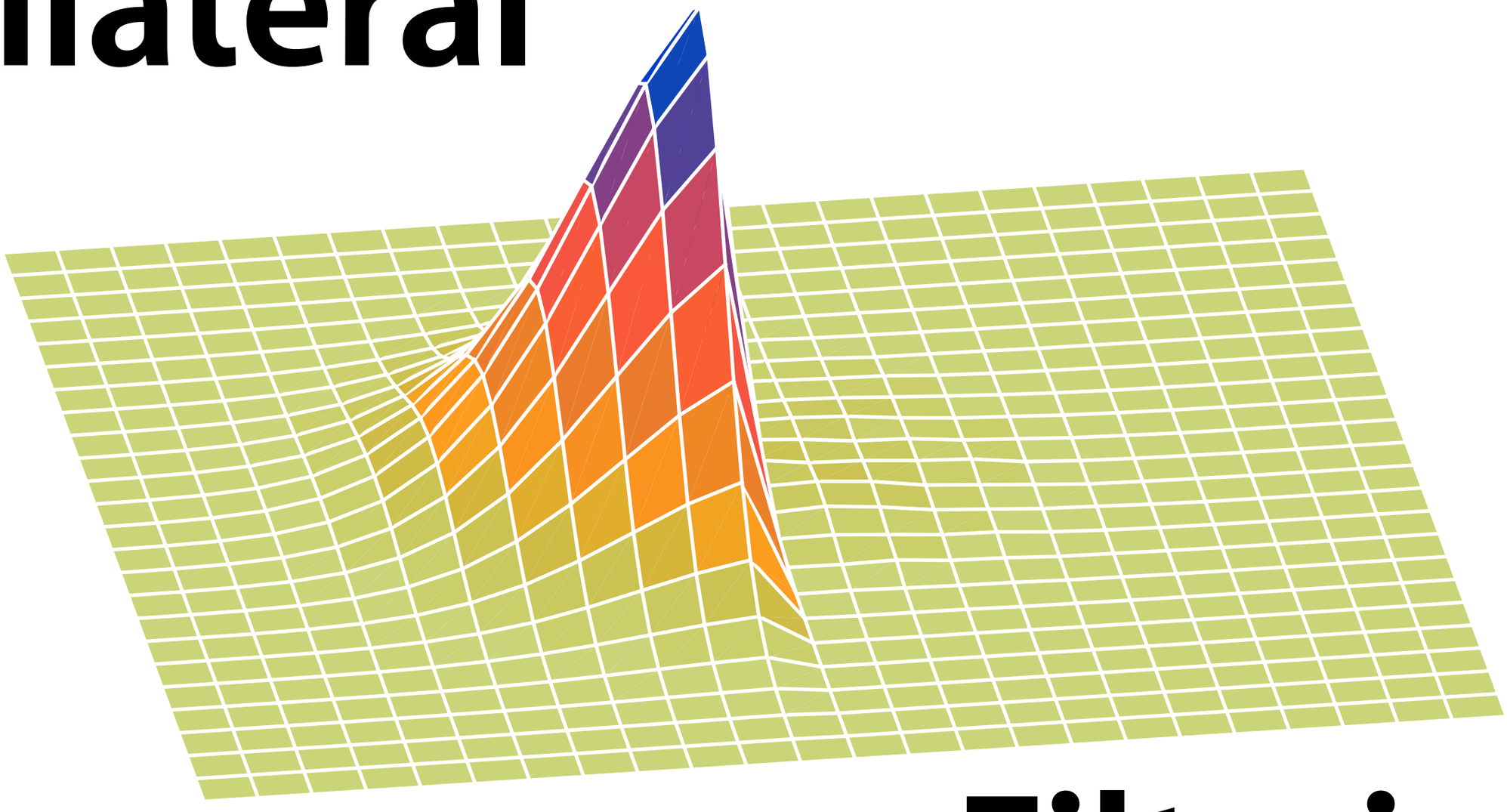
median filtering

input

median filtered

mean filtered

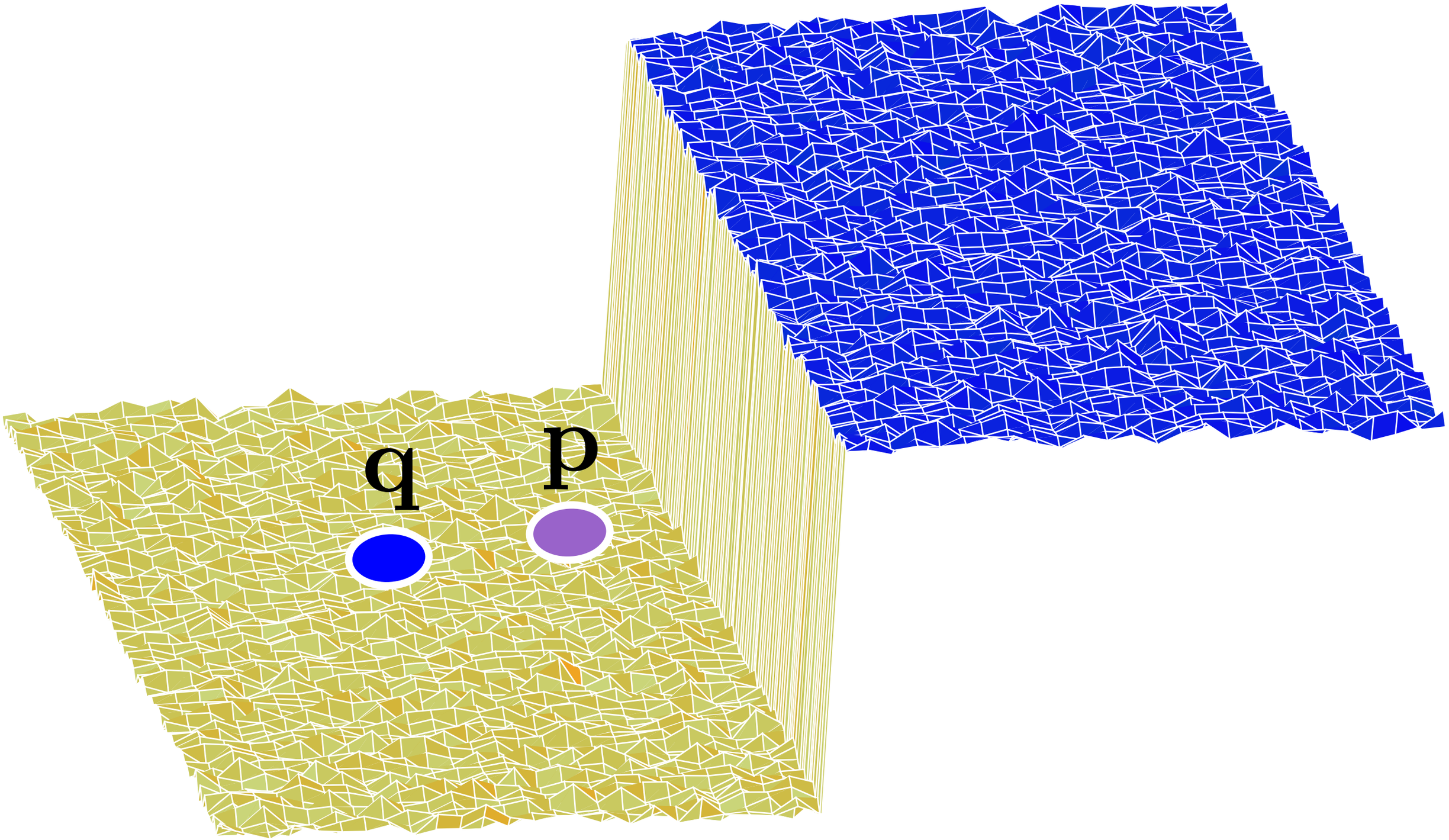
Bilateral



Filtering

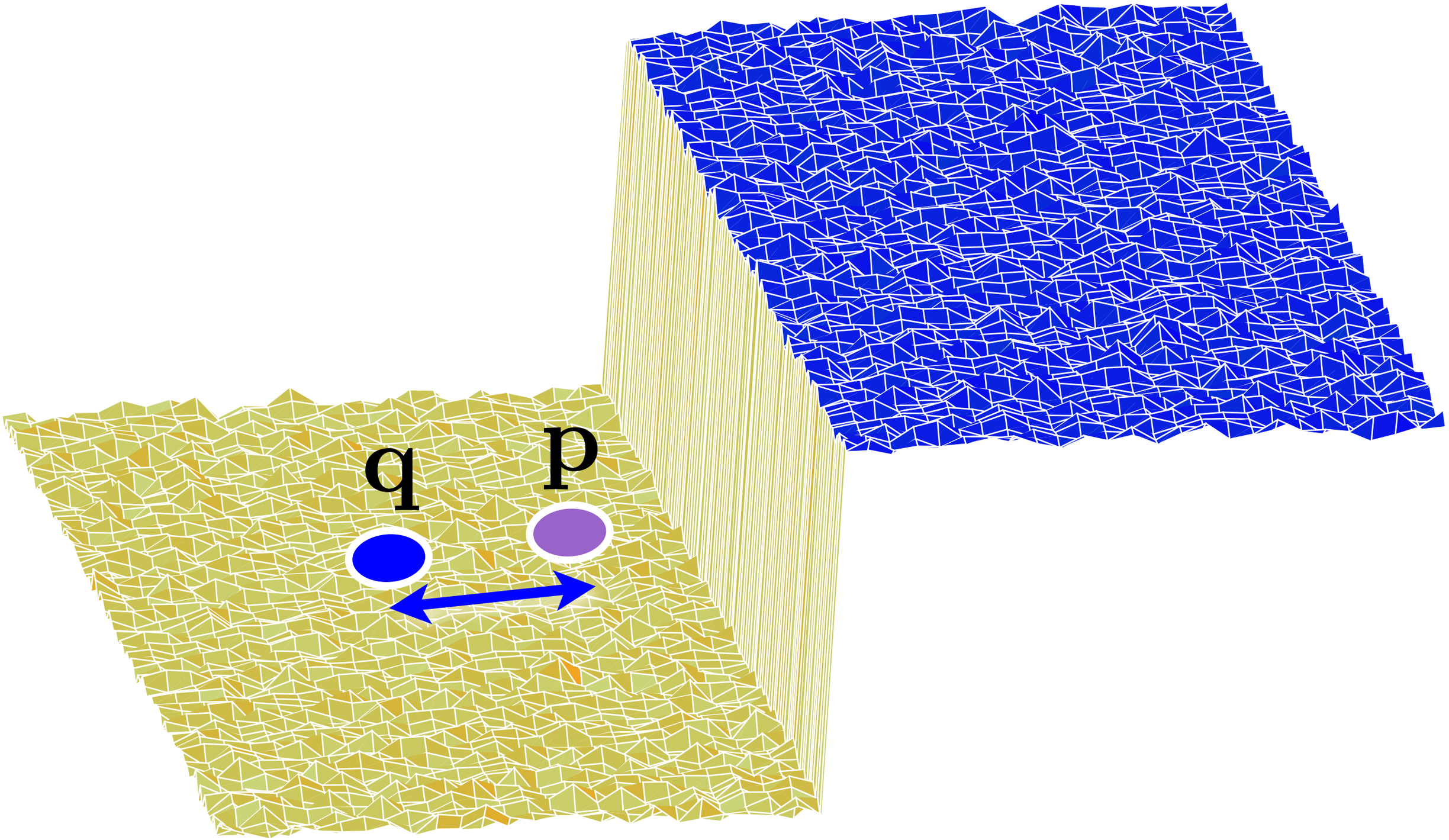
Gaussian smoothing

$$I'(\mathbf{p}) = \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, I(\mathbf{q})$$



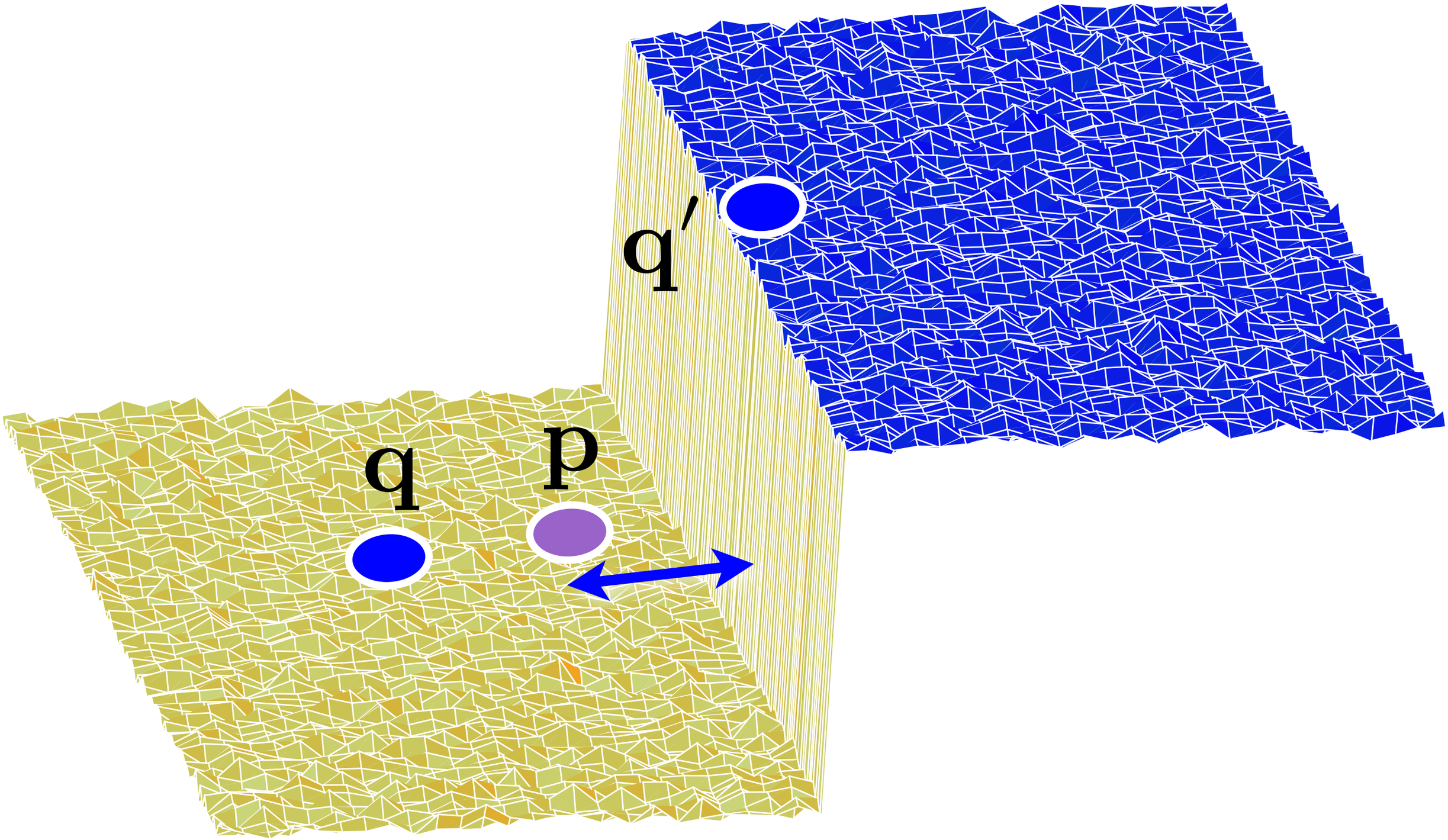
spatial weight

$$I'(\mathbf{p}) = \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, I(\mathbf{q})$$

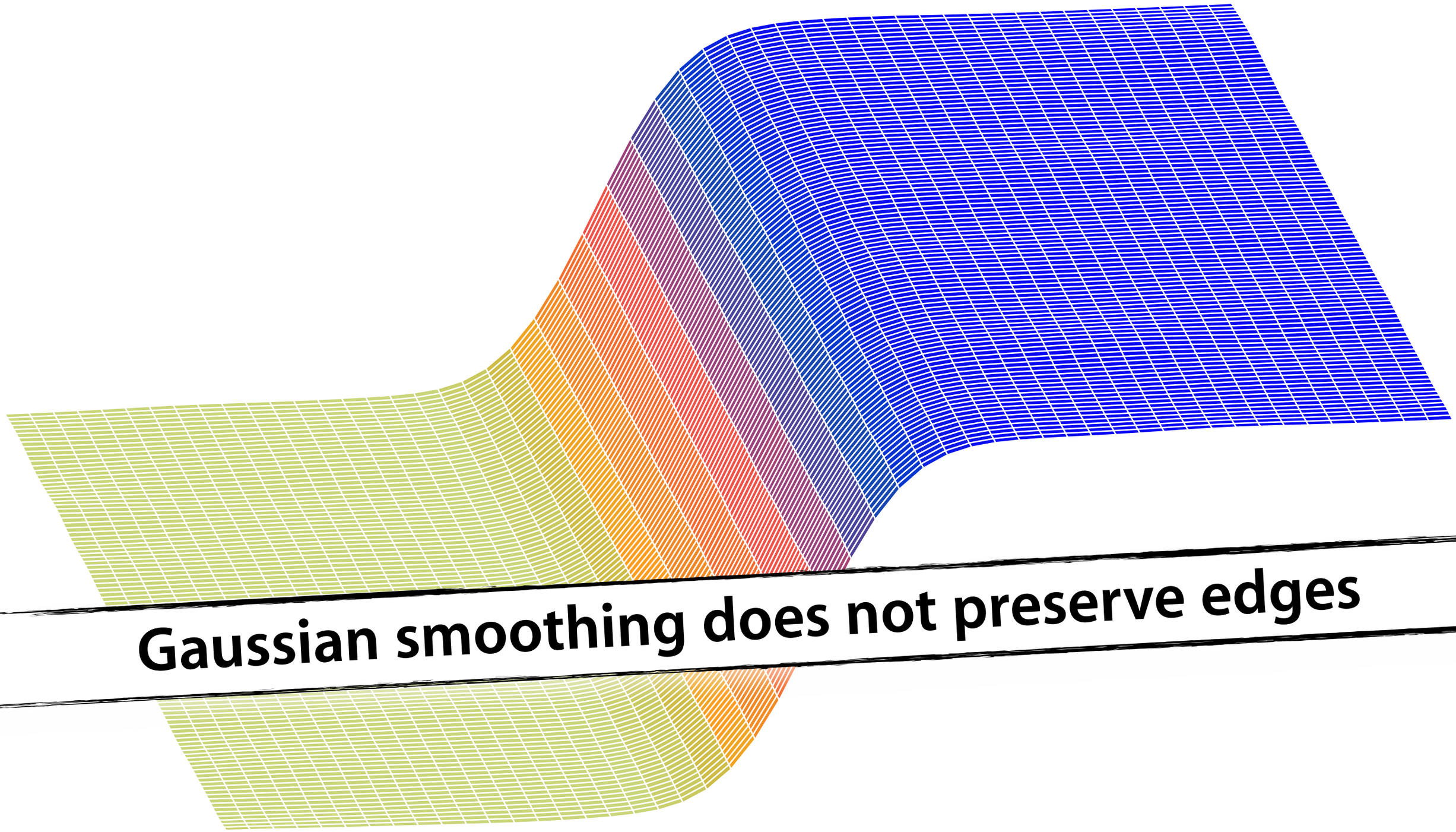


spatial weight

$$I'(\mathbf{p}) = \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, I(\mathbf{q})$$

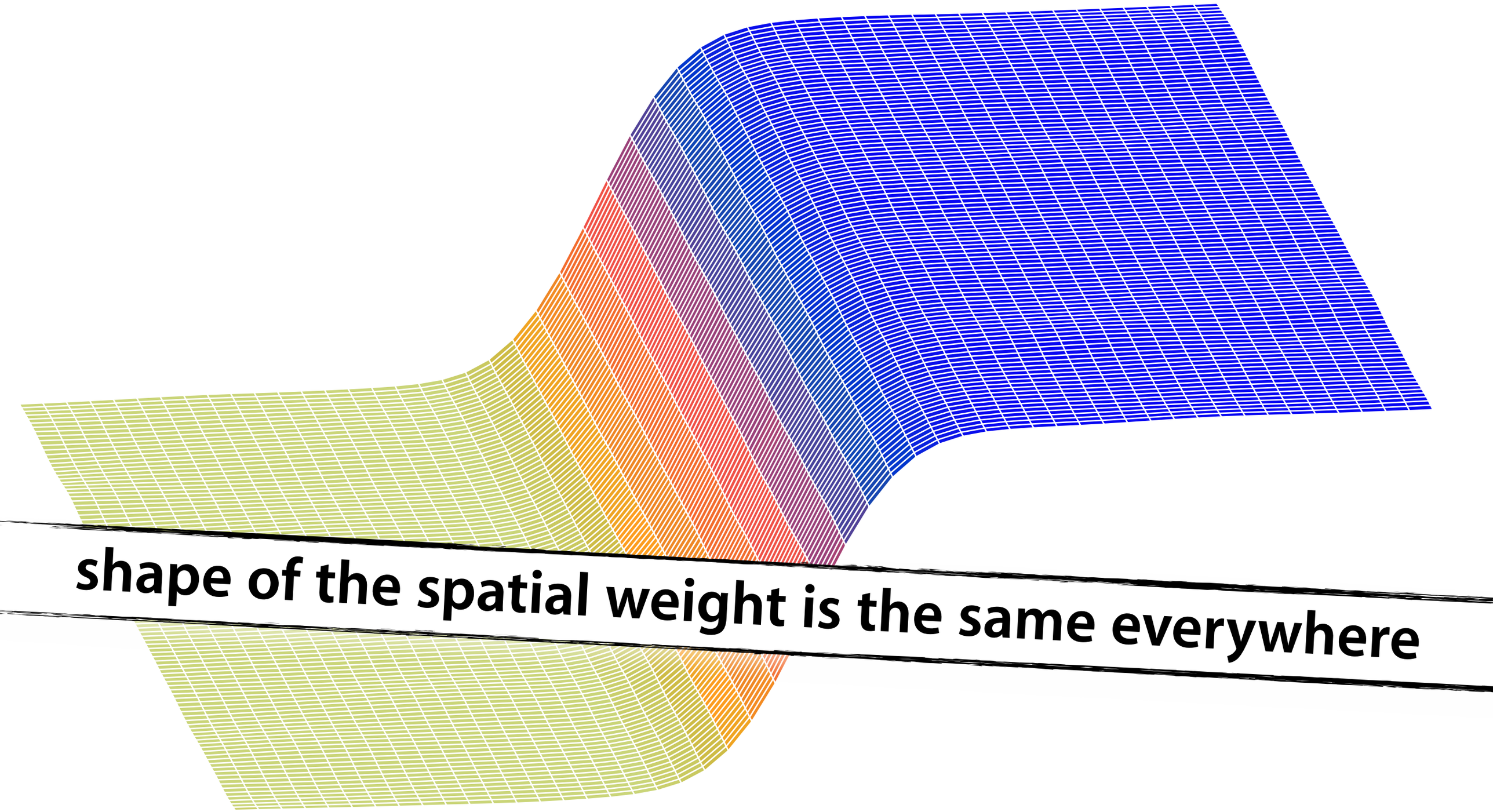


$$I'(\mathbf{p}) = \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, I(\mathbf{q})$$



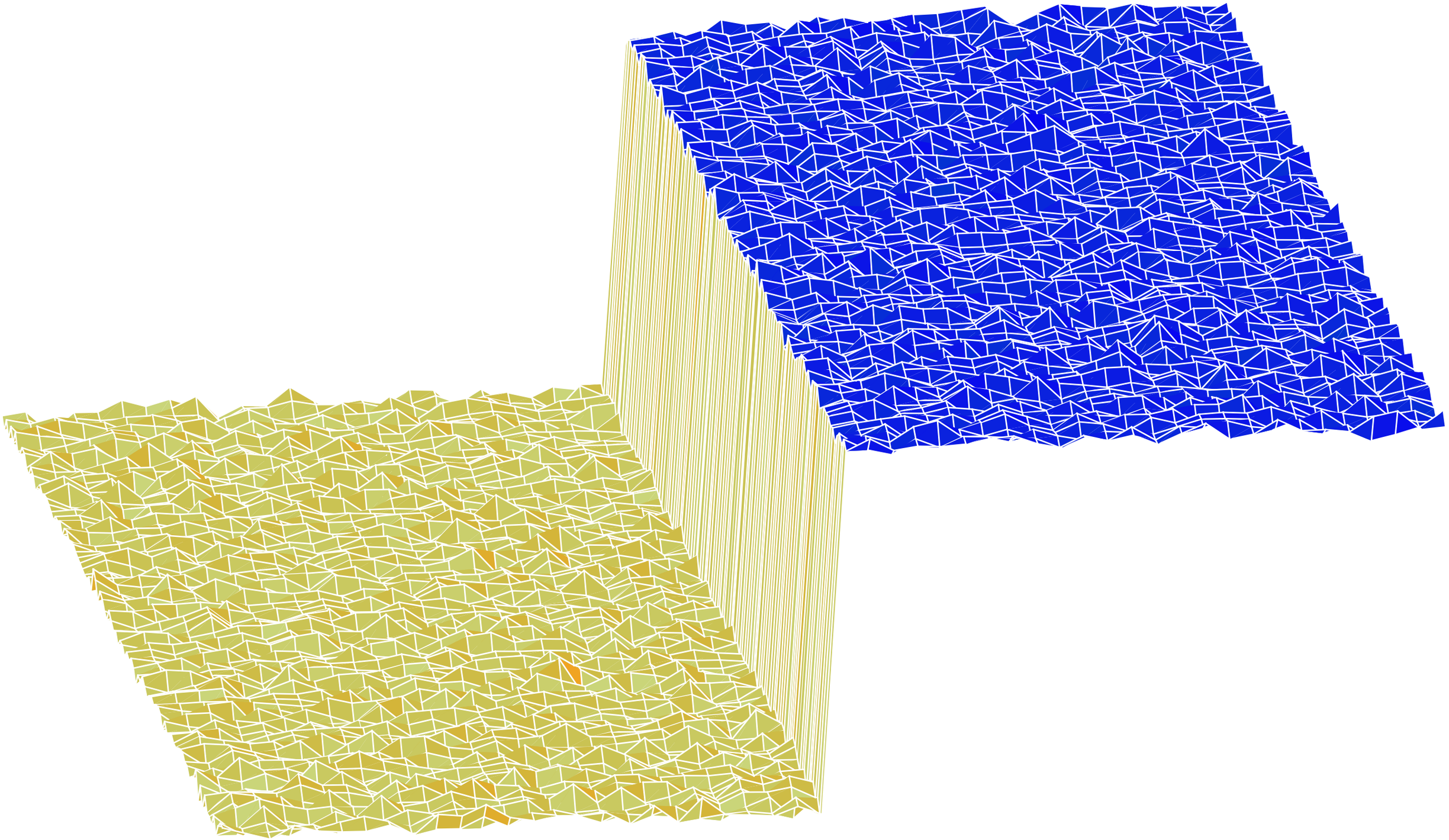
Gaussian smoothing does not preserve edges

$$I'(\mathbf{p}) = \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, I(\mathbf{q})$$

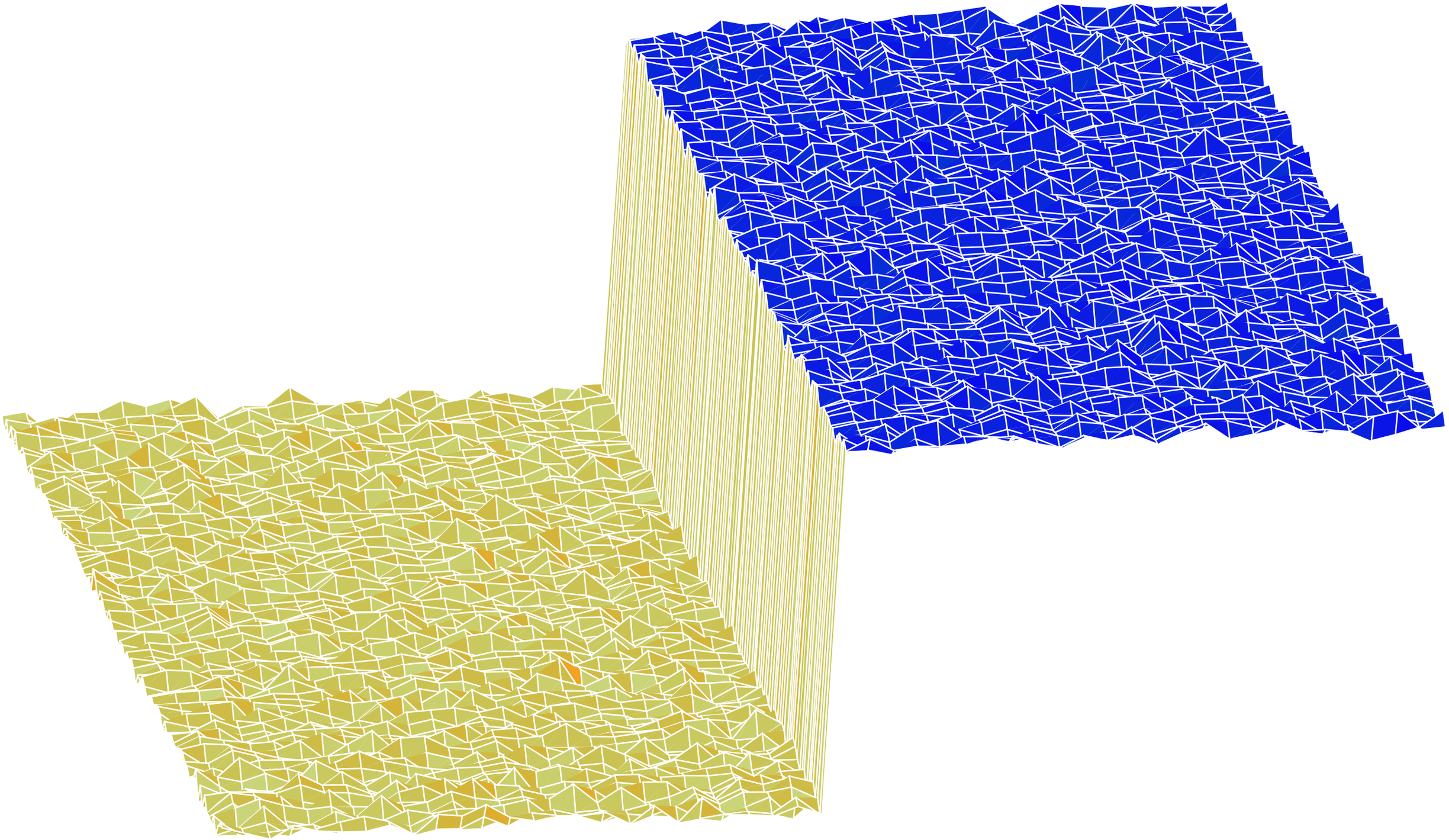


shape of the spatial weight is the same everywhere

$$I'(\mathbf{p}) = \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, I(\mathbf{q})$$

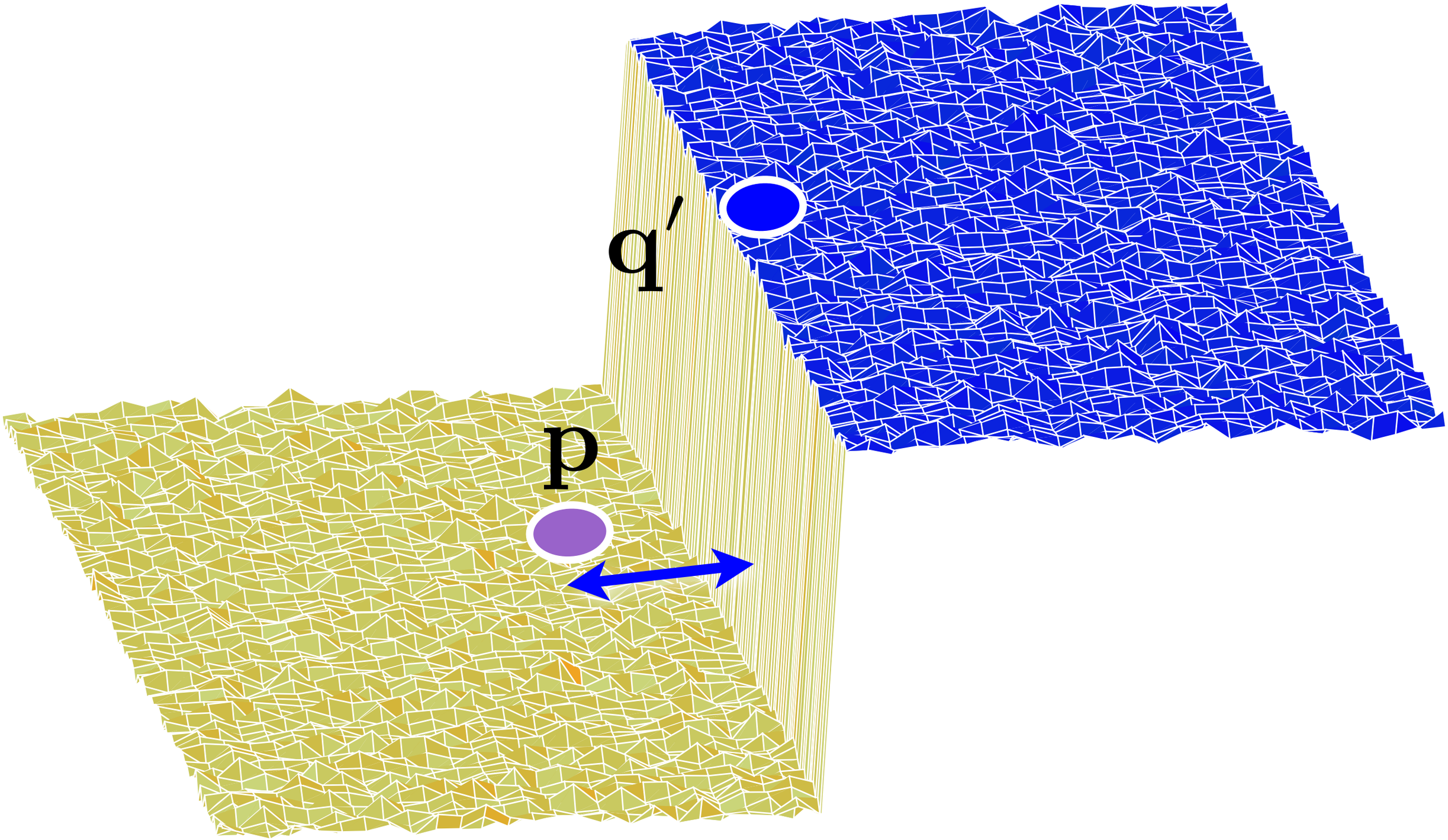


$$I'(\mathbf{p}) = w(\mathbf{p}) \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(\|I(\mathbf{p}) - I(\mathbf{q})\|)\, I(\mathbf{q})$$



spatial weight

$$I'(\mathbf{p}) = w(\mathbf{p}) \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(\|I(\mathbf{p}) - I(\mathbf{q})\|)\, I(\mathbf{q})$$

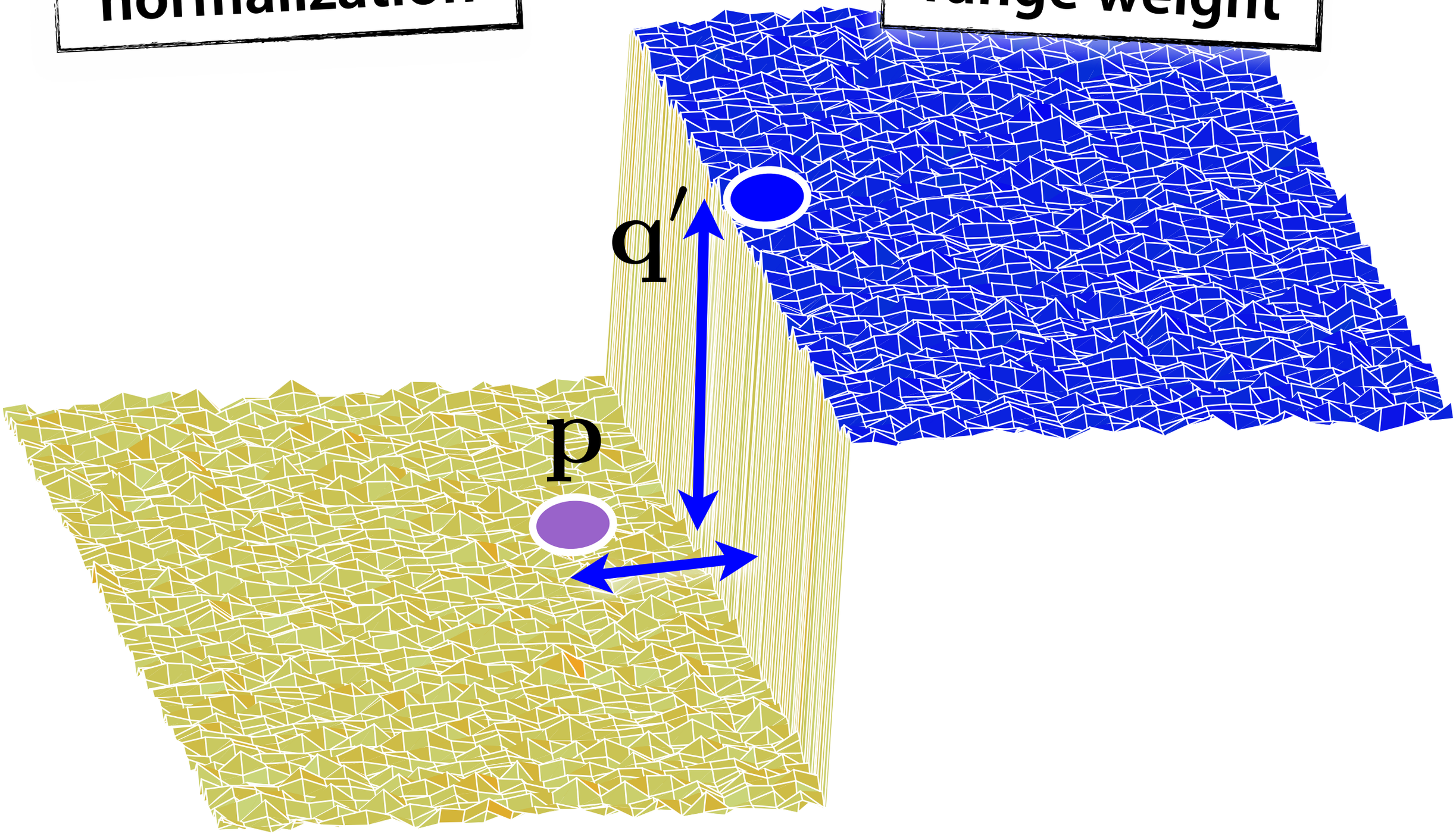


spatial weight

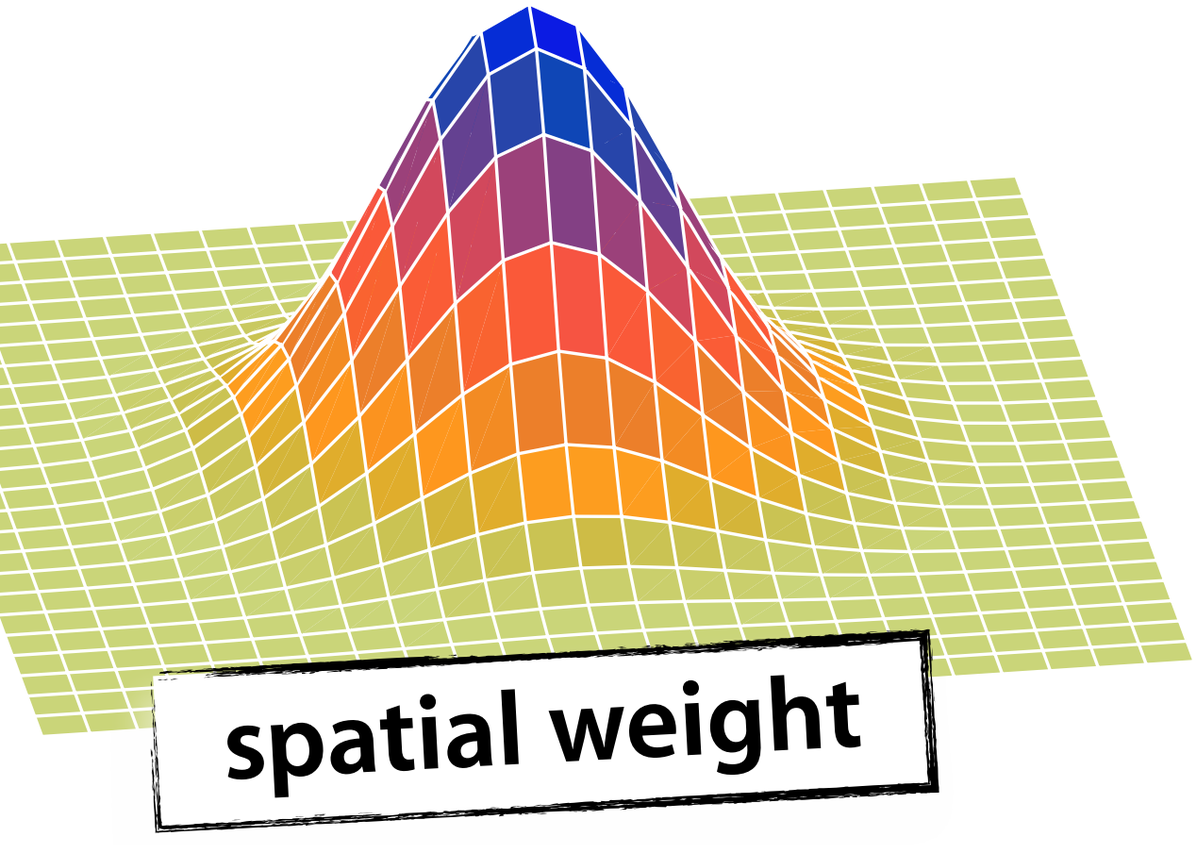
$$I'(\mathbf{p}) = w(\mathbf{p}) \sum G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(\|I(\mathbf{p})-I(\mathbf{q})\|)\, I(\mathbf{q})$$

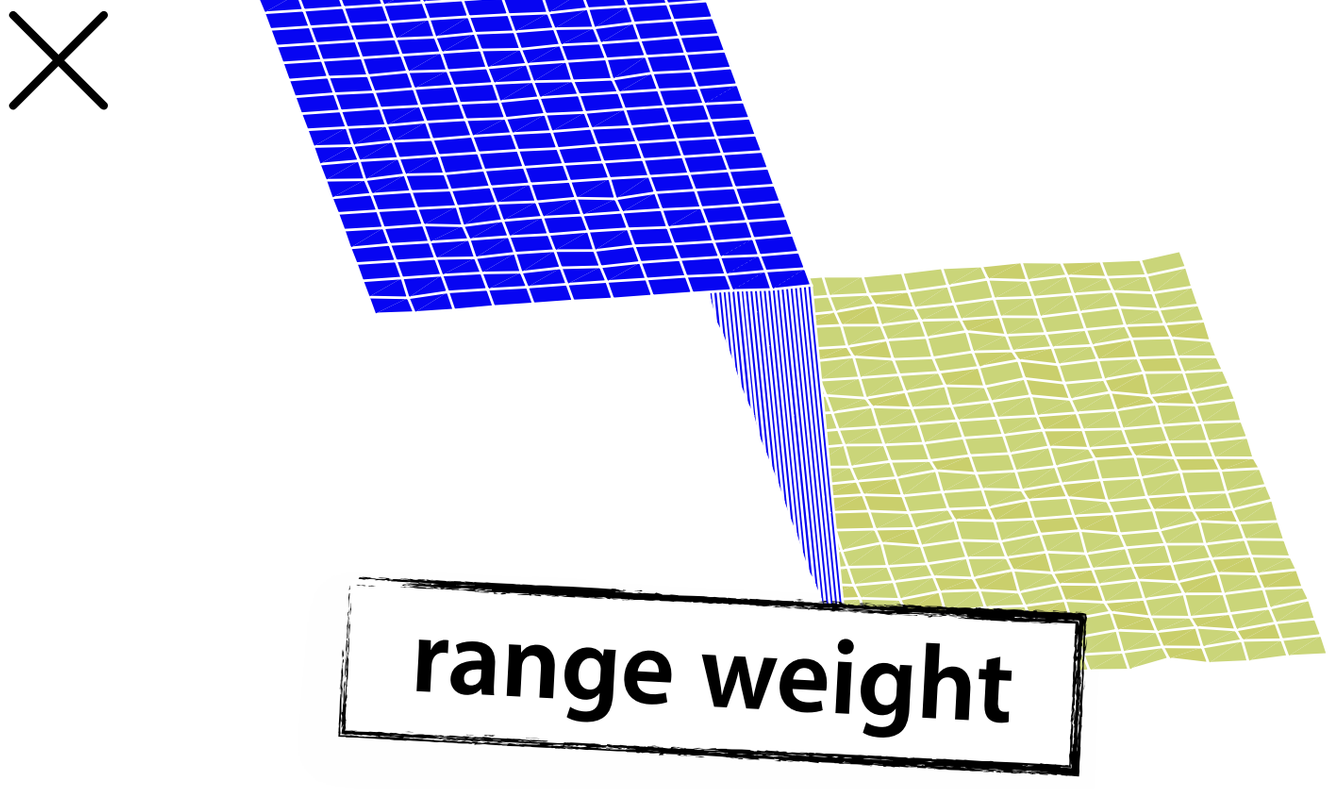
normalization

range weight

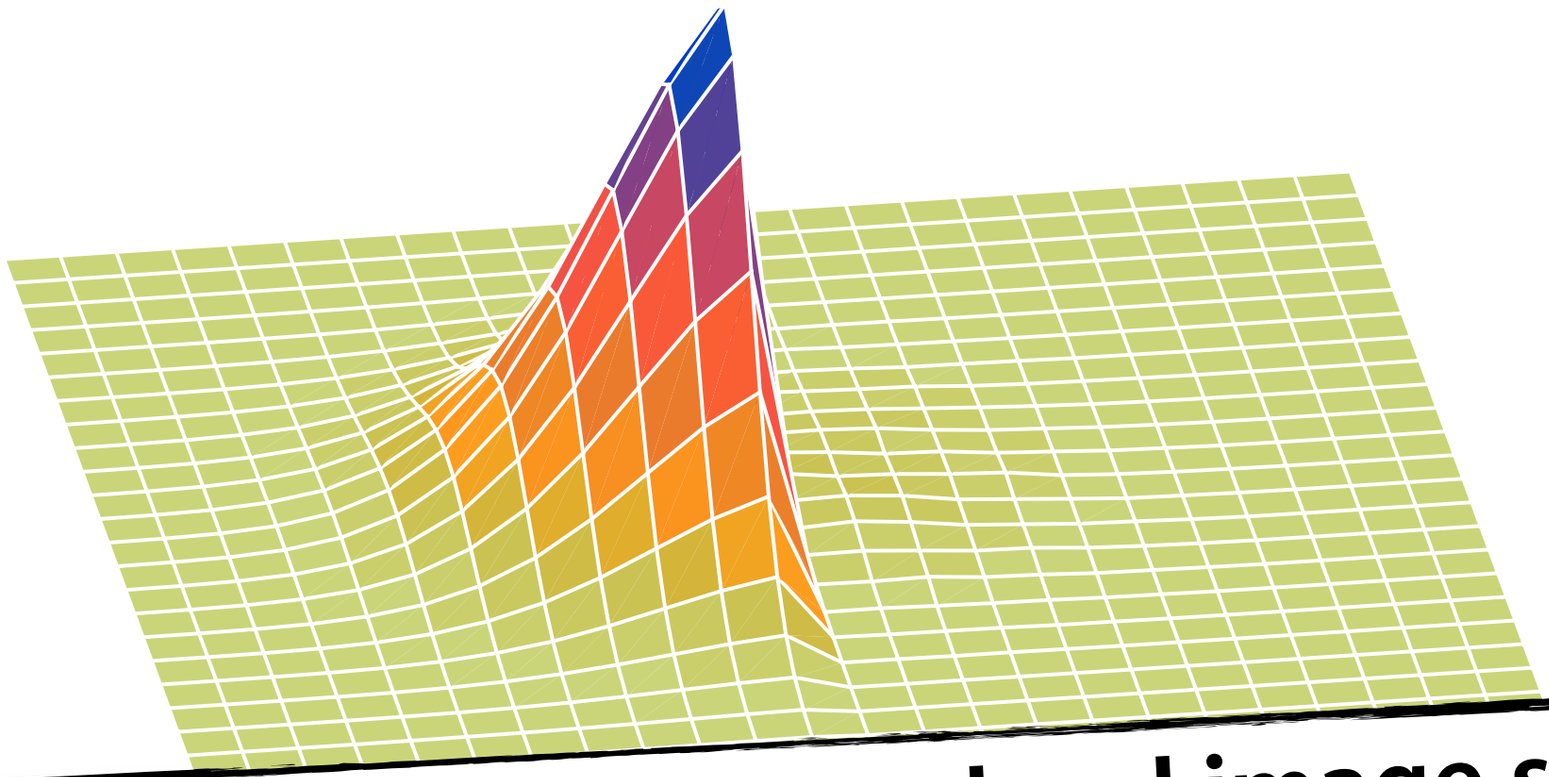


$$G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) \qquad G_{\sigma_r}(\|I(\mathbf{p})-I(\mathbf{q})\|)$$



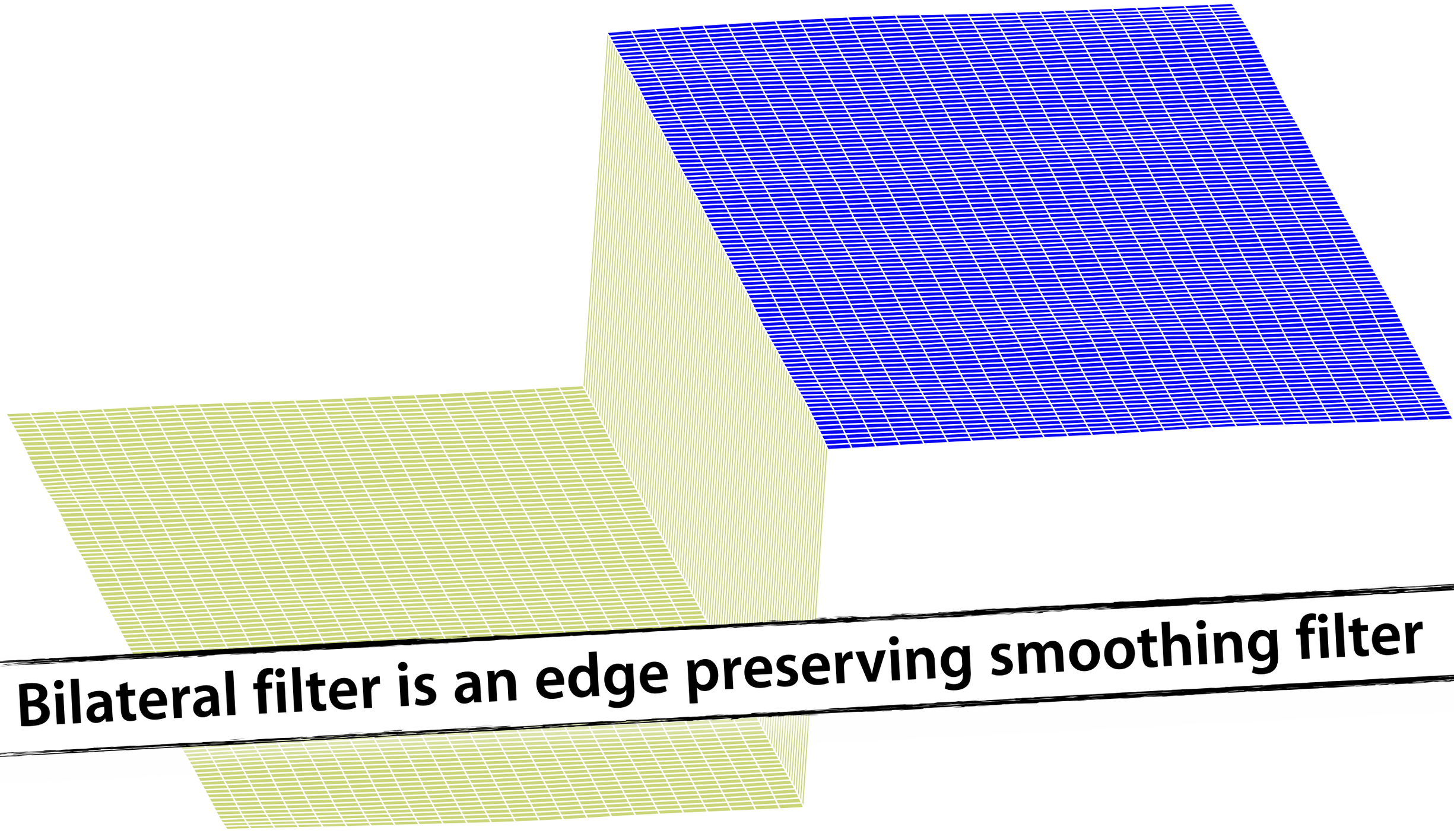


$$w(\mathbf{p})\,G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\,G_{\sigma_r}(\|I(\mathbf{p})-I(\mathbf{q})\|)$$



weight filter shape conforms to local image structure

$$I'(\mathbf{p}) = w(\mathbf{p}) \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(\|I(\mathbf{p}) - I(\mathbf{q})\|)\, I(\mathbf{q})$$



Bilateral filter is an edge preserving smoothing filter

Bilateral smoothing