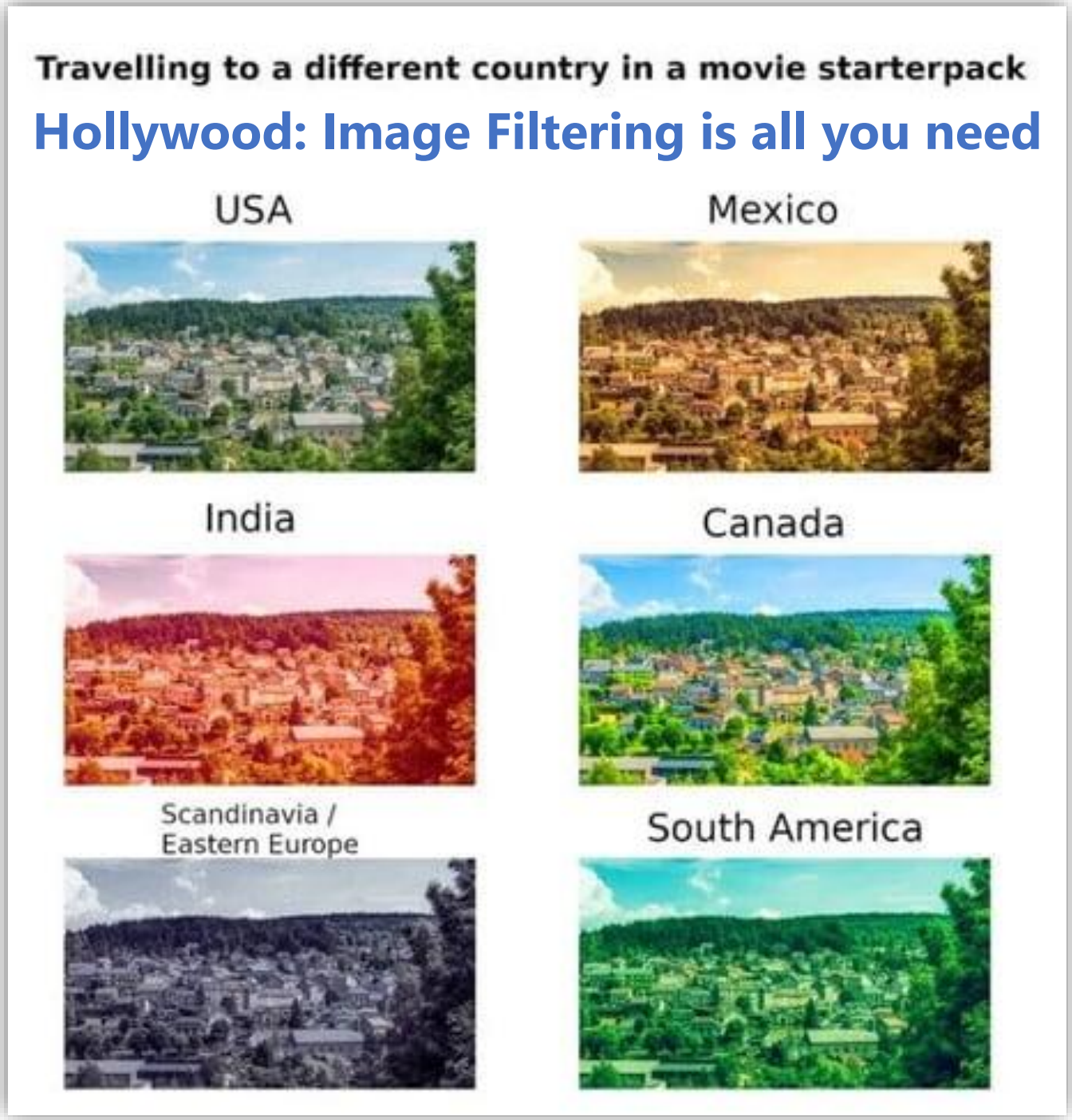
CMSC 472/672

Lecture 3

Image Filtering I: Convolution



Reminder: Course Staff

Instructor: Tejas Gokhale
Assistant Professor, CSEE

Wednesday 2:30 – 3:30 PM

ITE 342-B

gokhale@umbc.edu

Office Hours

TA: Yu Liu
Ph.D. Student (Research: Computer Graphics)

Monday 2:30 – 3:30 PM
& Tuesday 1 – 2 PM

ITE 340

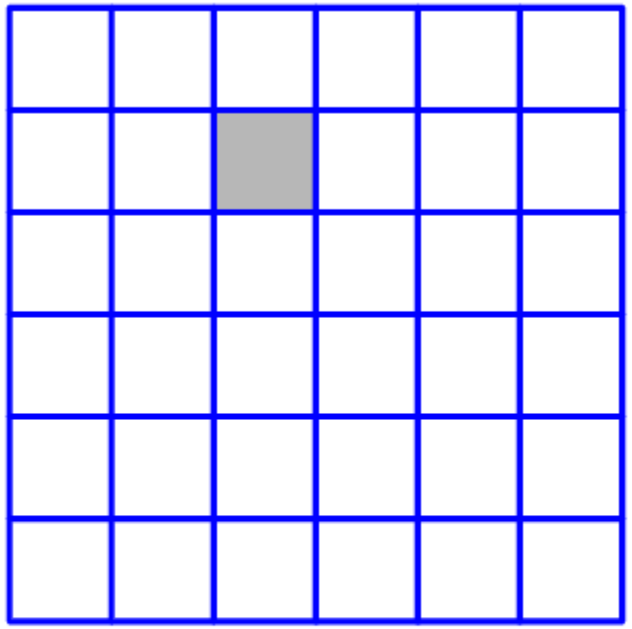
yul2@umbc.edu

Reminder: Scribing Signup and Project Teams

A	B	C	D	E	F
SCRIBING: Sign up for (at least) one scribing slot					
Date	Day	Scribe 1	Scribe 2	Scribe 3	DUE DATE
27-Aug	W		NO SCRIBING		3-Sep
1-Sep	M		NO SCRIBING		8-Sep
3-Sep	W	Jason Rojas	Joe Kim	Shivanand	10-Sep
8-Sep	M	Garvita Jain	Alvaro Molina	Sebastian Vilchis	15-Sep
10-Sep	W	Olivia Kaniecki	Joey Mule	Prajita Shrestha	17-Sep
15-Sep	M	Raymond Z.	Bemnet D.	Hallel Dereb	22-Sep
17-Sep	W	Bhavya Sri Sang	Ziwei Zhang		24-Sep
22-Sep	M	Thomas Getach	Rusham Bhatt	Nathan Dayie	29-Sep
24-Sep	W	Rajarshi Basak		Nabin Poudel	1-Oct
29-Sep	M				6-Oct
1-Oct	W	Jossie Batisso			8-Oct
6-Oct	M	Rahul Badwaik			13-Oct
8-Oct	W				15-Oct
13-Oct	M				20-Oct
15-Oct	W				22-Oct
20-Oct	M		NO SCRIBING		
22-Oct	W		NO SCRIBING		
27-Oct	M				3-Nov
29-Oct	W				5-Nov
3-Nov	M				10-Nov
5-Nov	W				12-Nov
10-Nov	M				17-Nov
12-Nov	W				19-Nov
17-Nov	M				24-Nov
19-Nov	W				26-Nov
24-Nov	M		NO SCRIBING		

H	I	J	K	L	M	N	O
PROJECT: List Your Project Teams Here (you can name your team too!)							
Team Name	Member 1	Member 2	Member 3	Member 4	Member 5	**Notes from Tejas**	
1	Raymond Zh	Thomas Get	Joe Kim			could think of adding another member?	
2	Nathaniel Therrien					approved solo project	
3	Parker Kopp	Olivia Kaniec	(looking for another member)				
4							
5							
6							
7							
8							
9							
10							
PROJECT: Available Students							
If you are looking for team members, put your name below and reach out to others							
Sebastian vilchis							
Alvaro Molina							
Jason Rojas, UNDERGRAD							

An image is a matrix of pixels

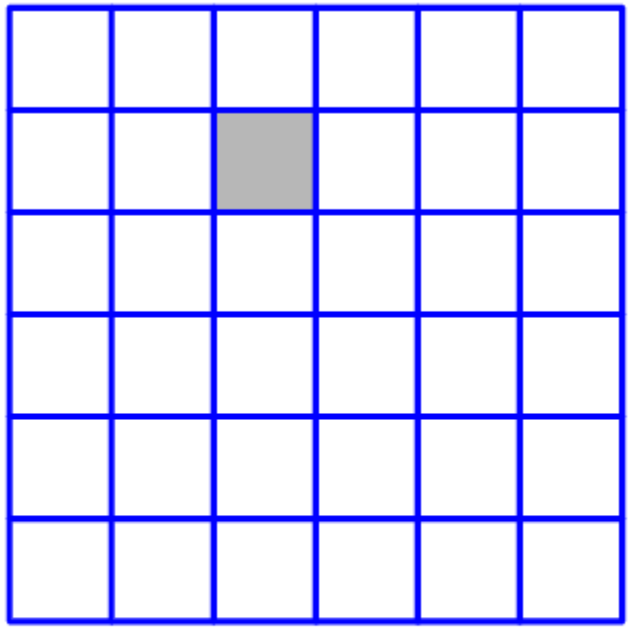


F[x,y]

An array of numbers (“pixels”)

x,y are integer column/row indices

An image is a matrix of pixels



F[x,y]

An array of numbers (“pixels”)

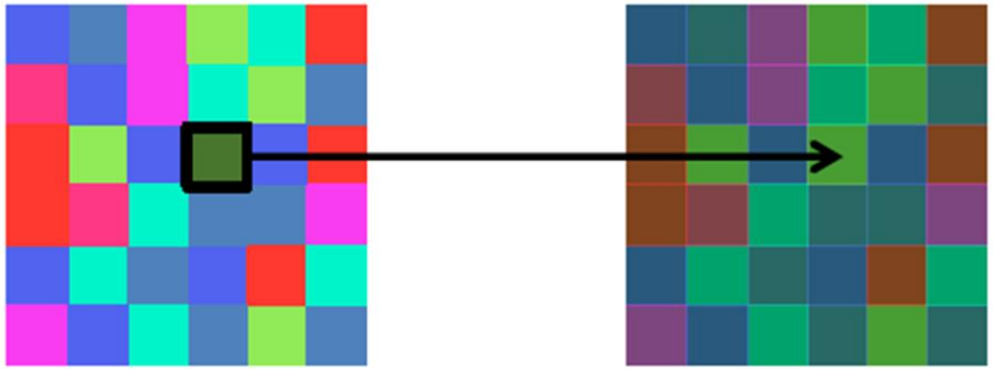
x,y are integer column/row indices

Is every matrix an image?

(This is an open-ended question in HW1)

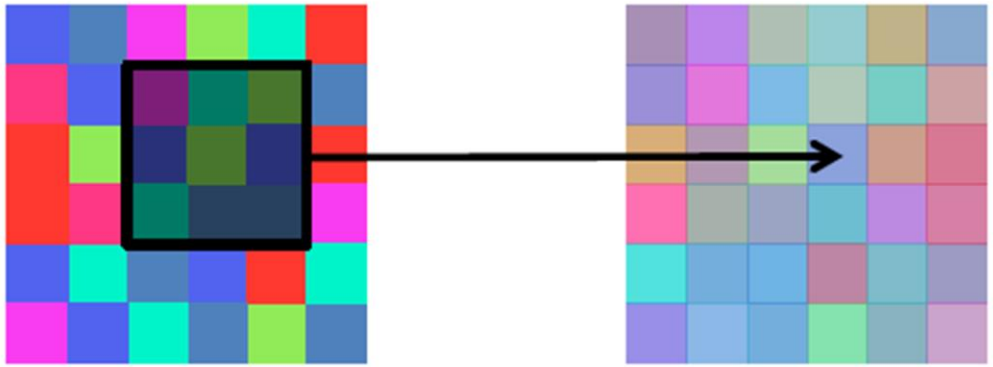
Point Processing vs Image Filtering

Point Operation



point processing

Neighborhood Operation



“filtering”

How would you implement these?

Examples of point processing

original

x

darken

$x - 128$

lower contrast

$\frac{x}{2}$

non-linear lower contrast

$\left(\frac{x}{255}\right)^{1/3} \times 255$

invert

$255 - x$

lighten

$x + 128$

raise contrast

$x \times 2$

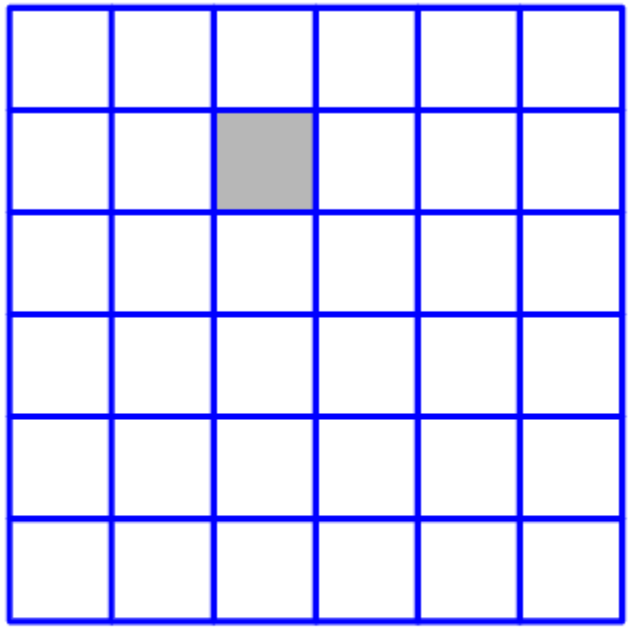
non-linear raise contrast

$\left(\frac{x}{255}\right)^{2} \times 255$

Python Implementation of Pointwise Operations

https://colab.research.google.com/drive/1Fk-L2PadlRusOkXcovqouZxI-IhHA6kJ?usp=sharing

An image is a matrix of pixels

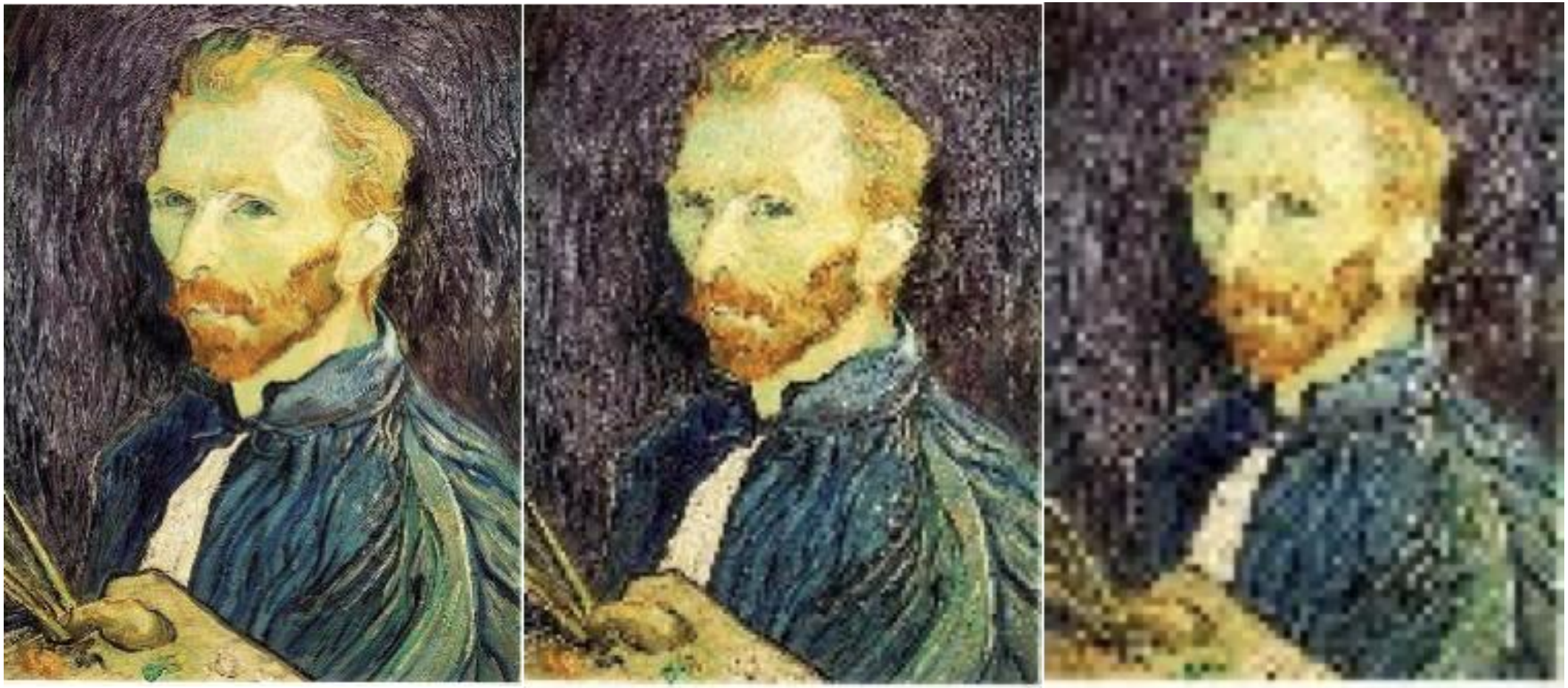


F[x,y]

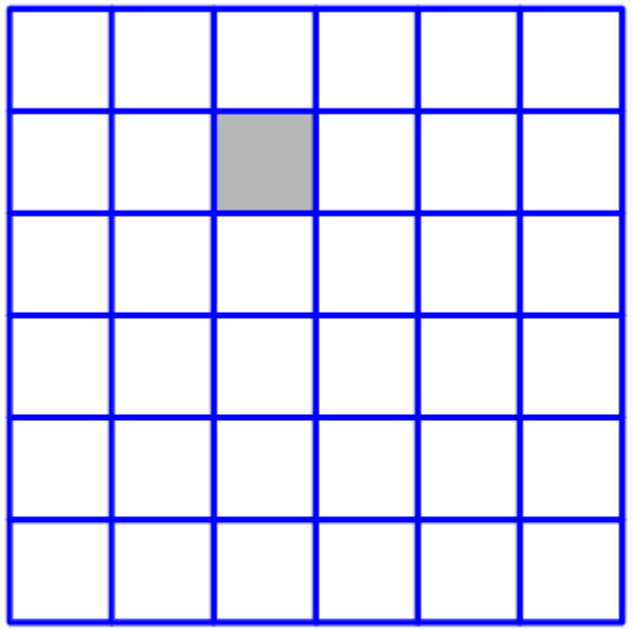
An array of numbers (“pixels”)

x,y are integer column/row indices

Subsampling: How would you achieve this?



1/2 1/4 (2x zoom) 1/8 (4x zoom)

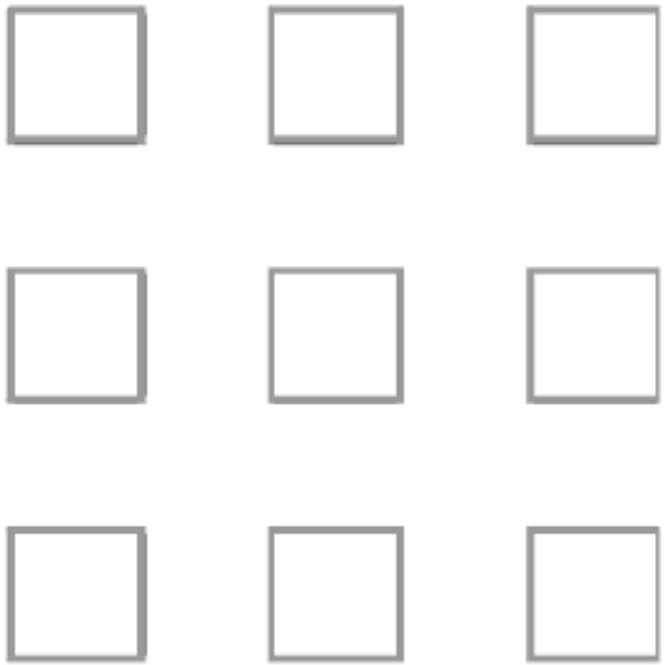


F[x,y]

An array of numbers (“pixels”)

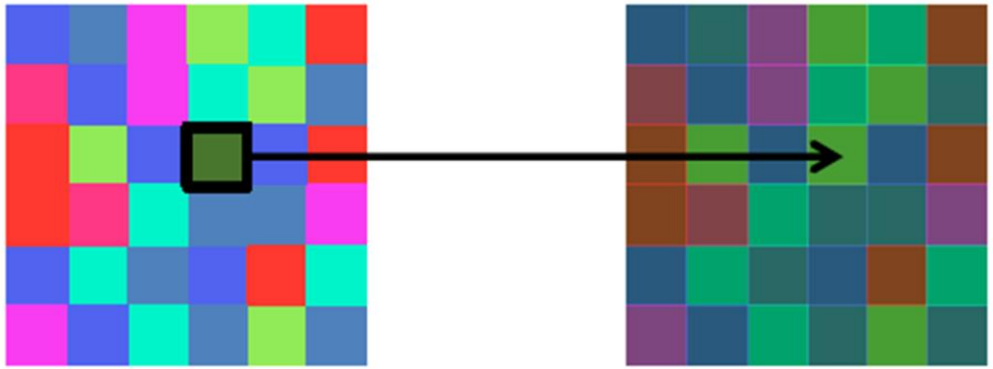
x,y are integer column/row indices

Throw away even rows and columns



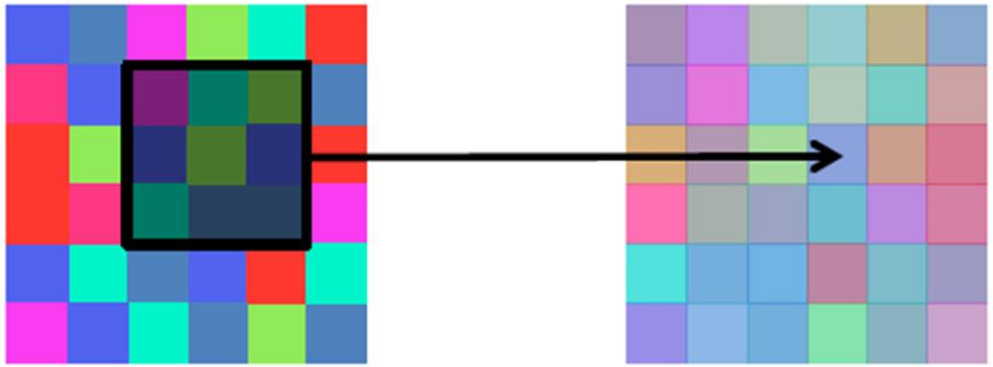
Point Processing vs Image Filtering

Point Operation



point processing

Neighborhood Operation



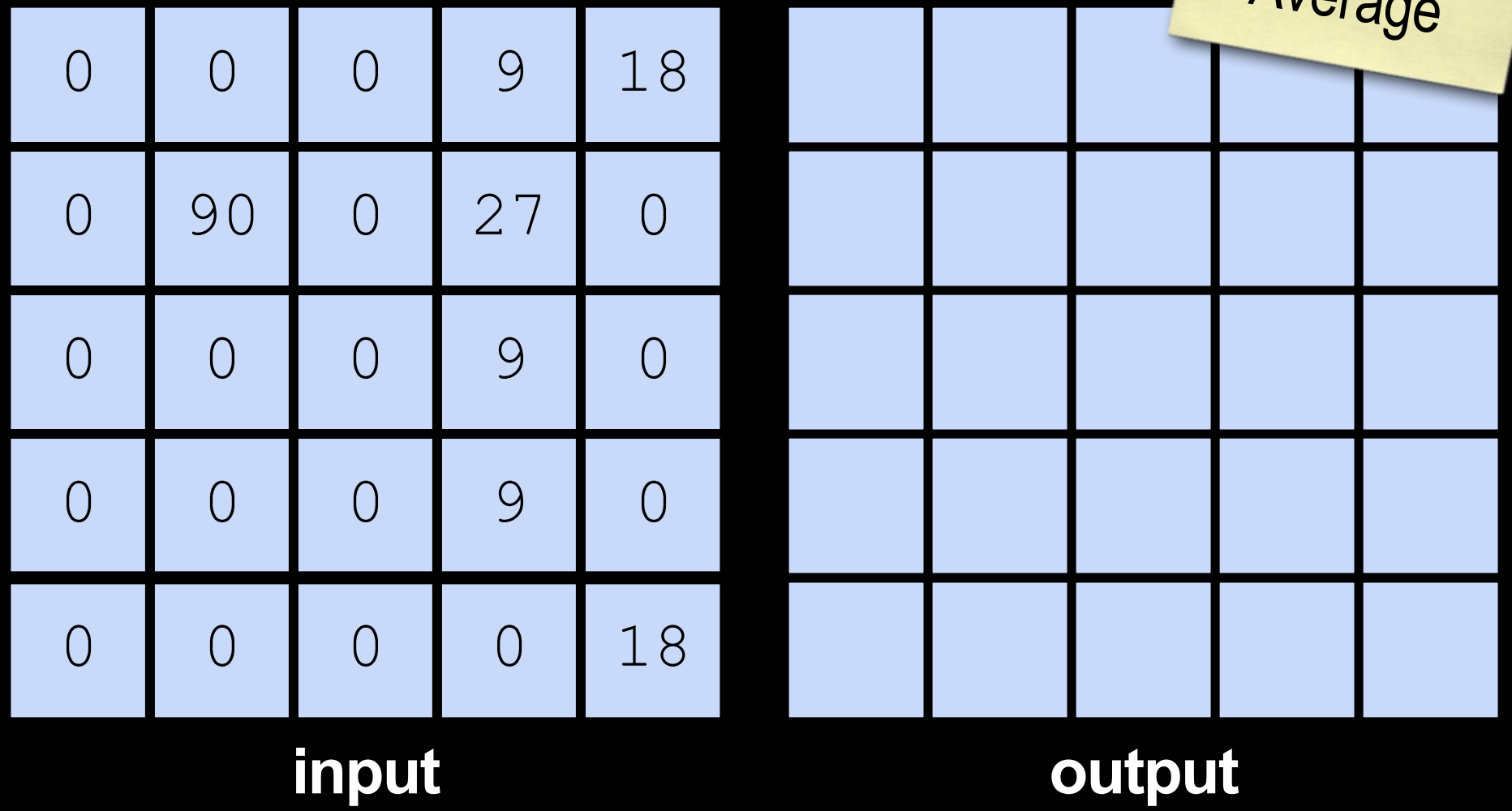
“filtering”

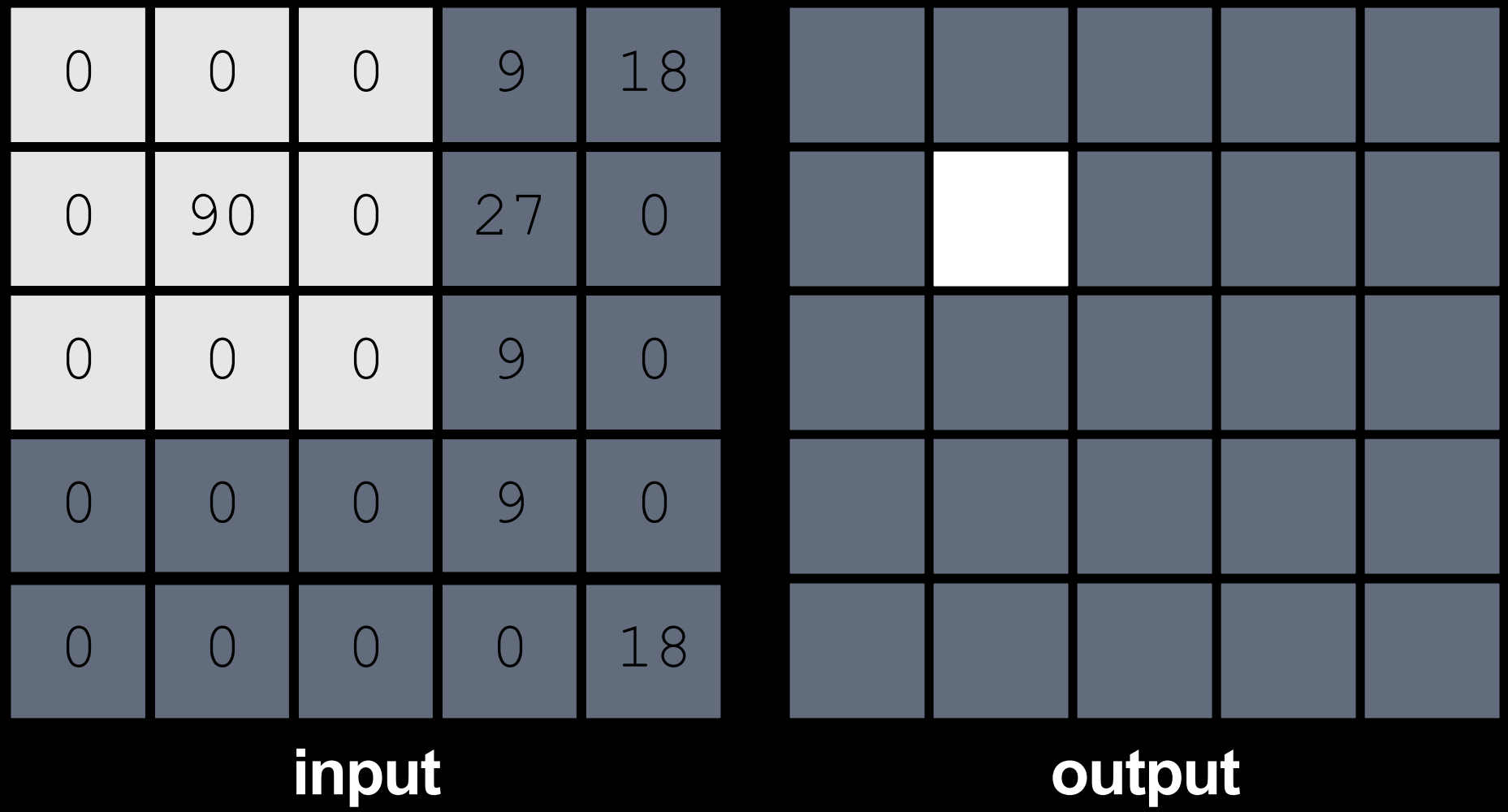
Example of Filtering: “Average Filter”

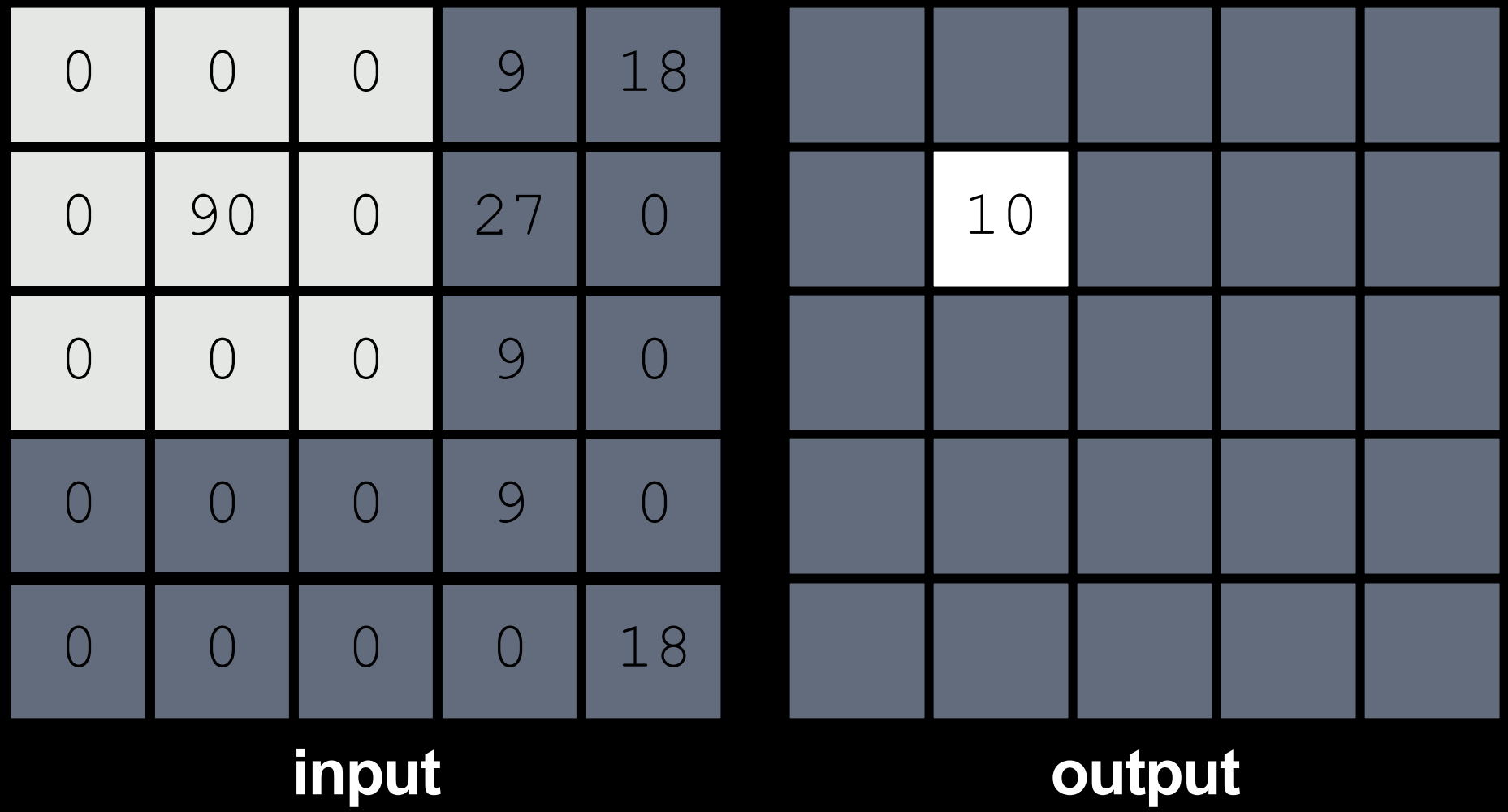
2D Average

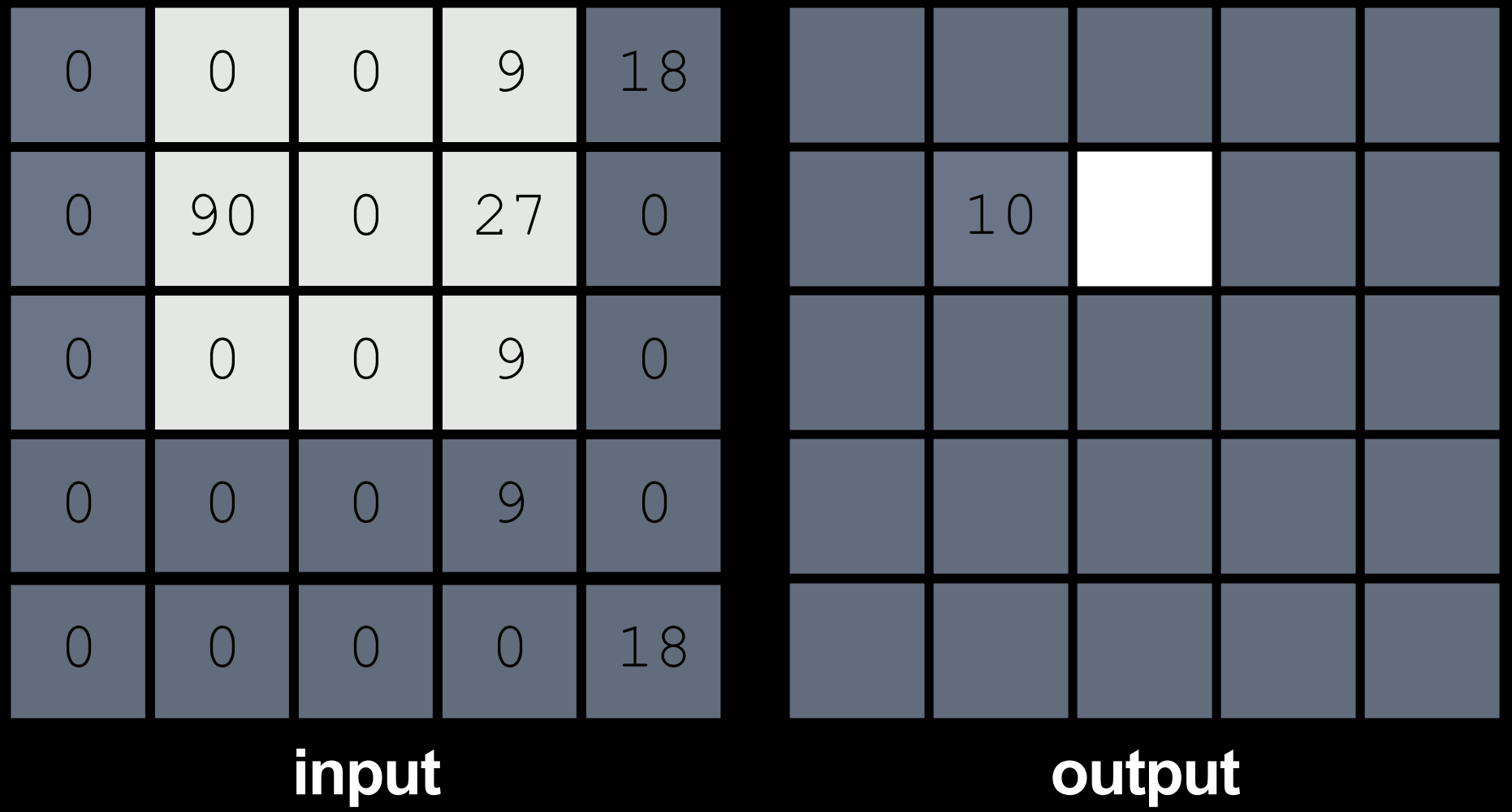
0	0	0	9	18
0	90	0	27	0
0	0	0	9	0
0	0	0	9	0
0	0	0	0	18

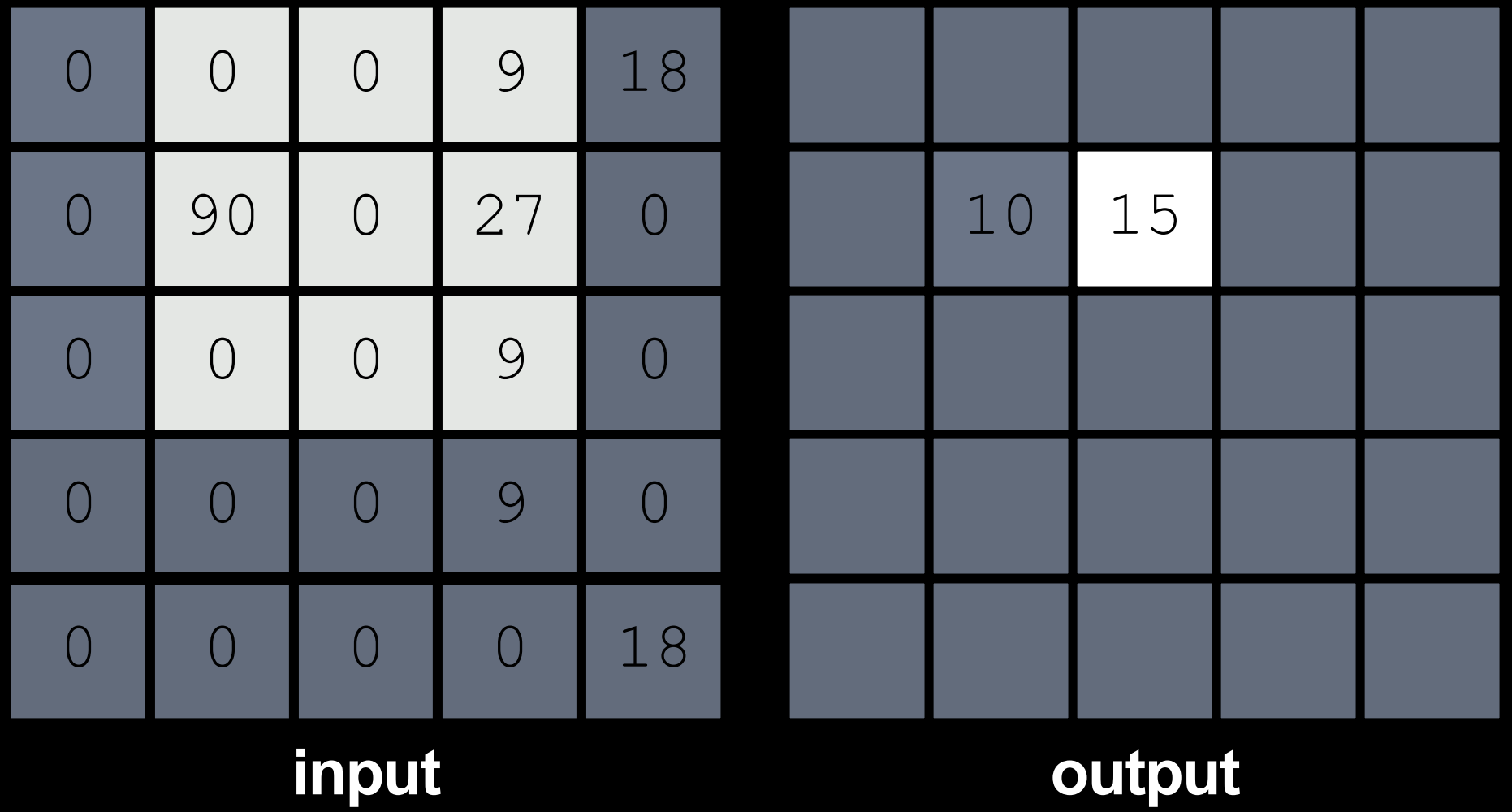
input

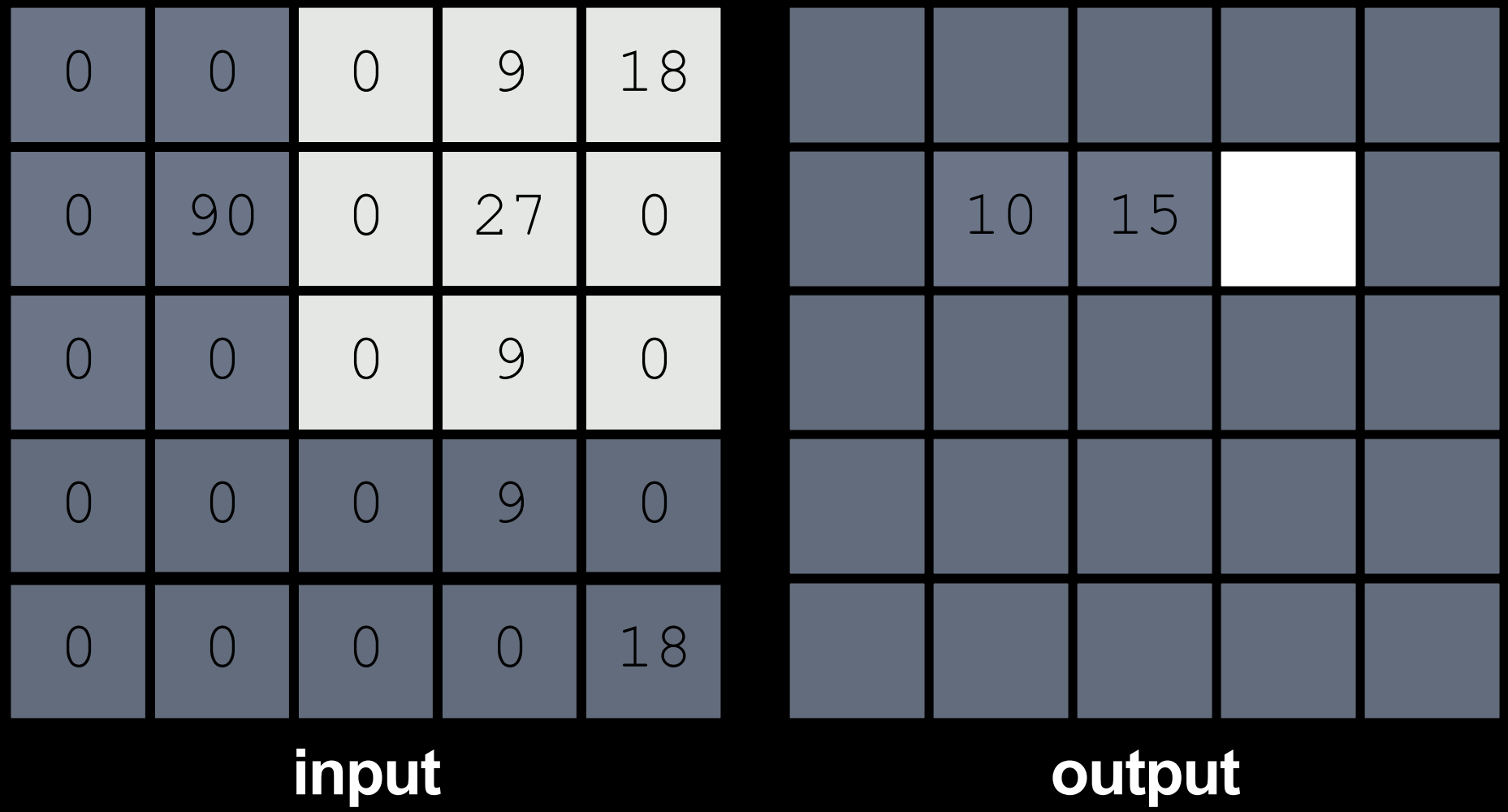

2D
Average
0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
output

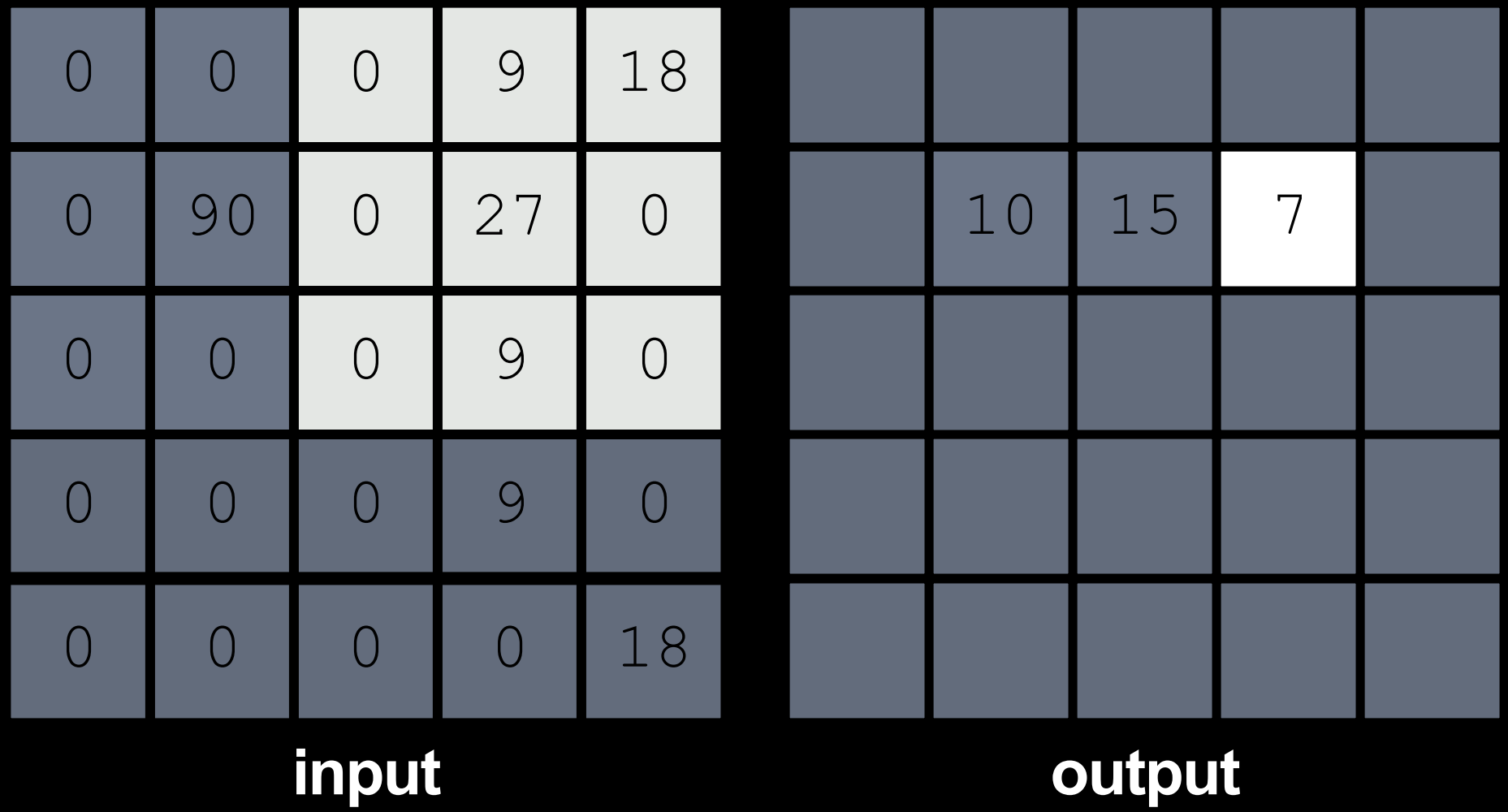

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
output

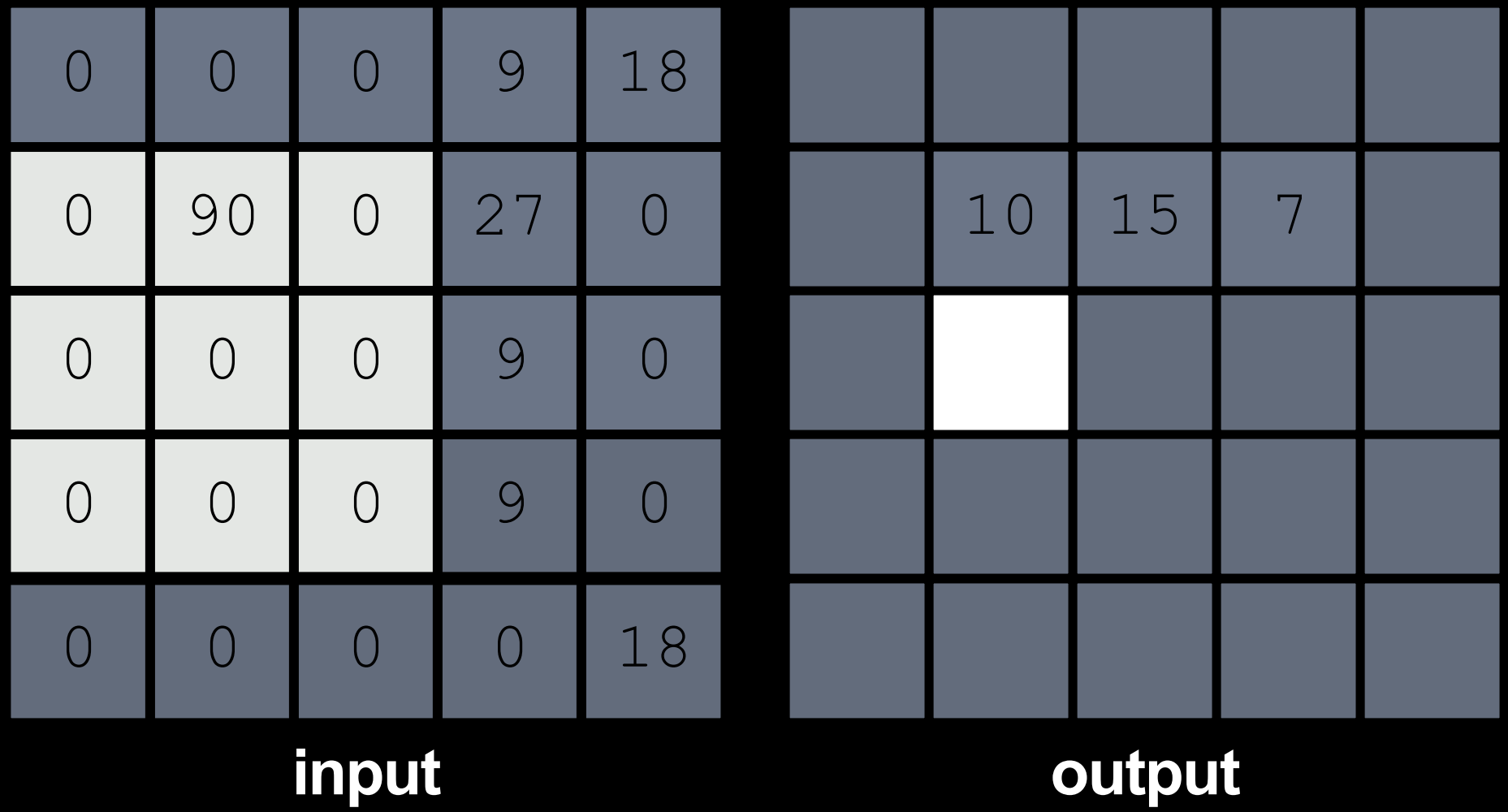

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
10
output

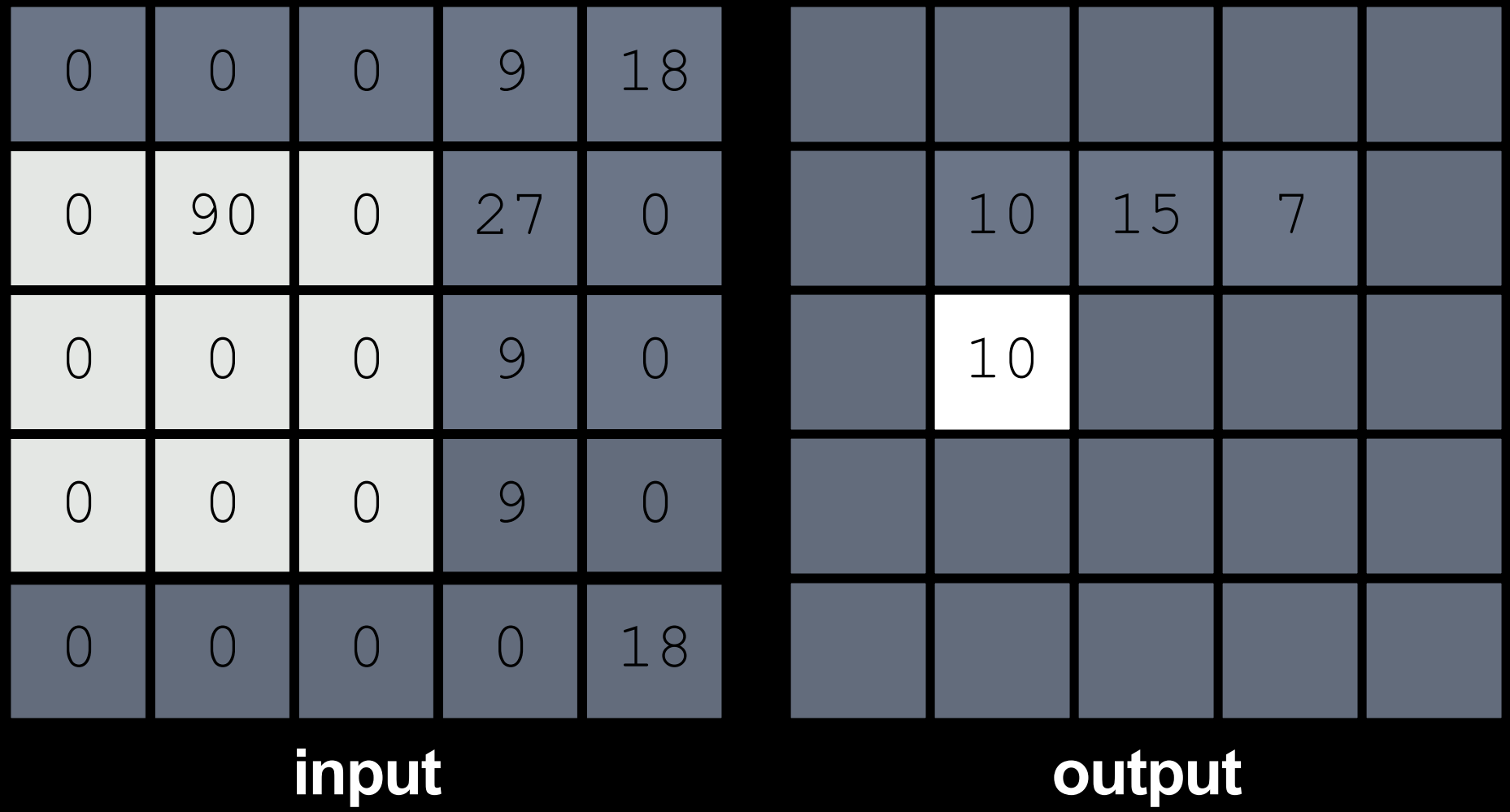

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
10
output

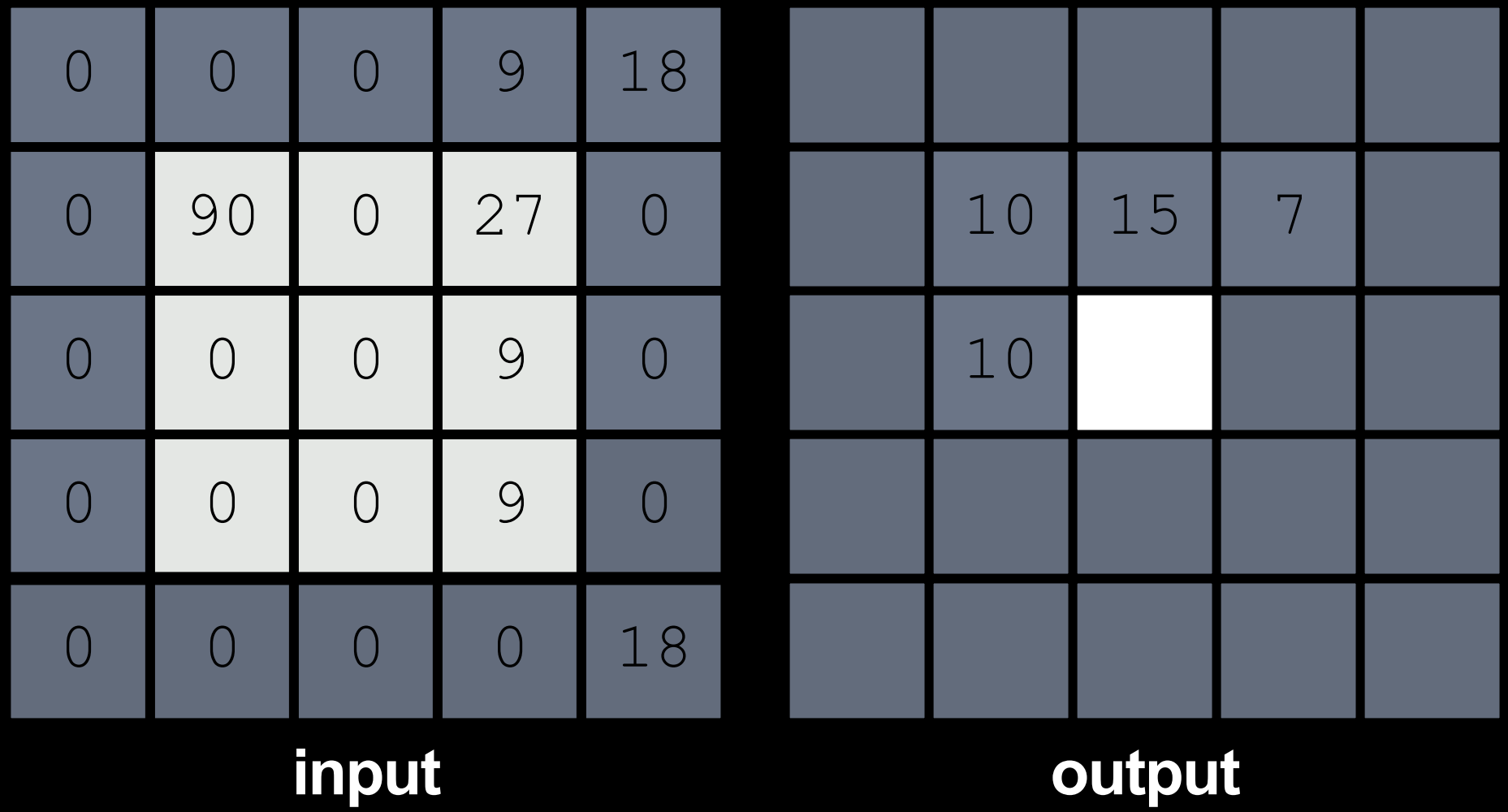

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
10 15
output

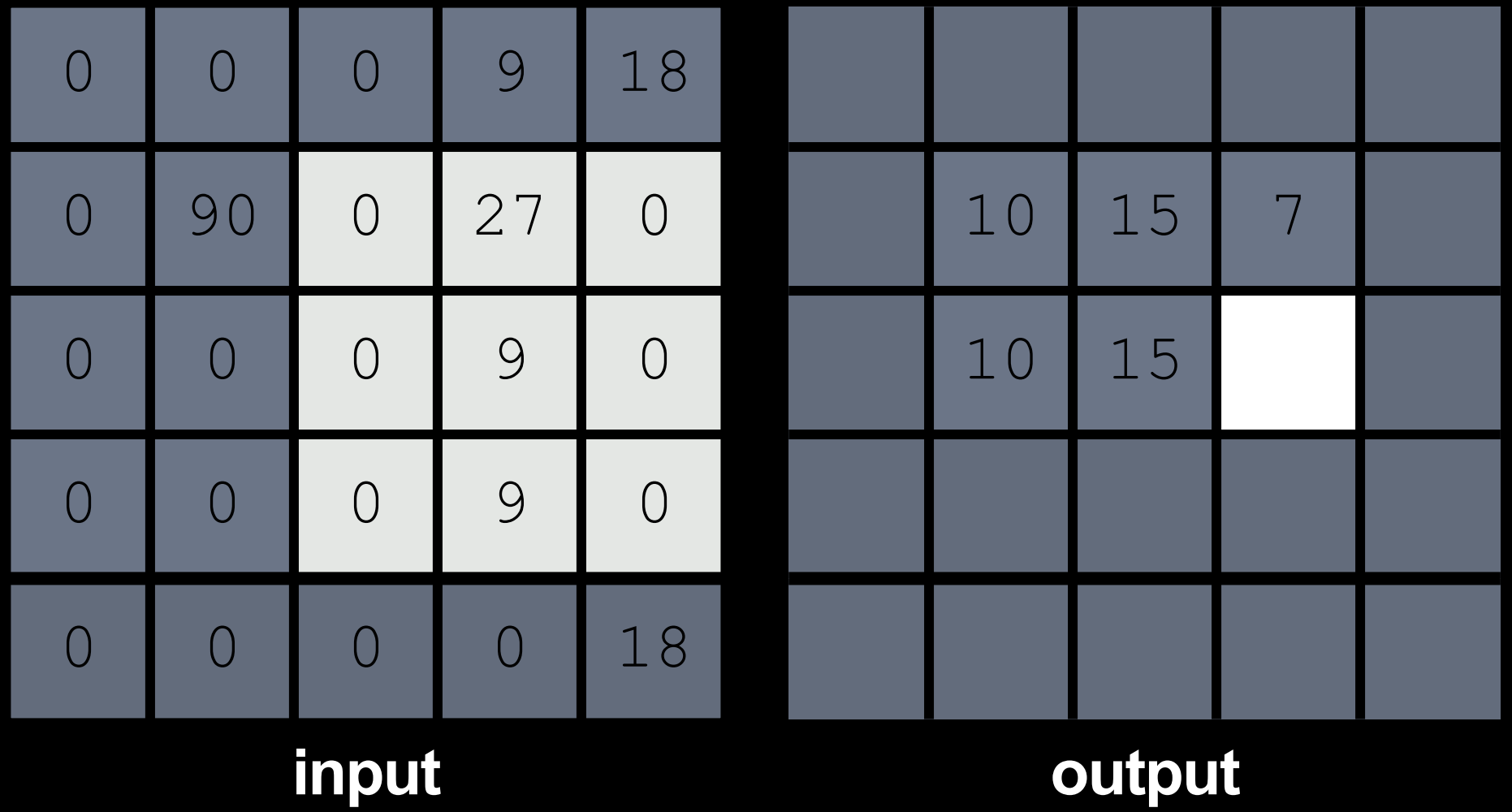

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
10 15
input
output

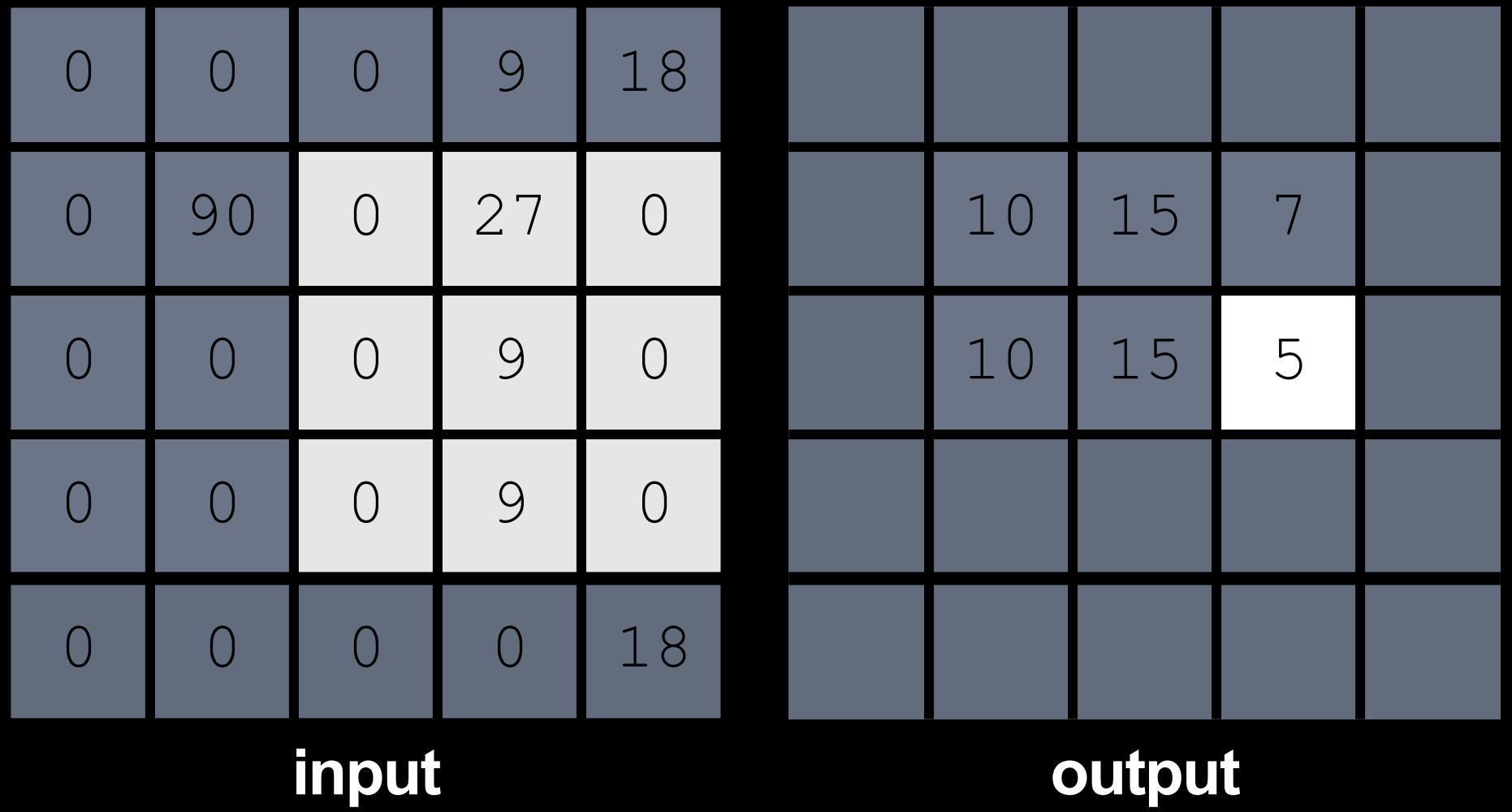

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
10 15 7
output

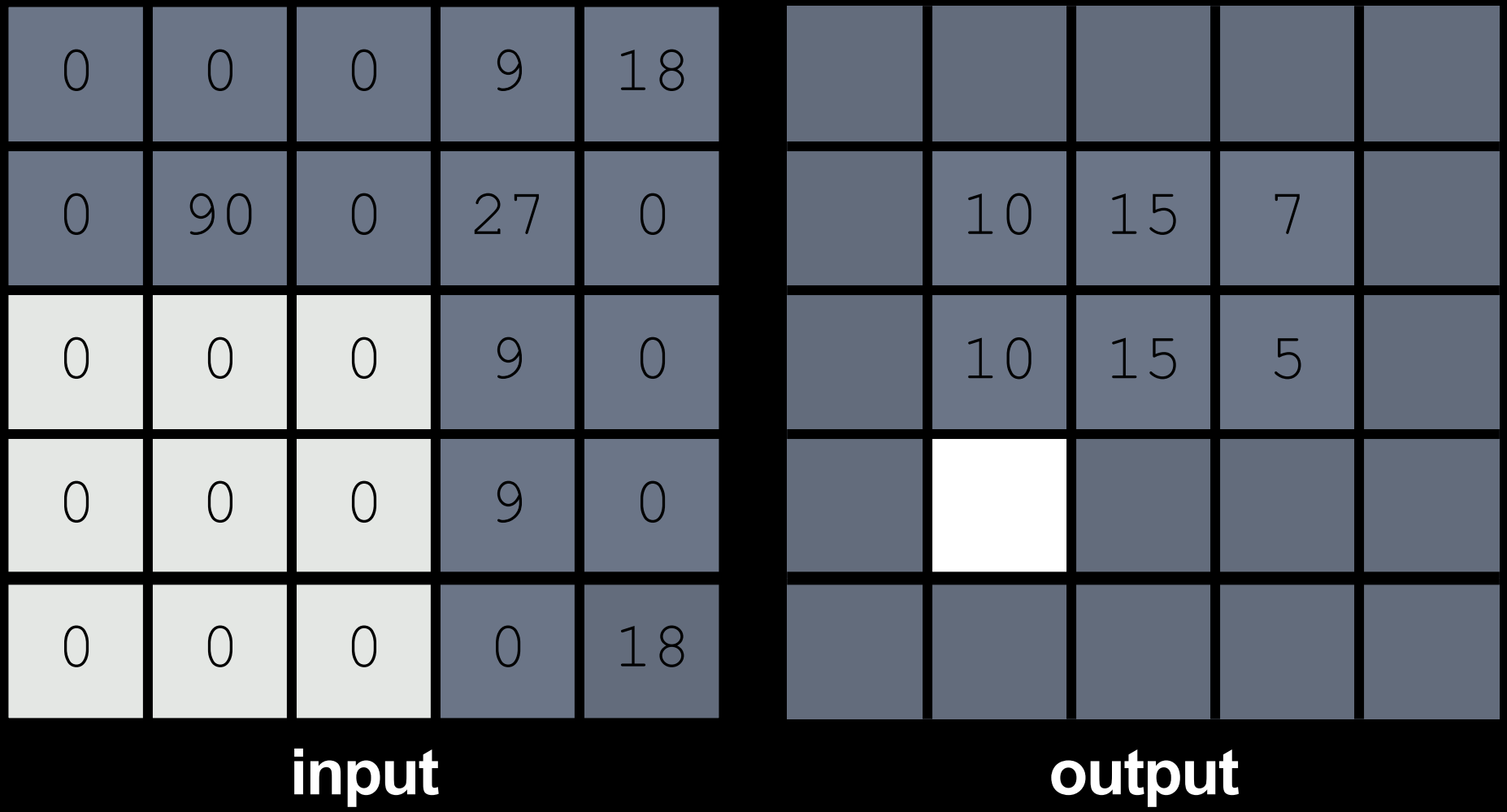

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
10 15 7
output

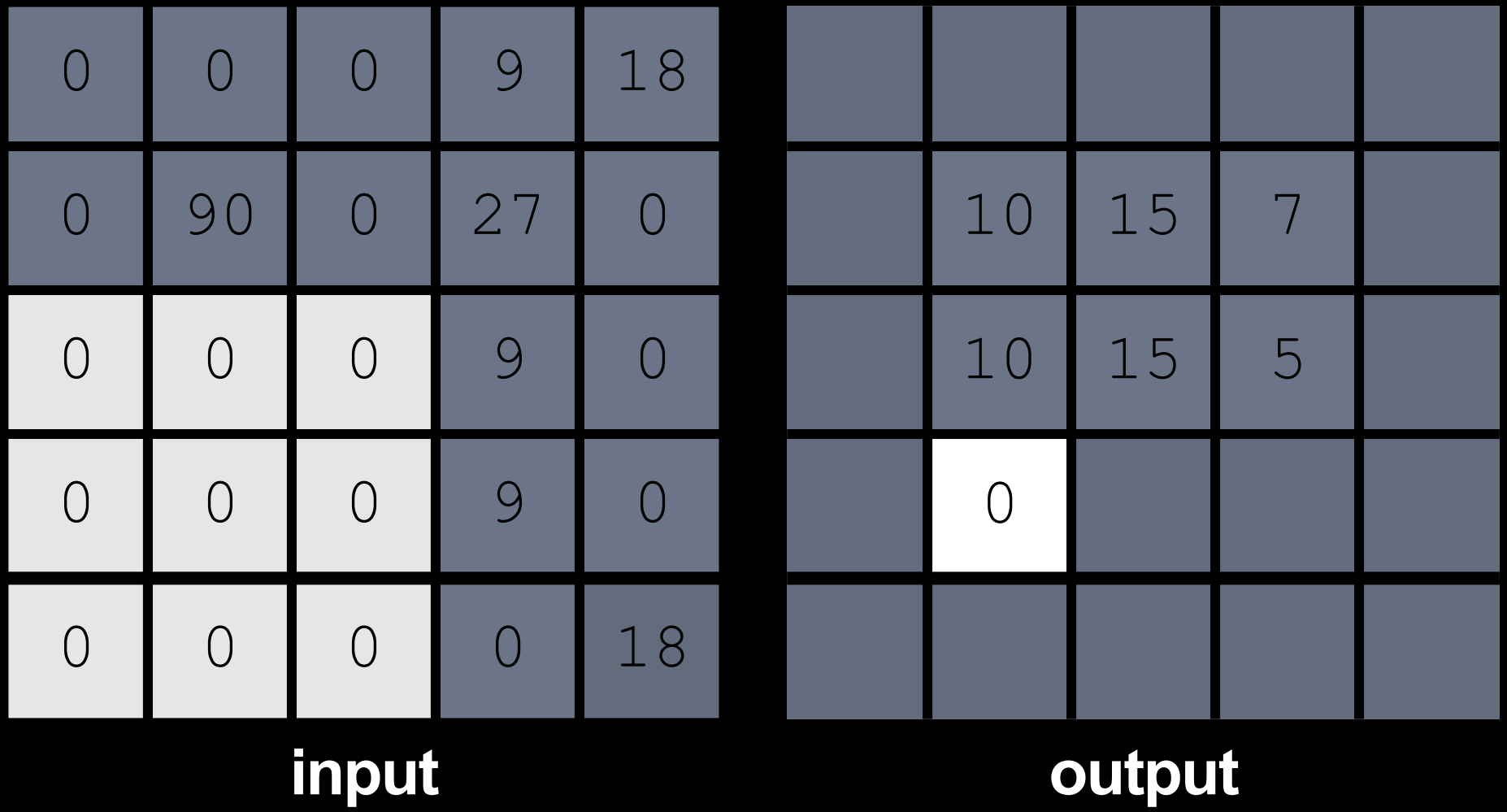

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
10 15 7
10
input
output

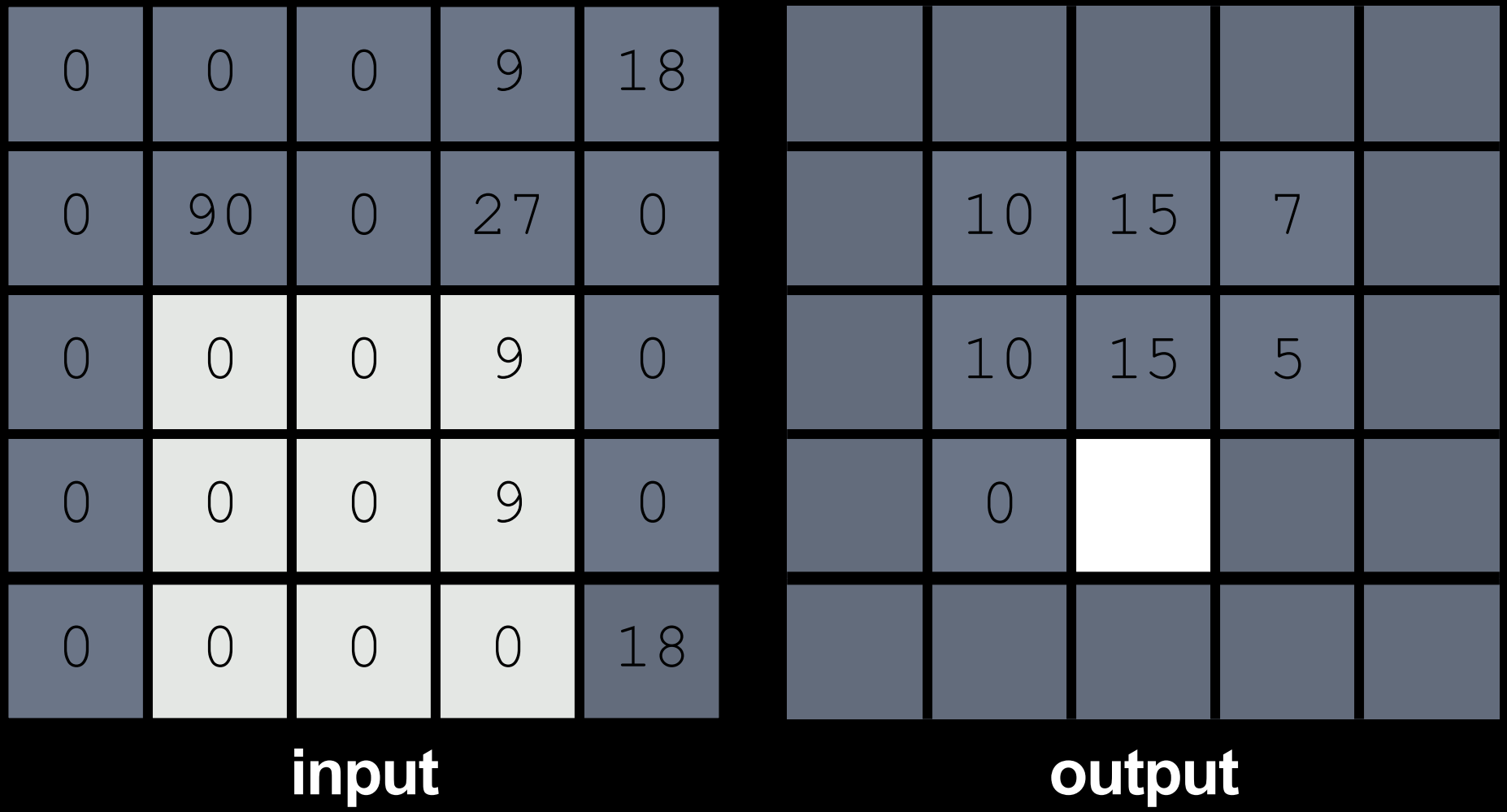

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
10 15 7
10
output

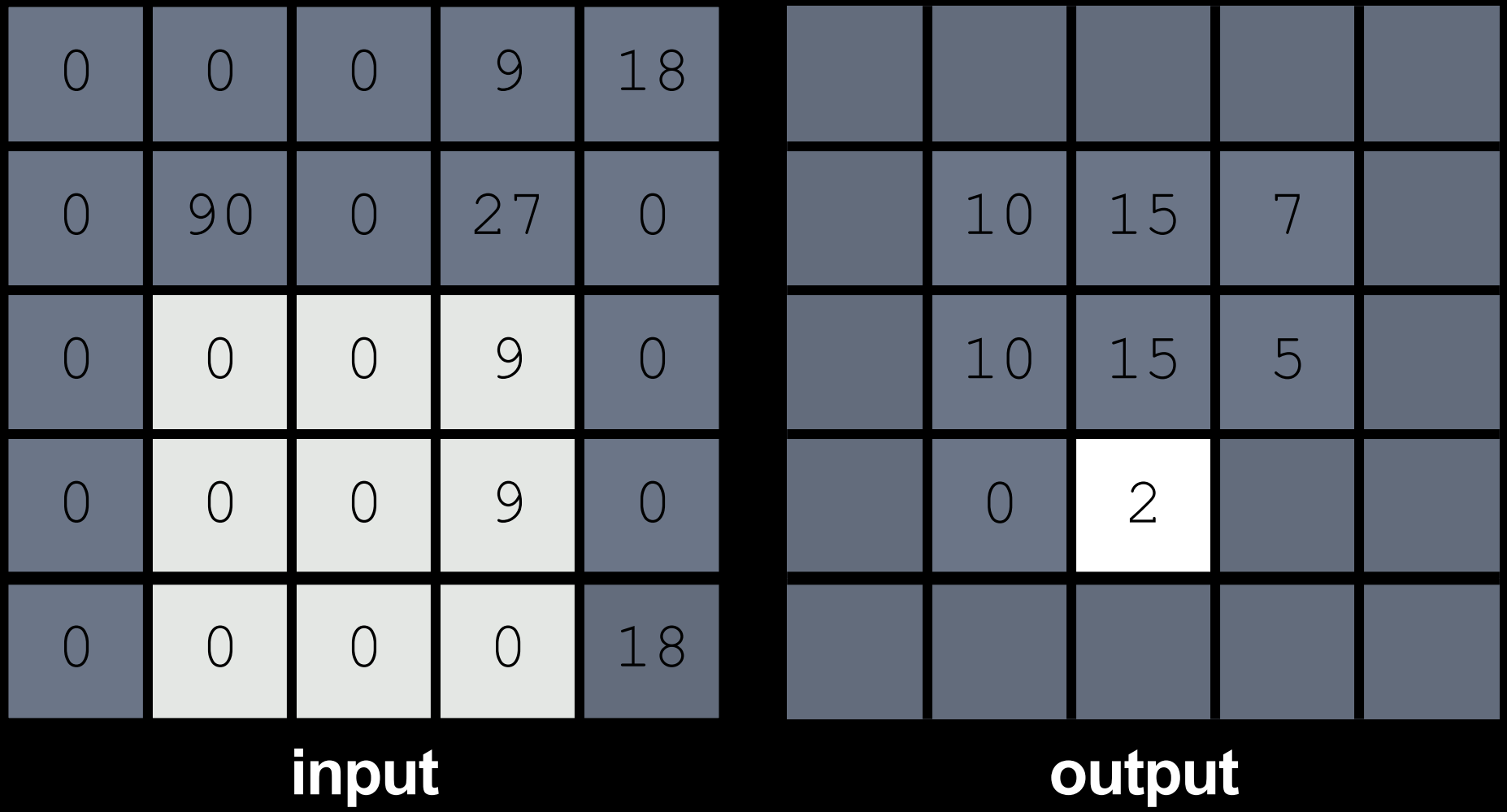

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
10 15 7
10 15
output

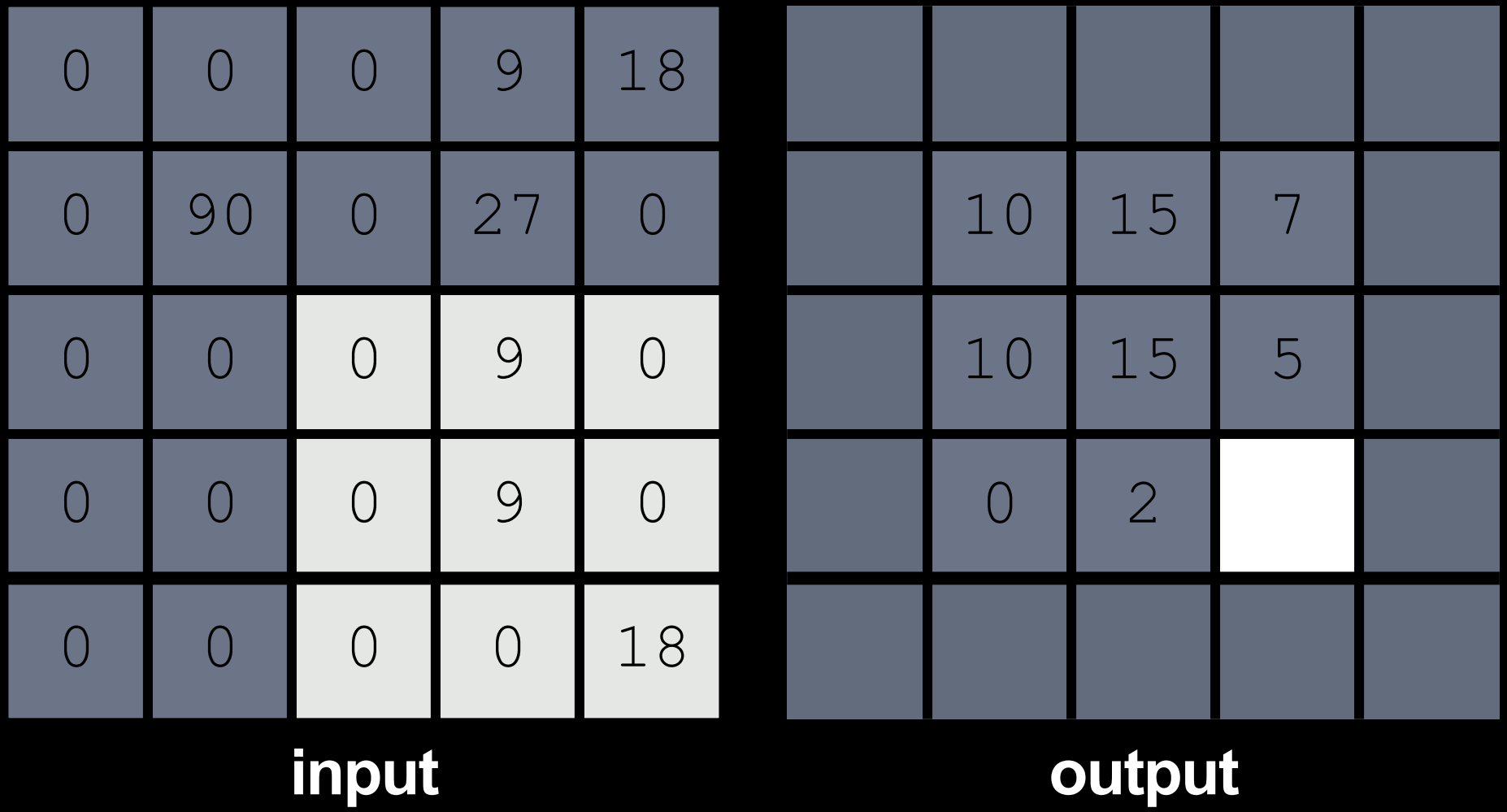

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
10 15 7
10 15 5
output

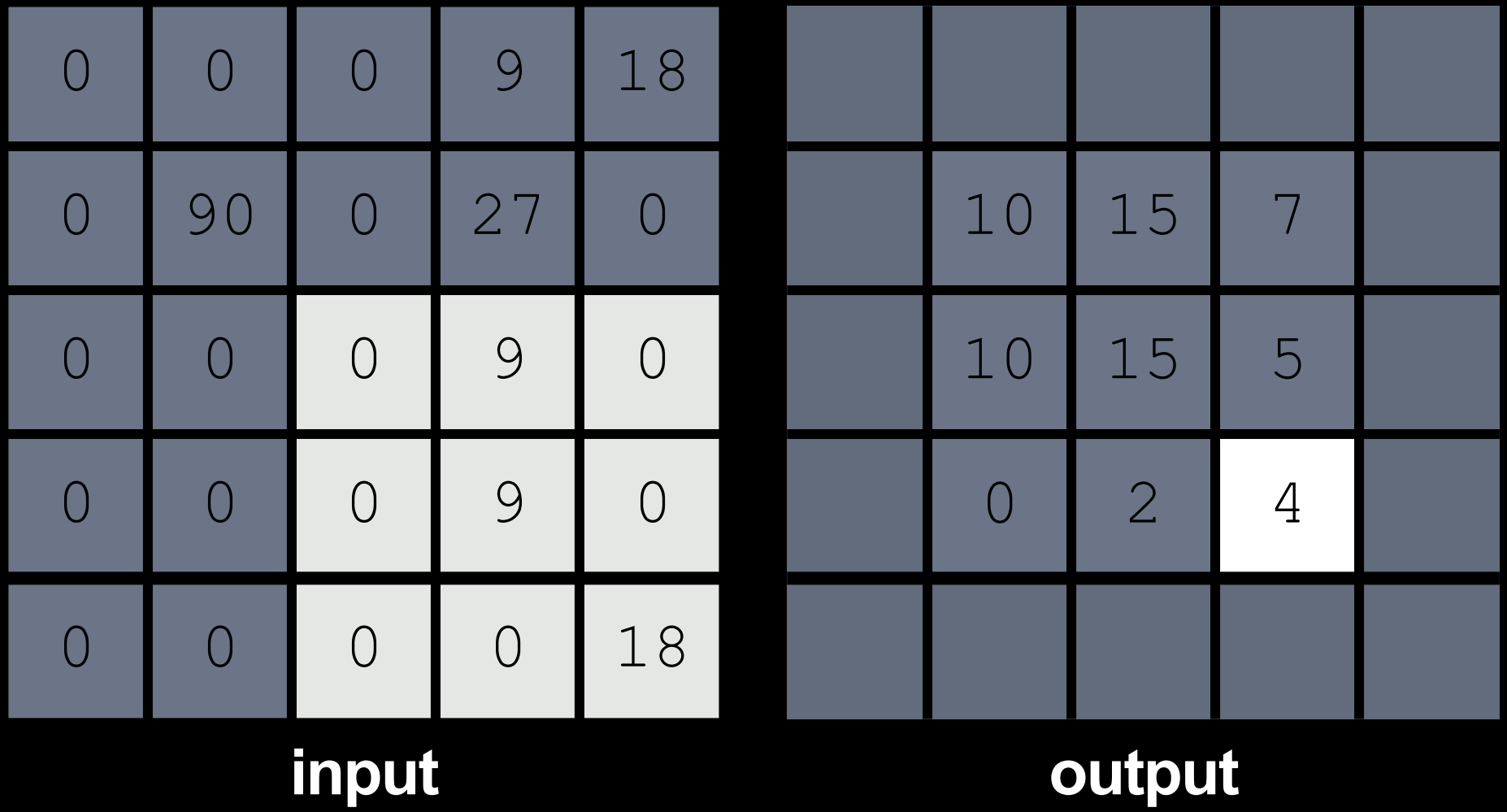

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
10 15 7
10 15 5
output


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
10 15 7
10 15 5
0
input
output

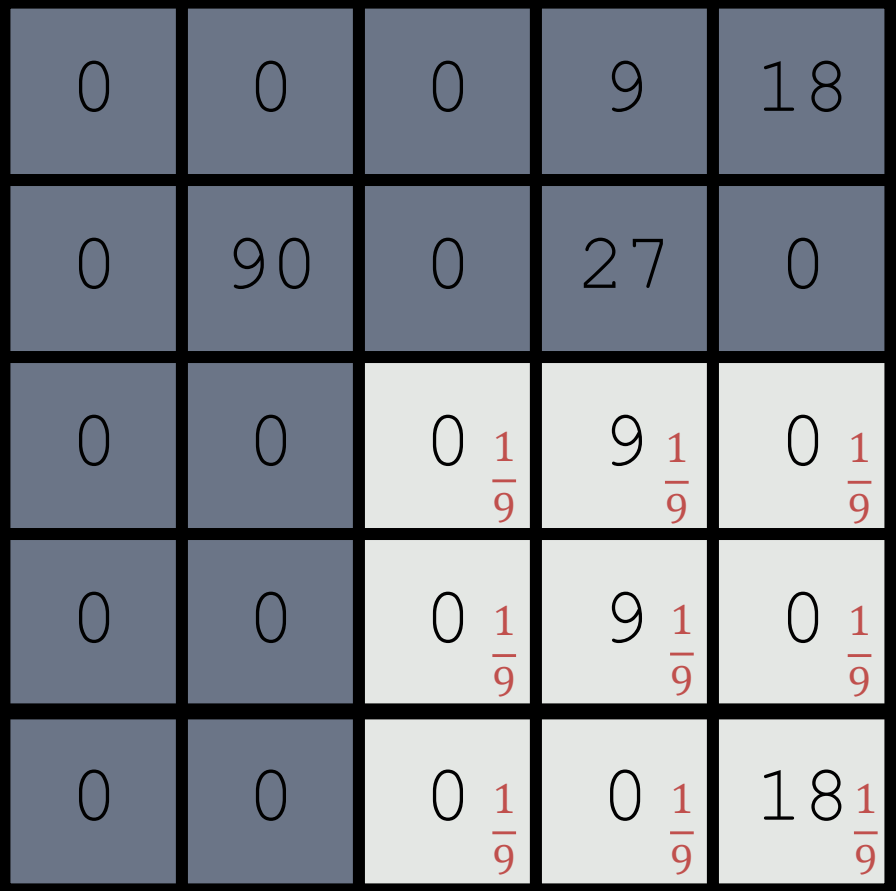

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
input
10 15 7
10 15 5
0
output


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
output

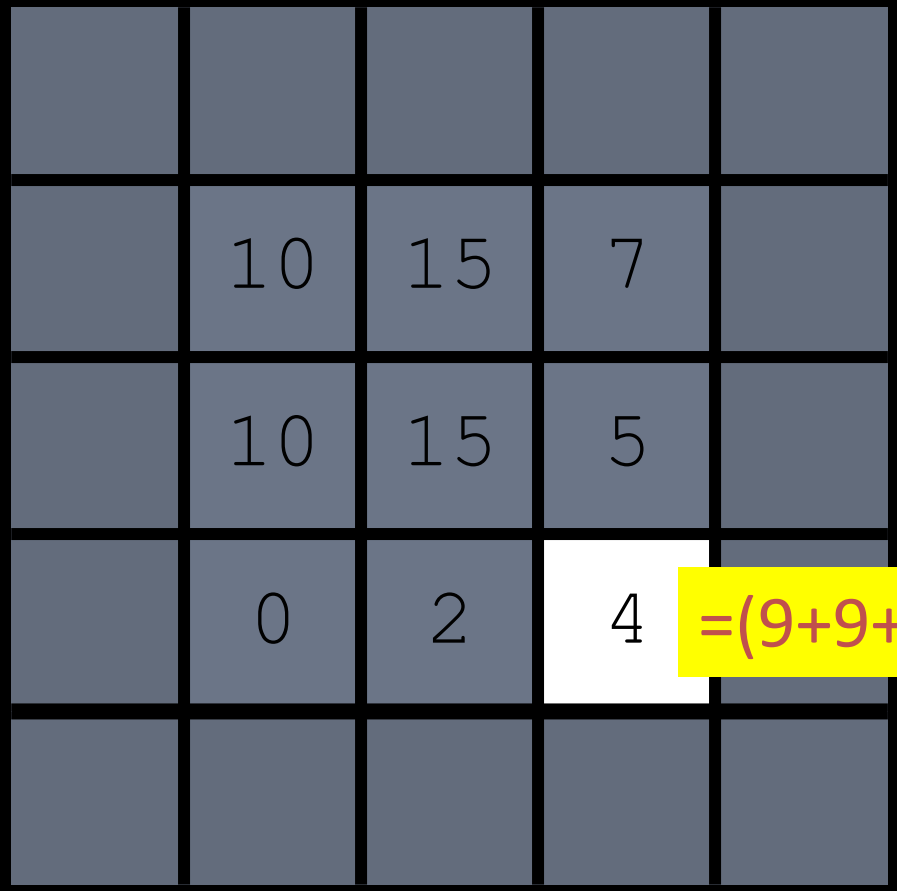

0 0 0 9 18
0 90 0 27 0
0 0 0 9 0
0 0 0 9 0
0 0 0 0 18
10 15 7
10 15 5
0 2
input
output


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
10 15 7
10 15 5
0 2 4
input
output

Equivalent: Pointwise Multiplication (and sum) with 3x3 matrix of all ones

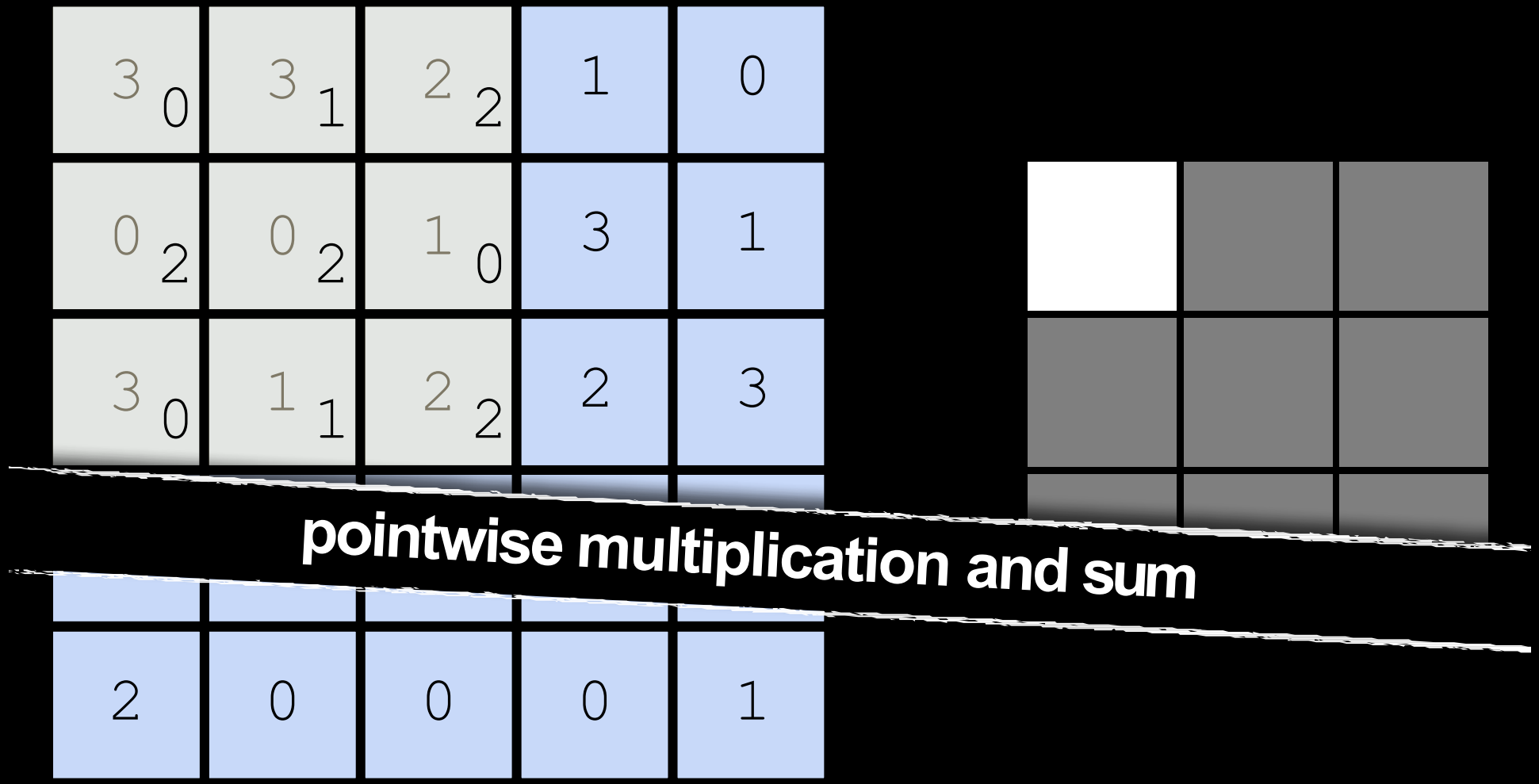


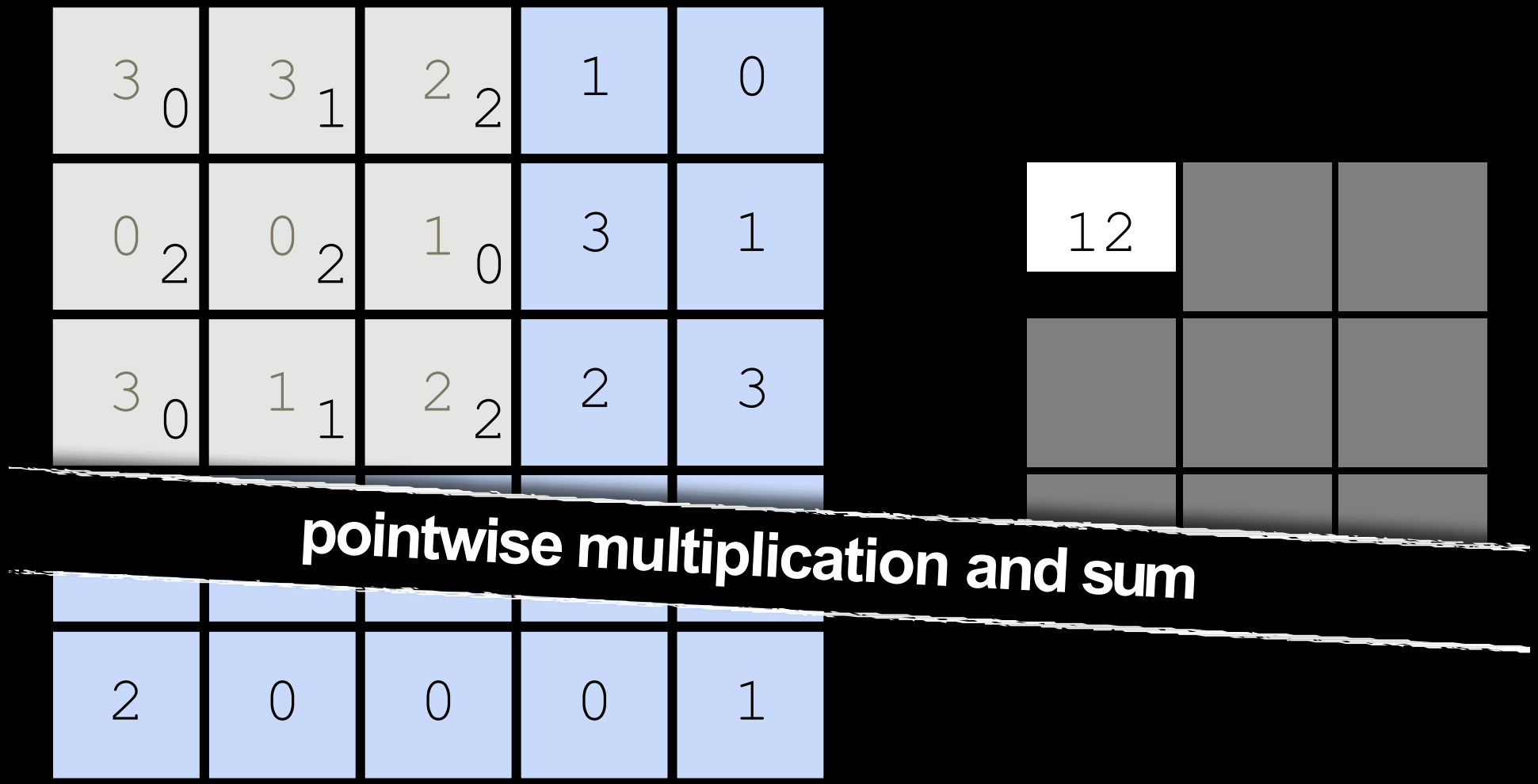
input



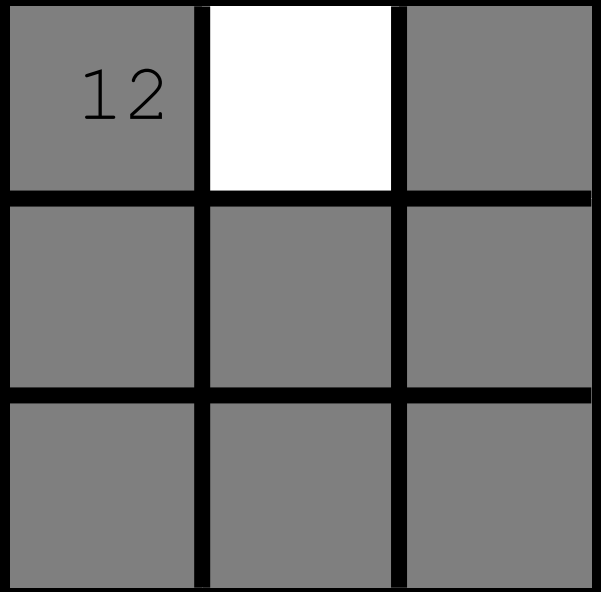
output

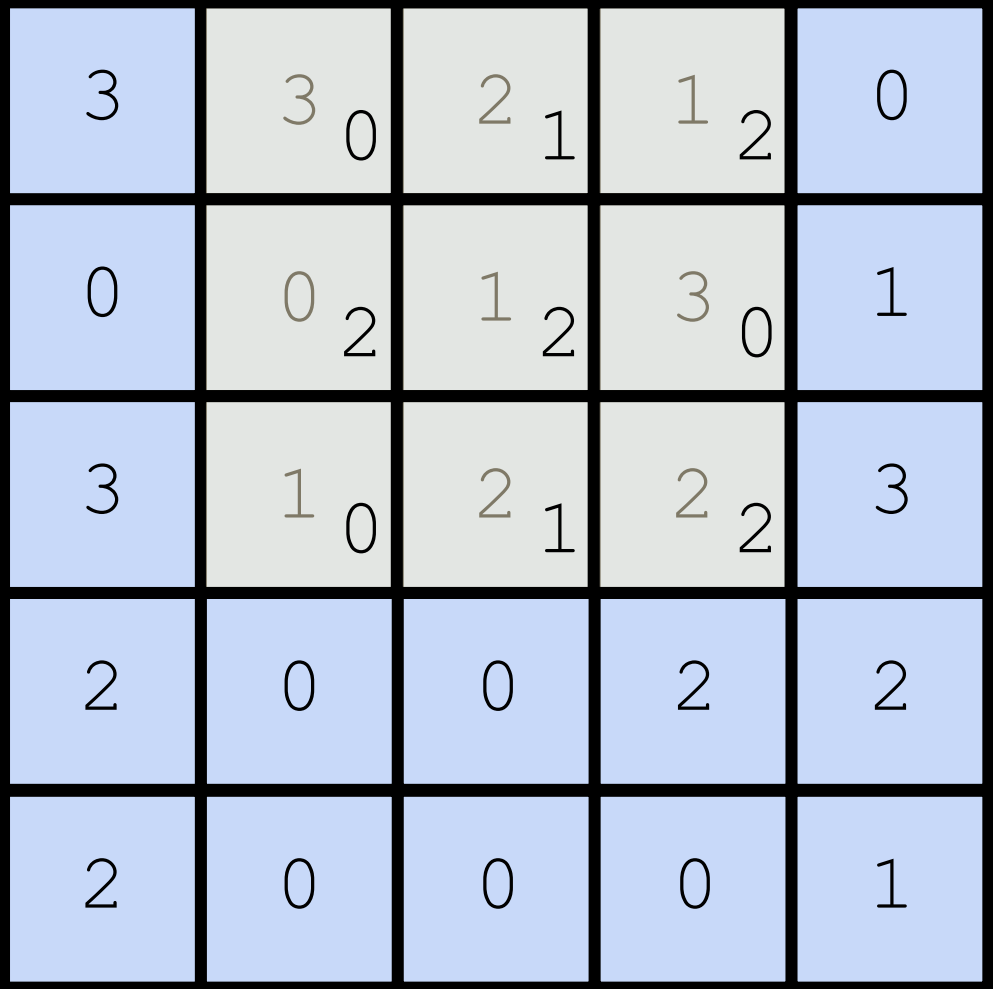
Generalized Averaging

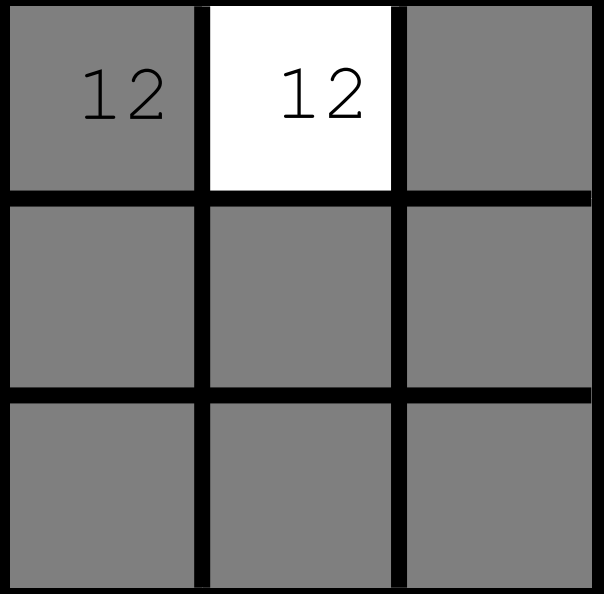

3 0
3 1
2 2
1
0
0 2
0 2
1 0
3
1
3 0
1 1
2 2
2
3
2
0
0
0
1
pointwise multiplication and sum

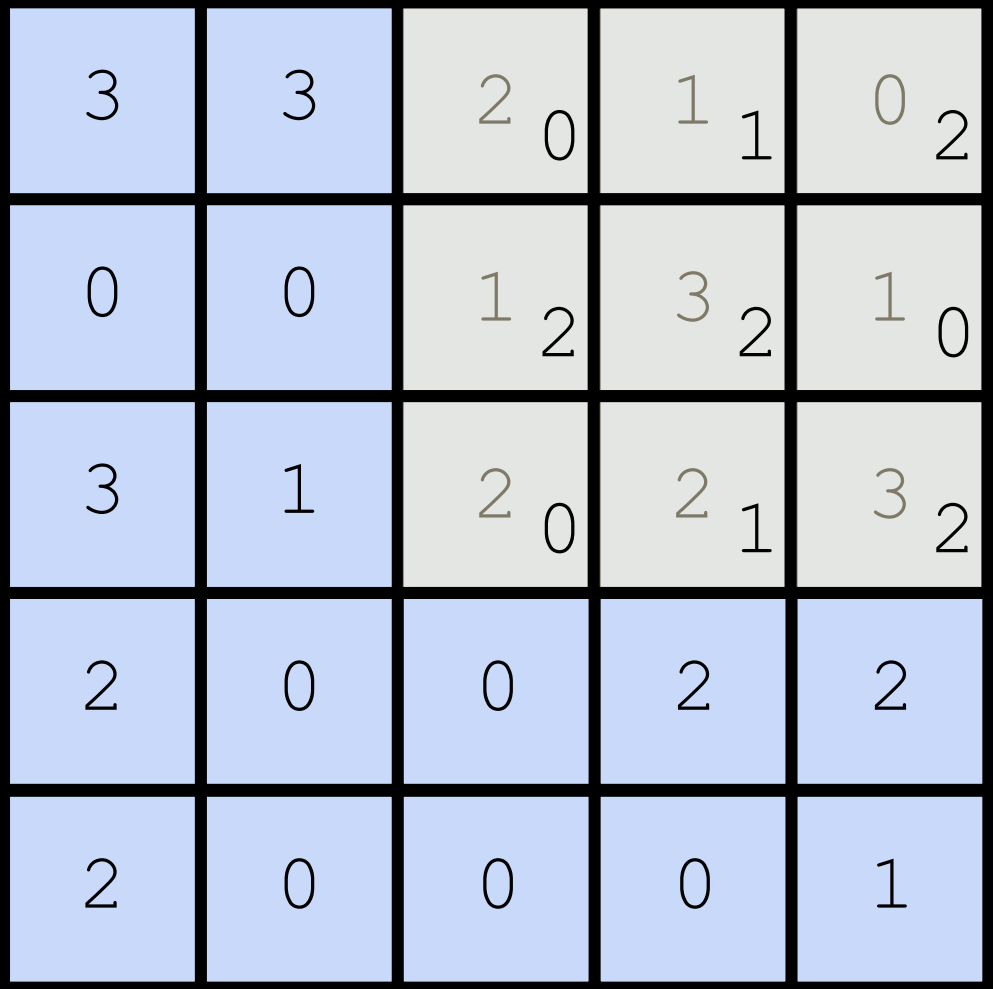

3 0
3 1
2 2
1
0
0 2
0 2
1 0
3
1
3 0
1 1
2 2
2
3
2
0
0
0
1
12
pointwise multiplication and sum

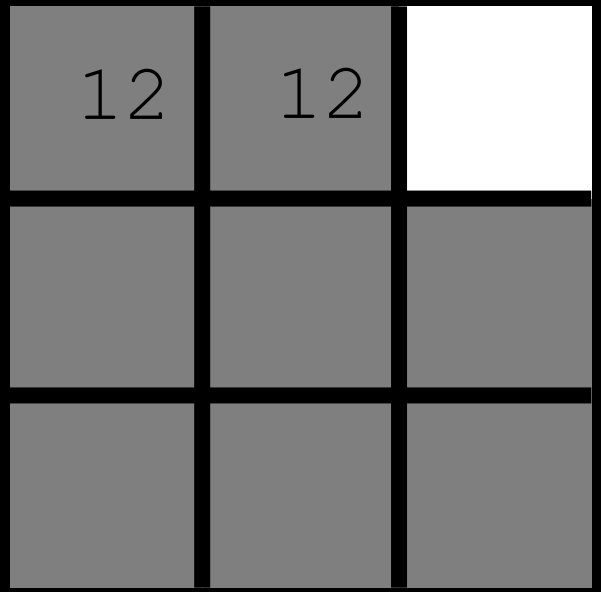
3	3 0	2 1	1 2	0
0	0 2	1 2	3 0	1
3	1 0	2 1	2 2	3
2	0	0	2	2
2	0	0	0	1

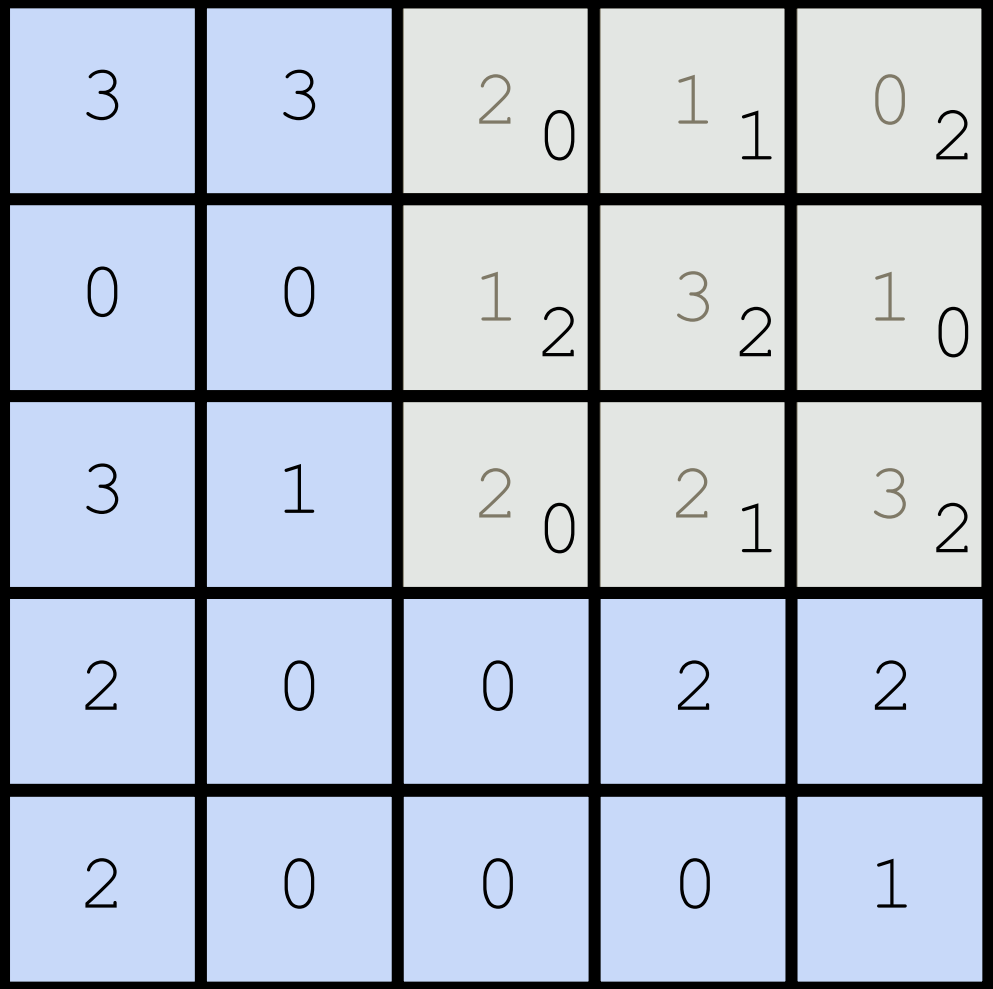


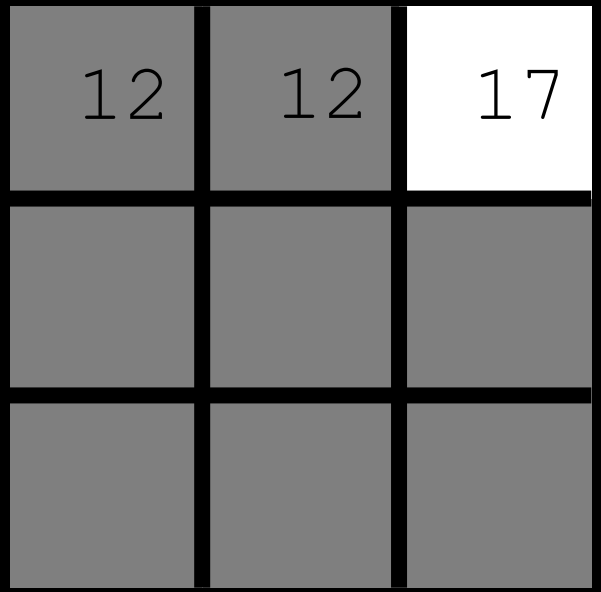

3
3 0
2 1
1 2
0
0
0 2
1 2
3 0
1
3
1 0
2 1
2 2
3
2
0
0
2
2
2
0
0
0
1

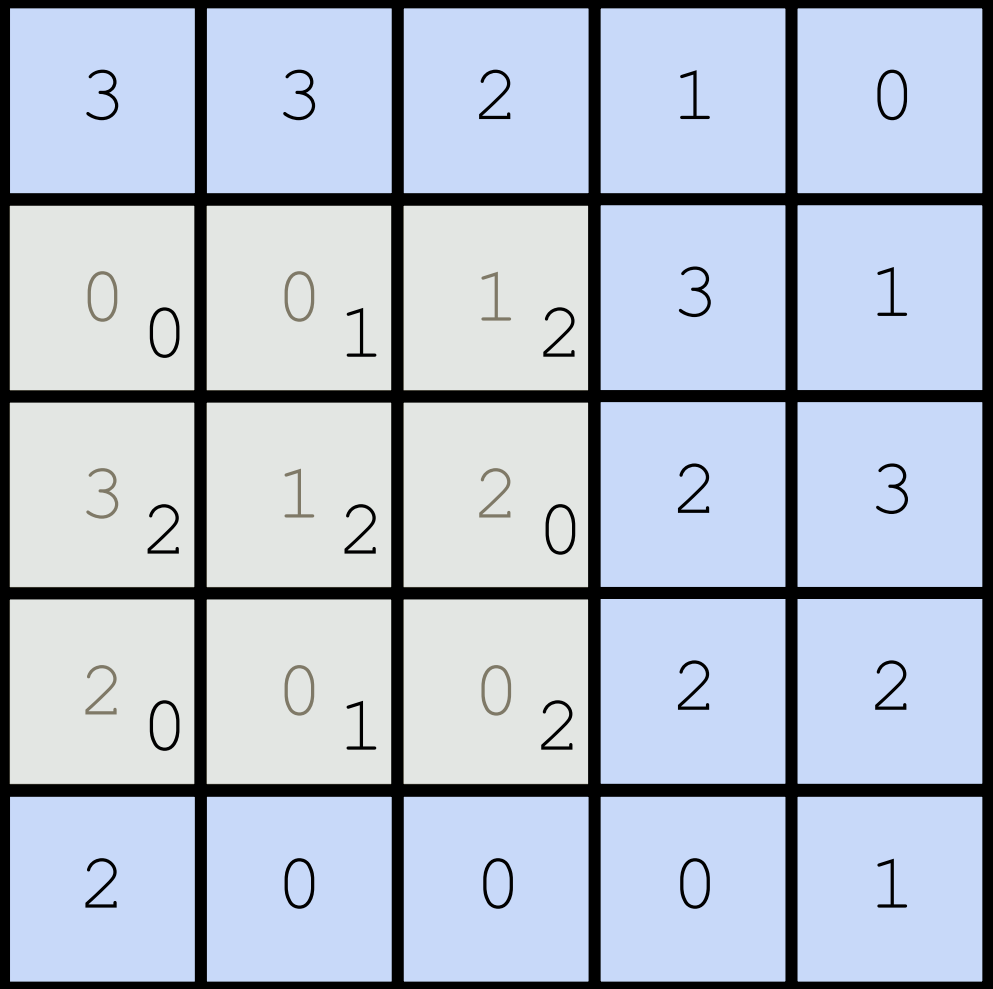

12
12

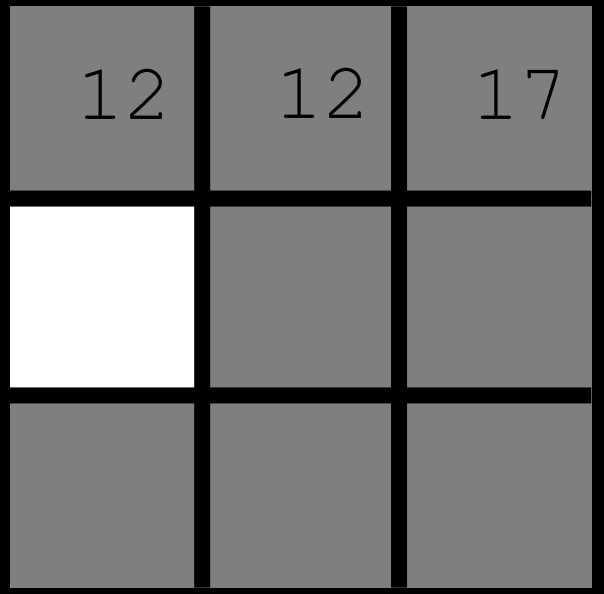

3 3 2 0 1 1 0 2
0 0 1 2 3 2 1 0
3 1 2 0 2 1 3 2
2 0 0 2 2
2 0 0 0 1

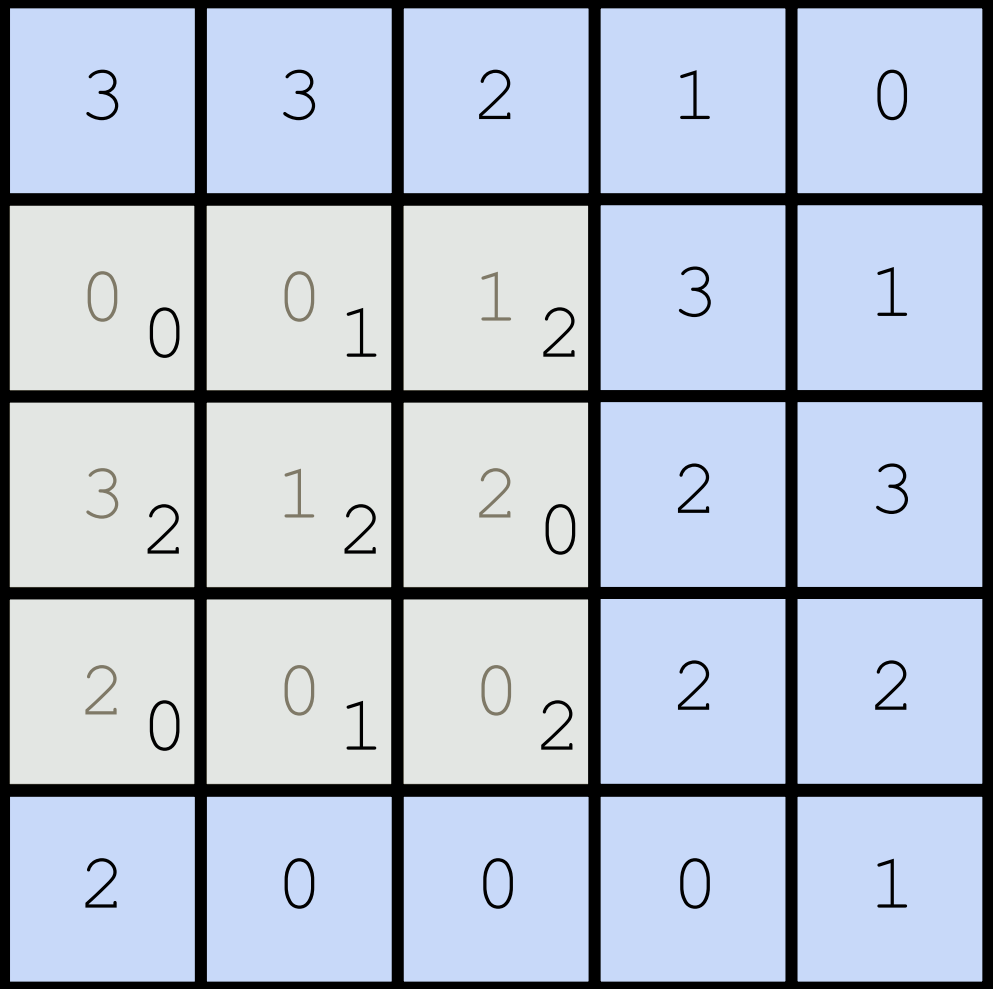

12 12

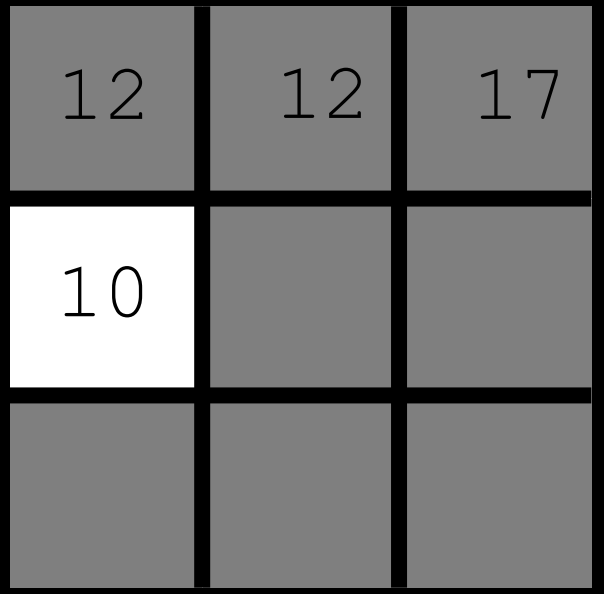

3 3 2 0 1 1 0 2
0 0 1 2 3 2 1 0
3 1 2 0 2 1 3 2
2 0 0 2 2
2 0 0 0 1

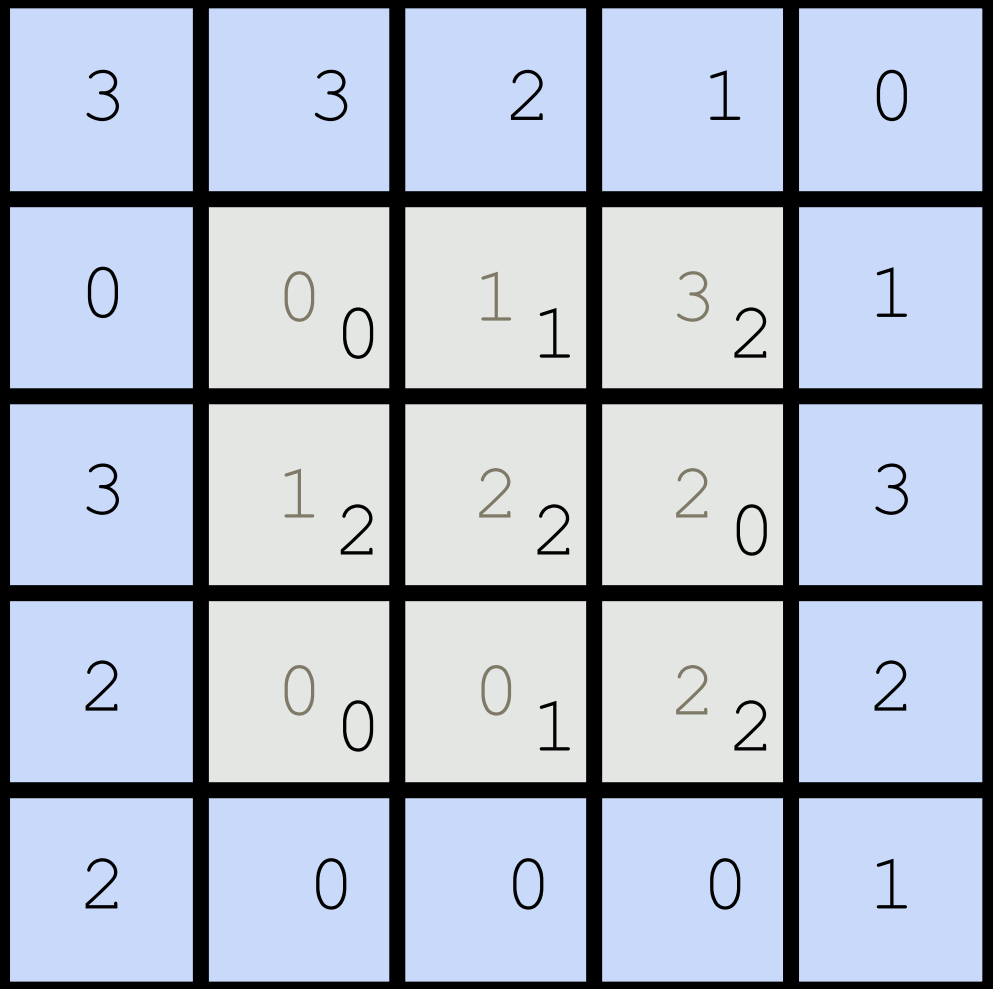

12 12 17

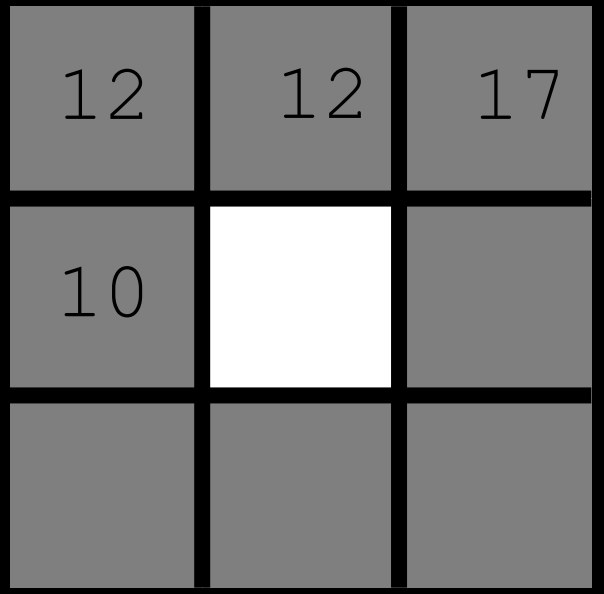

3 3 2 1 0
0 0 0 1 1 2 3 1
3 2 1 2 2 0 2 3
2 0 0 1 0 2 2 2
2 0 0 0 1

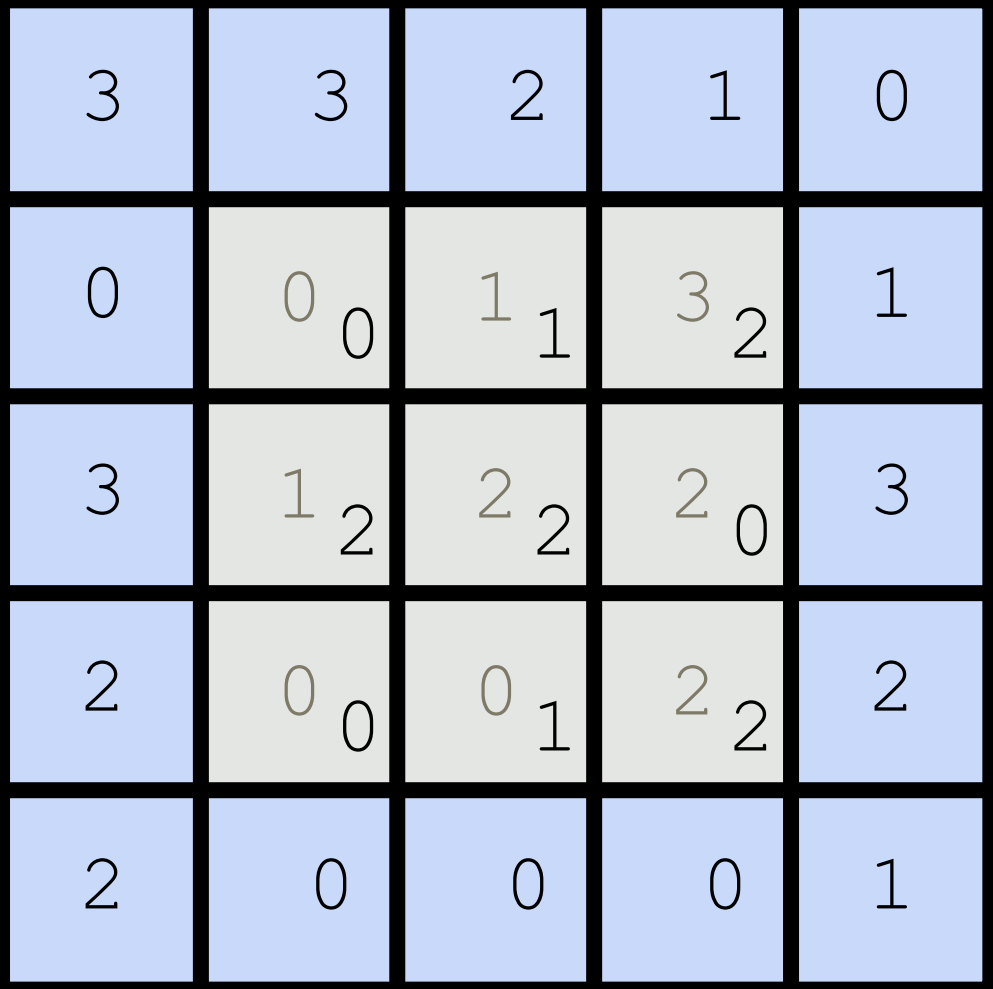

12 12 17

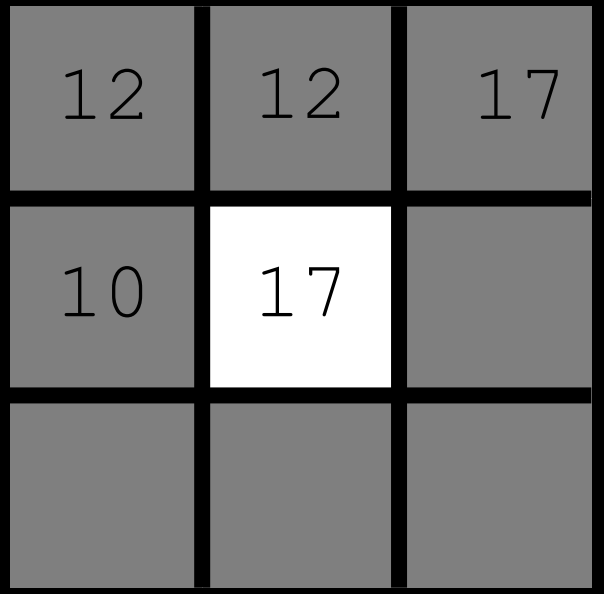

3 3 2 1 0
0 0 0 1 1 2 3 1
3 2 1 2 2 0 2 3
2 0 0 1 0 2 2 2
2 0 0 0 1

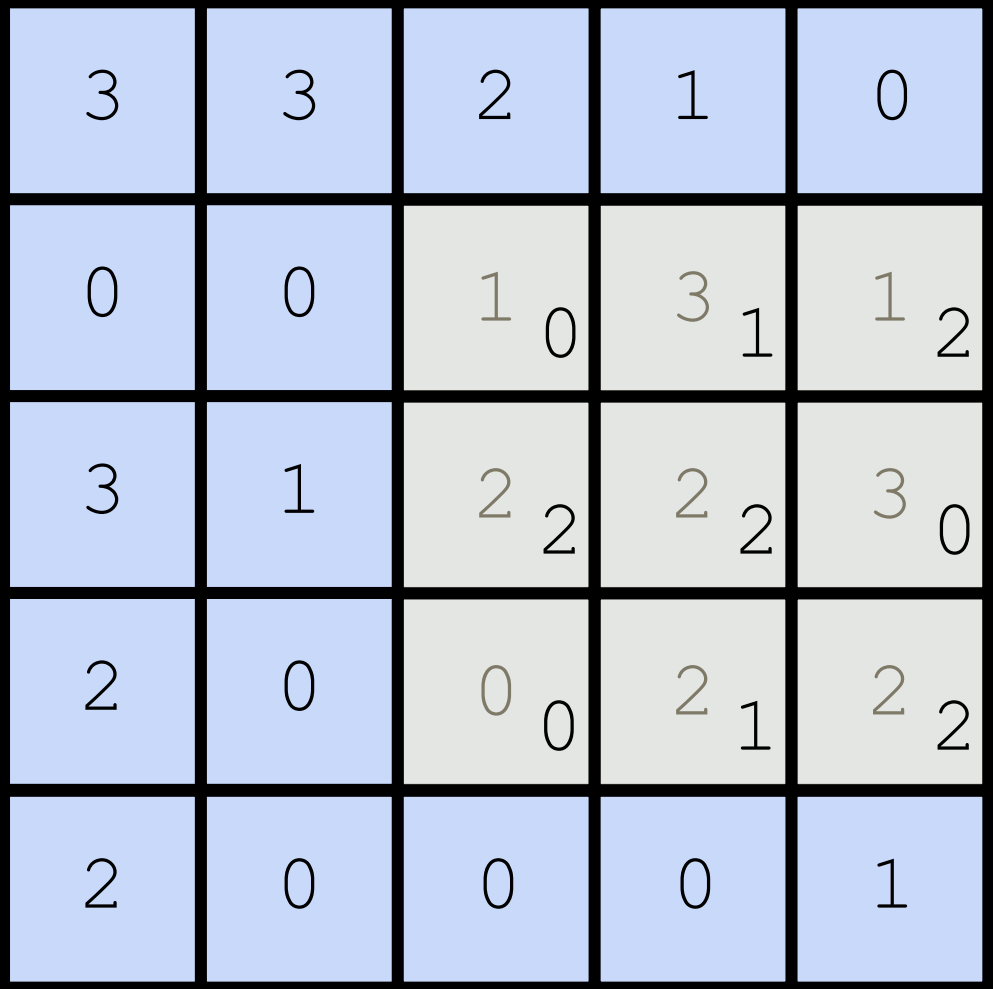

12 12 17
10

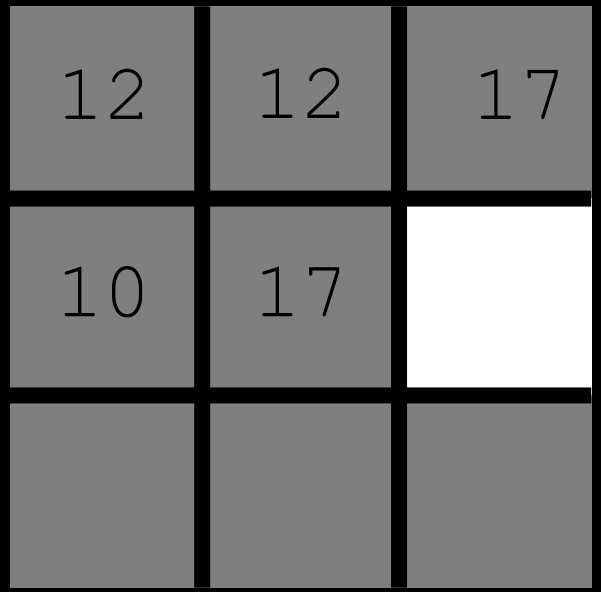

3 3 2 1 0
0 0 0 1 1 3 2 1
3 1 2 2 2 2 0 3
2 0 0 0 1 2 2 2
2 0 0 0 1

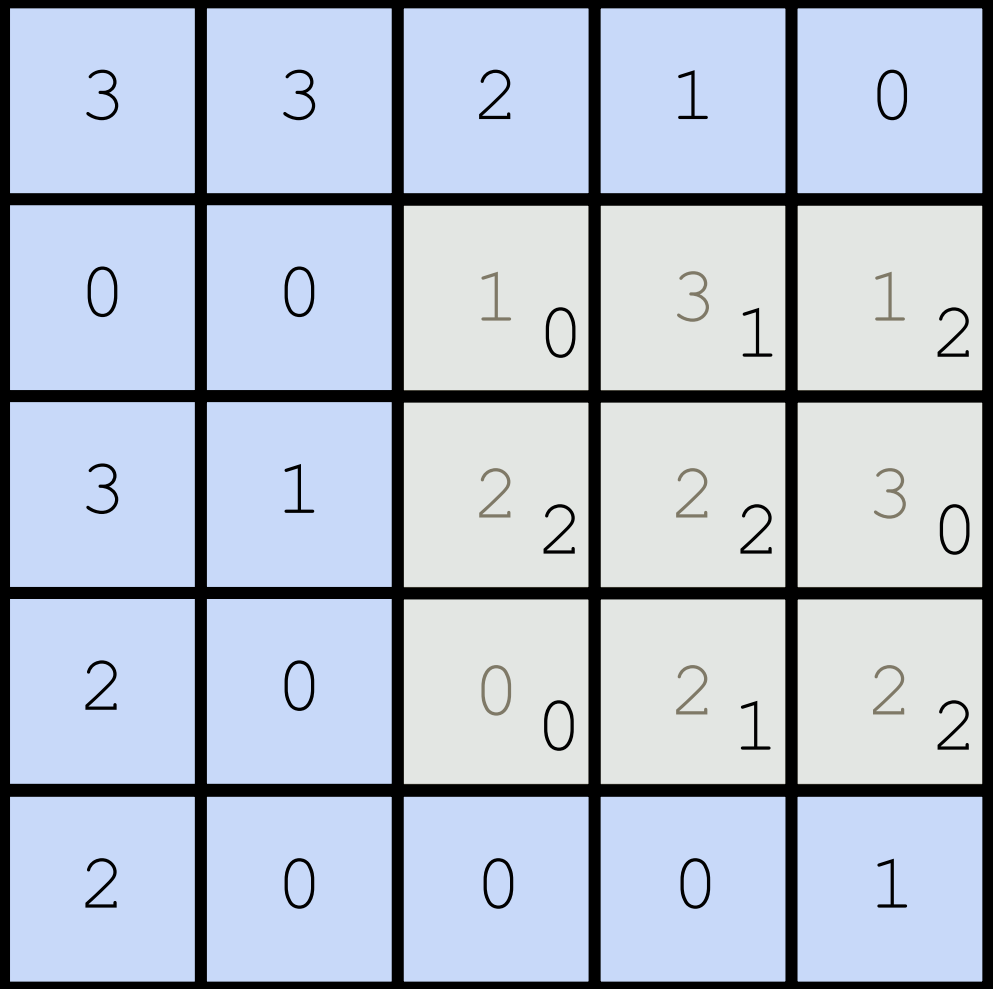

12 12 17
10

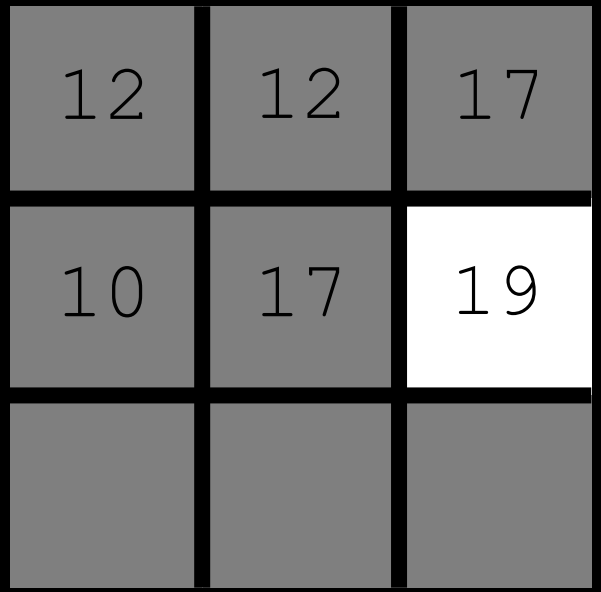

3 3 2 1 0
0 0 0 1 1 3 2 1
3 1 2 2 2 2 0 3
2 0 0 0 1 2 2 2
2 0 0 0 1

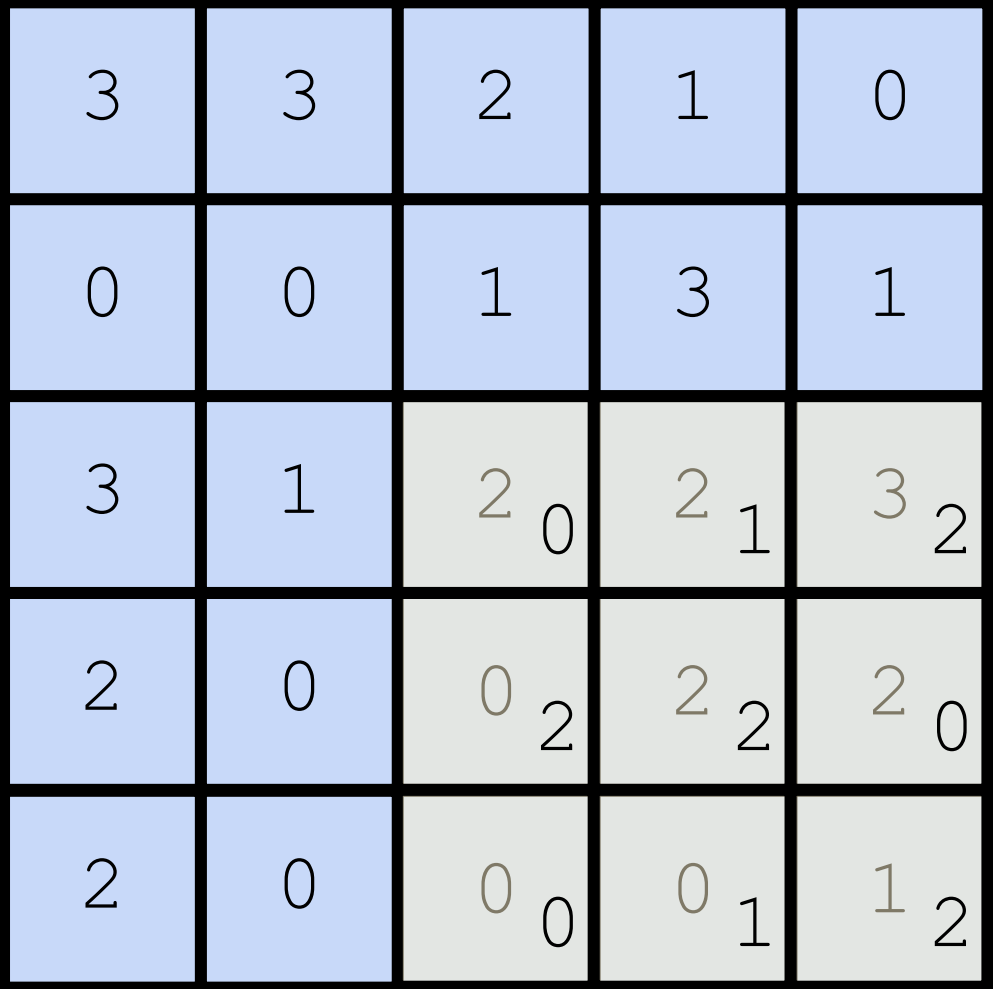

12 12 17
10 17

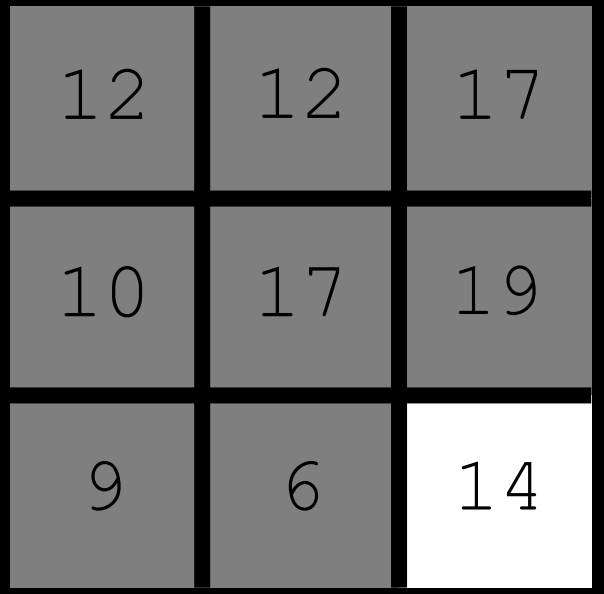

3 3 2 1 0
0 0 1 0 3 1 1 2
3 1 2 2 2 2 3 0
2 0 0 0 2 1 2 2
2 0 0 0 1


12 12 17
10 17


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


12 12 17
10 17 19


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

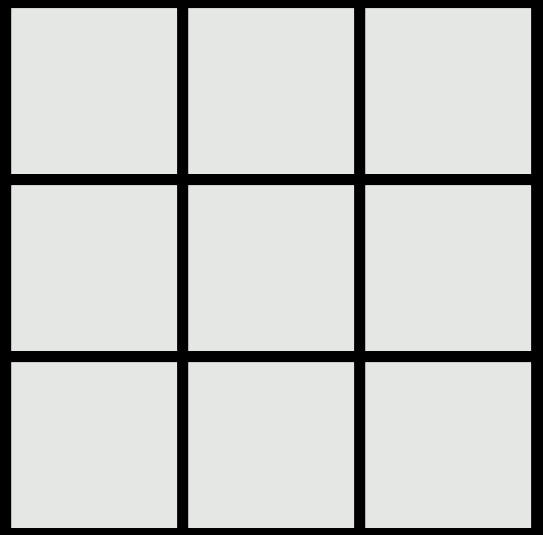

12 12 17
10 17 19
9 6 14

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

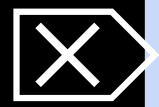
Correlation

12	12	17
10	17	19
9	6	14

Average Filtering



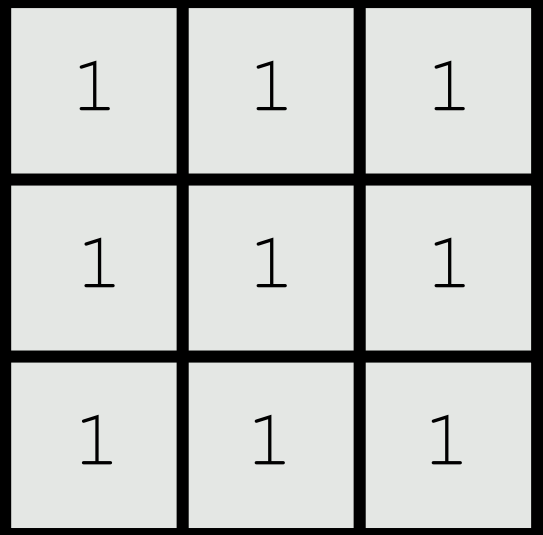
$G[x, y]$



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

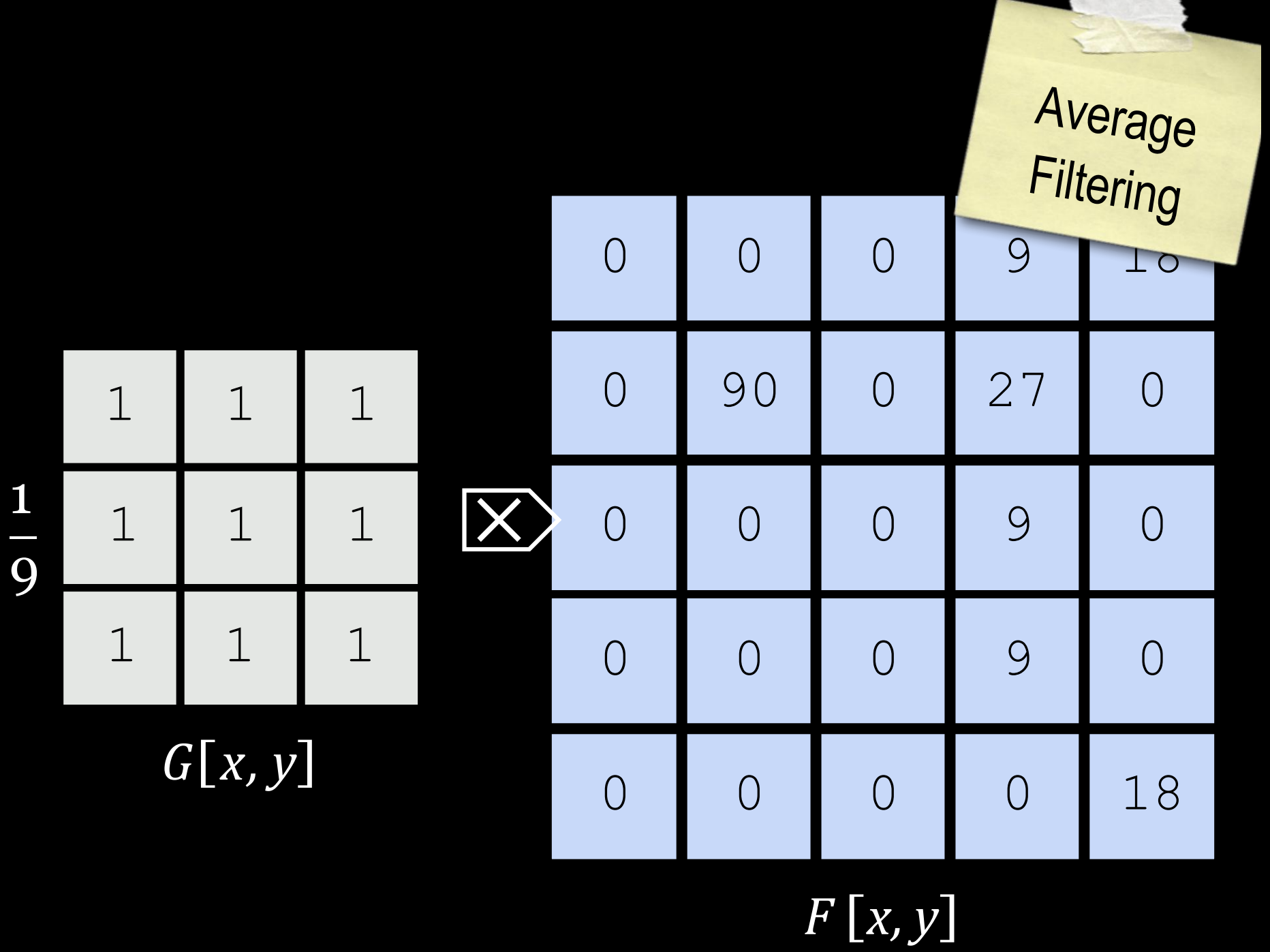
$F[x, y]$

Average Filtering



$G[x, y]$

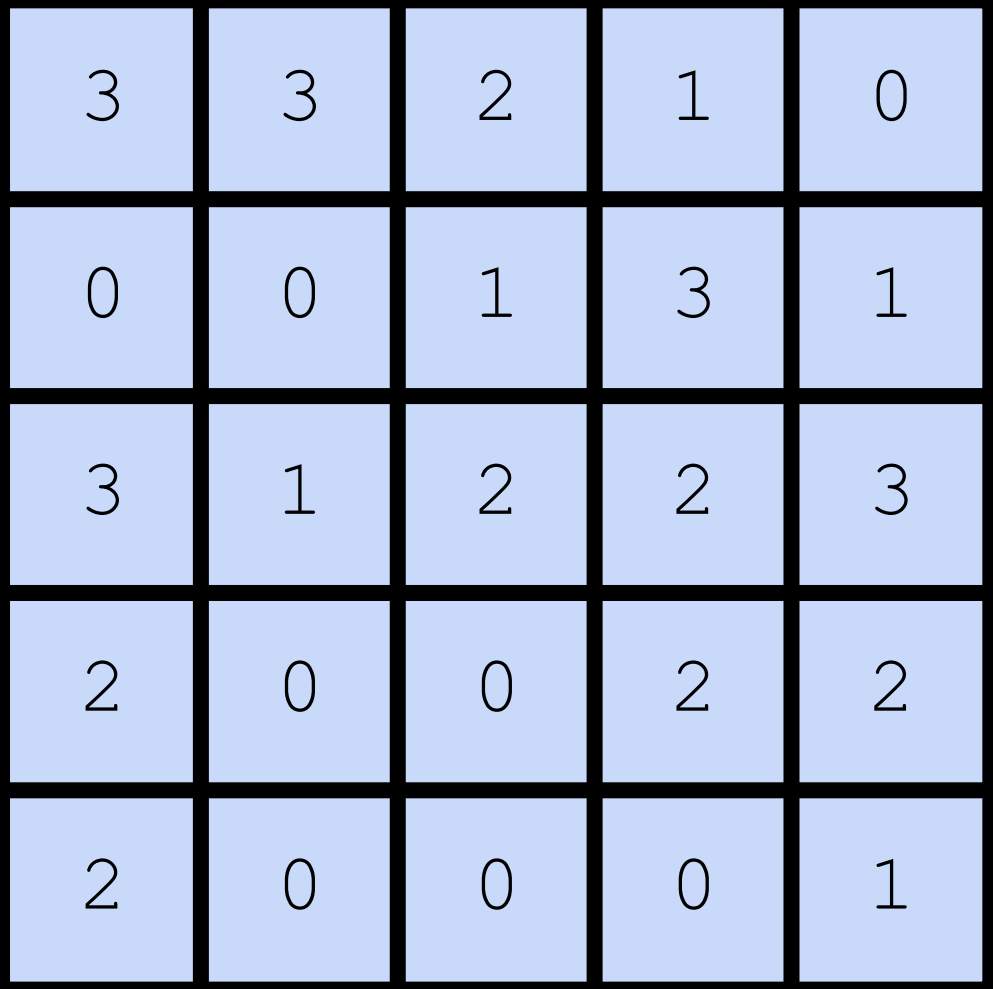
$F[x, y]$


Average Filtering
1/9
1 1 1
1 1 1
1 1 1
G[x, y]
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]


original

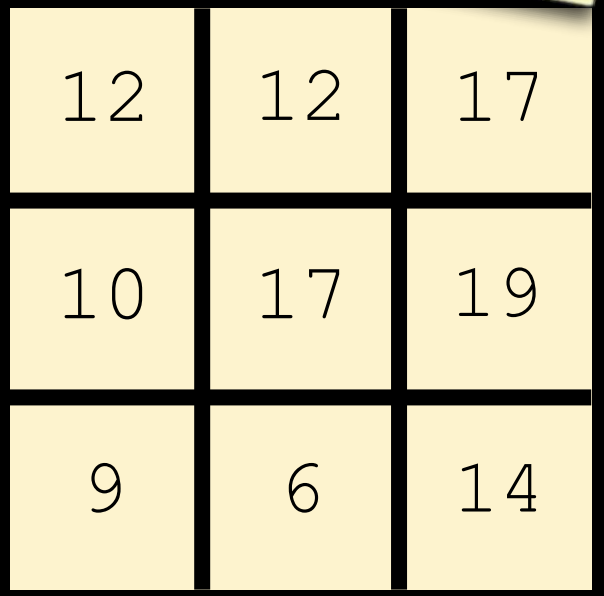

box filter
original


box filter
smoothing by averaging



Image

Correlation



Kernel / Filter

Convolution

(Correlation with a flipped kernel)

Convolution for 2D discrete signals

Definition of filtering as convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

filtered image

filter

input image

Convolution for 2D discrete signals

Definition of filtering as convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

filtered image

filter

input image

If the filter $f(i, j)$ is non-zero only within $-1 \leq i, j \leq 1$, then

$$(f * g)(x, y) = \sum_{i,j=-1}^{1} f(i, j) I(x - i, y - j)$$

The kernel we saw earlier is the 3x3 matrix representation of $f(i, j)$.

Convolution vs correlation

Definition of discrete 2D convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x - i, y - j)$$

Definition of discrete 2D correlation:

notice the lack of a flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j)I(x + i, y + j)$$

- Most of the time won't matter, because our kernels will be symmetric.
- Will be important when we discuss frequency-domain filtering

Example: box filter *aka* averaging filter

$$g[\cdot,\cdot]$$

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

Slide credit: David Lowe (UBC)

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10							

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	0	10	20						

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

	0	10	20	30					

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

	0	10	20	30	30				

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

	0	10	20	30	30				
				?					

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

	0	10	20	30	30				
						?			
				50					

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[\cdot,\cdot]$

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

$h[\cdot,\cdot]$

	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	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Box Filter

What does it do?

- Replaces each pixel with an average of its neighborhood

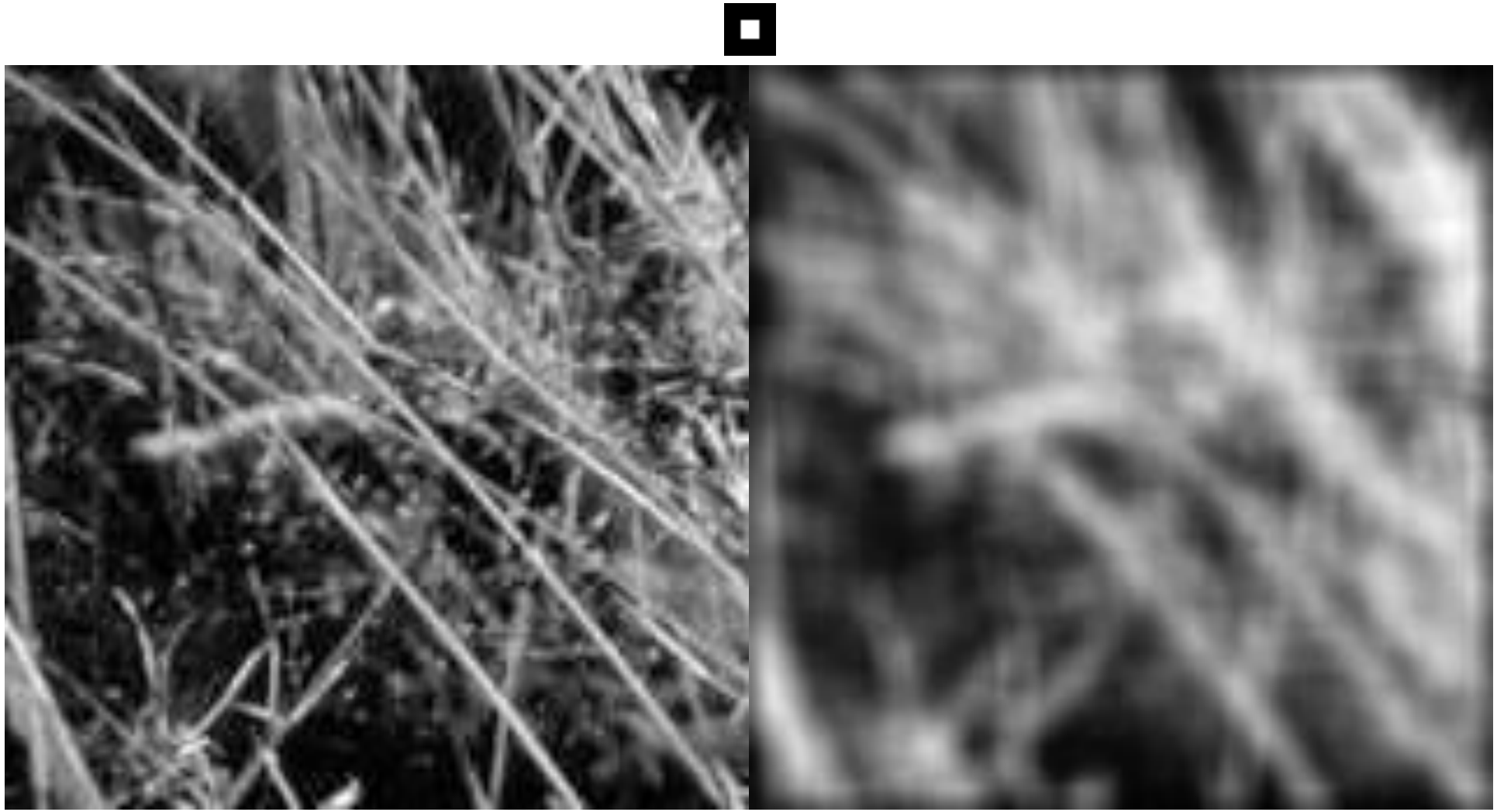
- Achieve smoothing effect (remove sharp features)

$g[\cdot,\cdot]$

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

Slide credit: David Lowe (UBC)

Smoothing with box filter



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

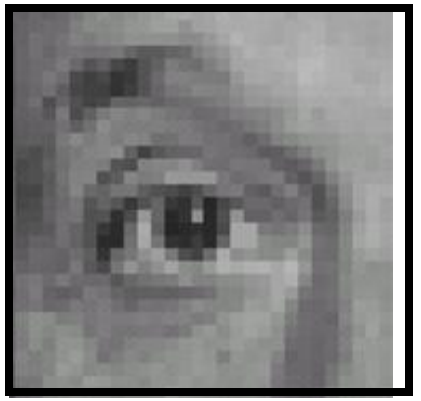
?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

Practice with linear filters

Original

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

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters

Original

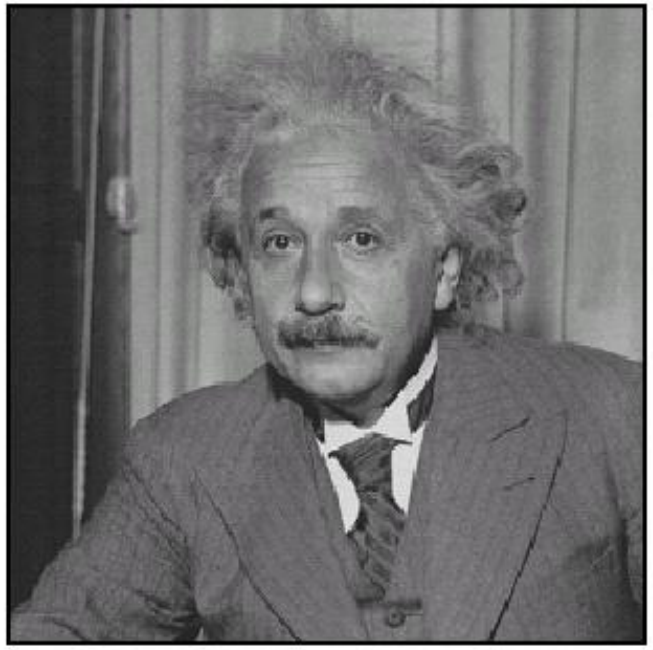
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

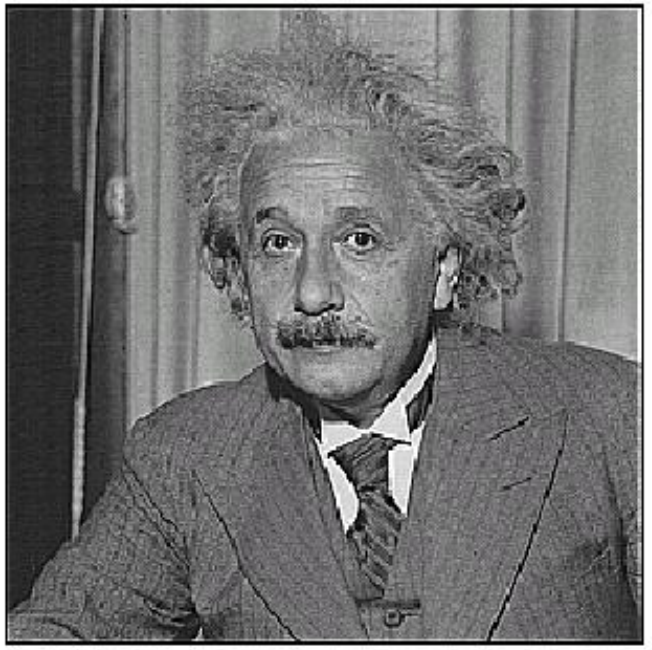
- Accentuates differences with local average

Source: D. Lowe

Sharpening



before



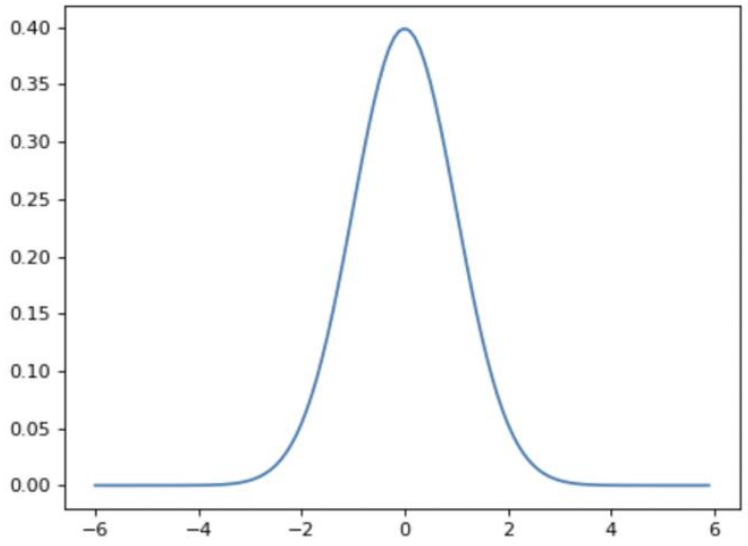
after

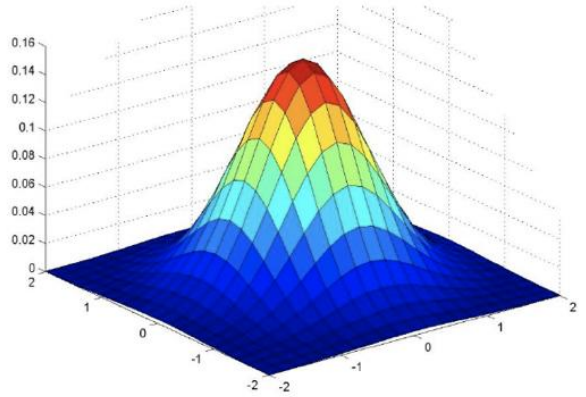
Source: D. Lowe

The Gaussian filter

$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

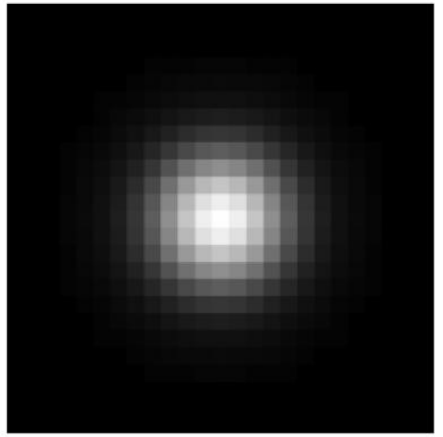
- named (like many other things) after Carl Friedrich Gauss





Gaussian kernel

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



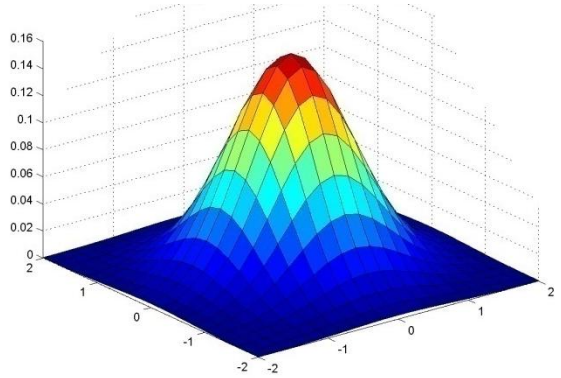
- weight falls off with distance from center pixel
- theoretically infinite, in practice truncated to some maximum distance (in this case, 3x3)

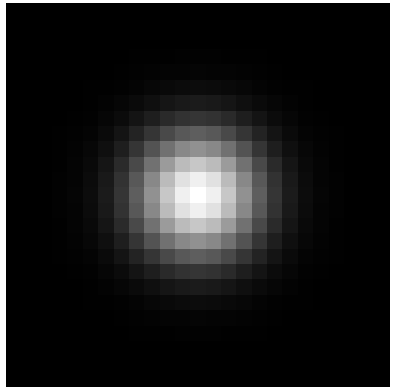
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

3x3 Gaussian kernel

Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

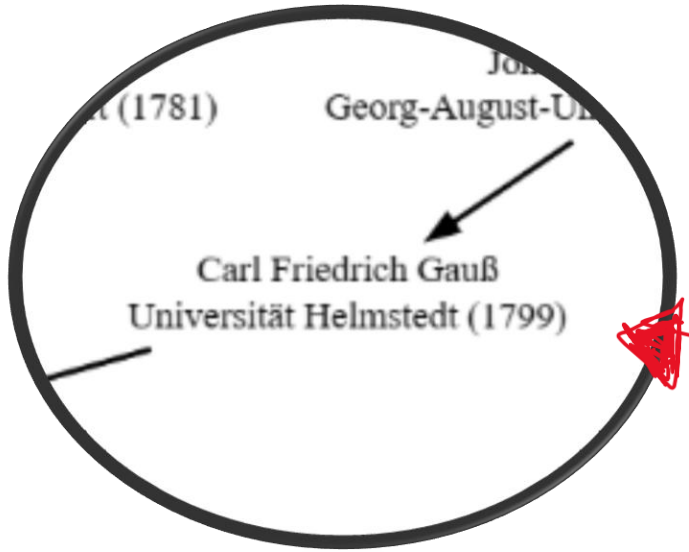
5 x 5, $\sigma = 1$

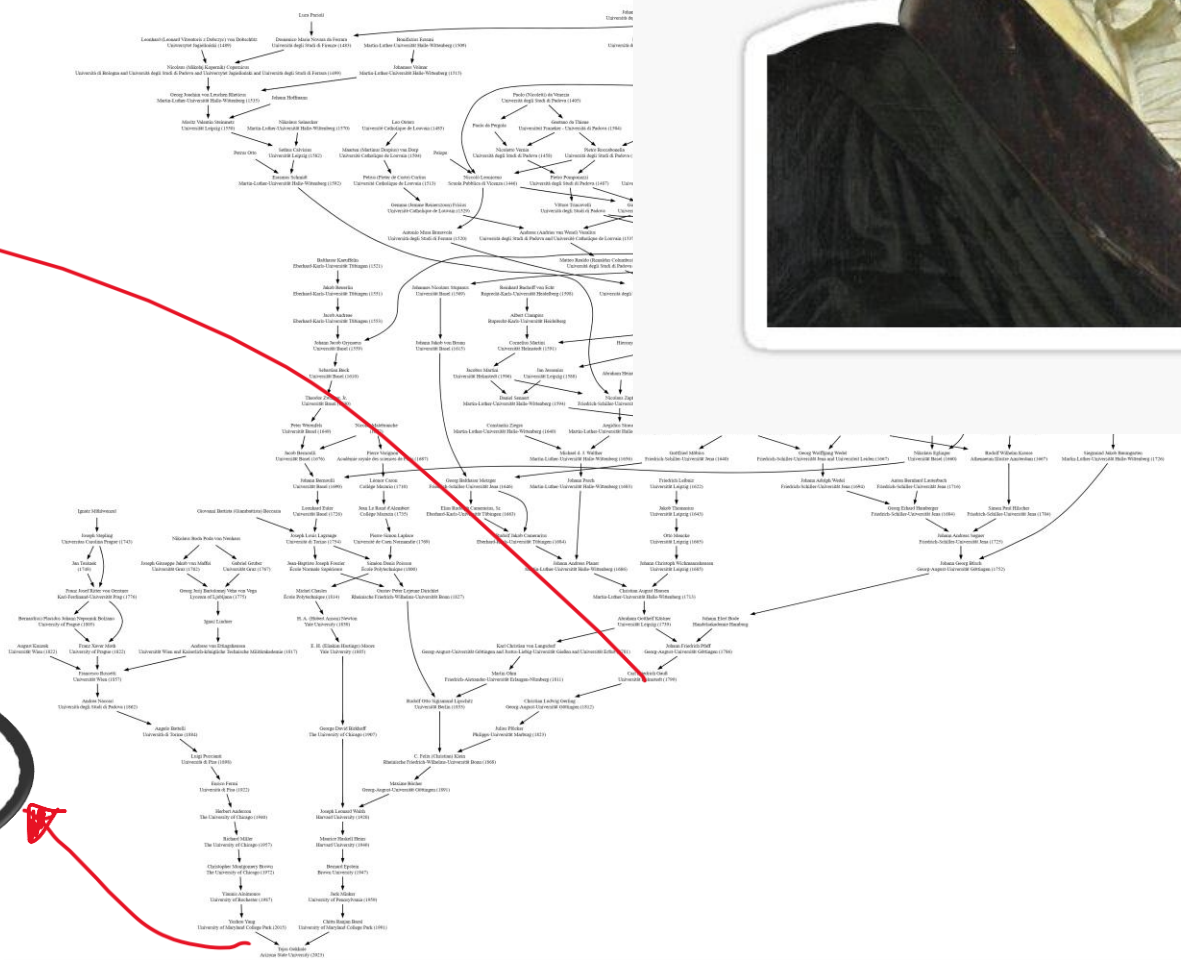
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

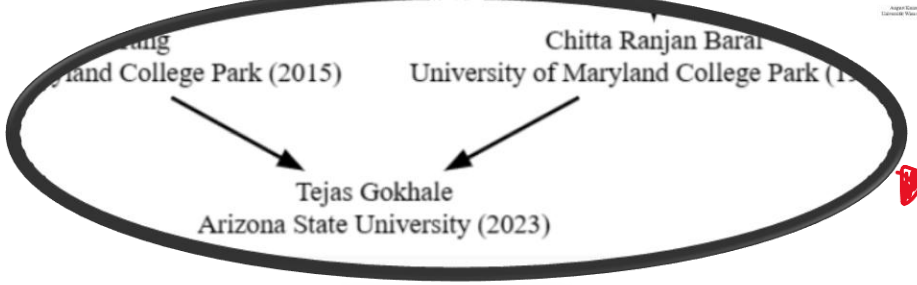
Slide credit: Christopher Rasmussen

Unrelated:

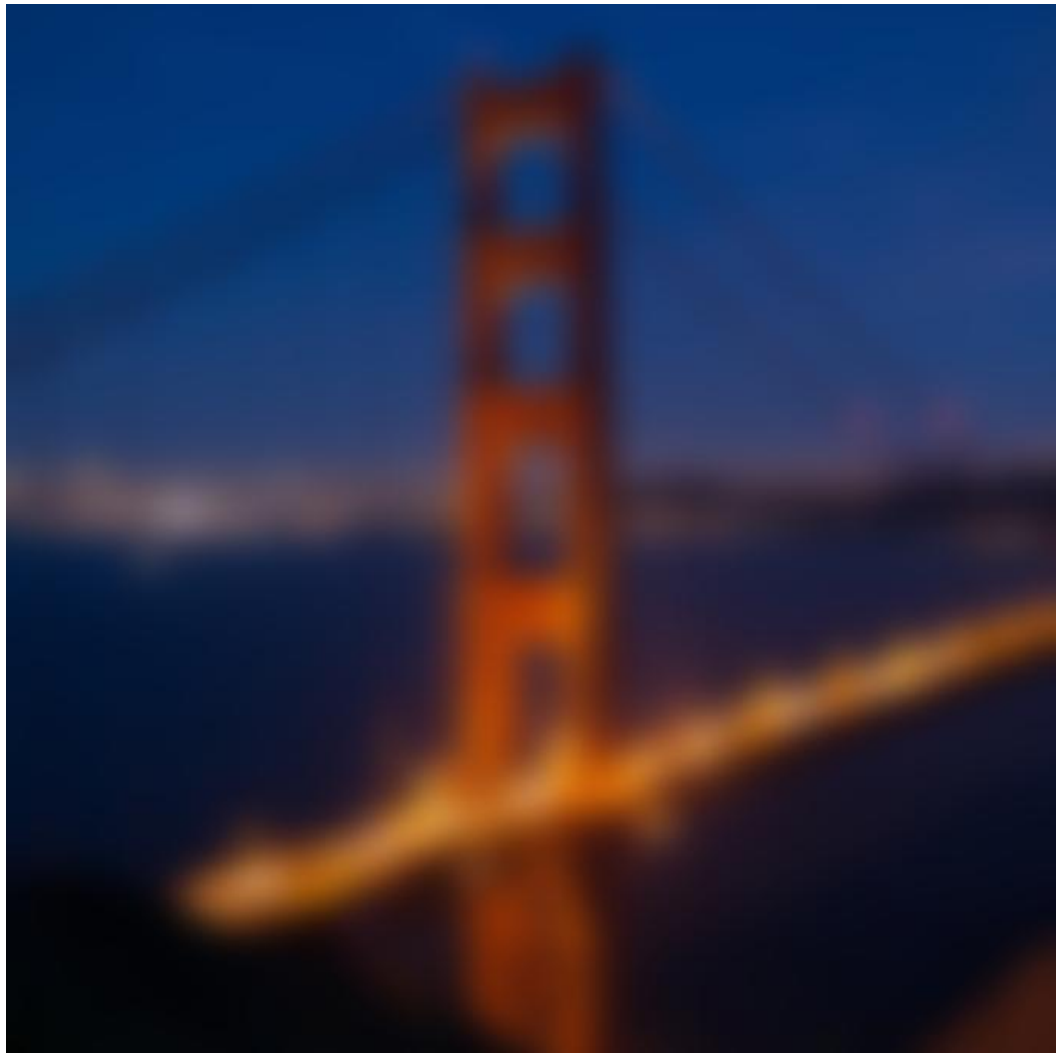
If you have a CS PhD from US/UK/EU, there's a very high change that Gauss is your ancestor



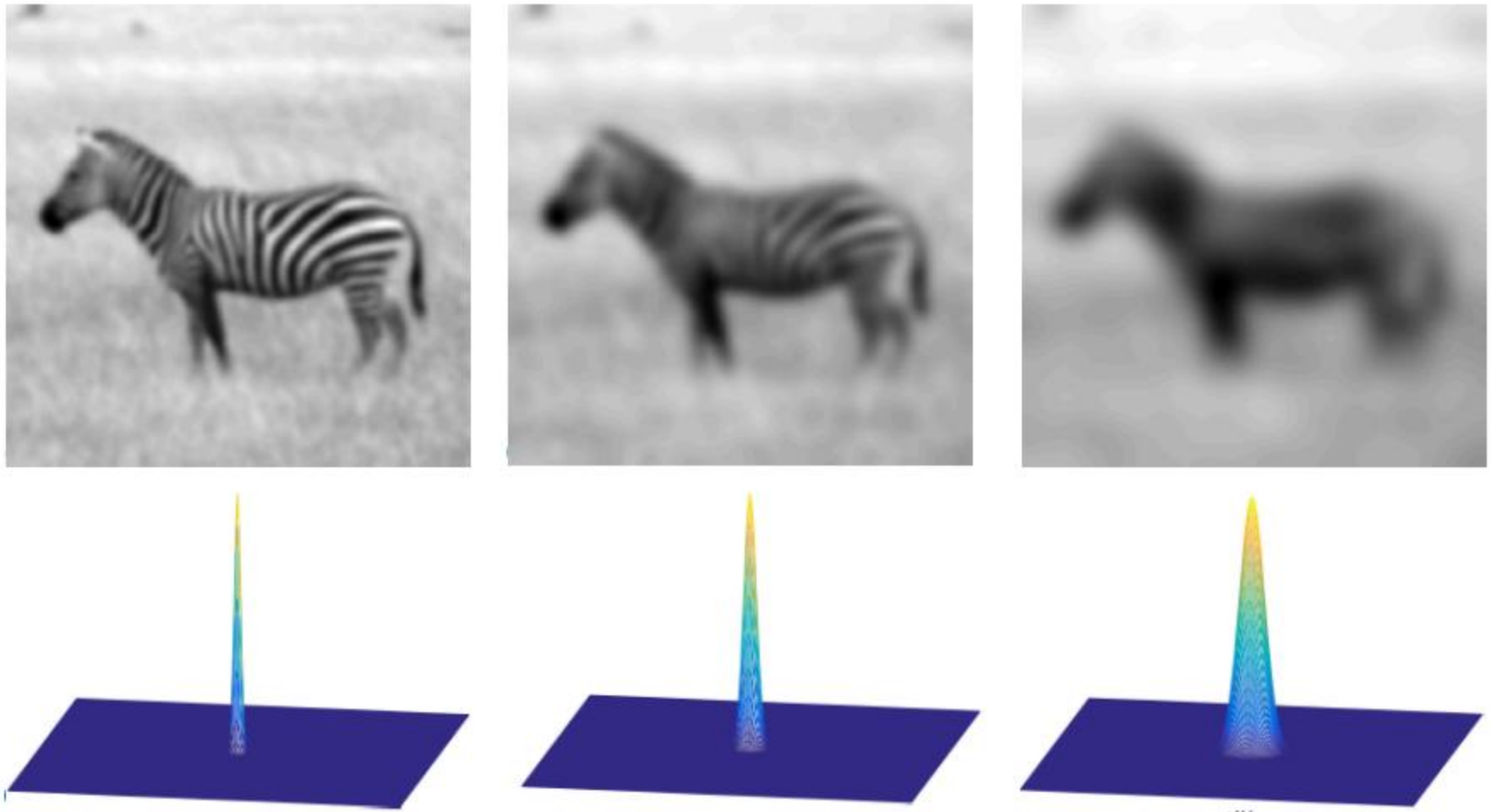




Gaussian filtering example

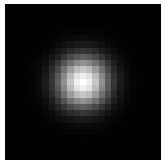


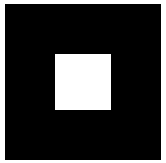
Scale

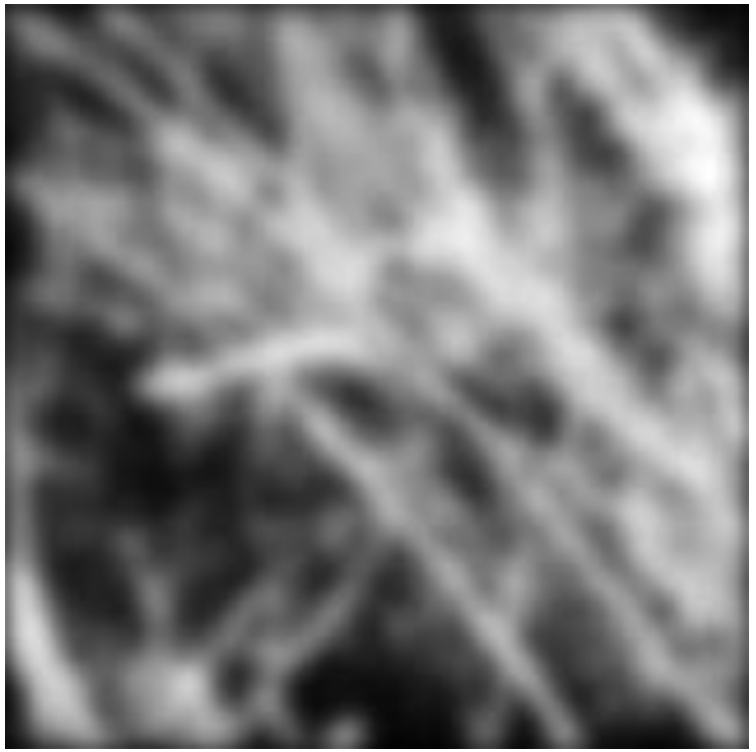


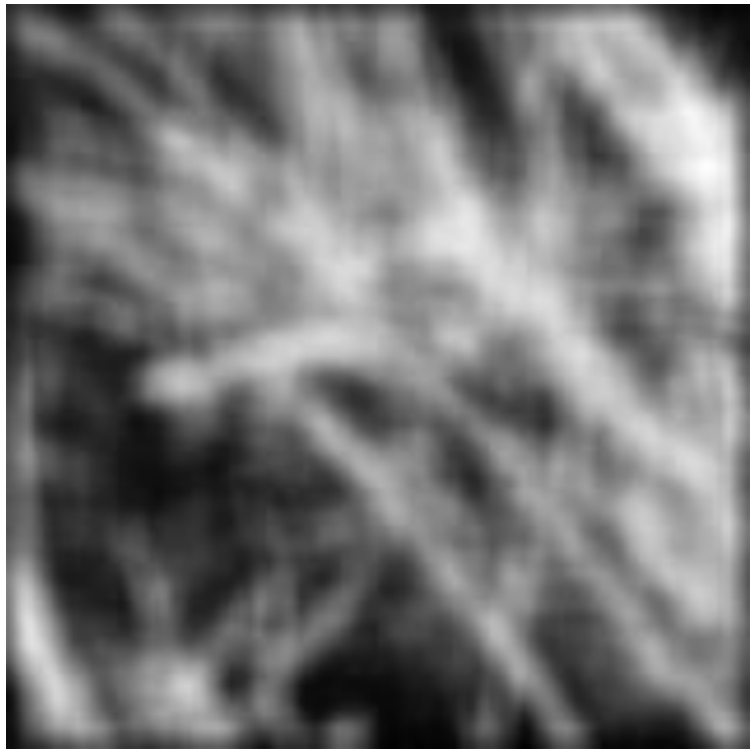
Slide from Antonio Torralba

Smoothing with Gaussian vs Box filter

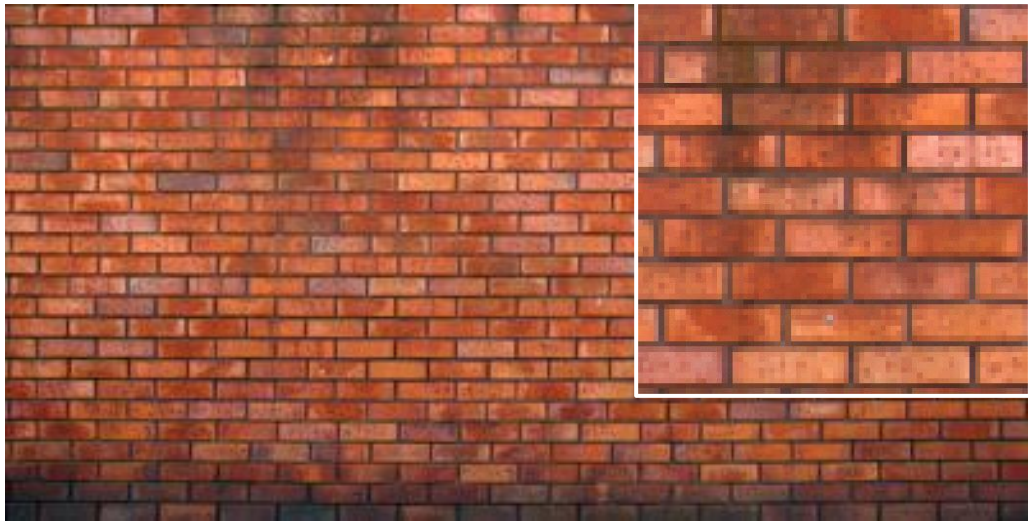




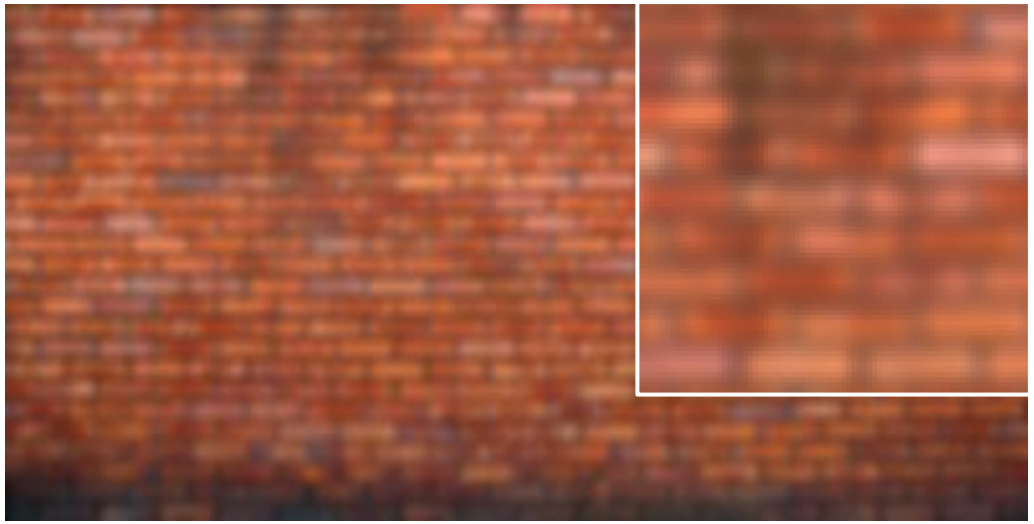




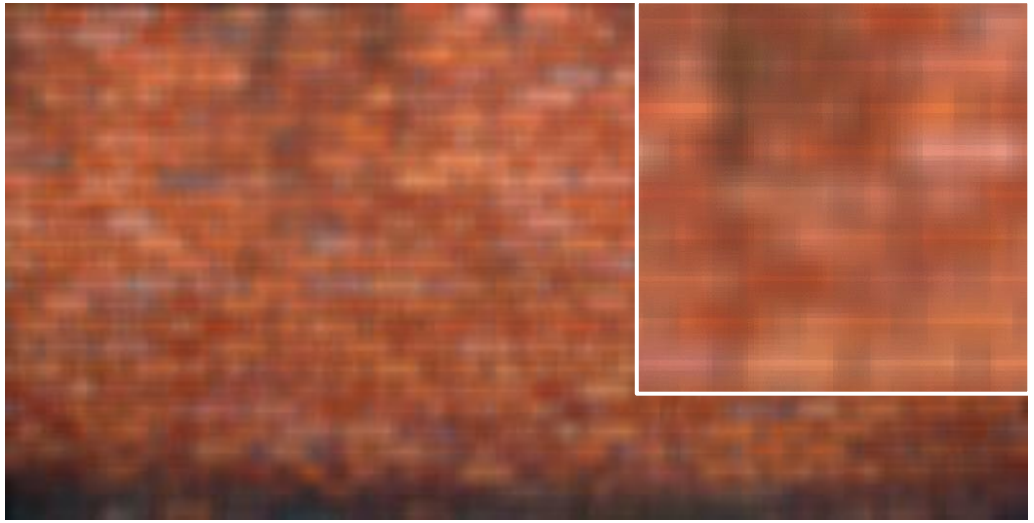
Smoothing with Gaussian vs Box filter



original



7x7 Gaussian



7x7 box

Which blur do you like better?

Edge Detection in Images

Edges are useful image features

People saying I can't make memes using edge detection

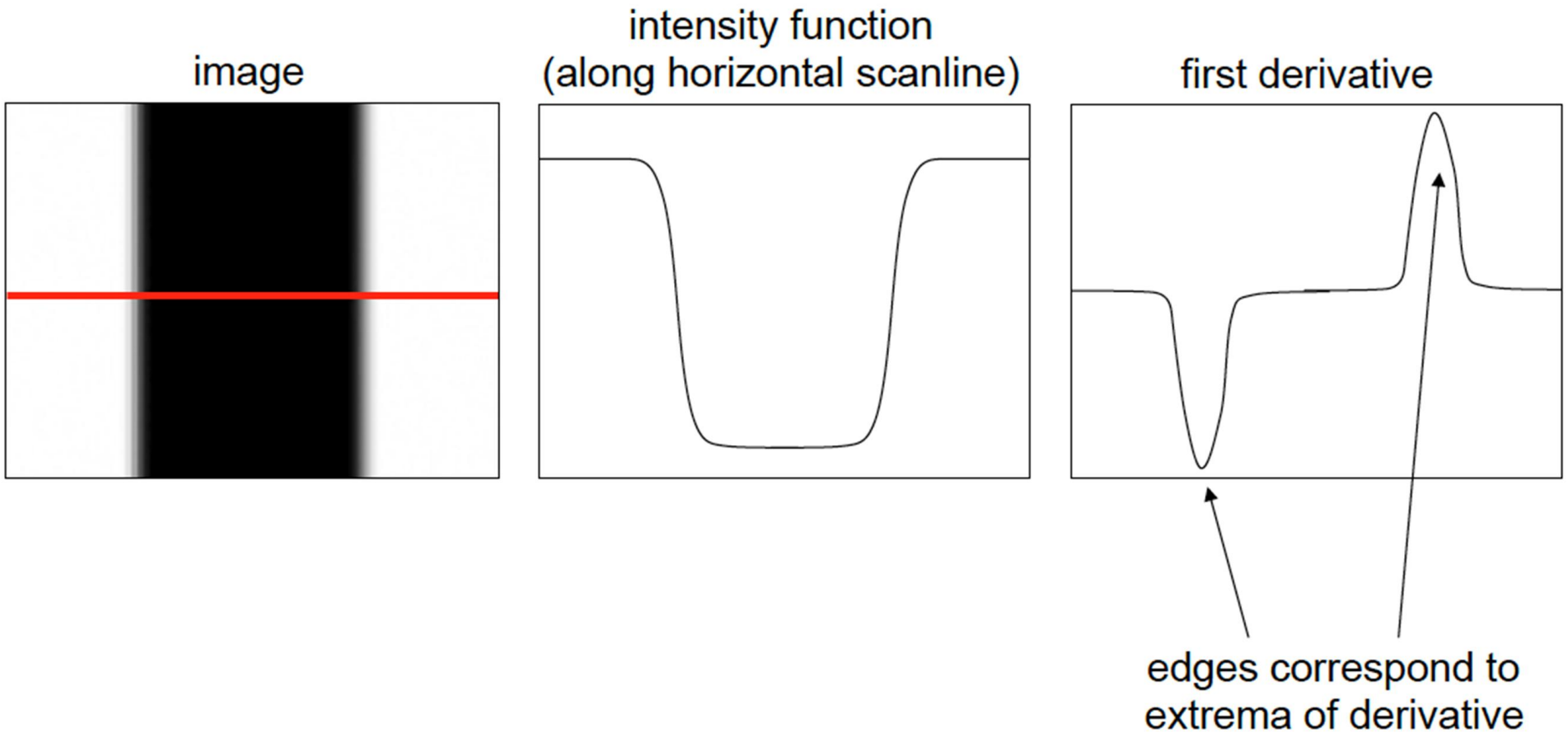
Me:

What are Image Edges?

- Sudden changes in image intensity
- Most shape information is in edges
- Shape contains a lot of semantic information (cats vs humans vs chairs vs apples)
- Edges are more compact than pixels (number of bits …)

What are Image Edges?

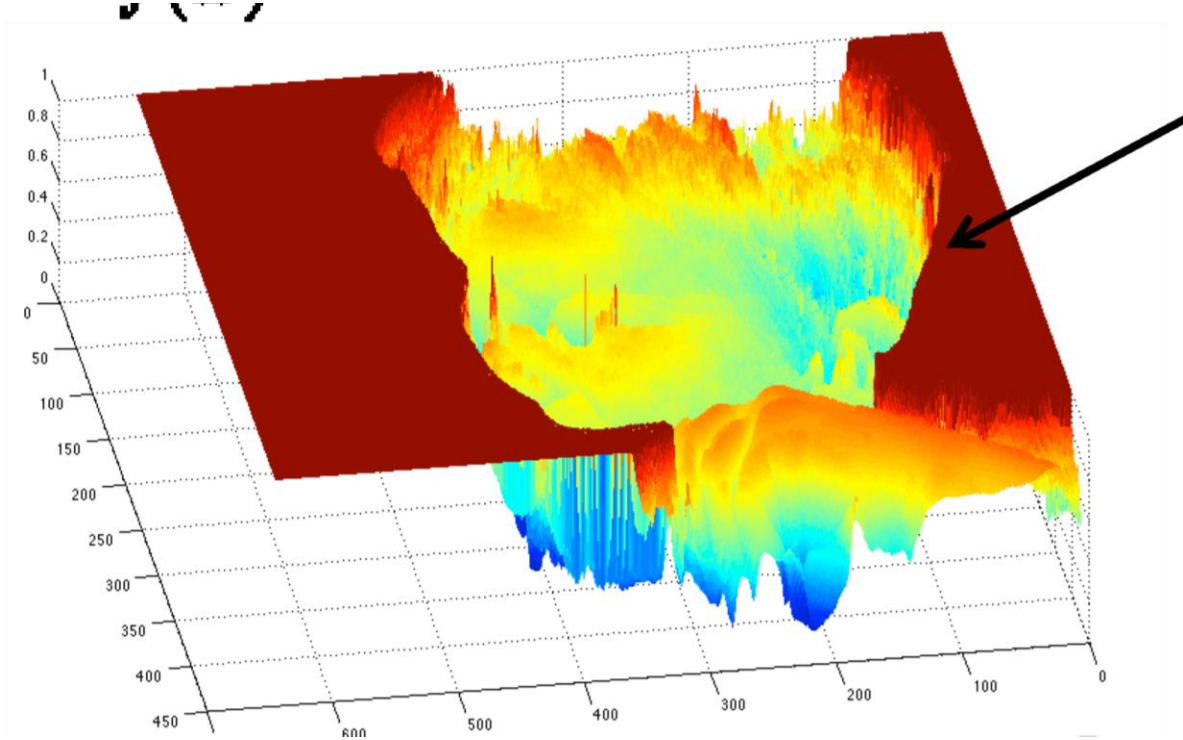
Edges = locations of rapid change in signal (image) intensity



Gradients!

- To locate rapid changes, compute gradients (first derivative)

grayscale image



Very sharp discontinuities in intensity.

Gradients in Practice: Finite Difference

- Forward difference

$$\frac{\partial f(x)}{\partial x} = \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

- Backward difference

$$\frac{\partial f(x)}{\partial x} = \frac{f(x) - f(x - \Delta x)}{\Delta x}$$

- Central difference

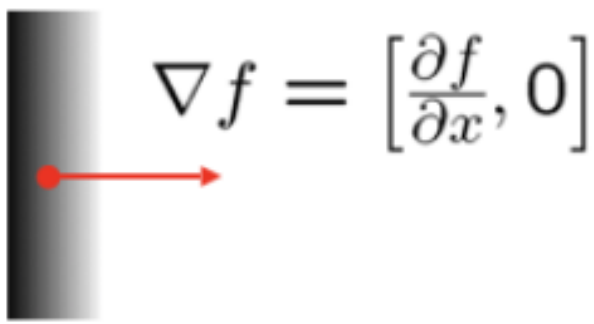
$$\frac{\partial f(x)}{\partial x} = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x}$$

Image gradient

The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Edge Detection using Finite Difference

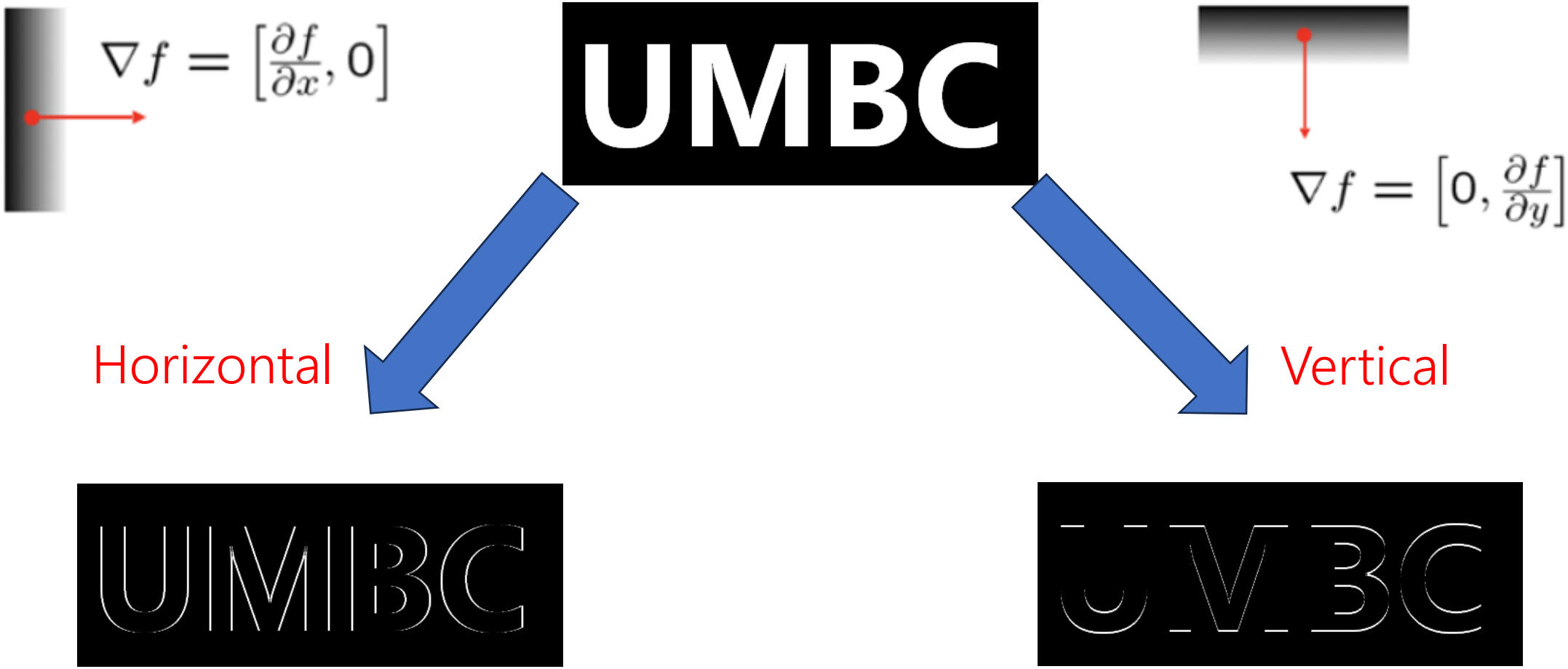
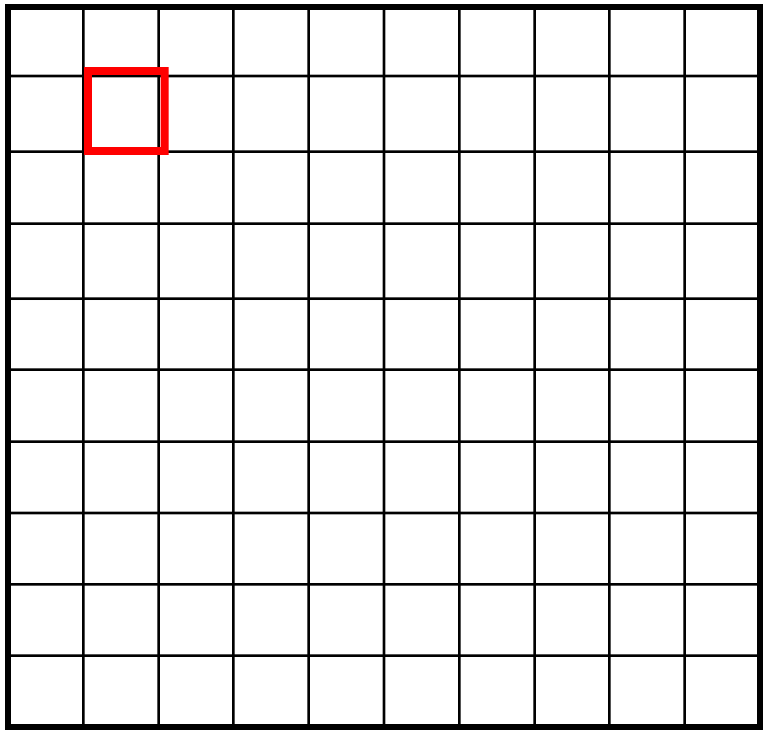


Image filtering

$g[\cdot,\cdot]$

$f[.,.]$

$h[.,.]$



$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Repeat with a different filter

$g[\cdot\,,\cdot\,]$

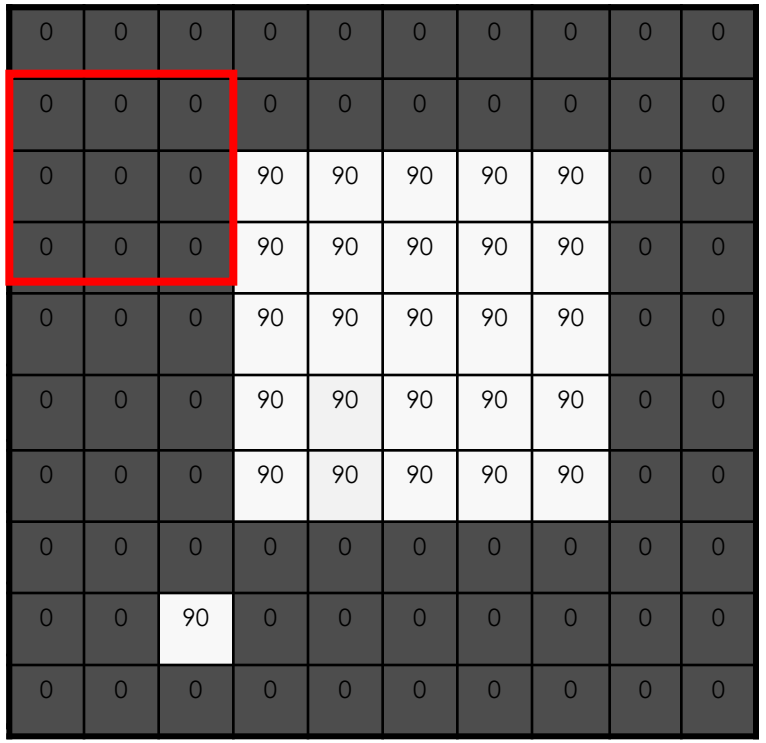
1	0	-1
1	0	-1
1	0	-1

Image filtering

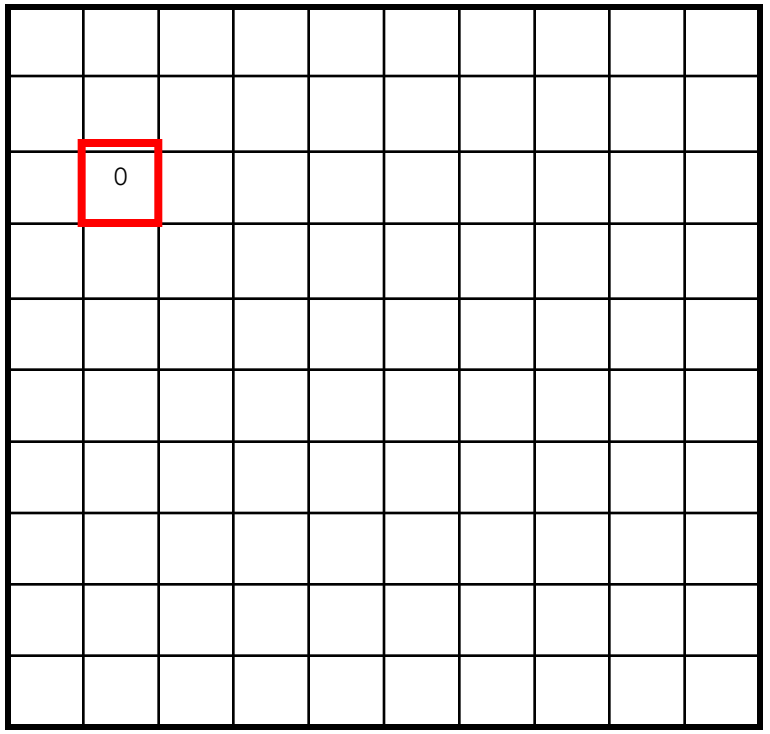
$g[\cdot,\cdot]$

1	0	-1
1	0	-1
1	0	-1

$f[.,.]$



$h[.,.]$



$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

$g[\cdot,\cdot]$

1	0	-1
1	0	-1
1	0	-1

$f[.,.]$

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	90	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

$h[.,.]$

	0	-180							

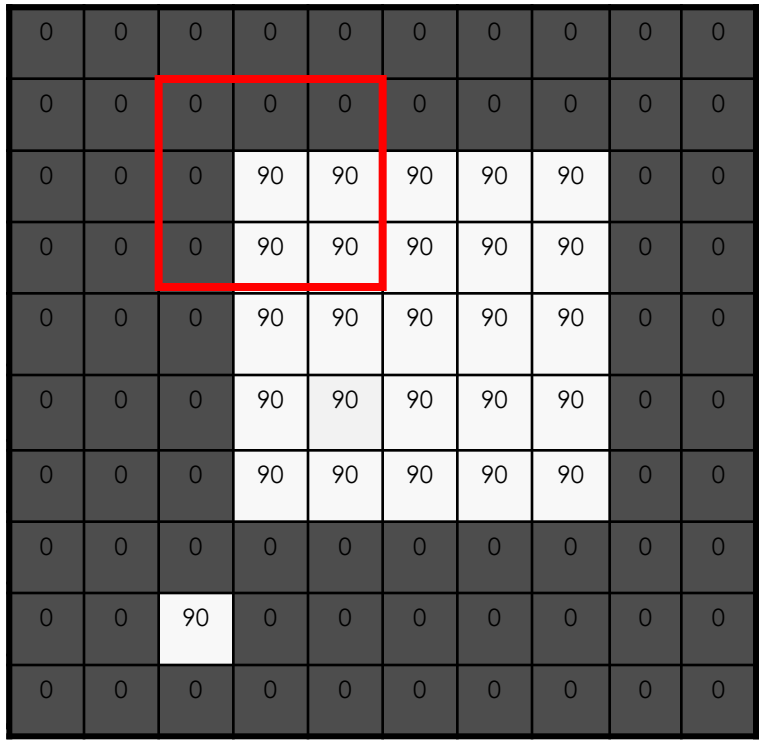
$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

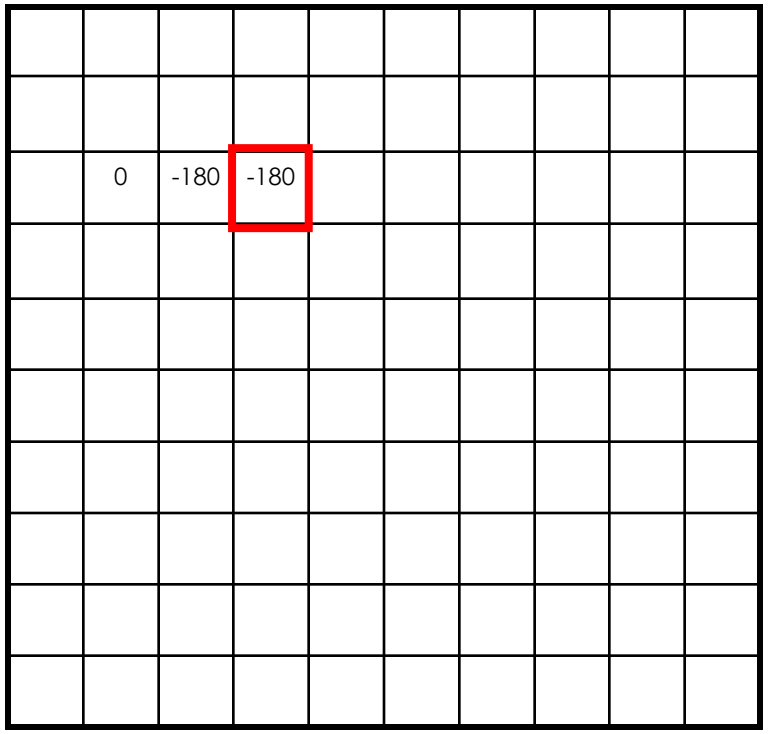
$g[\cdot,\cdot]$

1	0	-1
1	0	-1
1	0	-1

$f[.,.]$



$h[.,.]$



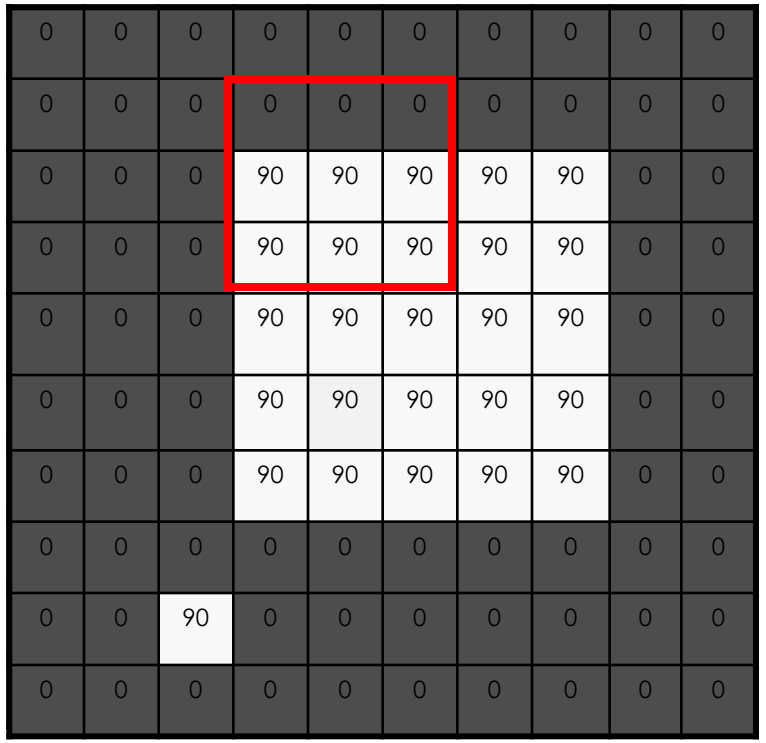
$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

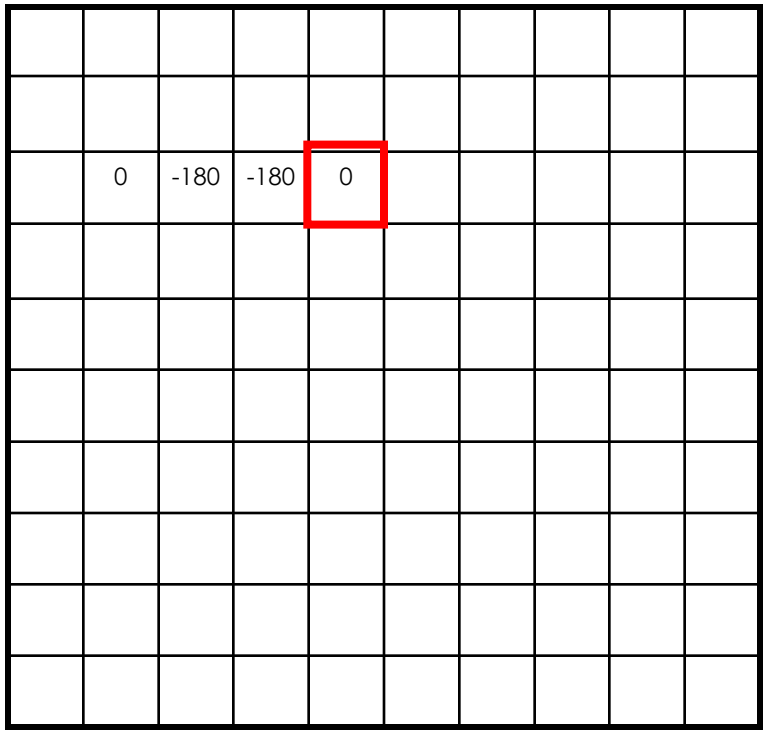
$g[\cdot,\cdot]$

1	0	-1
1	0	-1
1	0	-1

$f[.,.]$



$h[.,.]$



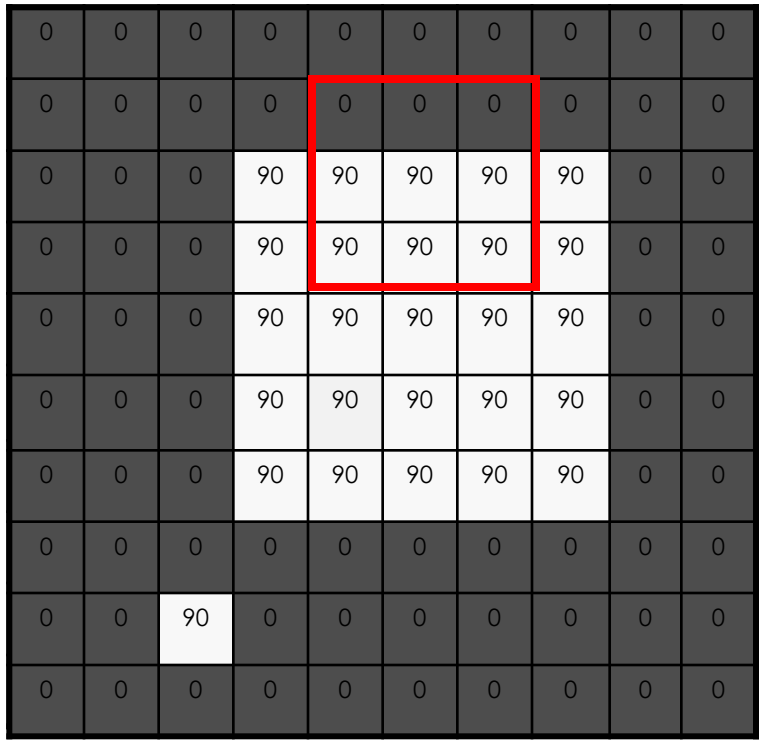
$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

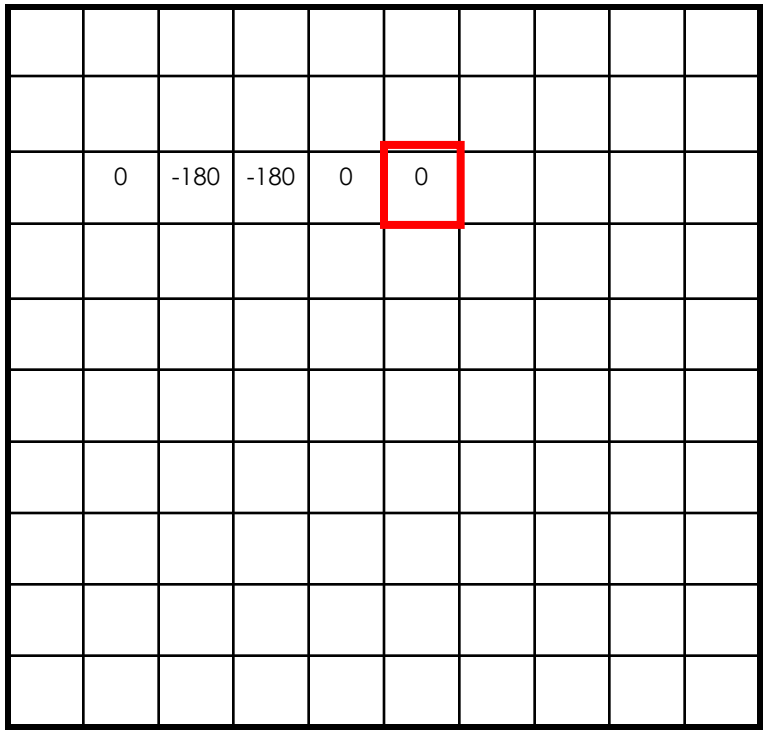
$g[\cdot,\cdot]$

1	0	-1
1	0	-1
1	0	-1

$f[.,.]$



$h[.,.]$



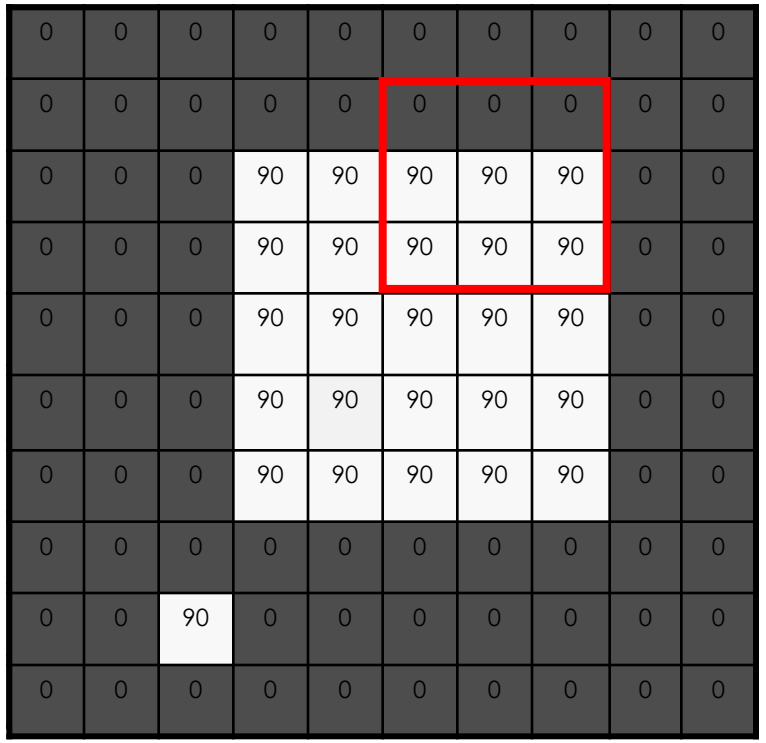
$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

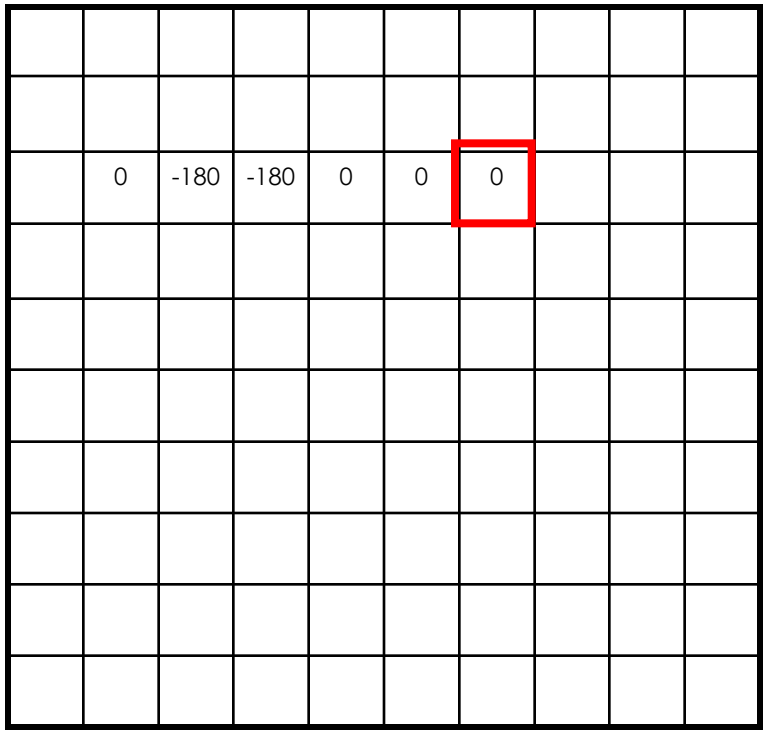
$g[\cdot,\cdot]$

1	0	-1
1	0	-1
1	0	-1

$f[.,.]$



$h[.,.]$



$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

$g[\cdot,\cdot]$

1	0	-1
1	0	-1
1	0	-1

$f[.,.]$

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	90	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

$h[.,.]$

	0	-180	-180	0	0	0	180	180	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Image filtering

$g[\cdot,\cdot]$

1	0	-1
1	0	-1
1	0	-1

$f[.,.]$

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	90	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

$h[.,.]$

	0	-180	-180	0	0	0	180	180	
	0	-270	-270	0	0	0	270	270	
	0	-270	-270	0	0	0	270	270	
	0	-270	-270	0	0	0	270	270	
	0	-180	-180	0	0	0	180	180	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Image filtering

EDGE DETECTION FILTER !!!

$g[\cdot,\cdot]$

1	0	-1
1	0	-1
1	0	-1

$f[.,.]$

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	90	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

$h[.,.]$

	0	-180	-180	0	0	0	180	180	
	0	-270	-270	0	0	0	270	270	
	0	-270	-270	0	0	0	270	270	
	0	-270	-270	0	0	0	270	270	
	0	-180	-180	0	0	0	180	180	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

We can do edge detection via convolution

For example, with this filter:

1	0	-1
1	0	-1
1	0	-1

There are other variants too ...

Edge Detection via Convolution

Sobel Filter

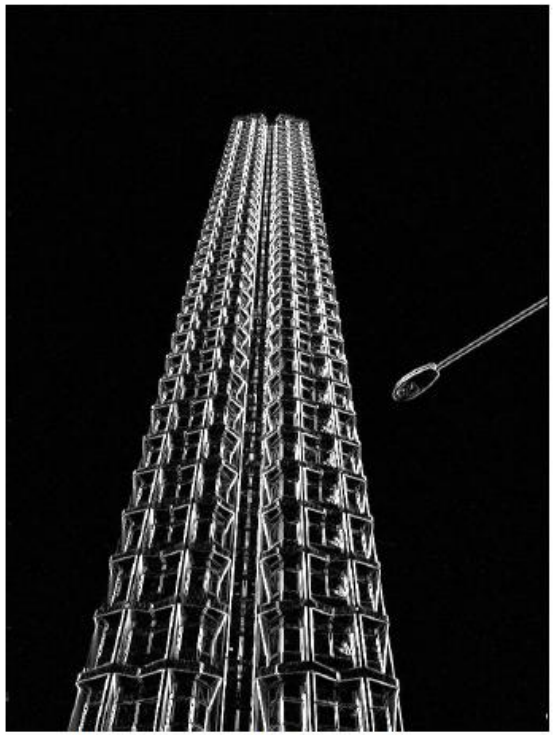
Horizontal Sober filter:

1	0	-1
2	0	-2
1	0	-1

Vertical Sobel filter:

1	2	1
0	0	0
-1	-2	-1

original



horizontal Sobel filter

vertical Sobel filter

Edge Detection via Convolution

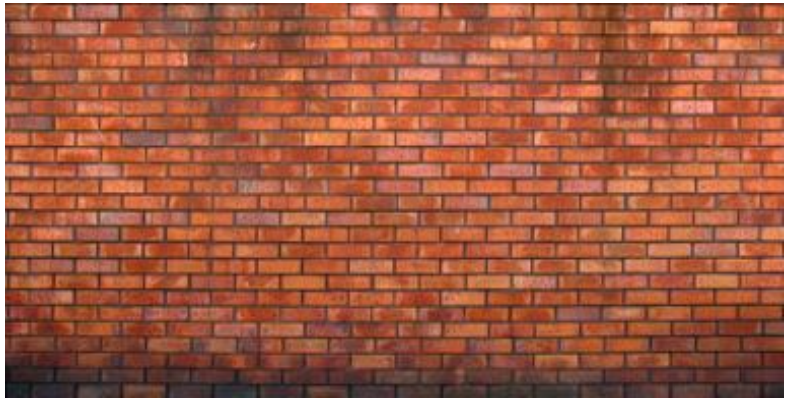
Sobel Filter

Horizontal Sober filter:

1	0	-1
2	0	-2
1	0	-1

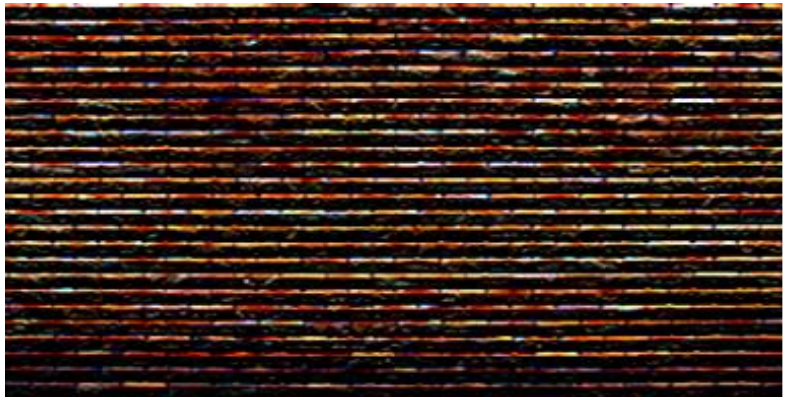
Vertical Sobel filter:

1	2	1
0	0	0
-1	-2	-1



original

horizontal Sobel filter



vertical Sobel filter

Many Types of Edge Filters

Sobel

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Roberts

0	1
-1	0

1	0
0	-1

Computing image gradients

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \boldsymbol{S}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

2. Convolve with the image to compute derivatives.

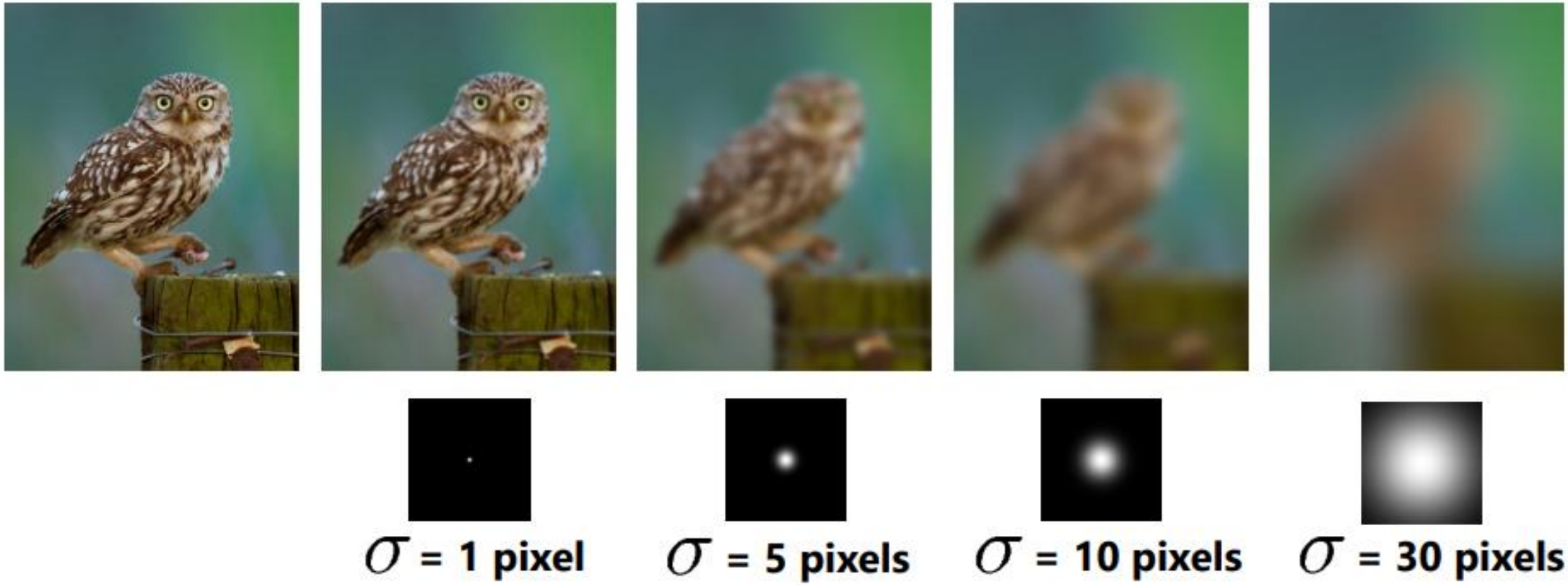
$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

3. Form the image gradient, and compute its direction and amplitude.

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \qquad \|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

gradient direction amplitude

Gaussian Filter Revisited (Smoothing / Blurring)



Sharpening Filter

- What does blurring take away?

(This "detail extraction" operation is also called a ***high-pass filter***)

Let's add it back:

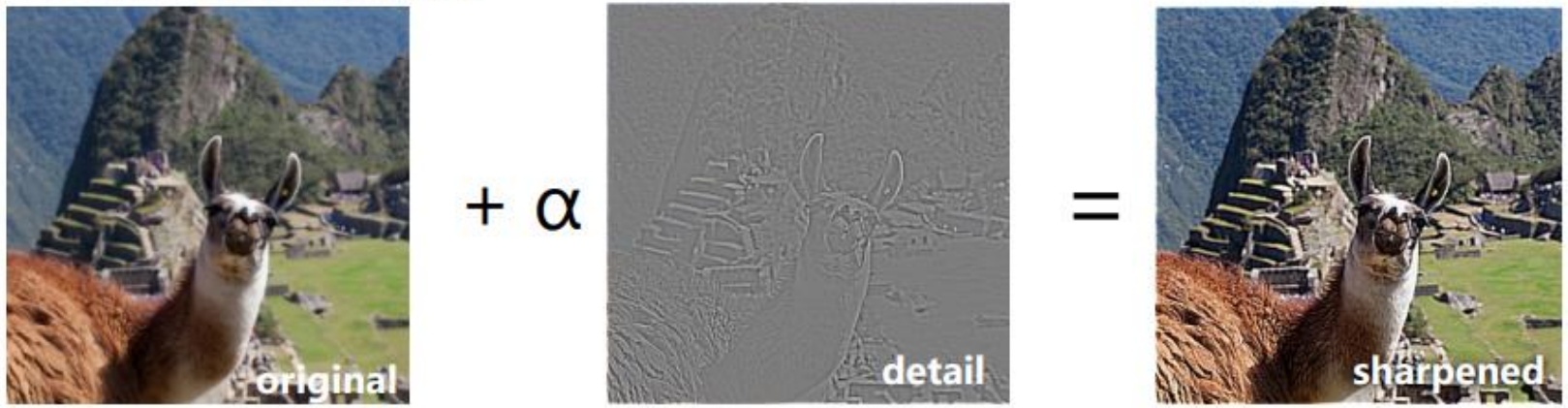
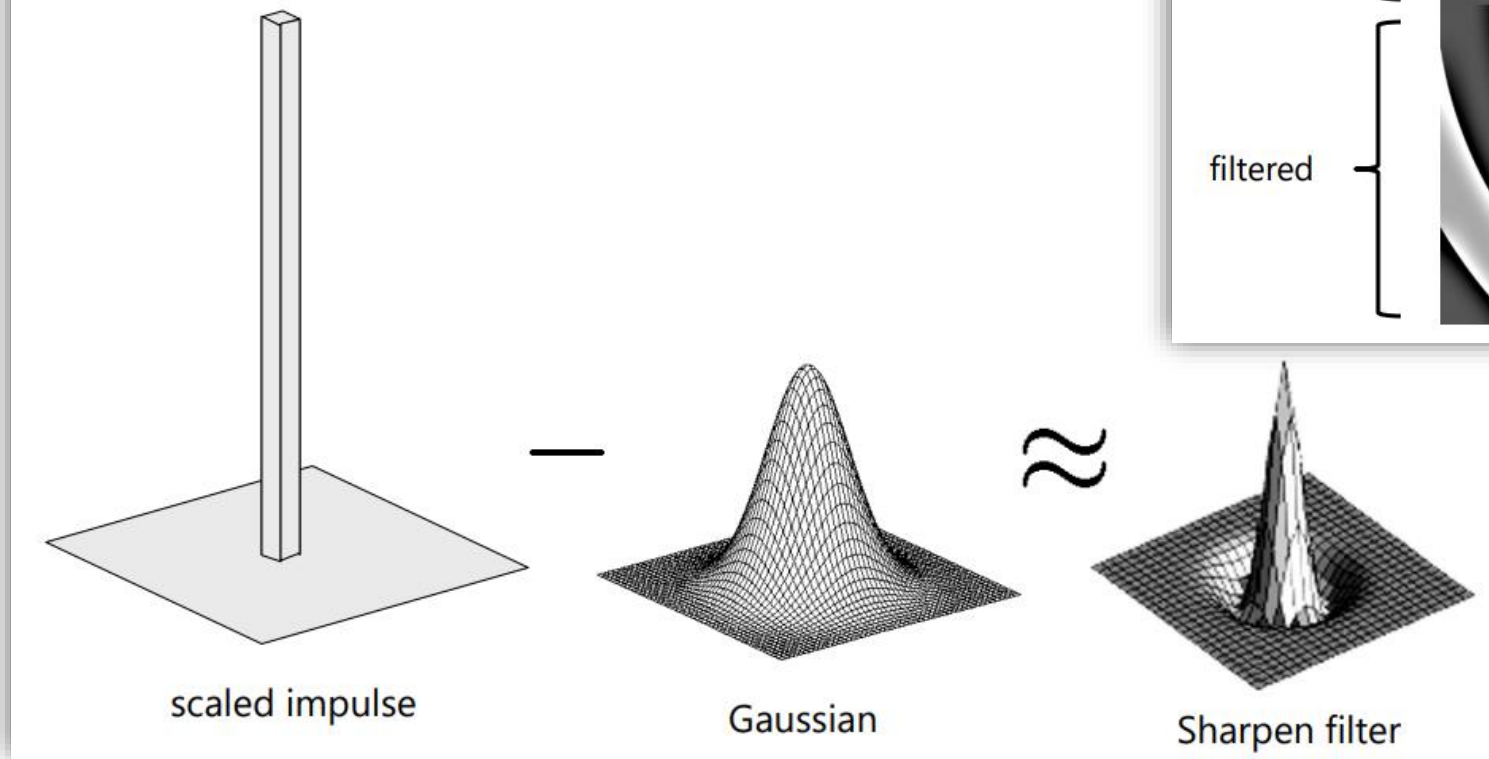


Photo credit: https://www.flickr.com/photos/geezaweezer/16089096376/

Sharpening Filter

$$F + \alpha\,(F - \underbrace{F * H}_{\text{blurred image}}) =$$

image



Thresholding

$$g(m,n) = \begin{cases} 255, & f(m,n) > A \\ 0 & otherwise \end{cases}$$

Question:

Does thresholding make storage efficient?

By how much (in this example)?