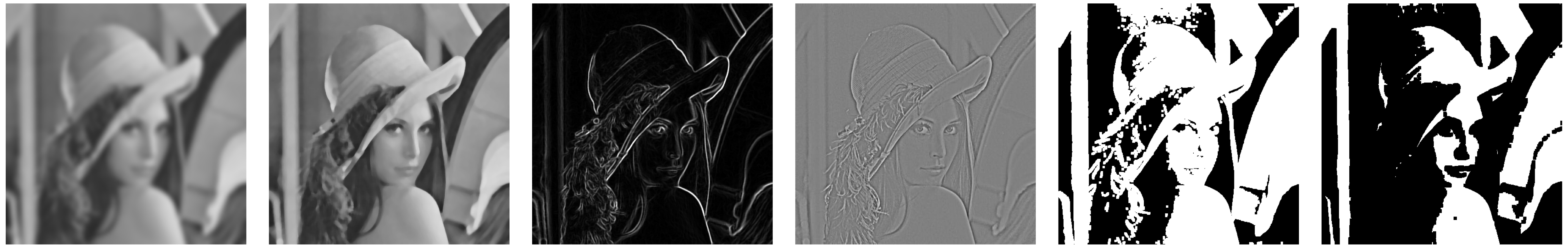
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 3: Image Filtering

(unless otherwise stated slides are taken or adopted from **Bob Woodham, Jim Little** and **Fred Tung**)

Goal

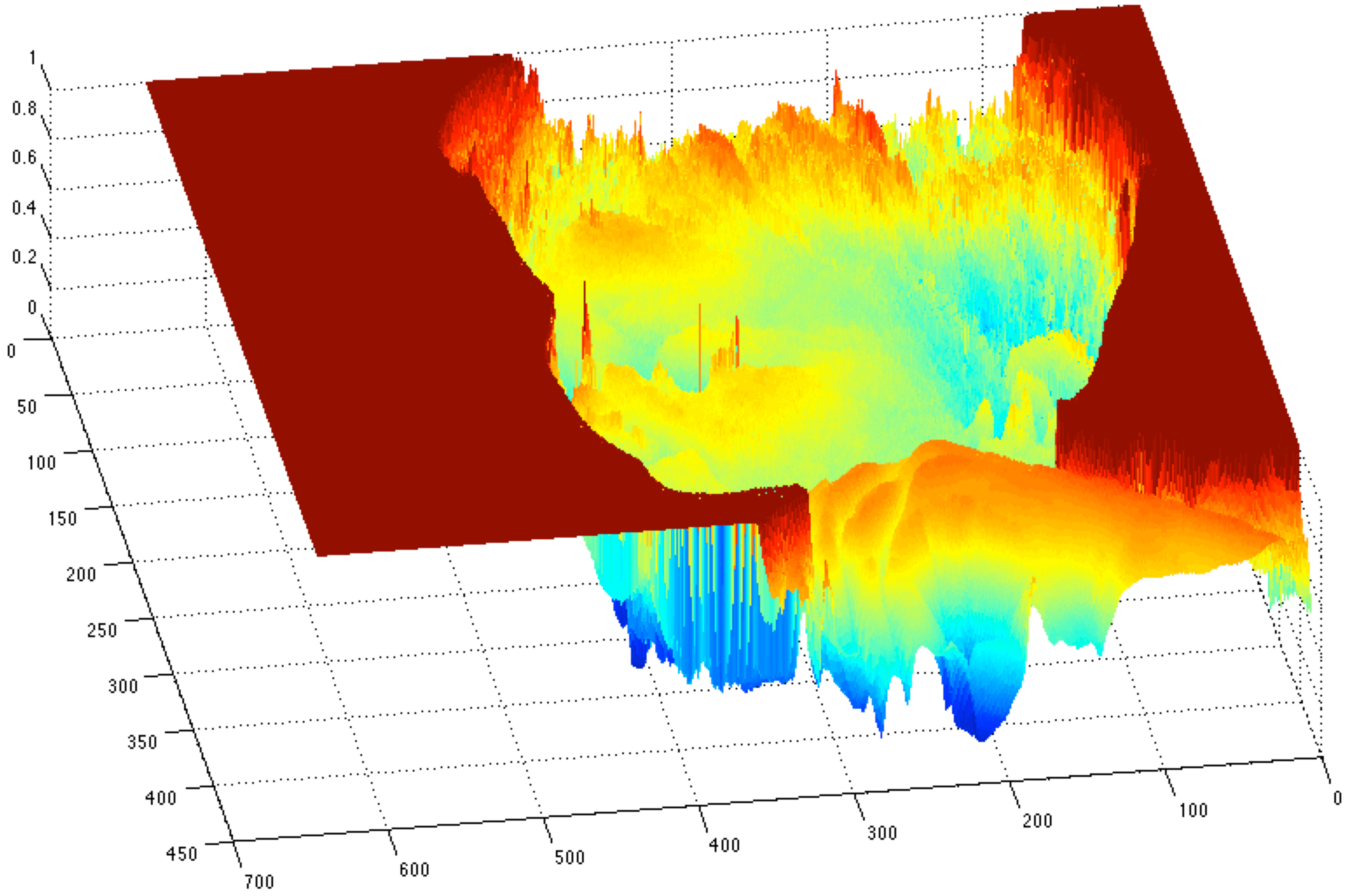
1. Learn how to mathematically describe image processing
2. Basic building blocks

Image as a **2D Function**

A (grayscale) image is a 2D function

grayscale image

$I(X, Y)$



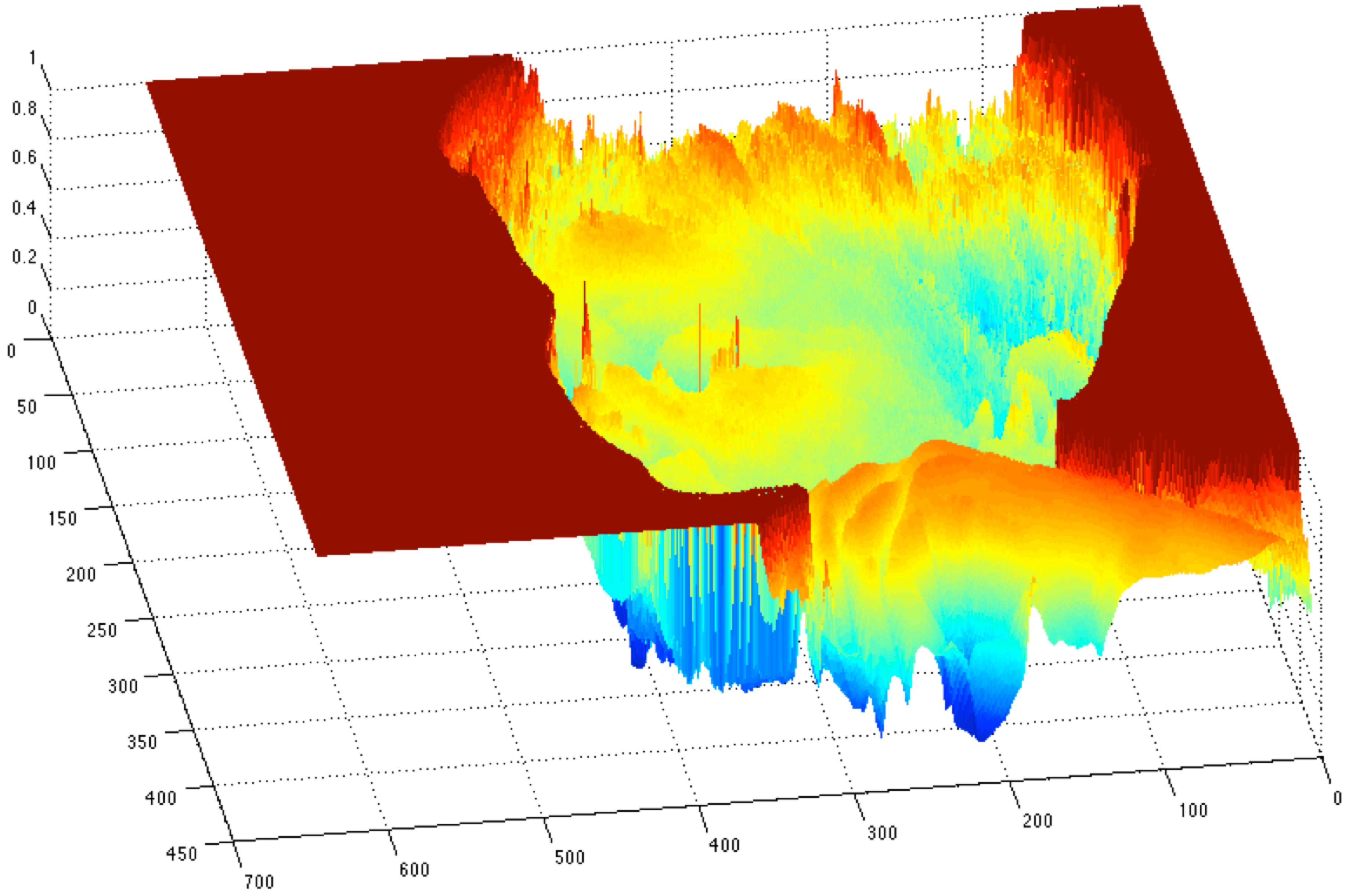
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Image as a **2D Function**

A (grayscale) image is a 2D function

grayscale image

$I(X, Y)$



domain: $(X, Y) \in ([1, width], [1, hight])$

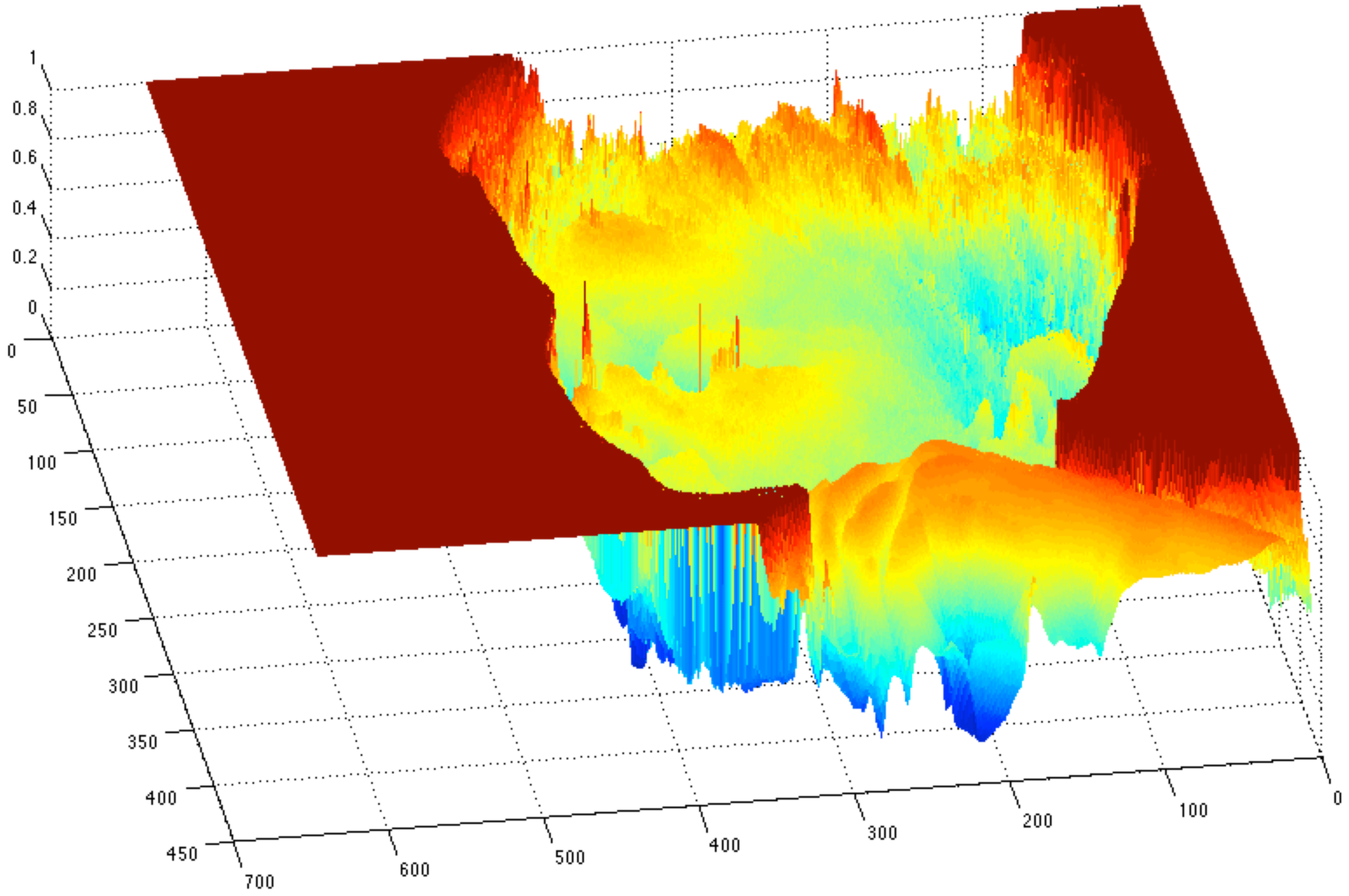
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Image as a **2D Function**

A (grayscale) image is a 2D function

grayscale image

$I(X, Y)$



What is the **range** of the image function?

domain: $(X, Y) \in ([1, width], [1, hight])$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Image as a **2D Function**

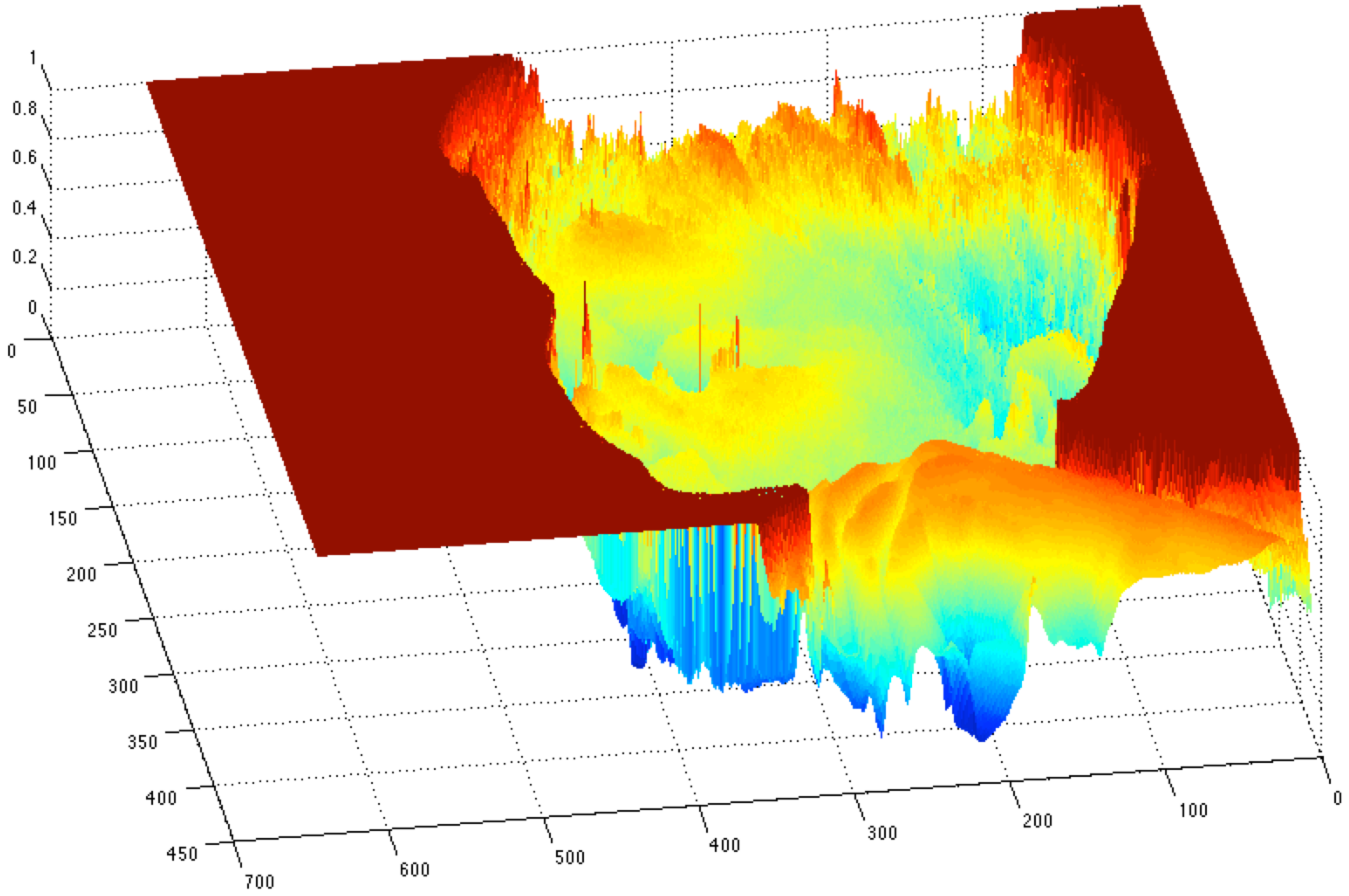
A (grayscale) image is a 2D function

grayscale image

What is the **range** of the image function?

$$I(X, Y) \in [0, 255] \in \mathbb{Z}$$

$I(X, Y)$



domain: $(X, Y) \in ([1, width], [1, hight])$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Adding two Images

Since images are functions, we can perform operations on them, e.g., **average**

$I(X,Y)$

$G(X,Y)$

$$\frac{I(X,Y)}{2} + \frac{G(X,Y)}{2}$$

Adding two Images

$$a = \frac{I(X,Y)}{2} + \frac{G(X,Y)}{2}$$

$$b = \frac{I(X,Y) + G(X,Y)}{2}$$

Adding two Images

$$a = \frac{I(X,Y)}{2} + \frac{G(X,Y)}{2}$$

$$b = \frac{I(X,Y) + G(X,Y)}{2}$$

Question:

$a = b$

$a > b$

$a < b$

Adding two Images

Red pixel in camera man image = 98

Red pixel in moon image = 200

$$\frac{98}{2} + \frac{200}{2} = 49 + 100 = 149$$

$$\frac{98 + 200}{2} = \frac{\lfloor 298 \rfloor}{2} = \frac{255}{2} = 127$$

Question:

$a = b$

$a > b$

$a < b$

Adding two Images

It is often convenient to convert images to **doubles** when doing processing

In Python

```
from PIL import Image
img = Image.open('cameraman.png')
import numpy as np
imgArr = np.asfarray(img)

# Or do this
import matplotlib.pyplot as plt
camera = plt.imread('cameraman.png');
```

Adding two Images

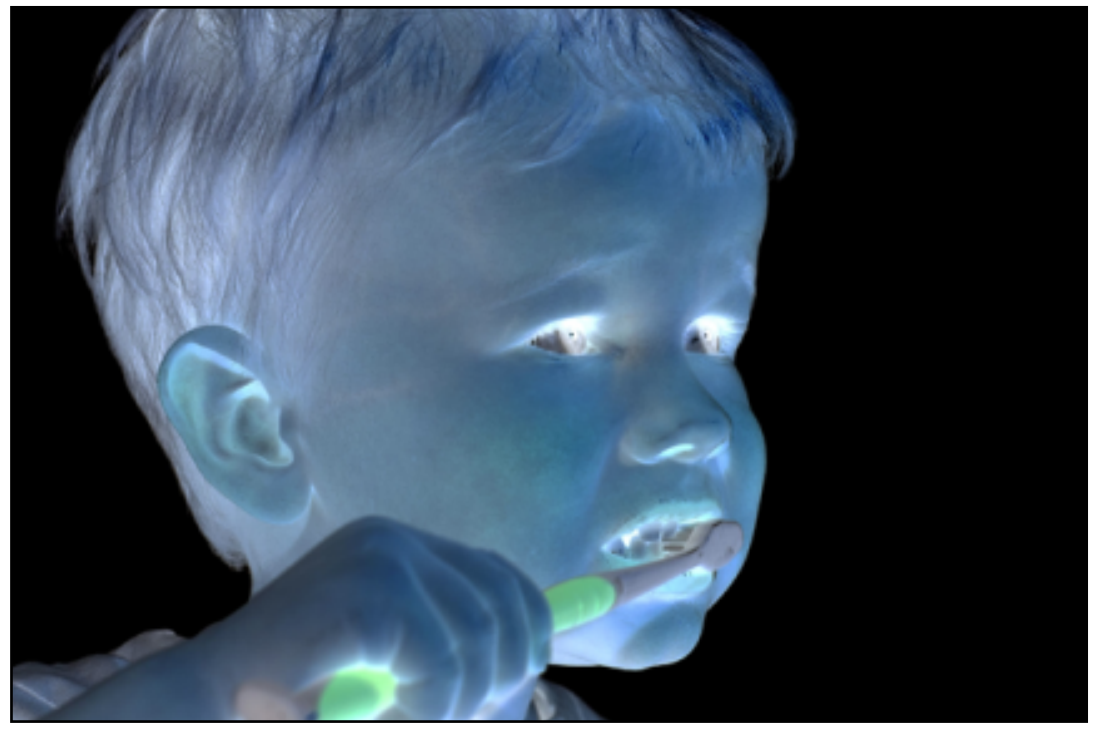
This will save you a **LOT** of headache in homeworks:

1. Convert to **doubles**
2. (optionally) Normalize image to [0,1] range (by dividing by 255)
3. Perform any **computations** needed
4. (optionally) Undo normalization (by multiplying by 255)
5. **Clamp** values between [0, 255]
6. Convert to **uint8**

What types of **transformations** can we do?

$I(X,Y)$

Filtering

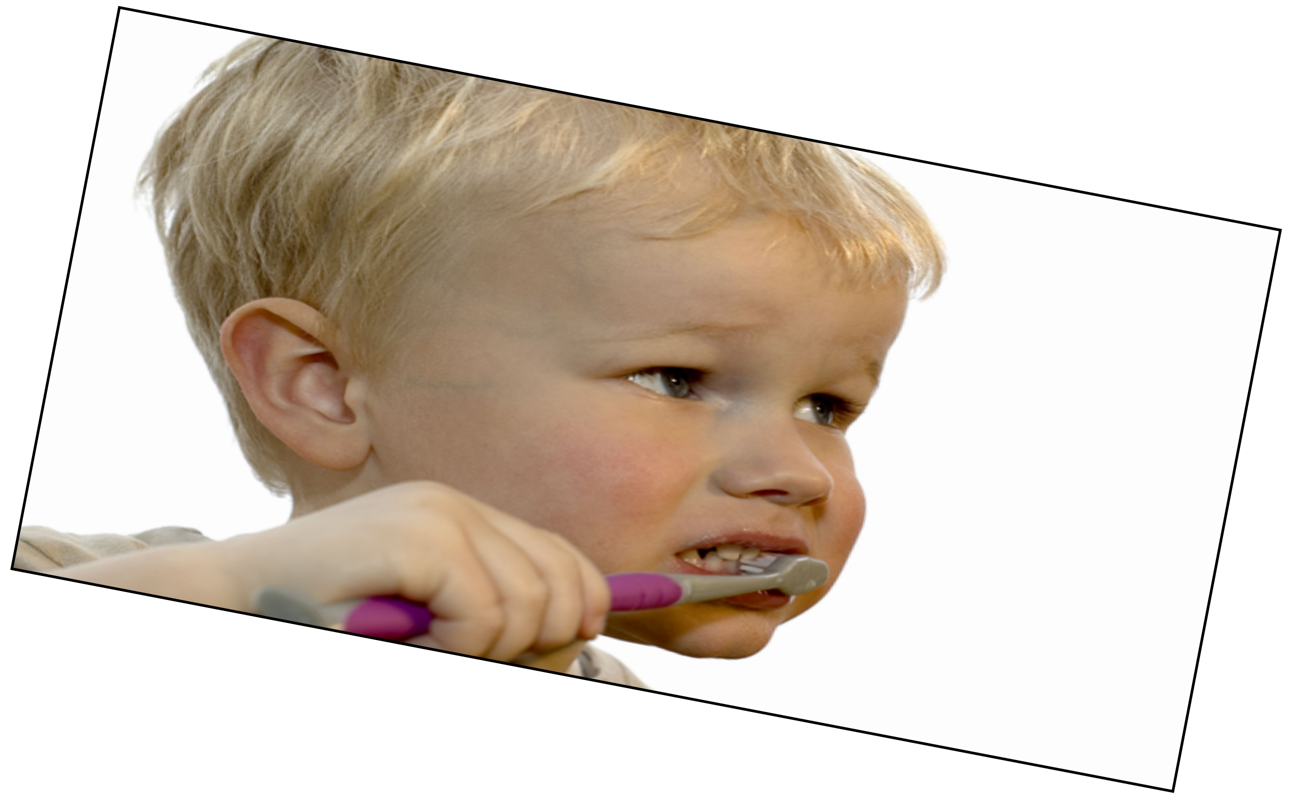


$I'(X,Y)$

changes range of image function

$I(X,Y)$

Warping



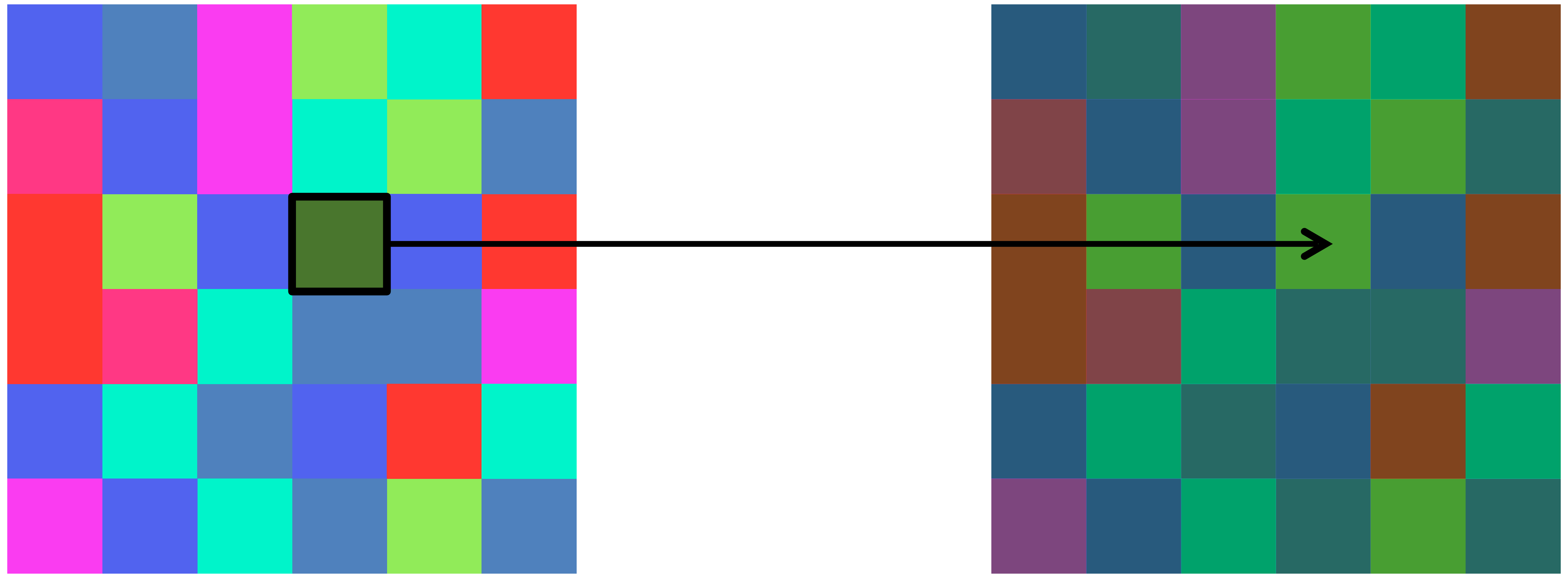
$I'(X,Y)$

changes domain of image function

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

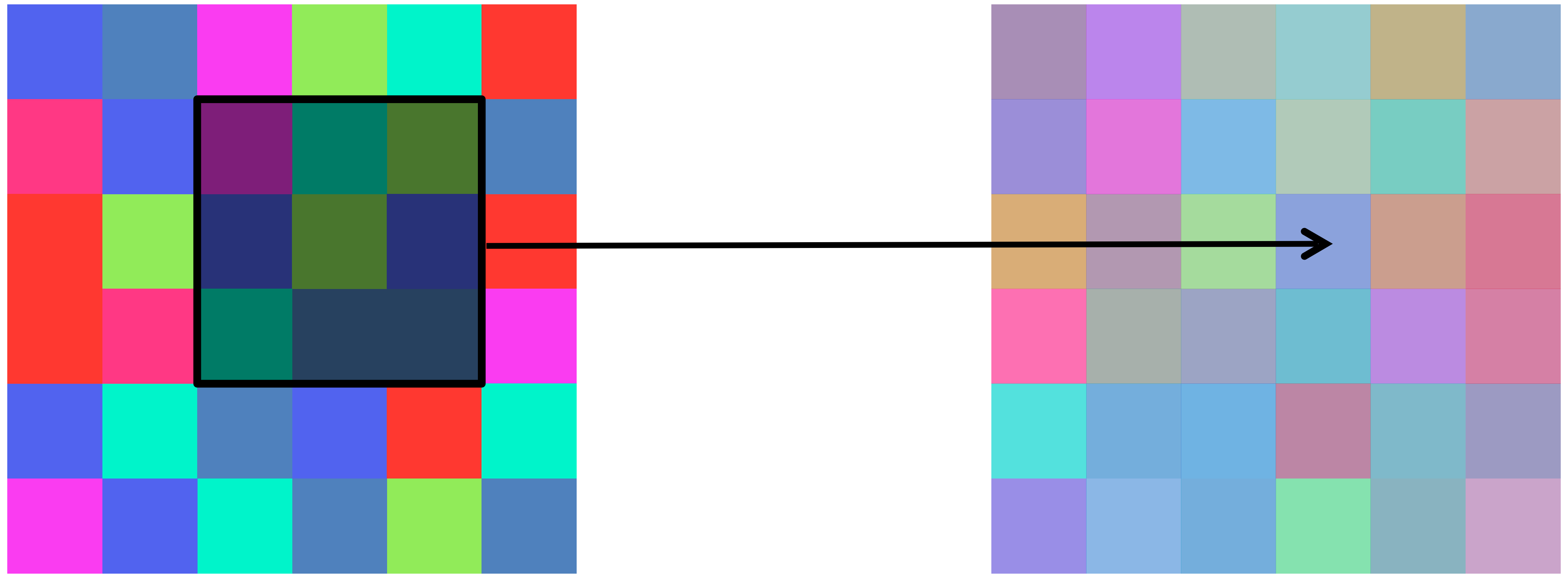
What types of **filtering** can we do?

Point Operation



point processing

Neighborhood Operation



“filtering”

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Examples of **Point Processing**

original

$I(X, Y)$

darken

lower contrast

non-linear lower contrast

invert

lighten

raise contrast

non-linear raise contrast

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Examples of **Point Processing**

original

darken

lower contrast

non-linear lower contrast

$I(X, Y)$

$I(X, Y) - 128$

invert

lighten

raise contrast

non-linear raise contrast

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Examples of **Point Processing**

original

$I(X,Y)$

darken

$I(X,Y) - 128$

lower contrast

$\frac{I(X,Y)}{2}$

non-linear lower contrast

invert

lighten

raise contrast

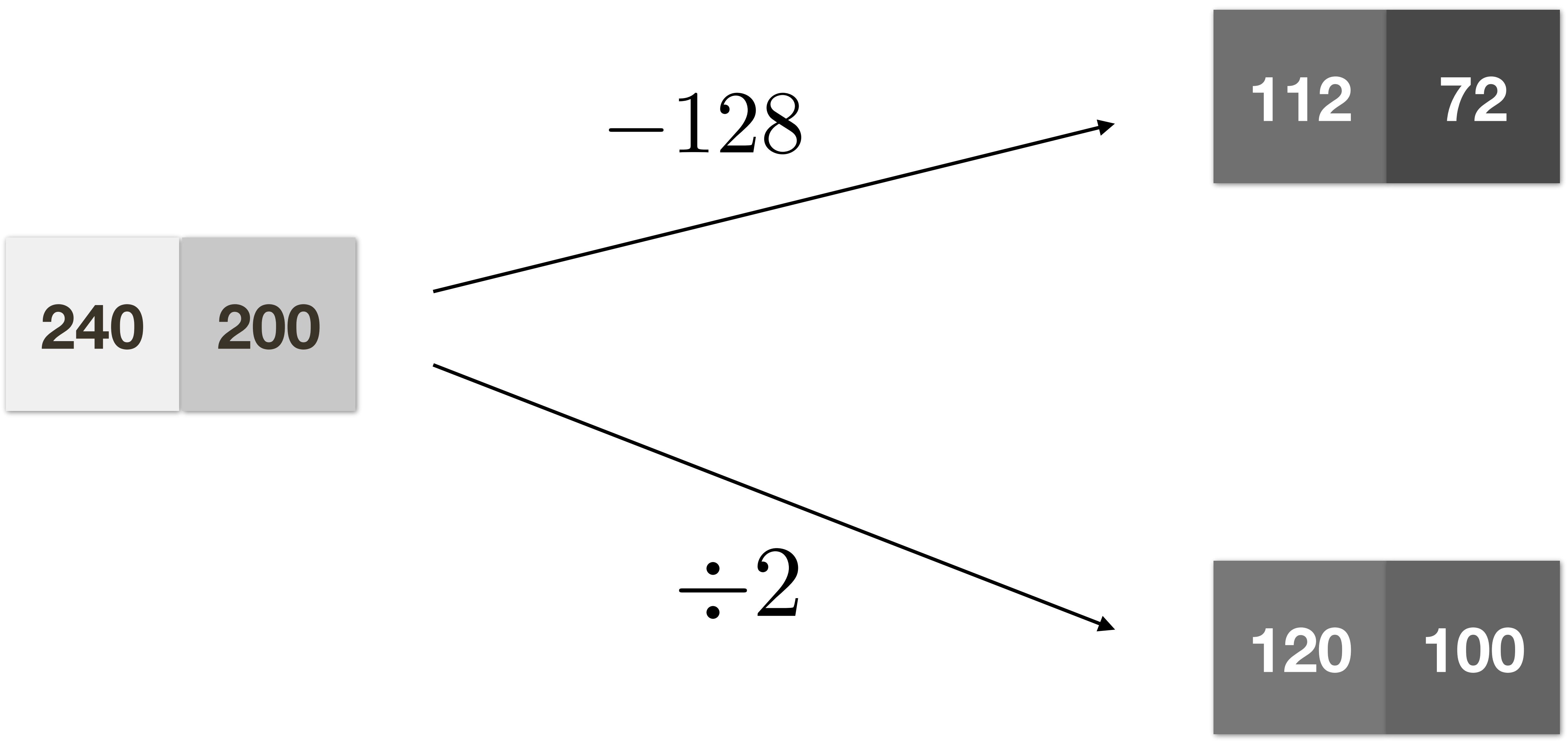
non-linear raise contrast

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Brightness v.s. **Contrast**

Brightness: all pixels get lighter/darker, relative difference between pixel values stays the same

Contrast: relative difference between pixel values becomes higher / lower



Examples of **Point Processing**

original

darken

lower contrast

non-linear lower contrast

$I(X, Y)$

$I(X, Y) - 128$

$\frac{I(X, Y)}{2}$

invert

lighten

raise contrast

non-linear raise contrast

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Examples of **Point Processing**

original

$I(X, Y)$

darken

$I(X, Y) - 128$

lower contrast

$\frac{I(X, Y)}{2}$

non-linear lower contrast

$\left(\frac{I(X, Y)}{255}\right)^{1/3} \times 255$

invert

lighten

raise contrast

non-linear raise contrast

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Examples of **Point Processing**

original

$I(X, Y)$

darken

$I(X, Y) - 128$

lower contrast

$$\frac{I(X, Y)}{2}$$

non-linear lower contrast

$$\left(\frac{I(X, Y)}{255}\right)^{1/3} \times 255$$

invert

$255 - I(X, Y)$

lighten

raise contrast

non-linear raise contrast

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Examples of **Point Processing**

original

$$I(X,Y)$$

darken

$$I(X,Y) - 128$$

lower contrast

$$\frac{I(X,Y)}{2}$$

non-linear lower contrast

$$\left(\frac{I(X,Y)}{255}\right)^{1/3} \times 255$$

invert

$$255 - I(X,Y)$$

lighten

$$I(X,Y) + 128$$

raise contrast

non-linear raise contrast

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Examples of **Point Processing**

original

$$I(X,Y)$$

darken

$$I(X,Y) - 128$$

lower contrast

$$\frac{I(X,Y)}{2}$$

non-linear lower contrast

$$\left(\frac{I(X,Y)}{255}\right)^{1/3} \times 255$$

invert

$$255 - I(X,Y)$$

lighten

$$I(X,Y) + 128$$

raise contrast

$$I(X,Y) \times 2$$

non-linear raise contrast

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Examples of **Point Processing**

original

$$I(X,Y)$$

darken

$$I(X,Y) - 128$$

lower contrast

$$\frac{I(X,Y)}{2}$$

non-linear lower contrast

$$\left(\frac{I(X,Y)}{255}\right)^{1/3} \times 255$$

invert

$$255 - I(X,Y)$$

lighten

$$I(X,Y) + 128$$

raise contrast

$$I(X,Y) \times 2$$

non-linear raise contrast

$$\left(\frac{I(X,Y)}{255}\right)^{2} \times 255$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Examples of **Point Processing**

original

$$I(X,Y)$$

darken

$$I(X,Y) - 128$$

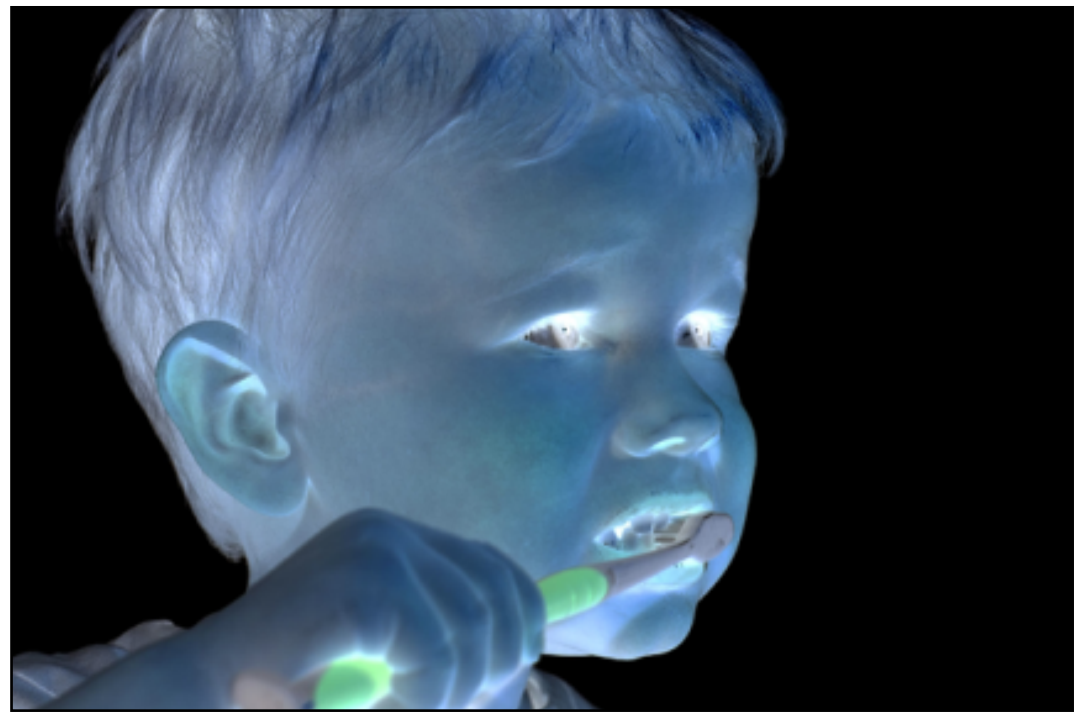
lower contrast

$$\frac{I(X,Y)}{2}$$

non-linear lower contrast

$$\left(\frac{I(X,Y)}{255}\right)^{1/3} \times 255$$

invert



$$255 - I(X,Y)$$

lighten

$$I(X,Y) + 128$$

raise contrast

$$I(X,Y) \times 2$$

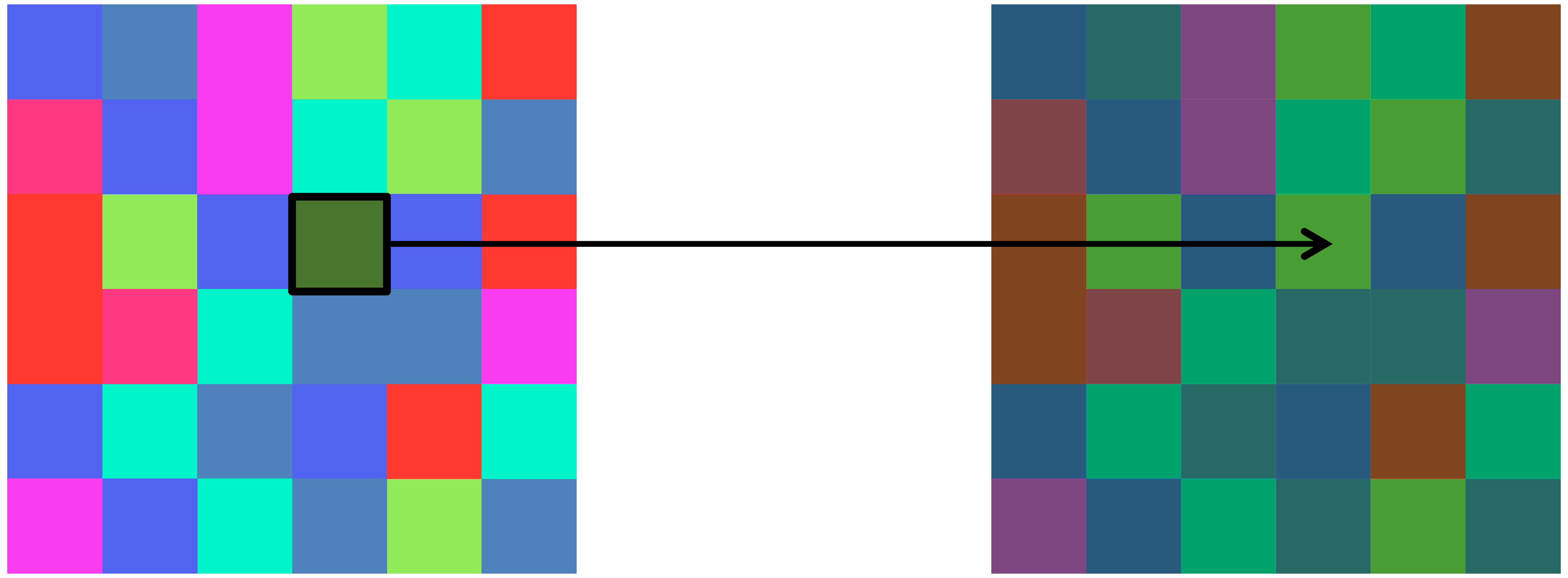
non-linear raise contrast

$$\left(\frac{I(X,Y)}{255}\right)^{2} \times 255$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

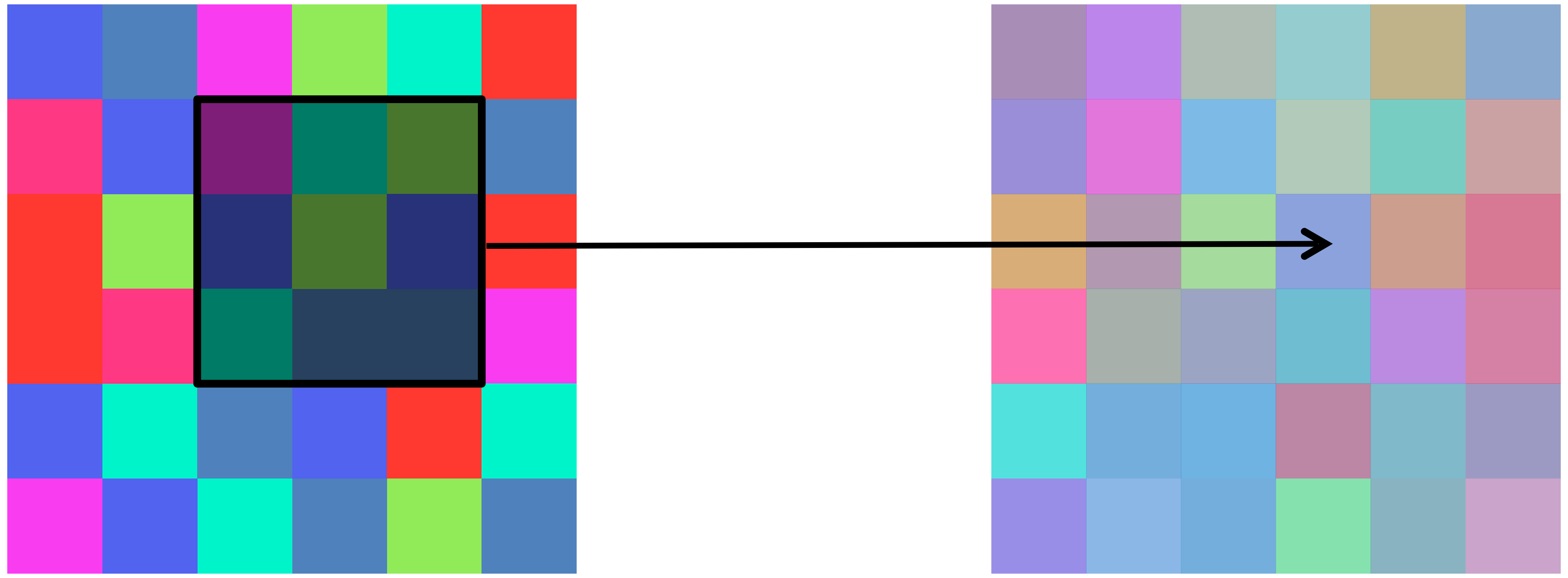
What types of **filtering** can we do?

Point Operation



point processing

Neighborhood Operation



"filtering"

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Neighborhood Operators (Filtering)

Original Image

blur

sharpen

edge filter

Non-Linear Neighborhood Operators (Filtering)

Original Image

edge preserving smoothing

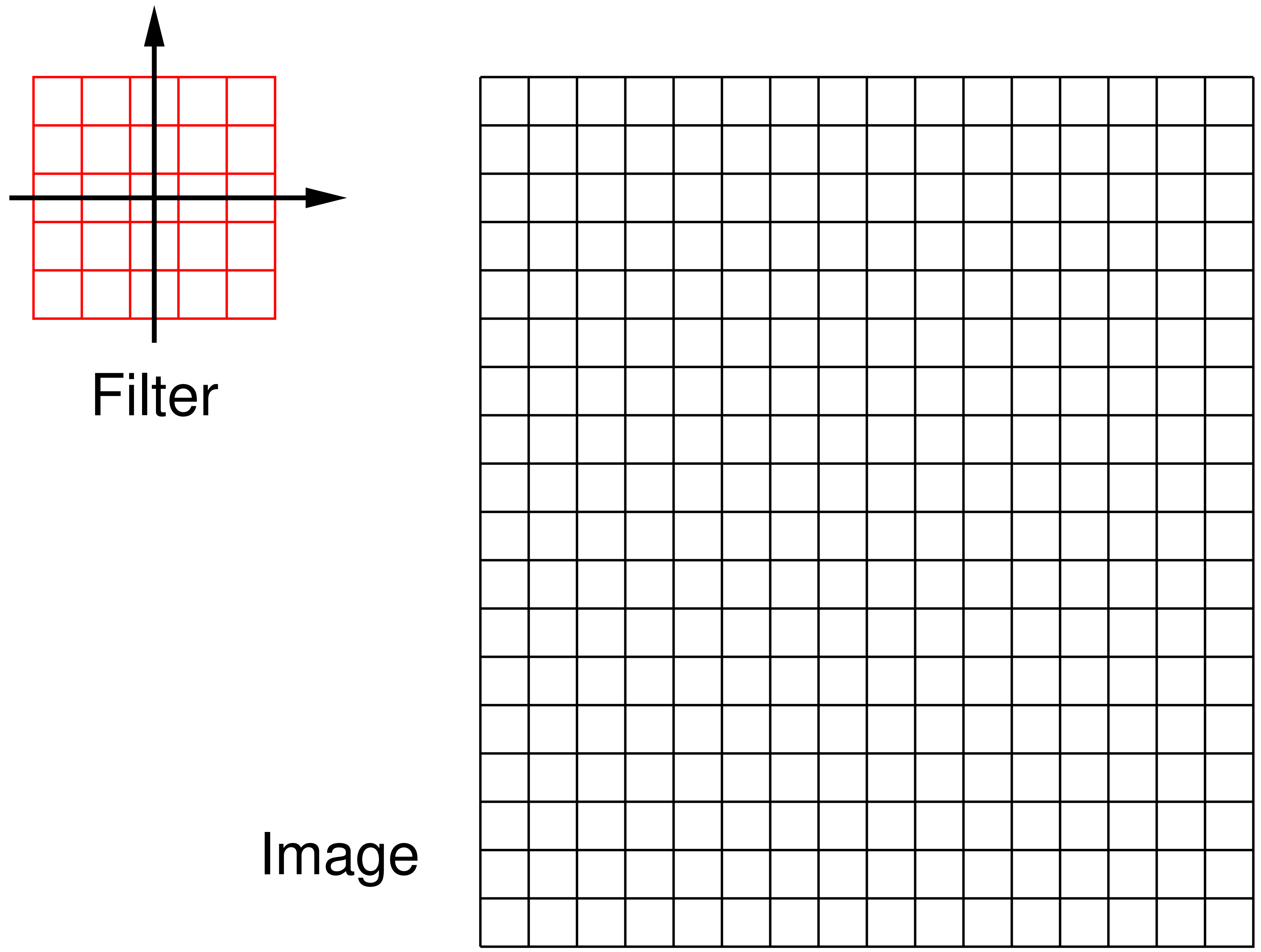
median

cenny edges

Linear **Filters**

Let $I(X,Y)$ be an $n \times n$ digital image (for convenience we let width = height)

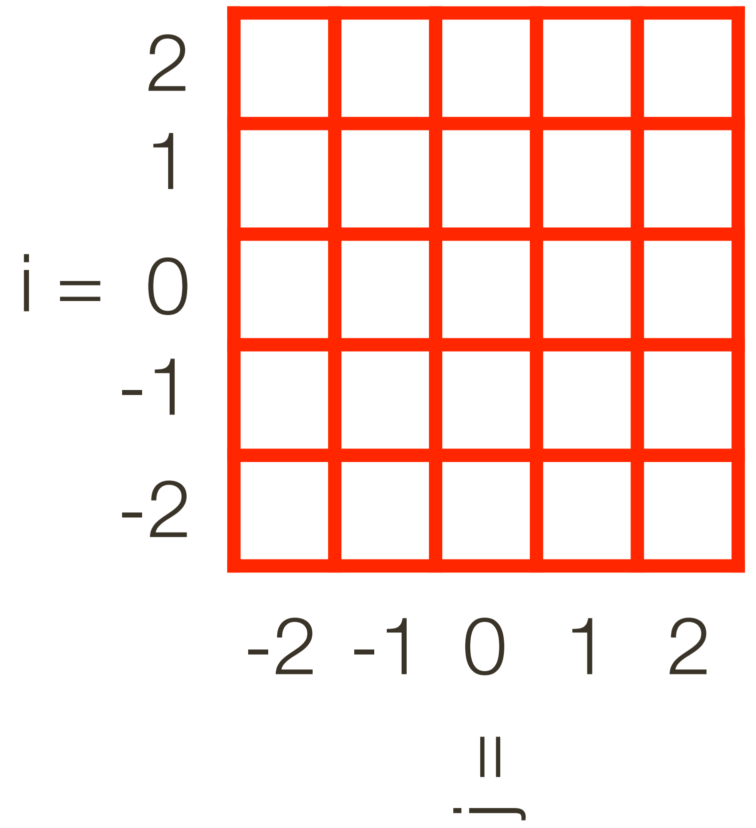
Let $F(X,Y)$ be another $m \times m$ digital image (our "**filter**" or "**kernel**")



For convenience we will assume m is odd. (Here, $m = 5$)

Linear **Filters**

Let $k = \left\lfloor \frac{m}{2} \right\rfloor$



Compute a new image, $I'(X, Y)$, as follows

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — $I'(X,Y)$

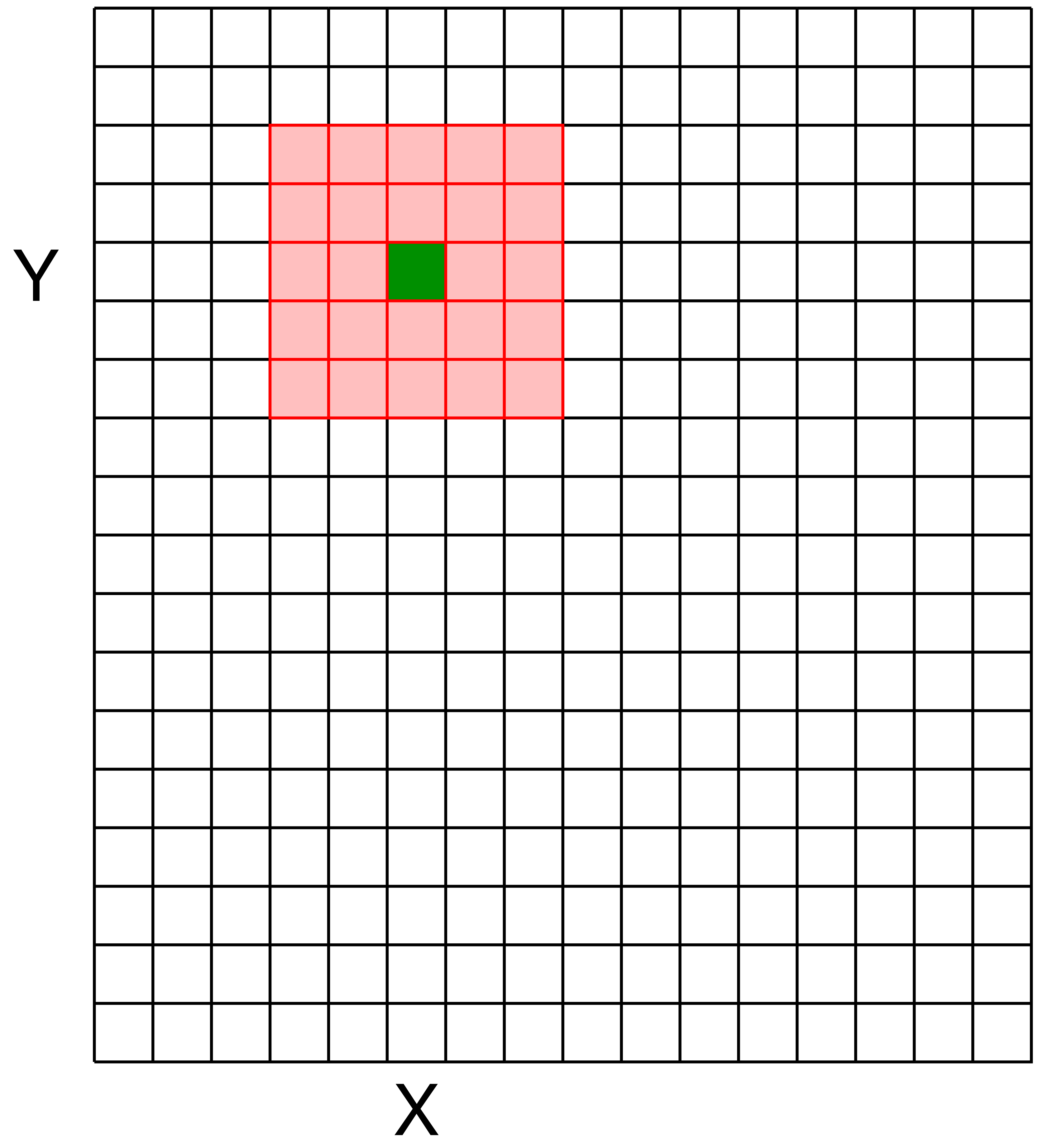
filter — $F(i,j)$

image (signal) — $I(X+i, Y+j)$

Intuition: each pixel in the output image is a linear combination of the same index pixel and its neighboring pixels in the original image

Linear **Filters**

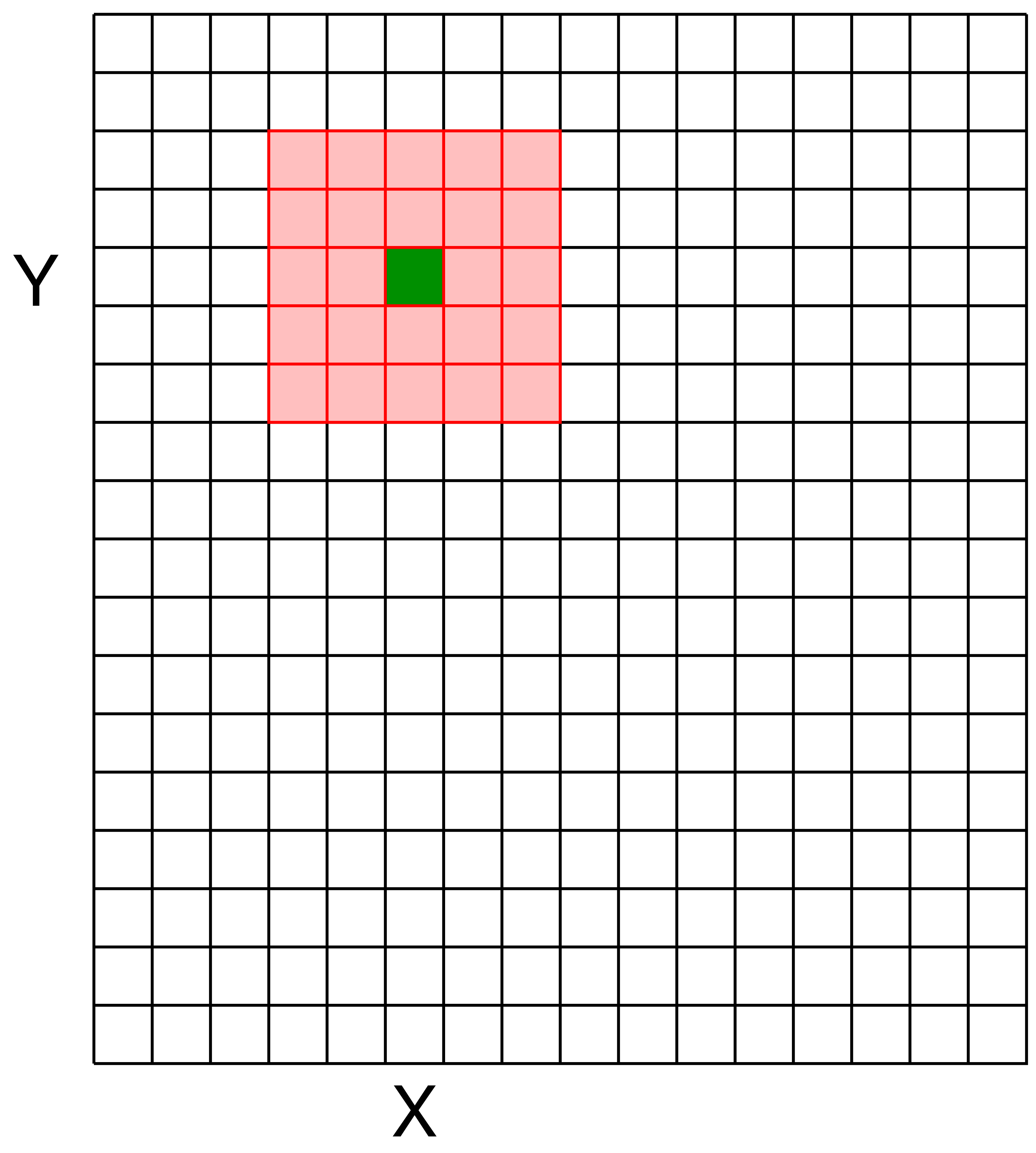
For a give X and Y, superimpose the filter on the image centered at (X, Y)



Linear **Filters**

For a give X and Y, superimpose the filter on the image centered at (X, Y)

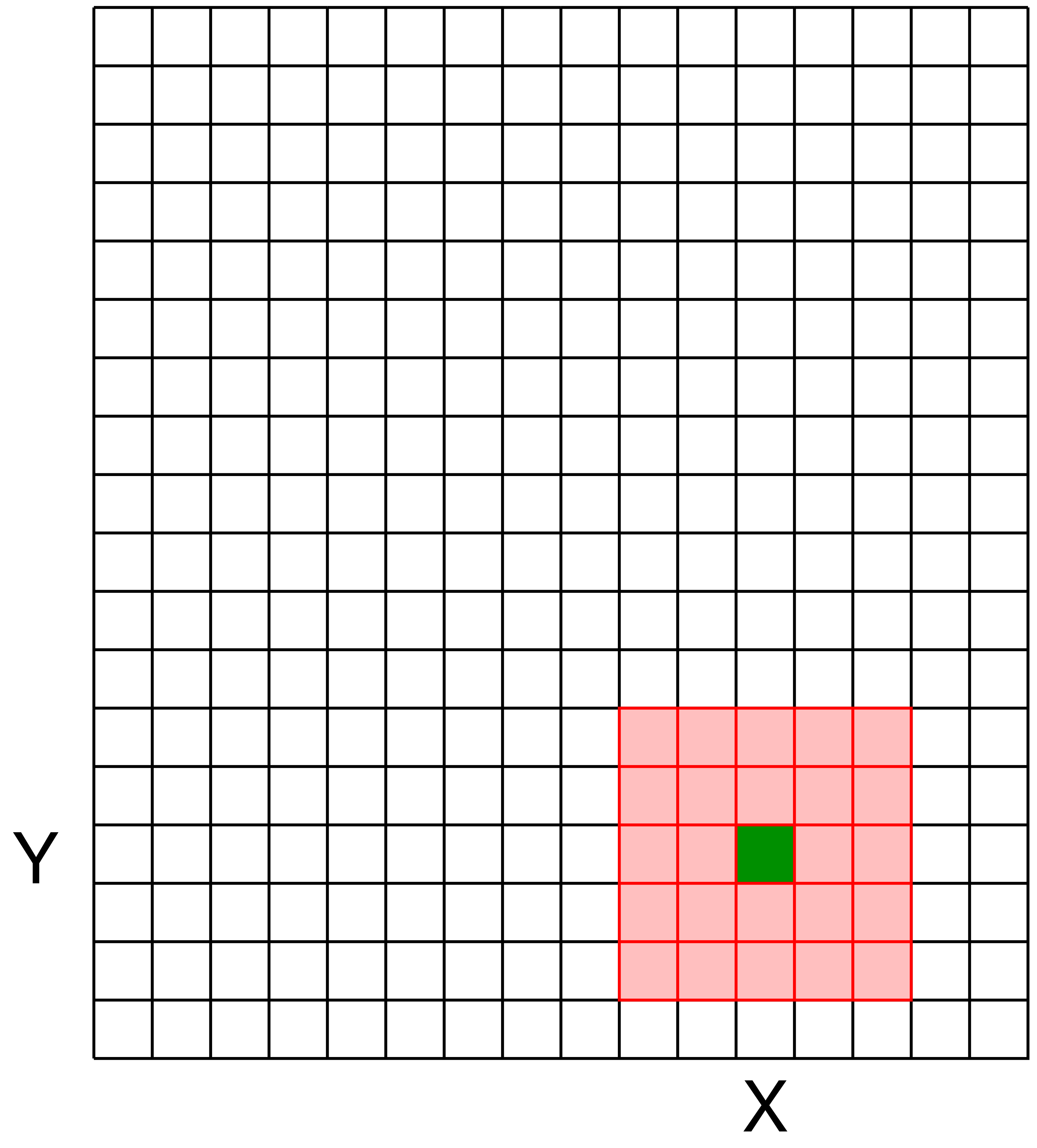
Compute the new pixel value, $I'(X, Y)$, as the sum of $m \times m$ values, where each value is the product of the original pixel value in $I(X, Y)$ and the corresponding values in the filter



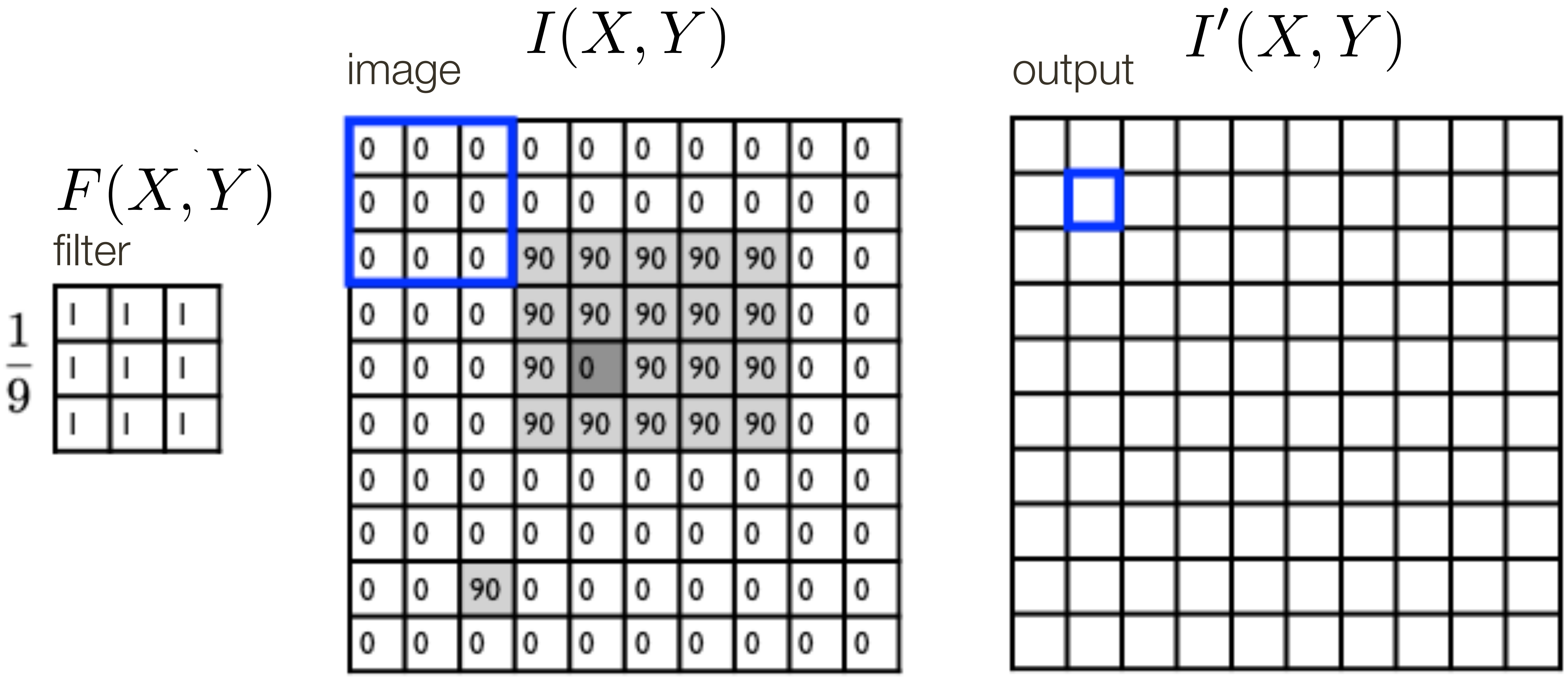
Linear **Filters**

The computation is repeated for each

$$(X, Y)$$



Linear Filter **Example**



$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output | filter | image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

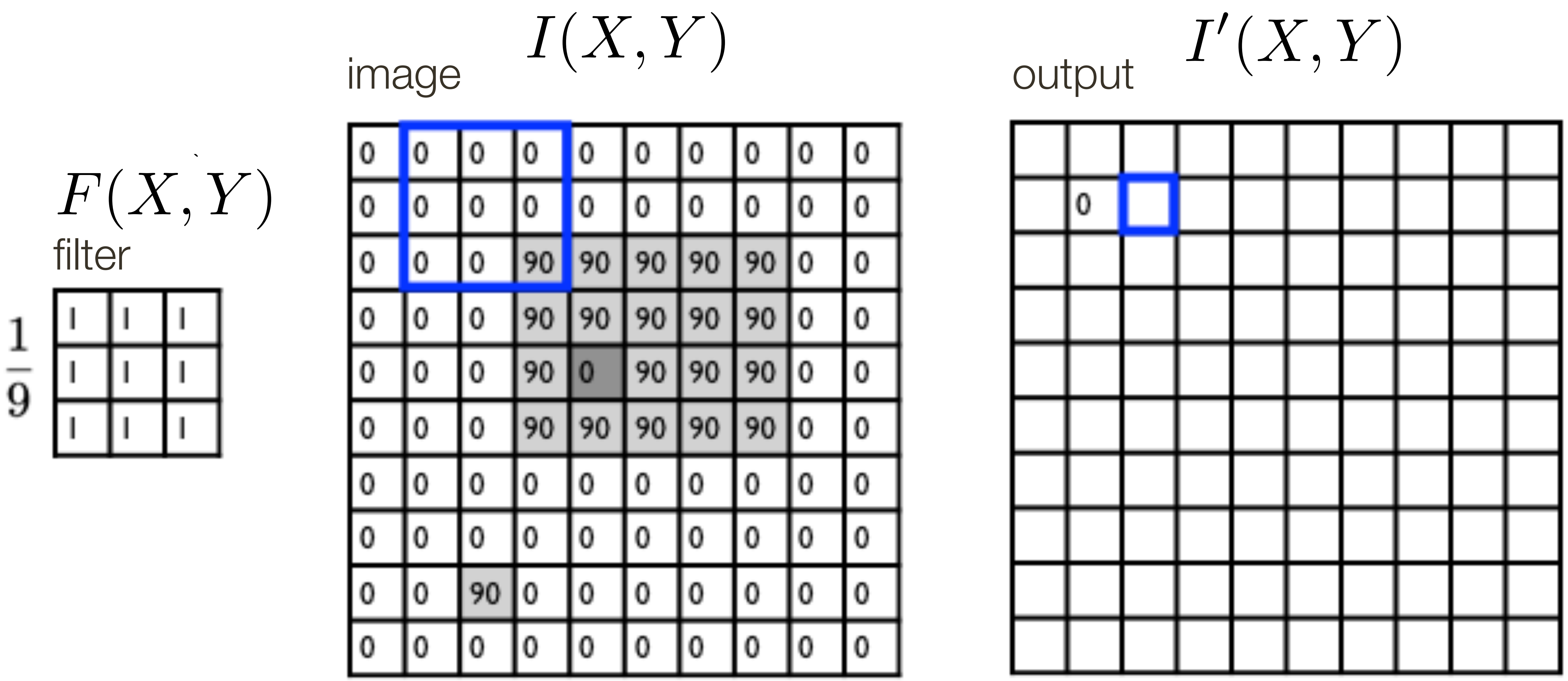
	0								

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

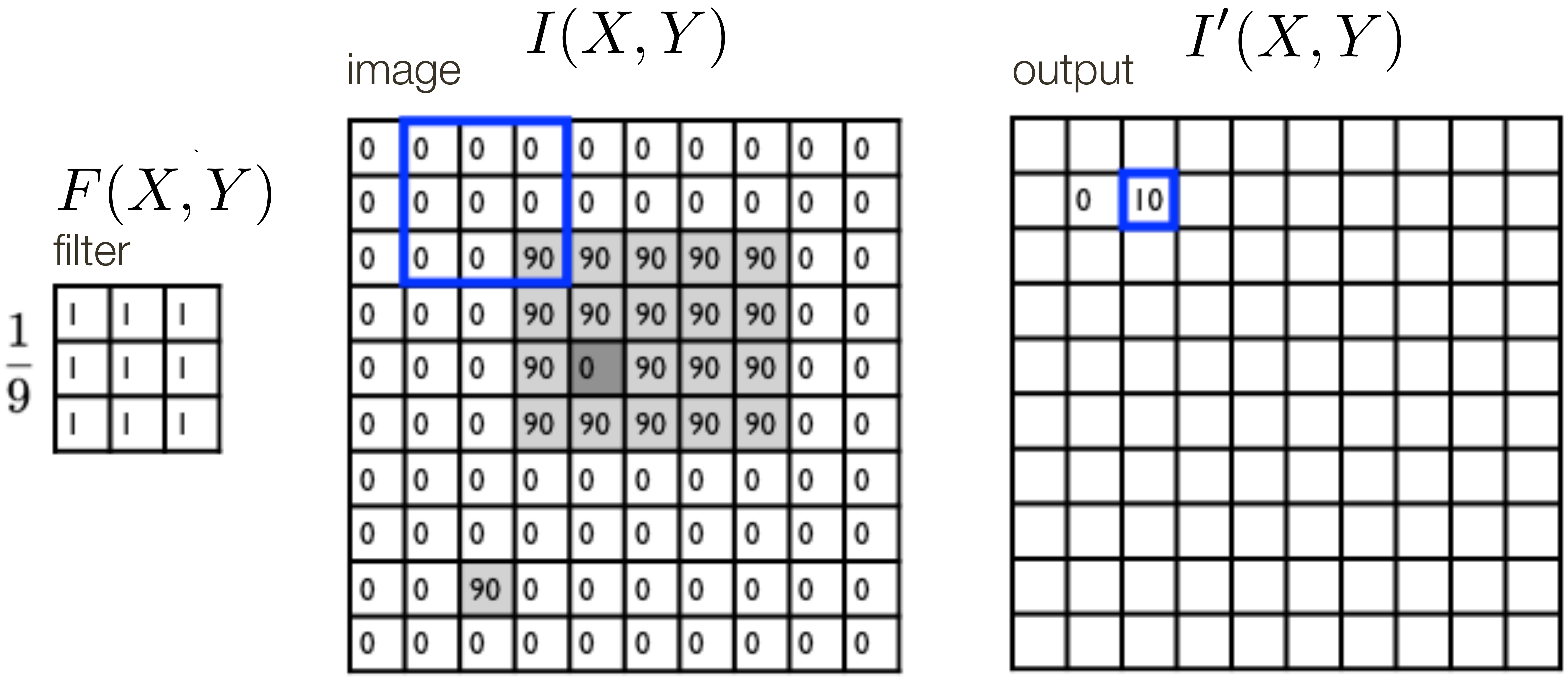


$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**



$$I'(X,Y) = \sum_{j=-k}^{k}\sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output | filter | image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

filter $F(X,Y)$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10							

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

filter $F(X,Y)$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20						

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20						

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) \, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

filter $F(X,Y)$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30					

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30	30				

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) \, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30	30	30			

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30	30	30	20		

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(i,j)}_{\text{filter}} \underbrace{I(X+i, Y+j)}_{\text{image (signal)}}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

filter $F(X,Y)$

$\frac{1}{9}$ ×

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0								

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(i,j)}_{\text{filter}} \underbrace{I(X+i, Y+j)}_{\text{image (signal)}}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

filter $F(X,Y)$

$\frac{1}{9}$ ×

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20							

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

filter $F(X,Y)$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40						

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

filter $F(X,Y)$

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

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0								

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$ filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30							

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(i,j)}_{\text{filter}} \underbrace{I(X+i, Y+j)}_{\text{image (signal)}}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X, Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X, Y)$

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	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	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

output $I'(X, Y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10								

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) \, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

filter $F(X,Y)$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(i,j)}_{\text{filter}} \underbrace{I(X+i, Y+j)}_{\text{image (signal)}}$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

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	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	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

output $I'(X,Y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear **Filters**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(X+i, Y+j)$$

output — $I'(X,Y)$

filter — $F(i,j)$

image (signal) — $I(X+i, Y+j)$

For a give X and Y, superimpose the filter on the image centered at (X,Y)

Compute the new pixel value, $I'(X,Y)$, as the sum of $m \times m$ values, where each value is the product of the original pixel value in $I(X,Y)$ and the corresponding values in the filter

Linear **Filters**

Let's do some accounting …

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) \, I(X+i, Y+j)$$

output

filter

image (signal)

Linear **Filters**

Let's do some accounting …

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

output | filter | image (signal)

At each pixel, (X,Y), there are $m \times m$ multiplications

Linear **Filters**

Let's do some accounting …

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(i,j)}_{\text{filter}} \underbrace{I(X+i, Y+j)}_{\text{image (signal)}}$$

At each pixel, (X,Y), there are $m \times m$ multiplications

There are $n \times n$ pixels in (X,Y)

Linear **Filters**

Let's do some accounting …

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k} \sum_{i=-k}^{k} \underbrace{F(i,j)}_{\text{filter}} \underbrace{I(X+i, Y+j)}_{\text{image (signal)}}$$

At each pixel, (X,Y), there are $m \times m$ multiplications

There are $n \times n$ pixels in (X,Y)

Total: $m^2 \times n^2$ multiplications

Linear **Filters**

Let's do some accounting …

$$\underbrace{I'(X,Y)}_{\text{output}} = \sum_{j=-k}^{k}\sum_{i=-k}^{k} \underbrace{F(i,j)}_{\text{filter}}\,\underbrace{I(X+i,Y+j)}_{\text{image (signal)}}$$

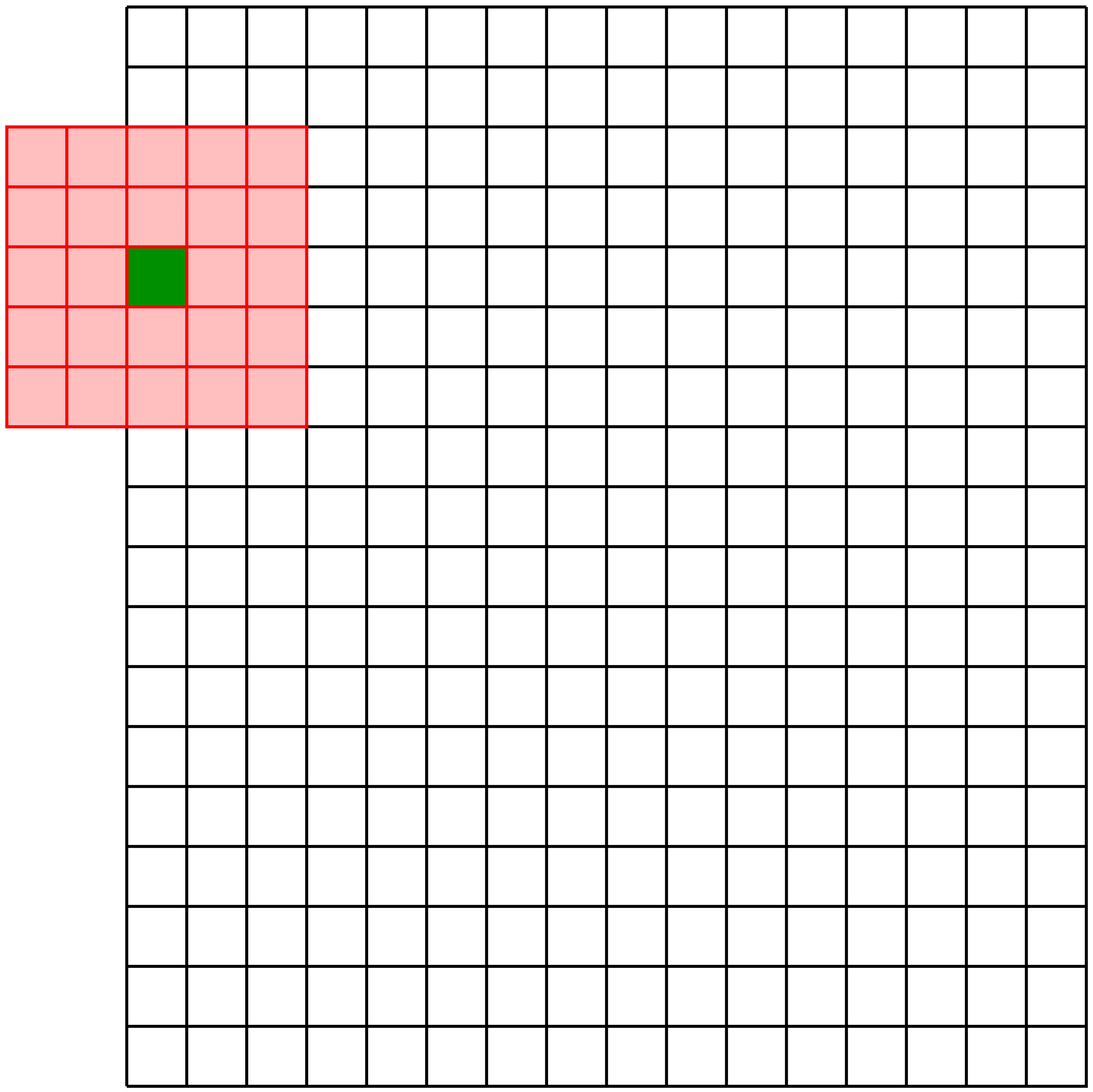
At each pixel, (X,Y), there are $m \times m$ multiplications

There are $n \times n$ pixels in (X,Y)

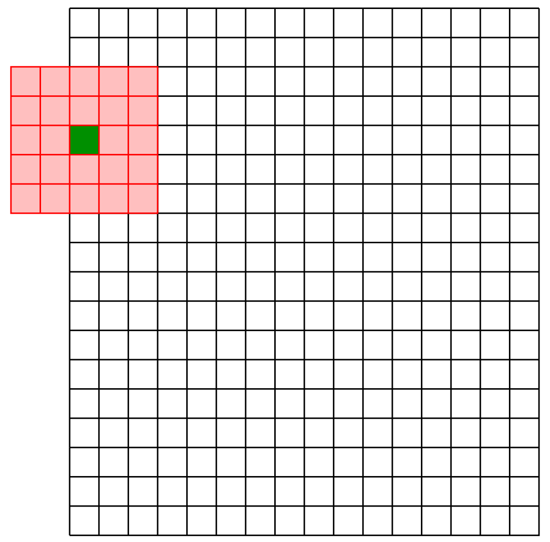
Total: $m^2 \times n^2$ multiplications

When m is fixed, small constant, this is $\mathcal{O}(n^2)$. But when $m \approx n$ this is $\mathcal{O}(m^4)$.

Linear Filters: **Boundary** Effects



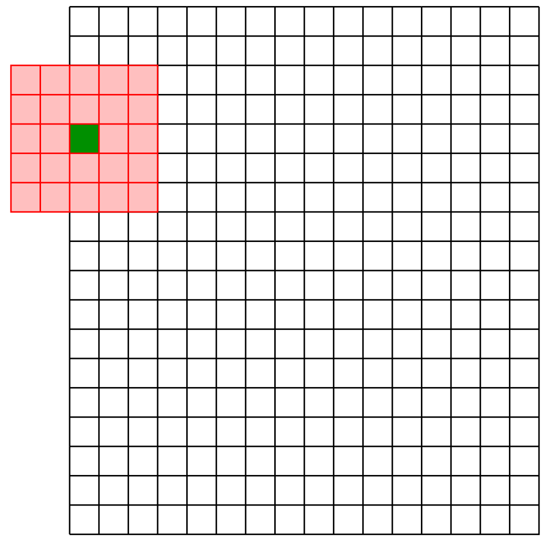
Linear Filters: **Boundary** Effects



Four standard ways to deal with boundaries:

1. **Ignore these locations:** Make the computation undefined for the top and bottom k rows and the leftmost and rightmost k columns

Linear Filters: **Boundary** Effects

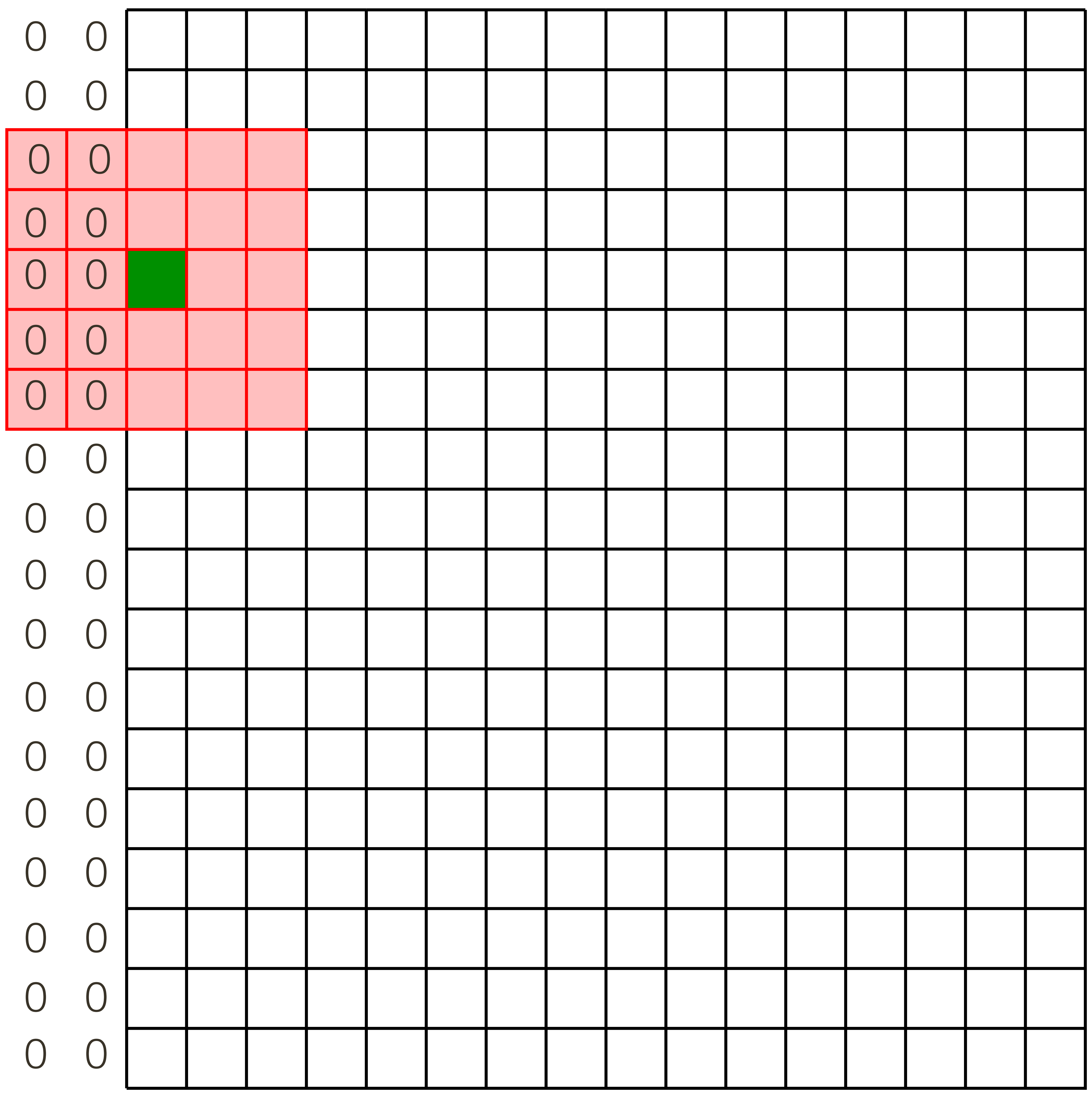


Four standard ways to deal with boundaries:

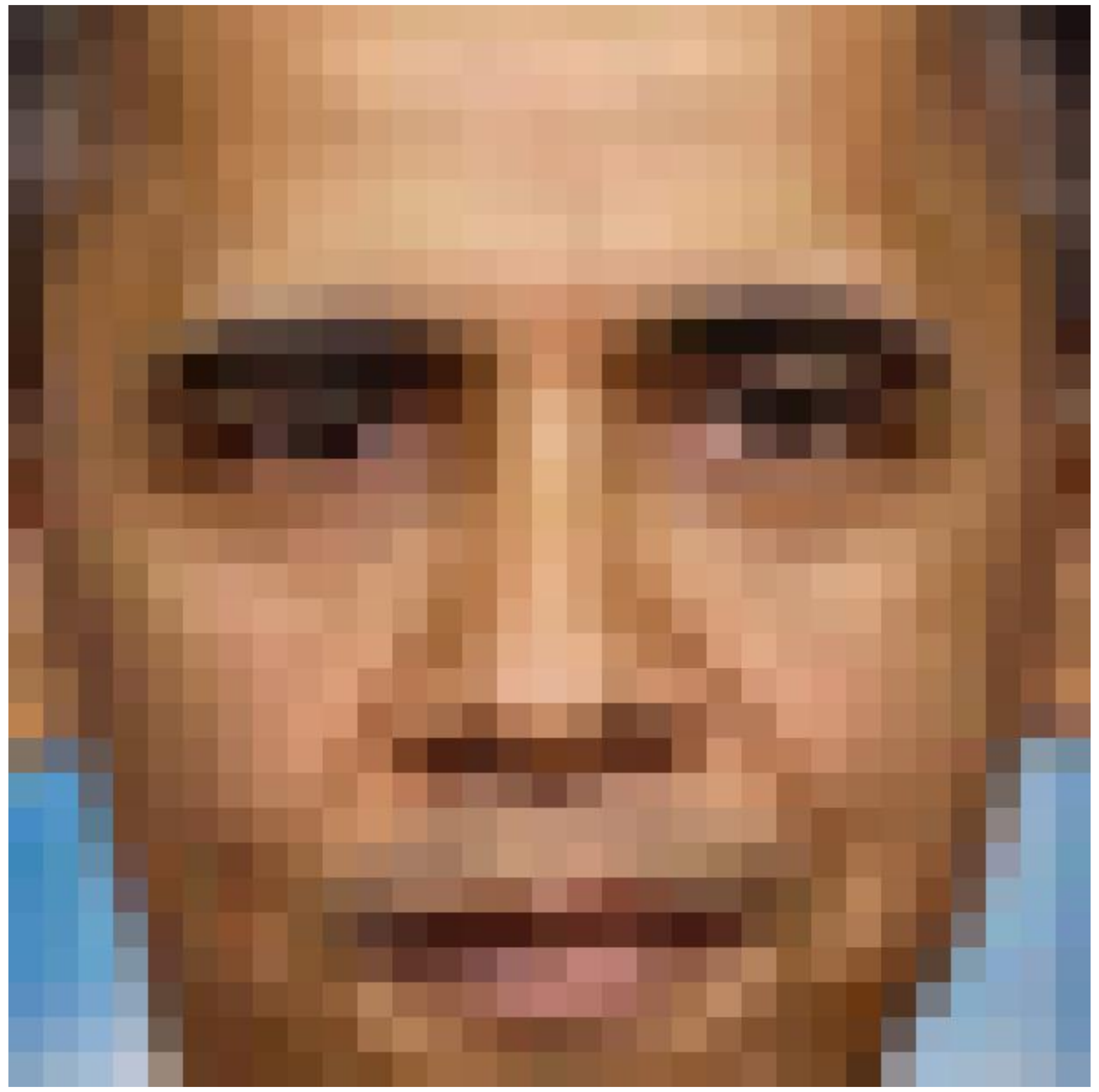
1. **Ignore these locations:** Make the computation undefined for the top and bottom k rows and the leftmost and rightmost k columns

2. **Pad the image with zeros**: Return zero whenever a value of I is required at some position outside the defined limits of X and Y

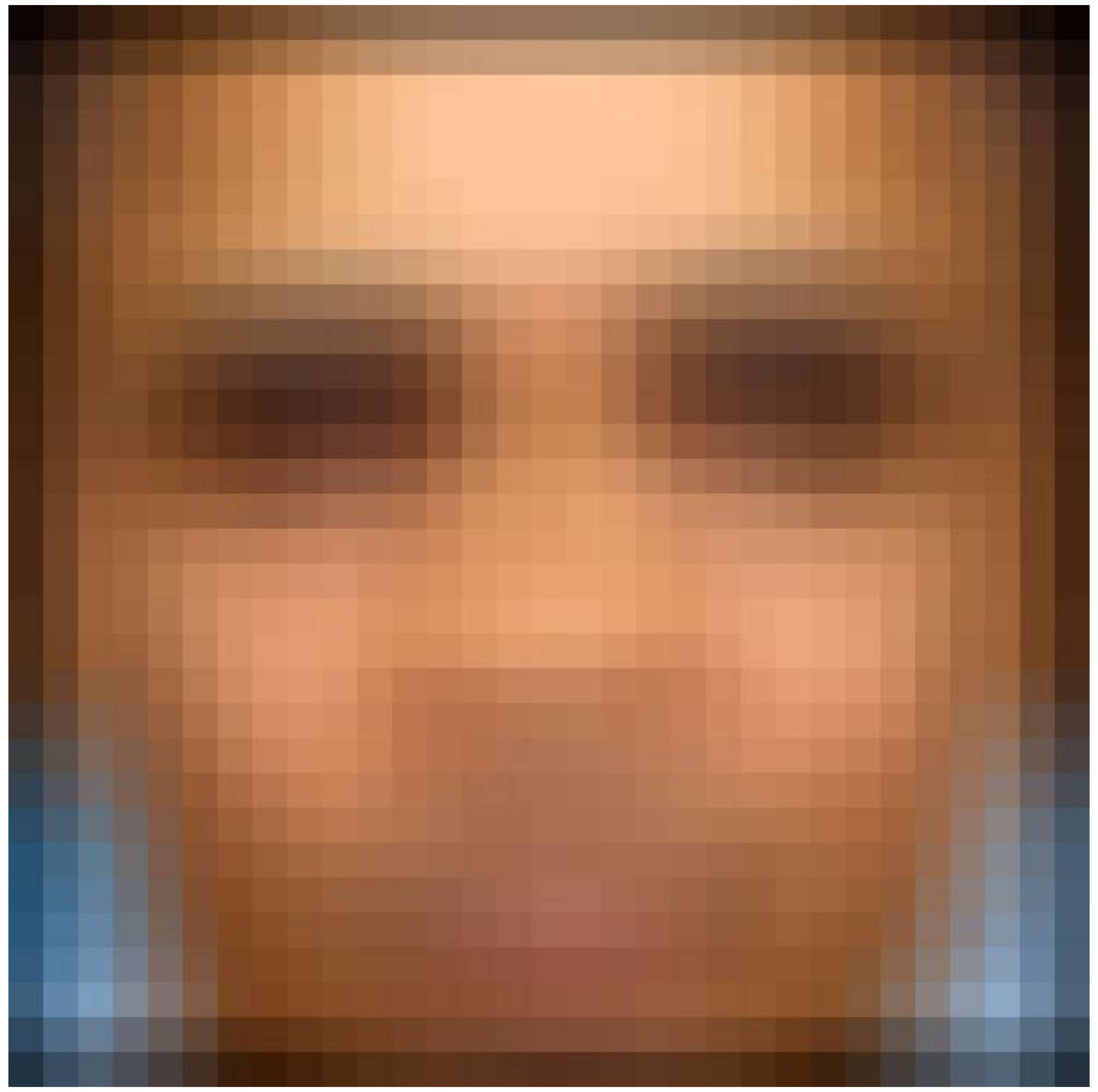
Linear Filters: **Boundary** Effects



Linear Filters: **Boundary** Effects

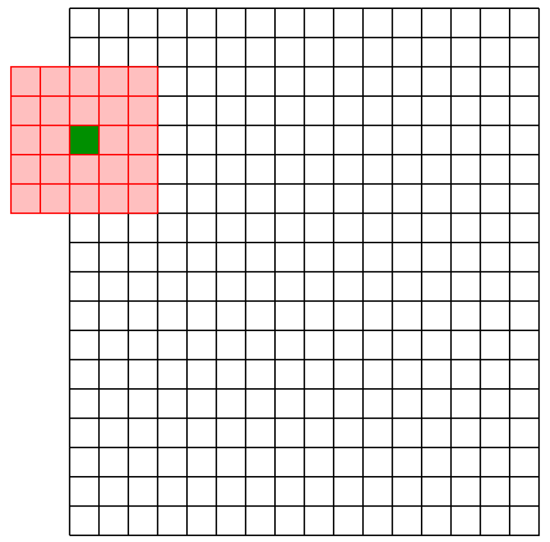


$$* \begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{matrix} =$$



Notice **decrease** in brightness at edges

Linear Filters: **Boundary** Effects



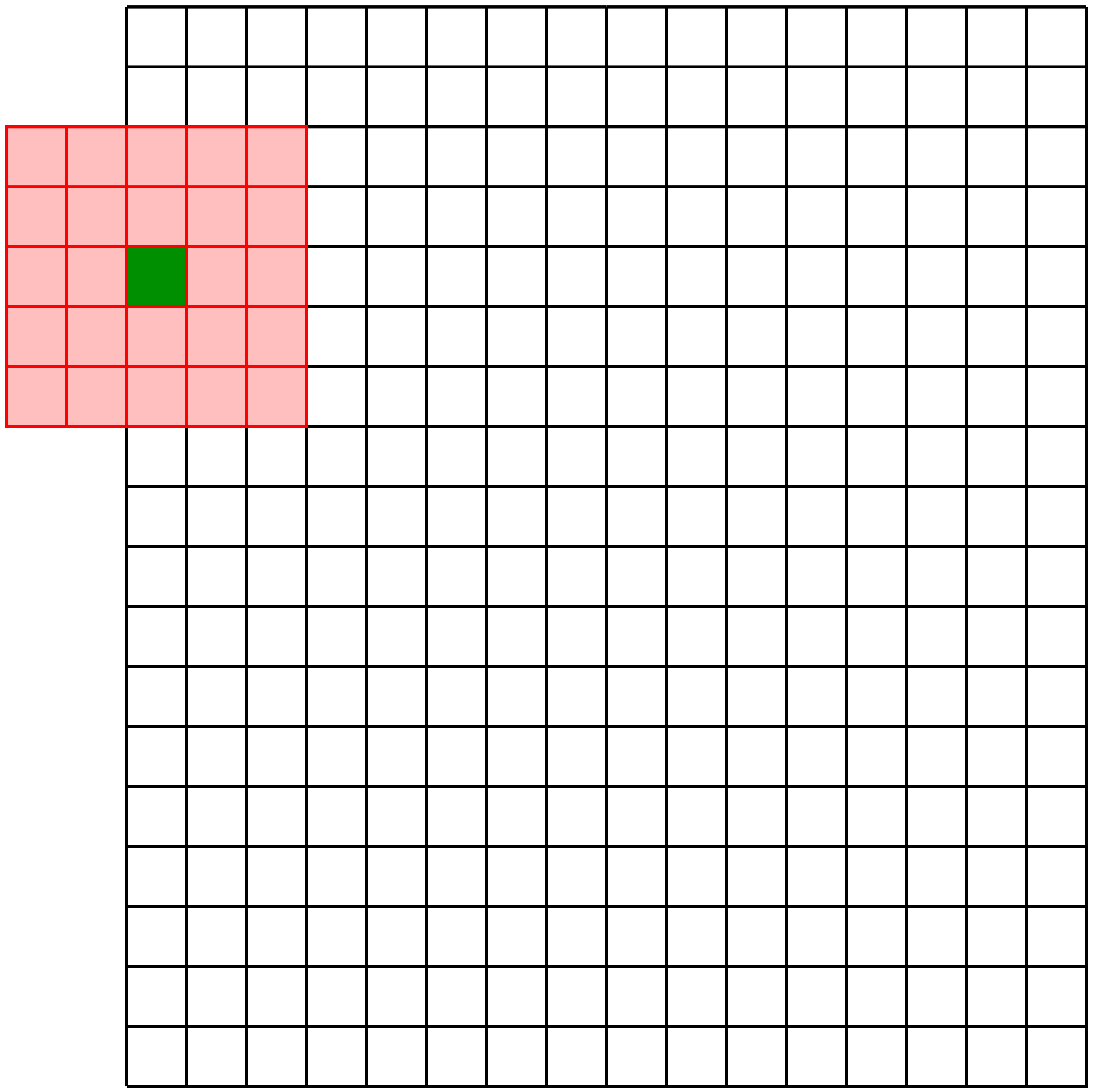
Four standard ways to deal with boundaries:

1. **Ignore these locations:** Make the computation undefined for the top and bottom k rows and the leftmost and rightmost k columns

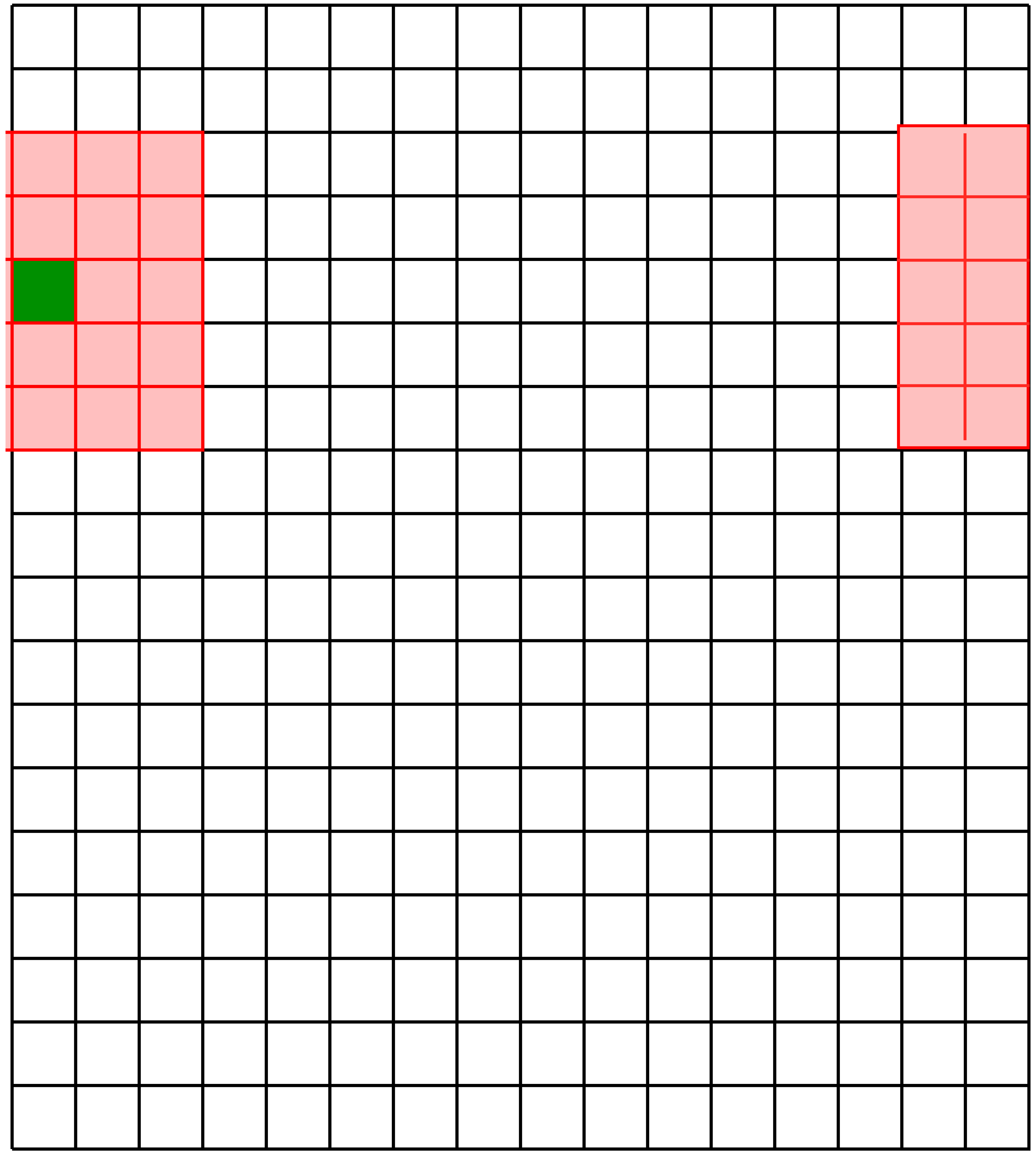
2. **Pad the image with zeros**: Return zero whenever a value of I is required at some position outside the defined limits of X and Y

3. **Assume periodicity**: The top row wraps around to the bottom row; the leftmost column wraps around to the rightmost column

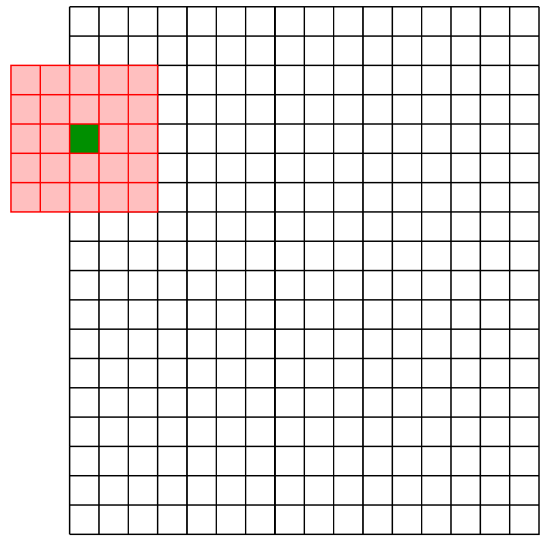
Linear Filters: **Boundary** Effects



Linear Filters: **Boundary** Effects



Linear Filters: **Boundary** Effects



Four standard ways to deal with boundaries:

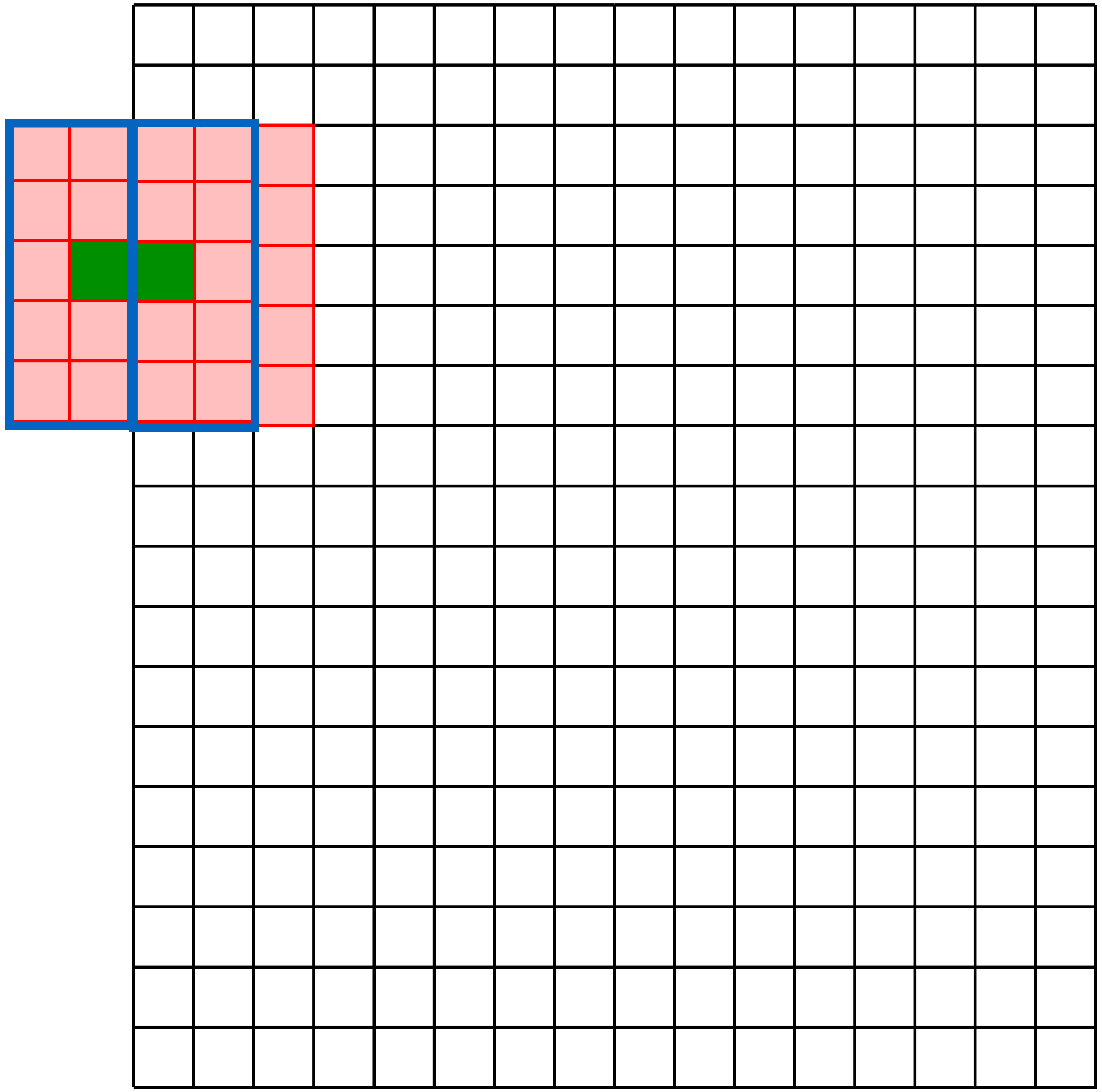
1. **Ignore these locations:** Make the computation undefined for the top and bottom k rows and the leftmost and rightmost k columns

2. **Pad the image with zeros**: Return zero whenever a value of I is required at some position outside the defined limits of X and Y

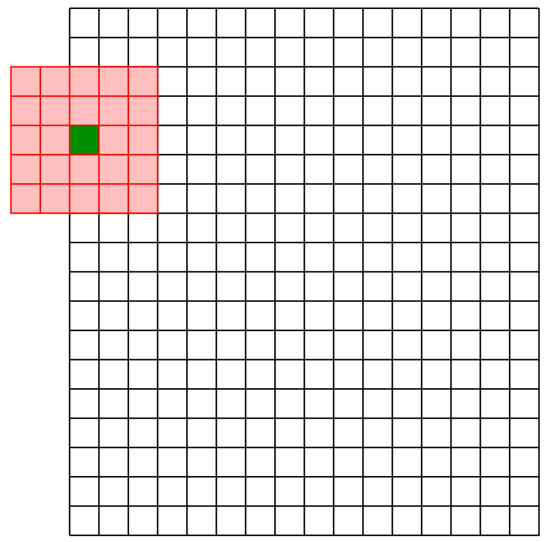
3. **Assume periodicity**: The top row wraps around to the bottom row; the leftmost column wraps around to the rightmost column

4. **Reflect boarder**: Copy rows/columns locally by reflecting over the edge

Linear Filters: **Boundary** Effects



Linear Filters: **Boundary** Effects



Four standard ways to deal with boundaries:

1. **Ignore these locations:** Make the computation undefined for the top and bottom k rows and the leftmost and rightmost k columns

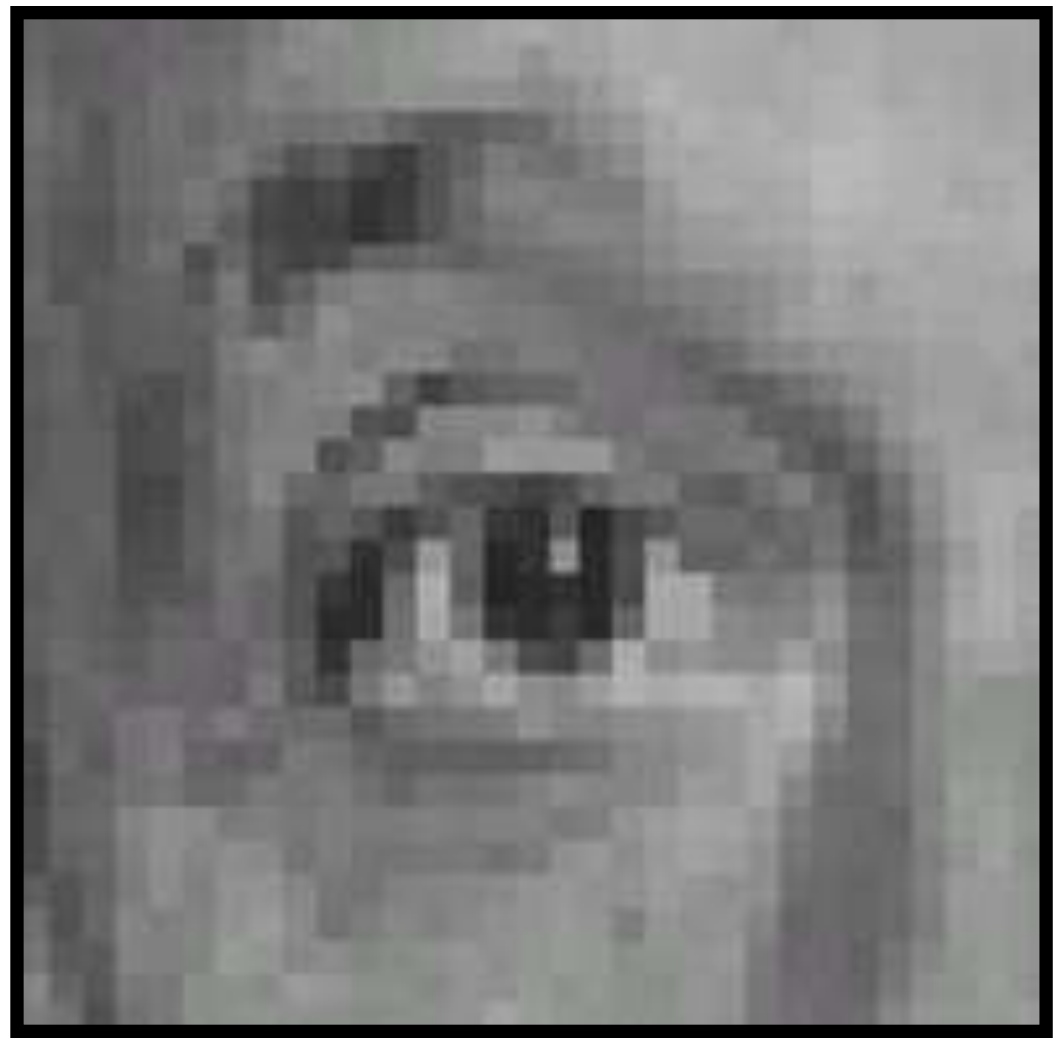
2. **Pad the image with zeros**: Return zero whenever a value of I is required at some position outside the defined limits of X and Y

3. **Assume periodicity**: The top row wraps around to the bottom row; the leftmost column wraps around to the rightmost column

4. **Reflect boarder**: Copy rows/columns locally by reflecting over the edge

A short exercise …

Example 1: Warm up



Original

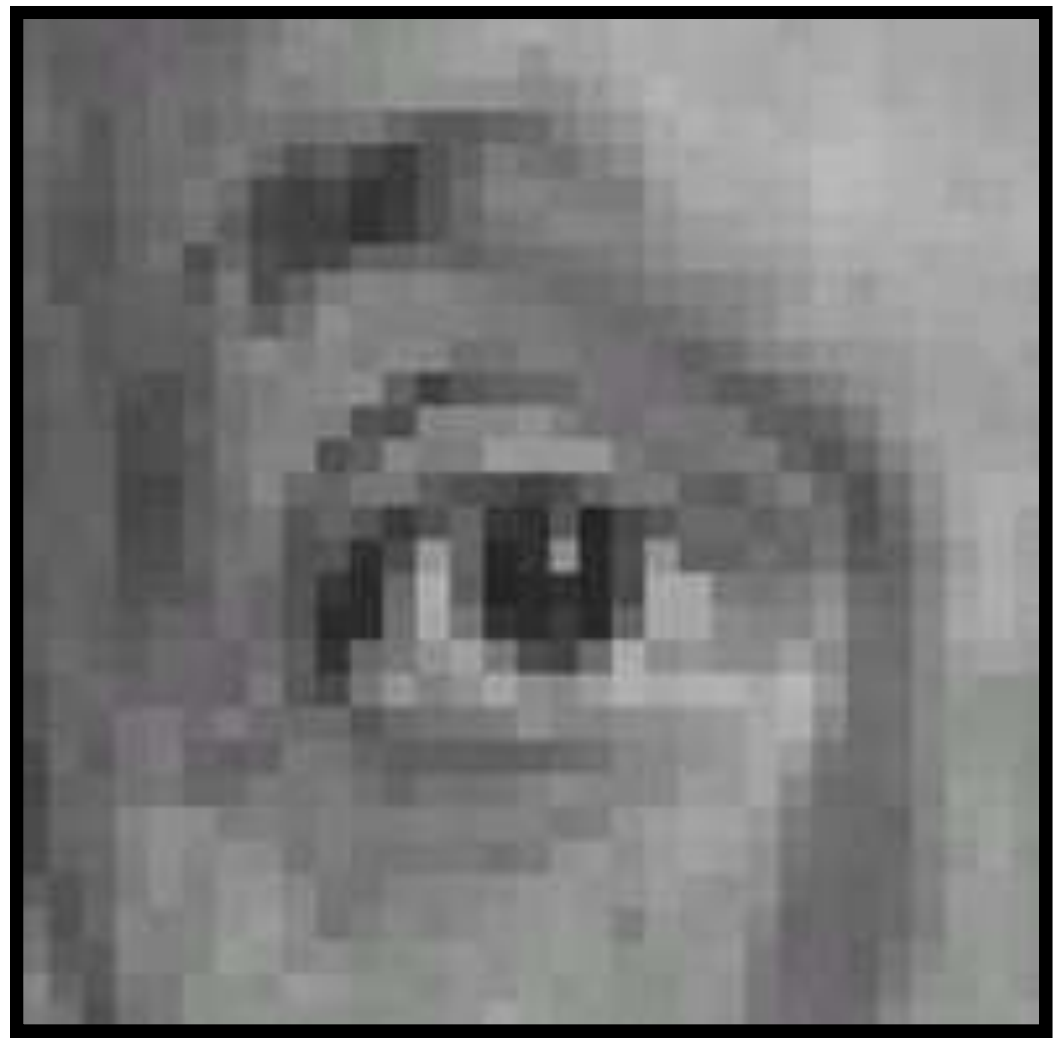
0	0	0
0	1	0
0	0	0

Filter

?

Result

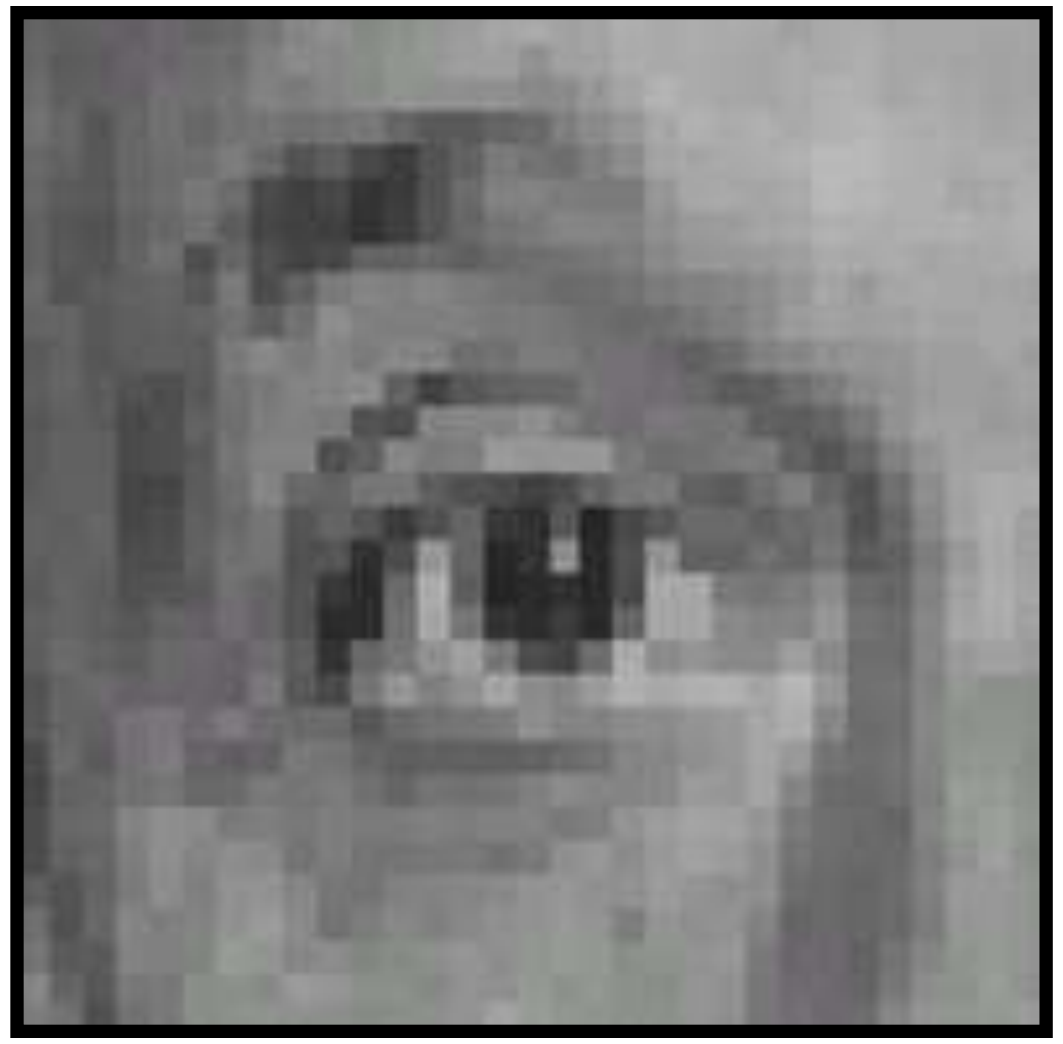
Example 1: Warm up



Original

0	0	0
0	1	0
0	0	0

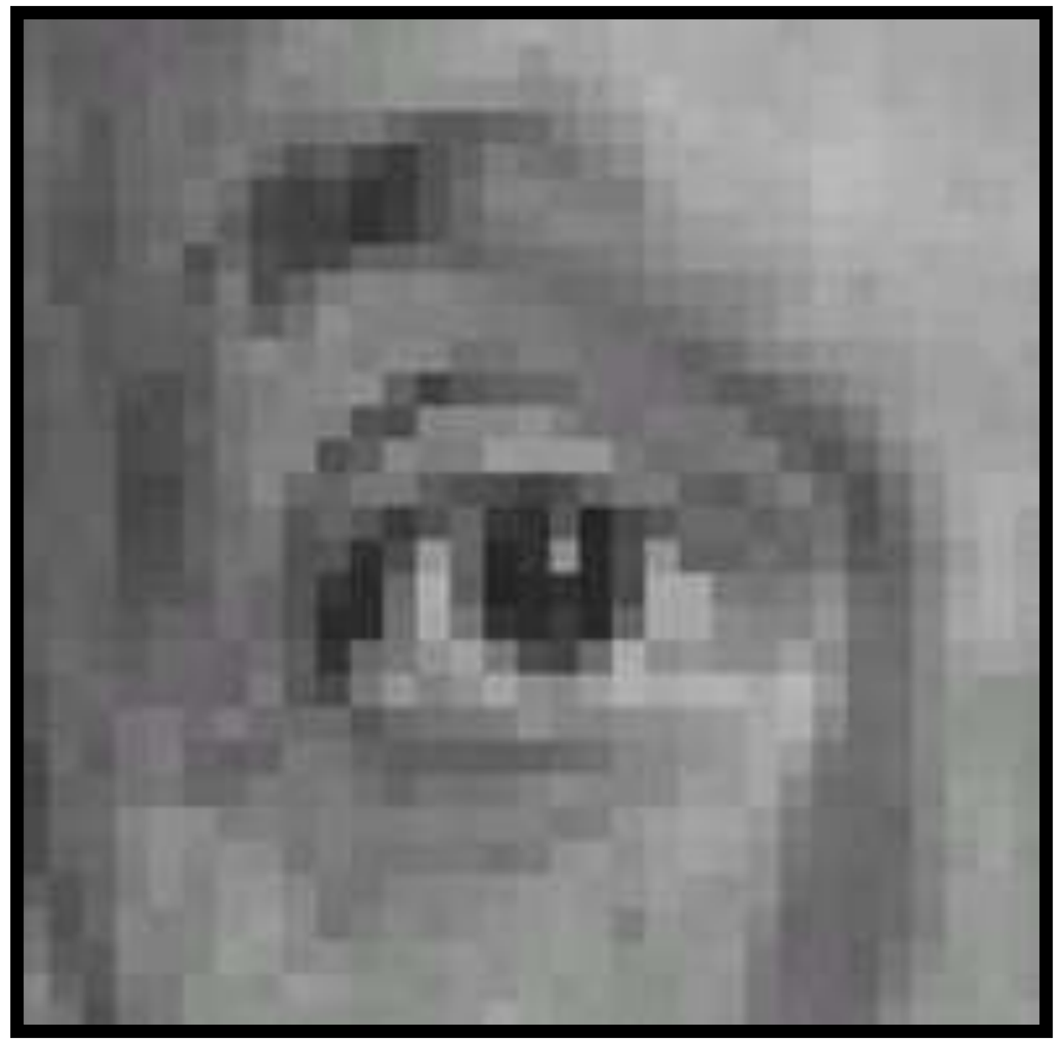
Filter



Result

(no change)

Example 2:



Original

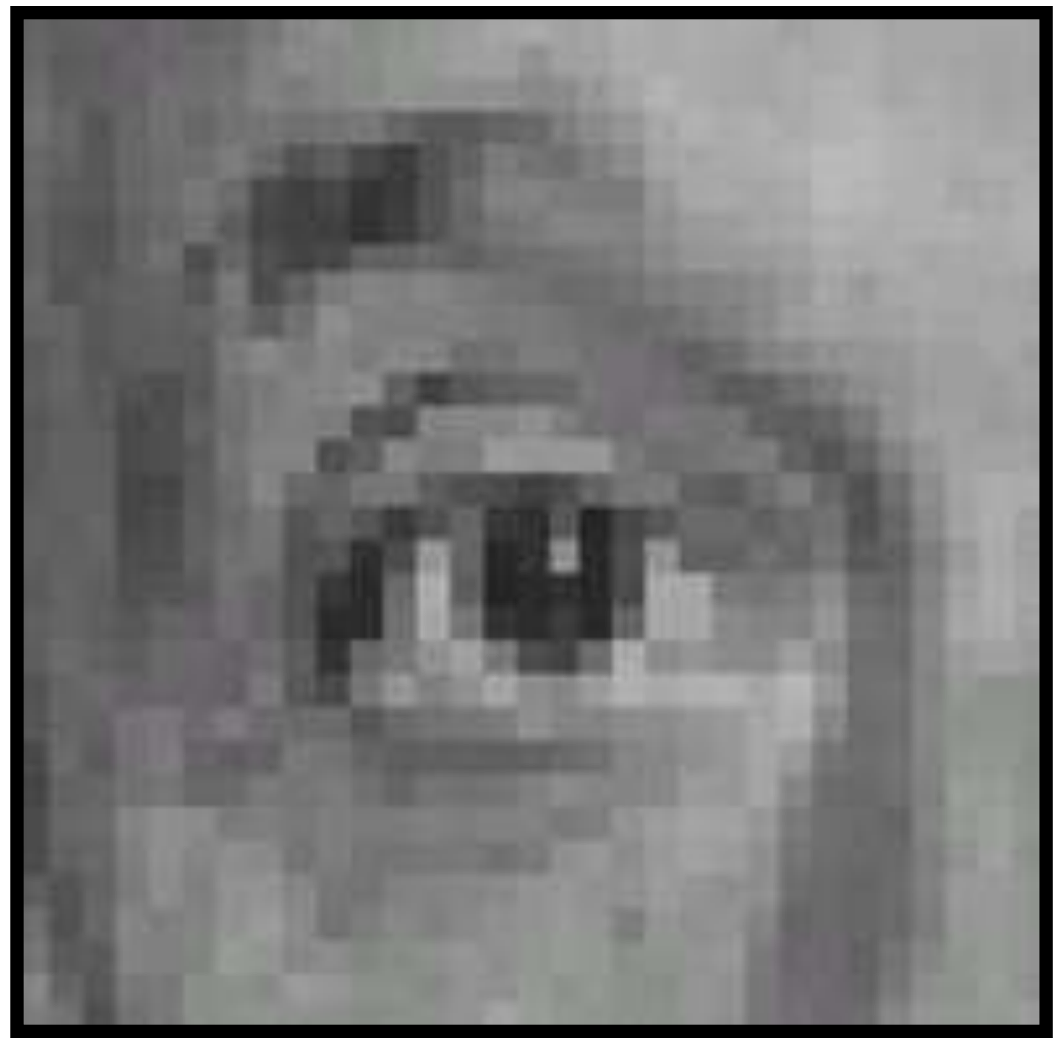
0	0	0
0	0	1
0	0	0

Filter

?

Result

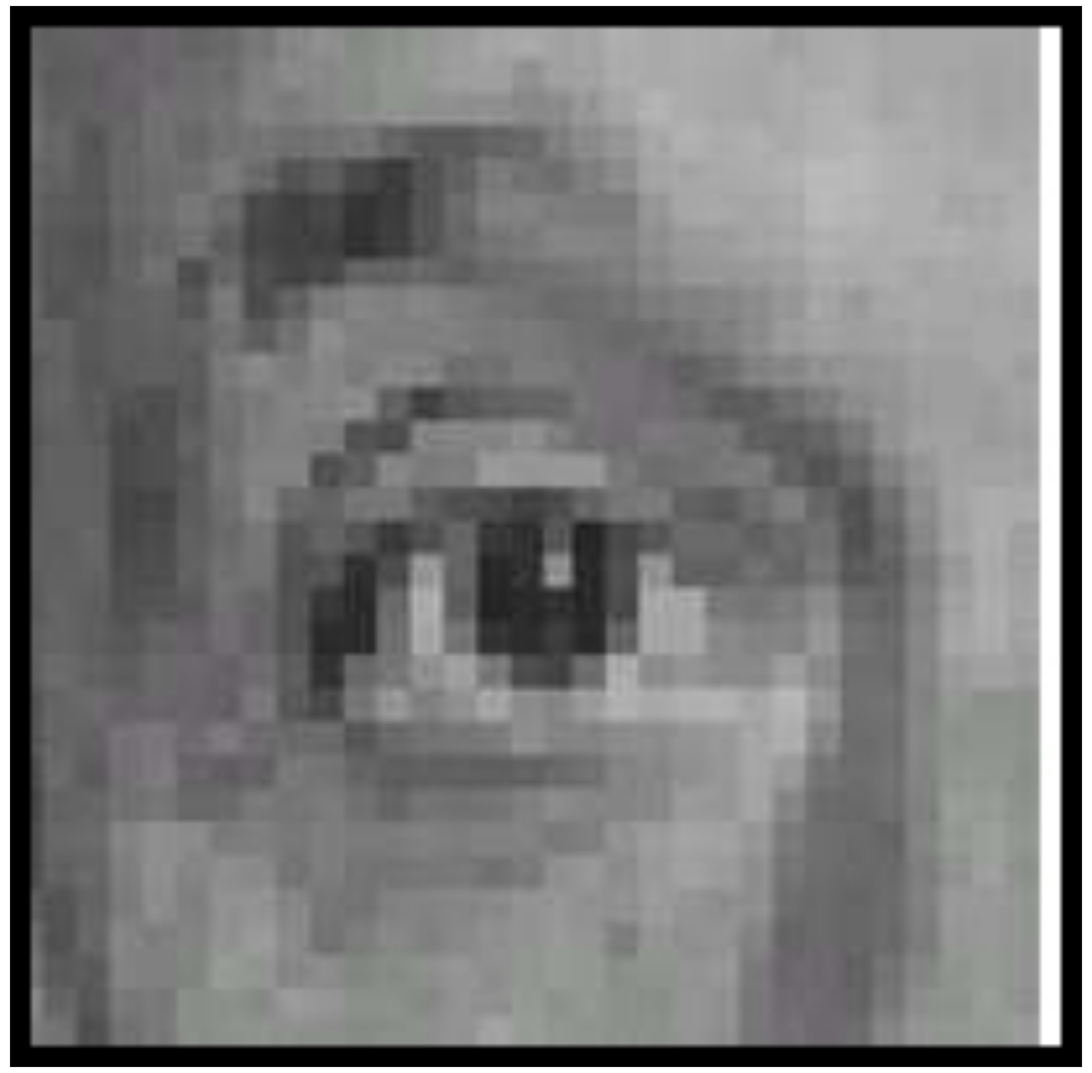
Example 2:



Original

0	0	0
0	0	1
0	0	0

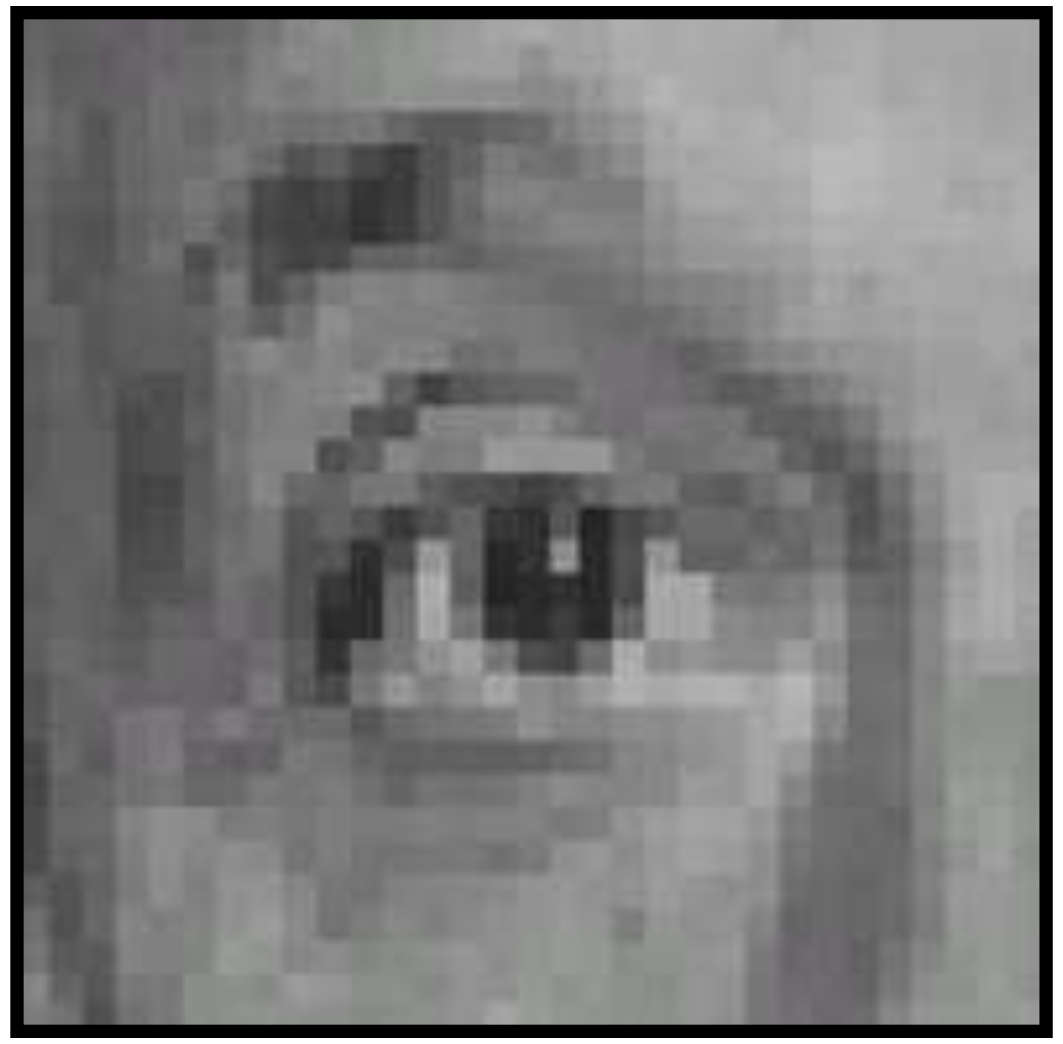
Filter



Result

(sift left by 1 pixel)

Example 3:



Original

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

1	1	1
1	1	1
1	1	1

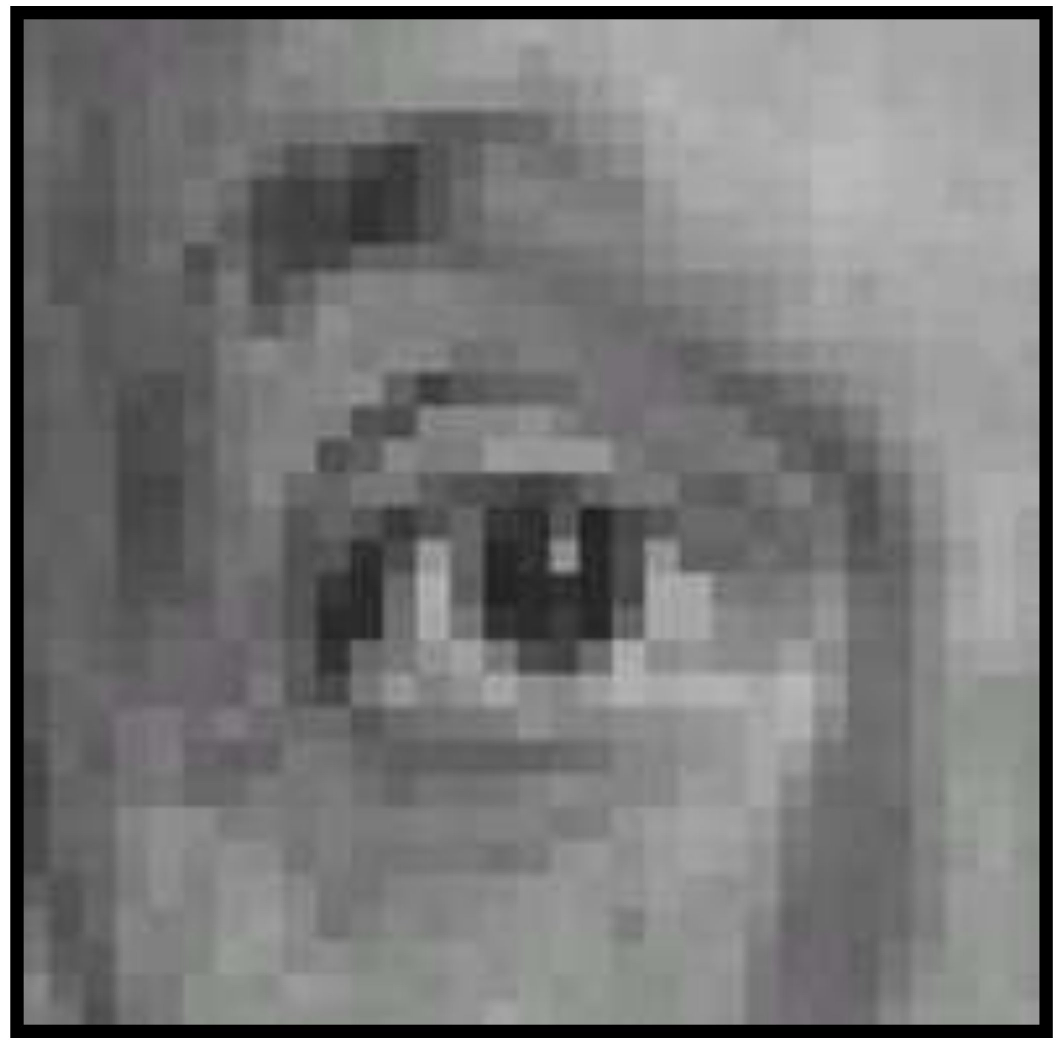
Filter

(filter sums to 1)

?

Result

Example 3:

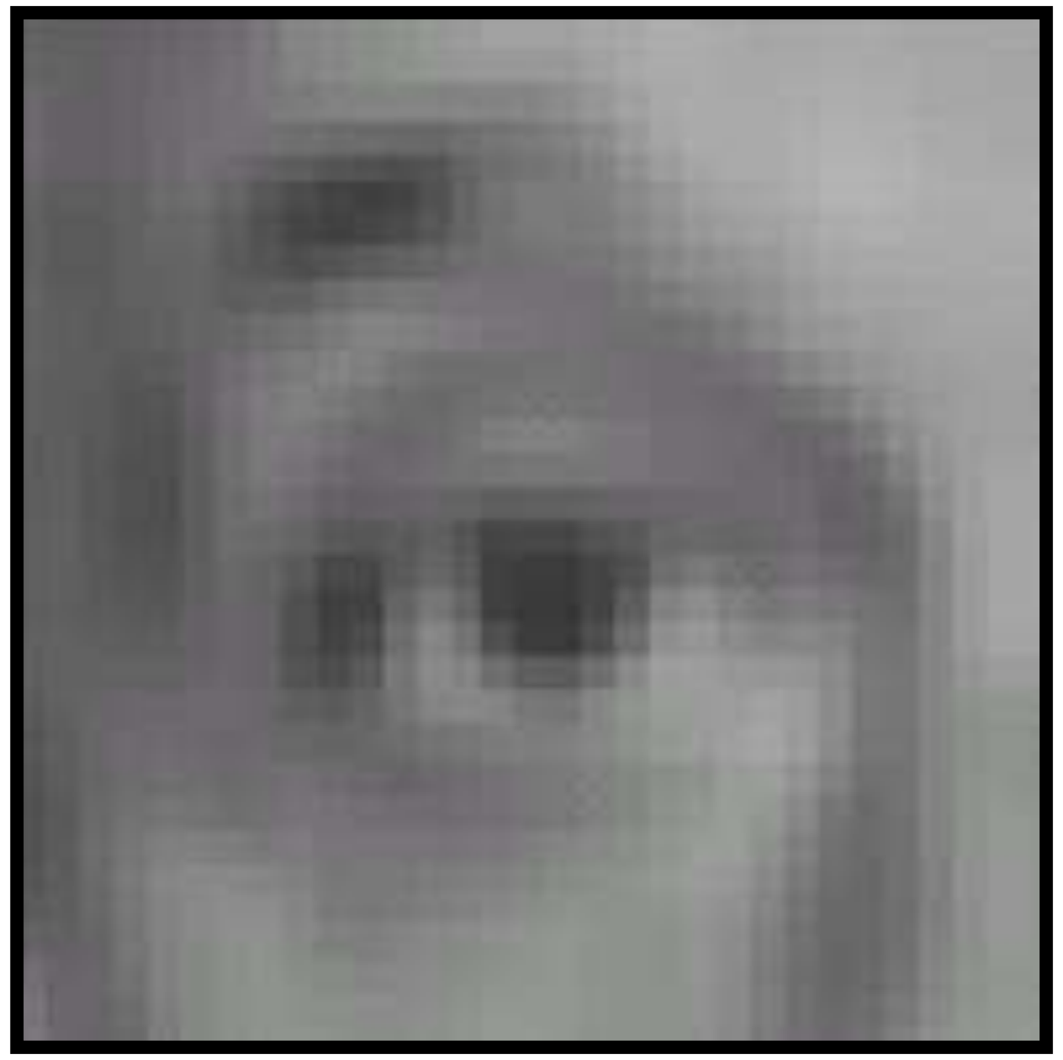


Original

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

Filter

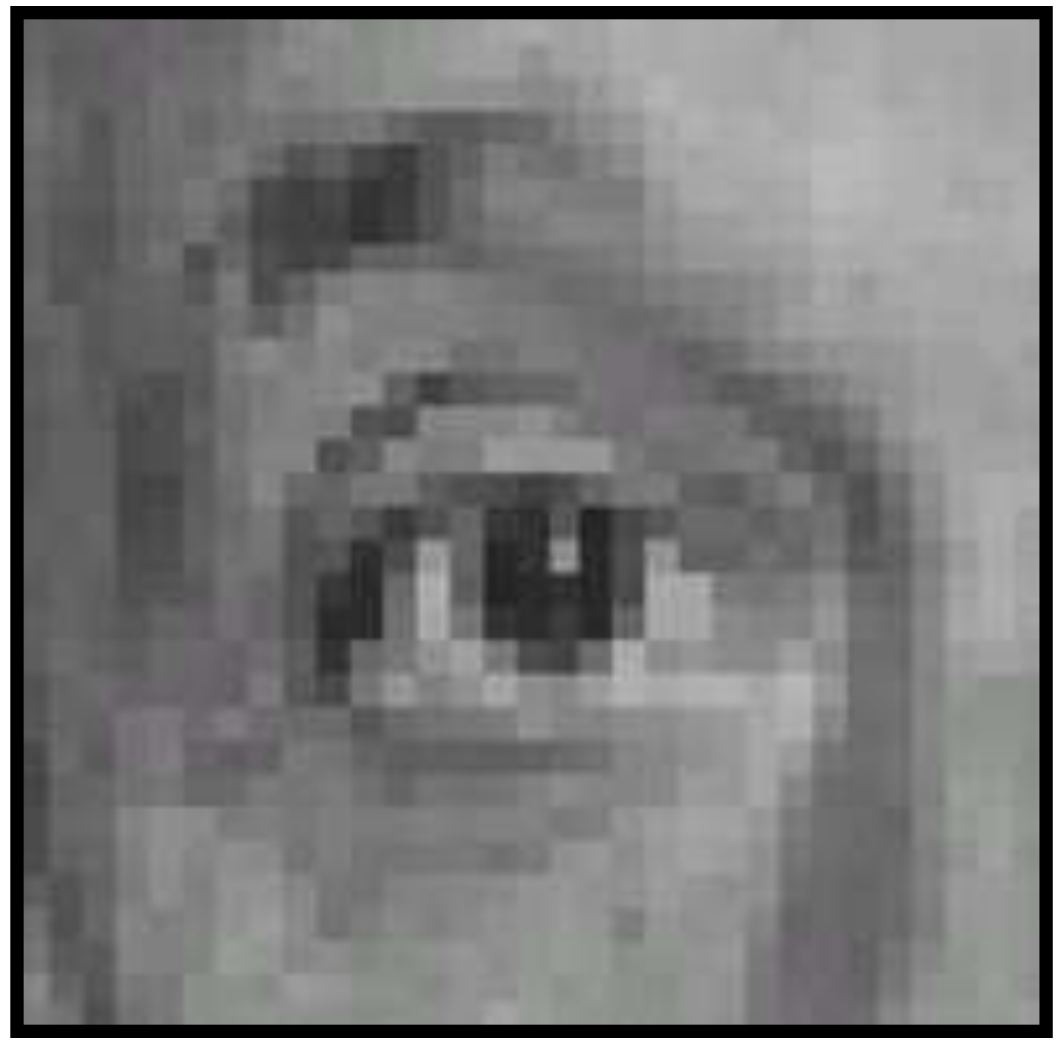
(filter sums to 1)



Result

(blur with a box filter)

Example 4:



Original

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

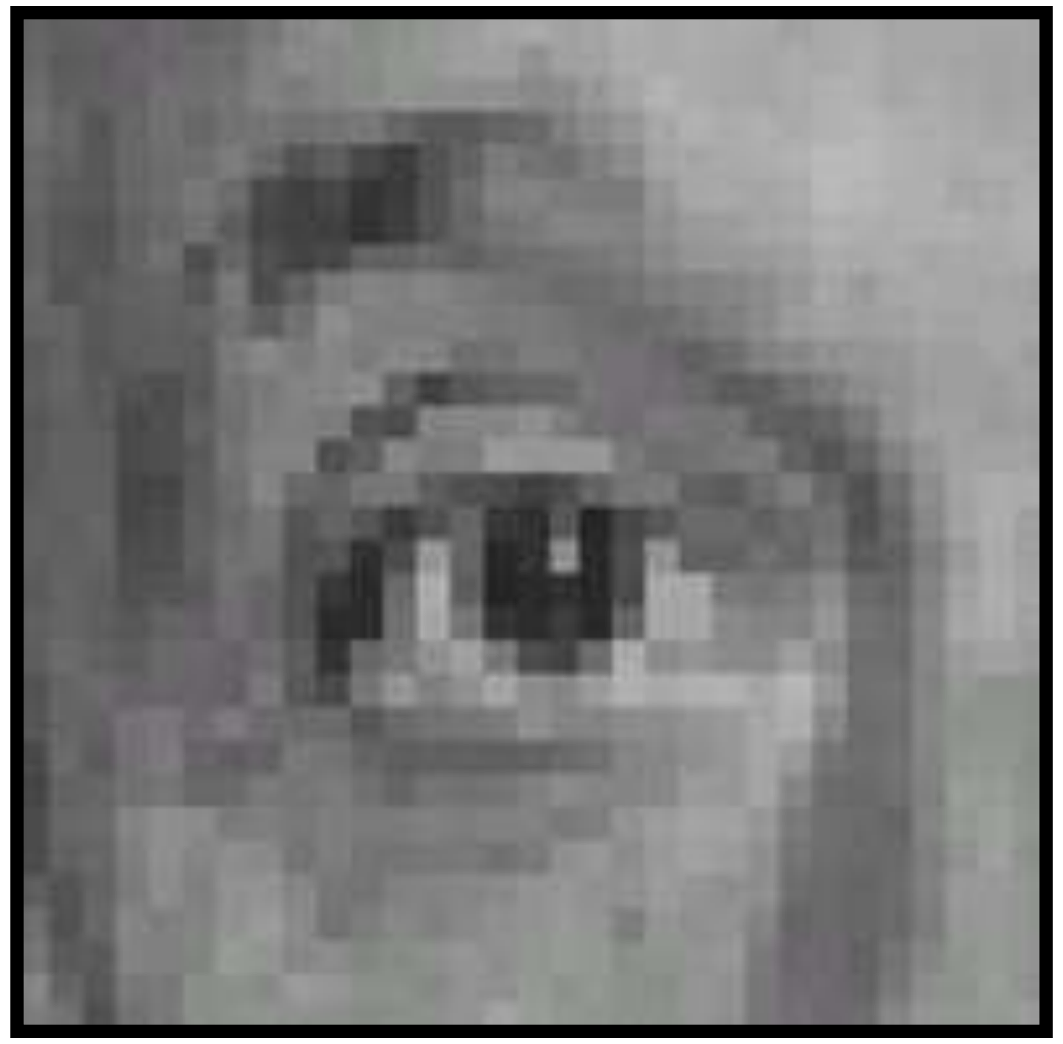
Filter

(filter sums to 1)

?

Result

Example 4:



Original

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

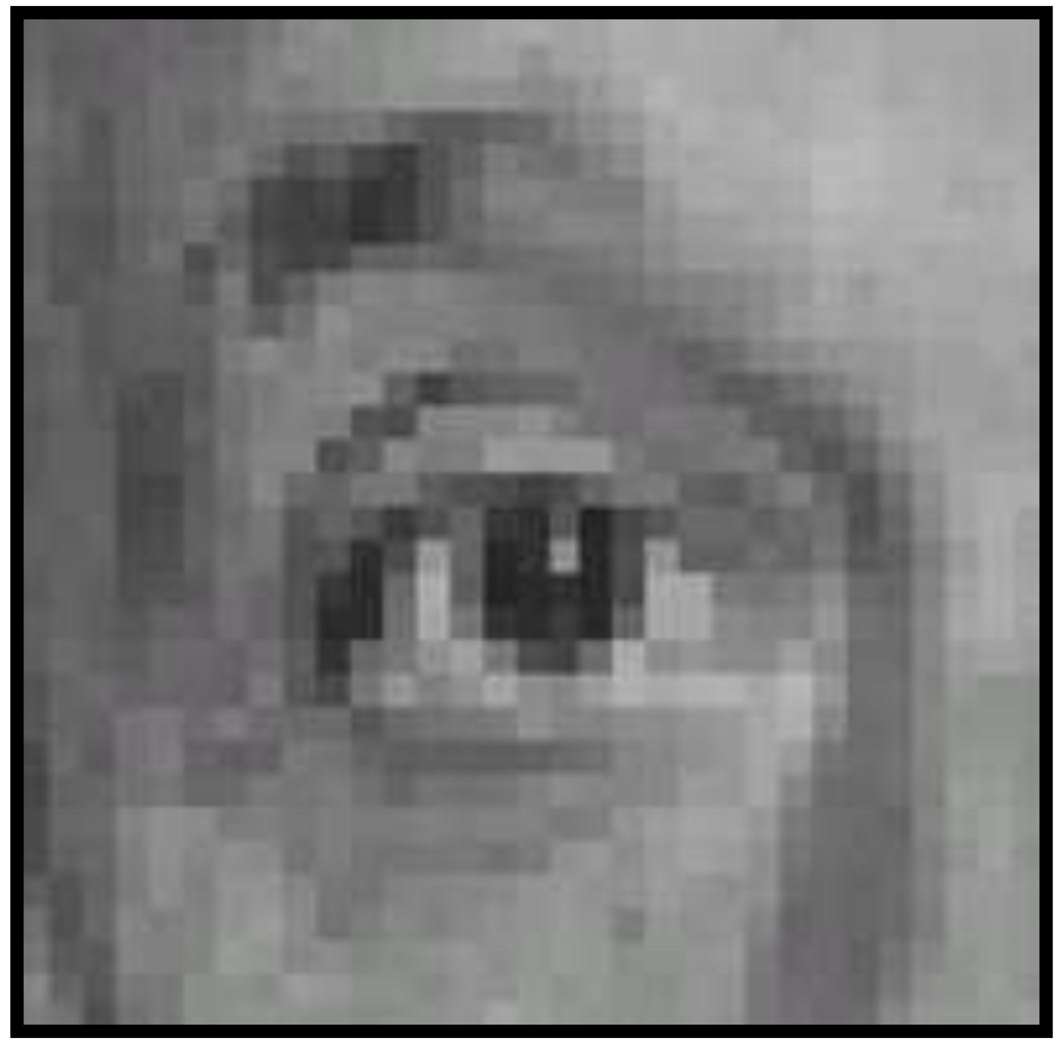
Filter

(filter sums to 1)

Result

(sharpening)

Example 4:



Original

(Scaled)
Image Itself

0	0	0
0	2	0
0	0	0

$-\frac{1}{9}$

Blurred Version

1	1	1
1	1	1
1	1	1

Filter

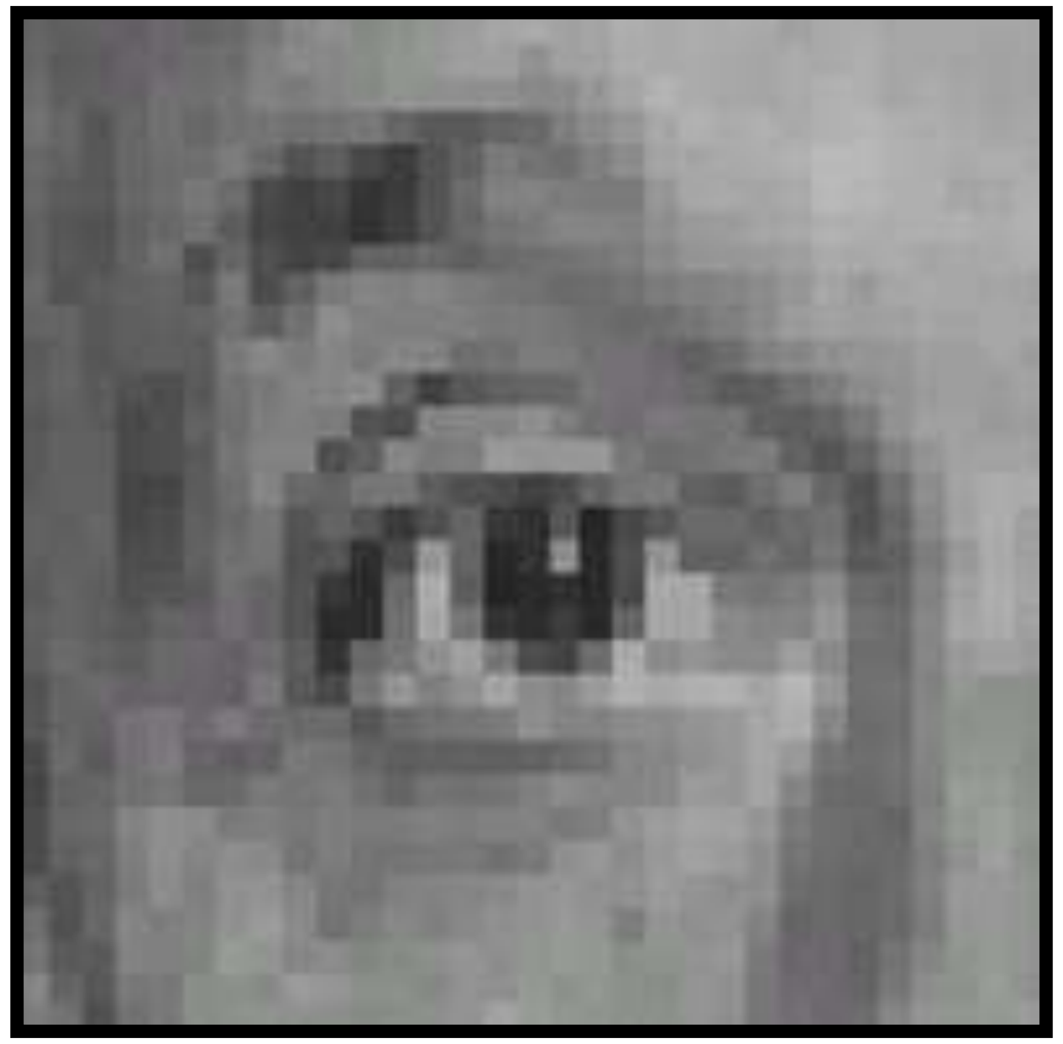
(filter sums to 1)

Result

(sharpening)

Example 4:

Why have filters sum up to 1?



Original

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

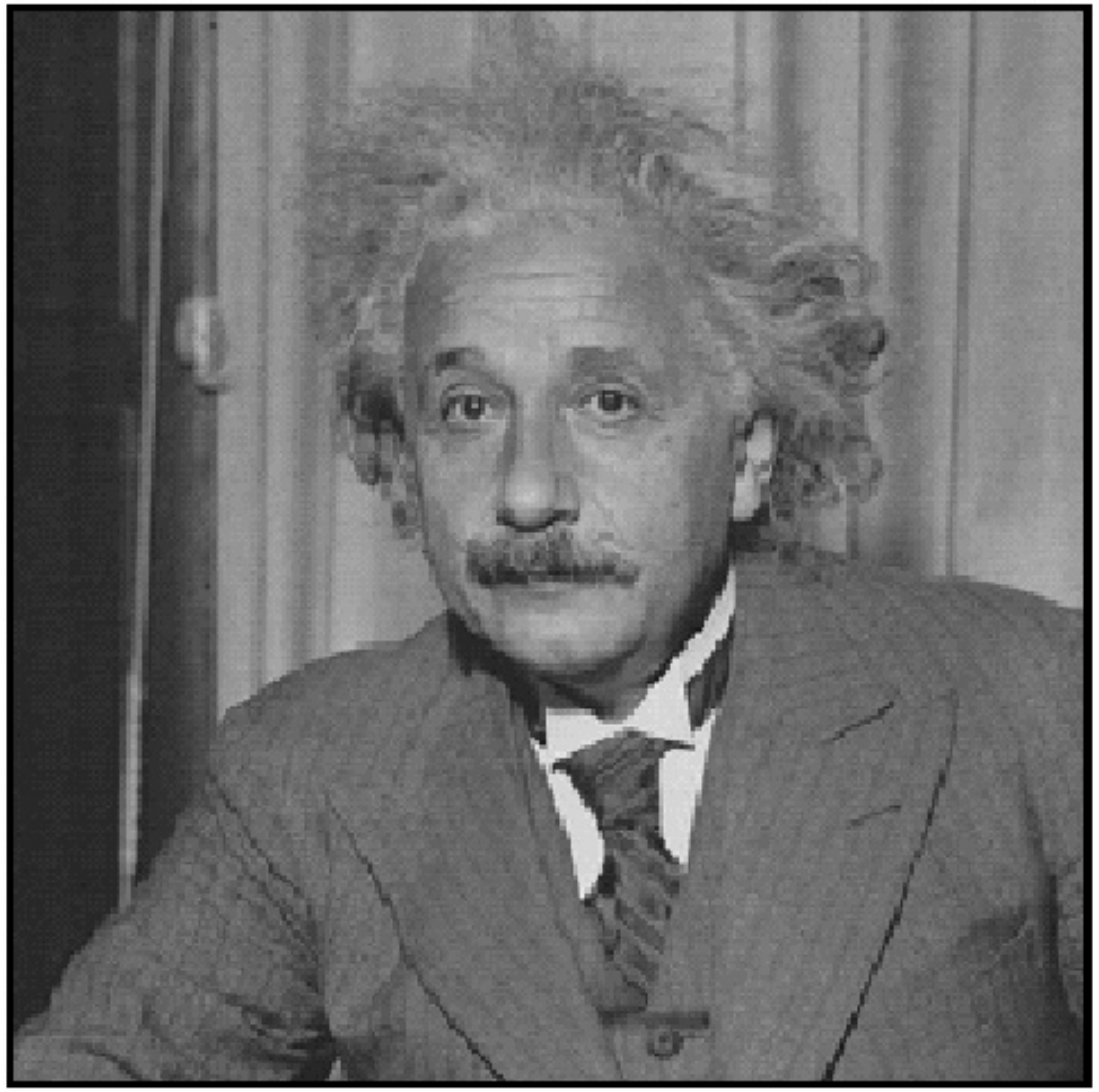
Filter

(filter sums to 1)

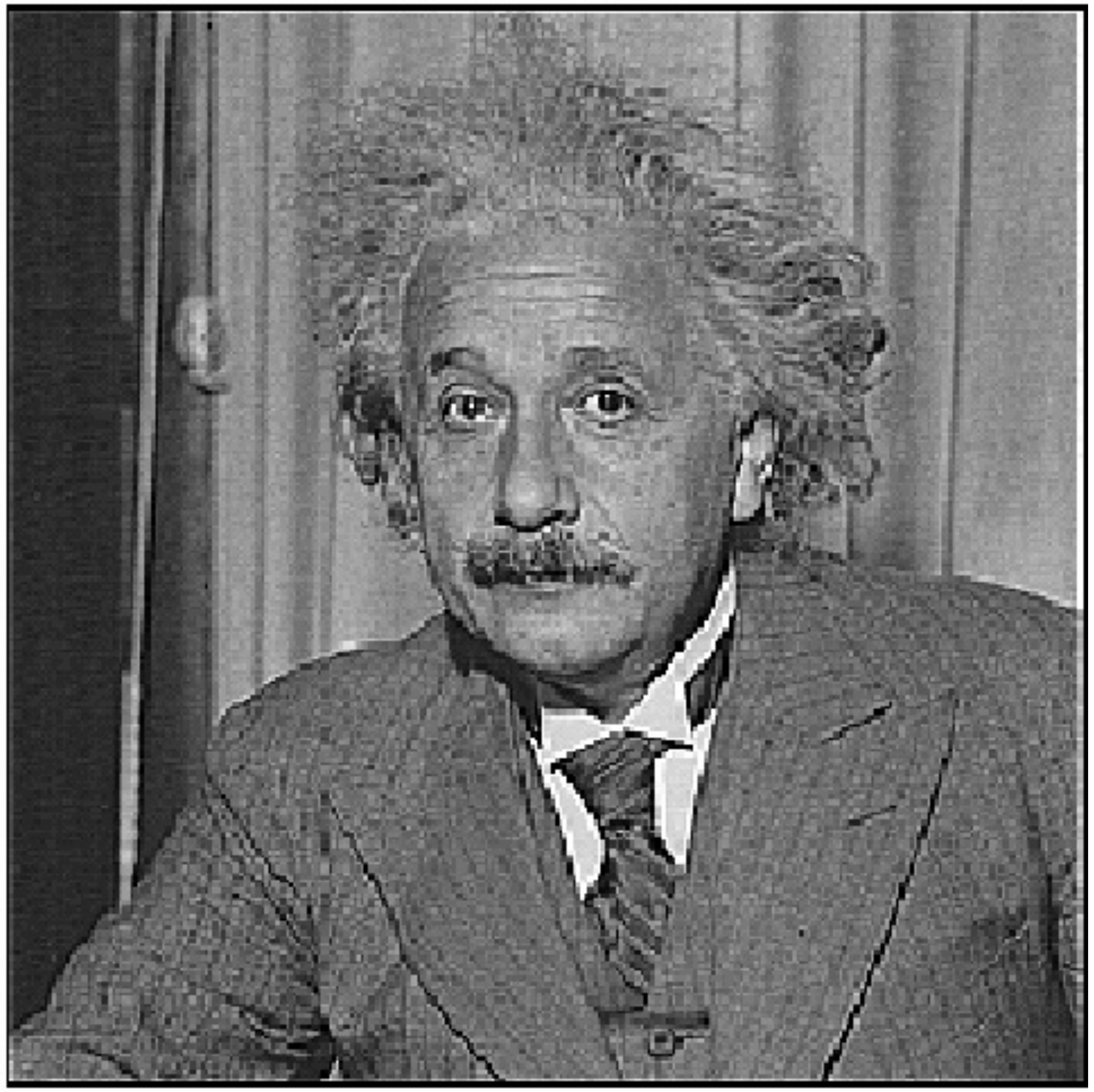
Result

(sharpening)

Example 4: Sharpening



Before



After

Example 4: Sharpening

Before

After

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

Definition: **Convolution**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X-i, Y-j)$$

Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

a	b	c
d	e	f
g	h	i

Filter

1	2	3
4	5	6
7	8	9

Image

Output

Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

a	b	c
d	e	f
g	h	i

Filter

1	2	3
4	5	6
7	8	9

Image

Output

= 1a + 2b + 3c
+ 4d + 5e + 6f
+ 7g + 8h + 9i

Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

Definition: **Convolution**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X-i, Y-j)$$

a	b	c
d	e	f
g	h	i

Filter

1	2	3
4	5	6
7	8	9

Image

Output

Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

Definition: **Convolution**

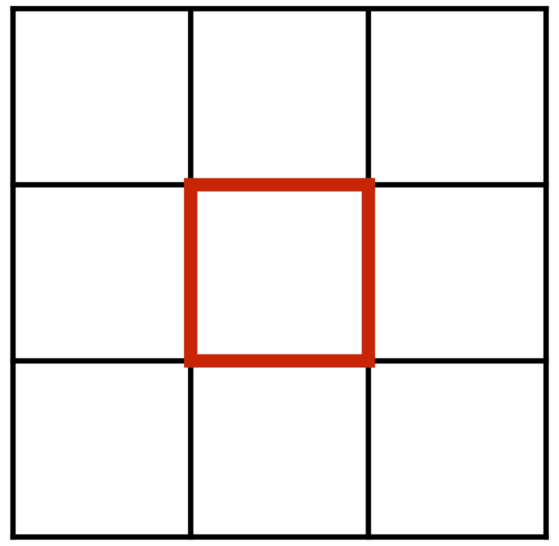
$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X-i, Y-j)$$

a	b	c
d	e	f
g	h	i

Filter

1	2	3
4	5	6
7	8	9

Image



Output

= 9a + 8b + 7c
+ 6d + 5e + 4f
+ 3g + 2h + 1i

Linear Filters: Correlation vs. Convolution

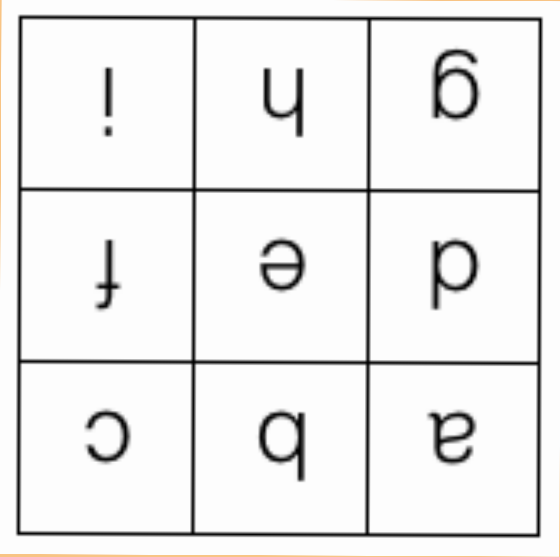
Definition: **Correlation**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

Definition: **Convolution**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X-i, Y-j)$$

Filter
(rotated by 180)

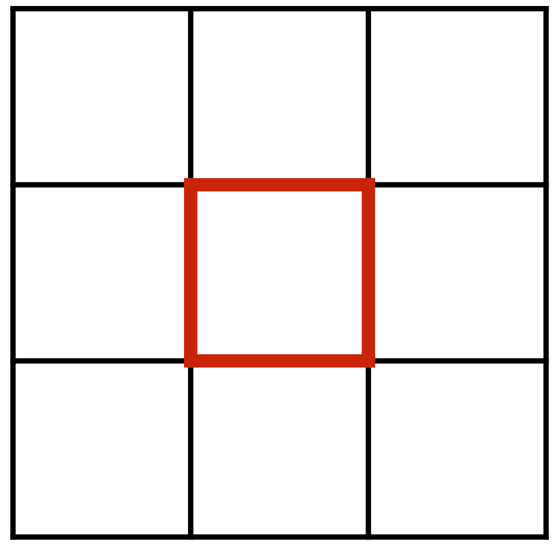


a	b	c
d	e	f
g	h	i

Filter

1	2	3
4	5	6
7	8	9

Image



Output

= 9a + 8b + 7c
+ 6d + 5e + 4f
+ 3g + 2h + 1i

Linear Filters: Correlation vs. Convolution

Definition: **Correlation**

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

Definition: **Convolution**

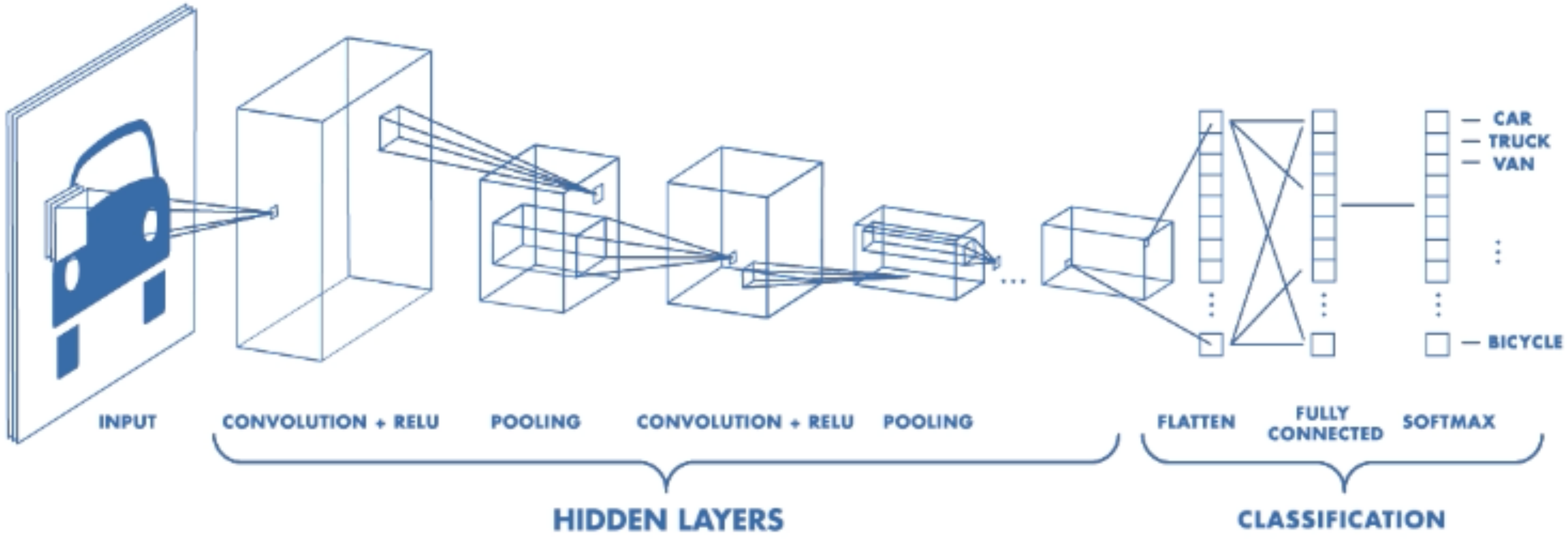
$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X-i, Y-j)$$

$$= \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(-i,-j) I(X+i, Y+j)$$

Note: if $F(X,Y) = F(-X,-Y)$ then correlation = convolution.

Preview: Why convolutions are important?

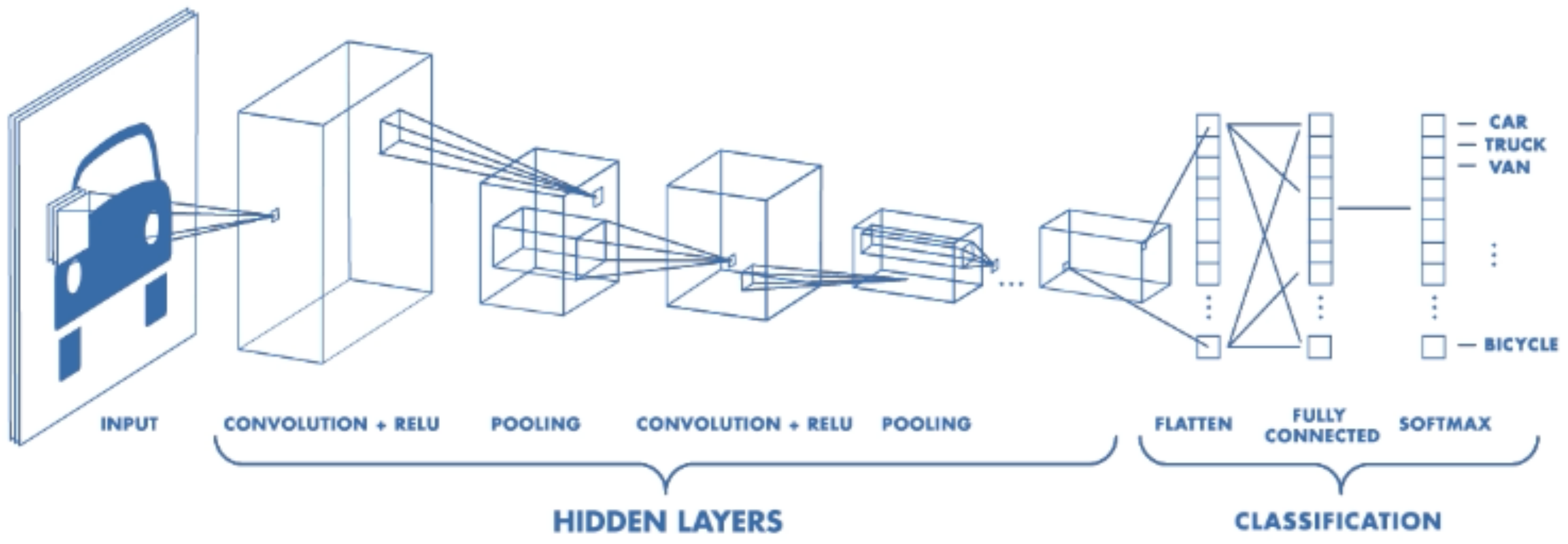
Who has heard of **Convolutional Neural Networks** (CNNs)?



Preview: Why convolutions are important?

Who has heard of **Convolutional Neural Networks** (CNNs)?

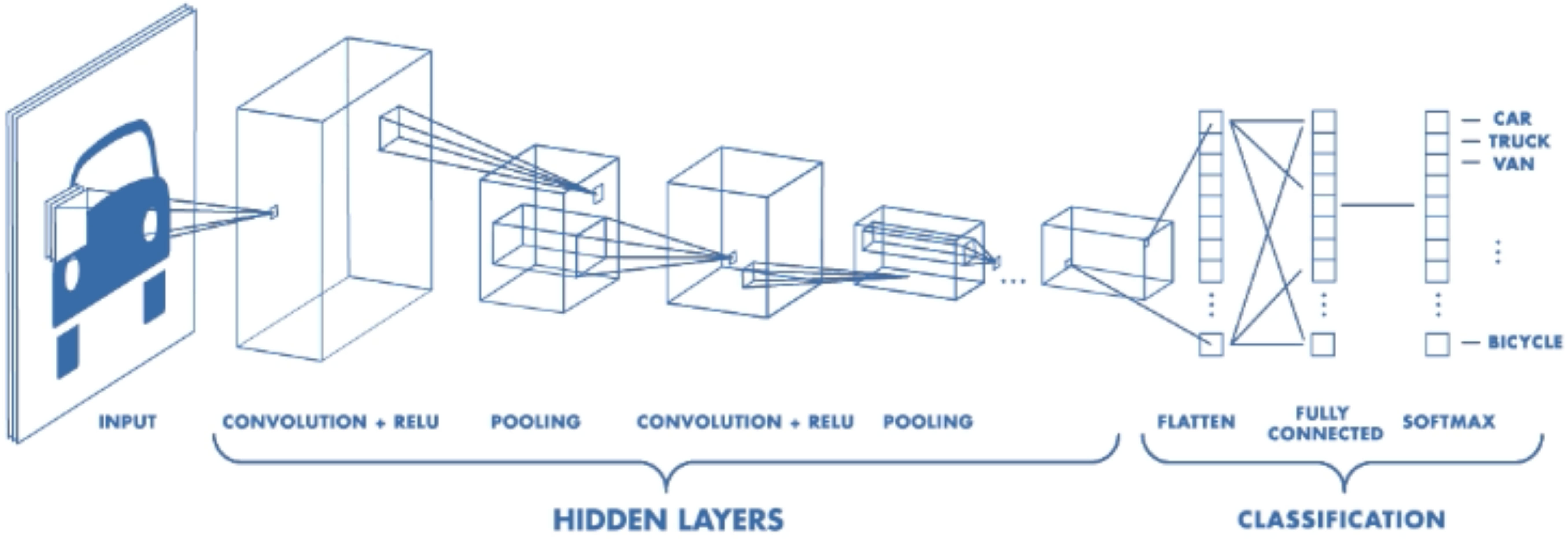
What about **Deep Learning**?



Preview: Why convolutions are important?

Who has heard of **Convolutional Neural Networks** (CNNs)?

What about **Deep Learning**?



Basic operations in CNNs are convolutions (with learned linear filters) followed by non-linear functions.

Note: This results in non-linear filters.