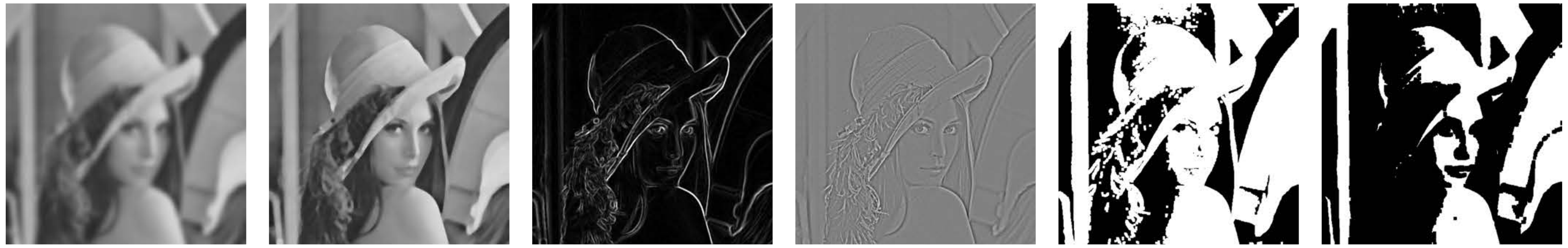
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 4: Image Filtering (continued)

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

Menu for Today

Topics:

- — Recap L3, more examples
- — **Box, Gaussian, Pillbox** filters
- — **Low/High Pass** Filters
- — **Separability**

Readings:

- — **Today's** Lecture: none
- — **Next** Lecture: Forsyth & Ponce (2nd ed.) 4.4

Reminders:

- — **Assignment 1:** Image Filtering and Hybrid Images

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 $I'(X,Y)$

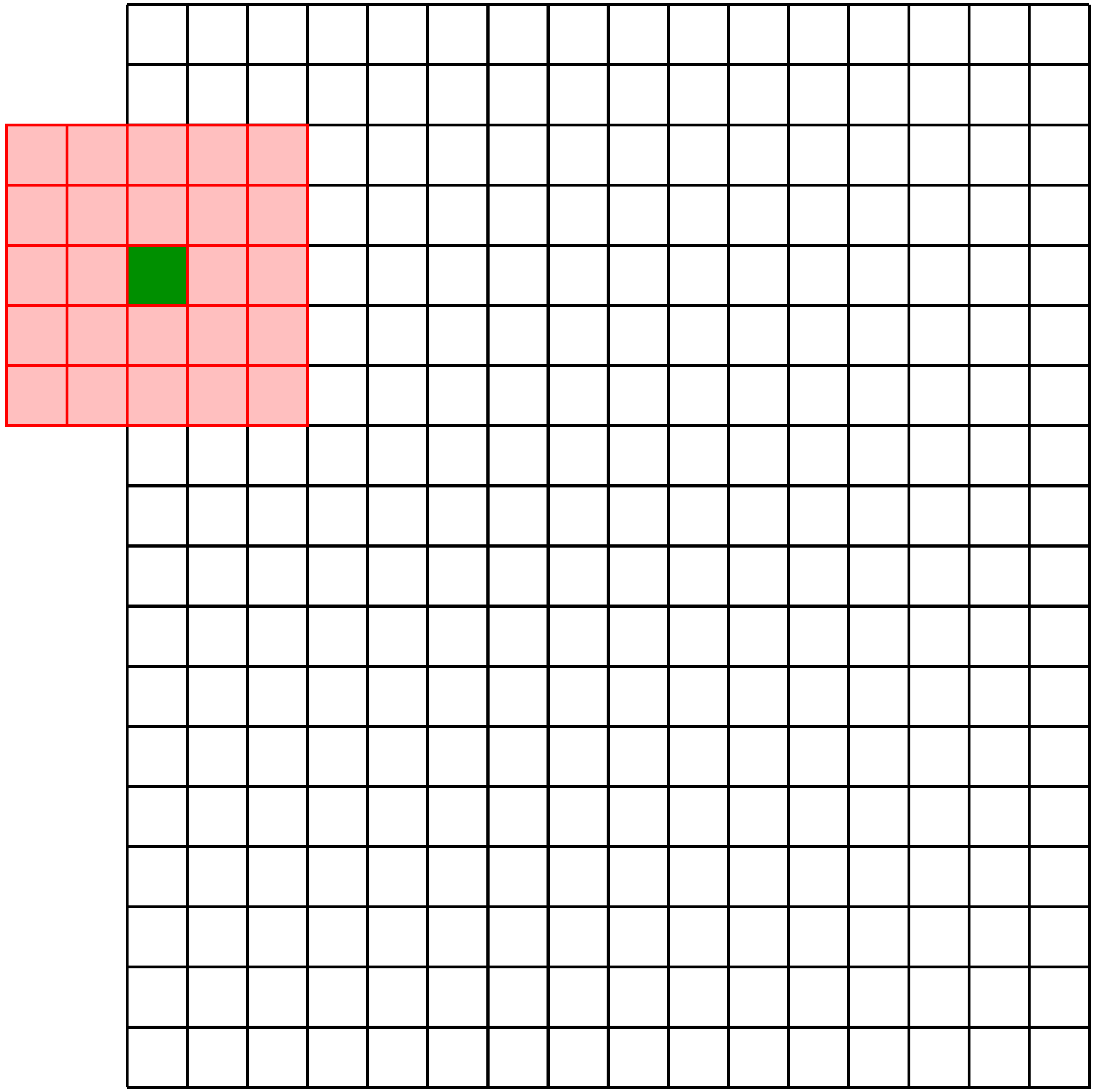
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

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

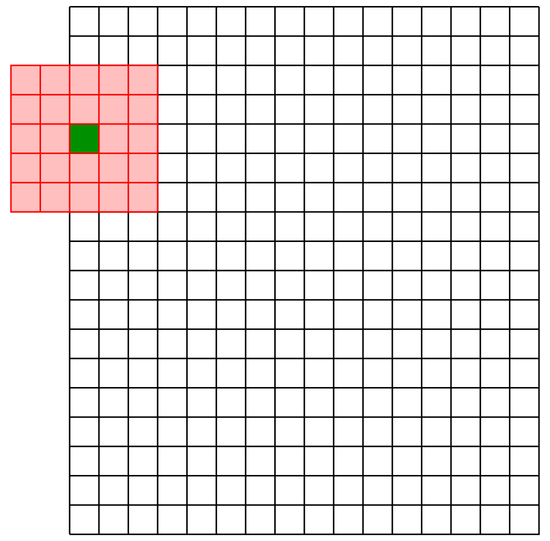
output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filters: **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

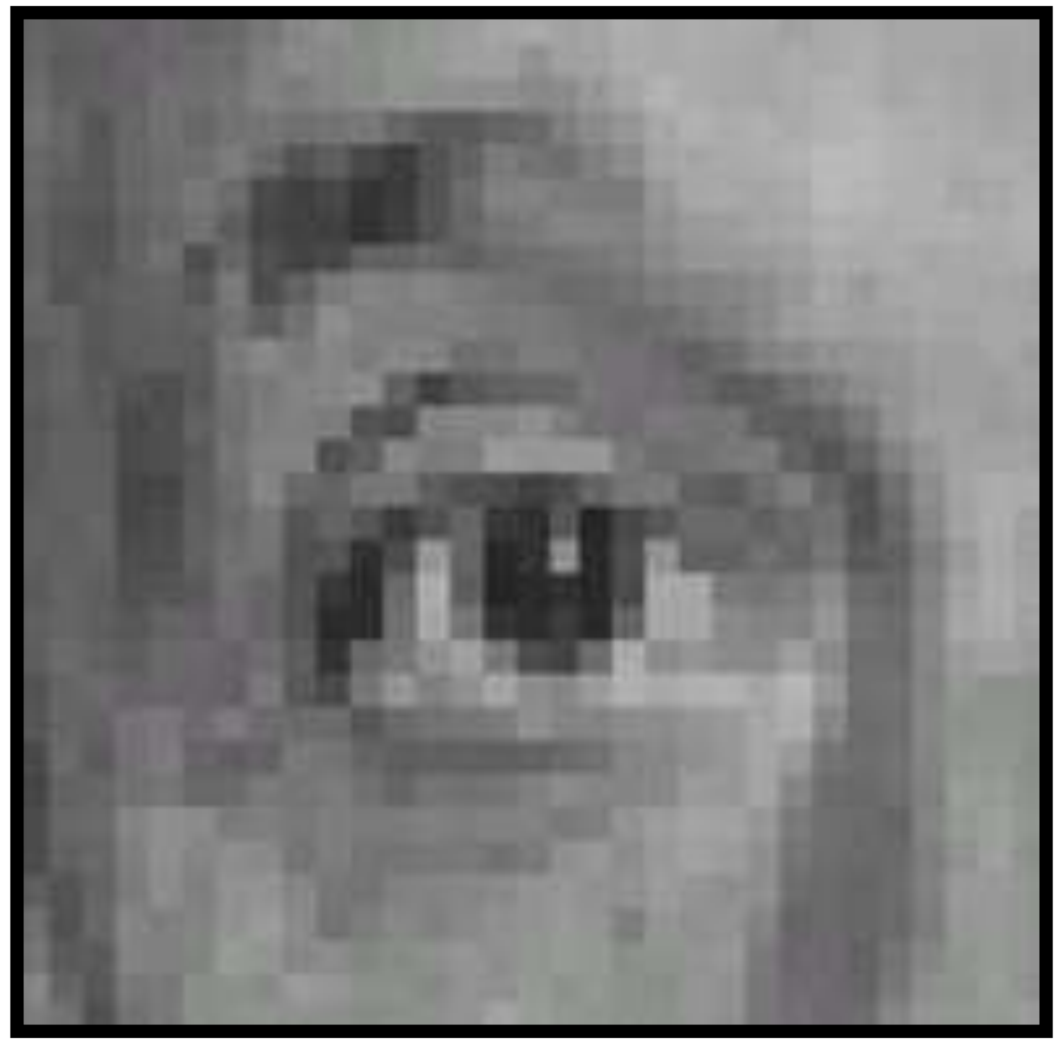
2. **Pad the image with zeros**: Return zero whenever a value of I is required at some position outside the defined limits of X and Y

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

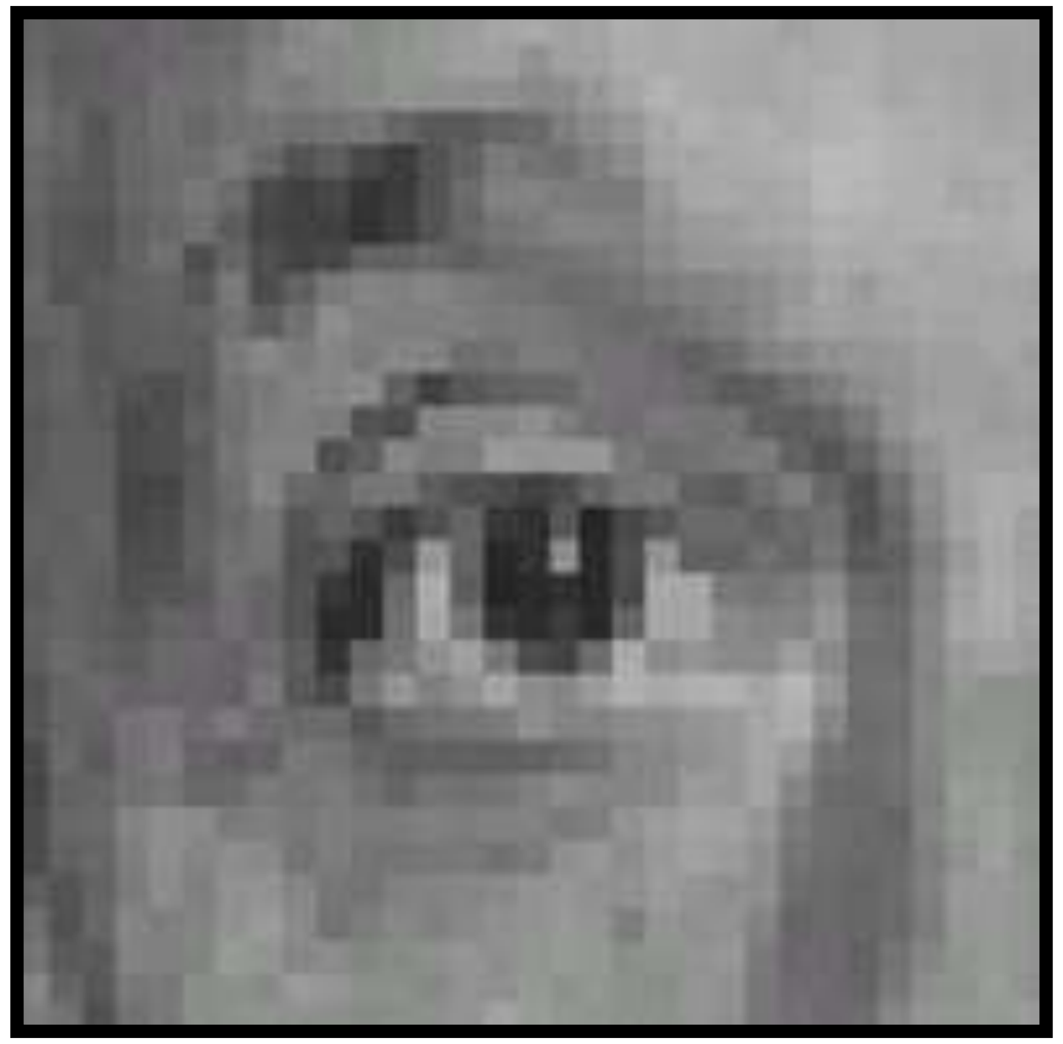
0	0	0
0	1	0
0	0	0

Filter

?

Result

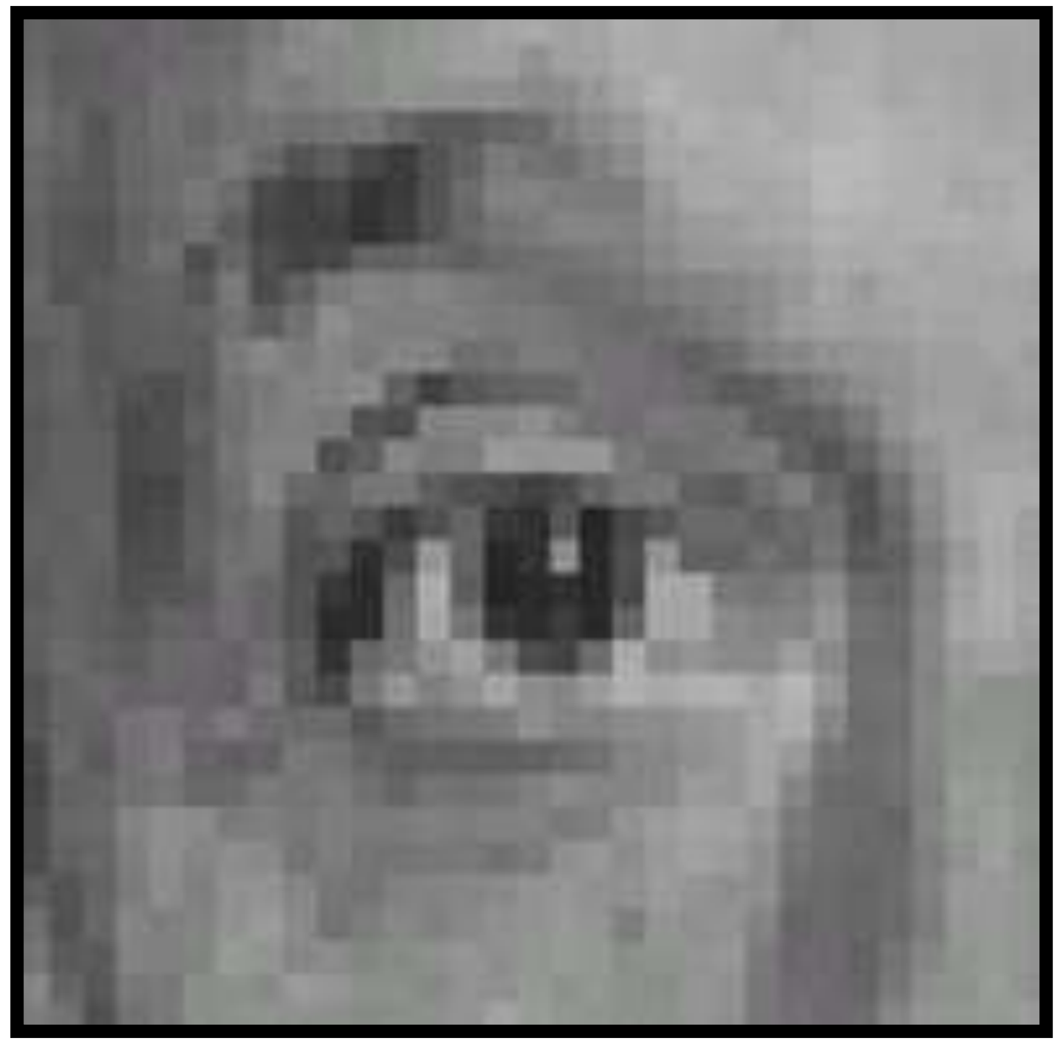
Example 1: Warm up



Original

0	0	0
0	1	0
0	0	0

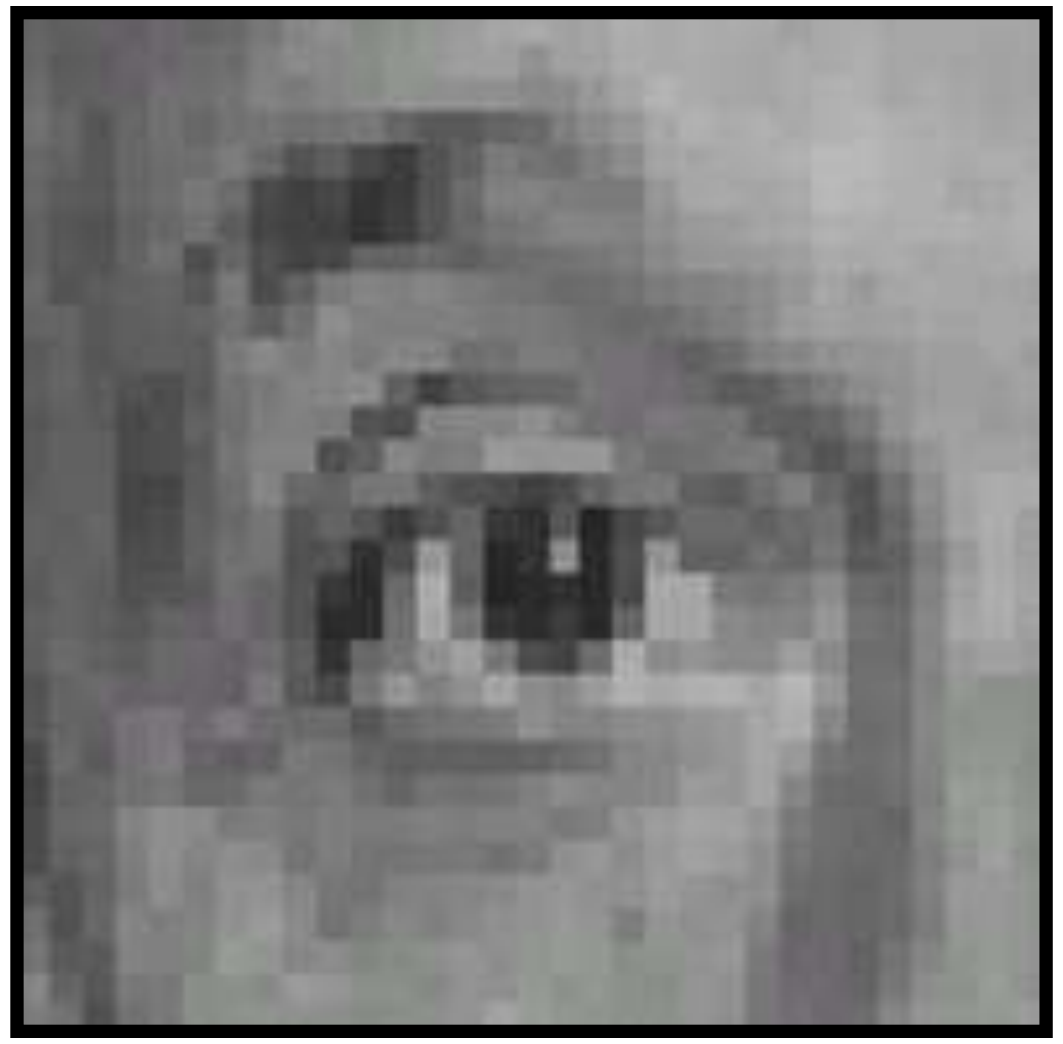
Filter



Result

(no change)

Example 2:



Original

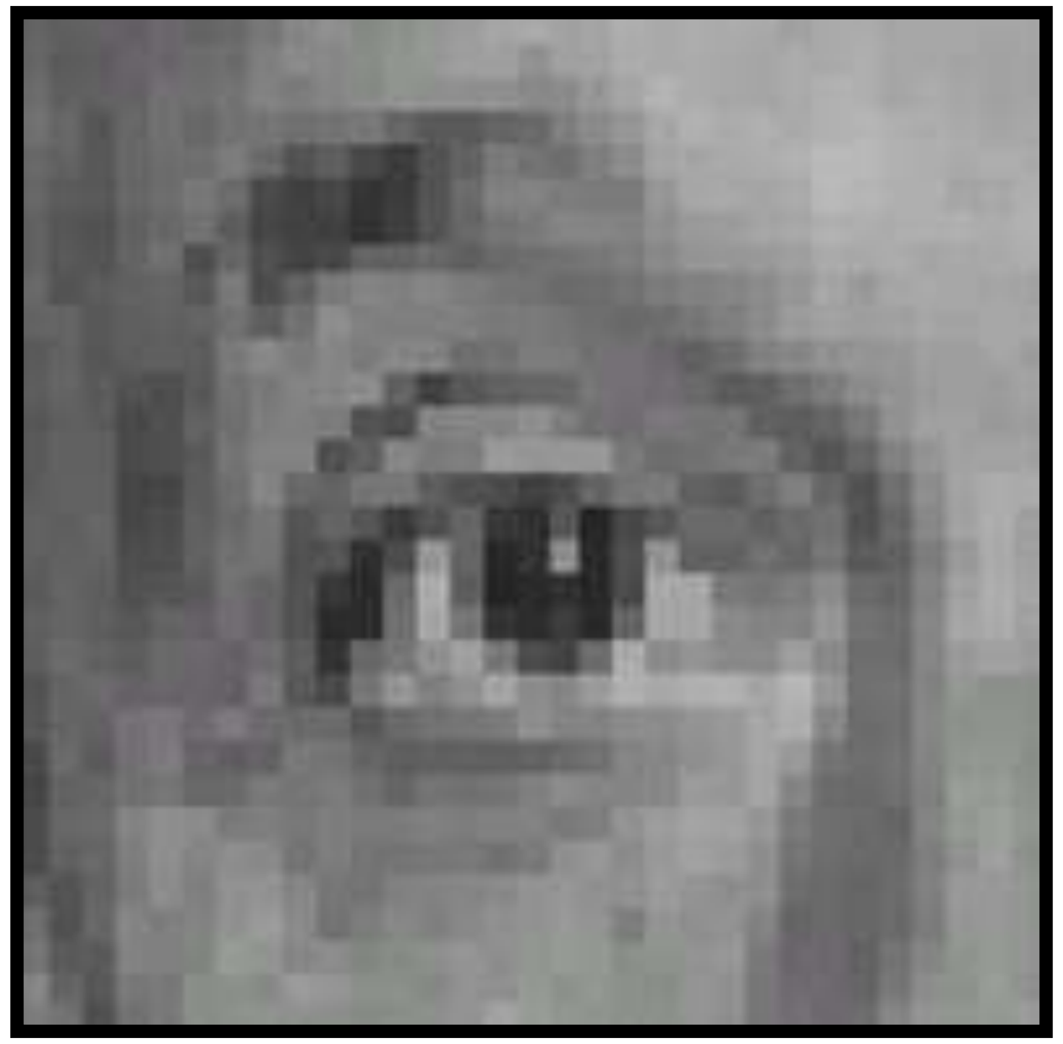
0	0	0
0	0	1
0	0	0

Filter

?

Result

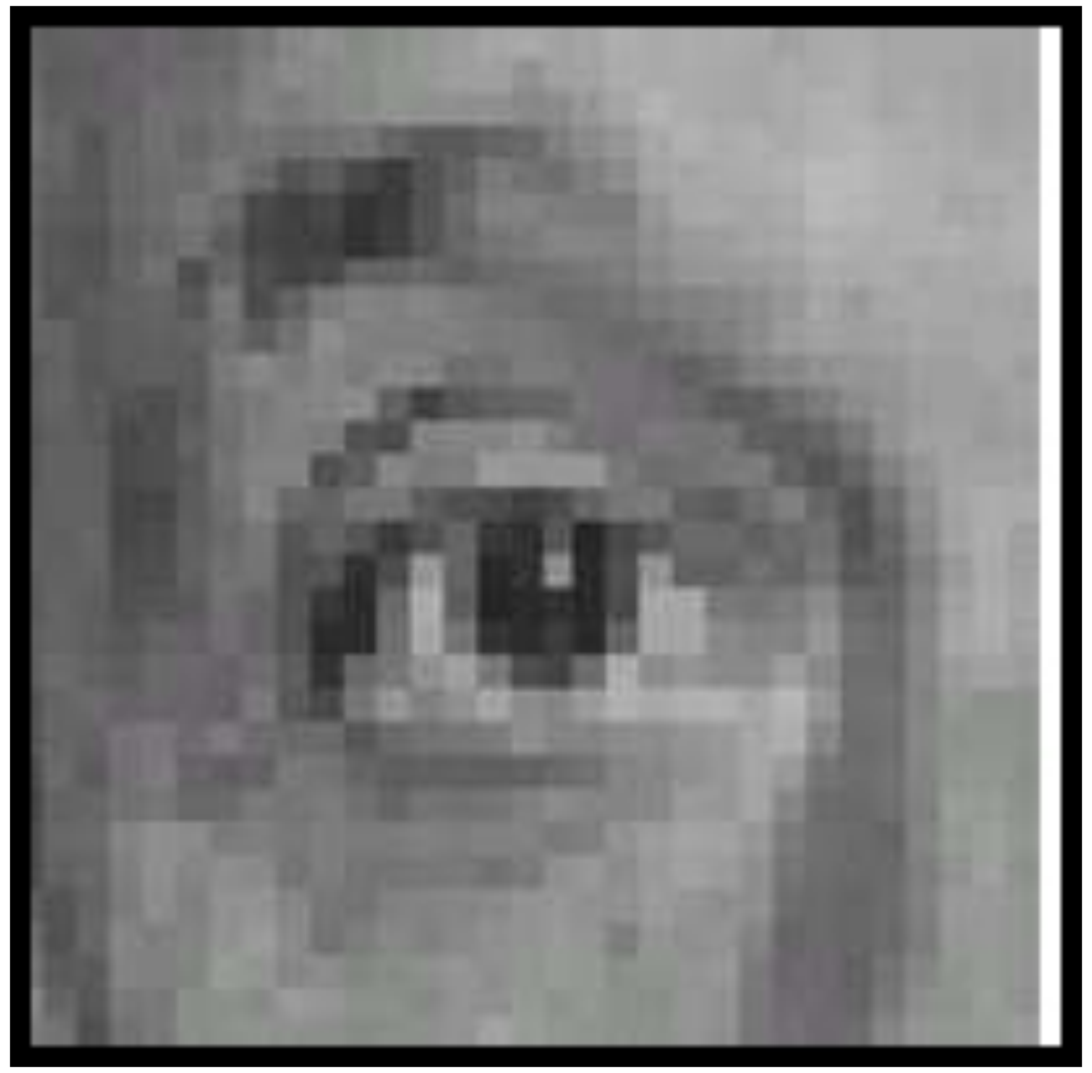
Example 2:



Original

0	0	0
0	0	1
0	0	0

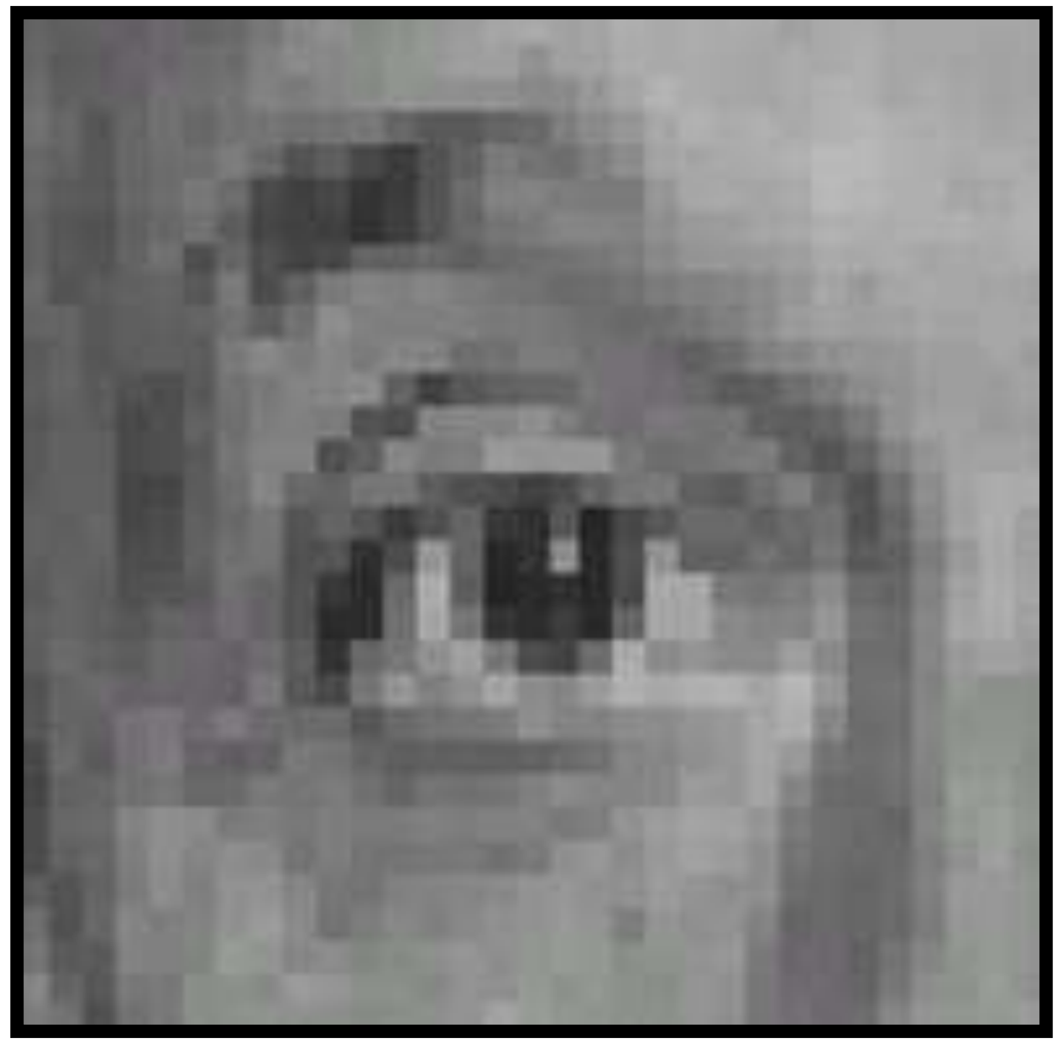
Filter



Result

(sift left by 1 pixel)

Example 3:



Original

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

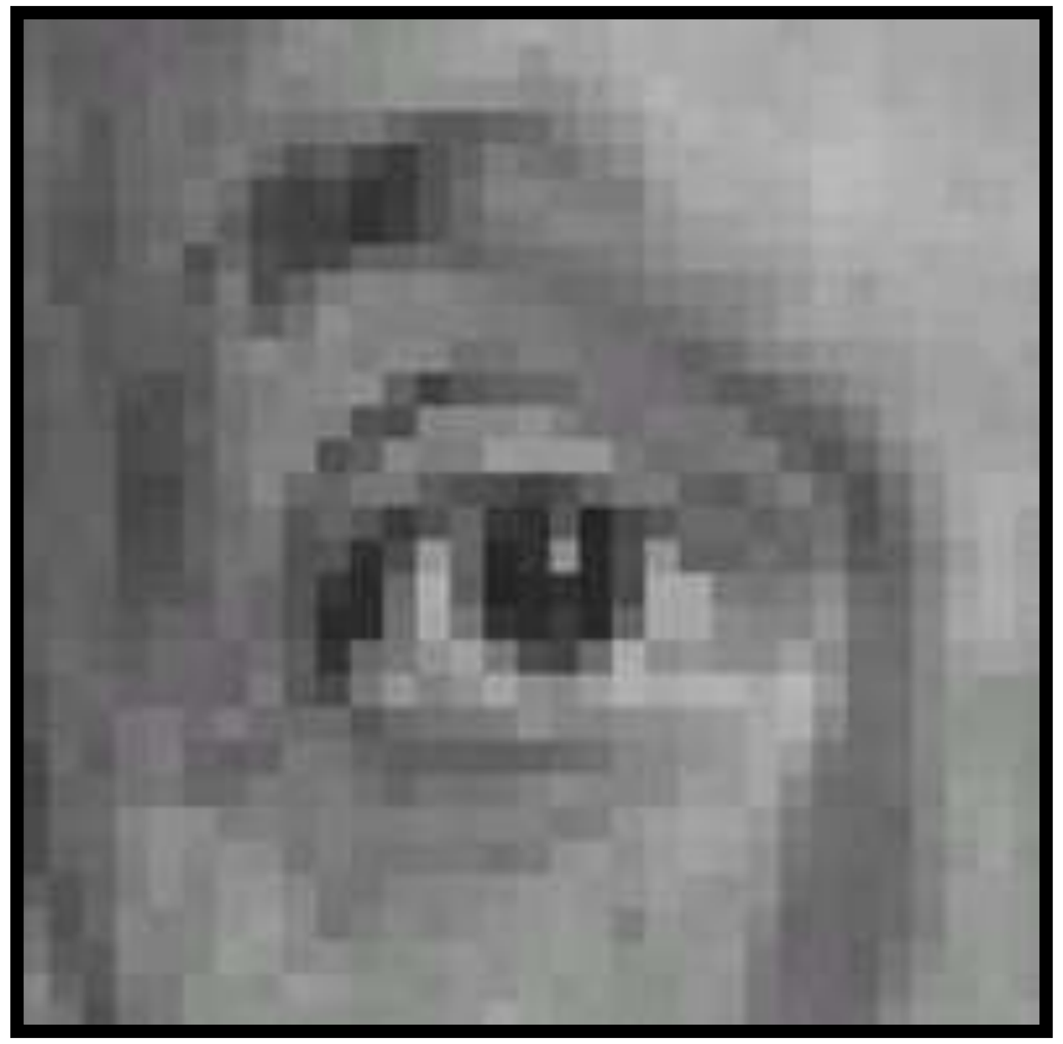
Filter

(filter sums to 1)

?

Result

Example 3:

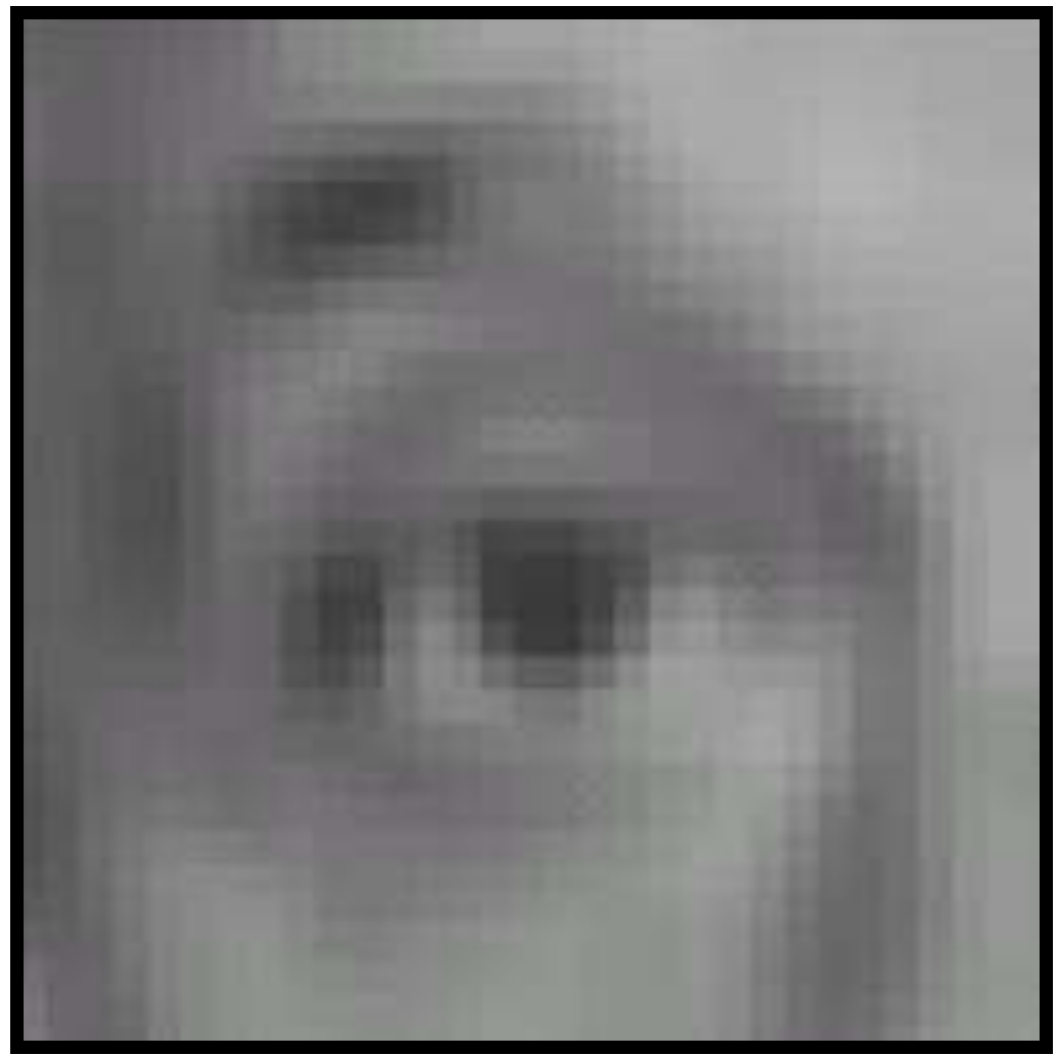


Original

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

Filter

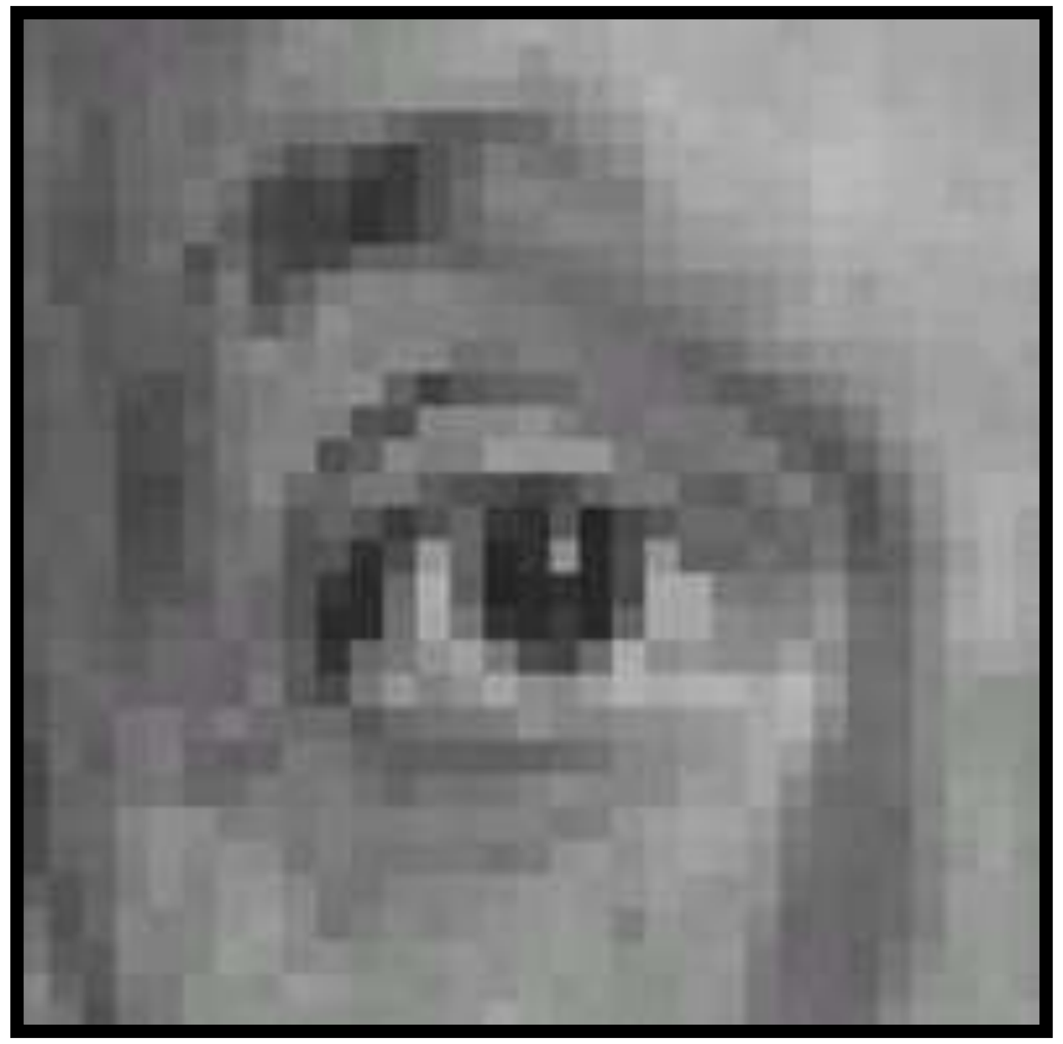
(filter sums to 1)



Result

(blur with a box filter)

Example 4:



Original

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

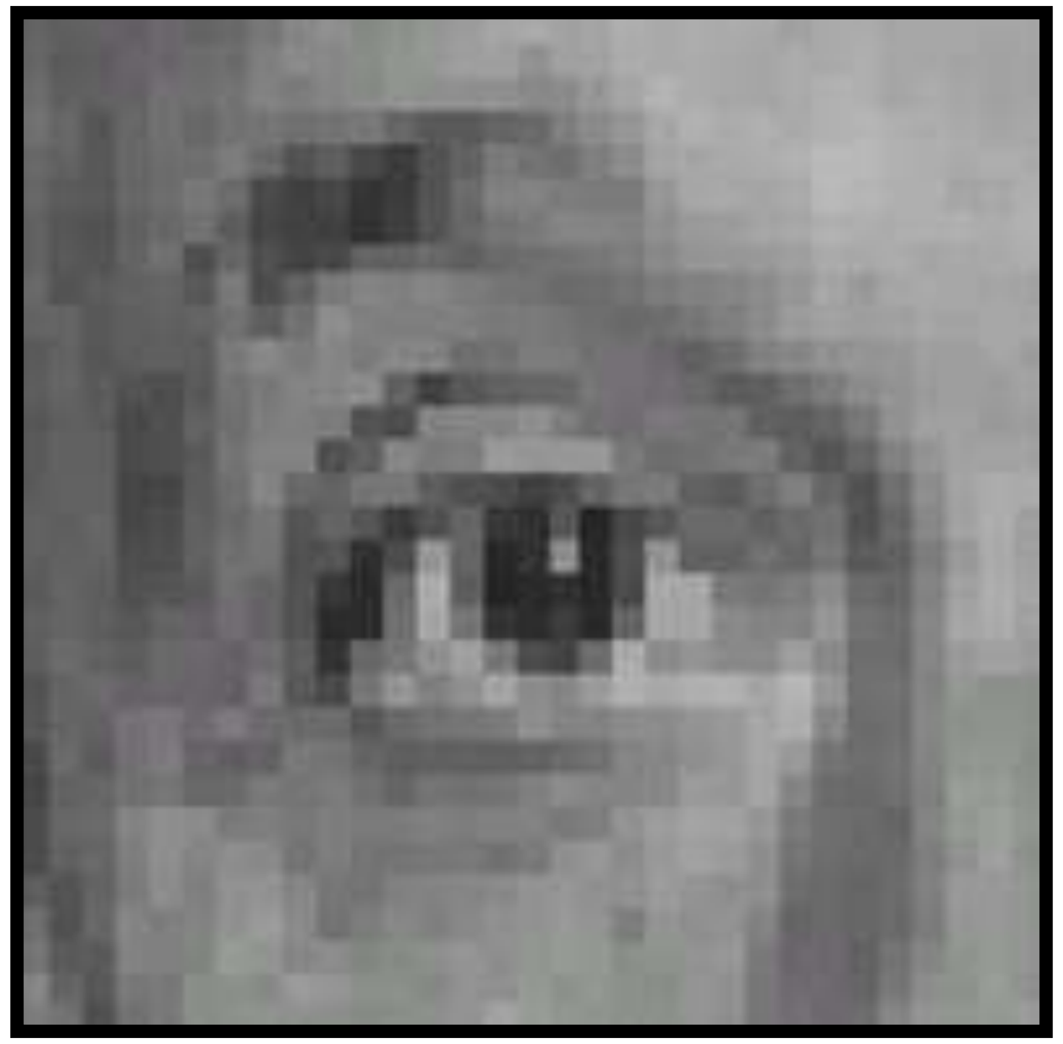
Filter

(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

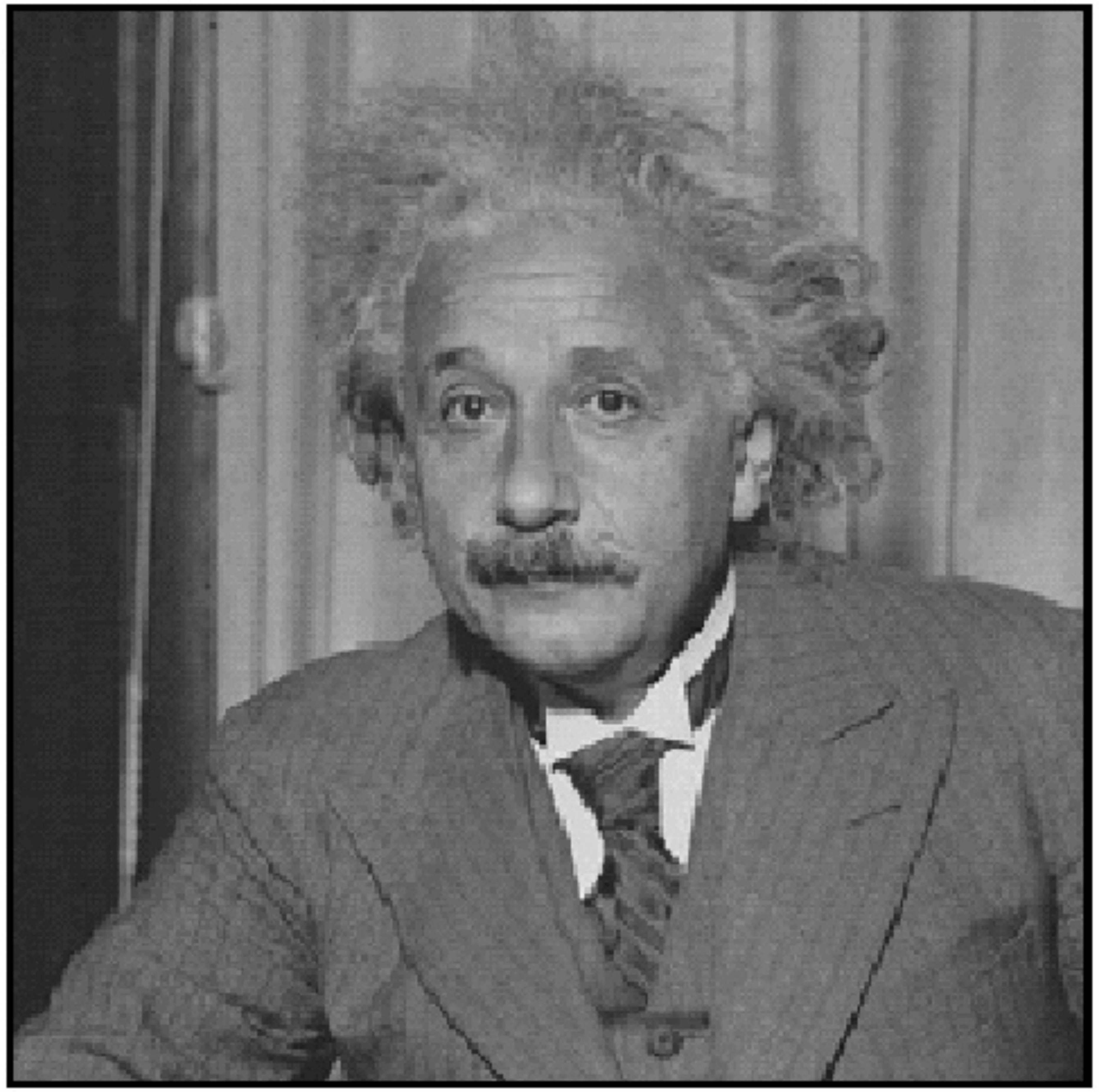
(filter sums to 1)

Why?

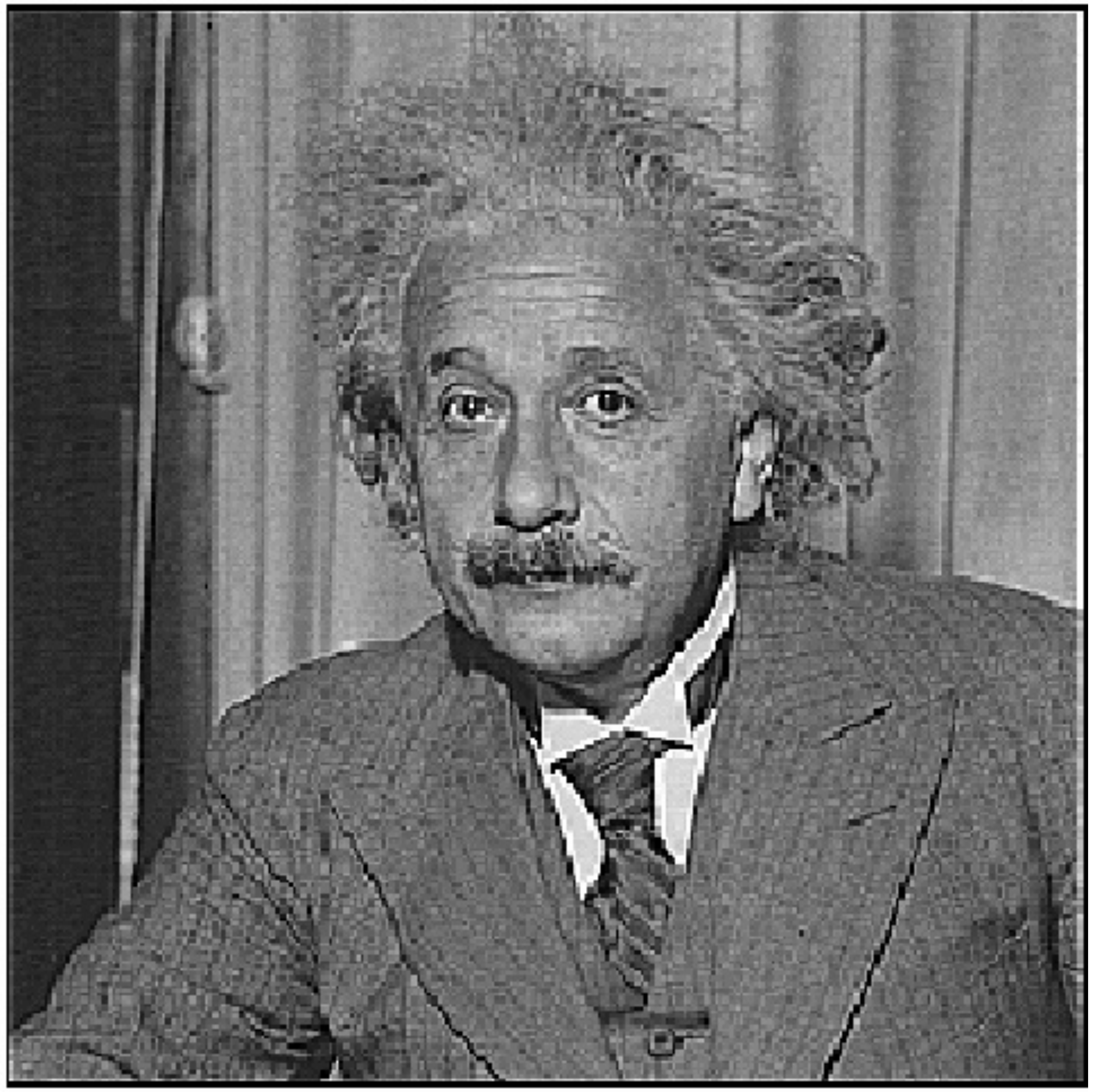
Result

(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**

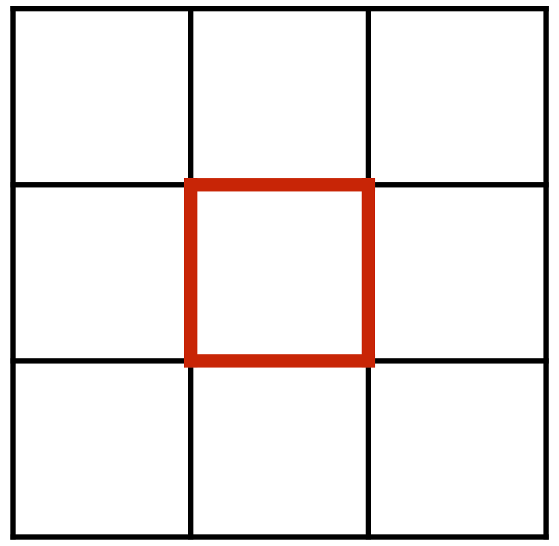
$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

a	b	c
d	e	f
g	h	i

Filter

1	2	3
4	5	6
7	8	9

Image



Output

= 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**

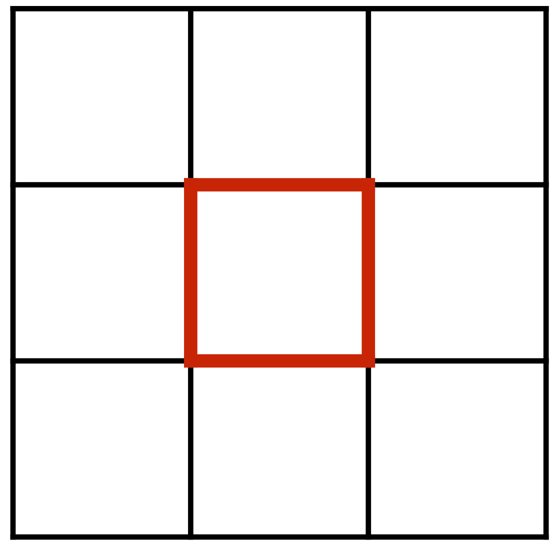
$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X-i, Y-j)$$

a	b	c
d	e	f
g	h	i

Filter

1	2	3
4	5	6
7	8	9

Image



Output

= 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)

i	h	g
f	e	d
c	b	a

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**

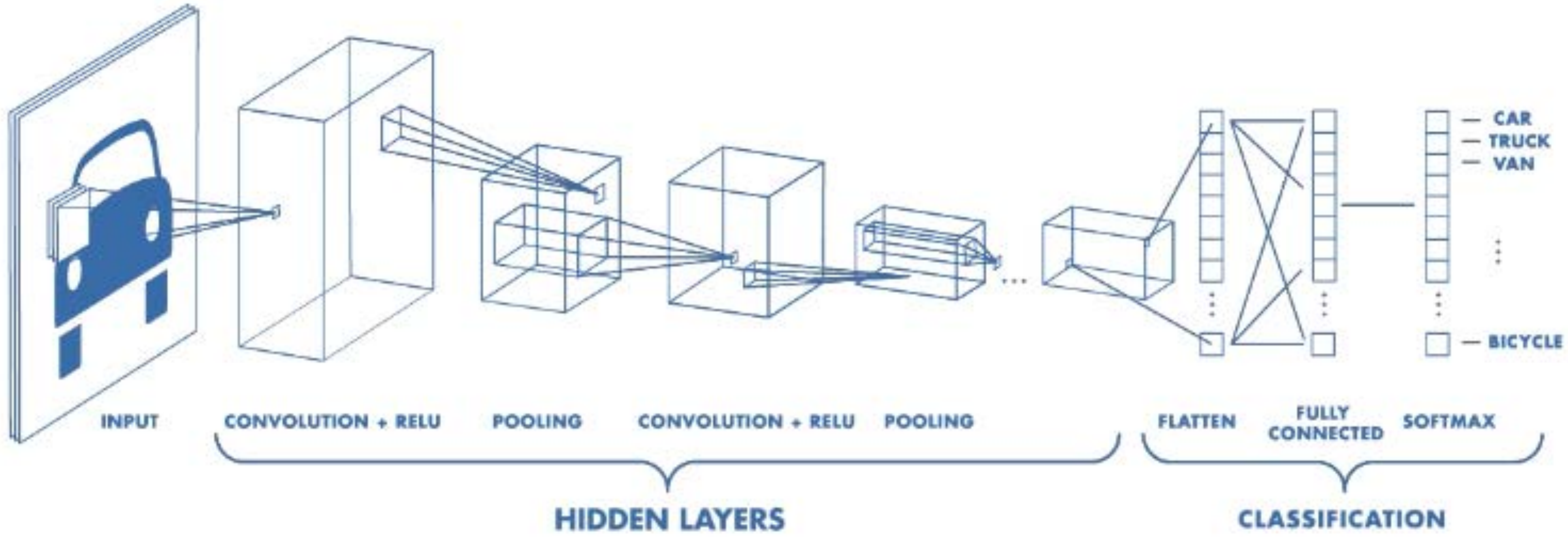
$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X-i, Y-j)$$

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

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

Preview: Why convolutions are important?

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



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

Note: This results in non-linear filters.

Linear Filters: **Properties**

3.3 Convolution as matrix multiplication

Linear Filters: **Properties**

Let $\otimes$ denote convolution. Let $I(X,Y)$ be a digital image

Superposition: Let F_1 and F_2 be digital filters

$$(F_1 + F_2) \otimes I(X,Y) = F_1 \otimes I(X,Y) + F_2 \otimes I(X,Y)$$

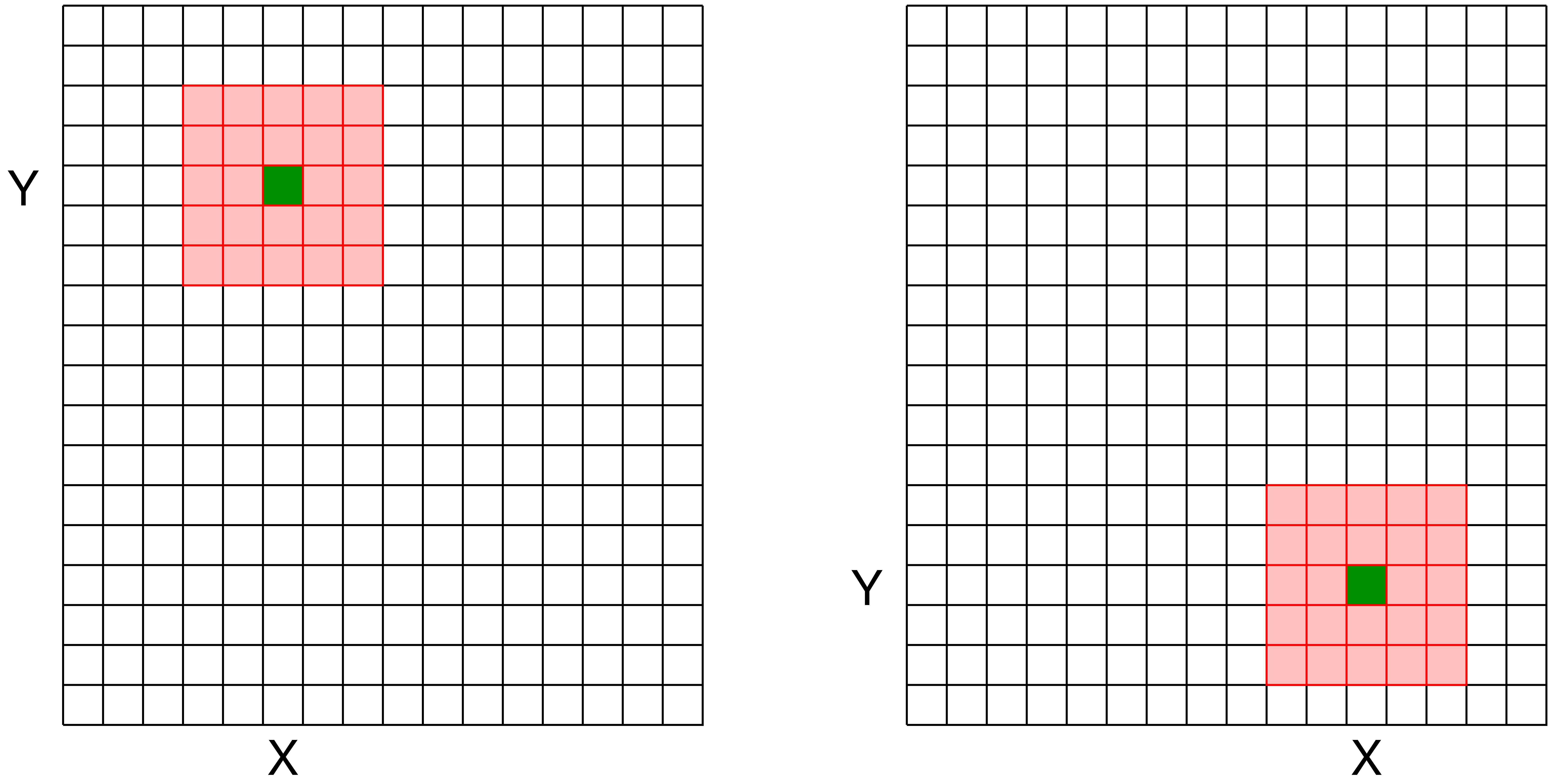
Scaling: Let F be digital filter and let k be a scalar

$$(kF) \otimes I(X,Y) = F \otimes (kI(X,Y)) = k(F \otimes I(X,Y))$$

Shift Invariance: Output is local (i.e., no dependence on absolute position)

Linear Filters: Shift Invariance

Same linear operation is applied everywhere, no dependence on absolute position



Linear Systems: Characterization Theorem

Any linear, shift invariant operation can be expressed as convolution

Up until now…

— The **correlation** of $F(X, Y)$ and $I(X, Y)$ is:

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

— **Visual interpretation**: Superimpose the filter F on the image I at (X, Y), perform an element-wise multiply, and sum up the values

— **Convolution** is like **correlation** except filter rotated 180°

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

Up until now…

Ways to handle **boundaries**

– **Ignore/discard**. Make the computation undefined for top/bottom k rows and left/right-most k columns

– **Pad with zeros**. Return zero whenever a value of I is required beyond the image bounds

– **Assume periodicity.** Top row wraps around to the bottom row; leftmost column wraps around to rightmost column.

Simple **examples** of filtering:

— copy, shift, smoothing, sharpening

Linear filter **properties**:

— superposition, scaling, shift invariance

Characterization Theorem: Any linear, shift-invariant operation can be expressed as a convolution

Smoothing

Smoothing (or blurring) is an important operation in a lot of computer vision

— Captured images are naturally **noisy**, smoothing allows removal of noise

— It is important for **re-scaling** of images, to avoid sampling artifacts

— Fake image **defocus** (e.g., depth of field) for artistic effects

(many other uses as well)

Smoothing with a **Box Filter**

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

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

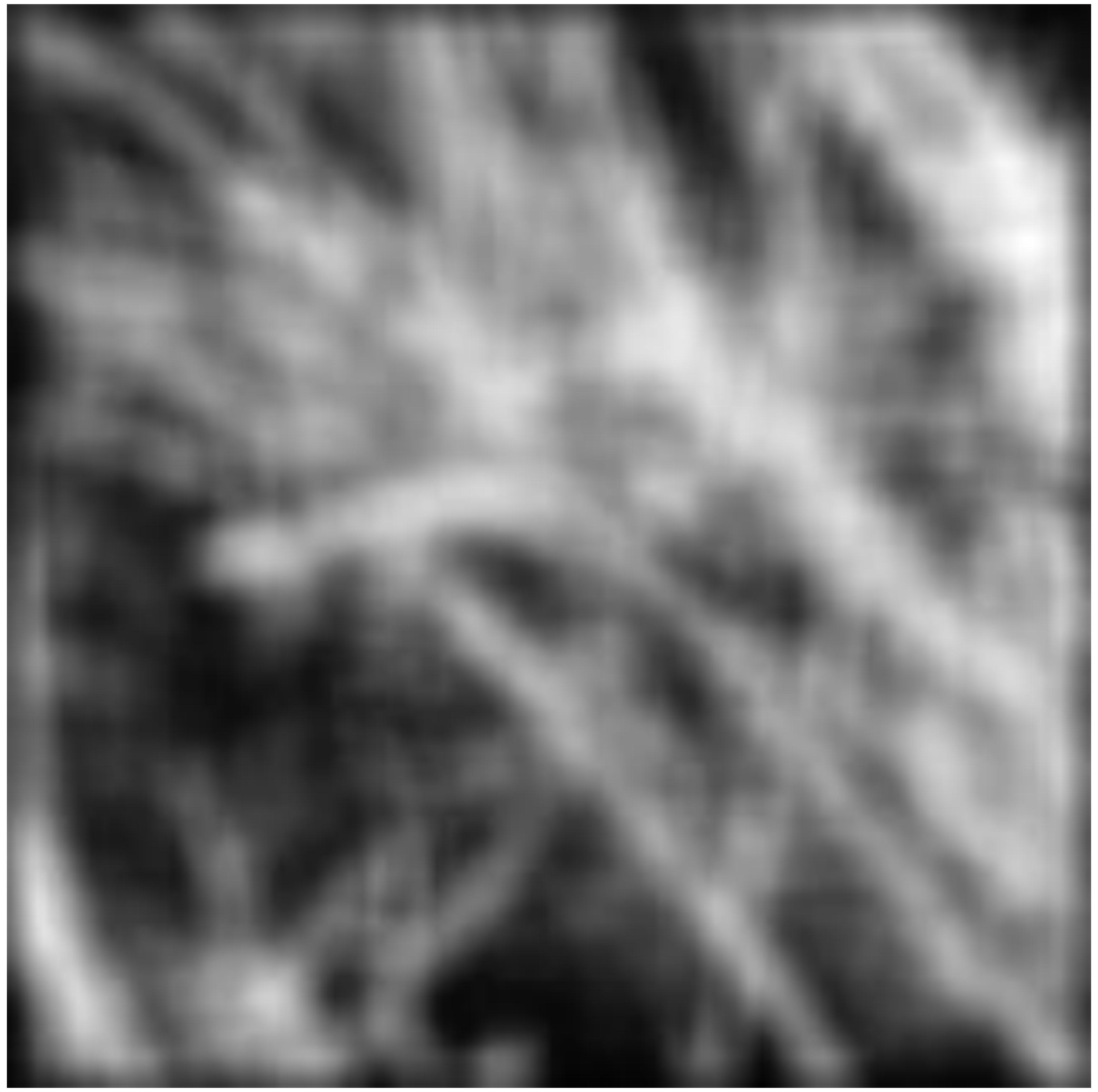
Filter has equal positive values that sum up to 1

Replaces each pixel with the average of itself and its local neighborhood

— Box filter is also referred to as **average filter** or **mean filter**

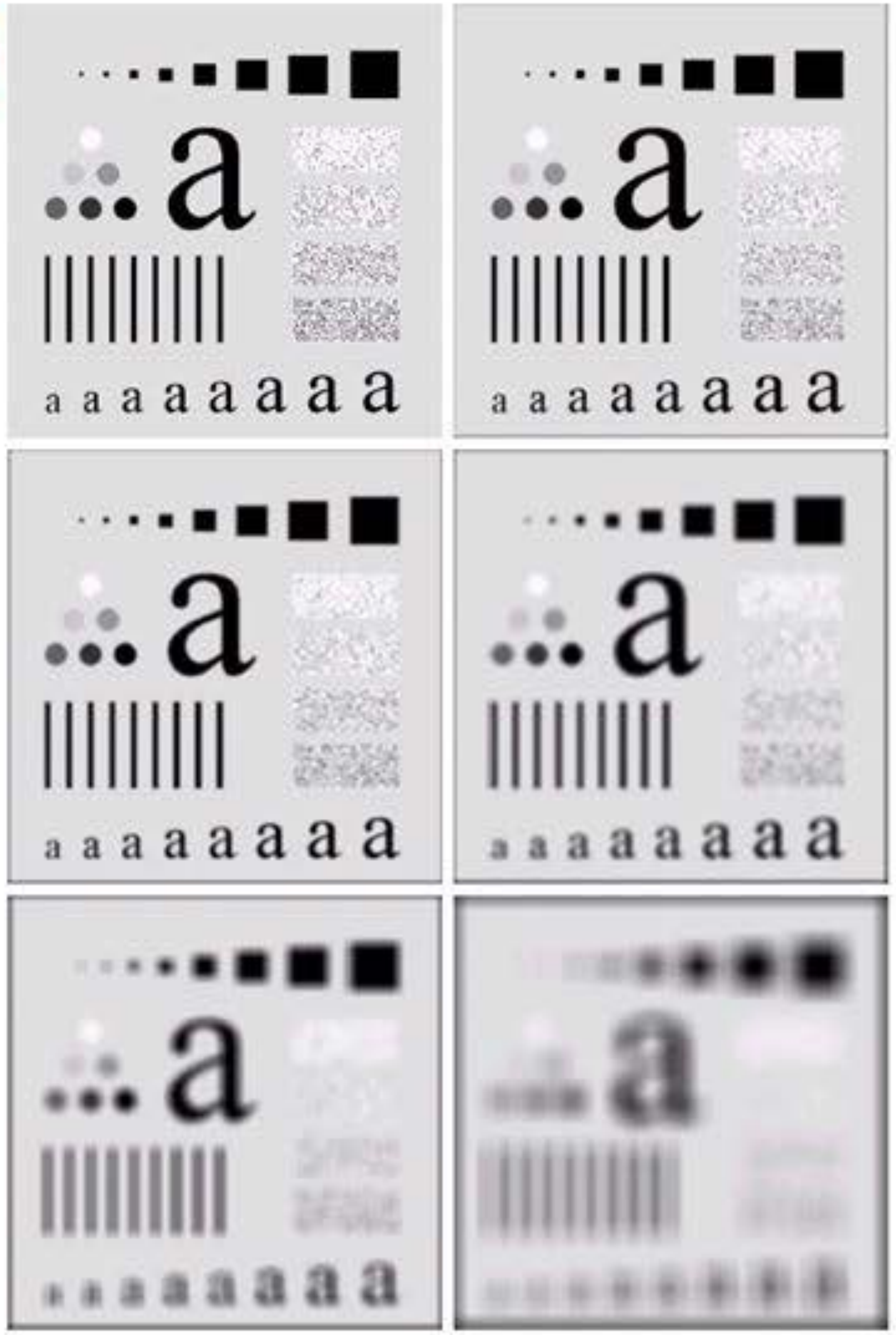
Why should the values sum to 1?

Smoothing with a **Box Filter**



Forsyth & Ponce (2nd ed.) Figure 4.1 (left and middle)

Smoothing with a **Box Filter**



Gonzales & Woods (3rd ed.) Figure 3.3

Smoothing with a **Box Filter**

Smoothing with a box **doesn't model lens defocus** well

— Smoothing with a box filter depends on direction

— e.g., Image in which the center point is 1 and every other point is 0

— Point spread function is a box

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Filter

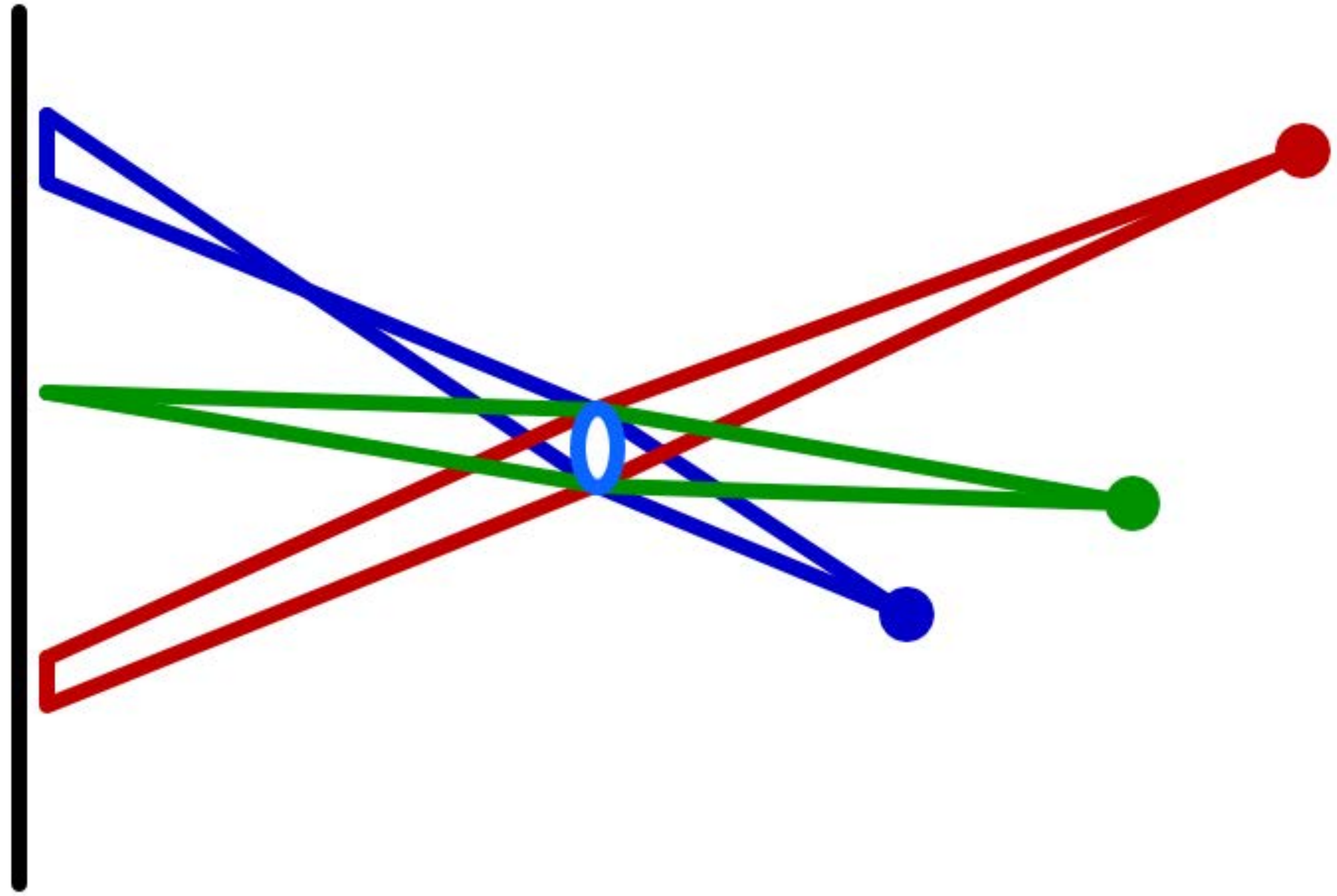
0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

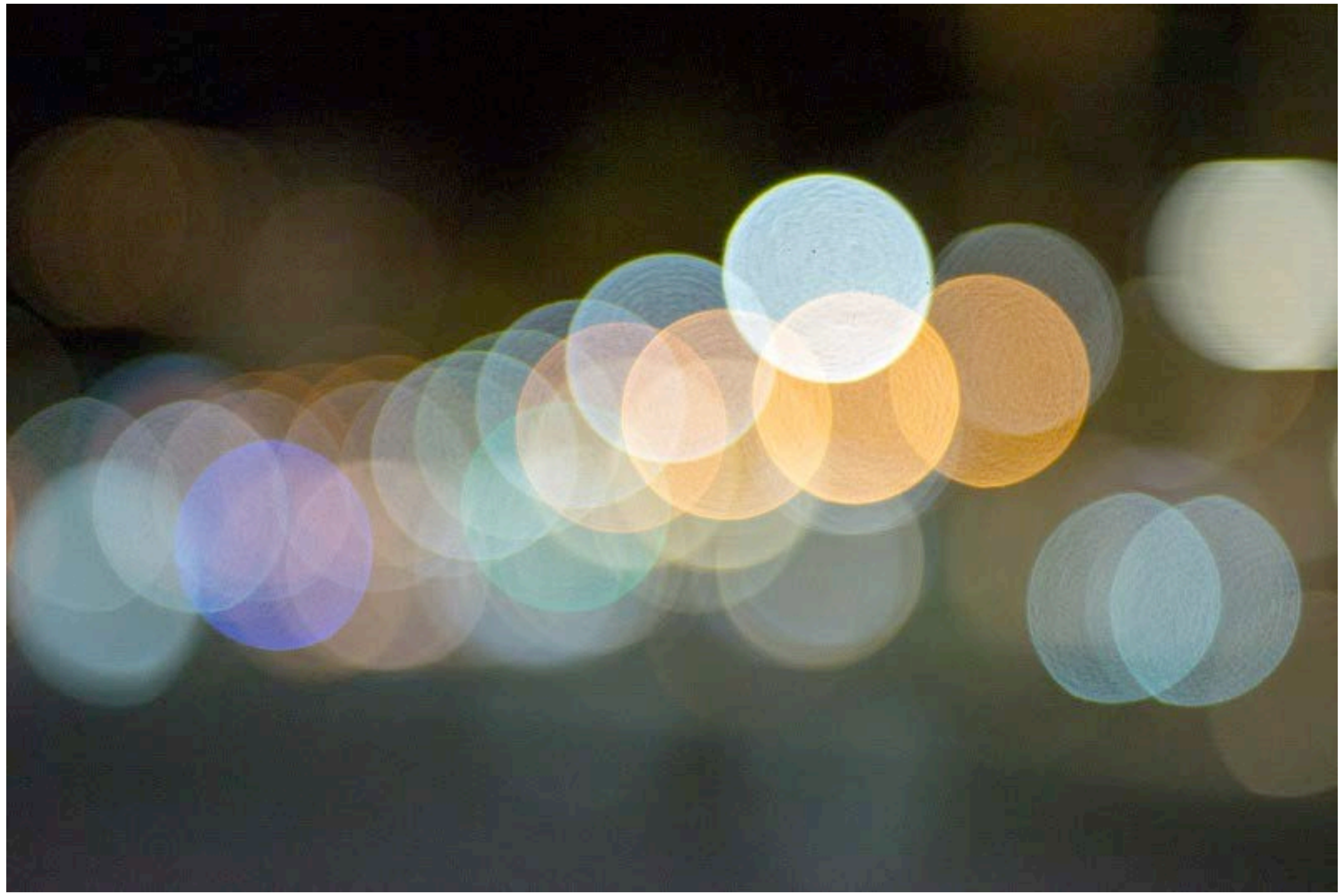
Image

0	0	0	0	0
0	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	0
0	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	0
0	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	0
0	0	0	0	0

Result

Smoothing: **Circular** Kernel





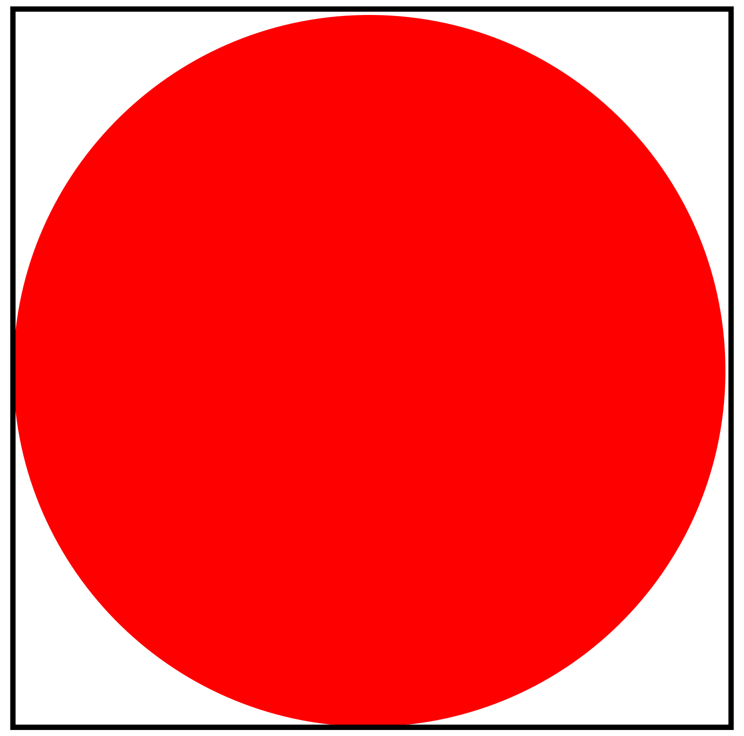
* image credit: https://catlikecoding.com/unity/tutorials/advanced-rendering/depth-of-field/circle-of-confusion/lens-camera.png

Pillbox Filter

Let the radius (i.e., half diameter) of the filter be r

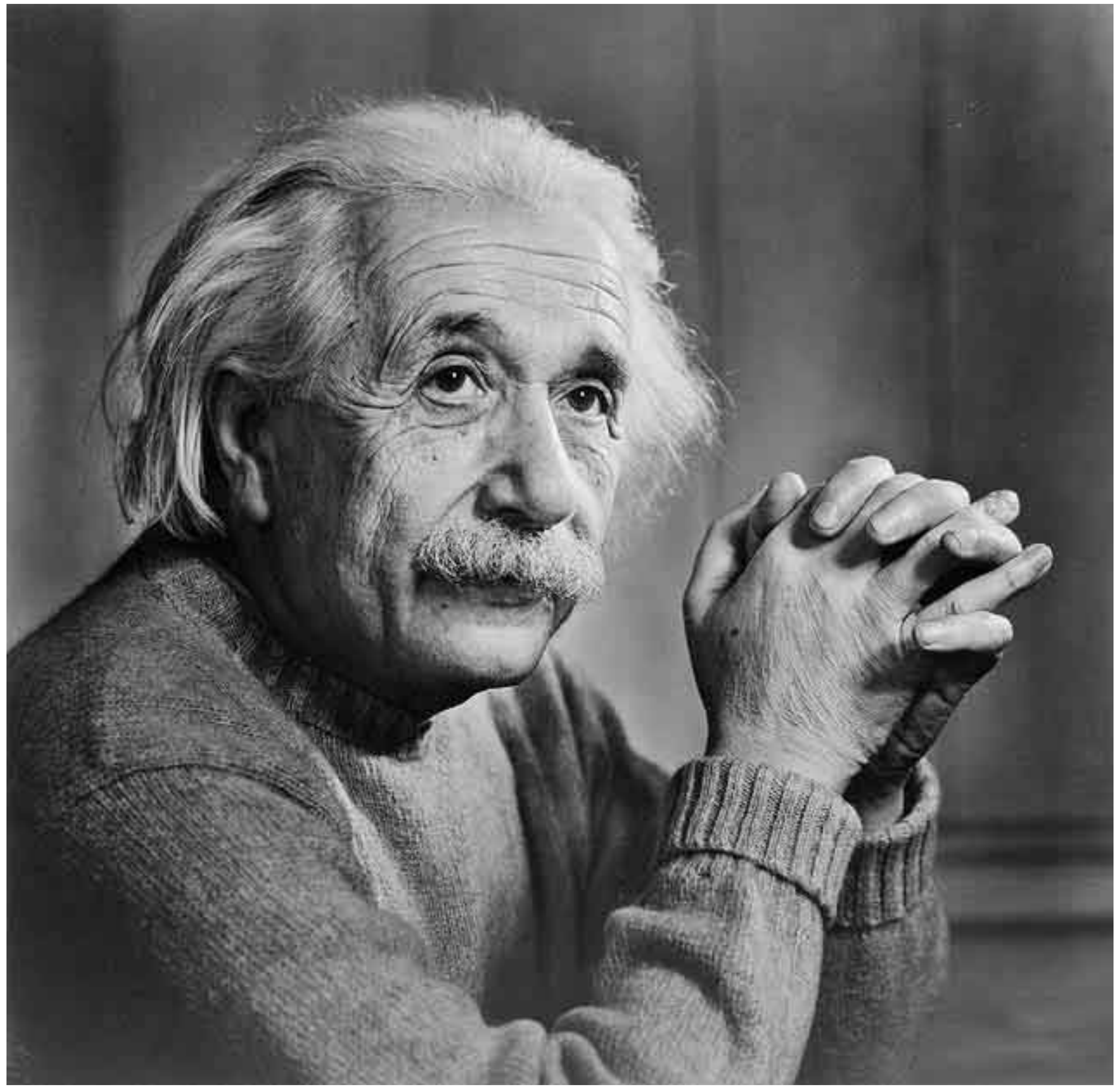
In a contentious domain, a 2D (circular) pillbox filter, $f(x, y)$, is defined as:

$$f(x,y) = \frac{1}{\pi r^2} \begin{cases} 1 & \text{if} \;\; x^2 + y^2 \leq r^2 \\ 0 & \text{otherwise} \end{cases}$$

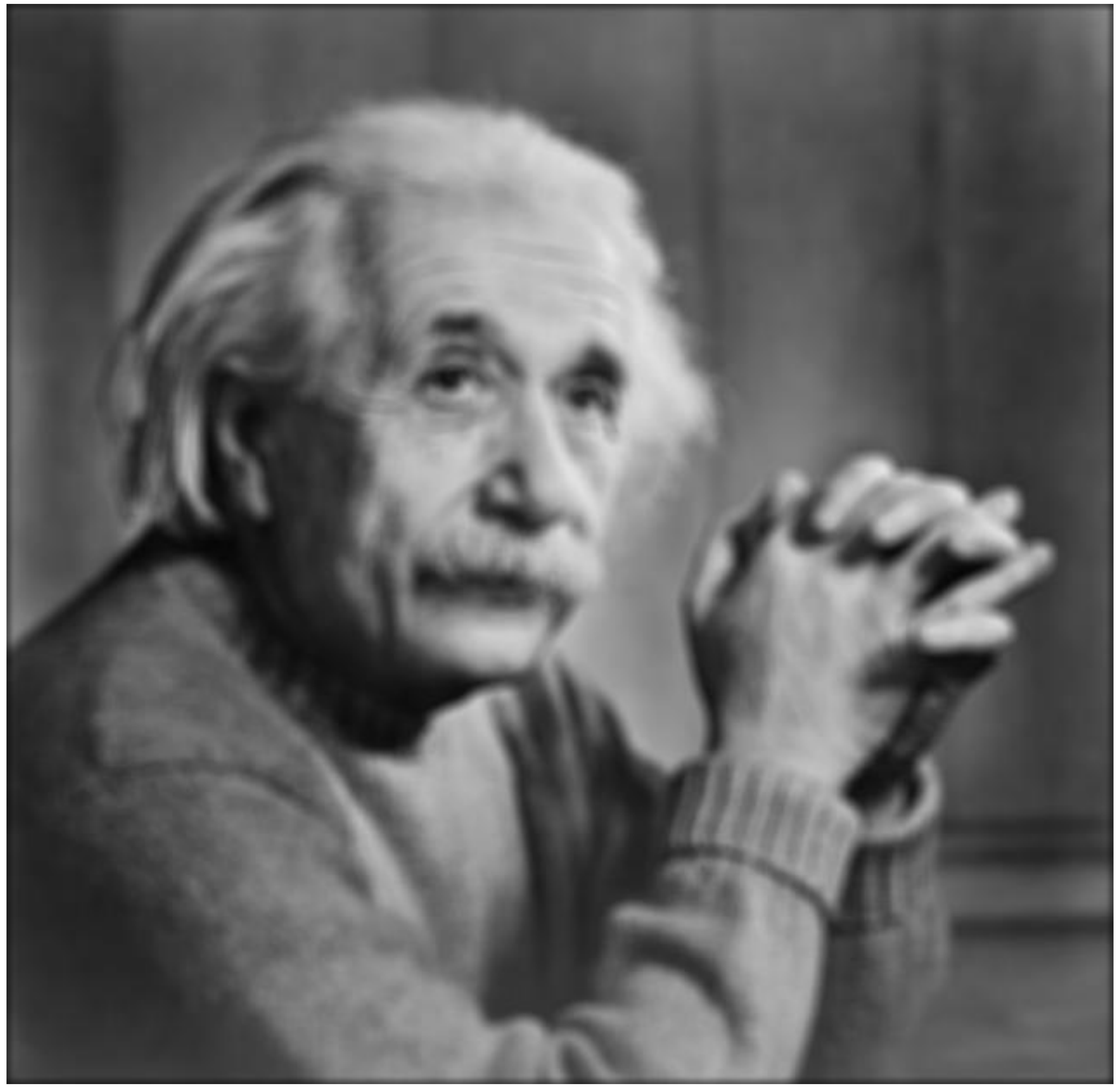


The scaling constant, $\frac{1}{\pi r^2}$, ensures that the area of the filter is one

Pillbox Filter

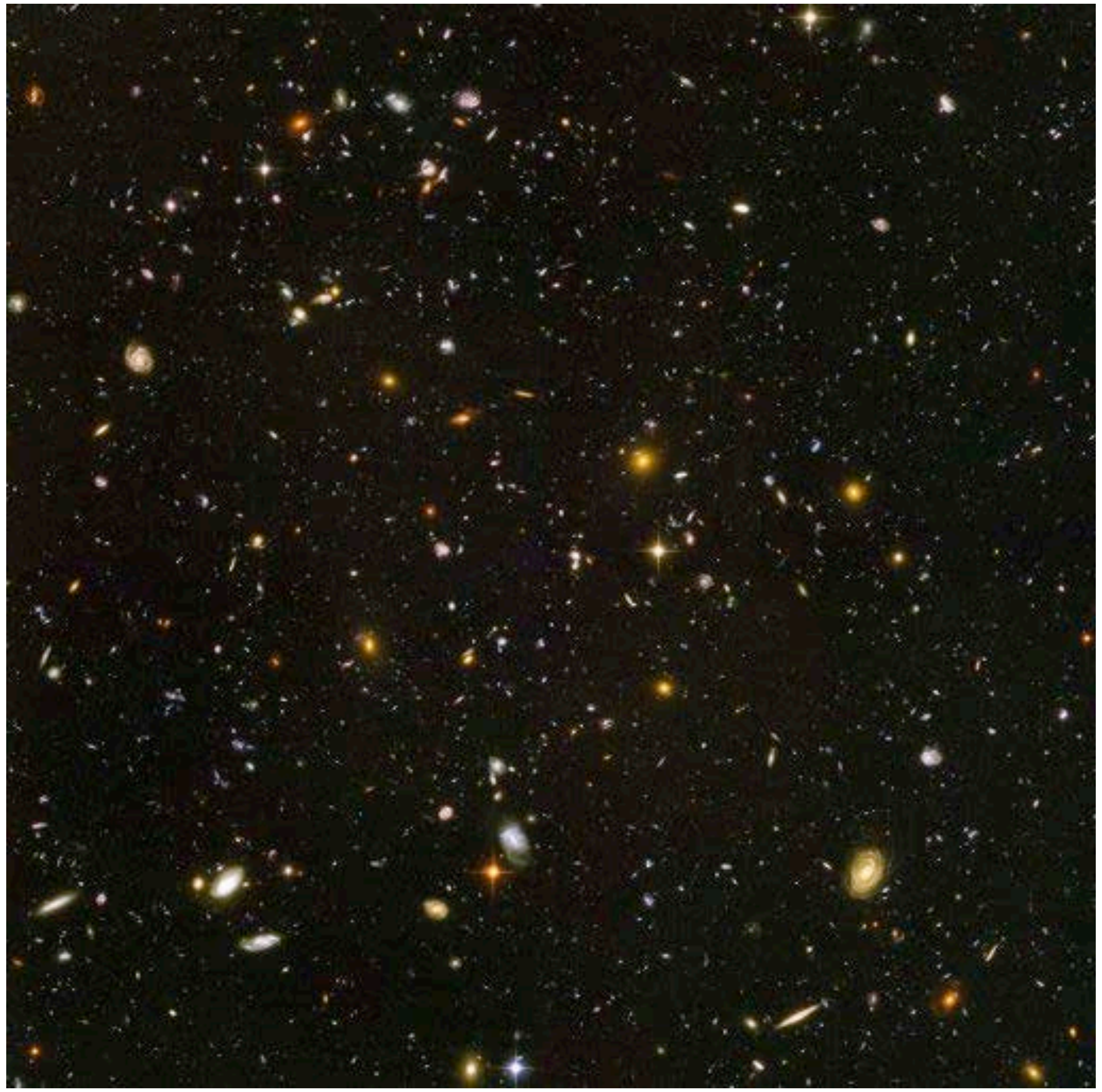


Original

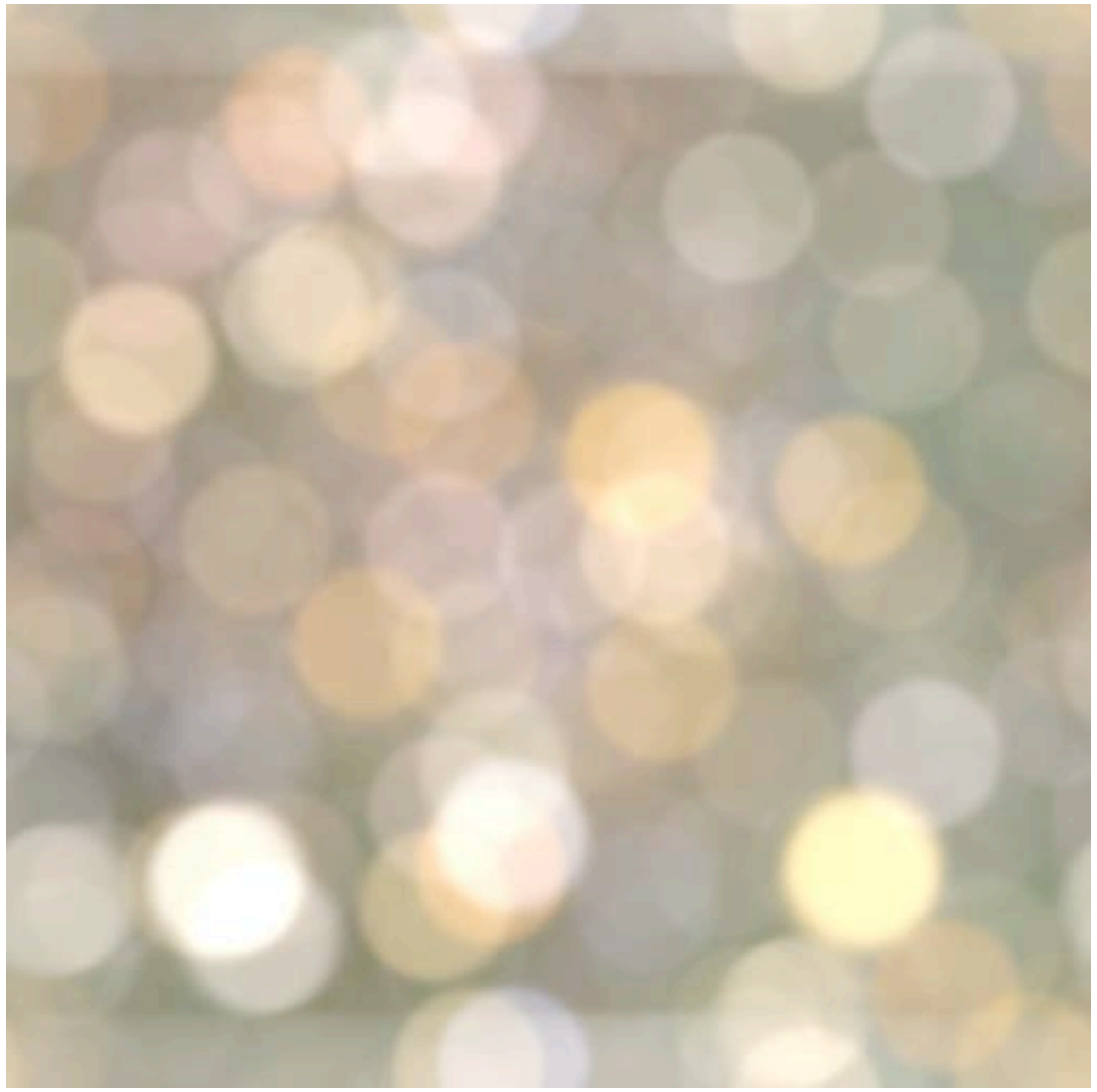


11 x 11 Pillbox

Pillbox Filter



Hubble Deep View



With Circular Blur

Images: yehar.com

Smoothing

Smoothing with a box **doesn’t model lens defocus** well

— Smoothing with a box filter depends on direction

— Image in which the center point is 1 and every other point is 0

Smoothing with a (circular) **pillbox** is a better model for defocus (in geometric optics)

The **Gaussian** is a good general smoothing model

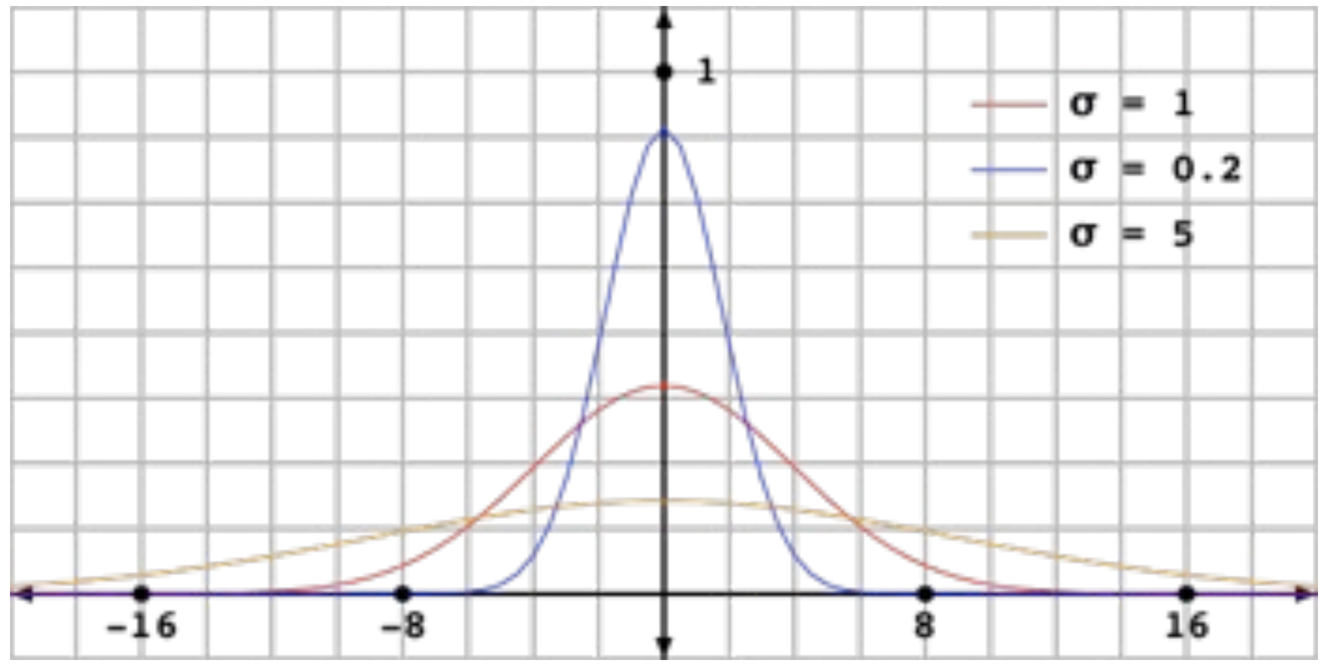
— for phenomena (that are the sum of other small effects)

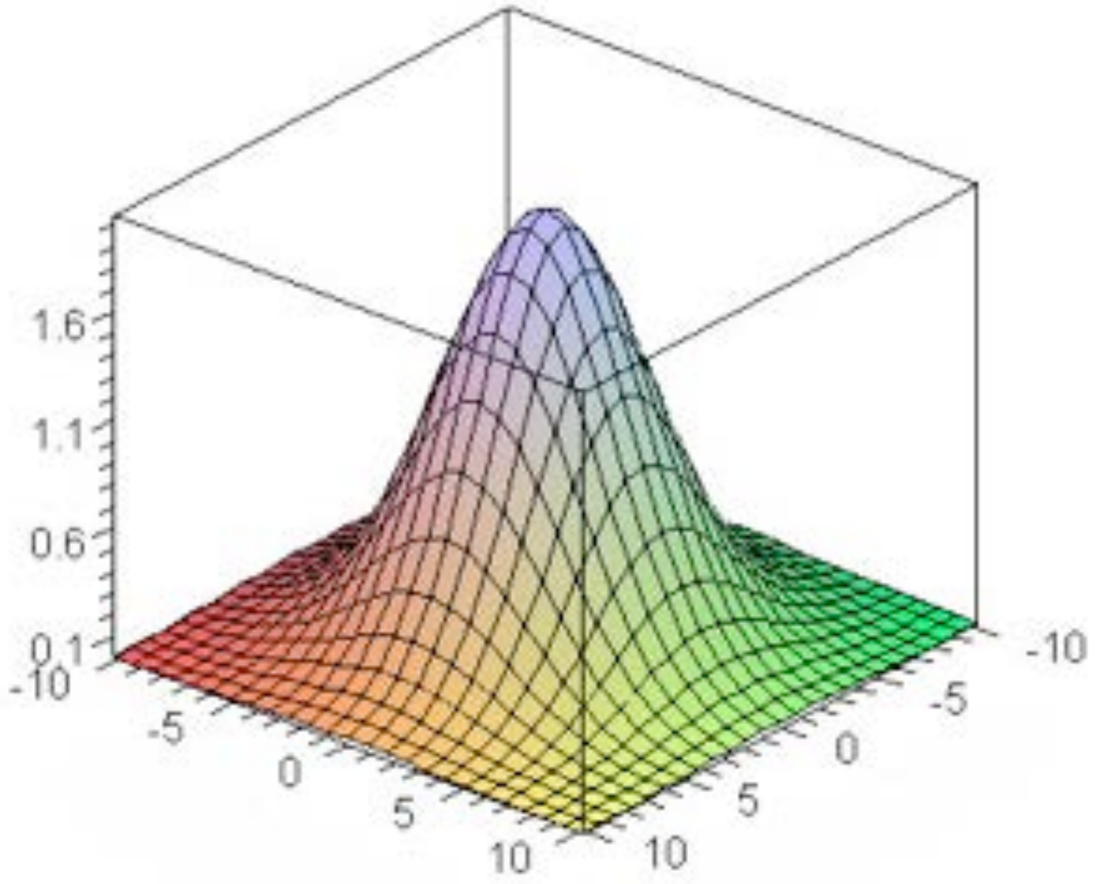
— whenever the Central Limit Theorem applies

Gaussian Blur

Gaussian kernels are often used for smoothing and resizing images

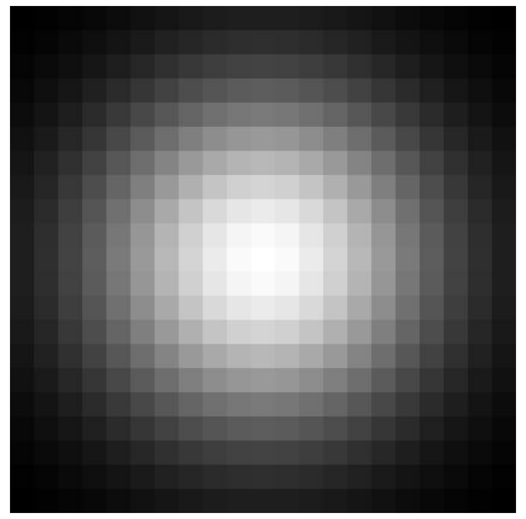
1D





2D

$*$



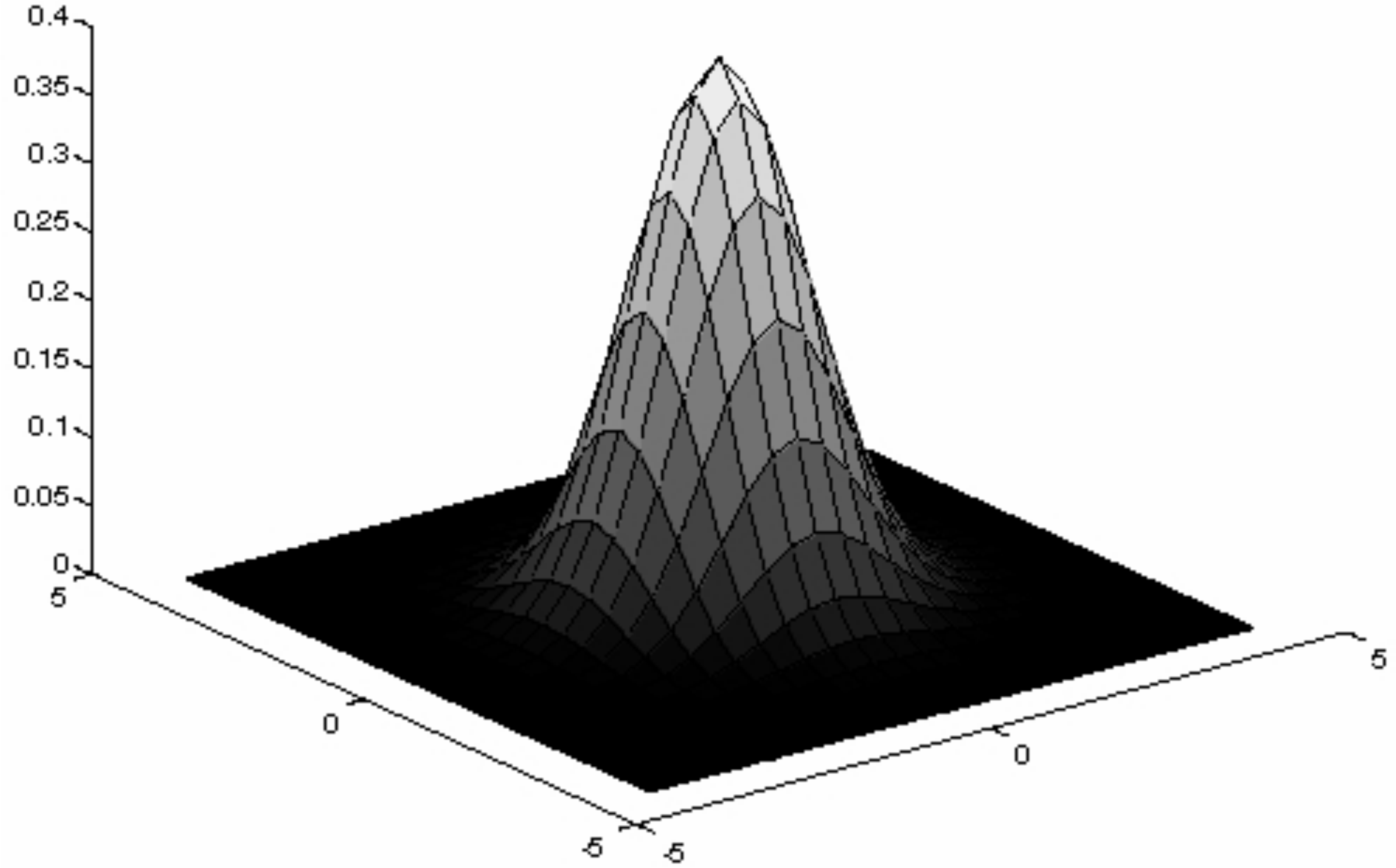
$=$

Smoothing with a **Gaussian**

Idea: Weight contributions of pixels by spatial proximity (nearness)

2D **Gaussian** (continuous case):

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



Forsyth & Ponce (2nd ed.)
Figure 4.2

4.1

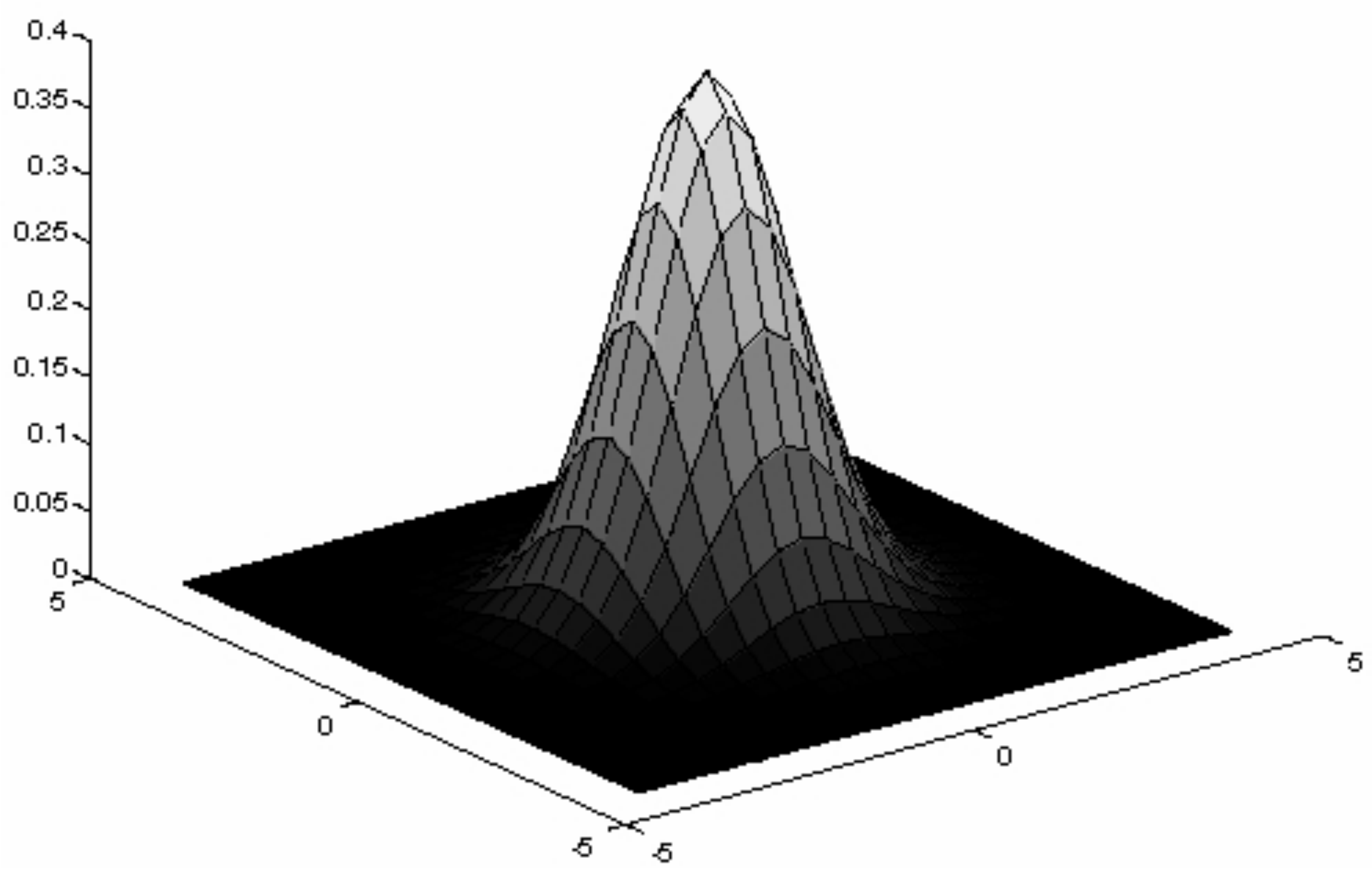
Example 6: Smoothing with a Gaussian

Idea: Weight contributions of pixels by spatial proximity (nearness)

2D **Gaussian** (continuous case):

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

Standard Deviation



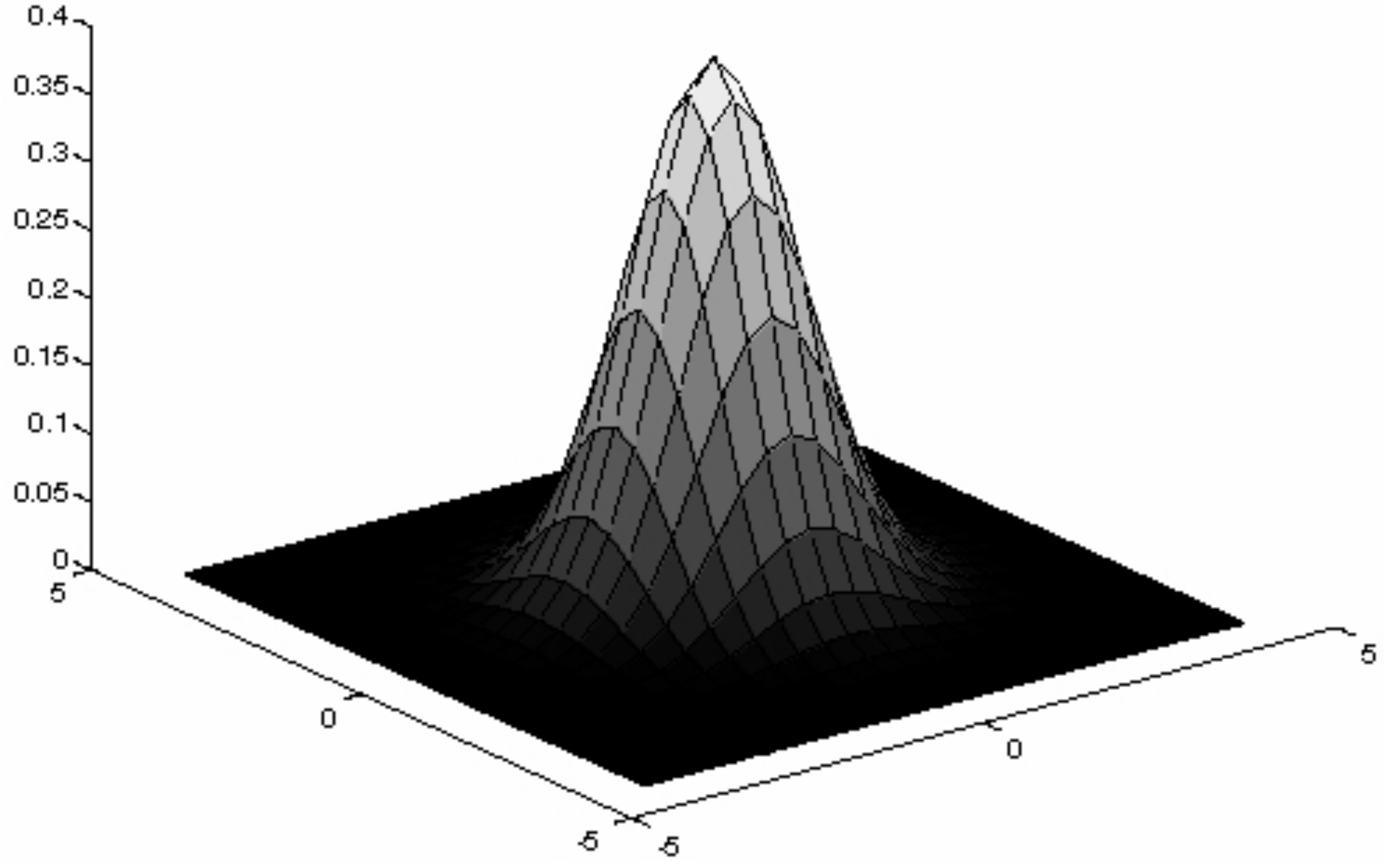
Forsyth & Ponce (2nd ed.)
Figure 4.2

Smoothing with a **Gaussian**

Idea: Weight contributions of pixels by spatial proximity (nearness)

2D **Gaussian** (continuous case):

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



Forsyth & Ponce (2nd ed.)
Figure 4.2

1. Define a continuous **2D function**
2. **Discretize it** by evaluating this function on the discrete pixel positions to obtain a filter

Example 6: Smoothing with a Gaussian

Quantized an truncated **3x3 Gaussian** filter:

$G_\sigma(-1,1)$	$G_\sigma(0,1)$	$G_\sigma(1,1)$
$G_\sigma(-1,0)$	$G_\sigma(0,0)$	$G_\sigma(1,0)$
$G_\sigma(-1,-1)$	$G_\sigma(0,-1)$	$G_\sigma(1,-1)$

Example 6: Smoothing with a Gaussian

Quantized an truncated **3x3 Gaussian** filter:

$G_\sigma(-1,1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{2}{2\sigma^2}}$
$G_\sigma(-1,0) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(0,0) = \frac{1}{2\pi\sigma^2}$	$G_\sigma(1,0) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{1}{2\sigma^2}}$
$G_\sigma(-1,-1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,-1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,-1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{2}{2\sigma^2}}$

Example 6: Smoothing with a Gaussian

Quantized an truncated **3x3 Gaussian** filter:

$G_\sigma(-1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$
$G_\sigma(-1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(0,0) = \frac{1}{2\pi\sigma^2}$	$G_\sigma(1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$
$G_\sigma(-1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$

With $\sigma = 1$:

0.059	0.097	0.059
0.097	0.159	0.097
0.059	0.097	0.059

Example 6: Smoothing with a **Gaussian**

Quantized an truncated **3x3 Gaussian** filter:

$G_\sigma(-1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$
$G_\sigma(-1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(0,0) = \frac{1}{2\pi\sigma^2}$	$G_\sigma(1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$
$G_\sigma(-1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$

With $\sigma = 1$:

0.059	0.097	0.059
0.097	0.159	0.097
0.059	0.097	0.059

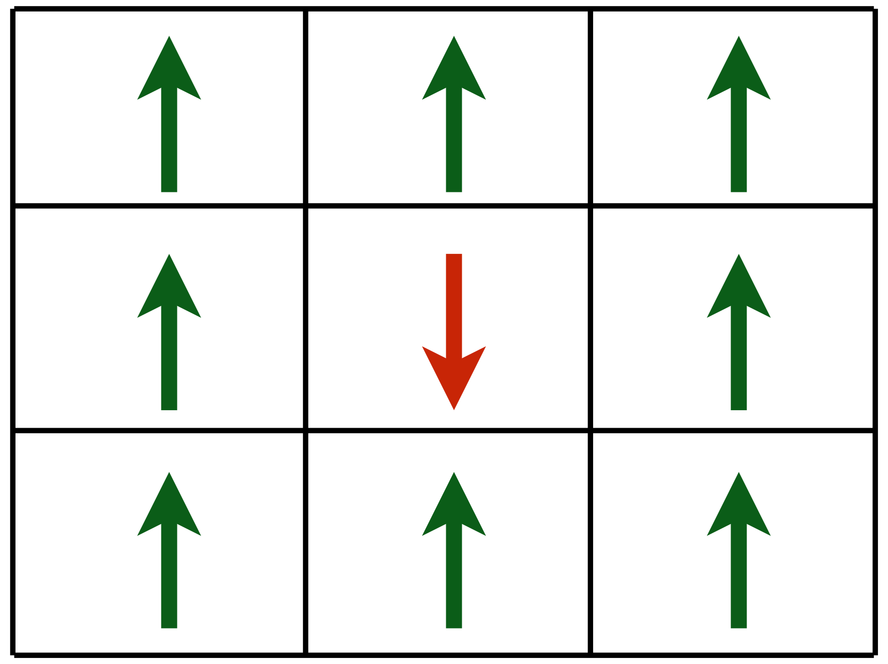
What happens if σ is larger?

Example 6: Smoothing with a **Gaussian**

Quantized an truncated **3x3 Gaussian** filter:

$G_\sigma(-1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$
$G_\sigma(-1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(0,0) = \frac{1}{2\pi\sigma^2}$	$G_\sigma(1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$
$G_\sigma(-1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$

With $\sigma = 1$:



What happens if σ is larger?

— **More** blur

Example 6: Smoothing with a **Gaussian**

Quantized an truncated **3x3 Gaussian** filter:

$G_\sigma(-1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$
$G_\sigma(-1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(0,0) = \frac{1}{2\pi\sigma^2}$	$G_\sigma(1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$
$G_\sigma(-1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$

With $\sigma = 1$:

0.059	0.097	0.059
0.097	0.159	0.097
0.059	0.097	0.059

What happens if σ is larger?

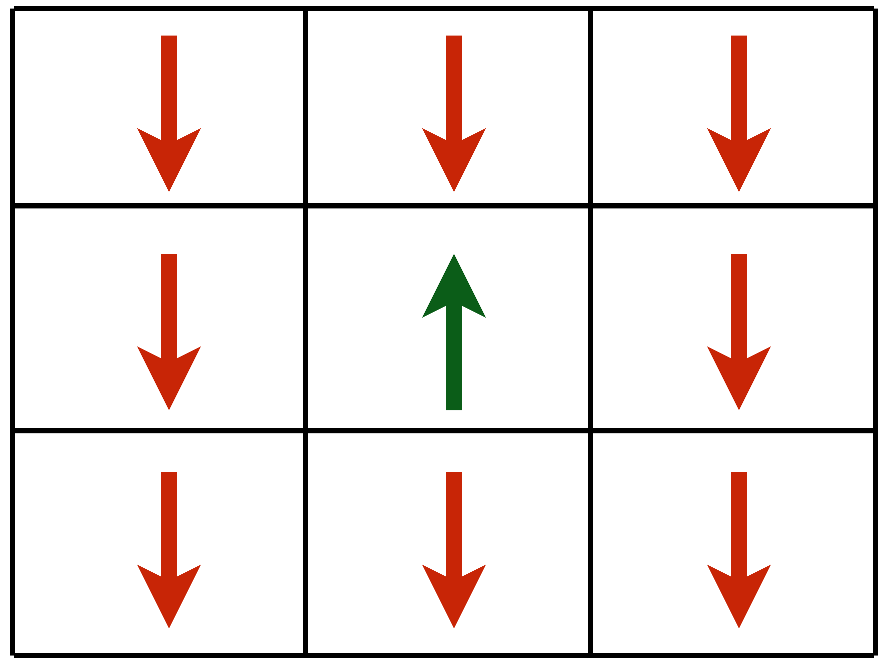
What happens if σ is smaller?

Example 6: Smoothing with a **Gaussian**

Quantized an truncated **3x3 Gaussian** filter:

$G_\sigma(-1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$
$G_\sigma(-1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(0,0) = \frac{1}{2\pi\sigma^2}$	$G_\sigma(1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$
$G_\sigma(-1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$

With $\sigma = 1$:

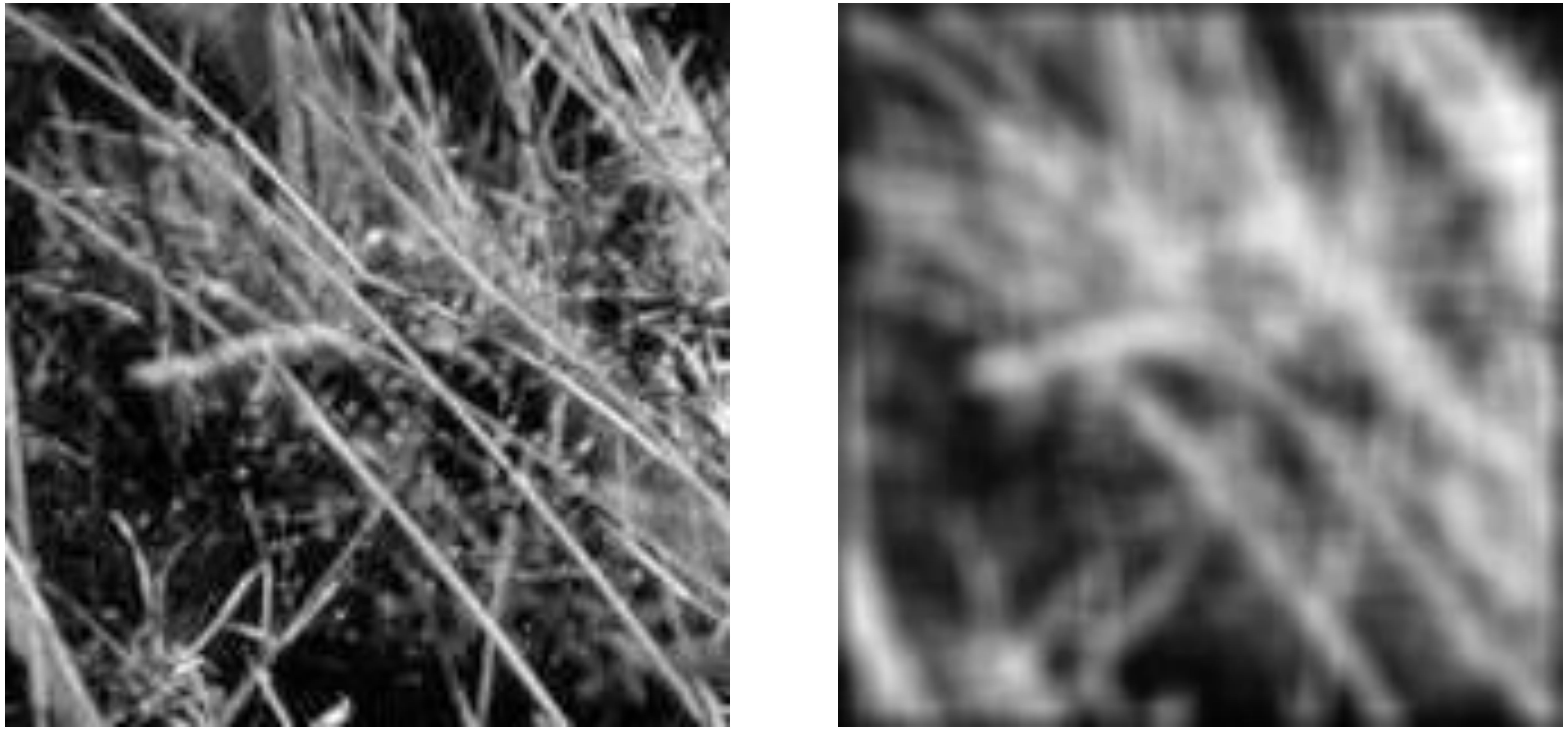


What happens if σ is larger?

What happens if σ is smaller?

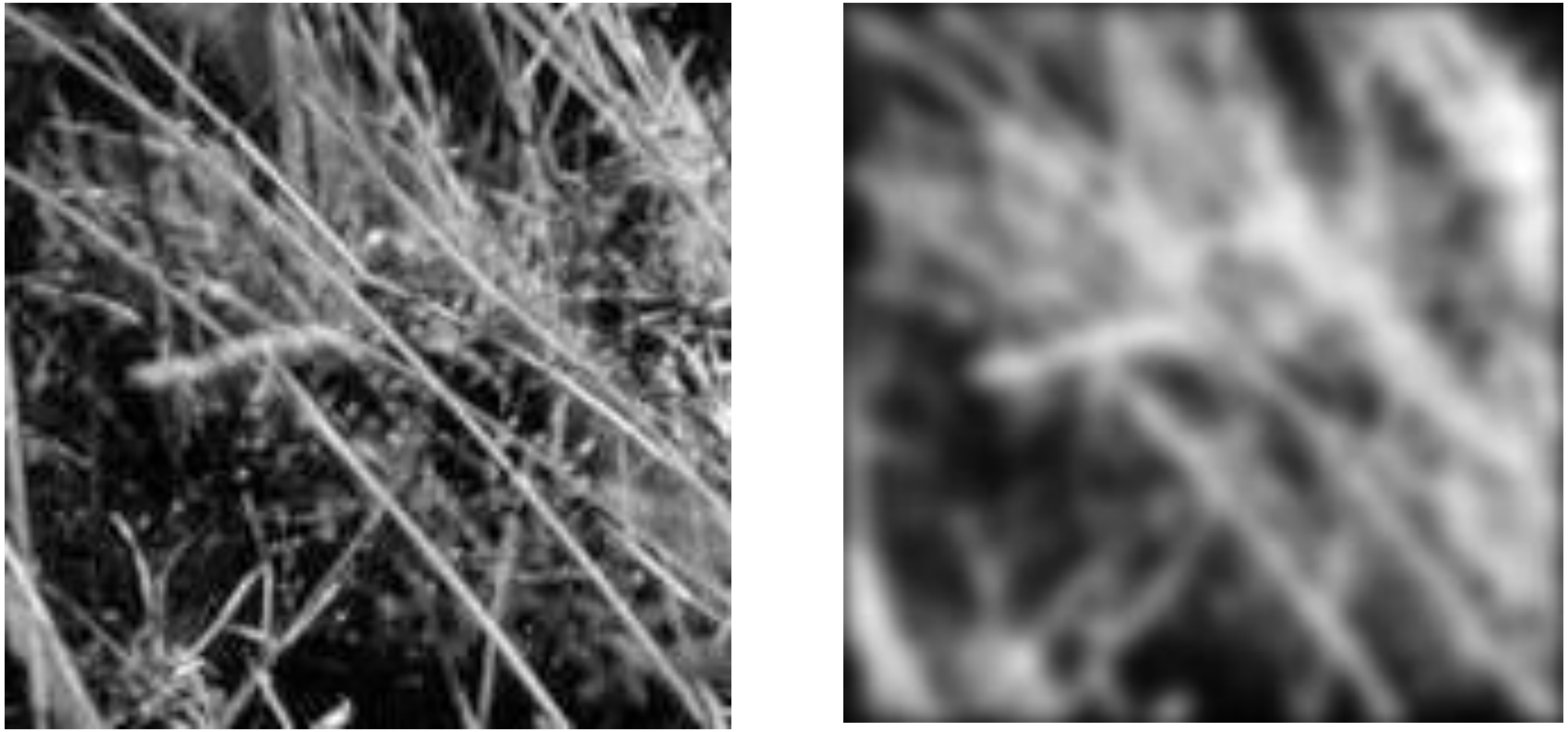
— **Less** blur

Smoothing with a **Box Filter**



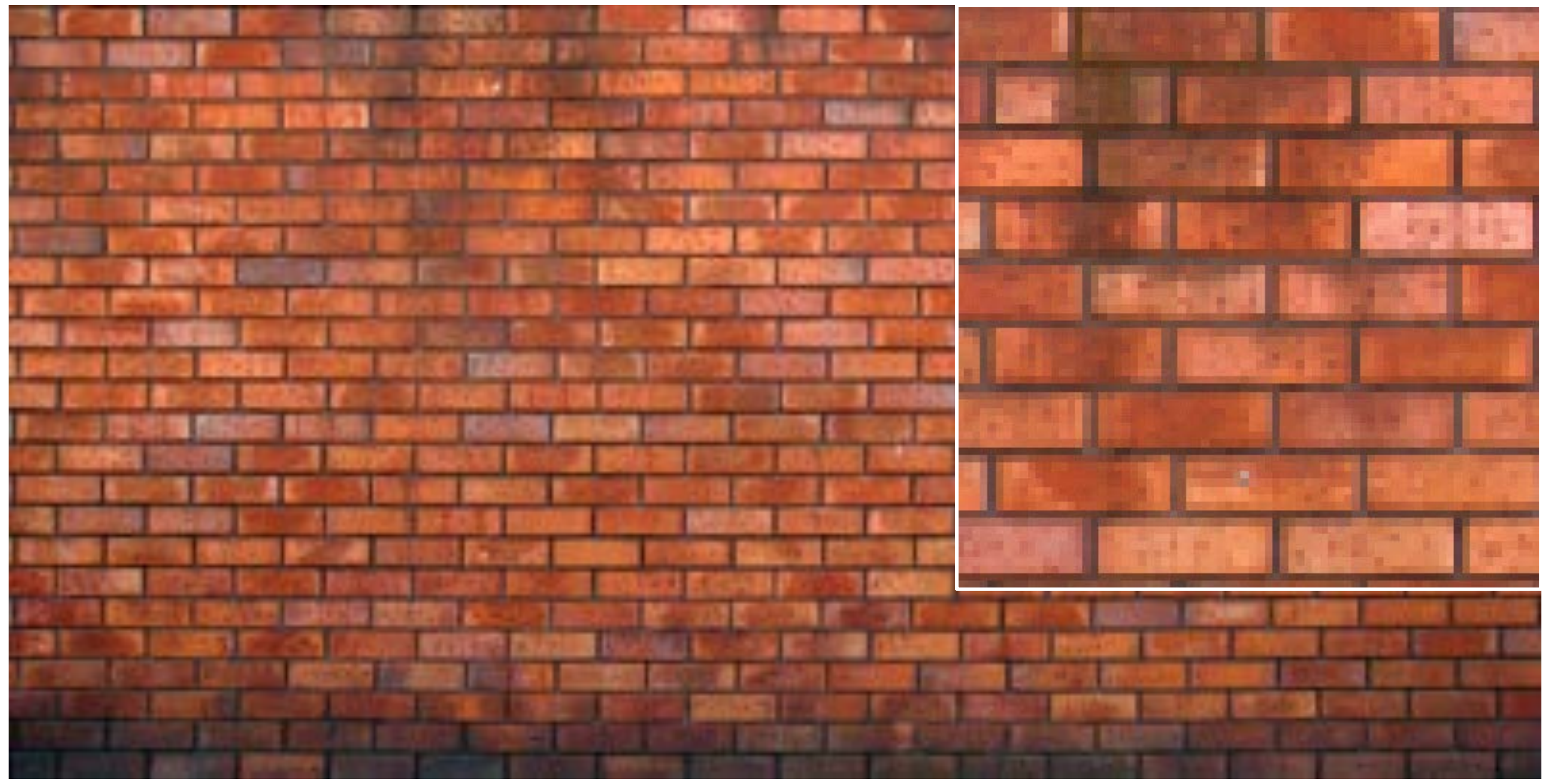
Forsyth & Ponce (2nd ed.) Figure 4.1 (left and middle)

Smoothing with a **Gaussian**

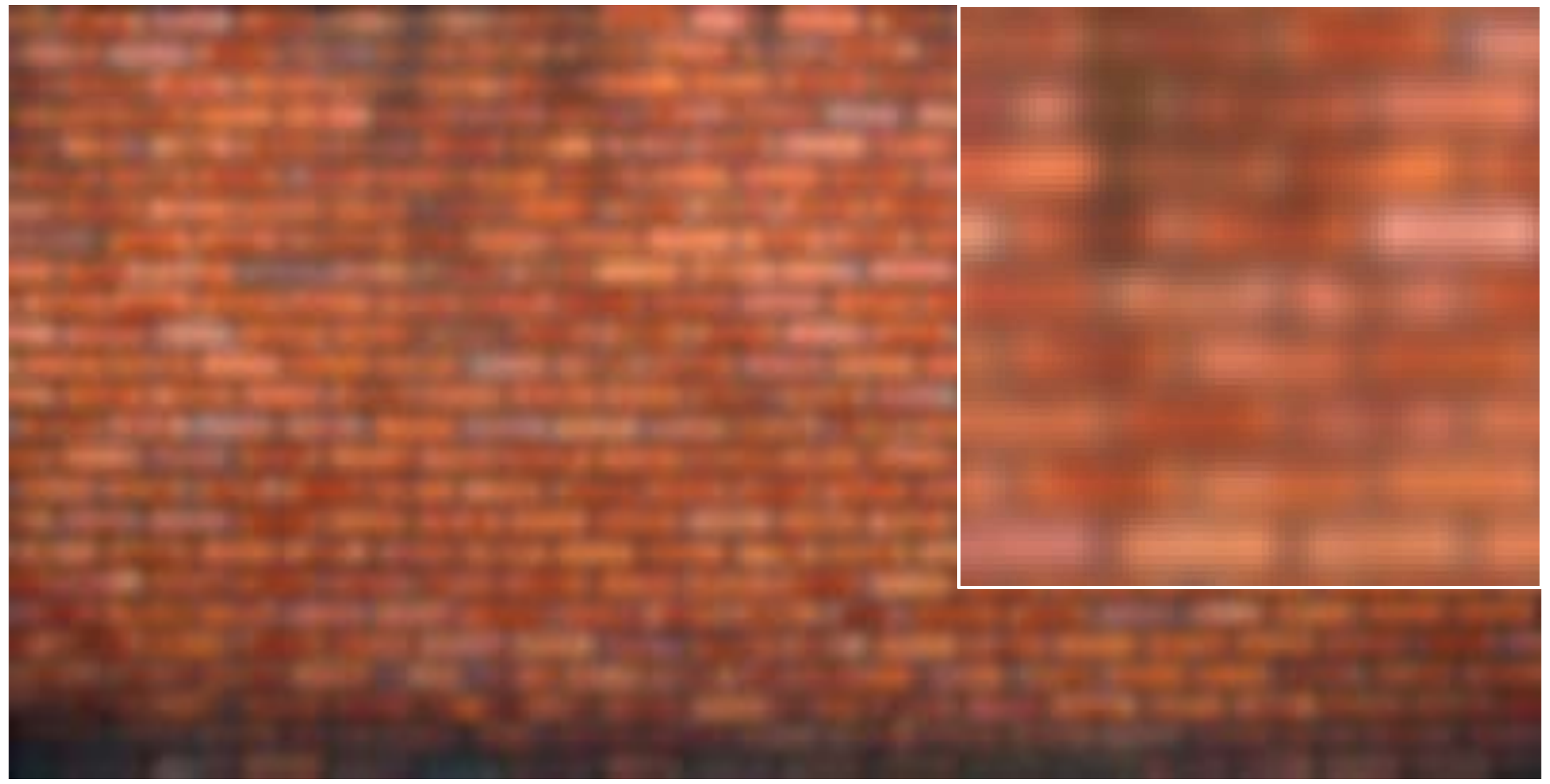


Forsyth & Ponce (2nd ed.) Figure 4.1 (left and right)

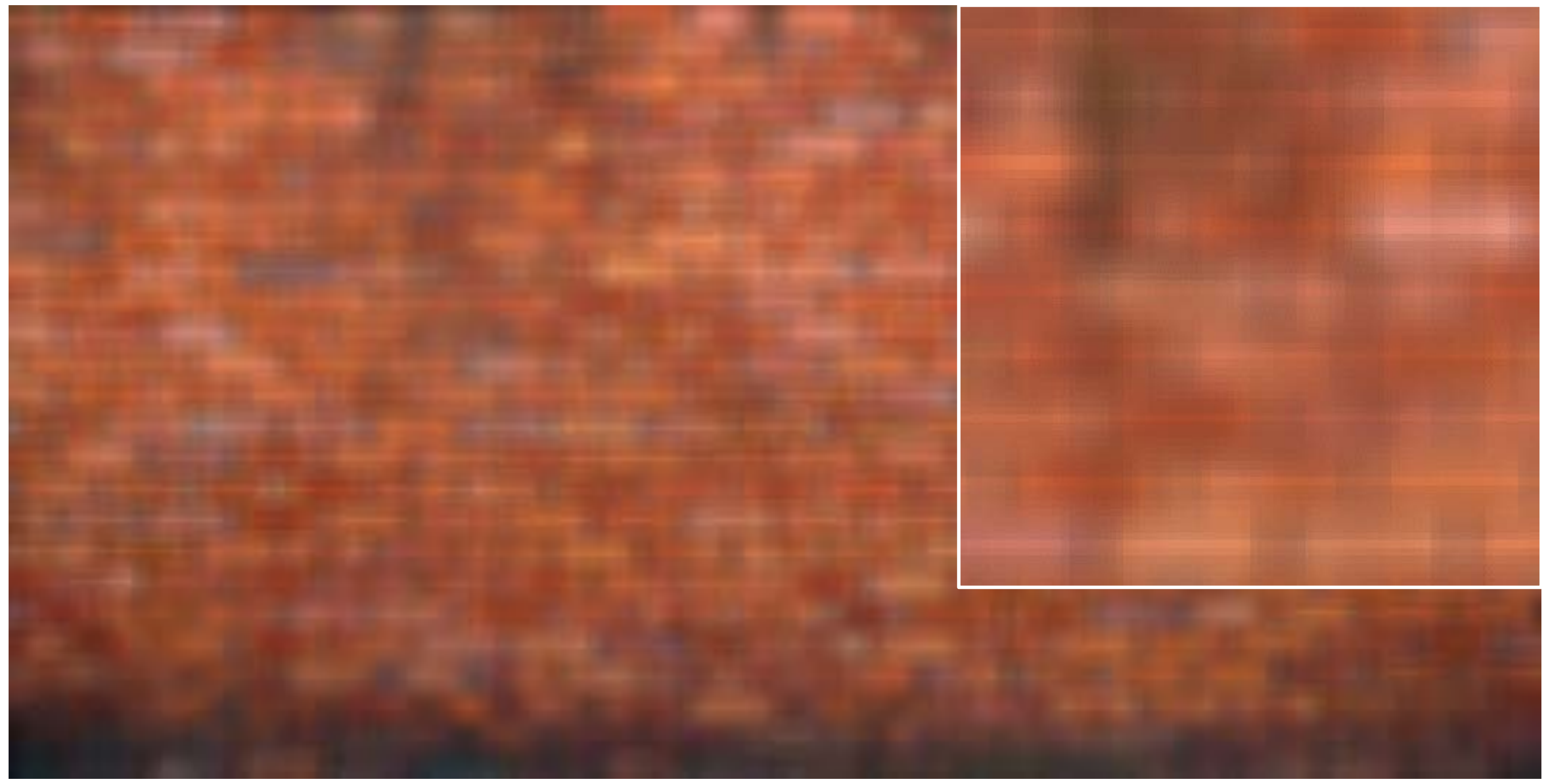
Box vs. **Gaussian** Filter



original



7x7 Gaussian



7x7 box

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Fun: How to get shadow effect?

University of
British
Columbia

Adopted from: Ioannis (Yannis) Gkioulekas (CMU)

Fun: How to get shadow effect?

University of
British
Columbia

Blur with a Gaussian kernel, then compose the blurred image with the original
(with some offset)

Adopted from: Ioannis (Yannis) Gkioulekas (CMU)

Example 6: Smoothing with a Gaussian

Quantized an truncated **3x3 Gaussian** filter:

$G_\sigma(-1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$
$G_\sigma(-1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(0,0) = \frac{1}{2\pi\sigma^2}$	$G_\sigma(1,0) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$
$G_\sigma(-1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,-1) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{2}{2\sigma^2}}$

With $\sigma = 1$:

0.059	0.097	0.059
0.097	0.159	0.097
0.059	0.097	0.059

What is the problem with this filter?

Example 6: Smoothing with a Gaussian

Quantized an truncated **3x3 Gaussian** filter:

$G_\sigma(-1,1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{2}{2\sigma^2}}$
$G_\sigma(-1,0) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(0,0) = \frac{1}{2\pi\sigma^2}$	$G_\sigma(1,0) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{1}{2\sigma^2}}$
$G_\sigma(-1,-1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{2}{2\sigma^2}}$	$G_\sigma(0,-1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{1}{2\sigma^2}}$	$G_\sigma(1,-1) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{2}{2\sigma^2}}$

With $\sigma = 1$:

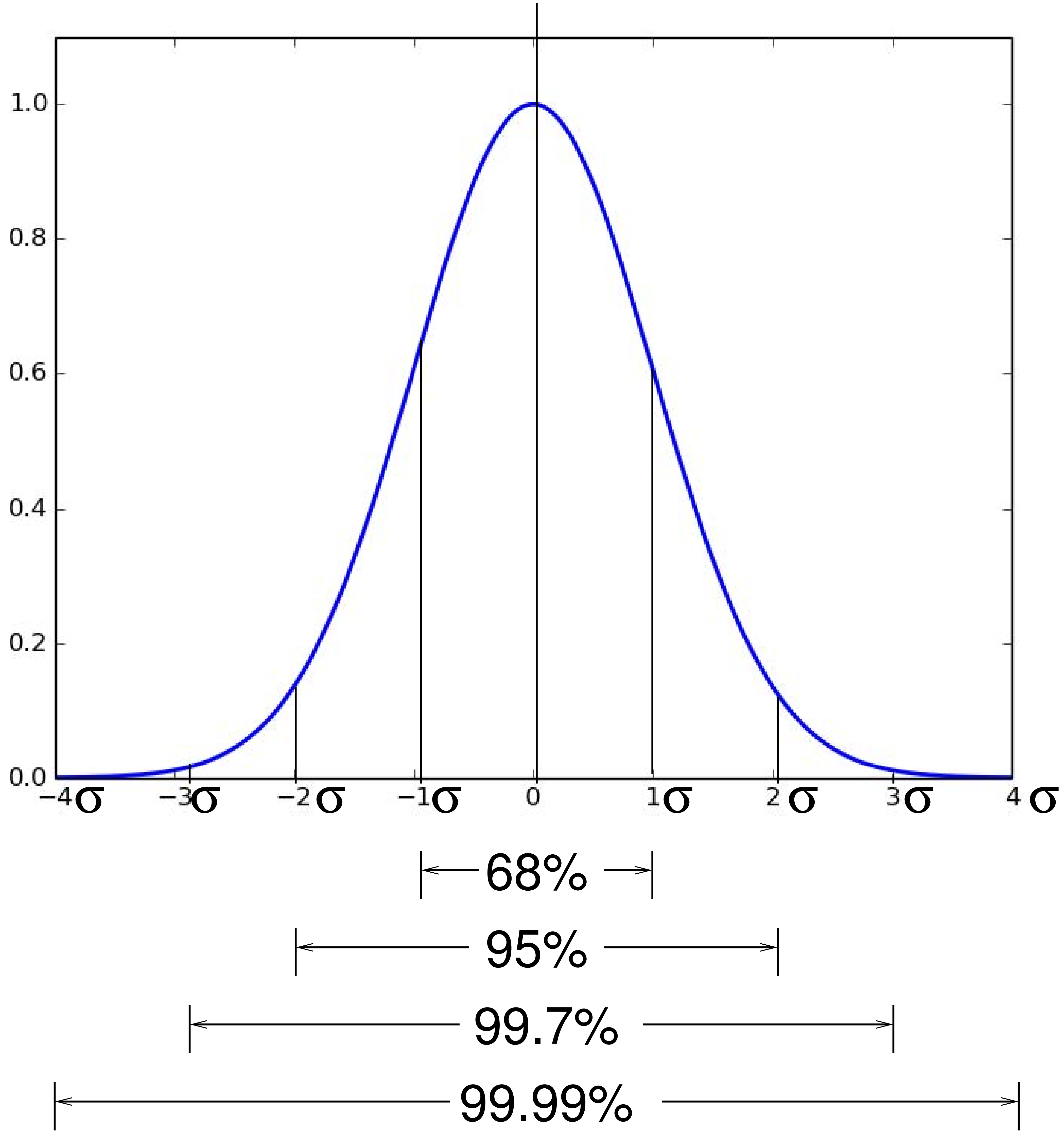
0.059	0.097	0.059
0.097	0.159	0.097
0.059	0.097	0.059

What is the problem with this filter?

does not sum to 1

truncated too much

Gaussian: Area Under the Curve



Example 6: Smoothing with a Gaussian

With $\sigma = 1$:

0.059	0.097	0.059
0.097	0.159	0.097
0.059	0.097	0.059

Better version of the Gaussian filter:

— sums to 1 (normalized)

— captures $\pm 2\sigma$

$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

A good guideline for the Gaussian filter is to capture $\pm 3\sigma$, for $\sigma = 1$ => 7x7 filter

Smoothing **Summary**

Smoothing with a box **doesn't model lens defocus** well

— Smoothing with a box filter depends on direction

— Point spread function is a box

Smoothing with a (circular) **pillbox** is a better model for defocus (in geometric optics)

The **Gaussian** is a good general smoothing model

— for phenomena (that are the sum of other small effects)

— whenever the Central Limit Theorem applies (avg of many independent rvs → normal dist)

Lets talk about **efficiency**

Efficient Implementation: **Separability**

A 2D function of x and y is **separable** if it can be written as the product of two functions, one a function only of x and the other a function only of y

Both the **2D box filter** and the **2D Gaussian filter** are **separable**

Both can be implemented as two 1D convolutions:
— First, convolve each row with a 1D filter
— Then, convolve each column with a 1D filter
— Aside: or vice versa

The **2D Gaussian** is the only (non trivial) 2D function that is both separable and rotationally invariant.

Separability: Box Filter Example

Standard (3x3)

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

$$F(X,Y) = F(X)F(Y)$$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

	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	30	10	10	0	0	0	0	

Separability: Box Filter Example

Standard (3x3)

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

$$F(X,Y) = F(X)F(Y)$$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

	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	30	10	10	0	0	0	0	

Separable

$I(X,Y)$

image

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

$F(X)$

filter

$\frac{1}{3}$

1	1	1

	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	
	0	30	60	90	90	90	60	30	
	0	30	60	90	90	90	60	30	
	0	30	30	60	60	90	60	30	
	0	30	60	90	90	90	60	30	
	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	
	30	30	30	30	0	0	0	0	
	0	0	0	0	0	0	0	0	

Separability: Box Filter Example

Standard (3x3)

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

$$F(X,Y) = F(X)F(Y)$$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

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	30	10	10	0	0	0	0	

Separable

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

$F(X)$

filter

$\frac{1}{3}$

1	1	1

	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	
	0	30	60	90	90	90	60	30	
	0	30	60	90	90	90	60	30	
	0	30	30	60	60	90	60	30	
	0	30	60	90	90	90	60	30	
	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	
	30	30	30	30	0	0	0	0	
	0	0	0	0	0	0	0	0	

$F(Y)$

filter

$\frac{1}{3}$

1
1
1

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	30	10	10	0	0	0	0	

Separability: Proof

Convolution with $F(X,Y) = F(X)F(Y)$ can be performed as 2 x 1D convolutions

4.2

Separability: How do you know if filter is separable?

If a 2D filter can be expressed as an outer product of two 1D filters

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} = \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} \odot \frac{1}{3}\begin{bmatrix}1\\ 1\\ 1\end{bmatrix}$$

Efficient Implementation: **Separability**

For example, recall the 2D **Gaussian**:

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

The 2D Gaussian can be expressed as a product of two functions, one a function of x and another a function of y

Efficient Implementation: **Separability**

For example, recall the 2D **Gaussian**:

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

$$= \underbrace{\left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}}\right)}_{\text{function of x}} \underbrace{\left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}}\right)}_{\text{function of y}}$$

The 2D Gaussian can be expressed as a product of two functions, one a function of x and another a function of y

Efficient Implementation: **Separability**

For example, recall the 2D **Gaussian**:

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

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

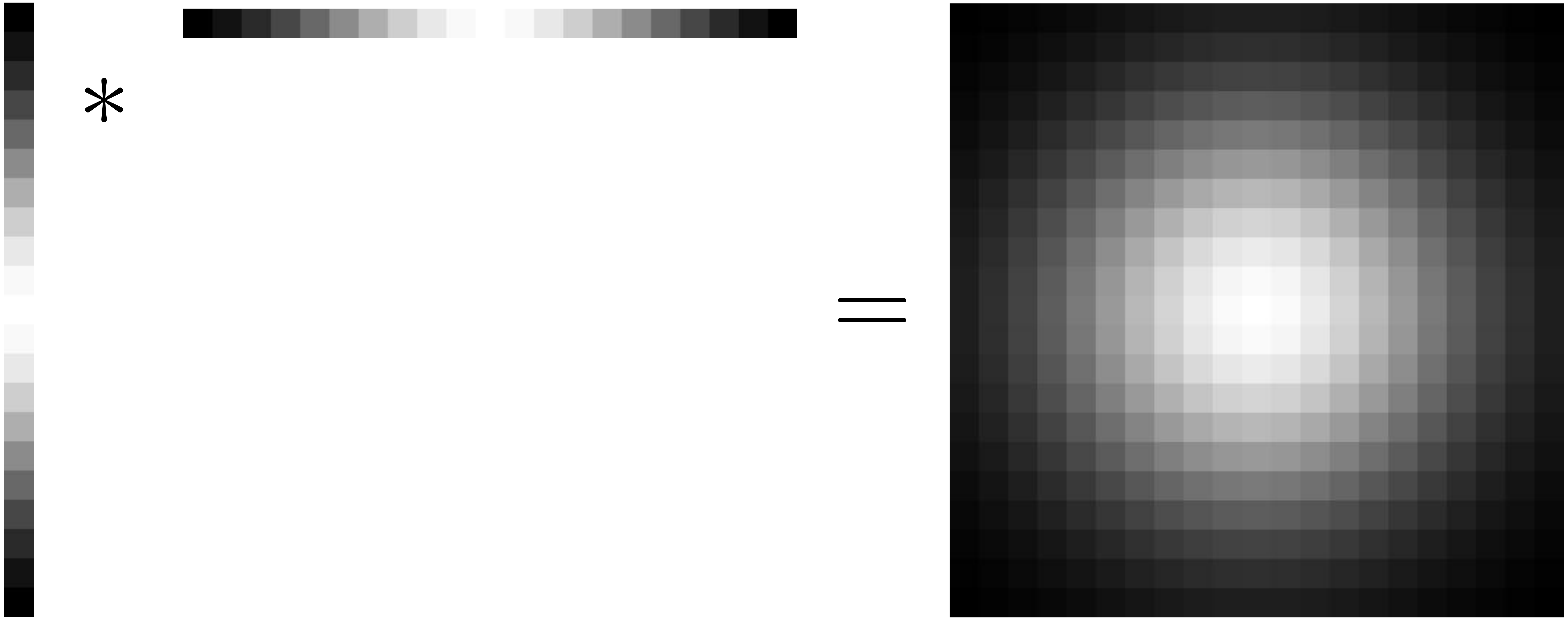
function of x | function of y

The 2D Gaussian can be expressed as a product of two functions, one a function of x and another a function of y

In this case the two functions are (identical) 1D Gaussians

Gaussian Blur

- 2D Gaussian filter can be thought of as an **outer product** or **convolution** of row and column filters



Example: Separable Gaussian Filter

$$\frac{1}{16}\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix} \otimes \frac{1}{16}\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix} = \frac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\4 & 16 & 24 & 16 & 4\\6 & 24 & 36 & 24 & 6\\4 & 16 & 24 & 16 & 4\\1 & 4 & 6 & 4 & 1\end{bmatrix}$$

Example: Separable Gaussian Filter

$$
\frac{1}{16}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \otimes \frac{1}{16}\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}
$$

Efficient Implementation: **Separability**

4.3

Efficient Implementation: **Separability**

Naive implementation of 2D **Gaussian**:

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

Efficient Implementation: **Separability**

Naive implementation of 2D **Gaussian**:

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

Separable 2D **Gaussian**:

Efficient Implementation: **Separability**

Naive implementation of 2D **Gaussian**:

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

Separable 2D **Gaussian**:

At each pixel, (X, Y), there are	$2m$	multiplications
There are	$n \times n$	pixels in (X, Y)
Total:	$2m \times n^2$	multiplications

Separable Filtering

2D Gaussian blur by horizontal/vertical blur

horizontal

vertical

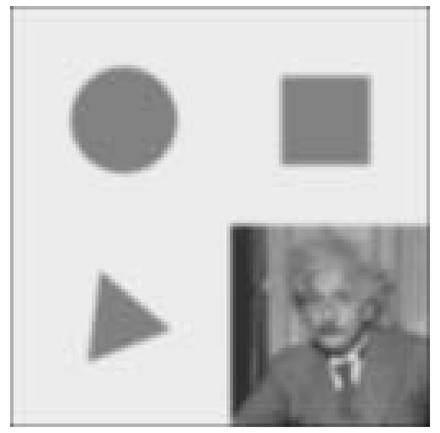
vertical

horizontal

Separable Filtering

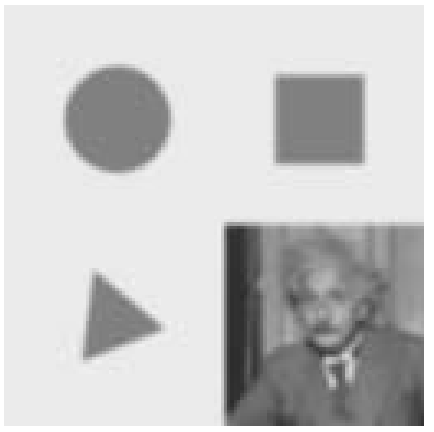
Several useful filters can be applied as independent row and column operations

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \qquad \frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$



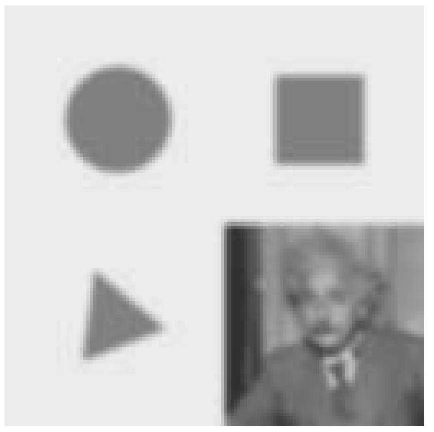
(a) box, $K = 5$

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \qquad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$



(b) bilinear

$$\frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \qquad \frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$



(c) “Gaussian”

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array}$$

(d) Sobel

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array} \qquad \frac{1}{2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline \end{array}$$

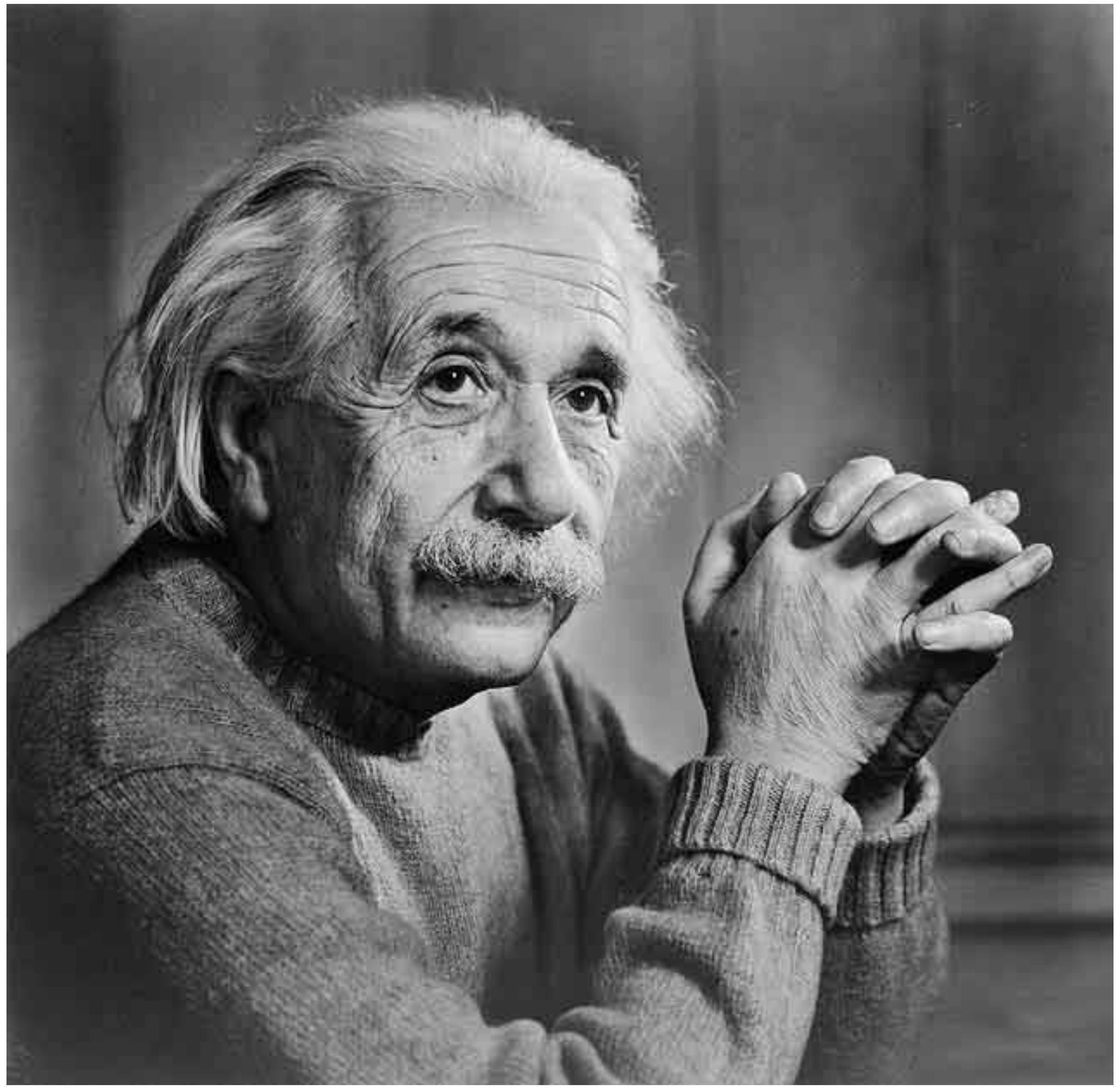
(e) corner

Example 7: Smoothing with a Pillbox

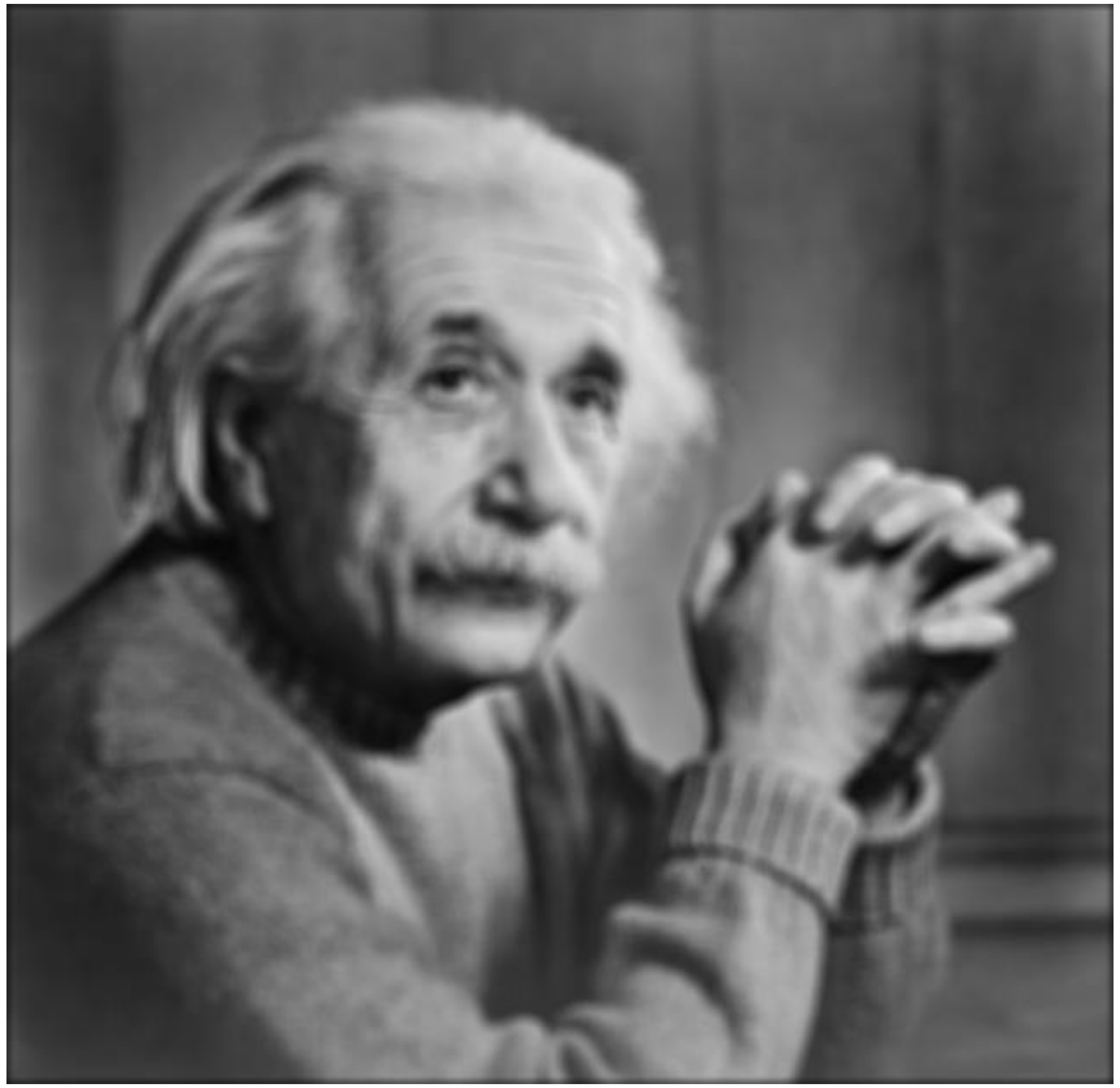
The 2D Gaussian is the only (non trivial) 2D function that is both **separable** and **rotationally invariant**.

A **2D pillbox** is rotationally invariant but **not separable** → harder to implement efficiently

Example 7: Smoothing with a Pillbox



Original



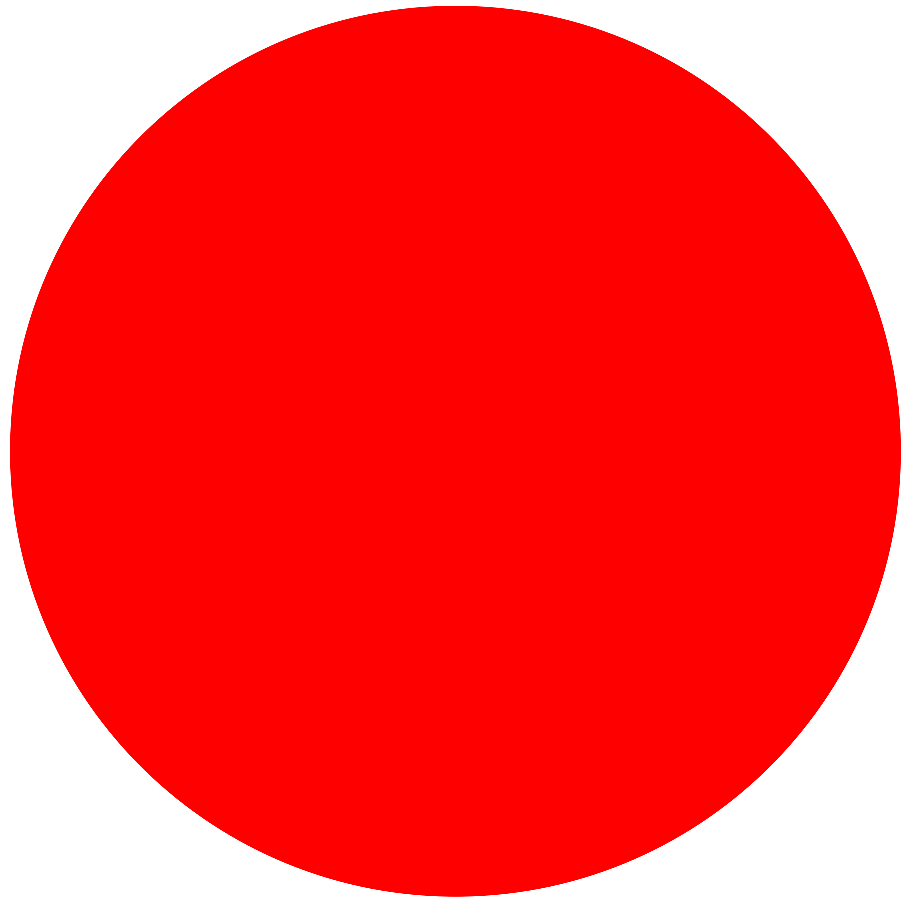
11 x 11 Pillbox

Low-pass Filtering = “Smoothing”

Box Filter

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

Pillbox Filter



Gaussian Filter

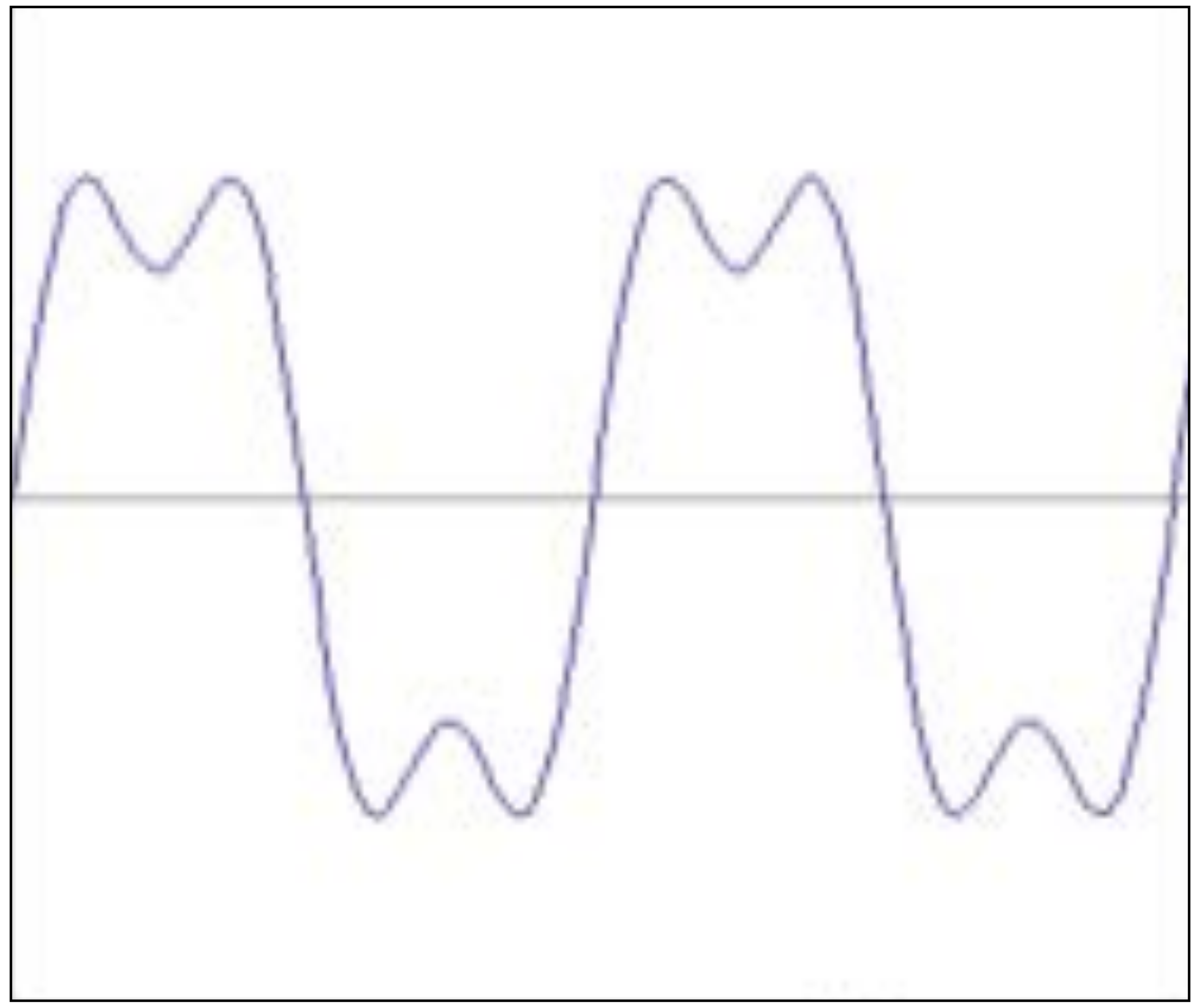
$$\frac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & 36 & 24 & 6\\ 4 & 16 & 24 & 16 & 4\\ 1 & 4 & 6 & 4 & 1\end{bmatrix}$$

All of these filters are **Low-pass Filters**

Low-pass filter: Low pass filter filters out all of the high frequency content of the image, only low frequencies remain

Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

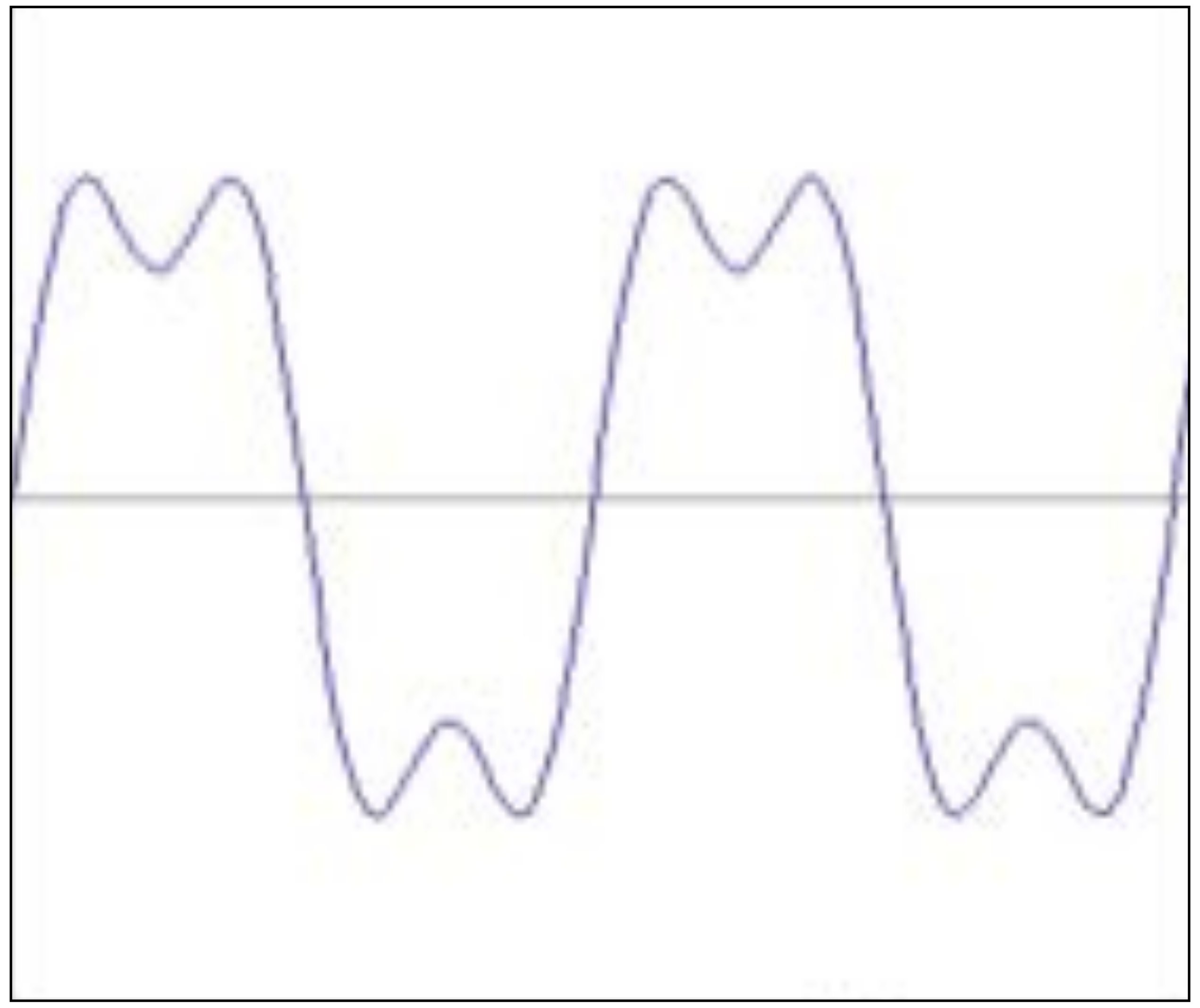


= ? + ?

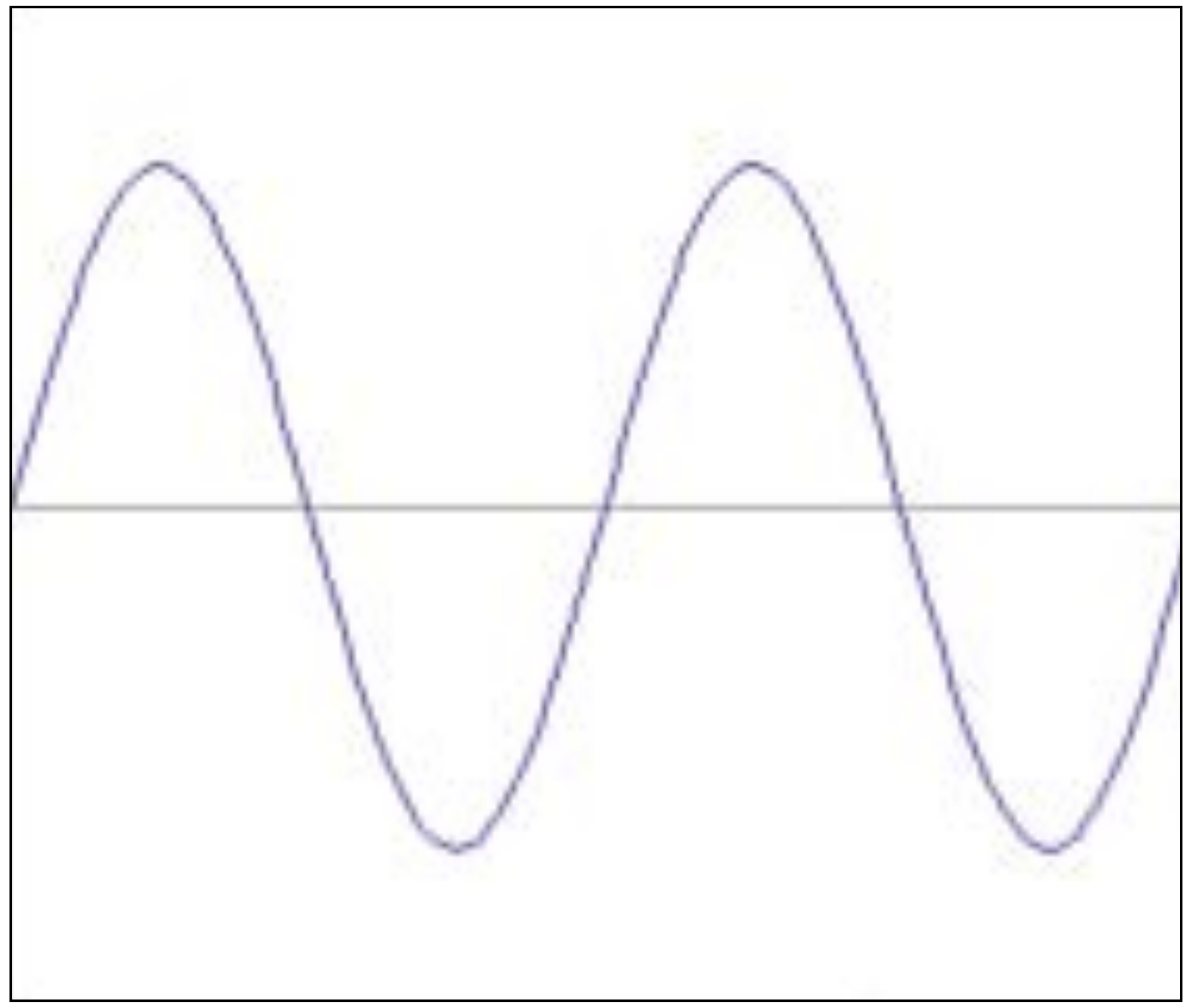
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?



=



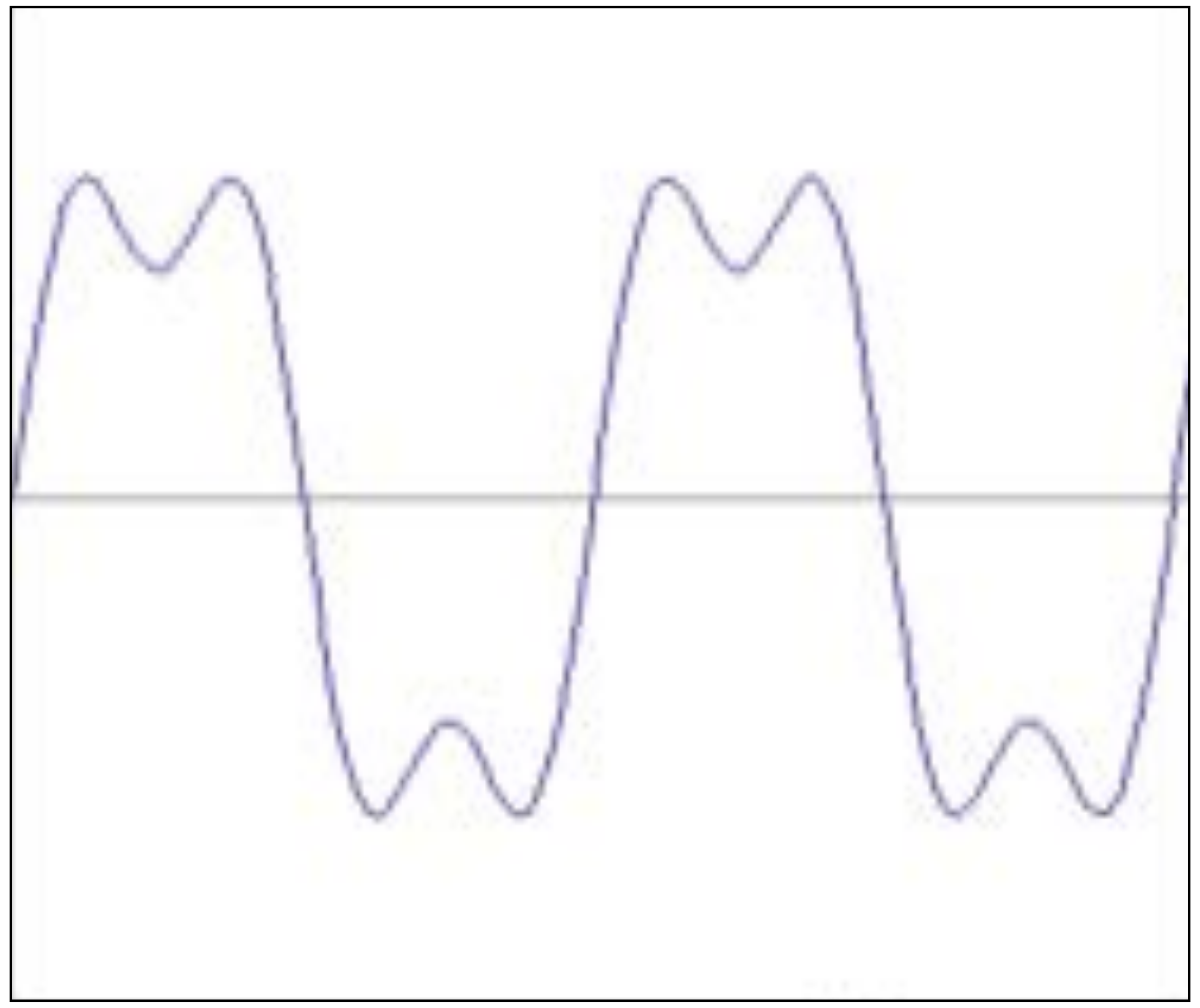
$\sin(2\pi x)$

\+ ?

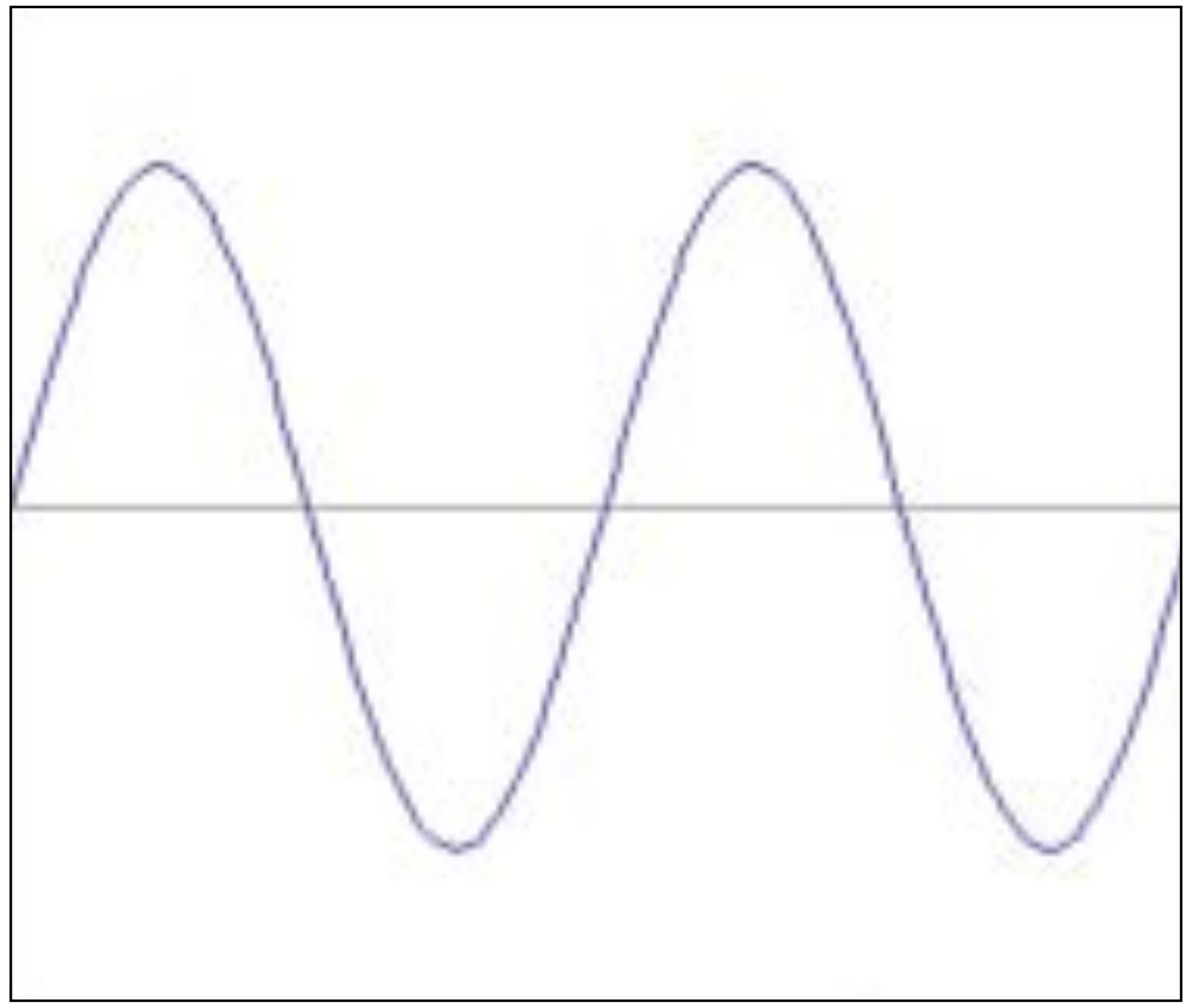
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

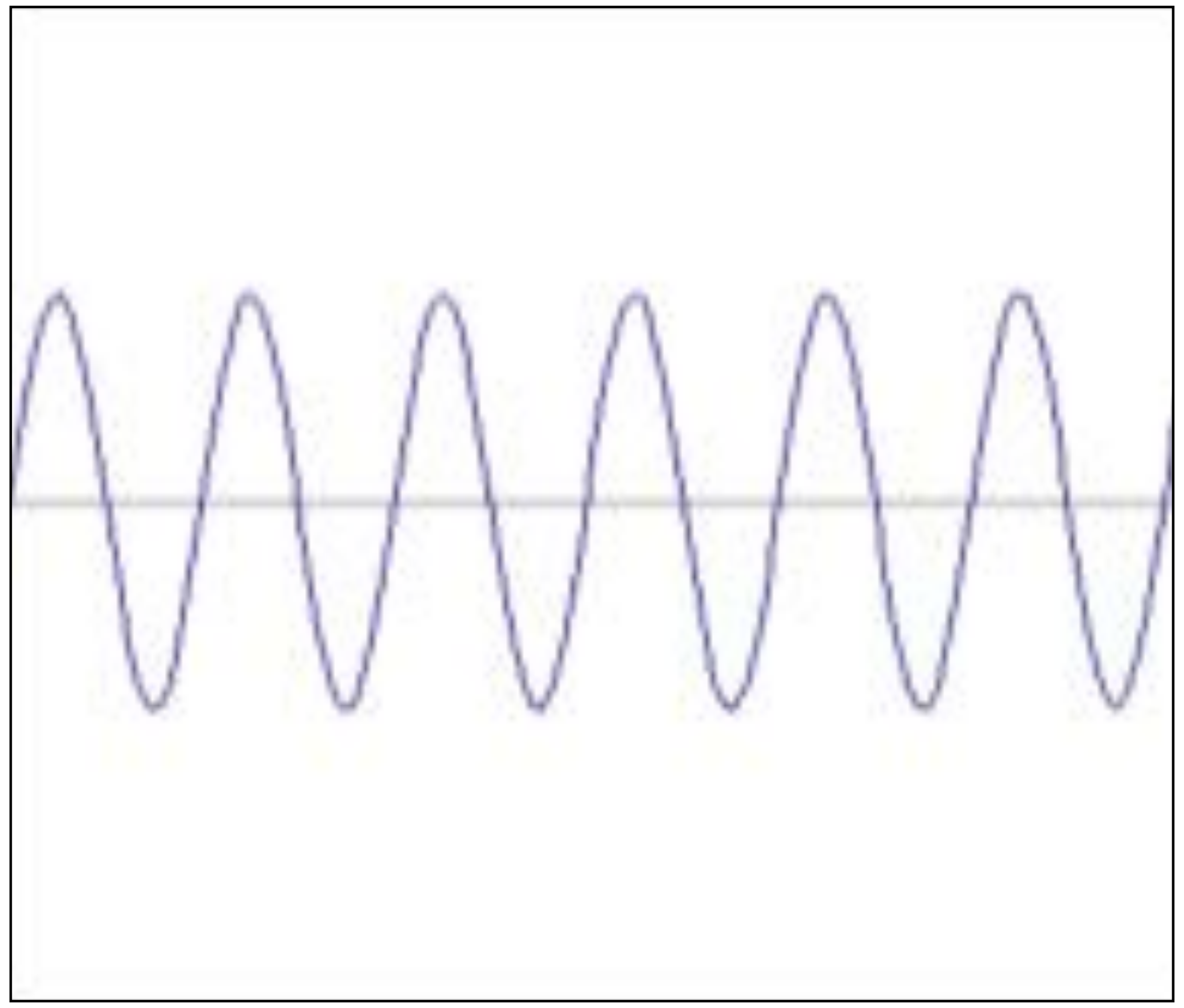


=



$\sin(2\pi x)$

\+

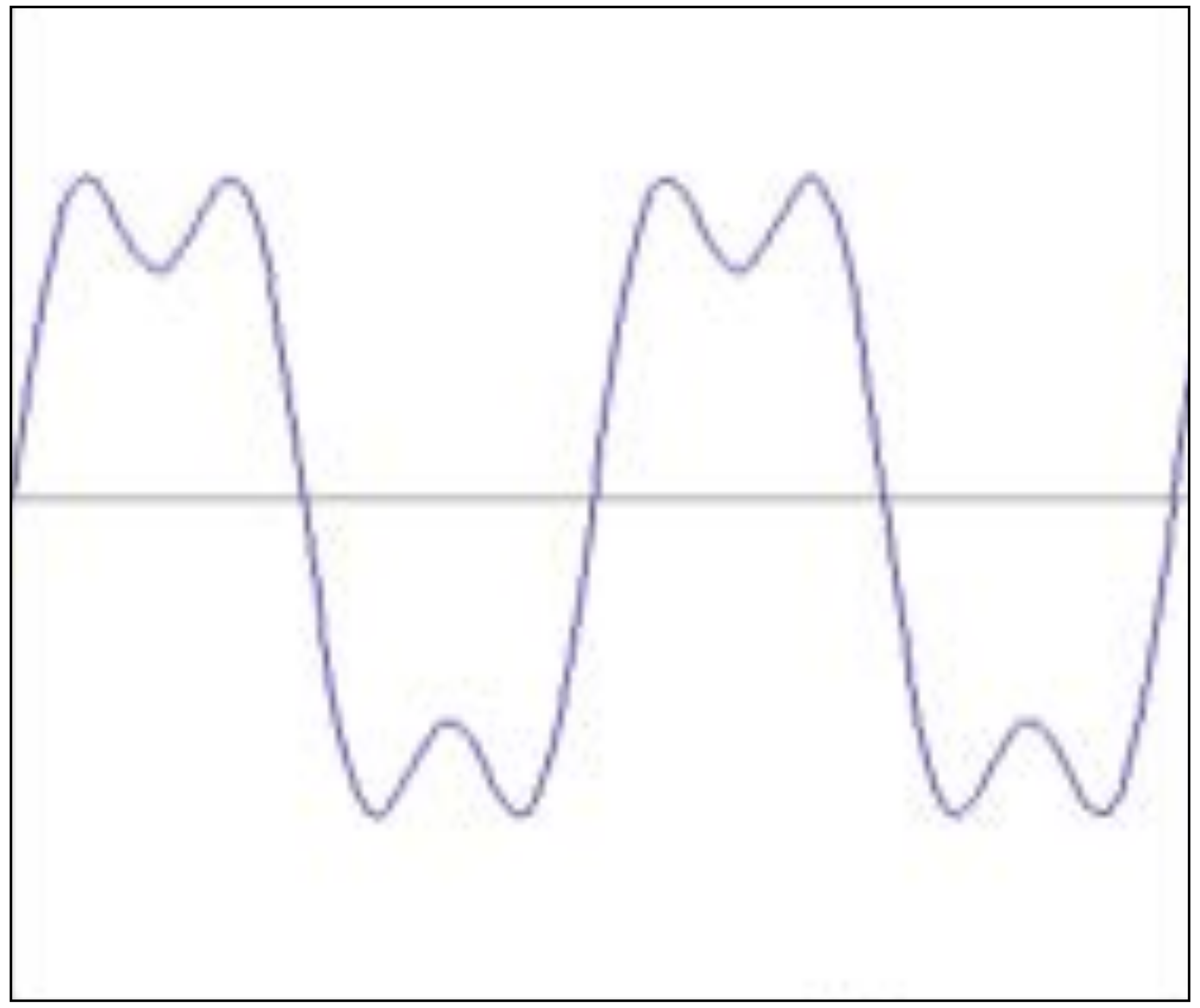


$\frac{1}{3}\sin(2\pi 3x)$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

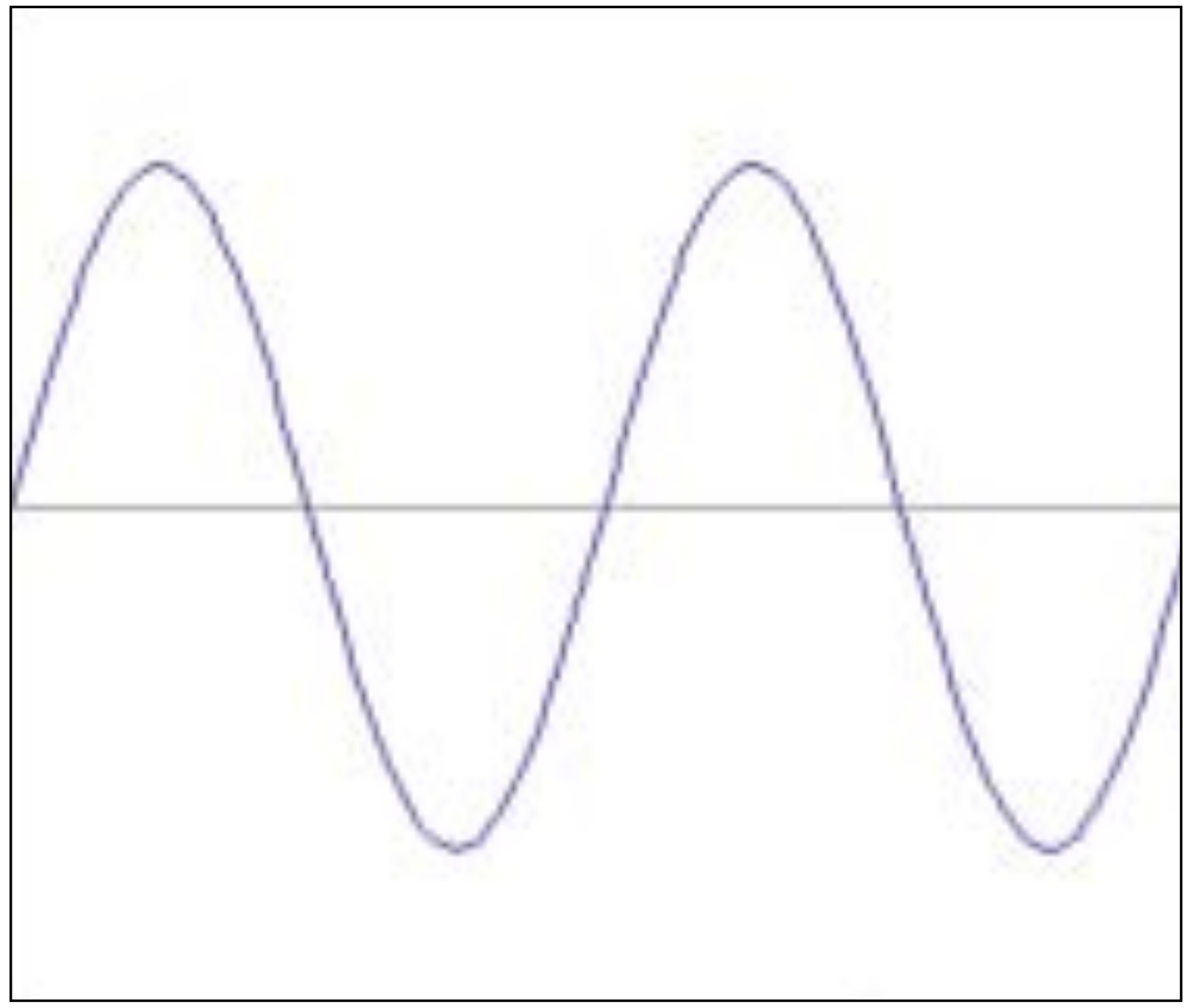
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?



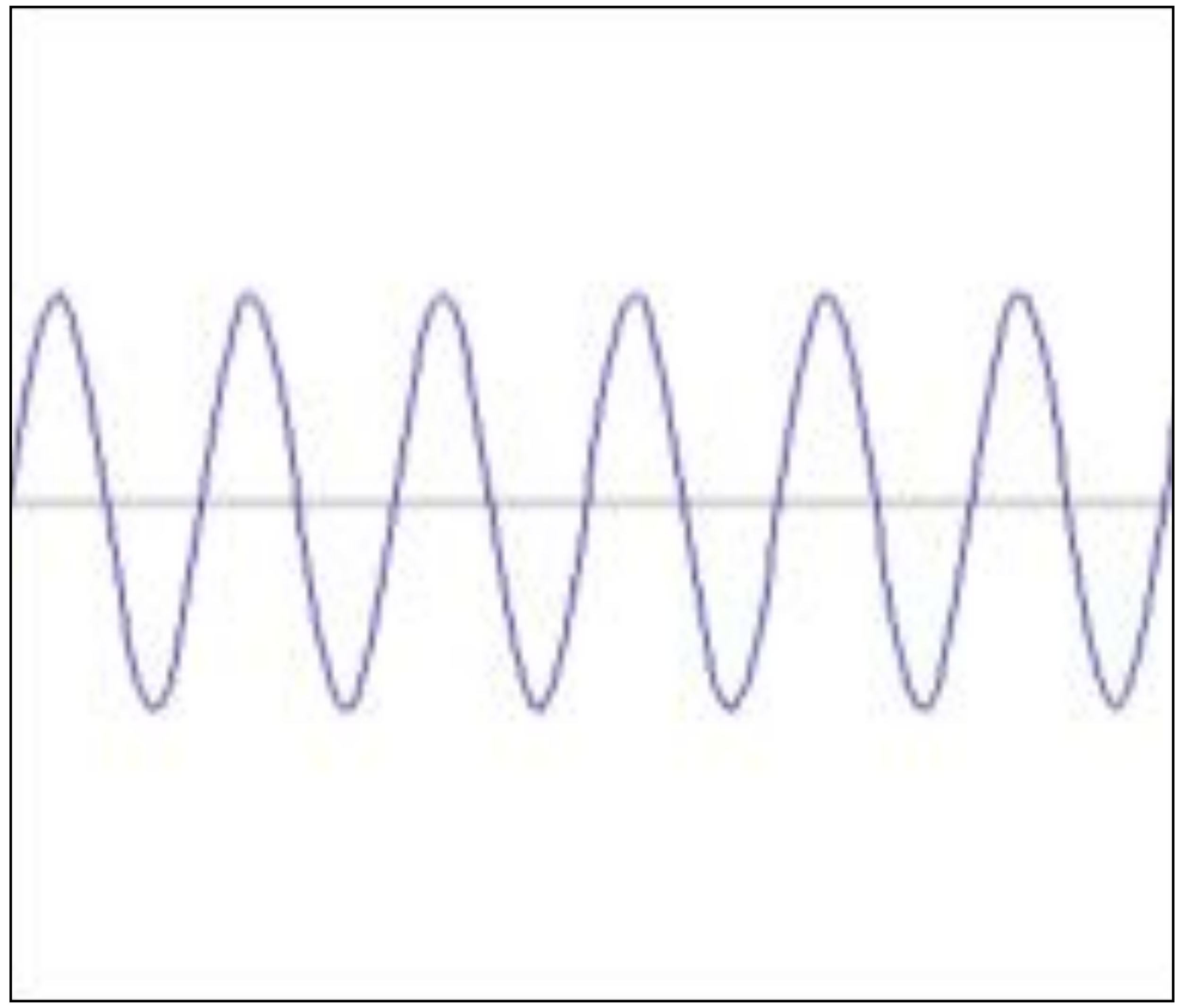
$$f(x) = \sin(2\pi x) + \frac{1}{3}\sin(2\pi 3x)$$

=



$$\sin(2\pi x)$$

+

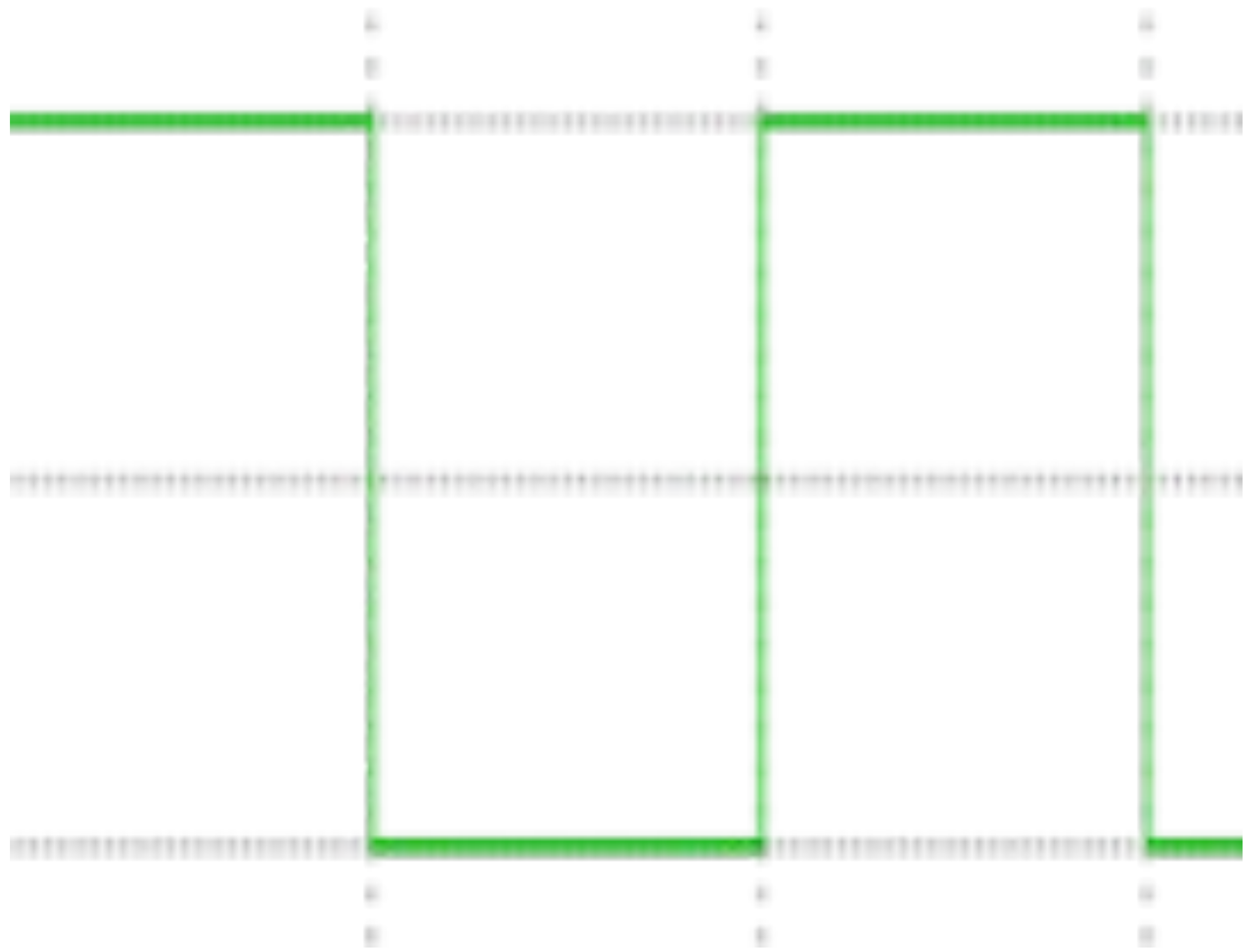


$$\frac{1}{3}\sin(2\pi 3x)$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?



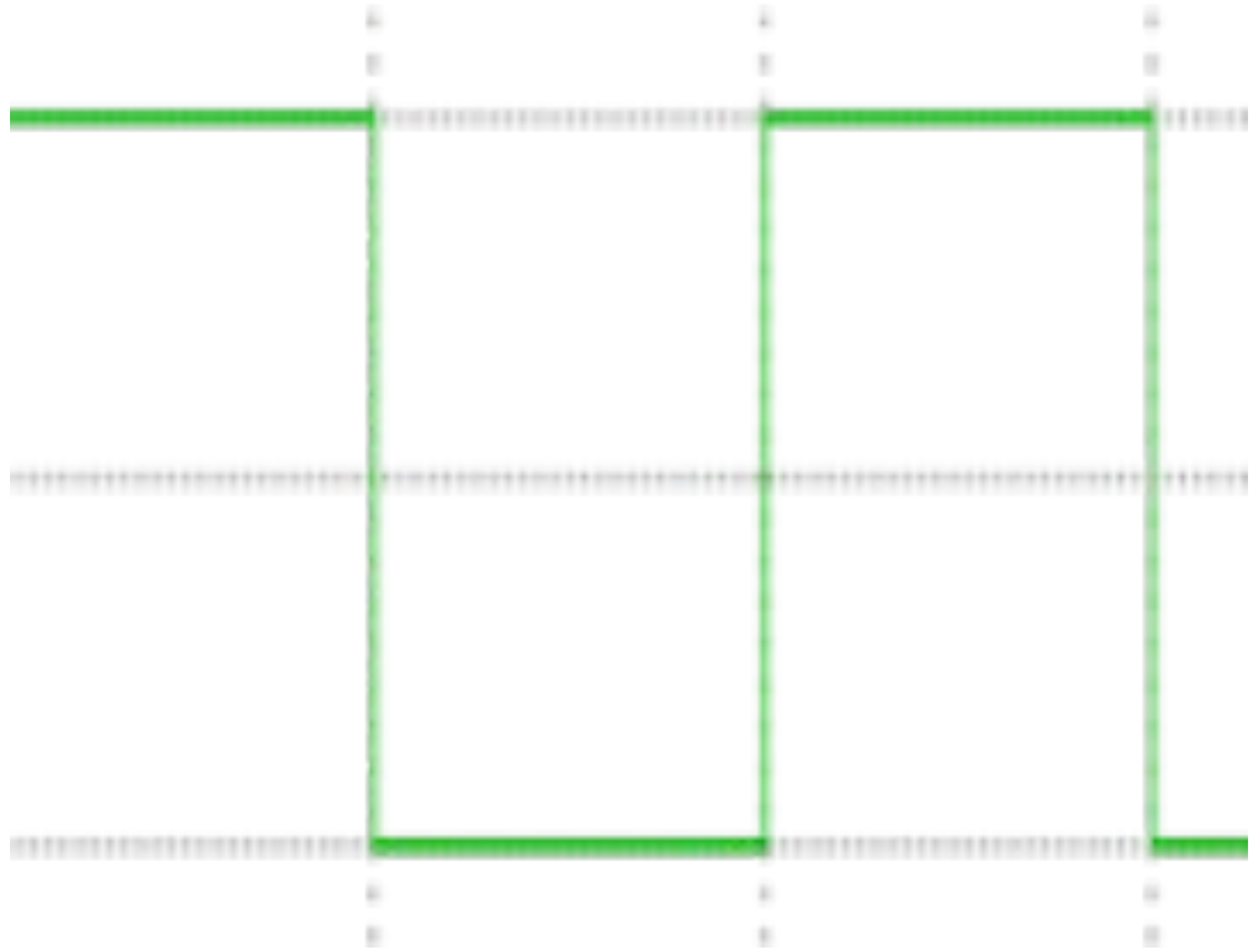
square wave

≈ ? + ?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

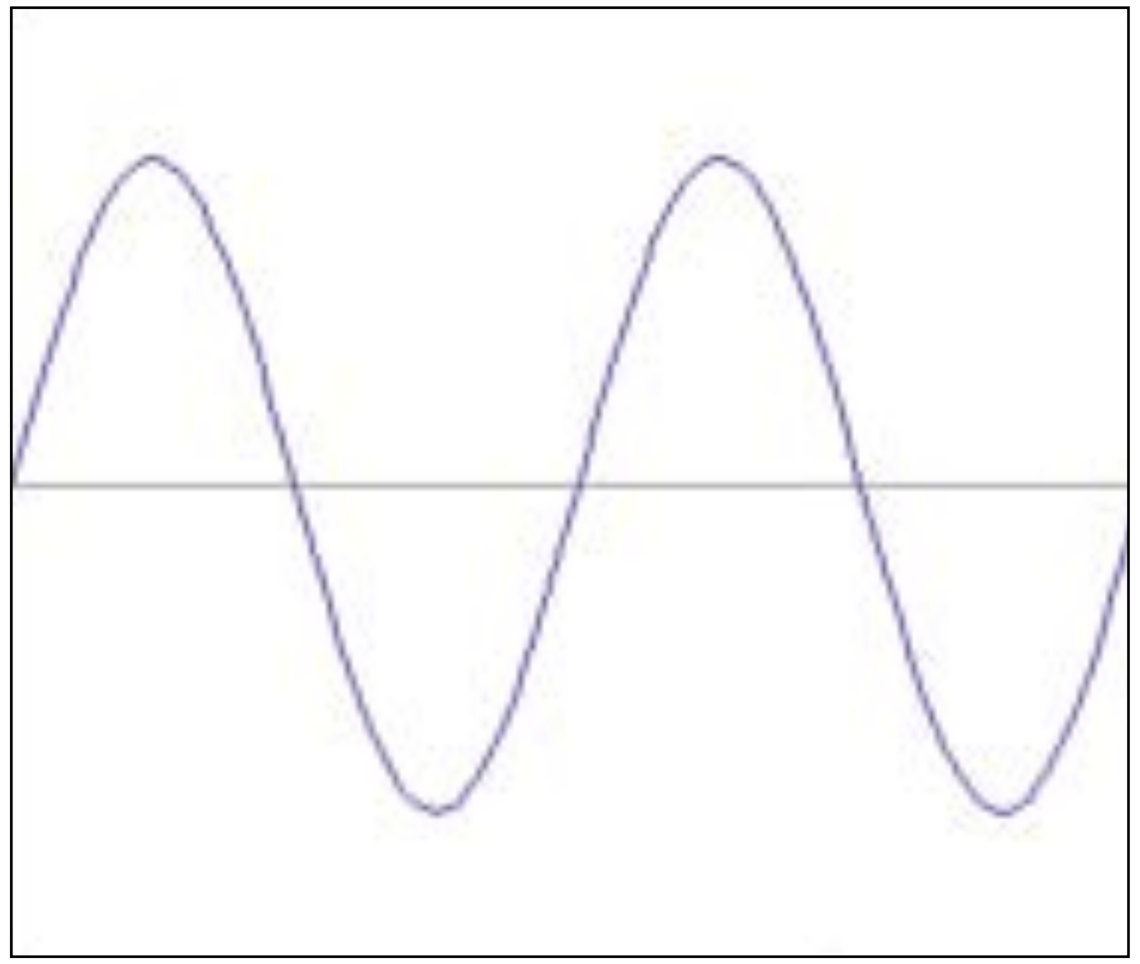
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

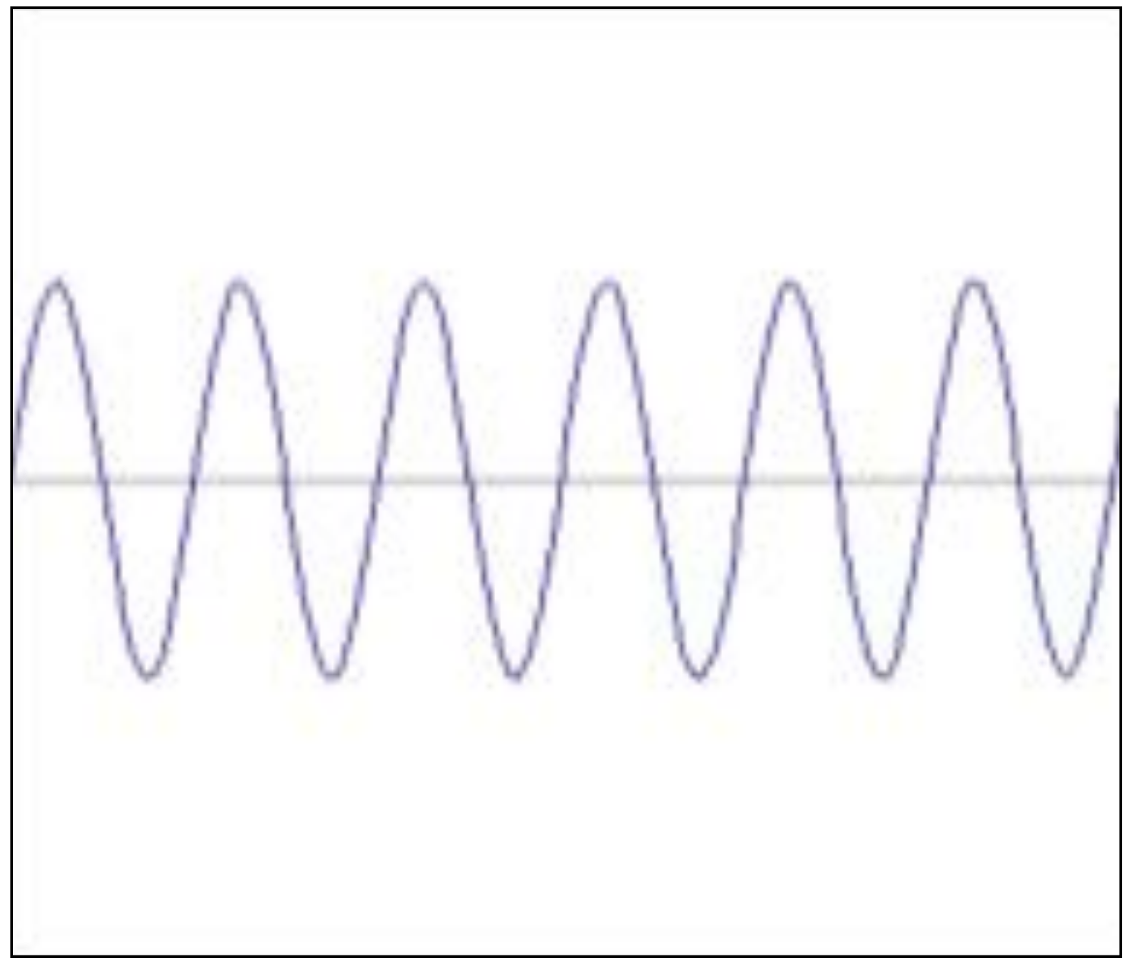


square wave

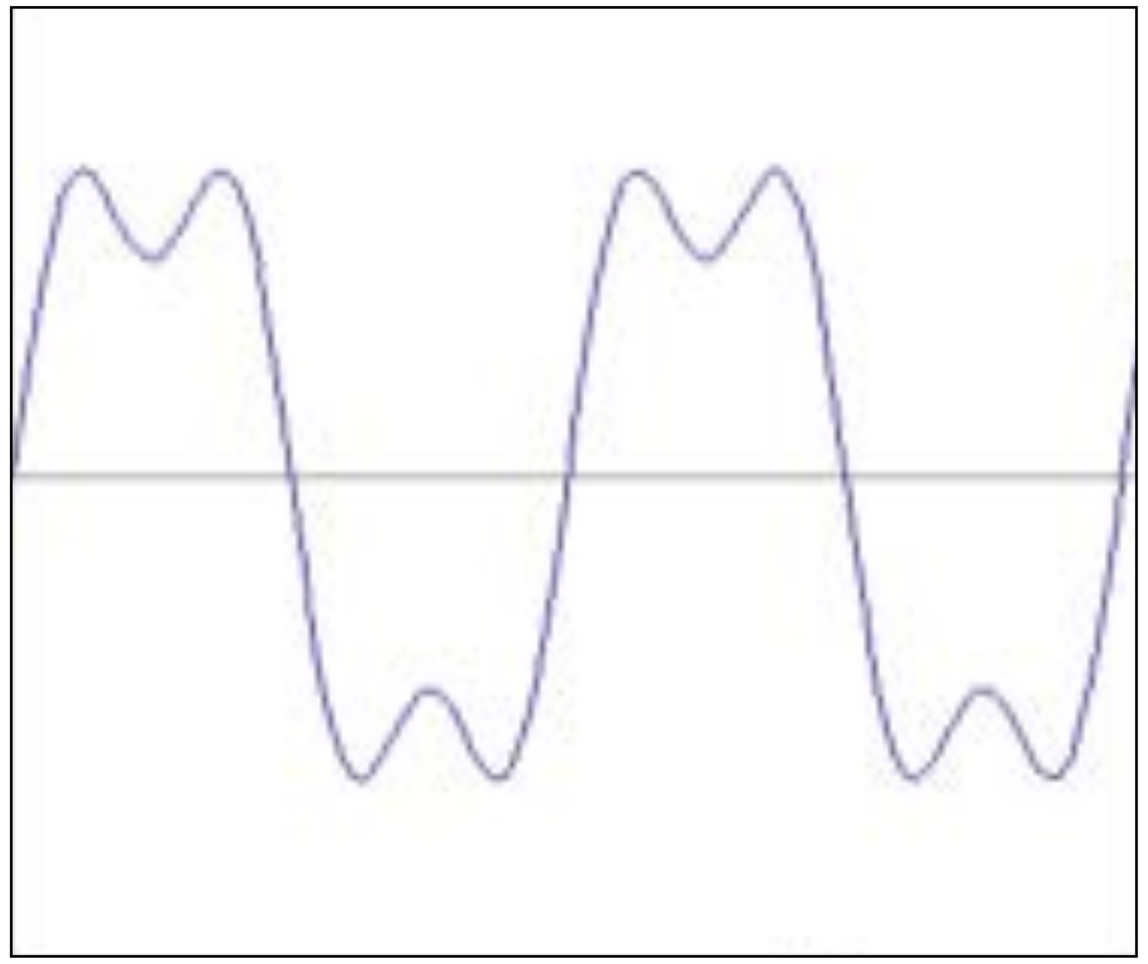
$\approx$



$+$



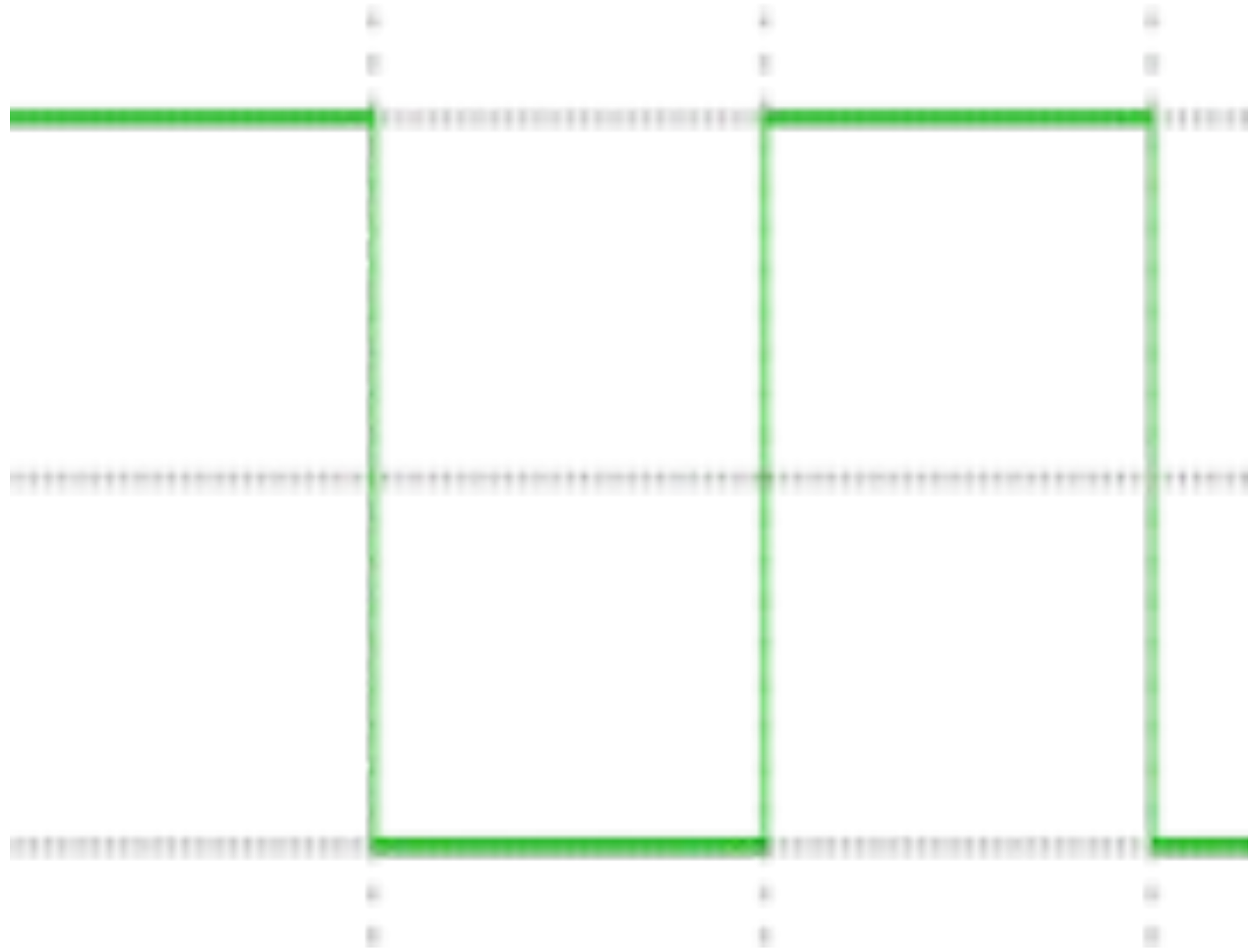
$=$



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

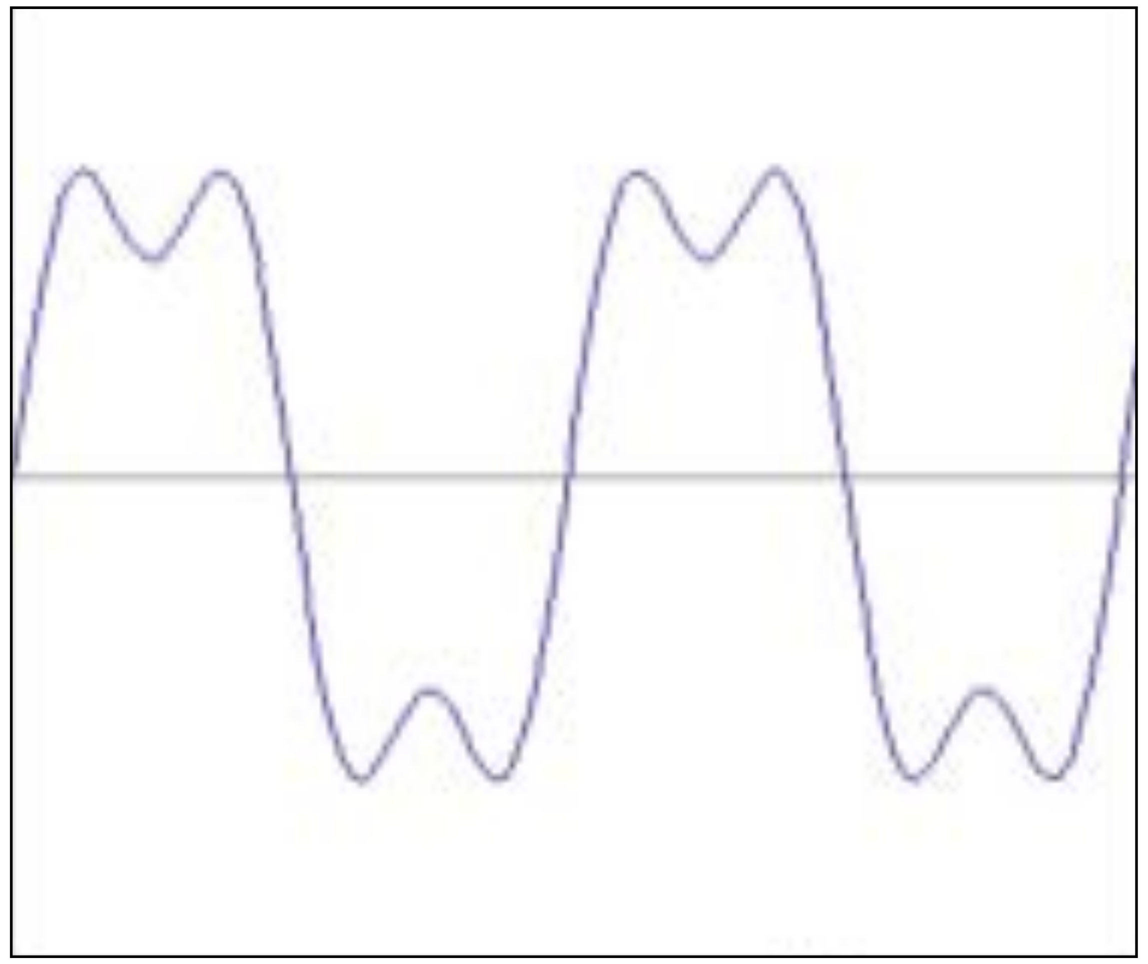
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

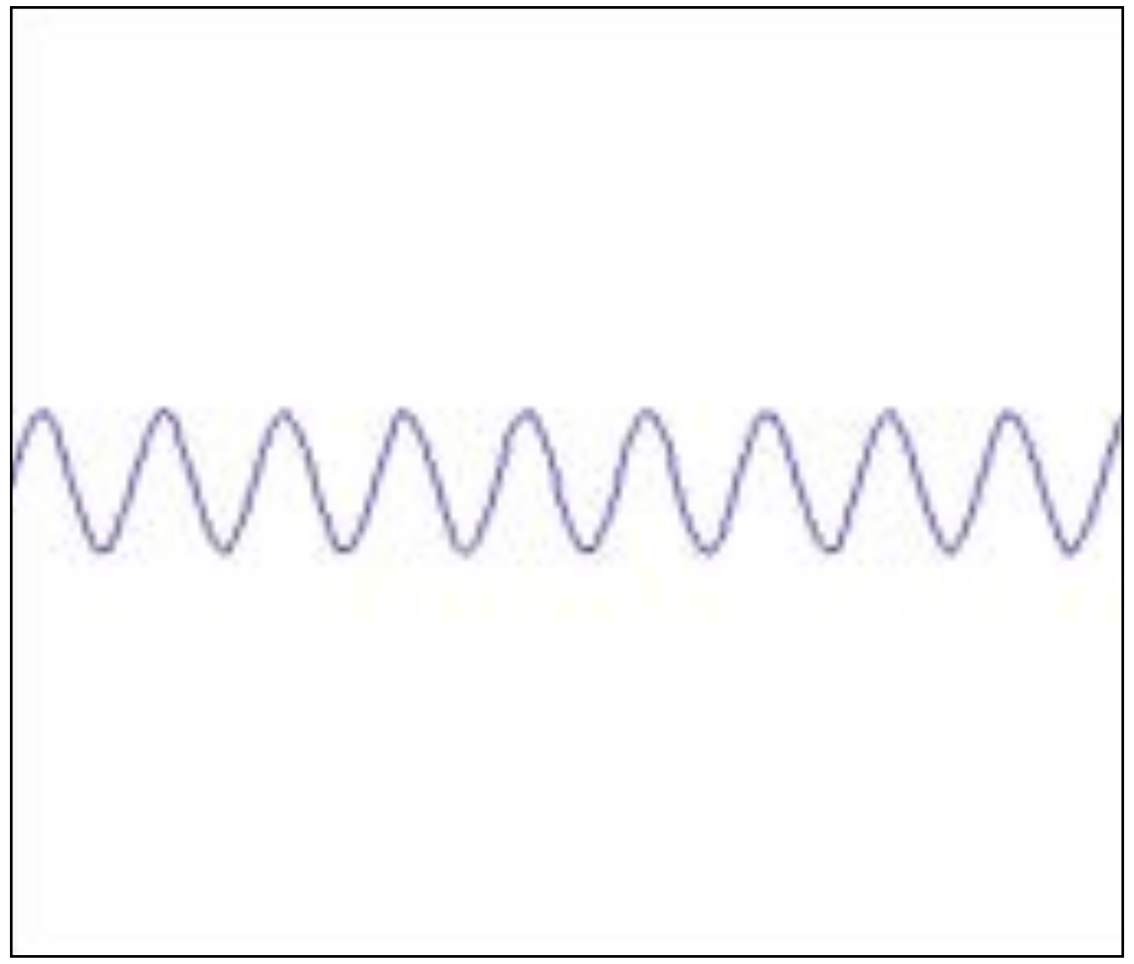


square wave

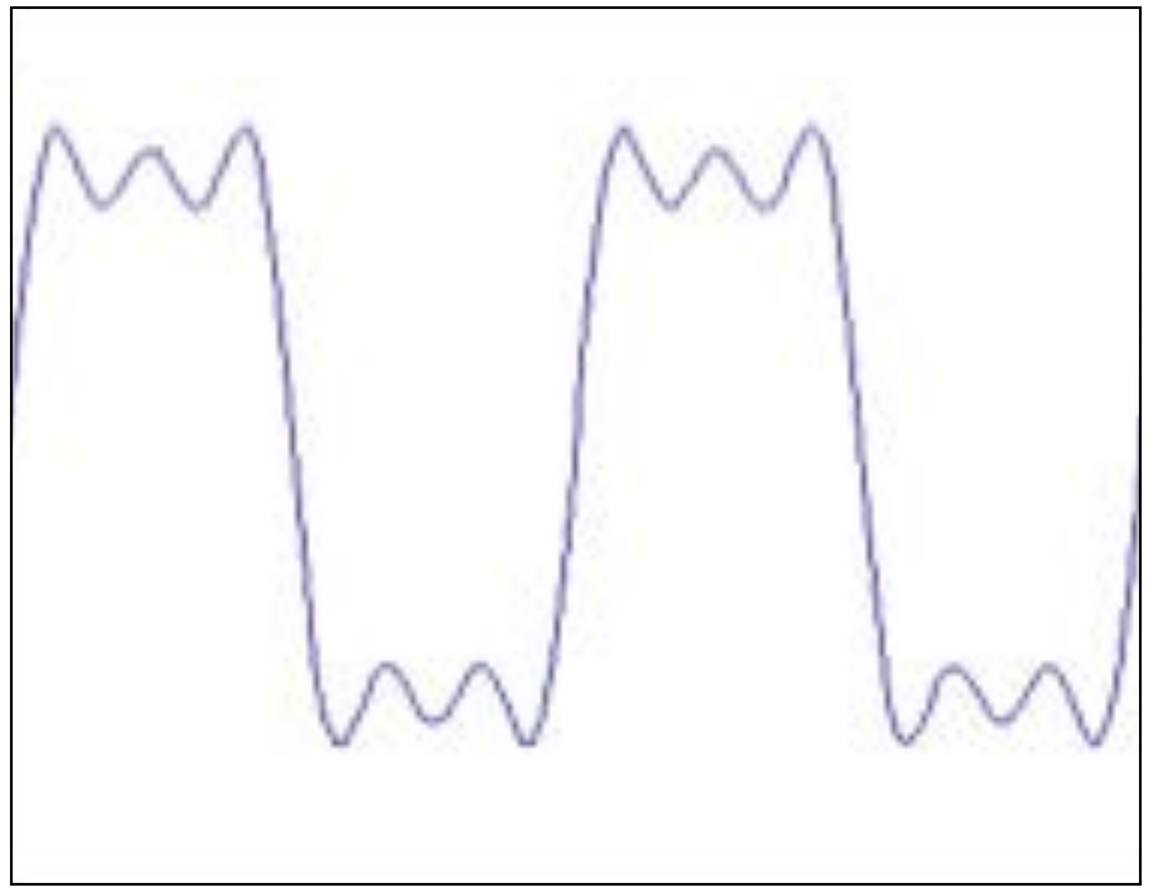
≈



+



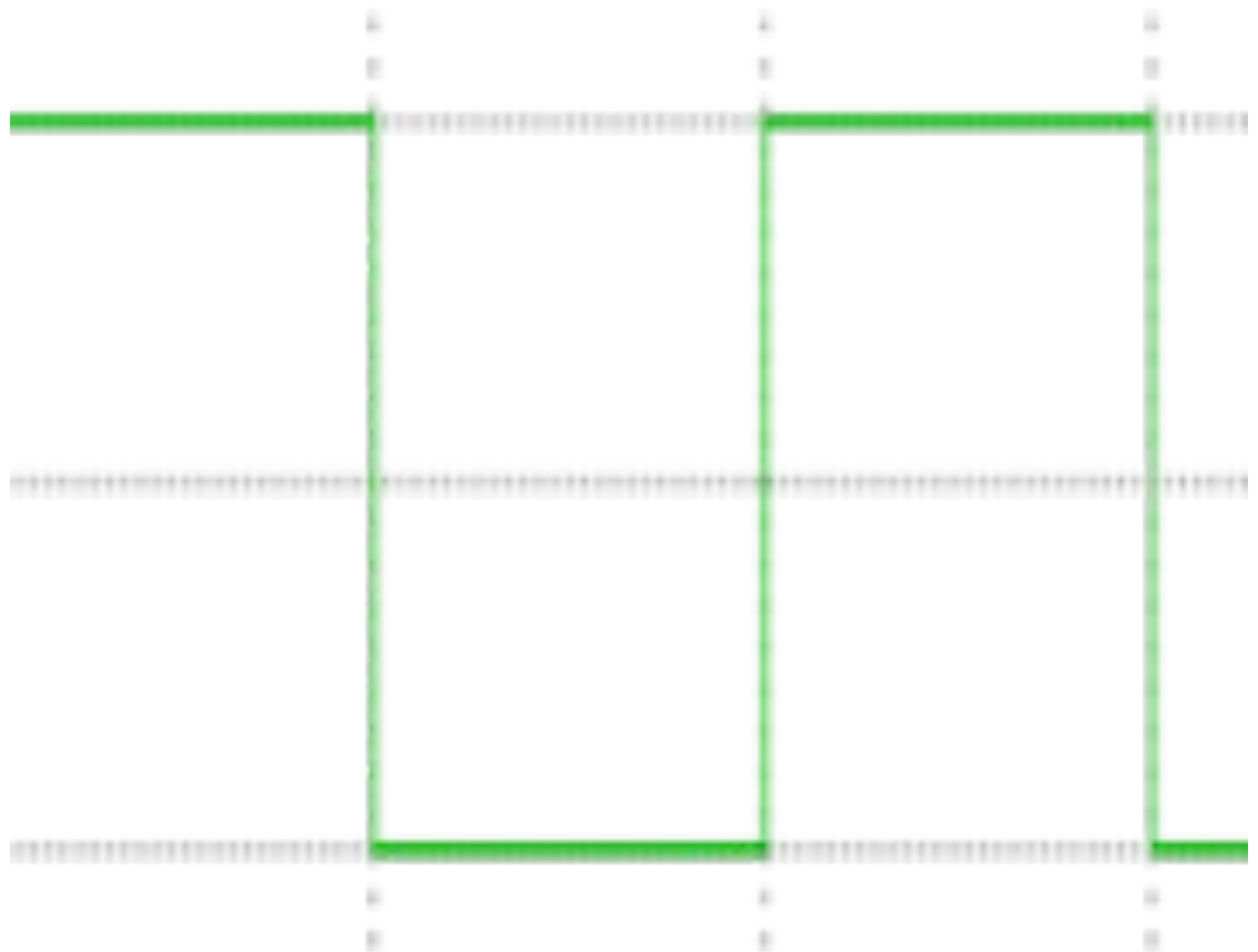
=



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

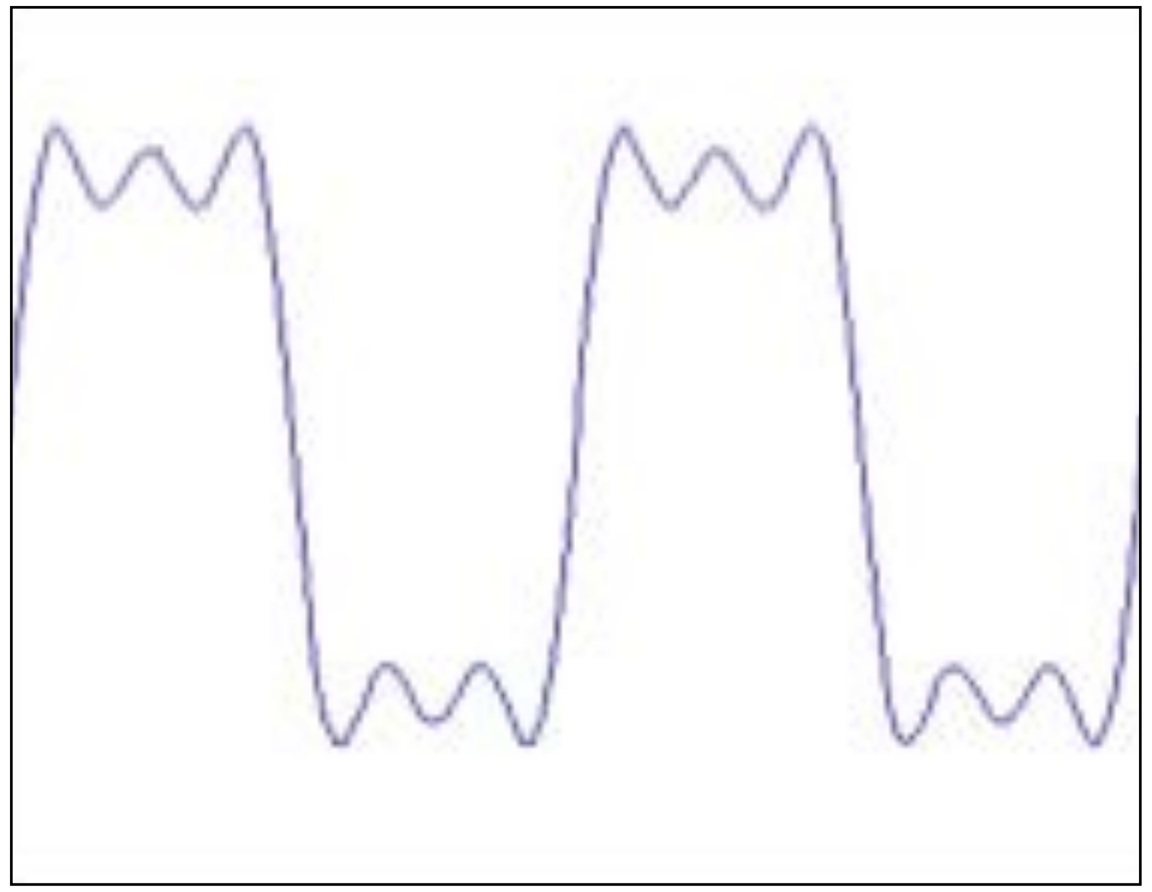
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

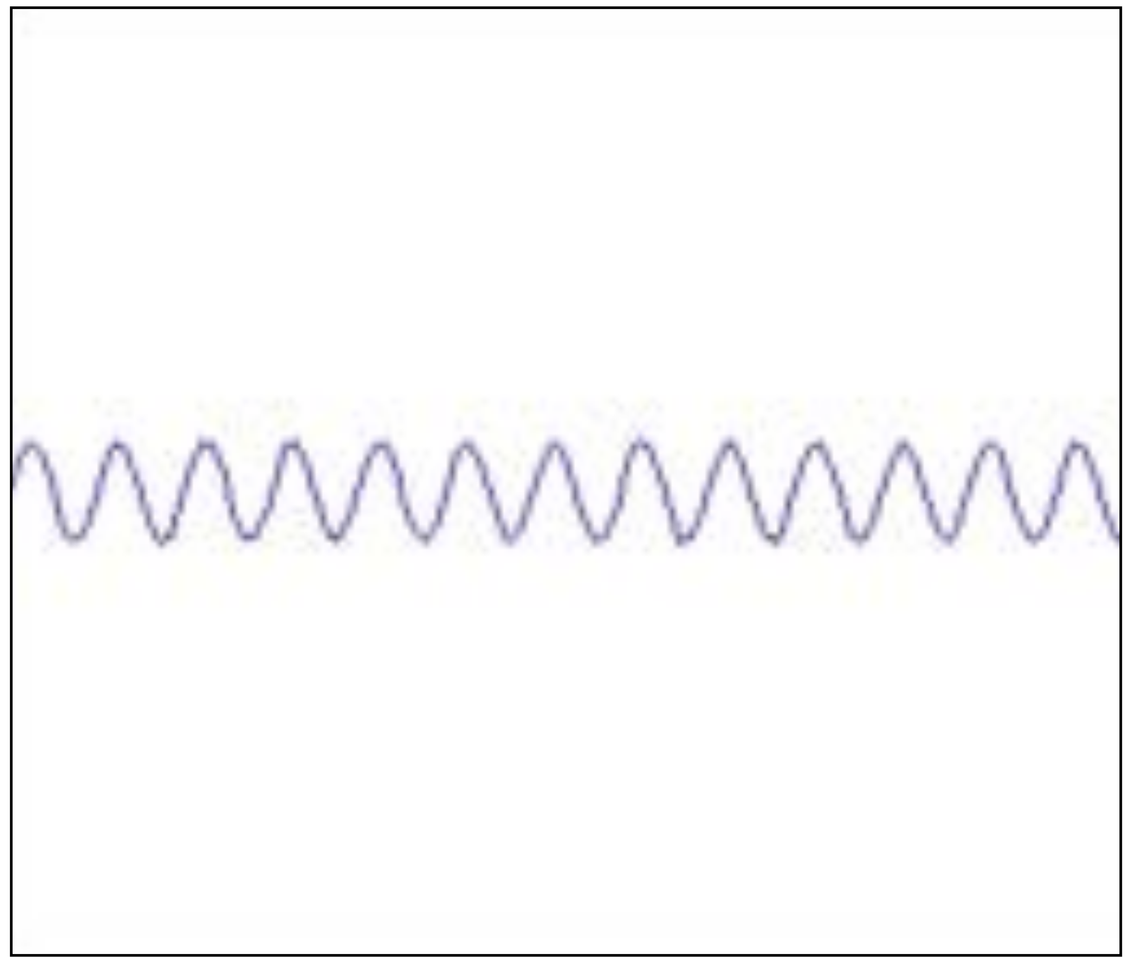


square wave

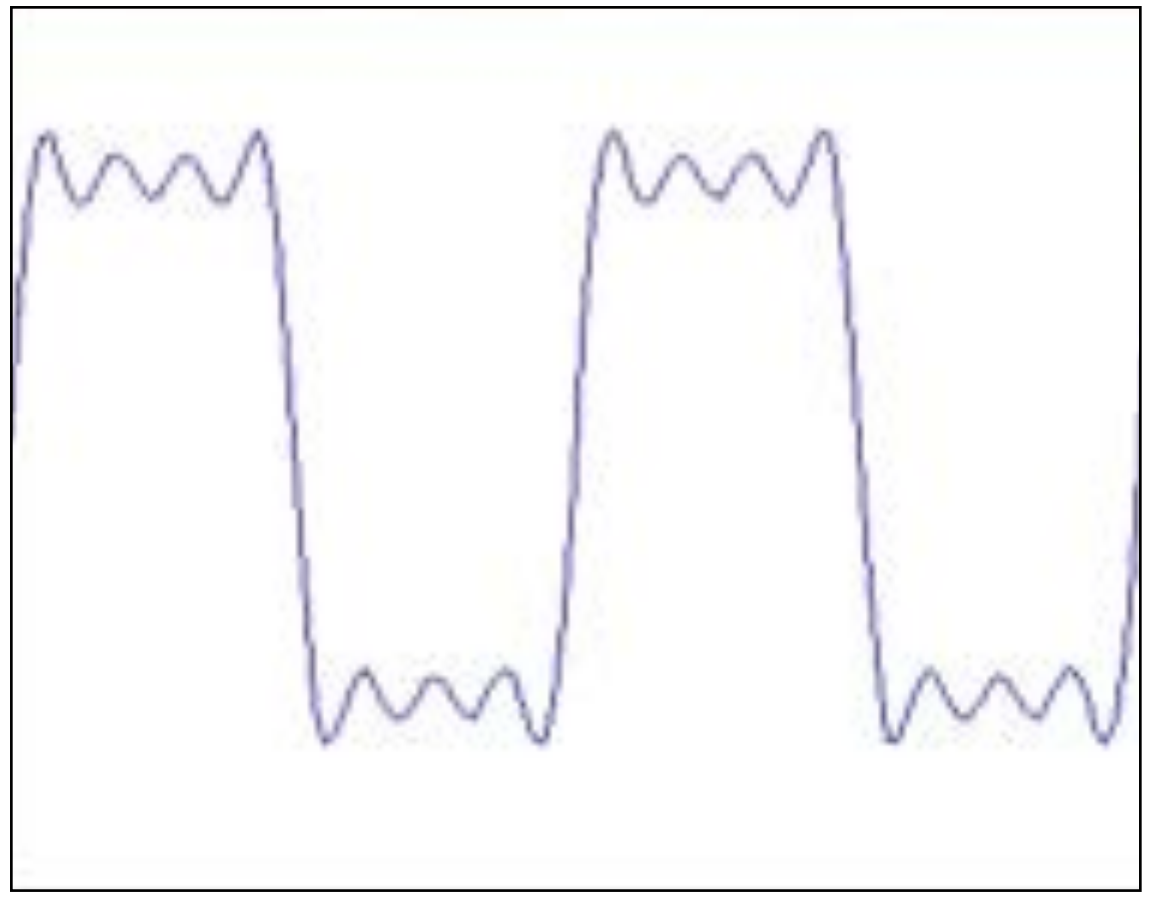
$\approx$



$+$



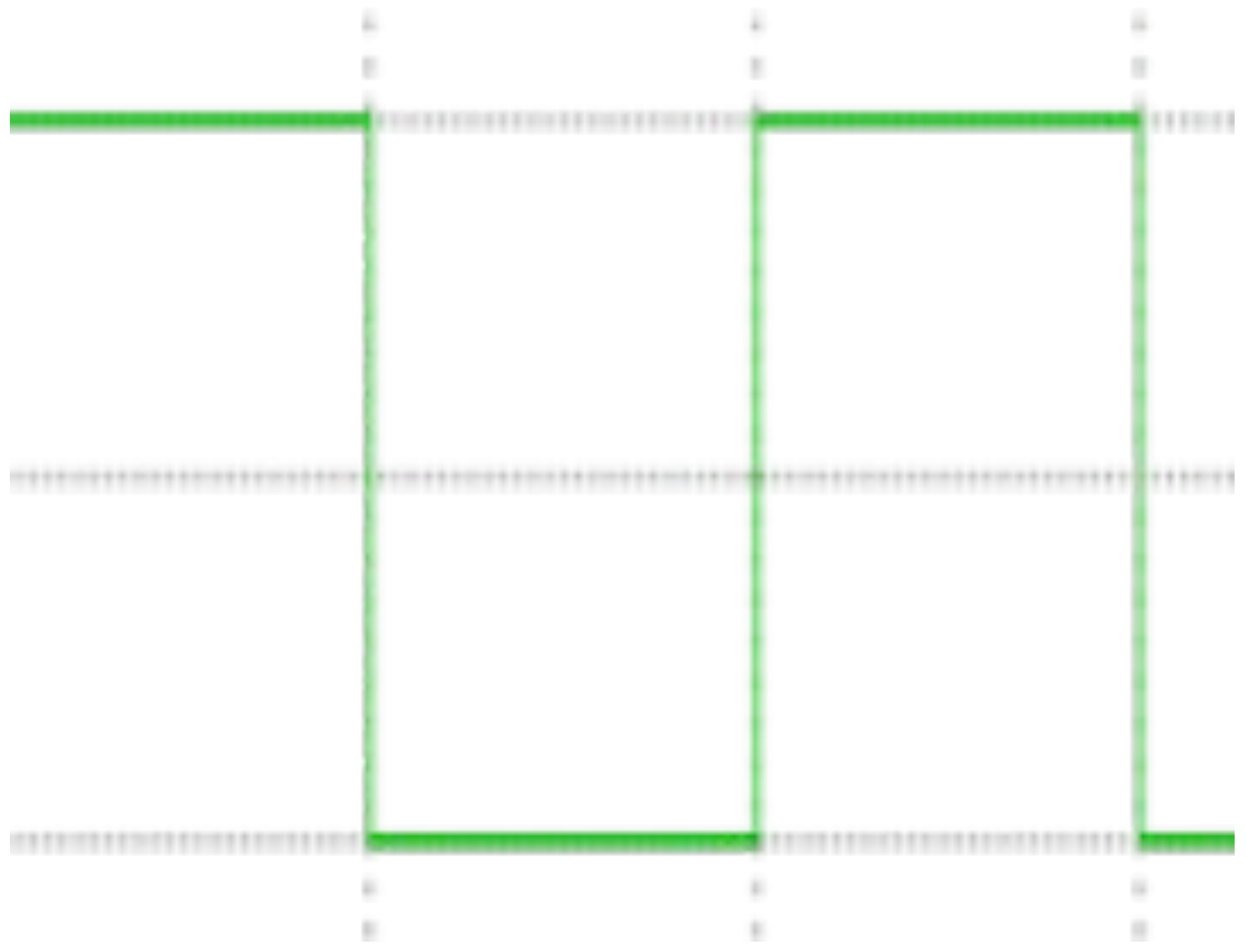
$=$



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

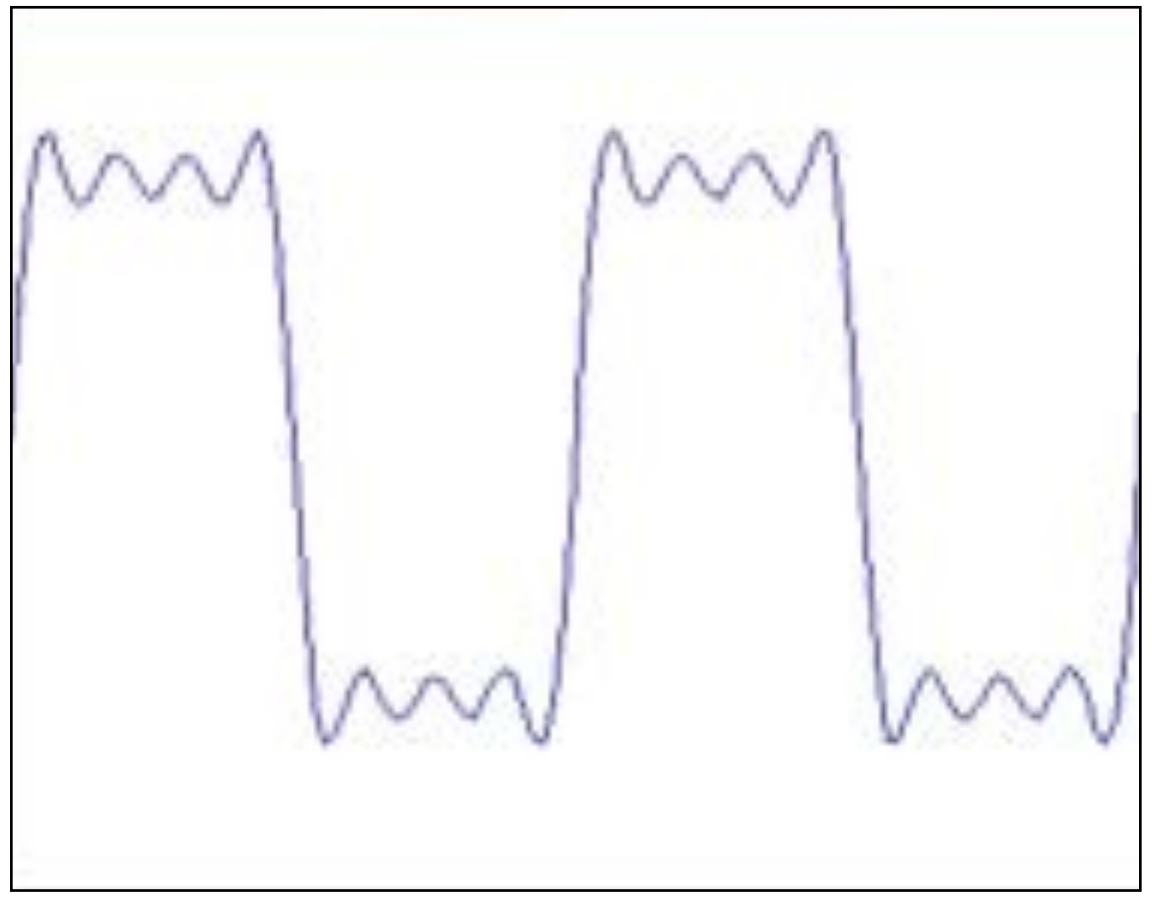
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

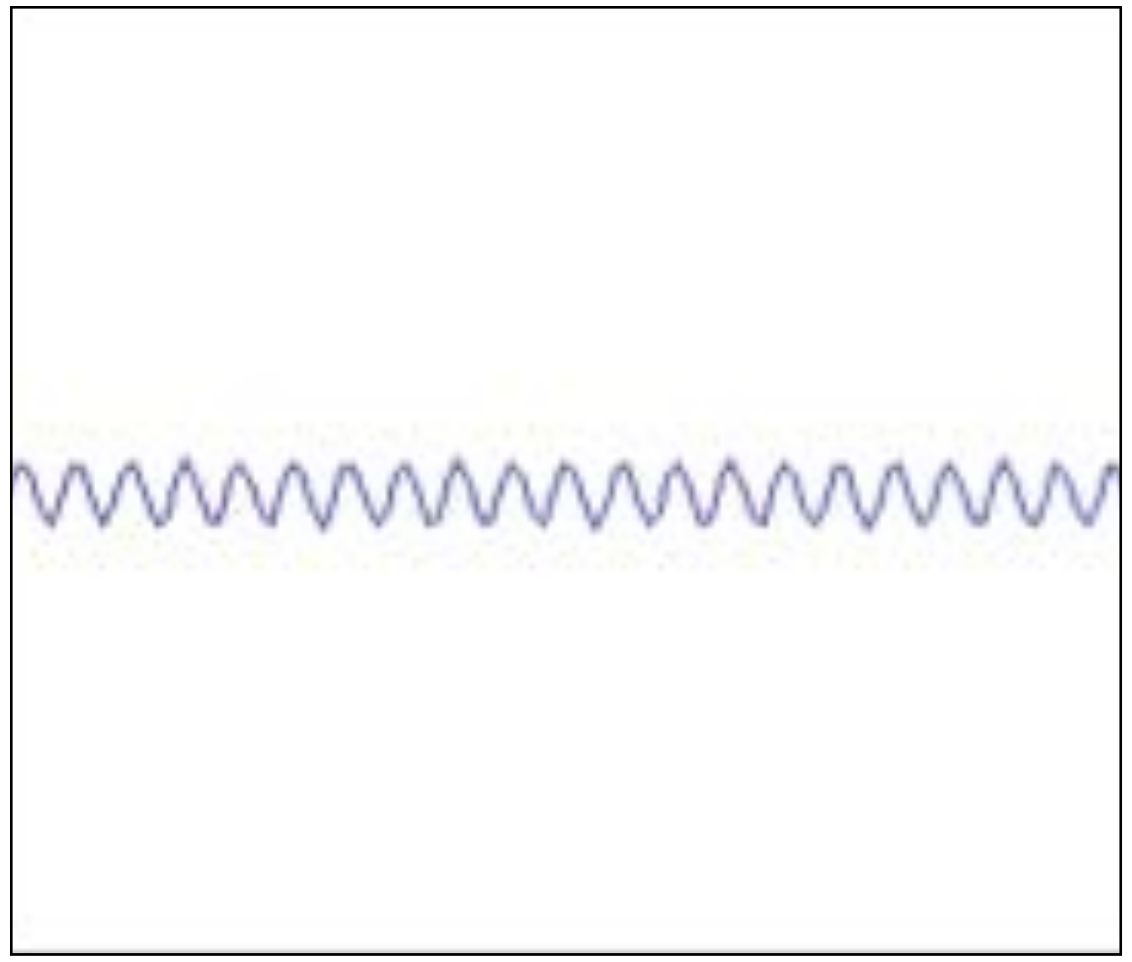


square wave

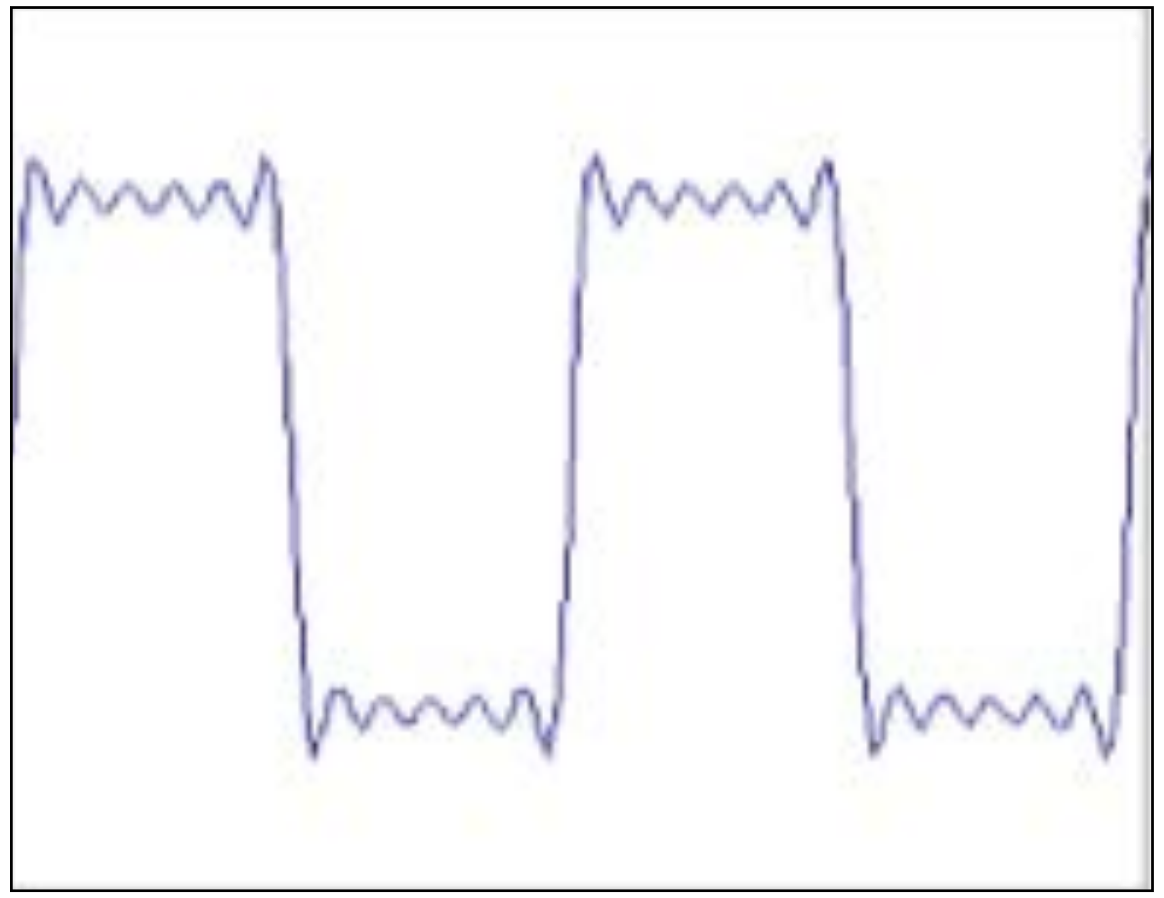
≈



+



=

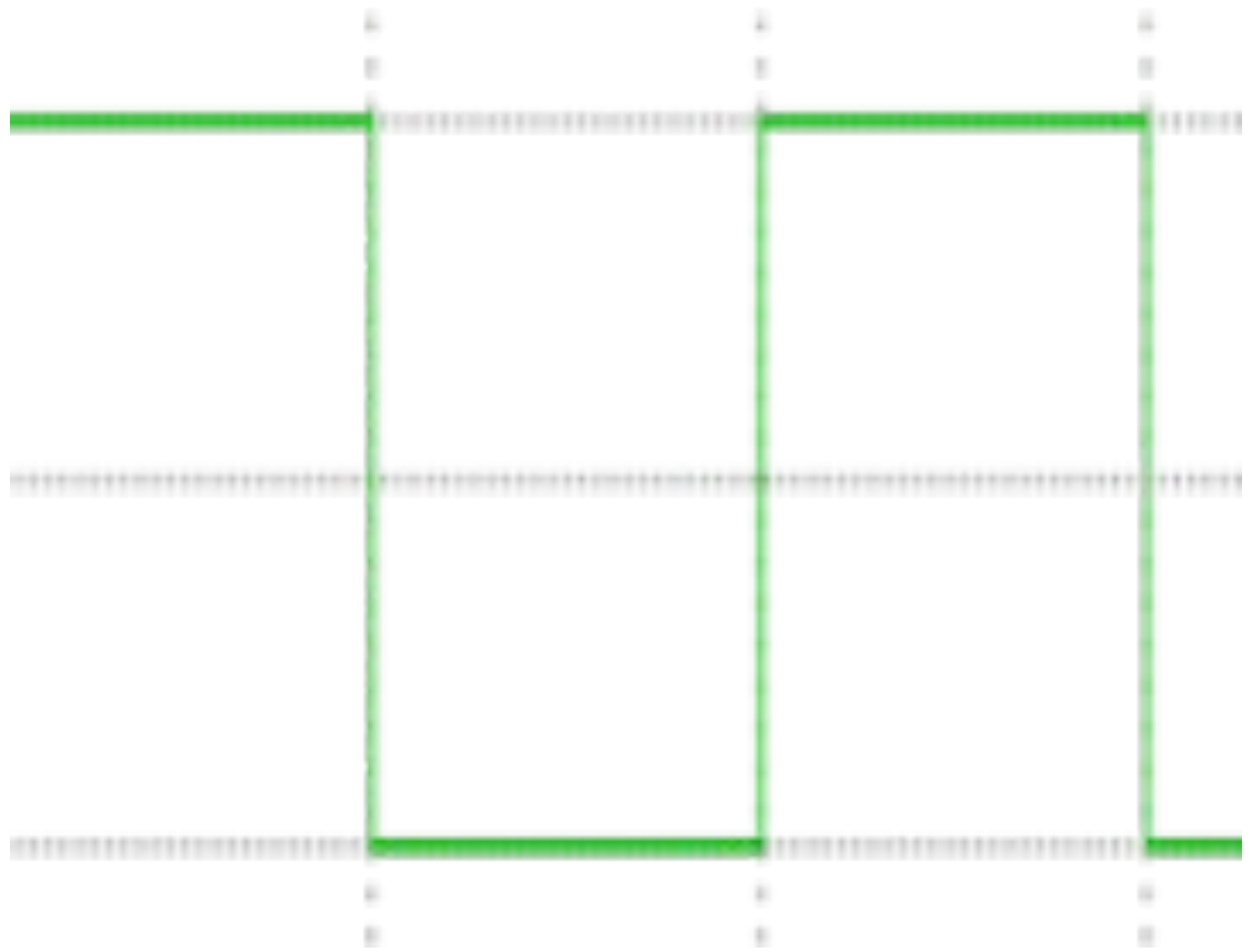


How would you express this mathematically?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?



square wave

$$= A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi k x)$$

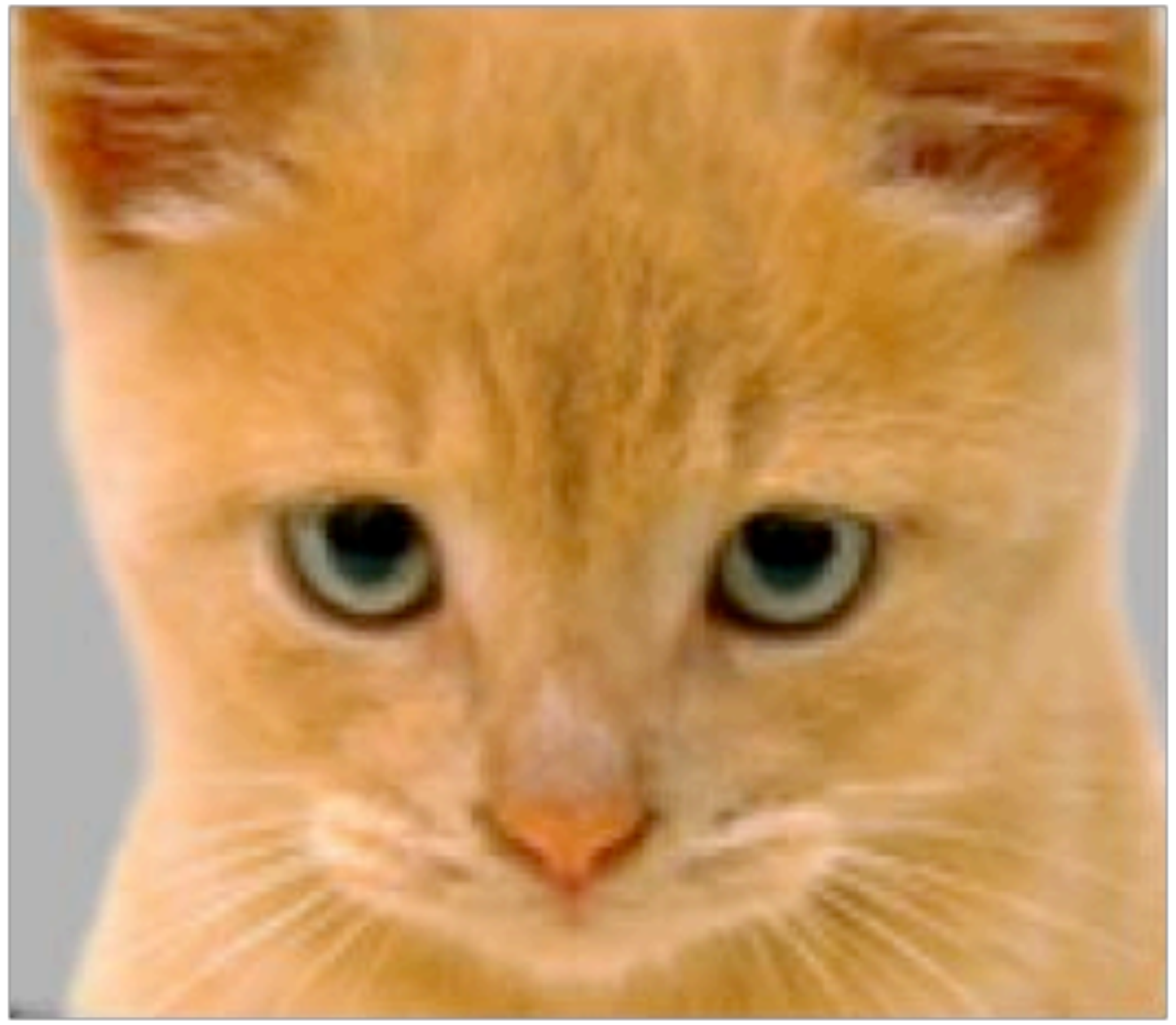
infinite sum of sine waves

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Low-Pass Filtering in **1D**

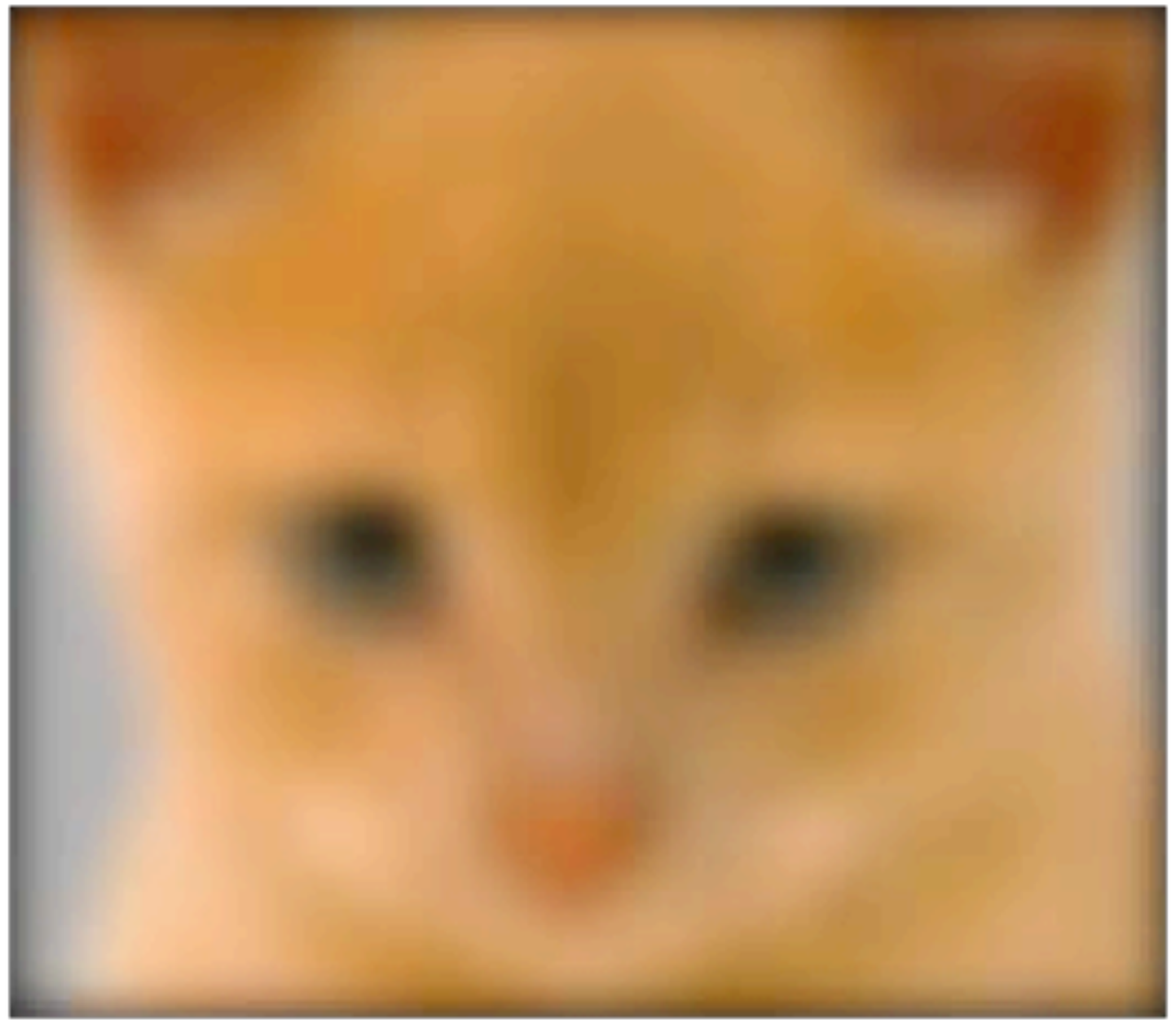
4.4

Assignment 1: **Low/High Pass** Filtering



Original

$$I(x, y)$$

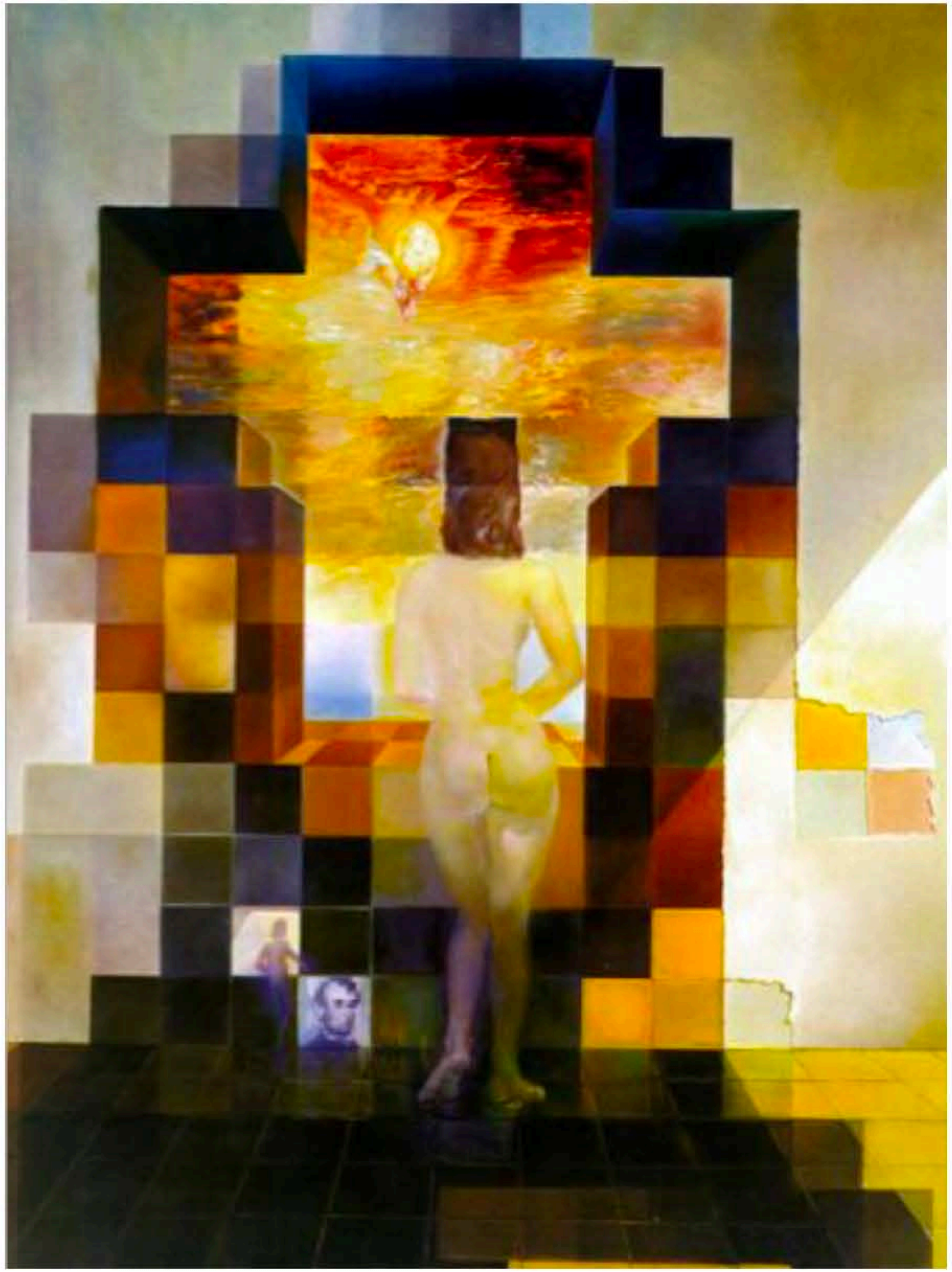


Low-Pass Filter

$$I(x, y) * g(x, y)$$

High-Pass Filter

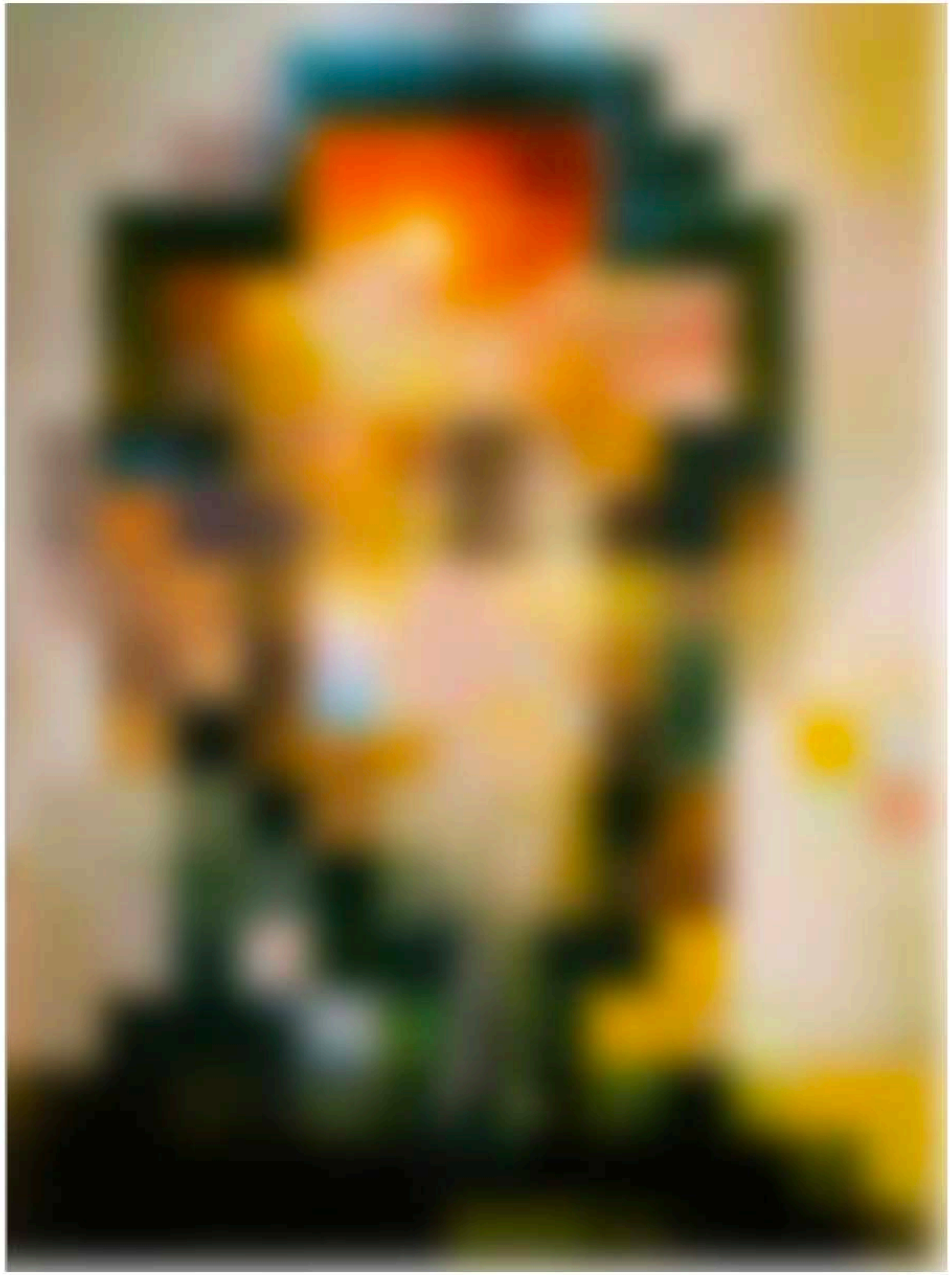
$$I(x, y) - I(x, y) * g(x, y)$$



Gala Contemplating the Mediterranean Sea Which at Twenty Meters Becomes the Portrait of Abraham Lincoln (Homage to Rothko)

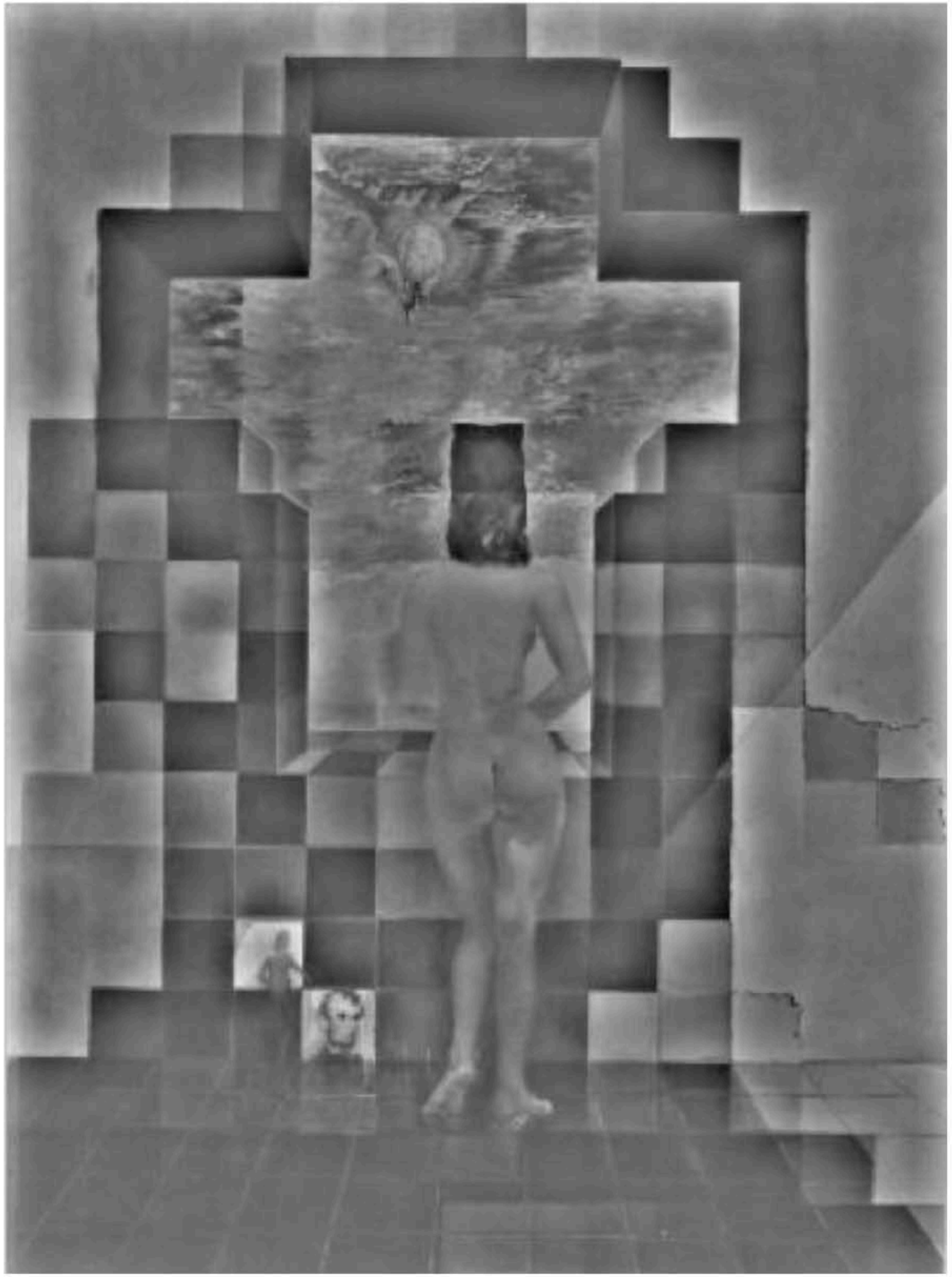
Salvador Dali, 1976

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)



Low-pass filtered version

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)



High-pass filtered version

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)