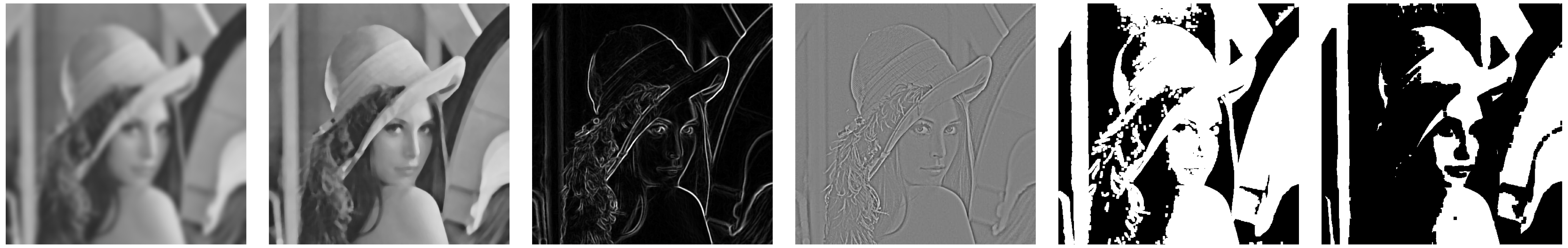
THE UNIVERSITY OF BRITISH COLUMBIA

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:

— **Box, Gaussian, Pillbox** filters

— **Separability**

— The **Convolution Theorem**

— **Fourier** Space Representations

Readings:

— **Today's** Lecture: none

— **Next** Lecture: Forsyth & Ponce (2nd ed.) 4.4

Reminders:

— **Assignment 1:** Image Filtering and Hybrid Images due **January 30th**

Today's "**fun**" Example: Rolling Shutter

Today's "**fun**" Example: Rolling Shutter

Today’s “**fun**” Example: Rolling Shutter

Today's "**fun**" Example: Rolling Shutter

Lecture 3: Re-cap Correlation

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

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

— **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.

Lecture 3: Re-cap 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.

Lecture 3: Re-cap

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

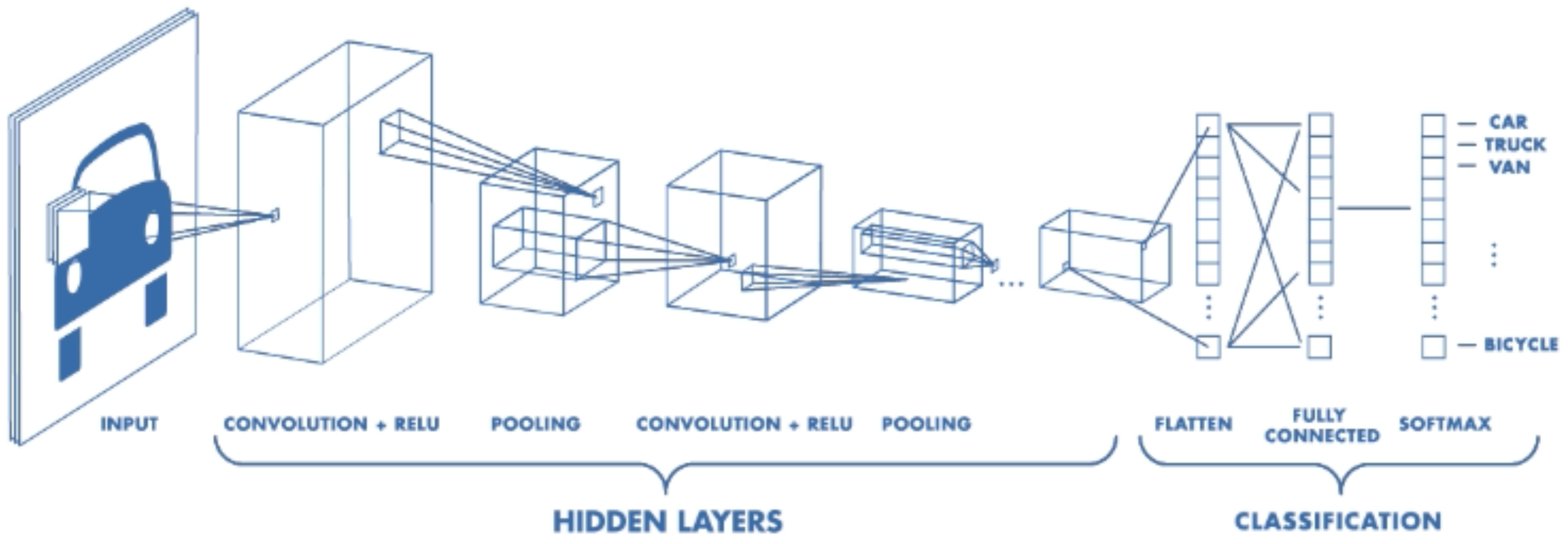
– **Assume periodicity.** Top row wraps around to the bottom row; leftmost column wraps around to rightmost column.

Simple **examples** of filtering:

— copy, shift, smoothing, sharpening

Preview: Why convolutions are important?

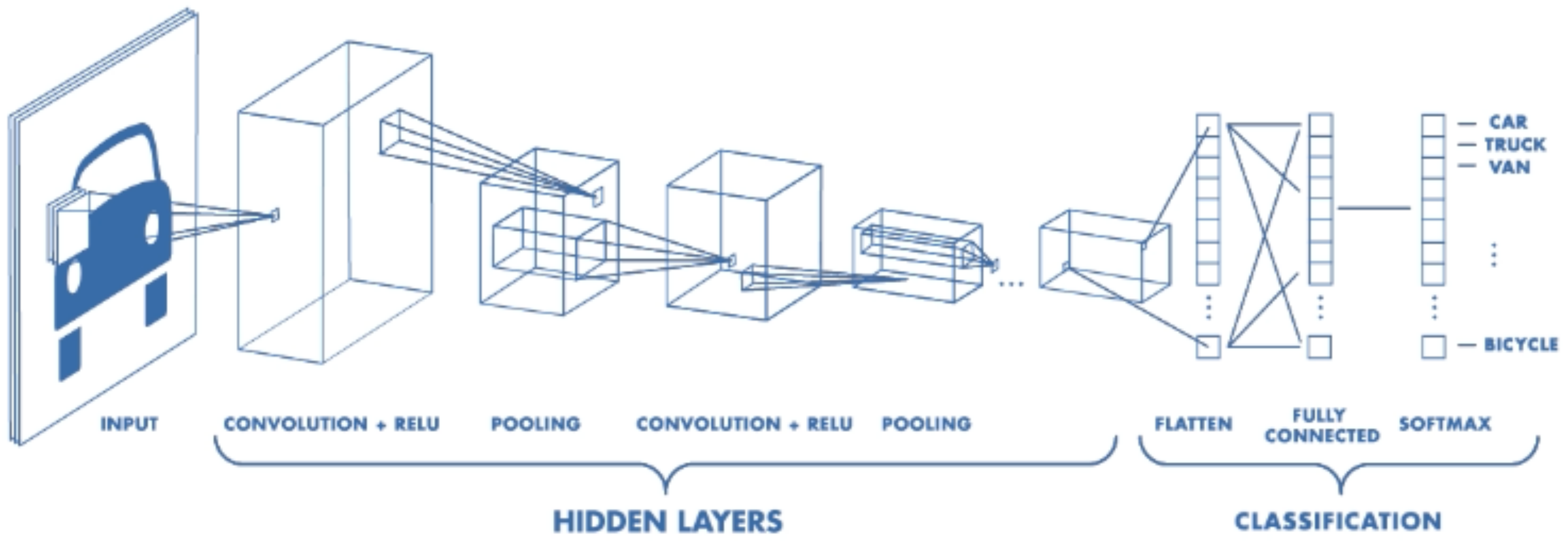
Who has heard of **Convolutional Neural Networks** (CNNs)?



Preview: Why convolutions are important?

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

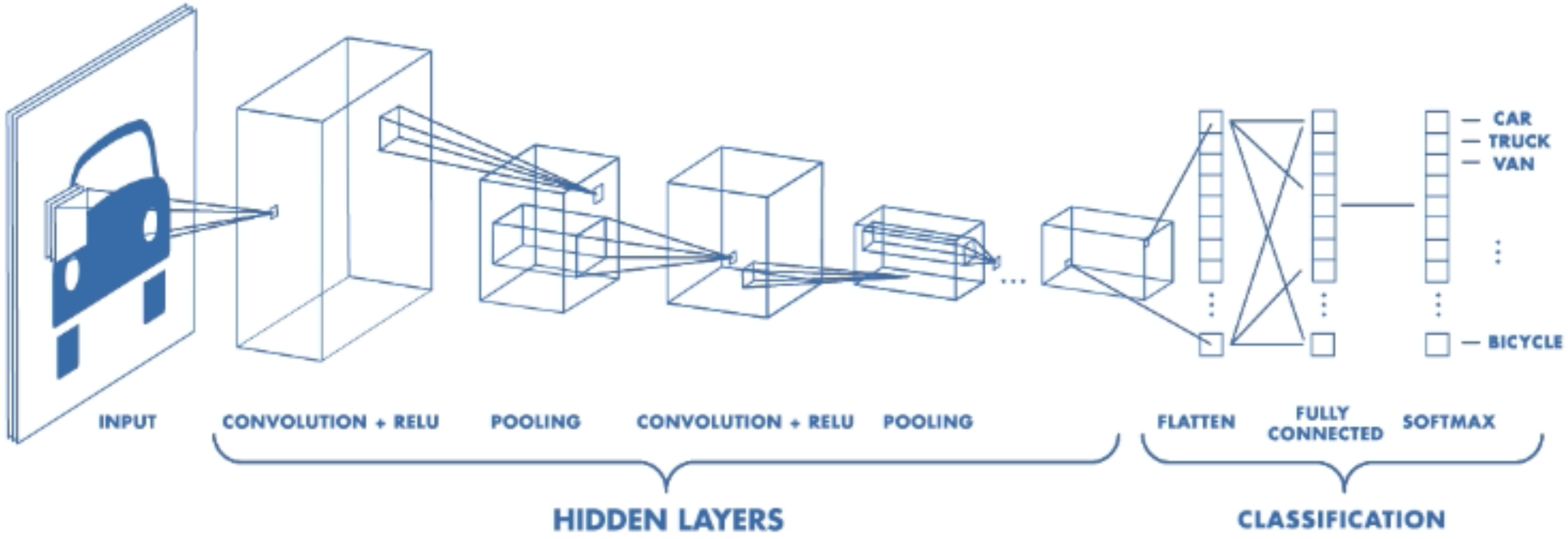
What about **Deep Learning**?



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.

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

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

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

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

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

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

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

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

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

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

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

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

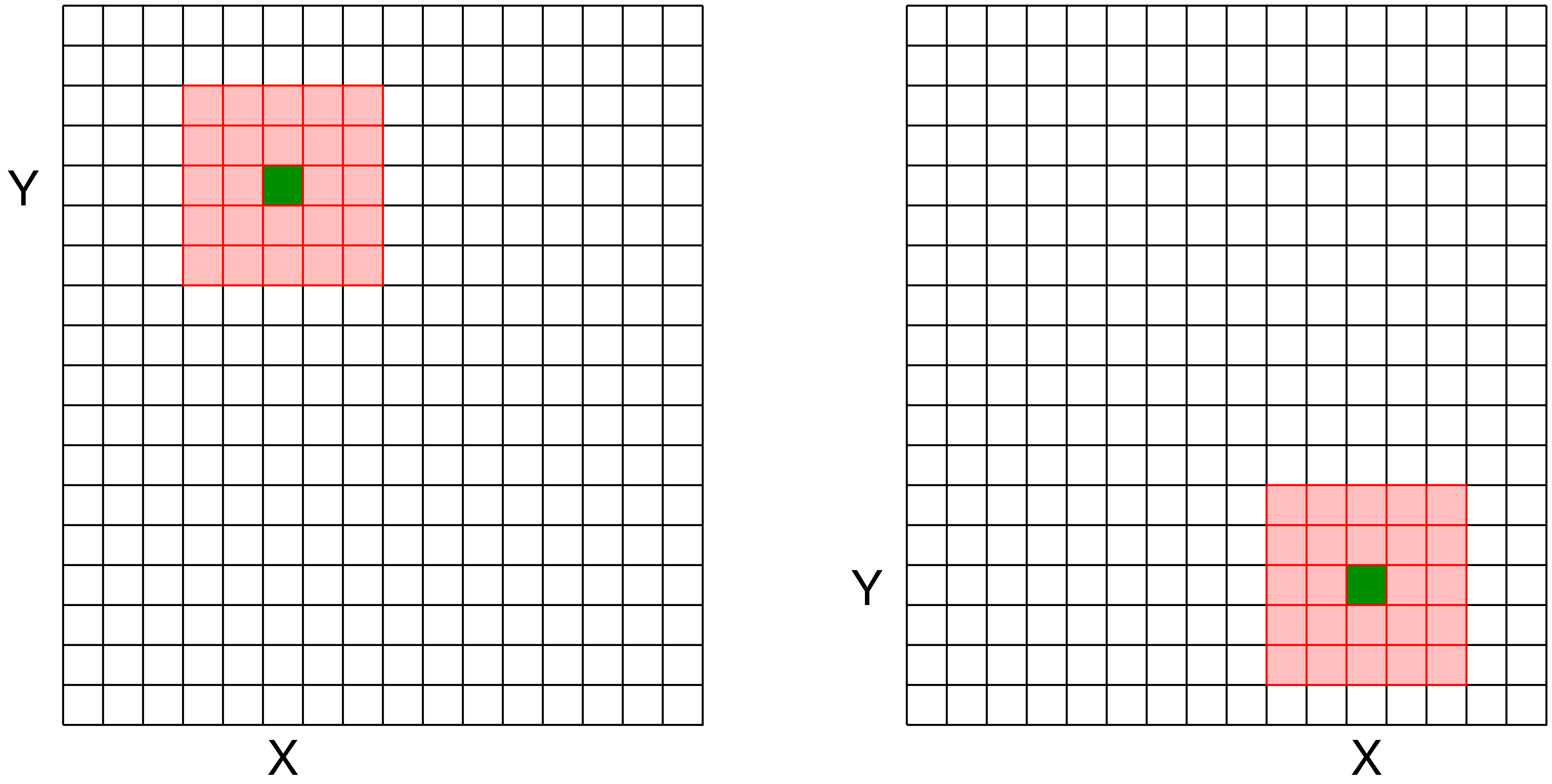
Scaling: Let F be digital filter and let k be a scalar

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

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

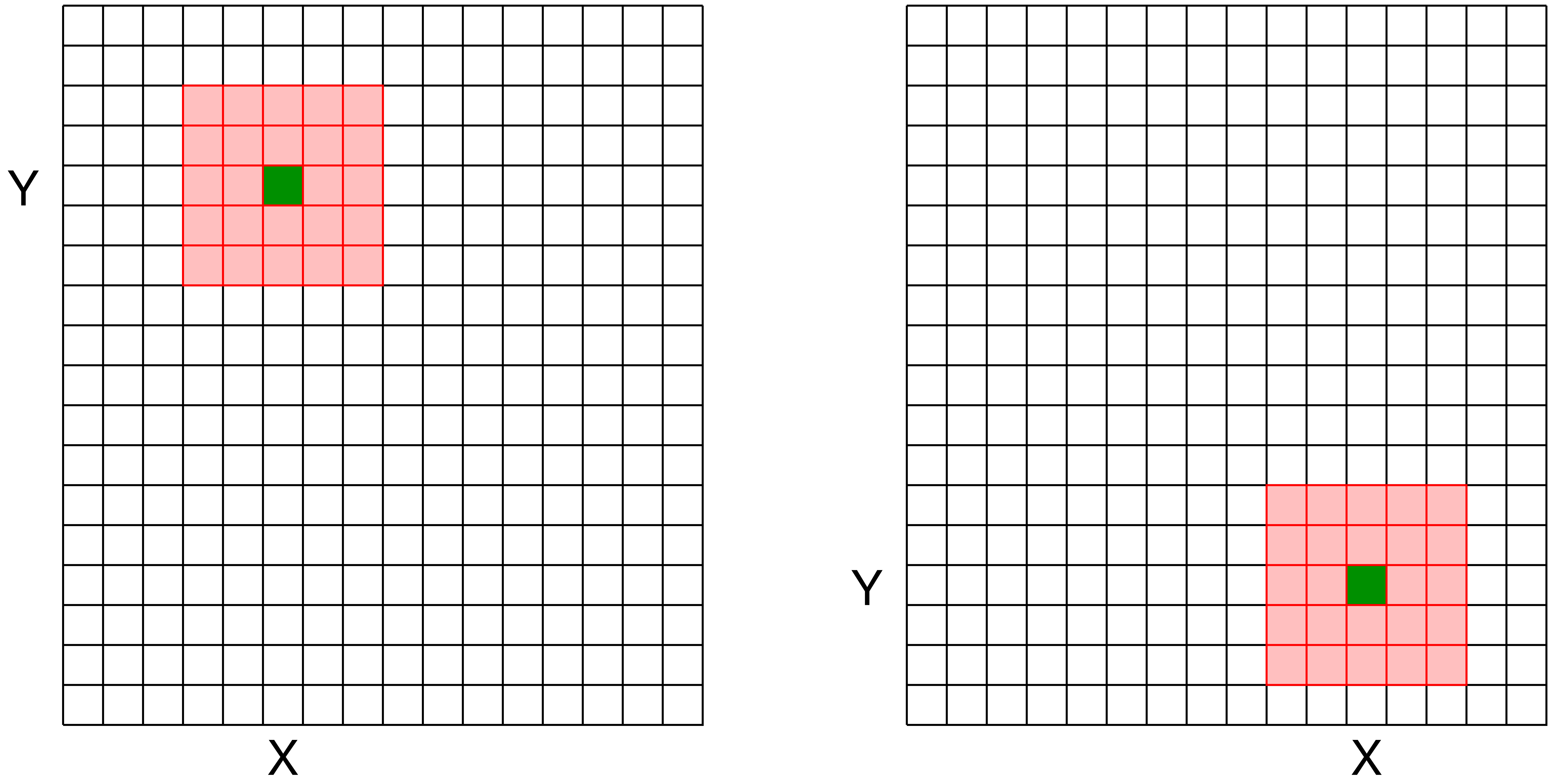
Linear Filters: Shift Invariance

Output does **not** depend on absolute position



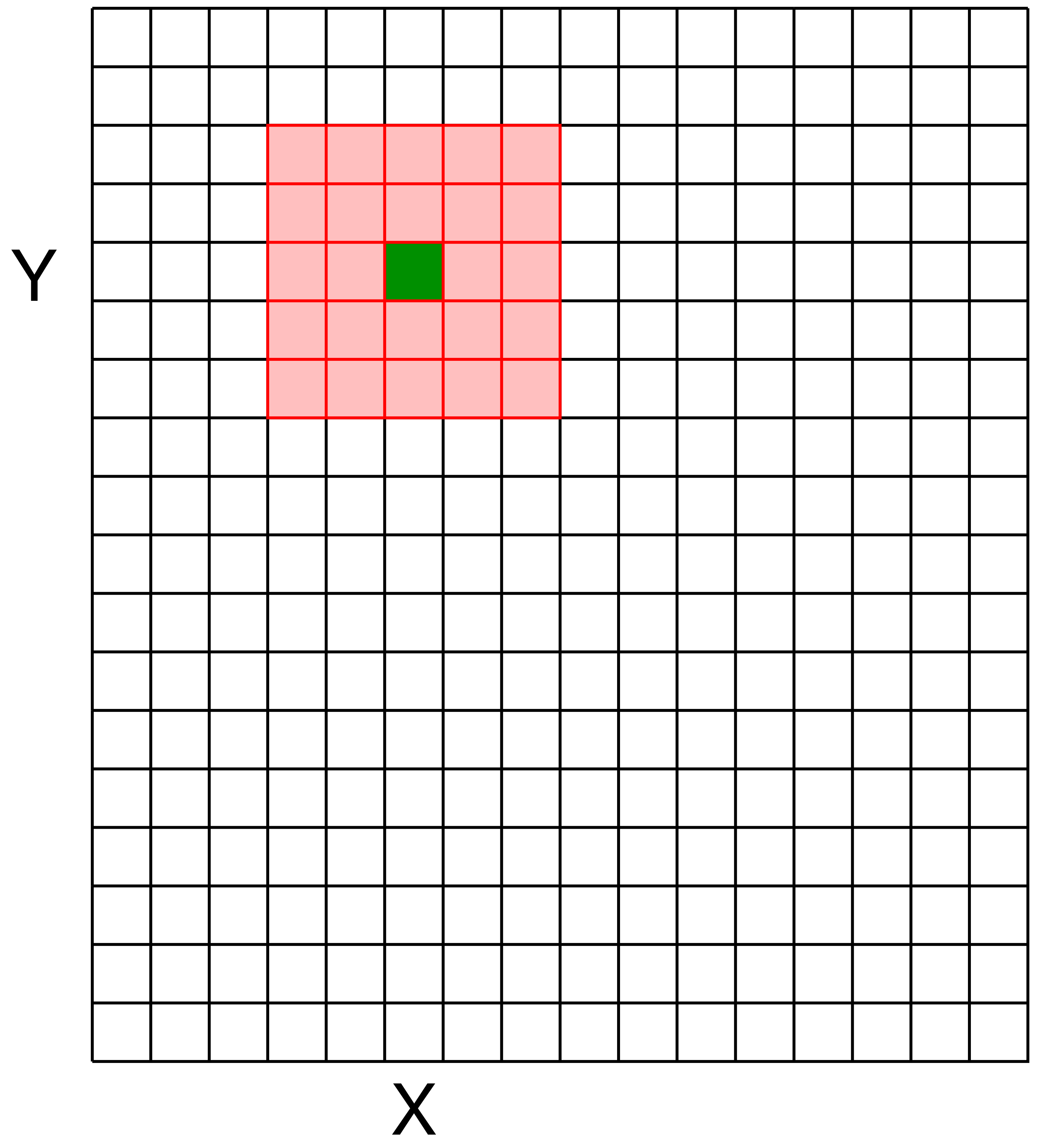
Linear Filters: Shift Invariance

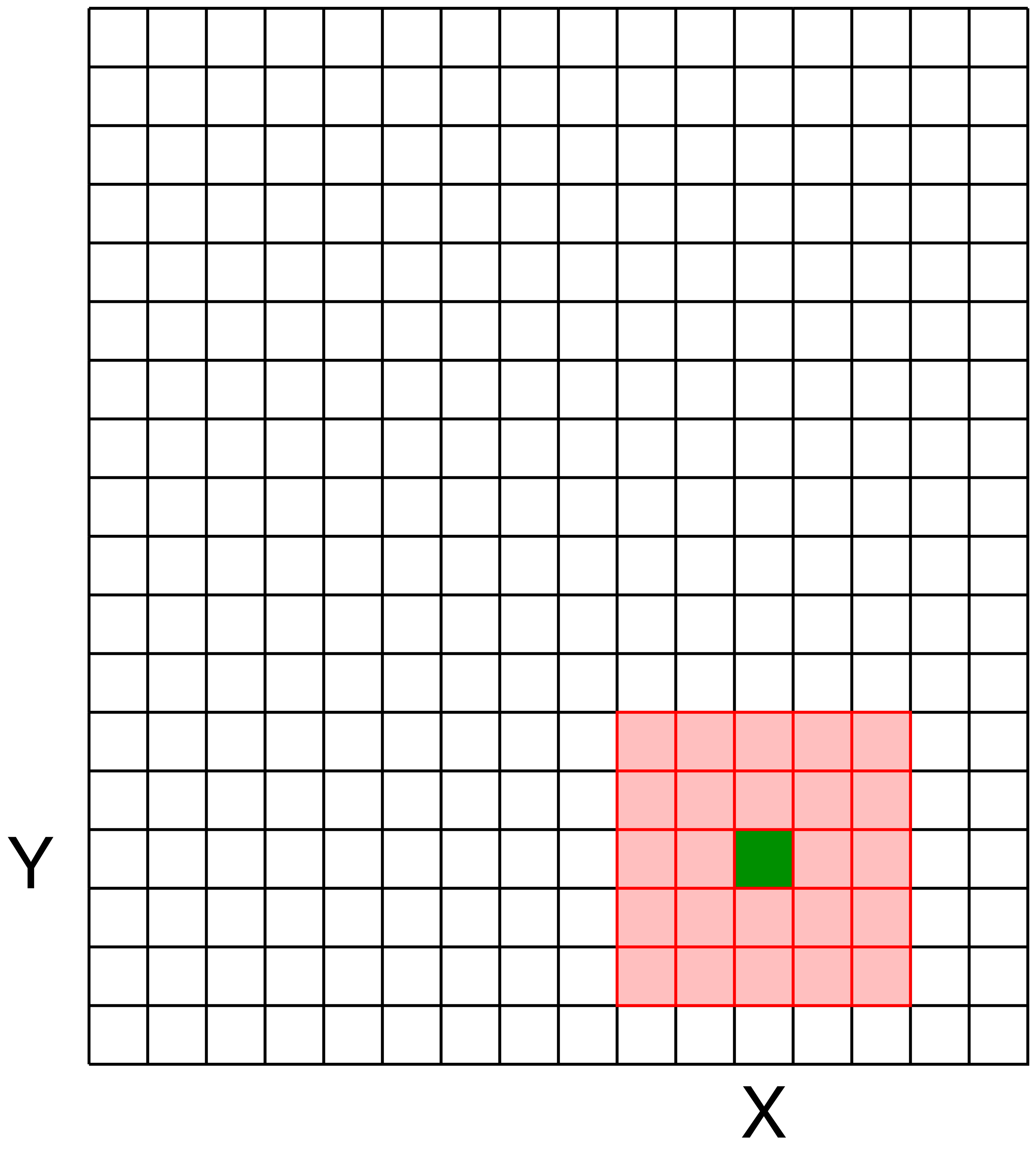
$$I'(X,Y) = f\left(F, I\left(X - \lfloor\frac{k}{2}\rfloor : X + \lfloor\frac{k}{2}\rfloor, Y - \lfloor\frac{k}{2}\rfloor : Y + \lfloor\frac{k}{2}\rfloor\right)\right)$$



Linear Filters: Shift Variant

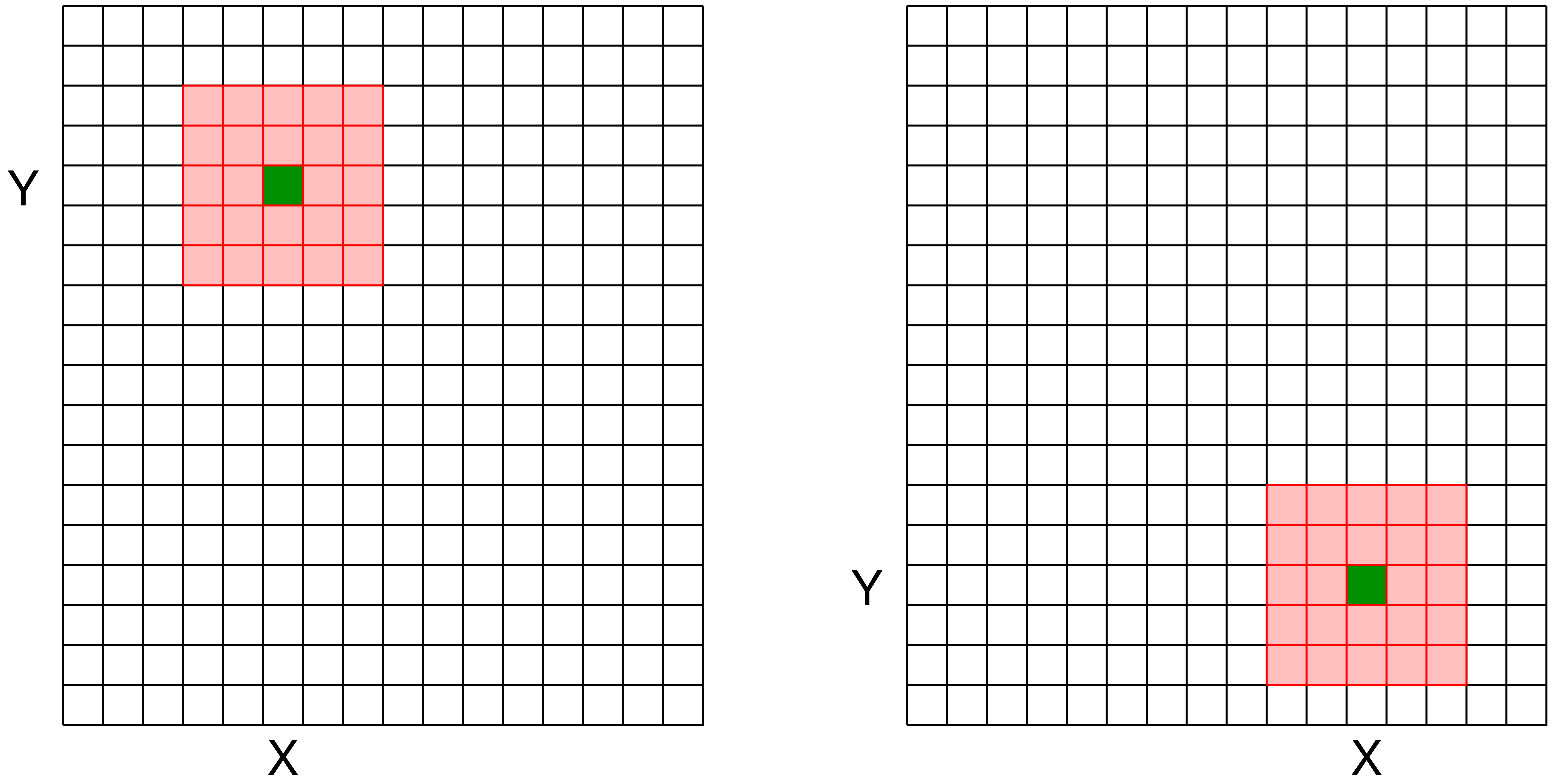
$$I'(X,Y) = f\left(F, I\left(X - \lfloor\frac{k}{2}\rfloor : X + \lfloor\frac{k}{2}\rfloor, Y - \lfloor\frac{k}{2}\rfloor : Y + \lfloor\frac{k}{2}\rfloor\right), X, Y\right)$$





Linear Filters: Shift Variant

$$I'(X,Y) = f\left(F_{X,Y}, I\left(X - \lfloor\frac{k}{2}\rfloor : X + \lfloor\frac{k}{2}\rfloor, Y - \lfloor\frac{k}{2}\rfloor : Y + \lfloor\frac{k}{2}\rfloor\right)\right)$$



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)

An operation is **linear** if it satisfies both **superposition** and **scaling**

Linear Systems: Characterization Theorem

Any linear, shift invariant operation can be expressed as 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**

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

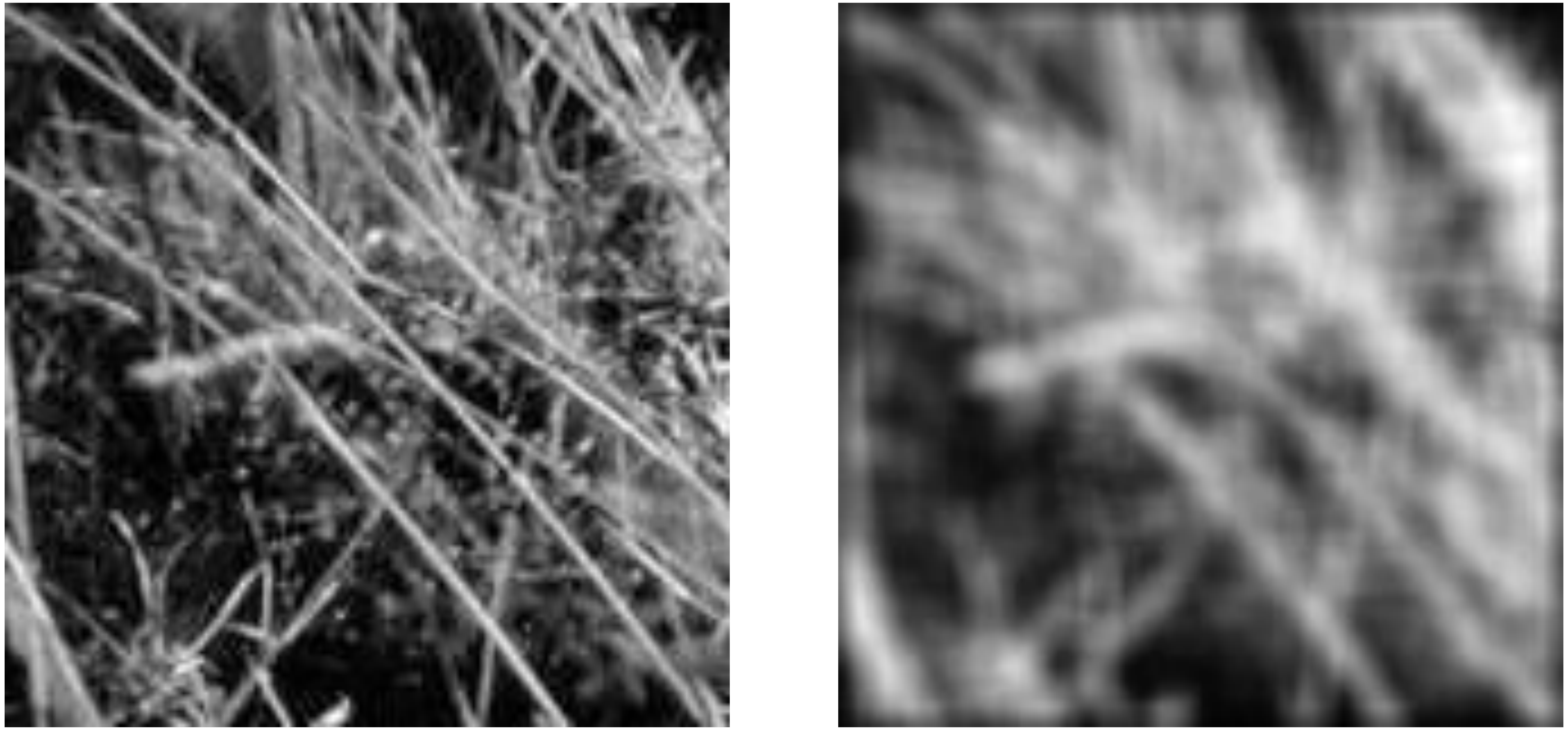
Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Filter has equal positive values that some 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**

Smoothing with a **Box Filter**



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

Smoothing with a **Box Filter**

What happens if we increase the width (size) of the box filter?

Smoothing with a **Box Filter**

Original

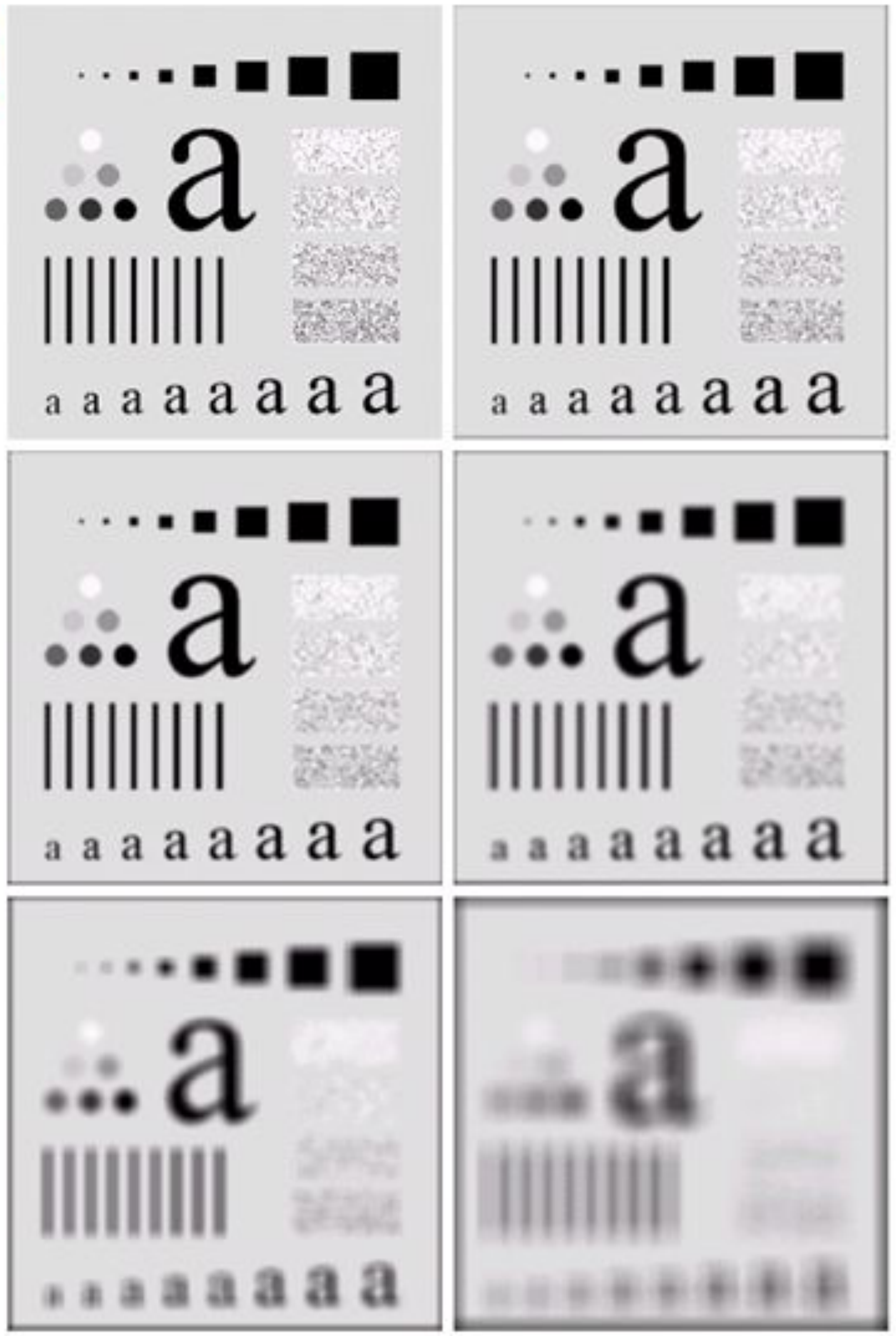
3x3

5x5

9x9

15x15

35x35



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

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

Smoothing with a **Box Filter**

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

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

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

Smoothing with a **Box Filter**

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

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

Filter

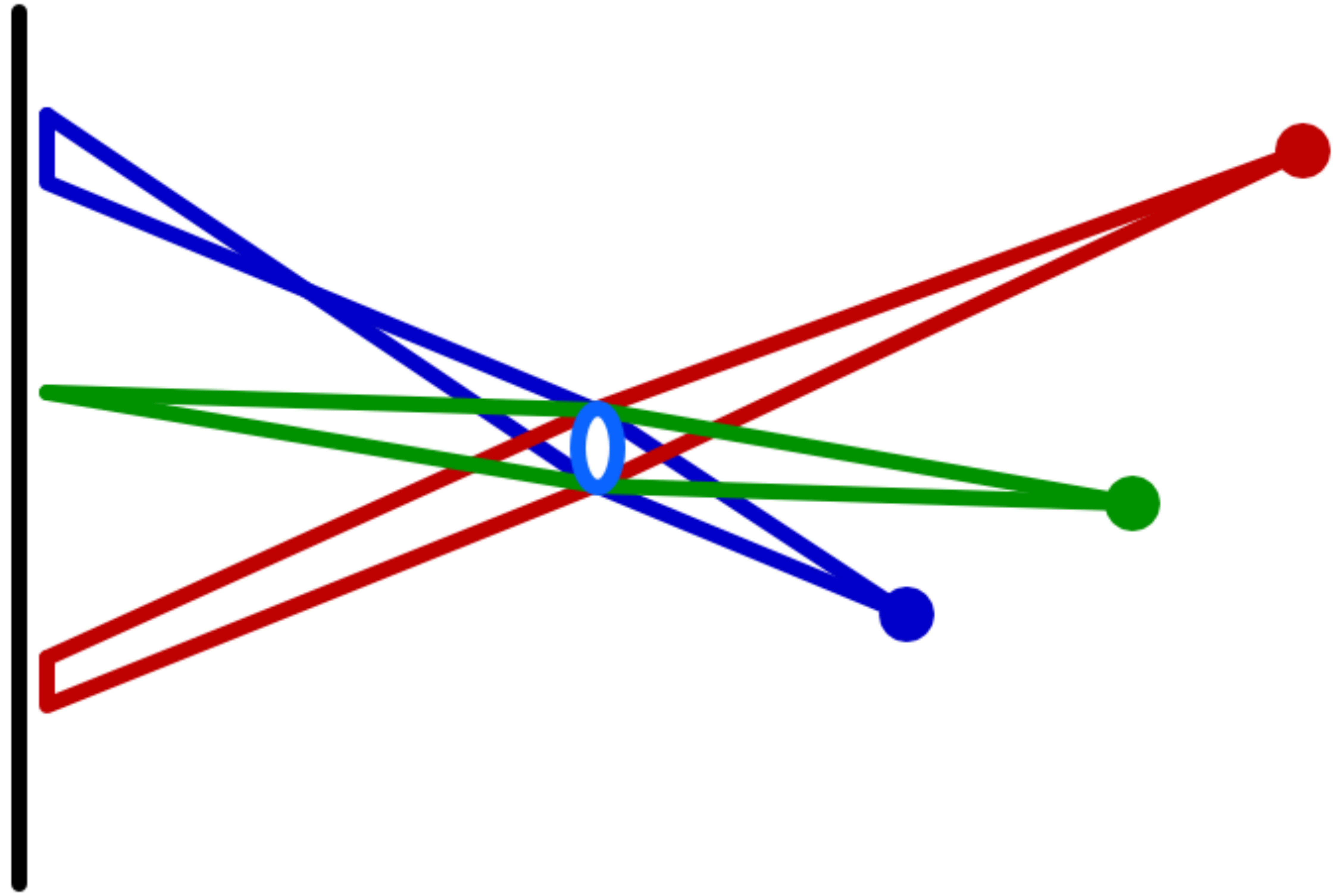
0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

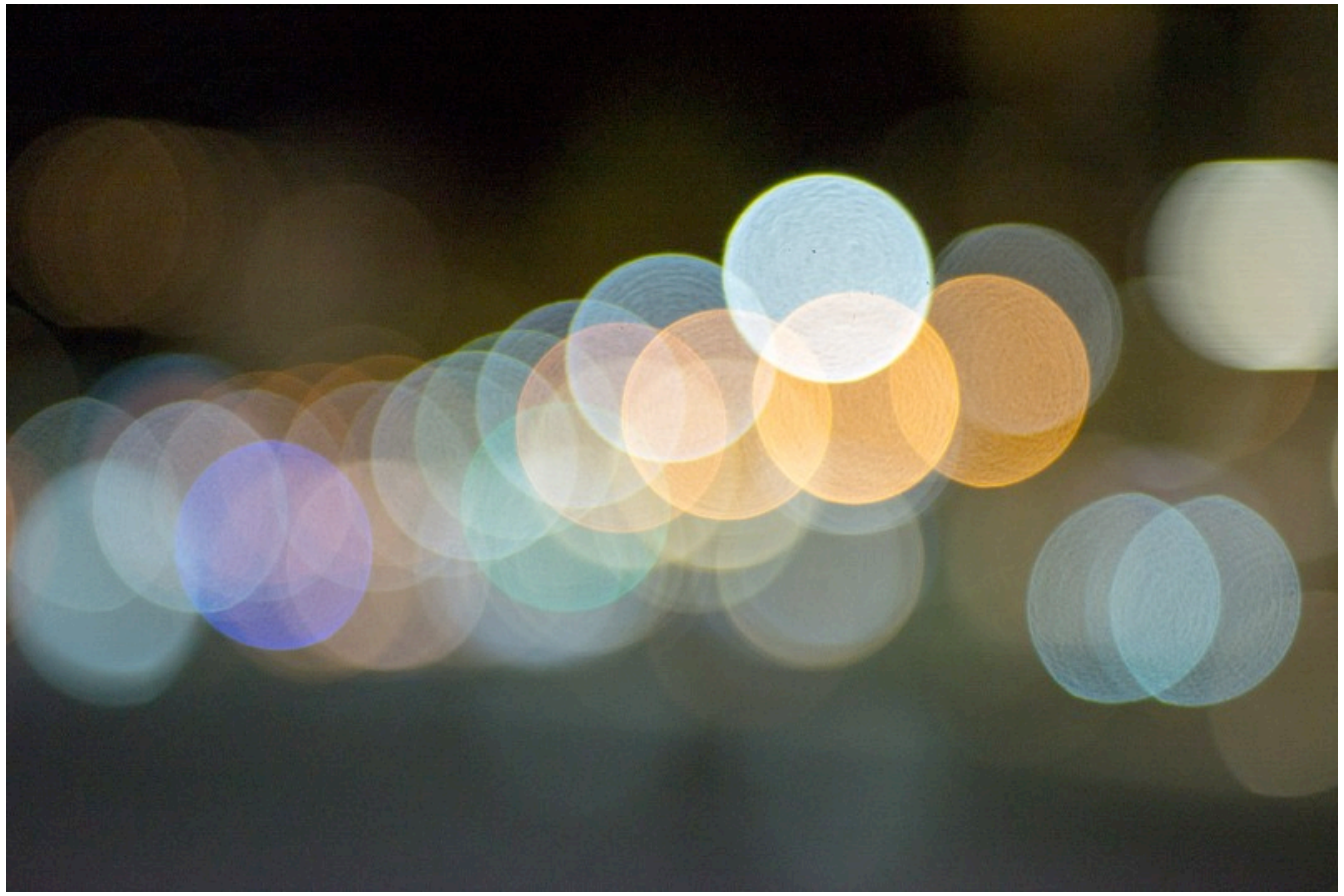
Image

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

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

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

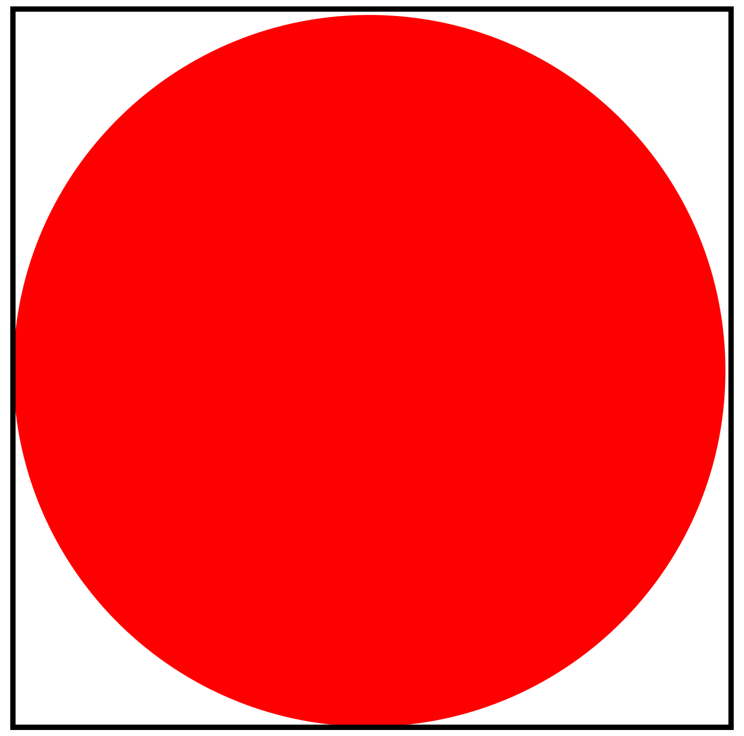
Smoothing with a (circular) **pillbox** is a better model for defocus (in geometric optics)

Pillbox Filter

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

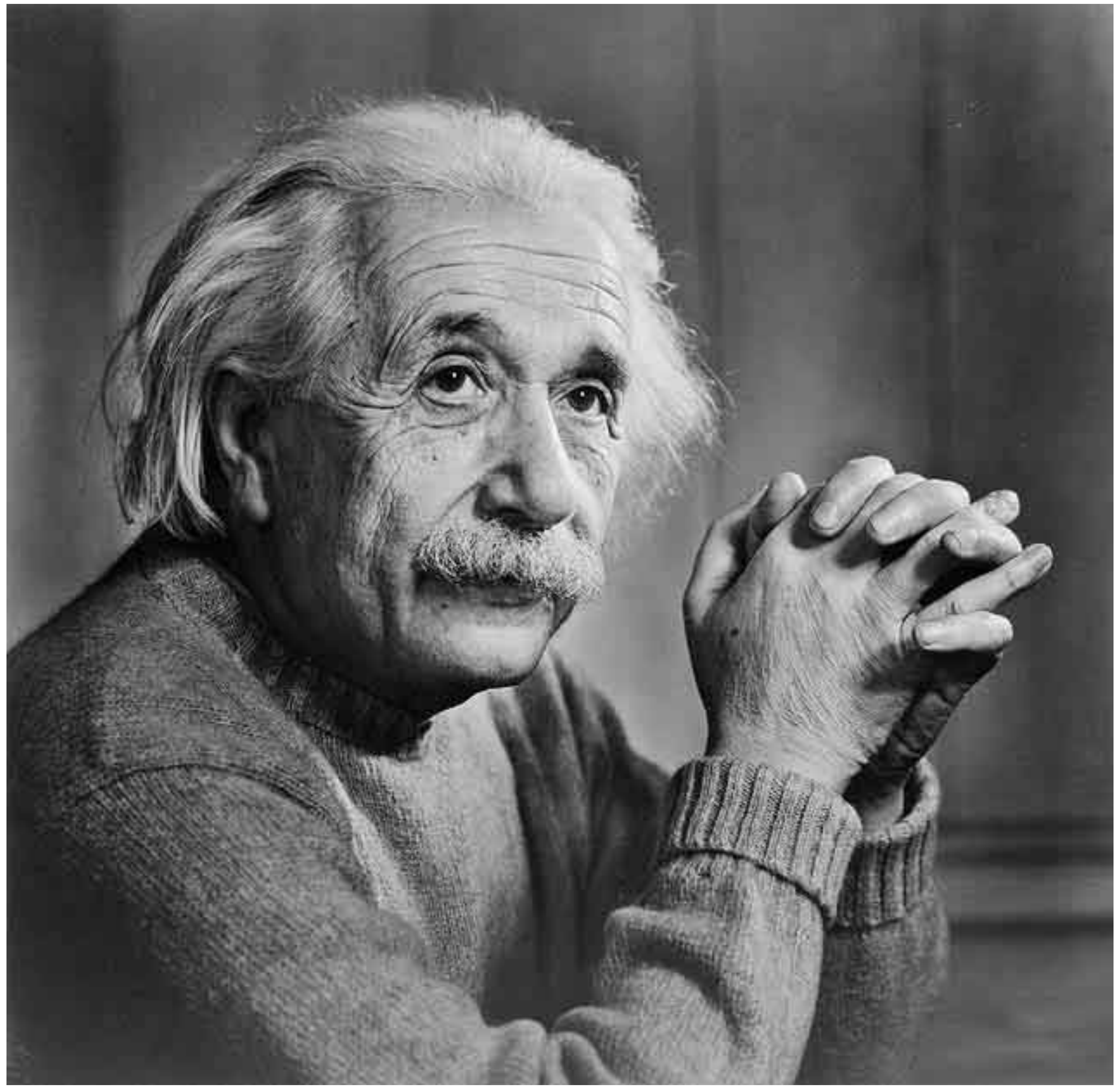
In a contentious domain, a 2D (circular) pillbox filter, $f(x, y)$, is defined as:

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

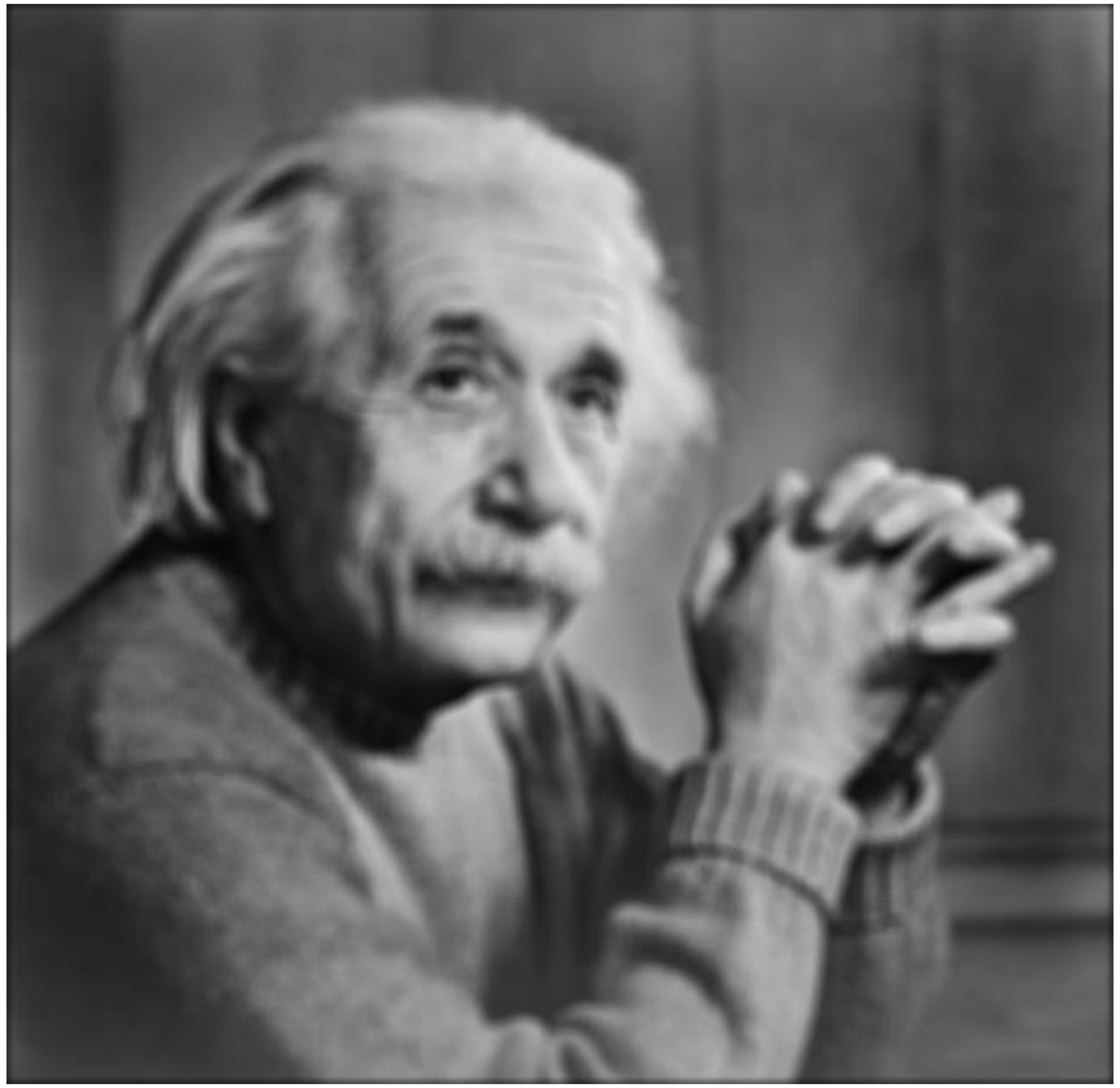


The scaling constant, $\frac{1}{\pi r^2}$, ensures that the area of the filter is one

Pillbox Filter

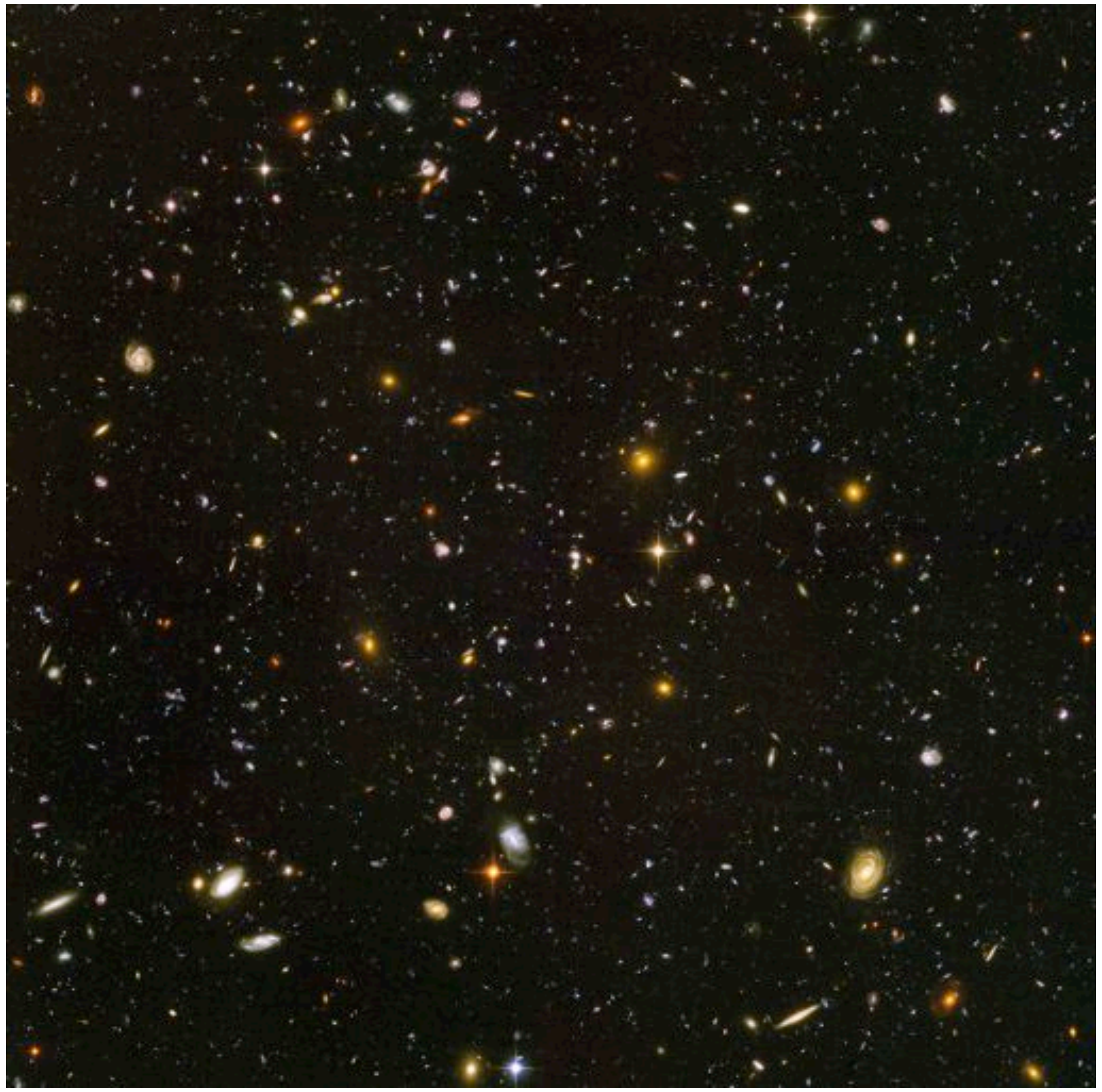


Original



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)

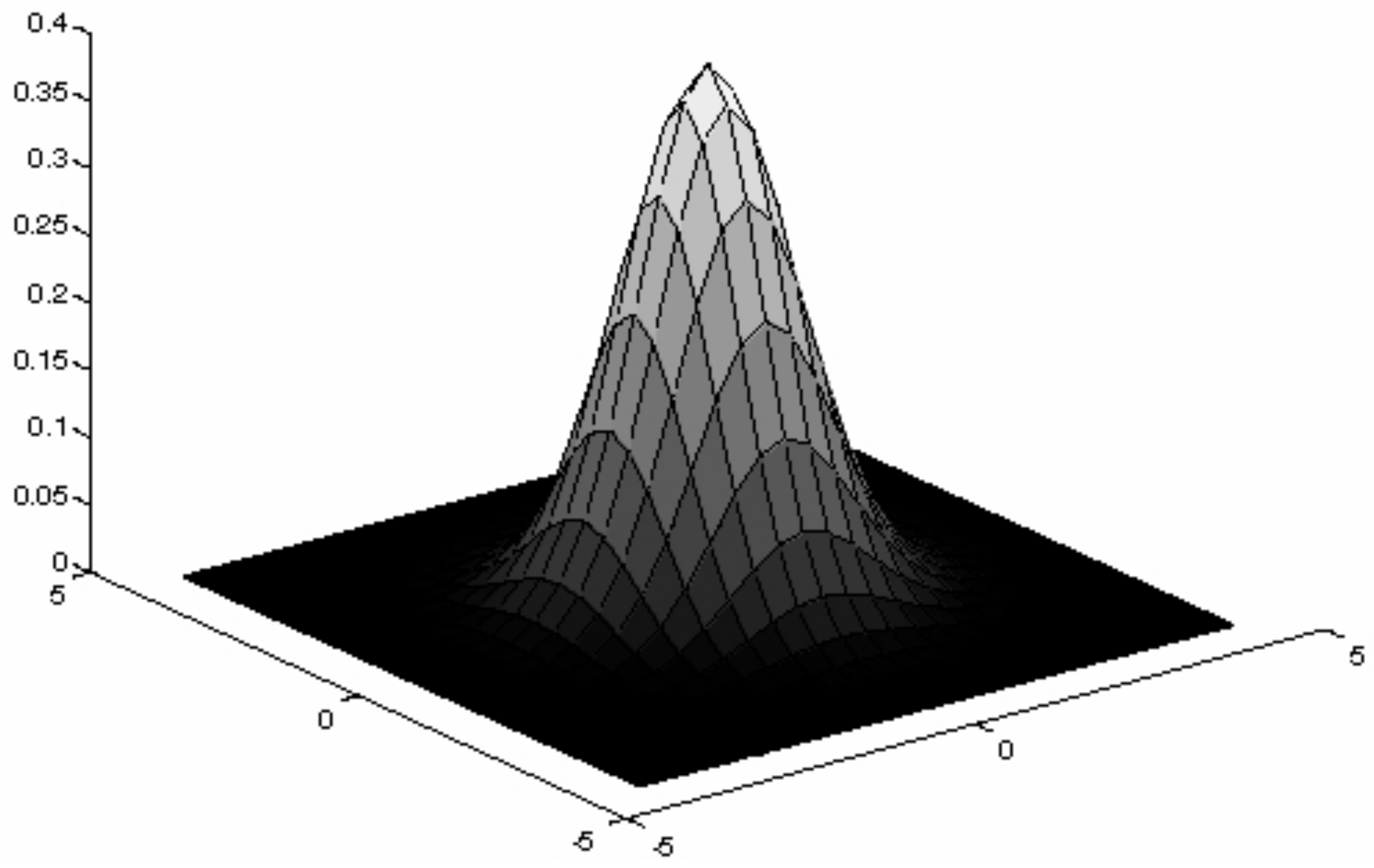
— whenever the Central Limit Theorem applies

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

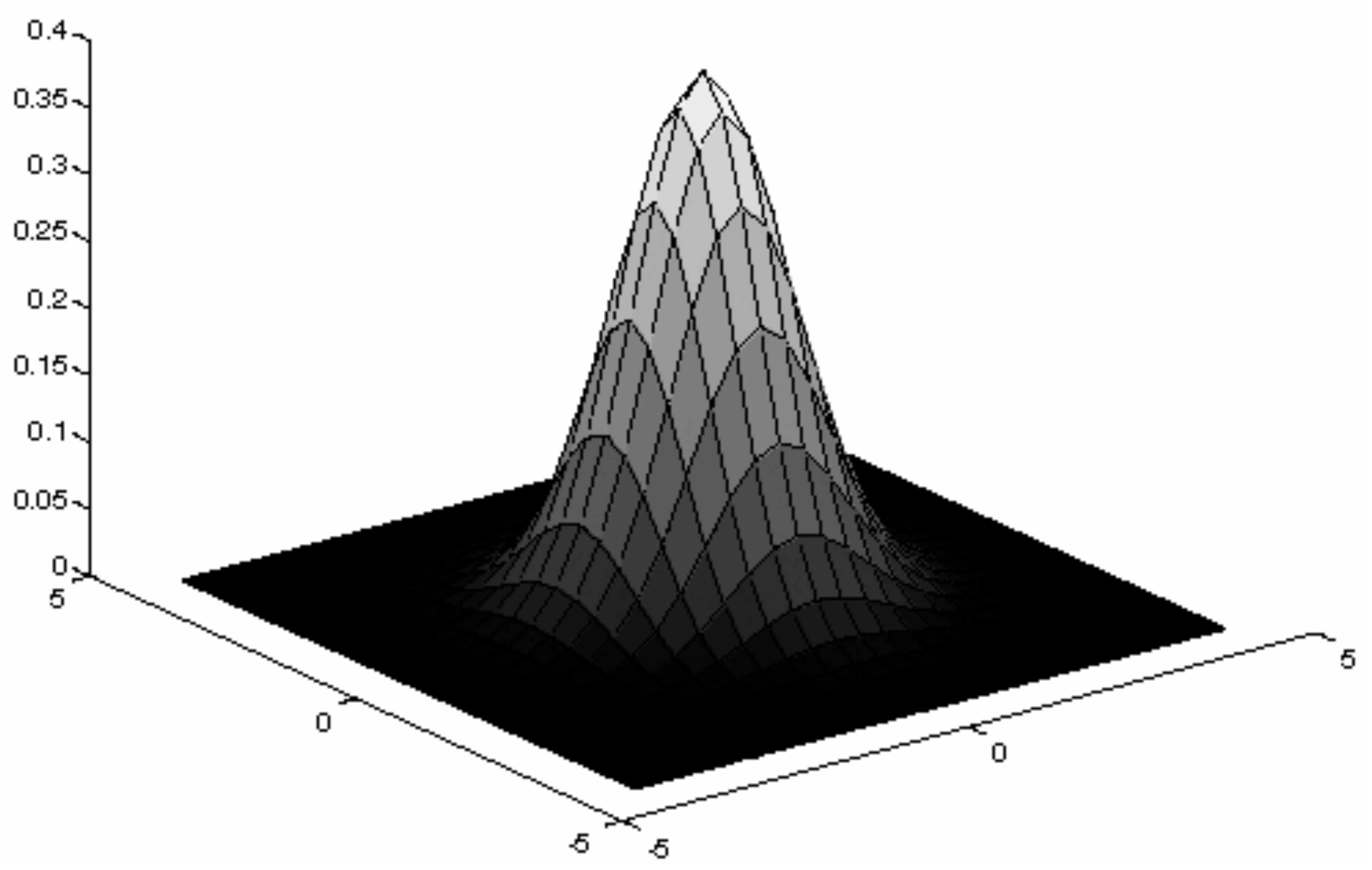
Smoothing with a **Gaussian**

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.)

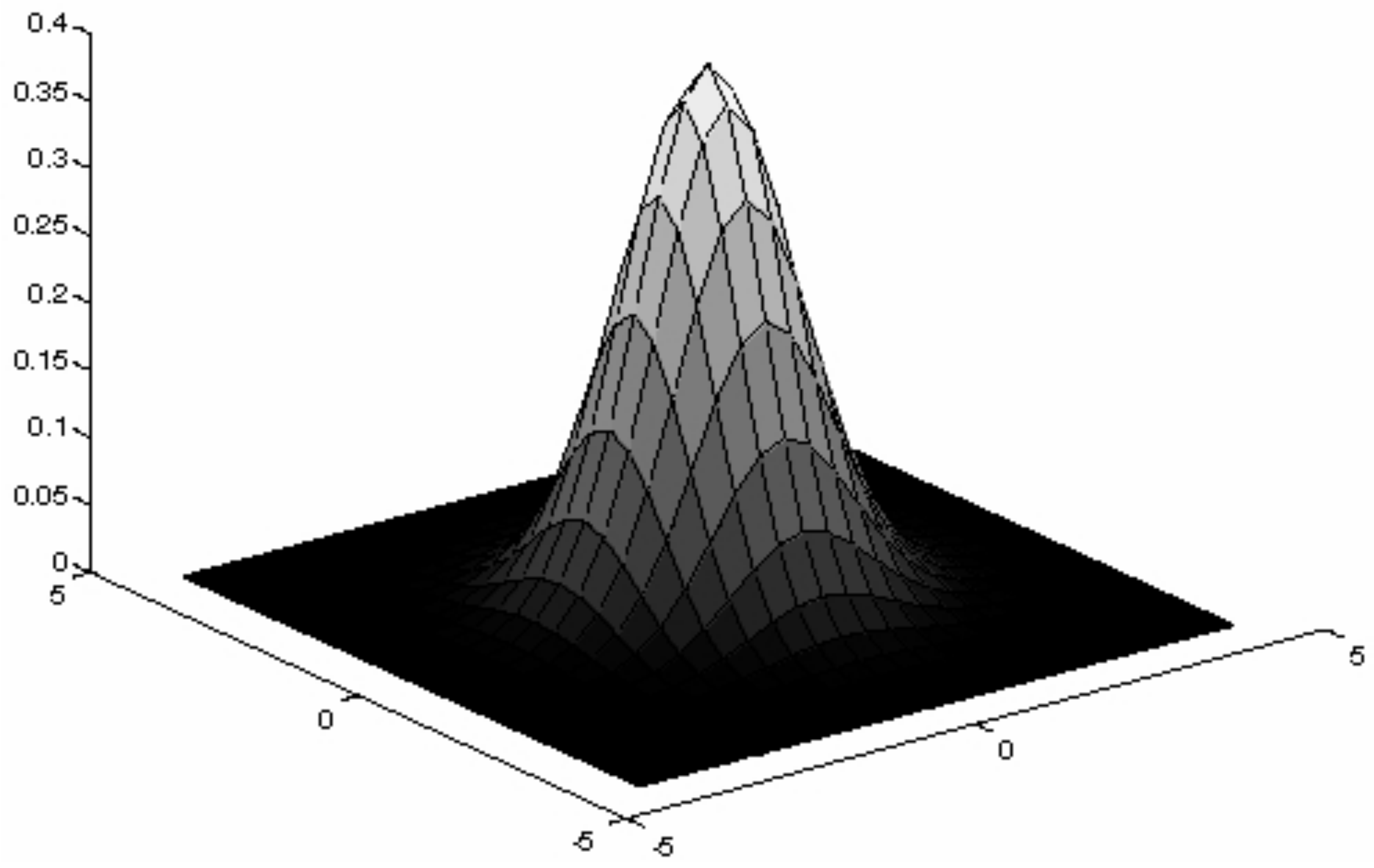
Figure 4.2

Smoothing with a **Gaussian**

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

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

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



1. Define a continuous **2D function**

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

Forsyth & Ponce (2nd ed.)
Figure 4.2

Smoothing with a **Gaussian**

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

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

Smoothing with a **Gaussian**

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

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

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

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

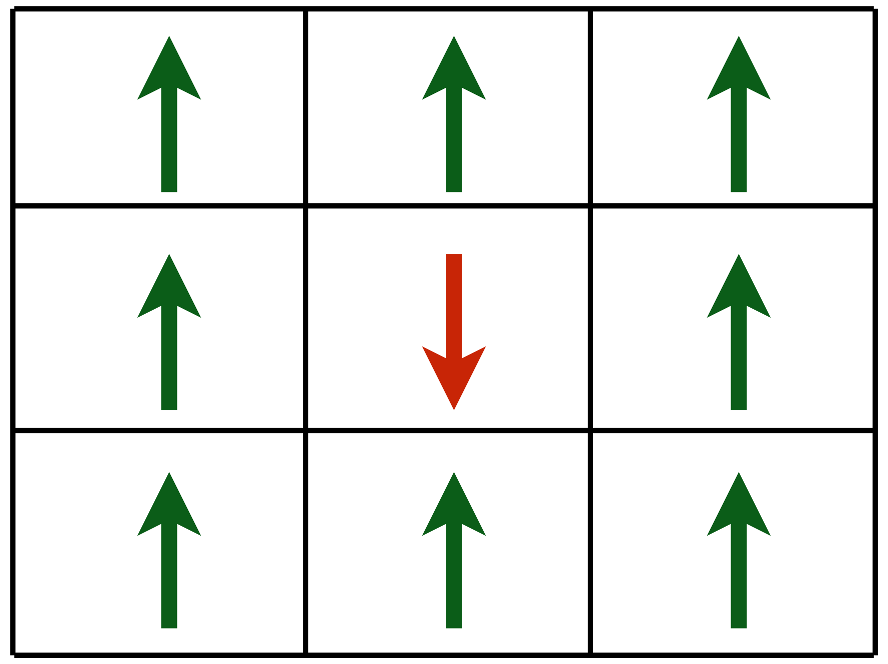
What happens if σ is larger?

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

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?

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 $\sigma = 1$:

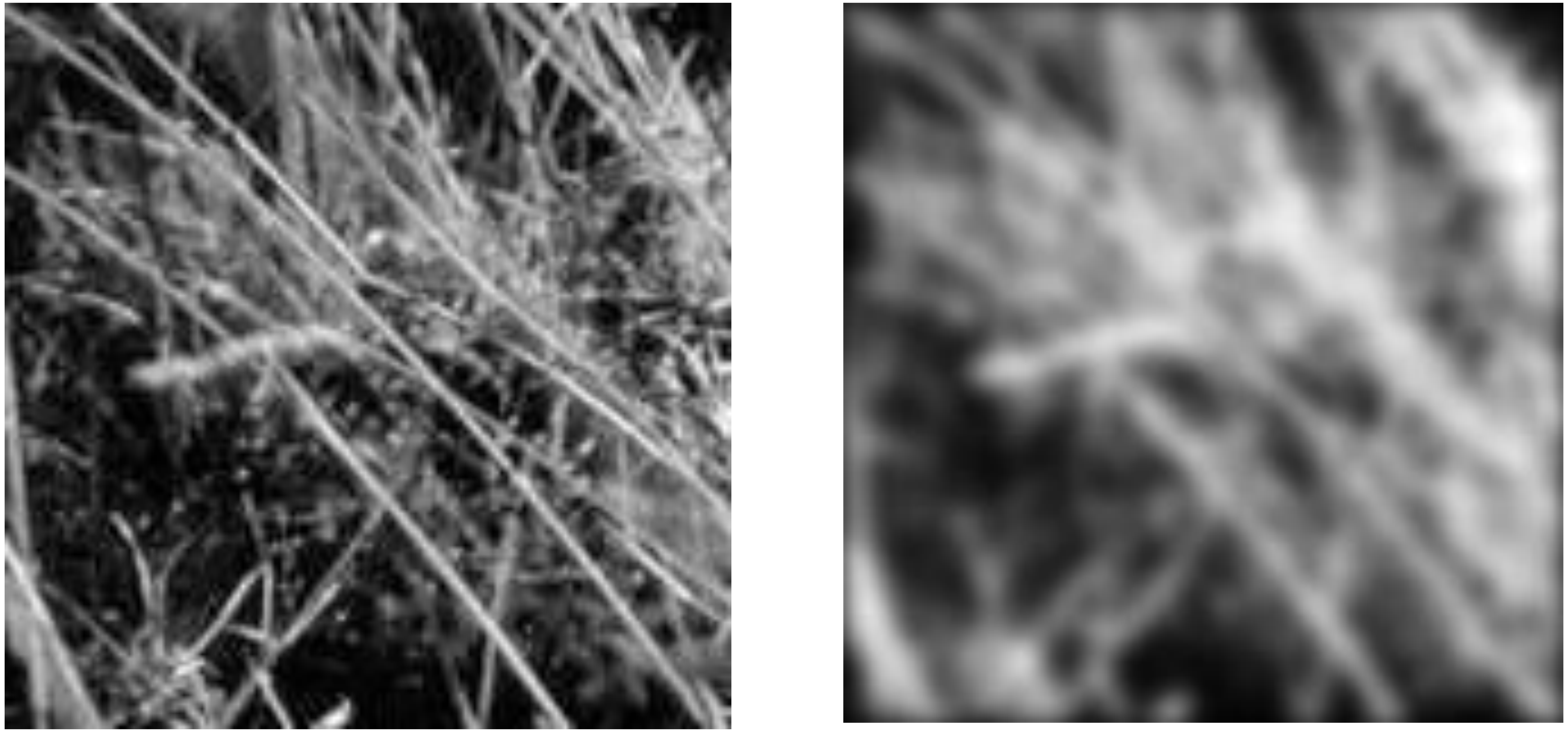
$\downarrow$	$\downarrow$	$\downarrow$
$\downarrow$	$\uparrow$	$\downarrow$
$\downarrow$	$\downarrow$	$\downarrow$

What happens if σ is larger?

What happens if σ is smaller?

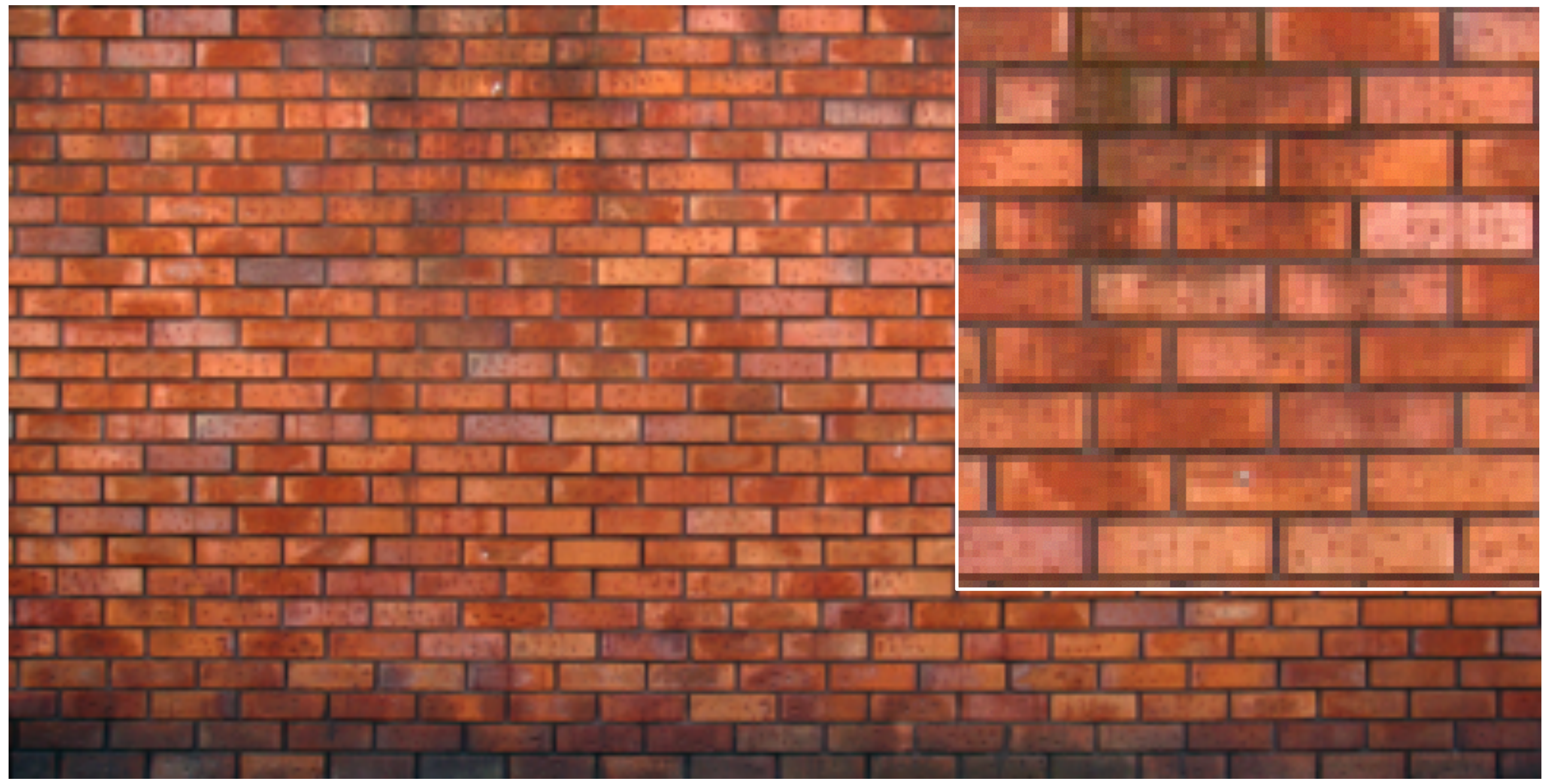
— **Less** blur

Smoothing with a **Gaussian**

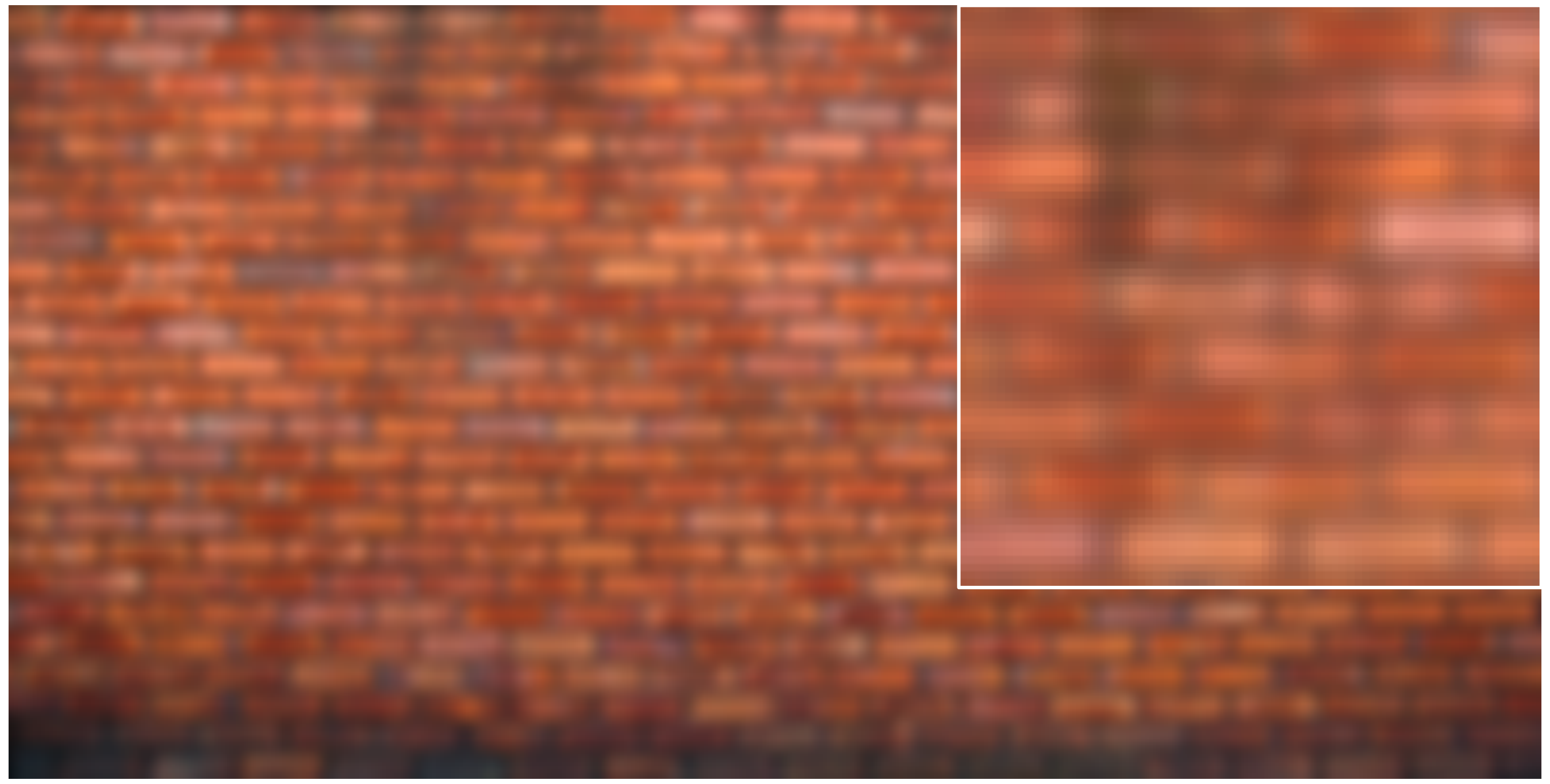


Forsyth & Ponce (2nd ed.) Figure 4.1 (left and right)

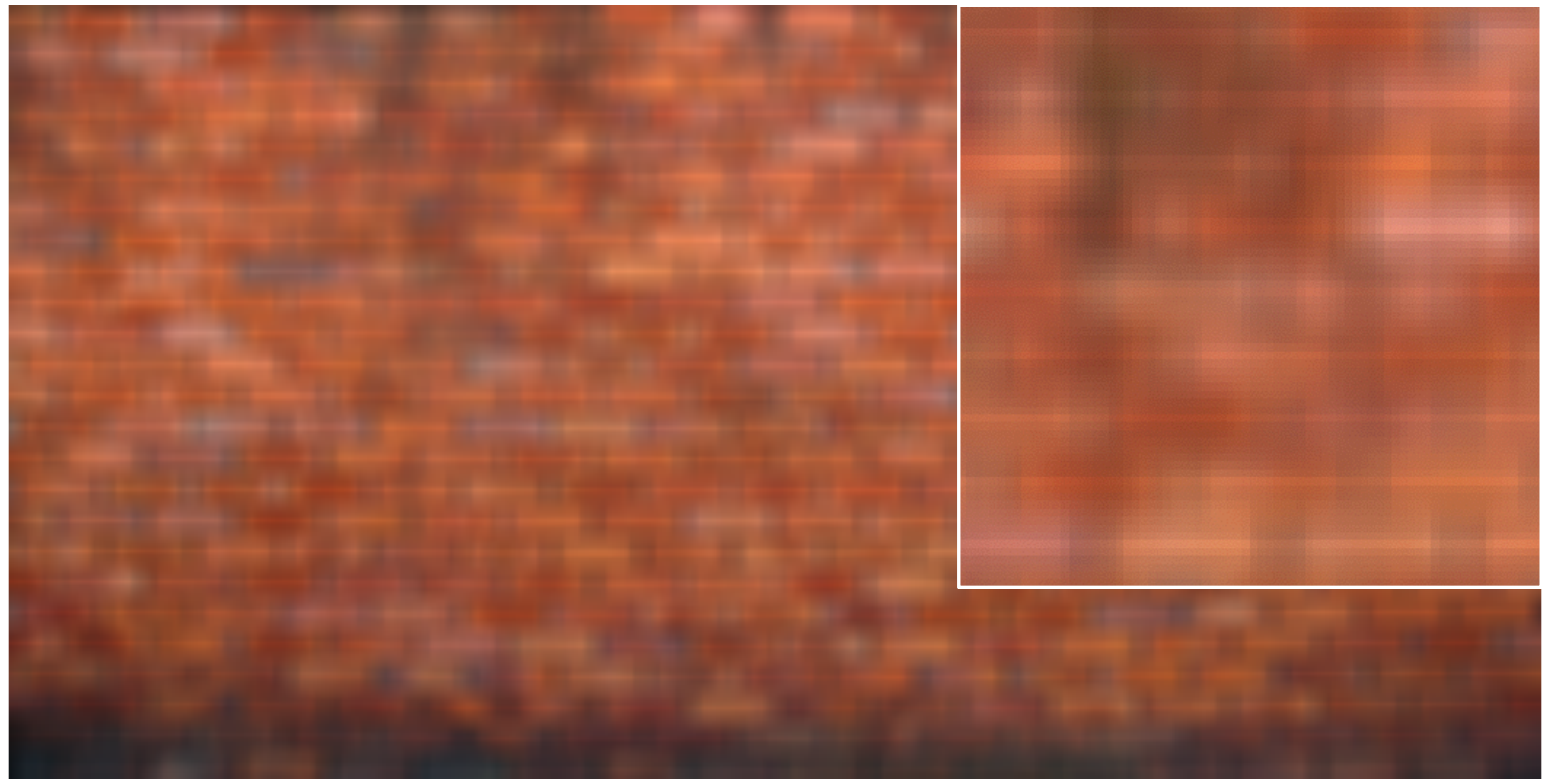
Box vs. **Gaussian** Filter



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 $\sigma = 1$:

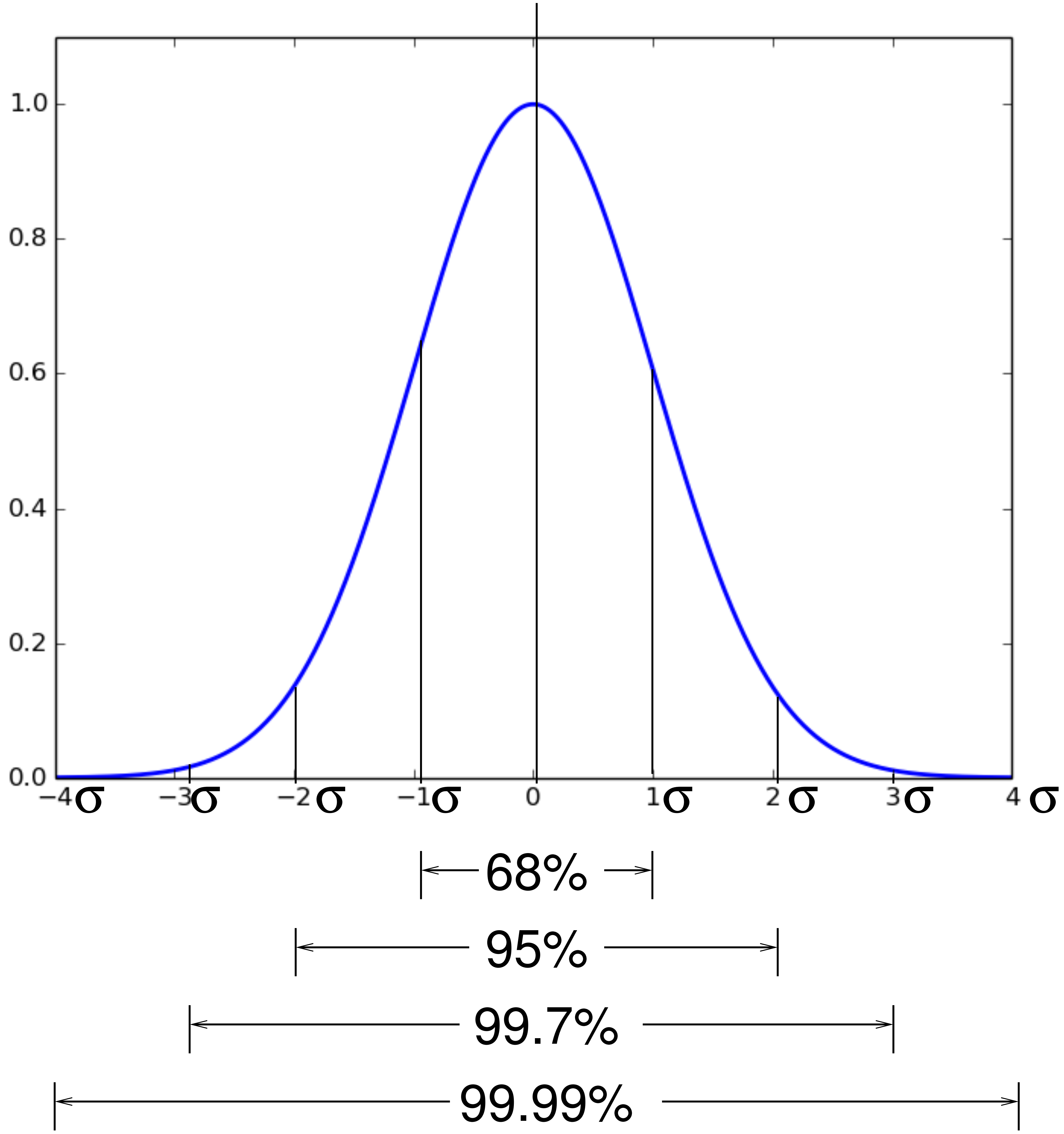
0.059	0.097	0.059
0.097	0.159	0.097
0.059	0.097	0.059

What is the problem with this filter?

does not sum to 1

truncated too much

Gaussian: Area Under the Curve



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

In general, you want the Gaussian filter to capture $\pm 3\sigma$, for $\sigma = 1$ => 7x7 filter

Exercise

With $\sigma = 5$ what filter size would be appropriate?

Exercise

With $\sigma = 5$ what filter size would be appropriate?

$$\sigma * 6 = 5 * 6 = 30 => 31 \times 31$$

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

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

	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

$$\frac{1}{3}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline \end{array}$$

	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)

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	

image $I(X,Y)$

output $I'(X,Y)$

Separable

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

	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: How do you know if filter is separable?

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

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \frac{1}{3}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline \end{array} \odot \frac{1}{3}\begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline 1 \\ \hline \end{array}$$

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**:

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

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

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**:

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

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

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

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

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

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

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

$$\frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

(a) box, $K = 5$

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

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

(b) bilinear

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

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

(c) “Gaussian”

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

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

(d) Sobel

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

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

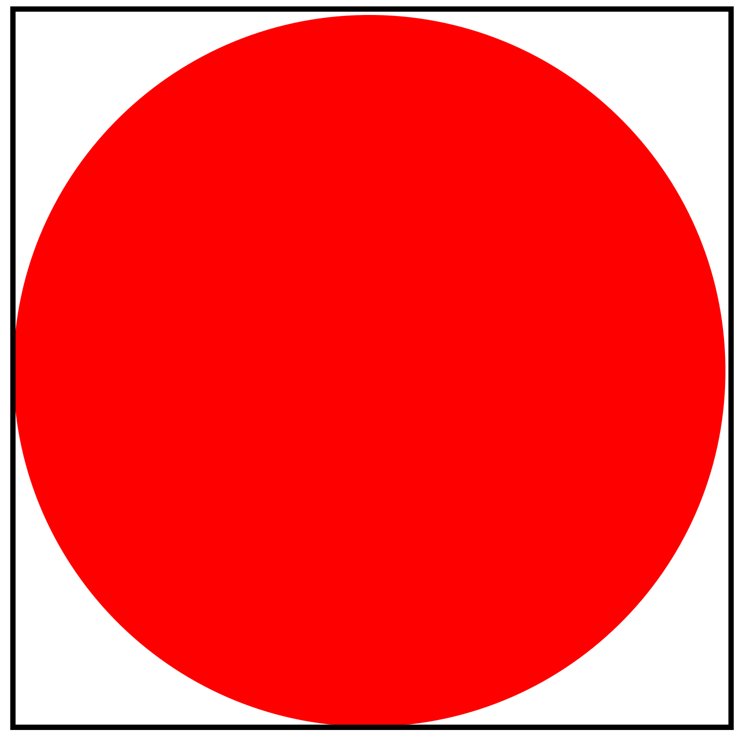
(e) corner

Smoothing with a **Pillbox**

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

Smoothing with a **Pillbox**

Recall that 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.

There are occasions when we want to convolve an image with a 2D pillbox. Thus, it worth exploring possibilities for **efficient implementation**.

Speeding Up **Convolution** (The Convolution Theorem)

Let z be the product of two numbers, x and y, that is,

$$z = xy$$

Speeding Up **Convolution** (The Convolution Theorem)

Let z be the product of two numbers, x and y, that is,

$$z = xy$$

Taking logarithms of both sides, one obtains

$$\ln z = \ln x + \ln y$$

Speeding Up **Convolution** (The Convolution Theorem)

Let z be the product of two numbers, x and y, that is,

$$z = xy$$

Taking logarithms of both sides, one obtains

$$\ln z = \ln x + \ln y$$

Therefore

$$z = \exp^{\ln z} = \exp^{(\ln x + \ln y)}$$

Speeding Up **Convolution** (The Convolution Theorem)

Let z be the product of two numbers, x and y, that is,

$$z = xy$$

Taking logarithms of both sides, one obtains

$$\ln z = \ln x + \ln y$$

Therefore.

$$z = \exp^{\ln z} = \exp^{(\ln x + \ln y)}$$

Interpretation: At the expense of two $\ln()$ and one $\exp()$ computations, multiplication is reduced to admission

Speeding Up **Rotation**

Another analogy: **2D rotation of a point by an angle** α about the origin

The standard approach, in Euclidean coordinates, involves a matrix multiplication

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

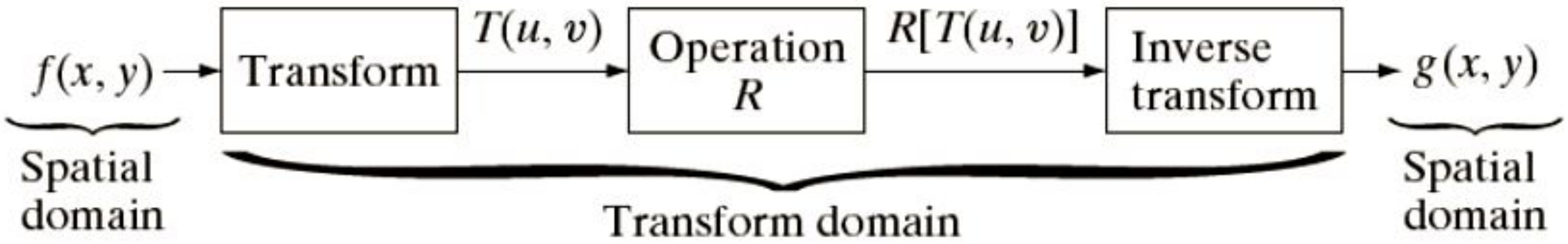
Suppose we transform to polar coordinates

$$(x, y) \to (\rho, \theta) \to (\rho, \theta + \alpha) \to (x', y')$$

Rotation becomes addition, at expense of one polar coordinate transform and one inverse polar coordinate transform

Speeding Up **Convolution** (The Convolution Theorem)

Similarly, some image processing operations become cheaper in a transform domain



Gonzales & Woods (3rd ed.) Figure 2.39

Speeding Up **Convolution** (The Convolution Theorem)

Convolution **Theorem**:

Let $$i'(x,y) = f(x,y) \otimes i(x,y)$$

then $$\mathcal{I}'(w_x, w_y) = \mathcal{F}(w_x, w_y)\, \mathcal{I}(w_x, w_y)$$

where $\mathcal{I}'(w_x, w_y)$, $\mathcal{F}(w_x, w_y)$, and $\mathcal{I}(w_x, w_y)$ are Fourier transforms of $i'(x,y)$, $f(x,y)$ and $i(x,y)$

At the expense of two **Fourier** transforms and one inverse Fourier transform, convolution can be reduced to (complex) multiplication

Lets take a **detour** …

What follows is for fun

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

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

Basic building block:

$$A \sin(\omega x + \phi)$$

Fourier's claim: Add enough of these to get <u>any</u> periodic signal you want!

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

Basic building block:

$$A \sin(\omega x + \phi)$$

amplitude

sinusoid

angular frequency

variable

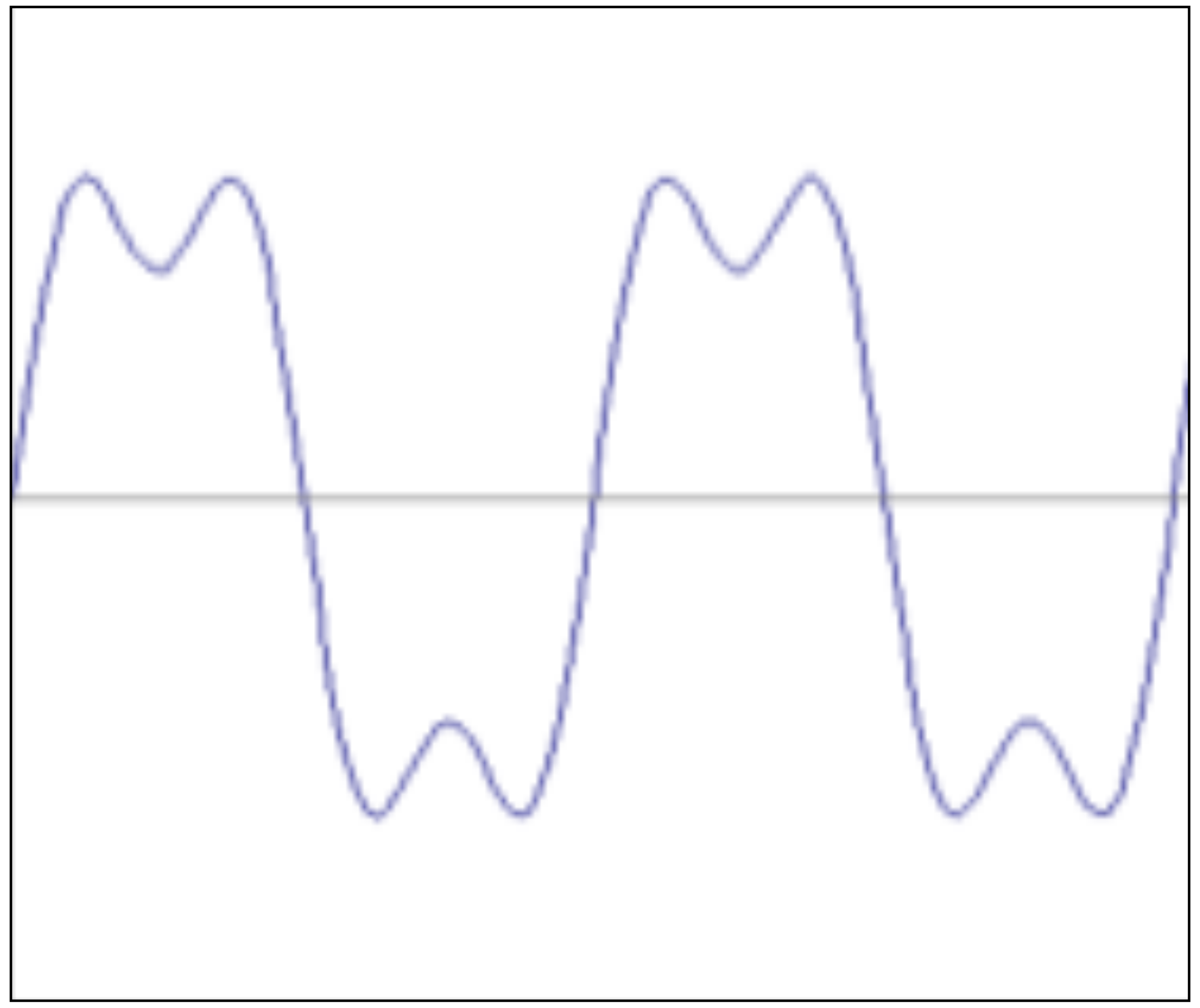
phase

Fourier's claim: Add enough of these to get <u>any</u> periodic signal you want!

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

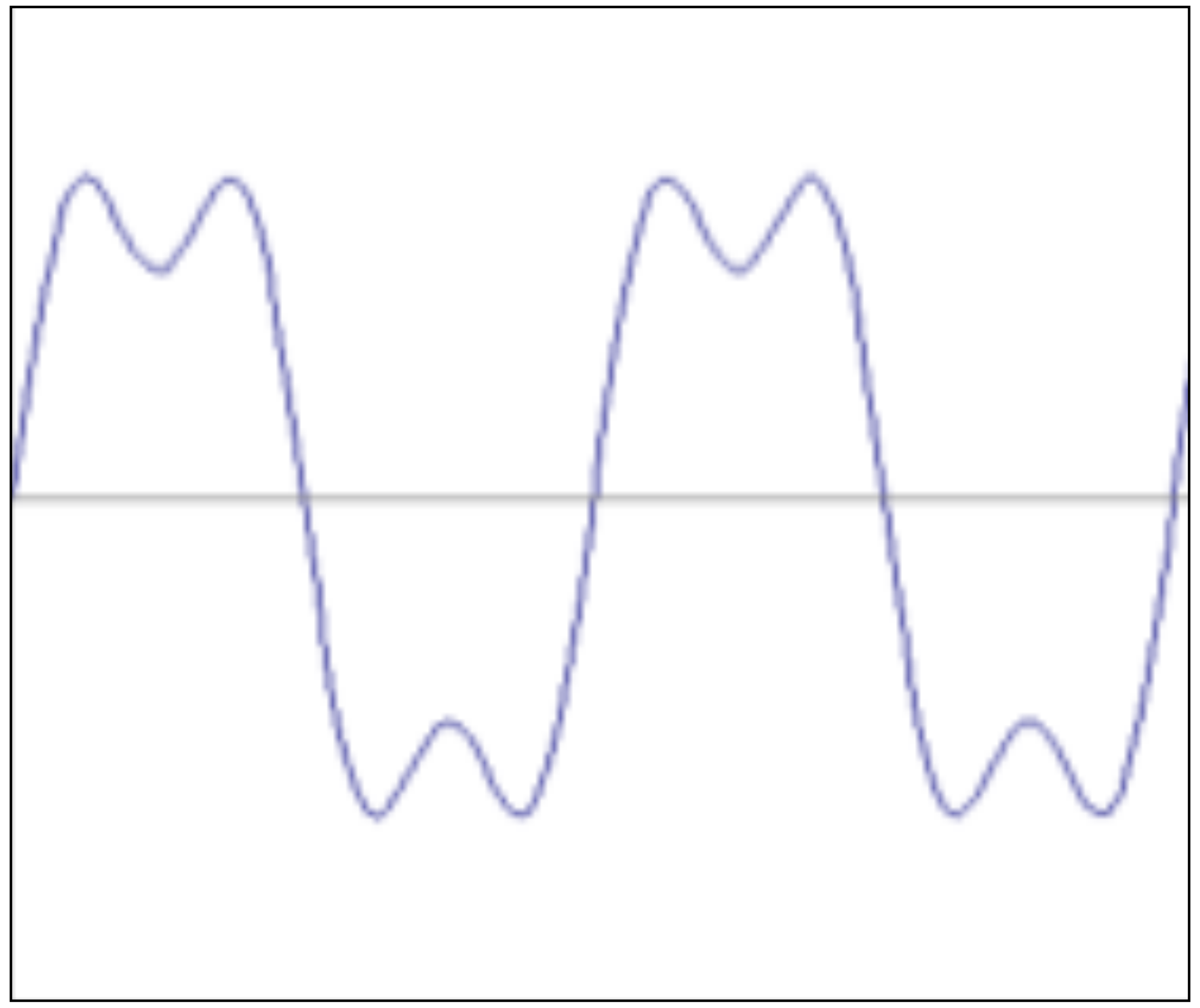
How would you generate this function?

 = ? + ?

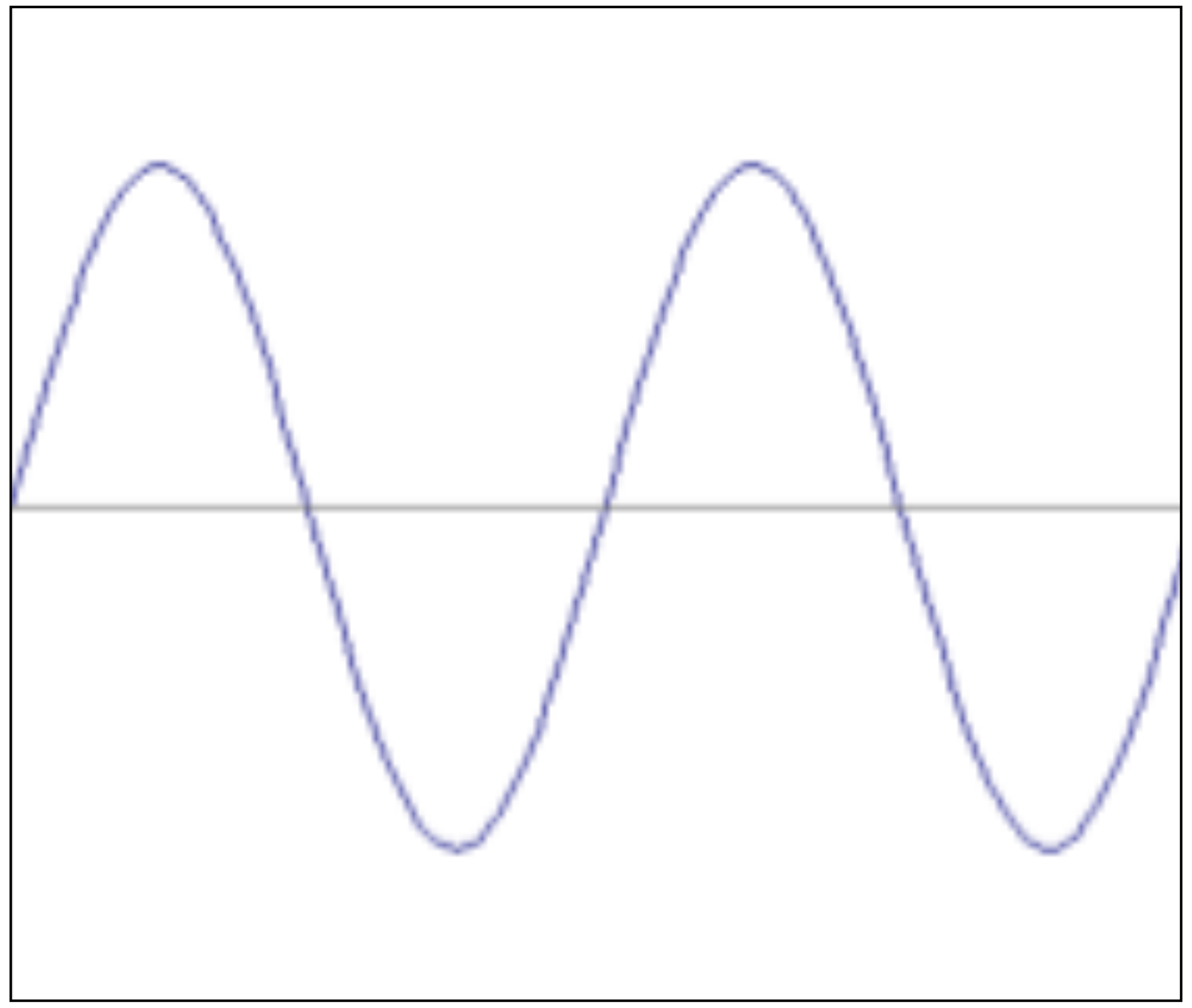
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

How would you generate this function?



=



$\sin(2\pi x)$

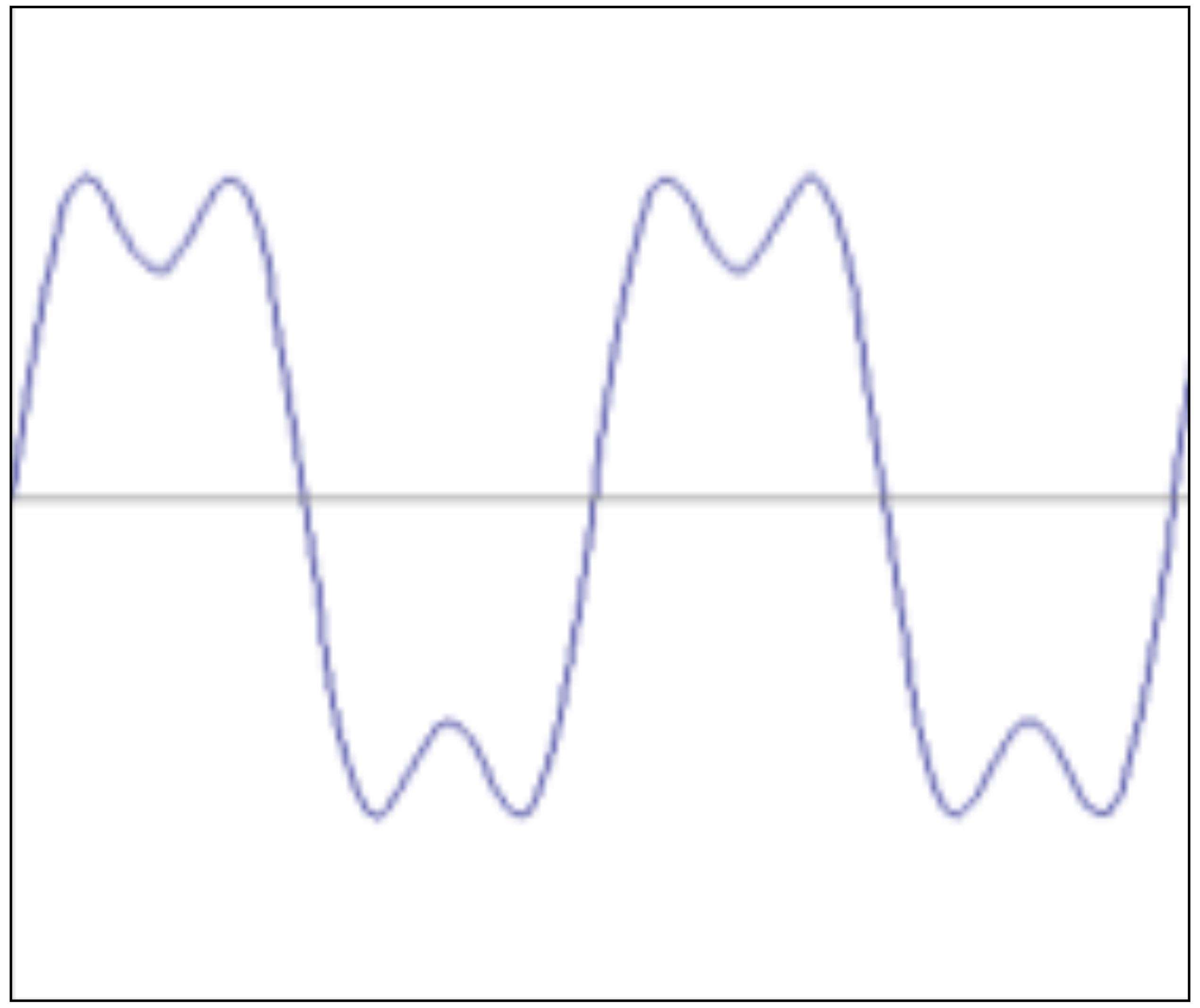
+

?

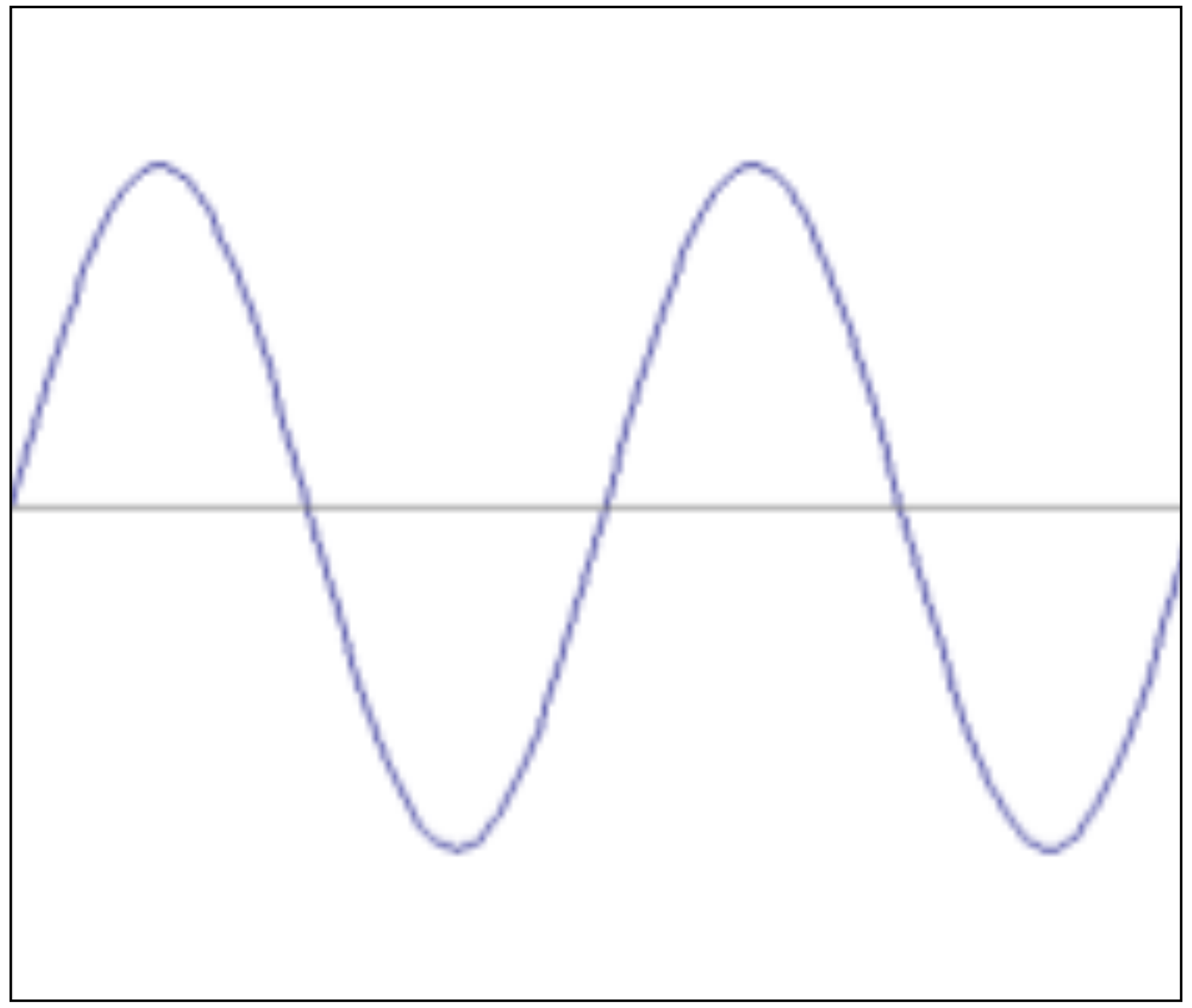
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

How would you generate this function?

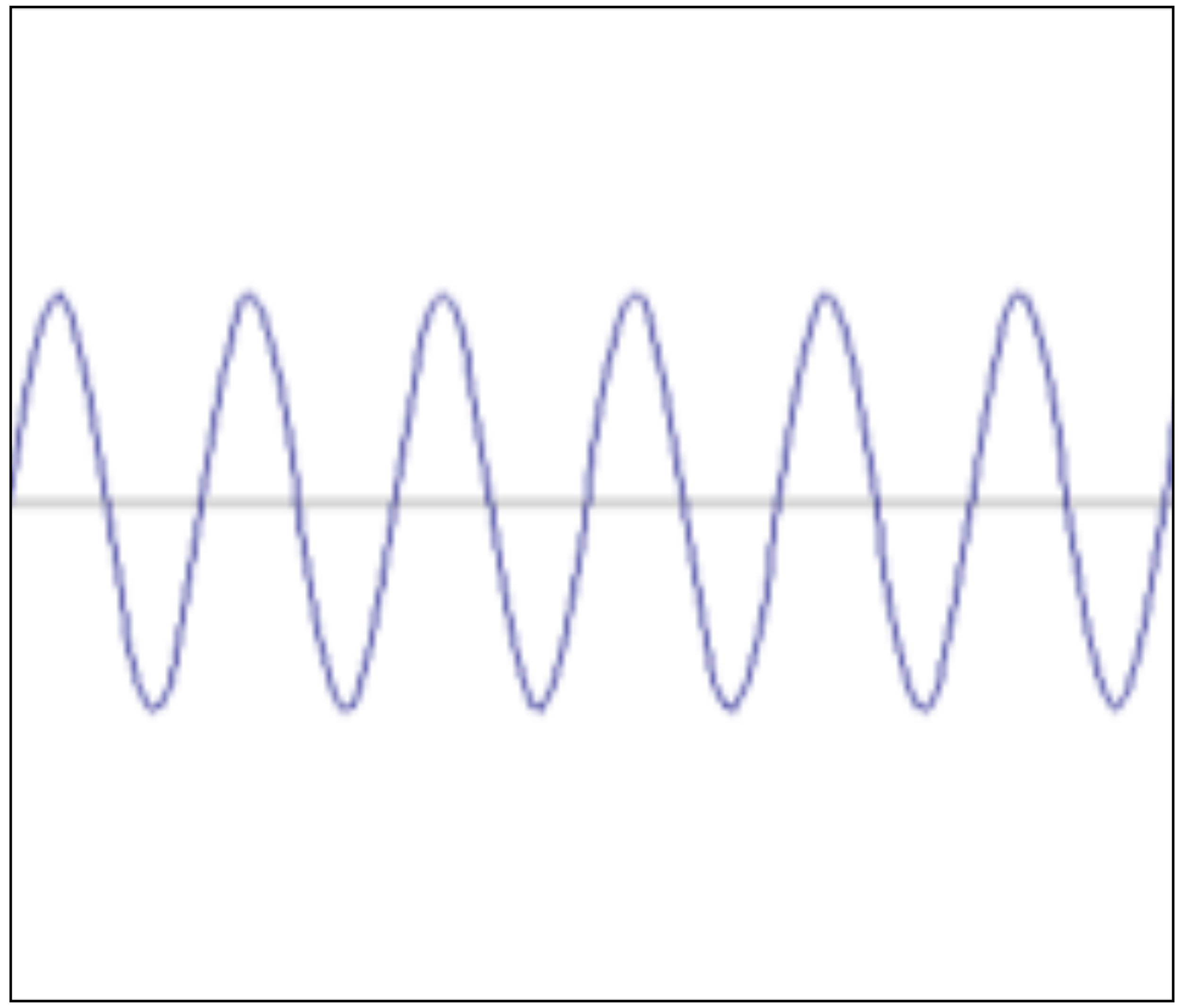


=



$\sin(2\pi x)$

+

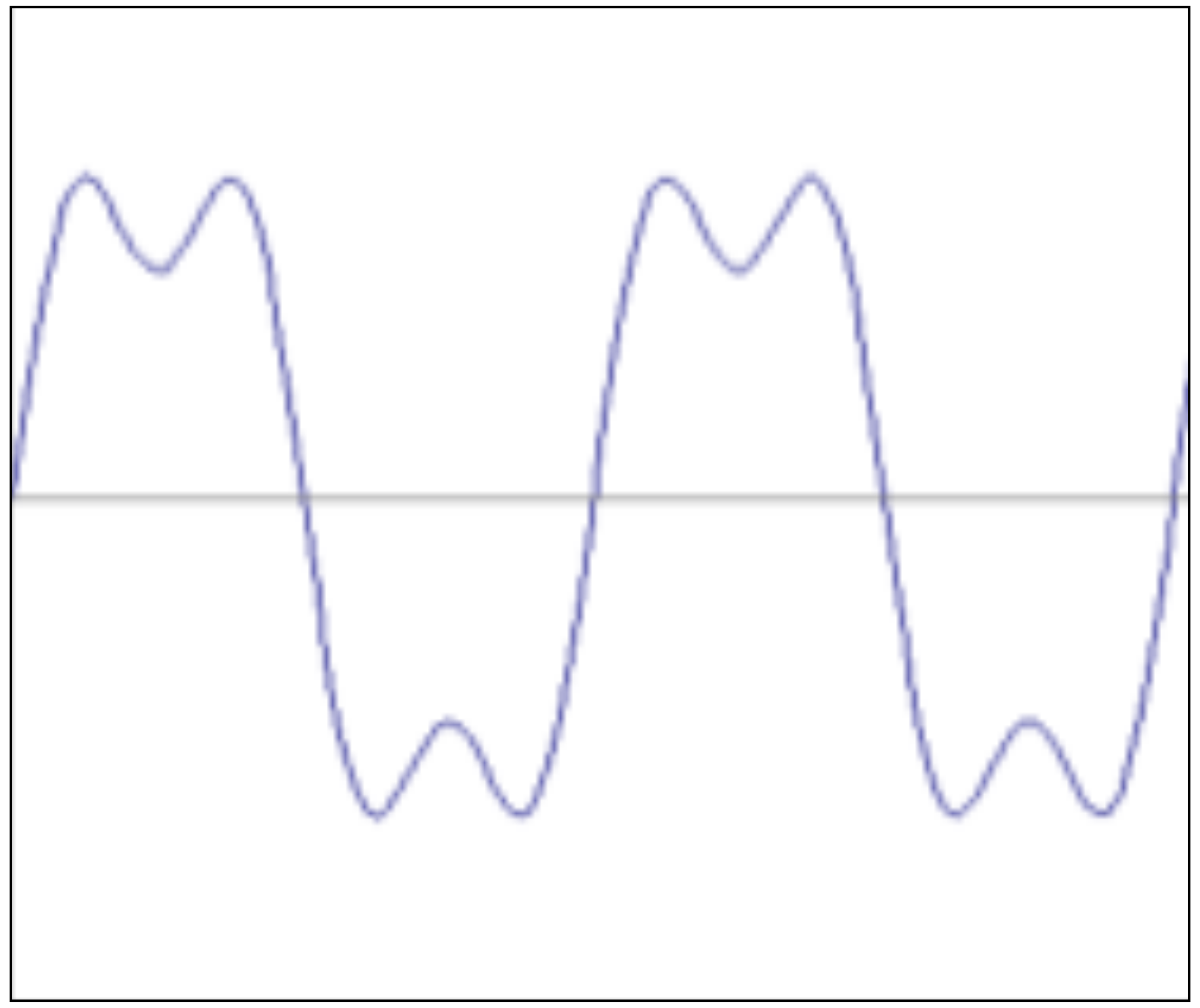


$\frac{1}{3}\sin(2\pi 3x)$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

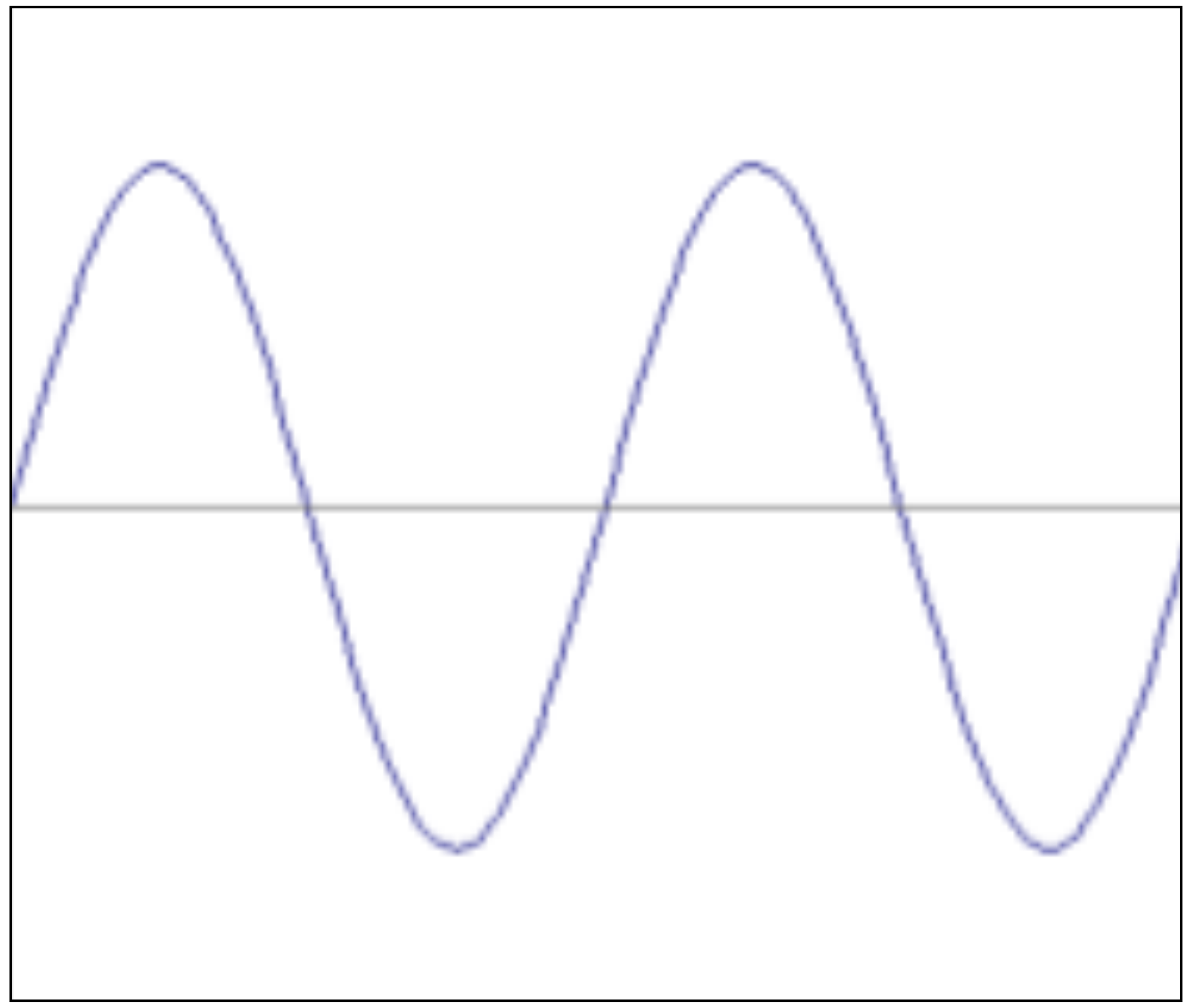
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?



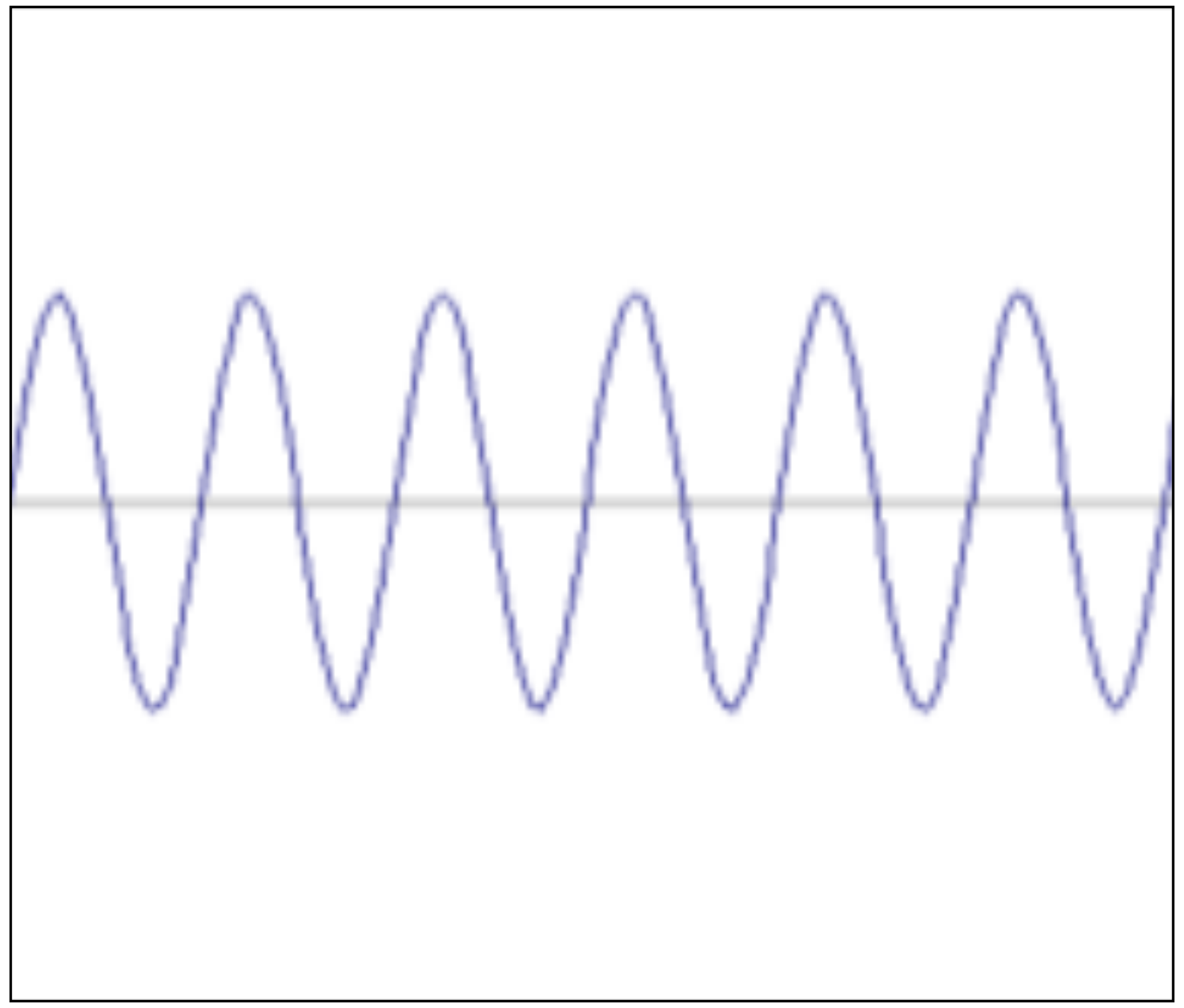
$$f(x) = \sin(2\pi x) + \frac{1}{3}\sin(2\pi 3x)$$

=



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

+

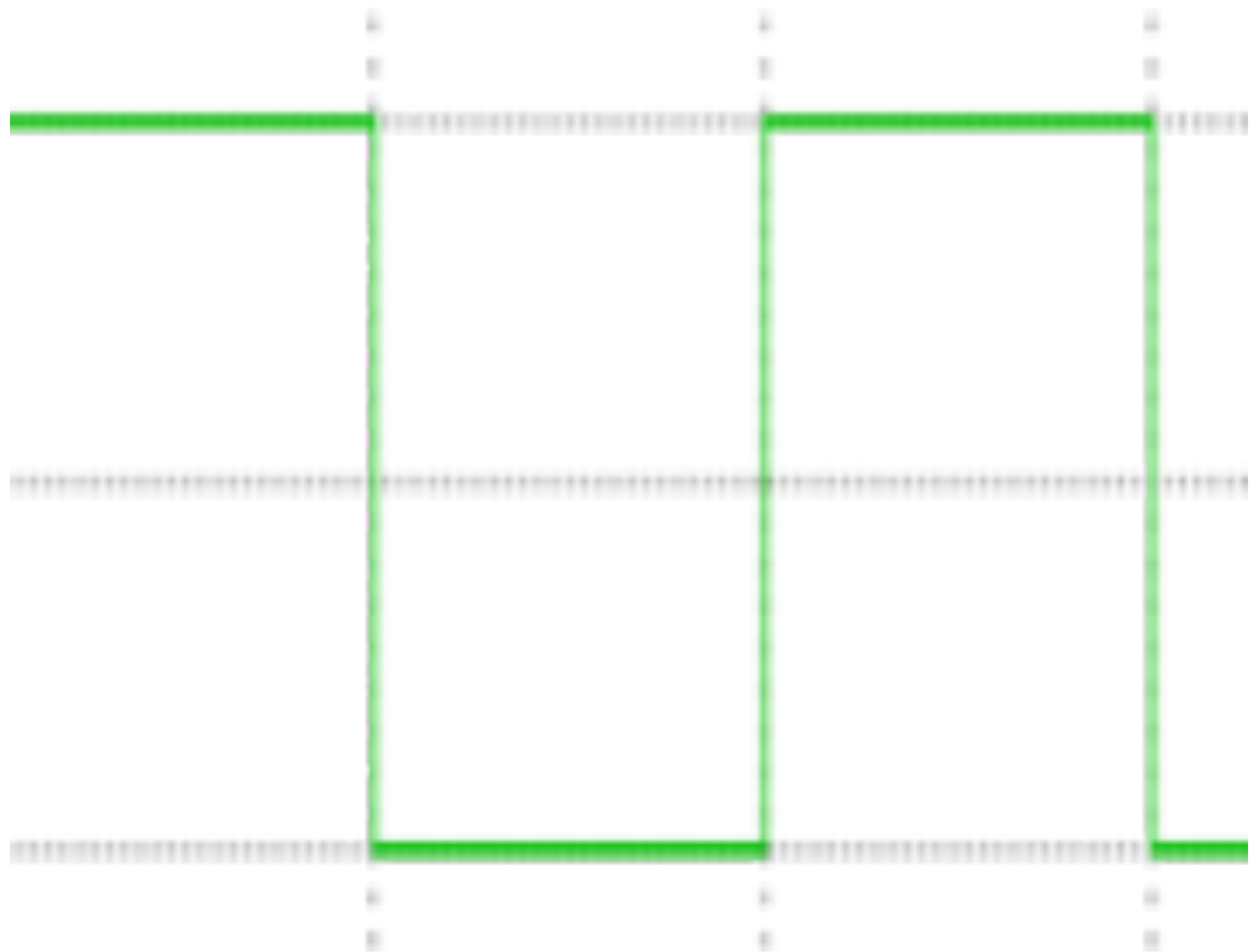


$$\frac{1}{3}\sin(2\pi 3x)$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

How would you generate this function?



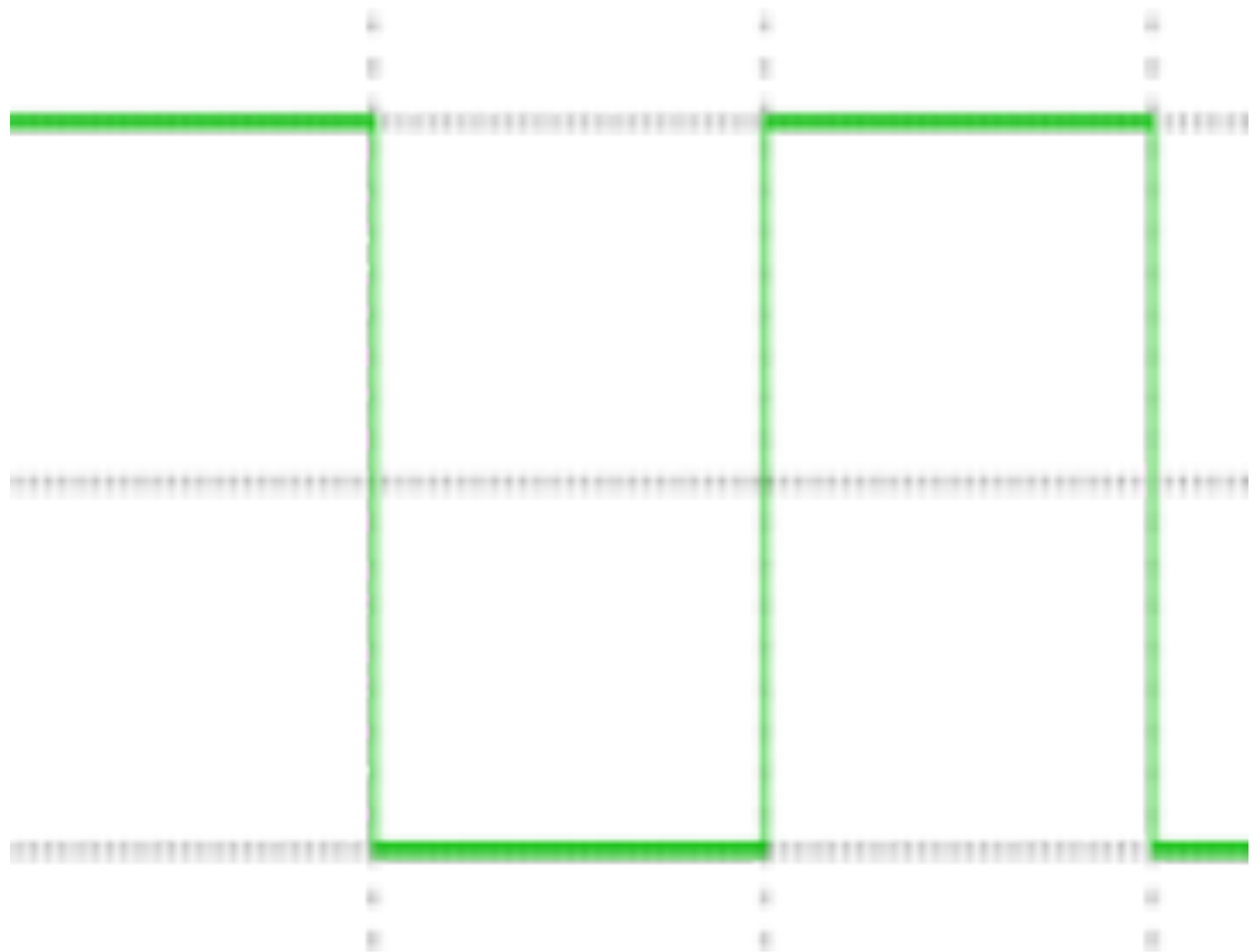
square wave

$\approx$? + ?

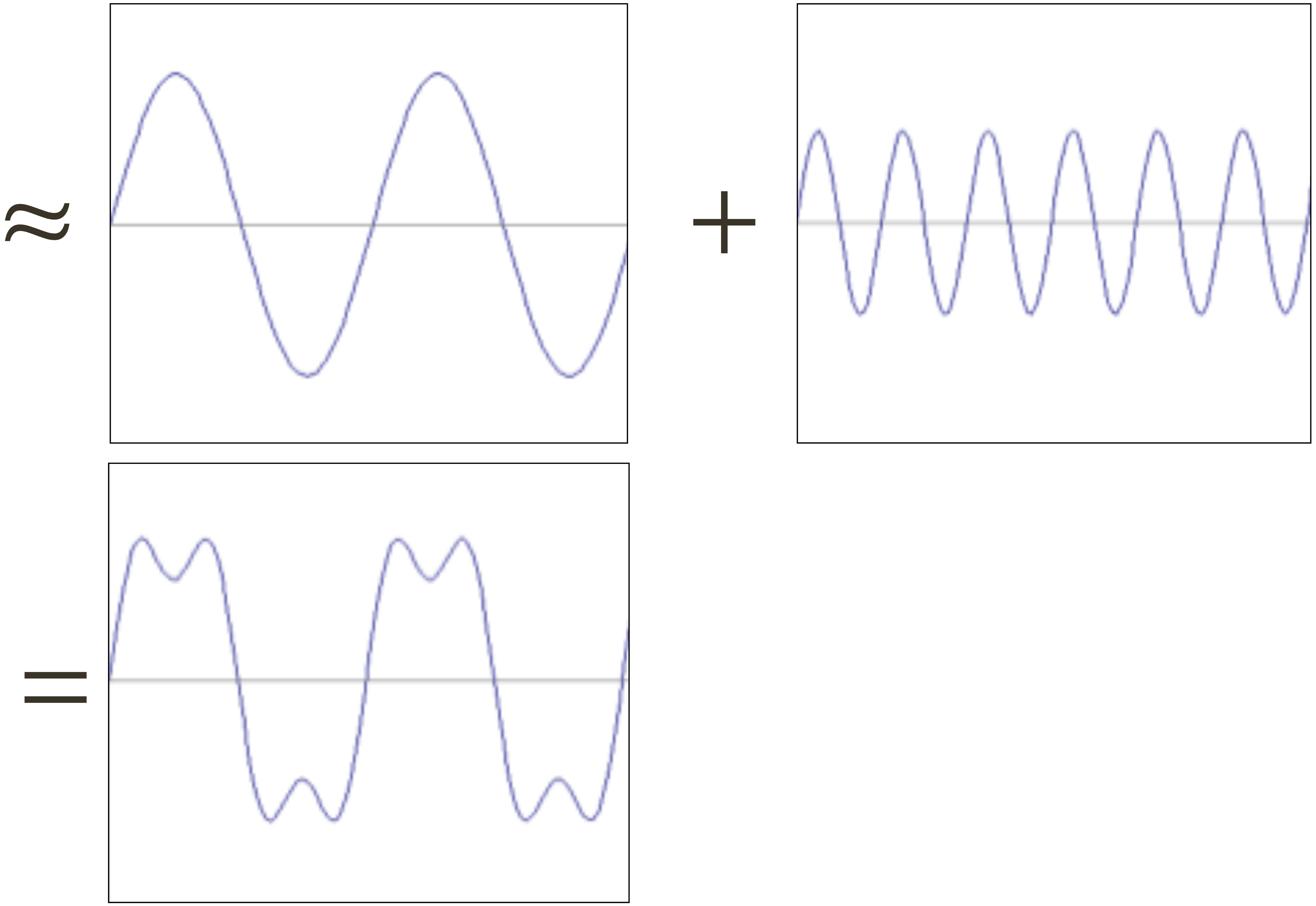
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

How would you generate this function?



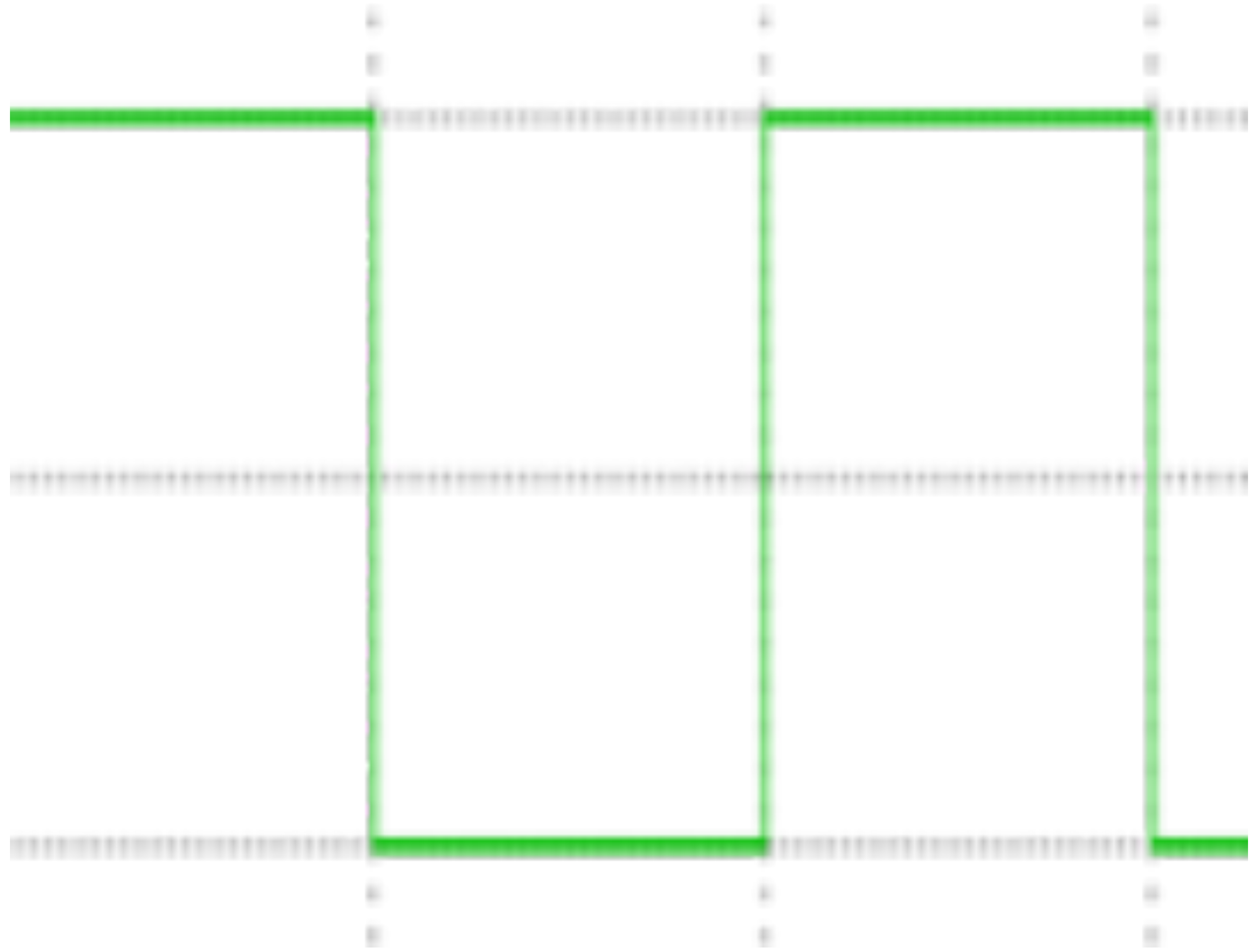
square wave



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

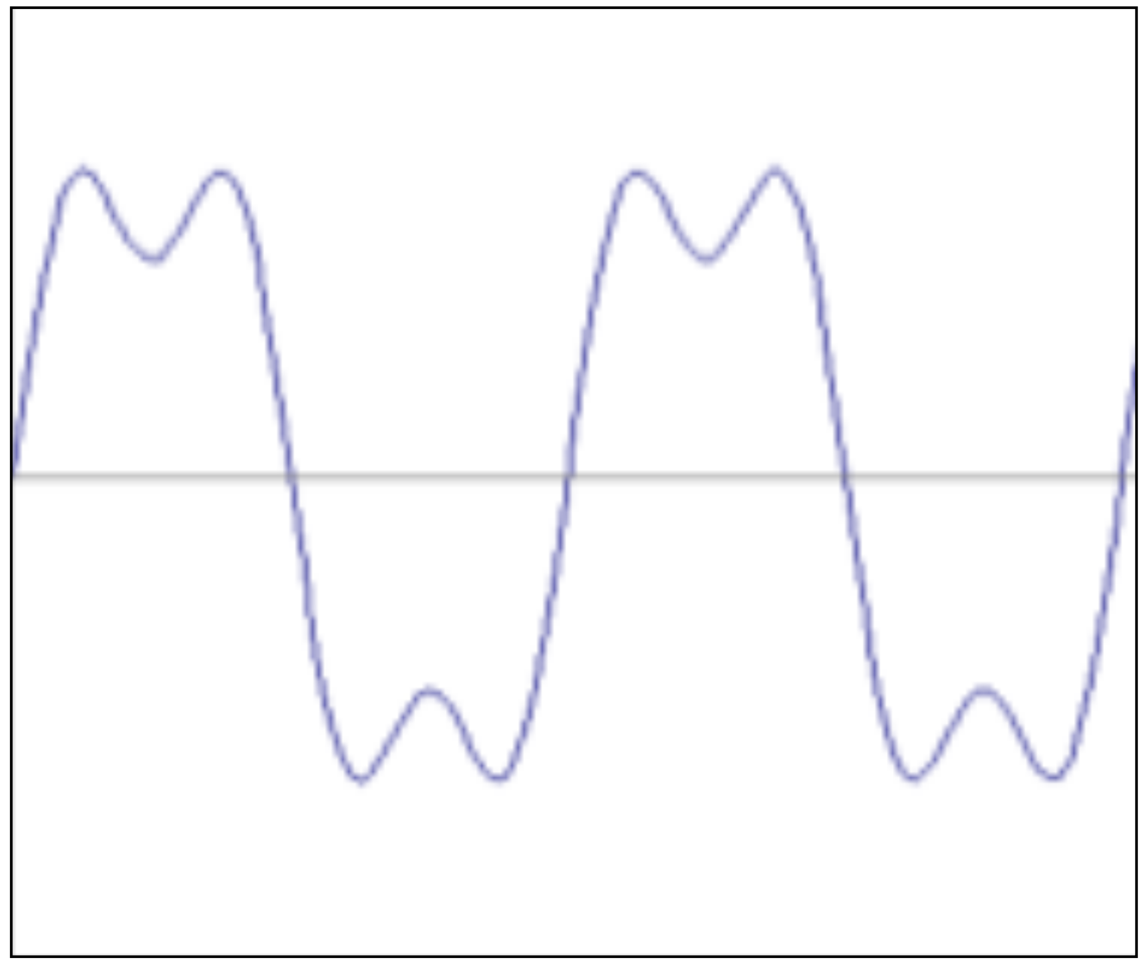
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

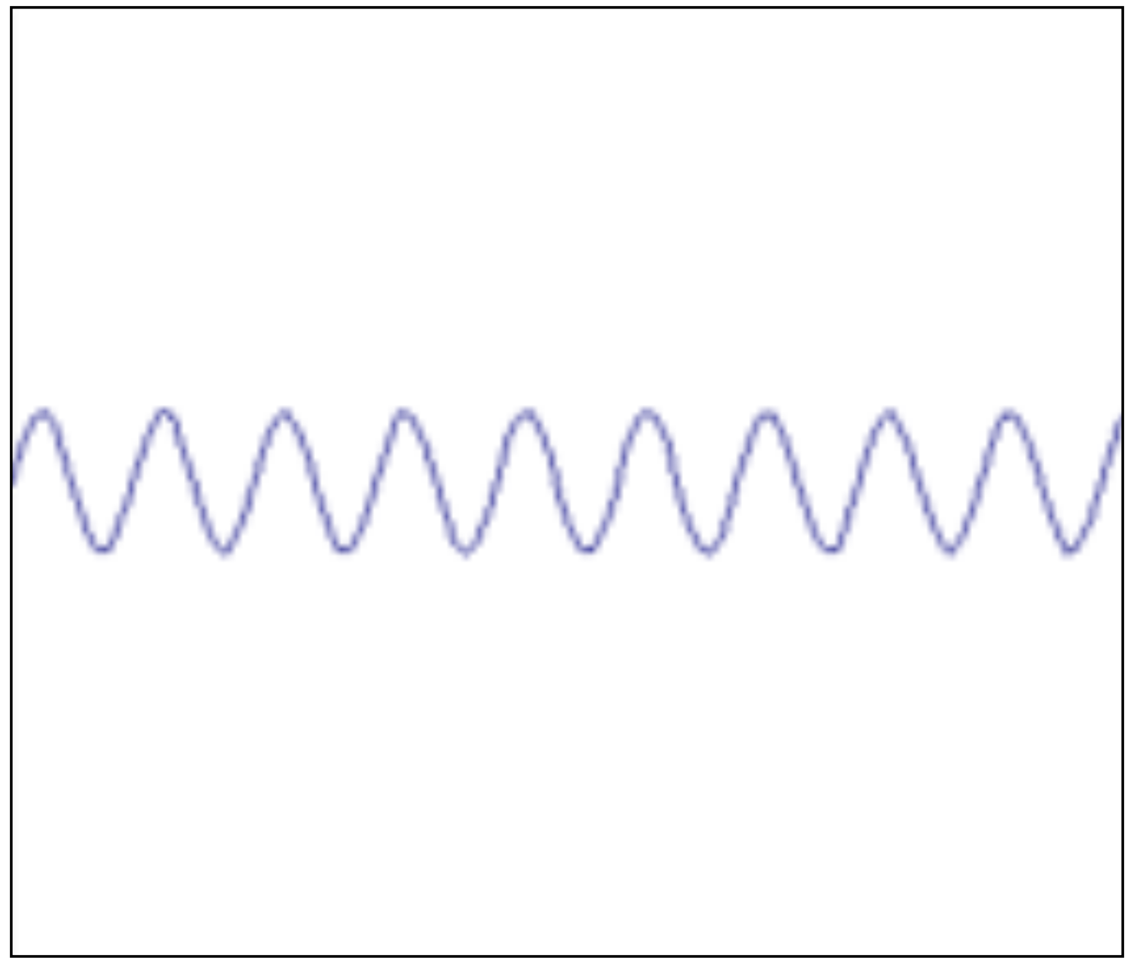


square wave

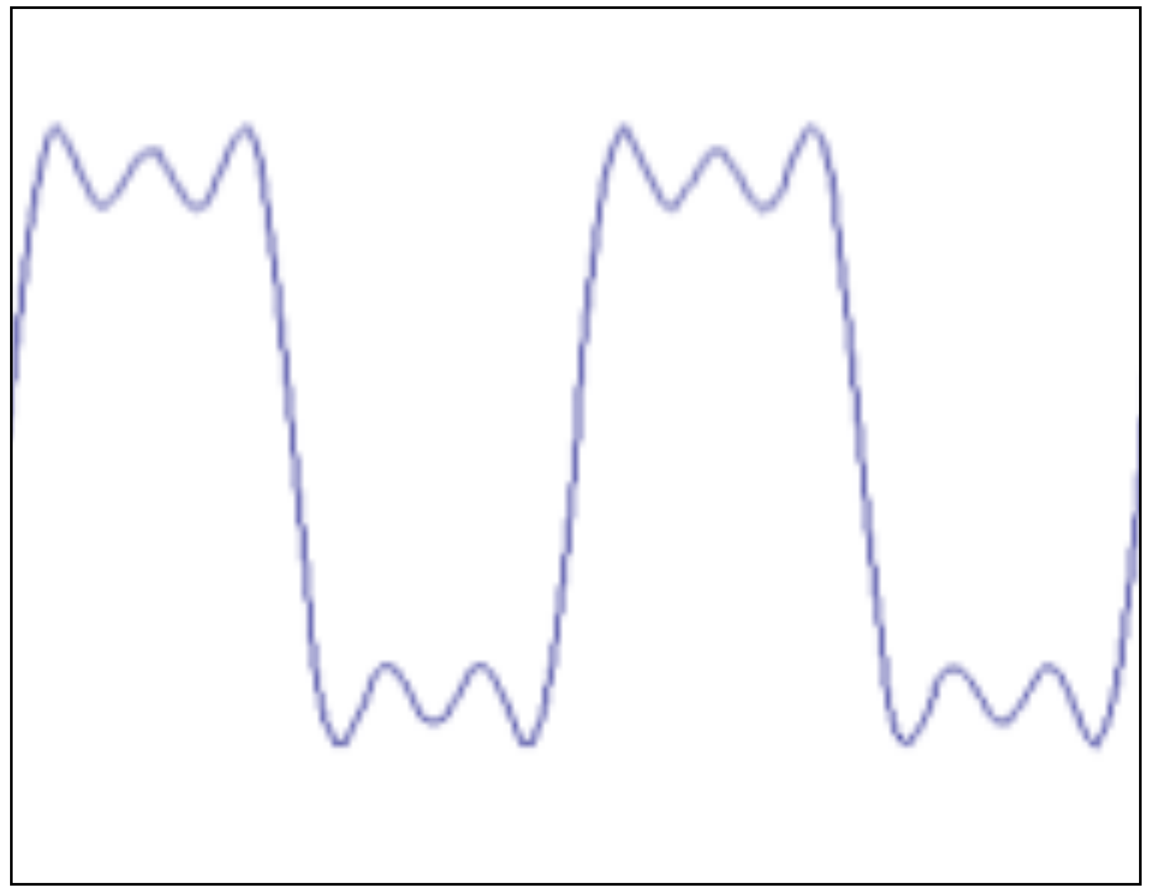
$\approx$



$+$



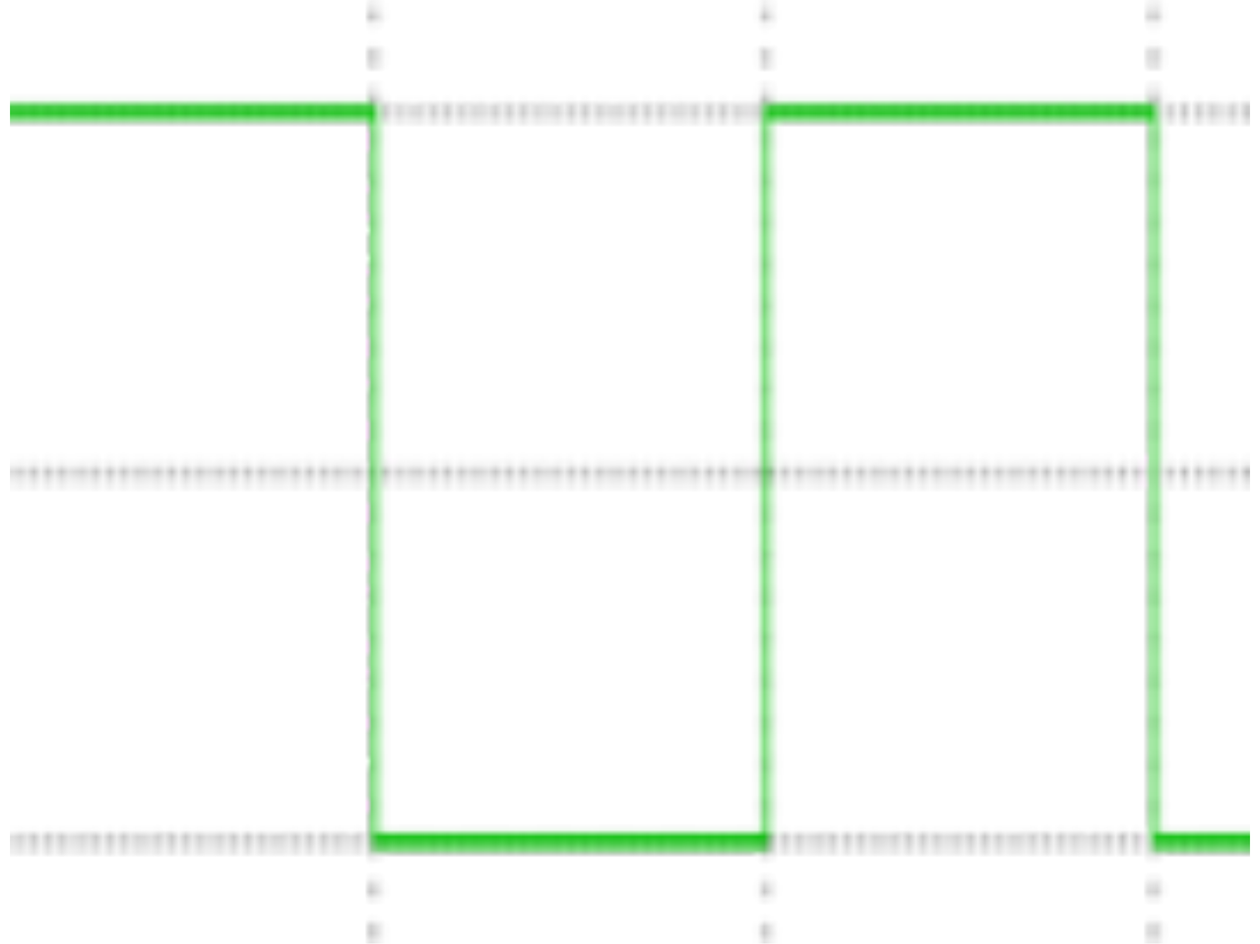
$=$



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

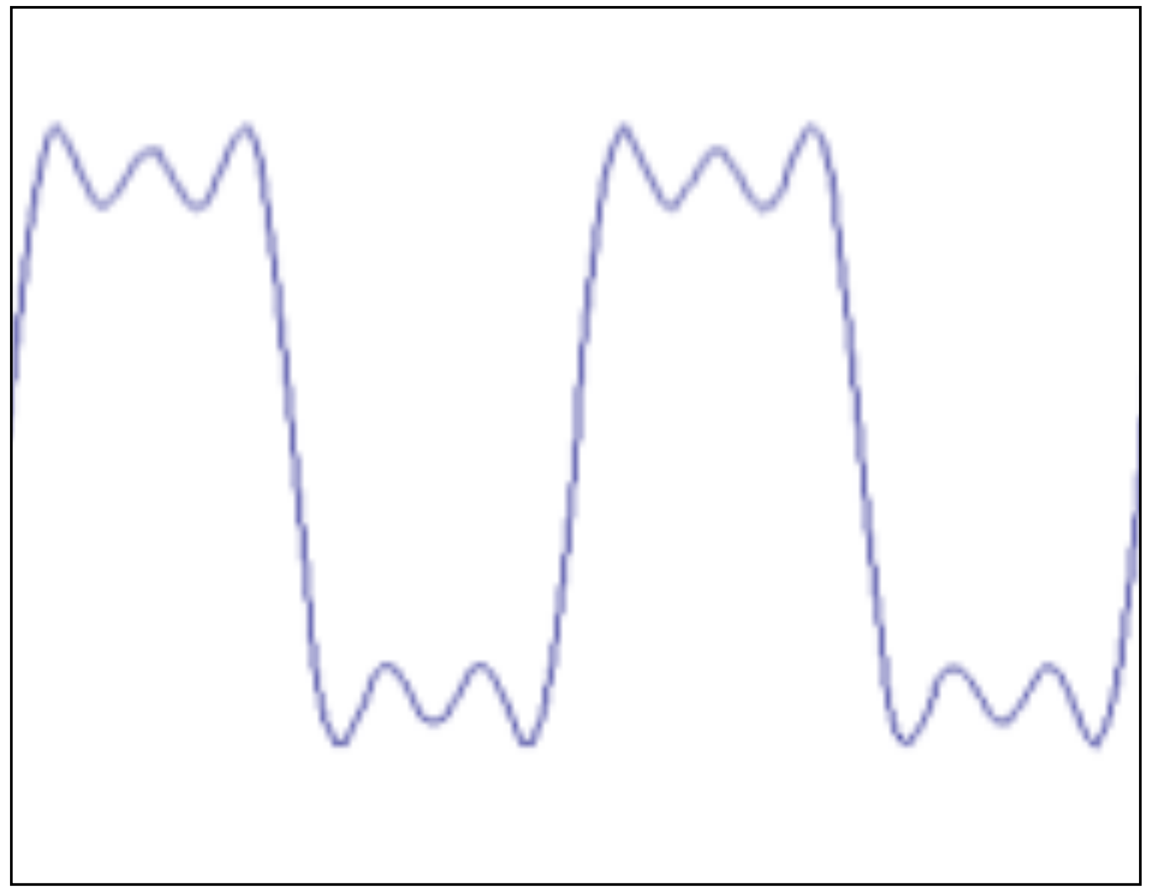
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

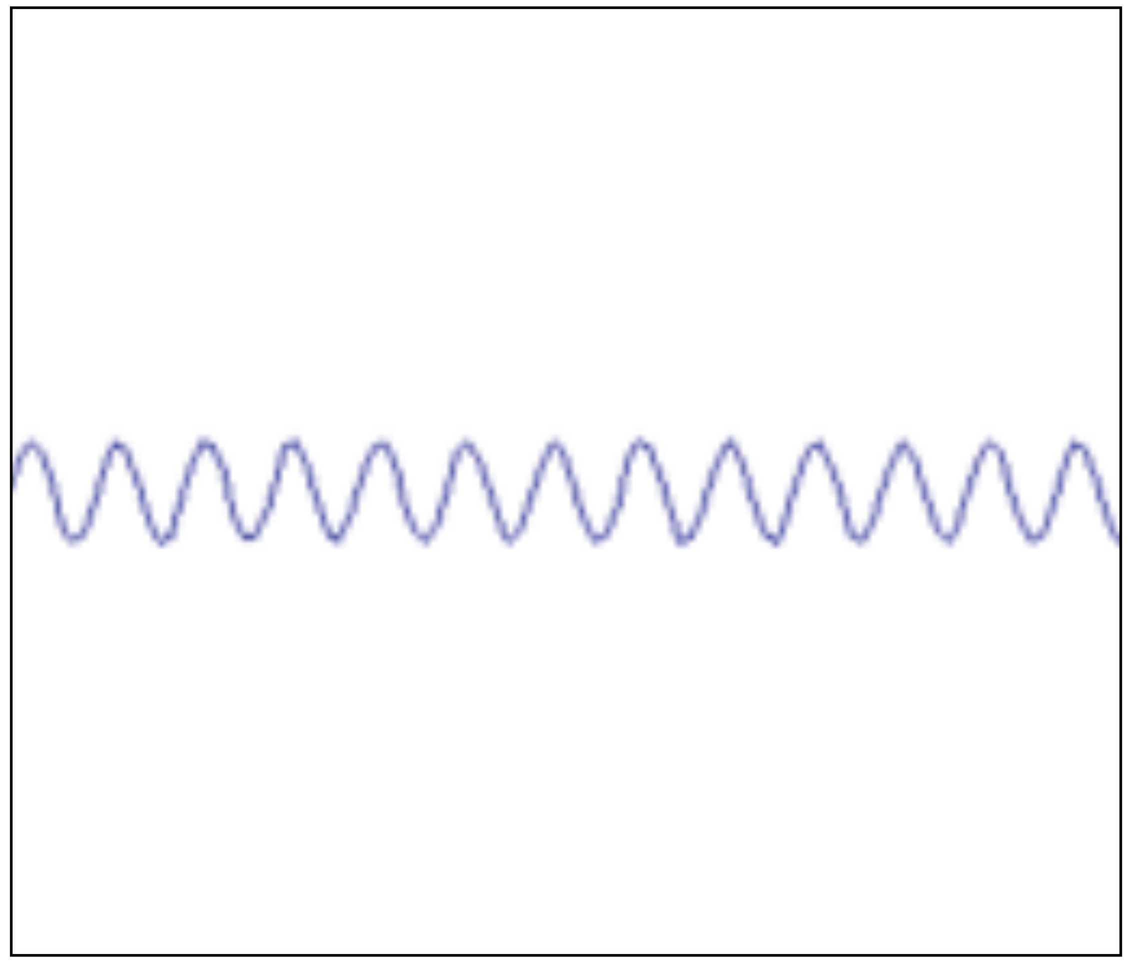


square wave

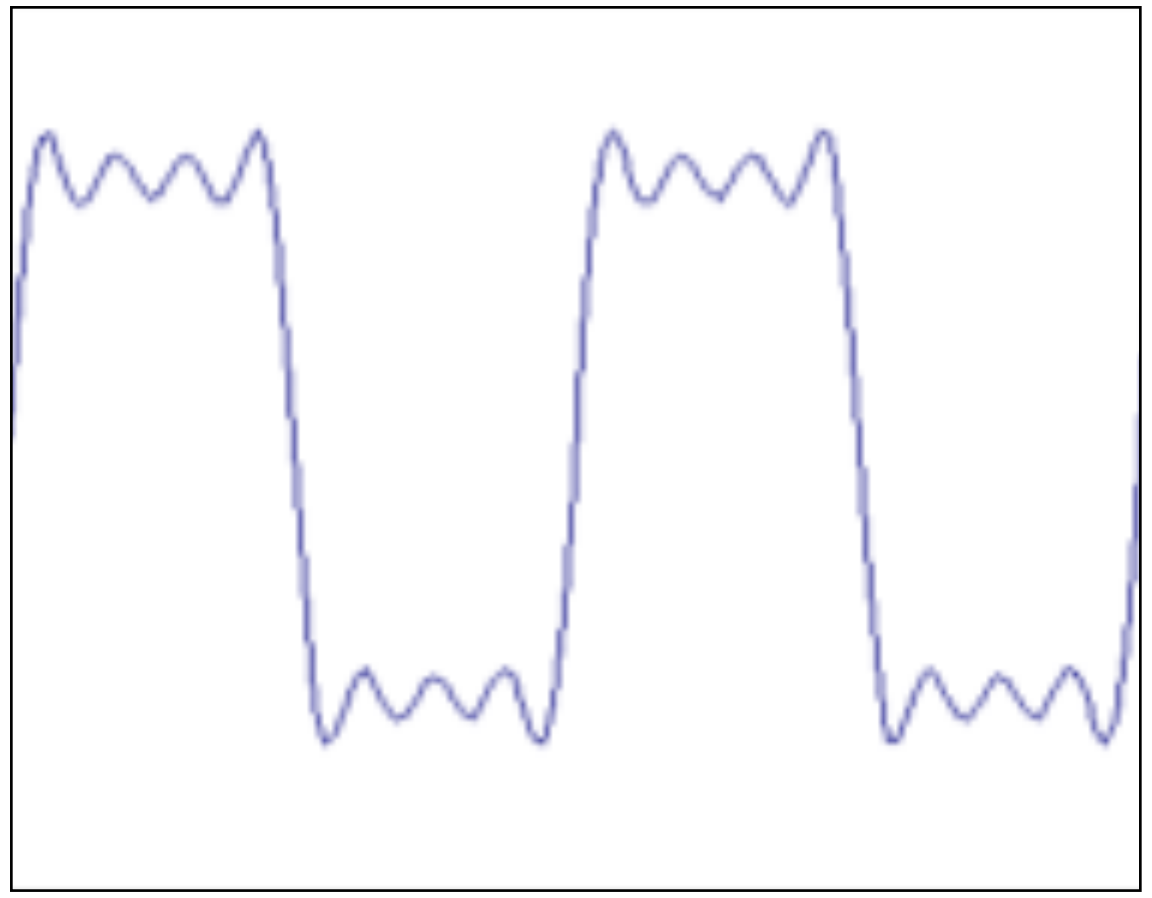
$\approx$



$+$



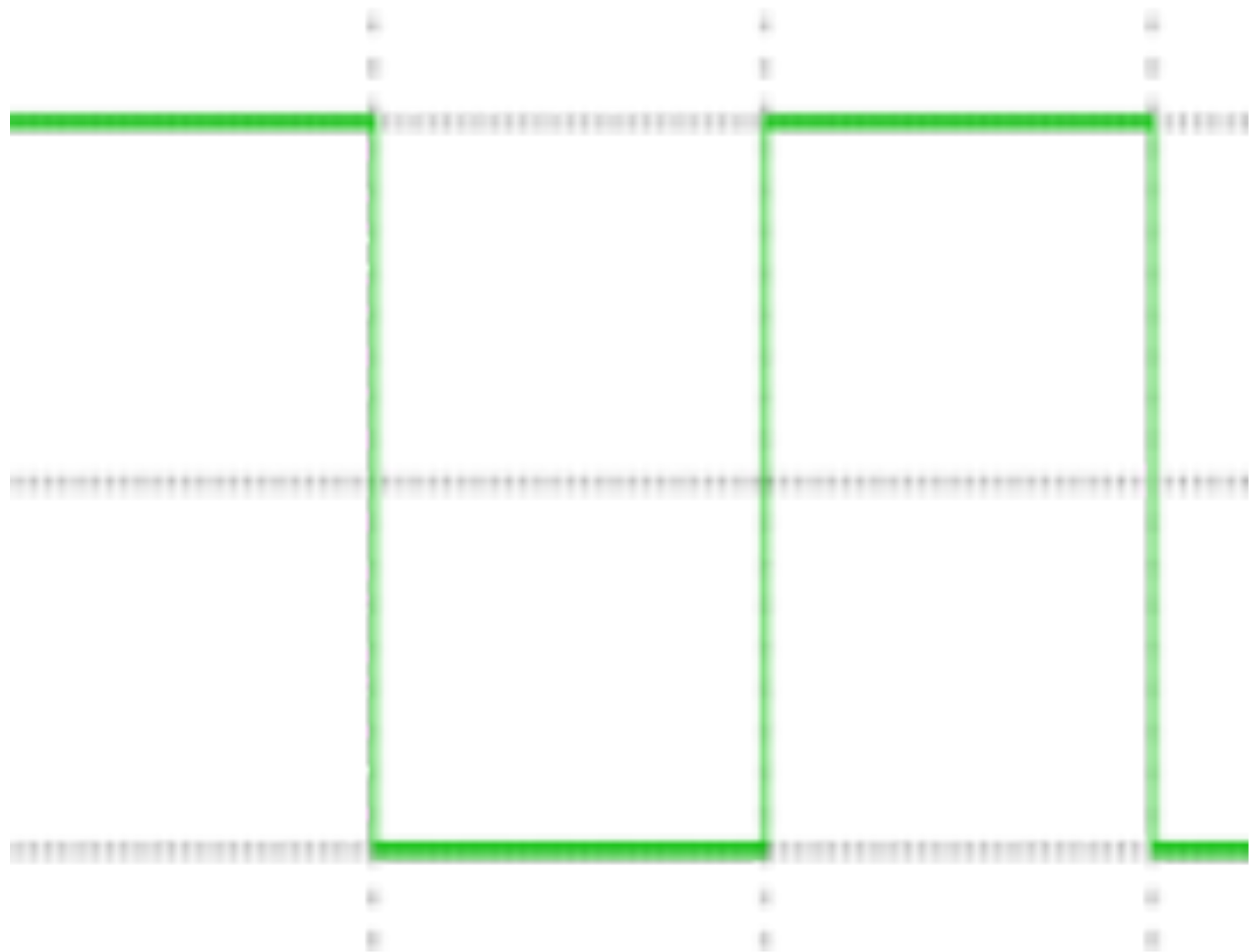
$=$



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

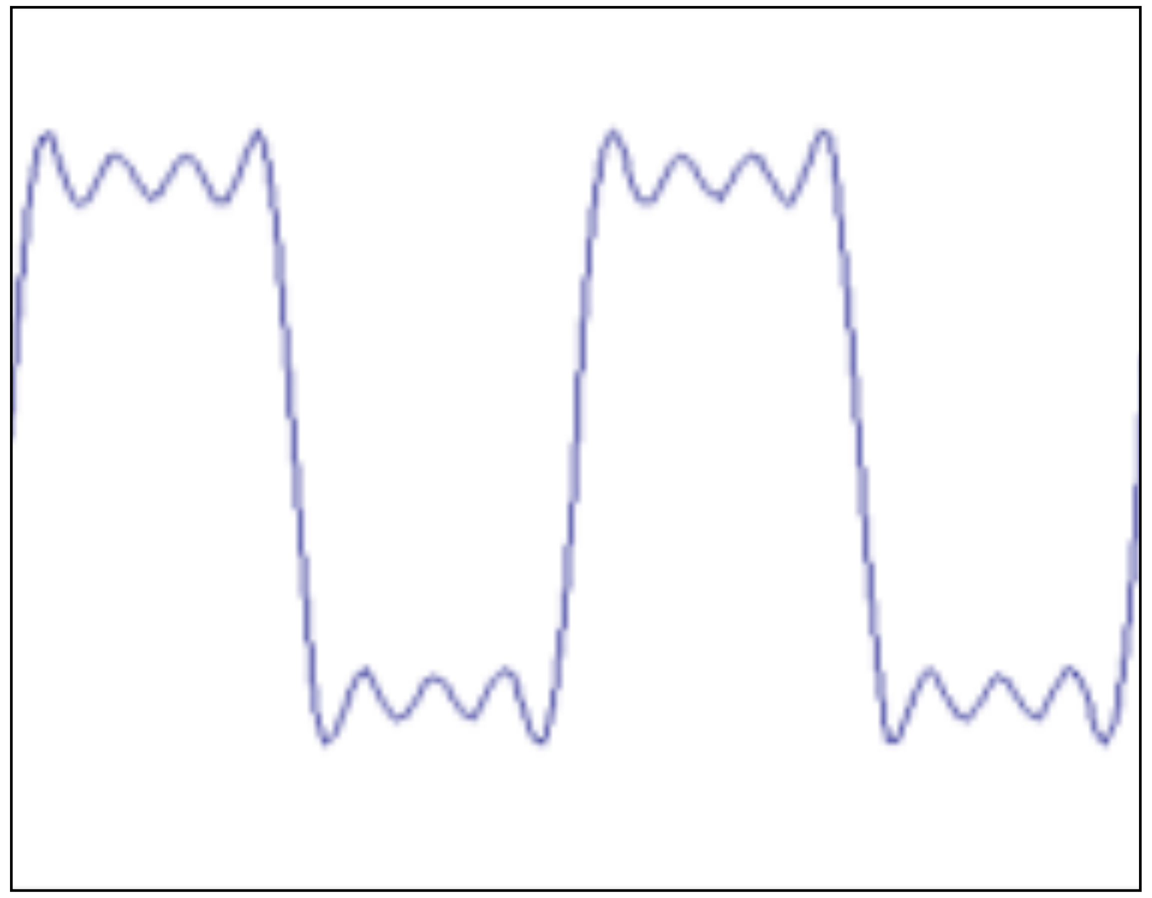
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

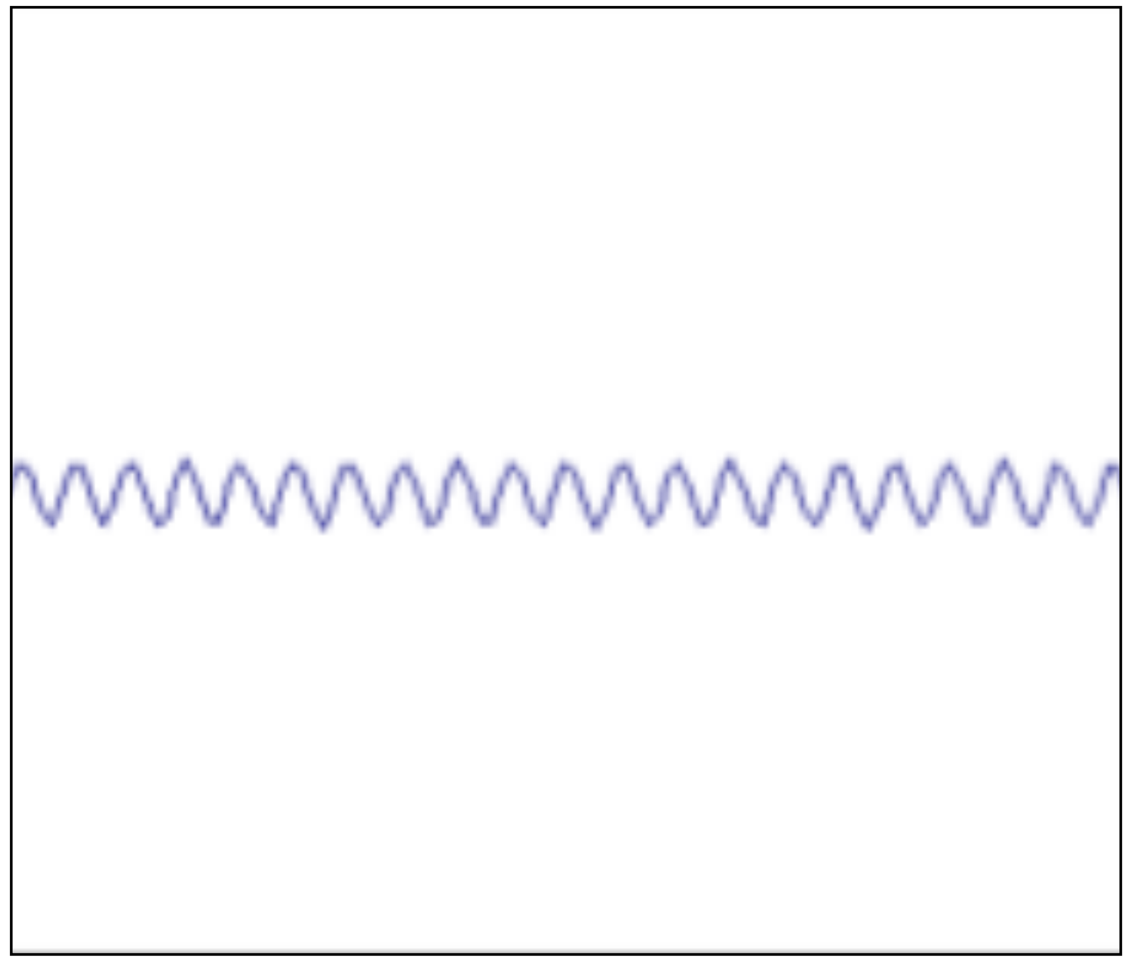


square wave

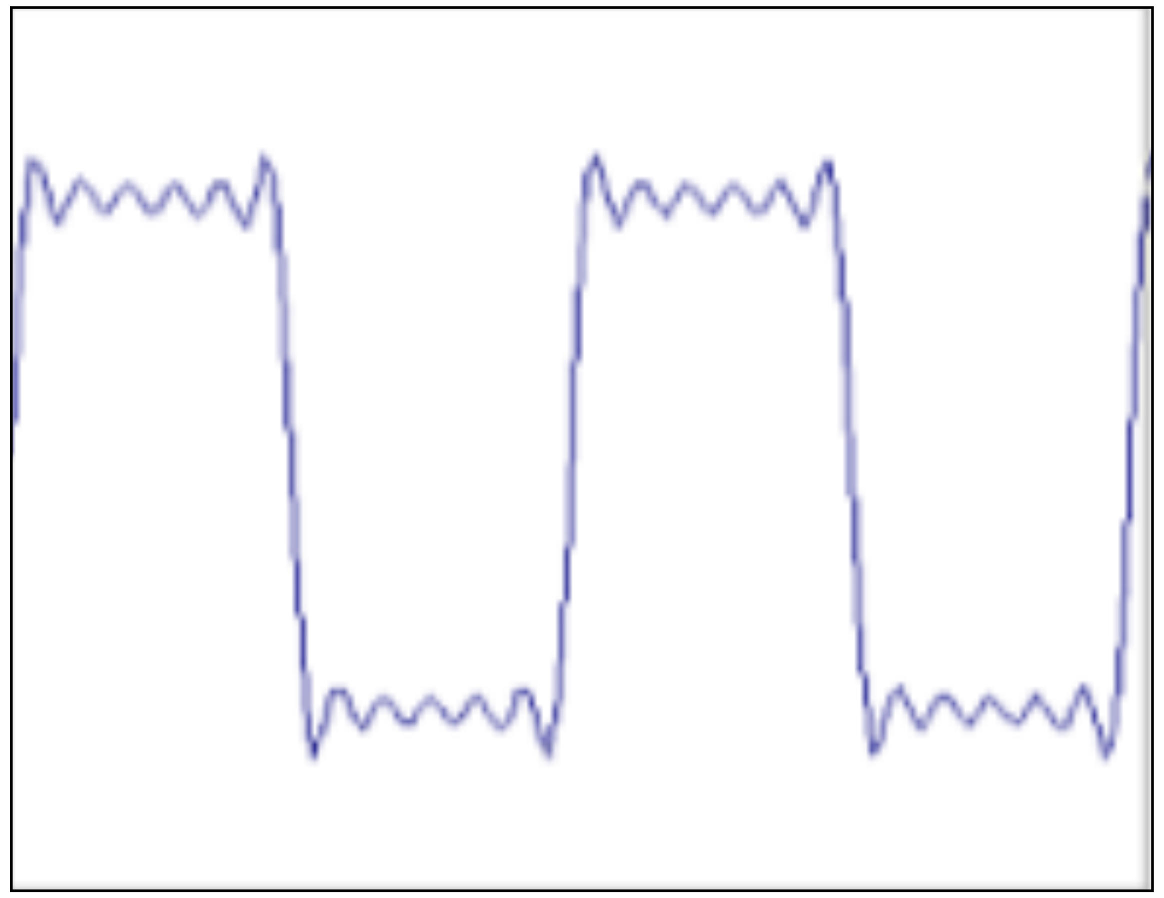
$\approx$



$+$



$=$

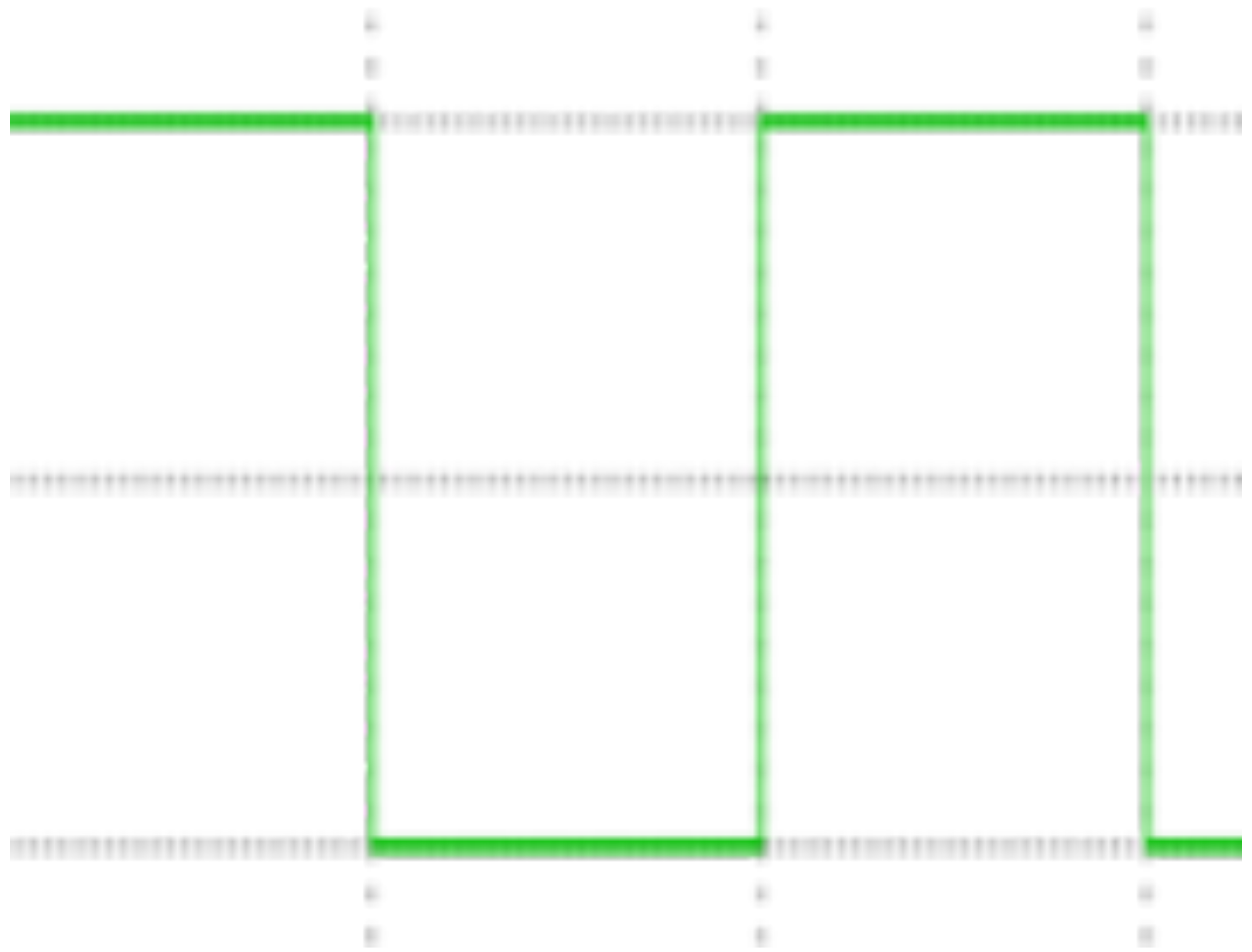


How would you express this mathematically?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

How would you generate this function?



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

square wave

infinite sum of sine waves

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

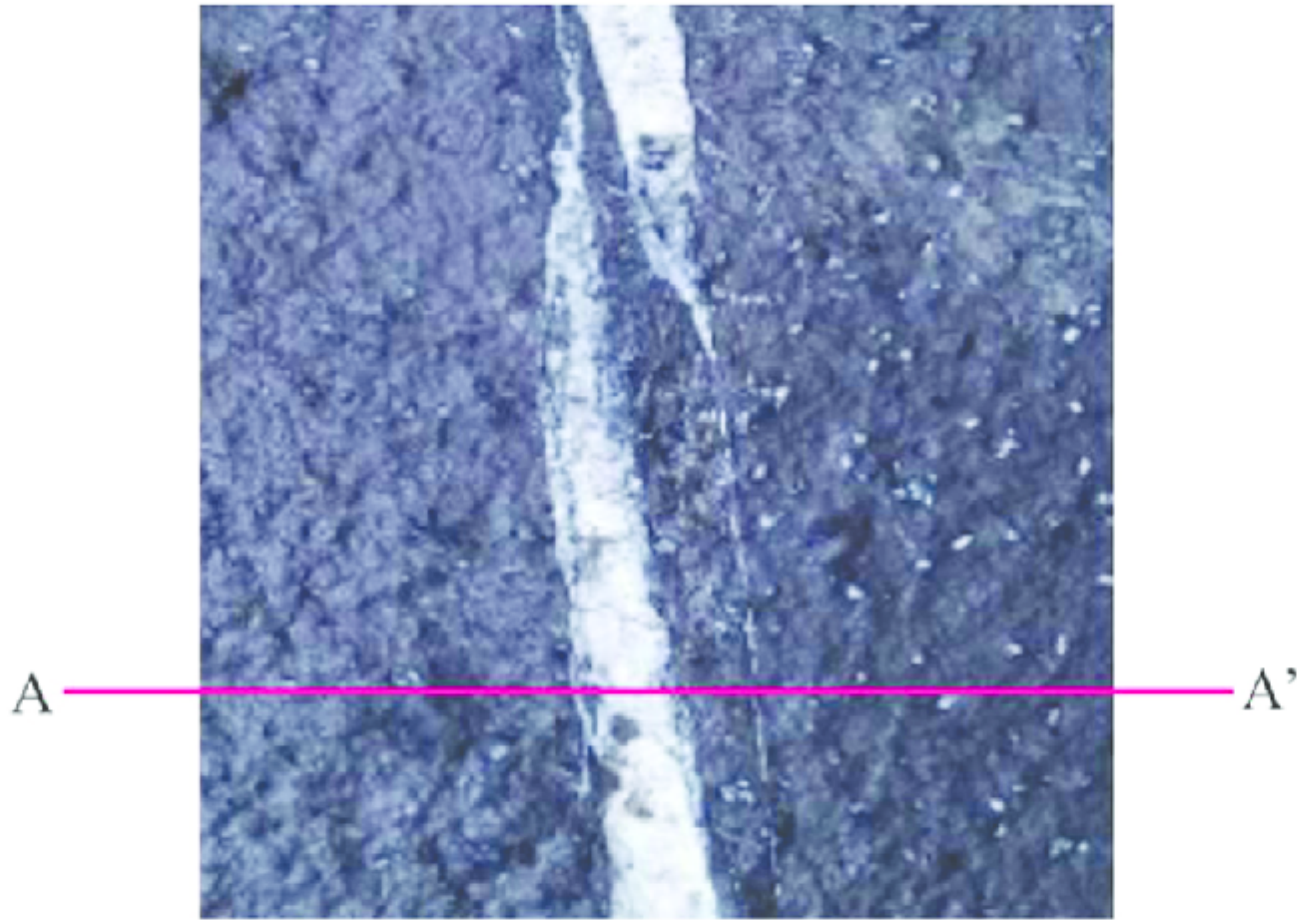
Basic building block:

$$A \sin(\omega x + \phi)$$

Fourier's claim: Add enough of these to get any periodic signal you want!

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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



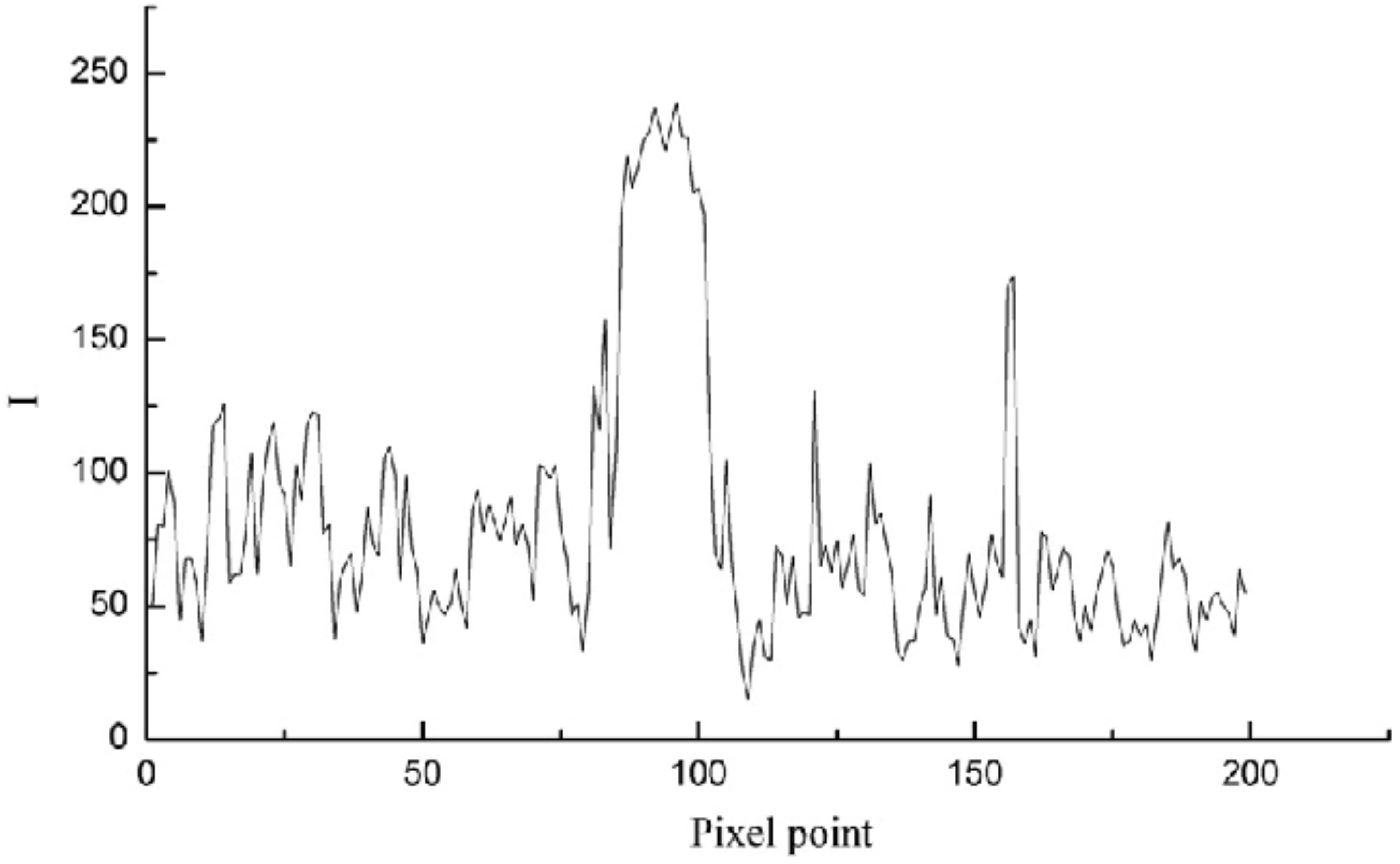


Image from: Numerical Simulation and Fractal Analysis of Mesoscopic Scale Failure in Shale Using Digital Images

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

What are “frequencies” in an image?

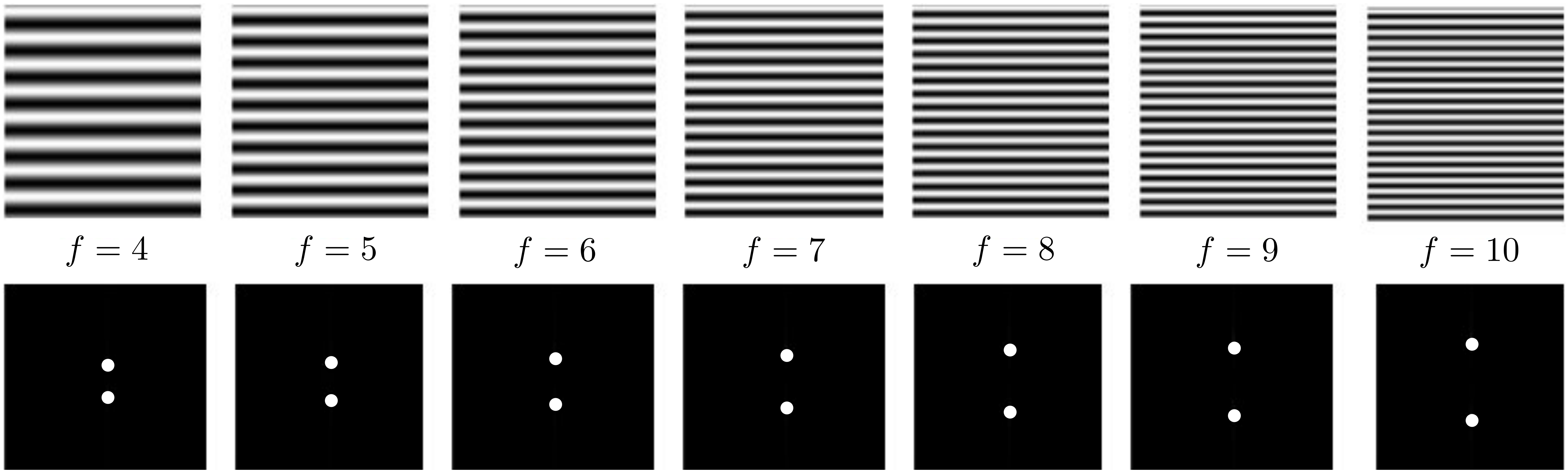
Spatial frequency

$f = 4$ $f = 5$ $f = 6$ $f = 7$ $f = 8$ $f = 9$ $f = 10$

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

What are “frequencies” in an image?

Spatial frequency

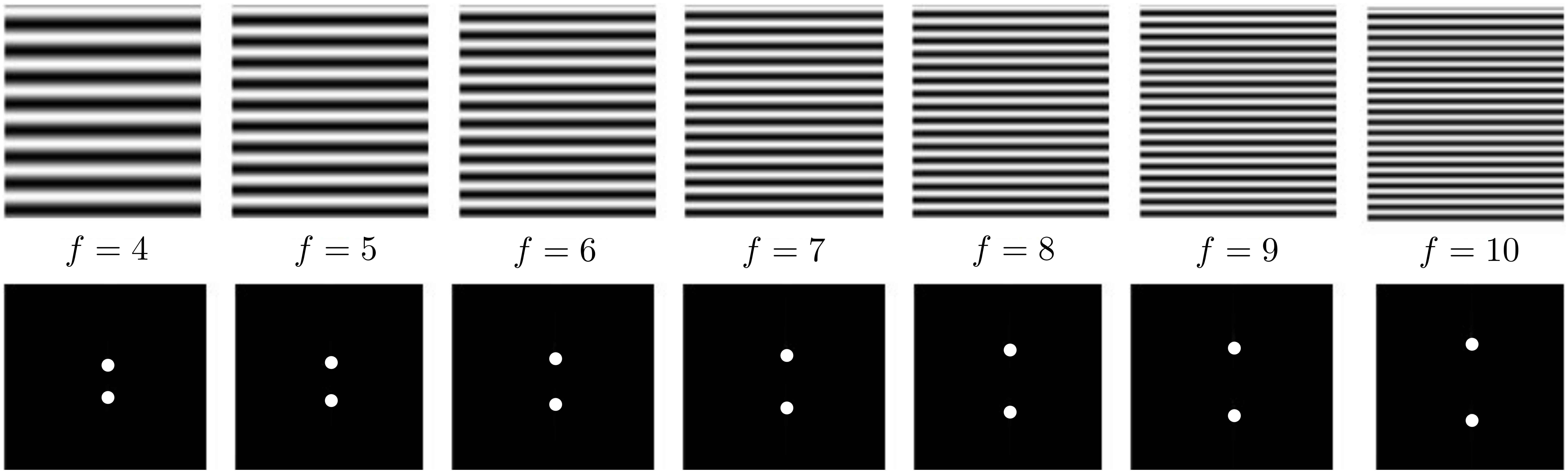


Amplitude (magnitude) of Fourier transform (phase does not show desirable correlations with image structure)

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

What are "frequencies" in an image?

Spatial frequency



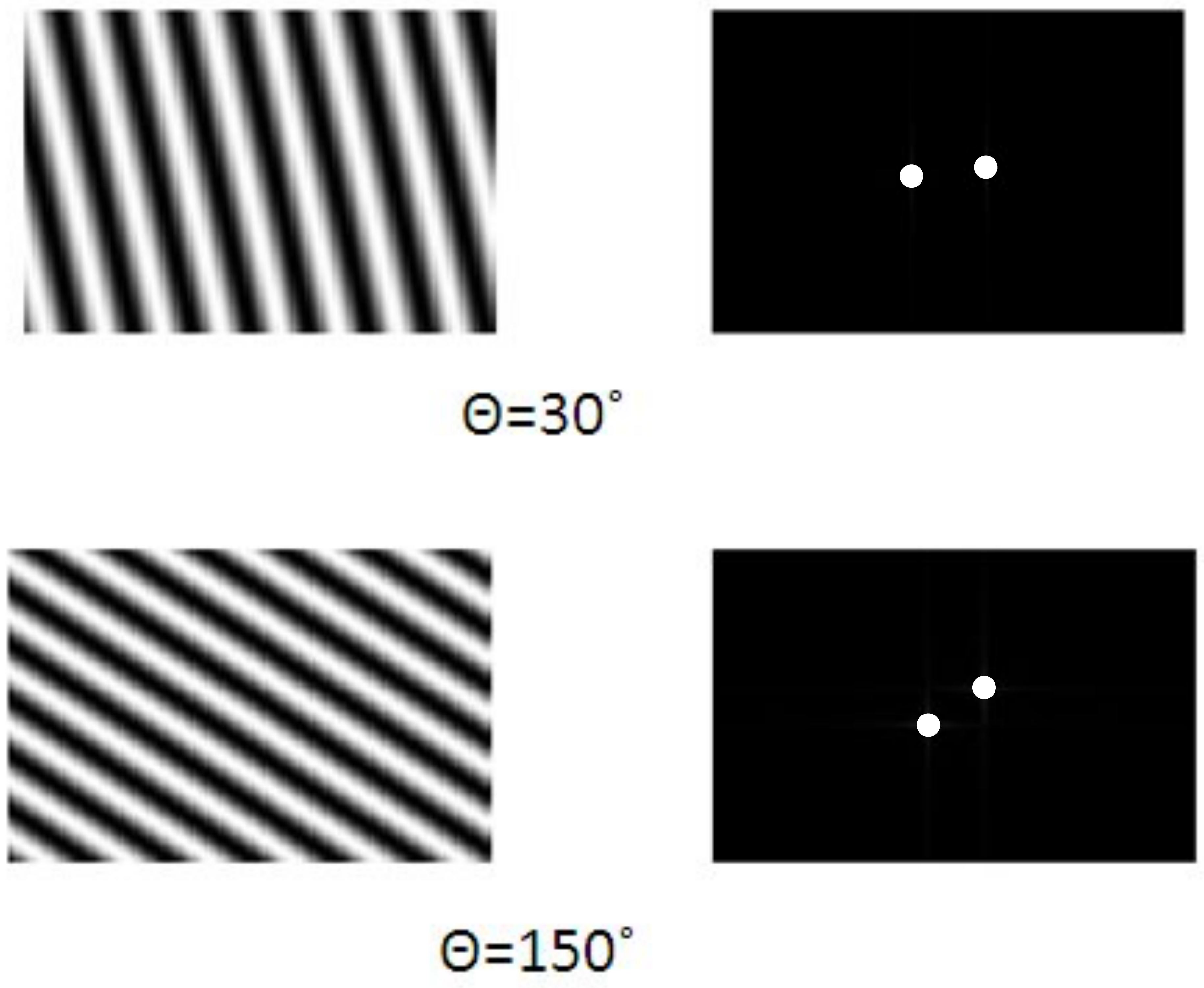
Amplitude (magnitude) of Fourier transform (phase does not show desirable correlations with image structure)

Observation: low frequencies close to the center

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

What are “frequencies” in an image?

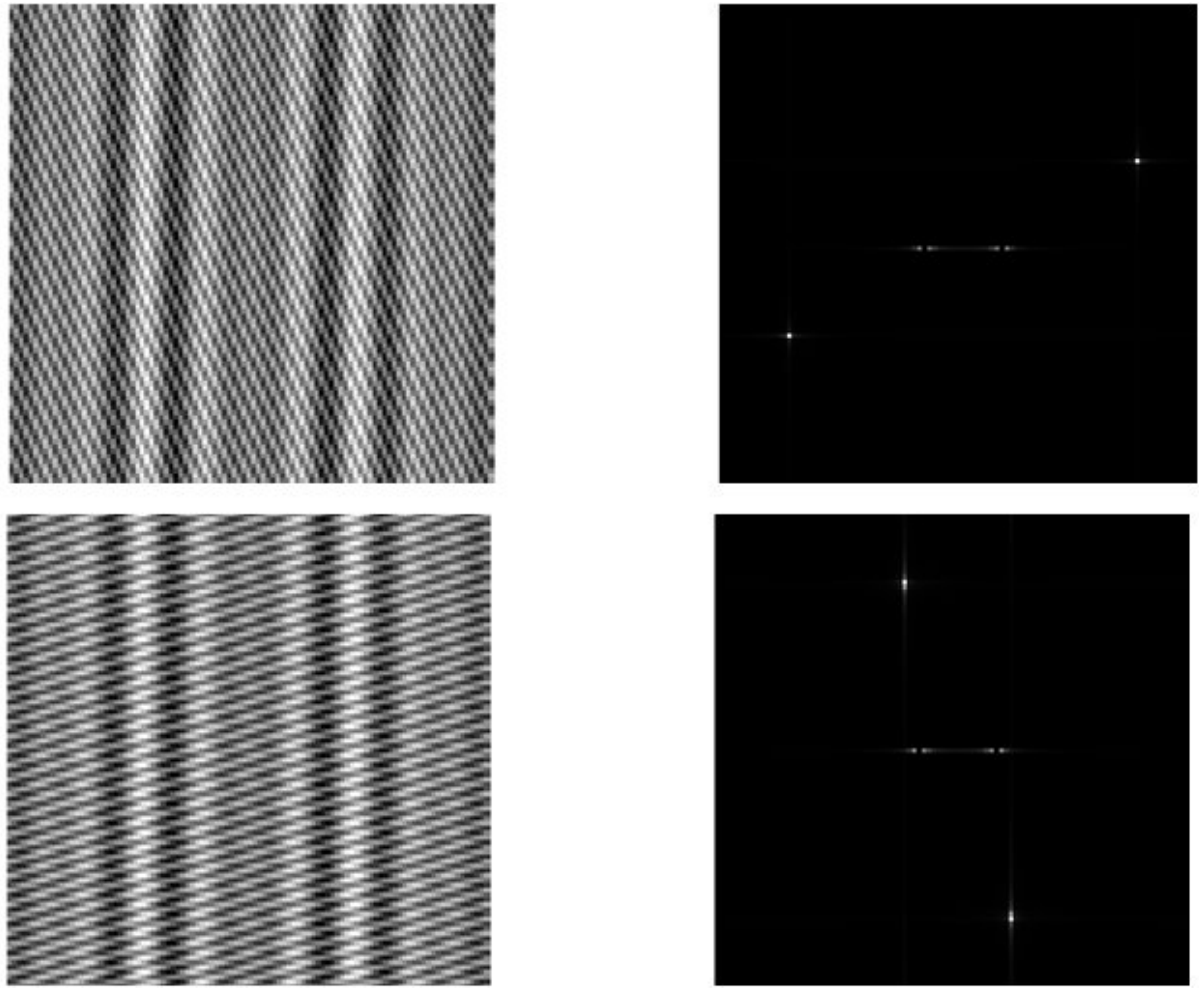
Spatial frequency



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

What are “frequencies” in an image?

Spatial frequency



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

Image

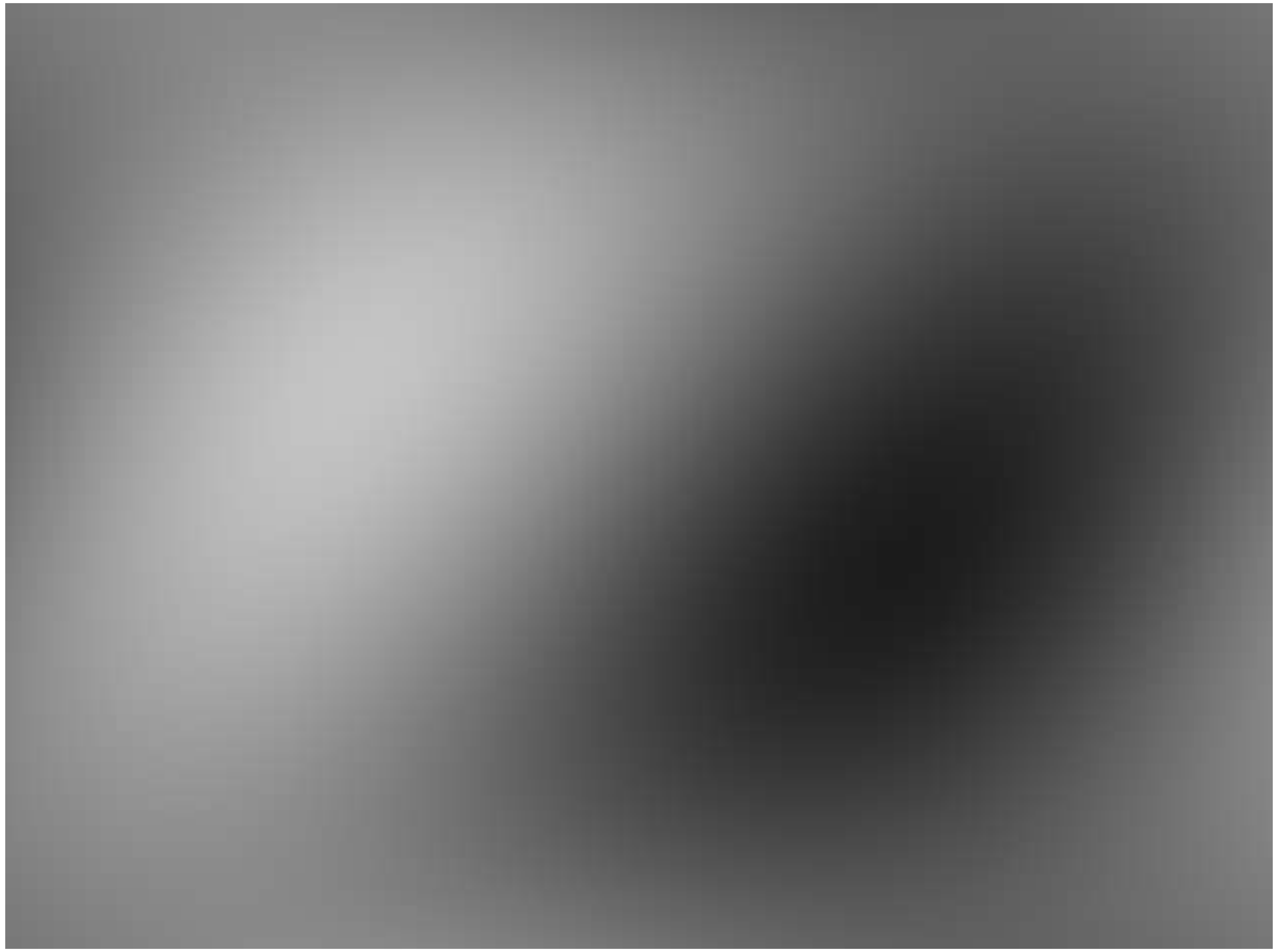
https://photo.stackexchange.com/questions/40401/what-does-frequency-mean-in-an-image/40410#40410

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

First (lowest) frequency, a.k.a. average

https://photo.stackexchange.com/questions/40401/what-does-frequency-mean-in-an-image/40410#40410

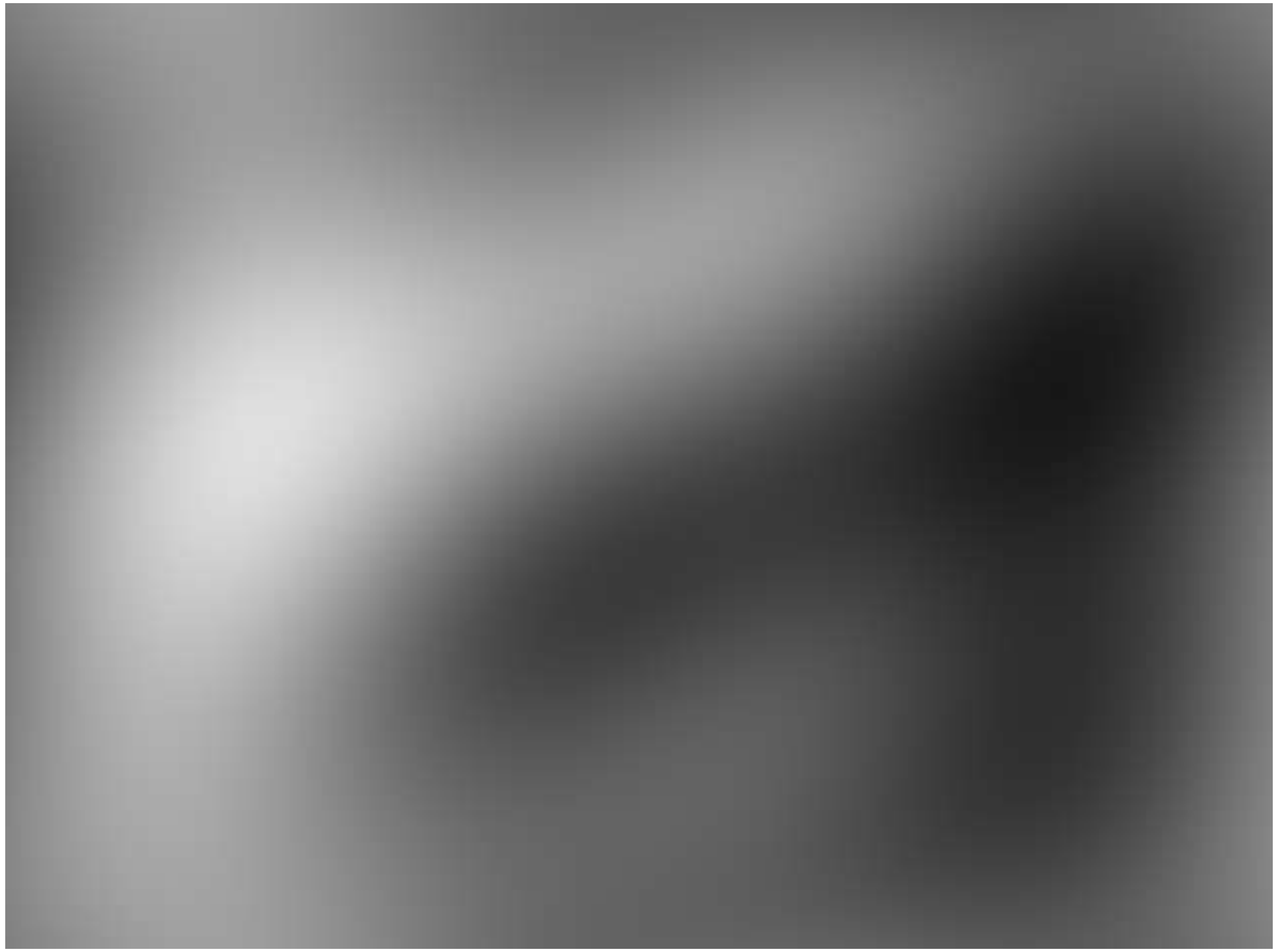
Fourier Transform (you will **NOT** be tested on this)



+ **Second** frequency

https://photo.stackexchange.com/questions/40401/what-does-frequency-mean-in-an-image/40410#40410

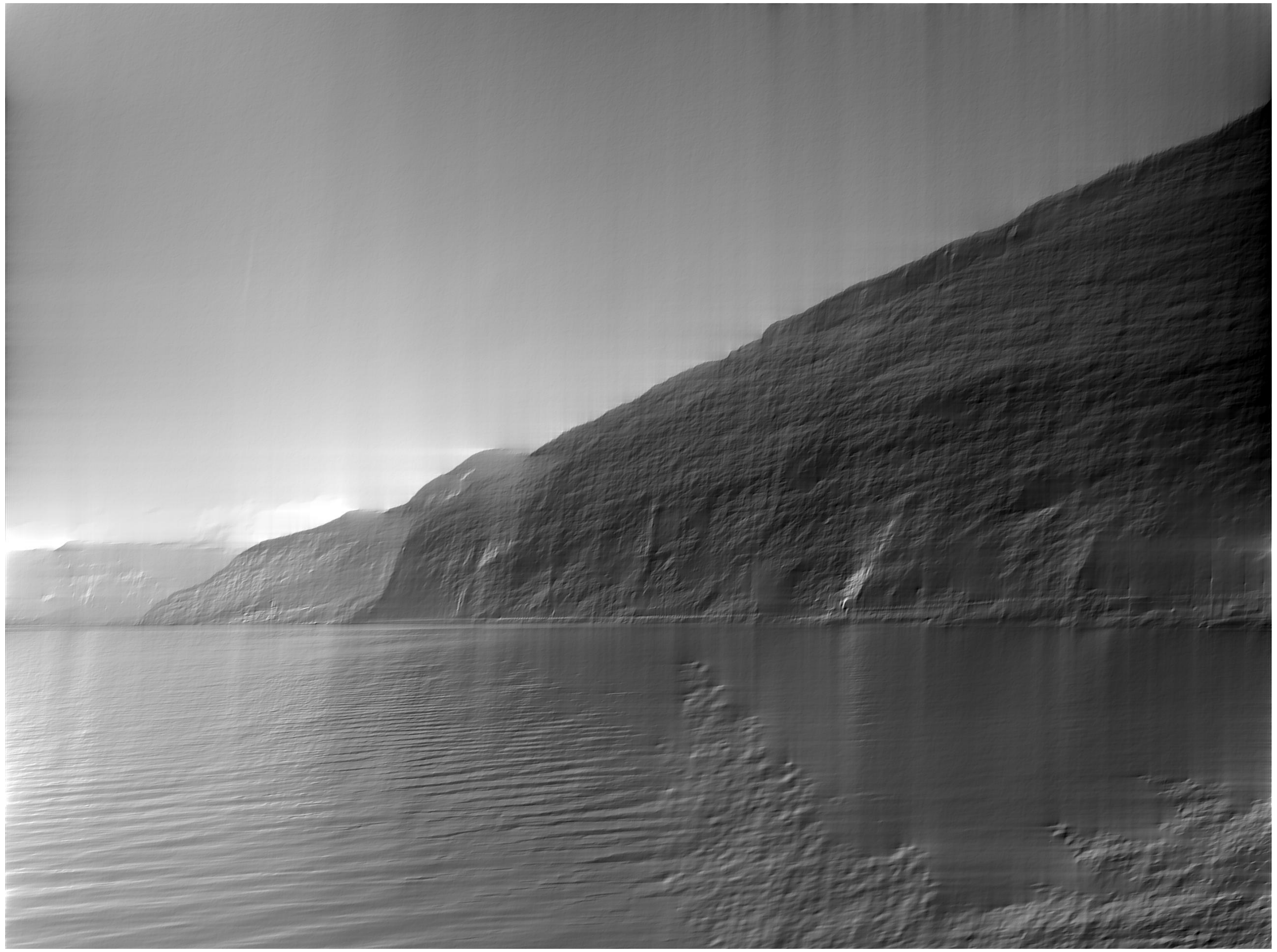
Fourier Transform (you will **NOT** be tested on this)



\+ **Third** frequency

https://photo.stackexchange.com/questions/40401/what-does-frequency-mean-in-an-image/40410#40410

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



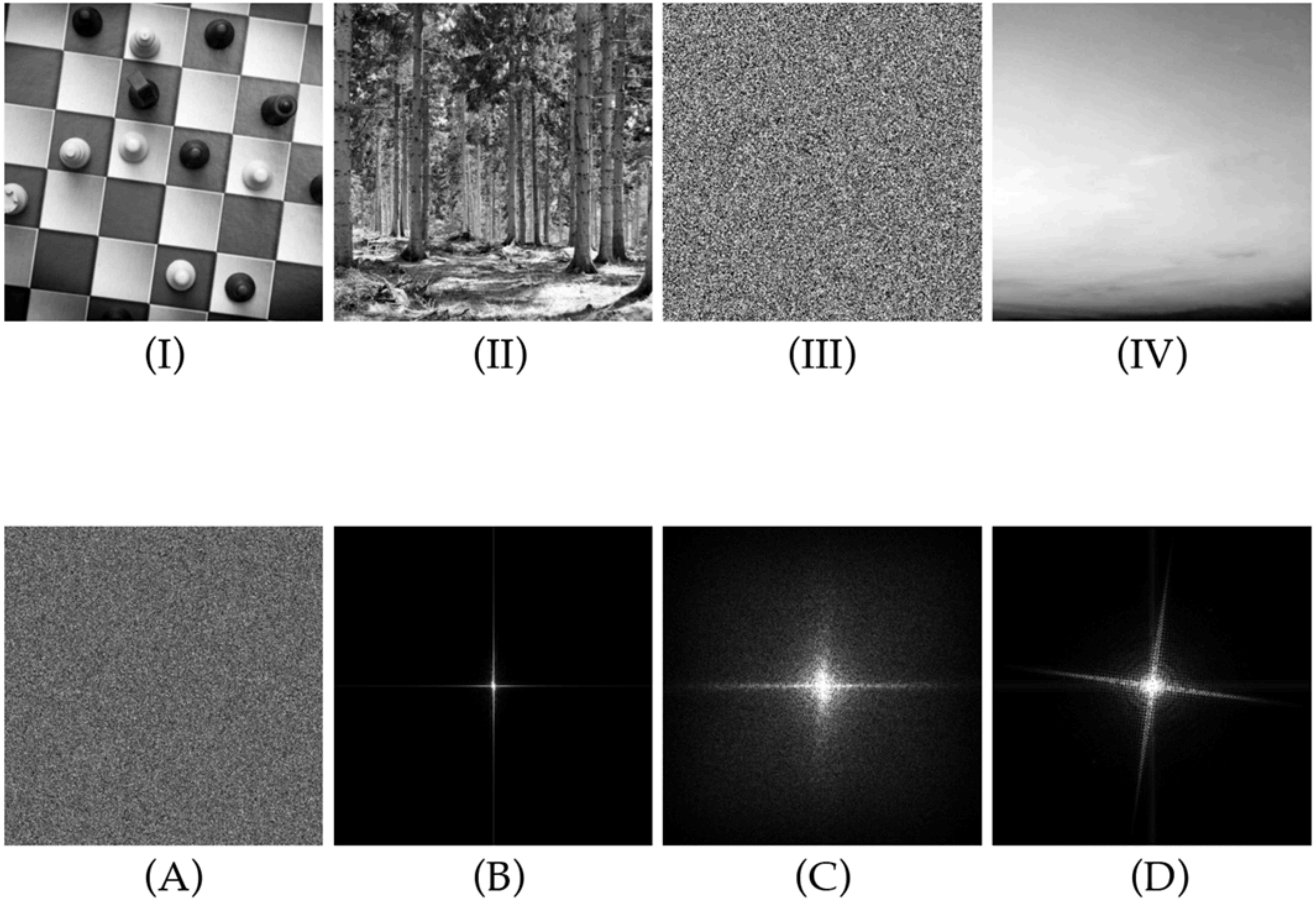
+ **50%** of frequencies

https://photo.stackexchange.com/questions/40401/what-does-frequency-mean-in-an-image/40410#40410

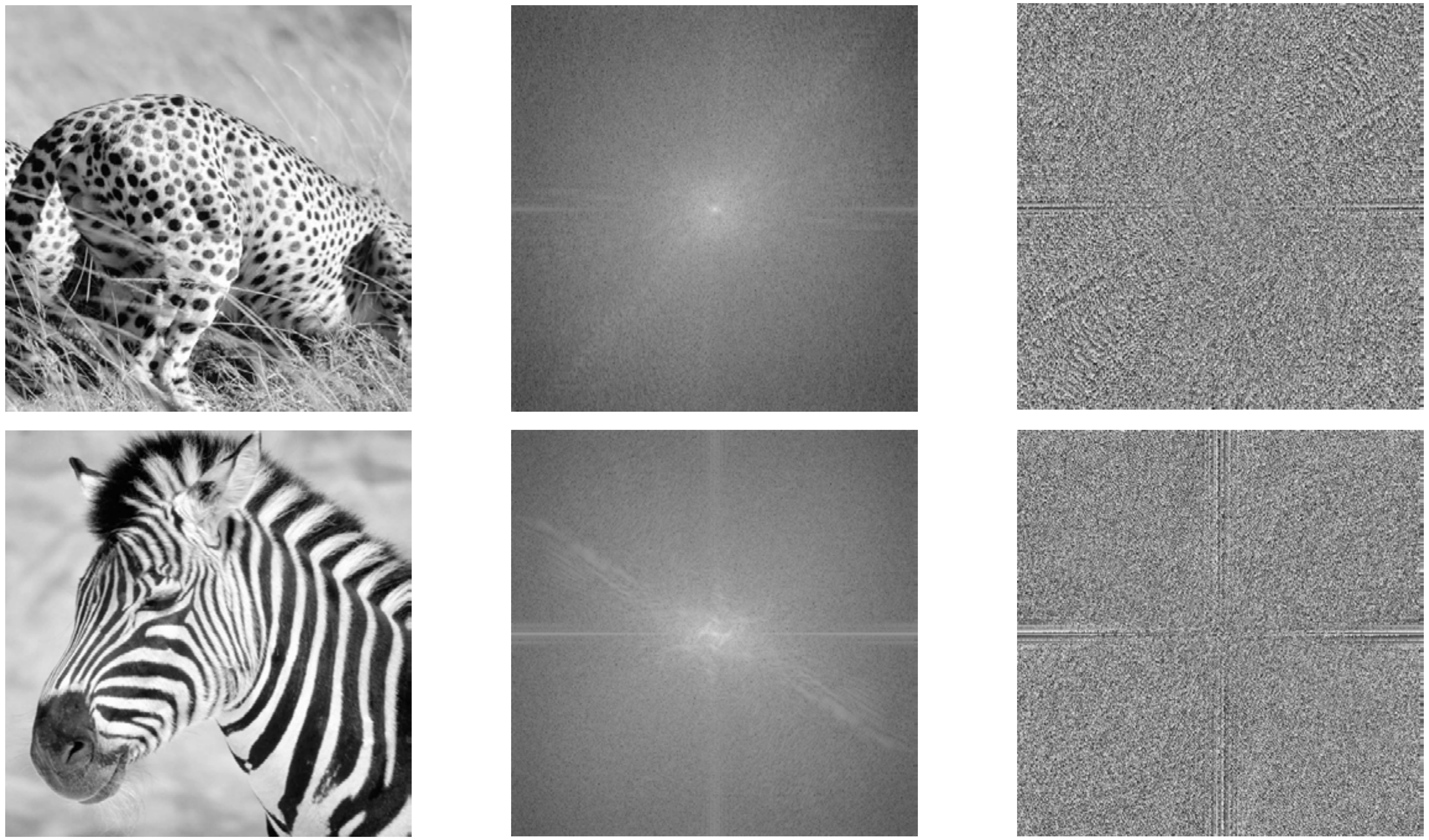
Fourier Transform (you will **NOT** be tested on this)

https://photo.stackexchange.com/questions/40401/what-does-frequency-mean-in-an-image/40410#40410

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



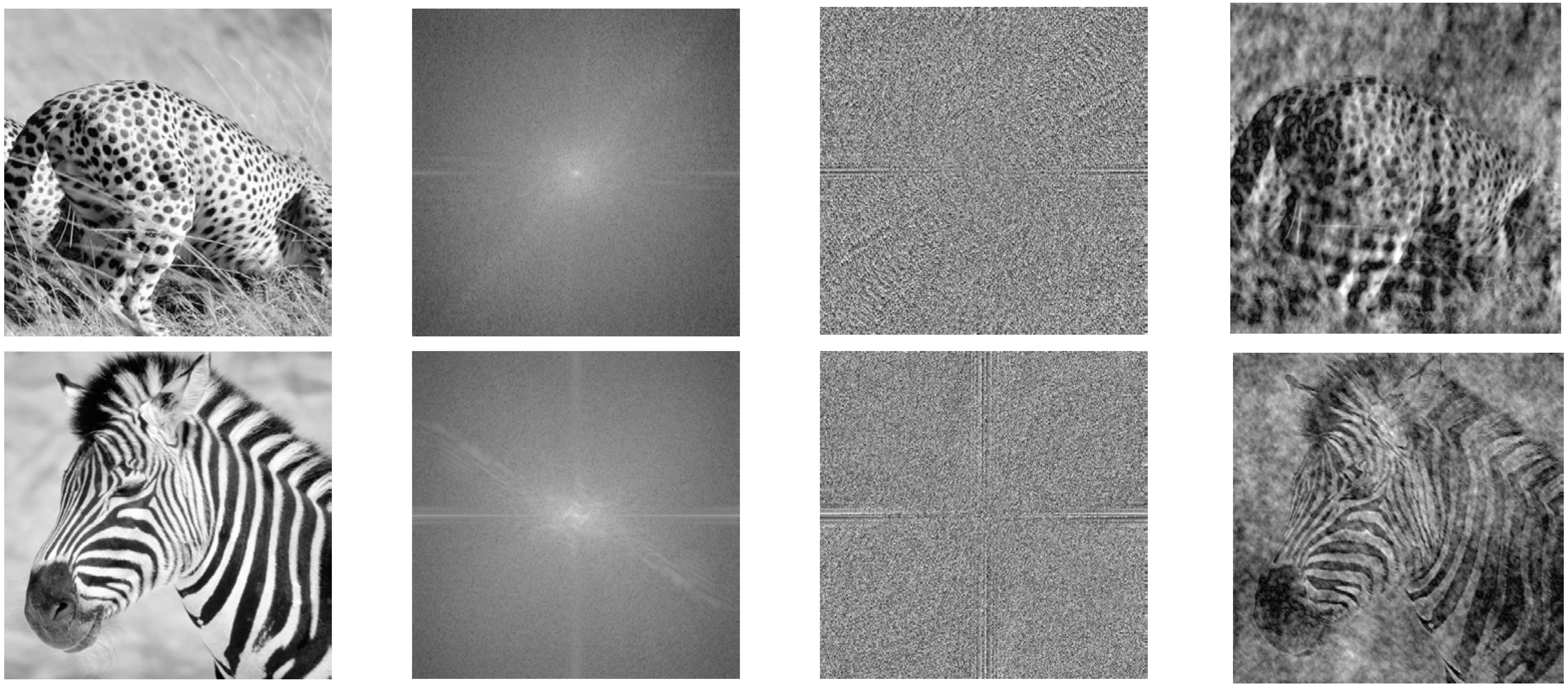
Fourier Transform (you will **NOT** be tested on this)



amplitude phase

Forsyth & Ponce (2nd ed.) Figure 4.6

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



cheetah phase with zebra amplitude

zebra phase with cheetah amplitude

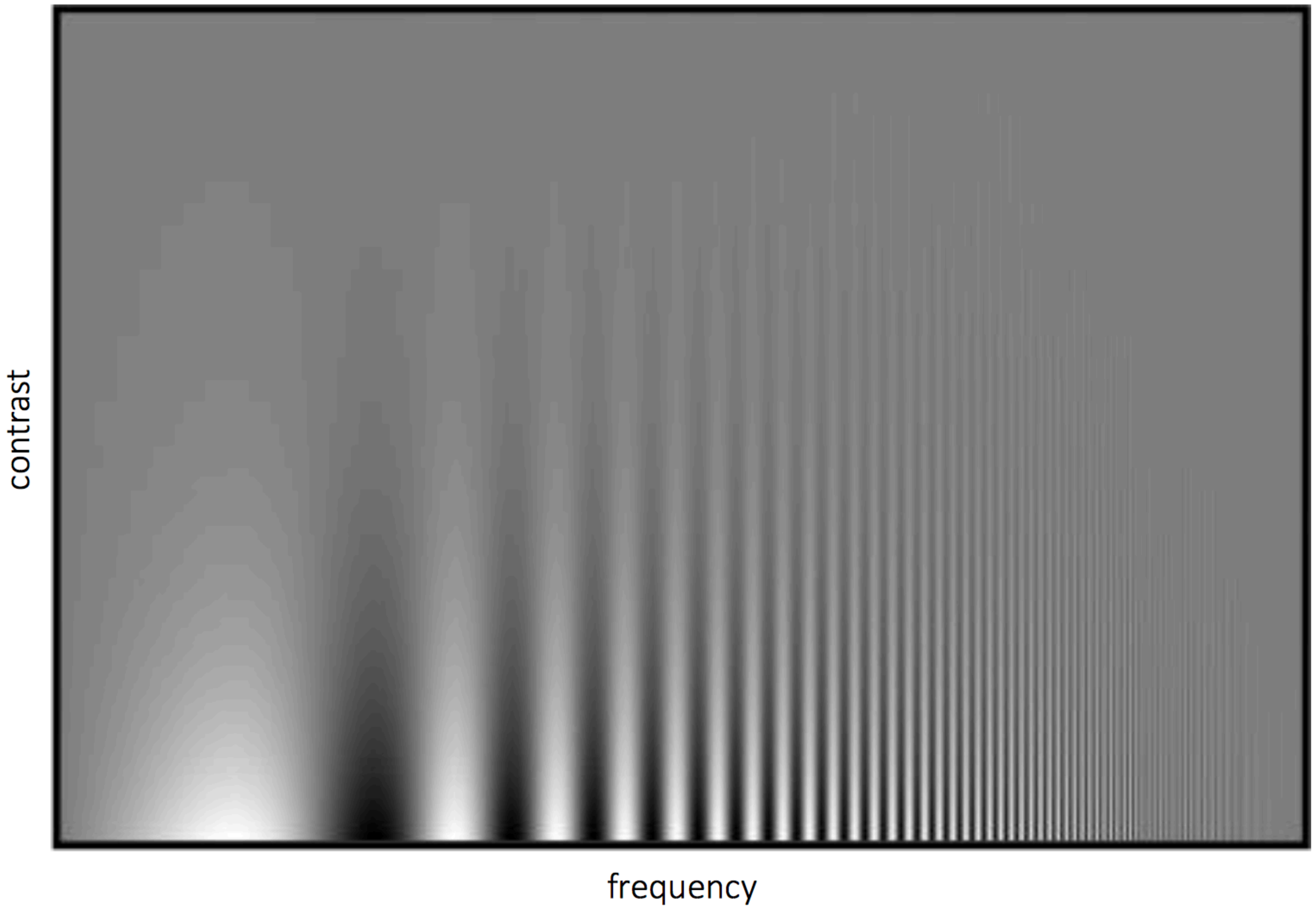
amplitude

phase

Forsyth & Ponce (2nd ed.) Figure 4.6

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

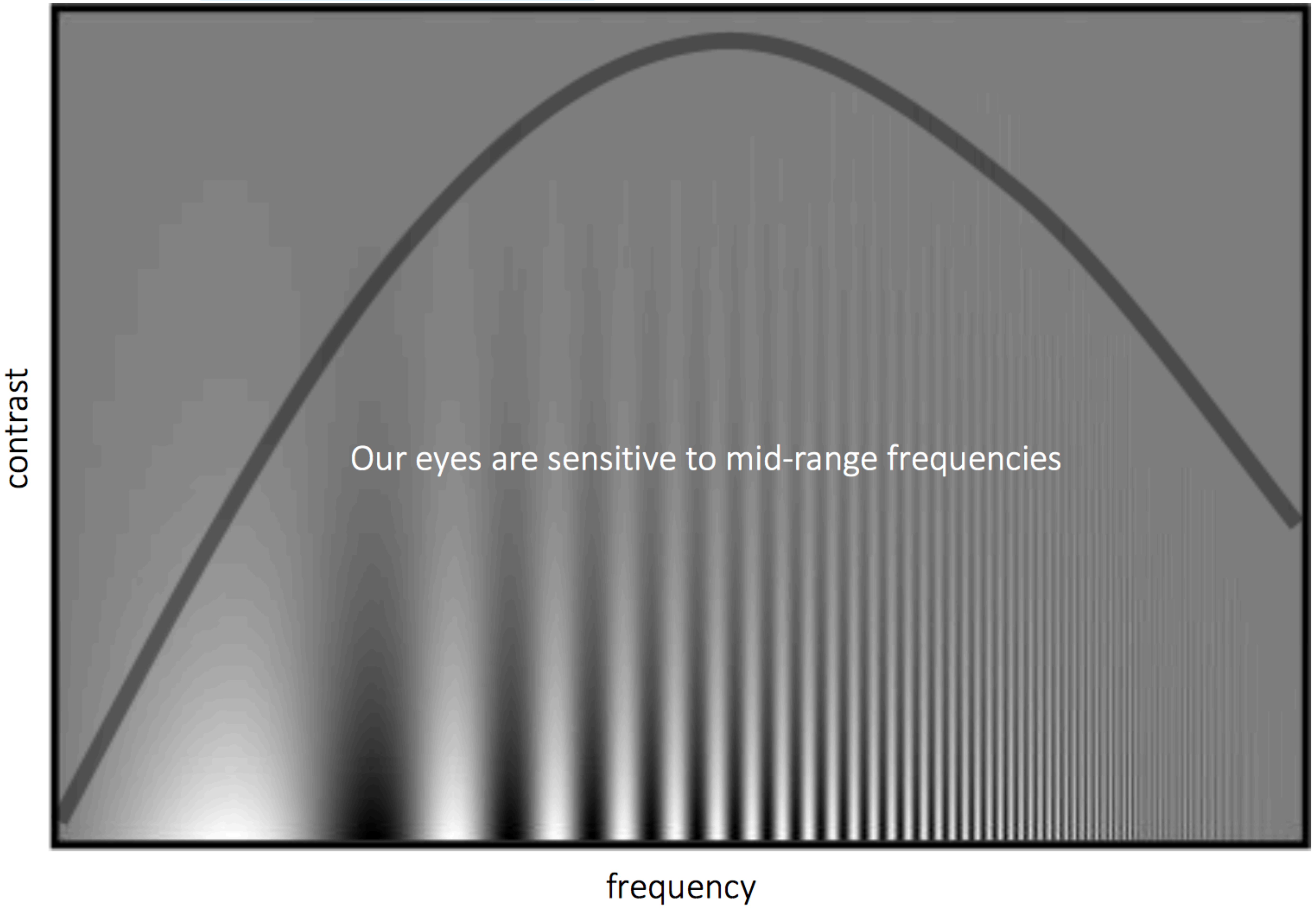
Experiment: Where of you see the stripes?



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

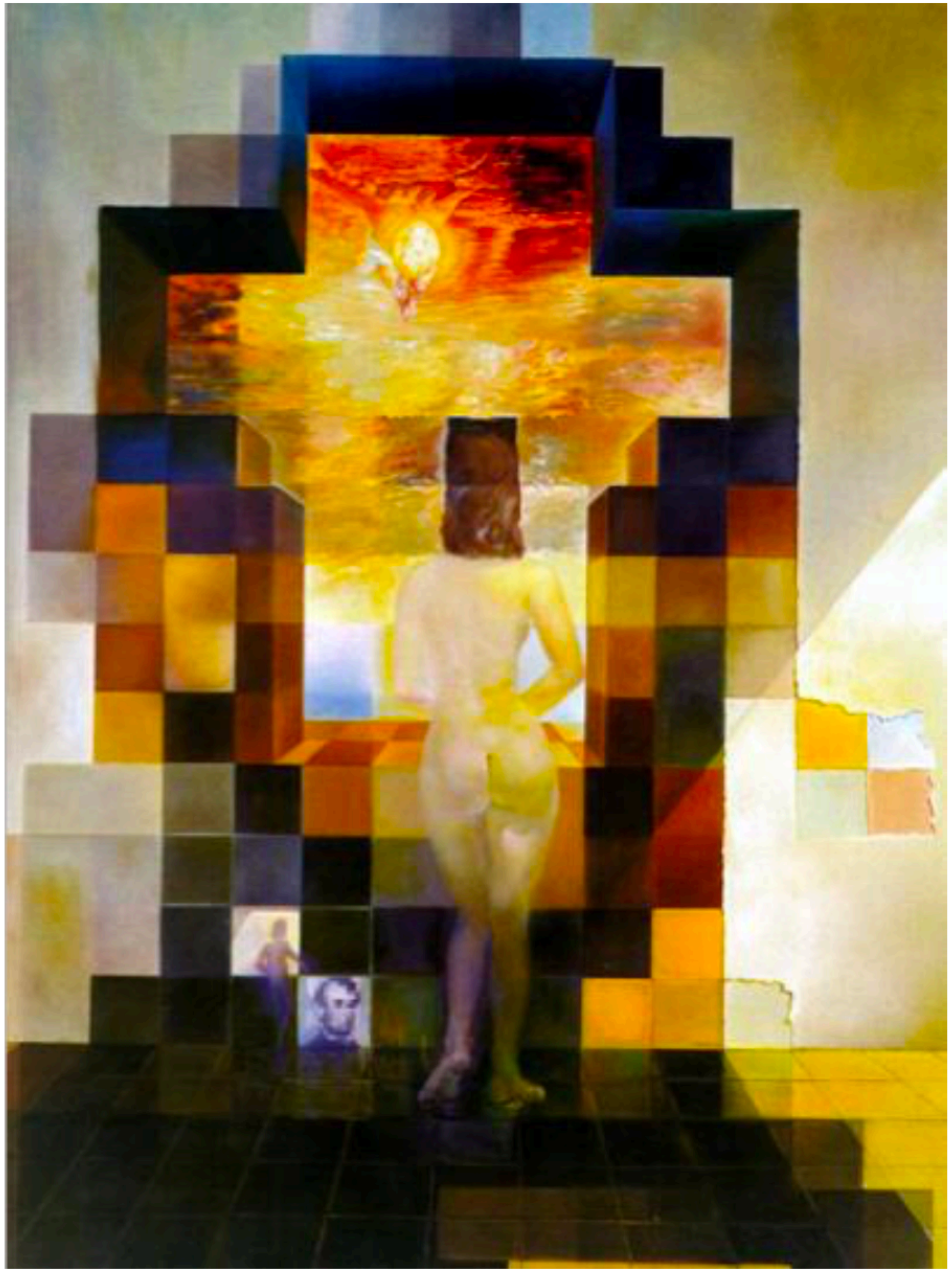
Campbell-Robson contrast sensitivity curve



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

What preceded was for fun

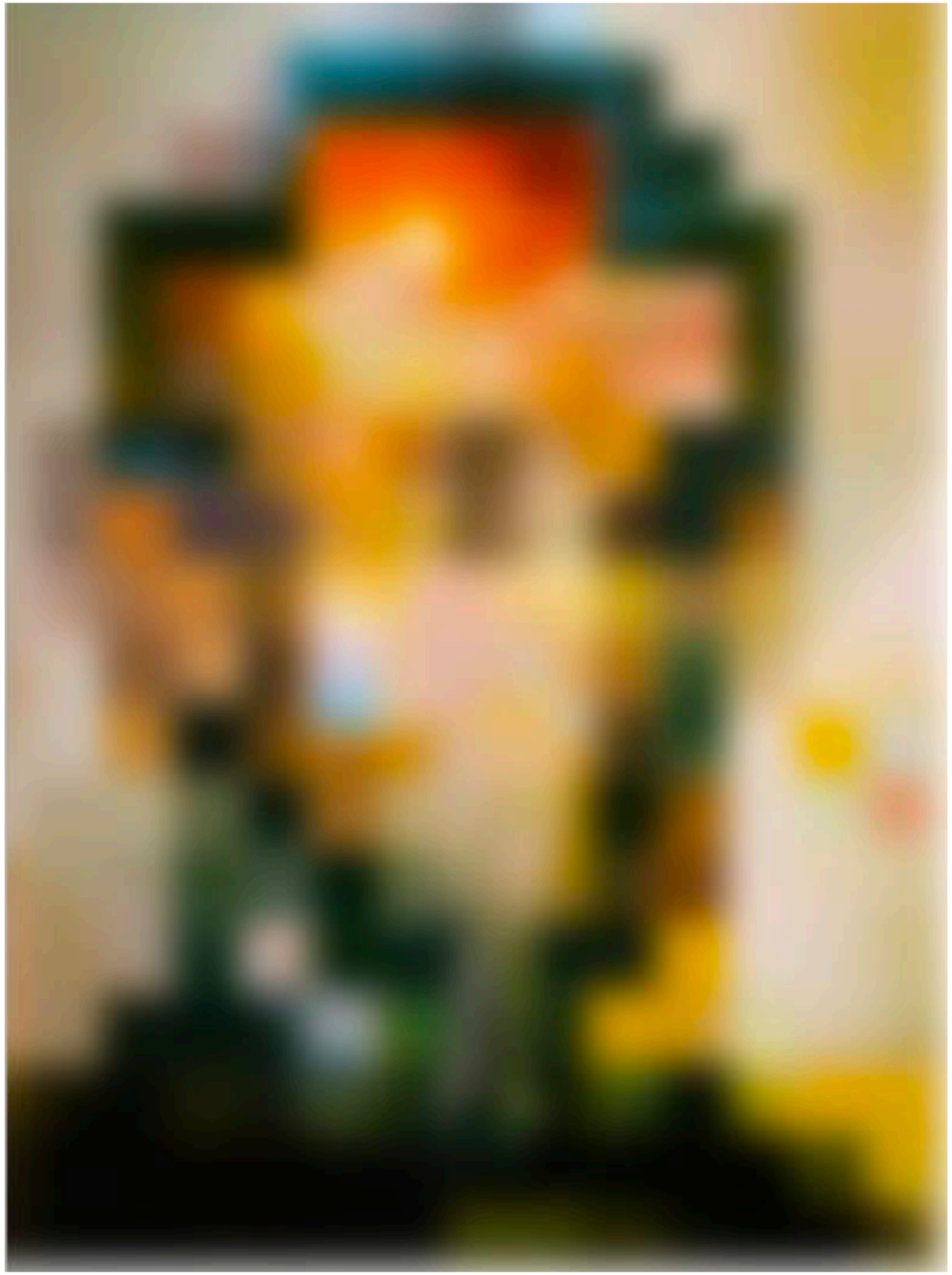
(you will **NOT** be tested on it)



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

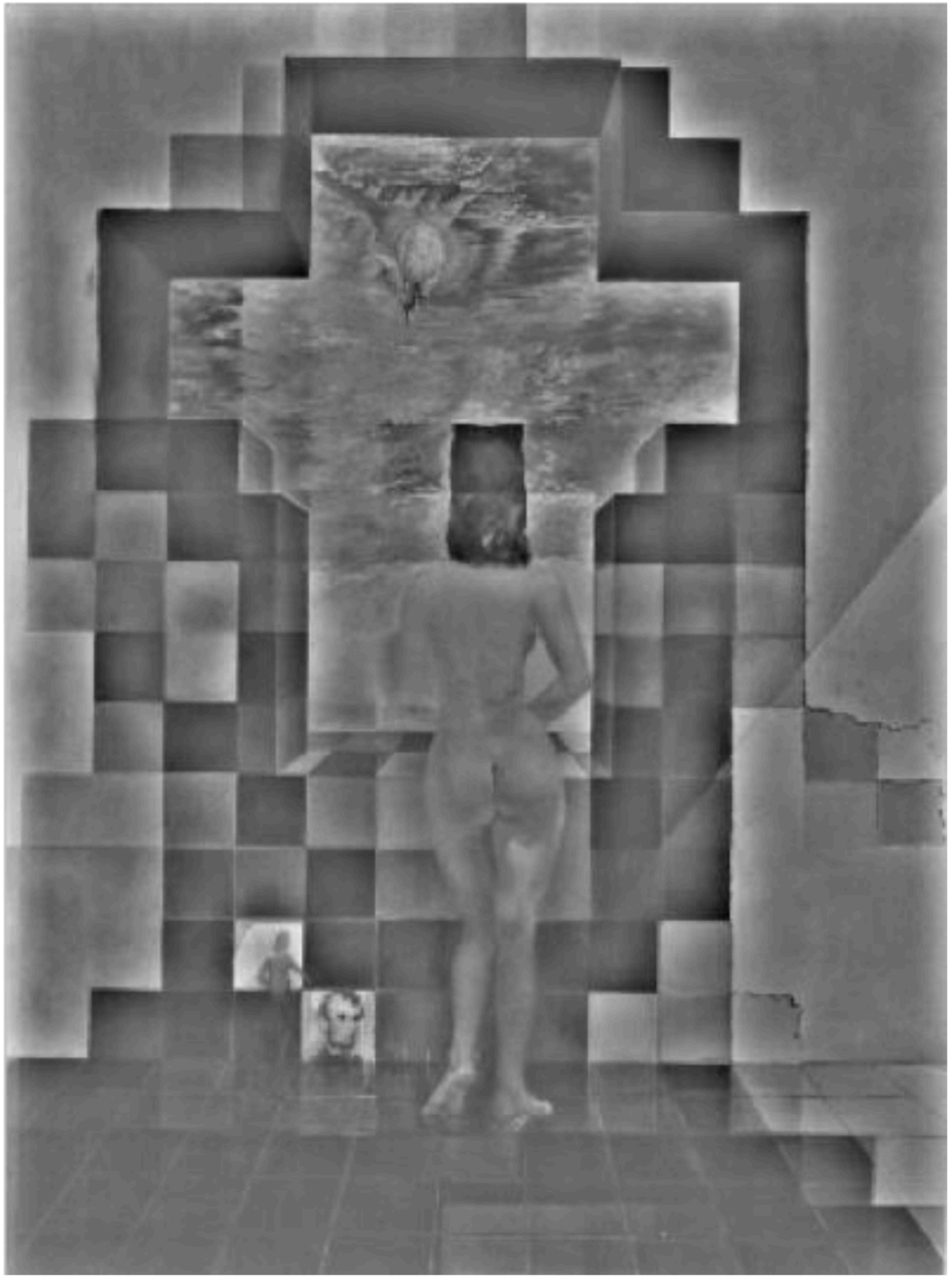
Salvador Dali, 1976

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)



Low-pass filtered version

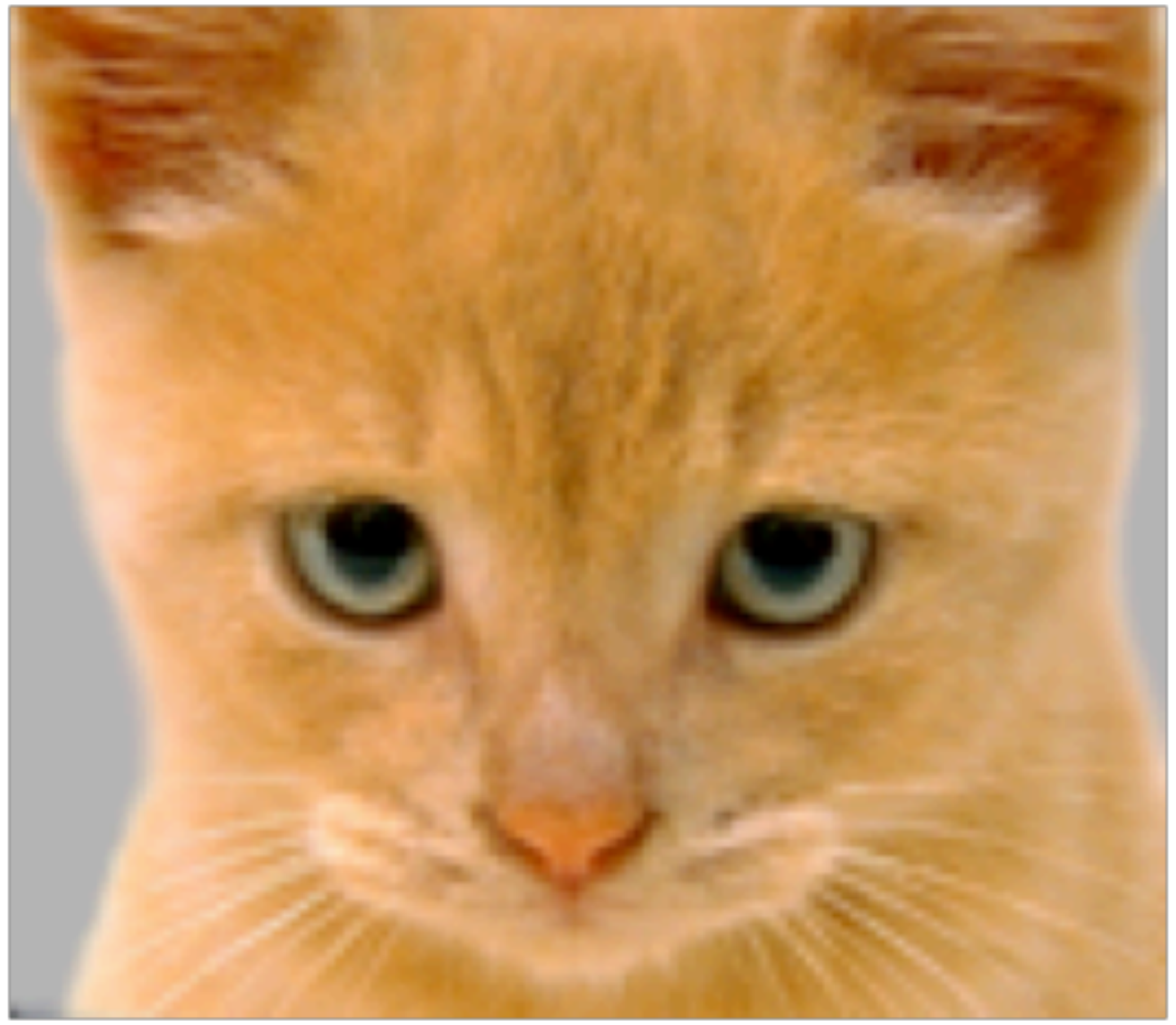
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)



High-pass filtered version

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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



Original

$I(x, y)$

Low-Pass Filter

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

High-Pass Filter

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