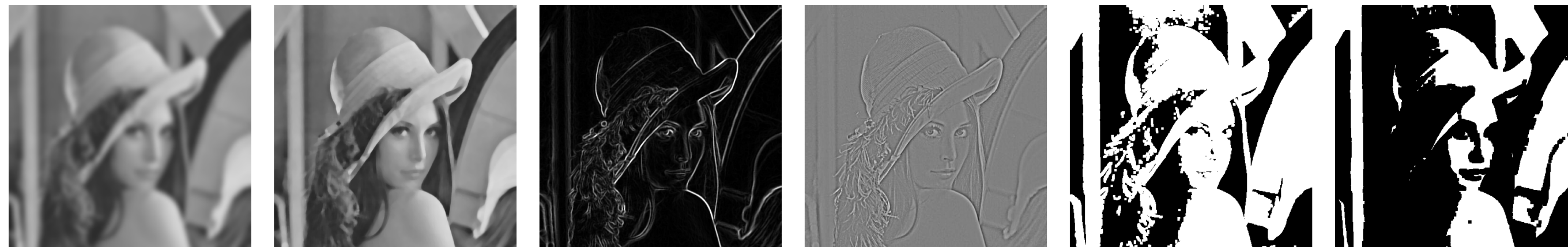




CPSC 425: Computer Vision



Lecture 5: Image Filtering (continued)

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

Menu for Today (September 19, 2024)

Topics:

- **Linear Filtering** recap + wrap up
- Efficient convolution, Fourier aside
- **Non-linear** Filters: Median, ReLU, Bilateral Filter

Readings:

- **Today's** Lecture: Szeliski 3.3-3.4, Forsyth & Ponce (2nd ed.) 4.4

Reminders:

- **Assignment 1:** Image Filtering and Hybrid Images due **September 24**

Today's “**fun**” Example: **Visual Question Answering**

<https://huggingface.co/spaces/nielsr/vilt-vqa>

Today's “**fun**” Example: **Clever Hans**



Clever Hans
(Orlov Trotter horse)

**Wilhelm
von Osten**

Today's “**fun**” Example: **Clever Hans**



Hans could get 89% of the math questions right

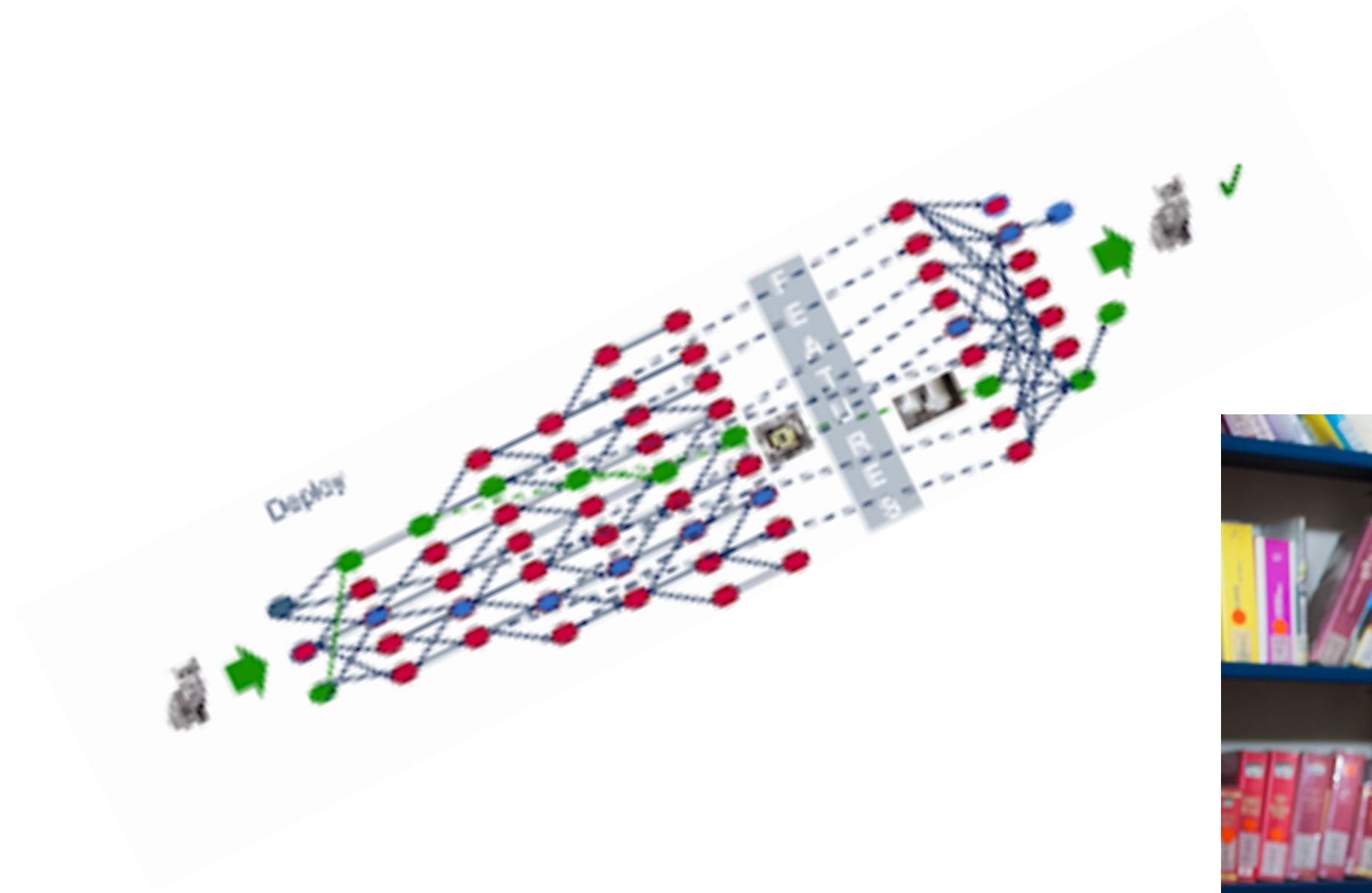
Today's “**fun**” Example: **Clever Hans**



The course was **smart**, just not in the way van Osten thought!

Hans could get 89% of the math questions right

Clever DNN



Visual Question Answering

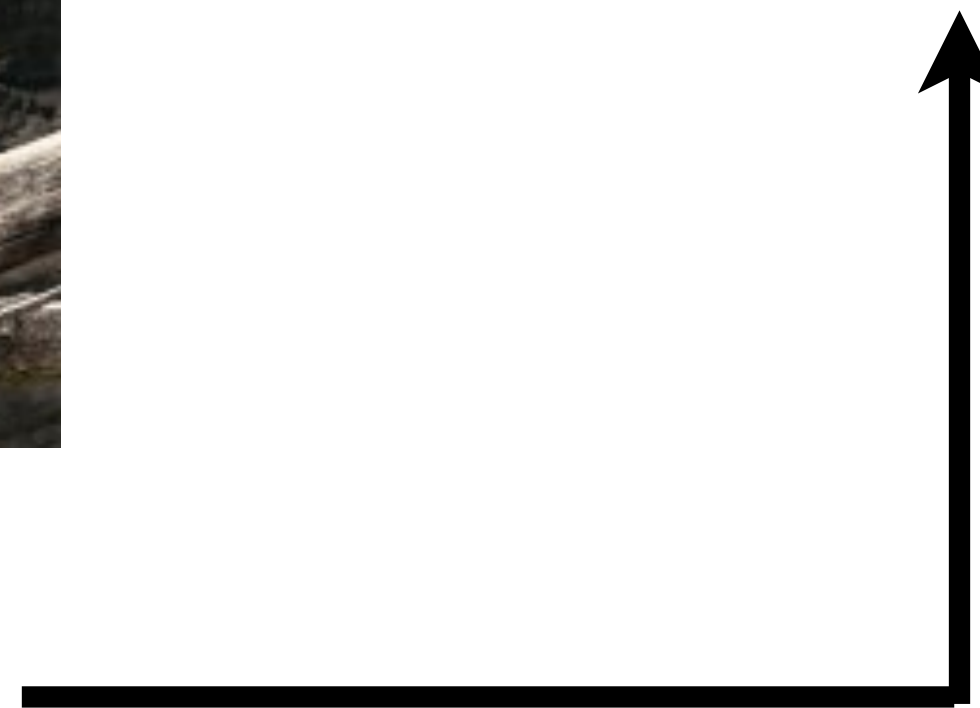


AI agent



Yes

Is there zebra climbing the tree?



iClicker Trial

(not graded)

Quiz 0, Question 1

Only one of the statements below is True. Which one?

- A) The answer to this question is A
- B) The answer to this question is B
- C) The answer stated in B) is True.
- D) The answer stated in A) is False
- E) There is no single answer that can be True among the above answers.

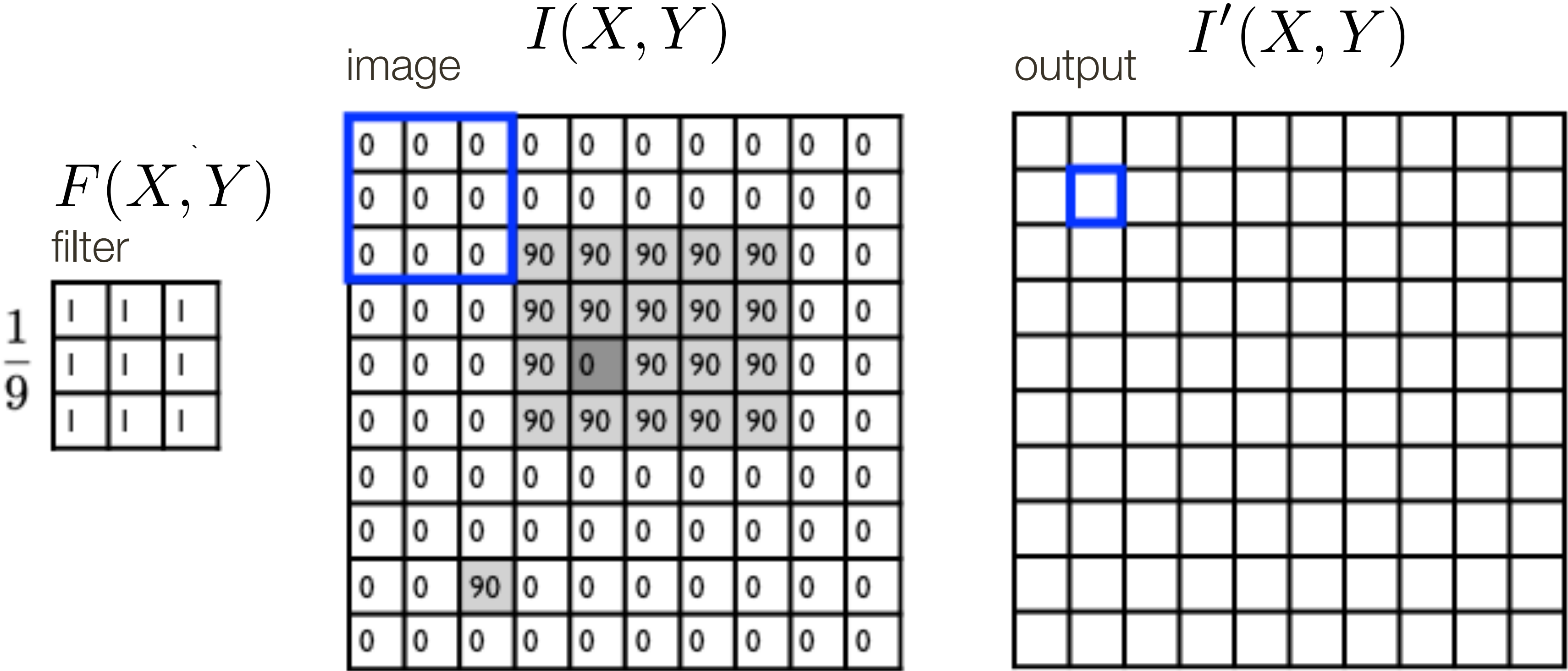
Quiz 0, Question 1

Only one of the statements below is True. Which one?

- A) The answer to this question is A
- B) The answer to this question is B
- C) The answer stated in B) is True.
- D) The answer stated in A) is False
- E) There is no single answer that can be True among the above answers.

Lecture 4 Recap

Lecture 4: Re-cap Linear Filter



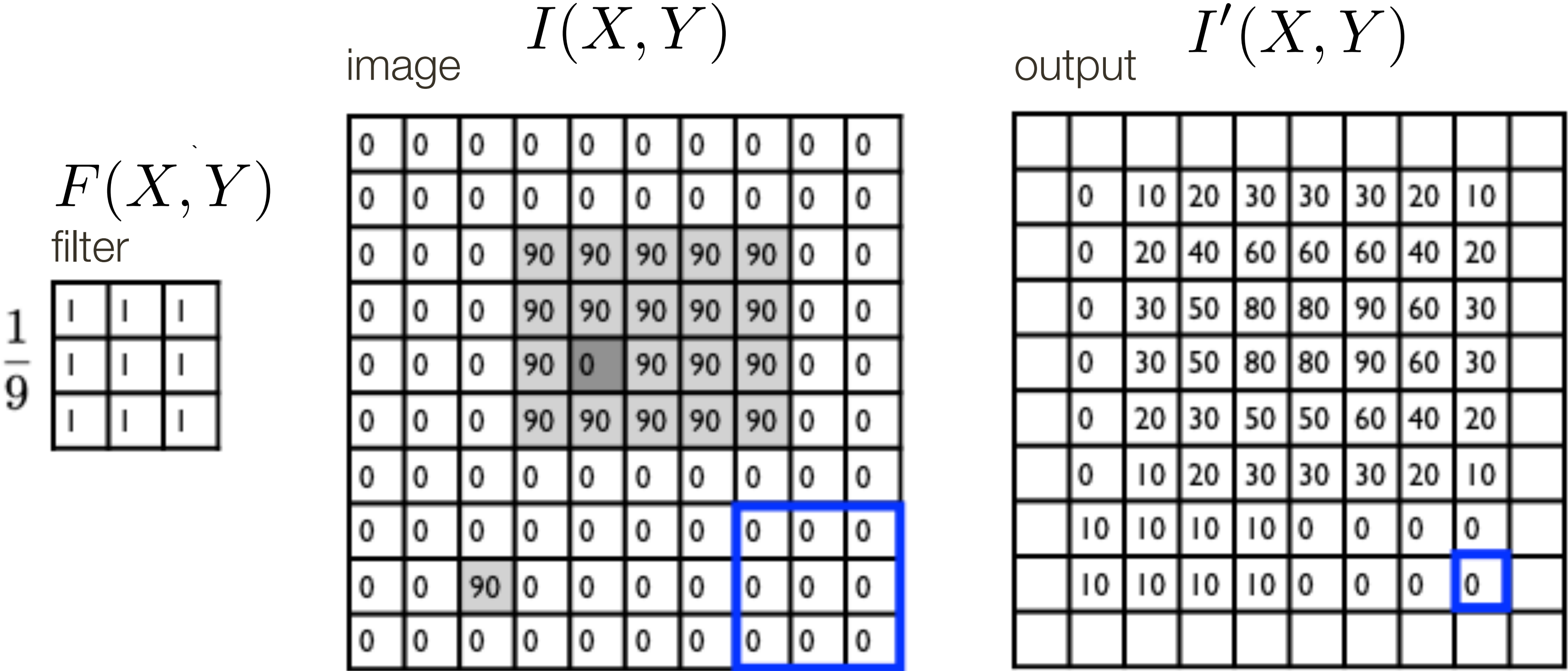
$I'(X, Y)$
output

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

$F(i, j)$
filter

$I(X + i, Y + j)$
image (signal)

Lecture 4: Re-cap Linear Filter



$I'(X, Y)$
output

$=$

$\sum_{j=-k}^k$

$\sum_{i=-k}^k$

$F(i, j)$
filter

$I(X + i, Y + j)$
image (signal)

Lecture 4: 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)

!	4	6
j	e	p
c	q	a

a	b	c
d	e	f
g	h	i

Filter

1	2	3
4	5	6
7	8	9

Image

Output

$$\begin{aligned} &= 9a + 8b + 7c \\ &\quad + 6d + 5e + 4f \\ &\quad + 3g + 2h + 1i \end{aligned}$$

Lecture 4: Re-cap 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)

Lecture 4: Re-cap Smoothing Filters

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

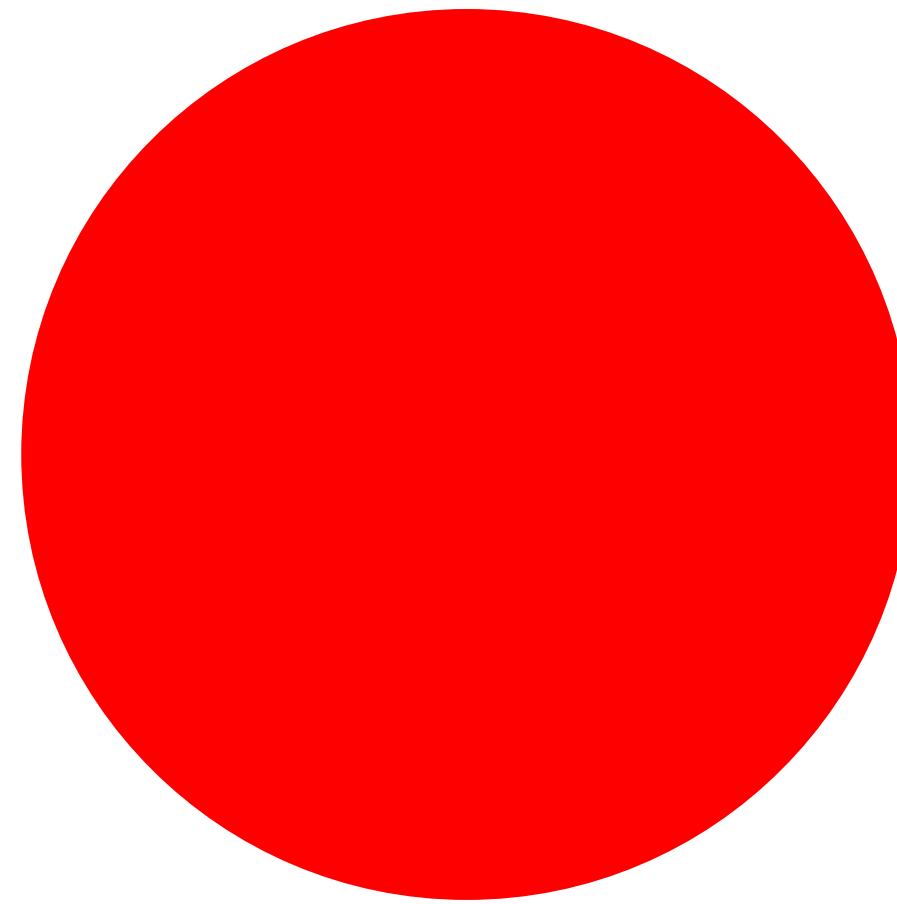
Low-pass Filtering = “Smoothing” / “Blurring”

Box Filter

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

1	1	1
1	1	1
1	1	1

Pillbox Filter



$$\frac{1}{256}$$

Gaussian Filter

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

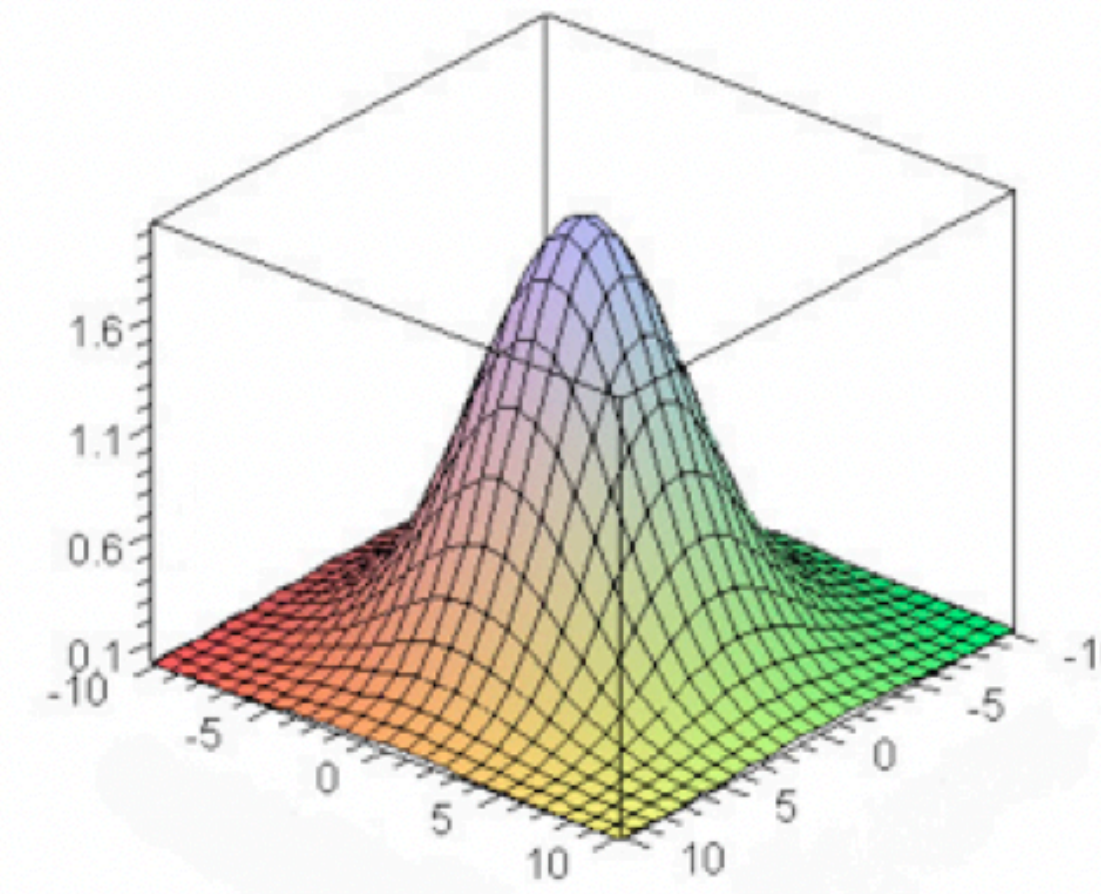
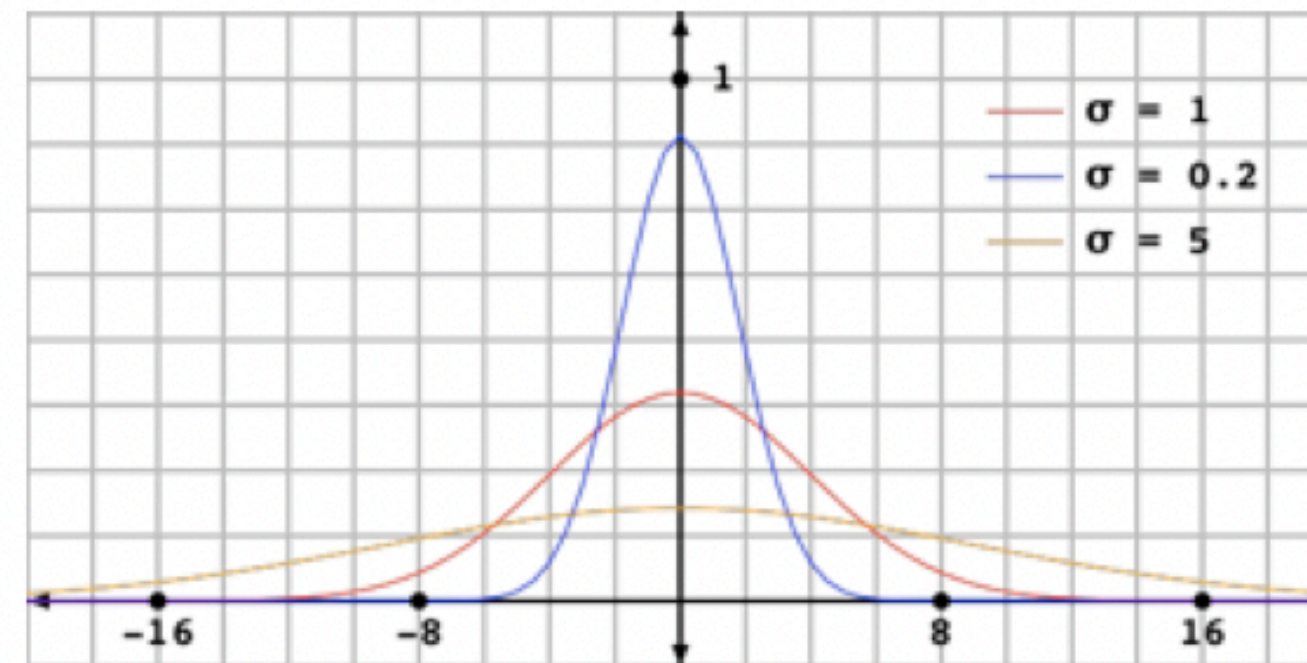
All of these filters are **Low-pass Filters**

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

Lecture 4: Re-cap **Gaussian** Smoothing Filter

Gaussian kernels are often used for smoothing and resizing images

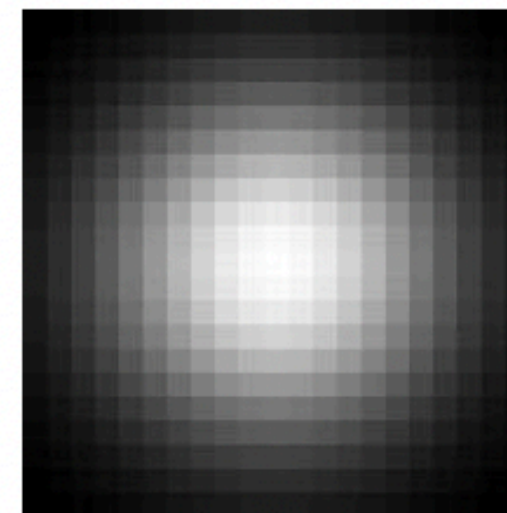
1D



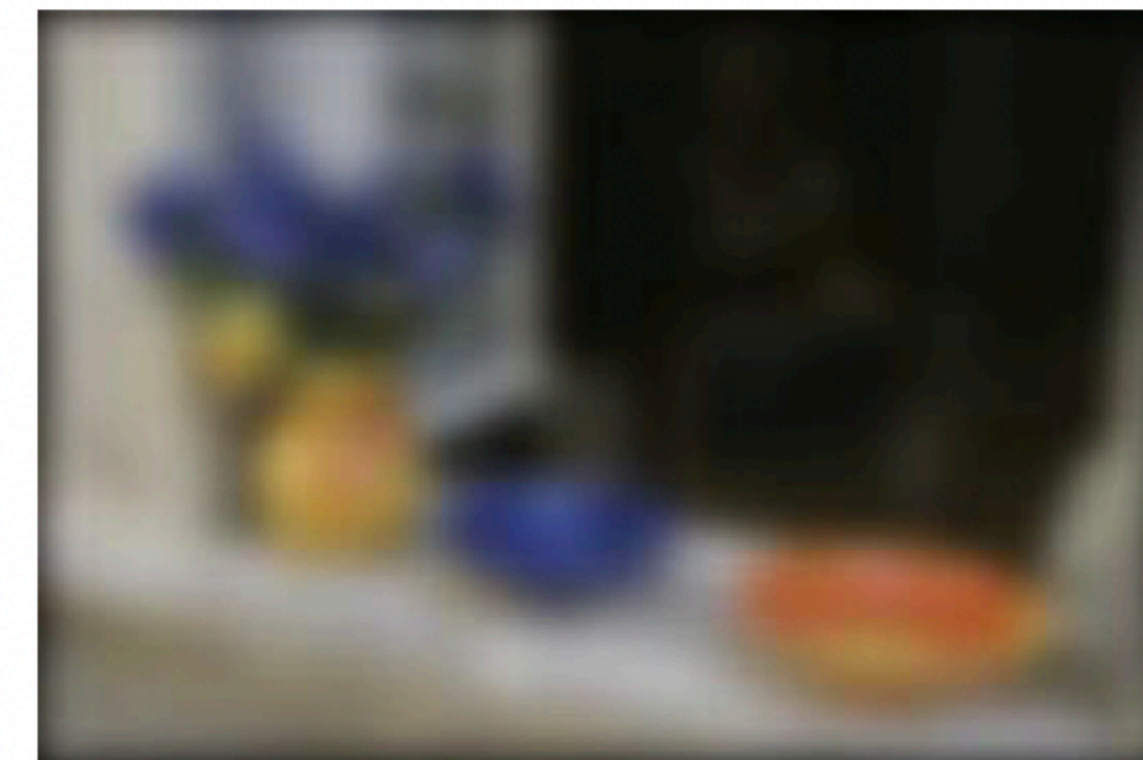
2D



*



=



Lecture 4: Re-cap 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 $\sigma = 1 \Rightarrow 7 \times 7$ filter

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: How do you know if filter is separable?

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

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

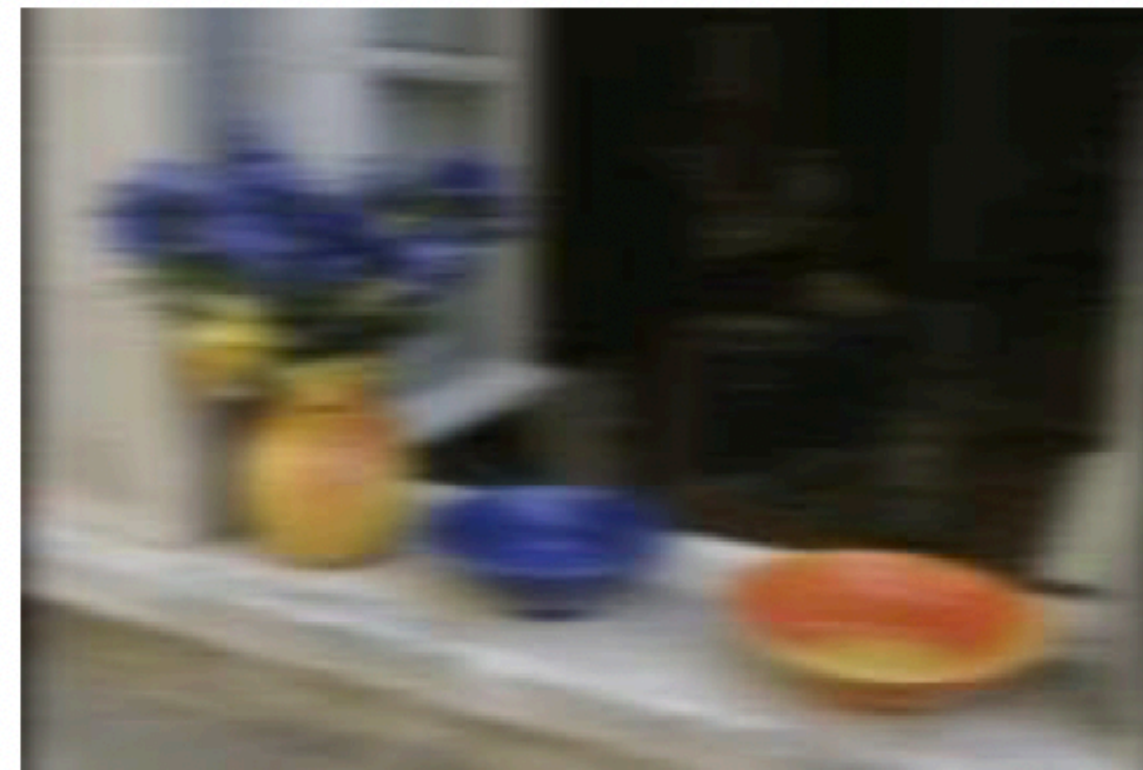
Separability: How do you know if filter is separable?

Mathematically: Rank of filter matrix is 1 (recall rank is number of linearly independent row vectors)

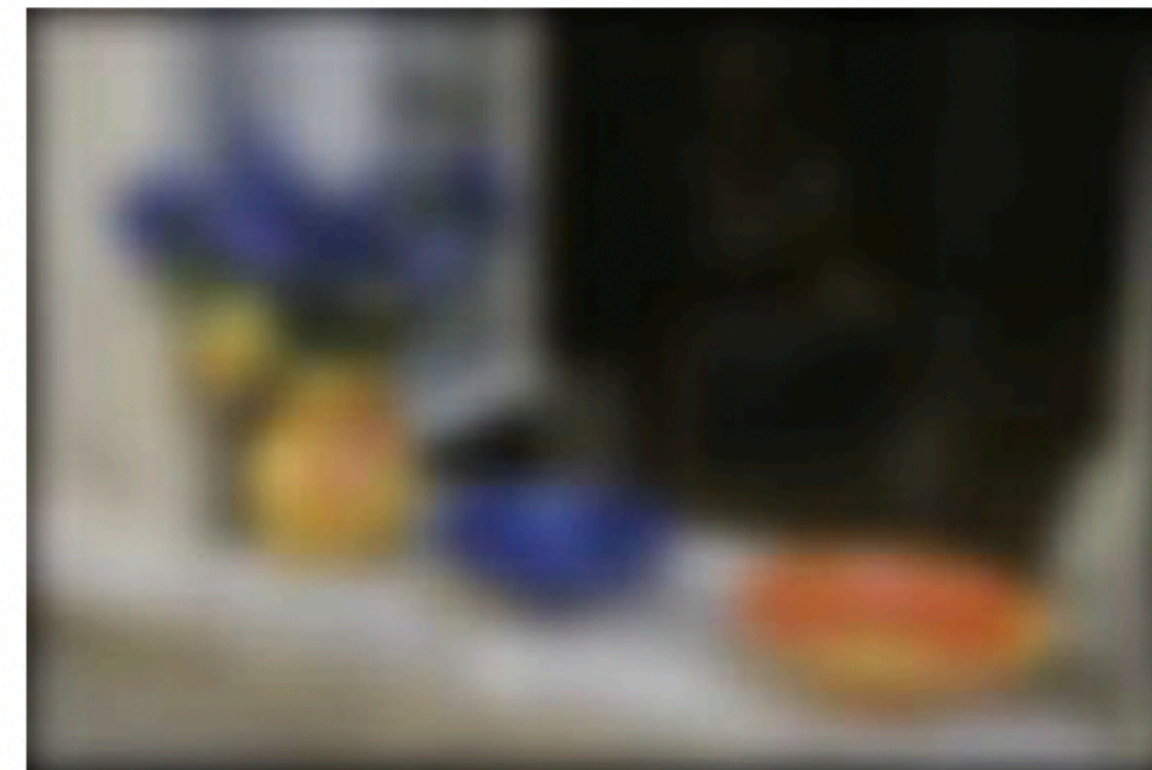
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \odot \frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Separable Gaussian Filter

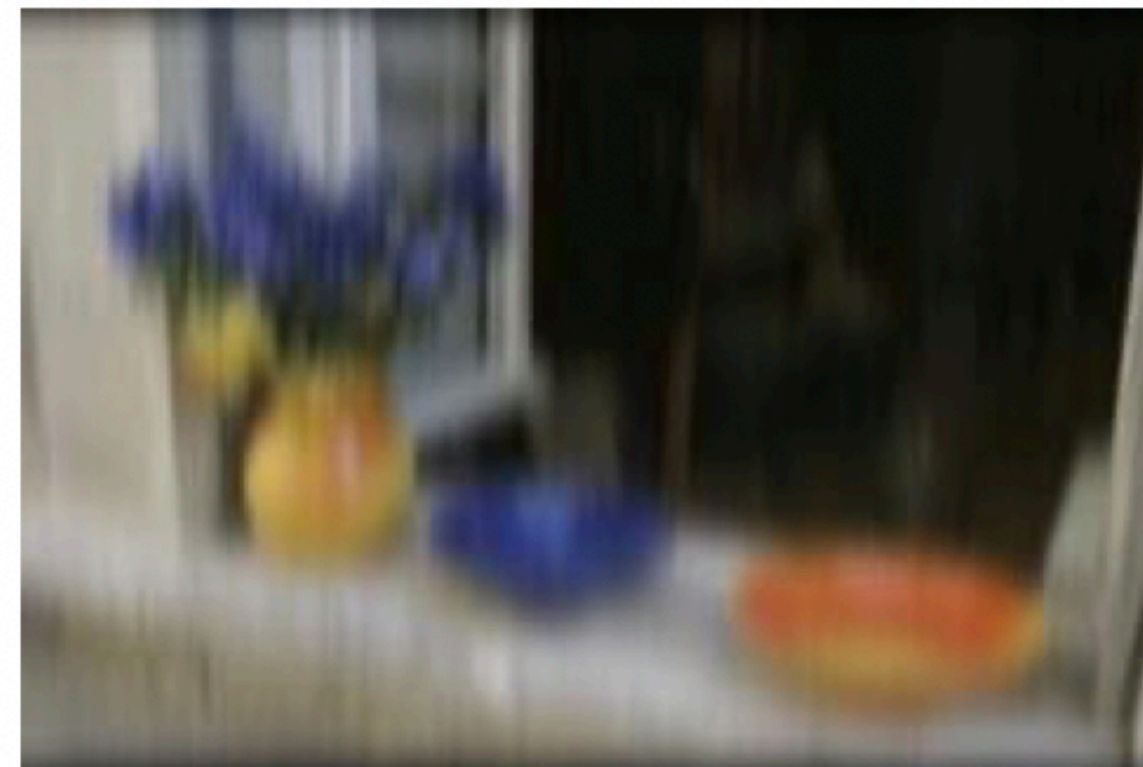
2D Gaussian blur by horizontal/vertical blur



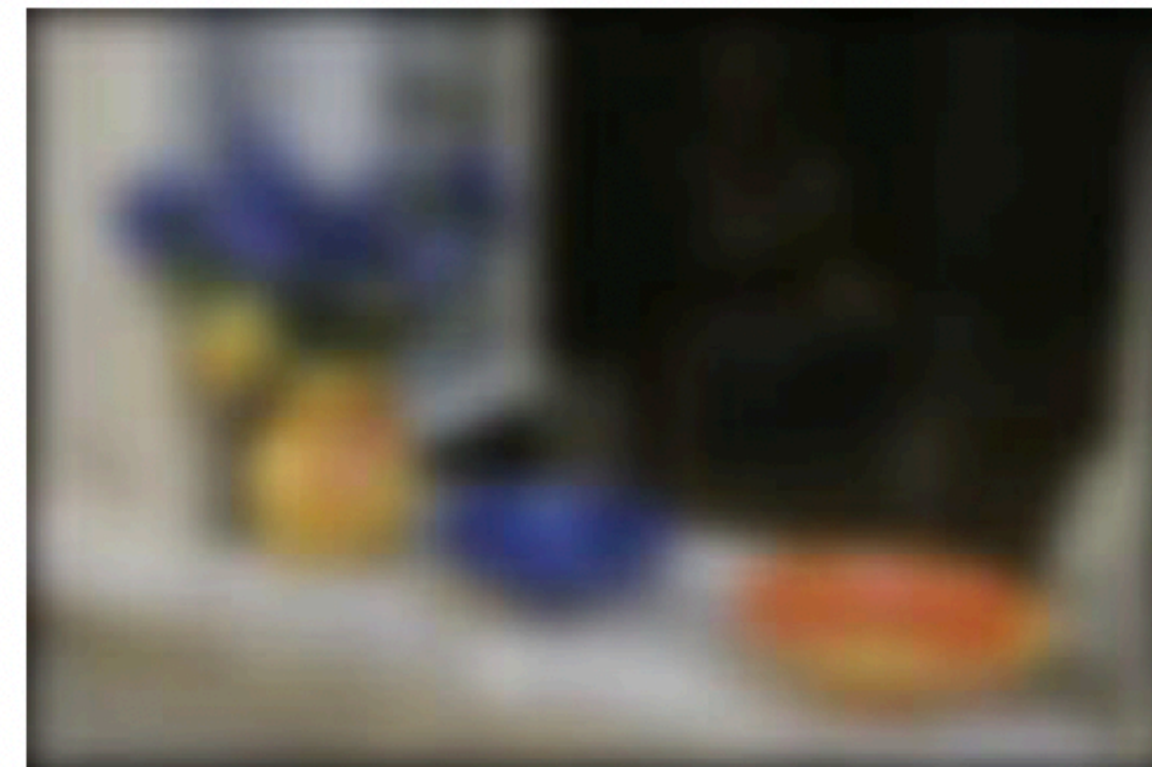
horizontal



vertical



vertical₂



horizontal

Efficient Implementation: **Separability**

Naive implementation of 2D **Filtering**:

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 **Filtering**:

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 **Filter**:

At each pixel, (X, Y) , there are $2m$ multiplications

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

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

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

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

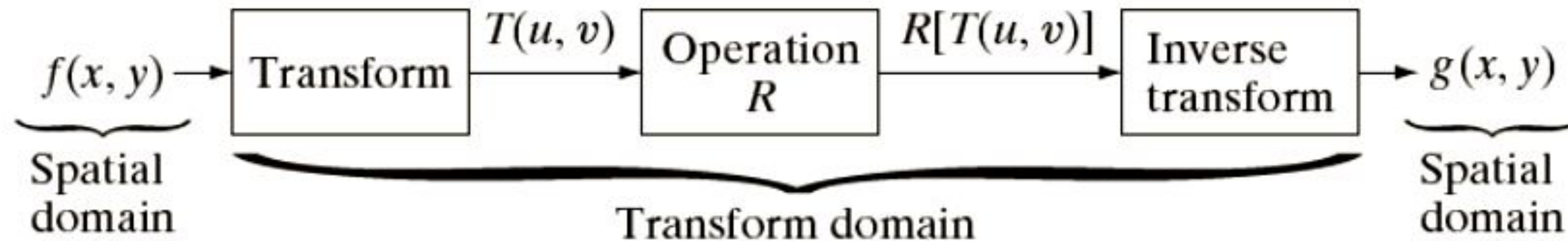
Therefore.

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

Interpretation: At the expense of two $\ln()$ and one $\exp()$ computations, multiplication is reduced to addition ... silly I know

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**:

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

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

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

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

General implementation of **convolution**:

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

Convolution if FFT space:

Cost of FFT/IFFT for image: $\mathcal{O}(n^2 \log n)$

Cost of FFT/IFFT for filter: $\mathcal{O}(m^2 \log m)$

Cost of convolution: $\mathcal{O}(n^2)$

Note: not a function of filter size !!!

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 any periodic signal you want!

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

Basic building block:

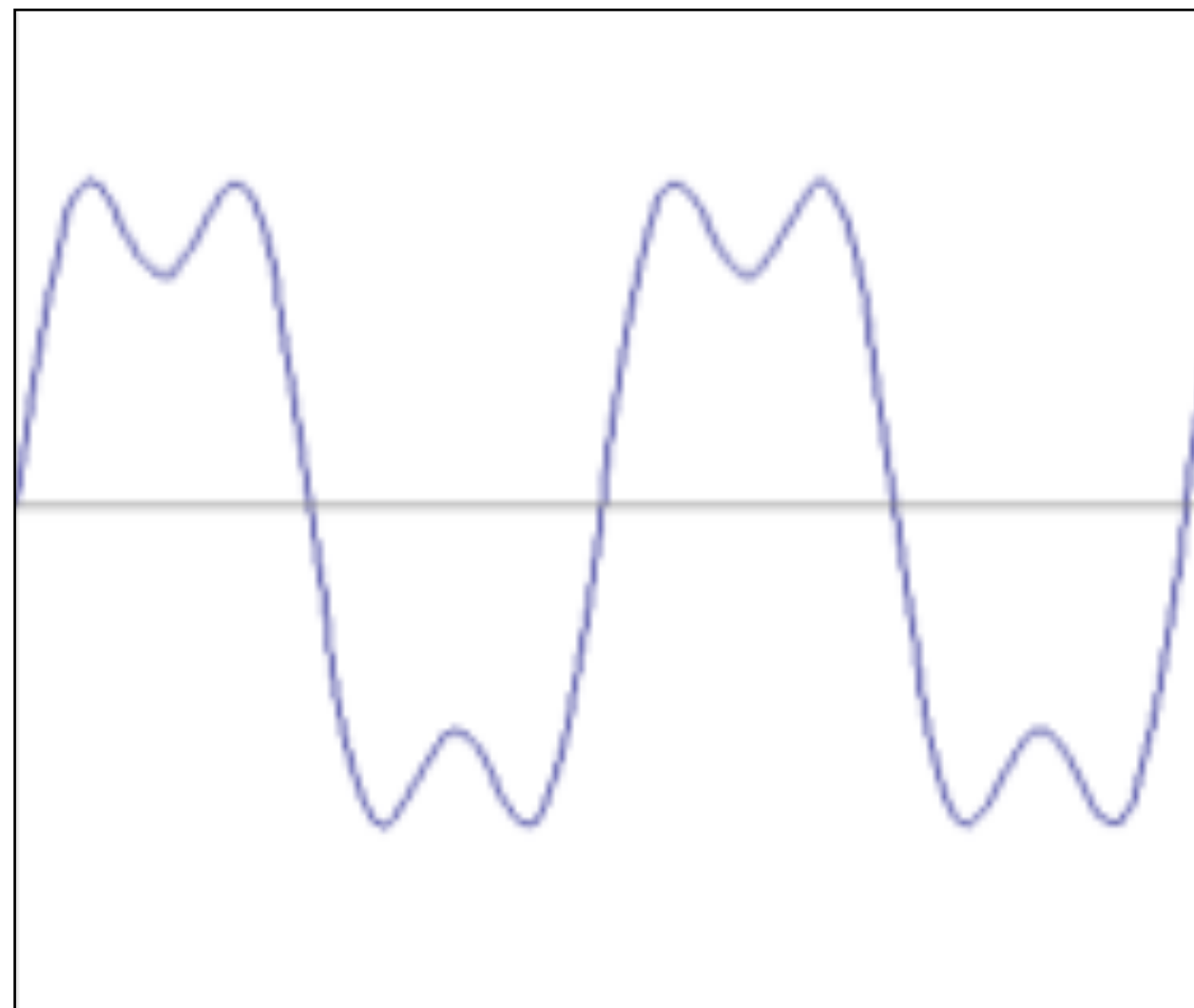
$$A \sin(\omega x + \phi)$$

A diagram showing the equation $A \sin(\omega x + \phi)$ with five labels and arrows pointing to its components: 'amplitude' points to A , 'sinusoid' points to $\sin$, 'angular frequency' points to ω , 'variable' points to x , and 'phase' points to ϕ .

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

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

How would you generate this function?



=

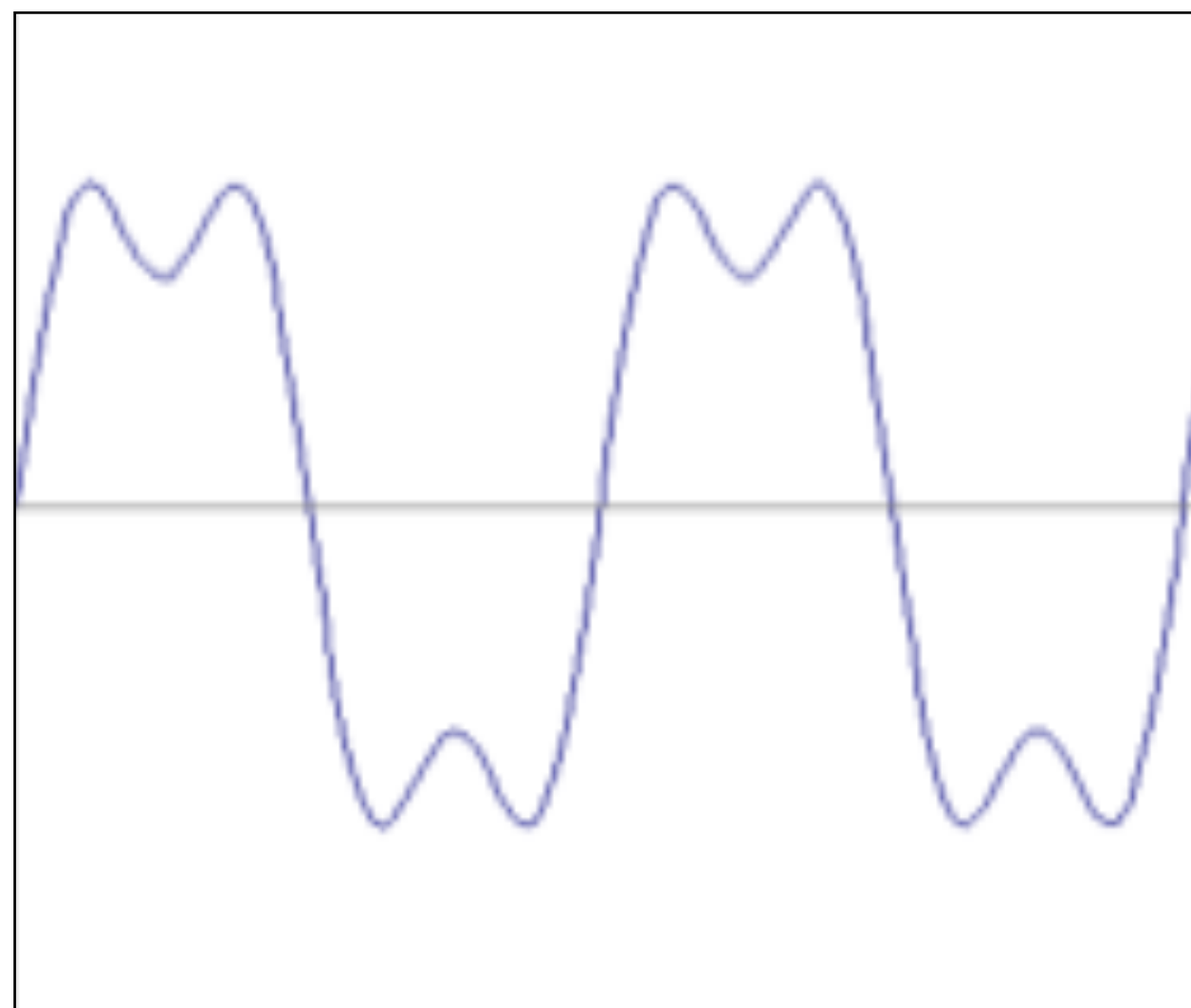
?

+

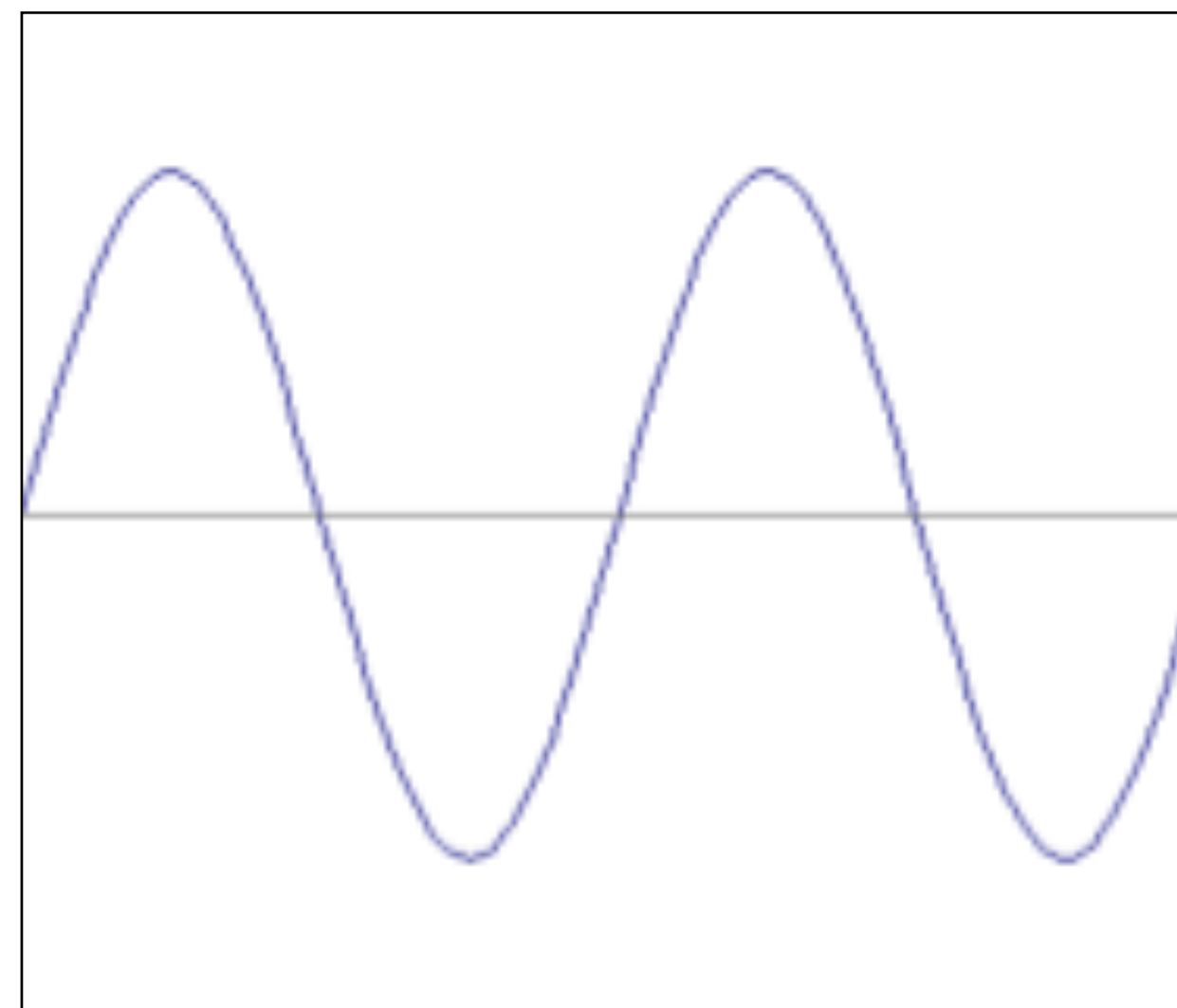
?

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

How would you generate this function?



=



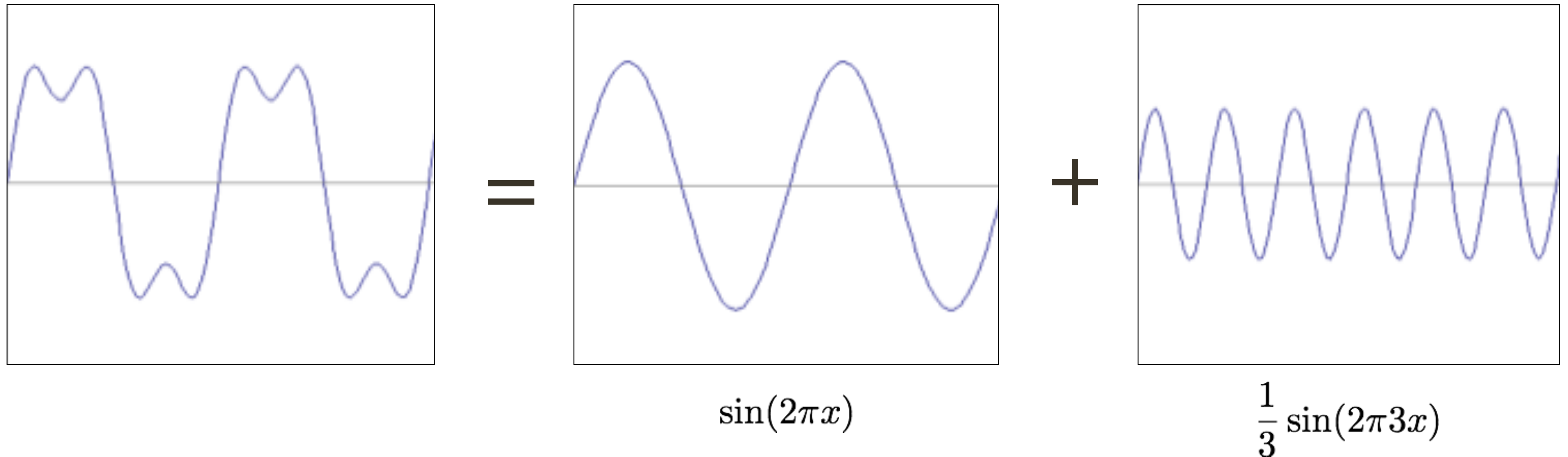
+

?

$\sin(2\pi x)$

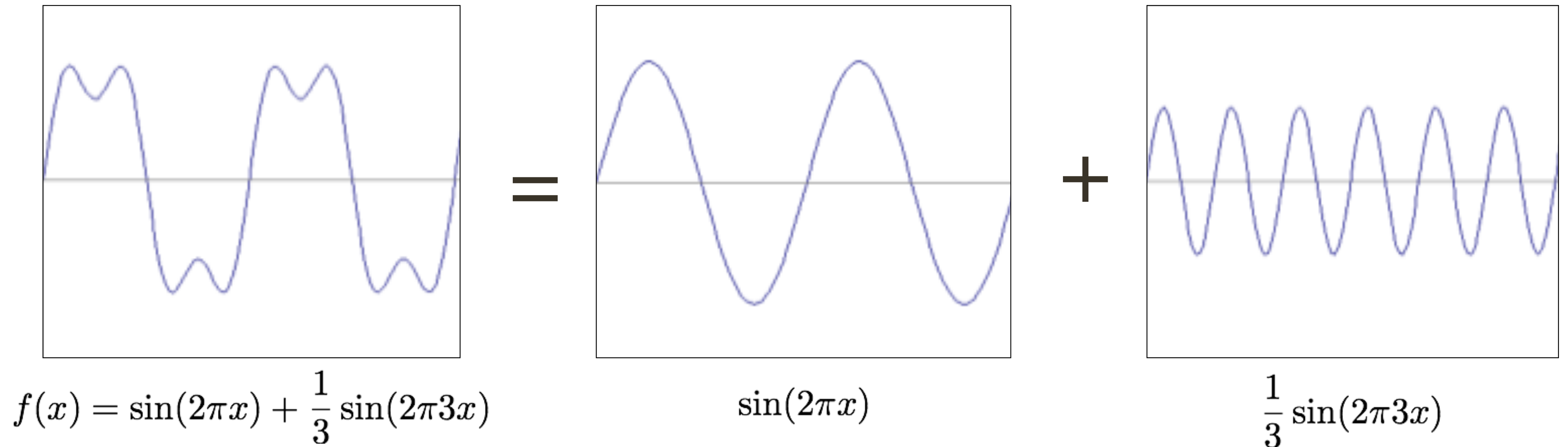
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?



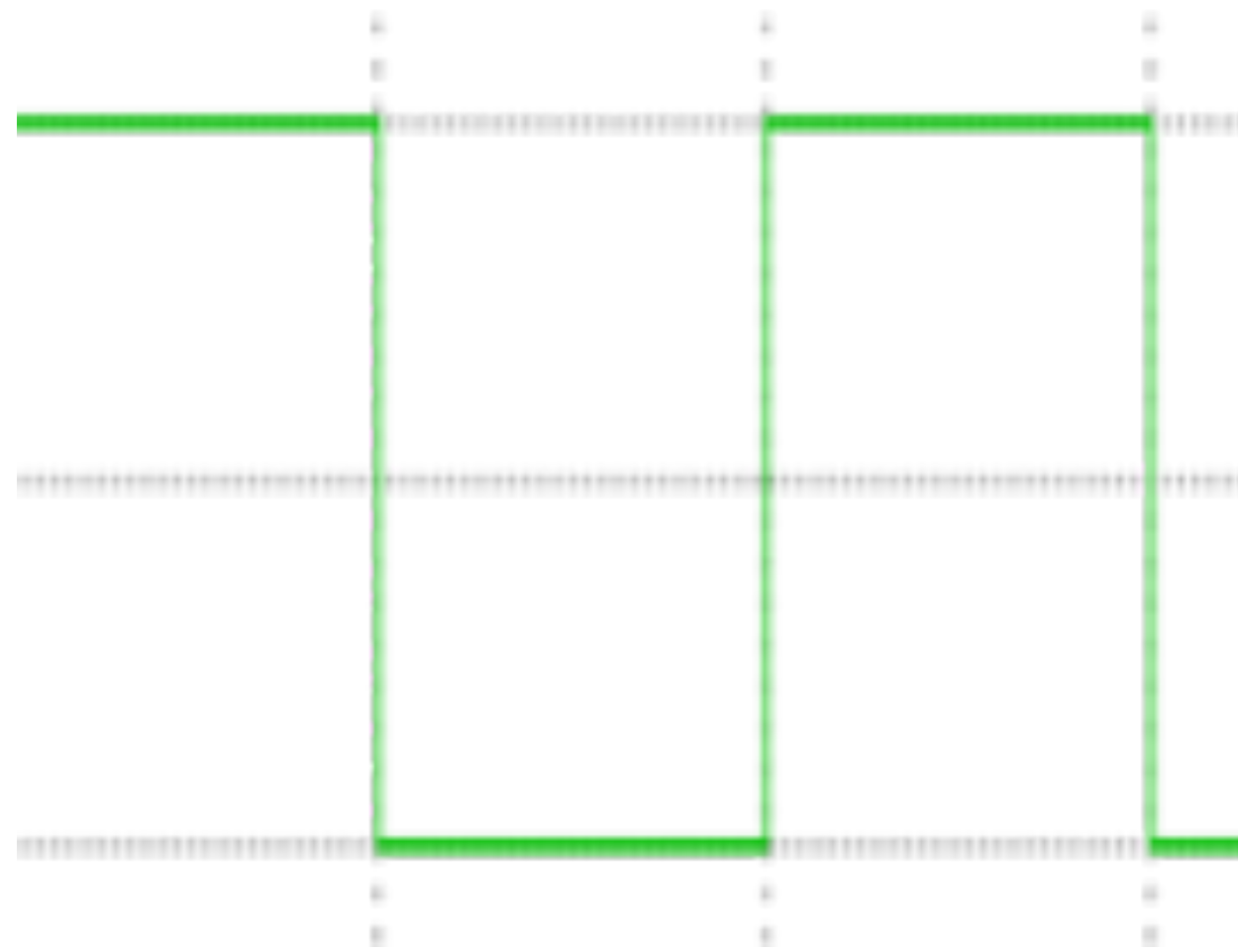
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?



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

How would you generate this function?

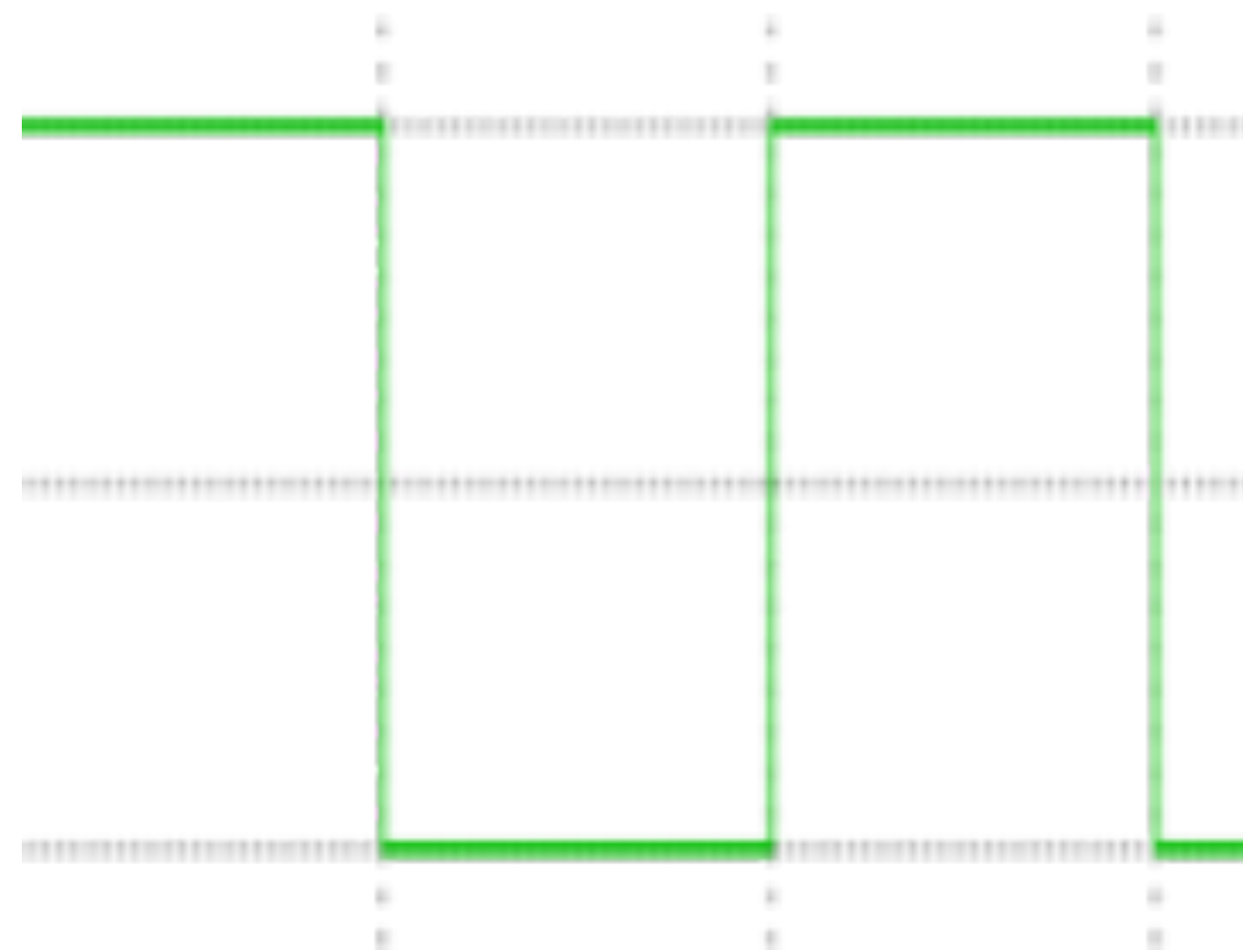


square wave

$\approx$? + ?

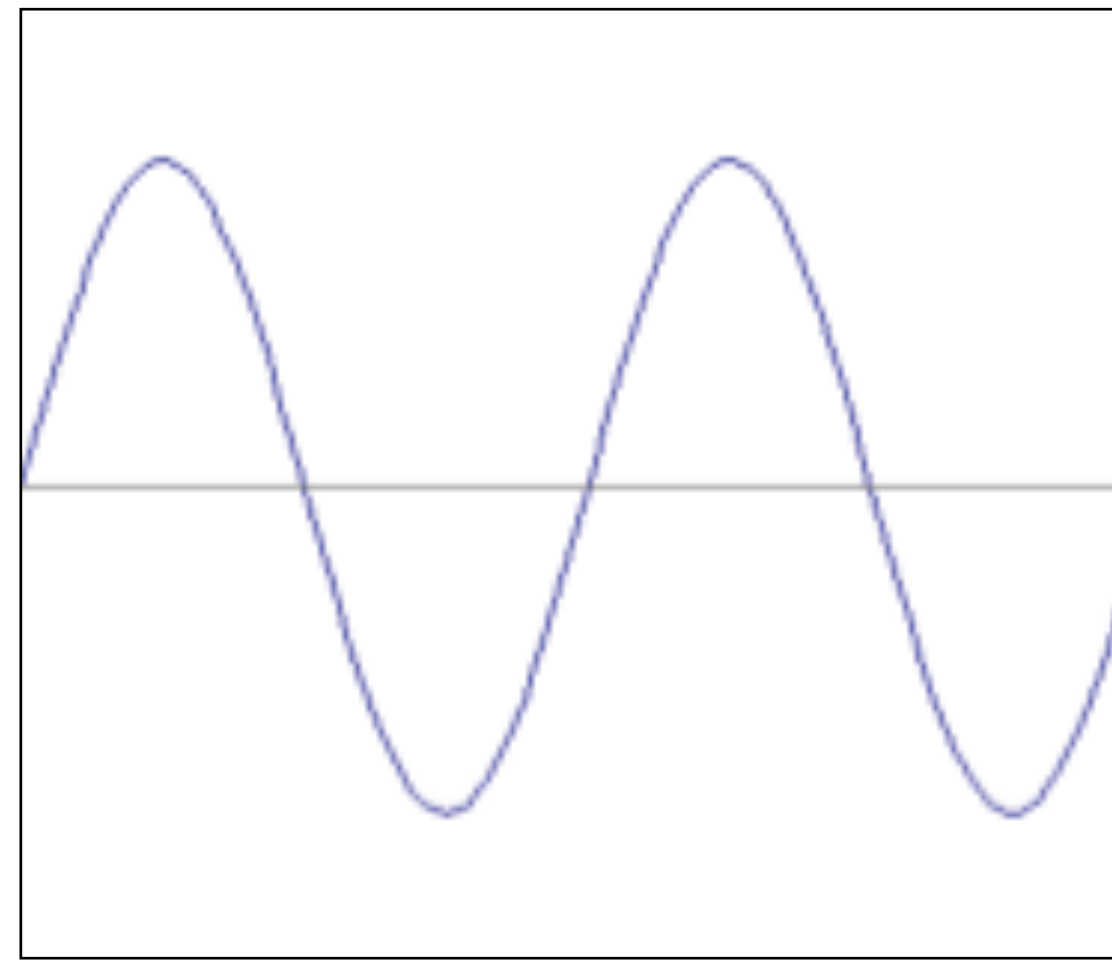
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

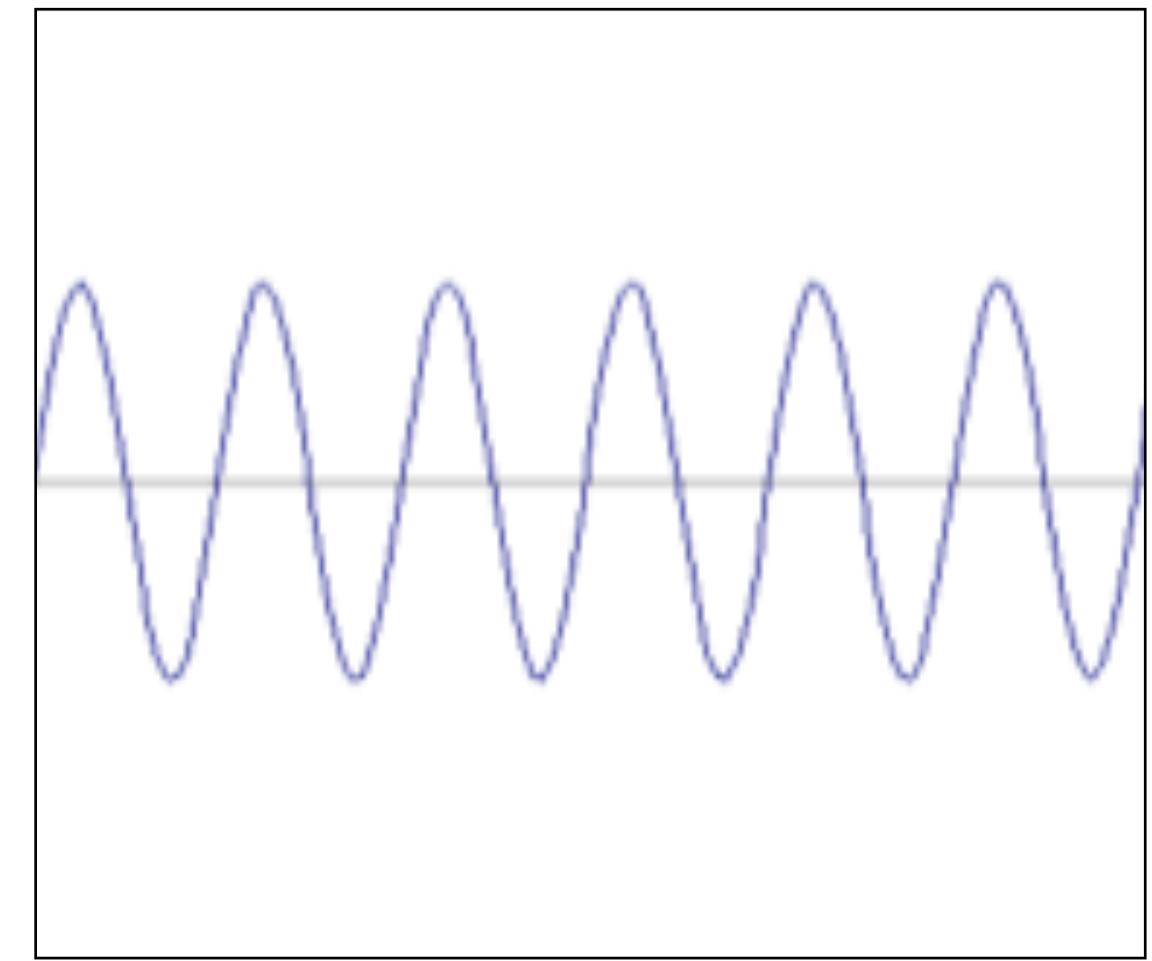


square wave

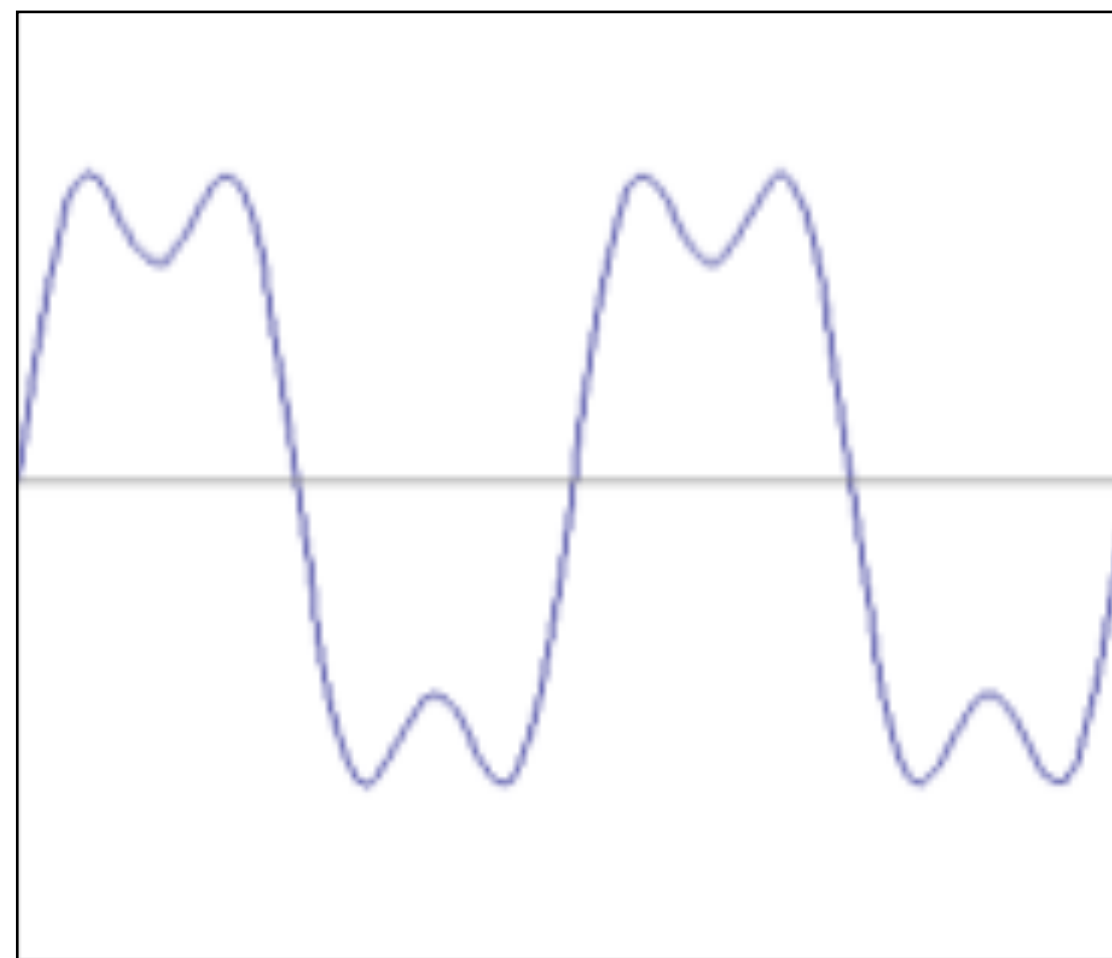
$\approx$



+

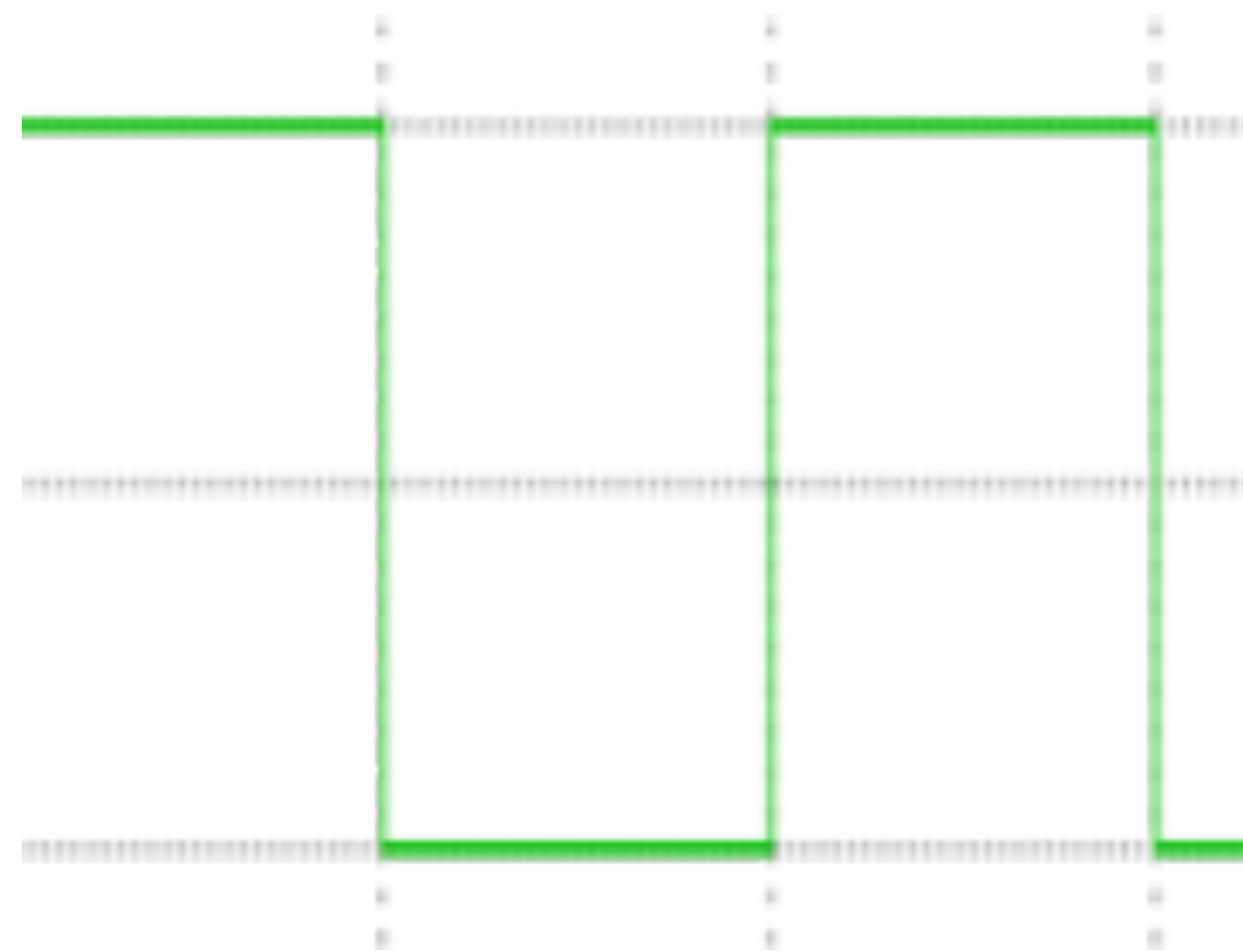


$=$



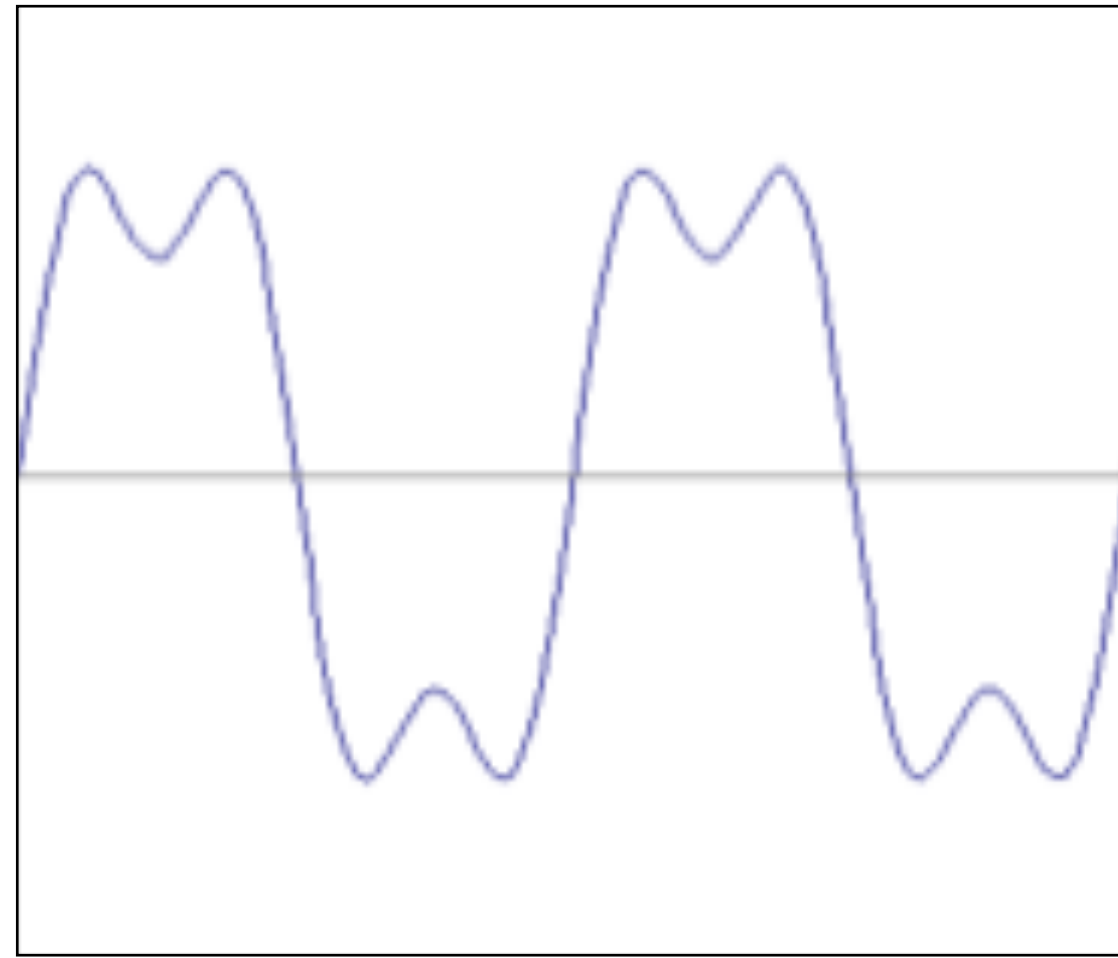
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

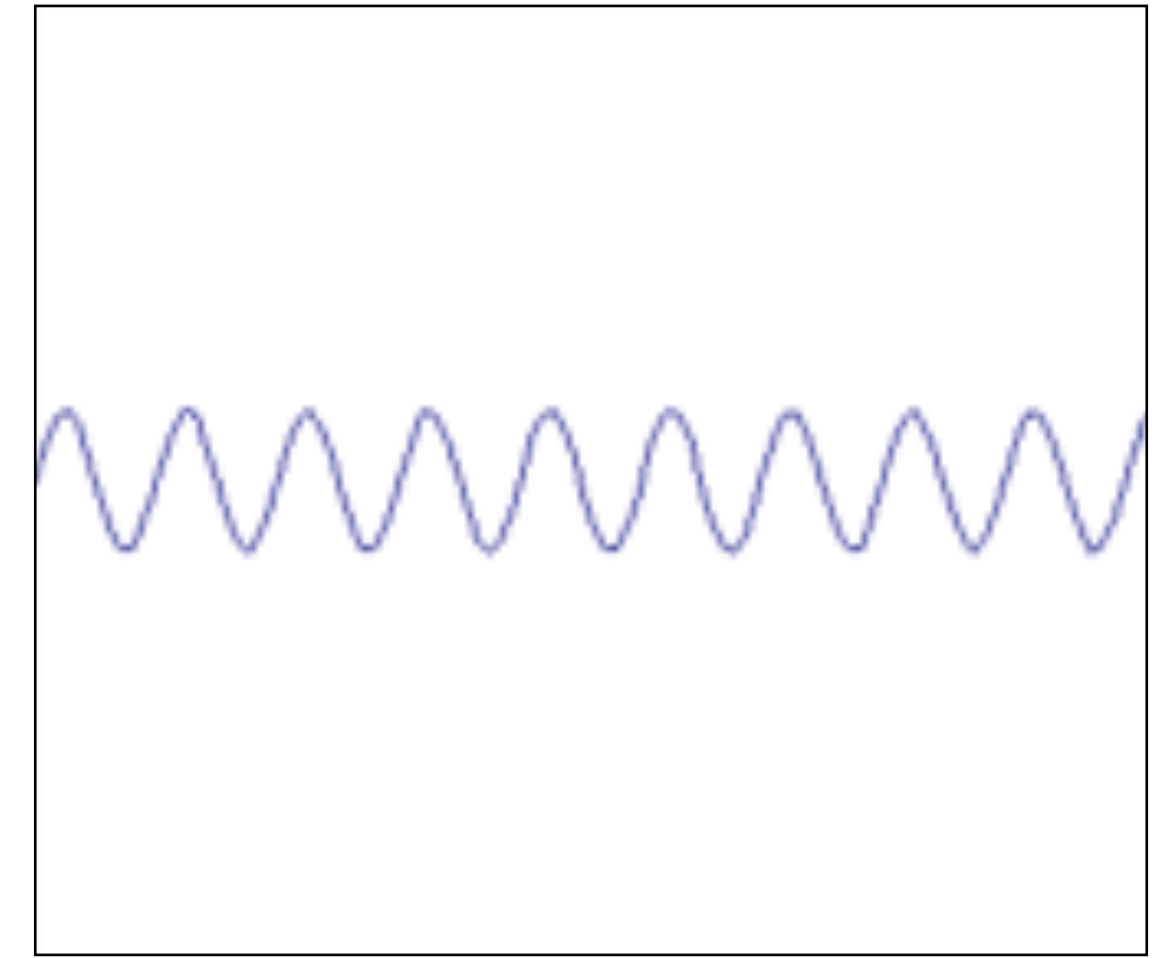


square wave

$\approx$



+

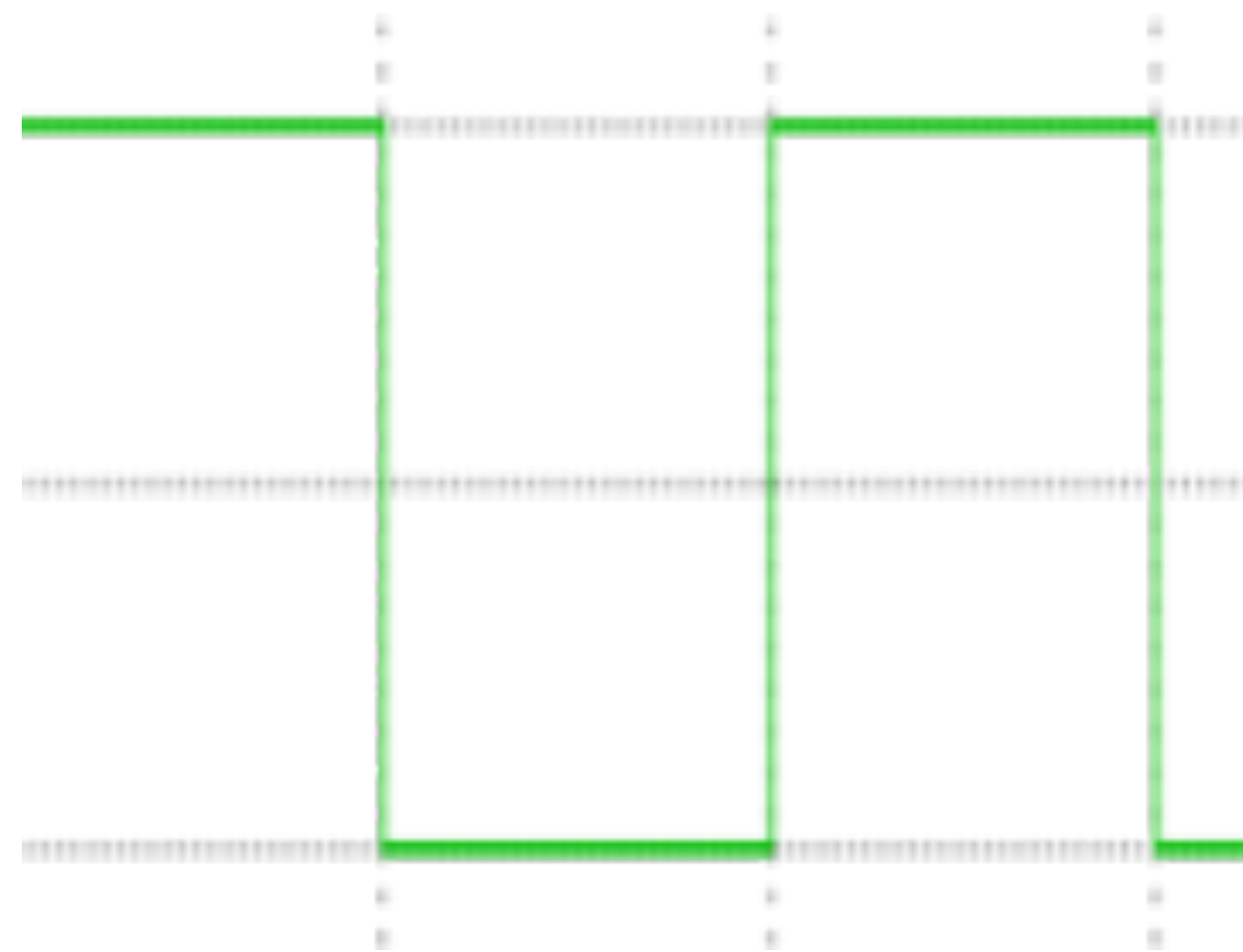


$=$



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

How would you generate this function?

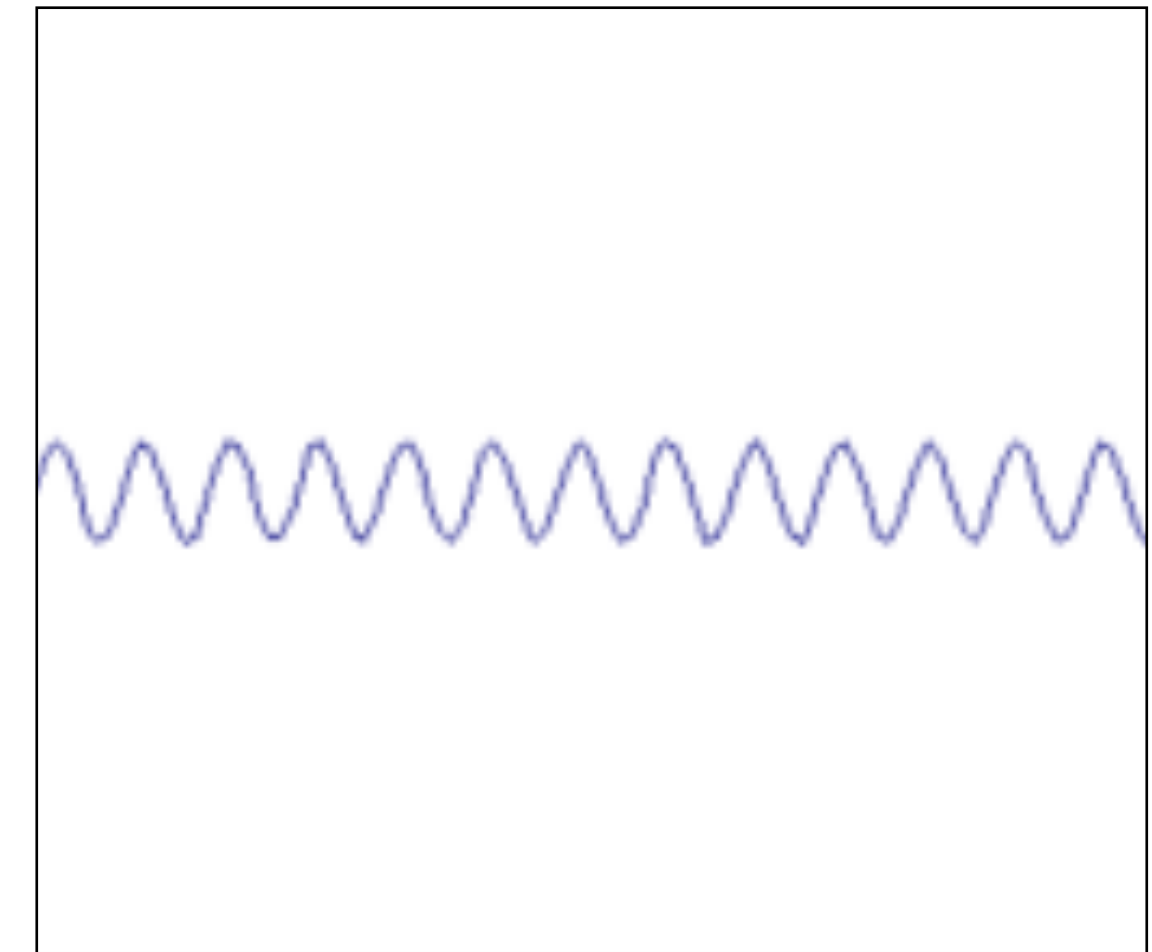


square wave

$\approx$



+

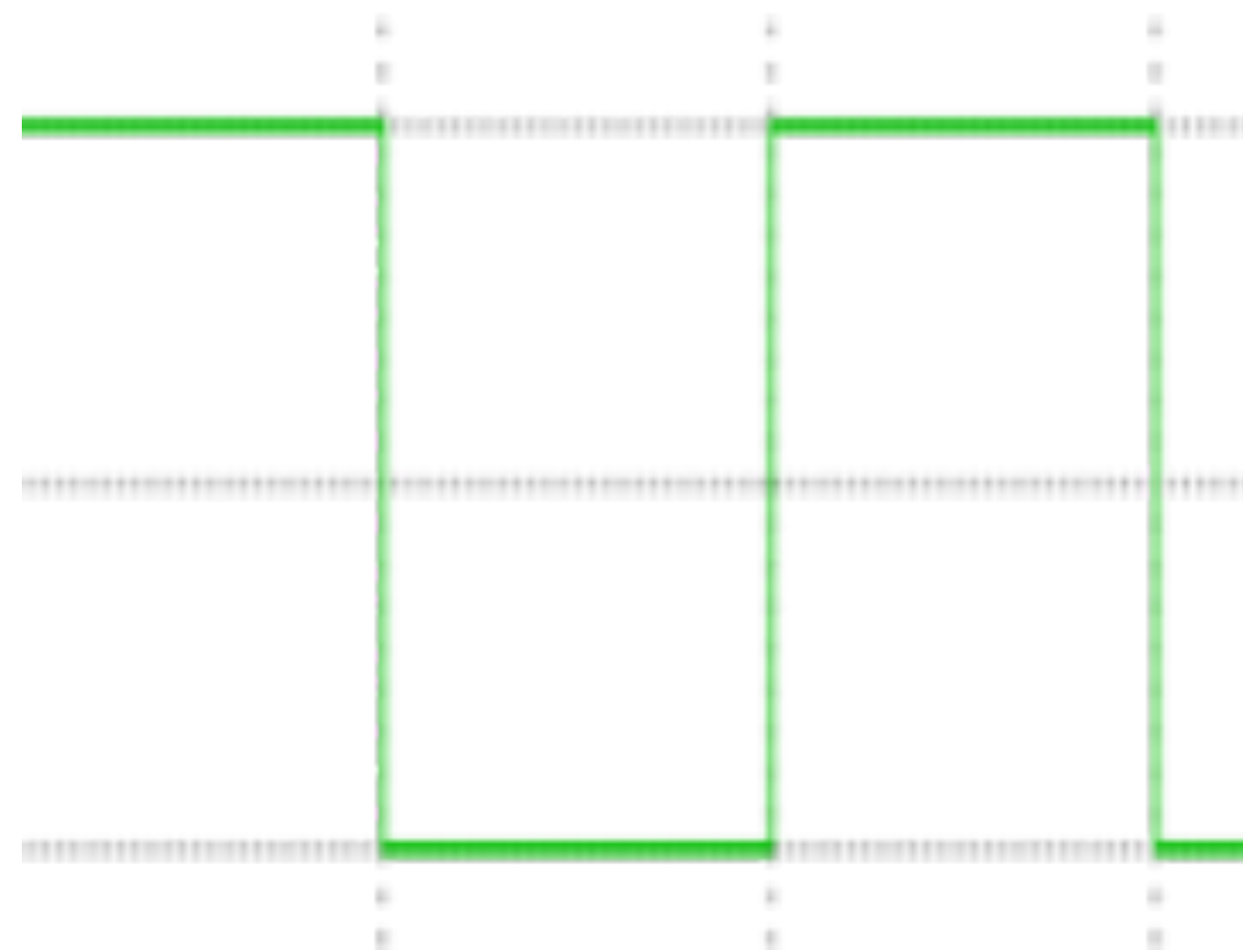


$=$



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

How would you generate this function?

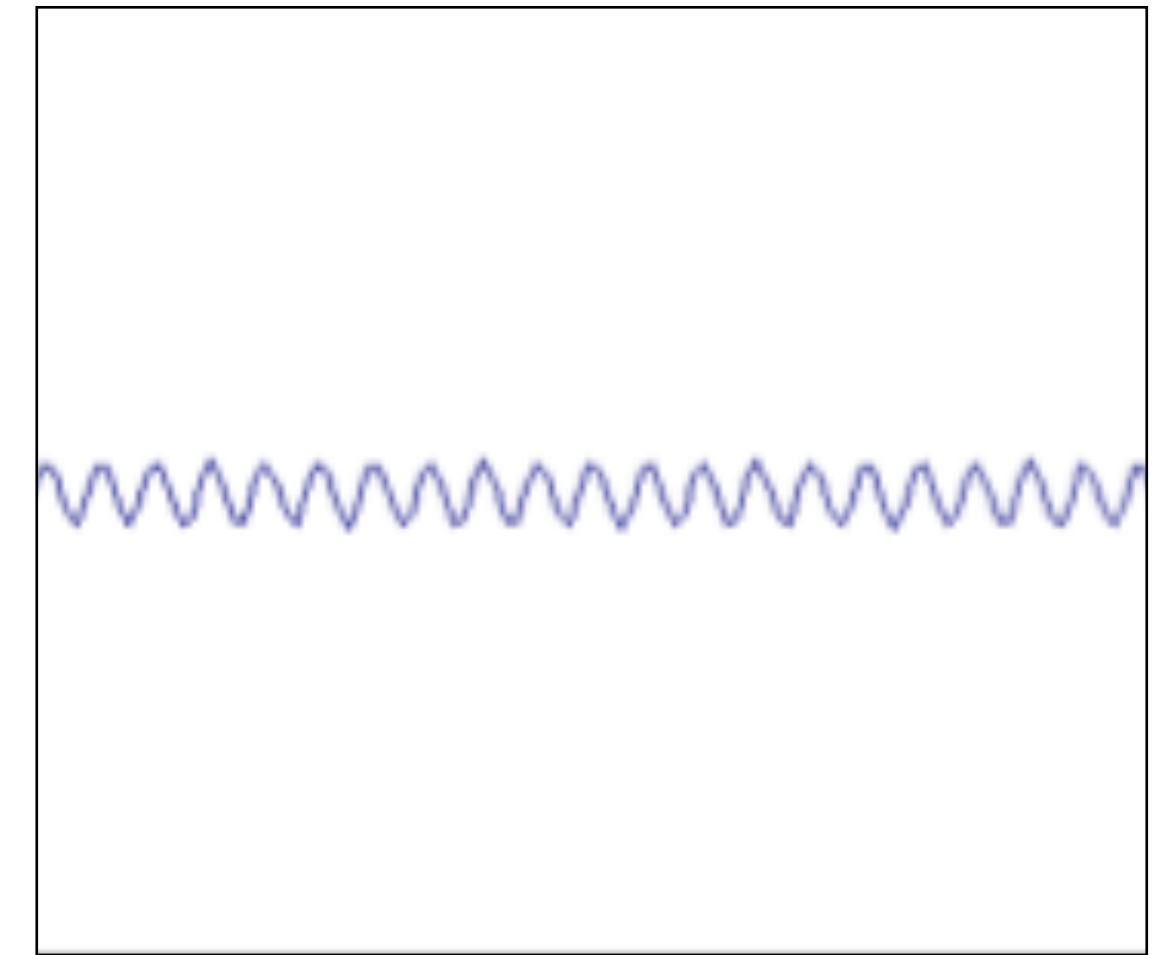


square wave

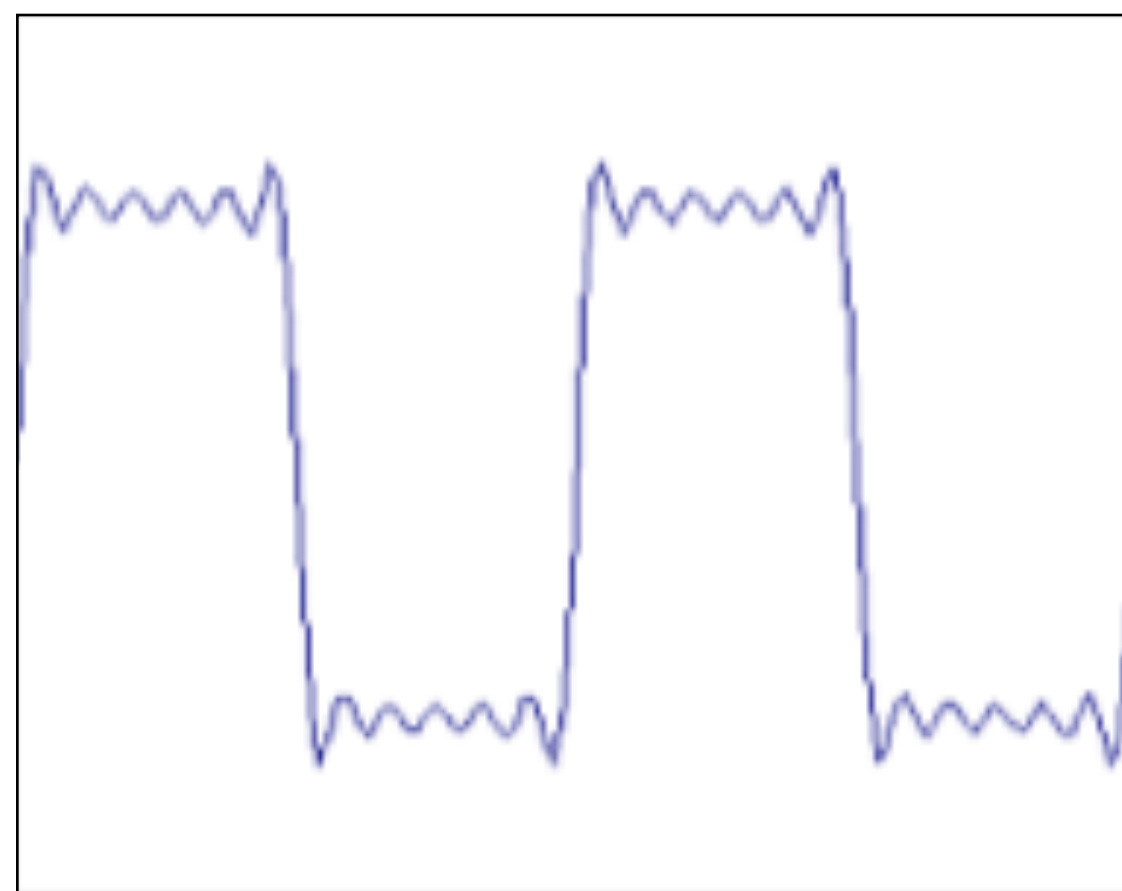
$\approx$



+



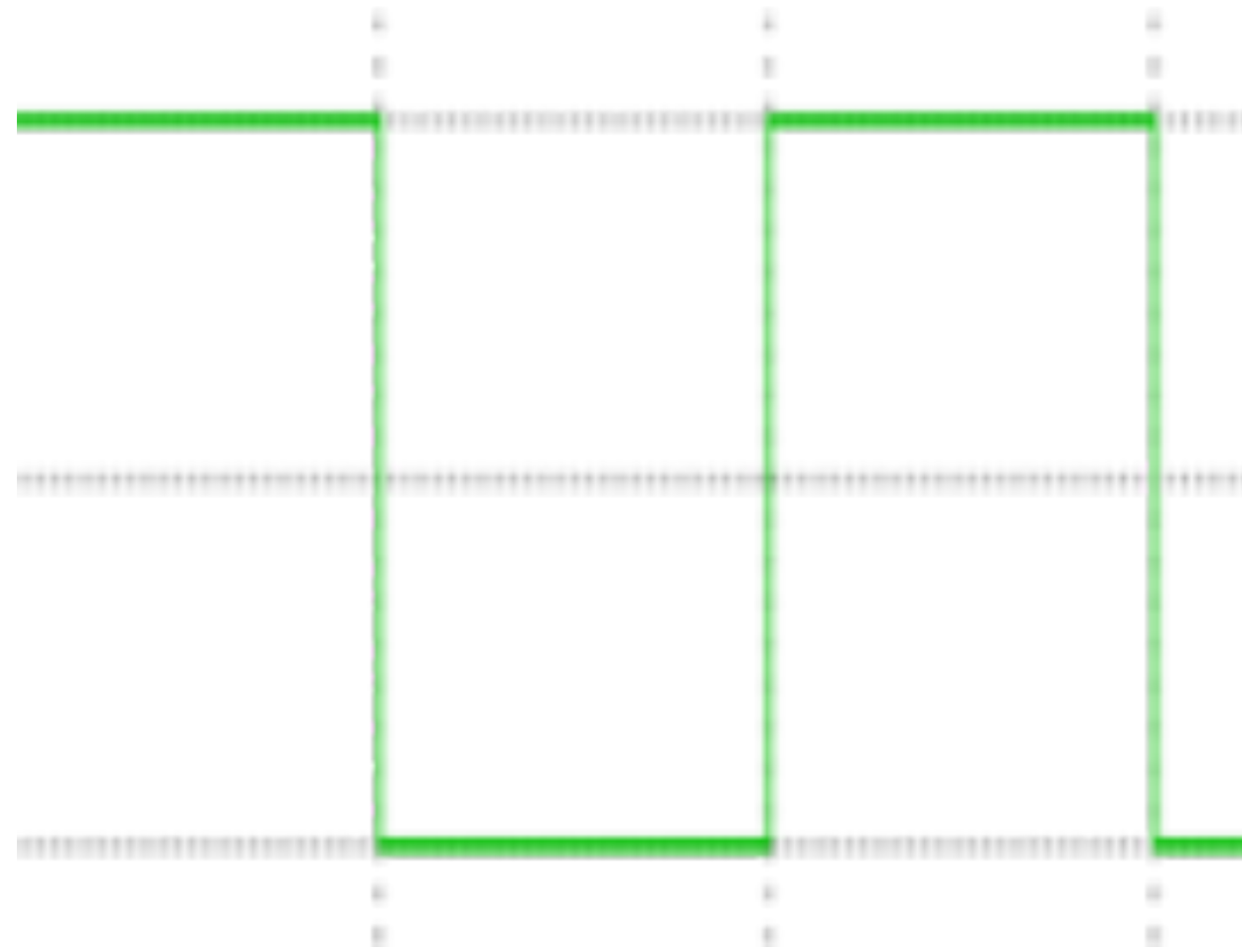
$=$



How would you
express this
mathematically?

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

How would you generate this function?



square wave

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

infinite sum of sine waves

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!

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

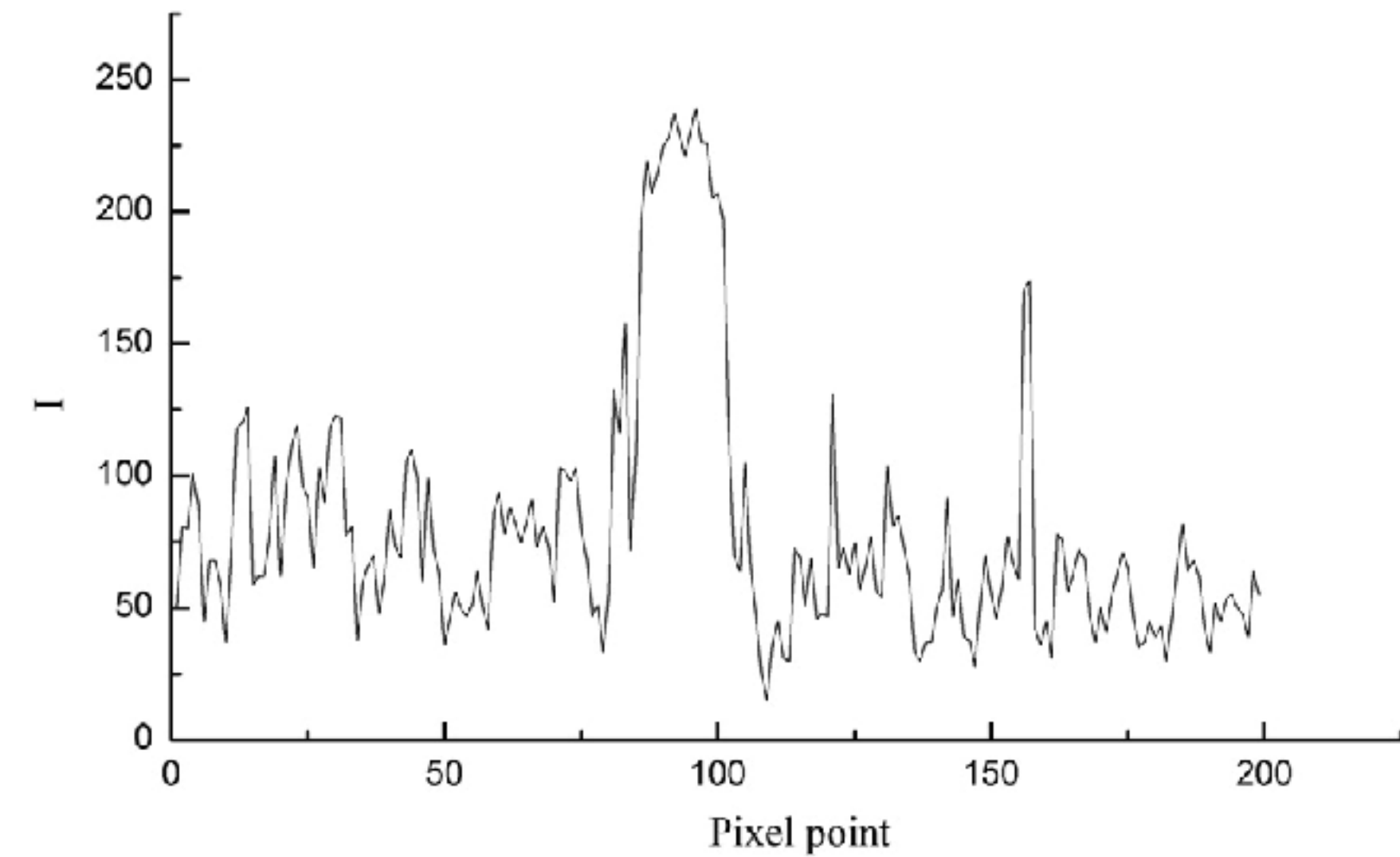
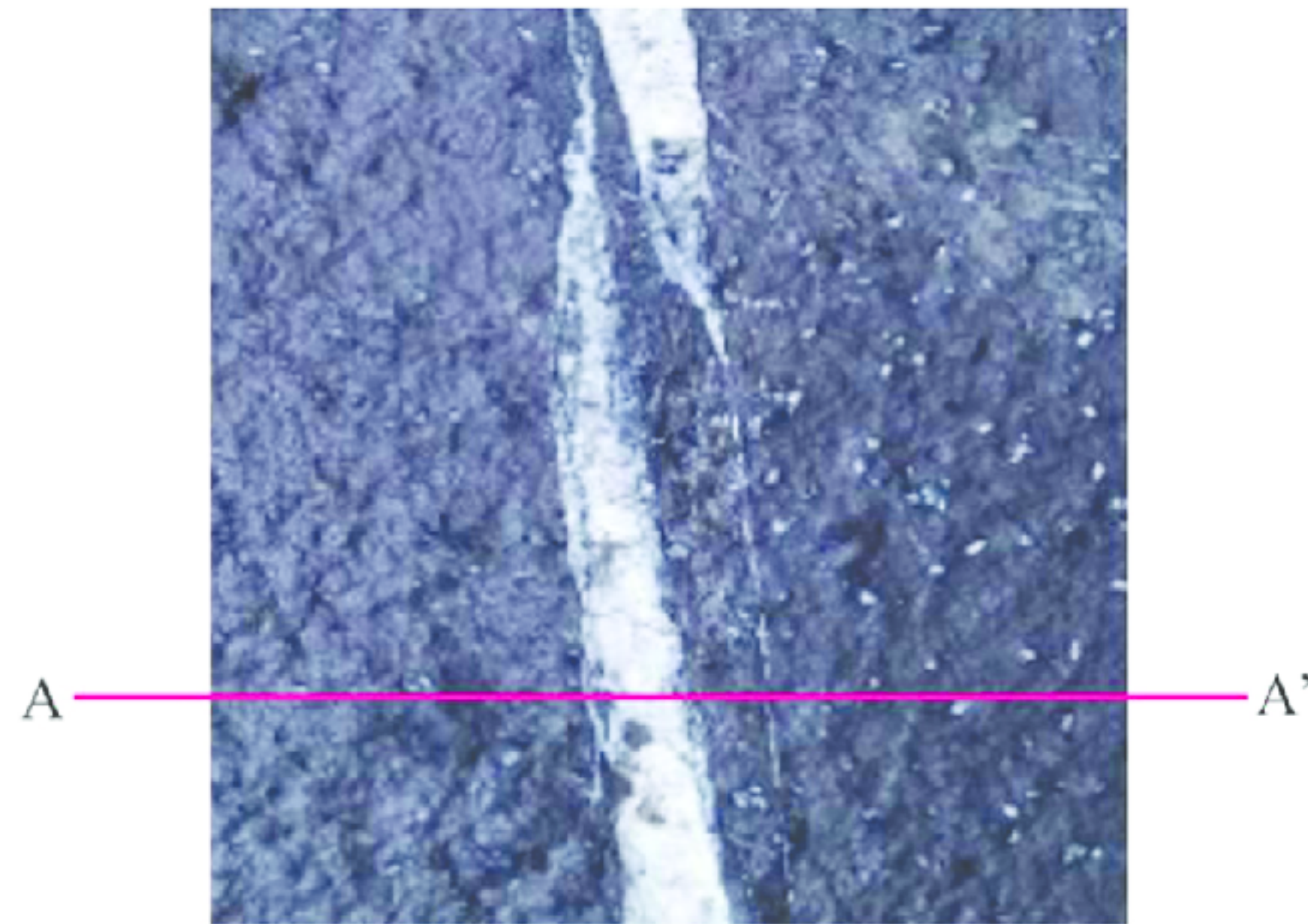


Image from: Numerical Simulation and Fractal Analysis of Mesoscopic Scale Failure in Shale Using Digital Images

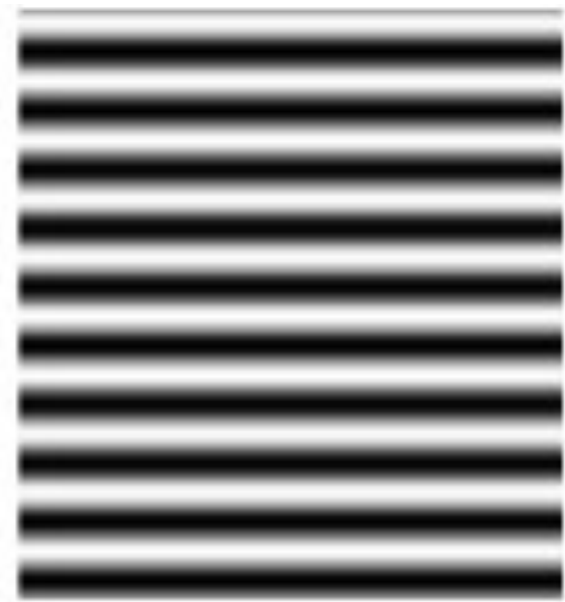
Fourier Transform (you will **NOT** be tested on this)

What are “frequencies” in an image?

Spatial frequency



$$f = 4$$



$$f = 5$$



$$f = 6$$



$$f = 7$$



$$f = 8$$



$$f = 9$$

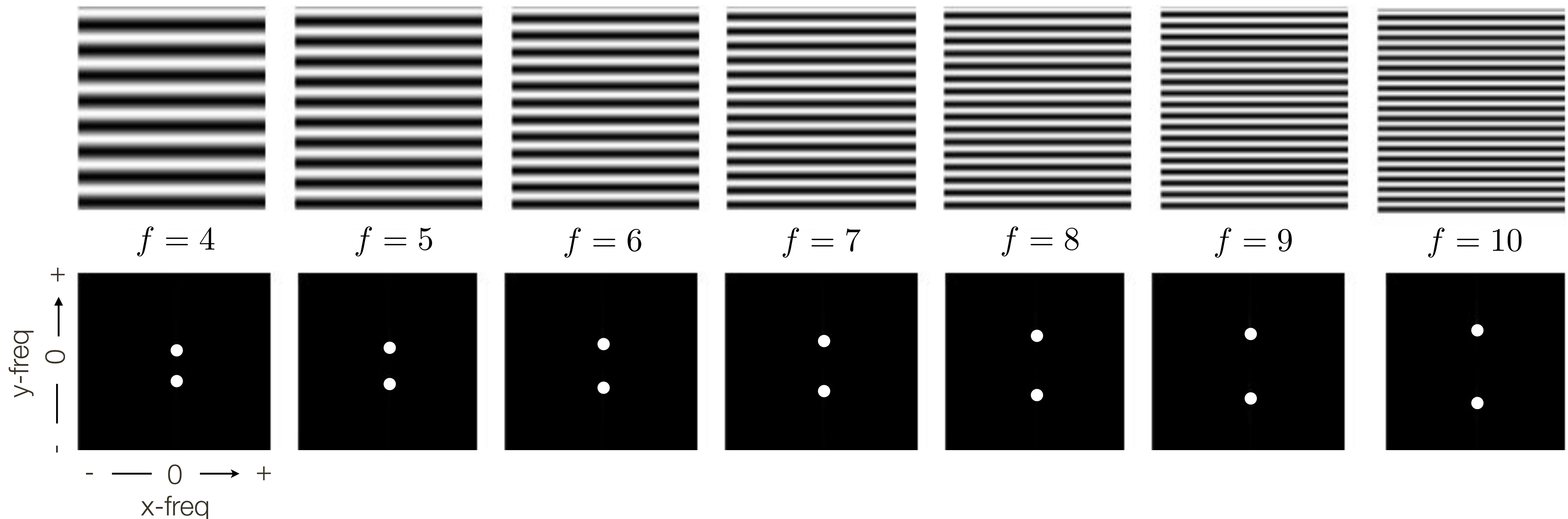


$$f = 10$$

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

What are “frequencies” in an image?

Spatial frequency

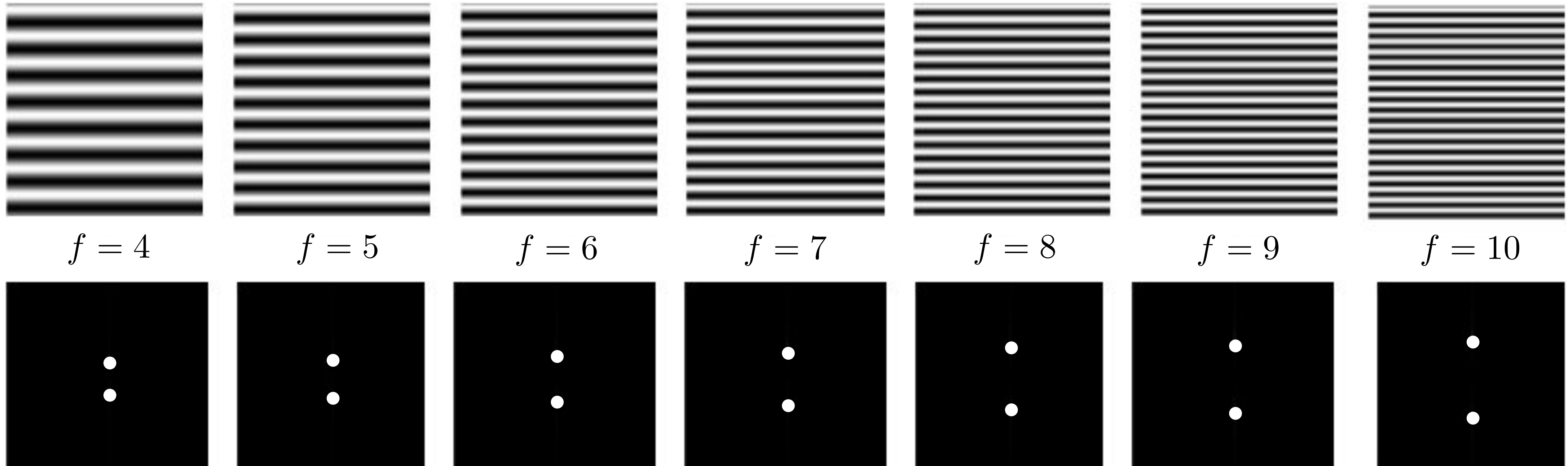


Amplitude (magnitude) of Fourier transform (phase does not show desirable correlations with image structure)

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

What are “frequencies” in an image?

Spatial frequency



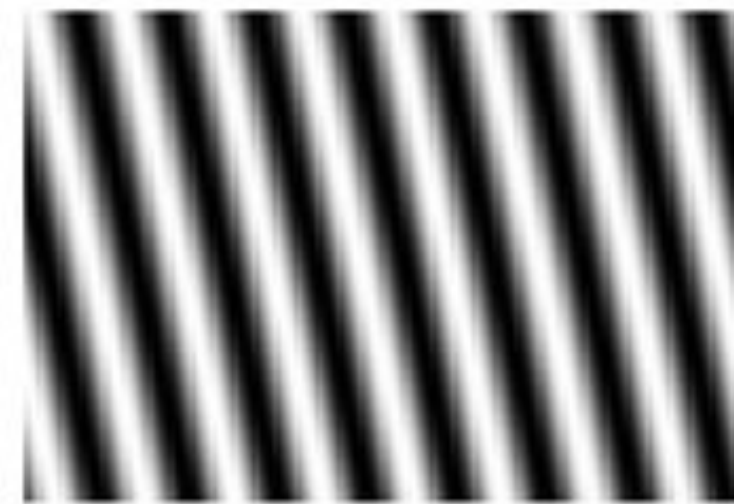
Amplitude (magnitude) of Fourier transform (phase does not show desirable correlations with image structure)

Observation: low frequencies close to the center

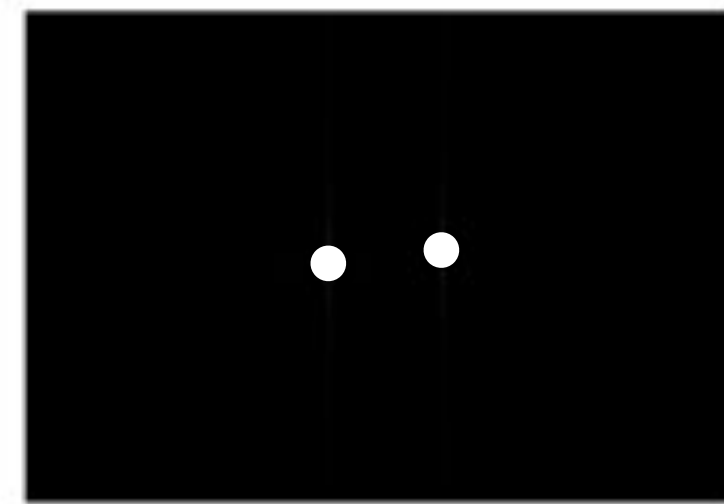
Fourier Transform (you will **NOT** be tested on this)

What are “frequencies” in an image?

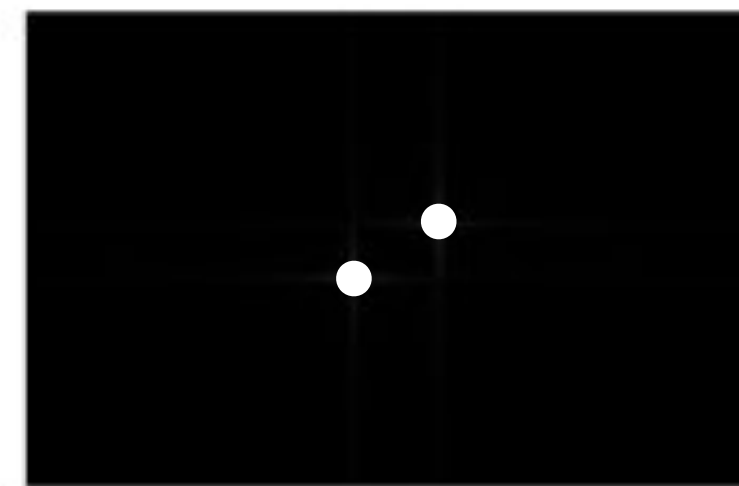
Spatial frequency



$\Theta=30^\circ$



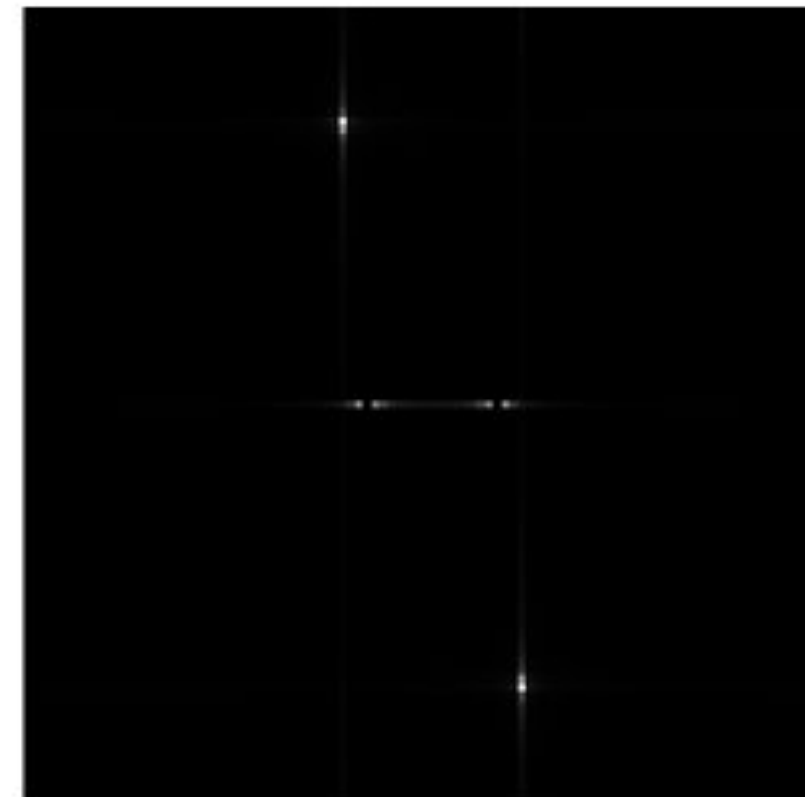
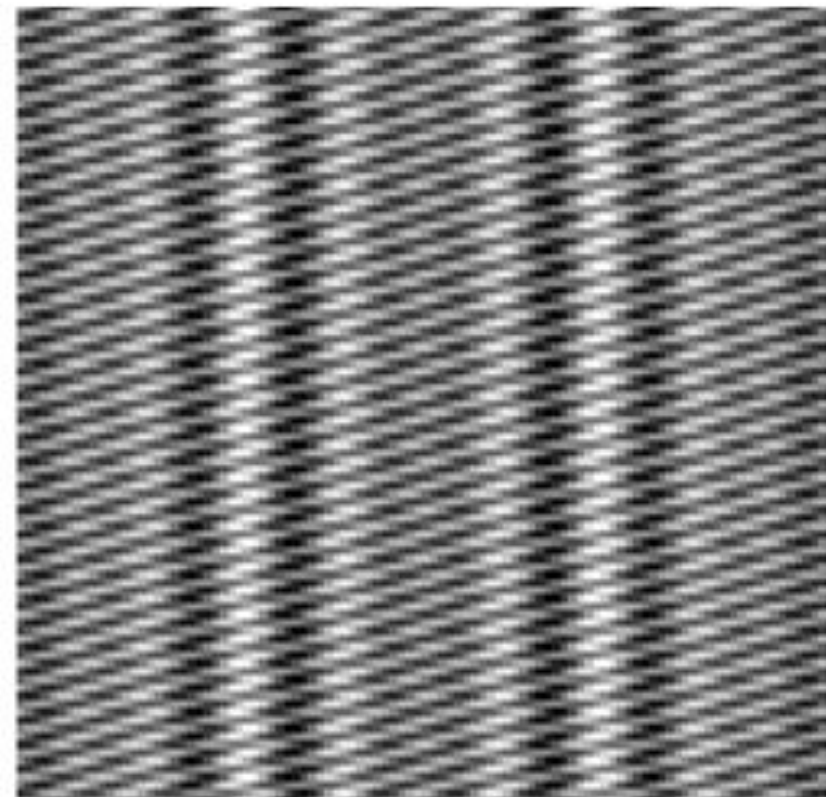
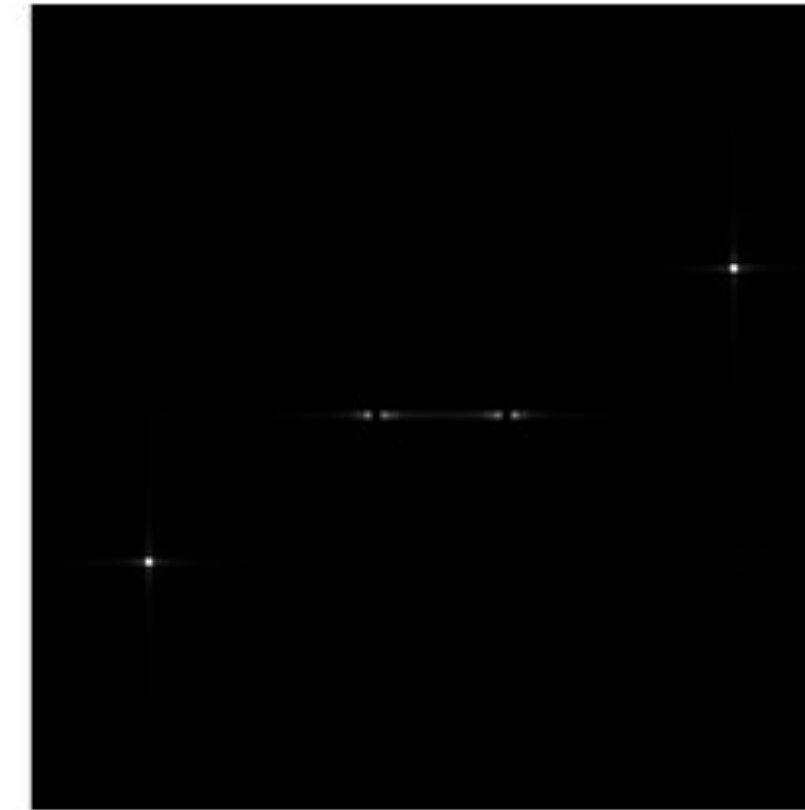
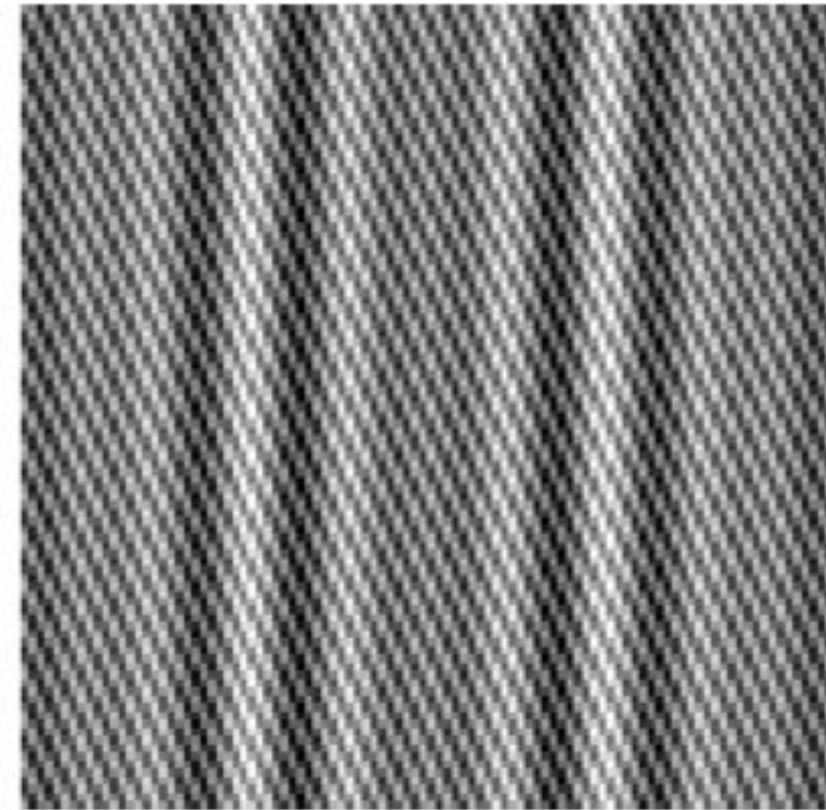
$\Theta=150^\circ$



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

What are “frequencies” in an image?

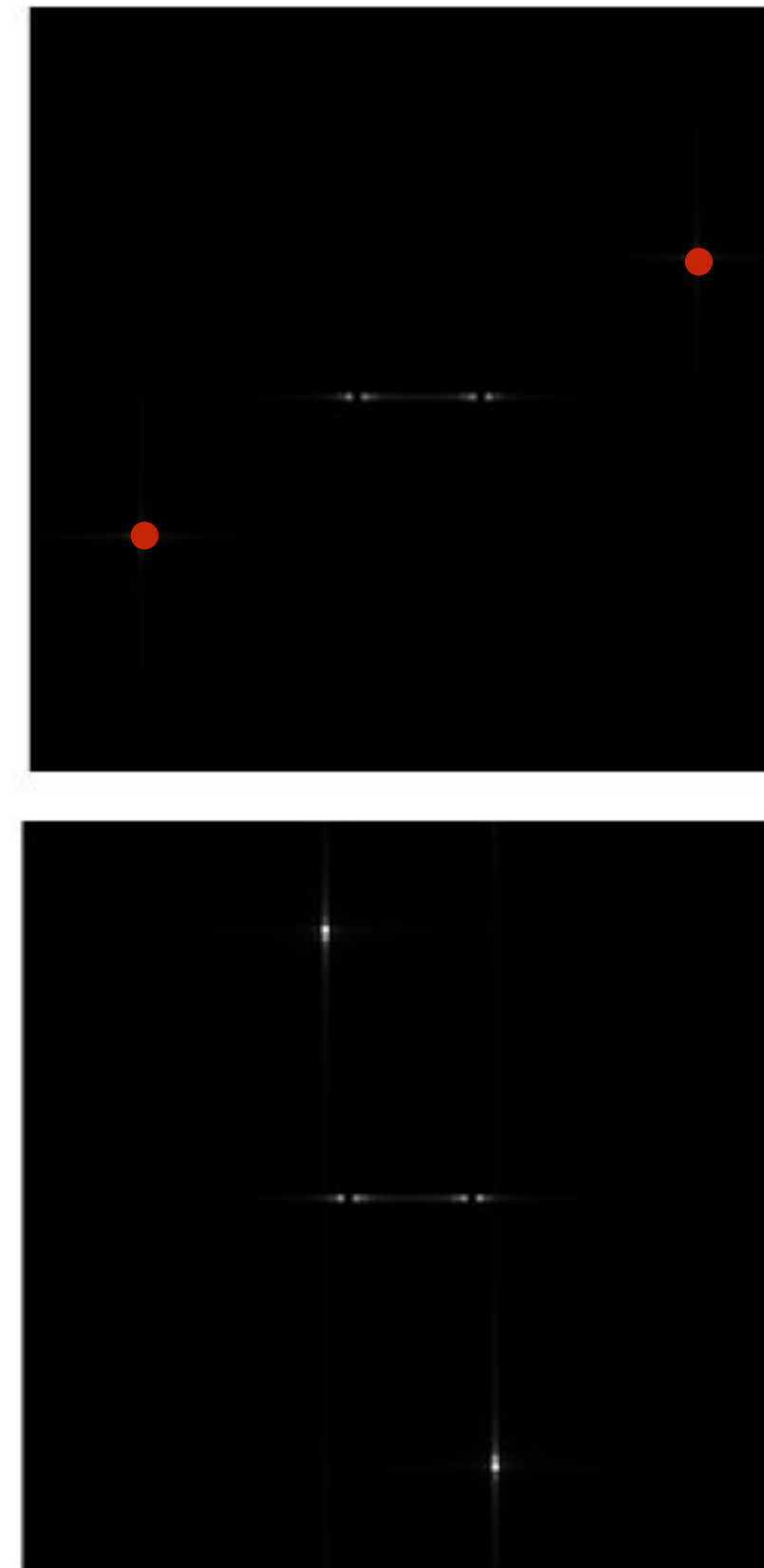
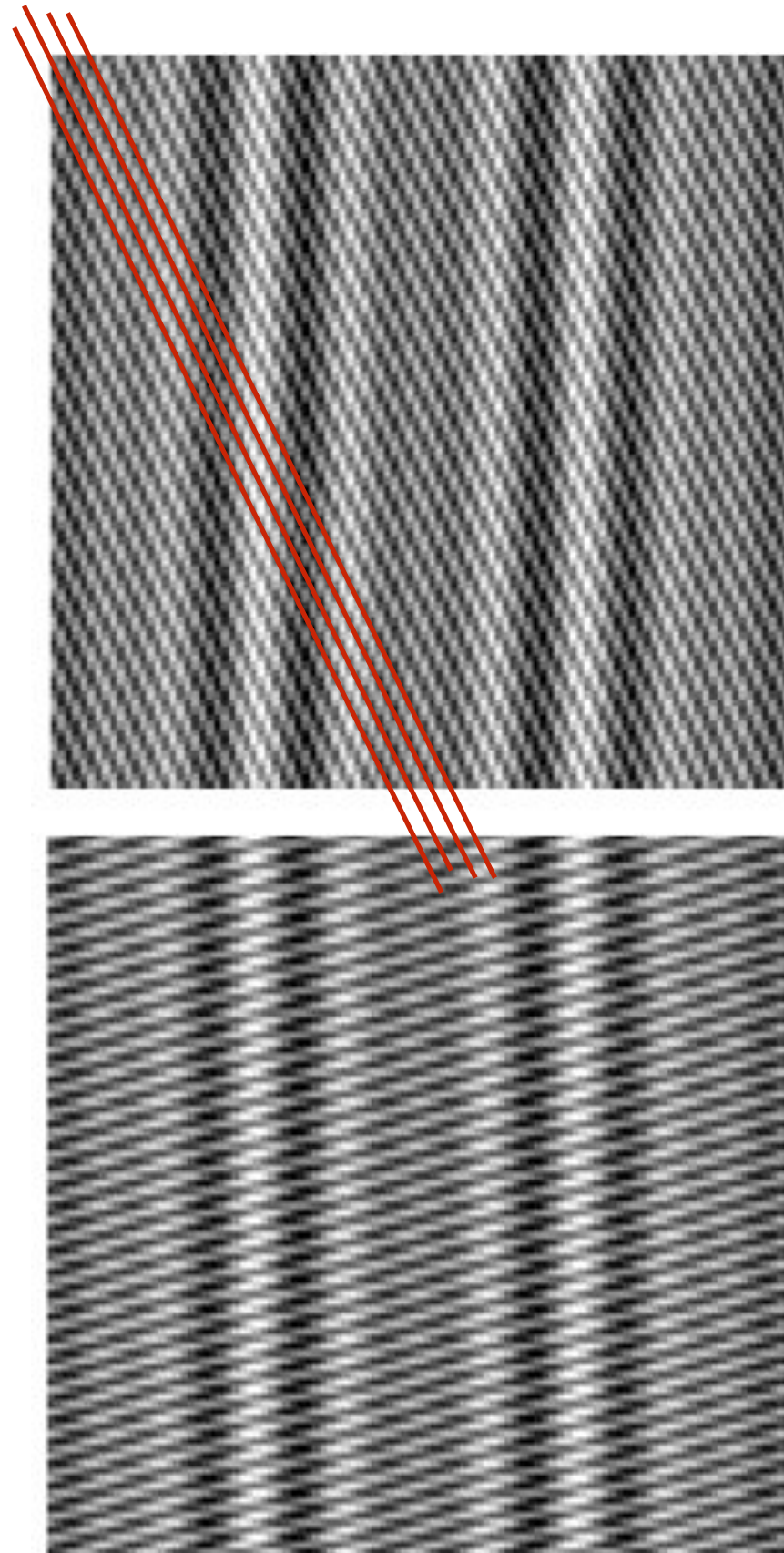
Spatial frequency



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

What are “frequencies” in an image?

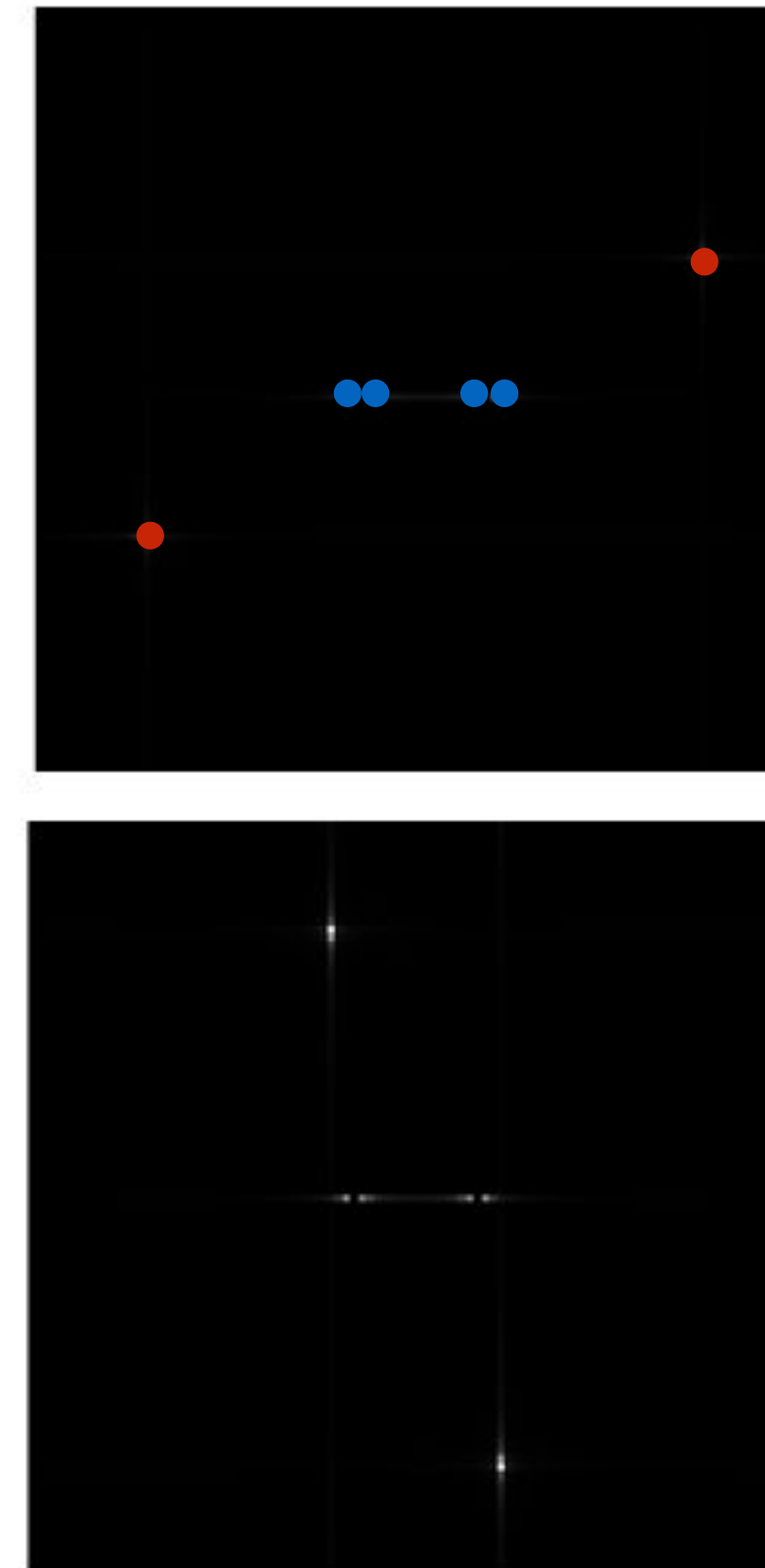
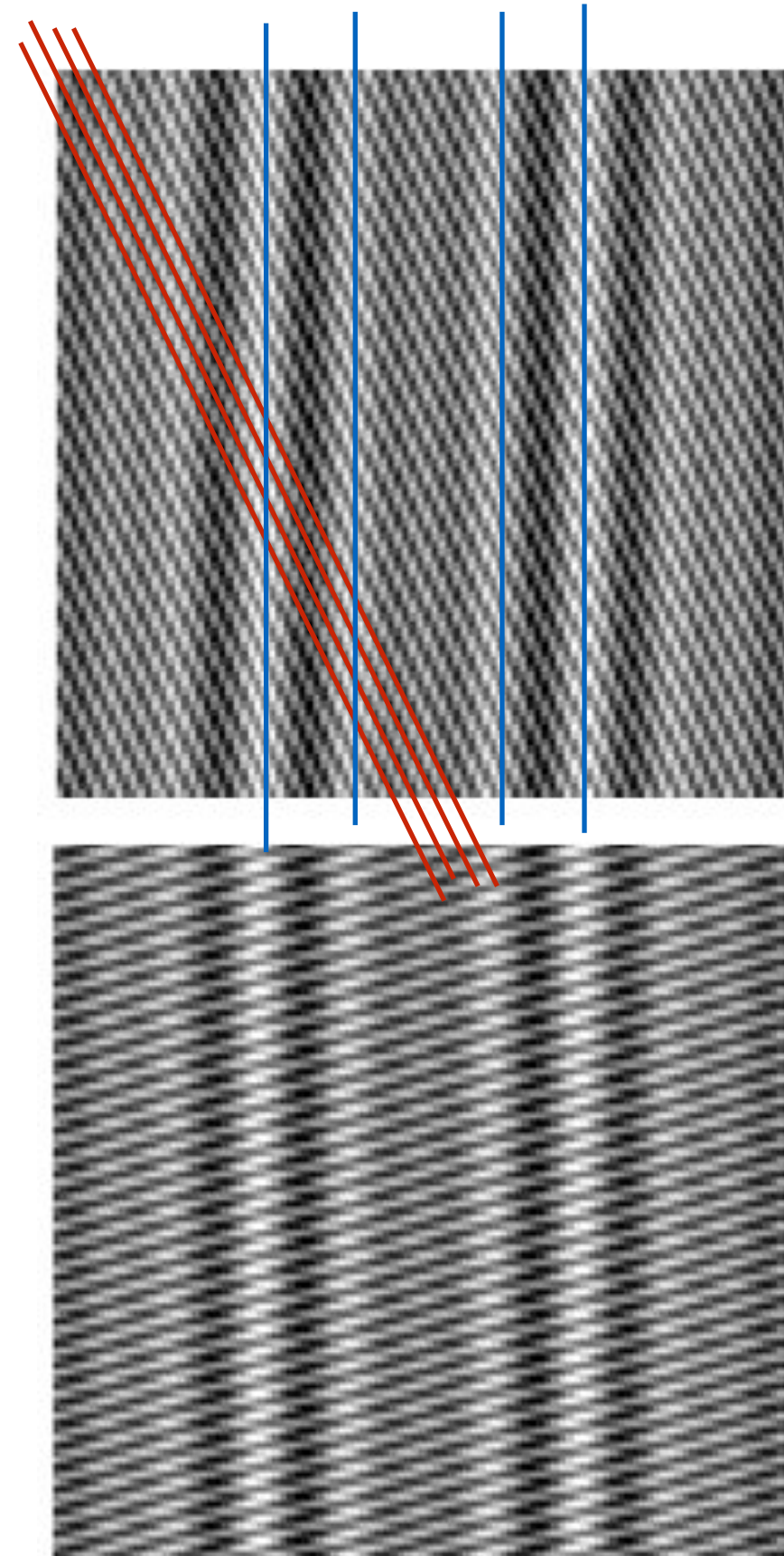
Spatial frequency



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

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>

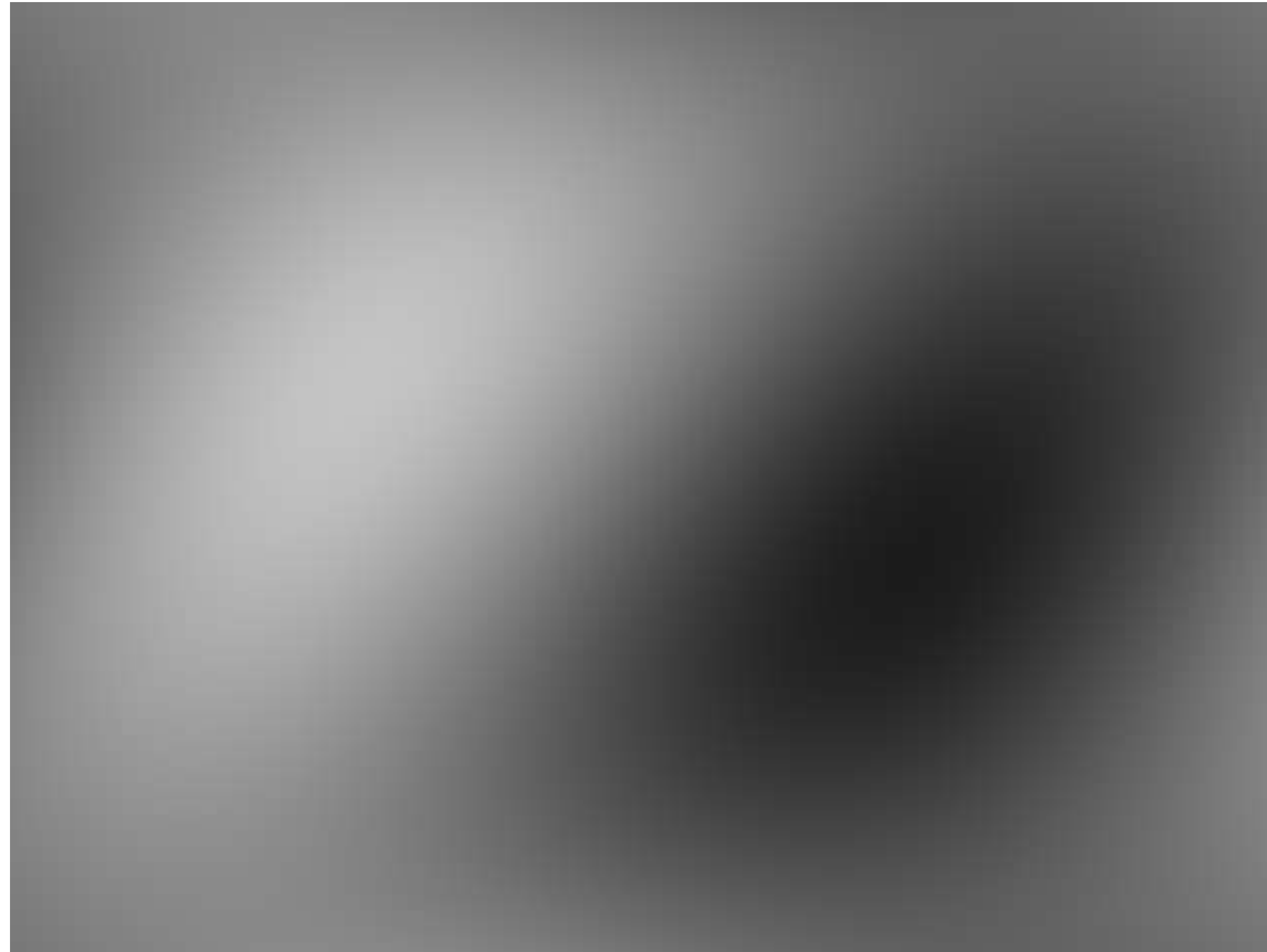
Fourier Transform (you will **NOT** be tested on this)



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

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

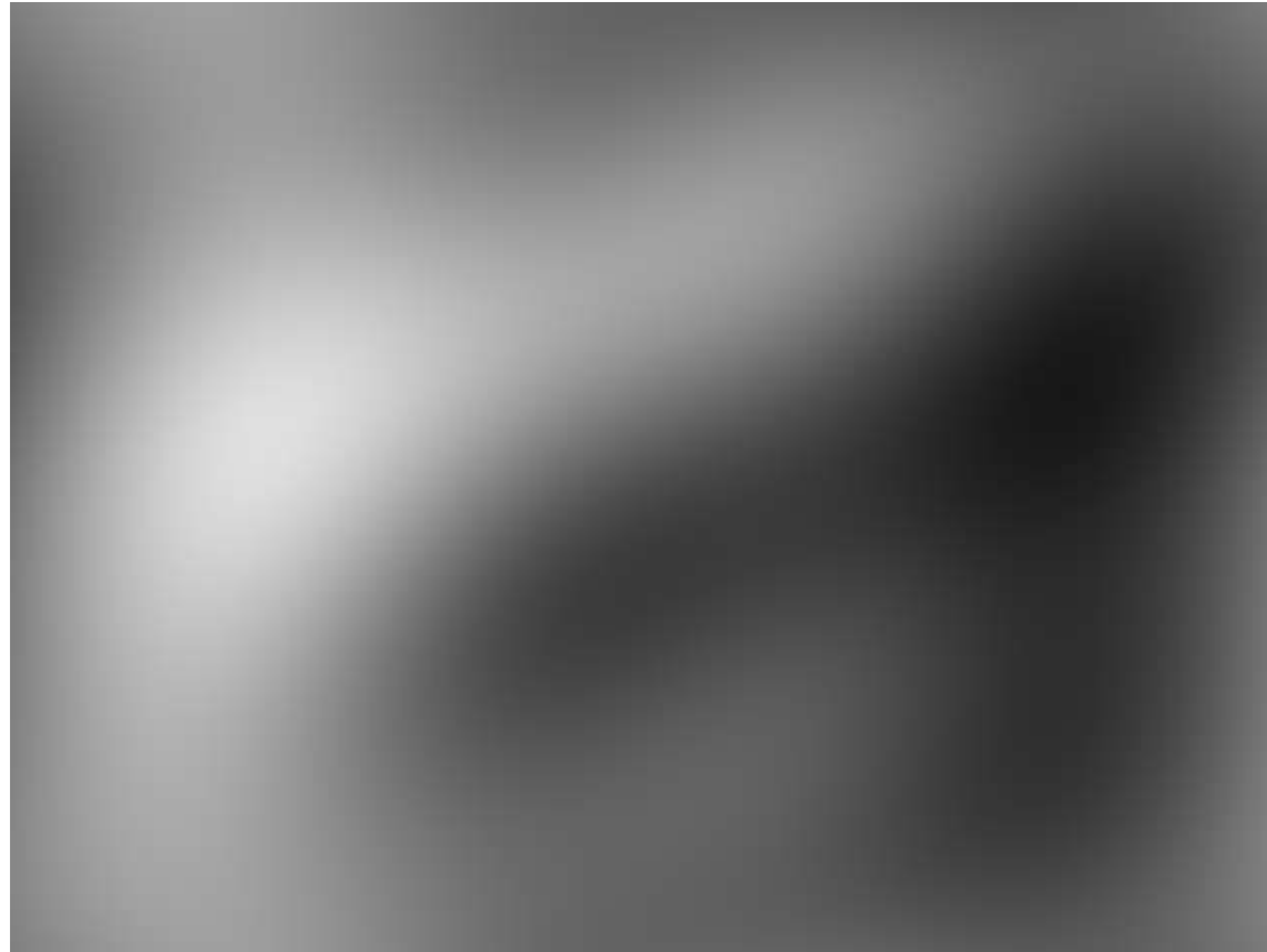
Fourier Transform (you will **NOT** be tested on this)



+ **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)



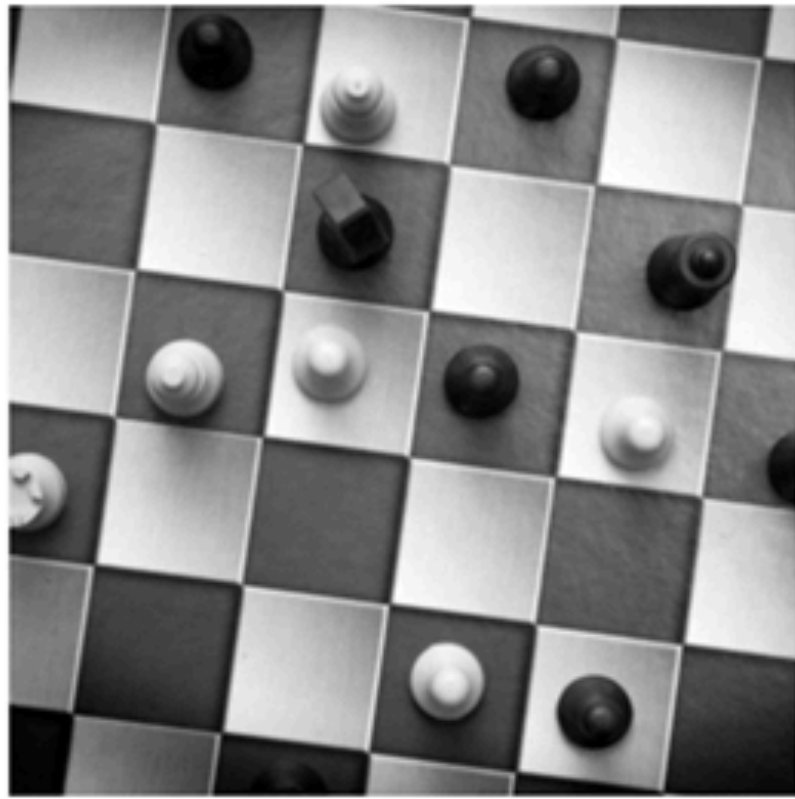
+ **50%** of frequencies

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

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



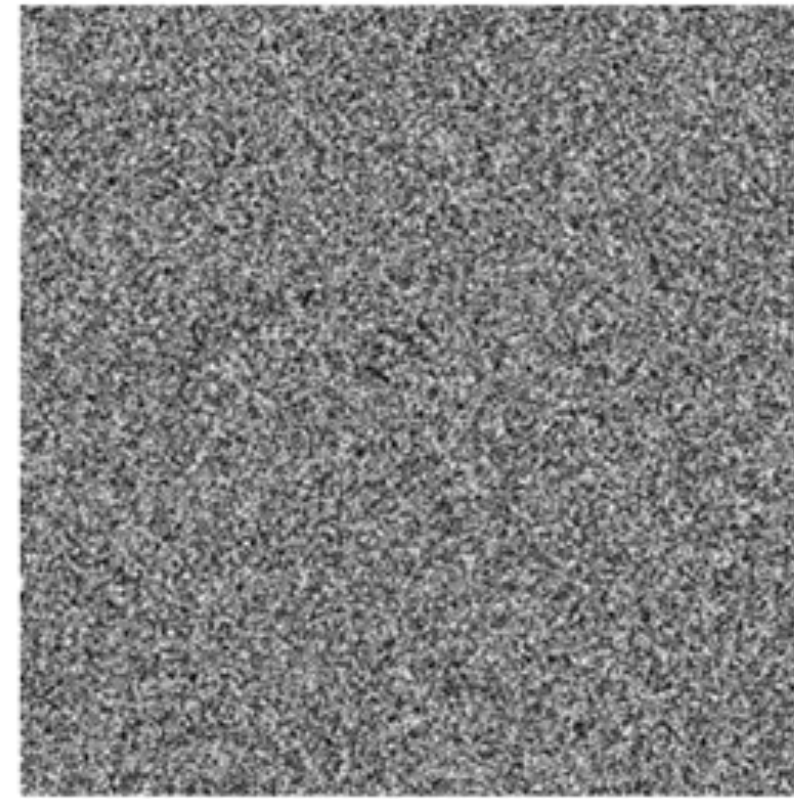
Fourier Transform (you will **NOT** be tested on this)



(I)



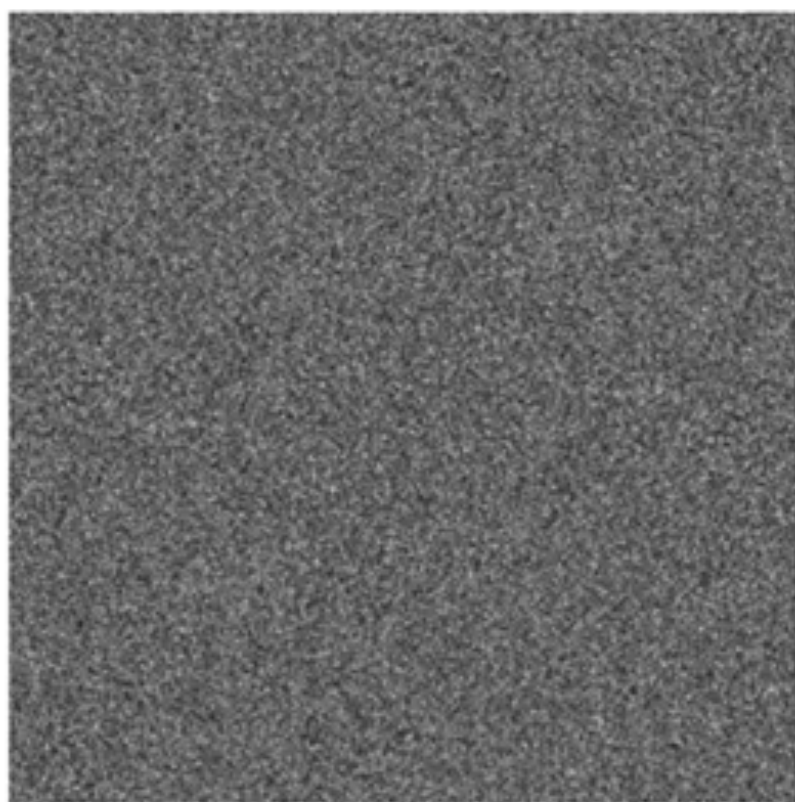
(II)



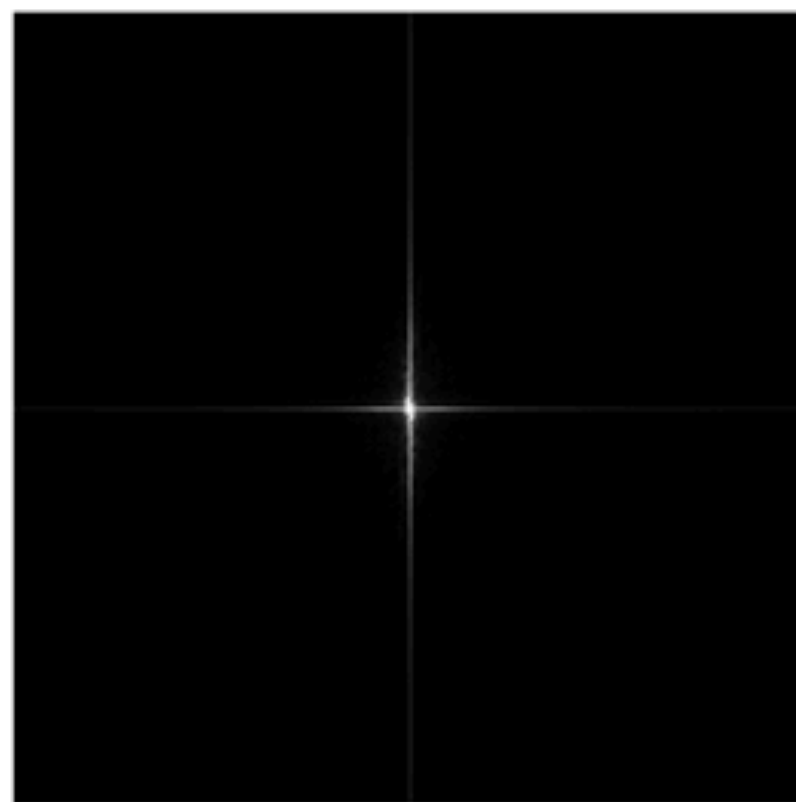
(III)



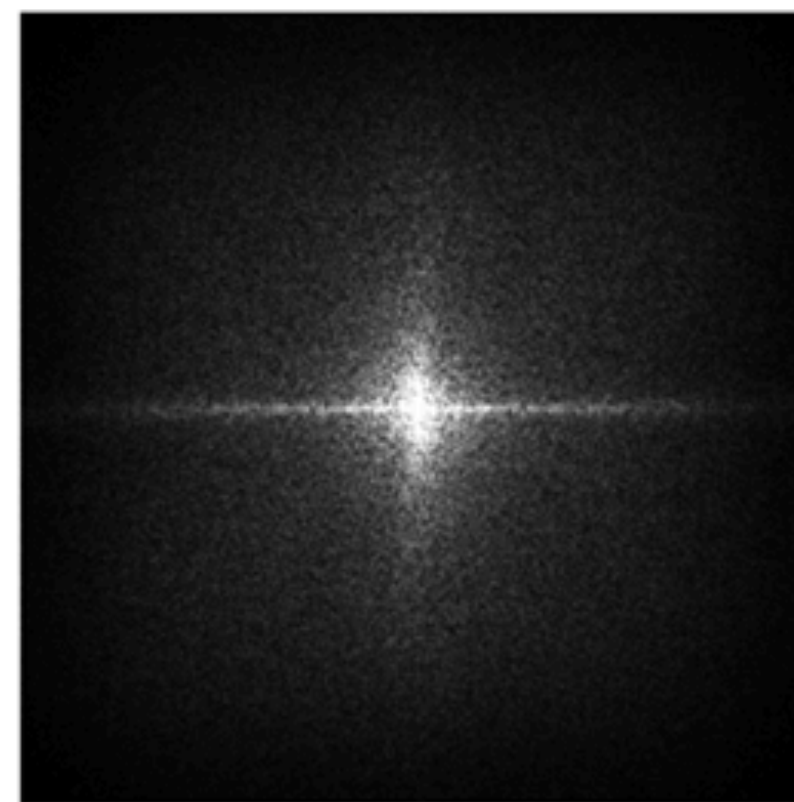
(IV)



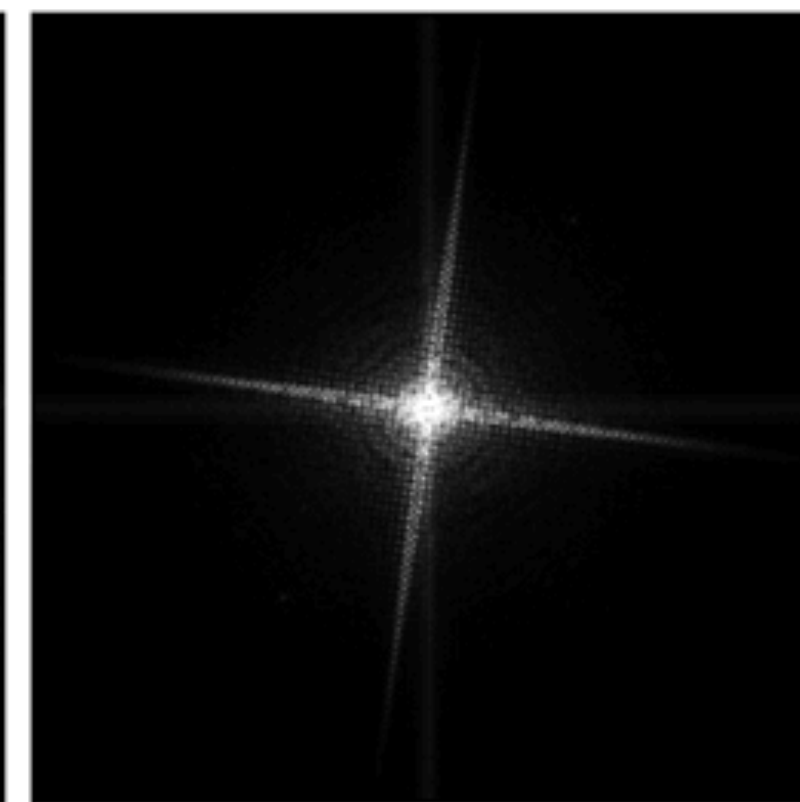
(A)



(B)



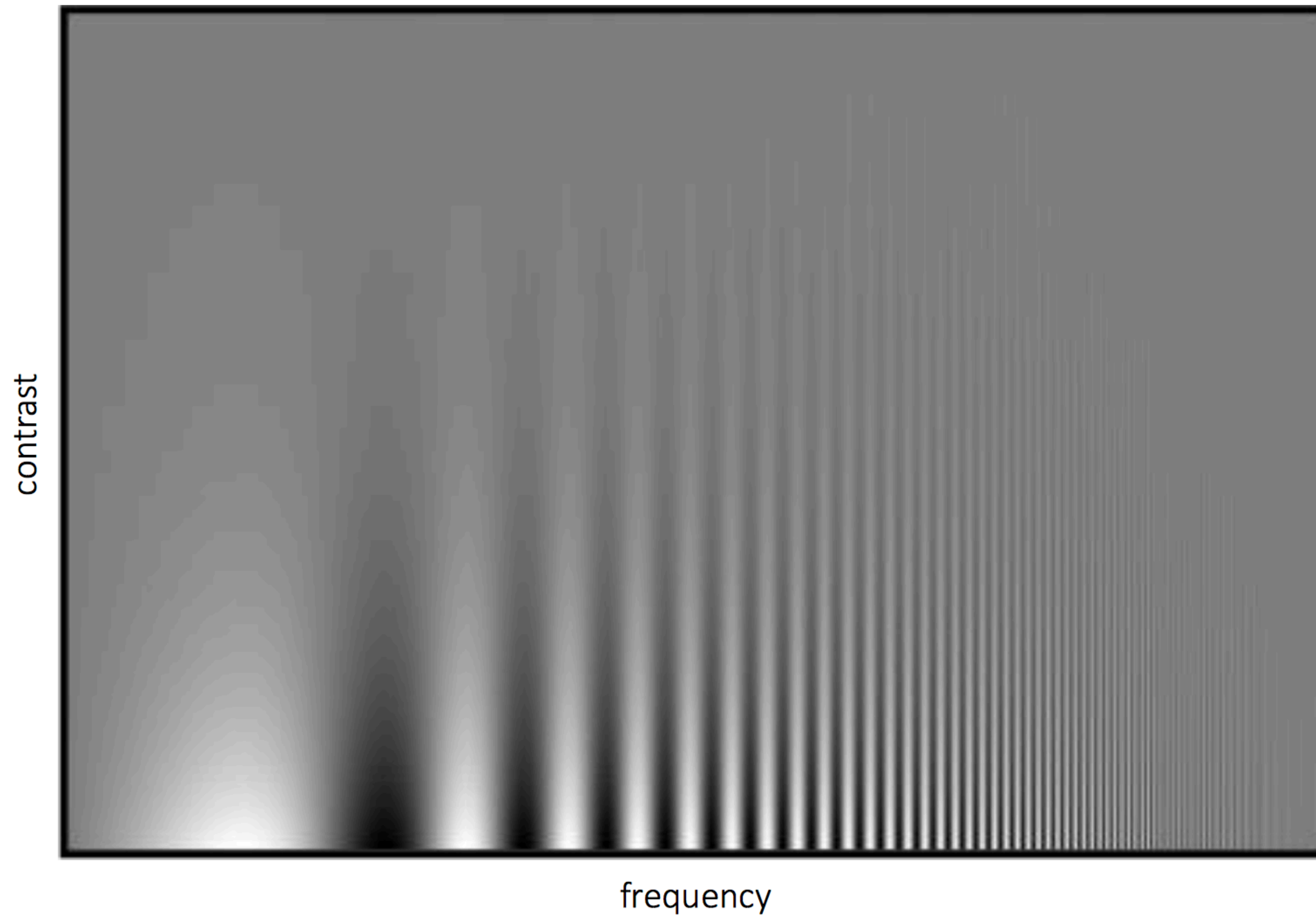
(C)



(D)

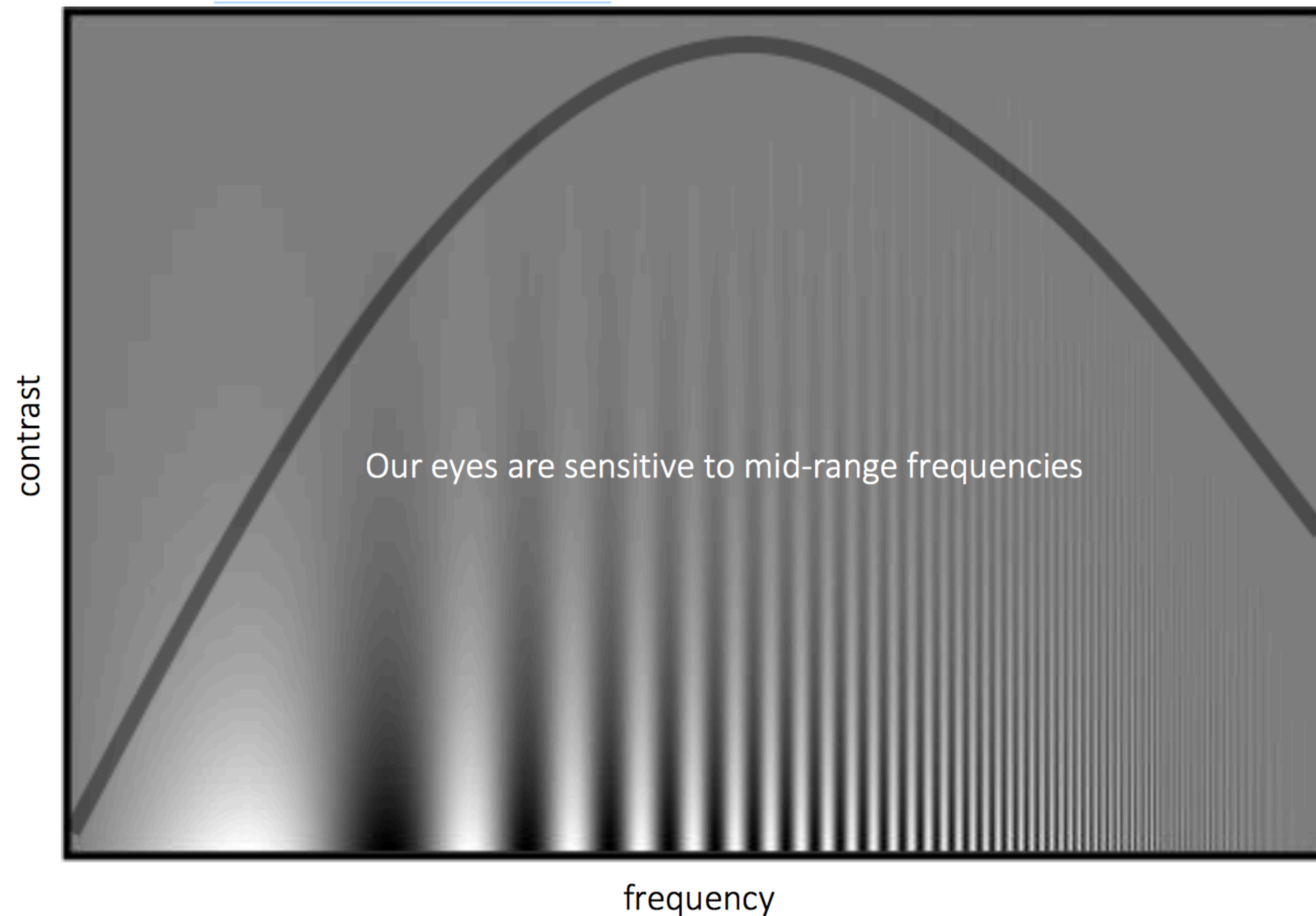
Fourier Transform (you will **NOT** be tested on this)

Experiment: Where of you see the stripes?



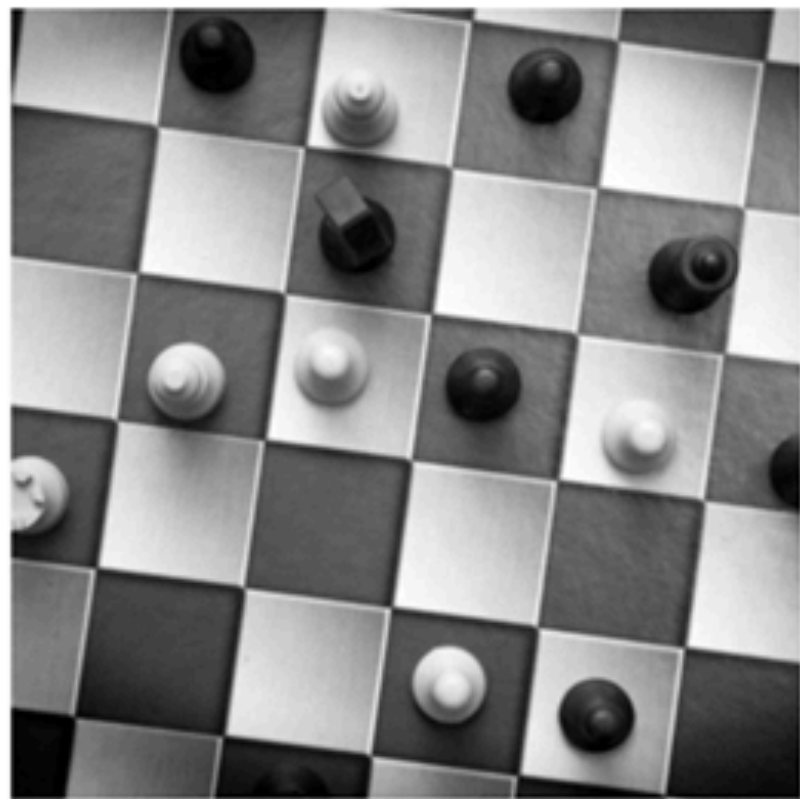
Fourier Transform (you will **NOT** be tested on this)

Campbell-Robson contrast sensitivity curve



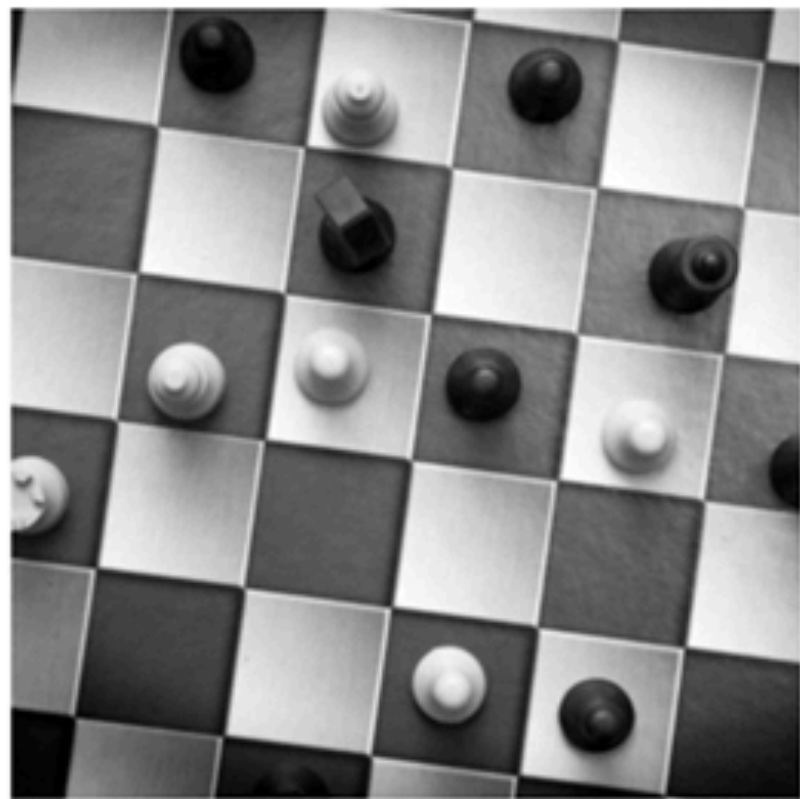
Fourier Transform (you will **NOT** be tested on this)

Distance to the screen will change the field of view of your eye and, as a result, frequency spectra of the image being formed on your retina



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

Distance to the screen will change the field of view of your eye and, as a result, frequency spectra of the image being formed on your retina



As you come **closer**, higher frequencies come into mid-range

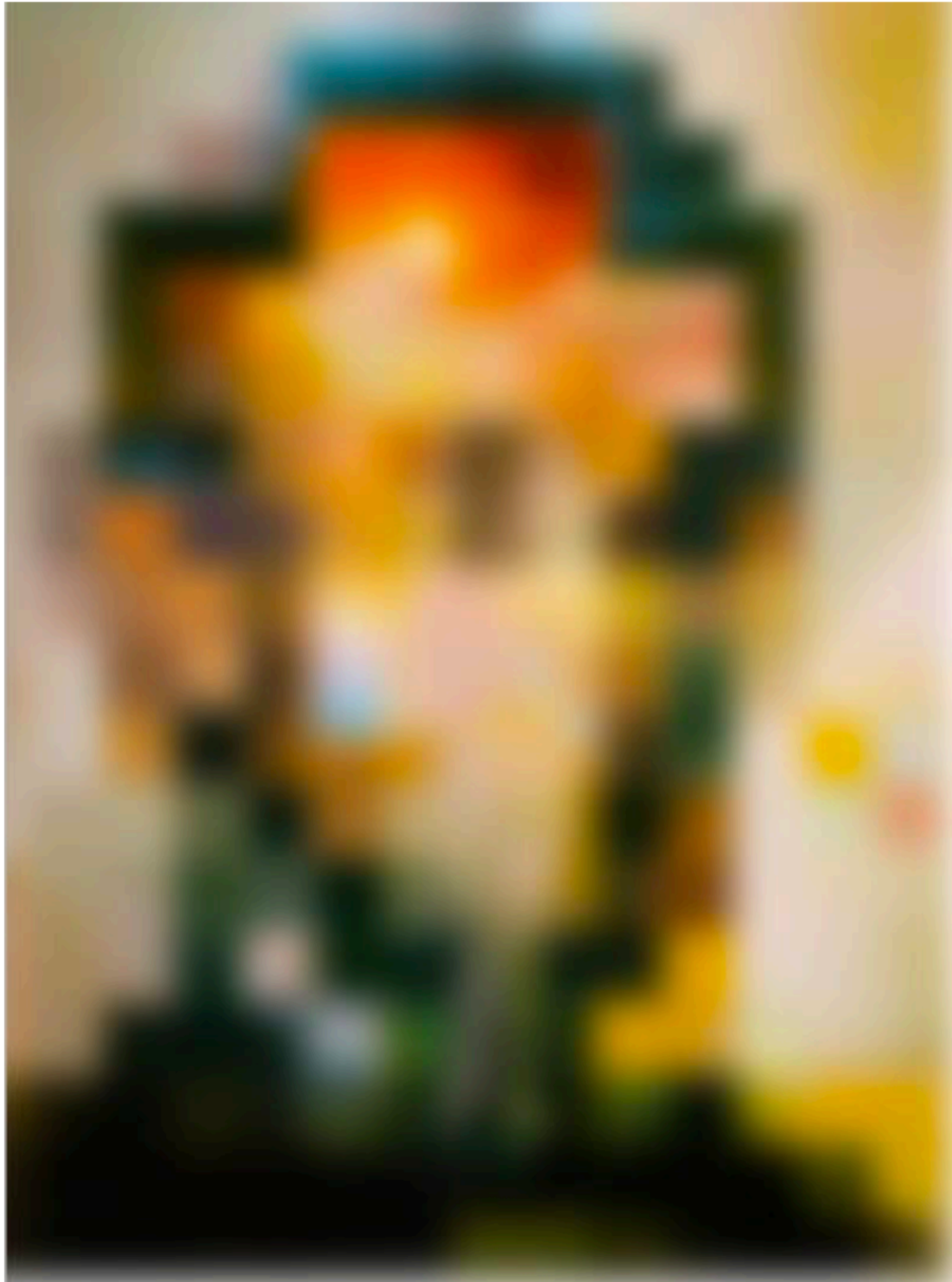
As you move **away**, low frequencies come into mid-range

... back from **detour**

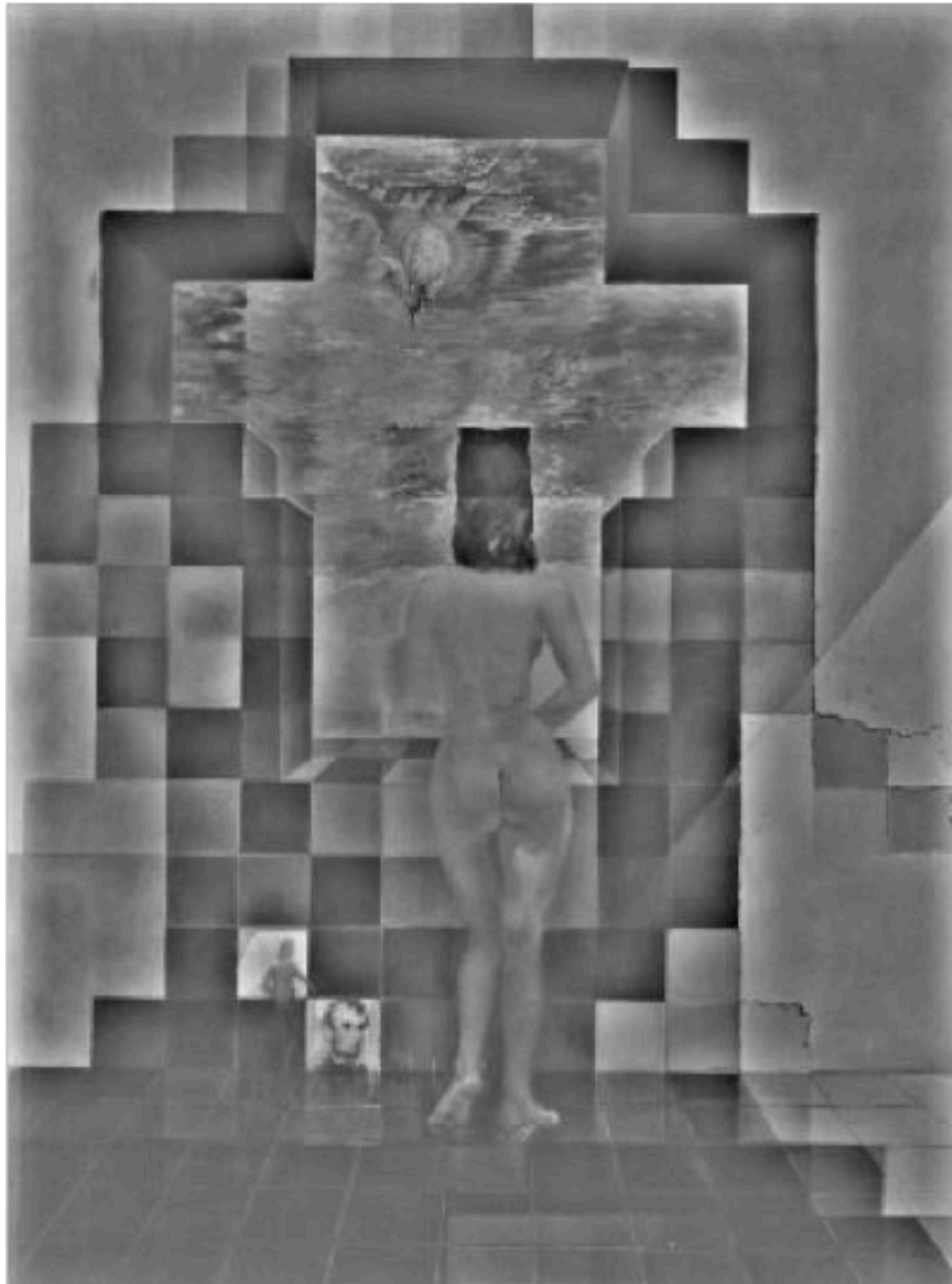


*Gala Contemplating the Mediterranean
Sea Which at Twenty Meters Becomes
the Portrait of Abraham Lincoln
(Homage to Rothko)*

Salvador Dali, 1976

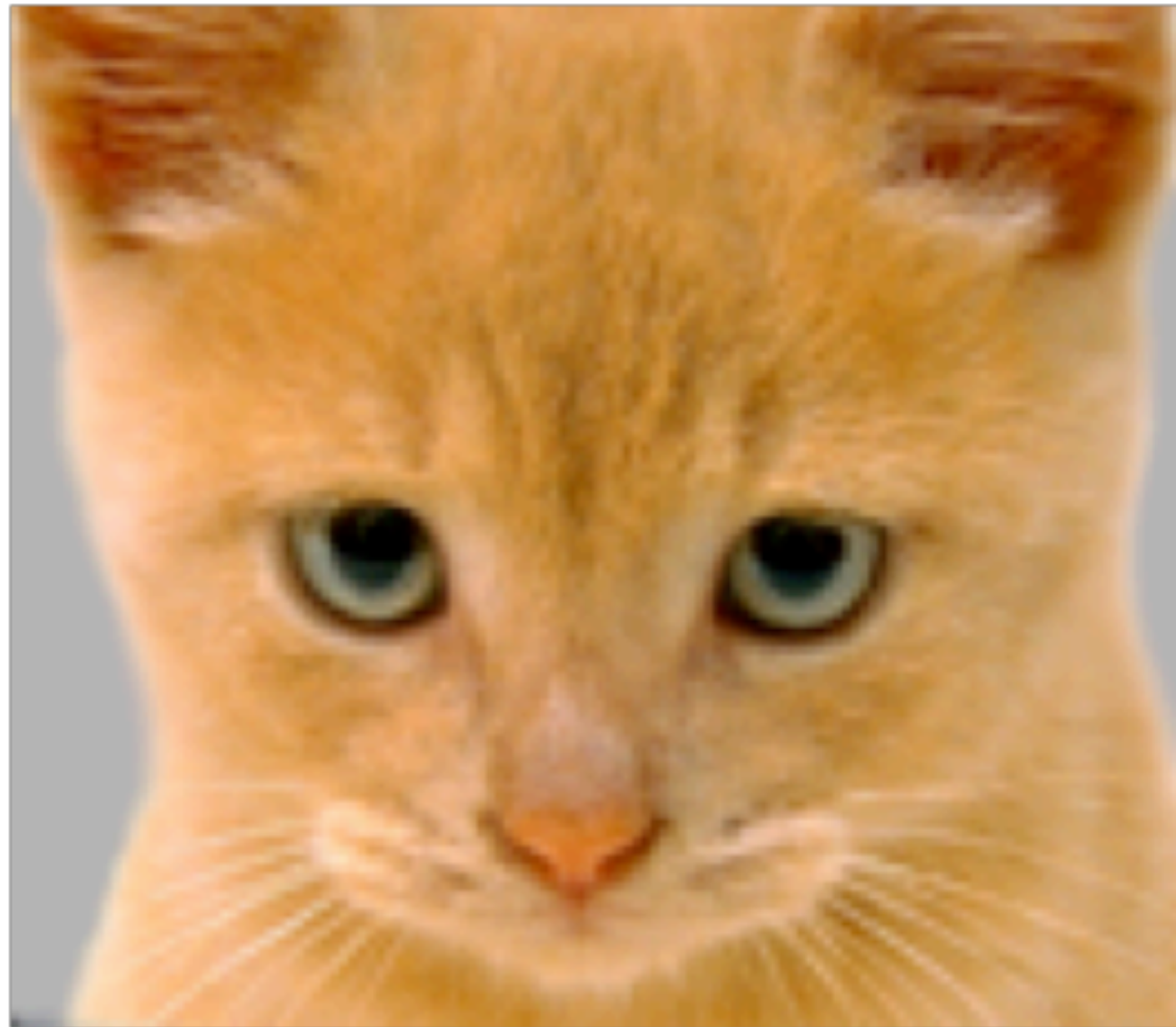


Low-pass filtered version



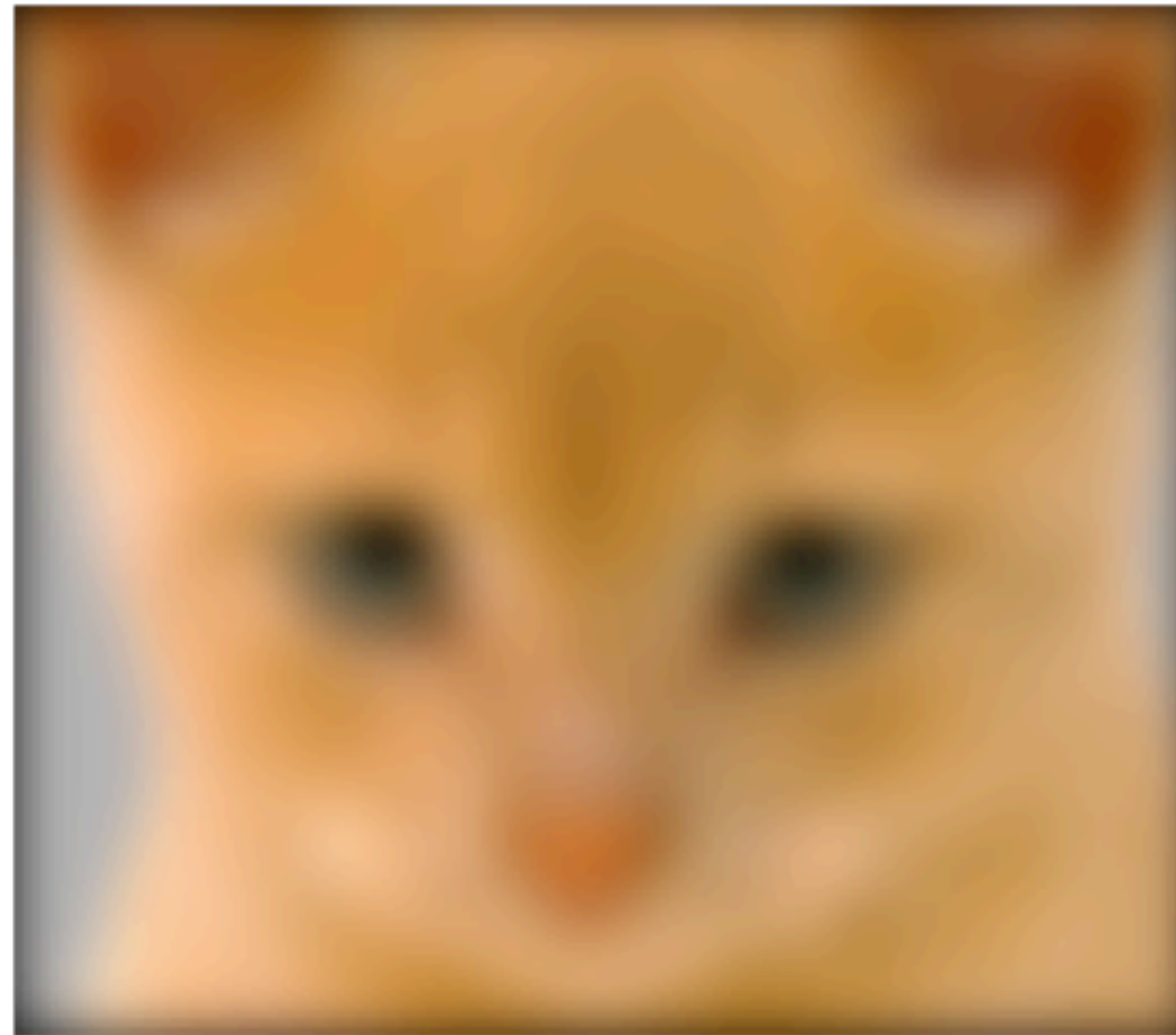
High-pass filtered version

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



Original

$$I(x, y)$$



Low-Pass Filter

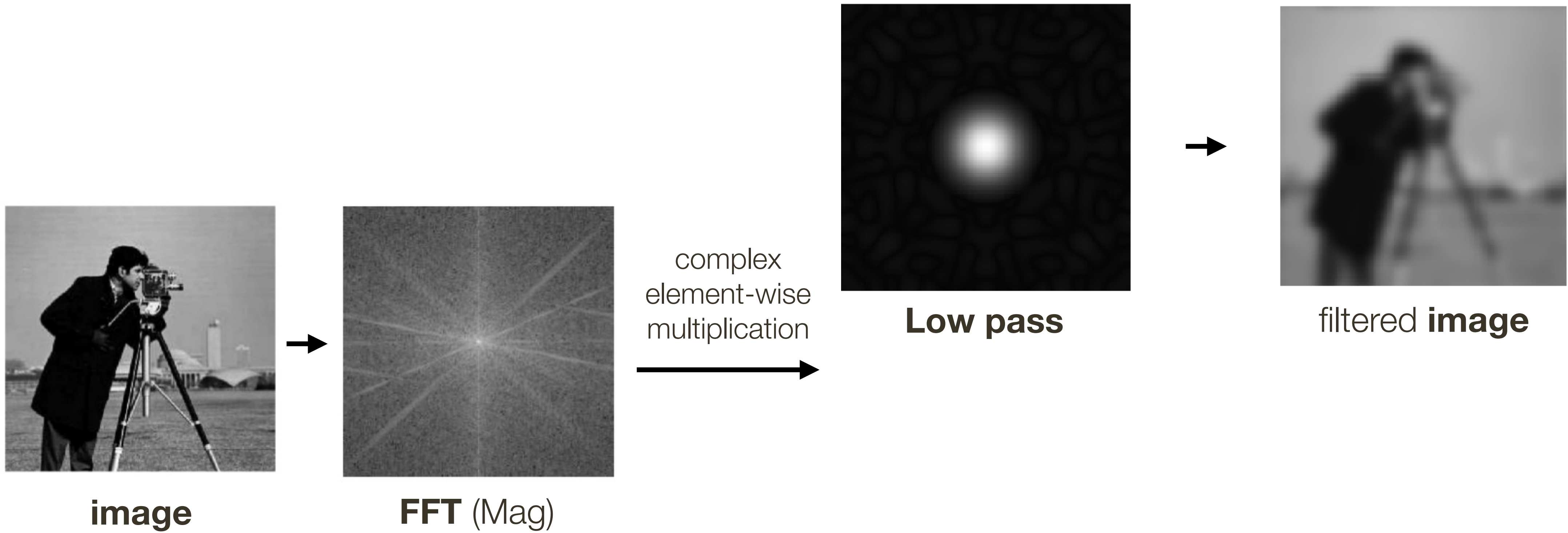
$$I(x, y) * g(x, y)$$



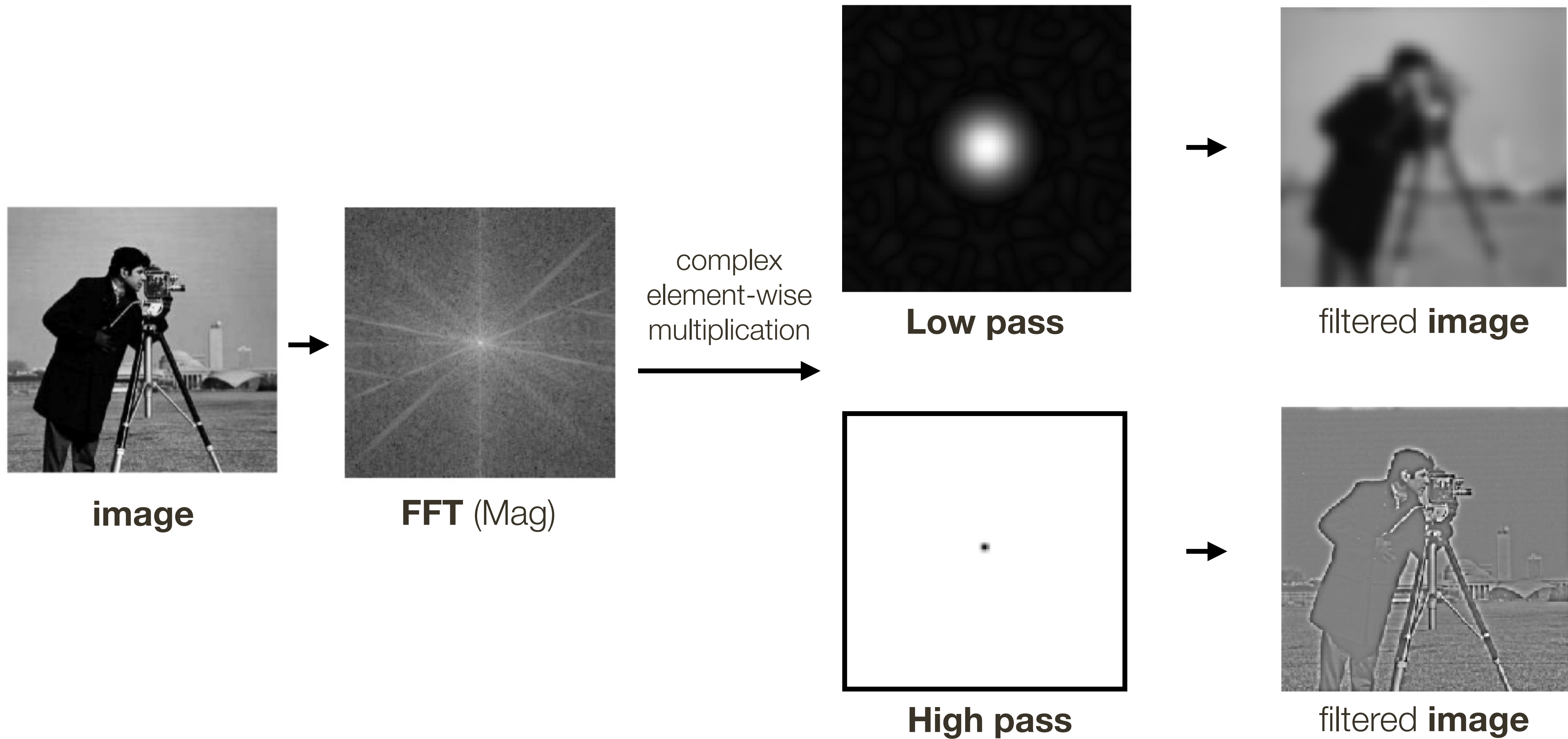
High-Pass Filter

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

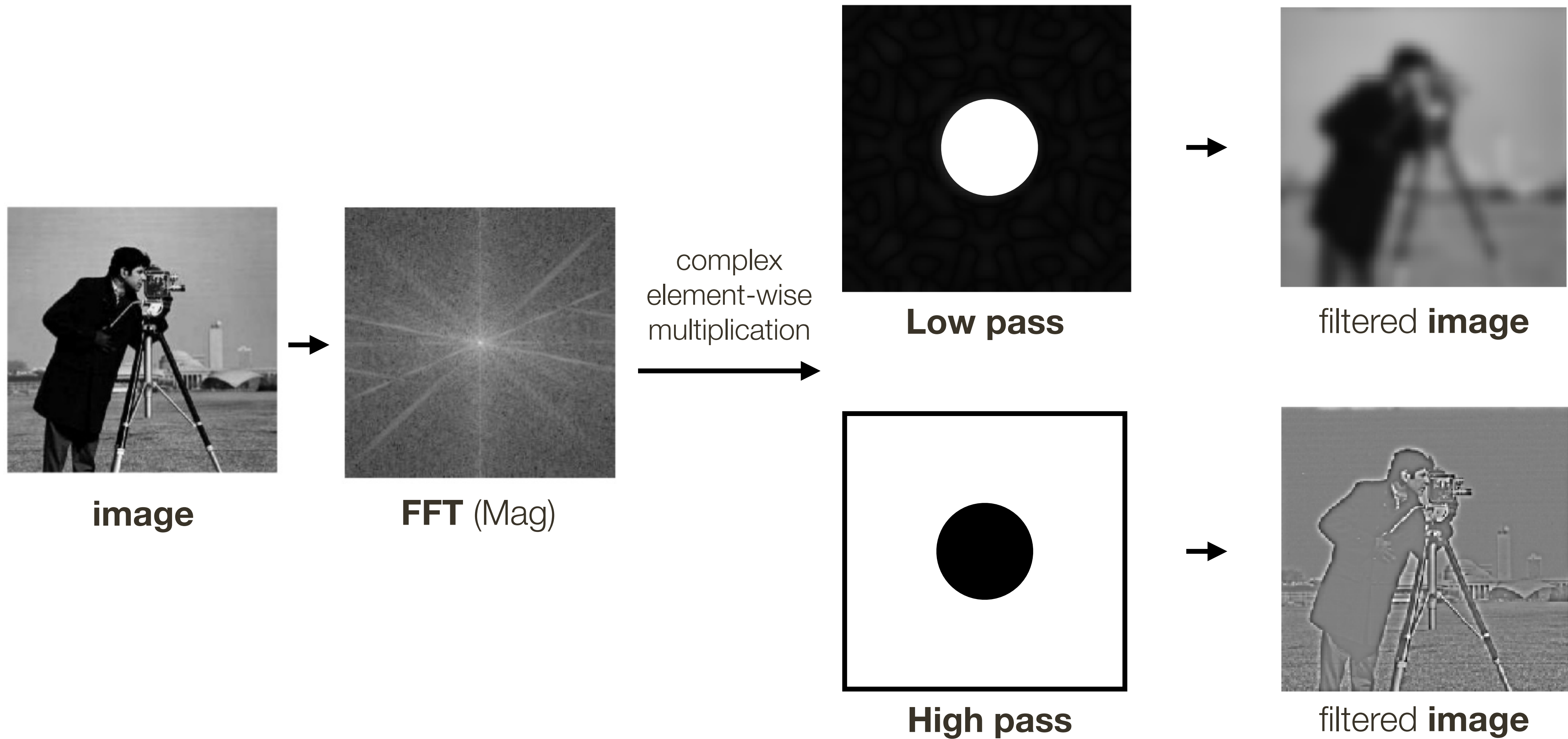
Low-pass / High-pass Filtering



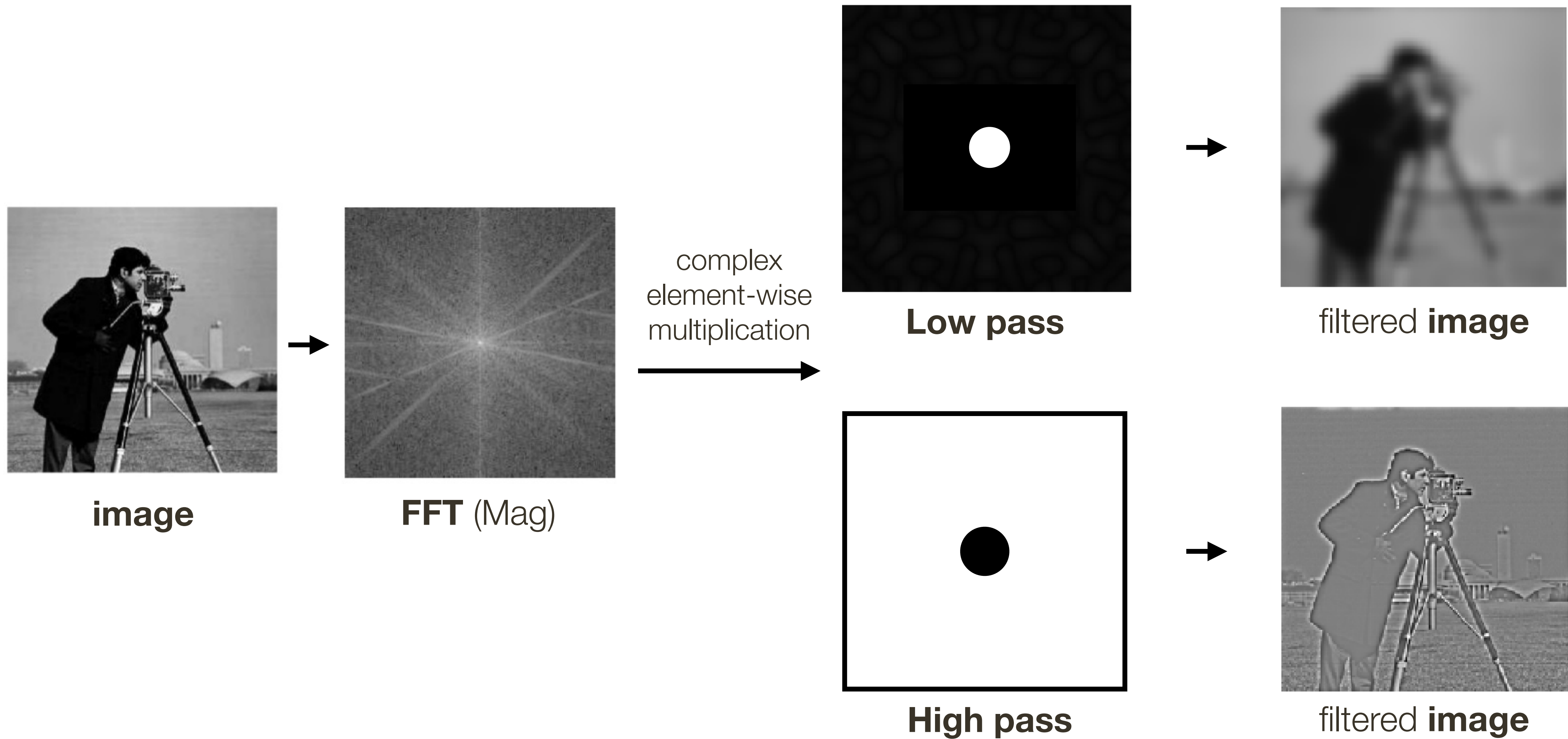
Low-pass / High-pass Filtering



Perfect **Low-pass** / **High-pass** Filtering



Perfect **Low-pass** / **High-pass** Filtering



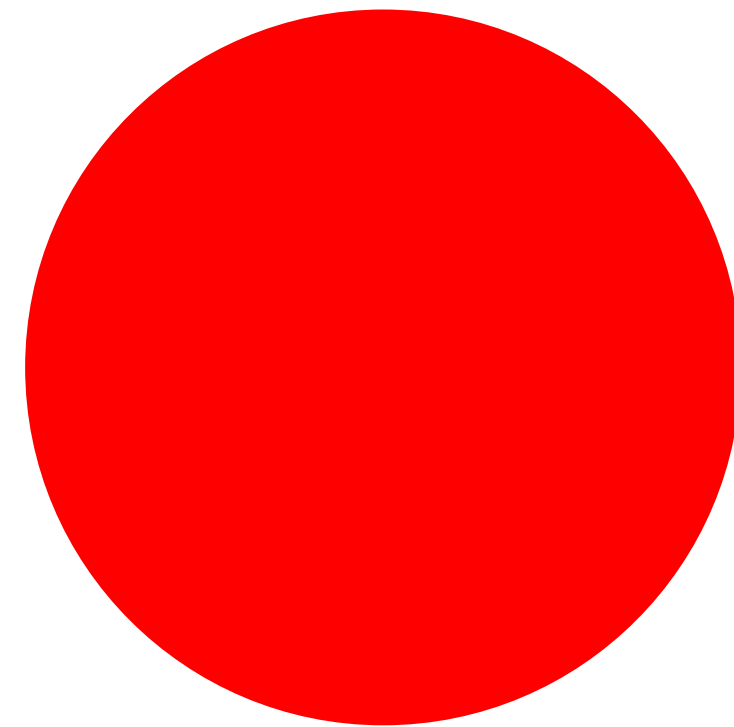
Low-pass Filtering = “Smoothing”?

Box Filter

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

1	1	1
1	1	1
1	1	1

Pillbox Filter



Gaussian Filter

$$\frac{1}{256}$$

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

Are all of these **low-pass** filters?

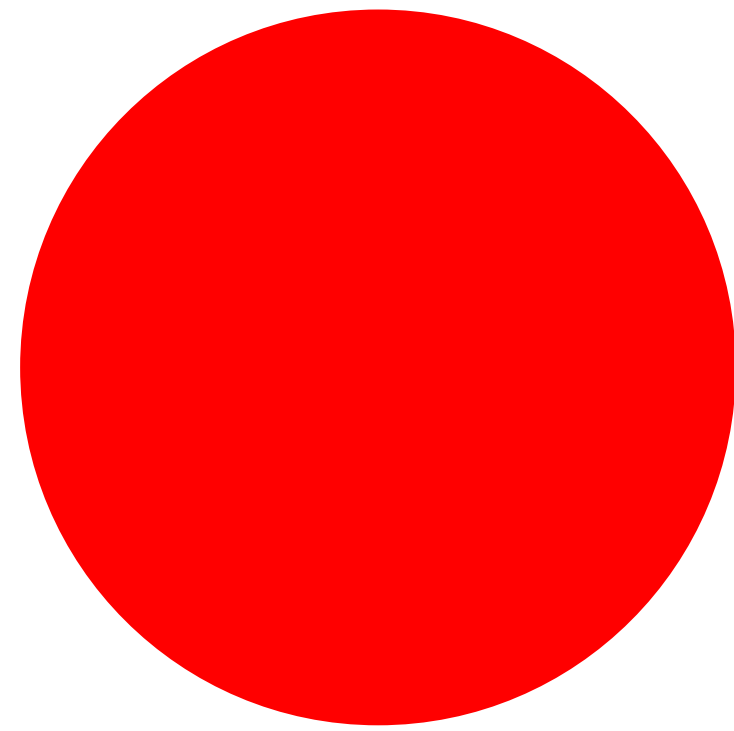
Low-pass Filtering = “Smoothing”

Box Filter

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

1	1	1
1	1	1
1	1	1

Pillbox Filter



Gaussian Filter

$$\frac{1}{256}$$

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

Are all of these **low-pass** filters?

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

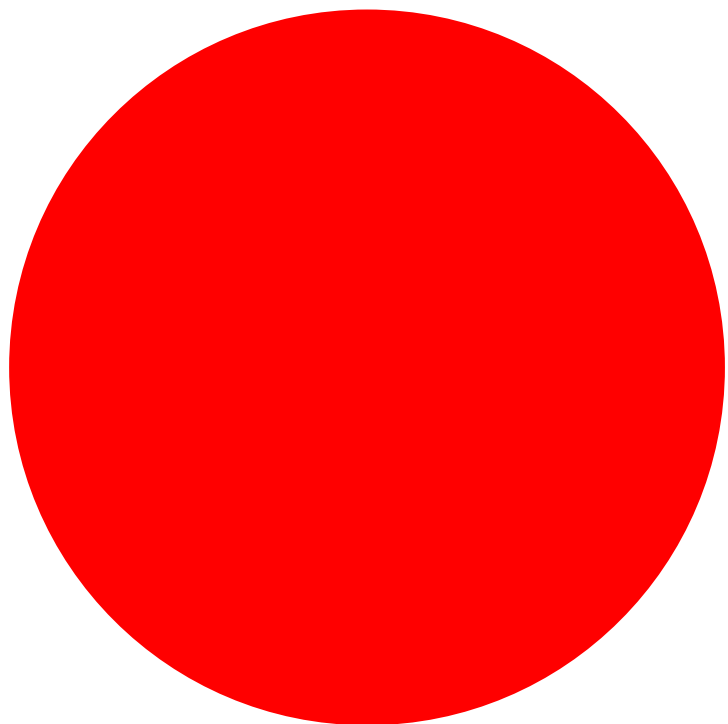
Low-pass Filtering = “Smoothing”

Box Filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Pillbox Filter



Gaussian Filter

$\frac{1}{256}$

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

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

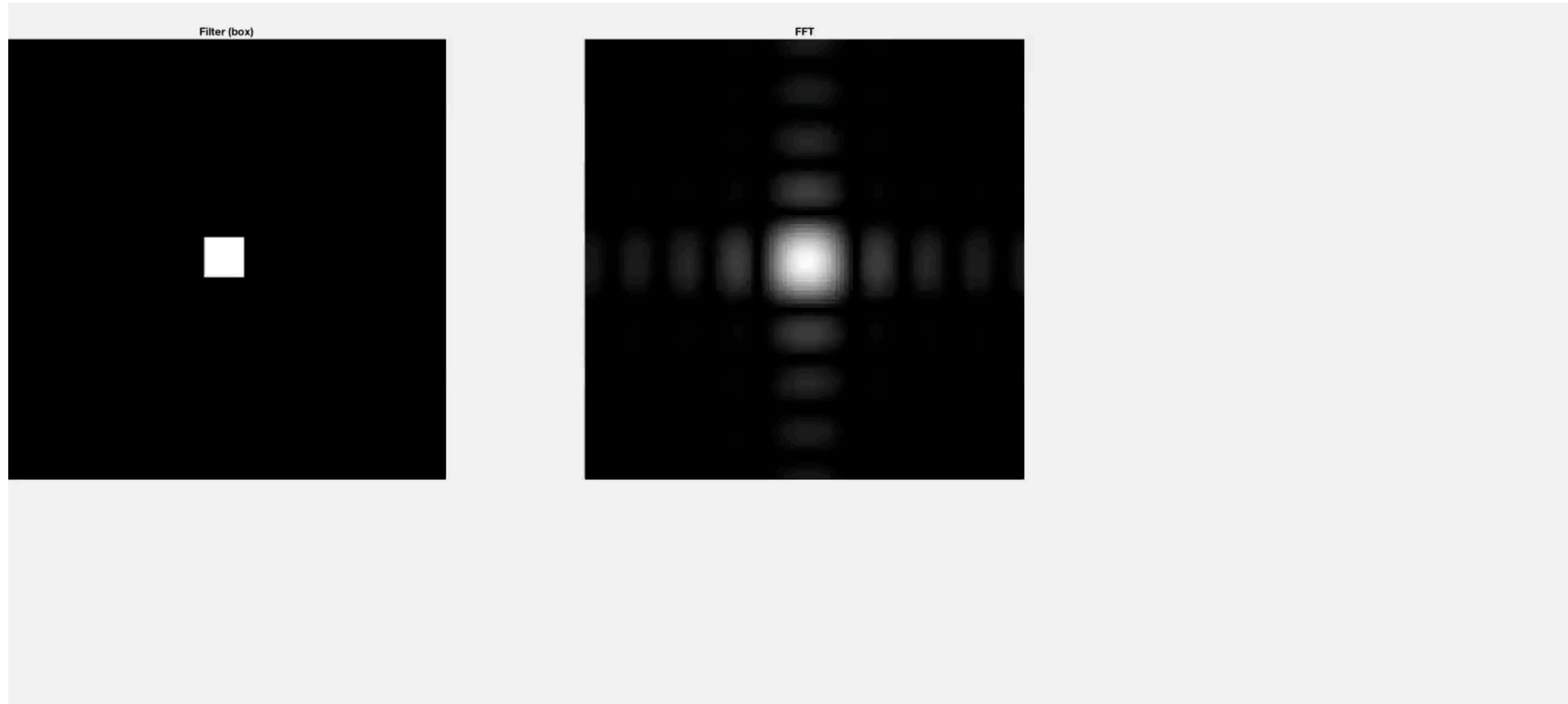
Are all of these **low-pass** filters?

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

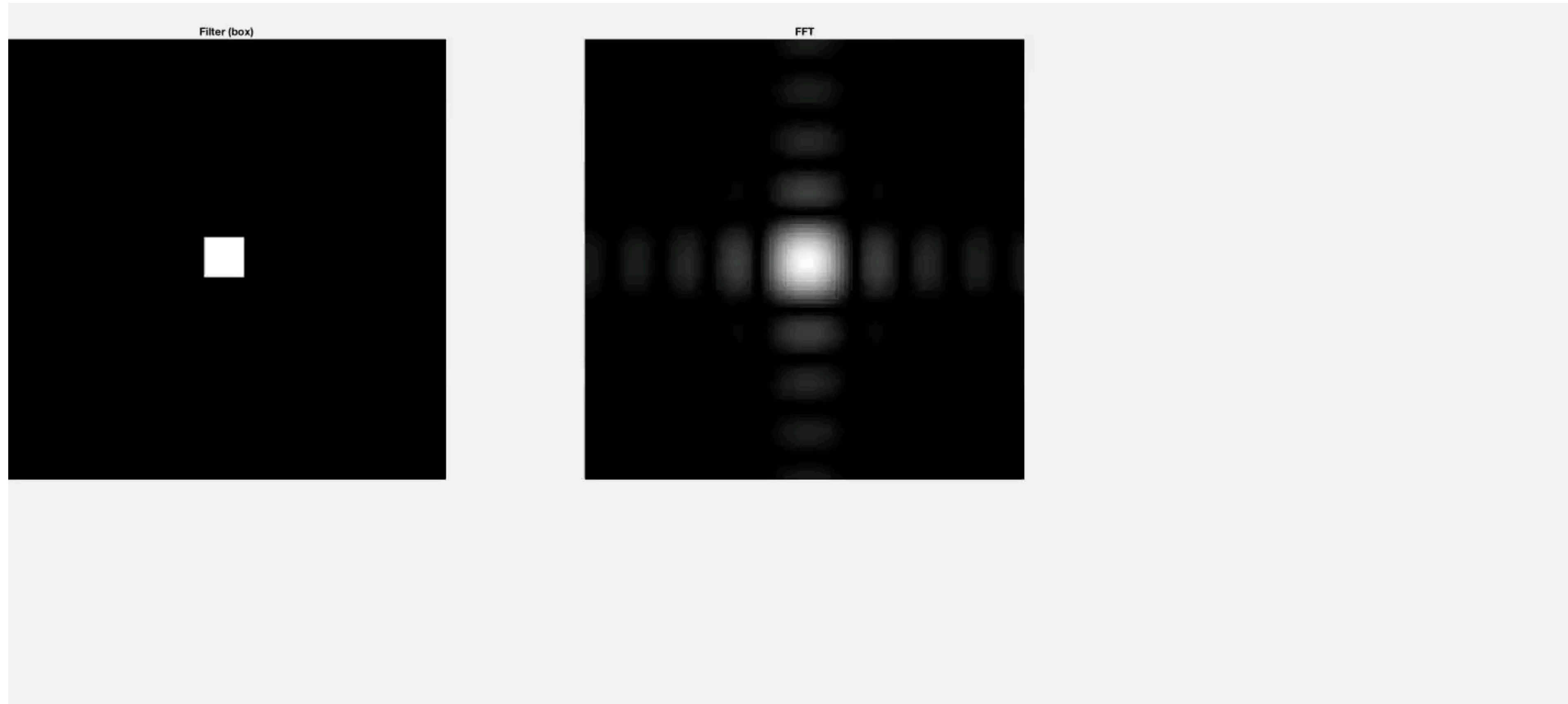
0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

Image

Low-pass Filtering = “Smoothing”



Low-pass Filtering = “Smoothing”



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 Filters: Additional Properties

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

— Convolution is **associative**. That is,

$$G \otimes (F \otimes I(X, Y)) = (G \otimes F) \otimes I(X, Y)$$

— Convolution is **symmetric**. That is,

$$(G \otimes F) \otimes I(X, Y) = (F \otimes G) \otimes I(X, Y)$$

Convolving $I(X, Y)$ with filter F and then convolving the result with filter G can be achieved in single step, namely convolving $I(X, Y)$ with filter $G \otimes F = F \otimes G$

Note: Correlation, in general, is **not associative**.

Associativity Example

A=
[[1 1 6]
[4 1 7]
[9 0 6]]

B=
[[6 6 4]
[1 9 5]
[3 3 8]]

A conv B=
[[40 84 105]
[97 137 130]
[96 107 83]]

B conv A=
[[40 84 105]
[97 137 130]
[96 107 83]]

$$\text{conv}(A, B) = \text{conv}(B, A)$$

A corr B=
[[34 111 79]
[78 159 124]
[109 97 102]]

B corr A=
[[102 97 109]
[124 159 78]
[79 111 34]]

$$\text{corr}(A, B) \neq \text{corr}(B, A)$$

Linear Filters: Additional Properties

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

— Convolution is **associative**. That is,

$$G \otimes (F \otimes I(X, Y)) = (G \otimes F) \otimes I(X, Y)$$

— Convolution is **symmetric**. That is,

$$(G \otimes F) \otimes I(X, Y) = (F \otimes G) \otimes I(X, Y)$$

Convolving $I(X, Y)$ with filter F and then convolving the result with filter G can be achieved in single step, namely convolving $I(X, Y)$ with filter $G \otimes F = F \otimes G$

Note: Correlation, in general, is **not associative**.

Example: Two Box Filters

```
filter = boxfilter(3)
```

```
signal.correlate2d(filter, filter, 'full')
```

 $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

3x3 **Box**

$\otimes$

 $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

3x3 **Box**

=

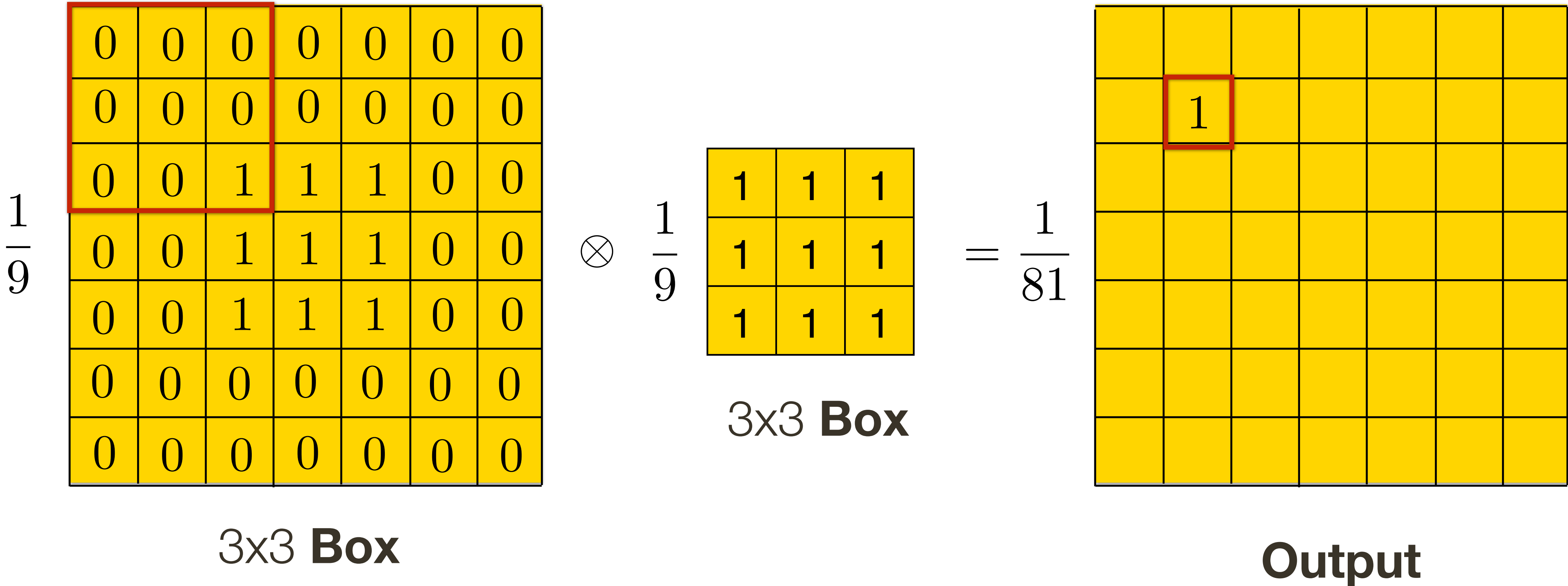
$\frac{1}{81}$

1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

Example: Two Box Filters

Treat one filter as padded “image”

Note, in this case you have to pad maximally until two filters no longer overlap



Example: Two Box Filters

Treat one filter as padded “image”

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

3x3 Box


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

1	1	1
1	1	1
1	1	1

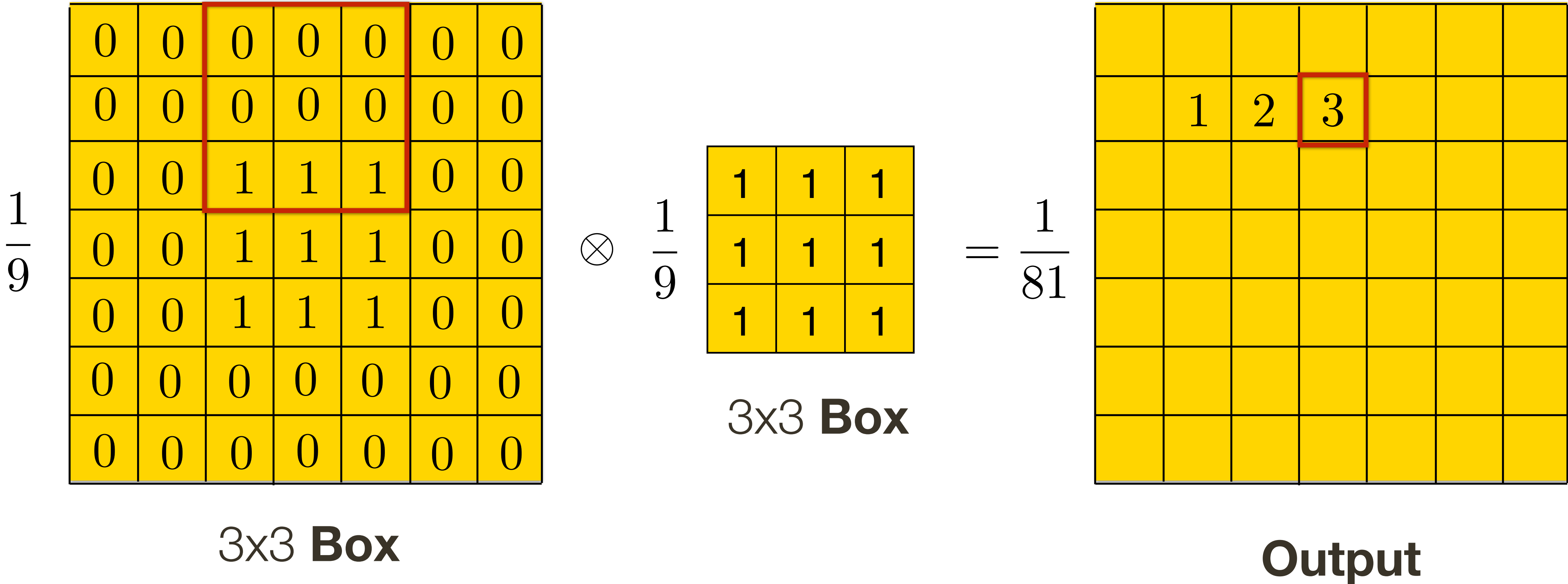
3x3 Box

$$= \frac{1}{81}$$
[illegible]

Output

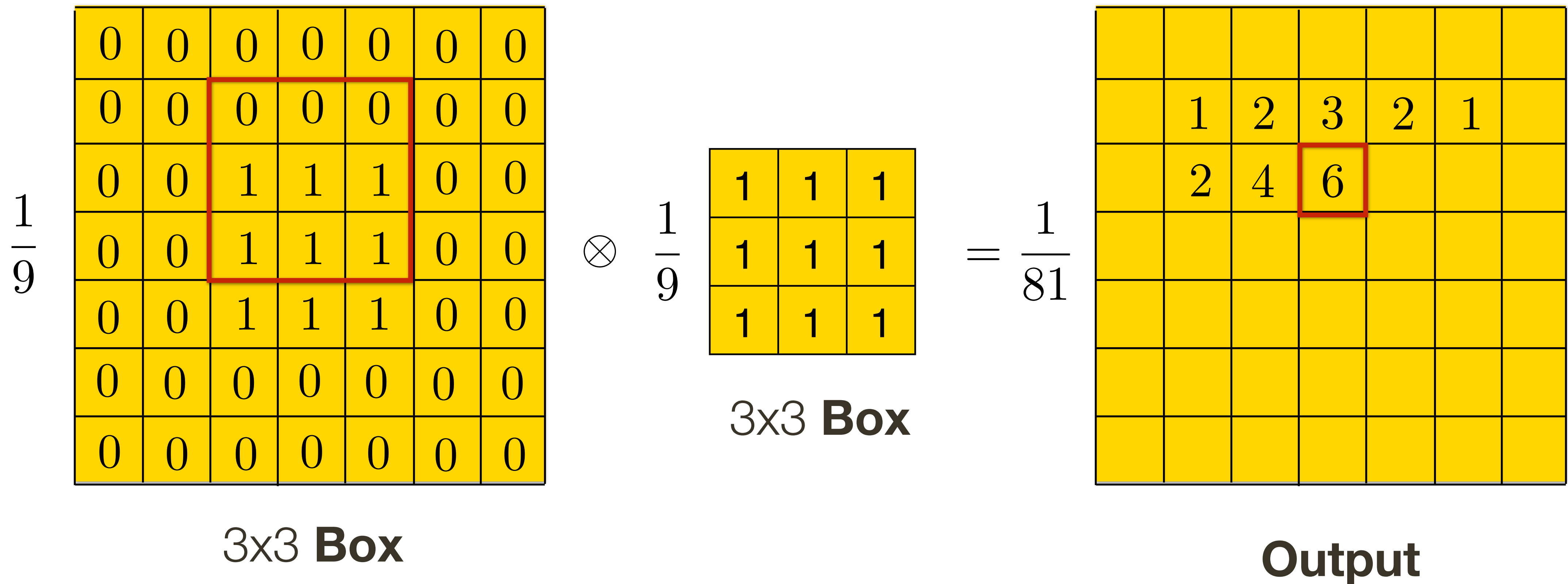
Example: Two Box Filters

Treat one filter as padded “image”



Example: Two Box Filters

Treat one filter as padded “image”



Example: Two Box Filters

Treat one filter as padded “image”

$$\frac{1}{9} \begin{array}{|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} \otimes \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \frac{1}{81} \begin{array}{|c|c|c|c|c|c|c|} \hline & & & & & & \\ \hline & 1 & 2 & 3 & 2 & 1 & \\ \hline & 2 & 4 & 6 & 4 & 2 & \\ \hline & 3 & 6 & 9 & 6 & 3 & \\ \hline & 2 & 4 & 6 & 4 & 2 & \\ \hline & 1 & 2 & 3 & 2 & 1 & \\ \hline & & & & & & \\ \hline \end{array}$$

3x3 Box **3x3 Box** **Output**

Example: Two Box Filters

Treat one filter as padded “image”

$$\frac{1}{9} \begin{array}{|c|c|c|c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} \otimes \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \frac{1}{81} \begin{array}{|c|c|c|c|c|} \hline 1 & 2 & 3 & 2 & 1 \\ \hline 2 & 4 & 6 & 4 & 2 \\ \hline 3 & 6 & 9 & 6 & 3 \\ \hline 2 & 4 & 6 & 4 & 2 \\ \hline 1 & 2 & 3 & 2 & 1 \\ \hline \end{array}$$

3x3 Box **3x3 Box** **Output**

Example: Two Box Filters

```
filter = boxfilter(3)
```

```
temp = signal.correlate2d(filter, filter, 'full')
```

```
signal.correlate2d(filter, temp, 'full')
```

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \otimes \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \otimes \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{729} \begin{bmatrix} 1 & 3 & 6 & 7 & 6 & 3 & 1 \\ 3 & 9 & 18 & 21 & 18 & 9 & 3 \\ 6 & 18 & 36 & 42 & 36 & 18 & 6 \\ 7 & 21 & 42 & 49 & 42 & 21 & 7 \\ 6 & 18 & 36 & 42 & 36 & 18 & 6 \\ 3 & 9 & 18 & 21 & 18 & 9 & 3 \\ 1 & 3 & 6 & 7 & 6 & 3 & 1 \end{bmatrix}$$

3x3 **Box** 3x3 **Box** 3x3 **Box**

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

Diagram illustrating the Kronecker product of two matrices:

Matrix 1 (5x5):

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

Matrix 2 (5x1):

1
4
6
4
1

Operation: $\otimes$

Result (5x5):

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
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Example: Separable Gaussian Filter

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

The diagram illustrates the separable convolution of a 9x5 grid with a 5x1 kernel. The first grid is a 9x5 matrix with a red box highlighting the 3x3 sub-region from row 4 to row 6 and column 2. The second grid is a 5x1 column vector with a red box highlighting the entire vector. The third grid is a 9x5 matrix with a red box highlighting the element at row 4, column 2, which is the result of the convolution of the highlighted 3x3 sub-region with the kernel.

Example: Separable Gaussian Filter

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

Example: Separable Gaussian Filter

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

Pre-Convoluting Filters

Convolving two filters of size $m \times m$ and $n \times n$ results in filter of size:

$$(n + m - 1) \times (n + m - 1)$$

More broadly for a set of K filters of sizes $m_k \times m_k$ the resulting filter will have size:

$$\left(m_1 + \sum_{k=2}^K (m_k - 1) \right) \times \left(m_1 + \sum_{k=2}^K (m_k - 1) \right)$$

Gaussian: An Additional Property

Let $\otimes$ denote convolution. Let $G_{\sigma_1}(x)$ and $G_{\sigma_2}(x)$ be two 1D Gaussians

$$G_{\sigma_1}(x) \otimes G_{\sigma_2}(x) = G_{\sqrt{\sigma_1^2 + \sigma_2^2}}(x)$$

Convolution of two Gaussians is another Gaussian

Special case: Convoluting with $G_{\sigma}(x)$ twice is equivalent to $G_{\sqrt{2}\sigma}(x)$

Non-linear Filters

We've seen that **linear filters** can perform a variety of image transformations

- shifting
- smoothing
- sharpening

In some applications, better performance can be obtained by using **non-linear filters**.

For example, the median filter (which is a very effective de-noising / smoothing filter) selects the **median** value from each pixel's neighborhood.

Median Filter

Take the **median value** of the pixels under the filter:

5	13	5	221
4	16	7	34
24	54	34	23
23	75	89	123
54	25	67	12

Image

Output

Median Filter

Take the **median value** of the pixels under the filter:

5	13	5	221
4	16	7	34
24	54	34	23
23	75	89	123
54	25	67	12

Image

4	5	5	7	13	16	24	34	54
---	---	---	---	----	----	----	----	----

Output

Median Filter

Take the **median value** of the pixels under the filter:

5	13	5	221
4	16	7	34
24	54	34	23
23	75	89	123
54	25	67	12

Image

4	5	5	7	13	16	24	34	54
---	---	---	---	----	----	----	----	----



	13		

Output

Median Filter

Effective at reducing certain kinds of noise, such as impulse noise (a.k.a 'salt and pepper' noise or 'shot' noise)

The median filter forces points with distinct values to be more like their neighbors



Image credit: https://en.wikipedia.org/wiki/Median_filter#/media/File:Medianfilterp.png

Bilateral Filter

An edge-preserving non-linear filter

Like a Gaussian filter:

- The filter weights depend on spatial distance from the center pixel
- Pixels nearby (in space) should have greater influence than pixels far away

Unlike a Gaussian filter:

- The filter weights also depend on range distance from the center pixel
- Pixels with similar brightness value should have greater influence than pixels with dissimilar brightness value

Bilateral Filter

Gaussian filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by:

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

(with appropriate normalization)

Bilateral Filter

Gaussian filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by:

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

(with appropriate normalization)

Bilateral filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by a product:

$$\exp^{-\frac{x^2+y^2}{2\sigma_d^2}} \exp^{-\frac{(I(X+x, Y+y) - I(X, Y))^2}{2\sigma_r^2}}$$

(with appropriate normalization)

Bilateral Filter

Gaussian filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by:

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

(with appropriate normalization)

Bilateral filter: weights of neighbor at a spatial offset (x, y) away from the center pixel $I(X, Y)$ given by a product:

domain kernel	$\exp^{-\frac{x^2+y^2}{2\sigma_d^2}}$	$\exp^{-\frac{(I(X+x, Y+y) - I(X, Y))^2}{2\sigma_r^2}}$	range kernel
-------------------------	---------------------------------------	---	------------------------

(with appropriate normalization)

Bilateral Filter

image $I(X, Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255

Bilateral Filter

image $I(X, Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255



image $I(X, Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Bilateral Filter

image $I(X, Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255



image $I(X, Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel
 $\sigma_d = 0.45$

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Bilateral Filter

image $I(X, Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255



image $I(X, Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel
 $\sigma_d = 0.45$

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Range Kernel

$$\sigma_r = 0.45$$

0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

(this is different for each locations in the image)

Bilateral Filter

image $I(X, Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255



image $I(X, Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel
 $\sigma_d = 0.45$

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Range Kernel
 $\sigma_r = 0.45$

0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

multiply



Range * Domain Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

(this is different for each locations in the image)

Bilateral Filter

image $I(X, Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255



image $I(X, Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel
 $\sigma_d = 0.45$

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Range Kernel
 $\sigma_r = 0.45$

0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

multiply



Range * Domain Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

sum to 1



0.11	0.16	0.03
0.16	0.26	0.01
0.11	0.16	0.01

(this is different for each locations in the image)

Bilateral Filter

image $I(X, Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255



image $I(X, Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel
 $\sigma_d = 0.45$

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Range Kernel
 $\sigma_r = 0.45$

0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

multiply



Range * Domain Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

(this is different for each locations in the image)

Σ

0.11	0.16	0.03
0.16	0.26	0.01
0.11	0.16	0.01

$\times$

0	0	0.9
0.1	0.1	1
0	0.1	1

$= 0.1$

Bilateral Filter

Bilateral Filter

image $I(X, Y)$

25	0	25	255	255	255
0	0	0	230	255	255
0	25	25	255	230	255
0	0	25	255	255	255



image $I(X, Y)$

0.1	0	0.1	1	1	1
0	0	0	0.9	1	1
0	0.1	0.1	1	0.9	1
0	0	0.1	1	1	1

Domain Kernel
 $\sigma_d = 0.45$

$$\sum \begin{bmatrix} 0.08 & 0.12 & 0.08 \\ 0.12 & 0.20 & 0.12 \\ 0.08 & 0.12 & 0.08 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0.9 \\ 0.1 & 0.1 & 1 \\ 0 & 0.1 & 1 \end{bmatrix} = 0.3$$

Gaussian Filter (only)

Range Kernel
 $\sigma_r = 0.45$

0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

multiply



Range * Domain Kernel

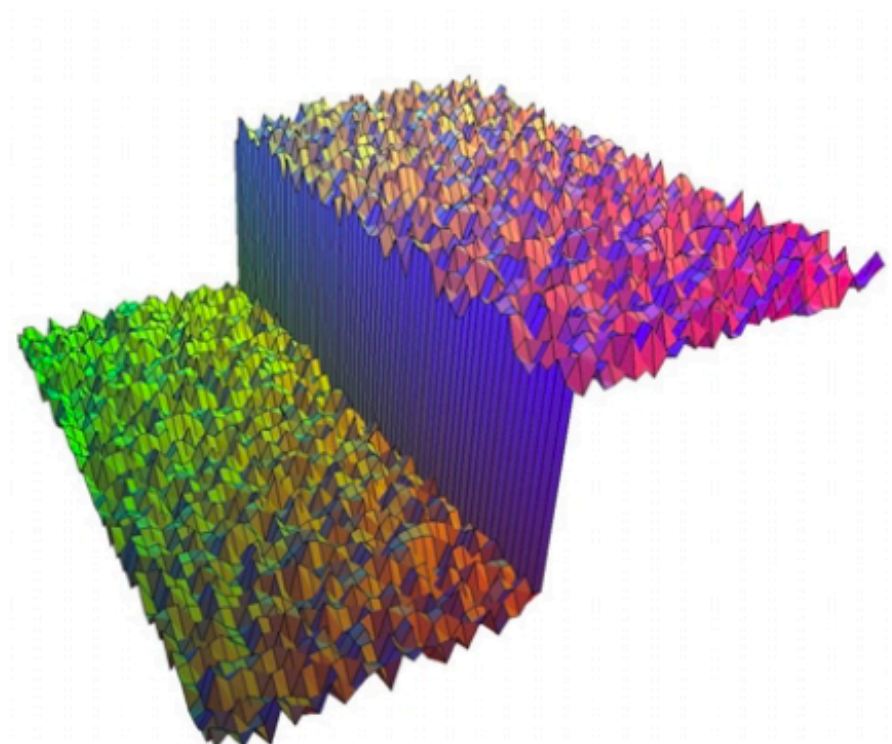
0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

(this is different for each locations in the image)

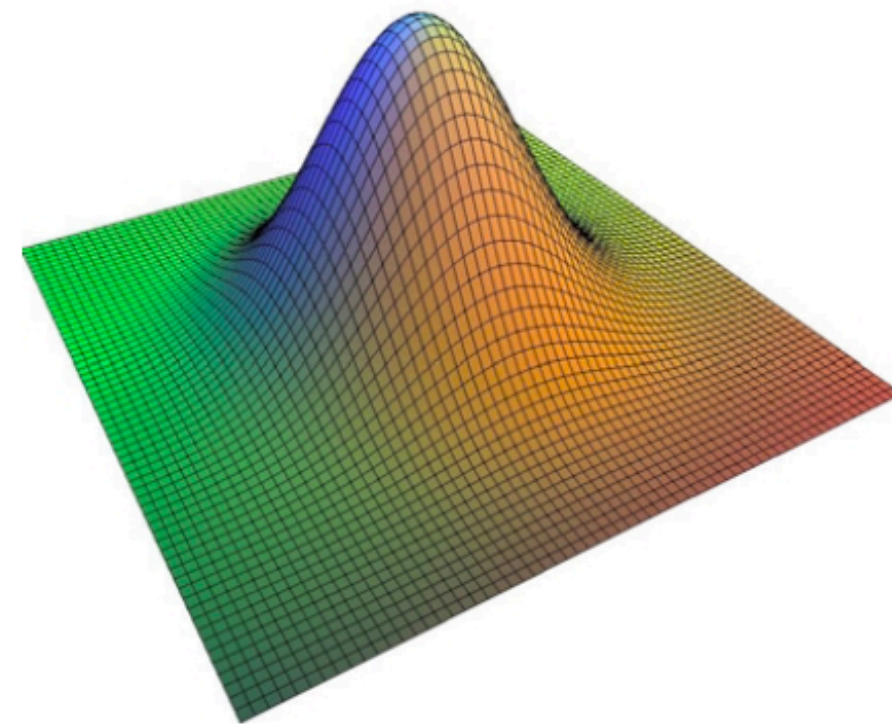
$$\sum \begin{bmatrix} 0.11 & 0.16 & 0.03 \\ 0.16 & 0.26 & 0.01 \\ 0.11 & 0.16 & 0.01 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0.9 \\ 0.1 & 0.1 & 1 \\ 0 & 0.1 & 1 \end{bmatrix} = 0.1$$

Bilateral Filter

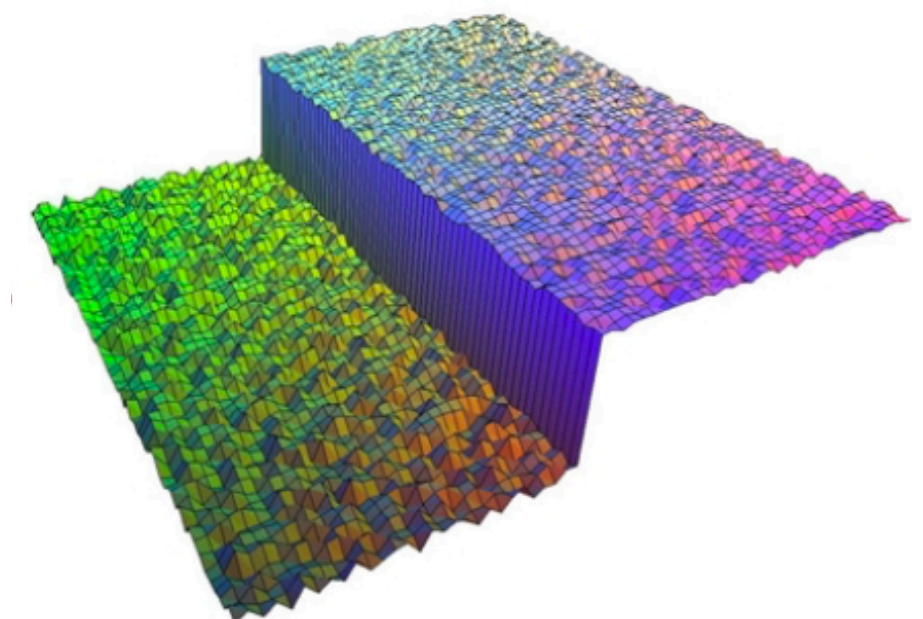
Bilateral Filter



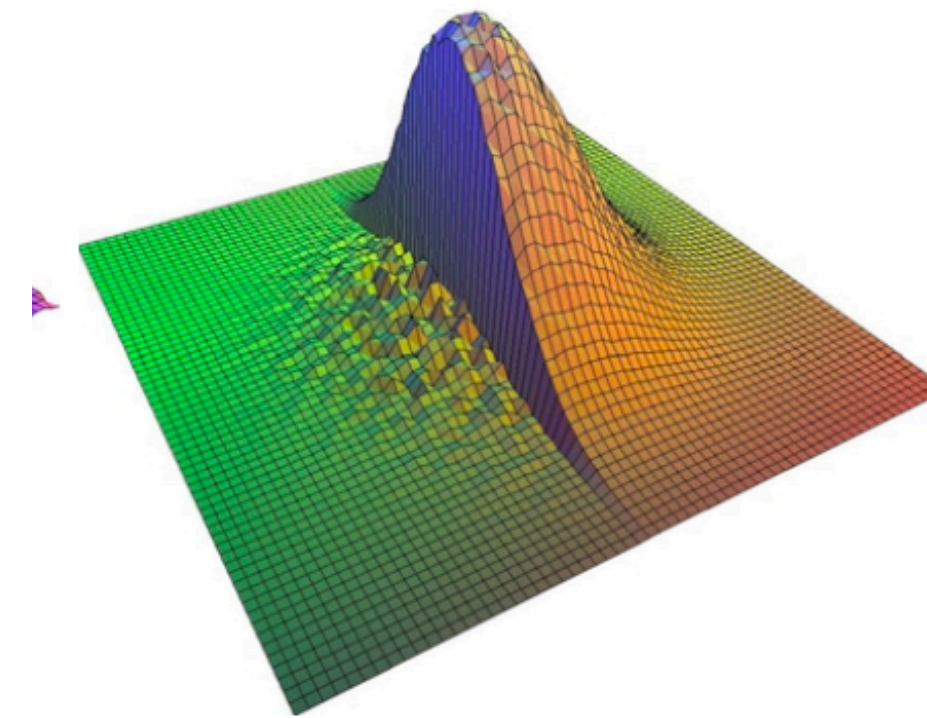
Input



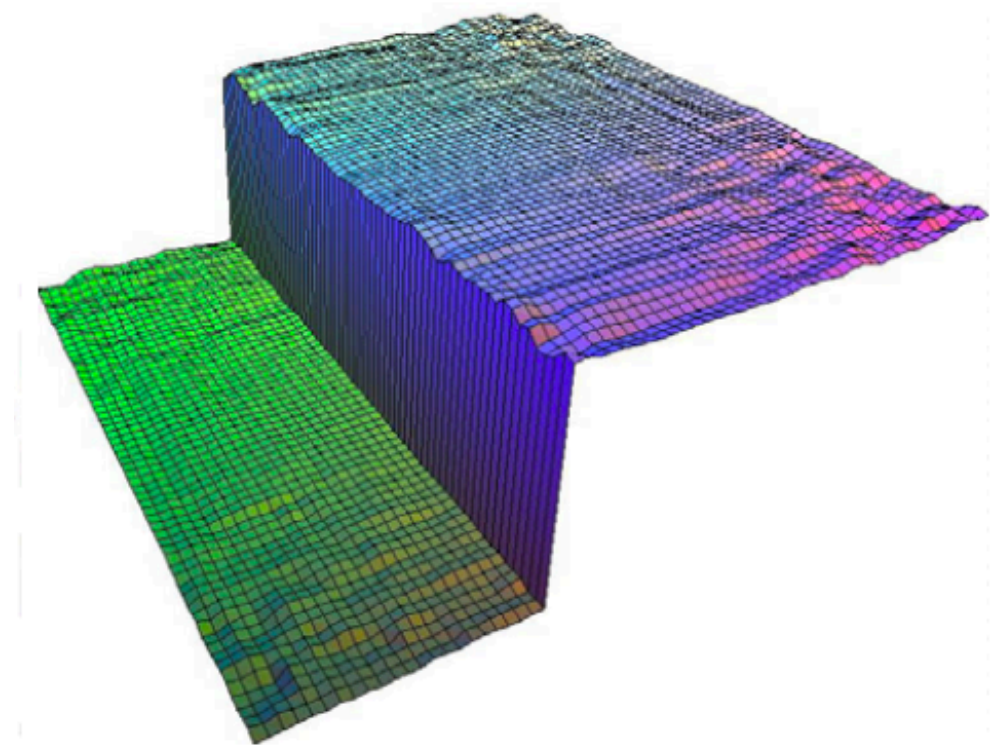
Domain Kernel



Range Kernel Influence



Bilateral Filter
(domain * range)



Output

Images from: Durand and Dorsey, 2002

Bilateral Filter Application: Denoising



Noisy Image



Gaussian Filter



Bilateral Filter

Bilateral Filter Application: Cartooning



Original Image



After 5 iterations of **Bilateral** Filter

Bilateral Filter Application: Flash Photography

Non-flash images taken under low light conditions often suffer from excessive **noise** and **blur**

But there are problems with **flash images**:

- colour is often unnatural
- there may be strong shadows or specularities

Idea: Combine flash and non-flash images to achieve better exposure and colour balance, and to reduce noise

Bilateral Filter Application: Flash Photography

System using 'joint' or 'cross' bilateral filtering:



Flash



No-Flash

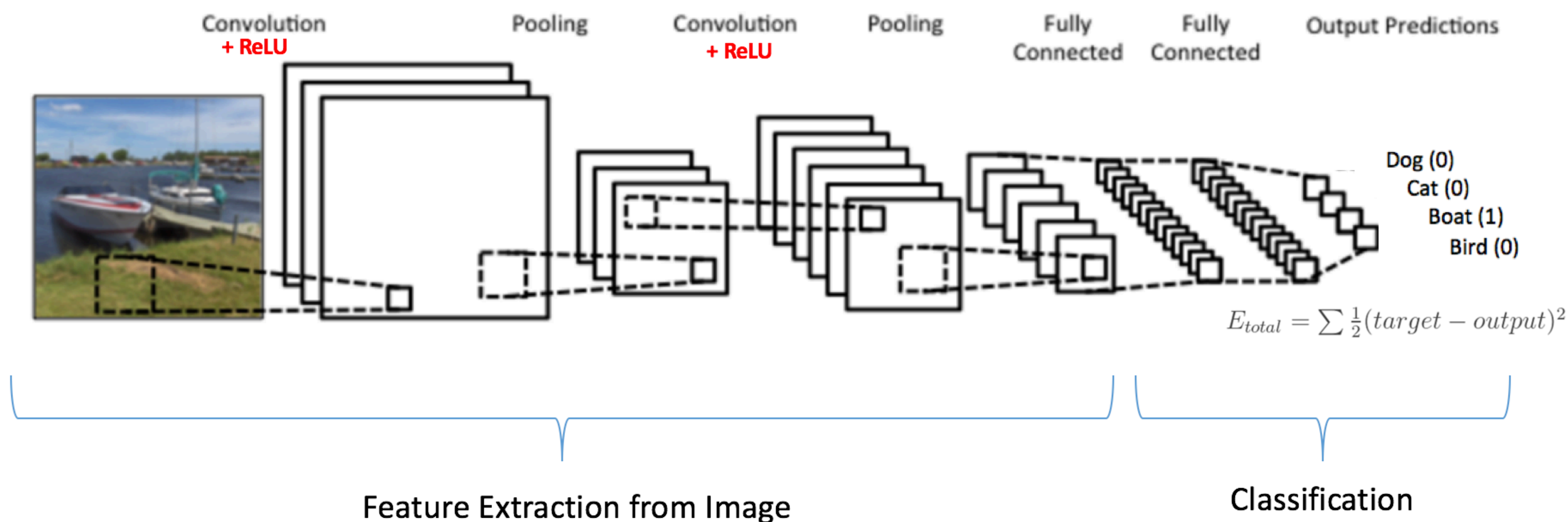


Detail Transfer with Denoising

'Joint' or 'Cross' bilateral: Range kernel is computed using a separate guidance image instead of the input image

Figure Credit: Petschnigg et al., 2004

Aside: Linear Filter with ReLU



9	3	5	-8
-6	2	-3	1
1	3	4	1
3	-4	5	1



9	3	5	0
0	2	0	1
1	3	4	1
3	0	5	1

Result of: Linear Image Filtering

After Non-linear ReLU

Summary

We covered two three **non-linear filters**: Median, Bilateral, ReLU

Separability (of a 2D filter) allows for more efficient implementation (as two 1D filters)

Convolution is **associative** and **symmetric**

Convolution of a Gaussian with a Gaussian is another Gaussian

The **median filter** is a non-linear filter that selects the median in the neighbourhood

The **bilateral filter** is a non-linear filter that considers both spatial distance and range (intensity) distance, and has edge-preserving properties