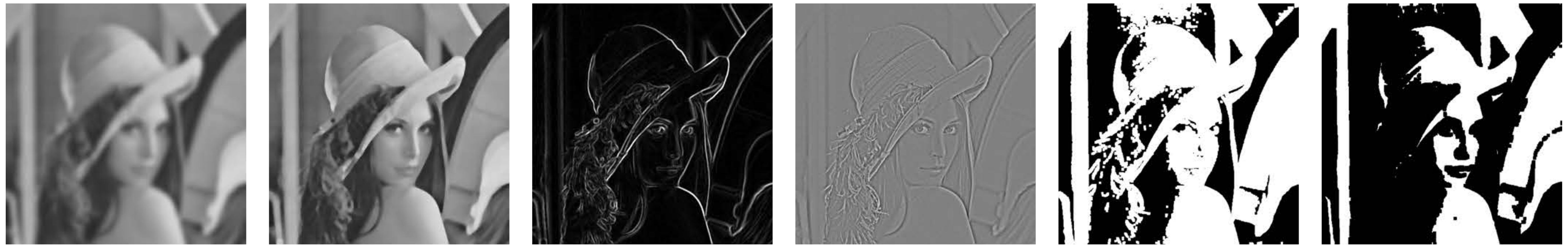
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 5: Image Filtering (final)

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

Menu for Today

Topics:

— **Linear Filtering** recap

— 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 **January 29th**

— **Quiz 1:** Due **tomorrow** (opens up after class), 15 min, 4 Questions

Linear Filter **Example**

$F(X,Y)$

filter

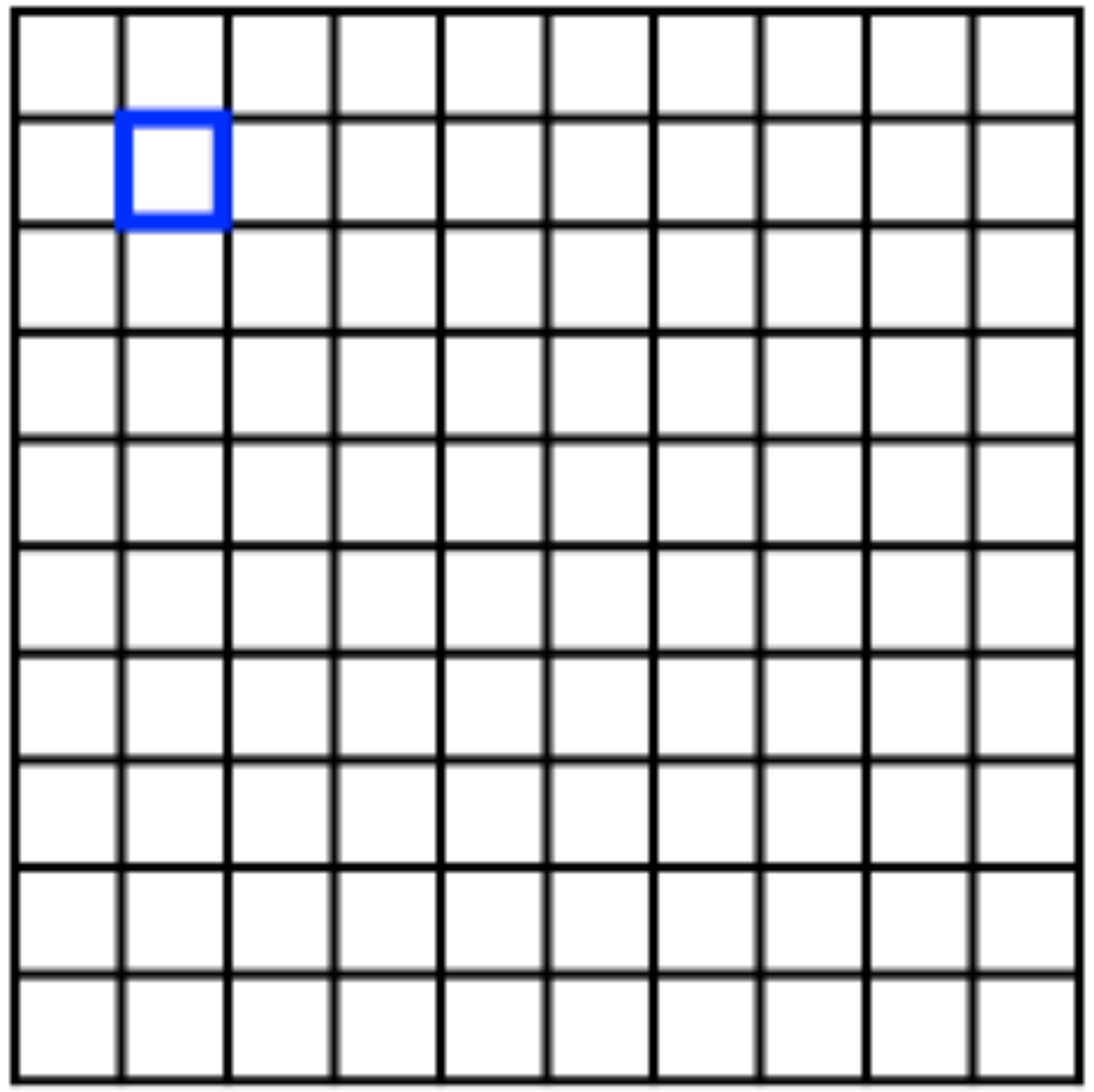
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

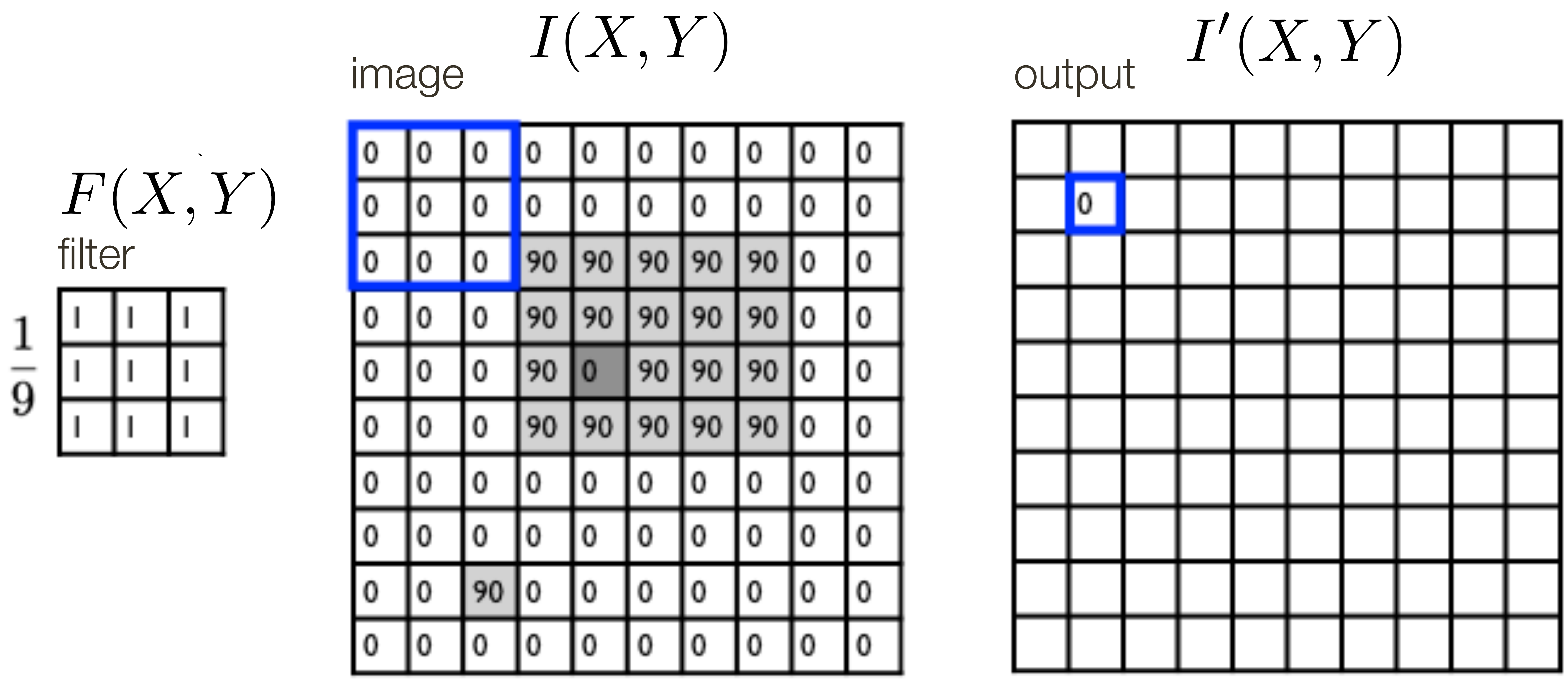


$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j) I(X+i, Y+j)$$

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**



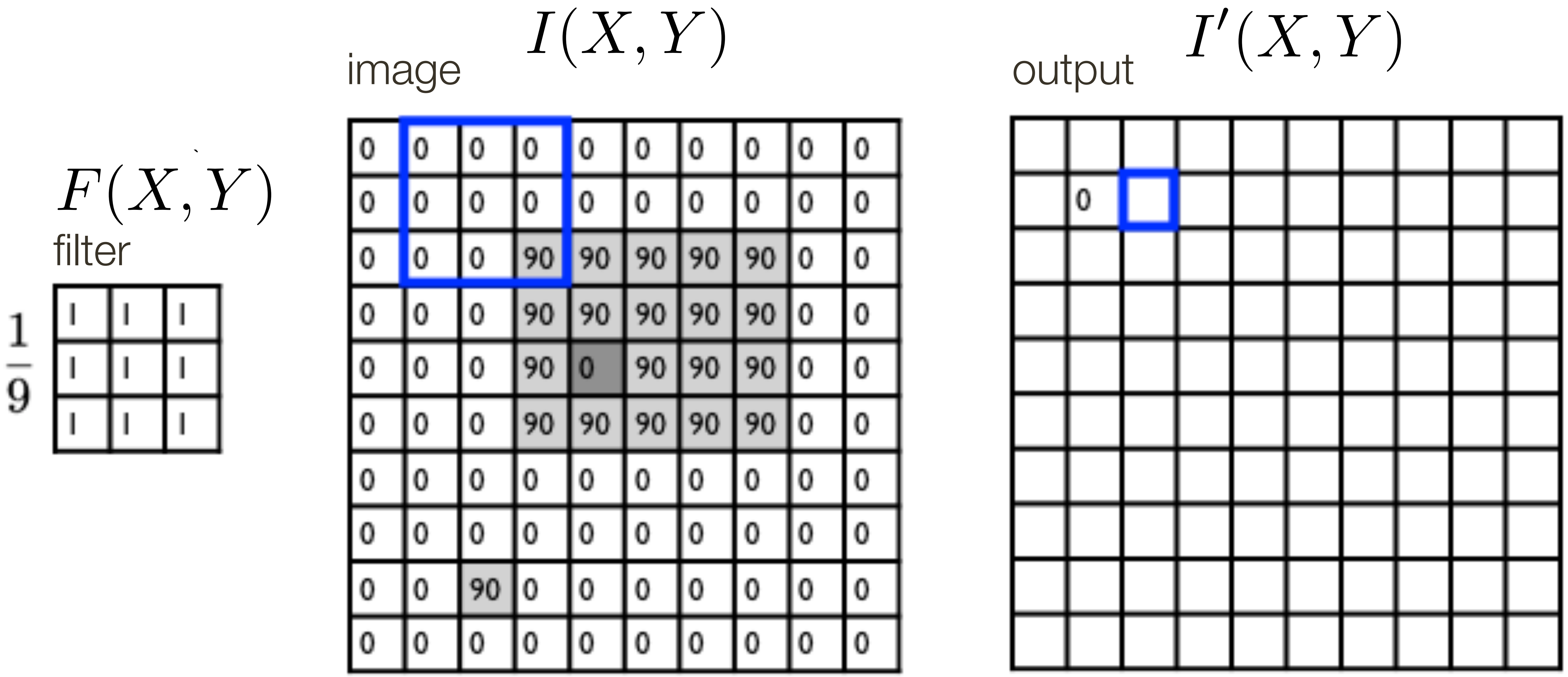
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

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

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**



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

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10							

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

output | filter | image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10							

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

output · filter · image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

filter $F(X,Y)$

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

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20						

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

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20						

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

output | filter | image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

filter $F(X,Y)$

$\frac{1}{9}$ ×

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

	0	10	20	30					

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

output — filter — image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filter **Example**

$F(X,Y)$

filter

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $I(X,Y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $I'(X,Y)$

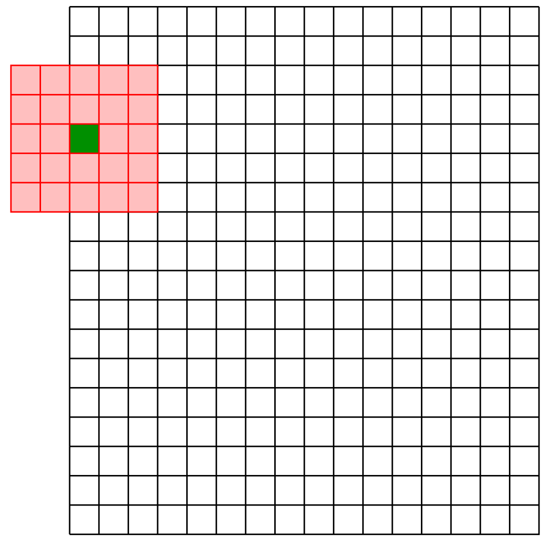
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	0	10	20	30	30	30	20	10	
	10	10	10	10	0	0	0	0	
	10	10	10	10	0	0	0	0	

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

output | filter | image (signal)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Linear Filters: **Boundary** Effects



Four standard ways to deal with boundaries:

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

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

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

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

Lecture 4: Re-cap

Linear filtering (one interpretation):
— new pixels are a weighted sum of original pixel values
— “filter” defines weights

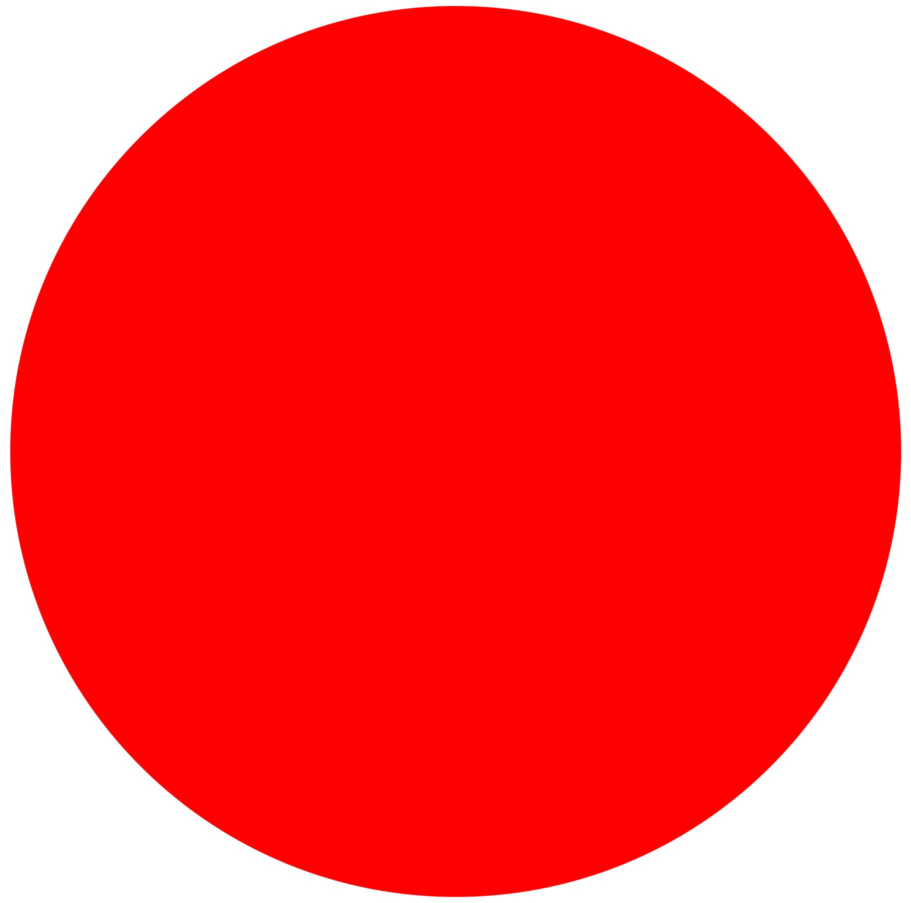
Linear filtering (another interpretation):
— each pixel creates a scaled copy of point spread function in its location
— “filter” specifies the point spread function

Low-pass Filtering = “Smoothing”

Box Filter

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

Pillbox Filter



Gaussian Filter

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

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

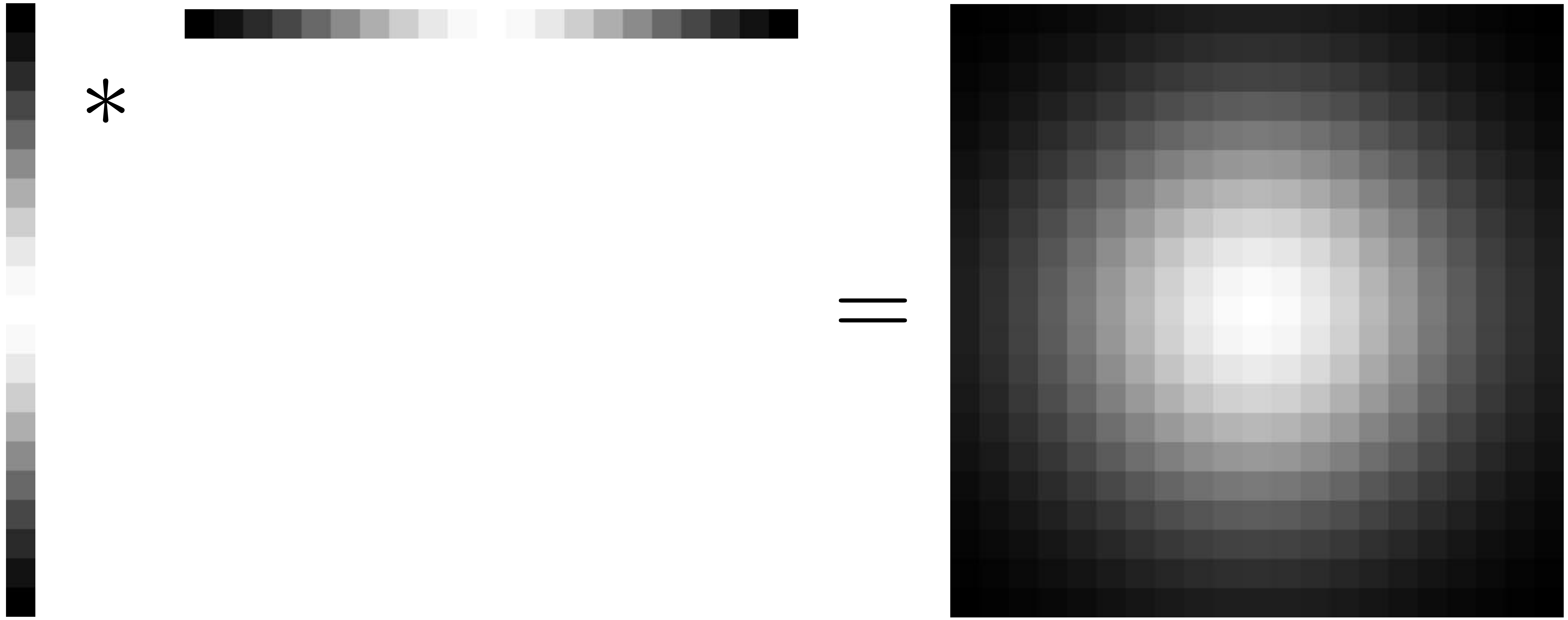
Low-pass filter: Low pass filter filters out all of the high frequency content of the image, only low frequencies remain

Example: Separable Filter

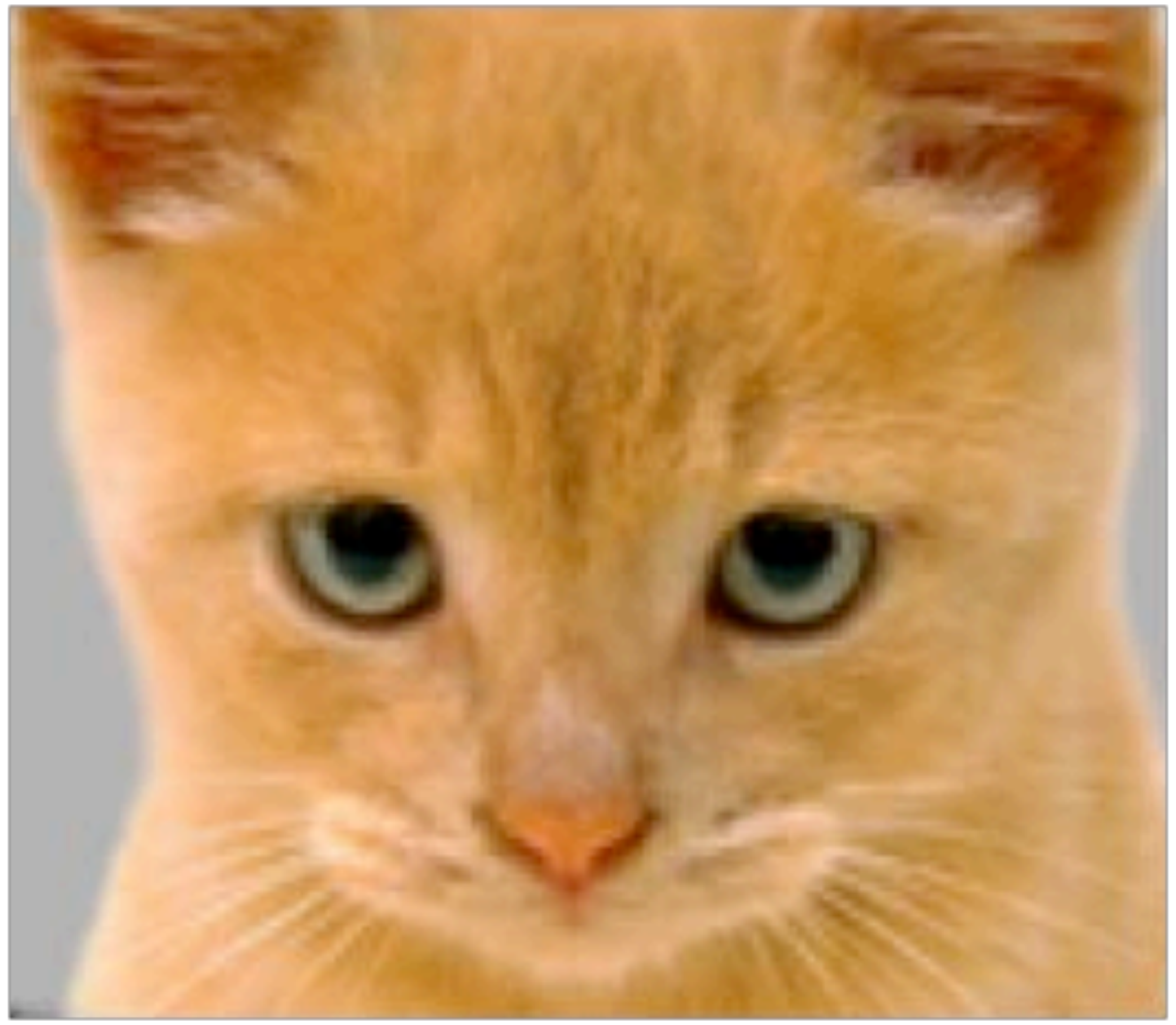
$$\frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \otimes \frac{1}{16}\begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Gaussian Blur

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

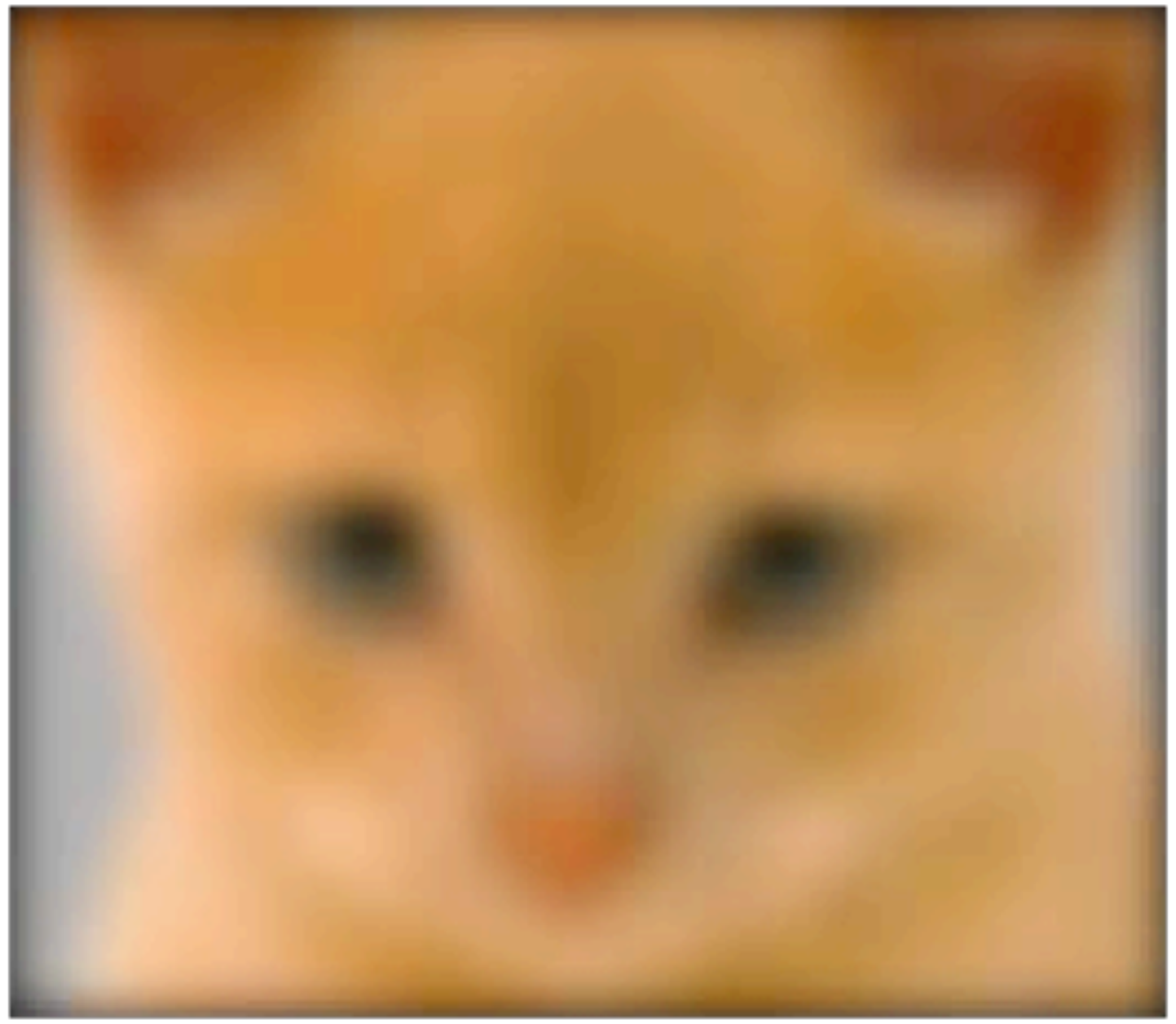


Assignment 1: **Low/High Pass** Filtering



Original

$$I(x,y)$$



Low-Pass Filter

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

High-Pass Filter

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

Point Spread Function

Optional subtitle

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

$*$

1	2	3
4	5	6
7	8	9

$=$

0	0	0	0	0	0	0	0
0	9	8	7	0	0	0	0
0	6	5	4	0	0	0	0
0	3	2	1	0	0	0	0
0	0	0	0	9	8	7	0
0	0	0	0	6	5	4	0
0	0	0	0	3	2	1	0
0	0	0	0	0	0	0	0

Point Spread Function

Optional subtitle

$$
\begin{bmatrix}
1 & 1 & 1 & 2 & 3 & 0 & 0 & 0 \\
1 & 1 & 1 & 2 & 3 & 0 & 0 & 0 \\
1 & 1 & 1 & 2 & 3 & 0 & 0 & 0 \\
4 & 4 & 4 & 5 & 6 & 0 & 0 & 0 \\
7 & 7 & 7 & 8 & 9 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix}
=
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 9 & 8 & 7 & 0 & 0 & 0 & 0 \\
0 & 6 & 5 & 4 & 0 & 0 & 0 & 0 \\
0 & 3 & 2 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 9 & 8 & 7 & 0 \\
0 & 0 & 0 & 0 & 6 & 5 & 4 & 0 \\
0 & 0 & 0 & 0 & 3 & 2 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Advanced Convolution Topics

- Multiple filters
- Fourier transforms

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**. (think of subtraction)

Symmetricity 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]]
```

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

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

$$conv(A, B) = conv(B, A)$$

$$corr(A, B) \neq corr(B, A)$$

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**. (think of subtraction)

Example: Two Box Filters

filter = boxfilter(3)

signal.correlate2d(filter, filter,′ full′)

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} \otimes \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} = \frac{1}{81}\begin{bmatrix}1 & 2 & 3 & 2 & 1\\2 & 4 & 6 & 4 & 2\\3 & 6 & 9 & 6 & 3\\2 & 4 & 6 & 4 & 2\\1 & 2 & 3 & 2 & 1\end{bmatrix}$$

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

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

$\frac{1}{9}$

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**

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

1	1	1
1	1	1
1	1	1

3x3 **Box**

$= \frac{1}{81}$

	1					

Output

Example: Two Box Filters

Treat one filter as padded "image"

$$
\frac{1}{9}\begin{bmatrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{bmatrix} \otimes \frac{1}{9}\begin{bmatrix} 1&1&1\\ 1&1&1\\ 1&1&1 \end{bmatrix} = \frac{1}{81}\begin{bmatrix} &&&&&&\\ &1&2&&&&\\ &&&&&&\\ &&&&&&\\ &&&&&&\\ &&&&&&\\ &&&&&& \end{bmatrix}
$$

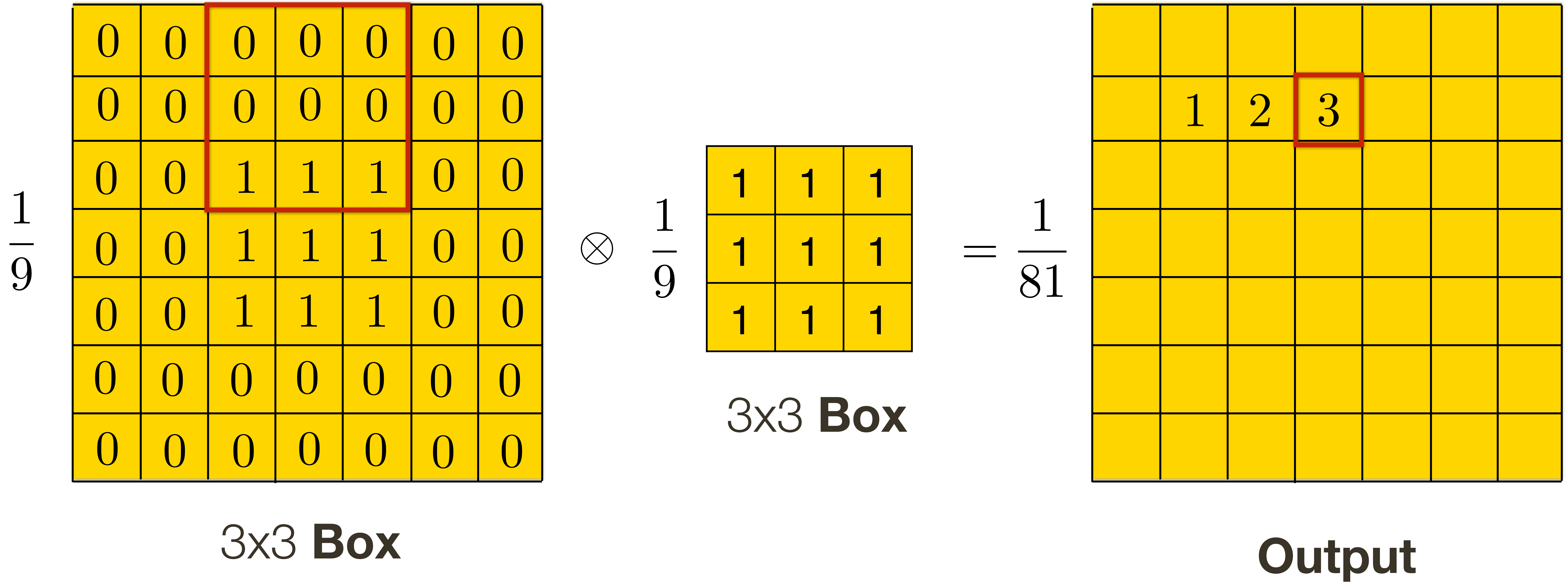
3x3 **Box**

3x3 **Box**

Output

Example: Two Box Filters

Treat one filter as padded "image"



Example: Two Box Filters

Treat one filter as padded "image"

$$\frac{1}{9}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \otimes \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{81}\begin{bmatrix} & & & & & & \\ & 1 & 2 & 3 & 2 & 1 & \\ & 2 & 4 & 6 & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{bmatrix}$$

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

Example: Two Box Filters

Treat one filter as padded “image”

$$\frac{1}{9}\begin{bmatrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&1&1&1&0&0\\0&0&1&1&1&0&0\\0&0&1&1&1&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix} \otimes \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} = \frac{1}{81}\begin{bmatrix} & & & & & & \\ &1&2&3&2&1& \\ &2&4&6&4&2& \\ &3&6&9&6&3& \\ &2&4&6&4&2& \\ &1&2&3&2&1& \\ & & & & & & \end{bmatrix}$$

3x3 **Box** ⊗ 3x3 **Box** = **Output**

Example: Two Box Filters

Treat one filter as padded "image"

$$\frac{1}{9}\begin{bmatrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{bmatrix} \otimes \frac{1}{9}\begin{bmatrix} 1&1&1\\ 1&1&1\\ 1&1&1 \end{bmatrix} = \frac{1}{81}\begin{bmatrix} 1&2&3&2&1\\ 2&4&6&4&2\\ 3&6&9&6&3\\ 2&4&6&4&2\\ 1&2&3&2&1 \end{bmatrix}$$

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**

5.1

Example: Separable Gaussian Filter

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

Example: Separable Gaussian Filter

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

Example: Separable Gaussian Filter

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

Example: Separable Gaussian Filter

$$
\frac{1}{16}
\begin{array}{|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 \\ \hline
1 & 4 & 6 & 4 & 1 \\ \hline
0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 \\ \hline
\end{array}
\otimes
\frac{1}{16}
\begin{array}{|c|}
\hline
1 \\ \hline
4 \\ \hline
6 \\ \hline
4 \\ \hline
1 \\ \hline
\end{array}
=
\frac{1}{256}
\begin{array}{|c|c|c|c|c|}
\hline
 & & & & \\ \hline
 & & & & \\ \hline
1 & 4 & 6 & 4 & 1 \\ \hline
4 & 16 & 24 & 16 & 4 \\ \hline
6 & 24 & 36 & 24 & 6 \\ \hline
4 & 16 & 24 & 16 & 4 \\ \hline
1 & 4 & 6 & 4 & 1 \\ \hline
 & & & & \\ \hline
 & & & & \\ \hline
\end{array}
$$

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-Convolving 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 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: Convolving with $G_\sigma(x)$ twice is equivalent to $G_{\sqrt{2}\sigma}(x)$

What follows is for fun

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

Convolution using **Fourier Transforms**

[Szeliski 3.4]

Convolution **Theorem**:

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

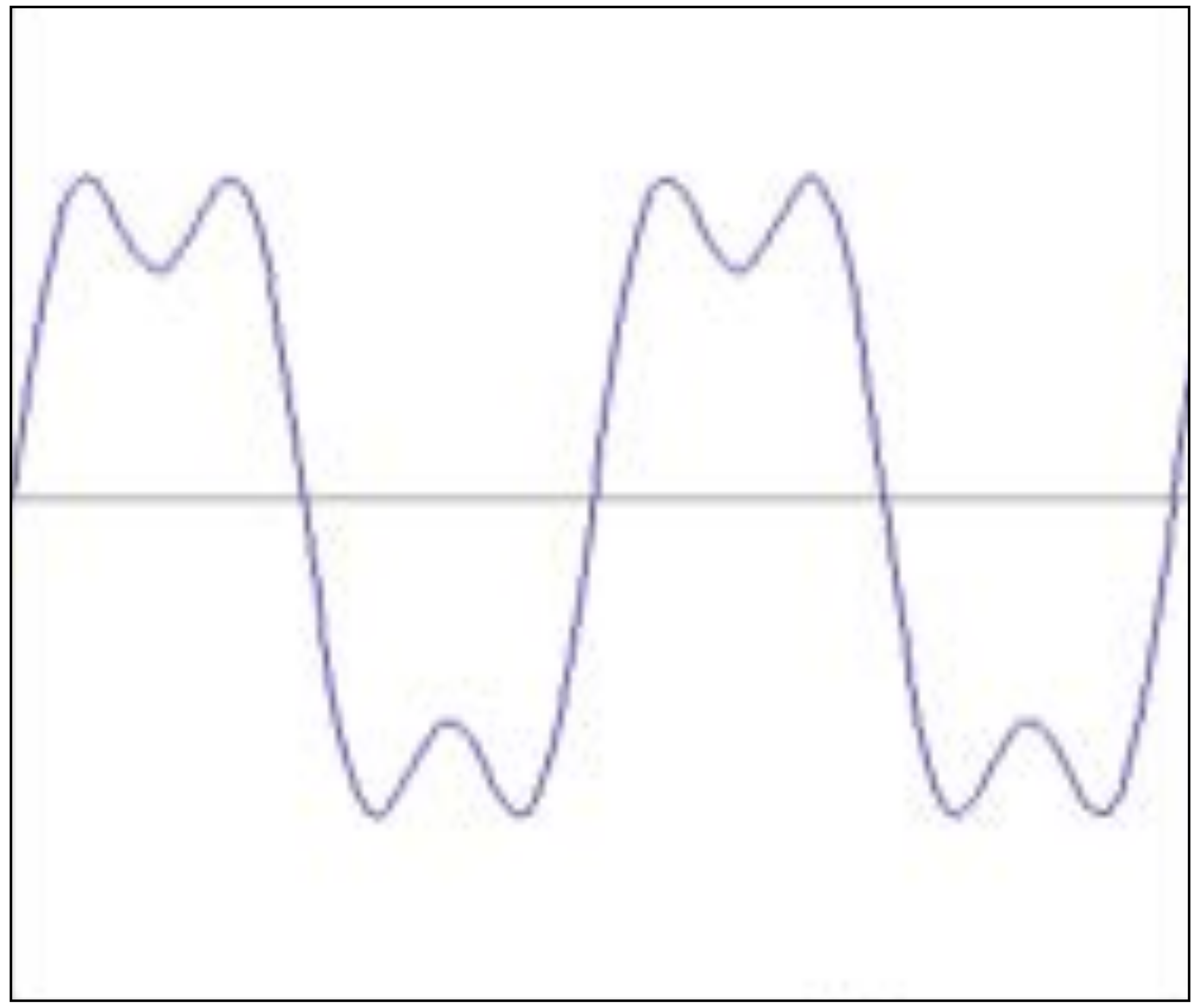
then $\mathcal{I}'(w_x, w_y) = \mathcal{F}(w_x, w_y)\, \mathcal{I}(w_x, w_y)$

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

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

How would you generate this function?

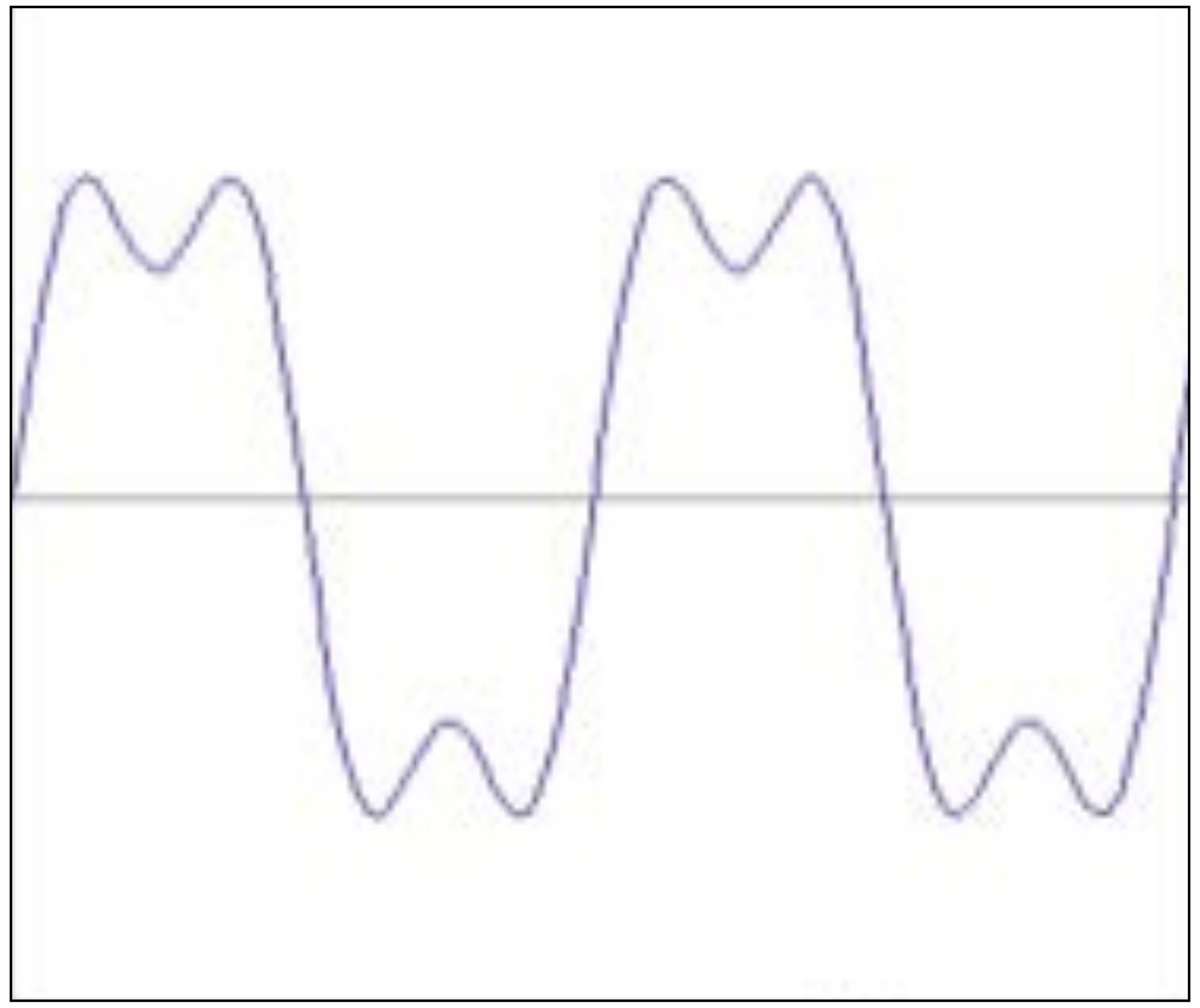


= ? + ?

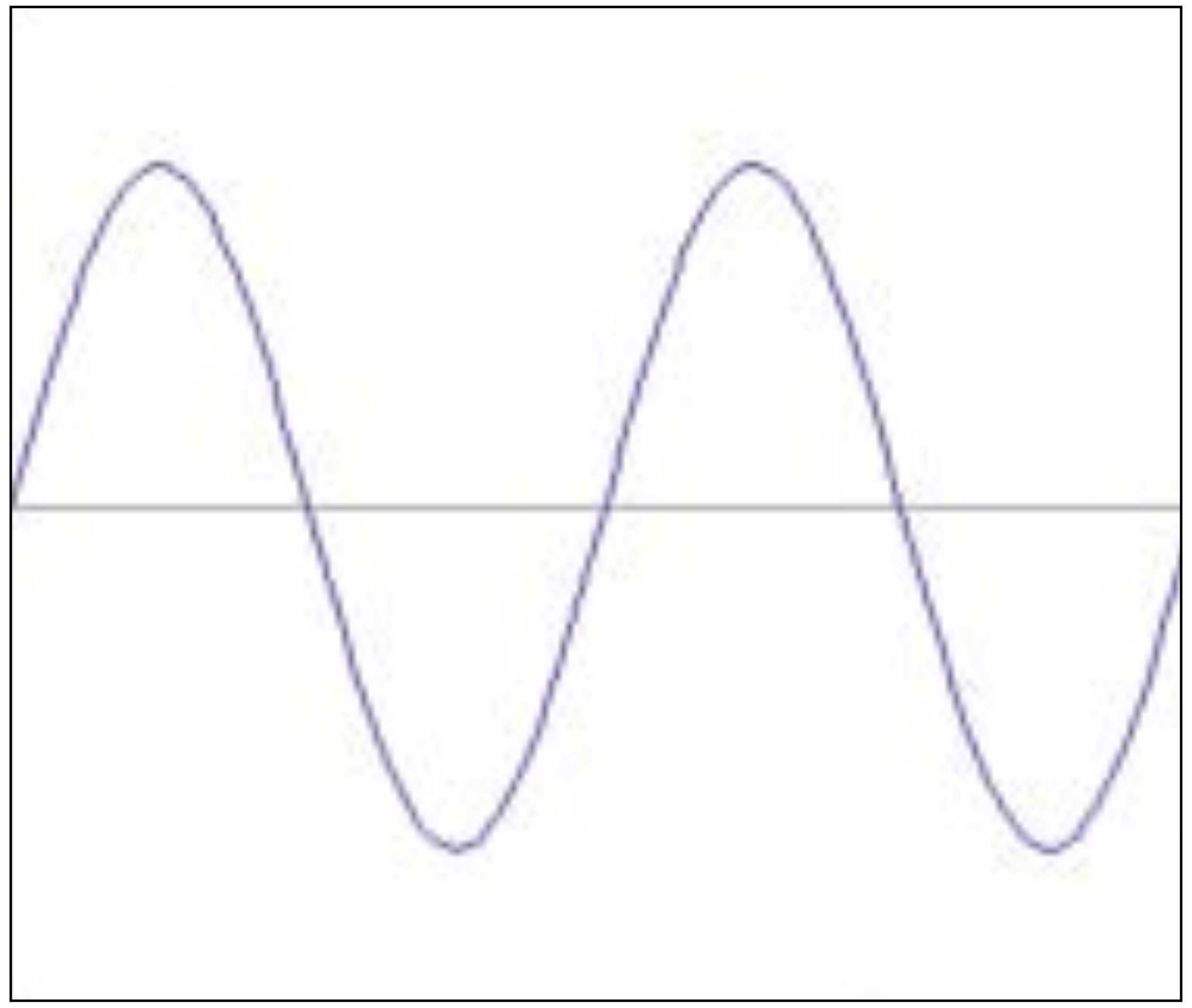
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

How would you generate this function?



=



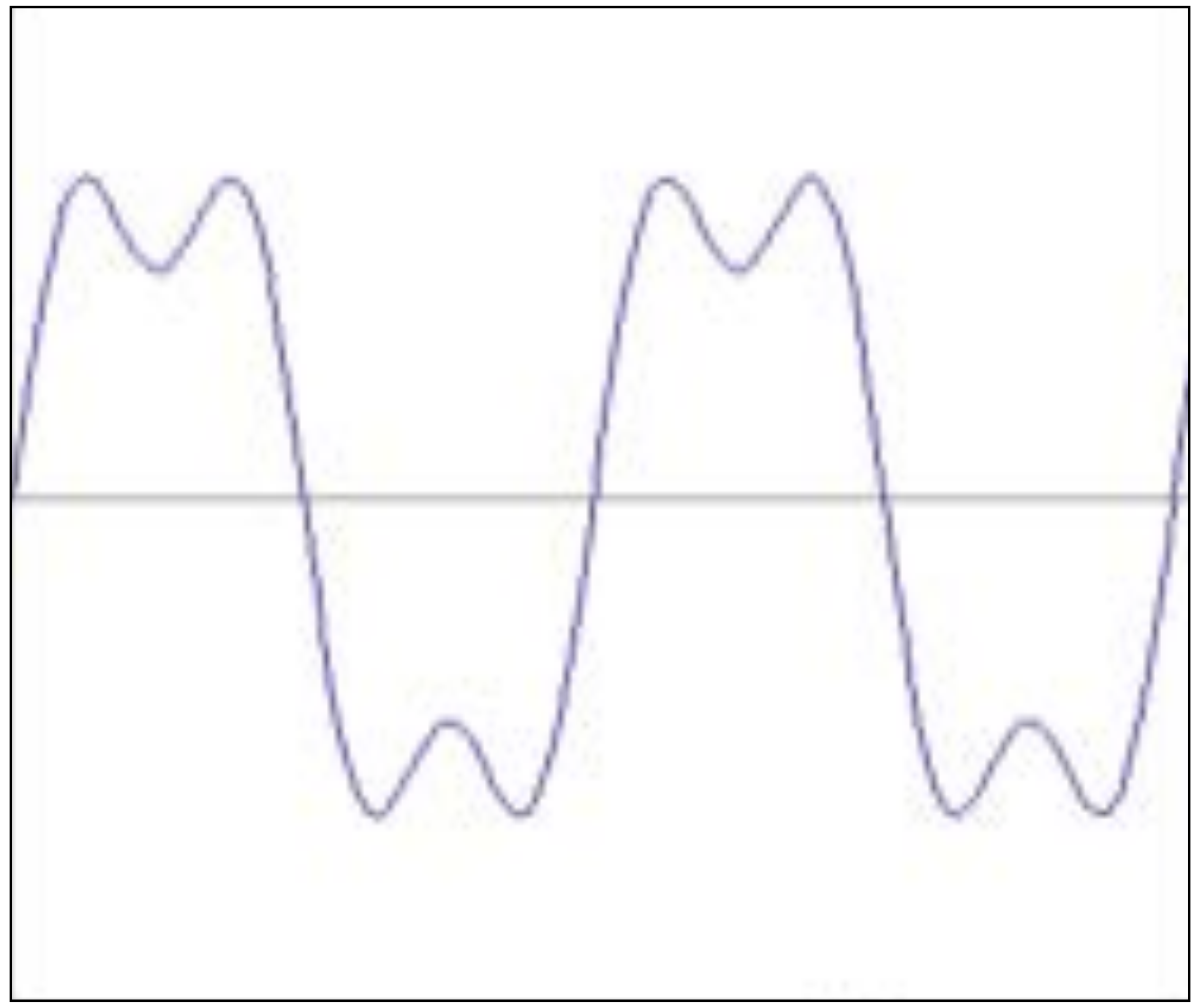
$\sin(2\pi x)$

\+ ?

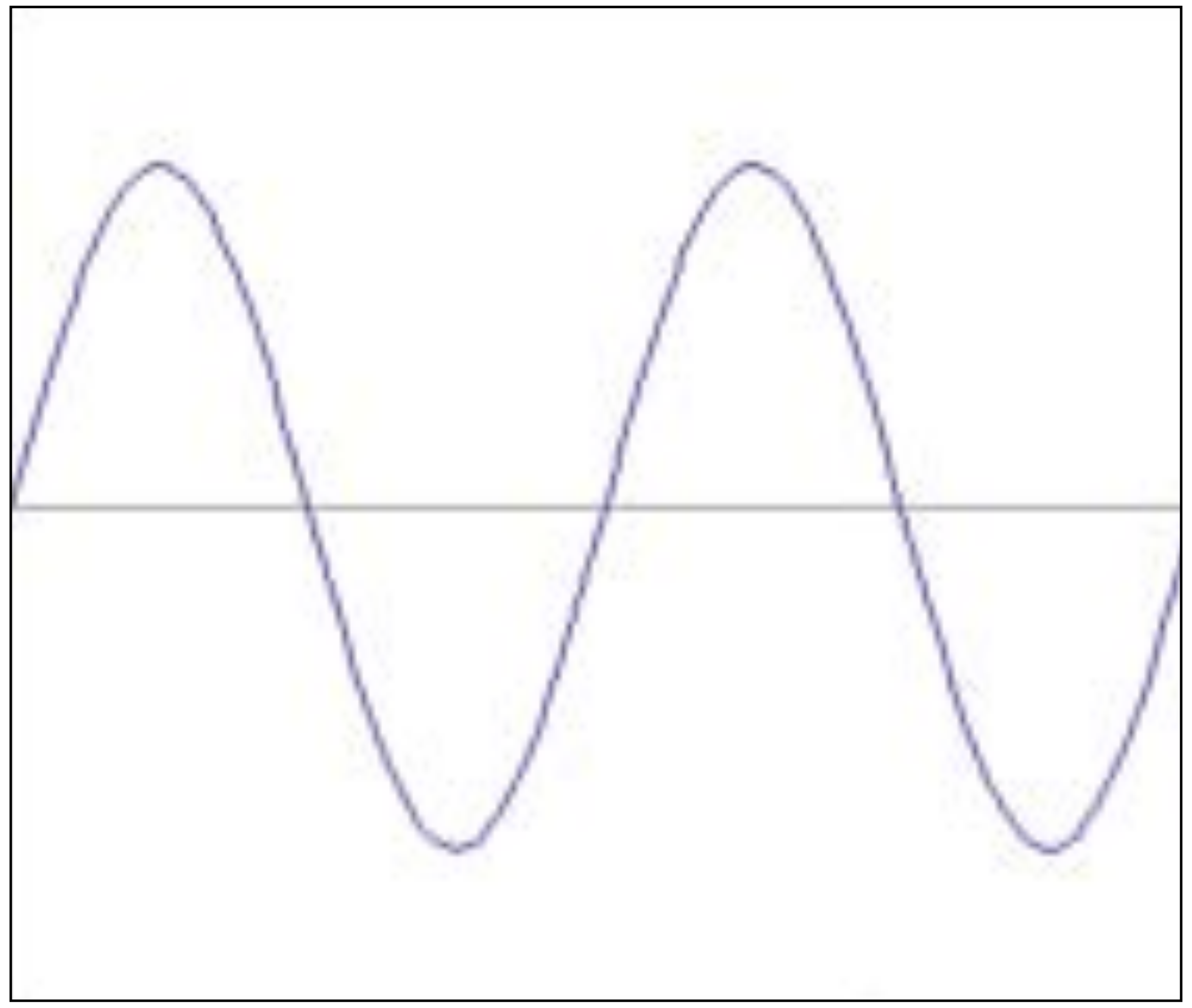
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

How would you generate this function?

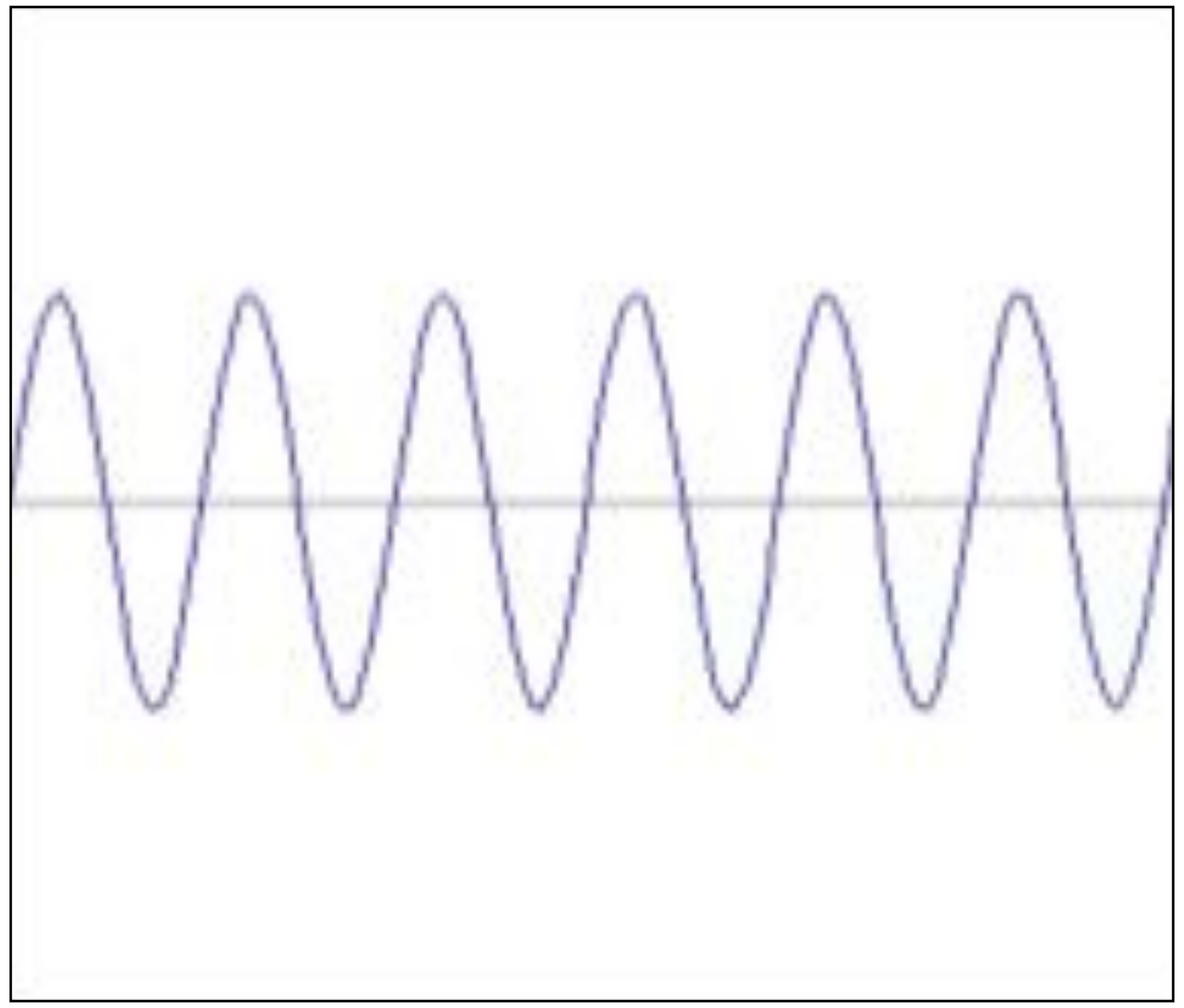


=



$\sin(2\pi x)$

+

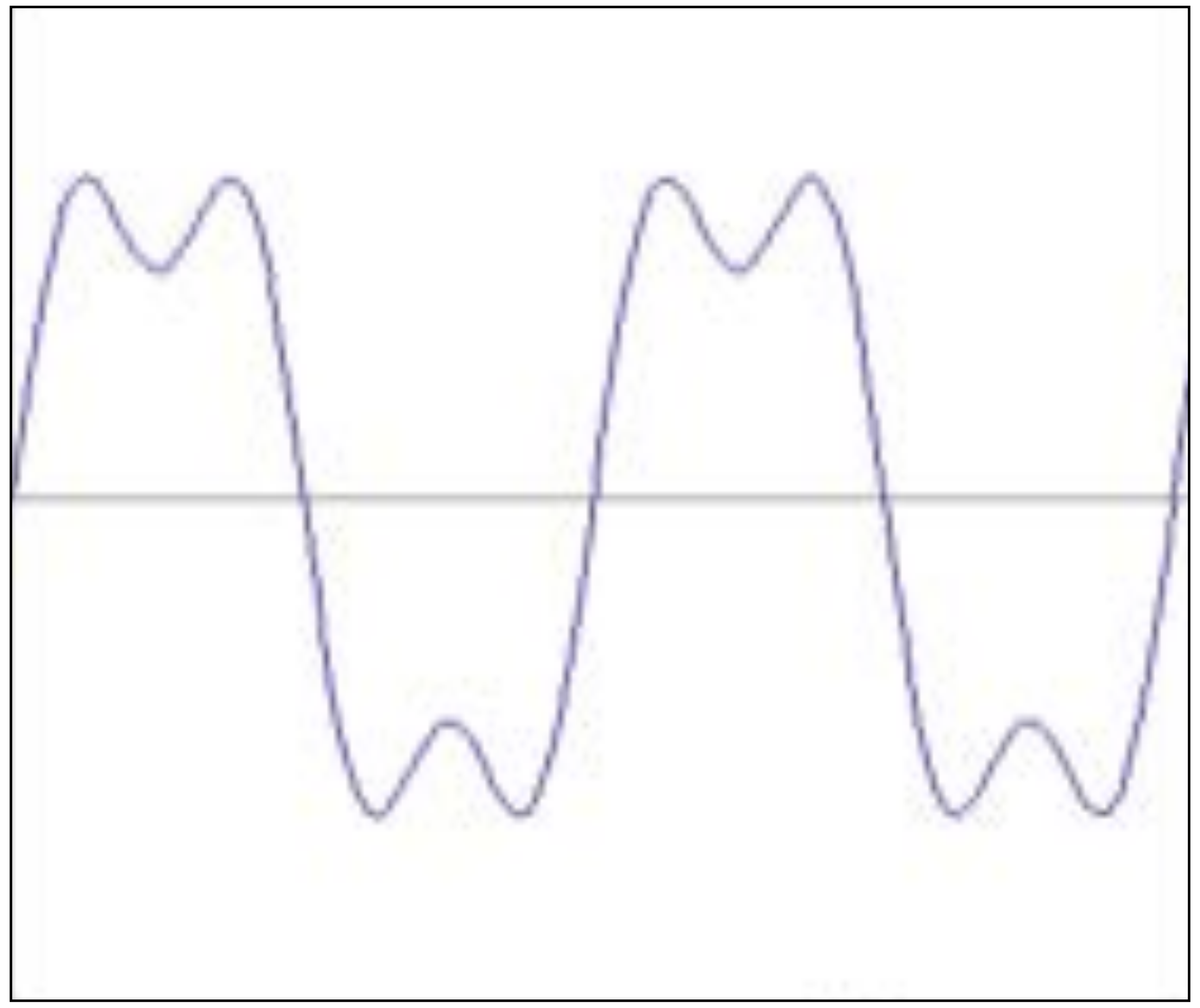


$\frac{1}{3}\sin(2\pi 3x)$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

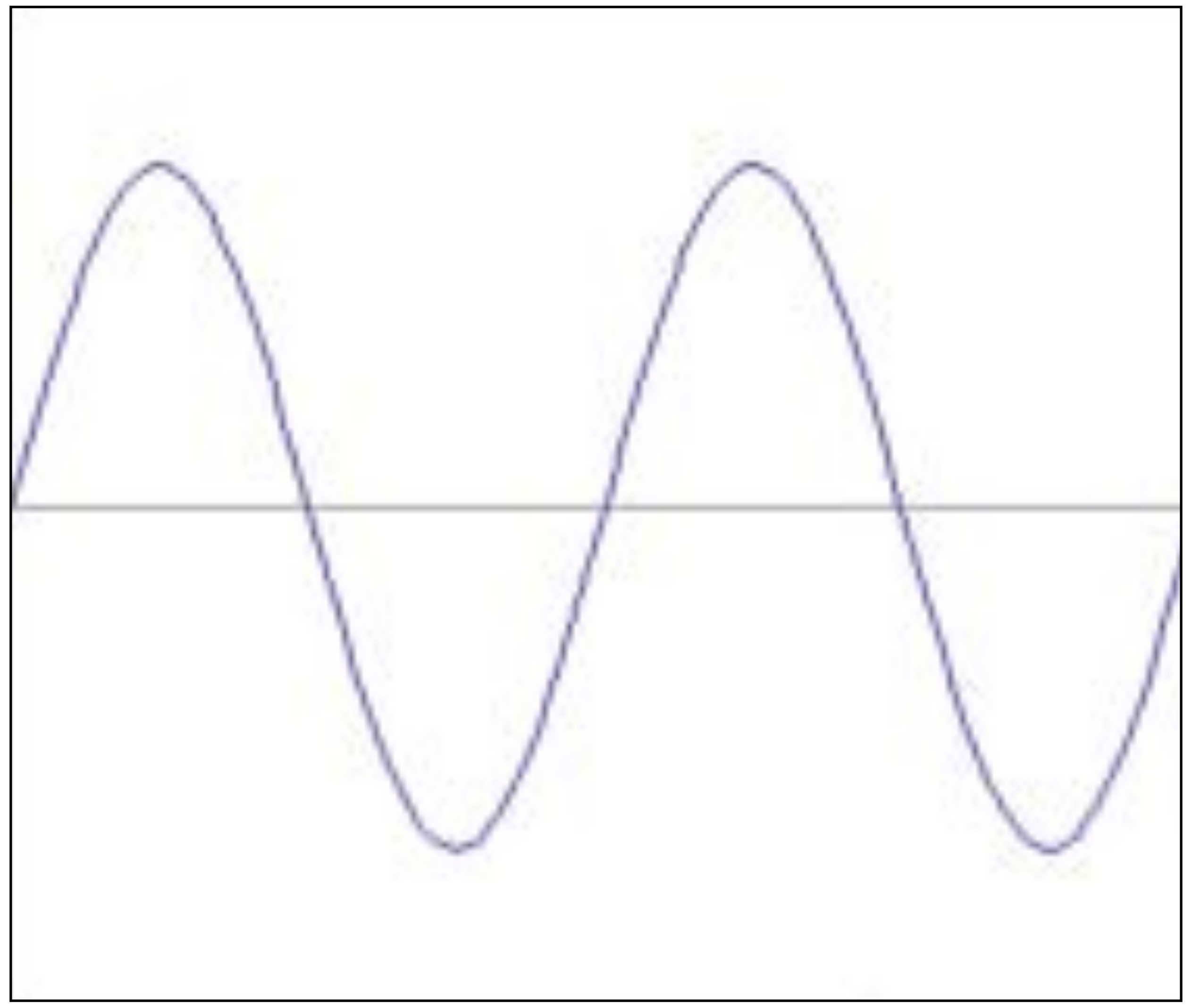
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?



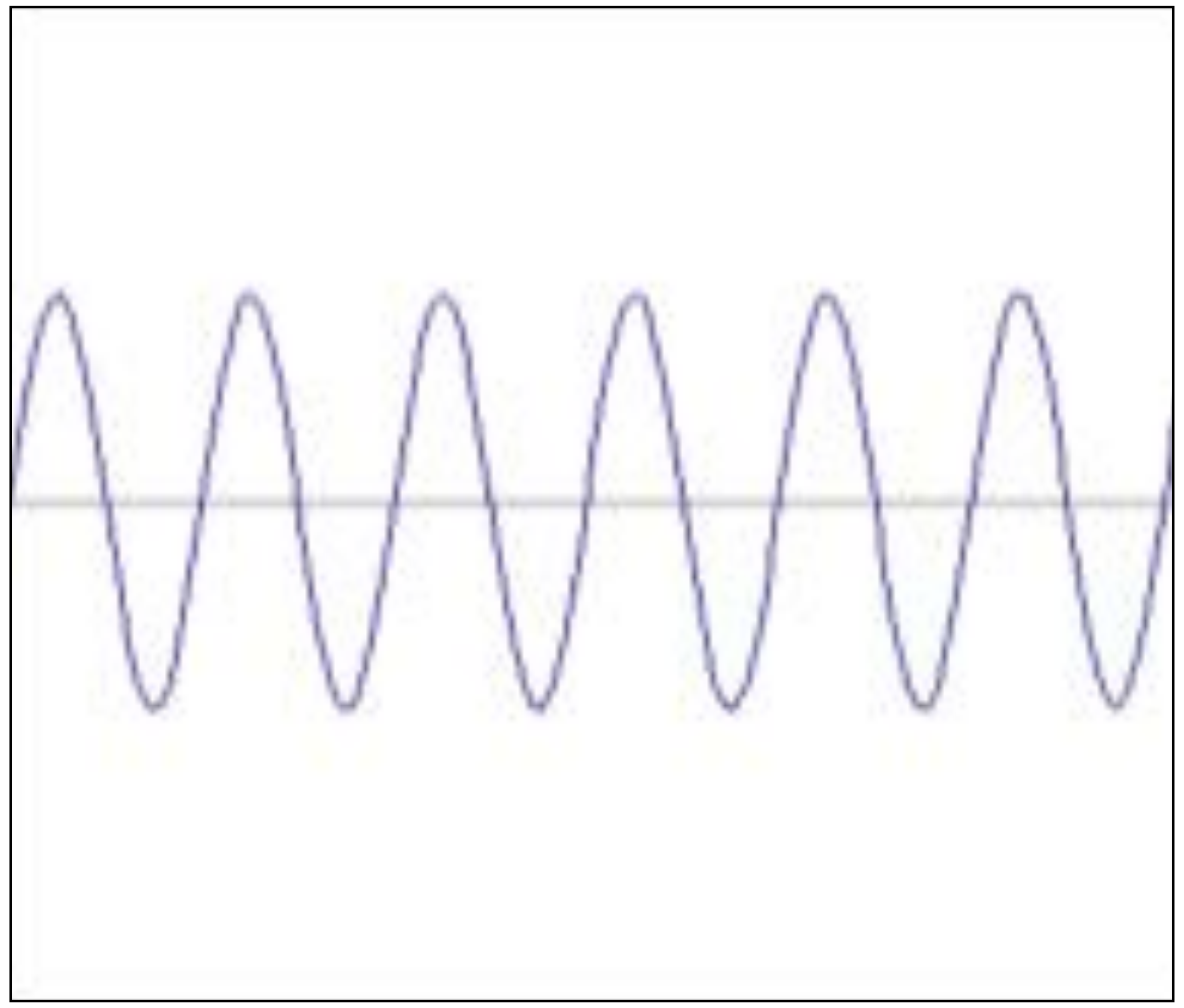
$$f(x) = \sin(2\pi x) + \frac{1}{3}\sin(2\pi 3x)$$

=



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

+

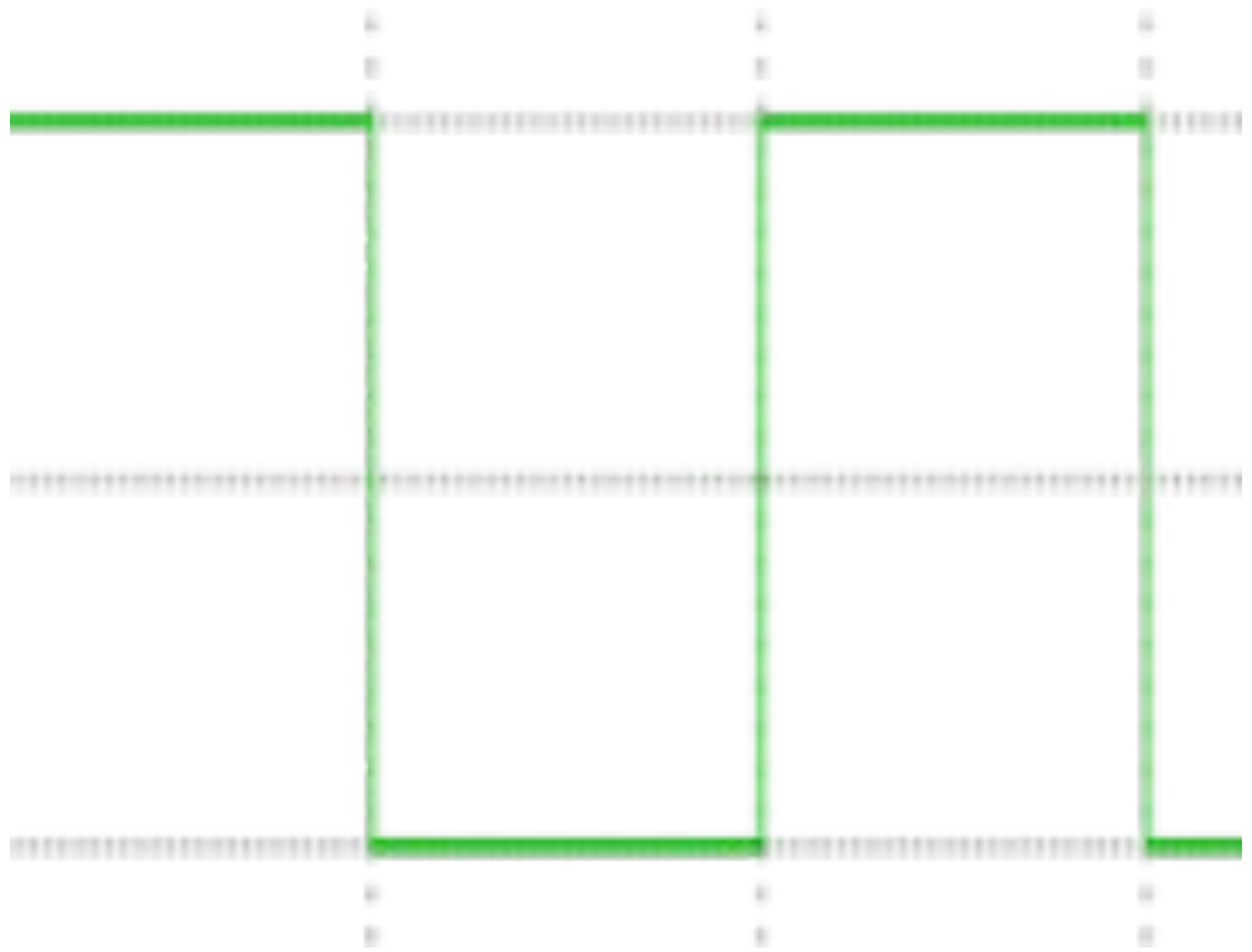


$$\frac{1}{3}\sin(2\pi 3x)$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

How would you generate this function?



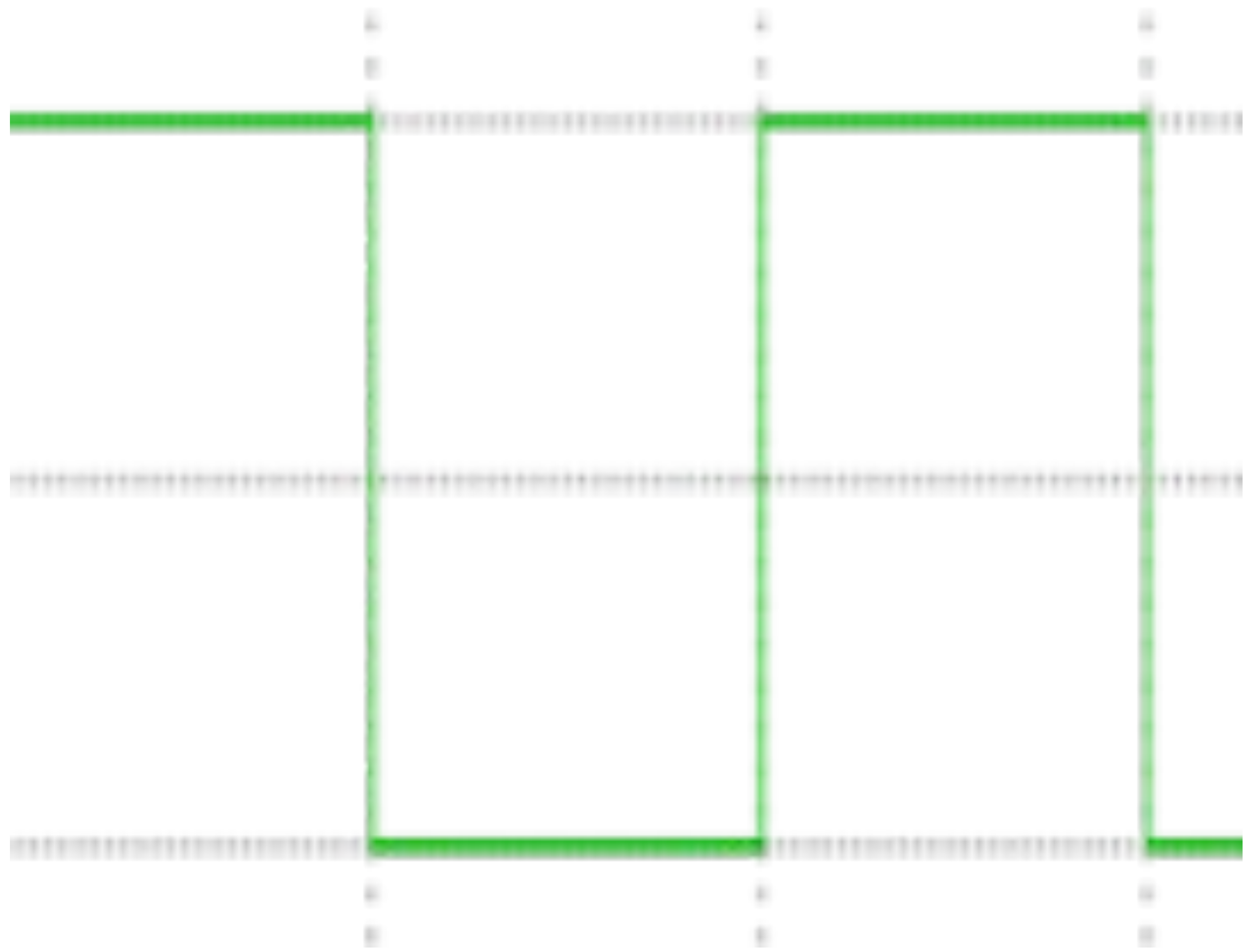
square wave

$\approx$? + ?

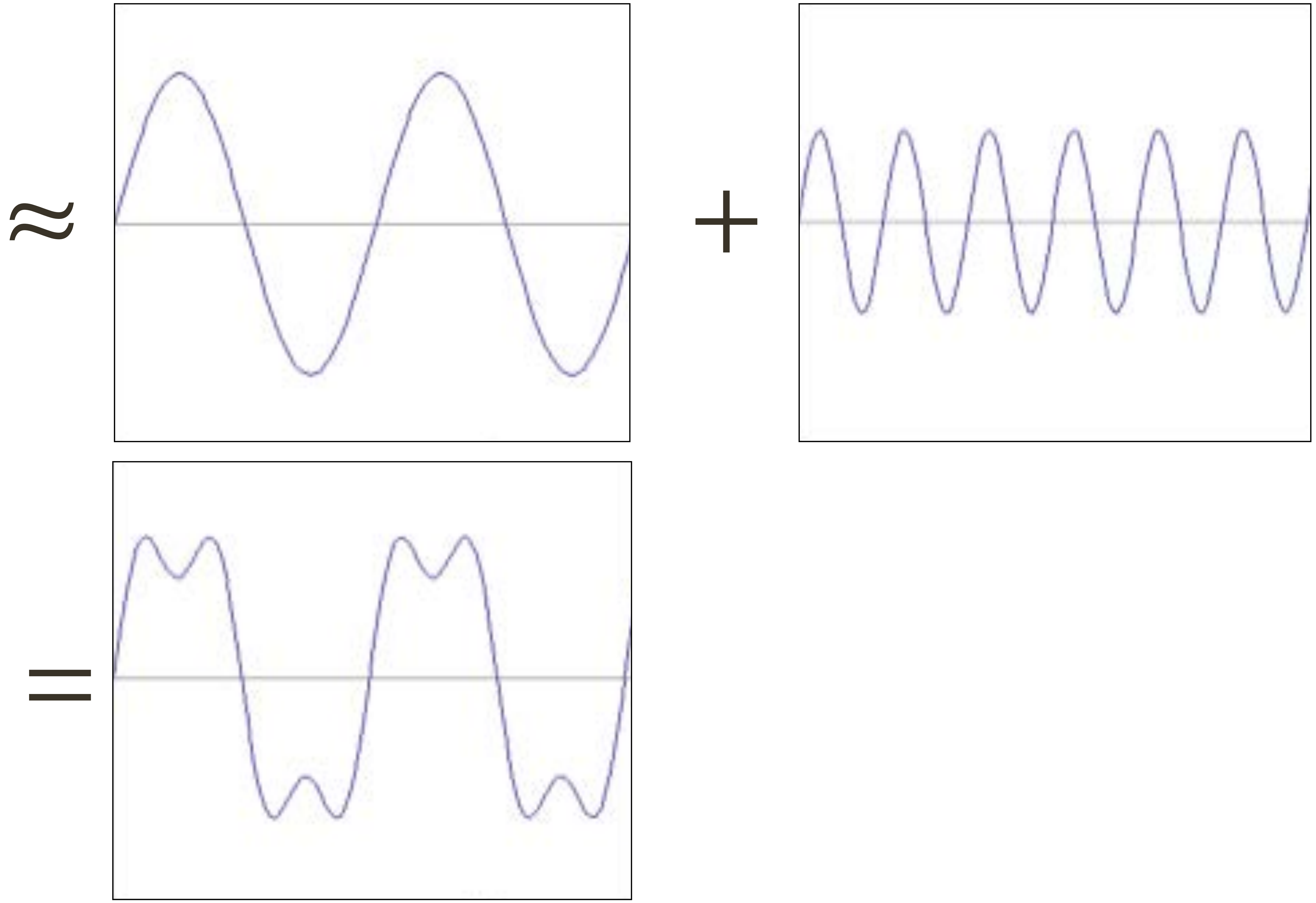
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

How would you generate this function?



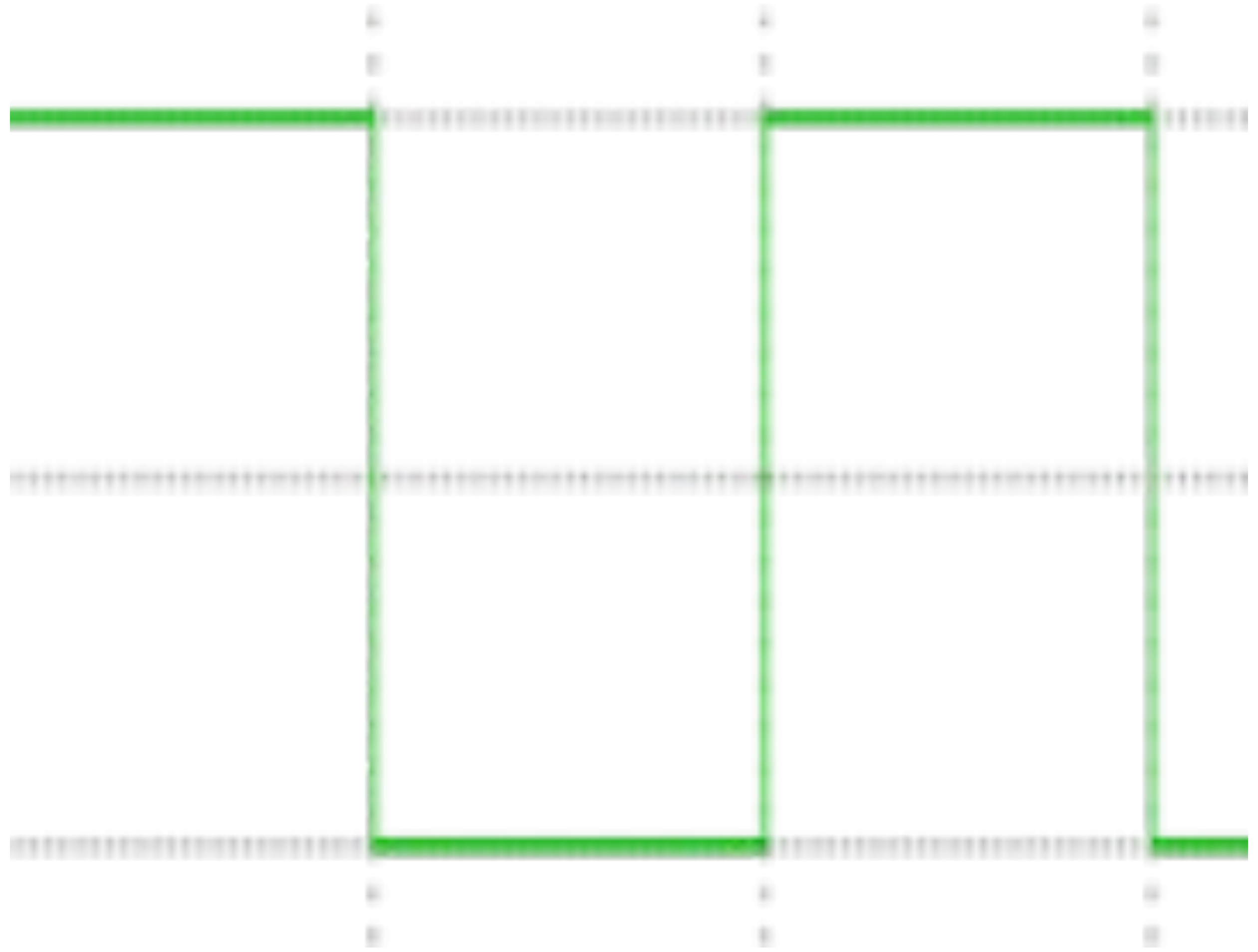
square wave



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

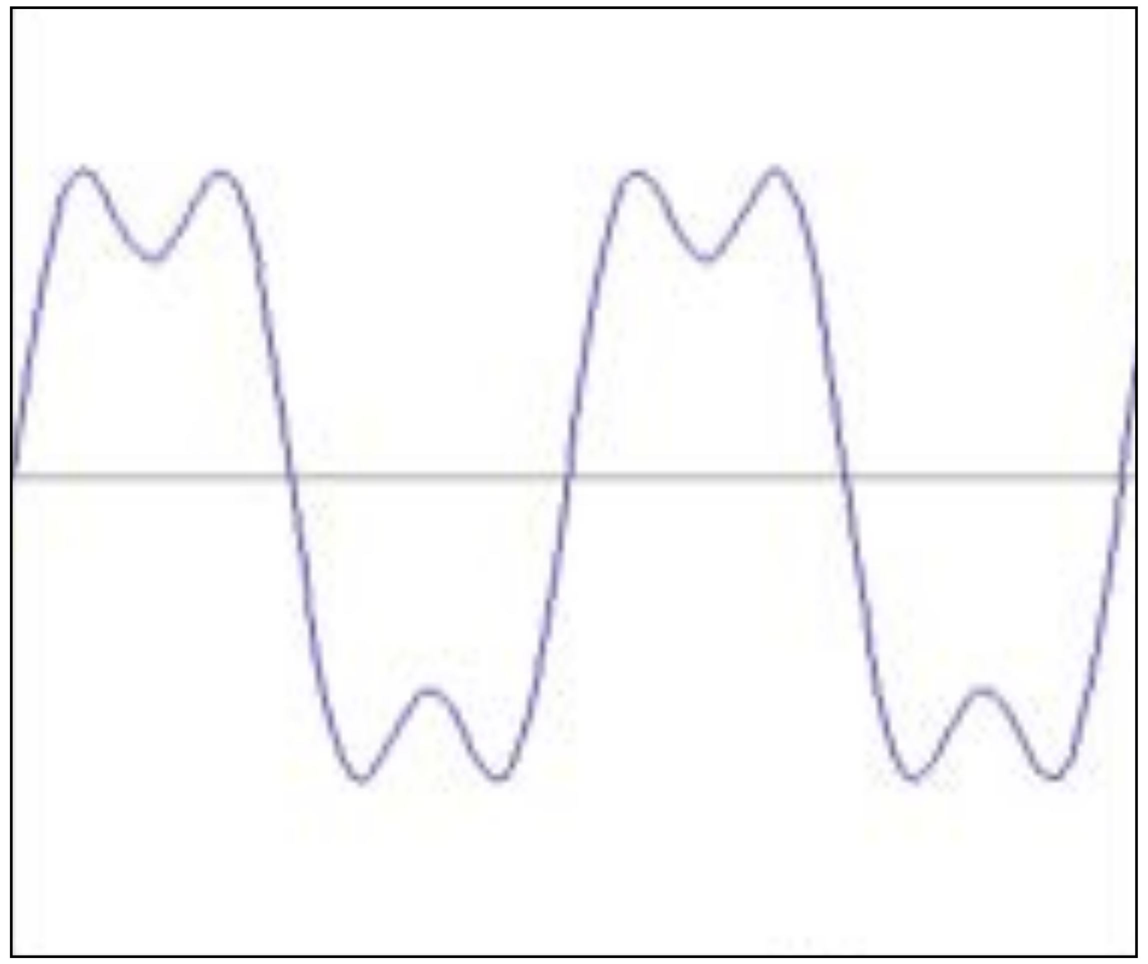
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

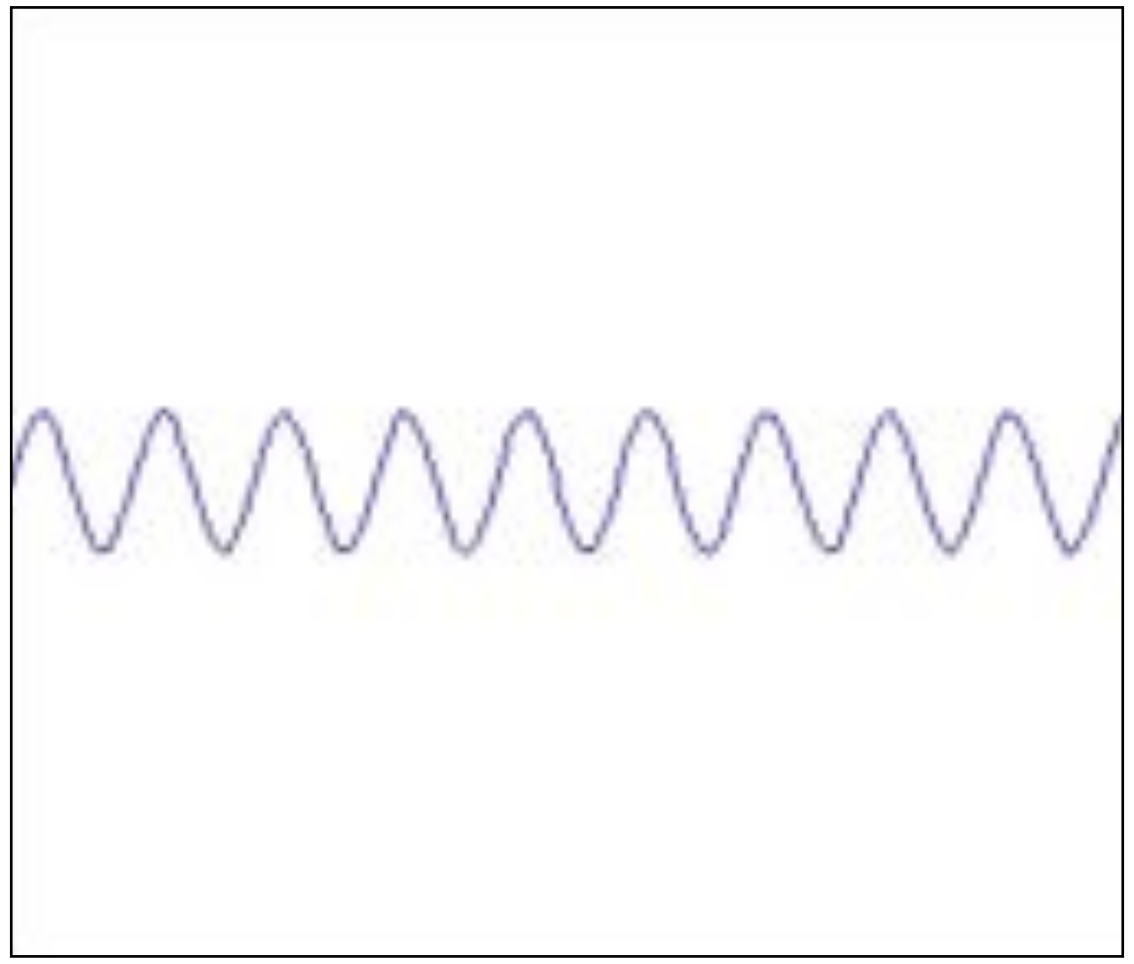


square wave

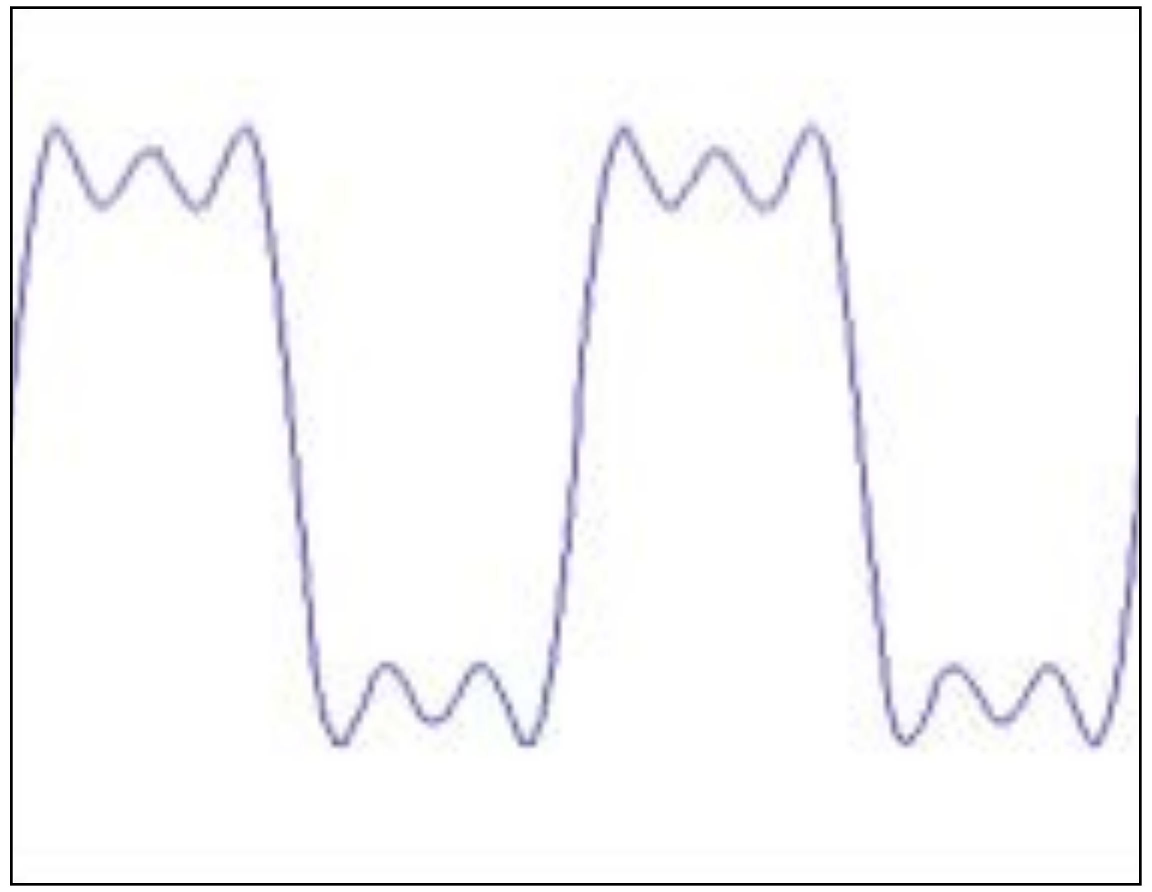
≈



+



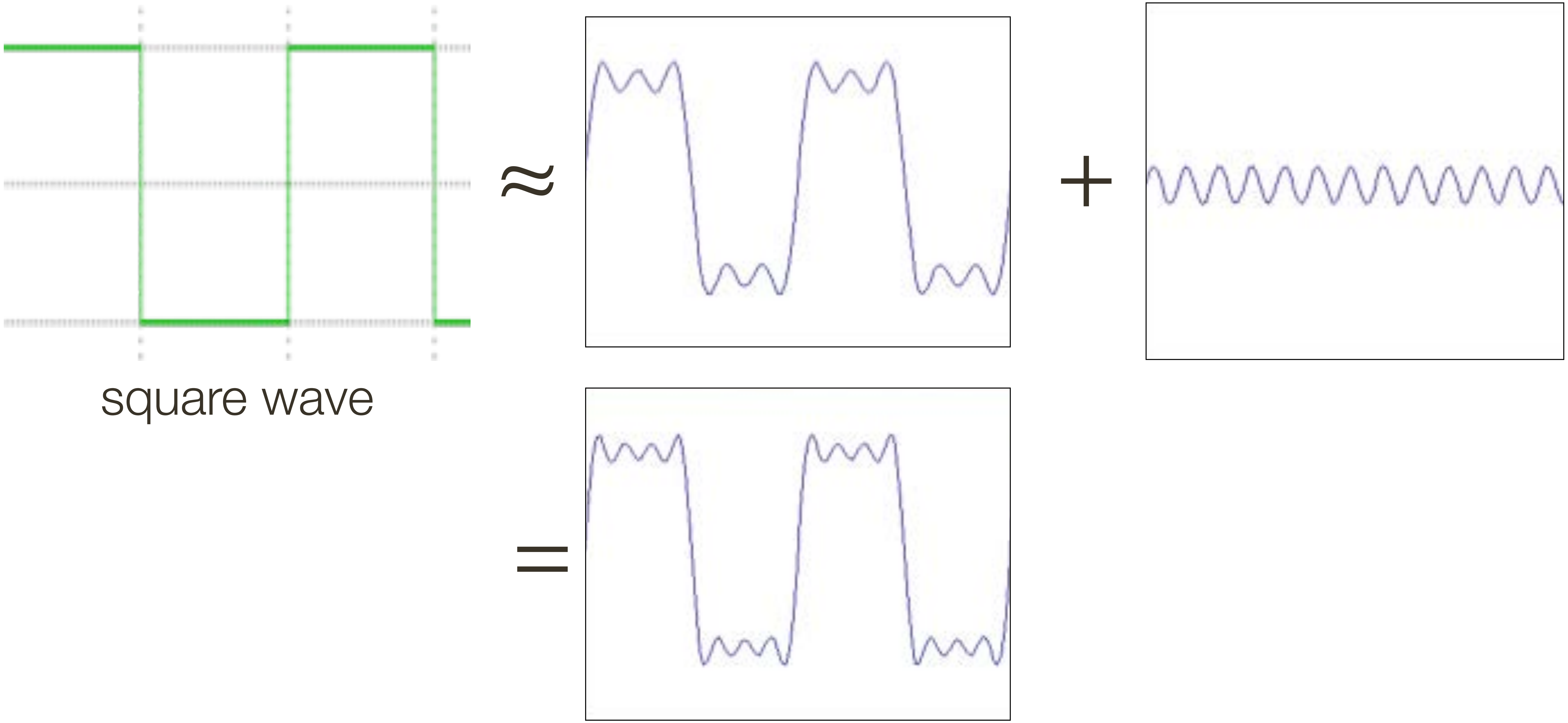
=



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

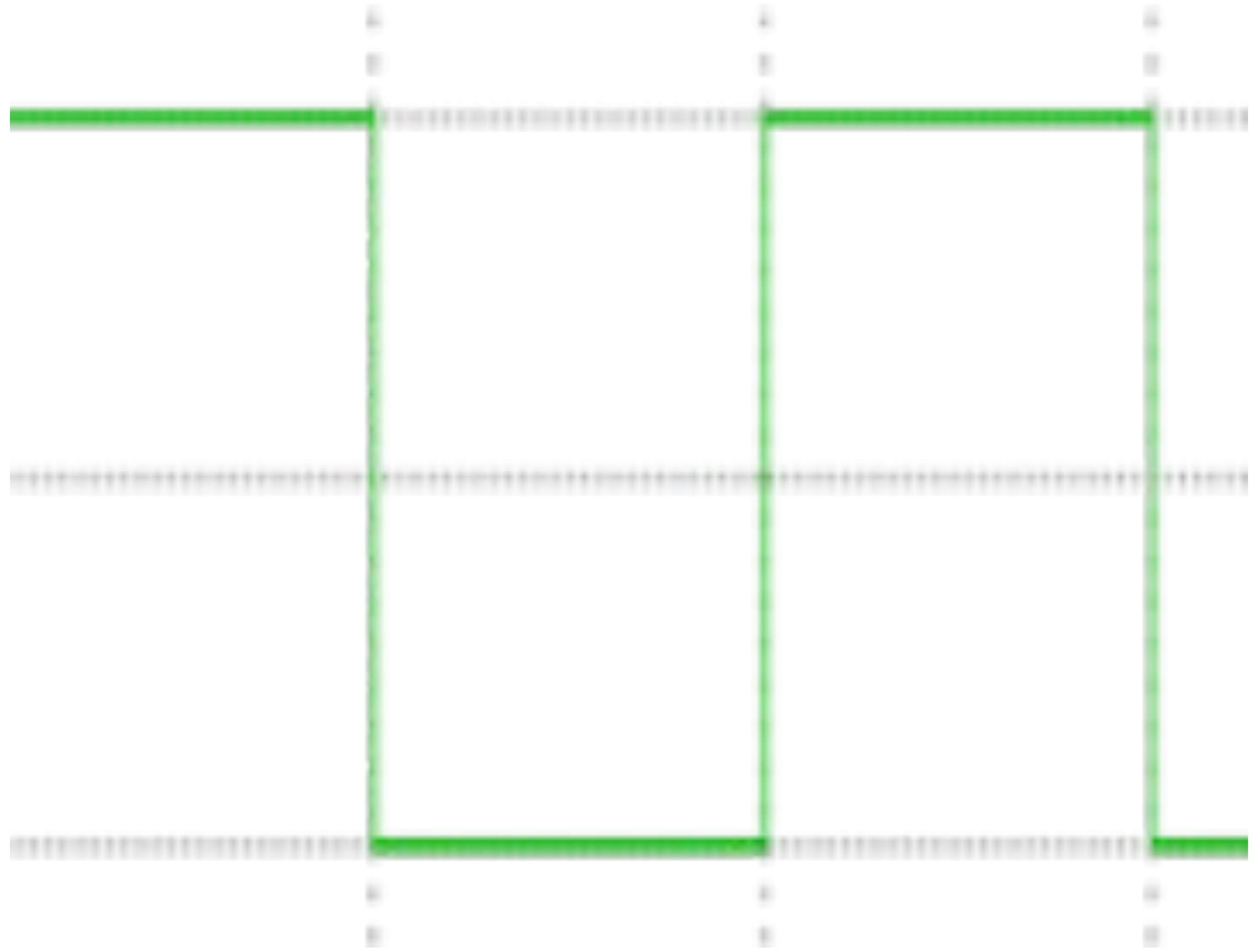
How would you generate this function?



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

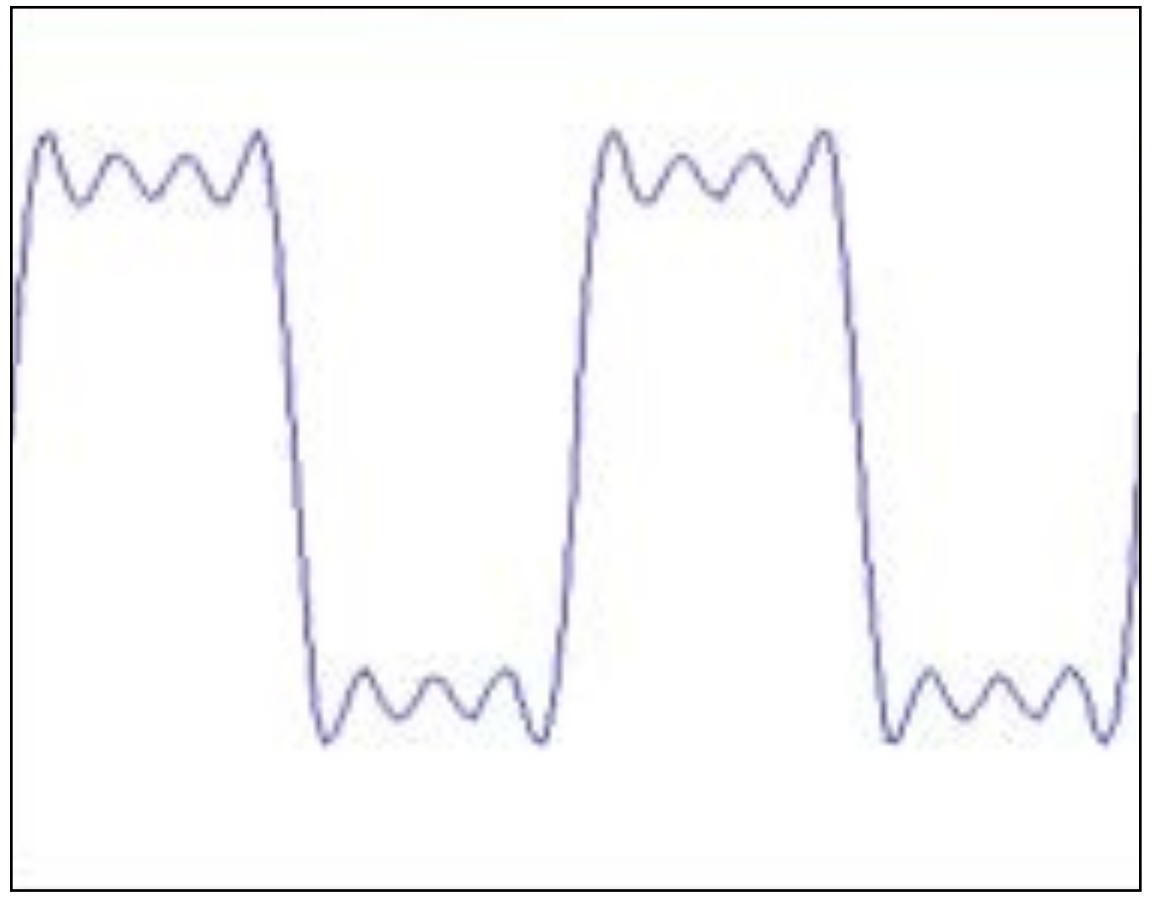
Fourier Transform (you will **NOT** be tested on this)

How would you generate this function?

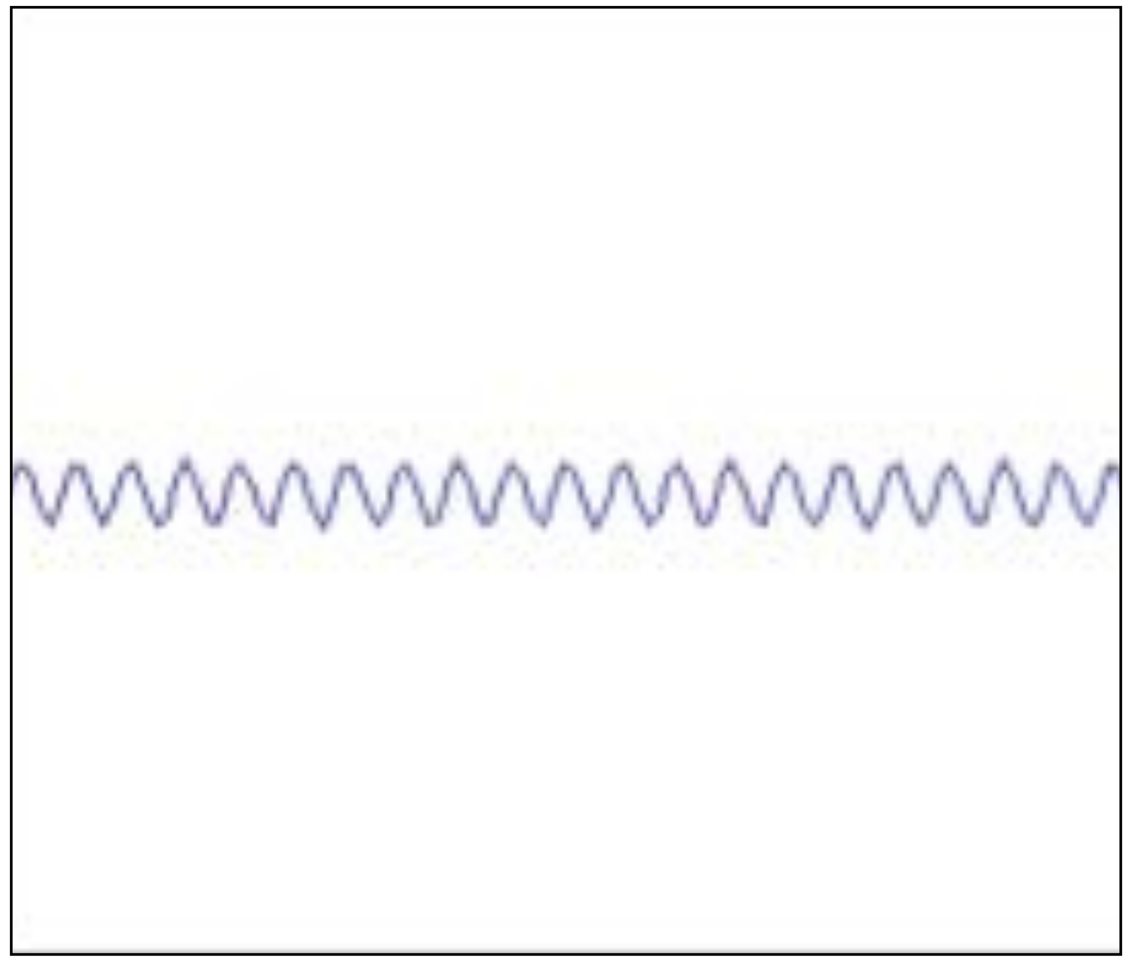


square wave

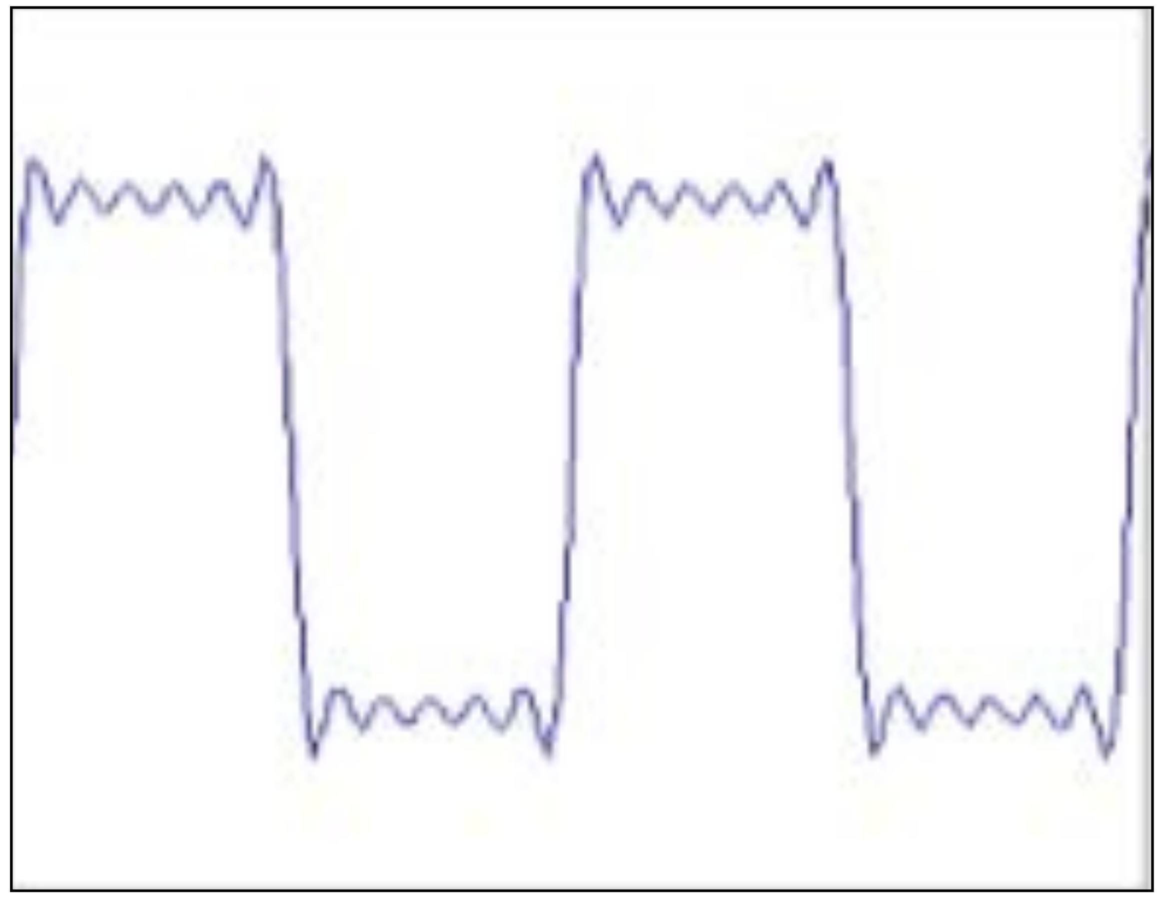
≈



+



=

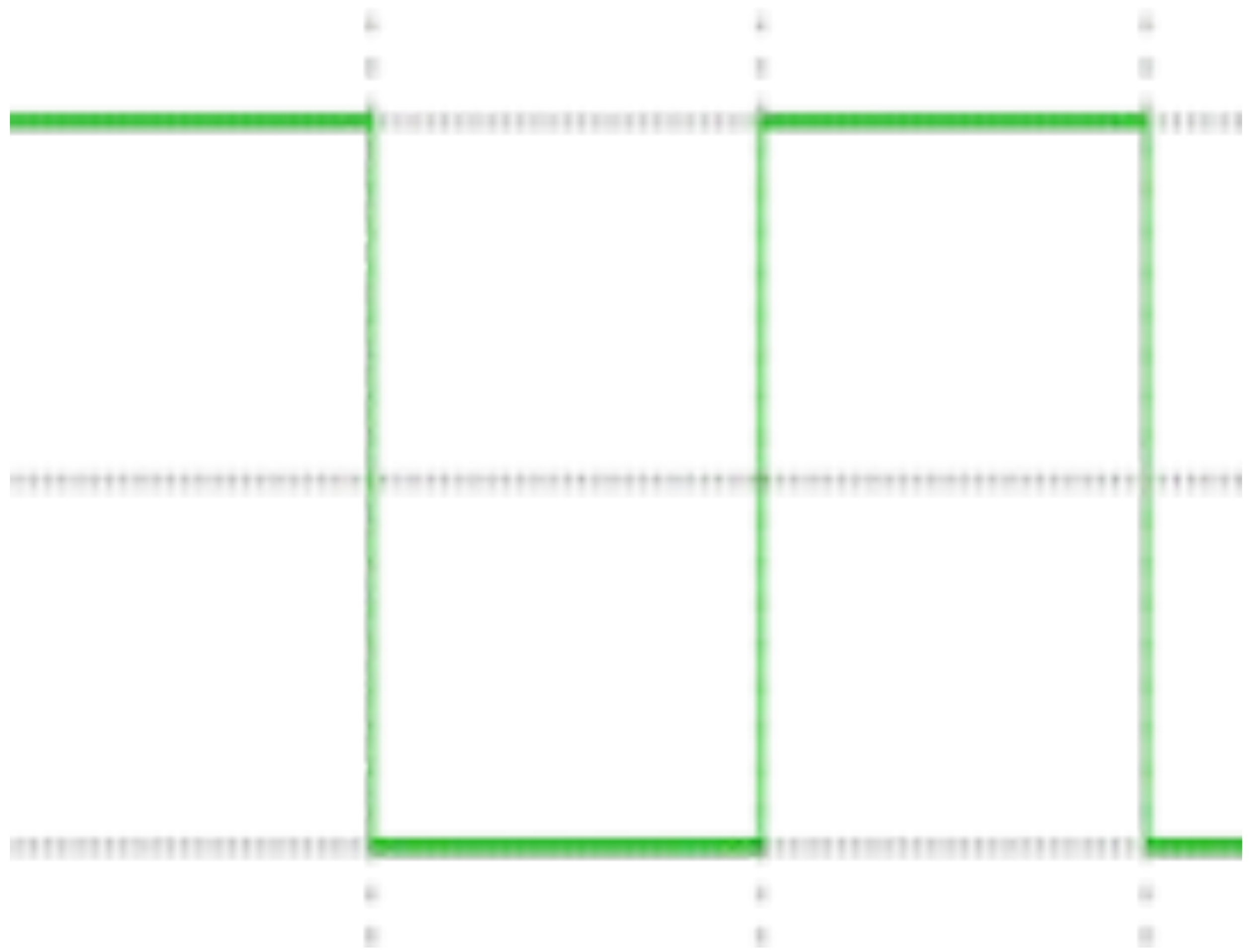


How would you express this mathematically?

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

How would you generate this function?



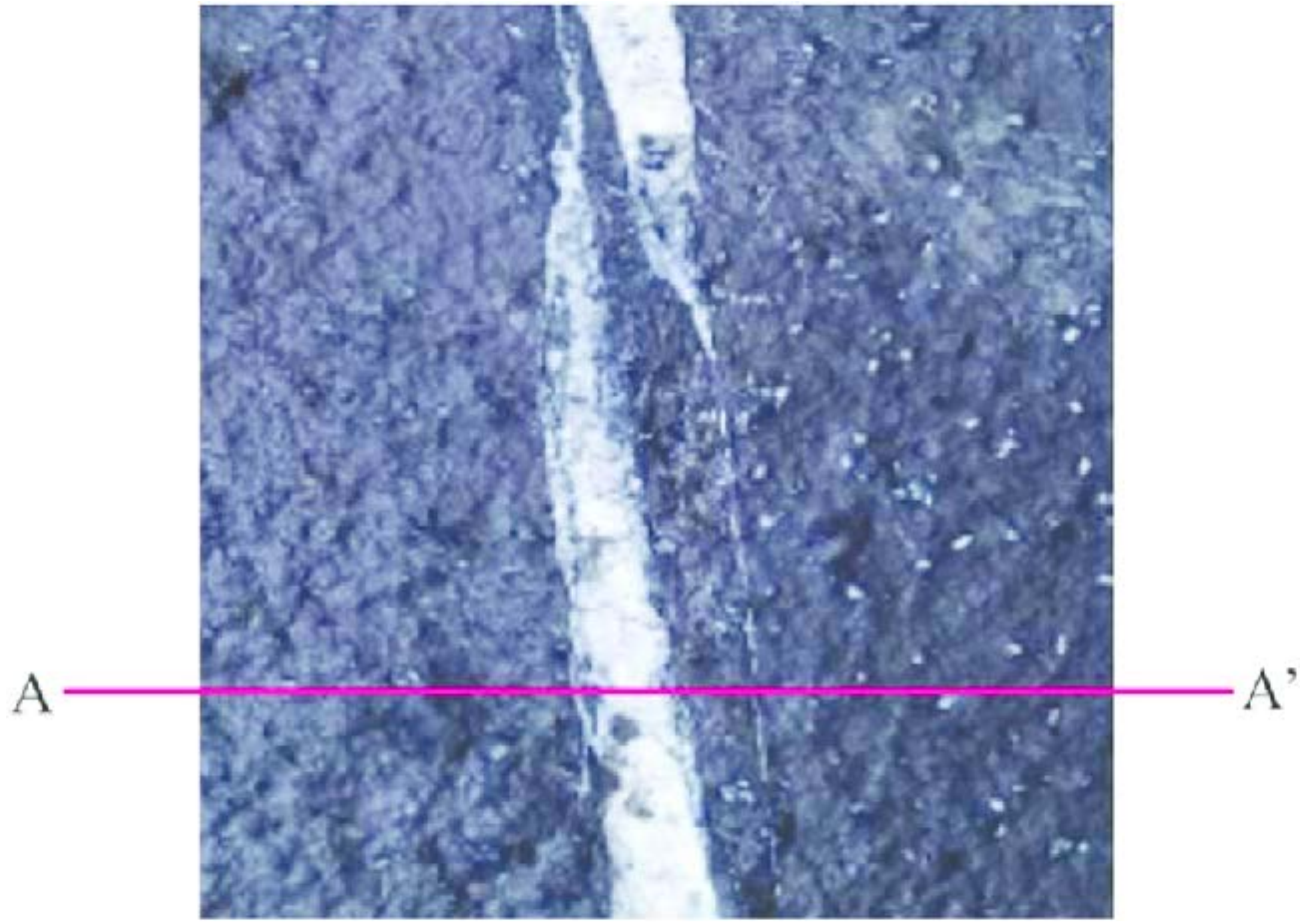
square wave

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

infinite sum of sine waves

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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



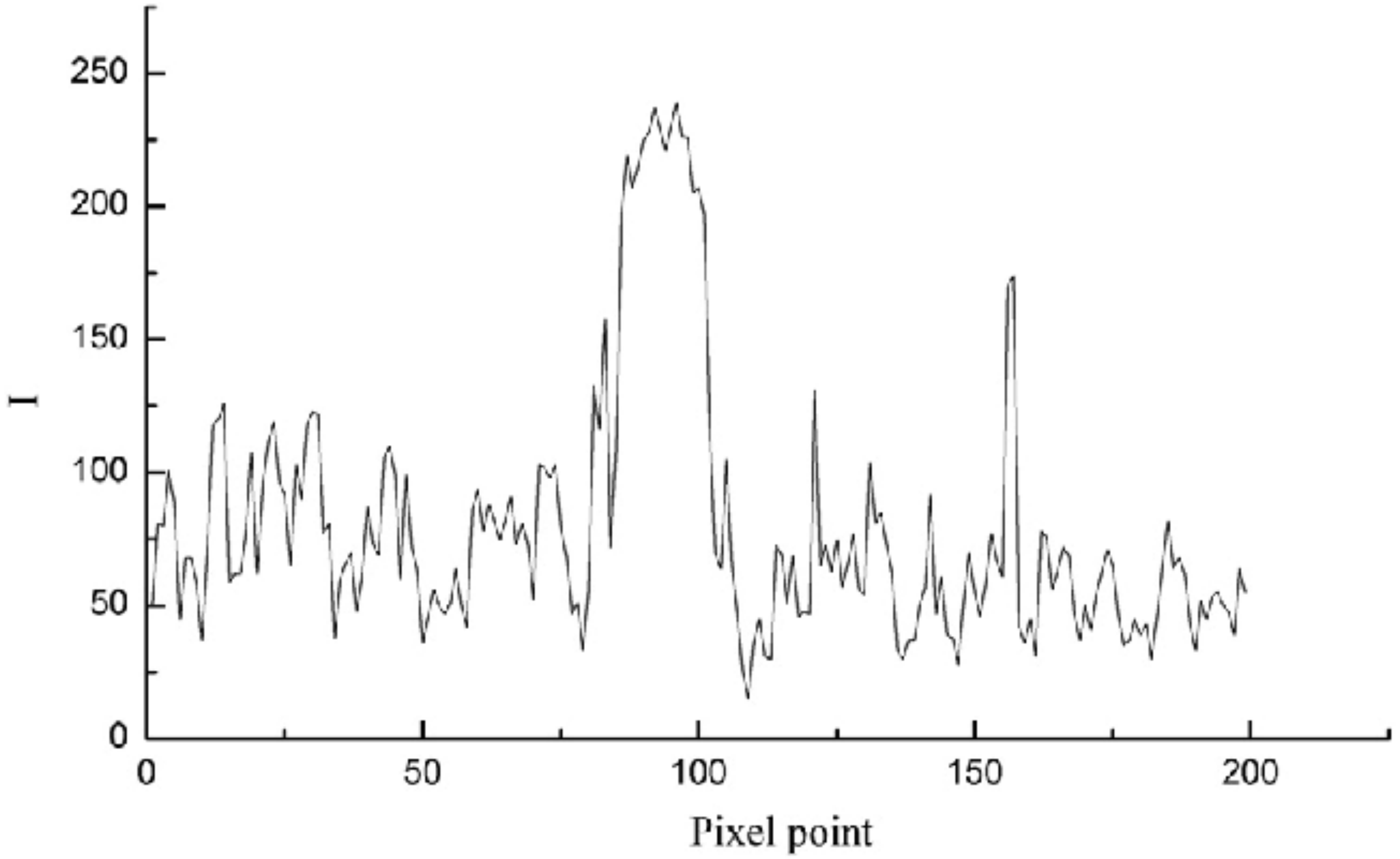


Image from: Numerical Simulation and Fractal Analysis of Mesoscopic Scale Failure in Shale Using Digital Images

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

What are “frequencies” in an image?

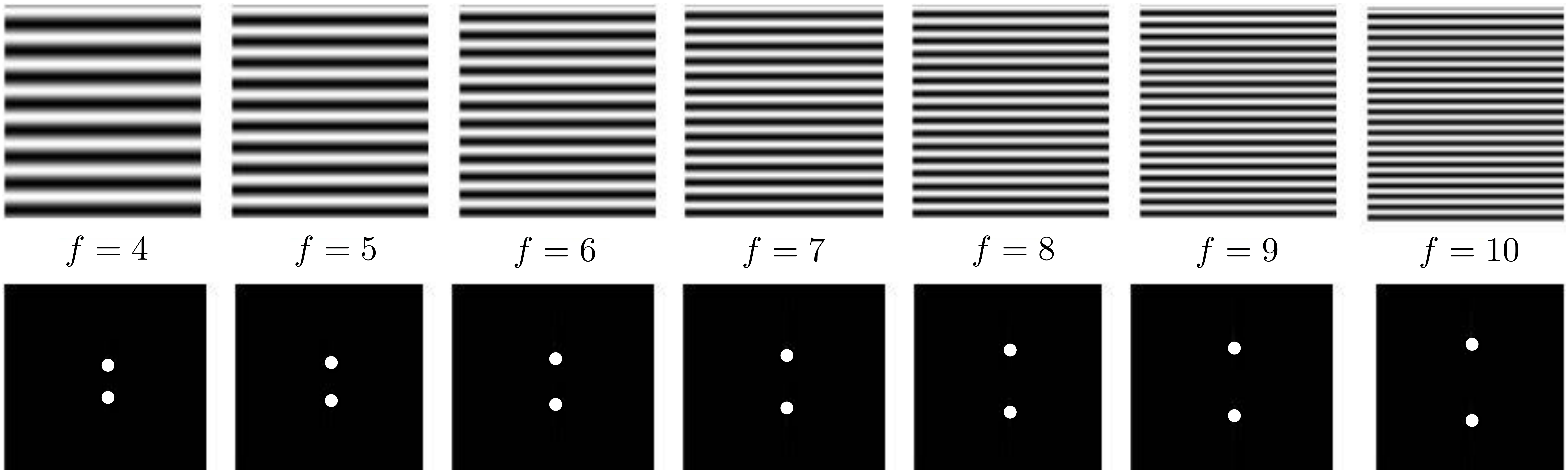
Spatial frequency

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

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

What are "frequencies" in an image?

Spatial frequency

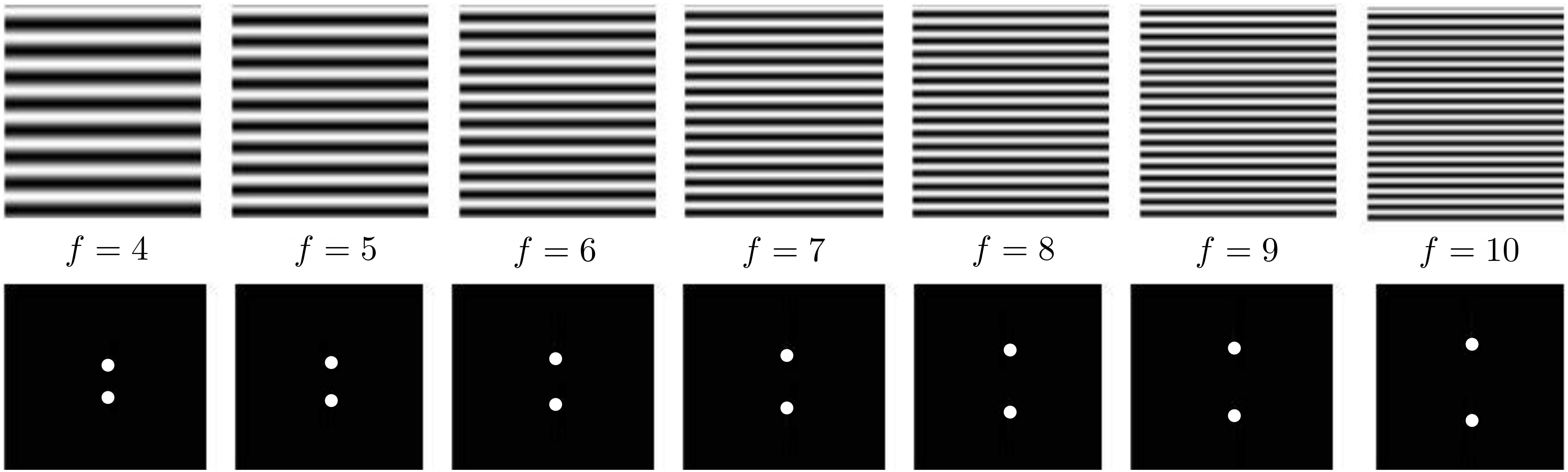


Amplitude (magnitude) of Fourier transform (phase does not show desirable correlations with image structure)

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

What are "frequencies" in an image?

Spatial frequency



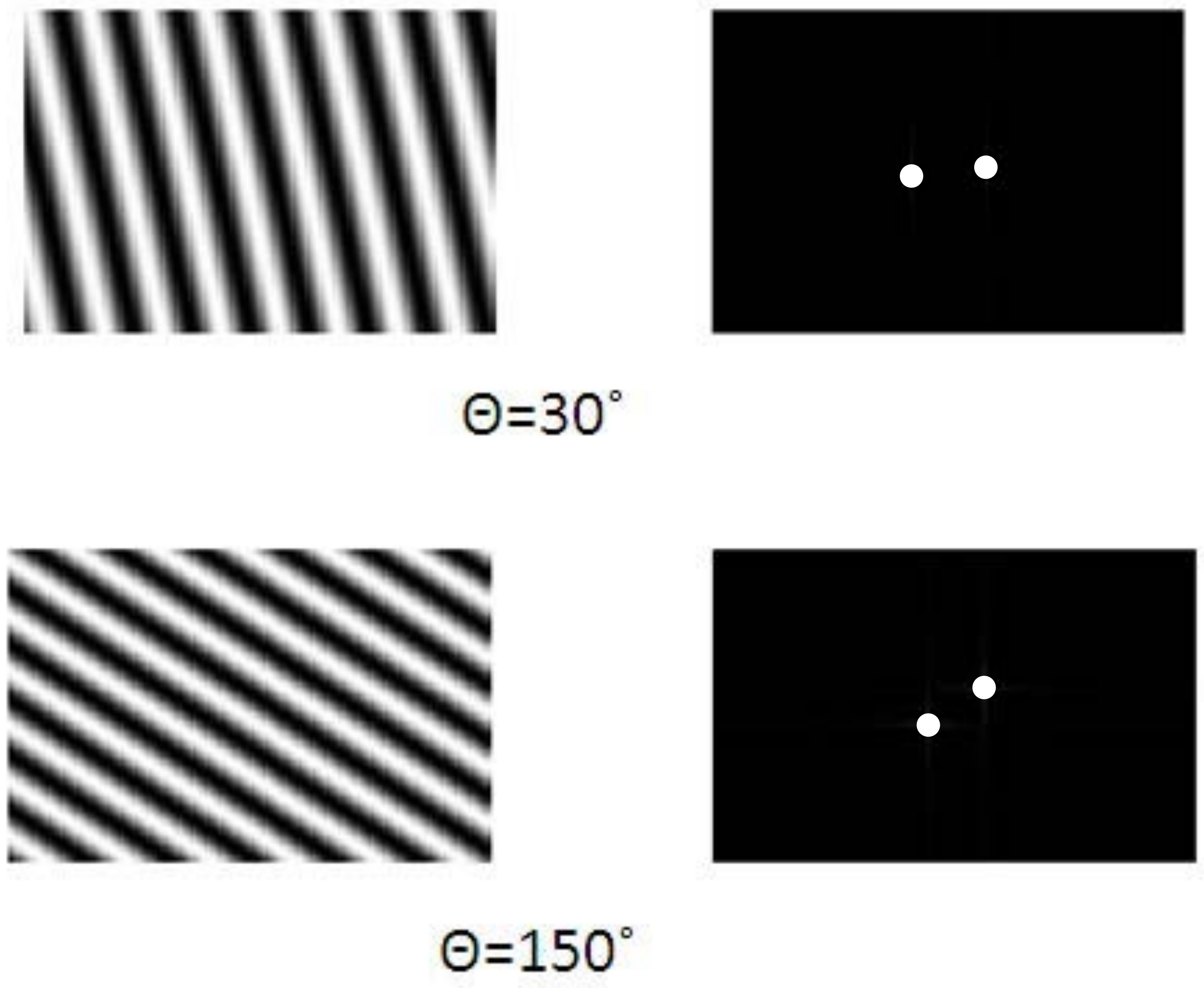
Amplitude (magnitude) of Fourier transform (phase does not show desirable correlations with image structure)

Observation: low frequencies close to the center

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

What are “frequencies” in an image?

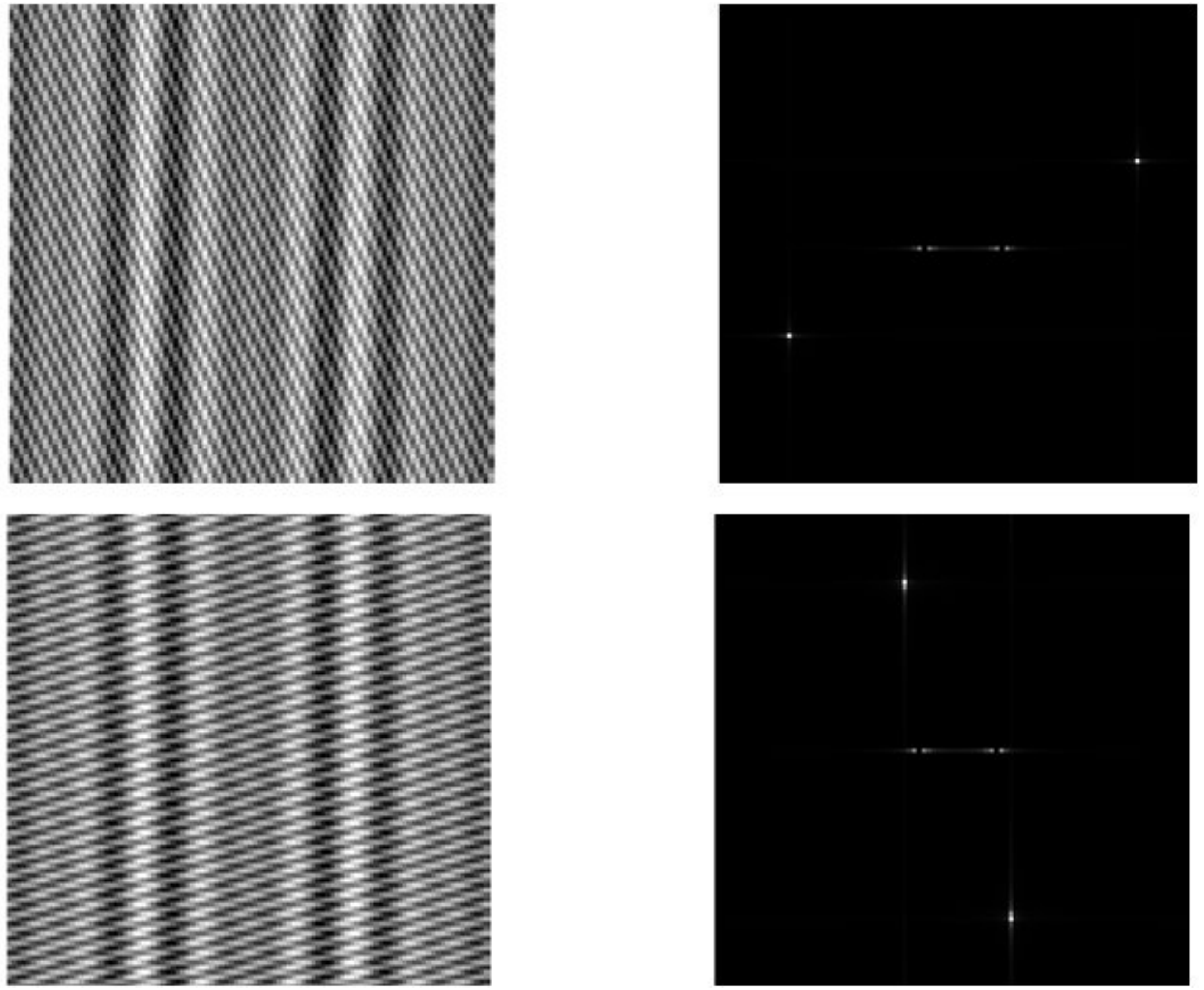
Spatial frequency



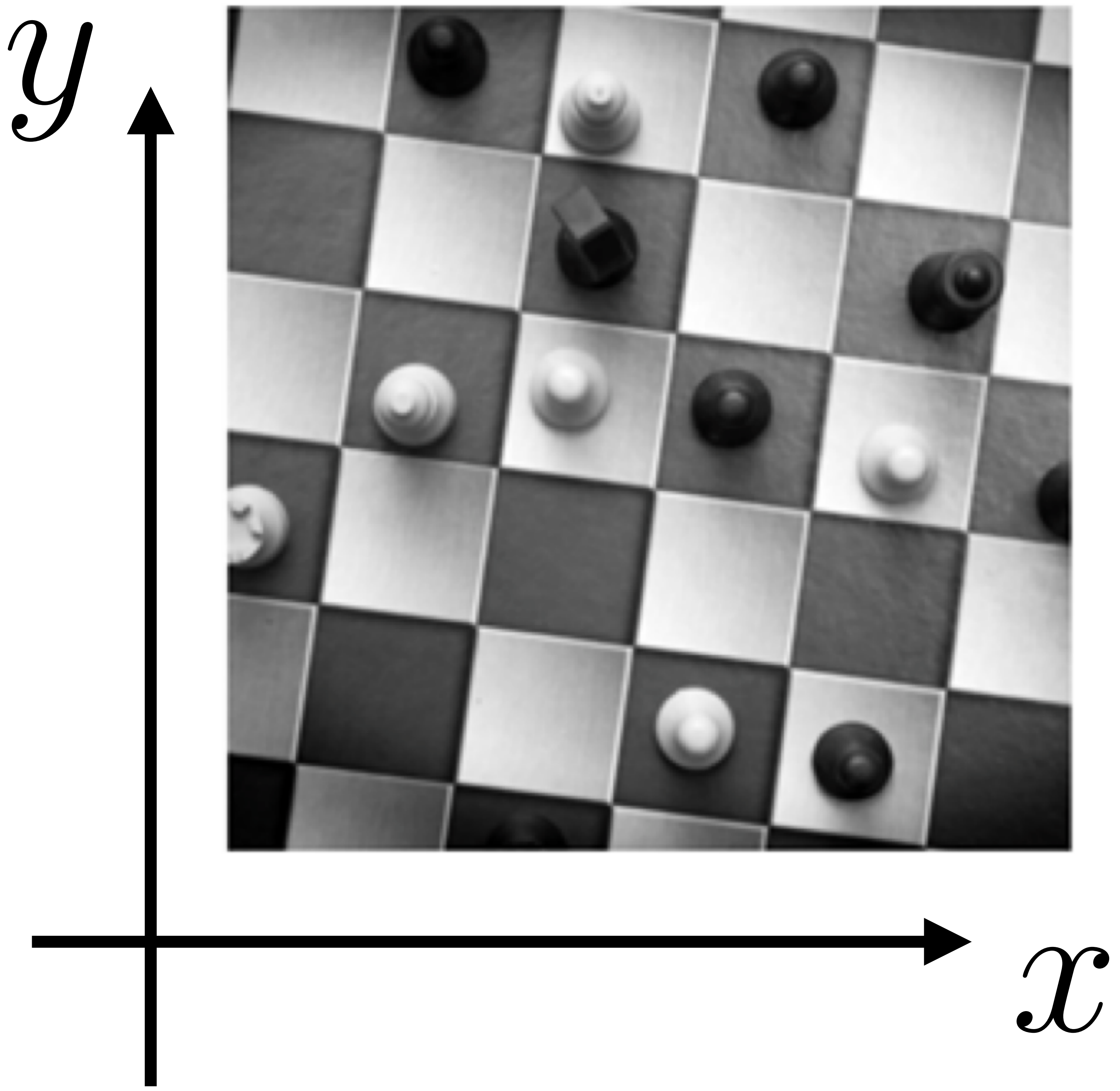
Fourier Transform (you will **NOT** be tested on this)

What are "frequencies" in an image?

Spatial frequency

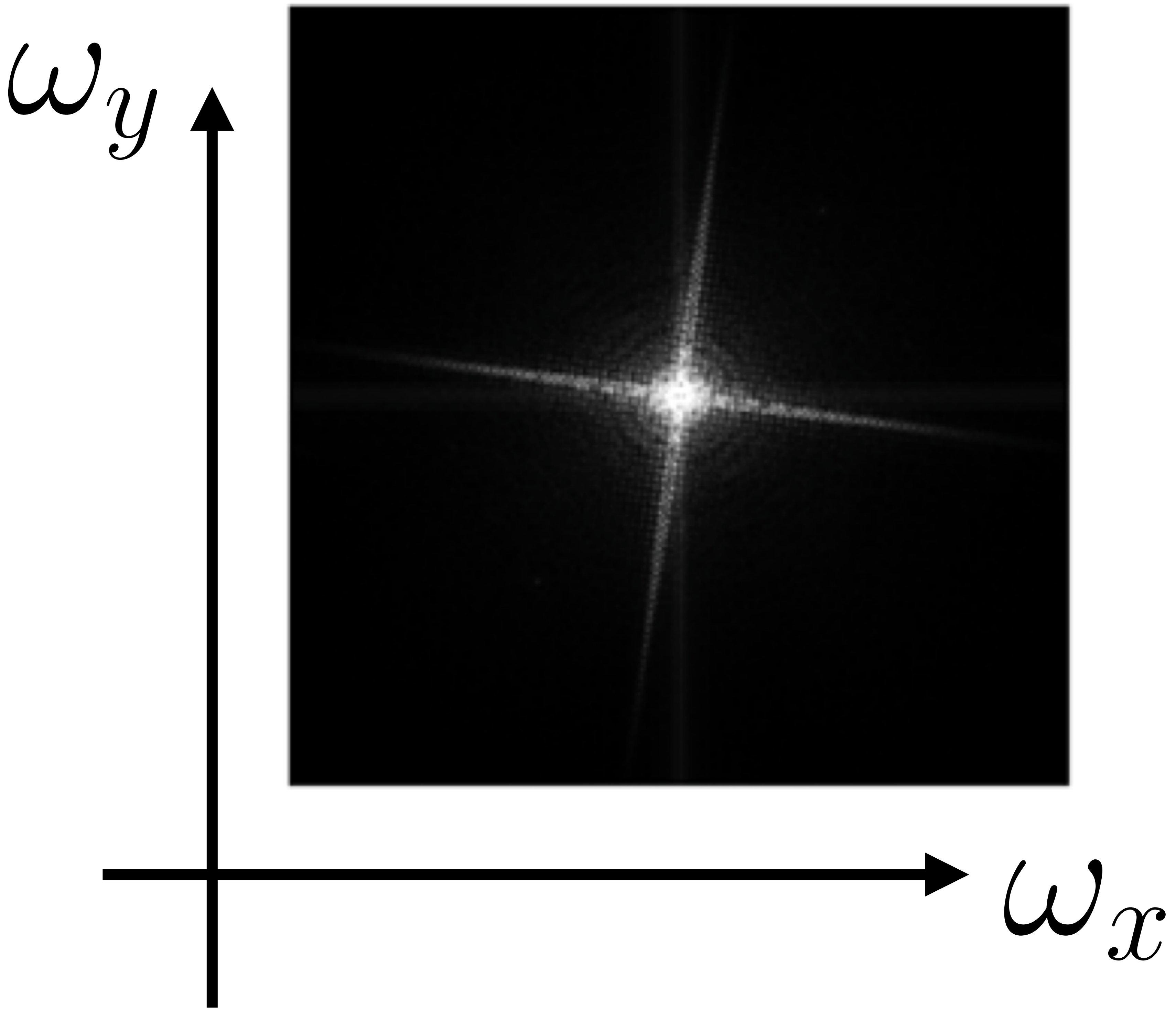


2D Fourier Transforms: Images



$$f(x, y)$$

Image

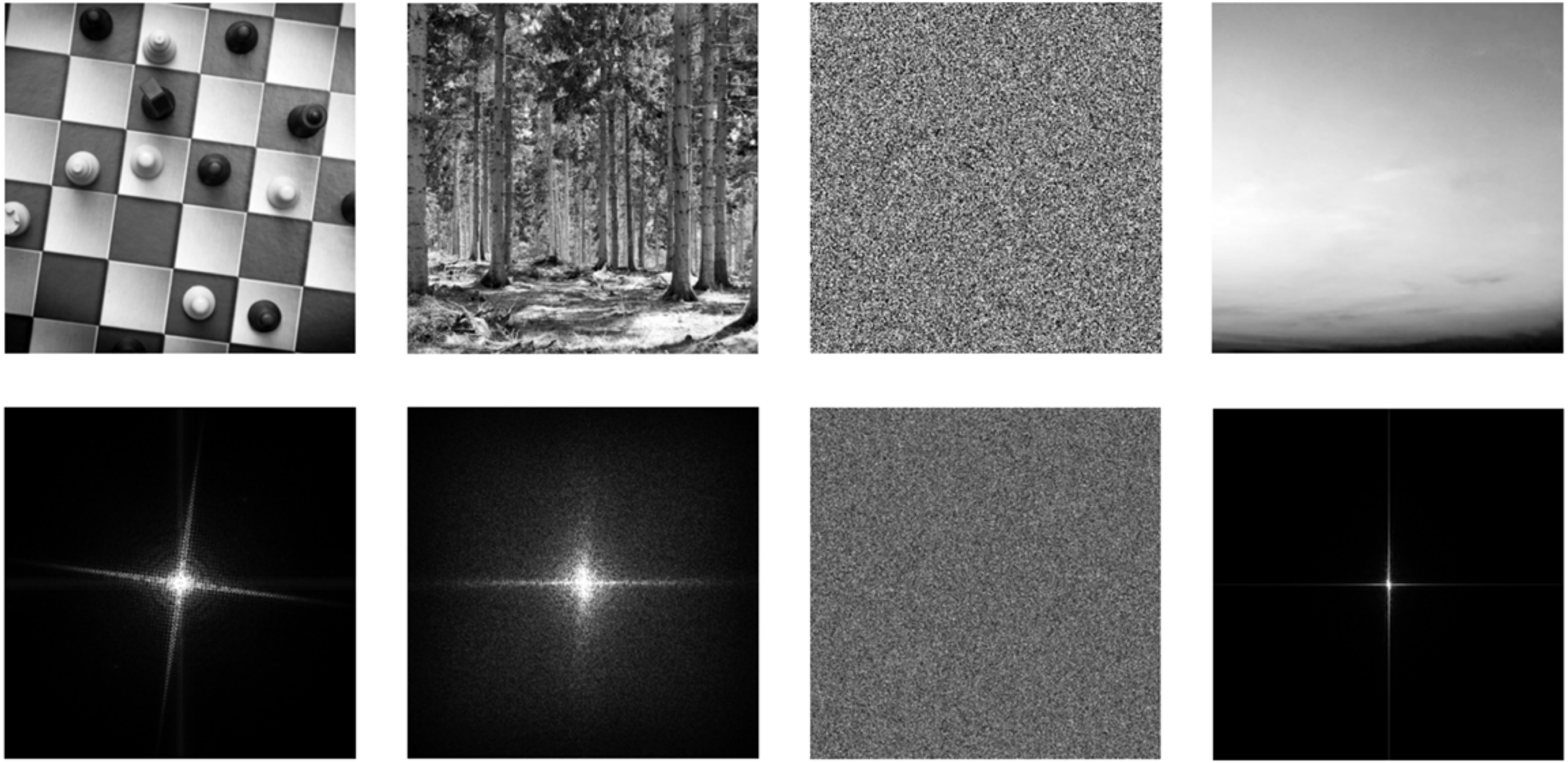


$$F(\omega_x, \omega_y)$$

Fourier Transform

5.2

2D Fourier Transforms: Images



Aside: You will not be tested on this …

Image

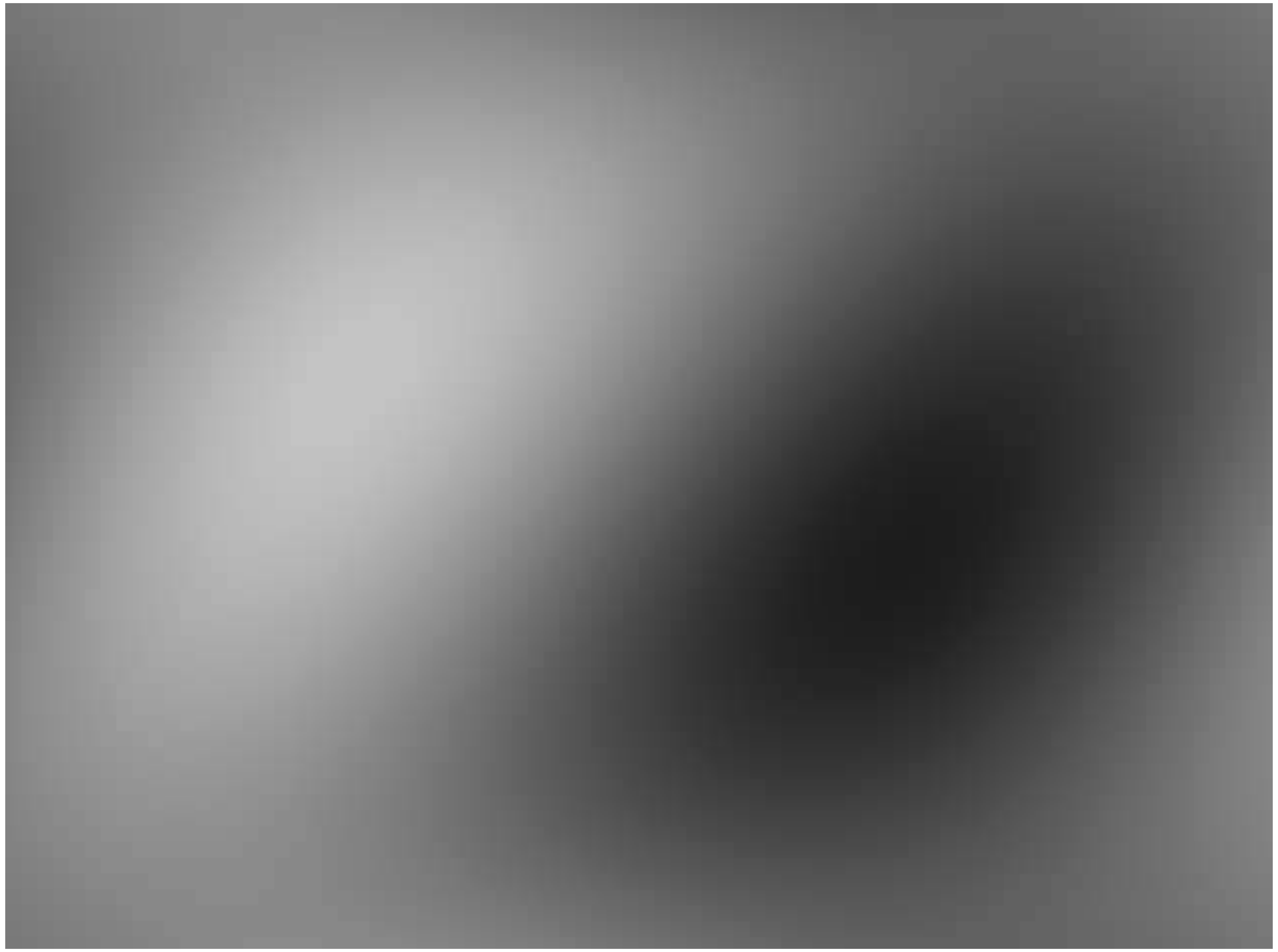
https://photo.stackexchange.com/questions/40401/what-does-frequency-mean-in-an-image/40410#40410

Aside: 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

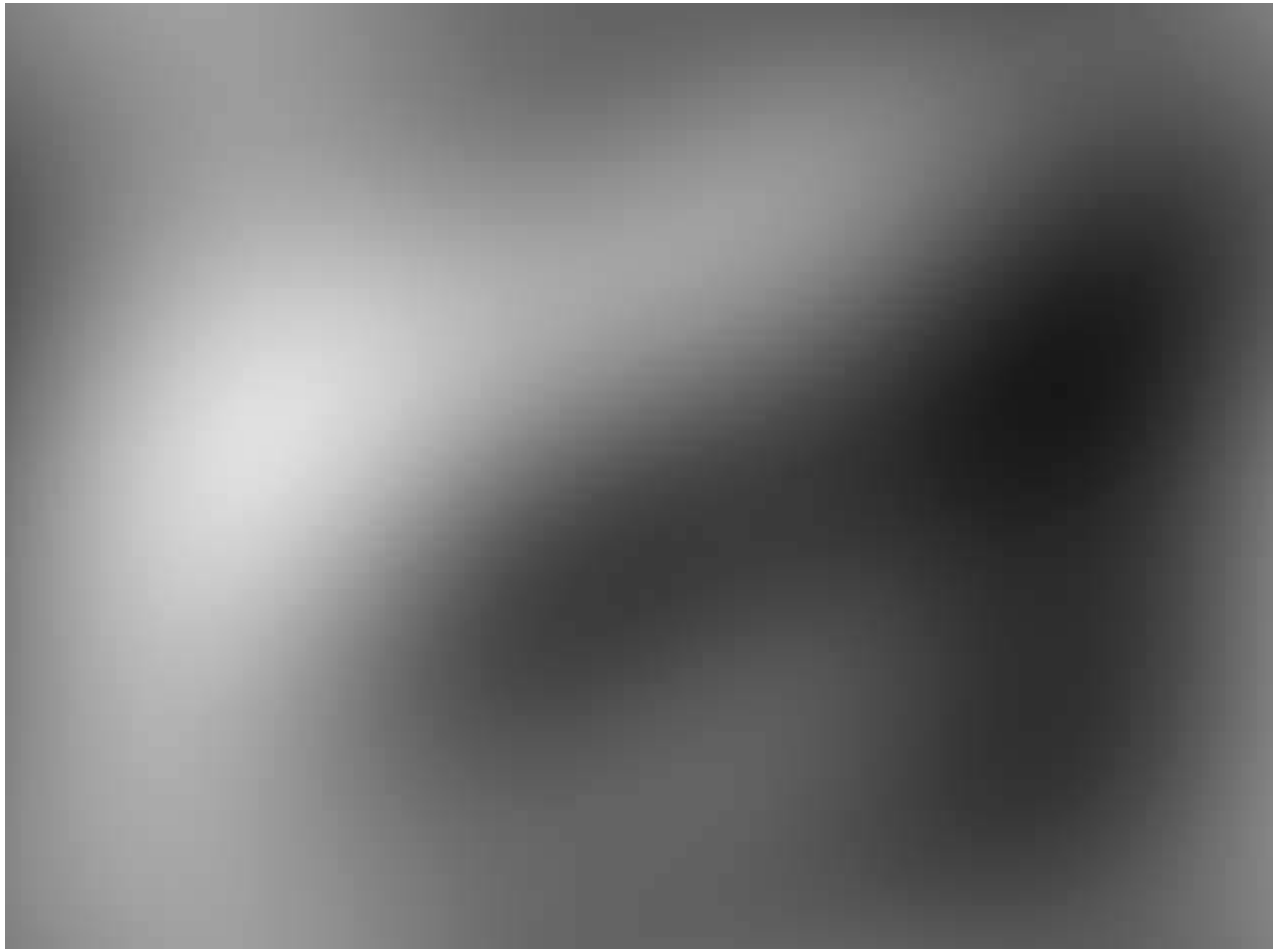
Aside: You will not be tested on this …



\+ **Second** frequency

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

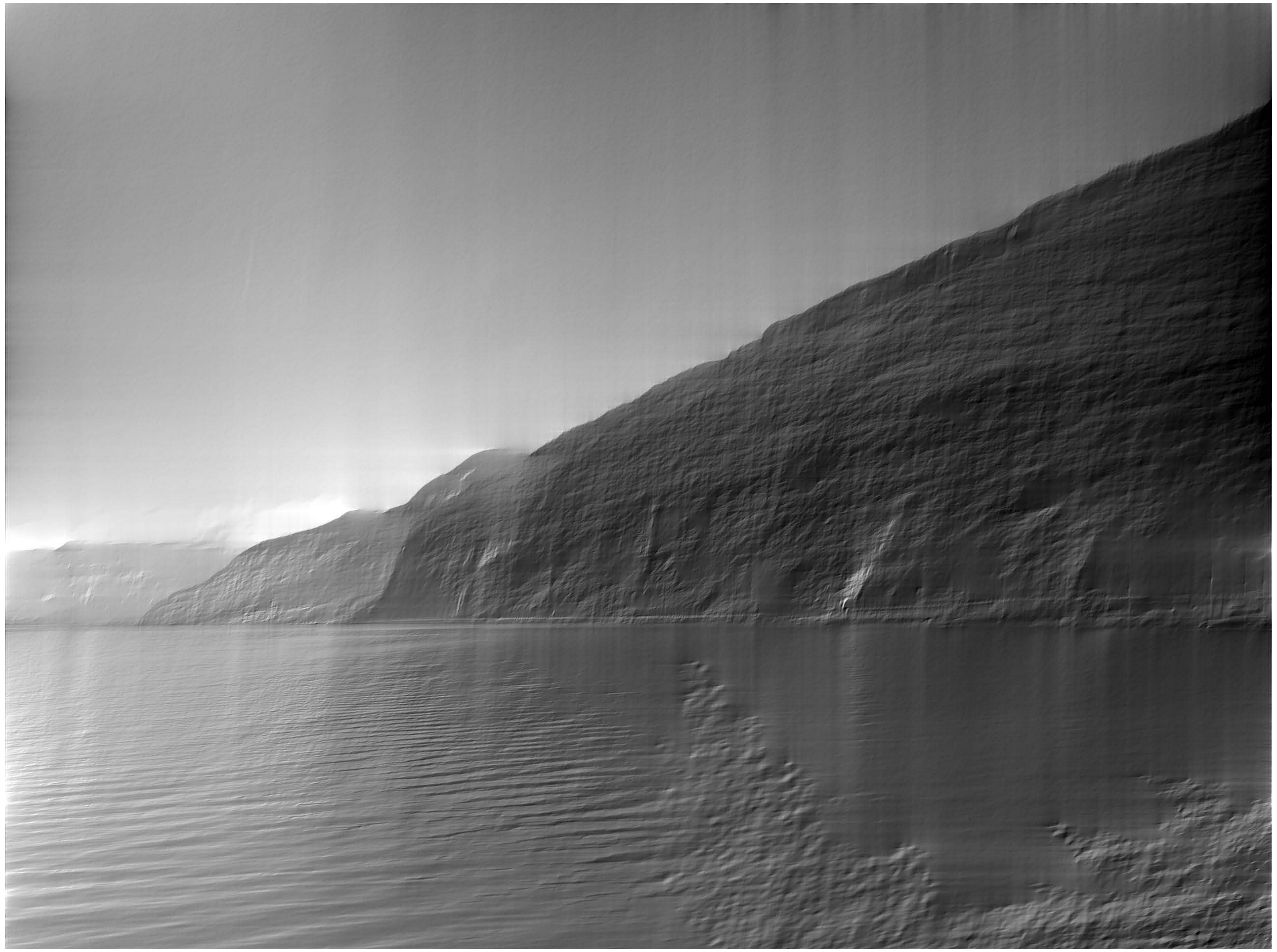
Aside: You will not be tested on this …



+ **Third** frequency

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

Aside: 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

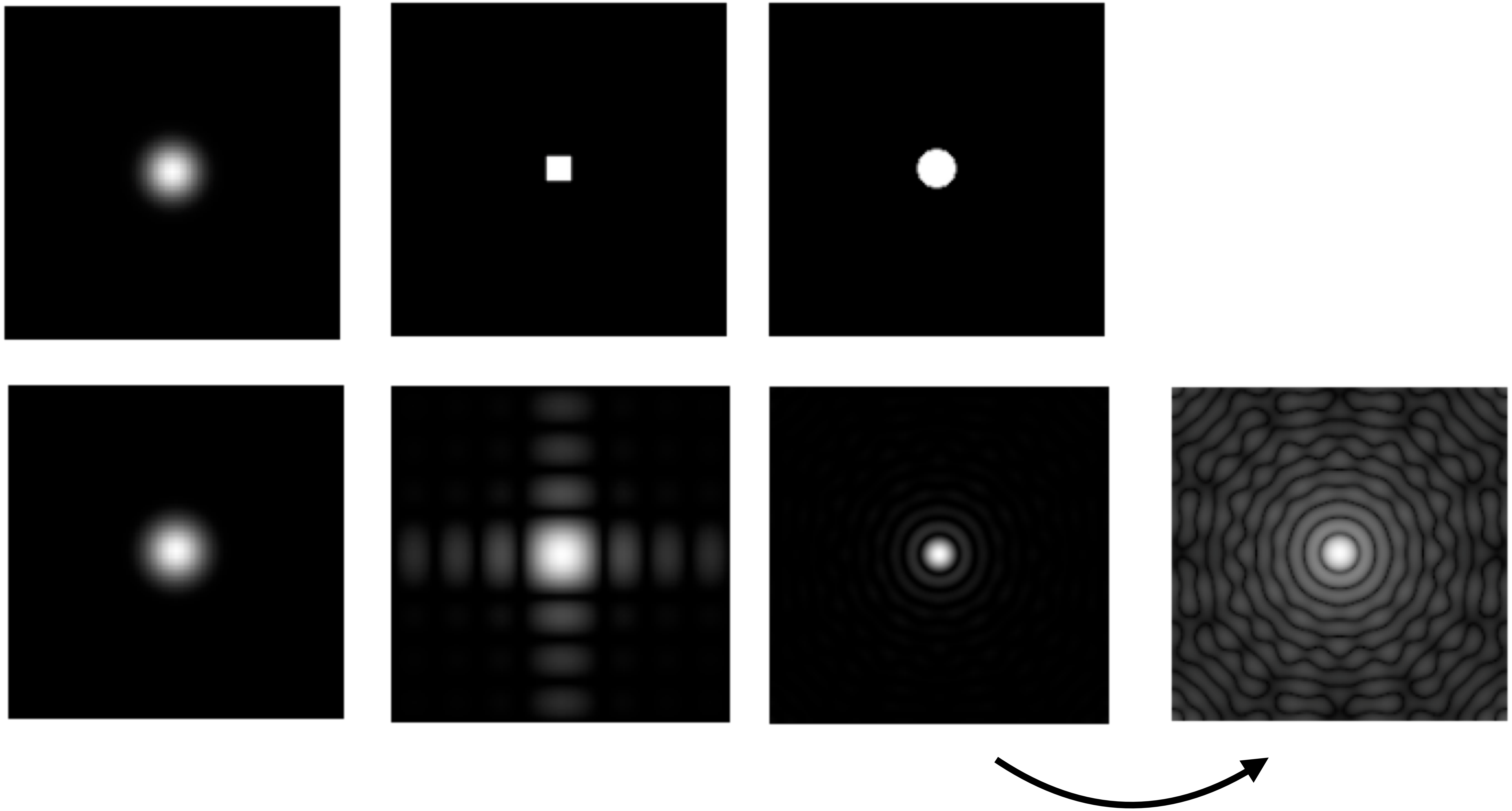
Aside: You will not be tested on this …

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

2D Fourier Transforms: Kernels

$$f(x, y)$$

$$F(\omega_x, \omega_y)$$



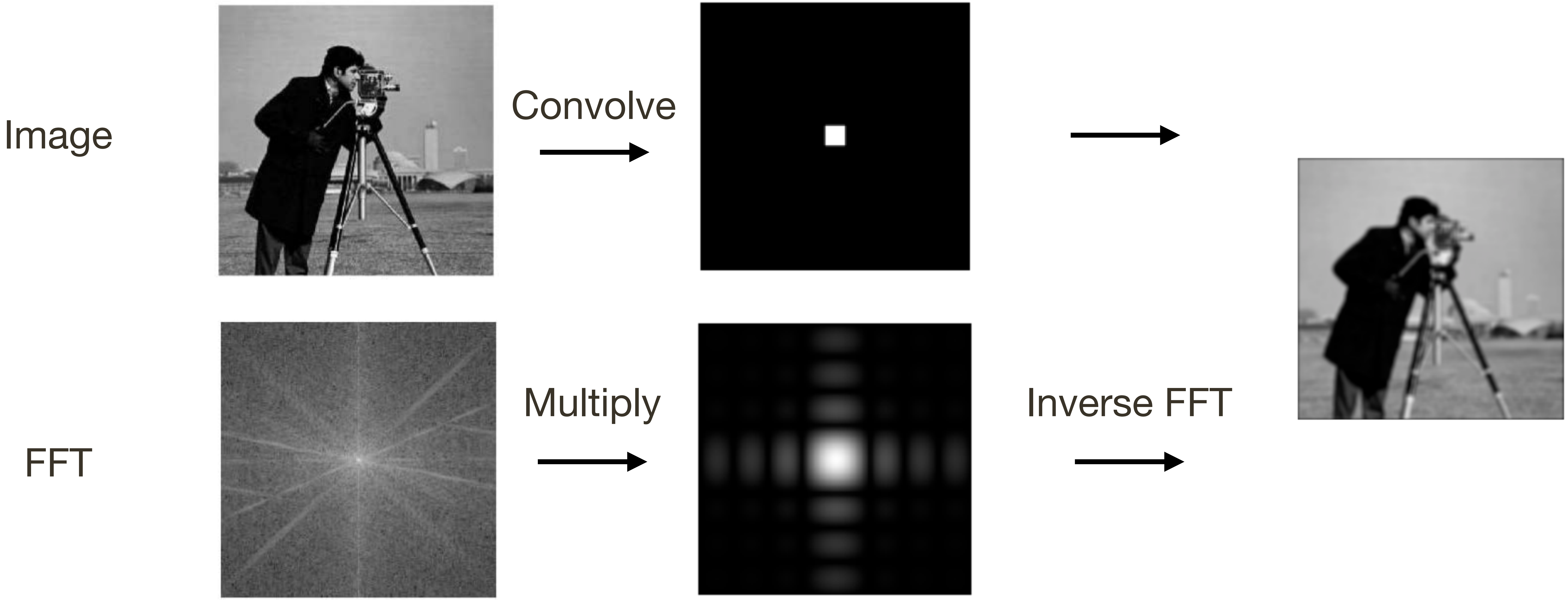
Compress power 0.5
to exaggerate lobes
(just for visualization)

Convolution using **Fourier Transforms**

Convolution **Theorem**:

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

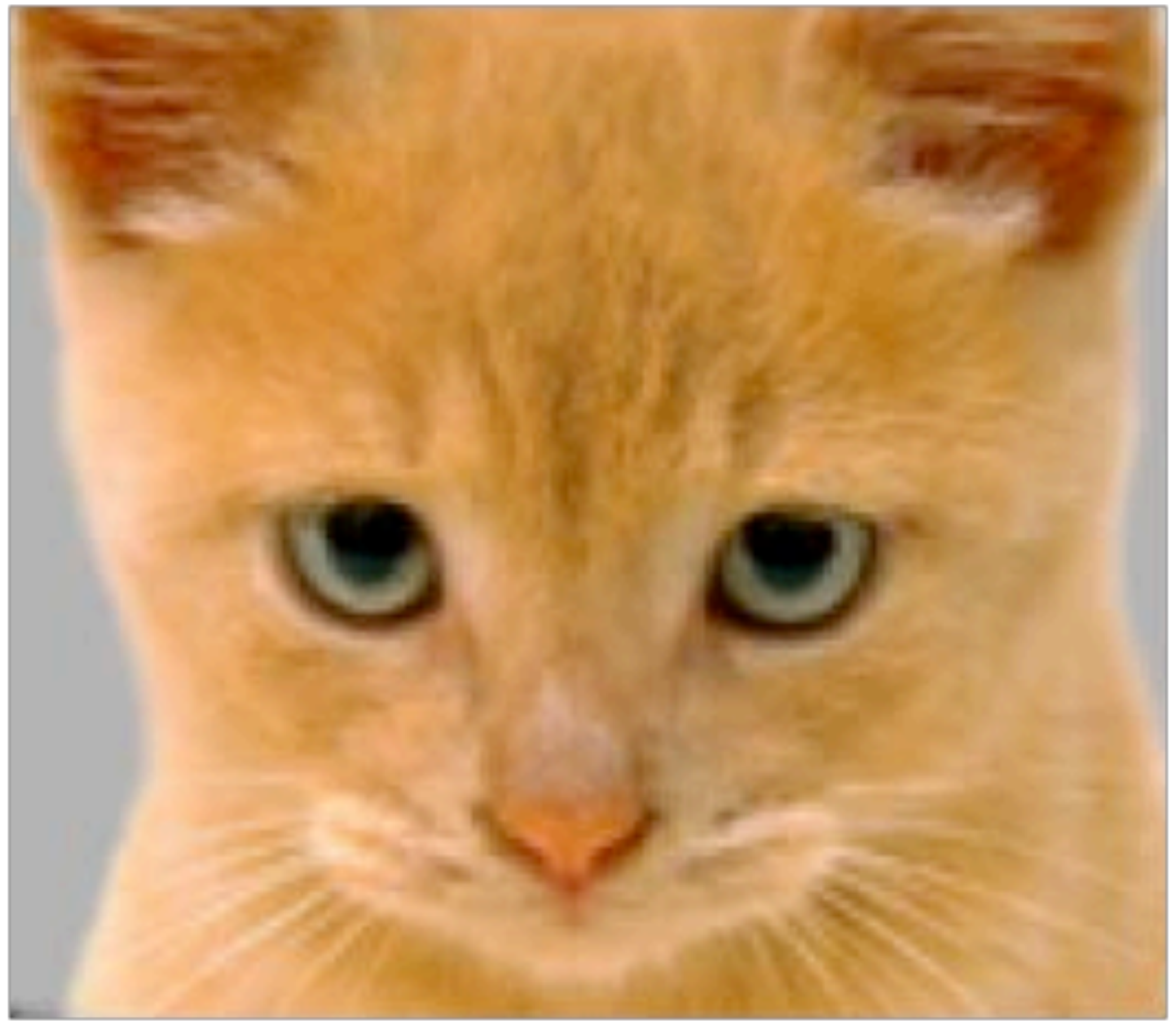
$$\mathcal{I}'(w_x, w_y) = \mathcal{F}(w_x, w_y)\, \mathcal{I}(w_x, w_y)$$



What preceded was for fun

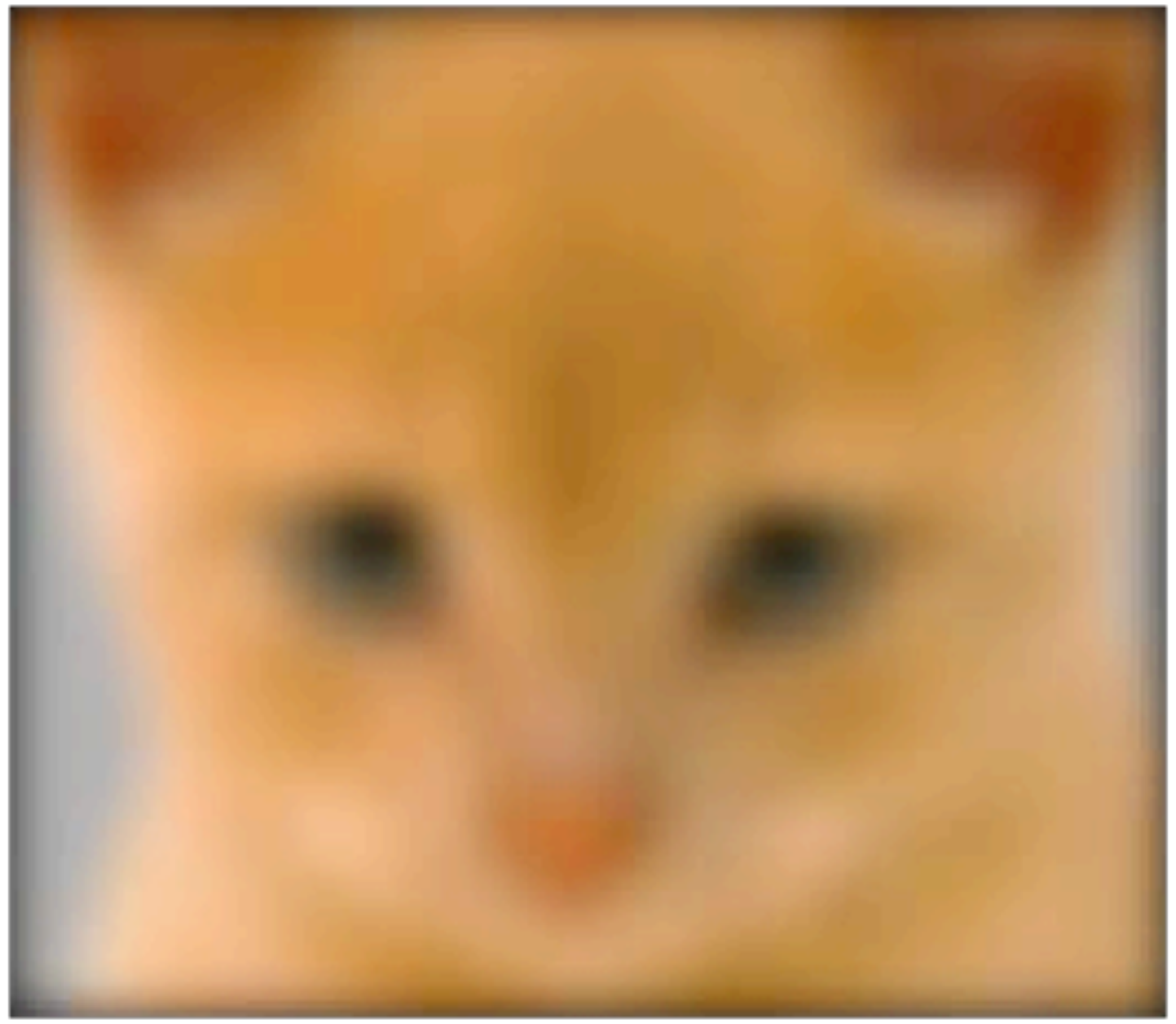
(you will **NOT** be tested on it)

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



Original

$I(x, y)$



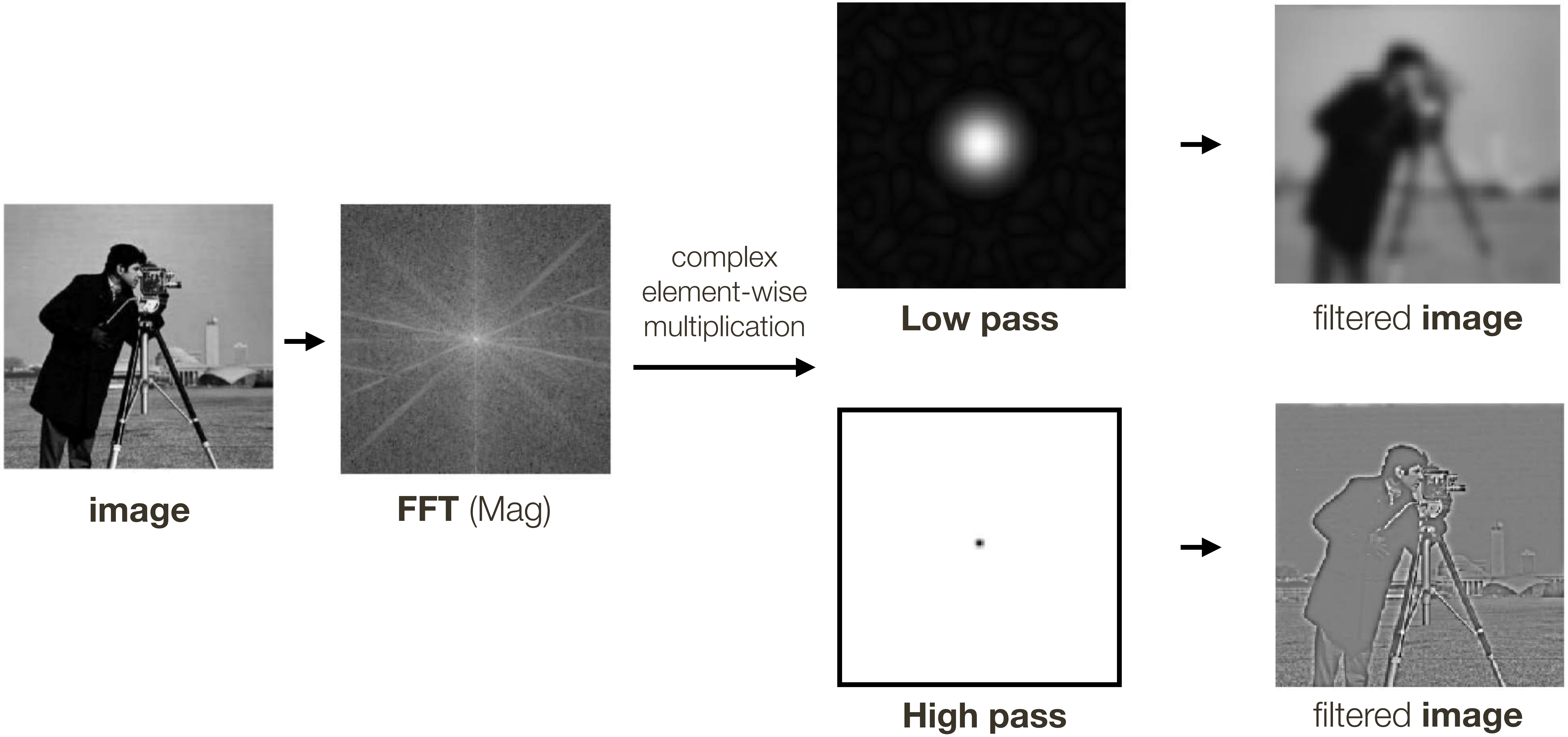
Low-Pass Filter

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

High-Pass Filter

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

Aside: You will not be tested on this …



Convolution using **Fourier Transforms**

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: $\mathcal{O}(m^2 \log m)$

Worthwhile if image and kernel are **both** large

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 selects the **median** value from each pixel's neighborhood.

Non-linear Filtering

"shot" noise

gaussian blurred

median filtered

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

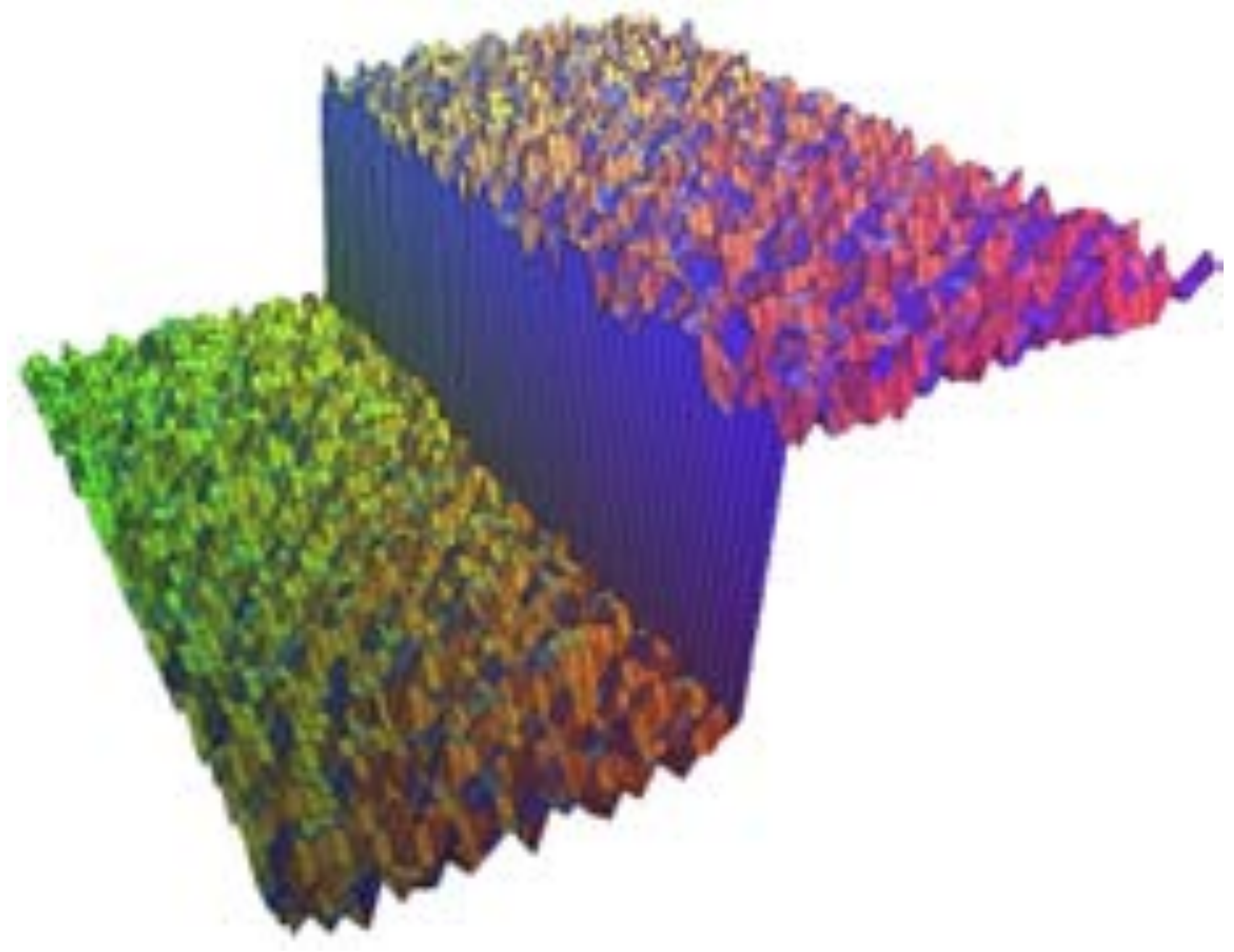
Median Filter

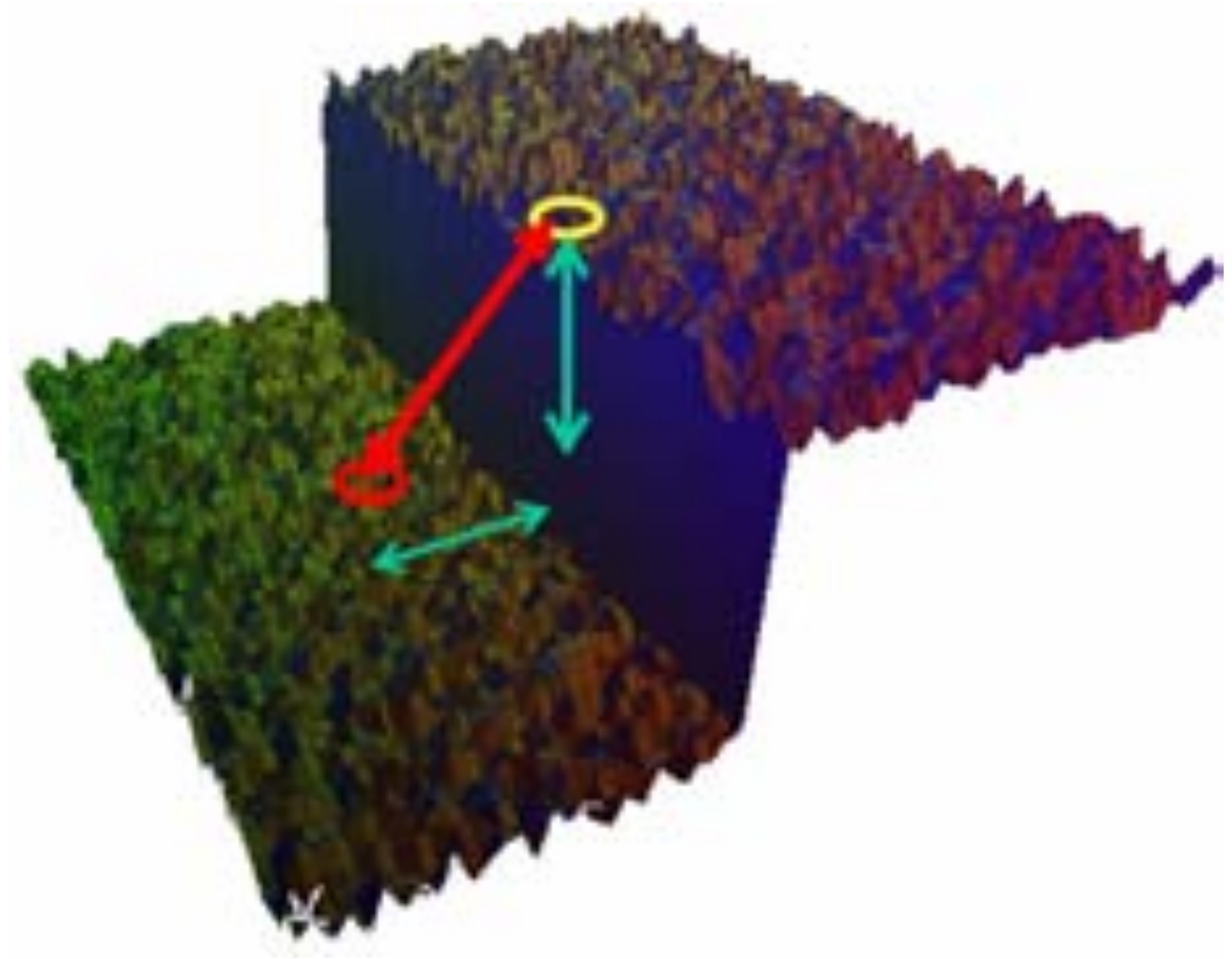
Effective at reducing certain kinds of noise, such as impulse noise (a.k.a 'salt and pepper' noise or 'shot' noise)

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





Suppose we want to smooth a noisy step function

A Gaussian kernel performs a weighted average of points over a spatial neighbourhood..

But this averages points both at the top and bottom of the step — blurring

Bilateral Filter idea: look at distances in **range** (value) as well as **space** x,y

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:

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

(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:

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

(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

→

Normalised

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

$\sigma_d = 1$

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

→

Normalised 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

$\sigma_d = 1$

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

(differences based on **centre pixel**)

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

→

Normalised
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
$\sigma_d = 1$

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

(differences based on **centre pixel**)

multiply →

Range * **Domain** Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

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

→

Normalised 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

$\sigma_d = 1$

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

(differences based on **centre pixel**)

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

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

→

Normalised 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

$\sigma_d = 1$

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

(differences based on **centre pixel**)

multiply →

Range * **Domain** Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

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

Bilateral Filter

Bilateral Filter

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

→

Normalised 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

$\sigma_d = 1$

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

Gaussian Filter (only)

Range Kernel

$\sigma_r = 0.45$

0.98	0.98	0.2
1	1	0.1
0.98	1	0.1

(differences based on **centre pixel**)

multiply →

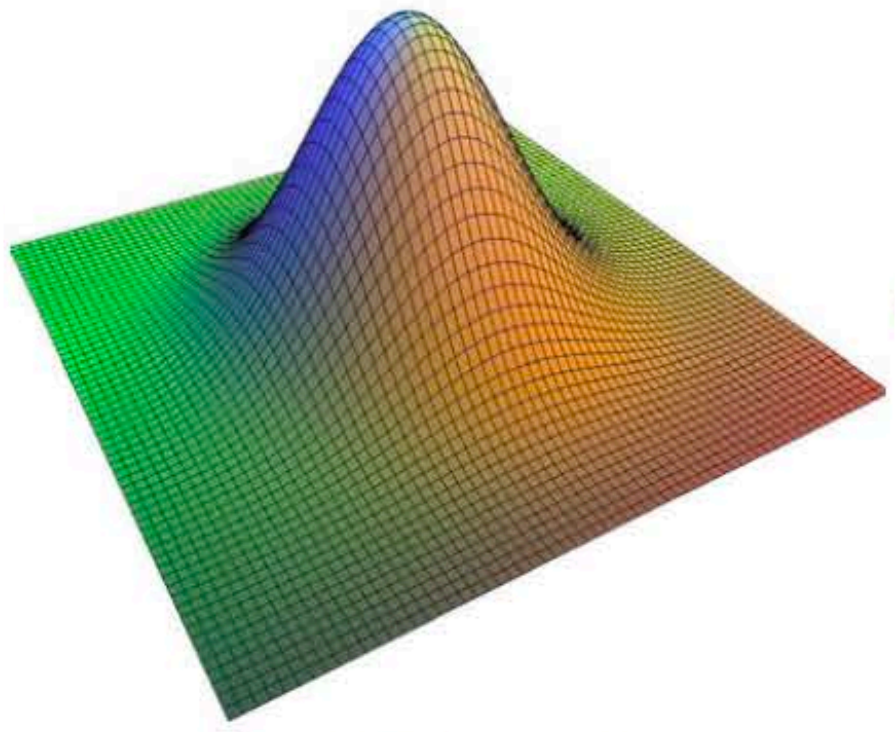
Range * **Domain** Kernel

0.08	0.12	0.02
0.12	0.20	0.01
0.08	0.12	0.01

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

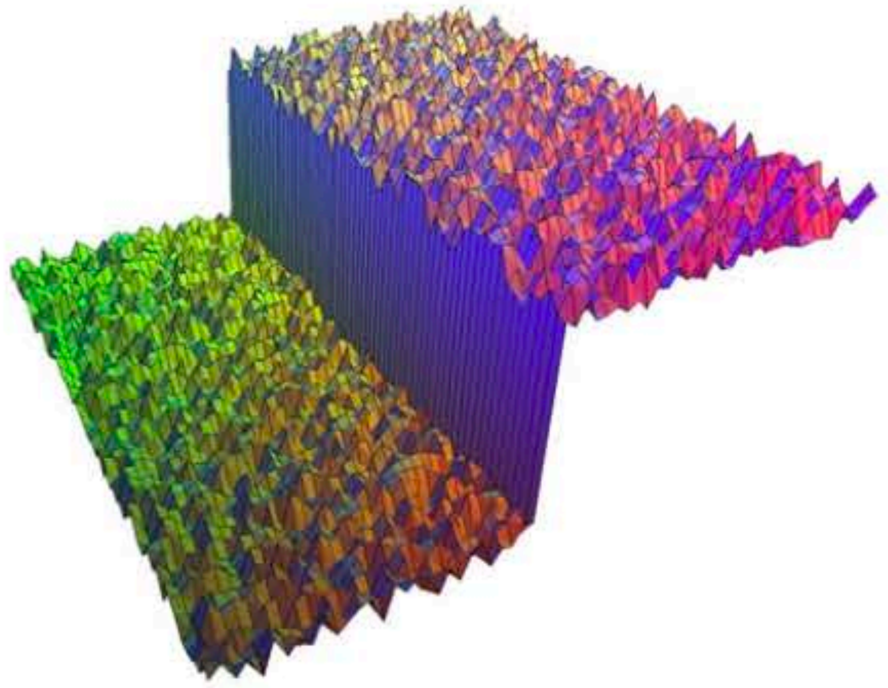
Bilateral Filter

Bilateral Filter

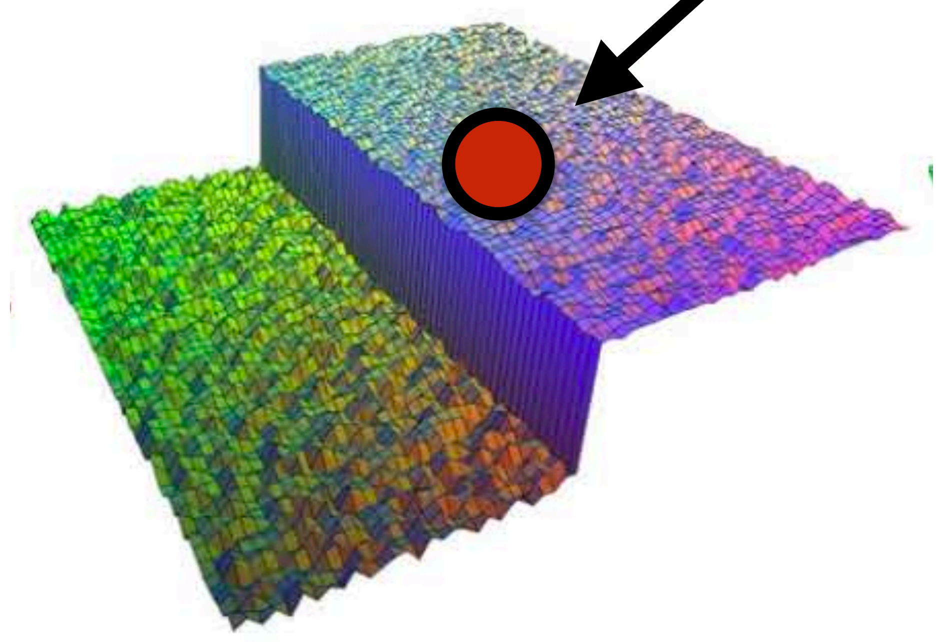


Domain Kernel

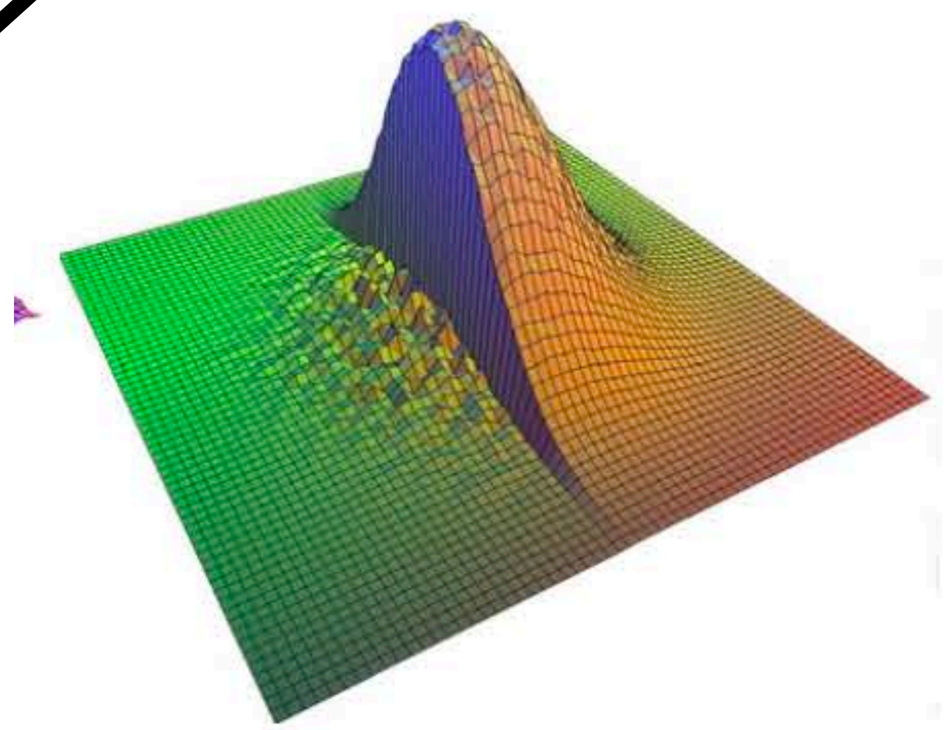
This example:
weights for point
on top of edge



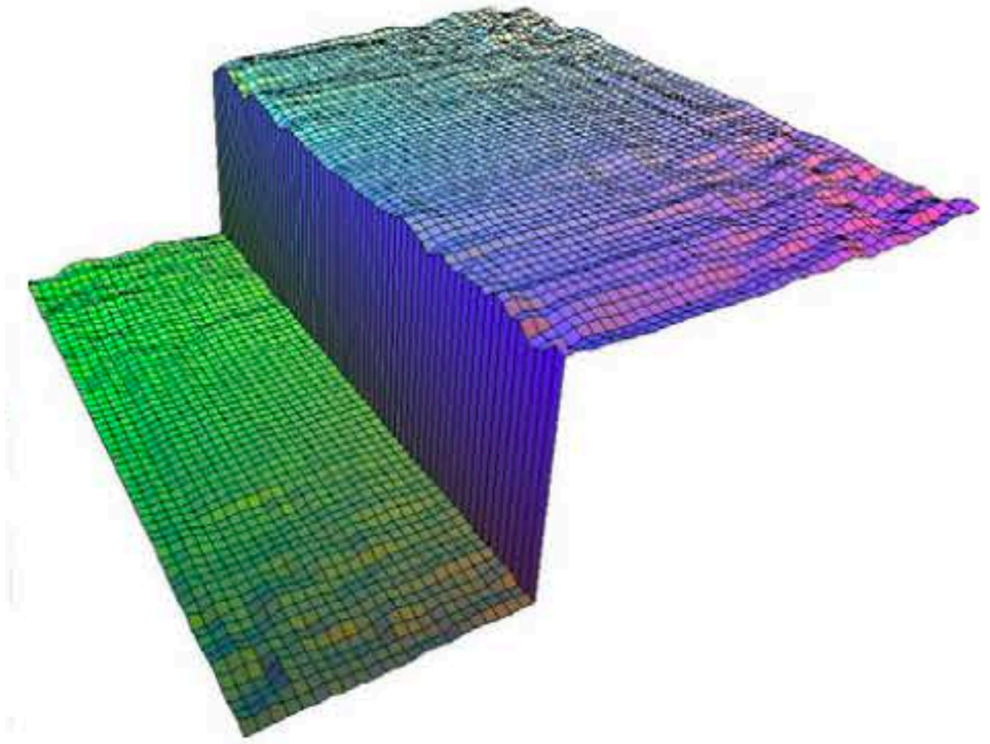
Input



Range Kernel Influence



Bilateral Filter
(domain * range)



Output

Images from: Durand and Dorsey, 2002

Bilateral Filter Application: Denoising

Noisy Image

Gaussian Filter

Bilateral Filter

Slide Credit: Alexander Wong

Bilateral Filter Application: Cartooning

Original Image

After 5 iterations of **Bilateral** Filter

Slide Credit: Alexander Wong

Bilateral Filter Application: Flash Photography

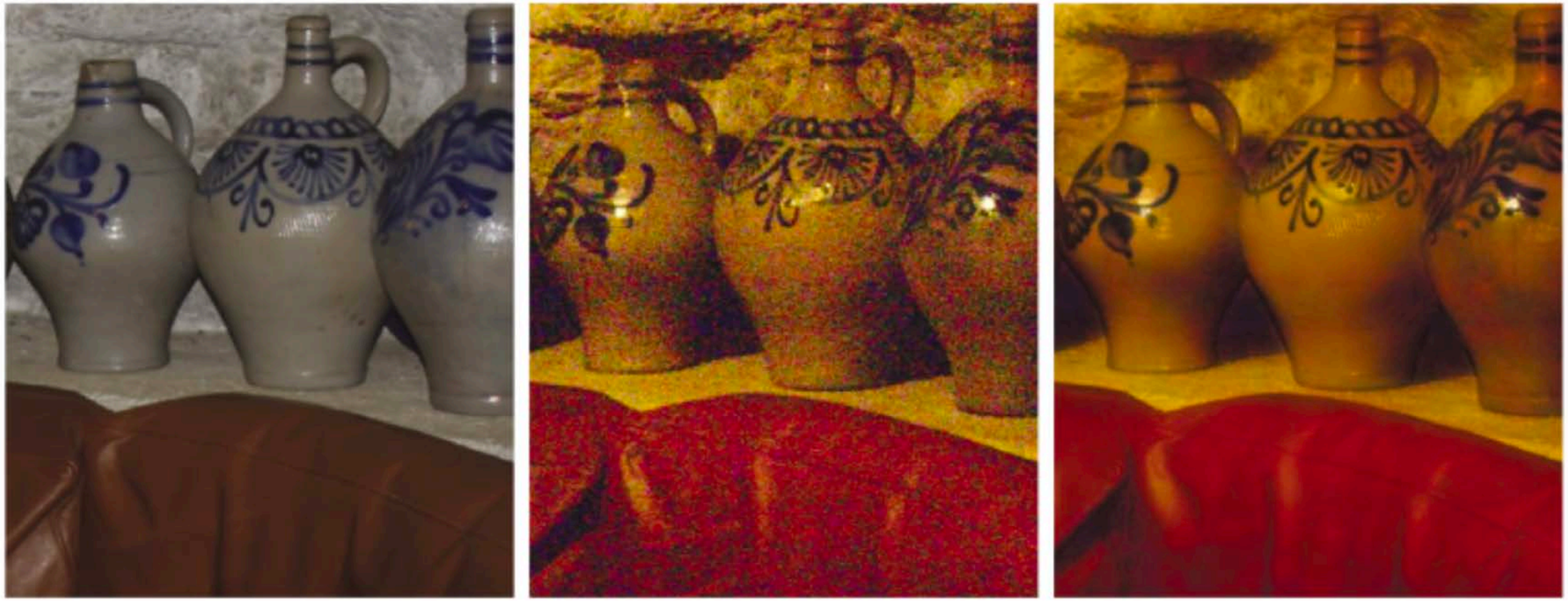
Non-flash images taken under low light conditions often suffer from excessive **noise** and **blur**

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:

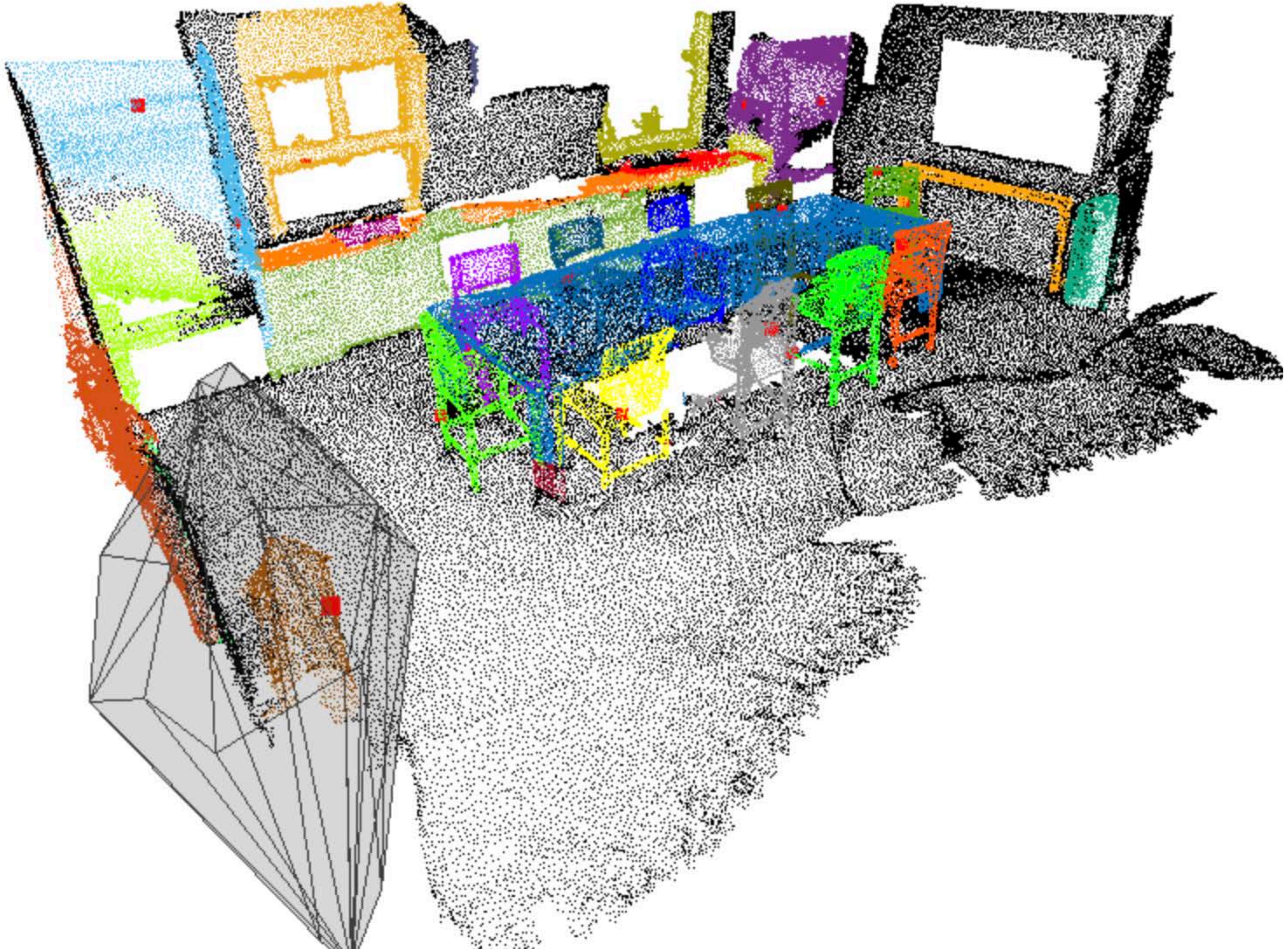


Flash   No-Flash   Detail Transfer with Denoising

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

Figure Credit: Petschnigg et al., 2004

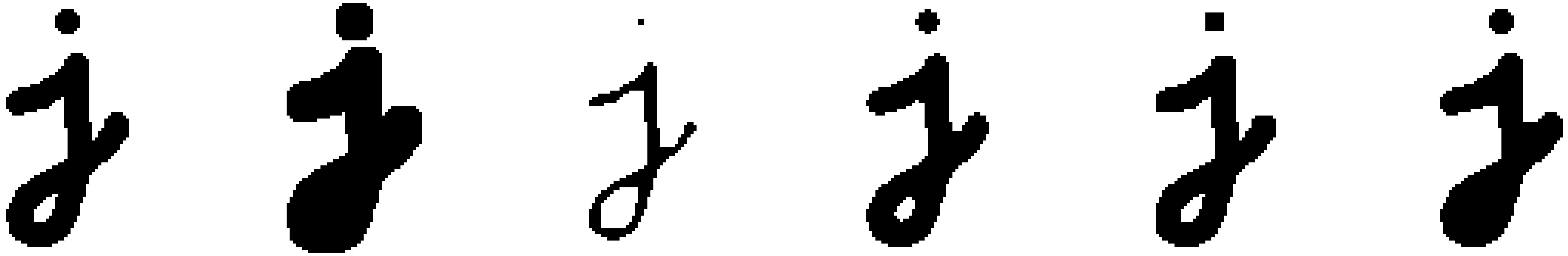
Bilateral Filter: “Modern” take



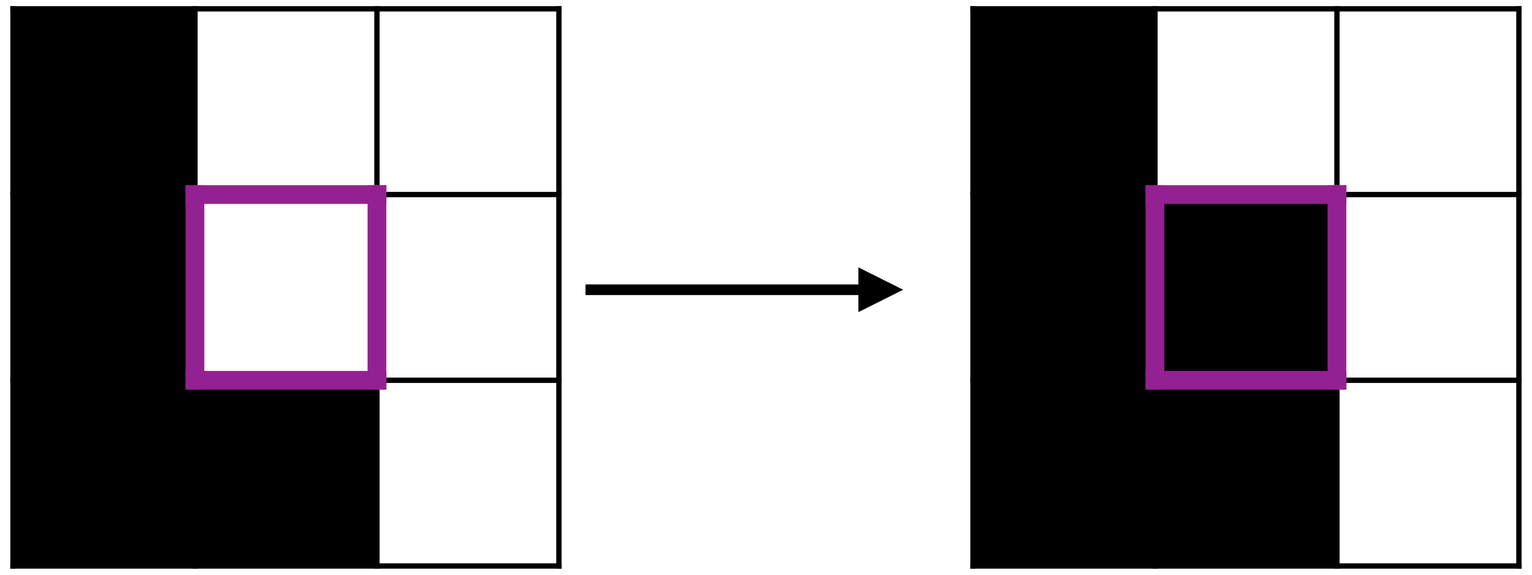
https://neuralbf.github.io/

Morphology

Optional subtitle



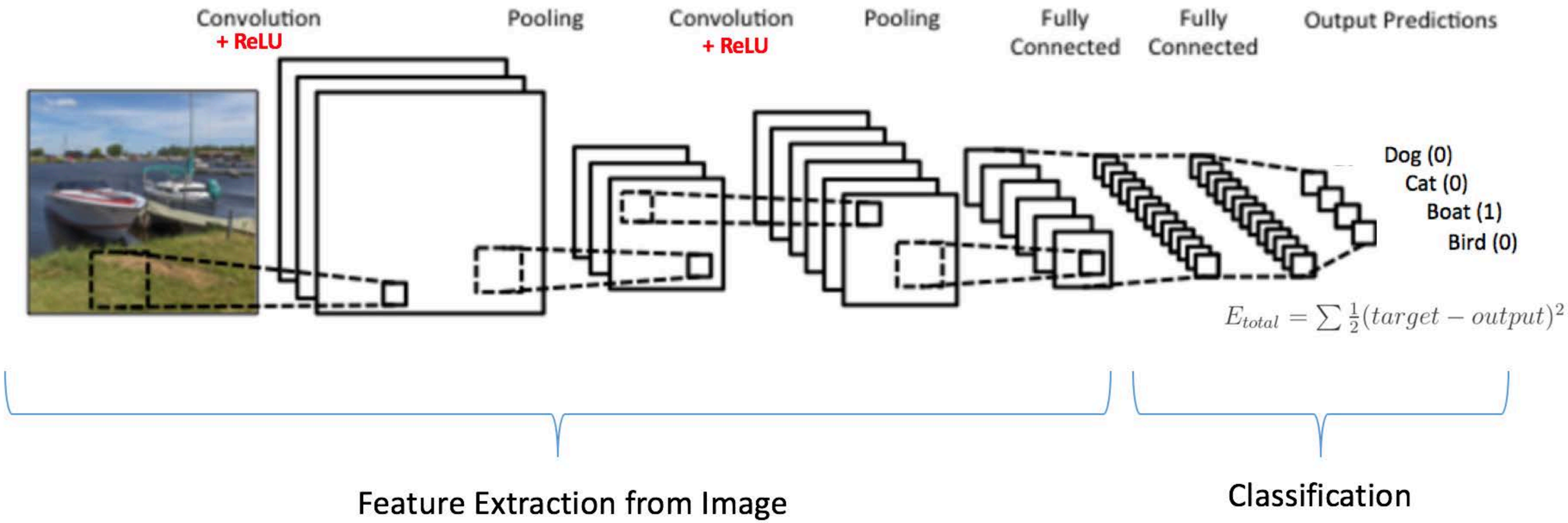
original dilate erode majority open close



Threshold function in local structuring element

close(.) = erode(dilate(.)) etc., see Szeliski 3.3.2

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

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

Speeding-up Convolution can be achieved using separable filters or Fourier Transforms if the filter and image are both large

Fourier Transforms give us a way to think about image processing operations in Frequency Space, e.g., low pass filter = removing high frequency components