CSC496: Deep learning in computer vision

Prof. Bei Xiao

Lecture 5: Signal processing

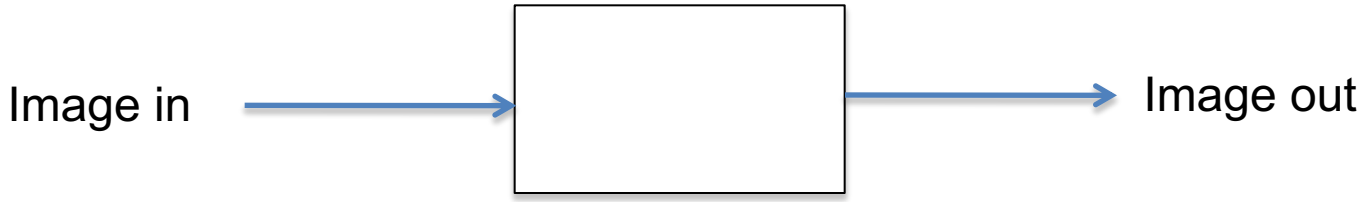
Today's class

- Linear Filters
- Image Convolution
- Fourier transform

- First in-class quiz: Monday, September 16th

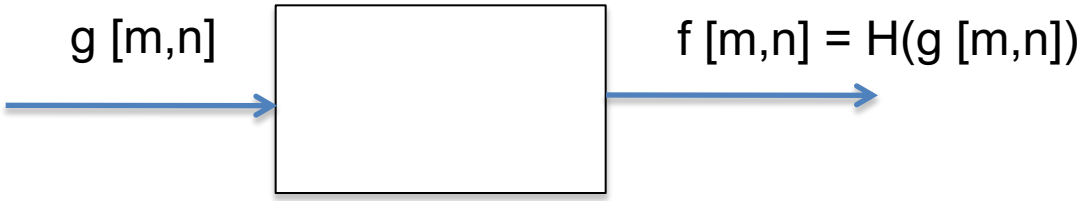
- Homework 2: Pin hole camera. Out September 15th, Due September 29th.

Filtering



We want to remove unwanted sources of variation, and keep the information relevant for whatever task we need to solve

Linear filtering

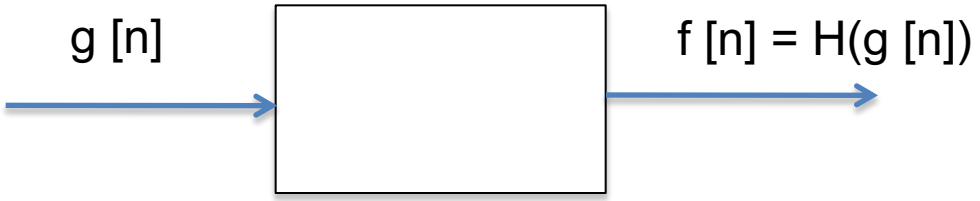


For a filter to be linear, it has to verify:

$$f[m,n] = H(a[m,n] + b[m,n]) = H(a[m,n]) + H(b[m,n])$$

$$f[m,n] = H(C\,a[m,n]) = C\,H(a[m,n])$$

Linear filtering



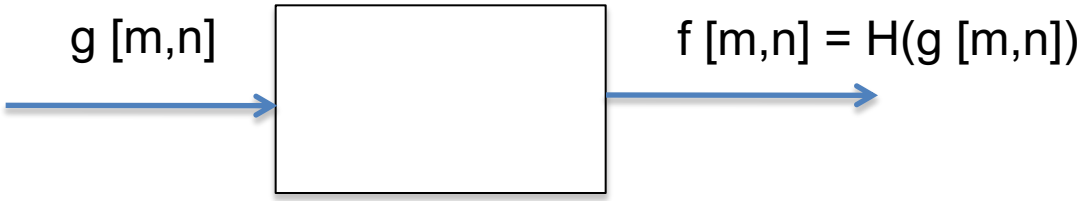
A linear filter in its most general form can be written as,
in 1D for a signal of length N:

$$f[n] = \sum_{k=0}^{N-1} h[n,k]\, g[k]$$

It is useful to make it more explicit by writing:

$$\begin{bmatrix} f[0] \\ f[1] \\ \vdots \\ f[M] \end{bmatrix} = \begin{bmatrix} h[0,0] & h[0,1] & \dots & h[0,N] \\ h[1,0] & h[1,1] & \dots & h[1,N] \\ \vdots & \vdots & \vdots & \vdots \\ h[M,N] & h[M,1] & \dots & h[M,N] \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ \vdots \\ g[N] \end{bmatrix}$$

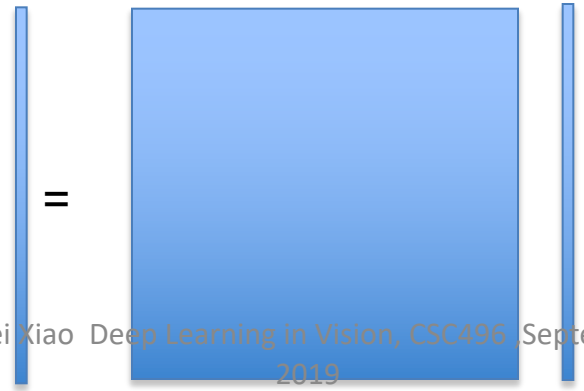
Linear filtering



In 2D:

$$f[n,m] = \sum_{k,\,l=0}^{N-1,\,M-1} h[n,m,k,l]\, g[k,l]$$

Which can also be written in matrix form as in the 1D case:



Credit picture: Fredo Durand

$$\begin{bmatrix} f[0] \\ f[1] \\ \vdots \\ f[M] \end{bmatrix} = \begin{bmatrix} h[0,0] & h[0,1] & \dots & h[0,N] \\ h[1,0] & h[1,1] & \dots & h[1,N] \\ \vdots & \vdots & \vdots & \vdots \\ h[M,N] & h[M,1] & \dots & h[M,N] \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ \vdots \\ g[N] \end{bmatrix}$$

Why should one pixel be treated differently than any another?

Credit picture: Fredo Durand

Translation Invariance

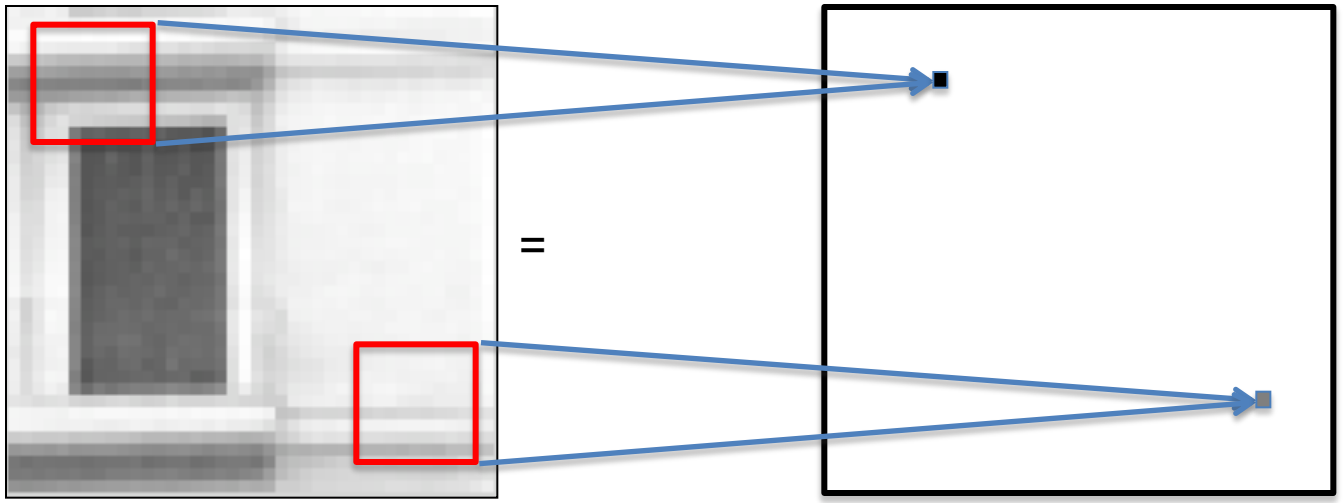
Rotation/Viewpoint Invariance

Size Invariance

Illumination Invariance

Matt Krause
mattkrau.se

A translation invariant filter

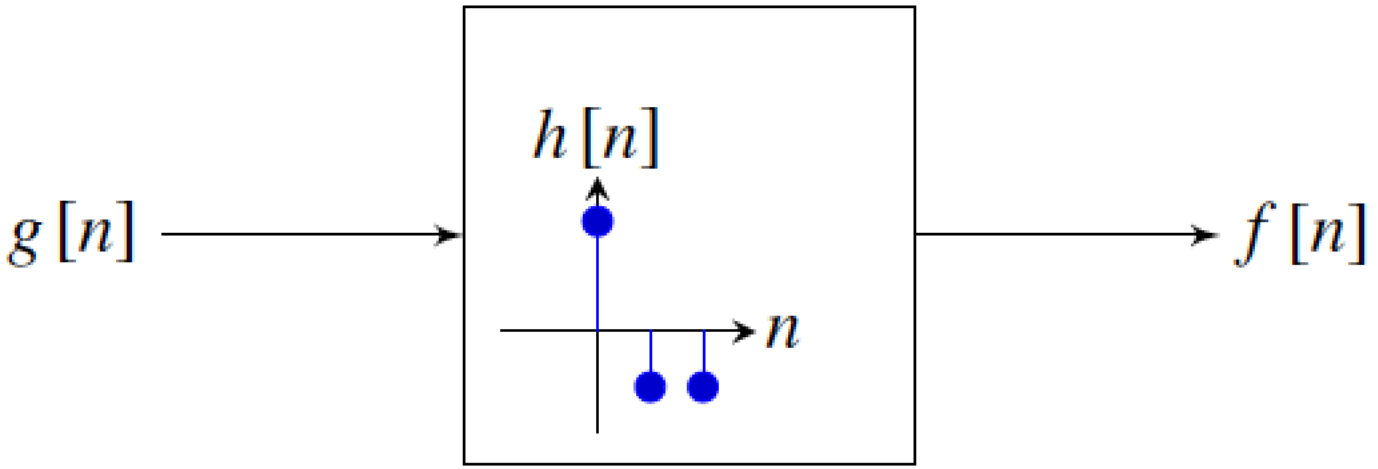


The same weighting occurs within each window

Convolution

$$f[n] = h \circ g = \sum_{k=0}^{N-1} h[n-k]\, g[k]$$

For example: h = [2, -1, -1]

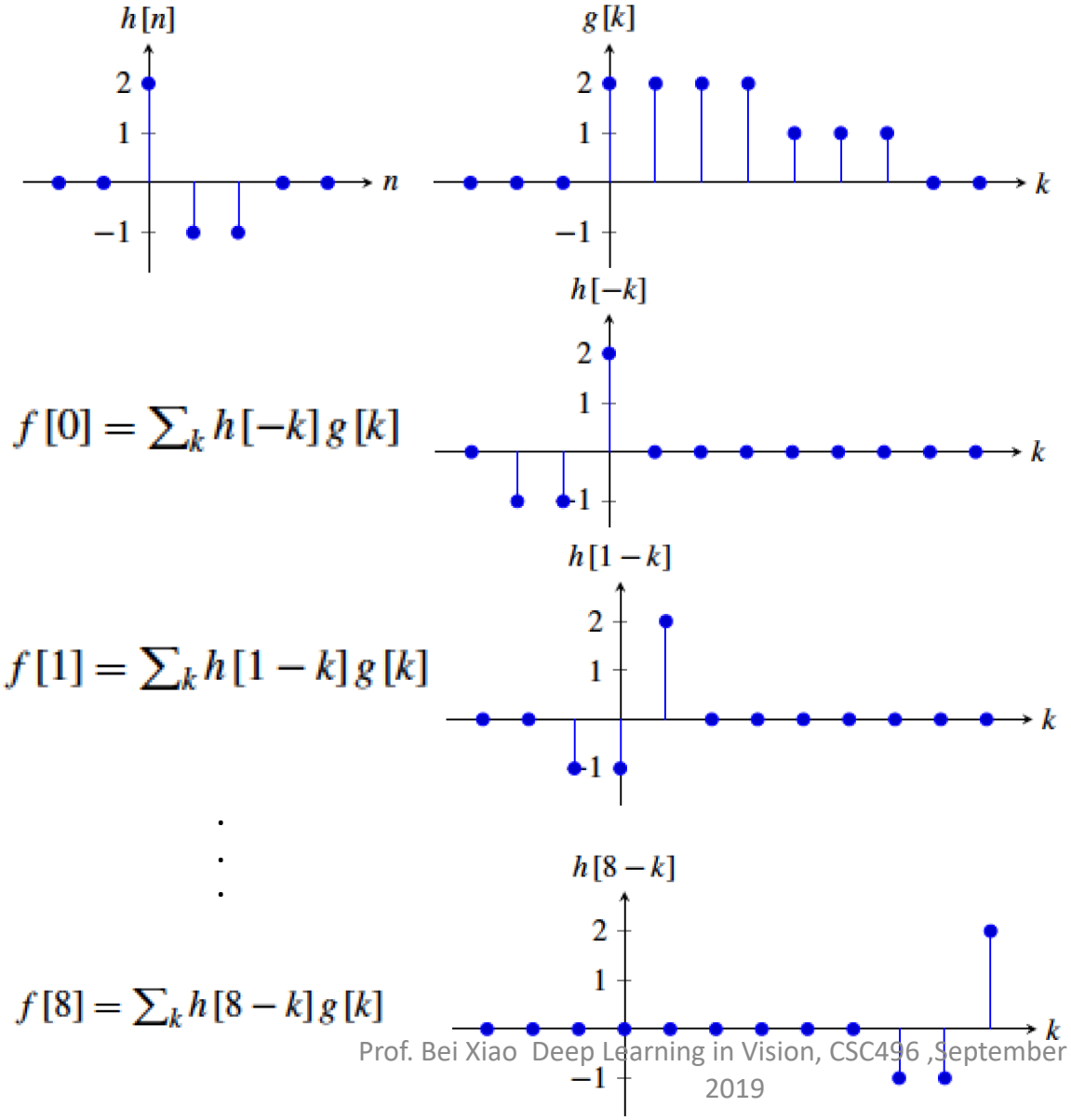


Convolution

In the 1D case, it helps to make explicit the structure of the matrix:

$$\begin{bmatrix} f[0] \\ f[1] \\ f[2] \\ \vdots \\ f[N] \end{bmatrix} = \begin{bmatrix} h[0] & h[-1] & h[-2] & \dots & h[-N] \\ h[1] & h[0] & h[-1] & \dots & h[1-N] \\ h[2] & h[1] & h[0] & \dots & h[2-N] \\ \vdots & \vdots & \vdots & \vdots & \\ h[N] & h[N-1] & h[N-2] & \dots & h[0] \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ g[2] \\ \vdots \\ g[N] \end{bmatrix}$$

Convolution



$f[0] = \sum_k h[-k]\, g[k]$

$f[1] = \sum_k h[1-k]\, g[k]$

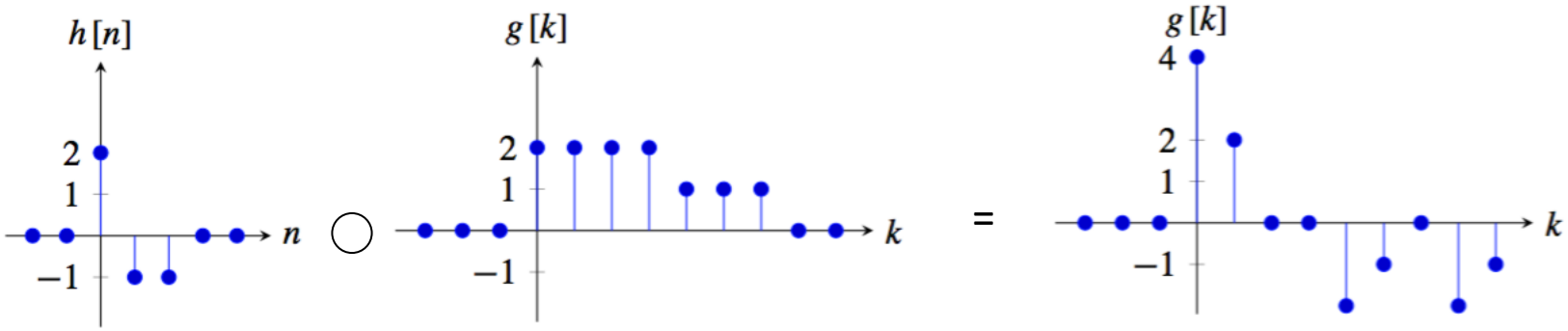
$f[8] = \sum_k h[8-k]\, g[k]$

$f[0] = 4$

$f[1] = 2$

$f[8] = -1$

Convolution



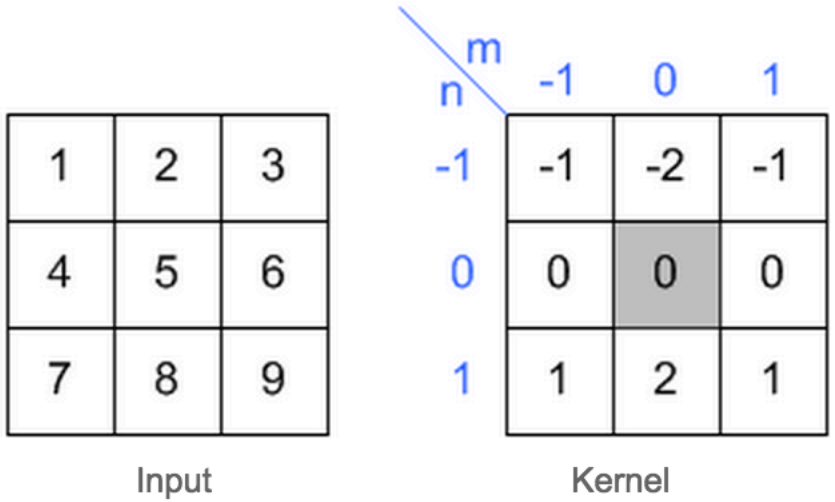
2D convolution

$*$

1	0	-1
2	0	-2
1	0	-1

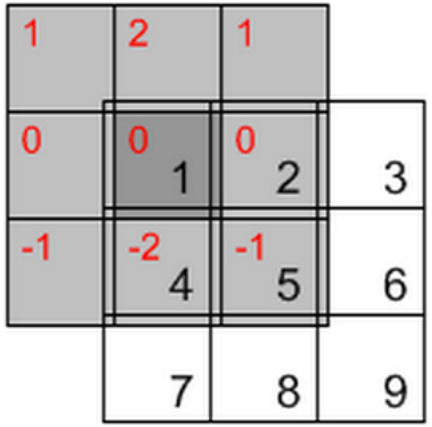
Let’s take out paper and pen

- What is the result of the following convolution?



Let’s take out a paper and a pen

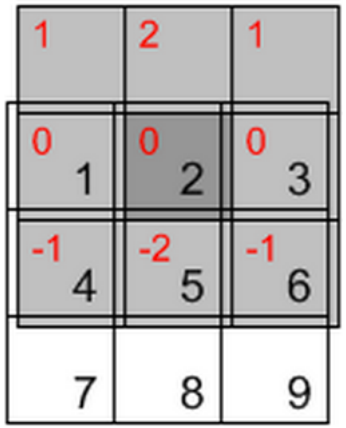
- What is the result of the following convolution?



1*0 + 2*0 + 1*0+
0*0 + 1*0 + 2*0+
(-1)*0 + (-2)*4 +(-1)*5 = -13

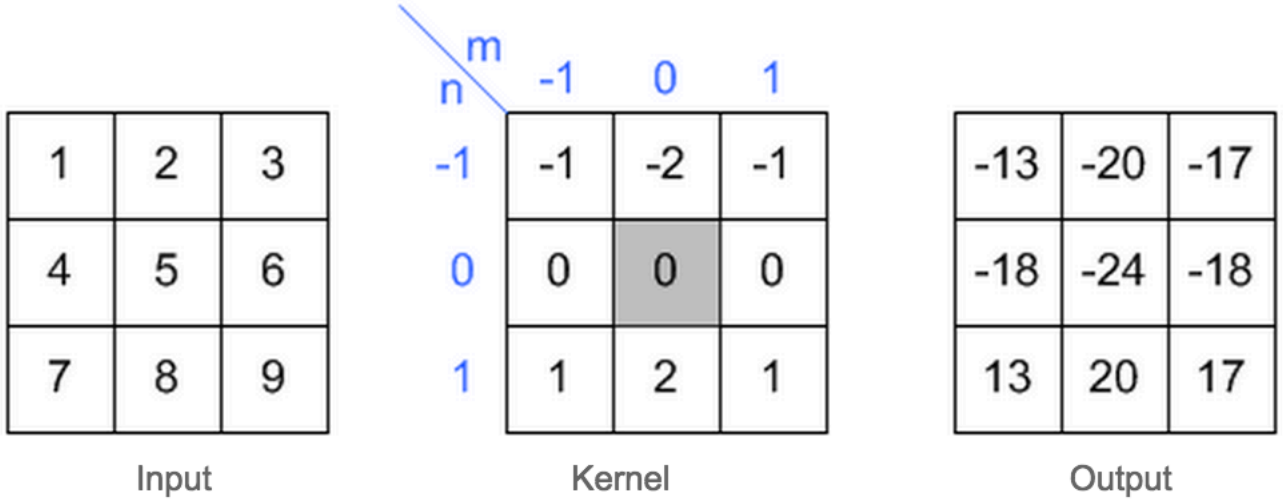
Let’s take out a paper and a pen

- What is the result of the following convolution?



Let’s take out paper and pen

- What is the result of the following convolution?

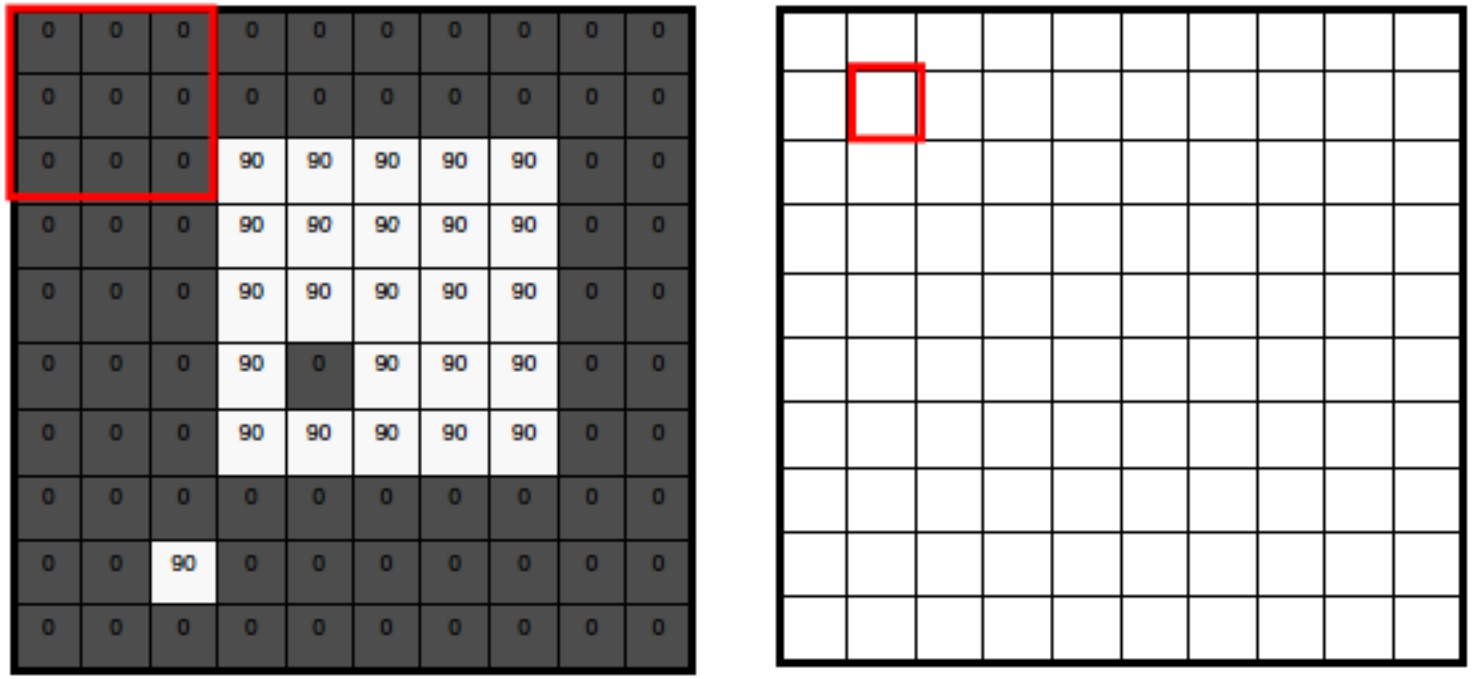


Check it with Python?

```
import numpy as np
from scipy import signal

# use np.array to create the matrix

# create the filter

# use signal.convolve2d
```

2D convolution

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



$$f[m,n] = h \circ g = \sum_{k,l} h[m-k, n-l]\, g[k,l]$$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

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

	0	10							

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

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

	0	10	20						

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

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

	0	10	20	30					

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

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

	0	10	20	30	30				

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

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

	0	10	20	30	30				
				?					

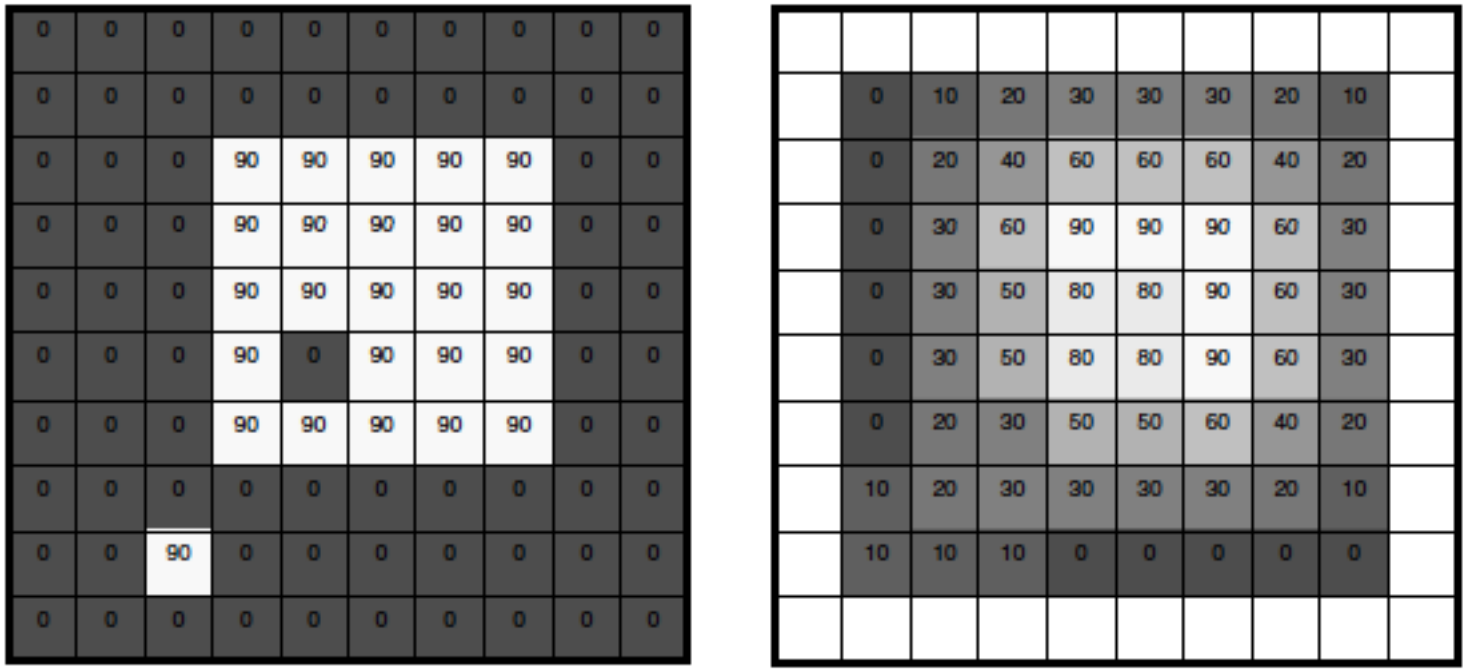
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

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

	0	10	20	30	30				
						?			
				50					

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



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

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

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

2D convolution

$$f[m,n] = h \circ g = \sum_{k,l} h[m-k, n-l]\, g[k,l]$$

m=0 1 2 …

111	115	113	111	112	111	112	111
135	138	137	139	145	146	149	147
163	168	188	196	206	202	206	207
180	184	206	219	202	200	195	193
189	193	214	216	104	79	83	77
191	201	217	220	103	59	60	68
195	205	216	222	113	68	69	83
199	203	223	228	108	68	71	77

g[m,n]

$\otimes$

-1	2	-1
-1	2	-1
-1	2	-1

h[m,n]

=

?	?	?	?	?	?	?	?
?	-5	9	-9	21	-12	10	?
?	-29	18	24	4	-7	5	?
?	-50	40	142	-88	-34	10	?
?	-41	41	264	-175	-71	0	?
?	-24	37	349	-224	-120	-10	?
?	-23	33	360	-217	-134	-23	?
?	?	?	?	?	?	?	?

f[m,n]

Properties of the convolution

Commutative

$$h[n] \circ g[n] = g[n] \circ h[n]$$

Associative

$$h[n] \circ g[n] \circ q[n] = h[n] \circ (g[n] \circ q[n]) = (h[n] \circ g[n]) \circ q[n]$$

Distributive with respect to the sum

$$h[n] \circ (f[n] + g[n]) = h[n] \circ f[n] + h[n] \circ g[n]$$

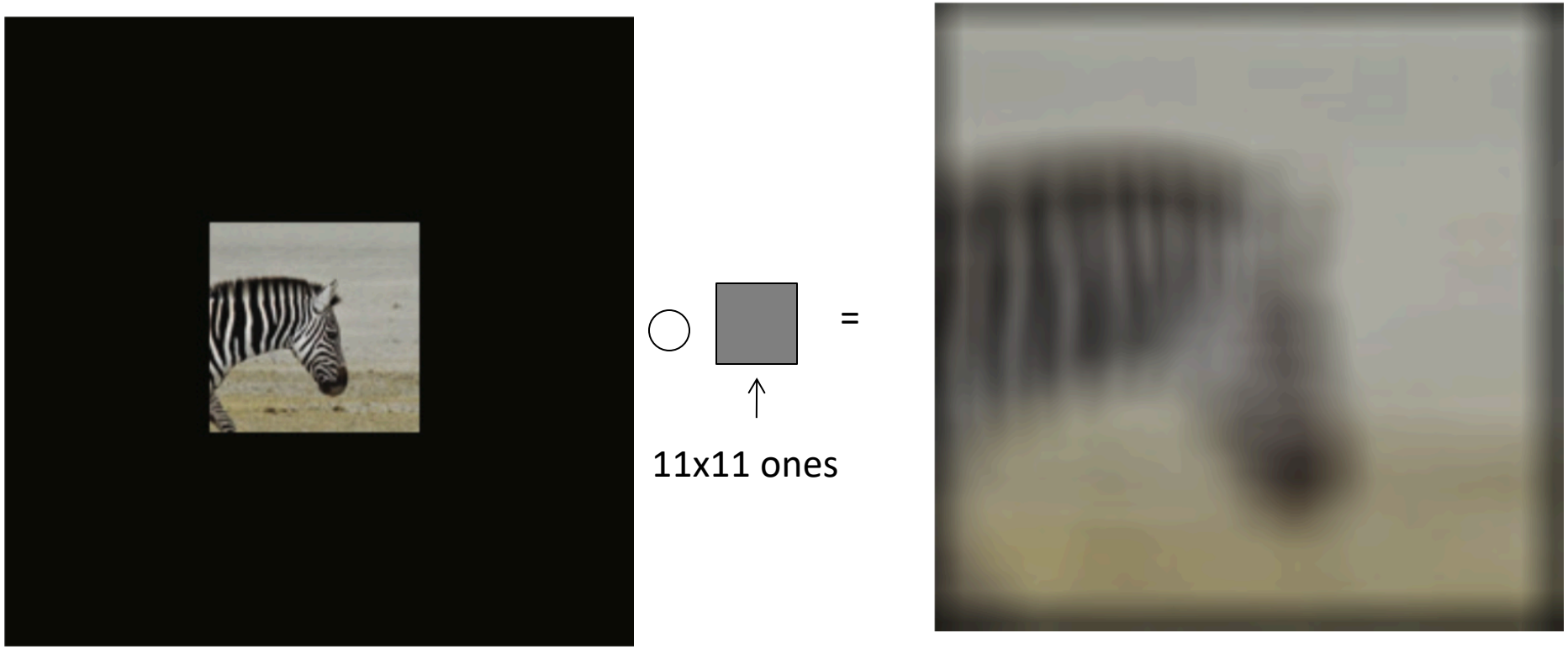
Shift property

$$f[n - n_0] = h[n] \circ g[n - n_0] = h[n - n_0] \circ g[n]$$

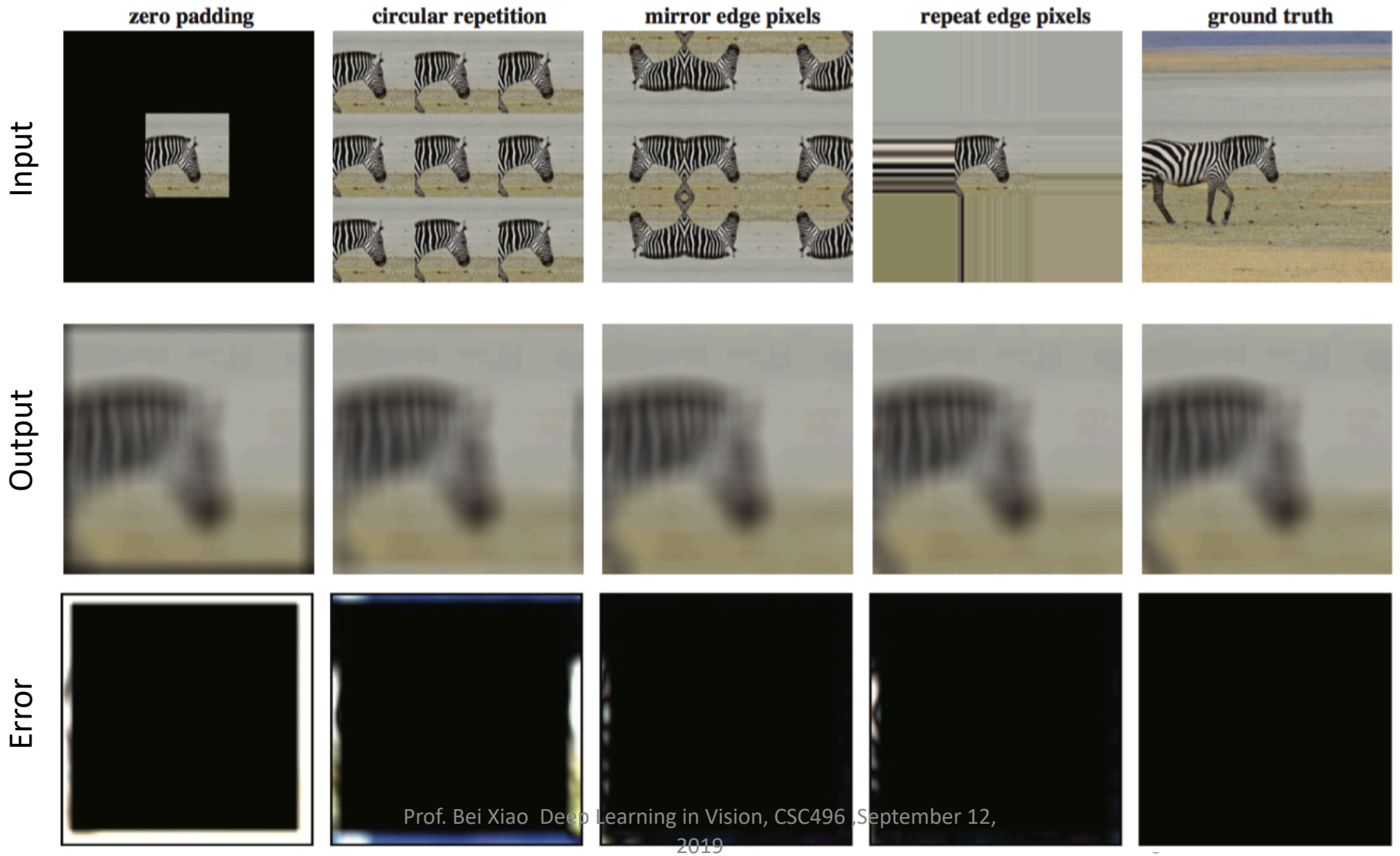
Handling boundaries

Handling boundaries

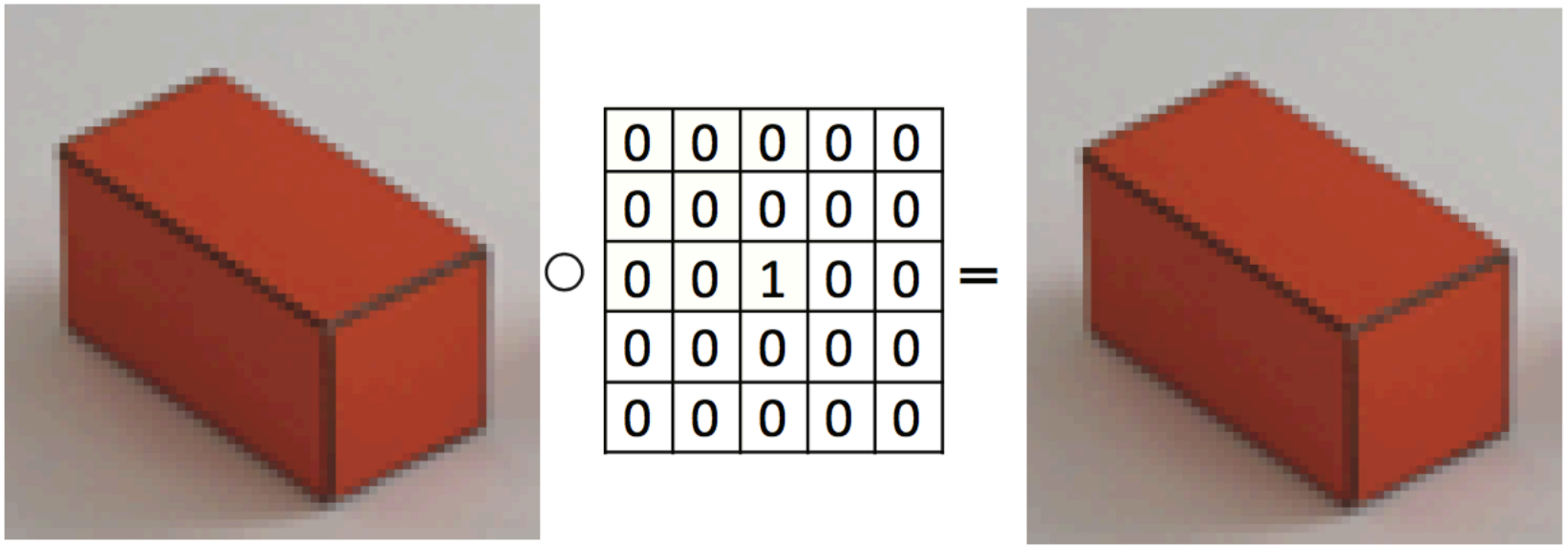
Zero padding



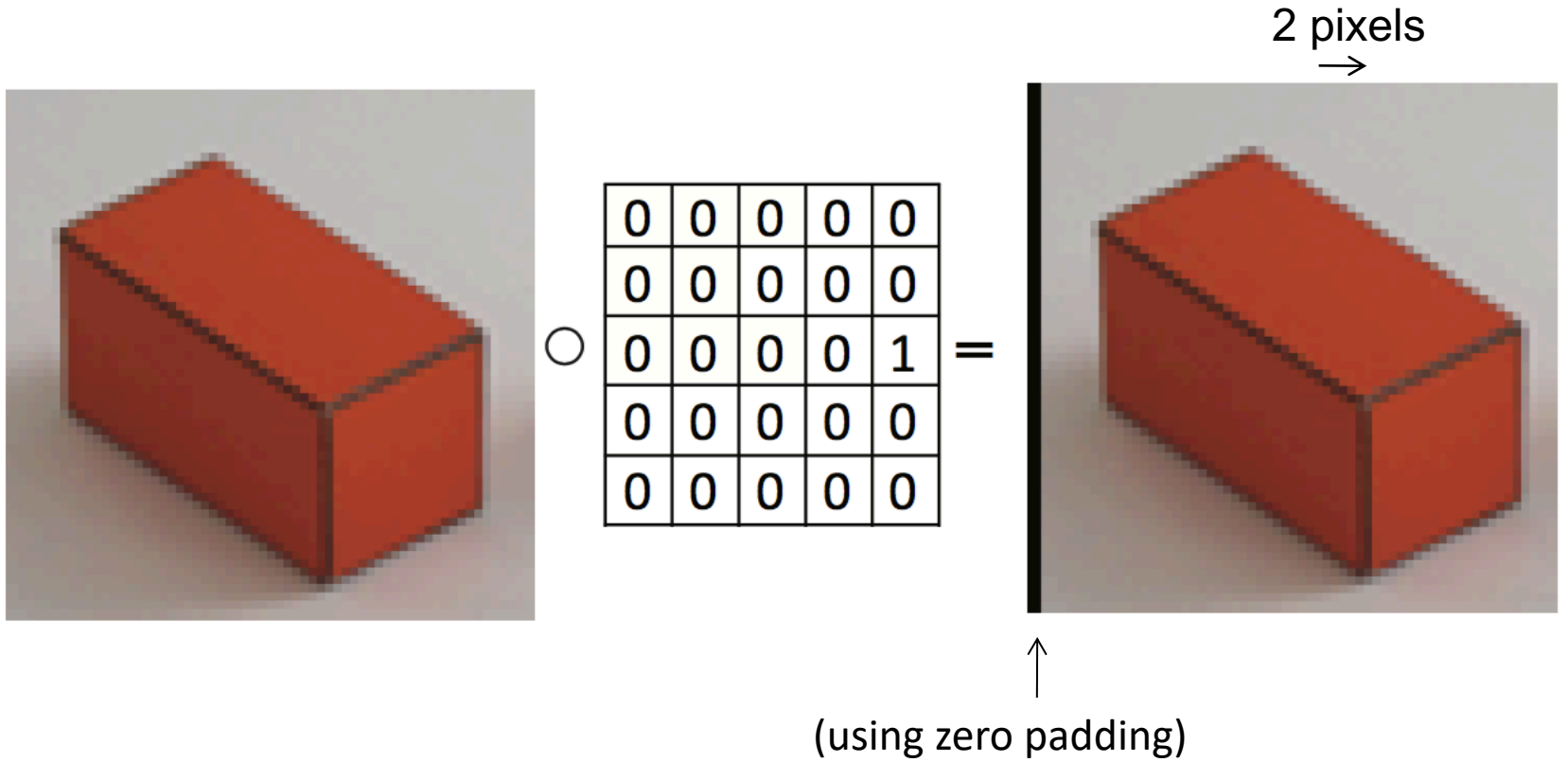
Handling boundaries



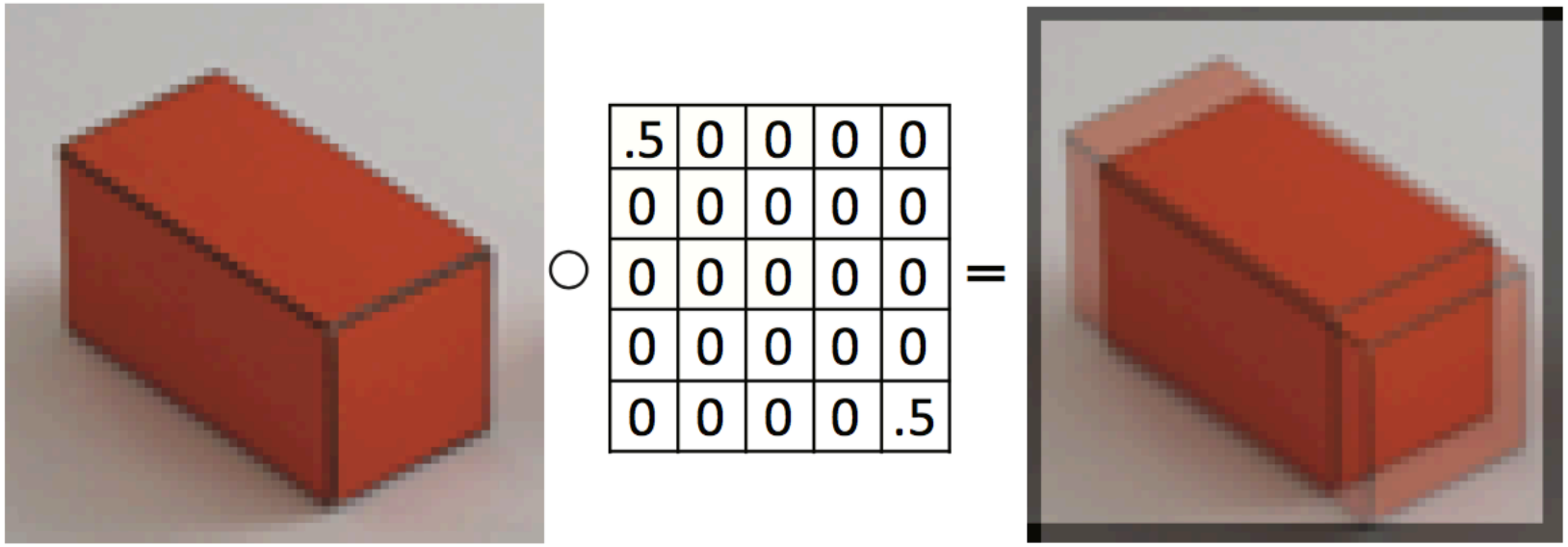
Examples



Examples



Examples



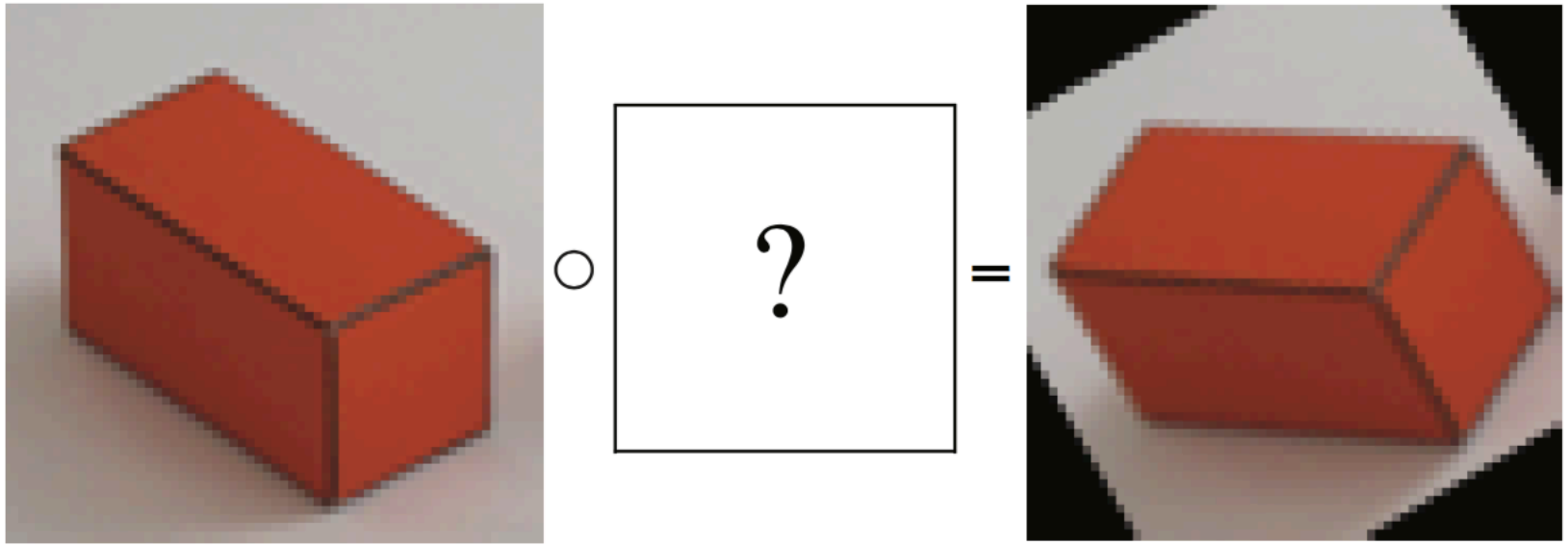
Take-home exercise

Take the simple vision image as an example

Apply convolution of various filters on them using python

Show the results

For example, what is the kernel? Will check your answer next time.



Rectangular filter

g[m,n]

$\otimes$

h[m,n]

=

f[m,n]

Rectangular filter

g[m,n]

$\otimes$

h[m,n]

=

f[m,n]

Rectangular filter

g[m,n]

$\otimes$

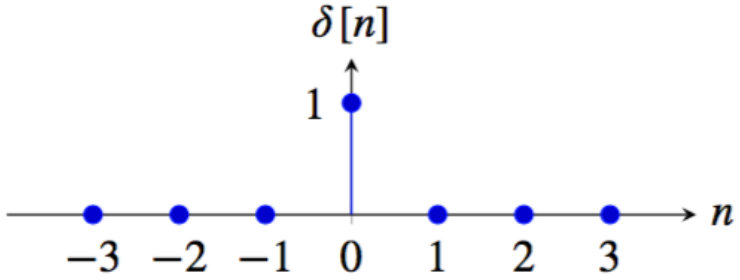
h[m,n]

=

f[m,n]

Important signals

The impulse



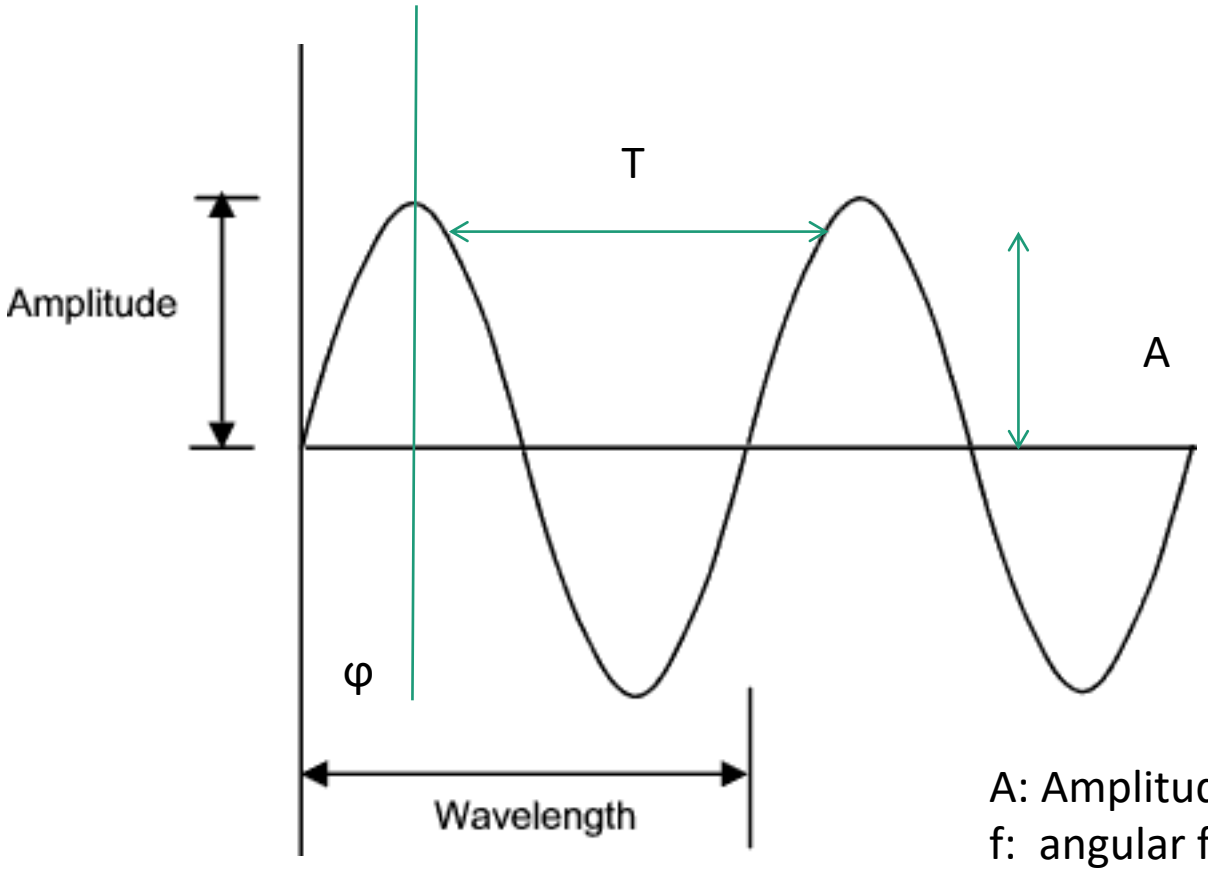
The result of convolving a signal *g*[n] with the impulse signal is the same signal:

$$f[n] = \delta \circ g = \sum_k \delta[n-k]\, g[k] = g[n]$$

Convolving a signal *f* with a translated impulse δ [*n* − *n*0] results in a translated signal:

$$f[n-n_0] = \delta[n-n_0] \circ f[n]$$

Sine Wave



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

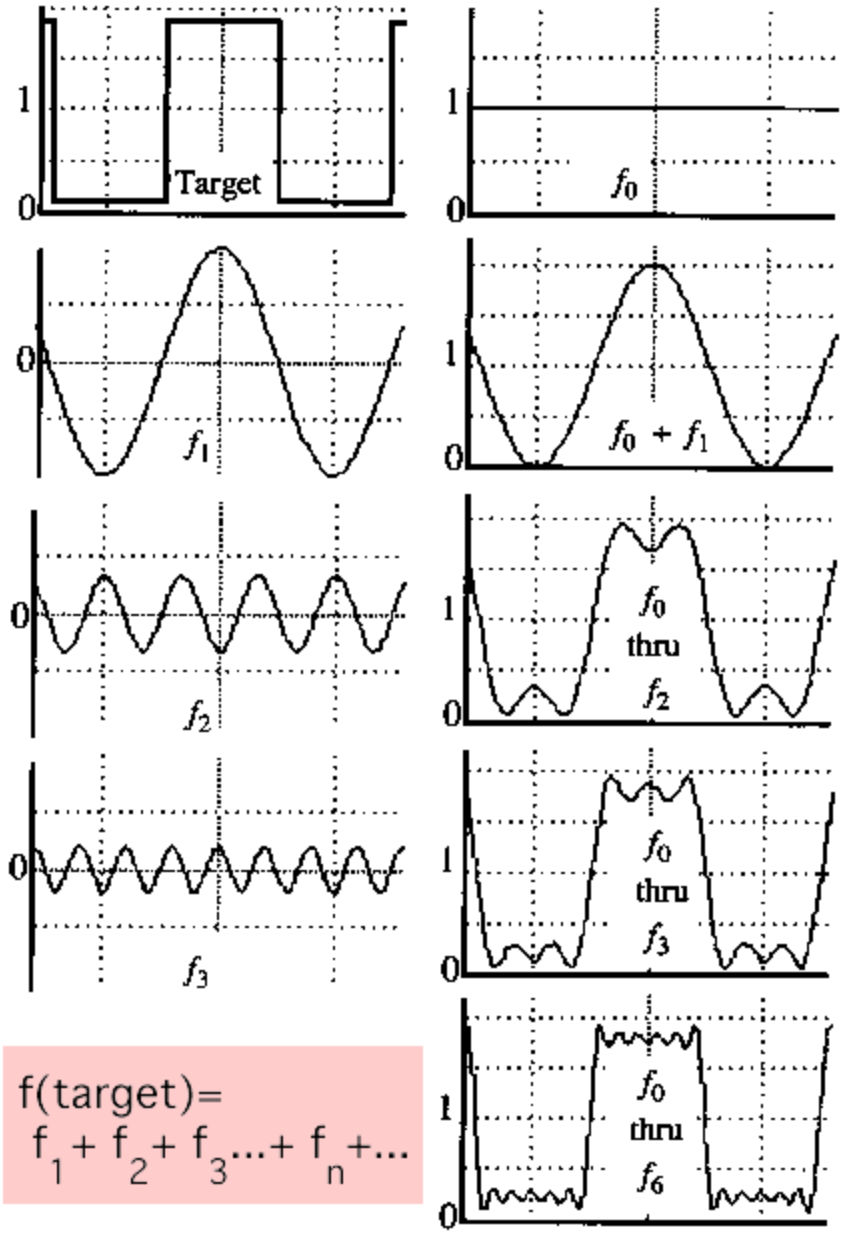
A: Amplitude
f: angular frequency, ω/2π (HZ)
Number of oscillations that occur each second of time
φ: phase

A sum of sines

Our building block:

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

Add enough of them to get any signal *g(x)* you want!



f(target)=
$f_1 + f_2 + f_3 ... + f_n + ...$

Important signals

Cosine and sine waves

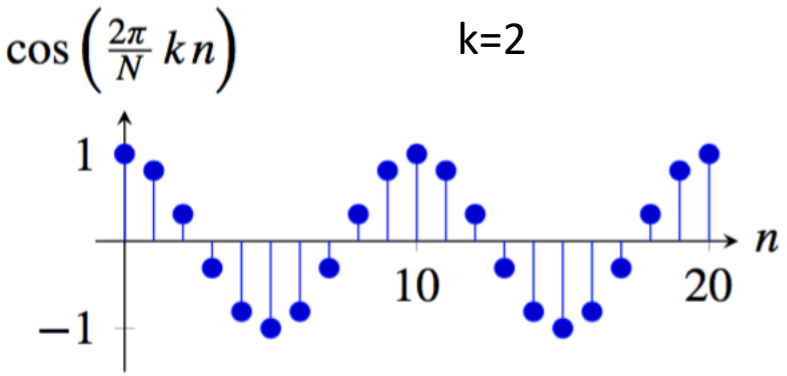
$$s(t) = A \sin(w\,t - \theta)$$

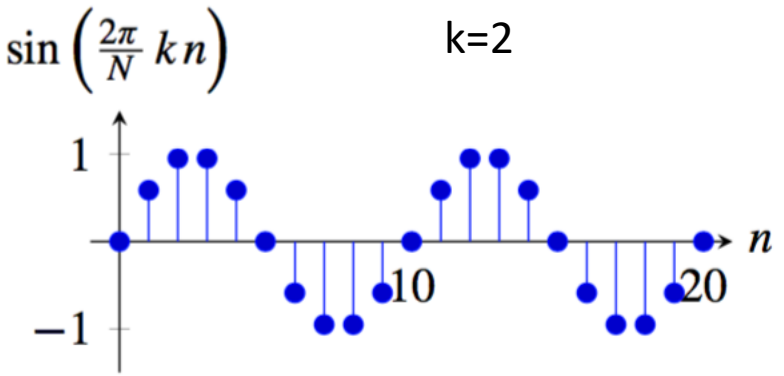
$$s_k[n] = \sin\left(\frac{2\pi}{N} k n\right) \qquad c_k[n] = \cos\left(\frac{2\pi}{N} k n\right)$$

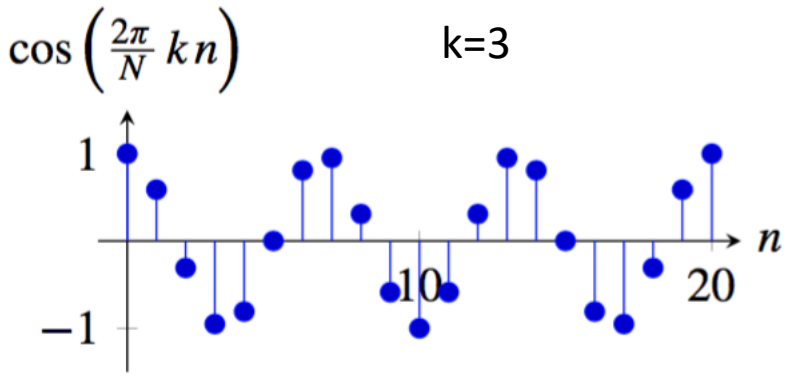
$k \in [1, N/2]$ denotes the number of wave cycles that will occur within the region of support

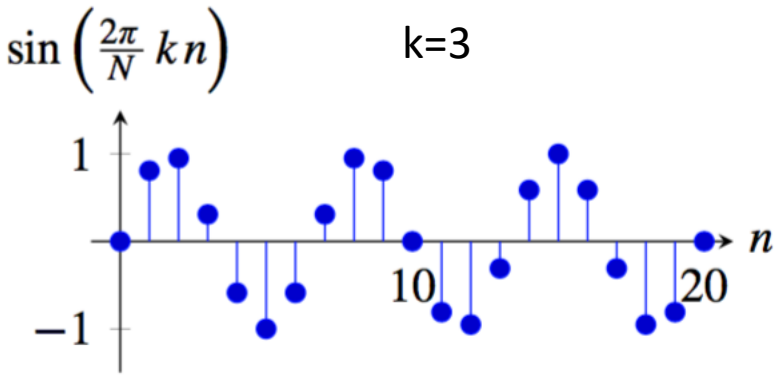
Important signals

Cosine and sine waves

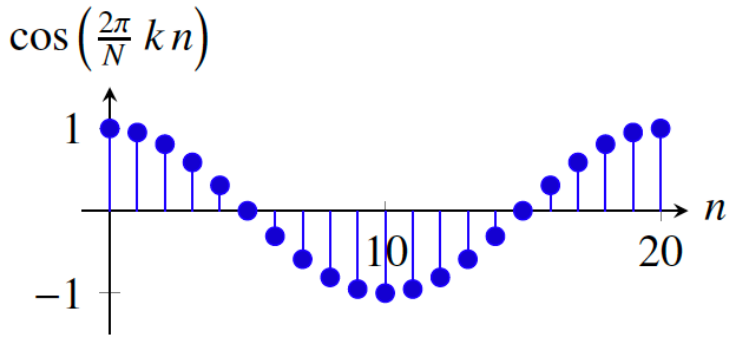




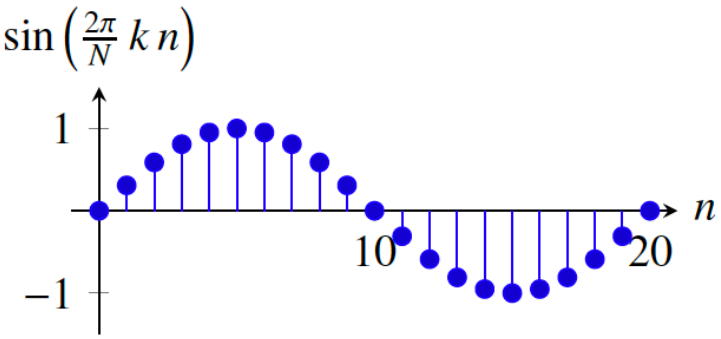




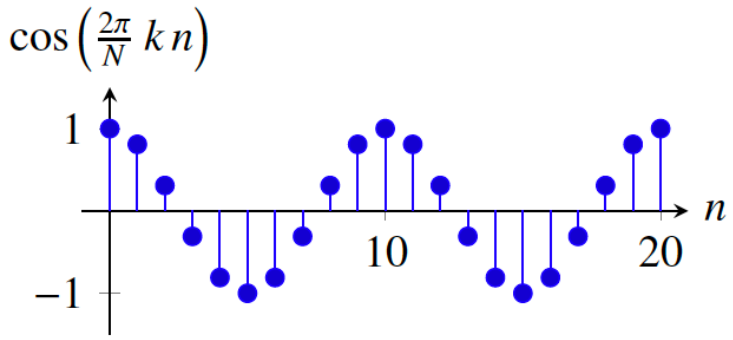
a)


cos (2π/N k n)
1
−1
10
20
n

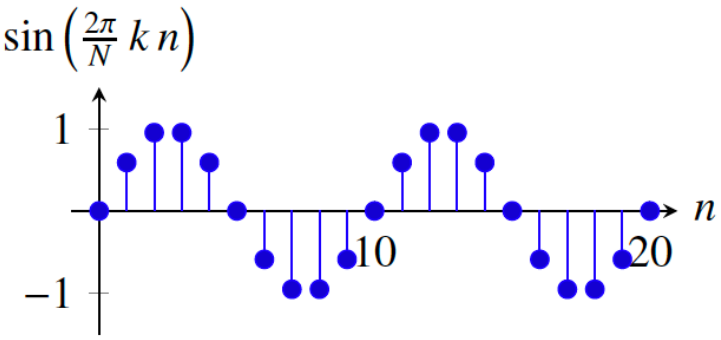
b)


sin (2π/N k n)
1
−1
10
20
n

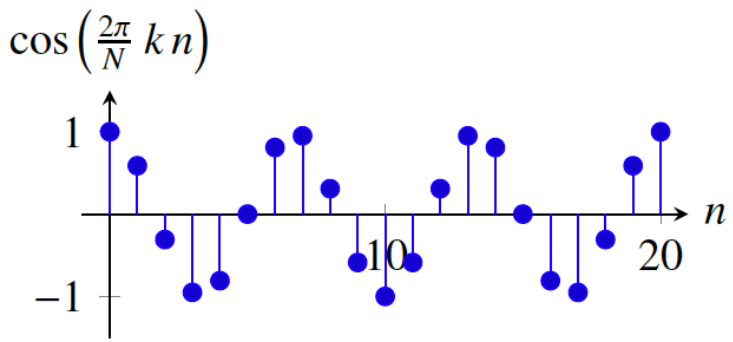
c)


cos (2π/N k n)
1
−1
10
20
n

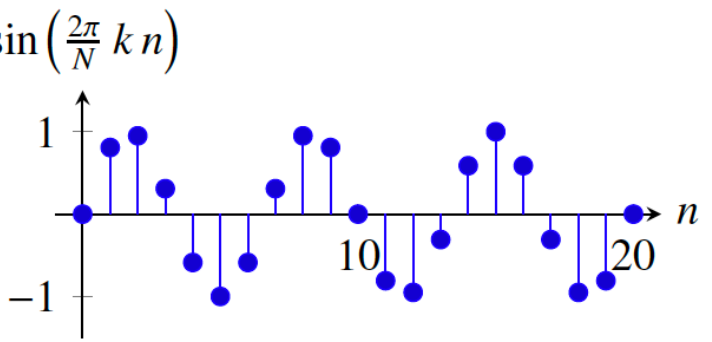
d)


sin (2π/N k n)
1
−1
10
20
n

e)

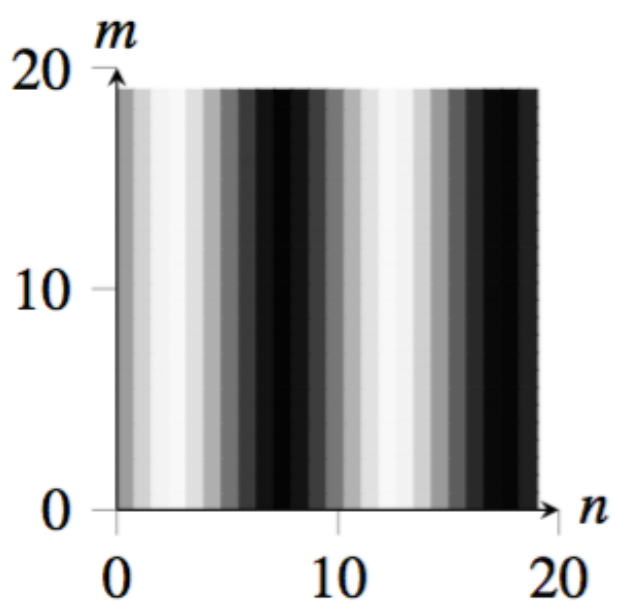

cos (2π/N k n)
1
−1
10
20
n

f)

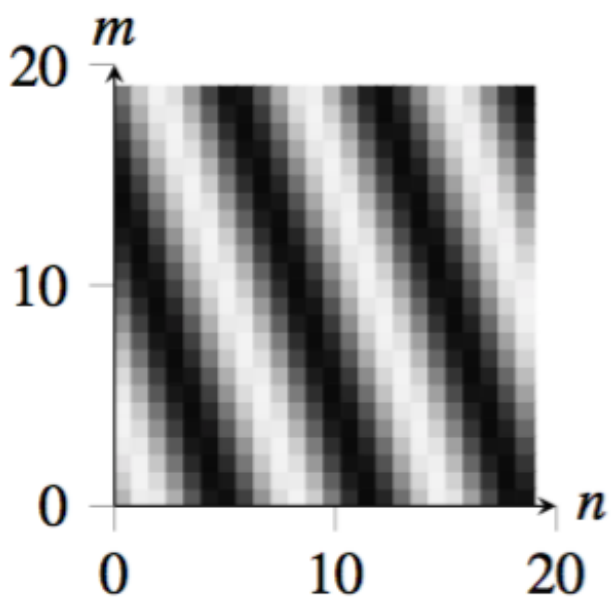

sin (2π/N k n)
1
−1
10
20
n

Waves in 2D

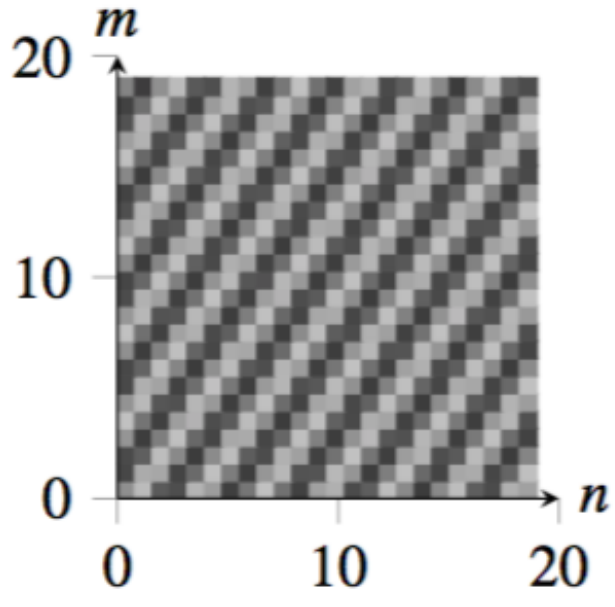
$$s_{u,v}[n,m] = A\sin\left(2\pi\left(\frac{un}{N}+\frac{vm}{M}\right)\right) \qquad c_{u,v}[n,m] = A\cos\left(2\pi\left(\frac{un}{N}+\frac{vm}{M}\right)\right)$$



u = 2, v = 0



u = 3, v = 1



u = 7, v = −5

Important signals

Complex exponential

$$s(t) = A \exp(j w t)$$

Eular's formula

$$e^{ix} = \cos x + i \sin x$$

In discrete time (setting A = 1), we can write:

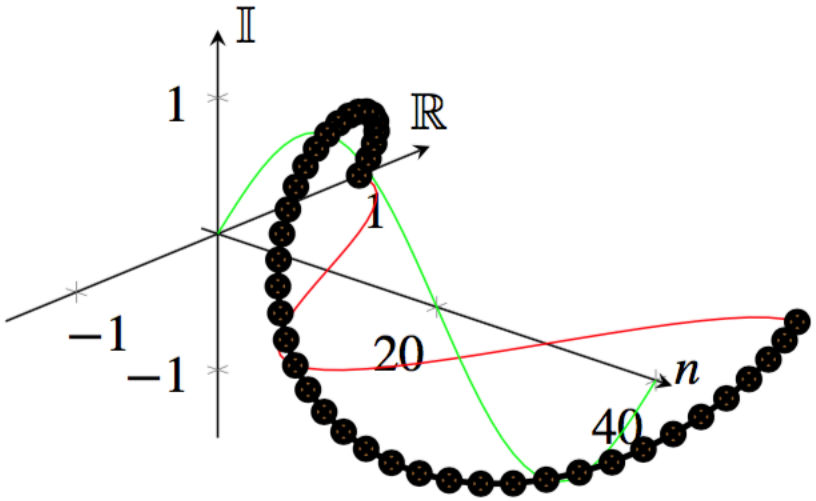
$$e_k[n] = \exp\left(j\frac{2\pi}{N} k n\right) = \cos\left(\frac{2\pi}{N} k n\right) + j \sin\left(\frac{2\pi}{N} k n\right)$$

And in 2D, the complex exponential wave is:

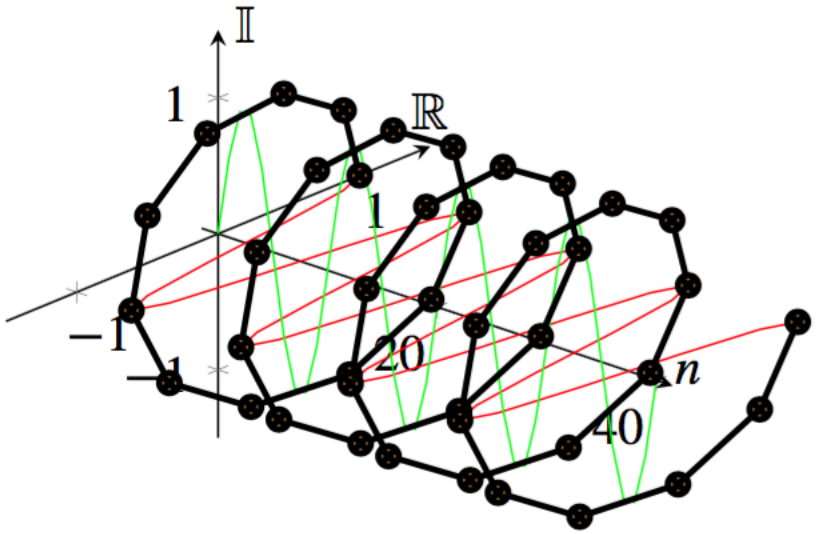
$$e_{u,v}[n,m] = \exp\left(2\pi j \left(\frac{u n}{N} + \frac{v m}{M}\right)\right)$$

Important signals

Complex exponential



$N = 40$, $k = 1$

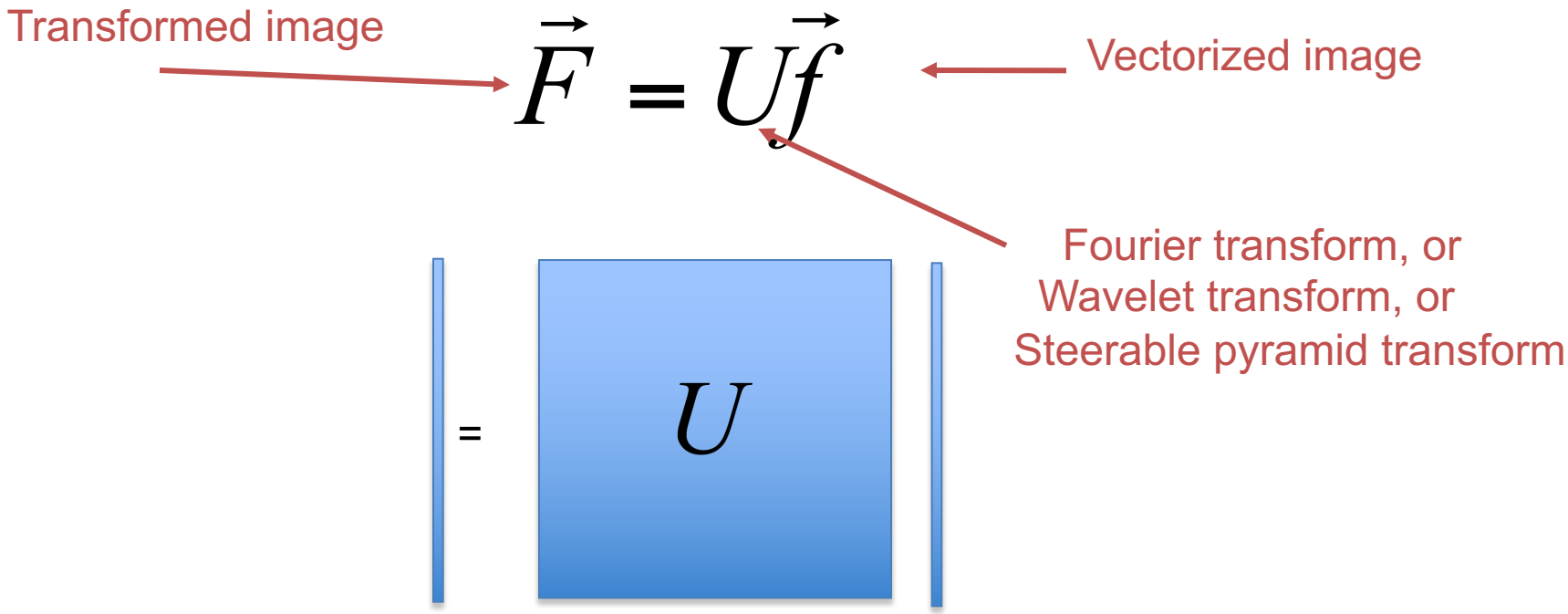


$N = 40$, $k = 3$

Impulses, sine and cosine waves or complex exponentials form each an orthogonal basis for signals of length N

Linear image transformations

- In analyzing images, it's often useful to make a change of basis.



Self-inverting transforms

$$\vec{F} = U\vec{f} \longleftrightarrow \vec{f} = U^{-1}\vec{F}$$

Same basis functions are used for the inverse transform

$$\vec{f} = U^{-1}\vec{F}$$

$$= U^{+}\vec{F}$$

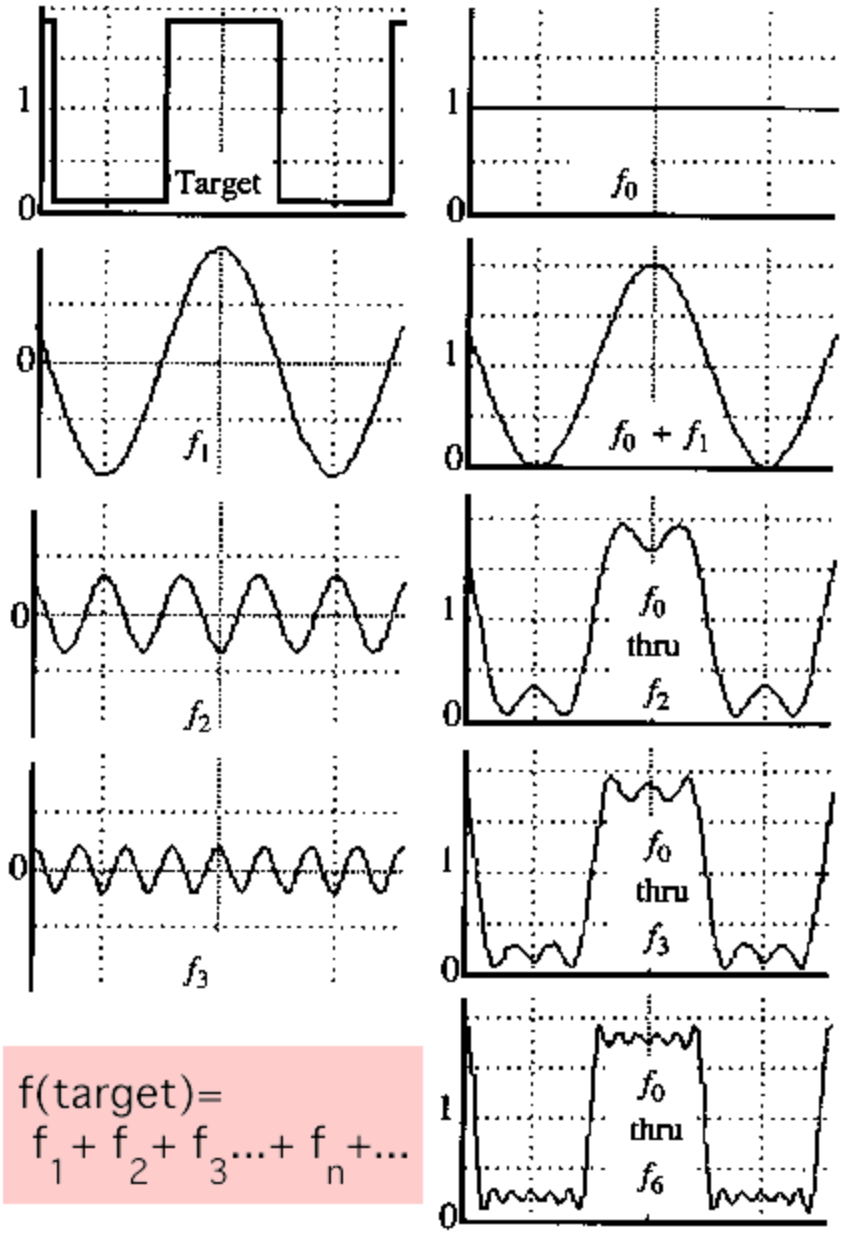
U transpose and complex conjugate

A sum of sines

Our building block:

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

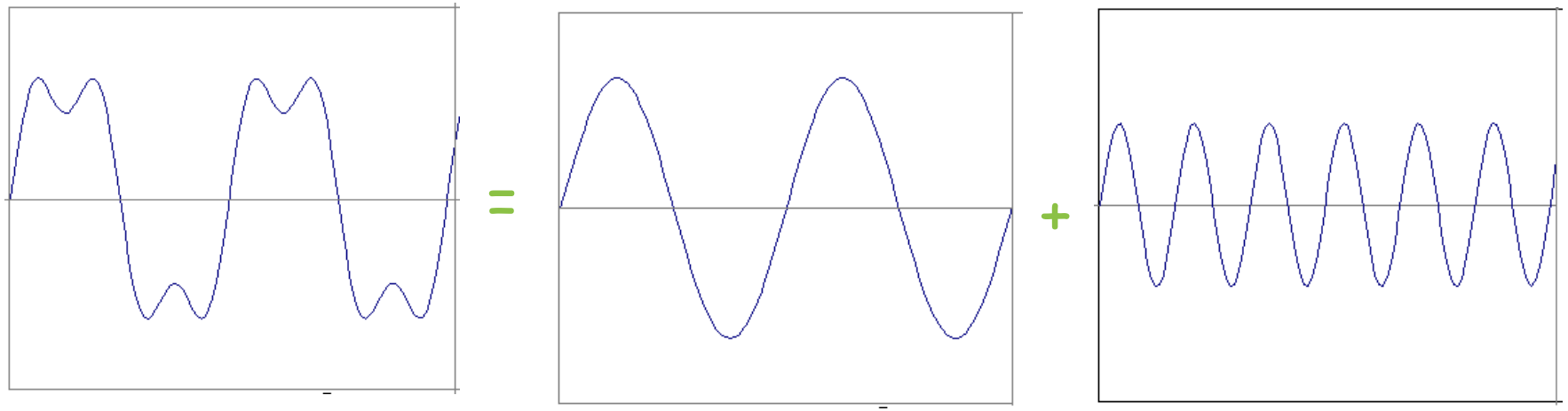
Add enough of them to get any signal *g(x)* you want!

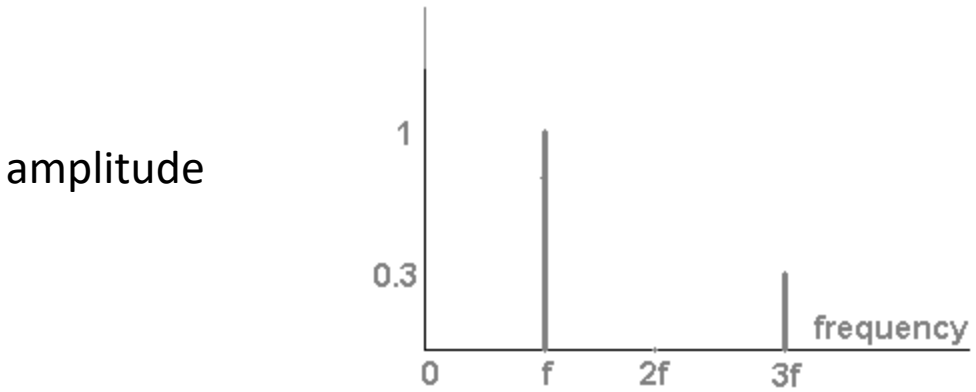


f(target)=
$f_1 + f_2 + f_3 ... + f_n + ...$

Frequency Spectra

- example : $g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi(3f)\, t)$





Slides: Efros

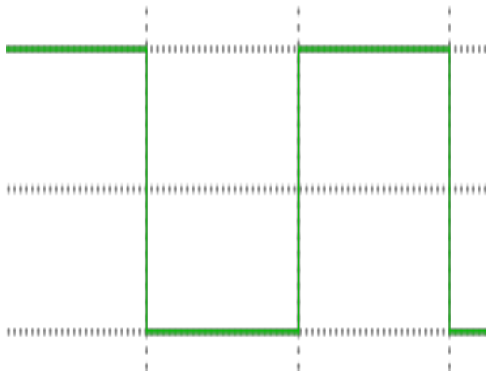
Why decompose signals into sine waves?

- Why don’t we represent the system with some other periodic signals?
- Sine wave is the only waveform that doesn’t change shape when subject to a linear-time-invariant (LTI) system. Input sinusoids, output will be sinusoids.
- Convolution, image domain filtering, is LTI.

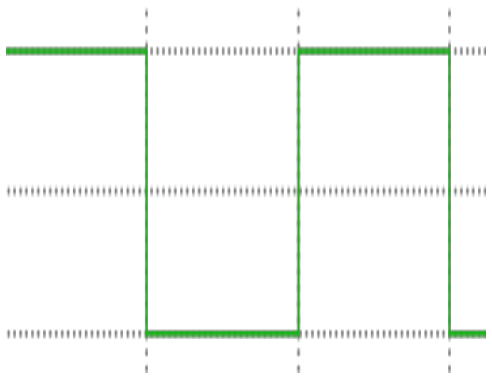
Fourier transform

Frequency Spectra

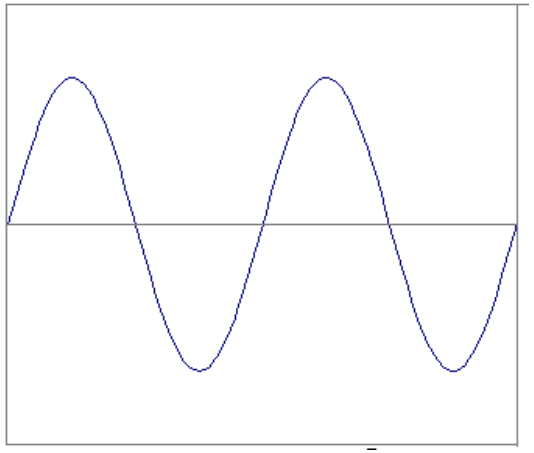
- Consider a square wave f(x)



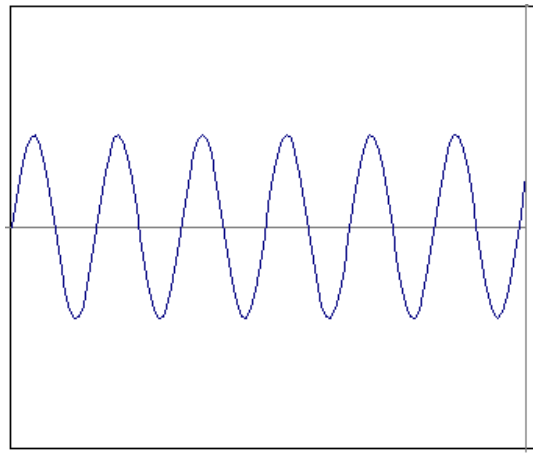
Frequency Spectra



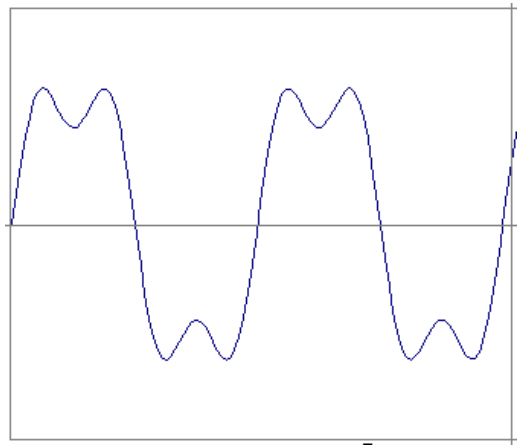
=



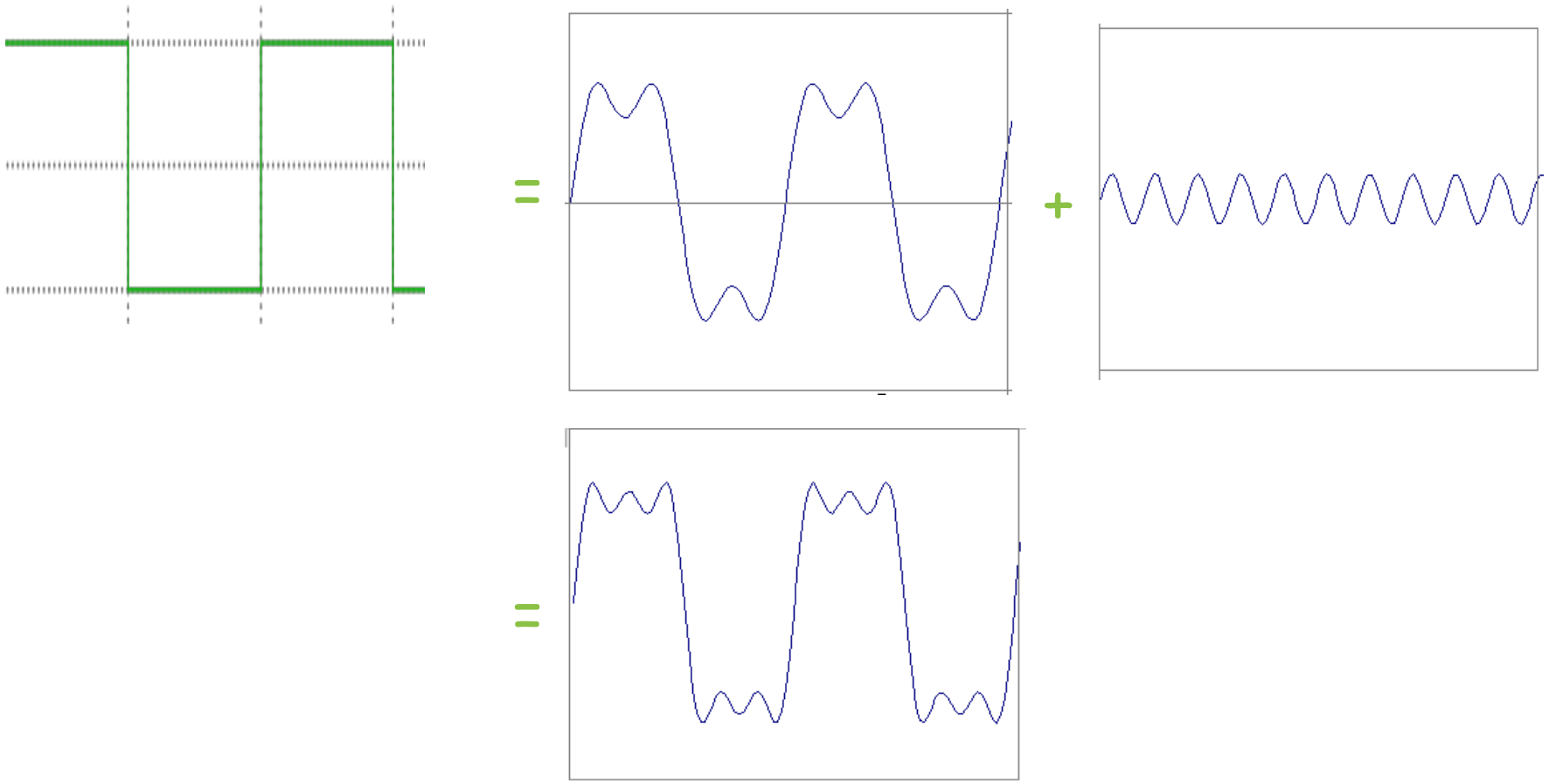
+



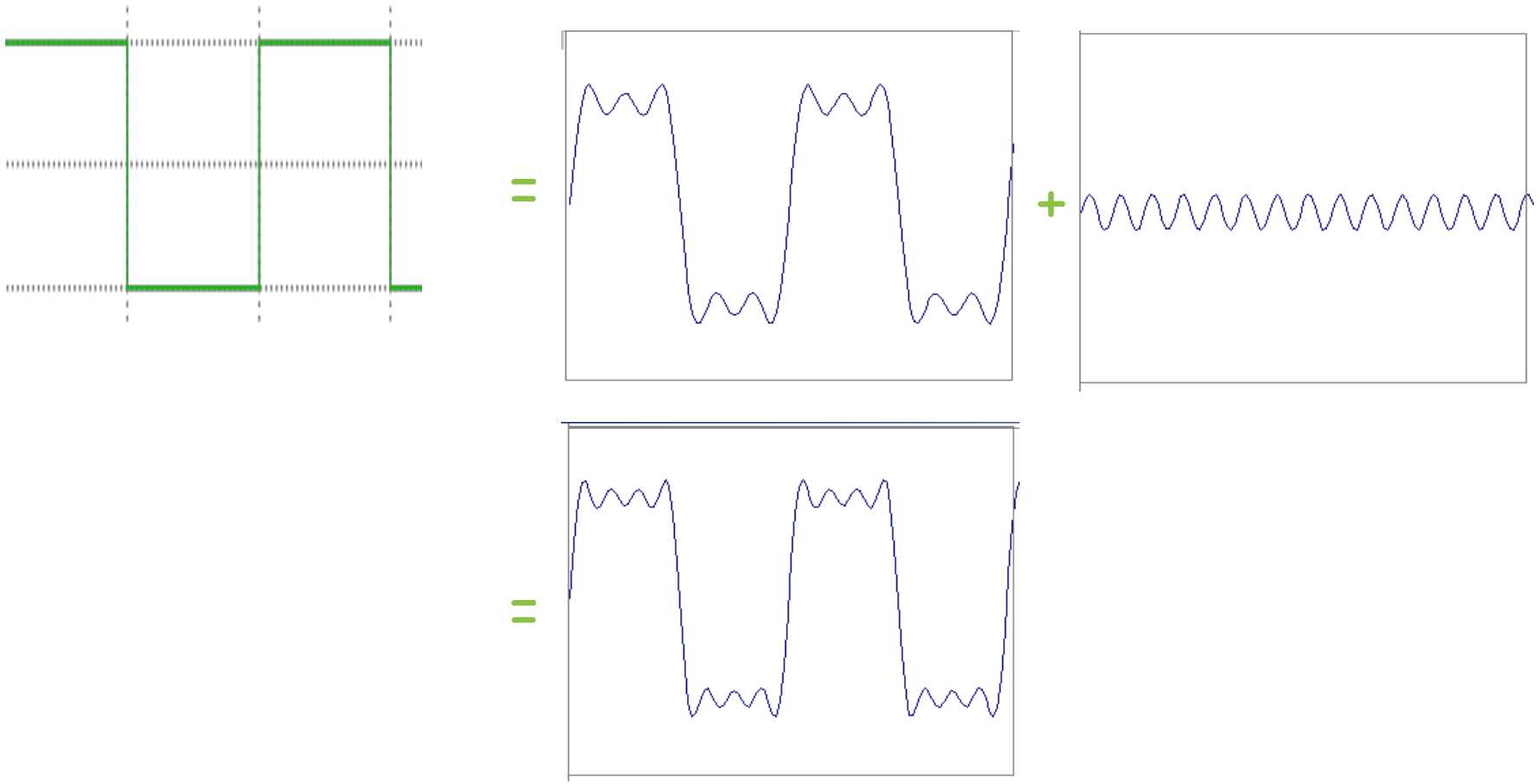
=



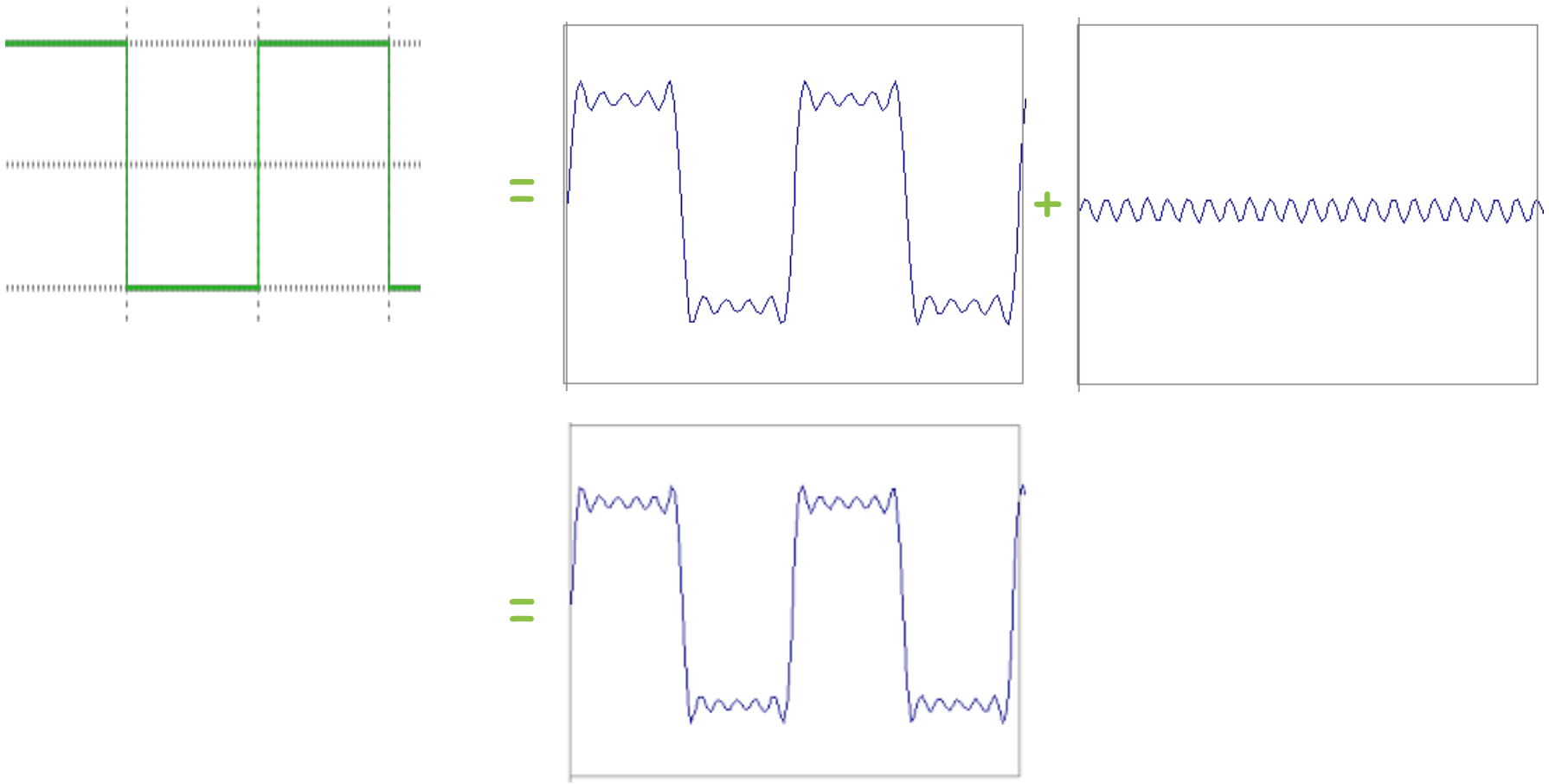
Frequency Spectra



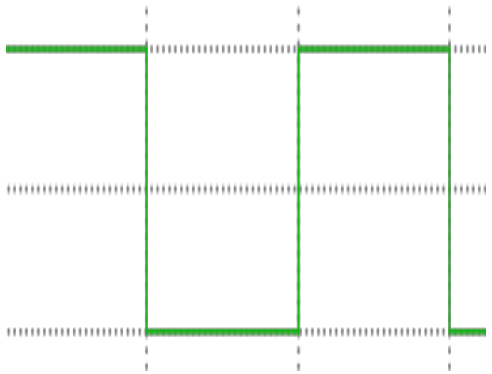
Frequency Spectra

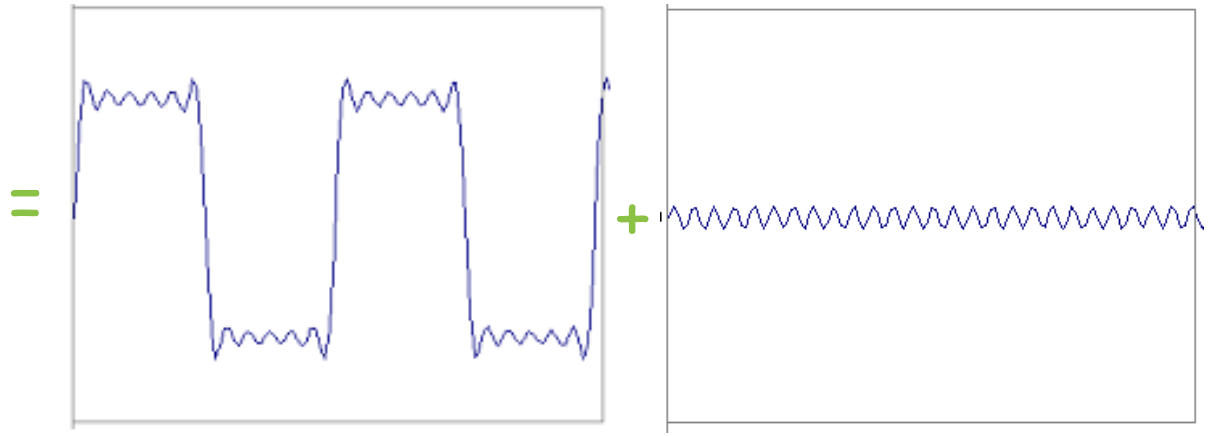


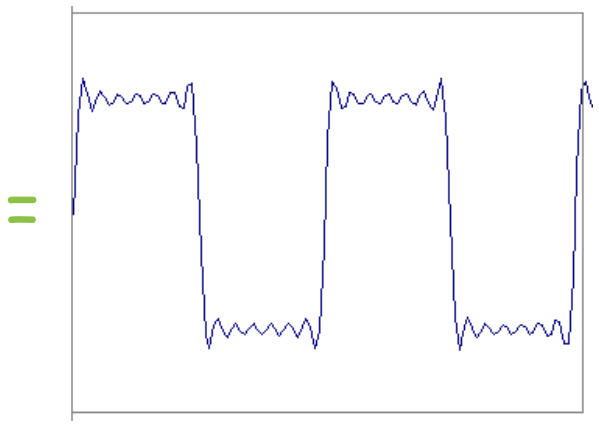
Frequency Spectra



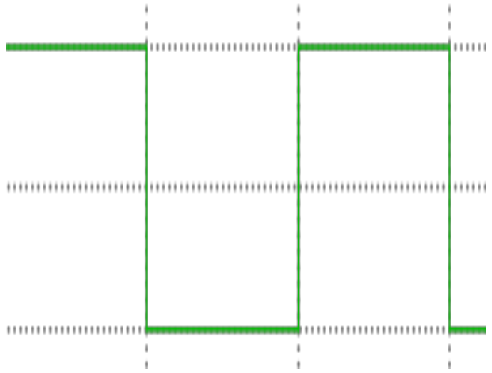
Frequency Spectra





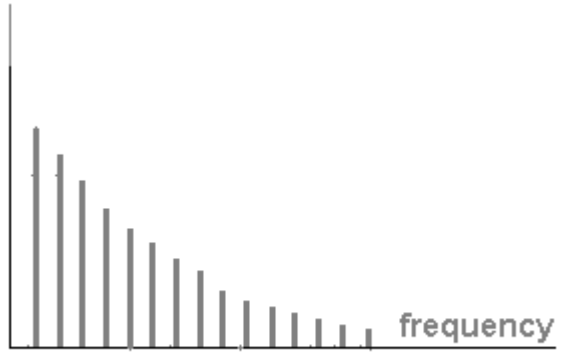


Frequency Spectra



=

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



Fourier Transform

- Fourier Transform

Inverse Fourier transform

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \qquad f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega$$

Time f(t)  Frequency f(w)

What is transformation? It is a mapping between two domains

The Discrete Fourier transform

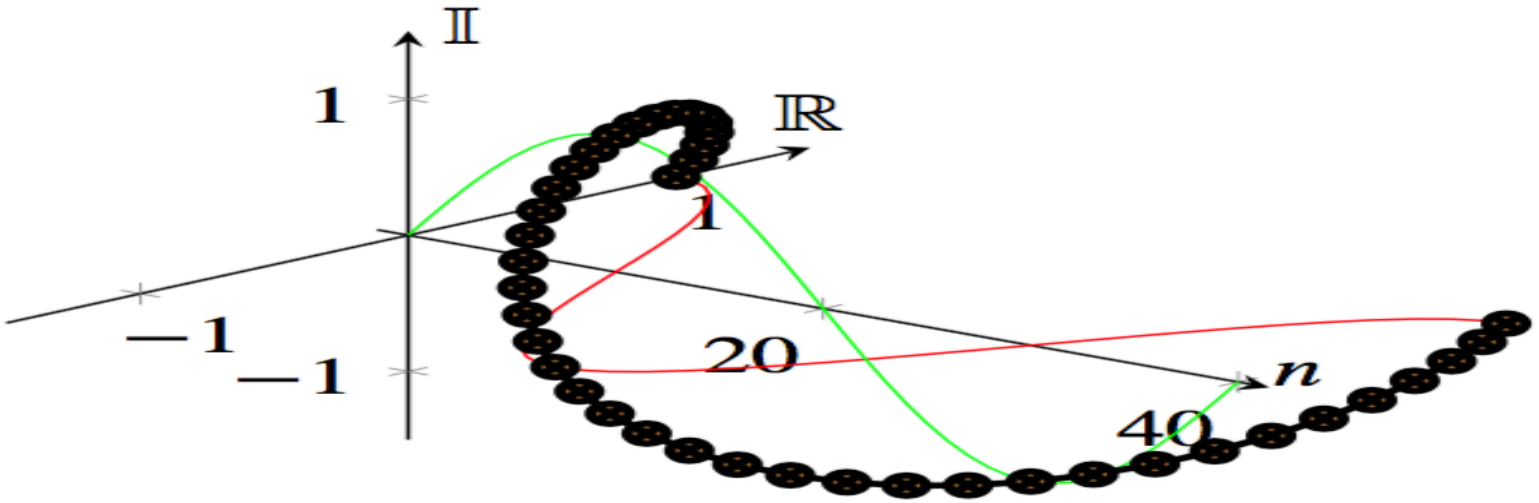
Discrete Fourier Transform (DFT) transforms an image
$f[m, m]$ into the complex image Fourier transform $F[u, v]$ as:

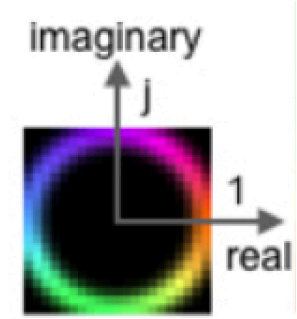
$$F[u,v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

The inverse Fourier transform is:

$$f[n,m] = \frac{1}{NM} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F[u,v] \exp\left(+2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

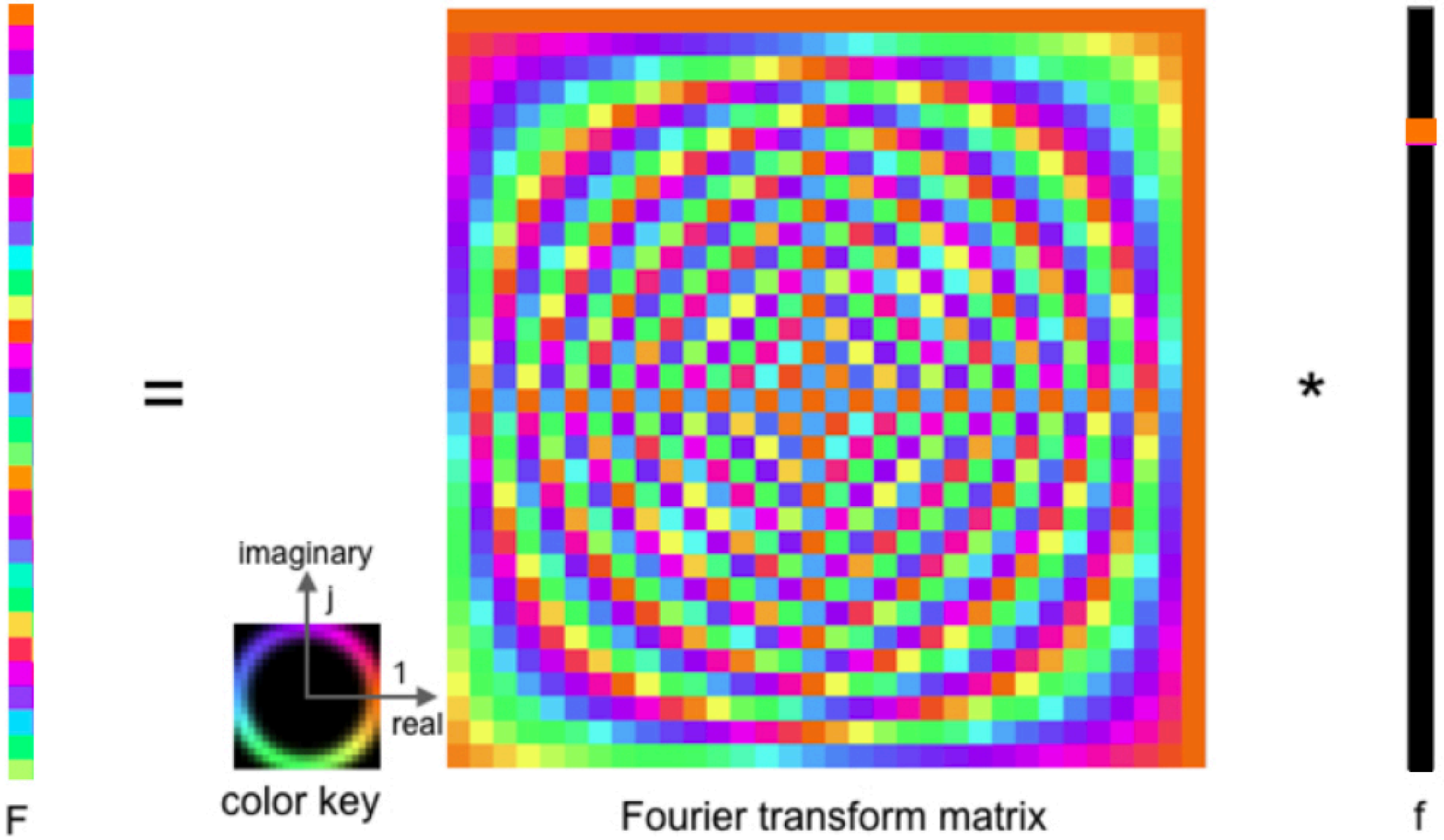
Discrete Fourier transform visualization





color key

Fourier transform visualization



Some useful transforms

Fourier transform of the Delta function $\delta[n, m]$:

$$F[u,v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \delta[n,m] \exp\left(-2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right) = 1$$

Observation: if we think in terms of the inverse DFT, this means that:

$$\delta[n,m] = \frac{1}{NM} \sum_{u=-N/2}^{N/2} \sum_{v=-M/2}^{M/2} \exp\left(2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

Some useful transforms

The Fourier transform of the cosine wave

$$\cos\left(2\pi\left(\frac{u_0\, n}{N}+\frac{v_0\, m}{M}\right)\right)$$

is:

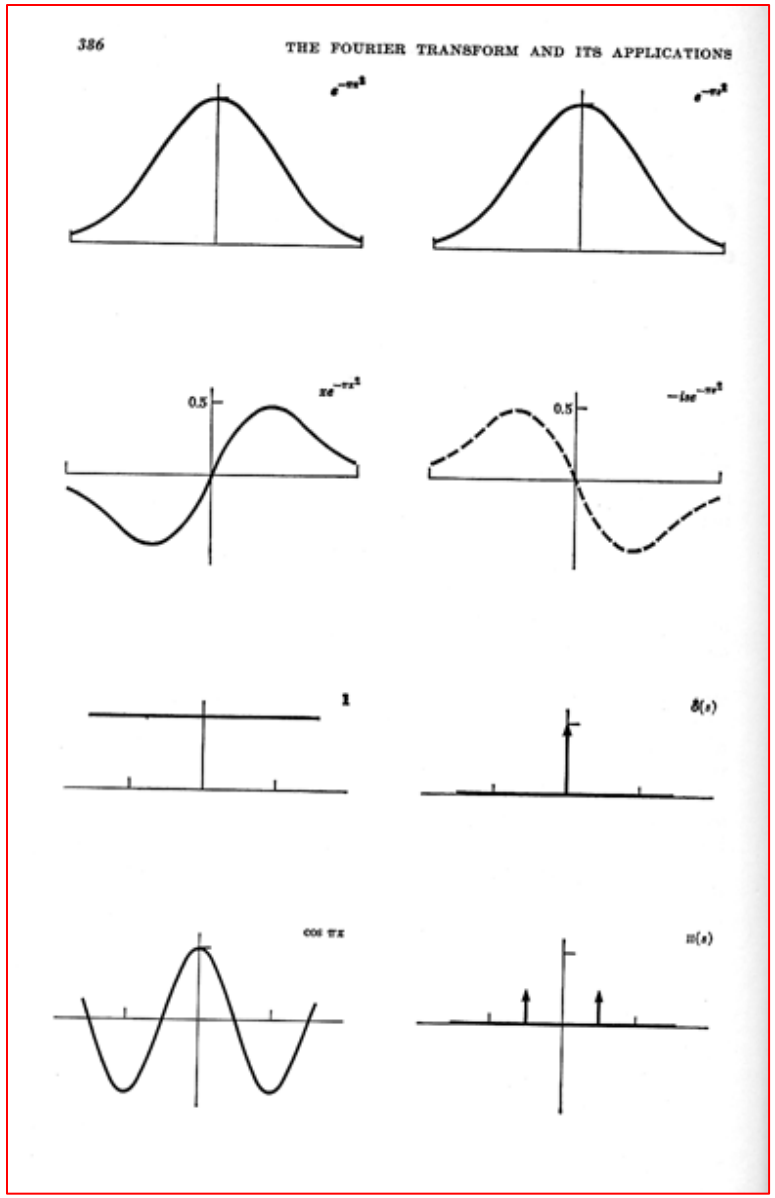
$$F[u,v]=\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\cos\left(2\pi\left(\frac{u_0\, n}{N}+\frac{v_0\, m}{M}\right)\right)\exp\left(-2\pi j\left(\frac{u\, n}{N}+\frac{v\, m}{M}\right)\right)=$$

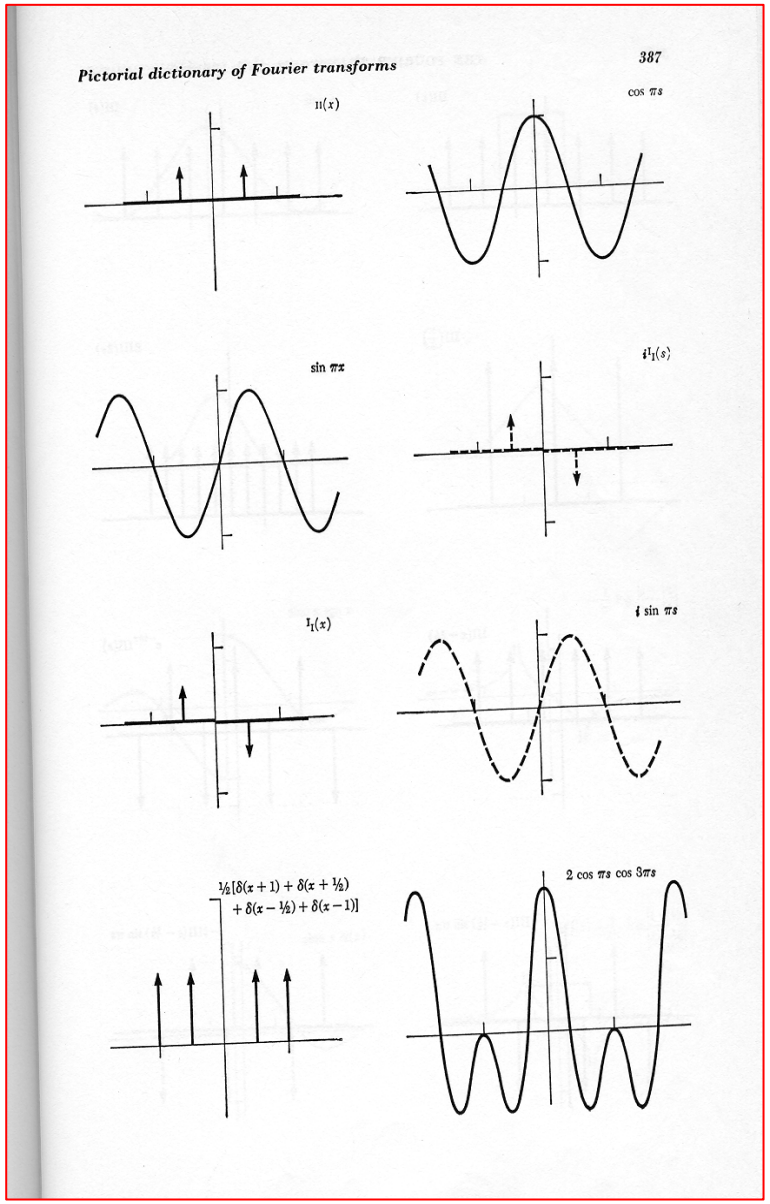
$$=\frac{1}{2}\left(\delta[u-u_0,v-v_0]+\delta[u+u_0,v+v_0]\right)$$

Same for the sine wave:

$$\sin\left(2\pi\left(\frac{u_0\, n}{N}+\frac{v_0 m}{M}\right)\right)\leftrightarrow F[u,v]=\frac{1}{2j}\left(\delta[u-u_0,v-v_0]-\delta[u+u_0,v+v_0]\right)$$

Bracewell's pictorial dictionary of Fourier transform pairs





Bracewell, The Fourier Transform and its Applications, McGraw Hill 1978

Szeliski, Computer Vision, 2010

Name	Signal		Transform
impulse	$\delta(x)$	$\Leftrightarrow$	1
shifted impulse	$\delta(x-u)$	$\Leftrightarrow$	$e^{-j\omega u}$
box filter	$\text{box}(x/a)$	$\Leftrightarrow$	$a\text{sinc}(a\omega)$
tent	$\text{tent}(x/a)$	$\Leftrightarrow$	$a\text{sinc}^2(a\omega)$
Gaussian	$G(x;\sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega;\sigma^{-1})$
Laplacian of Gaussian	$(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2})G(x;\sigma)$	$\Leftrightarrow$	$-\frac{\sqrt{2\pi}}{\sigma}\omega^2 G(\omega;\sigma^{-1})$
Gabor	$\cos(\omega_0 x)G(x;\sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega\pm\omega_0;\sigma^{-1})$
unsharp mask	$(1+\gamma)\delta(x)$ $-\gamma G(x;\sigma)$	$\Leftrightarrow$	$(1+\gamma)-$ $\frac{\sqrt{2\pi}\gamma}{\sigma}G(\omega;\sigma^{-1})$
windowed sinc	$\text{rcos}(x/(aW))$ $\text{sinc}(x/a)$	$\Leftrightarrow$	(see Figure 3.29)

Table 3.2 Some useful (continuous) Fourier transform pairs: The dashed line in the Fourier transform of the shifted impulse indicates its (linear) phase. All other transforms have zero phase (they are real-valued). Note that the figures are not necessarily drawn to scale but are drawn to illustrate the general shape and characteristics of the filter or its response. In particular, the Laplacian of Gaussian is drawn inverted because it resembles more a "Mexican hat", as it is sometimes called.

2D Discrete Fourier Transform

$$F[u,v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

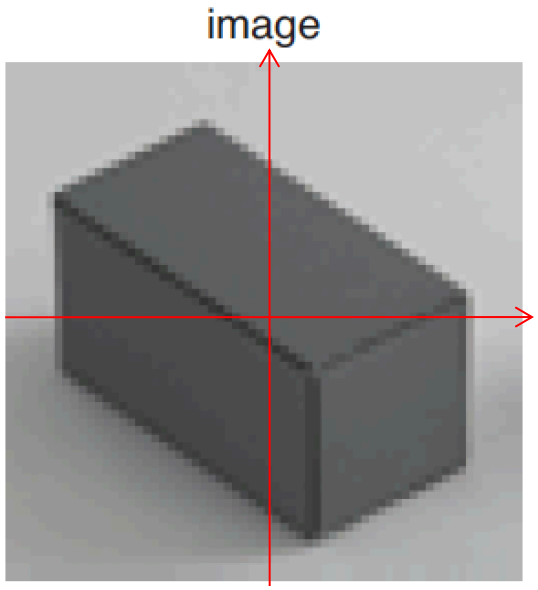
Using the real and imaginary components:

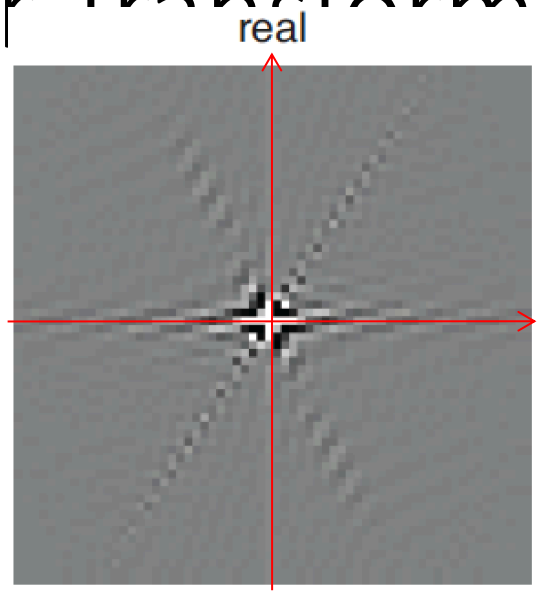
$$F[u,v] = Re\{F[u,v]\} + j\,Imag\{F[u,v]\}$$

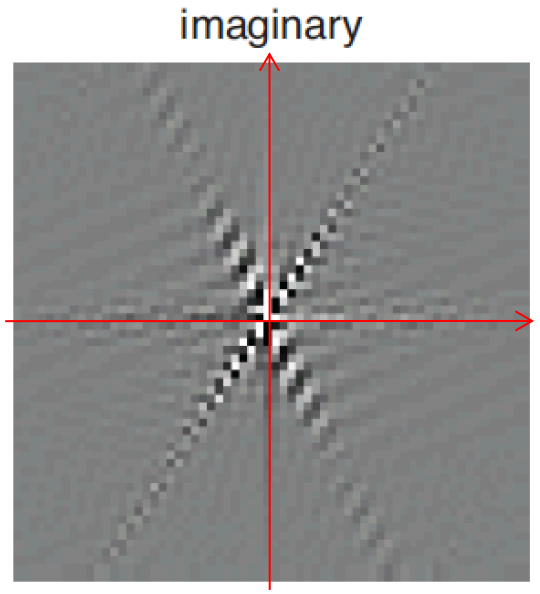
Or using a polar decomposition:

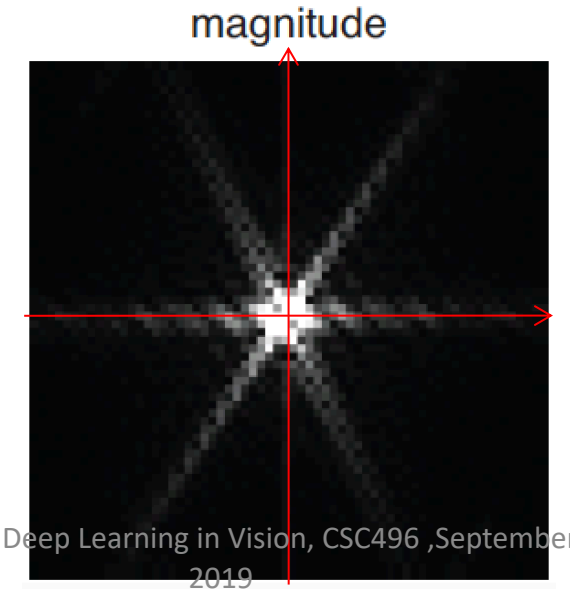
$$F[u,v] = A[u,v] \exp(j\theta[u,v])$$

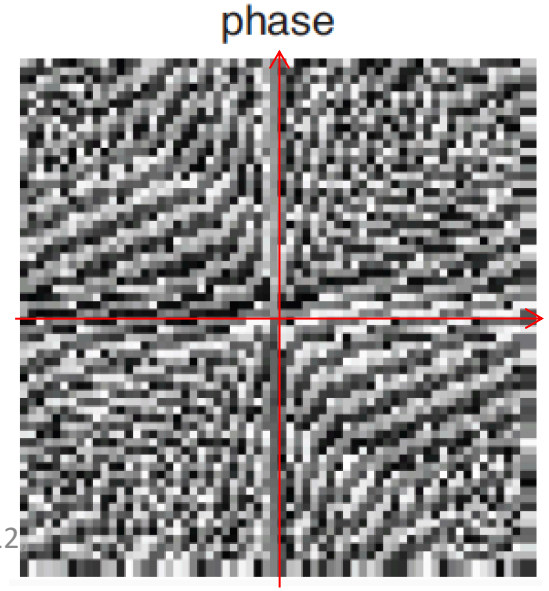
2D Discrete Fourier Transform











Properties for the DFT

$$F[u,v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

- Linearity
- Symmetry: Fourier transform of a real signal has coefficients that come in pairs, with $F[u, v]$ being the complex conjugate of $F[-u, -v]$.

Properties for the DFT

- Shift in space

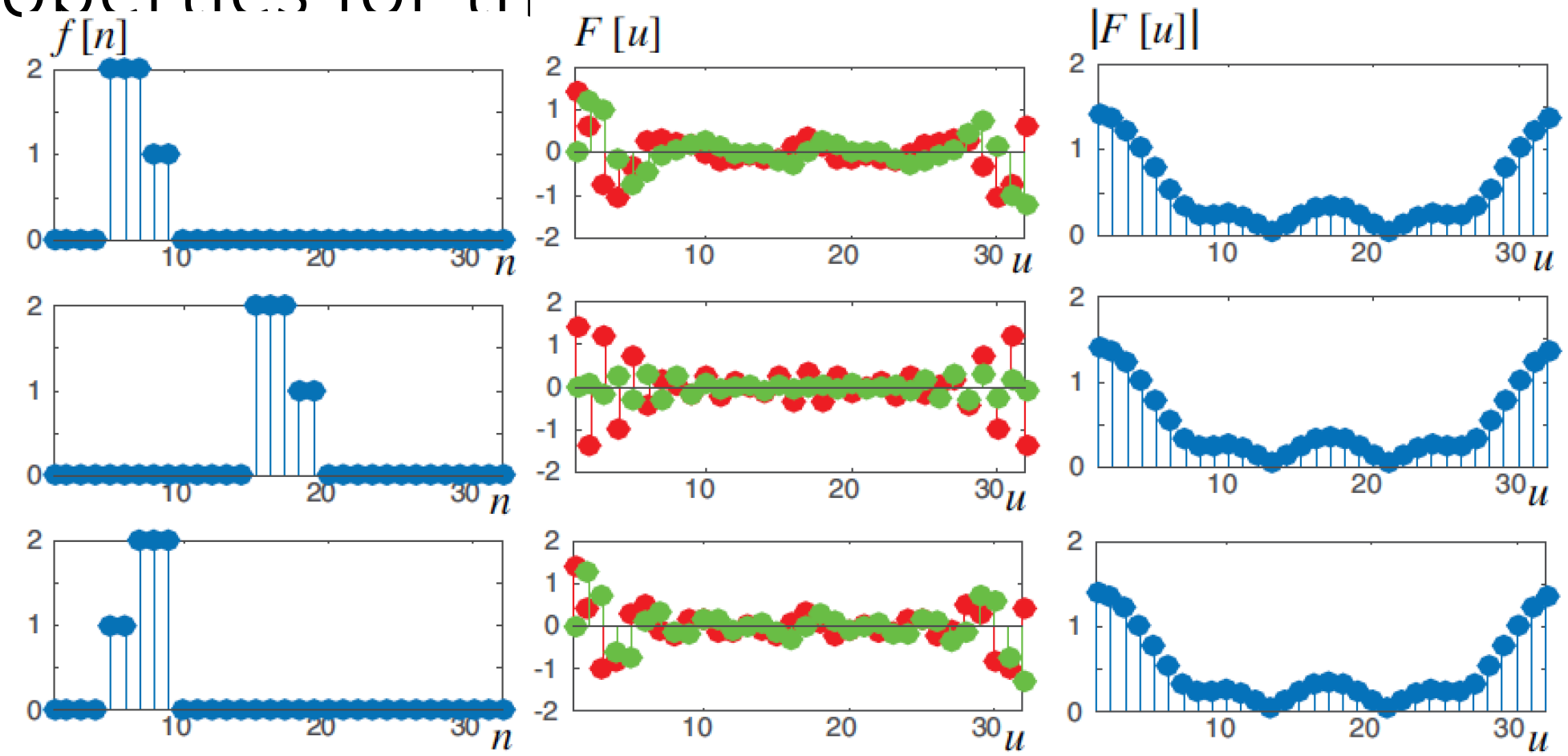
$$DFT\{f[n-n_0, m-m_0]\} =$$

$$= \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n-n_0, m-m_0] \exp\left(-2\pi j \left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right) =$$

$$= \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j \left(\frac{u\,(n+n_0)}{N} + \frac{v\,(m+m_0)}{M}\right)\right) =$$

$$= F[u,v] \exp\left(-2\pi j \left(\frac{u\,n_0}{N} + \frac{v\,m_0}{M}\right)\right)$$

Properties for the DFT



Only the phase changes! The magnitude is translation invariant.

Properties for the DFT

- Modulation

$$f[n,m]\cos\left(2\pi j\left(\frac{u_0 n}{N}+\frac{v_0 m}{M}\right)\right)$$

$$f[n,m]\exp\left(-2\pi j\left(\frac{u_0 n}{N}+\frac{v_0 m}{M}\right)\right)$$

Multiplying by a complex exponential results in a translation of the DFT

$$DFT\left\{f[n,m]\exp\left(-2\pi j\left(\frac{u_0 n}{N}+\frac{v_0 m}{M}\right)\right)\right\}=F[u-u_0,v-v_0]$$

Frequencies

DFT amplitude

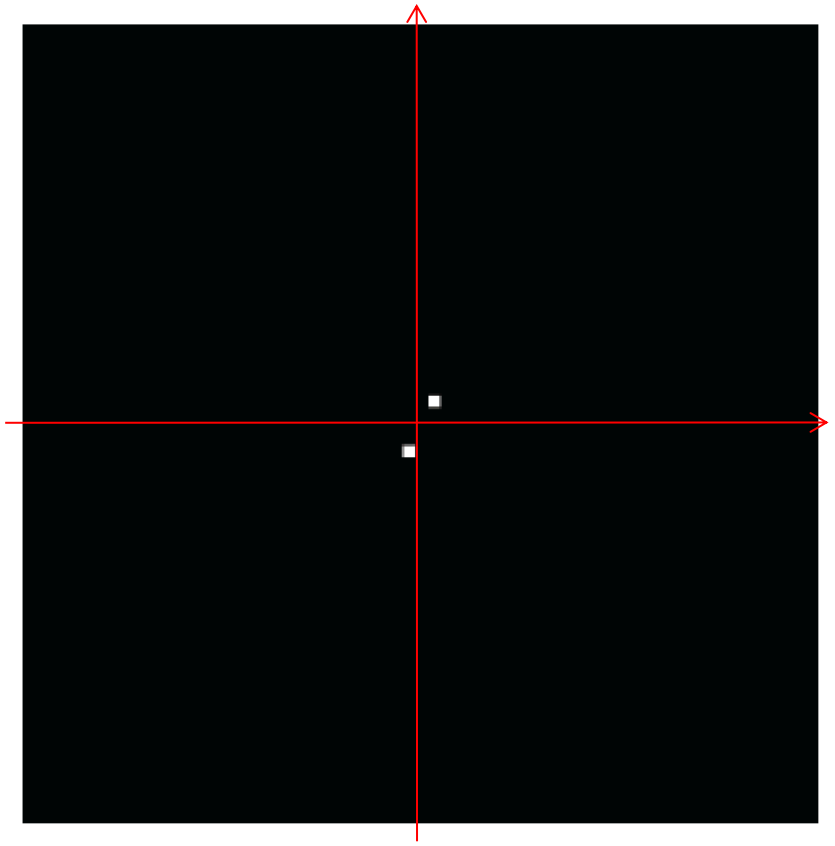
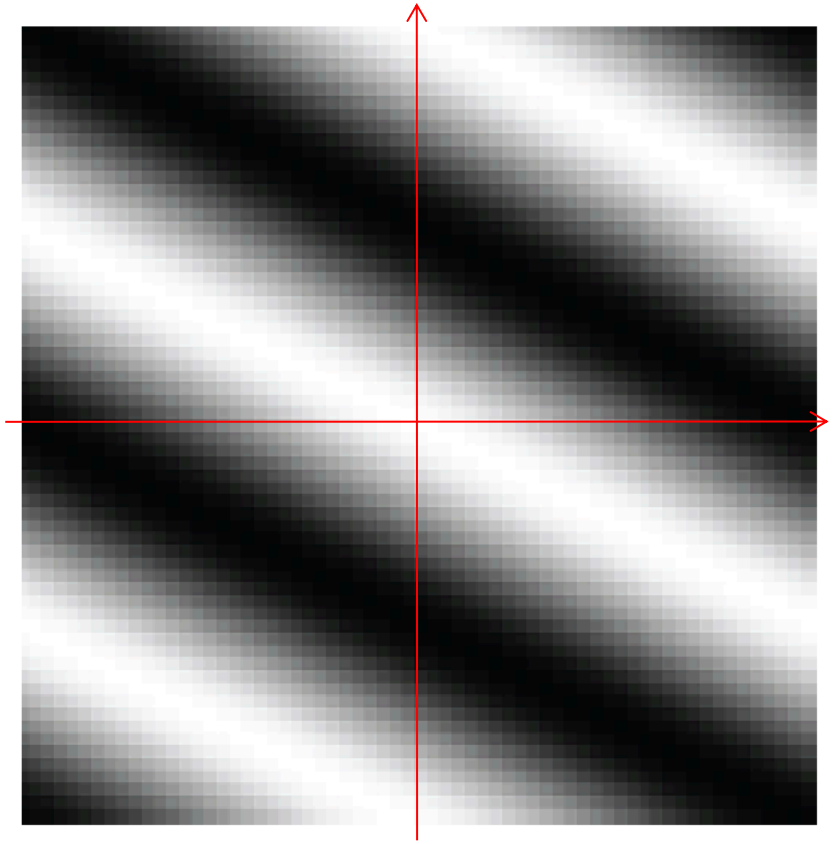
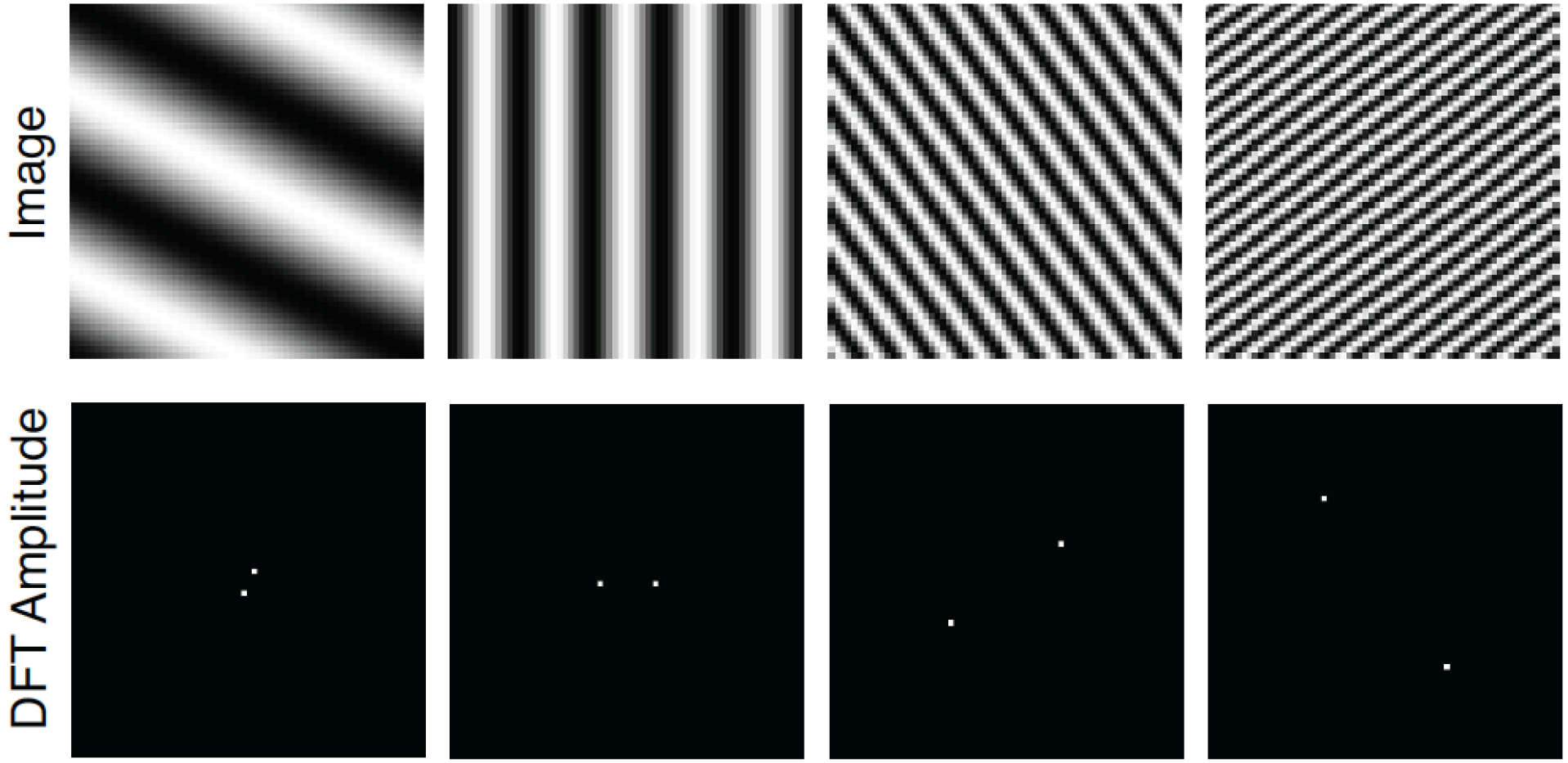


Image
(assuming zero phase)

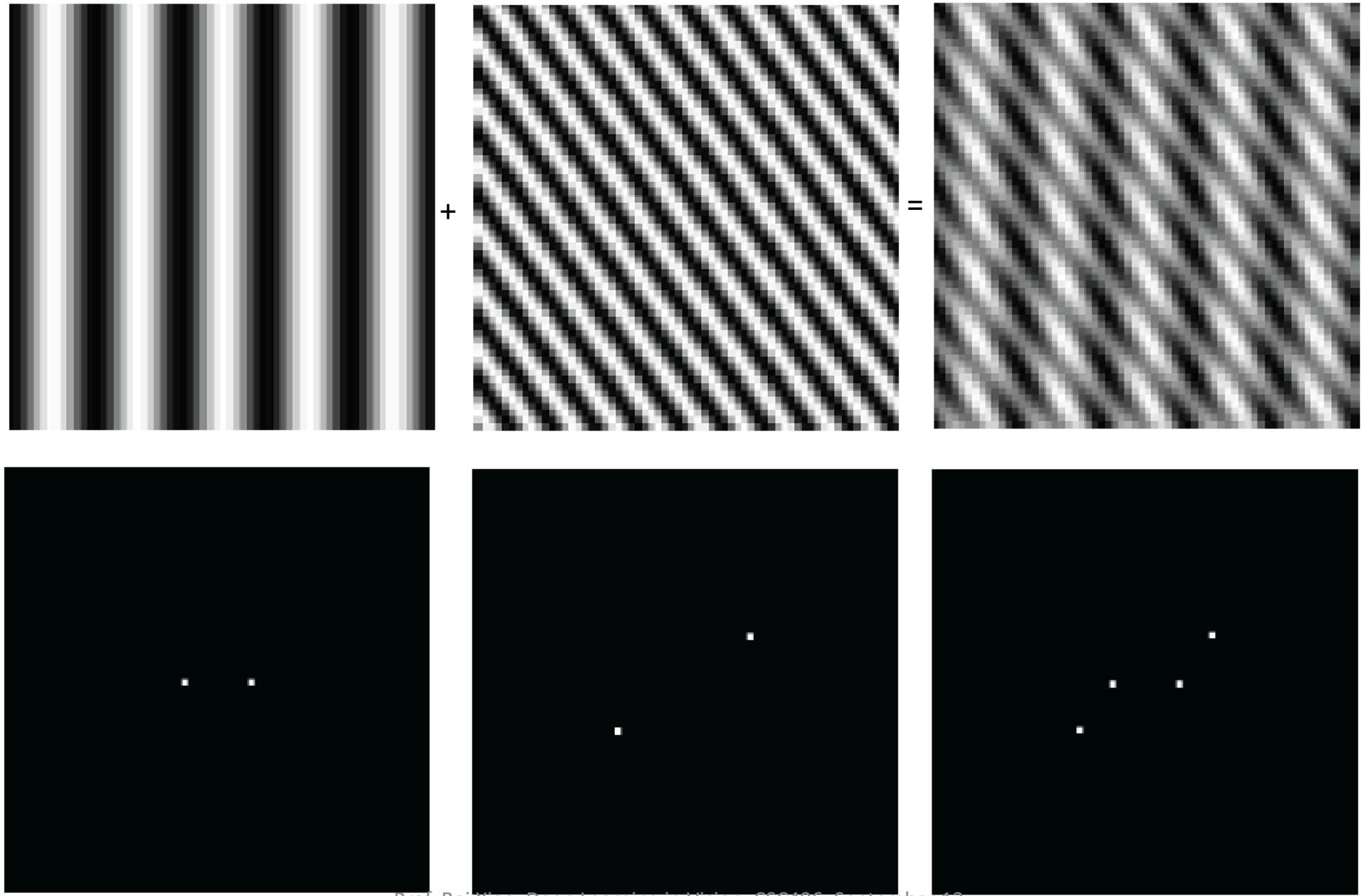


Images are 64x64 pixels. The wave is a cosine (if phase is zero).

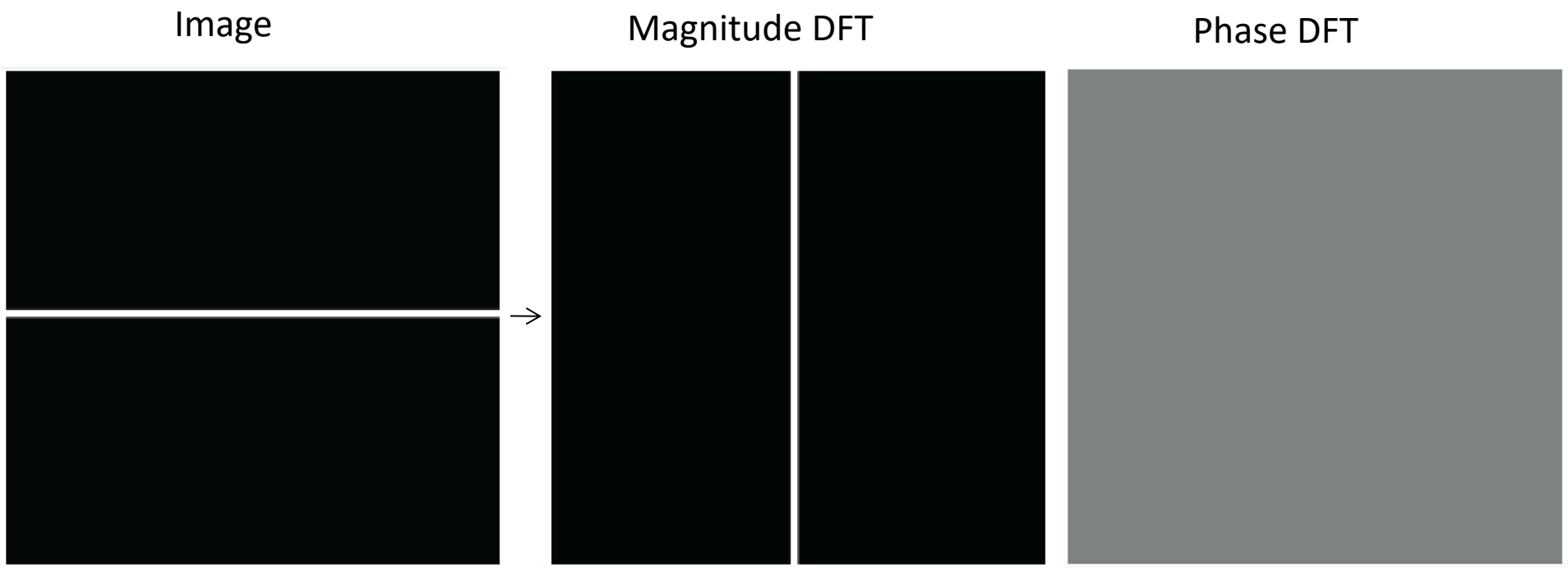
Frequencies



Images are 64x64 pixels. The wave is a cosine (if phase is zero).

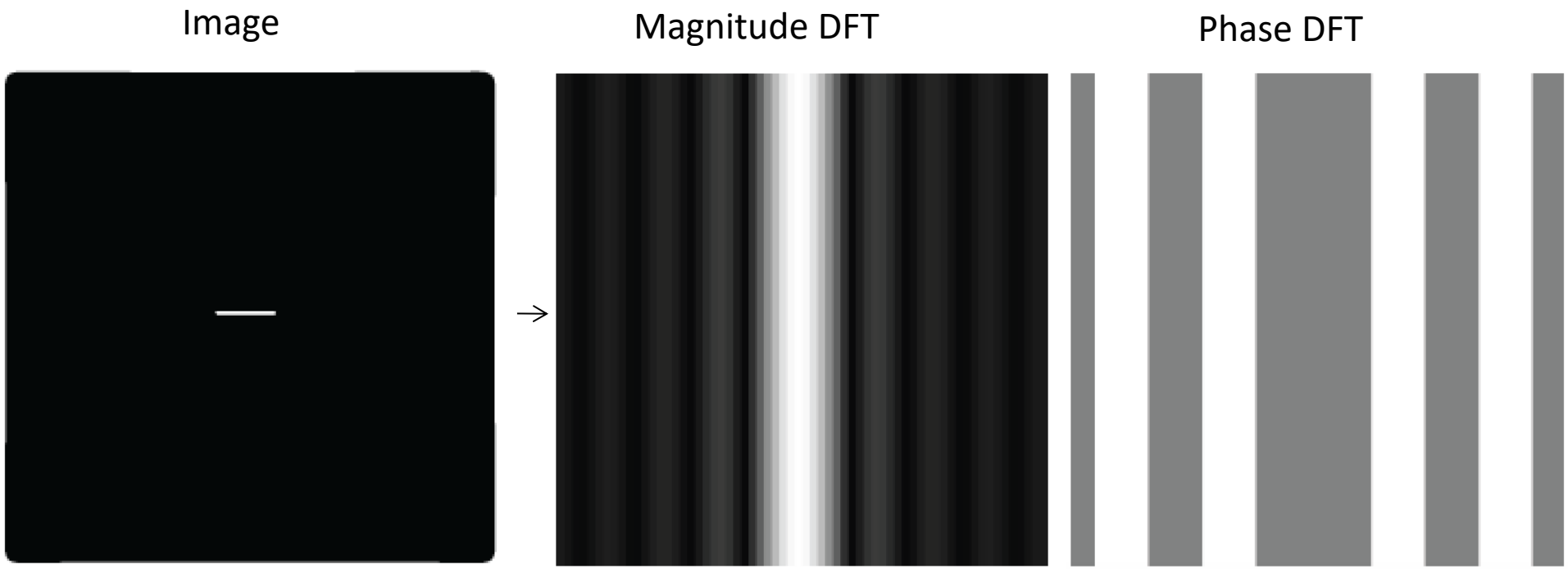


Some important Fourier transforms

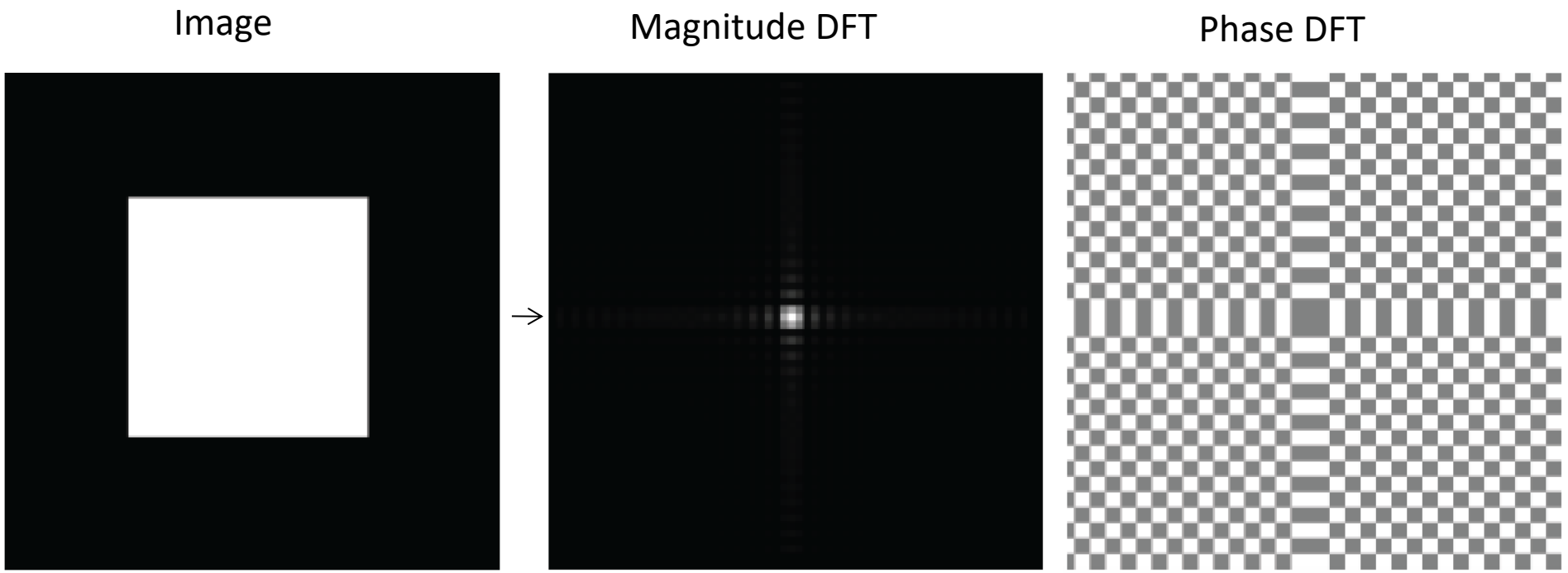


Images are 64x64 pixels.

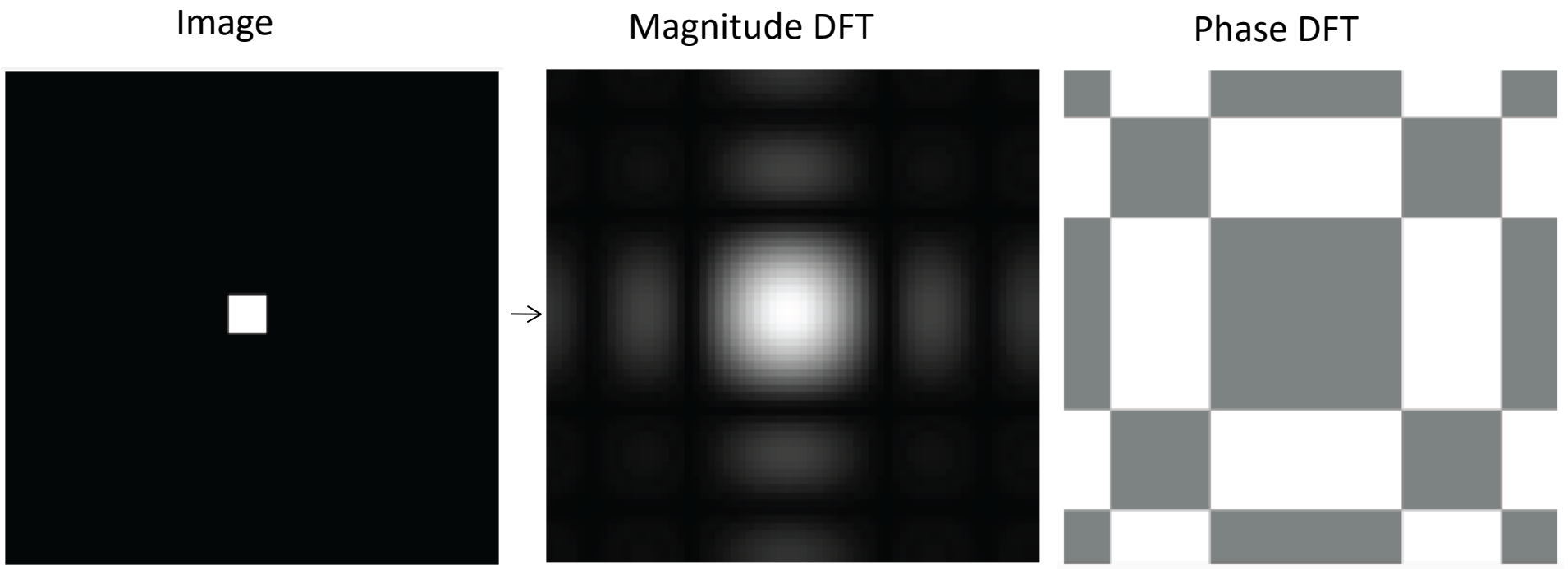
Some important Fourier transforms



Some important Fourier transforms

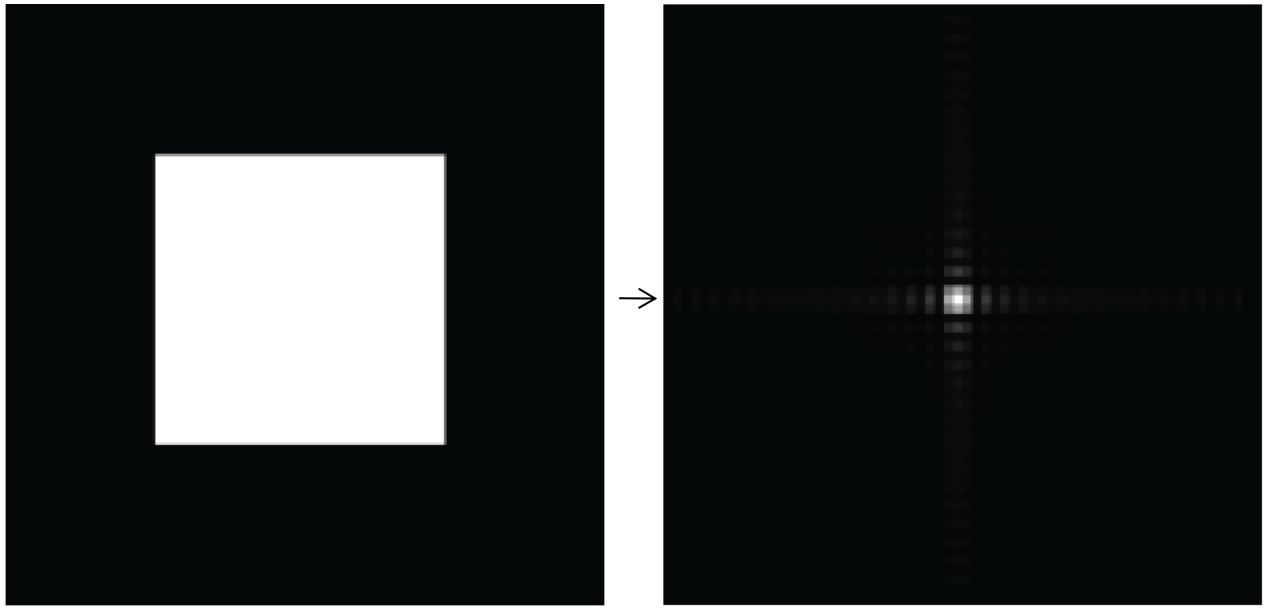


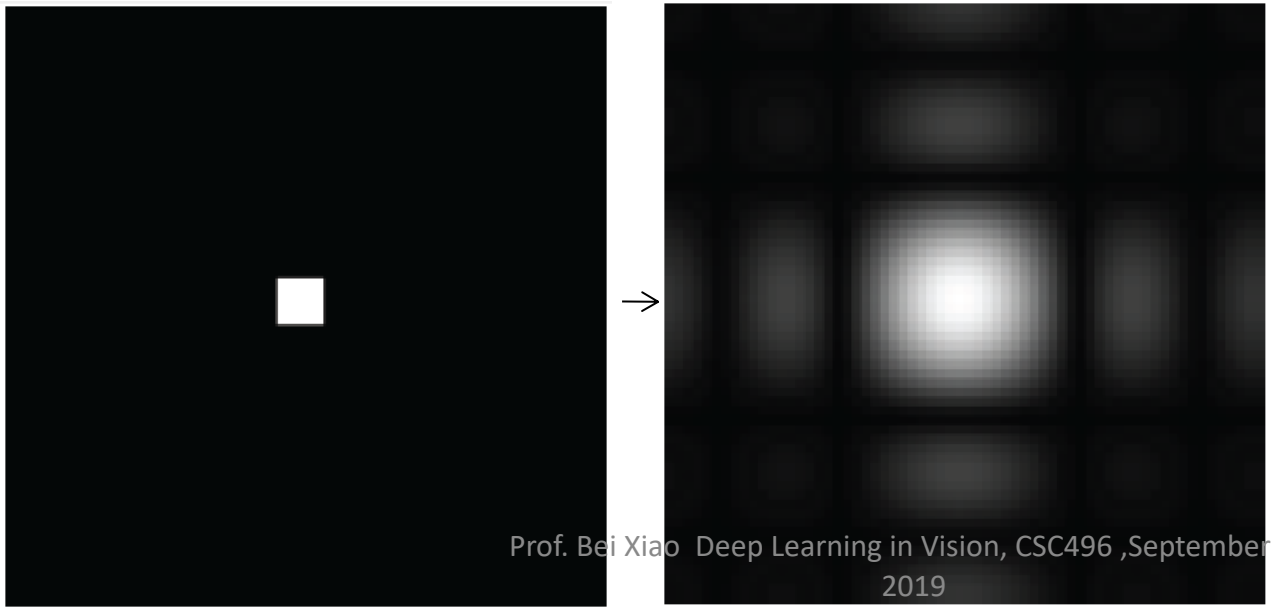
Some important Fourier transforms



Image

Magnitude DFT

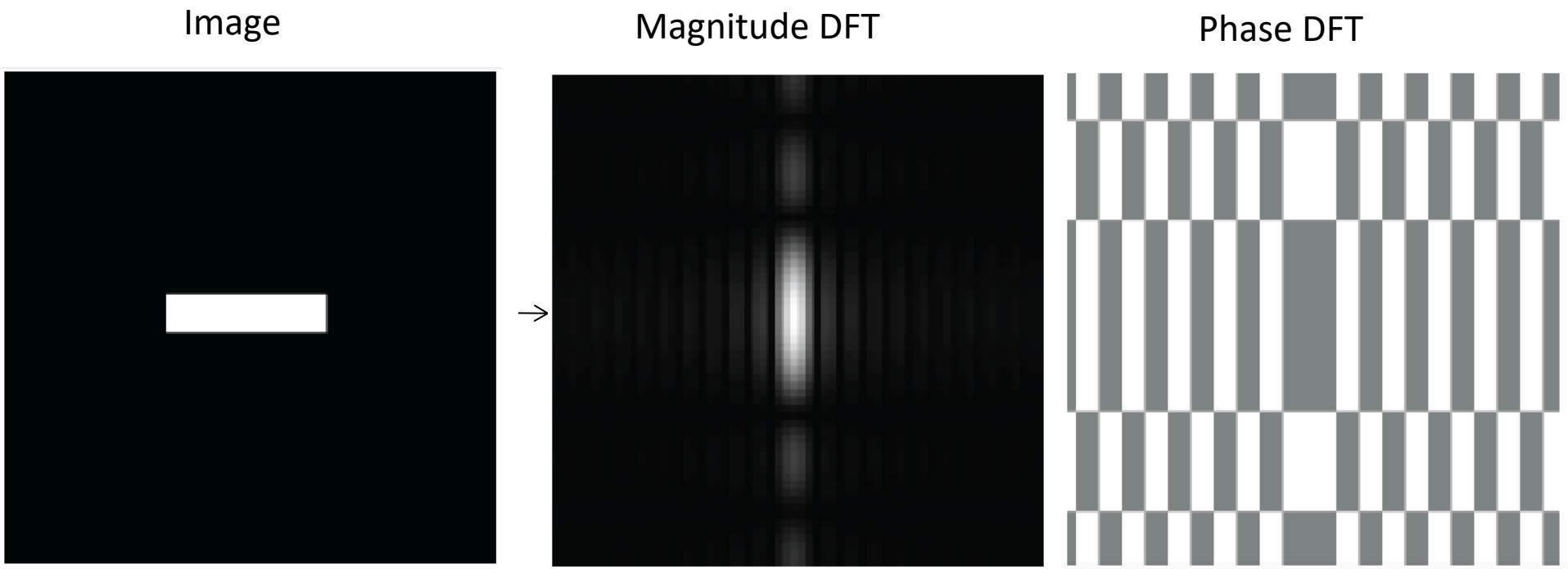




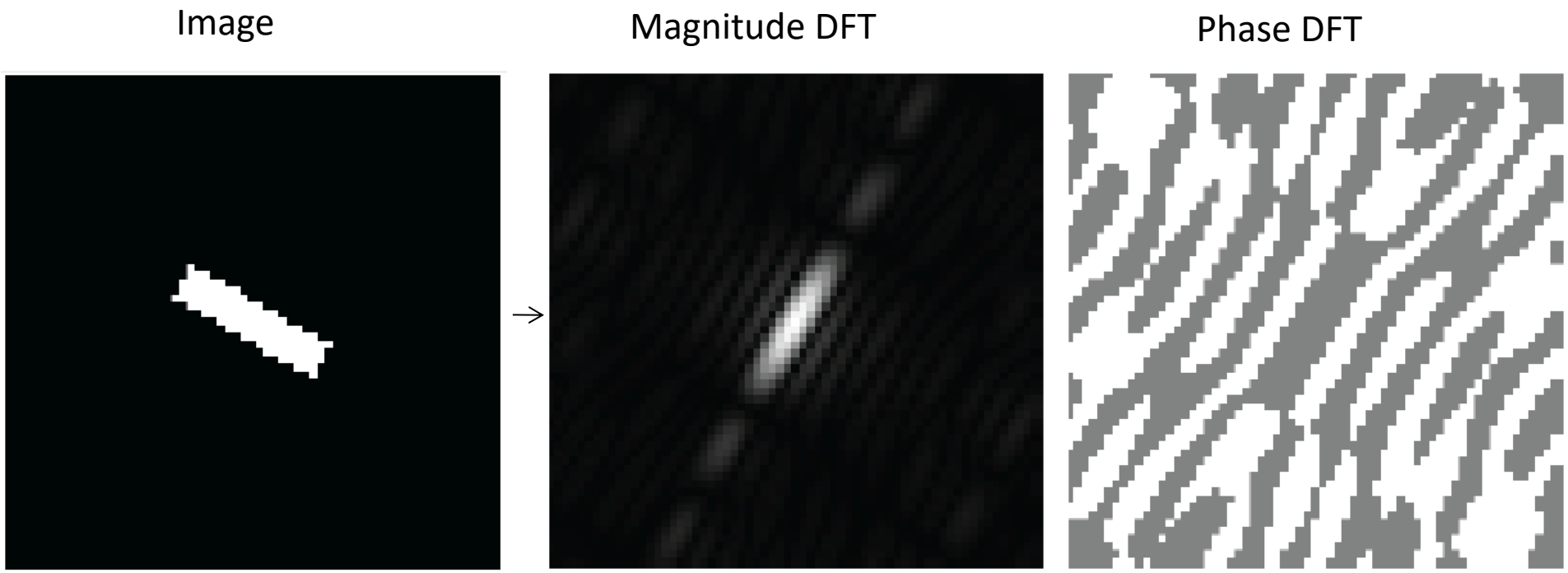
Scale

Small image
details produce content in
high spatial frequencies

Some important Fourier transforms

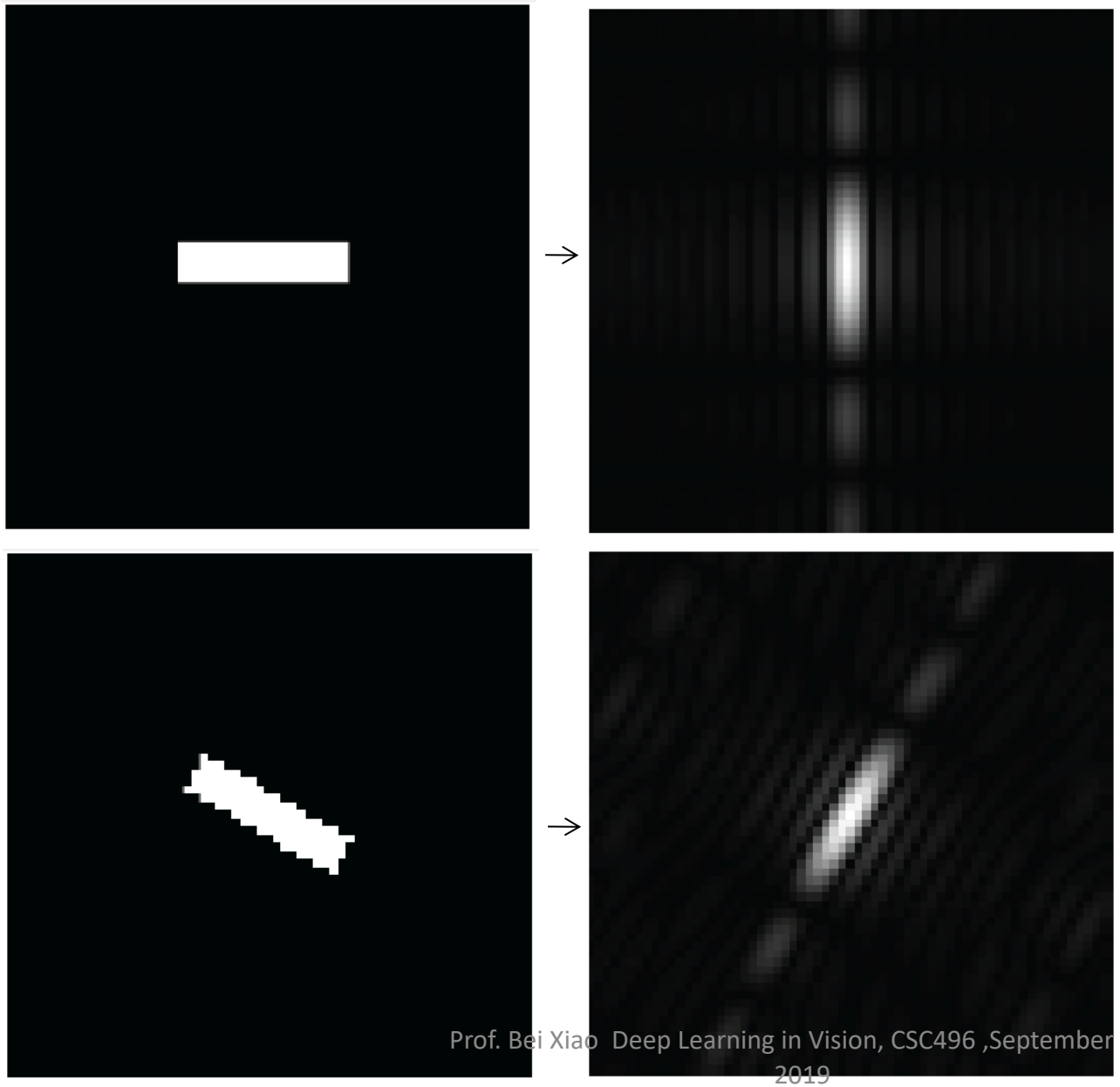


Some important Fourier transforms



Image

Magnitude DFT

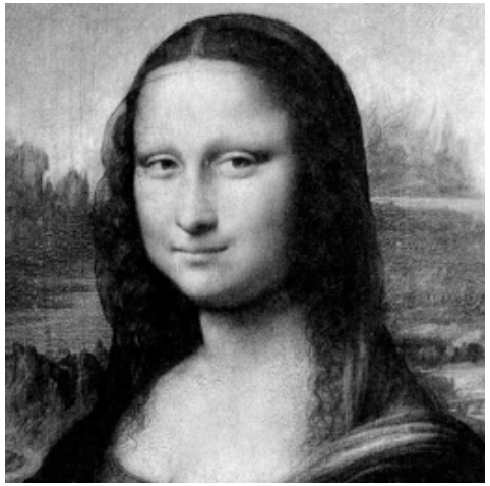


Orientation

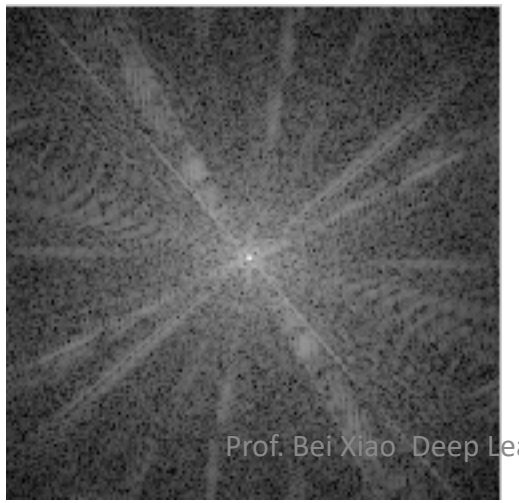
A line transforms to a line oriented perpendicularly to the first.

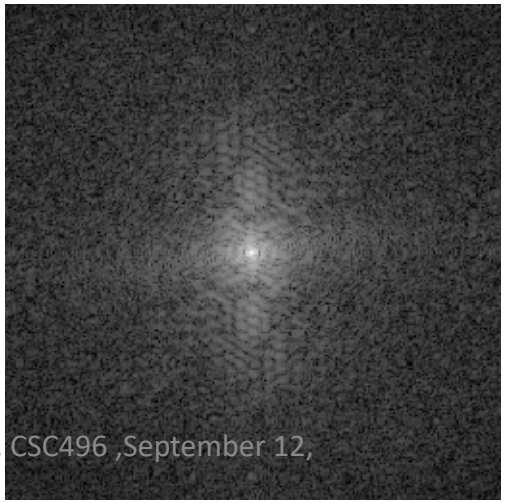
The Fourier Transform of some important images

Image



Log(1+Magnitude FT)





More properties for the DFT

$$F[u,v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

DFT of the convolution

$$f = g \circ h \longleftrightarrow F[u,v] = G[u,v]H[u,v]$$

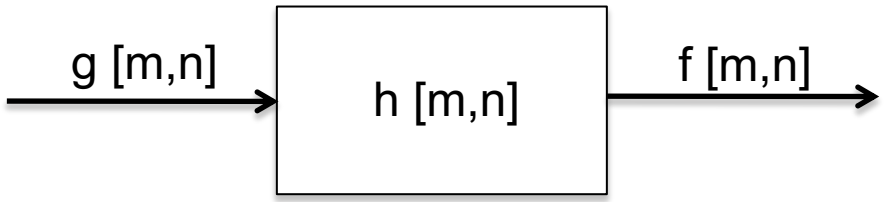
$$F[u,v] = DFT\{g \circ h\}$$

$$= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} g[m-k,n-l]\,h[k,l]\exp\left(-2\pi j\left(\frac{mu}{M}+\frac{nv}{N}\right)\right)$$

$$F[u,v] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} h[k,l]\sum_{m'=-k}^{M-k-1}\sum_{n'=-l}^{N-l-1} g[m',n']\exp\left(-2\pi j\left(\frac{(m'+k)u}{M}+\frac{(n'+l)v}{N}\right)\right)$$

$$F[u,v] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} G[u,v]\exp\left(-2\pi j\left(\frac{ku}{M}+\frac{lv}{N}\right)\right)h[k,l]$$

Linear filtering



In the spatial domain:

$$f[m,n] = h \circ g = \sum_{k,l} h[m-k, n-l]\, g[k,l]$$

In the frequency domain:

$$F[u,v] = G[u,v]\, H[u,v]$$

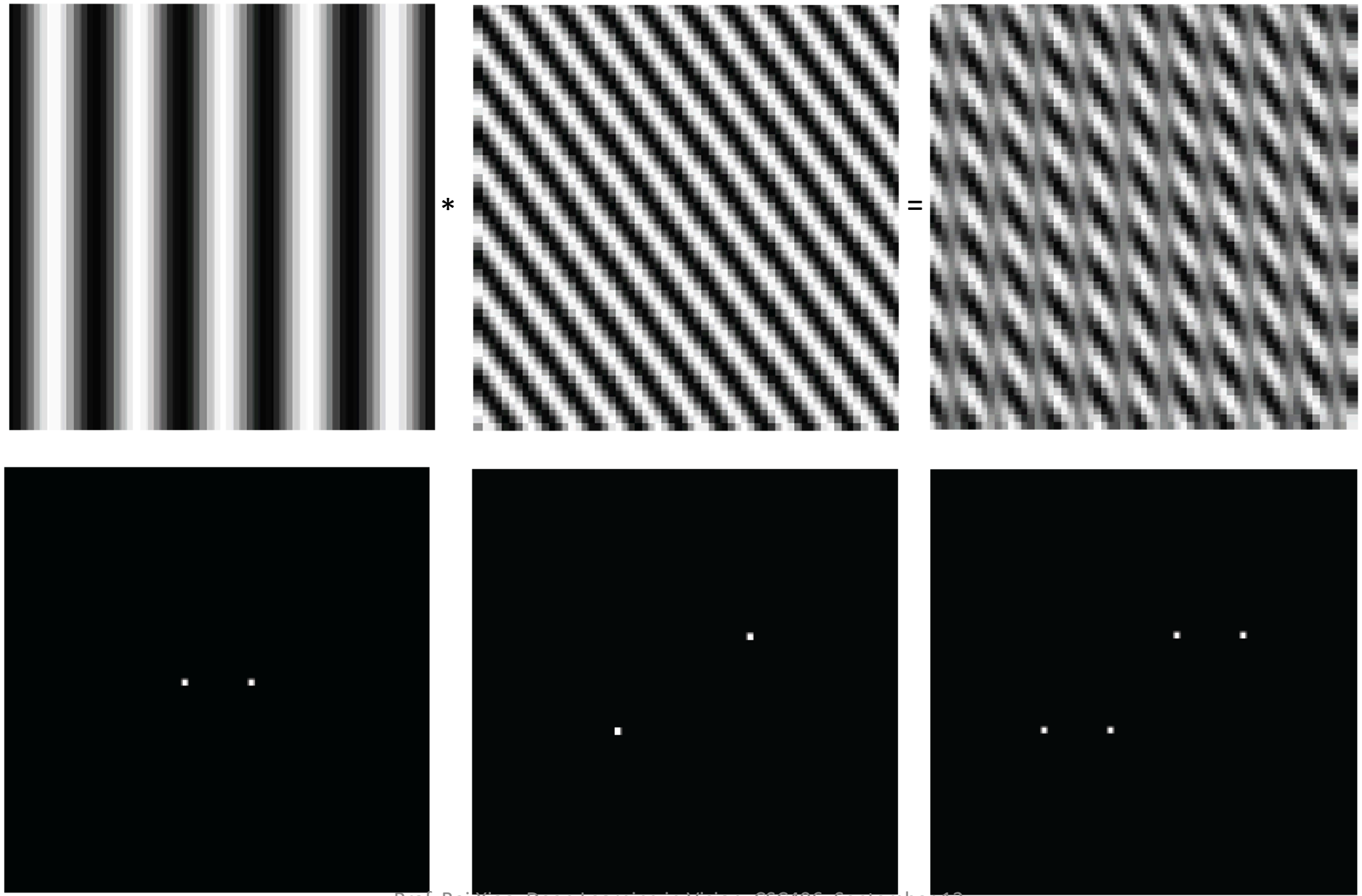
Product of images

The Fourier transform of the product of two images

$$f[n,m] = g[n,m]\,h[n,m]$$

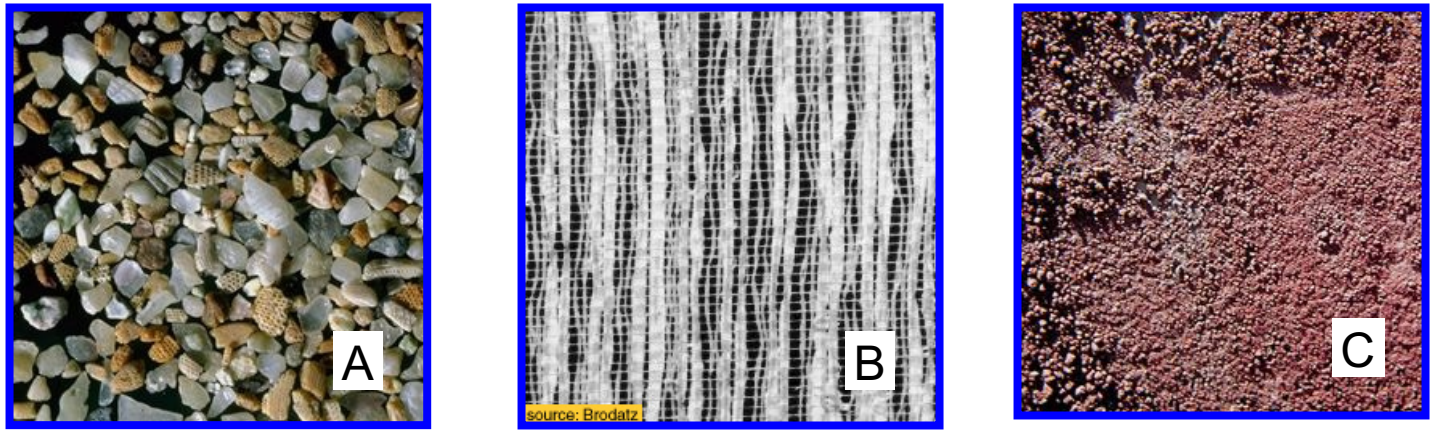
is the convolution of their DFTs:

$$F[u,v] = \frac{1}{NM} G[u,v] \circ H[u,v]$$

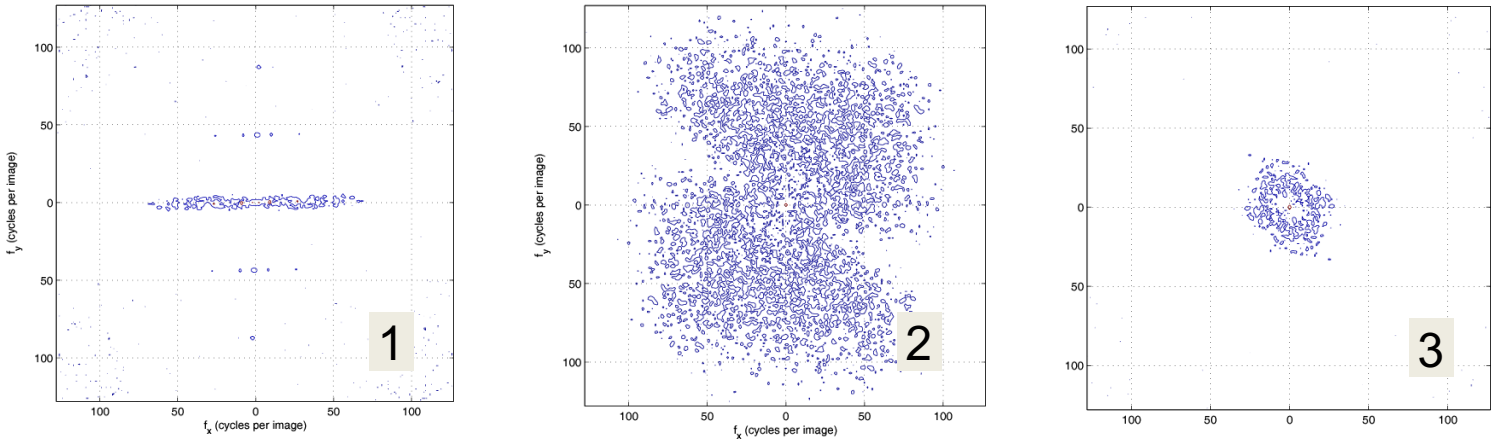


Game: find the right pairs

Images



DFT magnitude



fx(cycles/image pixel size) fx(cycles/image pixel size) fx(cycles/image pixel size)

The inverse Discrete Fourier transform

$$f[n,m] = \frac{1}{NM} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F[u,v] \exp\left(+2\pi j \left(\frac{u\,n}{N} + \frac{v\,m}{M}\right)\right)$$

How does summing waves ends up giving back a picture?

Take-home reading and next class

- First quiz
- More on flourier transform.
- Multiscale pyramids

- Lecture 4&5 notes: SignalProcessing.pdf

- Linear filtering: page 111-120
- http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf