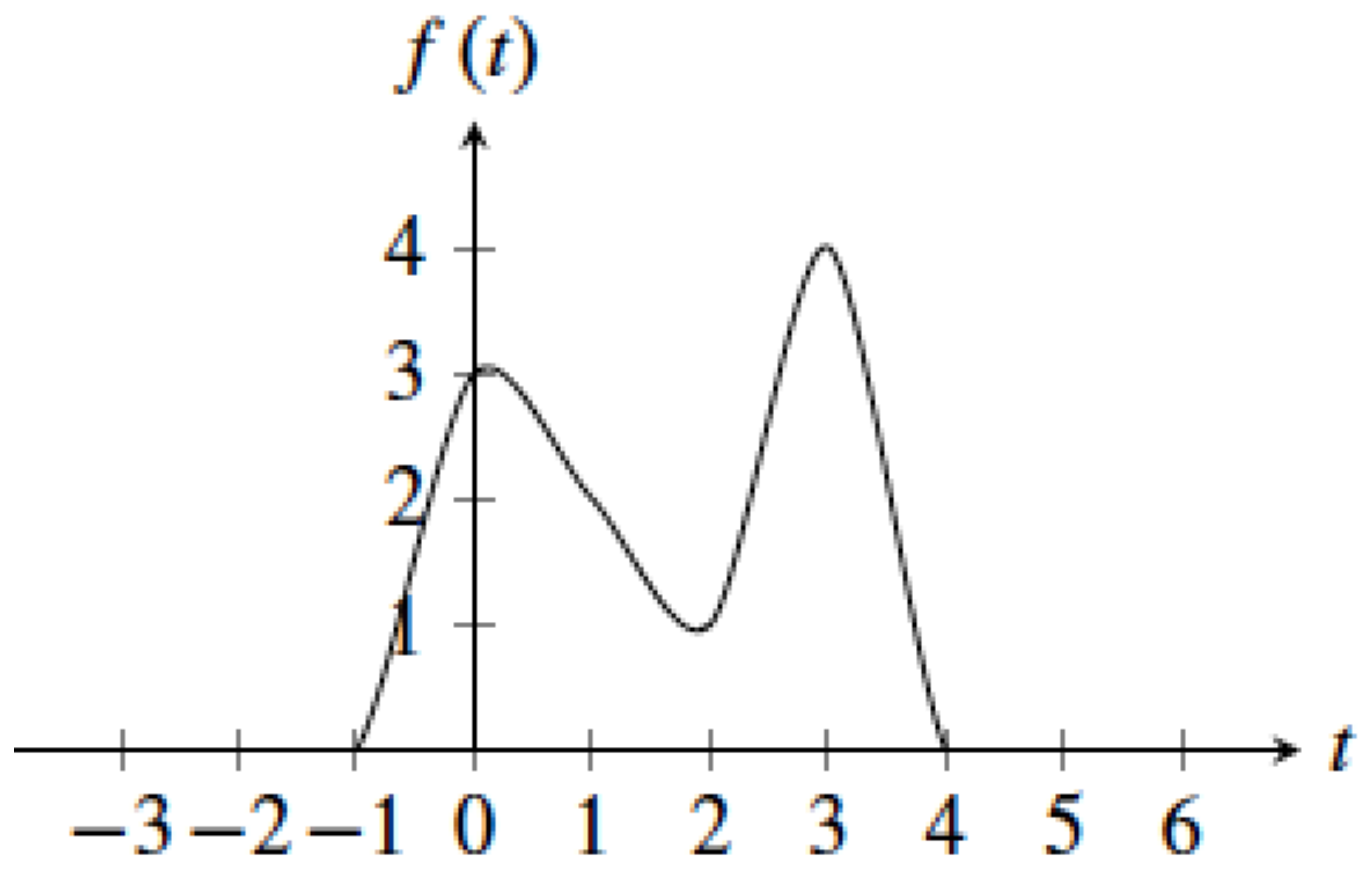
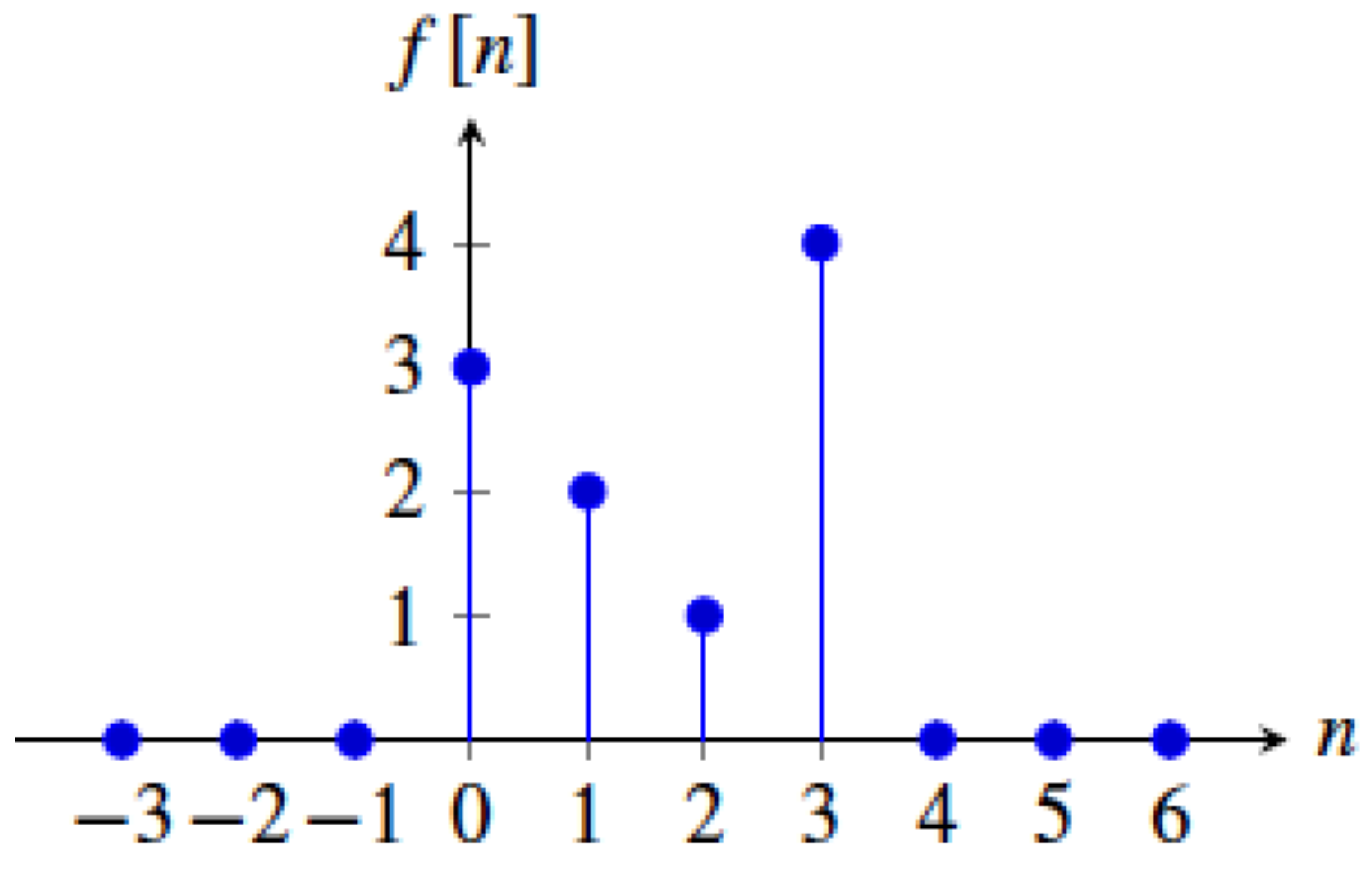
Lecture 4

Signal Processing

6.003 Signals and systems



Time continuous signal

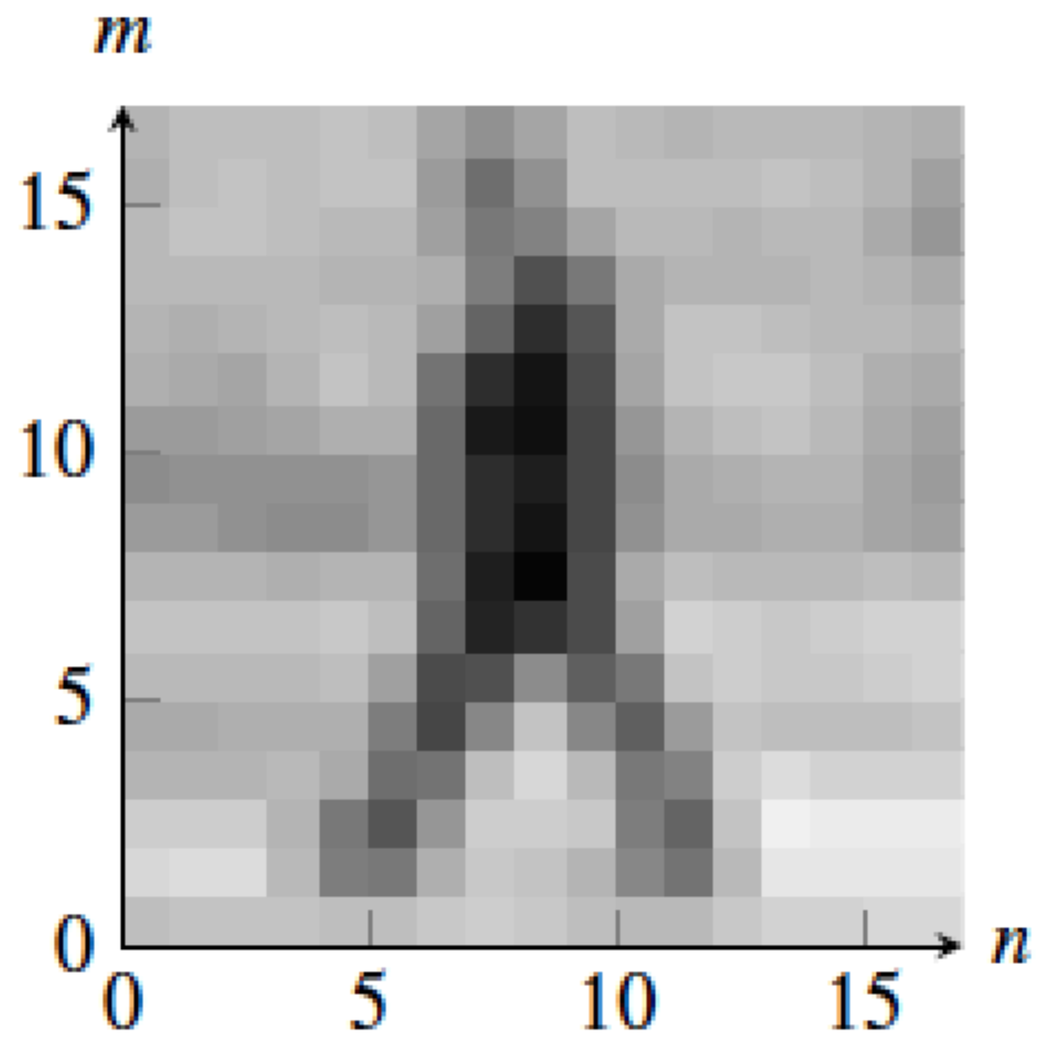


Time discrete signal

Review class notes!

Remember, an image is just an array of numbers

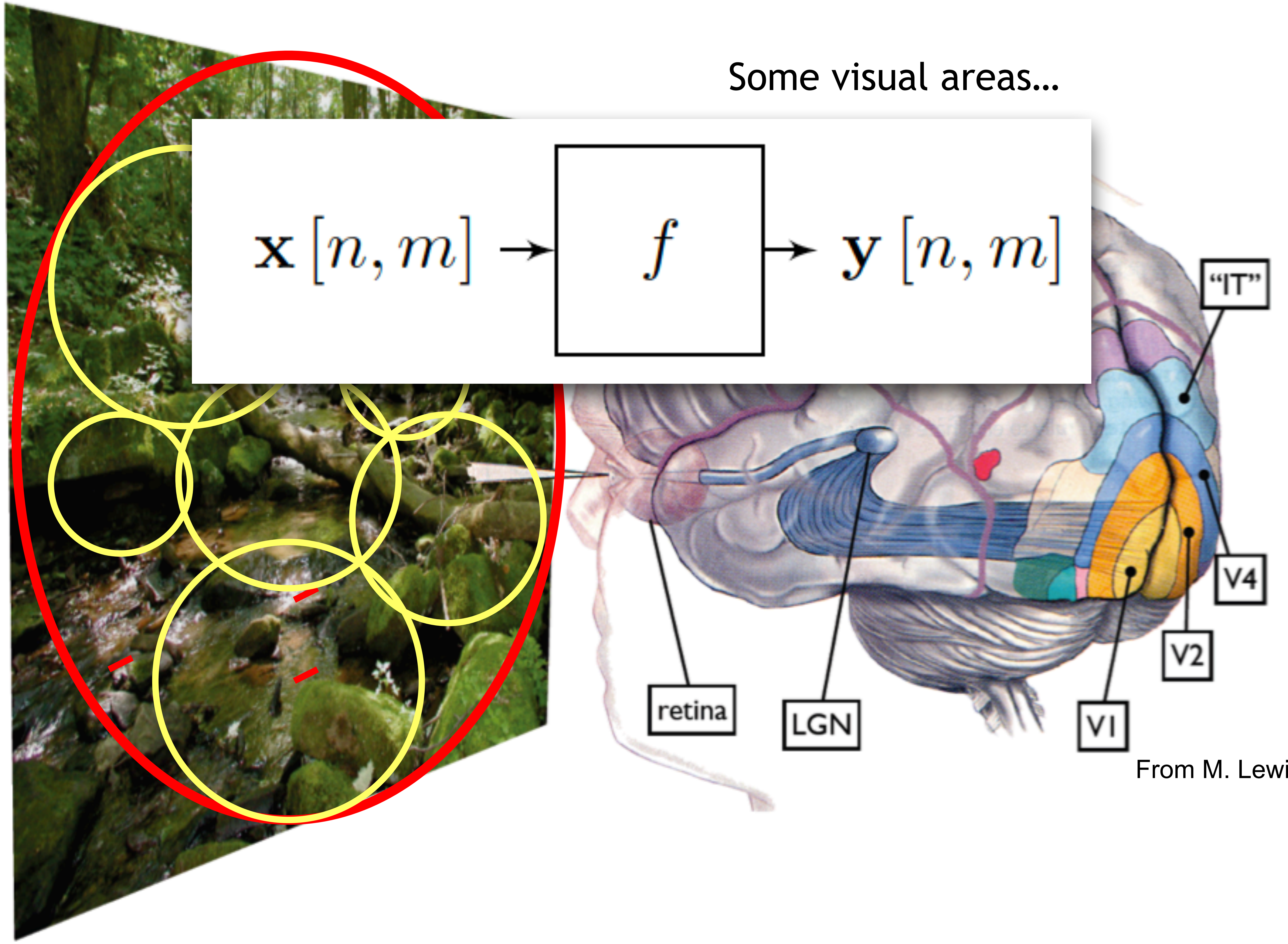
What we see



What the machine gets

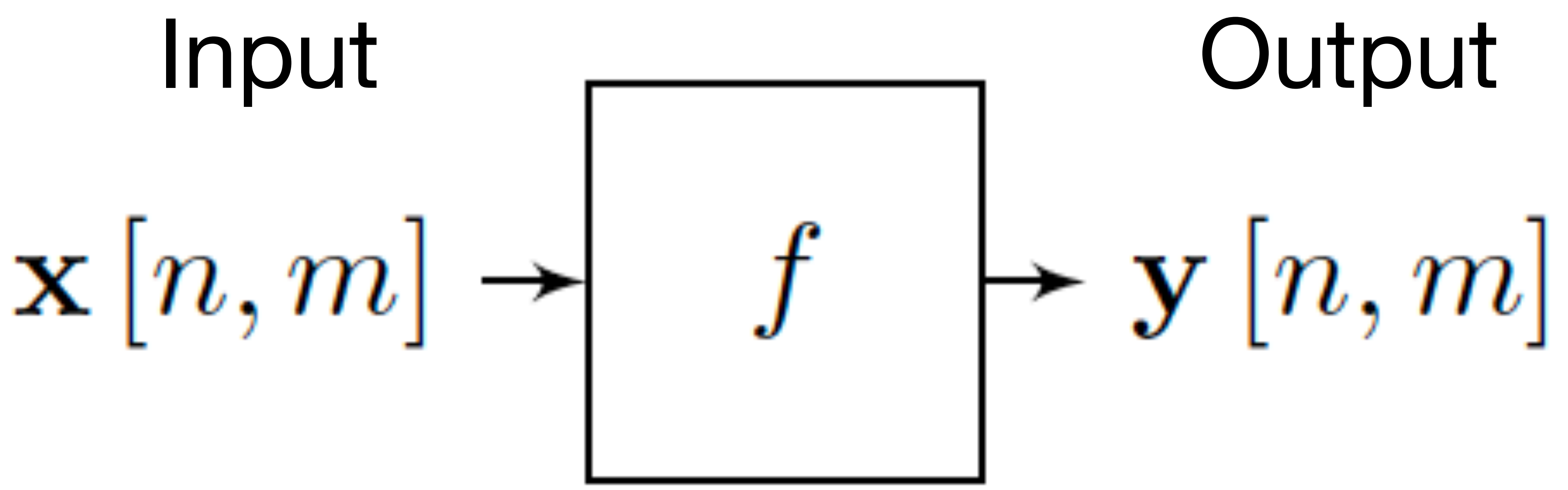
$$
\mathbf{I} = \begin{bmatrix}
160 & 175 & 171 & 168 & 168 & 172 & 164 & 158 & 167 & 173 & 167 & 163 & 162 & 164 & 160 & 159 & 163 & 162 \\
149 & 164 & 172 & 175 & 178 & 179 & 176 & 118 & 97 & 168 & 175 & 171 & 169 & 175 & 176 & 177 & 165 & 152 \\
161 & 166 & 182 & 171 & 170 & 177 & 175 & 116 & 109 & 169 & 177 & 173 & 168 & 175 & 175 & 159 & 153 & 123 \\
171 & 174 & 177 & 175 & 167 & 161 & 157 & 138 & 103 & 112 & 157 & 164 & 159 & 160 & 165 & 169 & 148 & 144 \\
163 & 163 & 162 & 165 & 167 & 164 & 178 & 167 & 77 & 55 & 134 & 170 & 167 & 162 & 164 & 175 & 168 & 160 \\
173 & 164 & 158 & 165 & 180 & 180 & 150 & 89 & 61 & 34 & 137 & 186 & 186 & 182 & 175 & 165 & 160 & 164 \\
152 & 155 & 146 & 147 & 169 & 180 & 163 & 51 & 24 & 32 & 119 & 163 & 175 & 182 & 181 & 162 & 148 & 153 \\
134 & 135 & 147 & 149 & 150 & 147 & 148 & 62 & 36 & 46 & 114 & 157 & 163 & 167 & 169 & 163 & 146 & 147 \\
135 & 132 & 131 & 125 & 115 & 129 & 132 & 74 & 54 & 41 & 104 & 156 & 152 & 156 & 164 & 156 & 141 & 144 \\
151 & 155 & 151 & 145 & 144 & 149 & 143 & 71 & 31 & 29 & 129 & 164 & 157 & 155 & 159 & 158 & 156 & 148 \\
172 & 174 & 178 & 177 & 177 & 181 & 174 & 54 & 21 & 29 & 136 & 190 & 180 & 179 & 176 & 184 & 187 & 182 \\
177 & 178 & 176 & 173 & 174 & 180 & 150 & 27 & 101 & 94 & 74 & 189 & 188 & 186 & 183 & 186 & 188 & 187 \\
160 & 160 & 163 & 163 & 161 & 167 & 100 & 45 & 169 & 166 & 59 & 136 & 184 & 176 & 175 & 177 & 185 & 186 \\
147 & 150 & 153 & 155 & 160 & 155 & 56 & 111 & 182 & 180 & 104 & 84 & 168 & 172 & 171 & 164 & 168 & 167 \\
184 & 182 & 178 & 175 & 179 & 133 & 86 & 191 & 201 & 204 & 191 & 79 & 172 & 220 & 217 & 205 & 209 & 200 \\
184 & 187 & 192 & 182 & 124 & 32 & 109 & 168 & 171 & 167 & 163 & 51 & 105 & 203 & 209 & 203 & 210 & 205 \\
191 & 198 & 203 & 197 & 175 & 149 & 169 & 189 & 190 & 173 & 160 & 145 & 156 & 202 & 199 & 201 & 205 & 202 \\
153 & 149 & 153 & 155 & 173 & 182 & 179 & 177 & 182 & 177 & 182 & 185 & 179 & 177 & 167 & 176 & 182 & 180
\end{bmatrix}
$$

Some visual areas...



From M. Lewicky

Signals and systems



One important class of systems is the set of linear systems.

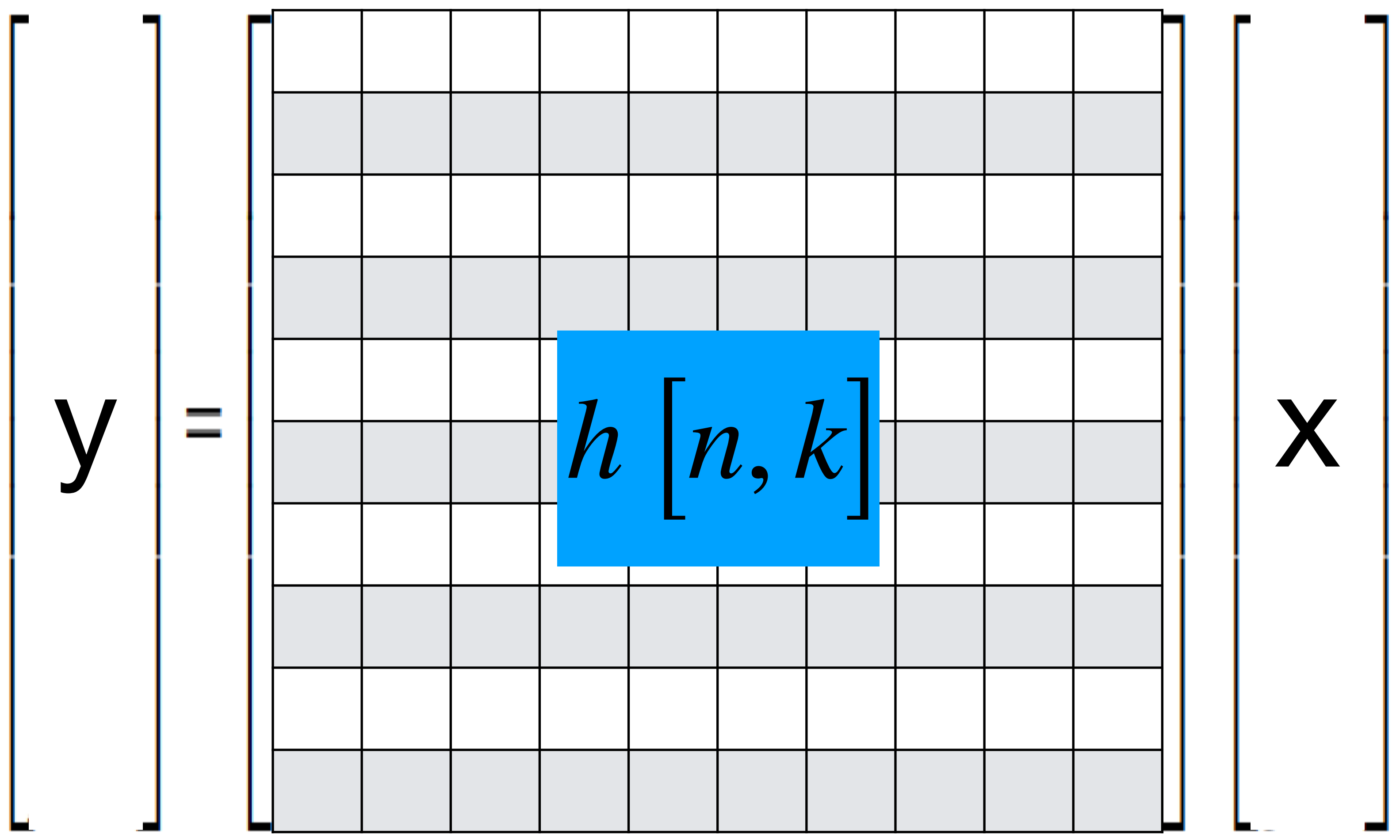
A function f is linear if it satisfies:

$$f(\alpha \mathbf{x}) = \alpha f(\mathbf{x})$$

$$f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$

Linear system: $\mathbf{y} = f(\mathbf{x})$

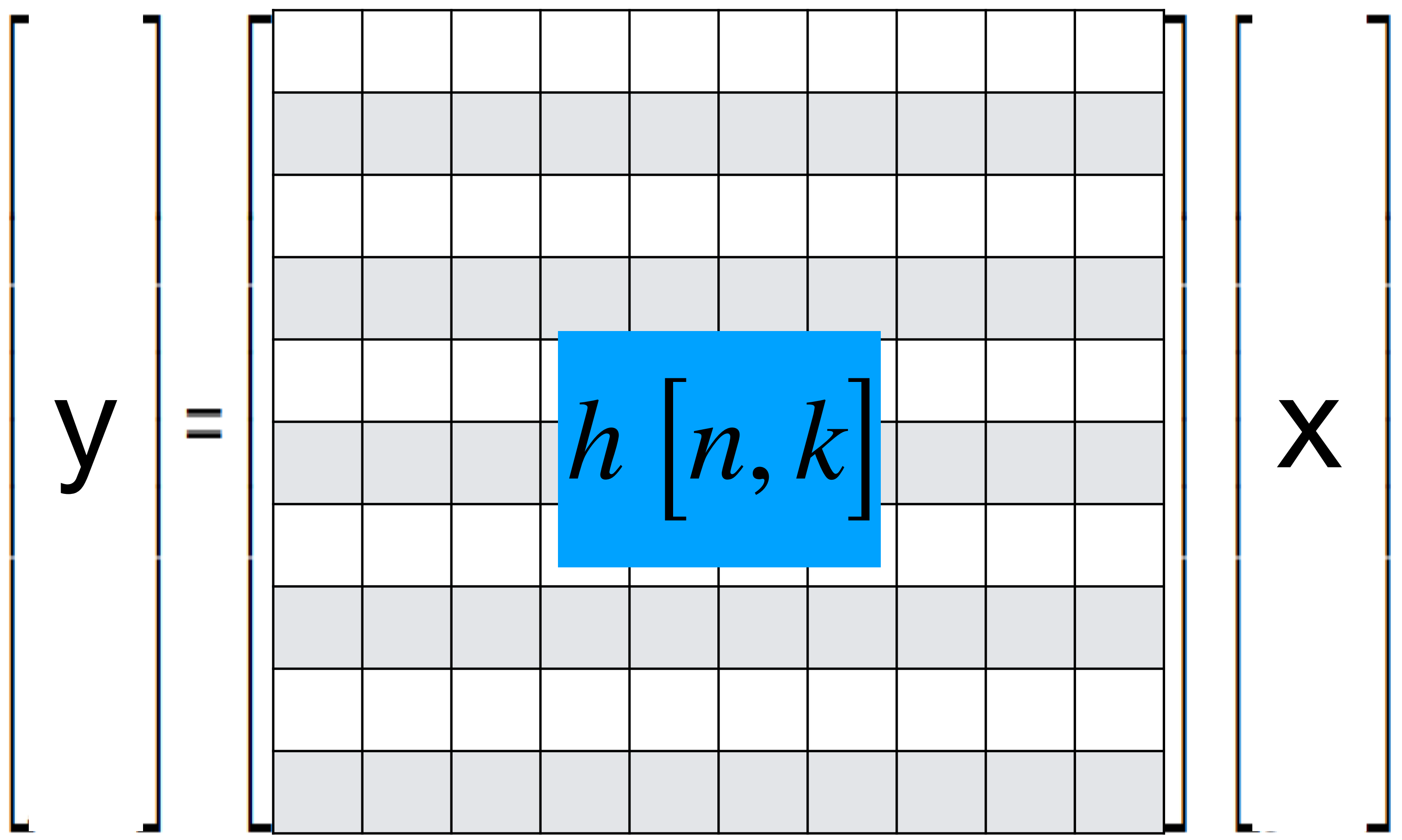
A linear function f can be written
as a matrix multiplication:



n indexes rows,
k indexes columns

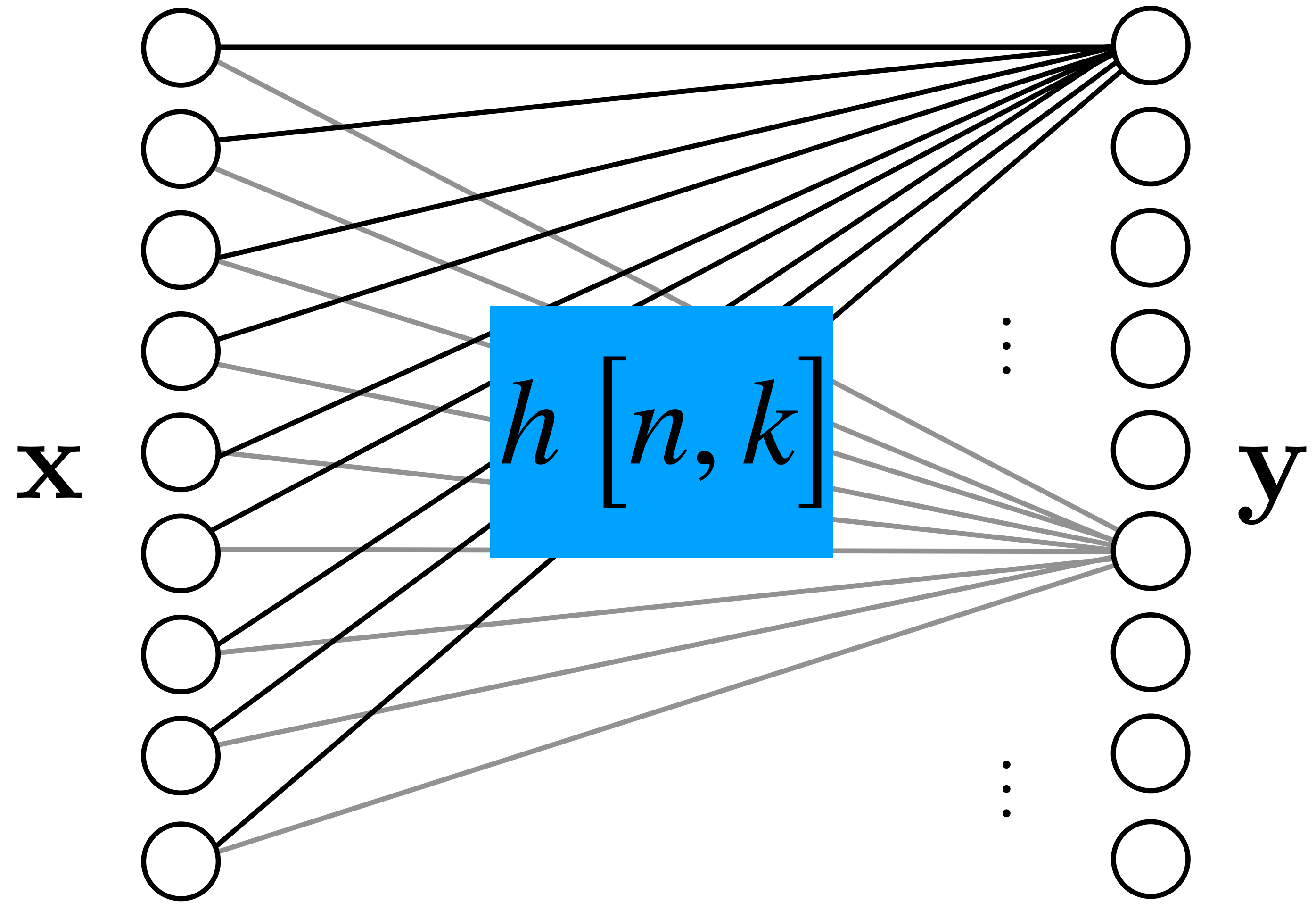
Linear system: $\mathbf{y} = f(\mathbf{x})$

A linear function f can be written as a matrix multiplication:



n indexes rows,
k indexes columns

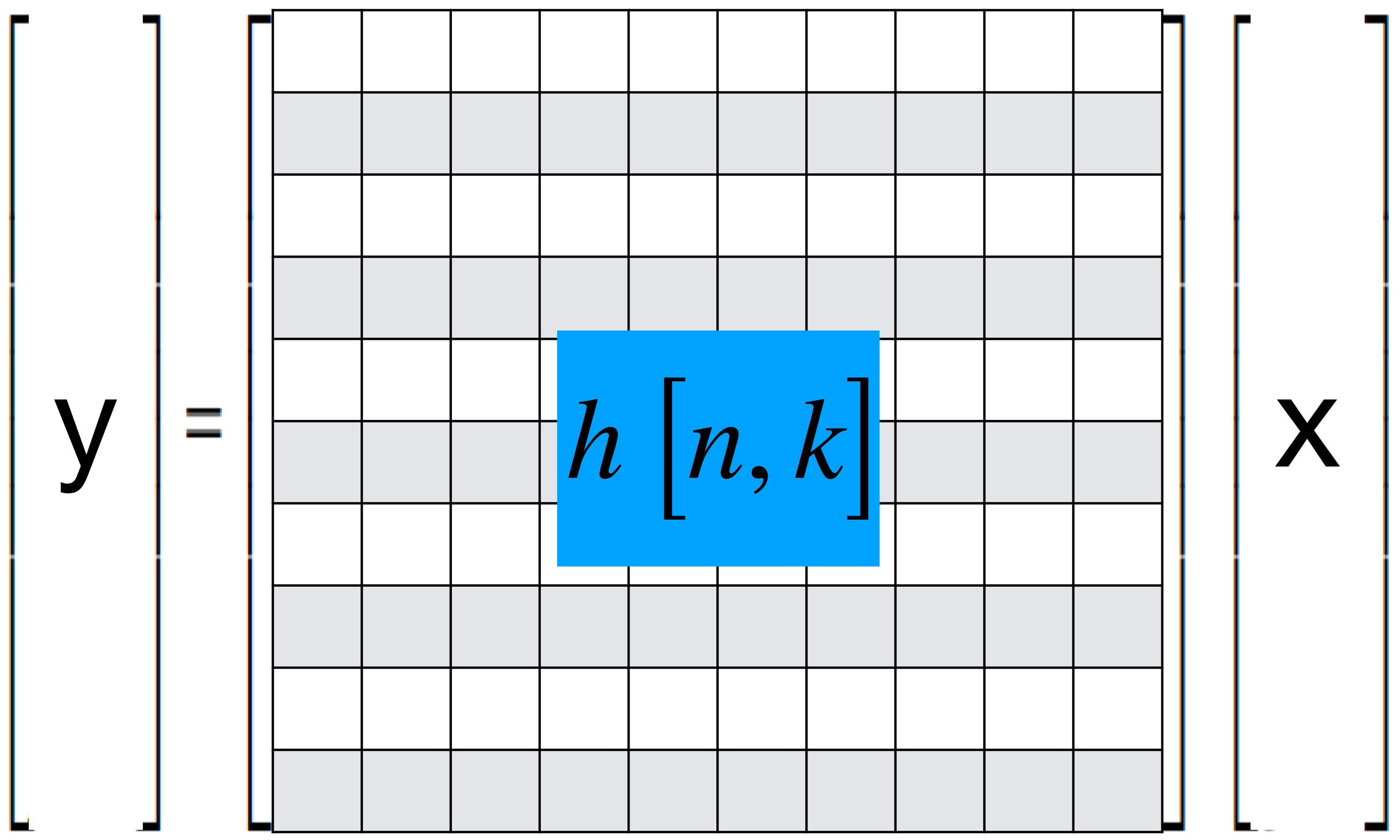
It can also be represented as a fully connected linear neural network



$h[n,k]$ Is the strength of the connection between x[k] and y[n]

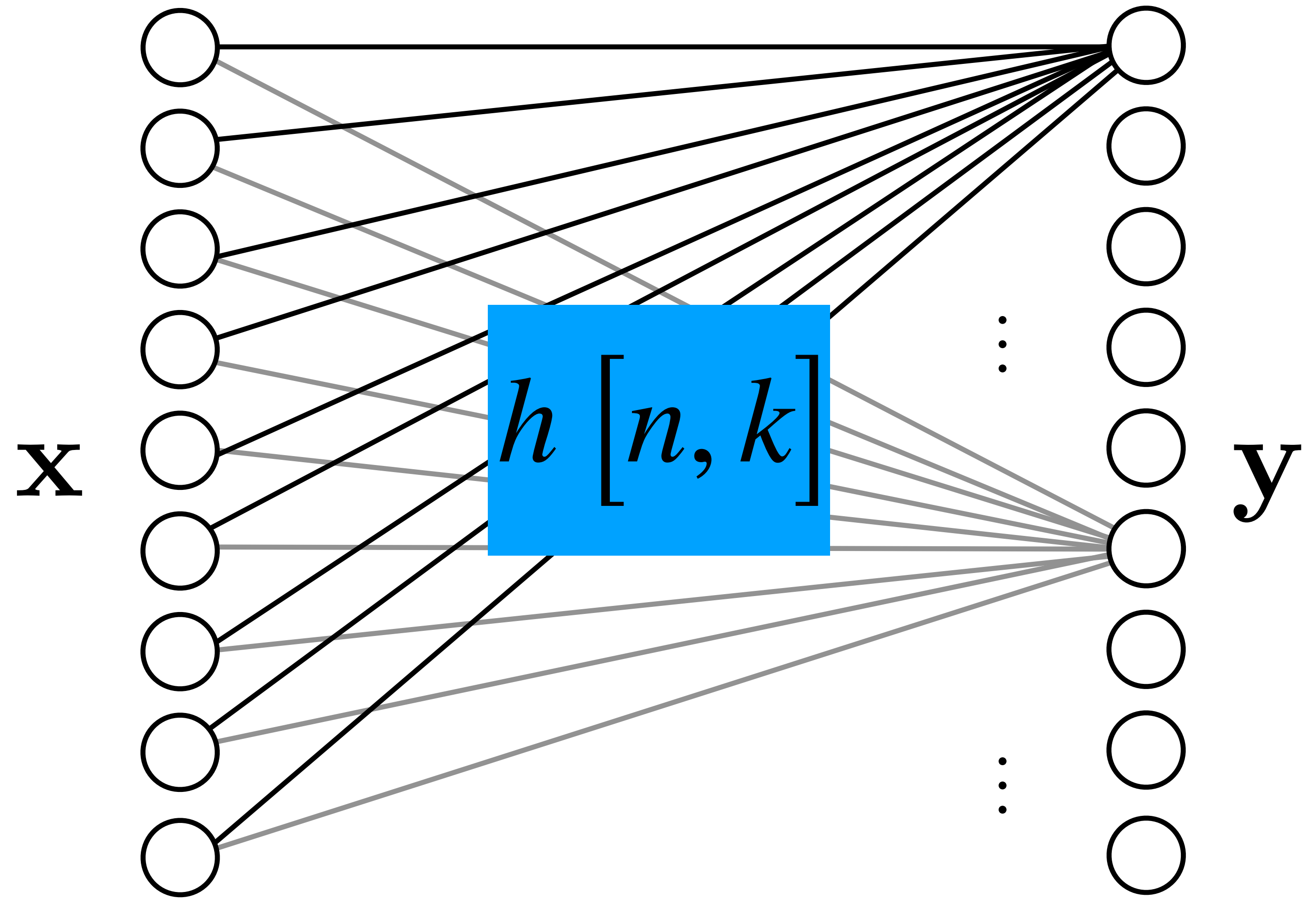
Linear system: $\mathbf{y} = f(\mathbf{x})$

A linear function f can be written as a matrix multiplication:



n indexes rows,
k indexes columns

It can also be represented as a fully connected linear neural network



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

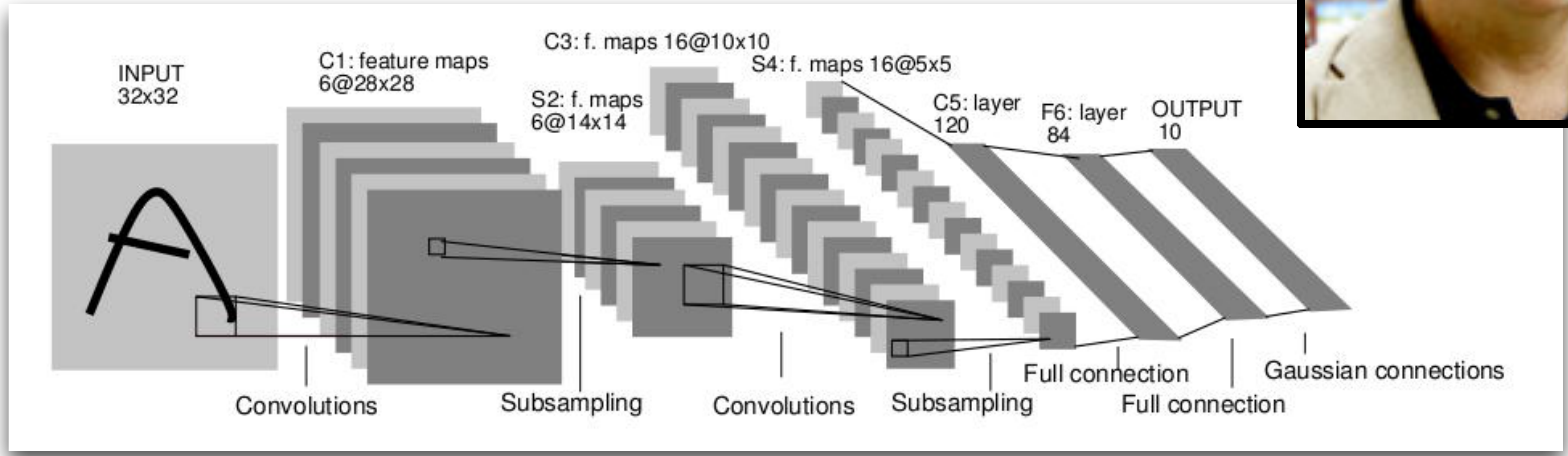
$h[n,k]$ Is the strength of the connection between x[k] and y[n]

Quiz: what operation is linear?

Quiz: what operation is linear?

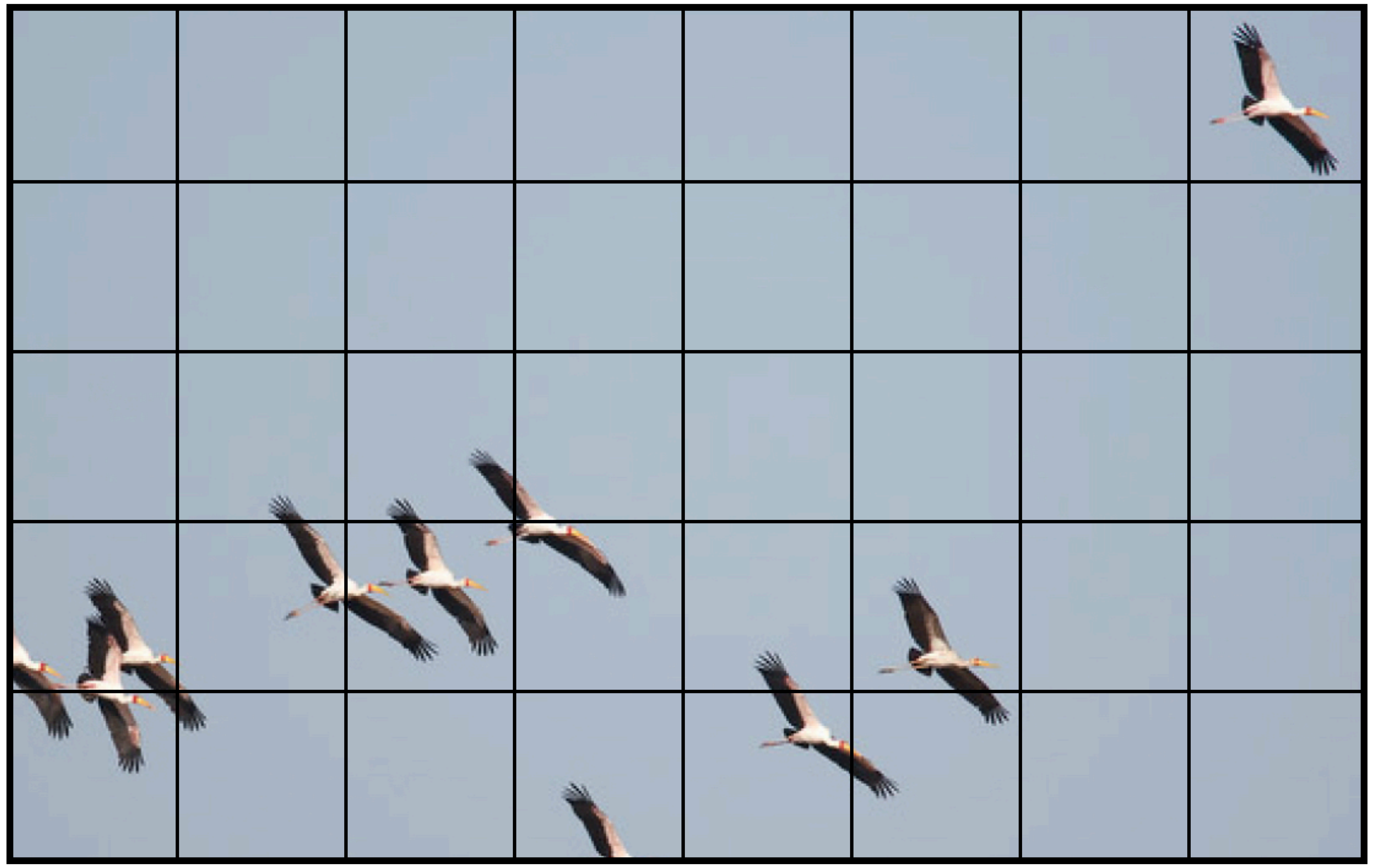
Convolutional Neural Networks

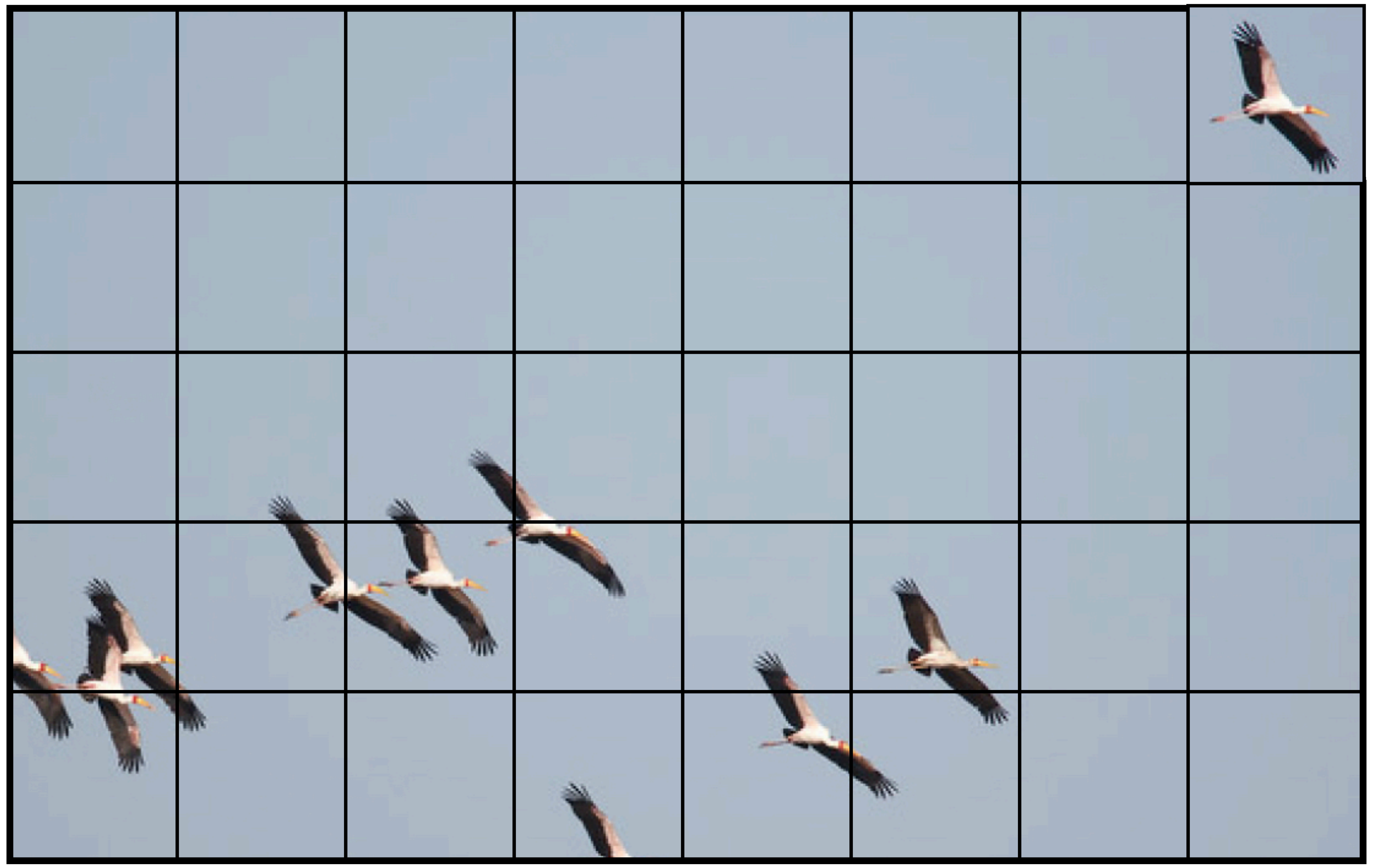
- Neural network with specialized connectivity structure

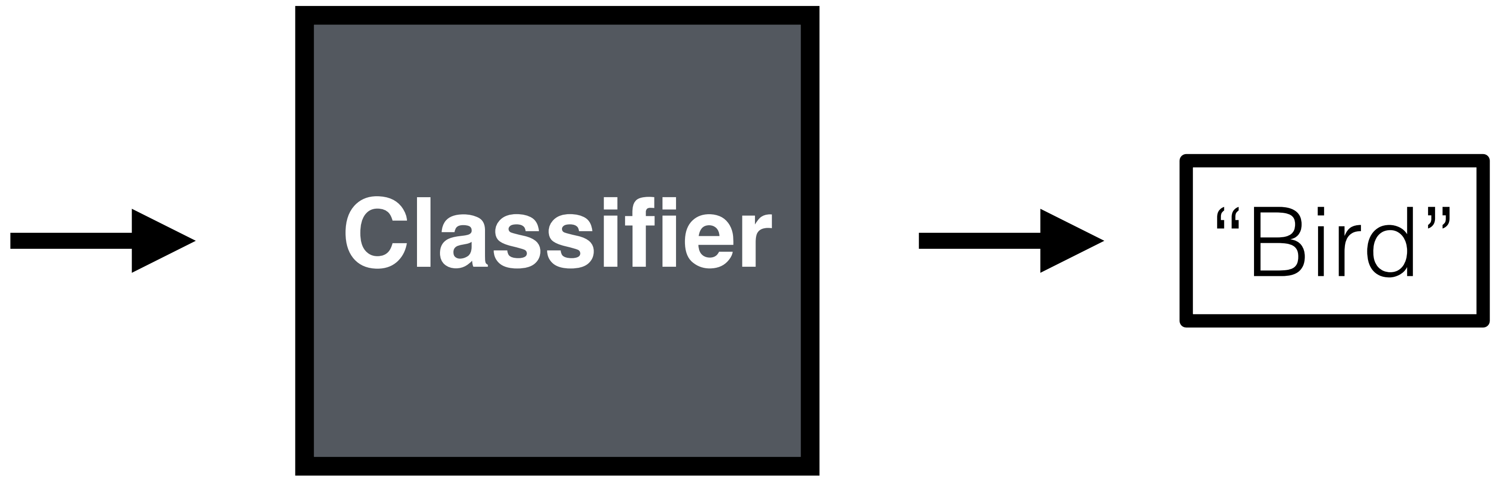


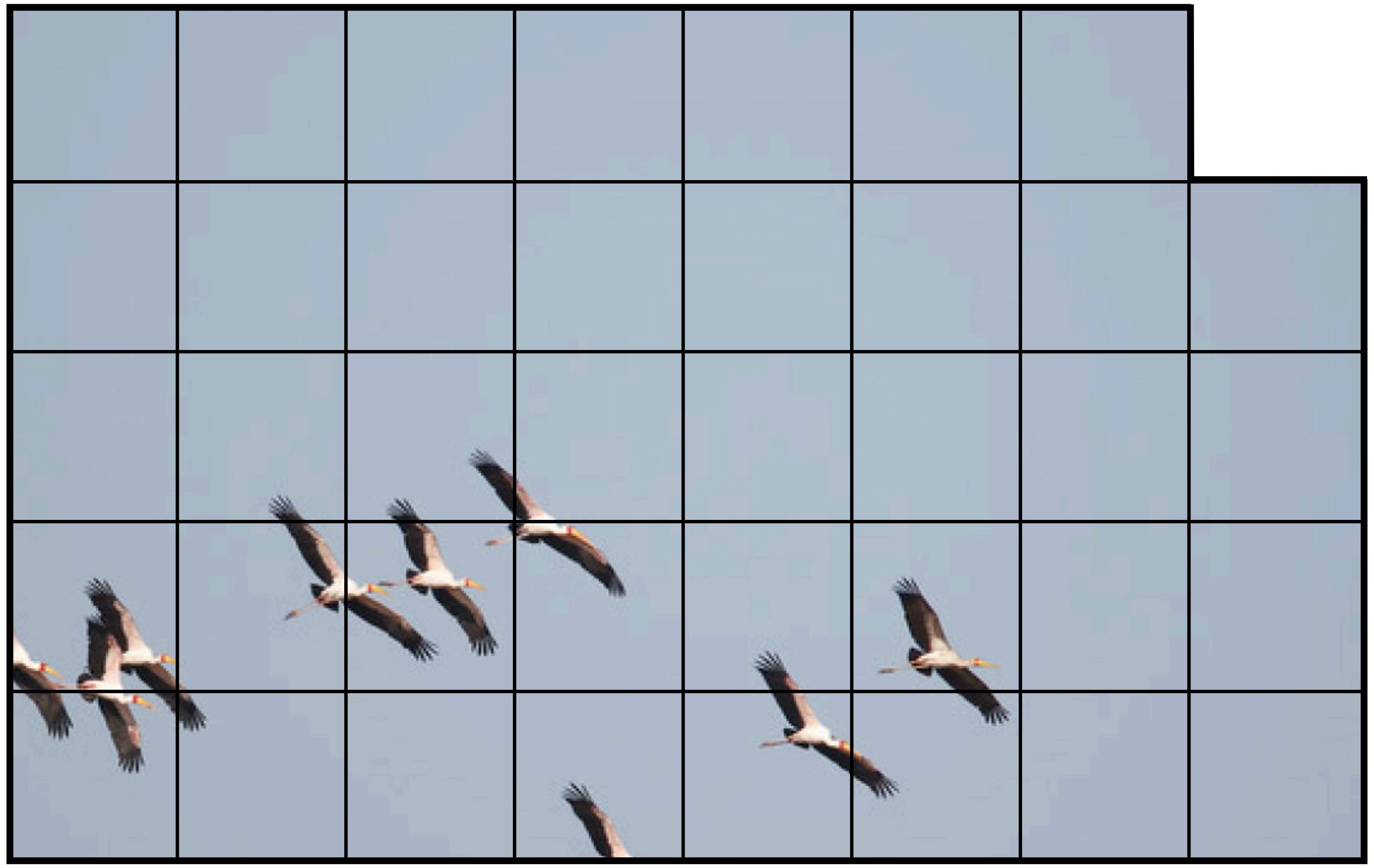
LeCun et al. 1989

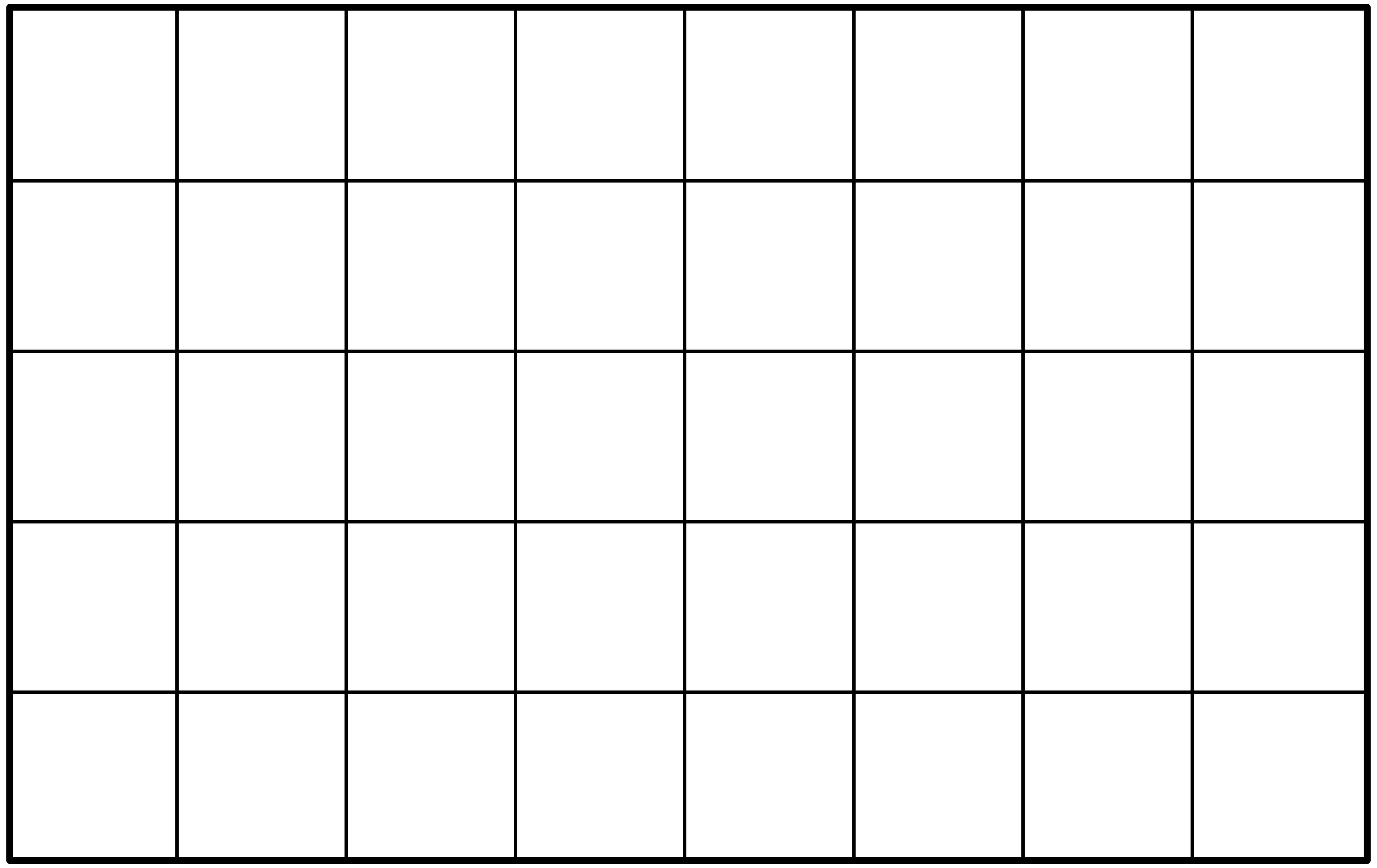
We need translation invariance



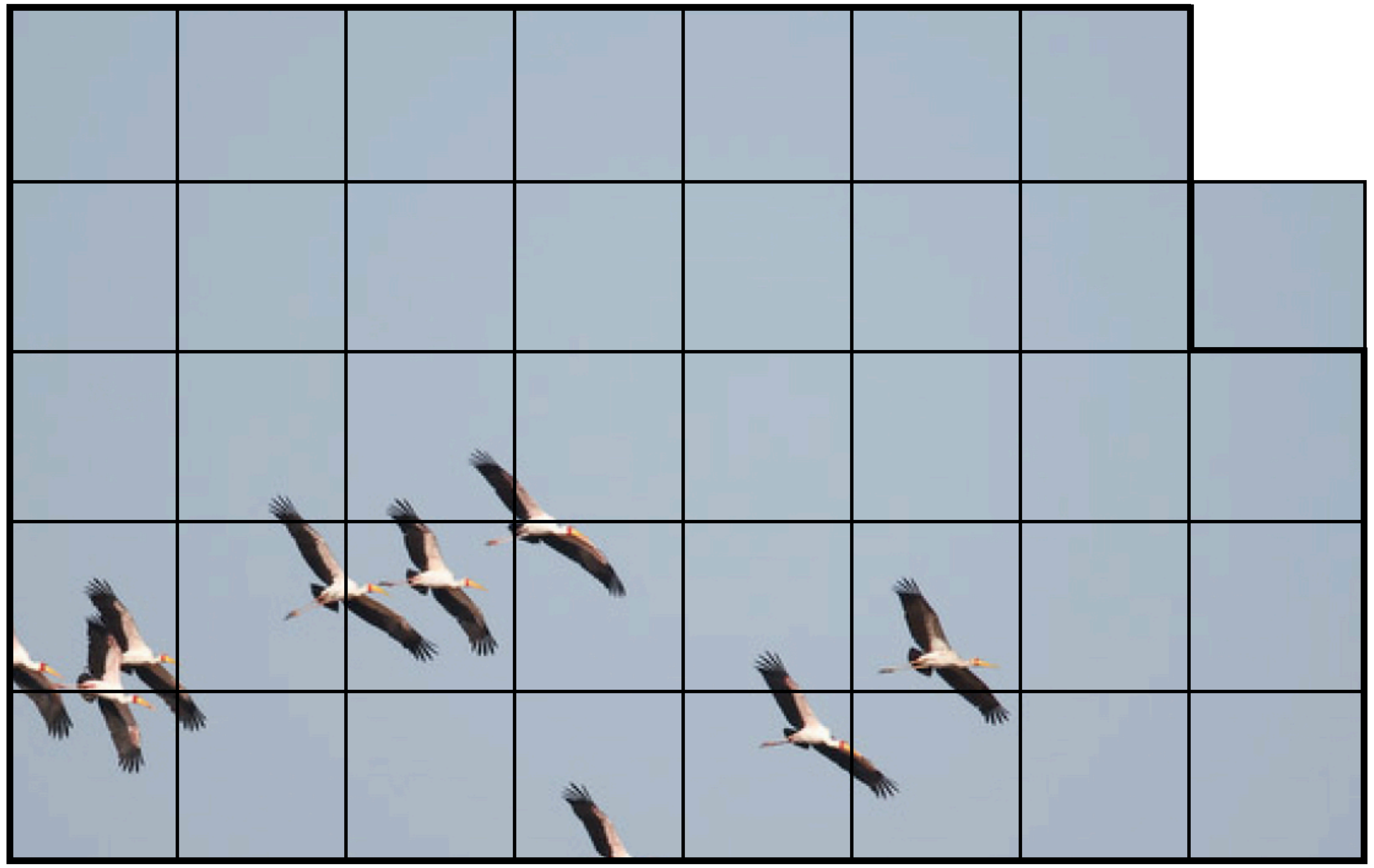


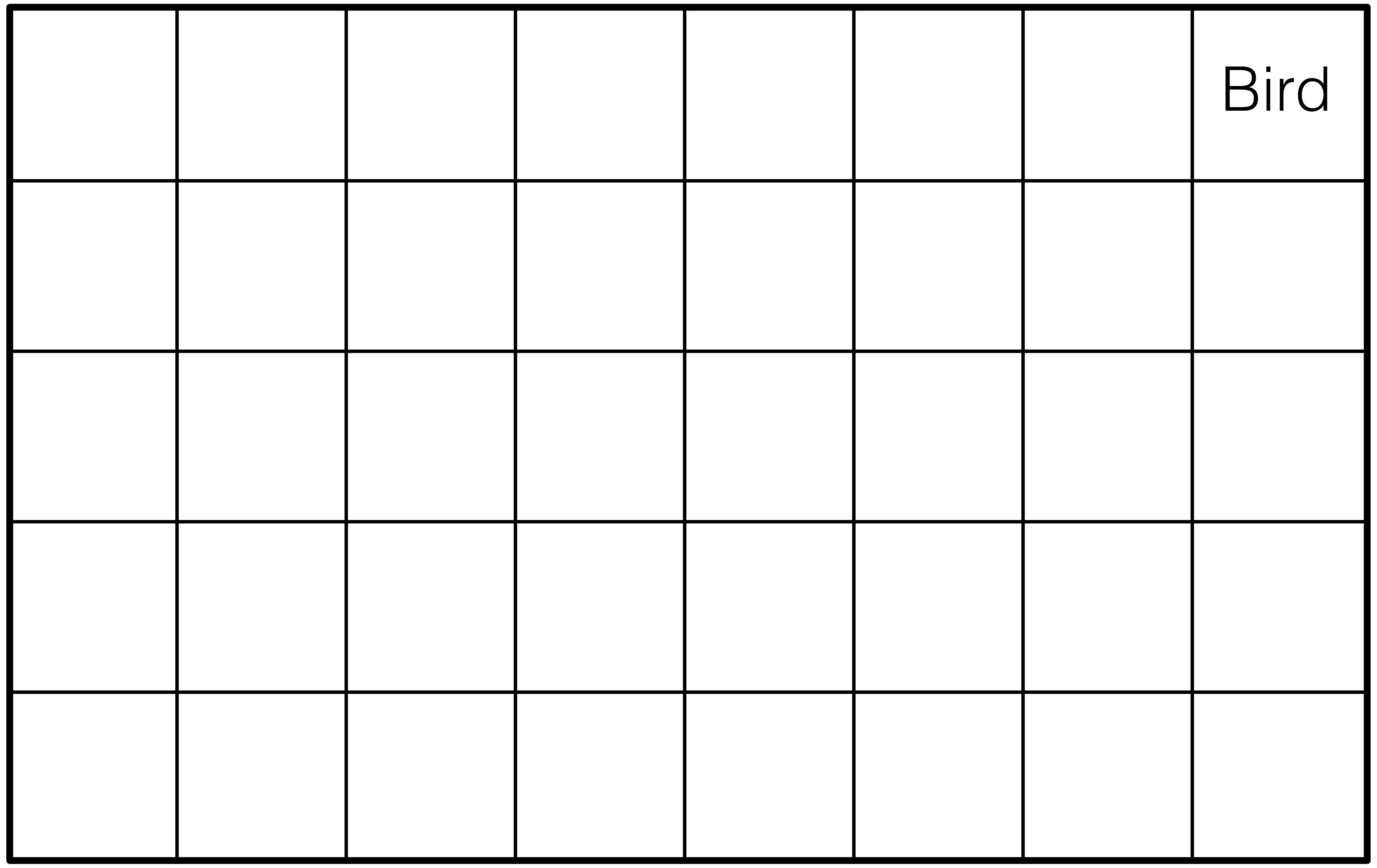

Classifier
“Bird”



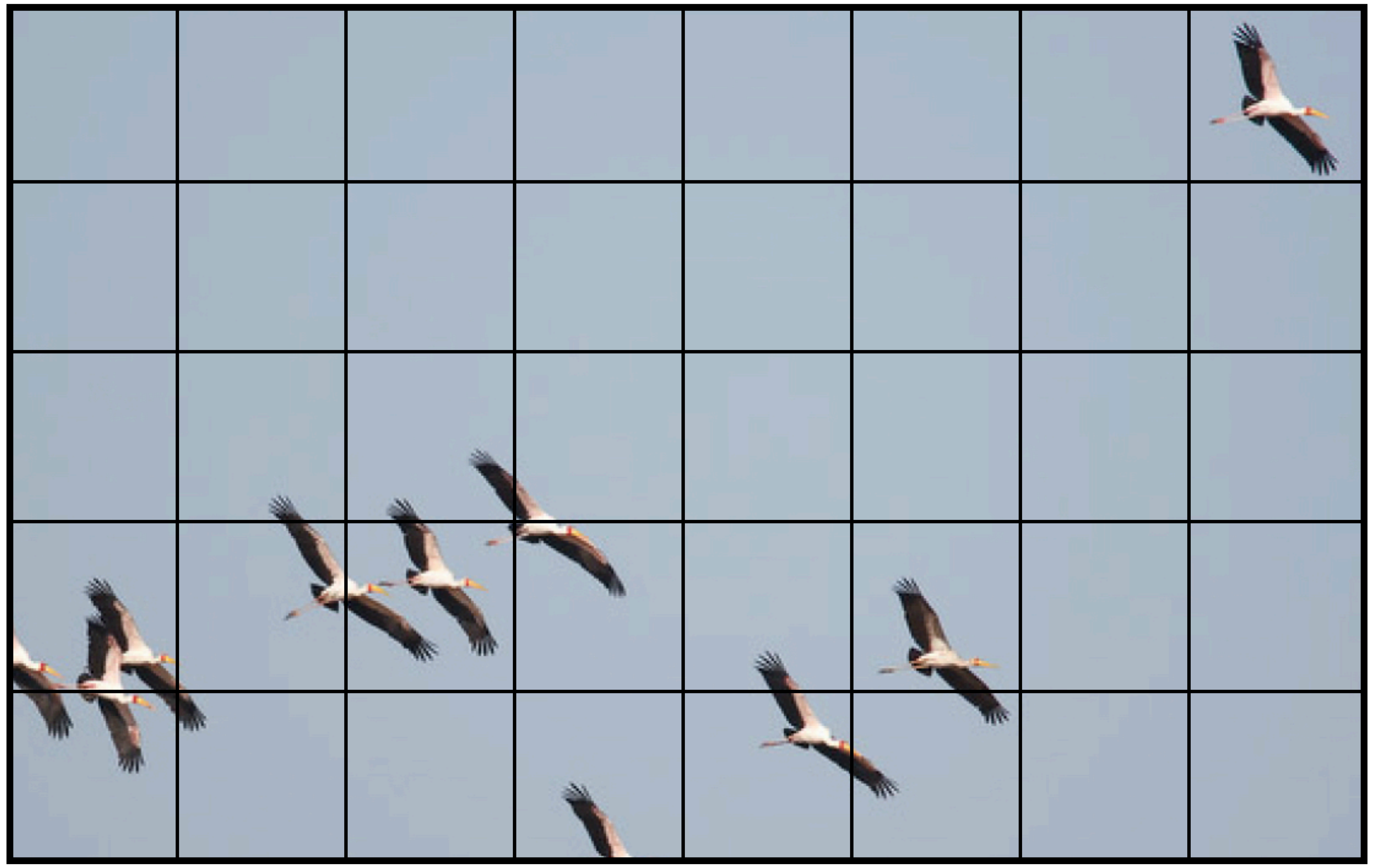



Classifier
"Bird"

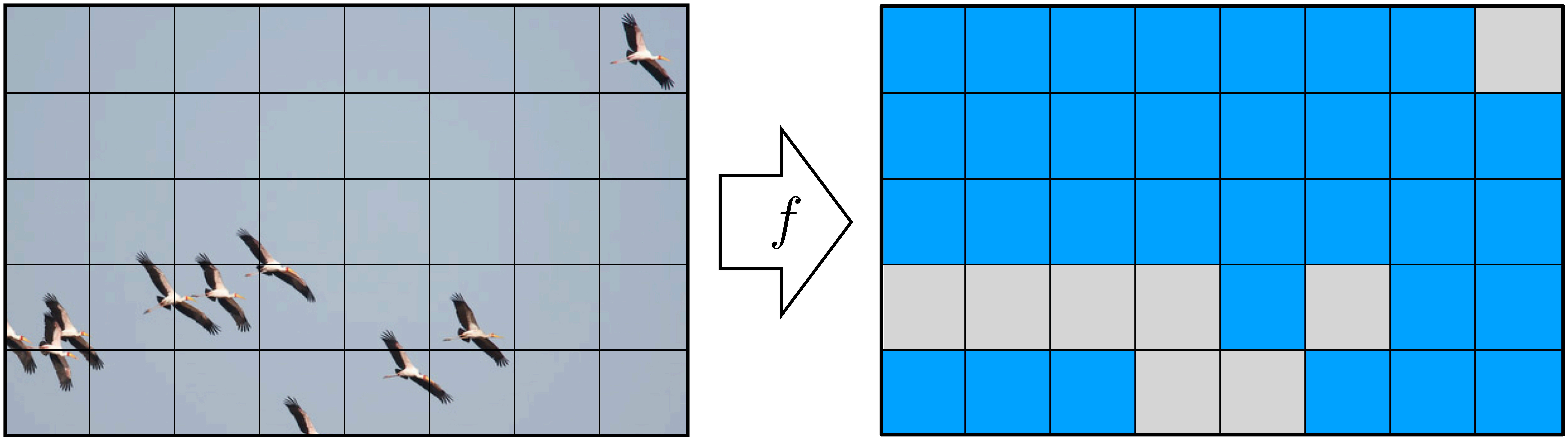



Bird

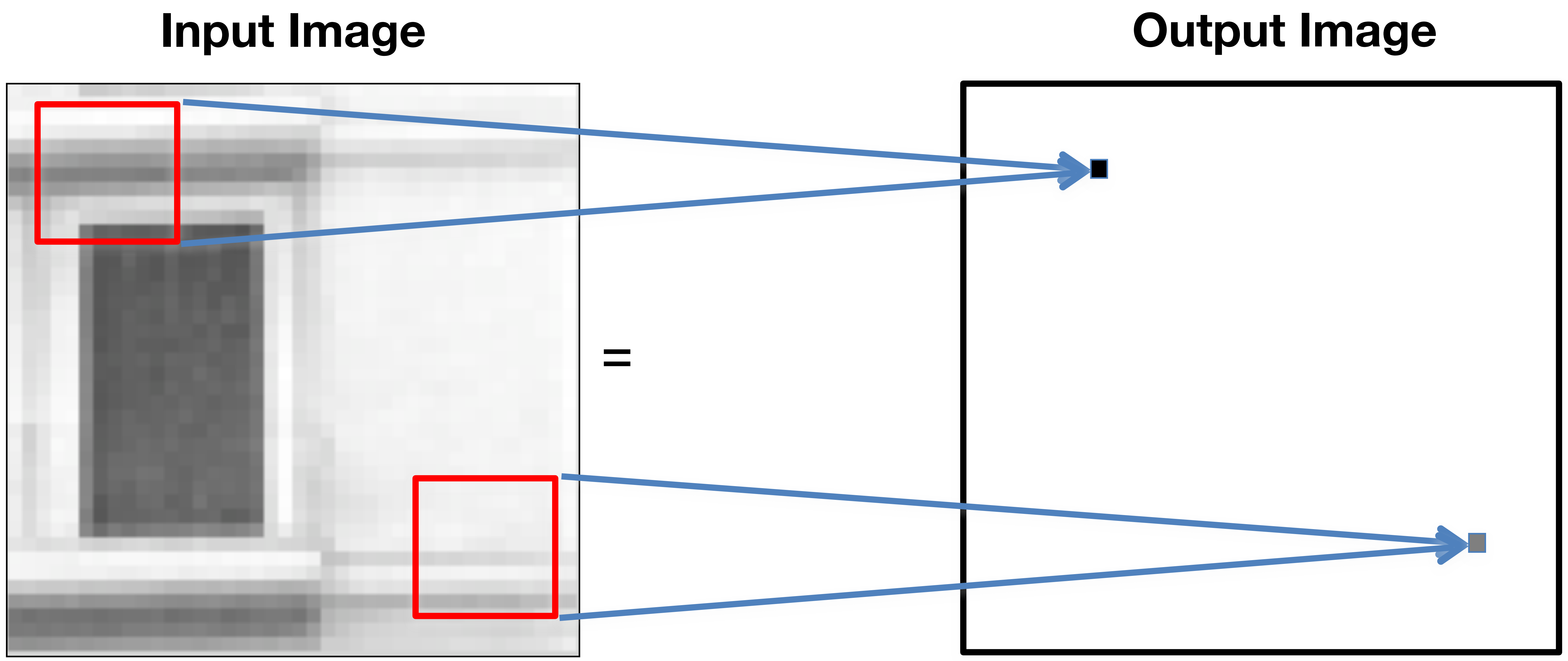

Classifier
"Sky"



Sky	Sky	Sky	Sky	Sky	Sky	Sky	Bird
Sky	Sky	Sky	Sky	Sky	Sky	Sky	Sky
Sky	Sky	Sky	Sky	Sky	Sky	Sky	Sky
Bird	Bird	Bird	Sky	Bird	Sky	Sky	Sky
Sky	Sky	Sky	Bird	Sky	Sky	Sky	Sky


f

Convolution



The same weighting occurs within each window

Convolution: running example

x

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

Convolution: running example

x

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

Convolution kernel

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

Convolution: running example

x

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

Convolution kernel

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

=

y

Convolution: running example

x

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

Convolution kernel

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

=

y

Convolution: running example

x

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

Convolution kernel

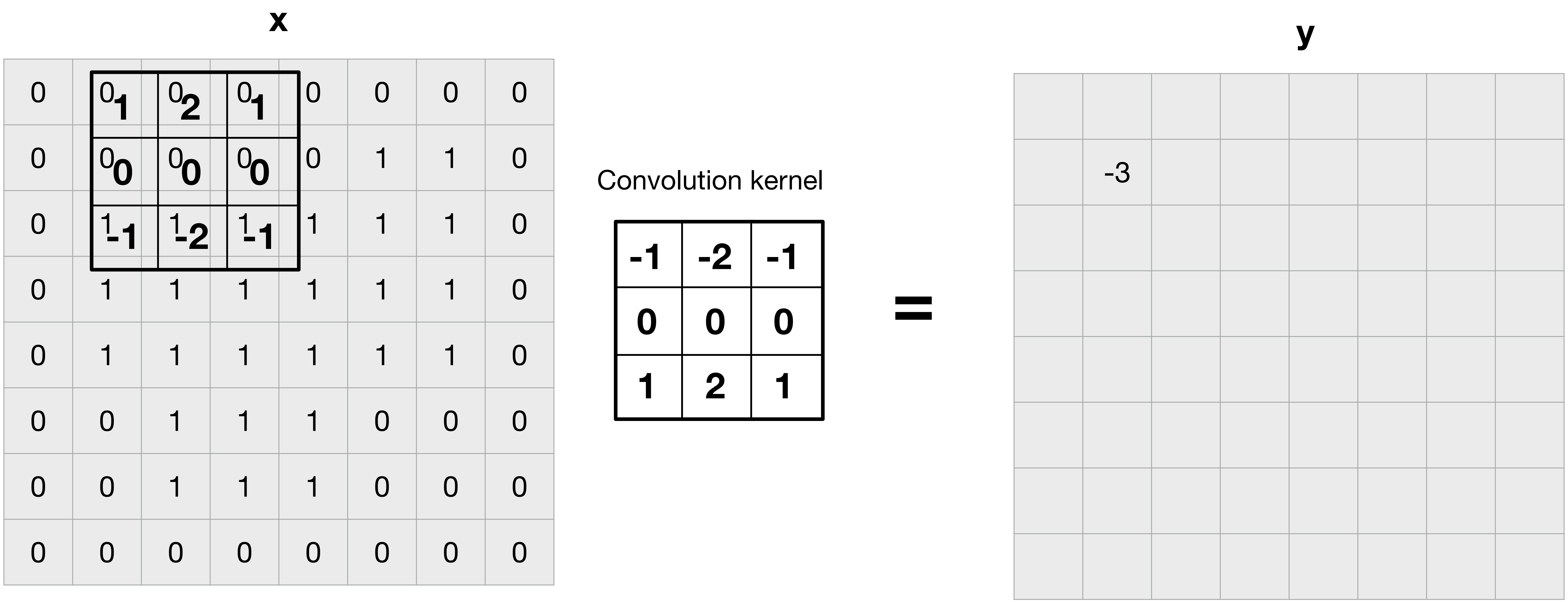
-1	**-2**	**-1**
0	**0**	**0**
1	**2**	**1**

=

y

	-3						

Convolution: running example



Convolution: running example

x

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

Convolution kernel

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

=

y

	-3	-4					

Convolution: running example

x

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

Convolution kernel

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

=

y

	-3	-4	-4				

Convolution: running example

x

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

Convolution kernel

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

=

y

	-3	-4	-4	-4			

Convolution: running example

x

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

Convolution kernel

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

=

y

	-3	-4	-4	-4	-4		

Convolution: running example

x

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

Convolution kernel

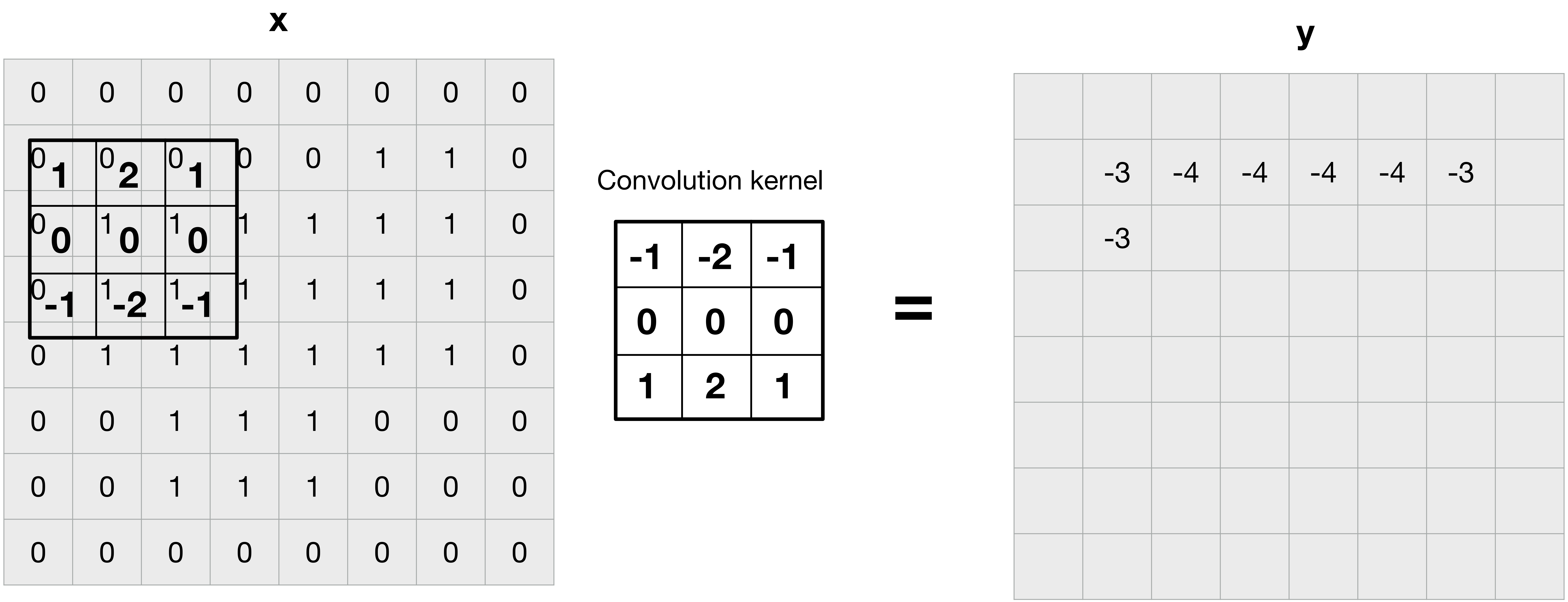
-1	**-2**	**-1**
0	**0**	**0**
1	**2**	**1**

=

y

	-3	-4	-4	-4	-4	-3	

Convolution: running example



Convolution: running example

x

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

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

Convolution kernel

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

=

y

	-3	-4	-4	-4	-4	-3	
	-3	-4	-4	-3	-1	0	
	0	0	0	0	0	0	
	2	1	0	1	3	3	
	2	1	0	1	3	3	
	1	3	4	3	1	0	

Convolution: running example

x

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

Convolution kernel

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

=

y

?							
	-3	-4	-4	-4	-4	-3	
	-3	-4	-4	-3	-1	0	
	0	0	0	0	0	0	
	2	1	0	1	3	3	
	2	1	0	1	3	3	
	1	3	4	3	1	0	

Convolution: running example

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

x

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

Convolution kernel

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

=

y

?							
	-3	-4	-4	-4	-4	-3	
	-3	-4	-4	-3	-1	0	
	0	0	0	0	0	0	
	2	1	0	1	3	3	
	2	1	0	1	3	3	
	1	3	4	3	1	0	

Convolution: running example

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

x

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

Convolution kernel

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

=

y

0							
	-3	-4	-4	-4	-4	-3	
	-3	-4	-4	-3	-1	0	
	0	0	0	0	0	0	
	2	1	0	1	3	3	
	2	1	0	1	3	3	
	1	3	4	3	1	0	

Convolution vs cross-correlation

Convolution $$y[m,n] = x \circ h = \sum_{k,l=-N}^{N} x[m-k, n-l]\, h[k,l]$$

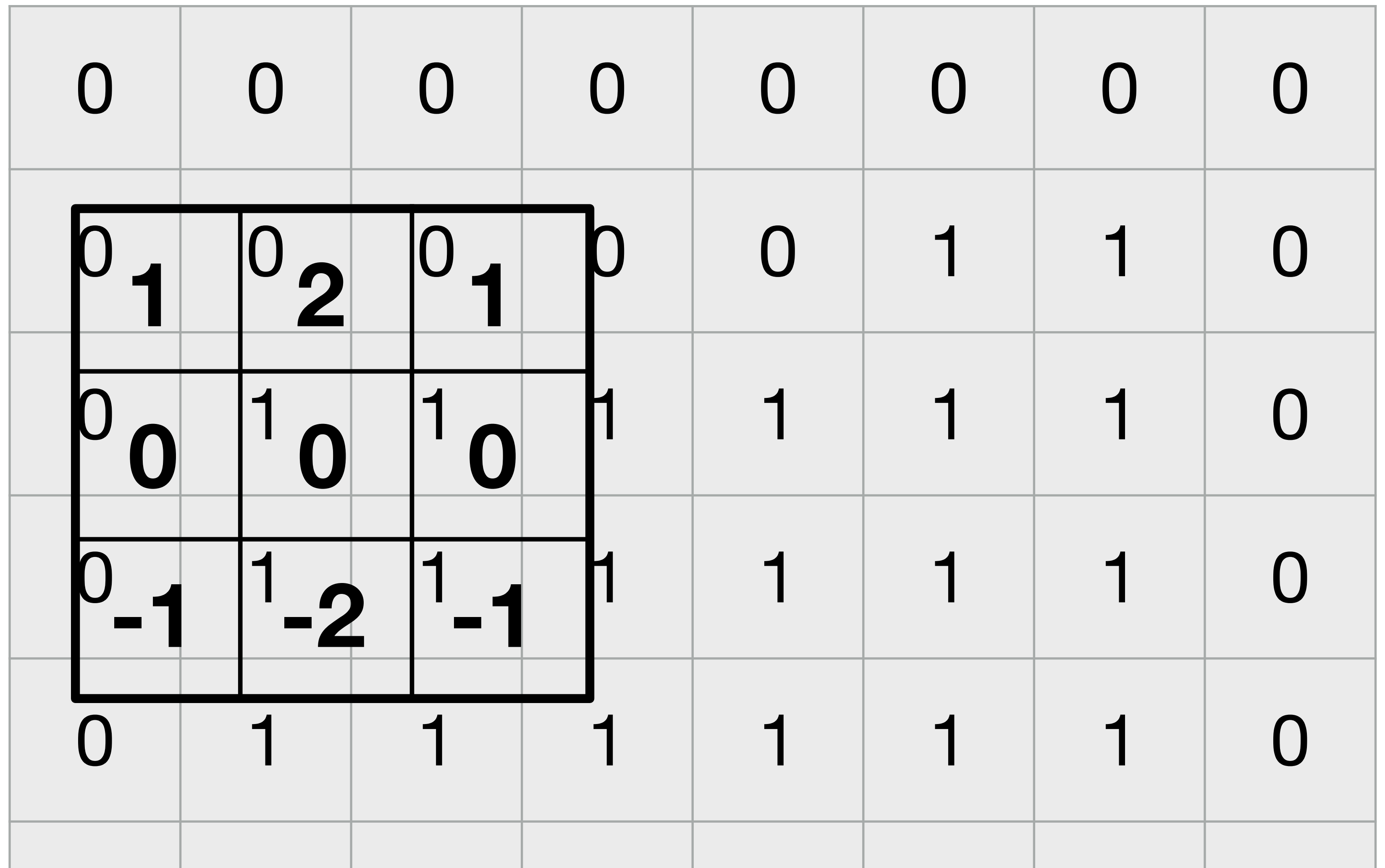
Cross-correlation $$y[m,n] = x * h = \sum_{k,l=-N}^{N} x[m+k, n+l]\, h[k,l]$$

In the convolution, the kernel *h* is inverted left-right and up-down, while in the cross-correlation is not

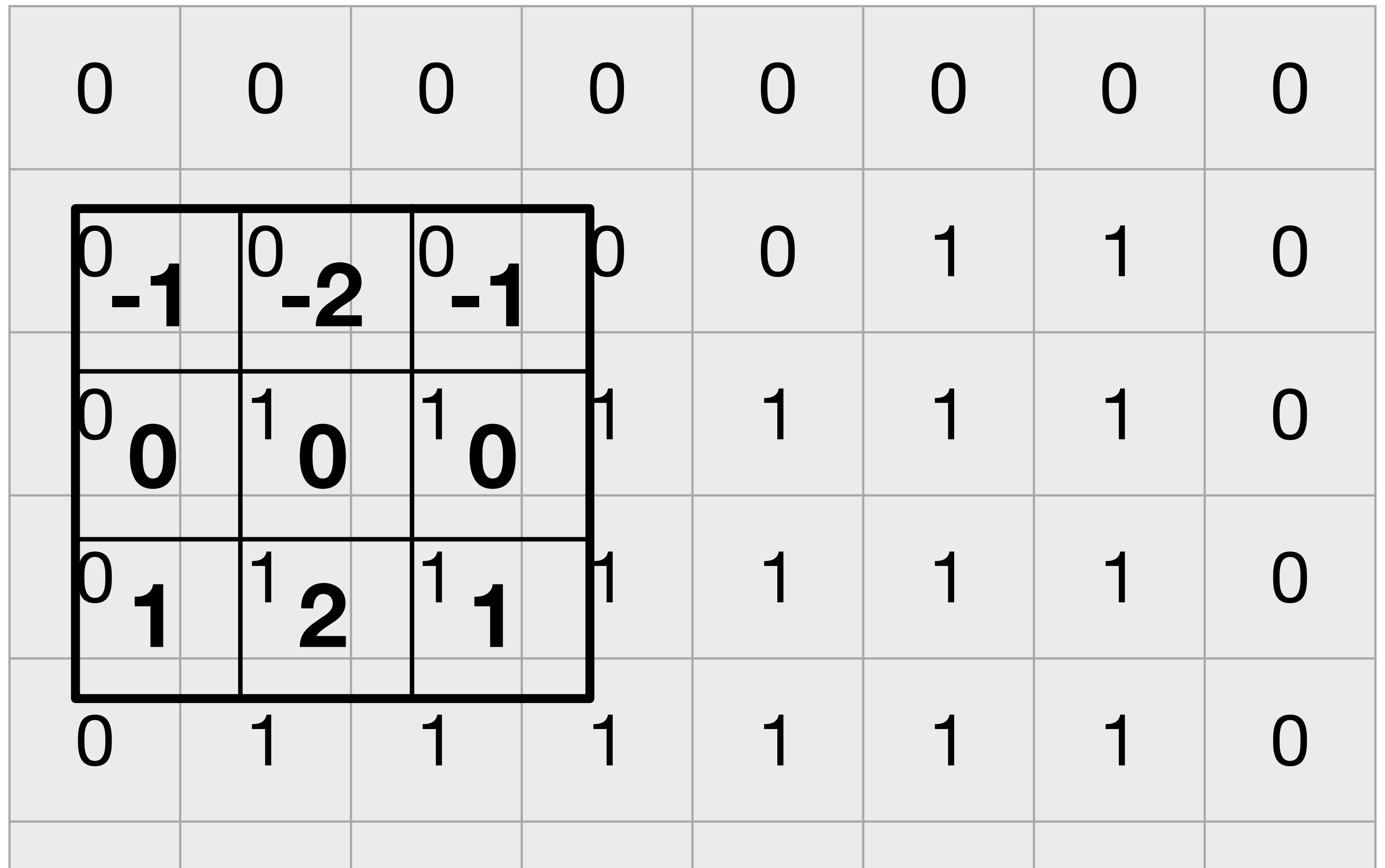
Kernel

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

Convolution



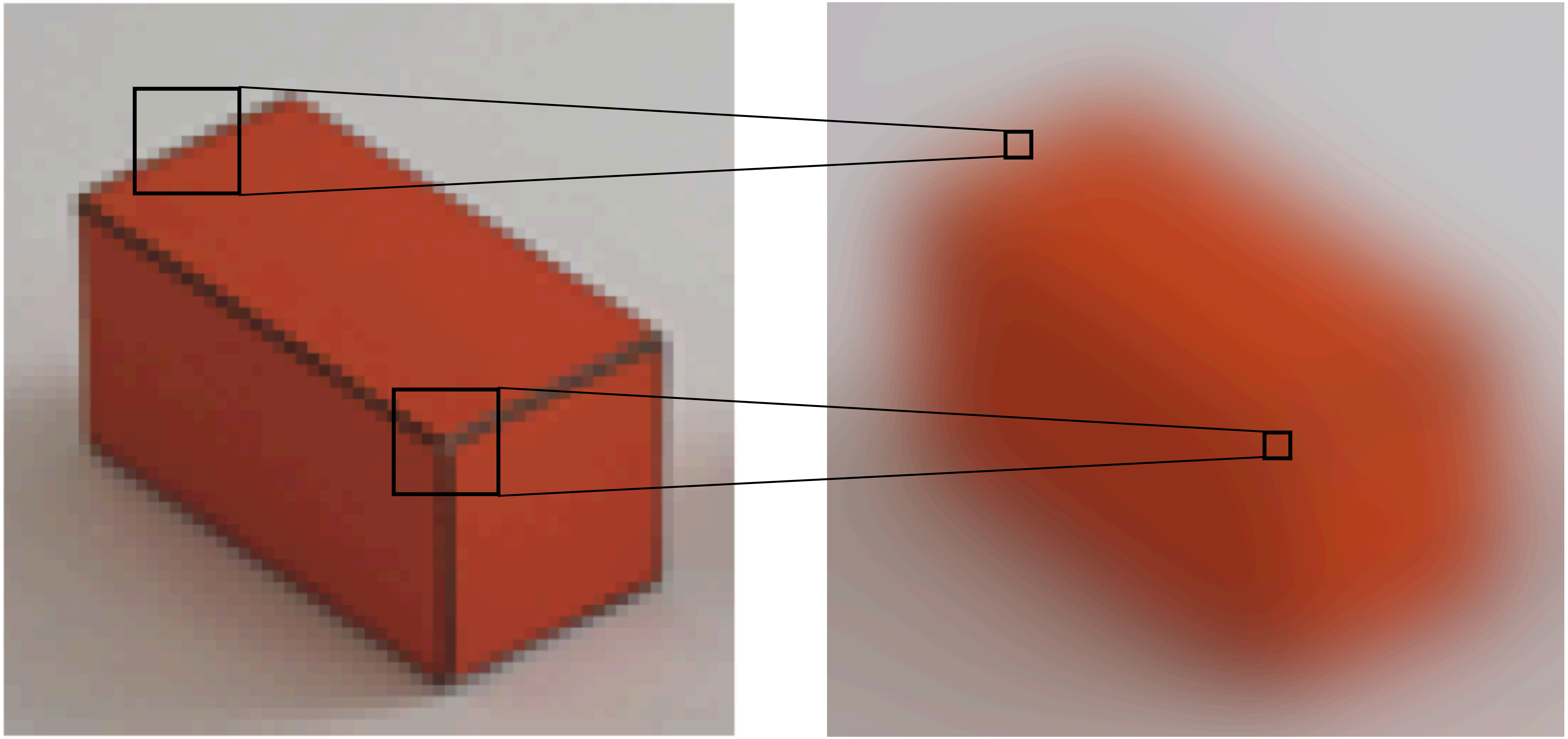
Cross-correlation



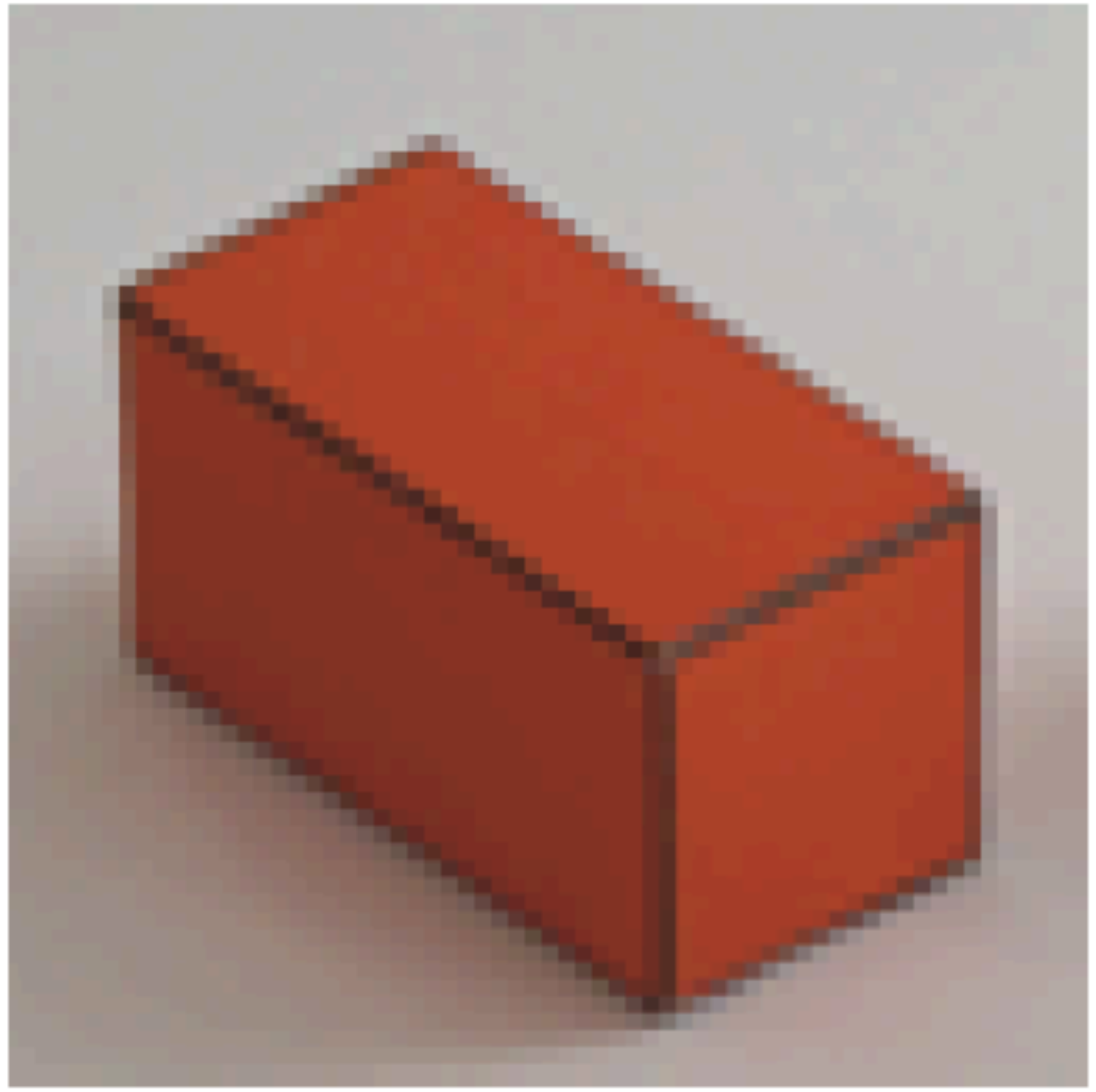
Quiz: what operation is the result of a convolution?

Quiz: what operation is the result of a convolution?

Examples



Defocus/blurring

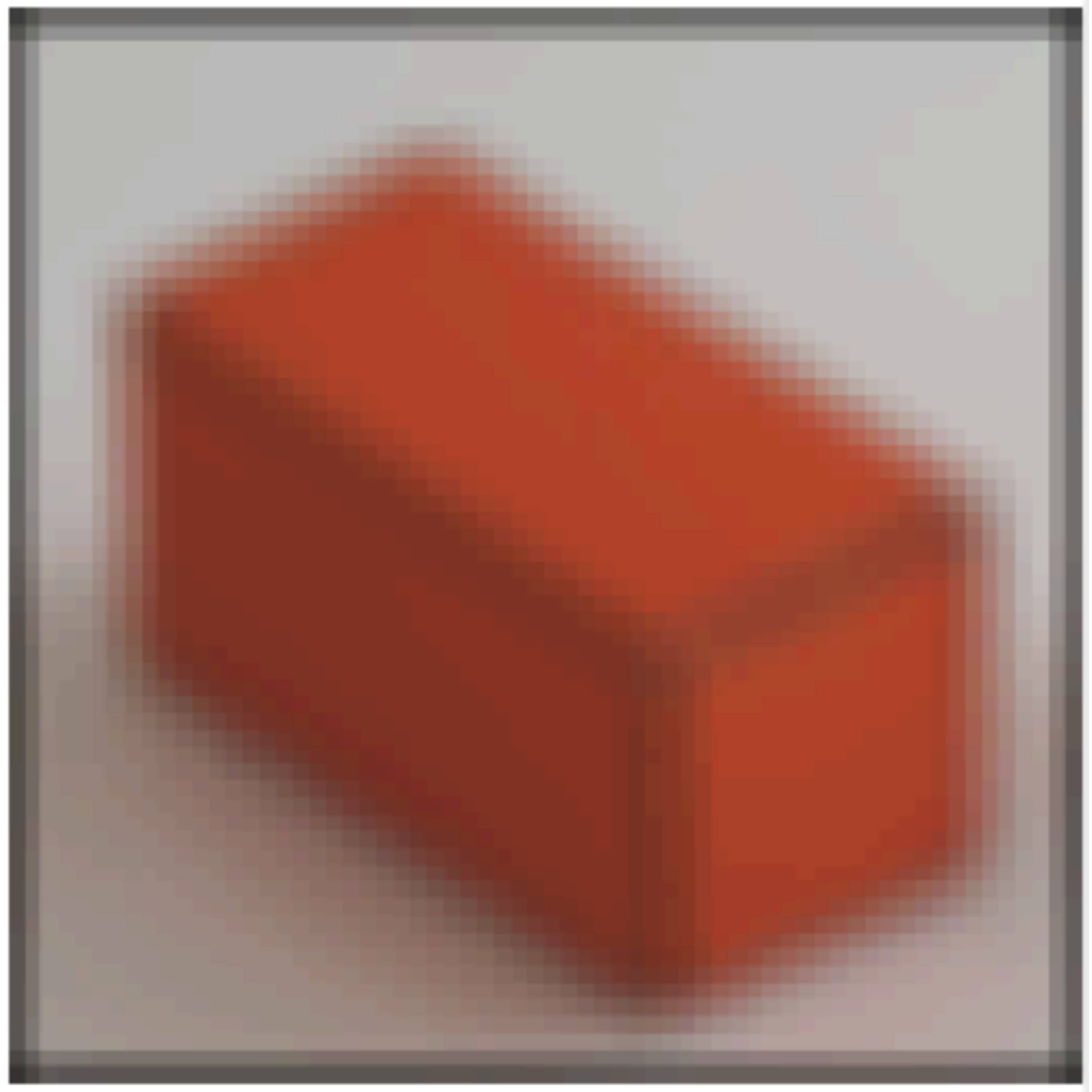


$\circ$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

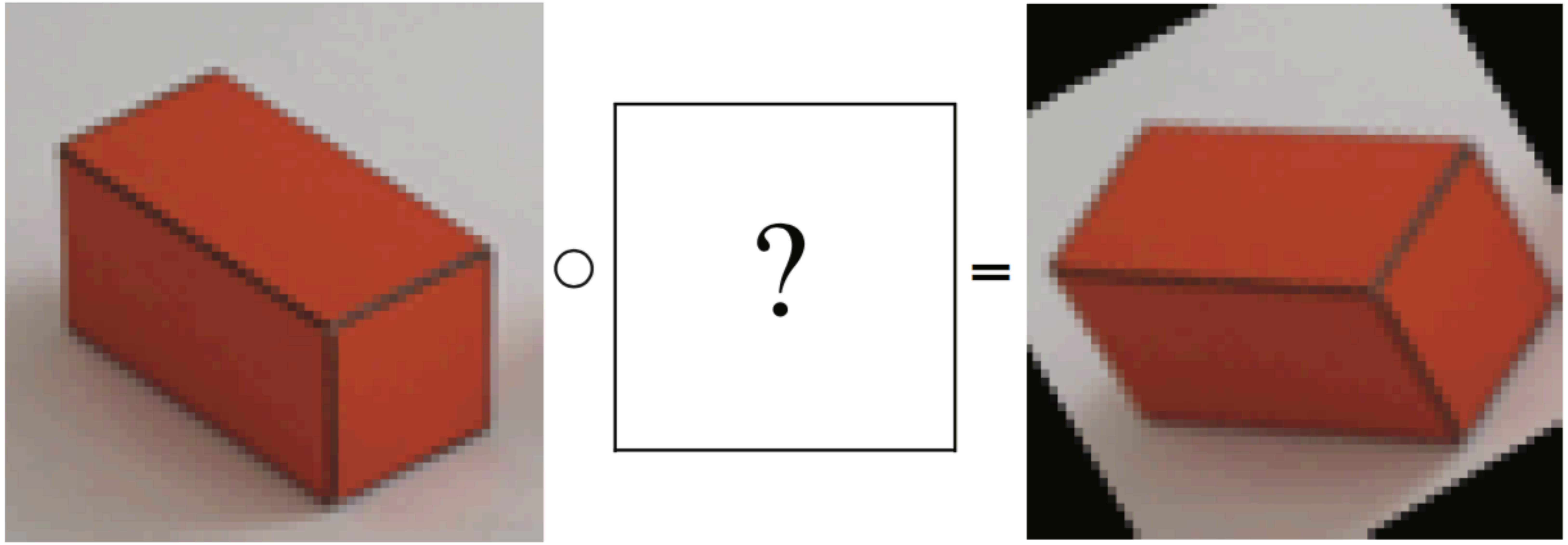
/25

=

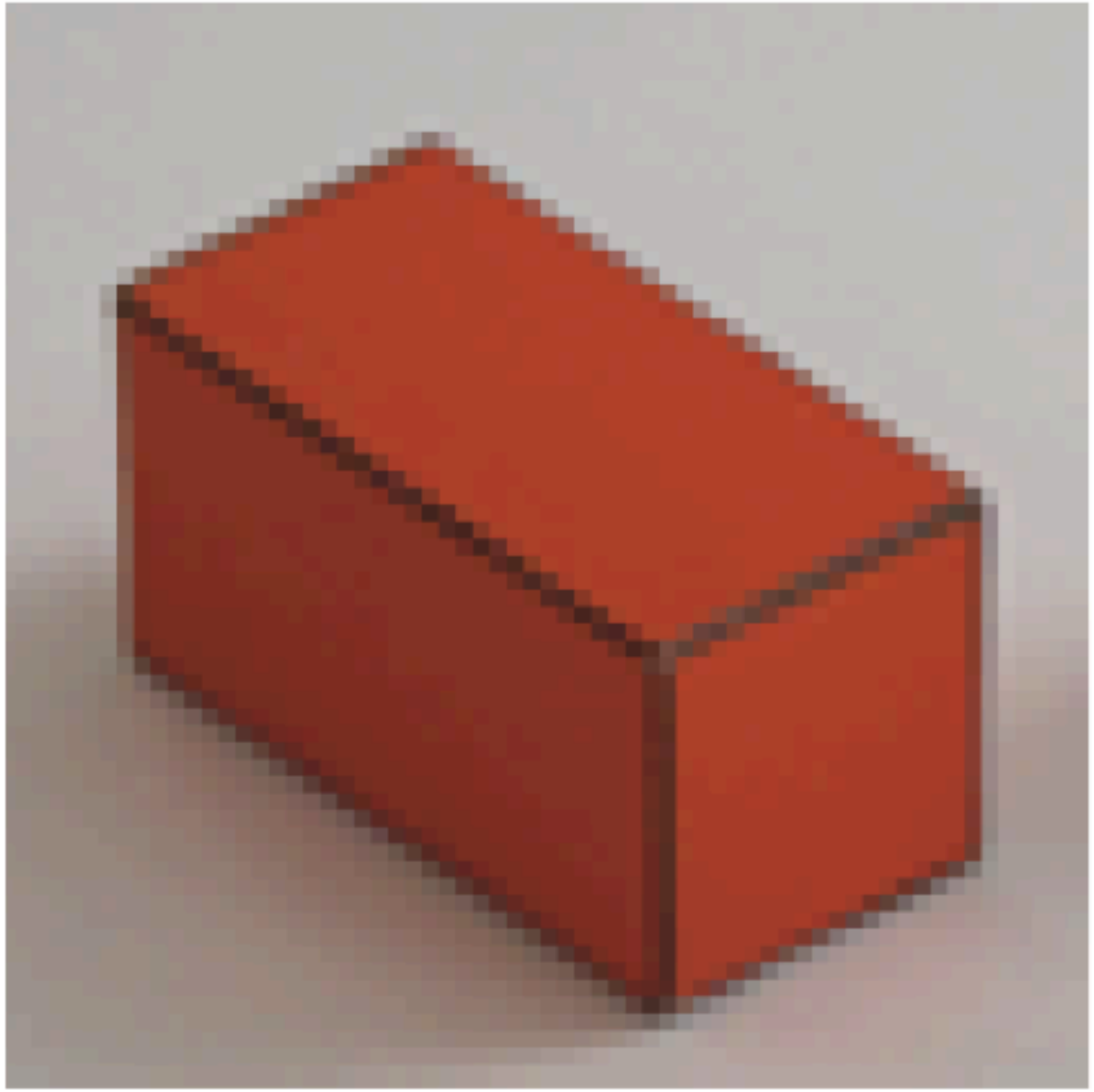


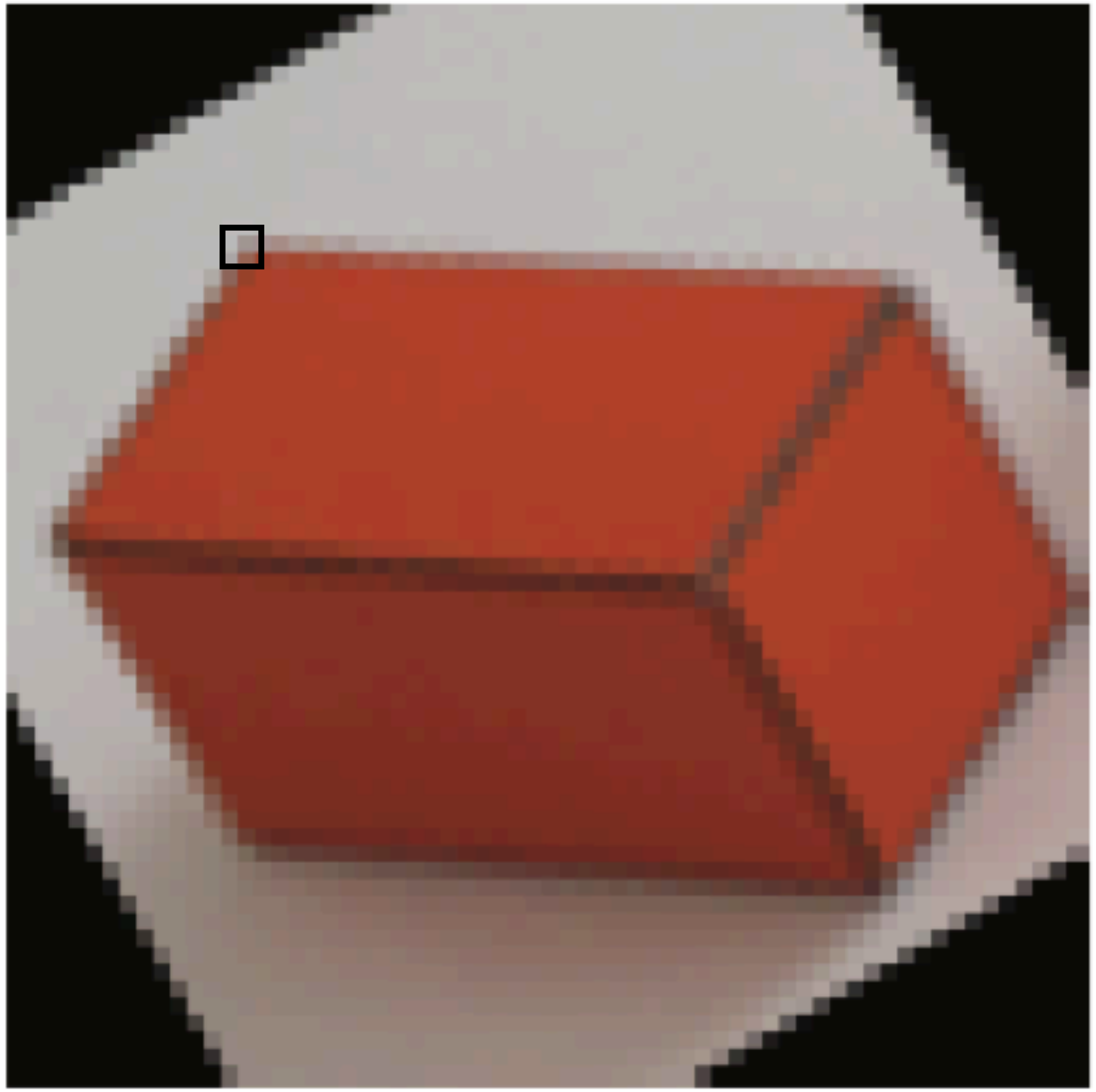
Computes the local average over windows of size 5 x 5 pixels

Examples

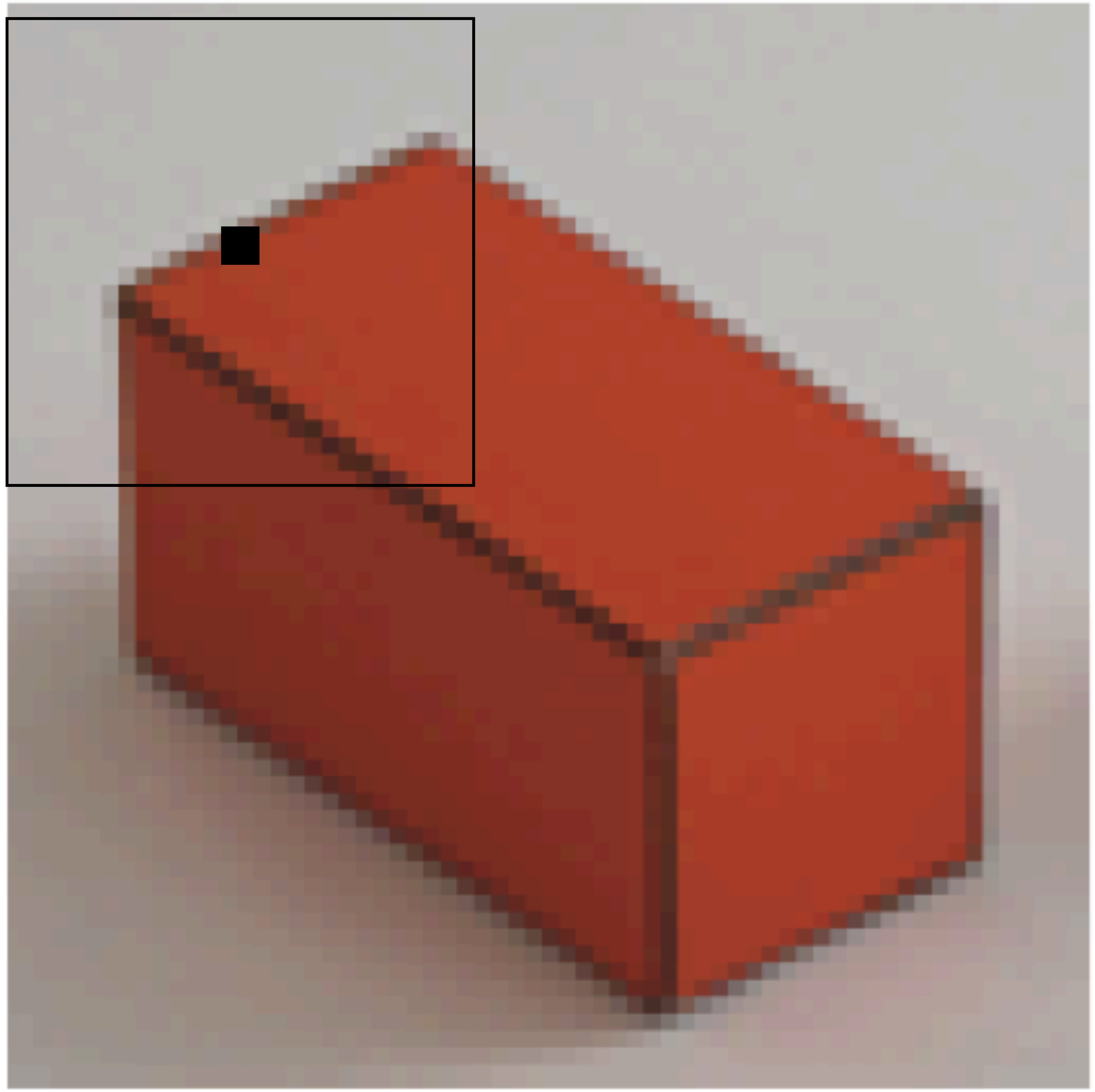


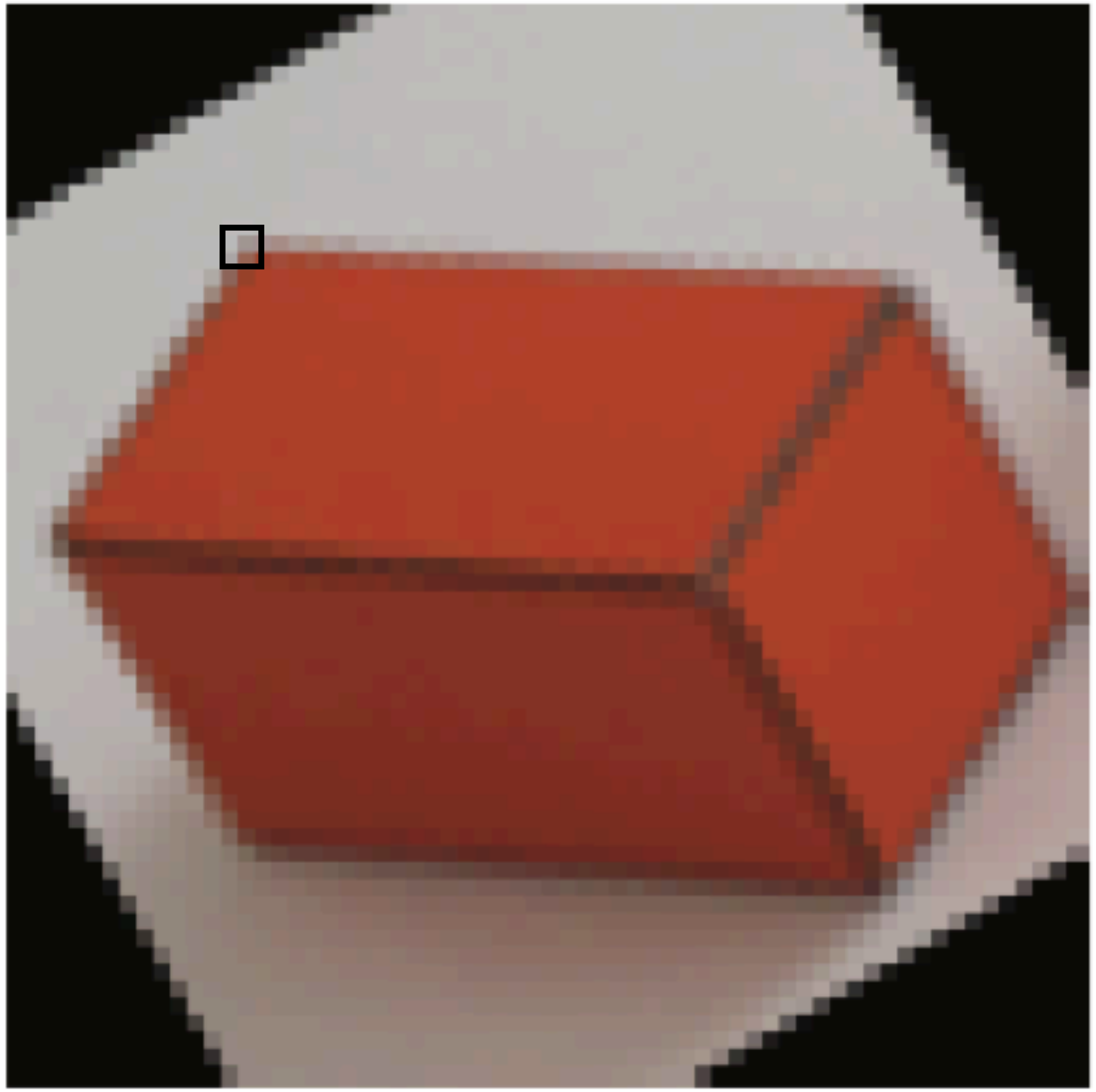
Examples



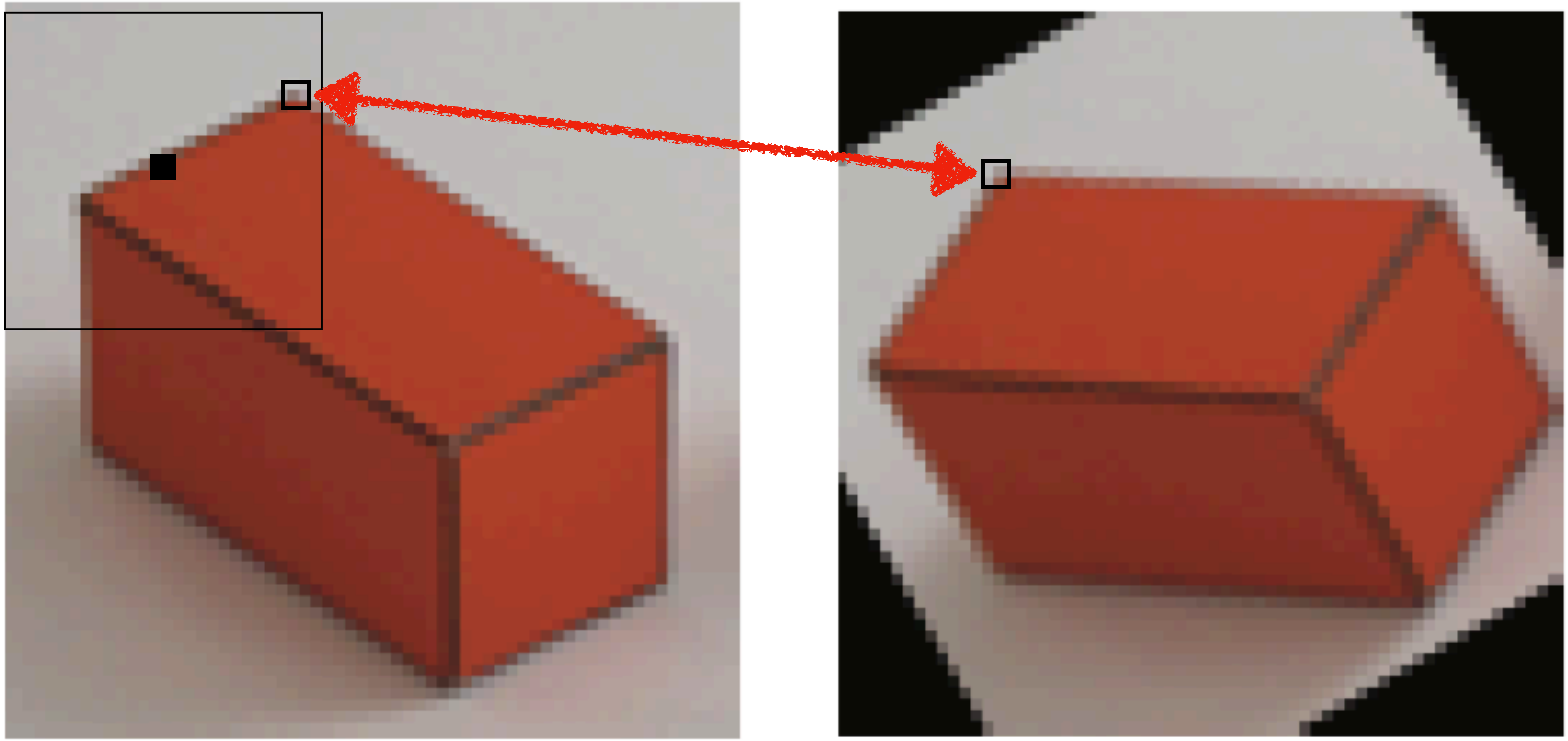


Examples

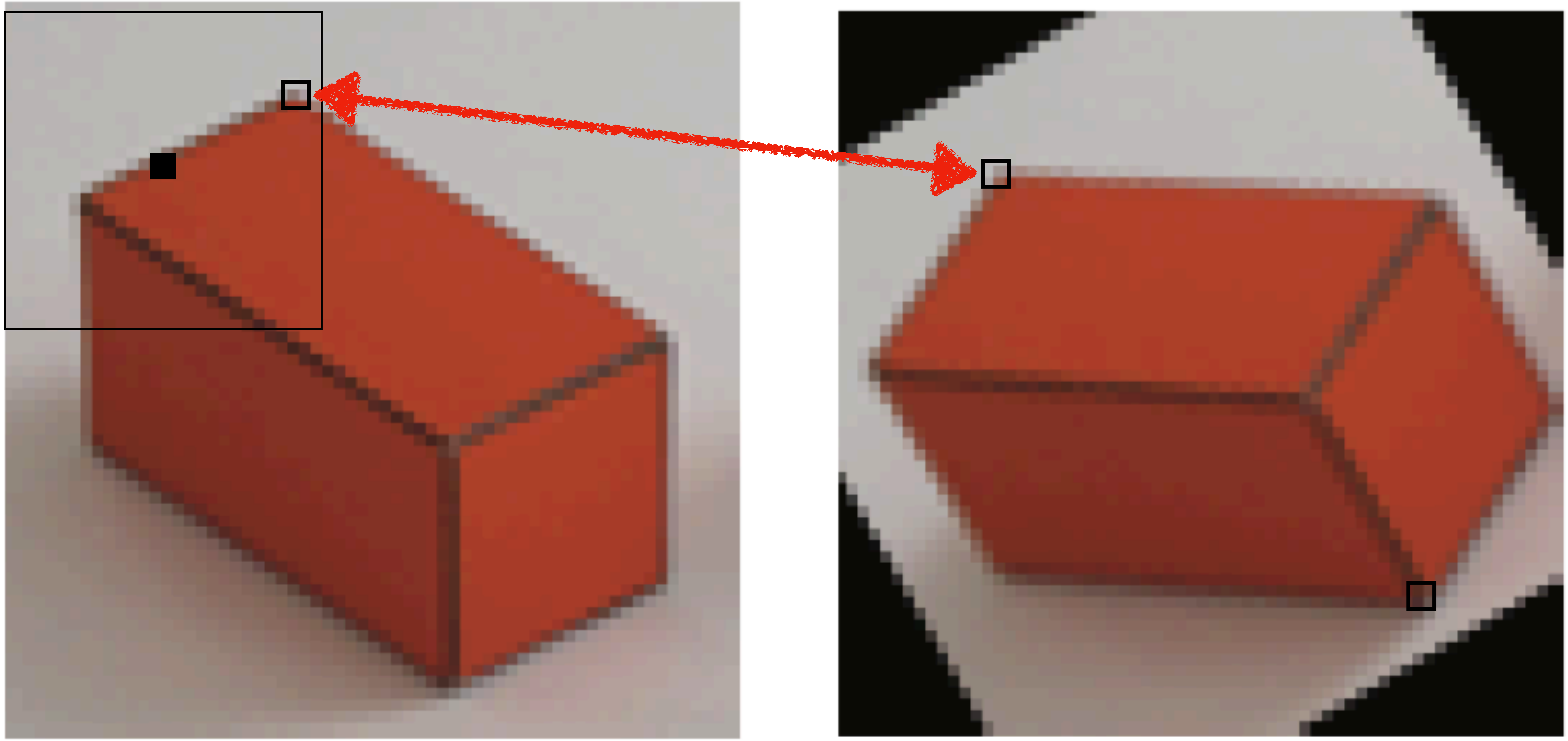




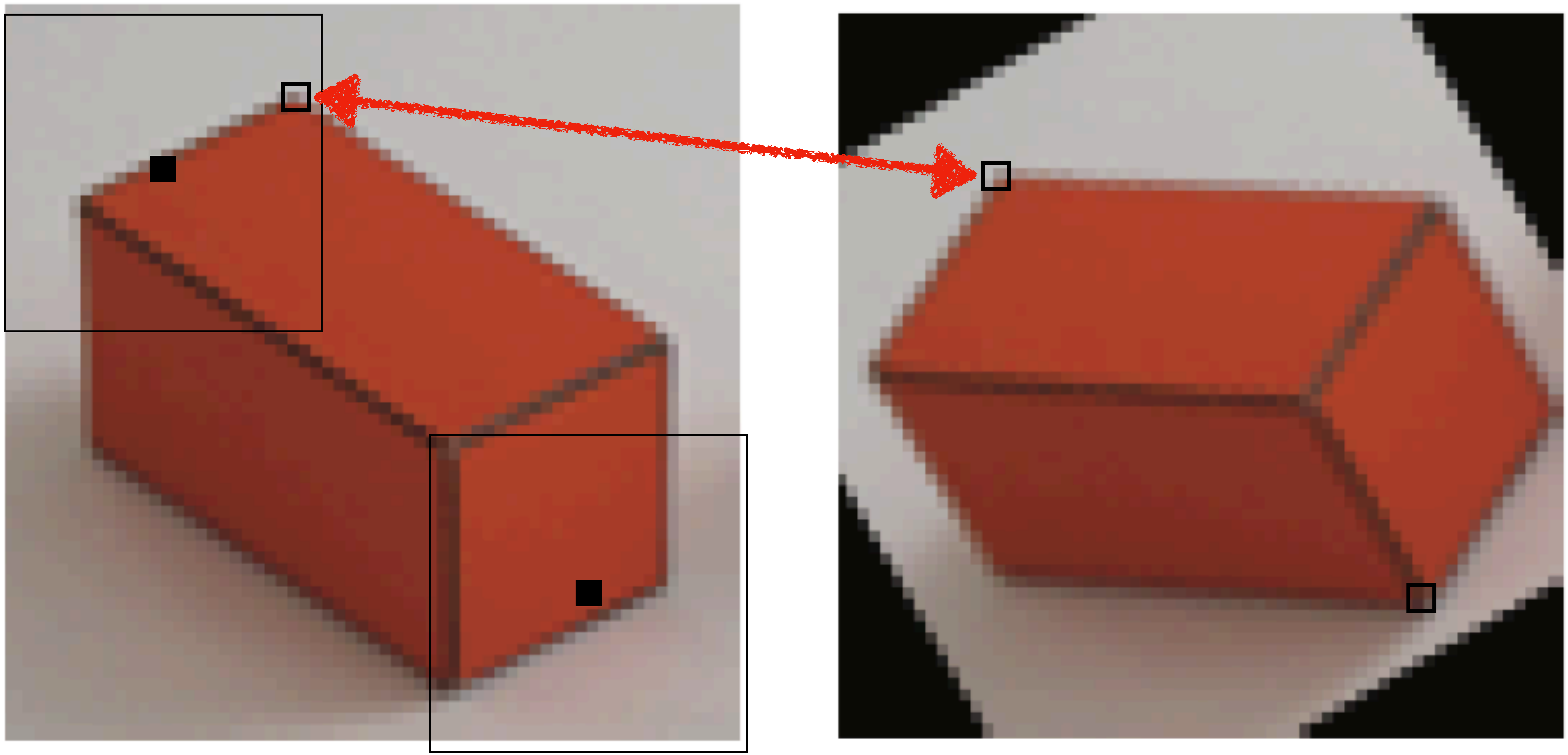
Examples



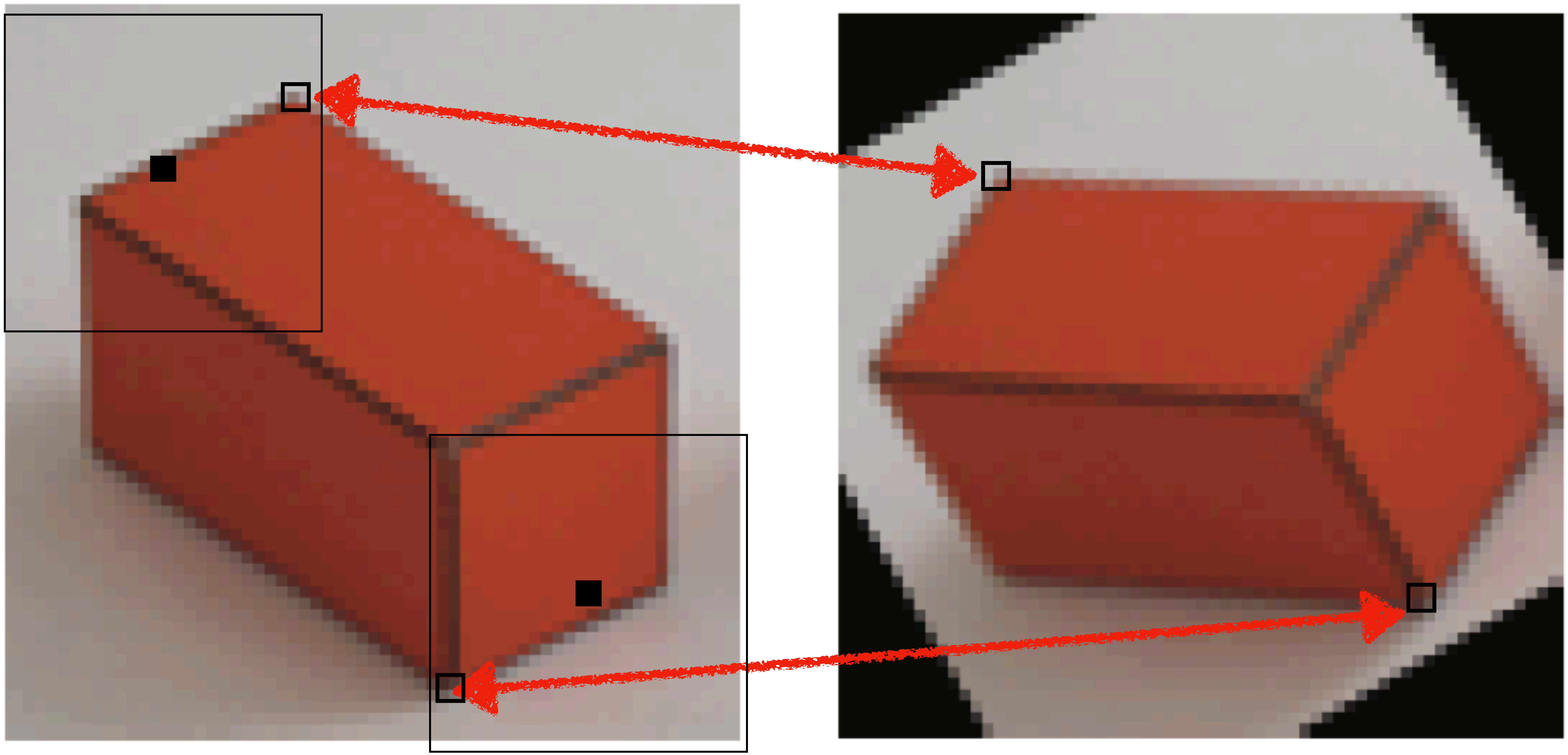
Examples



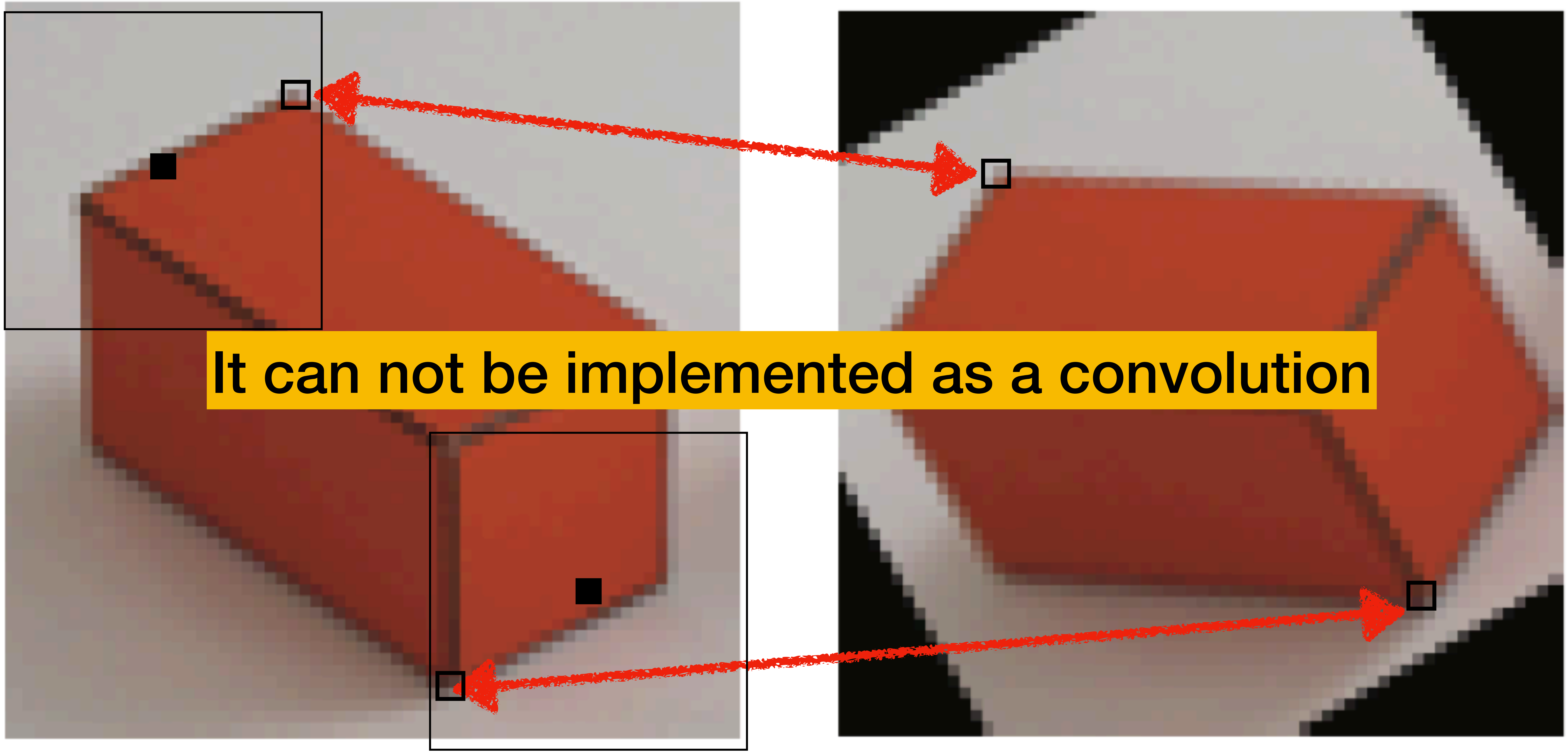
Examples



Examples

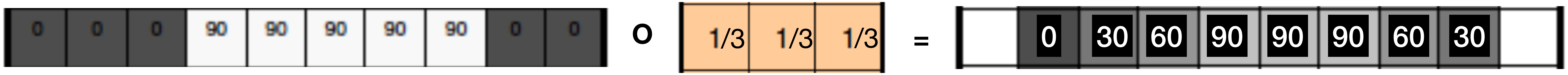


Examples



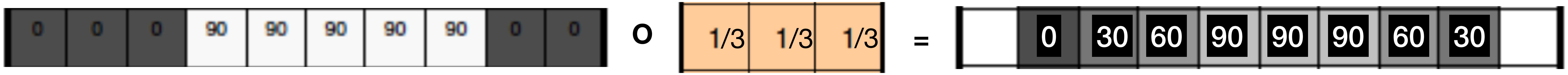
Convolution as matrix multiplication

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

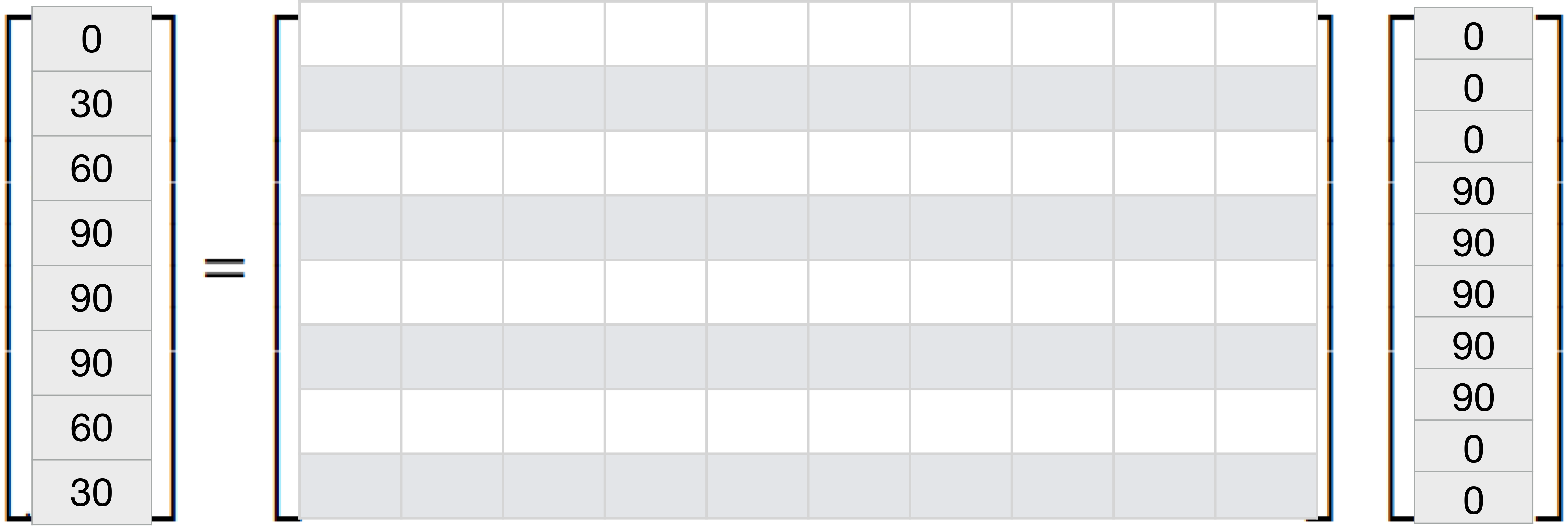


Convolution as matrix multiplication

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

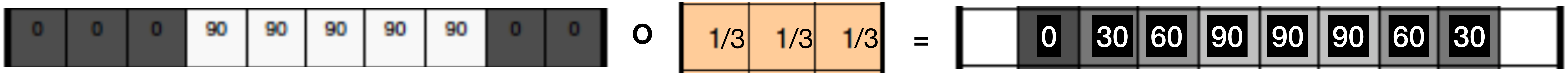


In the 1D case, it helps to make explicit the structure of the matrix:

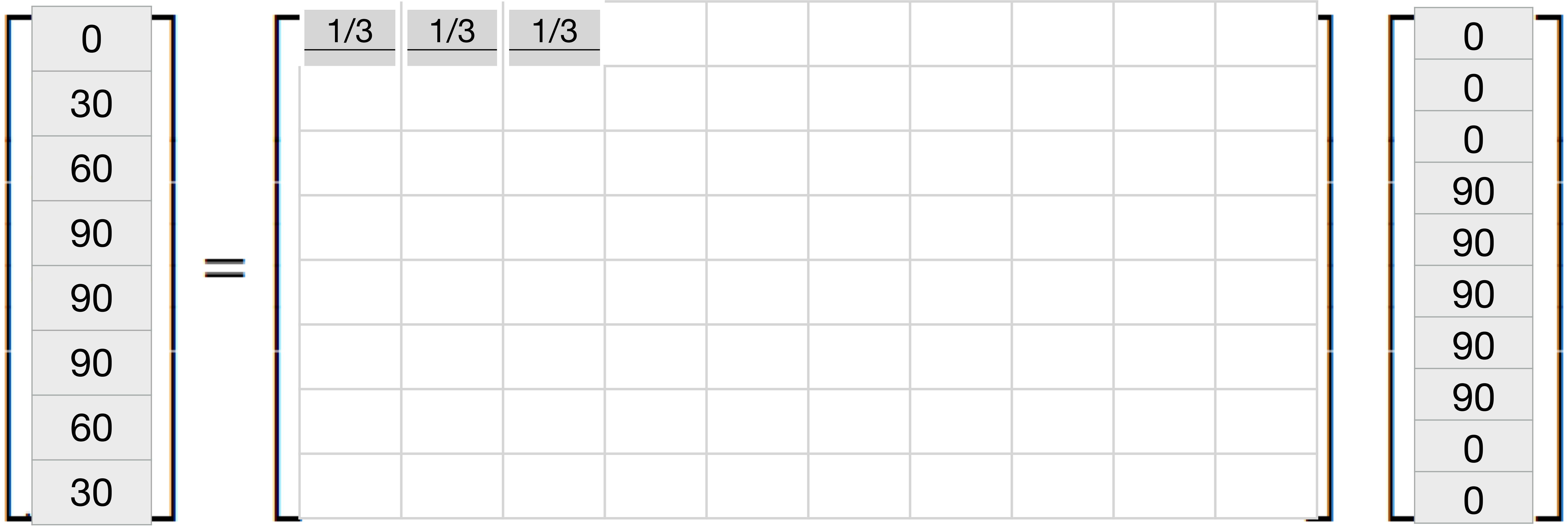


Convolution as matrix multiplication

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

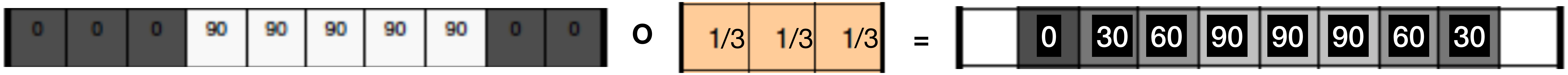


In the 1D case, it helps to make explicit the structure of the matrix:



Convolution as matrix multiplication

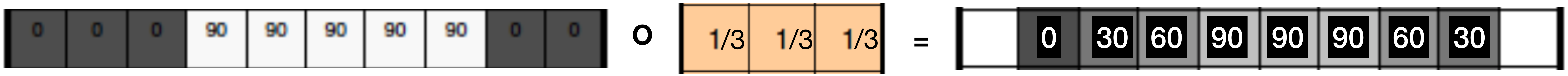
In the 1D case, it helps to make explicit the structure of the matrix:



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

Convolution as matrix multiplication

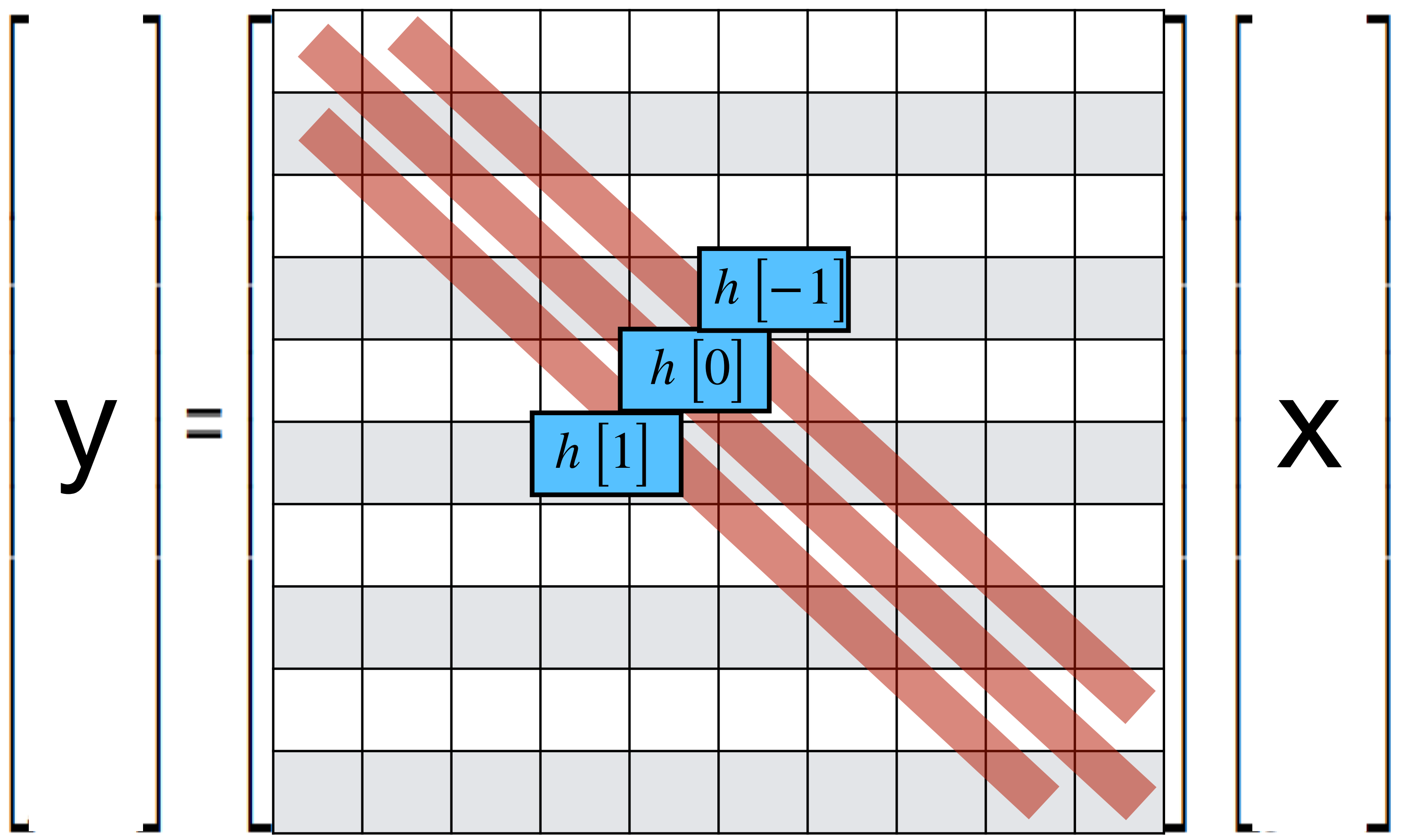
In the 1D case, it helps to make explicit the structure of the matrix:



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

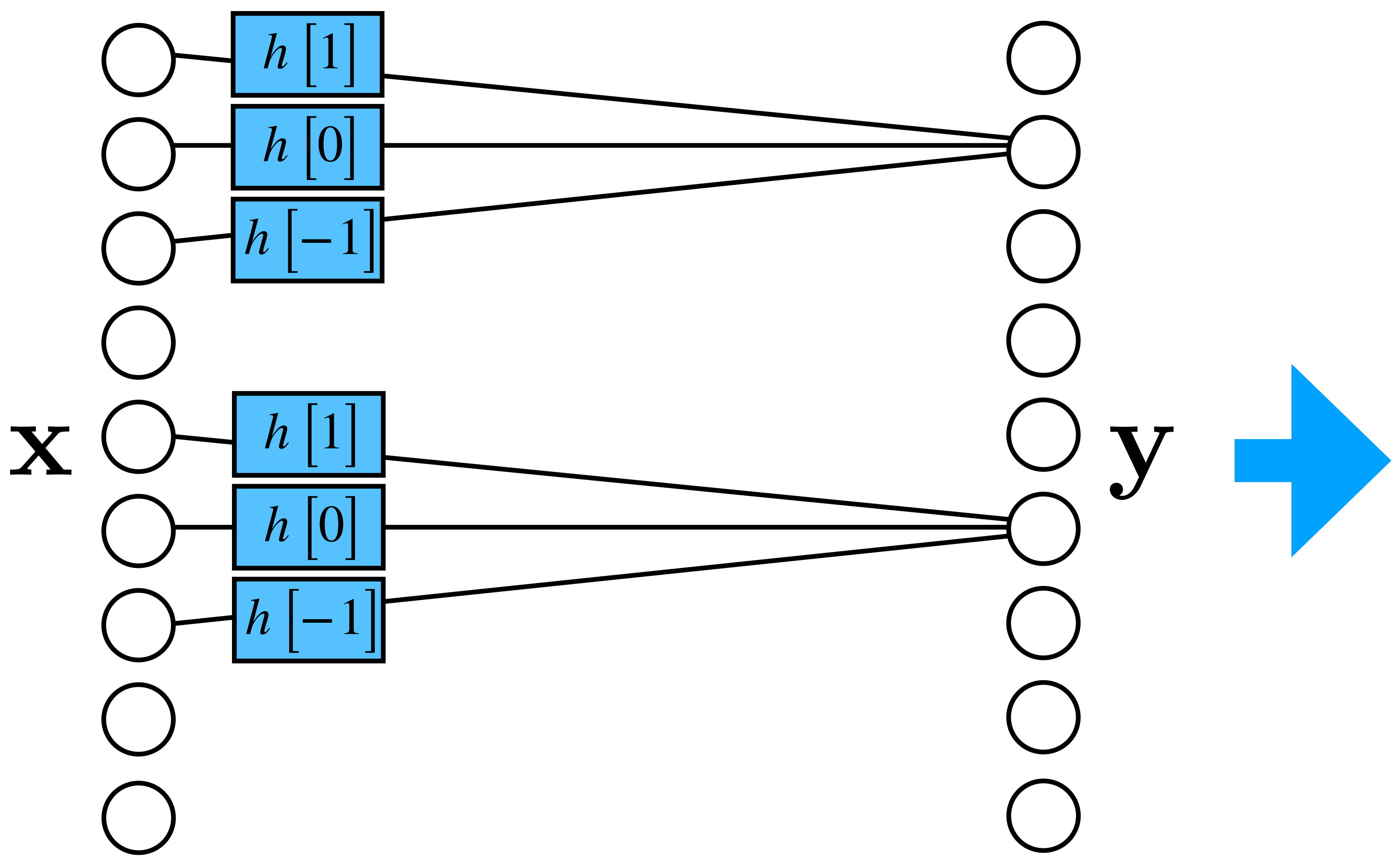
Linear translation invariant system:

A LTI function f can be written as a matrix multiplication:



$h\left[n-k\right]$ n indexes rows, k indexes columns

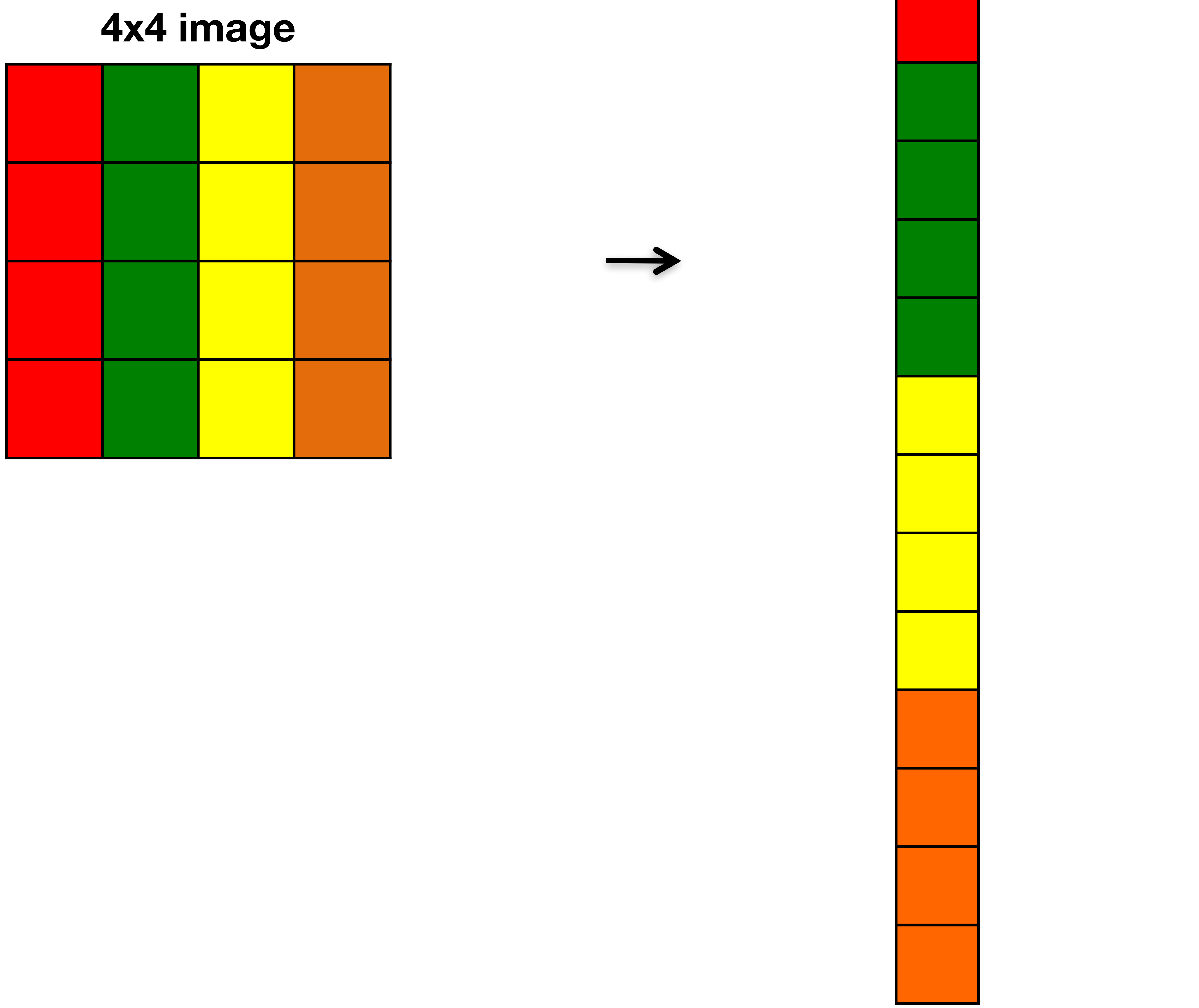
It can also be represented as a convolutional layer of neural net:

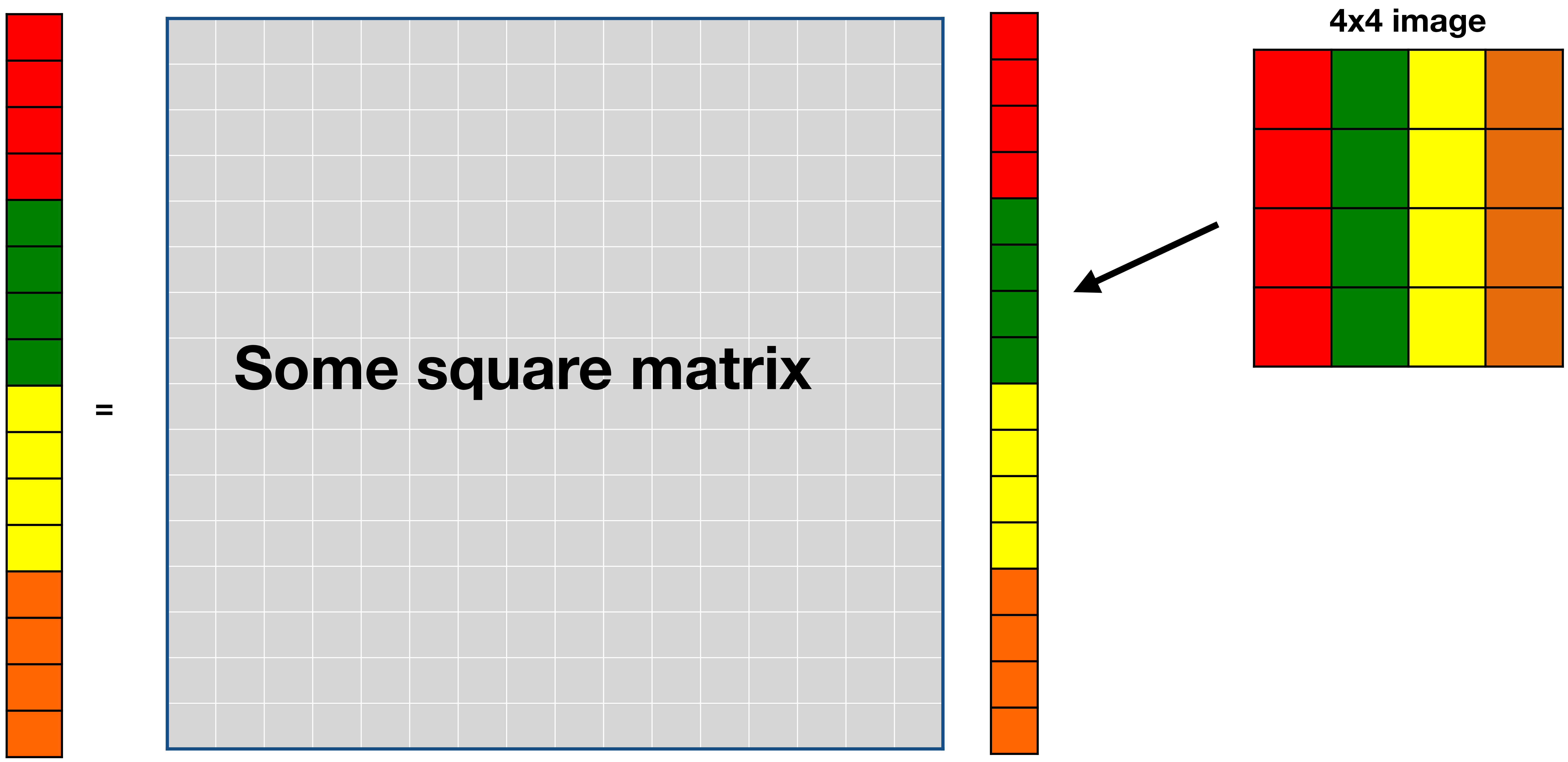


$$y\left[n\right] = \sum_{k=-1}^{1} h\left[k\right] x\left[n-k\right]$$

$h\left[n-k\right]$ Is the strength of the connection between x[k] and y[n]

Images are turned into column vectors by concatenating all image columns




=
Some square matrix
4x4 image

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]

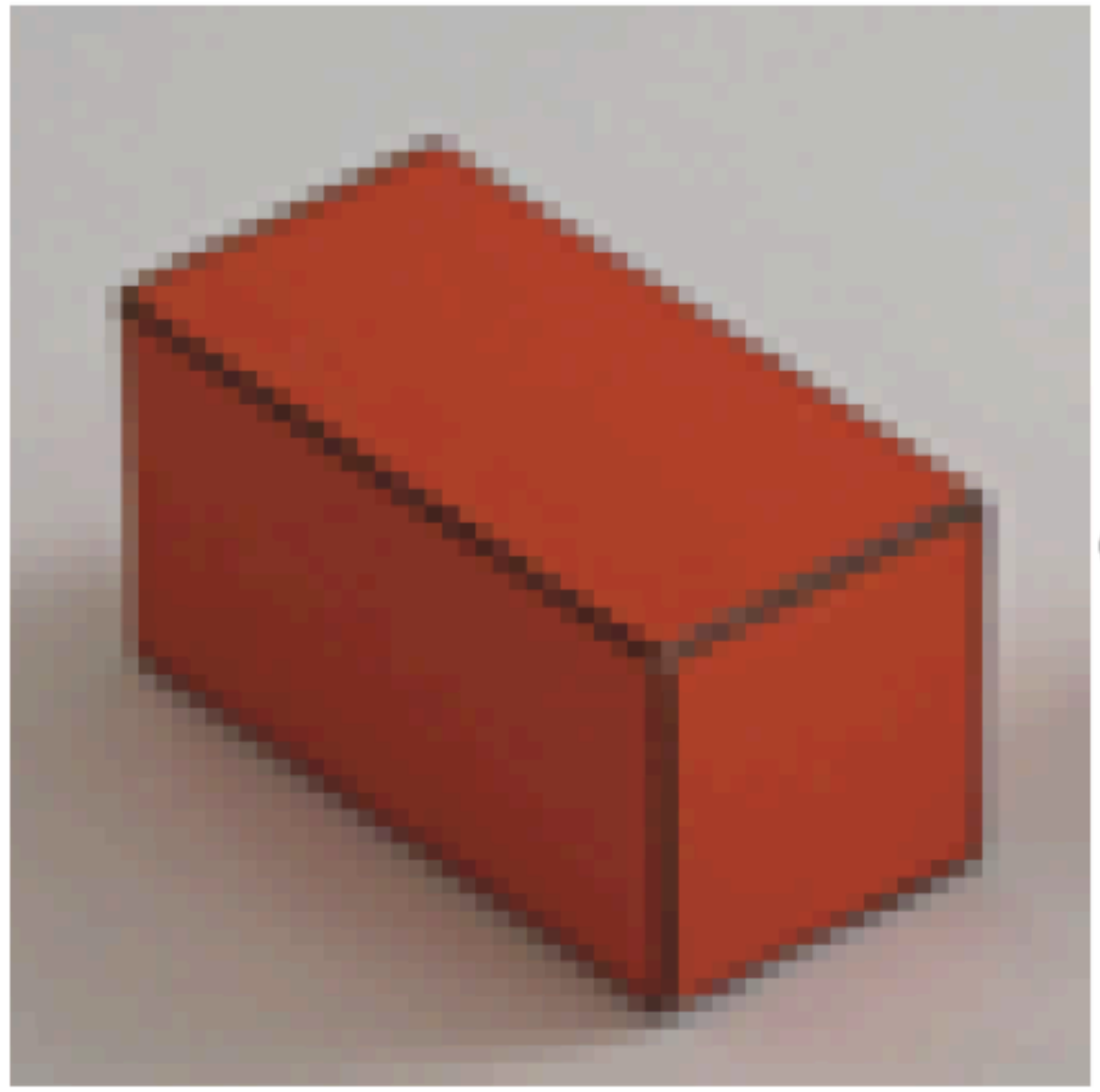
$\otimes$

h[m,n]

=

f[m,n]

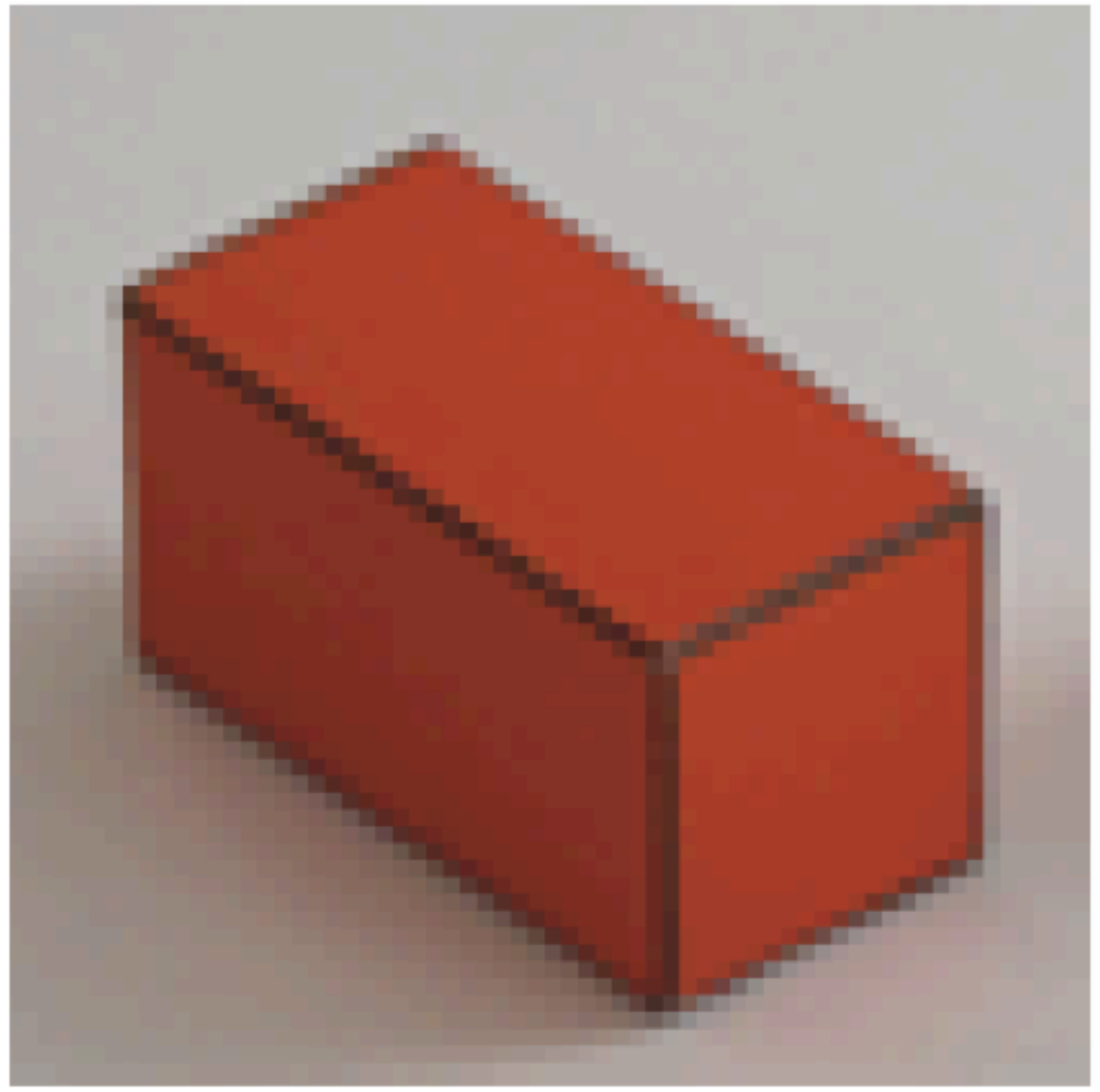
The identity



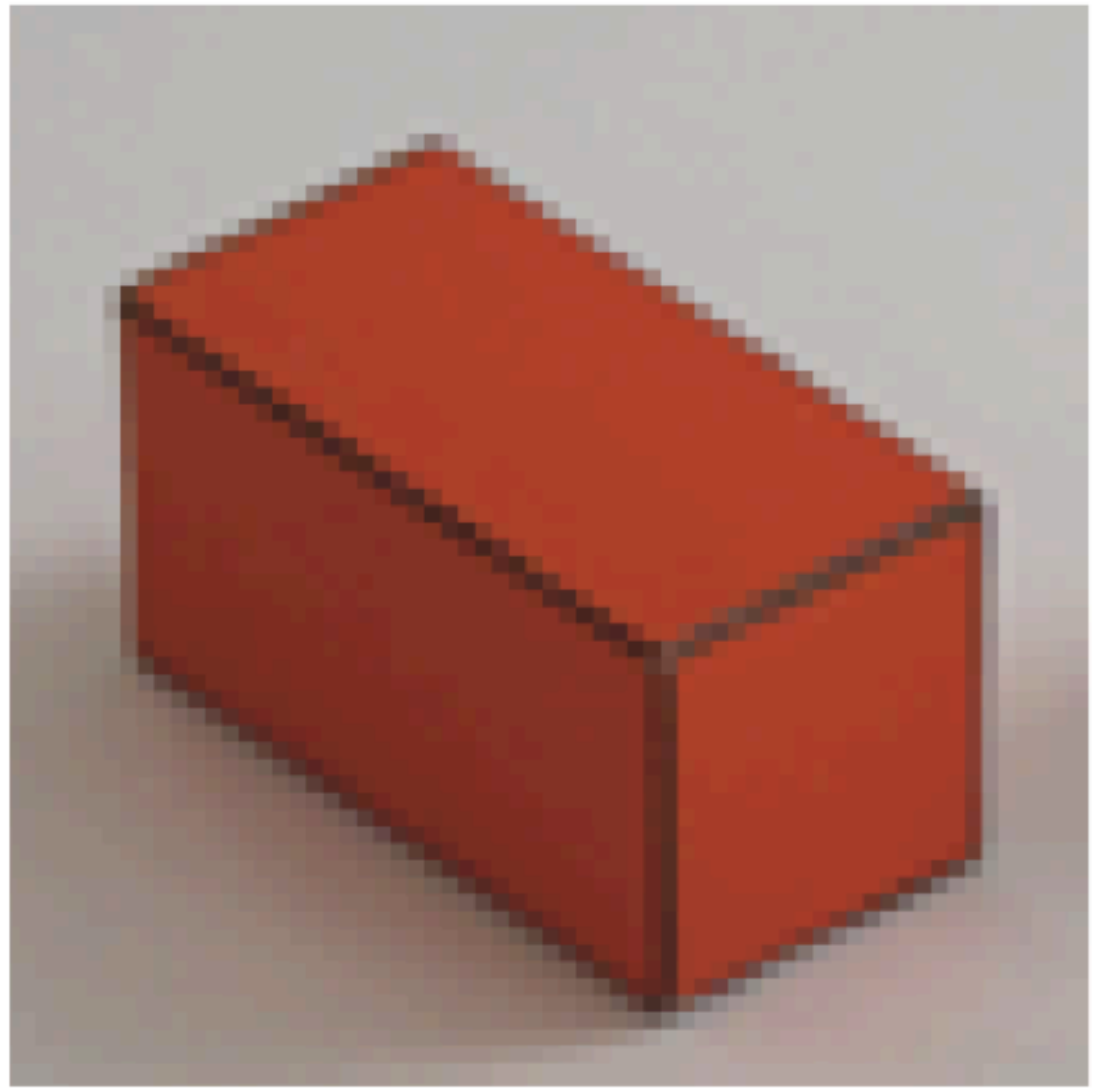
$\circ$

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

$=$



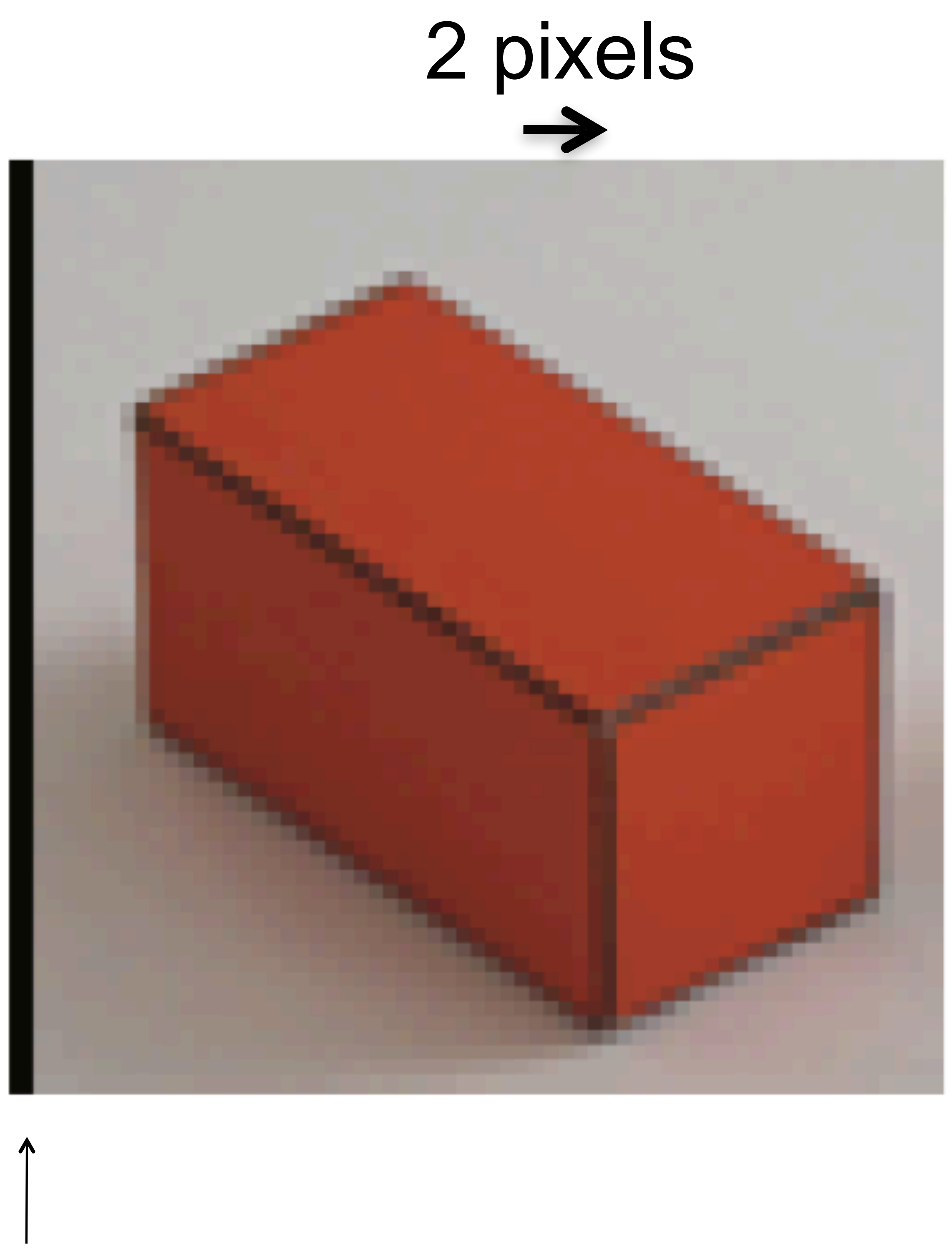
A shift



○

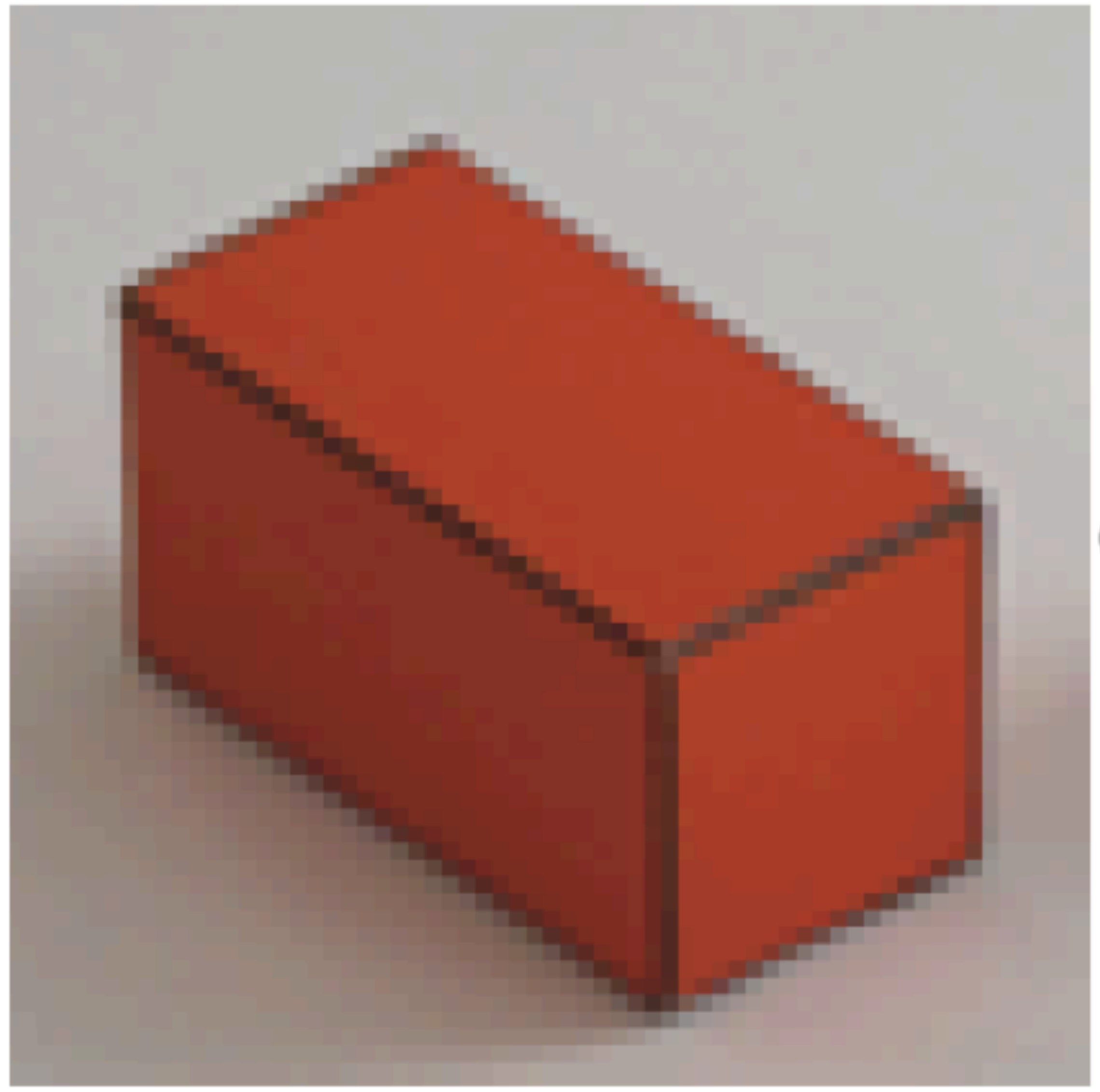
0	0	0	0	0
0	0	0	0	0
0	0	0	0	1
0	0	0	0	0
0	0	0	0	0

=



(using zero padding)

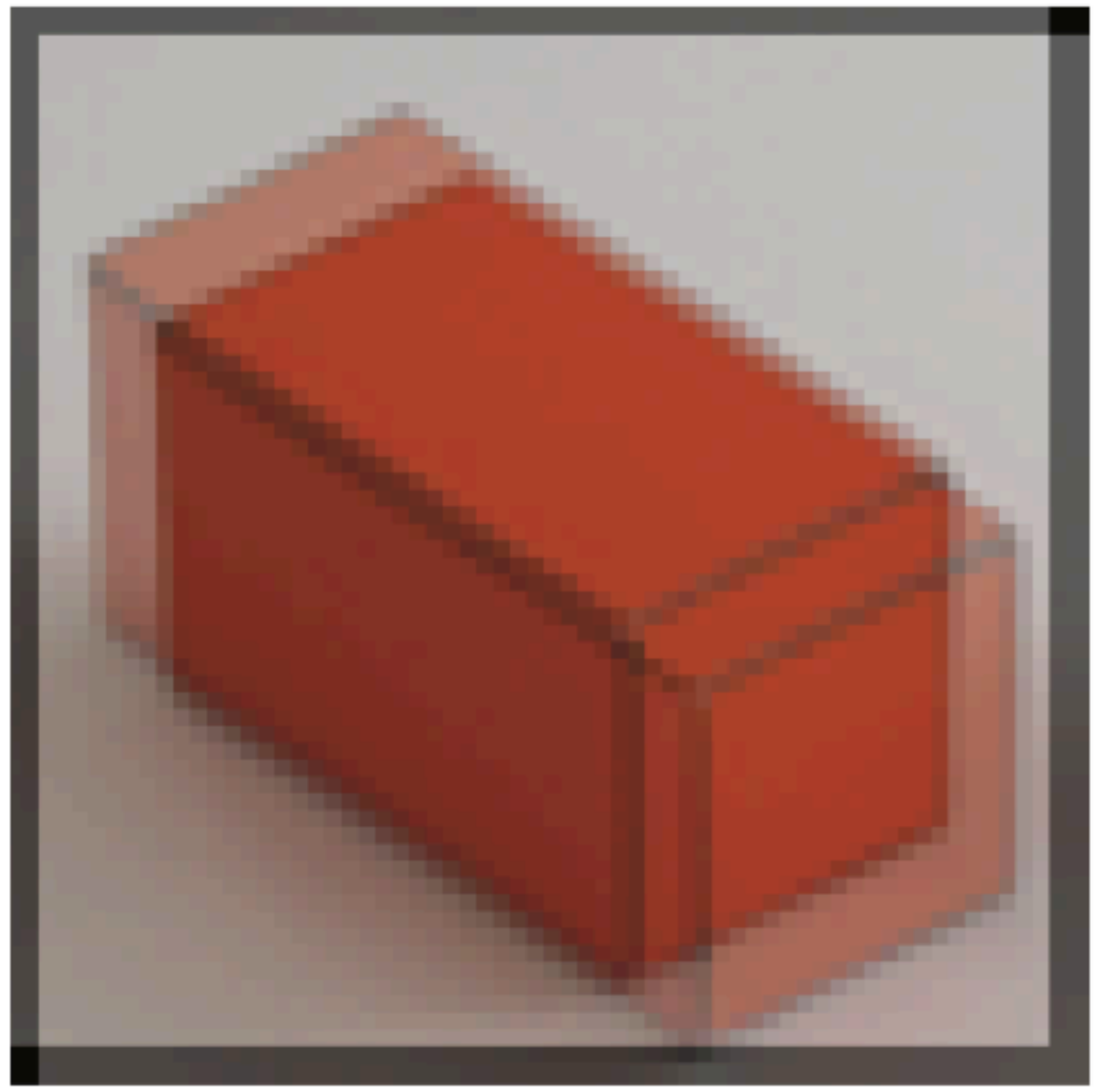
Examples



○

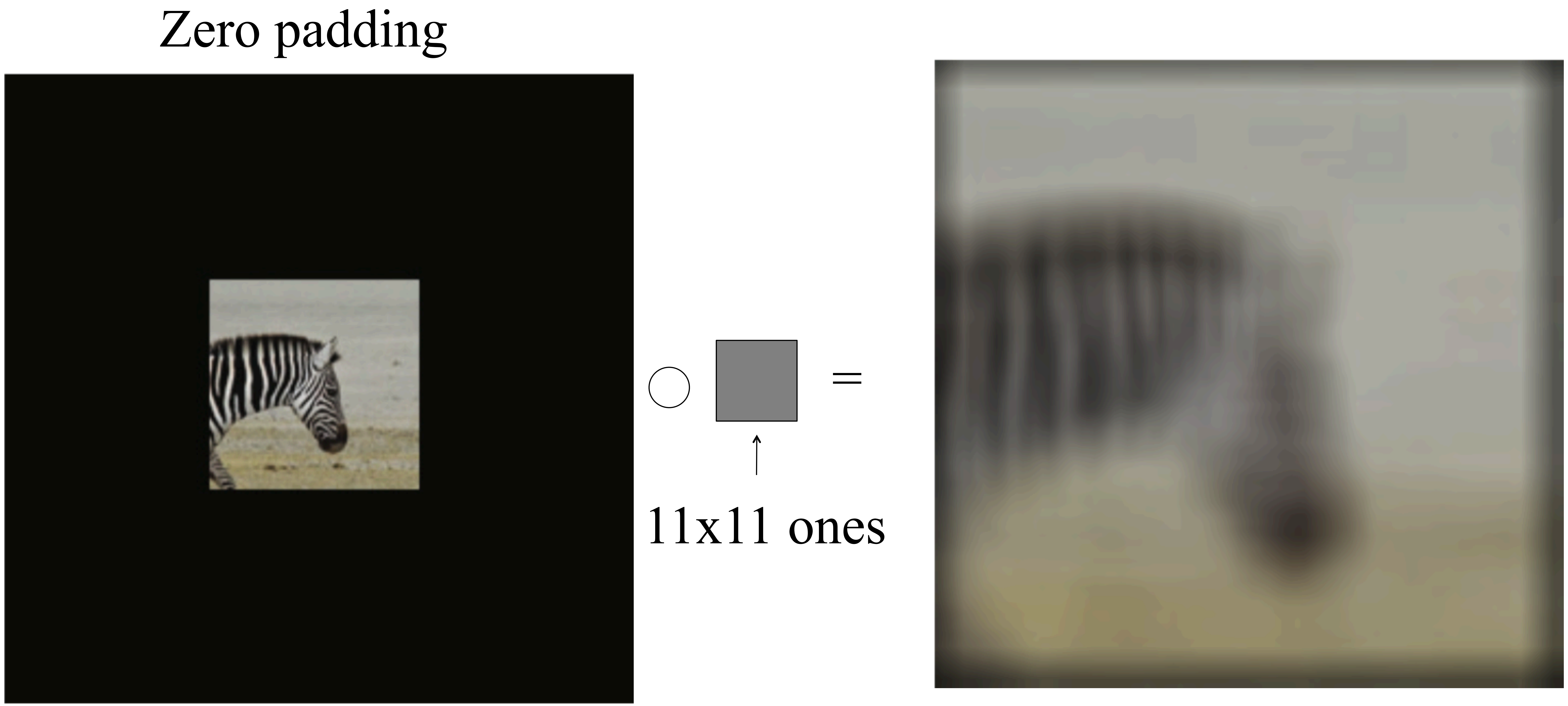
.5	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	.5

=



Handling boundaries

Handling boundaries



Handling boundaries

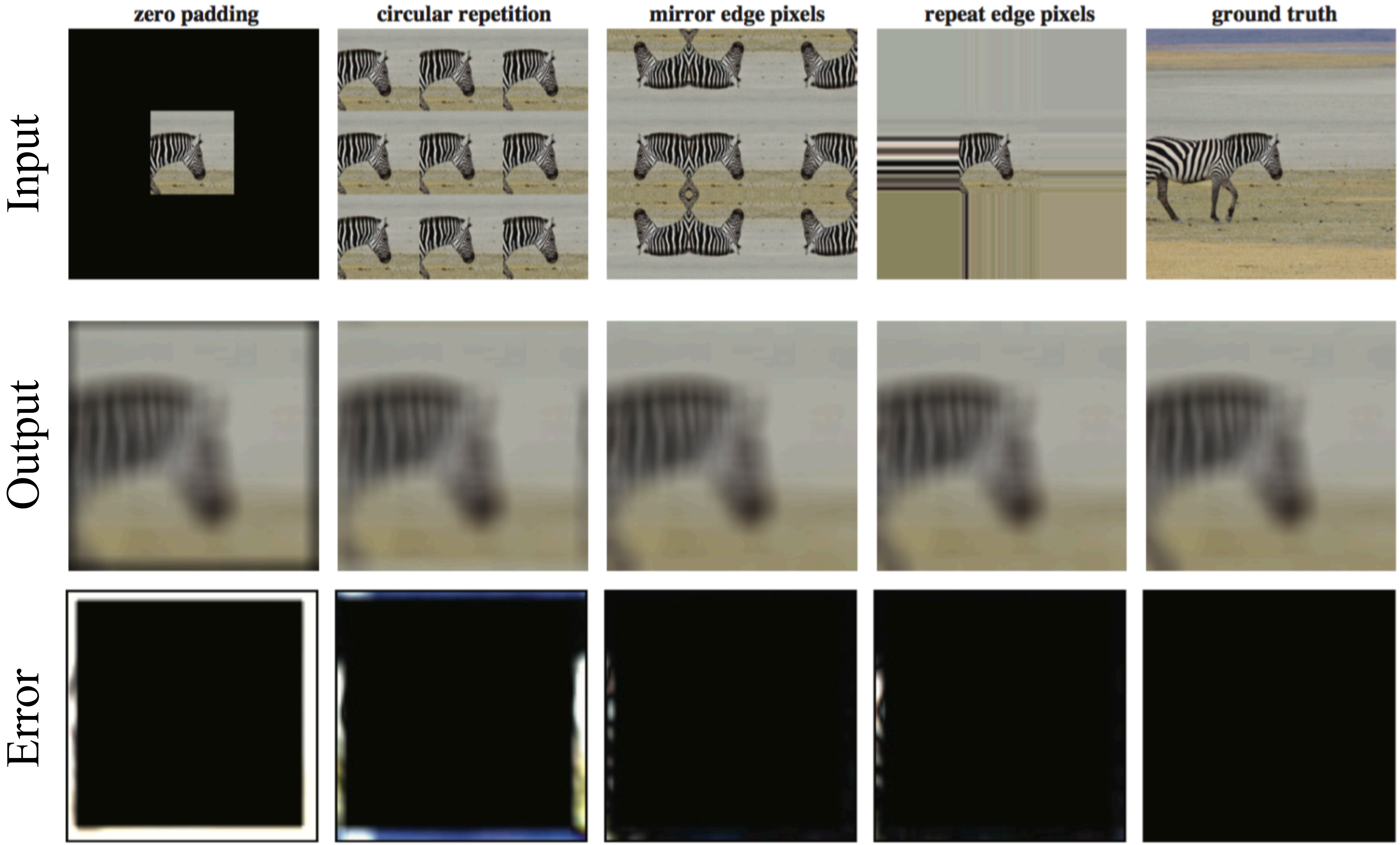
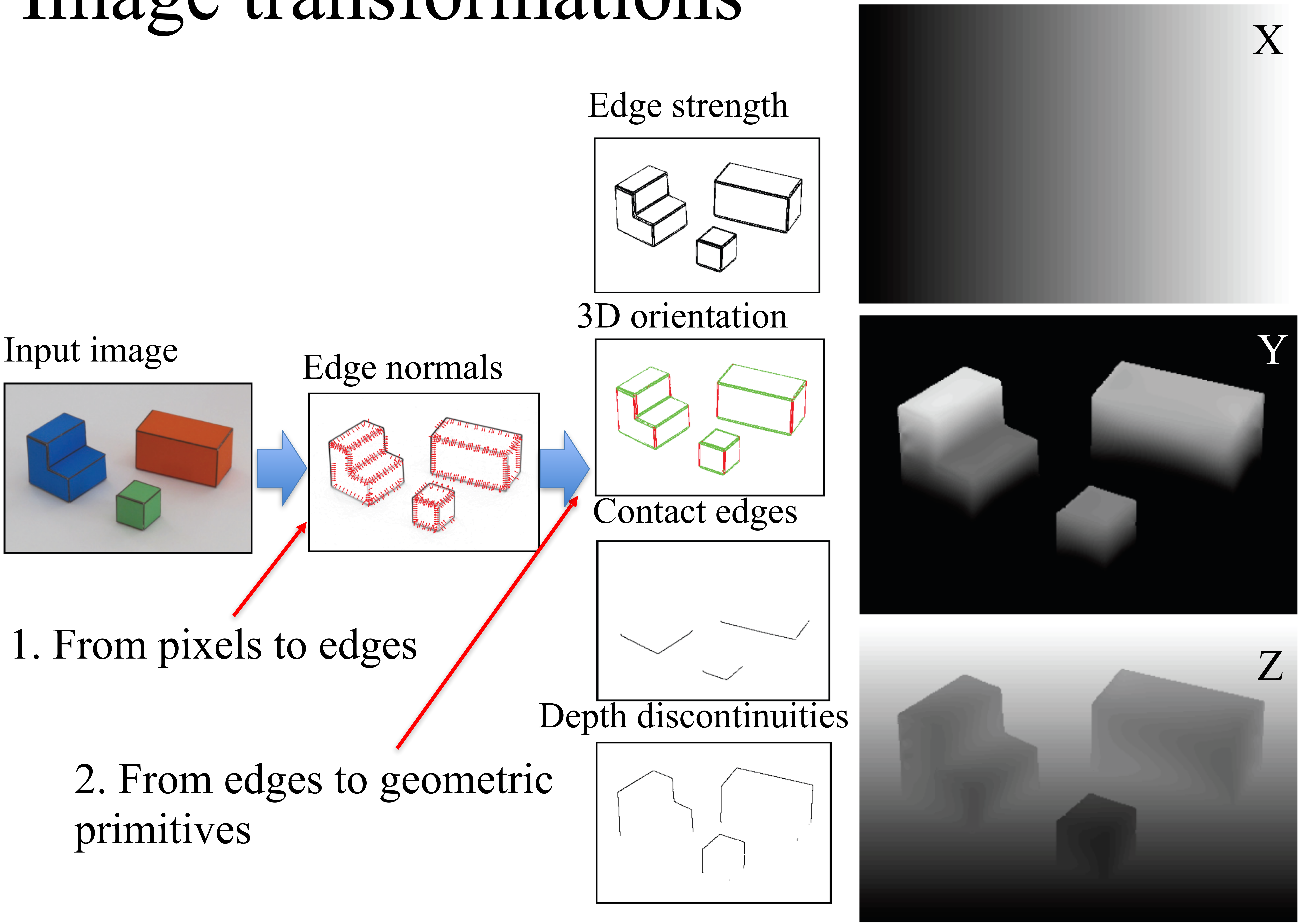


Image transformations



Linear image transformations

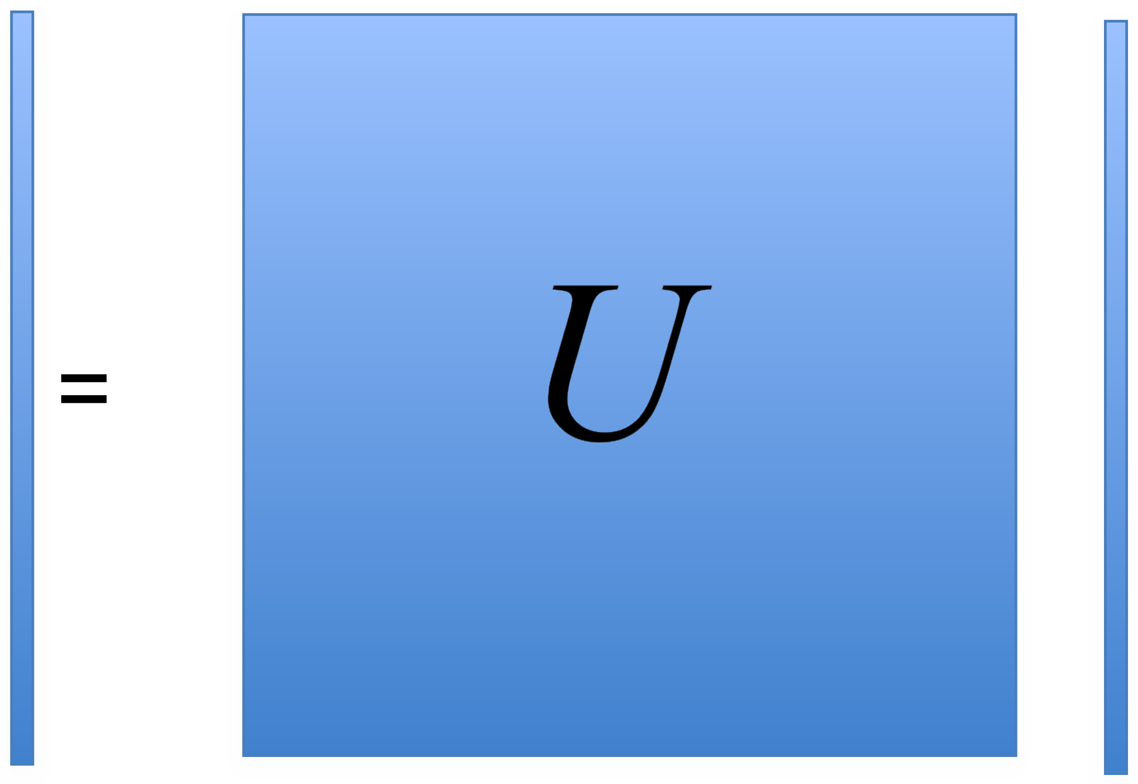
In analyzing images, it's often useful to make a change of basis.

Transformed image

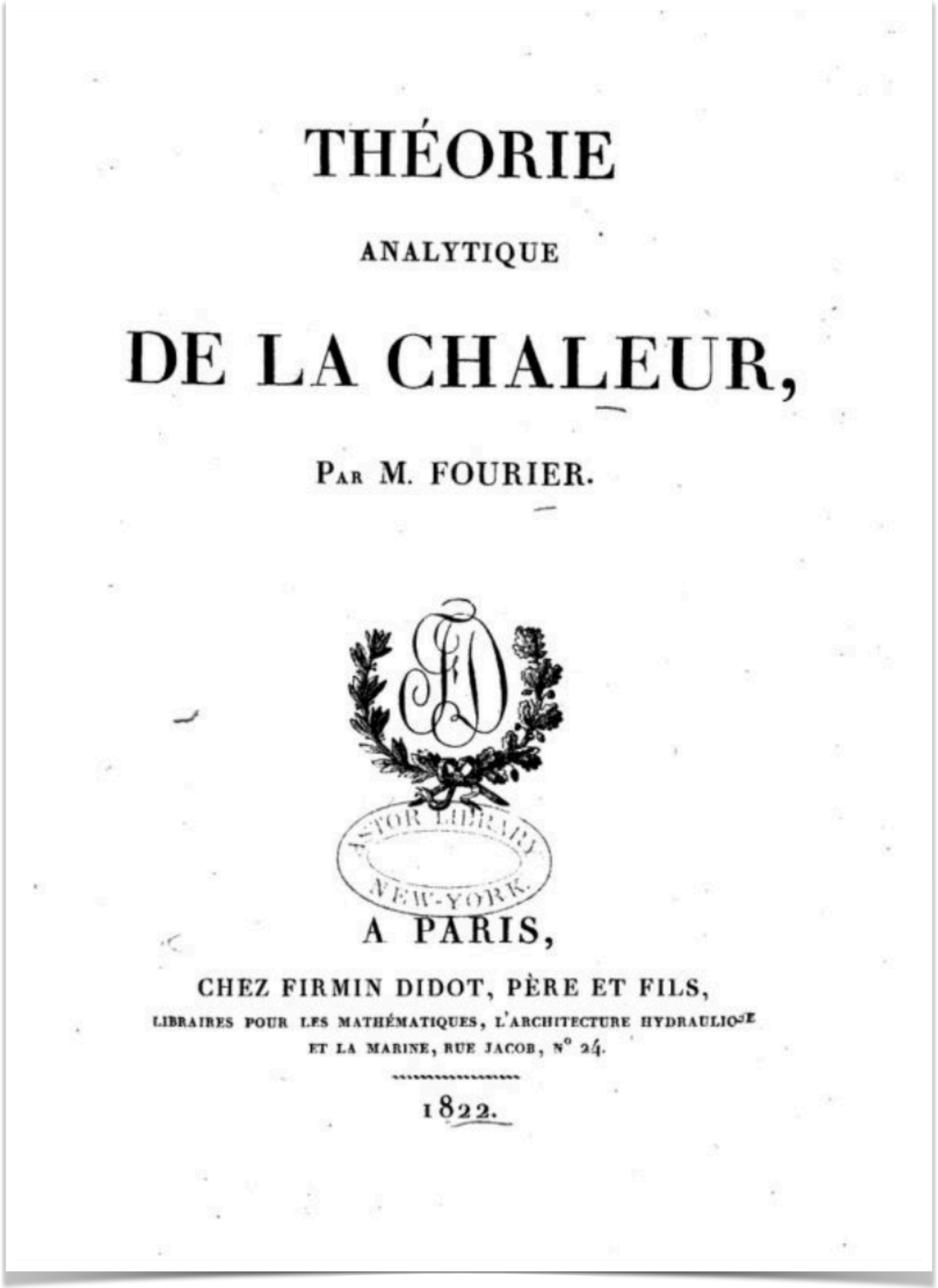
$$\vec{F} = U\vec{f}$$

Vectorized image

Fourier transform, or
Wavelet transform, or
Steerable pyramid transform



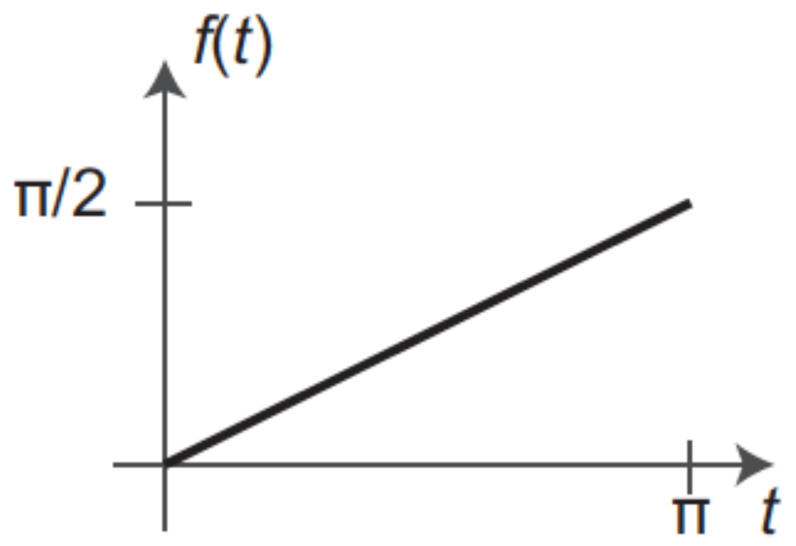
Fourier series



$$f(t) = a_1 \sin(t) + a_2 \sin(2t) + a_3 \sin(3t) + \ldots$$

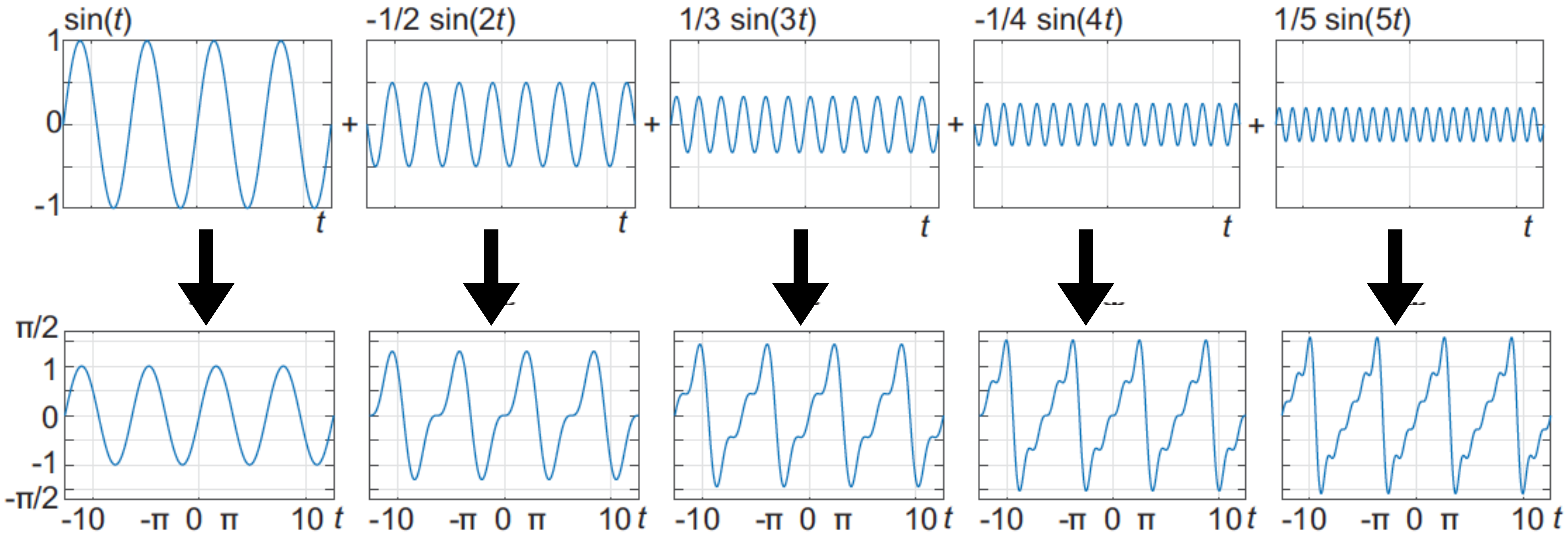
With $a_n = \frac{2}{\pi} \int_0^{\pi} f(t) \sin(nt) dt$

One of Fourier's original examples of sine series is the expansion of the ramp signal:



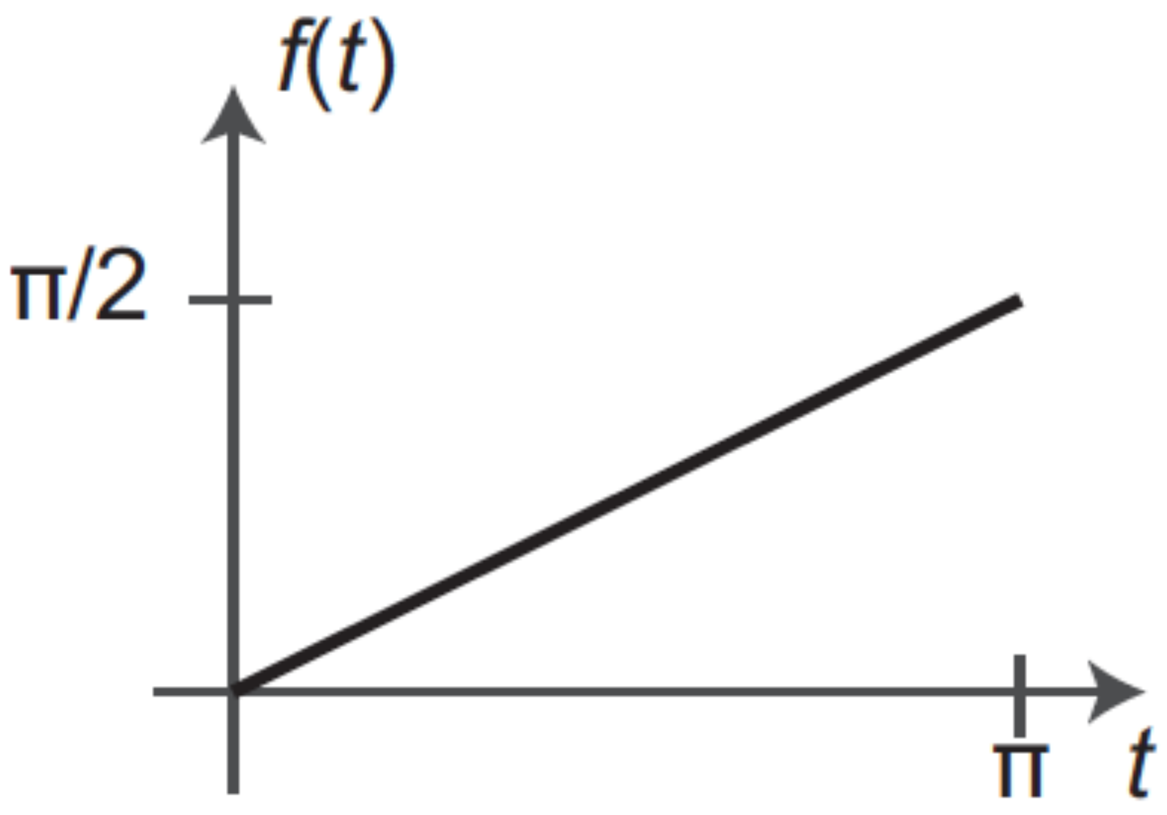
$$\frac{1}{2}t = \sin(t) - \frac{1}{2}\sin(2t) + \frac{1}{3}\sin(3t) - \frac{1}{4}\sin(4t) + \ldots$$

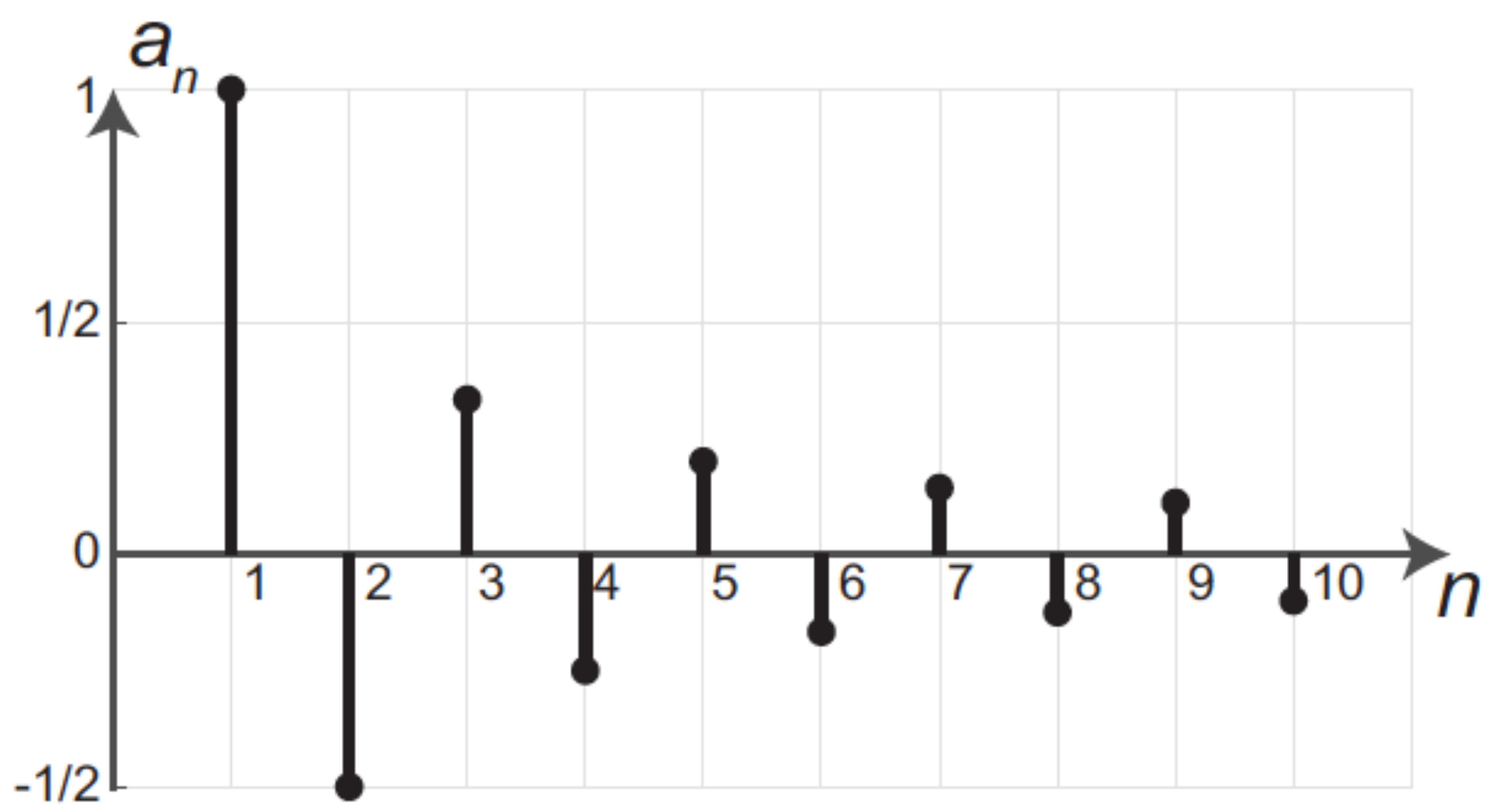
The result of this series approximates the ramp with increasing accuracy as we add more terms.



Fourier series as change of representation

$$\frac{1}{2}t = \sin(t) - \frac{1}{2}\sin(2t) + \frac{1}{3}\sin(3t) - \frac{1}{4}\sin(4t) + \ldots$$





It is useful to think of the Fourier series of a signal as a **change of representation**. Instead of representing the signal by the sequence of values specified by the original function *f*(*t*), the same function can be represented by the infinite sequence of coefficients (a_n).

The Discrete Fourier transform

Discrete Fourier Transform (DFT) transforms a signal *f* [*n*] into *F* [*u*] as:

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

The inverse of the DFT is:

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

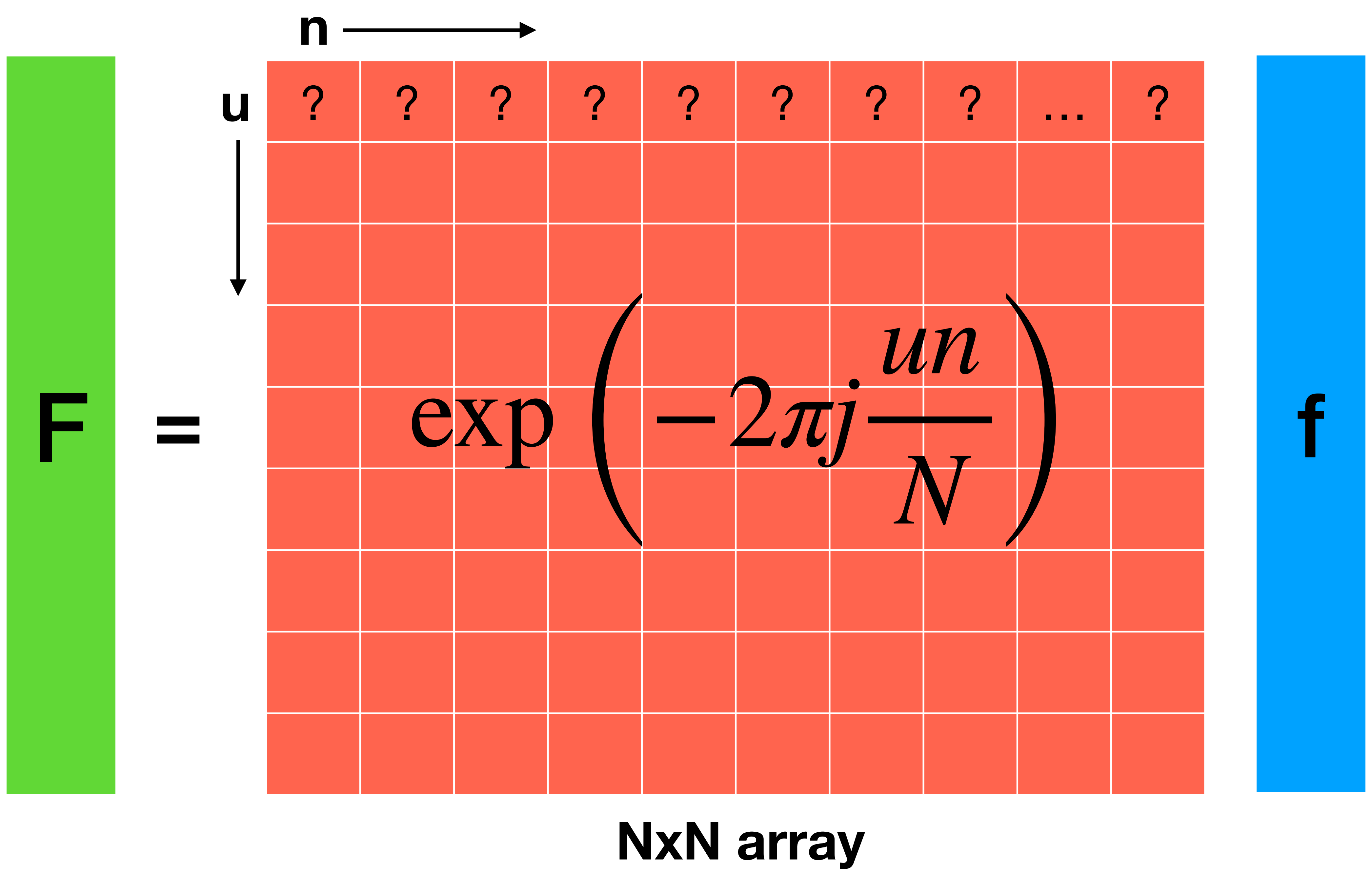
The signal *f* [*n*] is a weighted linear combination of complex exponentials with weights *F* [*u*]

The Discrete Fourier transform

Discrete Fourier Transform (DFT) transforms a signal *f* [*n*] into *F* [*u*] as:

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

Discrete Fourier Transform (DFT) is a linear operator. Therefore, we can write:



Lets visualize the transform coefficients

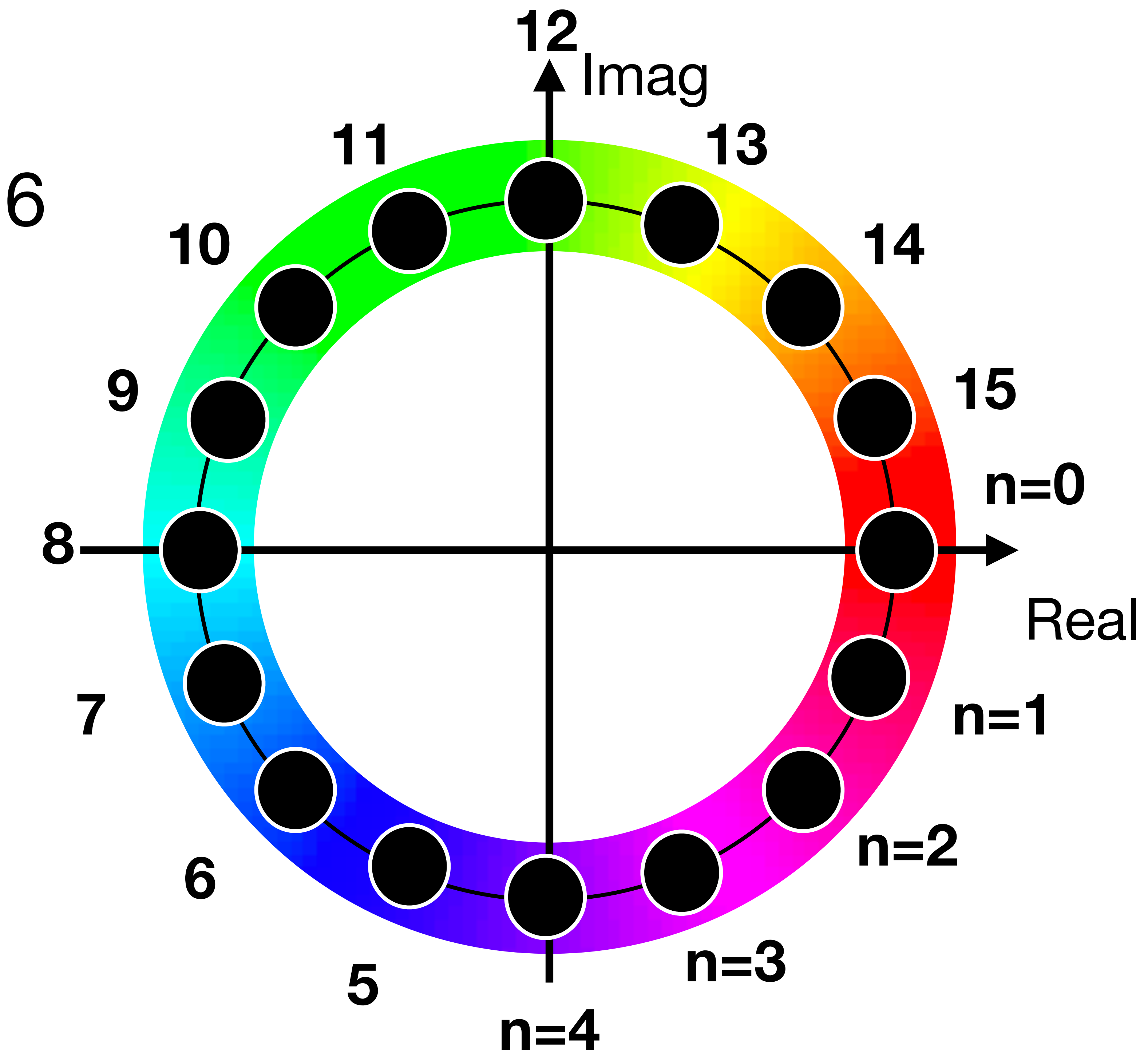
Visualizing the Fourier transform

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

$$\exp(\alpha j) = \cos(\alpha) + j \sin(\alpha)$$

$$\cos\left(2\pi\frac{un}{N}\right) - j \sin\left(2\pi\frac{un}{N}\right)$$

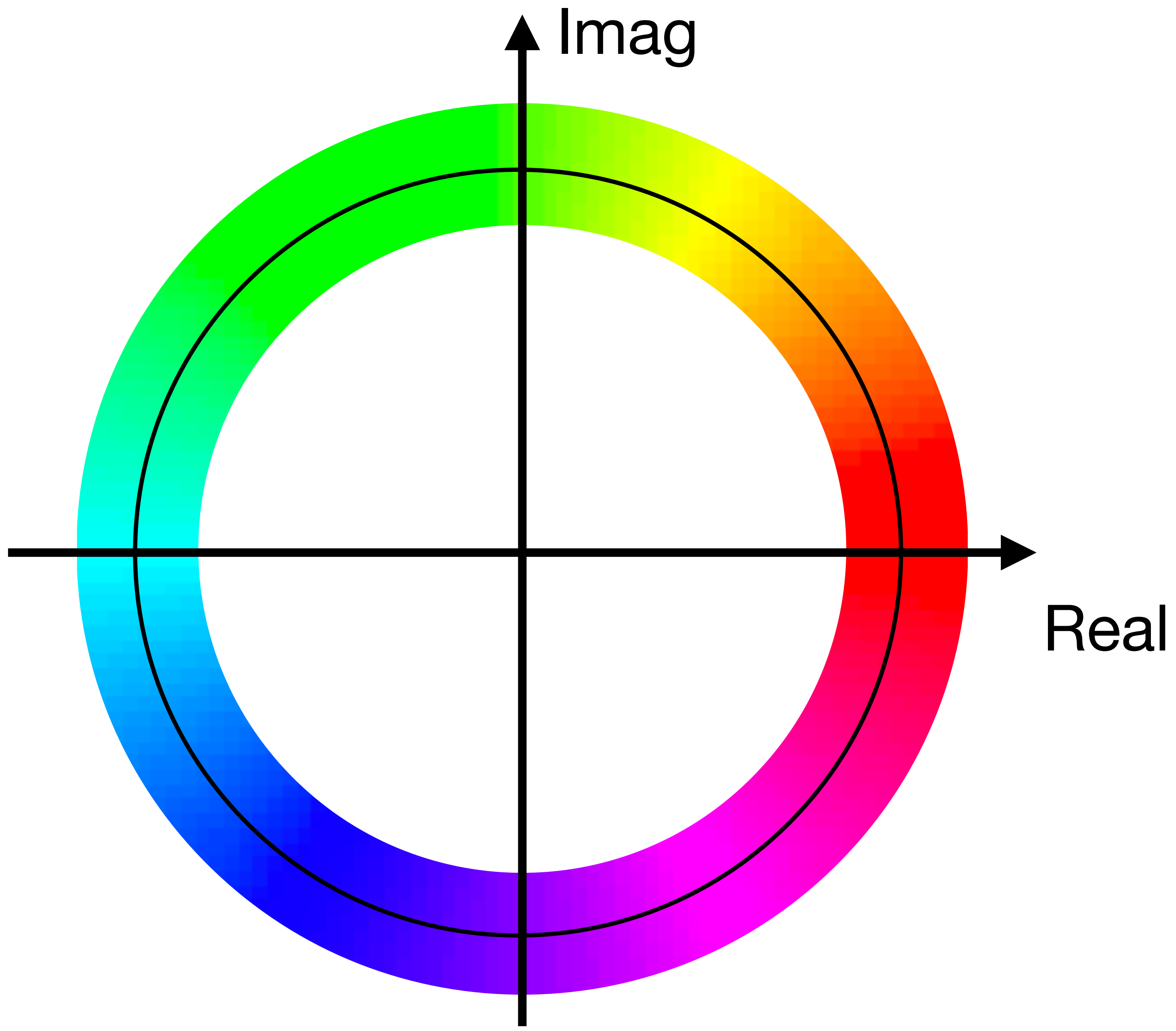
For:
u=1
N=16

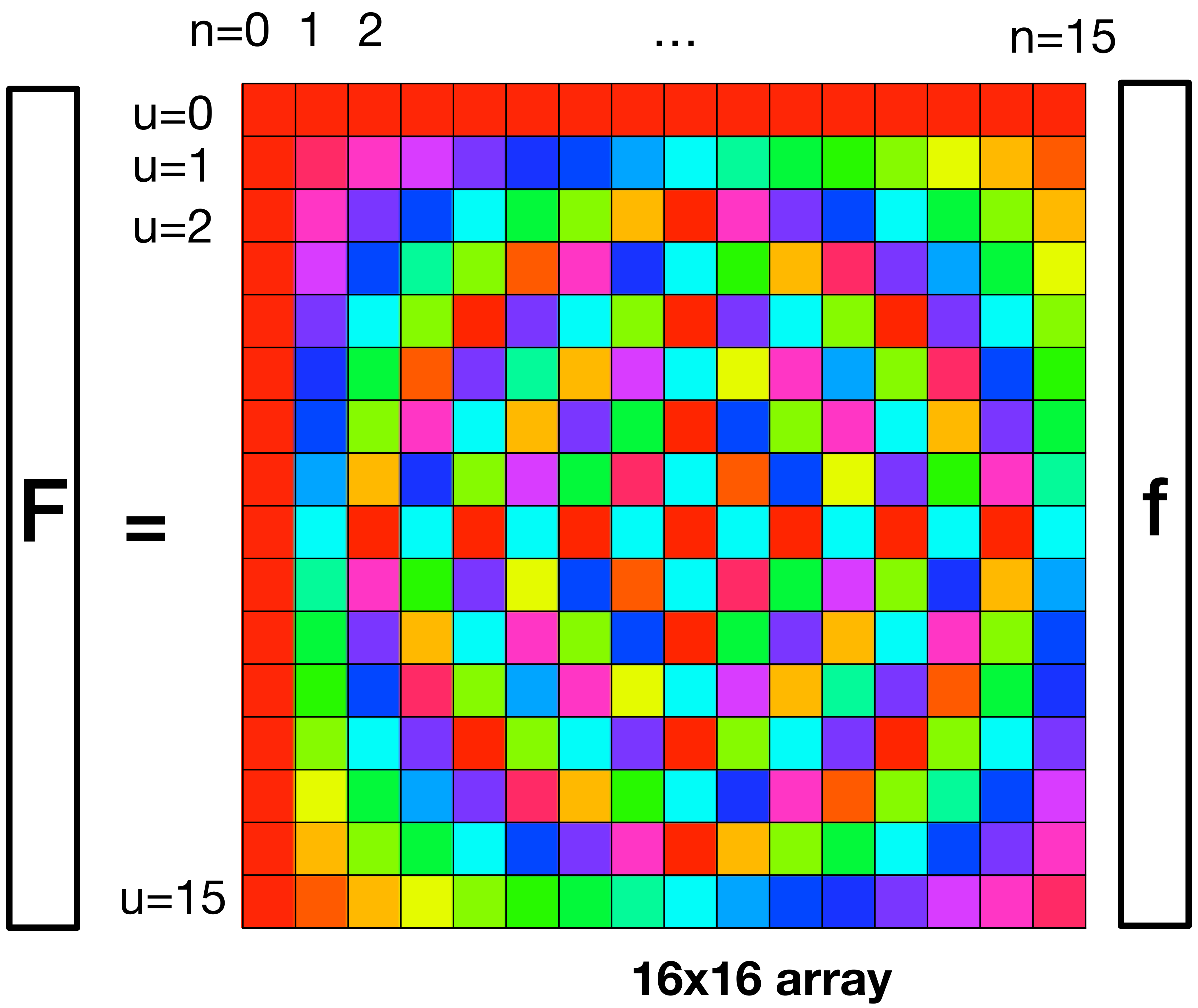


Visualizing the transform coefficients

$$\exp\left(-2\pi j \frac{un}{N}\right)$$

For N=16

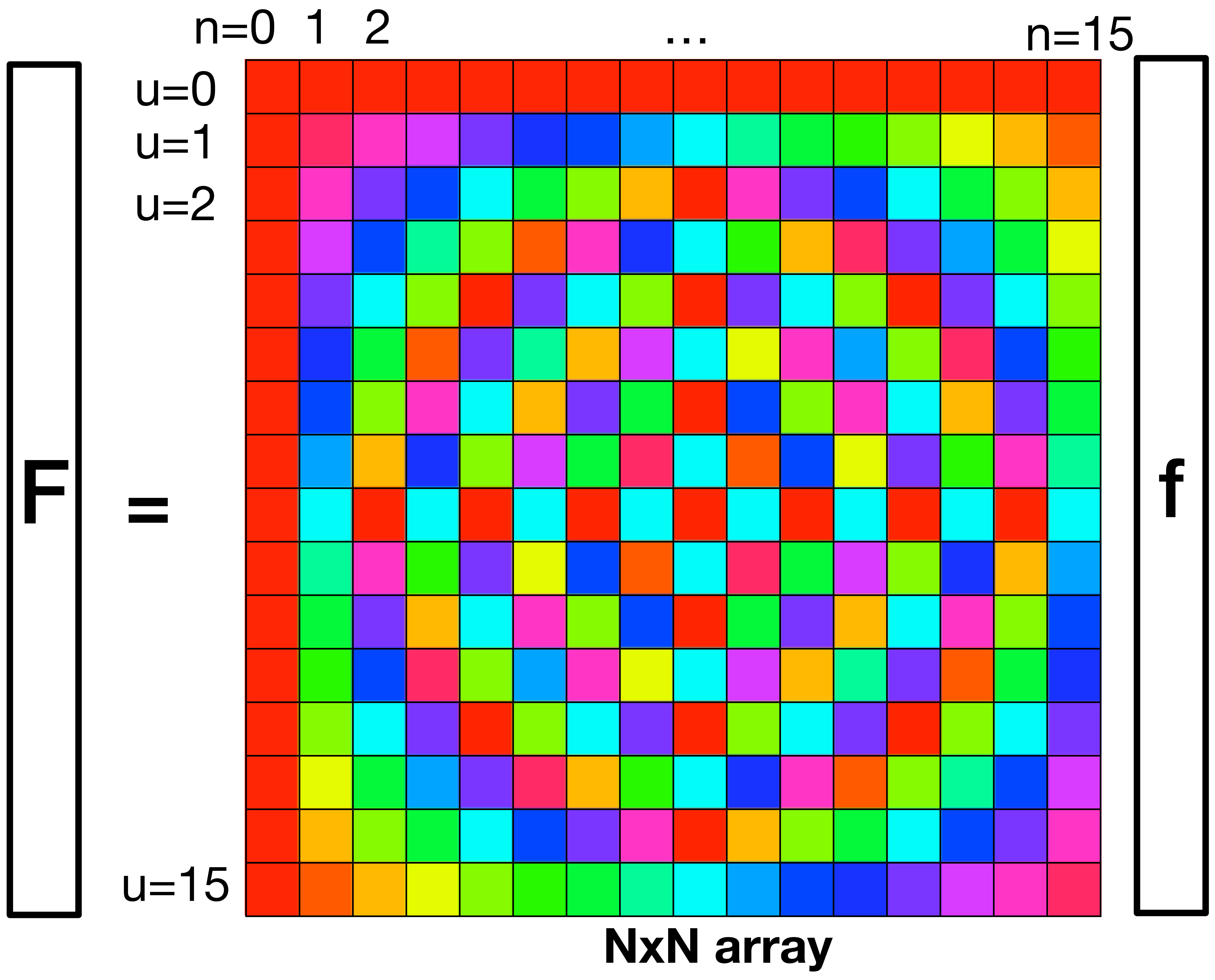




The inverse of the Discrete Fourier transform

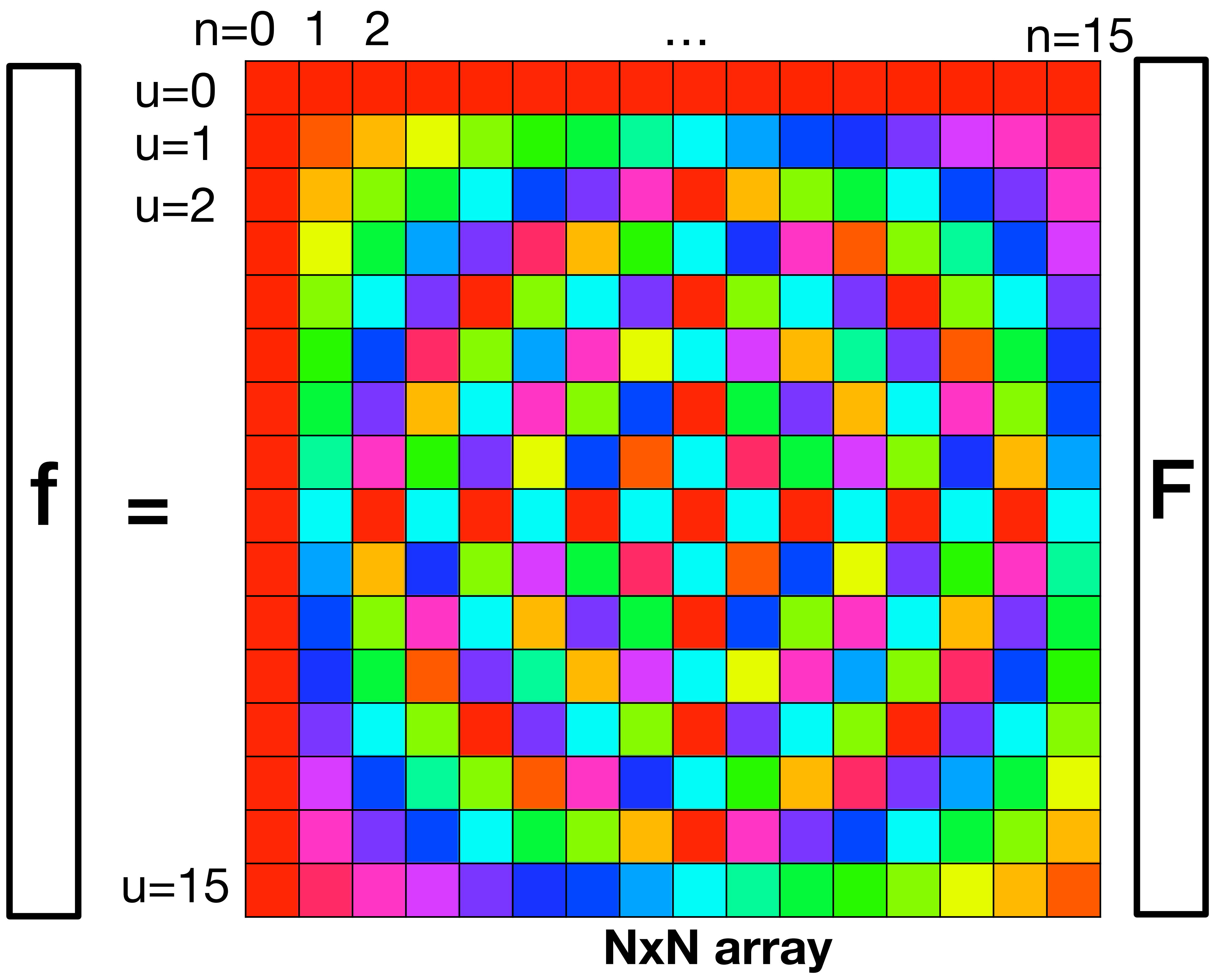
Discrete Fourier Transform (DFT):

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



Its inverse:

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



For images, the 2D DFT

1D Discrete Fourier Transform (DFT) transforms a signal *f* [*n*] into *F* [*u*] as:

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

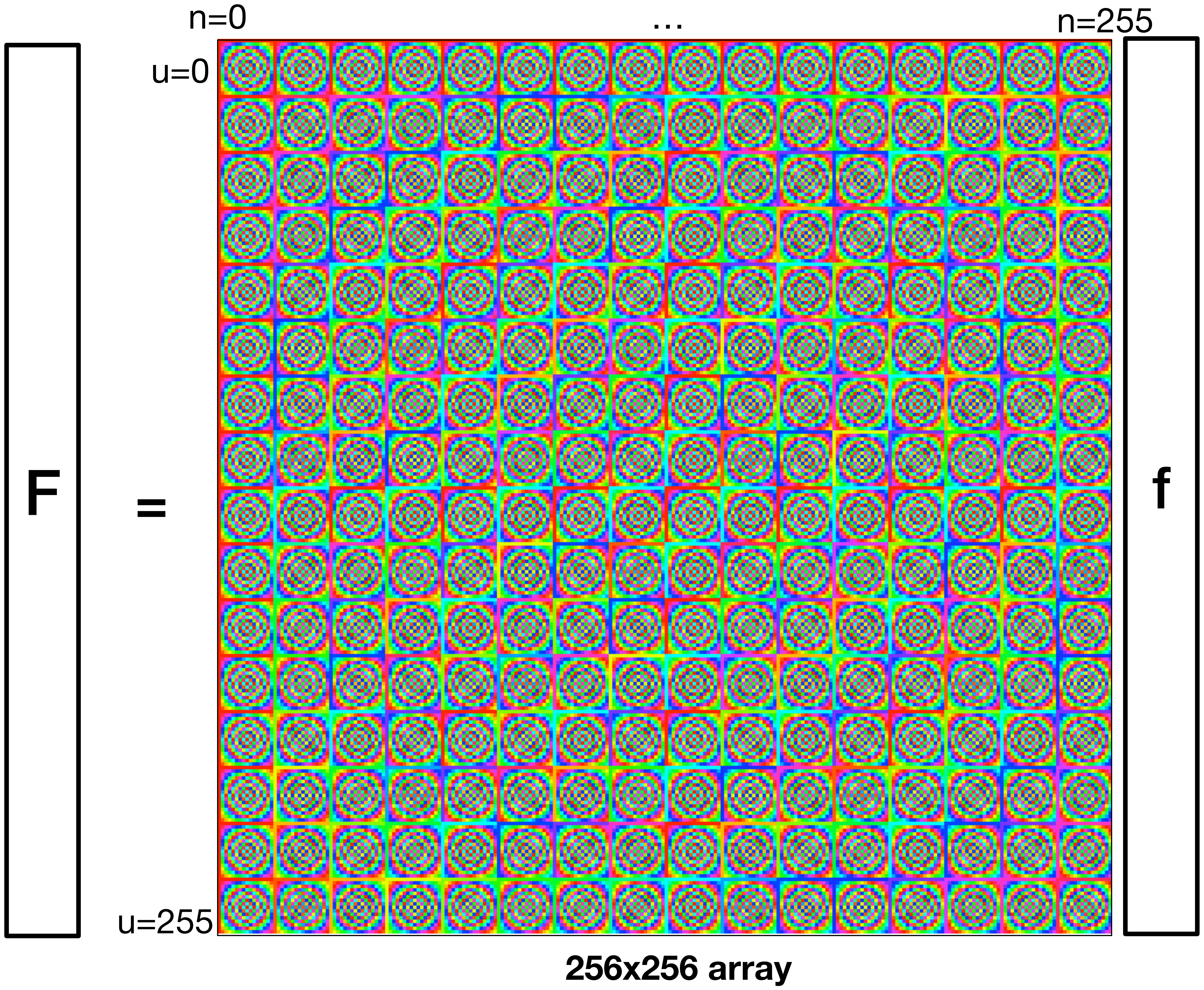
2D Discrete Fourier Transform (DFT) transforms an image *f* [*n,m*] into *F* [*u,v*] as:

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

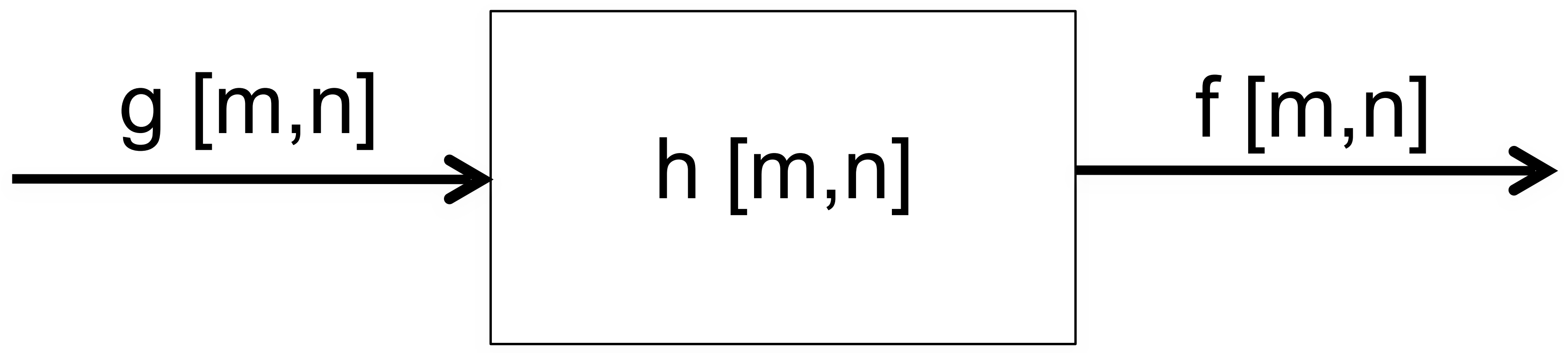
Visualizing the 2D DFT coefficients

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

For N=M=16



A remarkable property of Fourier transform



In the spatial domain, the output of f is the convolution:

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

In the frequency domain:

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

Terminology:

Impulse response: h [m,n]

Transfer function: $H[u,v]$

Dual convolution property

The Fourier transform of the convolution is the product of Fourier transforms

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

The Fourier transform of the product is the convolution of Fourier transforms

$$f[n,m] = g[n,m]\,h[n,m] \quad \longleftrightarrow \quad F[u,v] = \frac{1}{NM} G[u,v] \circ H[u,v]$$

Visualizing the image Fourier transform

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

The values of $F[u,v]$ are complex.

Using the real and imaginary components:

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

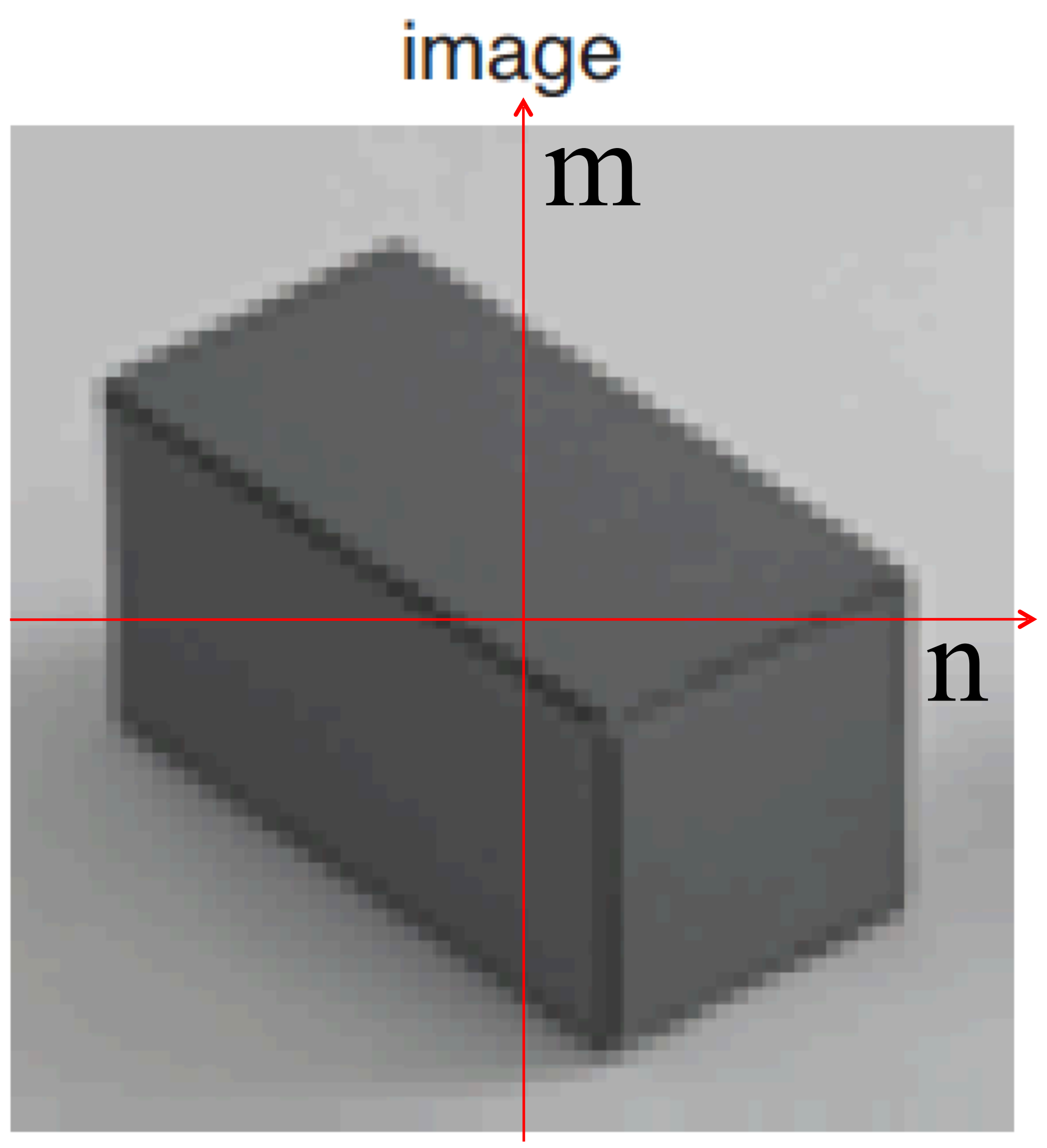
Or using a polar decomposition:

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

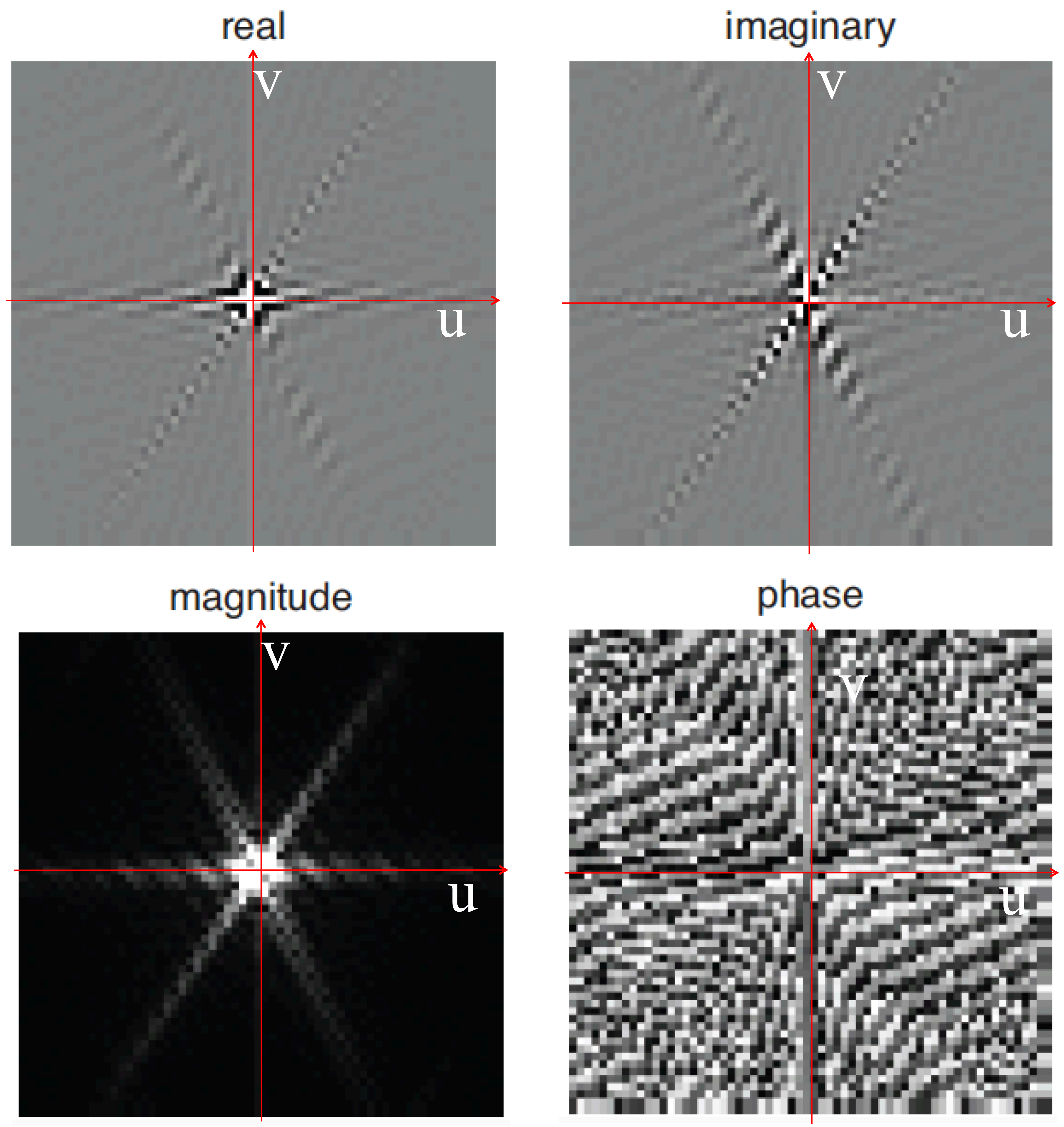
Amplitude — Phase

Visualizing the image Fourier transform

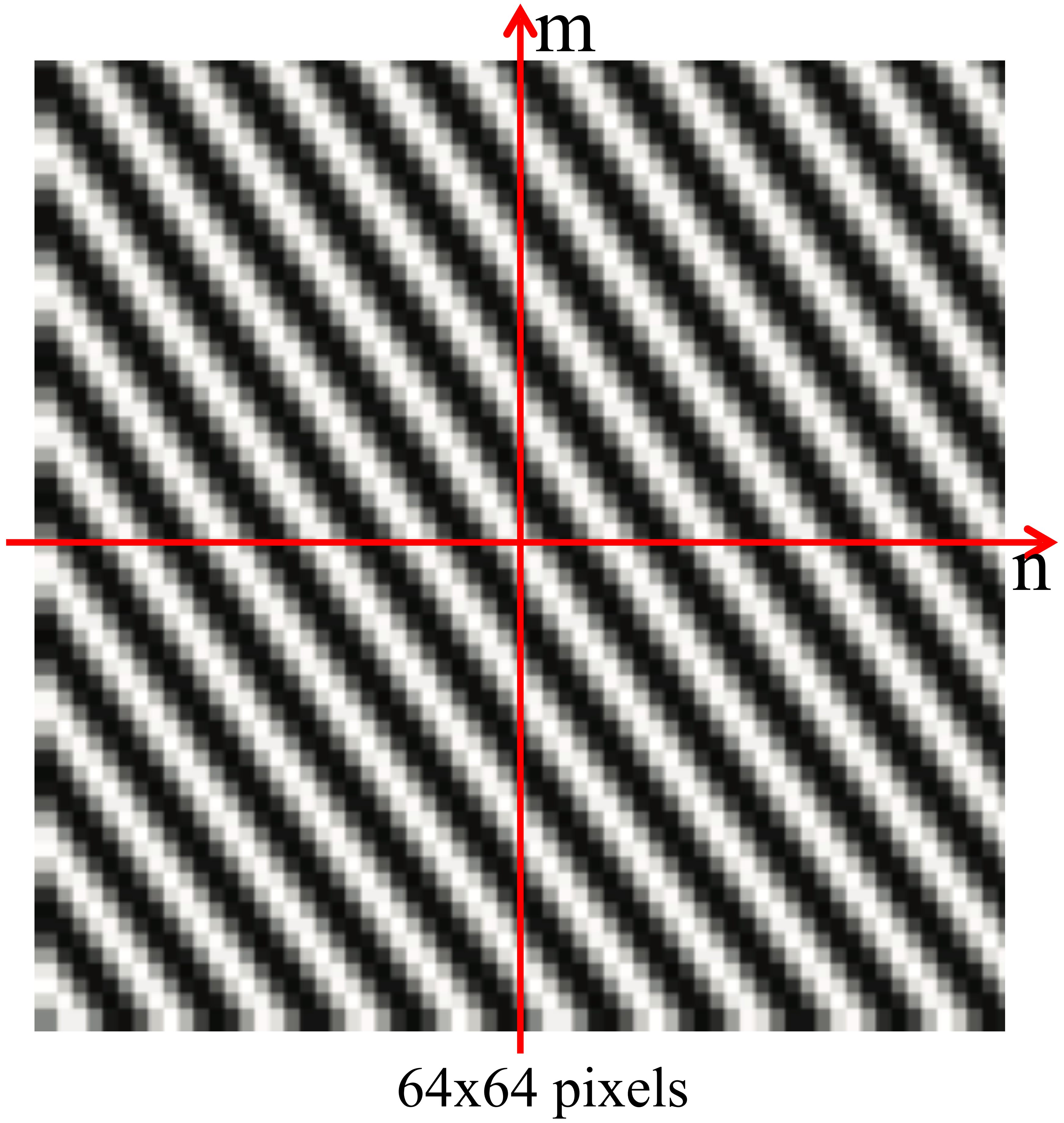
$f[n, m]$

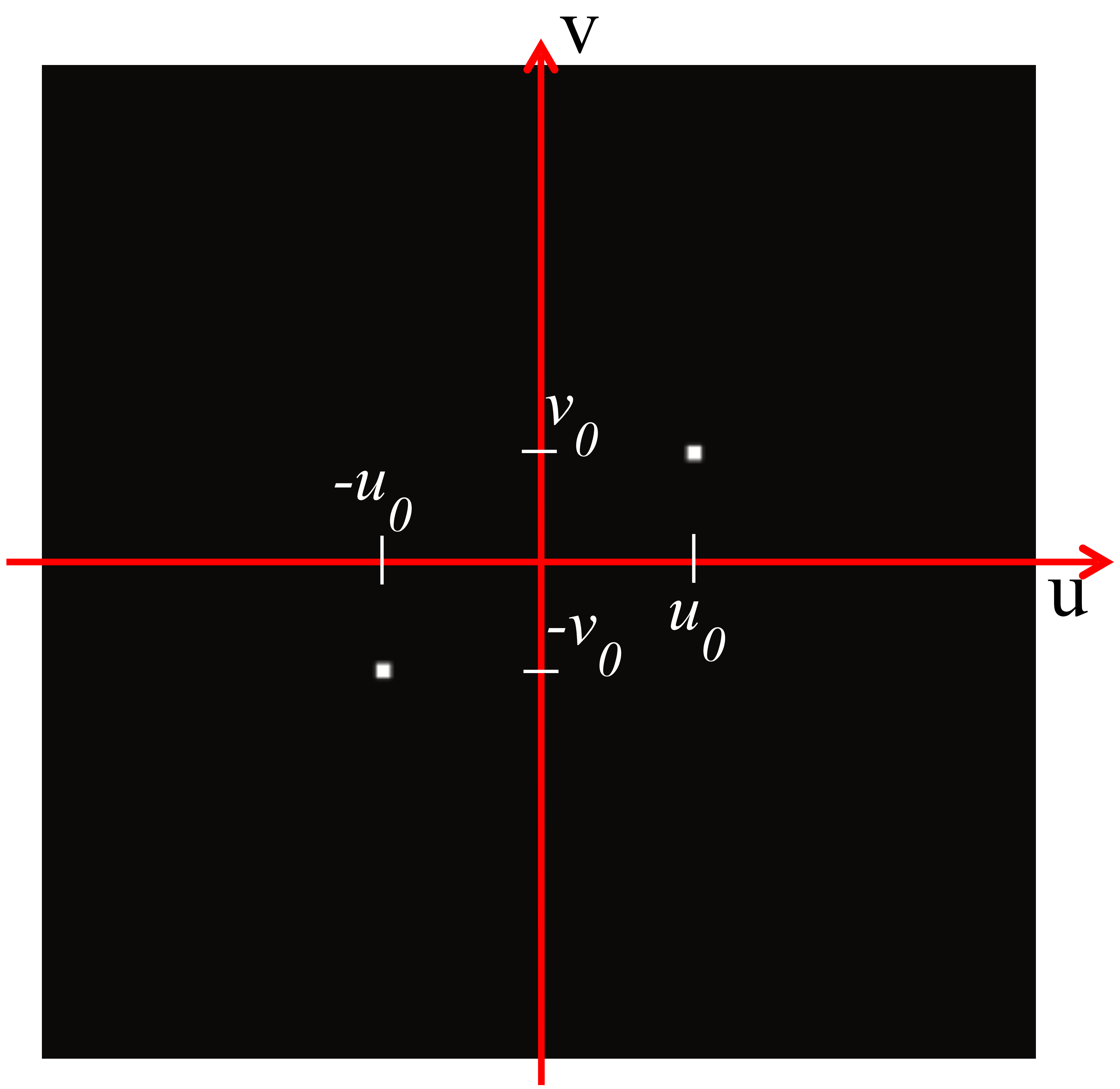


$F[u, v]$



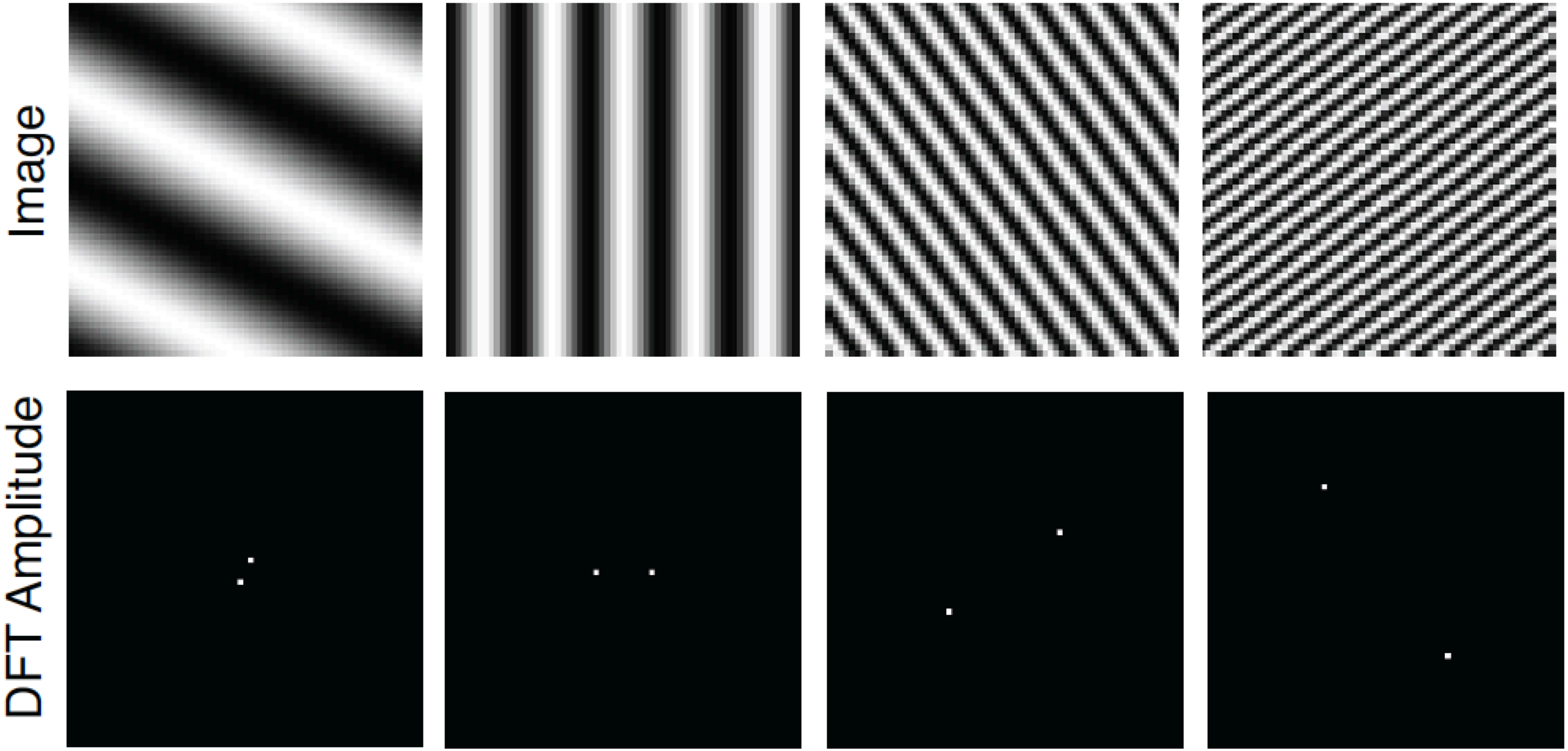
Simple Fourier transforms





$$\cos\left(2\pi\left(\frac{u_0\, n}{N} + \frac{v_0\, m}{M}\right)\right) \longleftrightarrow \frac{1}{2}\left(\delta\left[u - u_0, v - v_0\right] + \delta\left[u + u_0, v + v_0\right]\right)$$

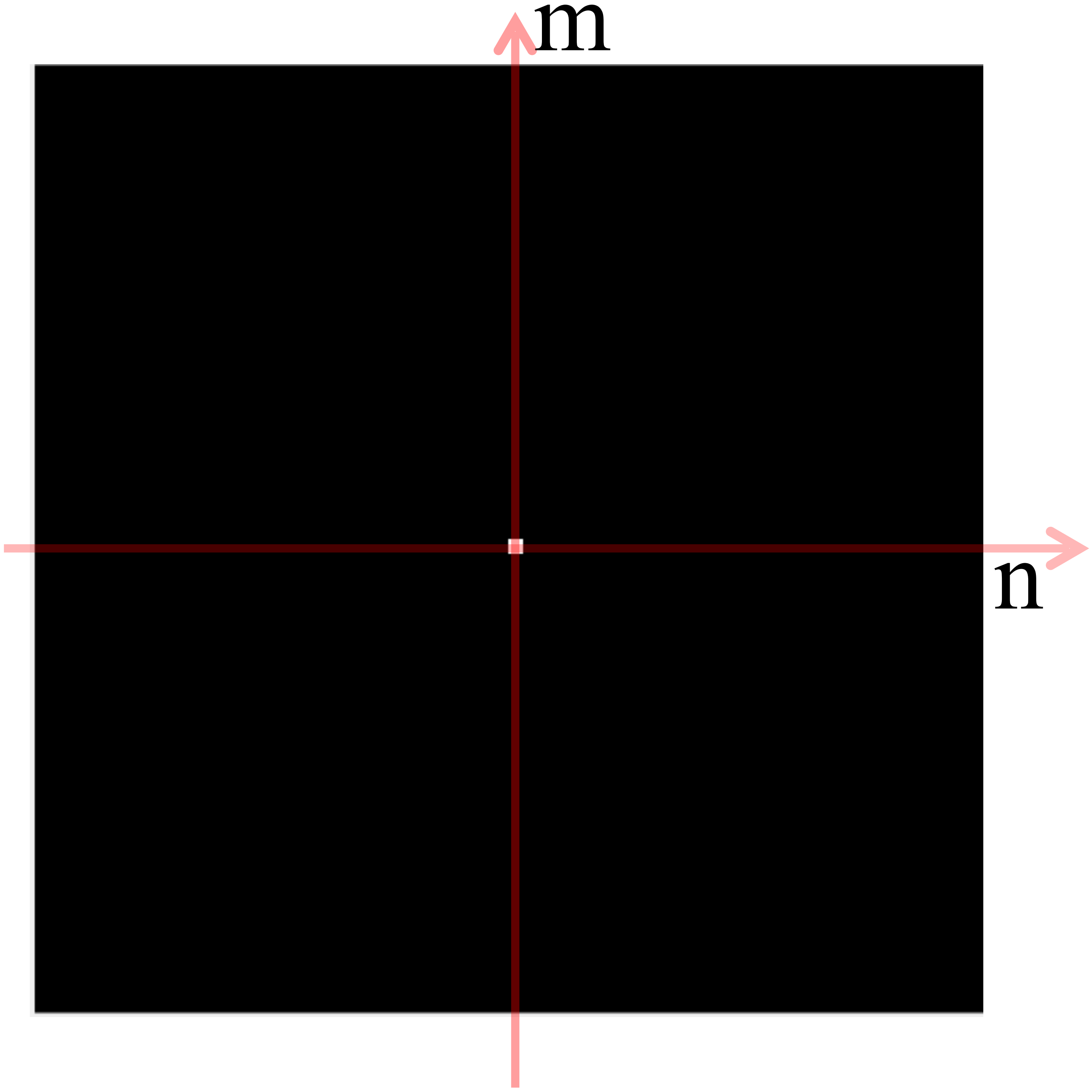
Simple Fourier transforms



Images are 64x64 pixels. The wave is a cosine, therefore DFT phase is zero.

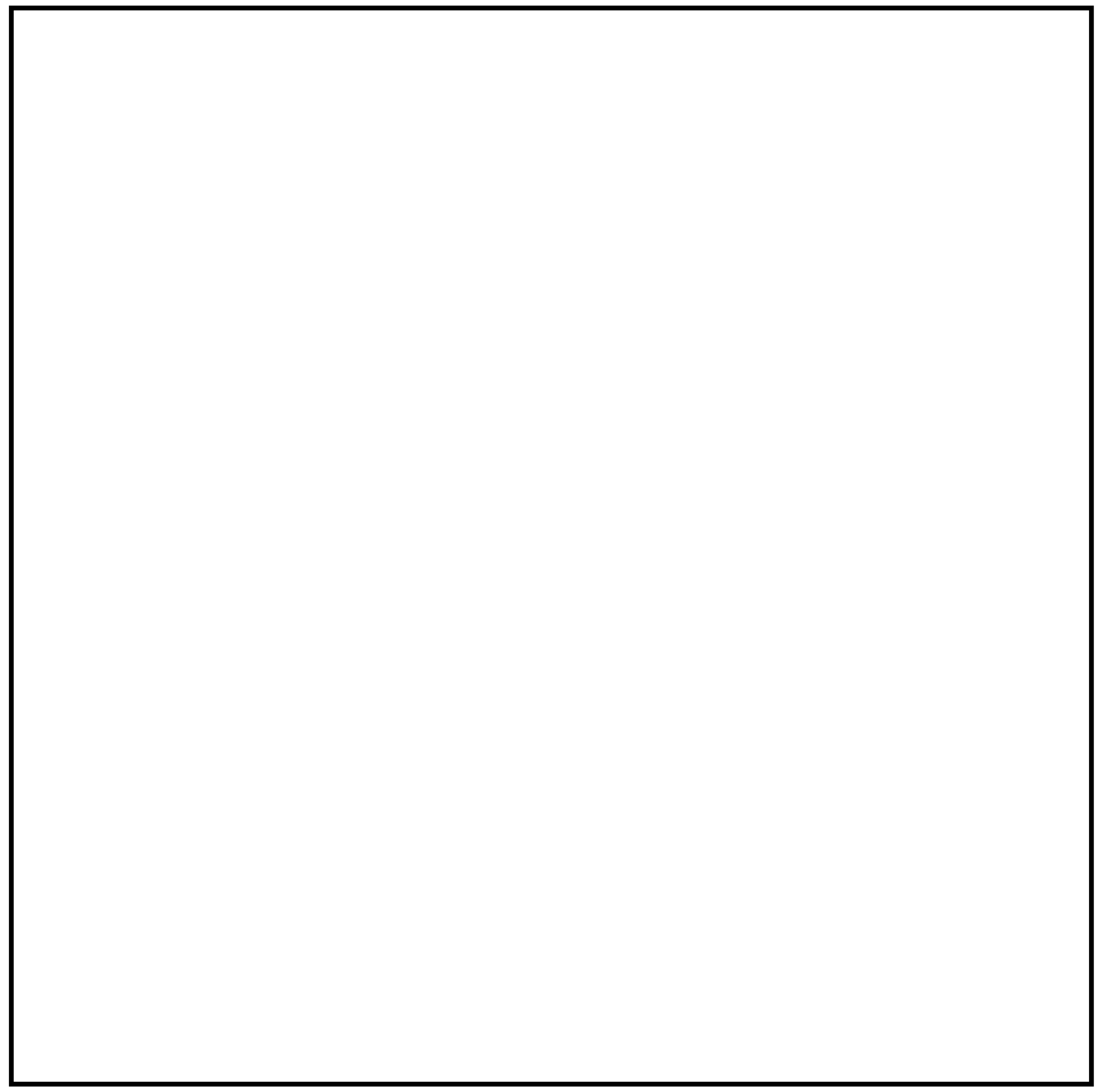
Some important Fourier transforms

Image



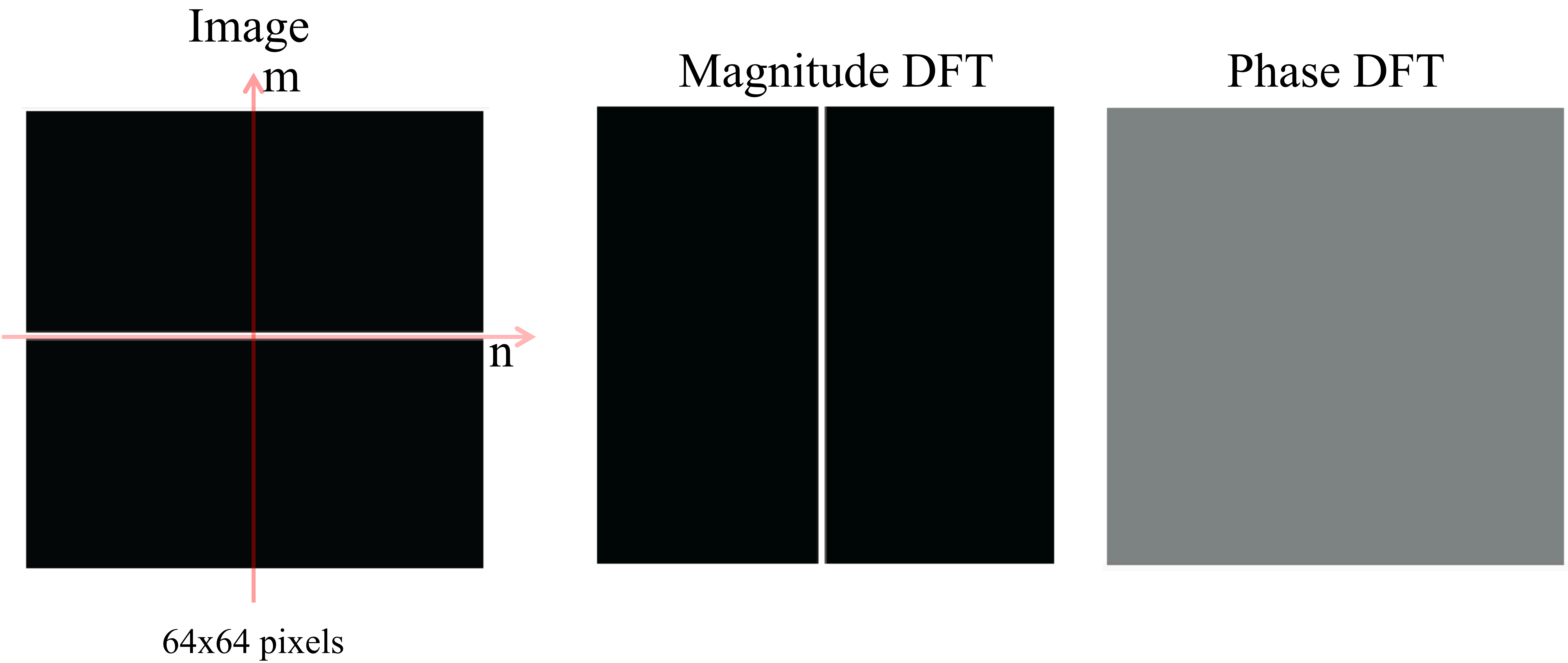
64x64 pixels

Magnitude DFT

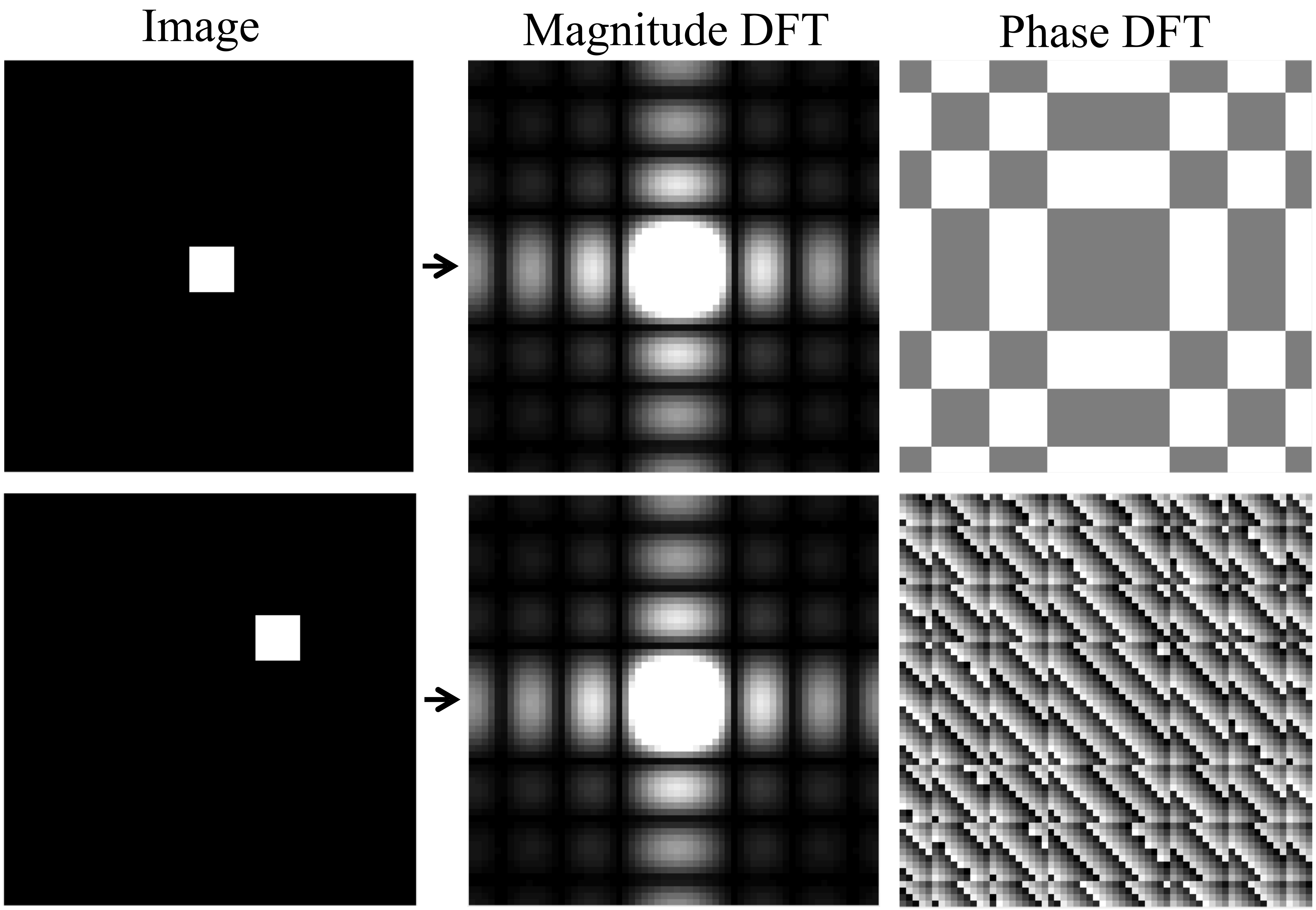


Phase DFT

Some important Fourier transforms



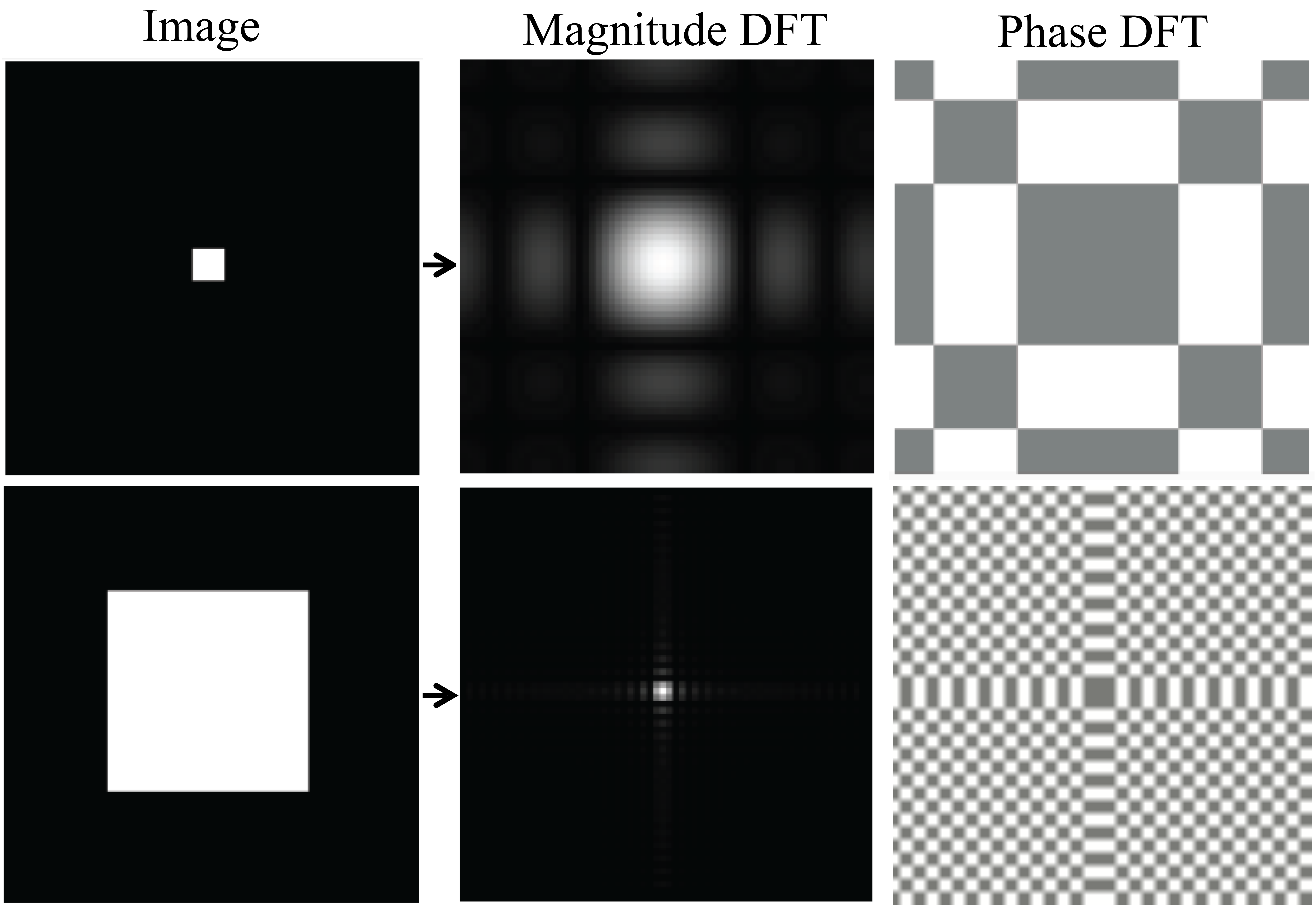
Some important Fourier transforms



Translation

Shifts of an image only produce changes on the phase of the DFT.

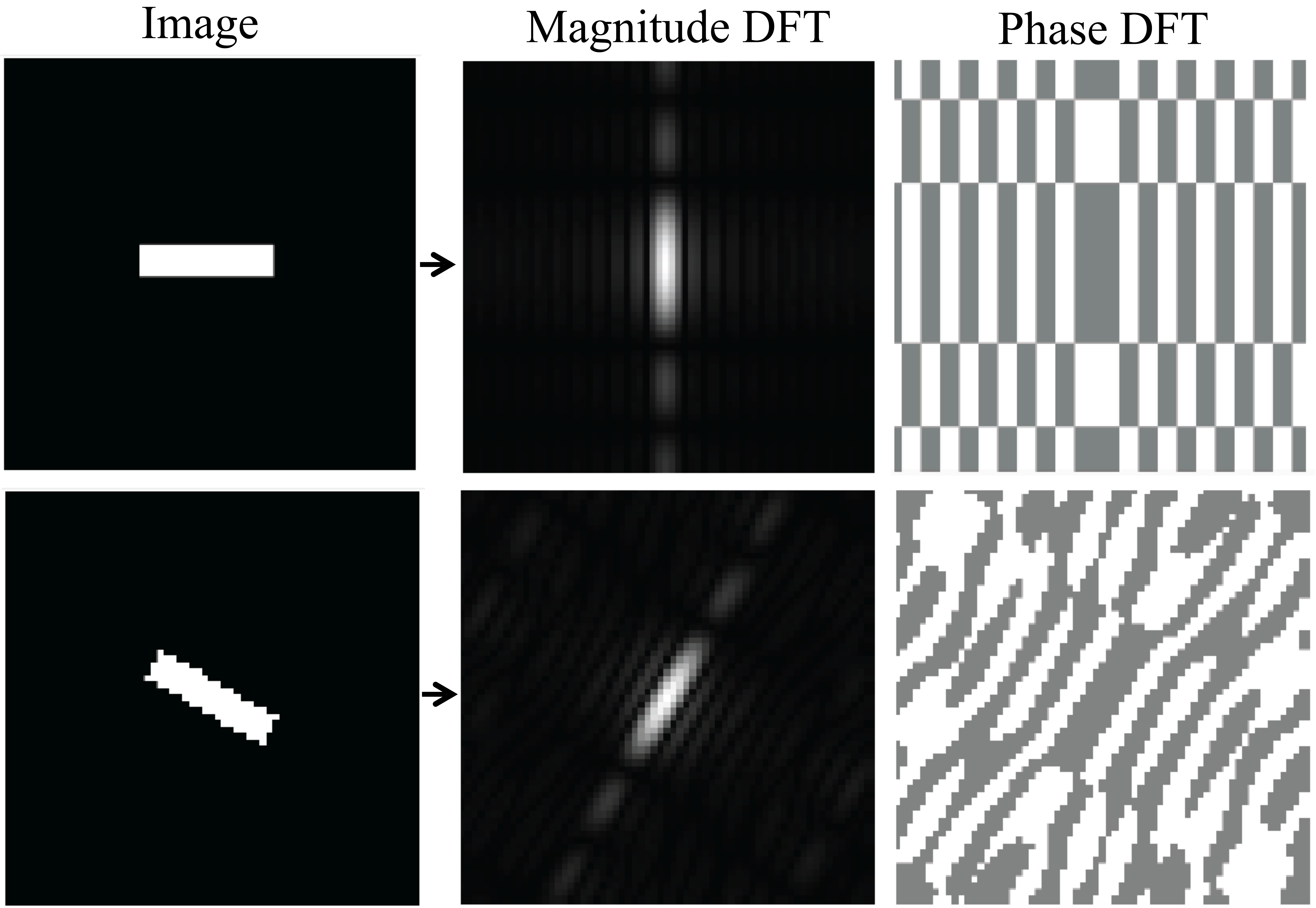
Some important Fourier transforms



Scale

Small image details produce content in high spatial frequencies

Some important Fourier transforms

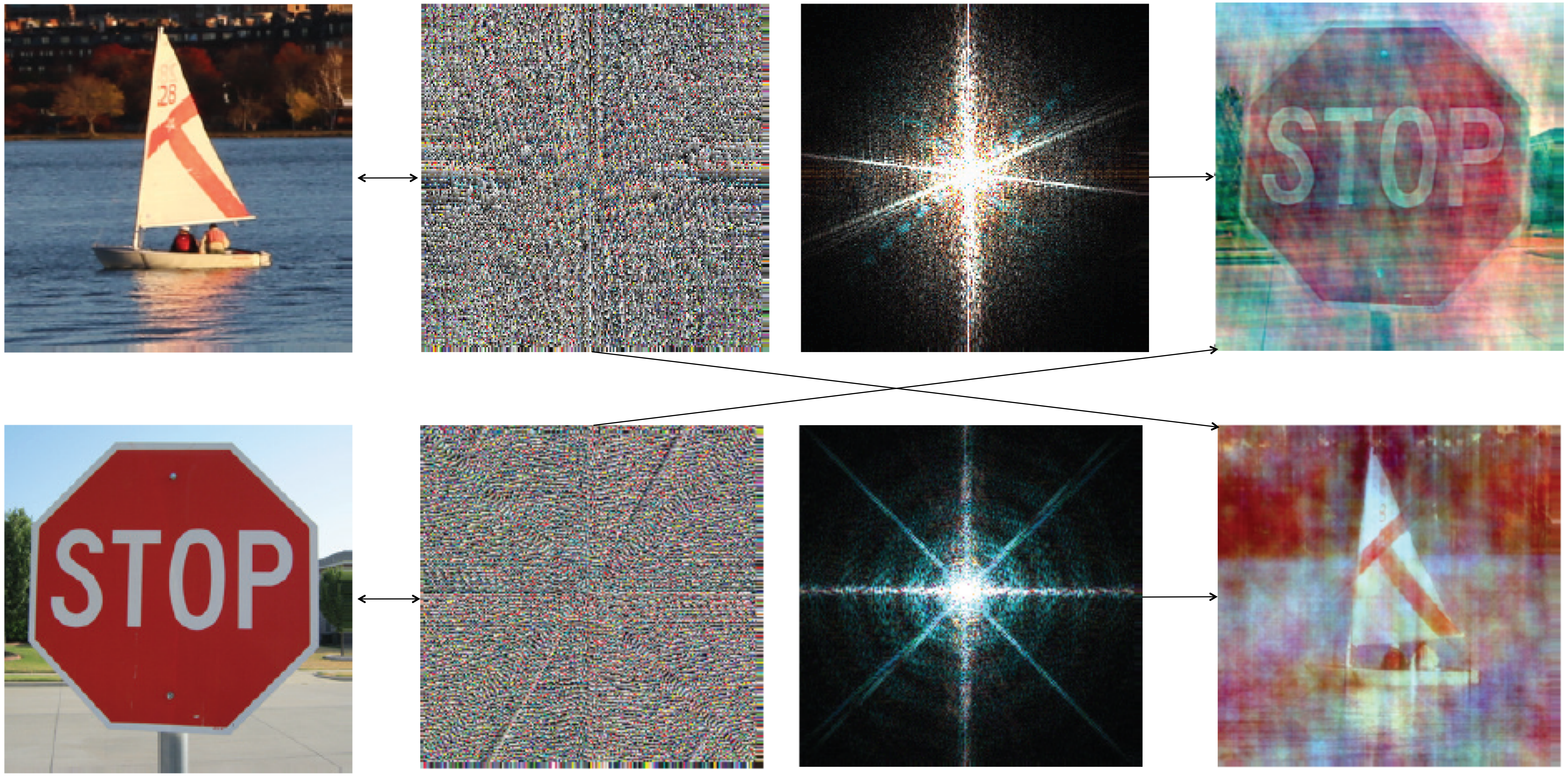


Orientation

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

Phase and Magnitude

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

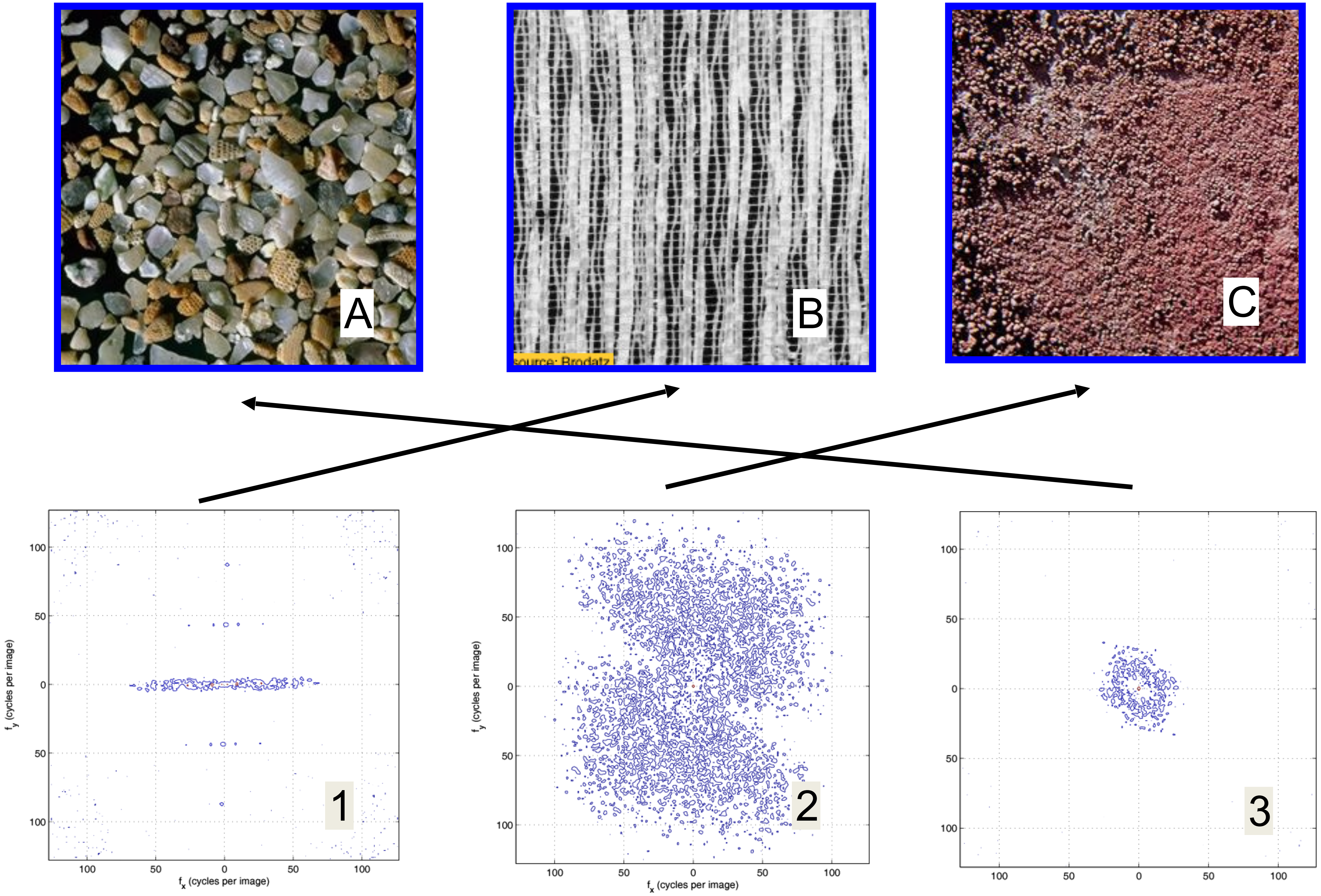


Each color channel is processed in the same way.

The DFT Game: find the right pairs

Images

DFT magnitude



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

The DFT Game: find the right pairs

(Solution in the class notes)

The inverse Discrete Fourier transform

2D Discrete Fourier Transform (DFT) transforms an image *f* [*n*,*m*] into *F* [*u*,*v*] as:

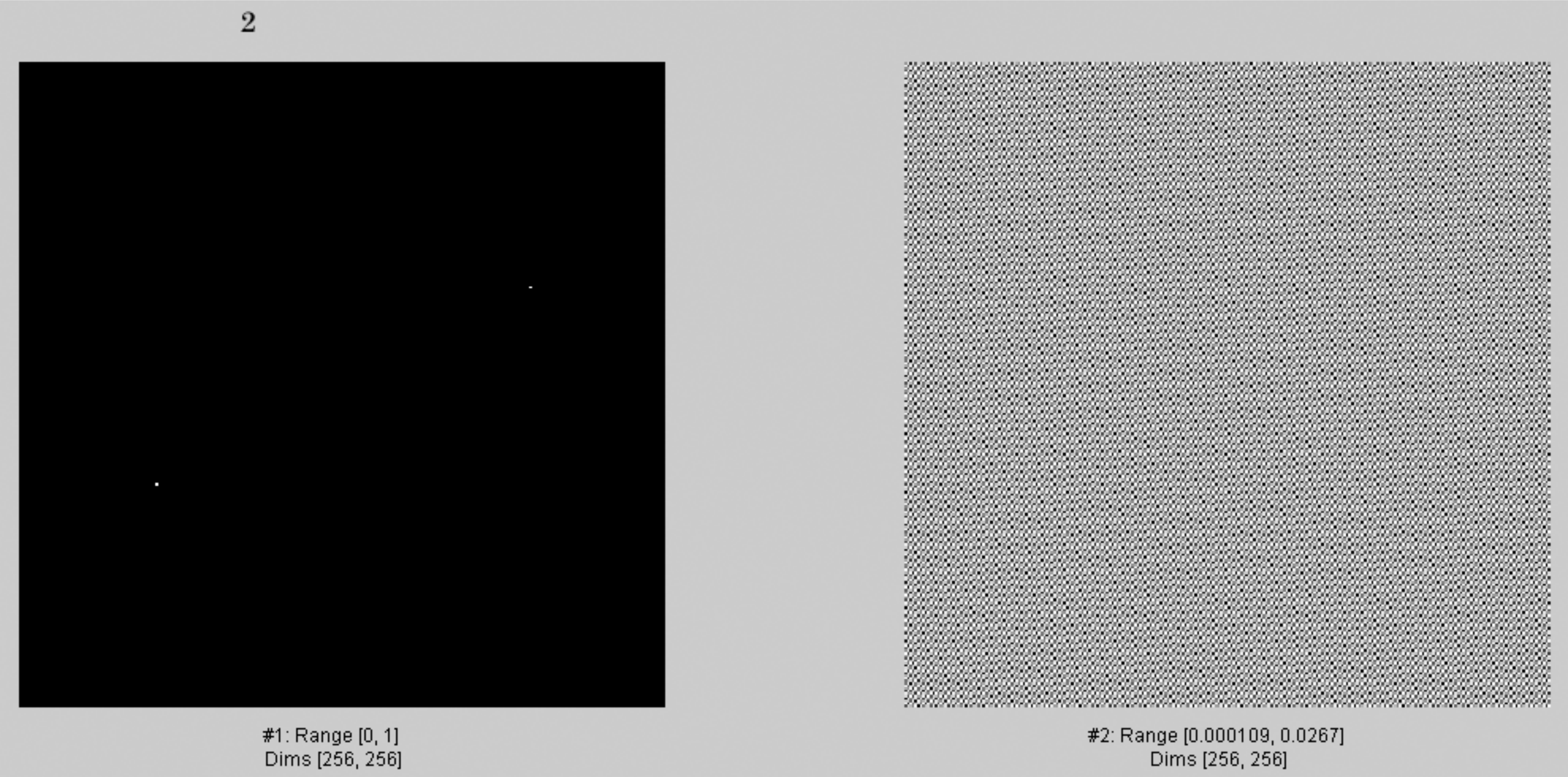
$$F[u,v] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

The inverse of the 2D DFT is:

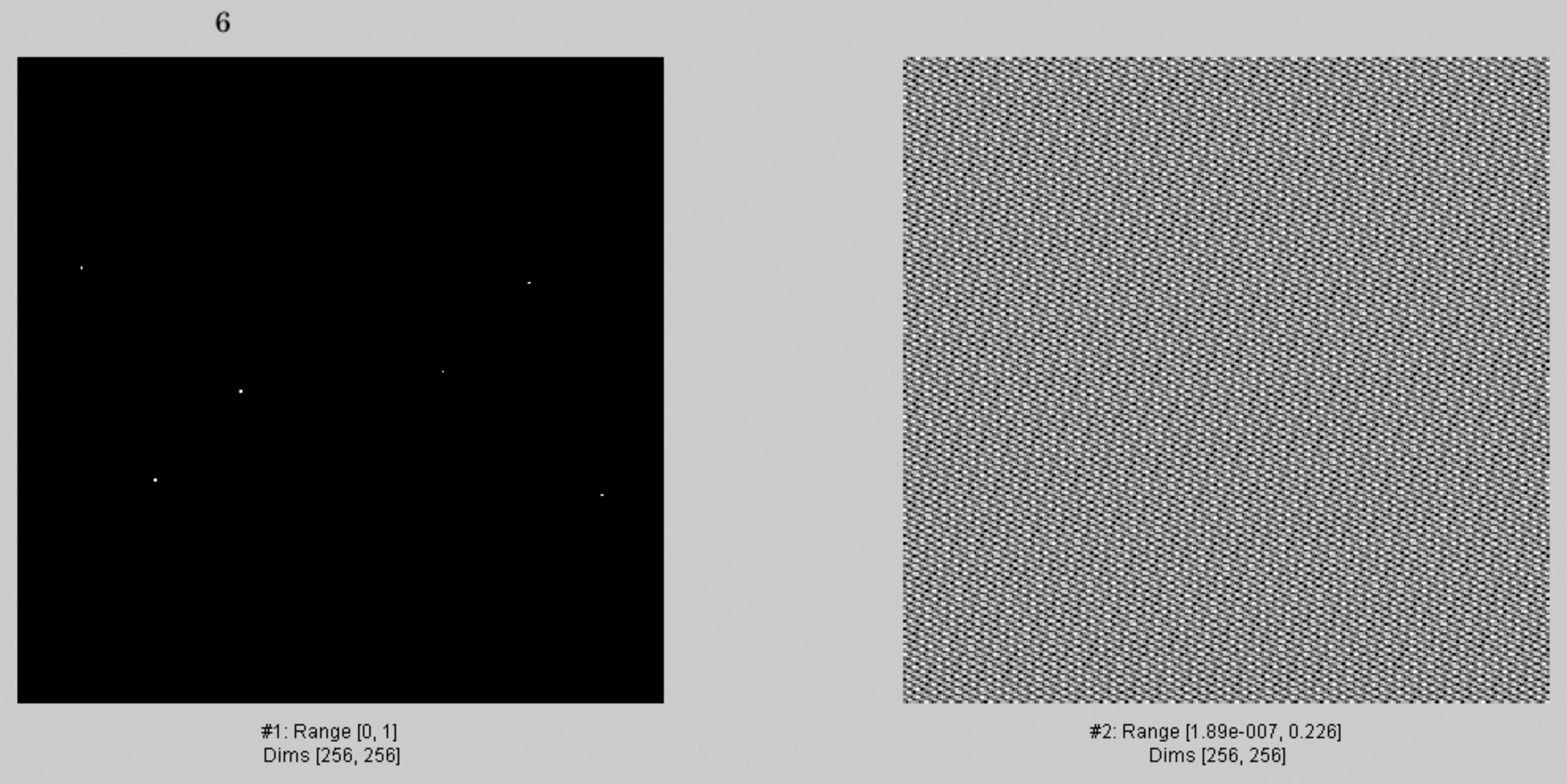
$$f[n,m] = \frac{1}{NM} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} F[u,v] \exp\left(+2\pi j \left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

How does summing waves ends up giving back a picture?

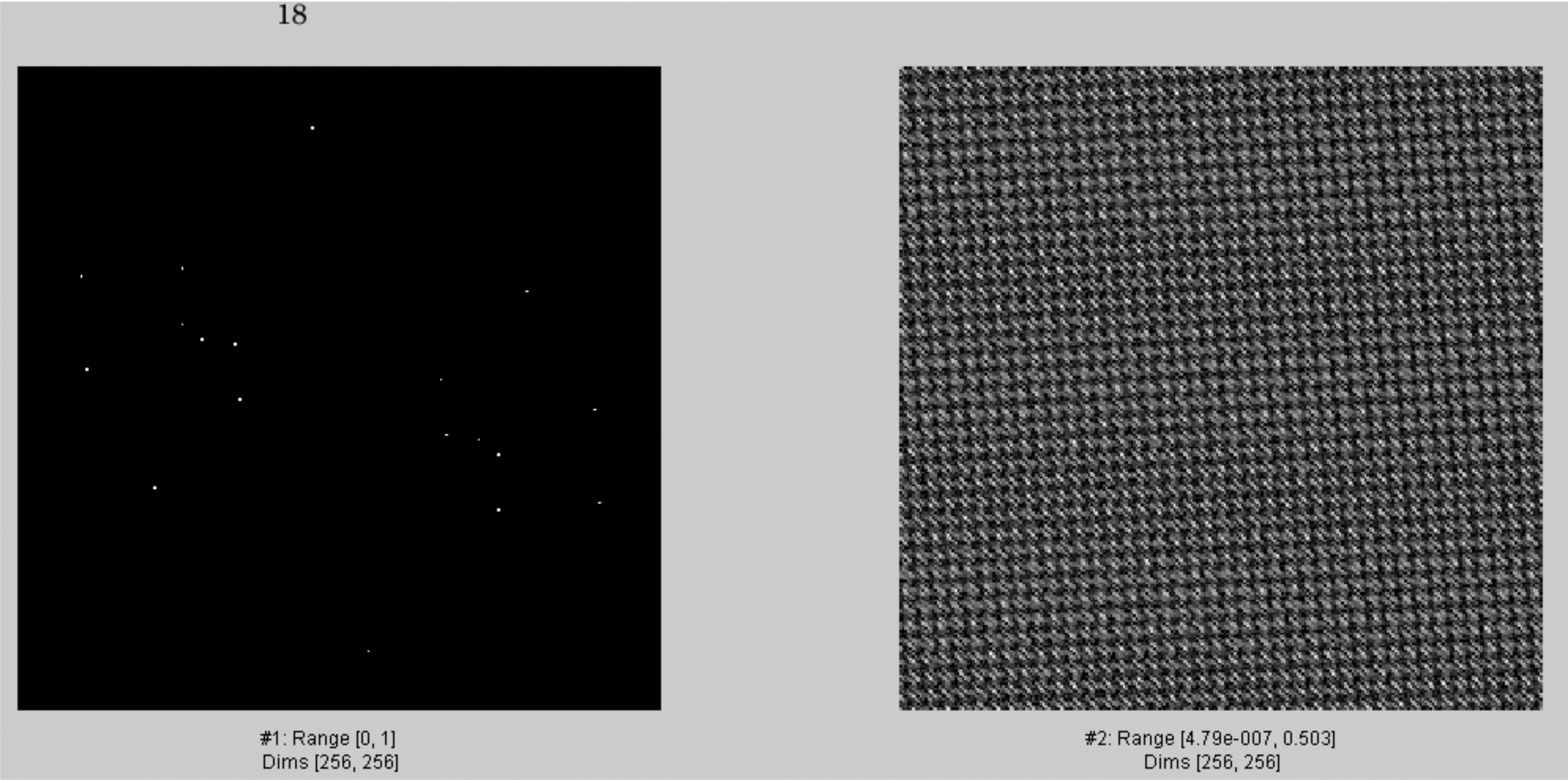
2



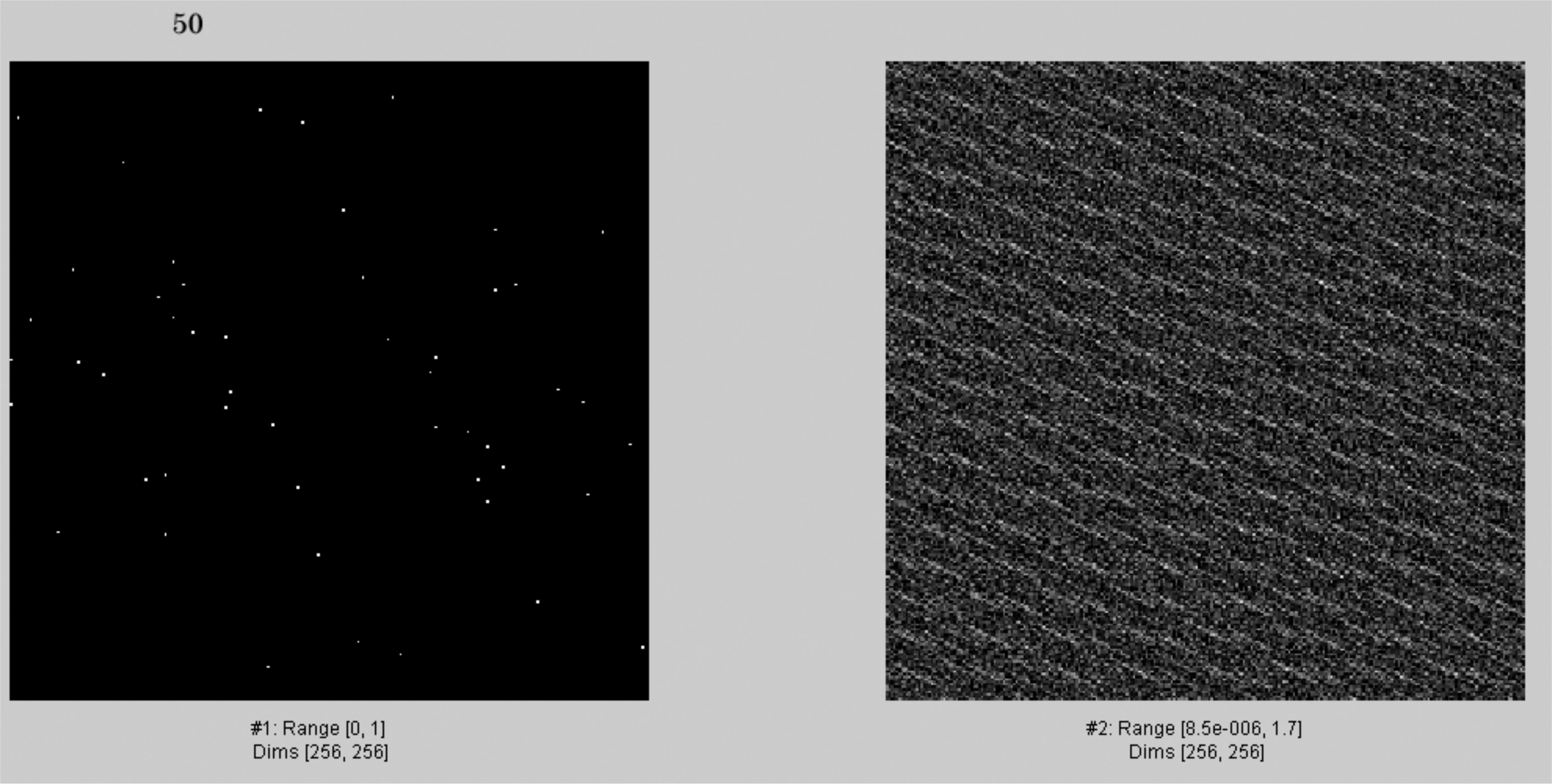
6



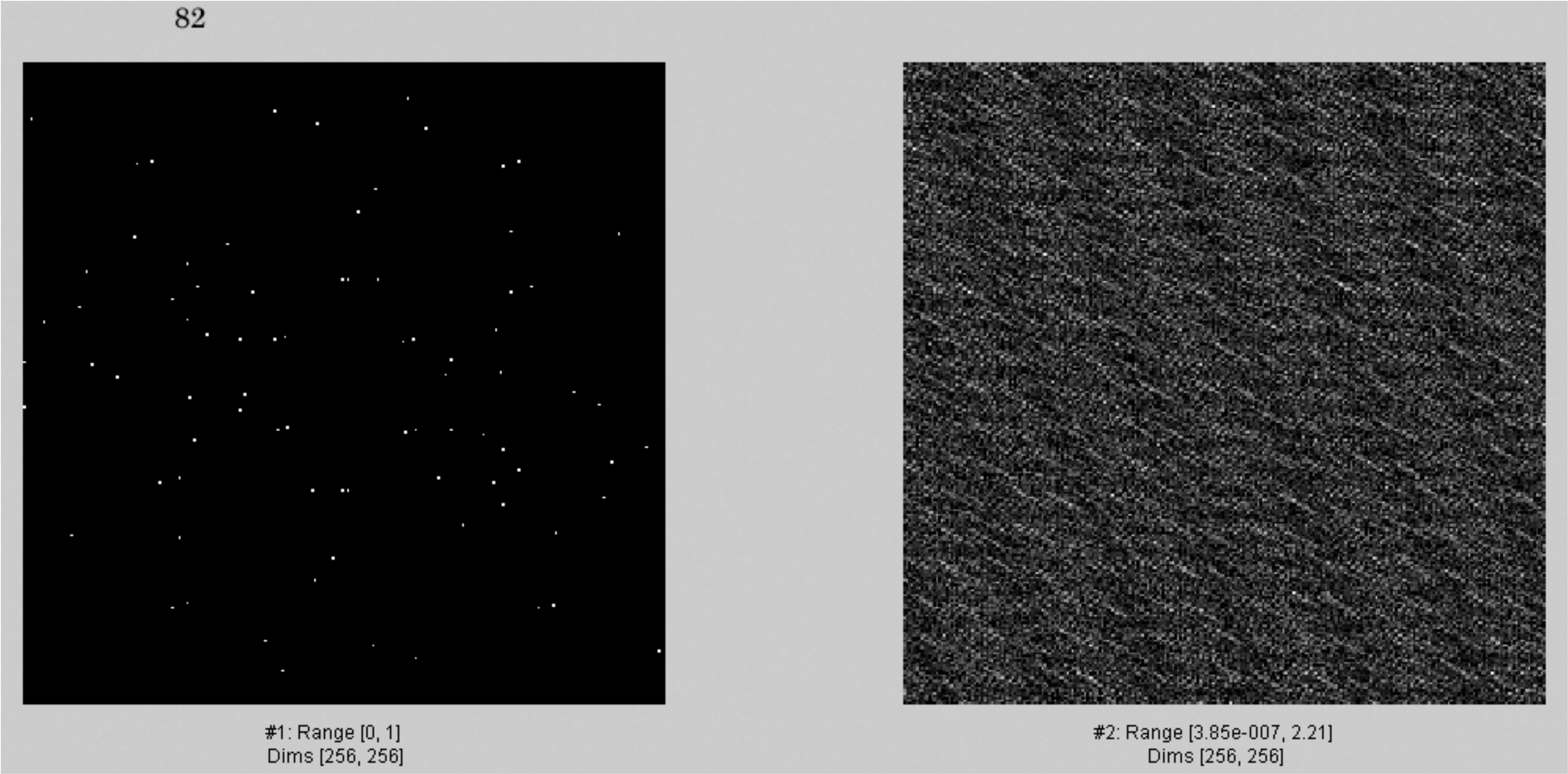
18



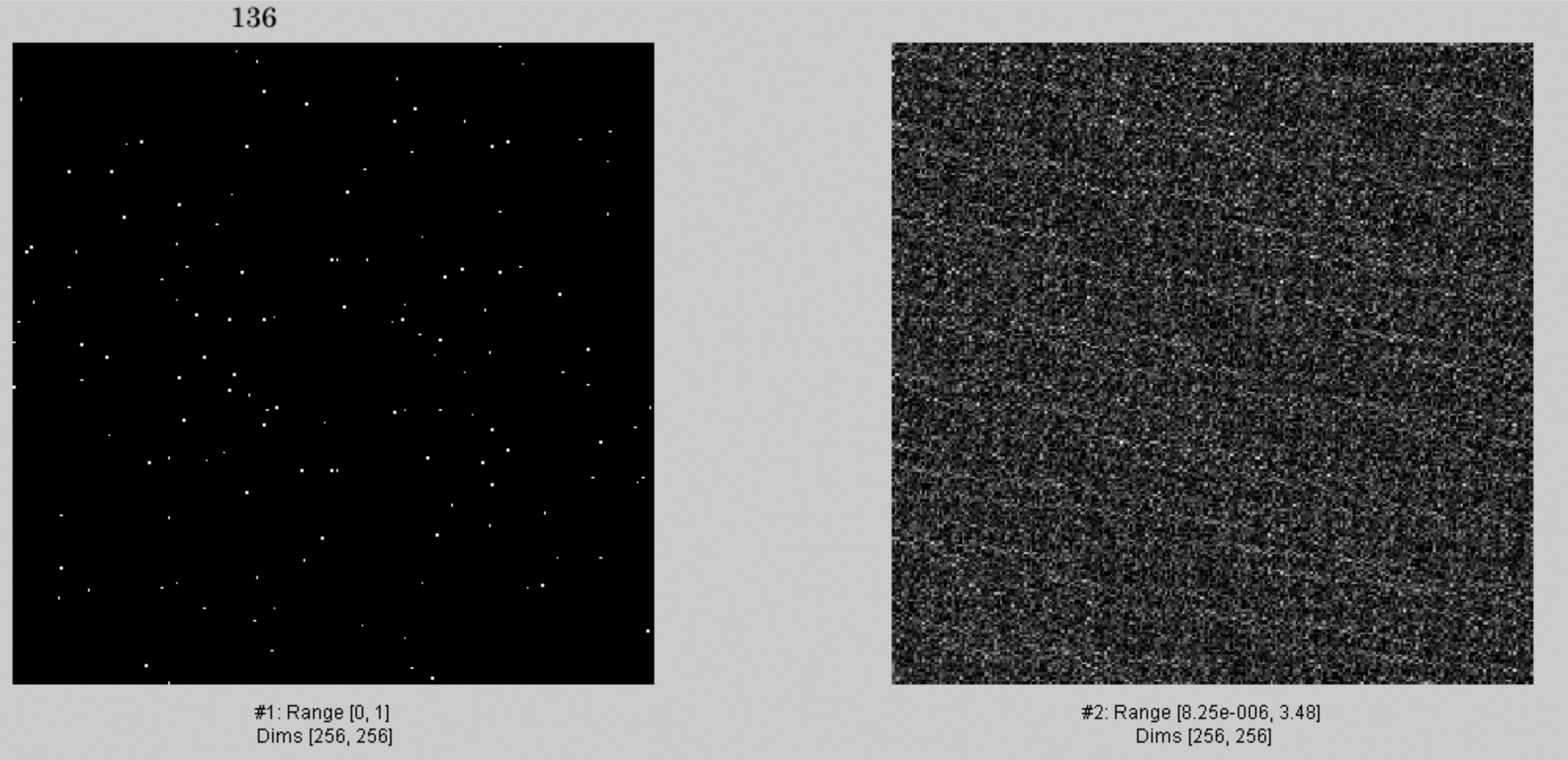
50



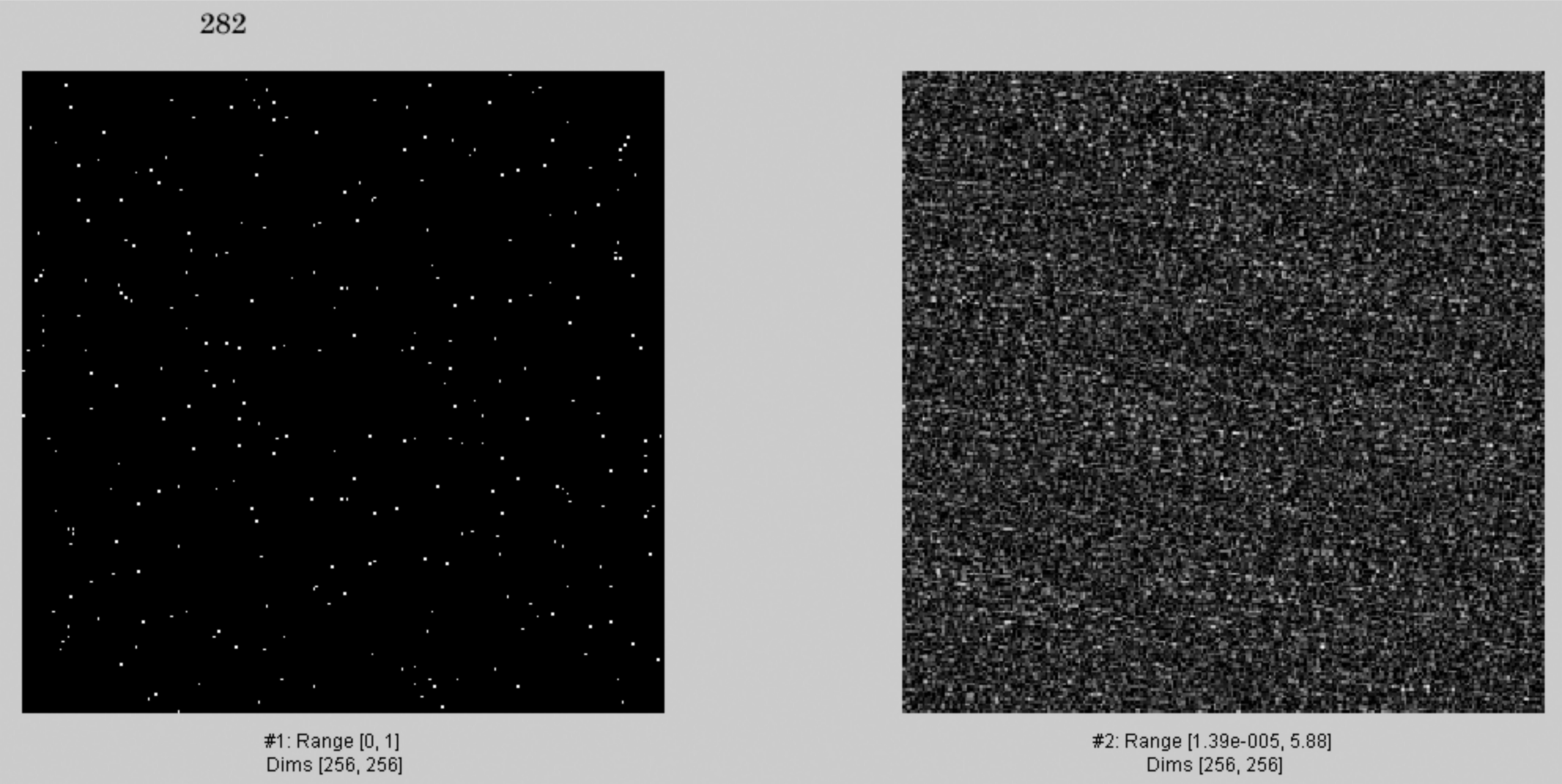
82



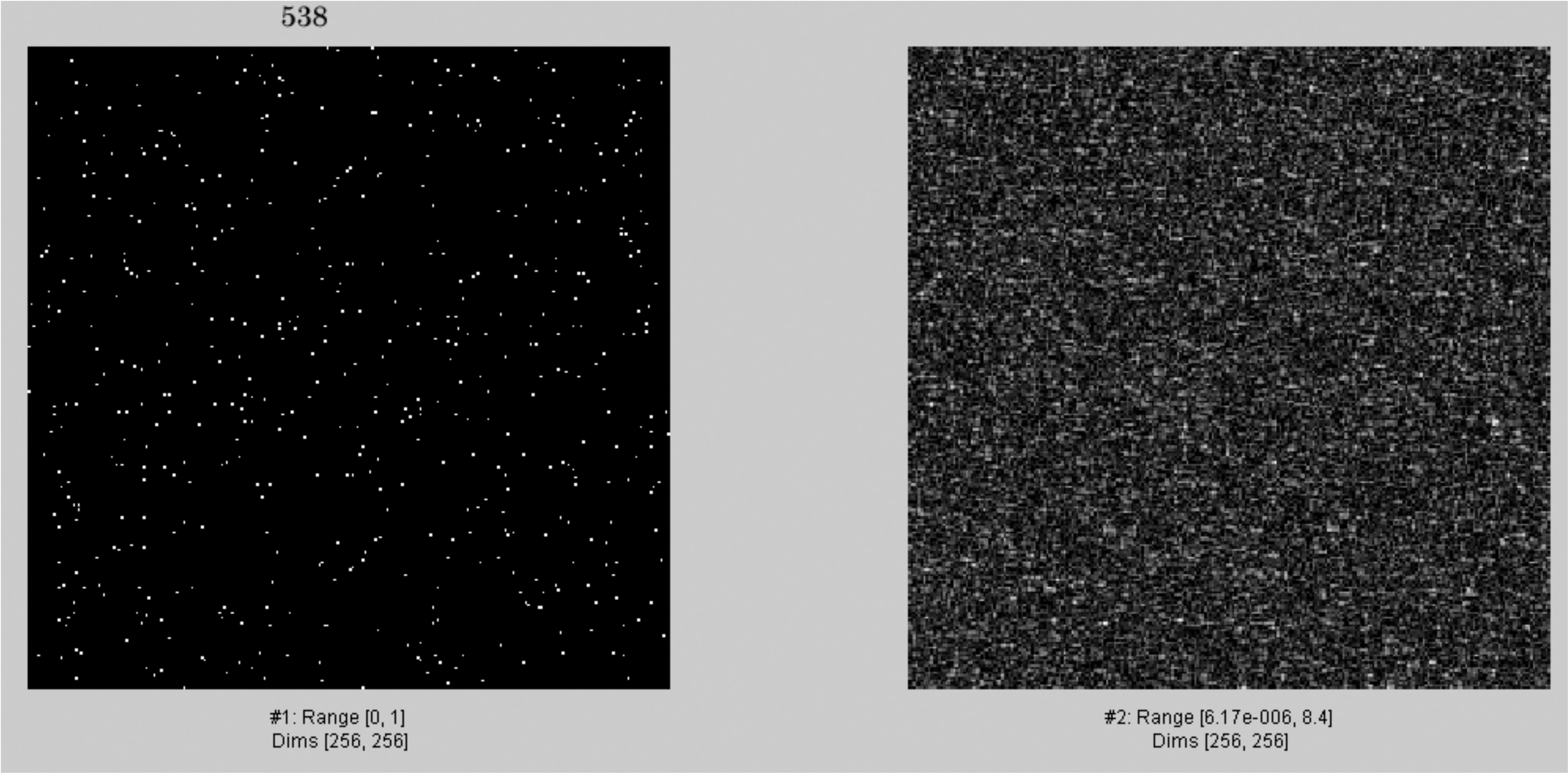
136



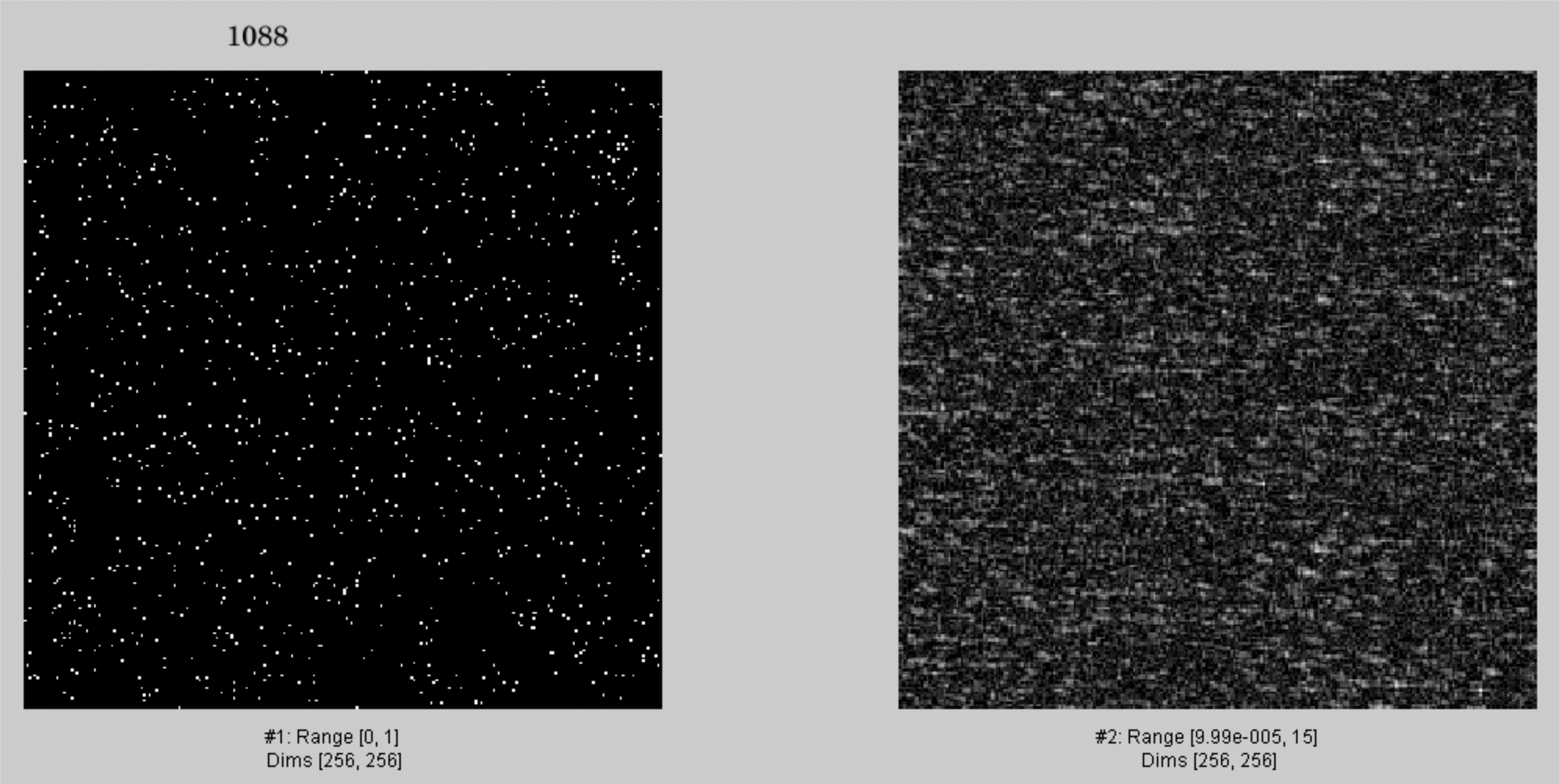
282



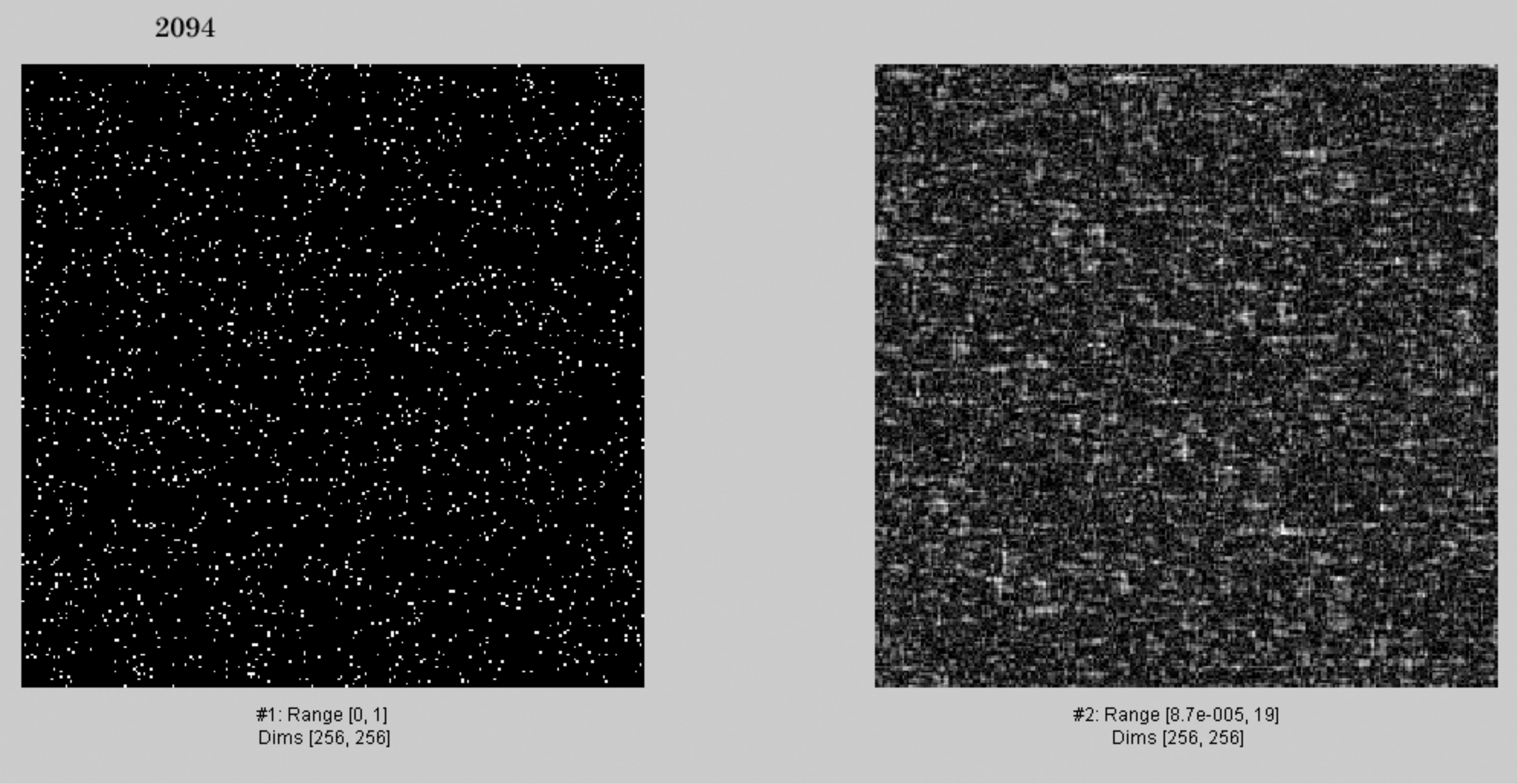
538



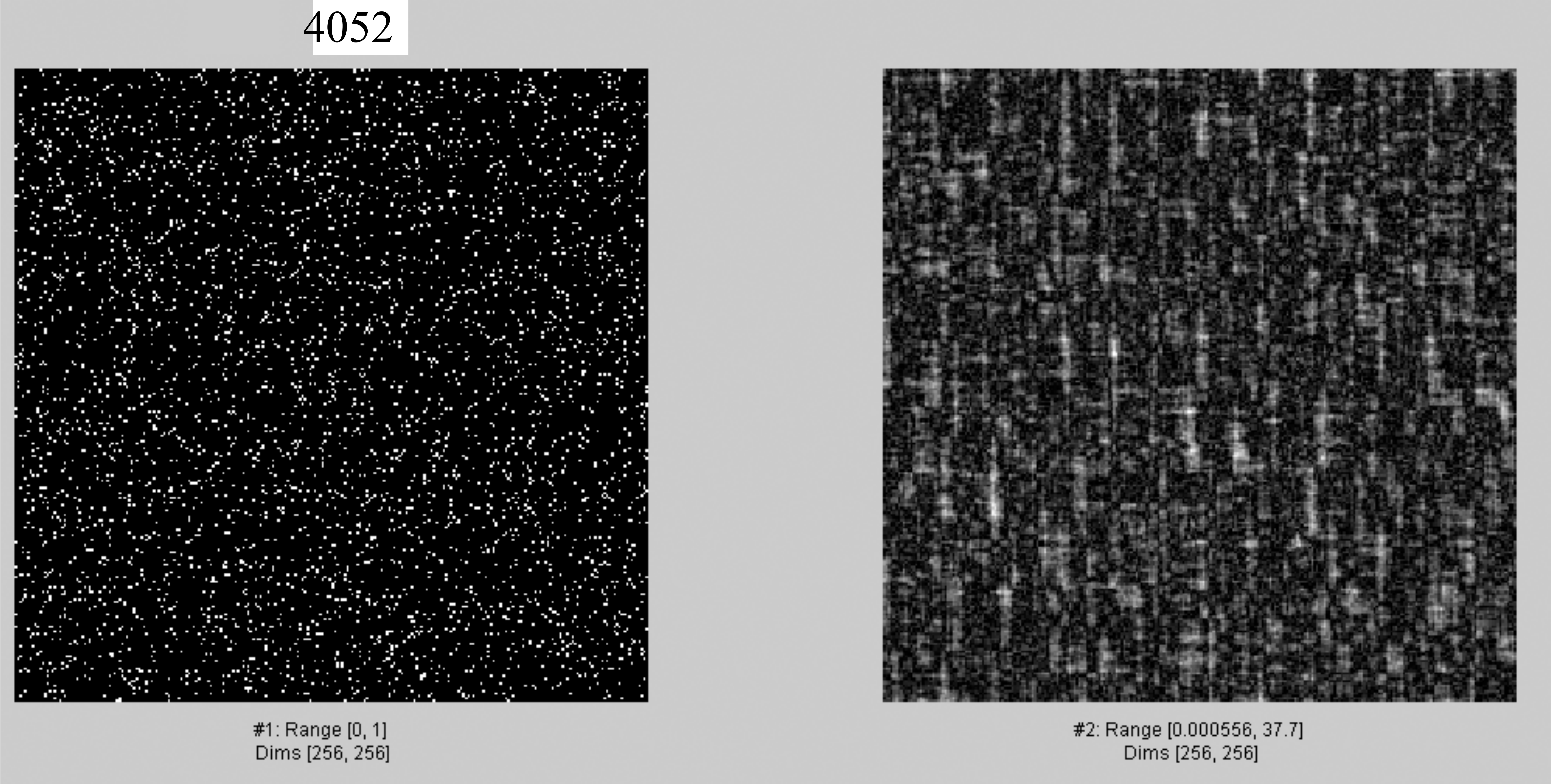
1088



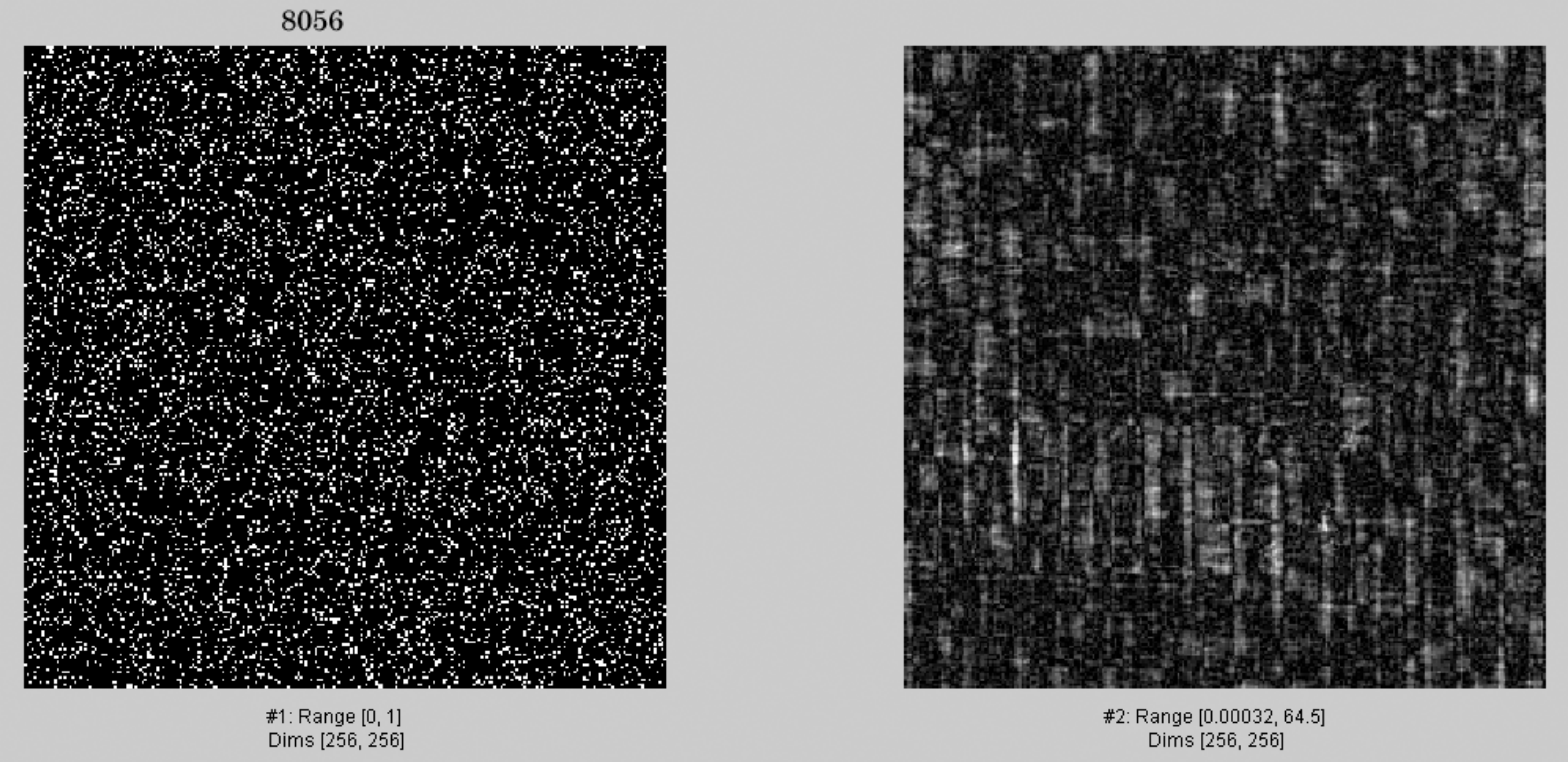
2094



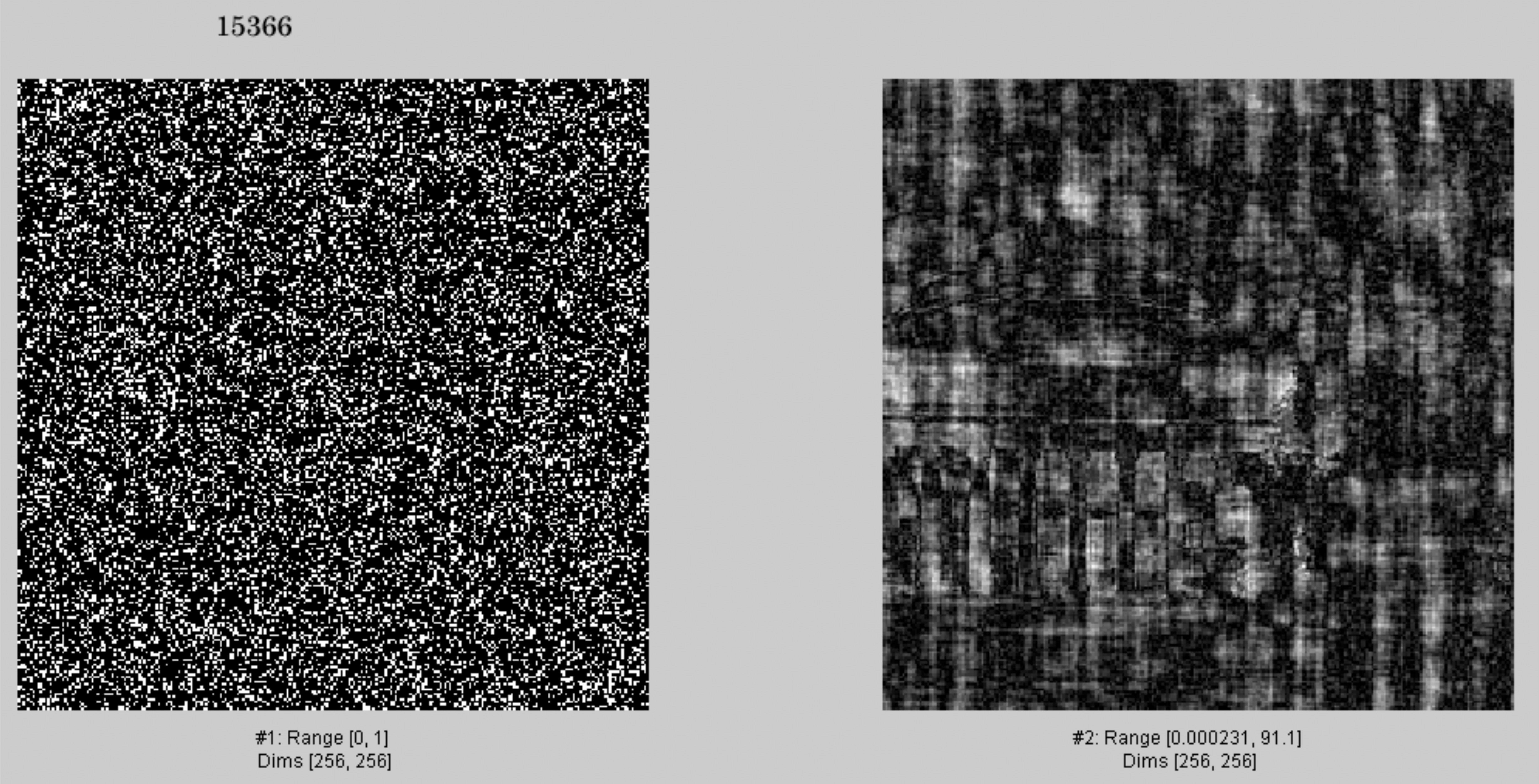
4052.



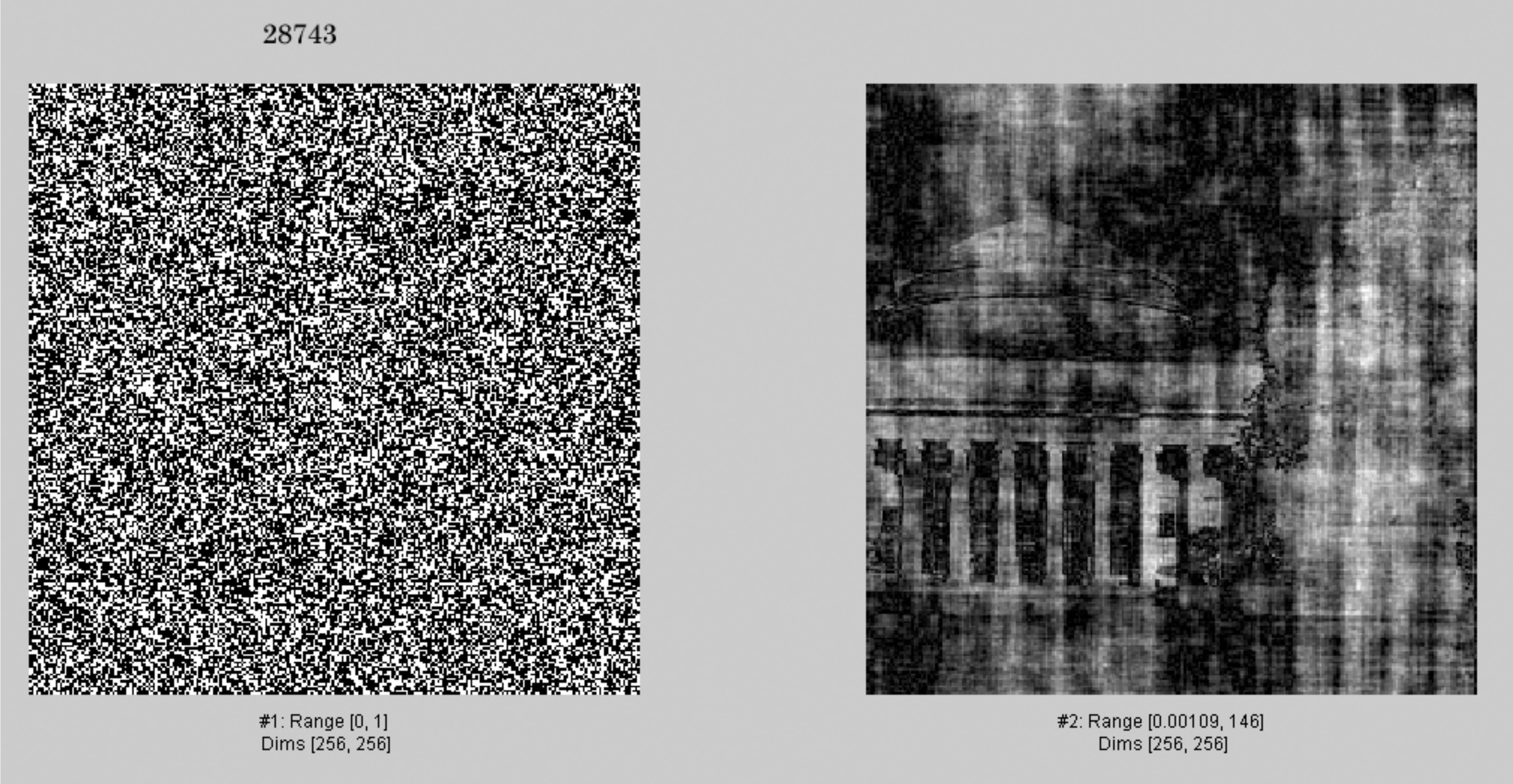
8056.



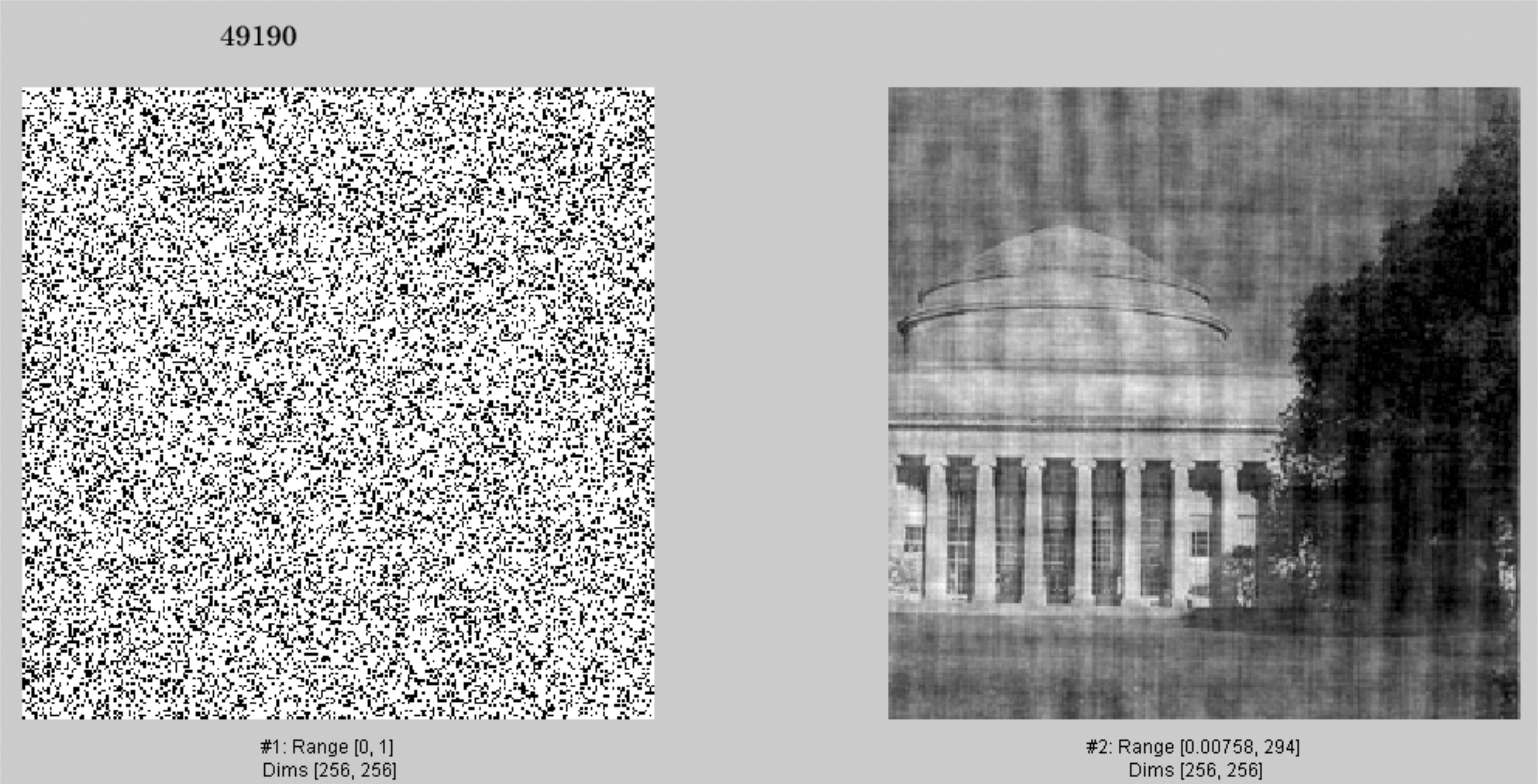
15366



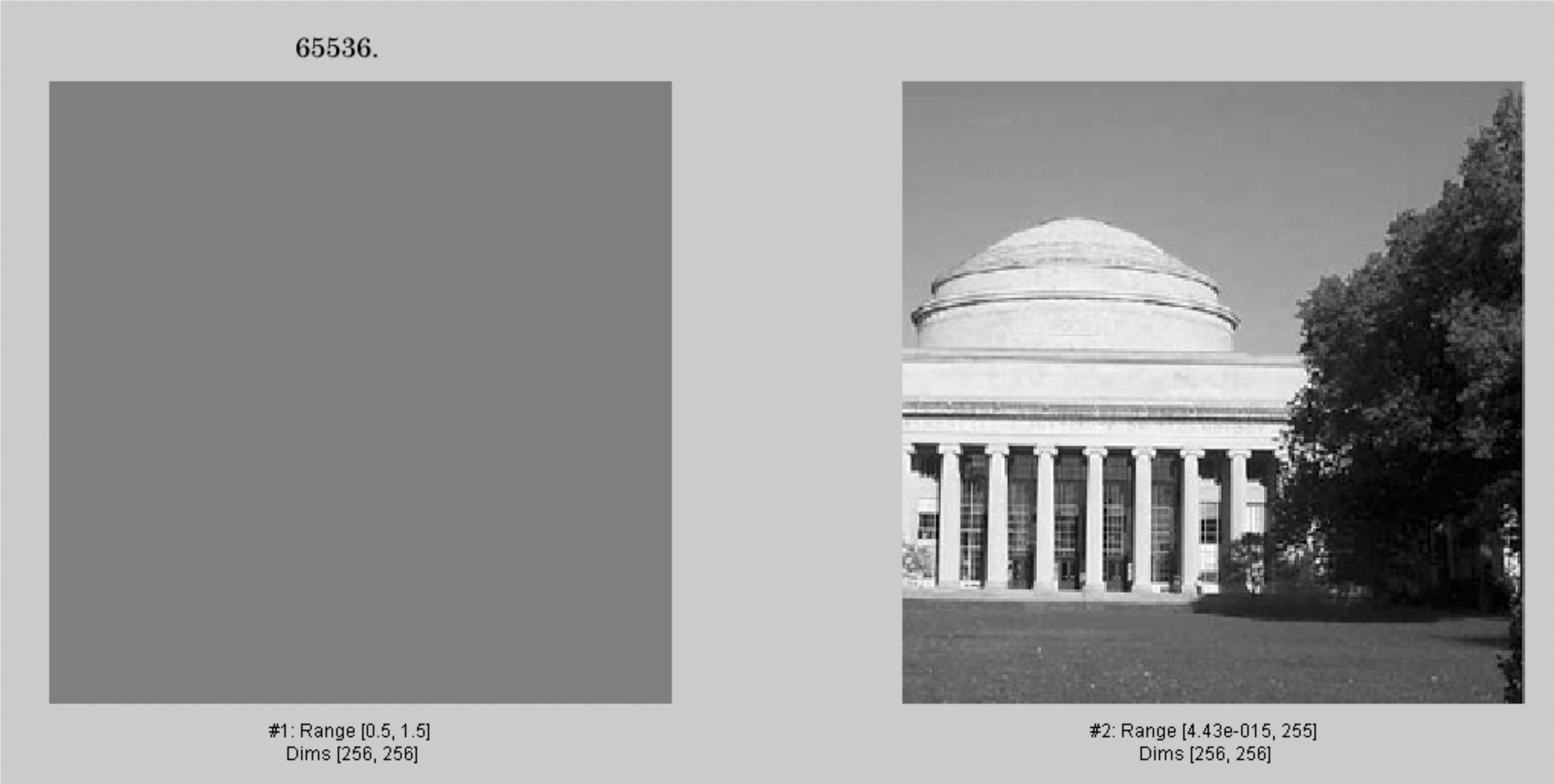
28743



49190.

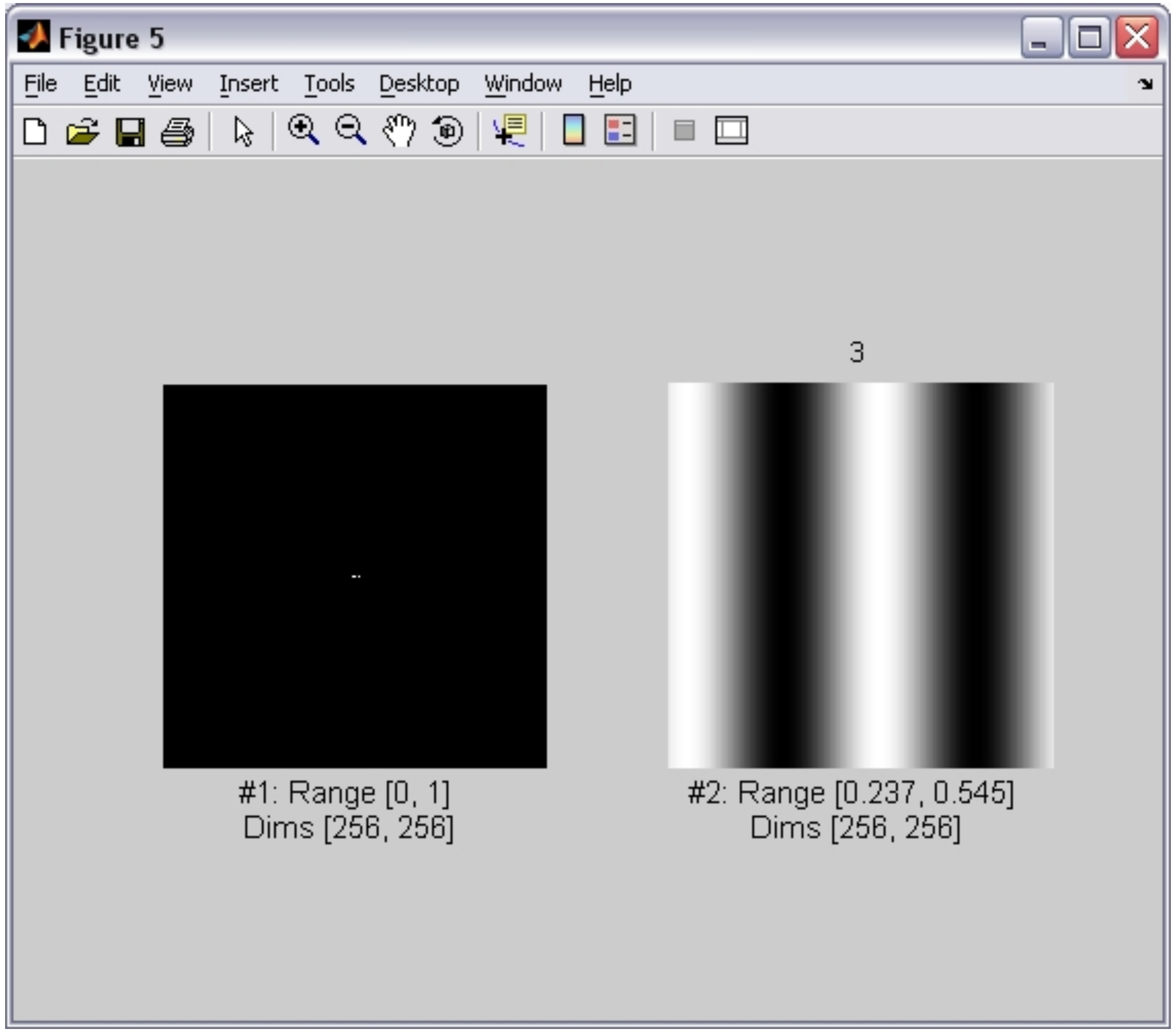


65536.



65536.
#1: Range [0.5, 1.5]
Dims [256, 256]
#2: Range [4.43e-015, 255]
Dims [256, 256]

3



Now, an analogous sequence of images, but selecting Fourier components in descending order of magnitude.

5

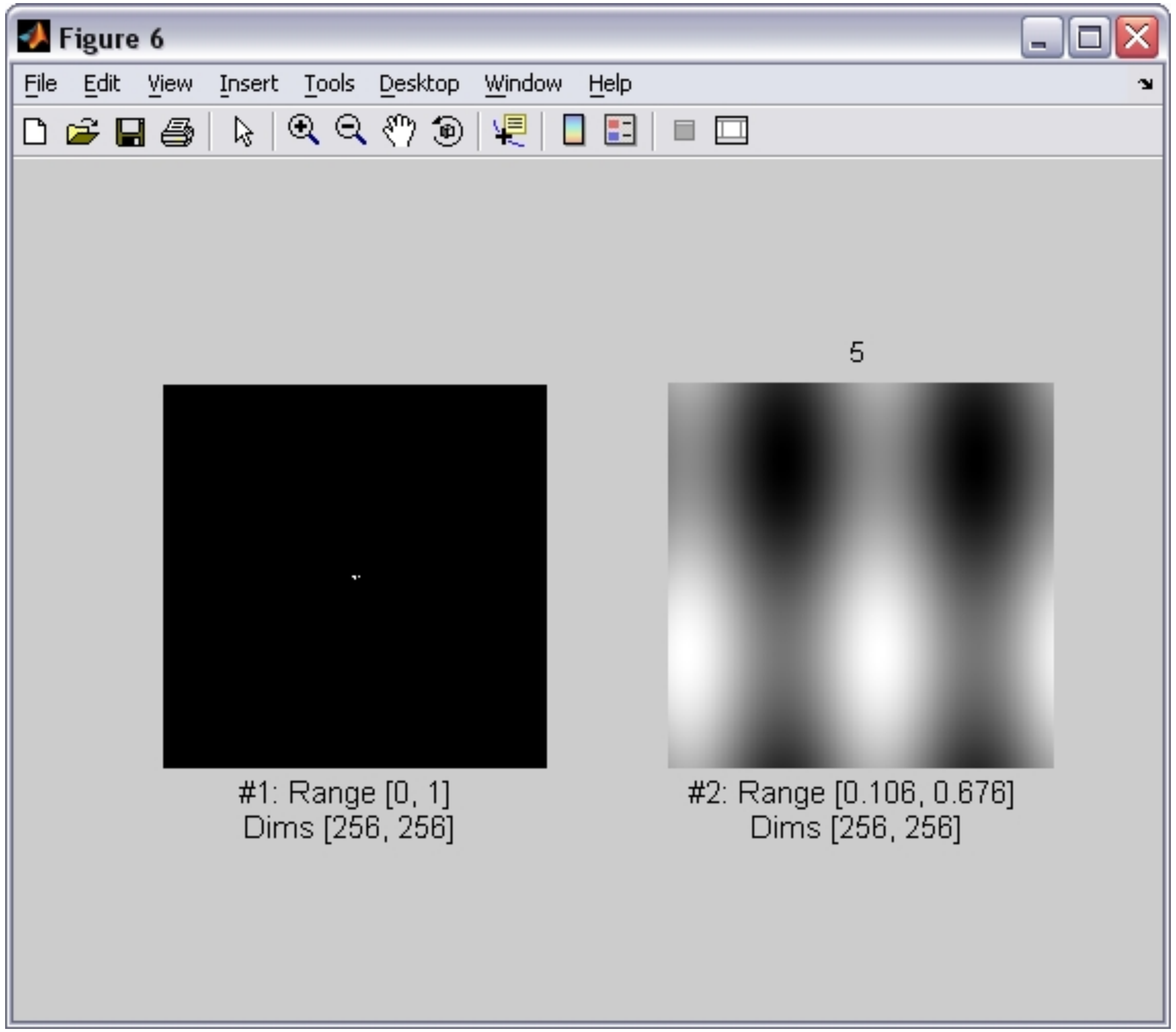


Figure 6
File Edit View Insert Tools Desktop Window Help
5
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.106, 0.676]
Dims [256, 256]

9

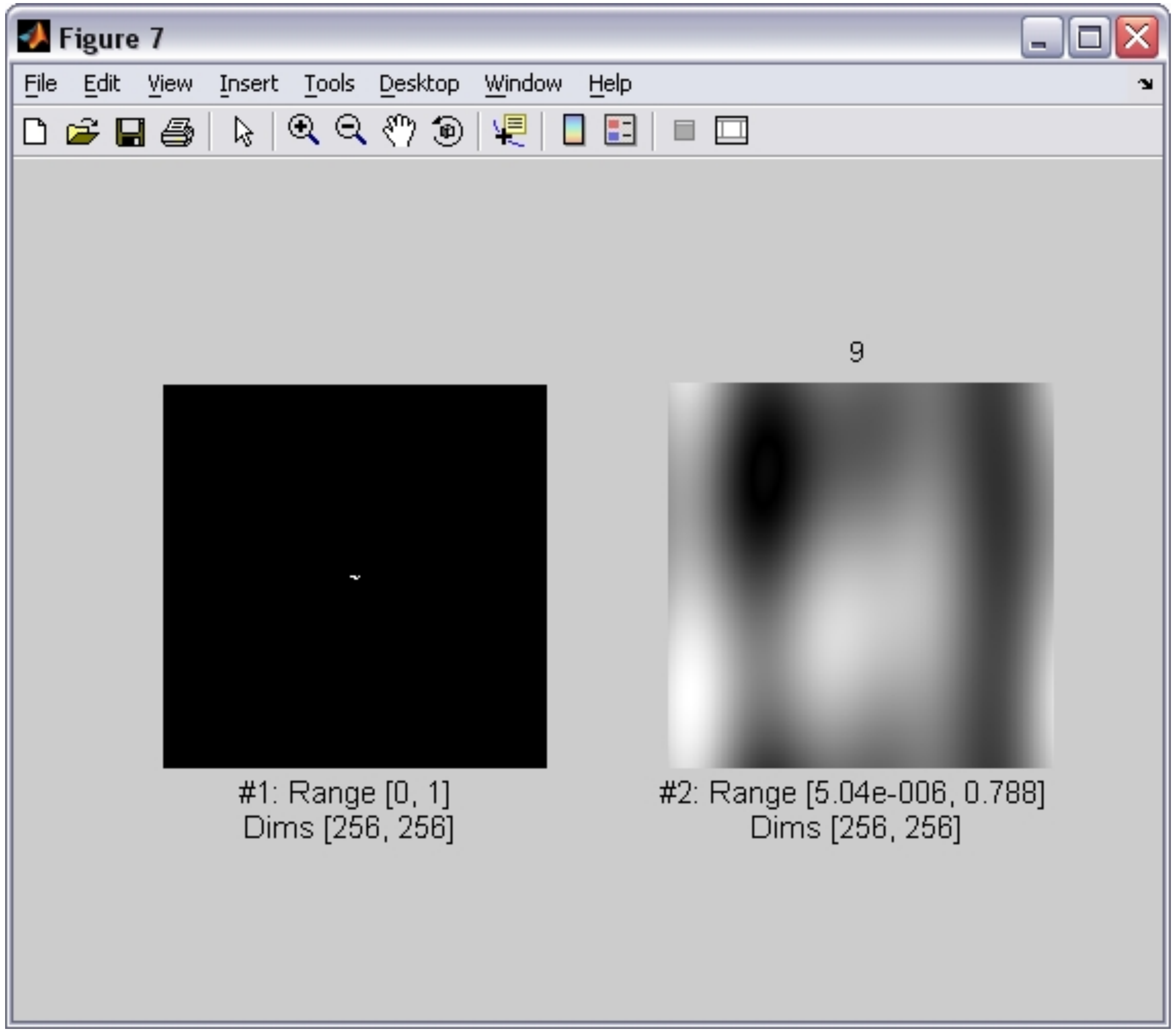


Figure 7
File Edit View Insert Tools Desktop Window Help
9
#1: Range [0, 1]
Dims [256, 256]
#2: Range [5.04e-006, 0.788]
Dims [256, 256]

17

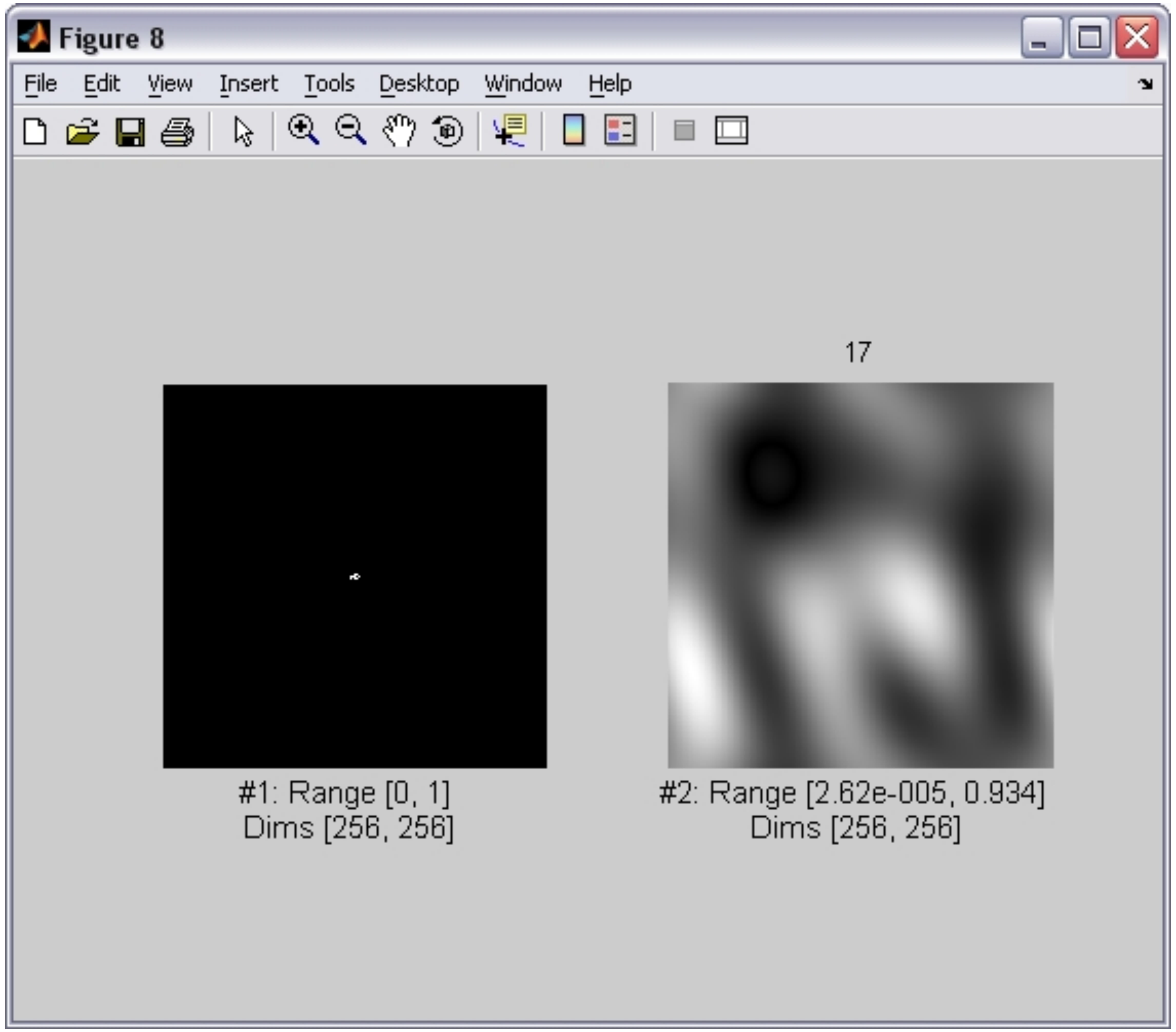


Figure 8
File Edit View Insert Tools Desktop Window Help
17
#1: Range [0, 1]
Dims [256, 256]
#2: Range [2.62e-005, 0.934]
Dims [256, 256]

33

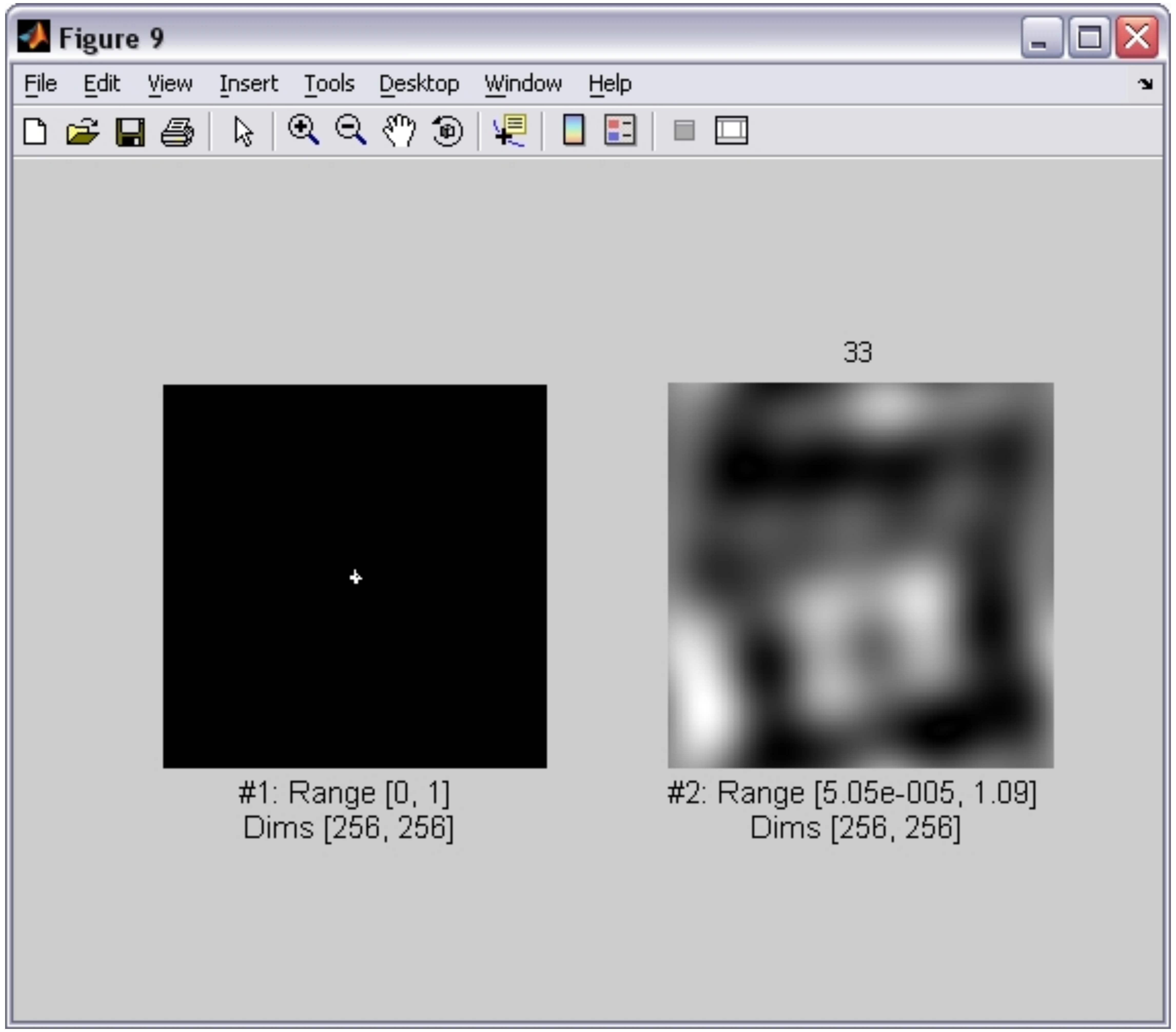


Figure 9
File Edit View Insert Tools Desktop Window Help
33
#1: Range [0, 1]
Dims [256, 256]
#2: Range [5.05e-005, 1.09]
Dims [256, 256]

65

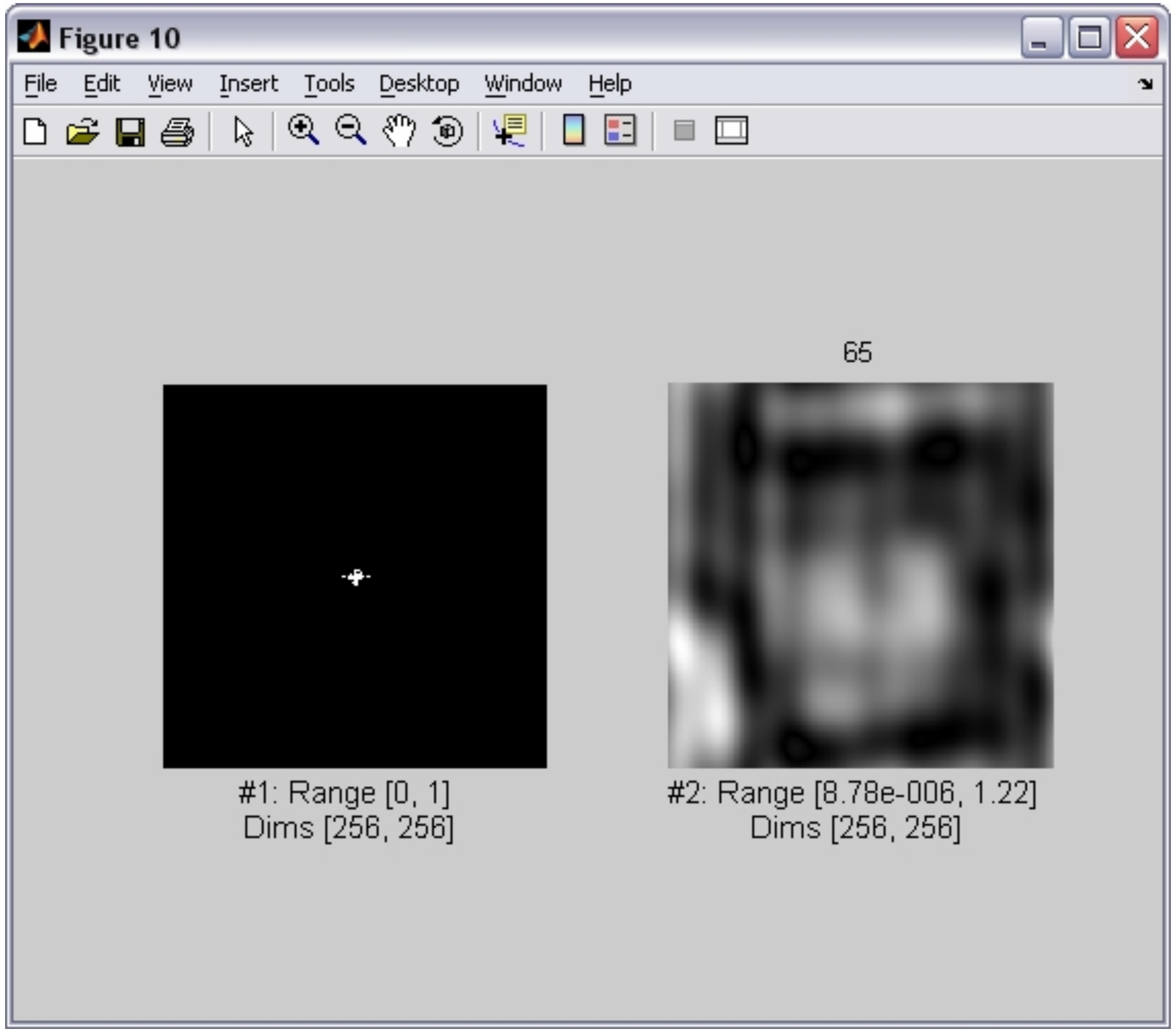


Figure 10
File Edit View Insert Tools Desktop Window Help
65
#1: Range [0, 1]
Dims [256, 256]
#2: Range [8.78e-006, 1.22]
Dims [256, 256]

129

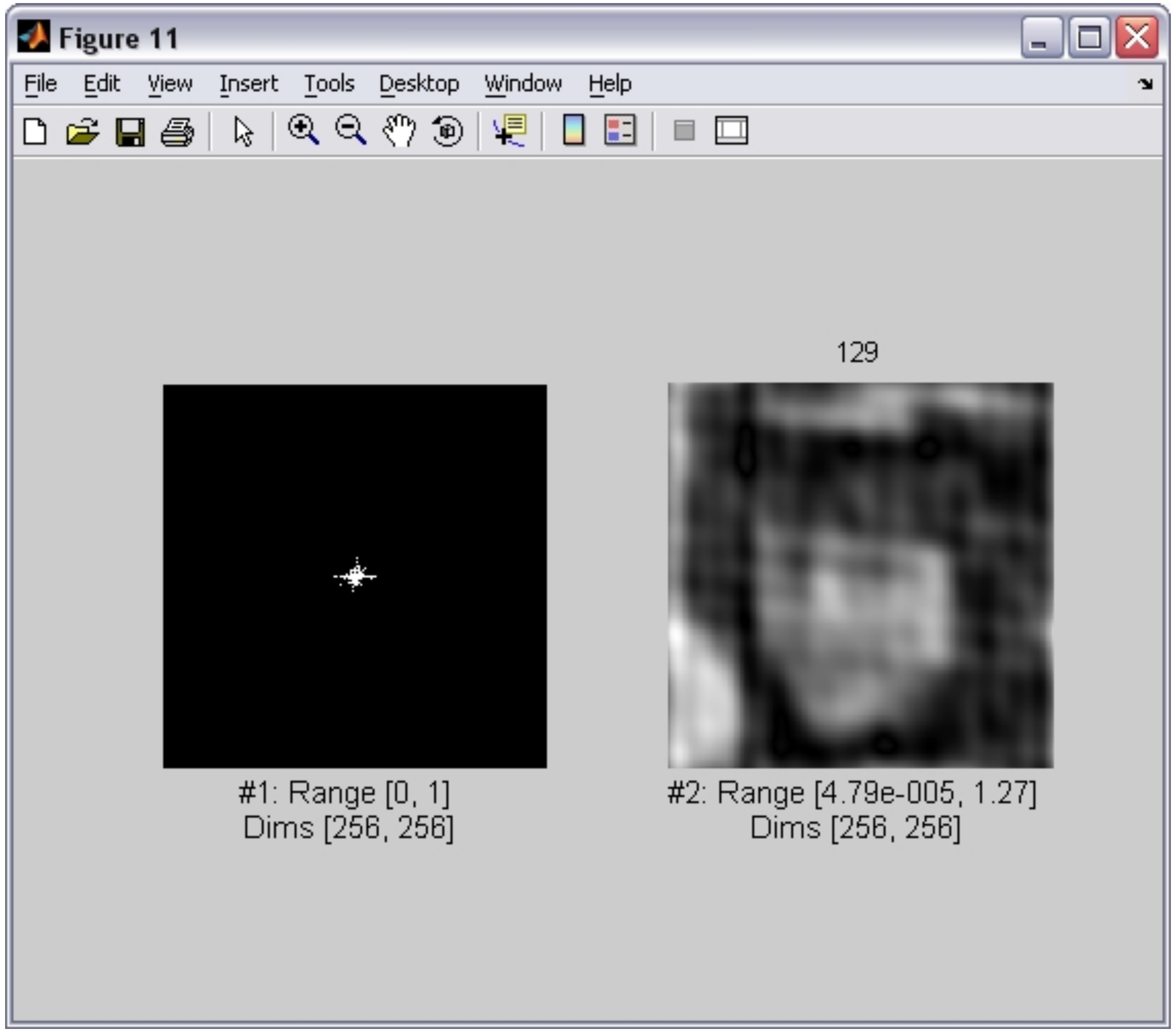


Figure 11
File Edit View Insert Tools Desktop Window Help
129
#1: Range [0, 1]
Dims [256, 256]
#2: Range [4.79e-005, 1.27]
Dims [256, 256]

257

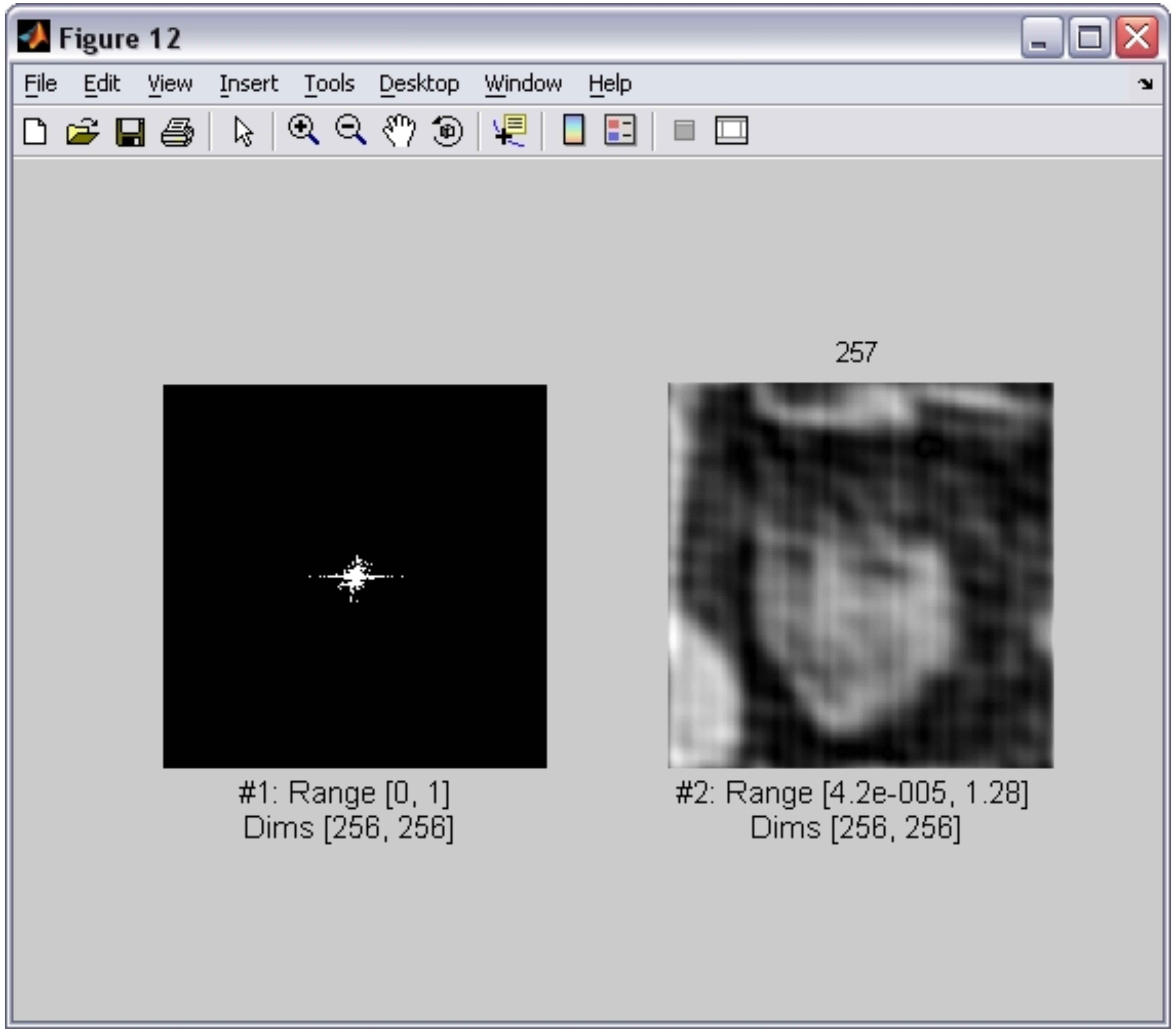


Figure 12
File Edit View Insert Tools Desktop Window Help
257
#1: Range [0, 1]
Dims [256, 256]
#2: Range [4.2e-005, 1.28]
Dims [256, 256]

513

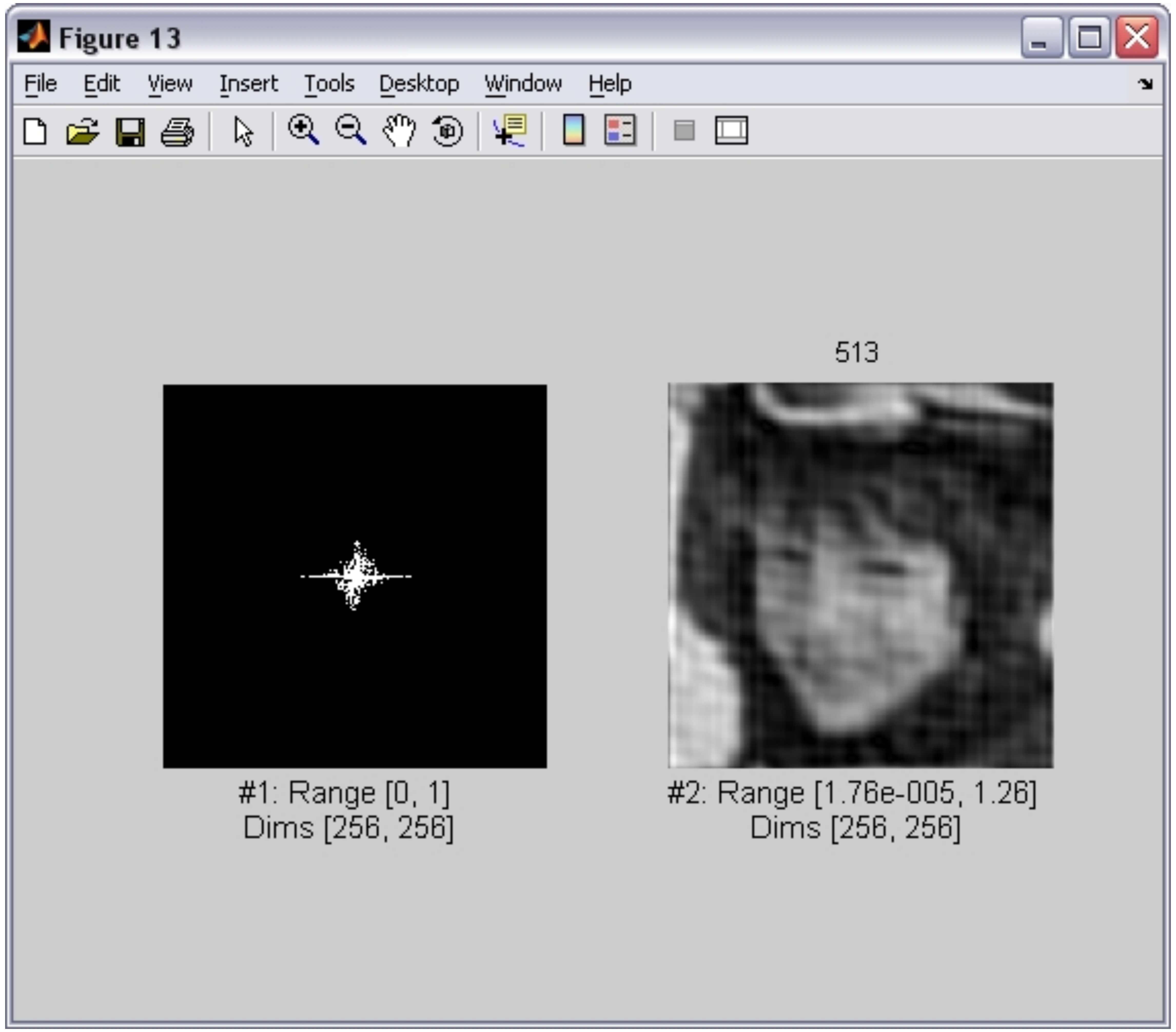


Figure 13
File Edit View Insert Tools Desktop Window Help
513
#1: Range [0, 1]
Dims [256, 256]
#2: Range [1.76e-005, 1.26]
Dims [256, 256]

1025

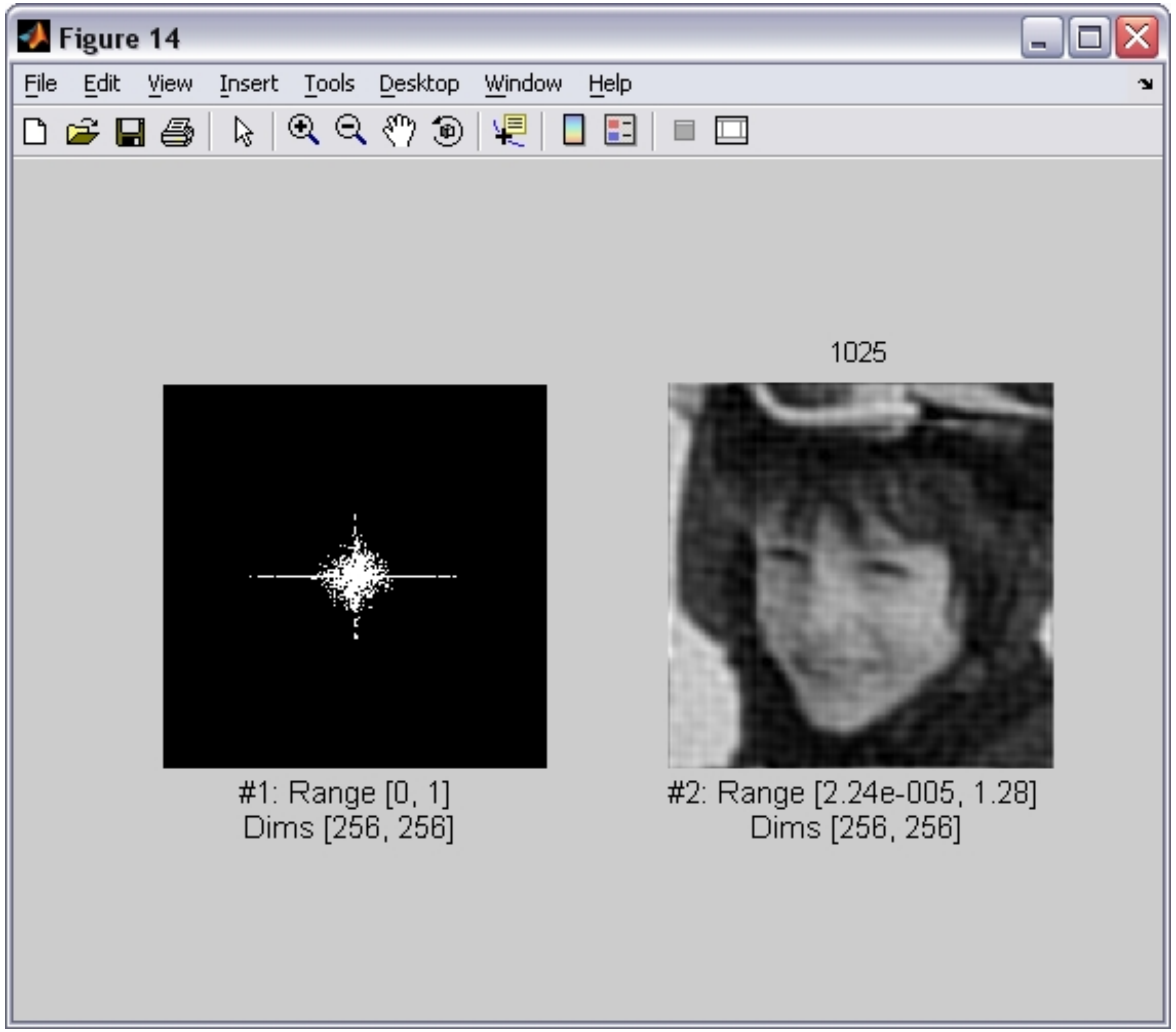


Figure 14
File Edit View Insert Tools Desktop Window Help
1025
#1: Range [0, 1]
Dims [256, 256]
#2: Range [2.24e-005, 1.28]
Dims [256, 256]

2049

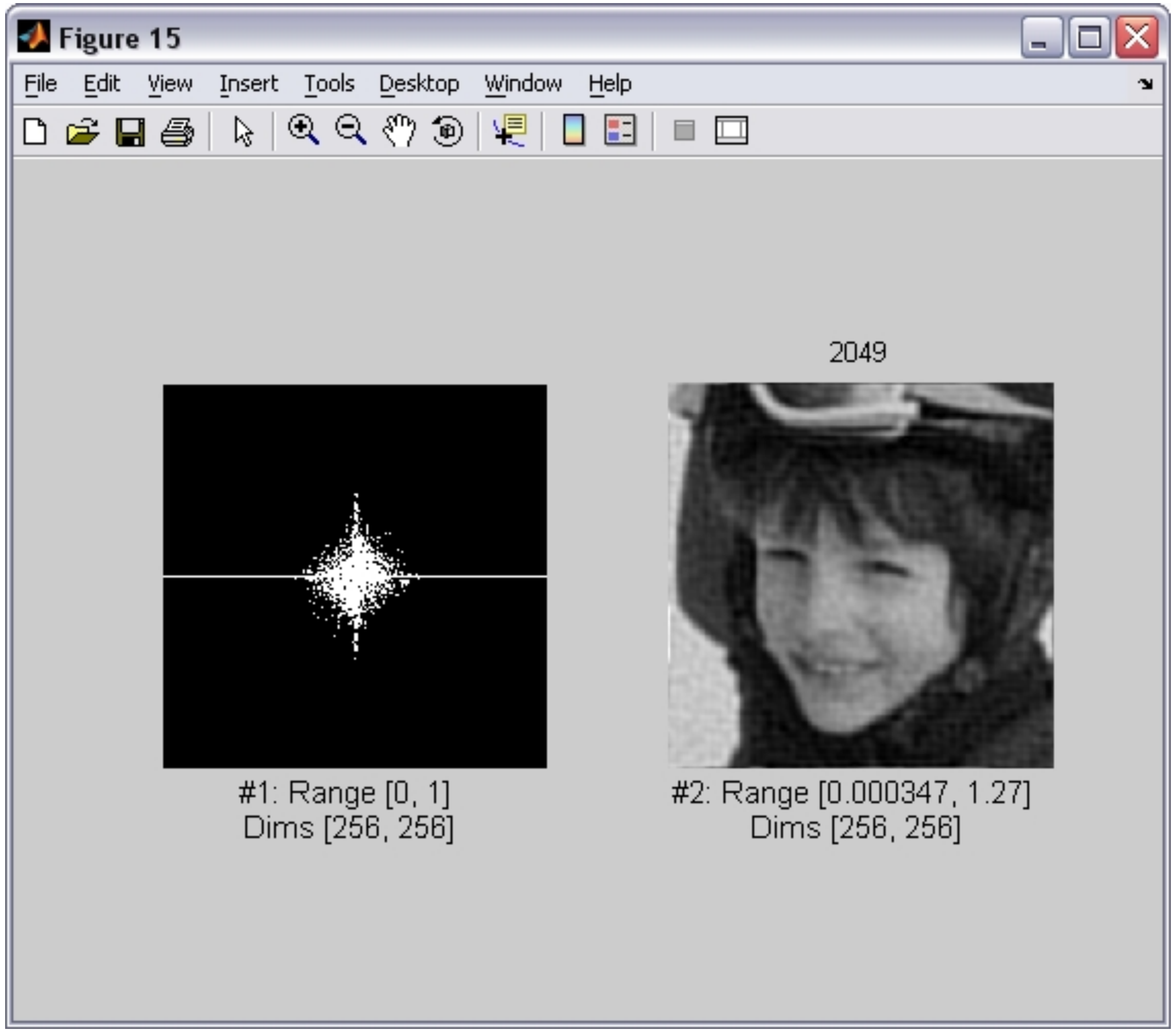


Figure 15
File Edit View Insert Tools Desktop Window Help
2049
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000347, 1.27]
Dims [256, 256]

4097

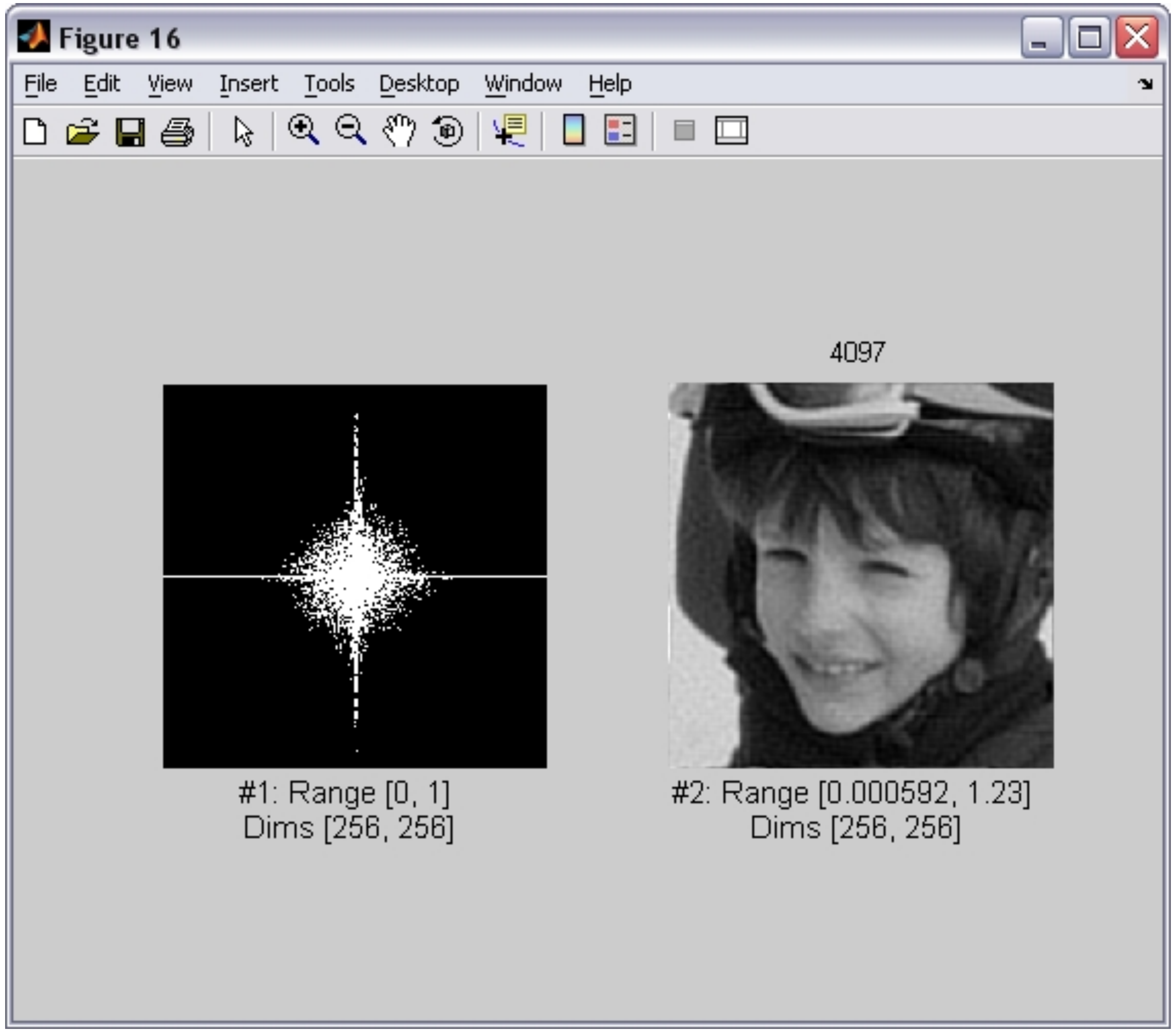


Figure 16
File Edit View Insert Tools Desktop Window Help
4097
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000592, 1.23]
Dims [256, 256]

8193

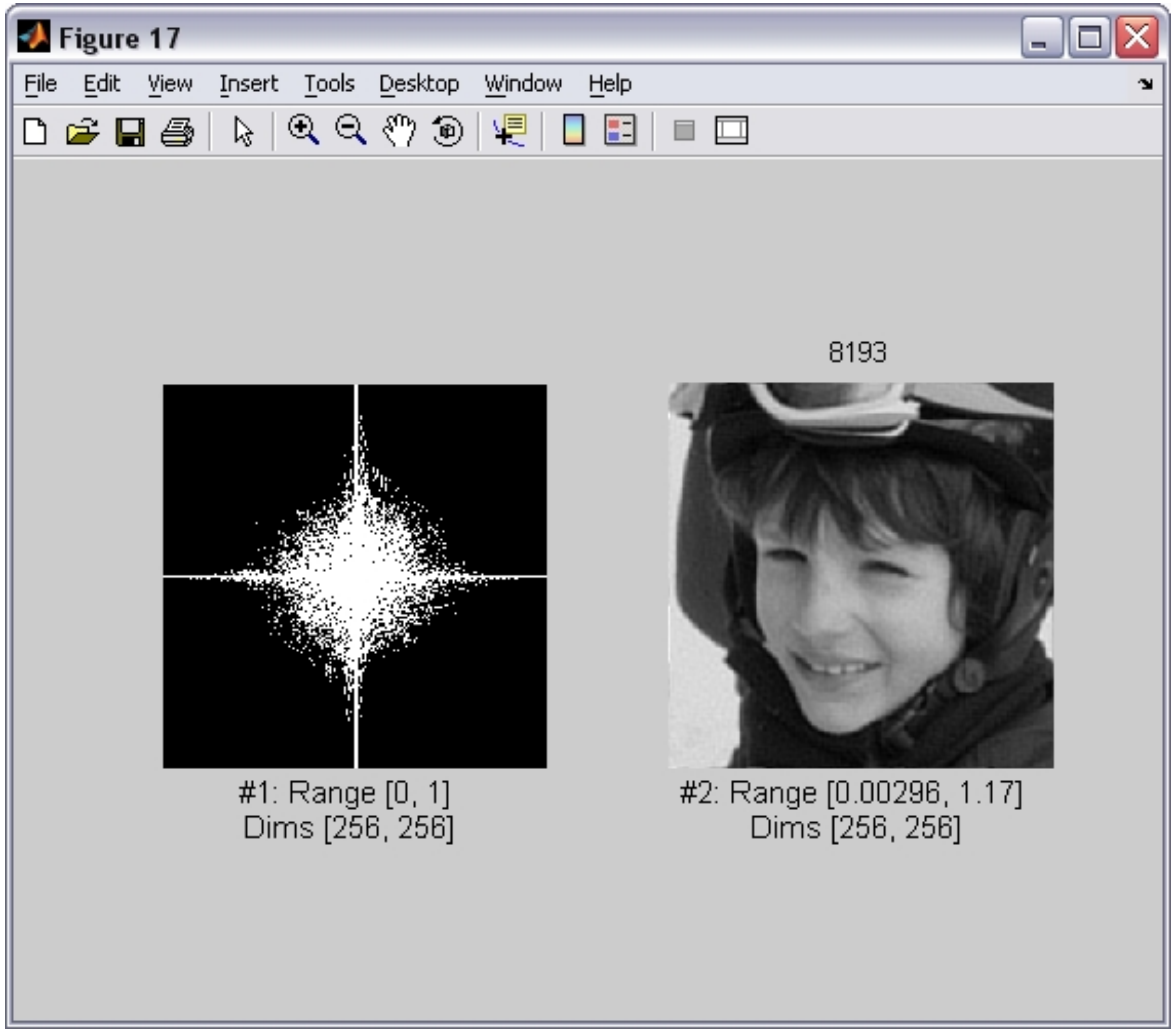


Figure 17
File Edit View Insert Tools Desktop Window Help
8193
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.00296, 1.17]
Dims [256, 256]

16385

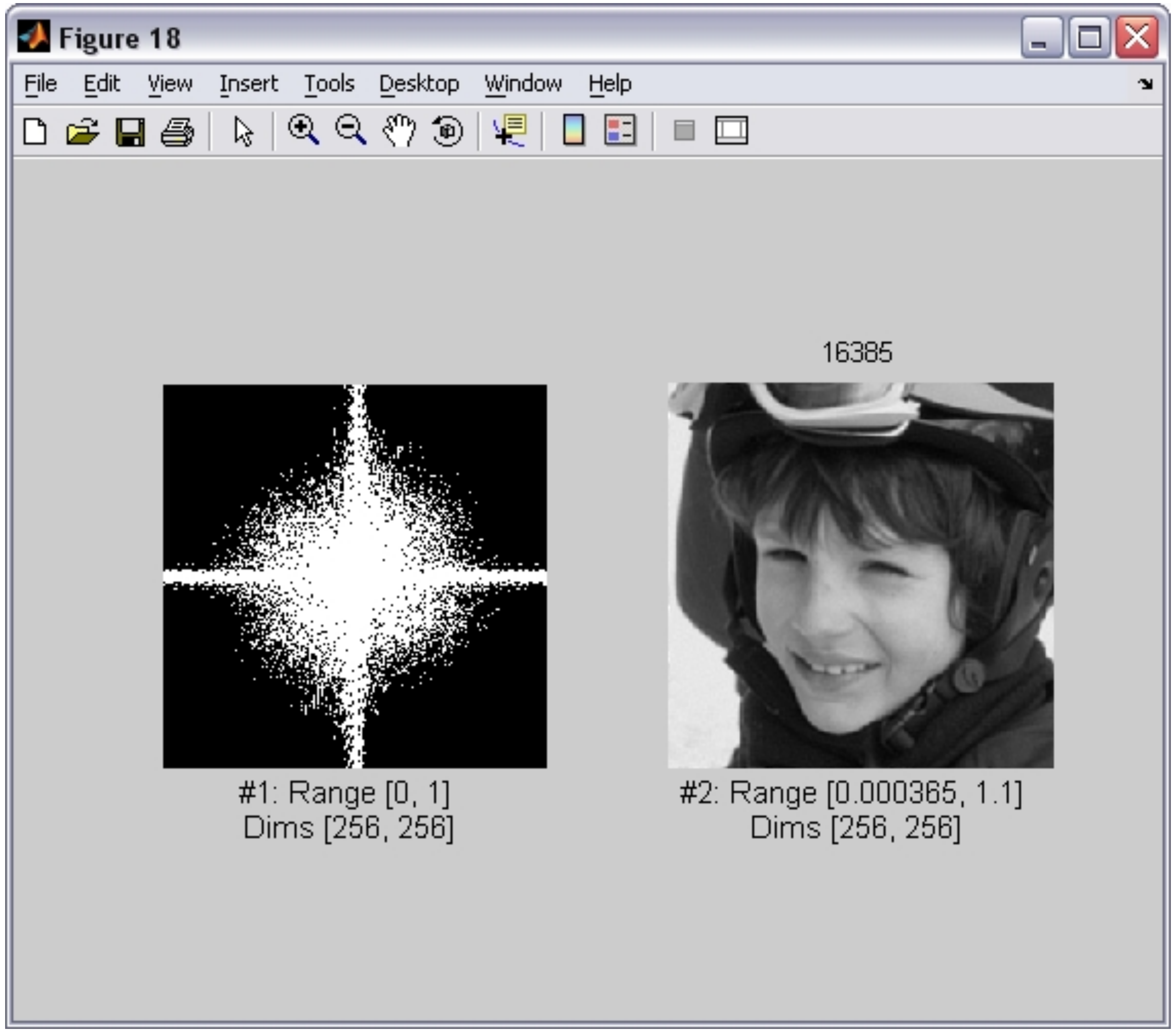


Figure 18
File Edit View Insert Tools Desktop Window Help
16385
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000365, 1.1]
Dims [256, 256]

32769

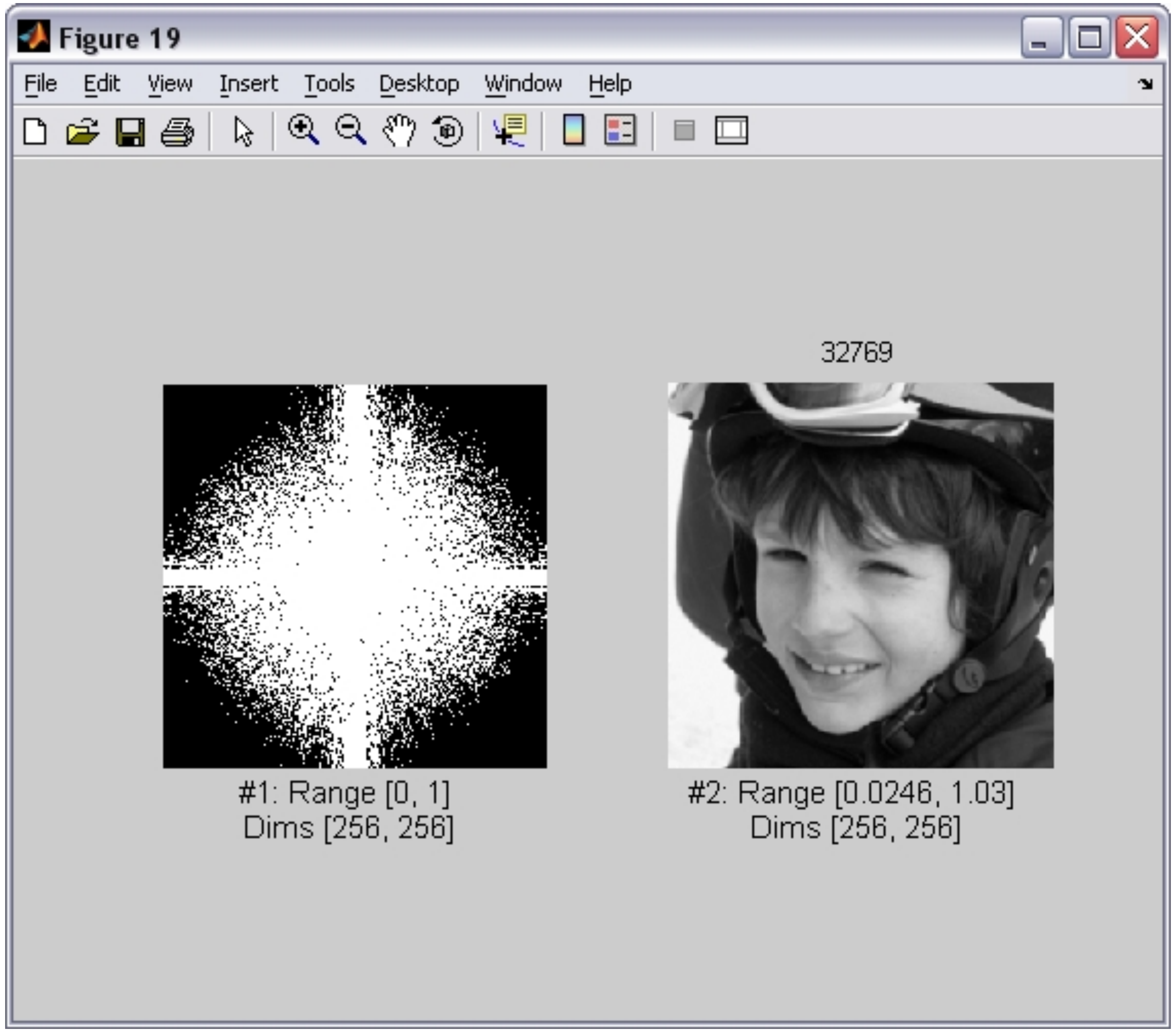


Figure 19
File Edit View Insert Tools Desktop Window Help
32769
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.0246, 1.03]
Dims [256, 256]

65536

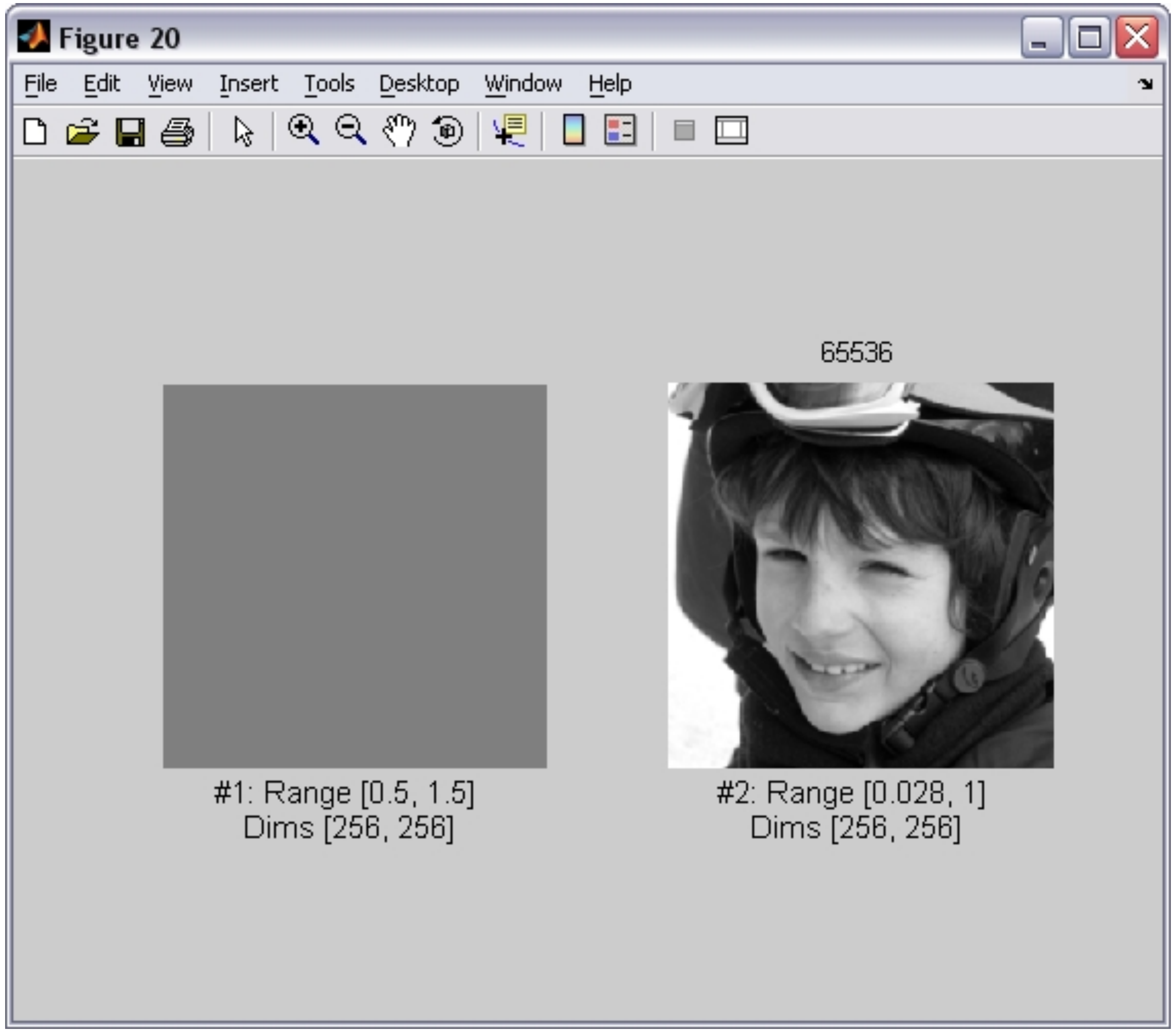
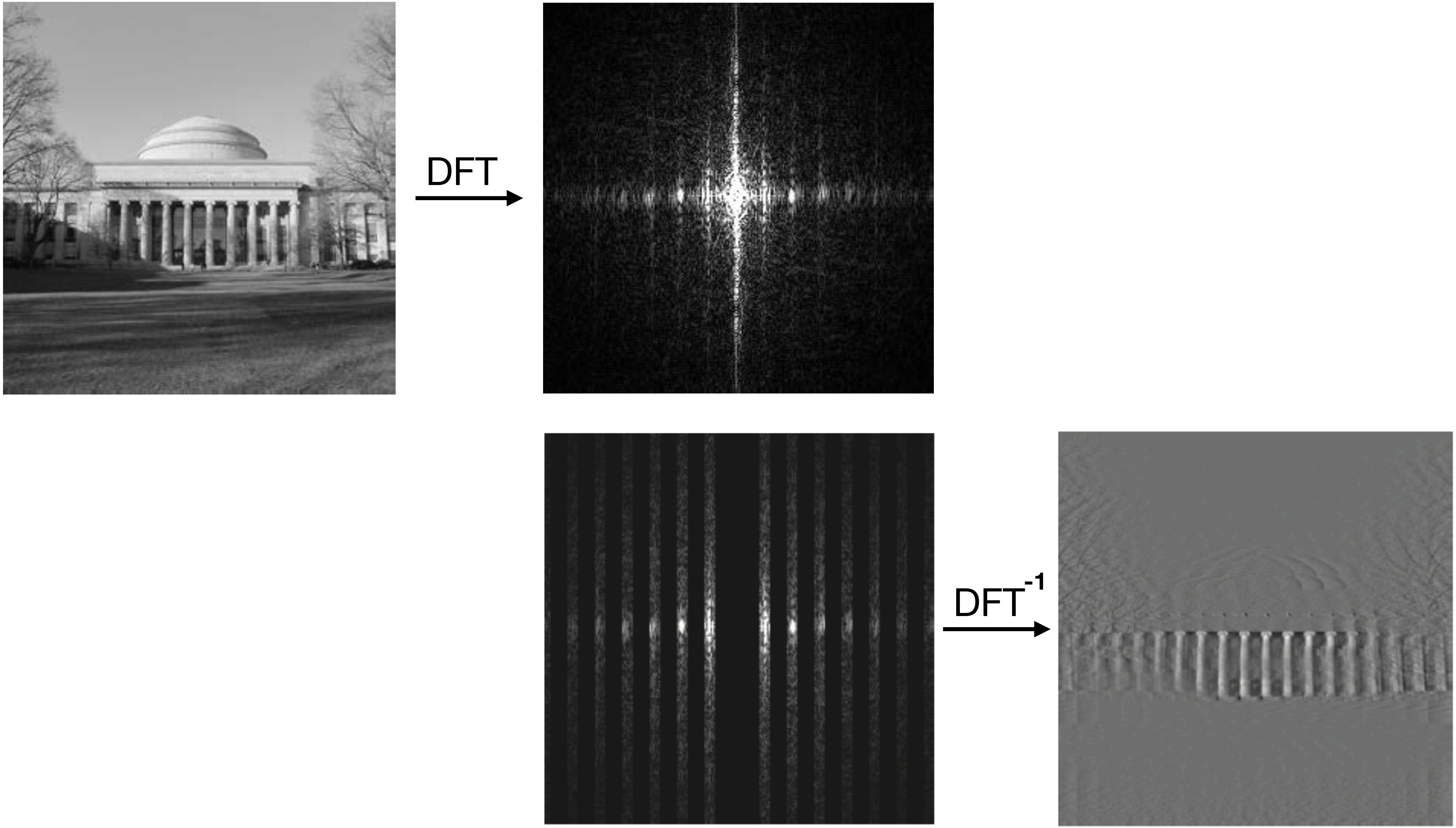
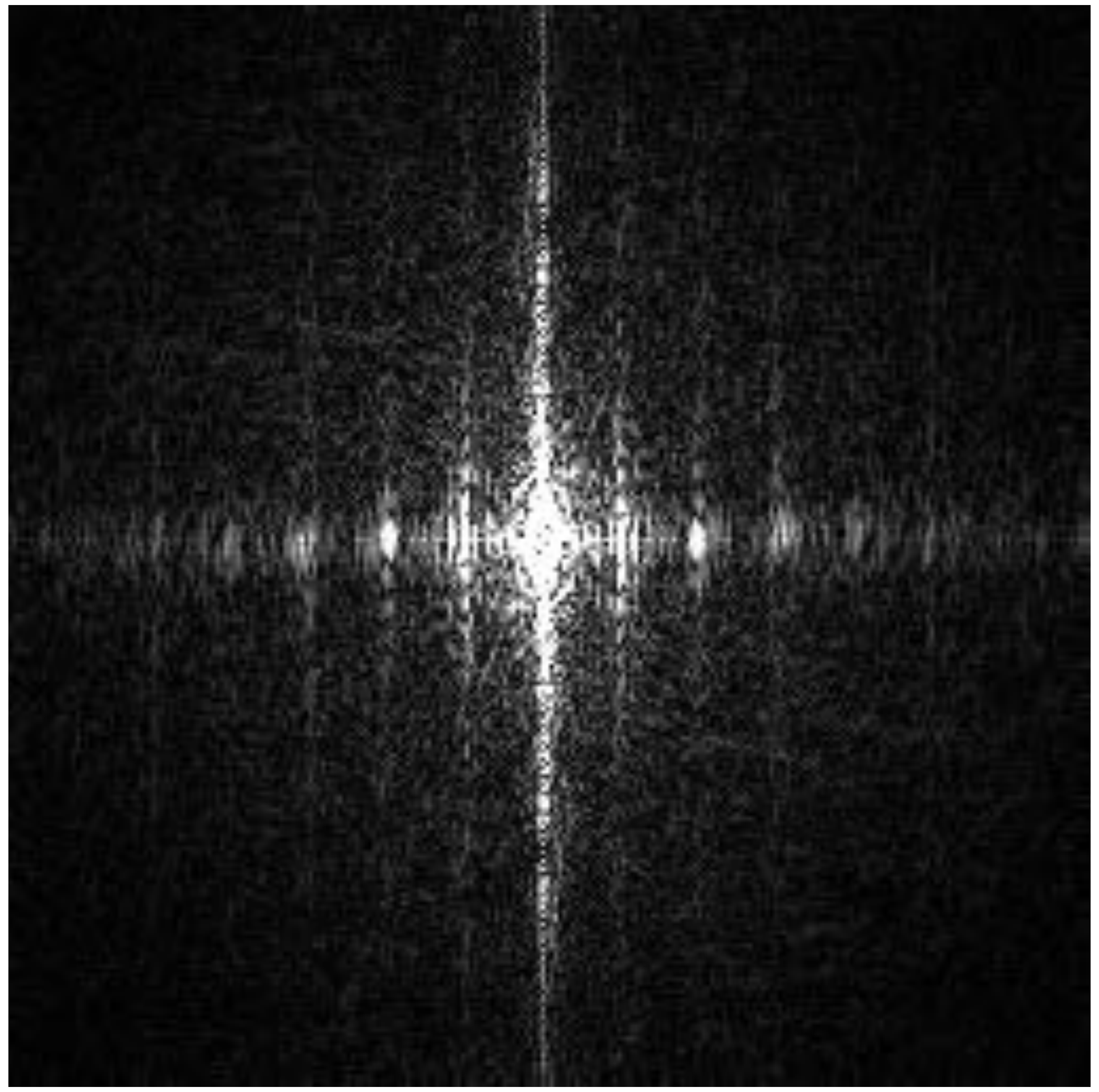
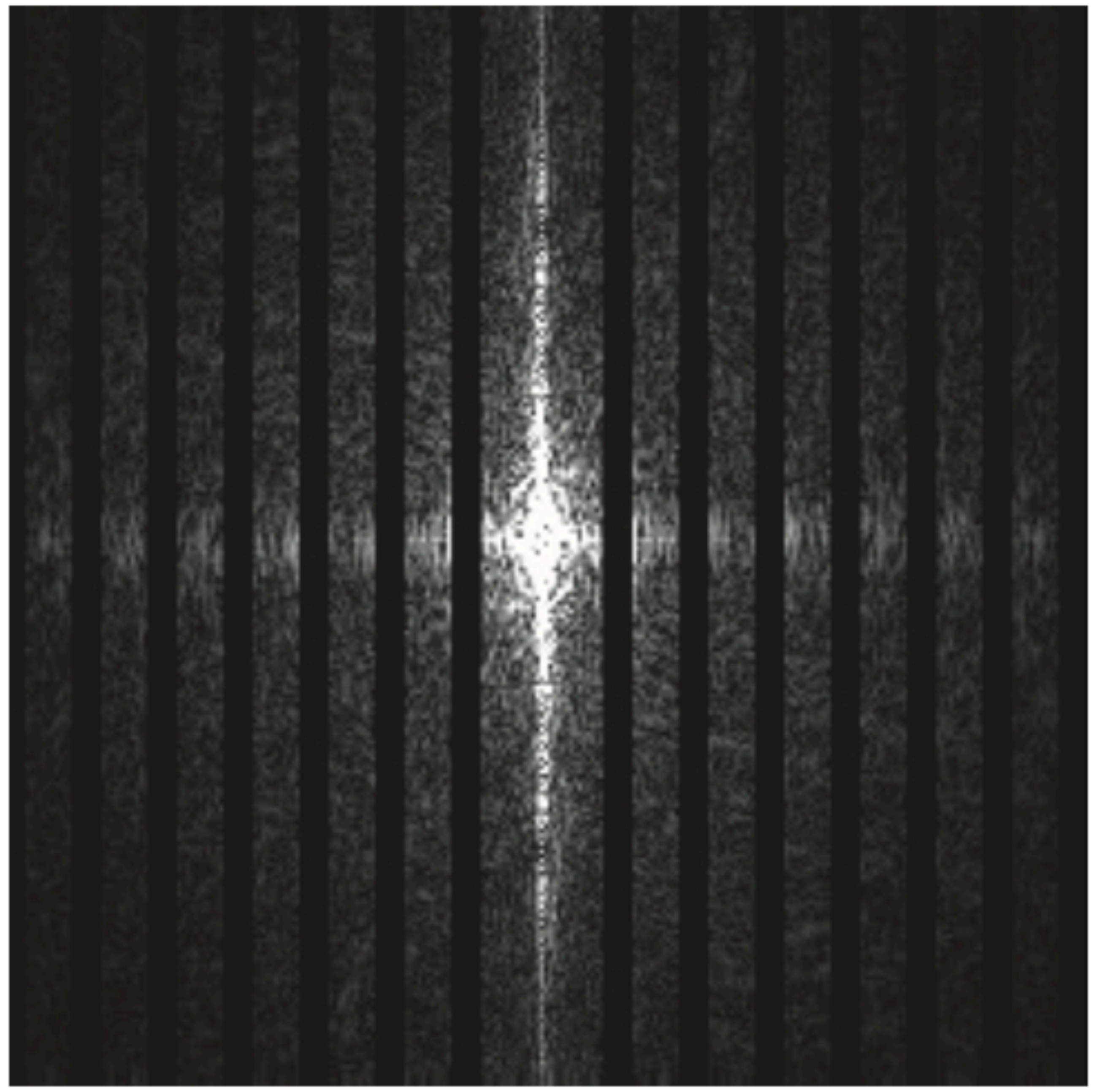


Figure 20
File Edit View Insert Tools Desktop Window Help
65536
#1: Range [0.5, 1.5]
Dims [256, 256]
#2: Range [0.028, 1]
Dims [256, 256]


DFT
DFT-1

DFT →





DFT^{-1} →