Introduction to Computer Vision

Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

Lecture 4

Fourier Transforms & Images

Learning outcomes:

- Image downsampling
- Aliasing
- Gaussian image pyramid
- Laplacian image pyramid
- Fourier series
- Frequency domain
- Fourier transform
- Frequency-domain filtering
- Revisiting sampling

Fourier Transforms & Images

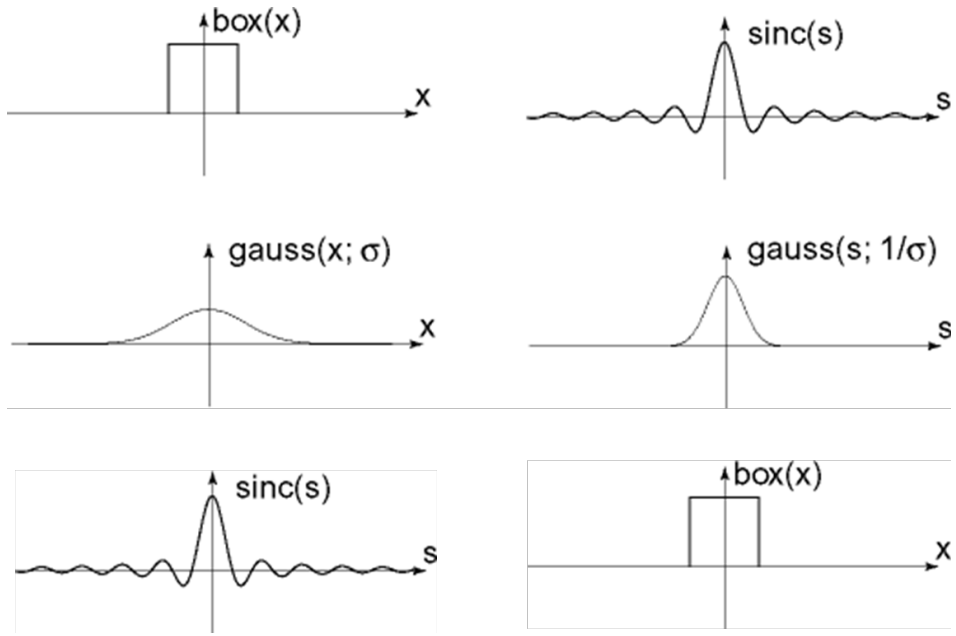
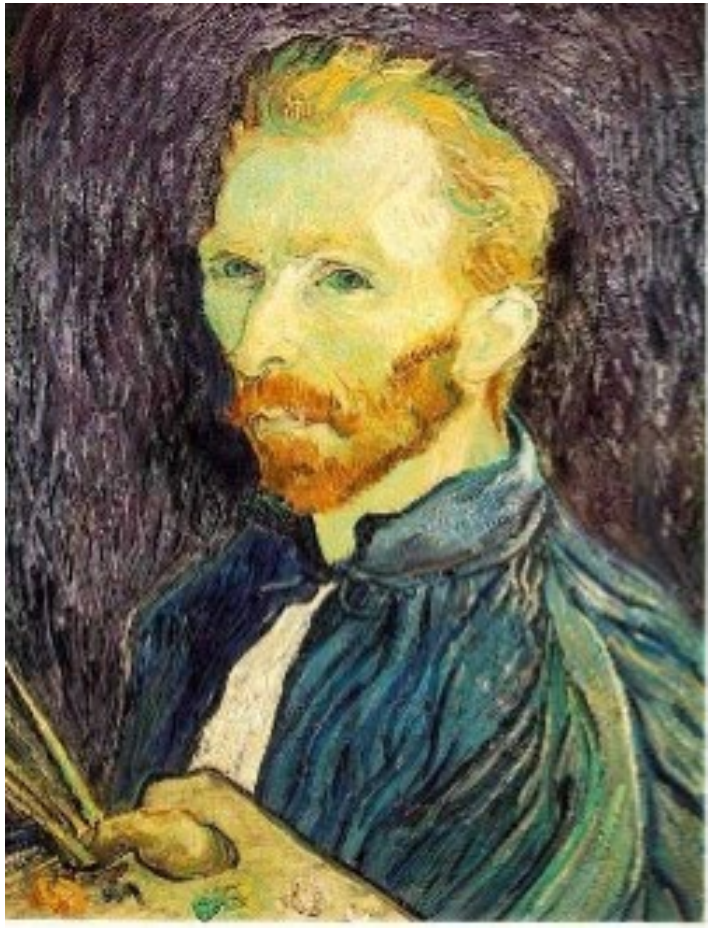


Image downsampling


This image is too big to fit on the screen.
How would you reduce it to half its size?

Naïve image downsampling



1/2

Throw away half the rows and columns

delete even rows
delete even columns

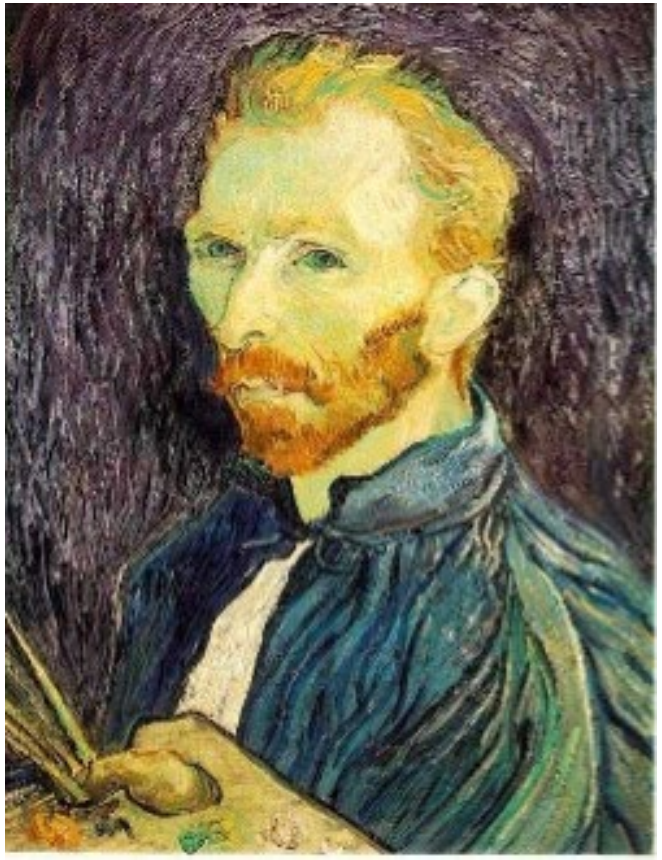
1/4

delete even rows
delete even columns

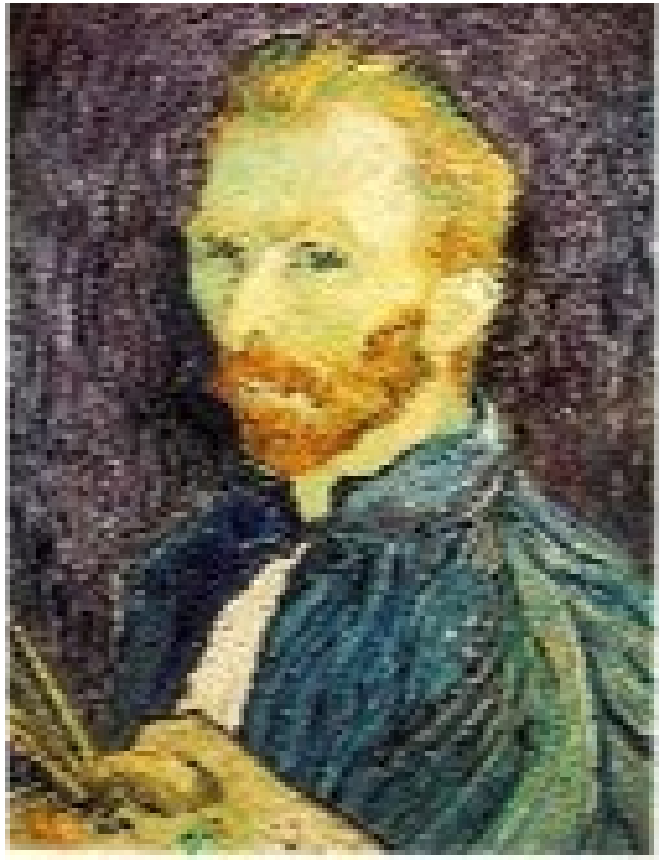
1/8

What is the problem with this approach?

Naïve image downsampling



1/2



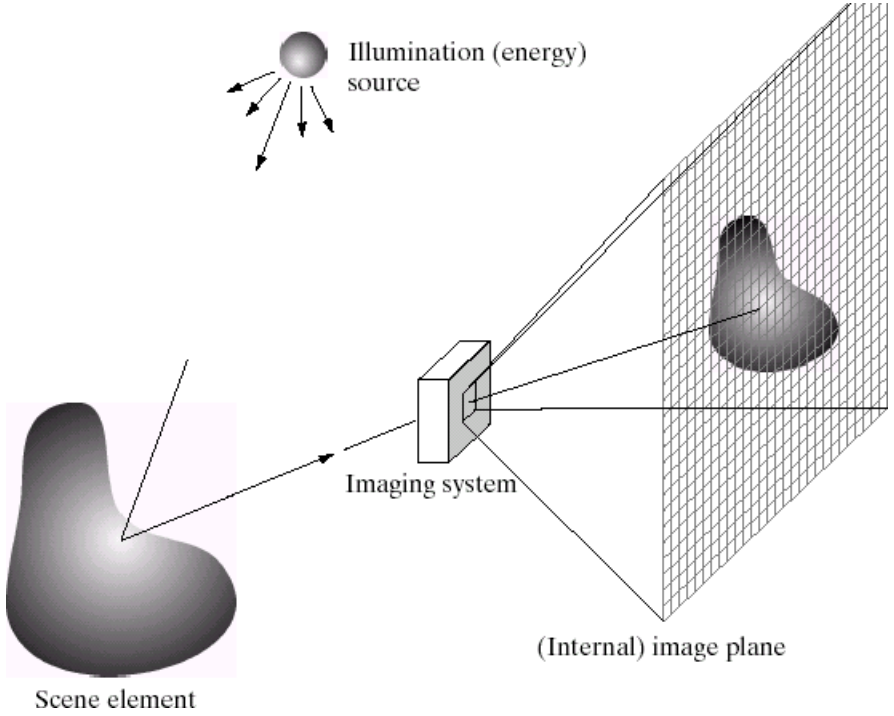
1/4 (2x zoom)

1/8 (4x zoom)

Why is the 1/8 image so pixelated (and do you know what this effect is called)?

Aliasing

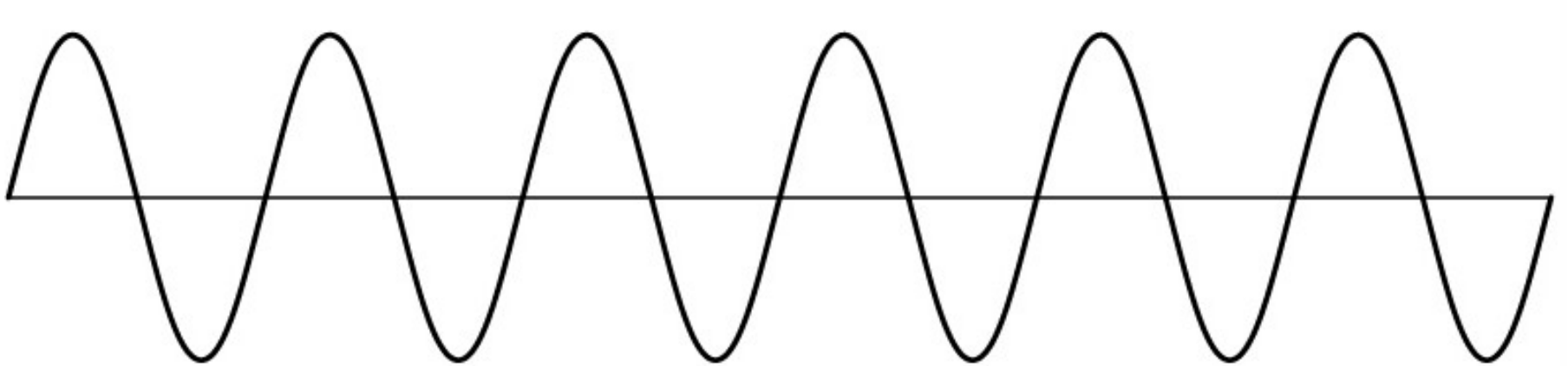
Reminder



Images are a *discrete*, or *sampled*, representation of a *continuous* world

Sampling

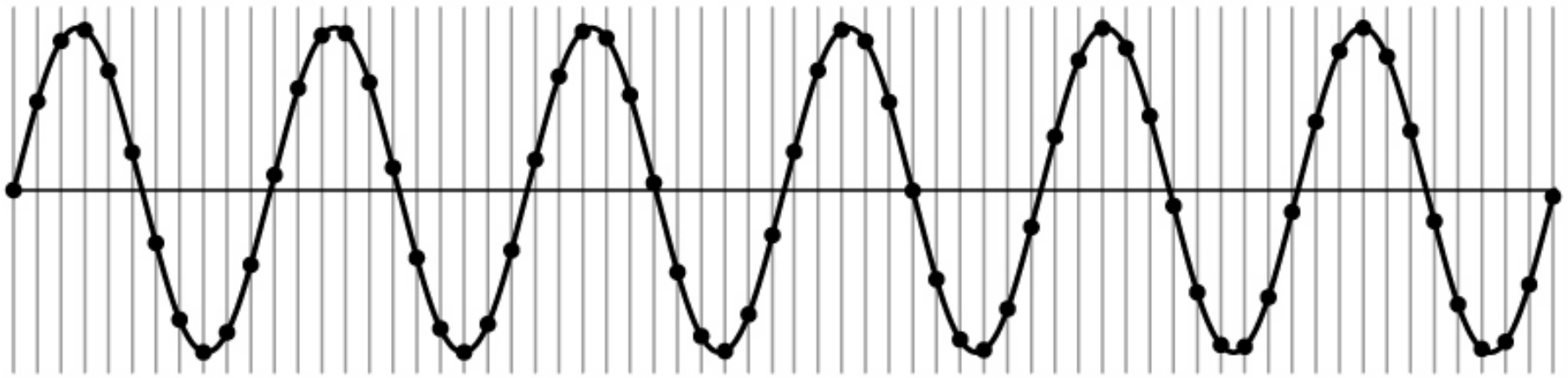
Very simple example: a sine wave



How would you discretize this signal?

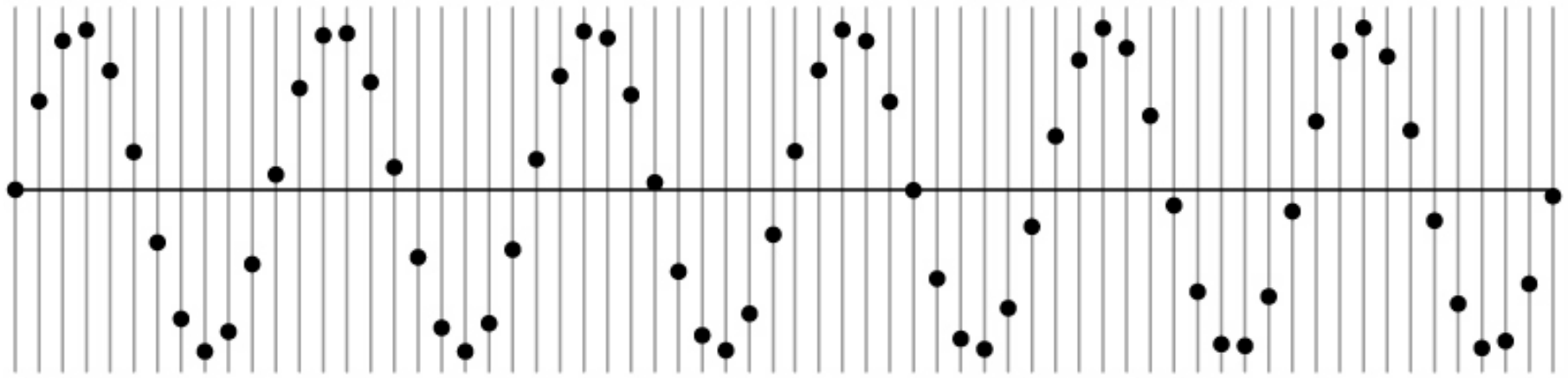
Sampling

Very simple example: a sine wave



Sampling

Very simple example: a sine wave

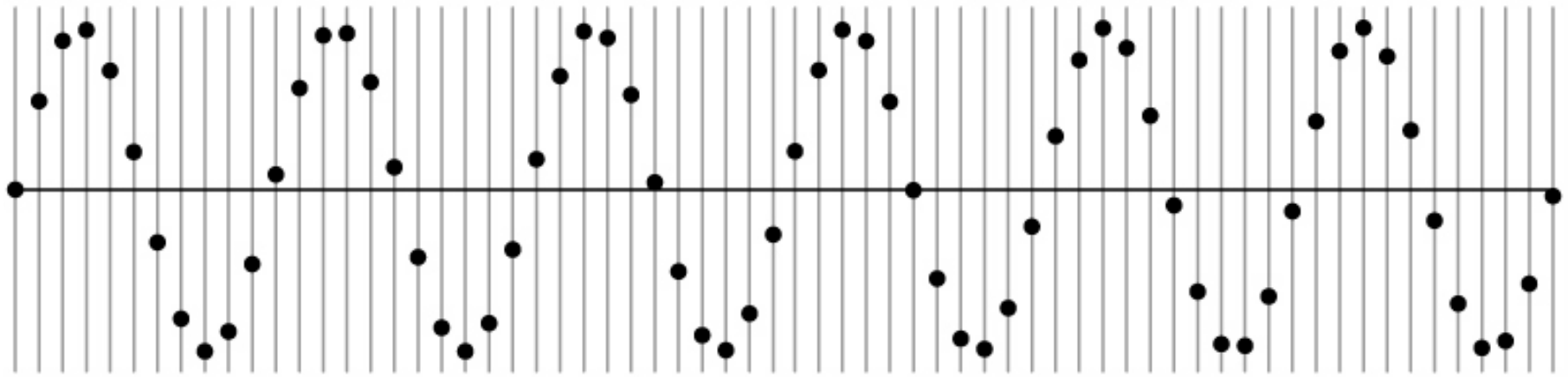


How many samples should I take?

Can I take as ***many*** samples as I want?

Sampling

Very simple example: a sine wave

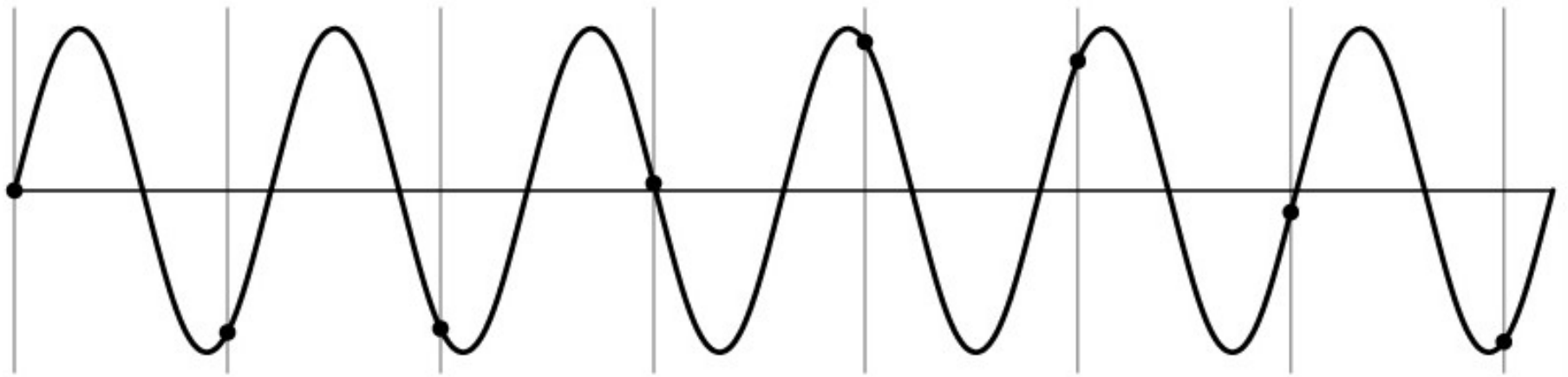


How many samples should I take?

Can I take as ***few*** samples as I want?

Undersampling

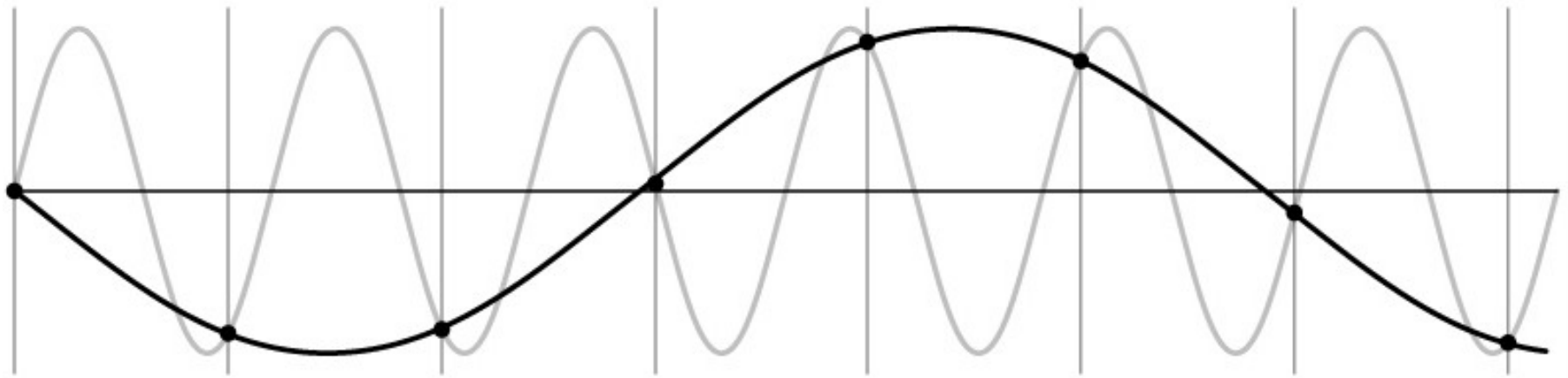
Very simple example: a sine wave



Unsurprising effect: information is lost.

Undersampling

Very simple example: a sine wave

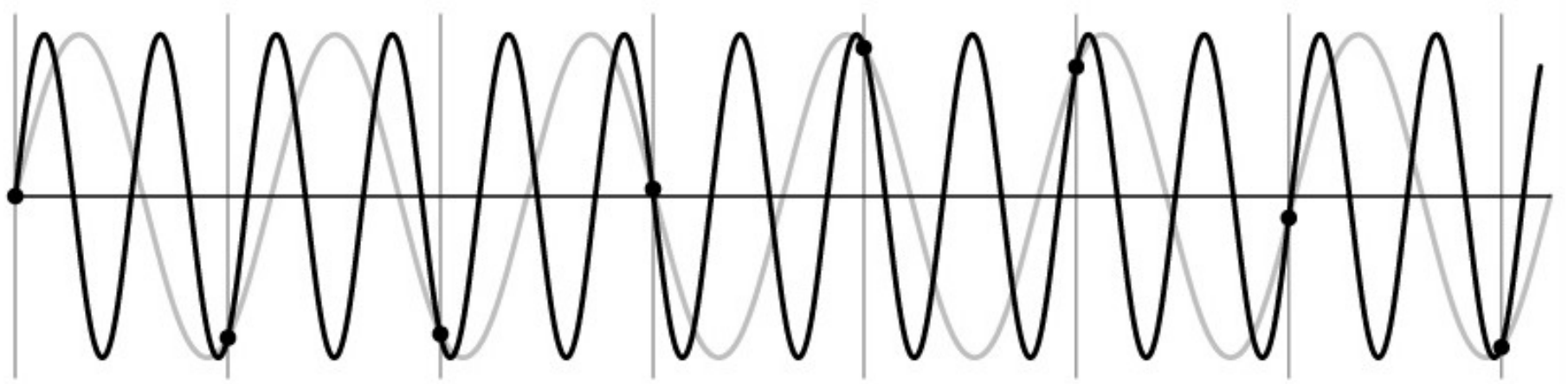


Unsurprising effect: information is lost.

Surprising effect: can confuse the signal with one of *lower* frequency.

Undersampling

Very simple example: a sine wave



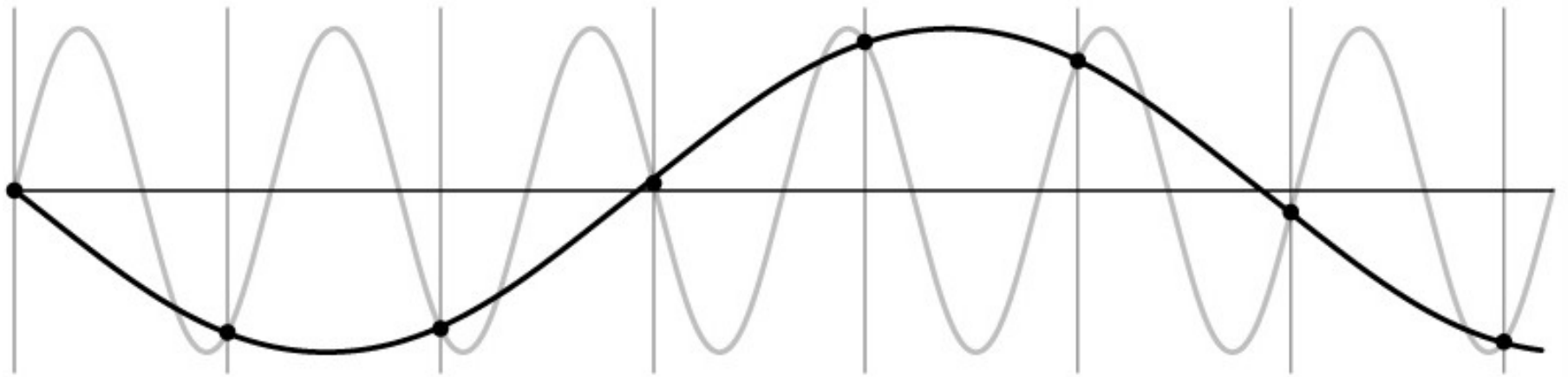
Unsurprising effect: information is lost.

Surprising effect: can confuse the signal with one of *lower* frequency.

Note: we could always confuse the signal with one of *higher* frequency.

Aliasing

Fancy term for: *Undersampling can disguise a signal as one of a lower frequency*

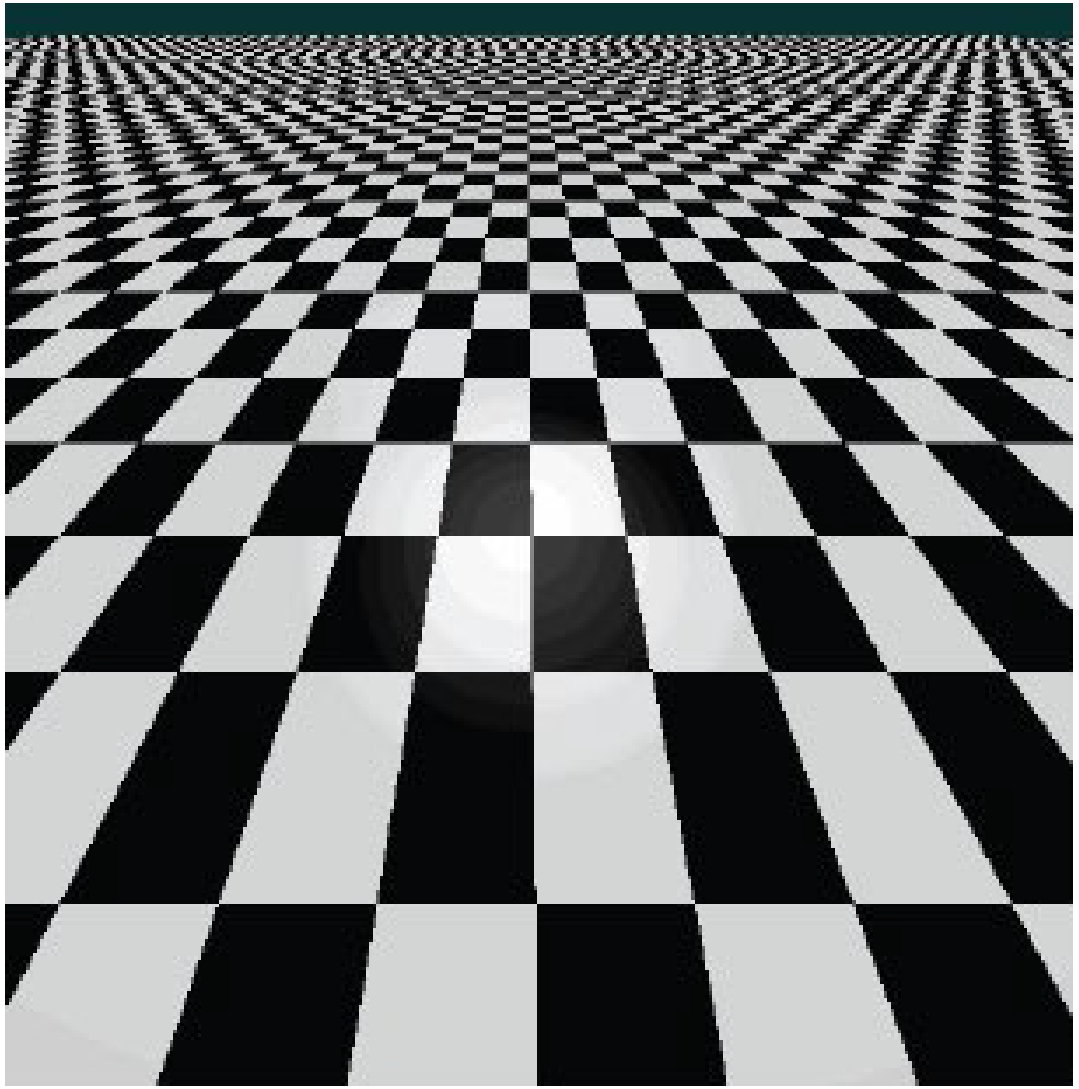


Unsurprising effect: information is lost.

Surprising effect: can confuse the signal with one of *lower* frequency.

Note: we could always confuse the signal with one of *higher* frequency.

Aliasing in textures



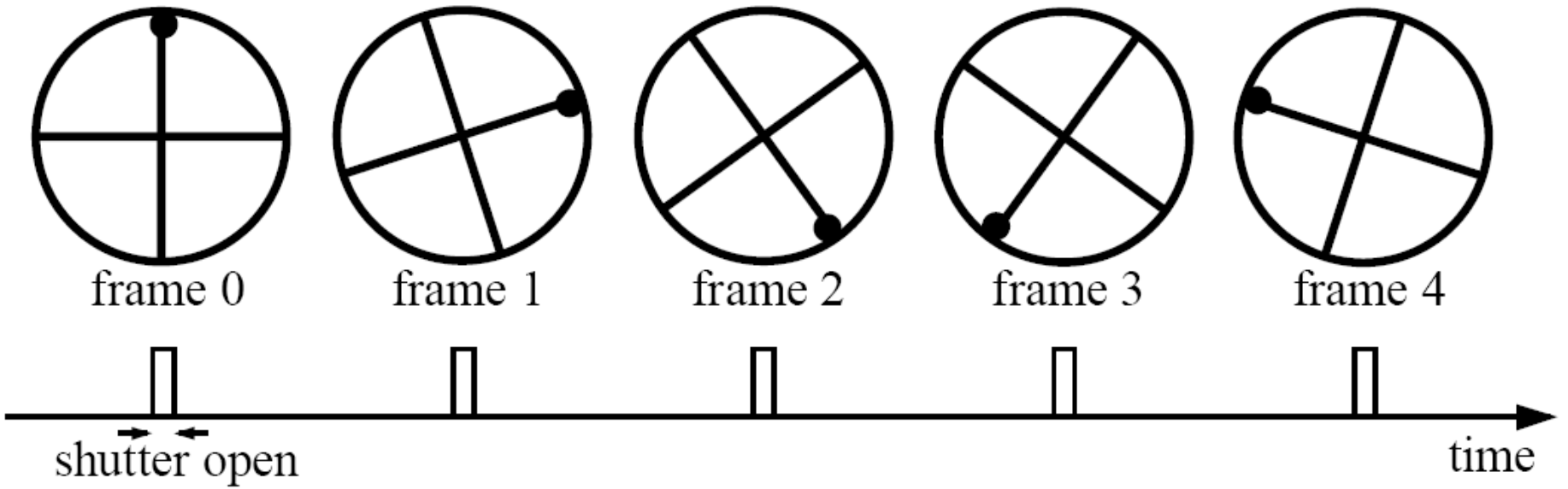
Aliasing in photographs

This is also known as “moire”

Temporal aliasing

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

https://youtu.be/HqKfCUW17QM?si=d4-y4Qsn8hacYN8e

Wagon wheel effect

https://youtu.be/yr3ngmRuGUc?si=0A5t_s5C_y0p39rB

Anti-aliasing

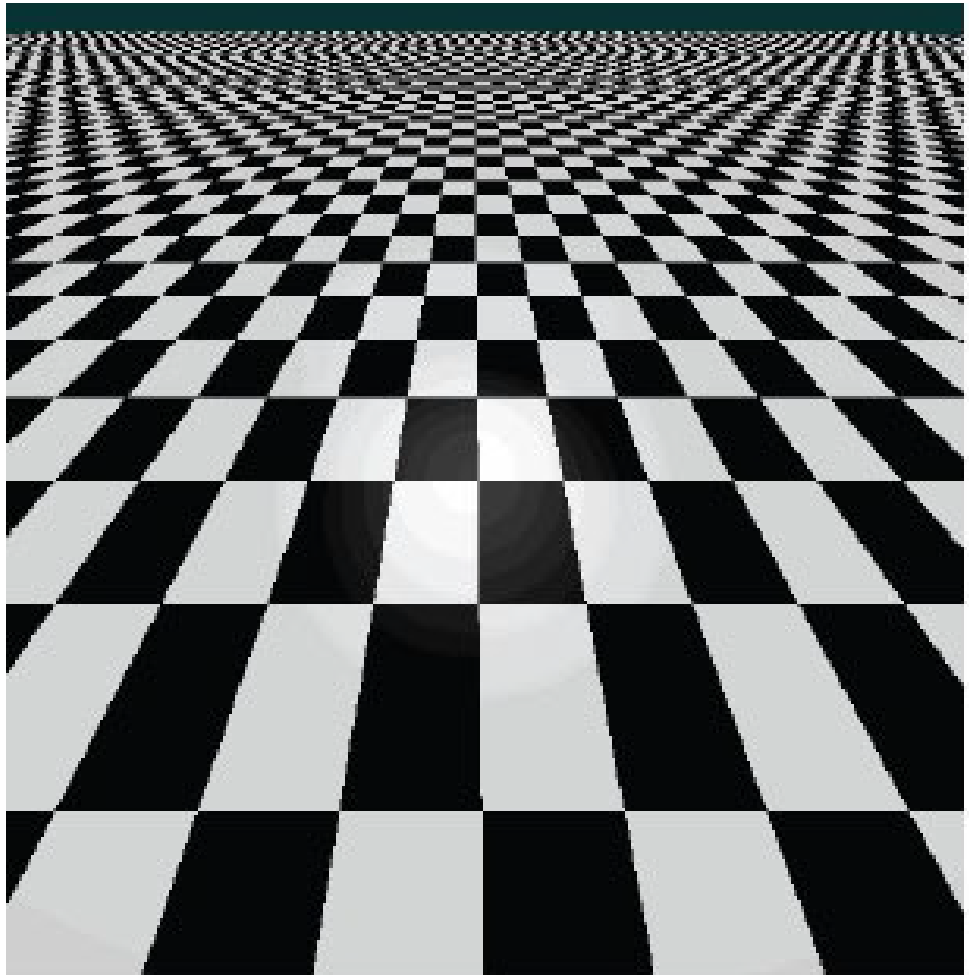
How would you deal with aliasing?

Anti-aliasing

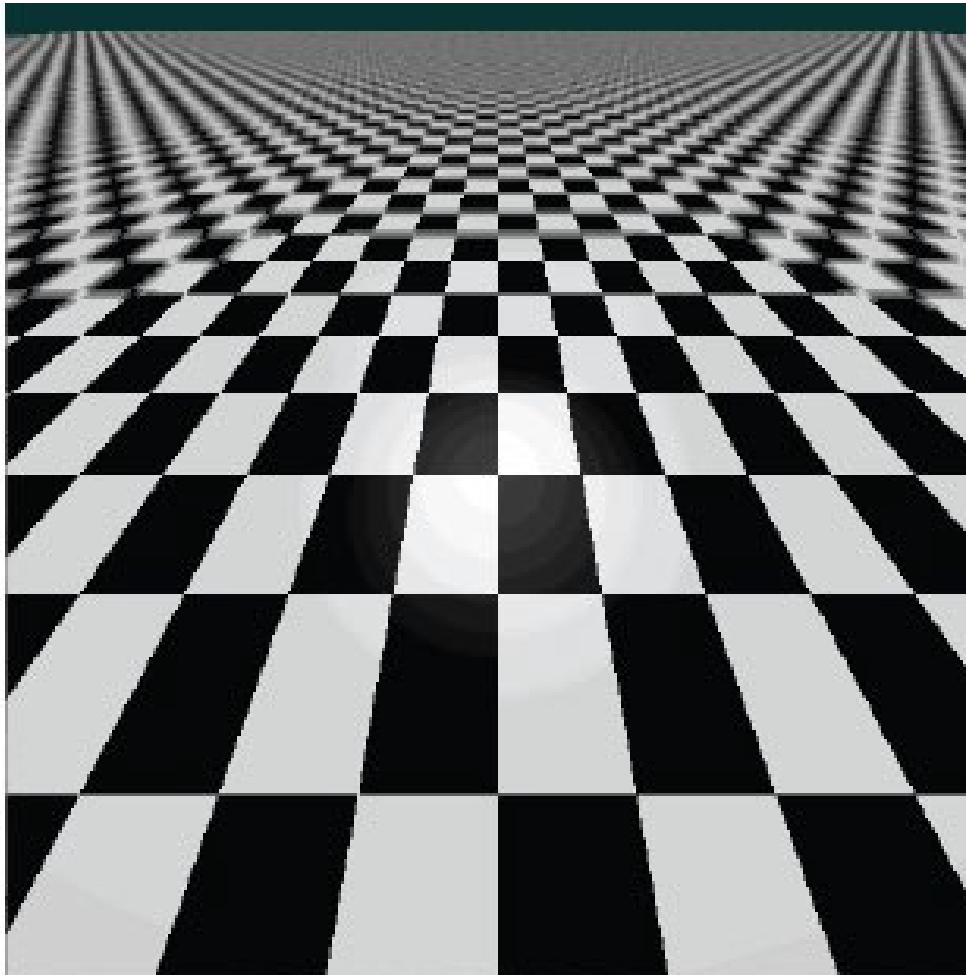
How would you deal with aliasing?

Approach 1: Oversample the signal

Anti-aliasing in textures



aliasing artifacts



anti-aliasing by oversampling

Anti-aliasing

How would you deal with aliasing?

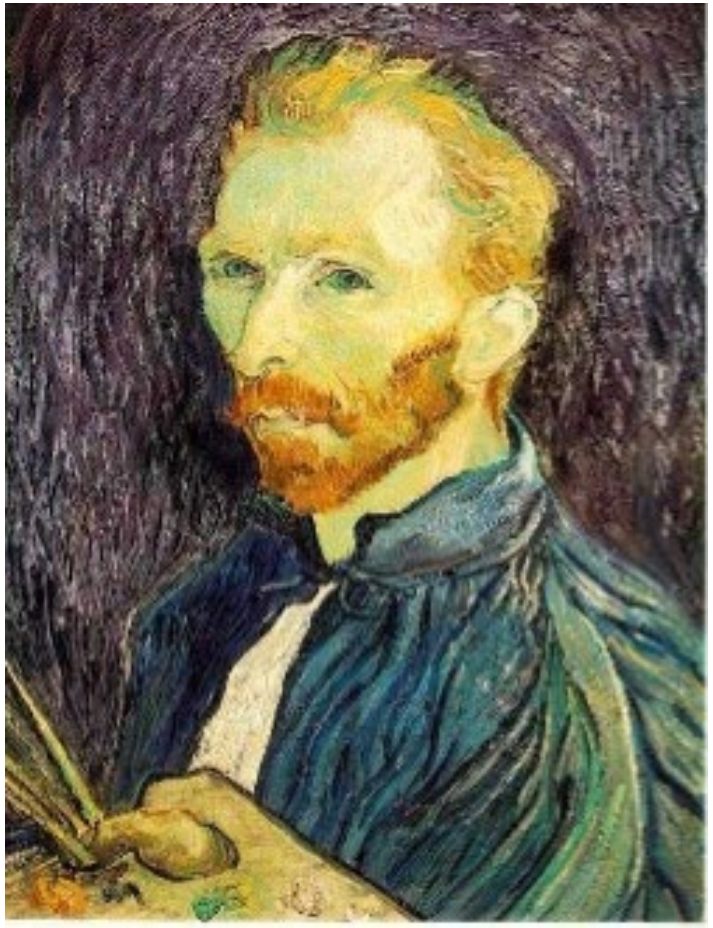
Approach 1: Oversample the signal

Approach 2: Smooth the signal

- Remove some of the detail effects that cause aliasing.
- Lose information, but better than aliasing artifacts.

How would you smooth a signal?

Better image downsampling



1/2

Apply a smoothing filter first, then throw away half the rows and columns

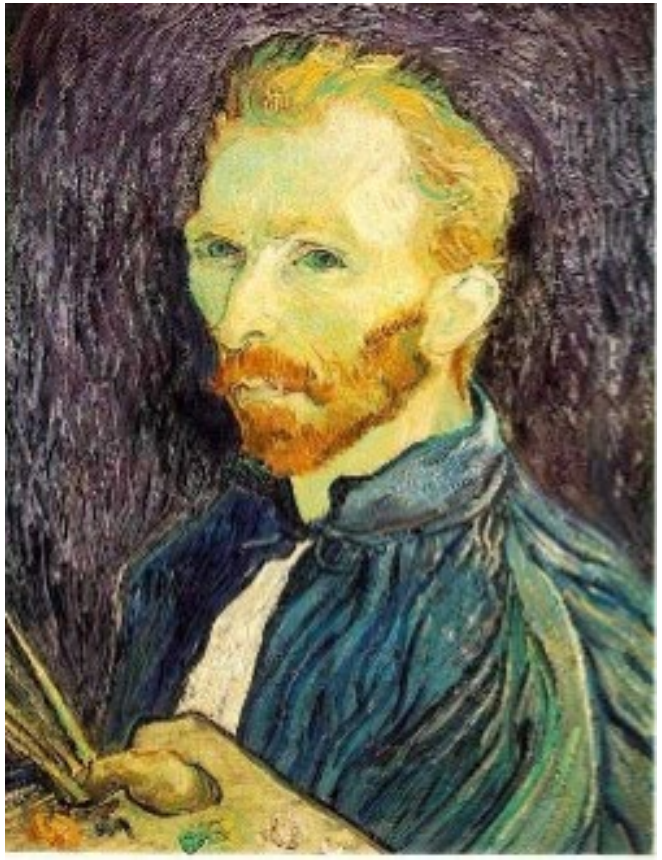
Gaussian filter
delete even rows
delete even columns

1/4

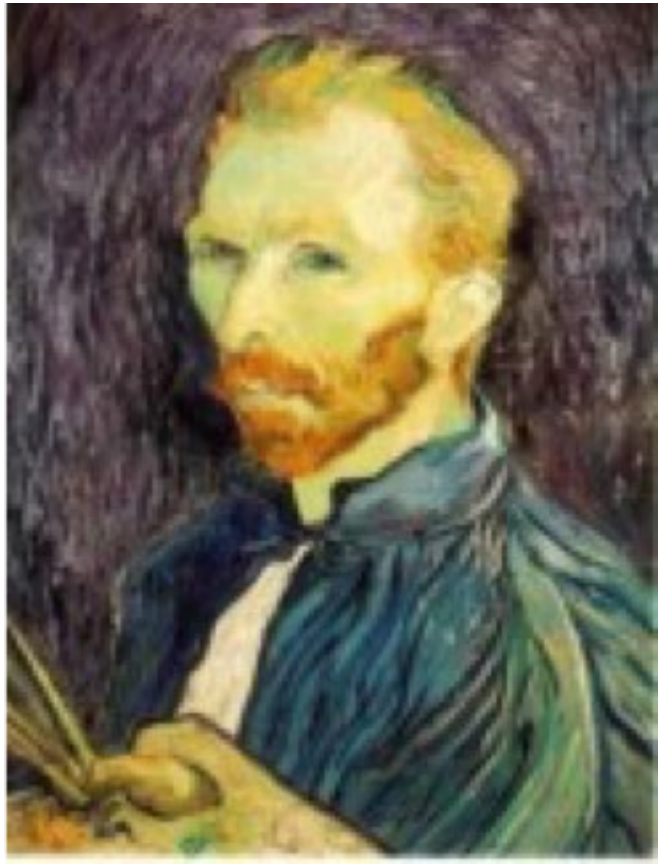
Gaussian filter
delete even rows
delete even columns

1/8

Better image downsampling



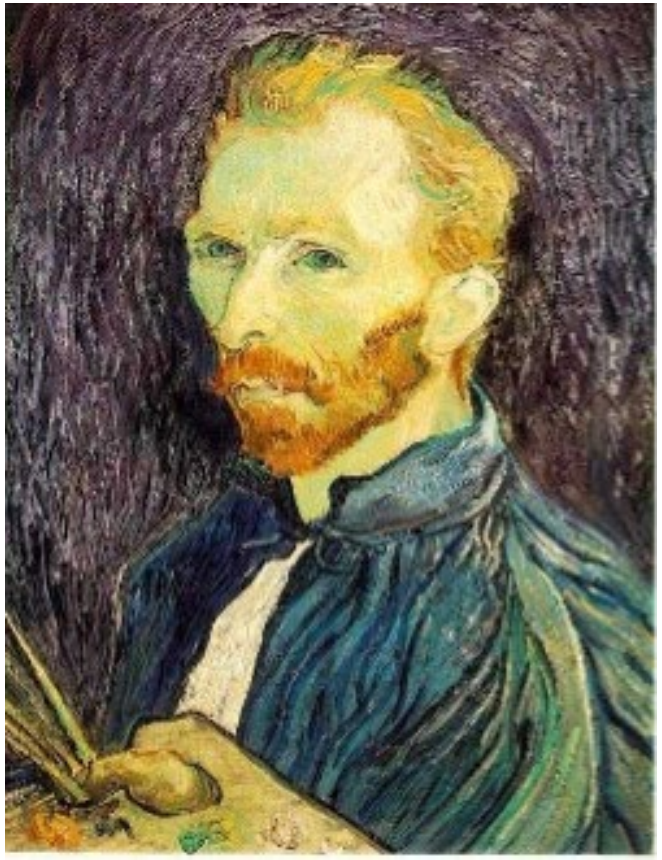
1/2



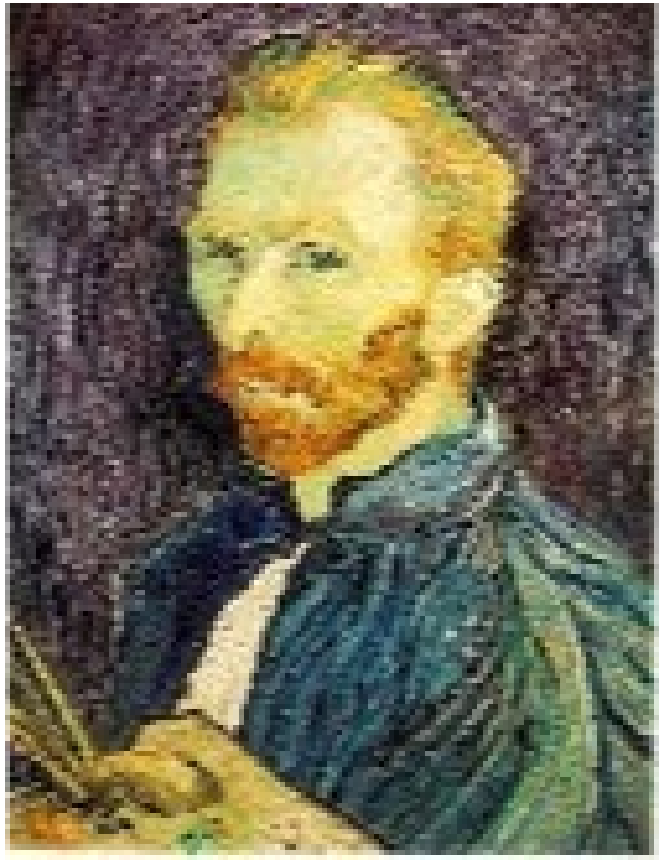
1/4 (2x zoom)

1/8 (4x zoom)

Naïve image downsampling



1/2



1/4 (2x zoom)

1/8 (4x zoom)

Anti-aliasing

Question 1: How much smoothing do I need to do to avoid aliasing?

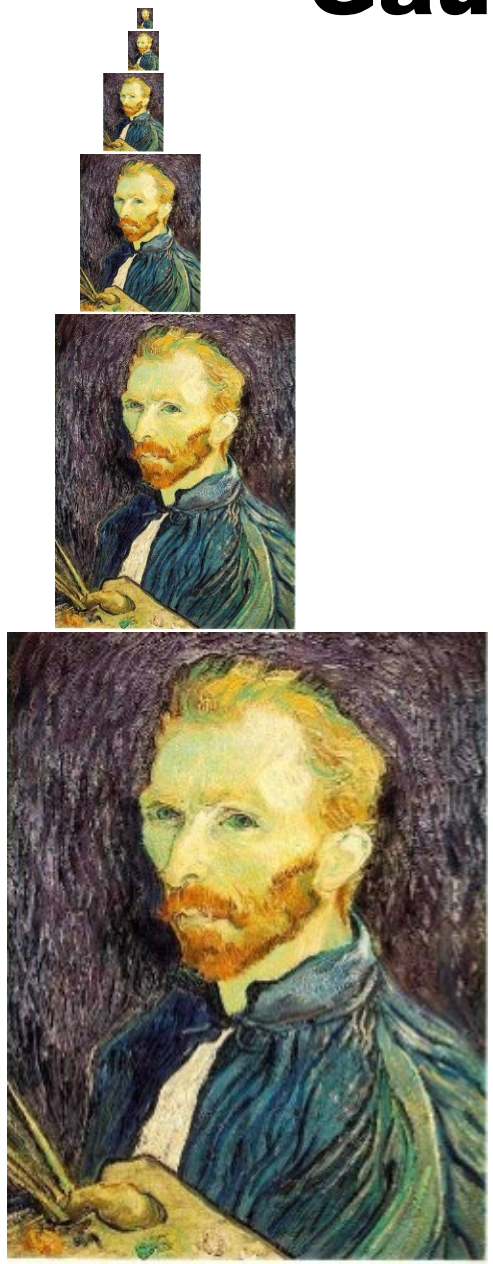
Question 2: How many samples do I need to take to avoid aliasing?

Answer to both: Enough to reach the Nyquist limit

We'll see what this means soon.

Gaussian image pyramid

Gaussian image pyramid



The name of this sequence of subsampled images

Constructing a Gaussian pyramid

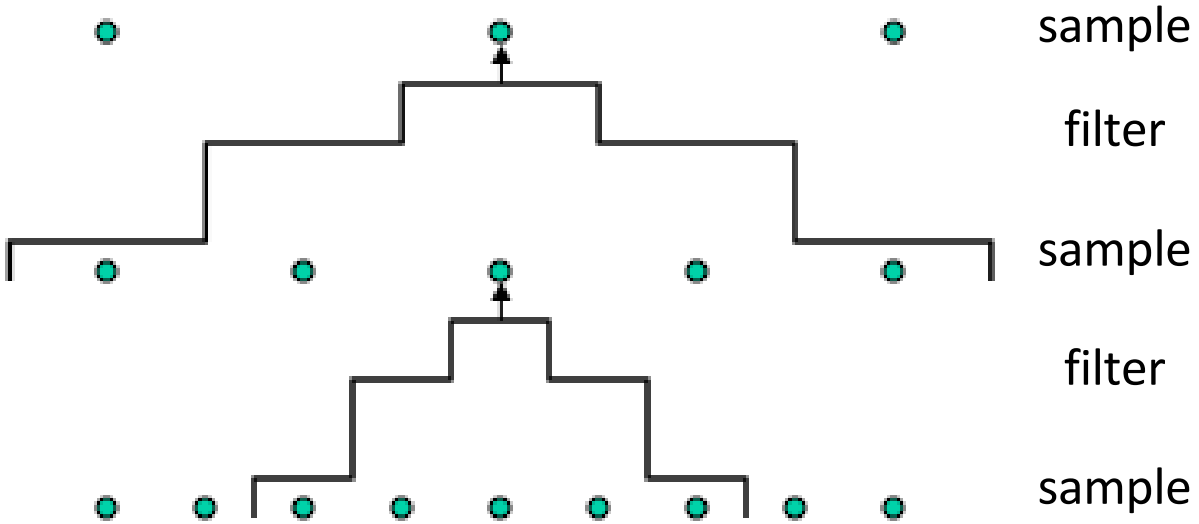
Algorithm

repeat:

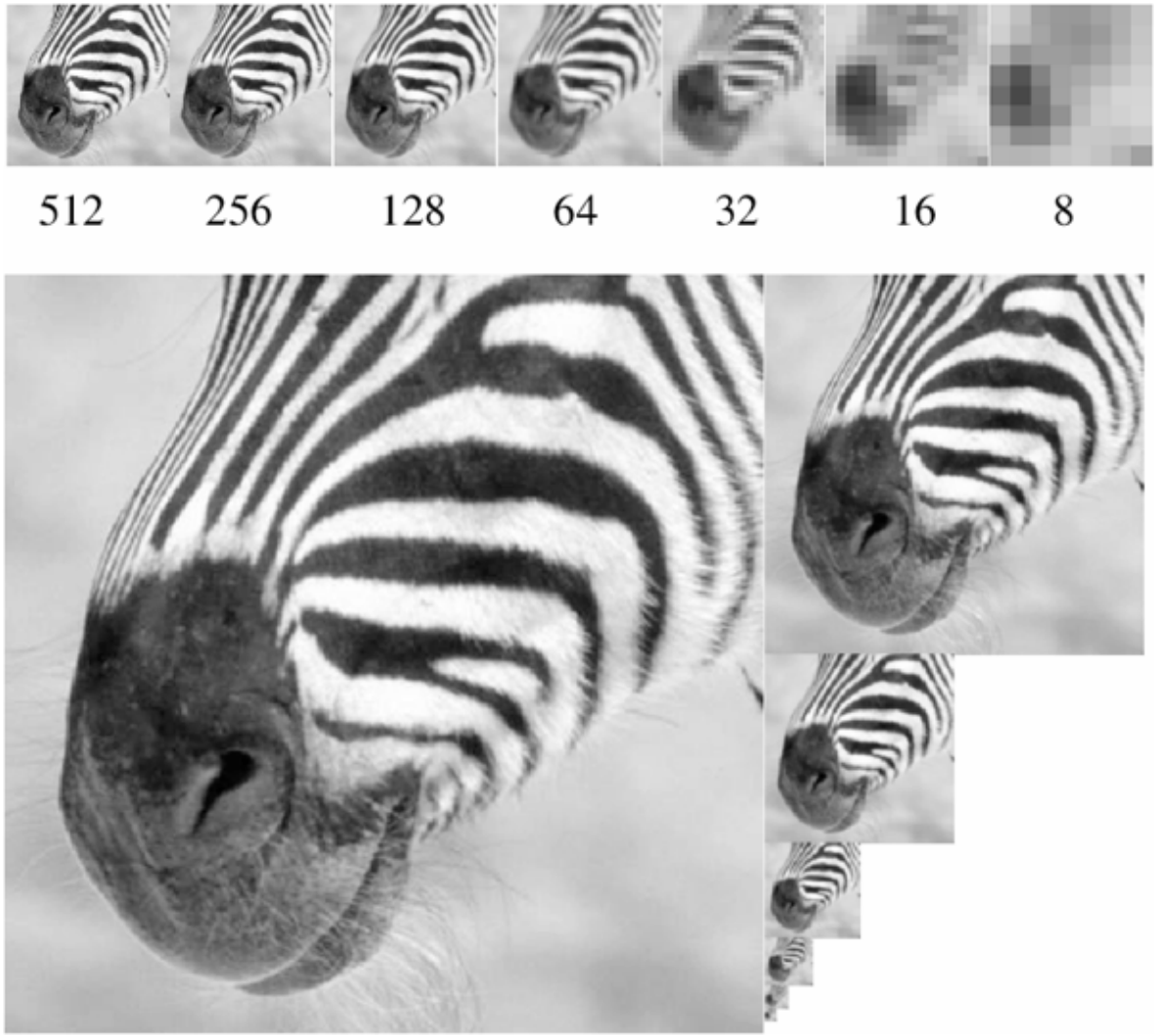
filter

subsample

until min resolution reached

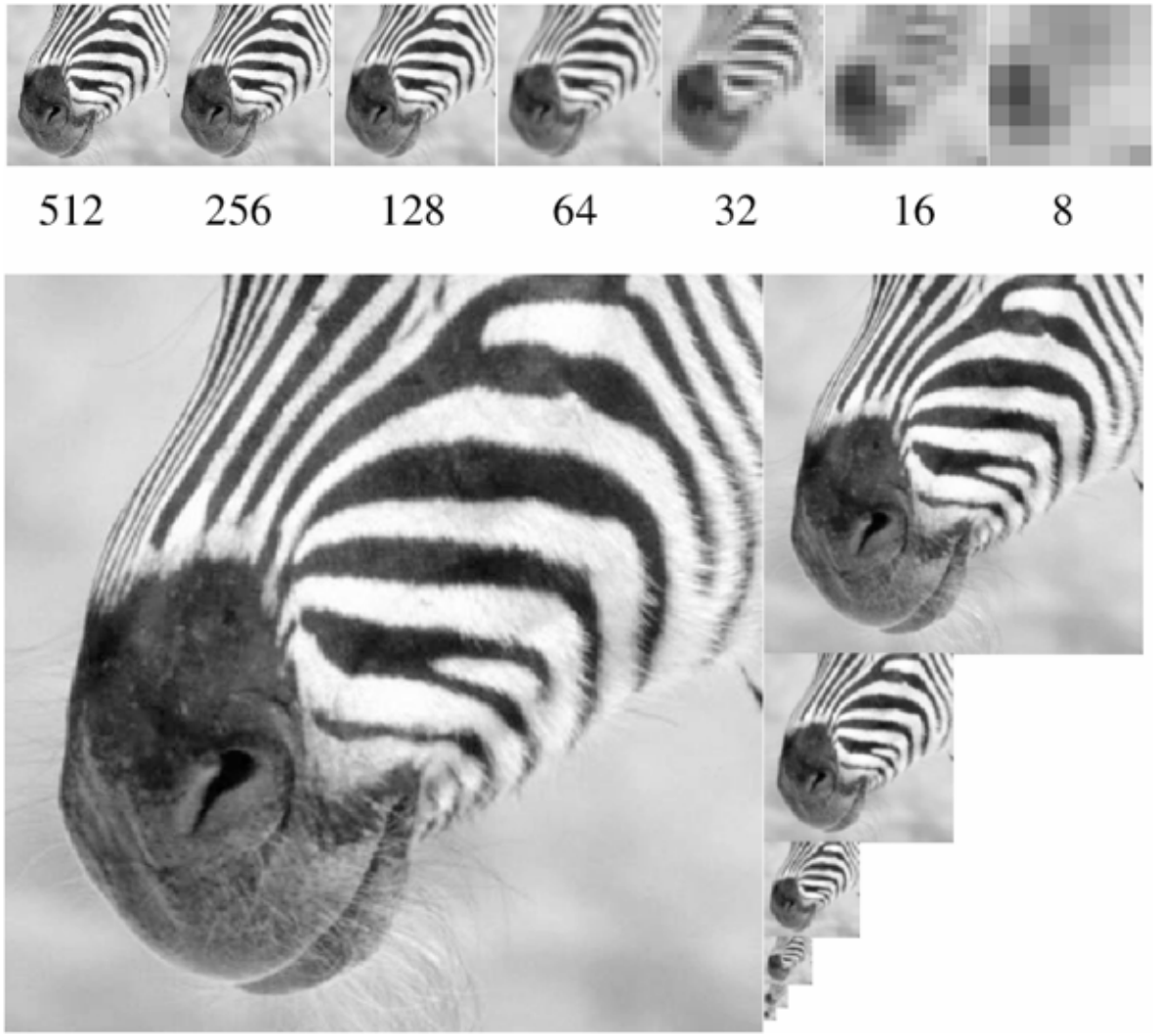


Some properties of the Gaussian pyramid



What happens to the details of the image?

Some properties of the Gaussian pyramid

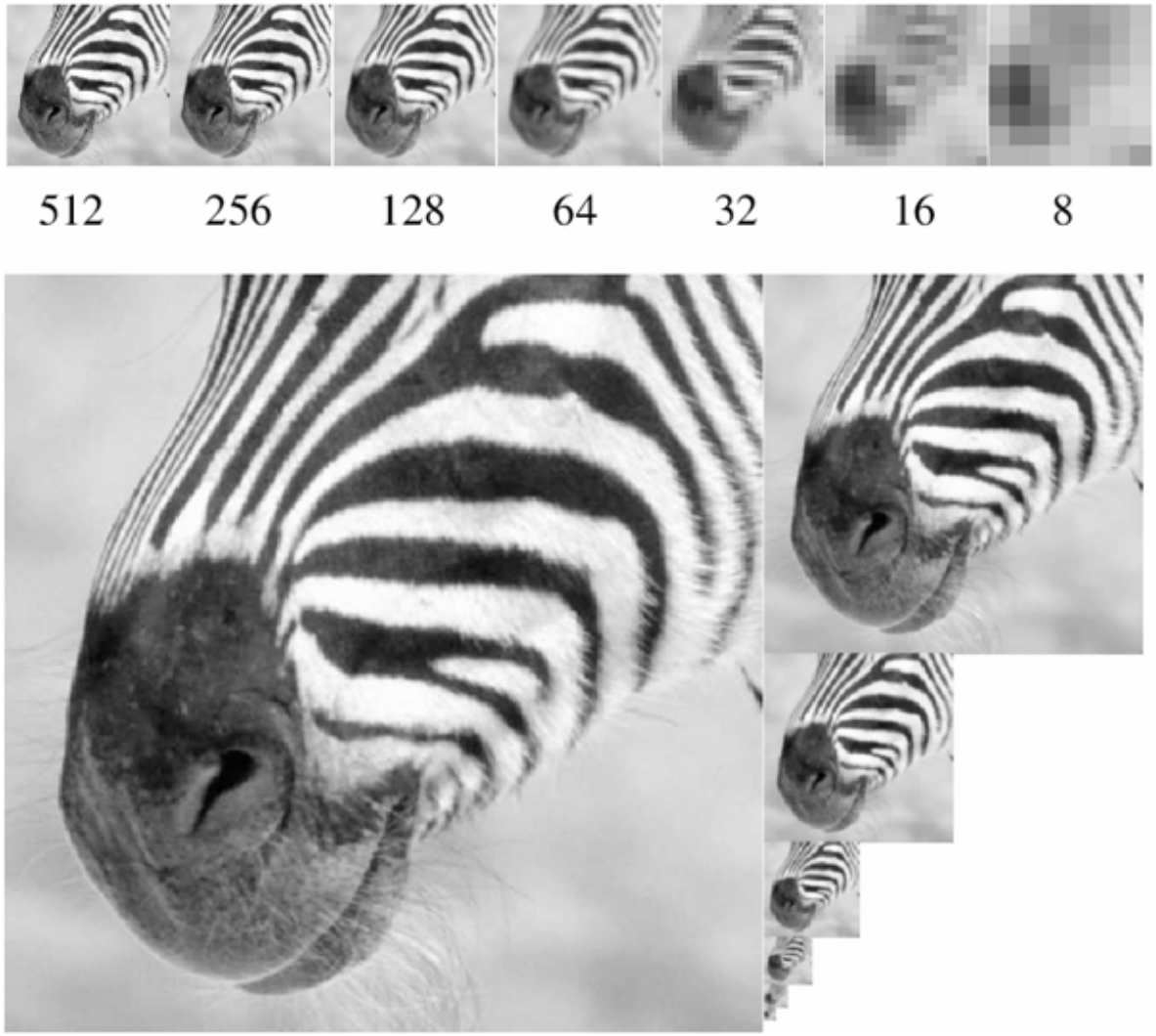


What happens to the details of the image?

- They get smoothed out as we move to higher levels.

What is preserved at the higher levels?

Some properties of the Gaussian pyramid



What happens to the details of the image?

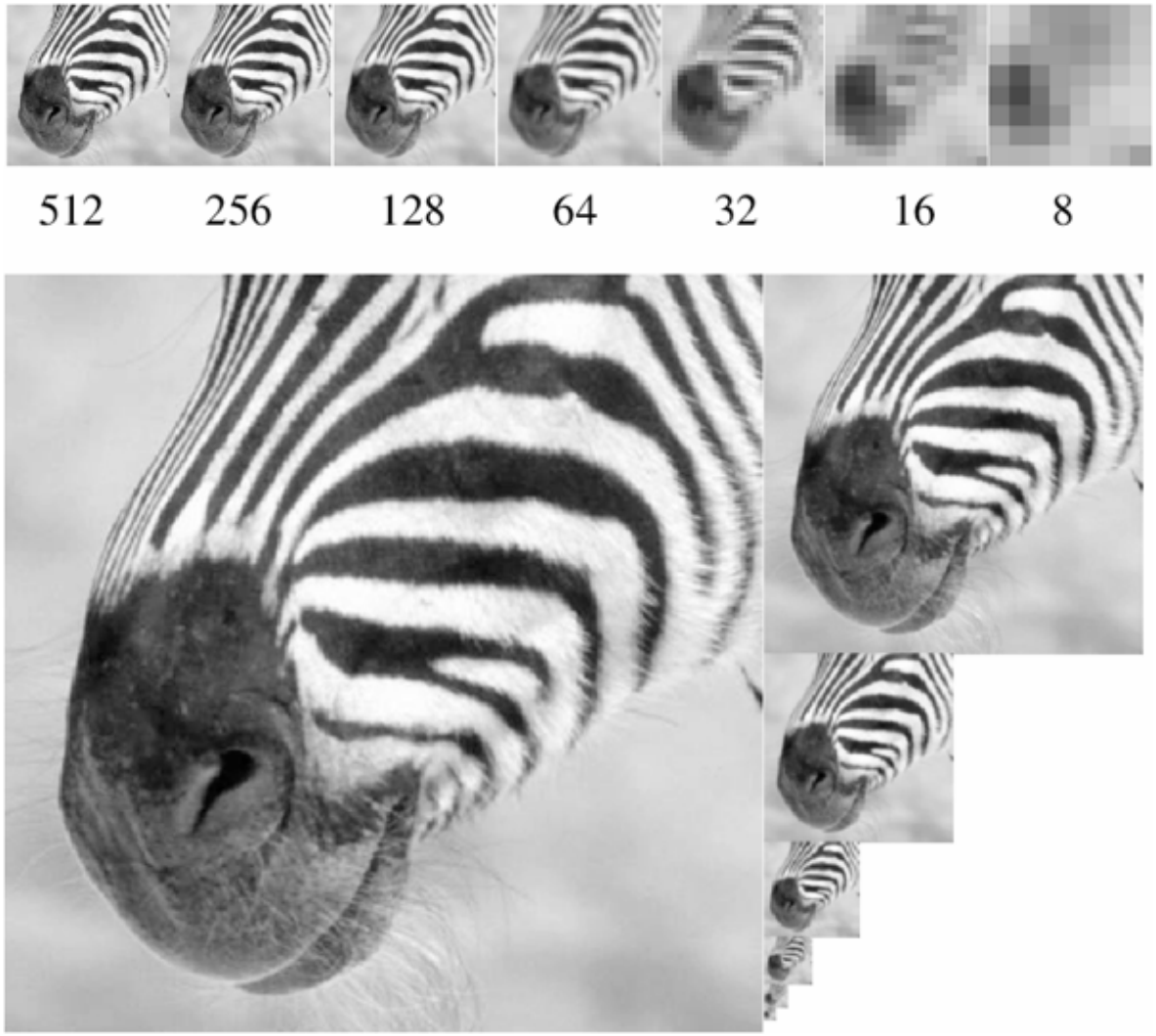
- They get smoothed out as we move to higher levels.

What is preserved at the higher levels?

- Mostly large uniform regions in the original image.

How would you reconstruct the original image from the image at the upper level?

Some properties of the Gaussian pyramid



What happens to the details of the image?

- They get smoothed out as we move to higher levels.

What is preserved at the higher levels?

- Mostly large uniform regions in the original image.

How would you reconstruct the original image from the image at the upper level?

- That's not possible.

Blurring is lossy

level 0

level 1 (before downsampling)

What does the residual look like?

Blurring is lossy

level 0

-

level 1 (before downsampling)

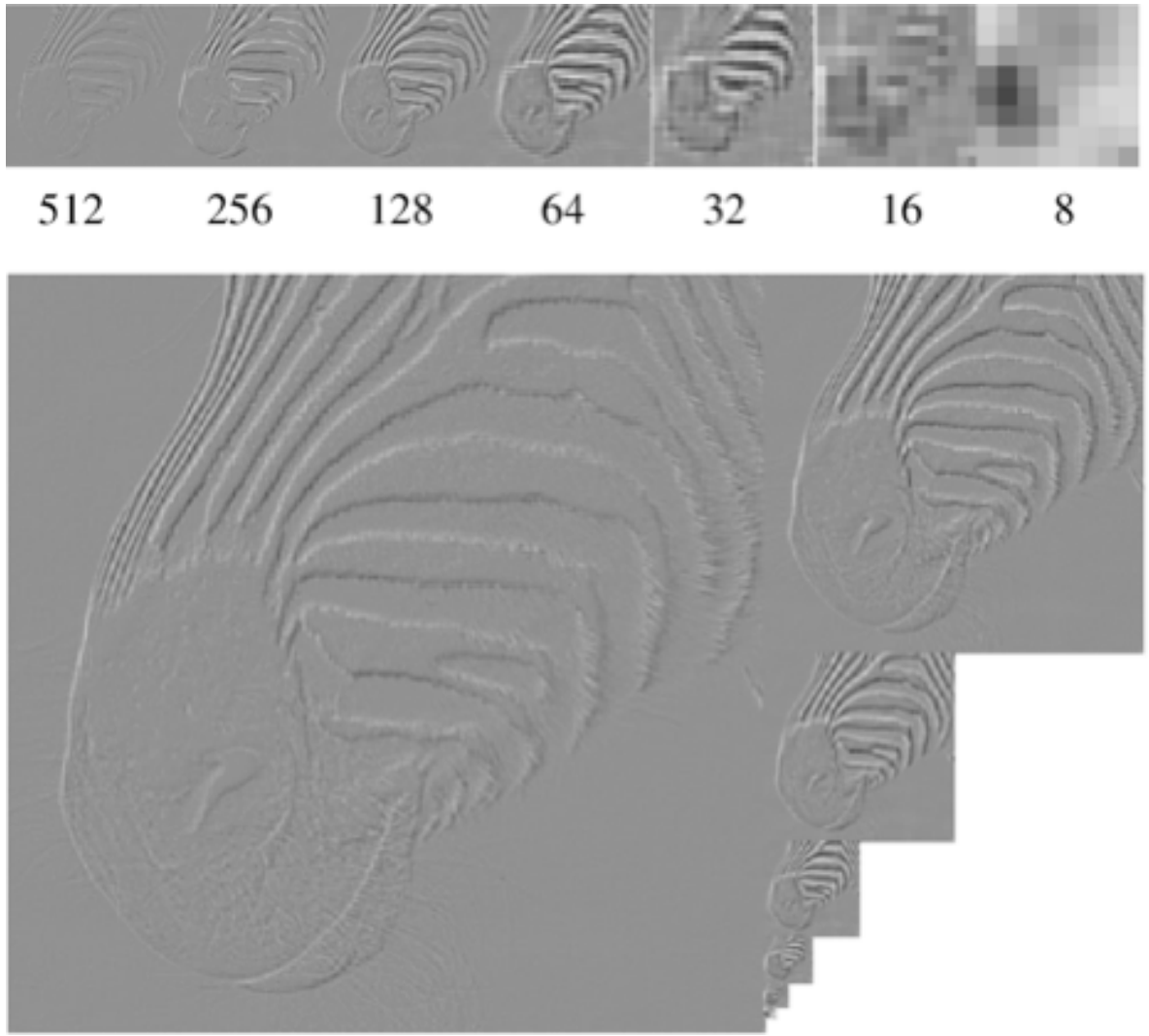
=

residual

Can we make a pyramid that is lossless?

Laplacian image pyramid

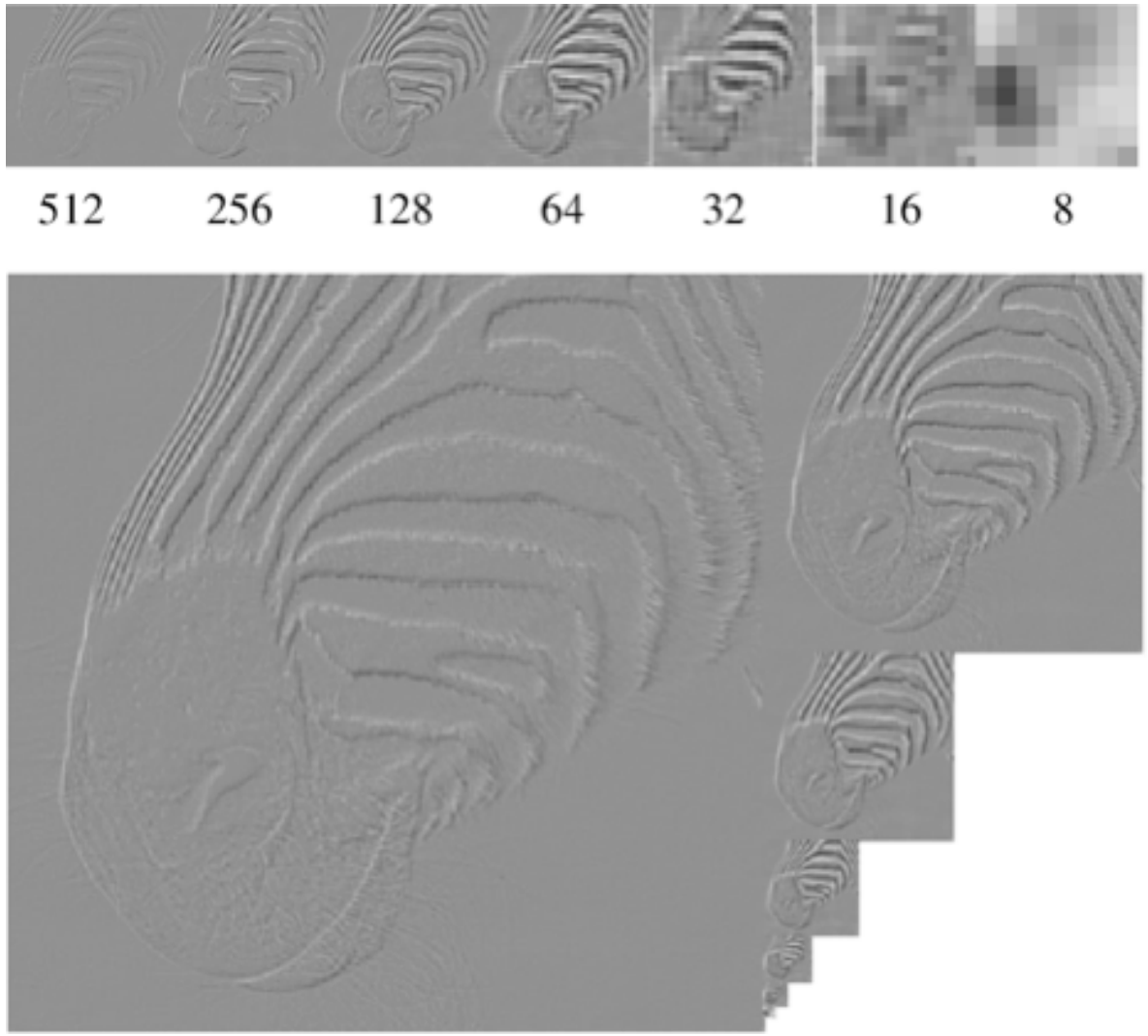
Laplacian image pyramid



At each level, retain the residuals instead of the blurred images themselves.

Can we reconstruct the original image using the pyramid?

Laplacian image pyramid



At each level, retain the residuals instead of the blurred images themselves.

Can we reconstruct the original image using the pyramid?

- Yes we can!

What do we need to store to be able to reconstruct the original image?

Let’s start by looking at just one level

level 0

=

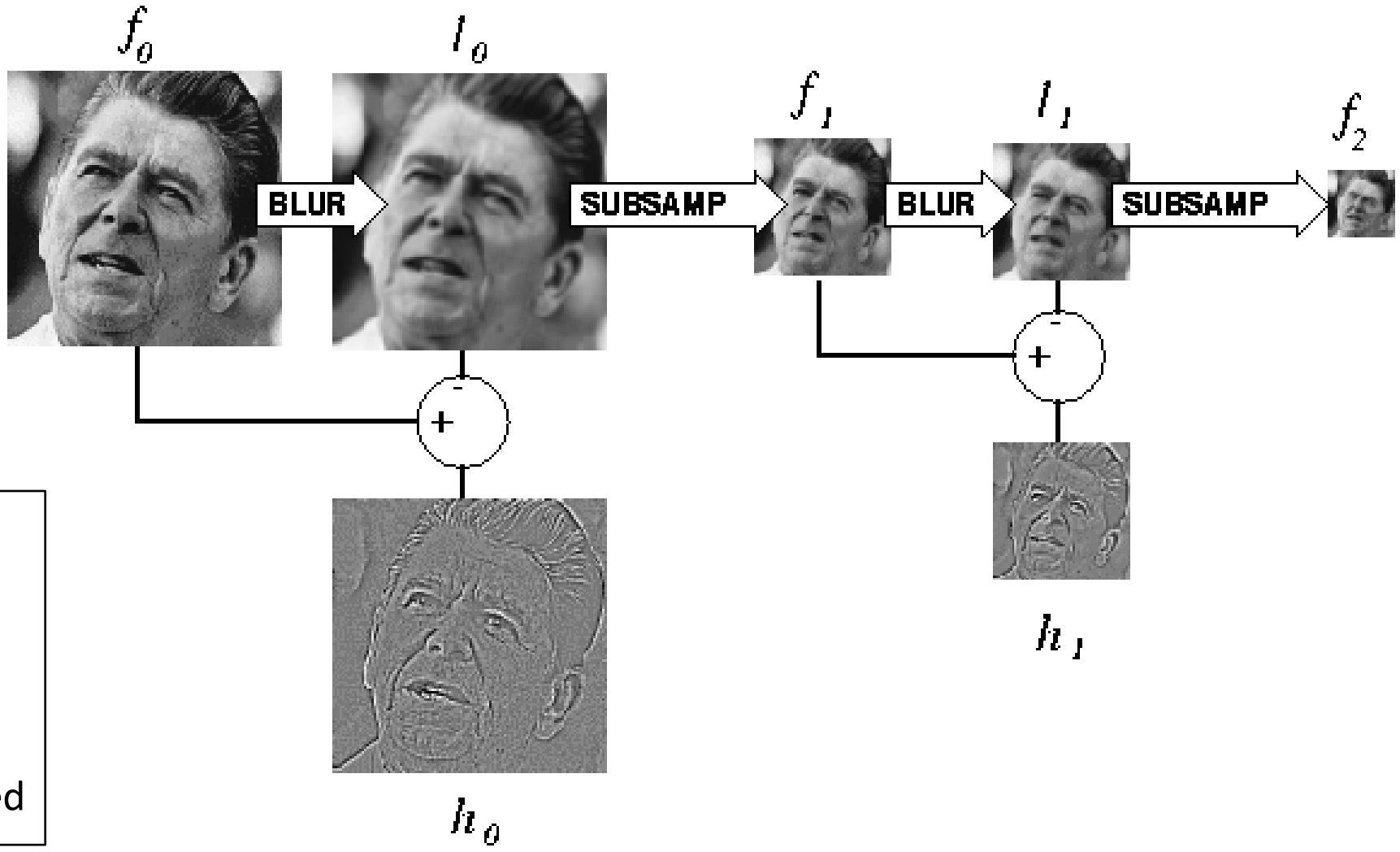
level 1 (upsampled)

+

residual

Does this mean we need to store both residuals and the blurred copies of the original?

Constructing a Laplacian pyramid



Algorithm

repeat:

filter

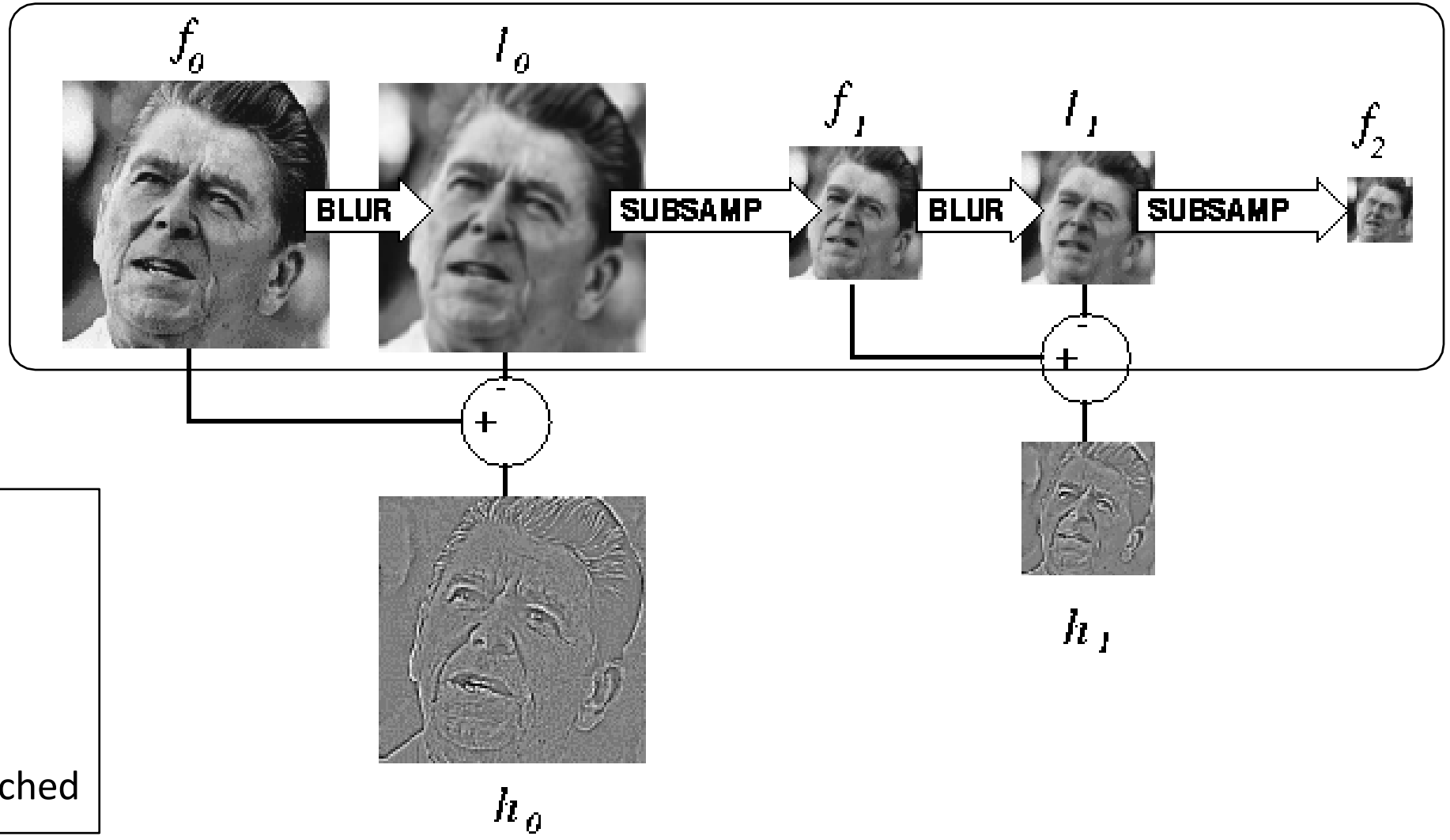
compute residual

subsample

until min resolution reached

Constructing a Laplacian pyramid

What is this part?



Algorithm

repeat:

filter

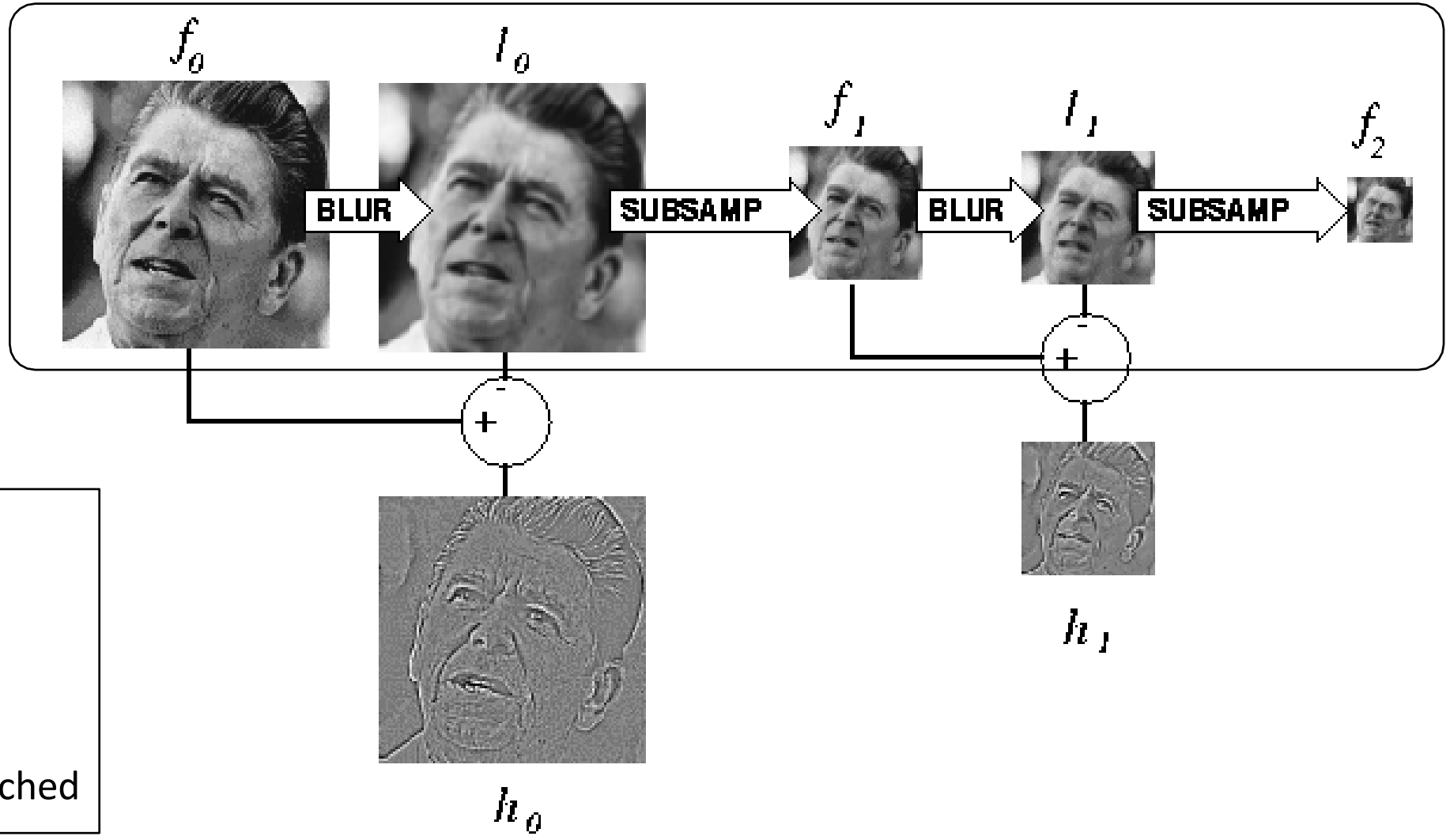
compute residual

subsample

until min resolution reached

Constructing a Laplacian pyramid

It's a Gaussian pyramid.



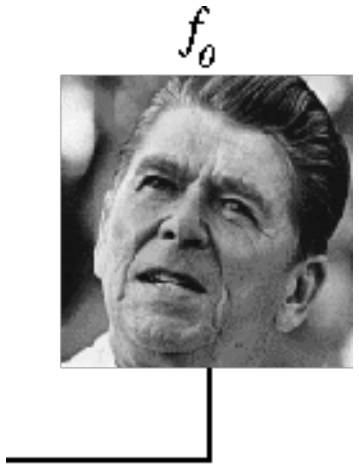
Algorithm

repeat:

- filter
- compute residual
- subsample

until min resolution reached

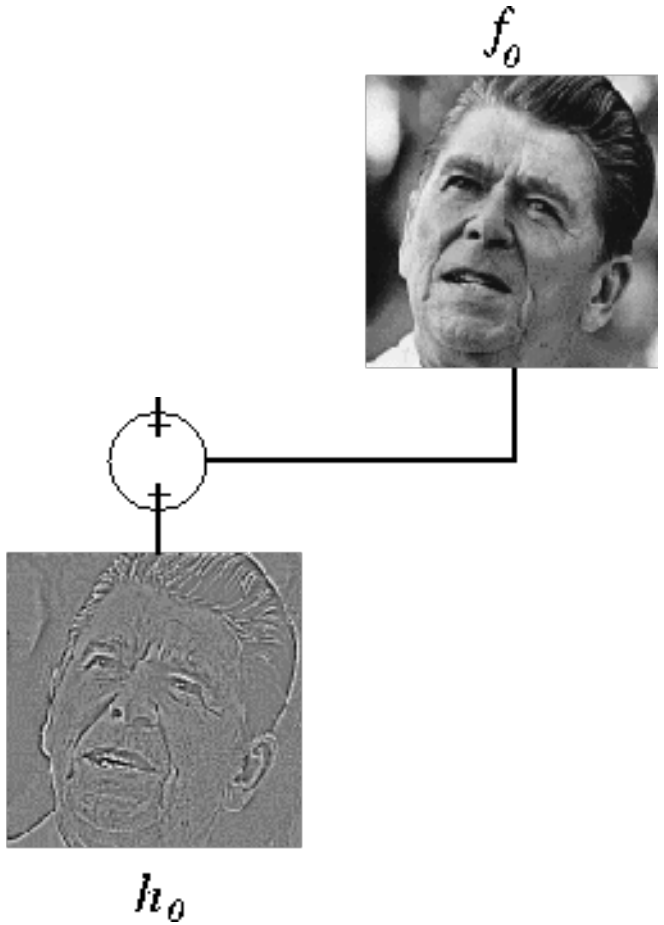
What do we need to construct the original image?



What do we need to construct the original image?

h_1

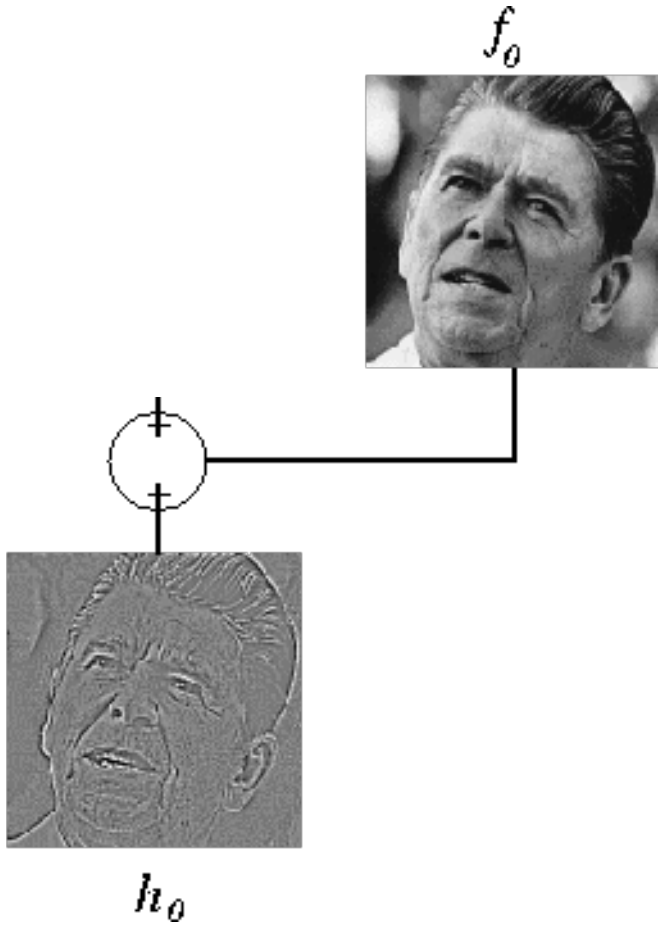
(1) residuals



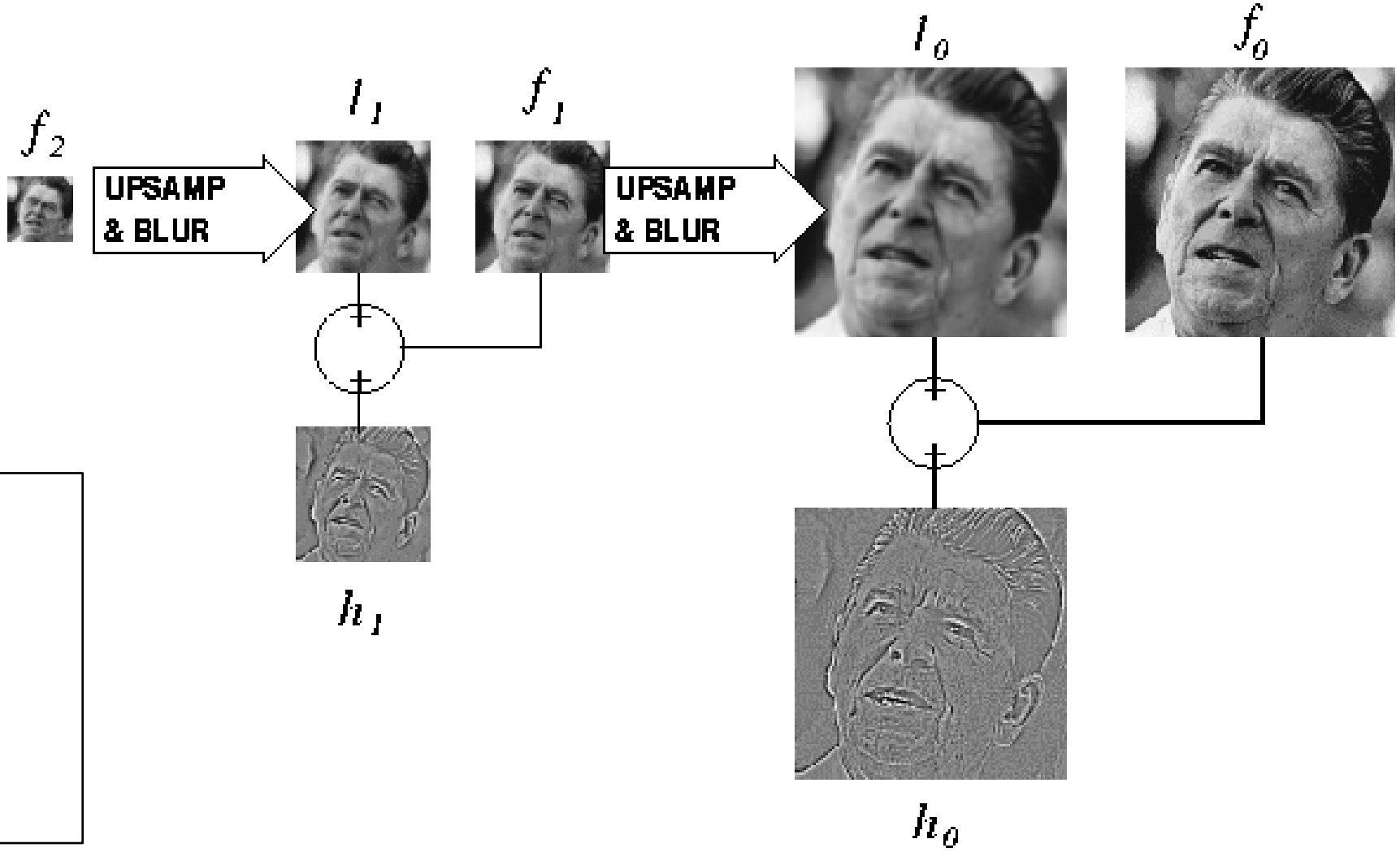
What do we need to construct the original image?

(2) smallest image

(1) residuals



Reconstructing the original image



Algorithm

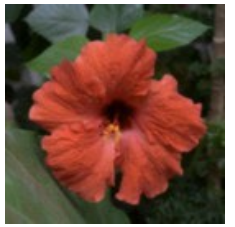
repeat:

upsample

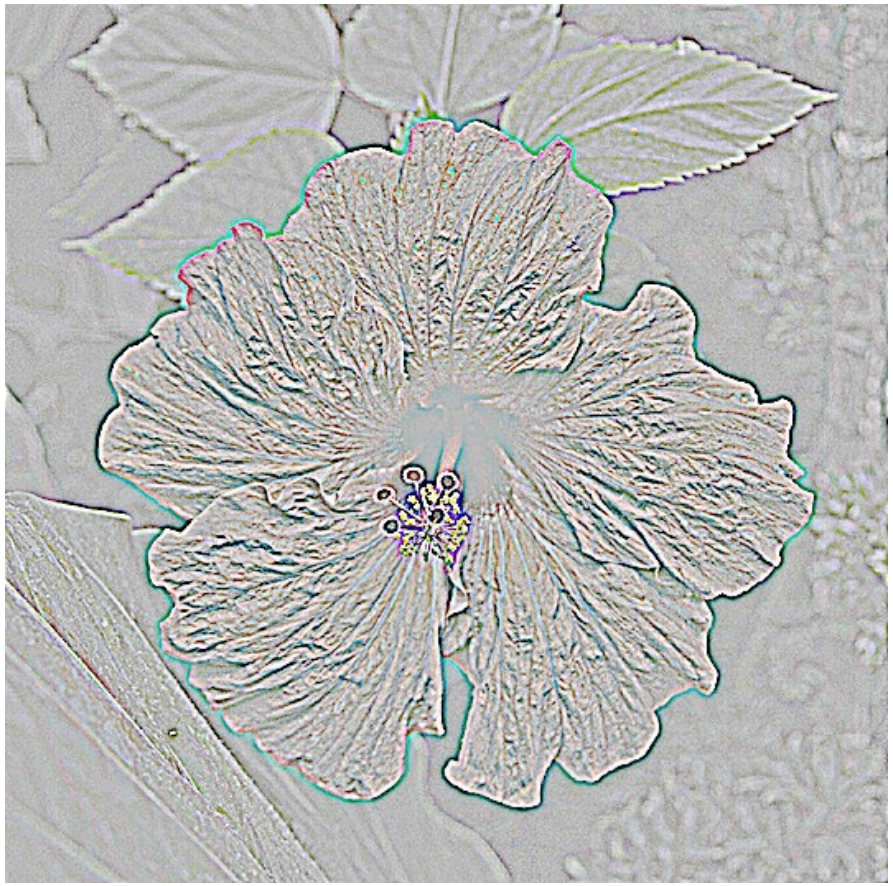
sum with residual

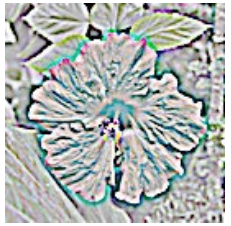
until orig resolution reached

Gaussian vs Laplacian Pyramid



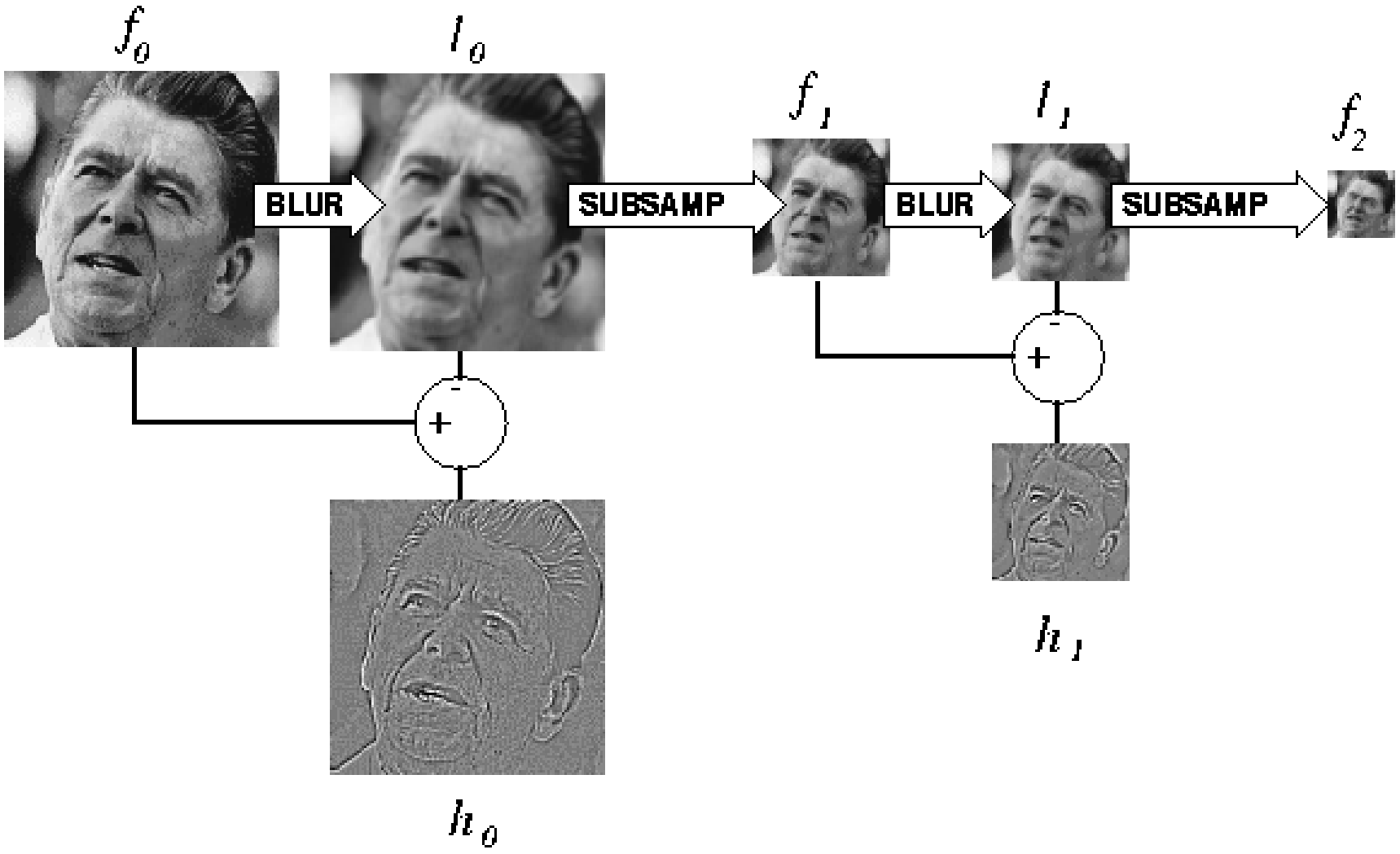
Shown in opposite order for space.





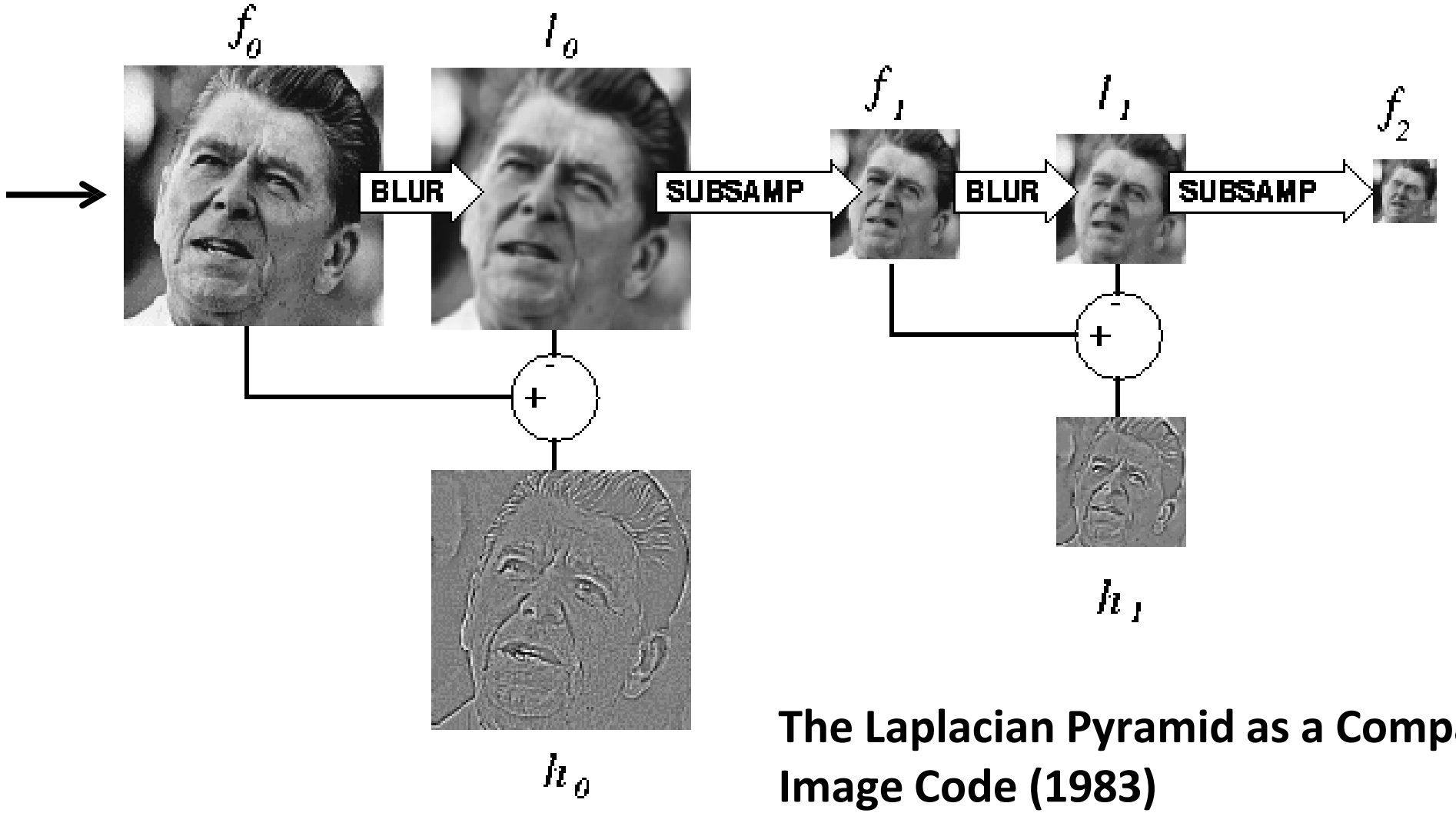
Which one takes more space to store?

Why Reagan?



Why Reagan?

Ronald Reagan was President when the Laplacian pyramid was invented



The Laplacian Pyramid as a Compact Image Code (1983)

Peter J. Burt , Edward H. Adelson

Still used extensively

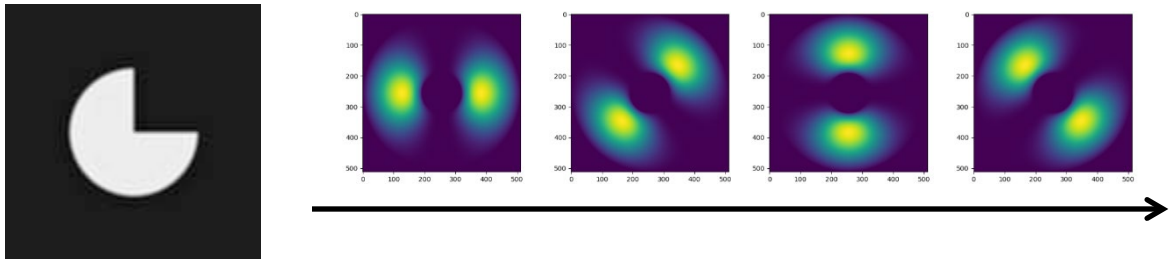
foreground details enhanced, background details reduced

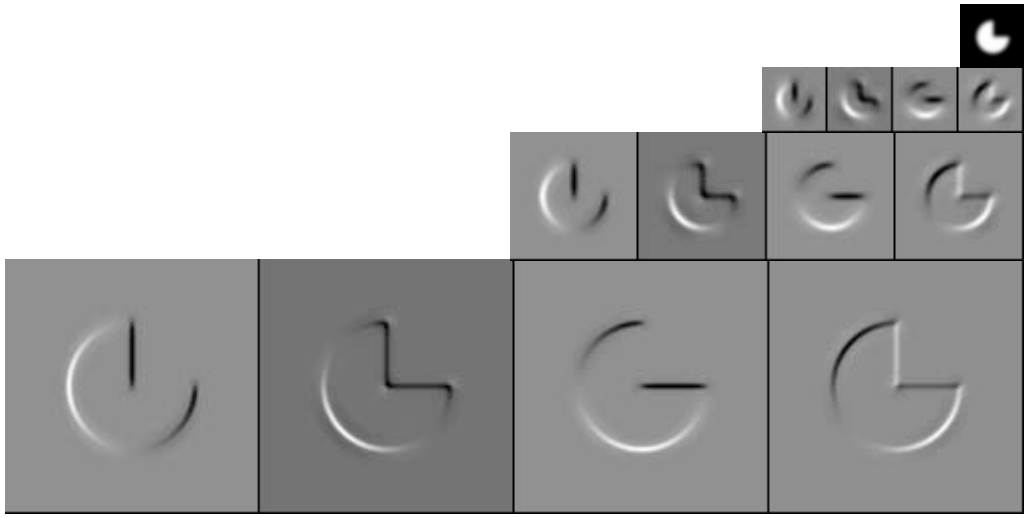
input image

user-provided mask

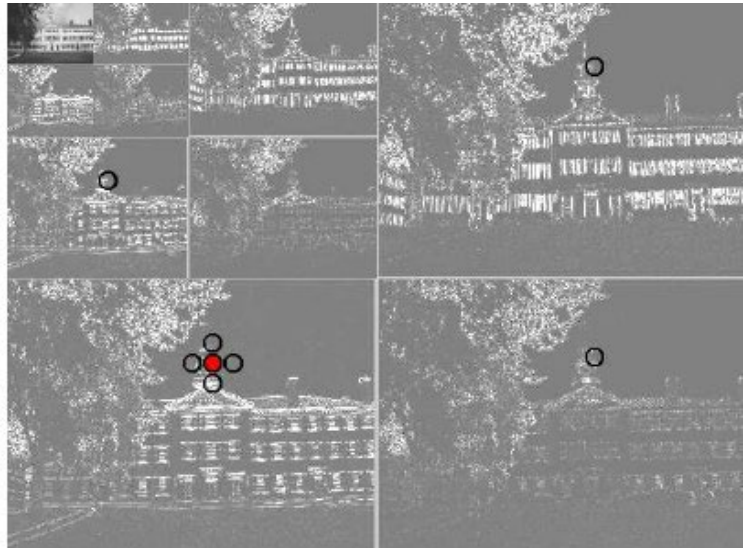
Other types of pyramids

Steerable pyramid: At each level keep multiple versions, one for each direction





Wavelets: Huge area in image processing



What are image pyramids used for?

image compression

multi-scale texture mapping

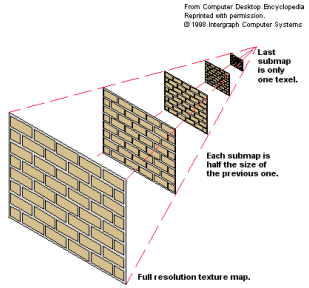
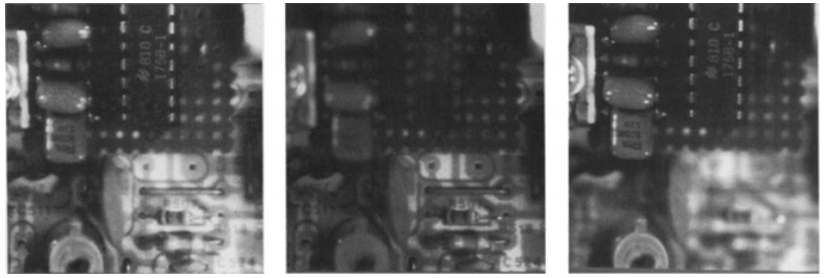
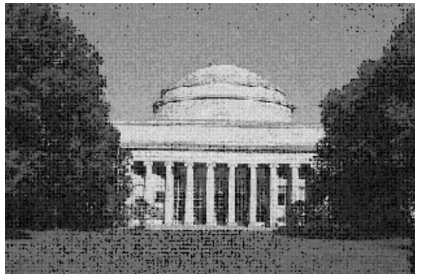


image blending

focal stack compositing

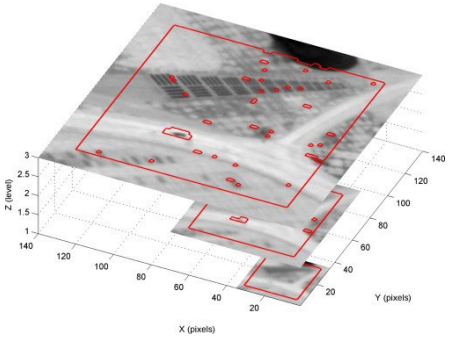


denoising



multi-scale detection

multi-scale registration



Fourier series

Some history

Who is this guy?

What is he famous for?

Jean Baptiste Joseph Fourier
(1768-1830)

What is he famous for?

Jean Baptiste Joseph Fourier
(1768-1830)

The Fourier series claim (1807):

'Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.'

… and apparently also for the discovery of the greenhouse effect

Is this claim true?

Jean Baptiste Joseph Fourier
(1768-1830)

The Fourier series claim (1807):

'Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.'

Is this claim true?

Jean Baptiste Joseph Fourier
(1768-1830)

The Fourier series claim (1807):

'Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.'

Well, almost.

- The theorem requires additional conditions.
- Close enough to be named after him.
- Very surprising result at the time.

Is this claim true?

Jean Baptiste Joseph Fourier
(1768-1830)

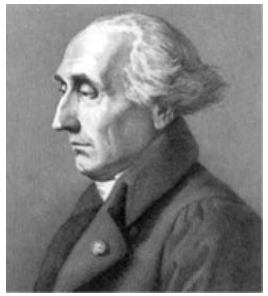
The Fourier series claim (1807):

'<u>Any</u> univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.'

Well, almost.

- The theorem requires additional conditions.
- Close enough to be named after him.
- Very surprising result at the time.

Malus



Lagrange

Legendre

Laplace

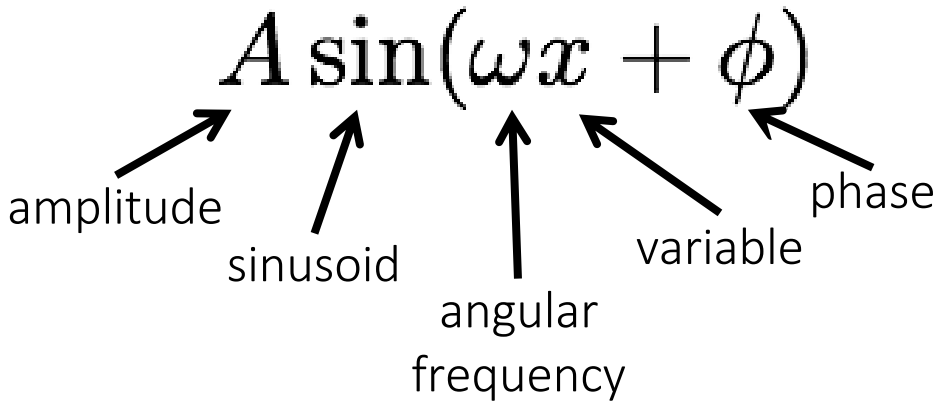
The committee examining his paper had expressed skepticism, in part due to not so rigorous proofs

Basic building block

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

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

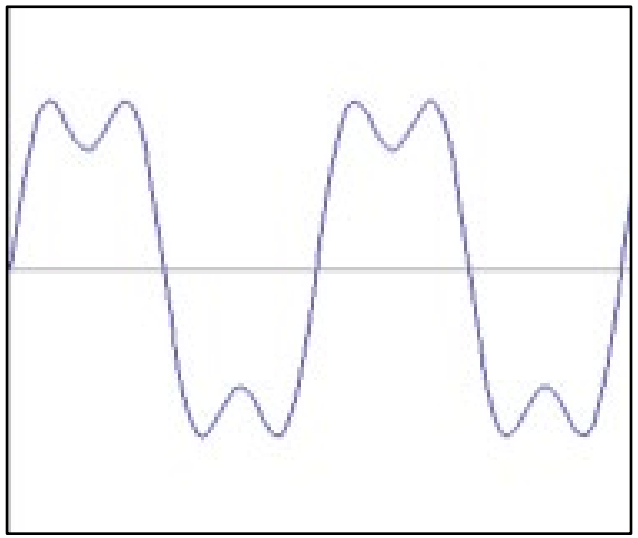
Basic building block



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

Examples

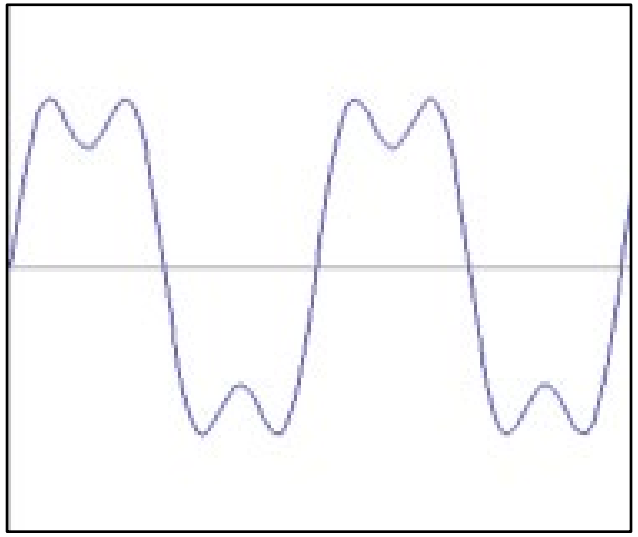
How would you generate this function?



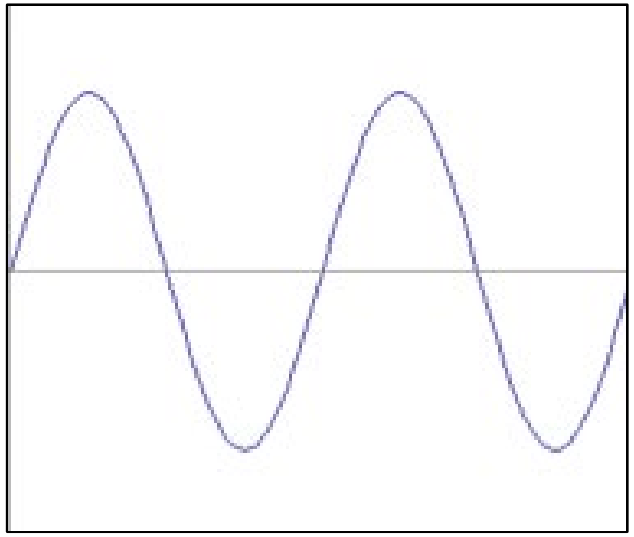
= ? + ?

Examples

How would you generate this function?



=

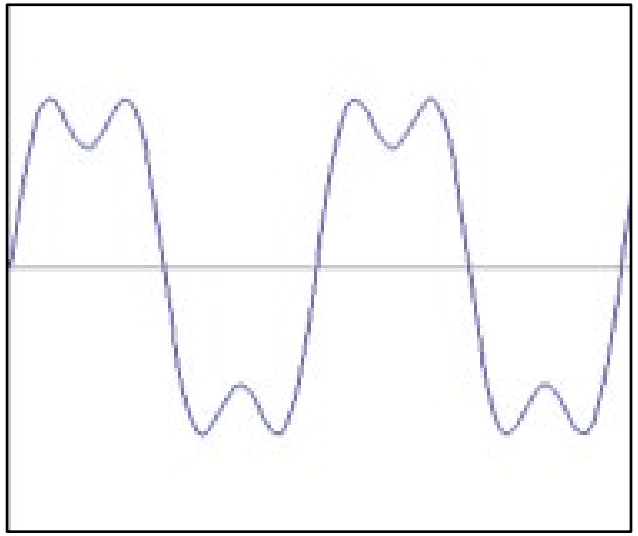


$\sin(2\pi x)$

+ ?

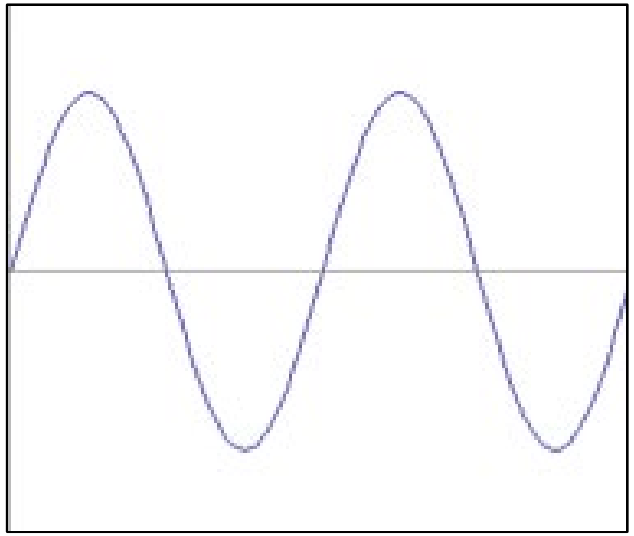
Examples

How would you generate this function?



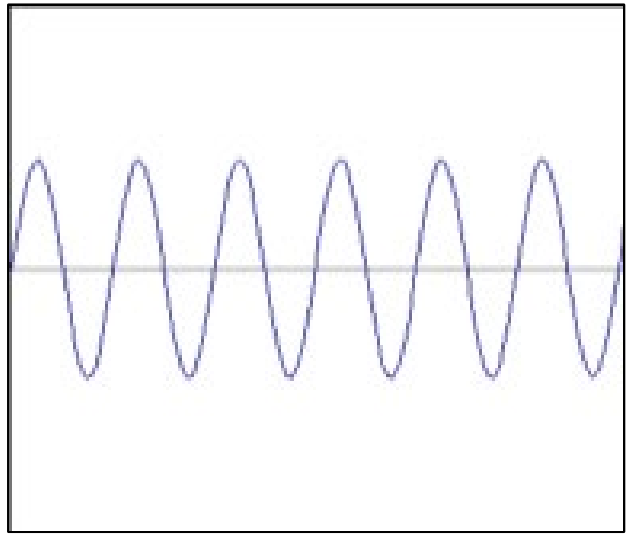
$$f(x) = \sin(2\pi x) + \frac{1}{3}\sin(2\pi 3x)$$

=



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

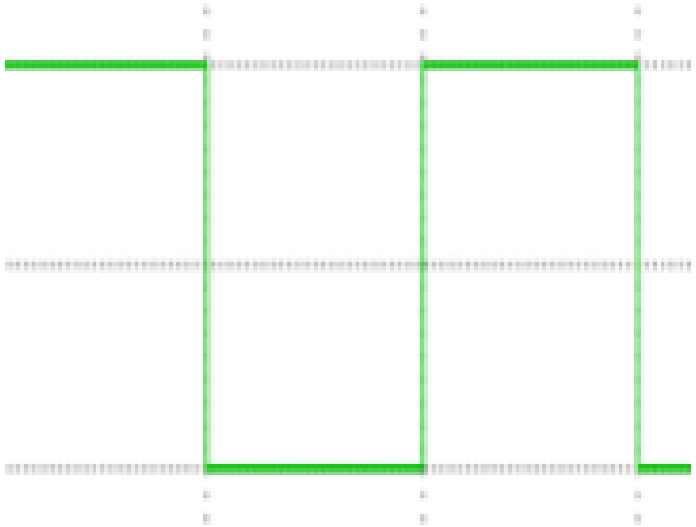
+



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

Examples

How would you generate this function?

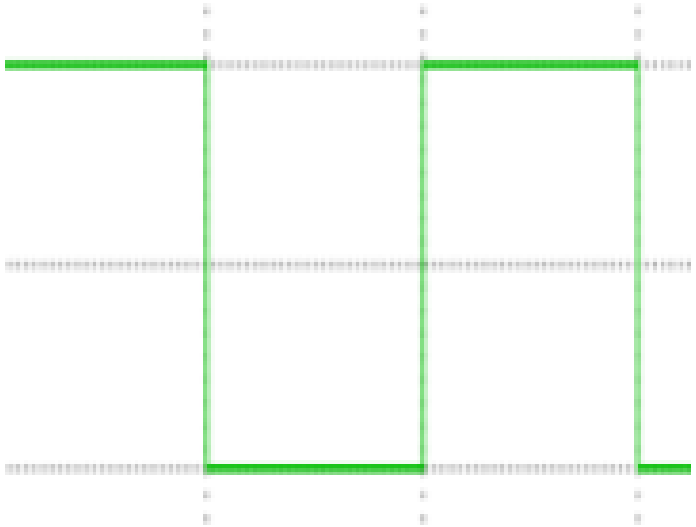


square wave

$= \quad ? \quad + \quad ?$

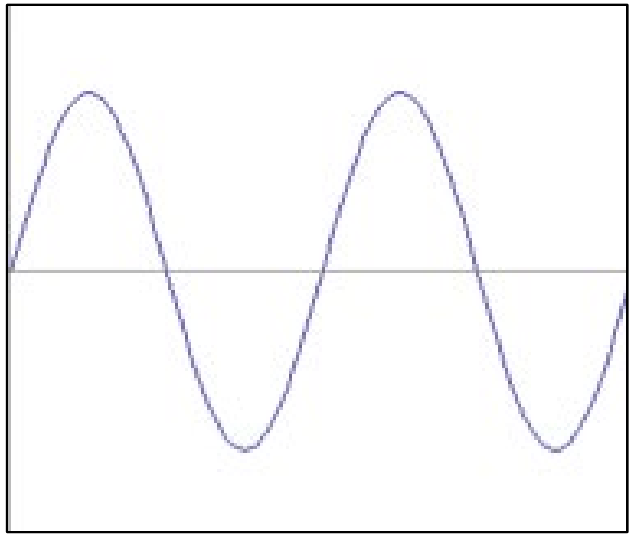
Examples

How would you generate this function?

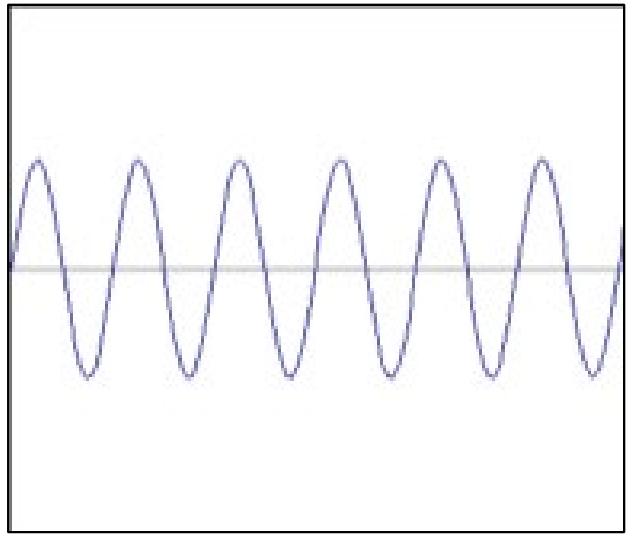


square wave

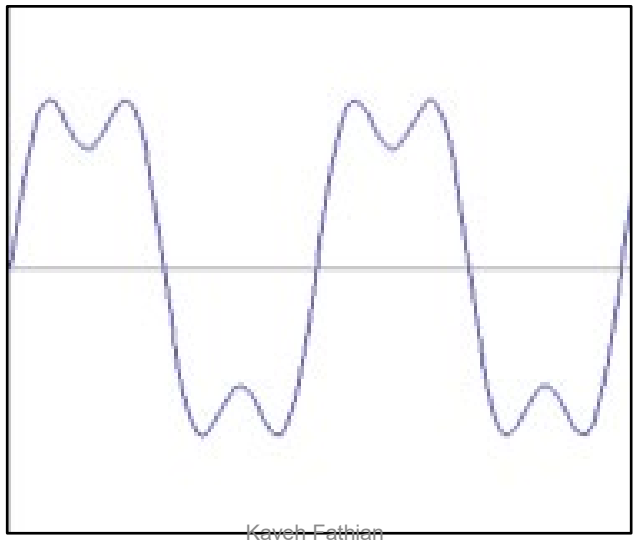
≈



+

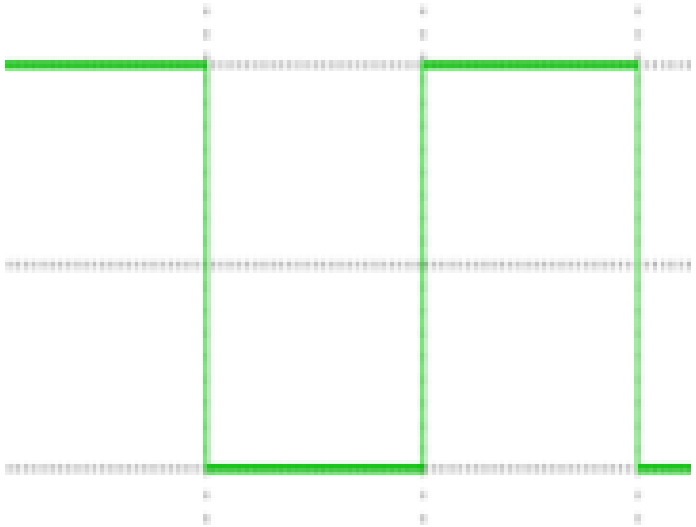


=



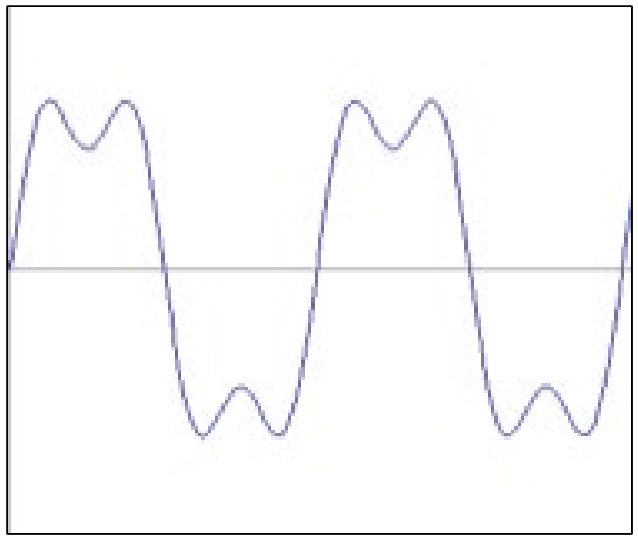
Examples

How would you generate this function?

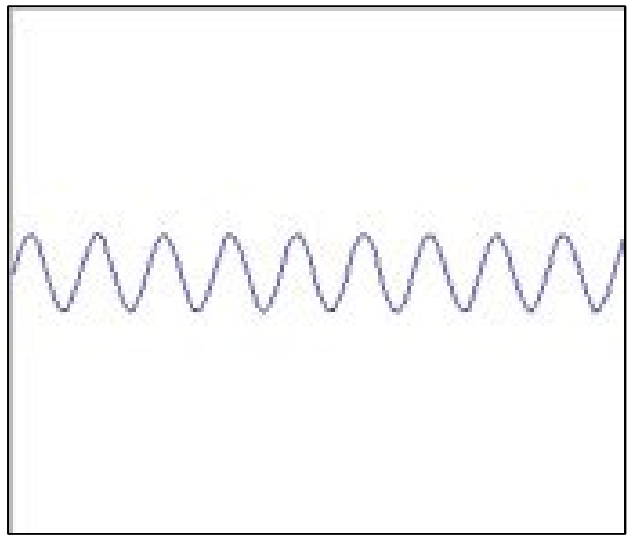


square wave

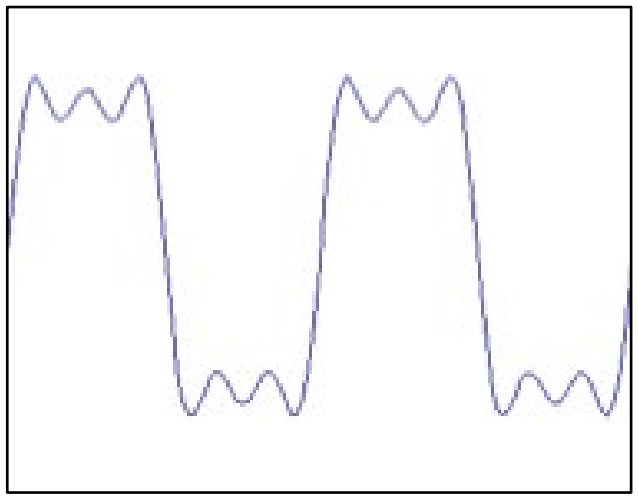
≈



+

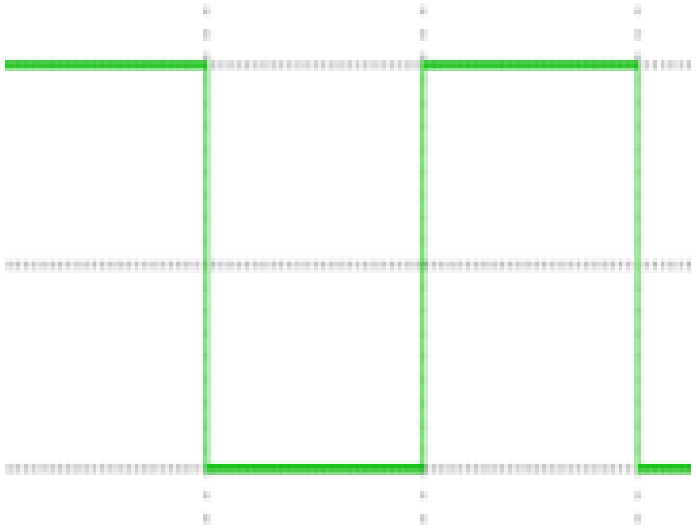


=



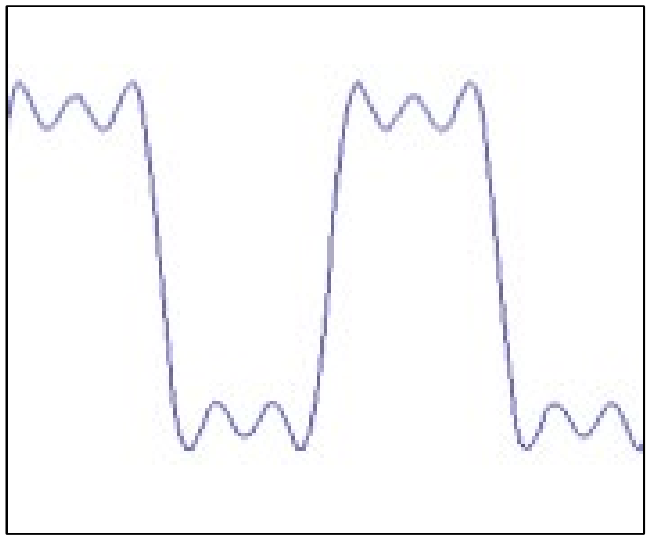
Examples

How would you generate this function?

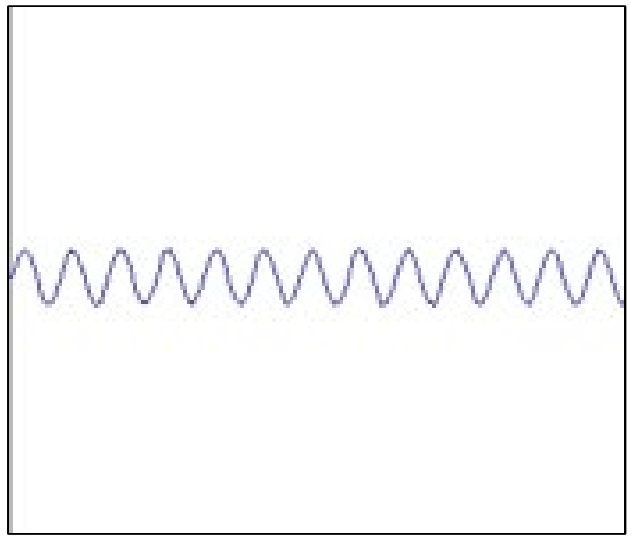


square wave

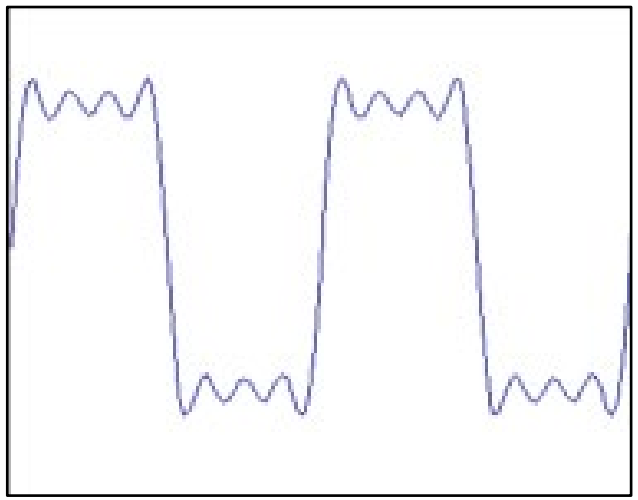
≈



+

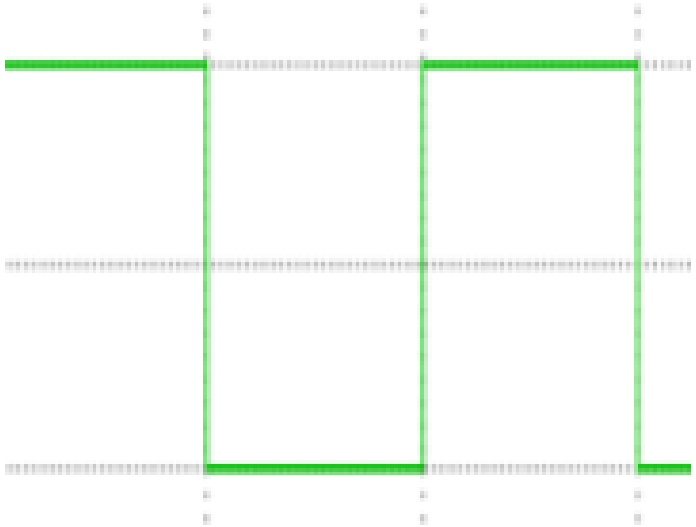


=



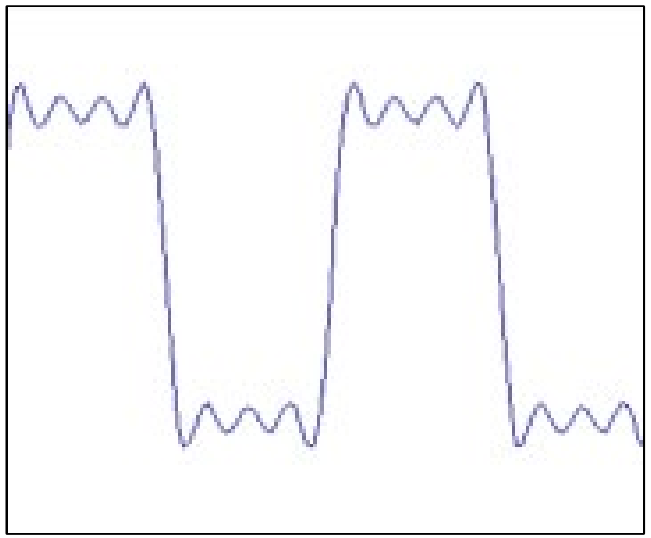
Examples

How would you generate this function?

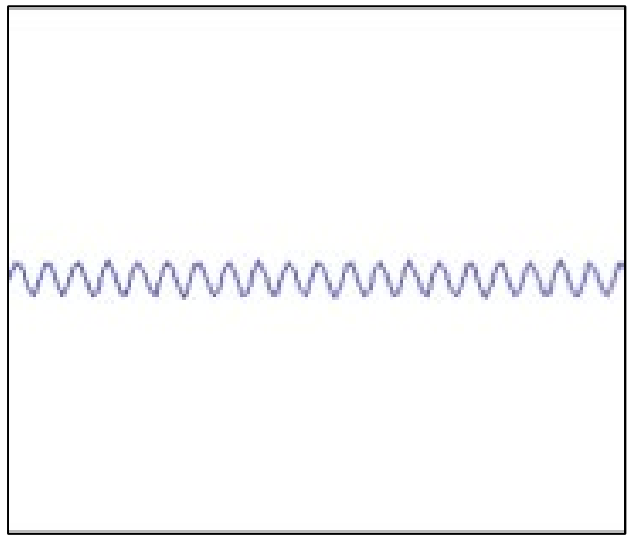


square wave

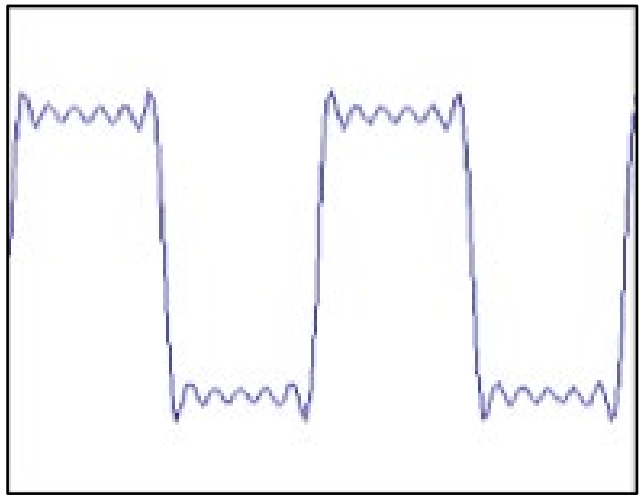
≈



+

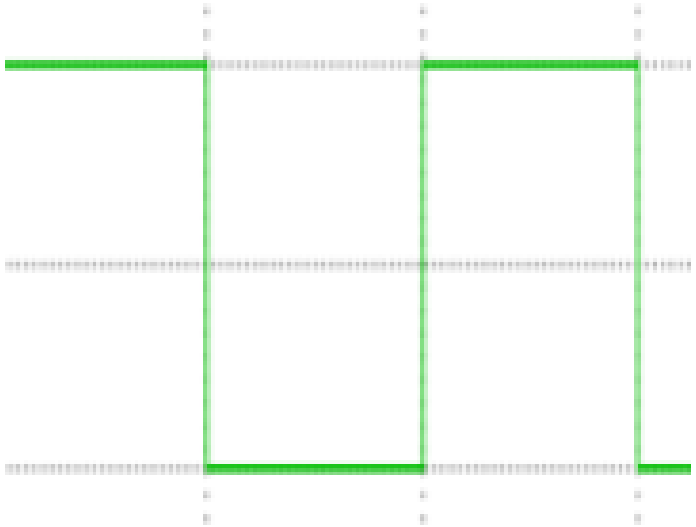


=



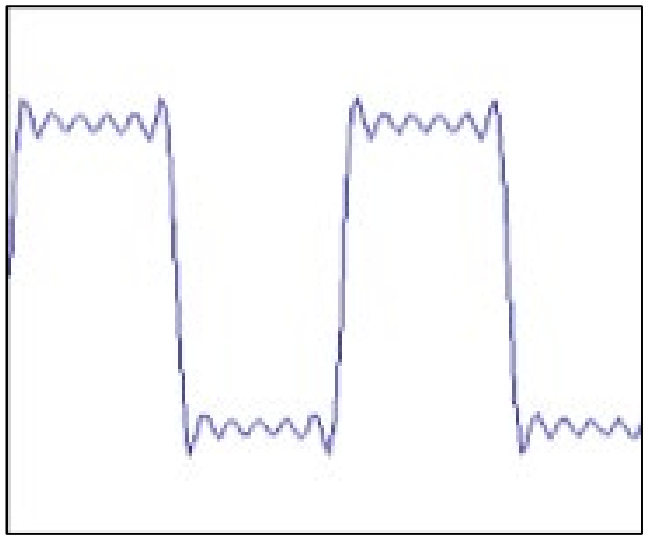
Examples

How would you generate this function?

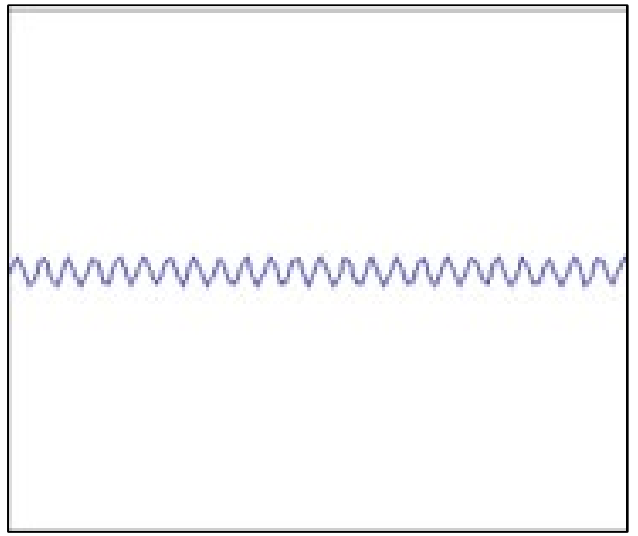


square wave

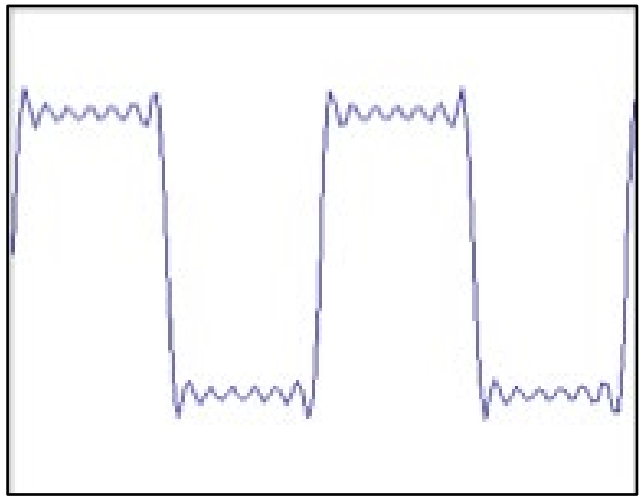
≈



+

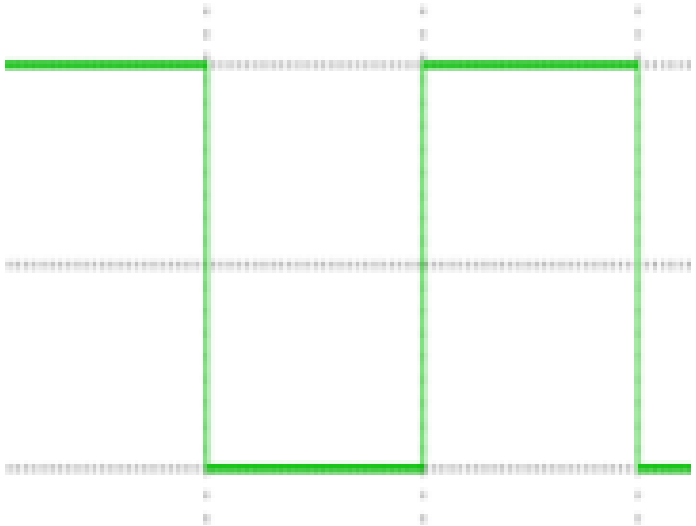


=



How would you express this mathematically?

Examples



=

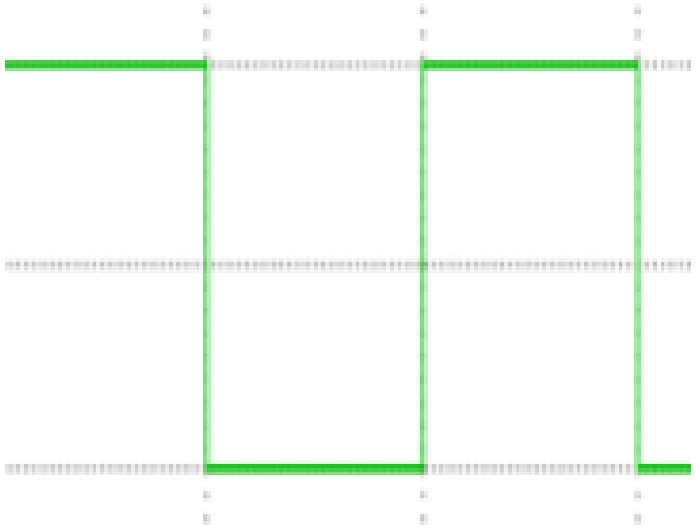
$$A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi k x)$$

square wave

infinite sum of sine waves

How would you visualize this in the frequency domain?

Examples

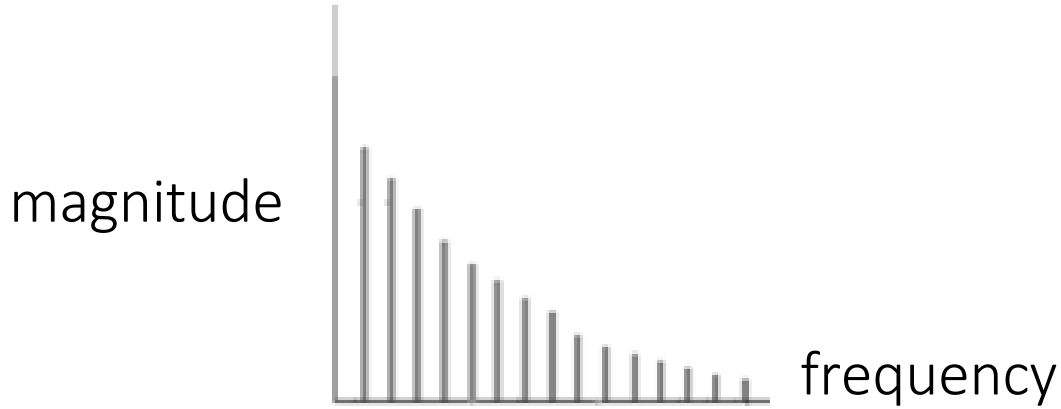


square wave

=

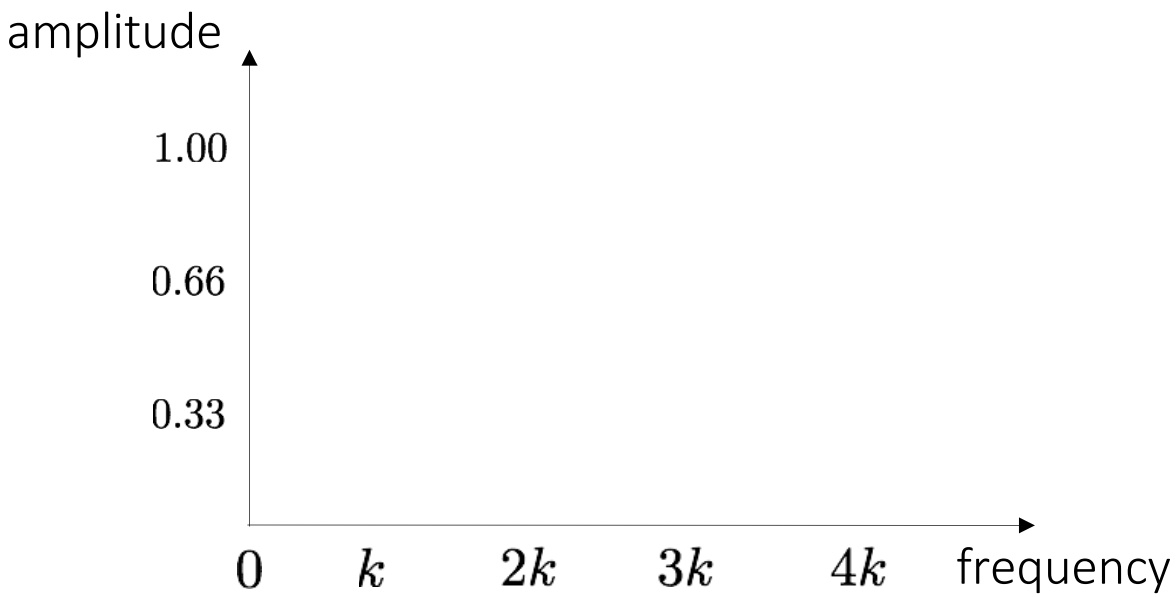
$$A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi k x)$$

infinite sum of sine waves



Frequency domain

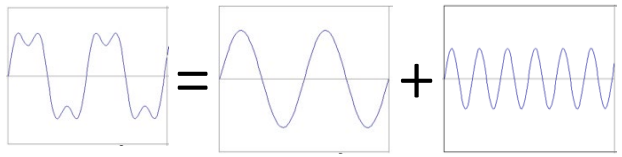
Visualizing the frequency spectrum

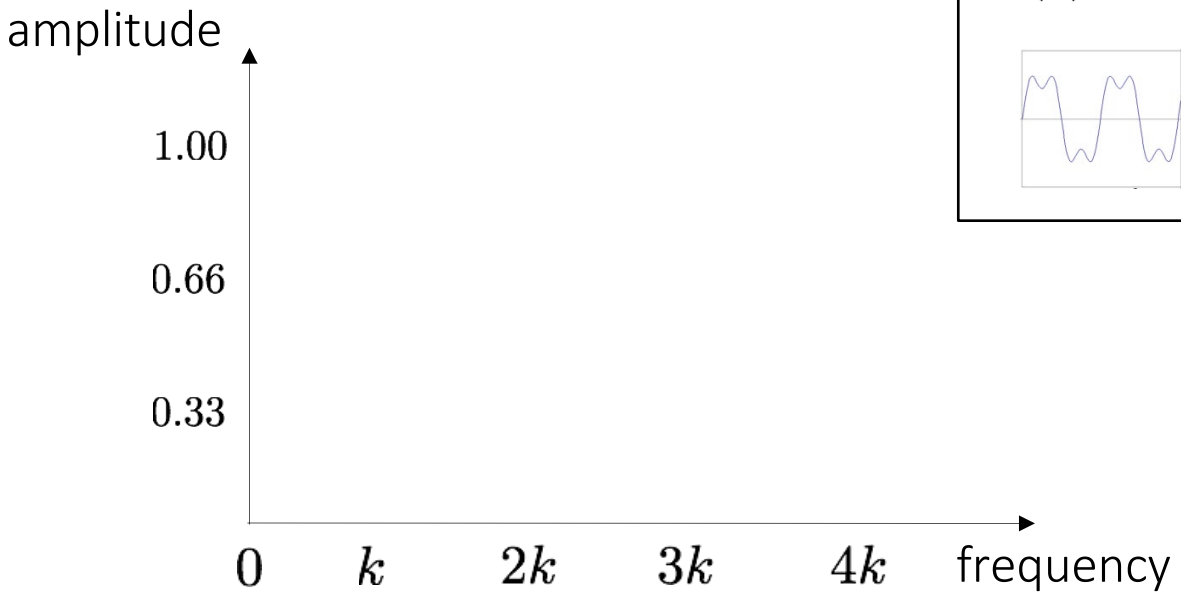


Visualizing the frequency spectrum

Recall the temporal domain visualization

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

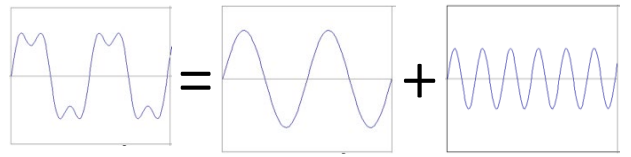


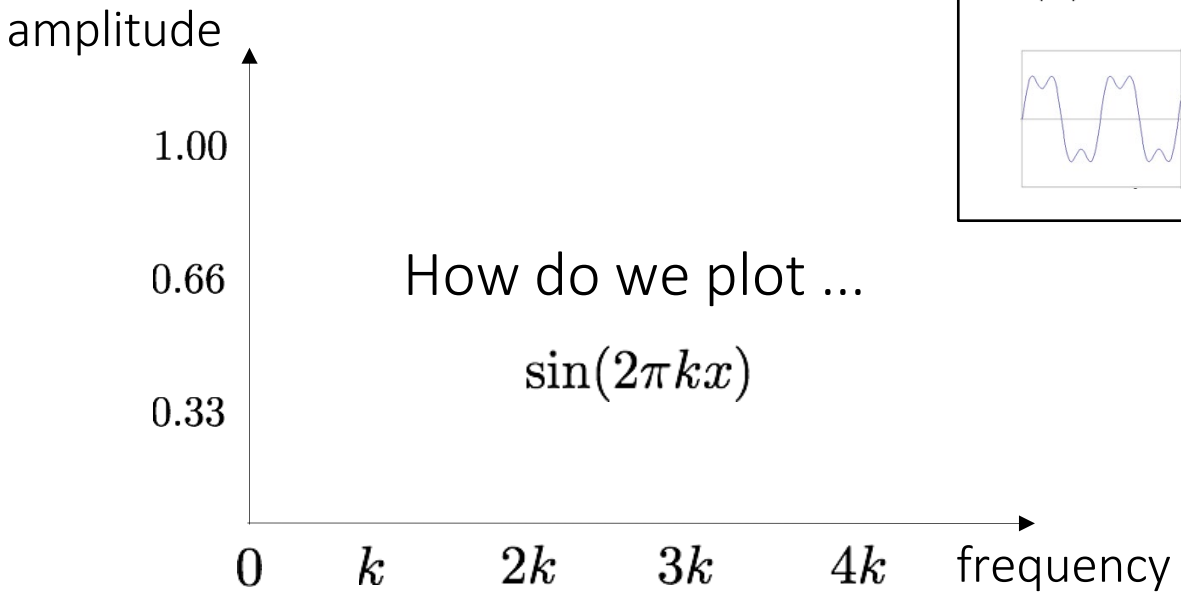


Visualizing the frequency spectrum

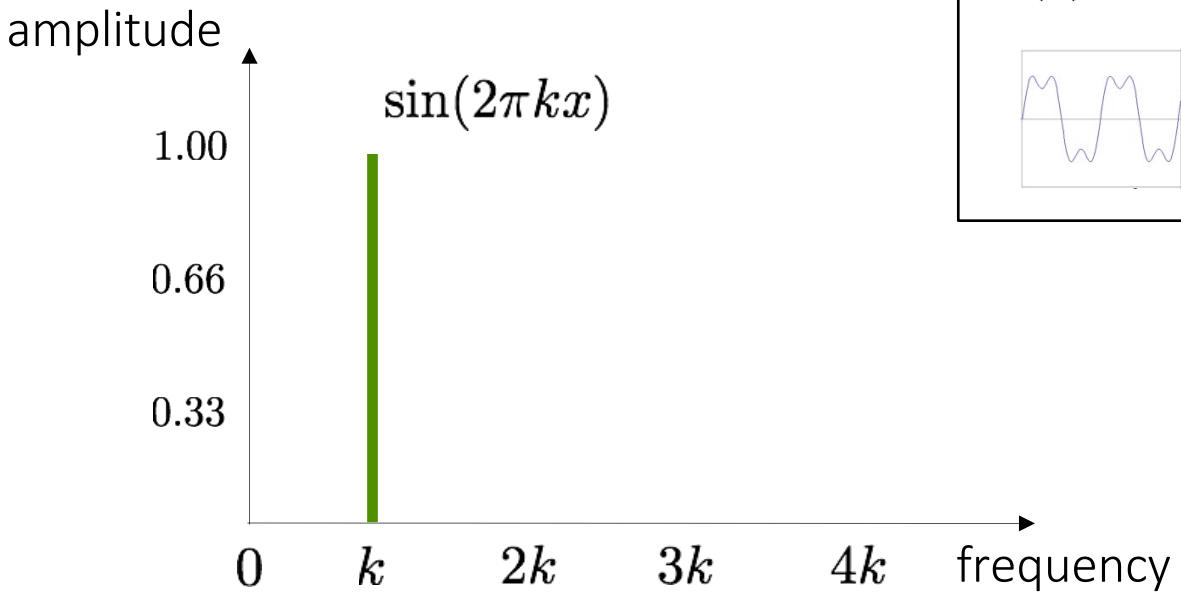
Recall the temporal domain visualization

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



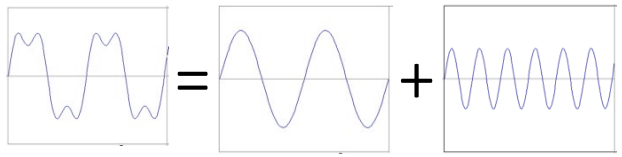


Visualizing the frequency spectrum



Recall the temporal domain visualization

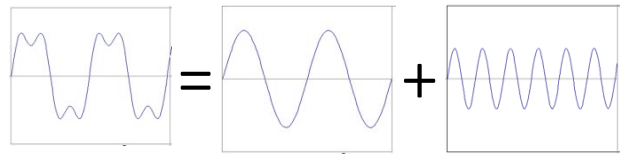
$$f(x) = \sin(2\pi kx) + \frac{1}{3}\sin(2\pi 3kx)$$

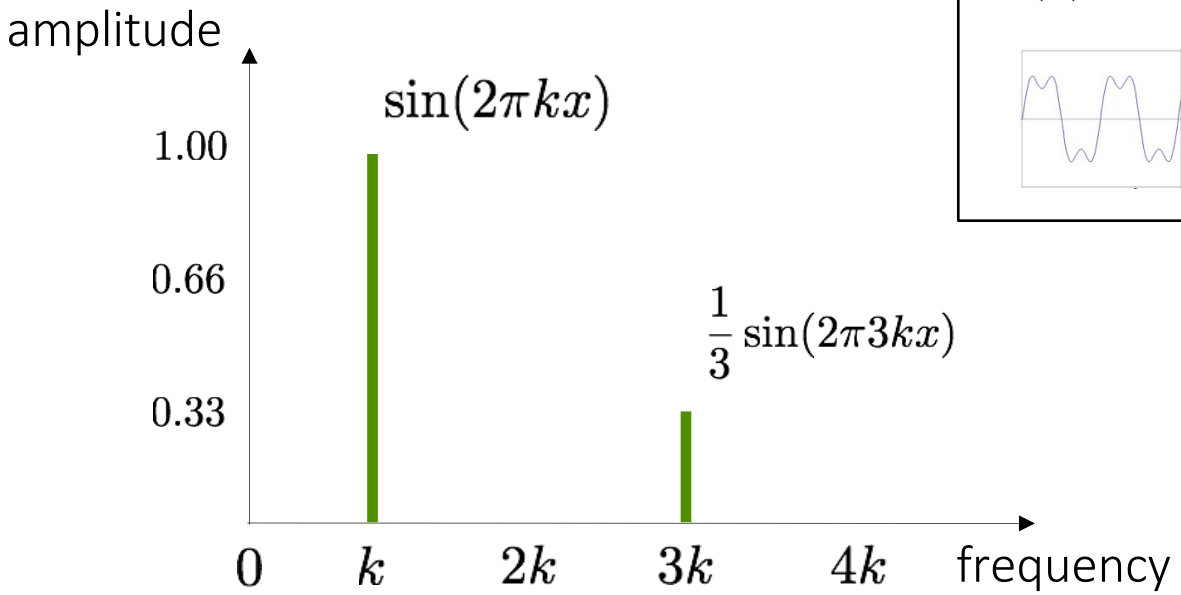


Visualizing the frequency spectrum

Recall the temporal domain visualization

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

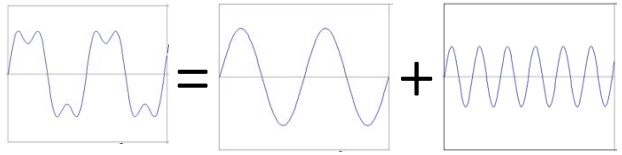




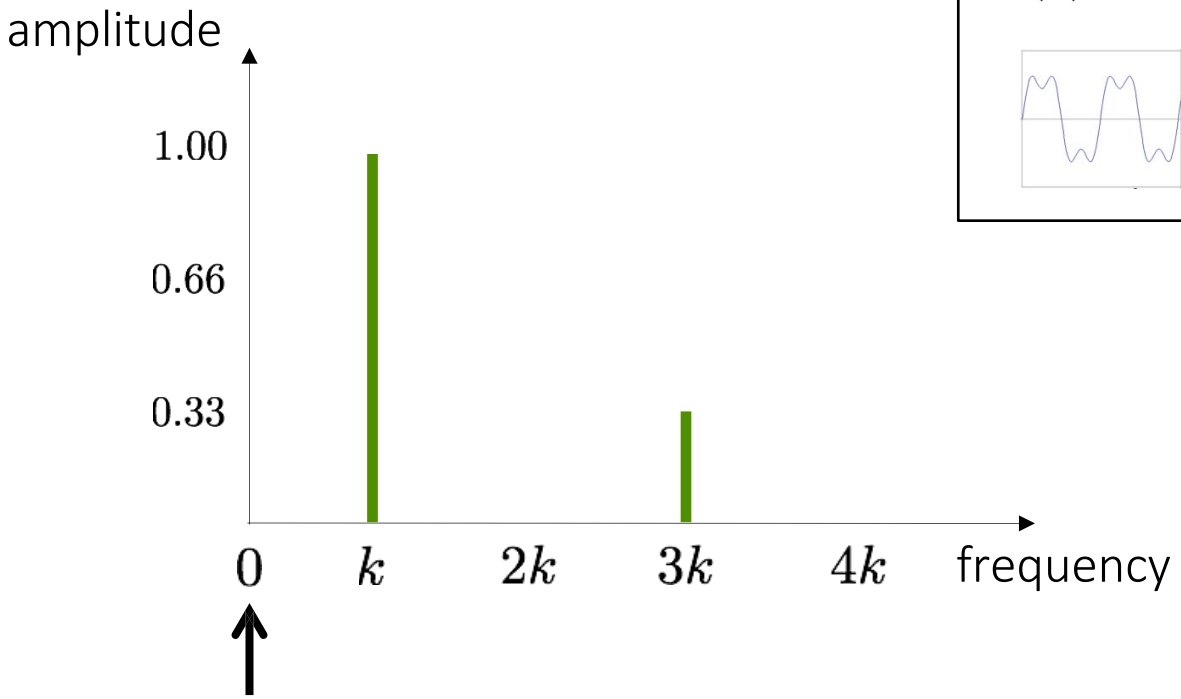
Visualizing the frequency spectrum

Recall the temporal domain visualization

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



not visualizing the symmetric negative part



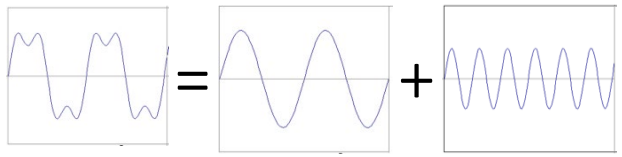
What is at zero frequency?

Need to understand this to understand the 2D version!

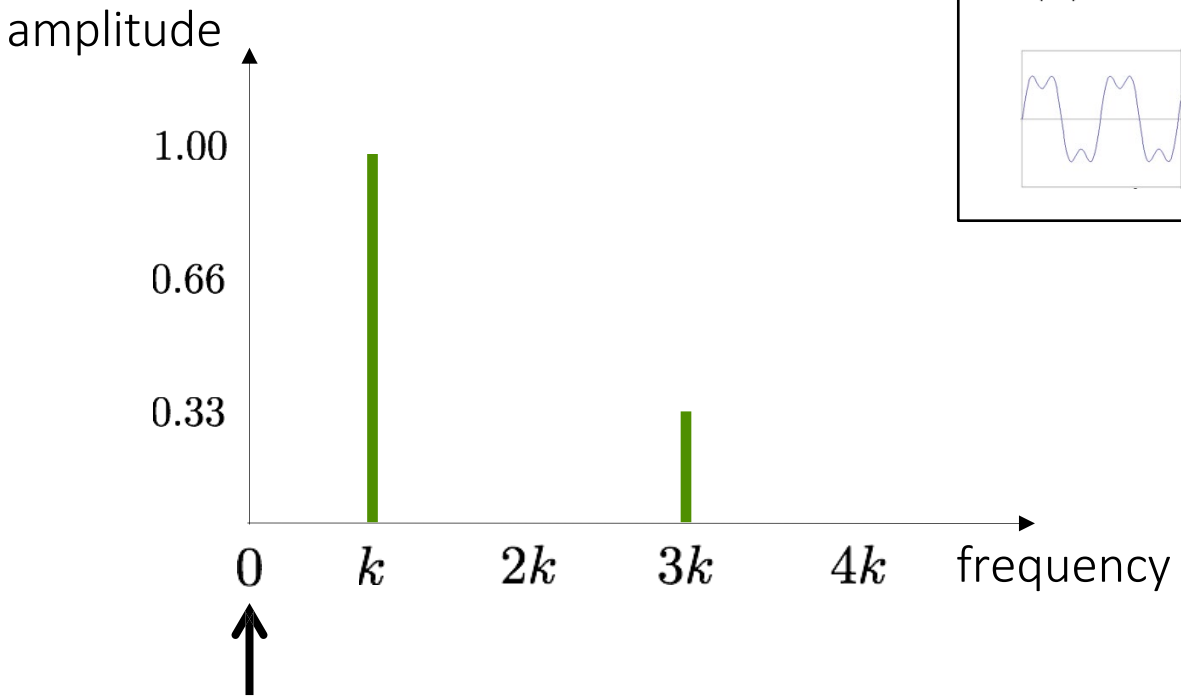
Visualizing the frequency spectrum

Recall the temporal domain visualization

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



not visualizing the symmetric negative part



signal average (zero for a sine wave with no offset)

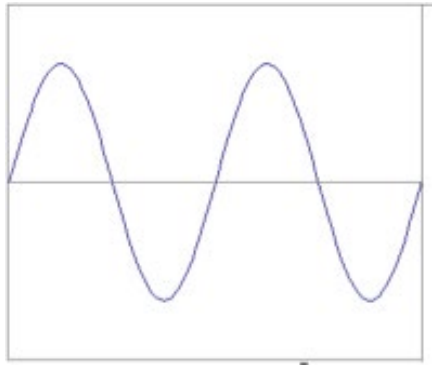
Need to understand this to understand the 2D version!

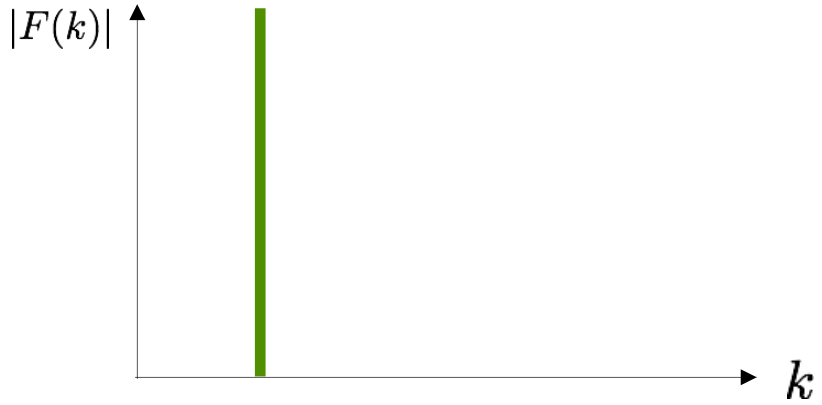
Examples

Spatial domain visualization

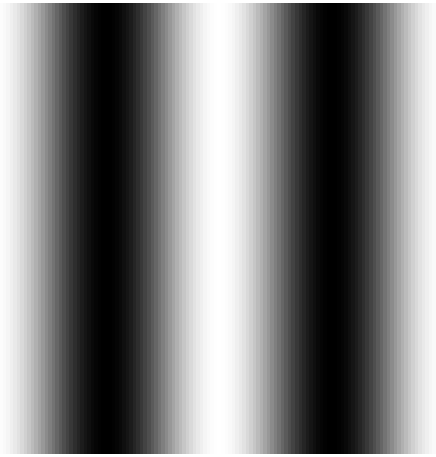
Frequency domain visualization

1D





2D



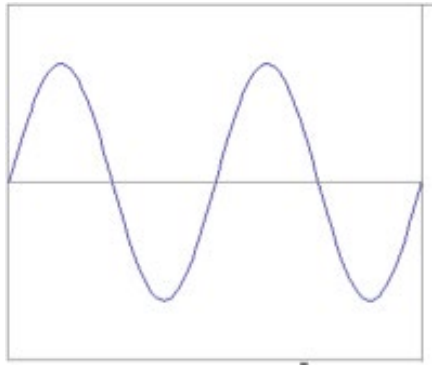
?

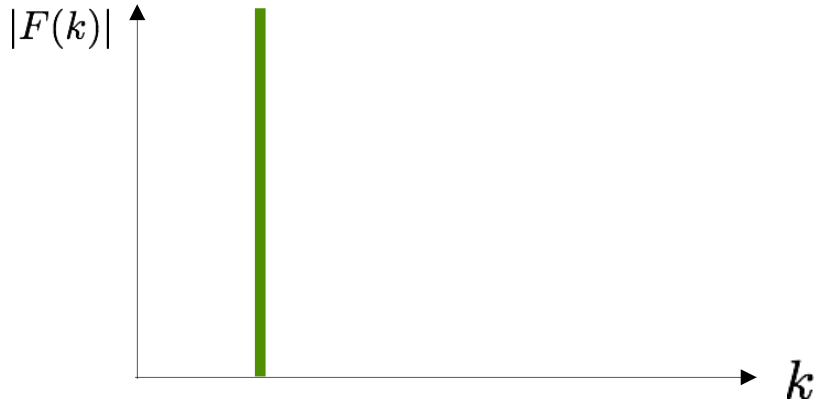
Examples

Spatial domain visualization

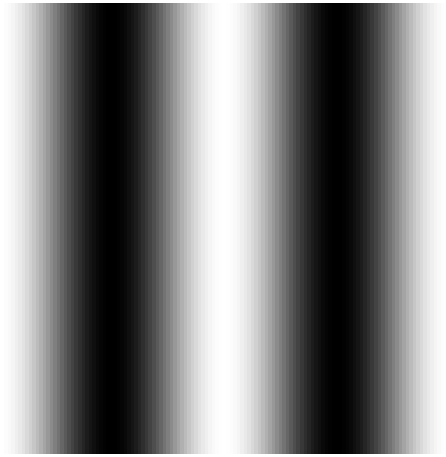
Frequency domain visualization

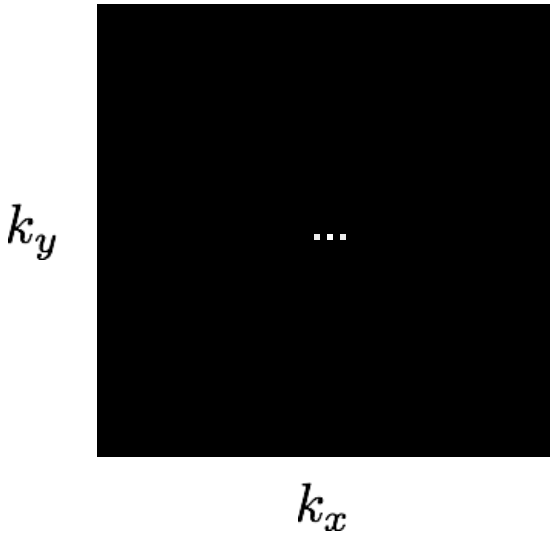
1D





2D

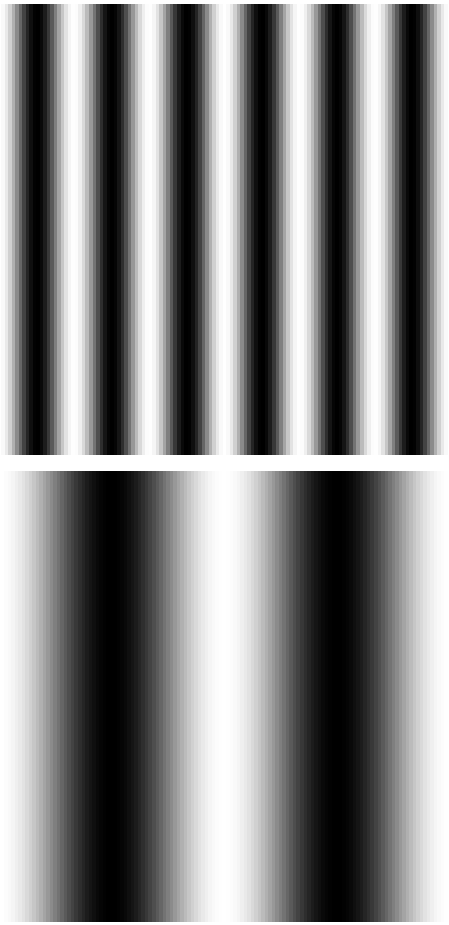




What do the three dots correspond to?

Examples

Spatial domain visualization

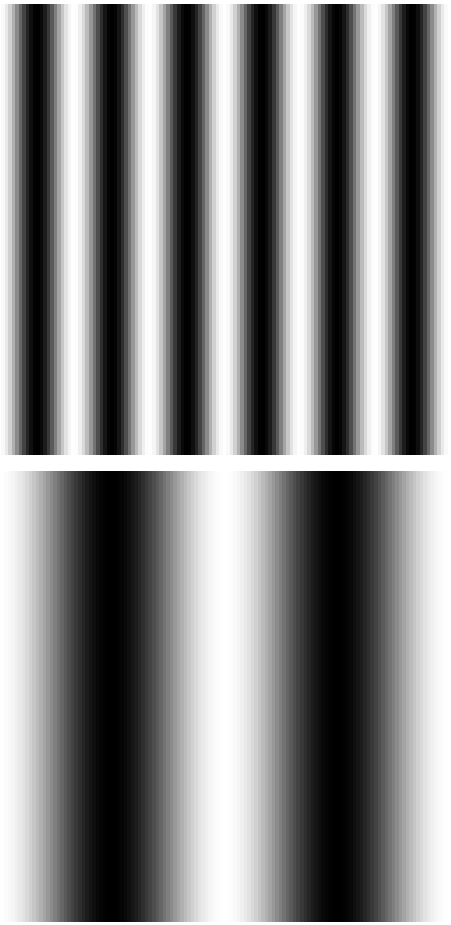


Frequency domain visualization

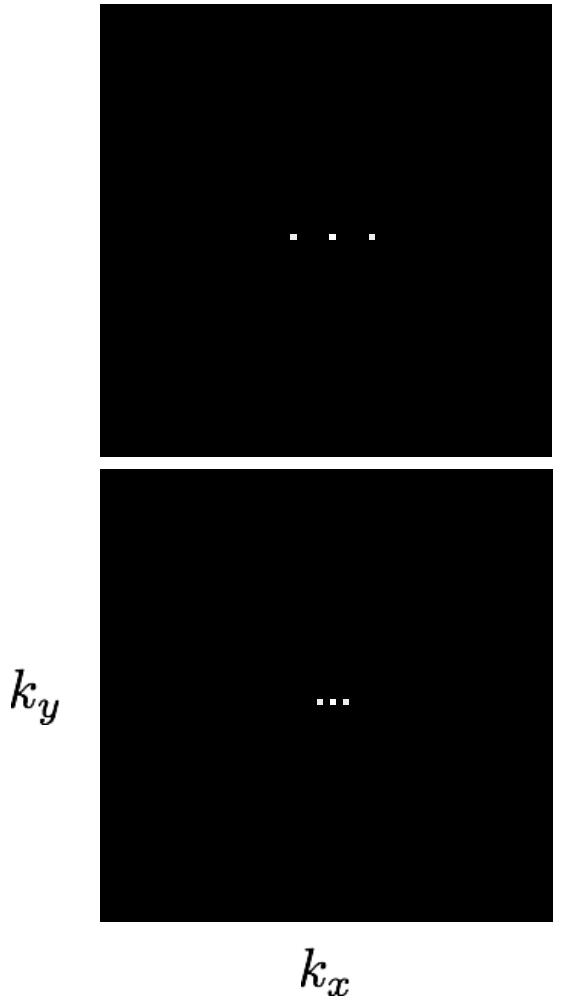
?

Examples

Spatial domain visualization

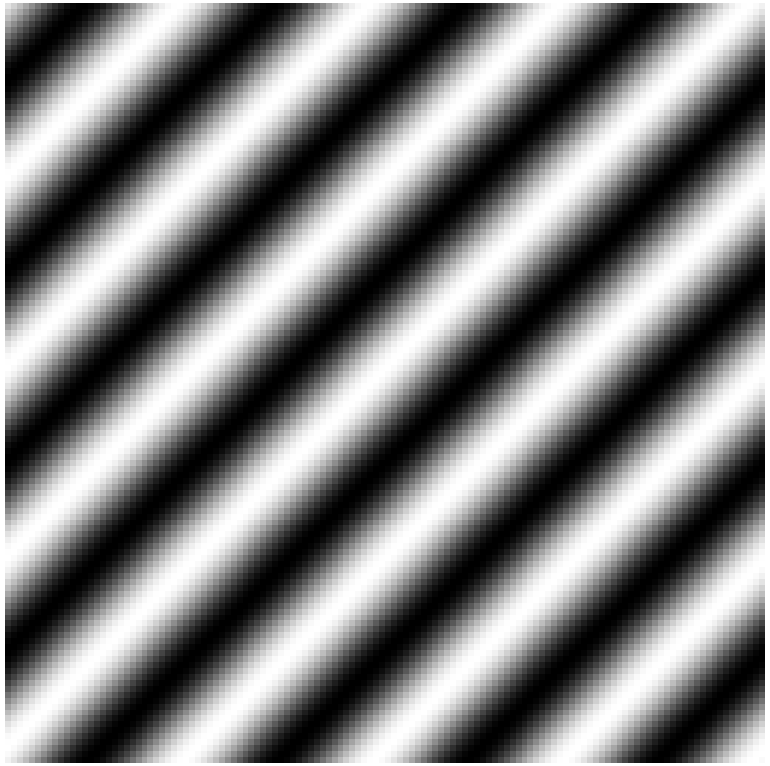


Frequency domain visualization



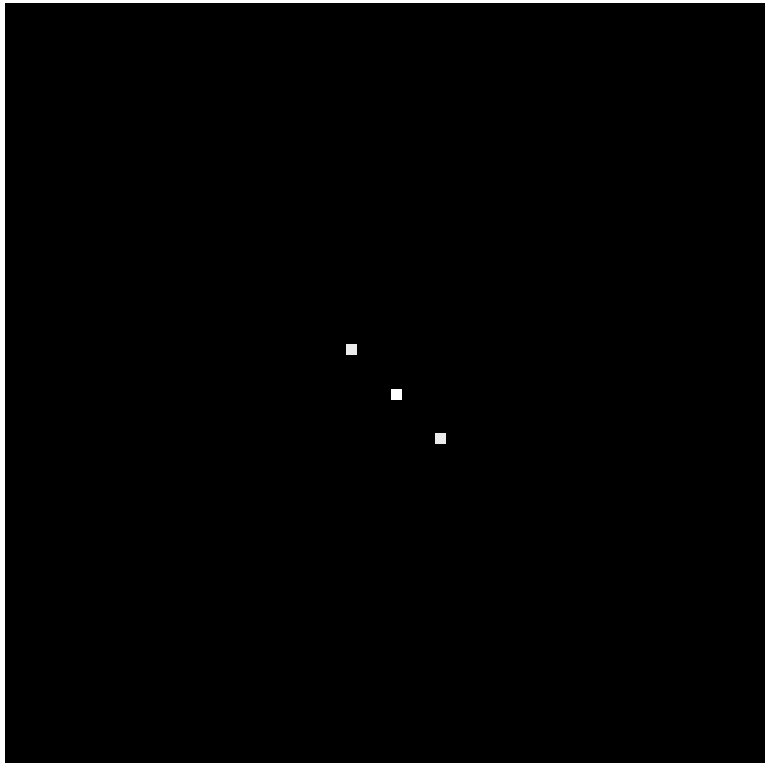
Examples

How would you generate this image with sine waves?



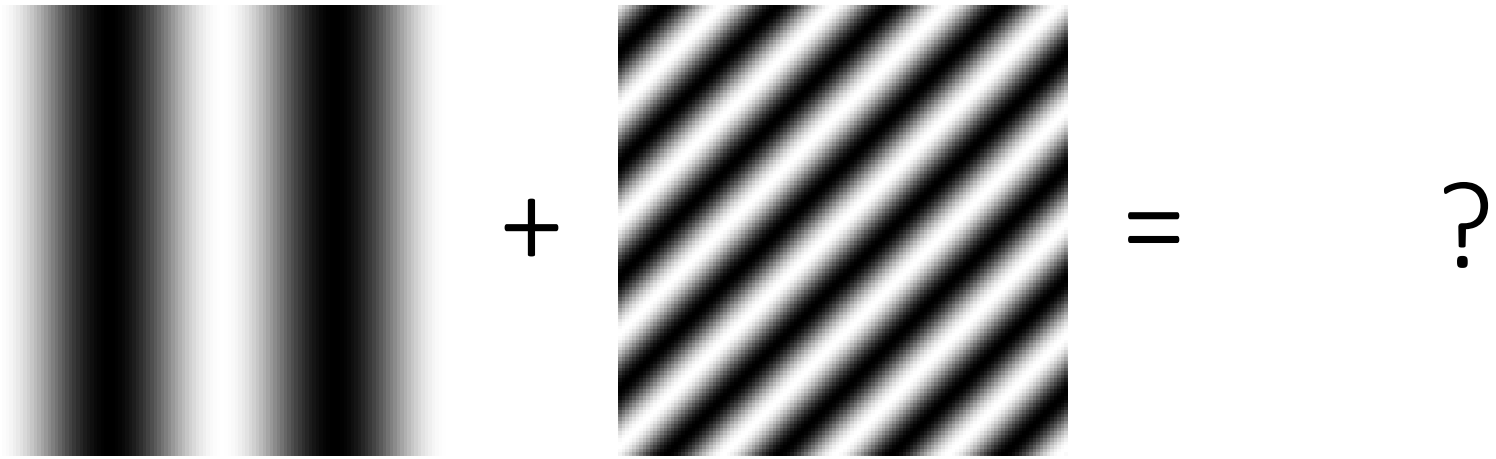
Examples

How would you generate this image with sine waves?

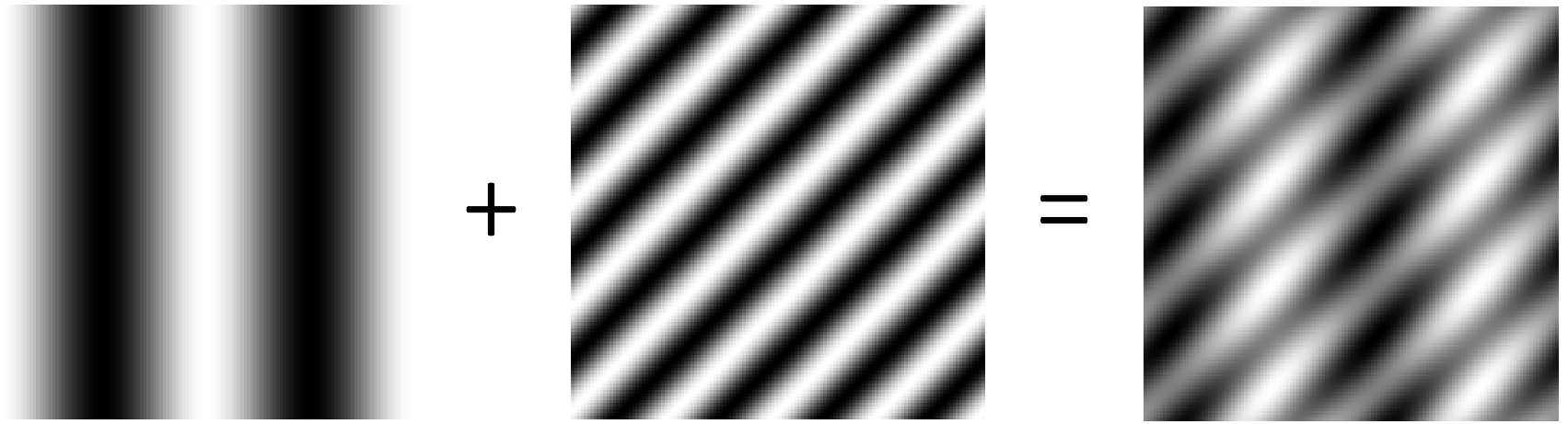


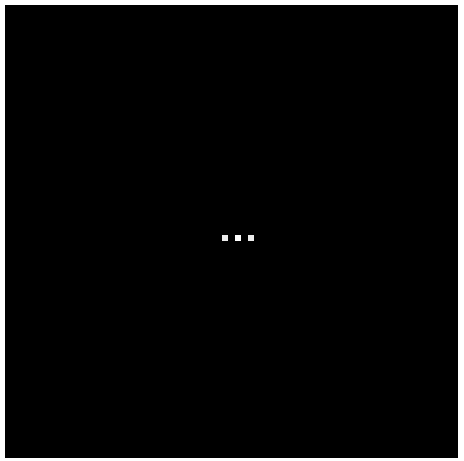
Has both an x and y components

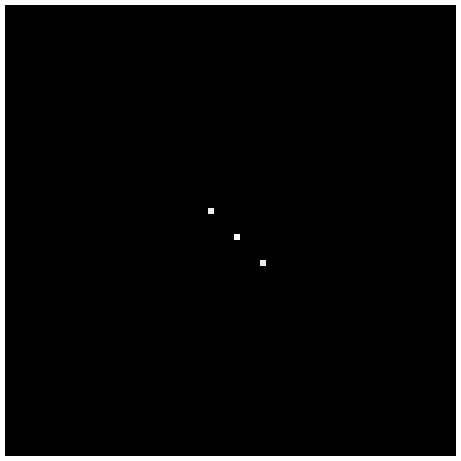
Examples



Examples

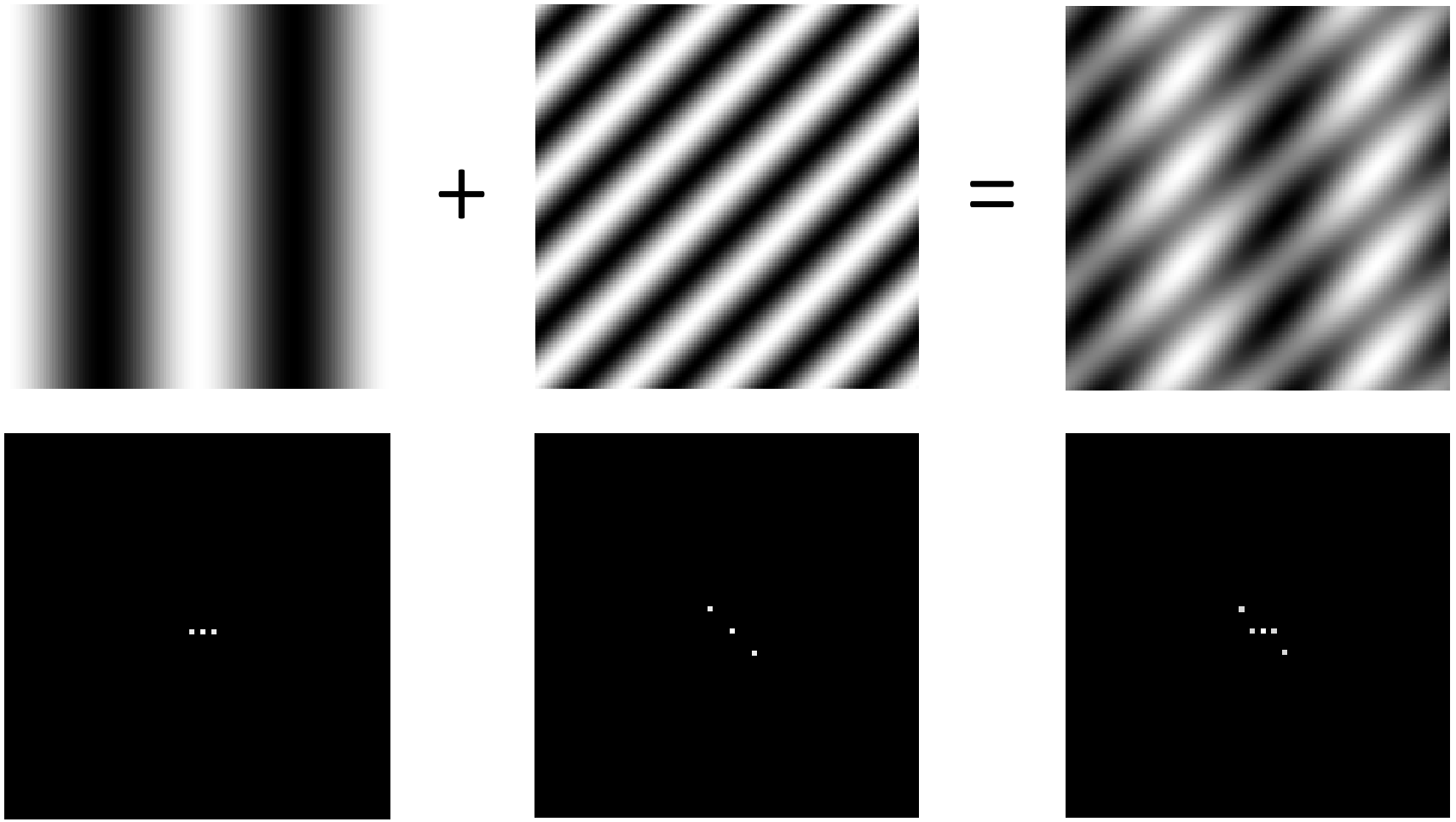






?

Examples



Basic building block

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

amplitude

sinusoid

angular frequency

variable

phase

What about non-periodic signals?

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

Fourier Transform

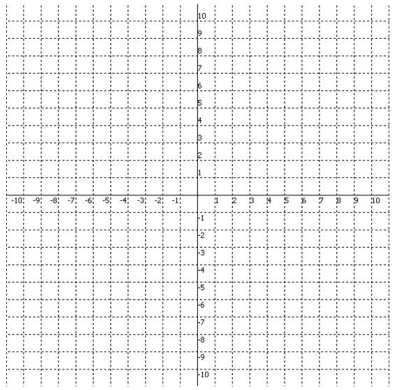
Recalling some basics

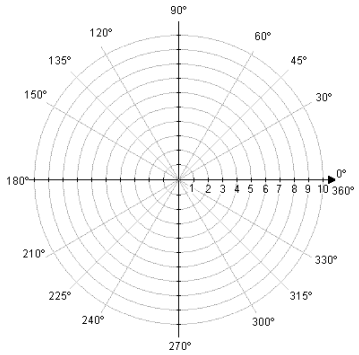
Complex numbers have two parts:

rectangular coordinates

$$R + jI$$

what's this? what's this?





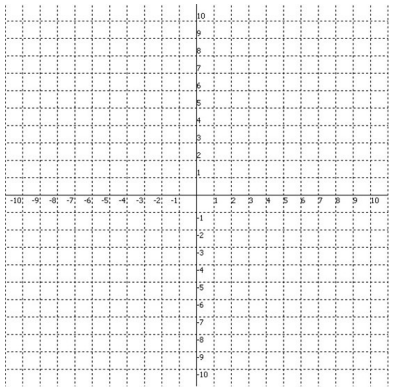
Recalling some basics

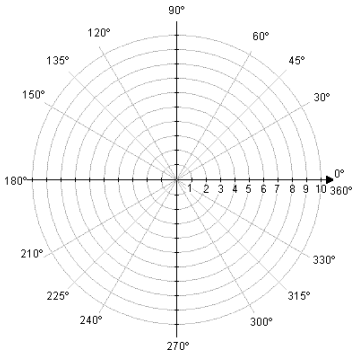
Complex numbers have two parts:

rectangular coordinates

$$R + jI$$

real imaginary





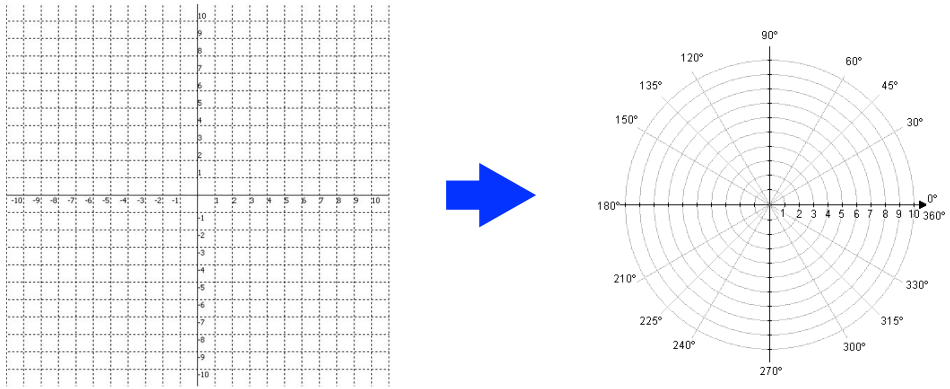
Recalling some basics

Complex numbers have two parts:

rectangular coordinates

$$R + jI$$

real imaginary



polar transform

Alternative reparameterization:

polar coordinates

$$r(\cos\theta + j\sin\theta)$$

how do we compute these?

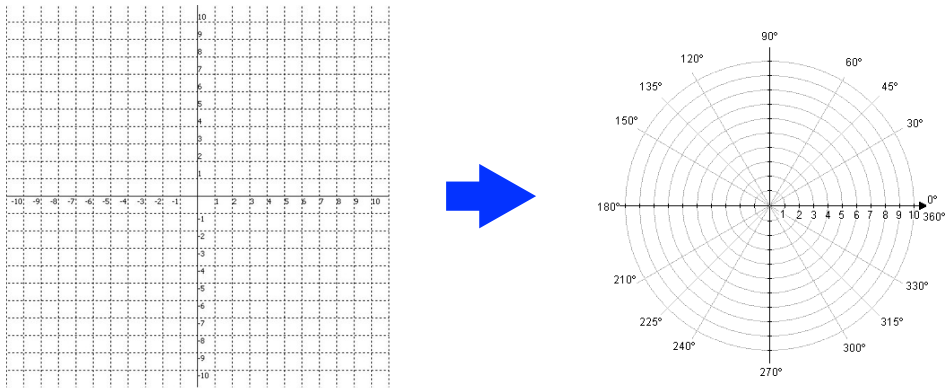
Recalling some basics

Complex numbers have two parts:

rectangular coordinates

$$R + jI$$

real imaginary



polar transform

Alternative reparameterization:

polar coordinates

$$r(\cos\theta + j\sin\theta)$$

polar transform

$$\theta = \tan^{-1}(\frac{I}{R}) \quad r = \sqrt{R^2 + I^2}$$

Recalling some basics

Complex numbers have two parts:

rectangular coordinates

$$R + jI$$

real imaginary

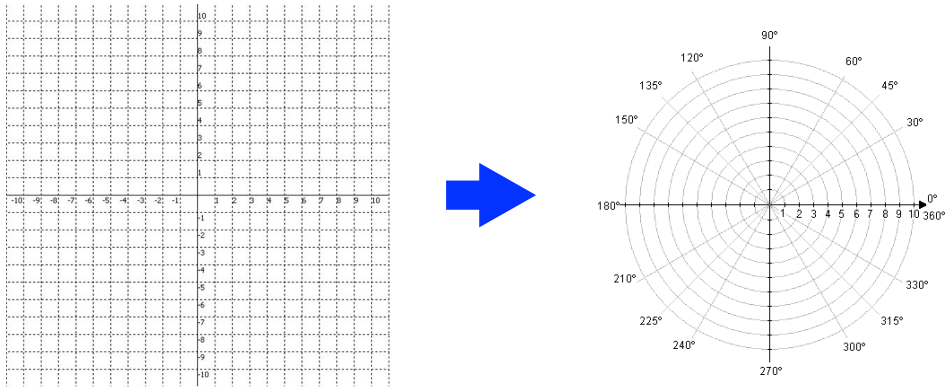
Alternative reparameterization:

polar coordinates

$$r(\cos\theta + j\sin\theta)$$

polar transform

$$\theta = \tan^{-1}(\frac{I}{R}) \quad r = \sqrt{R^2 + I^2}$$



polar transform

How do you write these in exponential form?

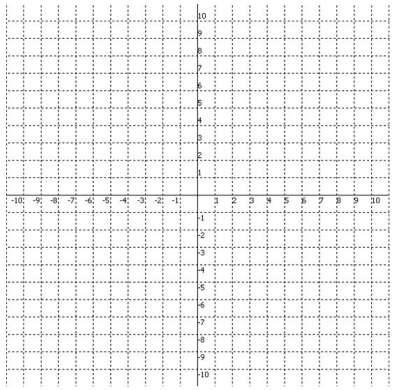
Recalling some basics

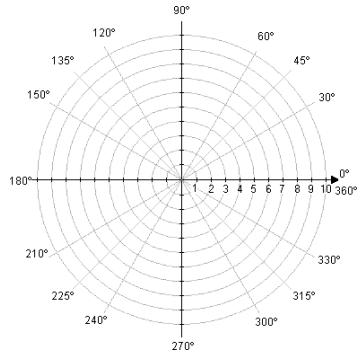
Complex numbers have two parts:

rectangular coordinates

$$R + jI$$

real imaginary





Alternative reparameterization:

polar coordinates

$$r(\cos\theta + j\sin\theta)$$

polar transform

$$\theta = \tan^{-1}(\frac{I}{R}) \quad r = \sqrt{R^2 + I^2}$$

or equivalently

$$re^{j\theta}$$

how did we get this?

exponential form

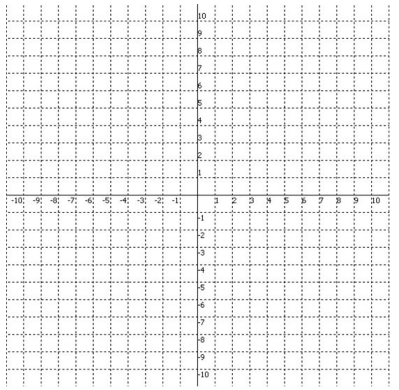
Recalling some basics

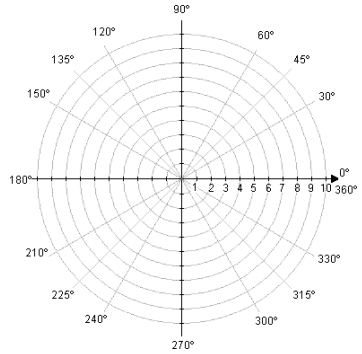
Complex numbers have two parts:

rectangular coordinates

$$R + jI$$

real imaginary





Alternative reparameterization:

polar coordinates

$$r(\cos\theta + j\sin\theta)$$

polar transform

$$\theta = \tan^{-1}(\frac{I}{R}) \quad r = \sqrt{R^2 + I^2}$$

or equivalently

$$re^{j\theta}$$

Euler's formula

$$e^{j\theta} = \cos\theta + j\sin\theta$$

exponential form

This will help us understand the Fourier transform equations

Fourier transform

	Fourier transform	inverse Fourier transform
continuous	$F(k) = \int_{\infty}^{-\infty} f(x) e^{-j2\pi kx} dx$	$f(x) = \int_{\infty}^{-\infty} F(k) e^{j2\pi kx} dk$
discrete	$F(k) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-j2\pi kx/N}$ $k = 0, 1, 2, \ldots, N-1$	$f(x) = \sum_{k=0}^{N-1} F(k) e^{j2\pi kx/N}$ $x = 0, 1, 2, \ldots, N-1$

Where is the connection to the ‘summation of sine waves’ idea?

Fourier transform

	Fourier transform	inverse Fourier transform
continuous	$F(k) = \int_{\infty}^{-\infty} f(x) e^{-j2\pi kx} dx$	$f(x) = \int_{\infty}^{-\infty} F(k) e^{j2\pi kx} dk$
discrete	$F(k) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-j2\pi kx/N}$ $k = 0, 1, 2, \ldots, N-1$	$f(x) = \sum_{k=0}^{N-1} F(k) e^{j2\pi kx/N}$ $x = 0, 1, 2, \ldots, N-1$

Where is the connection to the ‘summation of sine waves’ idea?

Fourier transform

Where is the connection to the 'summation of sine waves' idea?

$$f(x) = \sum_{k=0}^{N-1} F(k) e^{j 2\pi k x / N}$$

Euler's formula

$$e^{j\theta} = \cos\theta + j \sin\theta$$

sum over frequencies

$$f(x) = \sum_{k=0}^{N-1} F(k) \left\{ \cos(2\pi k x) + j \sin(2\pi k x) \right\}$$

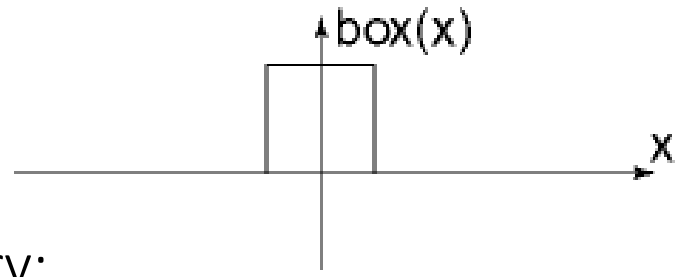
scaling parameter

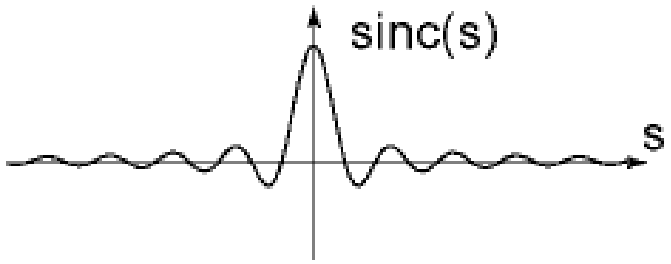
wave components

Fourier transform pairs

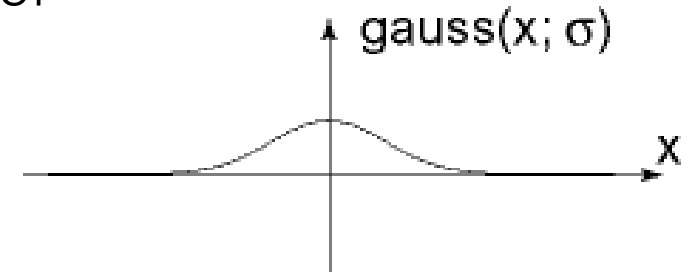
spatial domain

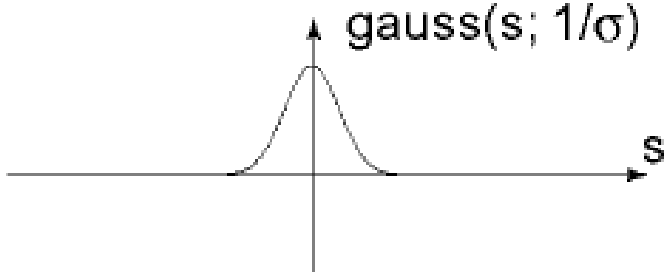
frequency domain

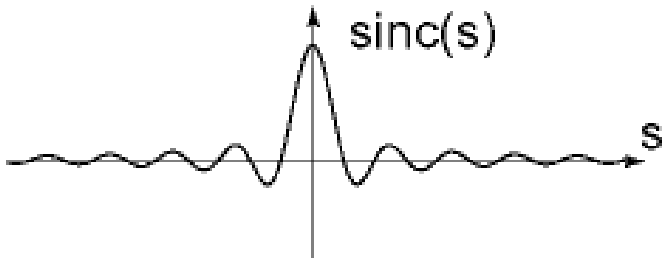


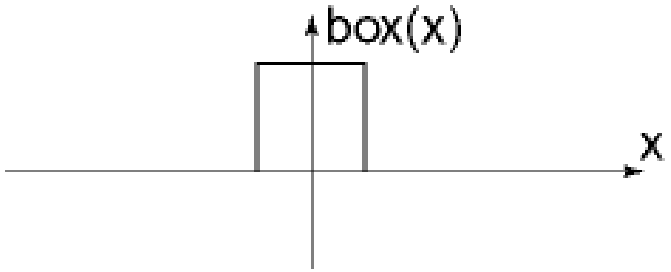


Note the symmetry:
duality property of
Fourier transform









Fourier transform for images

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:

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

Computing the discrete Fourier transform (DFT)

$$F(k) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-j2\pi kx/N}$$ is just a matrix multiplication:

$$\boldsymbol{F = W f}$$

$$\begin{bmatrix} F(0) \\ F(1) \\ F(2) \\ F(3) \\ \vdots \\ F(N-1) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & W^3 & \cdots & W^{N-1} \\ W^0 & W^2 & W^4 & W^6 & \cdots & W^{N-2} \\ W^0 & W^3 & W^6 & W^9 & \cdots & W^{N-3} \\ \vdots & & & & \ddots & \vdots \\ W^0 & W^{N-1} & W^{N-2} & W^{N-3} & \cdots & W^1 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \\ \vdots \\ f(N-1) \end{bmatrix} \qquad W = e^{-j2\pi/N}$$

In practice this is implemented using the *fast Fourier transform* (FFT) algorithm.

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:

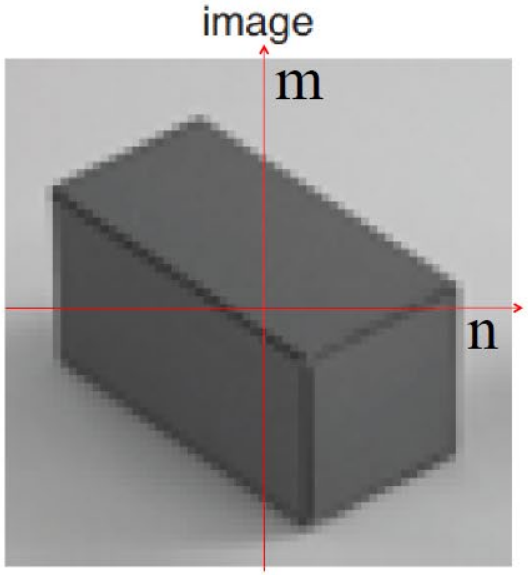
$$F[u,v] = A[u,v]\exp(j\theta[u,v])$$

Amplitude

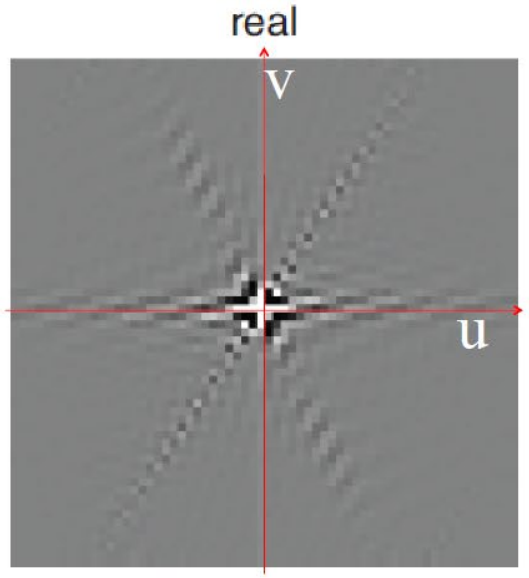
Phase

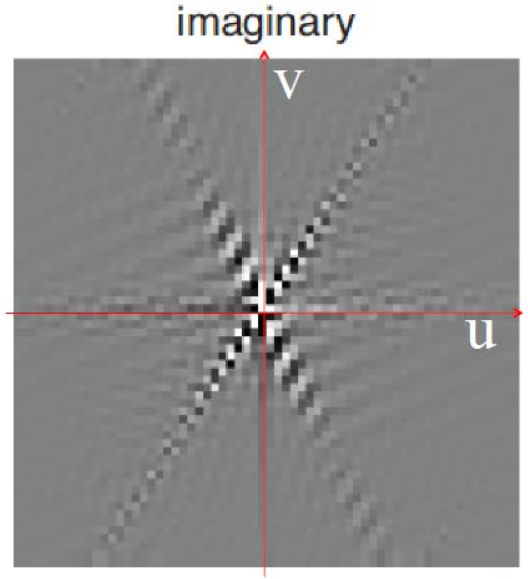
Visualizing the image Fourier transform

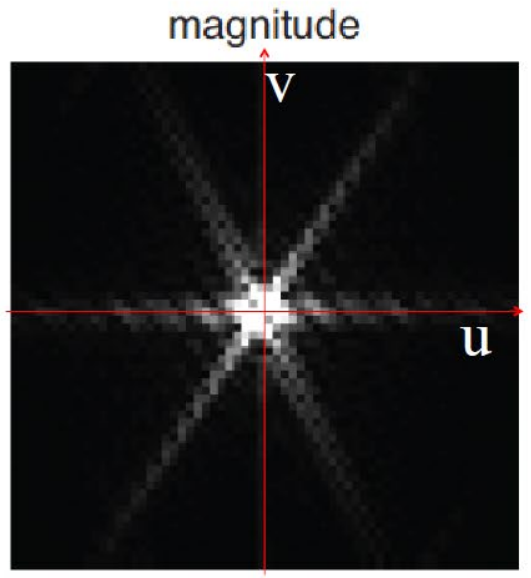
$f[n, m]$

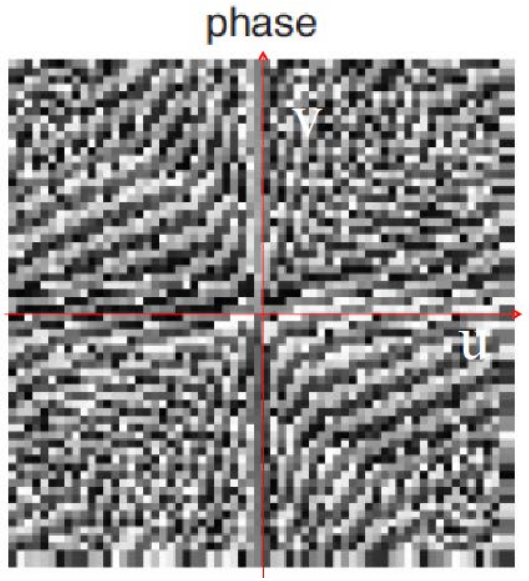


$F[u, v]$

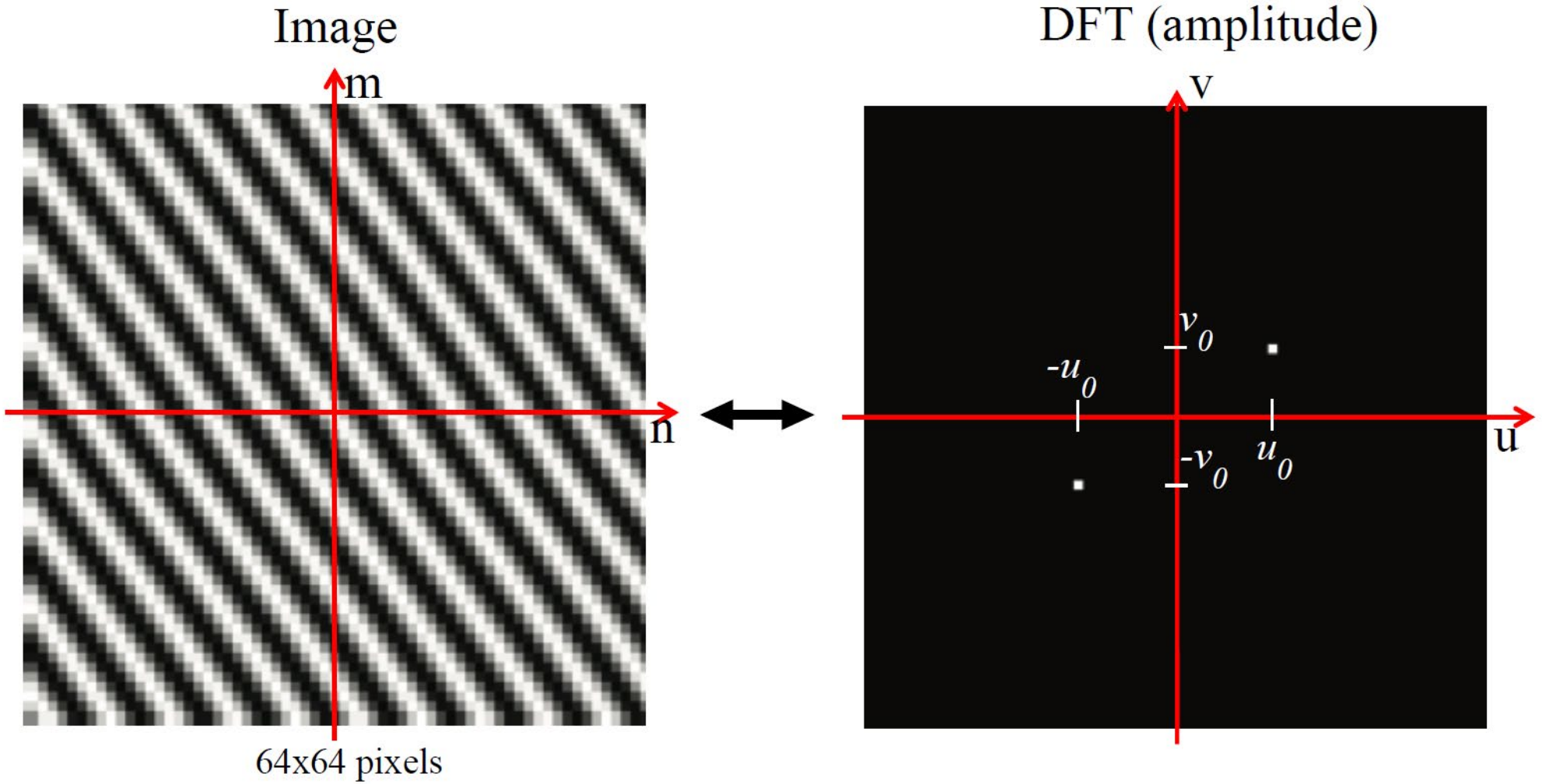






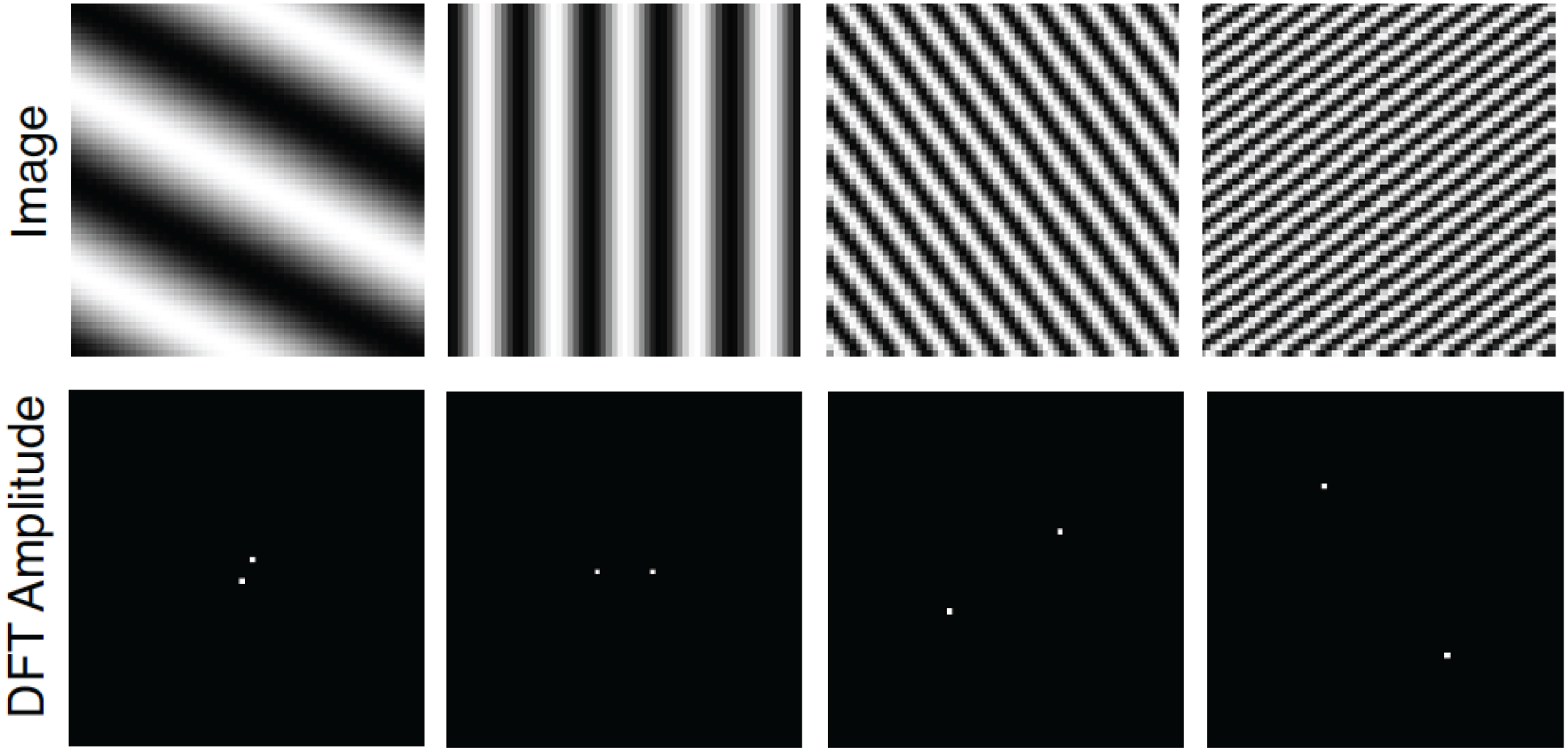


Visualizing the image Fourier transform



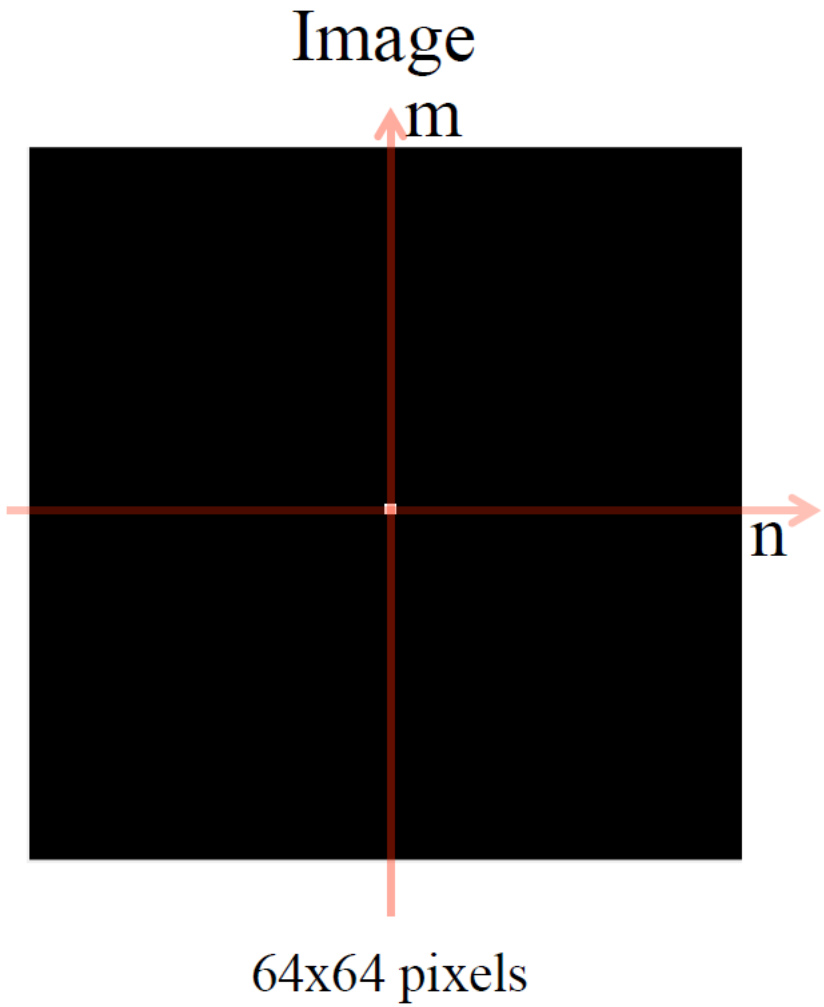
$$\cos\left(2\pi\left(\frac{u_0\,n}{N}+\frac{v_0\,m}{M}\right)\right) \quad \longleftrightarrow \quad \frac{1}{2}\left(\delta\left[u-u_0, v-v_0\right]+\delta\left[u+u_0, v+v_0\right]\right)$$

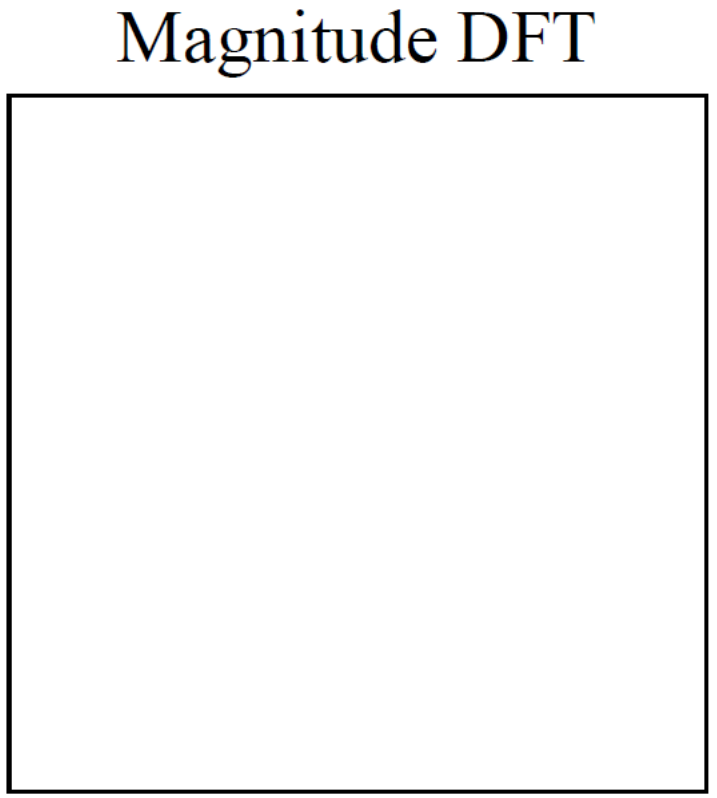
Simple Fourier transforms

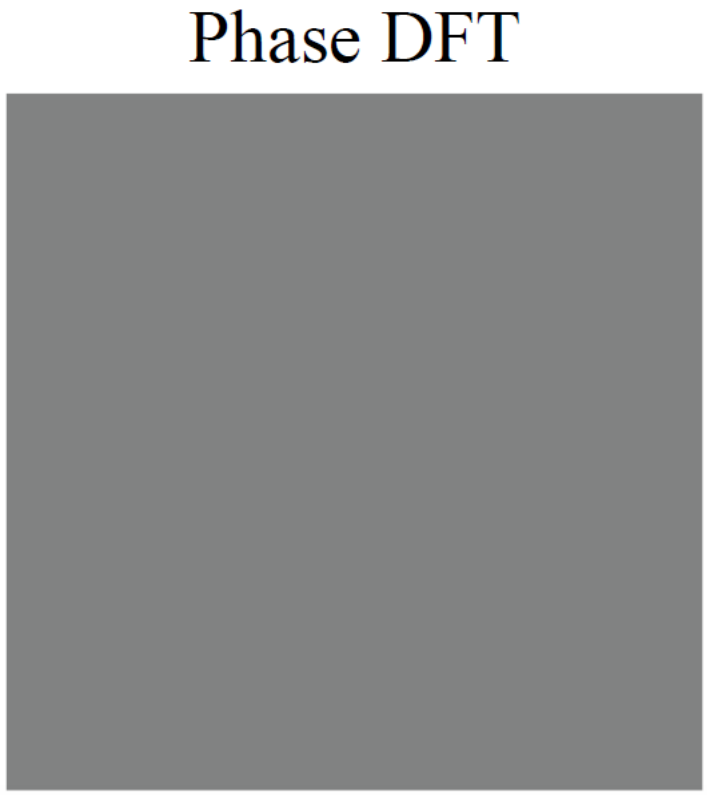


Images are 64x64 pixels. The wave is a cosine, therefore DFT phase is zero.

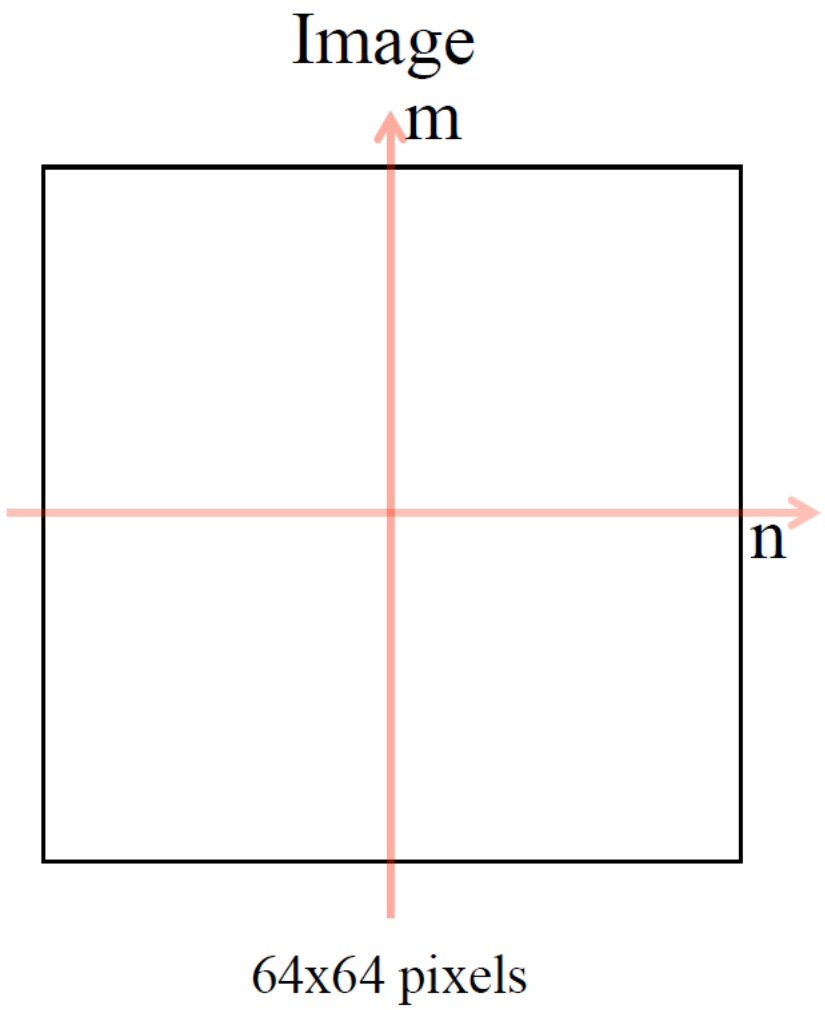
Some important Fourier transforms

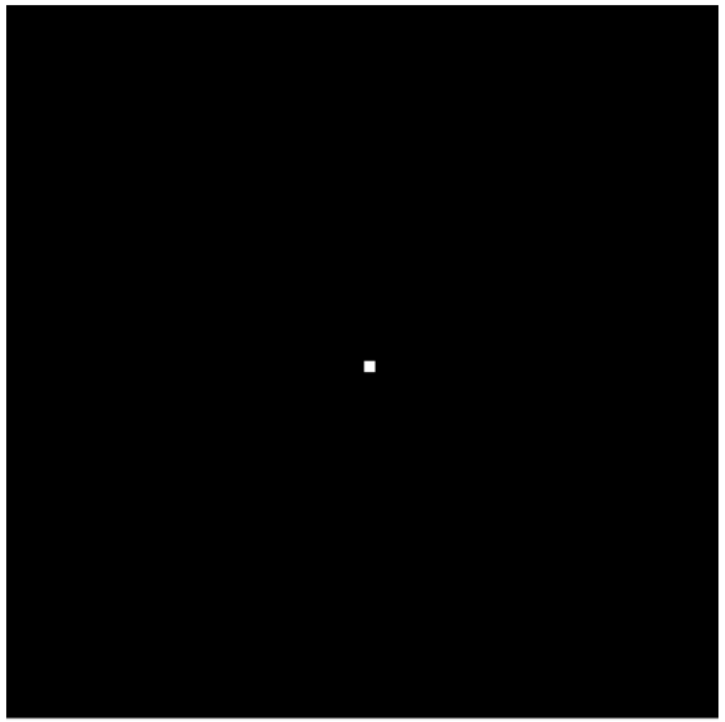




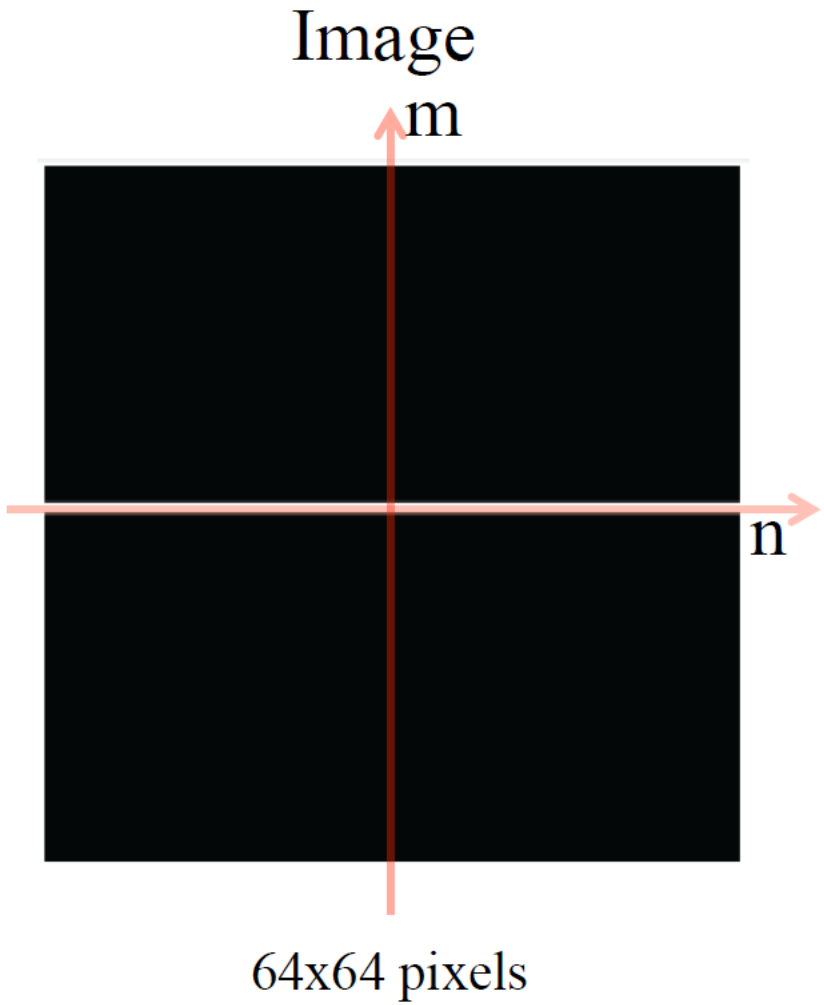


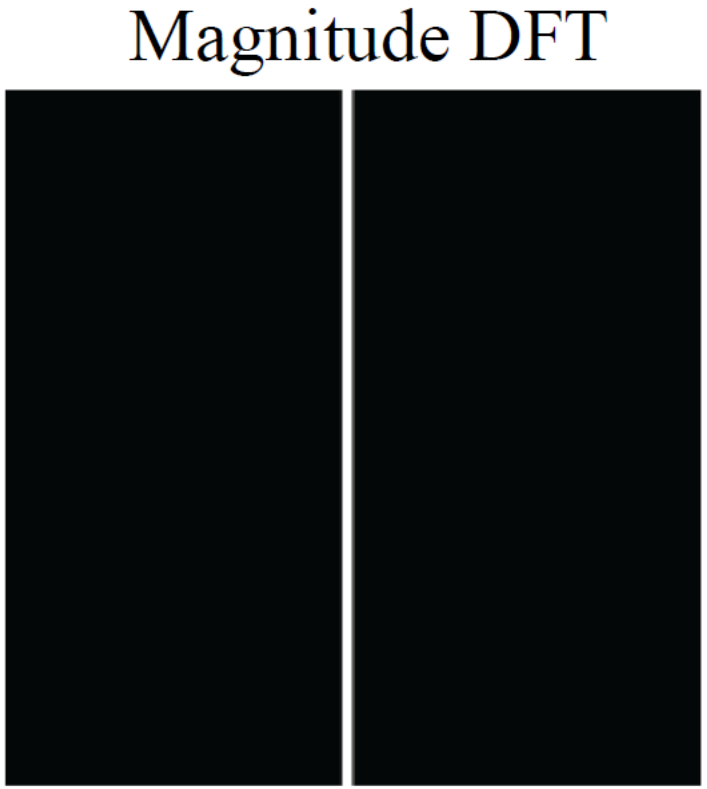
Some important Fourier transforms

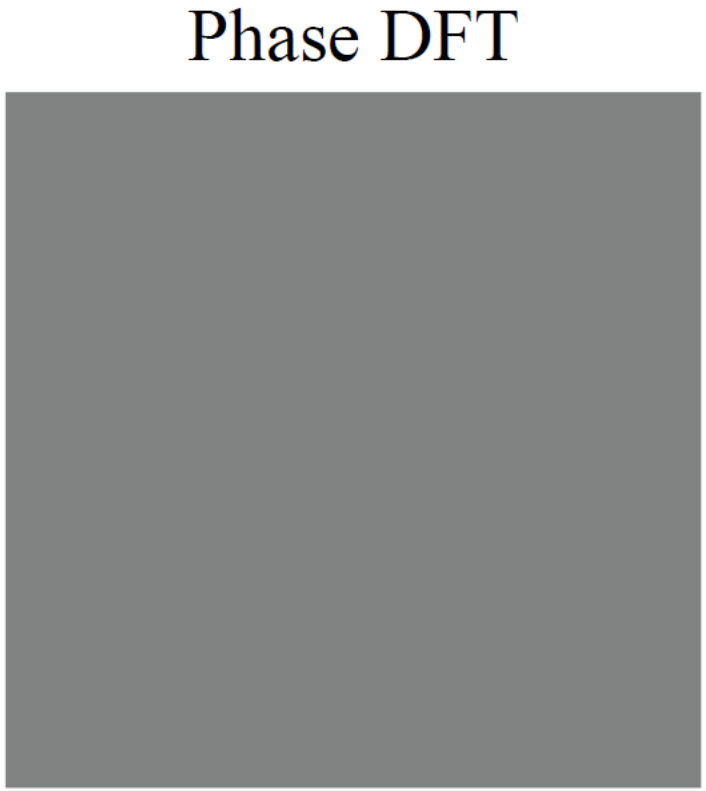




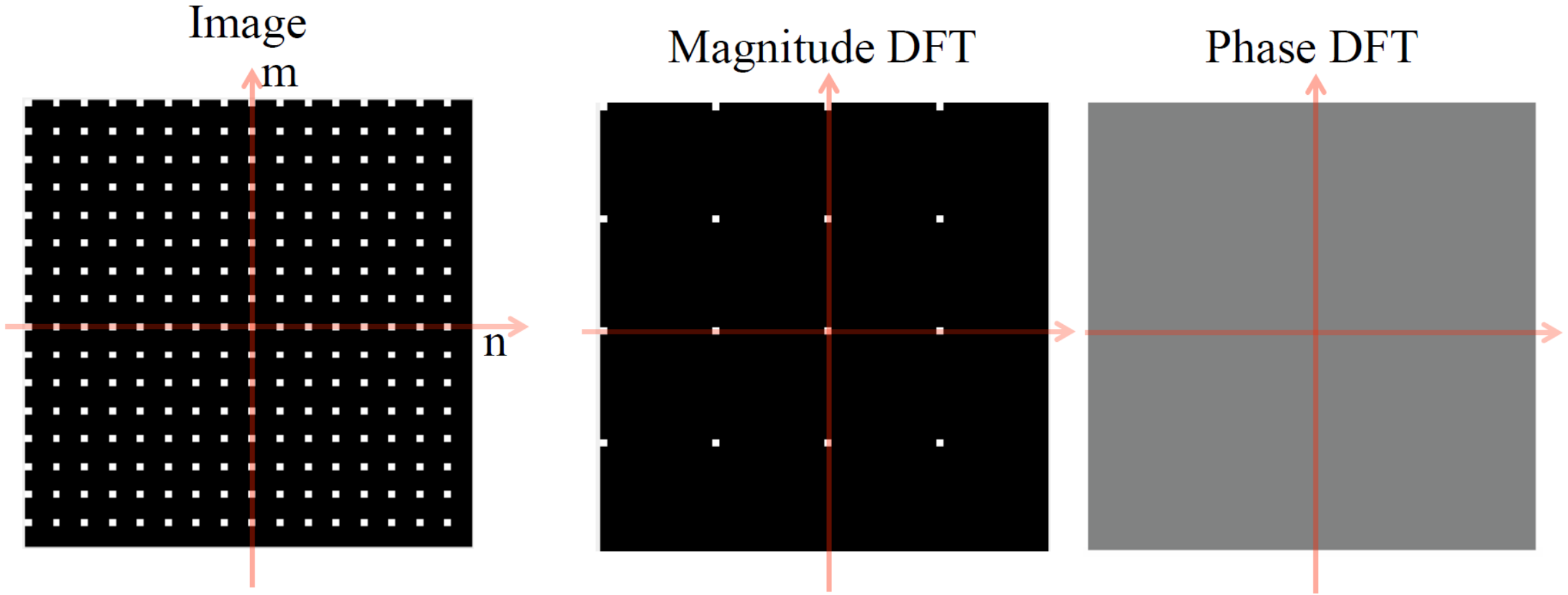
Some important Fourier transforms







Some important Fourier transforms



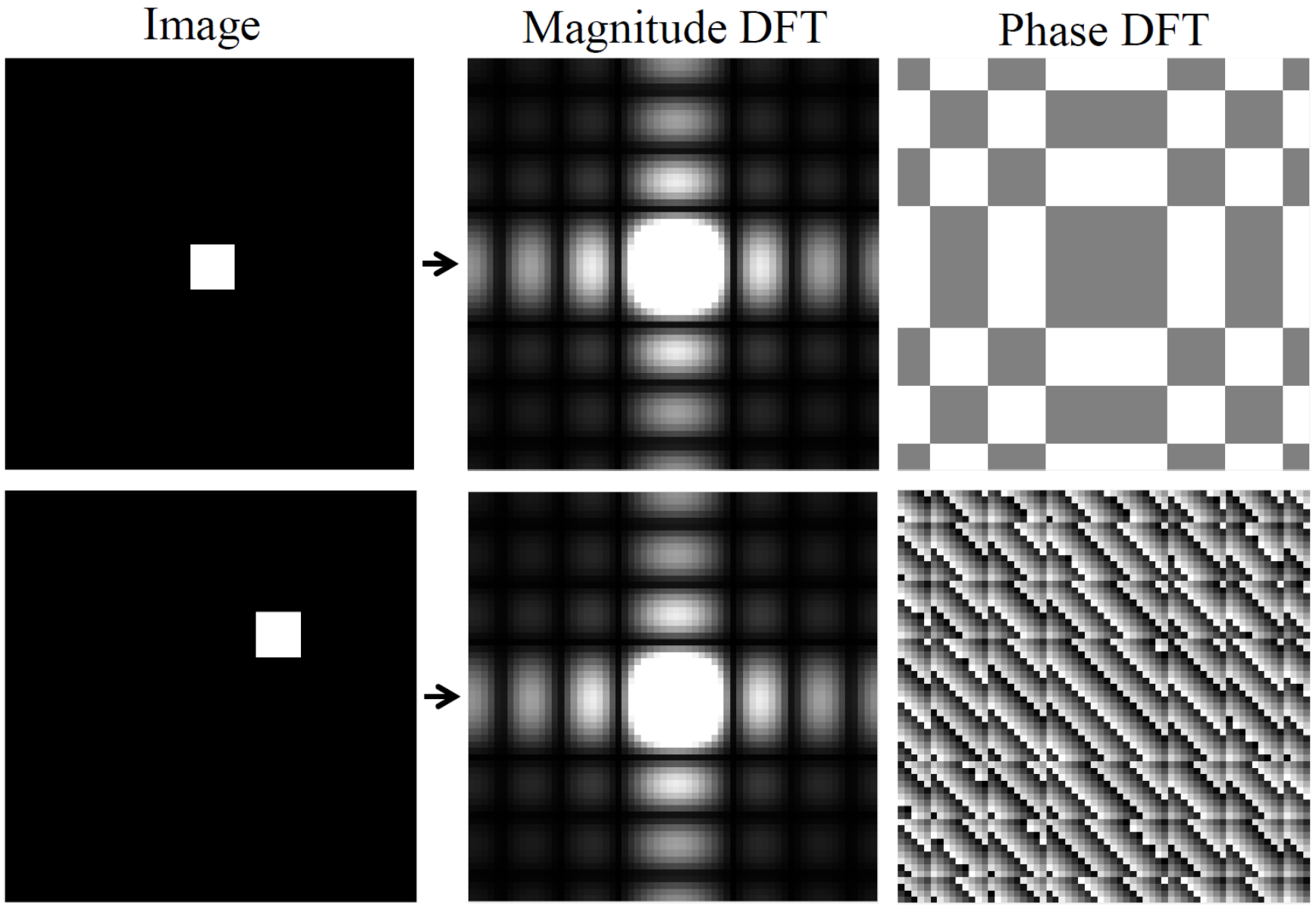
64x64 pixels

when M and N are divisible by k

$$\delta_k[n,m] = \sum_{s=0}^{N/k-1} \sum_{r=0}^{M/k-1} \delta[n - sk, m - rk] \quad \longleftrightarrow \quad \Delta_k[u,v] = \frac{NM}{k^2} \sum_{s=0}^{k-1} \sum_{r=0}^{k-1} \delta\left[u - s\frac{N}{k}, v - r\frac{M}{k}\right]$$

delta train, delta comb, impulse train

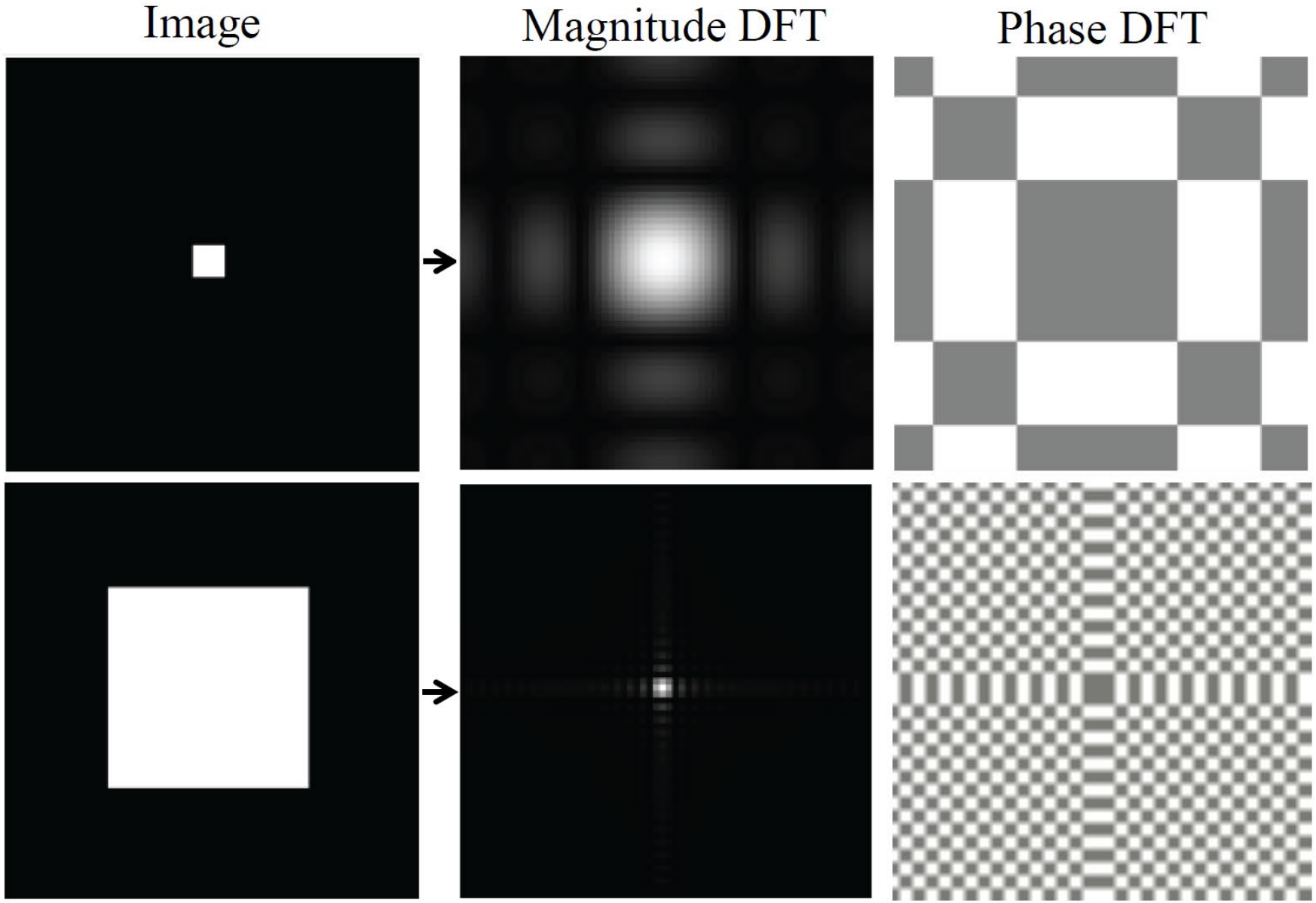
Some important Fourier transforms



Translation

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

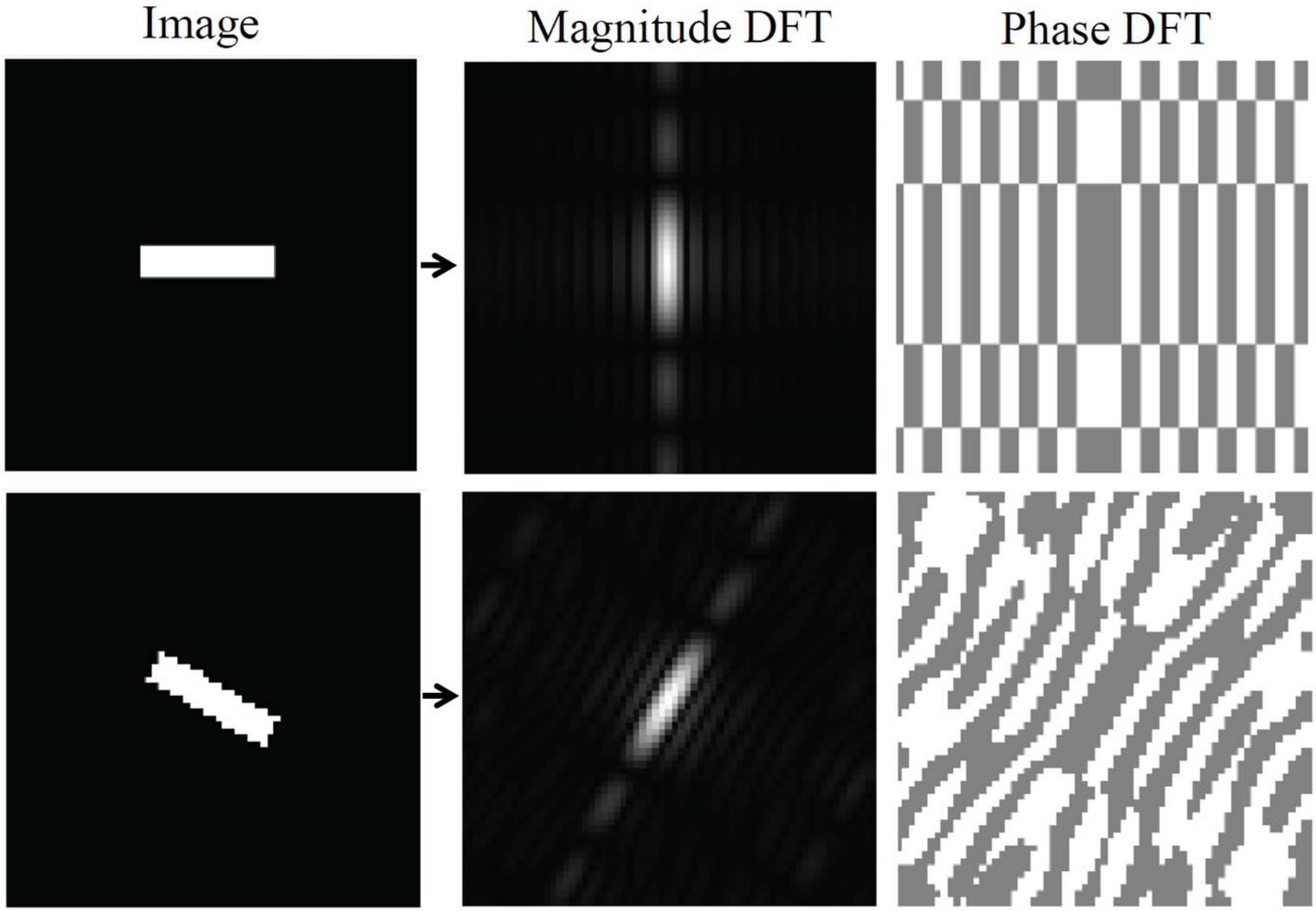
Some important Fourier transforms



Scale

Small image details produce content in high spatial frequencies

Some important Fourier transforms



Orientation

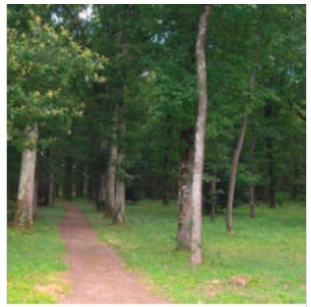
A line transforms to a line oriented perpendicularly to the first.

The DFT Game: find the right pairs

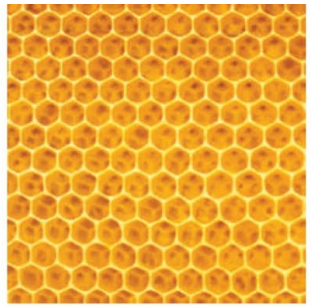

a)


b)


c)

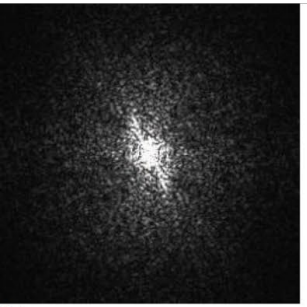

d)

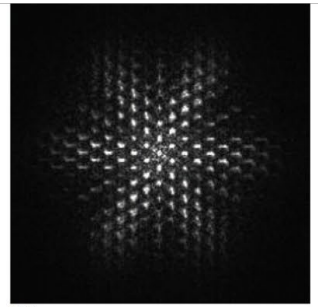

e)

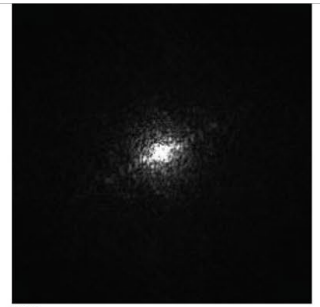

f)

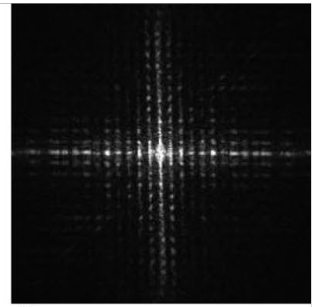

g)

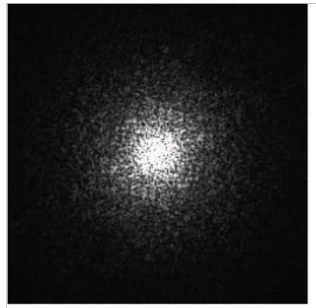

h)

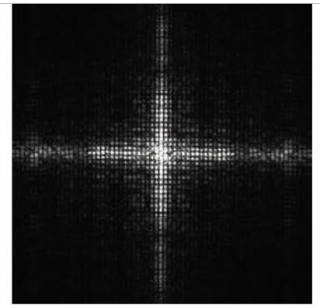

1)


2)

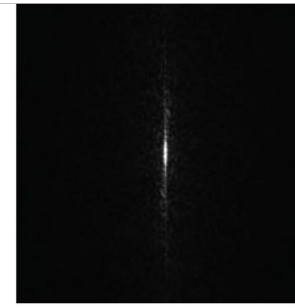

3)


4)


5)


6)


7)


8)

The inverse Discrete Fourier transform

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

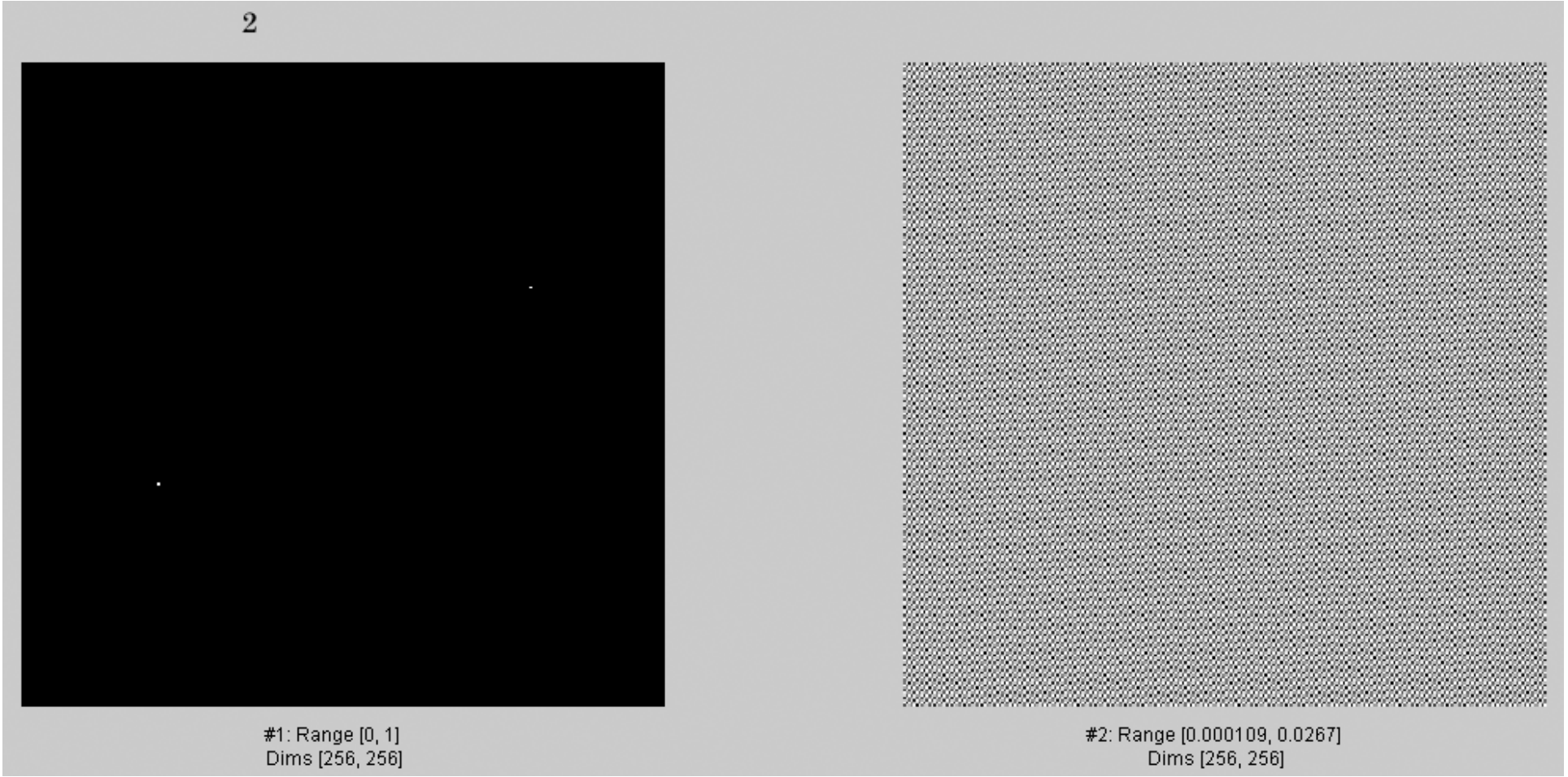
$$F[u,v] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j\left(\frac{un}{N}+\frac{vm}{M}\right)\right)$$

The inverse of the 2D DFT is:

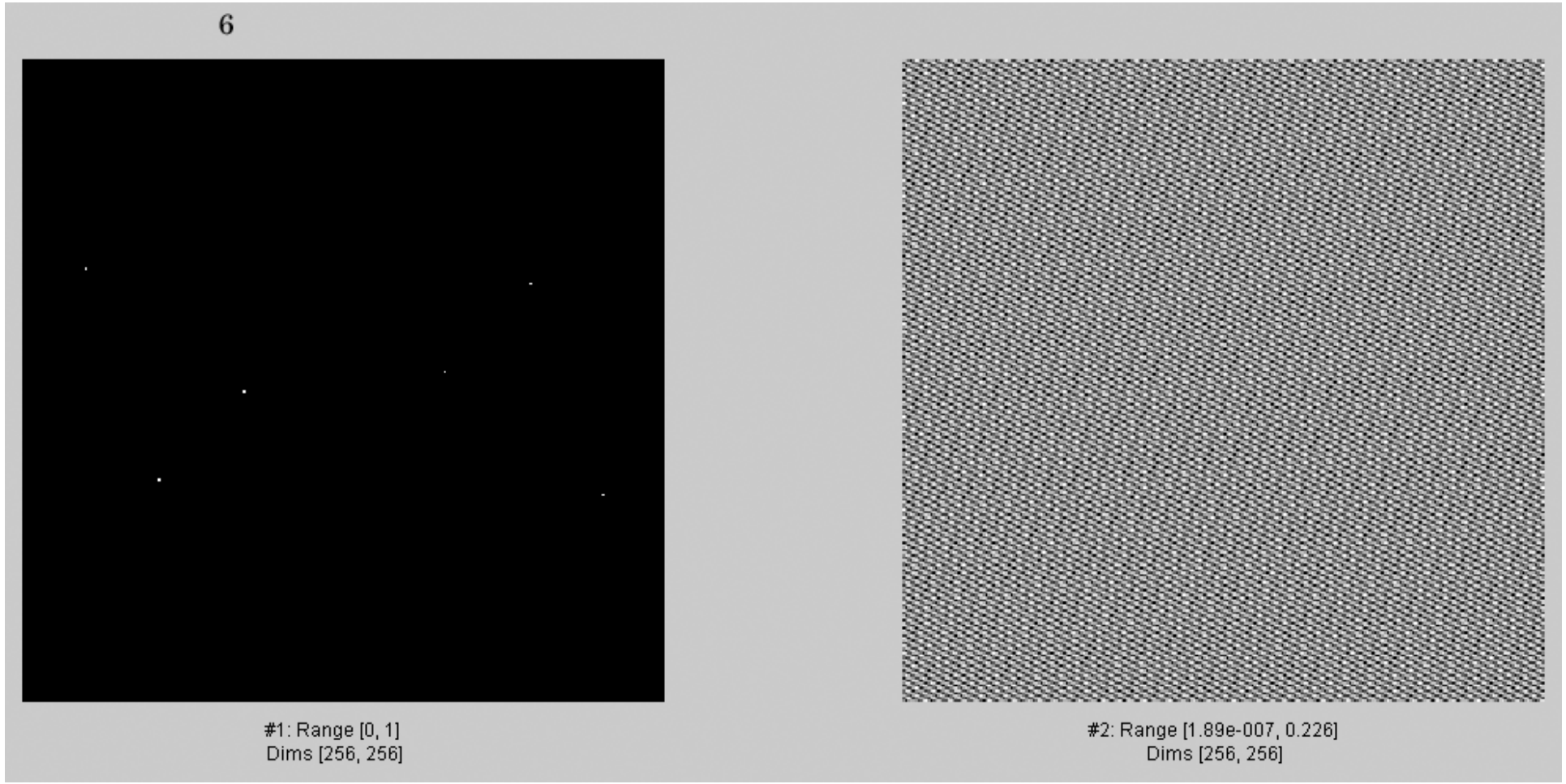
$$f[n,m] = \frac{1}{NM}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} F[u,v] \exp\left(+2\pi j\left(\frac{un}{N}+\frac{vm}{M}\right)\right)$$

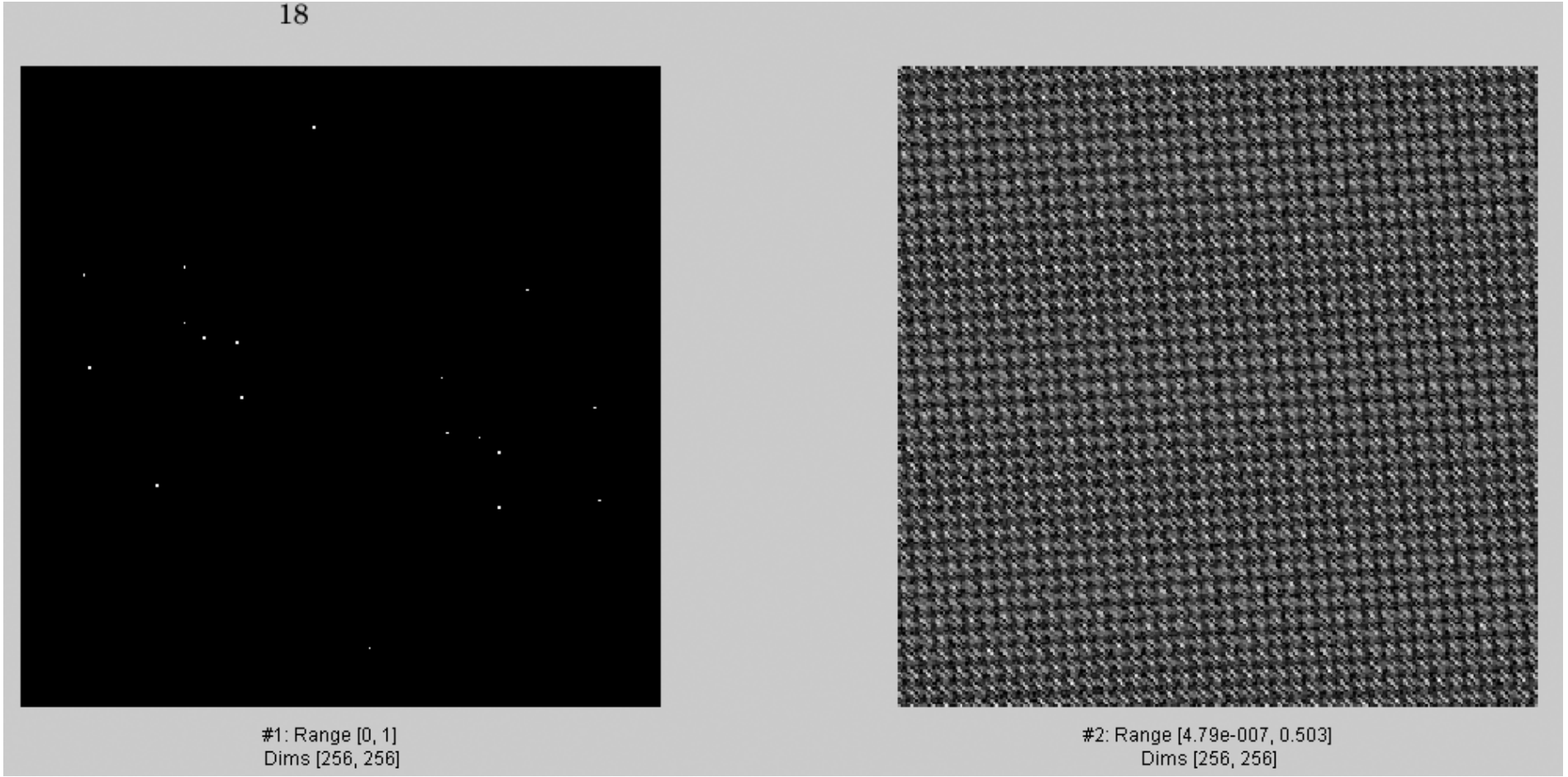
How does summing waves ends up giving back a picture?

2

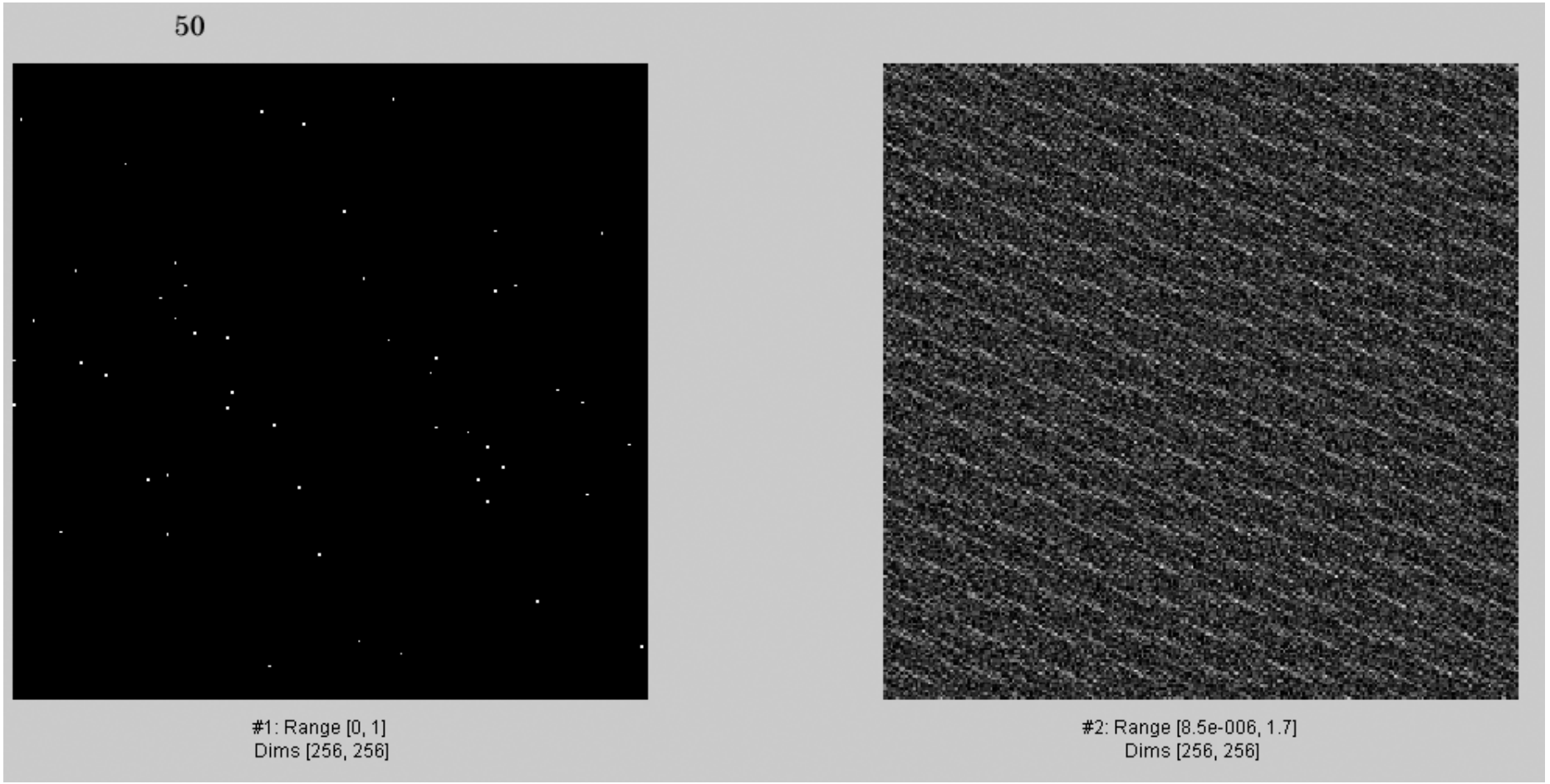


6

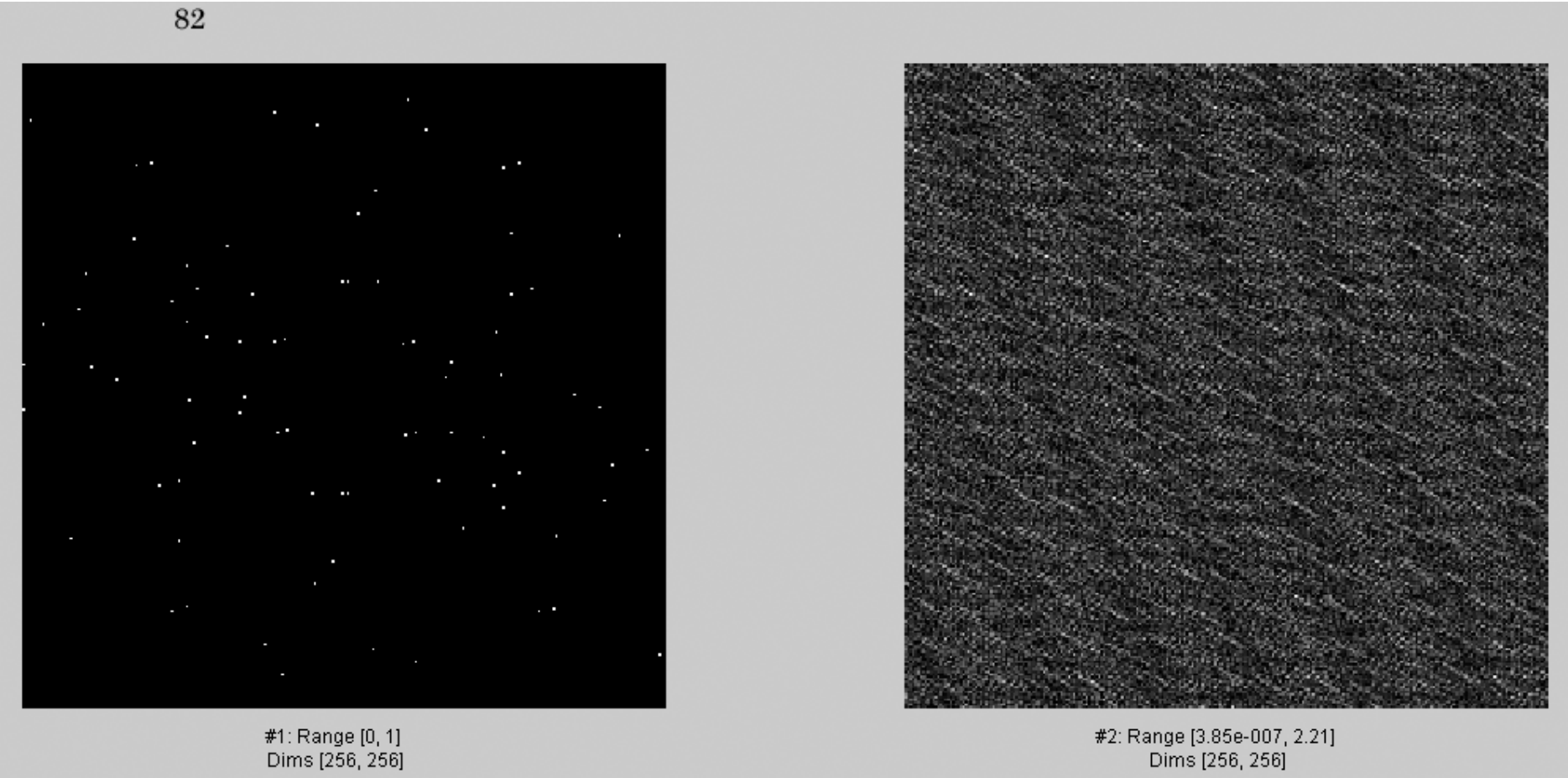



18
#1: Range [0, 1]
Dims [256, 256]
#2: Range [4.79e-007, 0.503]
Dims [256, 256]

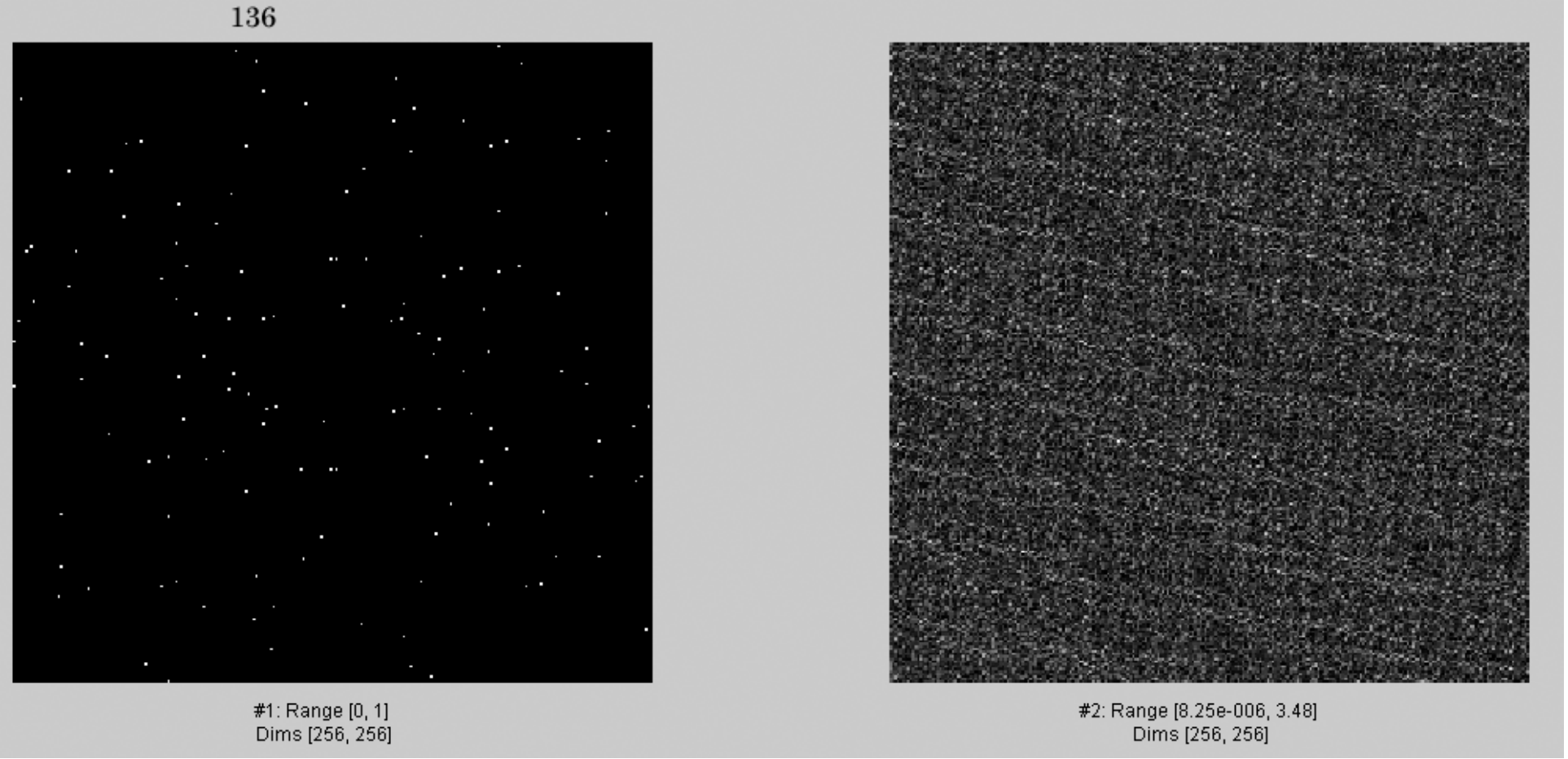
50



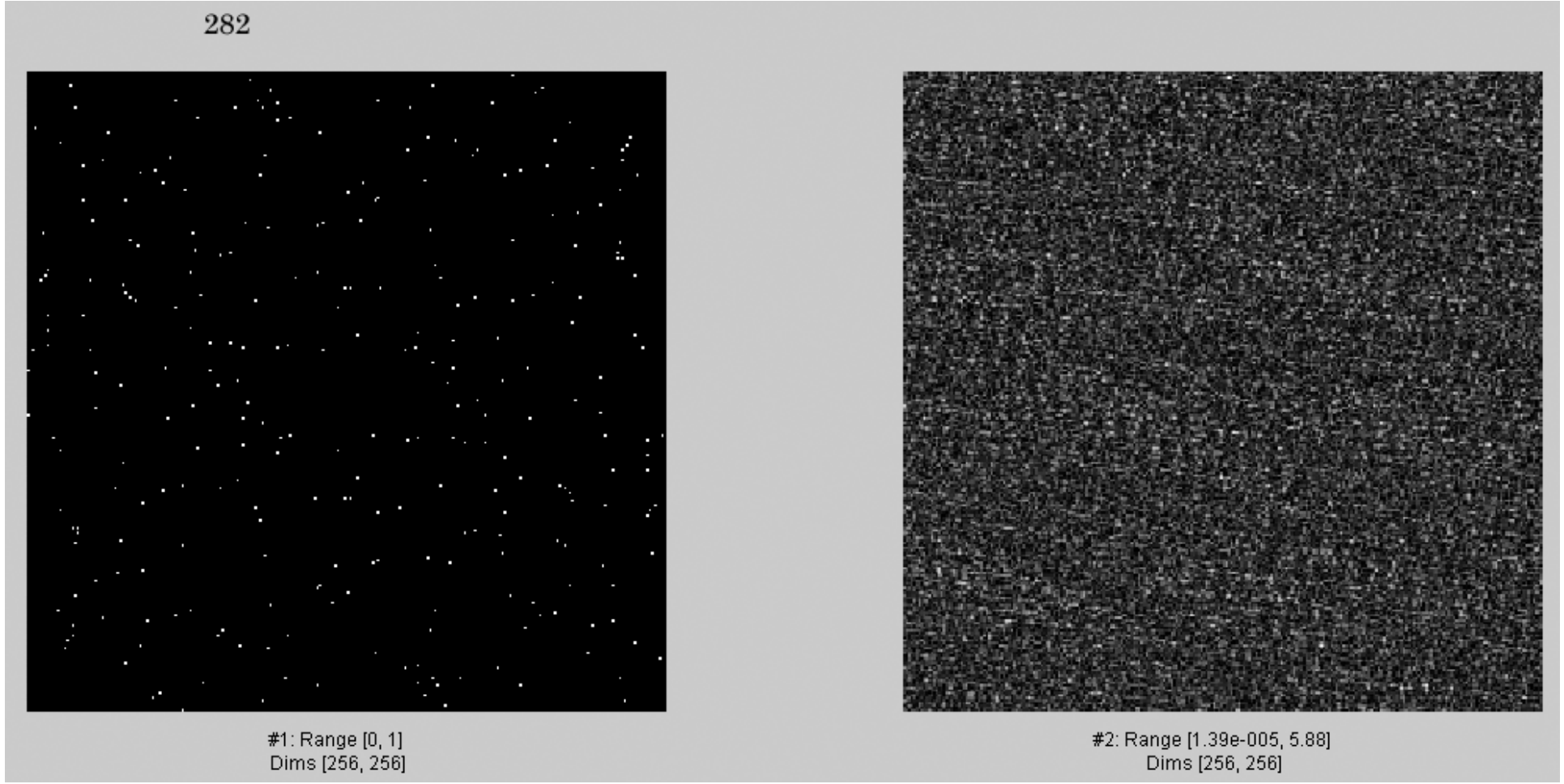
82



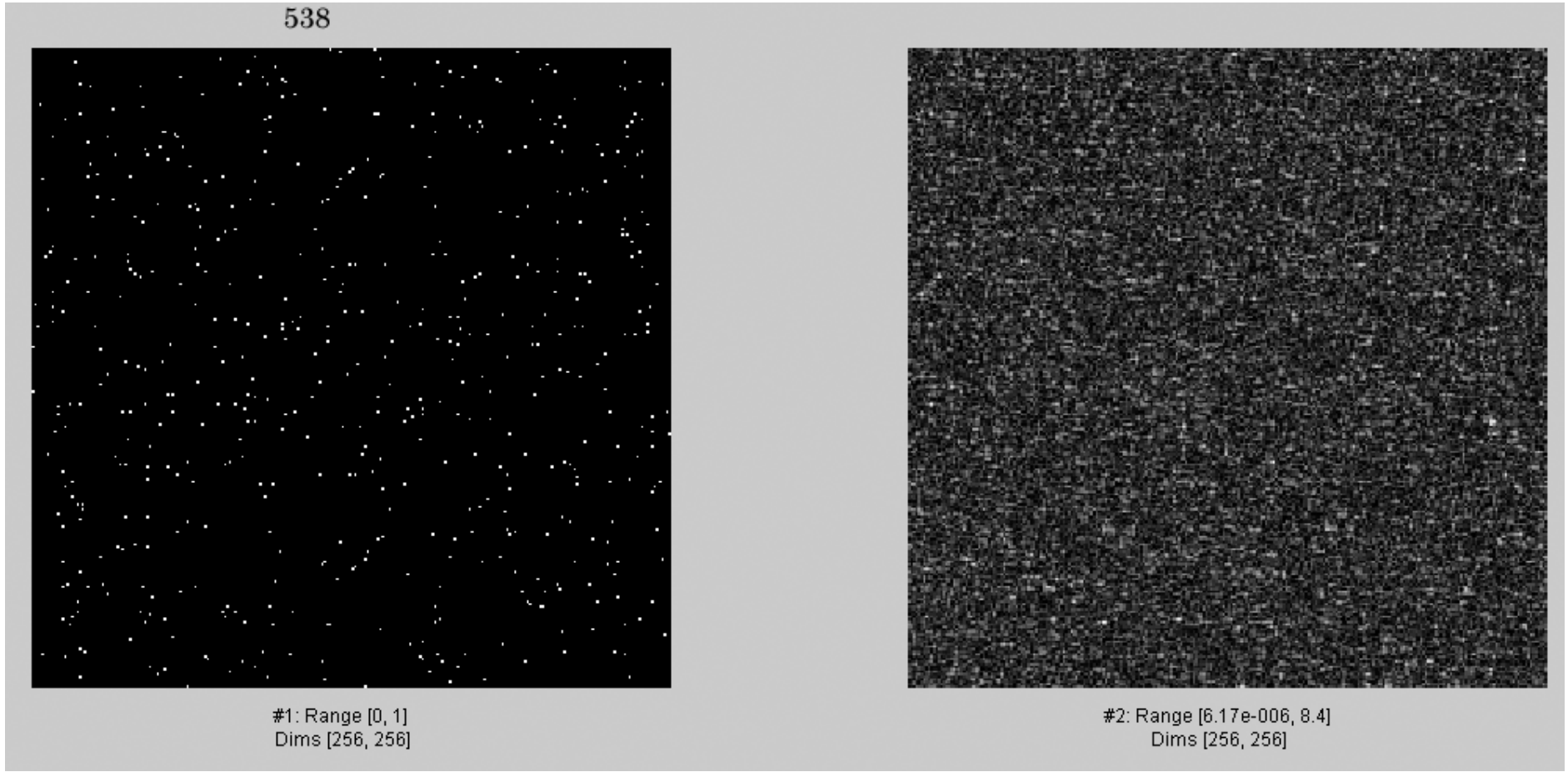
136



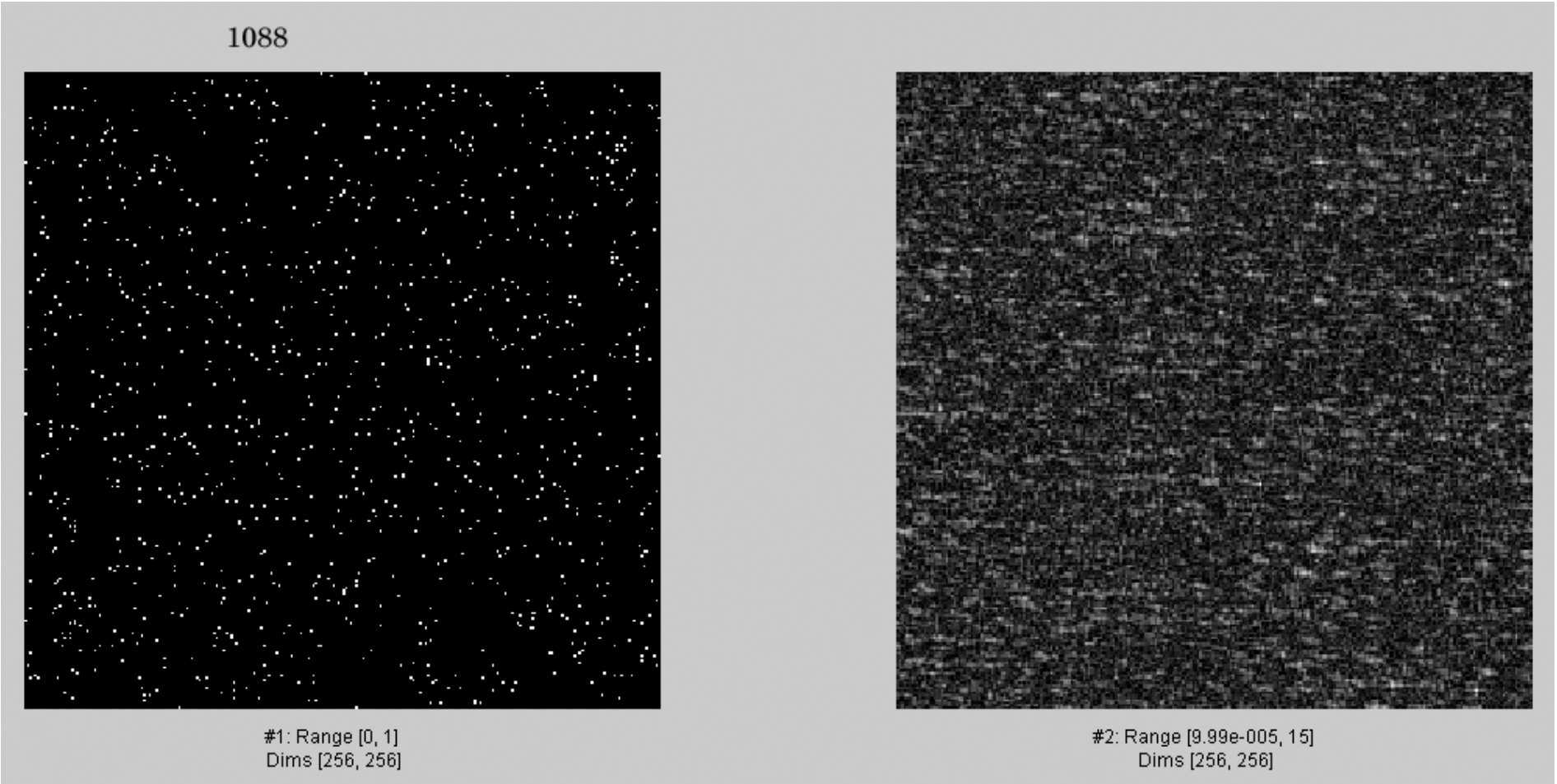
282



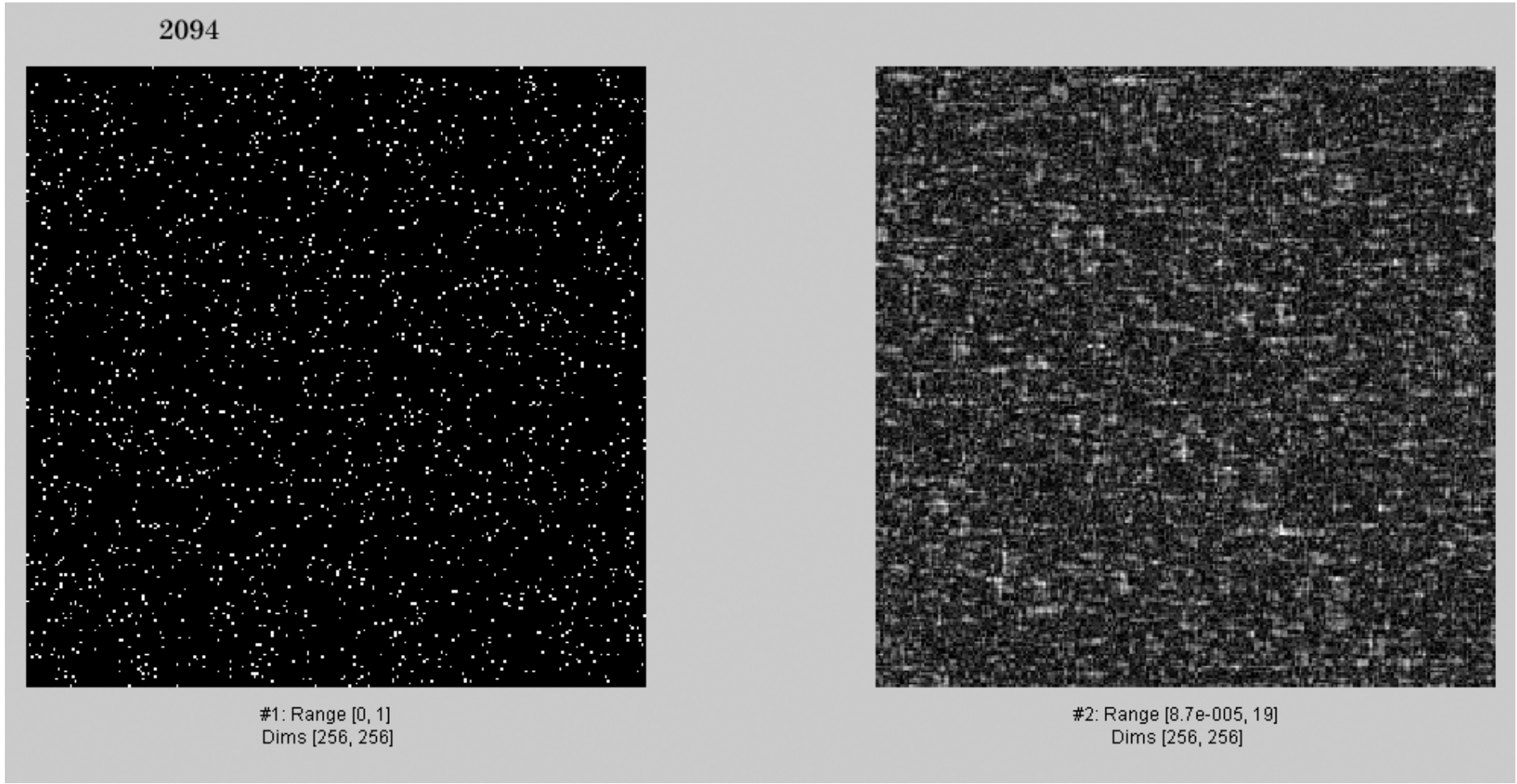
538



1088

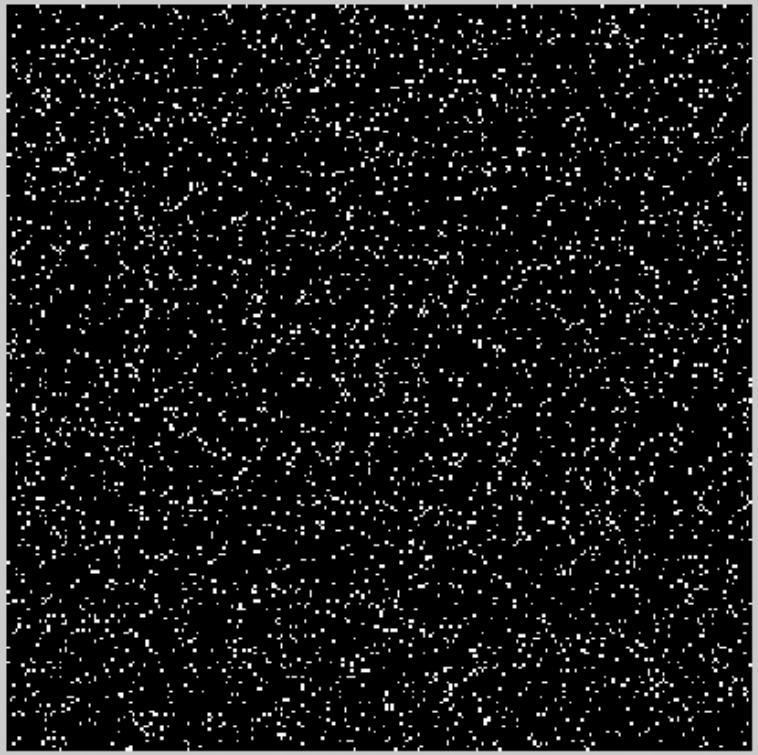


2094

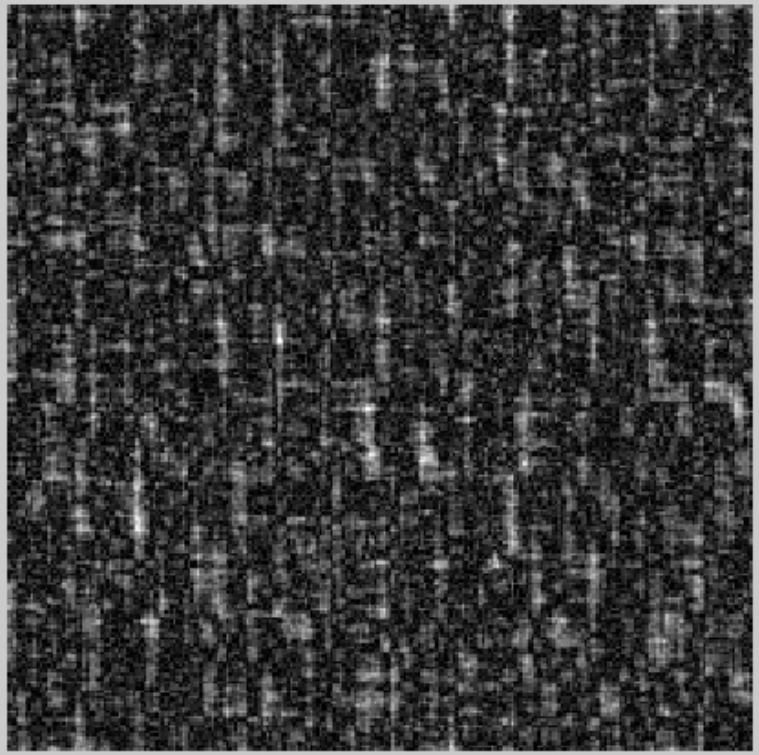


4052

4052

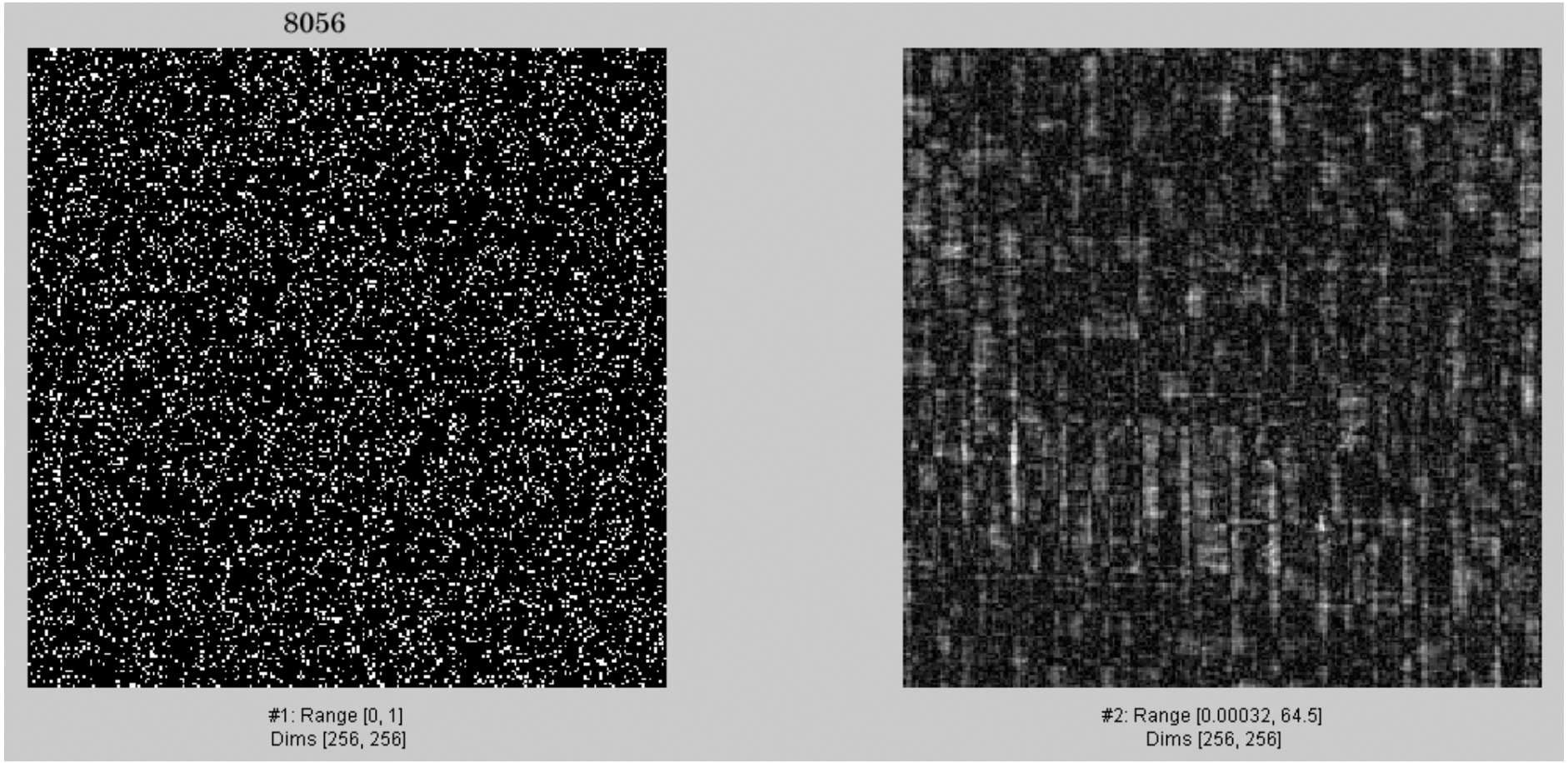


#1: Range [0, 1]
Dims [256, 256]

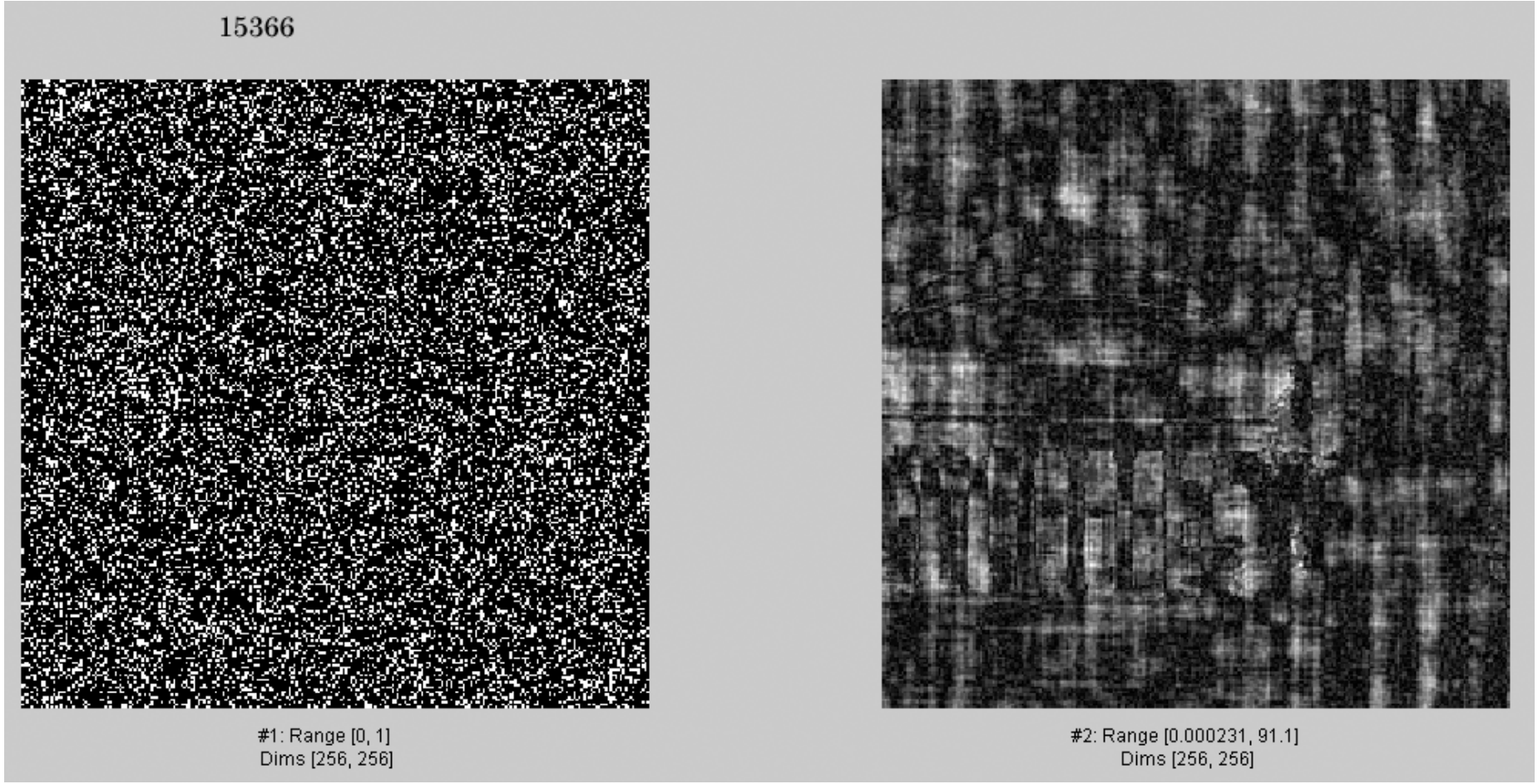


#2: Range [0.000556, 37.7]
Dims [256, 256]

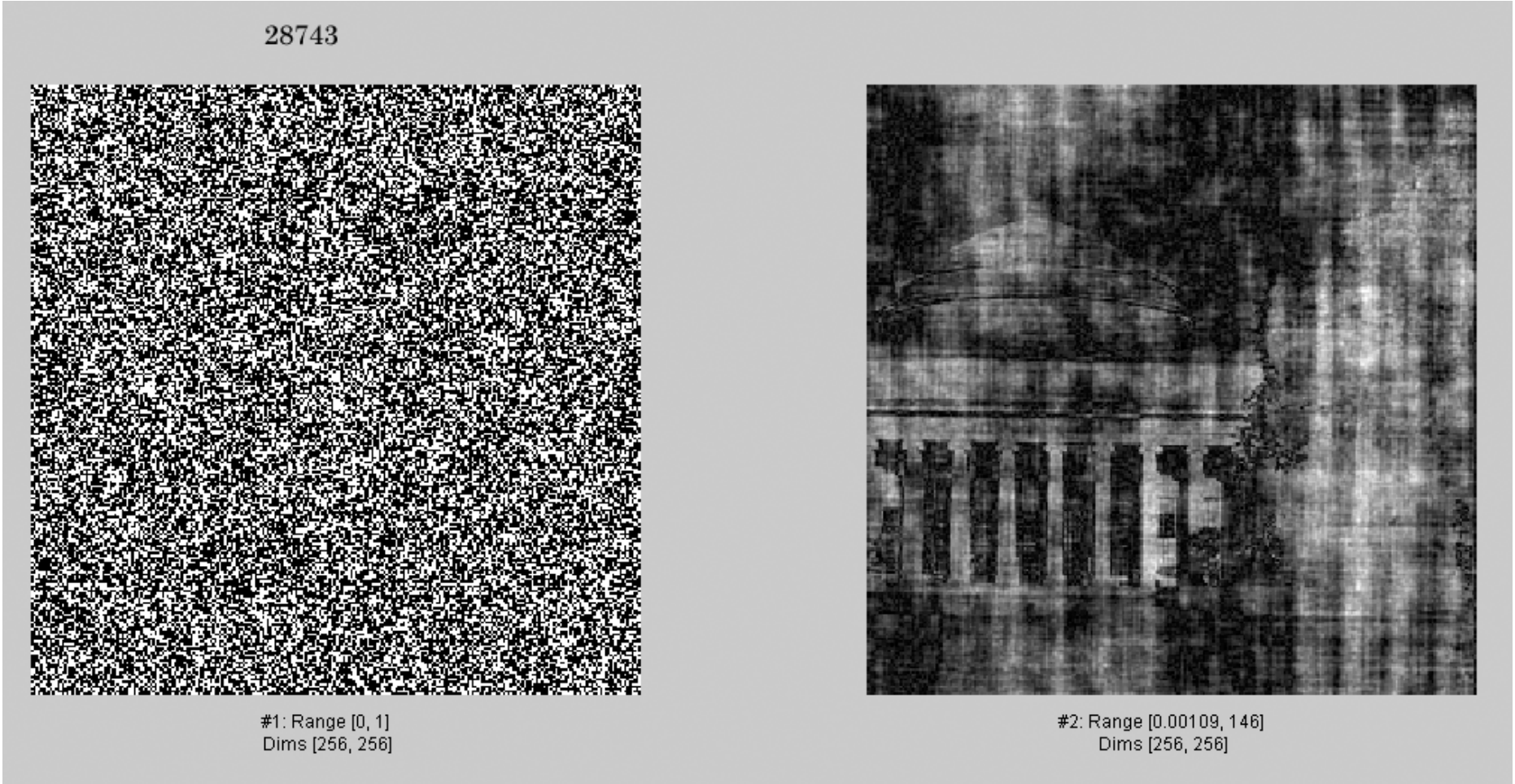
8056.



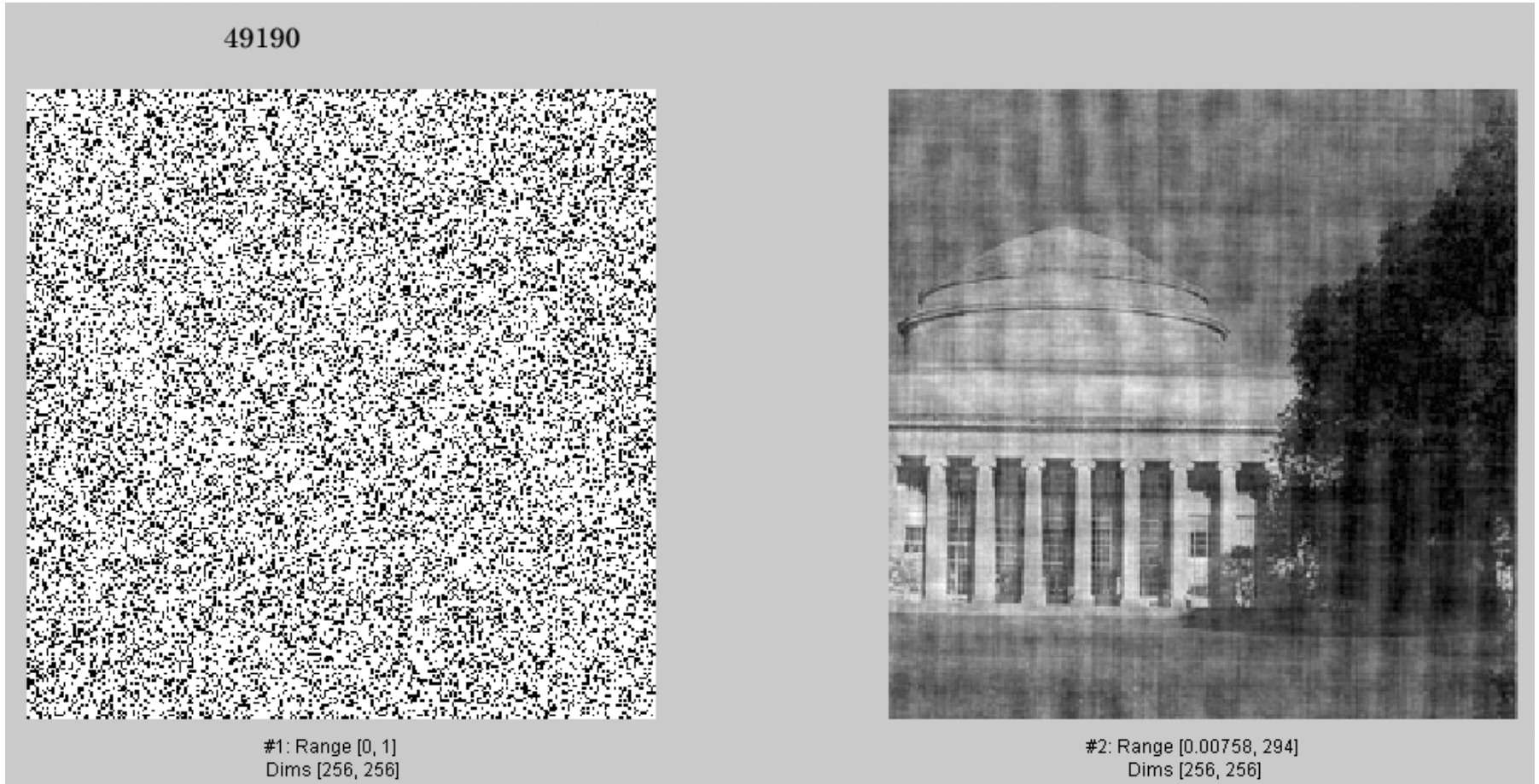
15366



28743



49190.



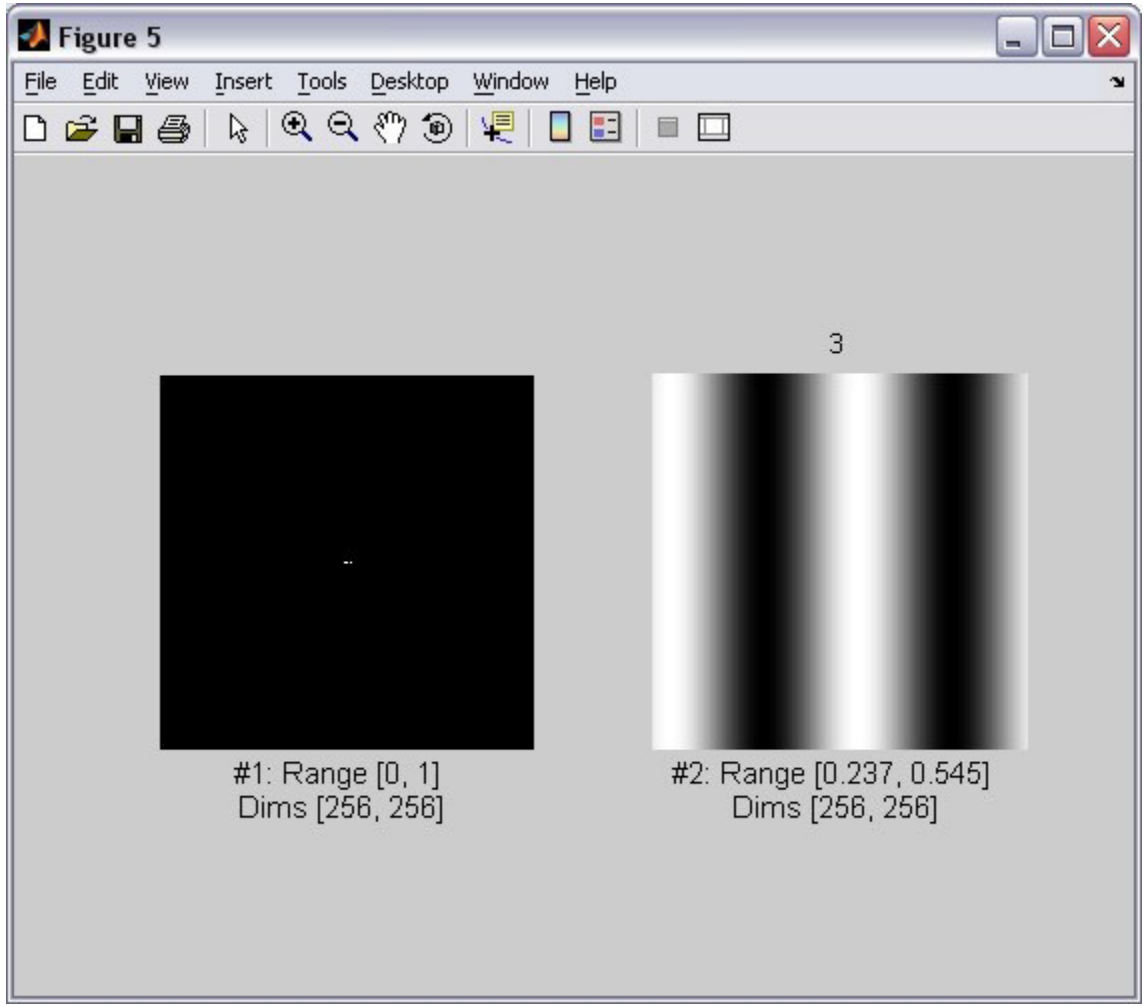
65536.

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

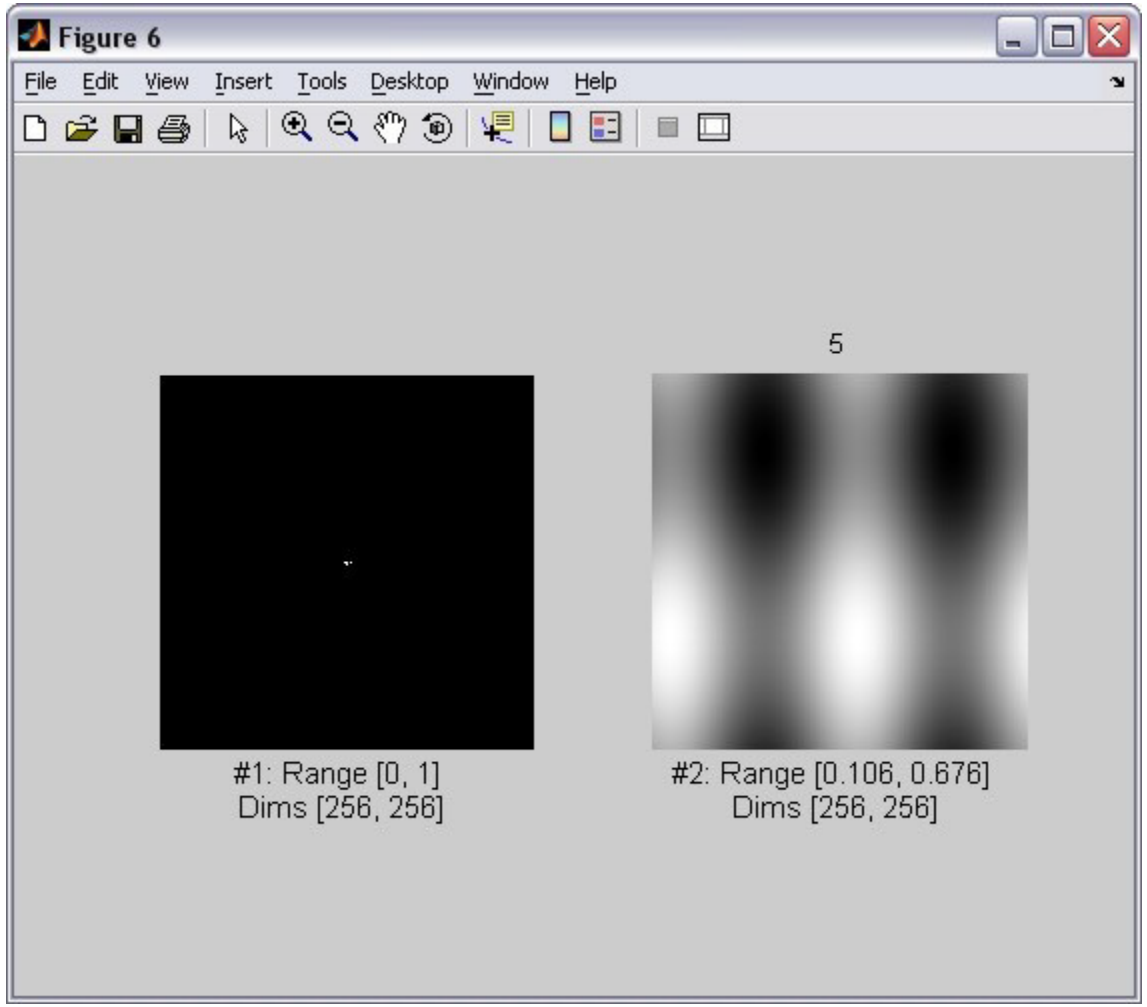


Figure 6
File Edit View Insert Tools Desktop Window Help
5
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.106, 0.676]
Dims [256, 256]

9

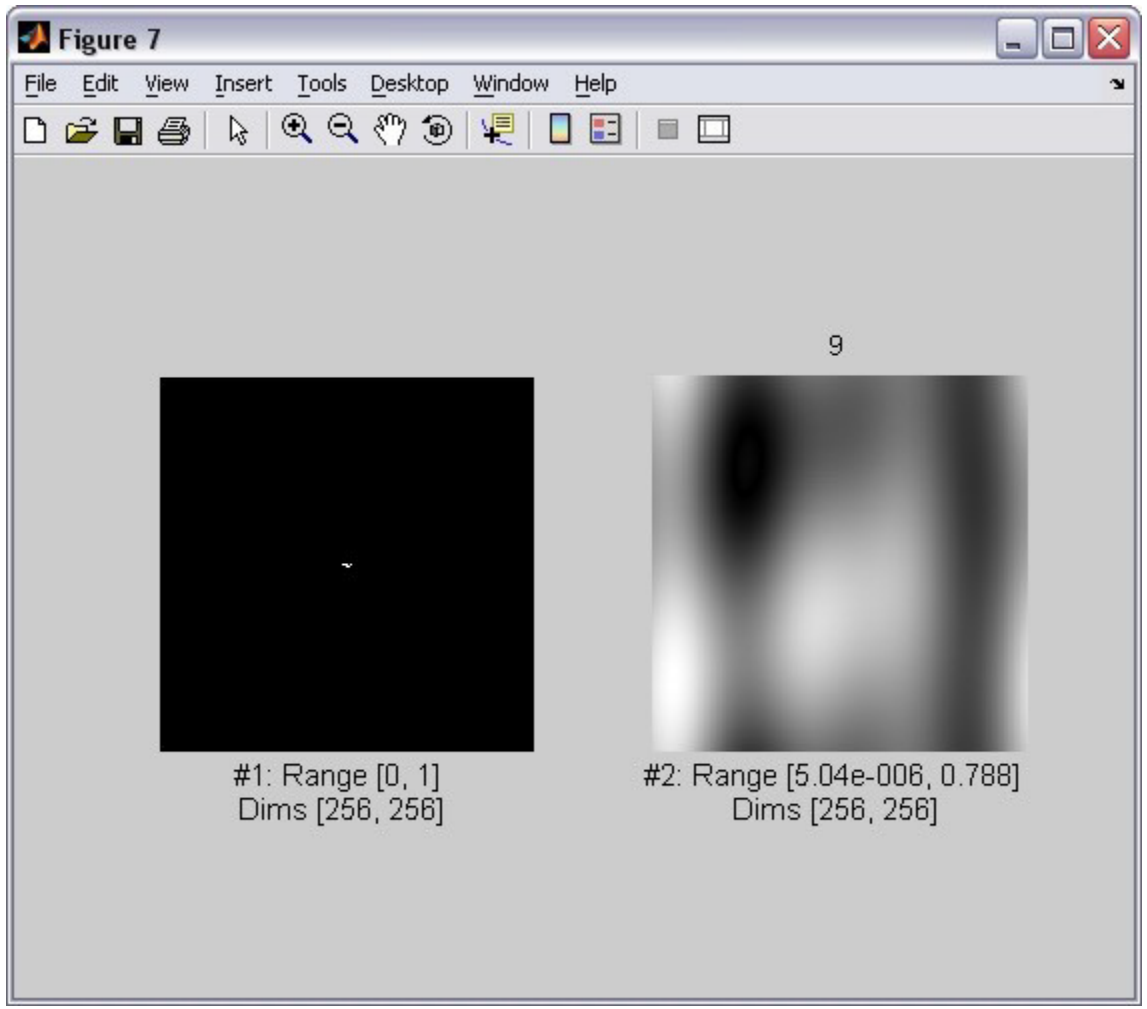


Figure 7
File Edit View Insert Tools Desktop Window Help
9
#1: Range [0, 1]
Dims [256, 256]
#2: Range [5.04e-006, 0.788]
Dims [256, 256]

17

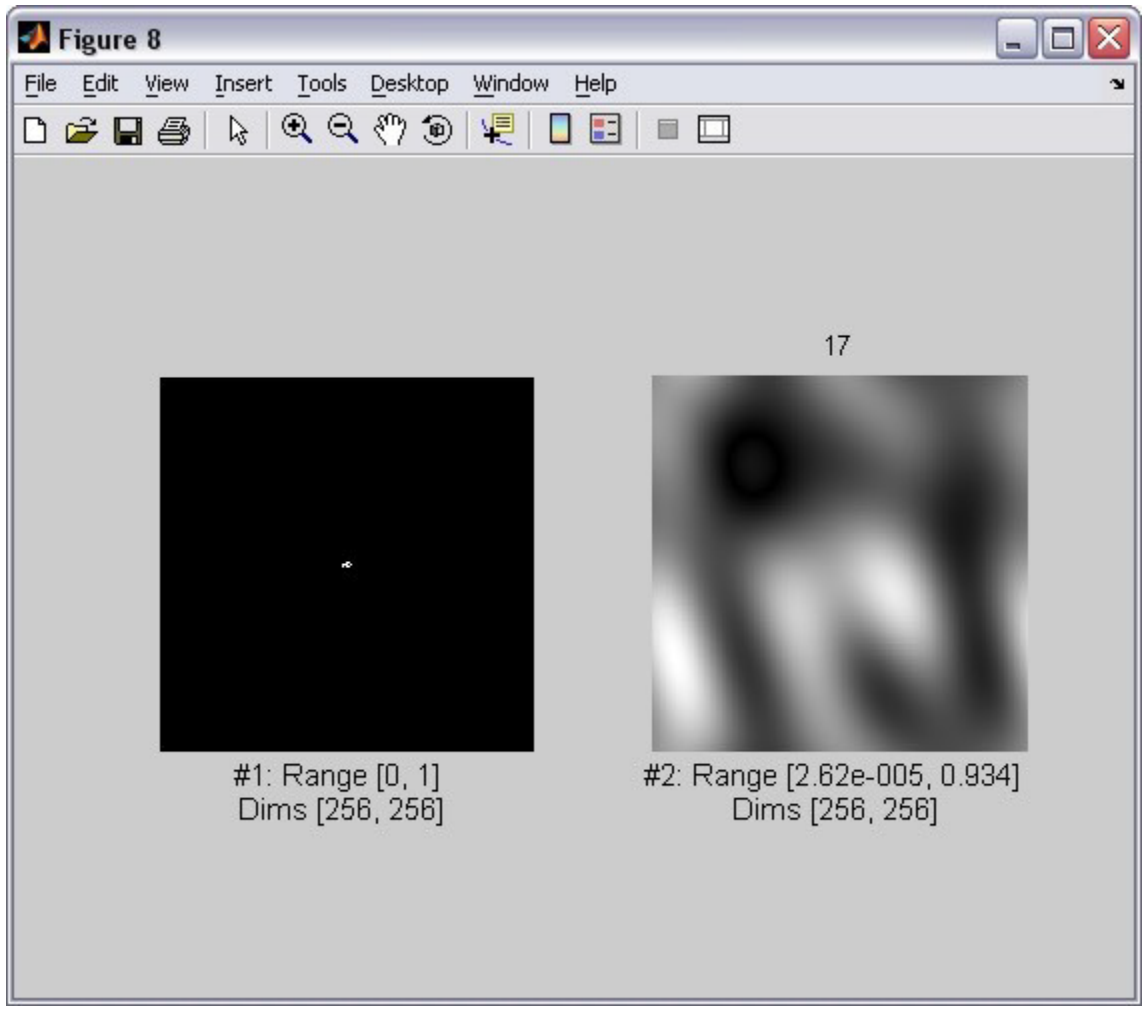


Figure 8
File Edit View Insert Tools Desktop Window Help
17
#1: Range [0, 1]
Dims [256, 256]
#2: Range [2.62e-005, 0.934]
Dims [256, 256]

33

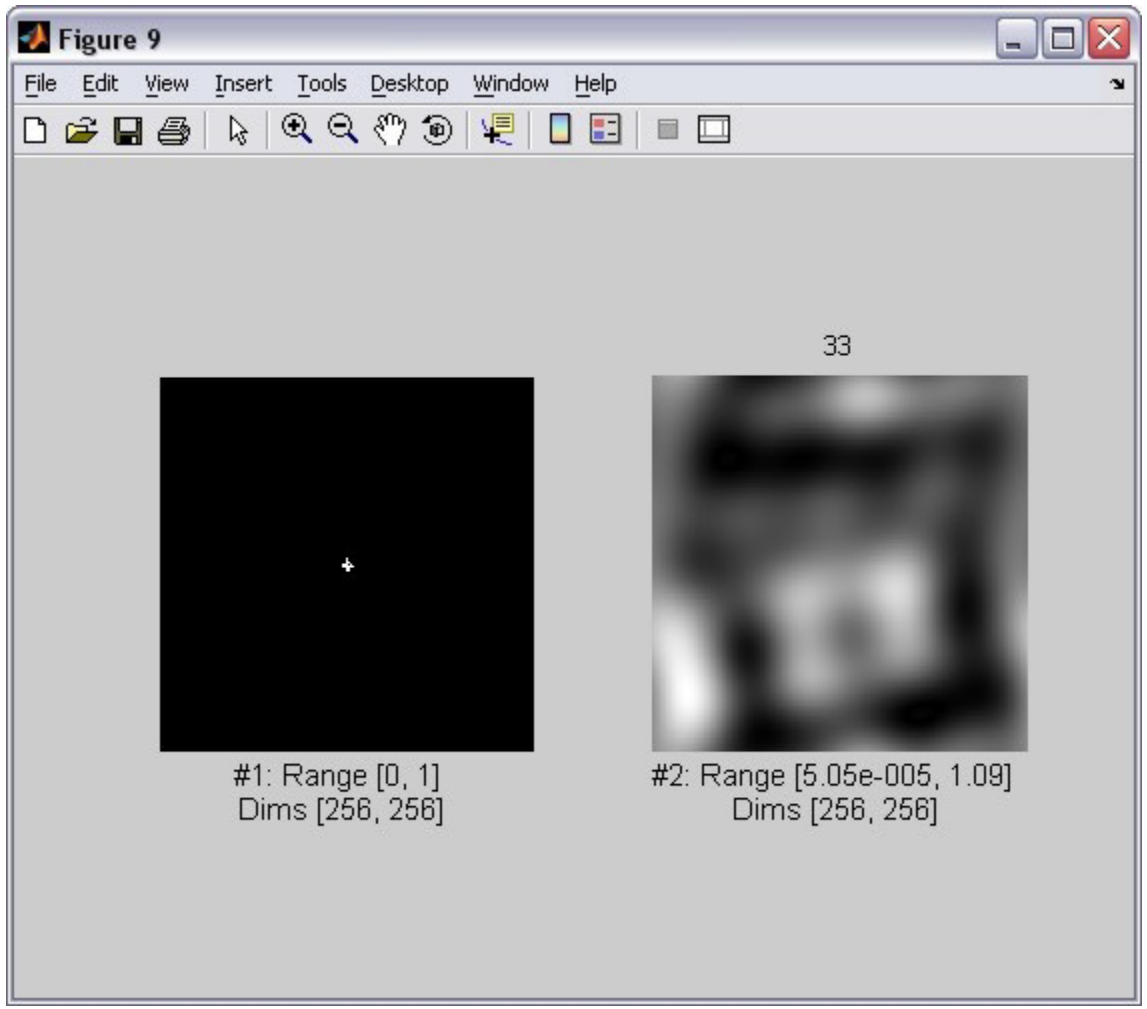


Figure 9
File Edit View Insert Tools Desktop Window Help
33
#1: Range [0, 1]
Dims [256, 256]
#2: Range [5.05e-005, 1.09]
Dims [256, 256]

65

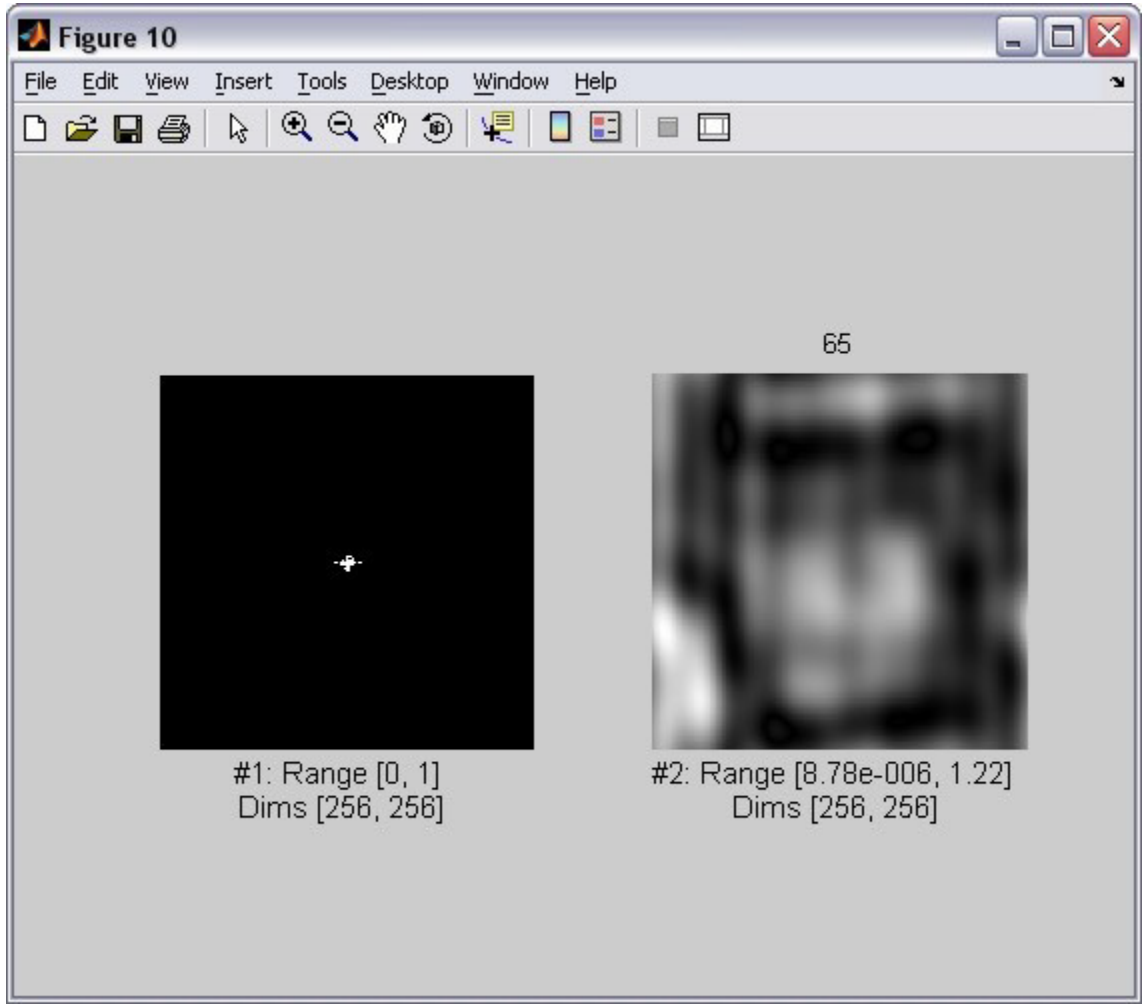


Figure 10
File Edit View Insert Tools Desktop Window Help
65
#1: Range [0, 1]
Dims [256, 256]
#2: Range [8.78e-006, 1.22]
Dims [256, 256]

129

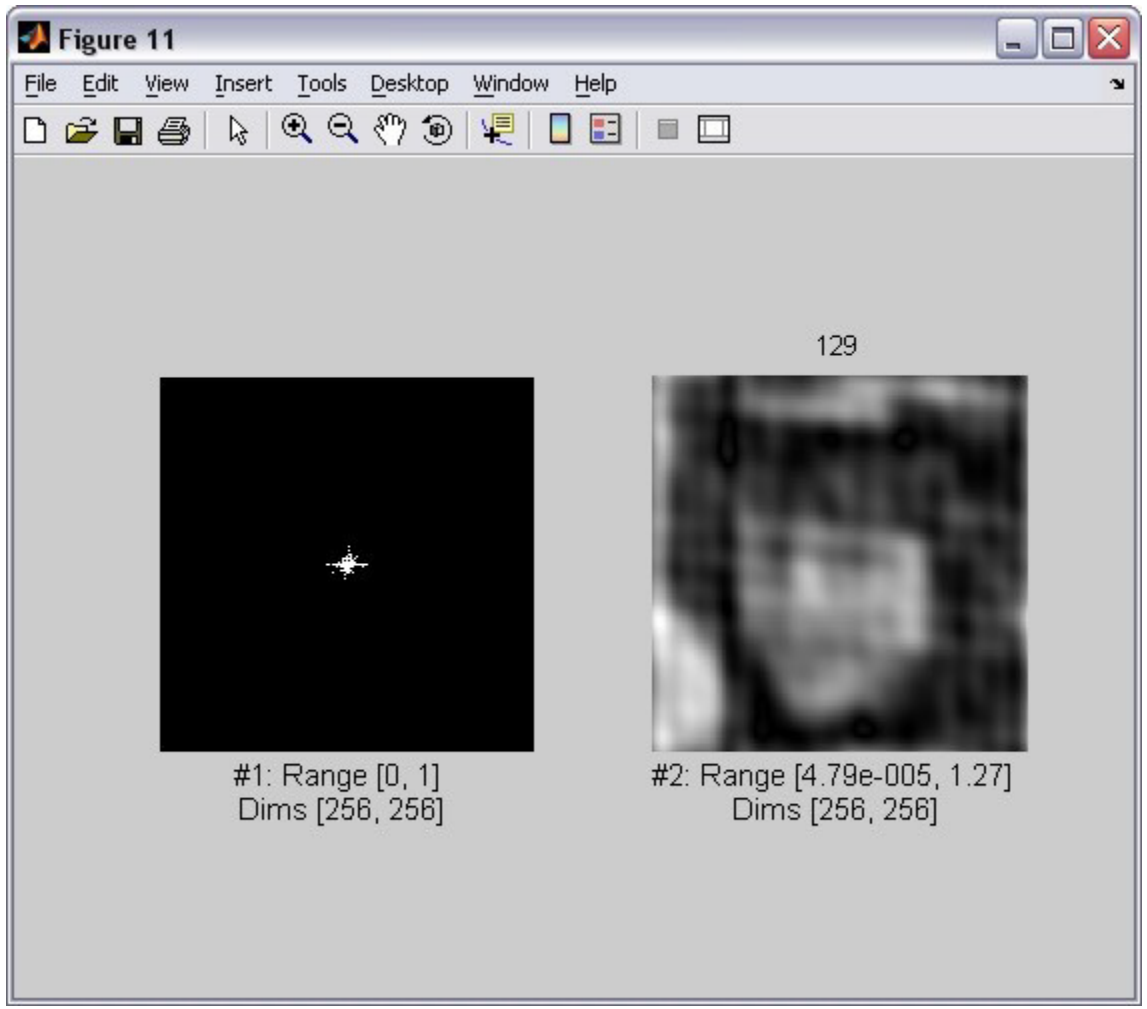


Figure 11
File Edit View Insert Tools Desktop Window Help
129
#1: Range [0, 1]
Dims [256, 256]
#2: Range [4.79e-005, 1.27]
Dims [256, 256]

257

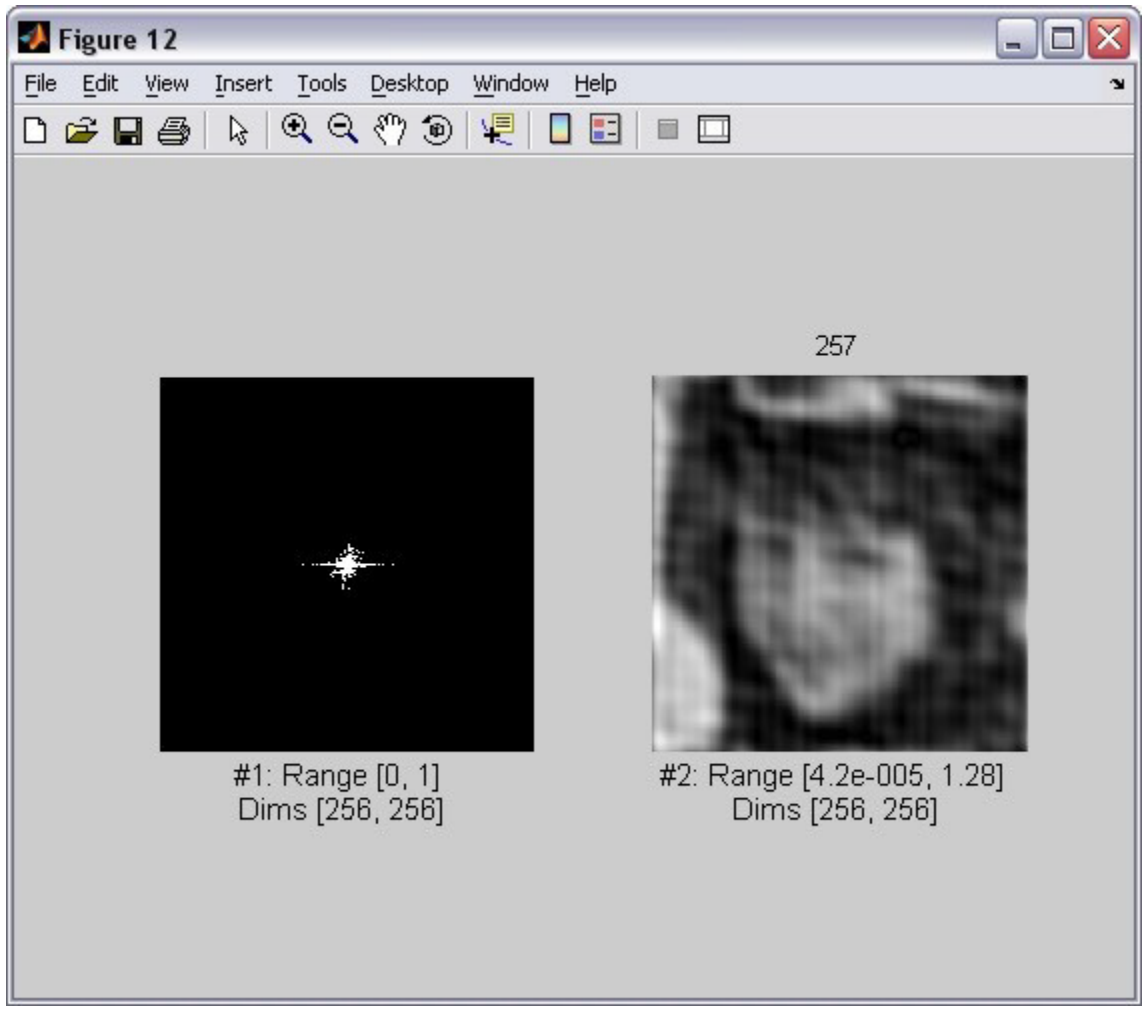


Figure 12
File Edit View Insert Tools Desktop Window Help
257
#1: Range [0, 1]
Dims [256, 256]
#2: Range [4.2e-005, 1.28]
Dims [256, 256]

513

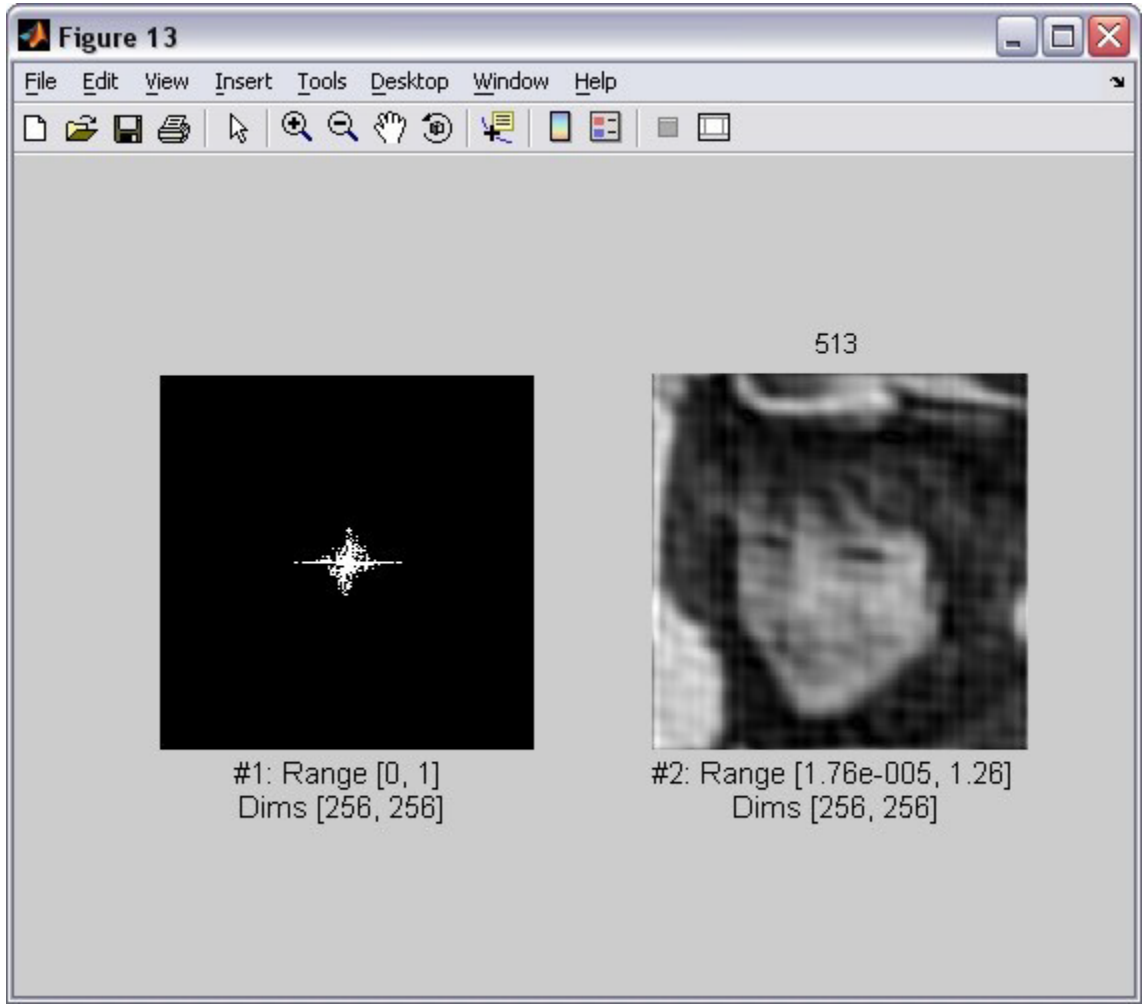


Figure 13
File Edit View Insert Tools Desktop Window Help
513
#1: Range [0, 1]
Dims [256, 256]
#2: Range [1.76e-005, 1.26]
Dims [256, 256]

1025

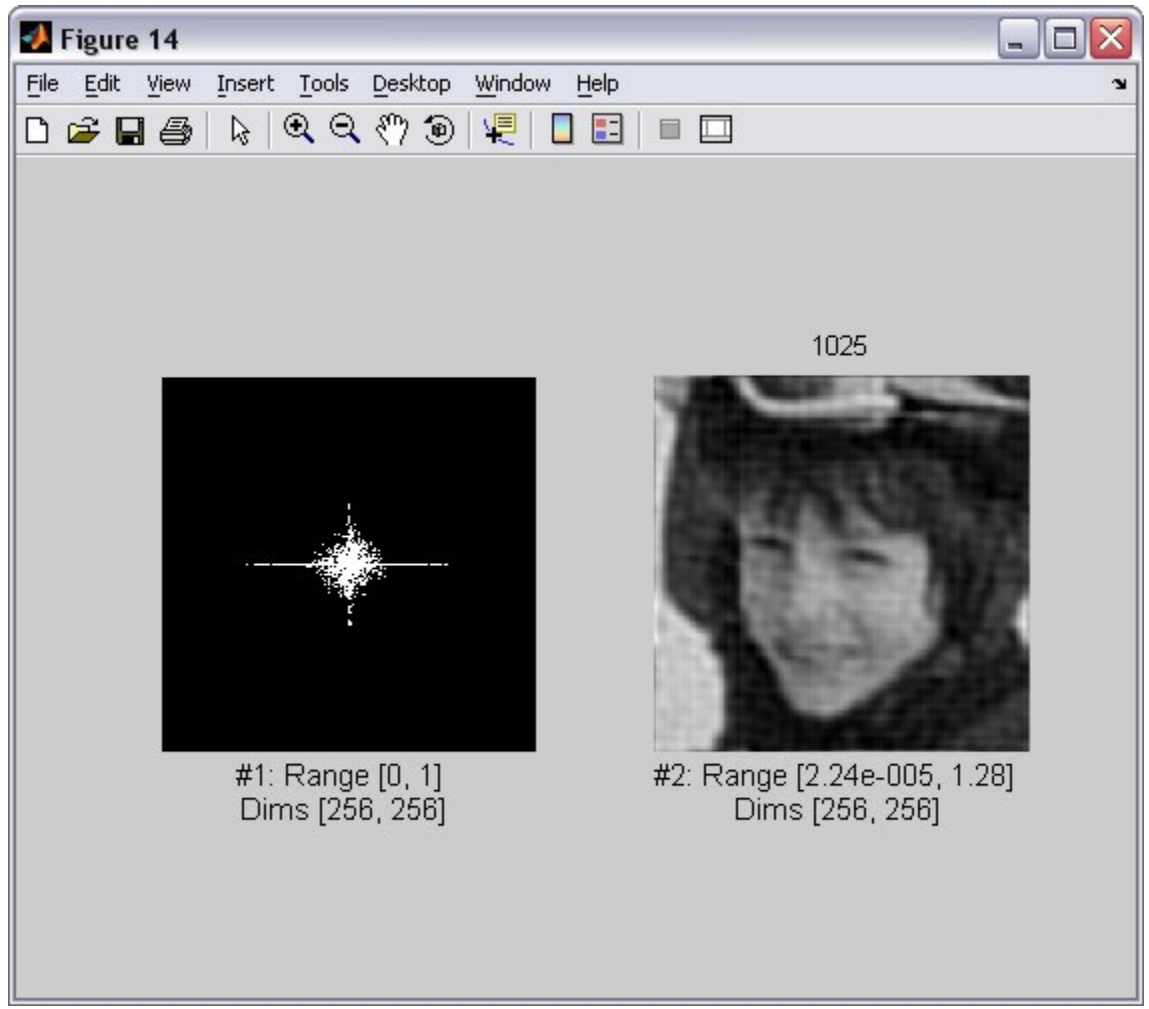


Figure 14
File Edit View Insert Tools Desktop Window Help
1025
#1: Range [0, 1]
Dims [256, 256]
#2: Range [2.24e-005, 1.28]
Dims [256, 256]

2049

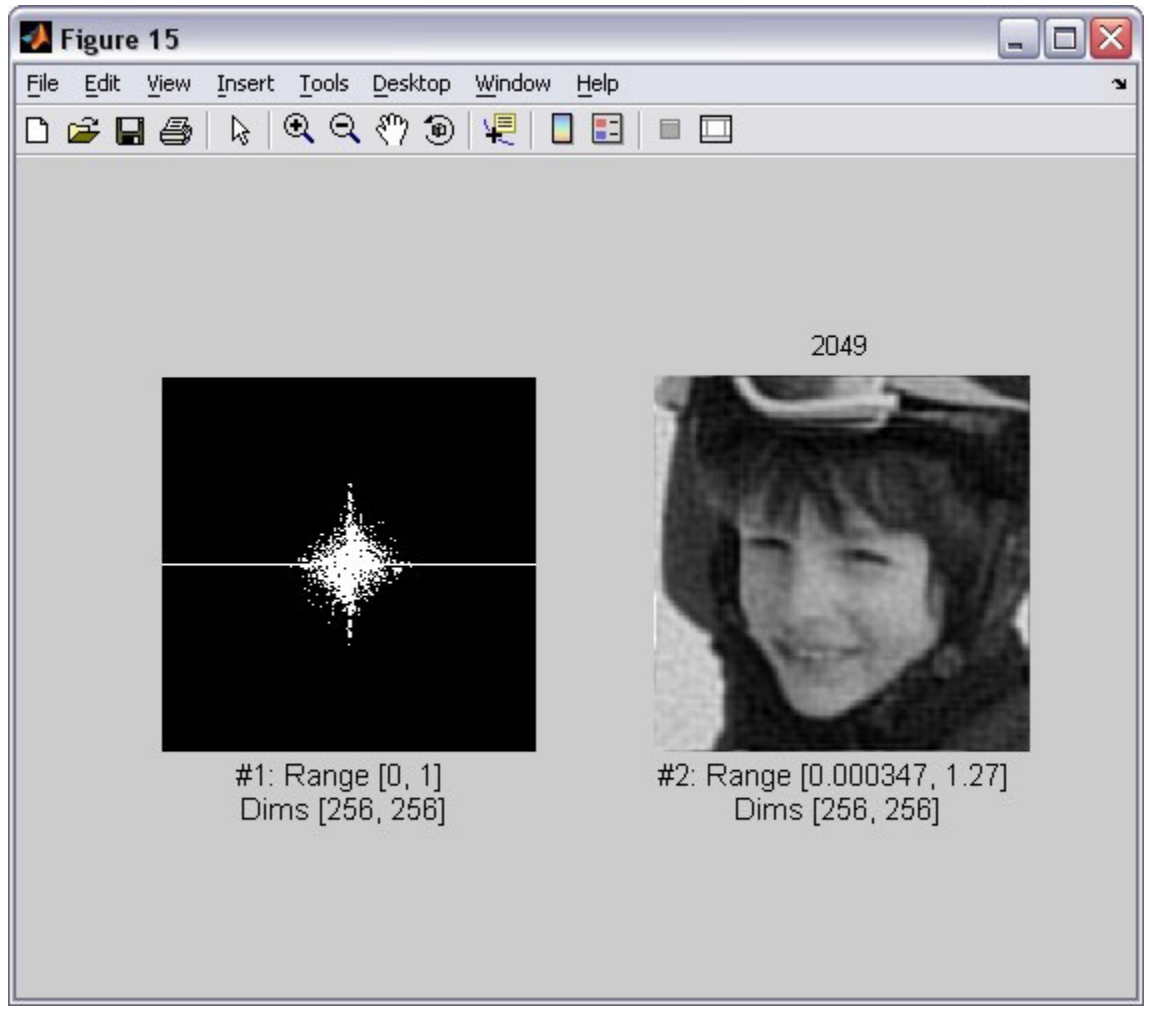


Figure 15
File Edit View Insert Tools Desktop Window Help
2049
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000347, 1.27]
Dims [256, 256]

4097

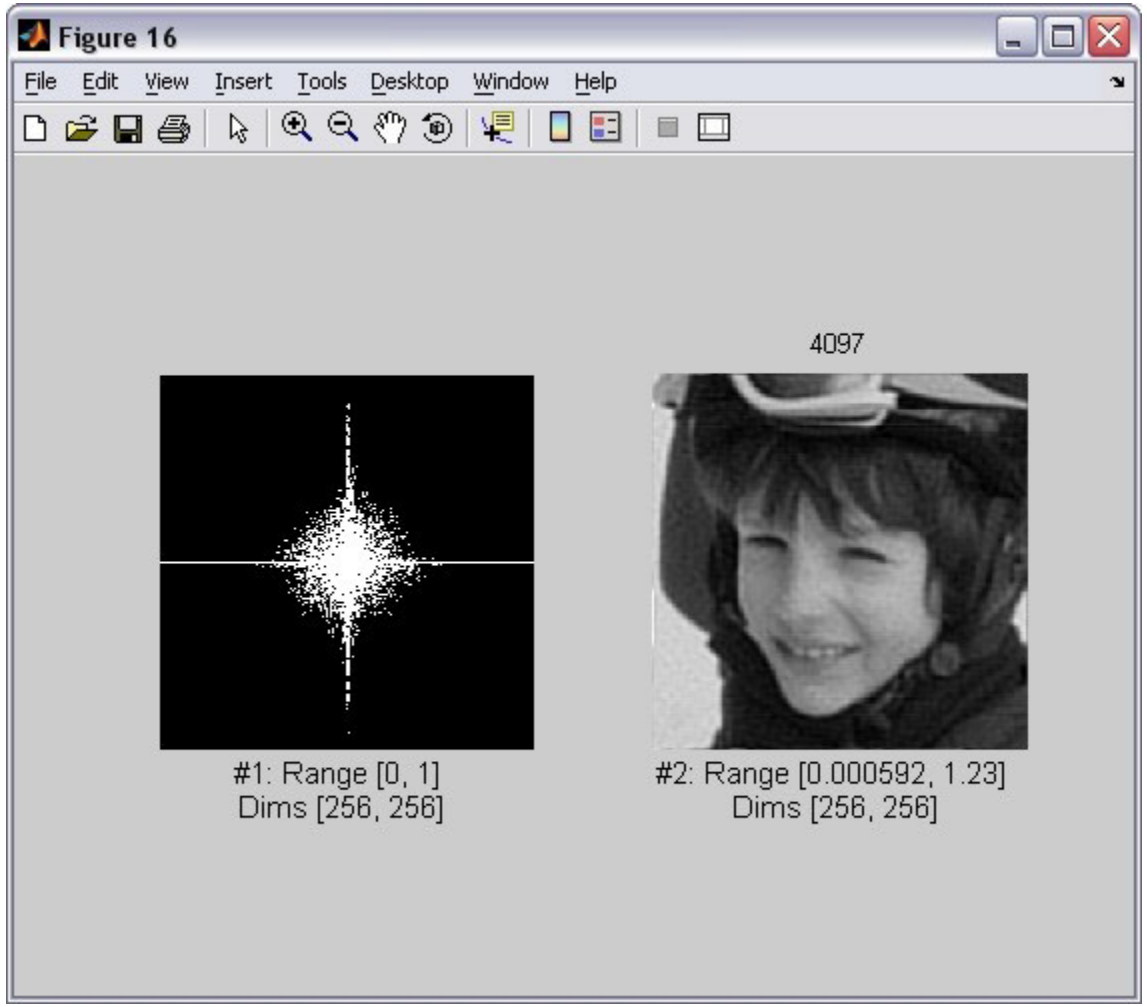


Figure 16
File Edit View Insert Tools Desktop Window Help
4097
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000592, 1.23]
Dims [256, 256]

8193

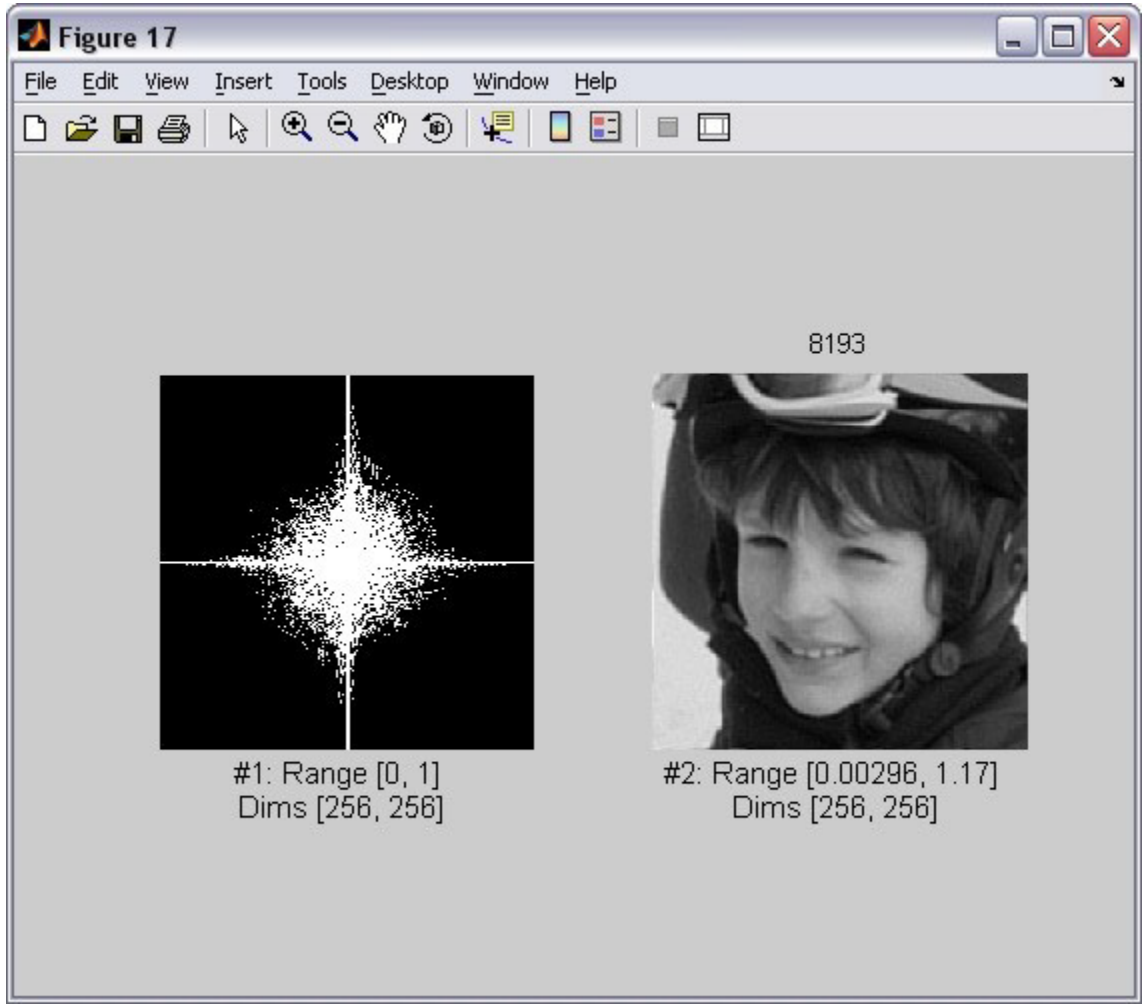


Figure 17
File Edit View Insert Tools Desktop Window Help
8193
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.00296, 1.17]
Dims [256, 256]

16385

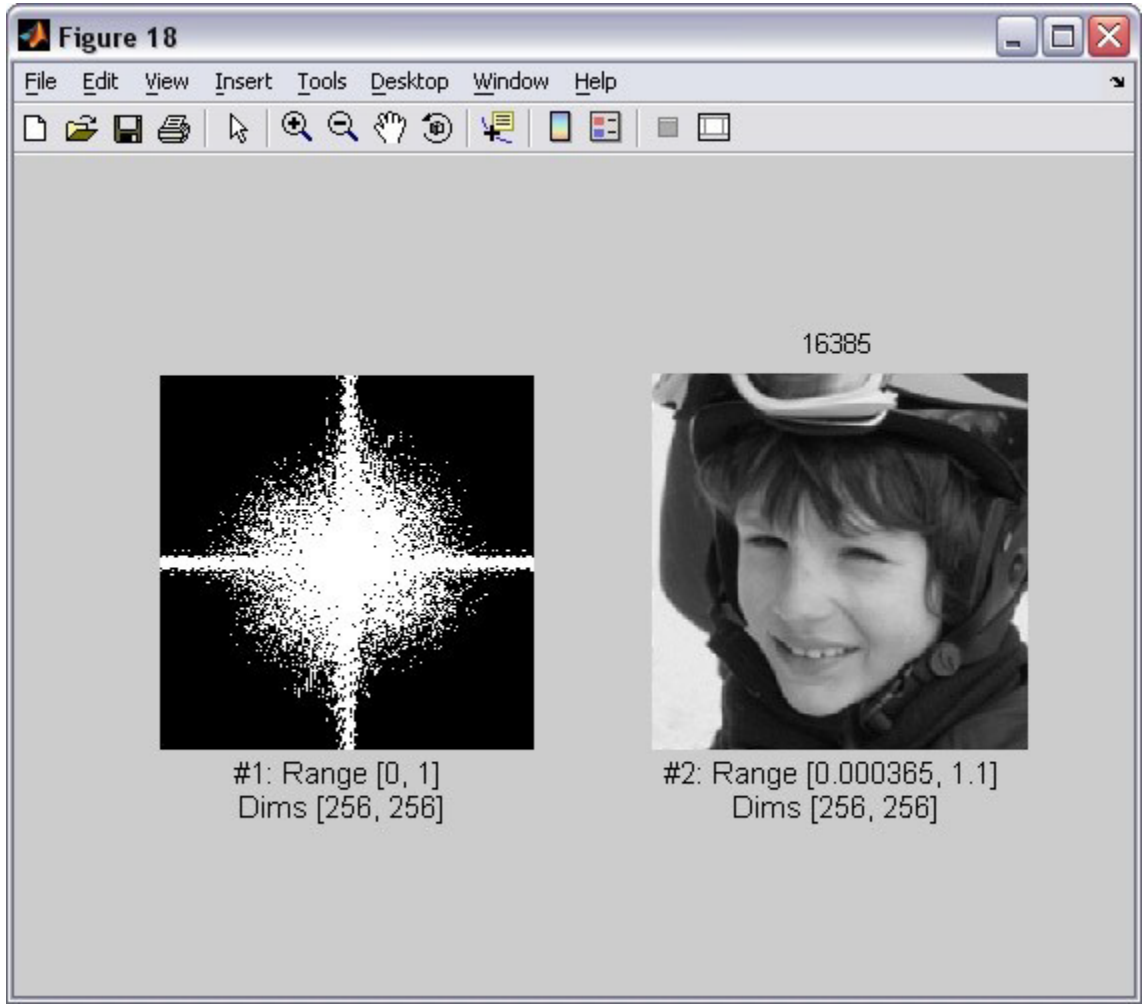


Figure 18
File Edit View Insert Tools Desktop Window Help
16385
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000365, 1.1]
Dims [256, 256]

32769

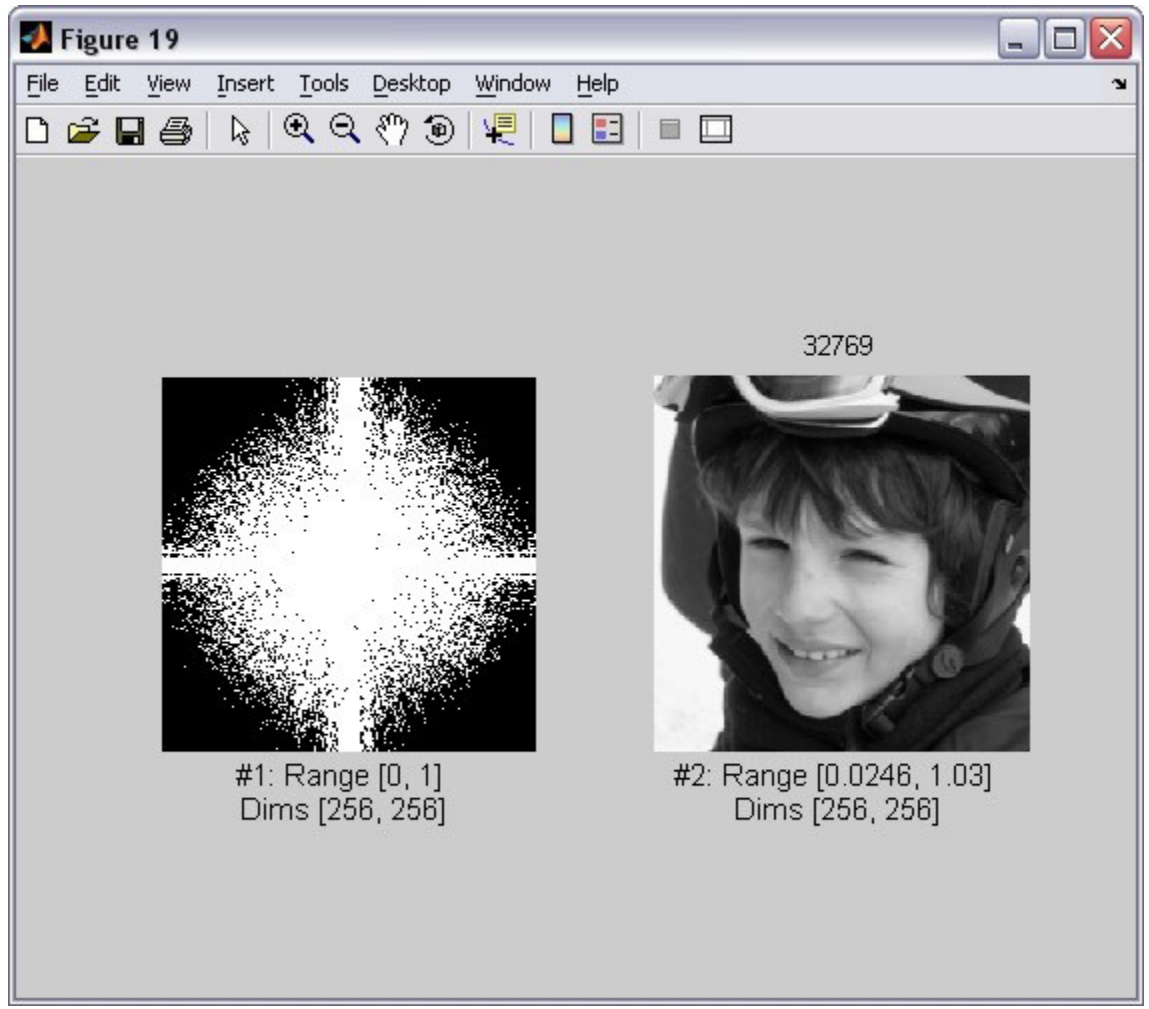


Figure 19
File Edit View Insert Tools Desktop Window Help
32769
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.0246, 1.03]
Dims [256, 256]

65536

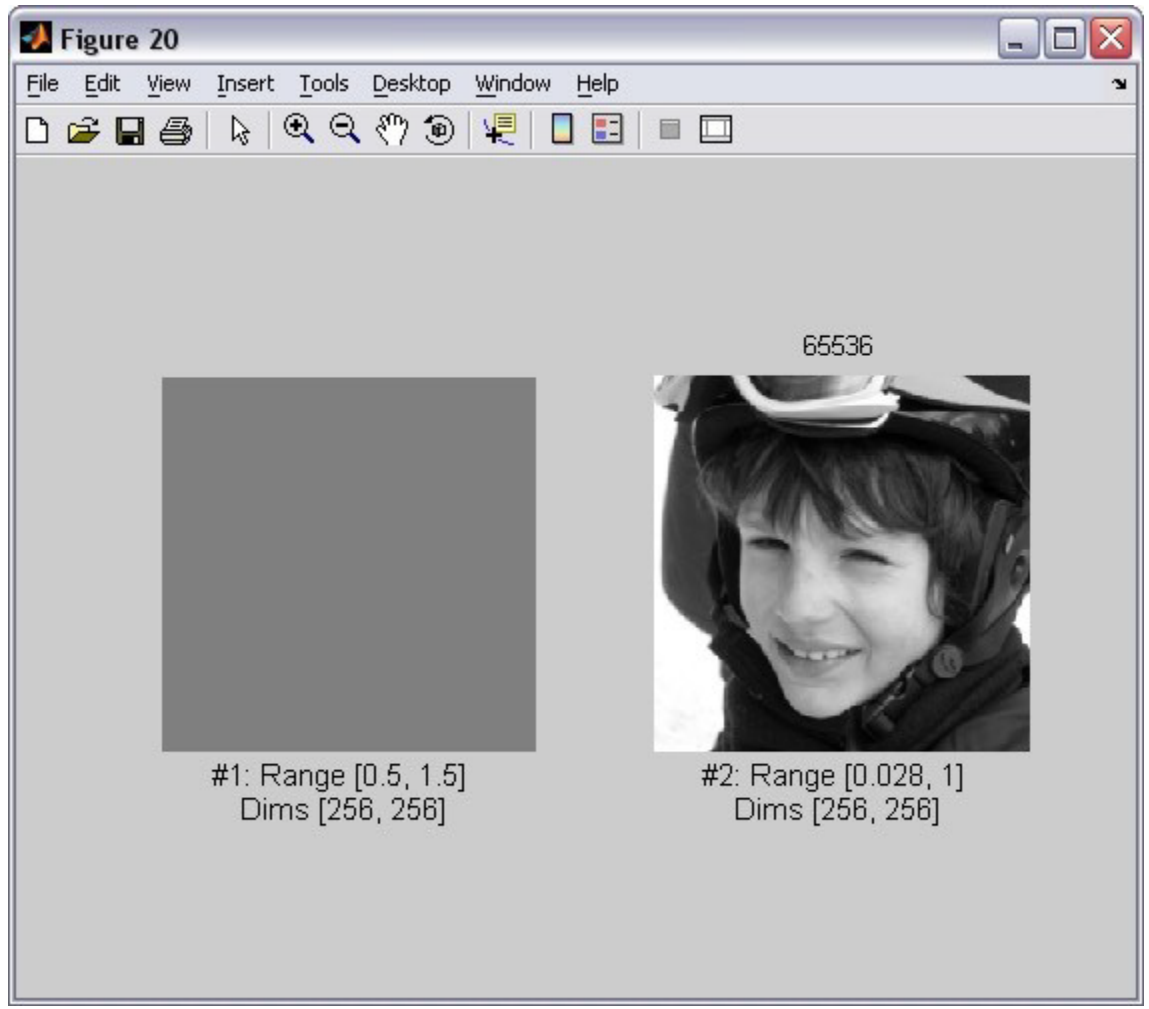
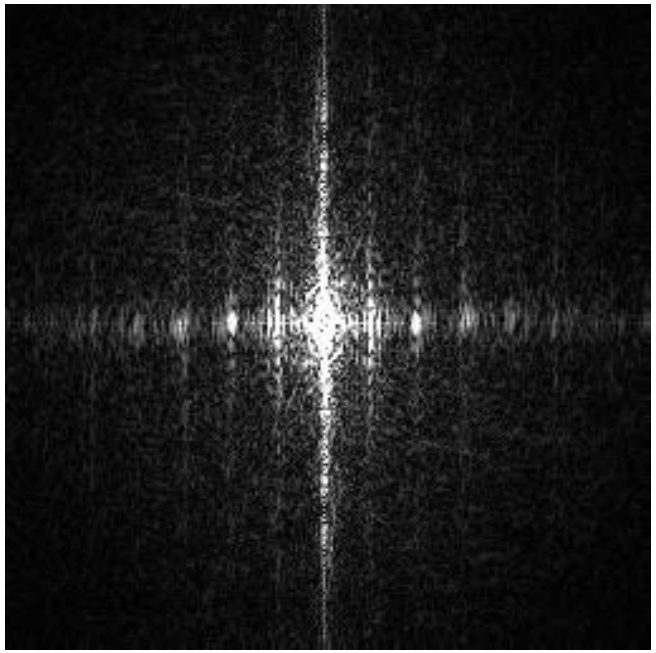
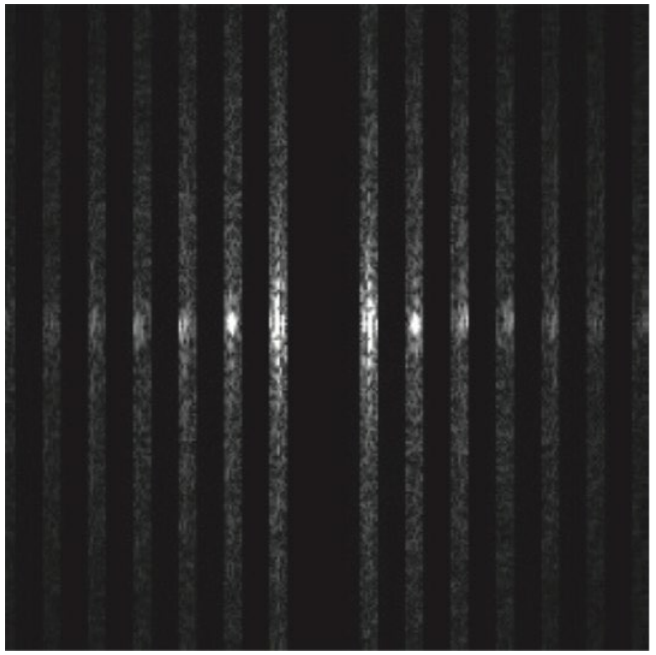


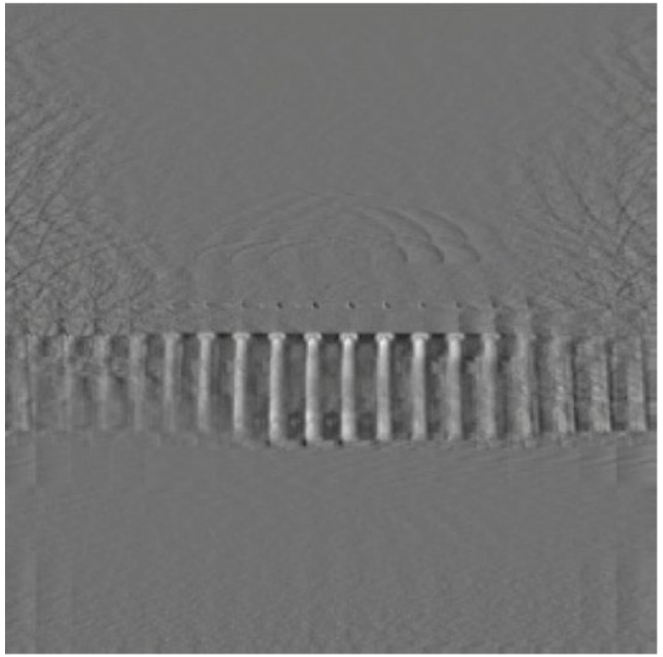
Figure 20
File Edit View Insert Tools Desktop Window Help
65536
#1: Range [0.5, 1.5]
Dims [256, 256]
#2: Range [0.028, 1]
Dims [256, 256]

DFT

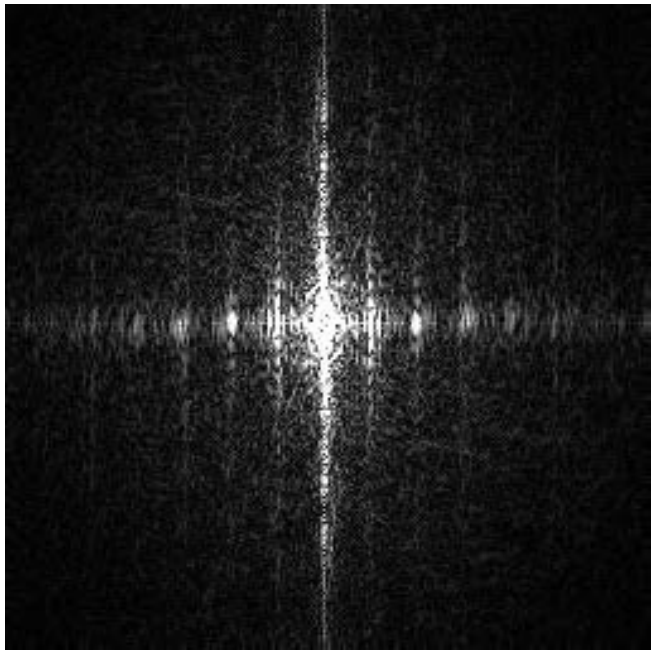


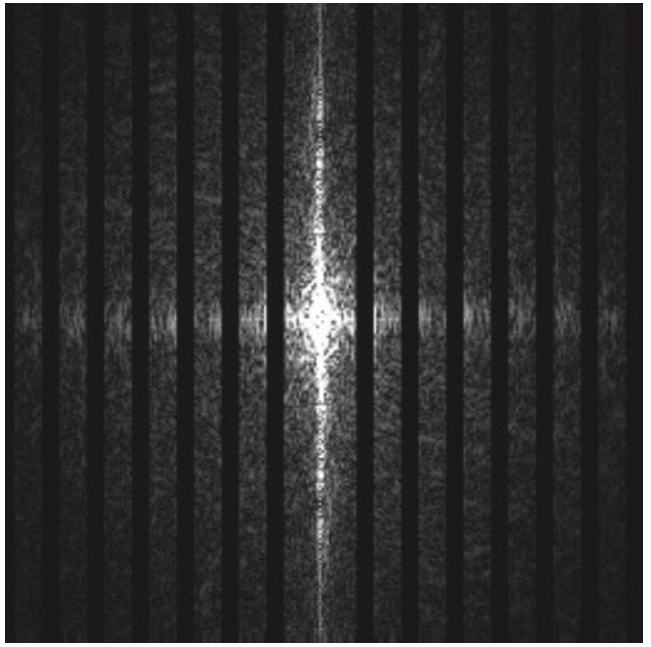


DFT^{-1}



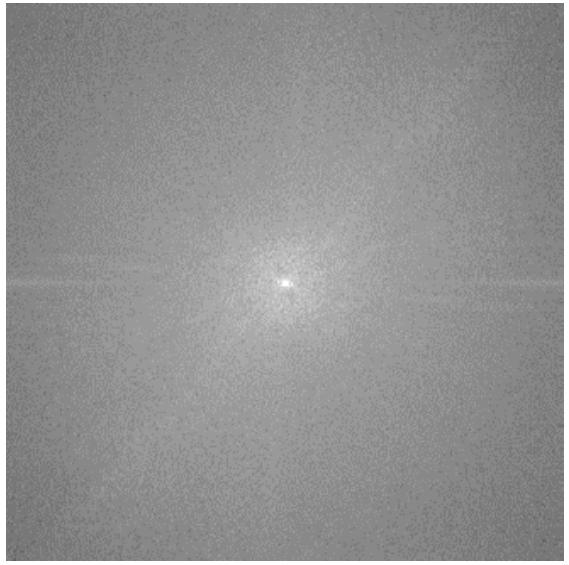
DFT

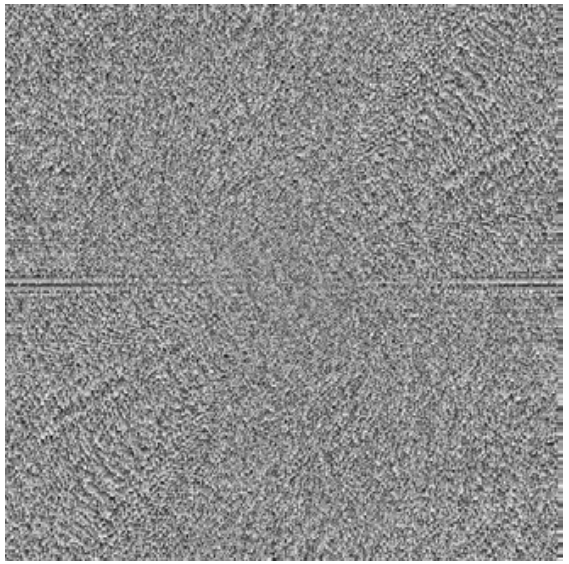


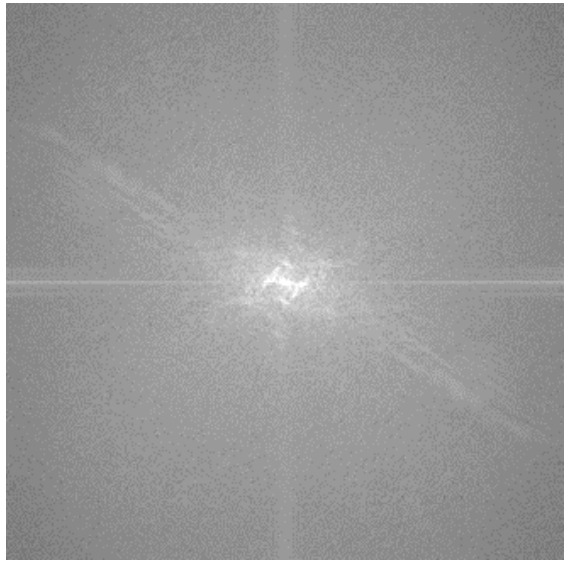


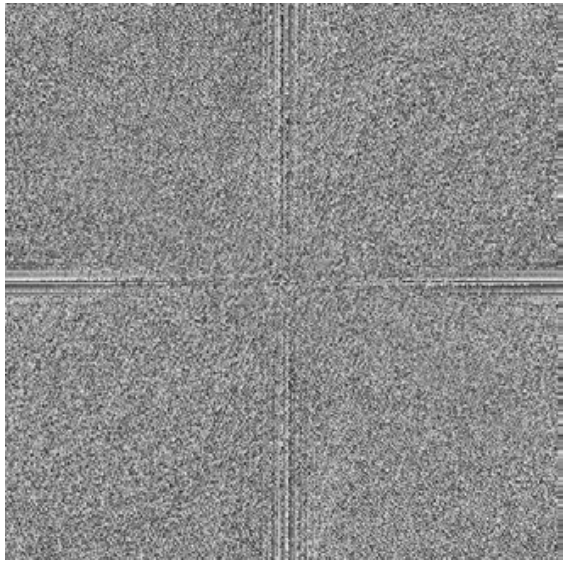
DFT^{-1}

Fourier transforms of natural images



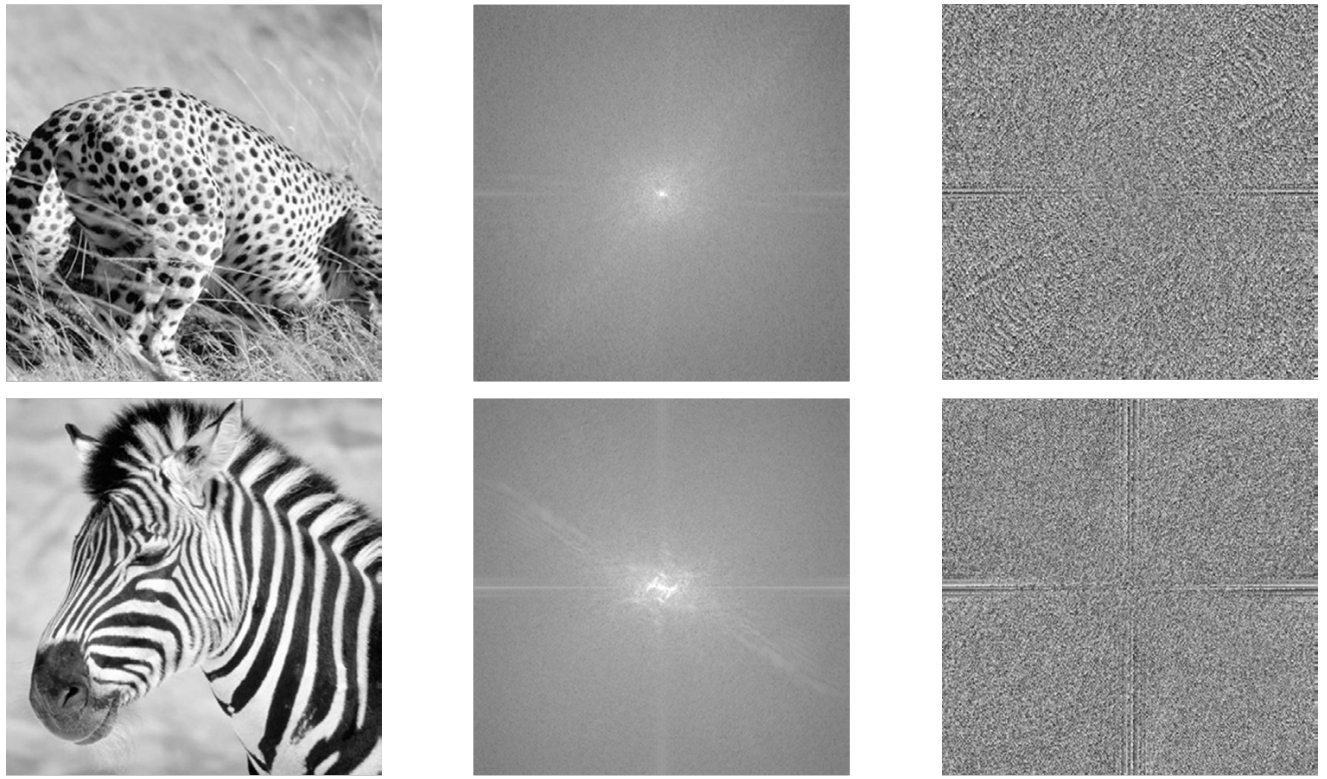






original amplitude phase

Fourier transforms of natural images

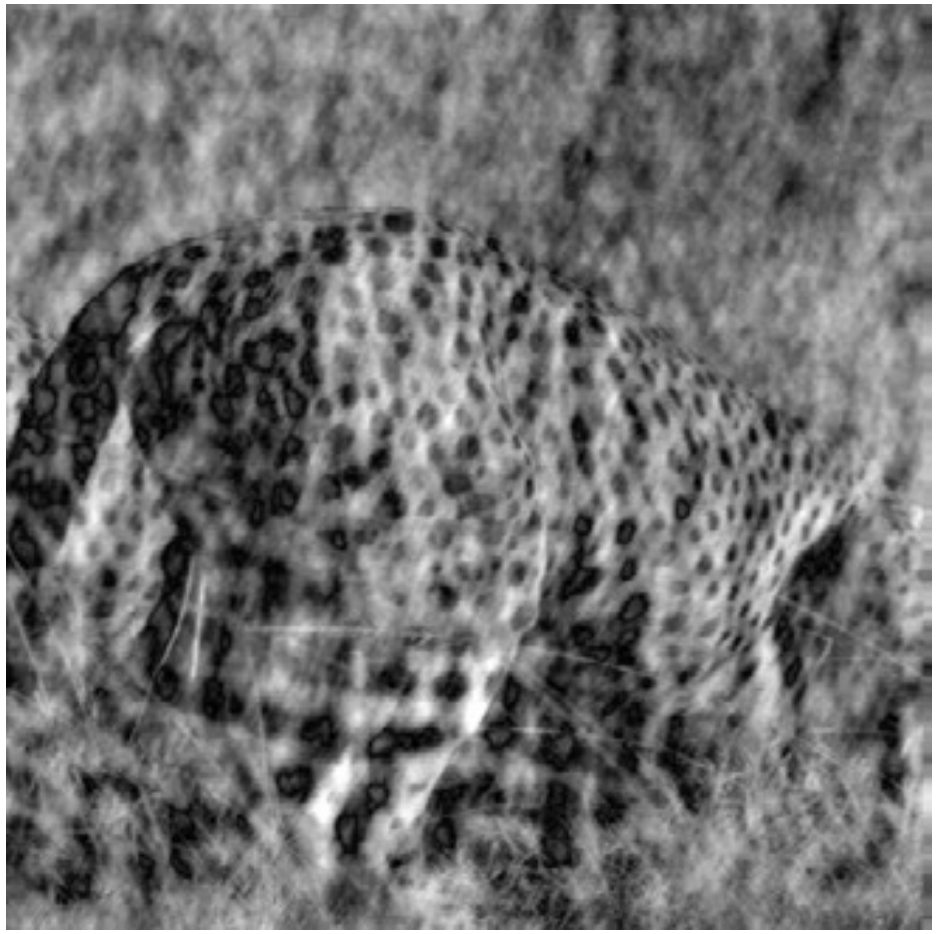


Curious fact:

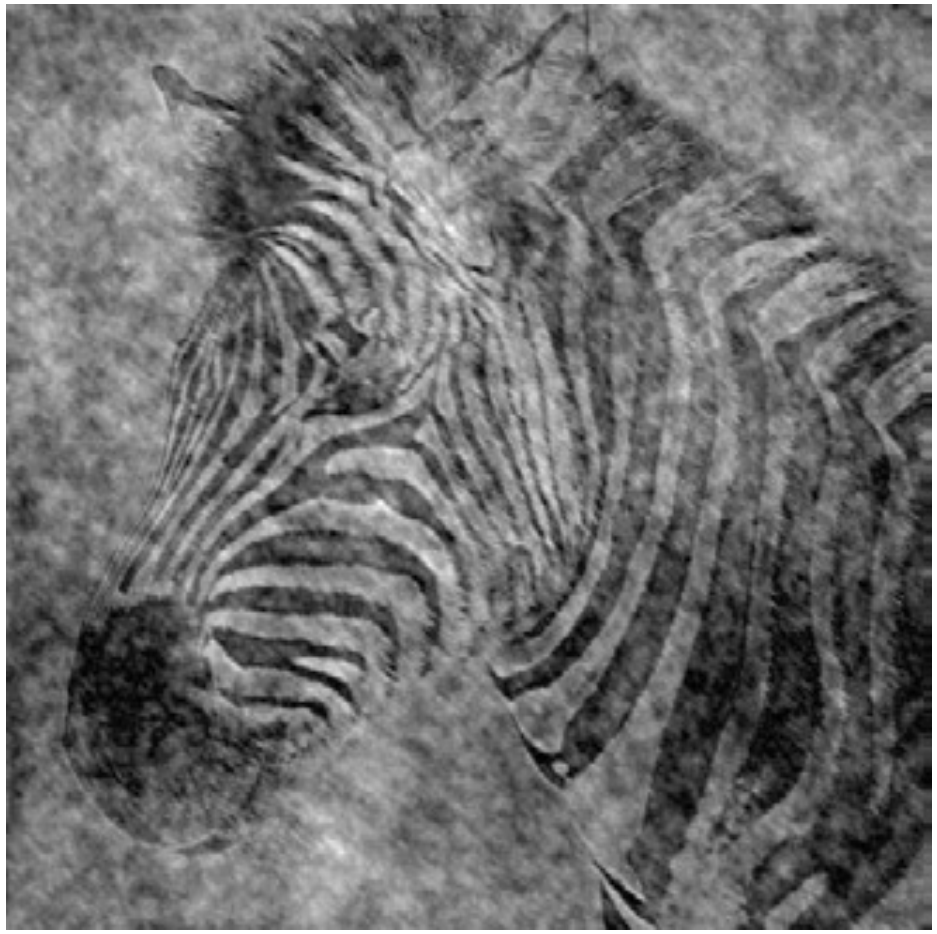
- All natural images have about the same magnitude transform
- Hence, phase seems to matter, but magnitude largely doesn't

Fourier transforms of natural images

Image phase matters!



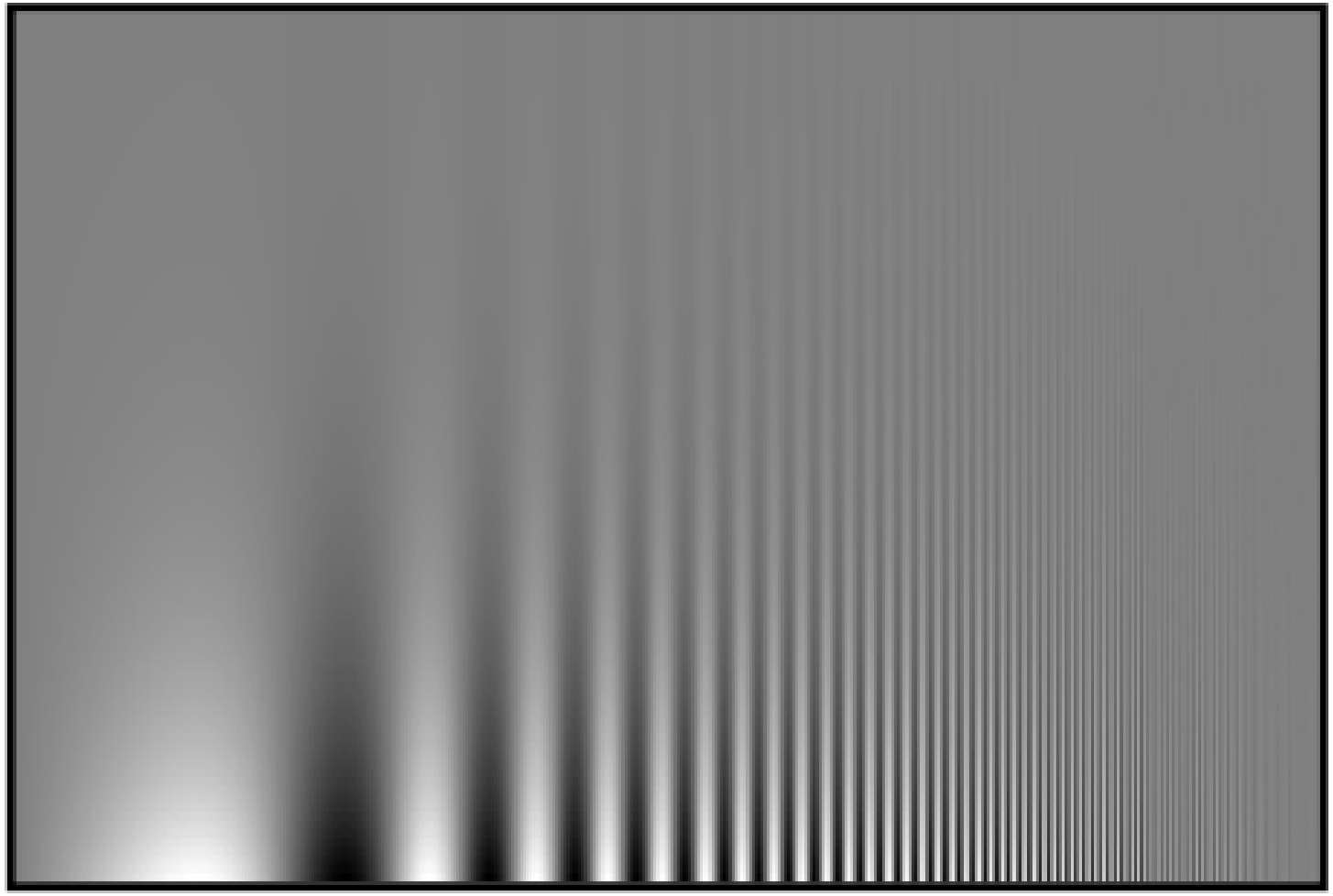
cheetah phase with zebra amplitude



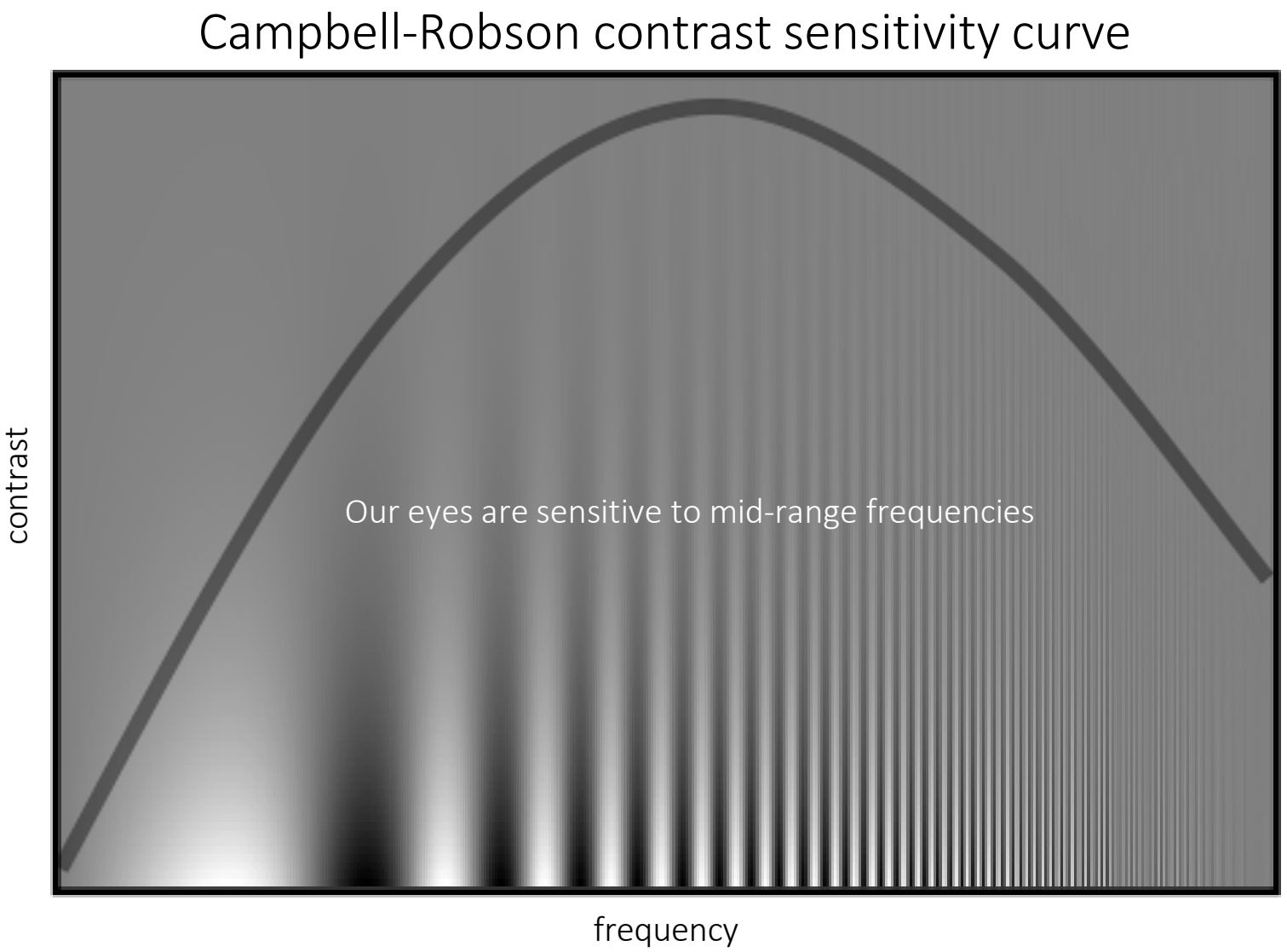
zebra phase with cheetah amplitude

Variable frequency sensitivity

Experiment: Where do you see the stripes?



Variable frequency sensitivity



- Early processing in humans filters for various orientations and scales of frequency
- Perceptual cues in the mid frequencies dominate perception

Frequency-domain filtering

The convolution theorem

Why do we care about all this?

The Fourier transform of the convolution of two functions is the product of their Fourier transforms:

$$\mathcal{F}\{g * h\} = \mathcal{F}\{g\}\mathcal{F}\{h\}$$

The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms:

$$\mathcal{F}^{-1}\{gh\} = \mathcal{F}^{-1}\{g\} * \mathcal{F}^{-1}\{h\}$$

Convolution in spatial domain is equivalent to multiplication in frequency domain!

Convolution for 1D continuous signals

What do we use convolution for?

Definition of linear shift-invariant filtering as convolution:

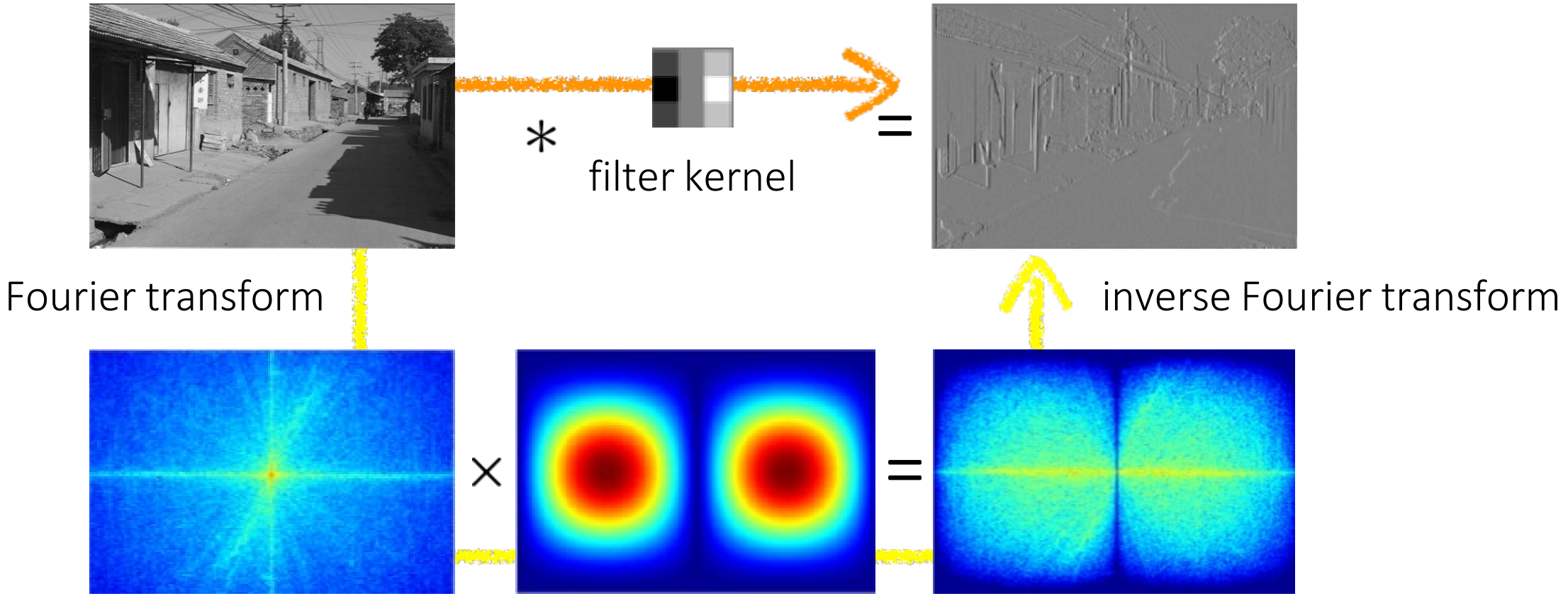
$$(f * g)(x) = \int_{-\infty}^{\infty} f(y)g(x-y)dy$$

filtered signal ↗ filter ↖ input signal

Using the convolution theorem, we can interpret and implement all types of linear shift-invariant filtering as multiplication in frequency domain.

Why implement convolution in frequency domain?

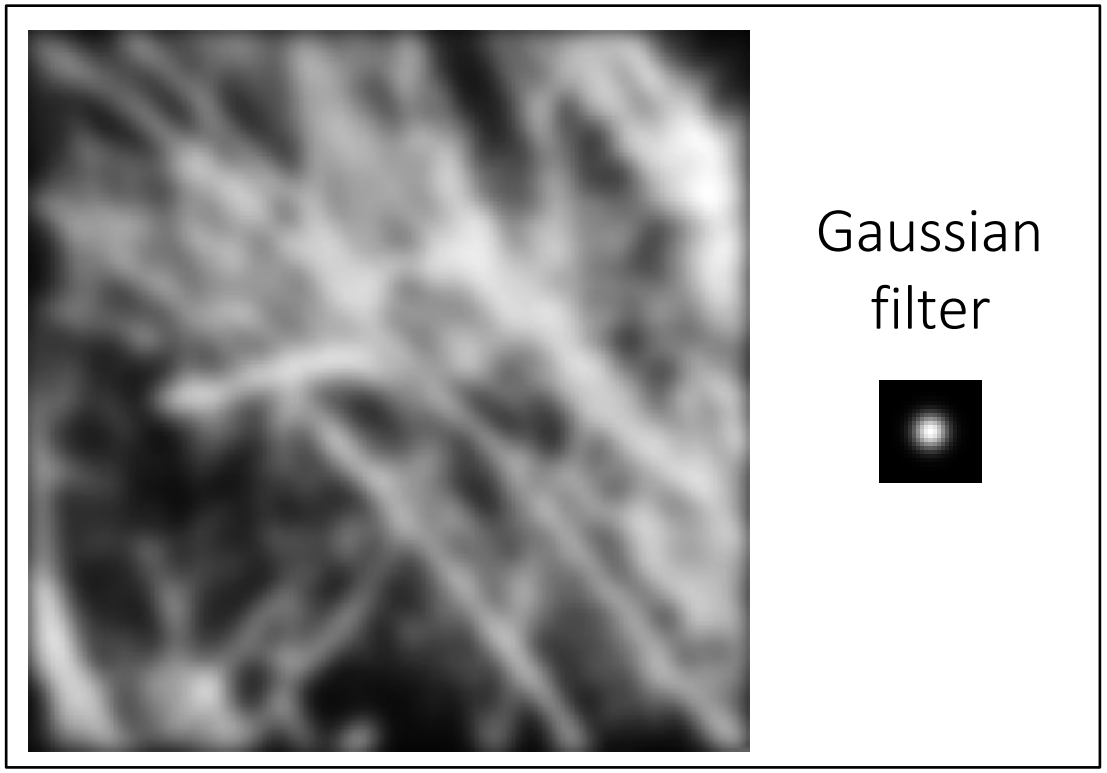
Spatial domain filtering

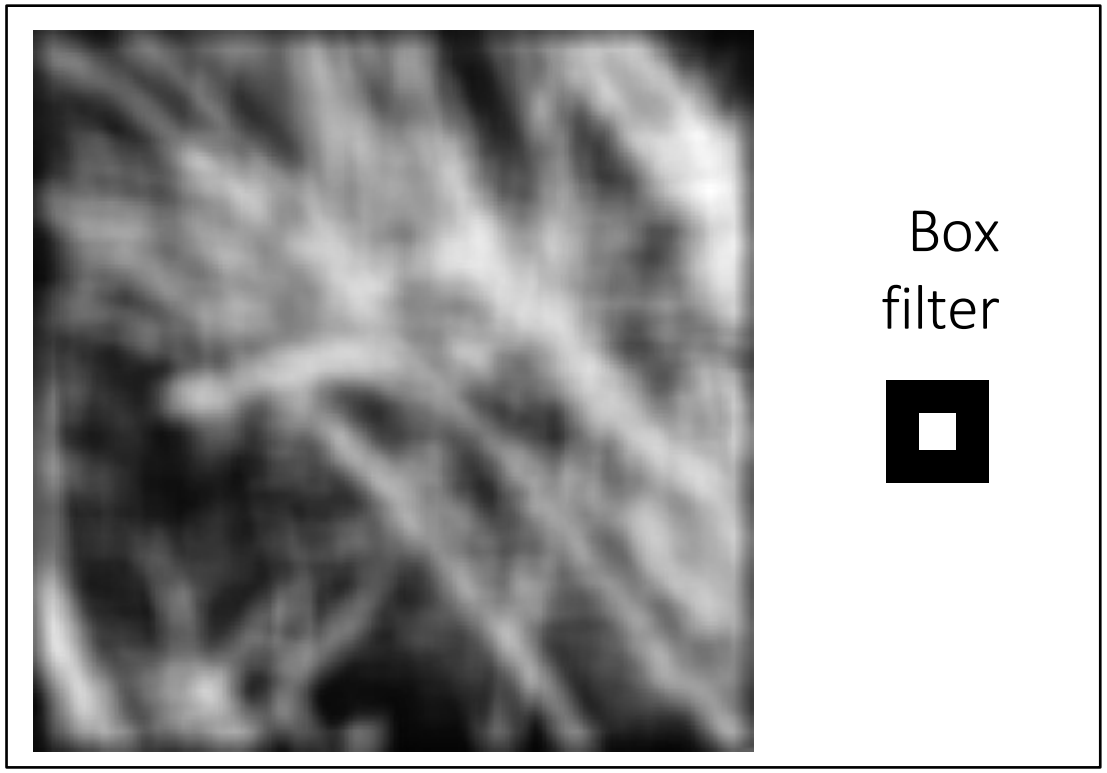


Frequency domain filtering

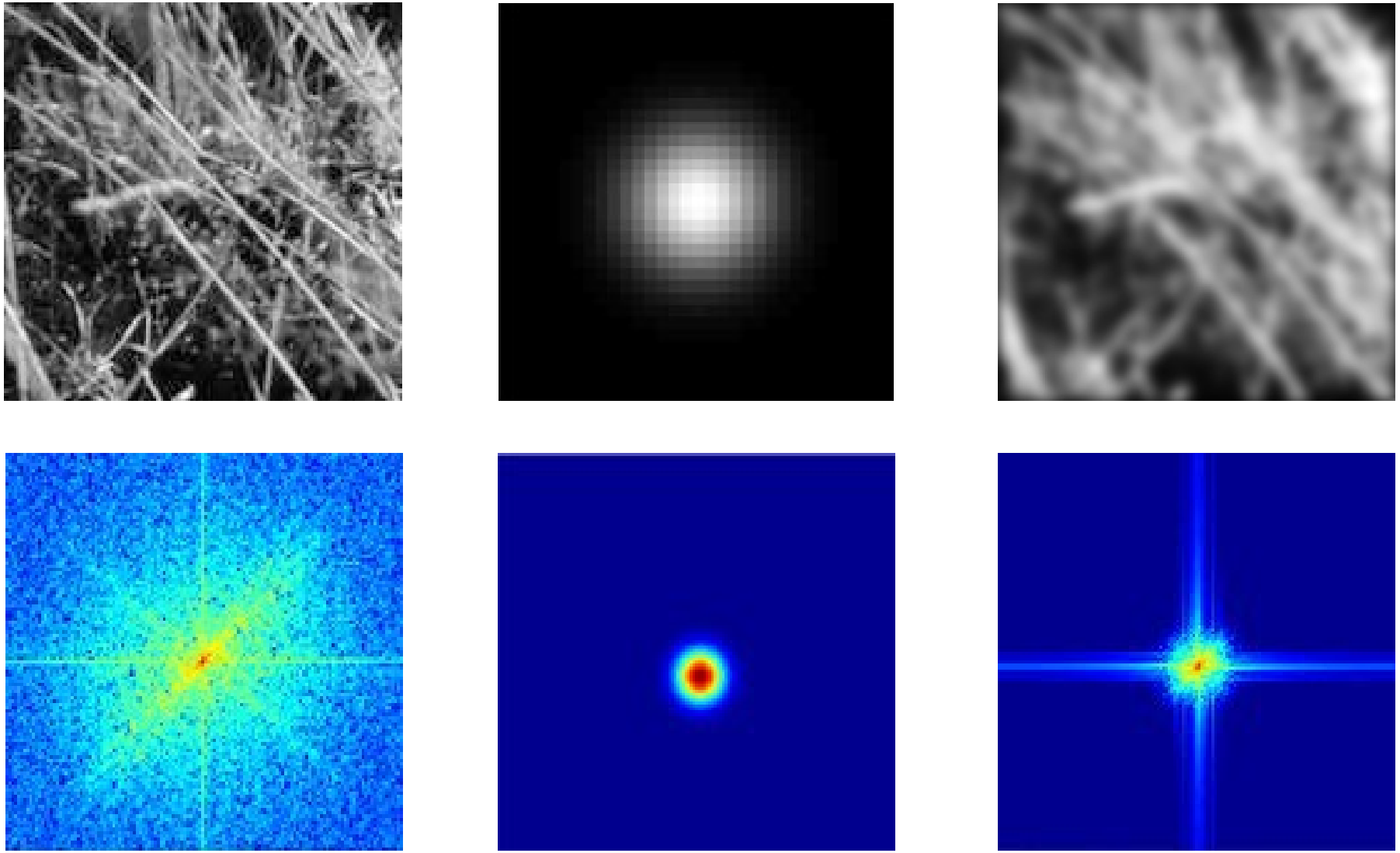
Revisiting blurring

Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

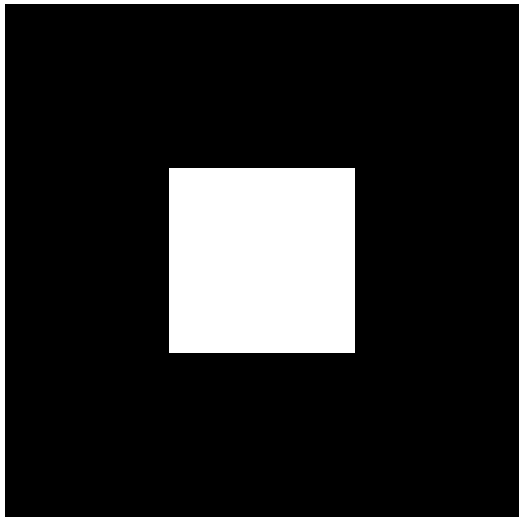
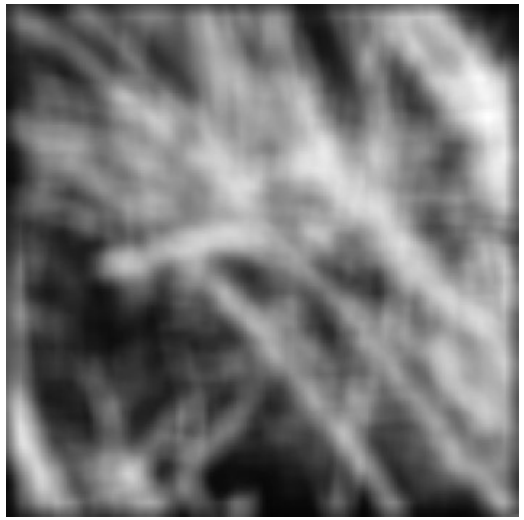
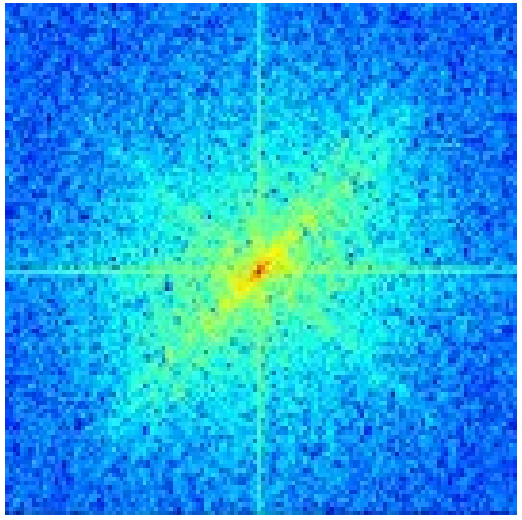
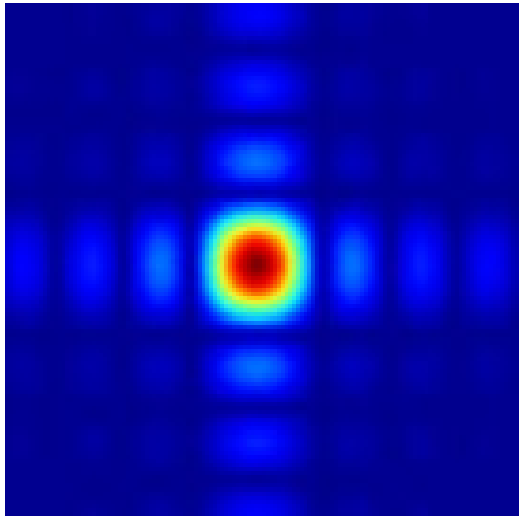
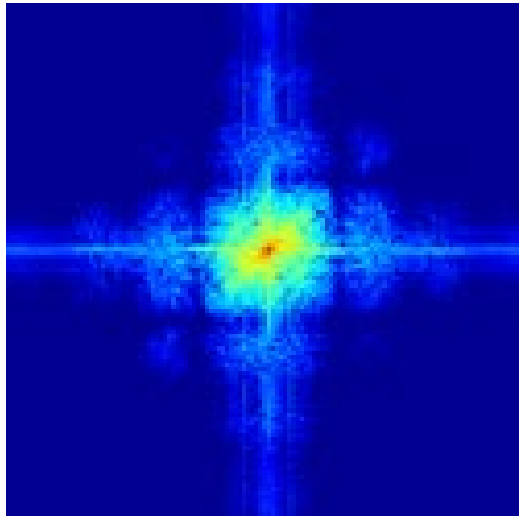




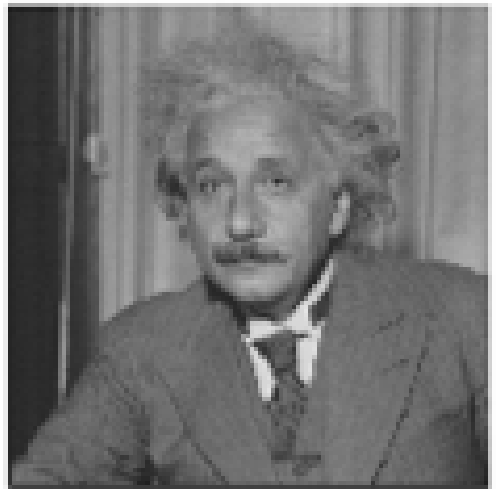
Gaussian blur



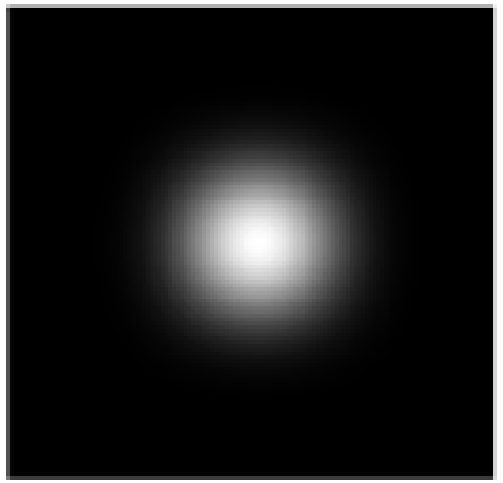
Box blur

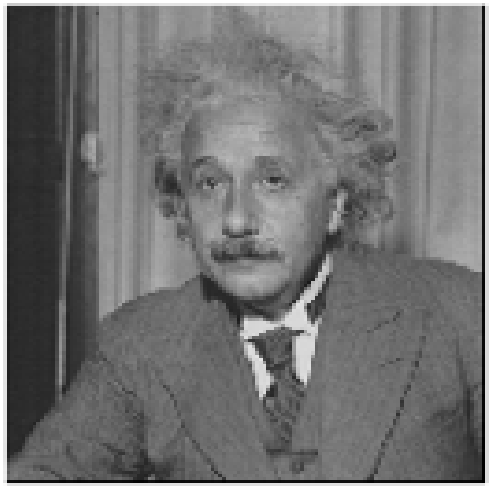







More filtering examples

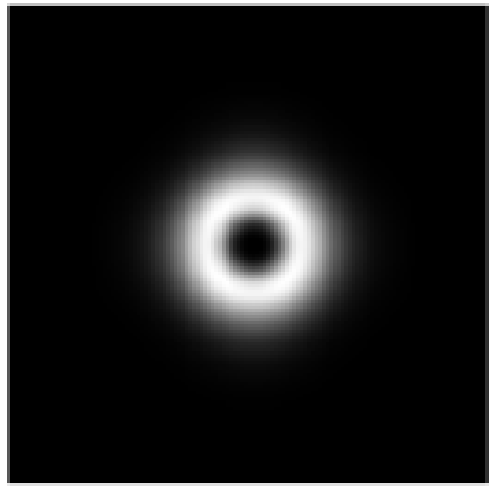


?



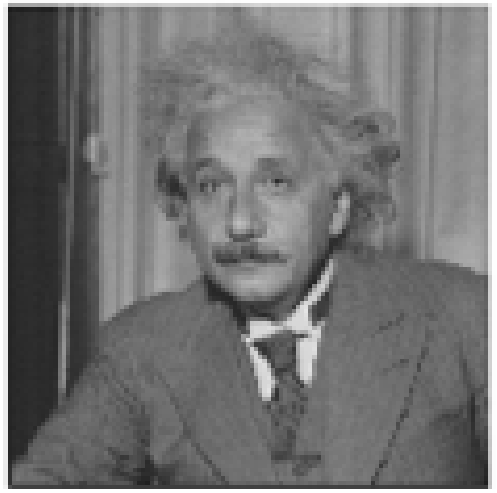


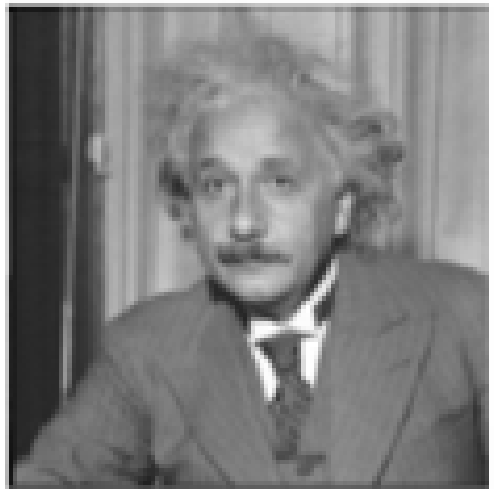
?



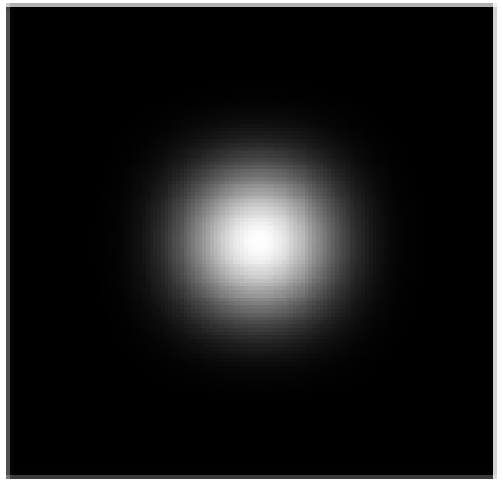
filters shown in frequency-domain

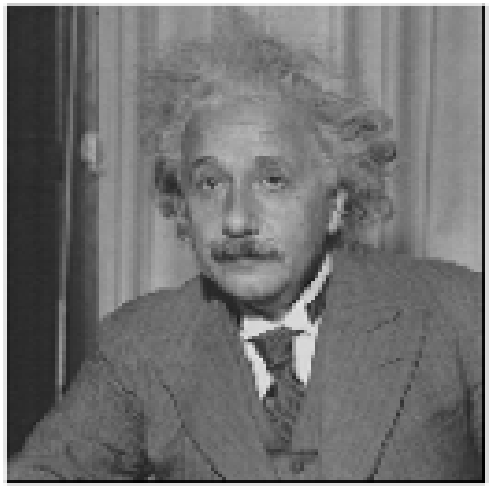
More filtering examples



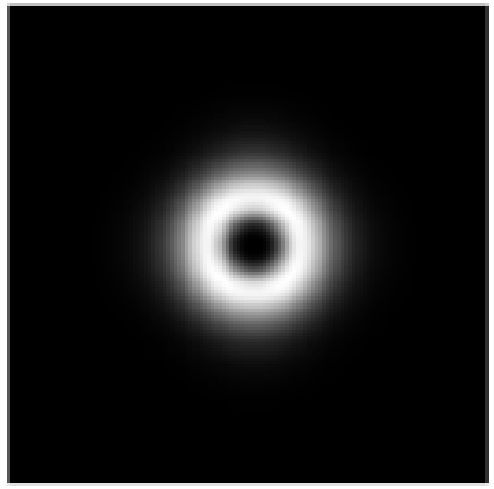


low-pass





band-pass

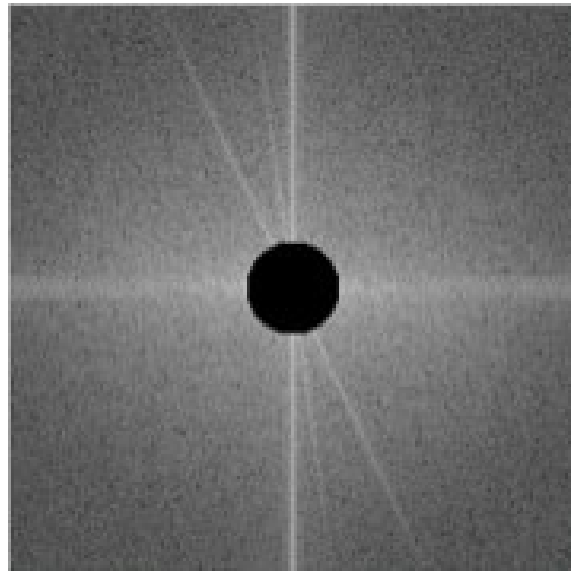


filters shown in frequency-domain

More filtering examples

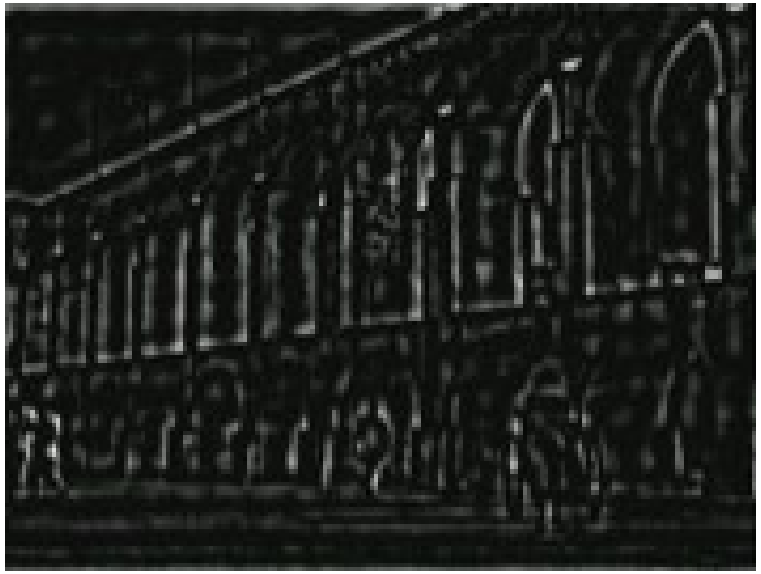
?

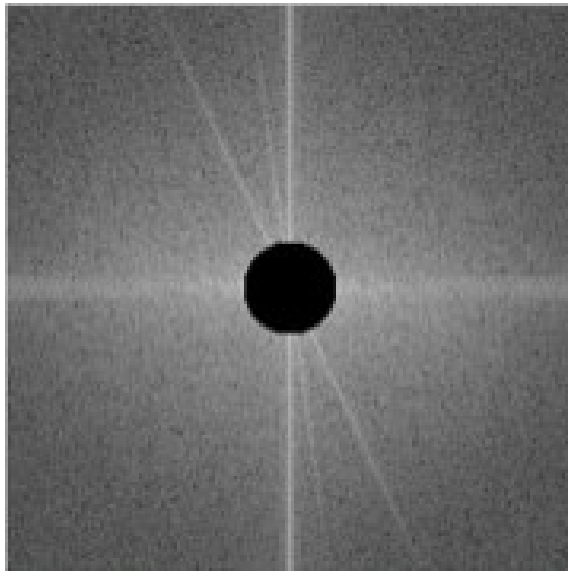
high-pass



More filtering examples

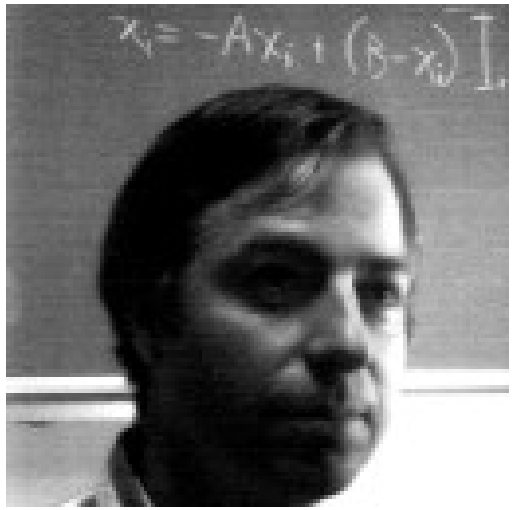
high-pass



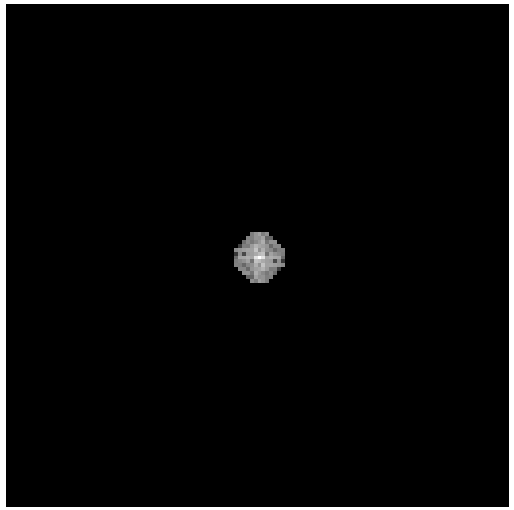


More filtering examples

original image

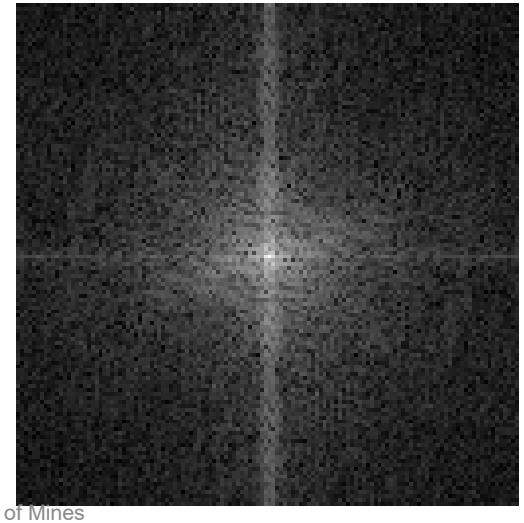


low-pass filter



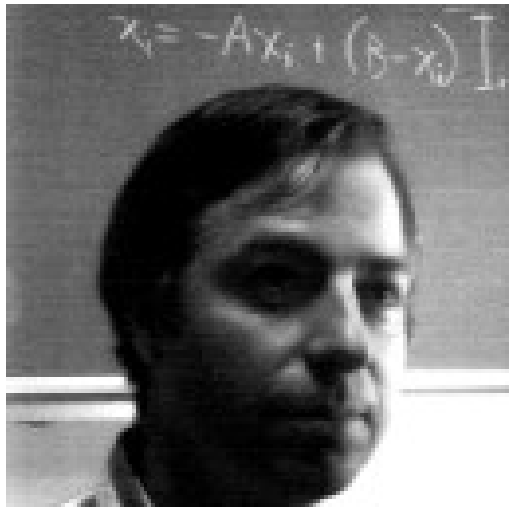
?

frequency magnitude

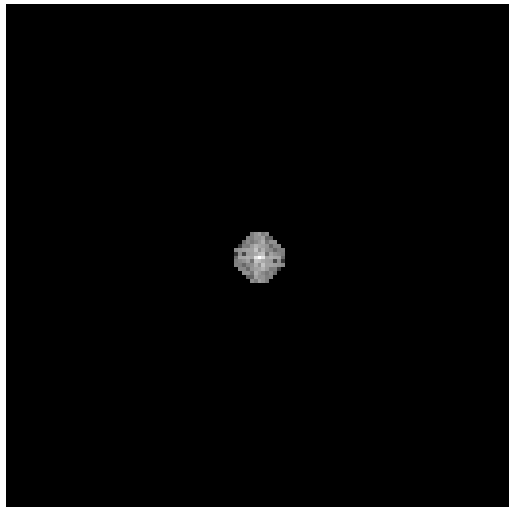


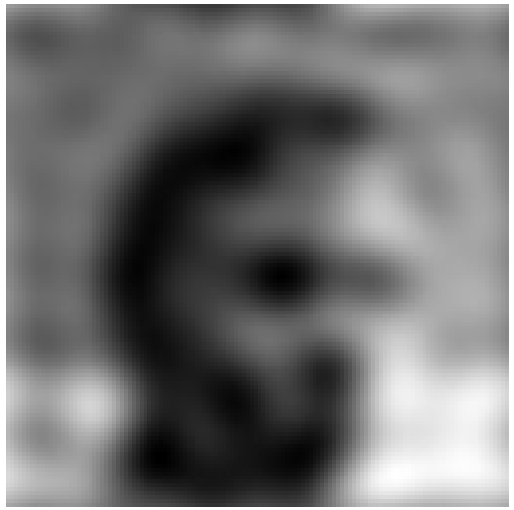
More filtering examples

original image

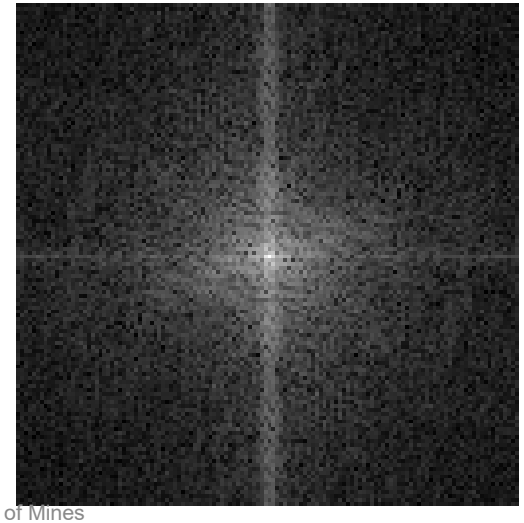


low-pass filter



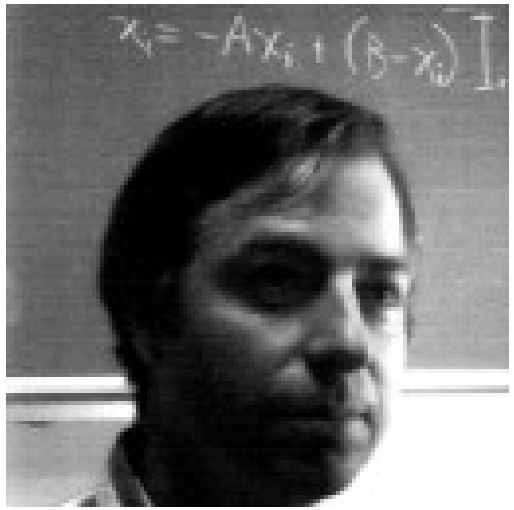


frequency magnitude

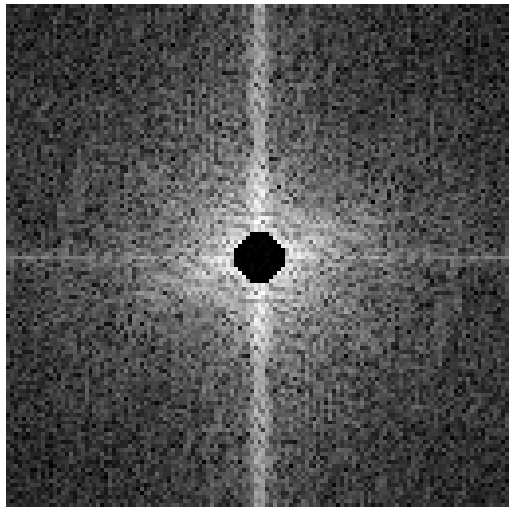


More filtering examples

original image

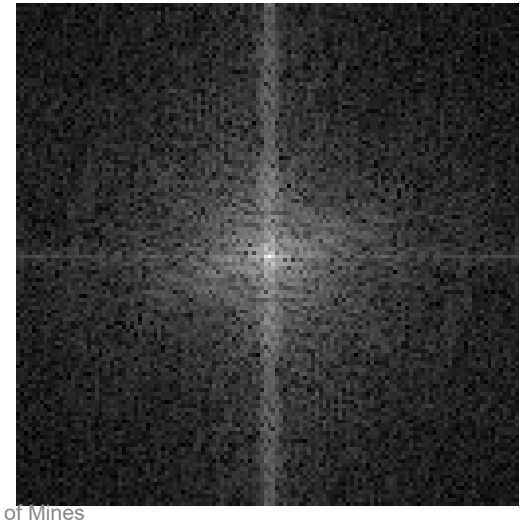


high-pass filter



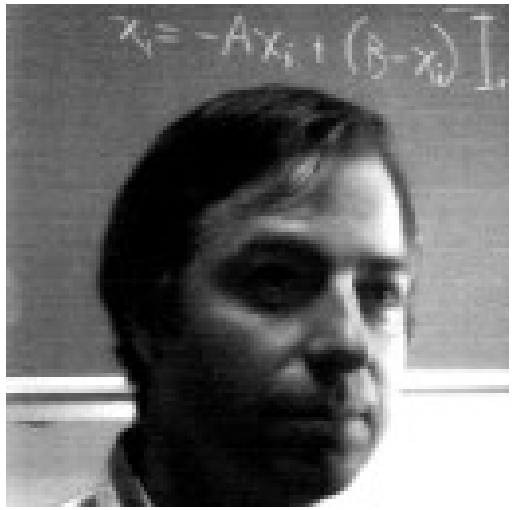
?

frequency magnitude

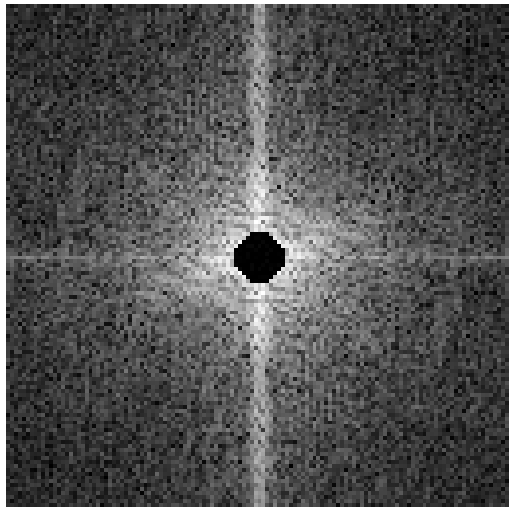


More filtering examples

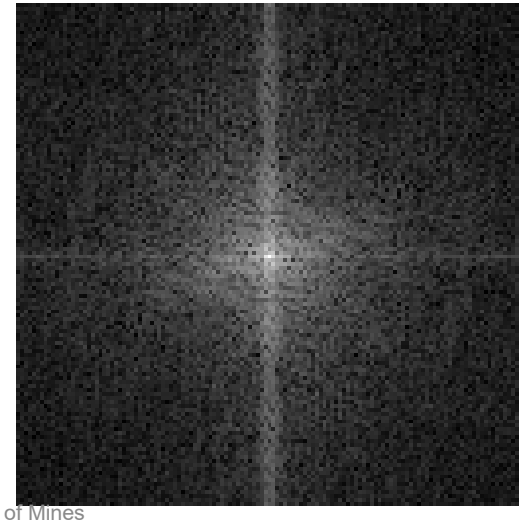
original image



high-pass filter

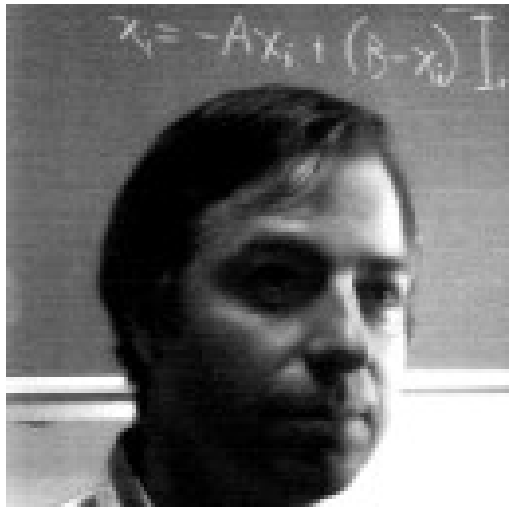


frequency magnitude

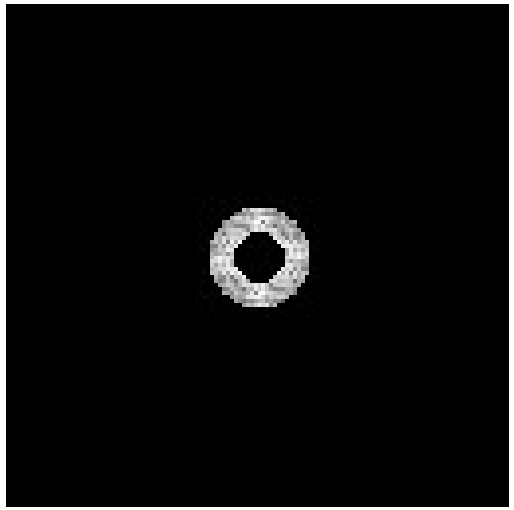


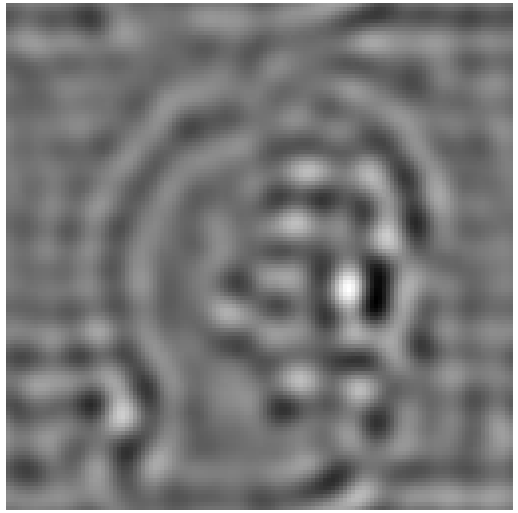
More filtering examples

original image

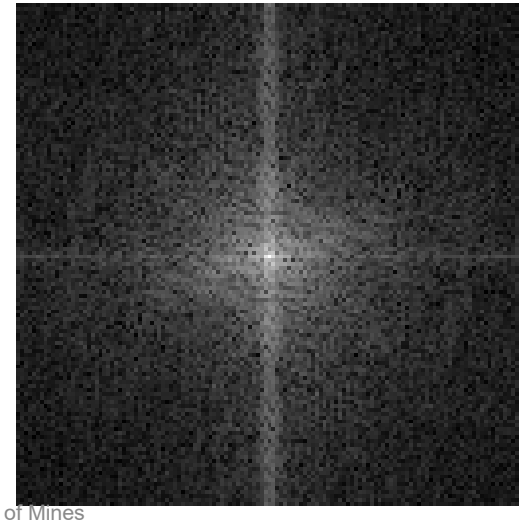


mid-pass filter



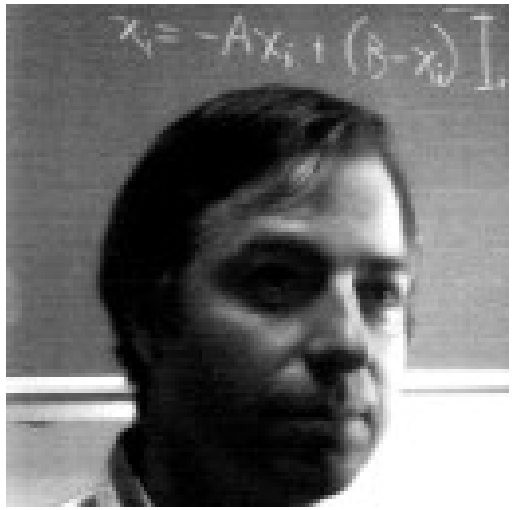


frequency magnitude

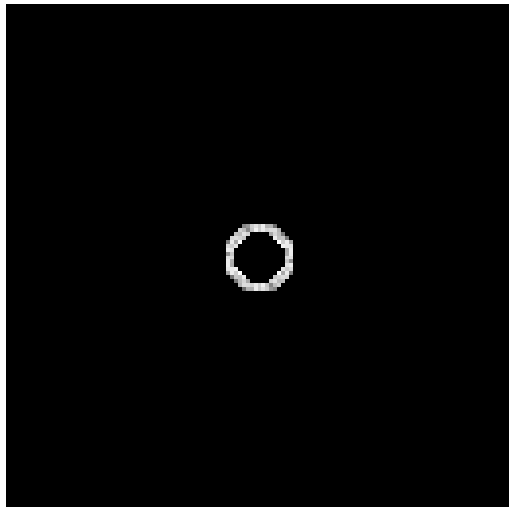


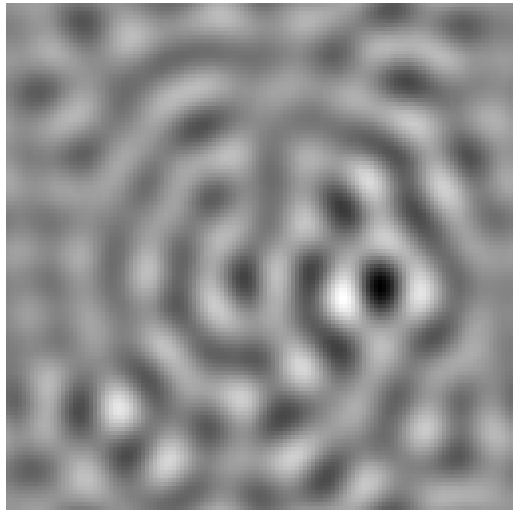
More filtering examples

original image

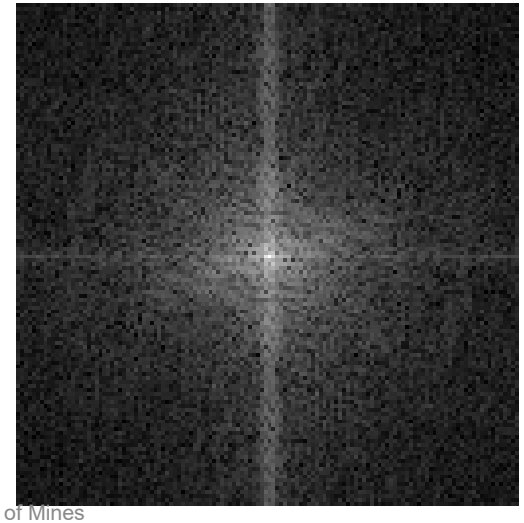


mid-pass filter



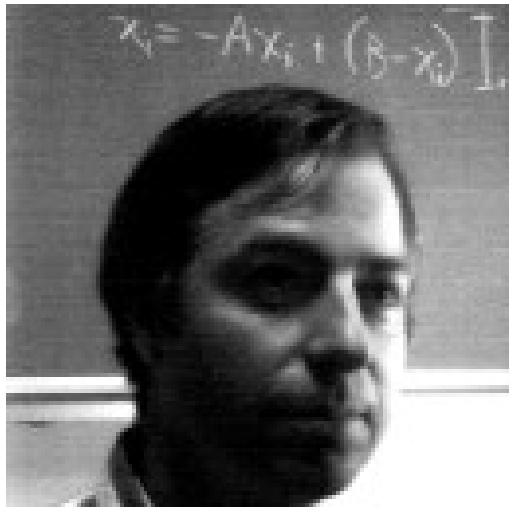


frequency magnitude

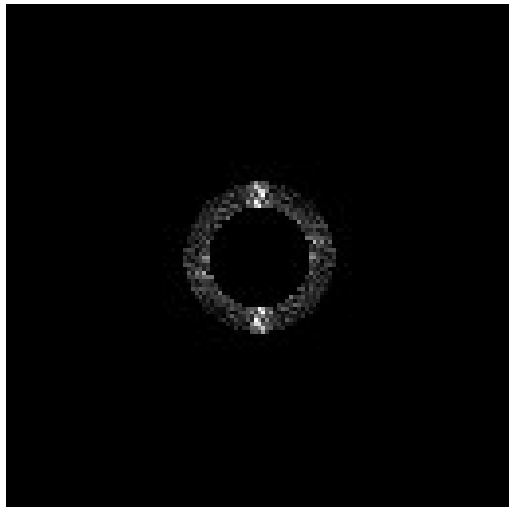


More filtering examples

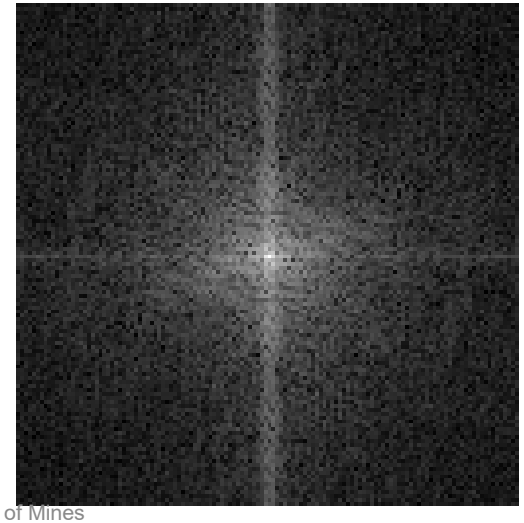
original image



mid-pass filter

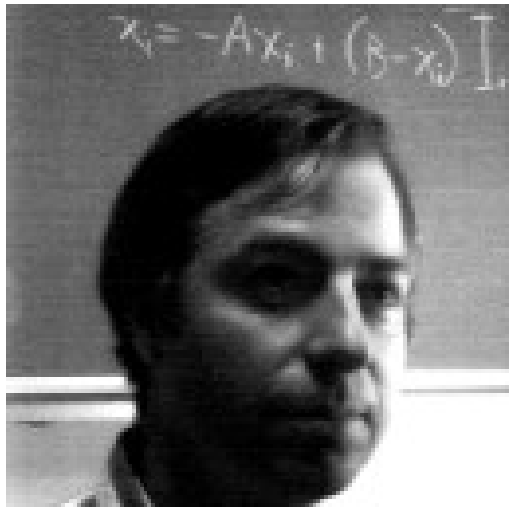


frequency magnitude

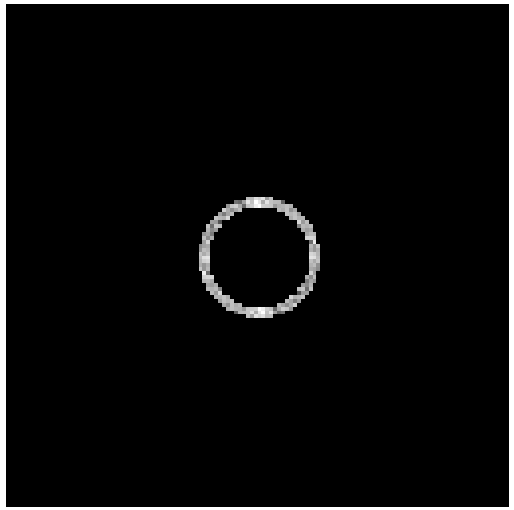


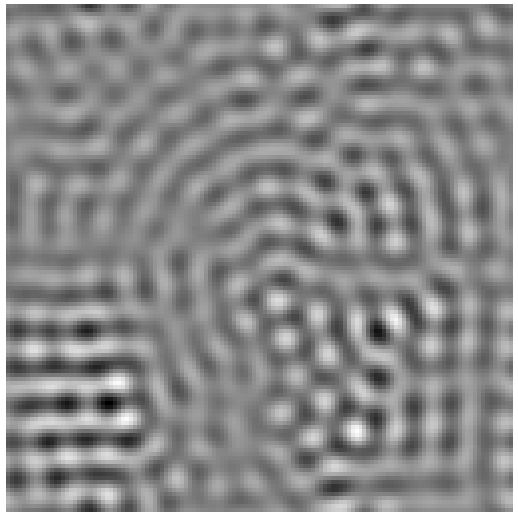
More filtering examples

original image

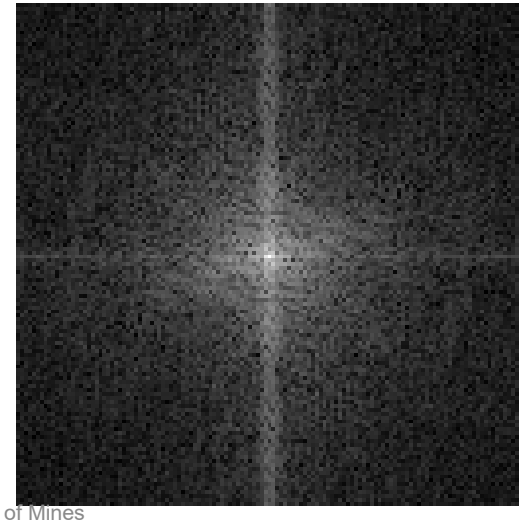


mid-pass filter





frequency magnitude



Revisiting sampling

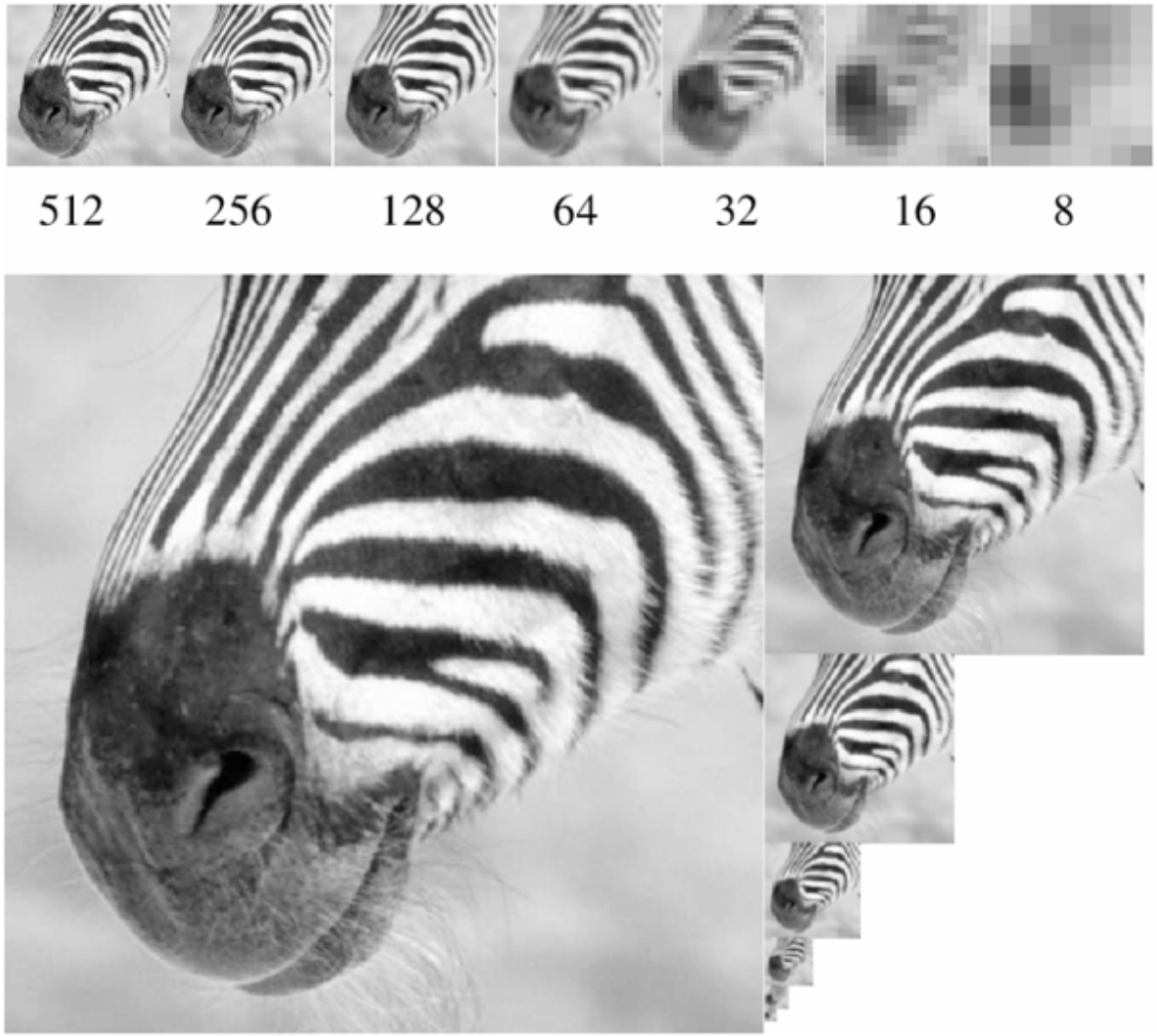
The Nyquist-Shannon sampling theorem

A continuous signal can be perfectly reconstructed from its discrete version using linear interpolation, if sampling occurred with frequency:

$$f_s \geq 2 f_{\max}$$ ← This is called the Nyquist frequency

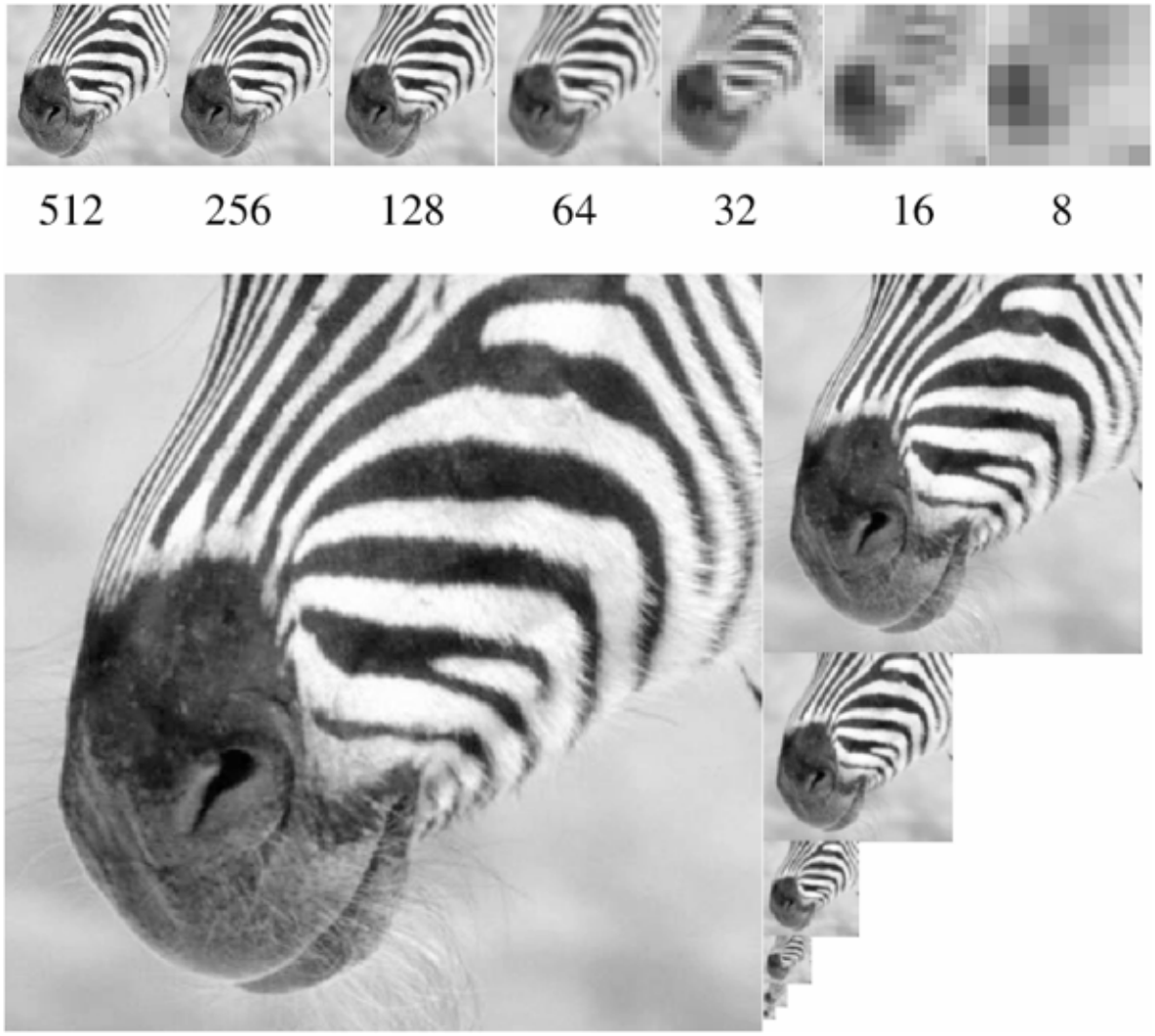
Equivalent reformulation: When downsampling, aliasing does not occur if samples are taken at the Nyquist frequency or higher.

Gaussian pyramid



How does the Nyquist-Shannon theorem relate to the Gaussian pyramid?

Gaussian pyramid



How does the Nyquist-Shannon theorem relate to the Gaussian pyramid?

- Gaussian blurring is low-pass filtering.
- By blurring we try to sufficiently decrease the Nyquist frequency to avoid aliasing.

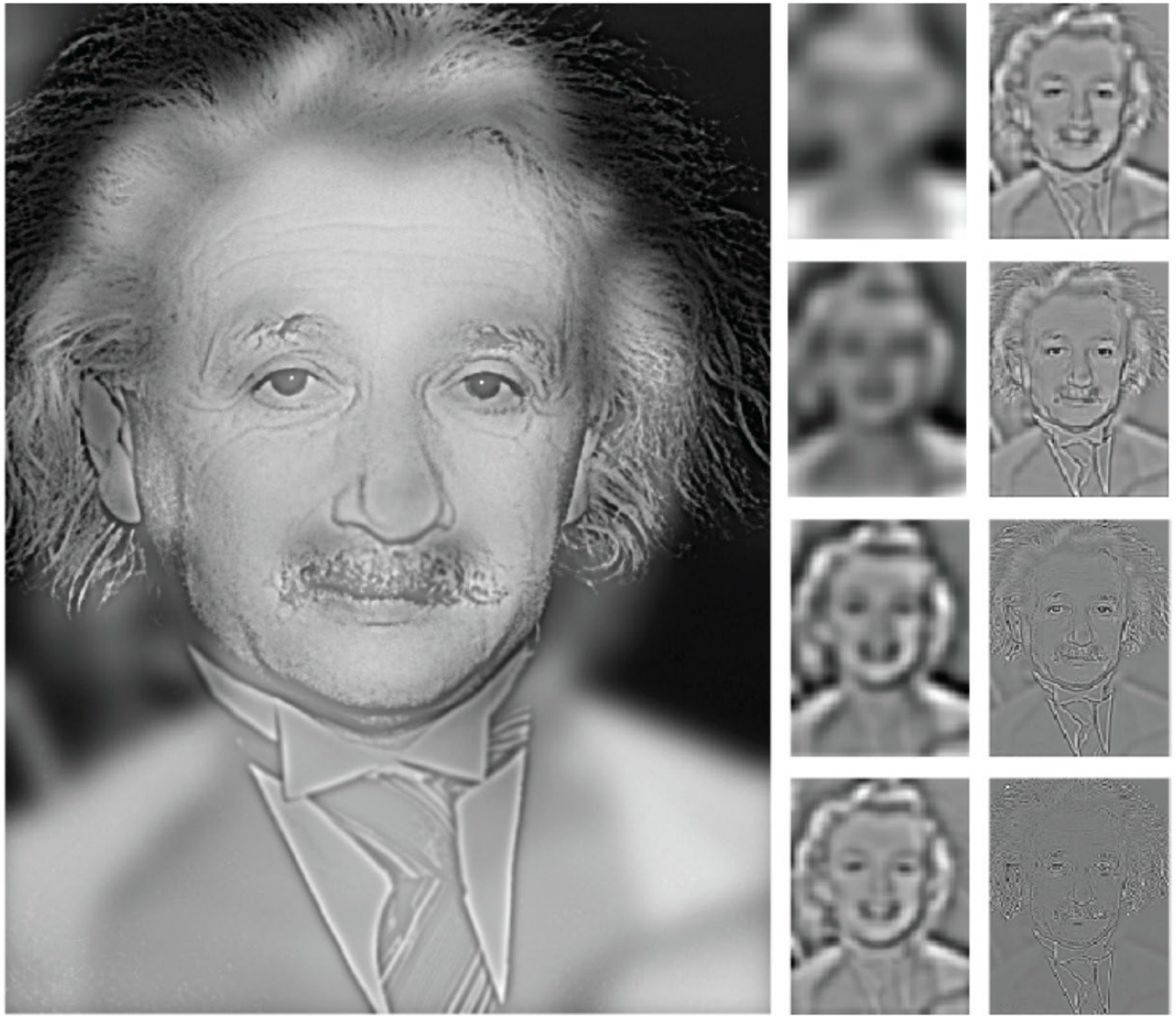
How large should the Gauss blur we use be?

Frequency-domain filtering in human vision

“Hybrid image”

Aude Oliva and Philippe Schyns

Frequency-domain filtering in human vision



http://cvcl.mit.edu/hybrid_gallery/gallery.html