6.891

Computer Vision and Applications

Prof. Trevor. Darrell

Lecture 2:

- Linear Filters and Convolution (review)
- Fourier Transform (review)
- Sampling and Aliasing (review)

Readings: F&P Chapter 7.1-7.6

Recap: Cameras, lenses, and calibration

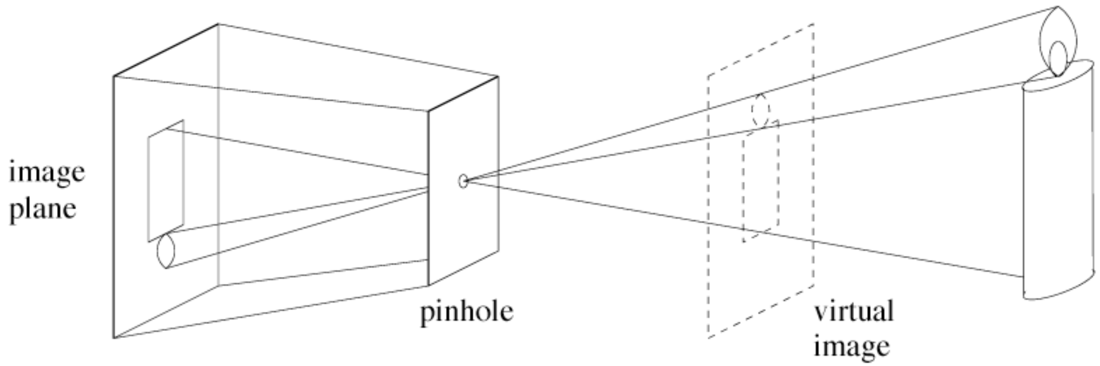
Last time:

- Camera models
- Projection equations
- Calibration methods

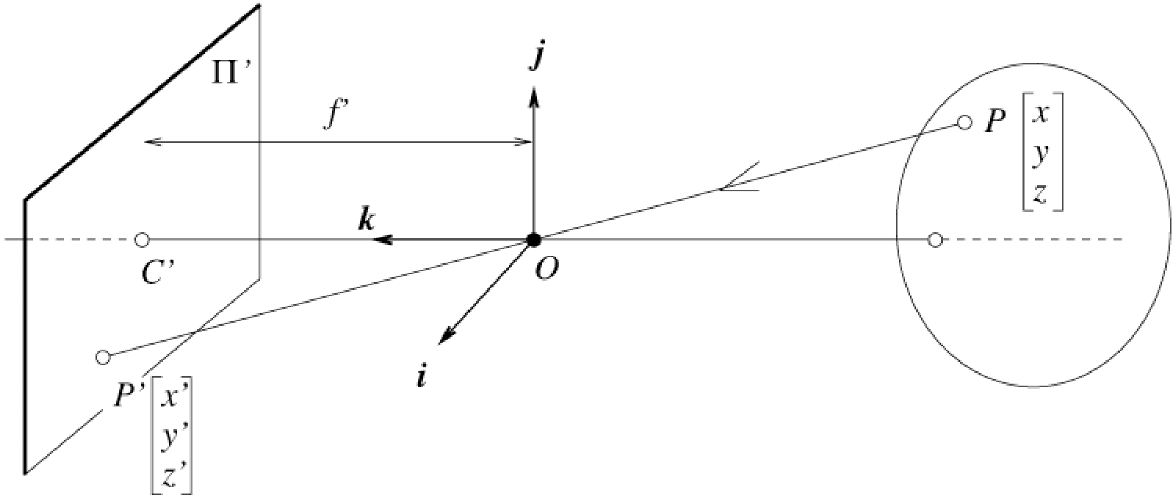
Images are projections of the 3-D world onto a 2-D plane…

Recap: pinhole/perspective

- Pinole camera model - box with a small hole in it:



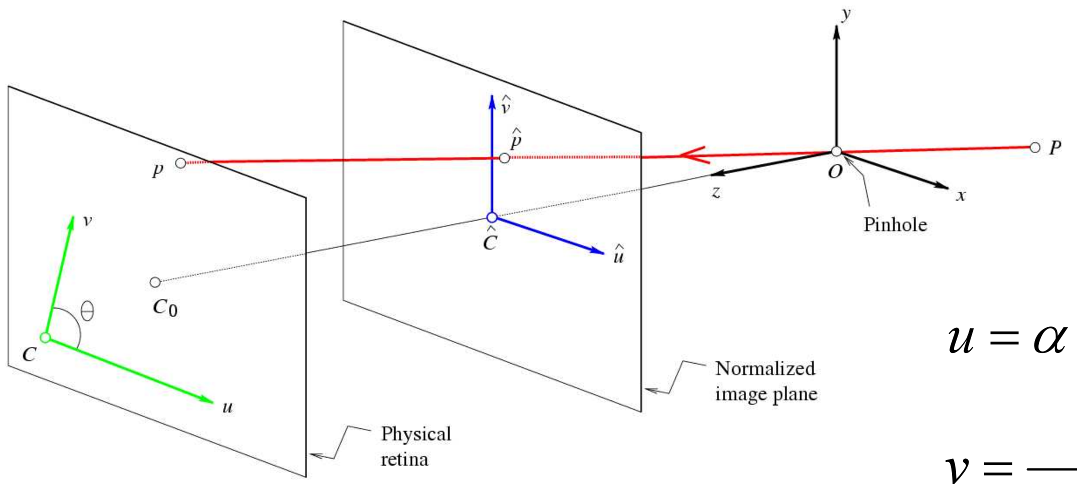
- Perspective projection:



$$\begin{cases} x' = f'\dfrac{x}{z} \\ y' = f'\dfrac{y}{z} \end{cases}$$

Forsyth&Ponce

Recap: Intrinsic parameters



$$u = \alpha \frac{x}{z} - \alpha \cot(\theta) \frac{y}{z} + u_0$$

$$v = \frac{\beta}{\sin(\theta)} \frac{y}{z} + v_0$$

Using homogenous coordinates,
we can write this as:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z} \begin{pmatrix} \alpha & -\alpha\cot(\theta) & u_0 & 0 \\ 0 & \dfrac{\beta}{\sin(\theta)} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

or:

$$\boxed{\vec{p} = \frac{1}{z} \begin{pmatrix} K & \vec{0} \end{pmatrix} \vec{P}_4}$$

Recap: Combining extrinsic and intrinsic calibration parameters

$$\vec{p} = \frac{1}{z} \begin{pmatrix} K & \vec{0} \end{pmatrix} \vec{P}$$

Intrinsic

$$^{C}P = {}^{C}_{W}R \; {}^{W}P + {}^{C}O_{W}$$

Extrinsic

$$\vec{p} = \frac{1}{z} \; K \begin{pmatrix} {}^{C}_{W}R & {}^{C}O_{W} \end{pmatrix} \vec{P}$$

$$\vec{p} = \frac{1}{z} \; M \; \vec{P}$$

Forsyth&Ponce

Other ways to write the same equation

pixel coordinates

world coordinates

$$\vec{p} = \frac{1}{z} M\vec{P}$$

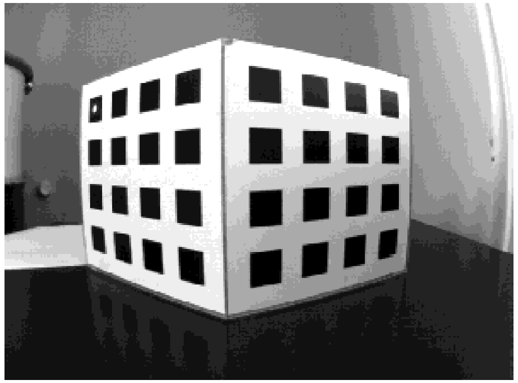
$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z} \begin{pmatrix} . & m_1^T & . & . \\ . & m_2^T & . & . \\ . & m_3^T & . & . \end{pmatrix} \begin{pmatrix} W_x \\ W_y \\ W_z \\ 1 \end{pmatrix}$$

$$\begin{cases} u = \dfrac{m_1 \cdot \vec{P}}{m_3 \cdot \vec{P}} \\[2ex] v = \dfrac{m_2 \cdot \vec{P}}{m_3 \cdot \vec{P}} \end{cases}$$

z is in the *camera* coordinate system, but we can solve for that, since $1 = \dfrac{m_3 \cdot \vec{P}}{z}$, leading to:

$$u = \frac{m_1 \cdot \vec{P}}{m_3 \cdot \vec{P}} \qquad v = \frac{m_2 \cdot \vec{P}}{m_3 \cdot \vec{P}}$$

Recap: Camera calibration



The Opti-CAL Calibration Target Image

Stack all these measurements of i=1…n points

$$(m_1 - u_i m_3) \cdot \vec{P}_i = 0$$
$$(m_2 - v_i m_3) \cdot \vec{P}_i = 0$$

into a big matrix:

$$\begin{pmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ \cdots & \cdots & \cdots \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

Today

Review of early visual processing

- Linear Filters and Convolution
- Fourier Transform
- Sampling and Aliasing

You should have been exposed to this material in previous courses; this lecture is just a (quick) review.

Administrivia:

- sign-up sheet
- introductions

What is image filtering?

- Modify the pixels in an image based on some function of a local neighborhood of the pixels.

10	5	3
4	5	1
1	1	7

Local image data

Some function →

	7	

Modified image data

Linear functions

- Simplest: linear filtering.
  - Replace each pixel by a linear combination of its neighbors.
- The prescription for the linear combination is called the "convolution kernel".

10	5	3
4	5	1
1	1	7

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

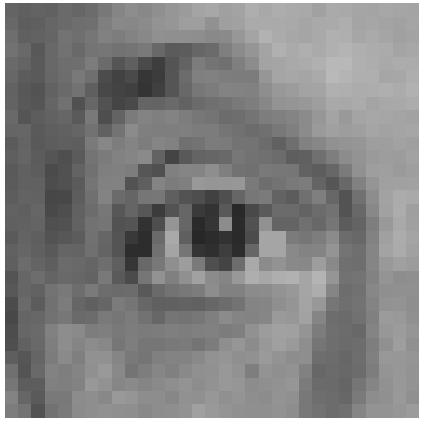
	7	

Modified image data

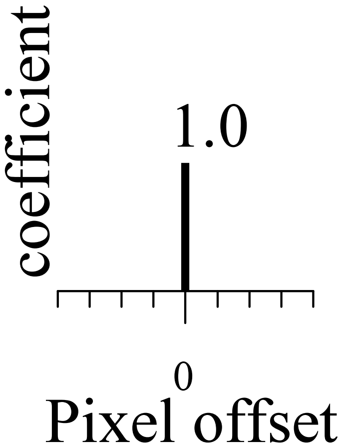
Convolution

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

Linear filtering (warm-up slide)

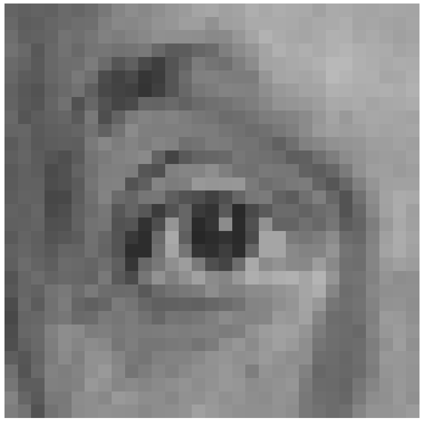


original

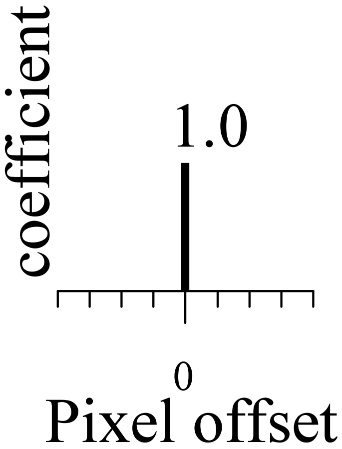


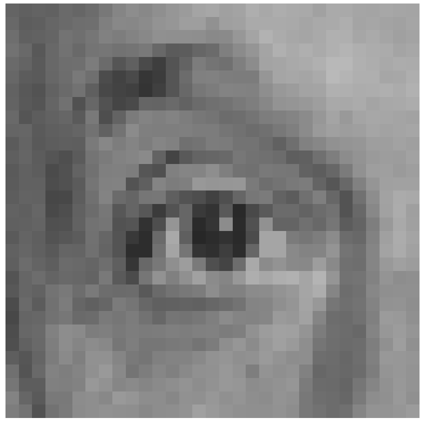
?

Linear filtering (warm-up slide)



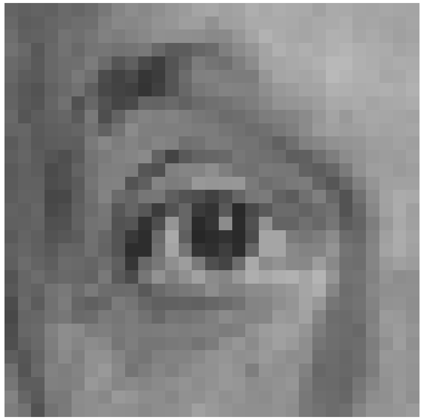
original



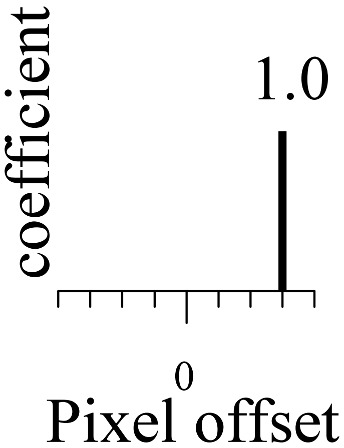


Filtered
(no change)

Linear filtering

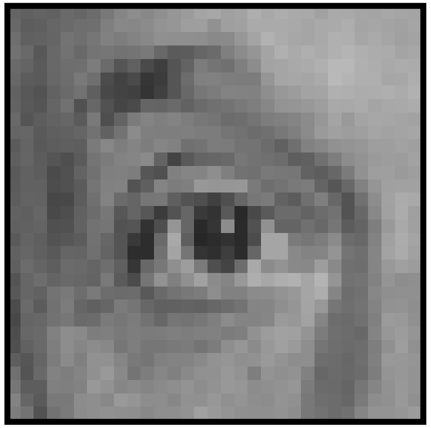


original

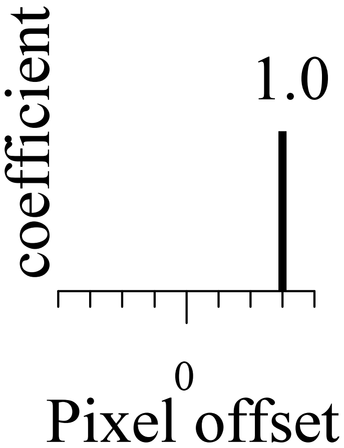


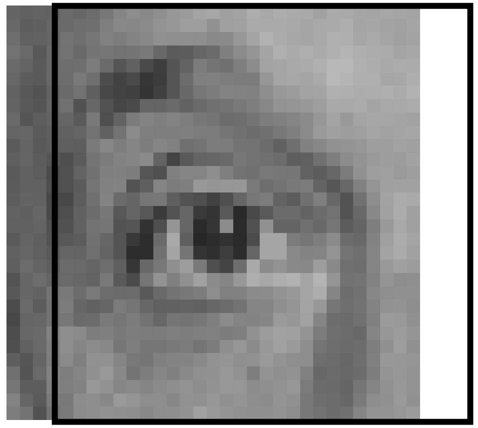
?

shift



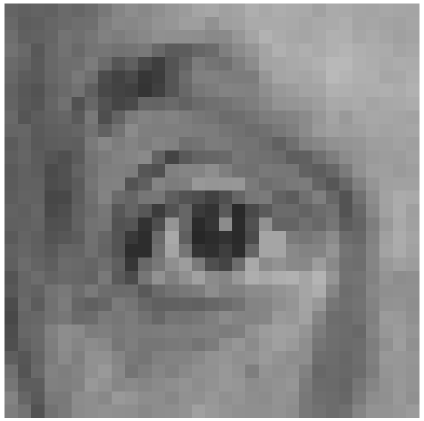
original



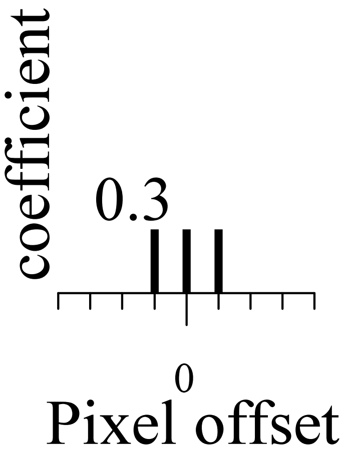


shifted

Linear filtering

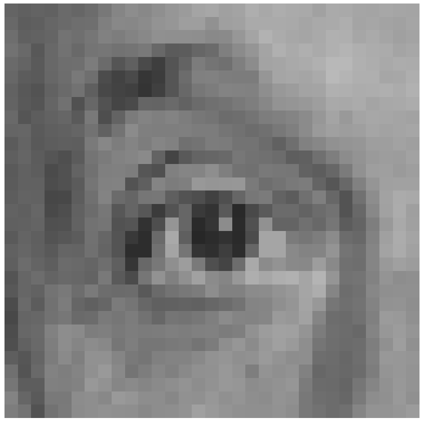


original



?

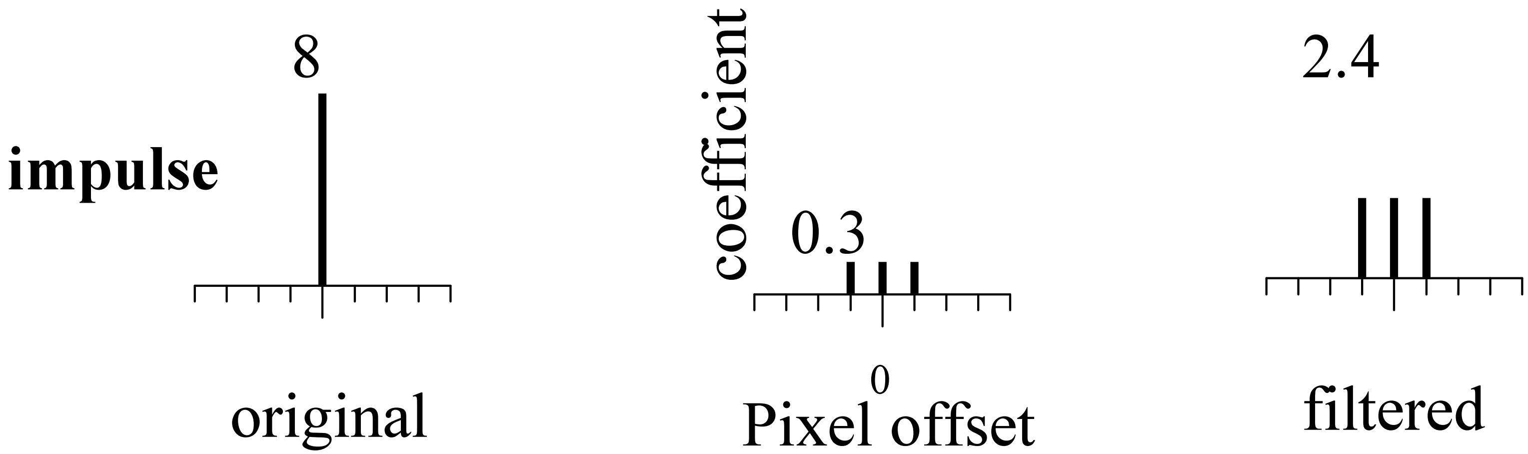
Blurring



original

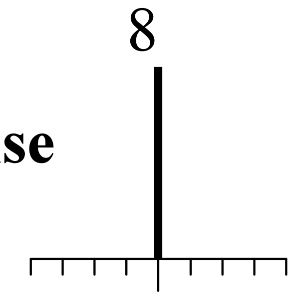
Blurred (filter applied in both dimensions).

Blur examples

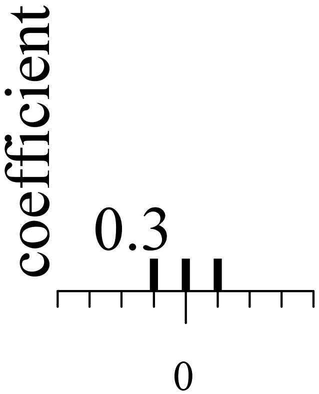


Blur examples

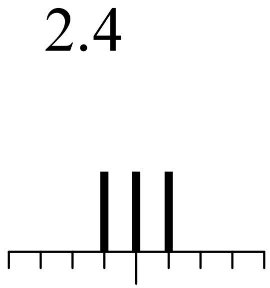
impulse



original

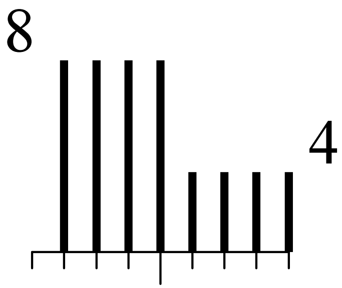


Pixel offset

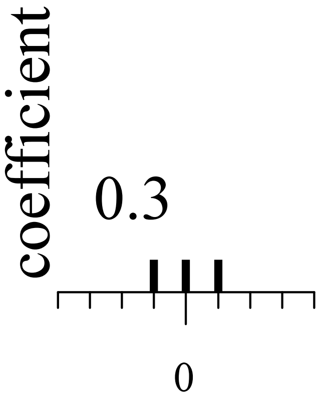


filtered

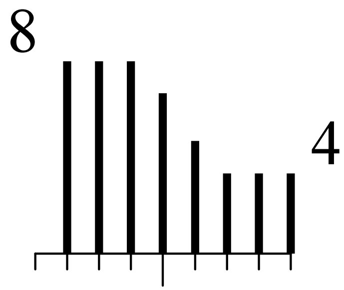
edge



original



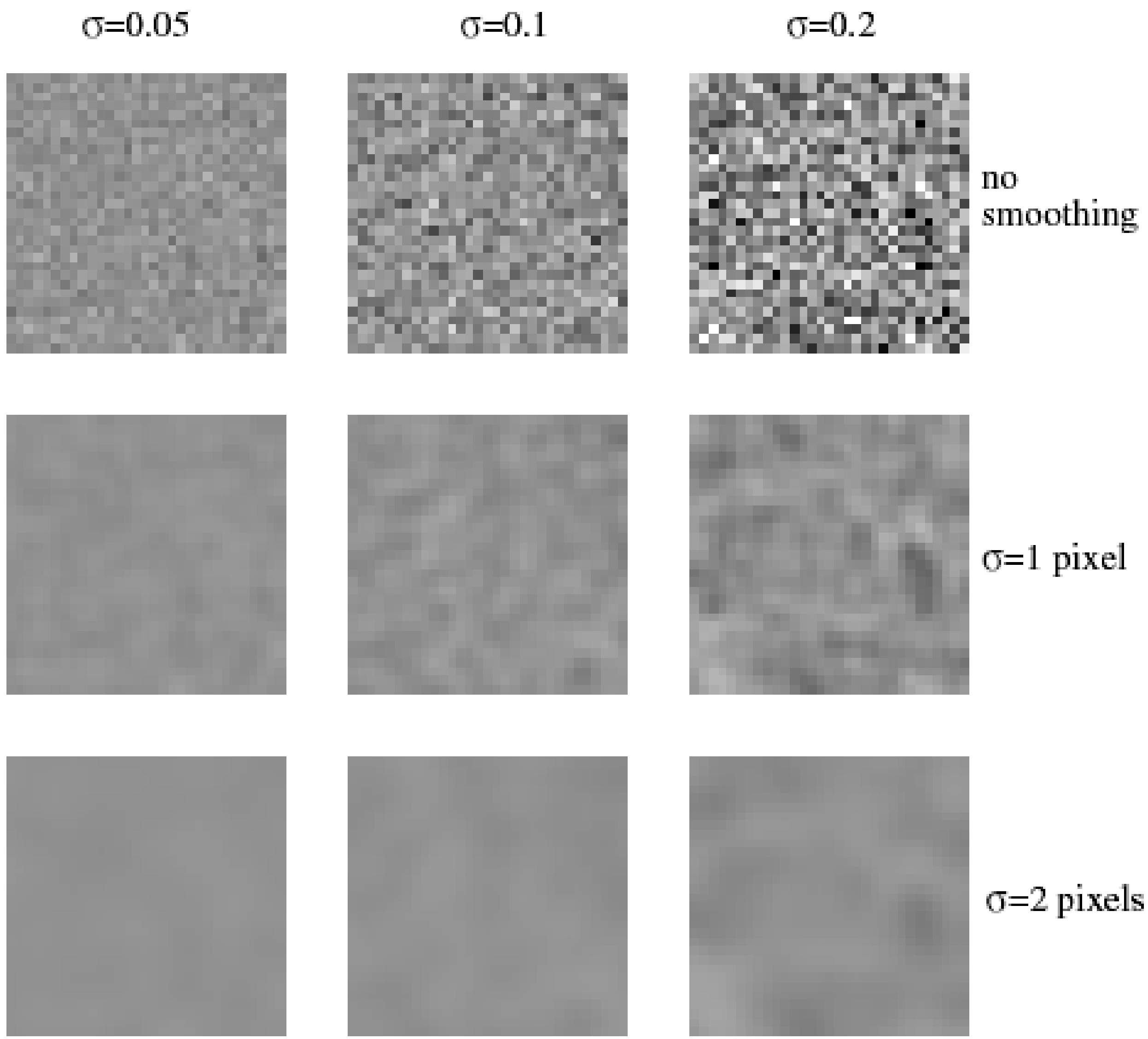
Pixel offset



filtered

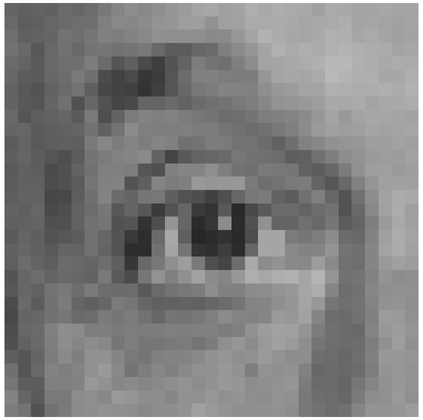
Smoothing reduces noise

- Generally expect pixels to "be like" their neighbours
  - surfaces turn slowly
  - relatively few reflectance changes
- Generally expect noise processes to be independent from pixel to pixel
- Implies that smoothing suppresses noise, for appropriate noise models
- Scale
  - the parameter in the symmetric Gaussian
  - as this parameter goes up, more pixels are involved in the average
  - and the image gets more blurred
  - and noise is more effectively suppressed

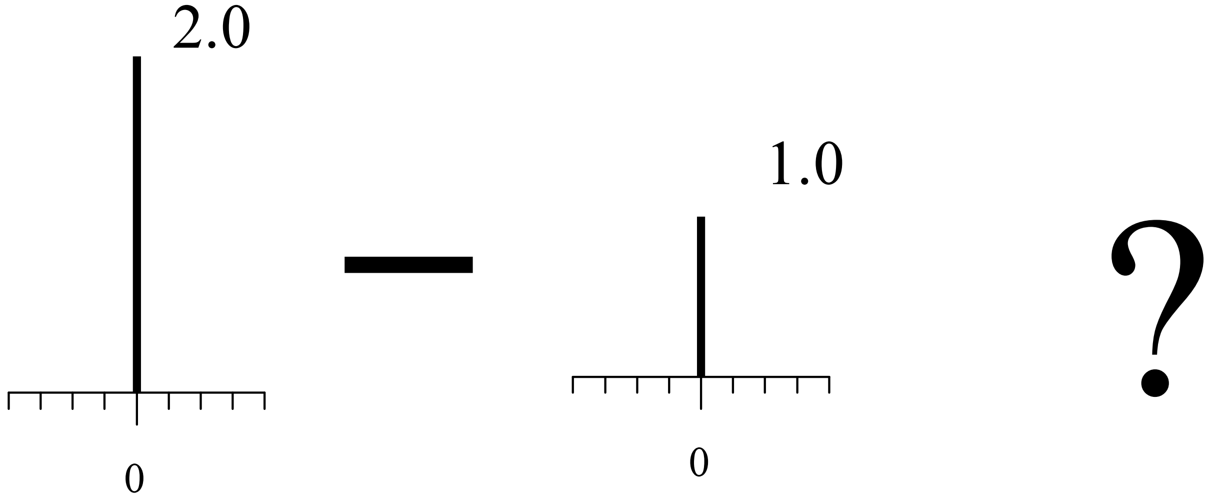


The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realisations of an image of gaussian noise.

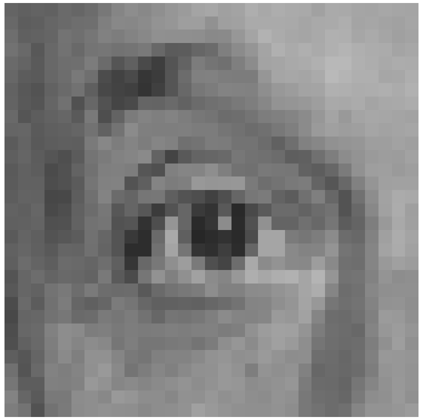
Linear filtering (warm-up slide)



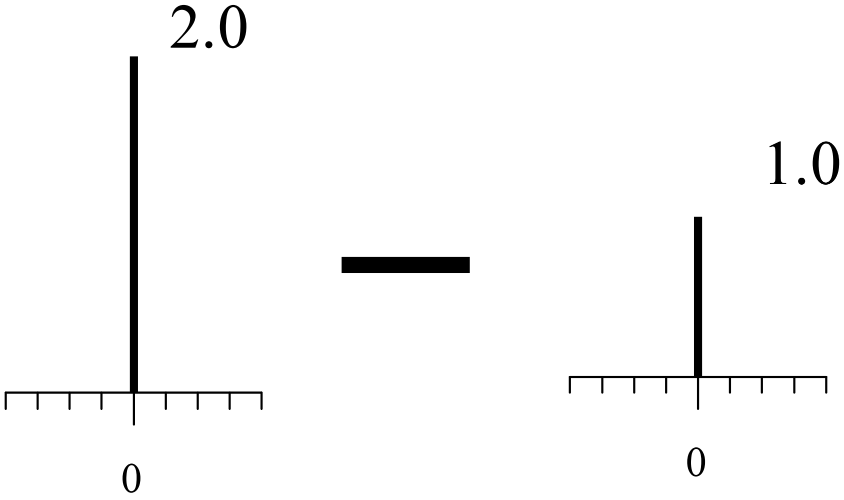
original

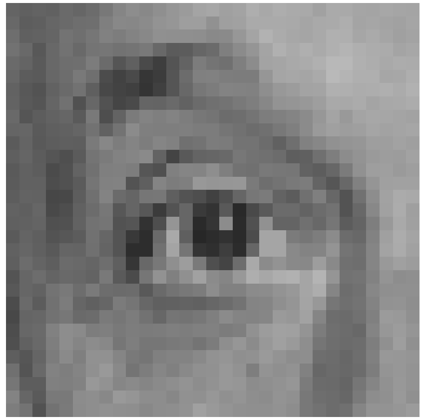


Linear filtering (no change)



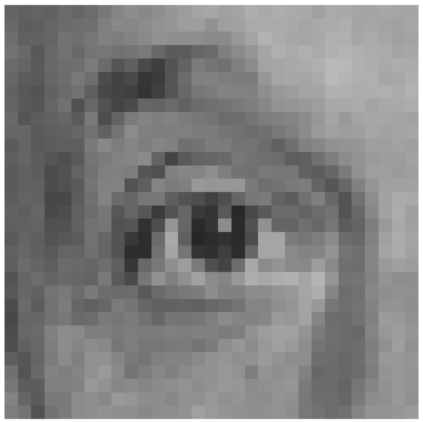
original



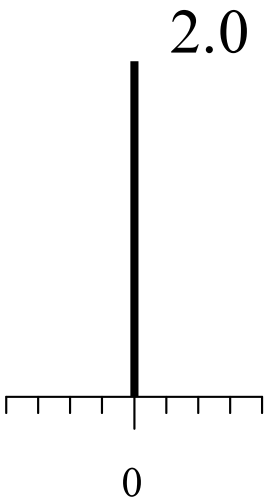


Filtered
(no change)

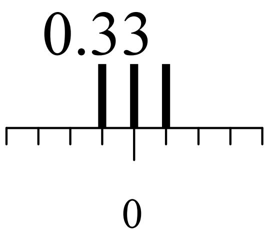
Linear filtering



original

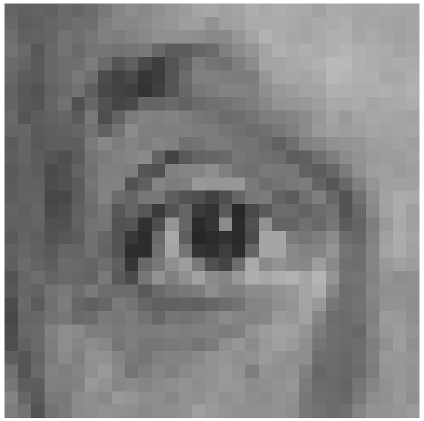


−



?

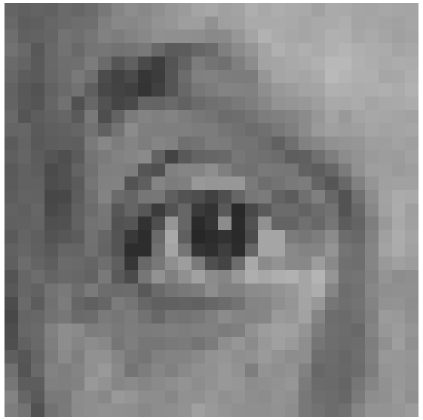
(remember blurring)



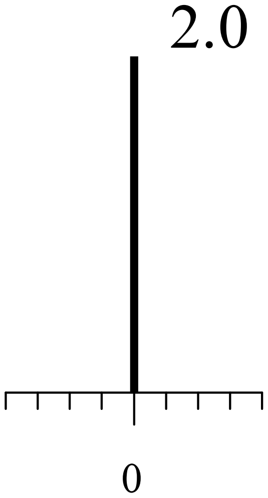
original

Blurred (filter applied in both dimensions).

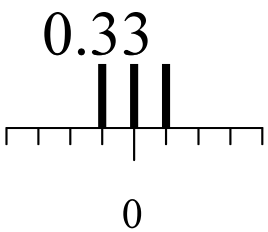
Linear filtering



original

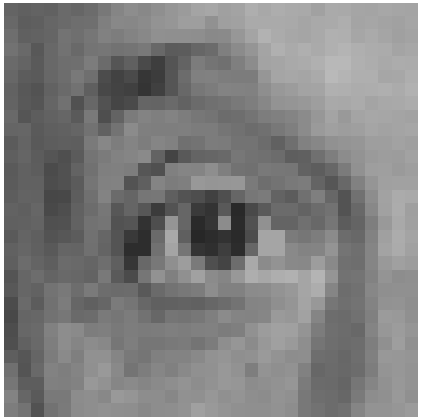


−

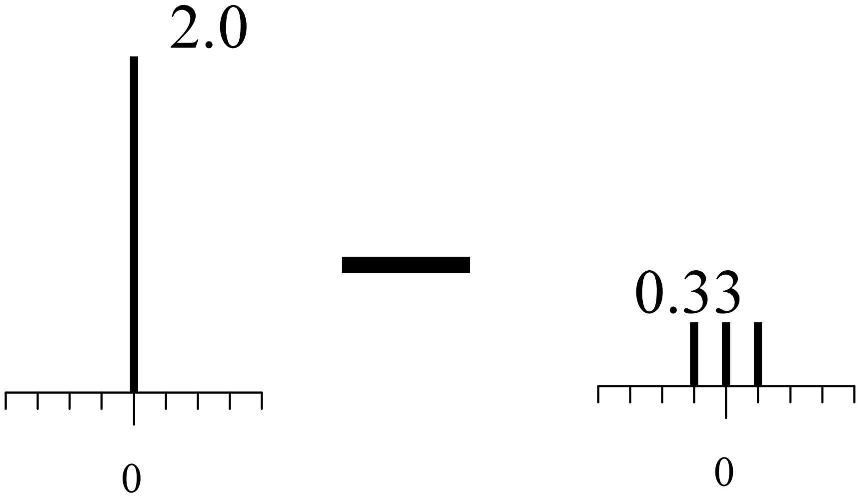


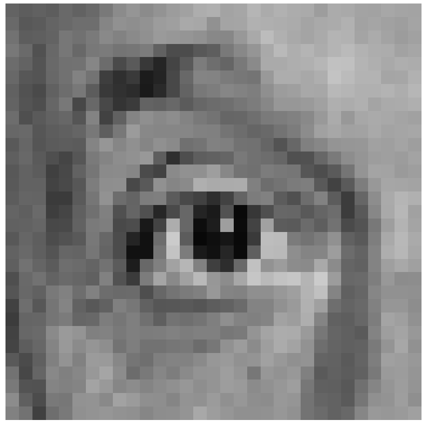
?

Sharpening



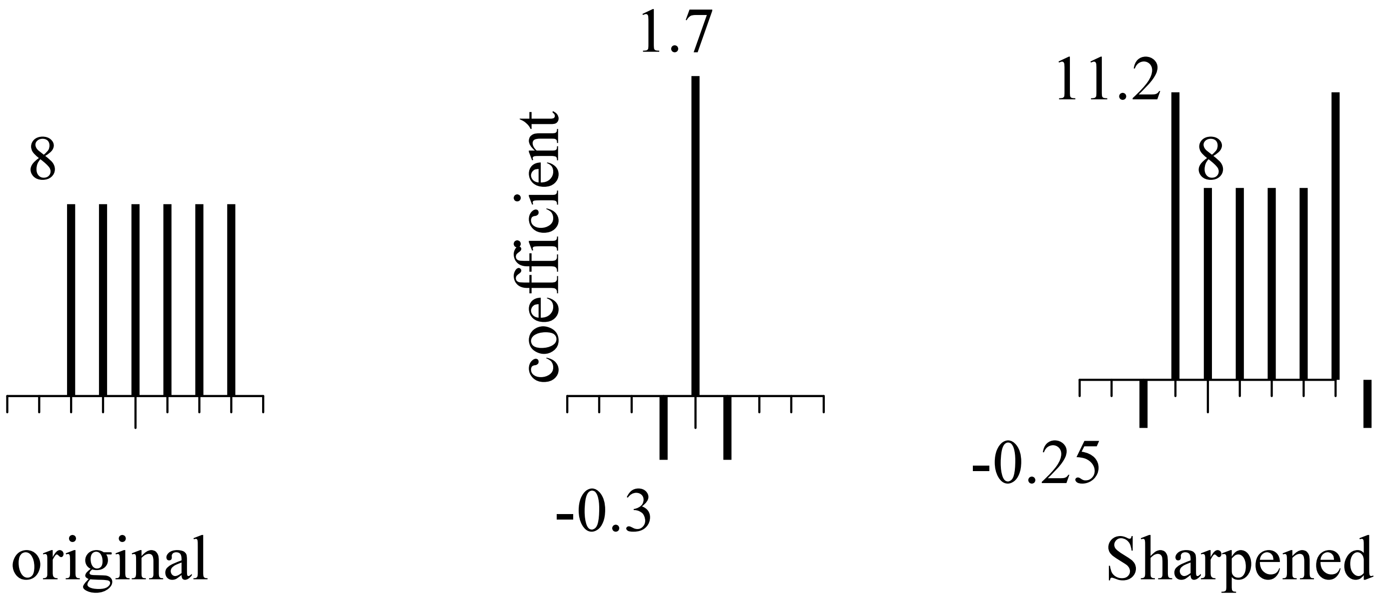
original





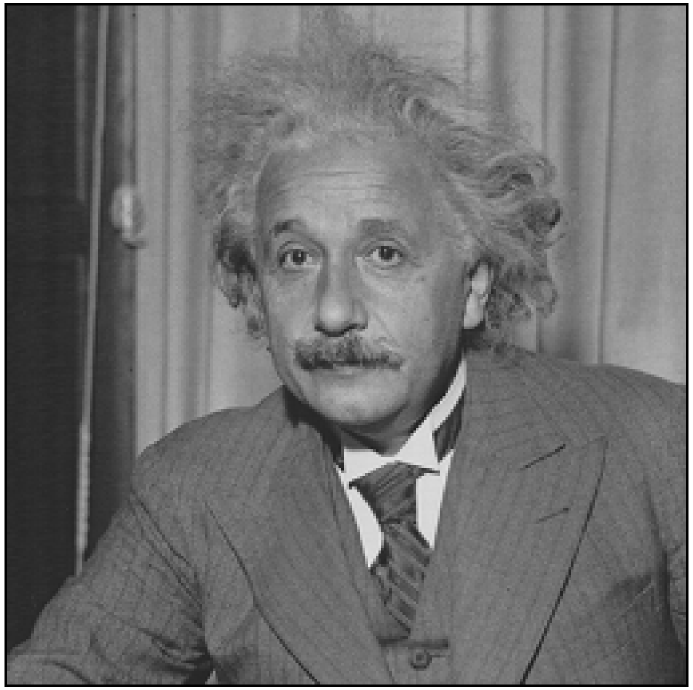
Sharpened
original

Sharpening example

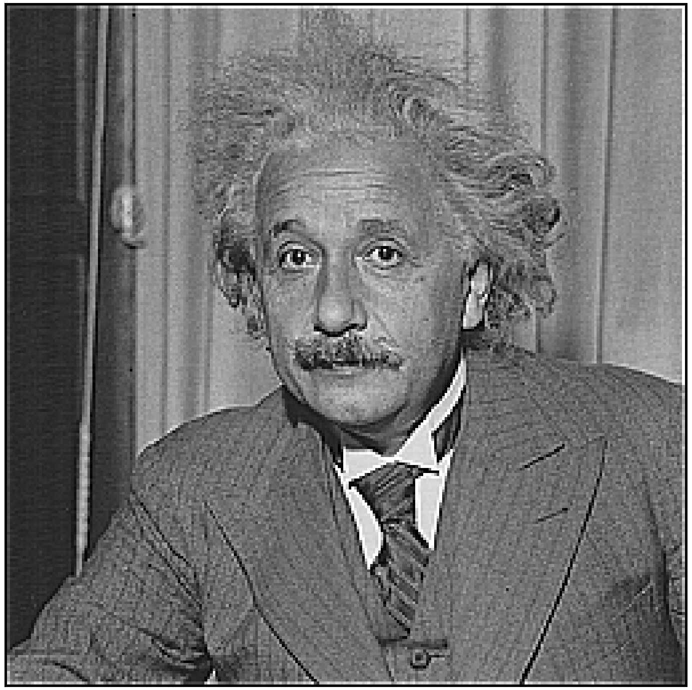


Sharpened
(differences are
accentuated; constant
areas are left untouched).

Sharpening



before



after

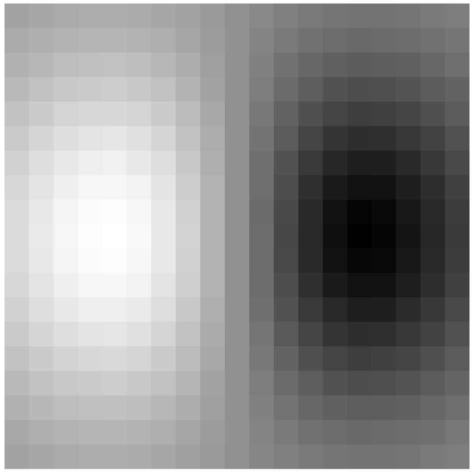
Gradients and edges

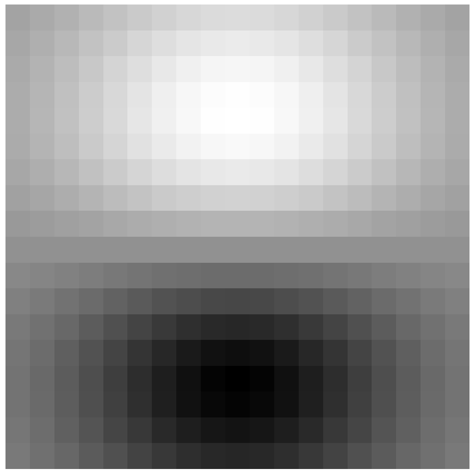
- Points of sharp change in an image are interesting:
  - change in reflectance
  - change in object
  - change in illumination
  - noise
- Sometimes called **edge points**

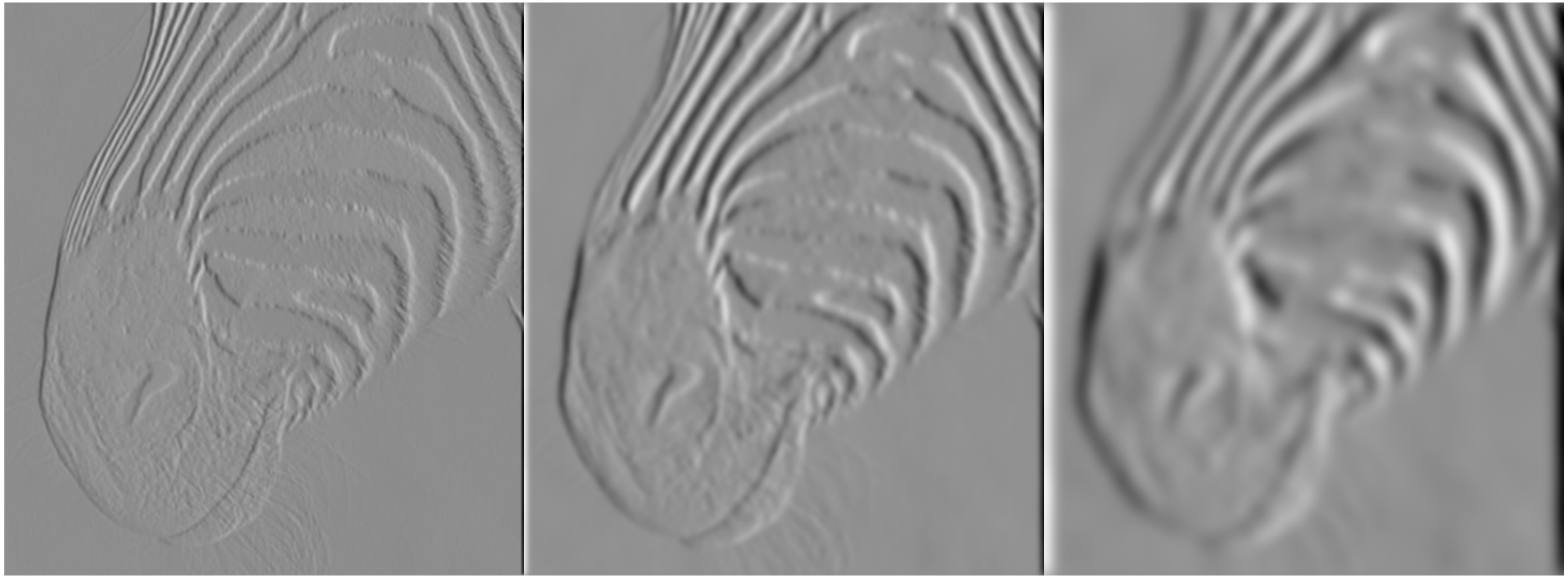
- General strategy
  - linear filters to estimate image gradient
  - mark points where gradient magnitude is particularly large wrt neighbours (ideally, curves of such points).

Smoothing and Differentiation

- Issue: noise
  - smooth before differentiation
  - two convolutions to smooth, then differentiate?
  - actually, no - we can use a derivative of Gaussian filter
    - because differentiation is convolution, and convolution is associative



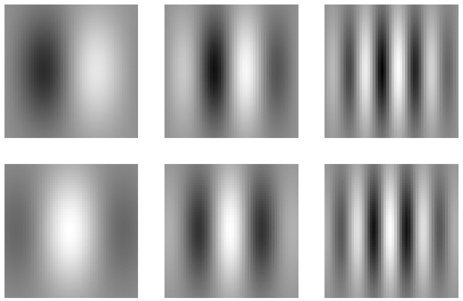




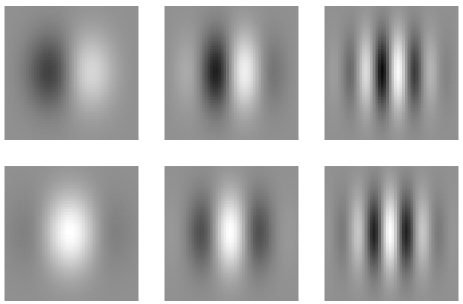
1 pixel 3 pixels 7 pixels

The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered.

Oriented filters



Gabor filters (Gaussian modulated harmonics) at different scales and spatial frequencies



Top row shows anti-symmetric (or odd) filters, bottom row the symmetric (or even) filters.

Linear image transformations

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

transformed image

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

Vectorized image

Fourier transform, or
Wavelet transform, or
Steerable pyramid transform

Self-inverting transforms

Same basis functions are used for the inverse transform

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

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

U transpose and complex conjugate

An example of such a transform: the Fourier transform

discrete domain

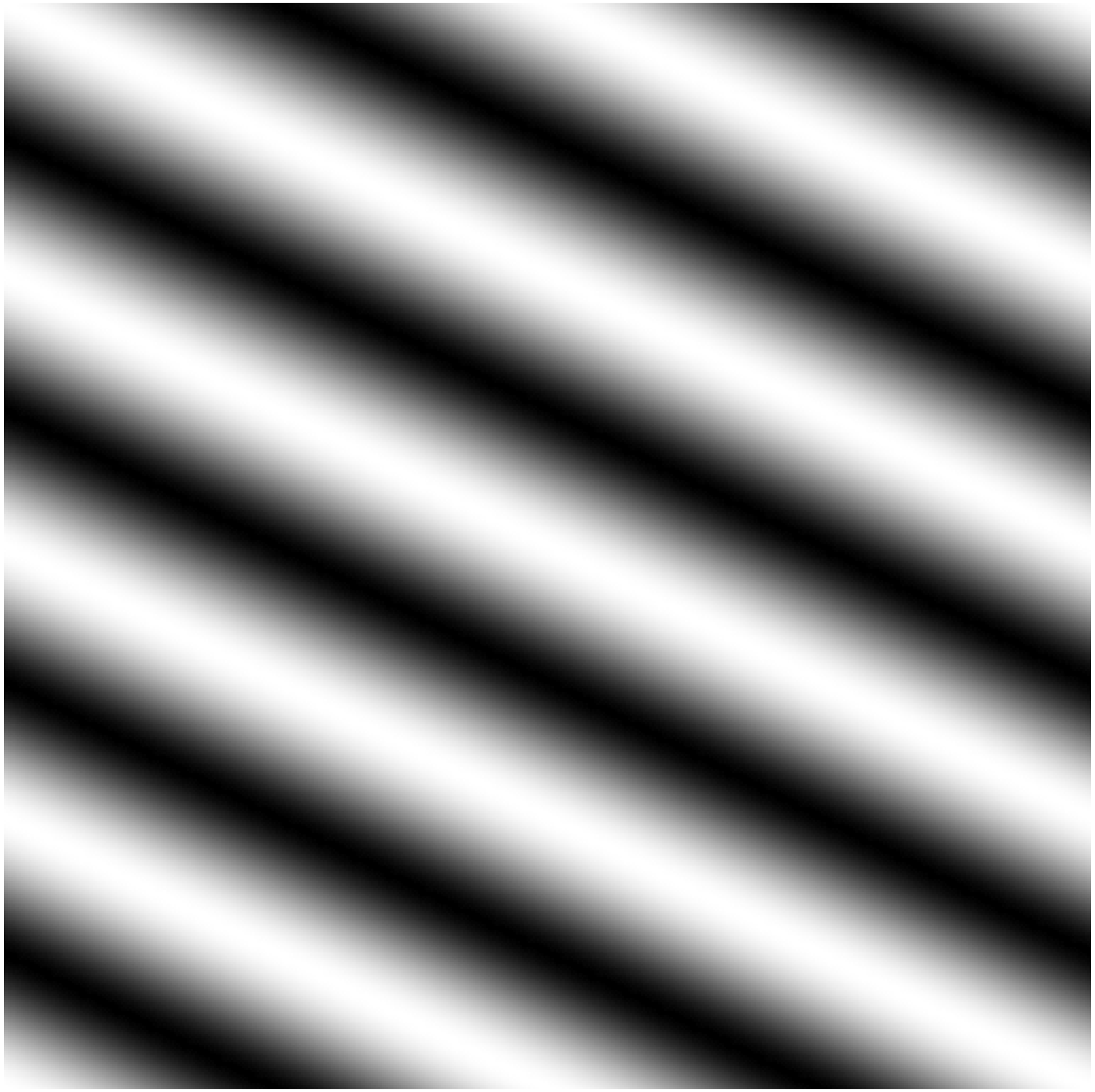
Forward transform

$$F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k,l] e^{-\pi i \left(\frac{km}{M} + \frac{\ln}{N} \right)}$$

Inverse transform

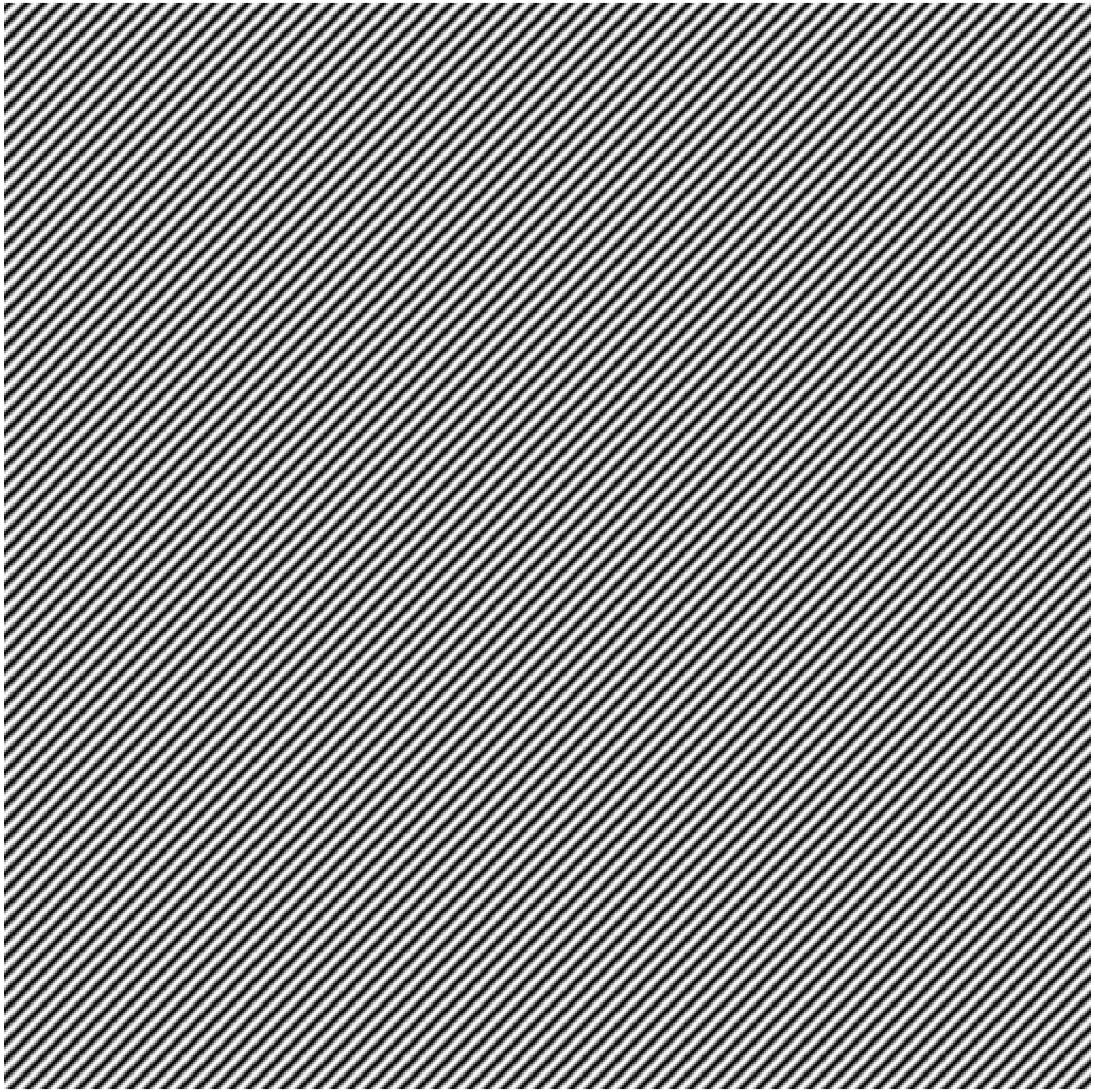
$$f[k,l] = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} F[m,n] e^{+\pi i \left(\frac{km}{M} + \frac{\ln}{N} \right)}$$

To get some sense of what basis elements look like, we plot a basis element --- or rather, its real part --- as a function of x,y for some fixed u, v. We get a function that is constant when (ux+vy) is constant. The magnitude of the vector (u, v) gives a frequency, and its direction gives an orientation. The function is a sinusoid with this frequency along the direction, and constant perpendicular to the direction.



Here u and v are larger than in the previous slide.

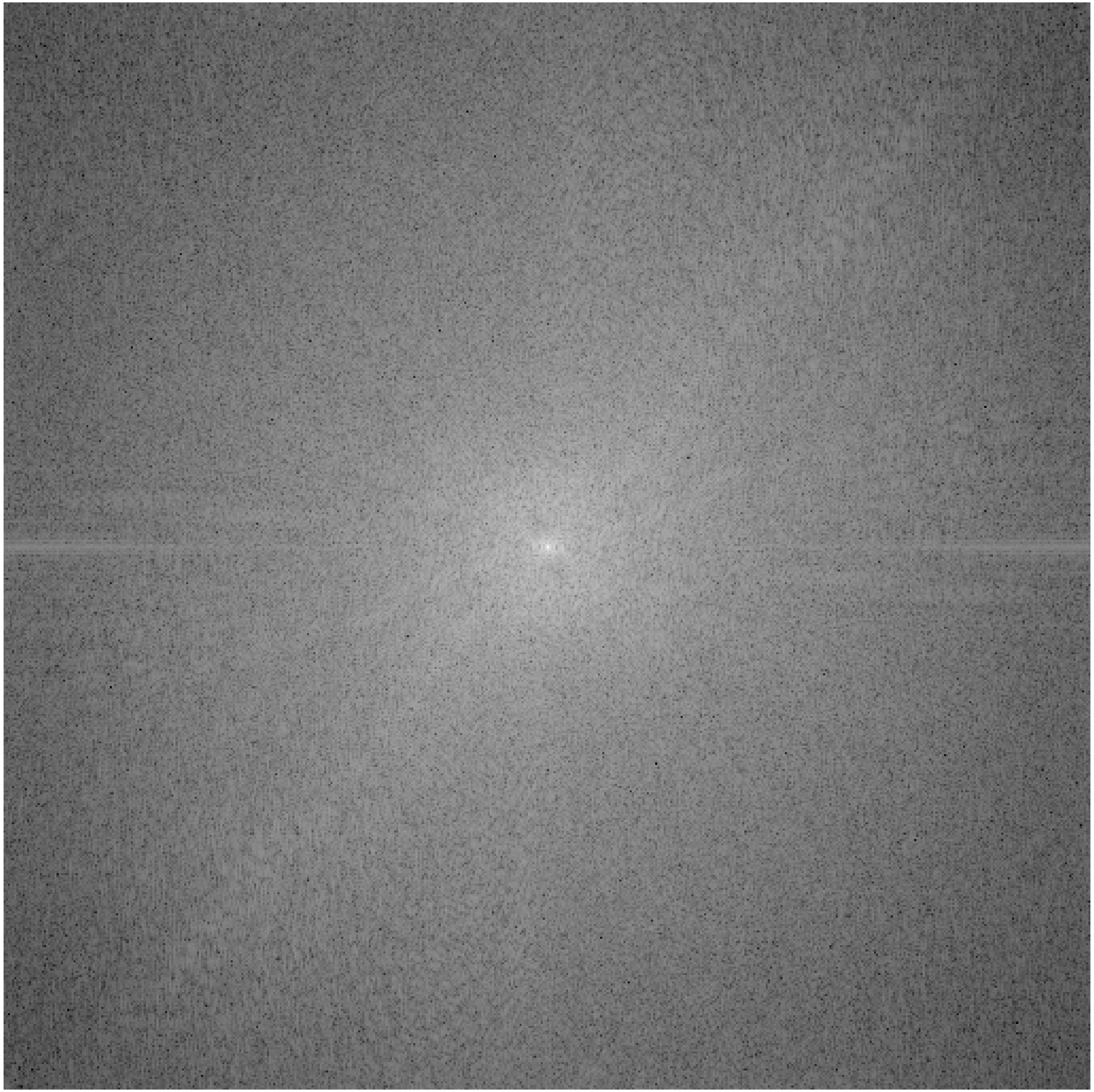
And larger still...



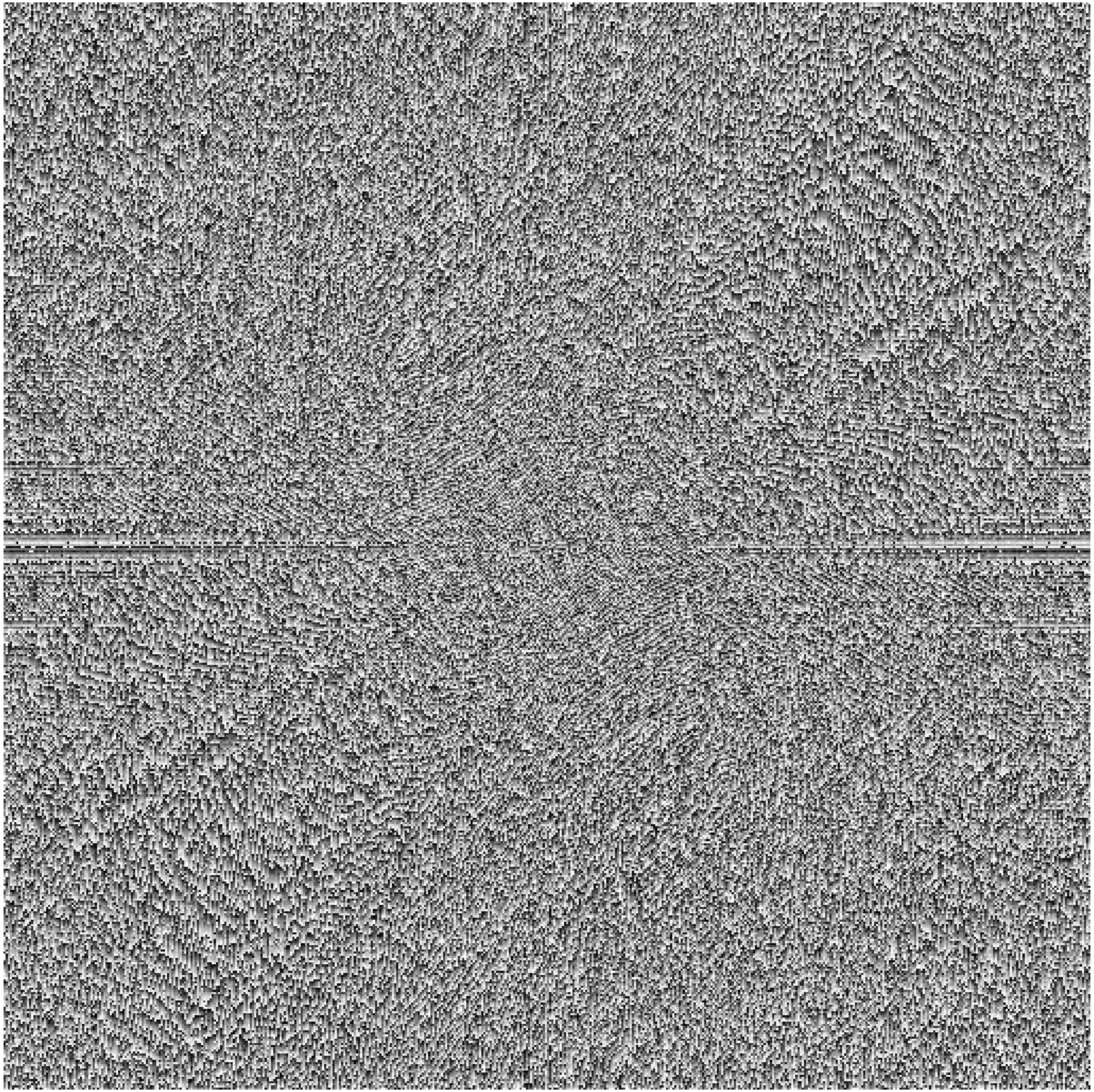
Phase and Magnitude

- Fourier transform of a real function is complex
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
- Magnitude is the magnitude of the complex transform
- Curious fact
  - all natural images have about the same magnitude transform
  - hence, phase seems to matter, but magnitude largely doesn't
- Demonstration
  - Take two pictures, swap the phase transforms, compute the inverse - what does the result look like?

This is the magnitude transform of the cheetah pic

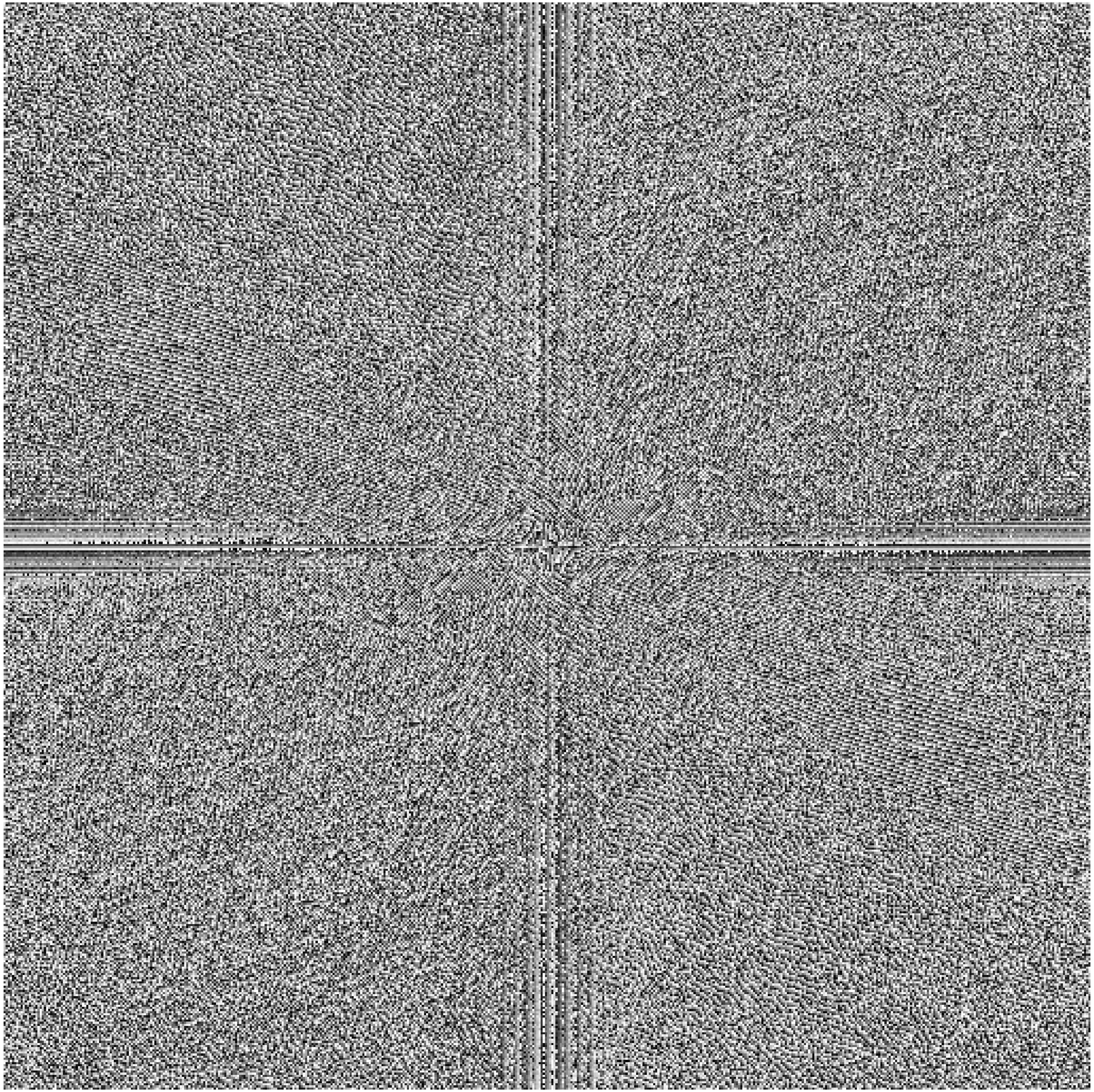


This is the phase transform of the cheetah pic

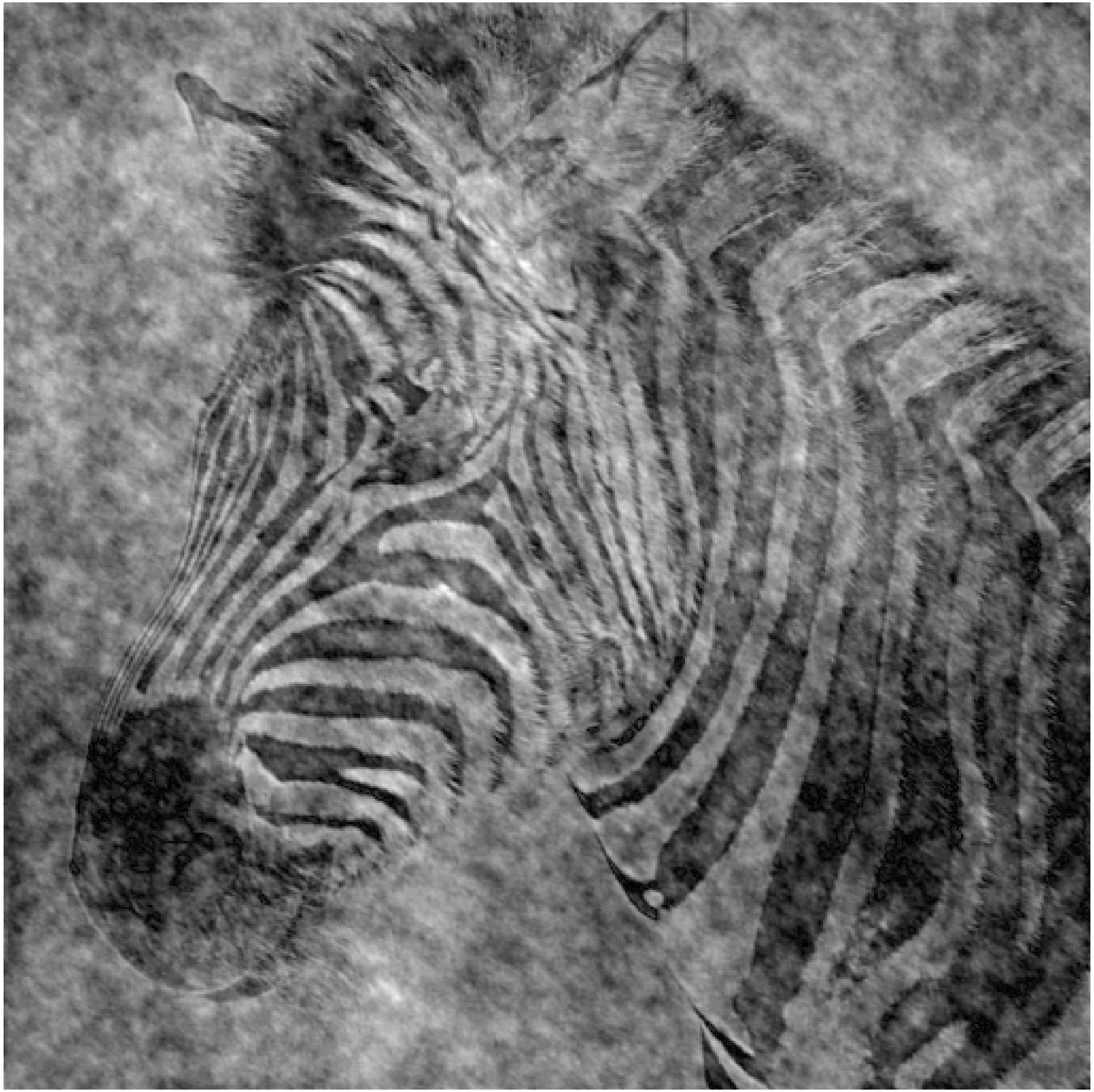


This is the magnitude transform of the zebra pic

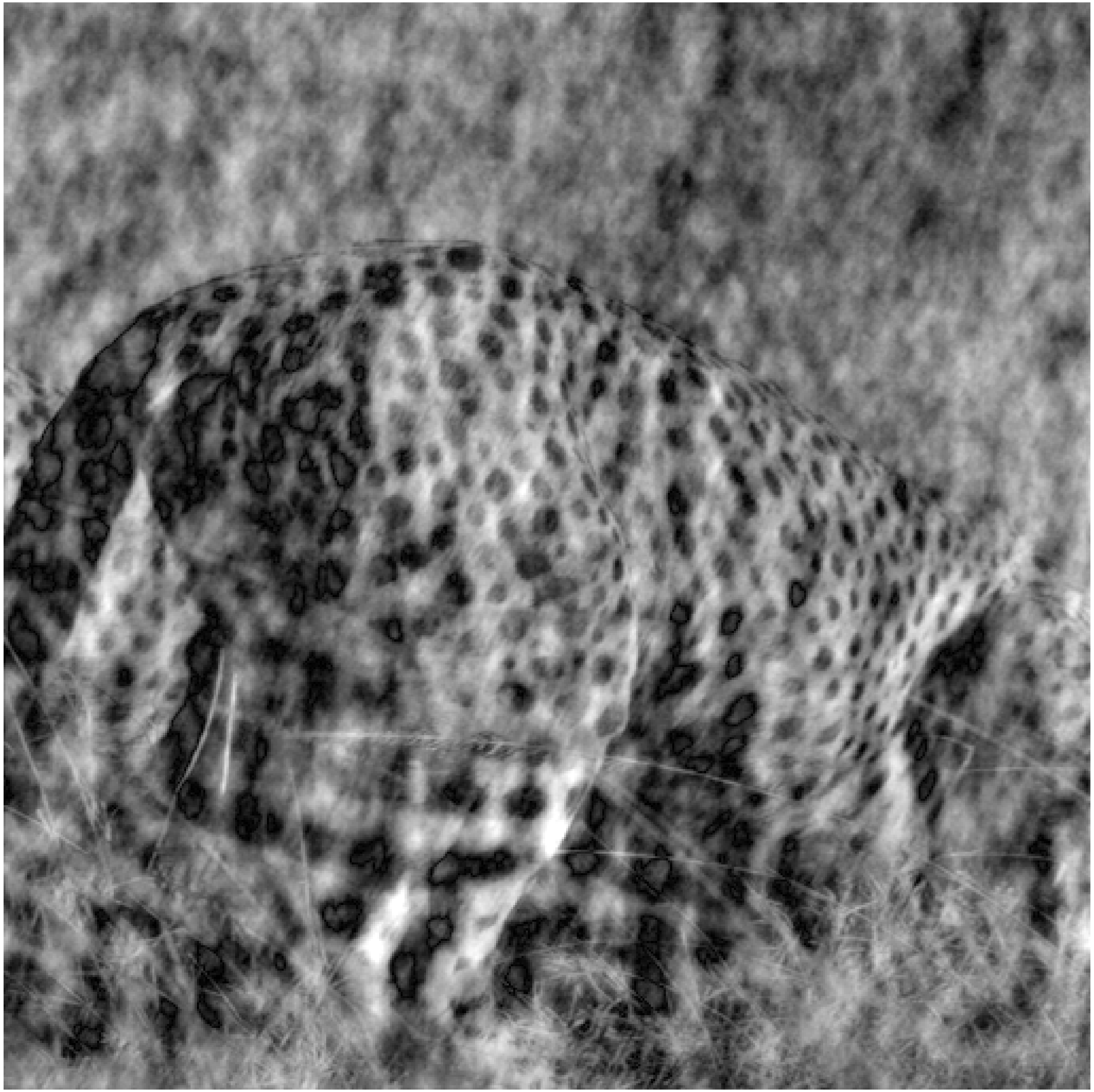
This is the phase transform of the zebra pic



Reconstruction with zebra phase, cheetah magnitude



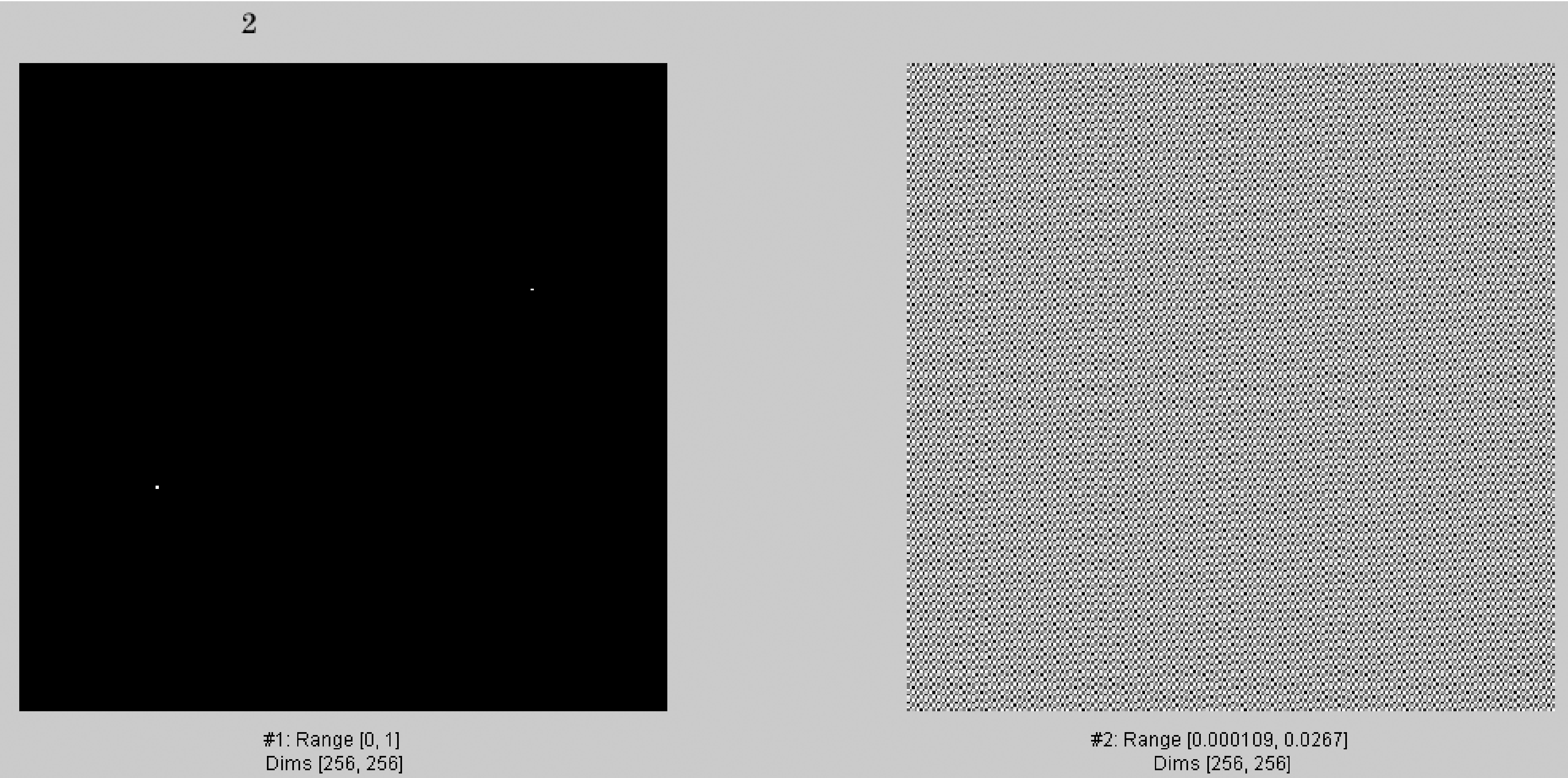
Reconstruction with cheetah phase, zebra magnitude



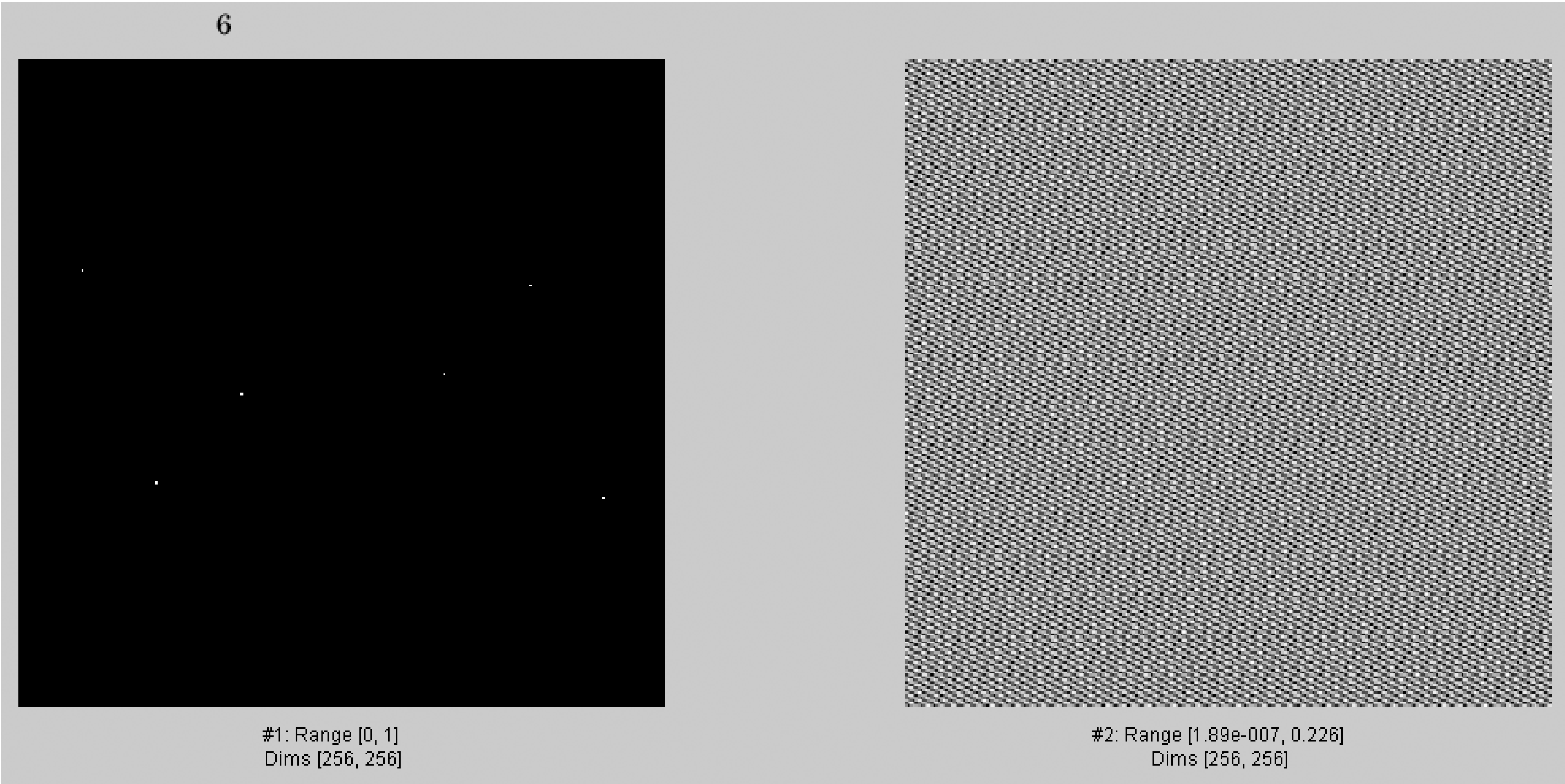
Example image synthesis with fourier basis.

- 16 images

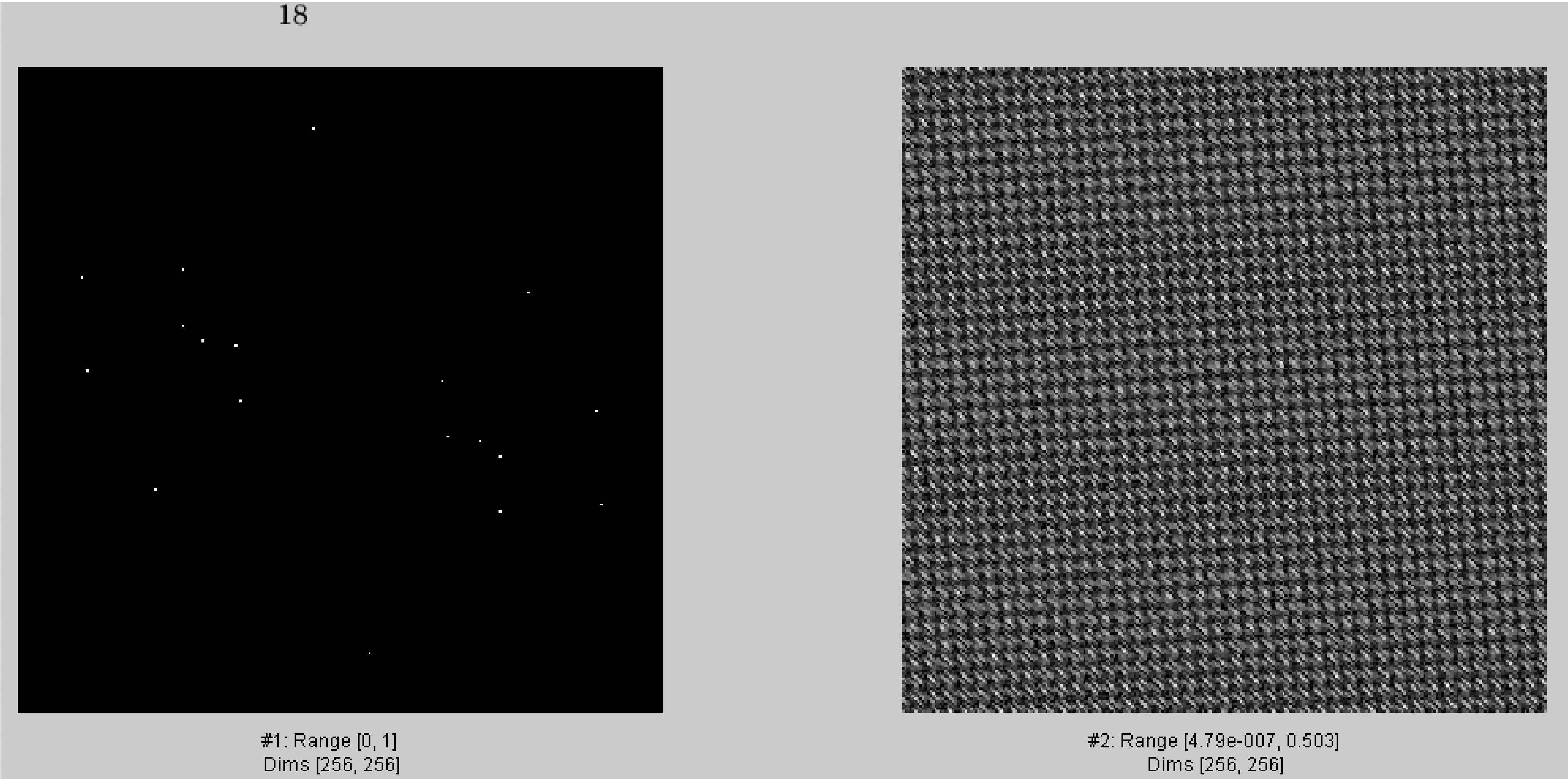
2



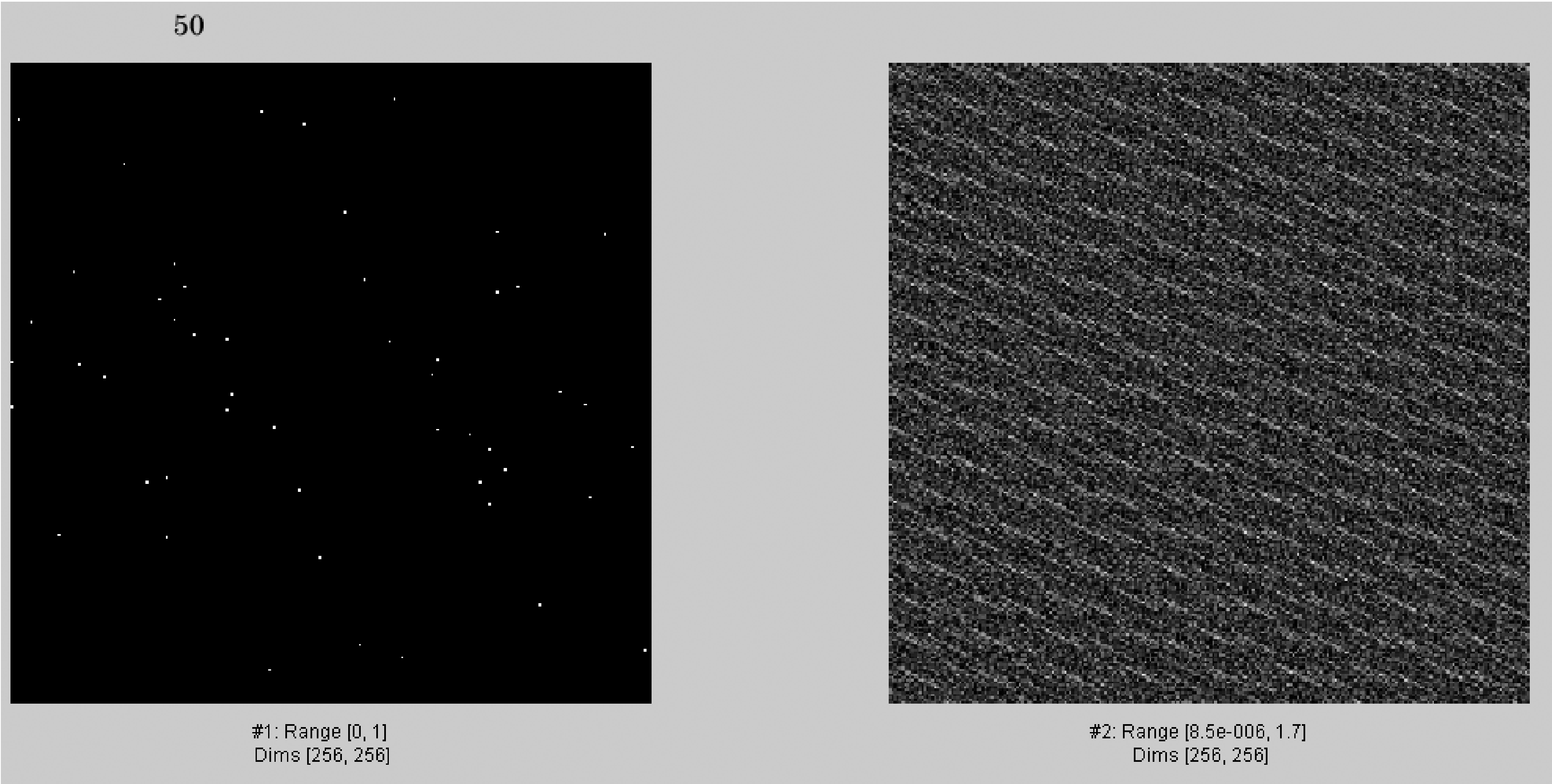
6



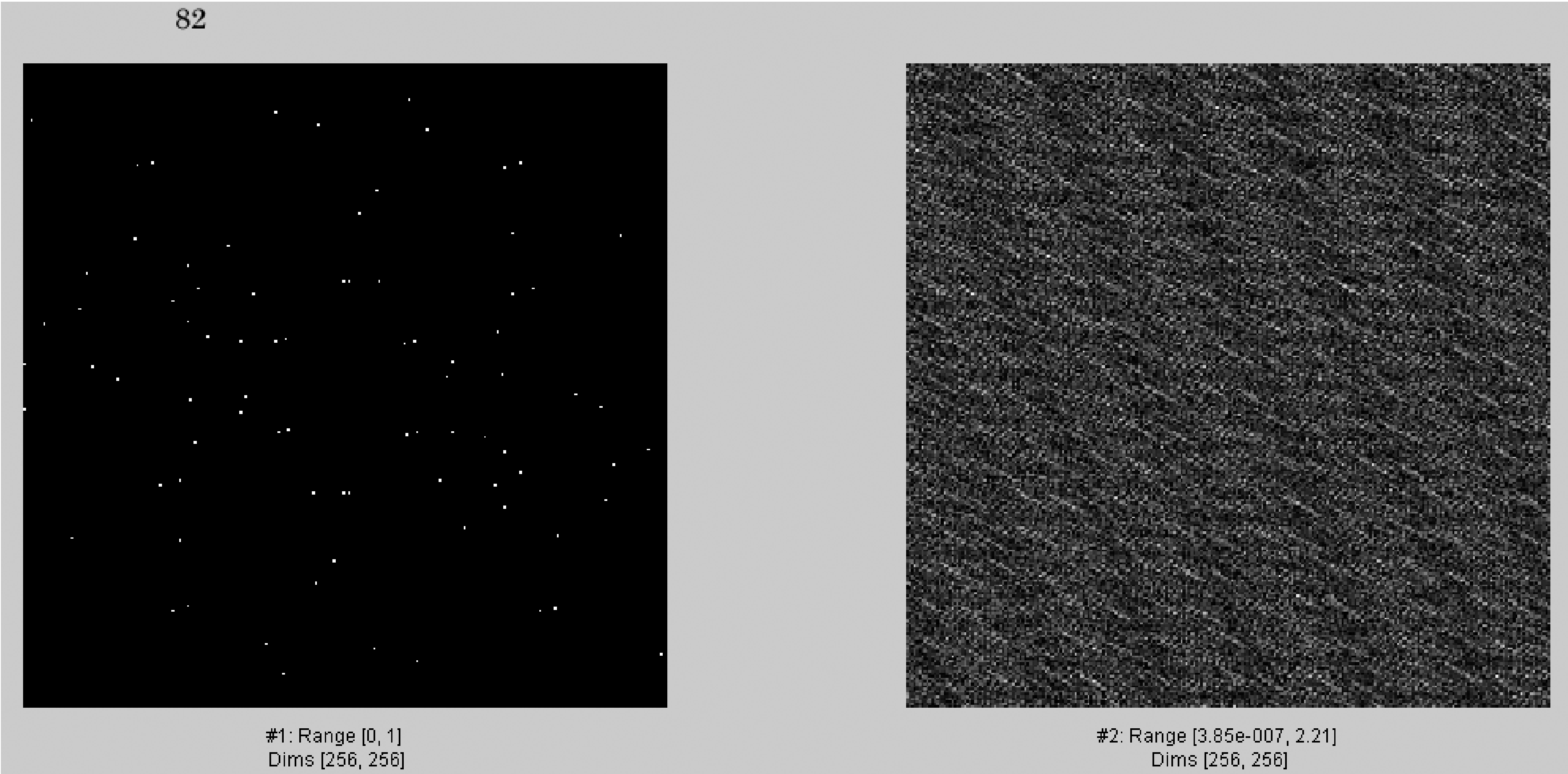
18



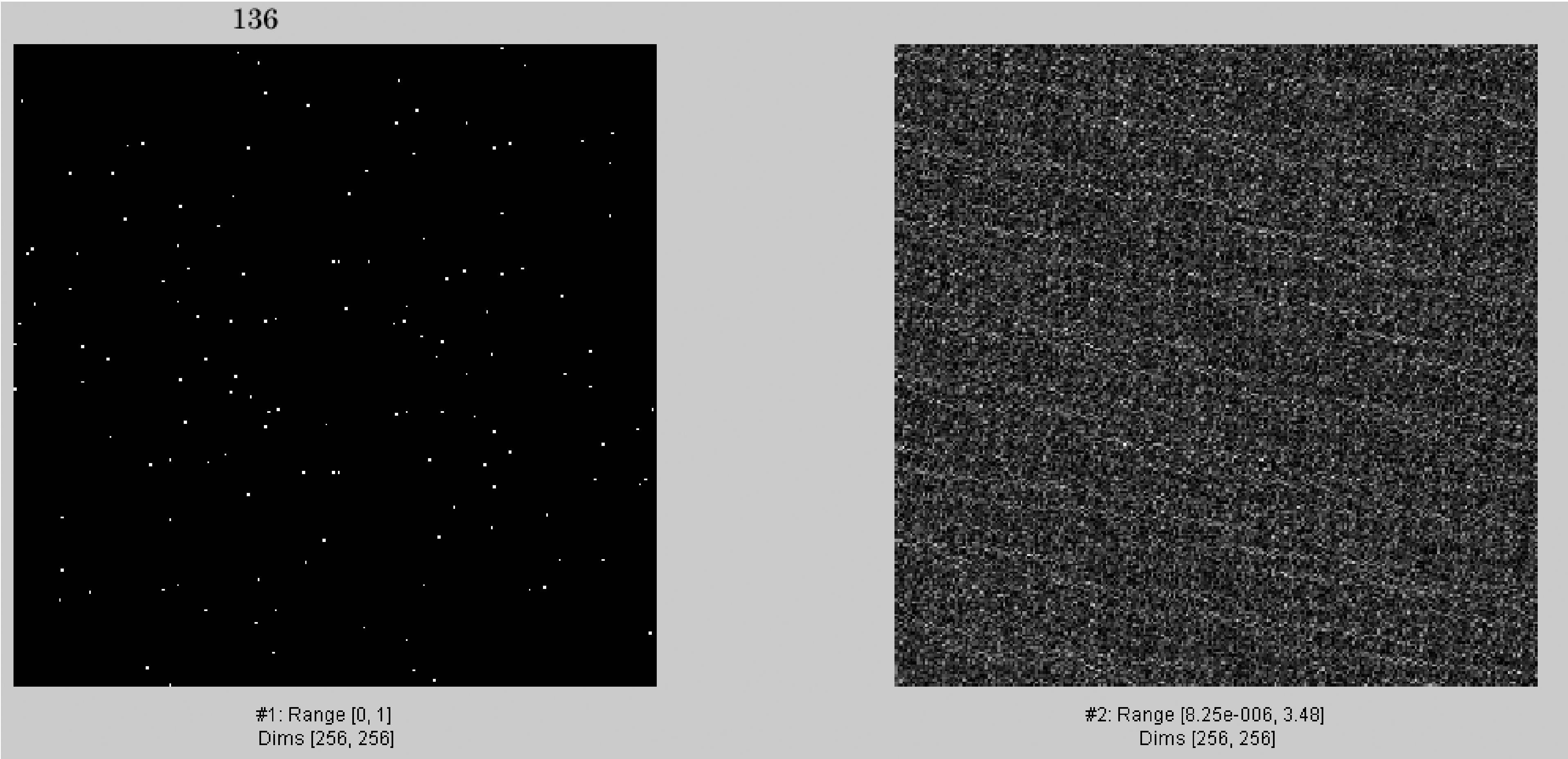
50



82

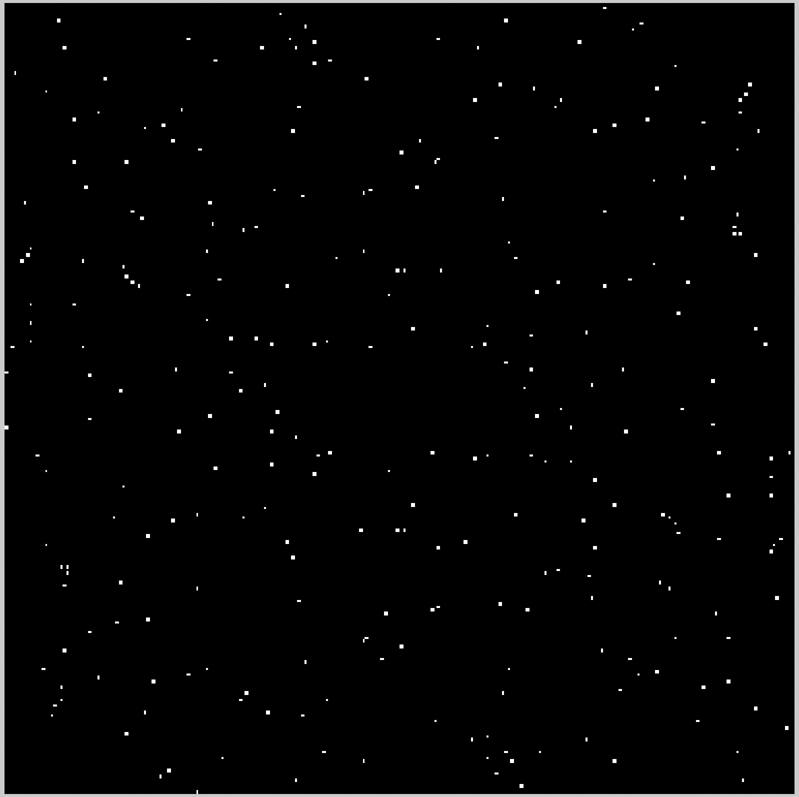


136



282

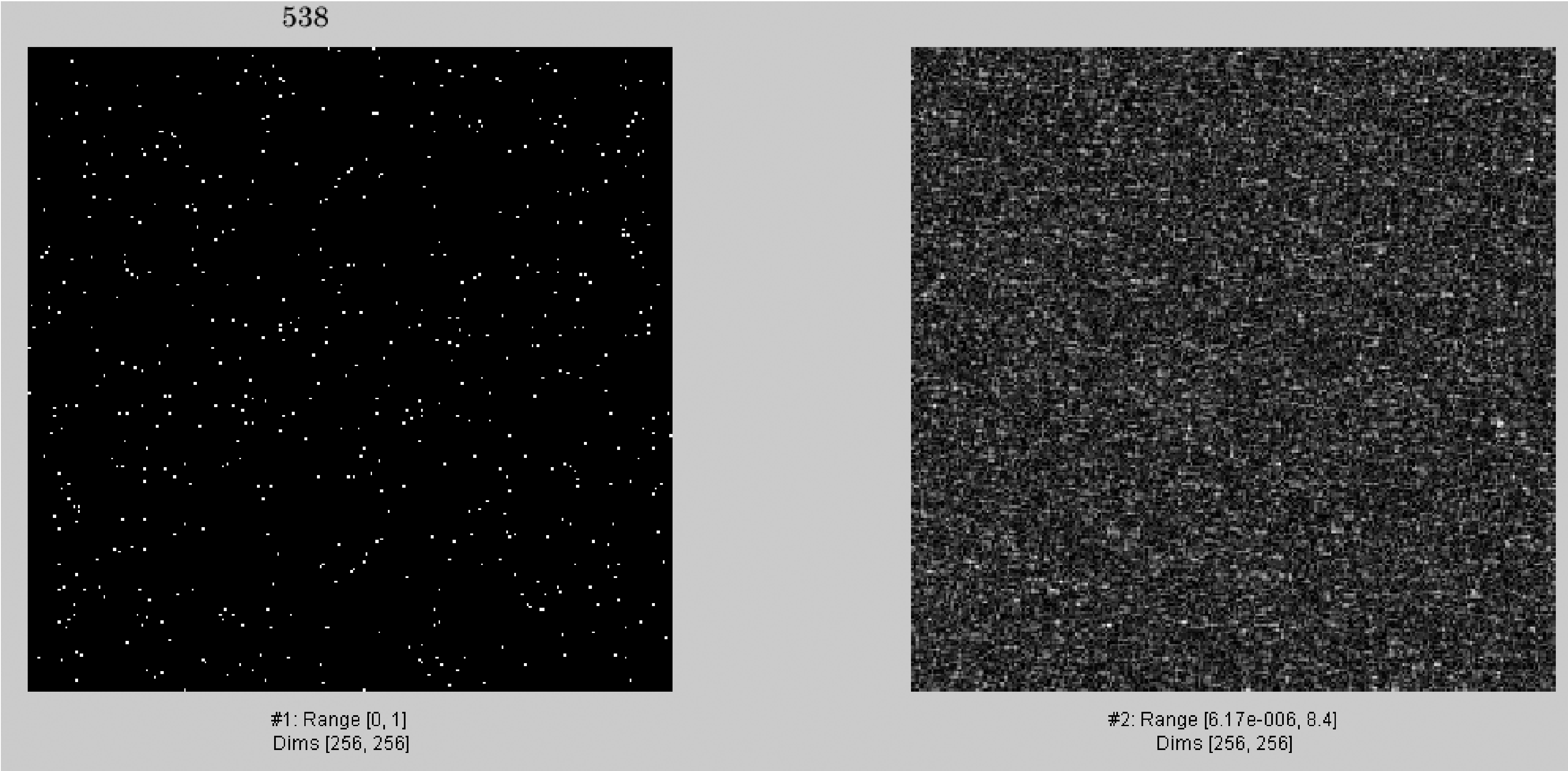
282



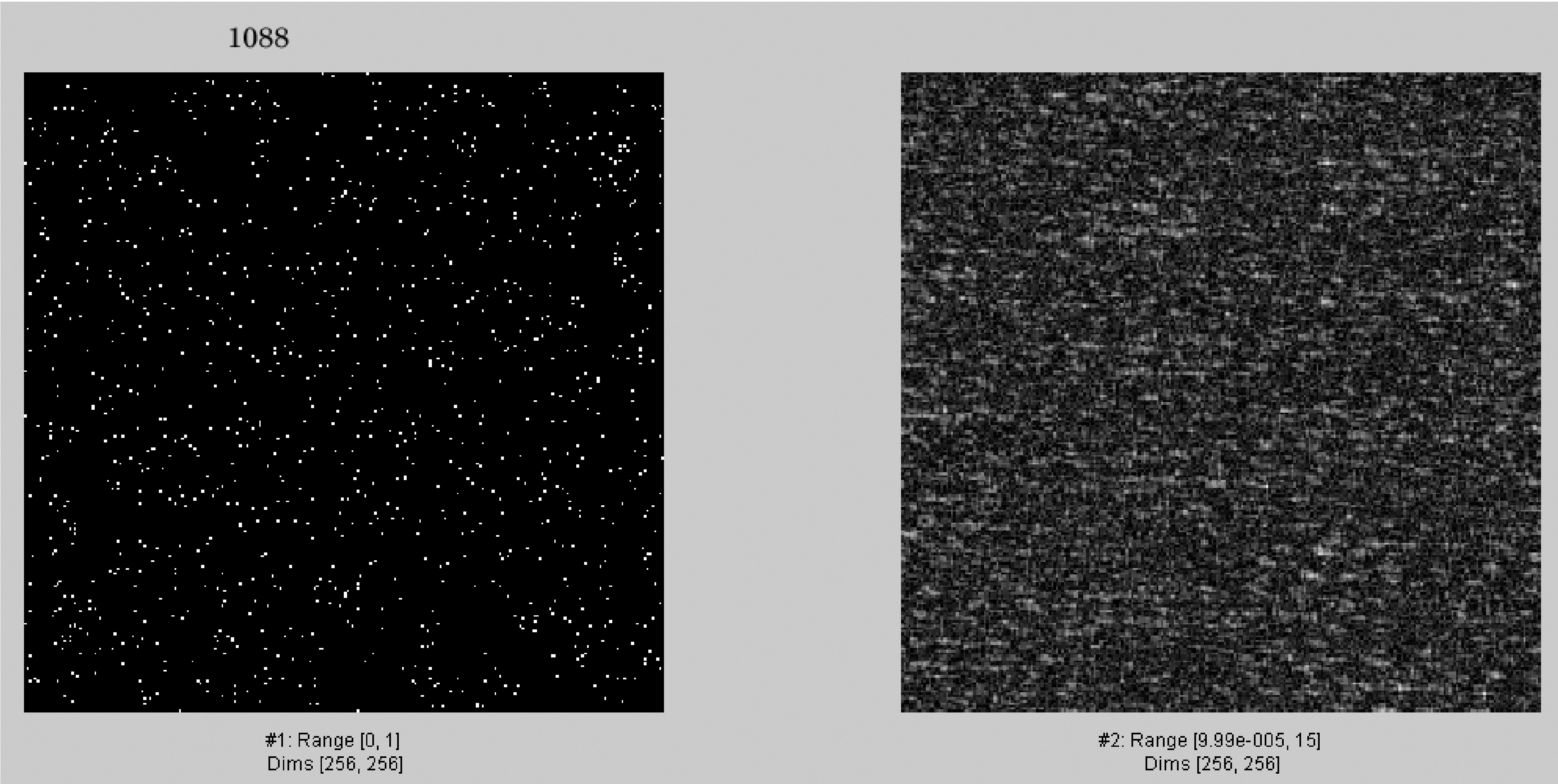
#1: Range [0, 1]
Dims [256, 256]

#2: Range [1.39e-005, 5.88]
Dims [256, 256]

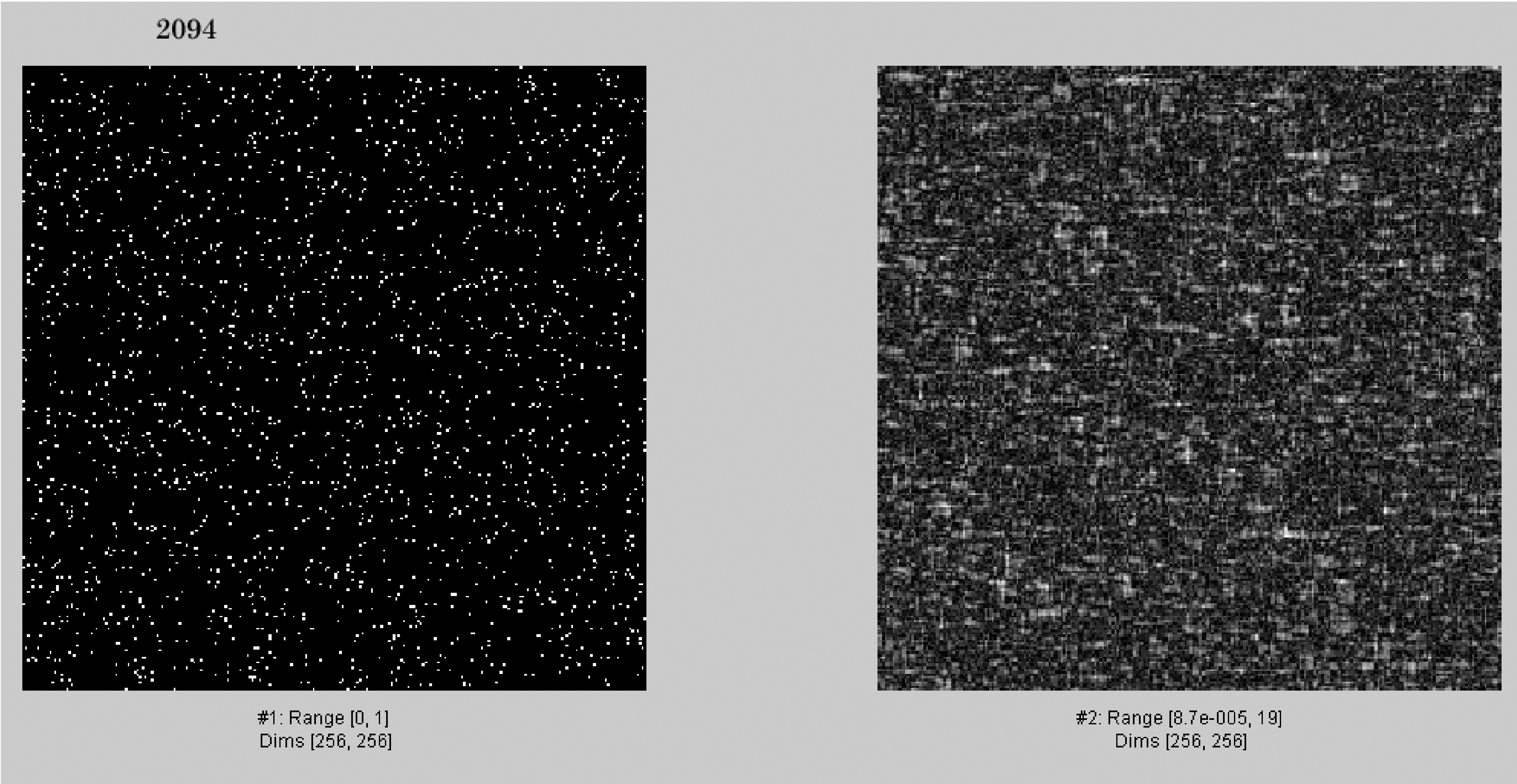
538



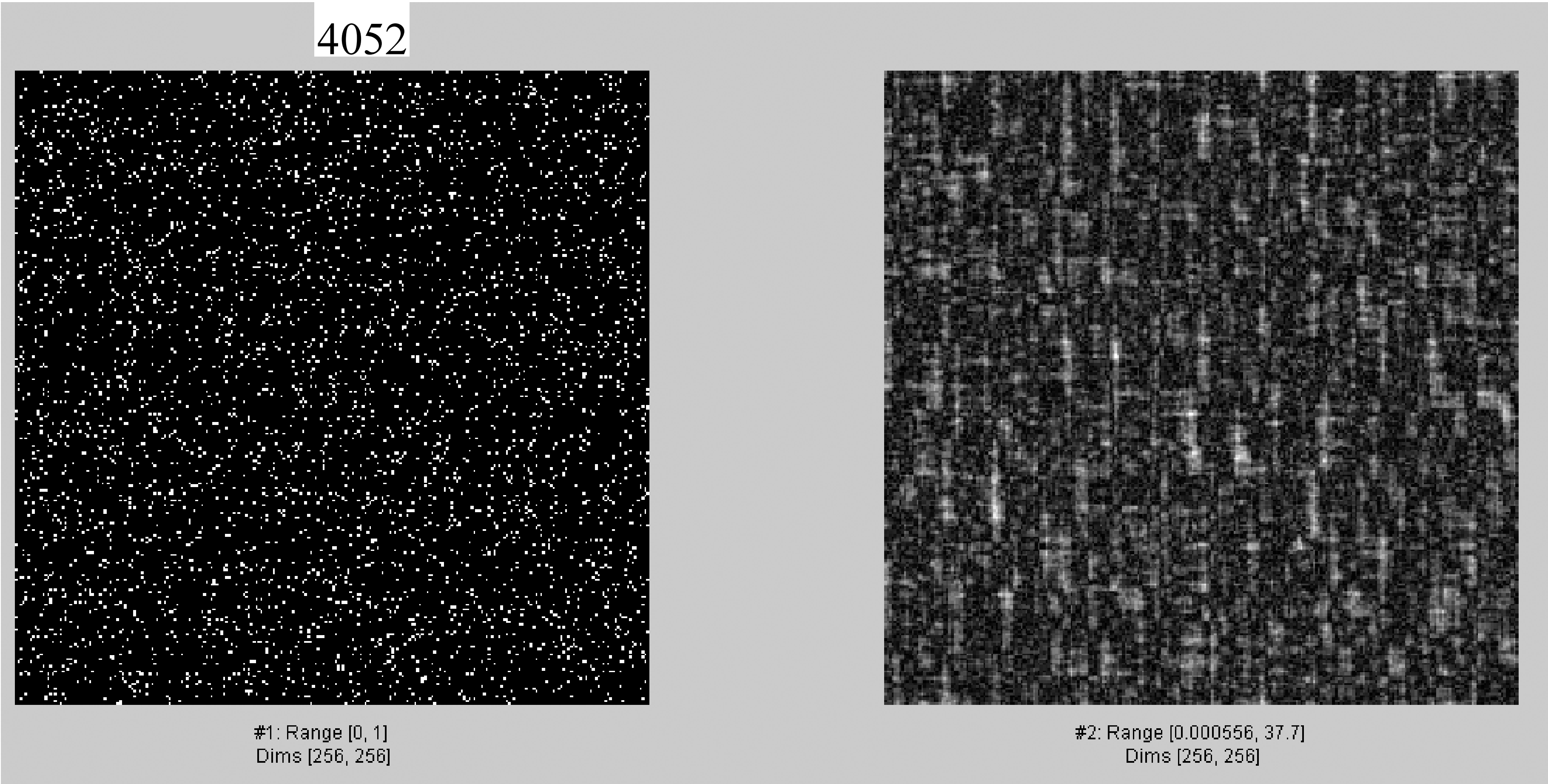
1088



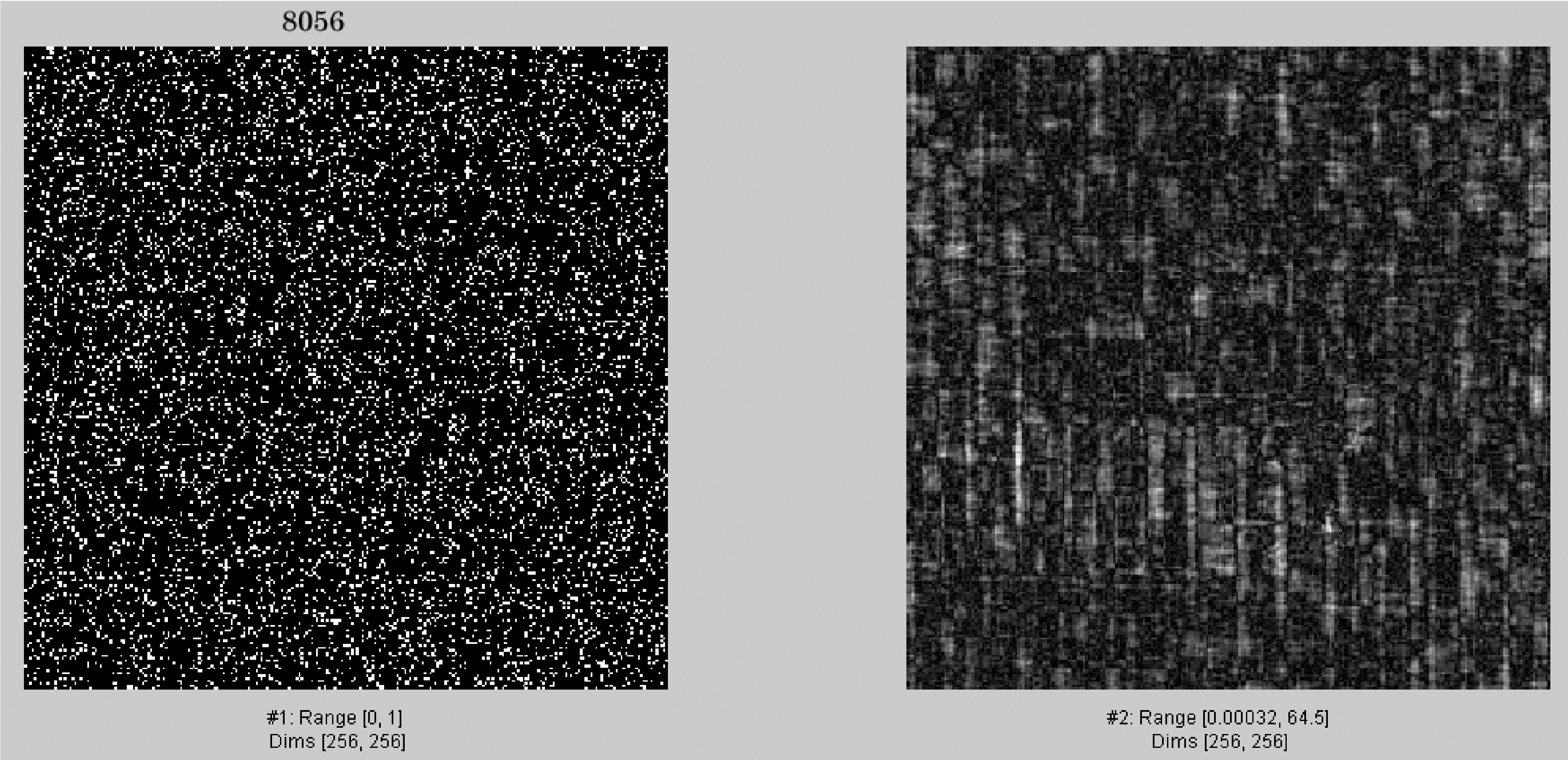
2094



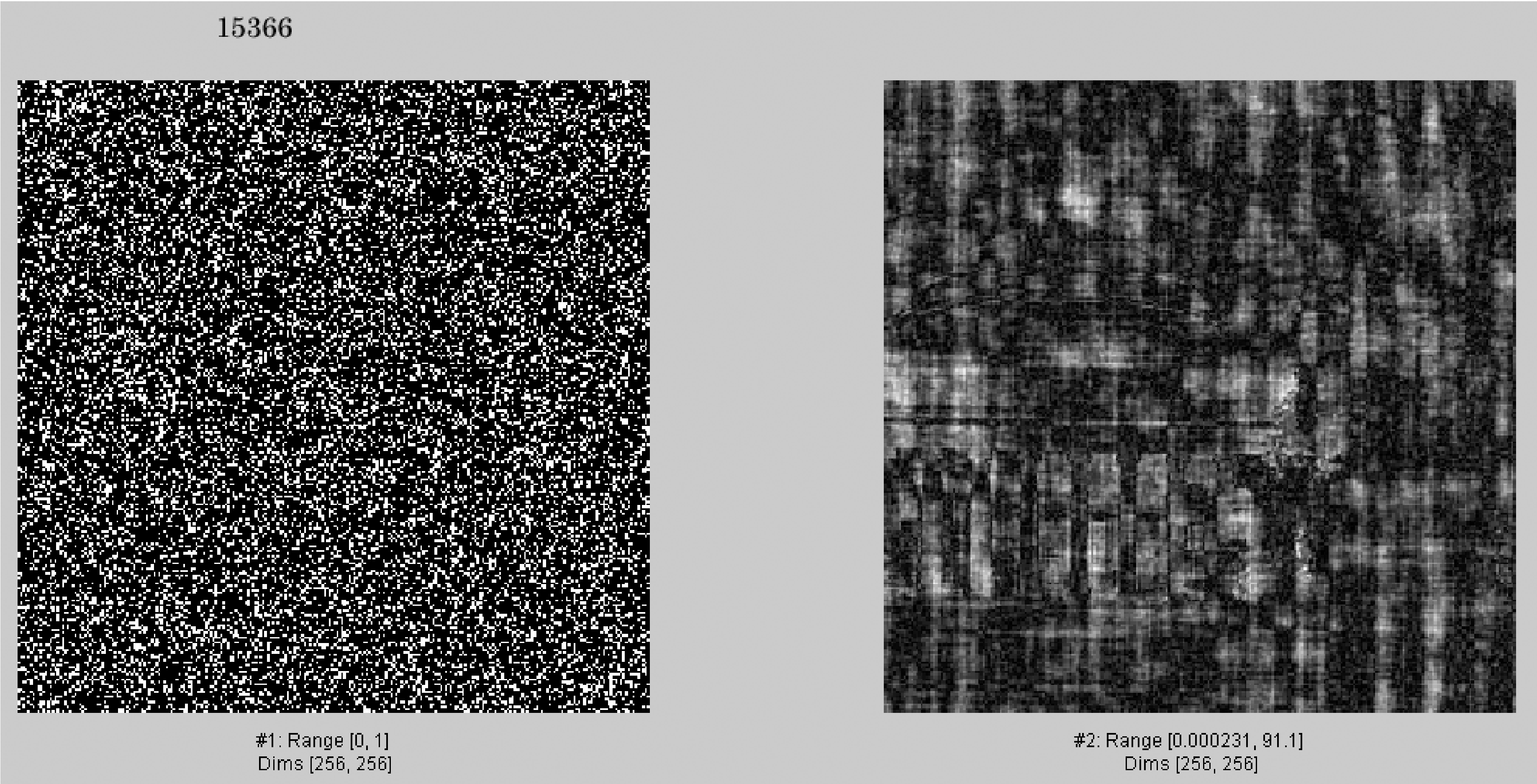
4052.



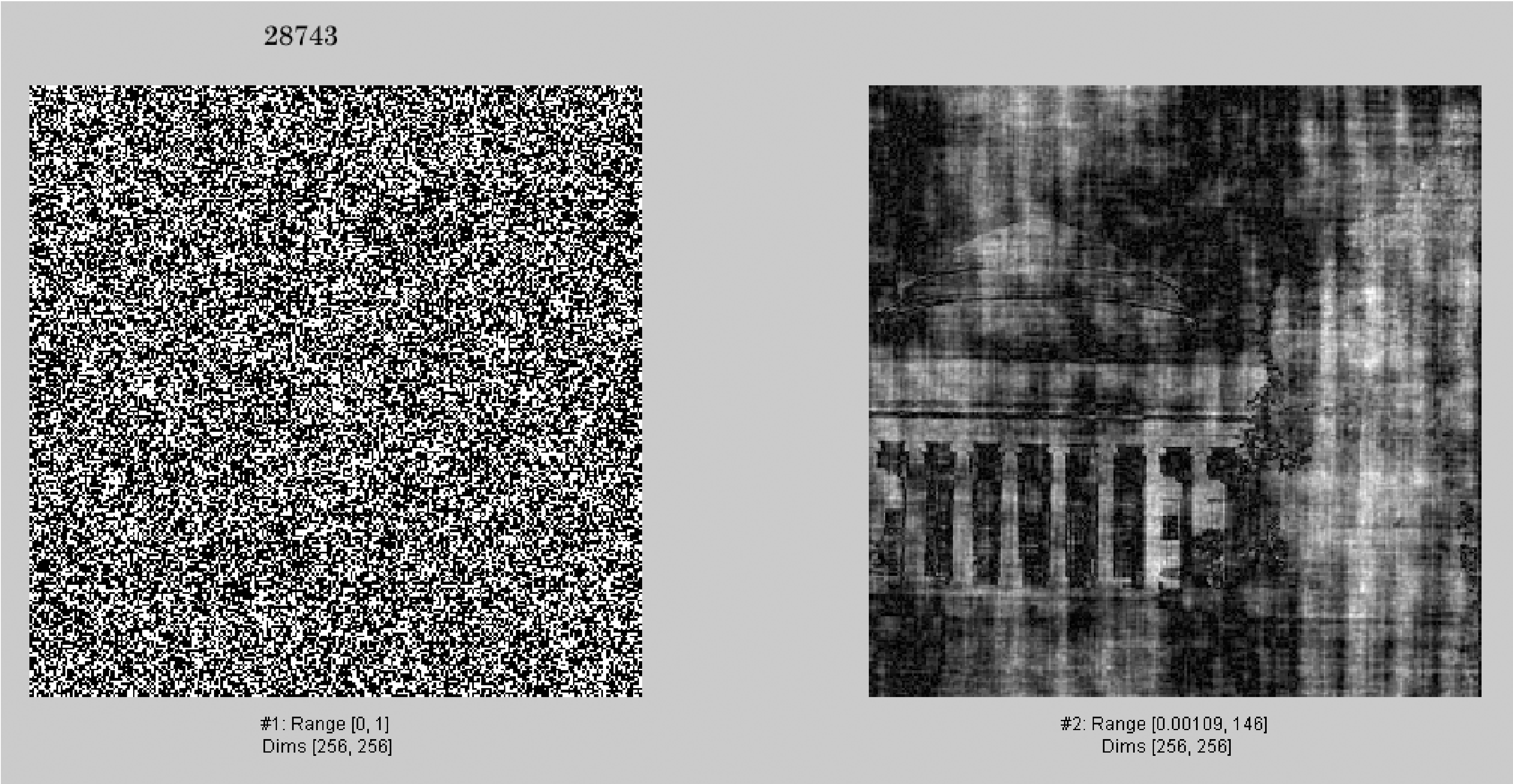
8056.



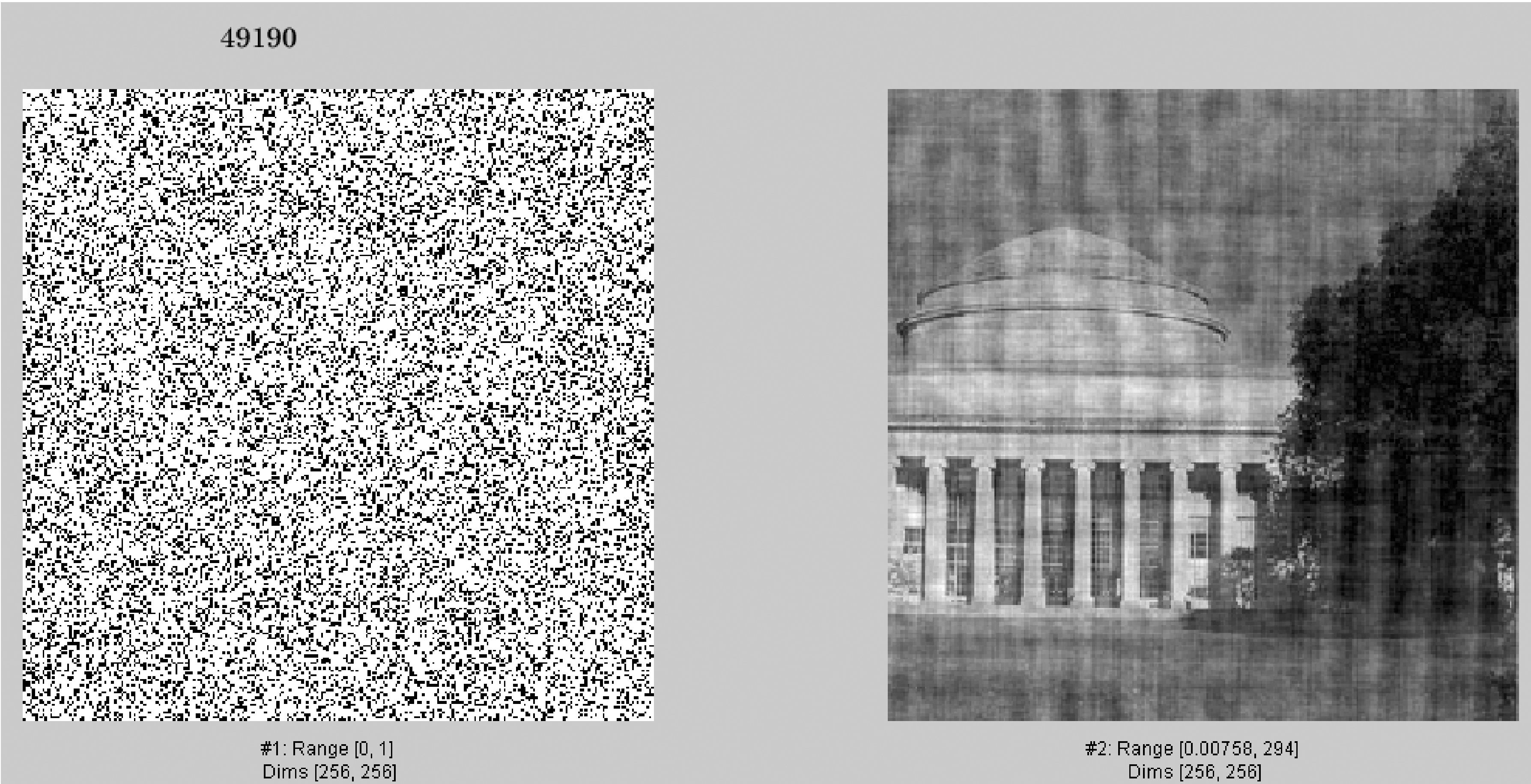
15366



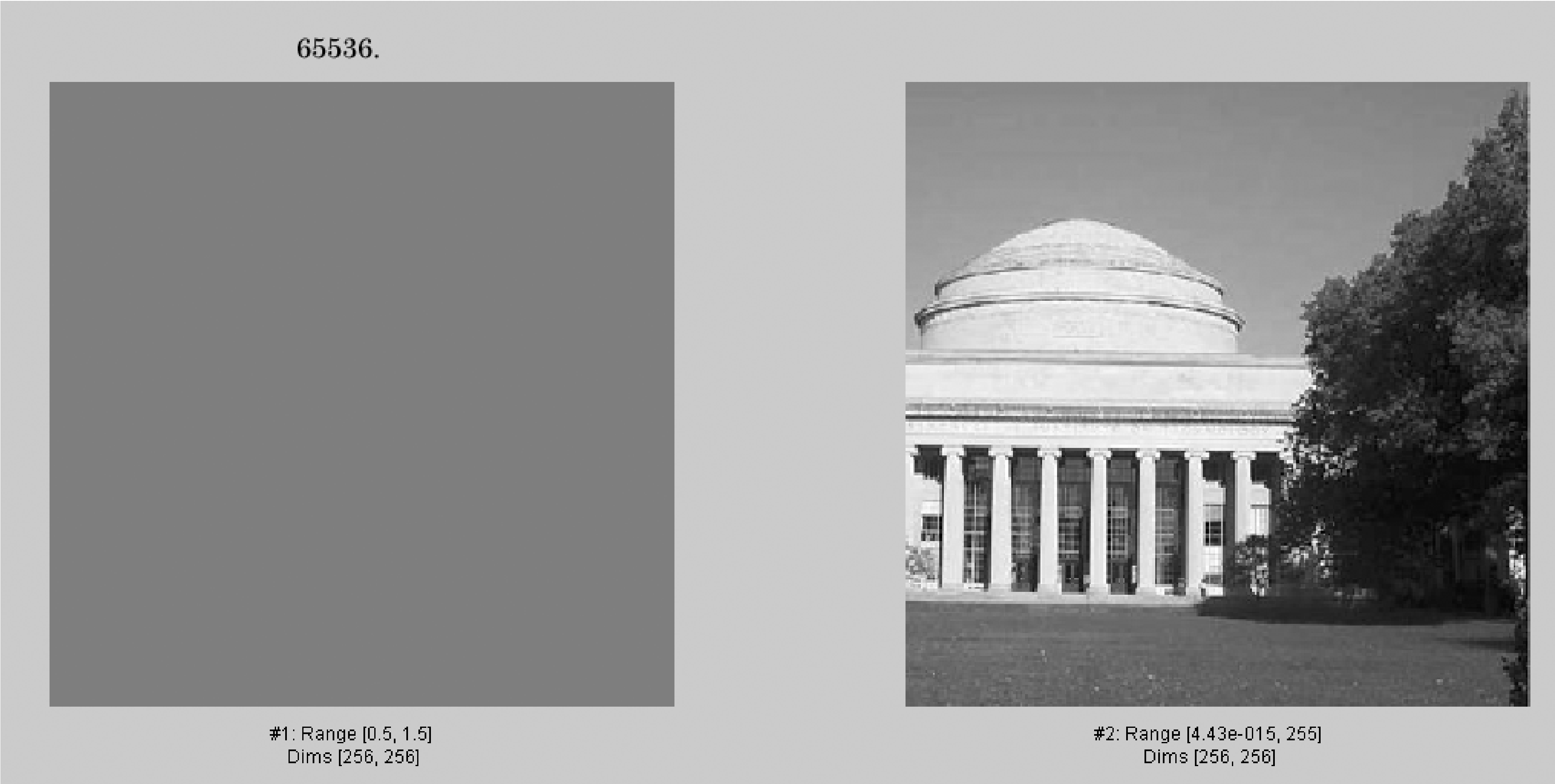
28743



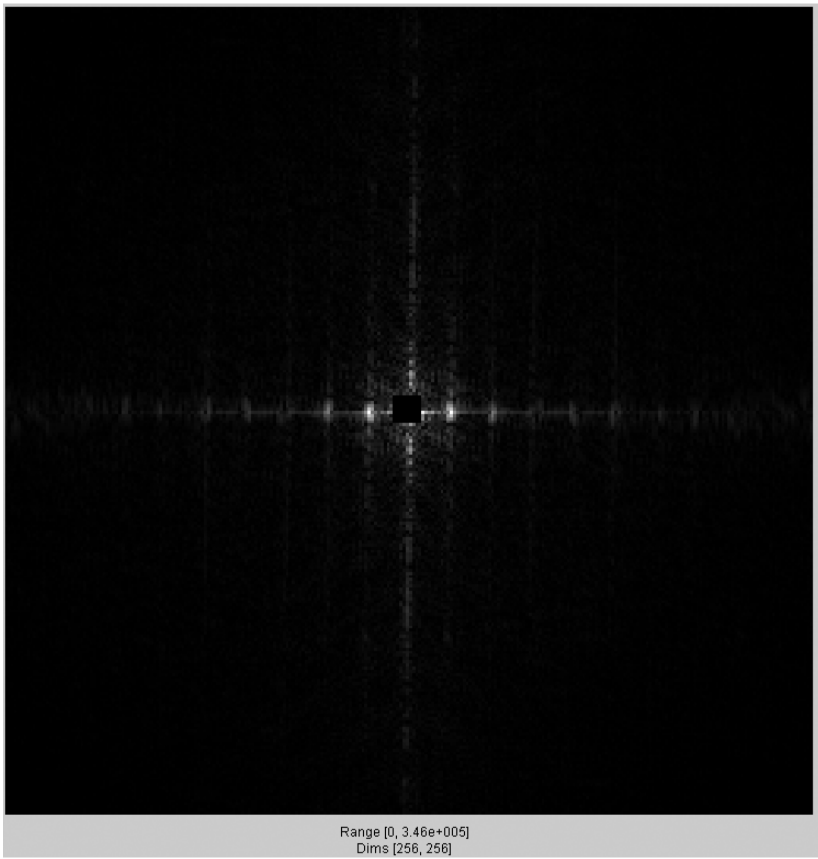
49190.

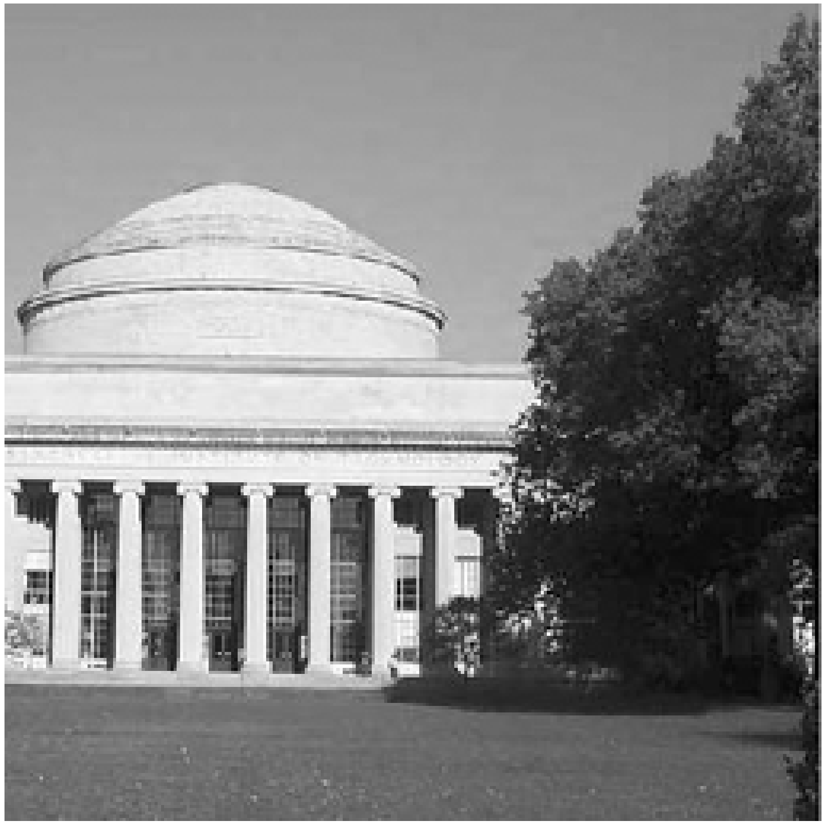


65536.

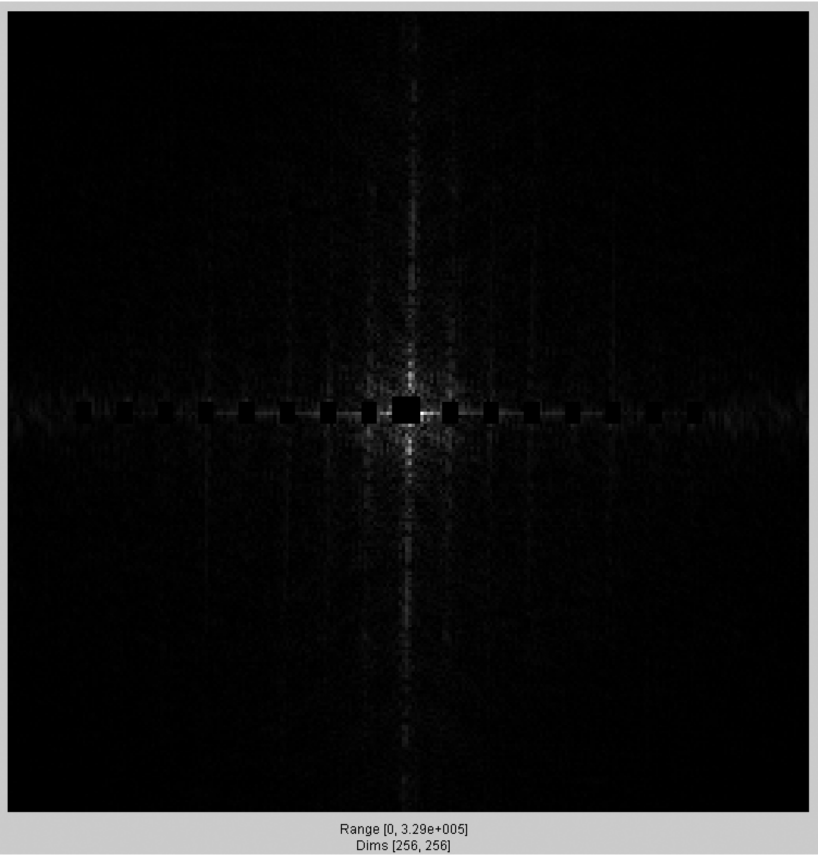

65536.
#1: Range [0.5, 1.5]
Dims [256, 256]
#2: Range [4.43e-015, 255]
Dims [256, 256]

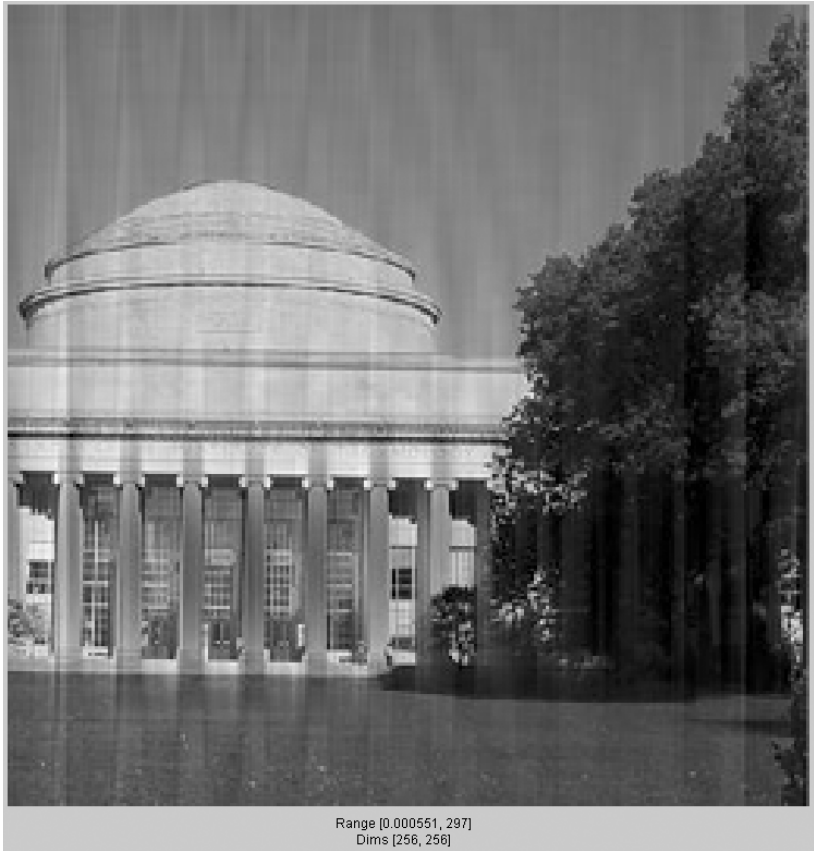
Fourier transform magnitude





Masking out the fundamental and harmonics from periodic pillars





Name as many functions as you can that retain that same functional form in the transform domain

TABLE 7.1 A variety of functions of two dimensions and their Fourier transforms. This table can be used in two directions (with appropriate substitutions for u, v and x, y) because the Fourier transform of the Fourier transform of a function is the function. Observant readers may suspect that the results on infinite sums of δ functions contradict the linearity of Fourier transforms. By careful inspection of limits, it is possible to show that they do not (see, e.g., Bracewell, 1995). Observant readers may also have noted that an expression for $\mathcal{F}(\frac{\partial f}{\partial y})$ can be obtained by combining two lines of this table.

Function	Fourier transform
$g(x, y)$	$\iint_{-\infty}^{\infty} g(x, y)e^{-i2\pi(ux+vy)}\, dx\, dy$
$\iint_{-\infty}^{\infty} \mathcal{F}(g(x, y))(u, v)e^{i2\pi(ux+vy)}\, du\, dv$	$\mathcal{F}(g(x, y))(u, v)$
$\delta(x, y)$	1
$\frac{\partial f}{\partial x}(x, y)$	$u\mathcal{F}(f)(u, v)$
$0.5\delta(x + a, y) + 0.5\delta(x - a, y)$	$\cos 2\pi a u$
$e^{-\pi(x^2+y^2)}$	$e^{-\pi(u^2+v^2)}$
$box_1(x, y)$	$\frac{\sin u}{u}\frac{\sin v}{v}$
$f(ax, by)$	$\frac{\mathcal{F}(f)(u/a, v/b)}{ab}$
$\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(x - i, y - j)$	$\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(u - i, v - j)$
$(f * *g)(x, y)$	$\mathcal{F}(f)\mathcal{F}(g)(u, v)$
$f(x - a, y - b)$	$e^{-i2\pi(au+bv)}\mathcal{F}(f)$
$f(x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$	$\mathcal{F}(f)(u\cos\theta - v\sin\theta, u\sin\theta + v\cos\theta)$

Discrete-time, continuous frequency Fourier transform

Many sequences can be represented by a Fourier integral of the form

$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega, \qquad (2.133)$$

where

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}. \qquad (2.134)$$

Oppenheim,
Schafer and
Buck,
Discrete-time
signal processing,
Prentice Hall,
1999

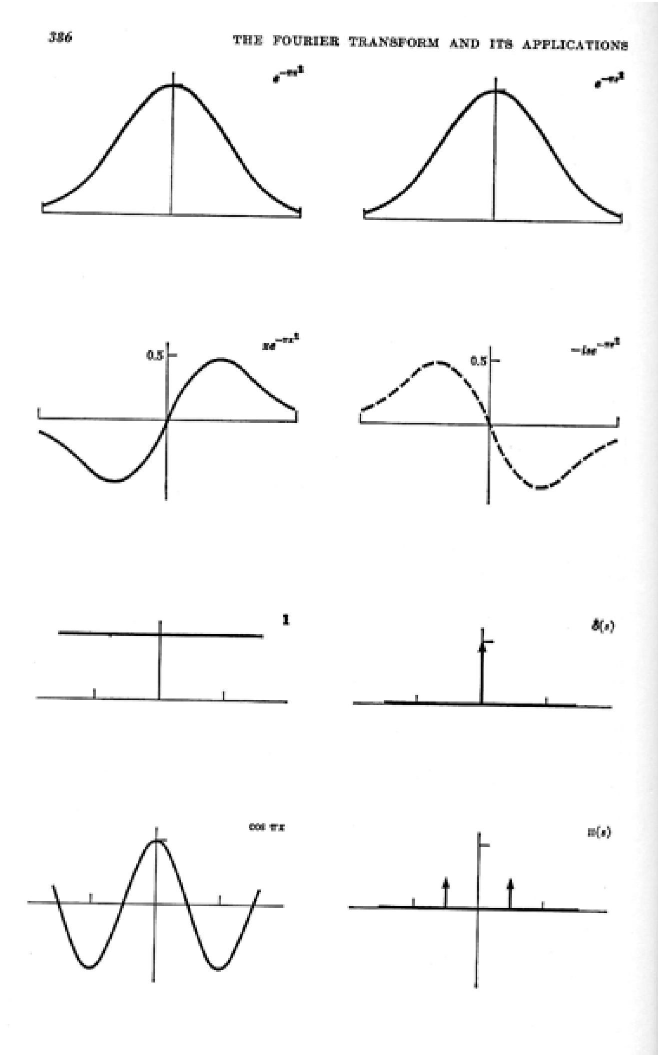
Discrete-time, continuous frequency Fourier transform pairs

TABLE 2.3 FOURIER TRANSFORM PAIRS

Sequence	Fourier Transform
1. $\delta[n]$	1
2. $\delta[n-n_0]$	$e^{-j\omega n_0}$
3. $1 \quad (-\infty < n < \infty)$	$\sum_{k=-\infty}^{\infty} 2\pi\delta(\omega+2\pi k)$
4. $a^n u[n] \quad (\|a\| < 1)$	$\frac{1}{1-ae^{-j\omega}}$
5. $u[n]$	$\frac{1}{1-e^{-j\omega}} + \sum_{k=-\infty}^{\infty} \pi\delta(\omega+2\pi k)$
6. $(n+1)a^n u[n] \quad (\|a\| < 1)$	$\frac{1}{(1-ae^{-j\omega})^2}$
7. $\frac{r^n \sin\omega_p(n+1)}{\sin\omega_p} u[n] \quad (\|r\| < 1)$	$\frac{1}{1-2r\cos\omega_p e^{-j\omega}+r^2e^{-j2\omega}}$
8. $\frac{\sin\omega_c n}{\pi n}$	$X(e^{j\omega}) = \begin{cases} 1, & \|\omega\| < \omega_c, \\ 0, & \omega_c < \|\omega\| \le \pi \end{cases}$
9. $x[n] = \begin{cases} 1, & 0 \le n \le M \\ 0, & \text{otherwise} \end{cases}$	$\frac{\sin[\omega(M+1)/2]}{\sin(\omega/2)} e^{-j\omega M/2}$
10. $e^{j\omega_0 n}$	$\sum_{k=-\infty}^{\omega} 2\pi\delta(\omega-\omega_0+2\pi k)$
11. $\cos(\omega_0 n+\phi)$	$\sum_{k=-\infty}^{\infty} [\pi e^{j\phi}\delta(\omega-\omega_0+2\pi k)+\pi e^{-j\phi}\delta(\omega+\omega_0+2\pi k)]$

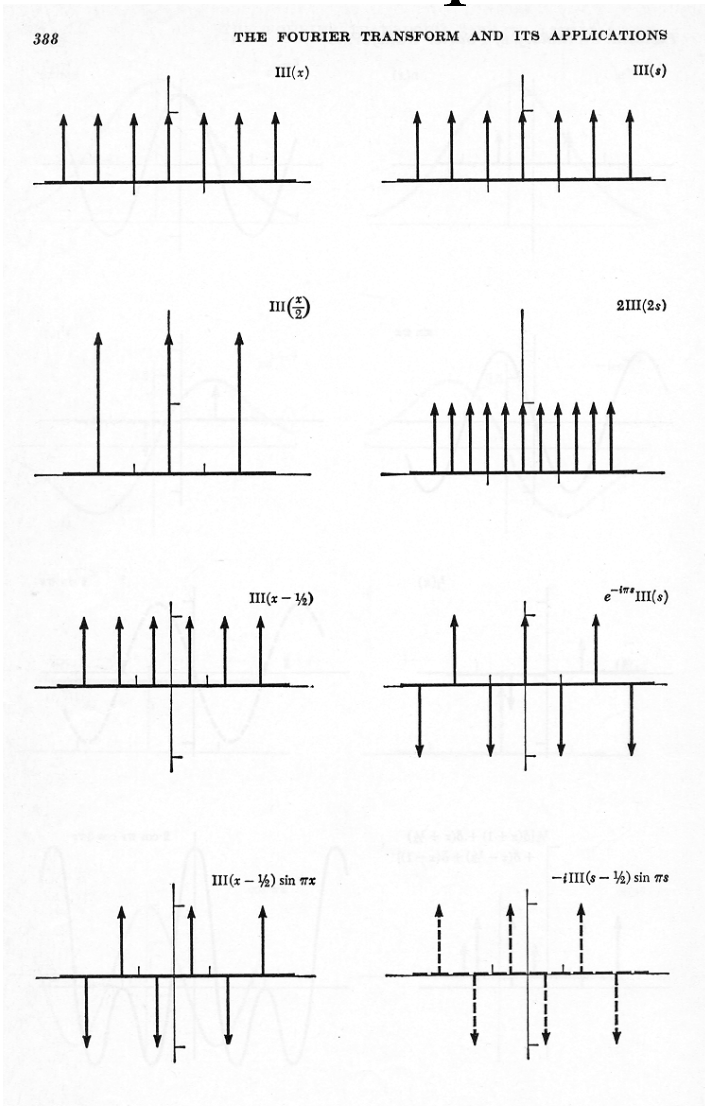
Oppenheim, Schafer and Buck, Discrete-time signal processing, Prentice Hall, 1999

Bracewell's pictorial dictionary of Fourier transform pairs



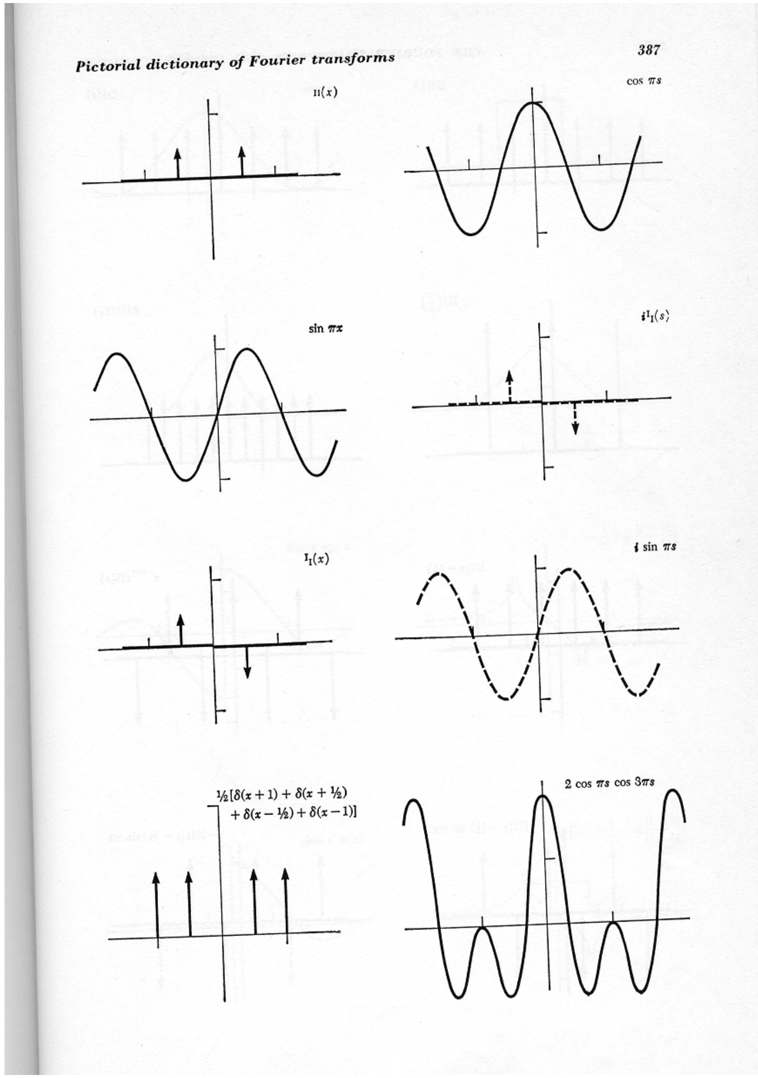
Bracewell, The Fourier Transform and its Applications, McGraw Hill 1978

Bracewell's pictorial dictionary of Fourier transform pairs



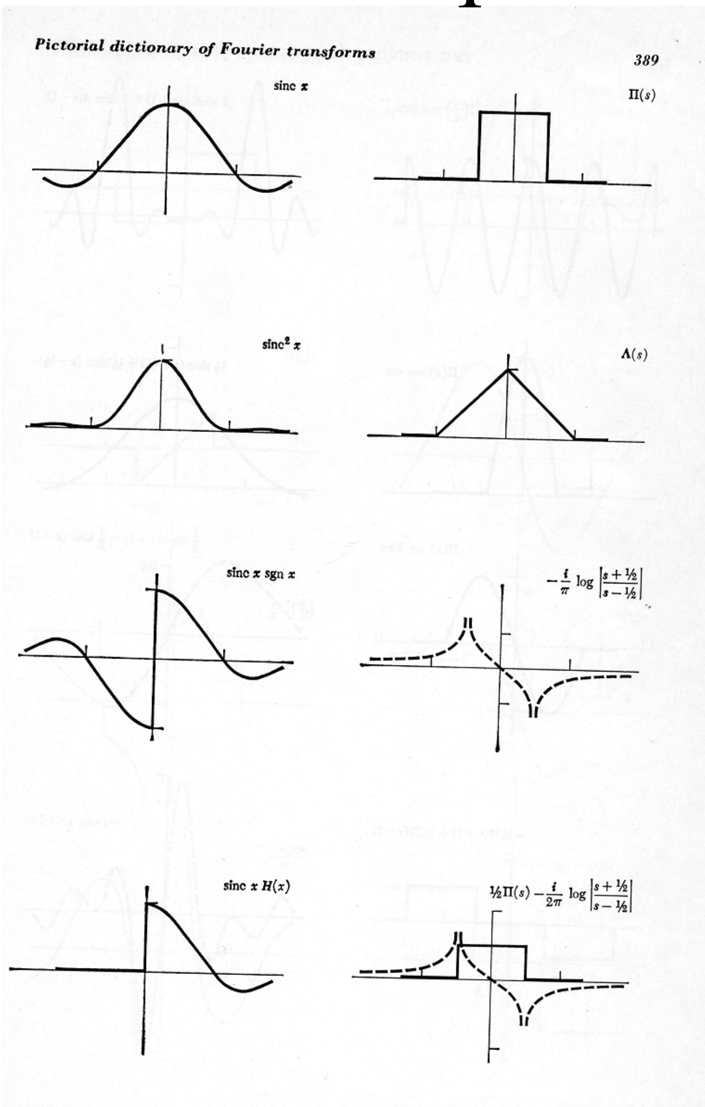
Bracewell, The Fourier Transform and its Applications, McGraw Hill 1978

Bracewell's pictorial dictionary of Fourier transform pairs



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

Bracewell's pictorial dictionary of Fourier transform pairs



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

Why is the Fourier domain particularly useful?

- It tells us the effect of linear convolutions.

Fourier transform of convolution

Consider a (circular) convolution of g and h

$$f = g \otimes h$$

Fourier transform of convolution

$f = g \otimes h$

Take DFT of both sides

$$F[m,n] = DFT\left(g \otimes h\right)$$

Fourier transform of convolution

$$f = g \otimes h$$

$$F[m,n] = DFT(g \otimes h)$$

Write the DFT and convolution explicitly

$$F[m,n] = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \sum_{k,l} g[u-k, v-l] h[k,l] e^{-\pi i \left(\frac{um}{M} + \frac{vn}{N} \right)}$$

Fourier transform of convolution

$$f = g \otimes h$$

$$F[m,n] = DFT(g \otimes h)$$

$$F[m,n] = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \sum_{k,l} g[u-k, v-l] h[k,l] e^{-\pi i \left(\frac{um}{M} + \frac{vn}{N} \right)}$$

Move the exponent in

$$= \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \sum_{k,l} g[u-k, v-l] e^{-\pi i \left(\frac{um}{M} + \frac{vn}{N} \right)} h[k,l]$$

Fourier transform of convolution

$$f = g \otimes h$$

$$F[m,n] = DFT(g \otimes h)$$

$$F[m,n] = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\sum_{k,l} g[u-k,v-l]h[k,l]e^{-\pi i\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

$$= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\sum_{k,l} g[u-k,v-l]e^{-\pi i\left(\frac{um}{M}+\frac{vn}{N}\right)}h[k,l]$$

Change variables in the sum

$$= \sum_{\mu=-k}^{M-k-1}\sum_{\upsilon=-l}^{N-l-1}\sum_{k,l} g[\mu,\upsilon]e^{-\pi i\left(\frac{(k+\mu)m}{M}+\frac{(l+\upsilon)n}{N}\right)}h[k,l]$$

Fourier transform of convolution

$$f = g \otimes h$$

$$F[m,n] = DFT(g \otimes h)$$

$$F[m,n] = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\sum_{k,l} g[u-k,v-l]h[k,l]e^{-\pi i\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

$$= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\sum_{k,l} g[u-k,v-l]e^{-\pi i\left(\frac{um}{M}+\frac{vn}{N}\right)}h[k,l]$$

$$= \sum_{\mu=-k}^{M-k-1}\sum_{\upsilon=-l}^{N-l-1}\sum_{k,l} g[\mu,\upsilon]e^{-\pi i\left(\frac{(k+\mu)m}{M}+\frac{(l+\upsilon)n}{N}\right)}h[k,l]$$

Perform the DFT (circular boundary conditions)

$$= \sum_{k,l} G[m,n]e^{-\pi i\left(\frac{km}{M}+\frac{\ln}{N}\right)}h[k,l]$$

Fourier transform of convolution

$$f = g \otimes h$$

$$F[m,n] = DFT(g \otimes h)$$

$$F[m,n] = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\sum_{k,l} g[u-k,v-l]h[k,l]e^{-\pi i\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

$$= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\sum_{k,l} g[u-k,v-l]e^{-\pi i\left(\frac{um}{M}+\frac{vn}{N}\right)}h[k,l]$$

$$= \sum_{\mu=-k}^{M-k-1}\sum_{\upsilon=-l}^{N-l-1}\sum_{k,l} g[\mu,\upsilon]e^{-\pi i\left(\frac{(k+\mu)m}{M}+\frac{(l+\upsilon)n}{N}\right)}h[k,l]$$

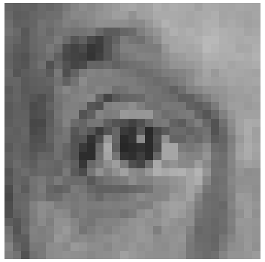
$$= \sum_{k,l} G[m,n]e^{-\pi i\left(\frac{km}{M}+\frac{\mathrm{ln}}{N}\right)}h[k,l]$$

Perform the other DFT (circular boundary conditions)

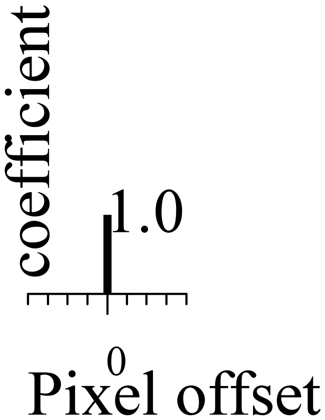
$$= G[m,n]H[m,n]$$

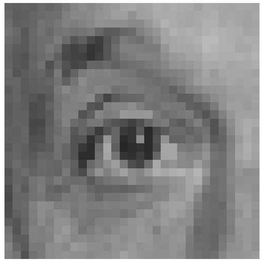
Analysis of our simple filters

Analysis of our simple filters



original

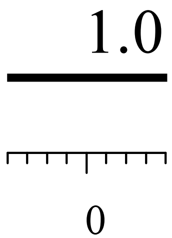




Filtered
(no change)

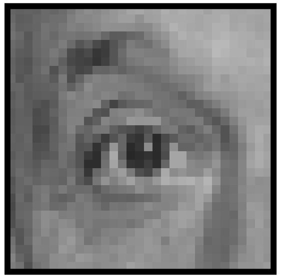
$$F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k,l] e^{-\pi i \left(\frac{km}{M} + \frac{\ln}{N} \right)}$$

$$= 1$$

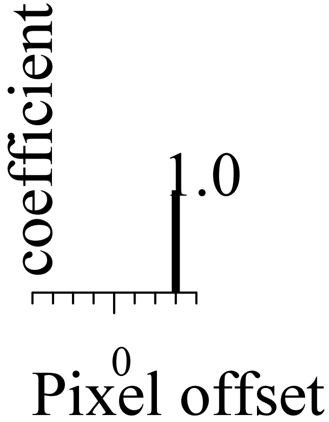


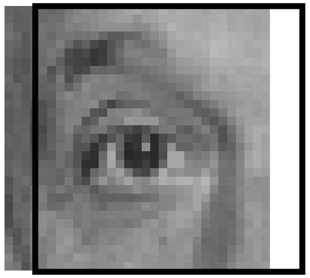
constant

Analysis of our simple filters



original

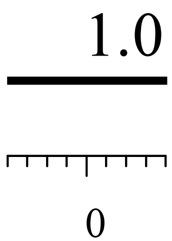




shifted

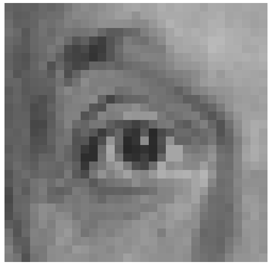
$$F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k-\delta, l] e^{-\pi i \left(\frac{km}{M} + \frac{\ln}{N} \right)}$$

$$= e^{-\pi i \frac{\delta m}{M}}$$

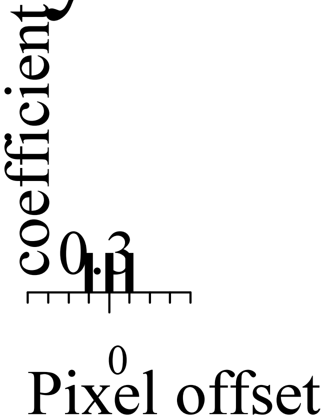


Constant magnitude, linearly shifted phase

Analysis of our simple filters



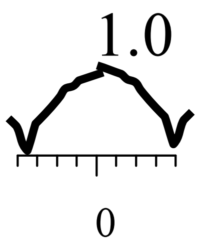
original



blurred

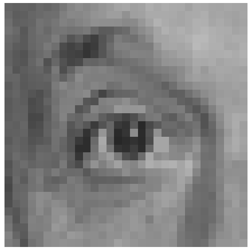
$$F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k,l] e^{-\pi i \left(\frac{km}{M} + \frac{\mathrm{ln}}{N} \right)}$$

$$= \frac{1}{3}\left(1 + 2\cos\left(\frac{\pi m}{M} \right) \right)$$

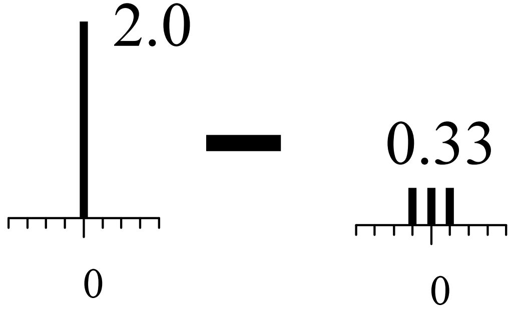


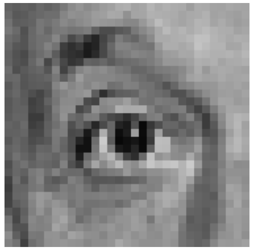
Low-pass filter

Analysis of our simple filters



original



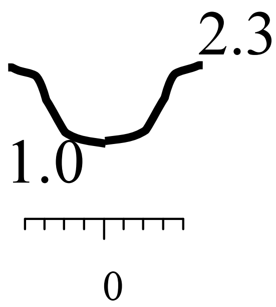


sharpened

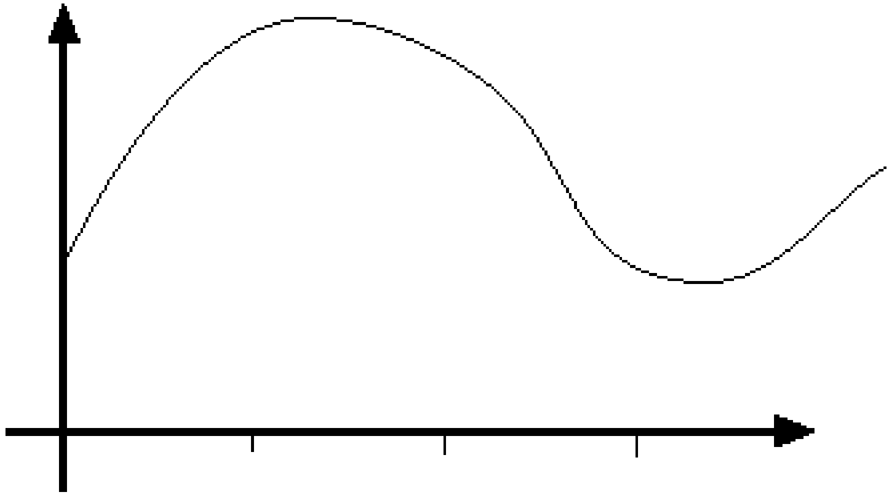
$$F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k,l] e^{-\pi i \left(\frac{km}{M} + \frac{\mathrm{ln}}{N} \right)}$$

$$= 2 - \frac{1}{3}\left(1 + 2\cos\left(\frac{\pi m}{M} \right) \right)$$

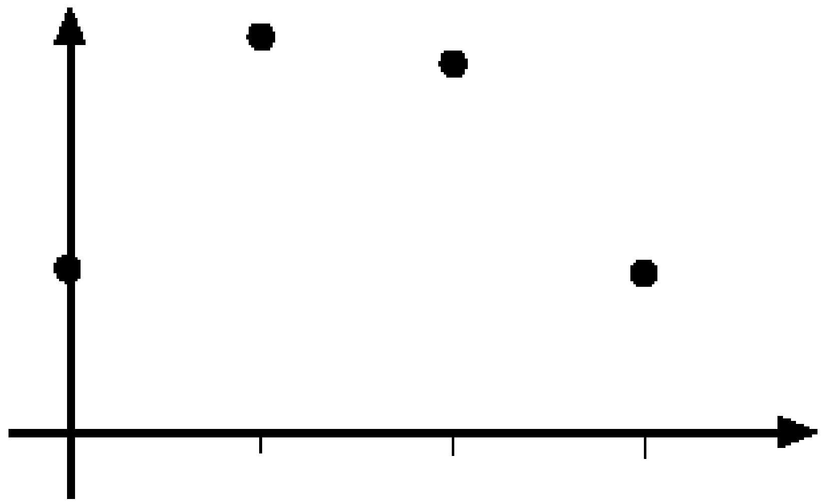
high-pass filter

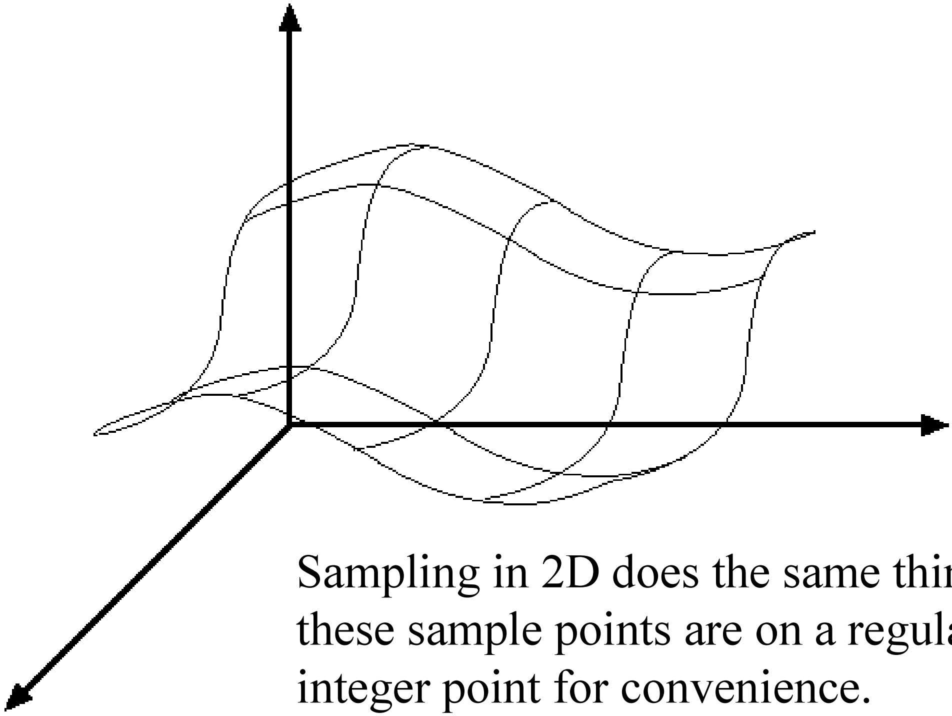


Sampling and aliasing

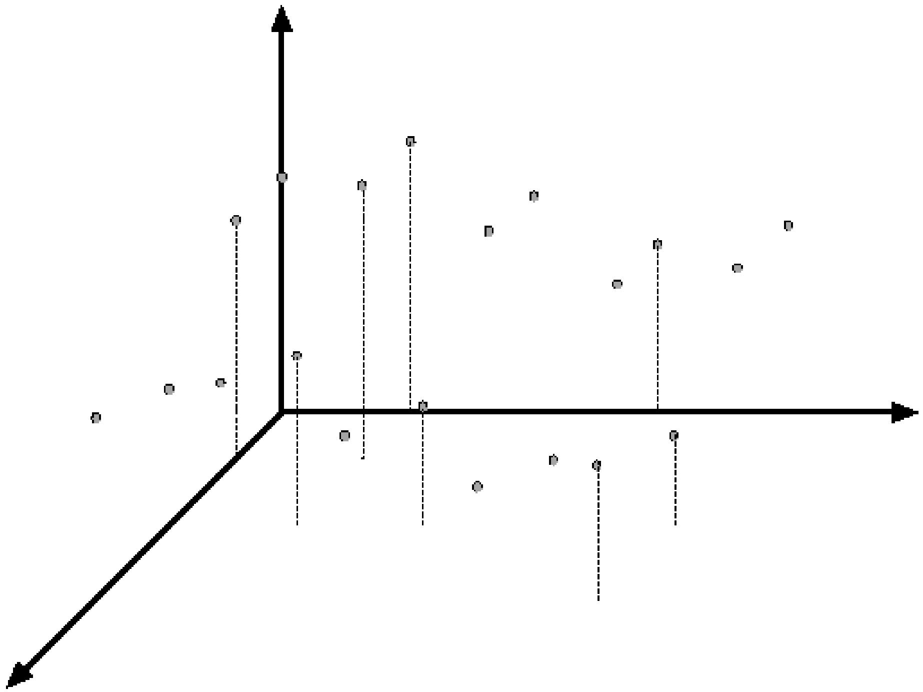


Sampling in 1D takes a continuous function and replaces it with a vector of values, consisting of the function's values at a set of sample points. We'll assume that these sample points are on a regular grid, and can place one at each integer for convenience.





Sampling in 2D does the same thing, only in 2D. We'll assume that these sample points are on a regular grid, and can place one at each integer point for convenience.



A continuous model for a sampled function

- We want to be able to approximate integrals sensibly
- Leads to
  - the delta function
  - model on right

$$\mathrm{Sample}_{\mathrm{2D}}\big(f(x,y)\big) = \sum_{i=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} f(x,y)\,\delta(x-i,y-j)$$

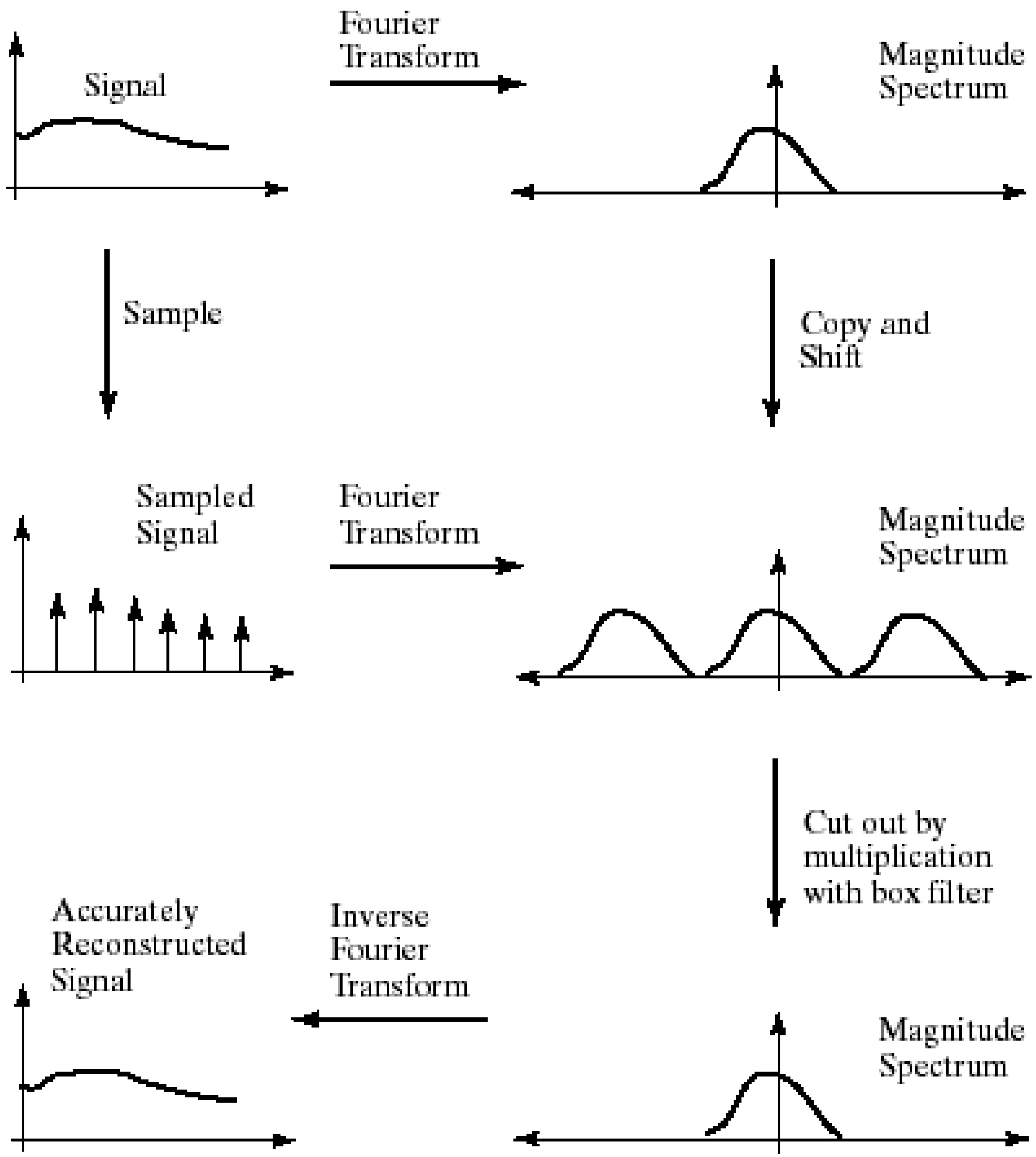
$$= f(x,y)\sum_{i=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} \delta(x-i,y-j)$$

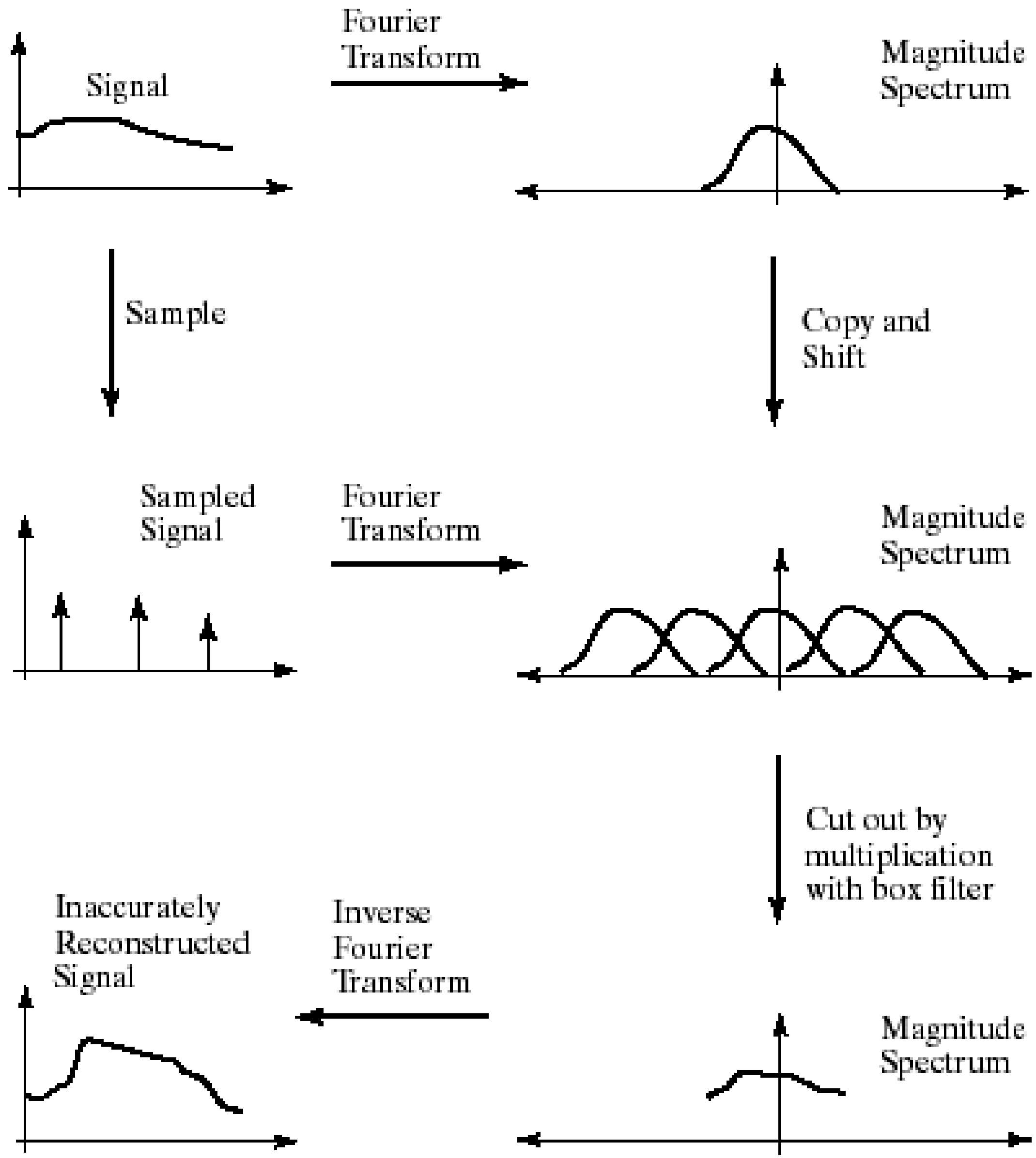
The Fourier transform of a sampled signal

$$F\left(\mathrm{Sample}_{2\mathrm{D}}\left(f(x,y)\right)\right) = F\left(f(x,y) \sum_{i=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} \delta(x-i, y-j) \right)$$

$$= F\left(f(x,y)\right) ** F\left(\sum_{i=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} \delta(x-i, y-j) \right)$$

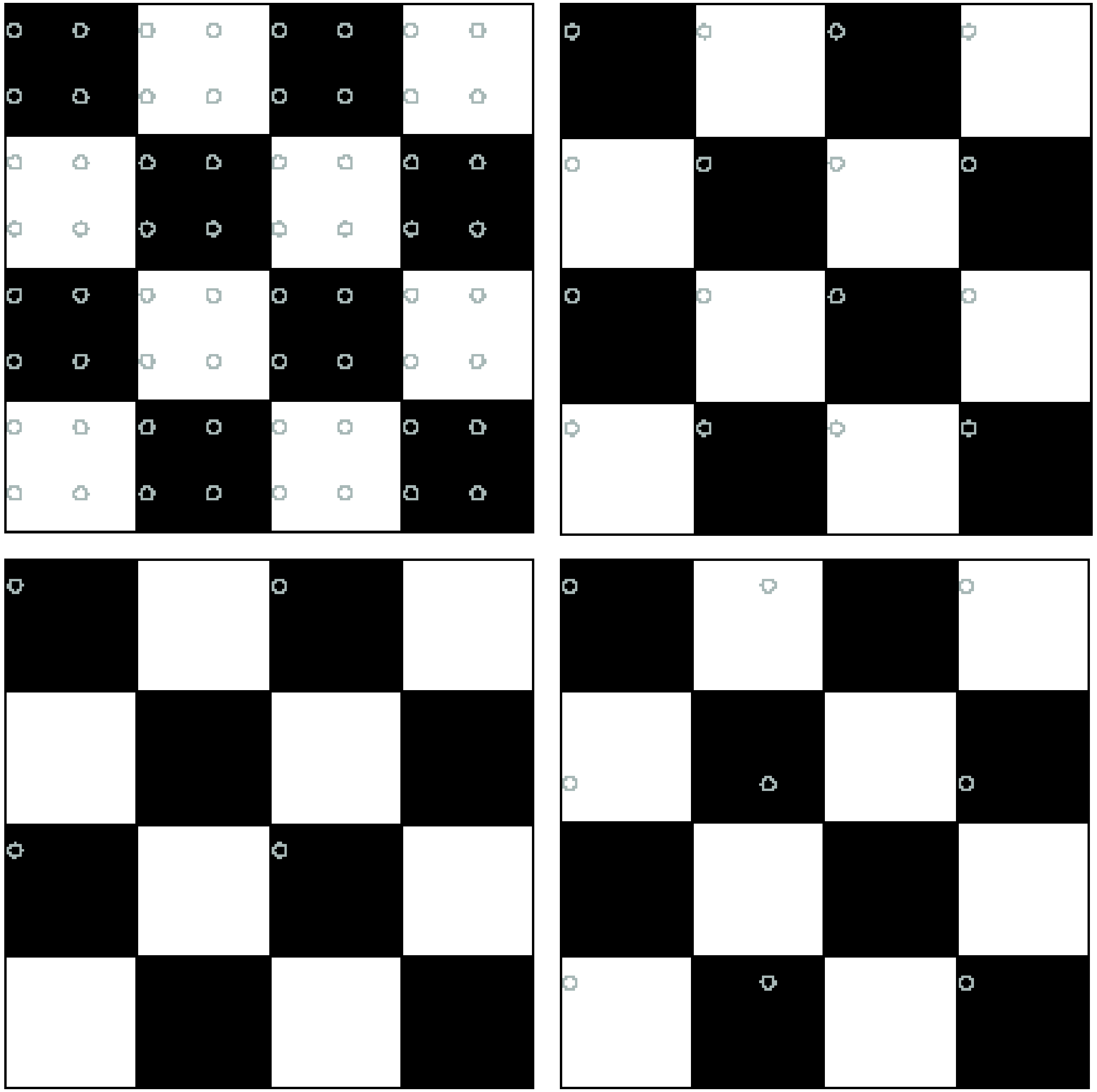
$$= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} F\left(u-i, v-j\right)$$


Signal
Fourier Transform
Magnitude Spectrum
Sample
Copy and Shift
Sampled Signal
Fourier Transform
Magnitude Spectrum
Cut out by multiplication with box filter
Accurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum

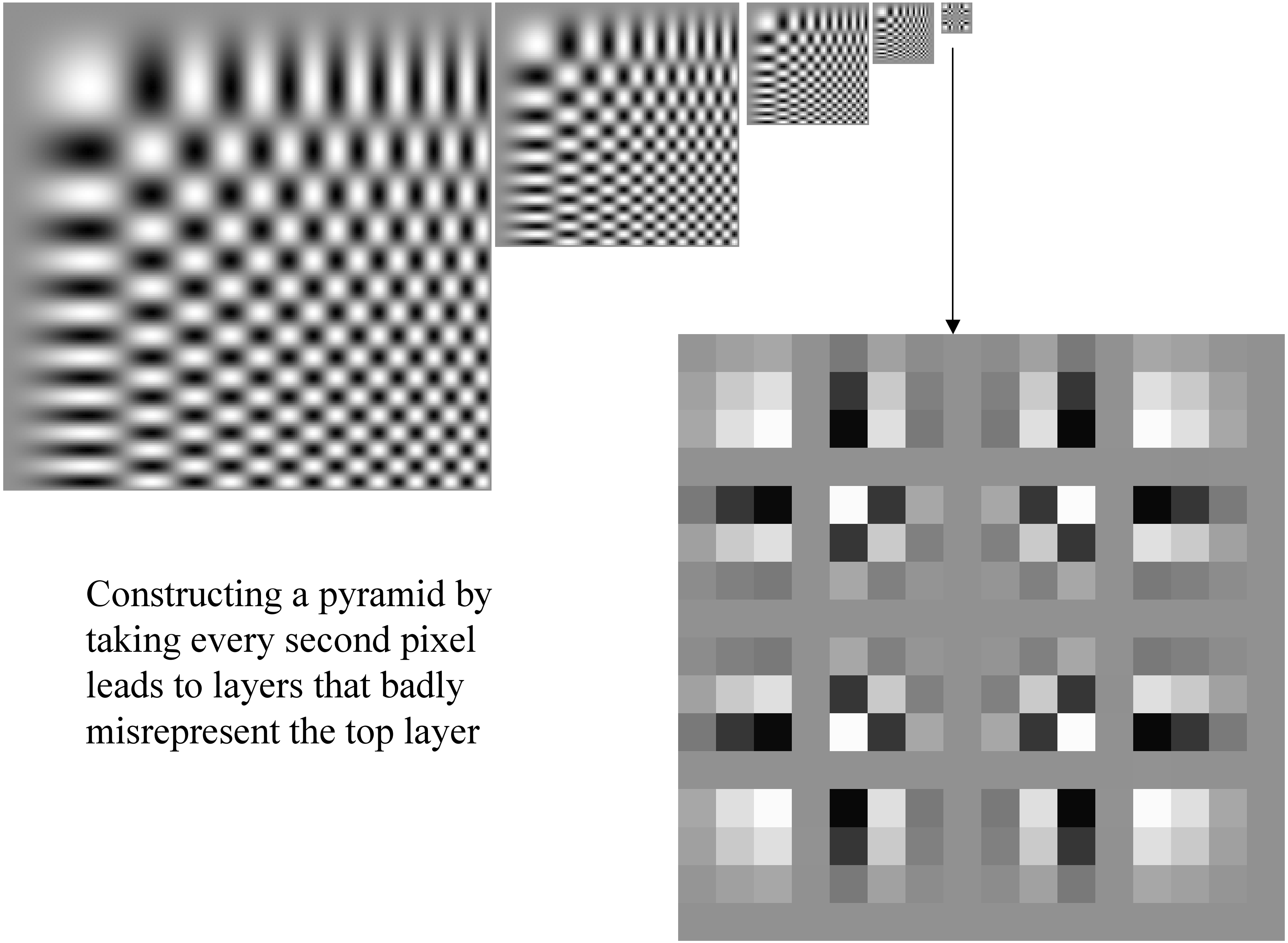

Signal
Fourier Transform
Magnitude Spectrum
Sample
Copy and Shift
Sampled Signal
Fourier Transform
Magnitude Spectrum
Cut out by multiplication with box filter
Inaccurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum

Aliasing

- Can’t shrink an image by taking every second pixel
- If we do, characteristic errors appear
  - In the next few slides
  - Typically, small phenomena look bigger; fast phenomena can look slower
  - Common phenomenon
    - Wagon wheels rolling the wrong way in movies
    - Checkerboards misrepresented in ray tracing
    - Striped shirts look funny on colour television



Resample the checkerboard by taking one sample at each circle. In the case of the top left board, new representation is reasonable. Top right also yields a reasonable representation. Bottom left is all black (dubious) and bottom right has checks that are too big.

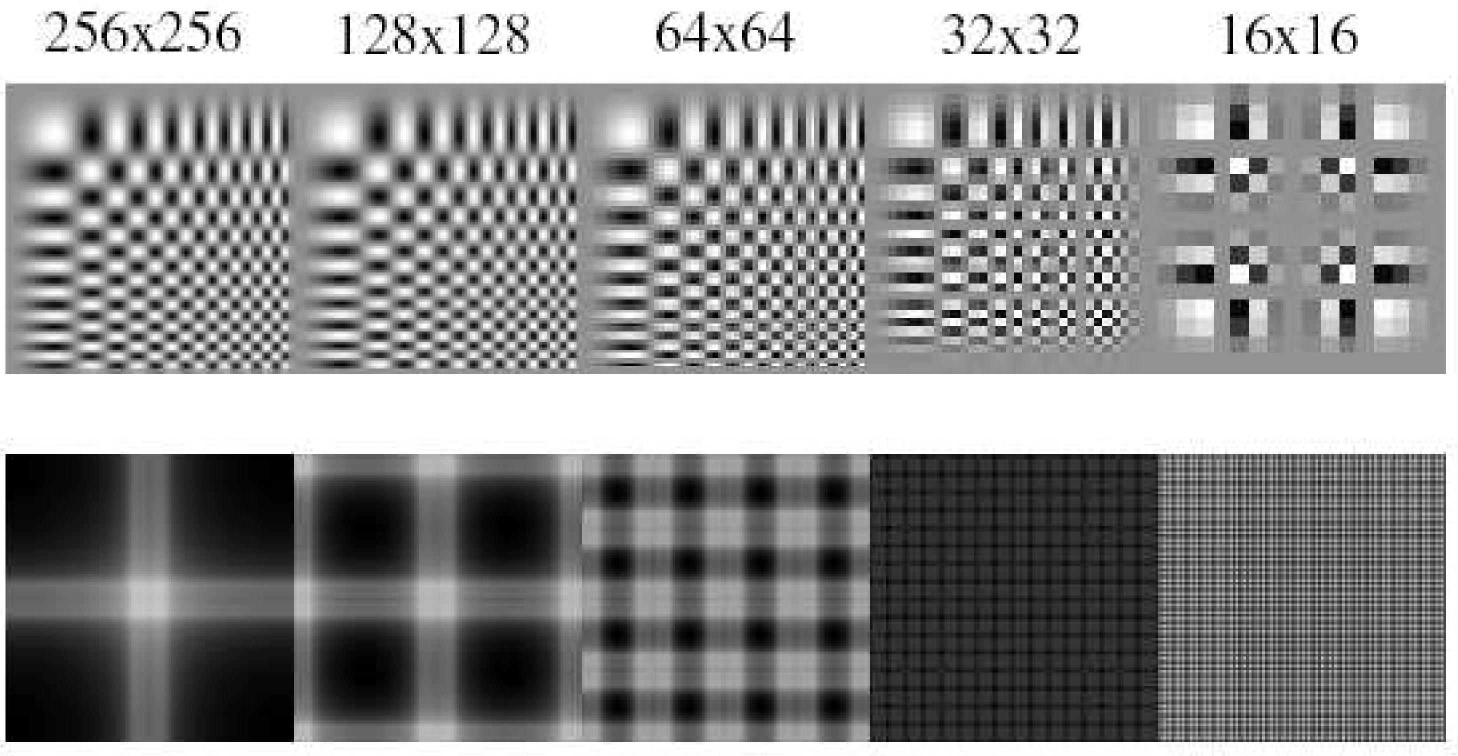


Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer

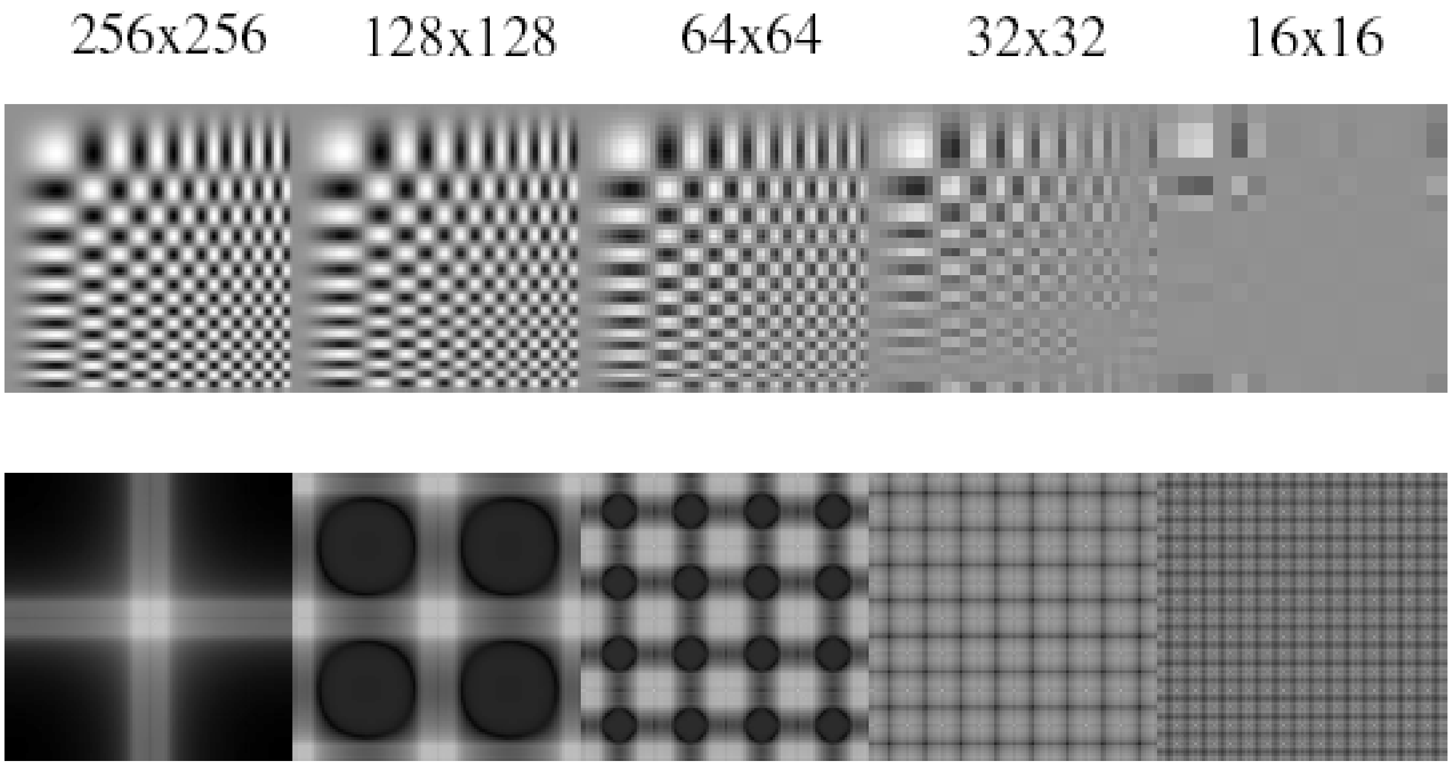
Smoothing as low-pass filtering

- The message of the FT is that high frequencies lead to trouble with sampling.
- Solution: suppress high frequencies before sampling
  - multiply the FT of the signal with something that suppresses high frequencies
  - or convolve with a low-pass filter
- A filter whose FT is a box is bad, because the filter kernel has infinite support
- Common solution: use a Gaussian
  - multiplying FT by Gaussian is equivalent to convolving image with Gaussian.

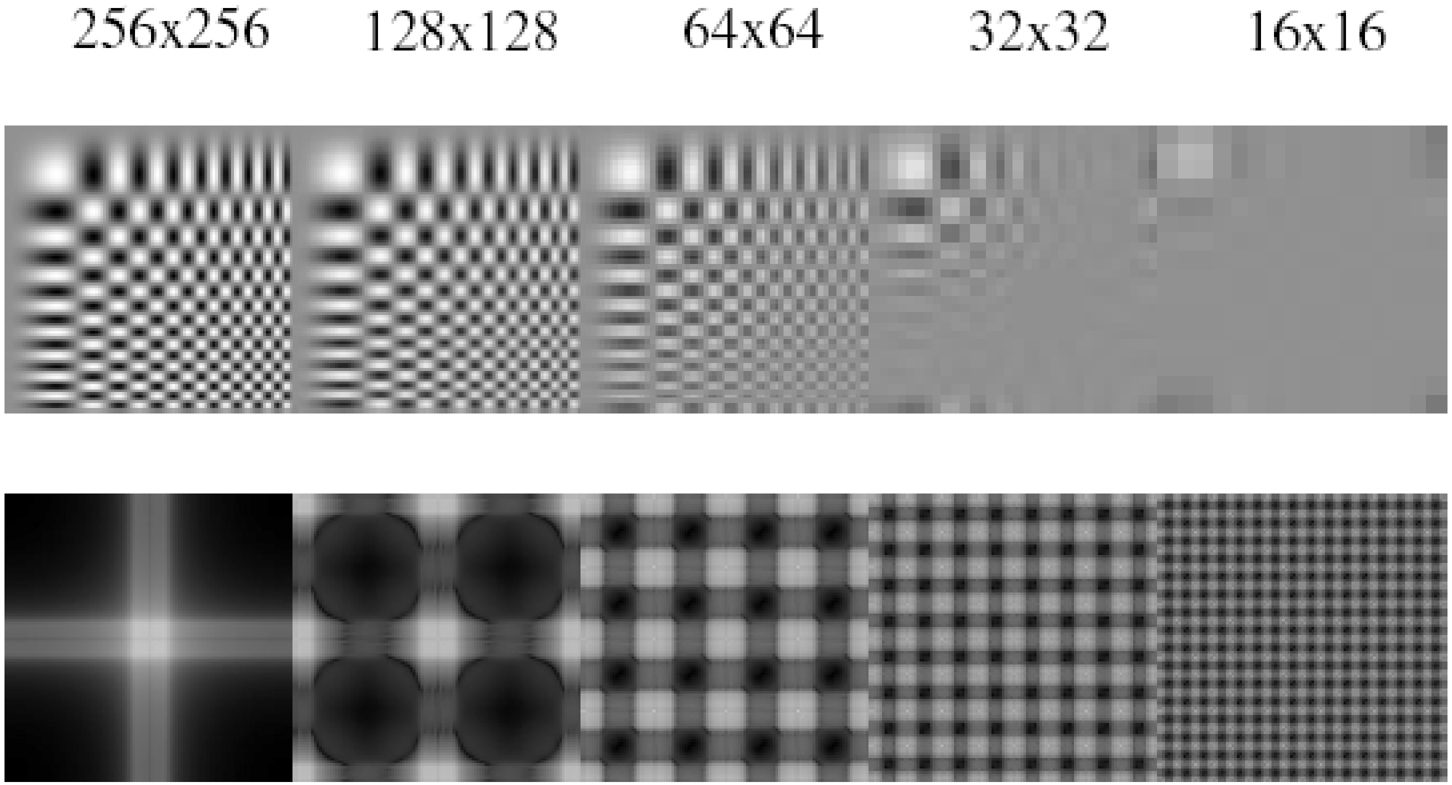
Sampling without smoothing. Top row shows the images, sampled at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.



Sampling with smoothing. Top row shows the images. We
get the next image by smoothing the image with a Gaussian with sigma 1 pixel,
then sampling at every second pixel to get the next; bottom row
shows the magnitude spectrum of these images.

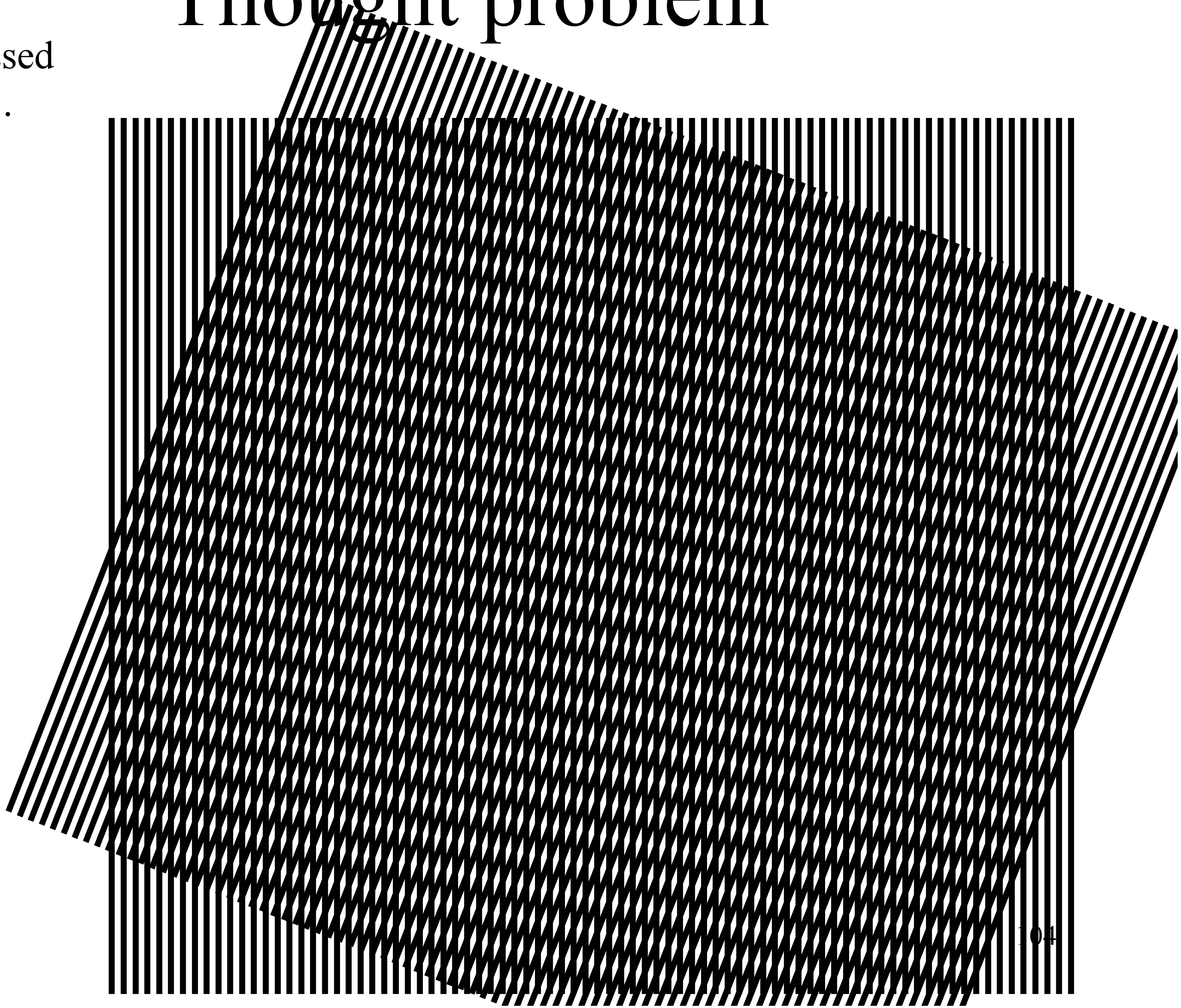


Sampling with smoothing. Top row shows the images. We
get the next image by smoothing the image with a Gaussian with sigma 1.4 pixels,
then sampling at every second pixel to get the next; bottom row
shows the magnitude spectrum of these images.



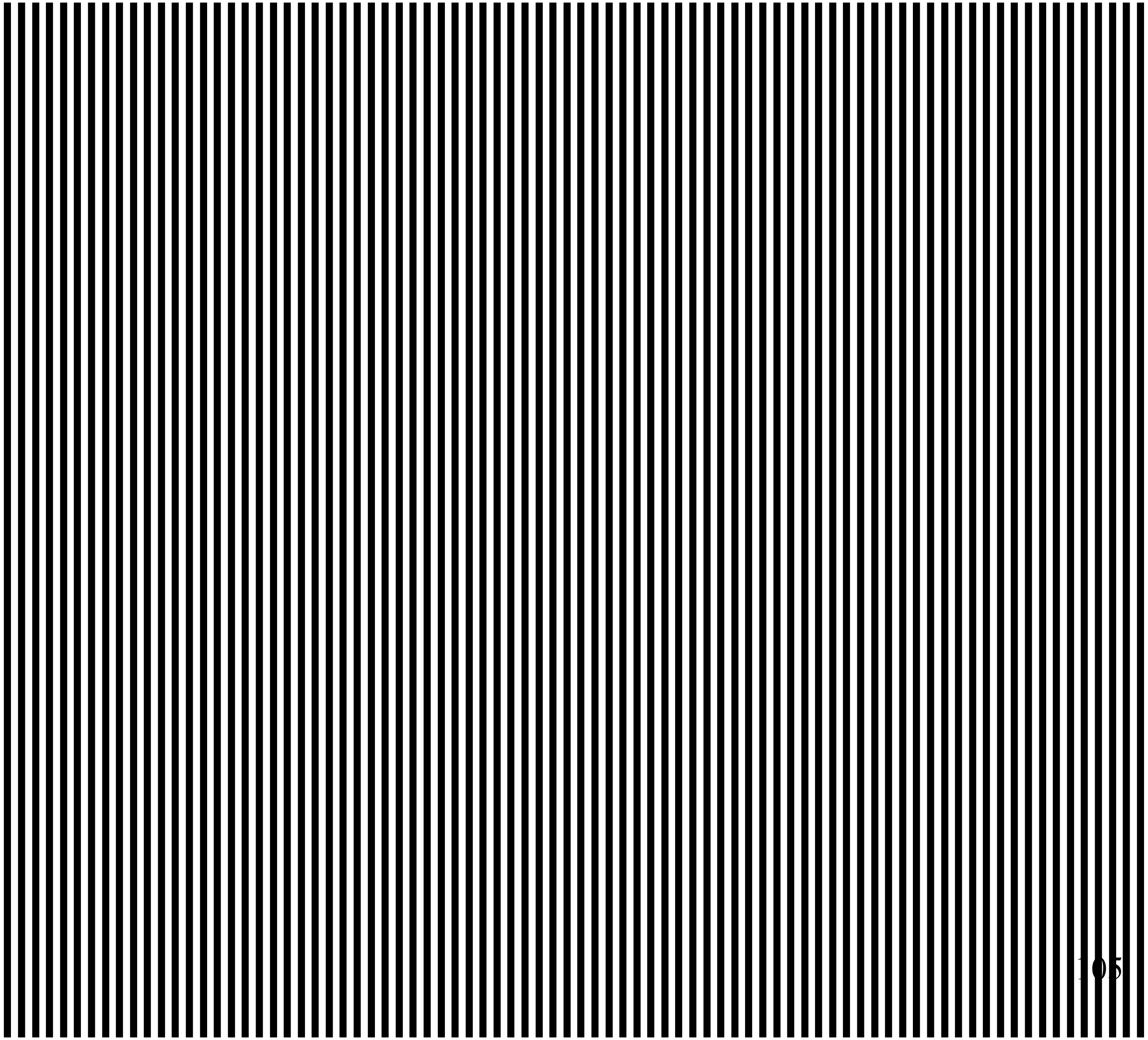
Thought problem

Analyze crossed gratings…



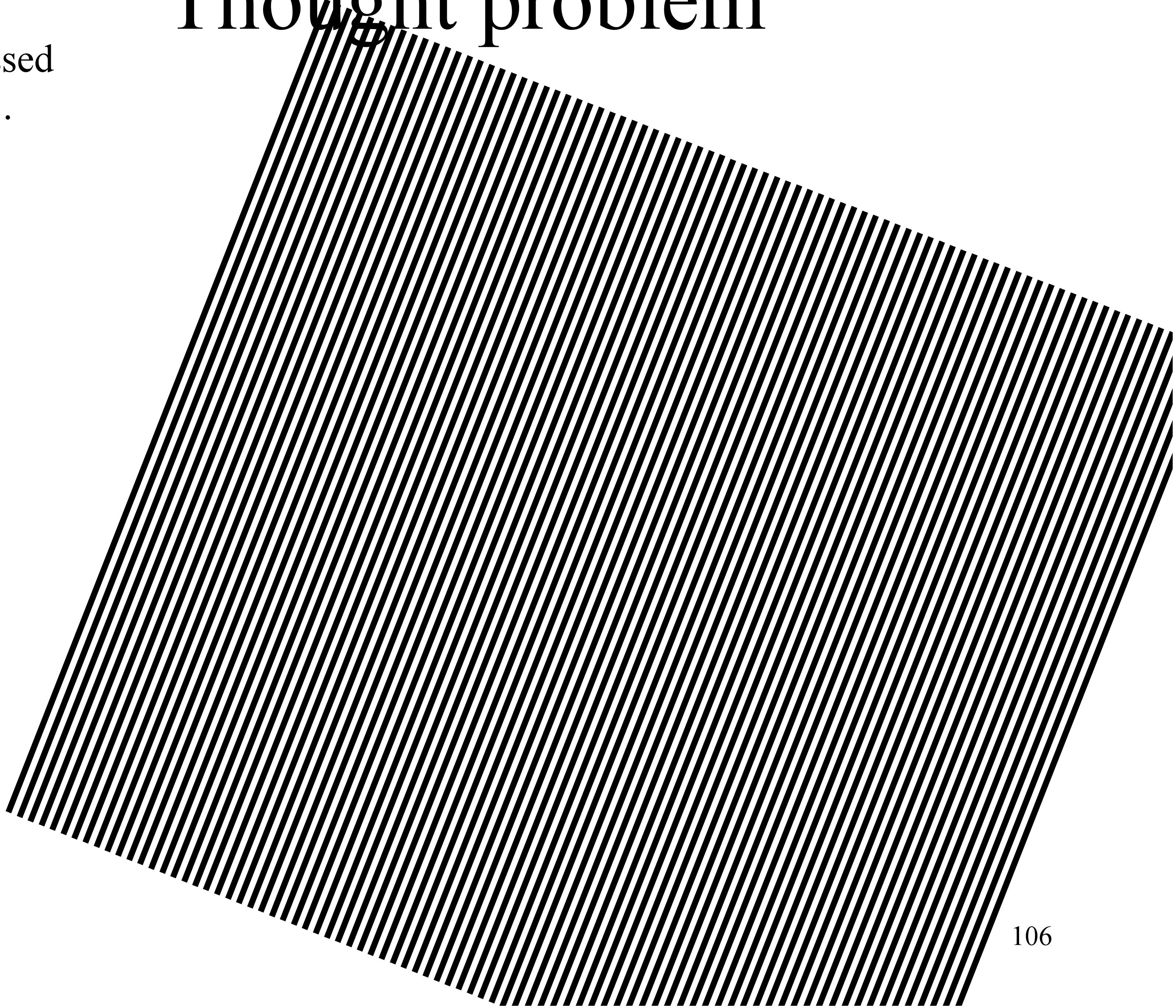
Thought problem

Analyze crossed gratings…



Thought problem

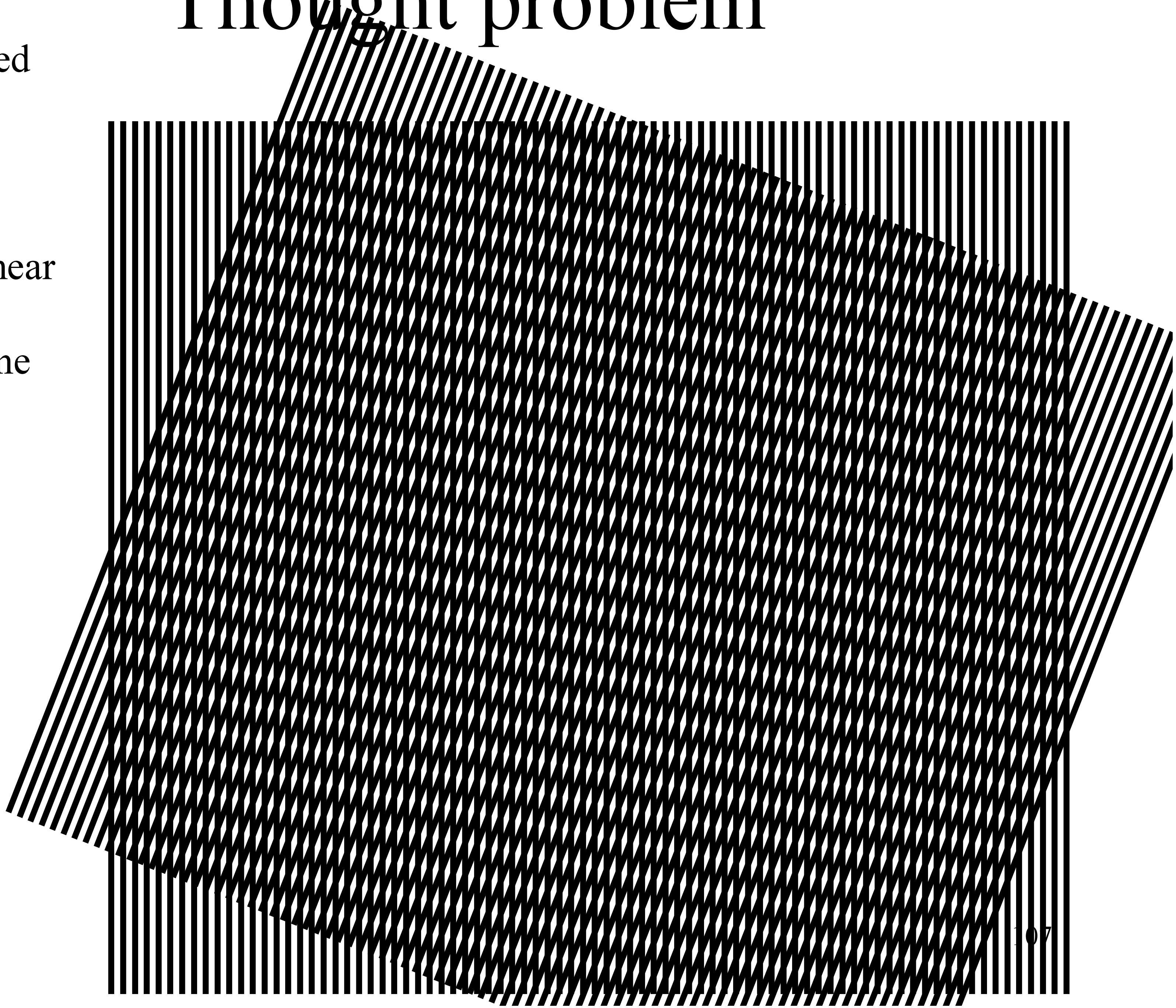
Analyze crossed gratings…



Thought problem

Analyze crossed gratings…

Where does perceived near horizontal grating come from?



What is a good representation for image analysis?

- Fourier transform domain tells you “what” (textural properties), but not “where”.
- Pixel domain representation tells you “where” (pixel location), but not “what”.
- Want an image representation that gives you a local description of image events—what is happening where.