6.801/6.866 Machine Vision

Syllabus

#	Date	Description	Readings	Assignments	Materials
1	9/5	Course Introduction		Pset #0 (not collected)	Freeman Slides Darrell Slides Matlab Tutorial Diary
2	9/10	Cameras, Lenses, and Sensors	Req: FP 1 Opt: H 2.1, 2.3		Freeman Slides
3	9/12	Radiometry and Shading Models I	Req: FP 2, 5.4; H 10 Opt: FP 4	Pset #1 Assigned	Freeman Slides
4	9/17	Radiometry and Shading Models II	"		Freeman Slides
5	9/19	Multiview Geometry	Req: FP 10		Darrell Slides
6	9/24	Stereo	Req: FP 11; H 13	Pset #1 Due	Darrell Slides
7	9/26	Color	Req: FP 6.1-6.4	Pset #2 Assigned	Freeman Slides
8	10/1	Shape from Shading	Req: H 11.1, 11.5-11.9 Opt: H 11.2-11.4		
9	10/3	Filtering I			
10	10/8	Filtering II		Pset #2 Due	
11	10/10	Intro to Bayesian Vision		Exam #1 Assigned	
	10/15	Columbus Day (NO LECTURE)			

Homeworks

- Problem set 1 returned today. Mean: 85. Std dev: 15
- Problem set 1 solutions posted (with password protection) by Tuesday
- Note revisions to problem set 2 matlab assignment
  - To clarify the problems and make them work out cleaner for you.
  - Modifying an existing solution to the revised version will be very simple.

Shape from shading

A shape-from-shading algorithm

$$s_{i,j} = \frac{1}{4}\left(\left(p_{i+1,j} - p_{i,j}\right)^2 + \left(p_{i,j+1} - p_{i,j}\right)^2 + \left(q_{i+1,j} - q_{i,j}\right)^2 + \left(q_{i,j+1} - q_{i,j}\right)^2\right)$$

local surface smoothness

$$r_{ij} = \left(E_{ij} - R_s\left(p_{ij}, q_{ij}\right)\right)^2$$

← local fidelity to image data

$$e = \sum_i \sum_j \left(s_{ij} + \lambda r_{ij}\right)$$

← global function to minimize

$$\frac{\partial e}{\partial p_{kl}} = 2\left(p_{kl} - \bar{p}_{kl}\right) - 2\lambda\left(E_{kl} - R_s\left(p_{kl}, q_{kl}\right)\right)\frac{\partial R_s}{\partial p}$$

$$\frac{\partial e}{\partial q_{kl}} = 2\left(q_{kl} - \bar{q}_{kl}\right) - 2\lambda\left(E_{kl} - R_s\left(p_{kl}, q_{kl}\right)\right)\frac{\partial R_s}{\partial q}$$

← derivative w.r.t. surface parameters

$$p_{kl} = \bar{p}_{kl} + \lambda\left(E_{kl} - R_s\left(p_{kl}, q_{kl}\right)\right)\frac{\partial R_s}{\partial p}$$

$$q_{kl} = \bar{q}_{kl} + \lambda\left(E_{kl} - R_s\left(p_{kl}, q_{kl}\right)\right)\frac{\partial R_s}{\partial q}$$

← set derivatives equal to zero.

Iterative algorithm

$$p_{kl}^{n+1} = \bar{p}_{kl}^{n} + \lambda\left(E_{kl} - R_s\left(p_{kl}^{n}, q_{kl}^{n}\right)\right)\frac{\partial R_s}{\partial p}$$

$$q_{kl}^{n+1} = \bar{q}_{kl}^{n} + \lambda\left(E_{kl} - R_s\left(p_{kl}^{n}, q_{kl}^{n}\right)\right)\frac{\partial R_s}{\partial q}$$

Results using this approach

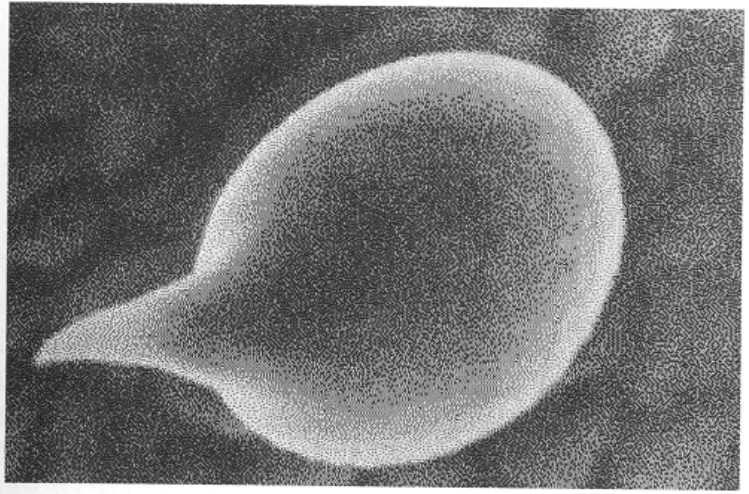


Figure 11-10. Display of the image of a small resin droplet on a flower of a *Cannabis sativa* plant. (Reproduced by permission from the book *Magnifications—Photography with the Scanning Electron Microscope* by David Scharf, Schocken Books, New York, 1977.)

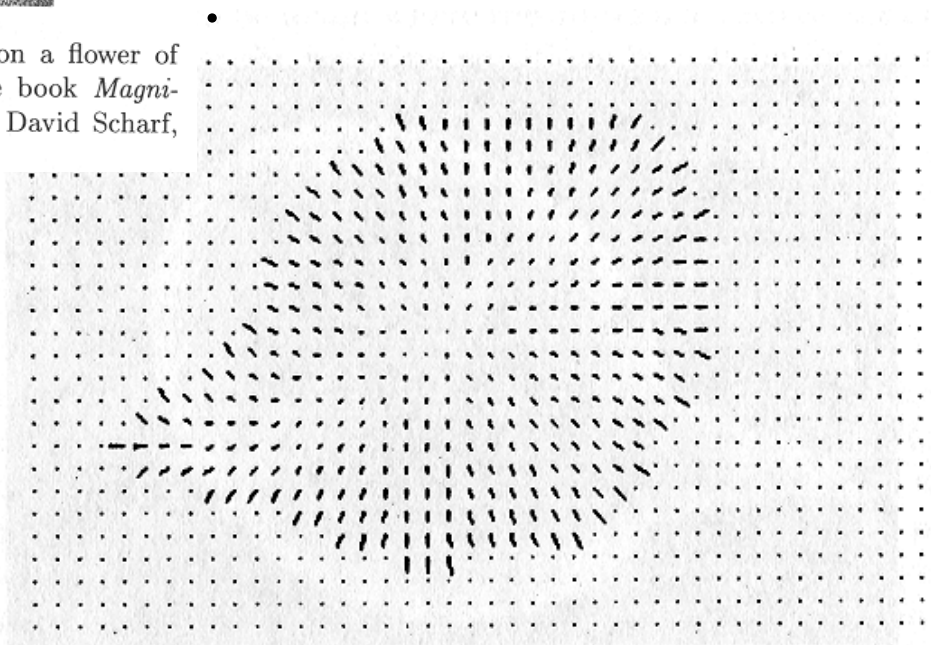


Figure 11-11. Needle diagram calculated by the iterative scheme under the assumption that the reflectance map is $\sec\theta_i$. (The surface orientation data are actually available on a finer grid; they are sampled coarsely here for display purposes.) The needle diagram is the estimate of the shape of the surface of the resin droplet shown in the previous figure. (Figure kindly provided by Katsushi Ikeuchi.)

Horn, 1986

2-d animation

3-d animation

Image filtering

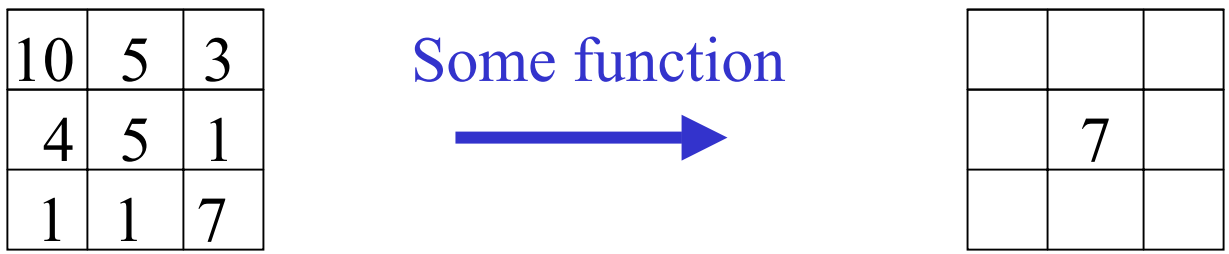
- Reading:
  - Chapter 7, F&P
- Recommended Reading:
  - Chapter 7, 8 Horn

Take 6.341, discrete-time signal processing

- If you want to process pixels, you need to understand signal processing well, so
  - Take 6.341
- Fantastic set of teachers:
  - Al Oppenheim
  - Greg Wornell
  - Jae Lim
- Web page: http://web.mit.edu/6.341/www/

What is image filtering?

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



Local image data

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	

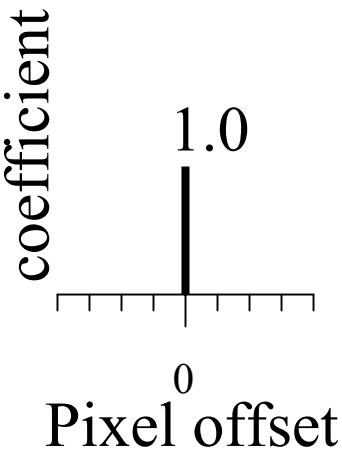
Modified image data

Convolution

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

Linear filtering (warm-up slide)

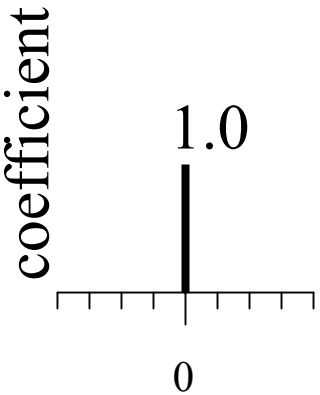
original

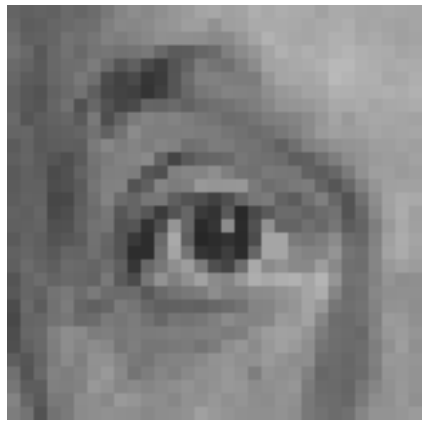


?

Linear filtering (warm-up slide)

original

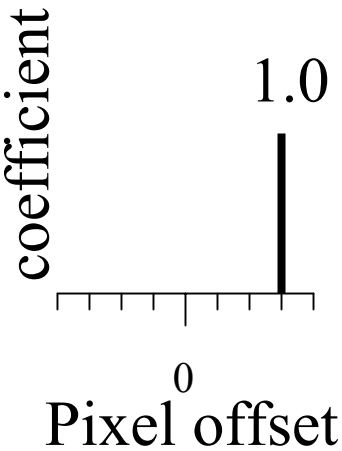




Filtered
(no change)

Linear filtering

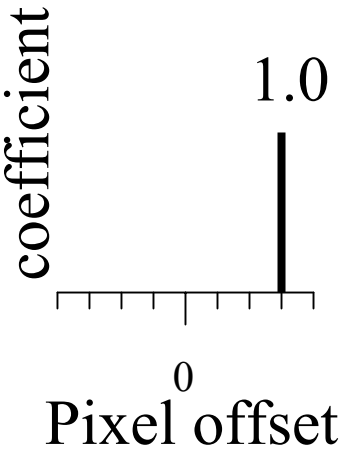
original

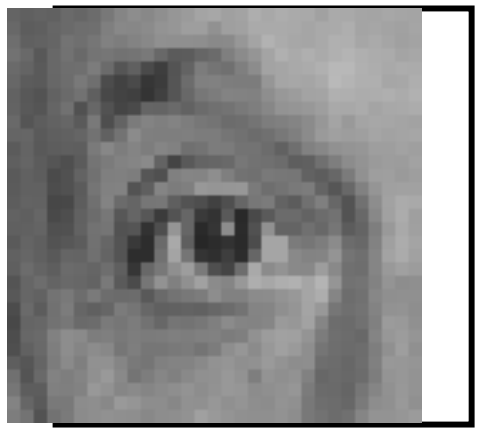


?

shift

original

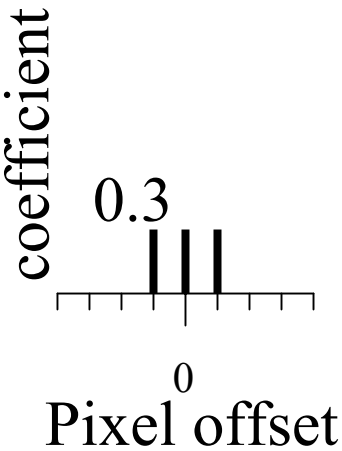




shifted

Linear filtering

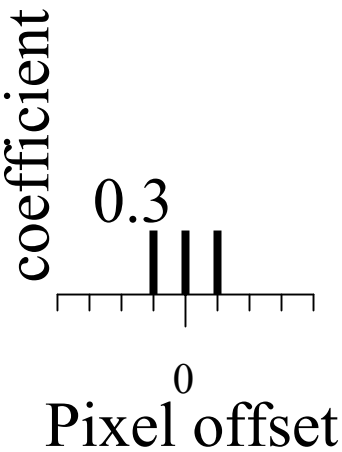
original



?

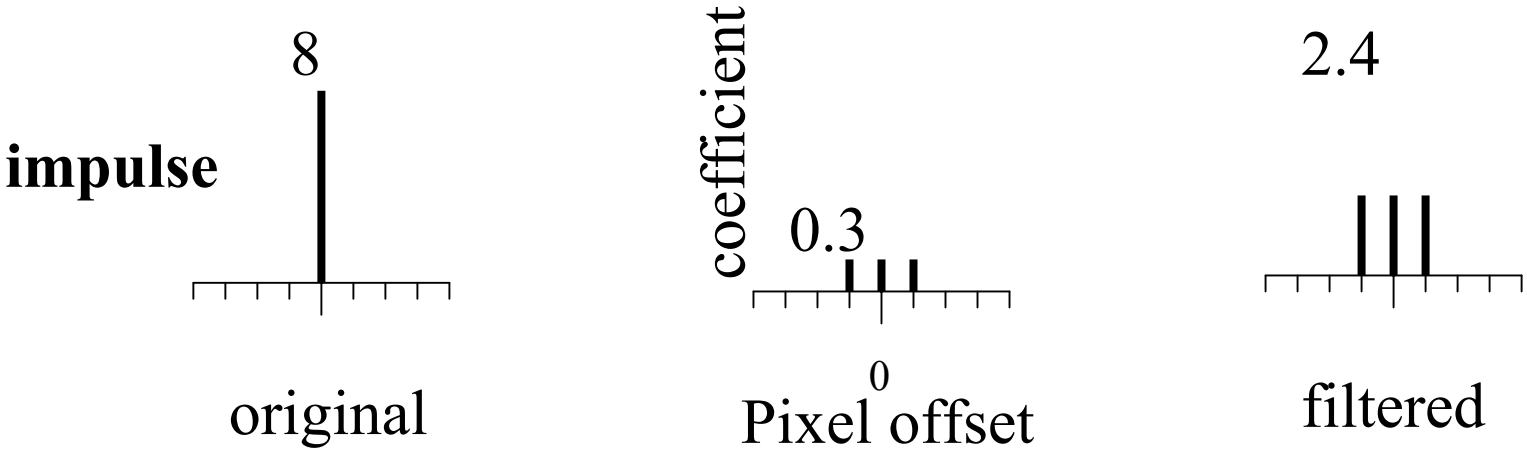
Blurring

original

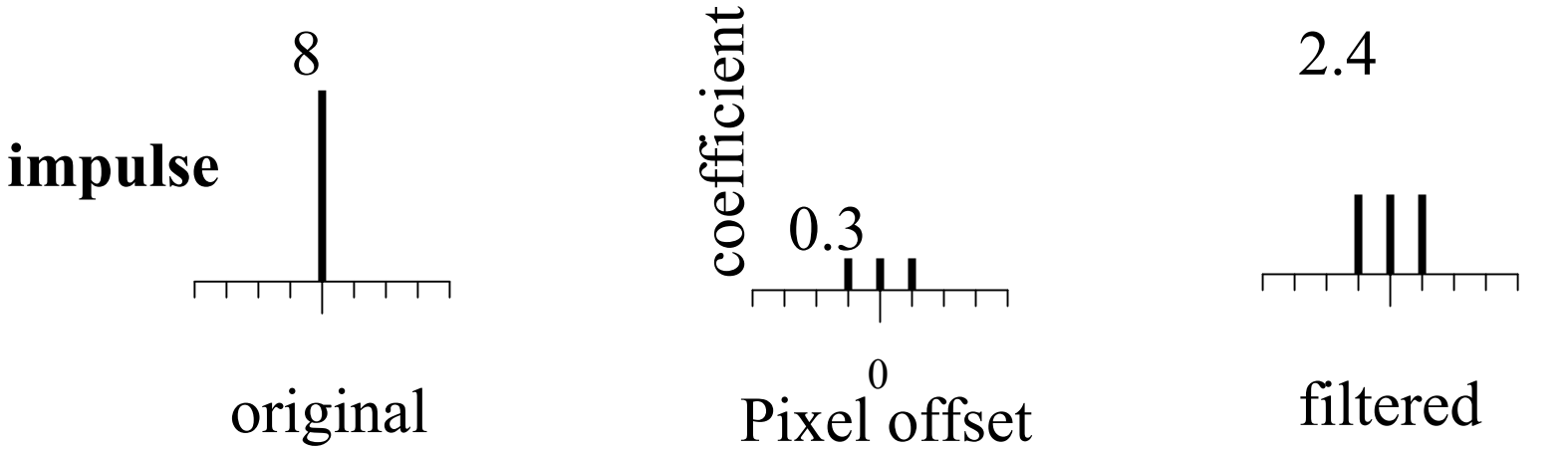


Blurred (filter applied in both dimensions).

Blur examples

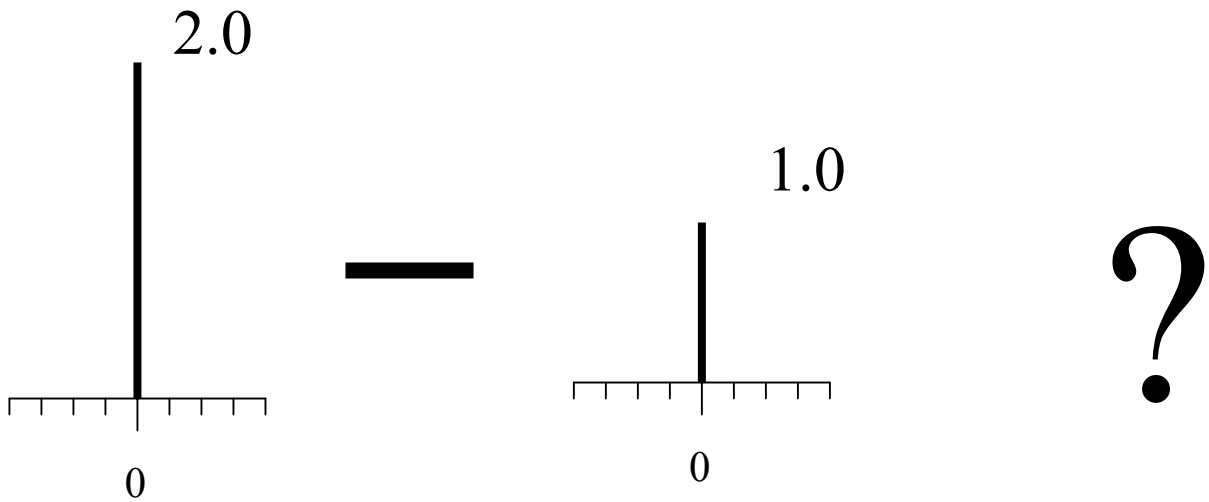


Blur examples

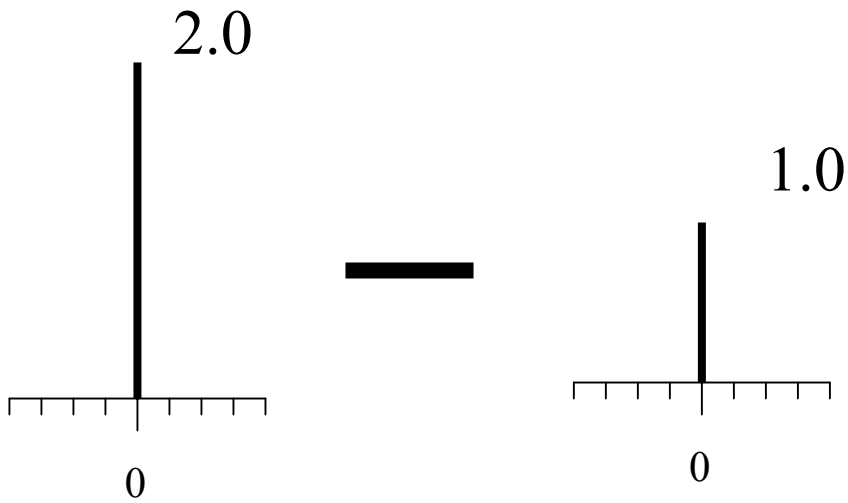


Linear filtering (warm-up slide)

original



Linear filtering (no change)

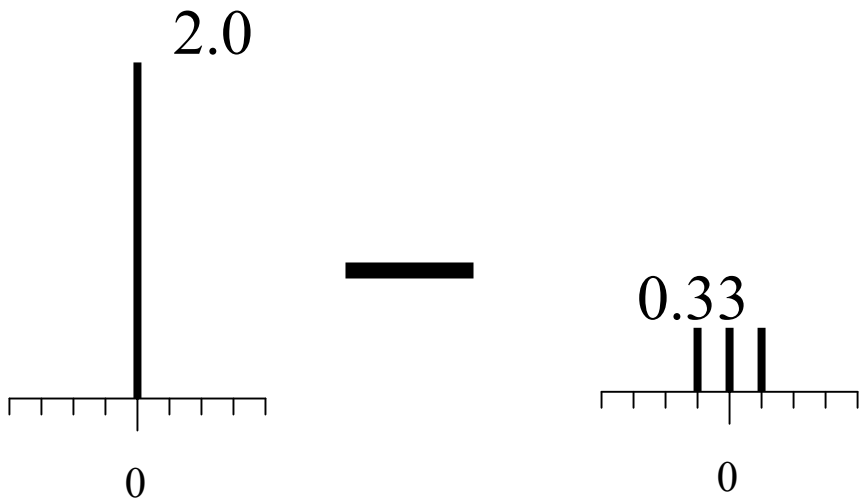


original

Filtered
(no change)

Linear filtering

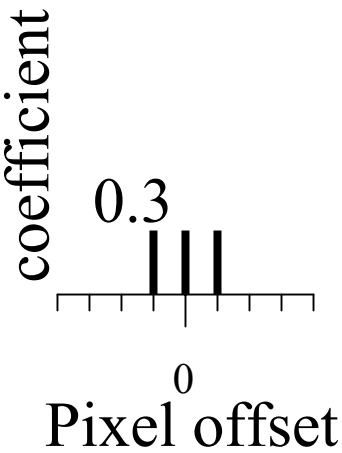
original



?

(remember blurring)

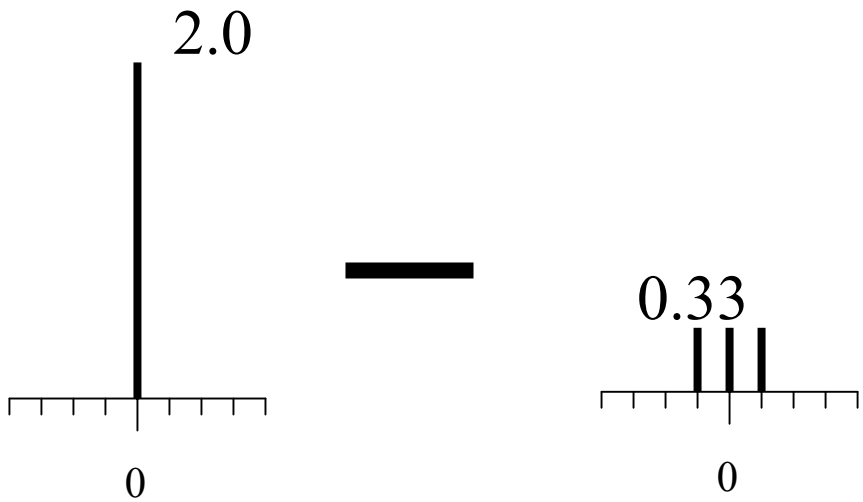
original

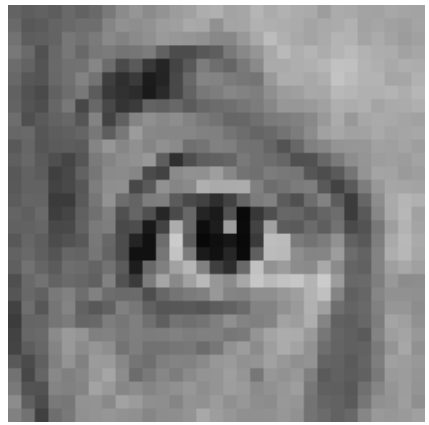


Blurred (filter applied in both dimensions).

Sharpening

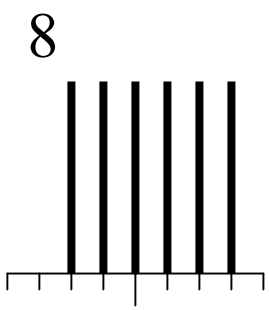
original



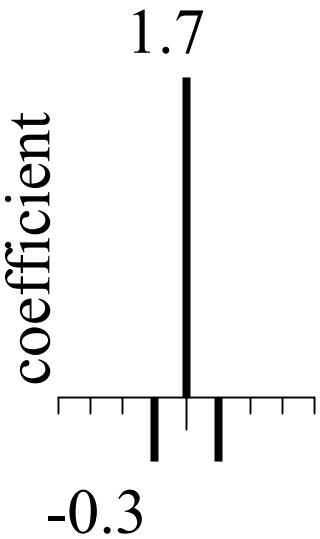


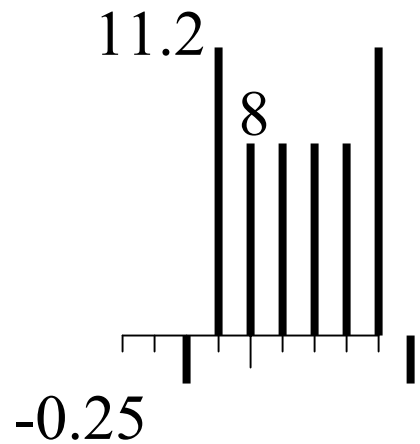
Sharpened original

Sharpening example



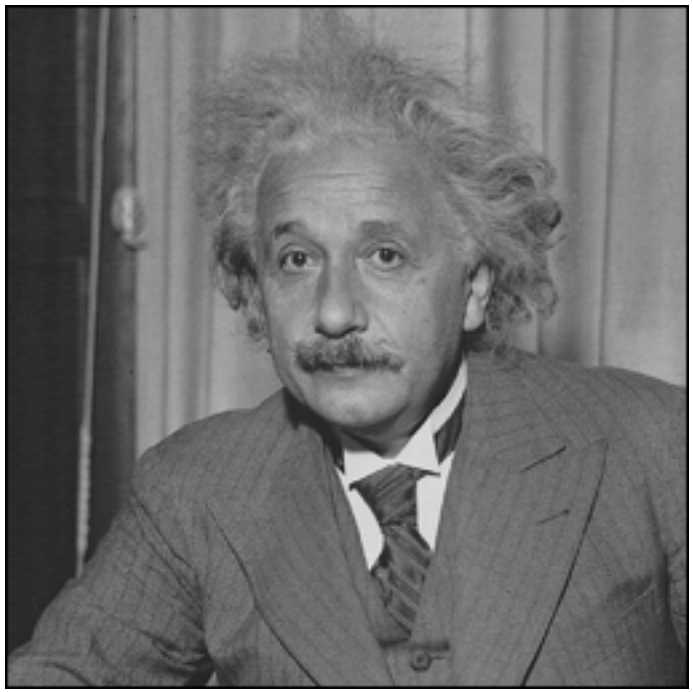
original



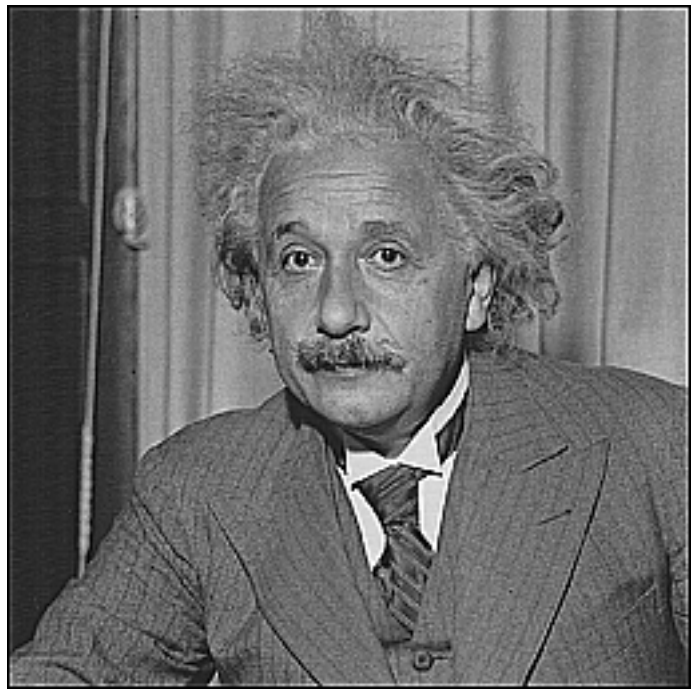


Sharpened (differences are accentuated; constant areas are left untouched).

Sharpening

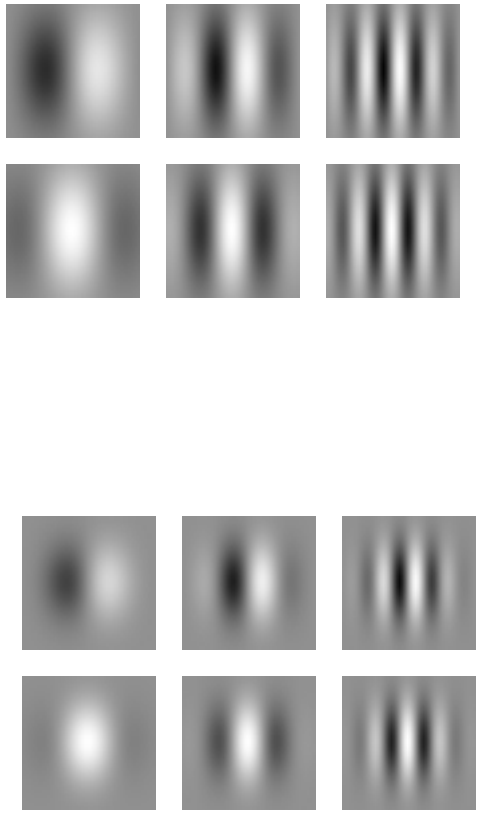


before



after

Oriented filters

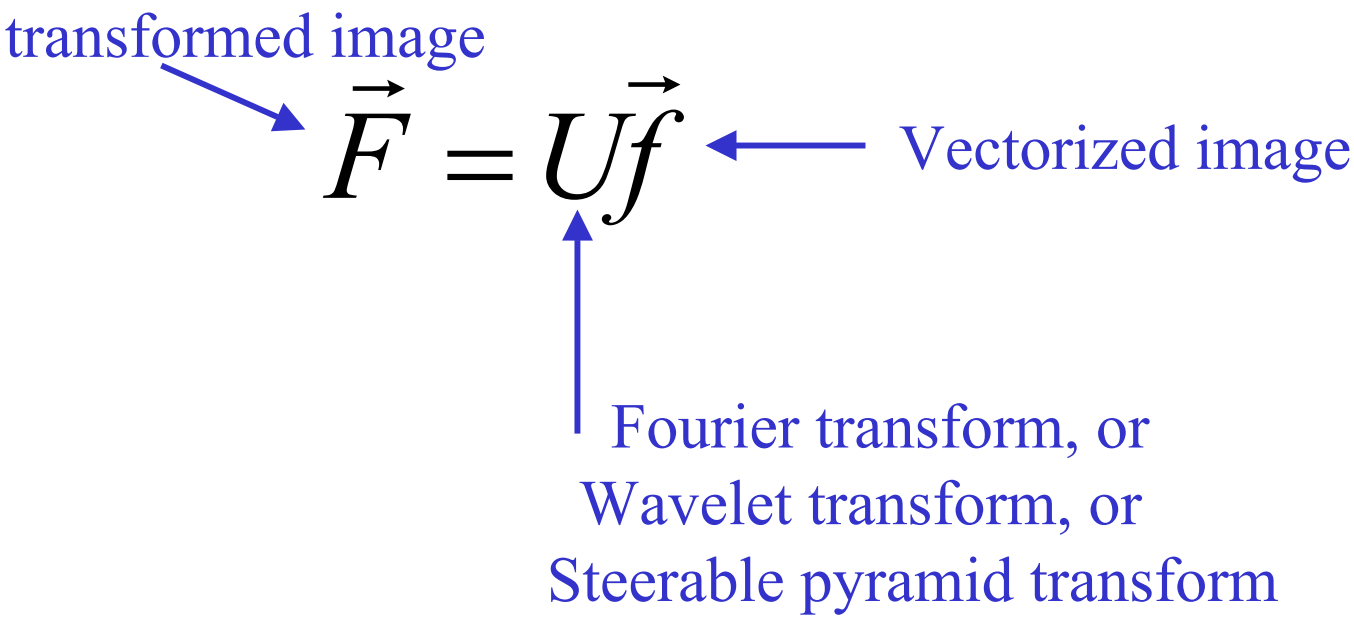


Gabor filters at different scales and spatial frequencies

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.



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

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

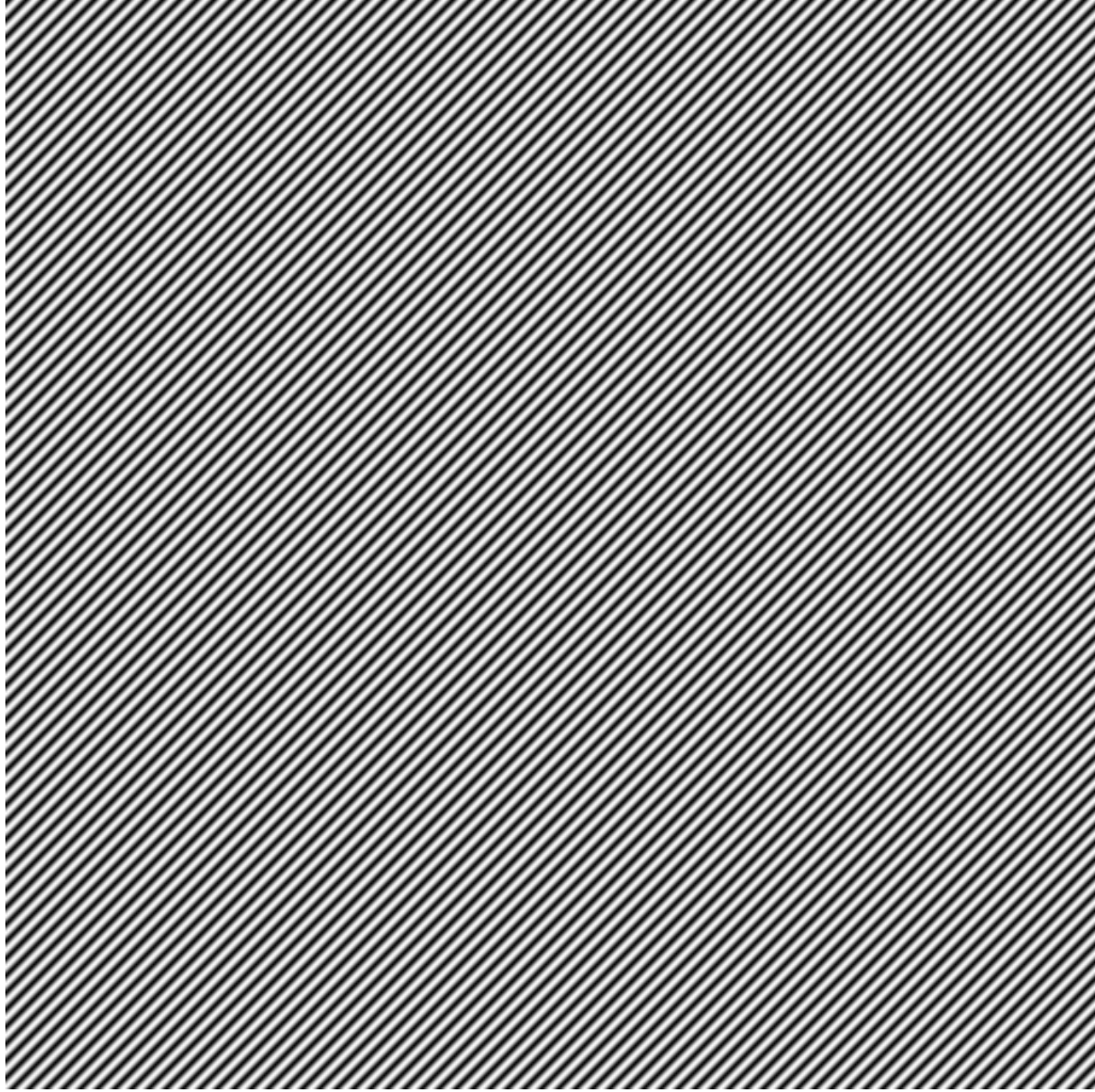
Inverse transform

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

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...

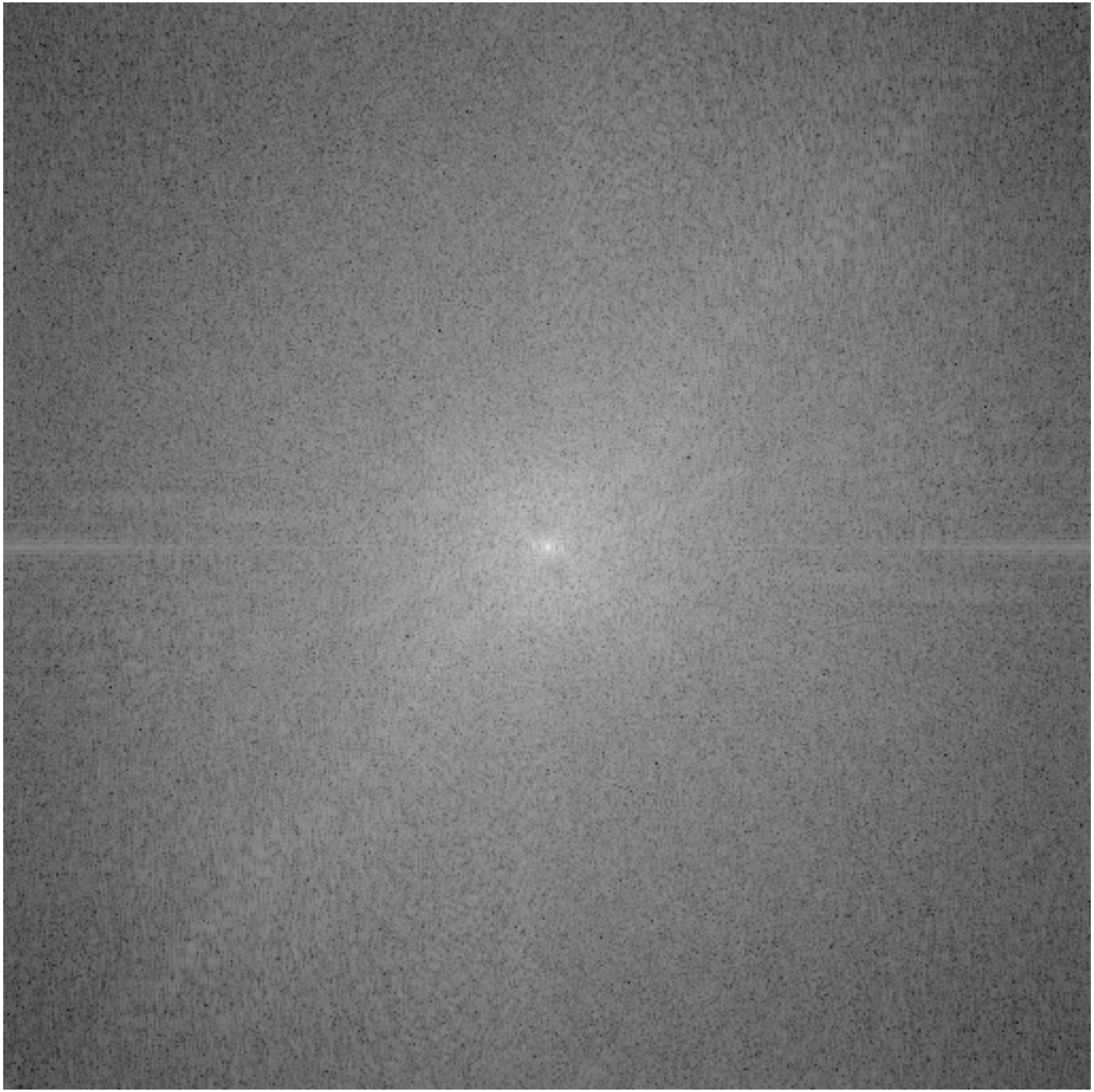


Phase and Magnitude

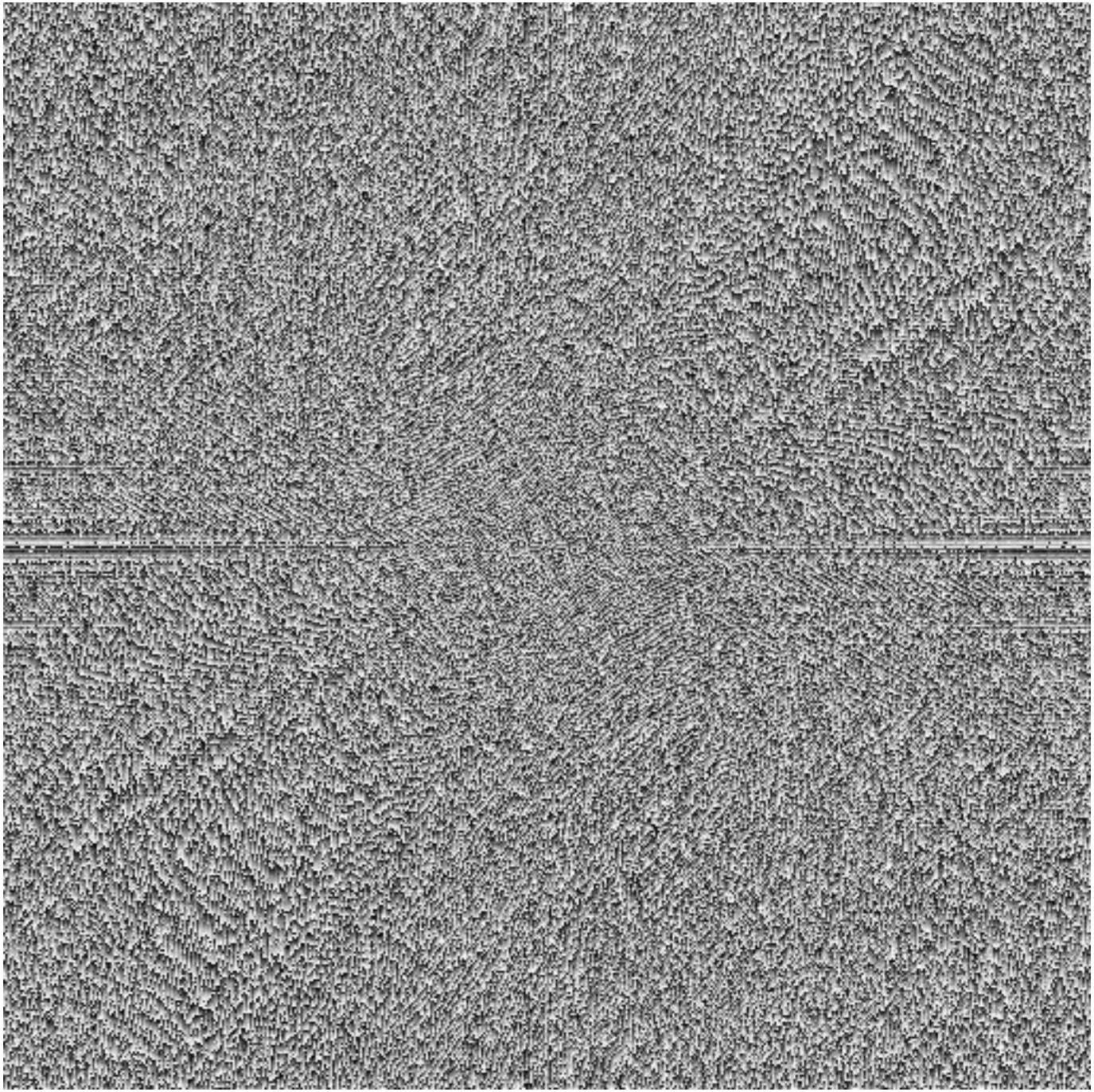
- Fourier transform of a real function is complex
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
- Magnitude is the magnitude of the complex transform

- 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?

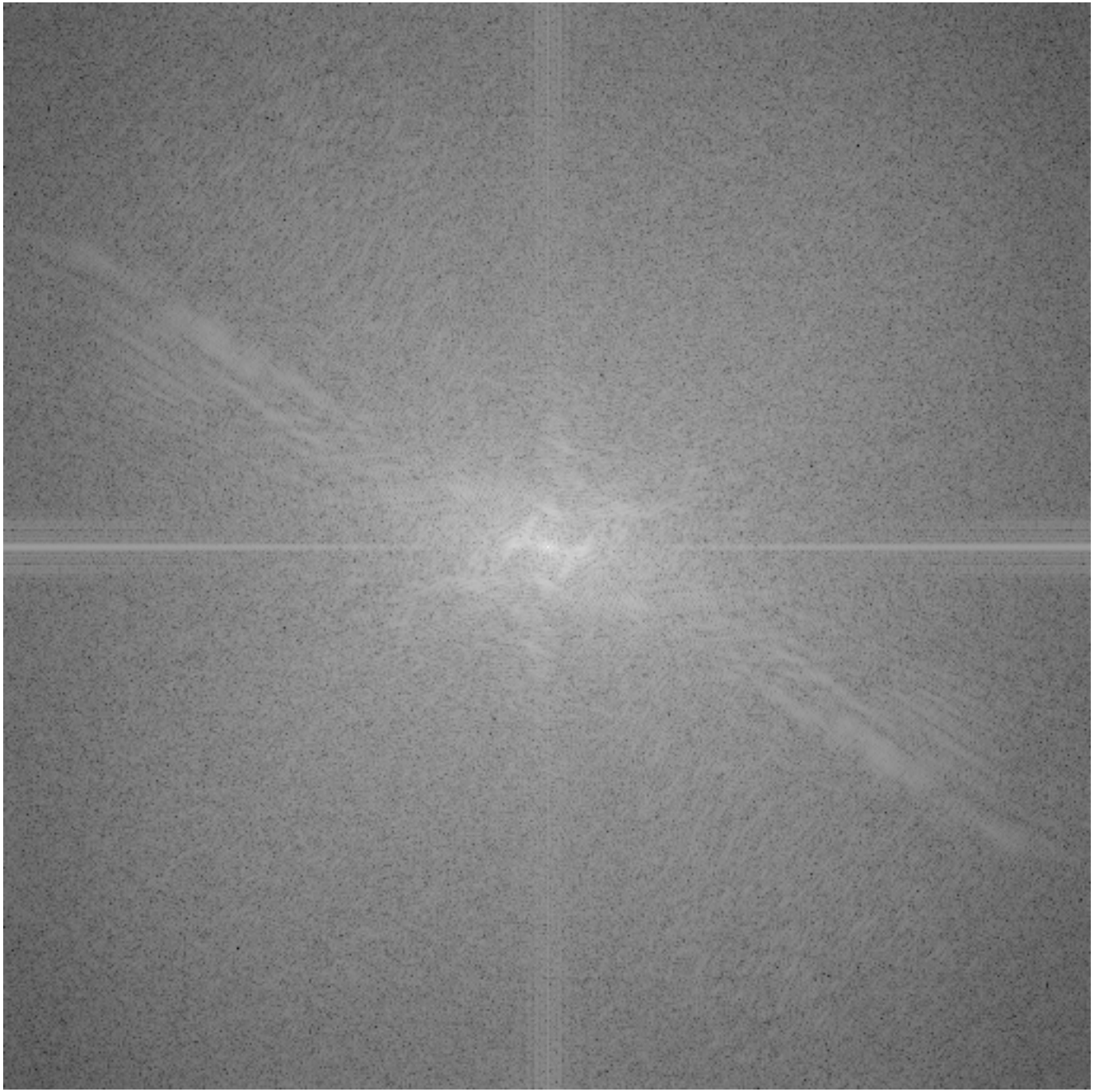
This is the magnitude transform of the cheetah pic



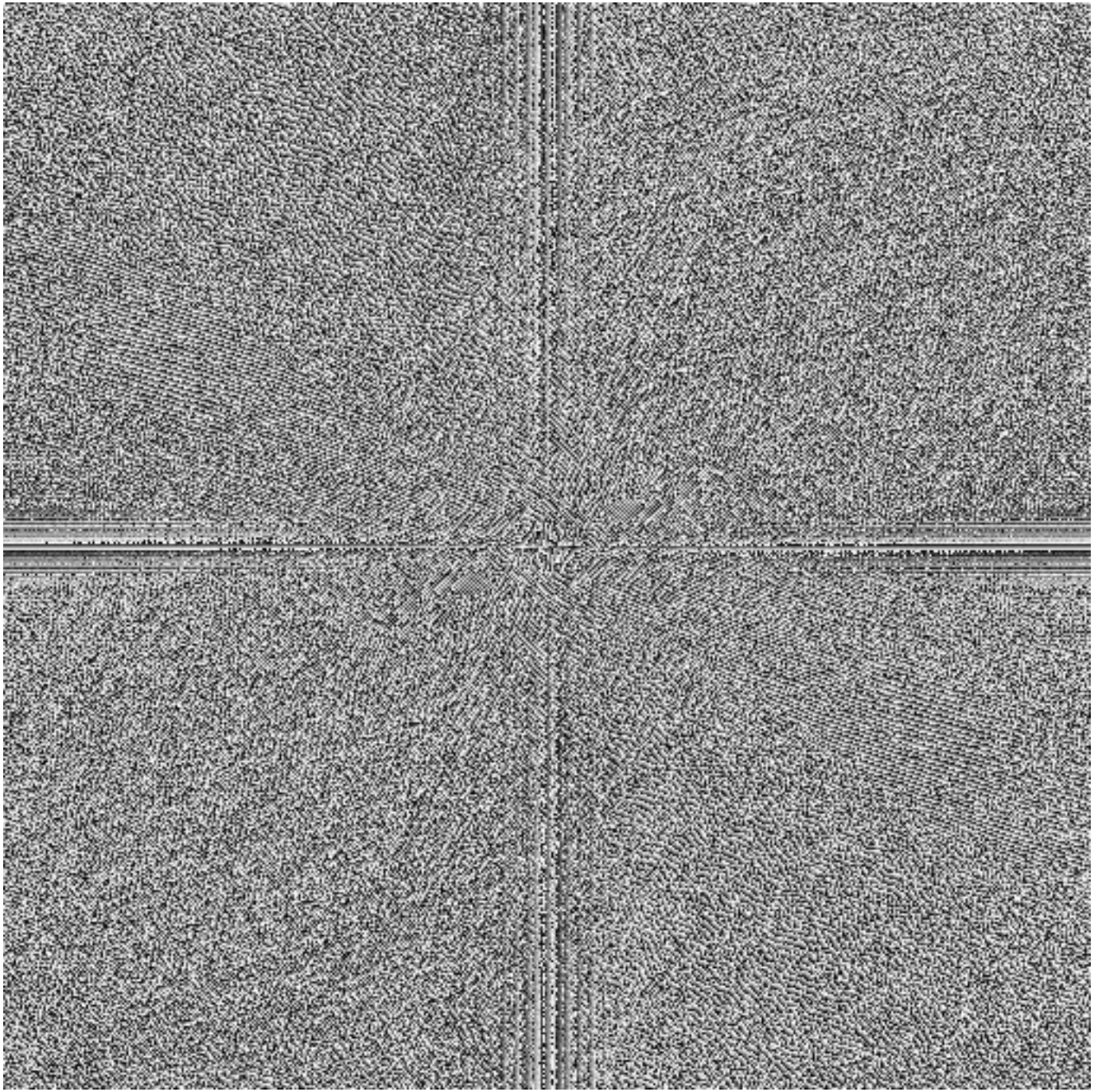
This is the phase transform of the cheetah pic



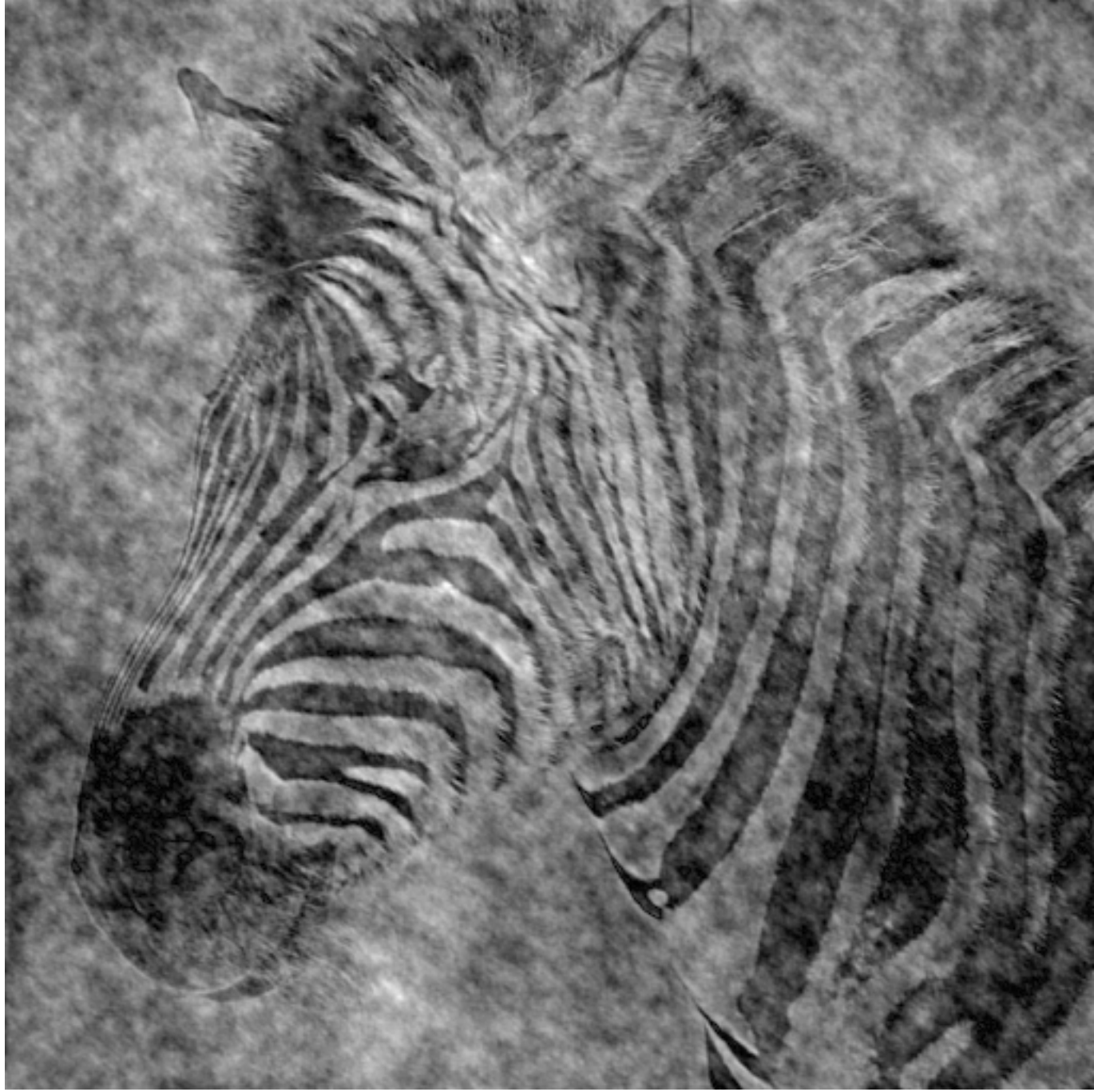
This is the magnitude transform of the zebra pic



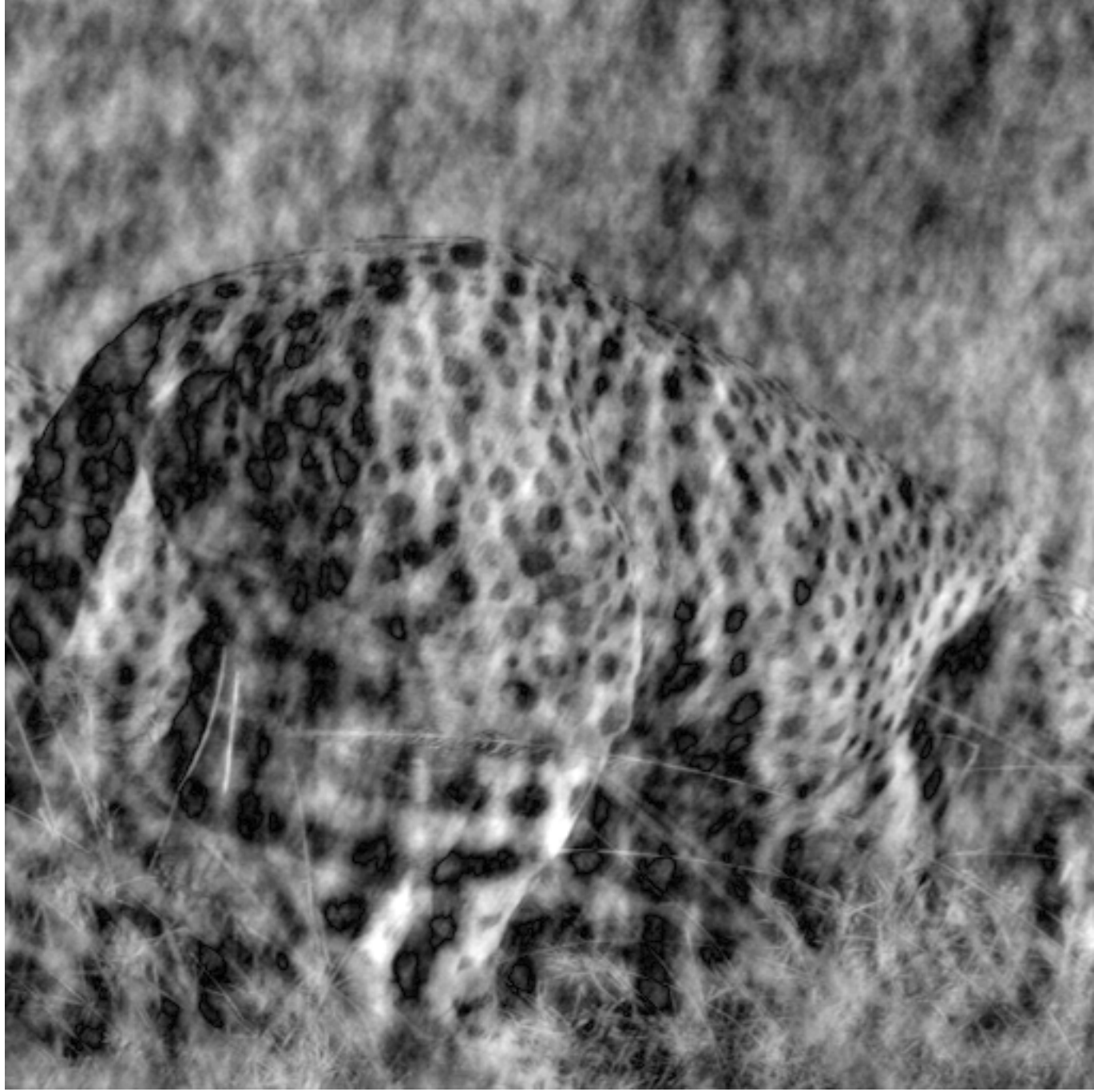
This is the phase transform of the zebra pic



Reconstruction with zebra phase, cheetah magnitude



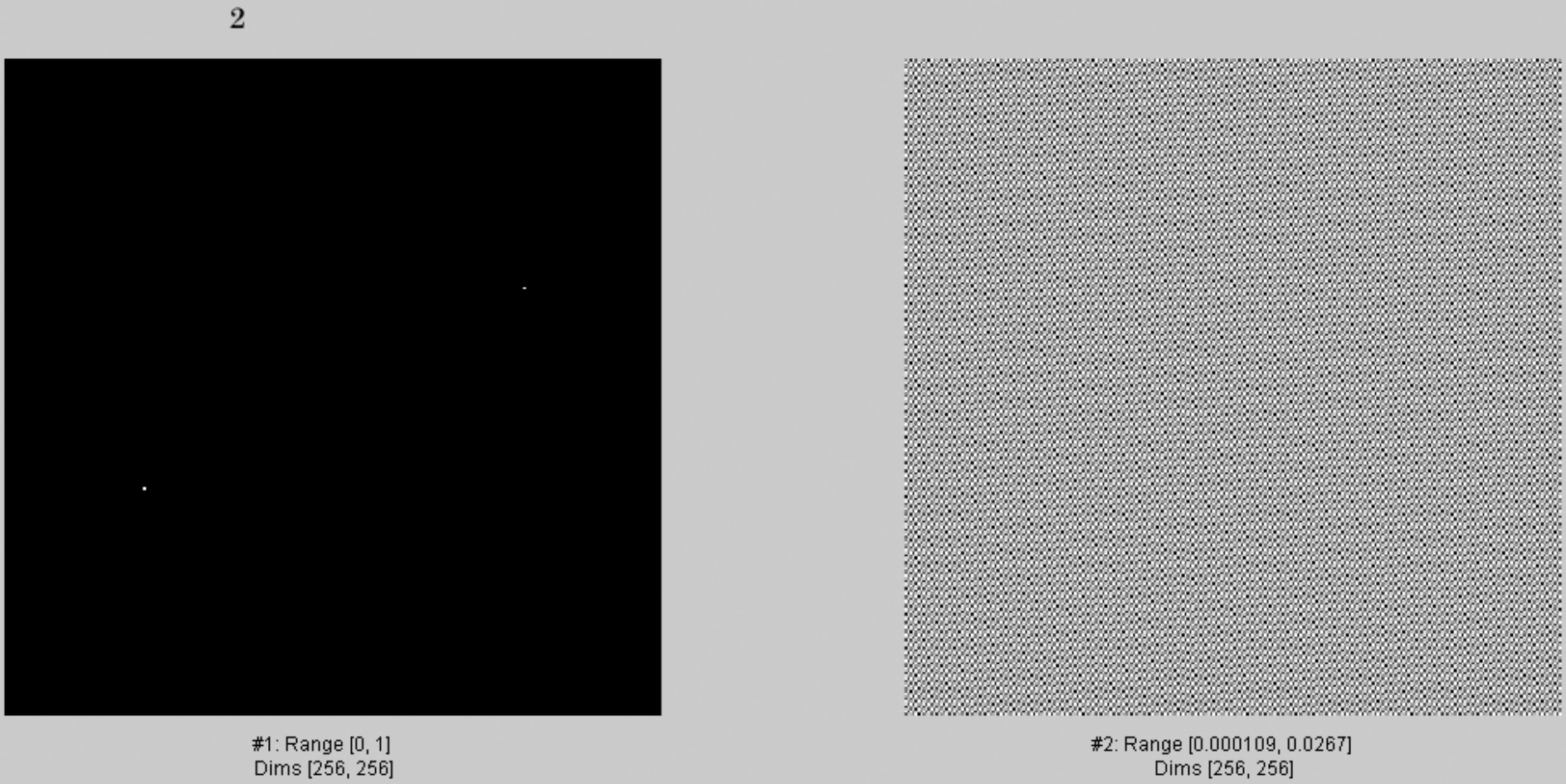
Reconstruction with cheetah phase, zebra magnitude



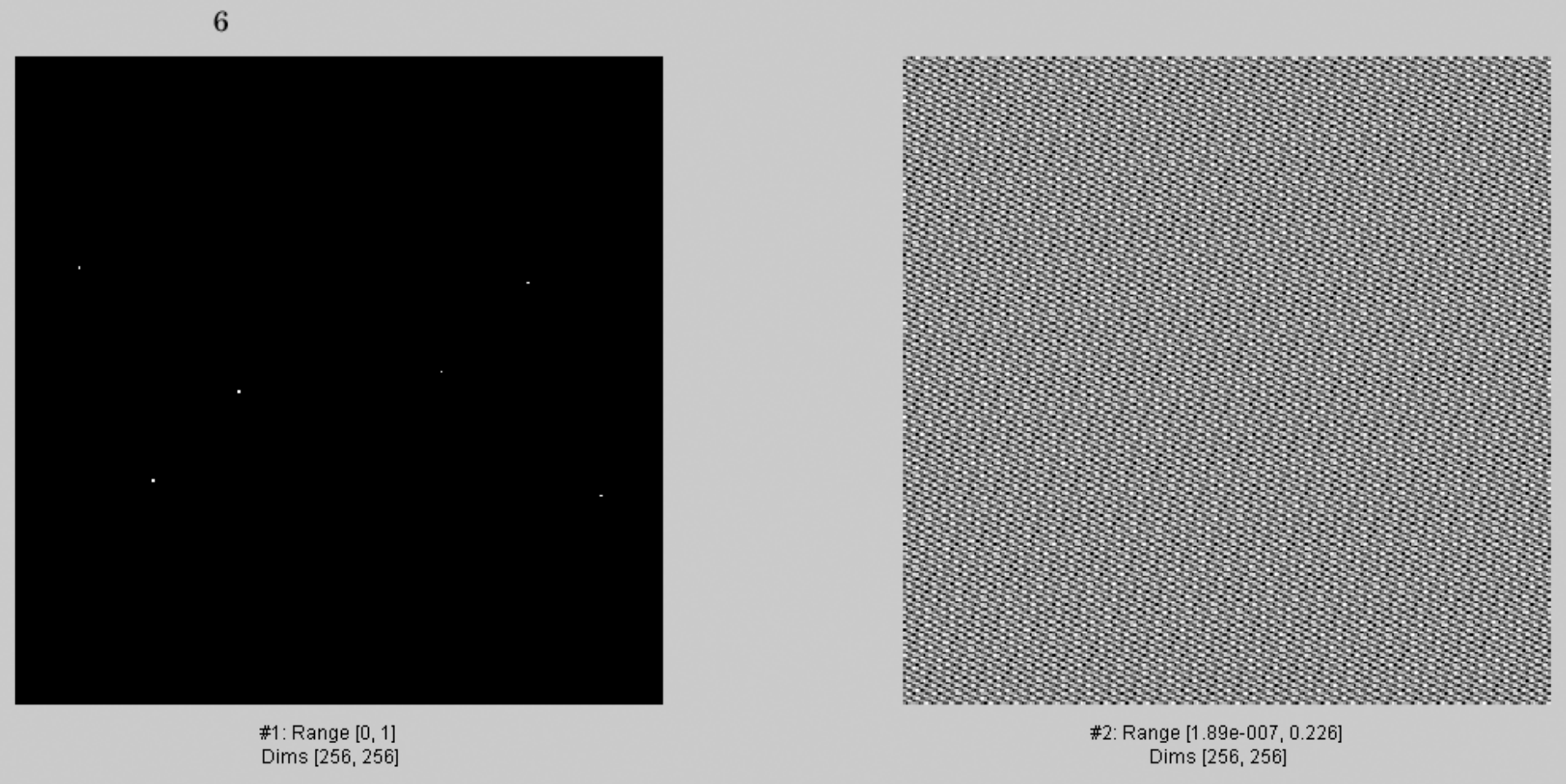
Example image synthesis with fourier basis.

- 16 images

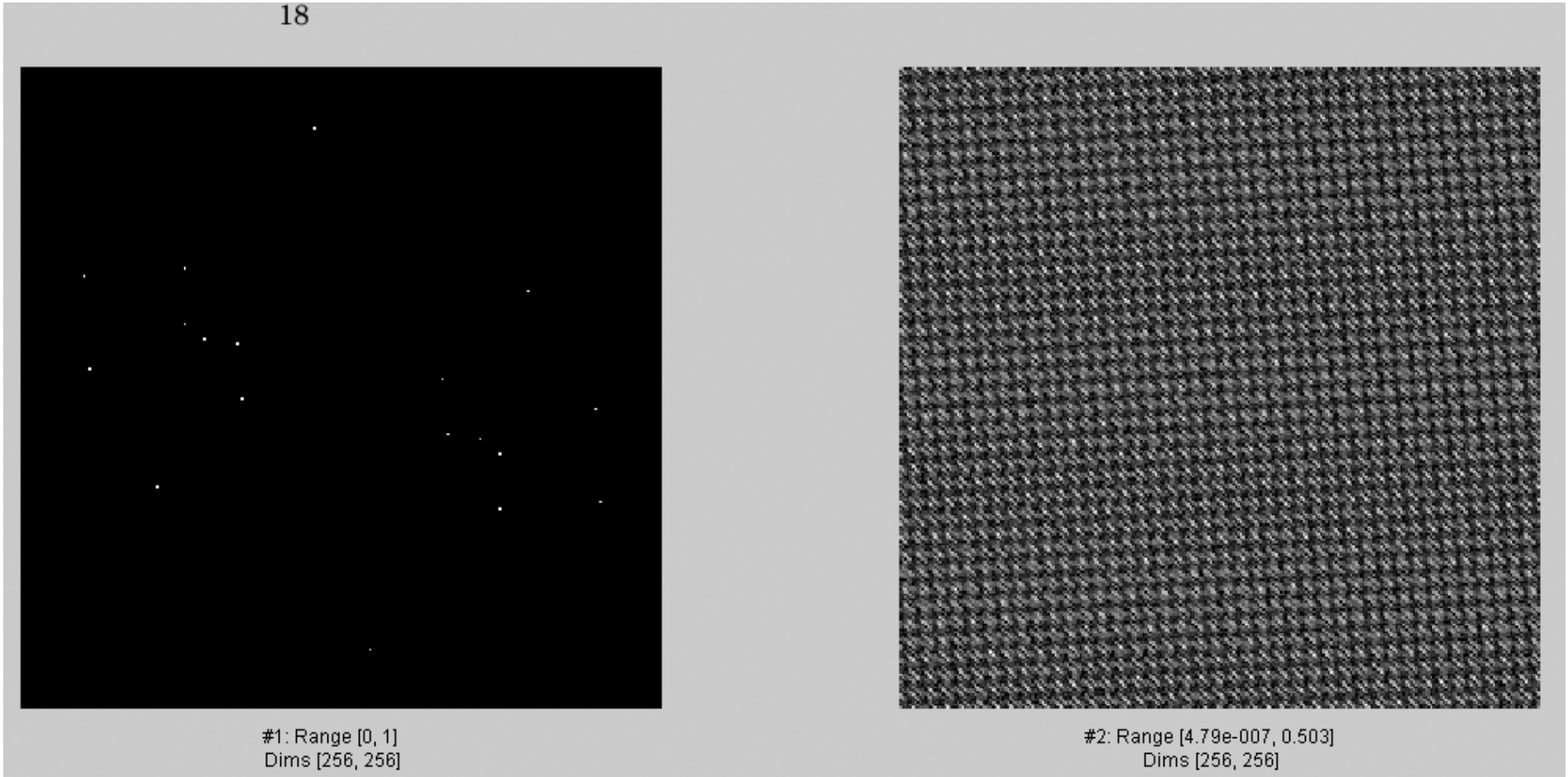
2



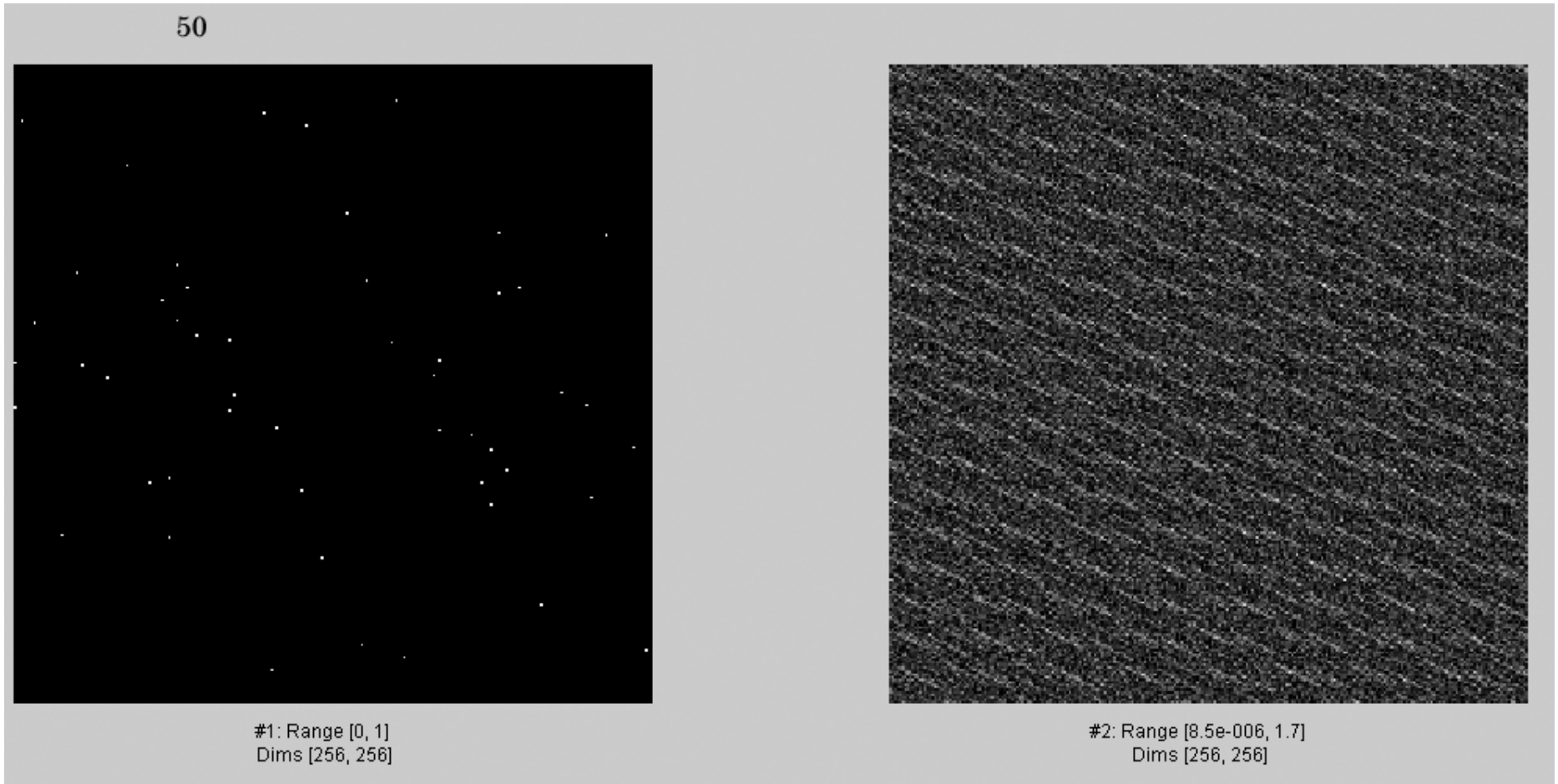
6



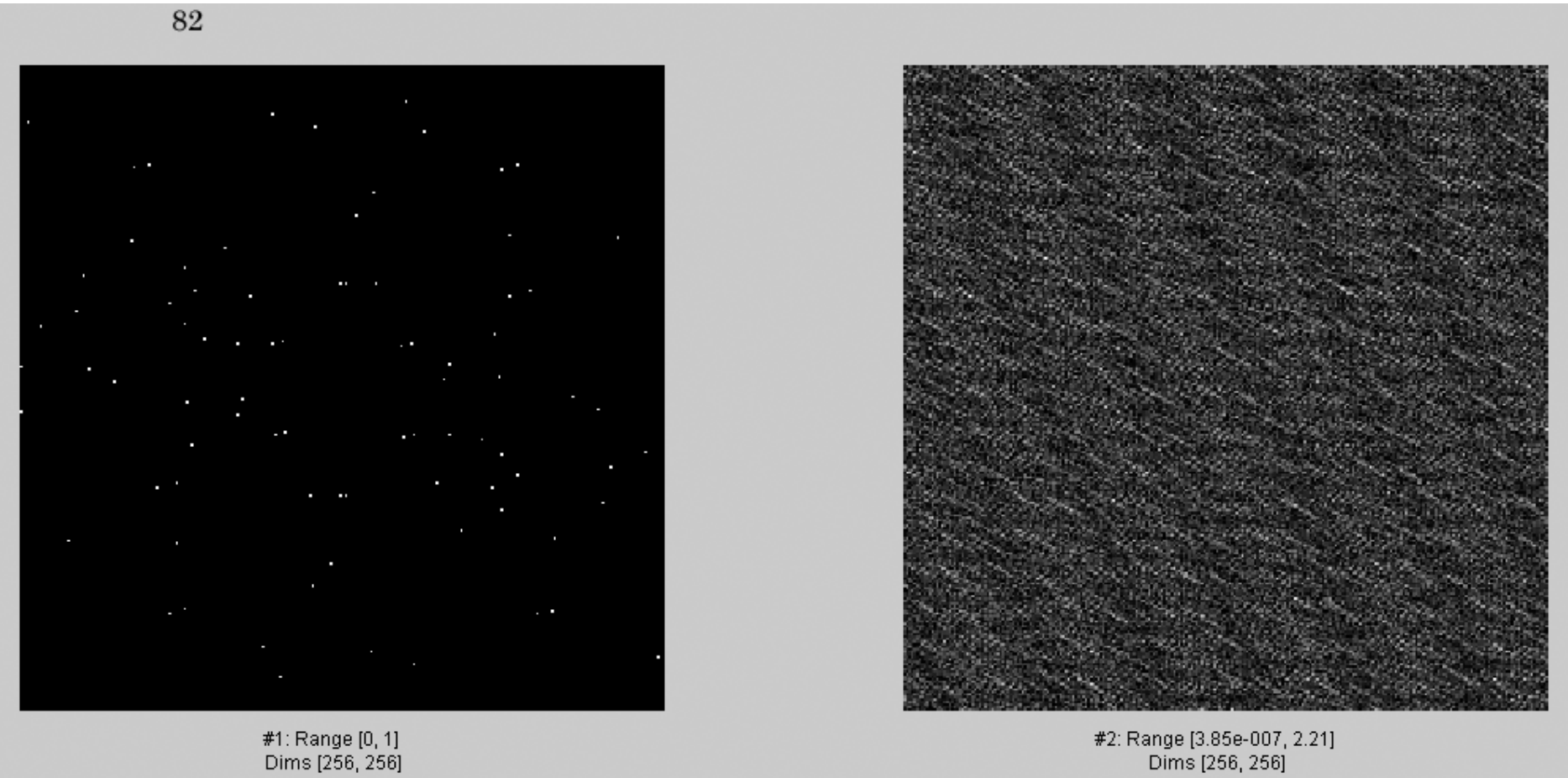
18



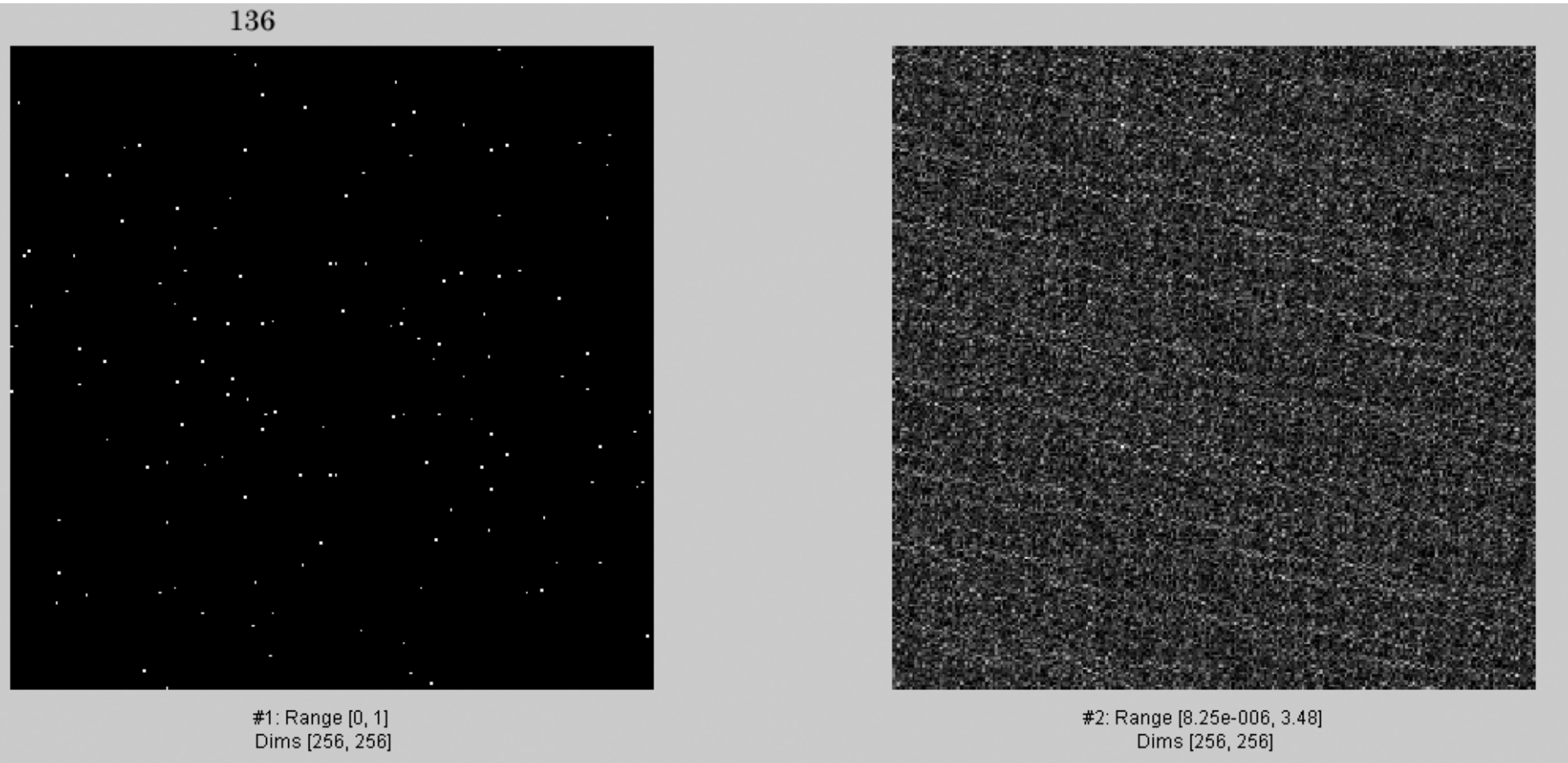
50



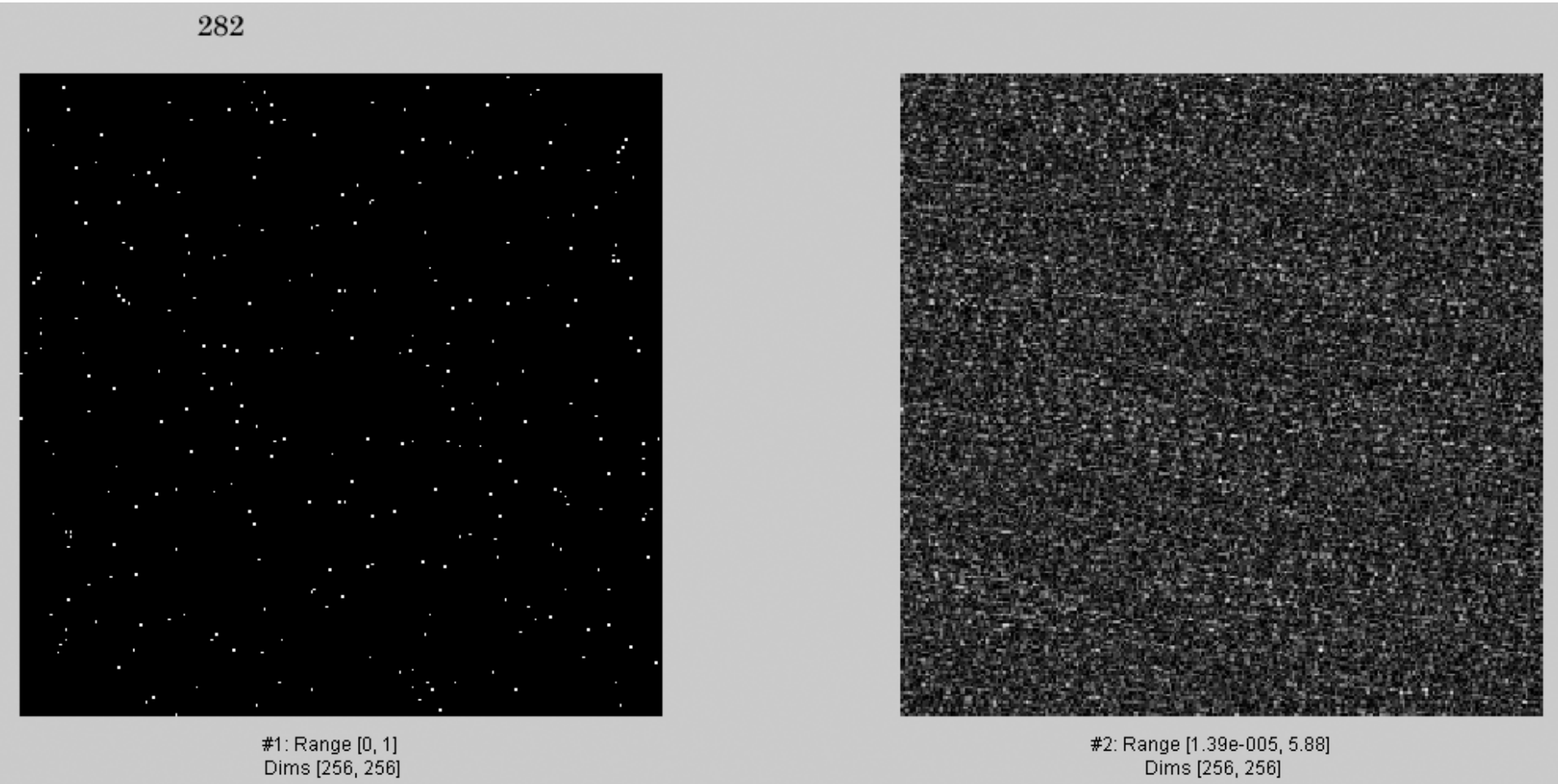
82



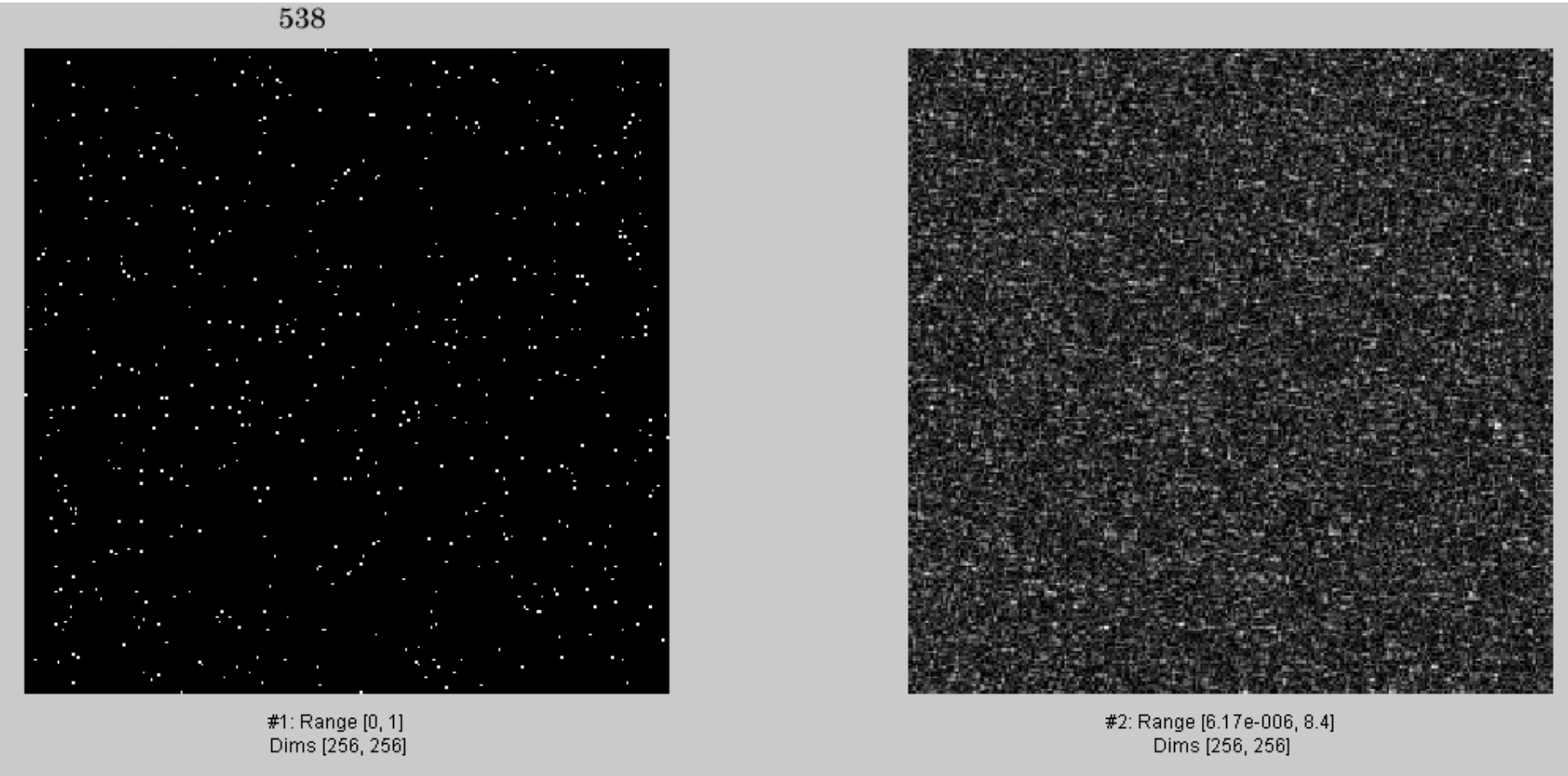
136



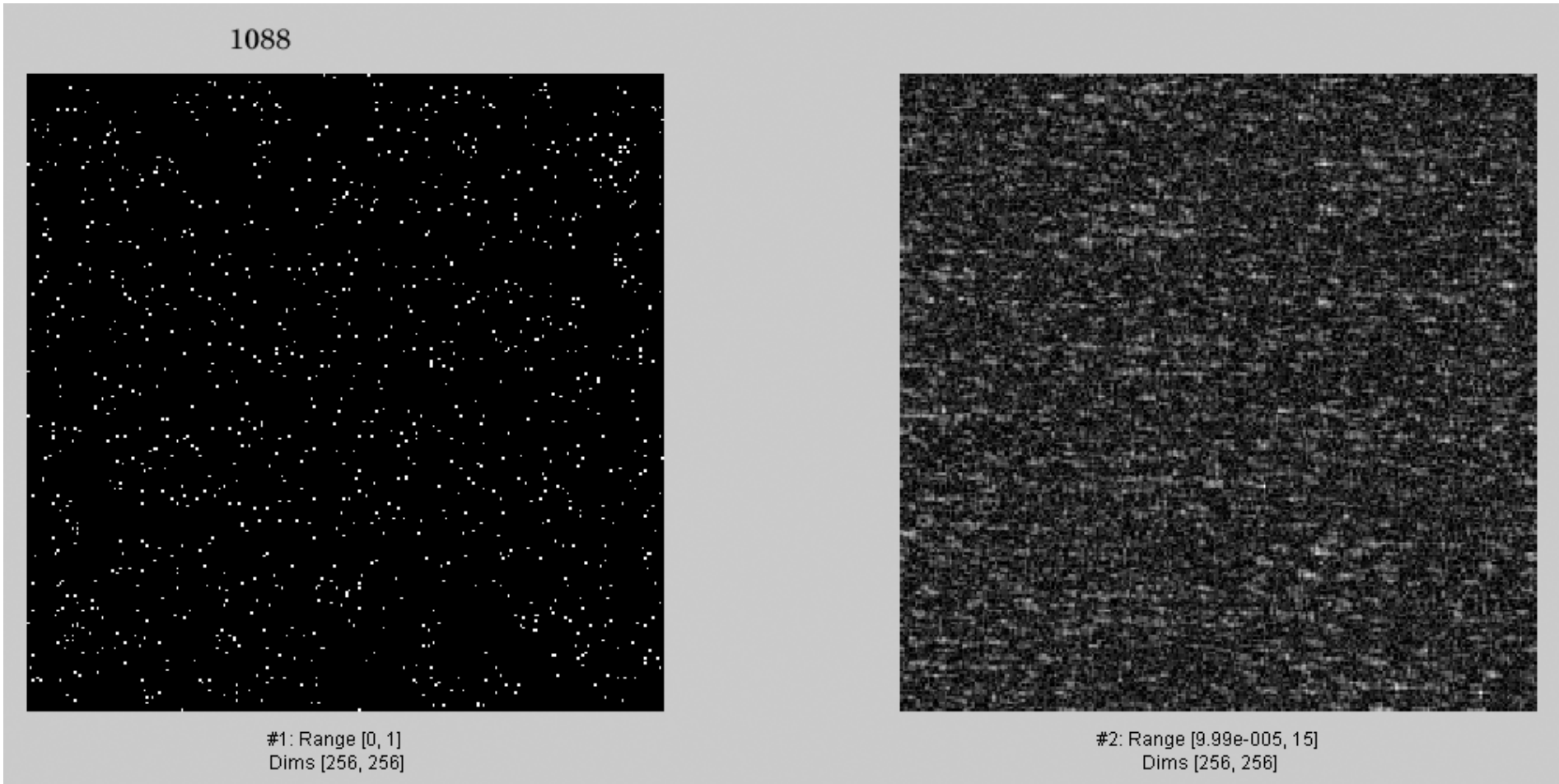
282



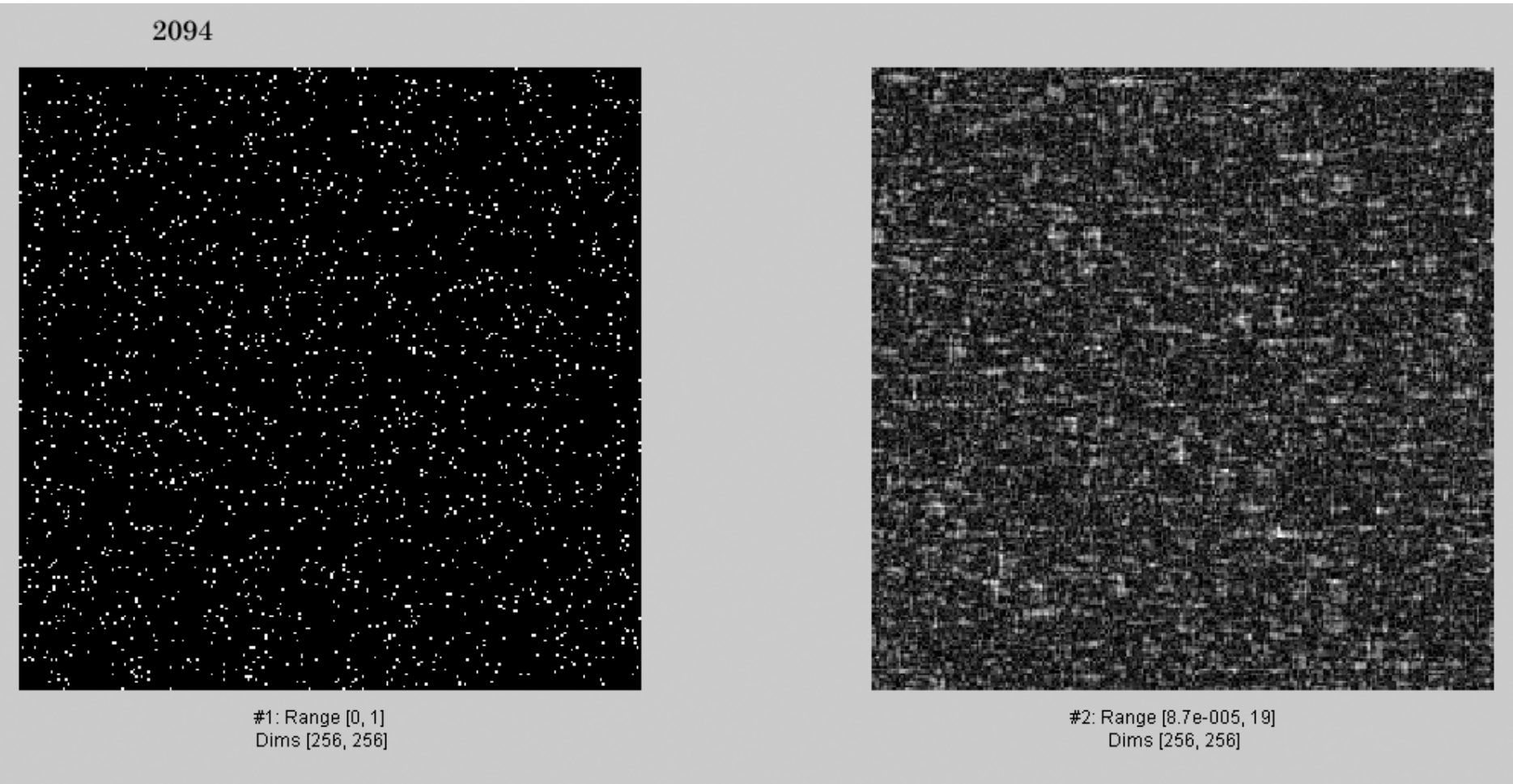
538



1088

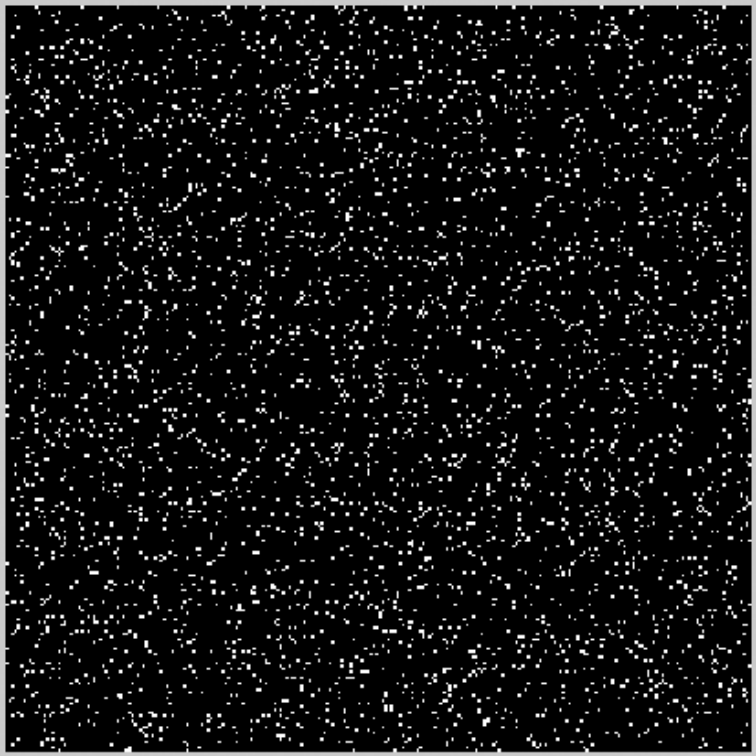


2094

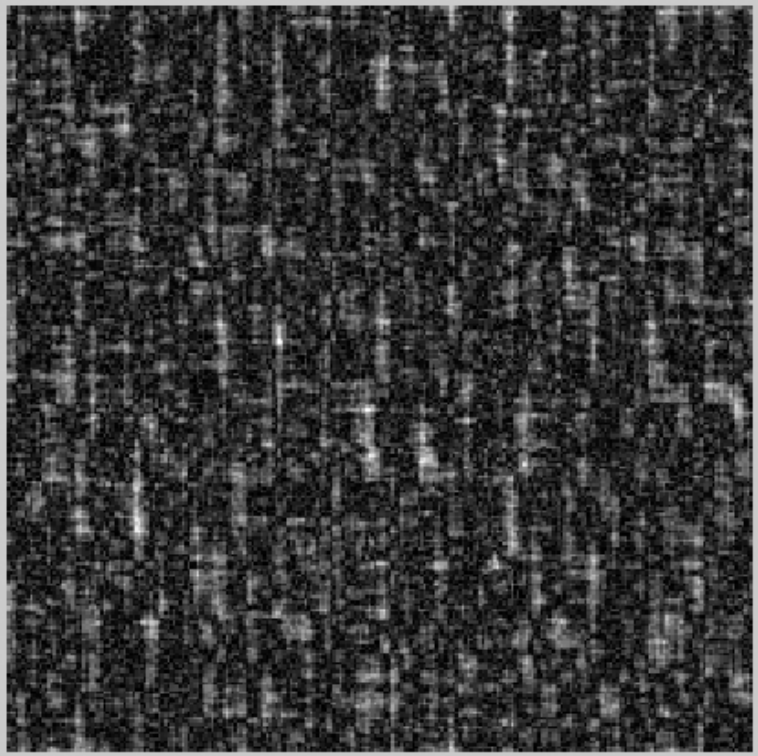


4052.

4052

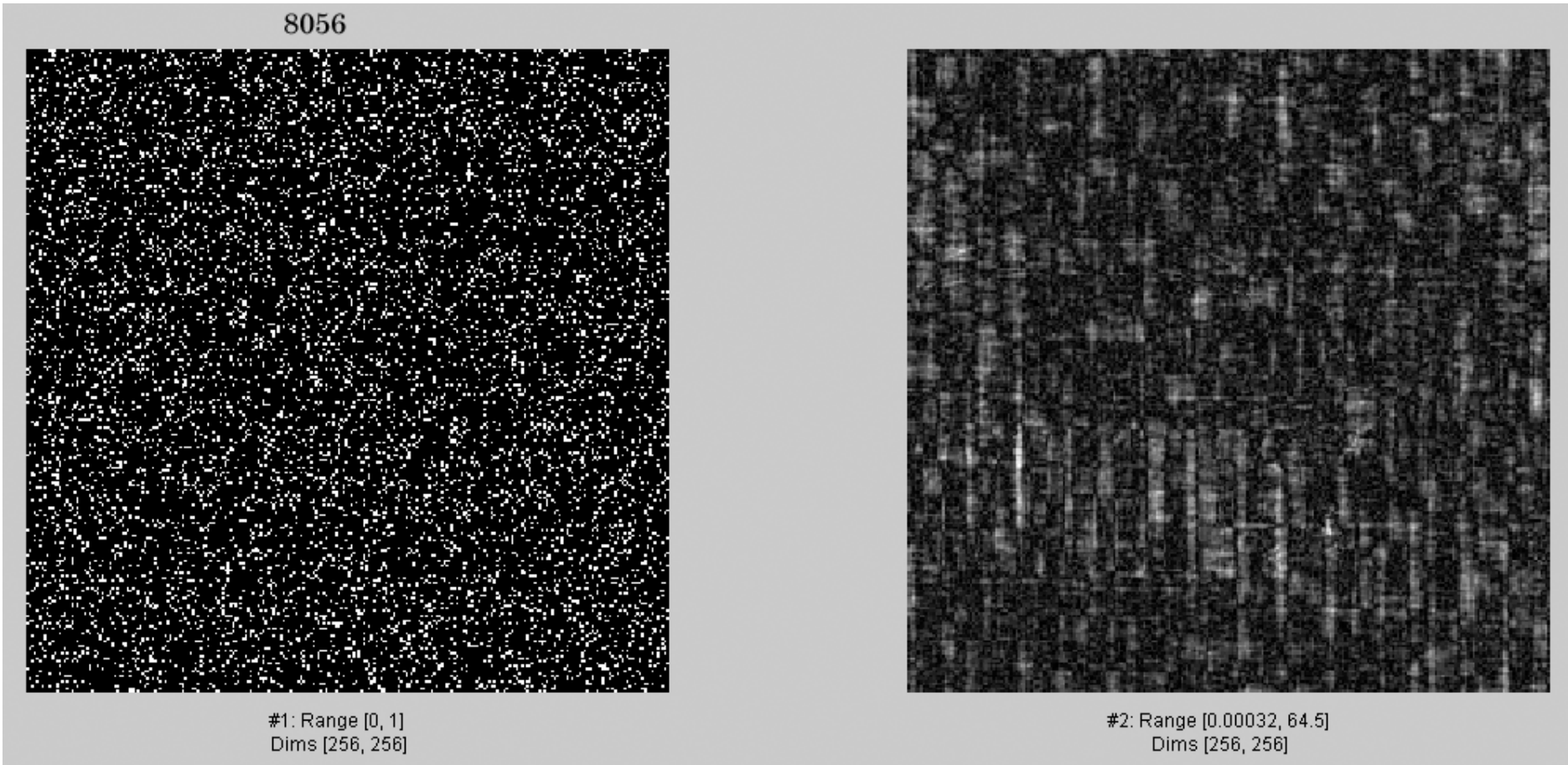


#1: Range [0, 1]
Dims [256, 256]



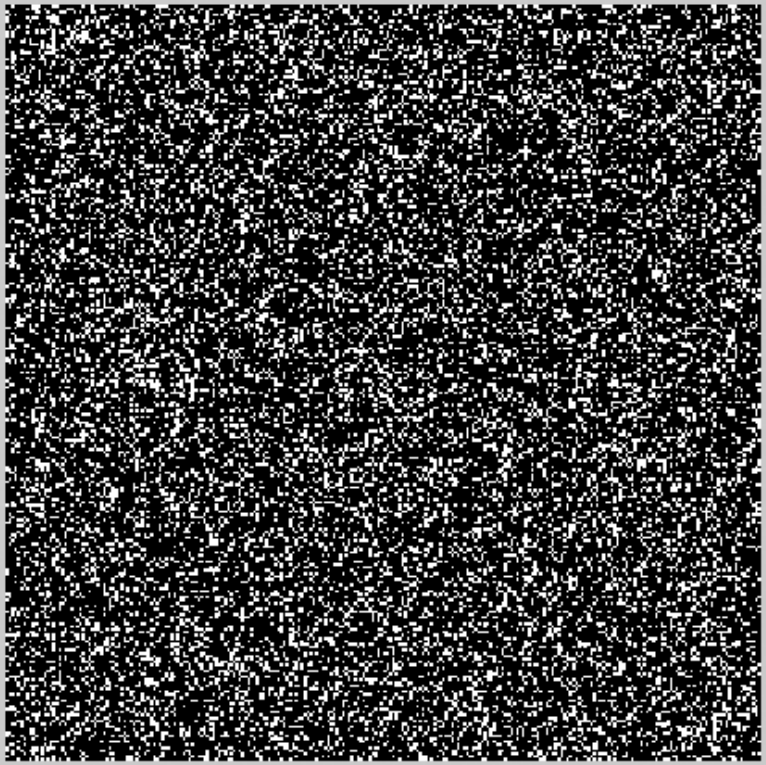
#2: Range [0.000556, 37.7]
Dims [256, 256]

8056.

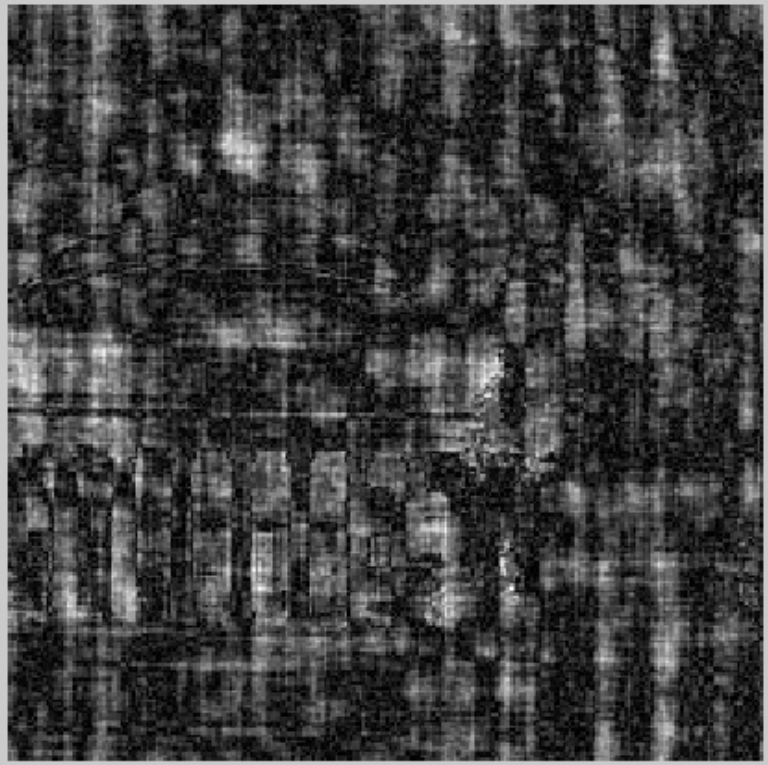


15366

15366



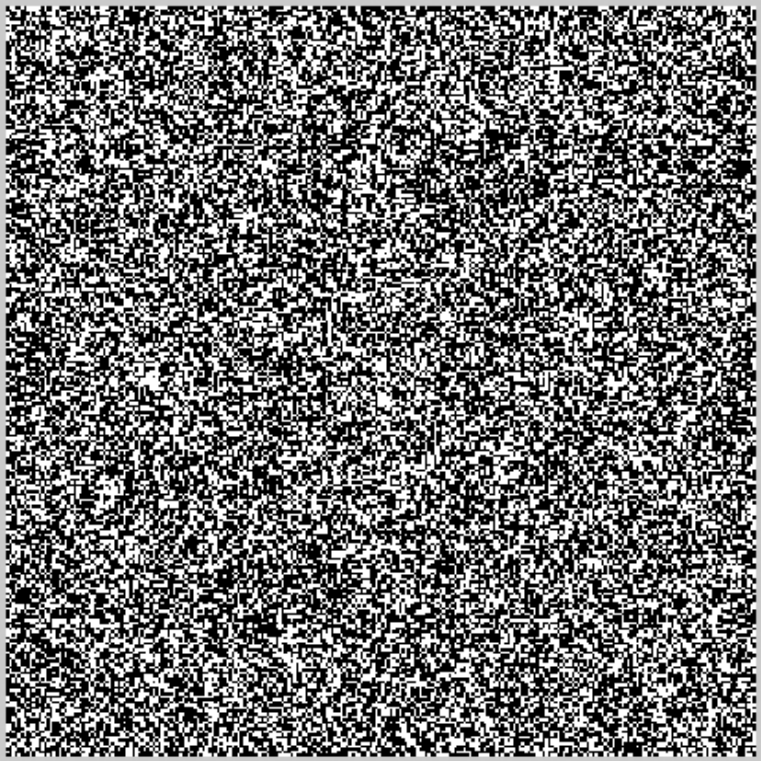
#1: Range [0, 1]
Dims [256, 256]



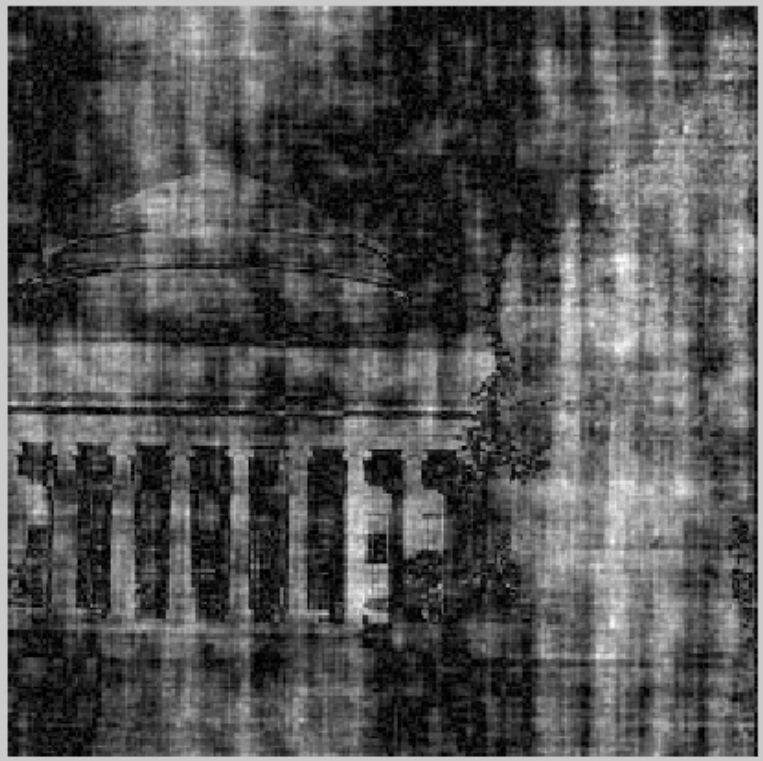
#2: Range [0.000231, 91.1]
Dims [256, 256]

28743

28743



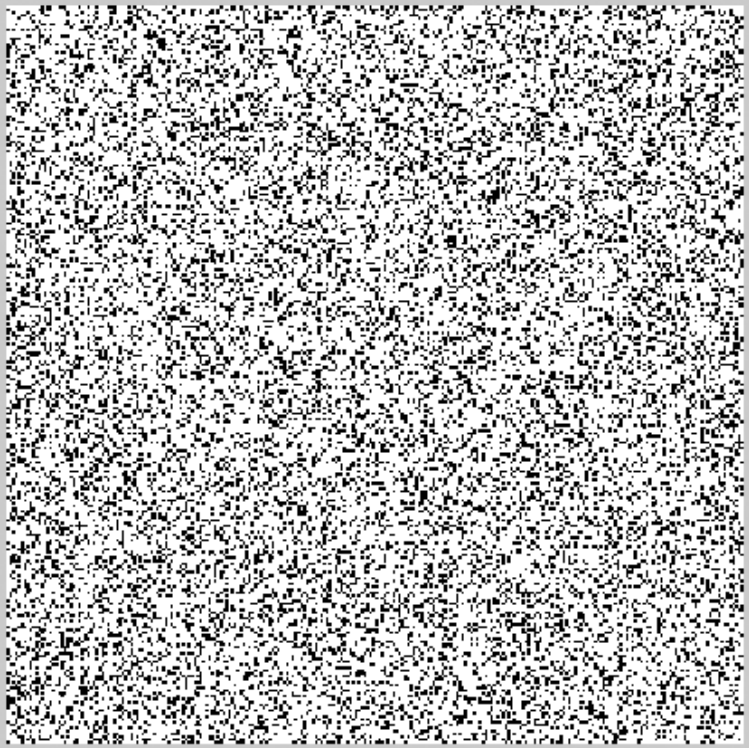
#1: Range [0, 1]
Dims [256, 256]



#2: Range [0.00109, 146]
Dims [256, 256]

49190.

49190



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

#2: Range [0.00758, 294]
Dims [256, 256]

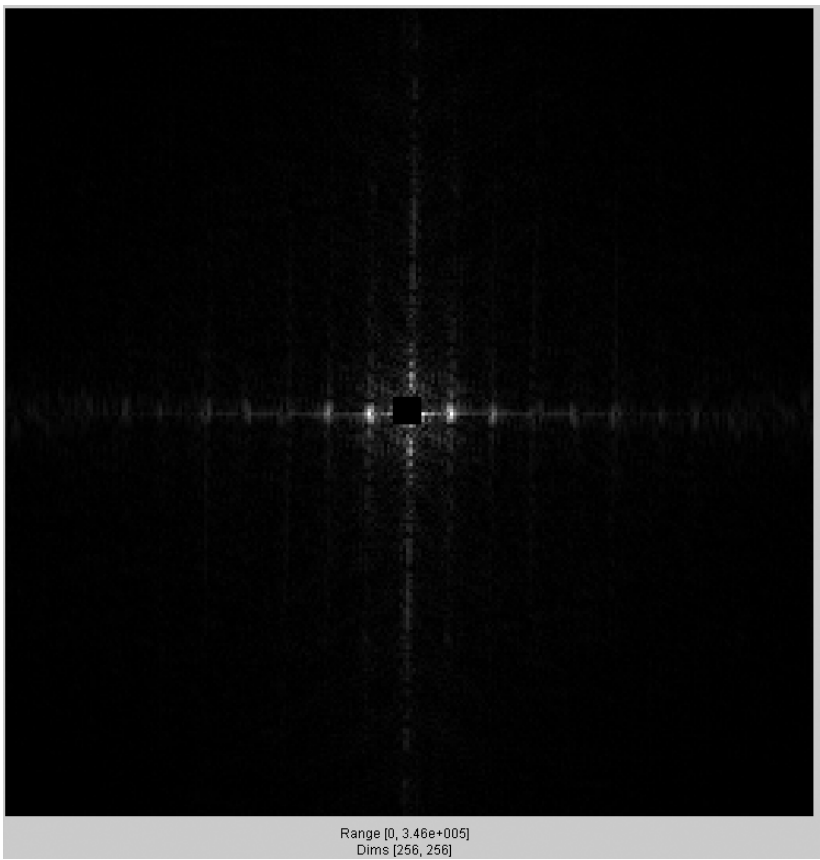
65536.

65536.

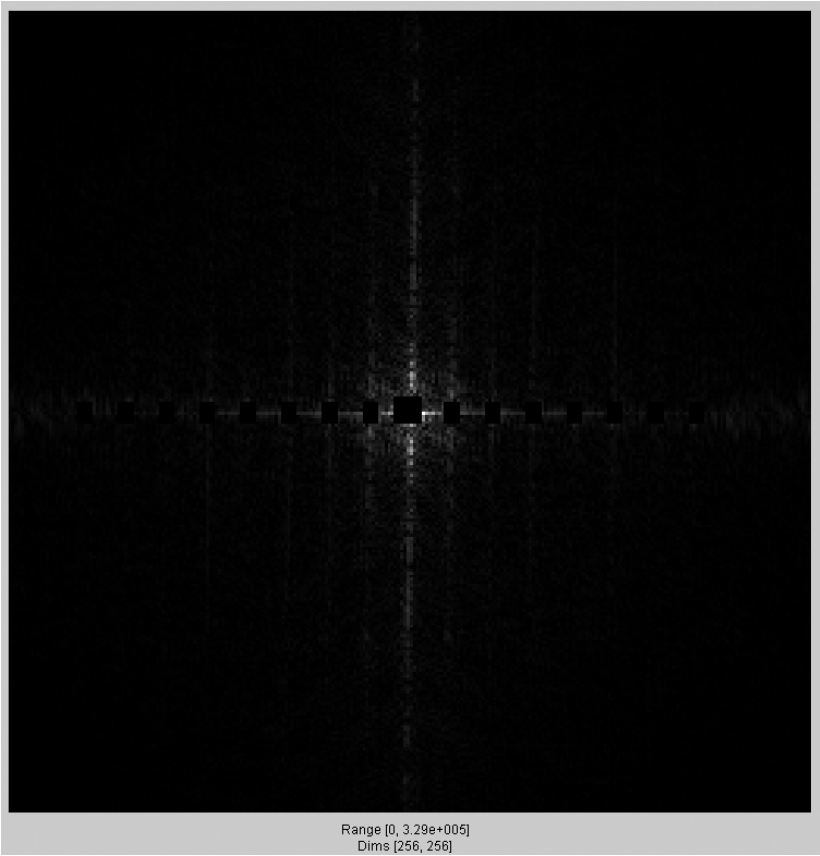
#1: Range [0.5, 1.5]
Dims [256, 256]

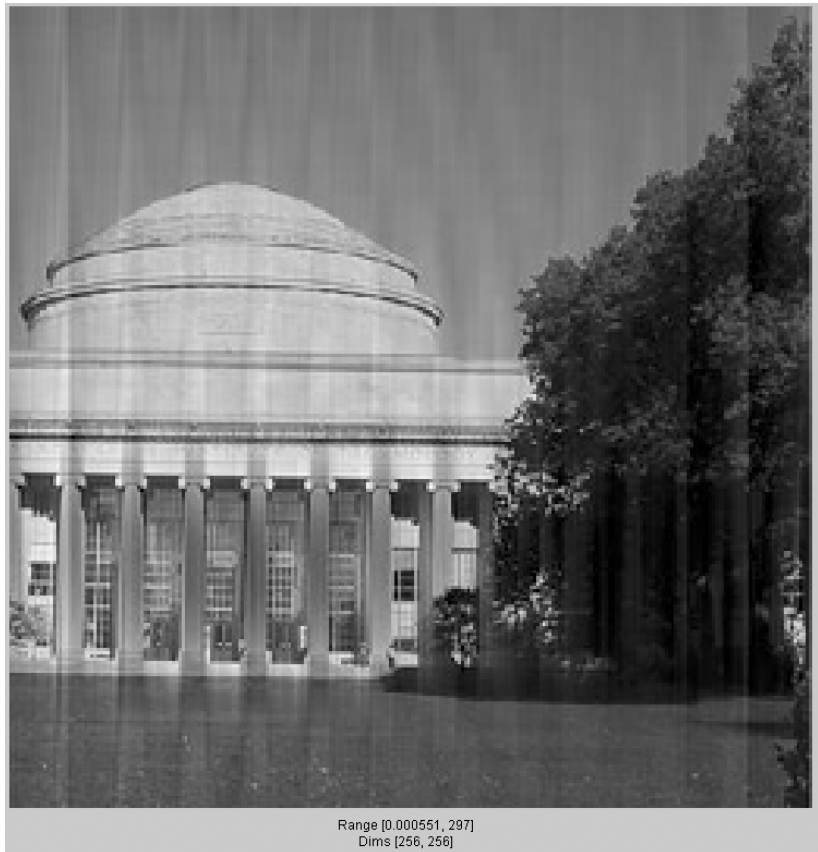
#2: Range [4.43e-015, 255]
Dims [256, 256]

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

Forsyth&Ponce

Discrete-time, continuous frequency Fourier transform

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

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

where

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

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

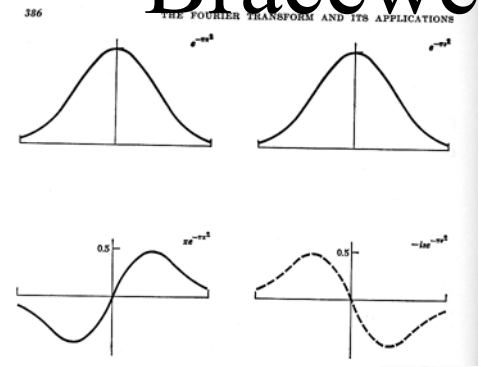
Discrete-time, continuous frequency Fourier transform pairs

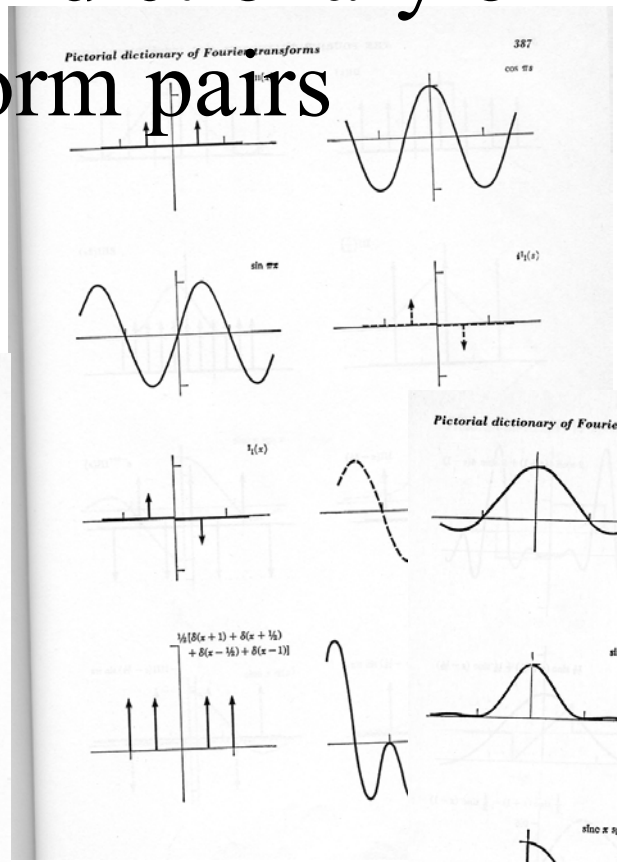
TABLE 2.3 FOURIER TRANSFORM PAIRS

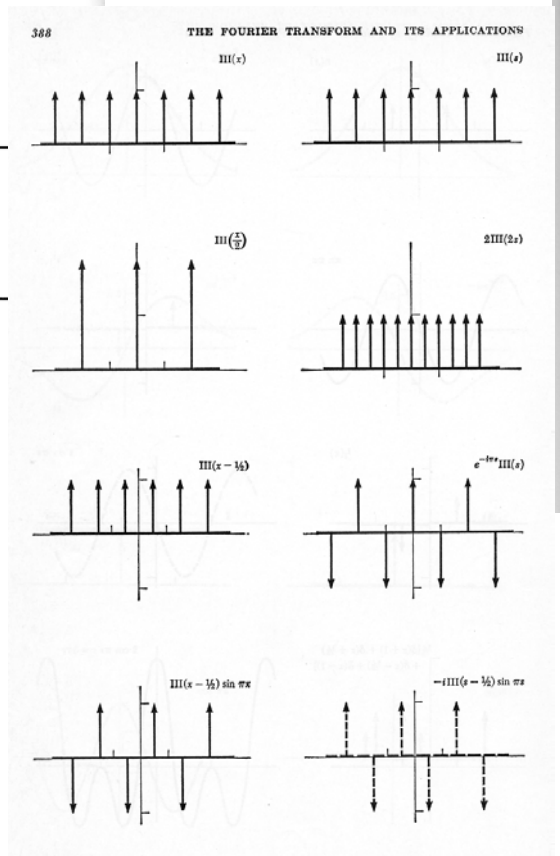
Sequence	Fourier Transform
1. $\delta[n]$	1
2. $\delta[n-n_0]$	$e^{-j\omega n_0}$
3. $1 \quad (-\infty < n < \infty)$	$\sum_{k=-\infty}^{\infty} 2\pi\delta(\omega+2\pi k)$
4. $a^n u[n] \quad (\lvert a\rvert < 1)$	$\frac{1}{1-ae^{-j\omega}}$
5. $u[n]$	$\frac{1}{1-e^{-j\omega}} + \sum_{k=-\infty}^{\infty}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

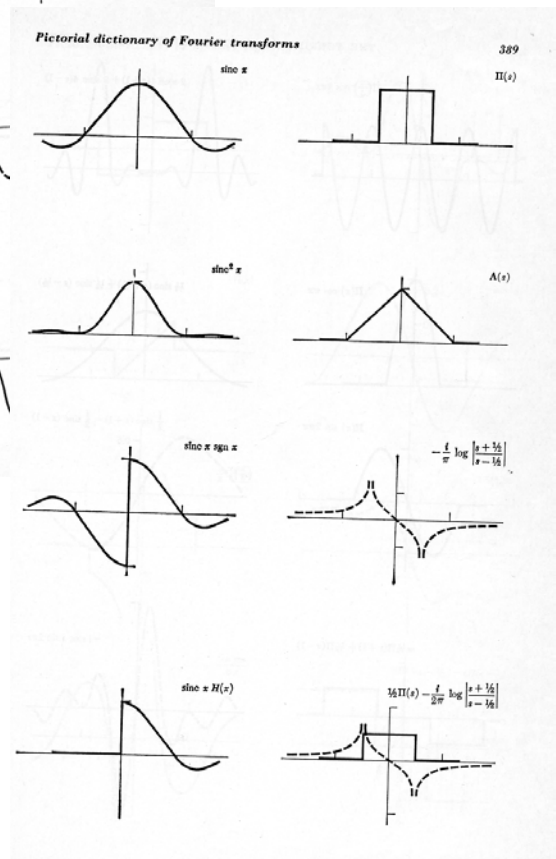
Oppenheim, Schafer and Buck, Discrete-time signal processing, Prentice Hall, 1999

Bracewell's pictorial dictionary of Fourier transform pairs









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

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

$$f = g \otimes h$$

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

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

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

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

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

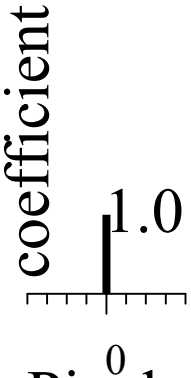
Perform the other DFT (circular boundary conditions)

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

Analysis of our simple filters

Analysis of our simple filters

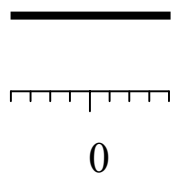
original



Filtered
(no change)

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

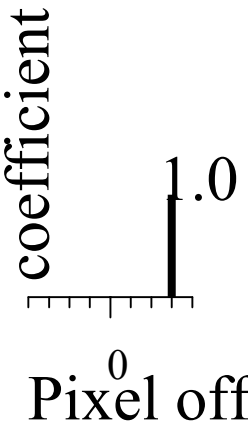
$$= 1$$

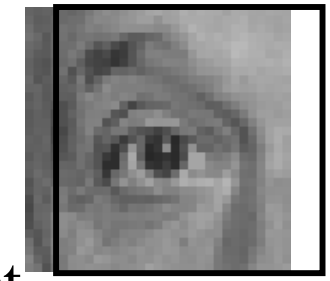


constant

Analysis of our simple filters

original

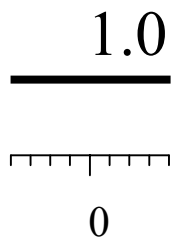




shifted

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

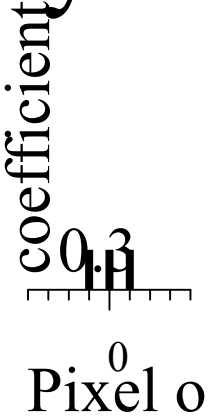
$$= e^{-\pi i\frac{\delta m}{M}}$$



Constant magnitude, linearly shifted phase

Analysis of our simple filters

original

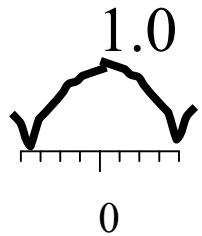


blurred

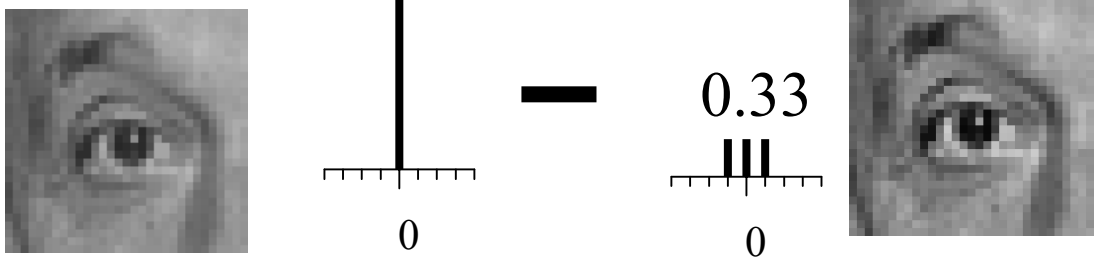
$$F[m,n] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} f[k,l]e^{-\pi i\left(\frac{km}{M}+\frac{\ln}{N}\right)}$$

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

Low-pass filter



Analysis of our simple filters



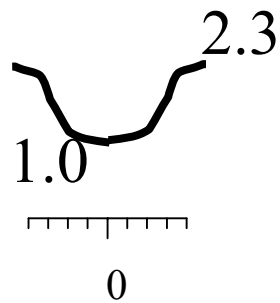
original

sharpened

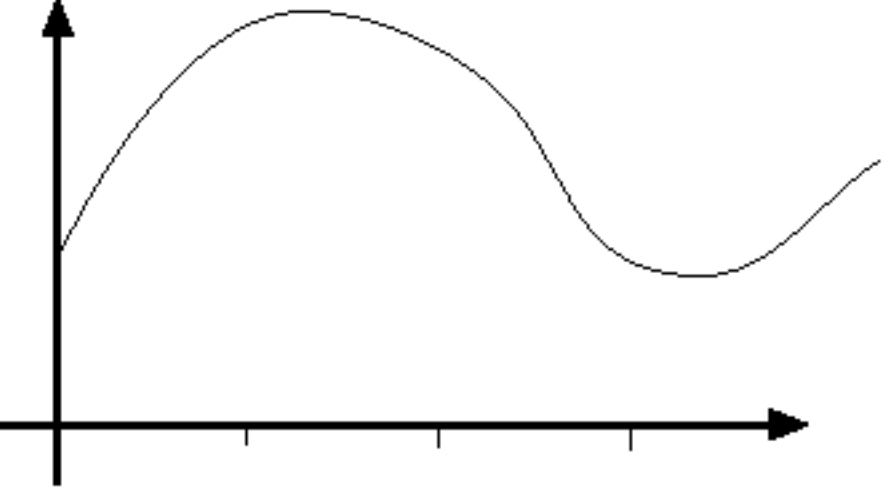
$$F[m,n] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} f[k,l]e^{-\pi i\left(\frac{km}{M}+\frac{\ln}{N}\right)}$$

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

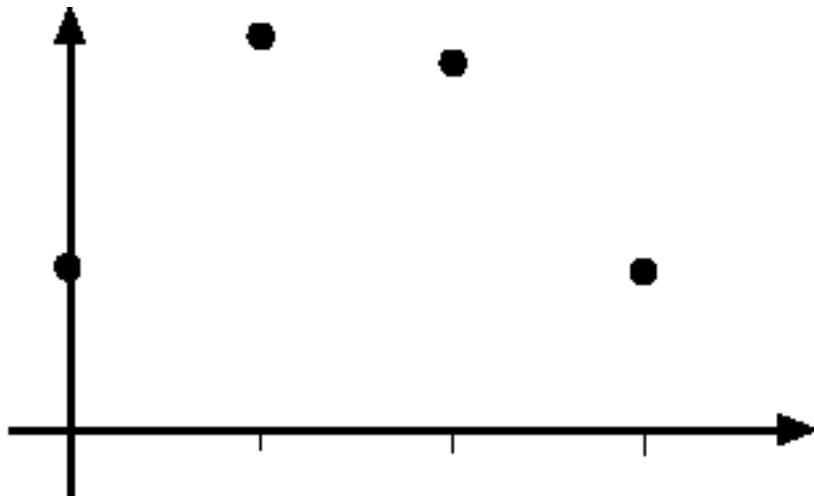
high-pass filter

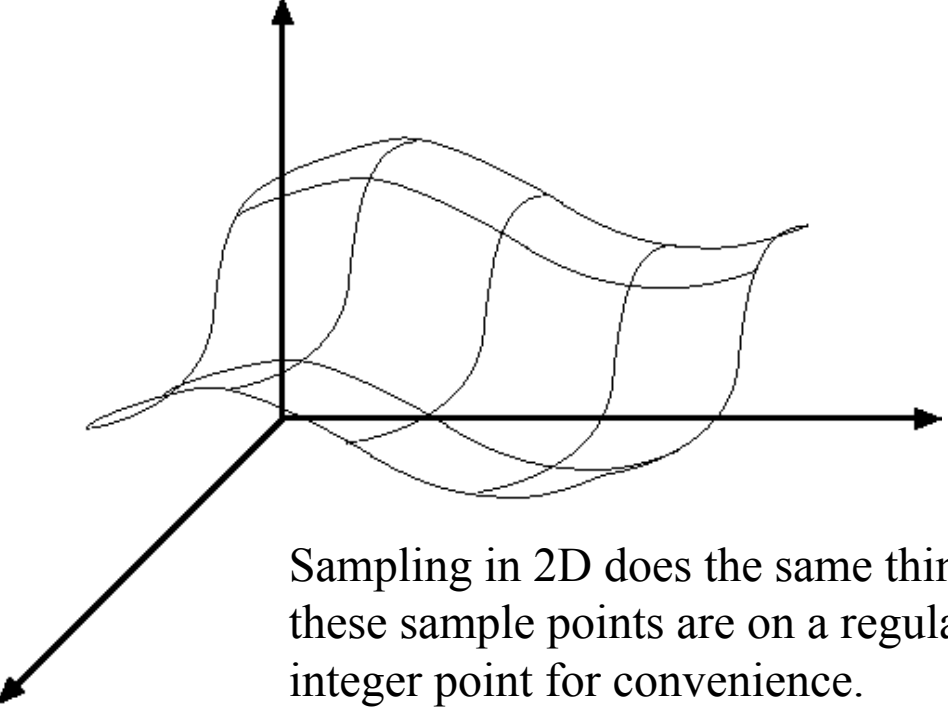


Sampling and aliasing

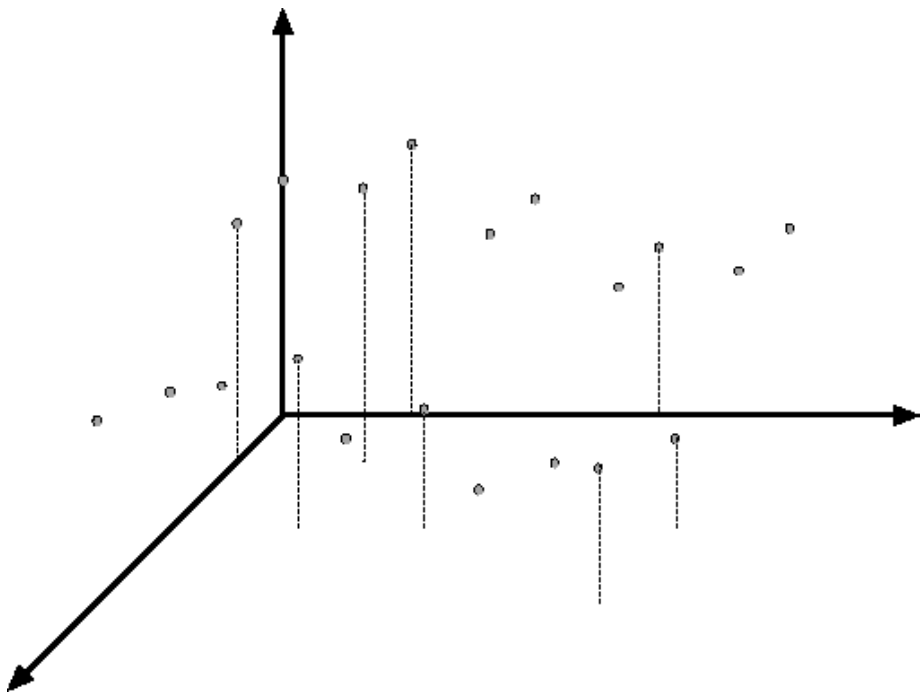


Sampling in 1D takes a continuous function and replaces it with a vector of values, consisting of the function's values at a set of sample points. We'll assume that these sample points are on a regular grid, and can place one at each integer for convenience.





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

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

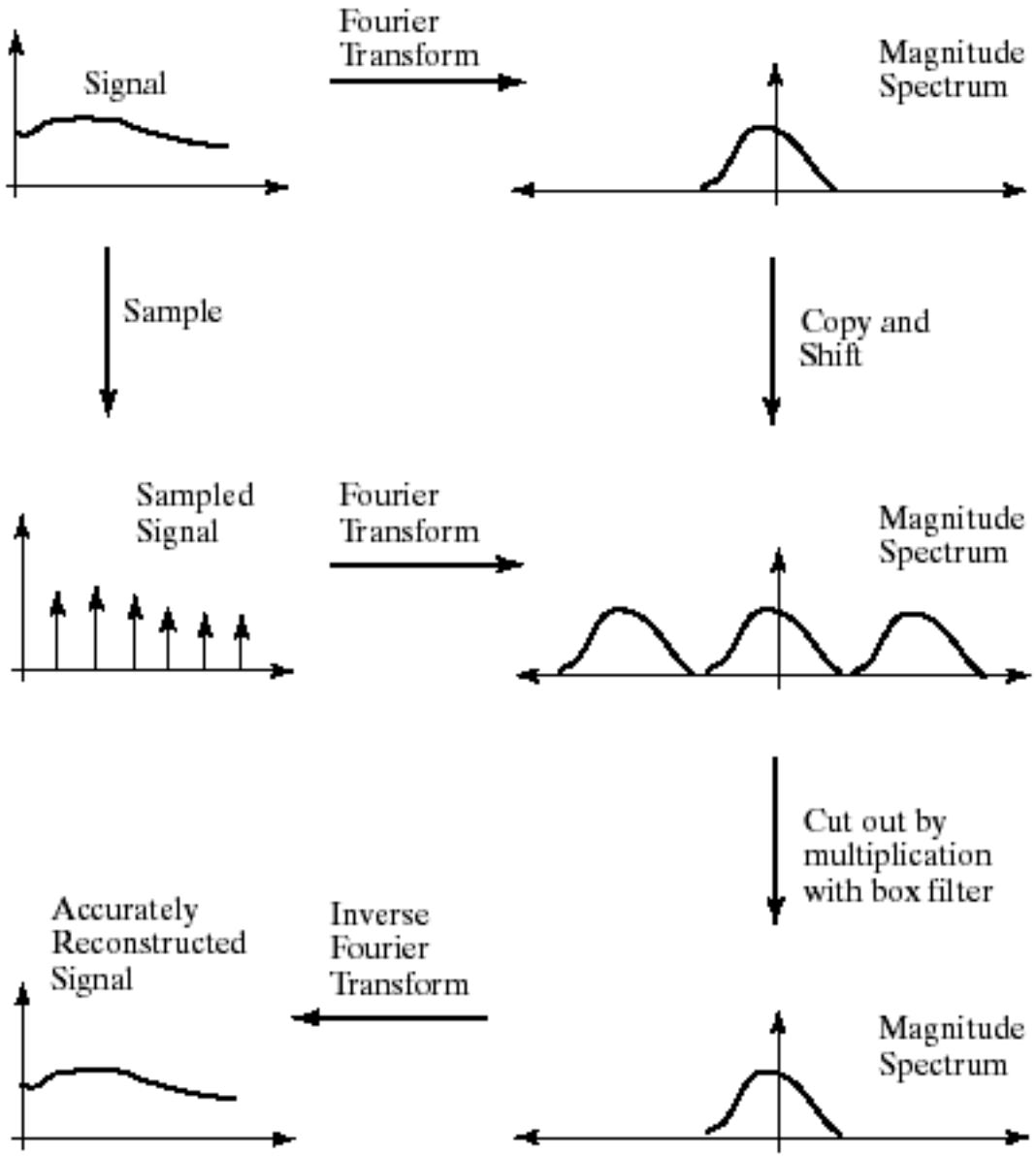
$$= f(x,y)\sum_{i=-\infty}^{\infty}\sum_{i=-\infty}^{\infty}\delta(x-i,y-j)$$

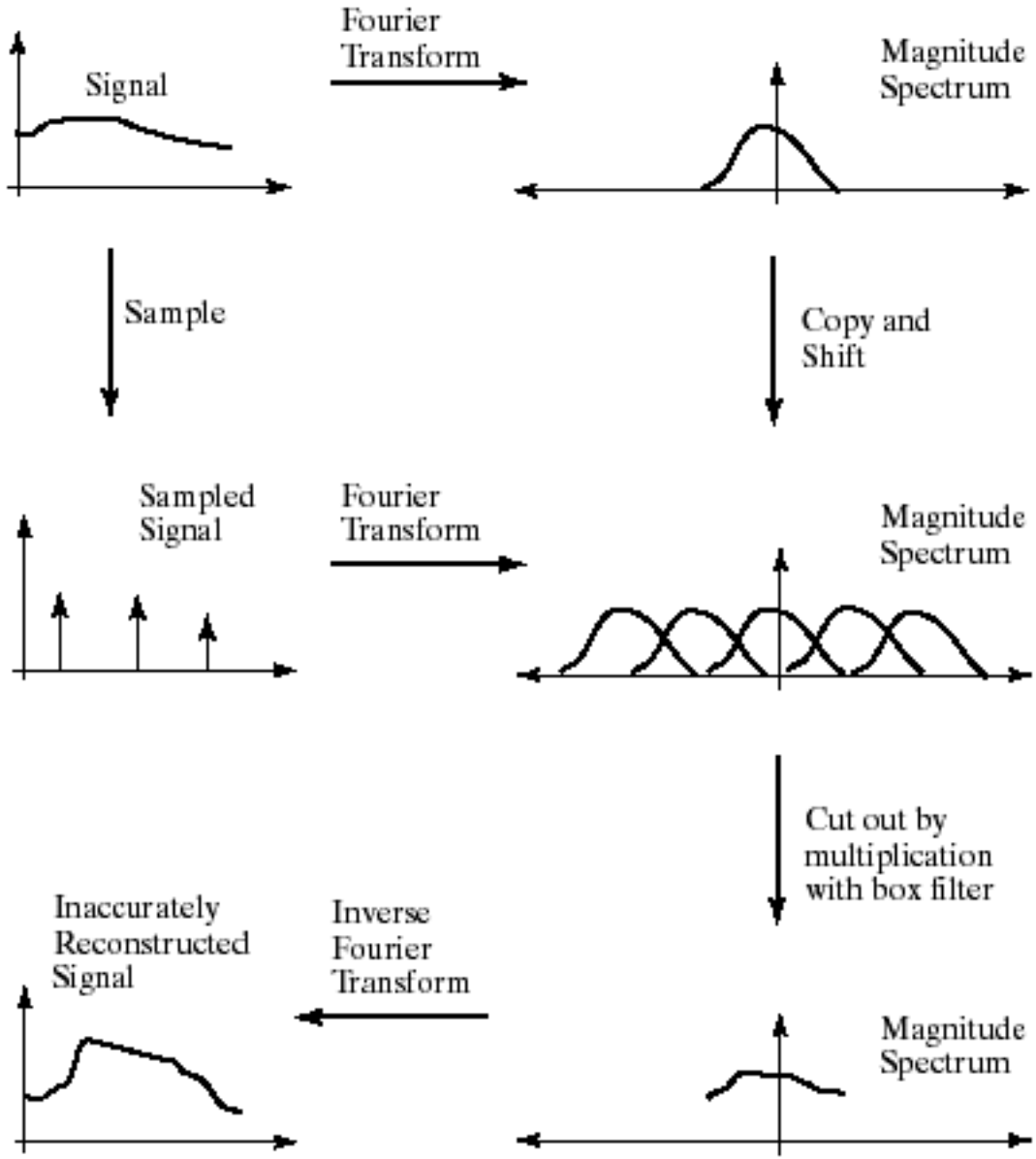
The Fourier transform of a sampled signal

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

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

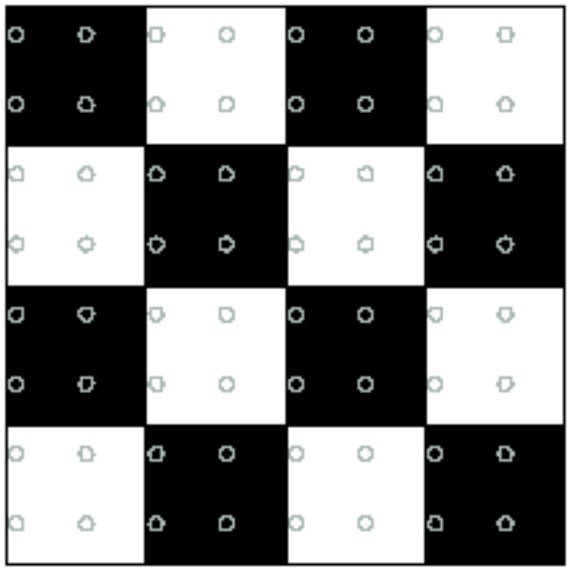
$$= \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}F(u-i,v-j)$$

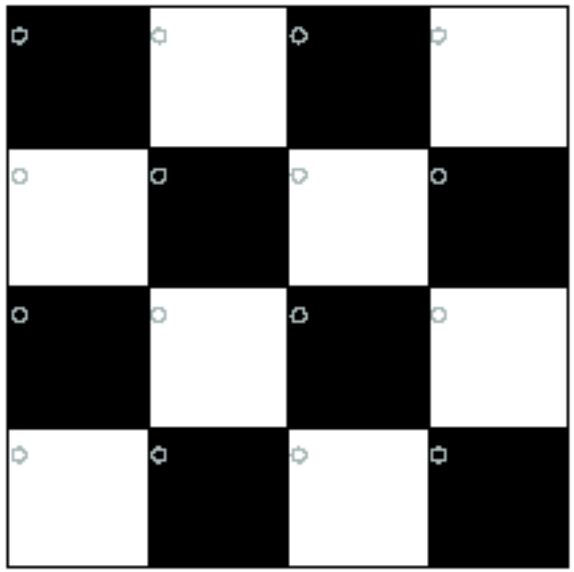

Signal
Fourier Transform
Magnitude Spectrum
Sample
Copy and Shift
Sampled Signal
Fourier Transform
Magnitude Spectrum
Cut out by multiplication with box filter
Accurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum

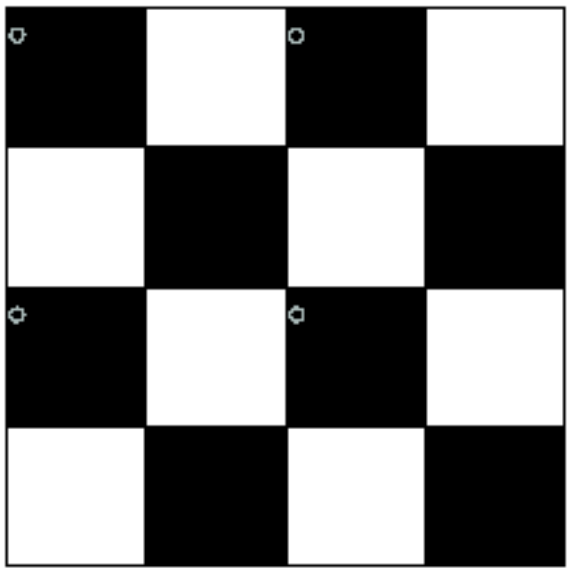

Signal
Fourier Transform
Magnitude Spectrum
Sample
Copy and Shift
Sampled Signal
Fourier Transform
Magnitude Spectrum
Cut out by multiplication with box filter
Inaccurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum

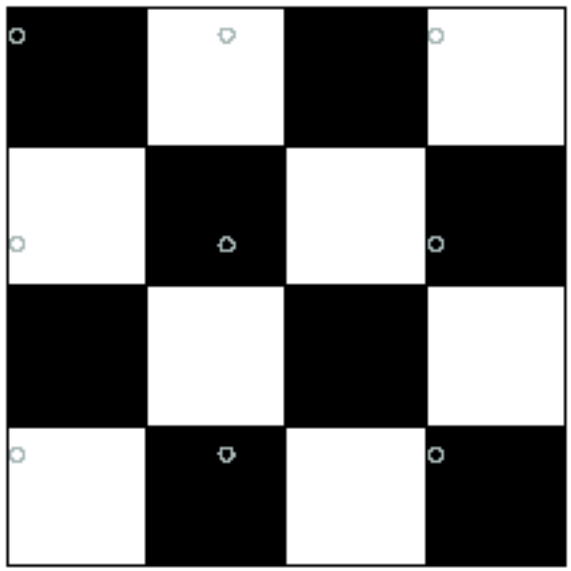
Aliasing

- 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

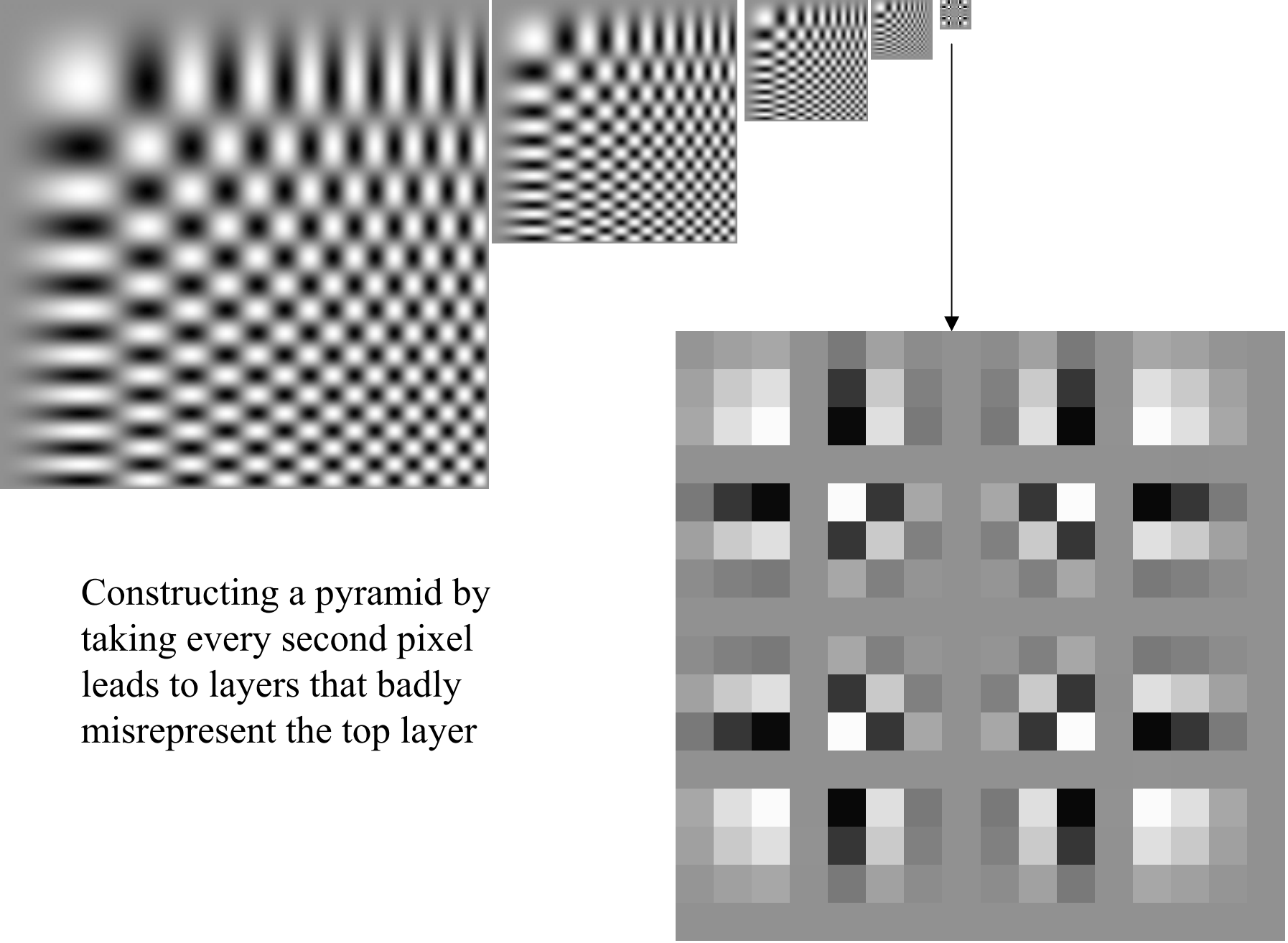








Resample the checkerboard by taking one sample at each circle. In the case of the top left board, new representation is reasonable.
Top right also yields a reasonable representation.
Bottom left is all black (dubious) and bottom right has checks that are too big.

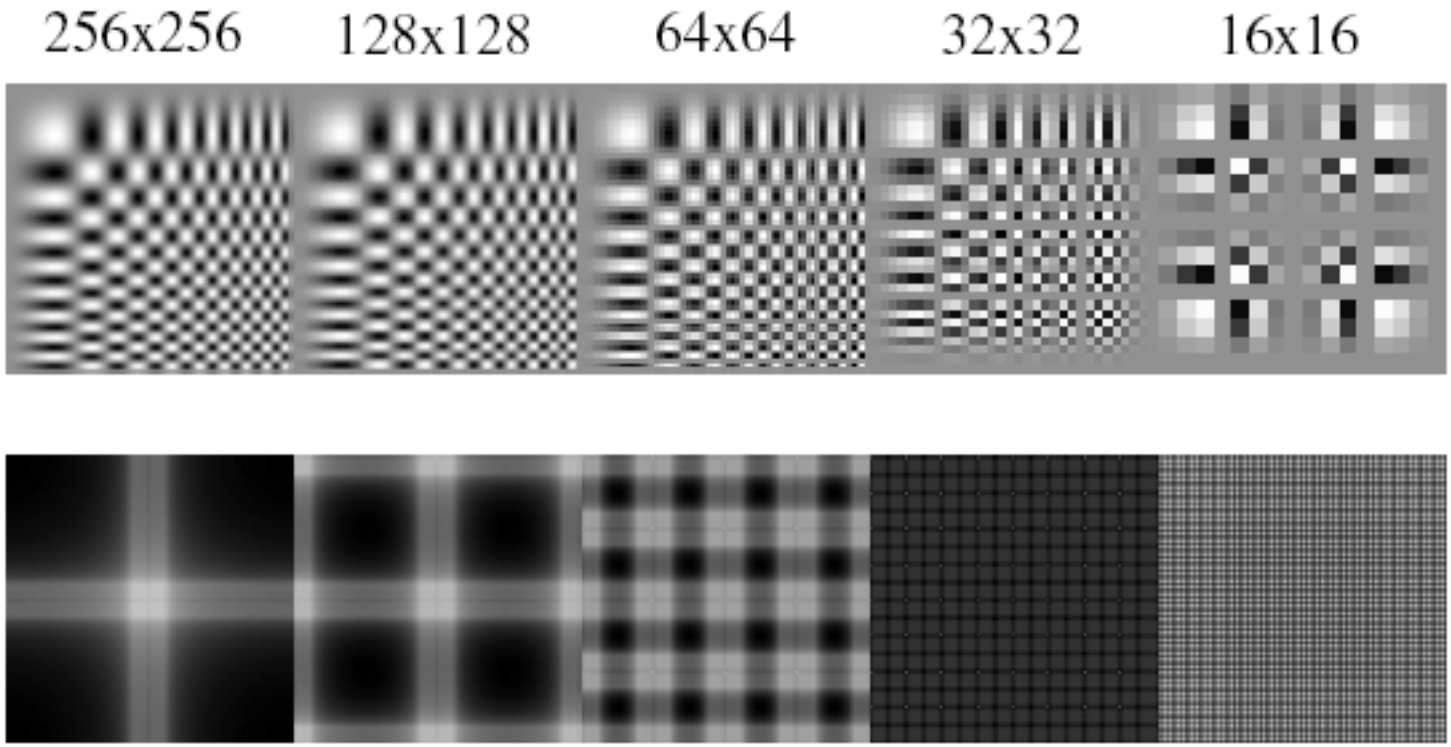


Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer

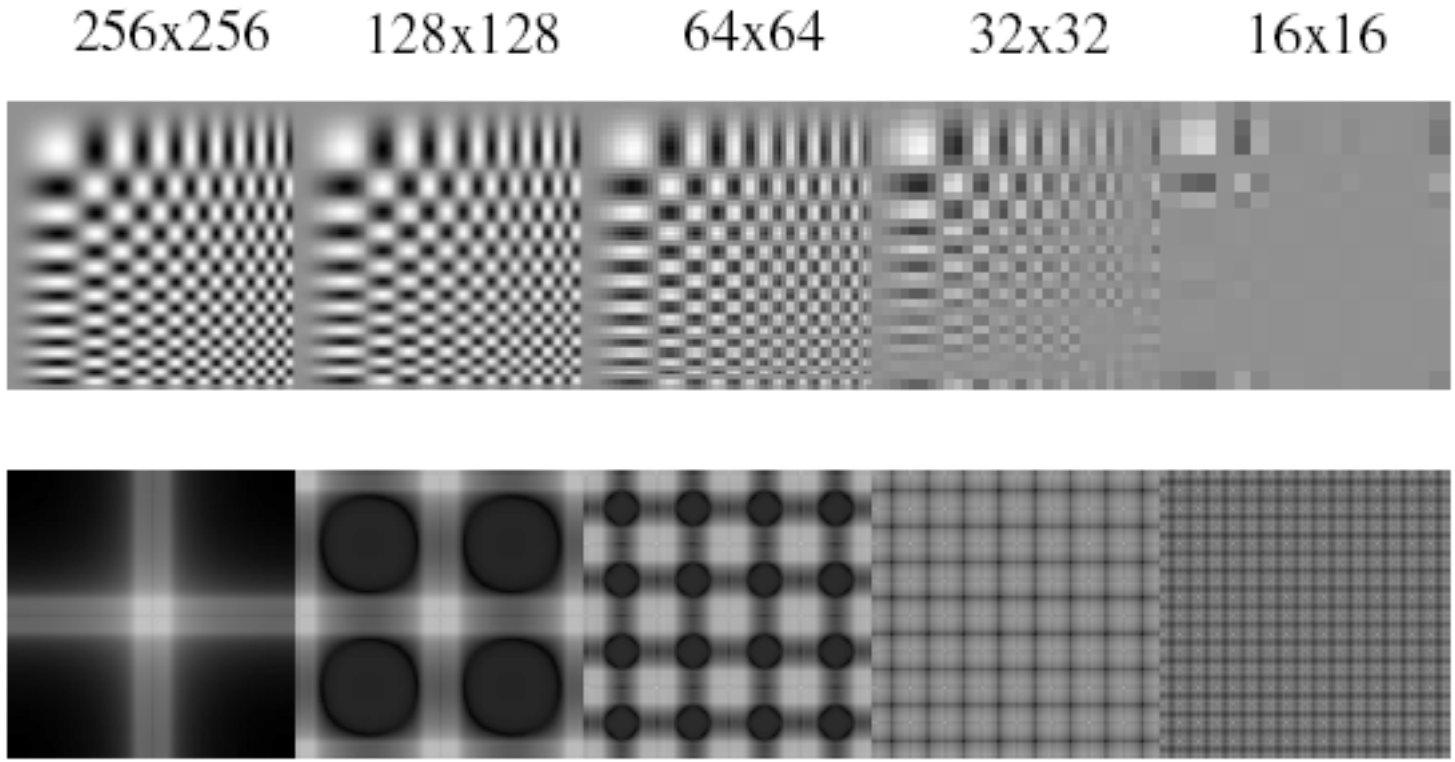
Smoothing as low-pass filtering

- 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.

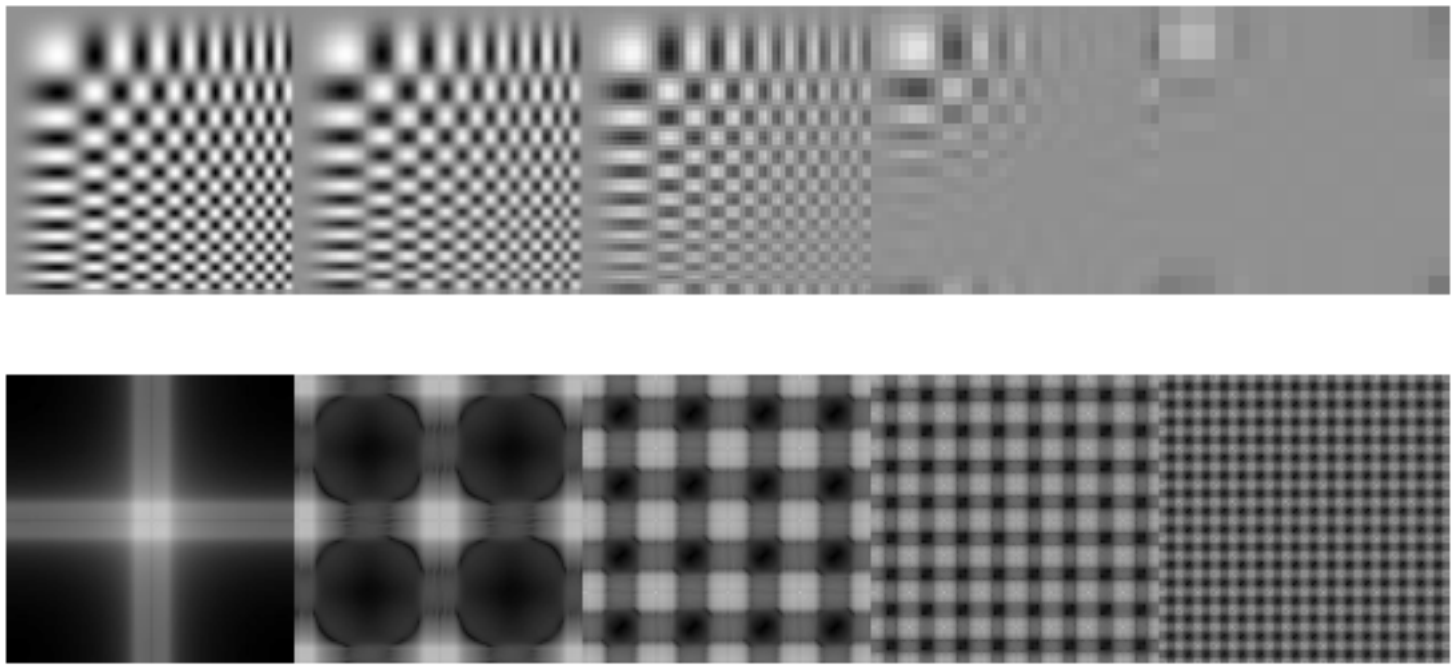


Sampling with smoothing. Top row shows the images. We get the next image by smoothing the image with a Gaussian with sigma 1 pixel, then sampling at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.



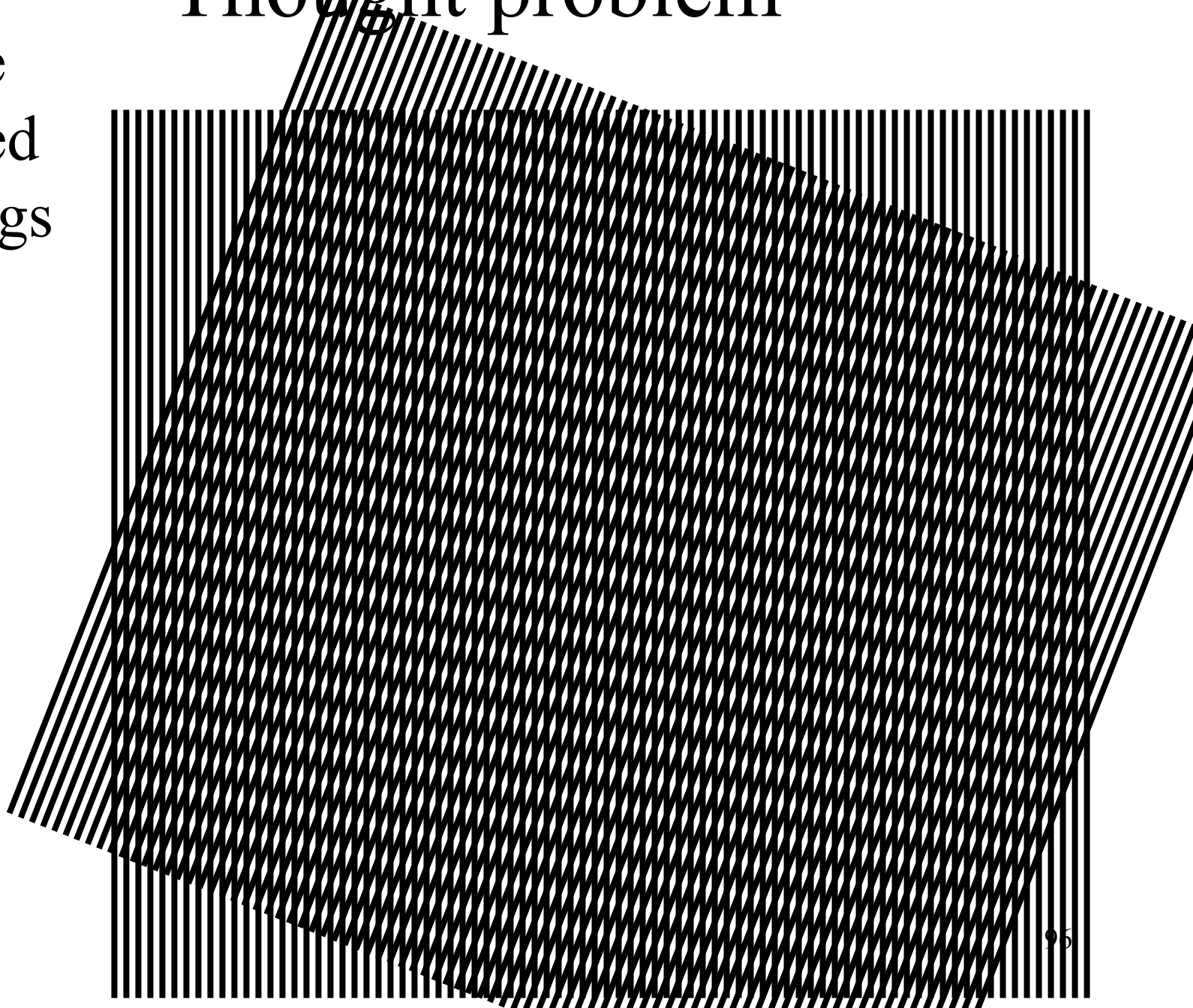
Sampling with smoothing. Top row shows the images. We get the next image by smoothing the image with a Gaussian with sigma 1.4 pixels, then sampling at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.

256x256 128x128 64x64 32x32 16x16



Thought problem

Analyze crossed gratings



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.

Image pyramids

The Gaussian pyramid

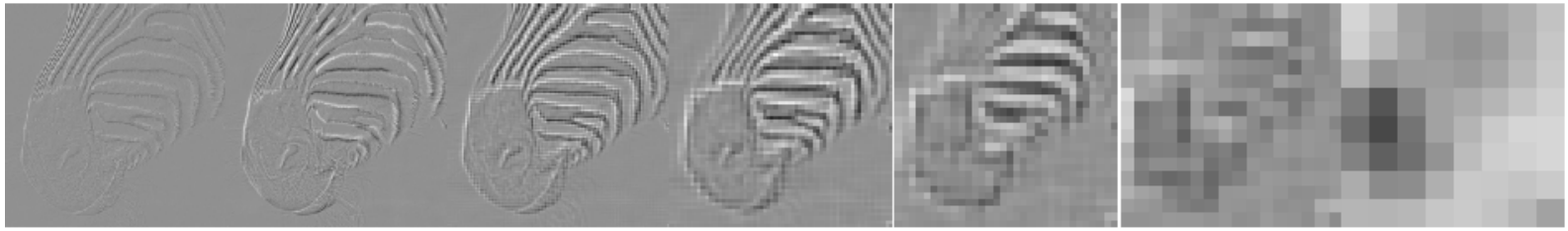
- Smooth with gaussians, because
  - a gaussian*gaussian=another gaussian
- Synthesis
  - smooth and sample
- Analysis
  - take the top image
- Gaussians are low pass filters, so repn is redundant

512 256 128 64 32 16 8

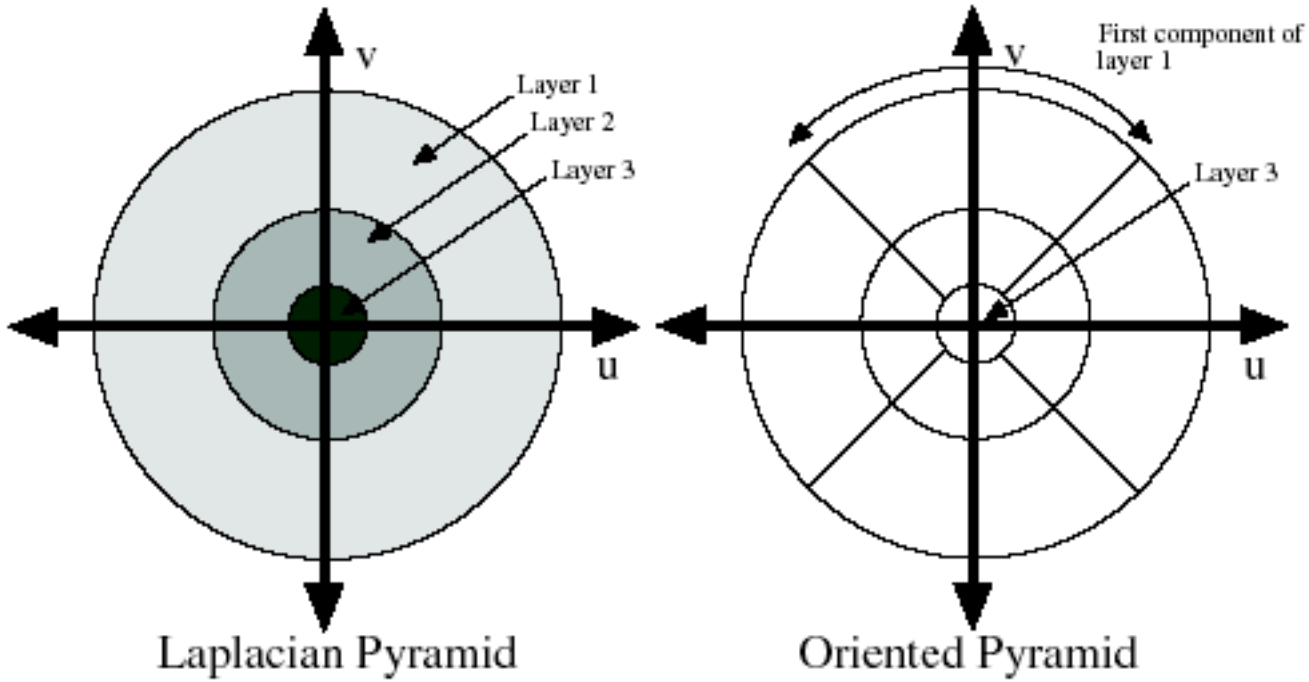
The Laplacian Pyramid

- Synthesis
  - preserve difference between upsampled Gaussian pyramid level and Gaussian pyramid level
  - band pass filter - each level represents spatial frequencies (largely) unrepresented at other levels
- Analysis
  - reconstruct Gaussian pyramid, take top layer

512 256 128 64 32 16 8

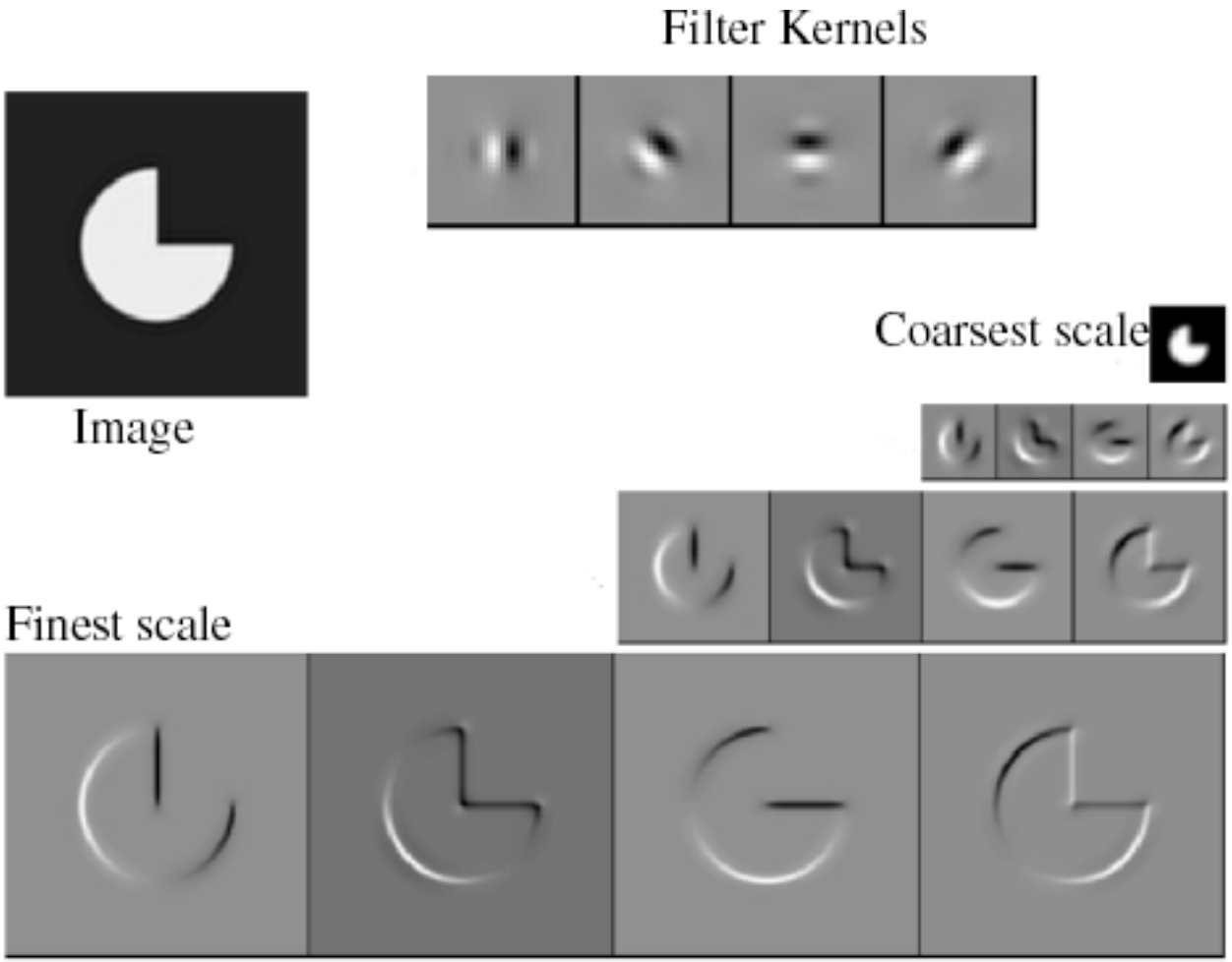


512 256 128 64 32 16 8


v
Layer 1
Layer 2
Layer 3
u
Laplacian Pyramid
v
First component of layer 1
Layer 3
u
Oriented Pyramid

Oriented pyramids

- Laplacian pyramid is orientation independent
- Apply an oriented filter to determine orientations at each layer
  - by clever filter design, we can simplify synthesis
  - this represents image information at a particular scale and orientation



Reprinted from "Shiftable MultiScale Transforms," by Simoncelli et al., IEEE Transactions on Information Theory, 1992, copyright 1992, IEEE