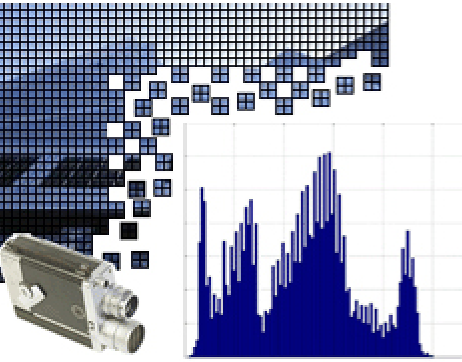
Edit View Favorites Tools Help

Back Search Favorites Media

ess http://courses.csail.mit.edu/6.869/ Go Lin



6.869 Advances in Computer Vision: Learning and Interfaces

Spring 2005

Tuesday and Thursday; 2:30 to 4:00pm in 36-153

Announcements

Course Information

- Syllabus
- Problem Sets and Exams
- Grading and Requirements
- Internet Resources

Contacts

http://courses.csail.mit.edu/6.869

Edit View Favorites Tools Help

Back Search Favorites Media

http://courses.csail.mit.edu/6.869/syllabus.html Go

Course Calendar

Lecture	Date	Description	Readings	Assignments	Materials
1	2/1	Course Introduction Cameras and Lenses	Req: FP 1.1, 2.1, 2.2, 2.3, 3.1, 3.2	PS0 out	
2	2/3	Image Filtering	Req: FP 7.1 - 7.6		
3	2/8	Image Representations: Pyramids	Req: FP 7.7, 9.2		
4	2/10	Image Statistics		PS0 due	
5	2/15	Texture	Req: FP 9.1, 9.3, 9.4	PS1 out	
6	2/17	Color	Req: FP 6.1-6.4		
7	2/22	Guest Lecture: Context in vision			
8	2/24	Guest Lecture: Medical Imaging		PS1 due	
9	3/1	Multiview Geometry	Req: Mikolajczyk and Schmid; FP 10	PS2 out	
10	3/3	Local Features	Req: Shi and Tomasi; Lowe		

Edit View Favorites Tools Help

Back Search Favorites Media

http://courses.csail.mit.edu/6.869/syllabus.html Go

Course Calendar

Today

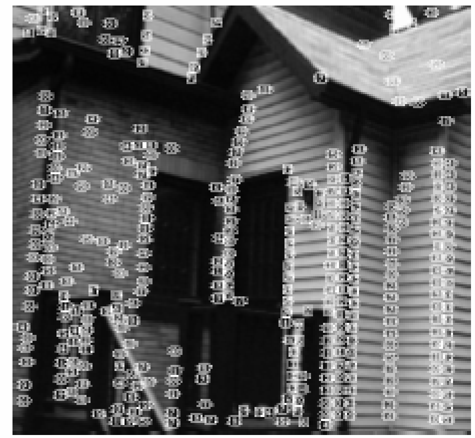
Lecture	Date	Description	Readings	Assignments	Materials
1	2/1	Course Introduction Cameras and Lenses	Req: FP 1.1, 2.1, 2.2, 2.3, 3.1, 3.2	PS0 out	
2	2/3	Image Filtering	Req: FP 7.1 - 7.6		

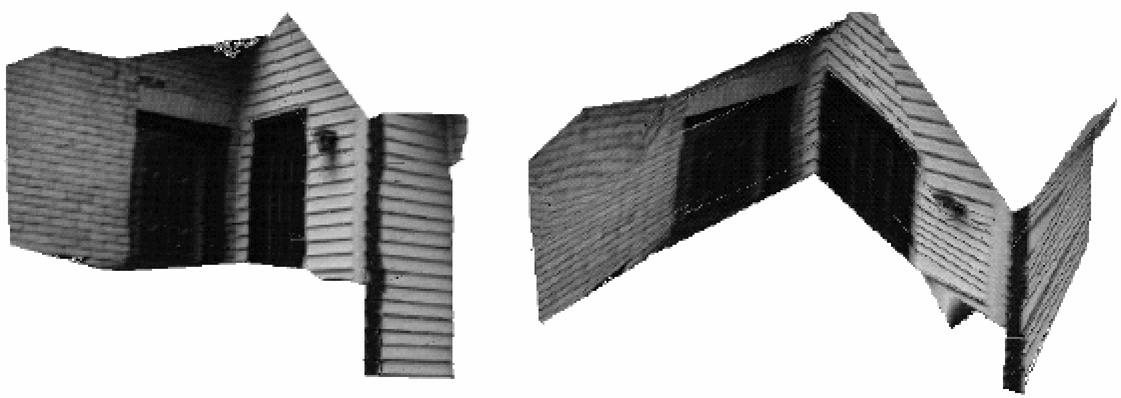
Motivation for camera calibration:

relating image measurements to positions out in the world

Frames from video data

Tracked feature points



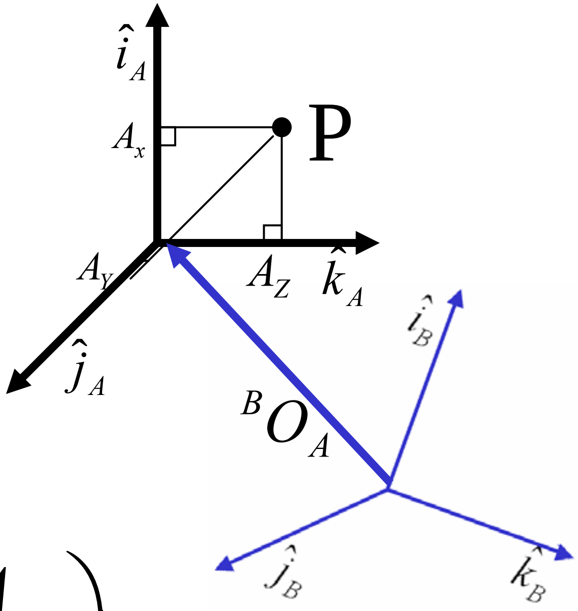


Inferred 3-d shape of building


slow

Translation and rotation

Let's write $^{B}P = {}^{B}_{A}R \; {}^{A}P + {}^{B}O_{A}$



as a single matrix equation:

$$\begin{pmatrix} B_X \\ B_Y \\ B_Z \\ 1 \end{pmatrix} = \begin{pmatrix} \boxed{\begin{matrix} - & - & - \\ - & {}^{B}_{A}R & - \\ - & - & - \end{matrix}} & \boxed{\begin{matrix} | \\ {}^{B}O_{A} \\ | \end{matrix}} \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix} \begin{pmatrix} A_X \\ A_Y \\ A_Z \\ 1 \end{pmatrix}$$

Homogenous coordinates

- Add an extra coordinate and use an equivalence relation
- for 3D
  - equivalence relation k*(X,Y,Z,T) is the same as (X,Y,Z,T)

- Motivation
  - Possible to write the action of a perspective camera as a matrix

Homogenous/non-homogenous transformations for a 3-d point

- From non-homogenous to homogenous coordinates: add 1 as the 4th coordinate, ie

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- From homogenous to non-homogenous coordinates: divide 1st 3 coordinates by the 4th, ie

$$\begin{pmatrix} x \\ y \\ z \\ T \end{pmatrix} \rightarrow \frac{1}{T}\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Homogenous/non-homogenous transformations for a 2-d point

- From non-homogenous to homogenous coordinates: add 1 as the 3rd coordinate, ie

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- From homogenous to non-homogenous coordinates: divide 1st 2 coordinates by the 3rd, ie

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \frac{1}{z}\begin{pmatrix} x \\ y \end{pmatrix}$$

Translation and rotation, written in each set of coordinates

Non-homogeneous coordinates

$$^{B}P = {}^{B}_{A}R \; {}^{A}P + {}^{B}O_{A}$$

Homogeneous coordinates

$$^{B}P = {}^{B}_{A}C \; {}^{A}P$$

where

$$C = \begin{pmatrix} \boxed{\begin{matrix} - & - & - \\ - & {}^{B}_{A}R & - \\ - & - & - \end{matrix}} & \boxed{\begin{matrix} | \\ {}^{B}O_{A} \\ | \end{matrix}} \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix}$$

Perspective projection, in homogenous coordinates

- Turn previous expression into HC's
  - HC's for 3D point are (X,Y,Z,T)
  - HC's for point in image are (U,V,W)

$$\begin{pmatrix} X \\ Y \\ \frac{Z}{f} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ \frac{Z}{f} \end{pmatrix} \to \frac{f}{Z} \begin{pmatrix} X \\ Y \end{pmatrix}$$

HC Non-HC

The projection matrix for orthographic projection, in homogenous coordinates

$$\begin{pmatrix} U \\ V \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} X \\ Y \end{pmatrix}$$

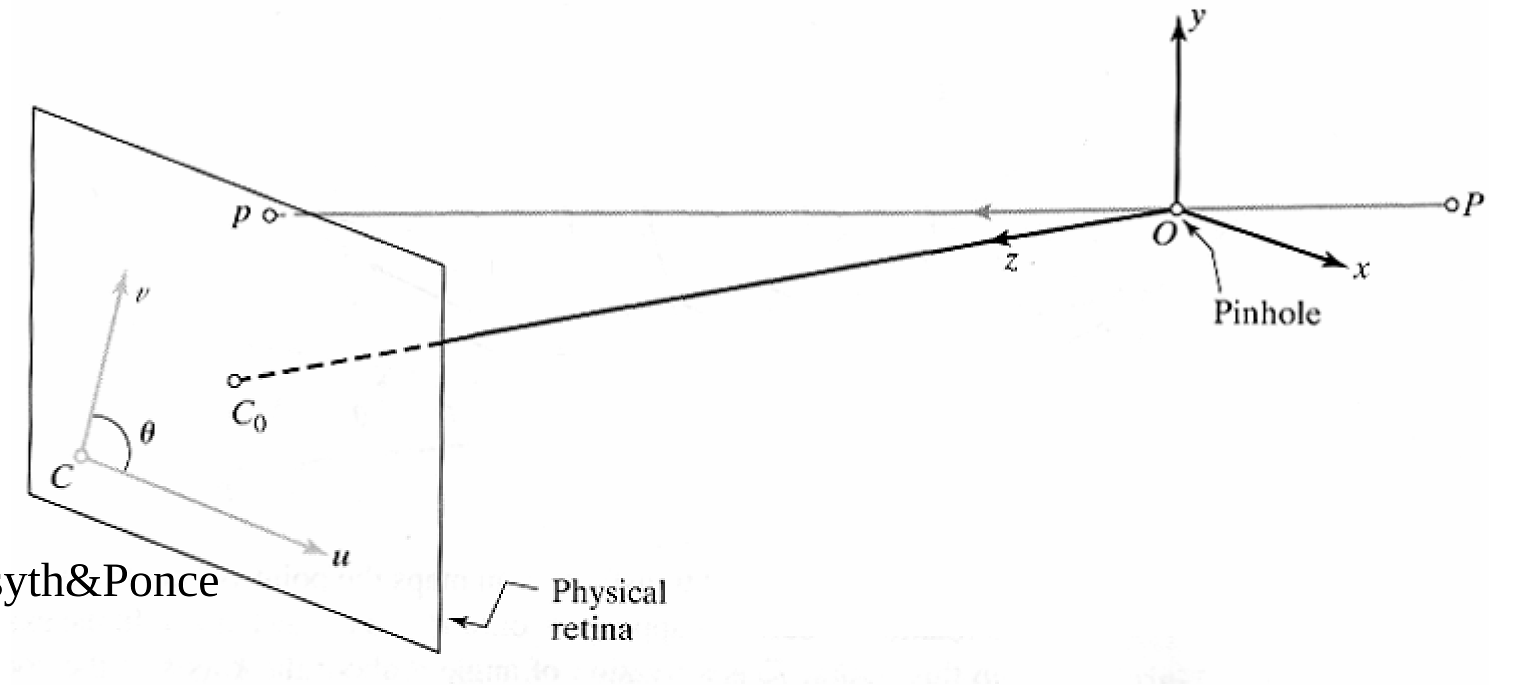
HC Non-HC

Camera calibration

Use the camera to tell you things about the world:

- Relationship between coordinates in the world and coordinates in the image: *geometric camera calibration.*
- (Relationship between intensities in the world and intensities in the image: *photometric camera calibration*, not covered in this course, see 6.801 or text)

Intrinsic parameters: from idealized world coordinates to pixel values



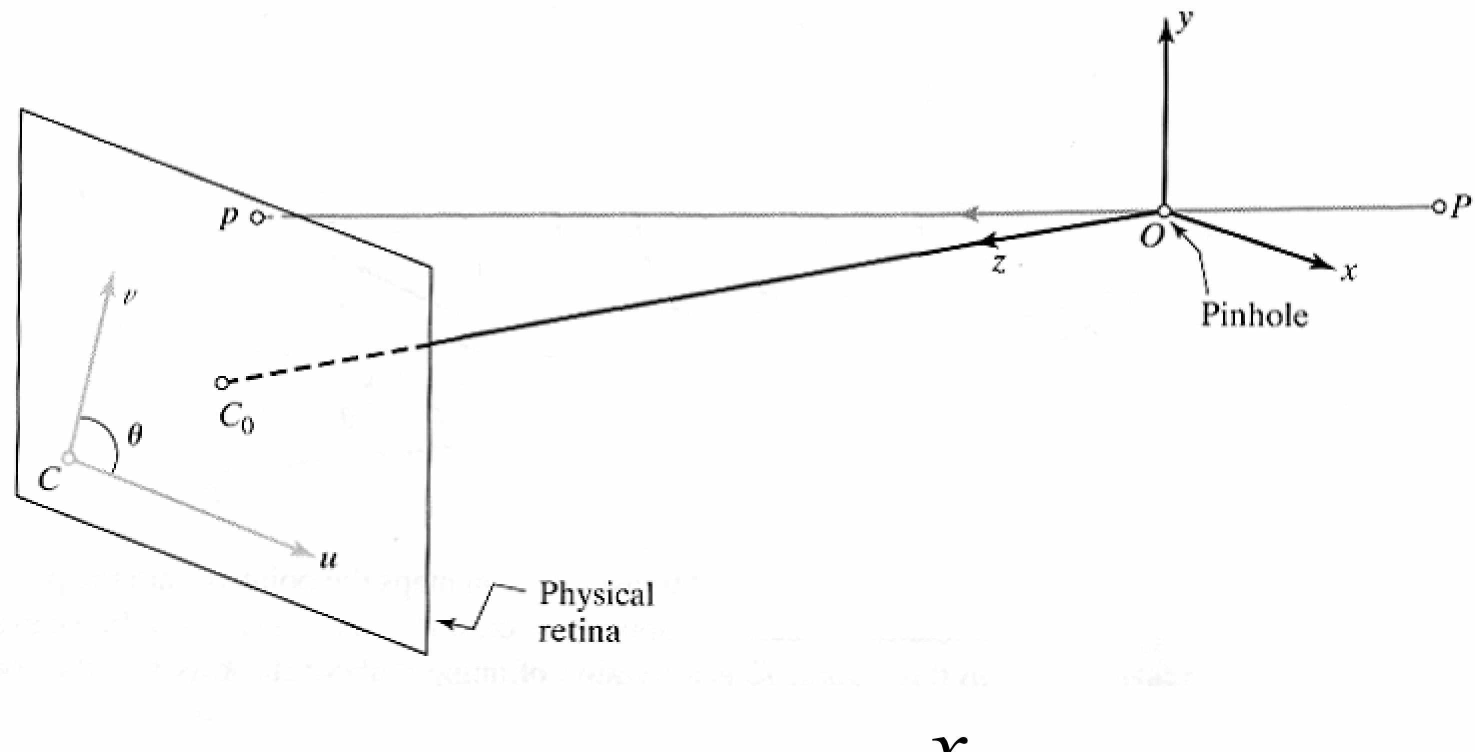
Forsyth&Ponce

Perspective projection

$$u = f\frac{x}{z}$$

$$v = f\frac{y}{z}$$

Intrinsic parameters

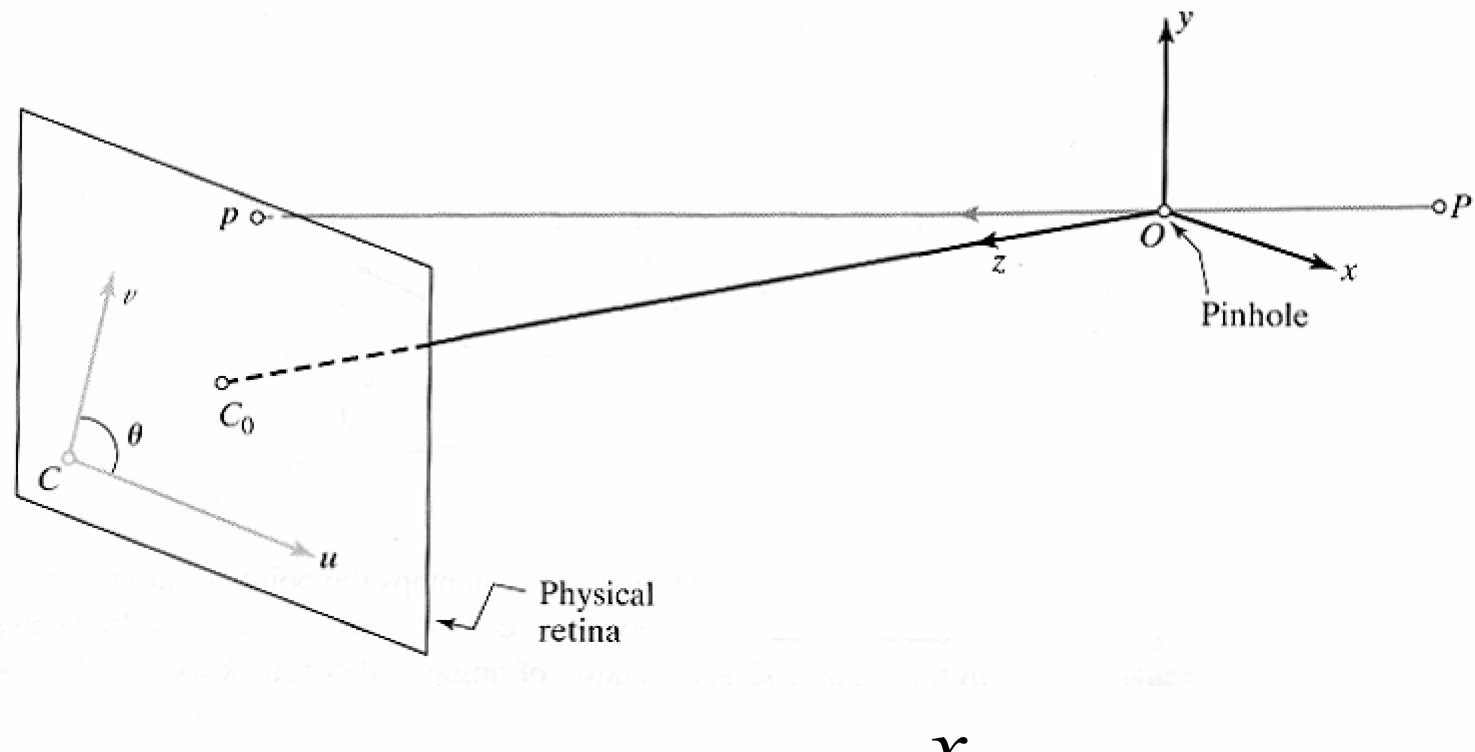


But “pixels” are in some arbitrary spatial units

$$u = \alpha \frac{x}{z}$$

$$v = \alpha \frac{y}{z}$$

Intrinsic parameters

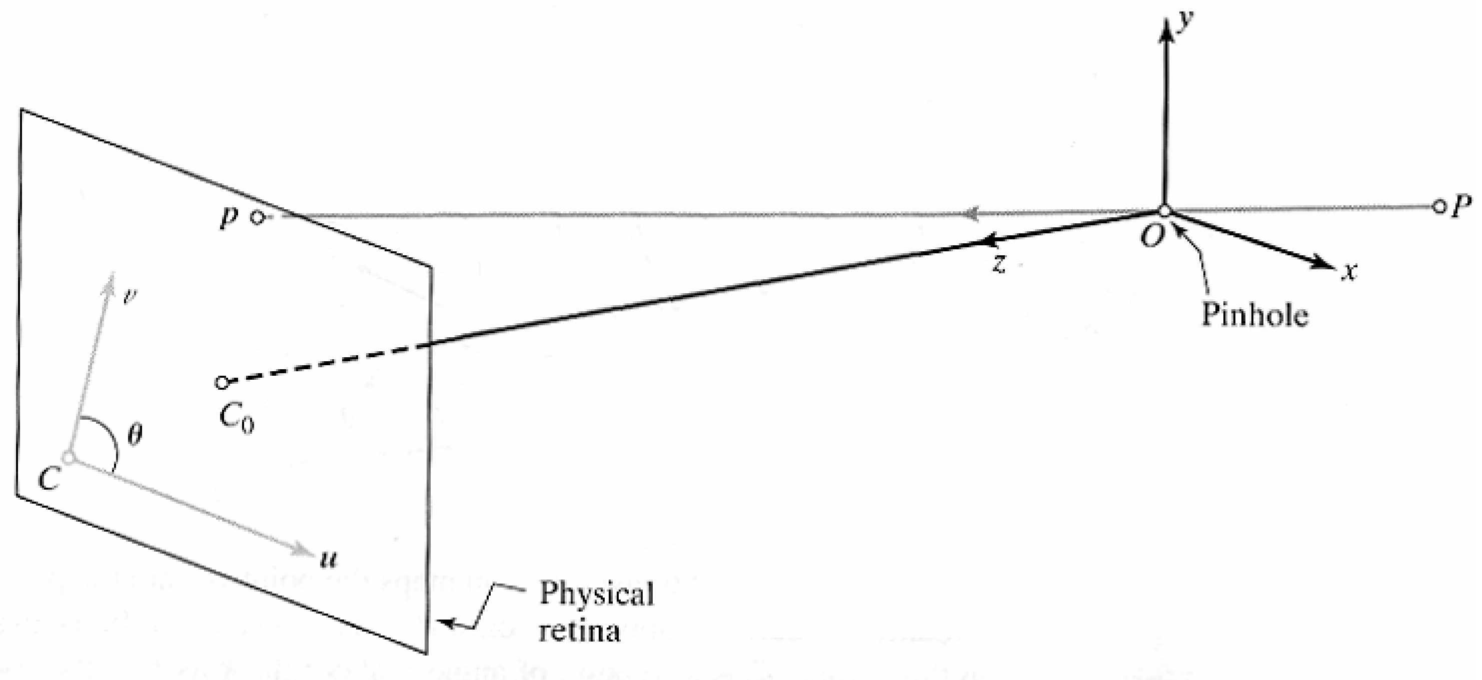


Maybe pixels are not square

$$u = \alpha \frac{x}{z}$$

$$v = \beta \frac{y}{z}$$

Intrinsic parameters

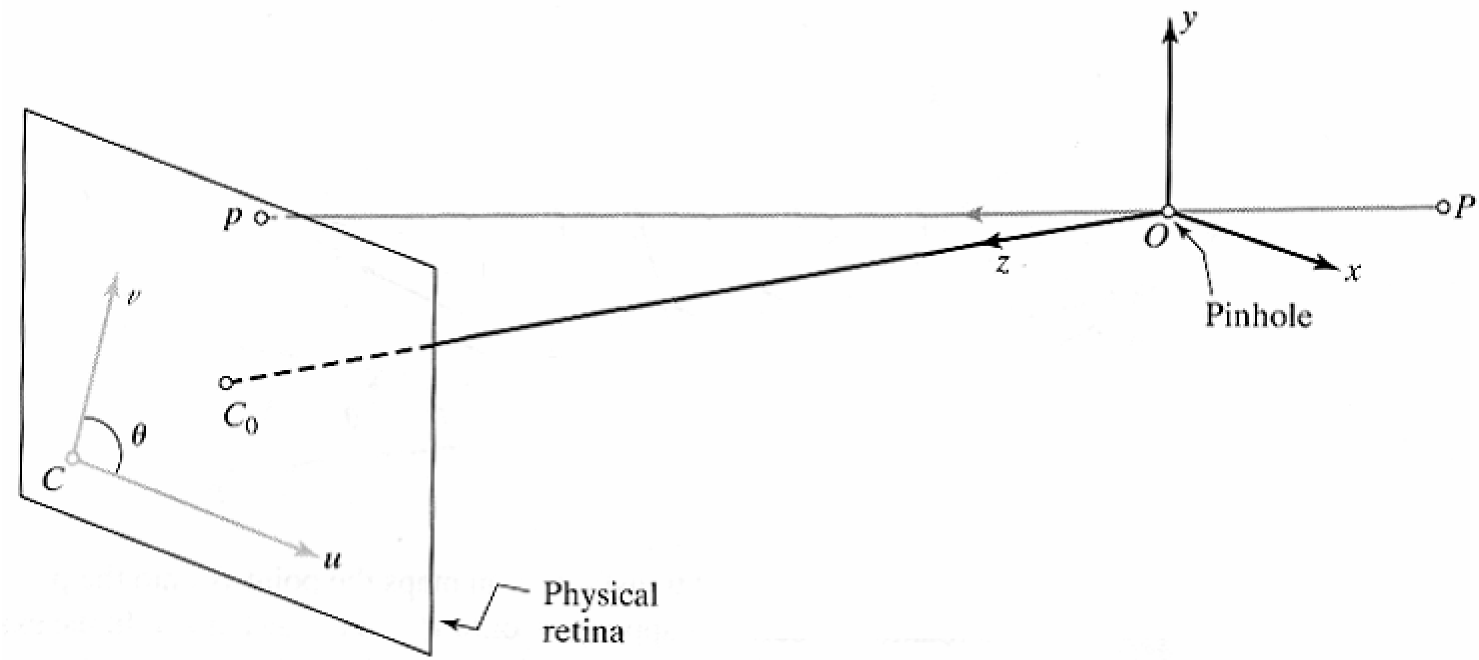


We don’t know the origin of our camera pixel coordinates

$$u = \alpha \frac{x}{z} + u_0$$

$$v = \beta \frac{y}{z} + v_0$$

Intrinsic parameters

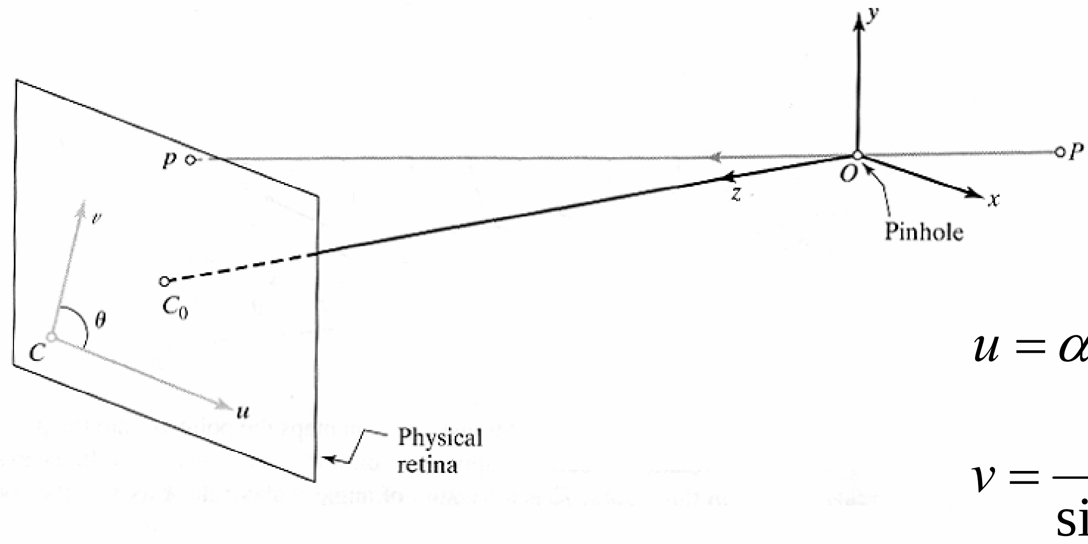


May be skew between camera pixel axes

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

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

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:

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

Extrinsic parameters: translation and rotation of camera frame

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

Non-homogeneous coordinates

$$\begin{pmatrix} {}^{C}\vec{P} \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \boxed{\begin{matrix} - & - & - \\ - & {}^{C}_{W}R & - \\ - & - & - \end{matrix}} & \boxed{\begin{matrix} | \\ {}^{C}O_{W} \\ | \end{matrix}} \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix} \begin{pmatrix} {}^{W}\vec{P} \\ \\ \\ \end{pmatrix}$$

Homogeneous coordinates

Combining extrinsic and intrinsic calibration parameters

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

Intrinsic

$$\begin{pmatrix} \\ {}^{C}\vec{P} \\ \\ \end{pmatrix} = \begin{pmatrix} \boxed{\begin{matrix} - & - & - \\ - & {}^{C}_{W}R & - \\ - & - & - \end{matrix}} & \boxed{\begin{matrix} | \\ {}^{C}O_{W} \\ | \end{matrix}} \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix} \begin{pmatrix} \\ \vec{P} \\ \\ \end{pmatrix}$$

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

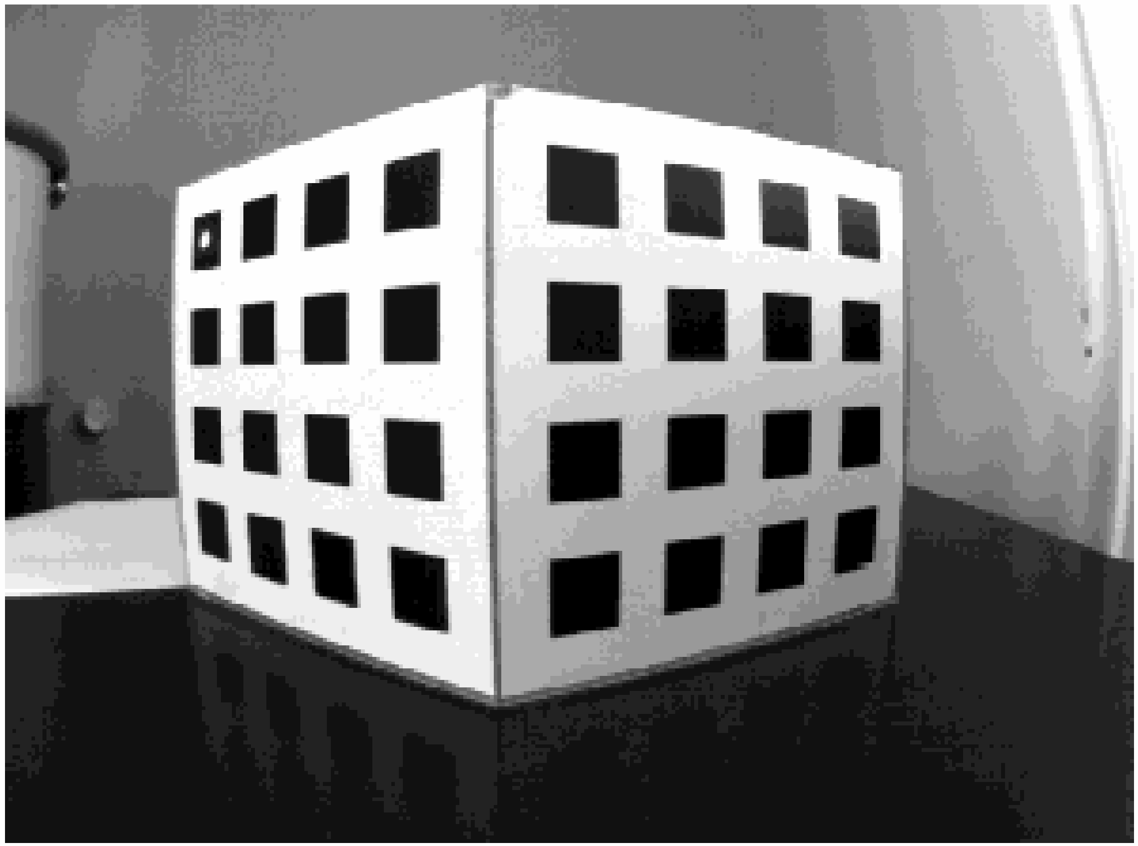
$$\vec{p} = \frac{1}{z} M \vec{P}$$

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

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

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

Calibration target



The Opti-CAL Calibration Target Image

http://www.kinetic.bc.ca/CompVision/opti-CAL.html

Camera calibration

From before, we had these equations relating image positions, u,v, to points at 3-d positions P (in homogeneous coordinates):

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

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

So for each feature point, i, we have:

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

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

Camera calibration

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:

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

Camera calibration

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

Showing all the elements:
$$\begin{pmatrix} P_{1x} & P_{1y} & P_{1z} & 1 & 0 & 0 & 0 & 0 & -u_1 P_{1x} & -u_1 P_{1y} & -u_1 P_{1z} & -u_1 \\ 0 & 0 & 0 & 0 & P_{1x} & P_{1y} & P_{1z} & 1 & -v_1 P_{1x} & -v_1 P_{1y} & -v_1 P_{1z} & -v_1 \\ & & & & & & \cdots & \cdots & \cdots & & & \\ P_{nx} & P_{ny} & P_{nz} & 1 & 0 & 0 & 0 & 0 & -u_n P_{nx} & -u_n P_{ny} & -u_n P_{nz} & -u_n \\ 0 & 0 & 0 & 0 & P_{nx} & P_{ny} & P_{nz} & 1 & -v_n P_{nx} & -v_n P_{ny} & -v_n P_{nz} & -v_n \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

Camera calibration

$$\begin{pmatrix} P_{1x} & P_{1y} & P_{1z} & 1 & 0 & 0 & 0 & 0 & -u_1P_{1x} & -u_1P_{1y} & -u_1P_{1z} & -u_1 \\ 0 & 0 & 0 & 0 & P_{1x} & P_{1y} & P_{1z} & 1 & -v_1P_{1x} & -v_1P_{1y} & -v_1P_{1z} & -v_1 \\ & & & & & \cdots & \cdots & \cdots & & & & \\ P_{nx} & P_{ny} & P_{nz} & 1 & 0 & 0 & 0 & 0 & -u_nP_{nx} & -u_nP_{ny} & -u_nP_{nz} & -u_n \\ 0 & 0 & 0 & 0 & P_{nx} & P_{ny} & P_{nz} & 1 & -v_nP_{nx} & -v_nP_{ny} & -v_nP_{nz} & -v_n \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

Q m = 0

We want to solve for the unit vector m (the stacked one) that minimizes $|Qm|^2$

The minimum eigenvector of the matrix Q^TQ gives us that (see Forsyth&Ponce, 3.1)

Once you have the M matrix, can recover the intrinsic and extrinsic parameters as in Forsyth&Ponce, sect. 3.2.2.

$$\mathcal{M} = \begin{pmatrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T & \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \dfrac{\beta}{\sin\theta} \boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T & \dfrac{\beta}{\sin\theta} t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{pmatrix}$$

Image filtering

- Reading:
  - Chapter 7, F&P

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.

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

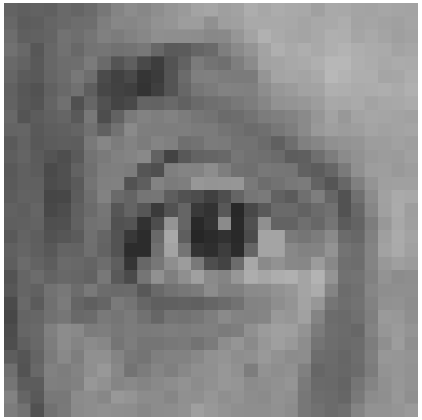
	7	

Modified image data

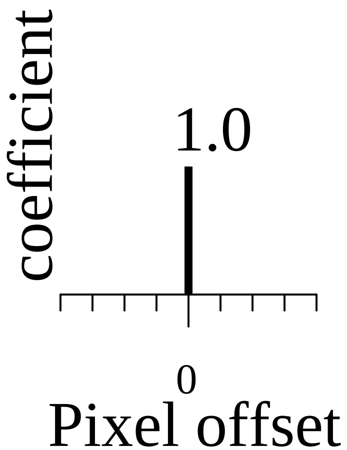
Convolution

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

Linear filtering (warm-up slide)

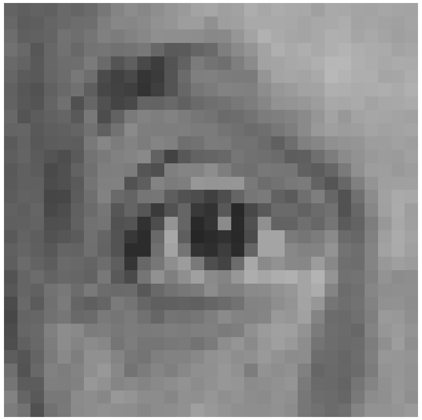


original

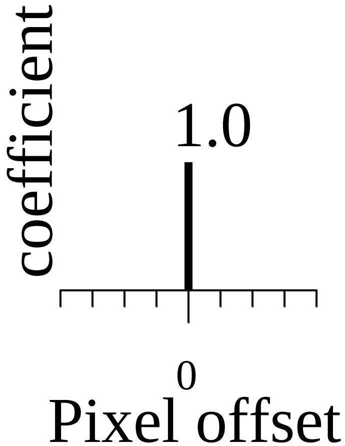


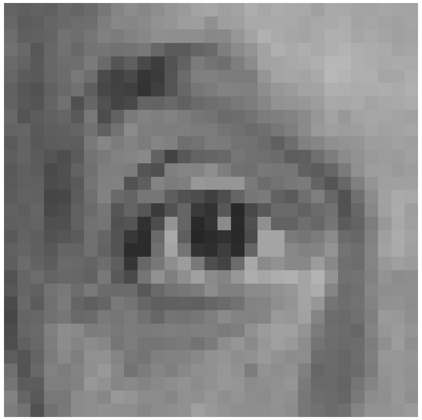
?

Linear filtering (warm-up slide)



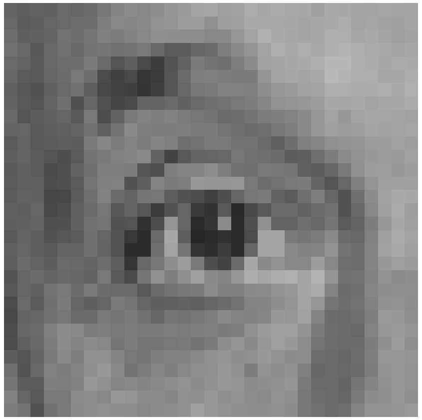
original



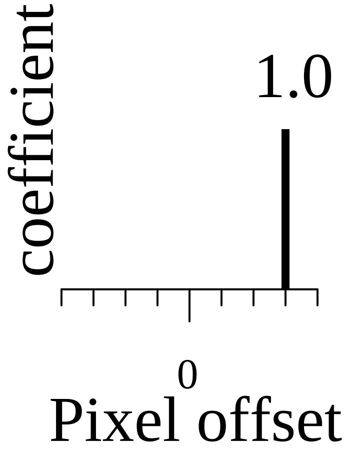


Filtered
(no change)

Linear filtering

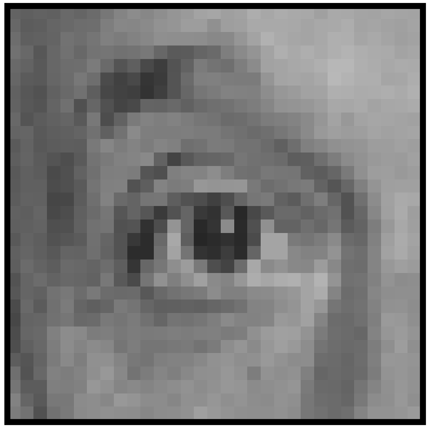


original

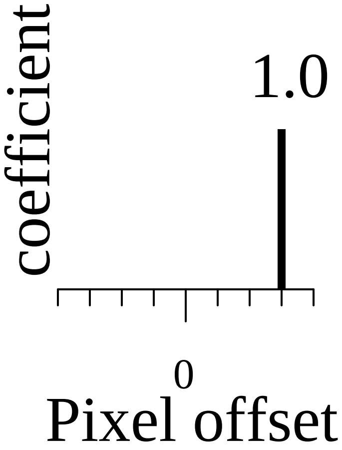


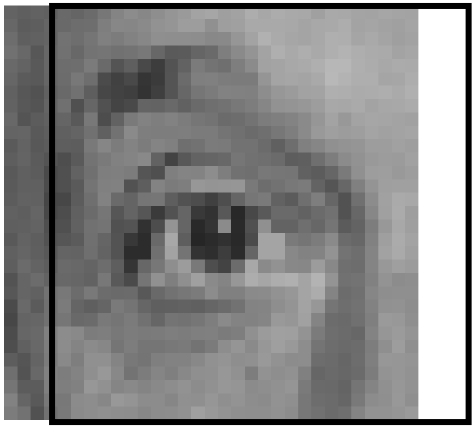
?

shift



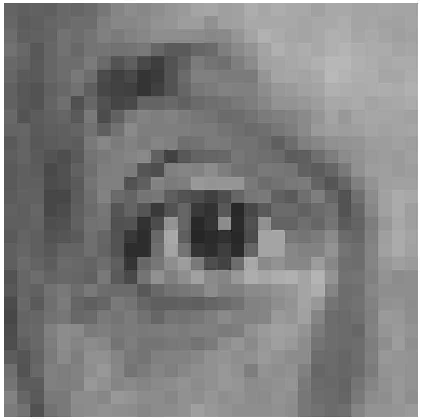
original



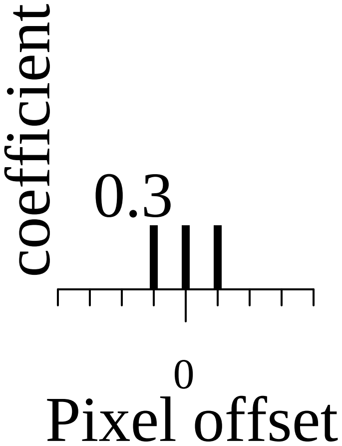


shifted

Linear filtering

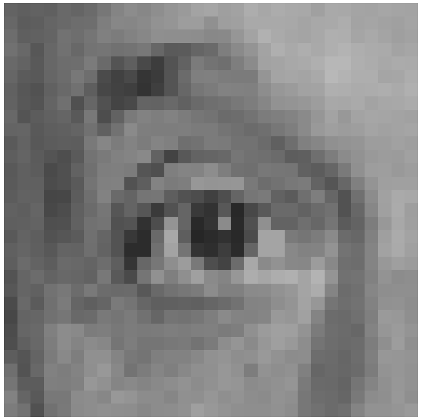


original



?

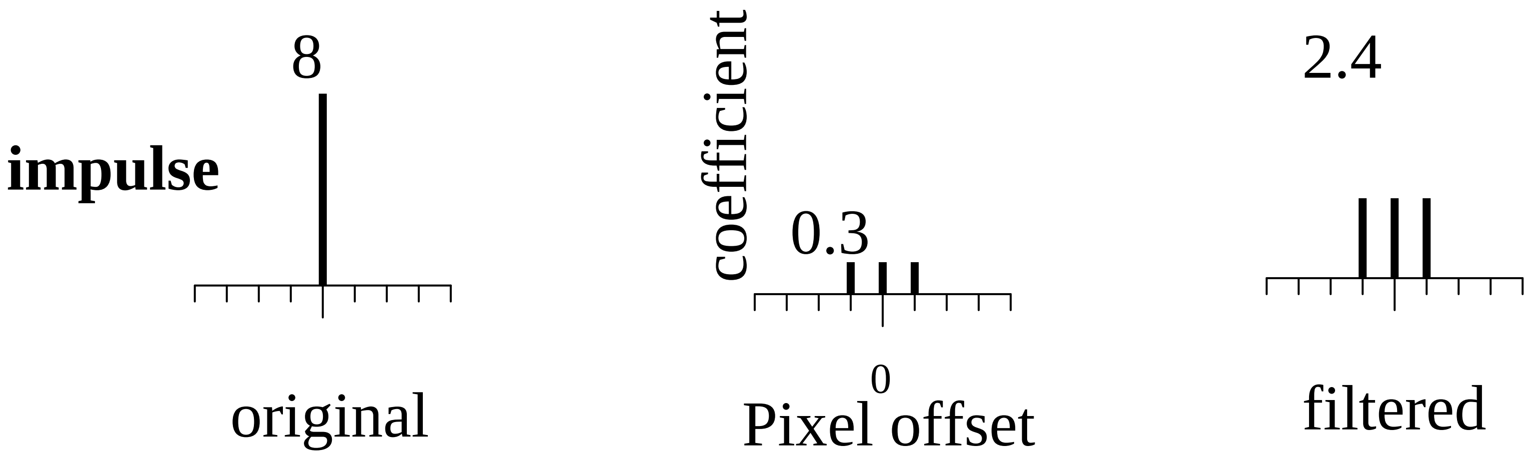
Blurring



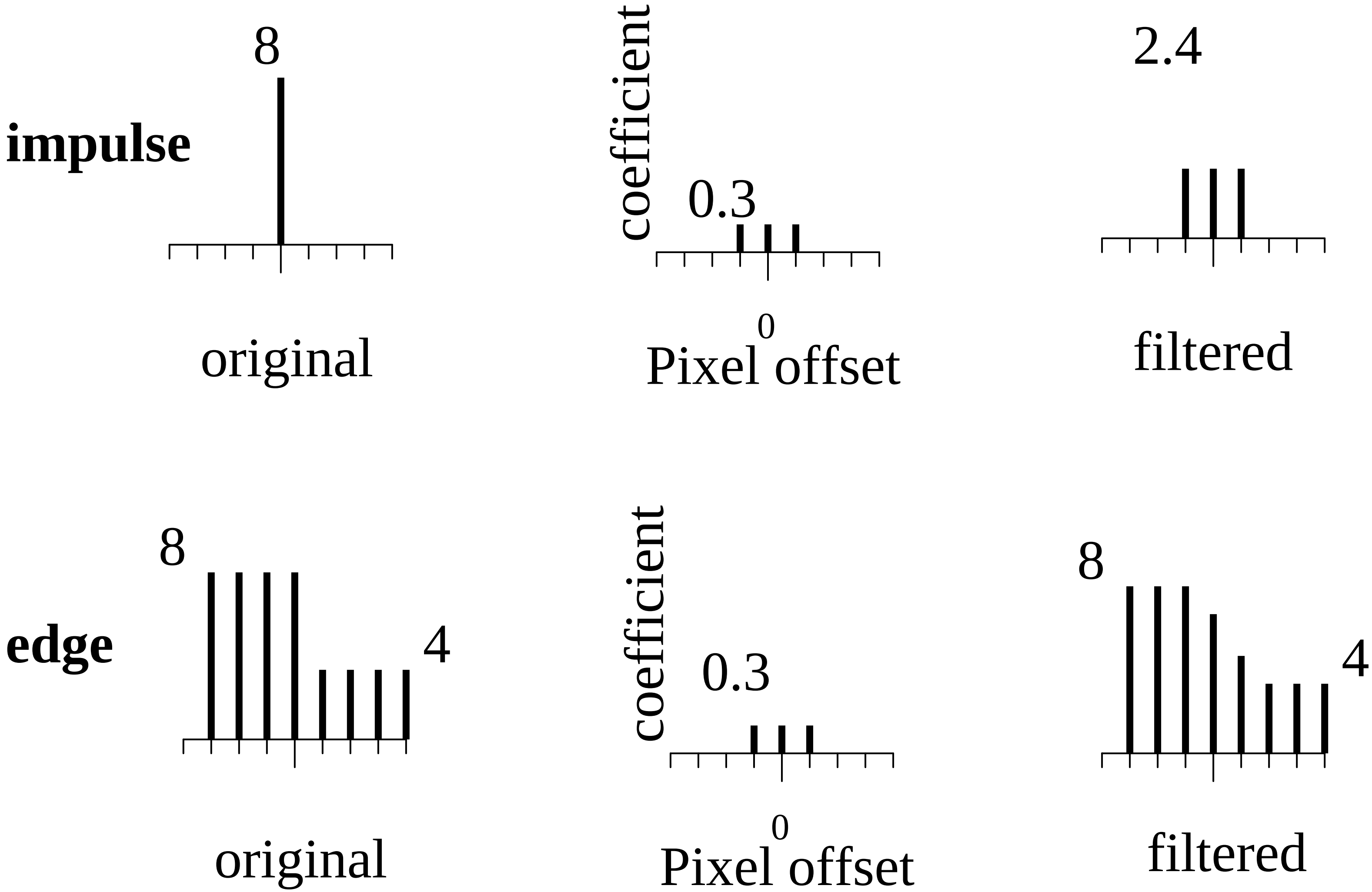
original

Blurred (filter applied in both dimensions).

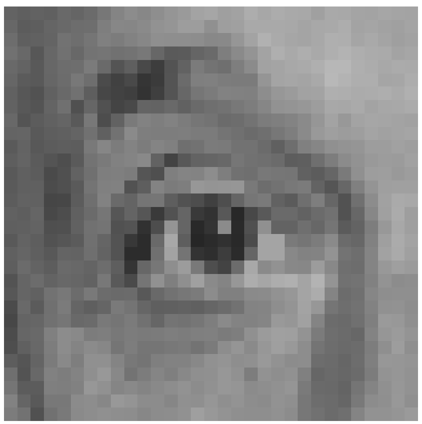
Blur examples



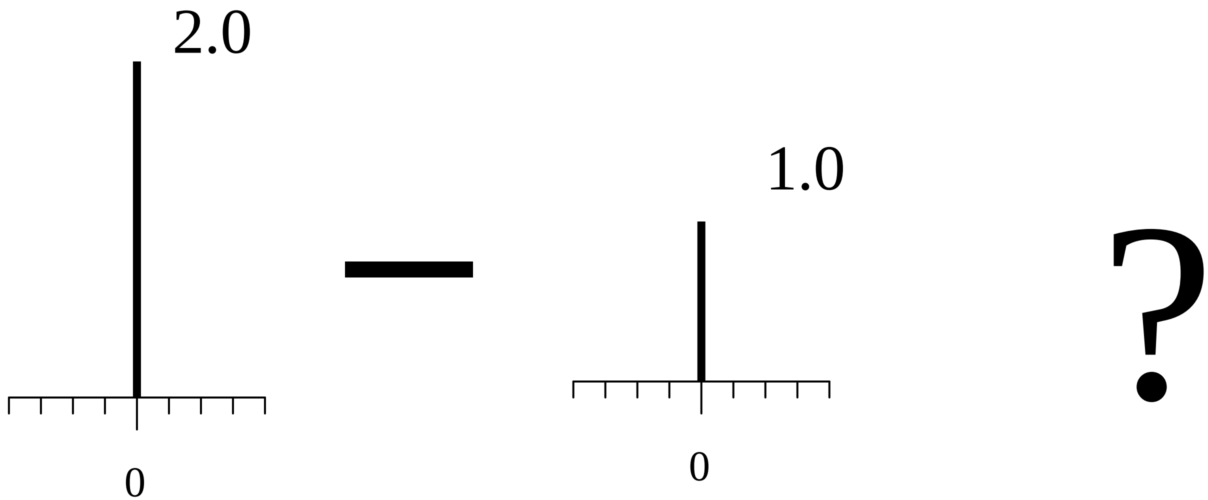
Blur examples



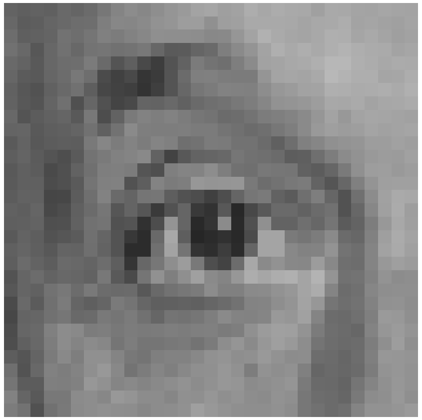
Linear filtering (warm-up slide)



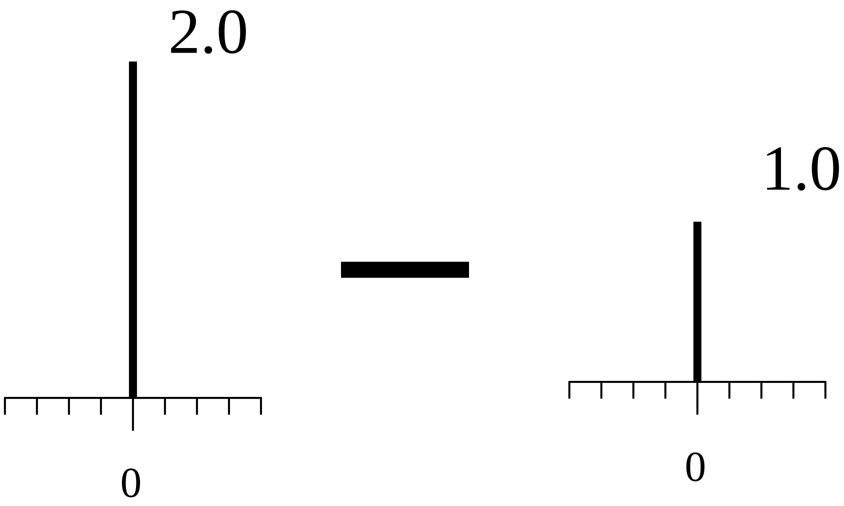
original

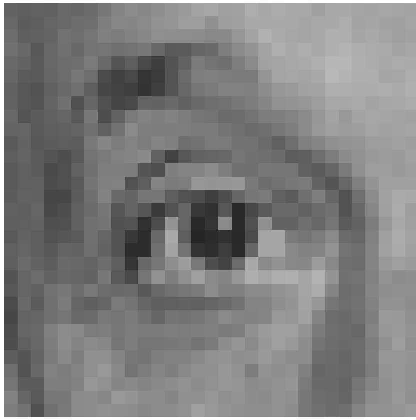


Linear filtering (no change)



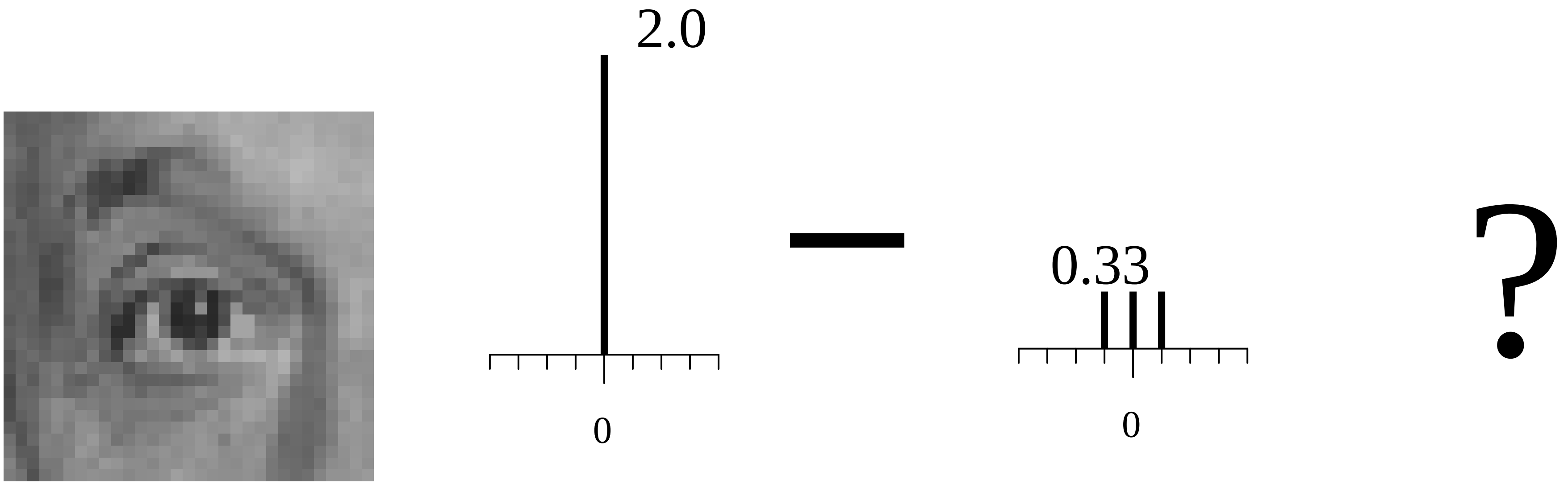
original





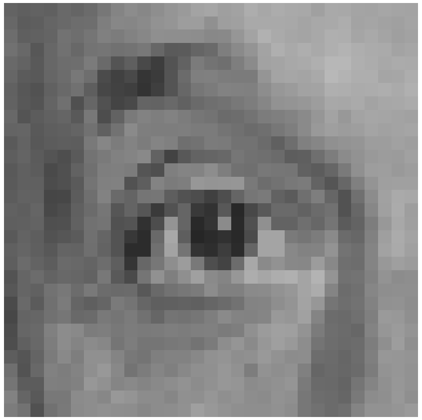
Filtered
(no change)

Linear filtering

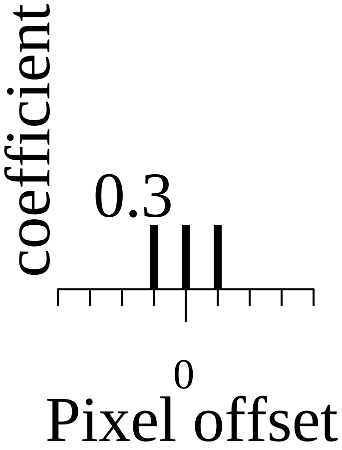


original

(remember blurring)

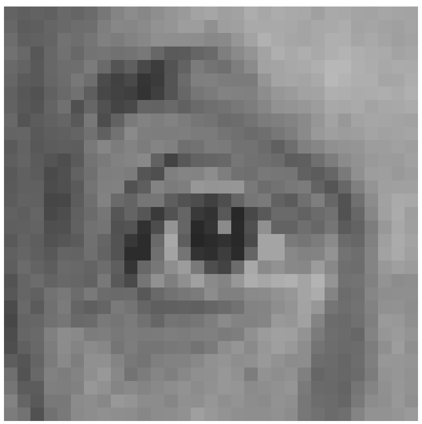


original

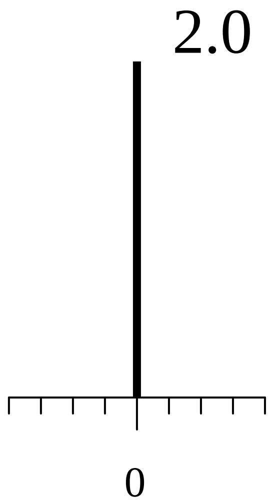


Blurred (filter applied in both dimensions).

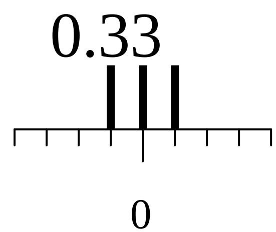
Sharpening

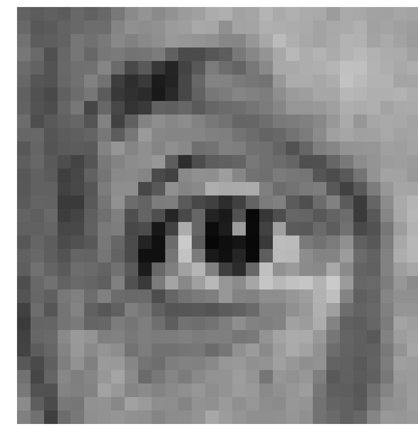


original



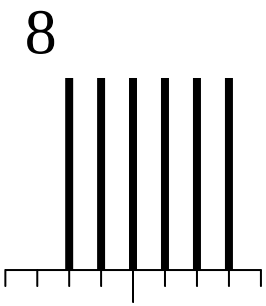
−



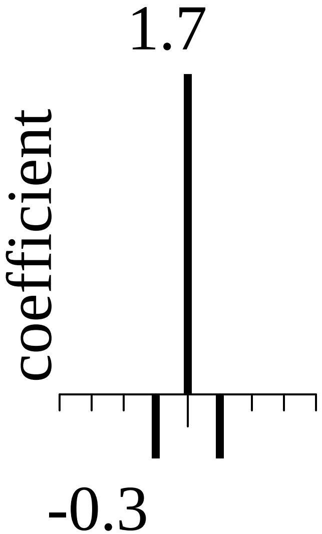


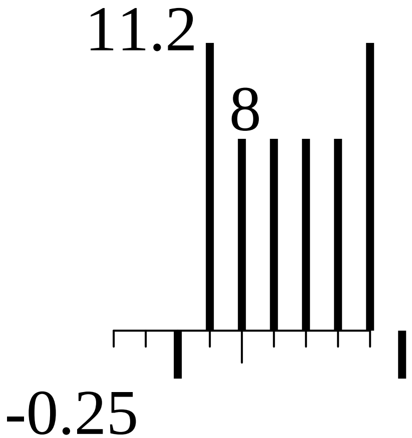
Sharpened
original

Sharpening example



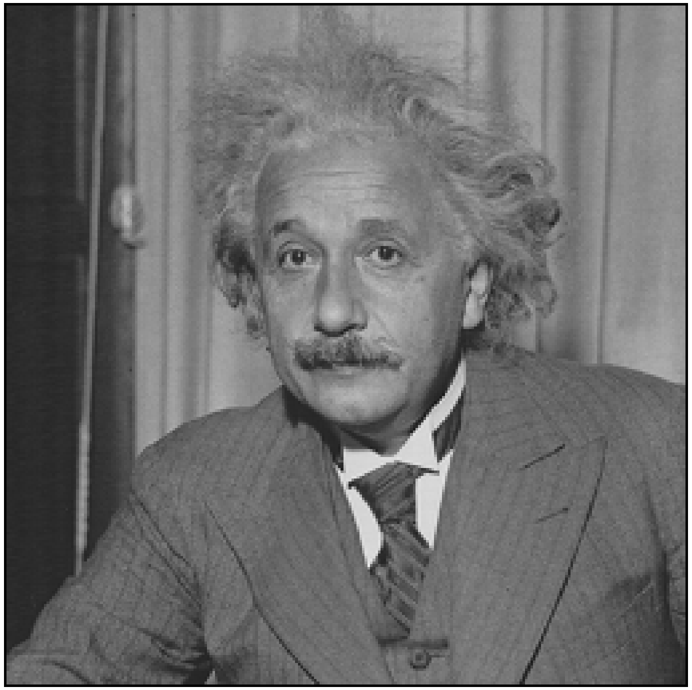
original



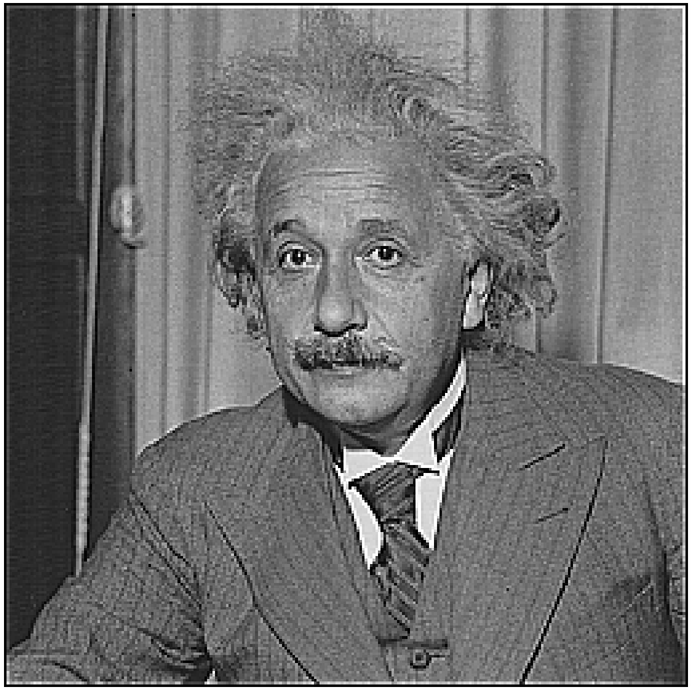


Sharpened
(differences are accentuated; constant areas are left untouched).

Sharpening

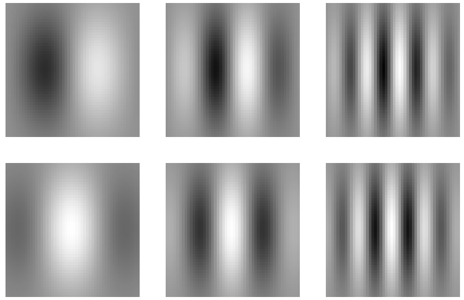


before



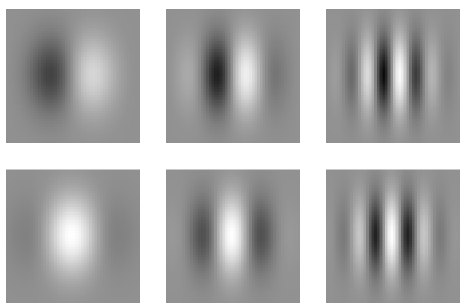
after

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.

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

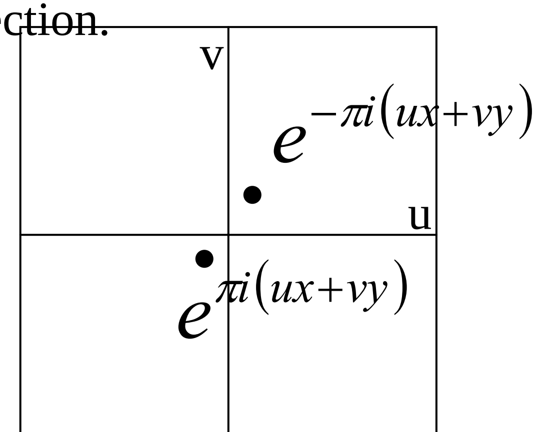
Forward transform

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

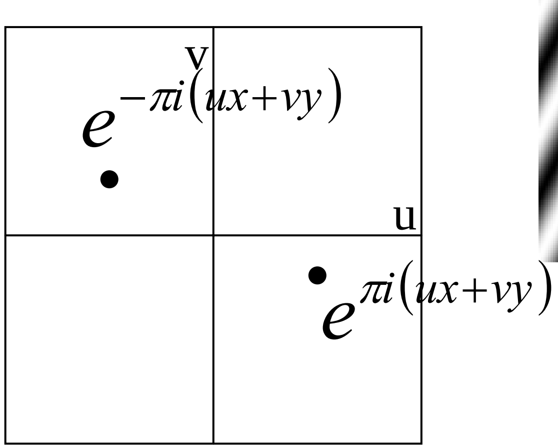
Inverse transform

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

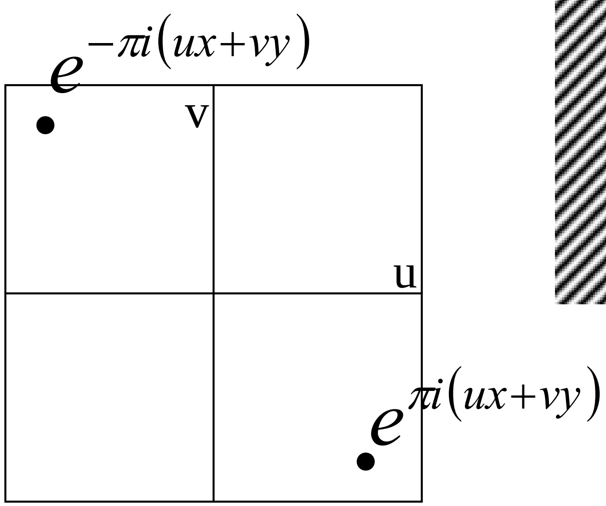
To get some sense of what basis elements look like, we plot a basis element --- or rather, its real part --- as a function of x,y for some fixed u, v. We get a function that is constant when (ux+vy) is constant. The magnitude of the vector (u, v) gives a frequency, and its direction gives an orientation.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

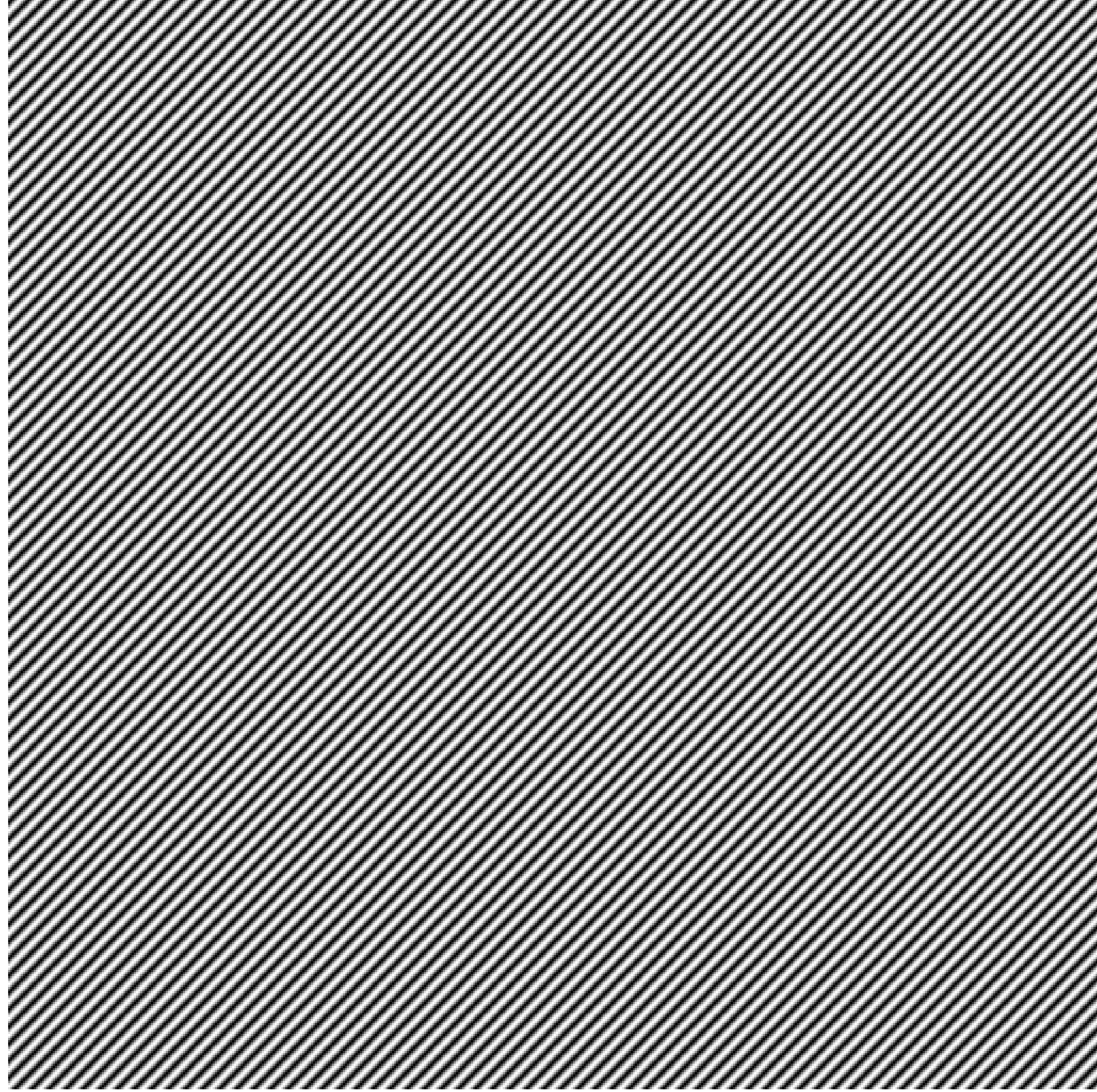


Here u and v are larger than in the previous slide.



And larger still...

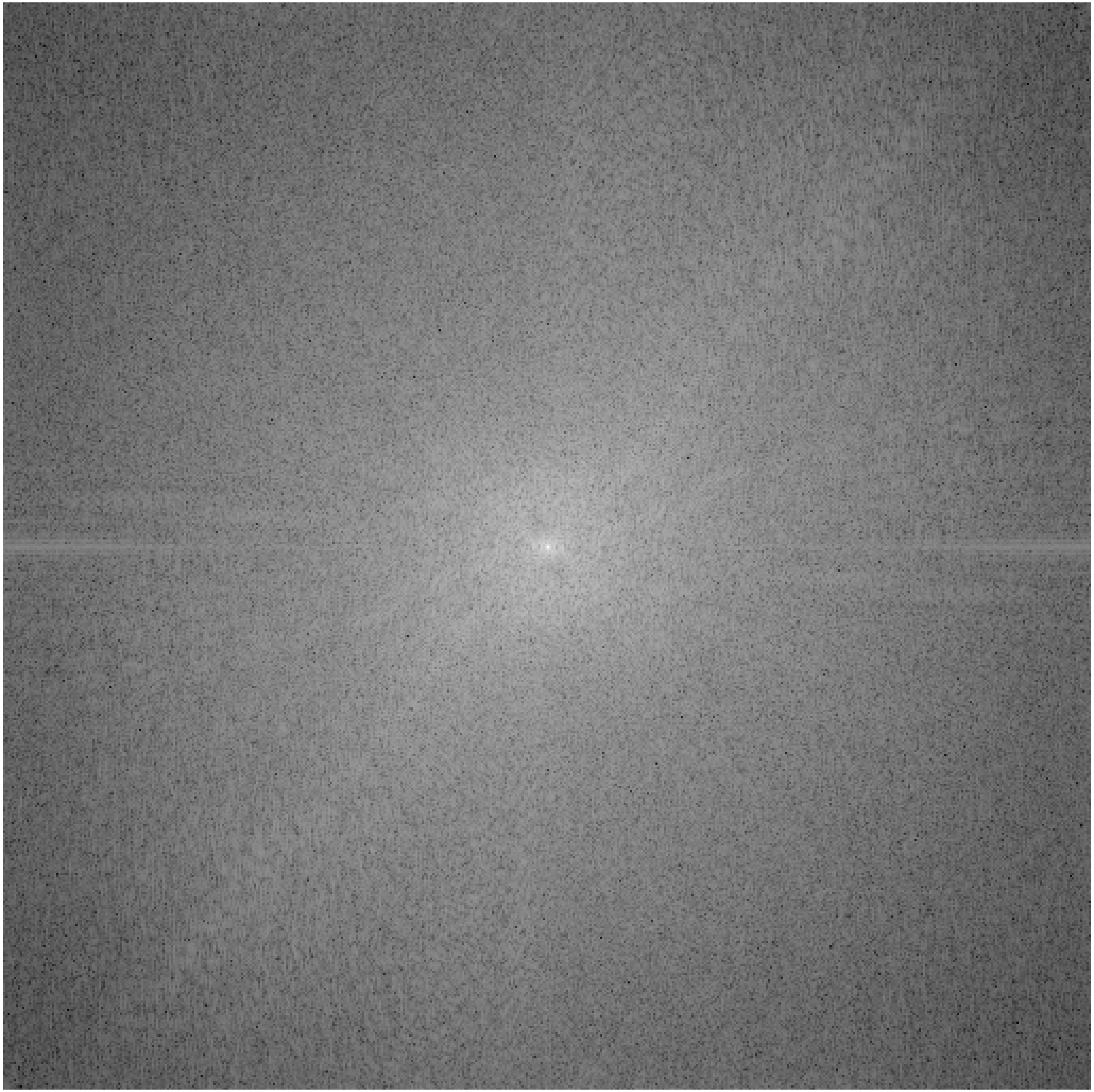




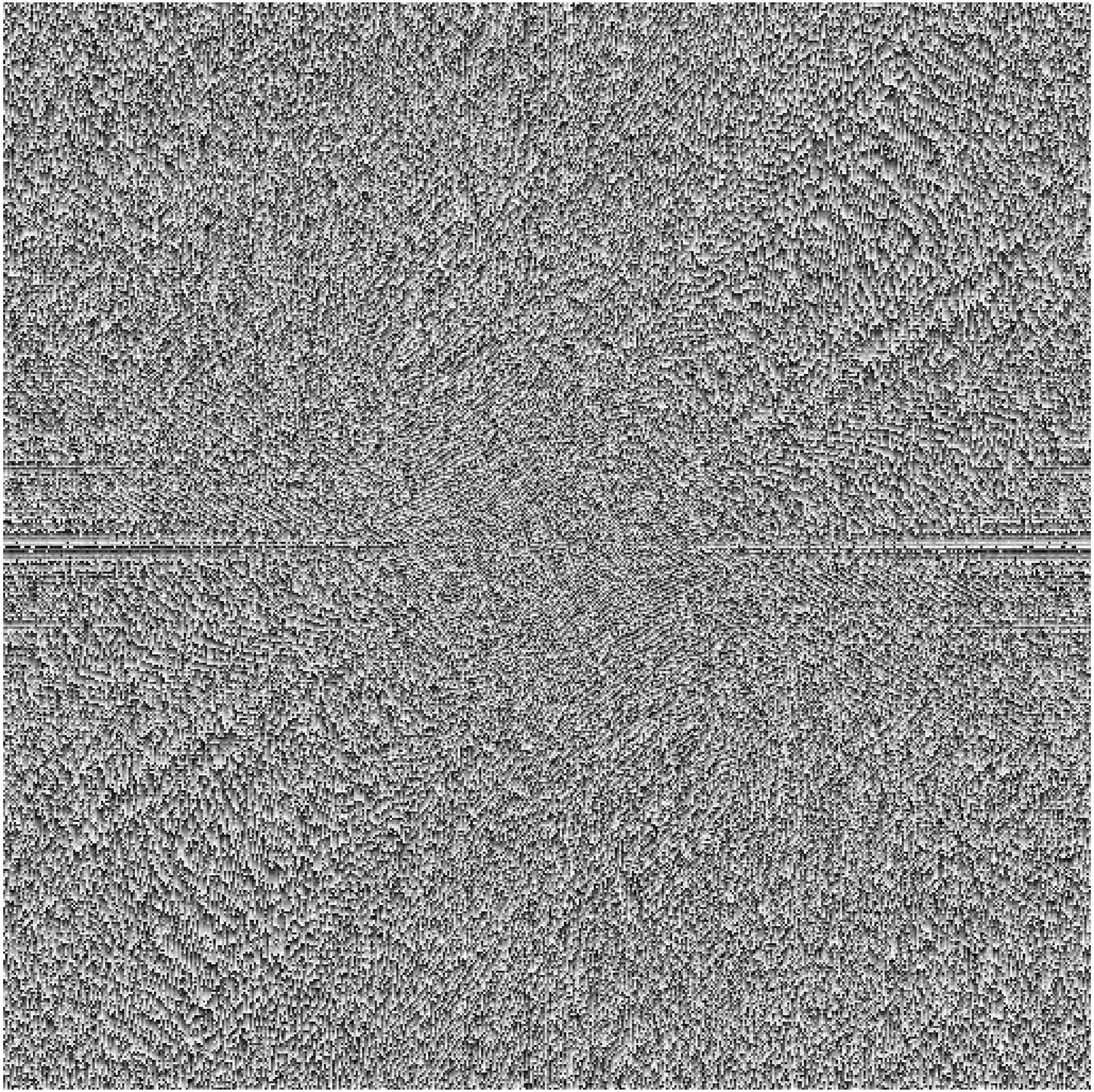
Phase and Magnitude

- 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?

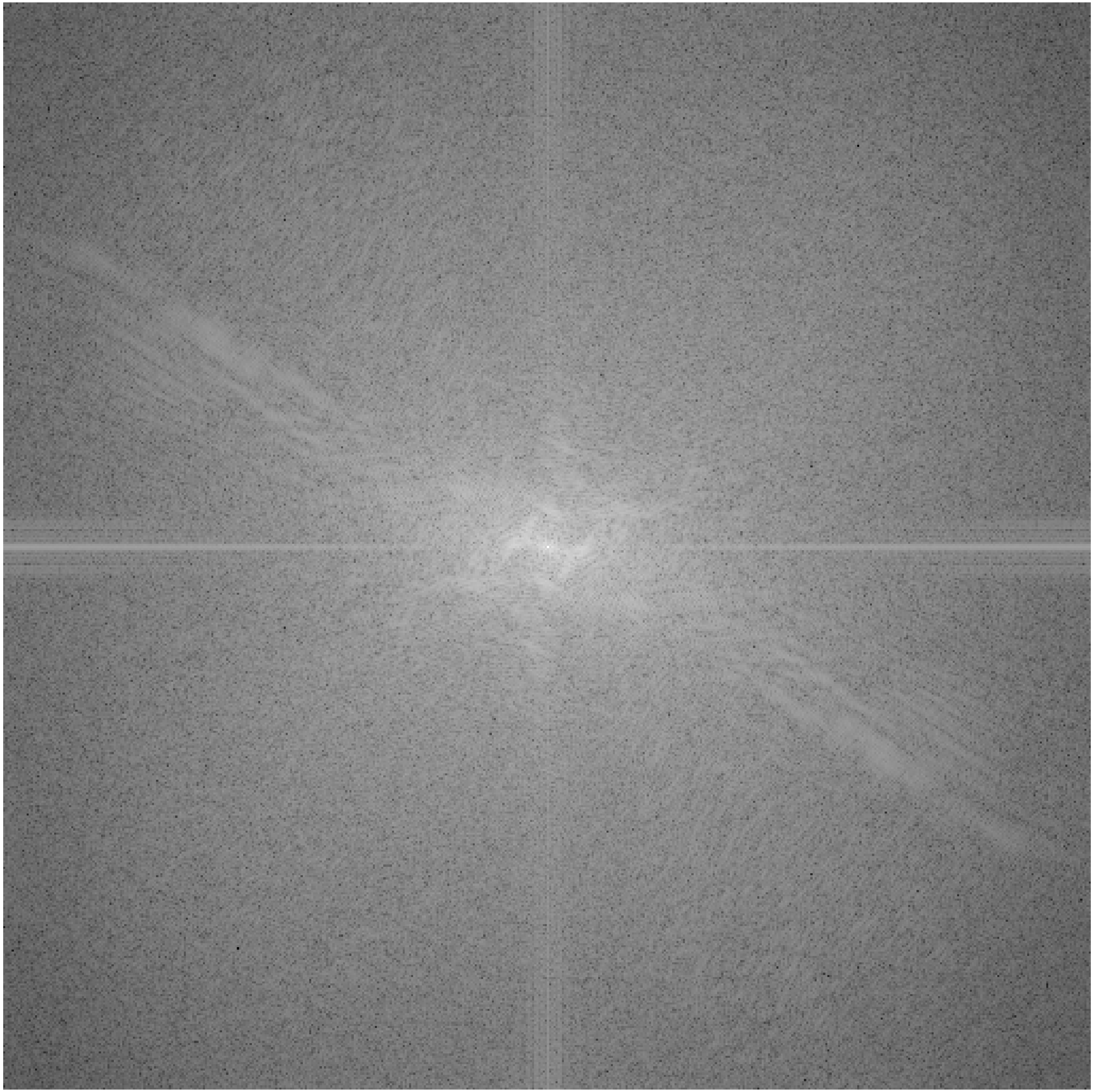
This is the magnitude transform of the cheetah pic



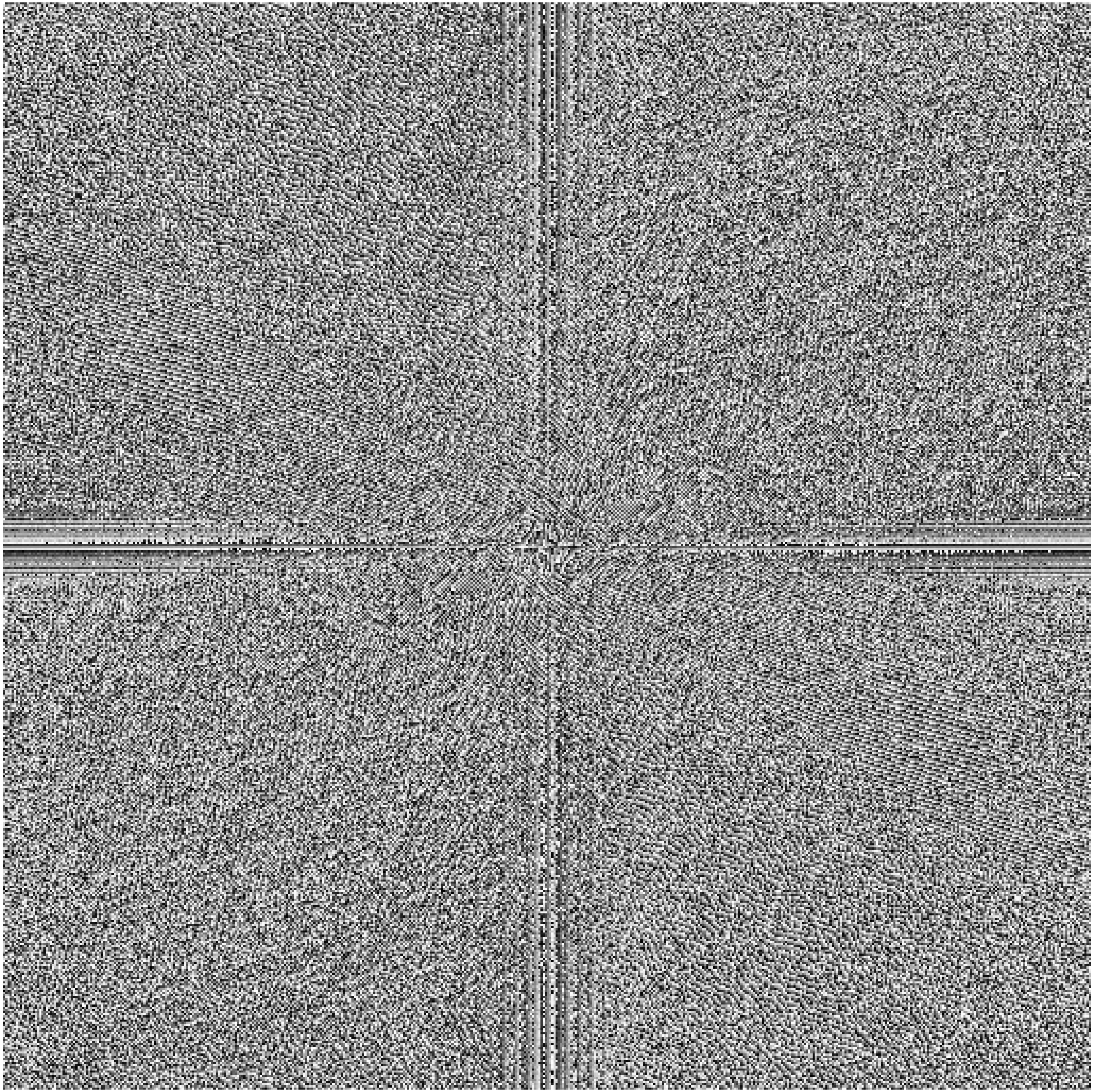
This is the phase transform of the cheetah pic



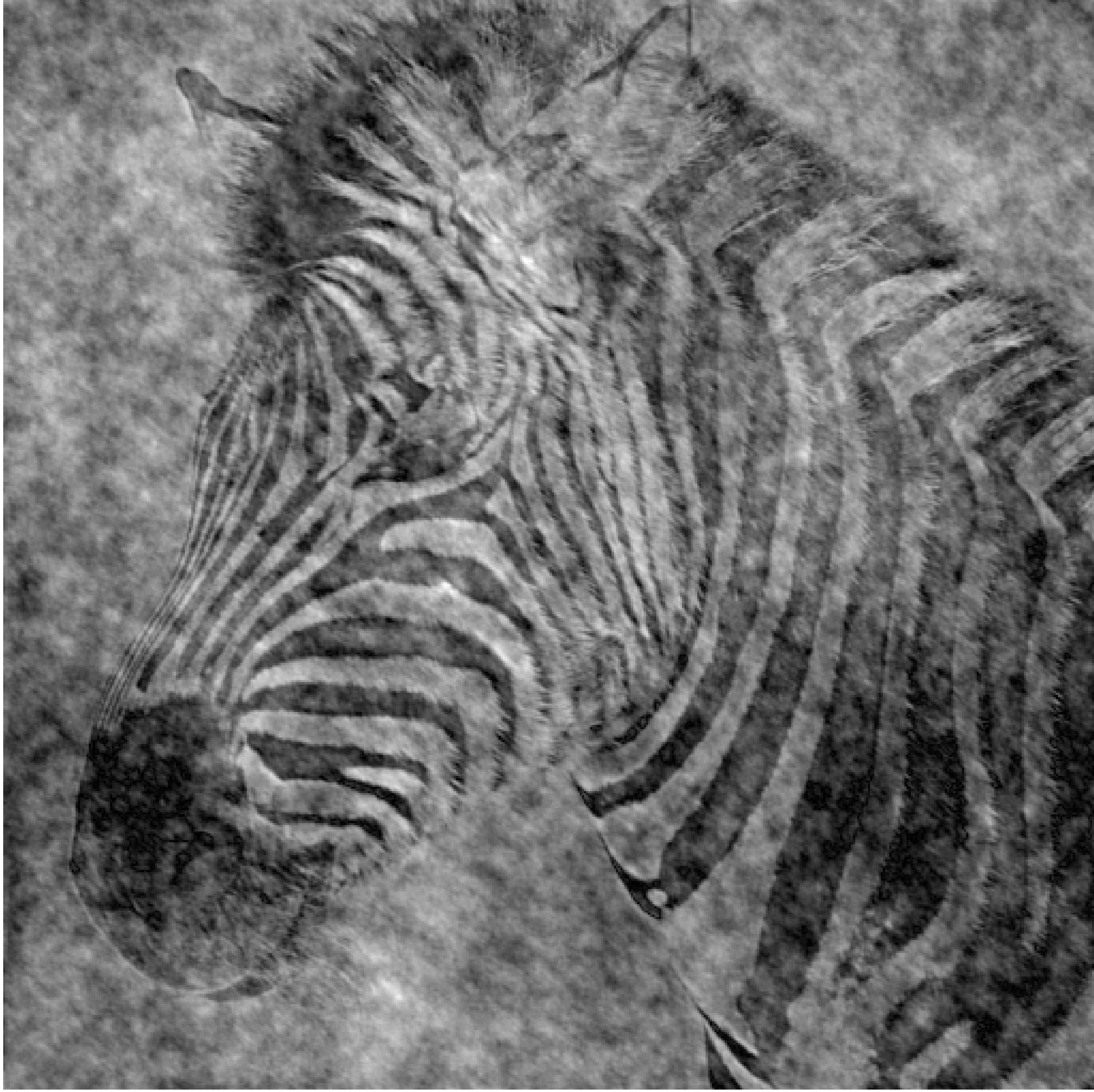
This is the magnitude transform of the zebra pic



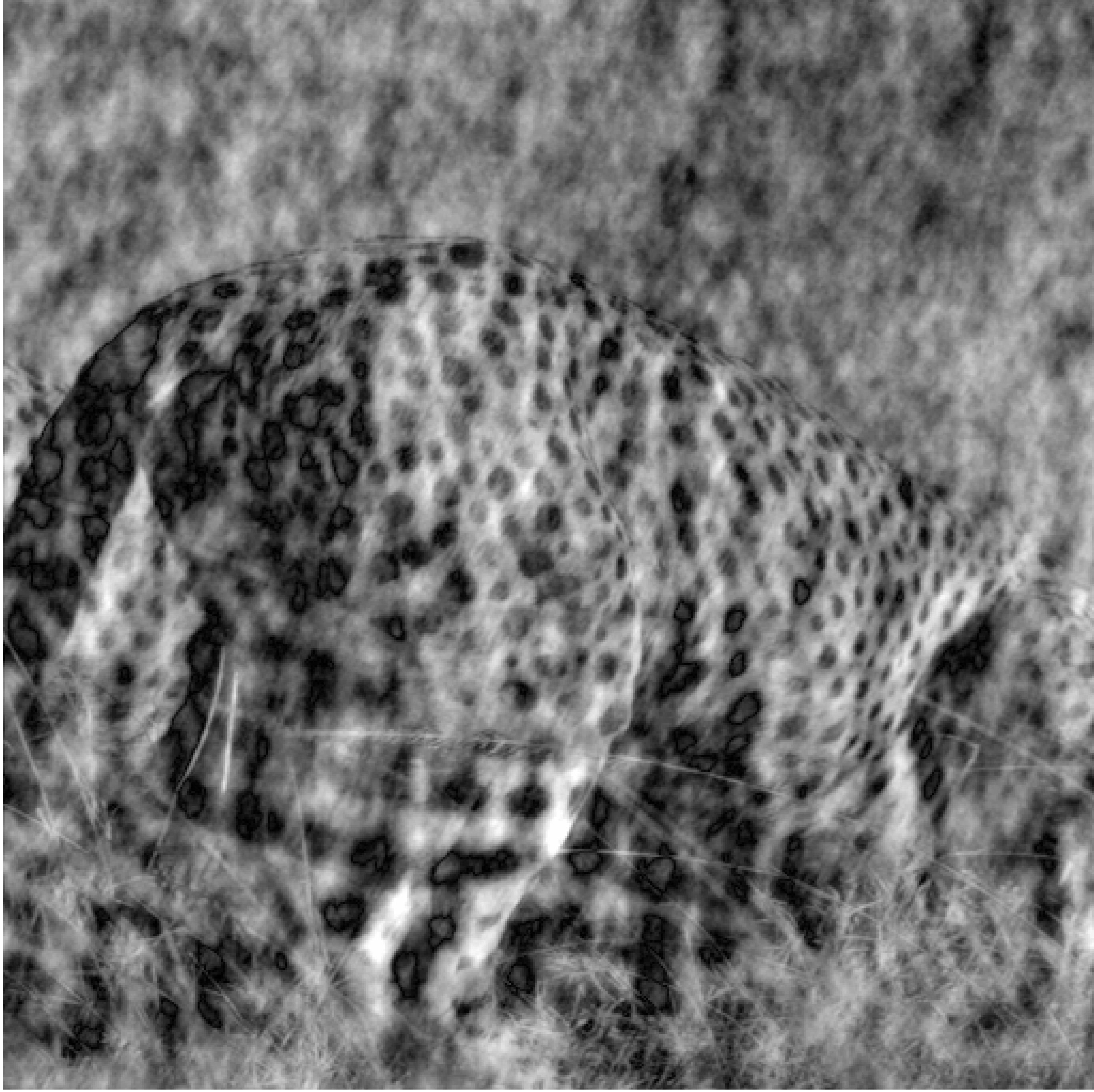
This is the phase transform of the zebra pic



Reconstruction with zebra phase, cheetah magnitude



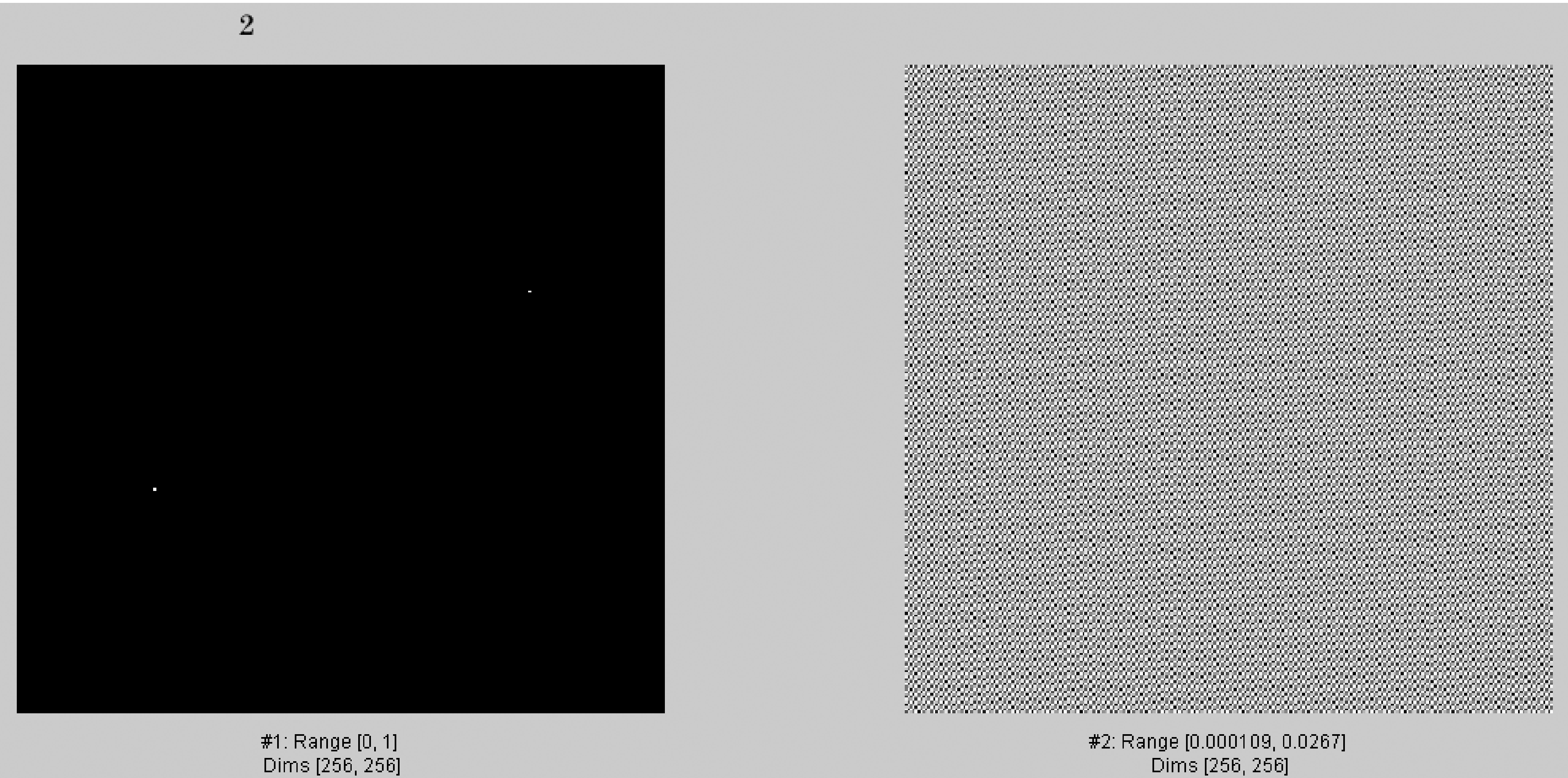
Reconstruction with cheetah phase, zebra magnitude



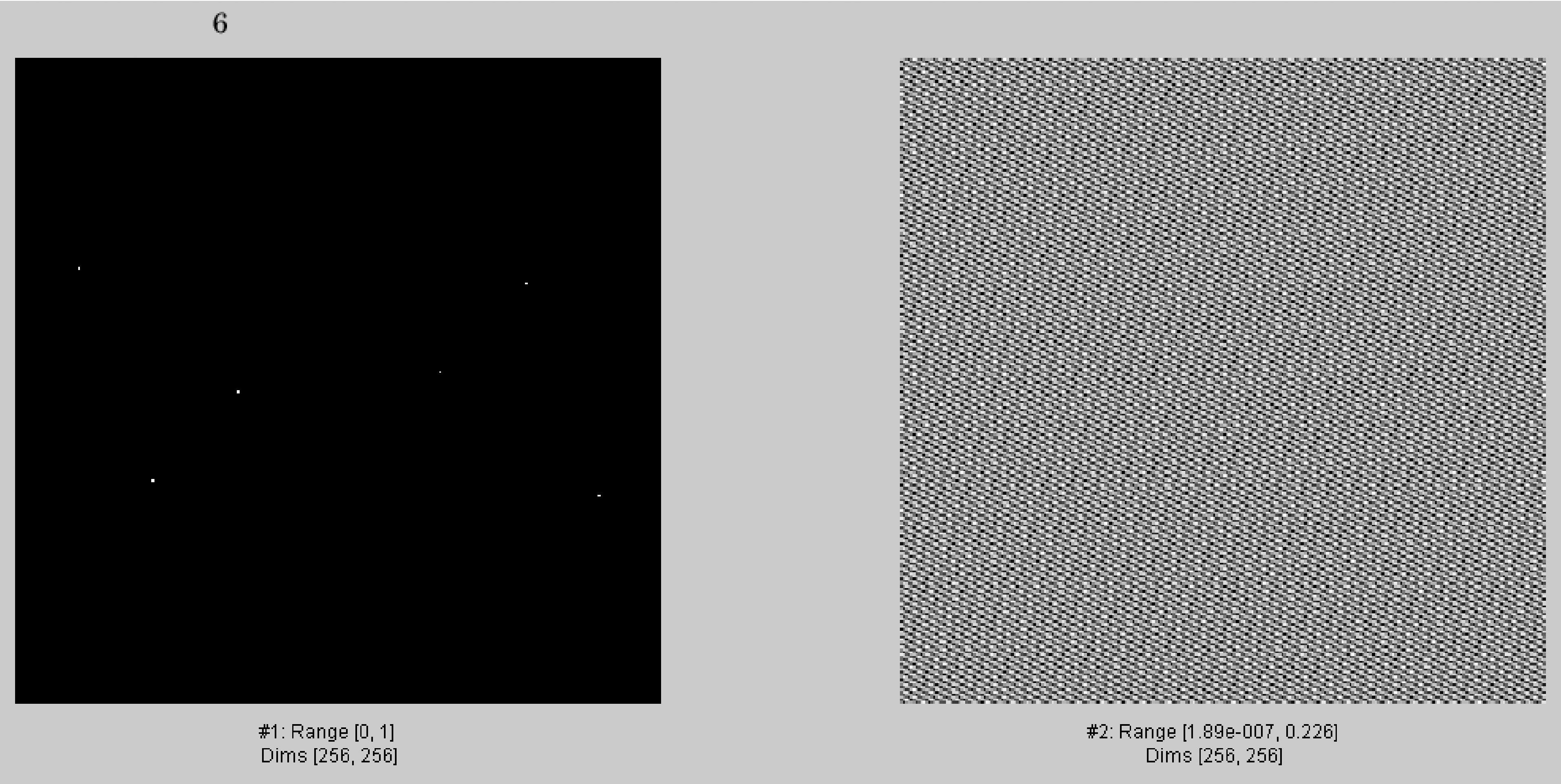
Example image synthesis with fourier basis.

- 16 images

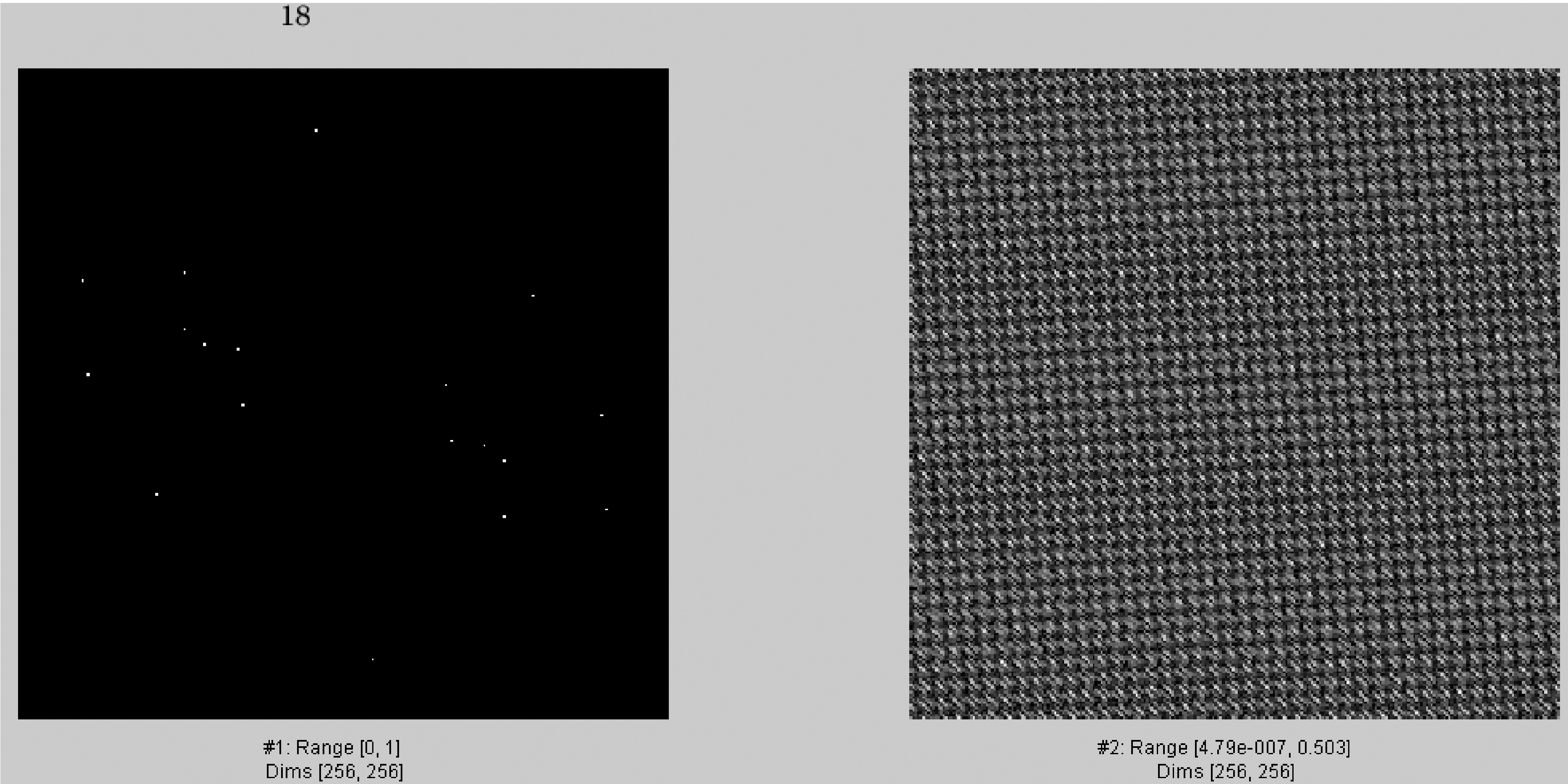
2



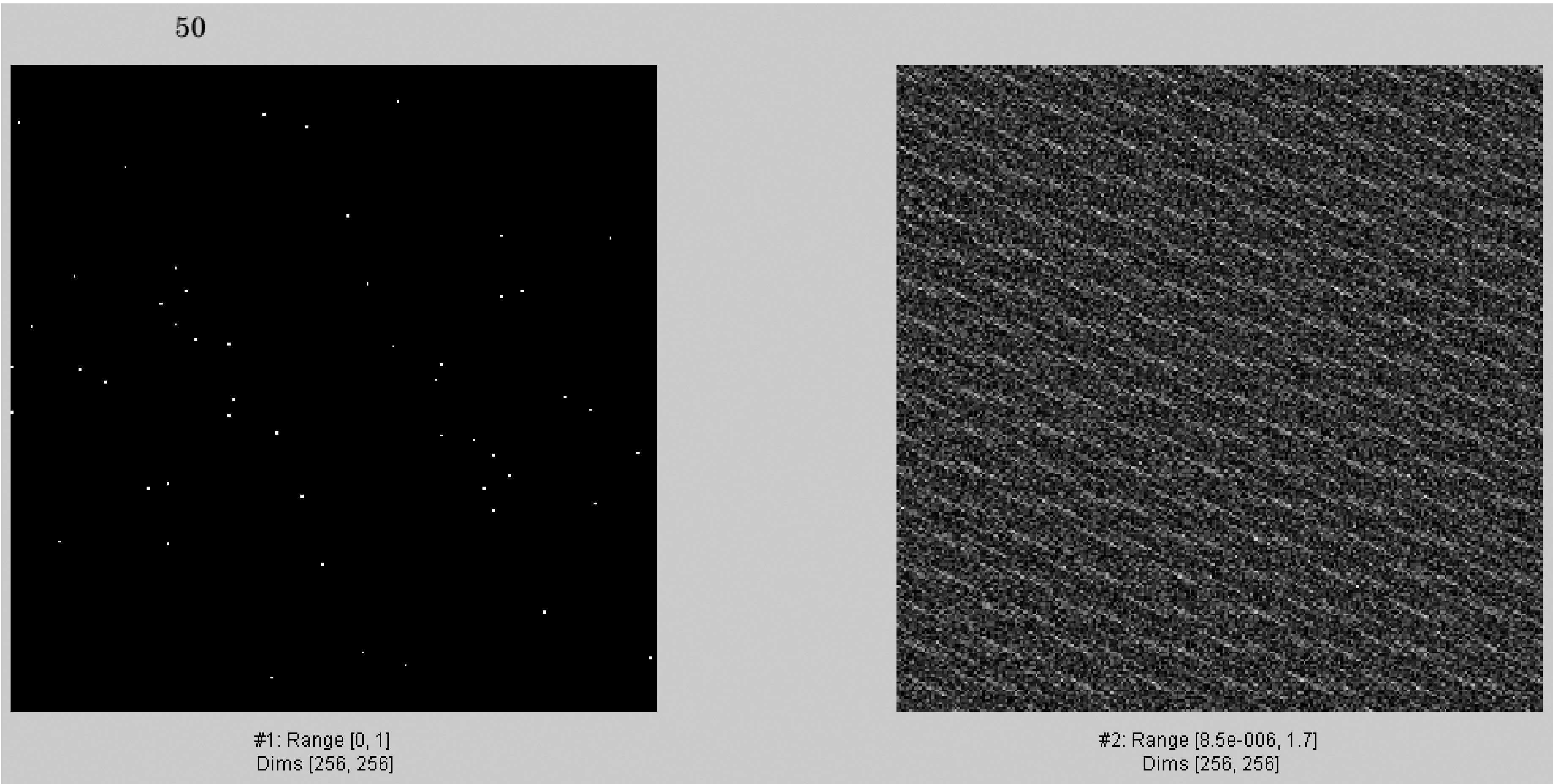
6



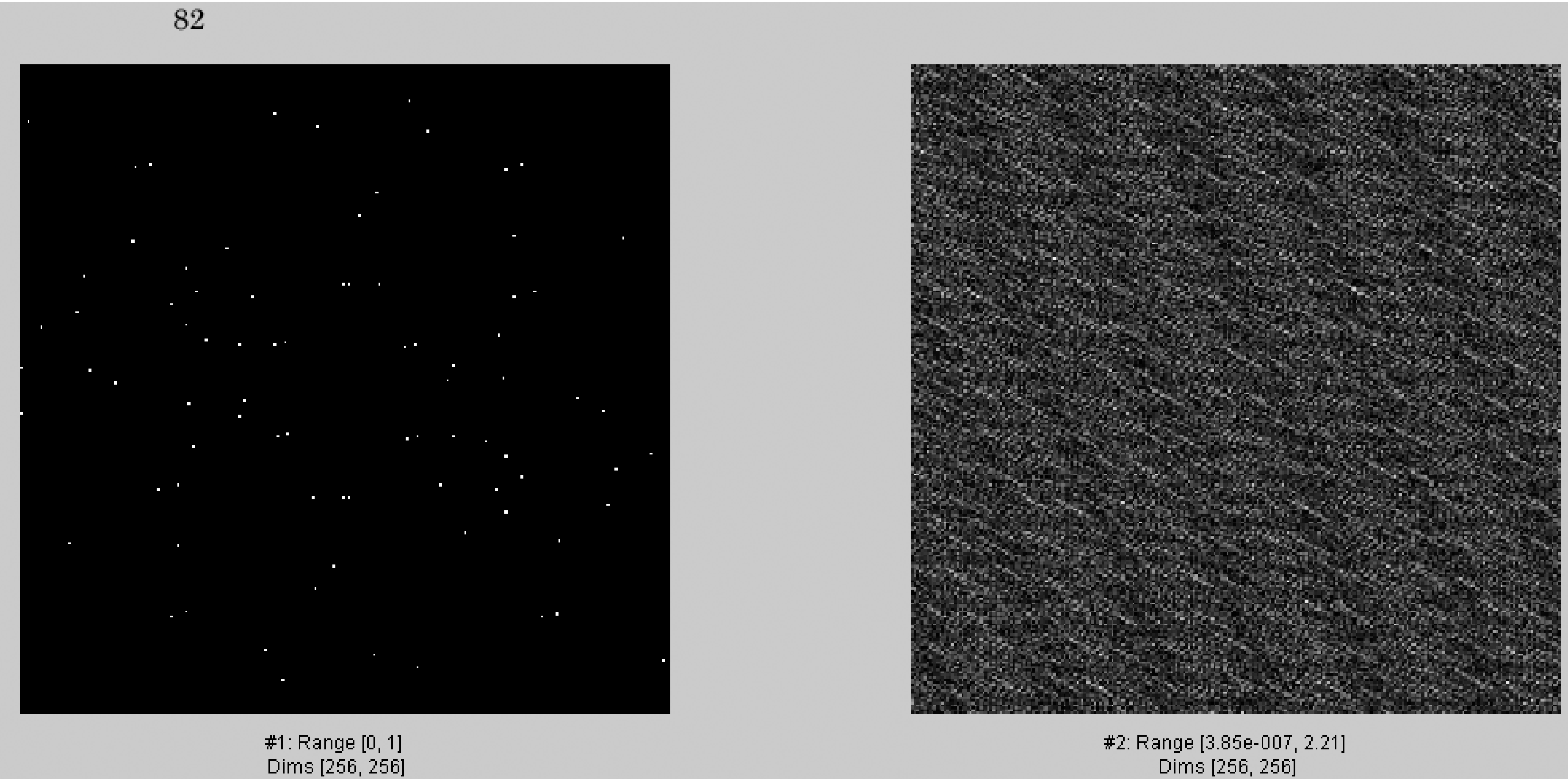
18



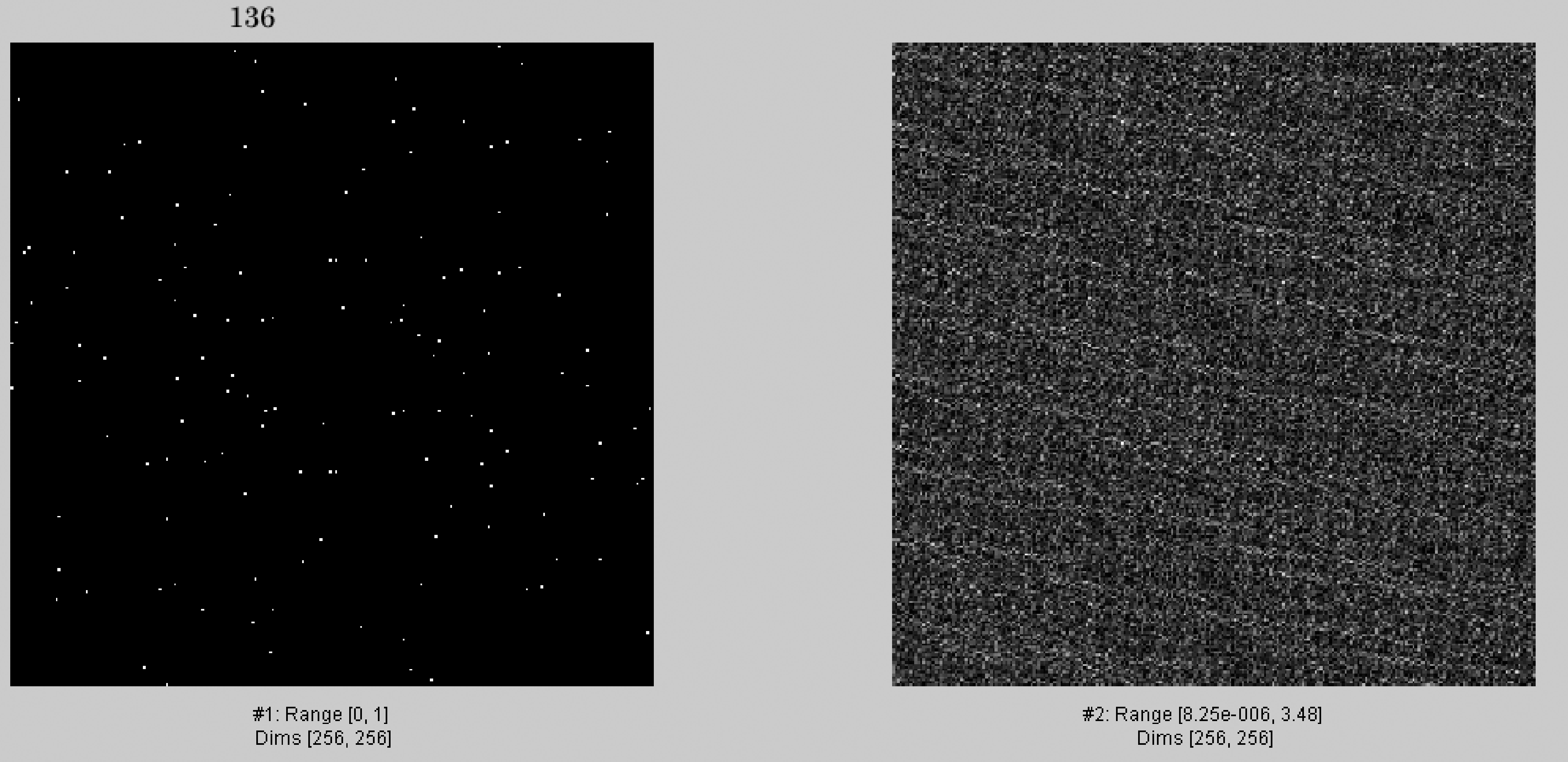
50



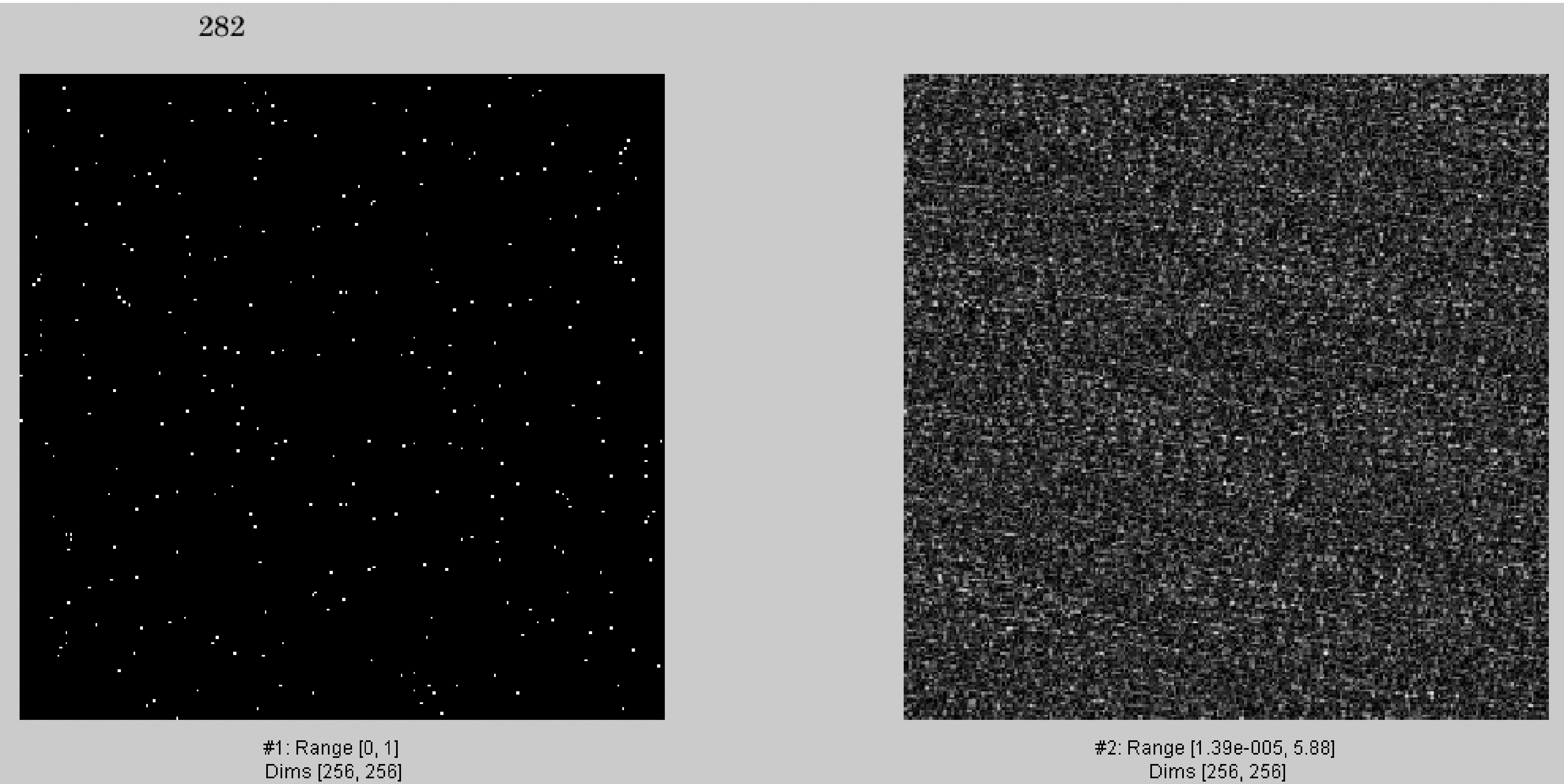
82



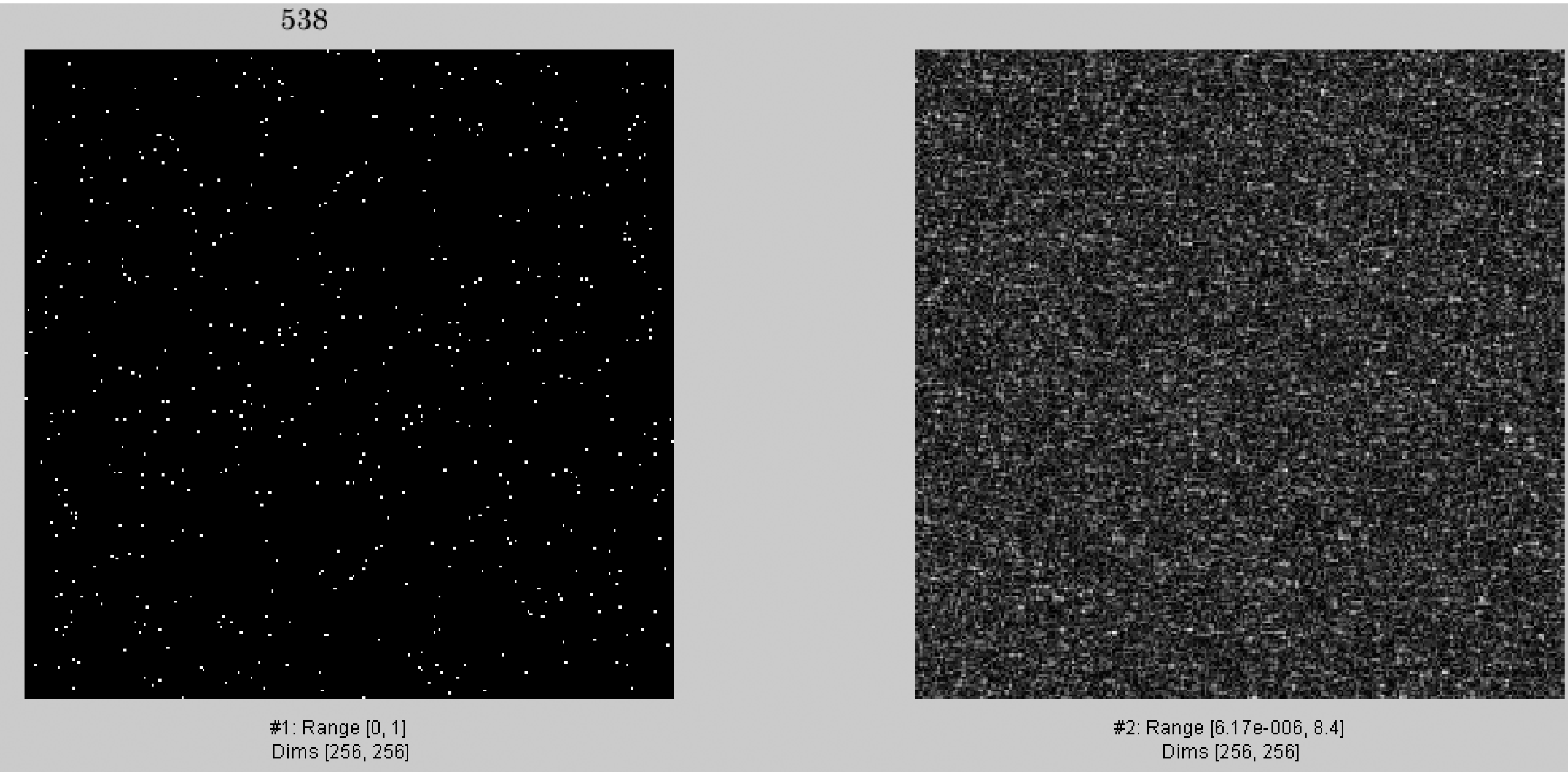
136



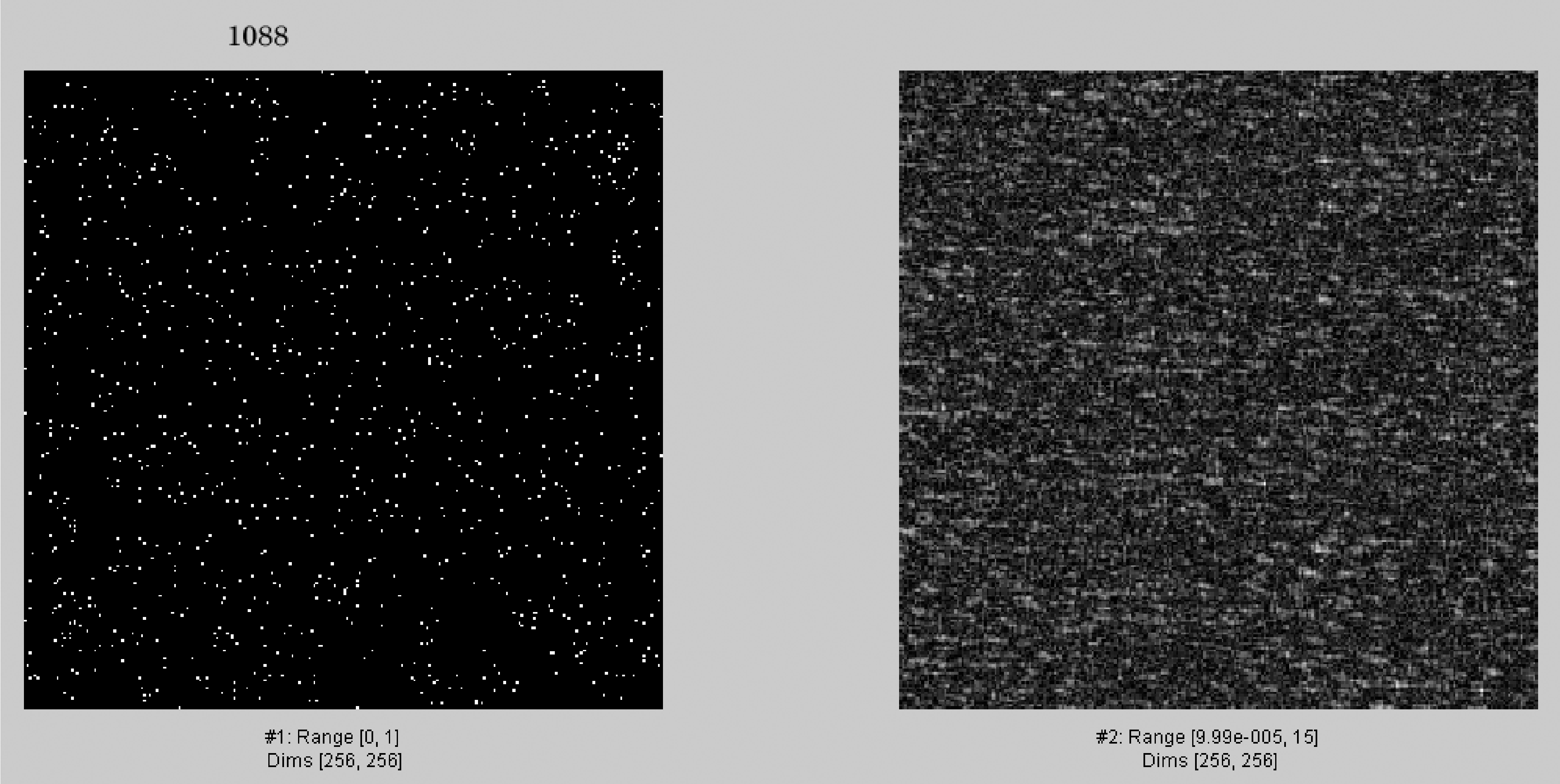
282



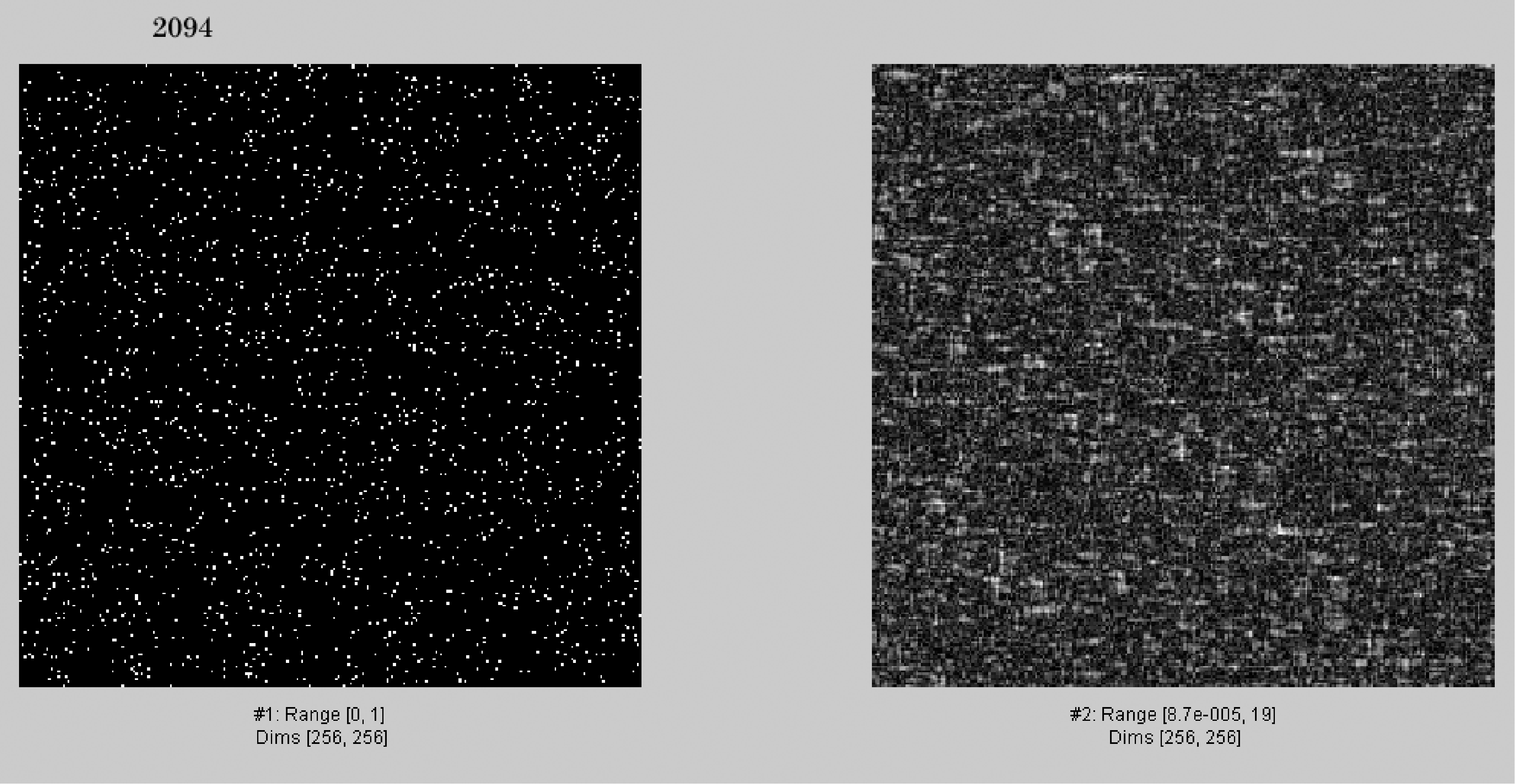
538



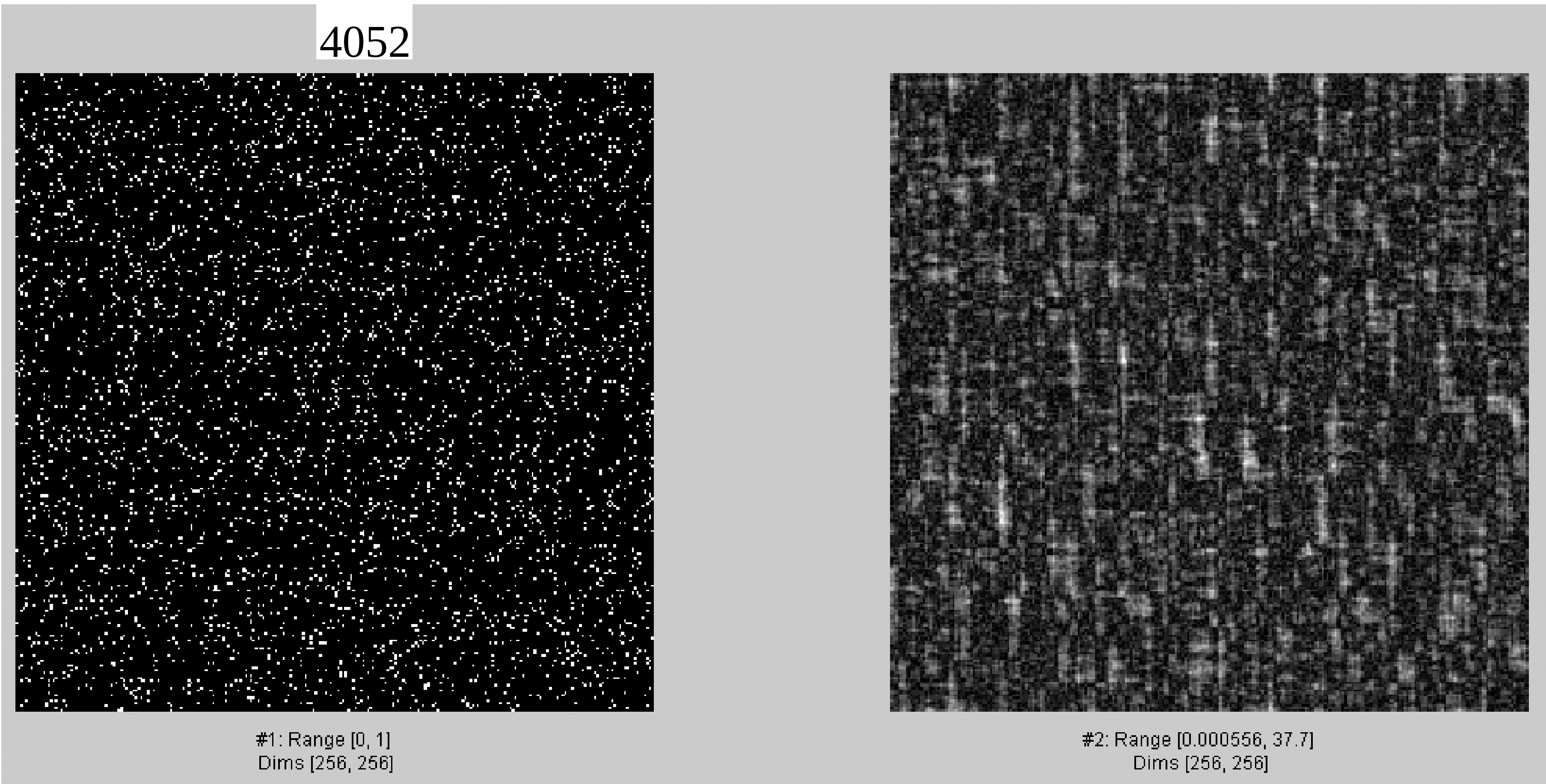
1088



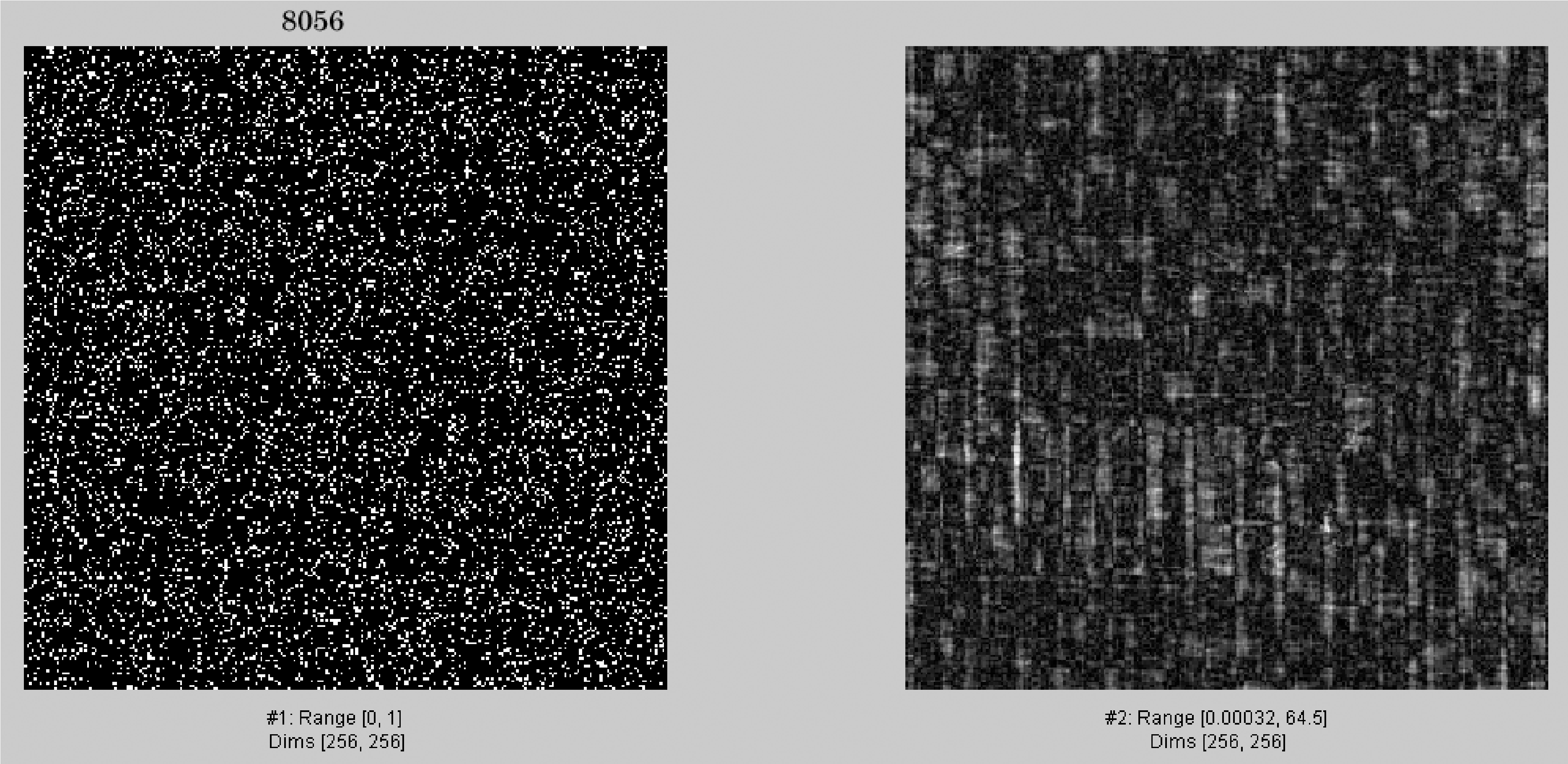
2094



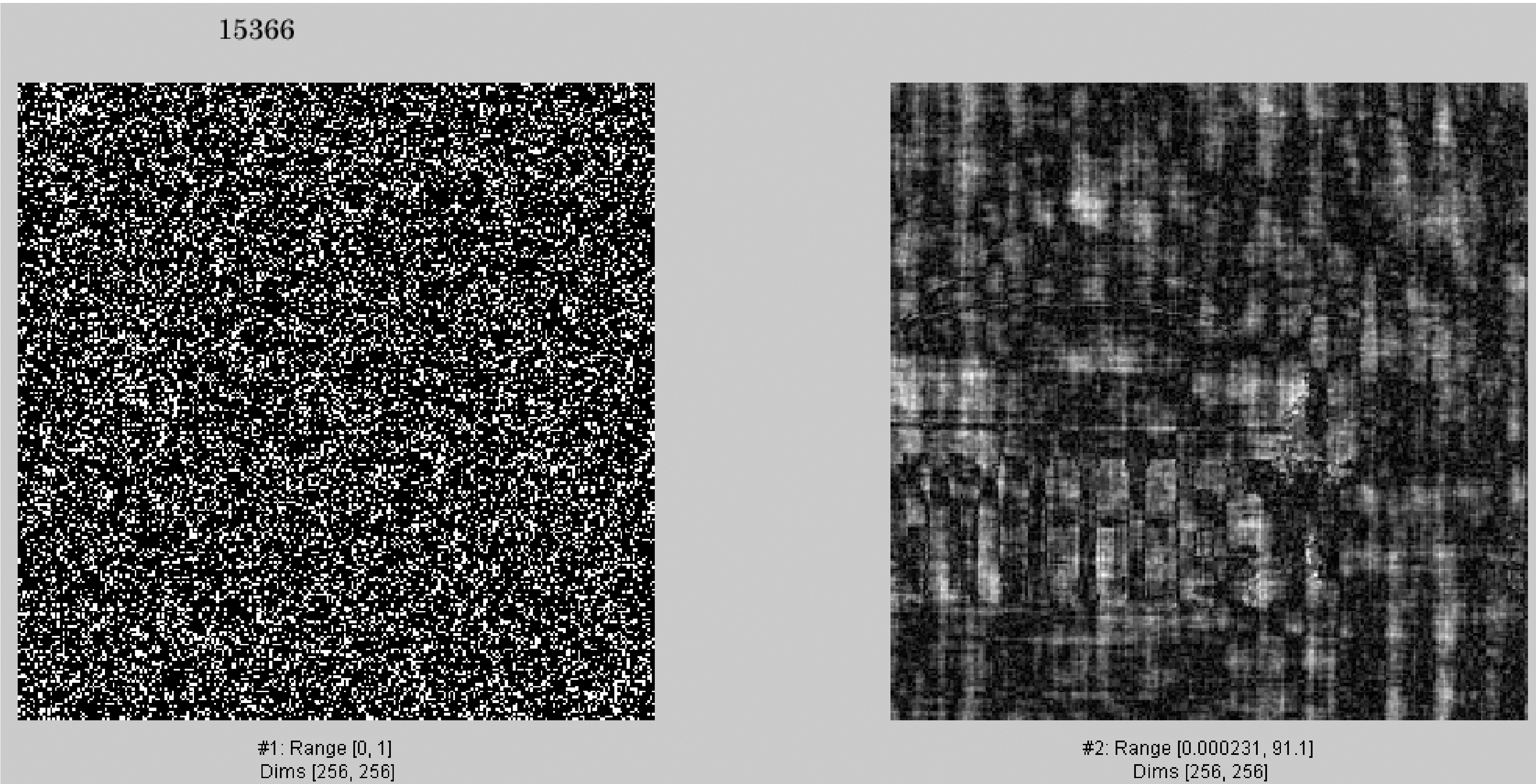
4052.



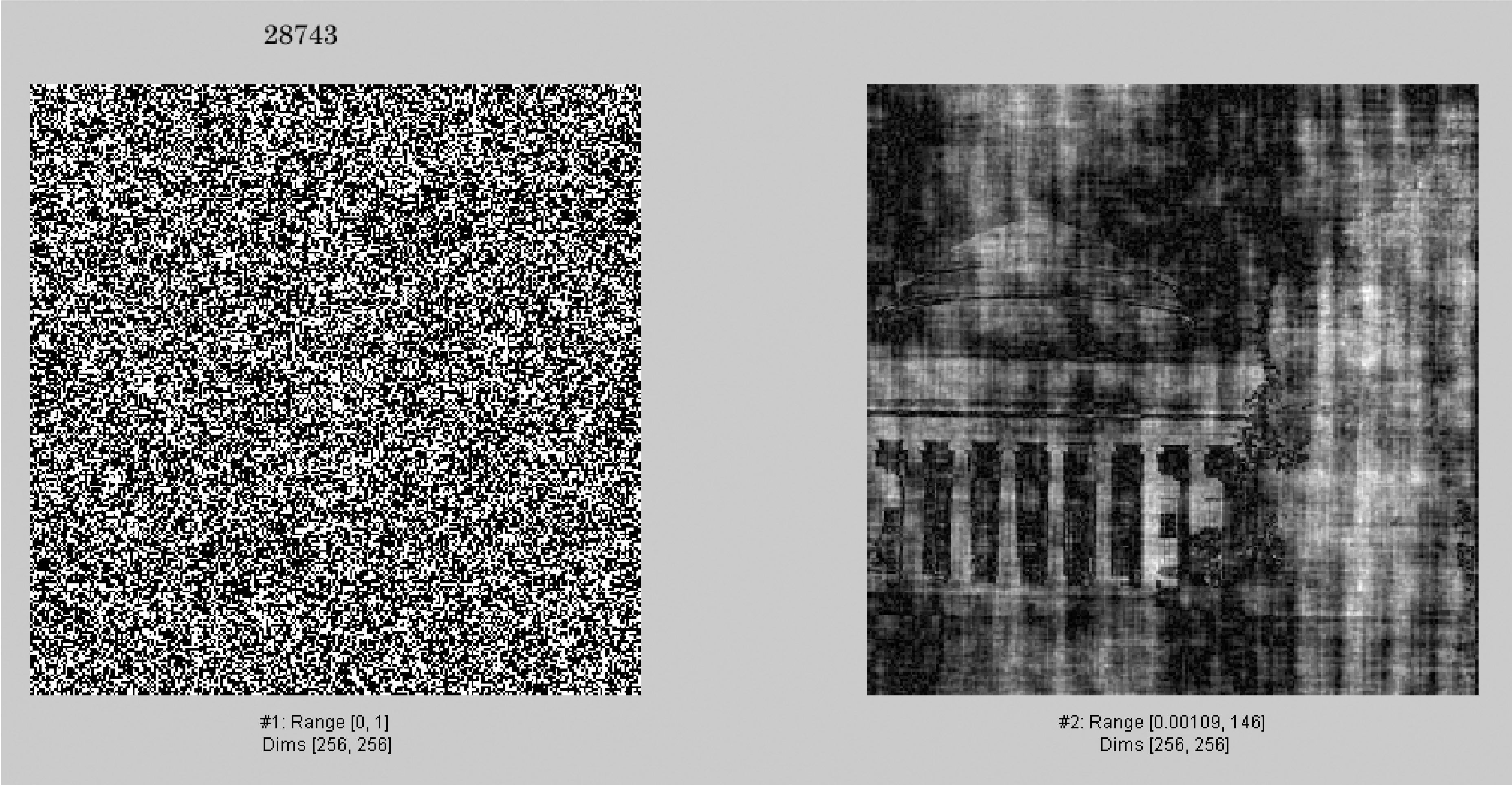
8056.



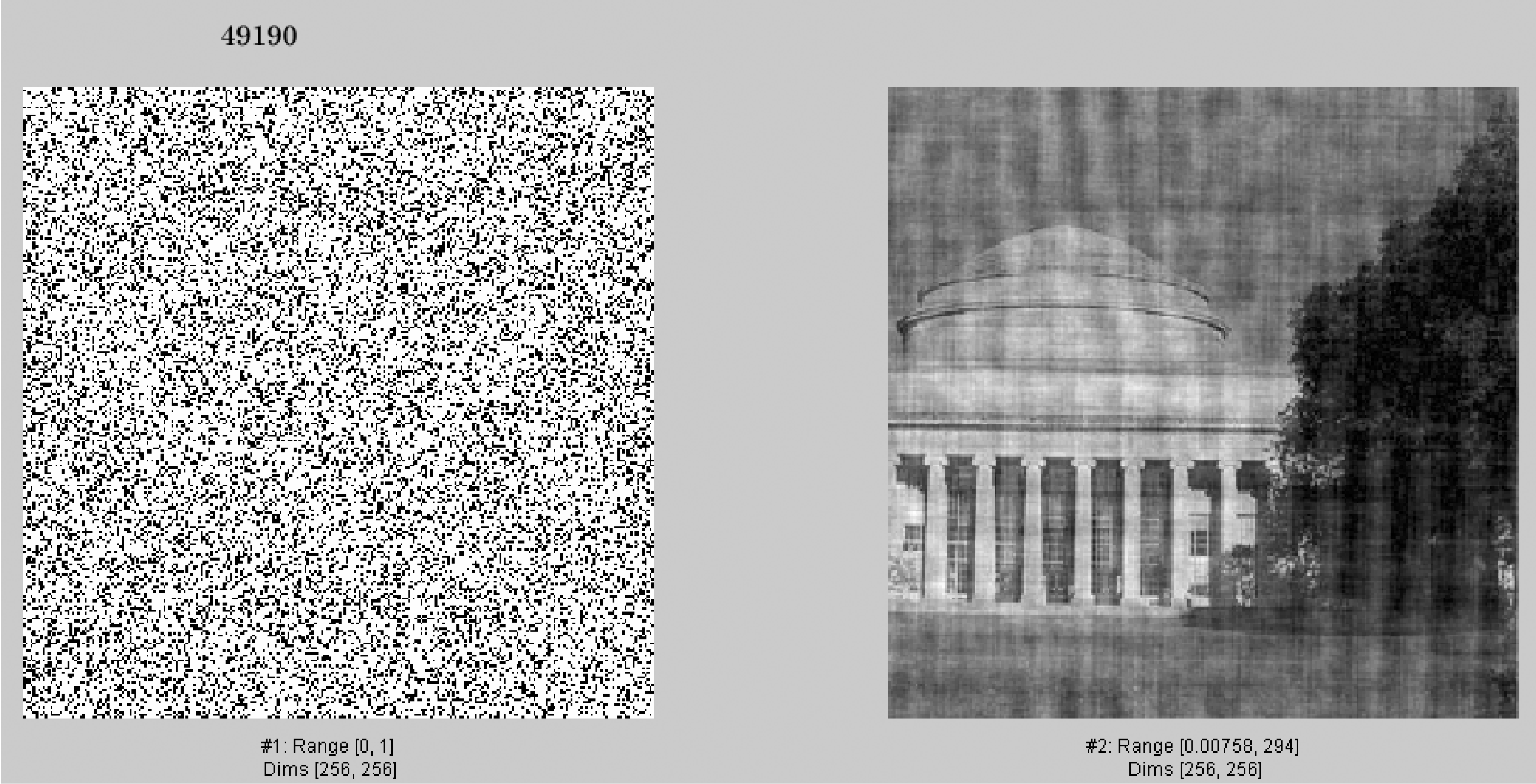
15366



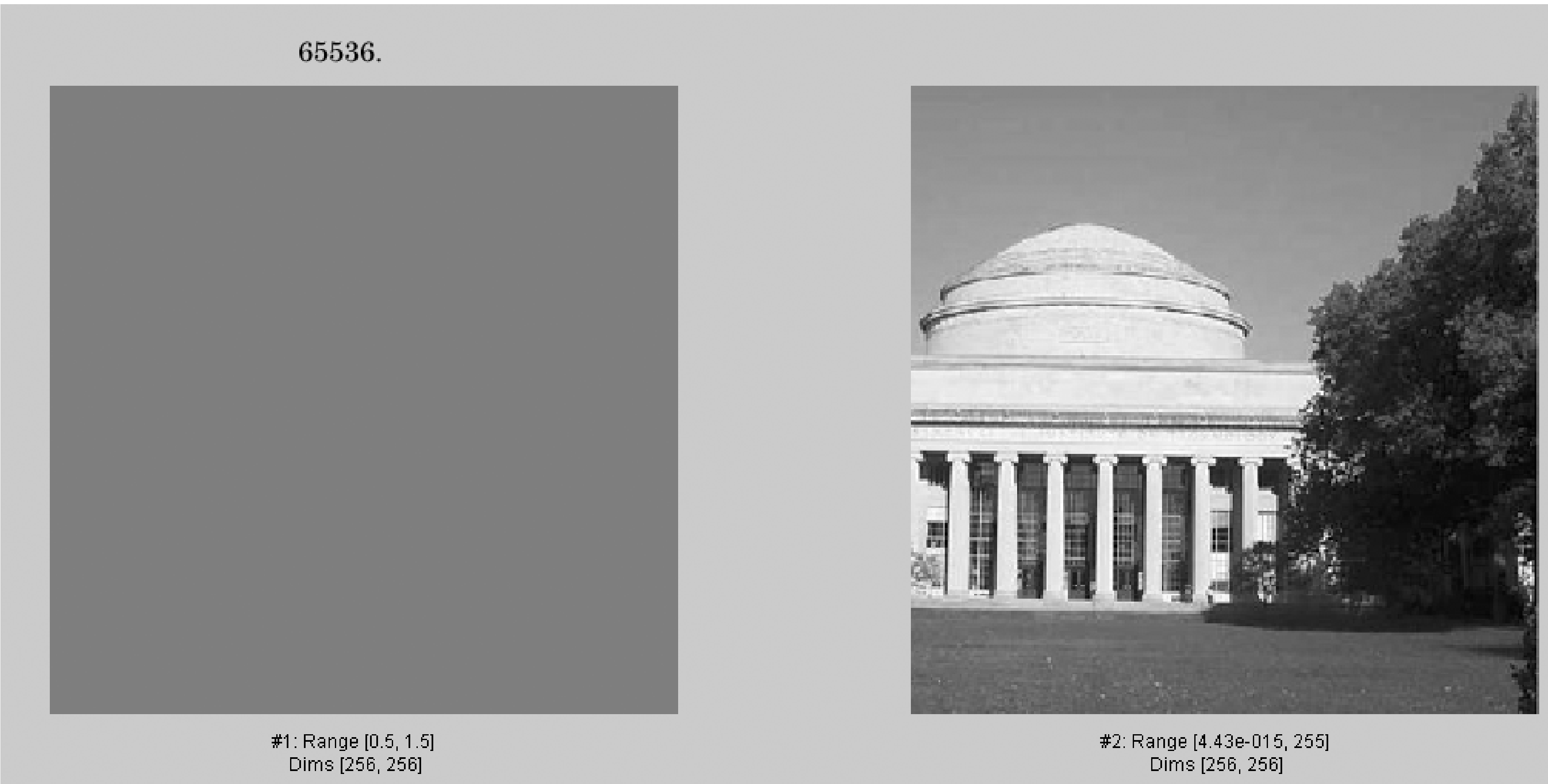
28743



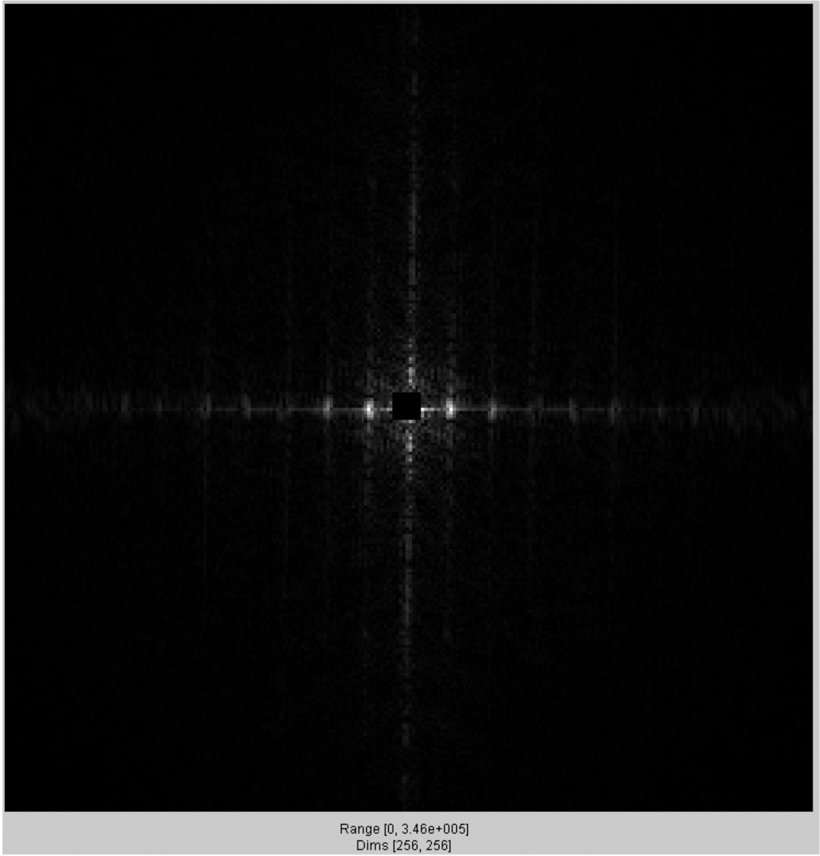
49190.



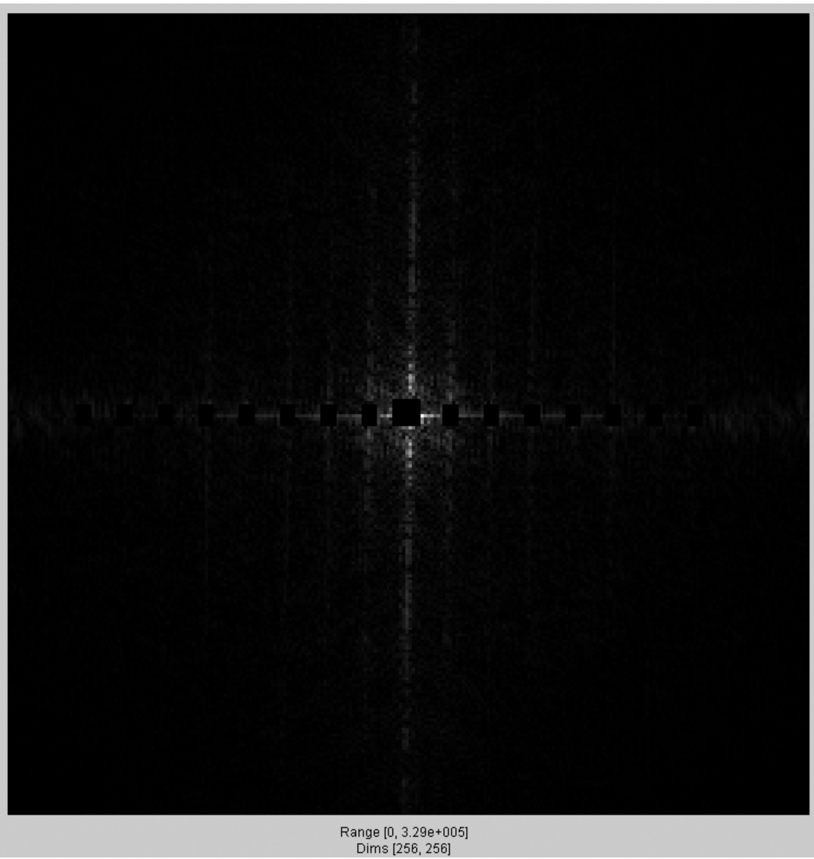
65536.

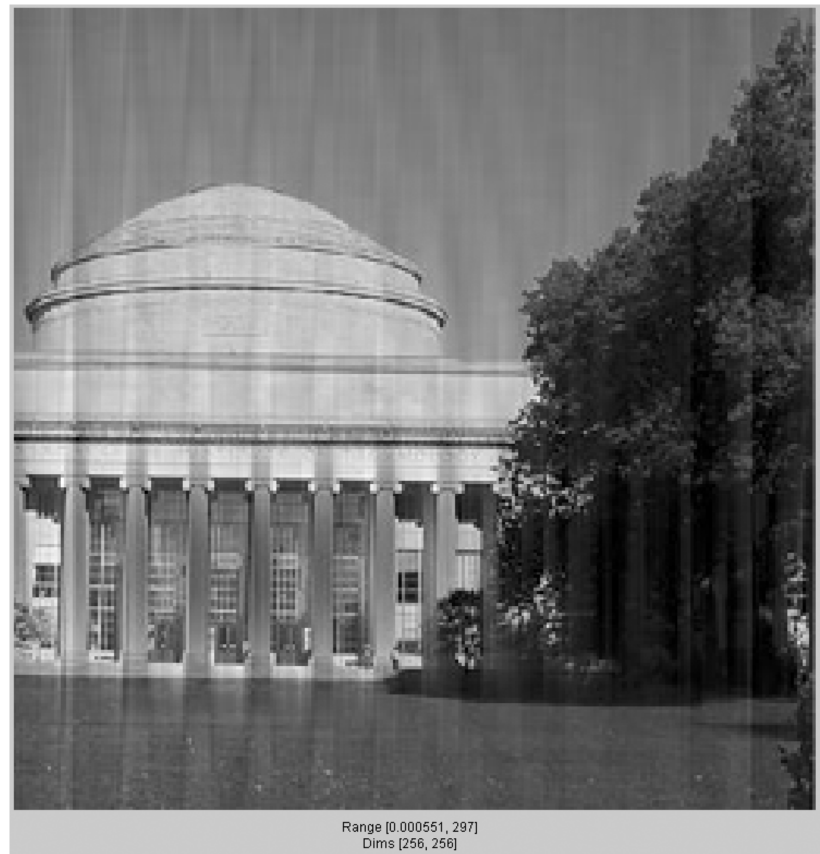

65536.
#1: Range [0.5, 1.5]
Dims [256, 256]
#2: Range [4.43e-015, 255]
Dims [256, 256]

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

Forsyth&Ponce

Discrete-time, continuous frequency Fourier transform

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

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

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

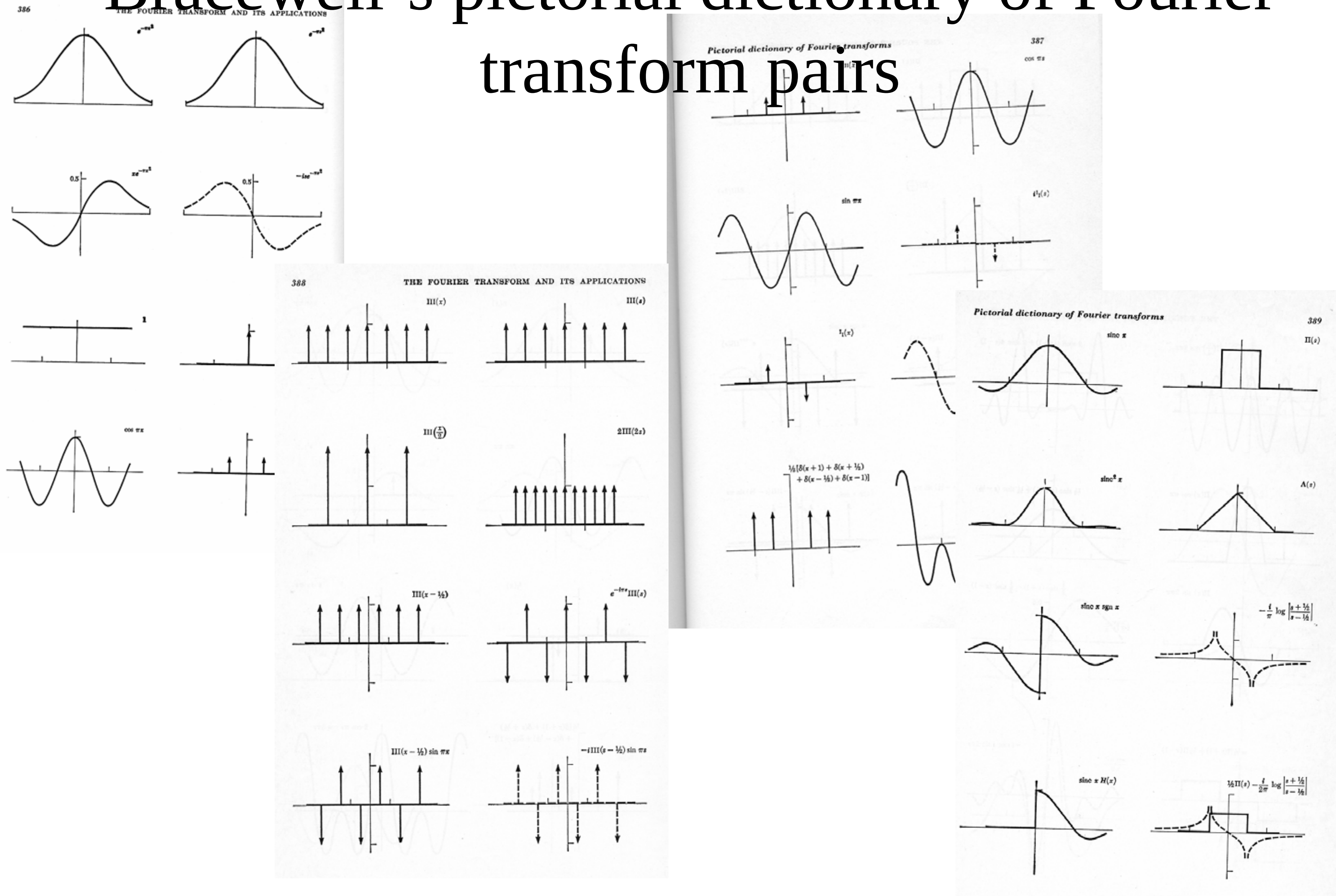
Discrete-time, continuous frequency Fourier transform pairs

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. $a^n u[n] \quad (\lvert a\rvert < 1)$	$\frac{1}{1-ae^{-j\omega}}$
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. $\frac{r^n \sin\omega_p(n+1)}{\sin\omega_p} u[n] \quad (\lvert r\rvert < 1)$	$\frac{1}{1-2r\cos\omega_p e^{-j\omega}+r^2 e^{-j2\omega}}$
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

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

Write the DFT and convolution explicitly

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

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

$$f = g \otimes h$$

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

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

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

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

$$f = g \otimes h$$

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

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

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

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

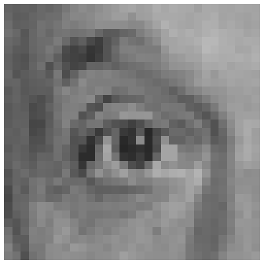
$$= \sum_{k,l} G[m,n] e^{-\pi i \left(\frac{km}{M} + \frac{\ln}{N}\right)} h[k,l]$$

Perform the other DFT (circular boundary conditions)

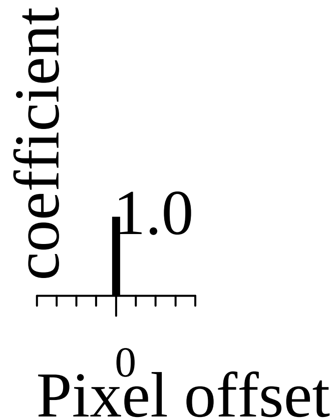
$$= G[m,n] H[m,n]$$

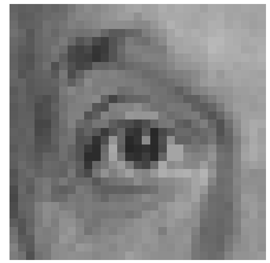
Analysis of our simple filters

Analysis of our simple filters



original

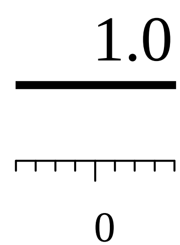




Filtered
(no change)

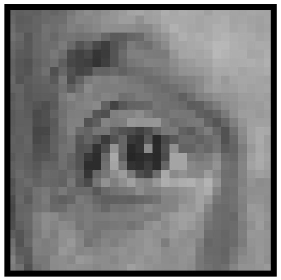
$$F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k,l] e^{-\pi i \left(\frac{km}{M} + \frac{\ln}{N} \right)}$$

$$= 1$$

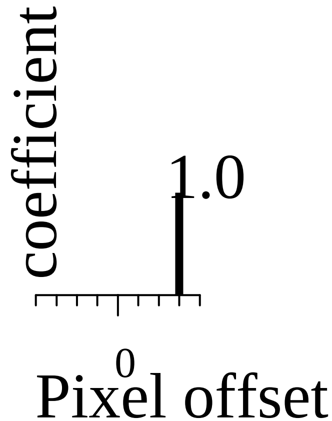


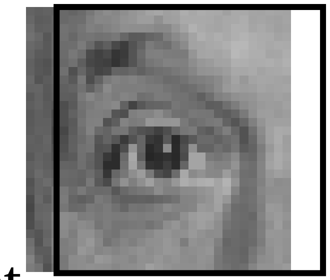
constant

Analysis of our simple filters



original

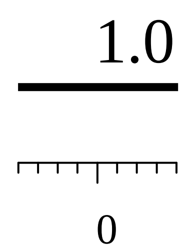




shifted

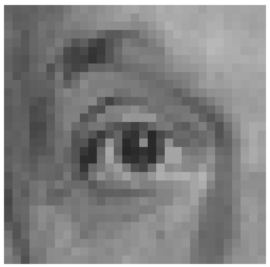
$$F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k-\delta, l] e^{-\pi i \left(\frac{km}{M} + \frac{\ln}{N} \right)}$$

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

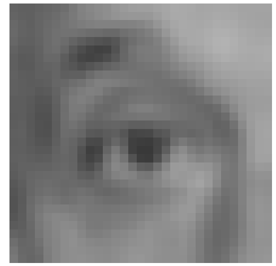


Constant magnitude, linearly shifted phase

Analysis of our simple filters



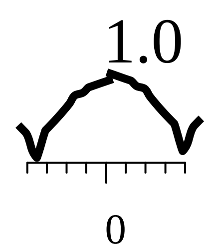
original



blurred

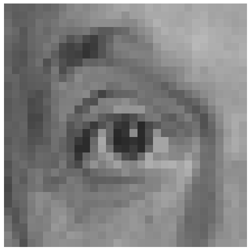
$$F[m,n]=\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} f[k,l]e^{-\pi i\left(\frac{km}{M}+\frac{\ln}{N}\right)}$$

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

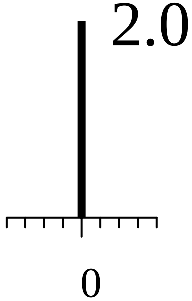


Low-pass filter

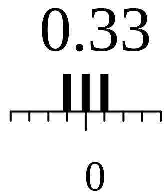
Analysis of our simple filters

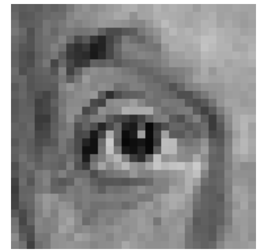


original



−



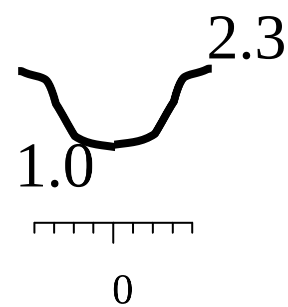


sharpened

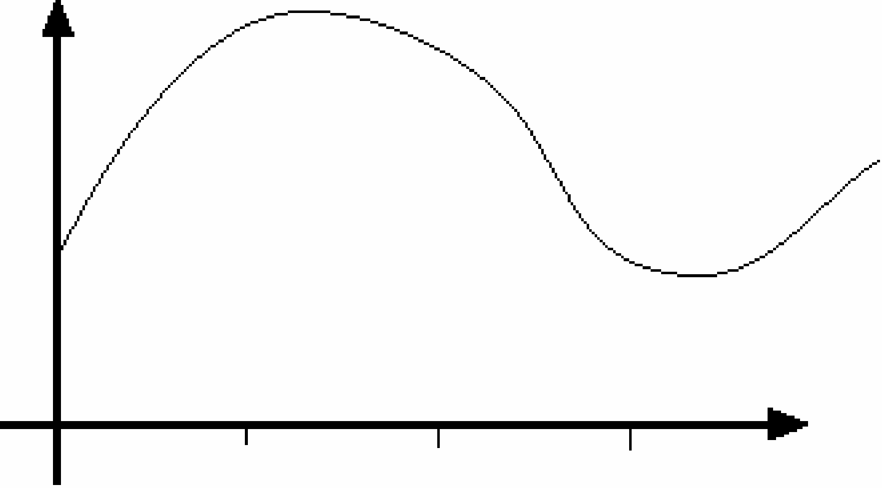
$$F[m,n] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} f[k,l]\, e^{-\pi i\left(\frac{km}{M}+\frac{\ln}{N}\right)}$$

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

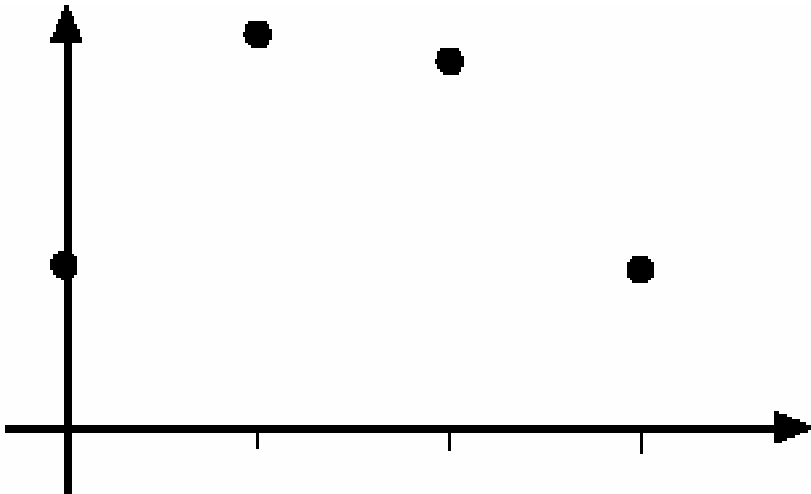
high-pass filter

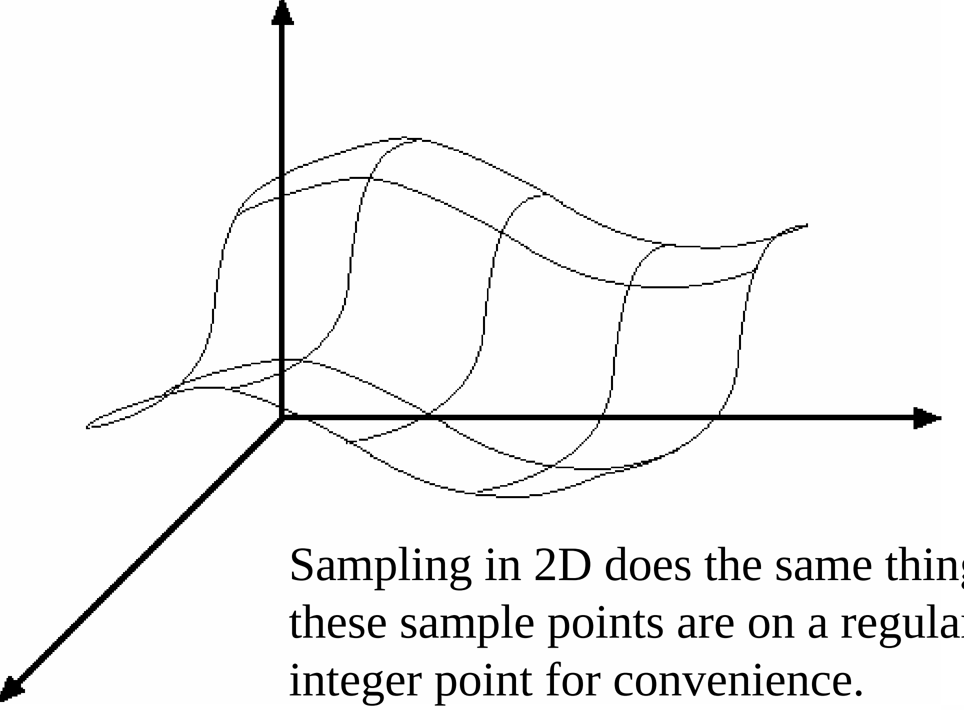


Sampling and aliasing

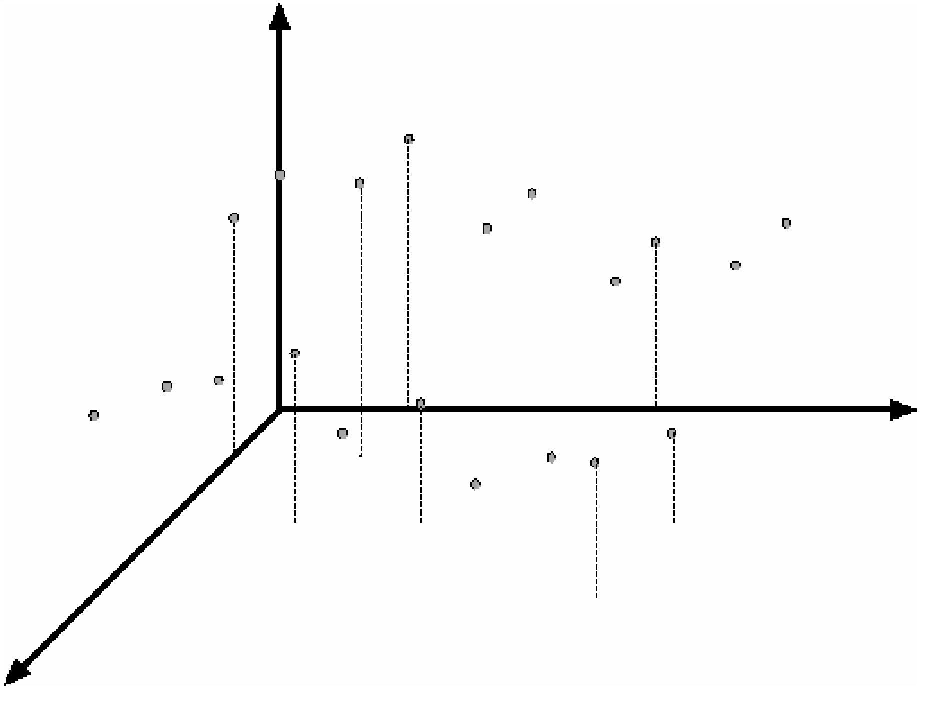


Sampling in 1D takes a continuous function and replaces it with a vector of values, consisting of the function's values at a set of sample points. We'll assume that these sample points are on a regular grid, and can place one at each integer for convenience.





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

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

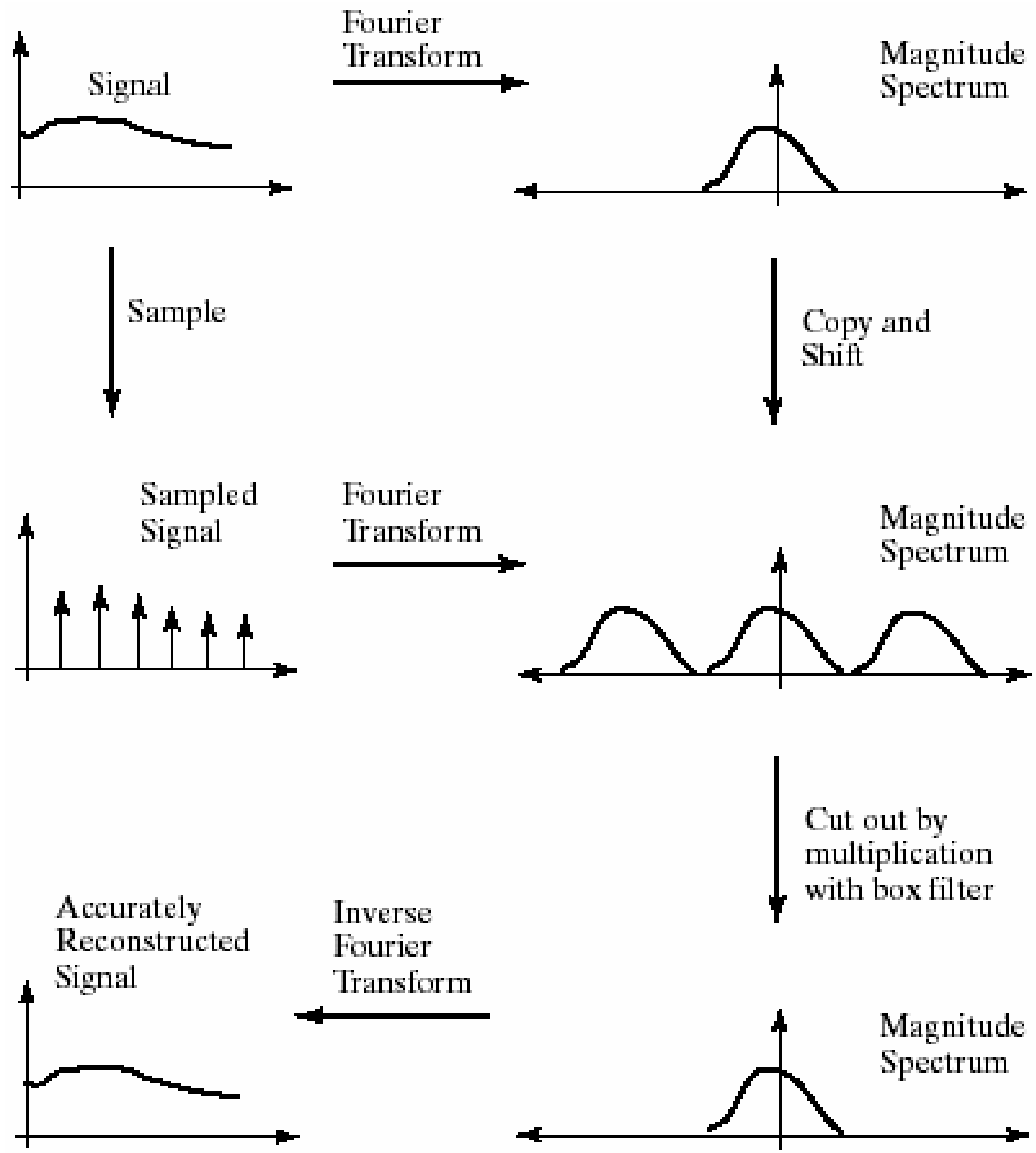
$$= f(x,y)\sum_{i=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} \delta(x-i,y-j)$$

The Fourier transform of a sampled signal

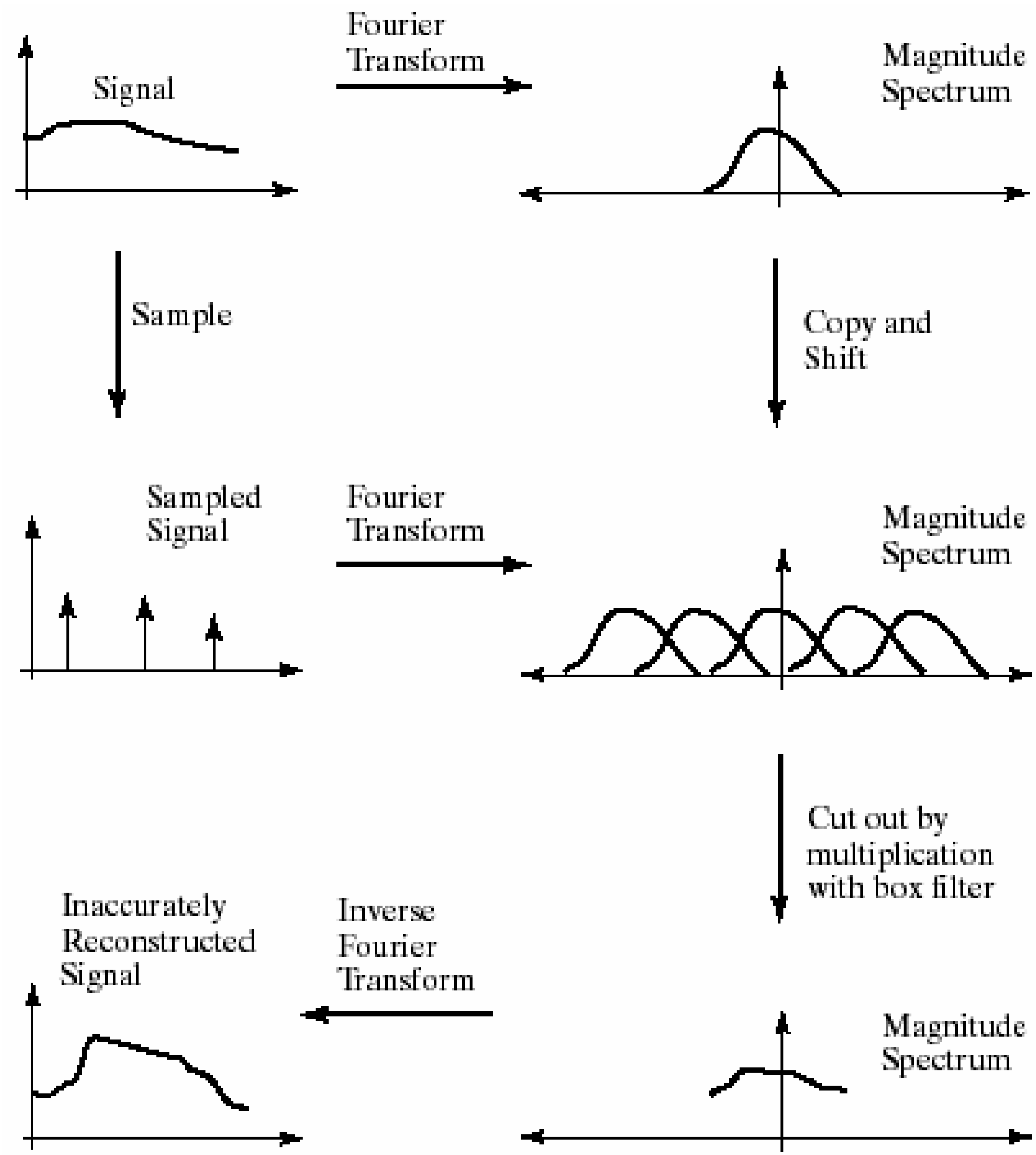
$$F\left(\mathrm{Sample}_{2\mathrm{D}}(f(x,y))\right) = F\left(f(x,y)\sum_{i=-\infty}^{\infty}\sum_{i=-\infty}^{\infty}\delta(x-i,y-j)\right)$$

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

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

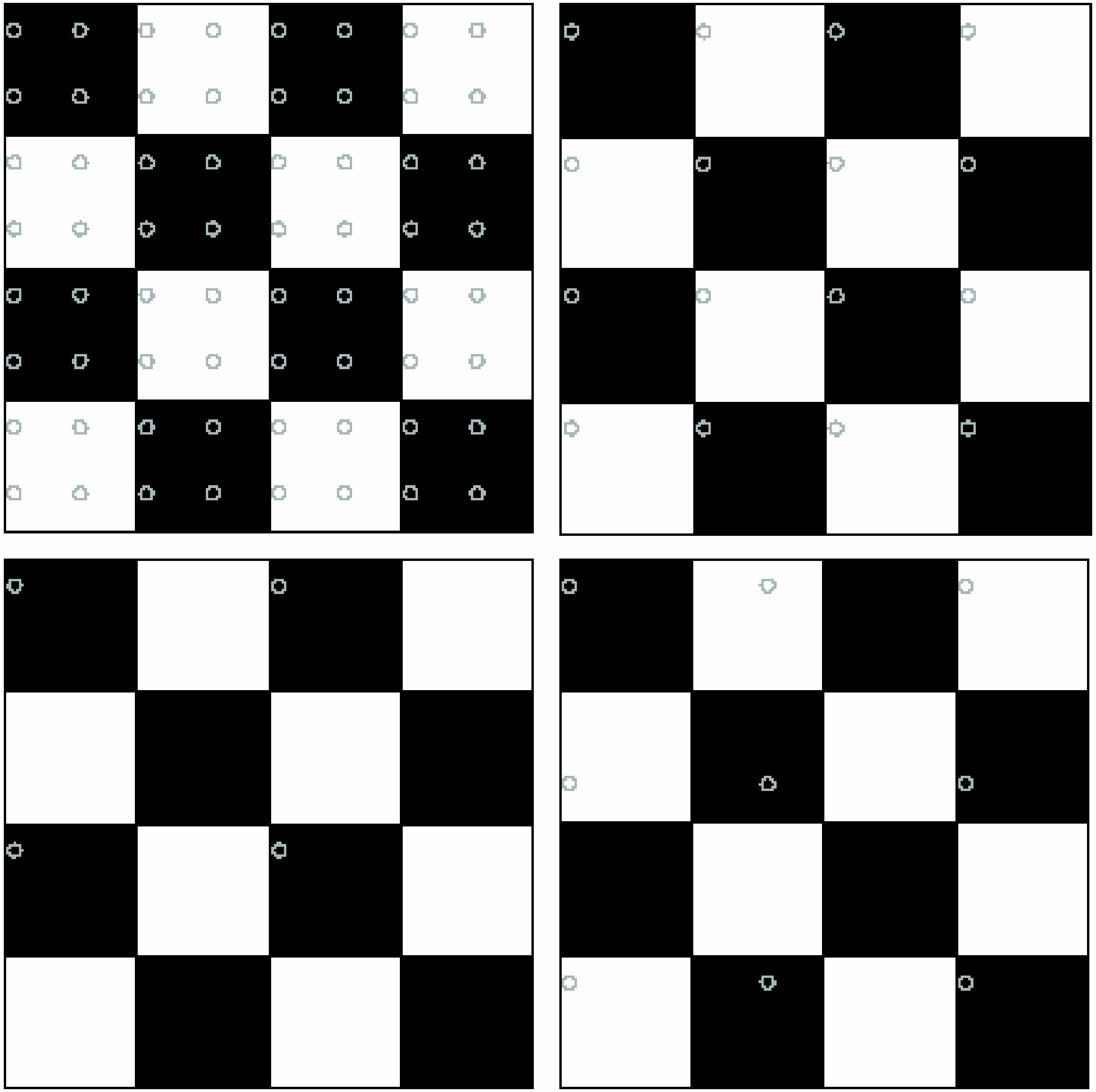


Fourier
Transform
Signal
Magnitude
Spectrum
Sample
Copy and
Shift
Sampled
Signal
Fourier
Transform
Magnitude
Spectrum
Cut out by
multiplication
with box filter
Accurately
Reconstructed
Signal
Inverse
Fourier
Transform
Magnitude
Spectrum

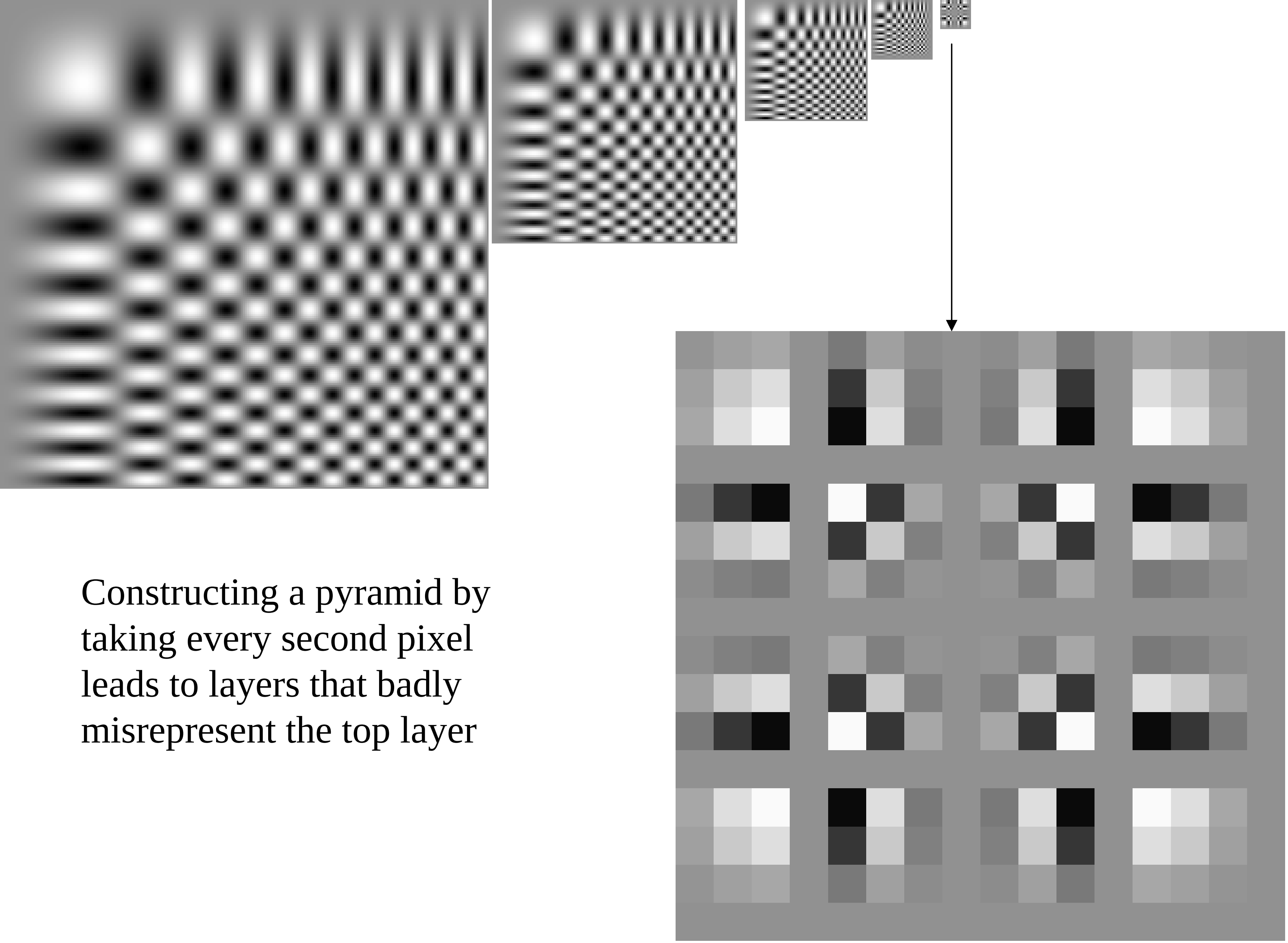

Signal
Fourier Transform
Magnitude Spectrum
Sample
Copy and Shift
Sampled Signal
Fourier Transform
Magnitude Spectrum
Cut out by multiplication with box filter
Inaccurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum

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.

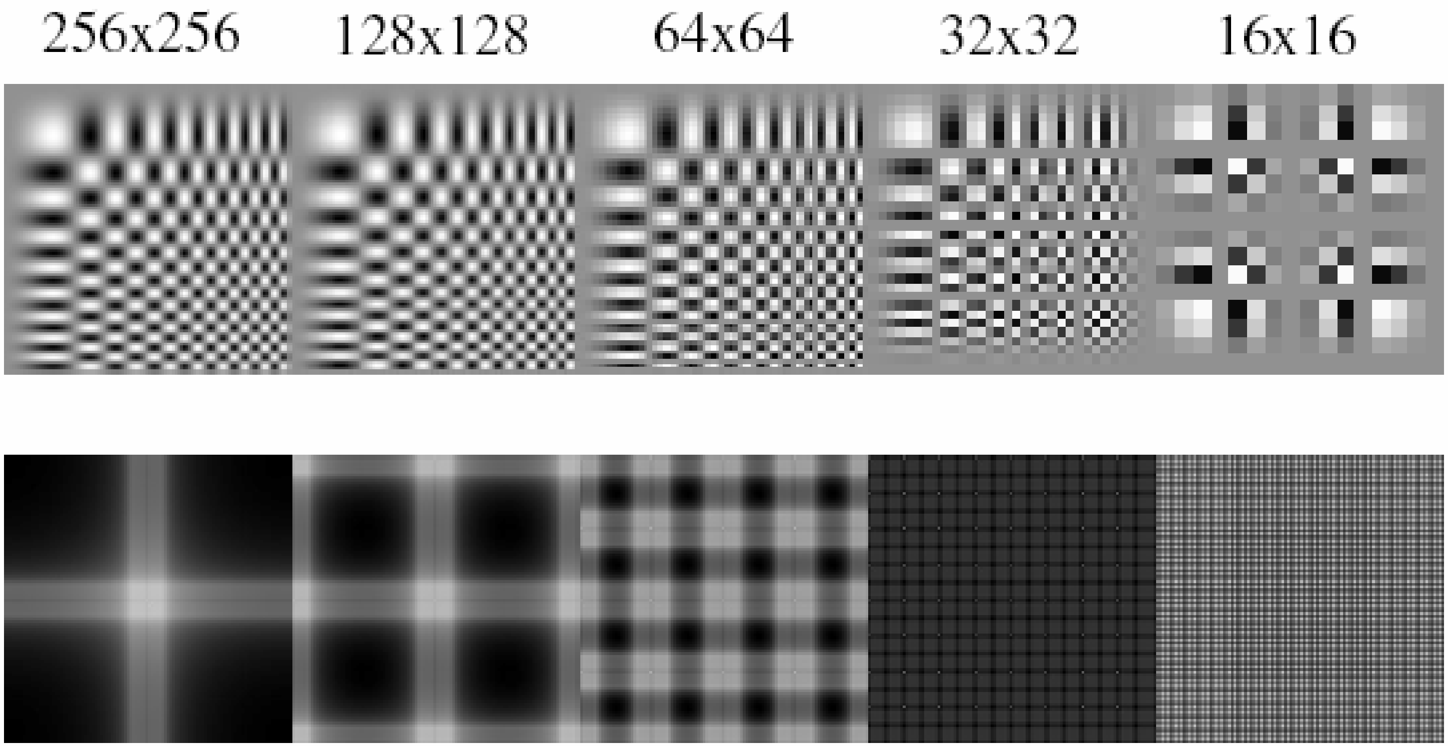


Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer

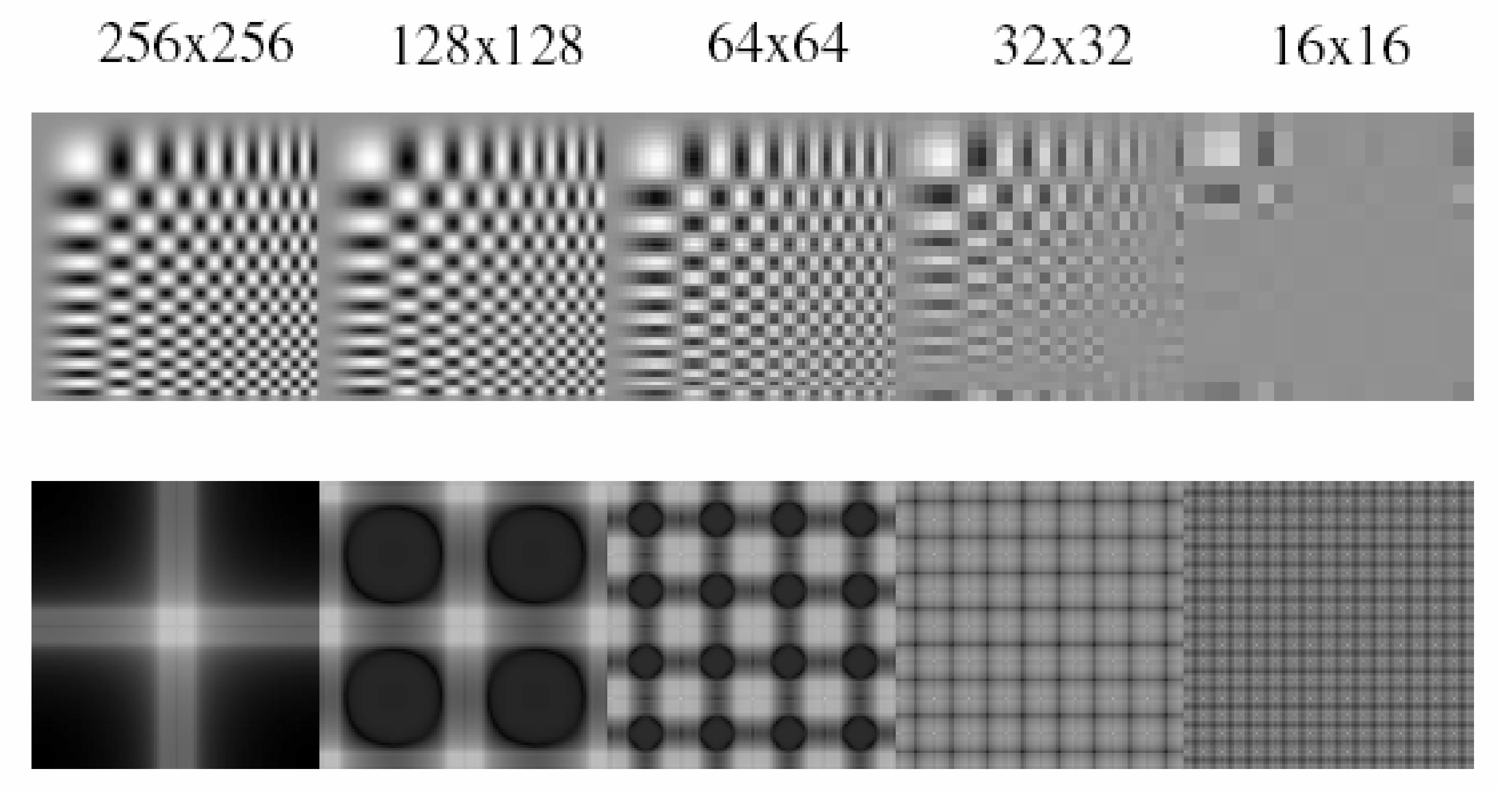
Smoothing as low-pass filtering

- 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.

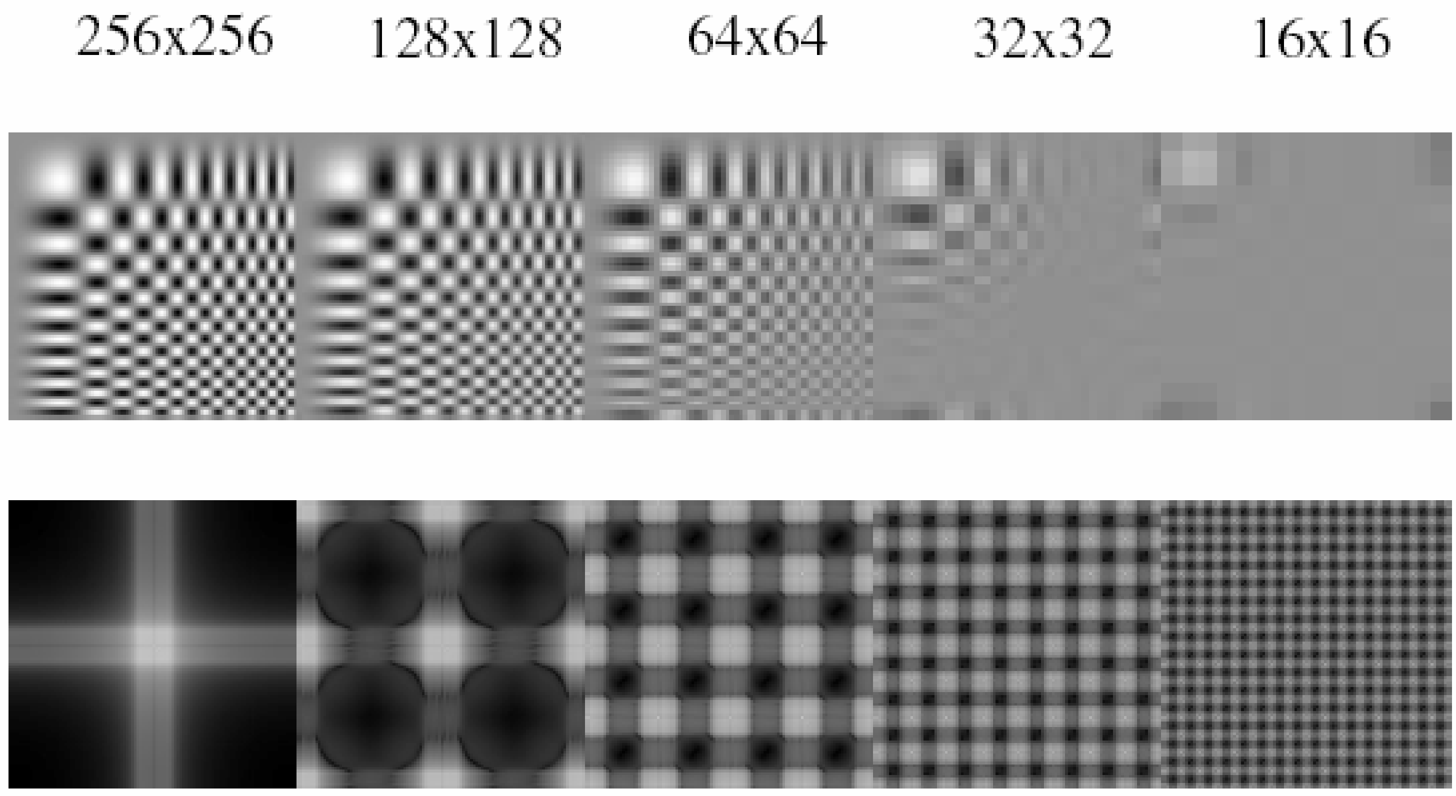
Sampling without smoothing. Top row shows the images, sampled at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.



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.

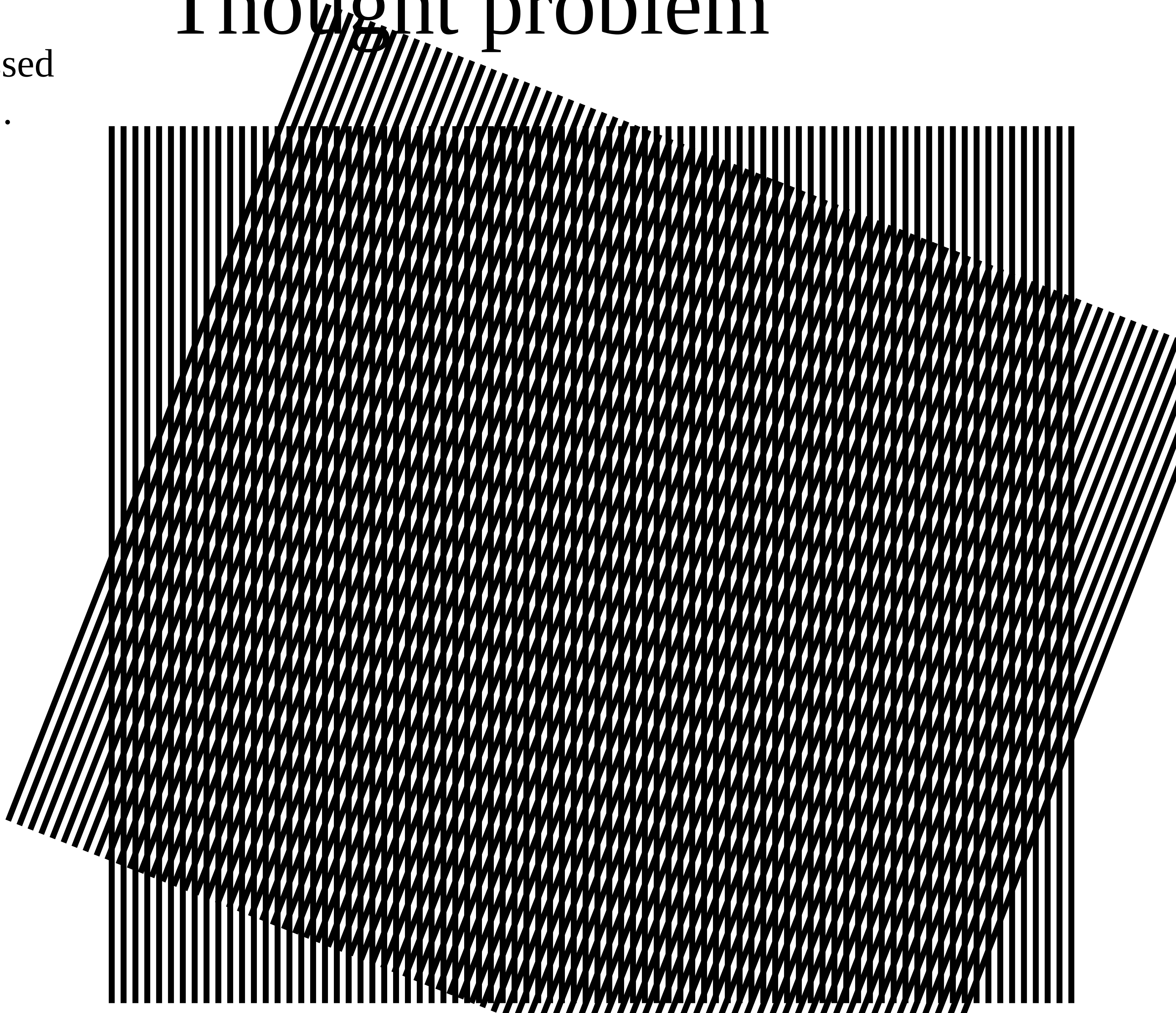


Sampling with smoothing. Top row shows the images. We get the next image by smoothing the image with a Gaussian with sigma 1.4 pixels, then sampling at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.



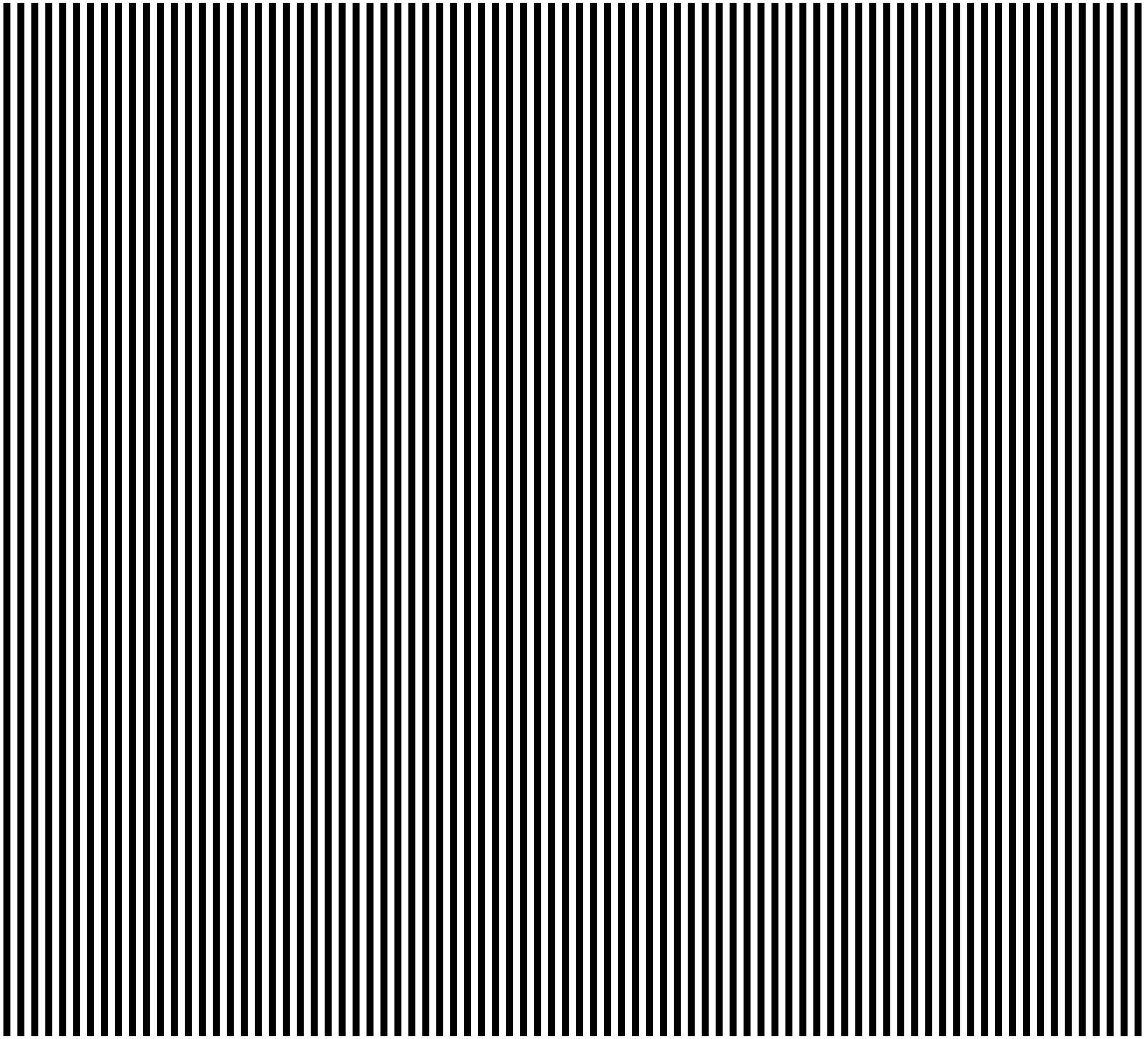
Thought problem

Analyze crossed gratings…



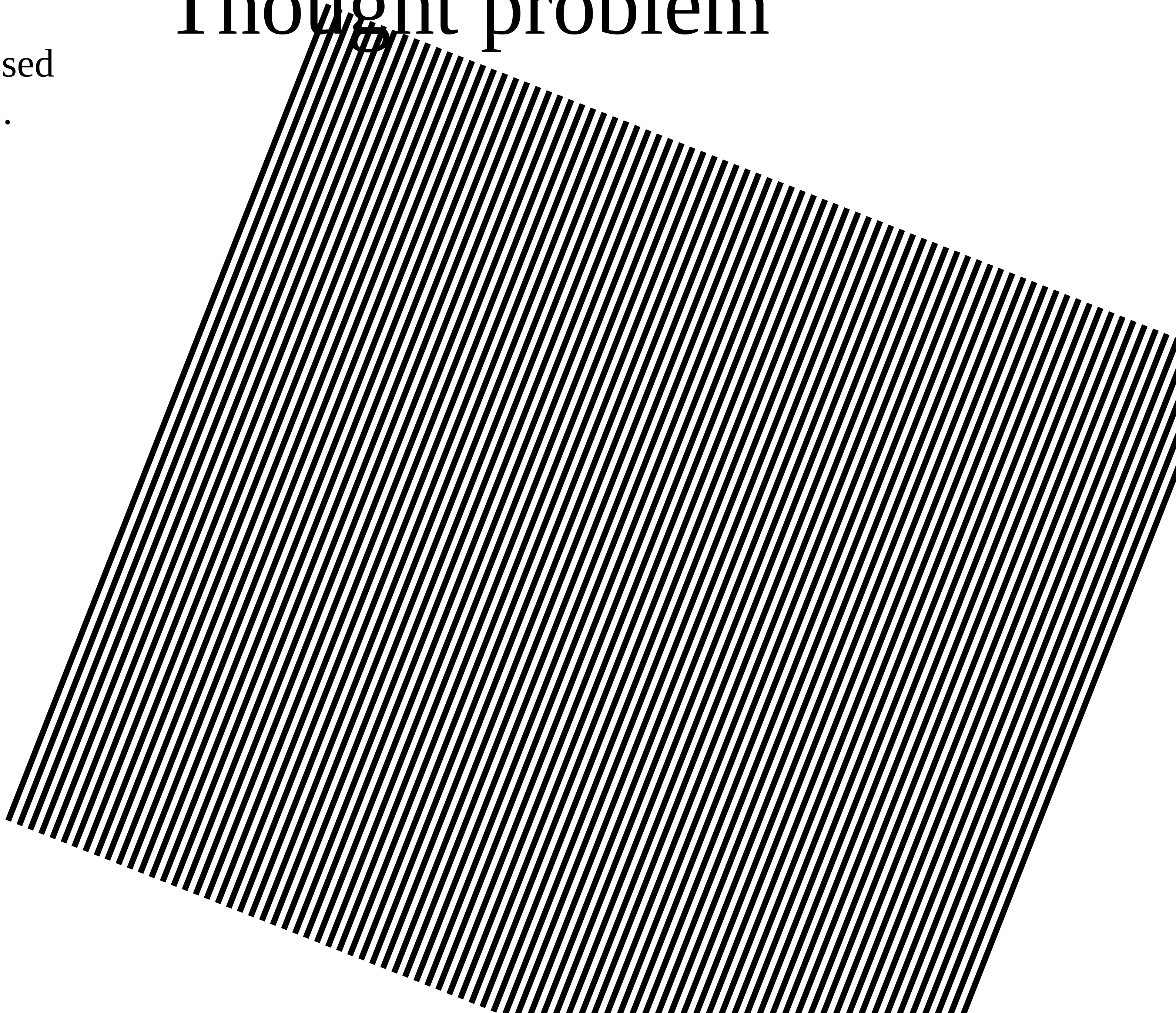
Thought problem

Analyze crossed gratings…



Thought problem

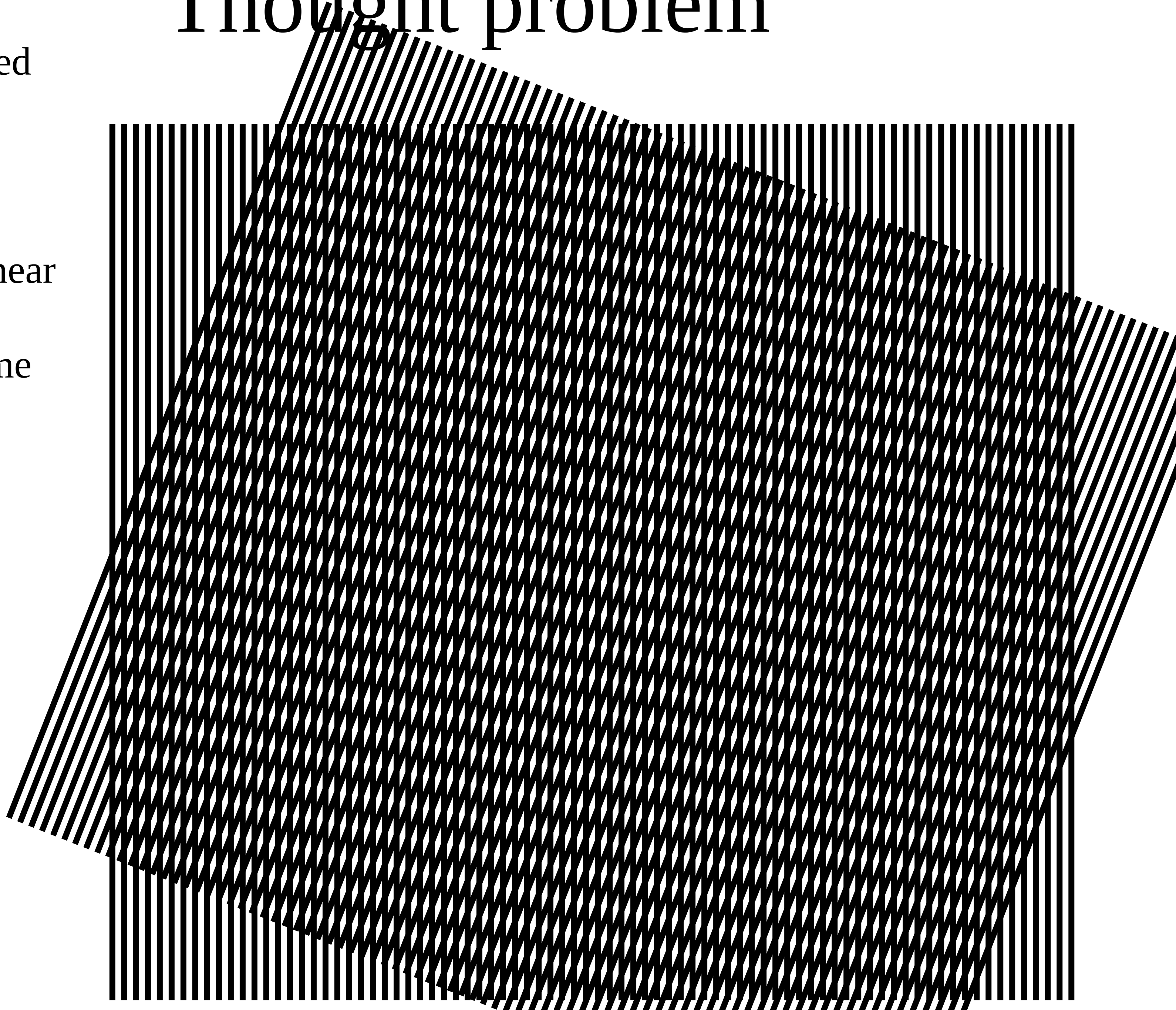
Analyze crossed gratings…

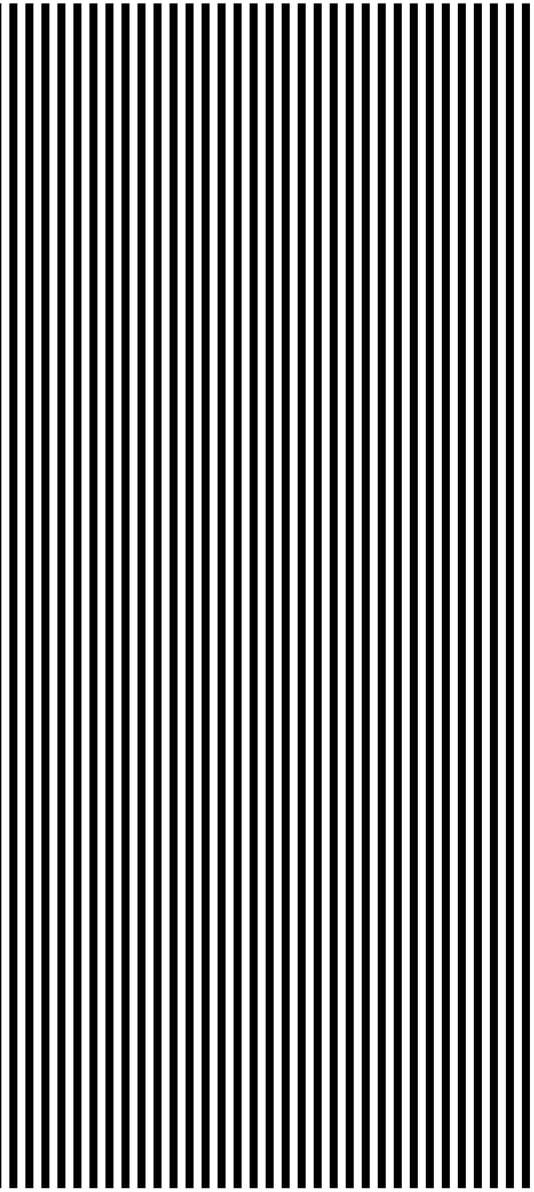


Thought problem

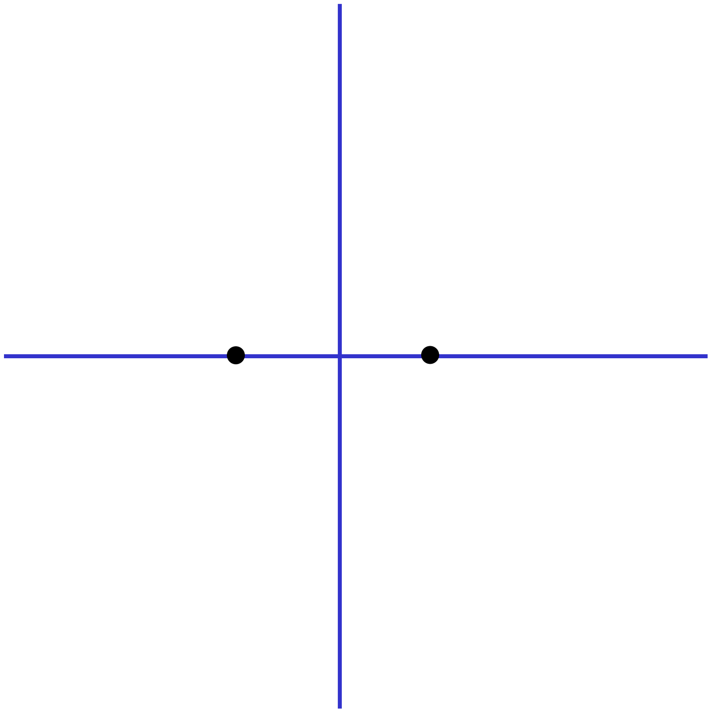
Analyze crossed gratings…

Where does perceived near horizontal grating come from?

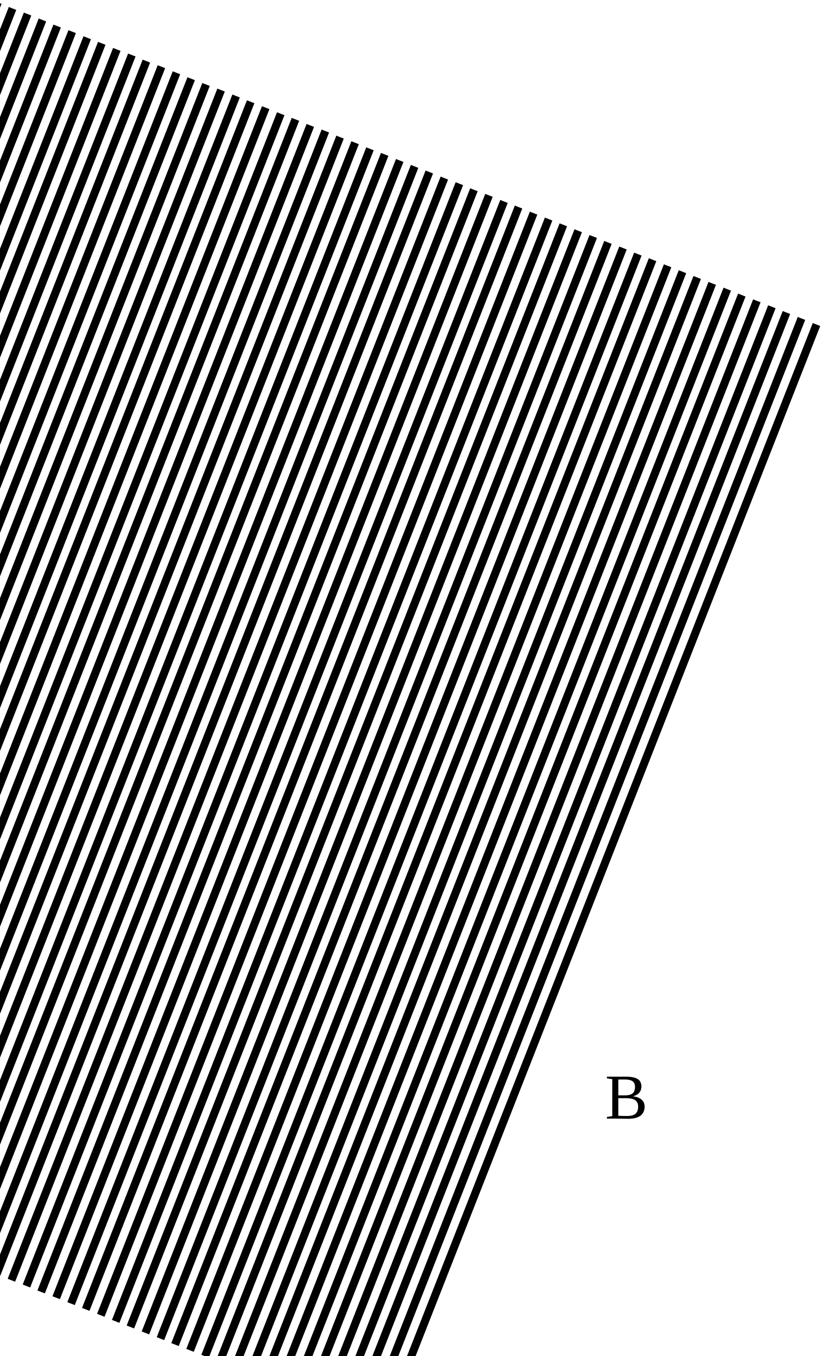




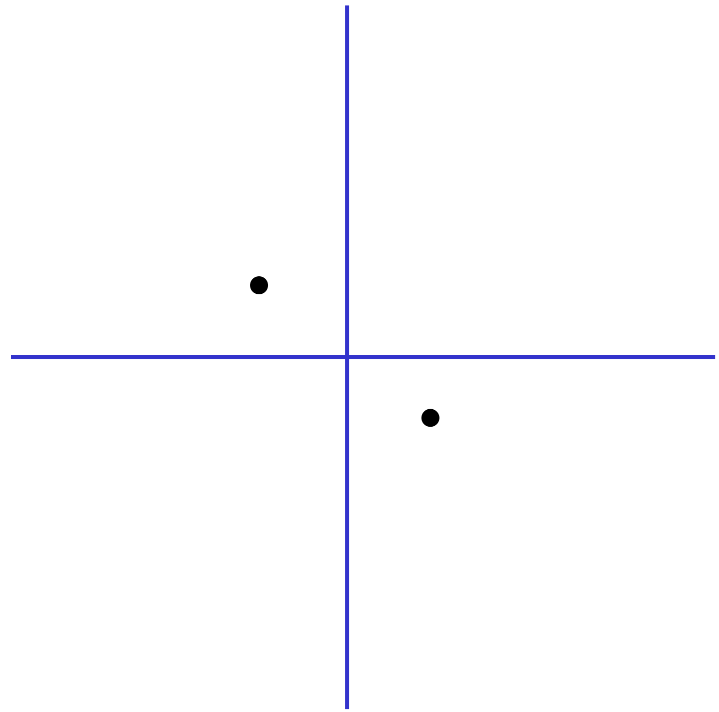
A



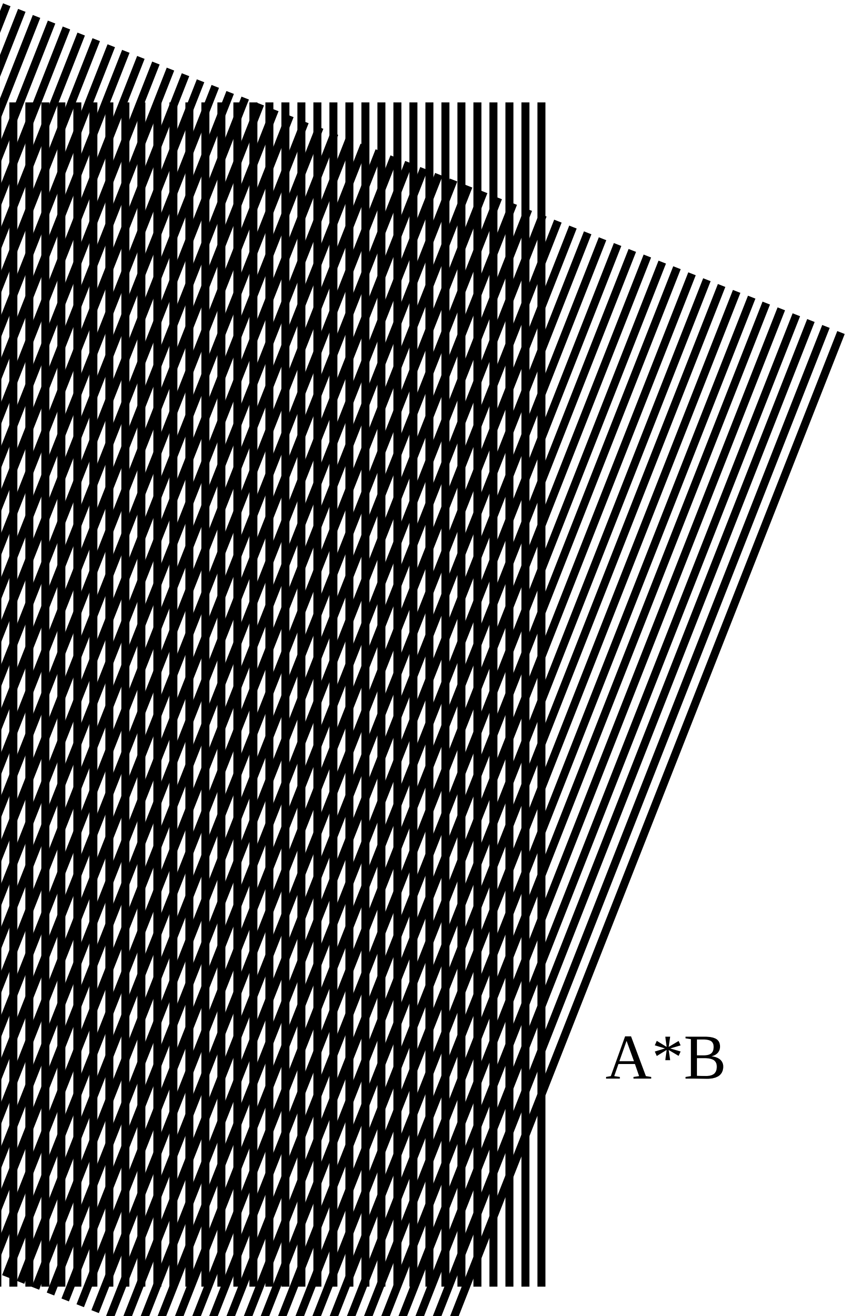
F(A)



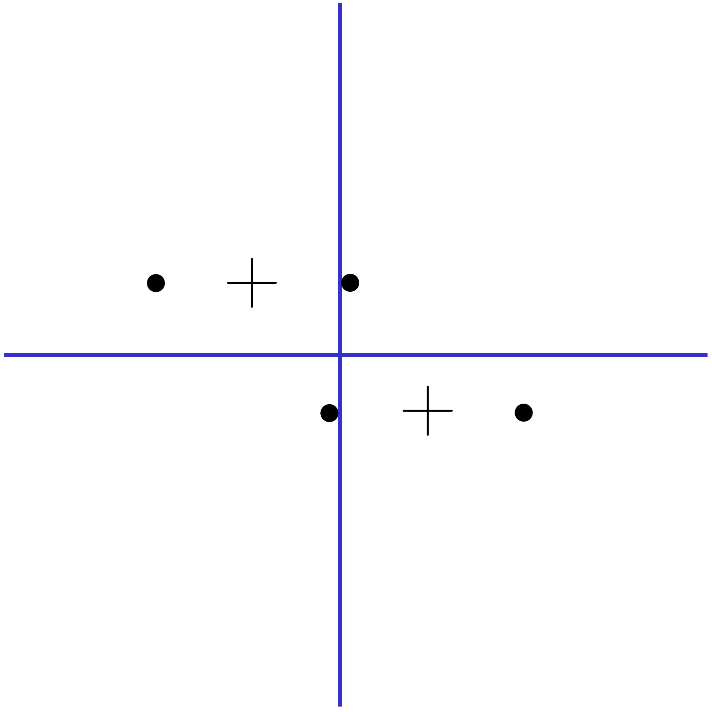
B



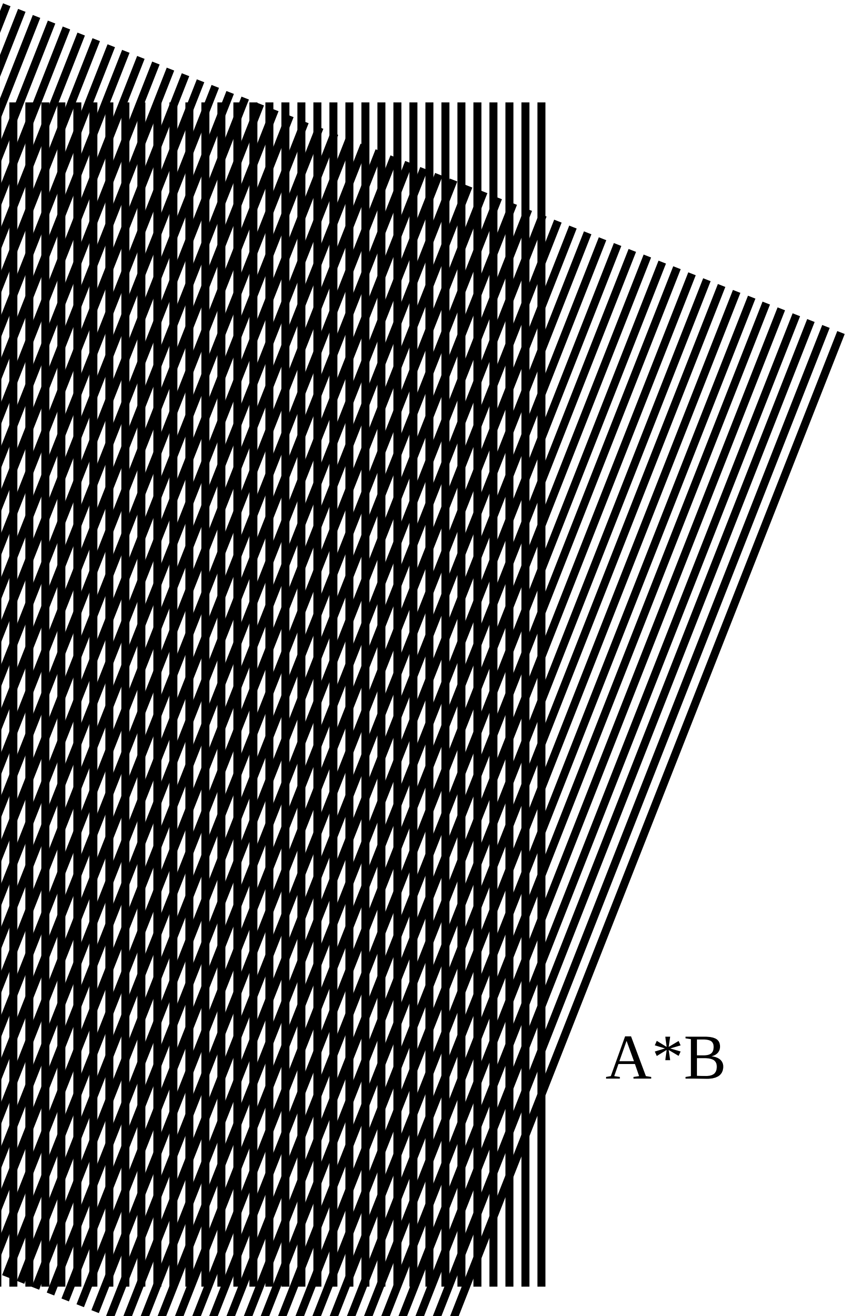
F(B)



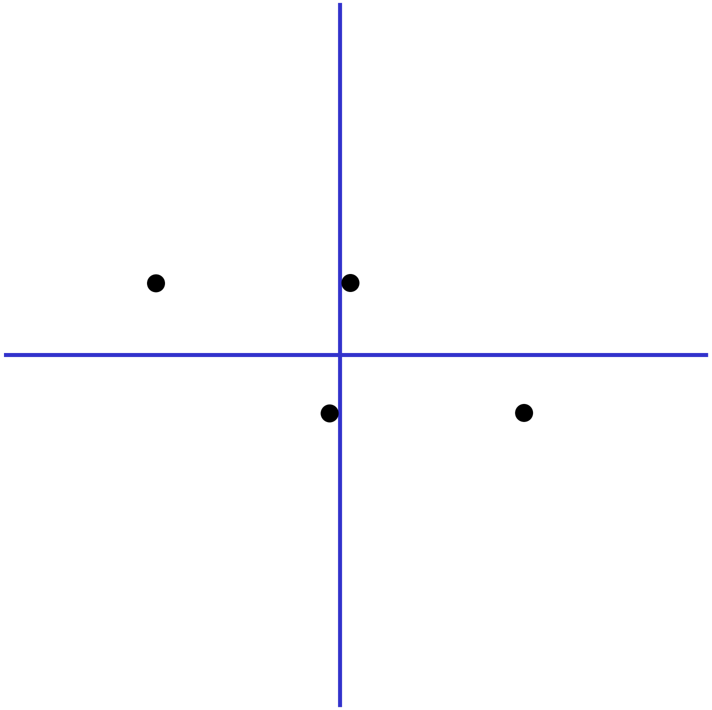
A*B



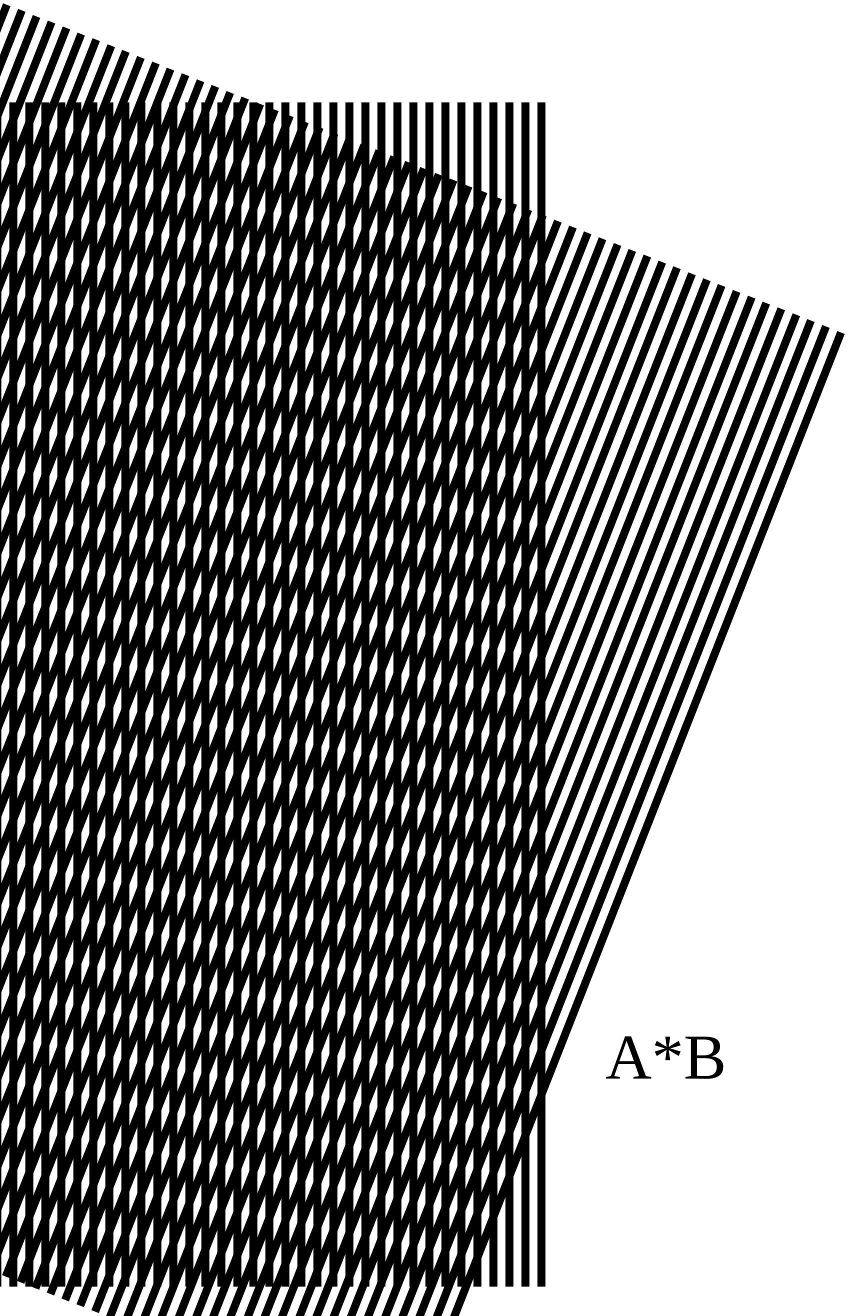
F(A)**F(B)



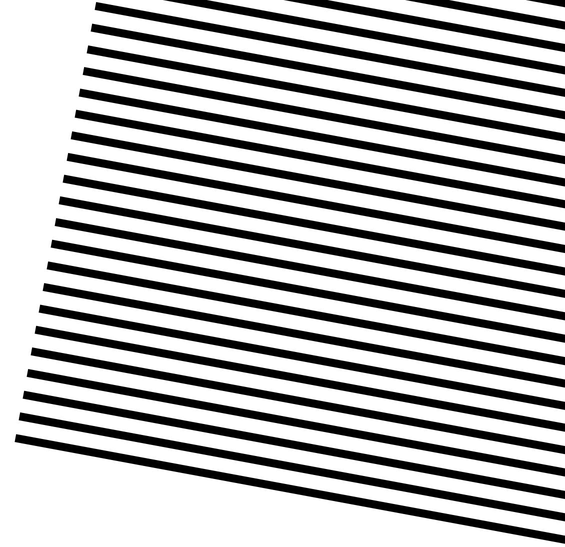
A*B



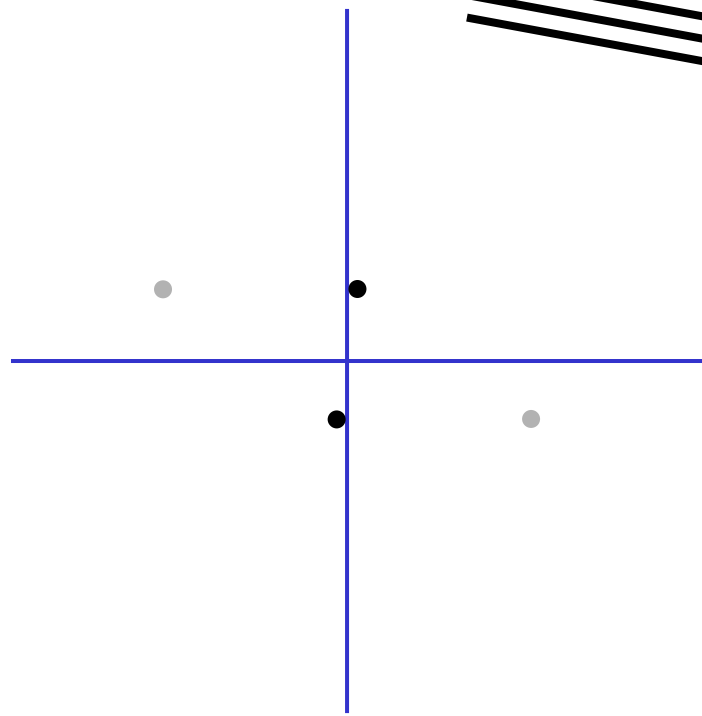
F(A)**F(B)



A*B



C



Lowpass(F(A)**F(B))
~=F(C)

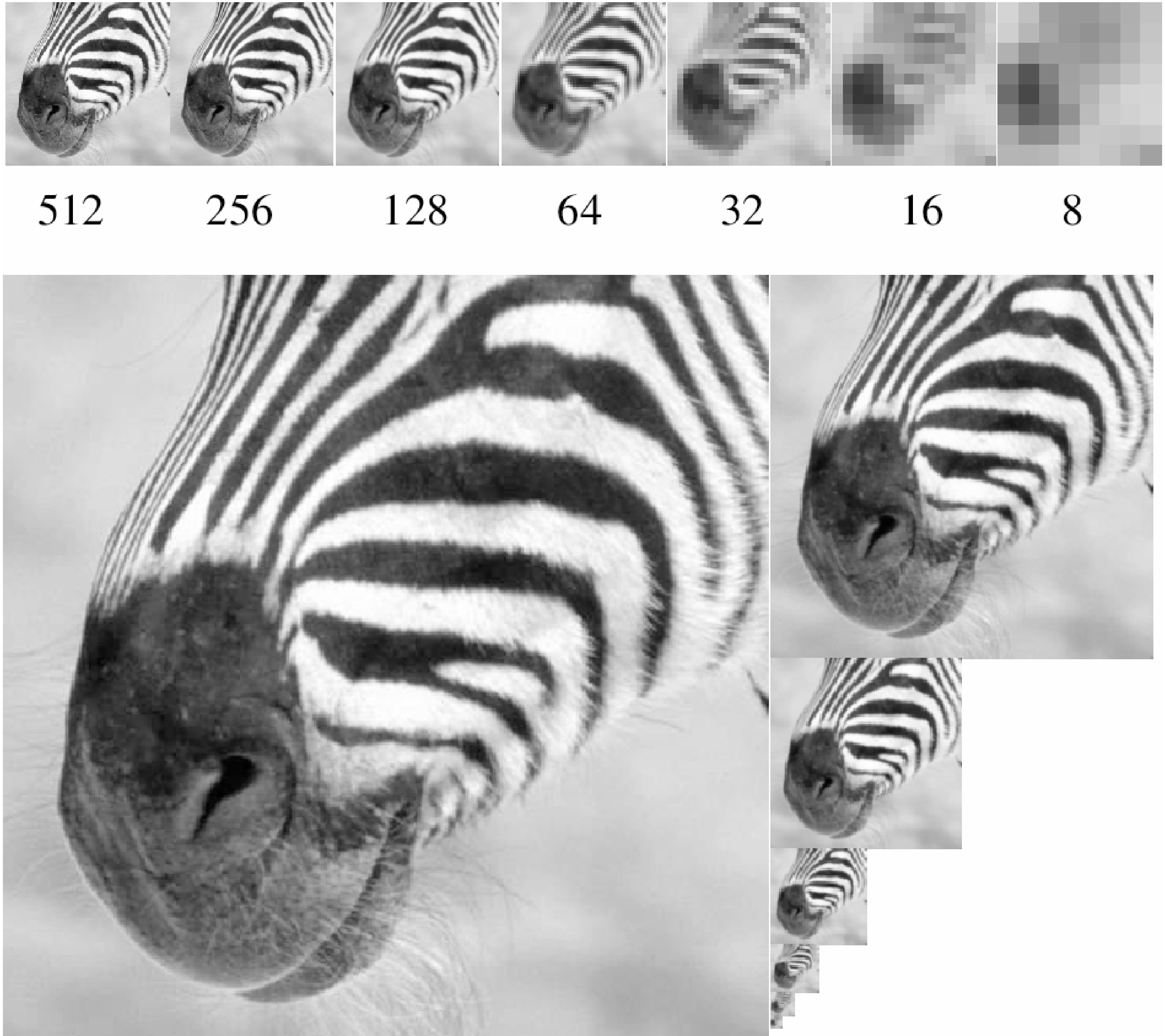
What is a good representation for image analysis?

- 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

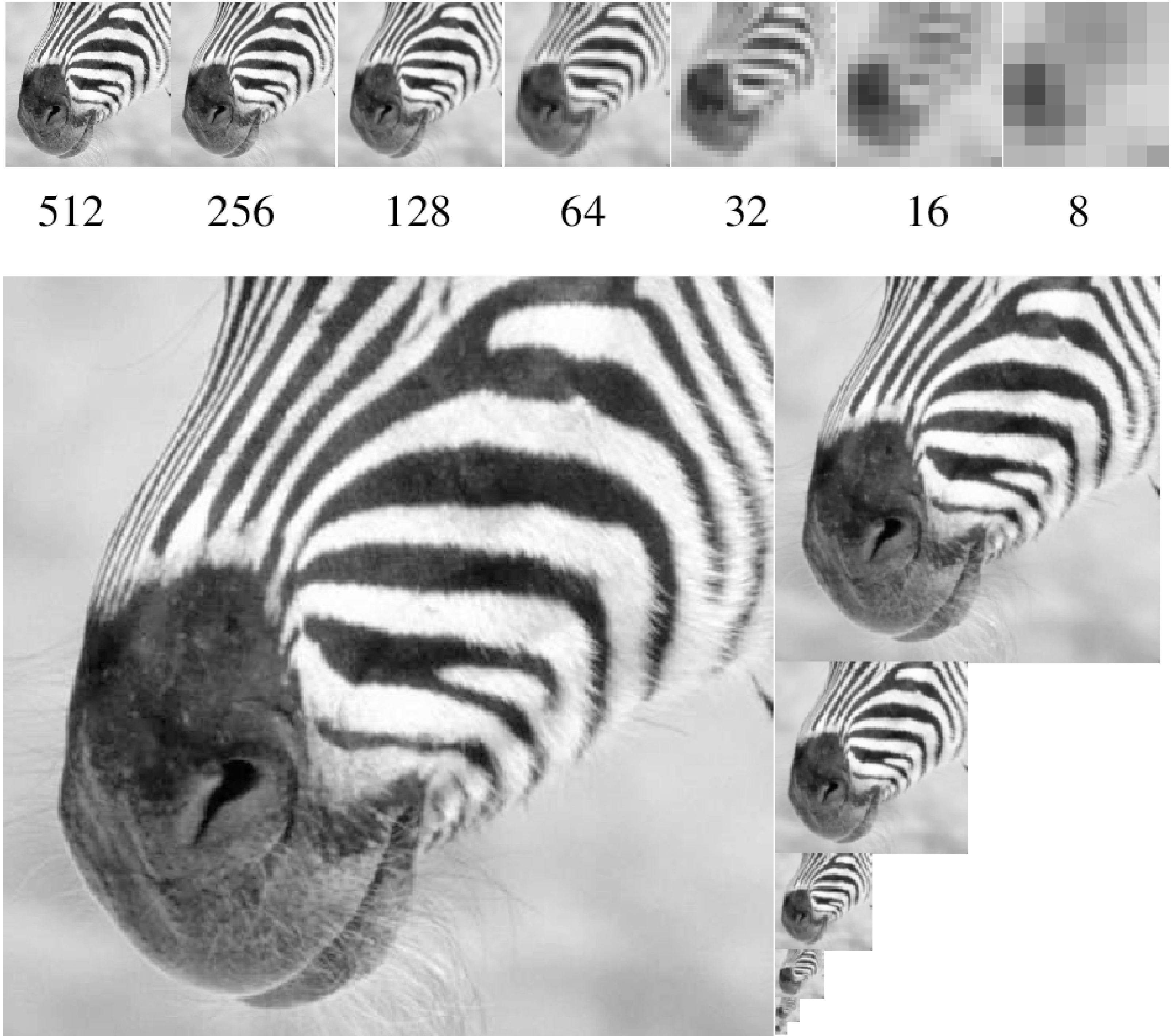
The Gaussian pyramid

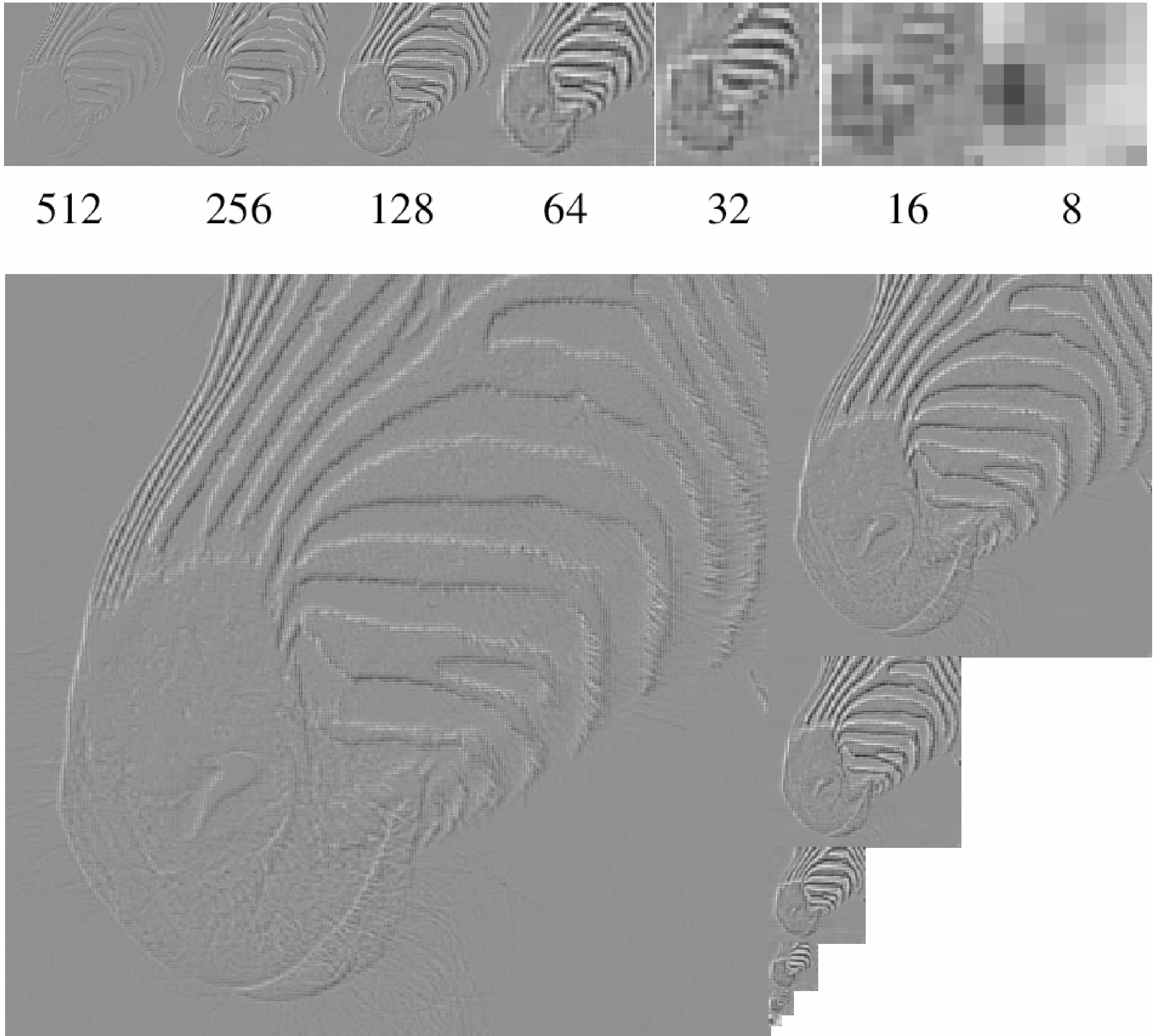
- Smooth with gaussians, because
  - a gaussian*gaussian=another gaussian
- Synthesis
  - smooth and sample
- Analysis
  - take the top image
- Gaussians are low pass filters, so repn is redundant

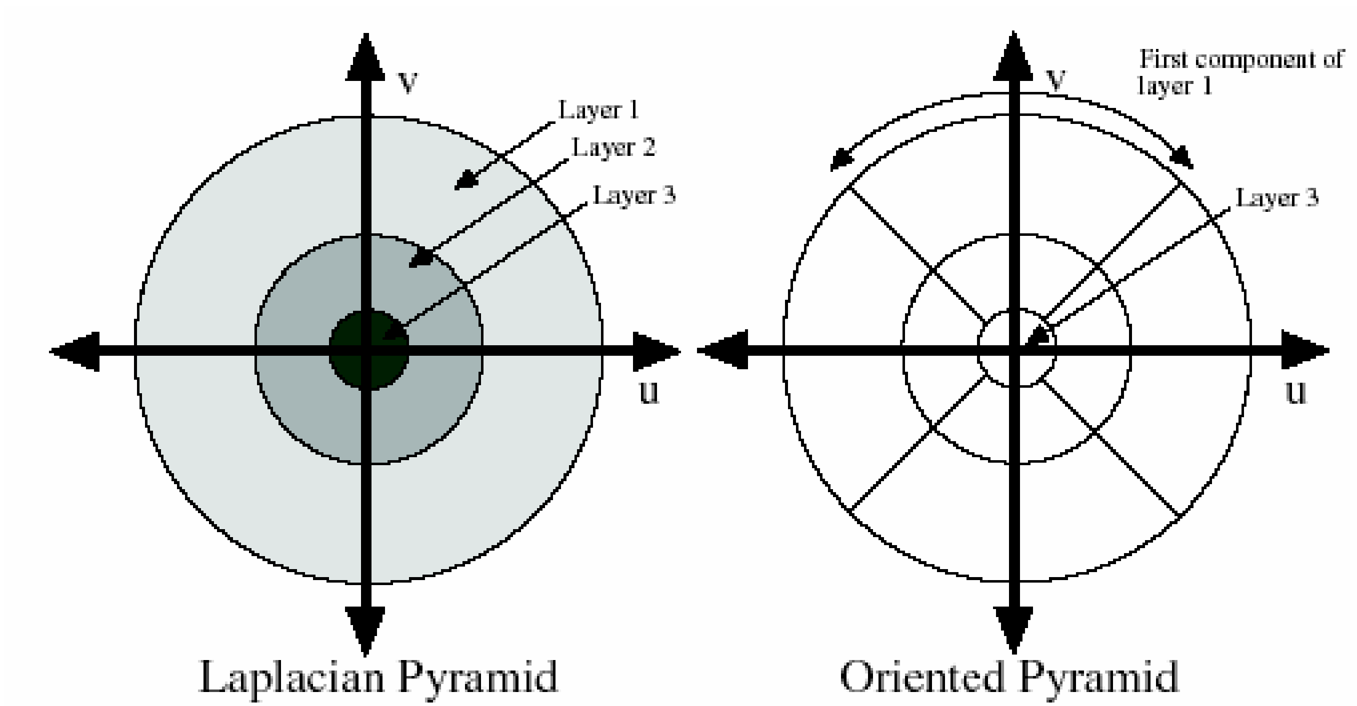

512
256
128
64
32
16
8

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

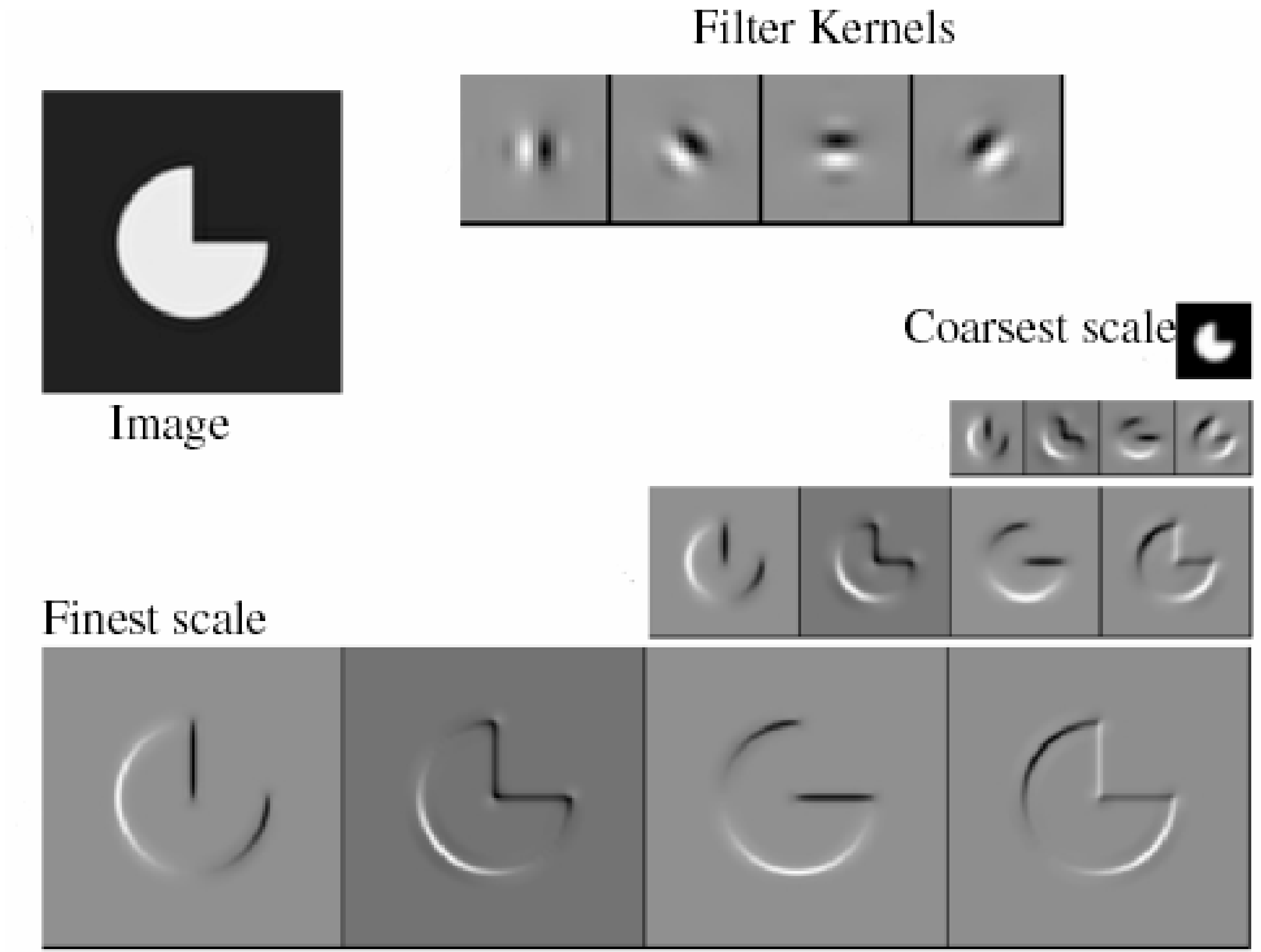

512
256
128
64
32
16
8


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