Color interpolation, image processing, and image representations

6.882

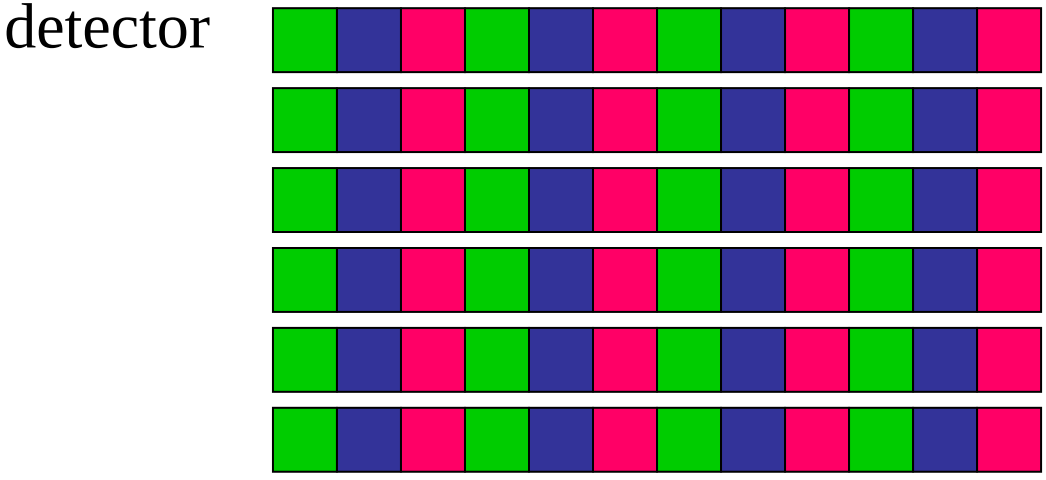
Bill Freeman and Fredo Durand

Feb. 23, 2006

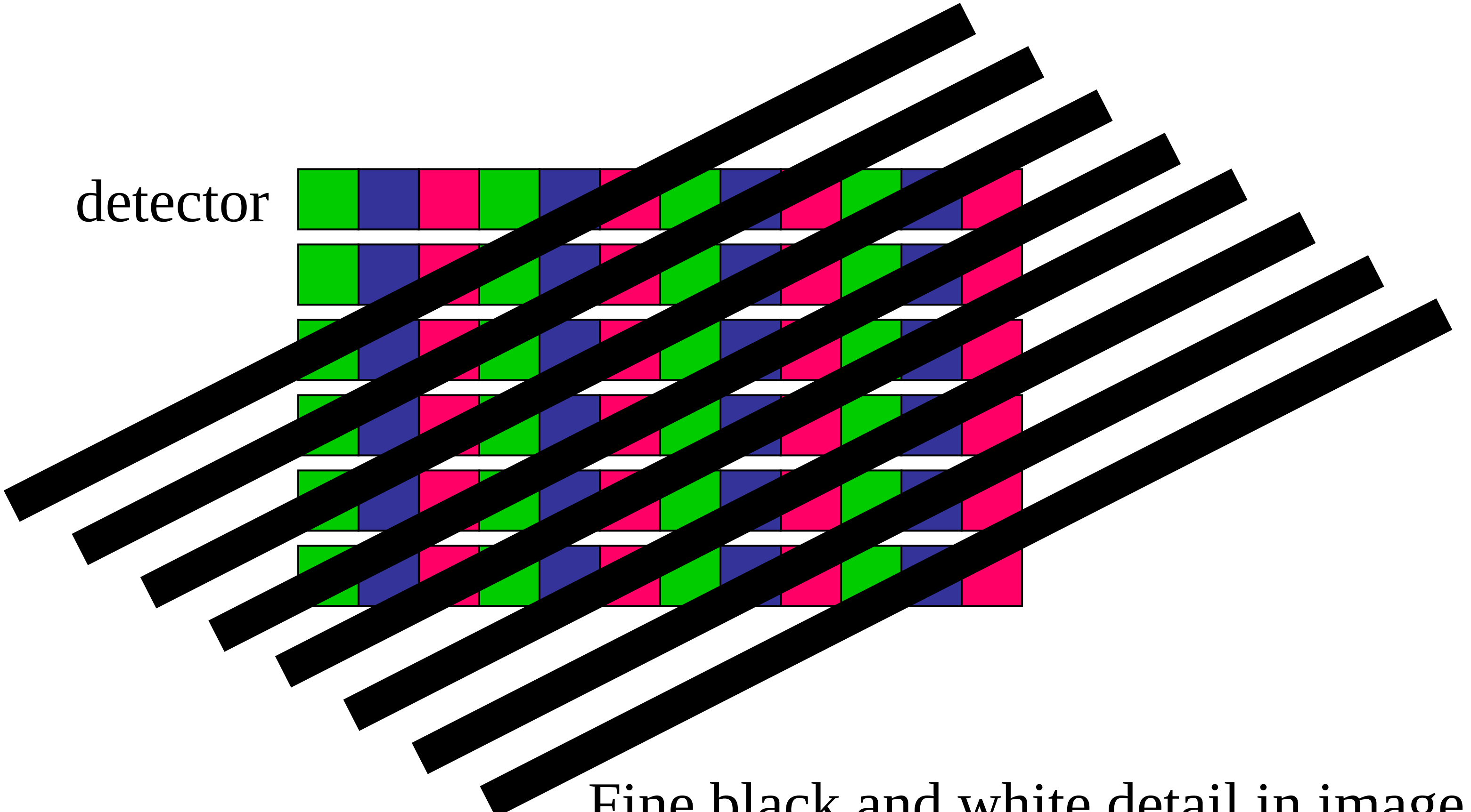
Initial announcements

- How was the homework?
- Bill will be miss office hours next week. Please feel free to e-mail to make an appointment at some different time.

CCD color filter pattern



The cause of color moire

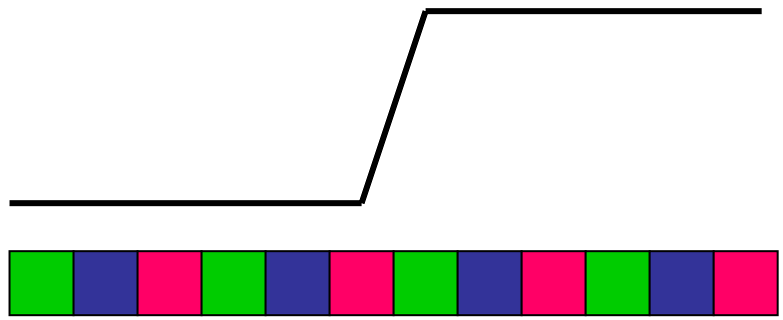


Fine black and white detail in image mis-interpreted as color information.

Black and white edge falling on color CCD detector

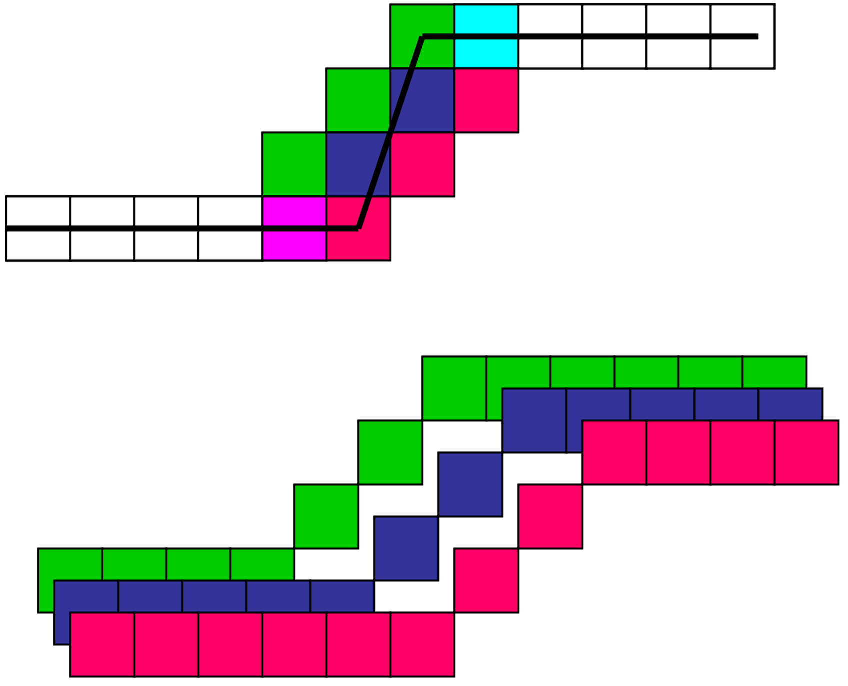
Black and white image (edge)

Detector pixel colors



(previous slides were the freq domain interpretation of aliasing. Here's the spatial domain interpretation.)

Color sampling artifact



Interpolated pixel colors,
for grey edge falling on colored
detectors (linear interpolation).

Typical color moire patterns

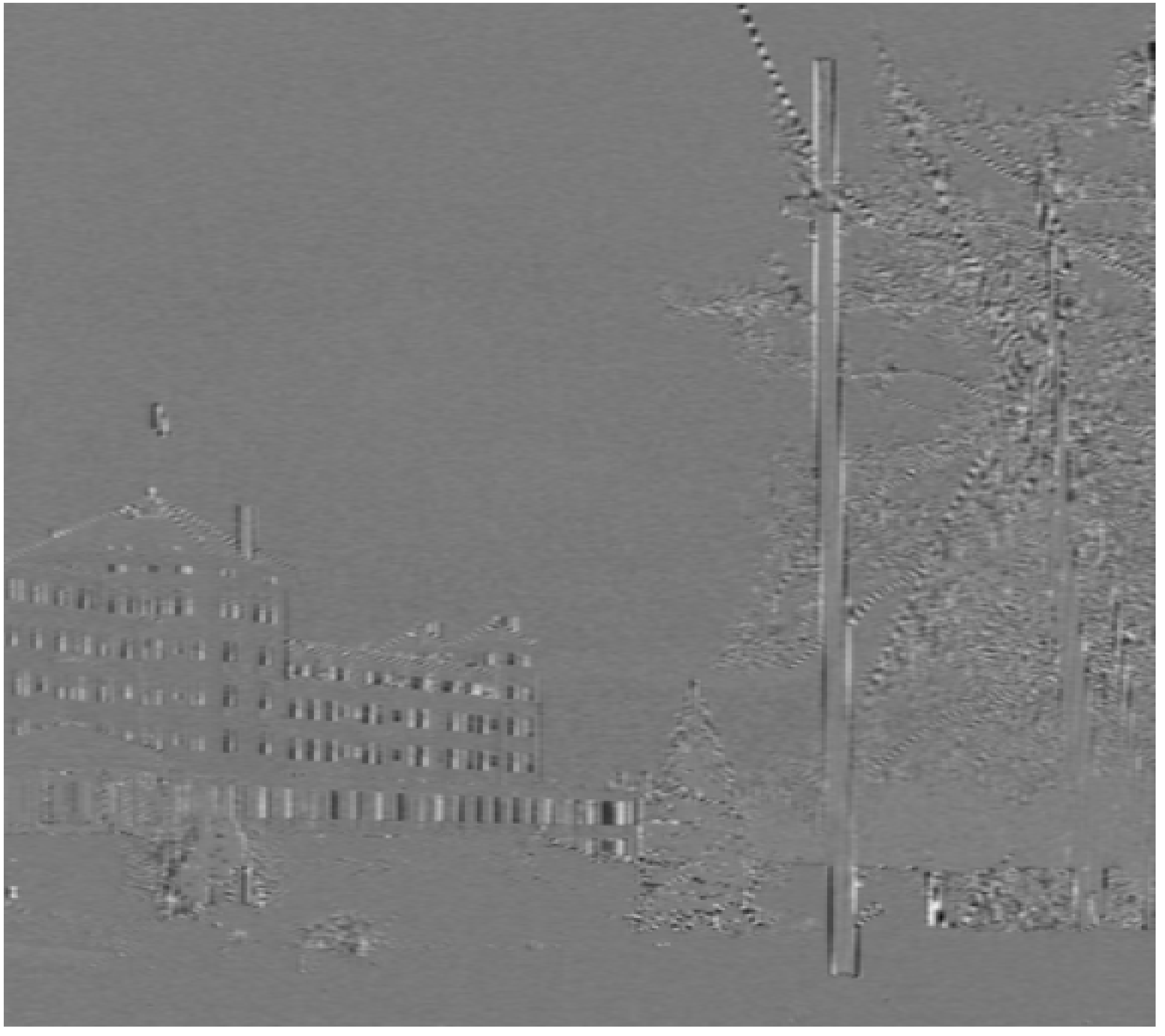
Blow-up of electronic camera image. Notice spurious colors in the regions of fine detail in the plants.

Color sampling artifacts

Motivation for median filter interpolation

The color fringe artifacts are obvious; we can point to them. Goal: can we characterize the color fringe artifacts mathematically? Perhaps that would lead to a way to remove them…

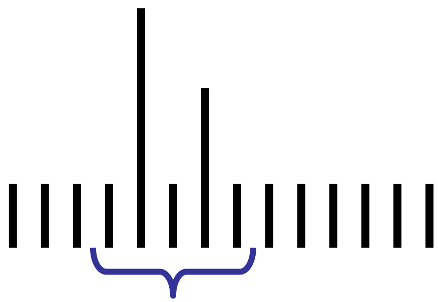
R-G, after linear interpolation



Median filter

Replace each pixel by the median over N pixels (5 pixels, for these examples). Generalizes to “rank order” filters.

In:

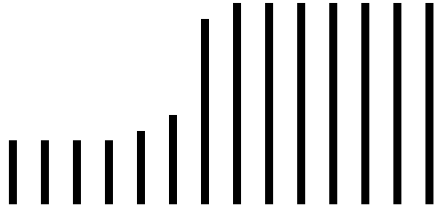


5-pixel neighborhood

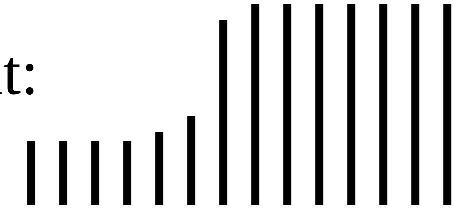
Out:

Spike noise is removed

In:



Out:



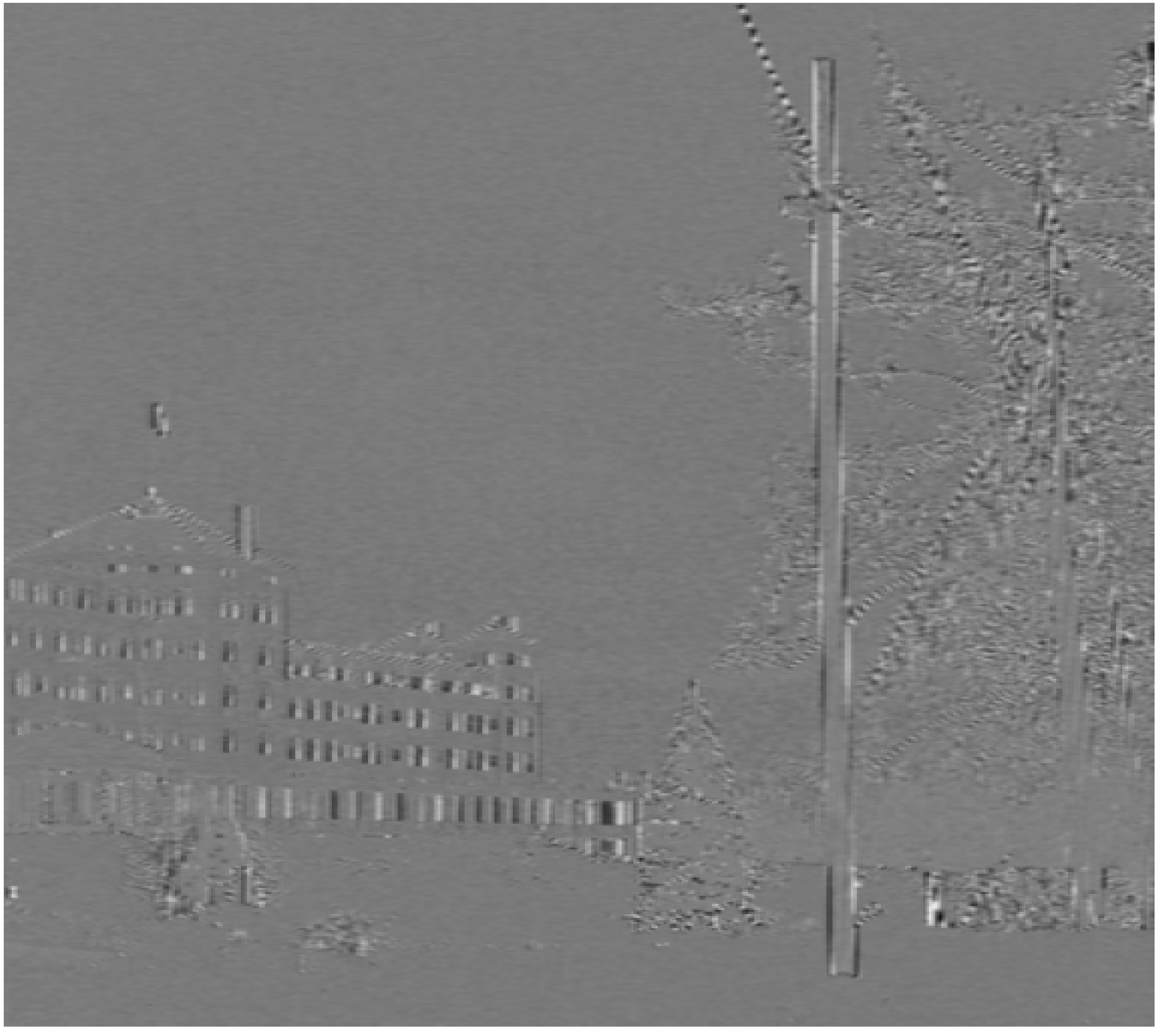
Monotonic edges remain unchanged

Degraded image

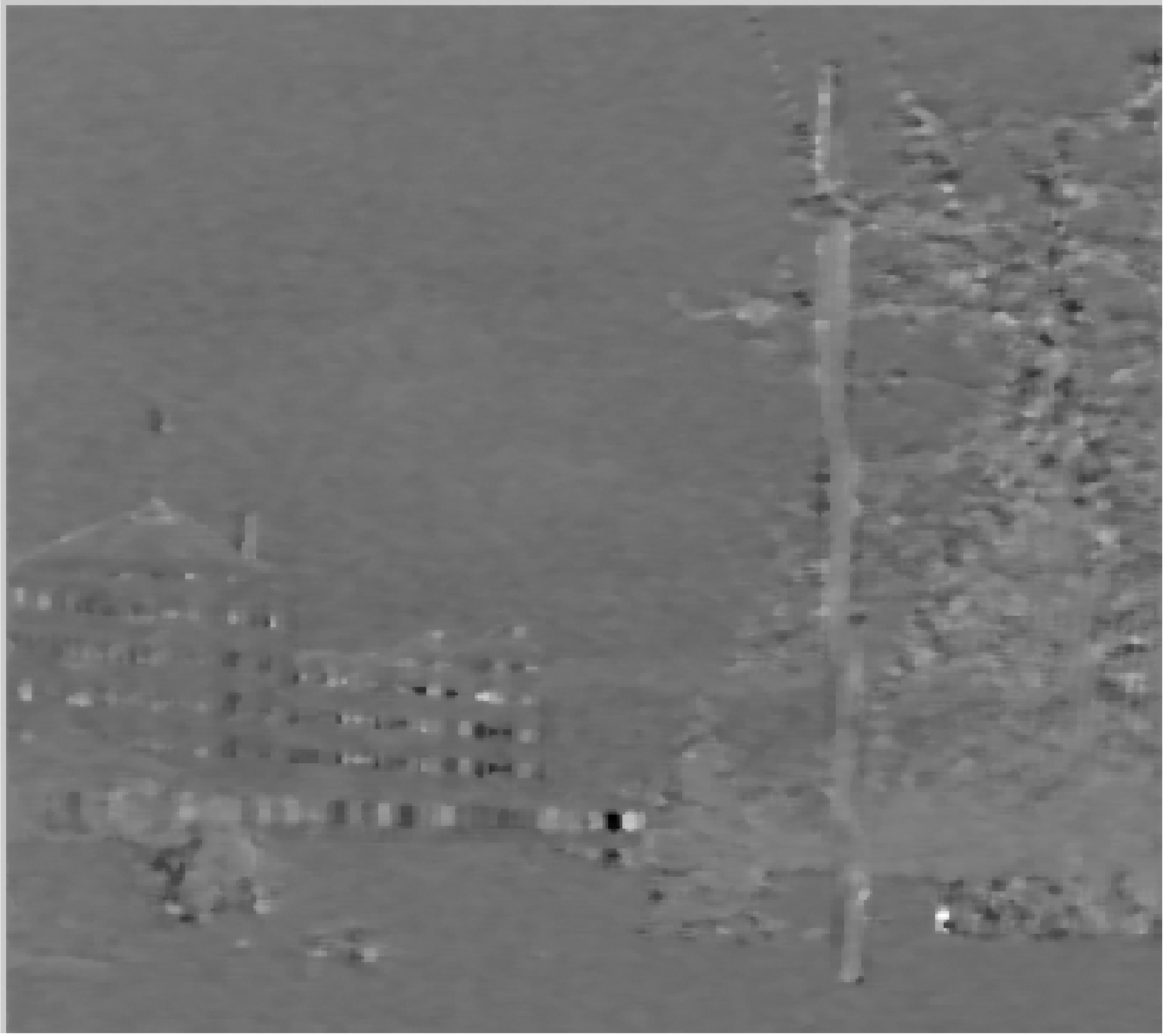
Radius 1 median filter

Radius 2 median filter

R-G, after linear interpolation



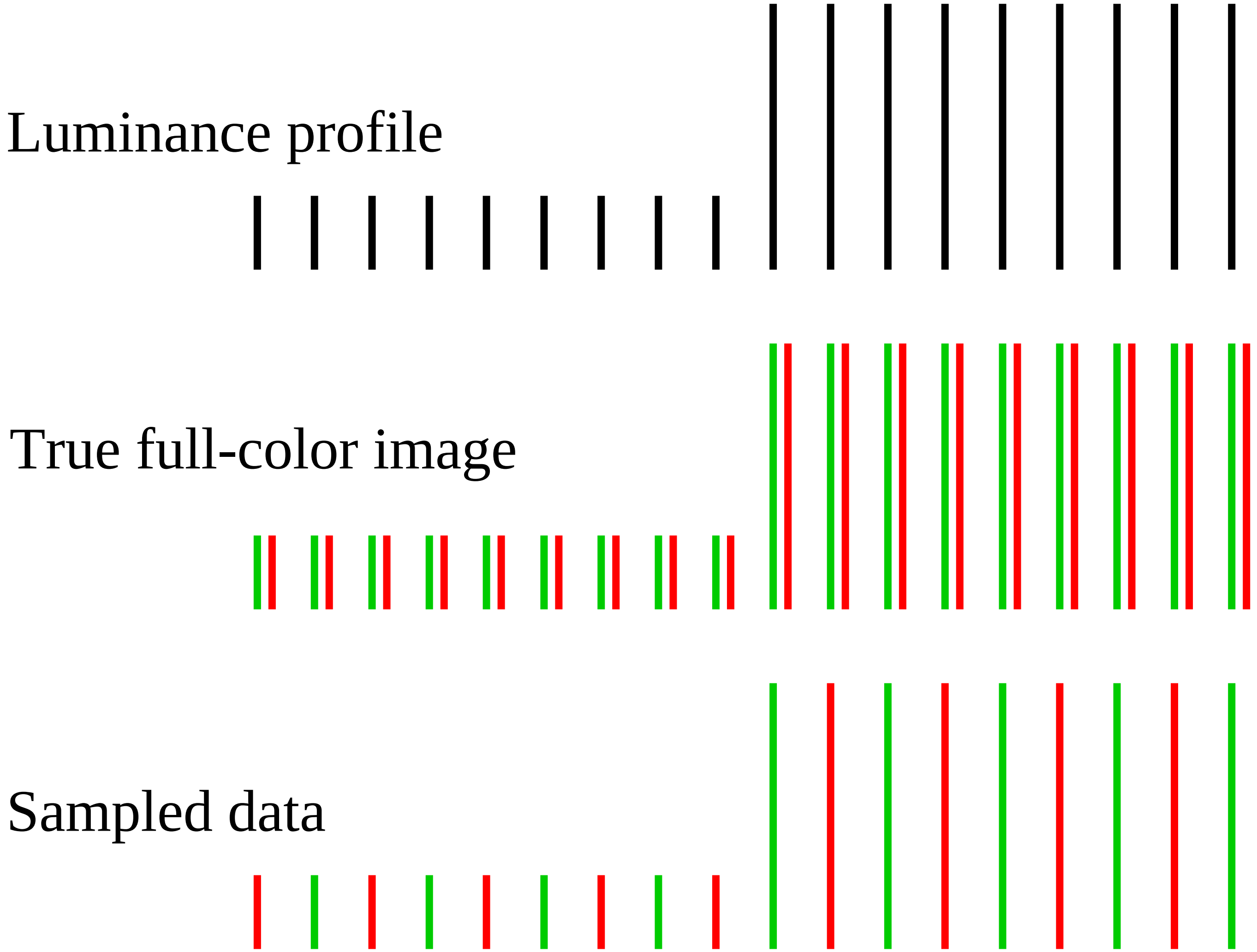
R – G, median filtered (5x5)



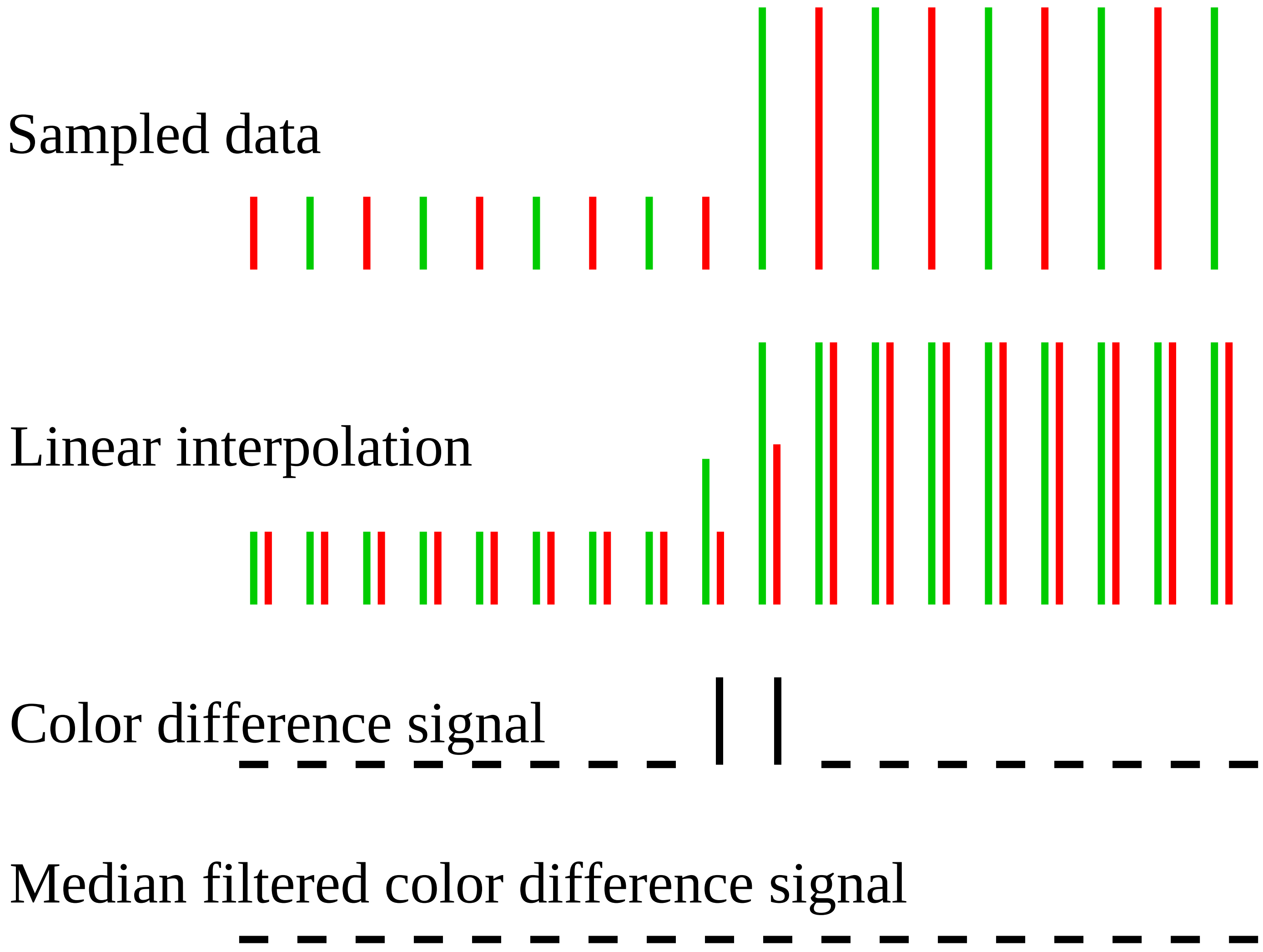
Median Filter Interpolation

- Perform first interpolation on isolated color channels.
- Compute color difference signals.
- Median filter the color difference signal.
- Reconstruct the 3-color image.

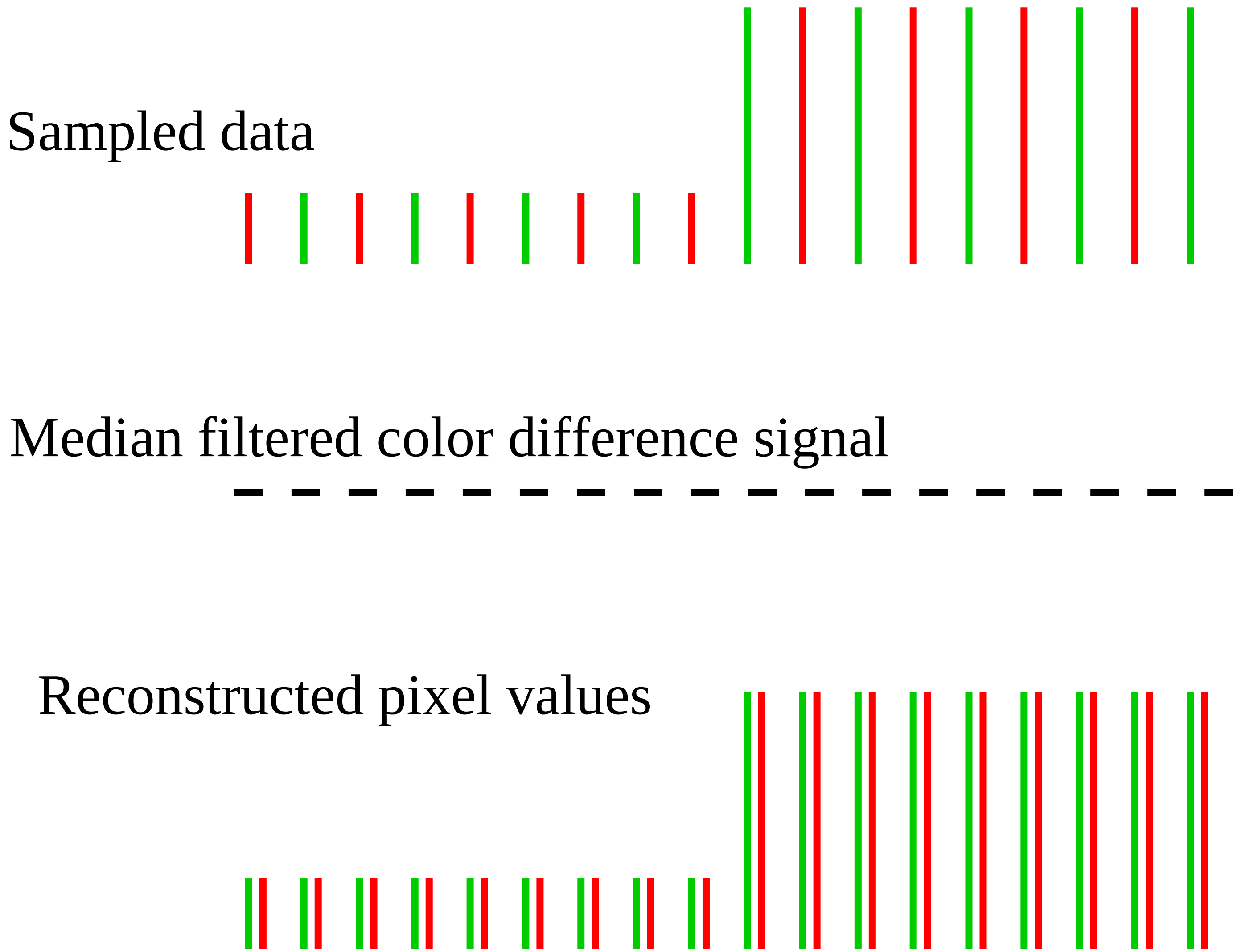
Two-color sampling of BW edge



Two-color sampling of BW edge



Two-color sampling of BW edge



Recombining the median filtered colors

Linear interpolation

Median filter interpolation

Beyond linear interpolation between samples of the same color

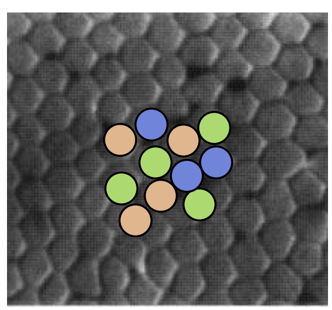
- Luminance highs
- Median filter interpolation
  - U.S. Patent 4,663,655
- Regression
- Gaussian method
- Regression, including non-linear terms
  - http://www1.cs.columbia.edu/CAVE/publications/pdfs/Schechner_ICCV01.pdf
- Multiple linear regressors
  - http://people.csail.mit.edu/billf/papers/cvpr04tappen.pdf
- Perceptual study
  - http://color.psych.upenn.edu/brainard/papers/LongereIEEE.pdf

Project ideas

(1) Develop a new color interpolation algorithm

(2) Study the tradeoffs in sensor color choice for image reconstruction:

Human vision uses randomly placed, very unsaturated color sensors. Cameras typically use regularly spaced, saturated color sensors. Which is better; why?



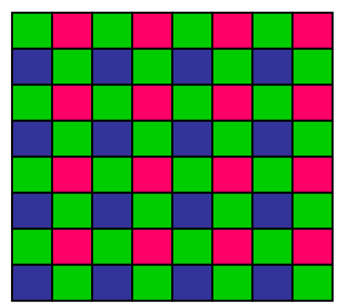


Image filtering

- Reading:
  - Chapter 7, Forsyth and Ponce
  - Oppenheim, Shafer, and Buck
  - Oppenheim and Willsky

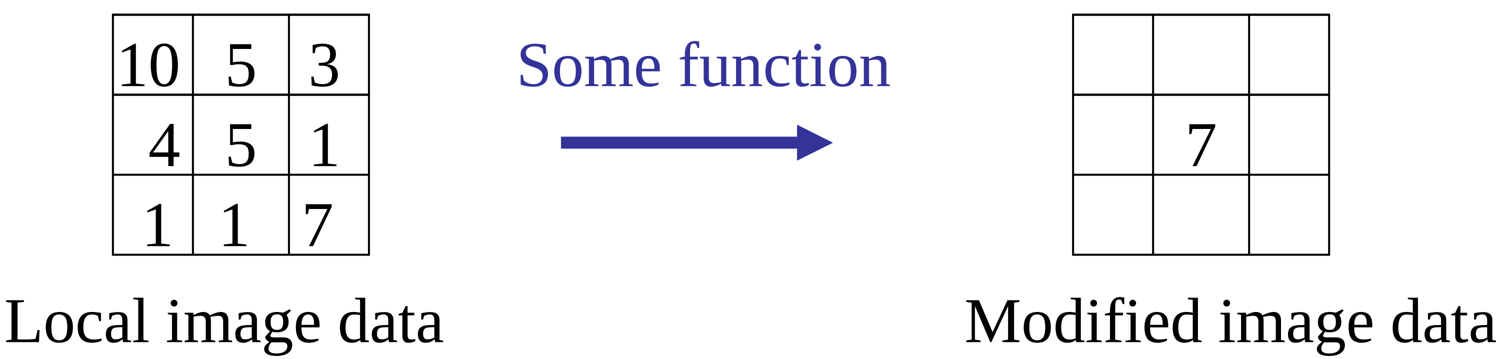
Take 6.341, discrete-time signal processing

- 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.



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

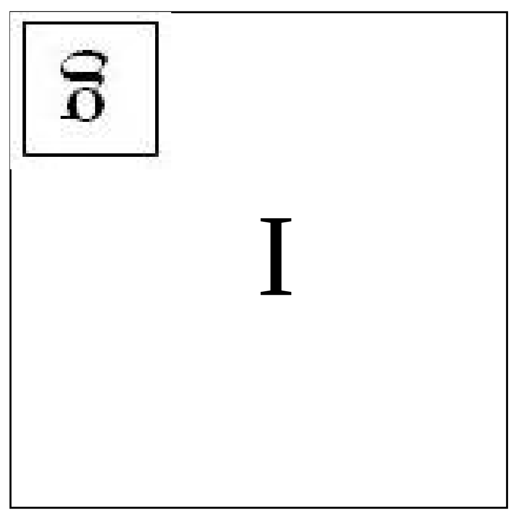
kernel

	7	

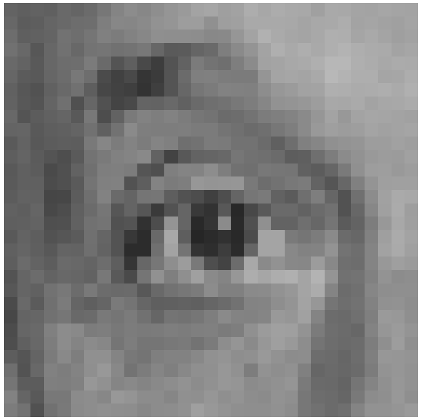
Modified image data

Convolution

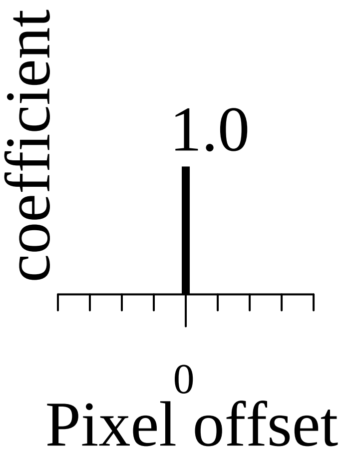
$$f[m,n] = I \otimes g = \sum_{k,l} I[m-k, n-l]\, g[k,l]$$



Linear filtering (warm-up slide)

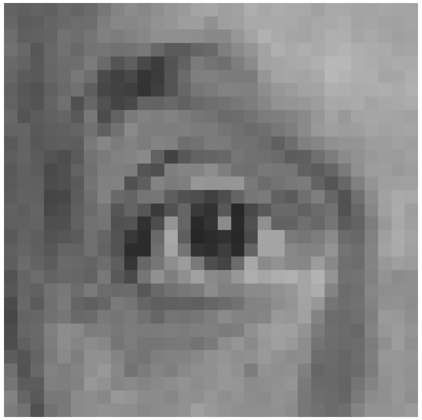


original

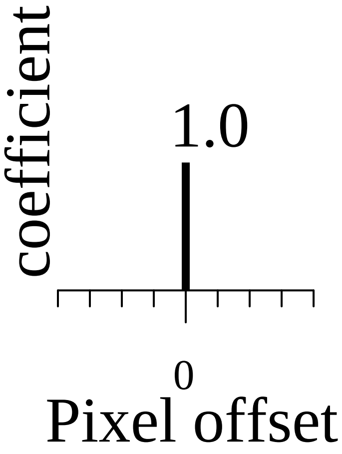


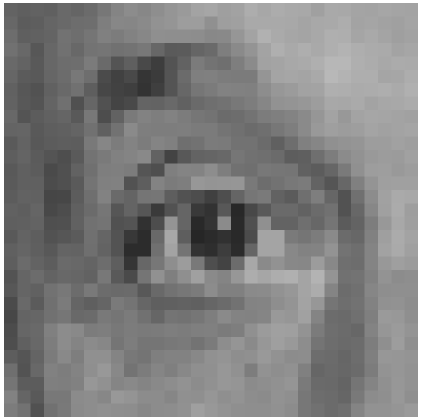
?

Linear filtering (warm-up slide)



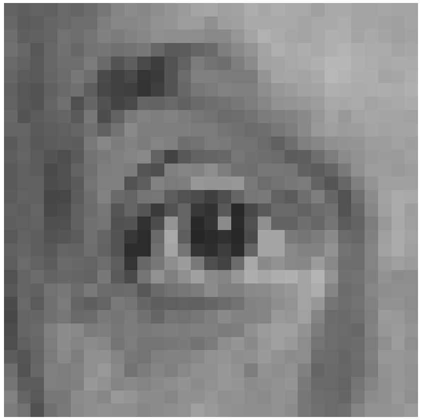
original



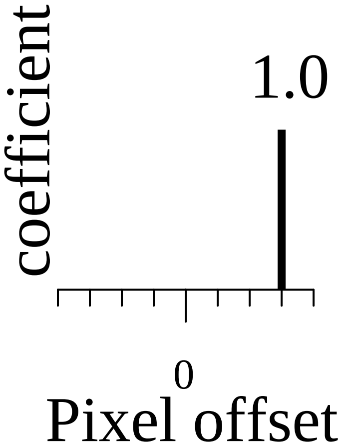


Filtered
(no change)

Linear filtering

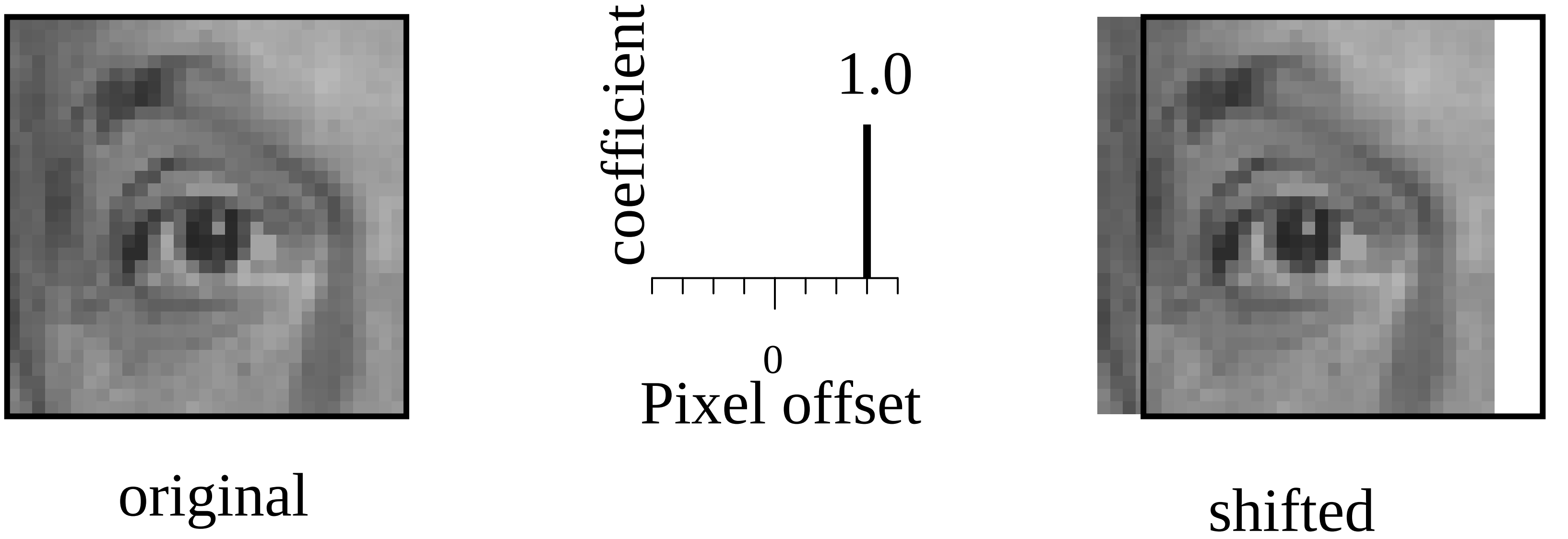


original

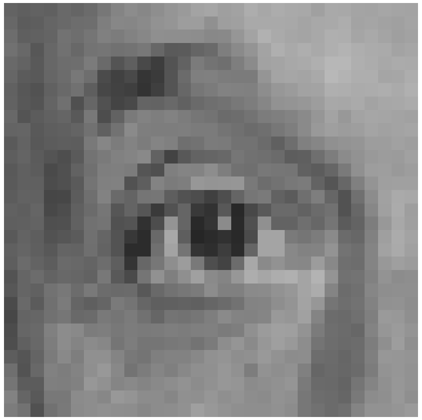


?

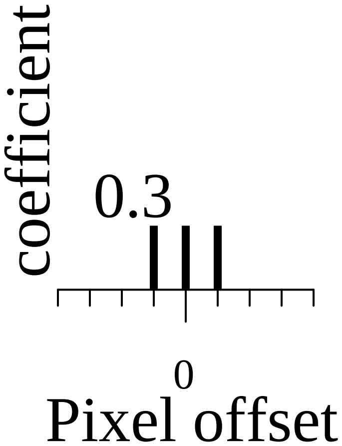
shift



Linear filtering

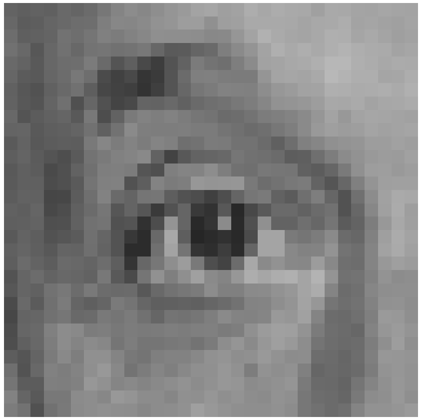


original

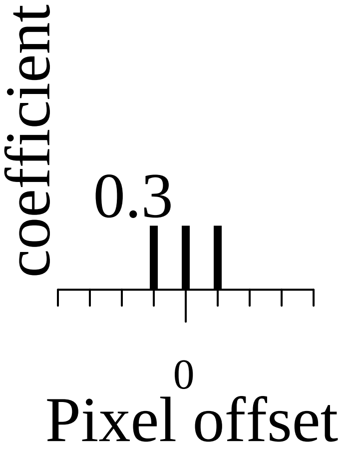


?

Blurring

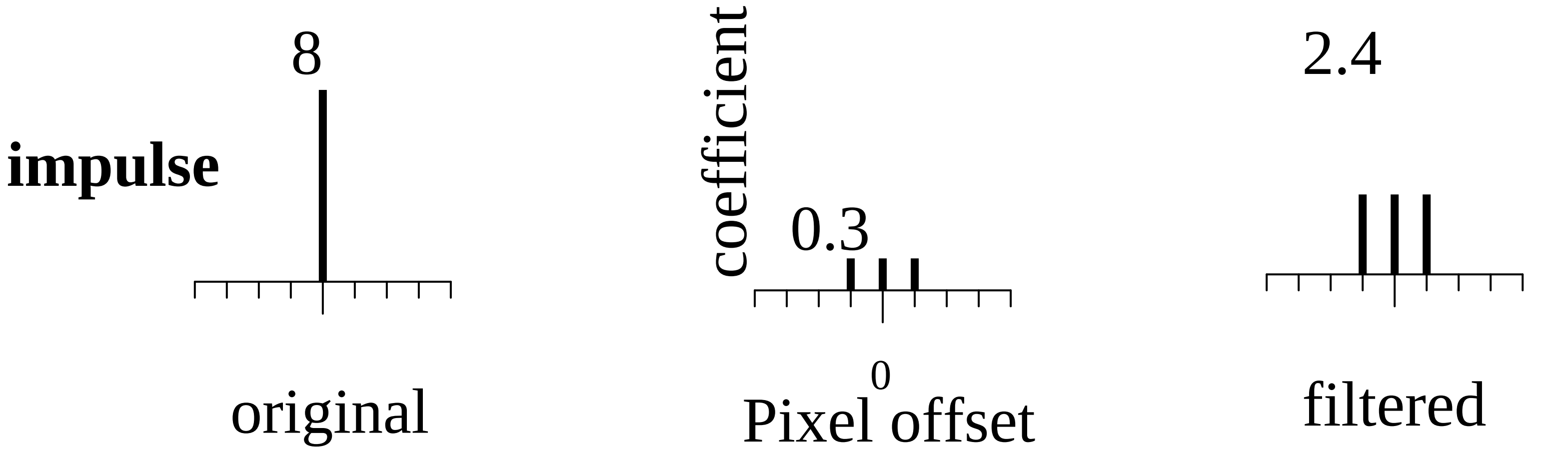


original

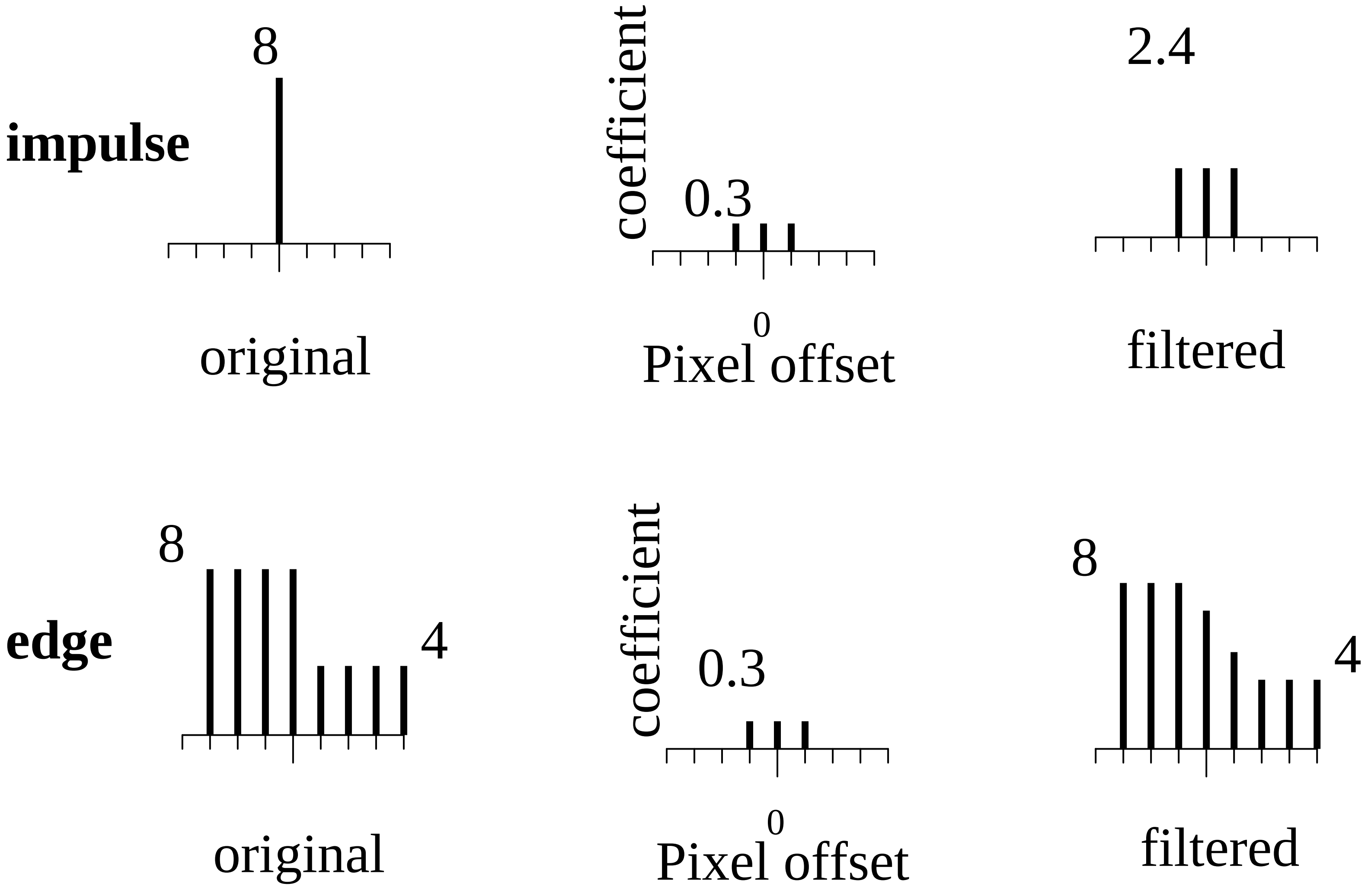


Blurred (filter applied in both dimensions).

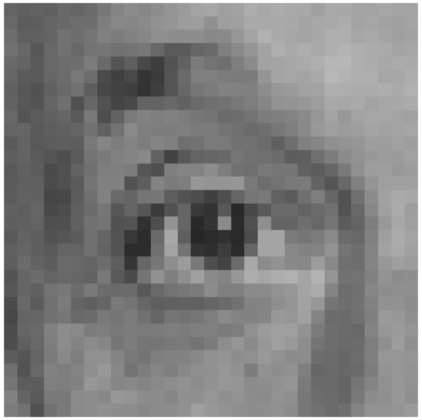
Blur examples



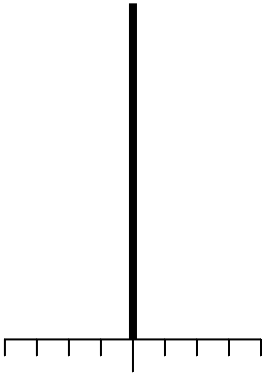
Blur examples



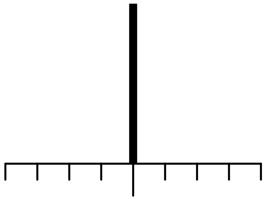
Linear filtering (warm-up slide)



original

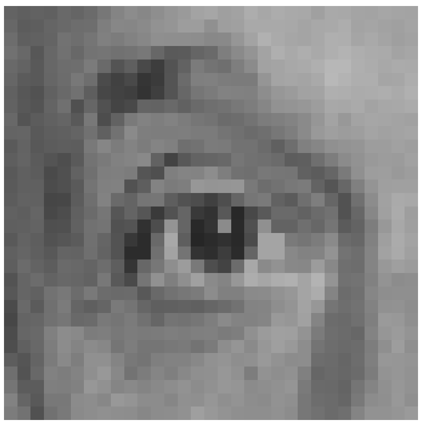


−

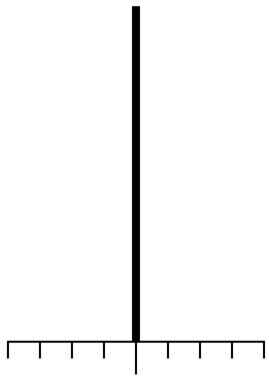


?

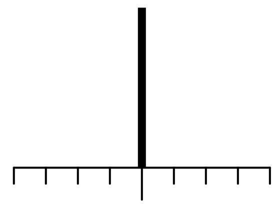
Linear filtering (no change)

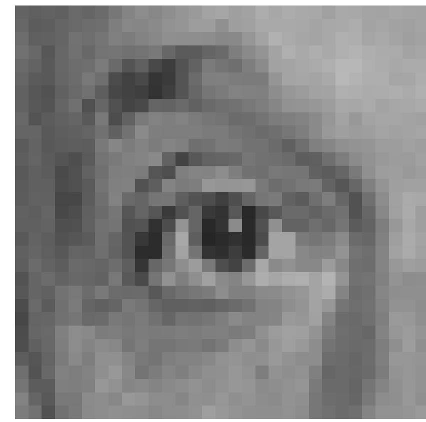


original



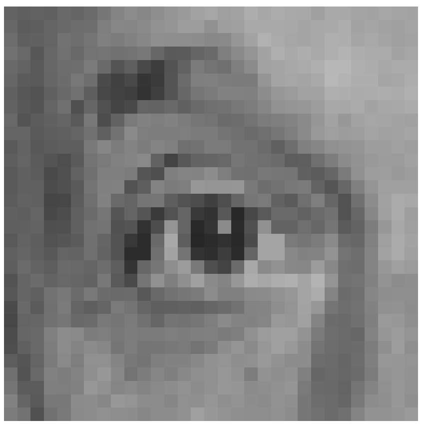
−



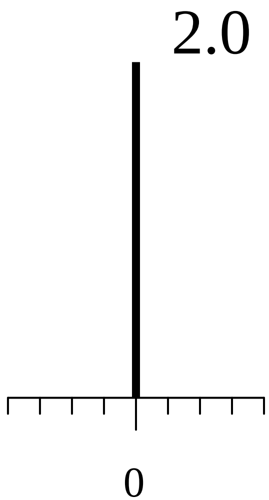


Filtered
(no change)

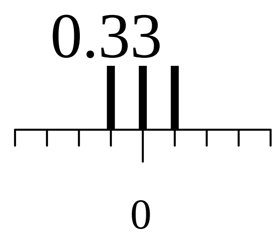
Linear filtering



original

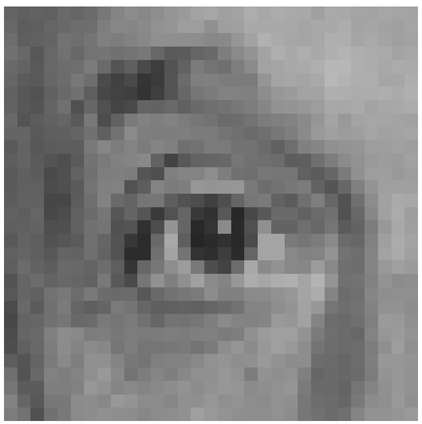


−

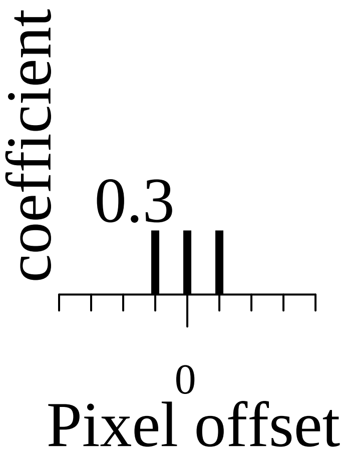


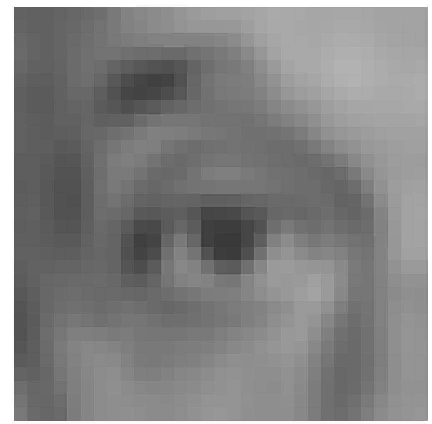
?

(remember blurring)



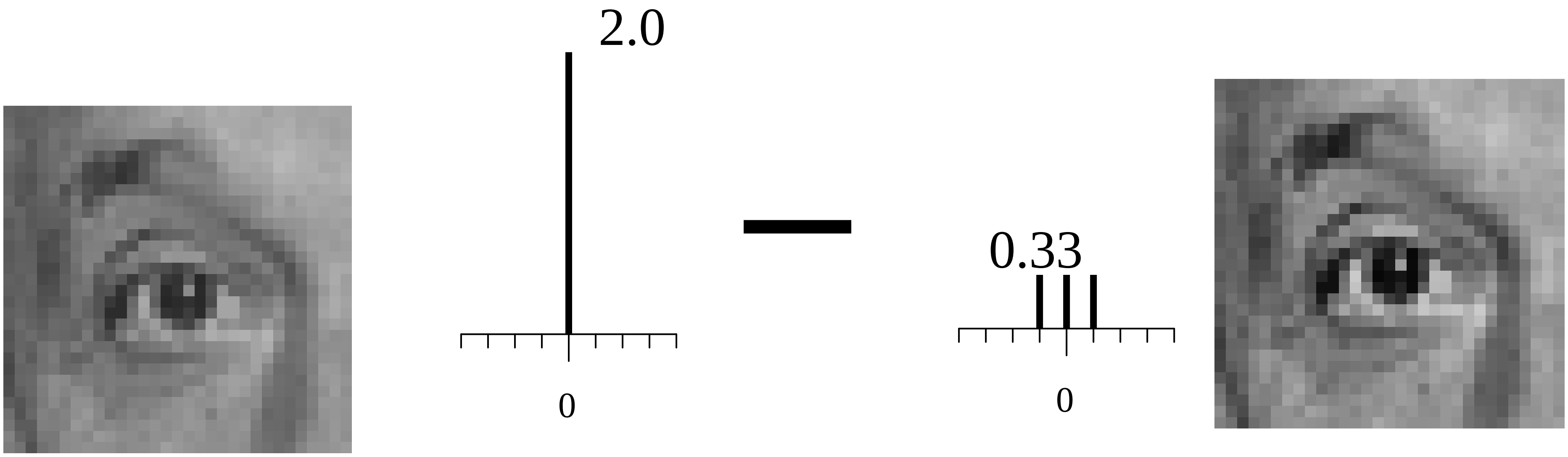
original





Blurred (filter applied in both dimensions).

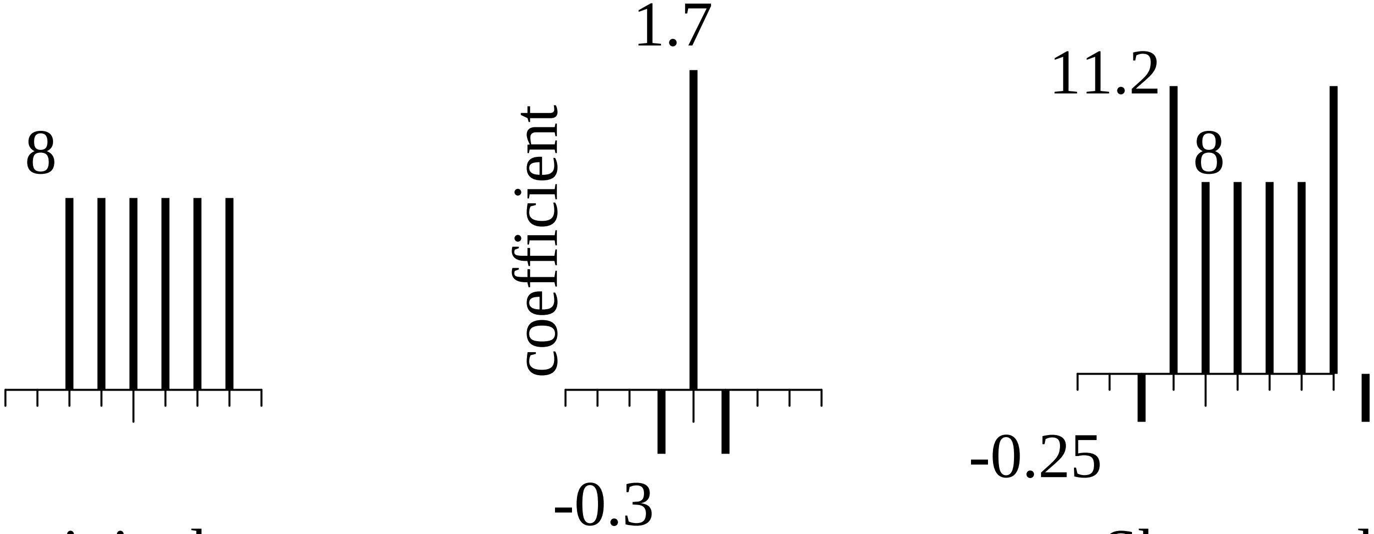
Sharpening



original

Sharpened original

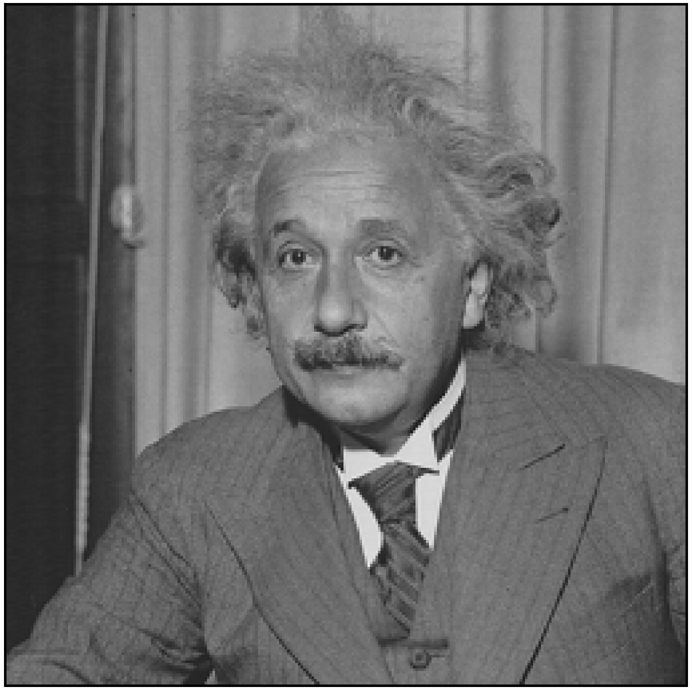
Sharpening example



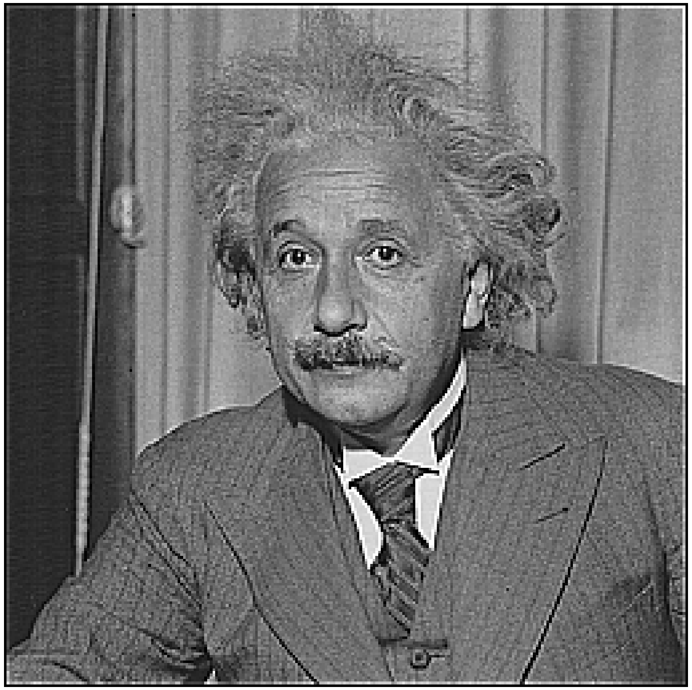
original

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

Sharpening



before



after

Spatial resolution and color

original

R

G

B

Blurring the G component

original

processed

R

G

B

Blurring the R component

original

processed

R

G

B

Blurring the B component

original

processed

R

G

B

From W. E. Glenn, in Digital Images and Human Vision, MIT Press, edited by Watson, 1993

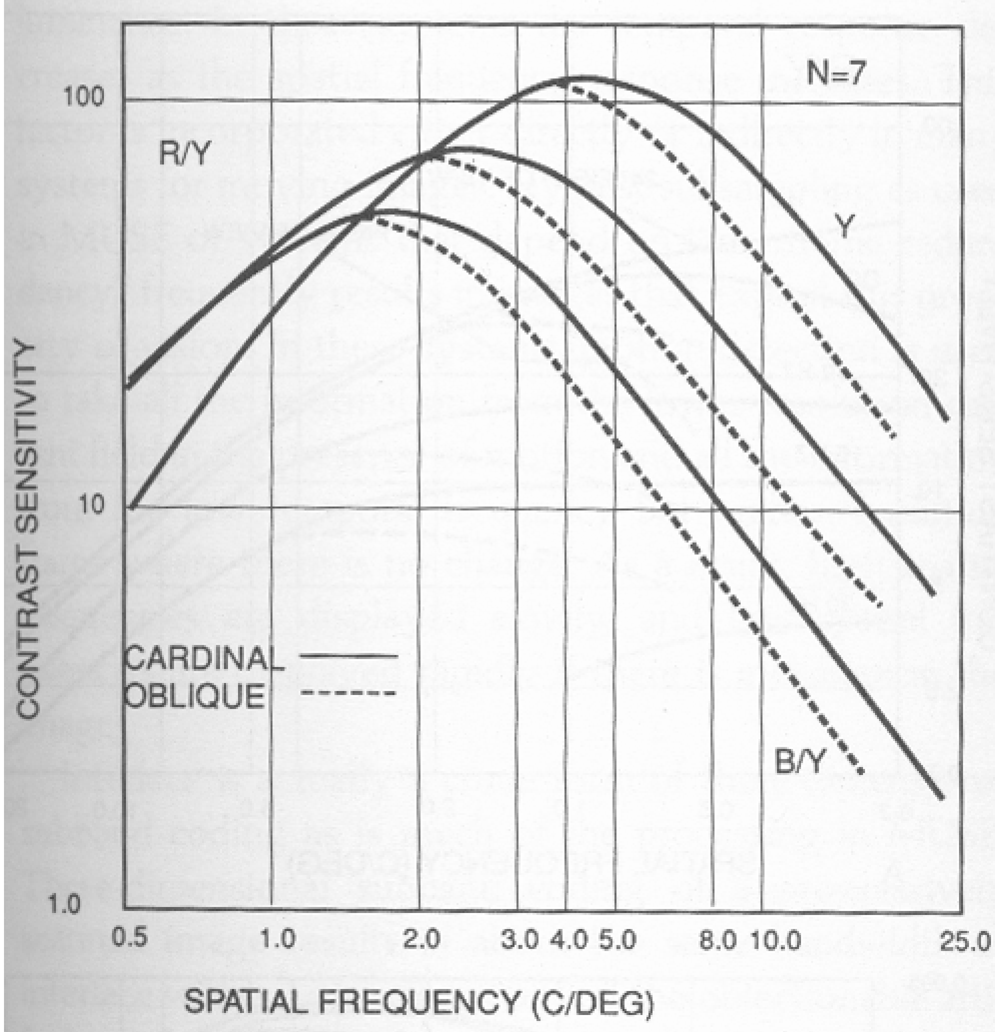


Figure 6.1
Contrast sensitivity threshold functions for static luminance gratings (Y) and isoluminance chromaticity gratings (R/Y,B/Y) averaged over seven observers.

Lab color components

L

a

b

A rotation of the color coordinates into directions that are more perceptually meaningful:
L: luminance,
a: red-green,
b: blue-yellow

Blurring the L Lab component

original

processed

L

a

b

Blurring the a Lab component

original

processed

L

a

b

Blurring the b Lab component

original

processed

L

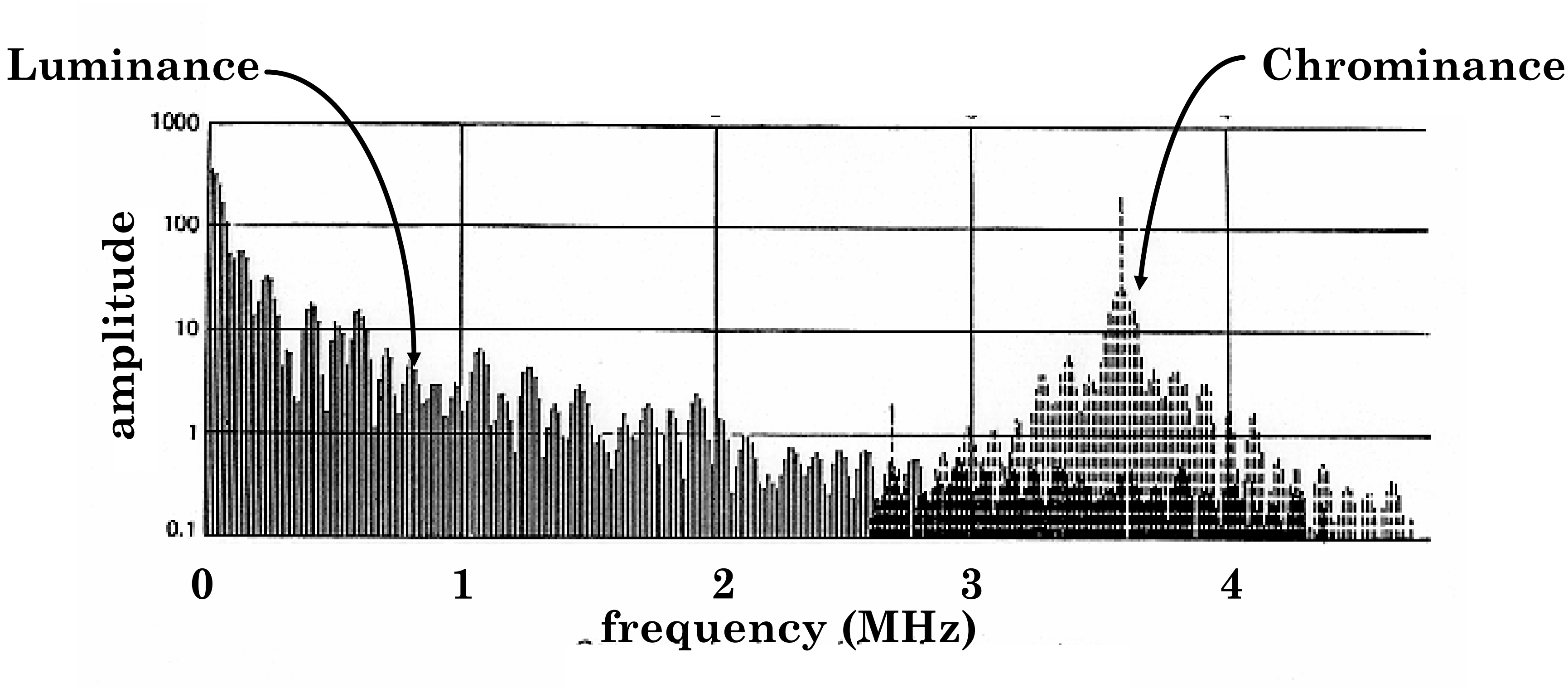
a

b

Application to image compression

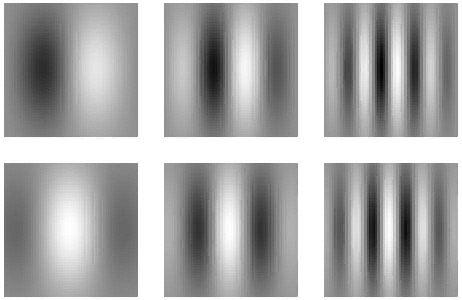
- (compression is about hiding differences from the true image where you can't see them).

Bandwidth (transmission resources) for the components of the television signal



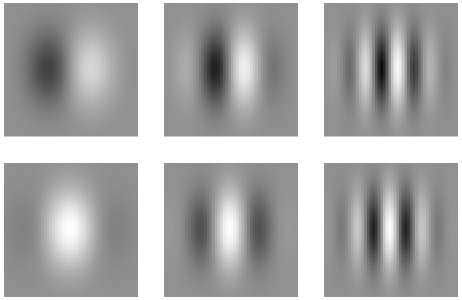
Understanding image perception allowed NTSC to add color to the black and white television signal (with some, but limited, incompatibility artifacts).

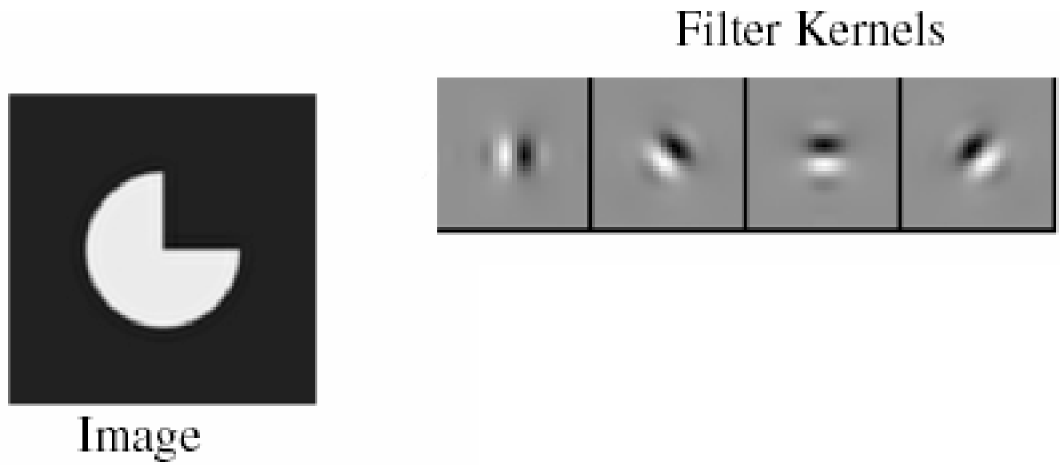
Oriented filters



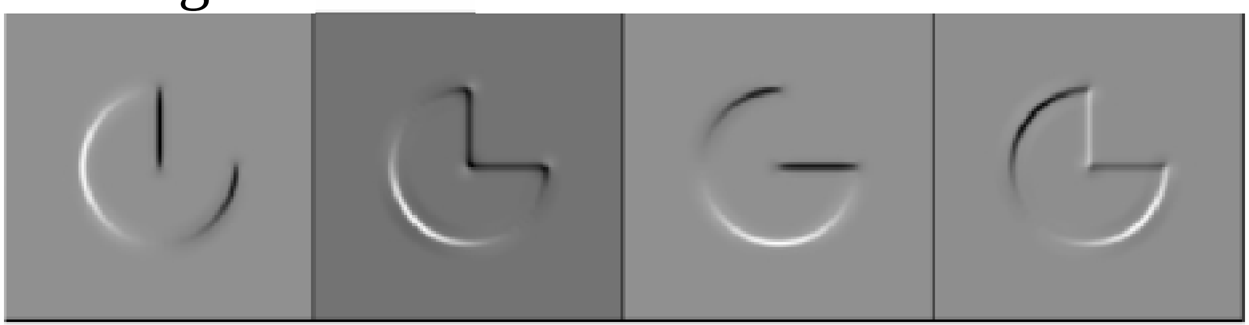
Gabor filters at different scales and spatial frequencies

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





Filtered images



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

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

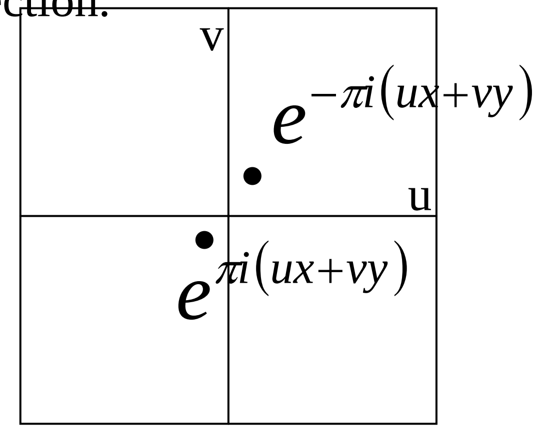
Forward transform

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

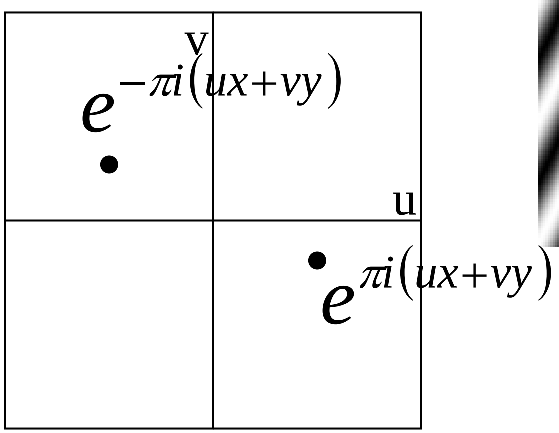
Inverse transform

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

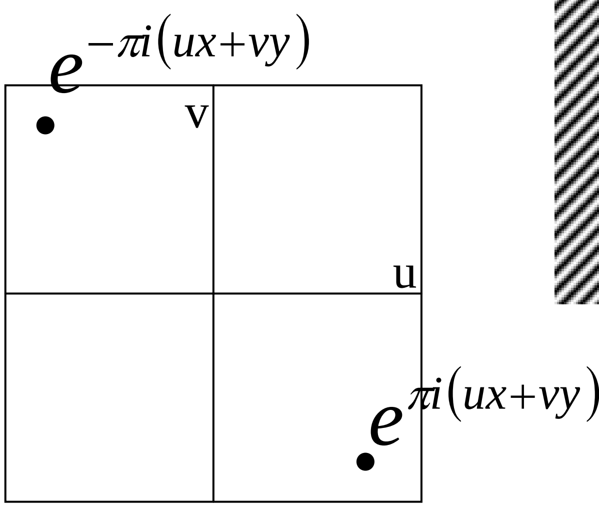
To get some sense of what basis elements look like, we plot a basis element --- or rather, its real part --- as a function of x,y for some fixed u, v. We get a function that is constant when (ux+vy) is constant. The magnitude of the vector (u, v) gives a frequency, and its direction gives an orientation.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

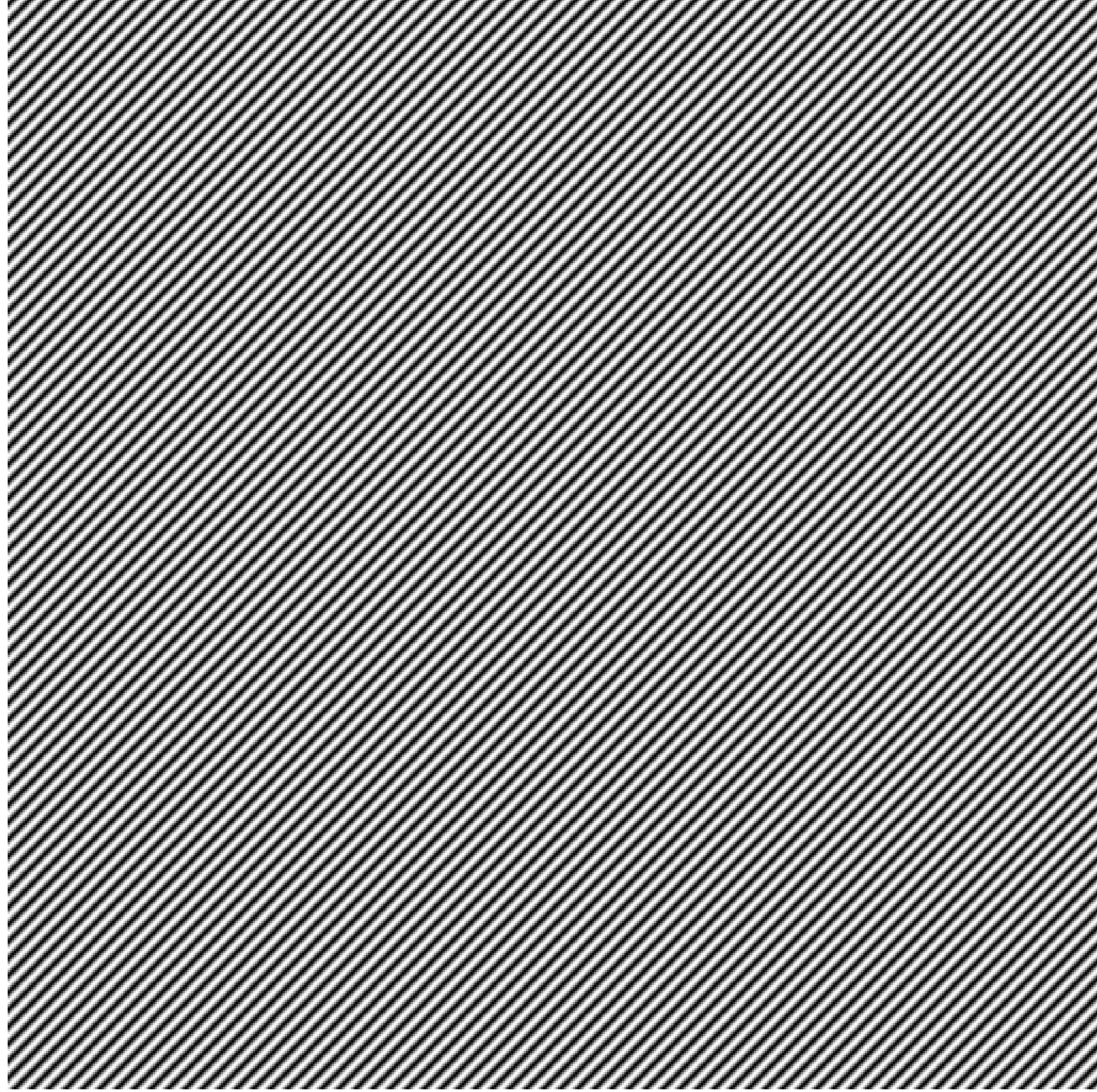


Here u and v are larger than in the previous slide.



And larger still...

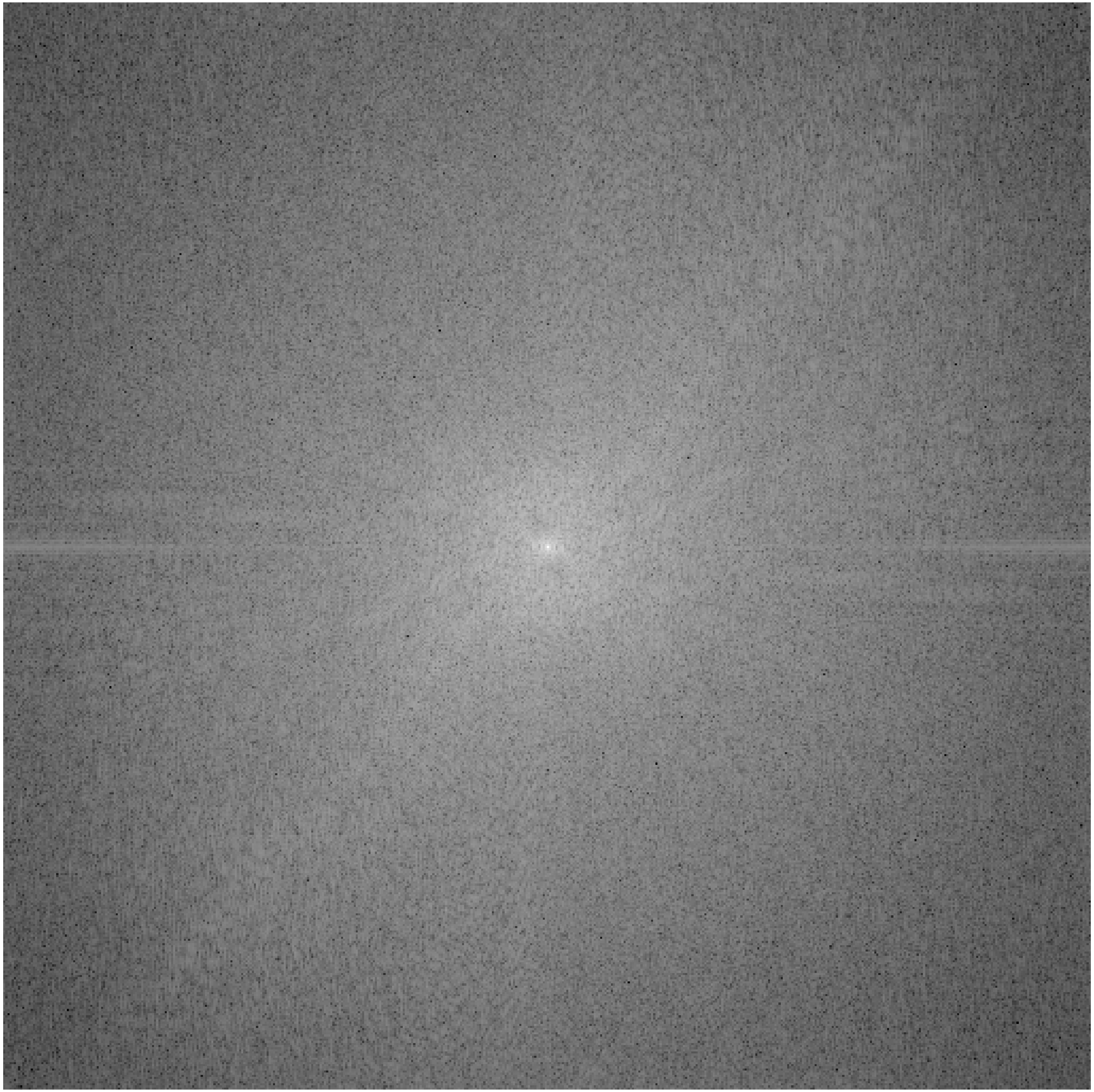




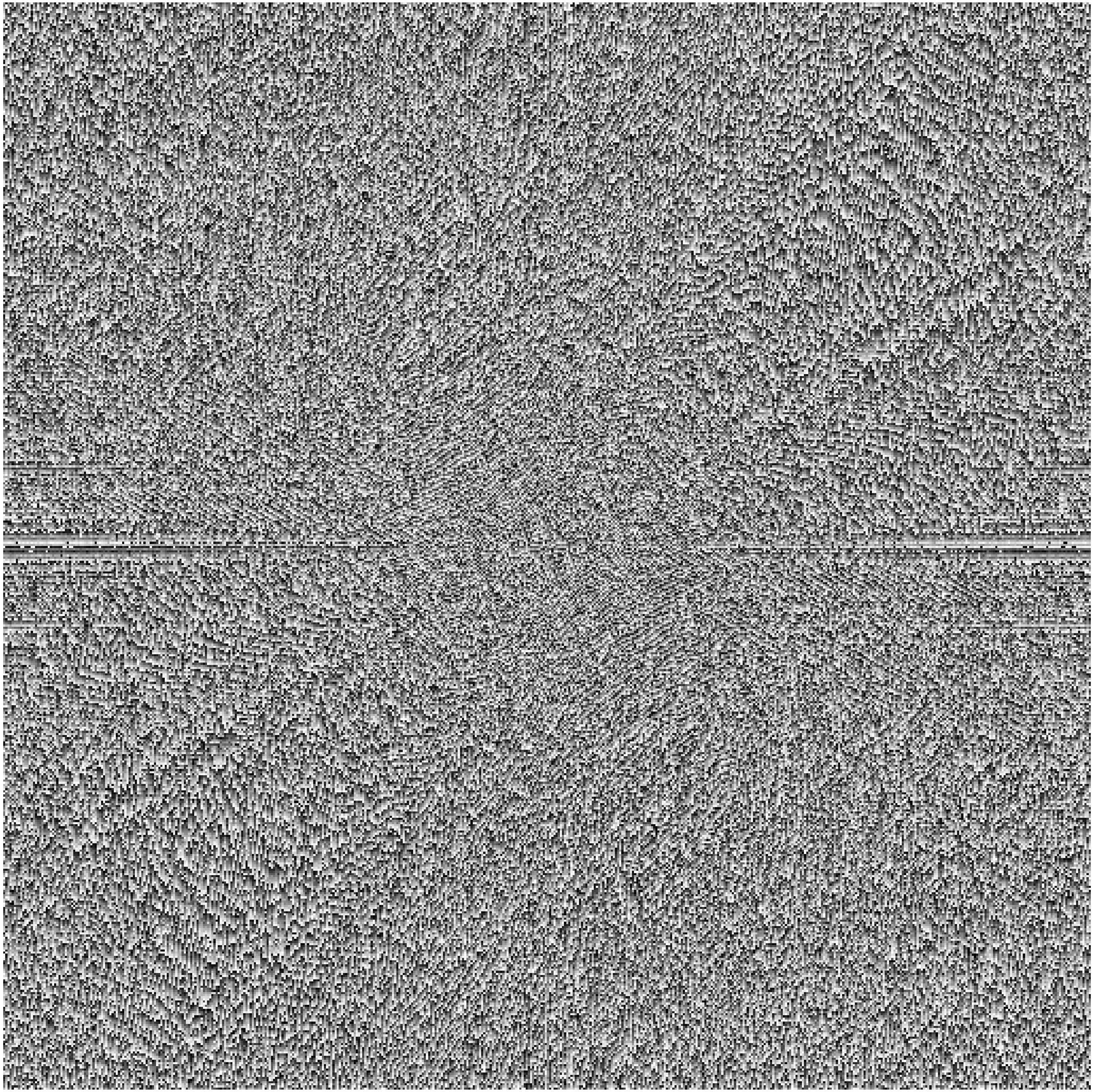
Phase and Magnitude

- 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?

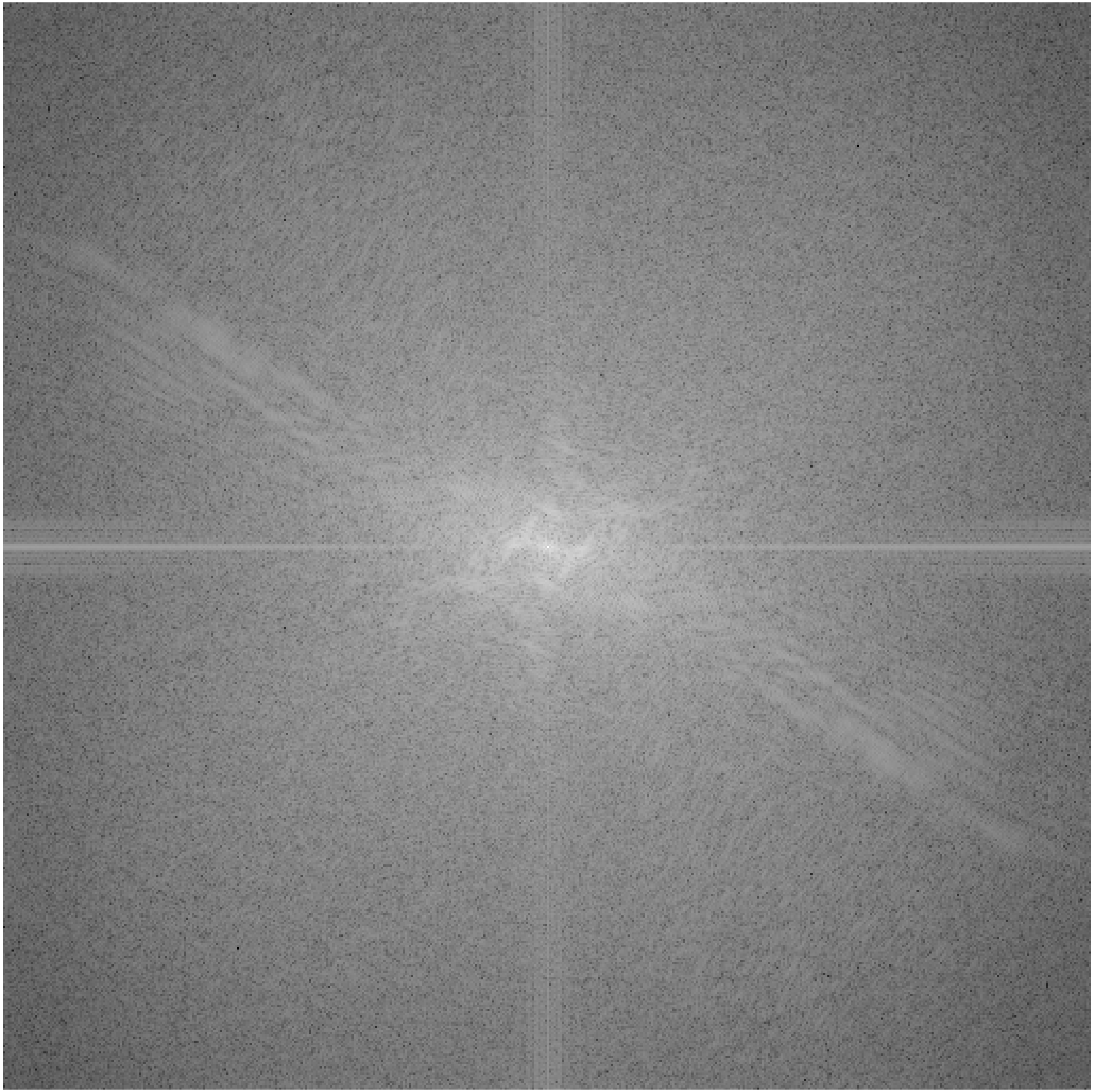
This is the magnitude transform of the cheetah pic



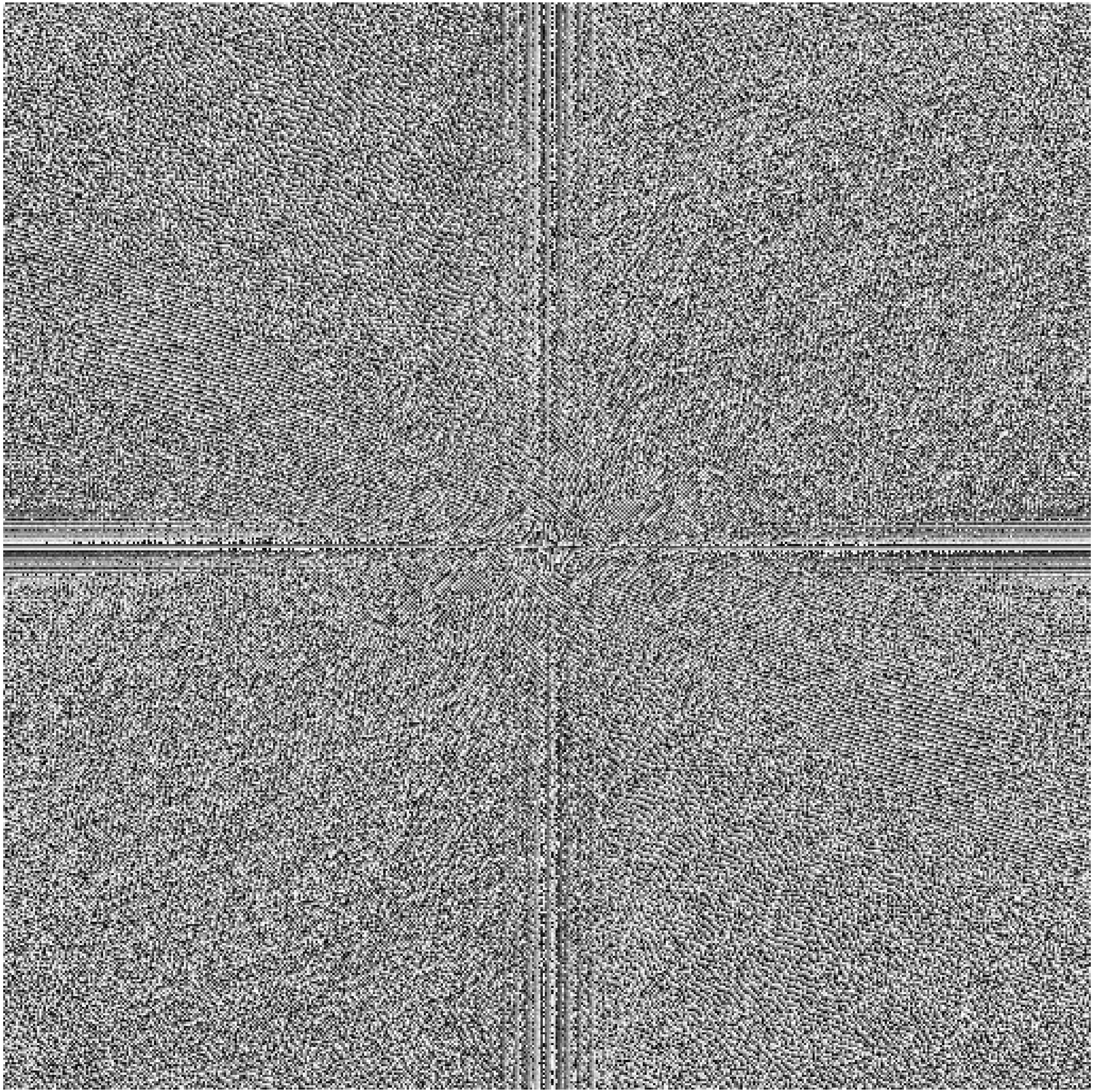
This is the phase transform of the cheetah pic



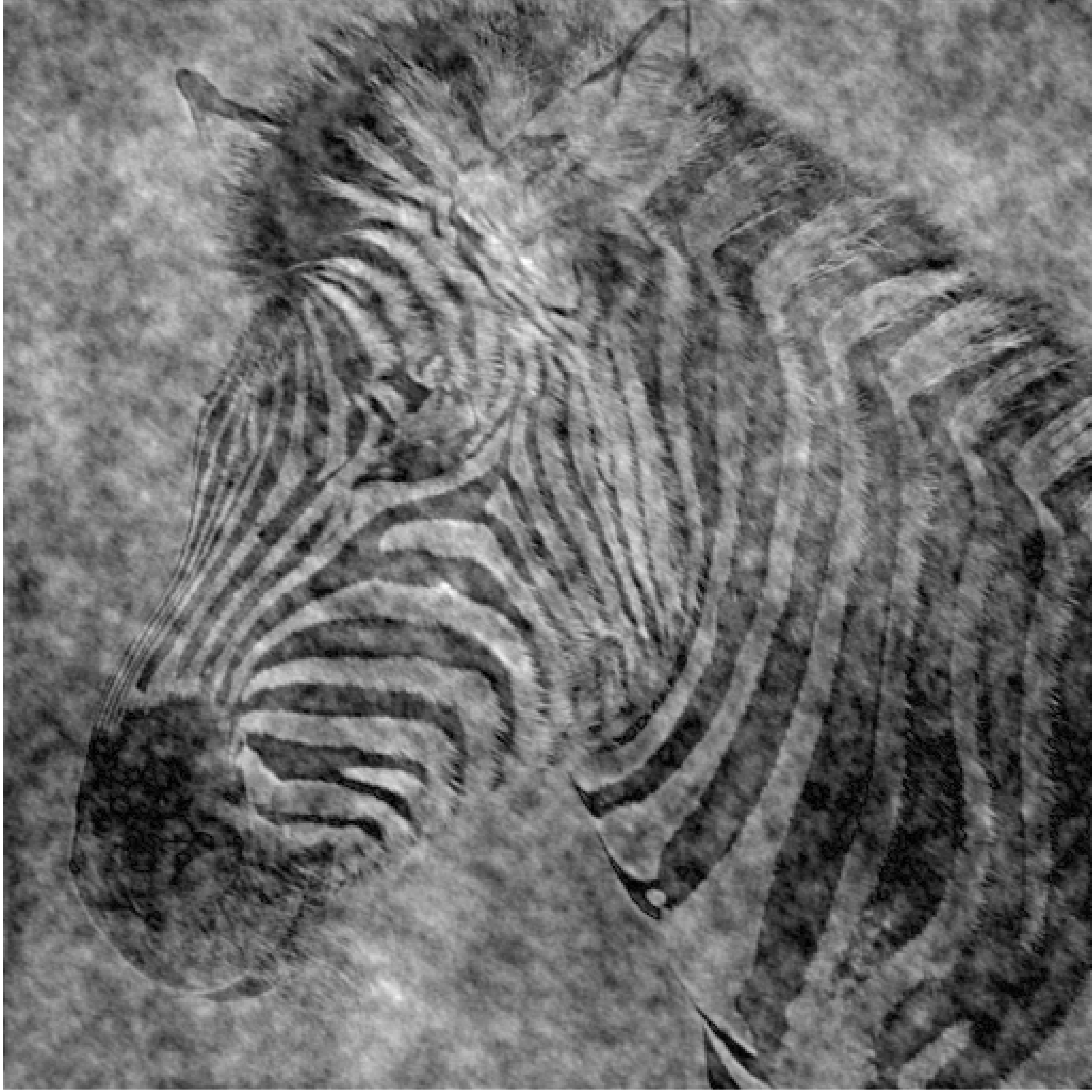
This is the magnitude transform of the zebra pic



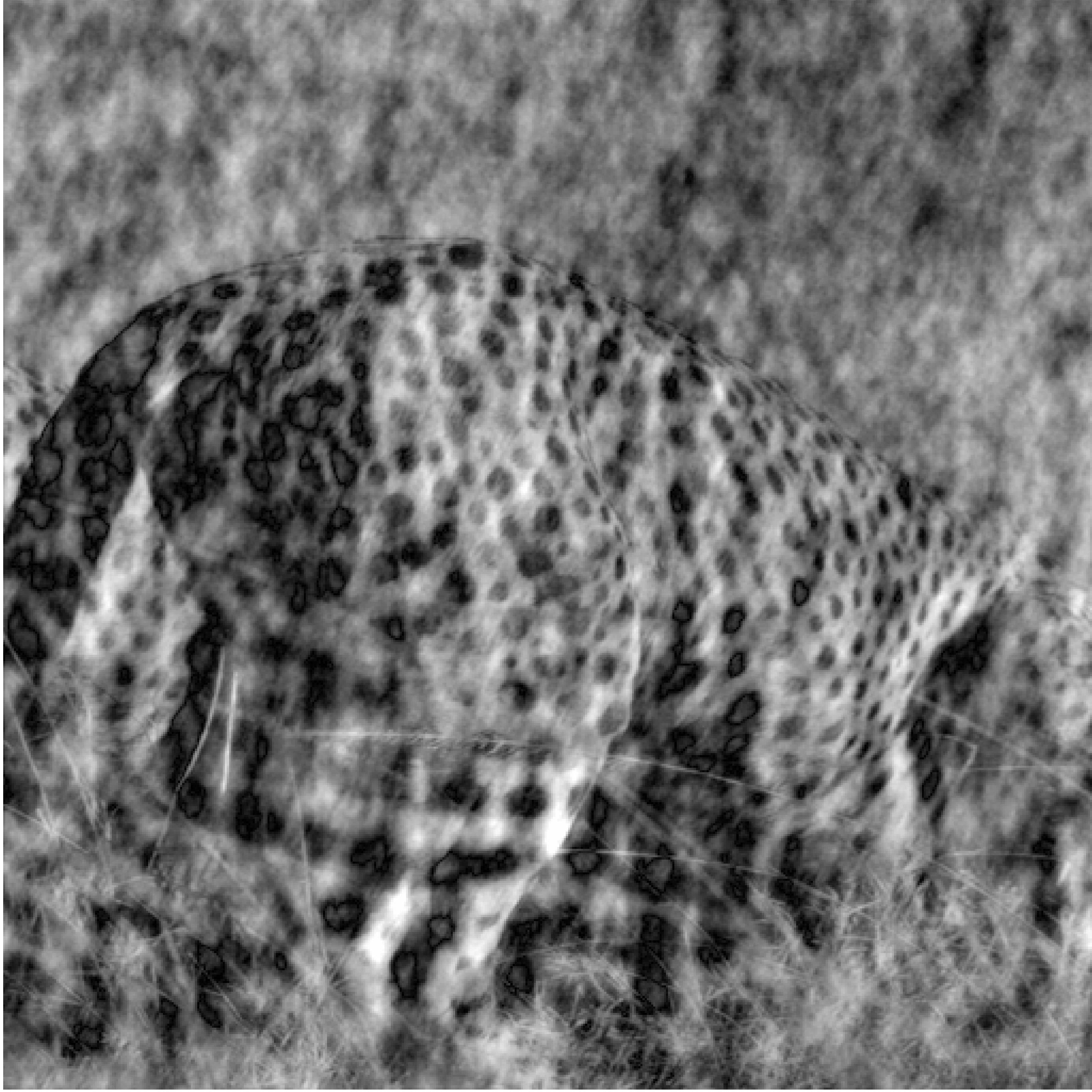
This is the phase transform of the zebra pic



Reconstruction with zebra phase, cheetah magnitude



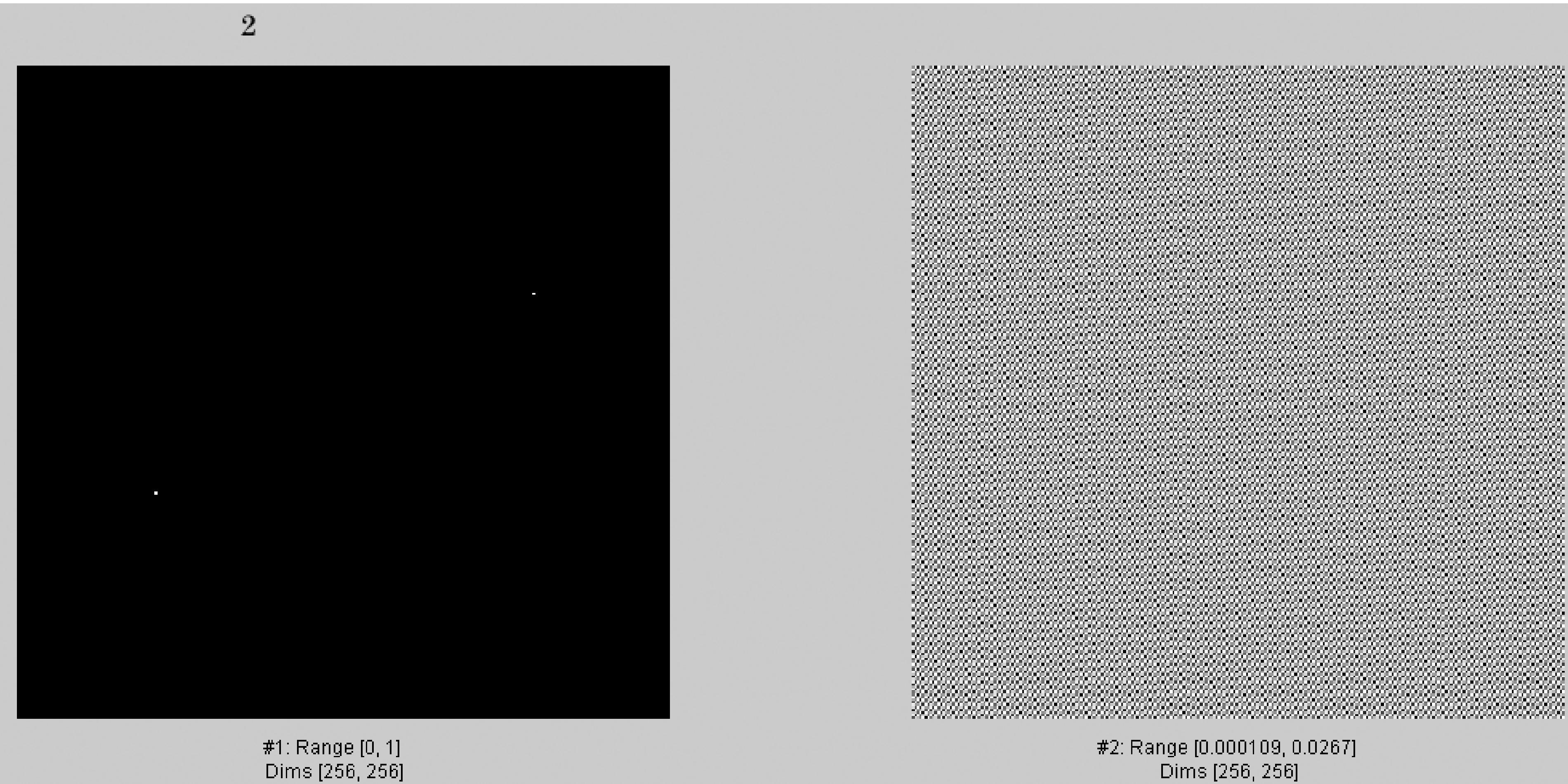
Reconstruction with cheetah phase, zebra magnitude



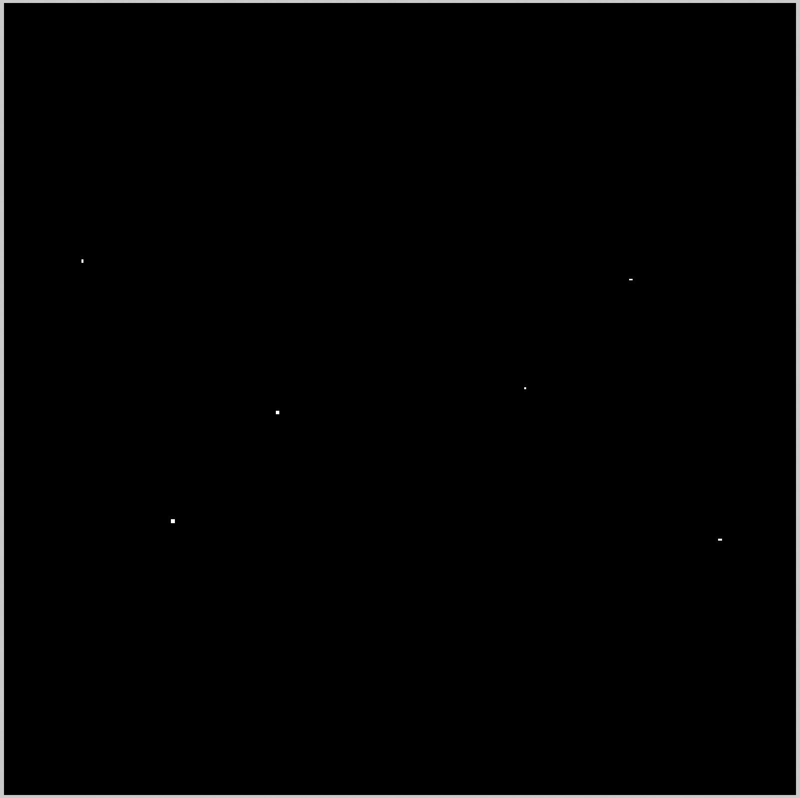
Example image synthesis with Fourier basis.

- Following are 16 images showing the reconstruction of an image from a random selection of Fourier basis functions.
- Note, the selection of basis functions to include was not made according to basis magnitude. Doing that would have given us an approximate version of the image much sooner.

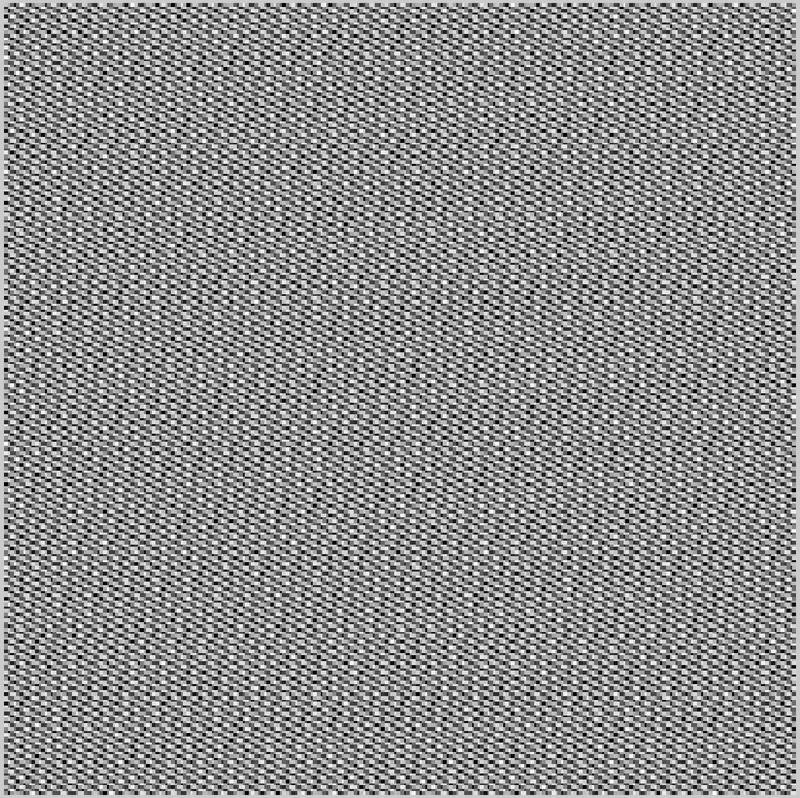
2



6

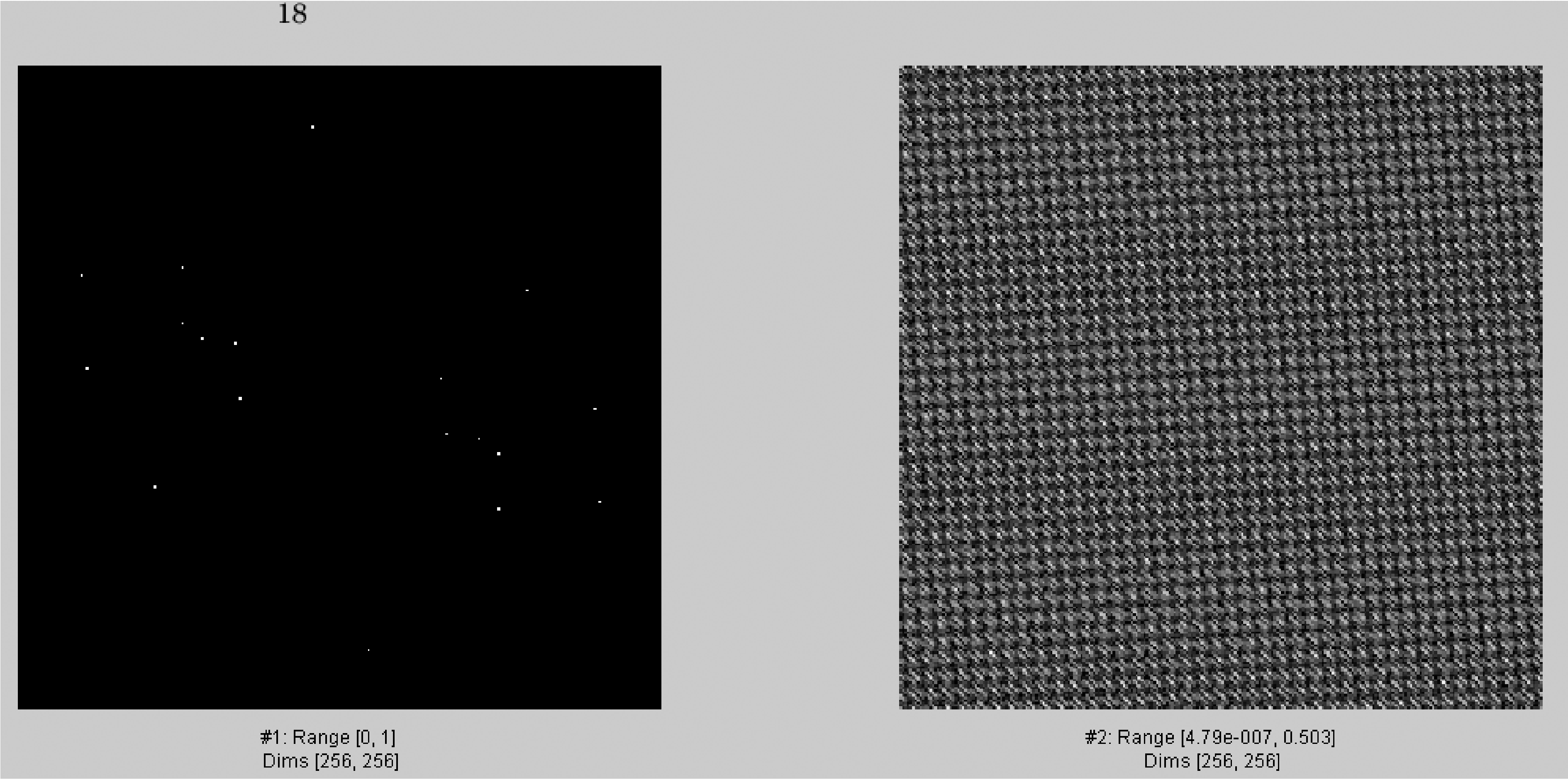


#1: Range [0, 1]
Dims [256, 256]

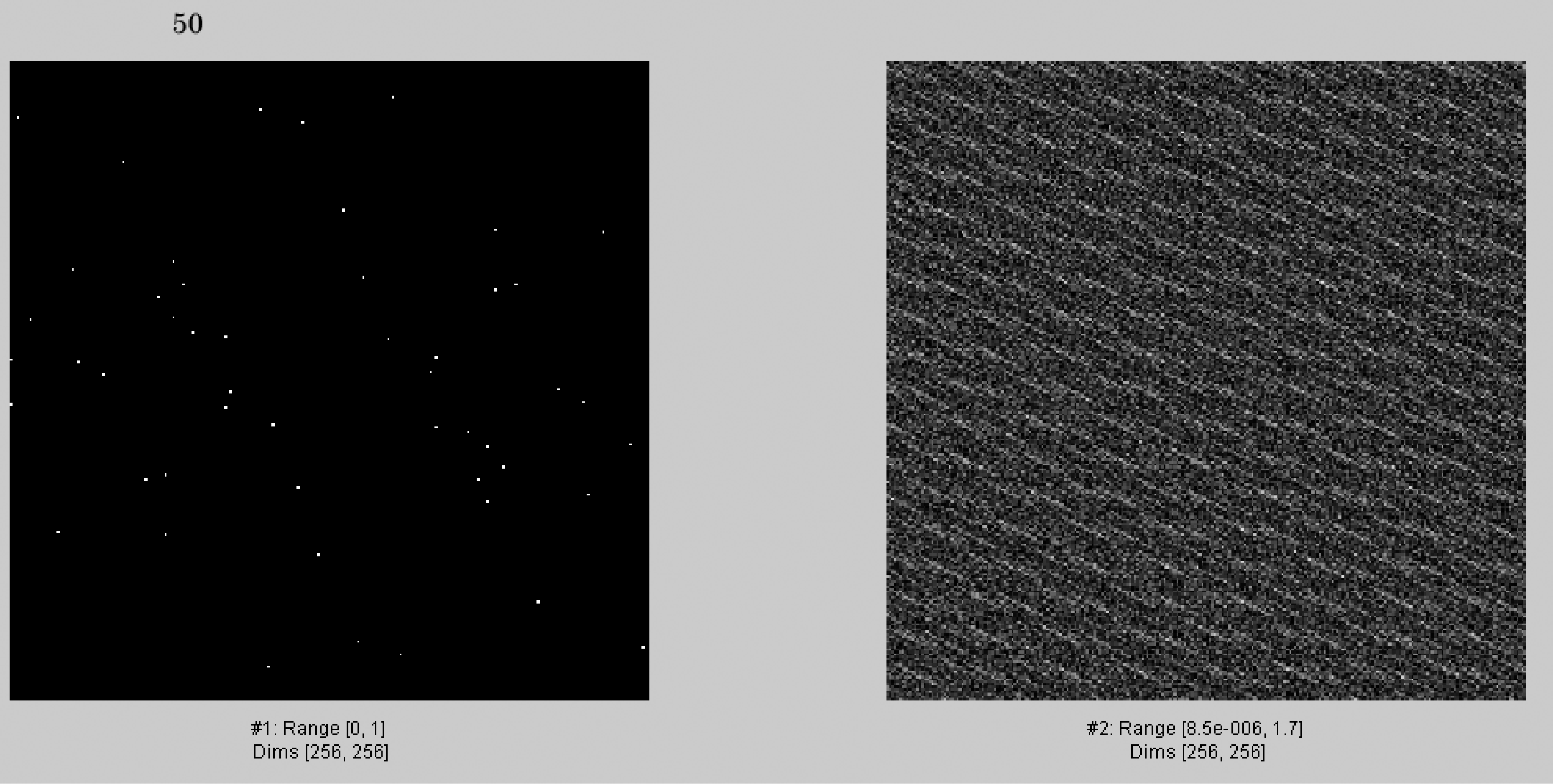


#2: Range [1.89e-007, 0.226]
Dims [256, 256]

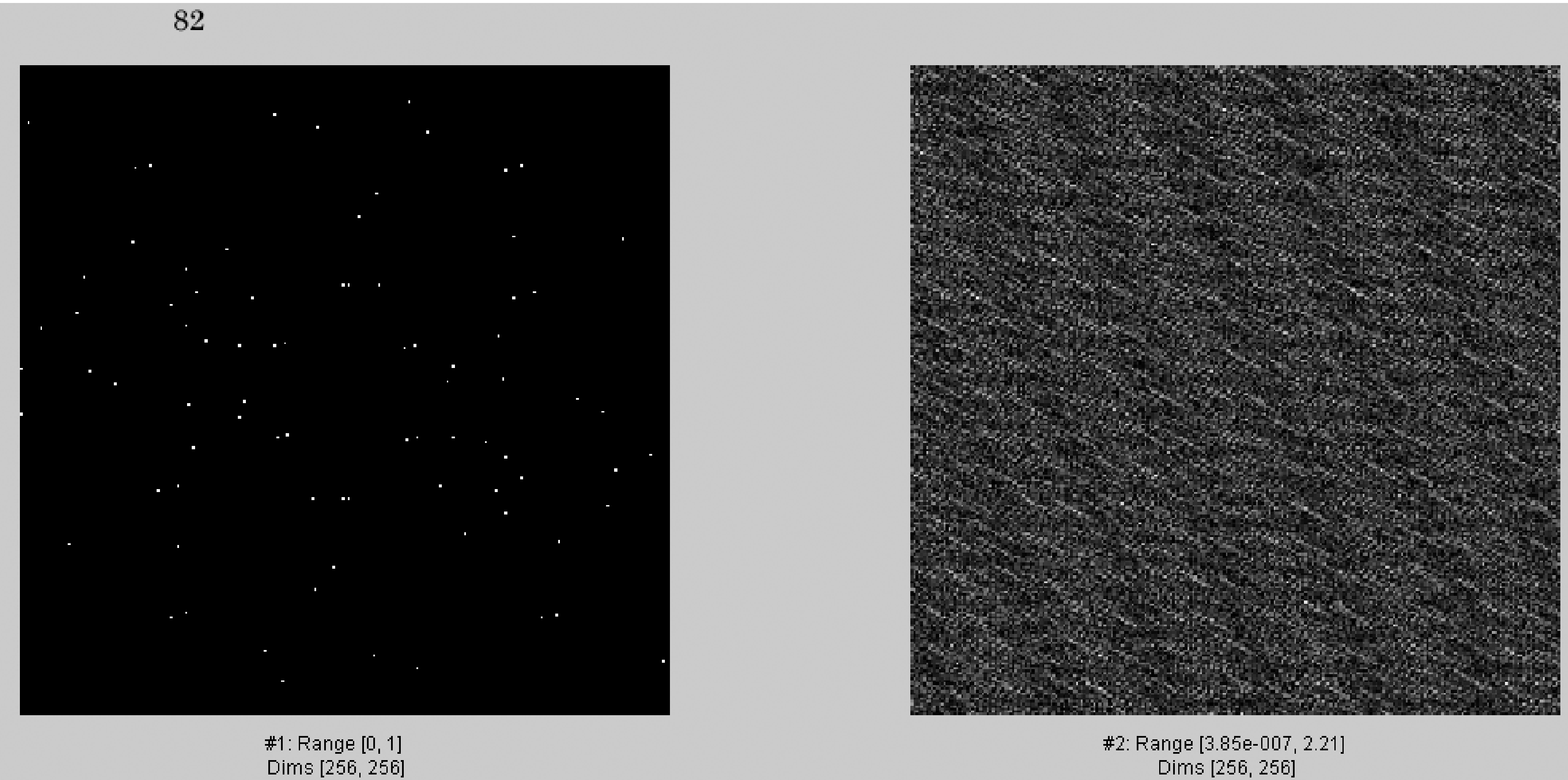
18



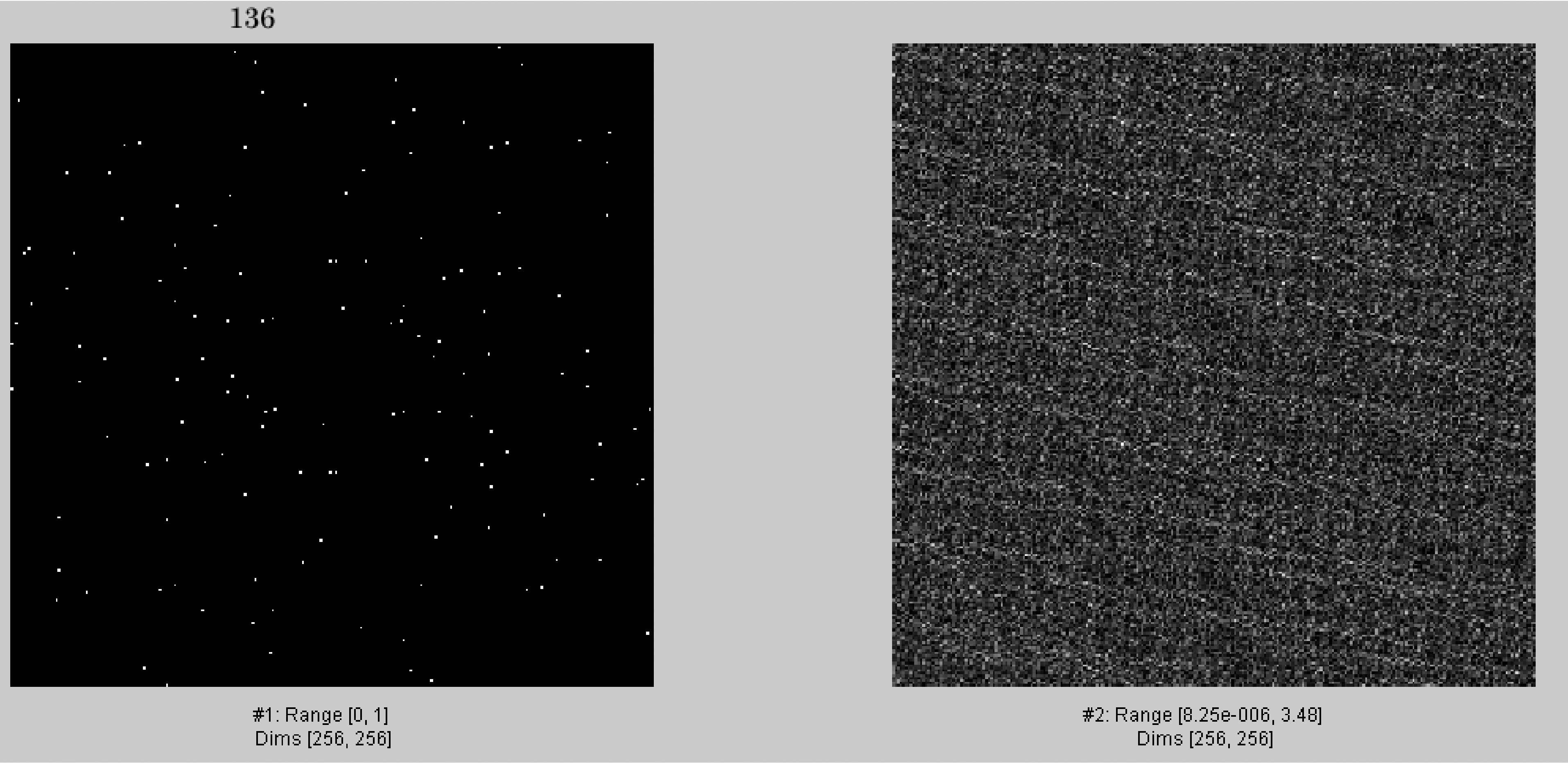
50



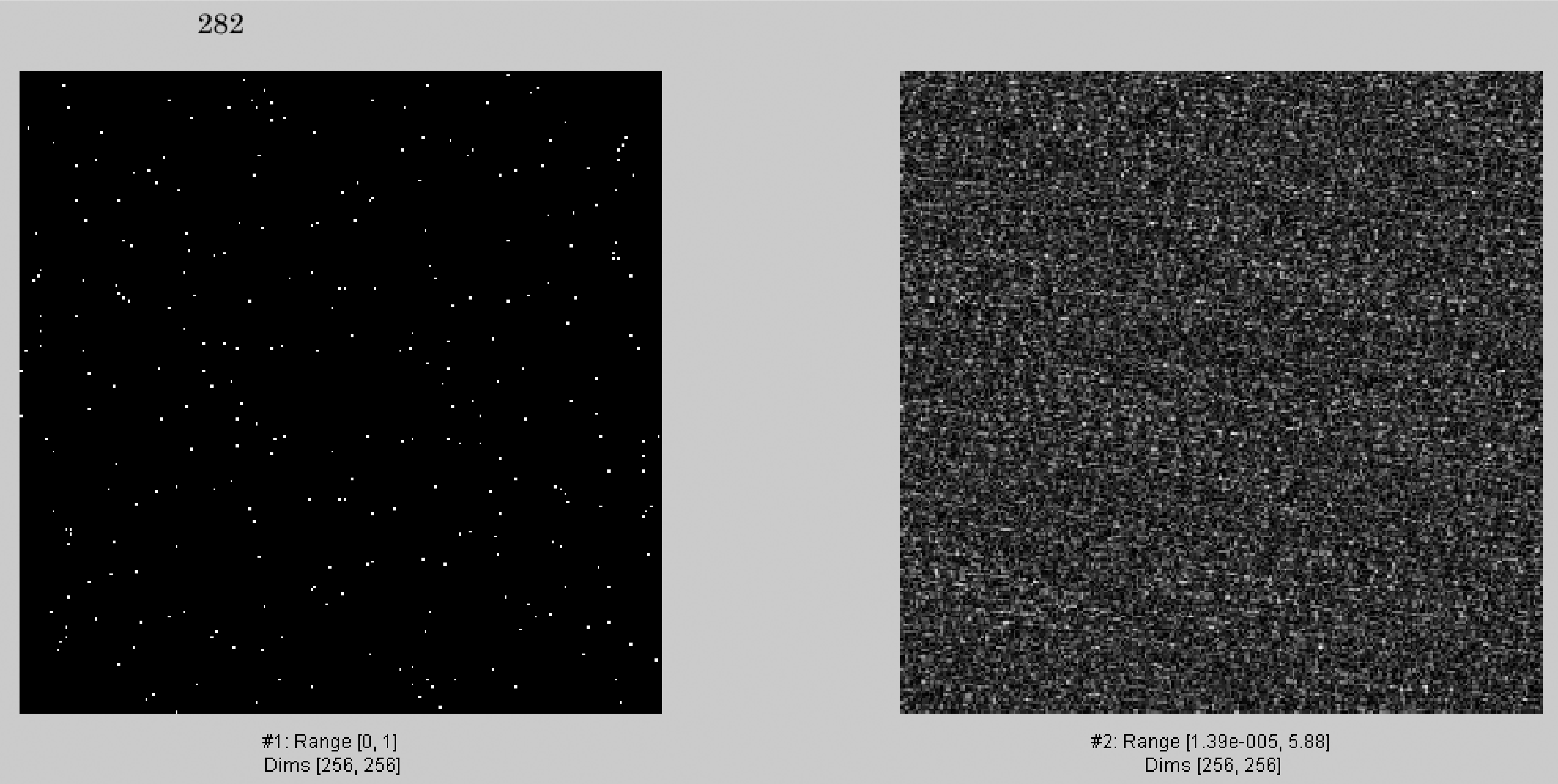
82



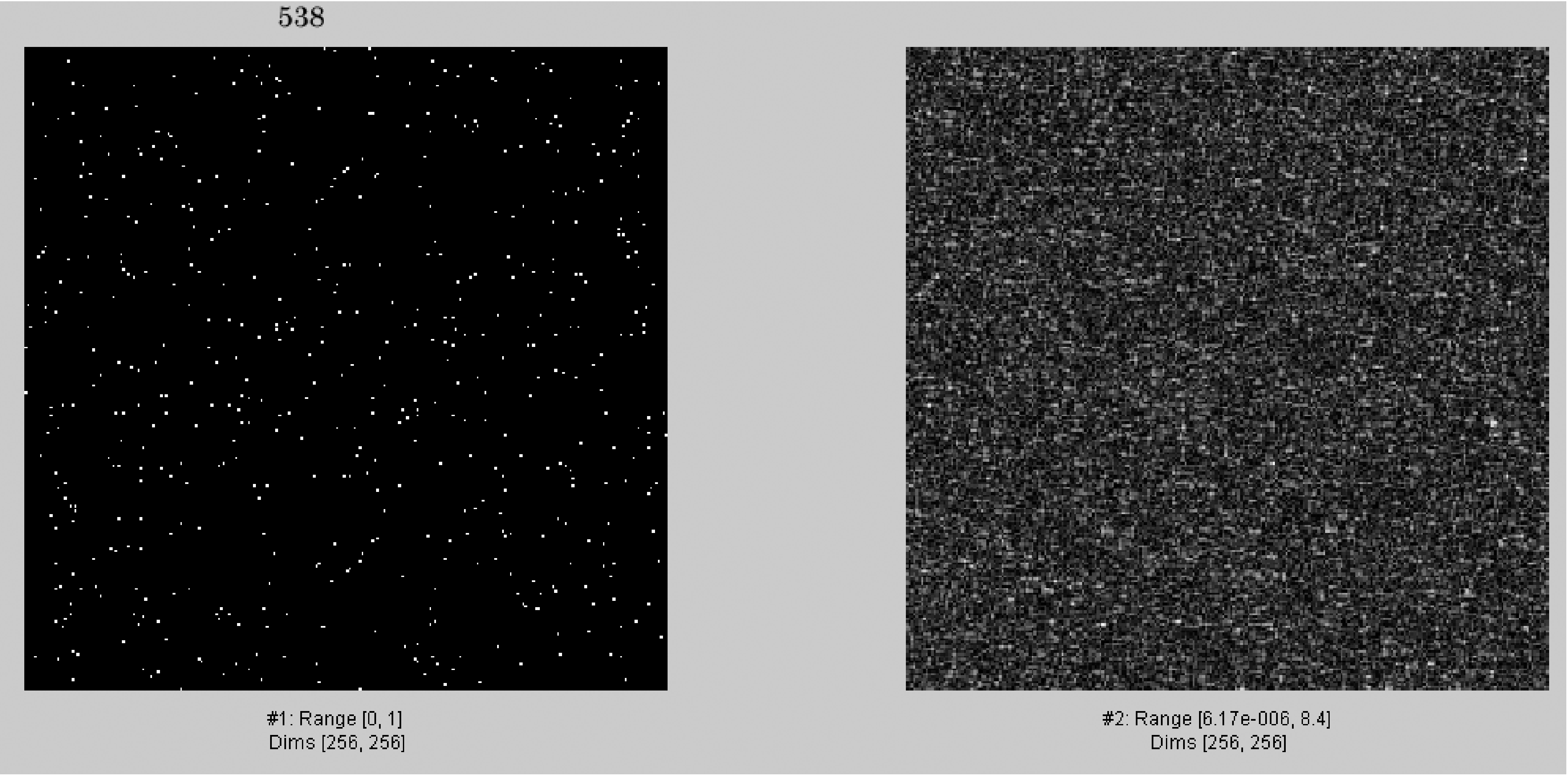
136



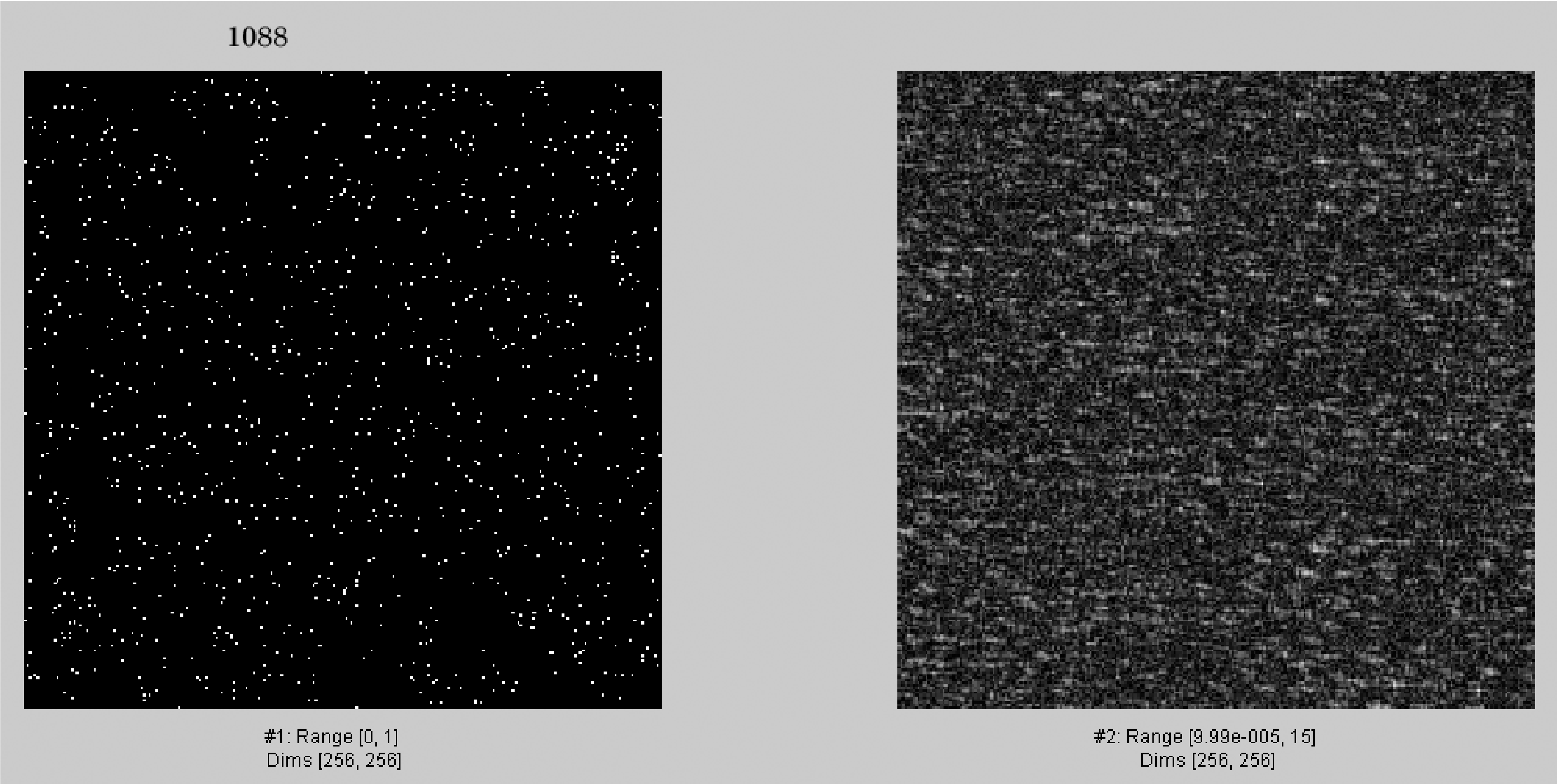
282



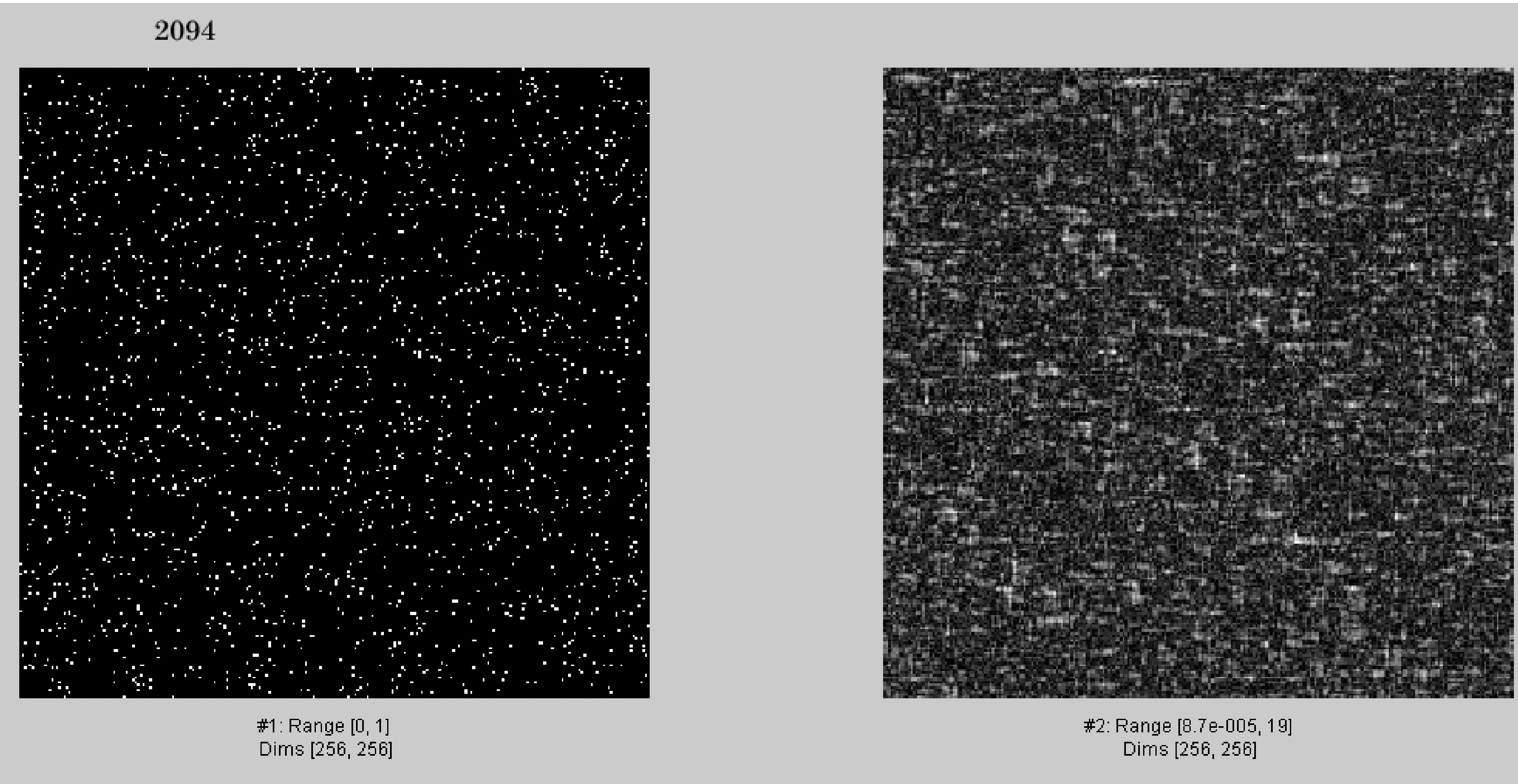
538



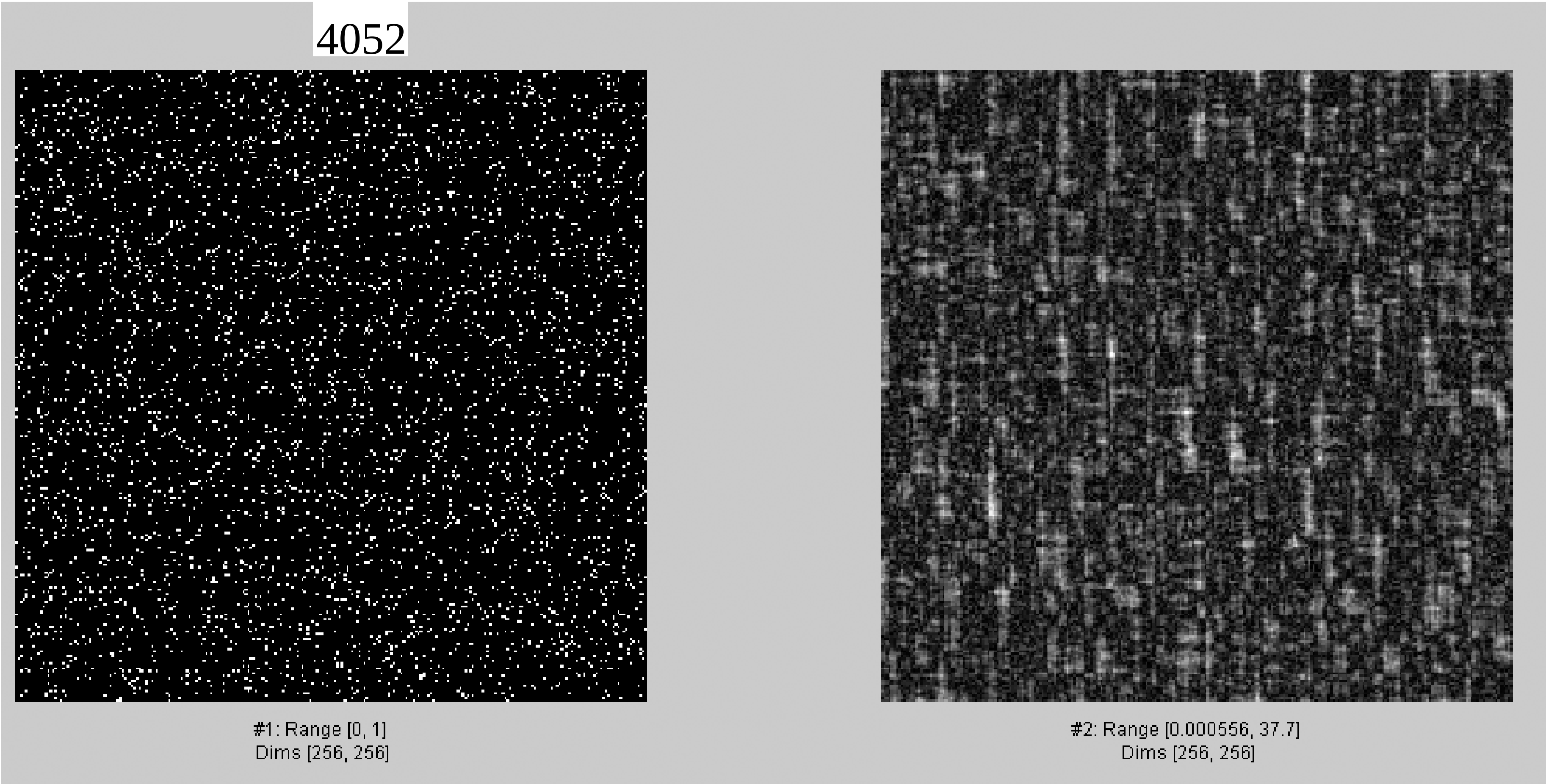
1088



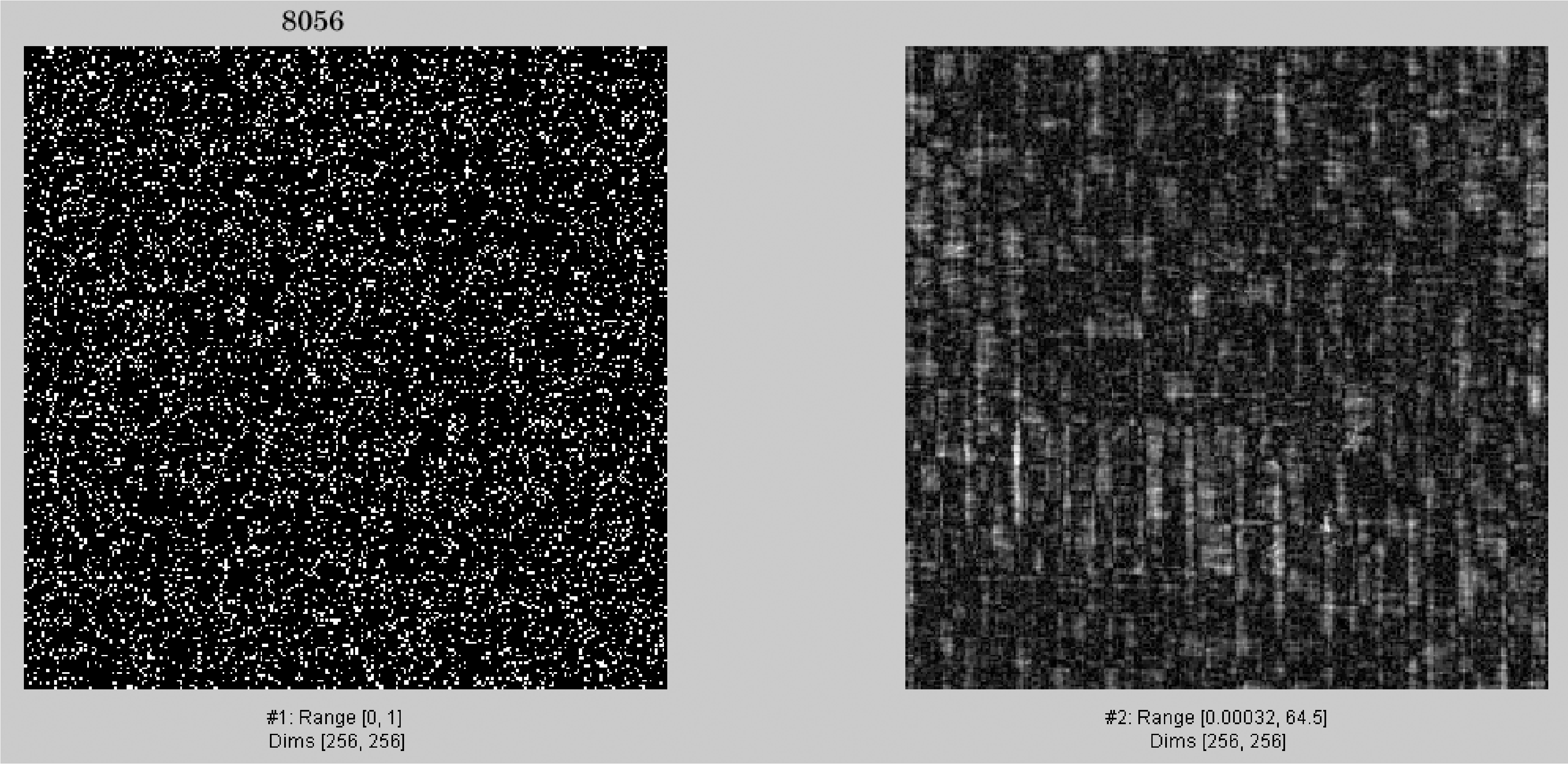
2094



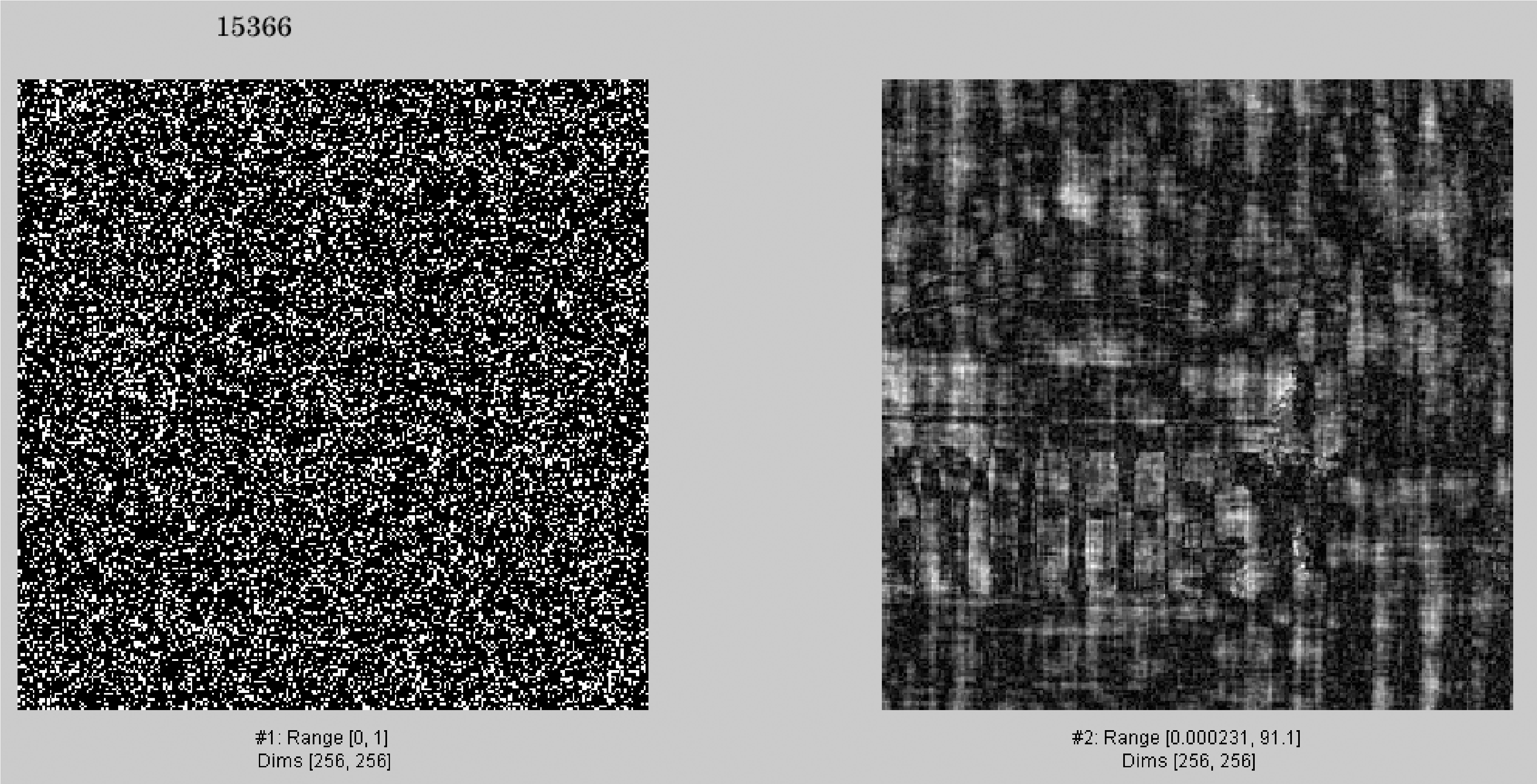
4052.



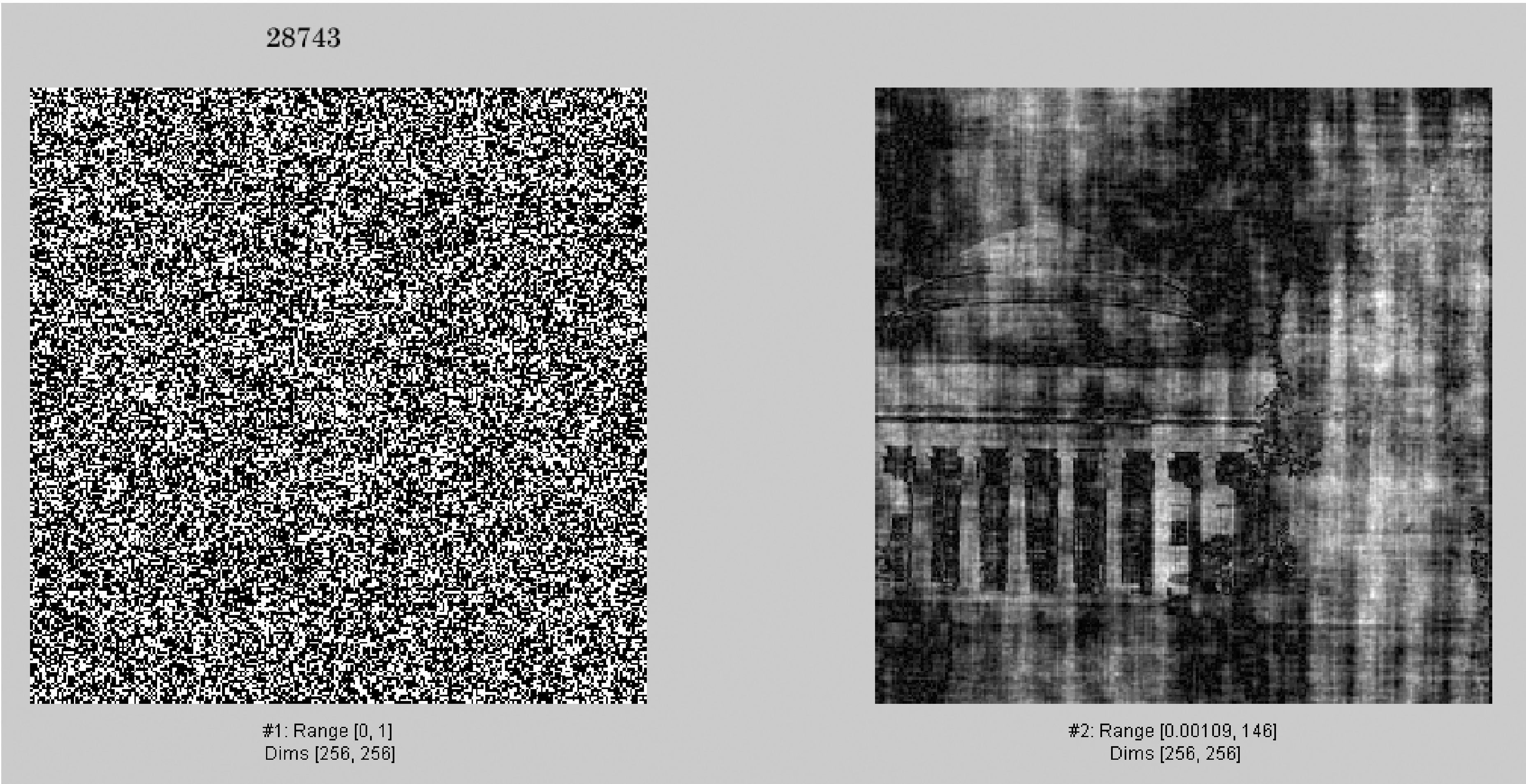
8056.



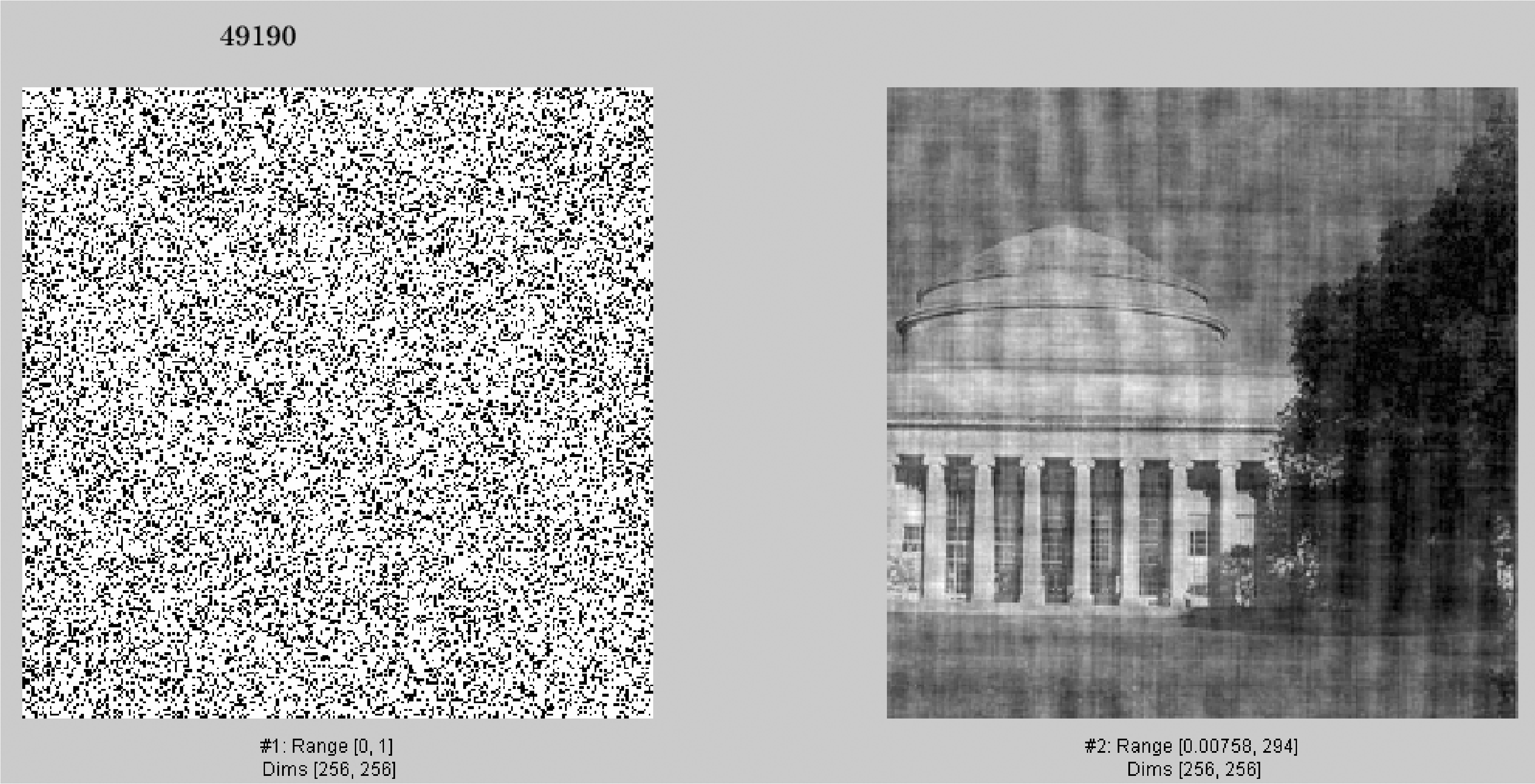
15366



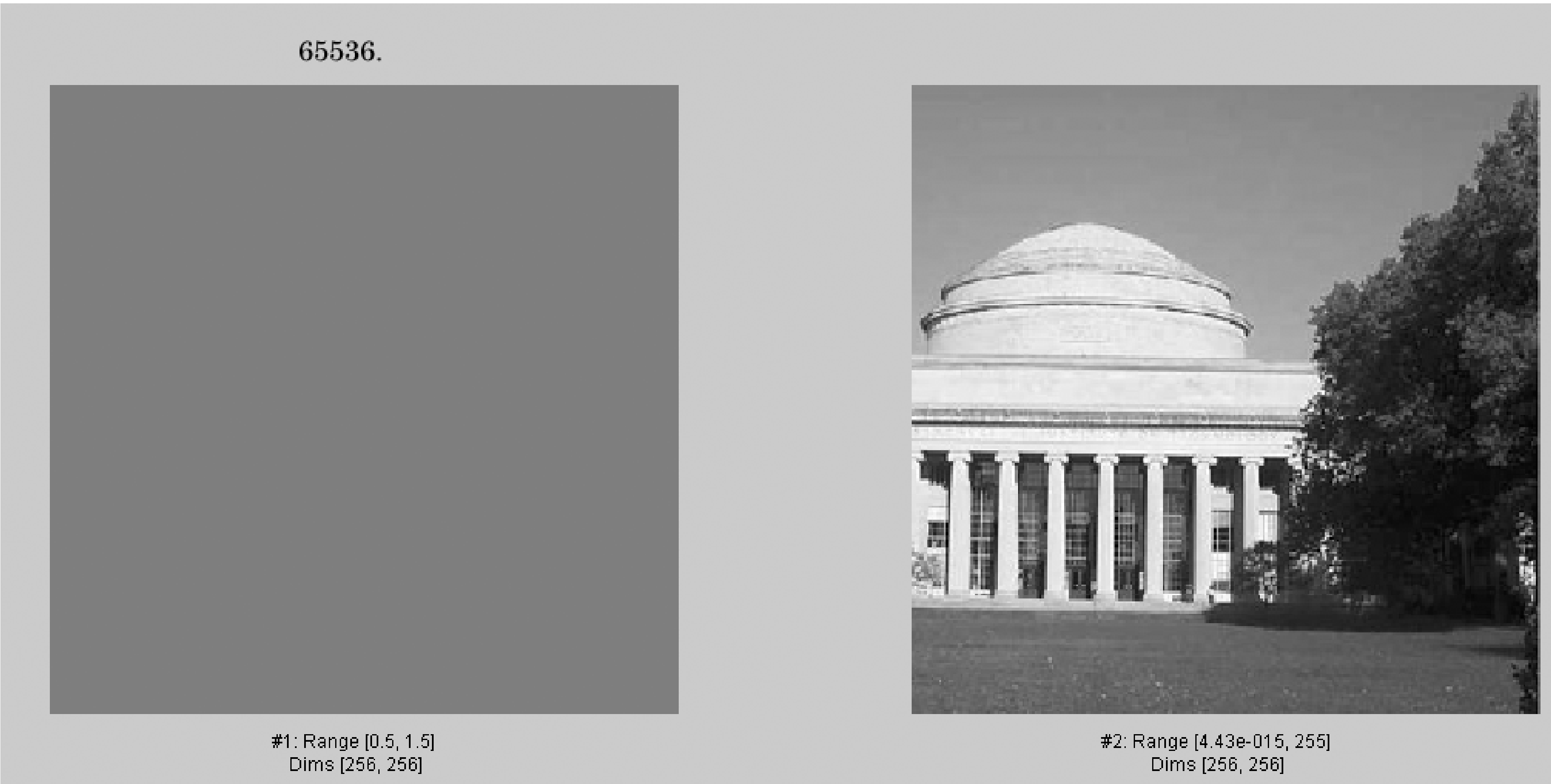
28743



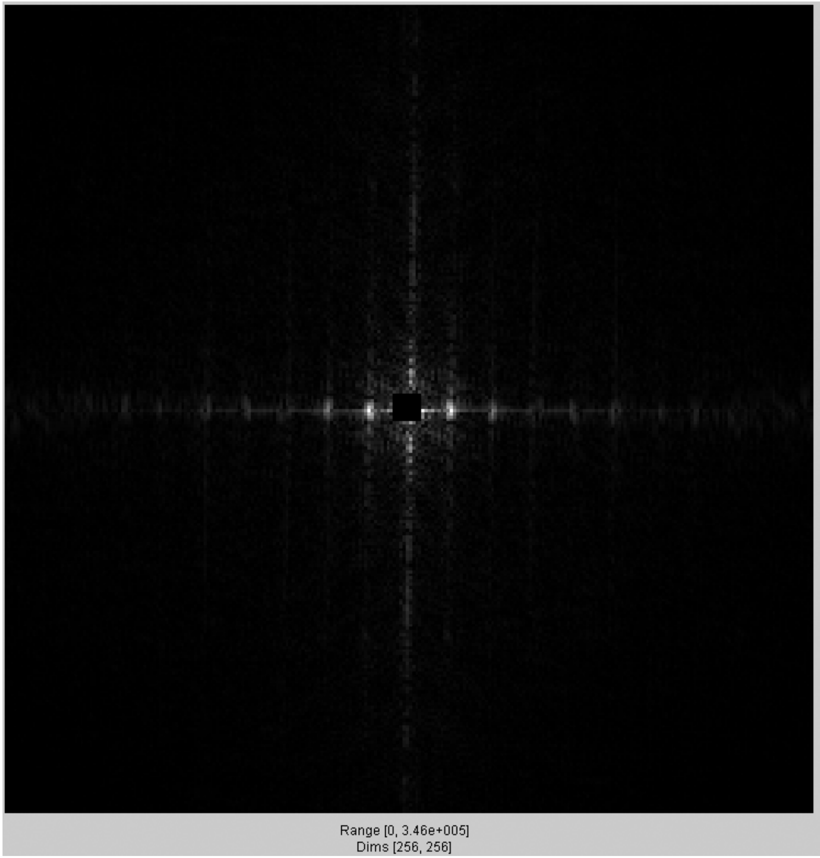
49190.



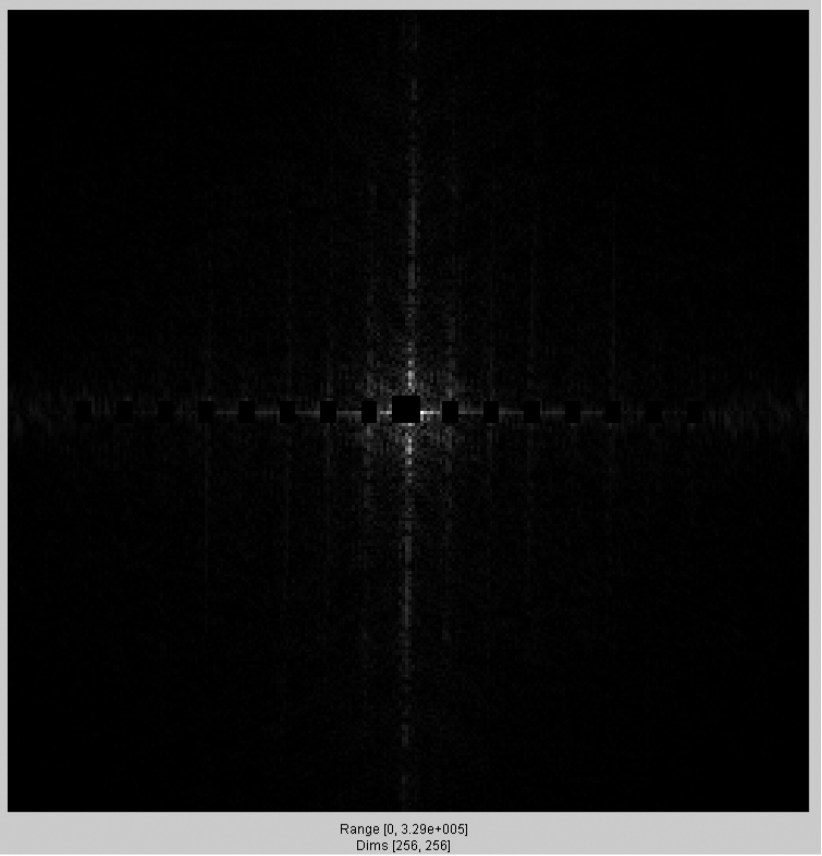
65536.

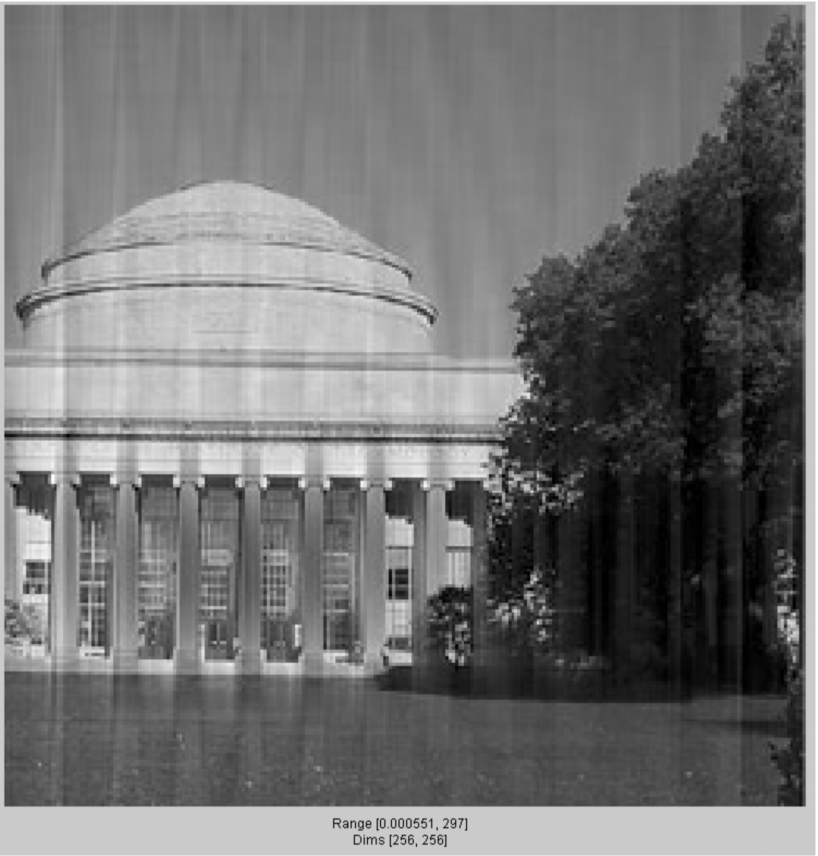

65536.
#1: Range [0.5, 1.5]
Dims [256, 256]
#2: Range [4.43e-015, 255]
Dims [256, 256]

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

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

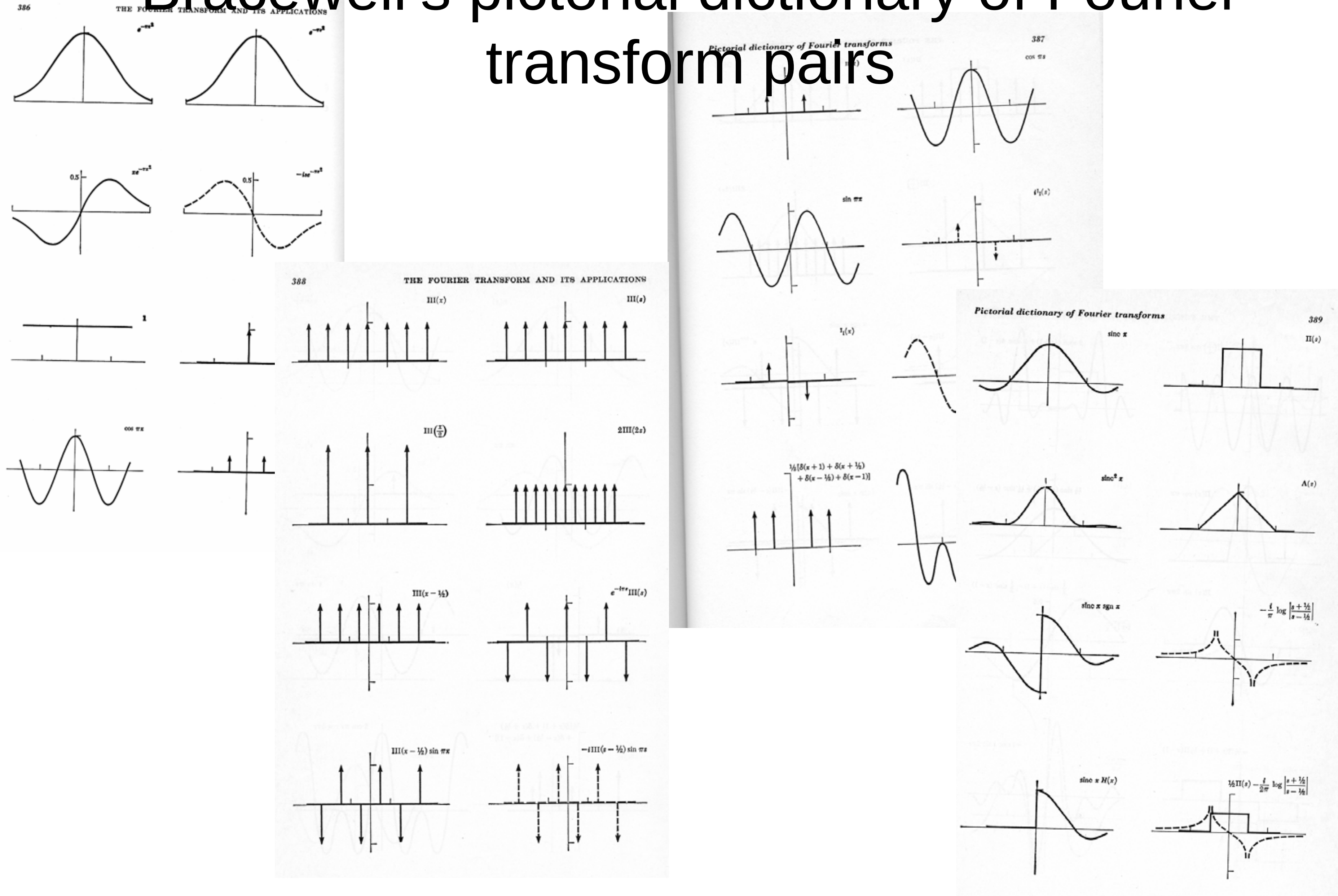
Discrete-time, continuous frequency Fourier transform pairs

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.
- There is a fast algorithm for performing the DFT, allowing for efficient signal filtering.
- The Fourier domain offers an alternative domain for understanding and manipulating the image.

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

$$f = g \otimes h$$

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

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

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

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

$$f = g \otimes h$$

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

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

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

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

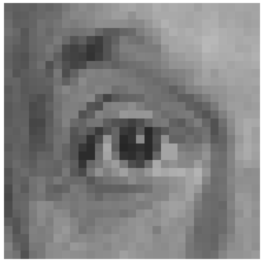
$$= \sum_{k,l} G[m,n] e^{-\pi i \left(\frac{km}{M} + \frac{\ln}{N}\right)} h[k,l]$$

Perform the other DFT (circular boundary conditions)

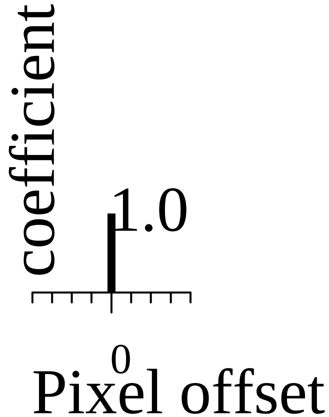
$$= G[m,n] H[m,n]$$

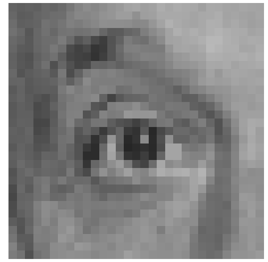
Analysis of our simple filters

Analysis of our simple filters



original

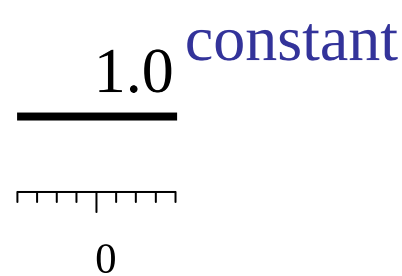




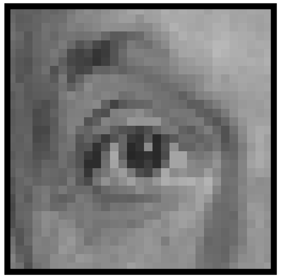
Filtered
(no change)

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

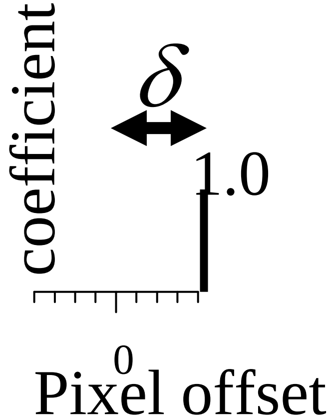
$$= 1$$

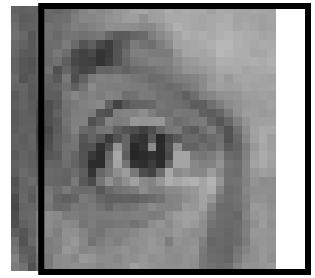


Analysis of our simple filters



original

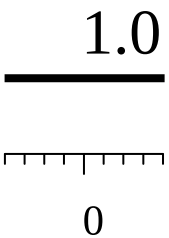




shifted

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

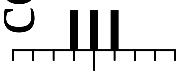
$$= e^{-\pi i \frac{\delta m}{M}}$$

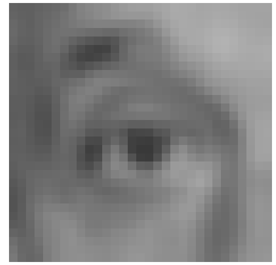


Constant magnitude, linearly shifted phase

Analysis of our simple filters

original

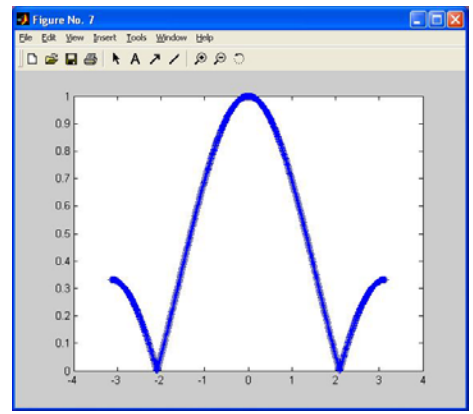




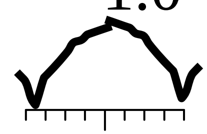
blurred

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

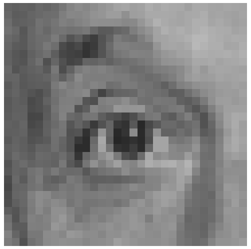
$$= \frac{1}{3} \left(1 + 2\cos\left(\frac{\pi m}{M} \right) \right)$$



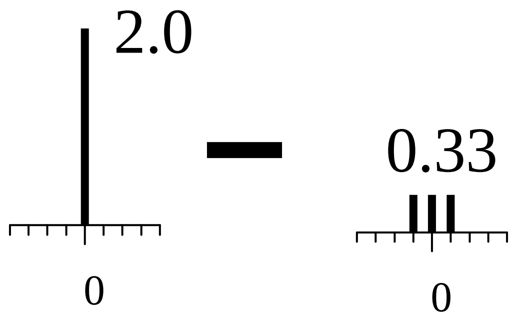
Low-pass filter

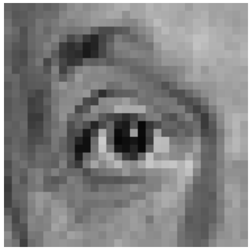


Analysis of our simple filters

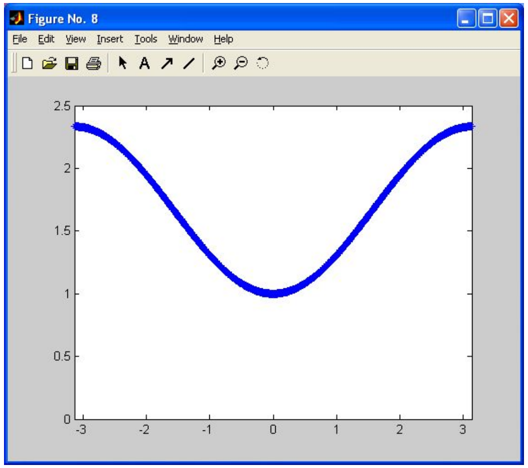


original





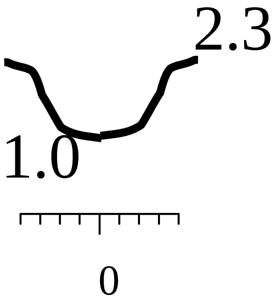
sharpened



high-pass filter

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

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

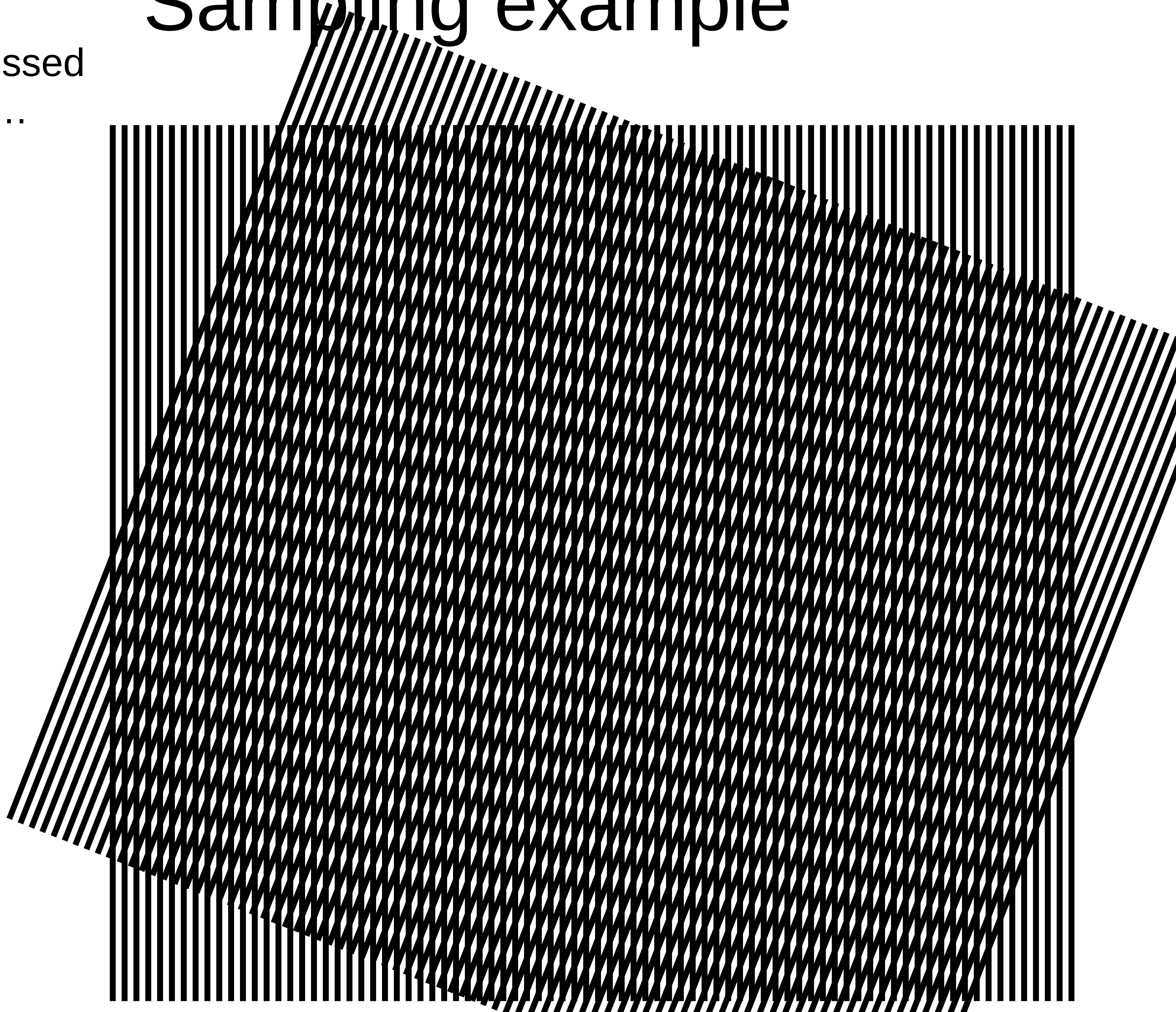


Convolution versus FFT

- 1-d FFT: O(NlogN) computation time, where N is number of samples.
- 2-d FFT: 2N(NlogN), where N is number of pixels on a side
- Convolution: K N^2, where K is number of samples in kernel
- Say $N=2^{10}$, K=100. 2-d FFT: 20 2^{20}, while convolution gives 100 2^{20}

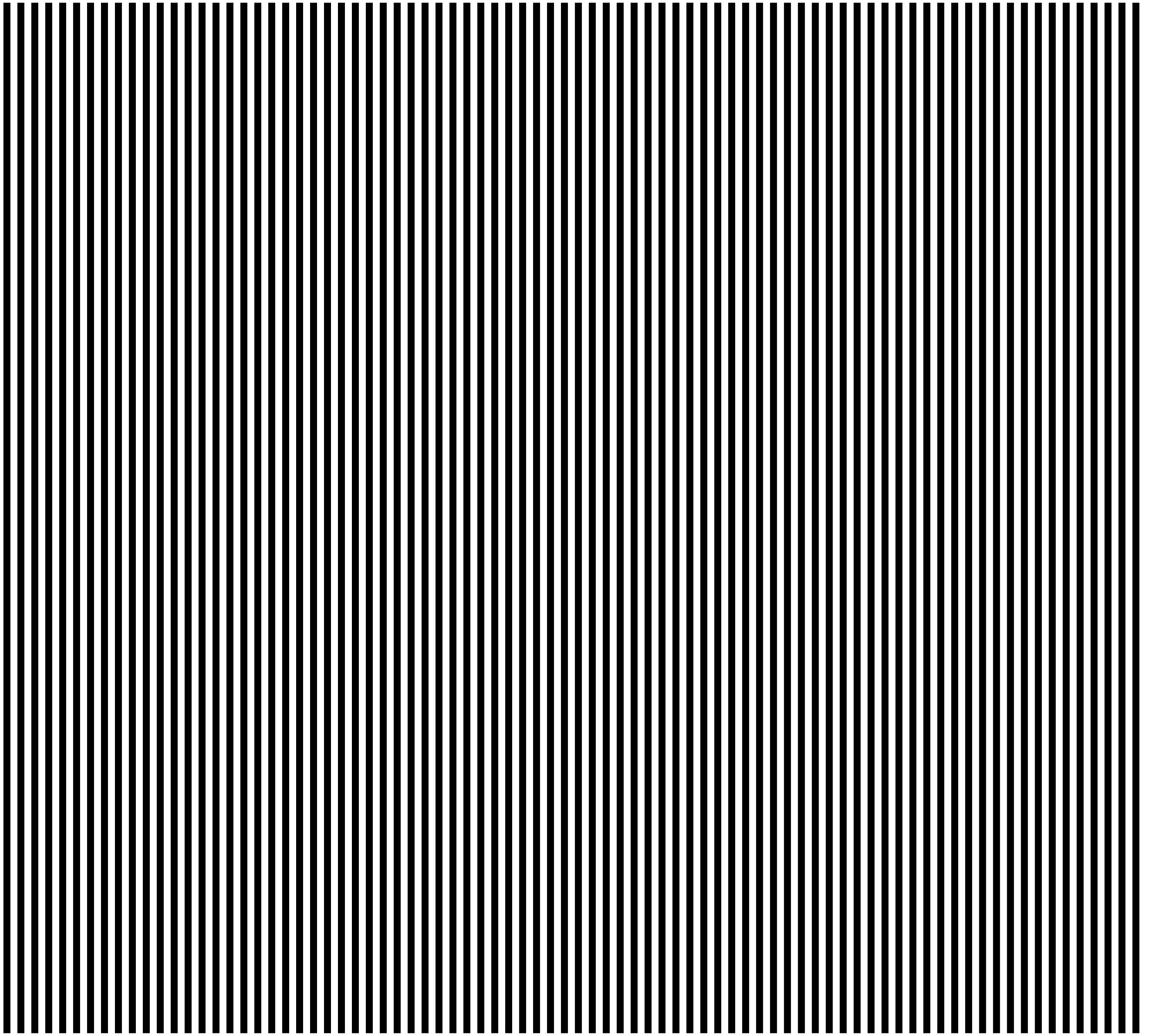
Sampling example

Analyze crossed gratings…



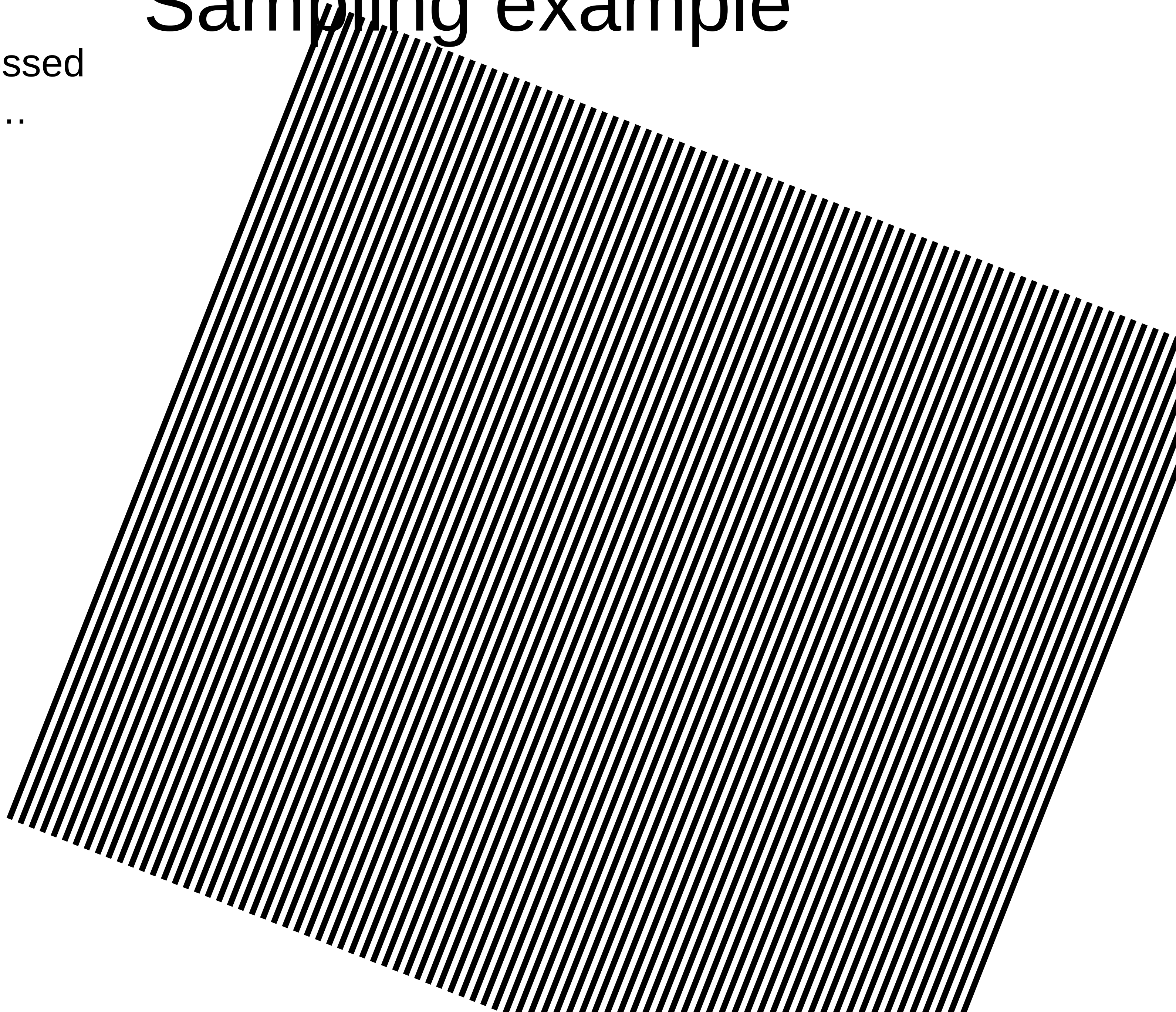
Sampling example

Analyze crossed gratings…



Sampling example

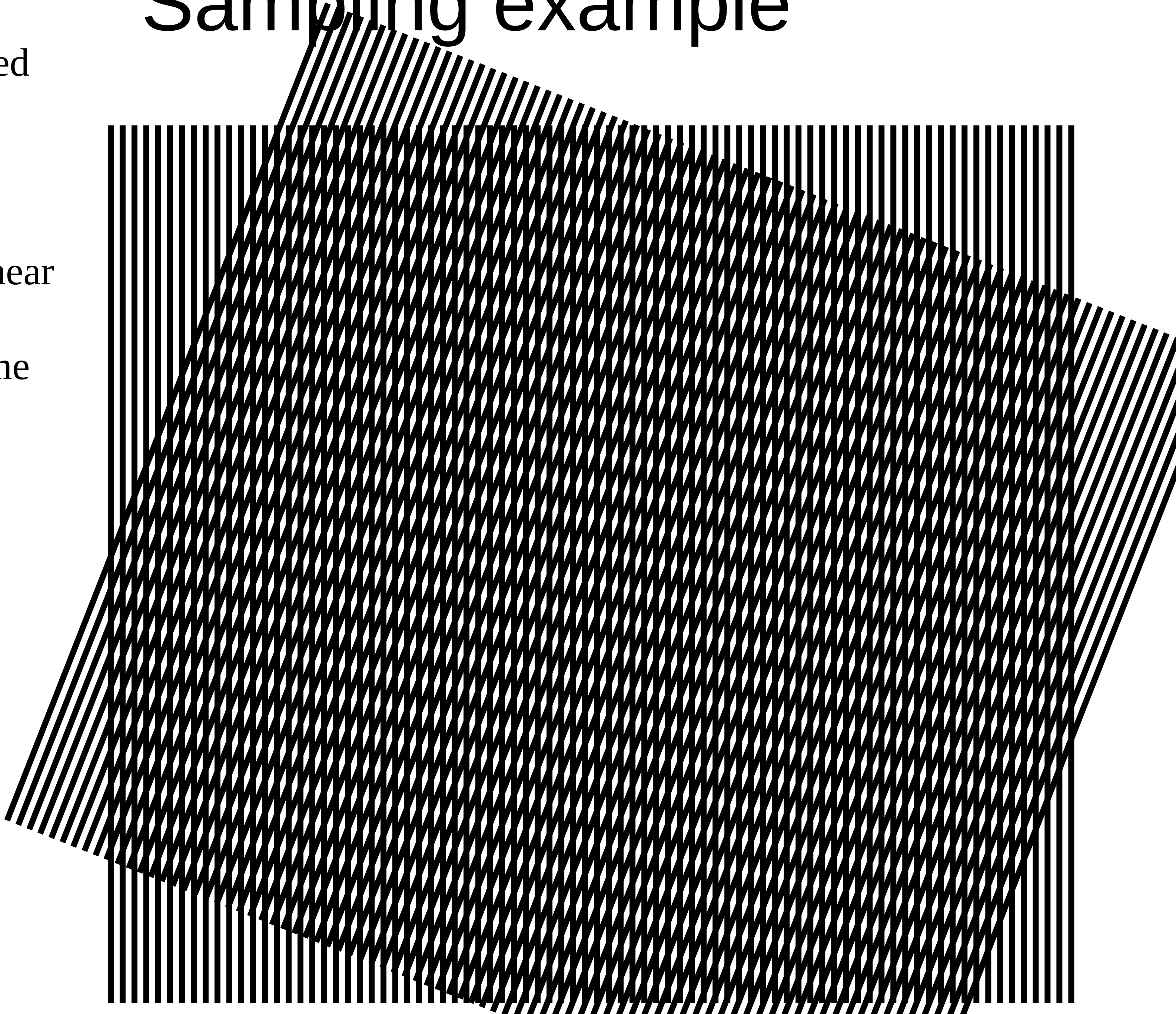
Analyze crossed gratings…

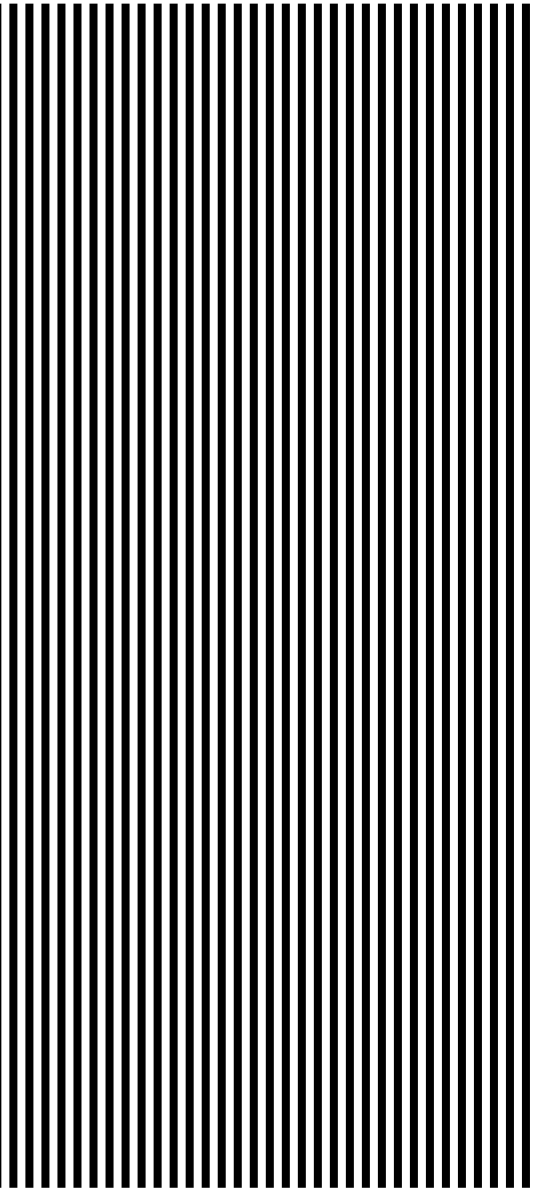


Sampling example

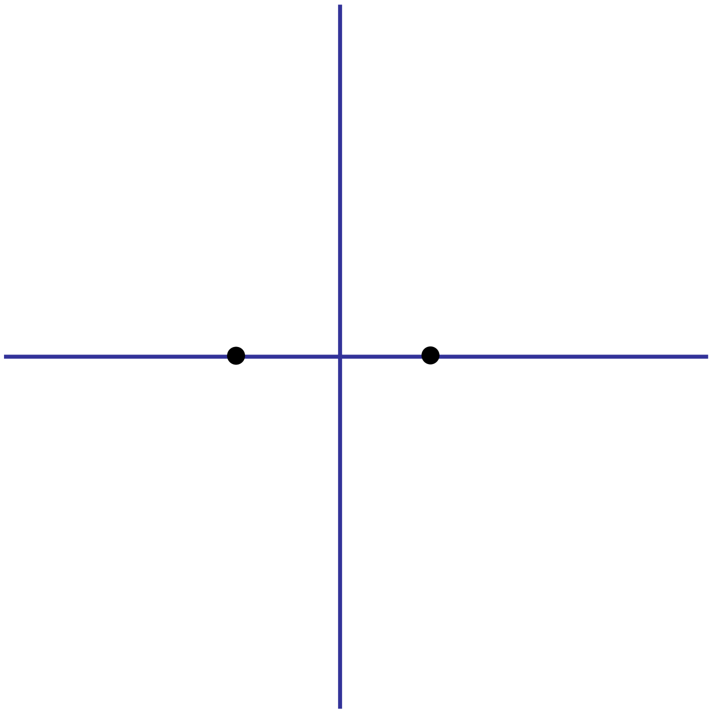
Analyze crossed gratings…

Where does perceived near horizontal grating come from?

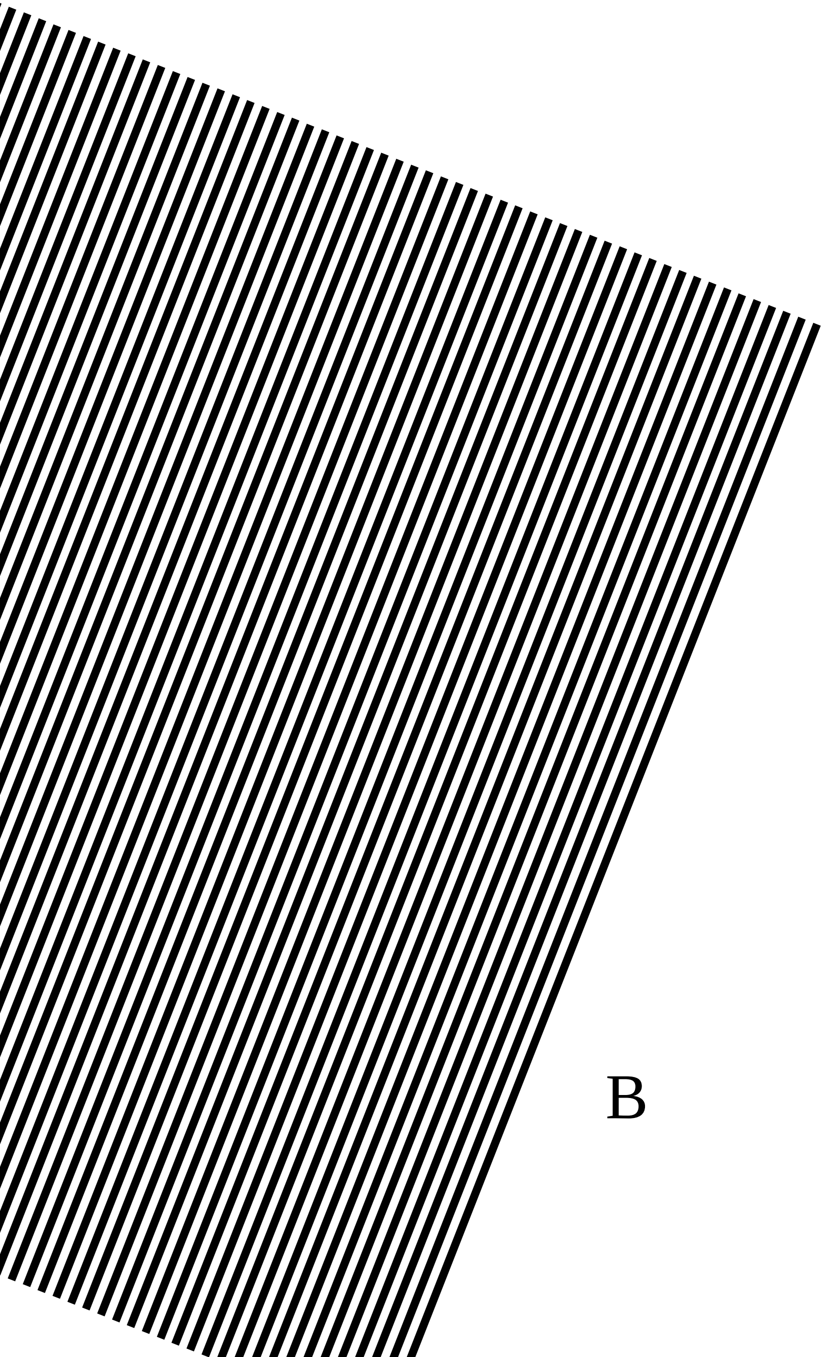




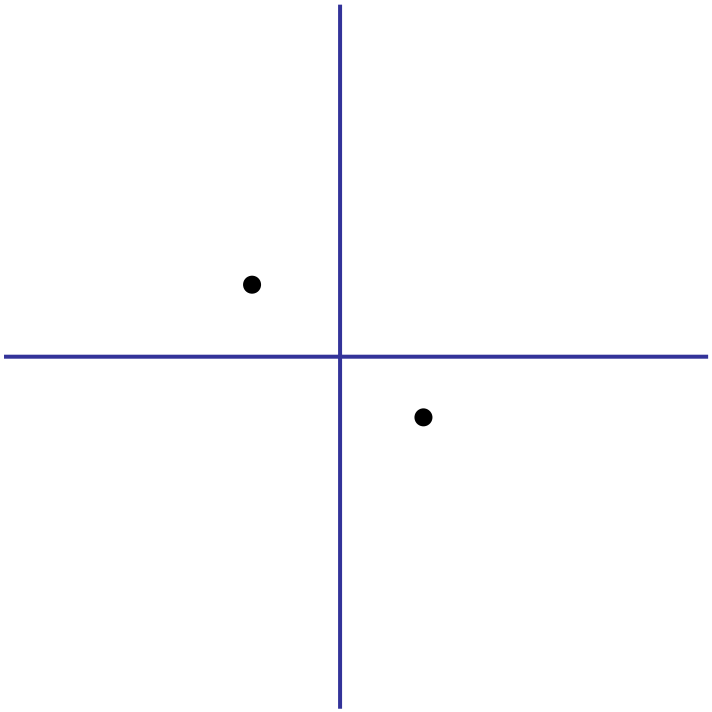
A



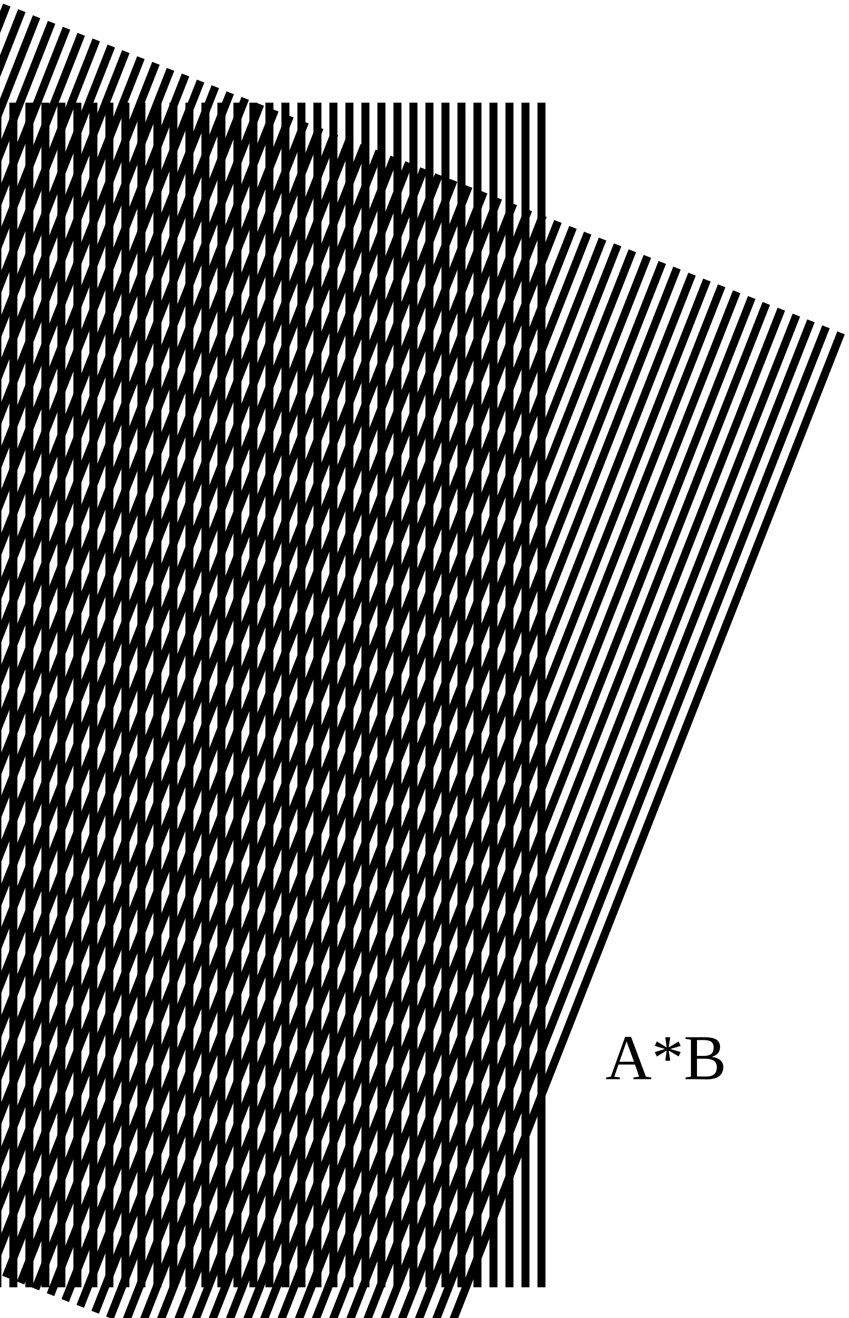
F(A)



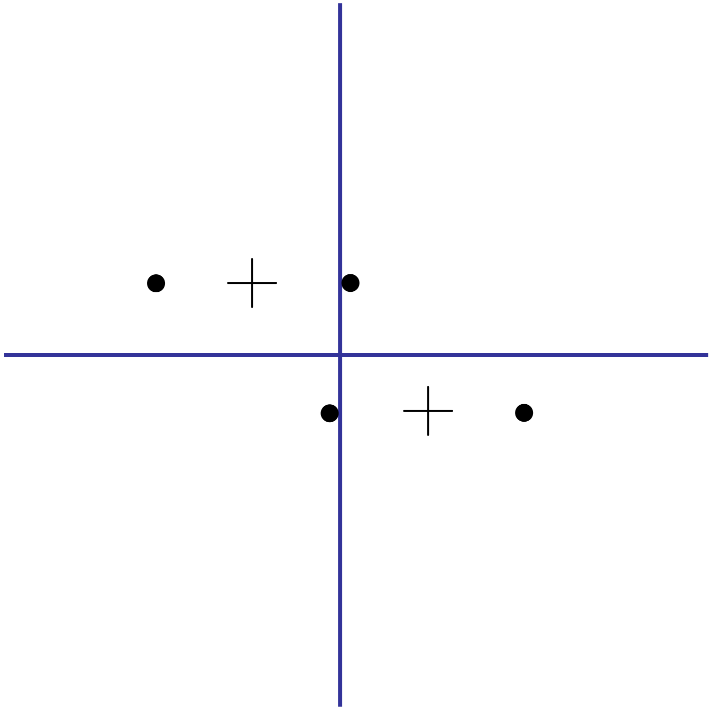
B



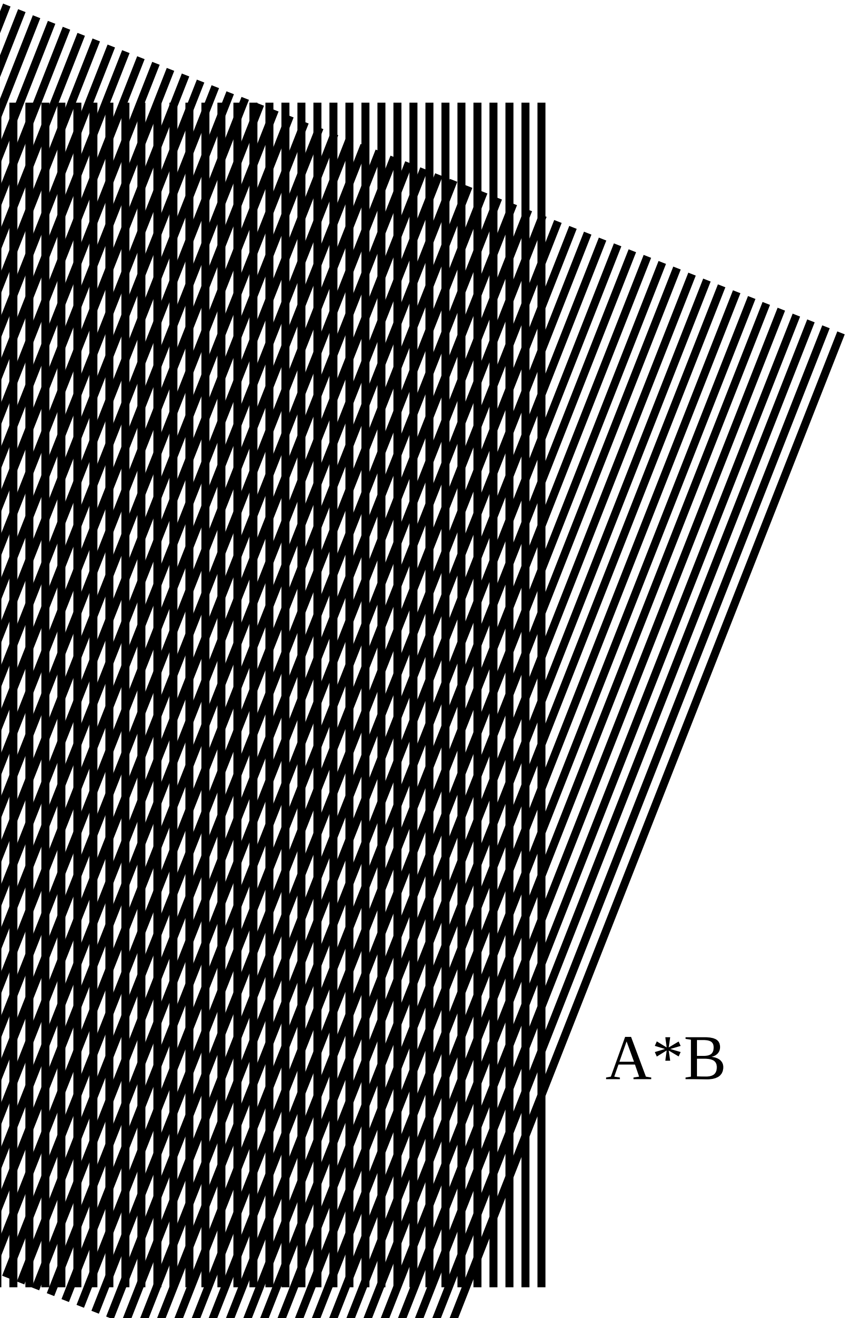
F(B)



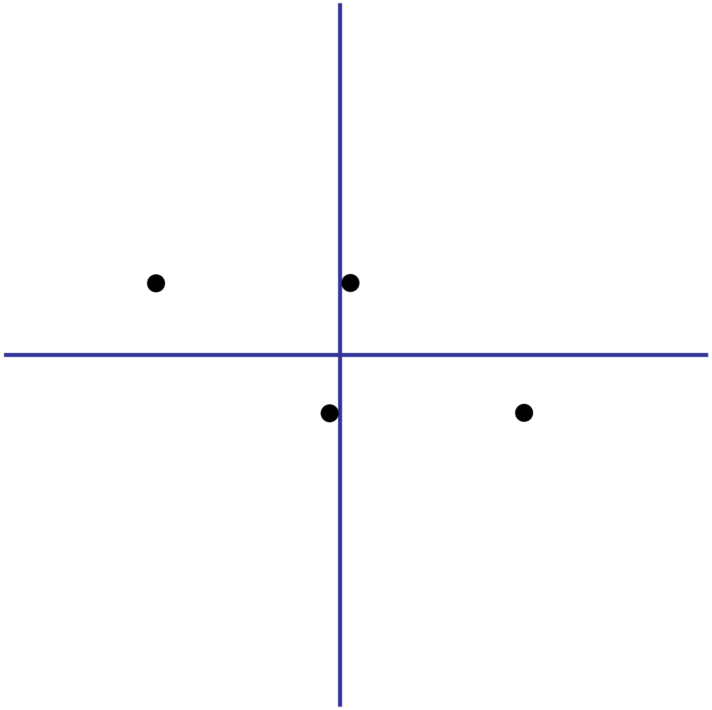
A*B



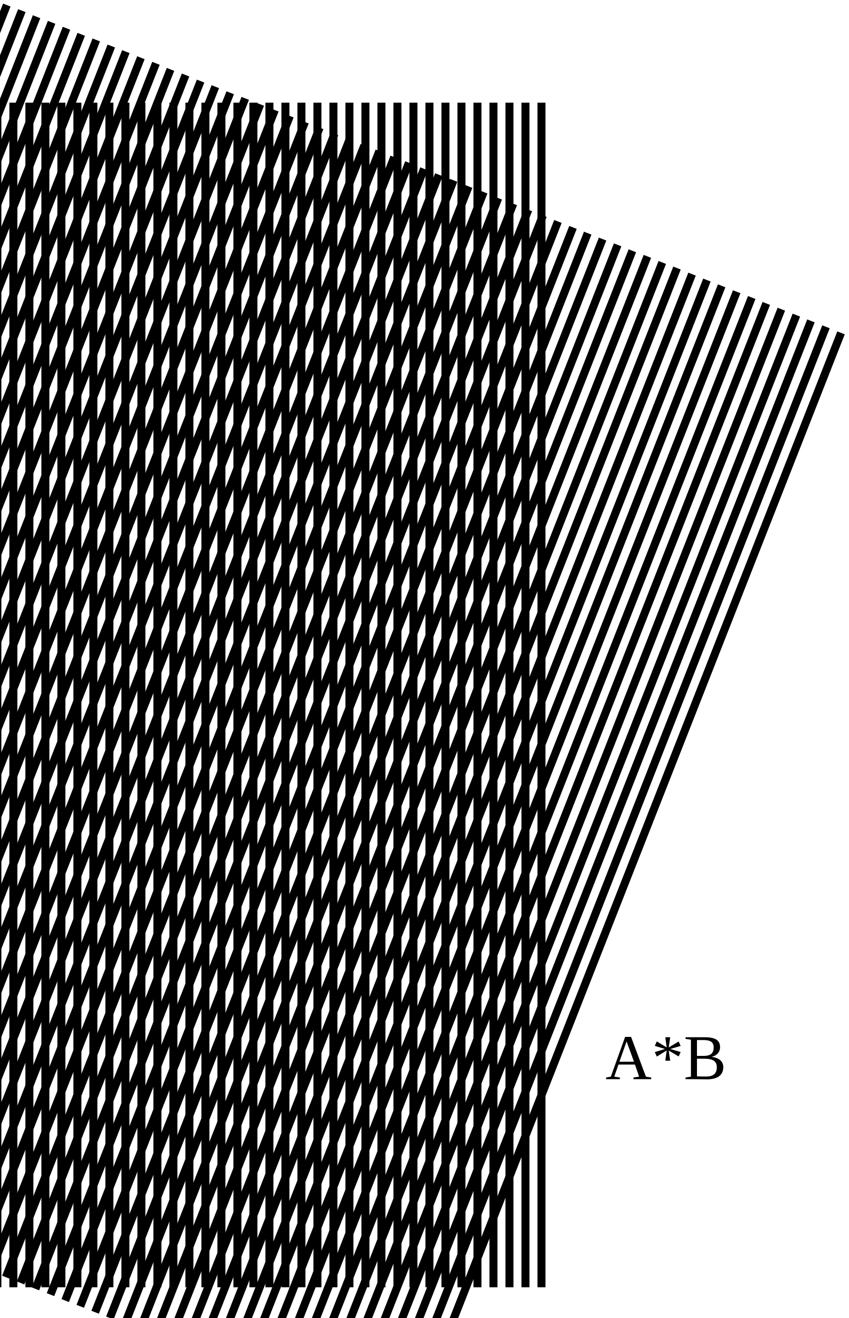
F(A)**F(B)



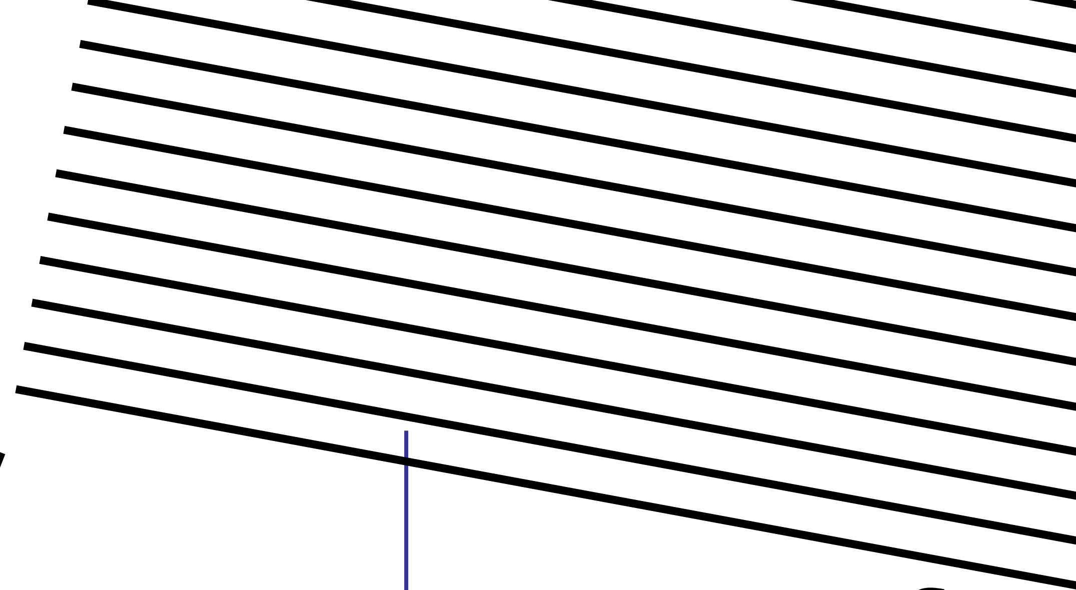
A*B



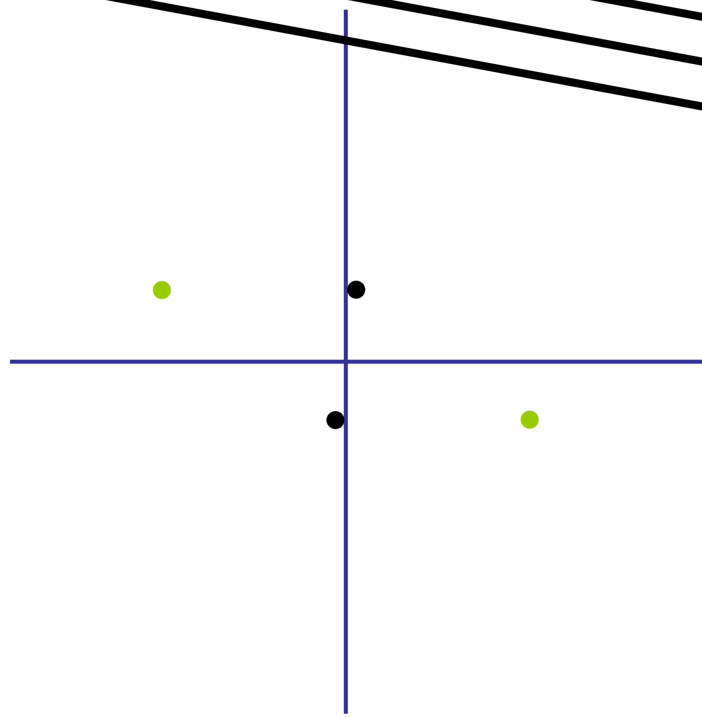
F(A)**F(B)



A*B

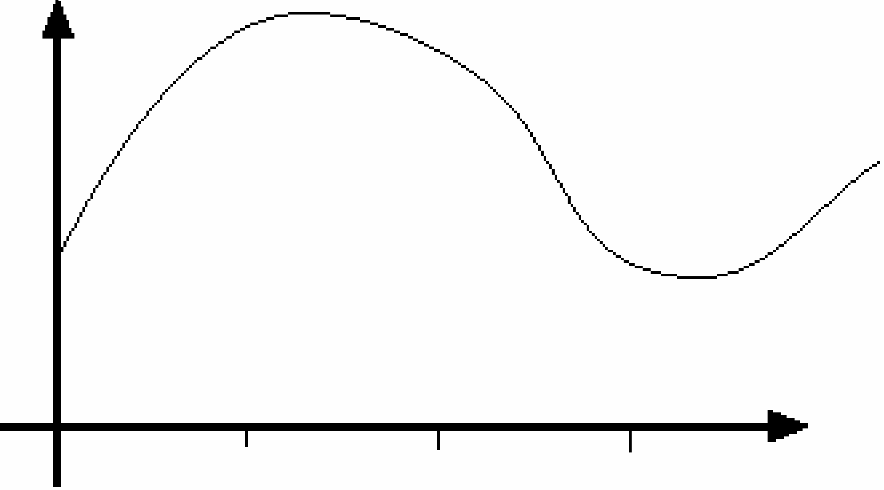


C

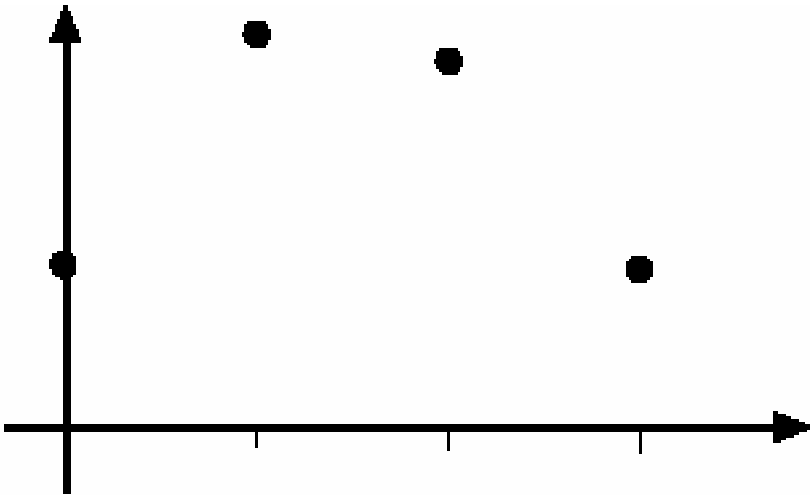


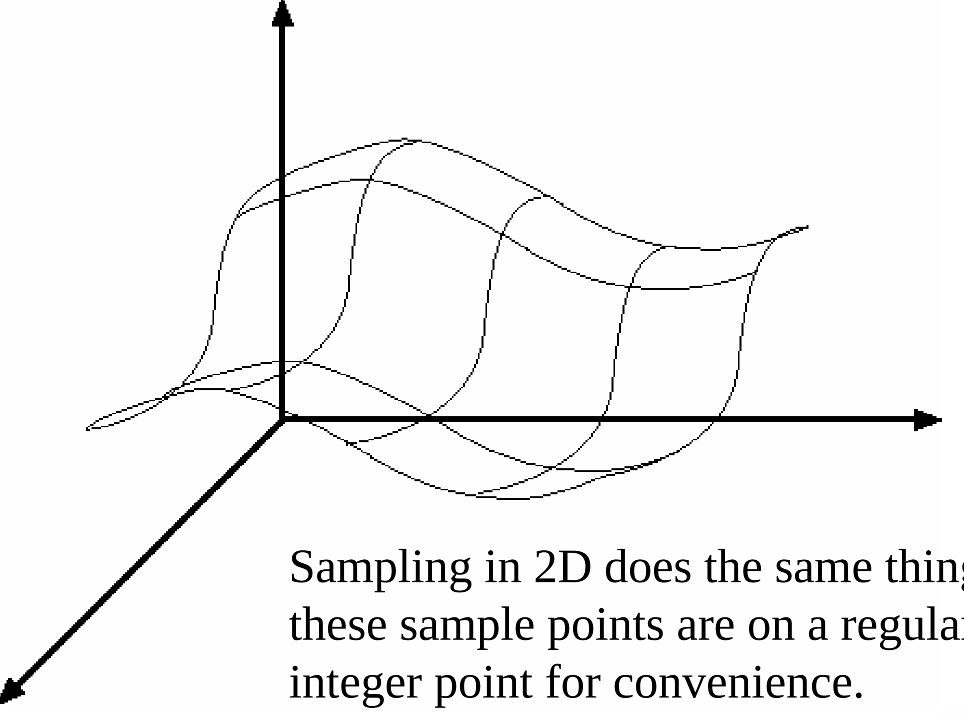
Lowpass(F(A)**F(B))
~=F(C)

Sampling and aliasing

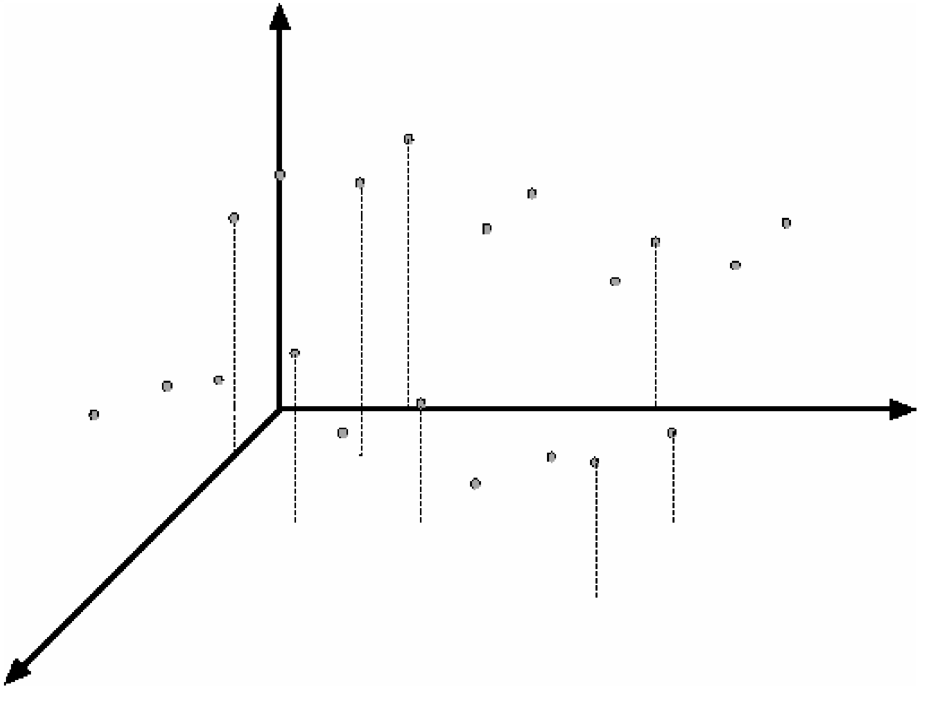


Sampling in 1D takes a continuous function and replaces it with a vector of values, consisting of the function's values at a set of sample points. We'll assume that these sample points are on a regular grid, and can place one at each integer for convenience.





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

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

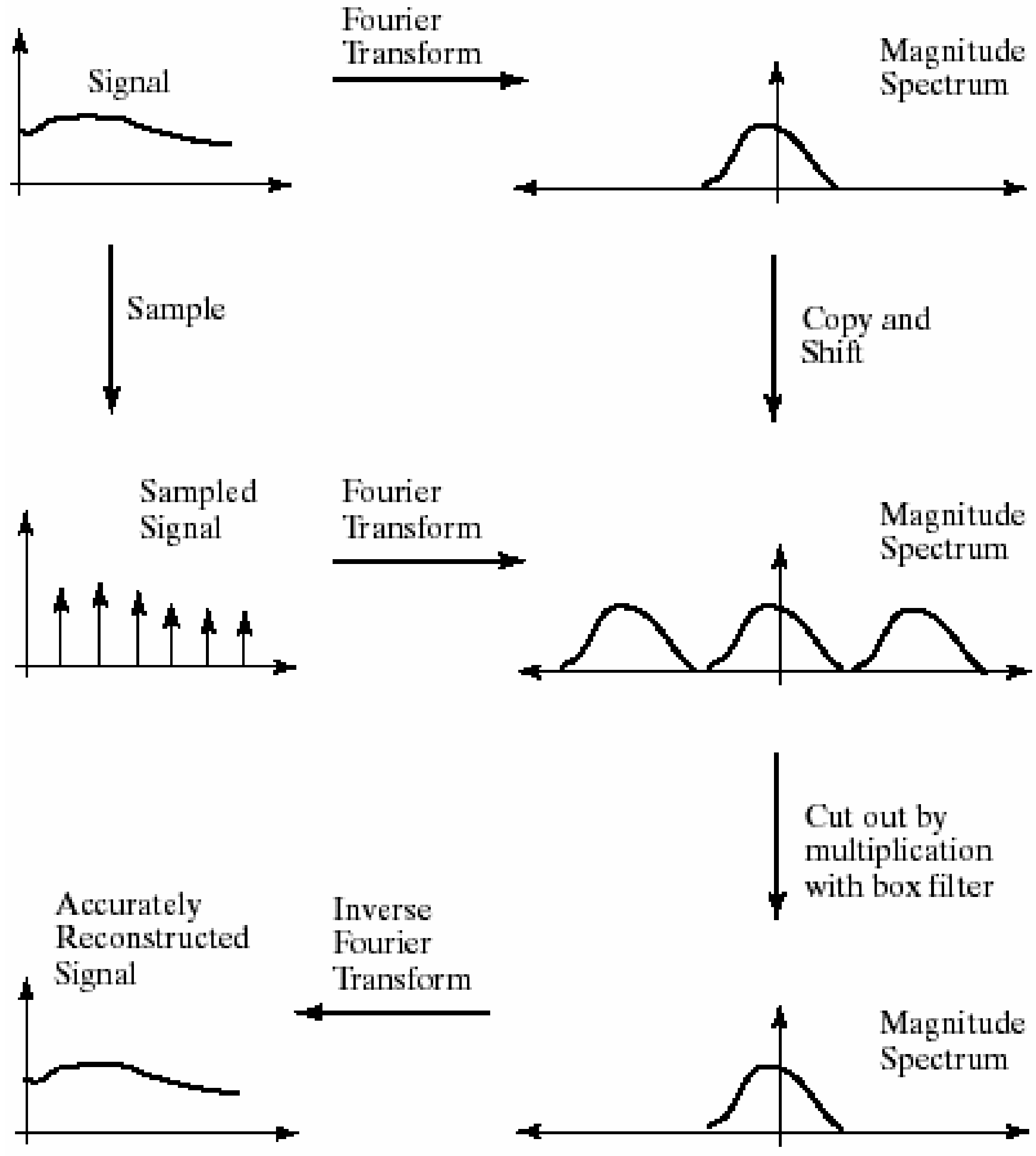
$$= f(x,y)\sum_{i=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} \delta(x-i, y-j)$$

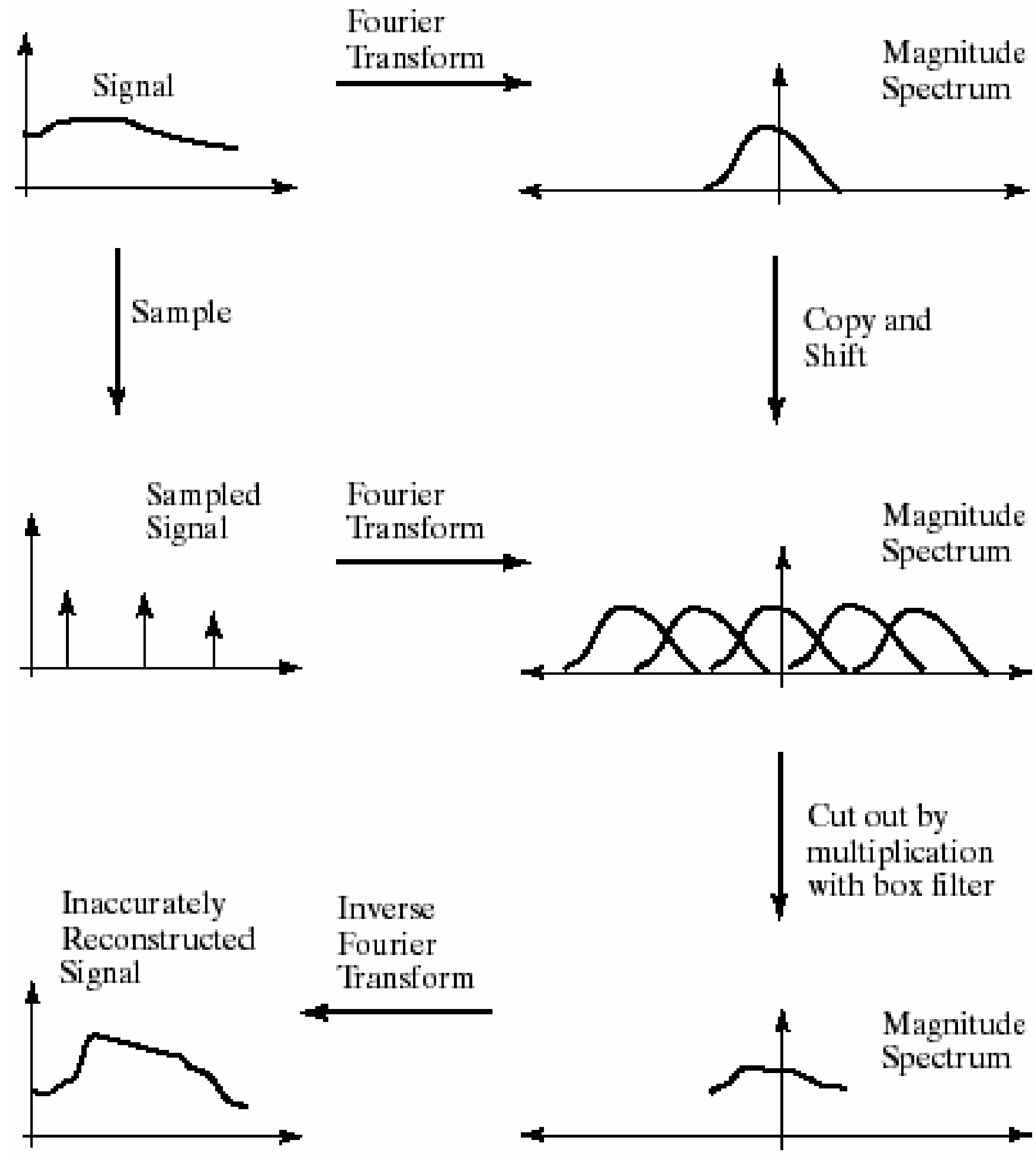
The Fourier transform of a sampled signal

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

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

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


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


Fourier
Transform
Signal
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

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.

Scaled representations

- Big bars (resp. spots, hands, etc.) and little bars are both interesting
  - Stripes and hairs, say
- Inefficient to detect big bars with big filters
  - And there is superfluous detail in the filter kernel
- Alternative:
  - Apply filters of fixed size to images of different sizes
  - Typically, a collection of images whose edge length changes by a factor of 2 (or root 2)
  - This is a pyramid (or Gaussian pyramid) by visual analogy

Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid

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

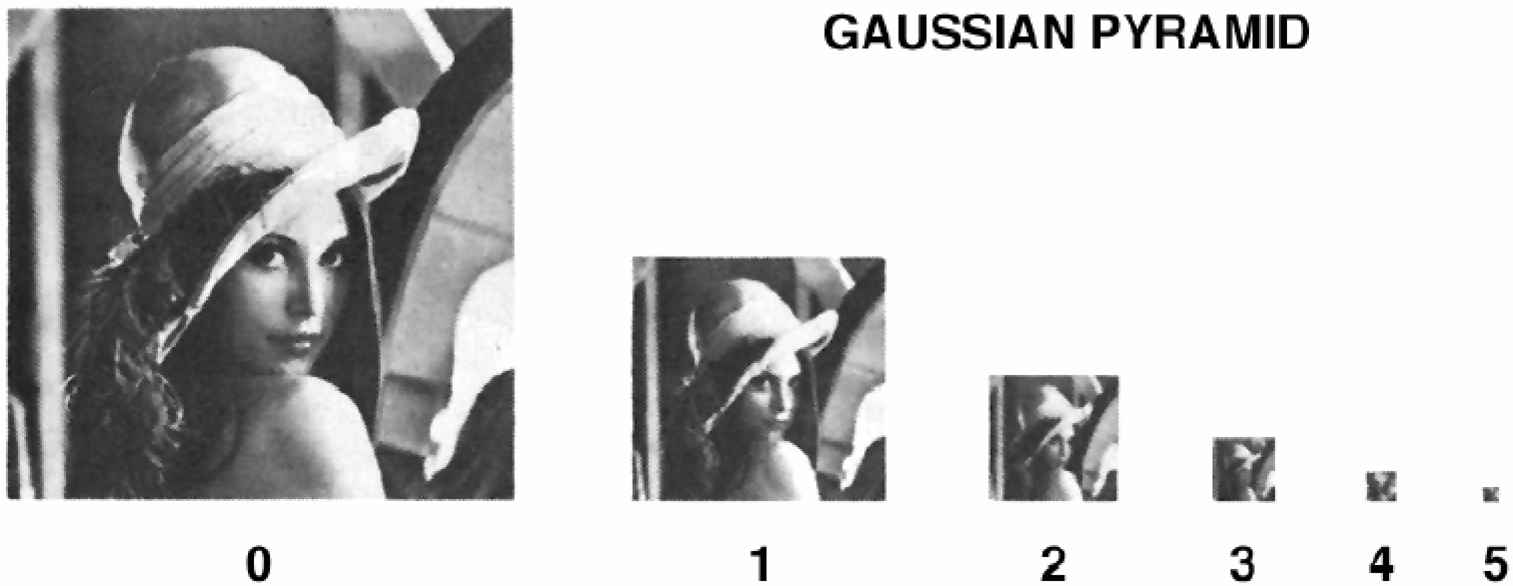
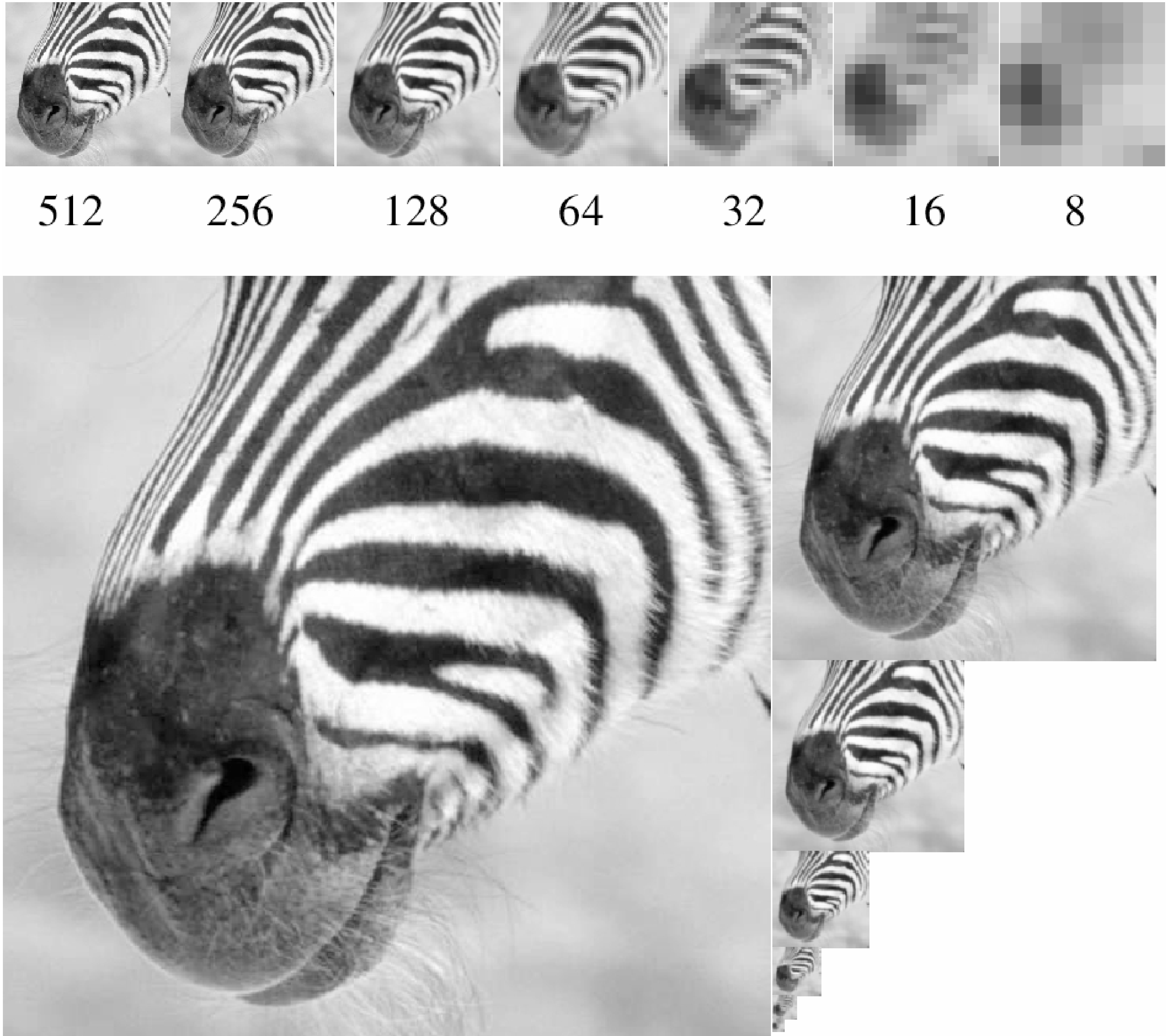


Fig. 4. First six levels of the Gaussian pyramid for the "Lady" image The original image, level 0, meusures 257 by 257 pixels and each higher level array is roughly half the dimensdons of its predecessor. Thus, level 5 measures just 9 by 9 pixels.

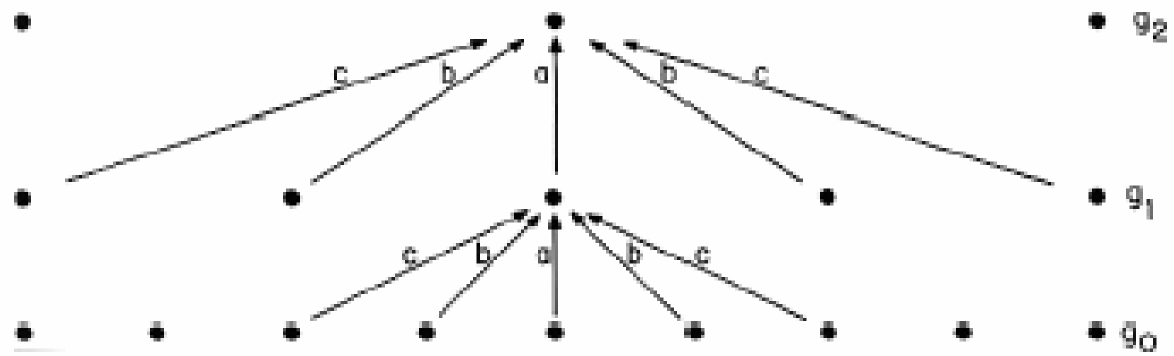
http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983


512
256
128
64
32
16
8

The computational advantage of pyramids

GAUSSIAN PYRAMID



g_0 = IMAGE

g_L = REDUCE $[g_{L-1}]$

Fig 1. A one-dimensional graphic representation of the process which generates a Gaussian pyramid Each row of dots represents nodes within a level of the pyramid. The value of each node in the zero level is just the gray level of a corresponding image pixel. The value of each node in a high level is the weighted average of node values in the next lower level. Note that node spacing doubles from level to level, while the same weighting pattern or "generating kernel" is used to generate all levels.

http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

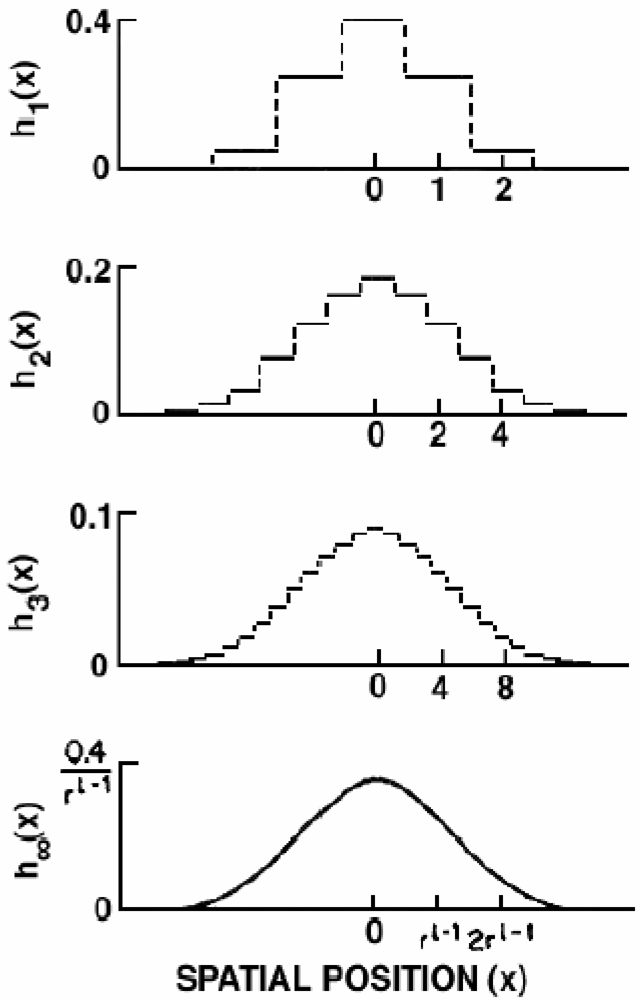
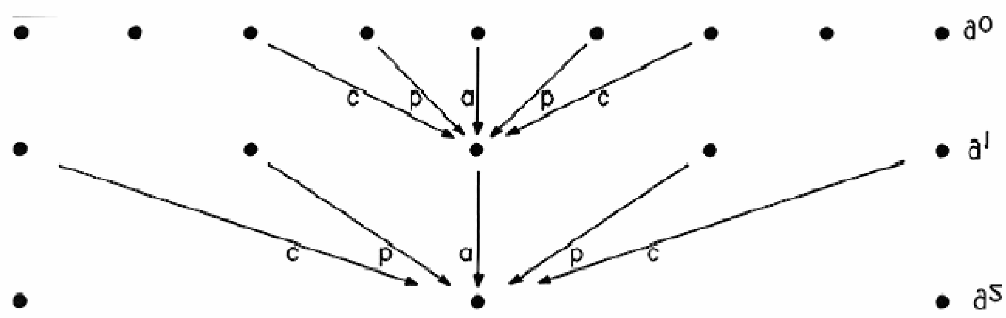


Fig. 2. The equivalent weighting functions $h_l(x)$ for nodes in levels 1, 2, 3, and infinity of the Gaussian pyramid. Note that axis scales have been adjusted by factors of 2 to aid comparison Here the parameter a of the generating kernel is 0.4, and the resulting equivalent weighting functions closely resemble the Gaussian probability density functions.



http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

Convolution and subsampling as a matrix multiply (1-d case)

U1 =

1	4	6	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	4	6	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	4	6	4	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	4	6	4	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	4	6	4	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	4	6	4	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	4	6	4	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	6	4	1	0

Next pyramid level

U2 =

1	4	6	4	1	0	0	0
0	0	1	4	6	4	1	0
0	0	0	0	1	4	6	4
0	0	0	0	0	0	1	4

b * a, the combined effect of the two pyramid levels

```
>> U2 * U1

ans =

  1   4  10  20  31  40  44  40  31  20  10   4   1   0   0   0   0   0   0   0
  0   0   0   0   1   4  10  20  31  40  44  40  31  20  10   4   1   0   0   0
  0   0   0   0   0   0   0   0   1   4  10  20  31  40  44  40  30  16   4   0
  0   0   0   0   0   0   0   0   0   0   0   0   1   4  10  20  25  16   4   0
```

Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid

Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid

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

Laplacian pyramid algorithm

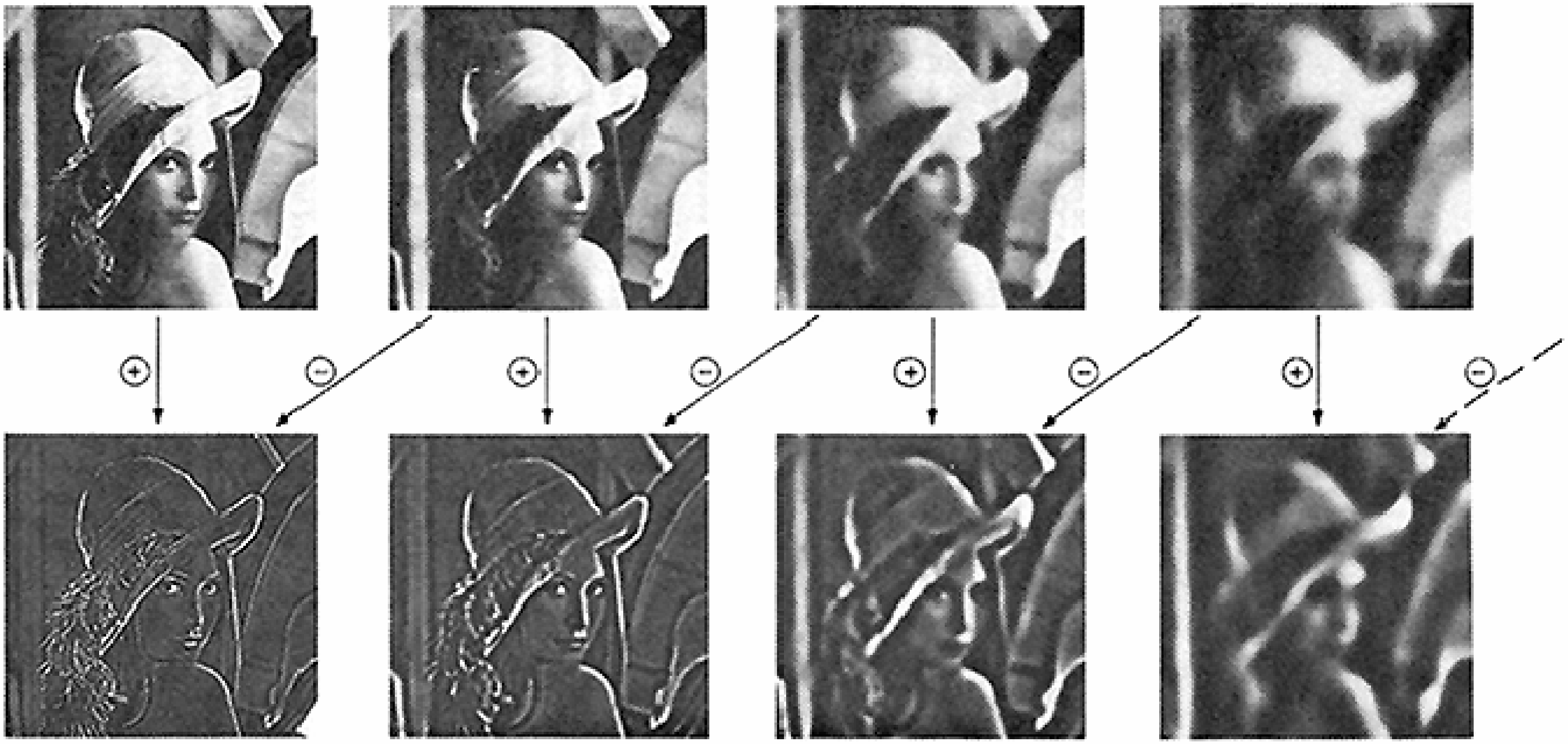
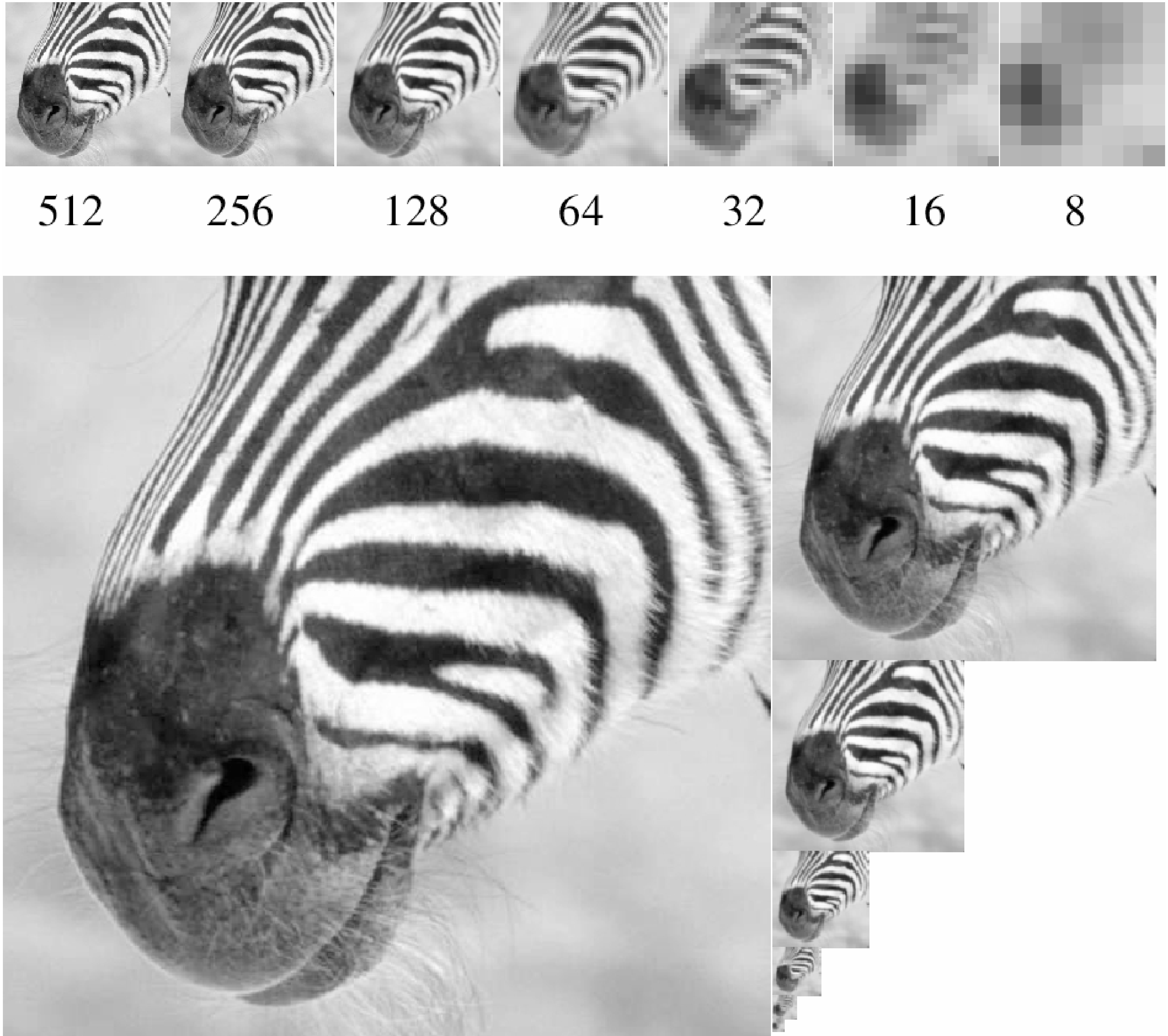
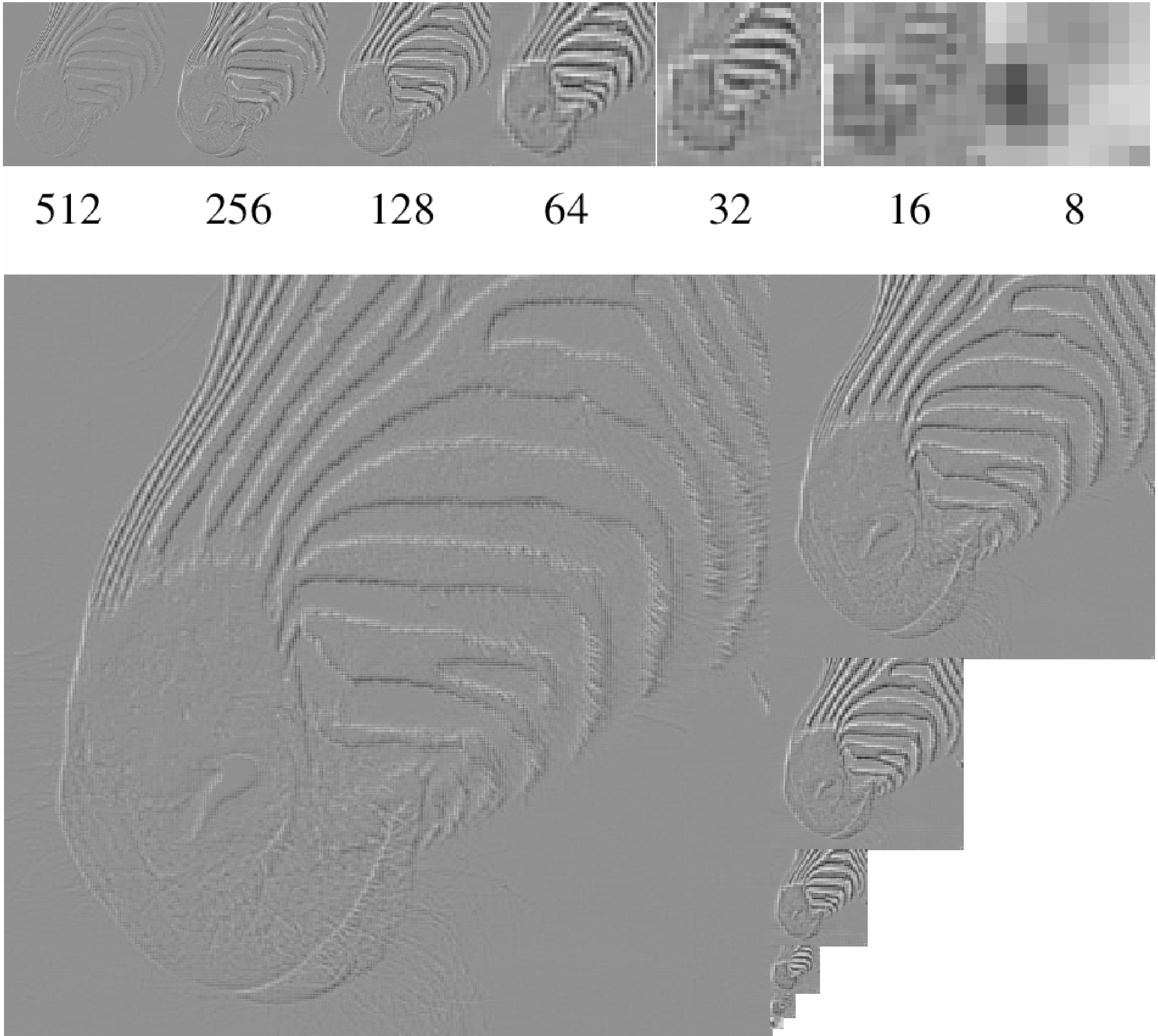


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983


512
256
128
64
32
16
8


512
256
128
64
32
16
8

Application to image compression

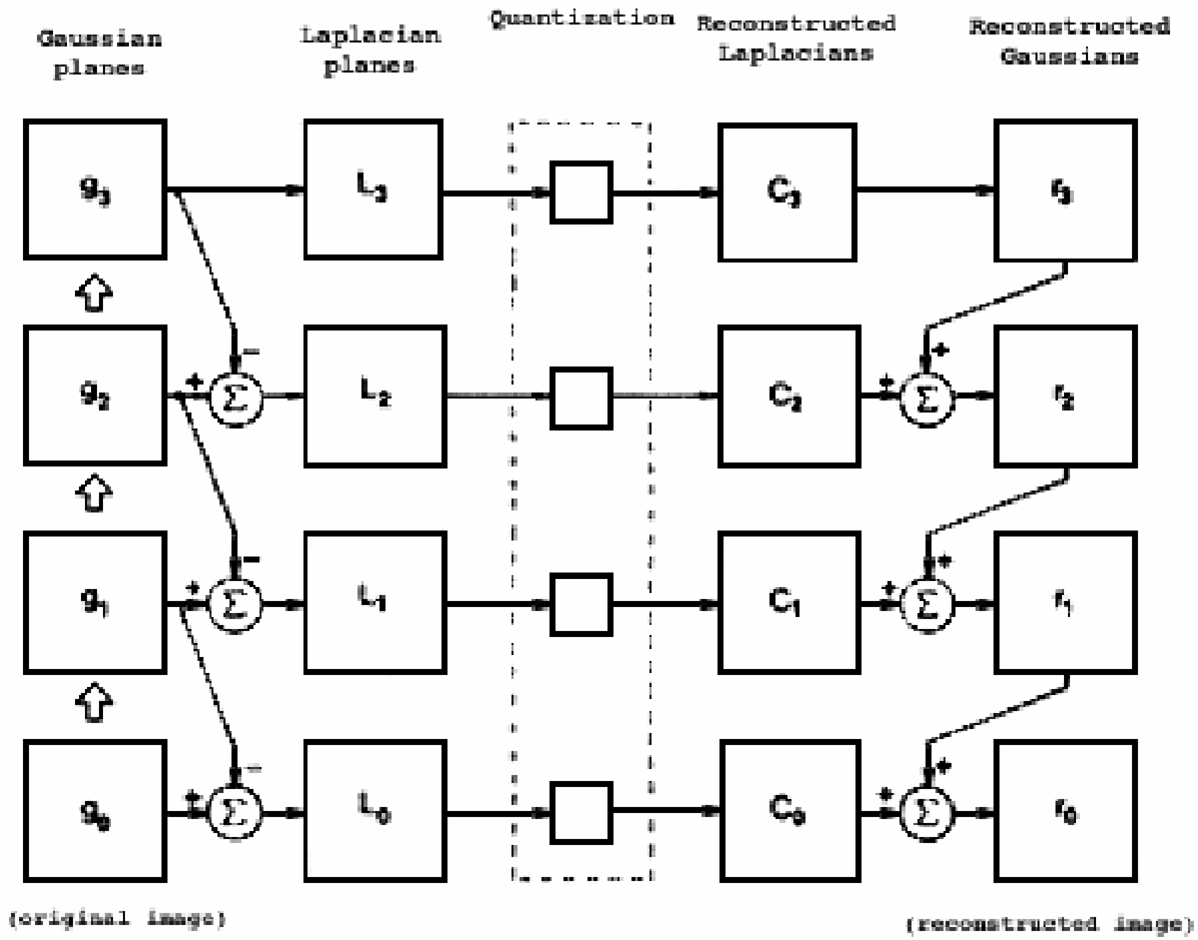


Fig. 10. A summary of the steps in Laplacian pyramid coding and decoding. First, the original image g_0 (lower left) is used to generate Gaussian pyramid levels g_1, g_2, … through repeated local averaging. Levels of the Laplacian pyramid L_0, L_1, … are then computed as the differences between adjacent Gaussian levels. Laplacian pyramid elements are quantized to yield the Laplacian pyramid code C_0, C_1, C_2, …. Finally, a reconstructed image r_0 is generated by summing levels of the code pyramid.

http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid

What is a good representation for image analysis?

(Goldilocks and the three representations)

- Fourier transform domain tells you "what" (textural properties), but not "where". In space, this representation is too spread out.
- Pixel domain representation tells you "where" (pixel location), but not "what". In space, this representation is too localized
- Want an image representation that gives you a local description of image events—what is happening where. That representation might be "just right".

Wavelets/QMF's

transformed image

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

Vectorized image

Fourier transform, or
Wavelet transform, or
Steerable pyramid transform

$$U = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

```
>> inv(U)

ans =

    0.5000    0.5000
    0.5000   -0.5000
```

U =

$$\begin{matrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{matrix}$$

```
>> inv(U)

ans =

    0.5000    0.5000         0         0         0         0         0         0
    0.5000   -0.5000         0         0         0         0         0         0
         0         0    0.5000    0.5000         0         0         0         0
         0         0    0.5000   -0.5000         0         0         0         0
         0         0         0         0    0.5000    0.5000         0         0
         0         0         0         0    0.5000   -0.5000         0         0
         0         0         0         0         0         0    0.5000    0.5000
         0         0         0         0         0         0    0.5000   -0.5000
```

Simoncelli and Adelson, in "Subband coding", Kluwer, 1990.

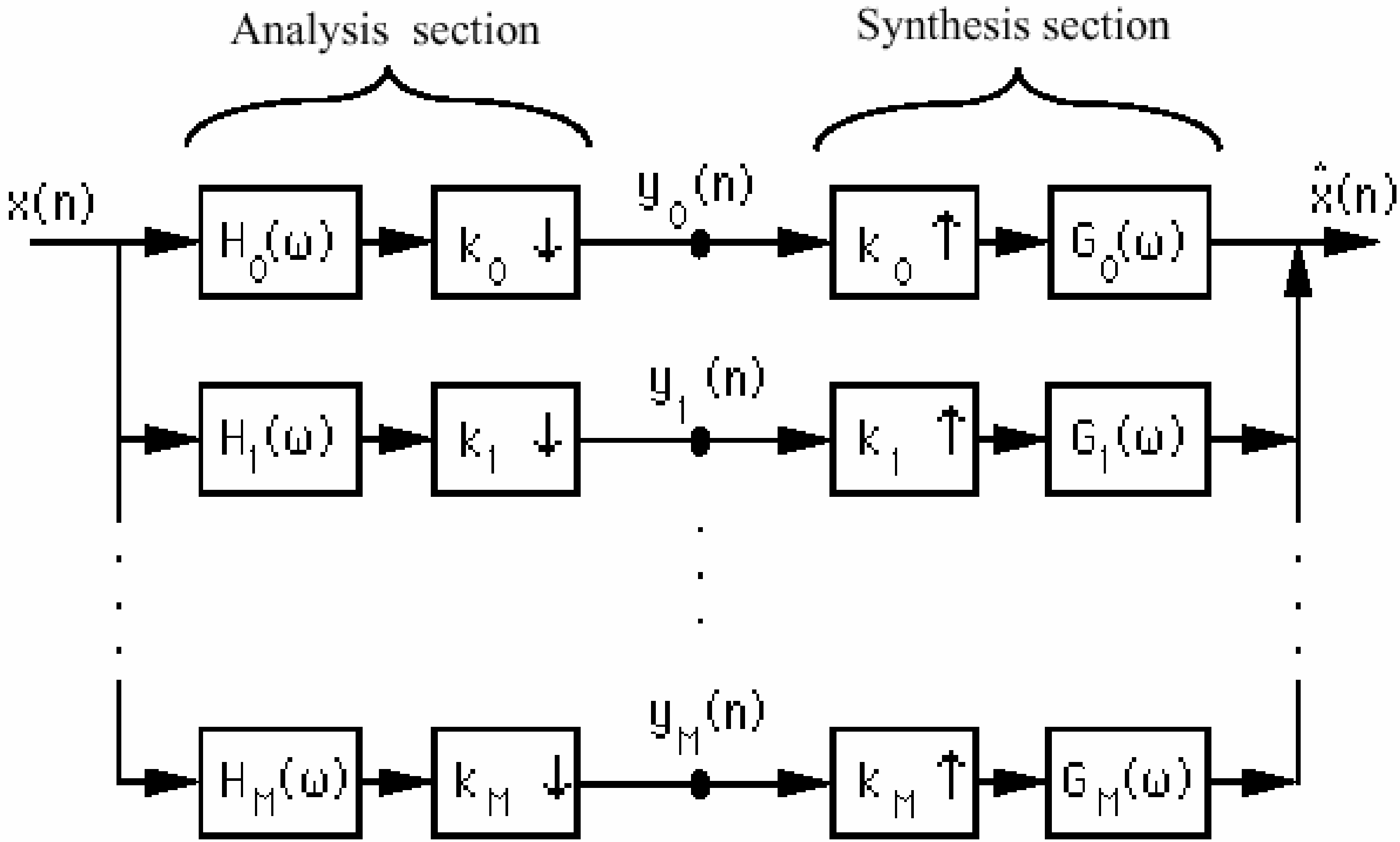


Figure 4.2: An analysis/synthesis filter bank.

Simoncelli and Adelson, in “Subband coding”, Kluwer, 1990.

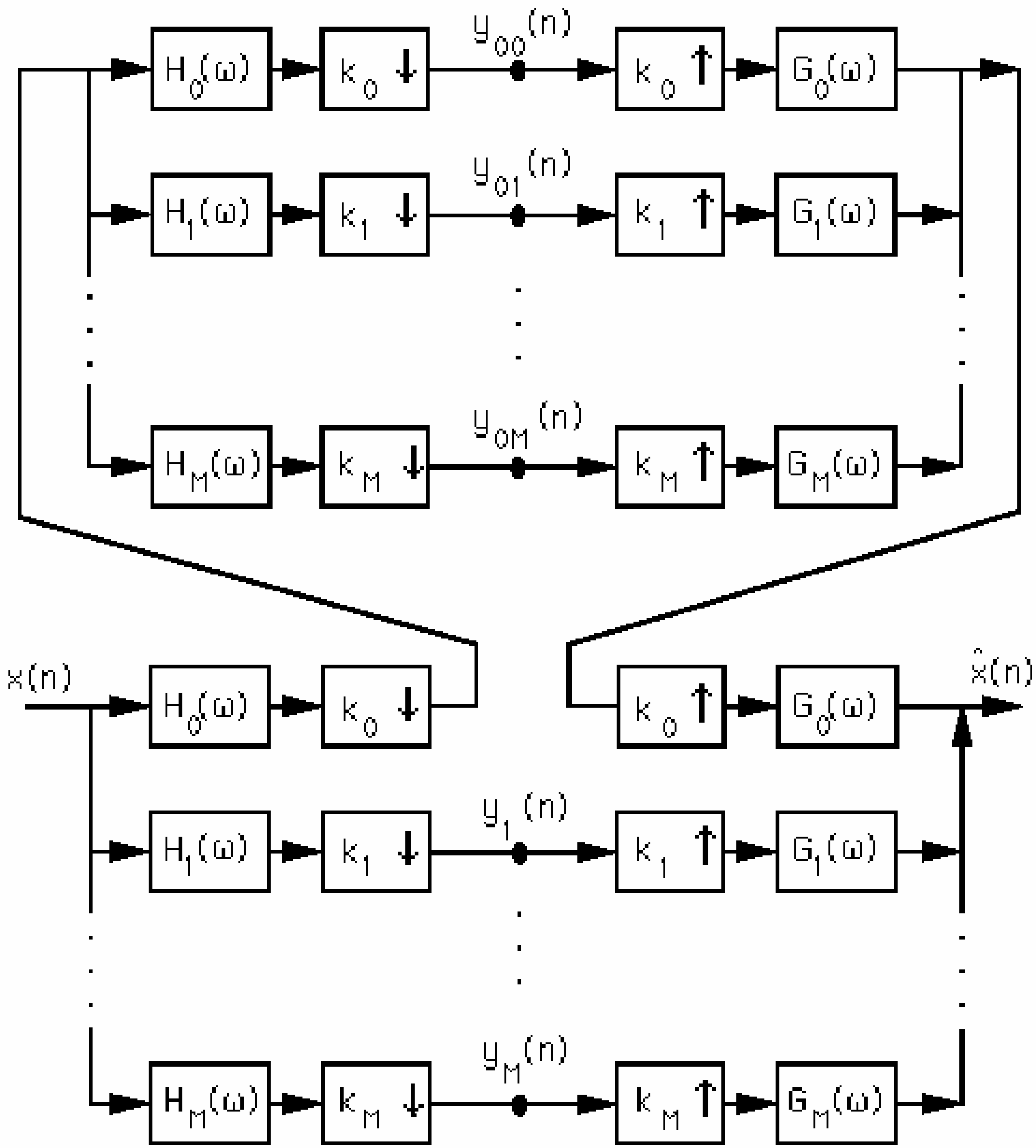


Figure 4.3: A non-uniformly cascaded analysis/synthesis filter bank.

Simoncelli and Adelson, in “Subband coding”, Kluwer, 1990.

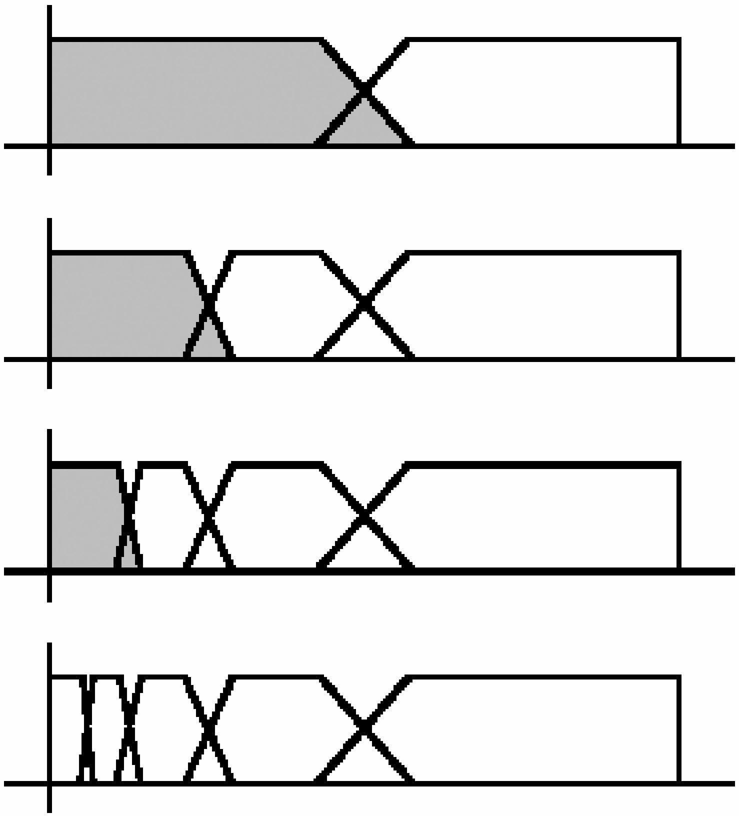


Figure 4.4: Octave band splitting produced by a four-level pyramid cascade of a two-band A/S system. The top picture represents the splitting of the two-band A/S system. Each successive picture shows the effect of re-applying the system to the lowpass subband (indicated in grey) of the previous picture. The bottom picture gives the final four-level partition of the frequency domain. All frequency axes cover the range from 0 to π.

Simoncelli and Adelson, in “Subband coding”, Kluwer, 1990.

n	QMF-5	QMF-9	QMF-13
0	0.8593118	0.7973934	0.7737113
1	0.3535534	0.41472545	0.42995453
2	-0.0761025	-0.073386624	-0.057827797
3		-0.060944743	-0.09800052
4		0.02807382	0.039045125
5			0.021651438
6			-0.014556438

Table 4.1: Odd-length QMF kernels. Half of the impulse response sample values are shown for each of the normalized lowpass QMF filters (All filters are symmetric about $n = 0$). The appropriate highpass filters are obtained by delaying by one sample and multiplying with the sequence $(-1)^n$.

Simoncelli and Adelson, in "Subband coding", Kluwer, 1990.

To create 2-d filters, apply the 1-d filters separably in the two spatial dimensions

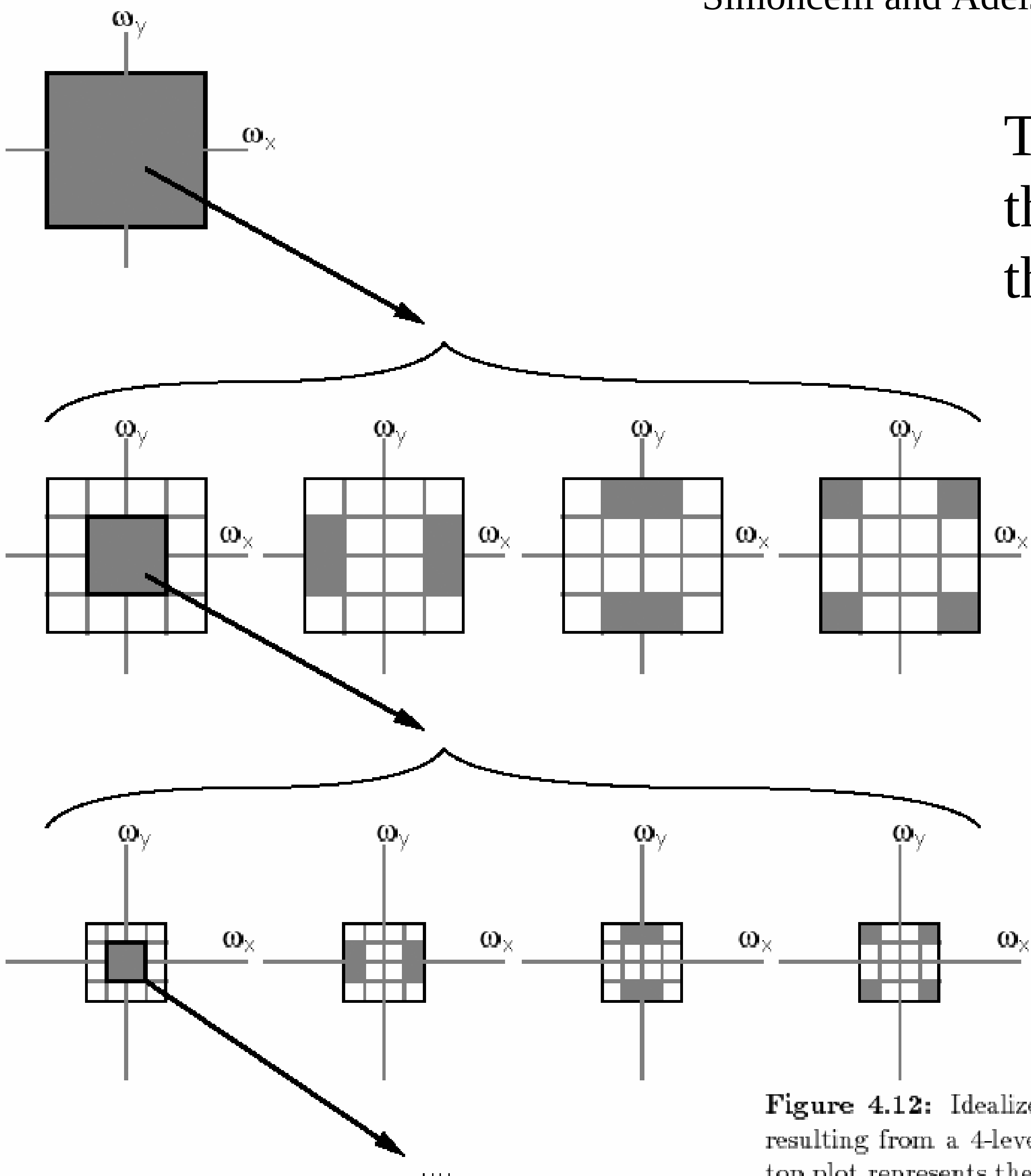
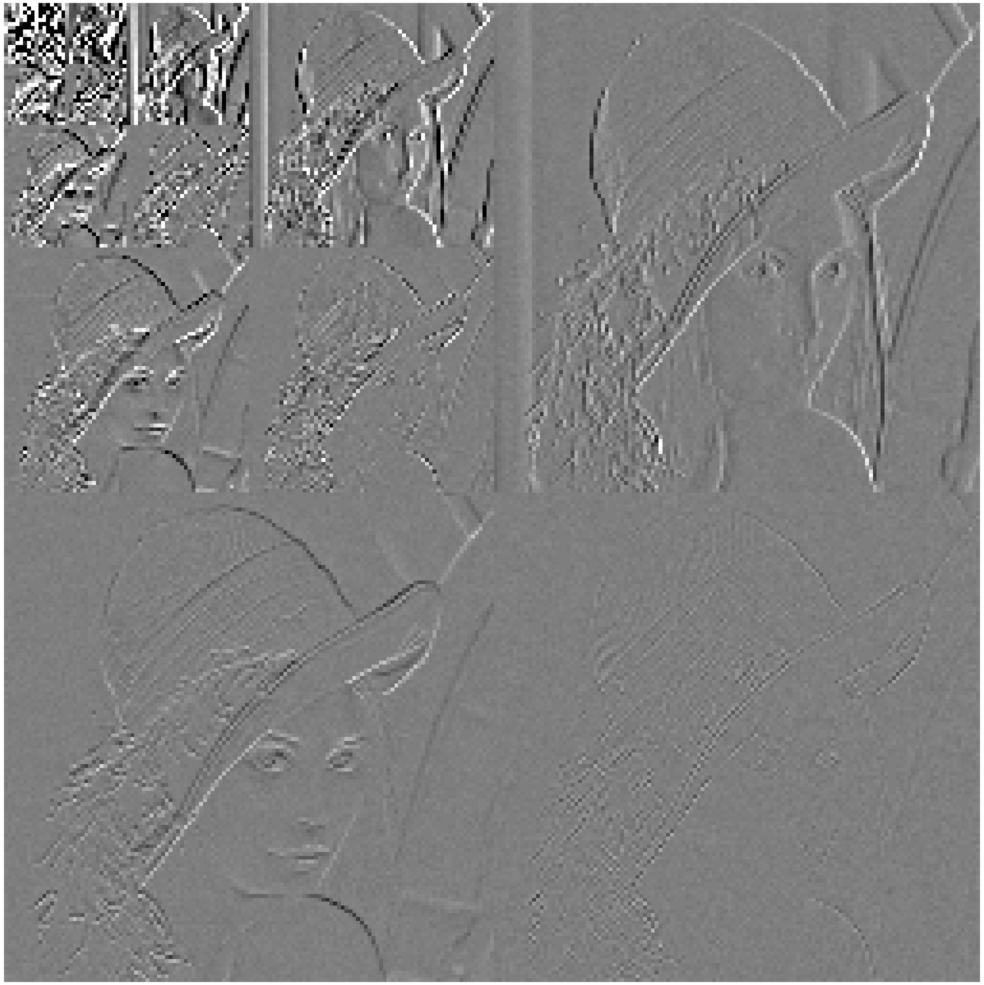


Figure 4.12: Idealized diagram of the partition of the frequency plane resulting from a 4-level pyramid cascade of separable 2-band filters. The top plot represents the frequency spectrum of the original image, with axes ranging from $-\pi$ to π. This is divided into four subbands at the next level. On each subsequent level, the lowpass subband (outlined in bold) is subdivided further.

Wavelet/QMF representation



Good and bad features of wavelet/QMF filters

- Bad:
  - Aliased subbands
  - Non-oriented diagonal subband
- Good:
  - Not overcomplete (so same number of coefficients as image pixels).
  - Good for image compression (JPEG 2000)

Image pyramids

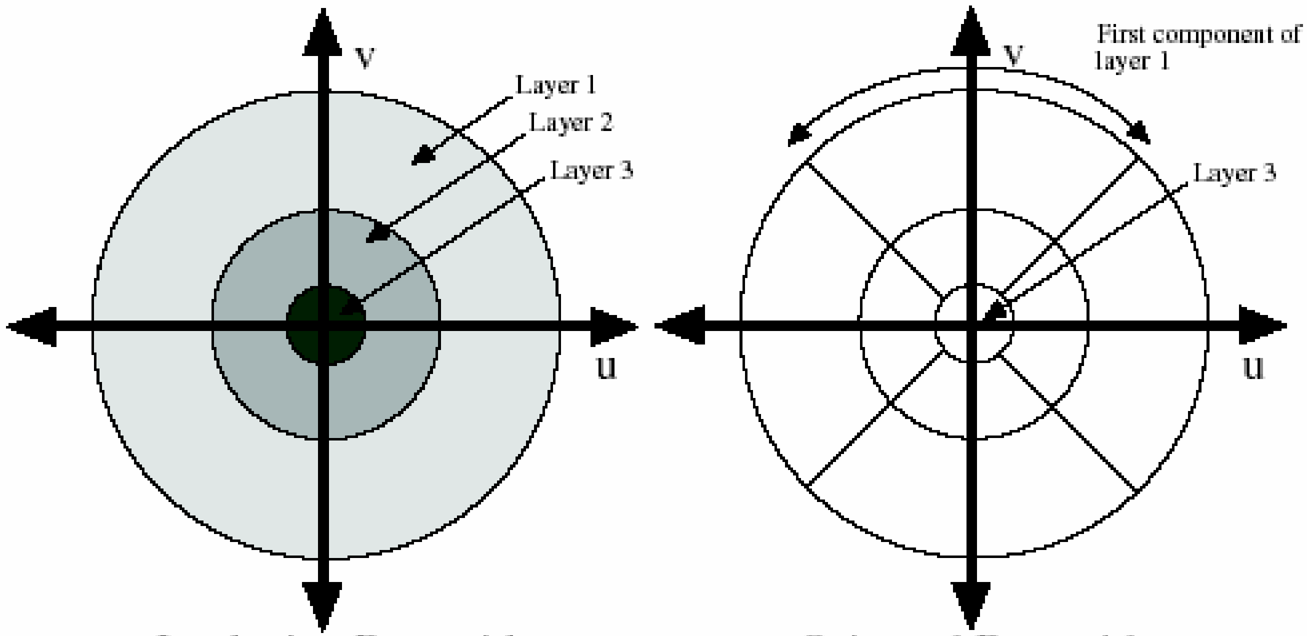
- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid

Steerable pyramids

- Good:
  - Oriented subbands
  - Non-aliased subbands
  - Steerable filters
- Bad:
  - Overcomplete
  - Have one high frequency residual subband, required in order to form a circular region of analysis in frequency from a square region of support in frequency.

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



Laplacian Pyramid Oriented Pyramid

	Laplacian Pyramid	Dyadic QMF/Wavelet	Steerable Pyramid
self-inverting (tight frame)	no	yes	yes
overcompleteness	$4/3$	1	$4k/3$
aliasing in subbands	perhaps	yes	no
rotated orientation bands	no	only on hex lattice [9]	yes

Table 1: Properties of the Steerable Pyramid relative to two other well-known multi-scale representations.

http://www.cns.nyu.edu/ftp/eero/simoncelli95b.pdf

Simoncelli and Freeman, ICIP 1995

But we need to get rid of the corner regions before starting the recursive circular filtering

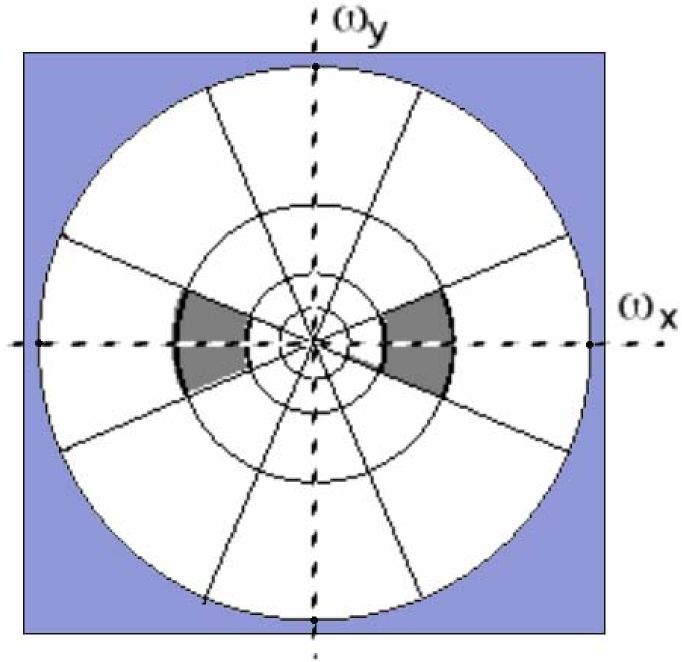
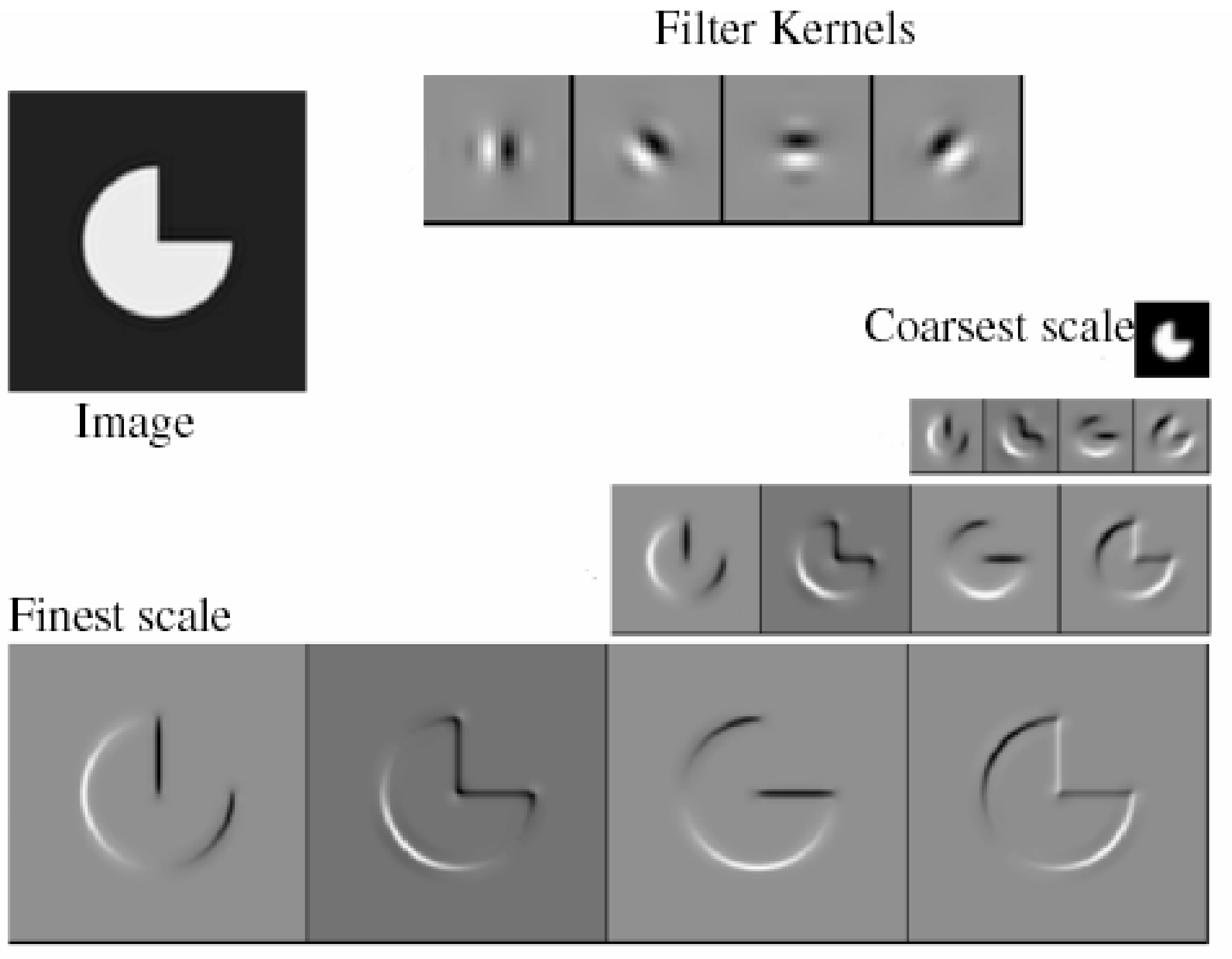


Figure 1. Idealized illustration of the spectral decomposition performed by a steerable pyramid with $k = 4$. Frequency axes range from $-\pi$ to π. The basis functions are related by translations, dilations and *rotations* (except for the initial highpass subband and the final lowpass subband). For example, the shaded region corresponds to the spectral support of a single (vertically-oriented) subband.

http://www.cns.nyu.edu/ftp/eero/simoncelli95b.pdf

Simoncelli and Freeman, ICIP 1995



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

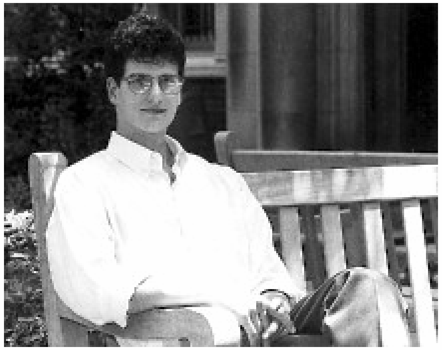
Matlab resources for pyramids (with tutorial)

http://www.cns.nyu.edu/~eero/software.html

Eero P. Simoncelli

Associate Investigator,
Howard Hughes Medical Institute

Associate Professor,
Neural Science and Mathematics,
New York University



Matlab resources for pyramids (with tutorial)

http://www.cns.nyu.edu/~eero/software.html

Laboratory for Computational Vision

Home | People | Research | Publications | Software

Publicly Available Software Packages

- Texture Analysis/Synthesis - Matlab code is available for analyzing and synthesizing visual textures. README | Contents | ChangeLog | Source code (UNIX/PC, gzip'ed tar file)
- EPWIC - Embedded Progressive Wavelet Image Coder. C source code available.
- **matlabPyrTools** - Matlab source code for multi-scale image processing. Includes tools for building and manipulating Laplacian pyramids, QMF/Wavelets, and steerable pyramids. Data structures are compatible with the Matlab wavelet toolbox, but the convolution code (in C) is faster and has many boundary-handling options. README, Contents, Modification list, UNIX/PC source or Macintosh source.
- The Steerable Pyramid, an (approximately) translation- and rotation-invariant multi-scale image decomposition. MatLab (see above) and C implementations are available.
- Computational Models of cortical neurons. Macintosh program available.
- EPIC - Efficient Pyramid (Wavelet) Image Coder. C source code available.
- OBVIUS [Object-Based Vision & Image Understanding System]: README / ChangeLog / Doc (225k) / Source Code (2.25M).
- CL-SHELL [Gnu Emacs <-> Common Lisp Interface]: README / Change Log / Source Code (119k).

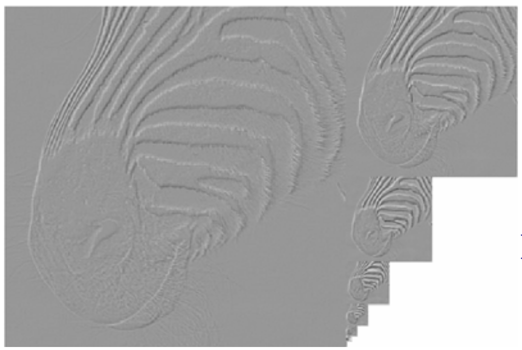
- Summary of pyramid representations

Image pyramids

- Gaussian

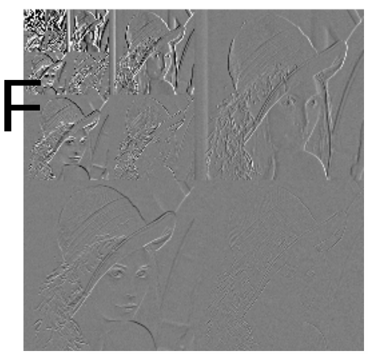
Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- Laplacian



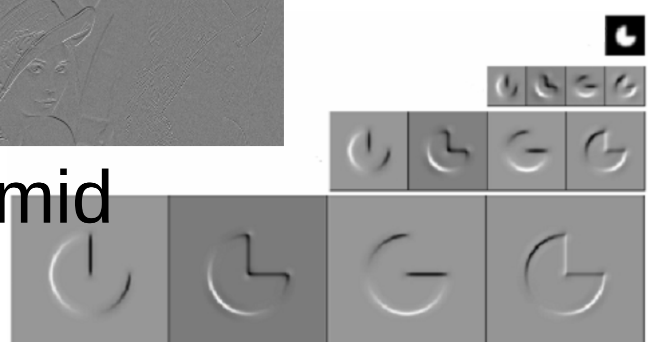
Shows the information added in Gaussian pyramid at each spatial scale. Useful for noise reduction & coding.

- Wavelet/QMF



Bandpassed representation, complete, but with aliasing and some non-oriented subbands.

- Steerable pyramid



Shows components at each scale and orientation separately. Non-aliased subbands. Good for texture and feature analysis.

Linear image transformations

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

transformed image

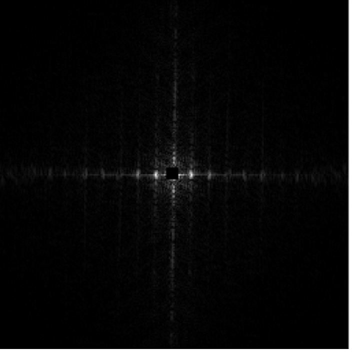
$$\vec{F} = U\vec{f}$$

Vectorized image

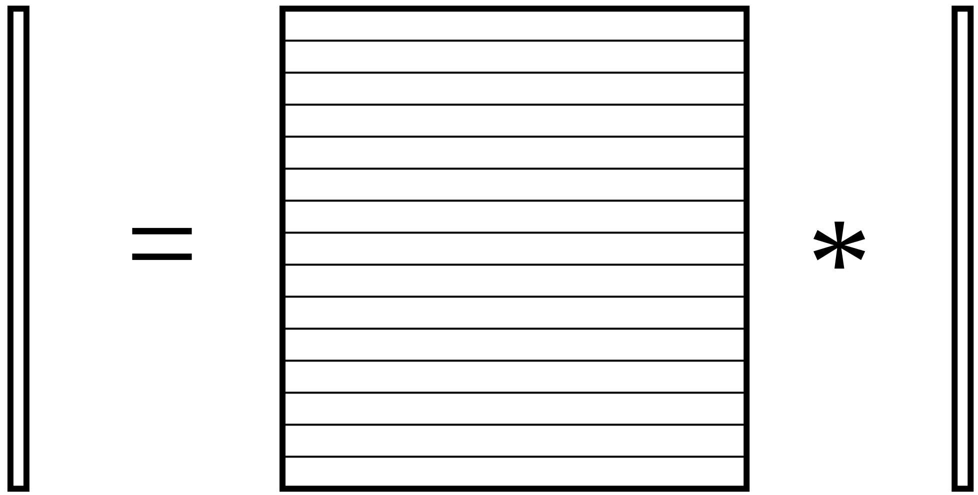
Fourier transform, or
Wavelet transform, or
Steerable pyramid transform

Schematic pictures of each matrix transform

- Shown for 1-d images
- The matrices for 2-d images are the same idea, but more complicated, to account for vertical, as well as horizontal, neighbor relationships.



Fourier transform

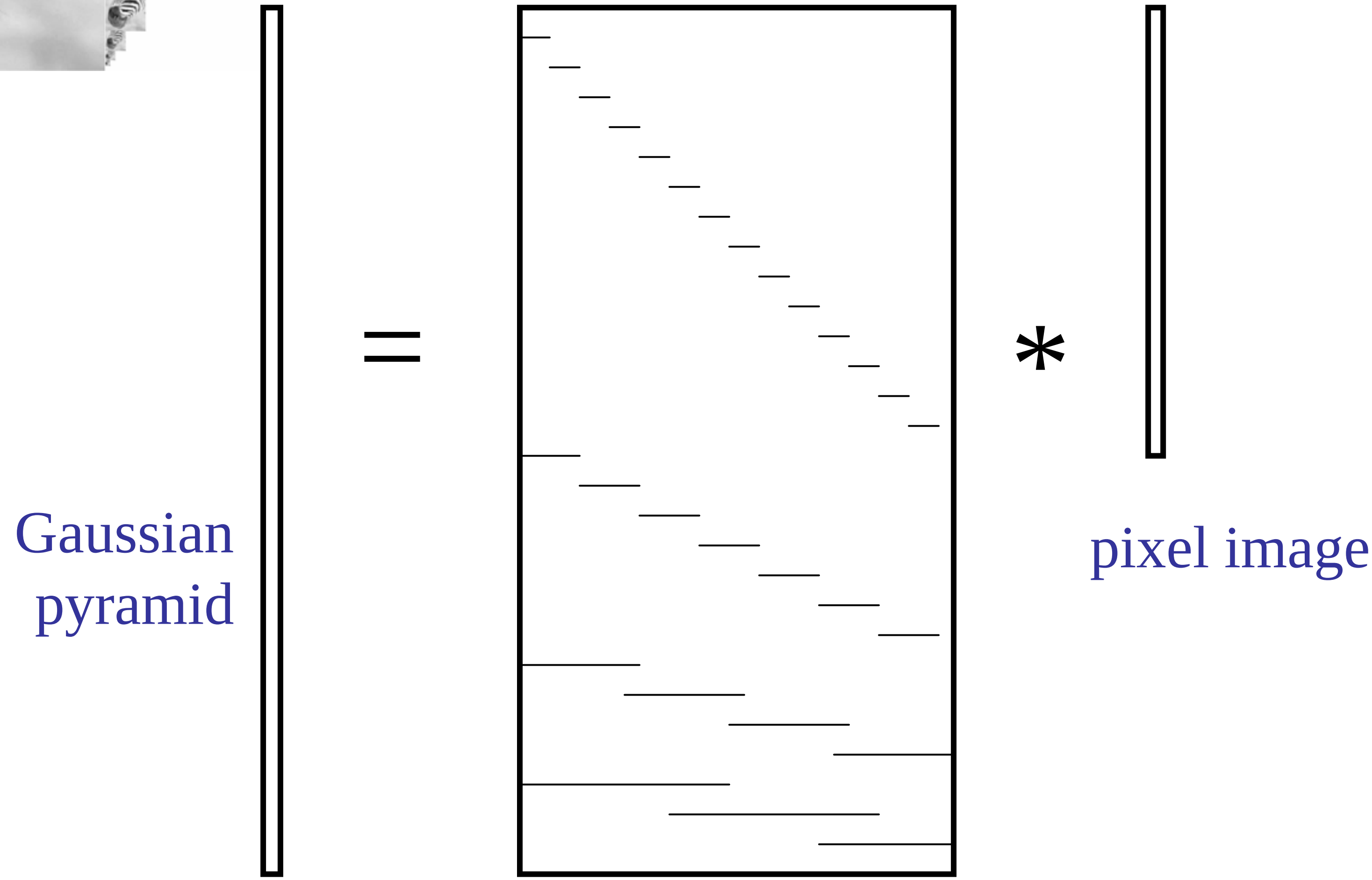


Fourier transform

Fourier bases are global: each transform coefficient depends on all pixel locations.

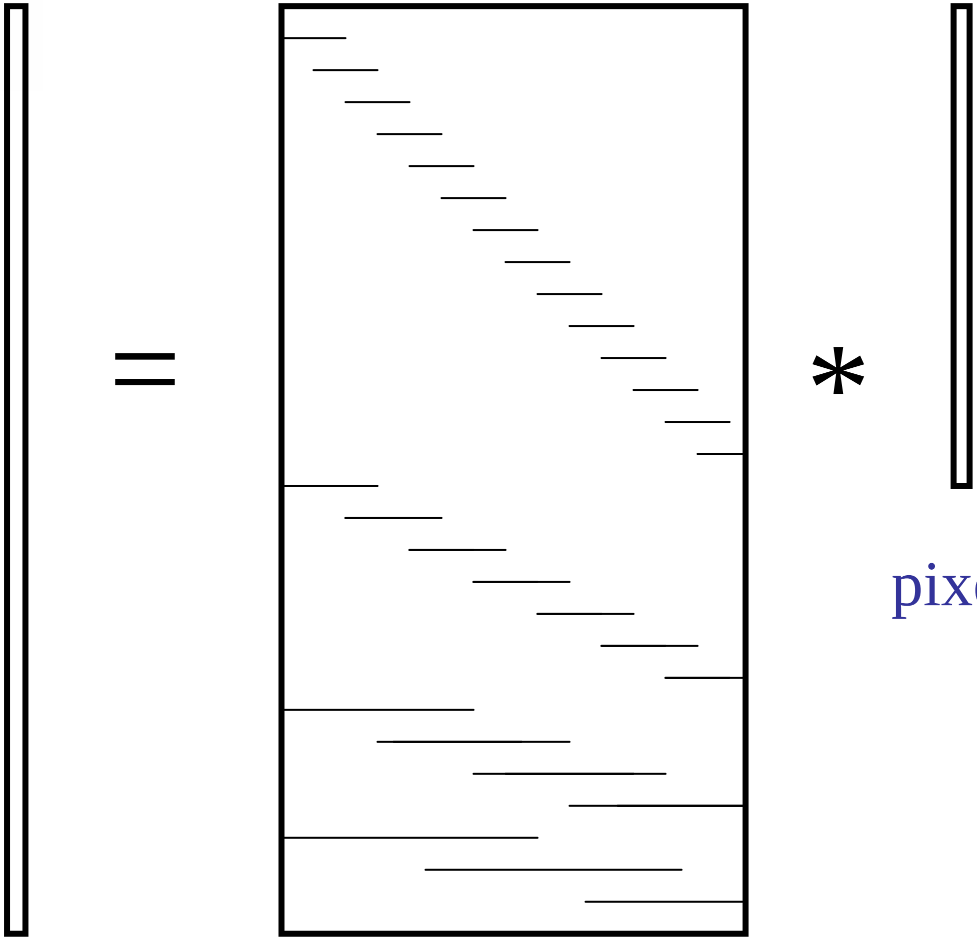
pixel domain image

Gaussian pyramid



Overcomplete representation.
Low-pass filters, sampled appropriately for their blur.

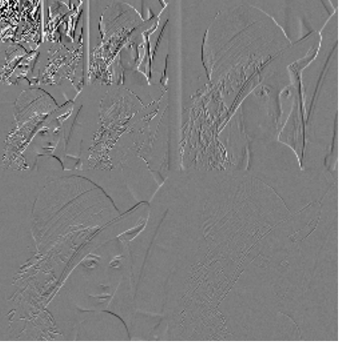
Laplacian pyramid



Laplacian pyramid

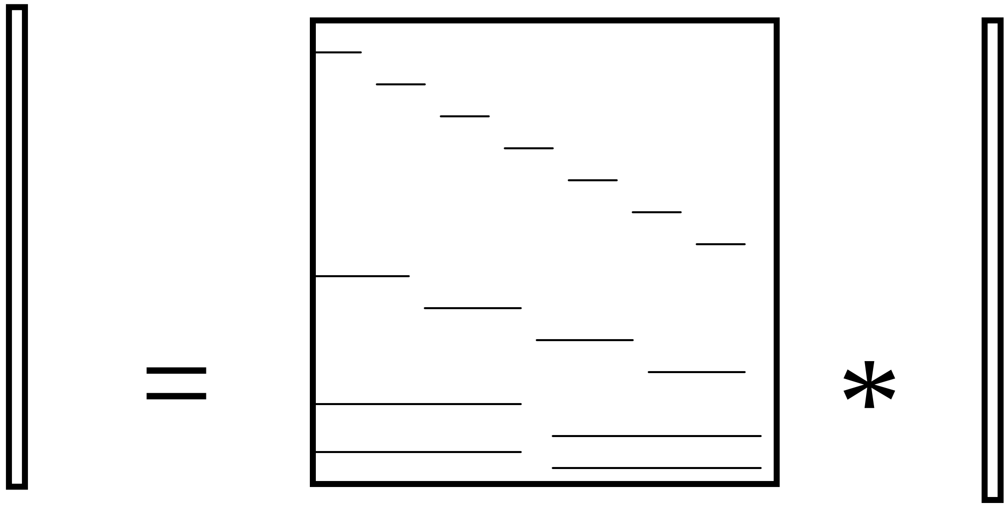
pixel image

Overcomplete representation.
Transformed pixels represent
bandpassed image information.



Wavelet (QMF) transform

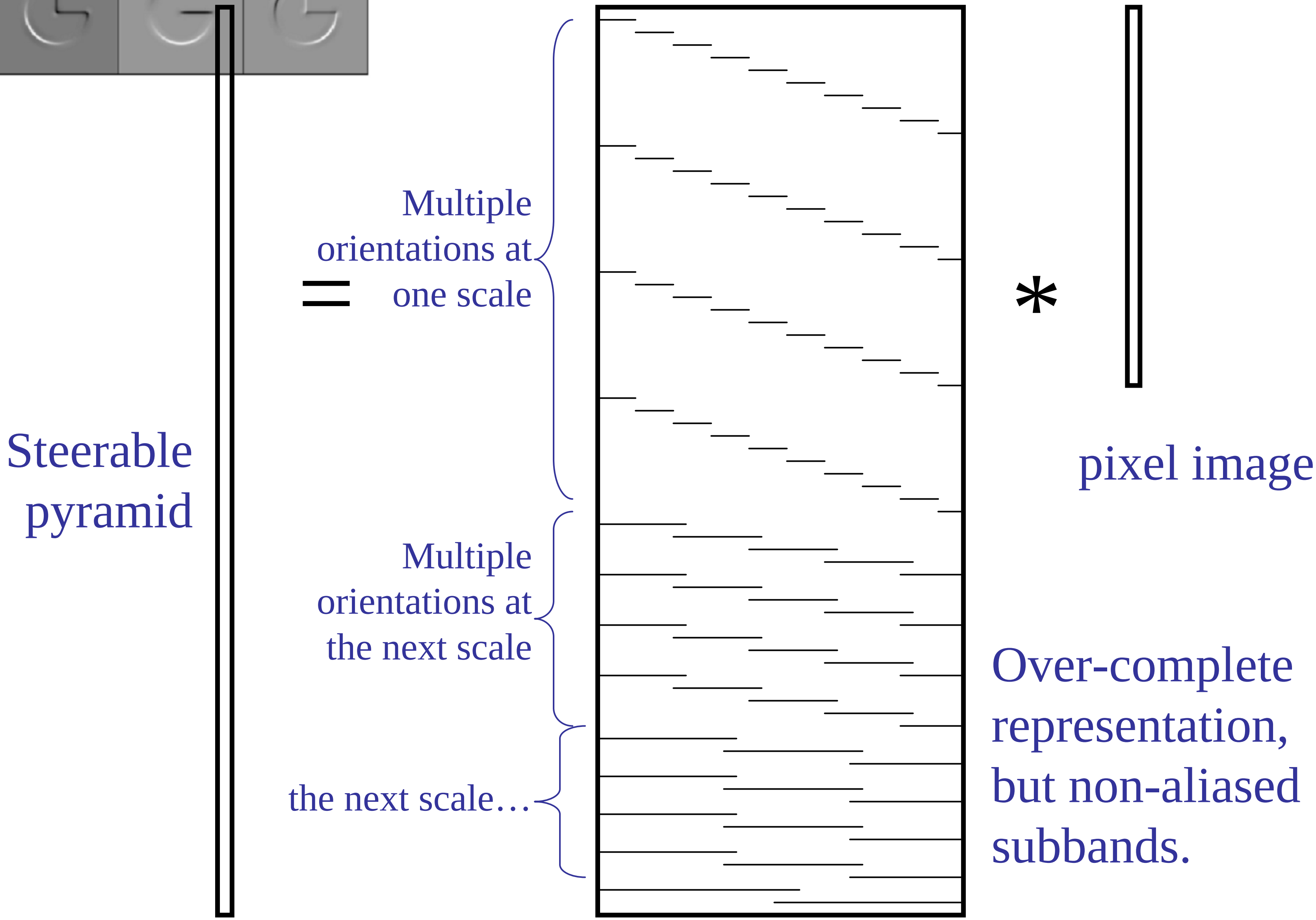
Wavelet pyramid



Ortho-normal transform (like Fourier transform), but with localized basis functions.

pixel image

Steerable pyramid



Matlab resources for pyramids (with tutorial)

http://www.cns.nyu.edu/~eero/software.html

lcv

Laboratory for Computational Vision

Home | People | Research | Publications | Software

Publicly Available Software Packages

- Texture Analysis/Synthesis - Matlab code is available for analyzing and synthesizing visual textures. README | Contents | ChangeLog | Source code (UNIX/PC, gzip'ed tar file)
- EPWIC - Embedded Progressive Wavelet Image Coder. C source code available.
- **matlabPyrTools** - Matlab source code for multi-scale image processing. Includes tools for building and manipulating Laplacian pyramids, QMF/Wavelets, and steerable pyramids. Data structures are compatible with the Matlab wavelet toolbox, but the convolution code (in C) is faster and has many boundary-handling options. README, Contents, Modification list, UNIX/PC source or Macintosh source.
- The Steerable Pyramid, an (approximately) translation- and rotation-invariant multi-scale image decomposition. MatLab (see above) and C implementations are available.
- Computational Models of cortical neurons. Macintosh program available.
- EPIC - Efficient Pyramid (Wavelet) Image Coder. C source code available.
- OBVIUS [Object-Based Vision & Image Understanding System]: README / ChangeLog / Doc (225k) / Source Code (2.25M).
- CL-SHELL [Gnu Emacs <-> Common Lisp Interface]: README / Change Log / Source Code (119k).

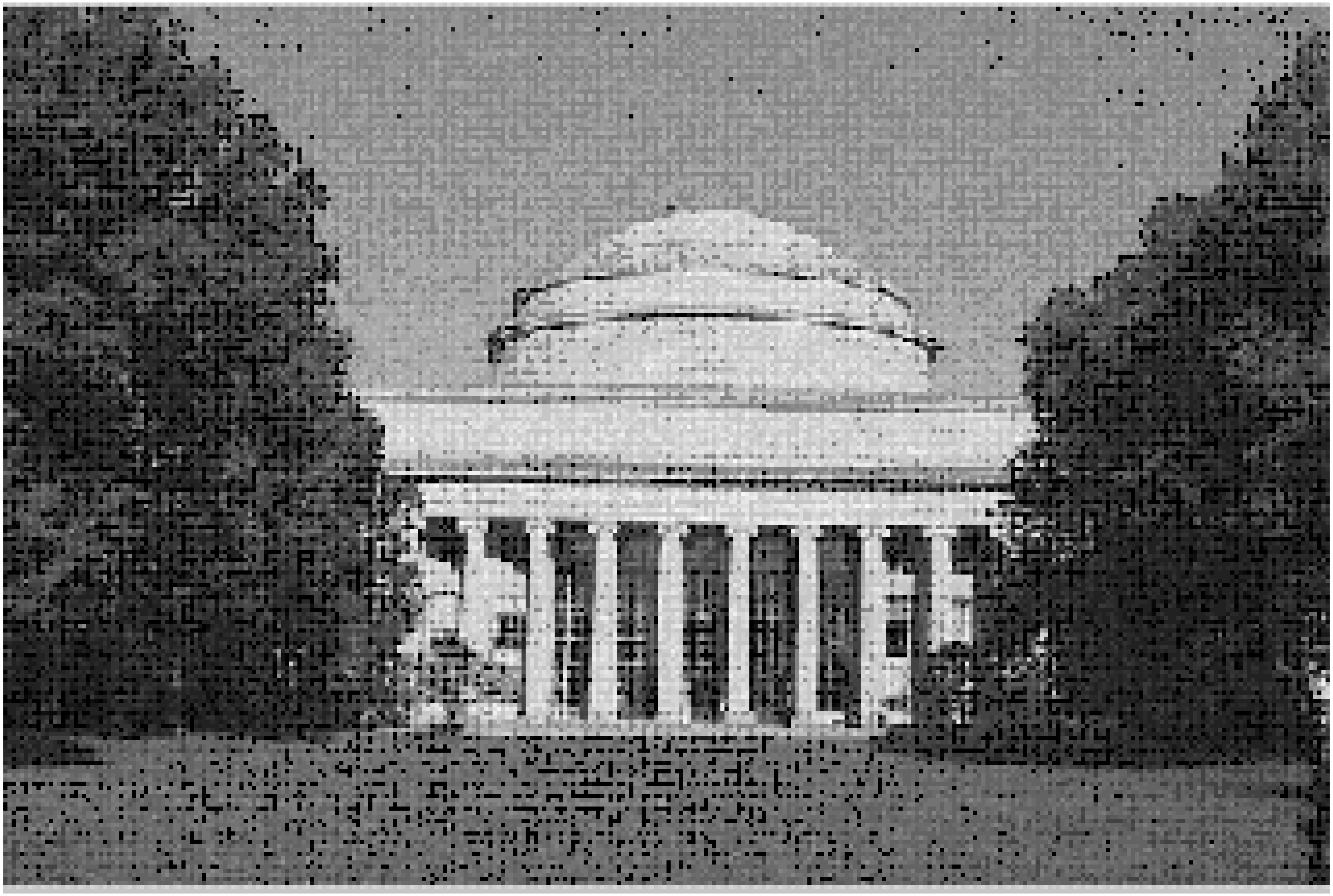
Why use these representations?

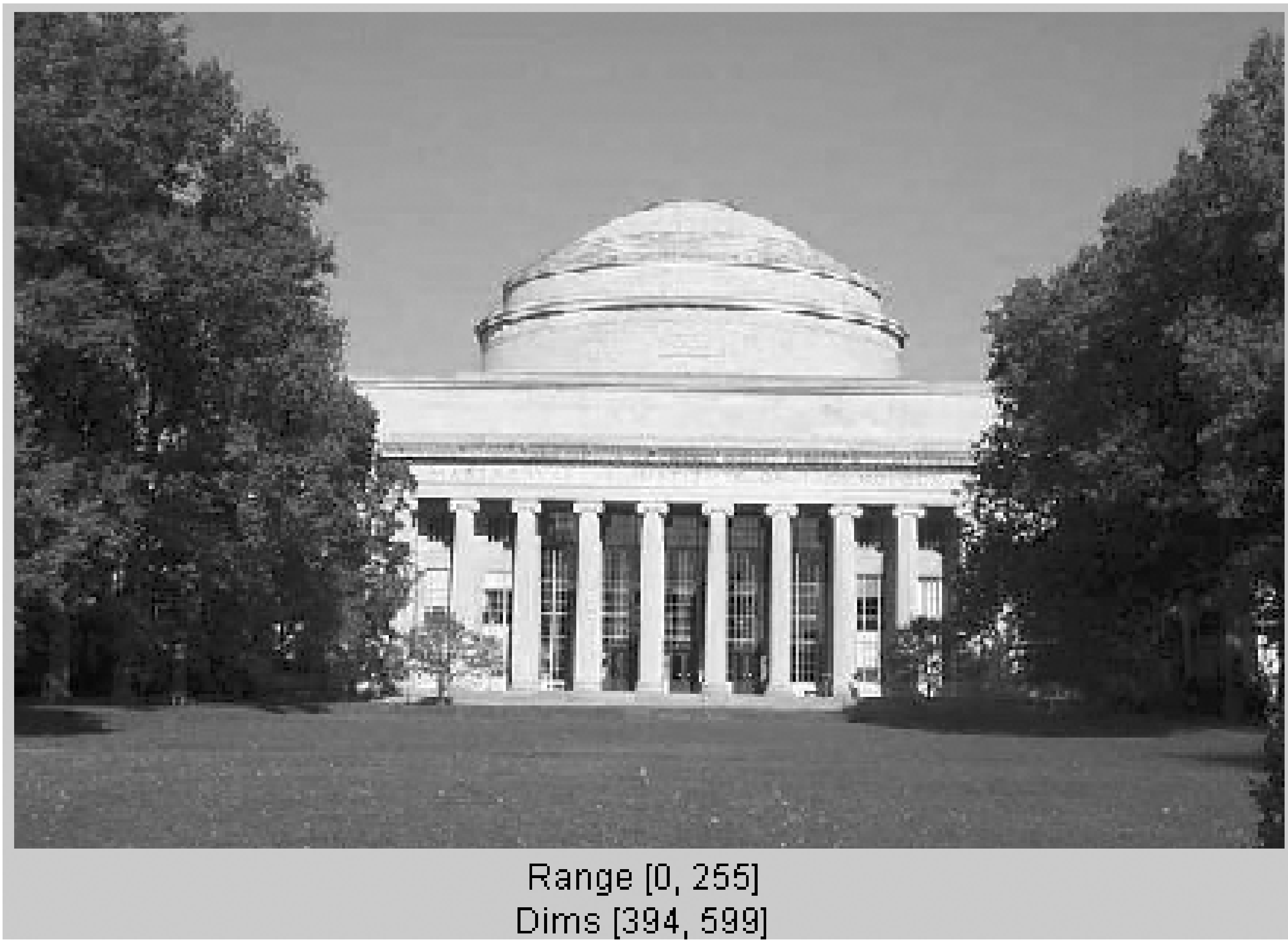
- Handle real-world size variations with a constant-size vision algorithm.
- Remove noise
- Analyze texture
- Recognize objects
- Label image features

end

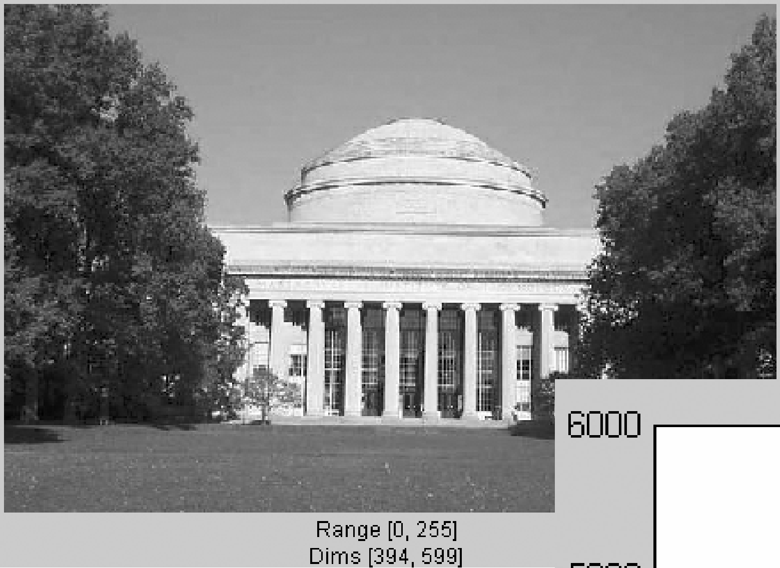
An application of image pyramids: noise removal

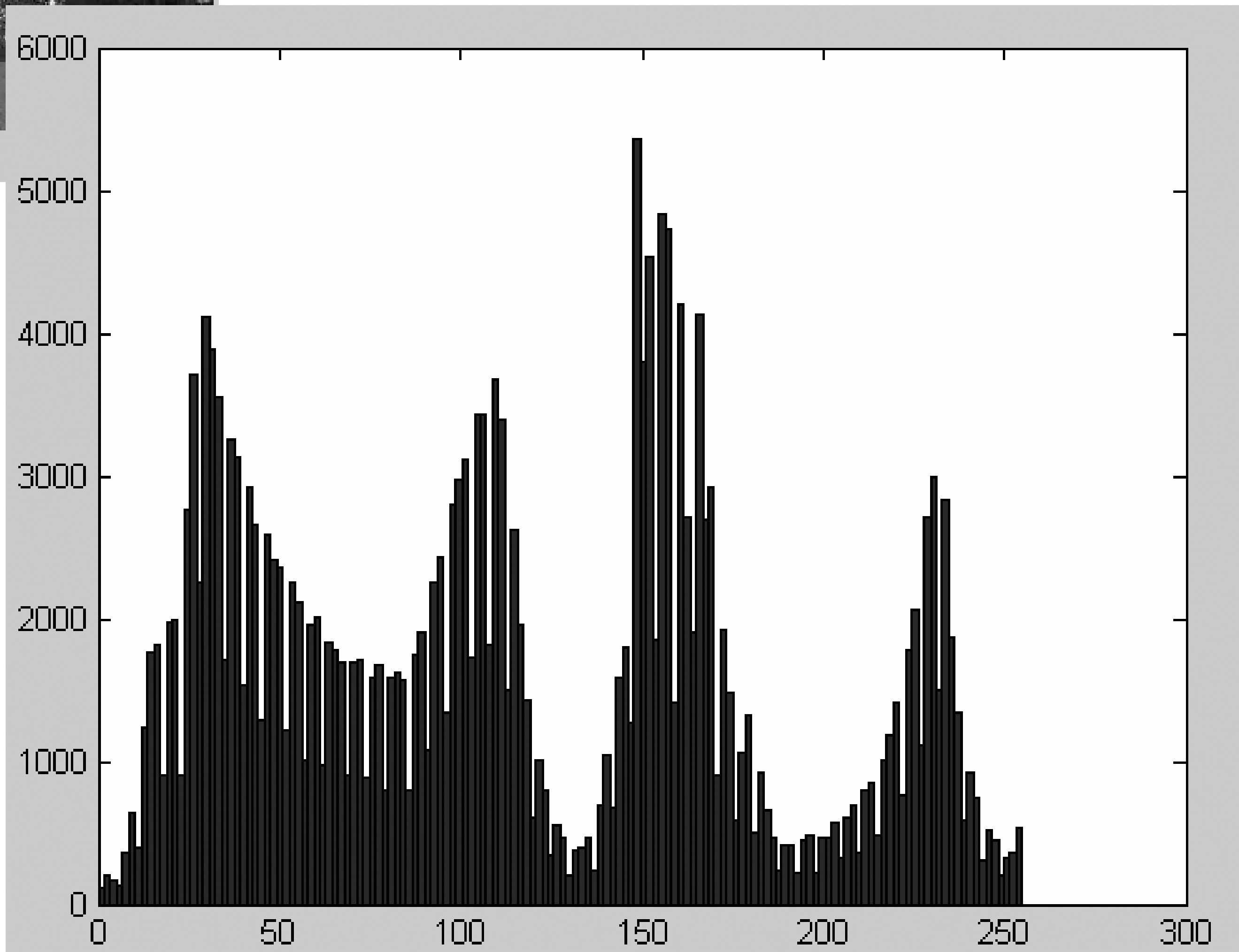
Image statistics (or, mathematically, how can you tell image from noise?)



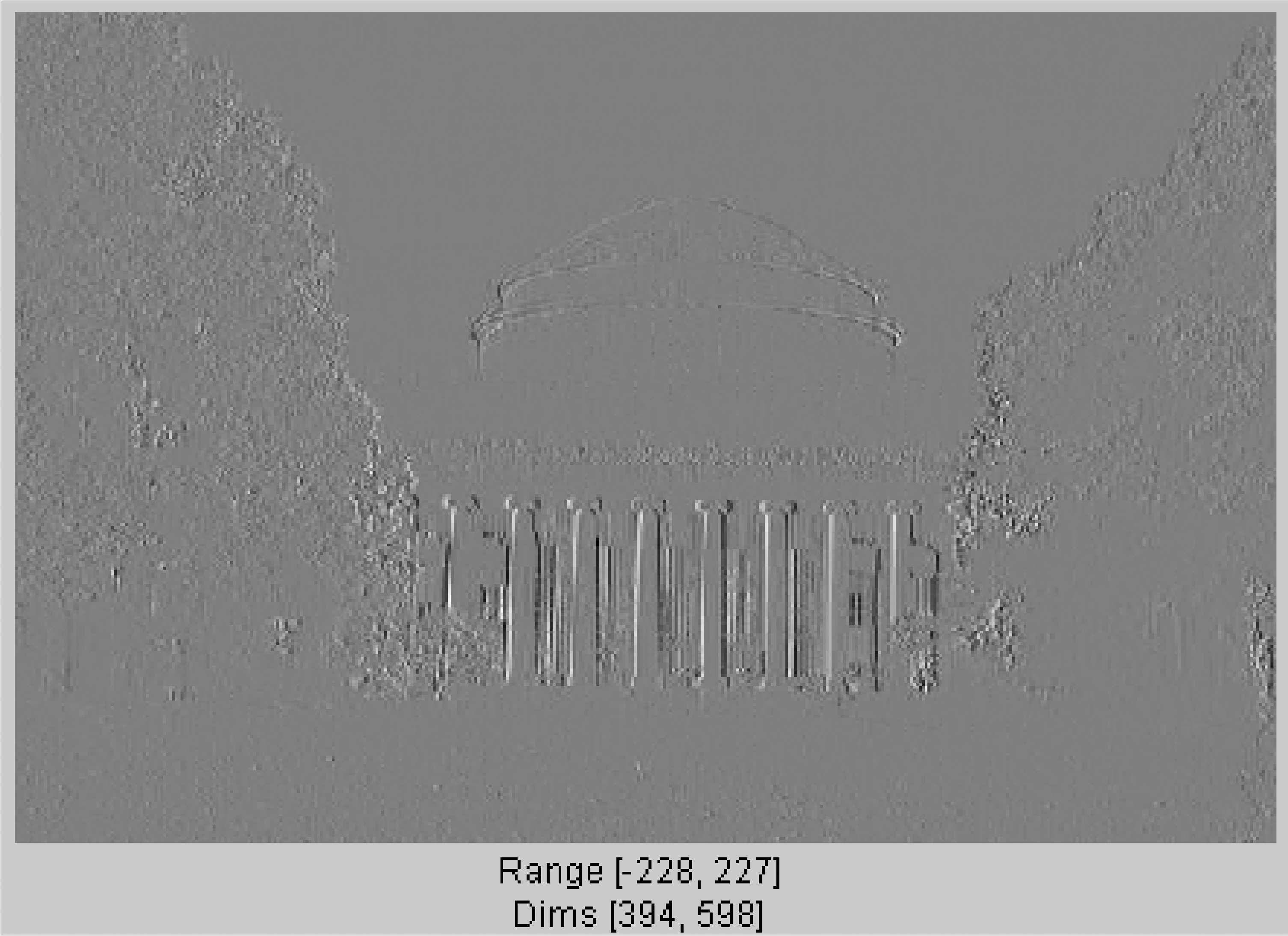

Range [0, 255]
Dims [394, 599]

Pixel representation image histogram

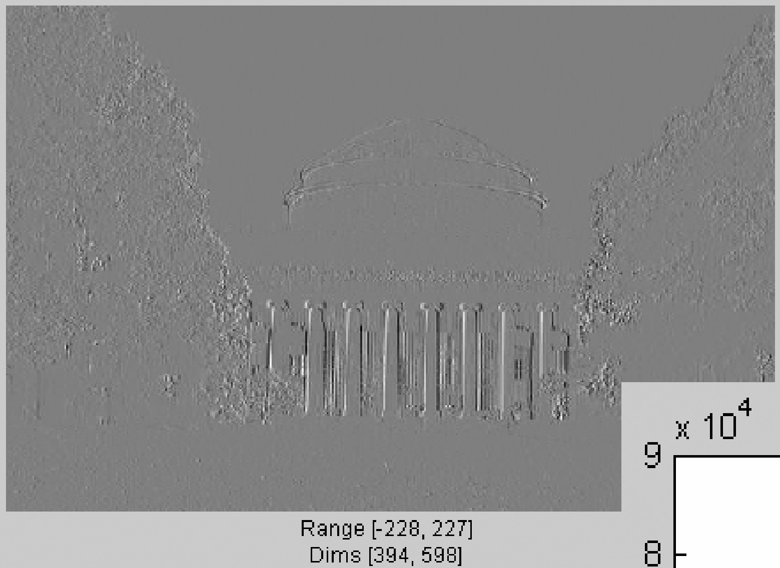


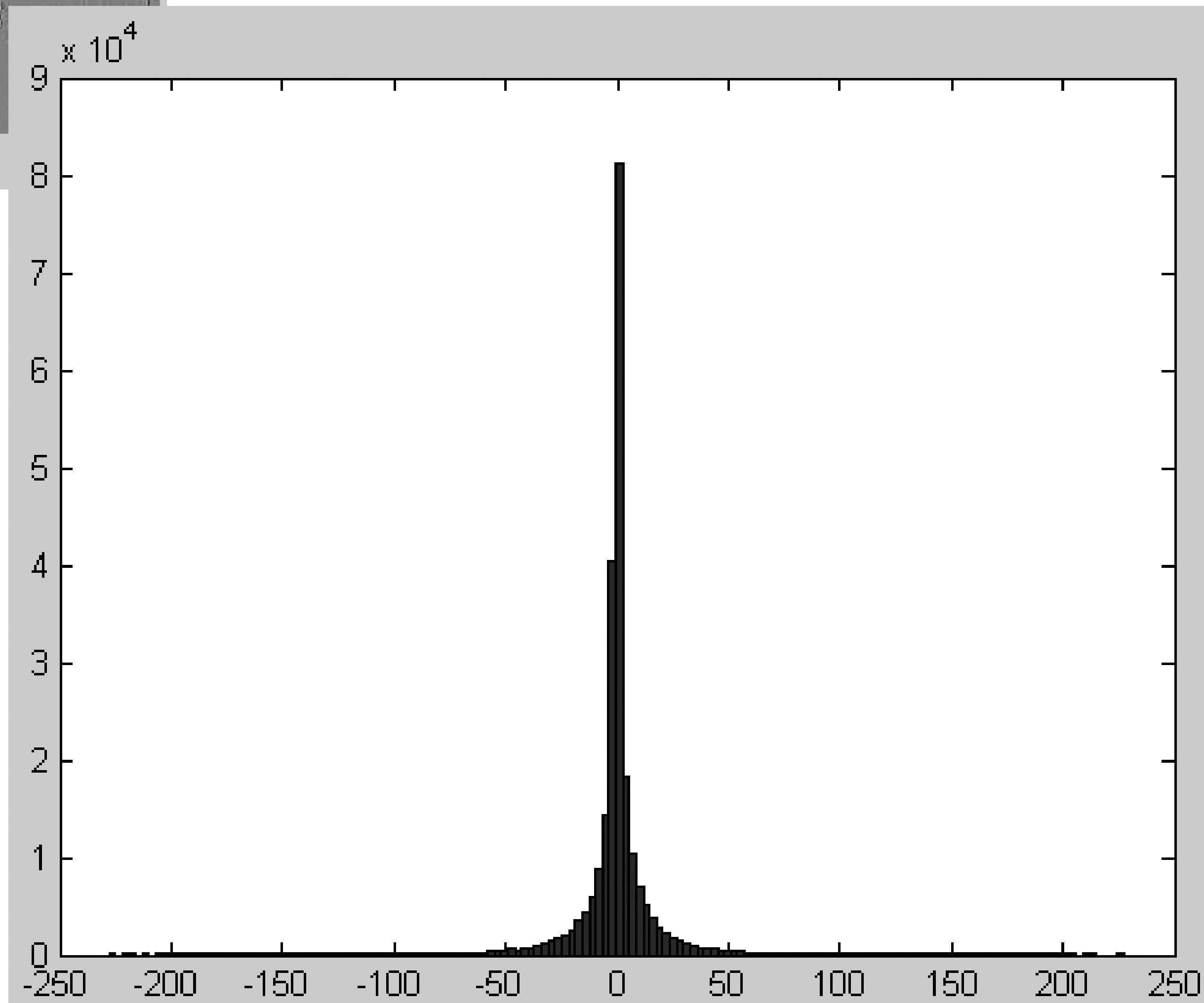


bandpass filtered image

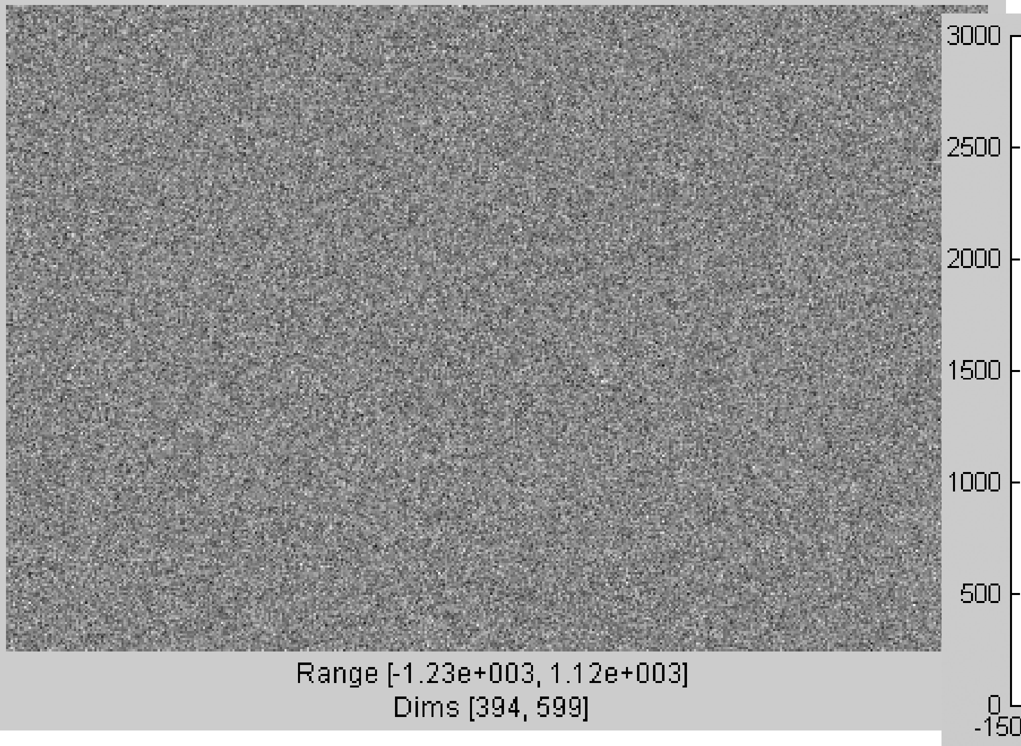


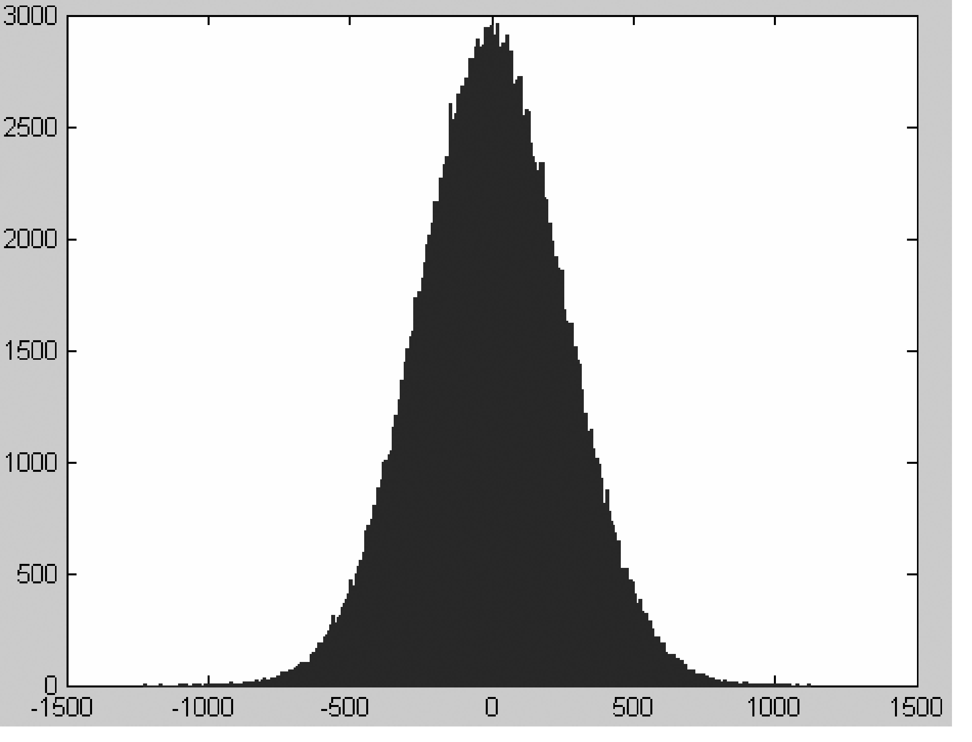
bandpassed representation image histogram



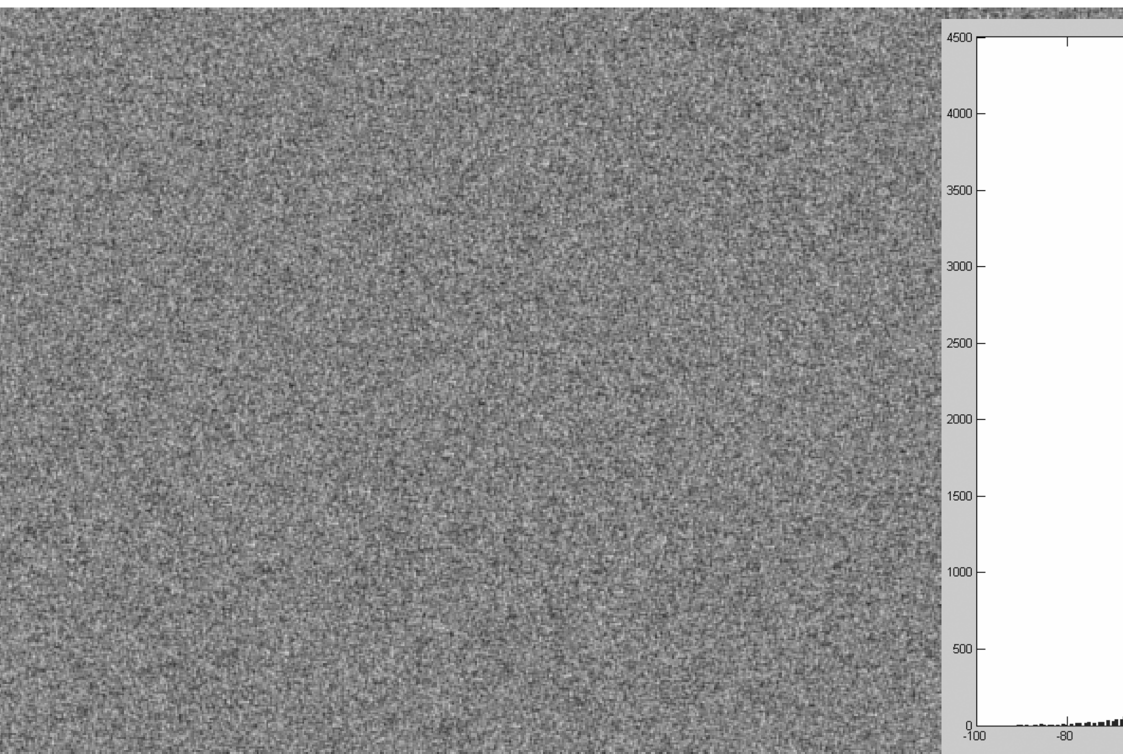


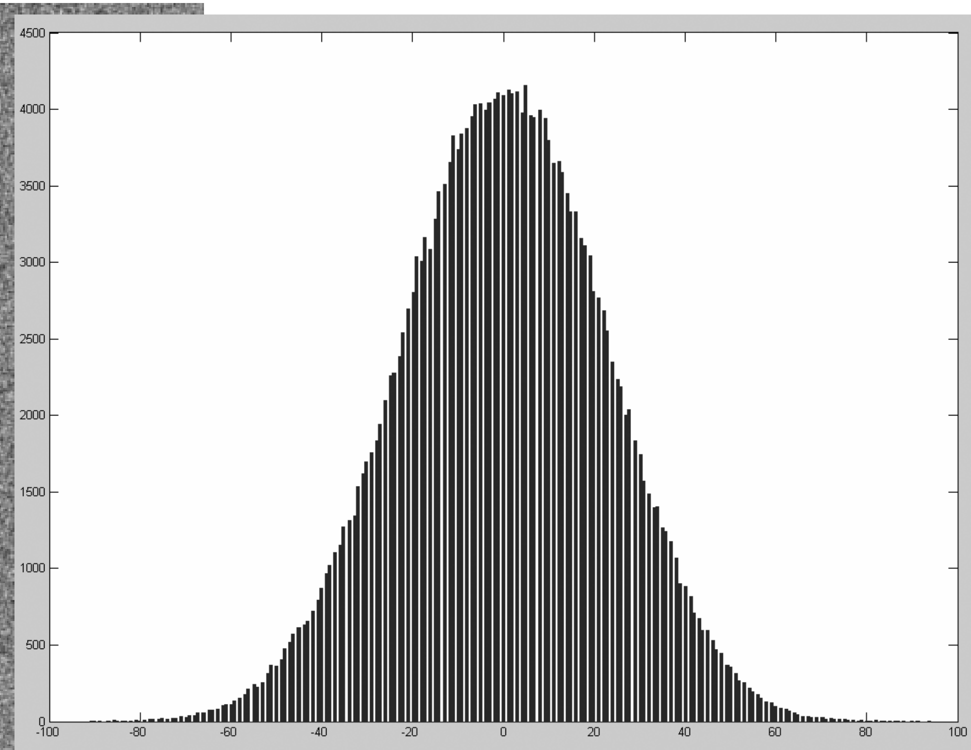
Pixel domain noise image and histogram



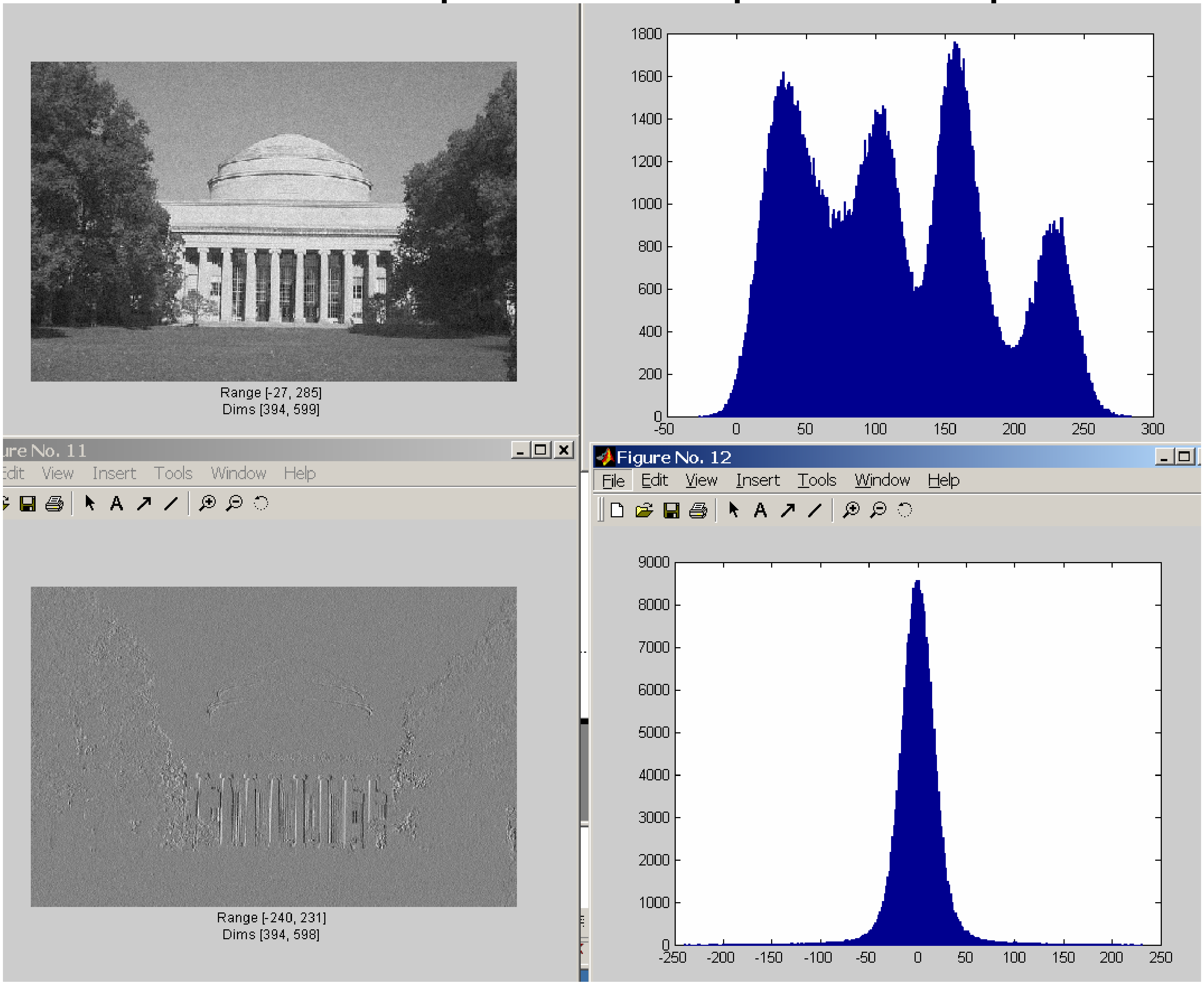


Bandpass domain noise image and histogram





Noise-corrupted full-freq and bandpass images



Bayes theorem

$$P(x, y) = P(x|y)\, P(y)$$

so

$$P(x|y)\, P(y) = P(y|x)\, P(x)$$

and

$$P(x|y) = P(y|x)\, P(x) \,/\, P(y)$$

The parameters you want to estimate

What you observe

Likelihood function

Prior probability

Constant w.r.t. parameters x.

Bayesian MAP estimator for clean bandpass coefficient values

Let x = bandpassed image value before adding noise.
Let y = noise-corrupted observation.

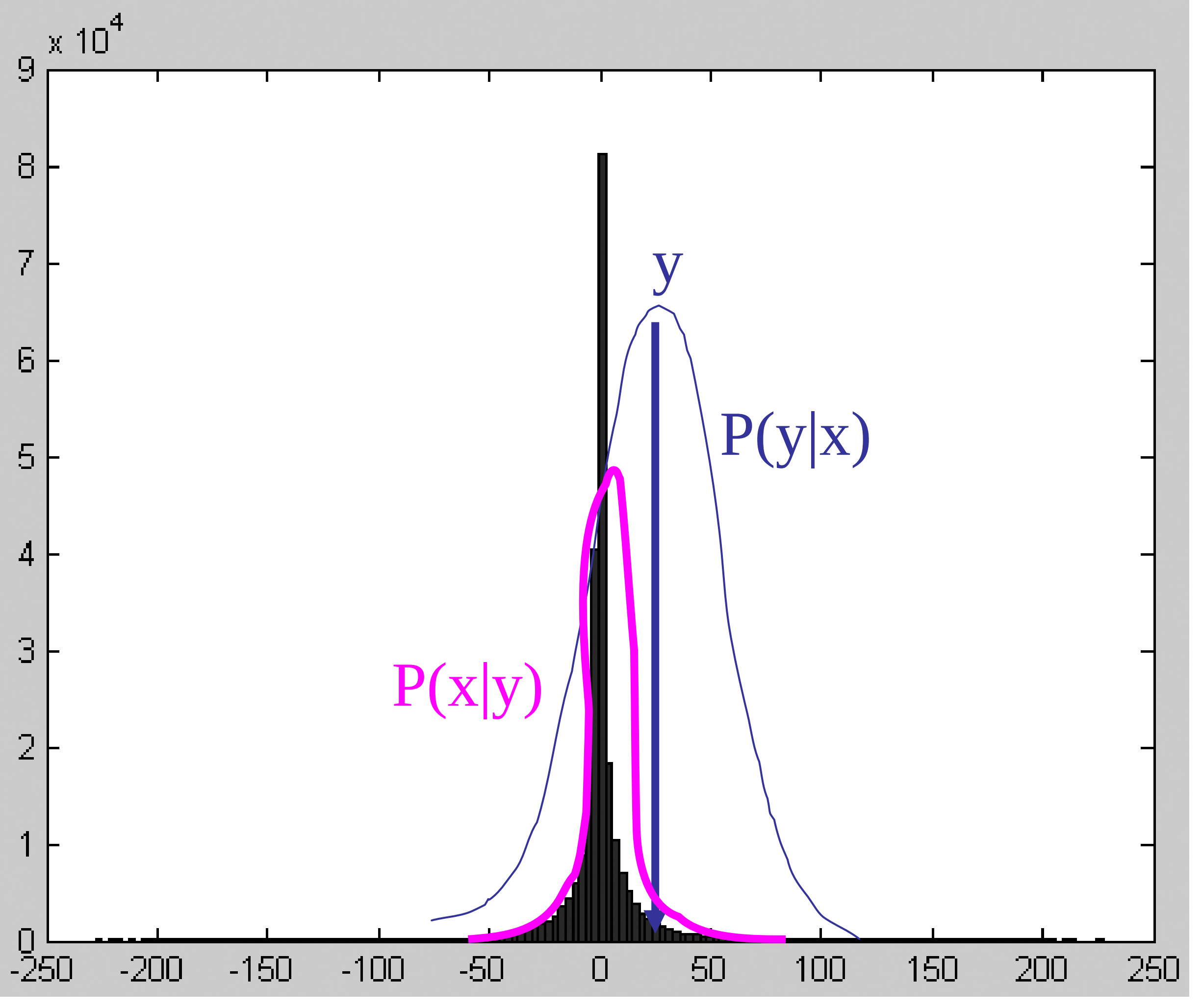
By Bayes theorem

P(x|y) = k P(y|x) P(x)

P(x)

P(y|x)

P(x|y)



Bayesian MAP estimator

Let x = bandpassed image value before adding noise.

Let y = noise-corrupted observation.

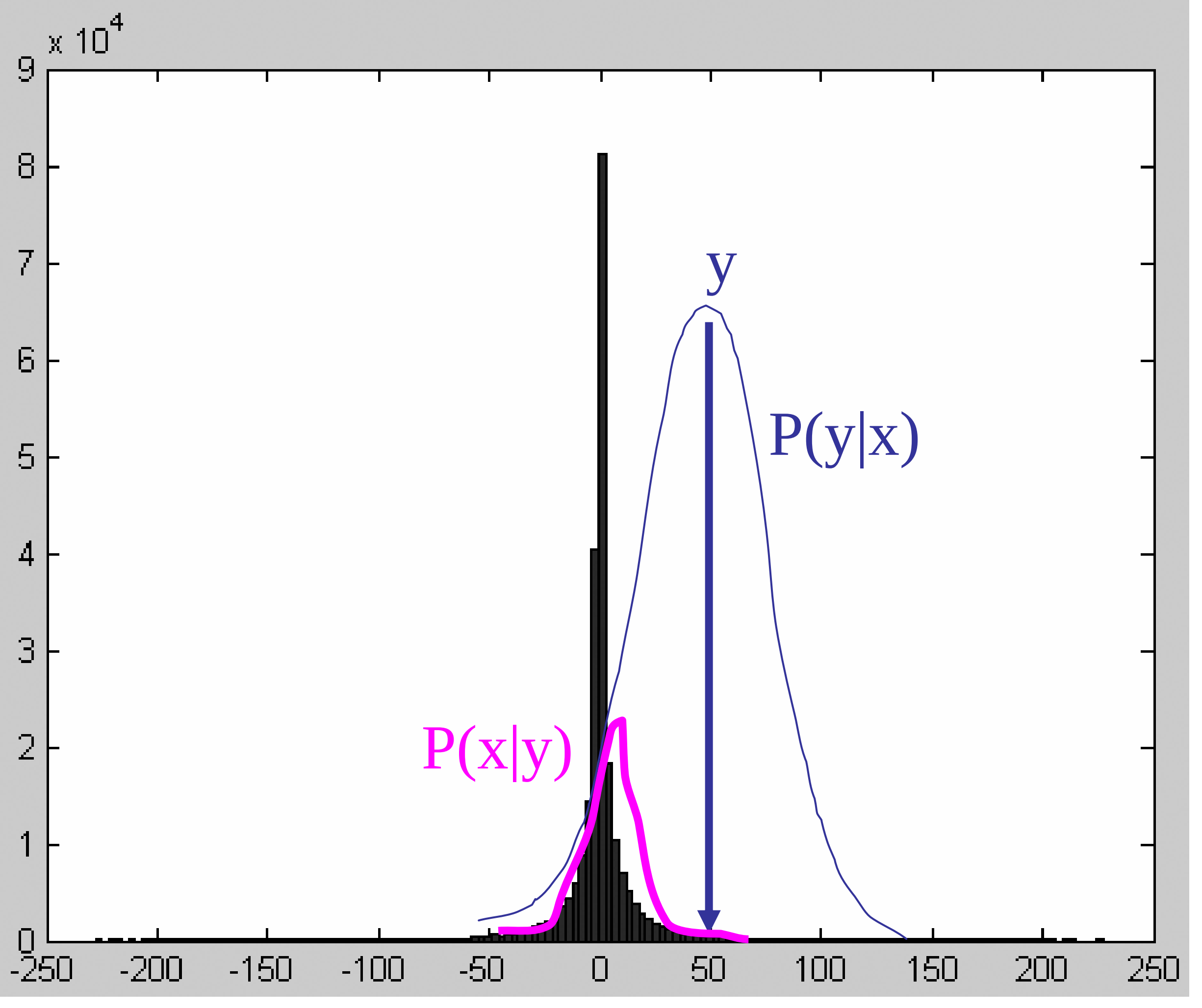
By Bayes theorem

P(x|y) = k P(y|x) P(x)

P(x)

P(y|x)

P(x|y)



Bayesian MAP estimator

Let x = bandpassed image value before adding noise.
Let y = noise-corrupted observation.

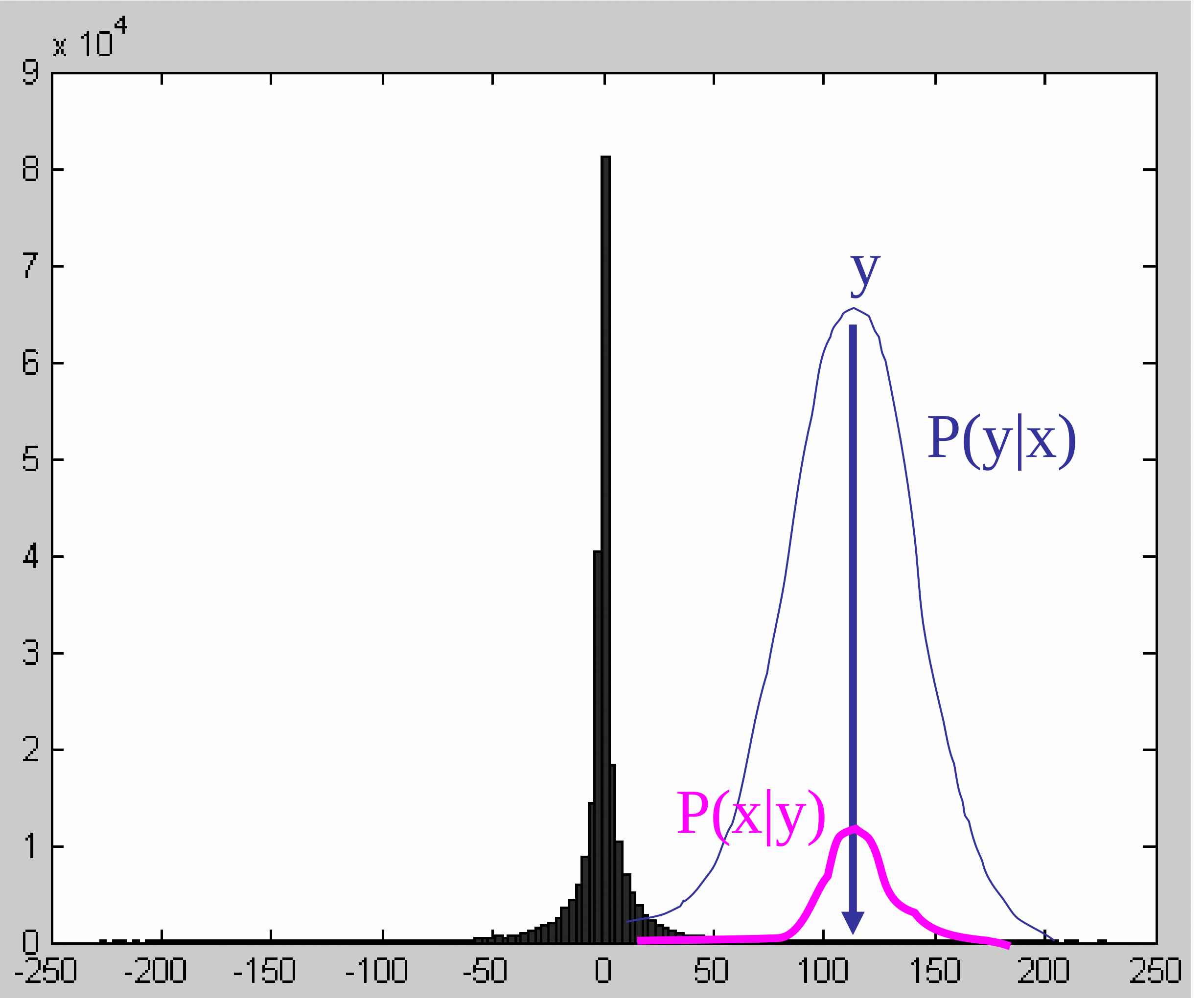
By Bayes theorem

P(x|y) = k P(y|x) P(x)

P(x)

P(y|x)

P(x|y)



MAP estimate, $\hat{x}$, as function of observed coefficient value, y

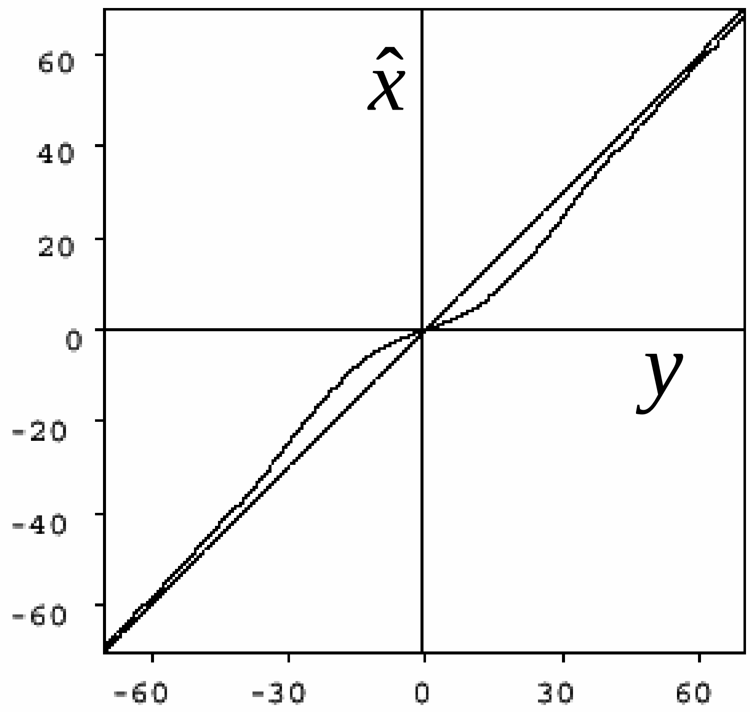


Figure 2: Bayesian estimator (symmetrized) for the signal and noise histograms shown in figure 1. Superimposed on the plot is a straight line indicating the identity function.

http://www-bcs.mit.edu/people/adelson/pub_pdfs/simoncelli_noise.pdf **Simoncelli and Adelson, Noise Removal via Bayesian Wavelet Coring**

Noise removal results

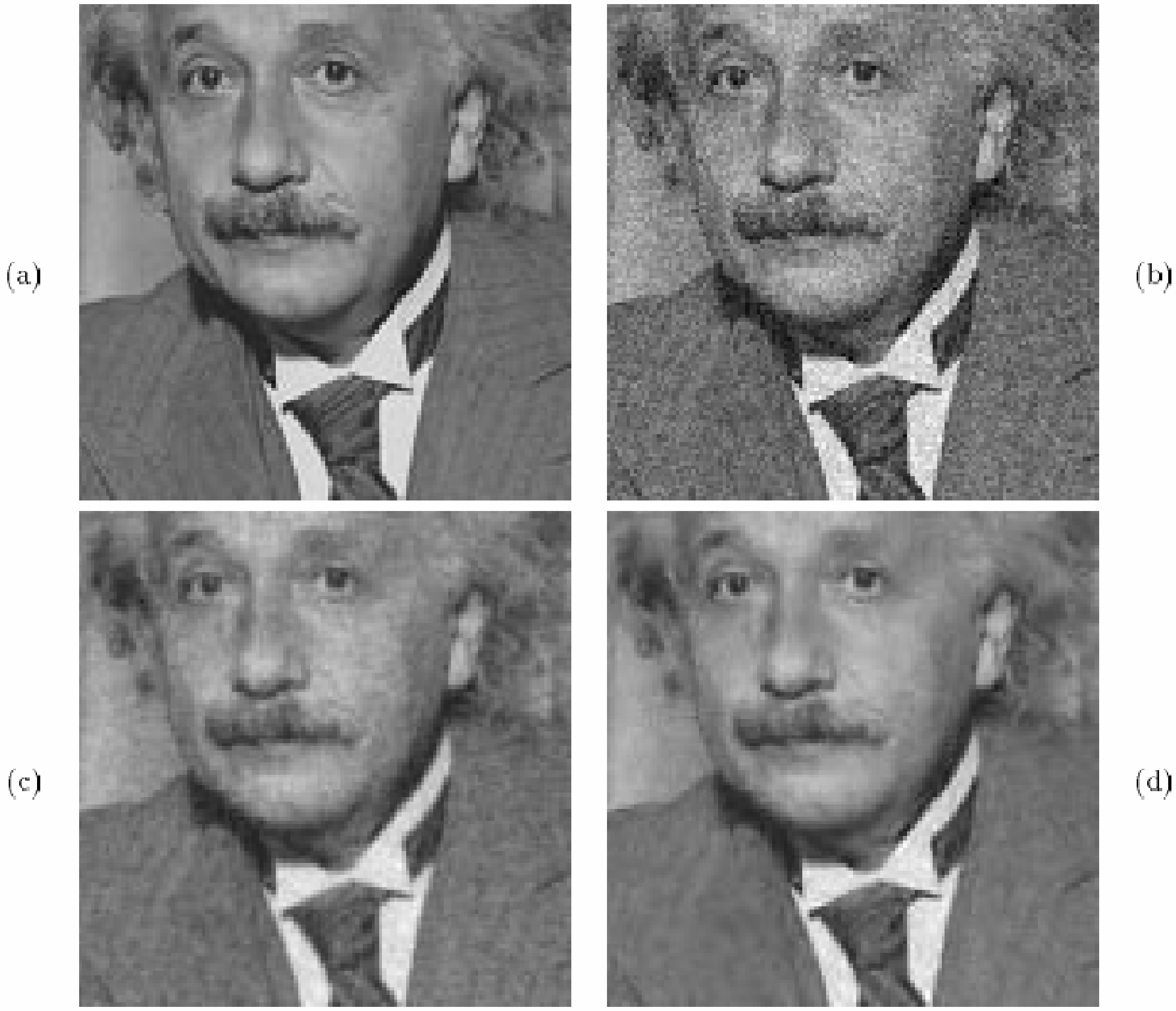


Figure 4: Noise reduction example. (a) Original image (cropped). (b) Image contaminated with additive Gaussian white noise (SNR = 9.00dB). (c) Image restored using (semi-blind) Wiener filter (SNR = 11.88dB). (d) Image restored using (semi-blind) Bayesian estimator (SNR = 13.82dB).

Simoncelli and Adelson, Noise Removal via Bayesian Wavelet Coring

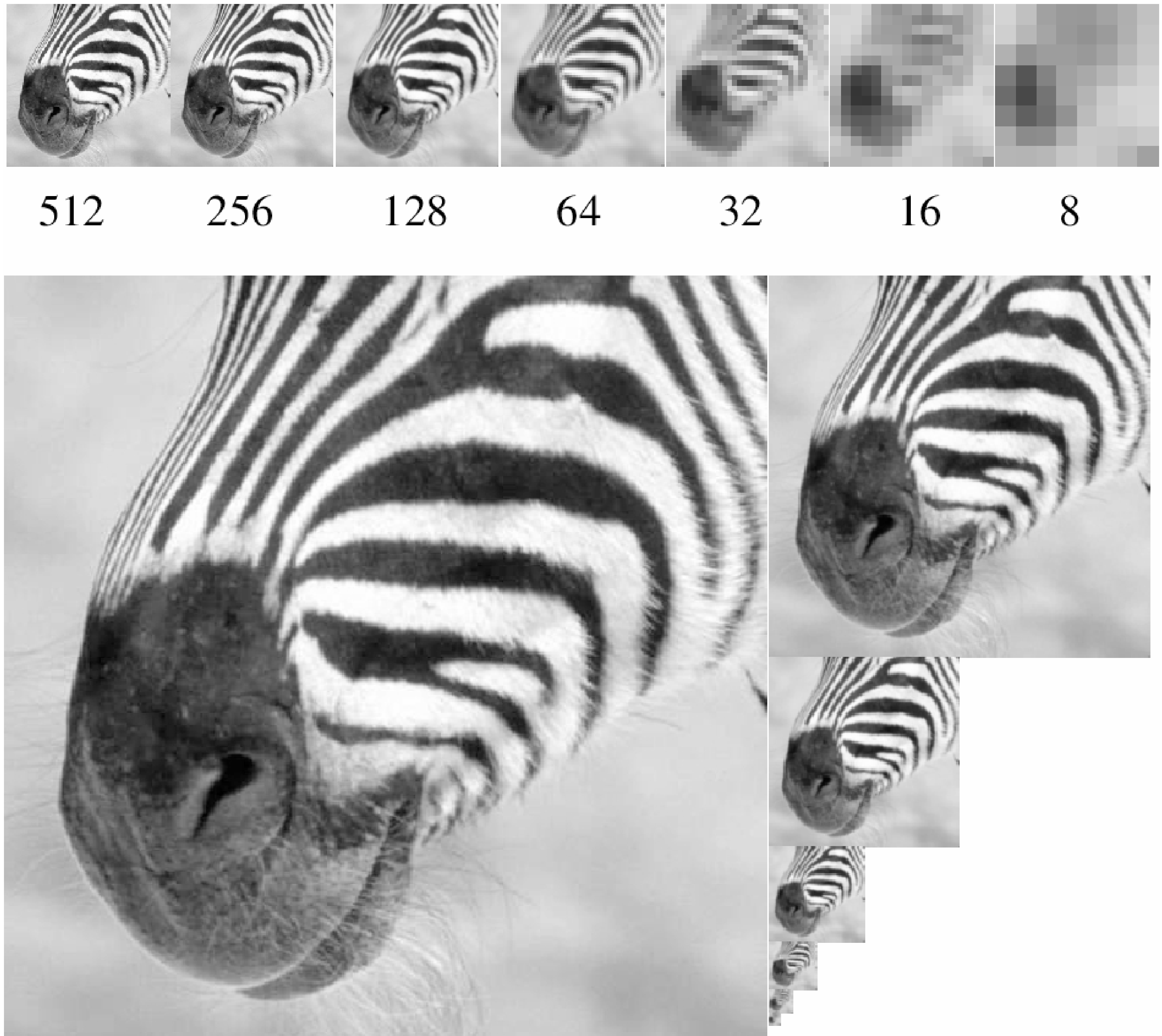
http://www-bcs.mit.edu/people/adelson/pub_pdfs/simoncelli_noise.pdf

Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid

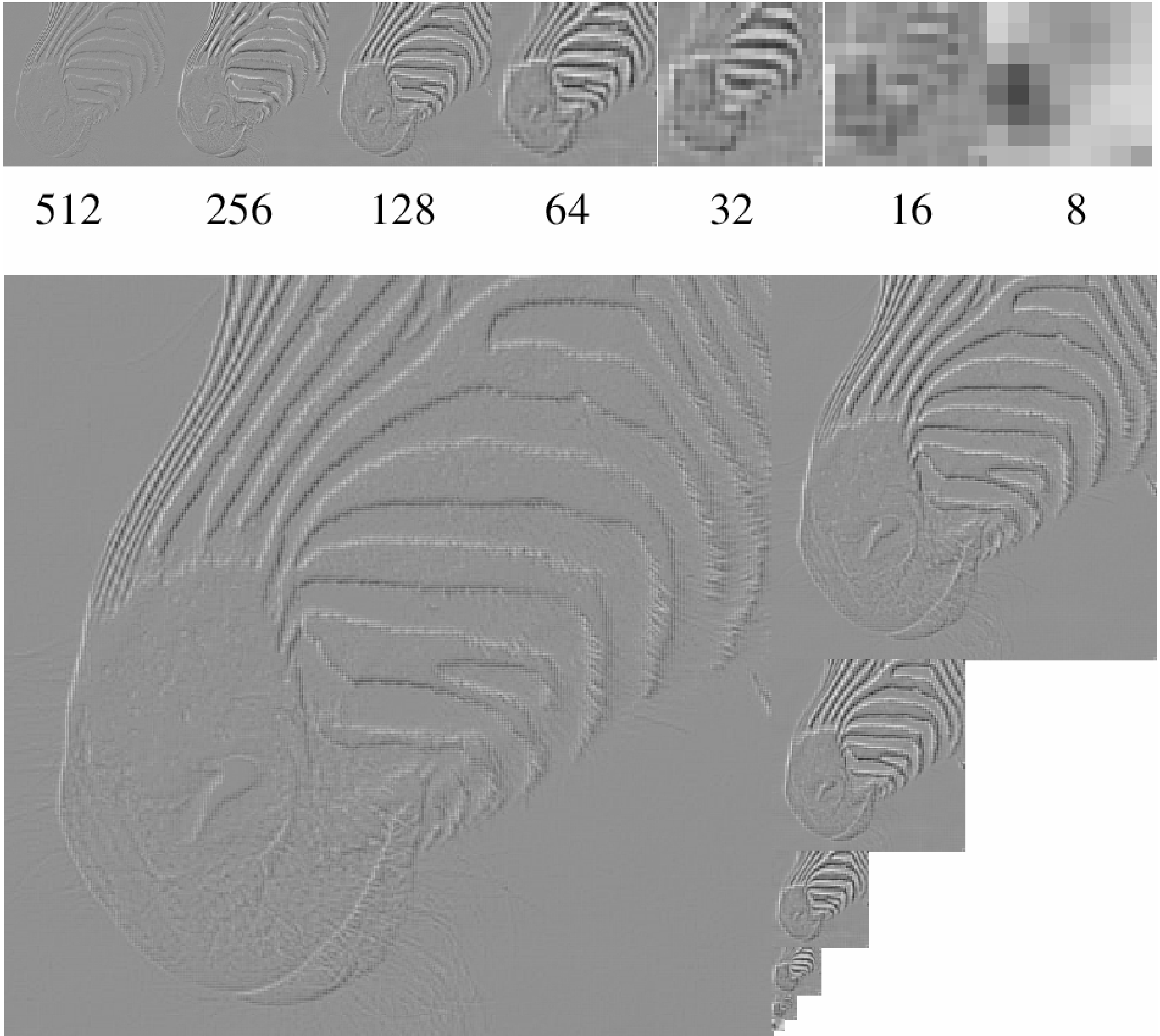
The Gaussian pyramid

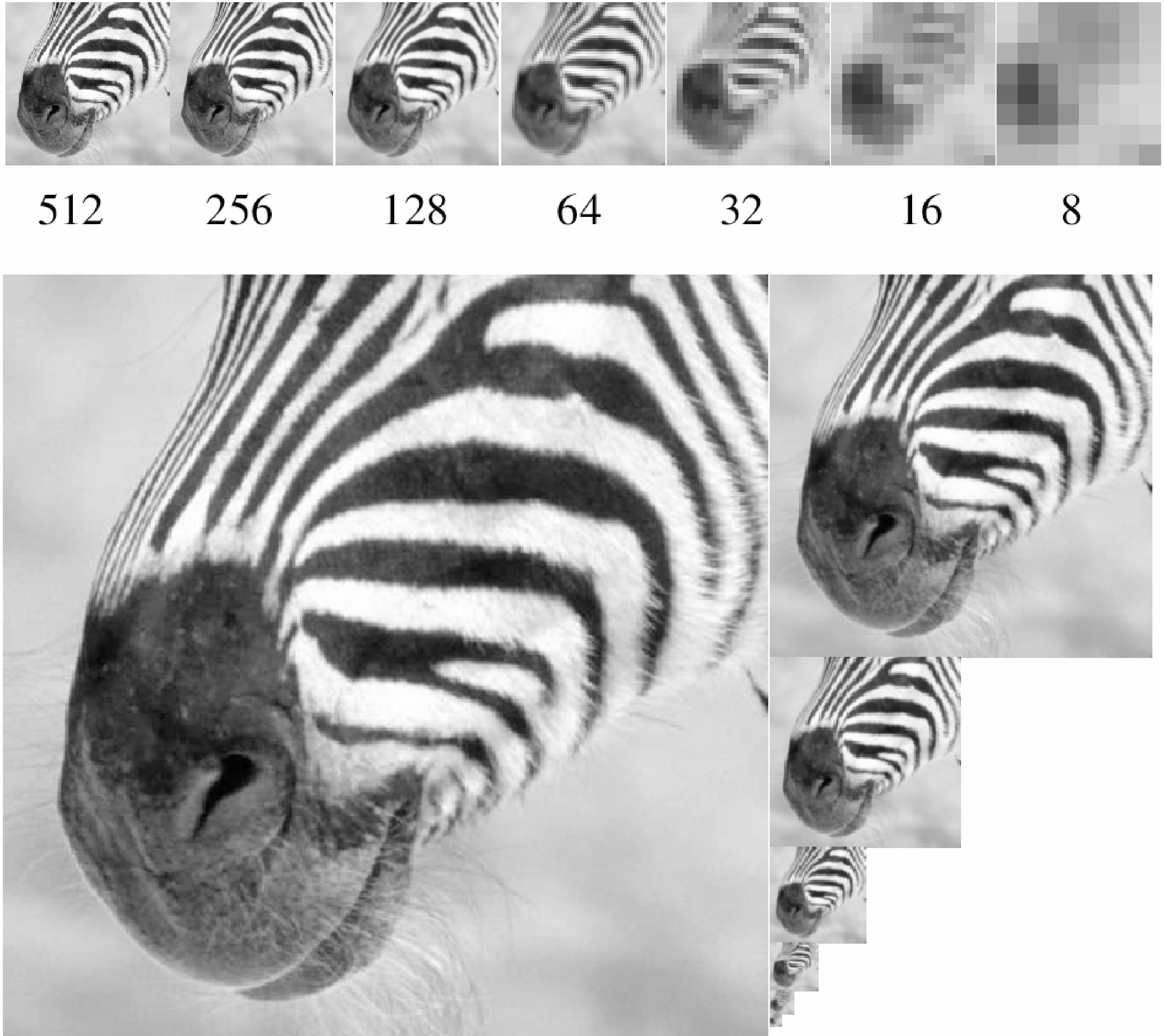
- Smooth with gaussians, because
  - a gaussian*gaussian=another gaussian
- Synthesis
  - smooth and sample
- Analysis
  - take the top image
- Gaussians are low pass filters, so representation is redundant

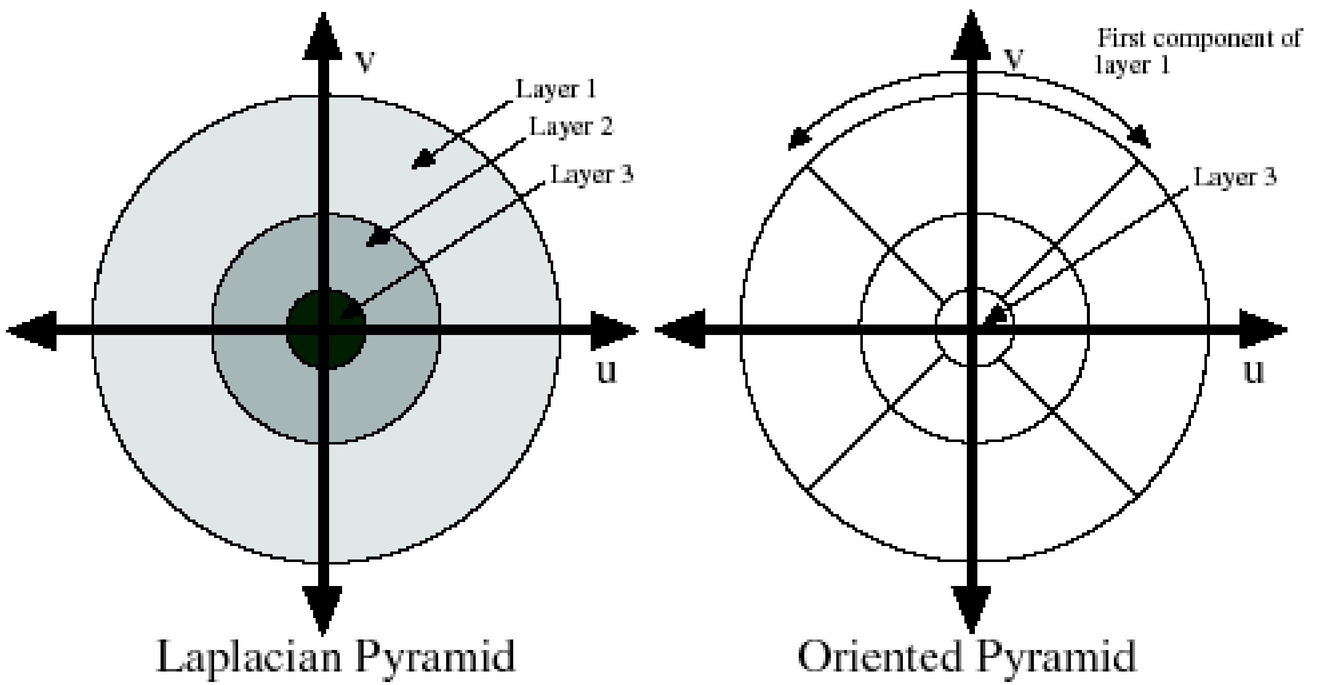

512
256
128
64
32
16
8

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

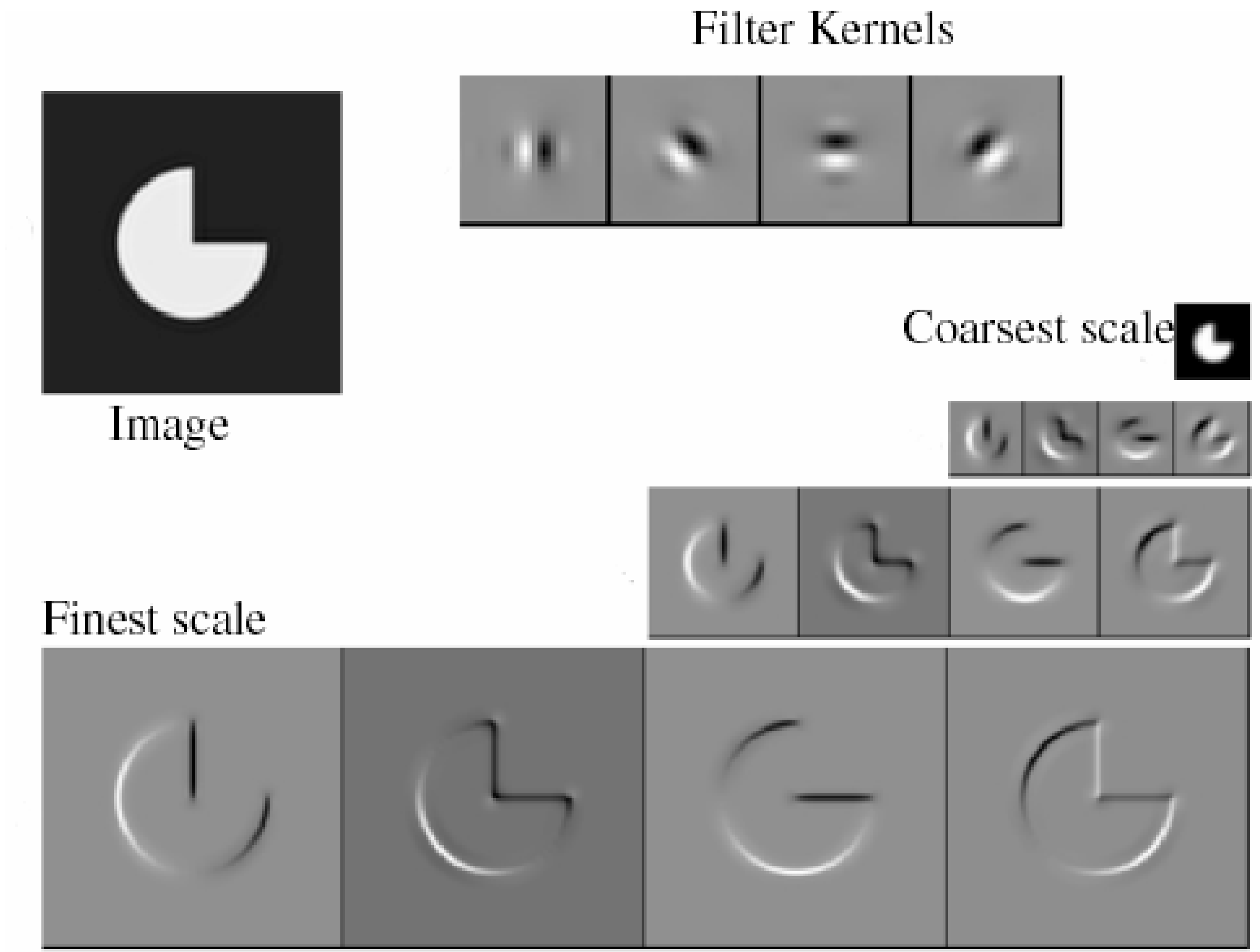

512
256
128
64
32
16
8


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



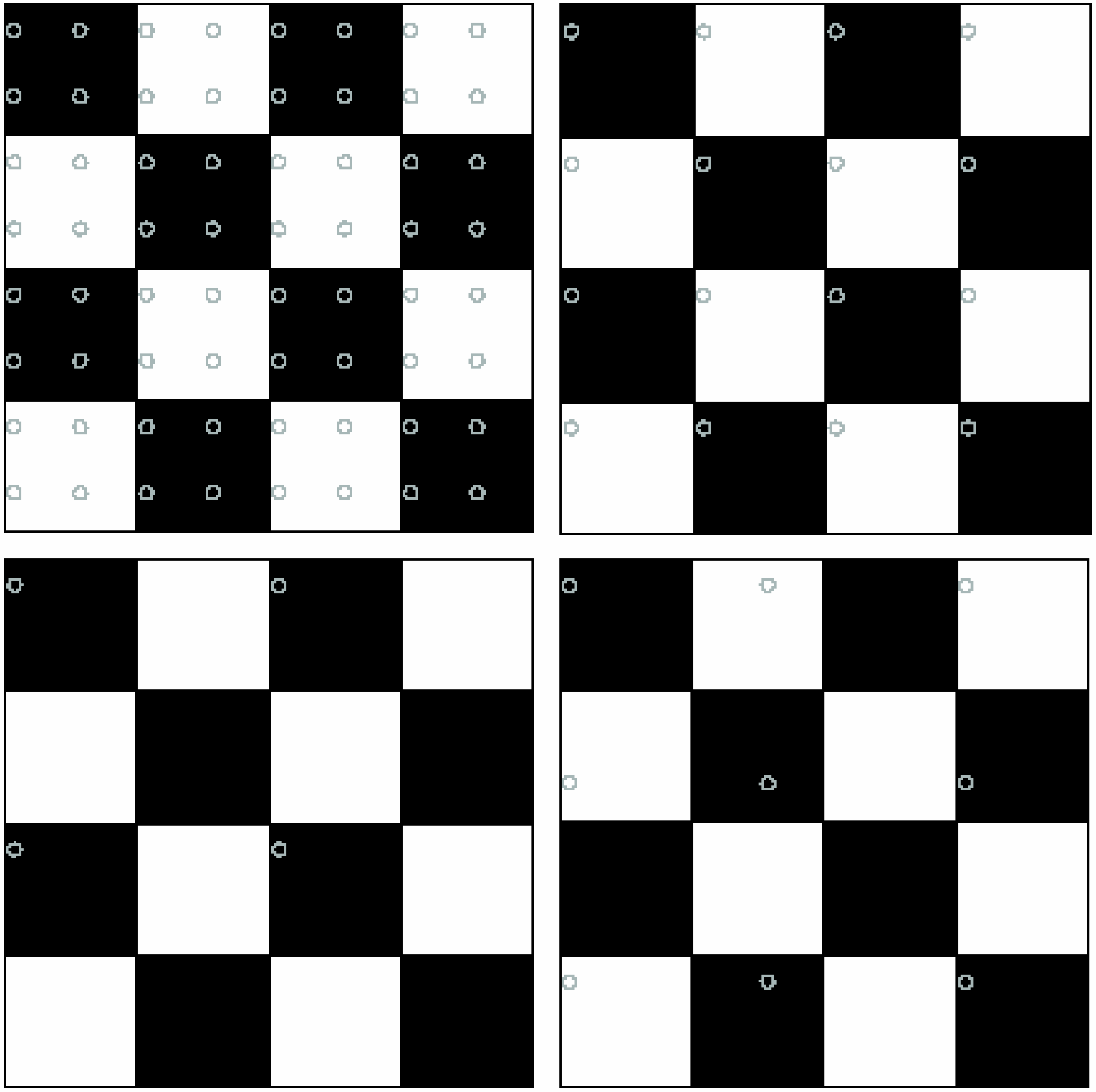
Reprinted from “Shiftable MultiScale Transforms,” by Simoncelli et al., IEEE Transactions on Information Theory, 1992, copyright 1992, IEEE

Reading

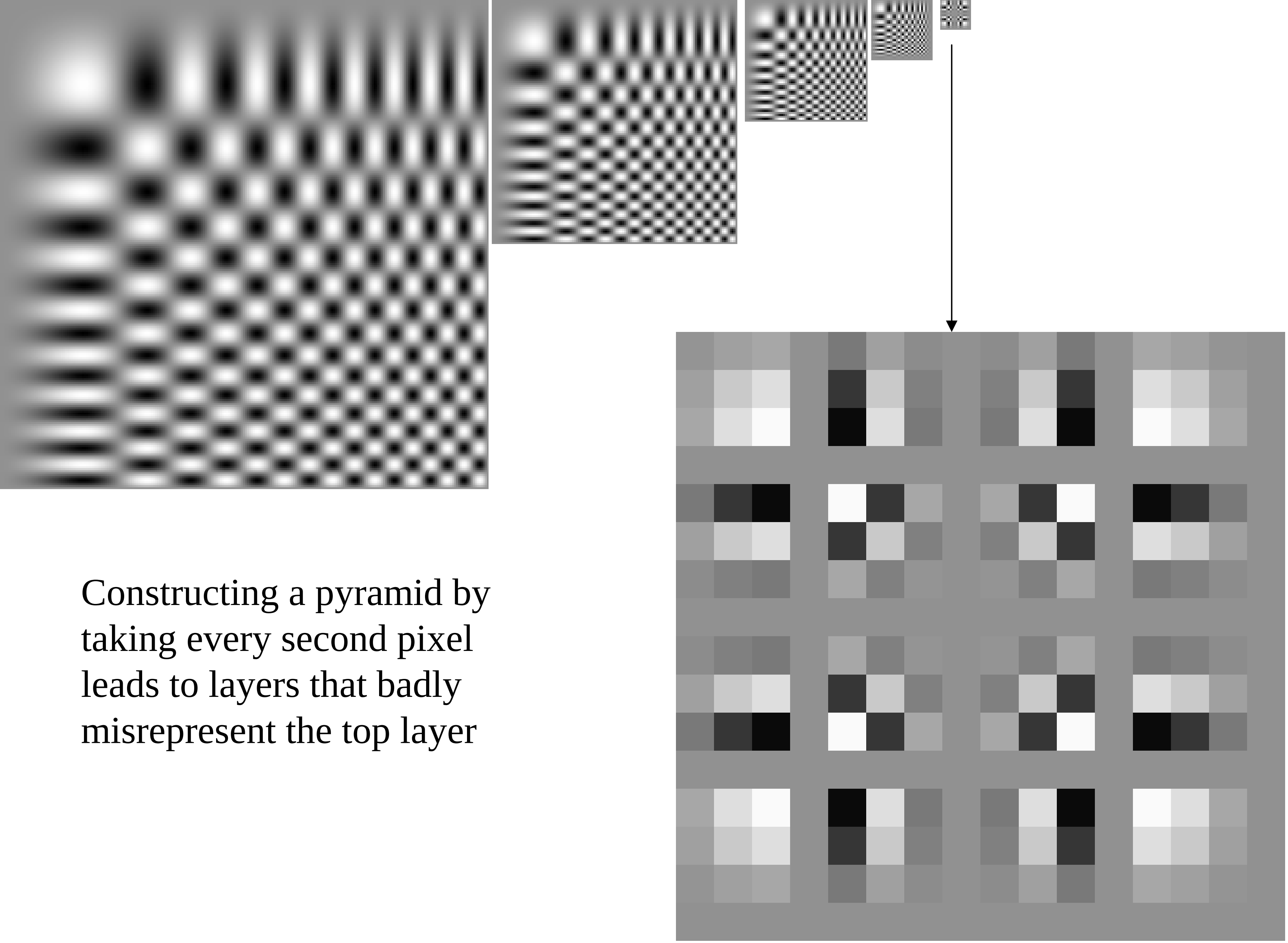
- Related to today's lecture:
  - Chapters 7.7, 9.2, Forsyth&Ponce..
  - Adelson article on pyramid representations, posted on web site.

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.

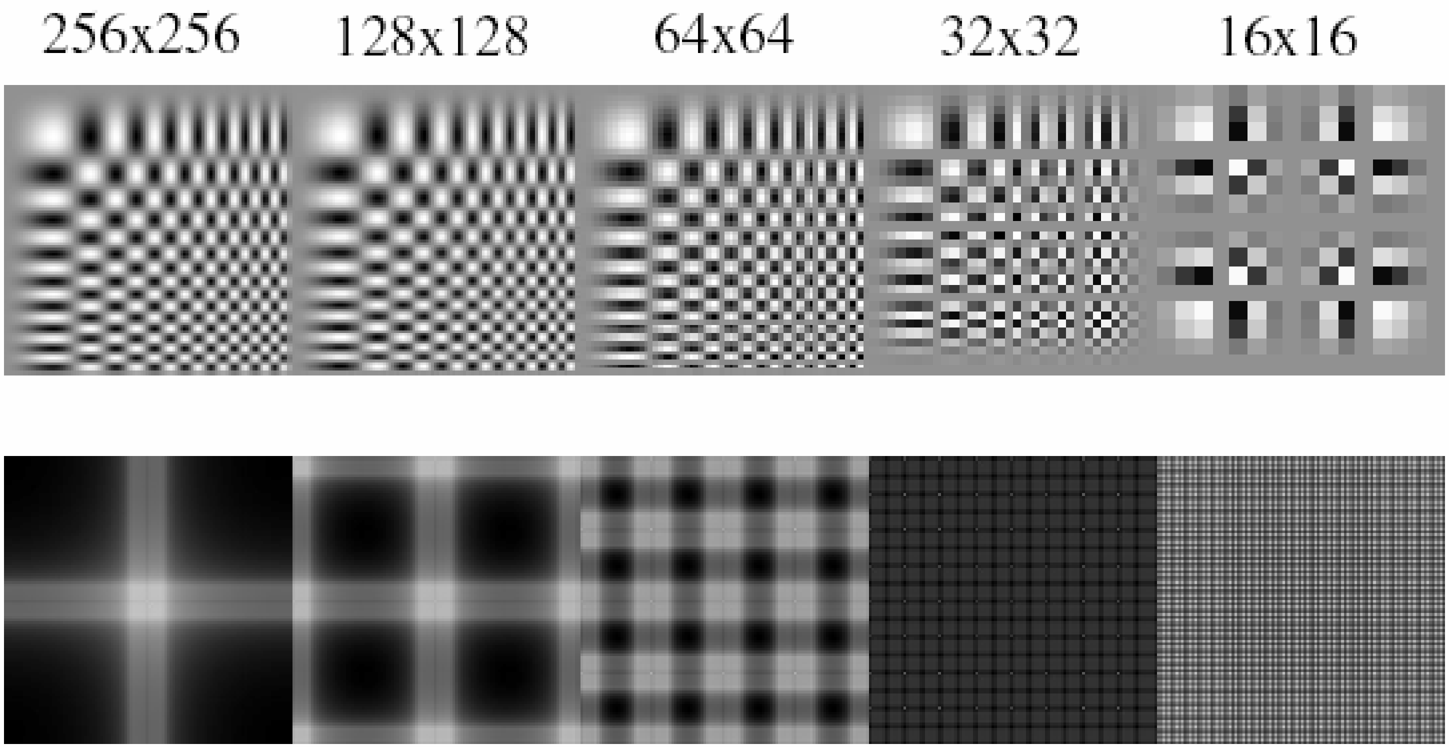


Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer

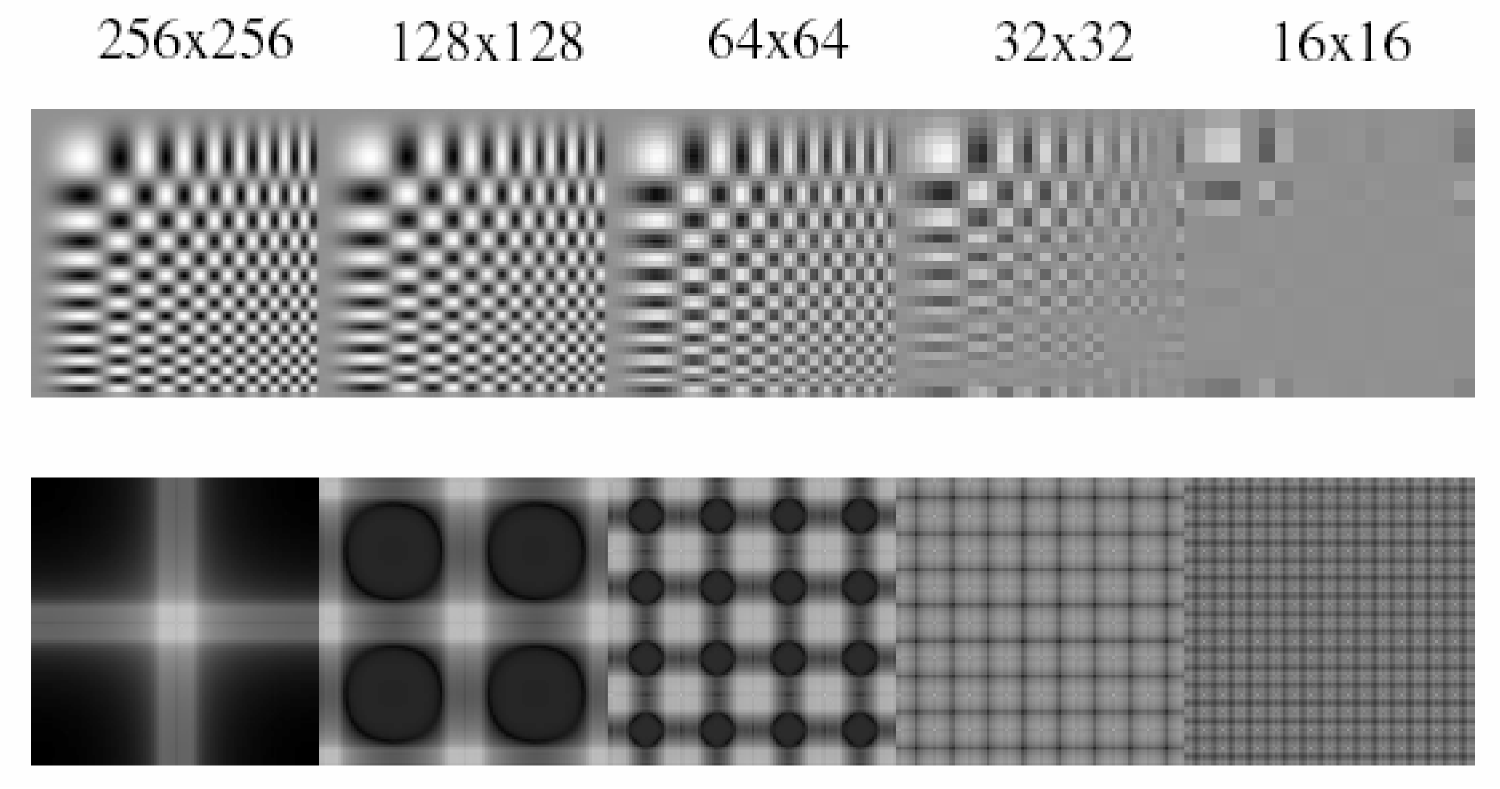
Smoothing as low-pass filtering

- 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.

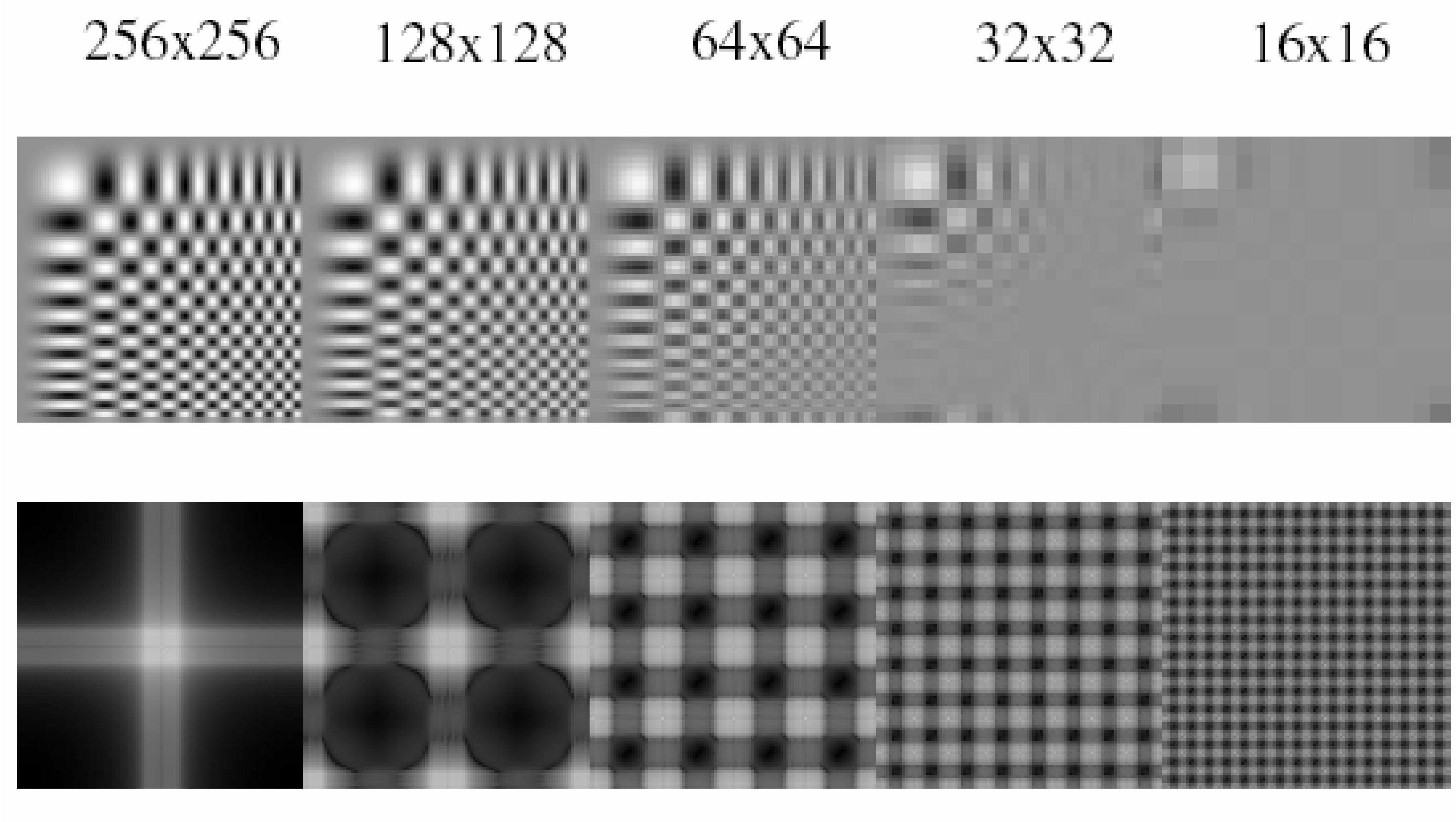
Sampling without smoothing. Top row shows the images, sampled at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.



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 more 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.



Sampling and aliasing