

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>

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	PSo 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		PSo 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		



Course Calendar

Lecture	Date	Description	Readings	Assignments	Materials
2	2/3	Image Filtering	Req: FP 7.1 - 7.6		
3	2/8	Image Representations: Pyramids	Req: FP 7.7, 9.2		

Today

Reading

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

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

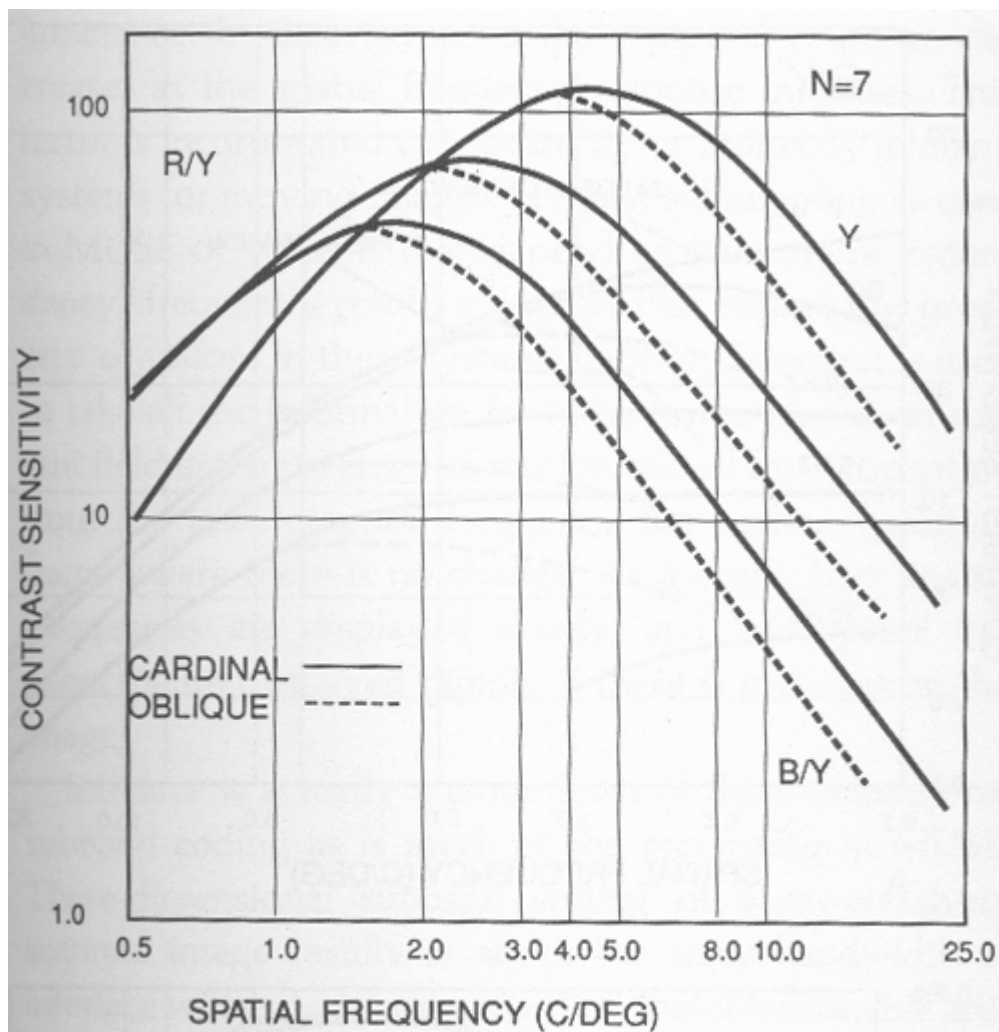


Figure 6.1

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 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

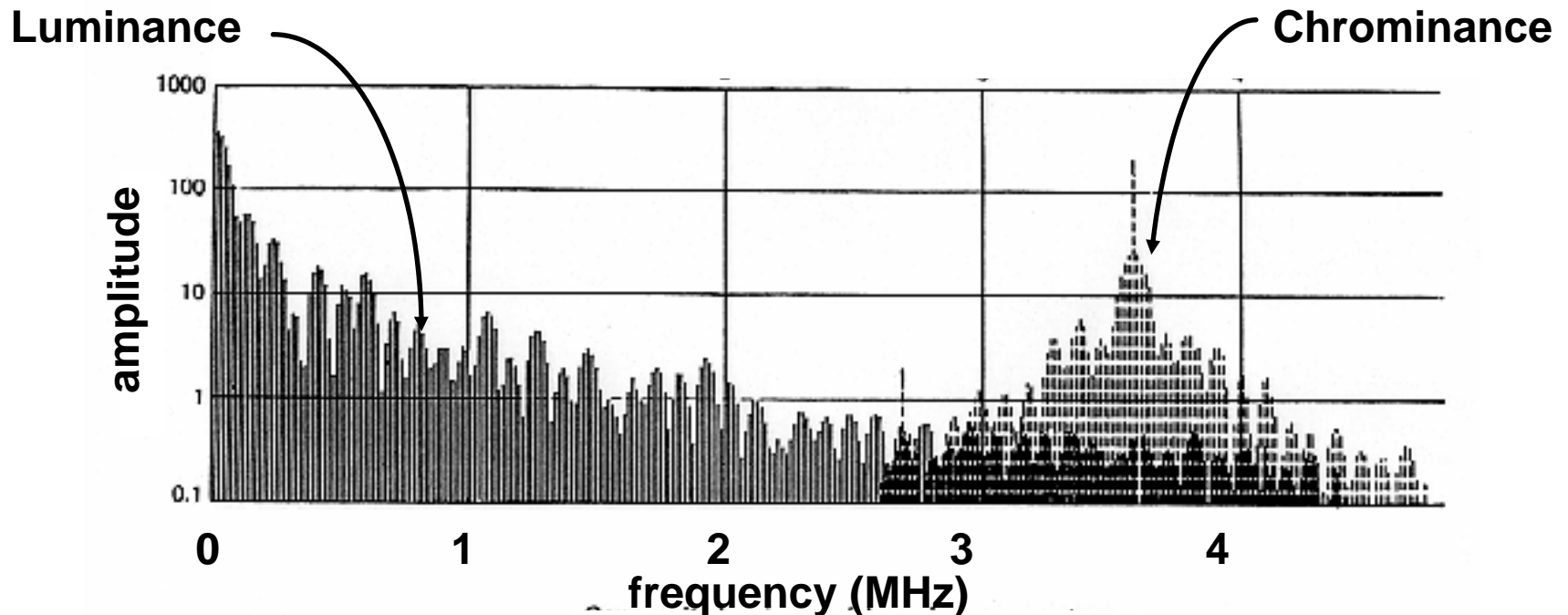


b

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

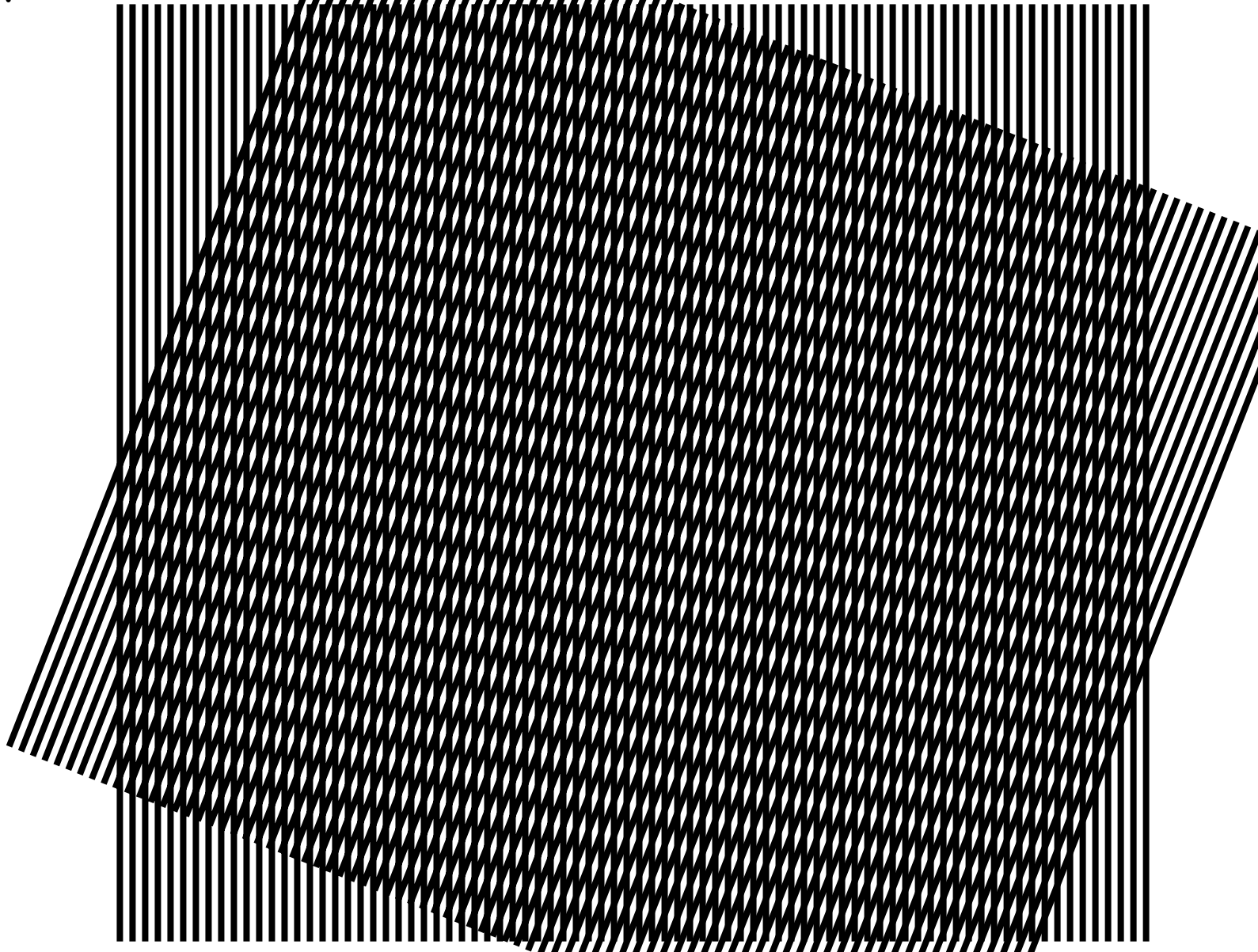


Understanding image perception allowed NTSC to add color to the black and white television signal (with some, but limited, incompatibility artifacts).

Sampling and aliasing

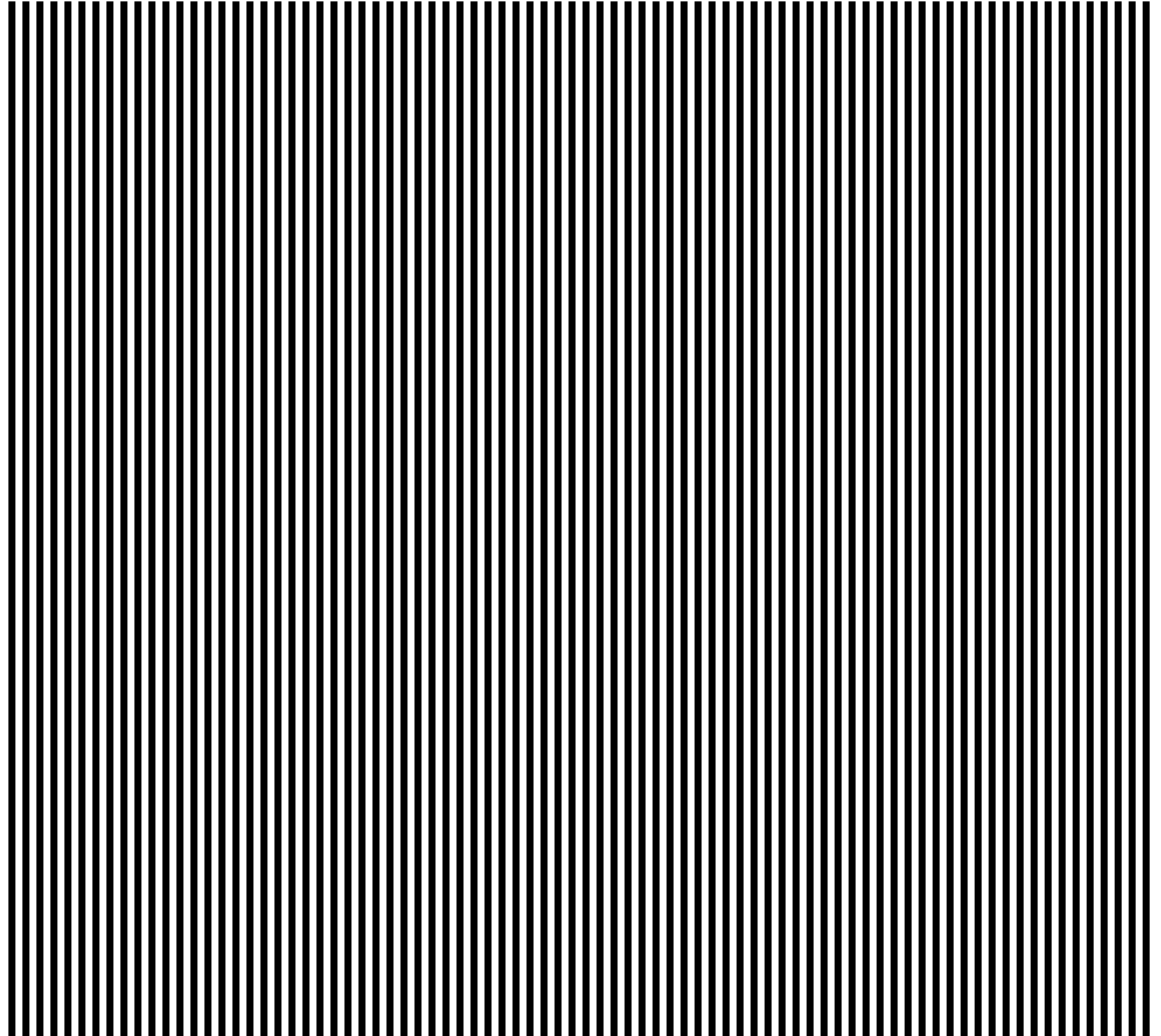
Sampling example

Analyze crossed
gratings...



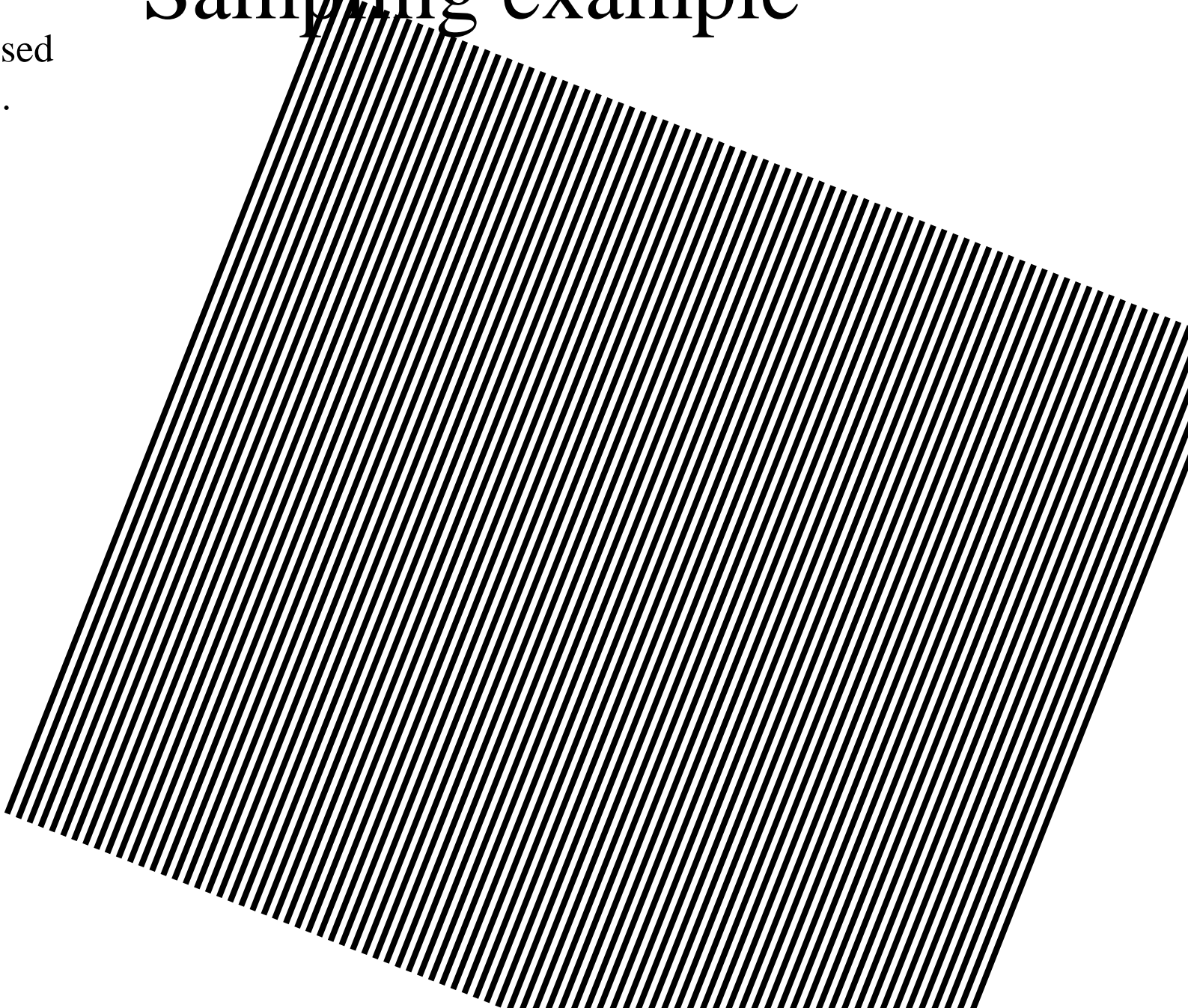
Sampling example

Analyze crossed
gratings...



Sampling example

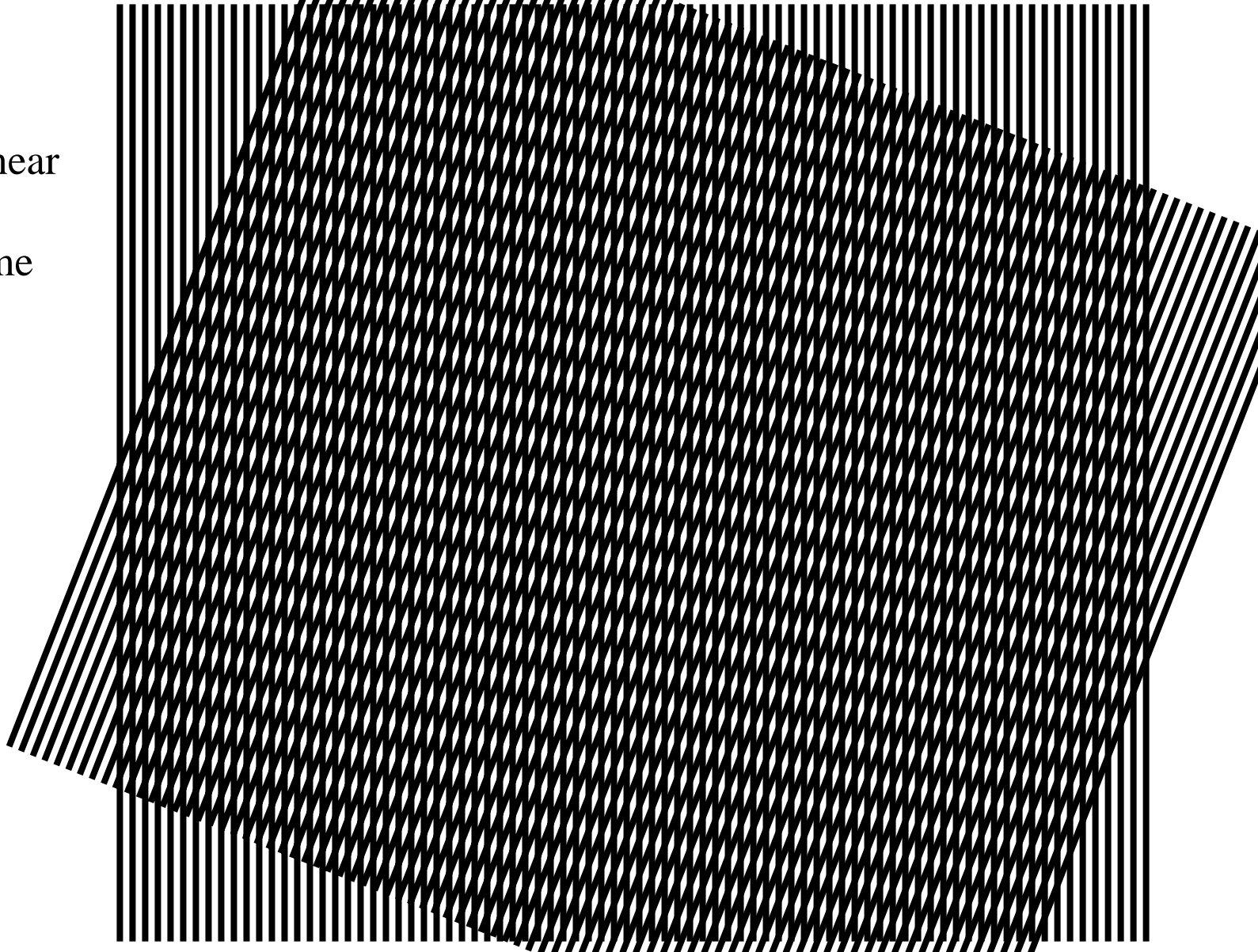
Analyze crossed
gratings...

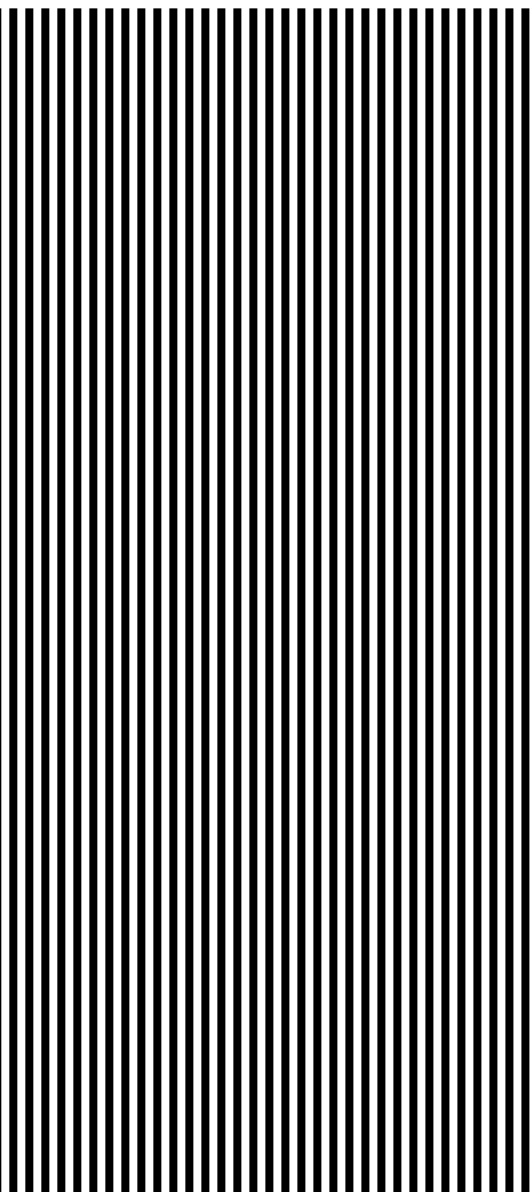


Sampling example

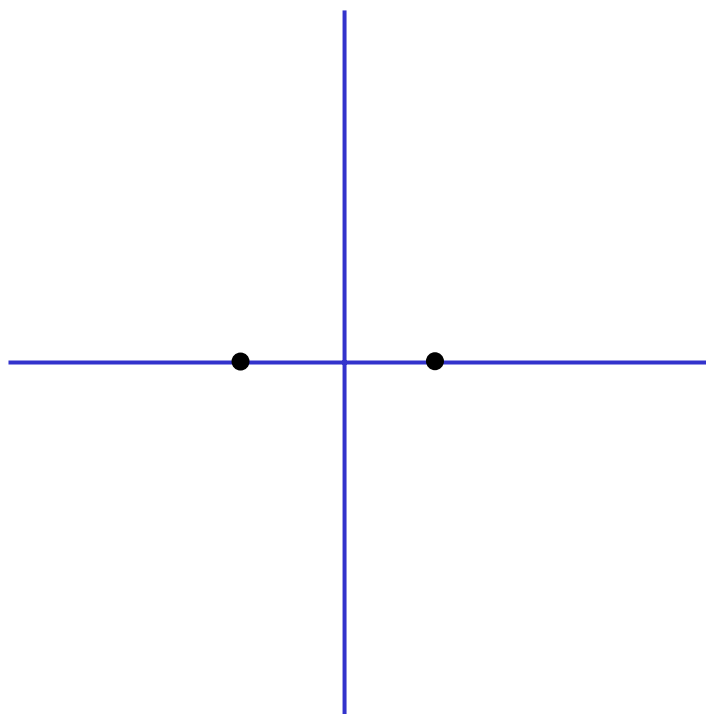
Analyze crossed
gratings...

Where does
perceived near
horizontal
grating come
from?

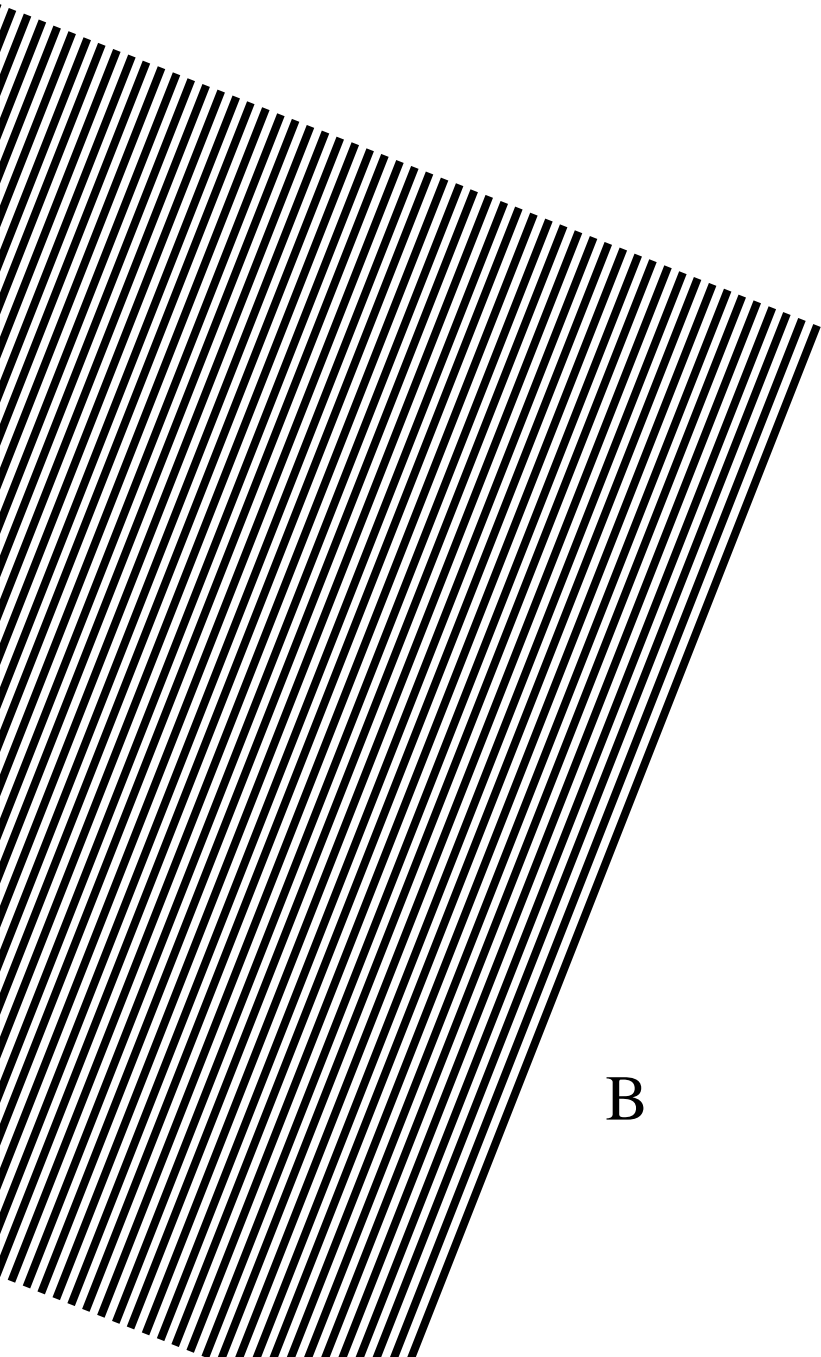




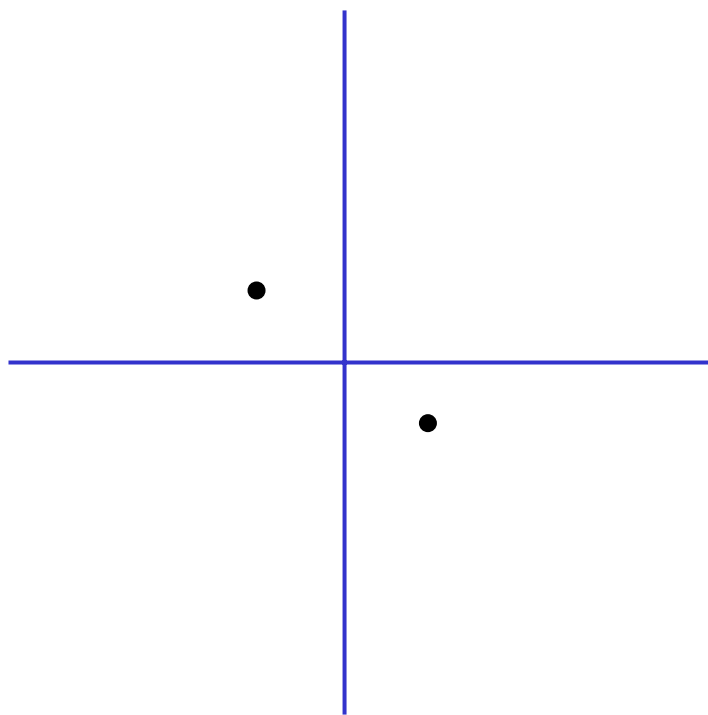
A



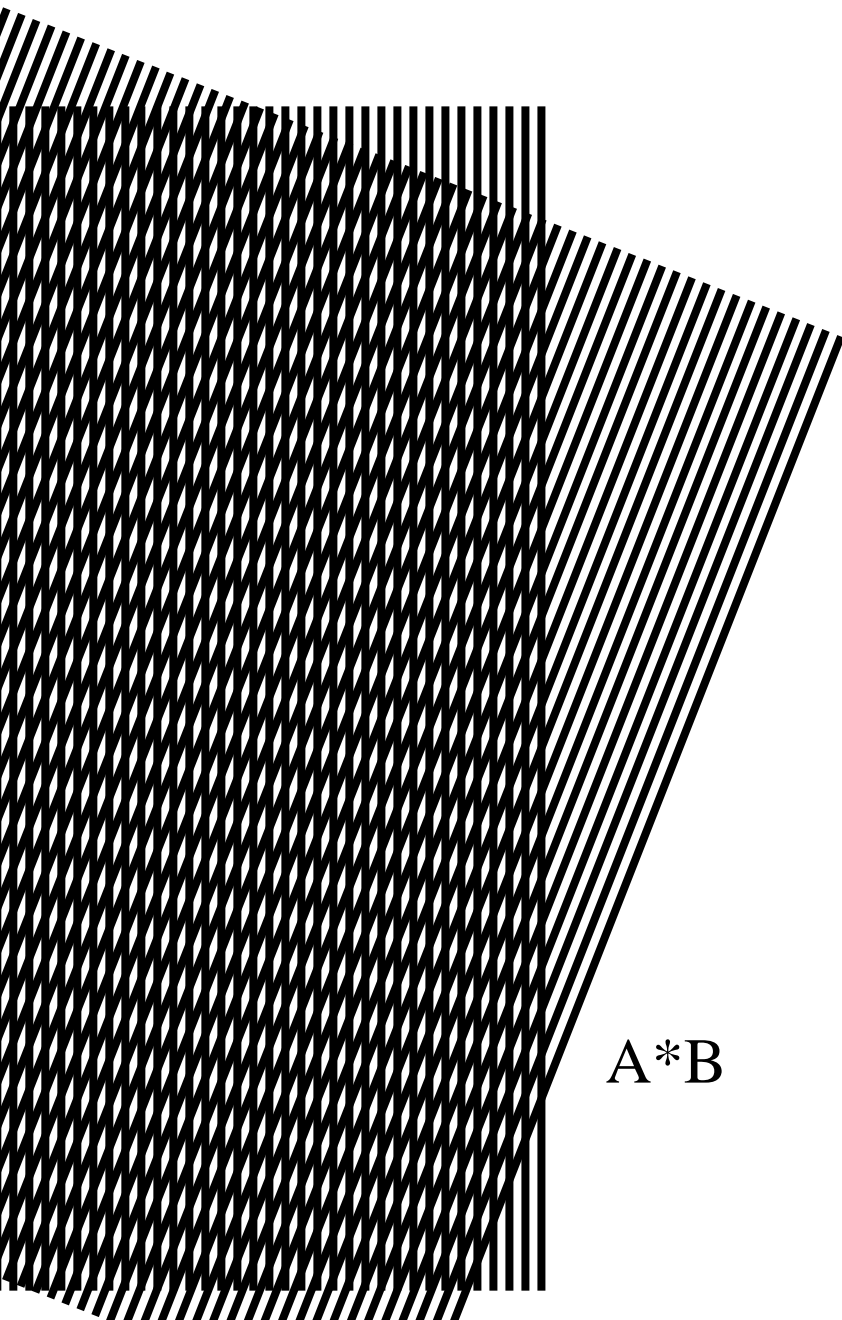
$F(A)$



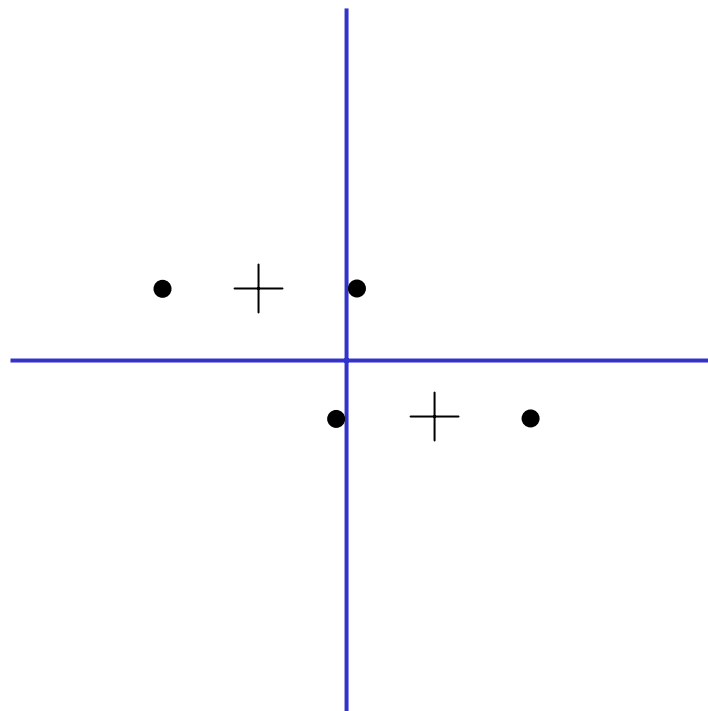
B



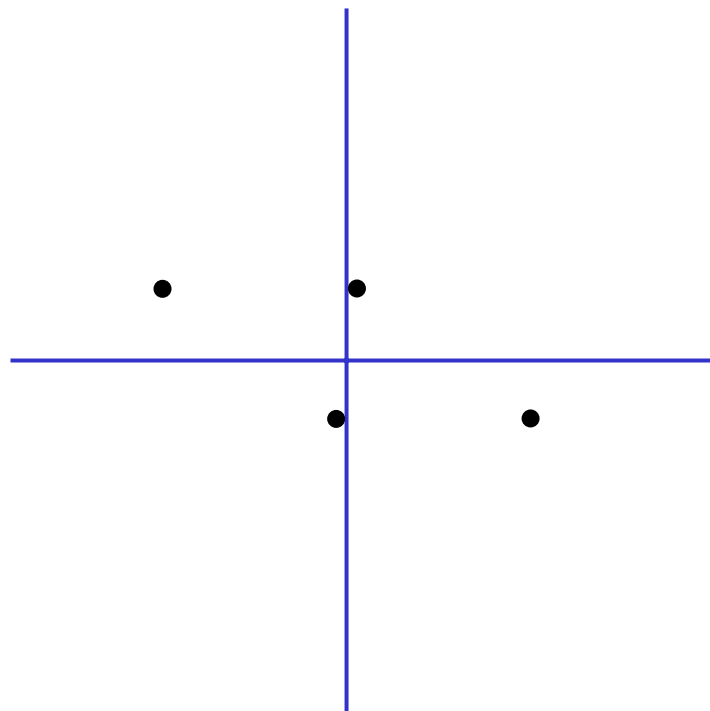
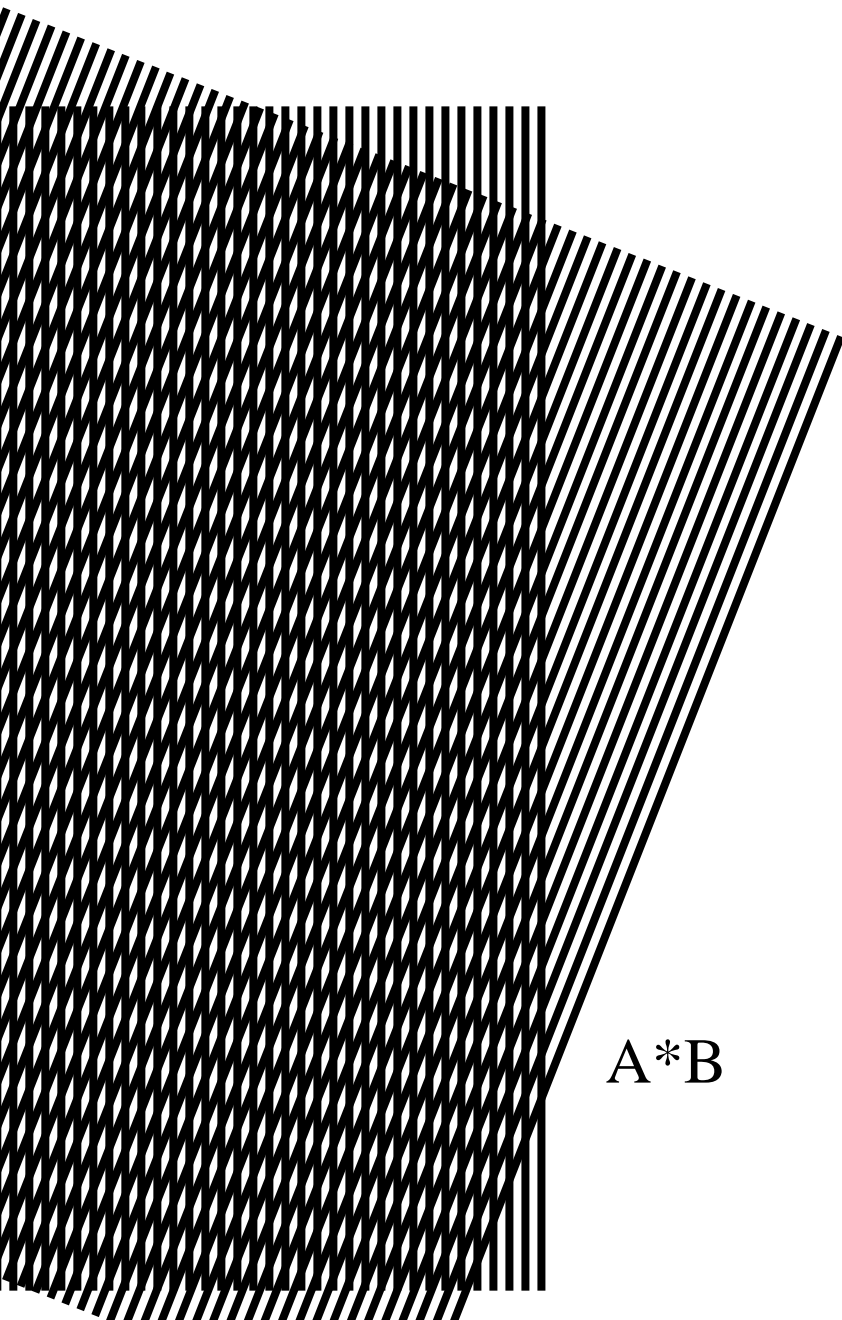
$F(B)$



$A * B$

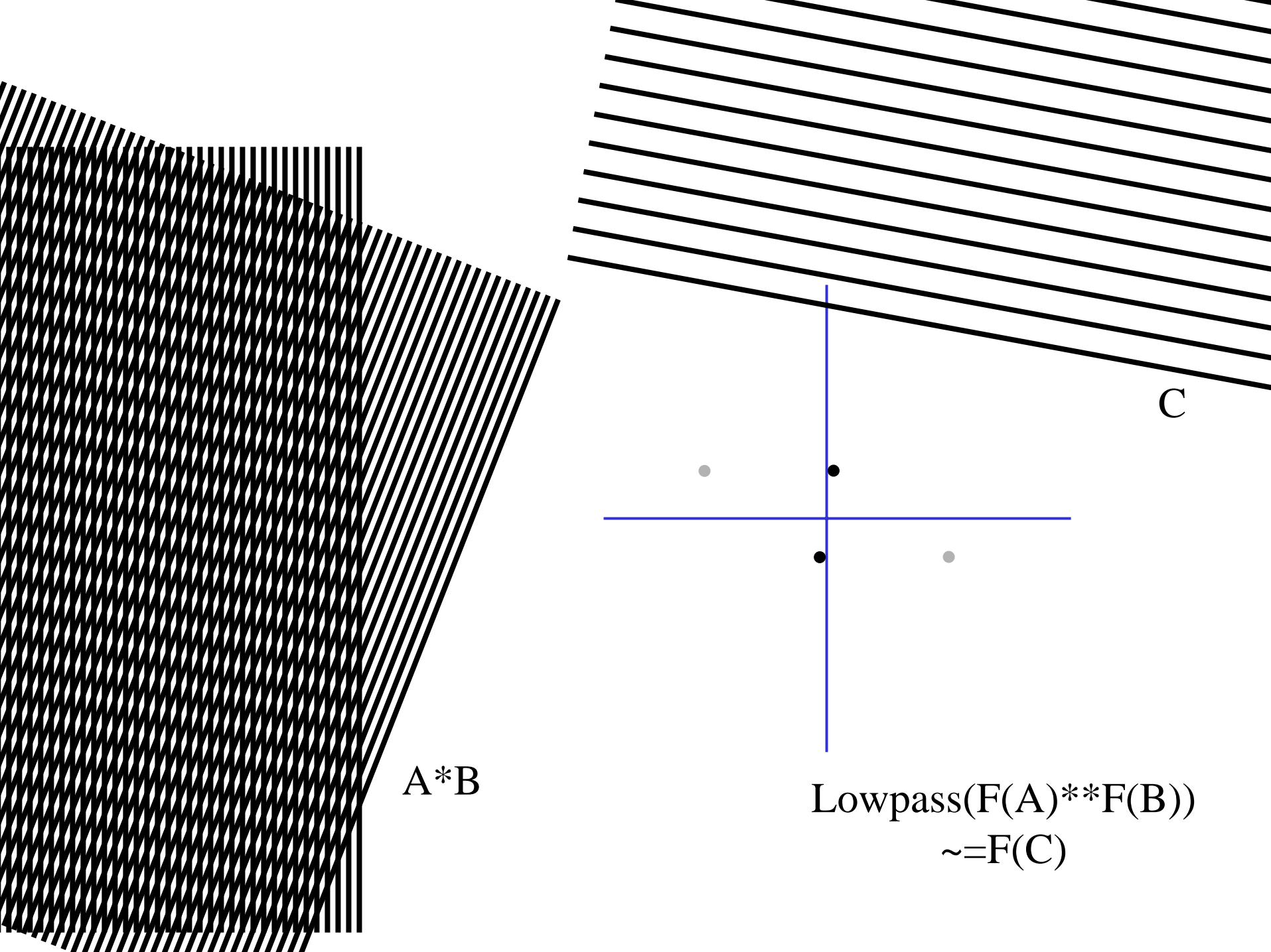


$F(A) ** F(B)$



$A*B$

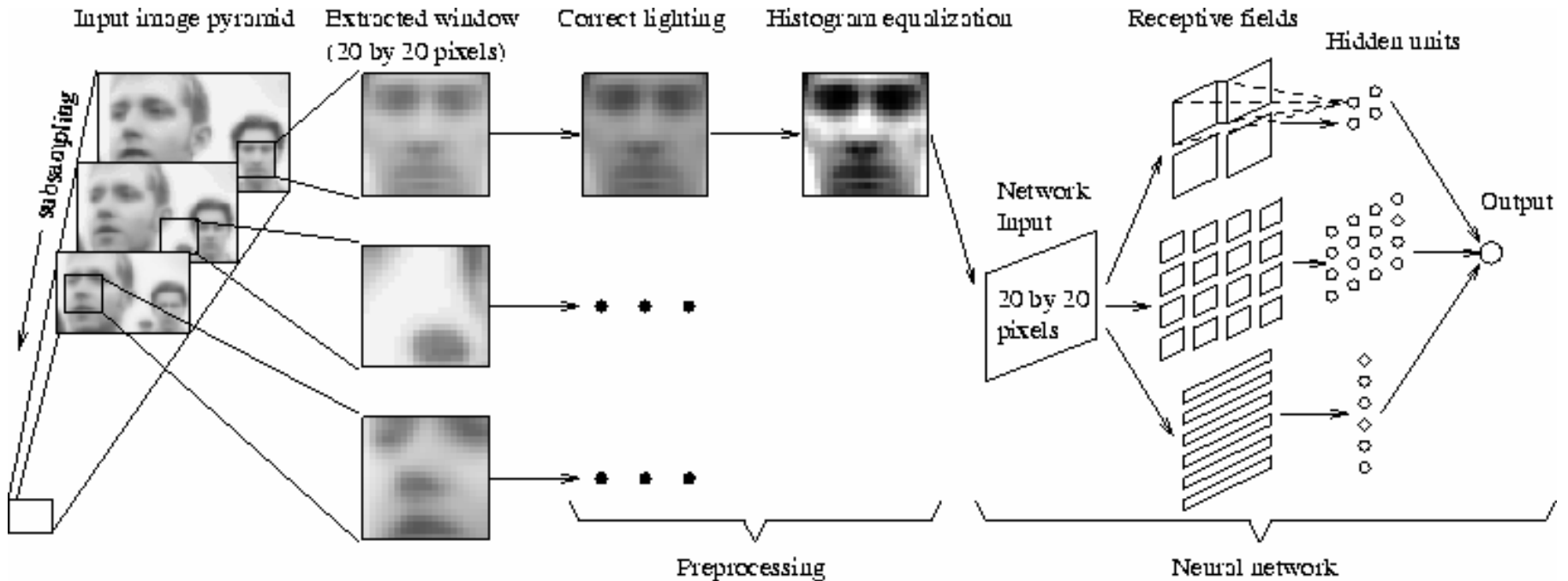
$F(A)**F(B)$



Scaled representations

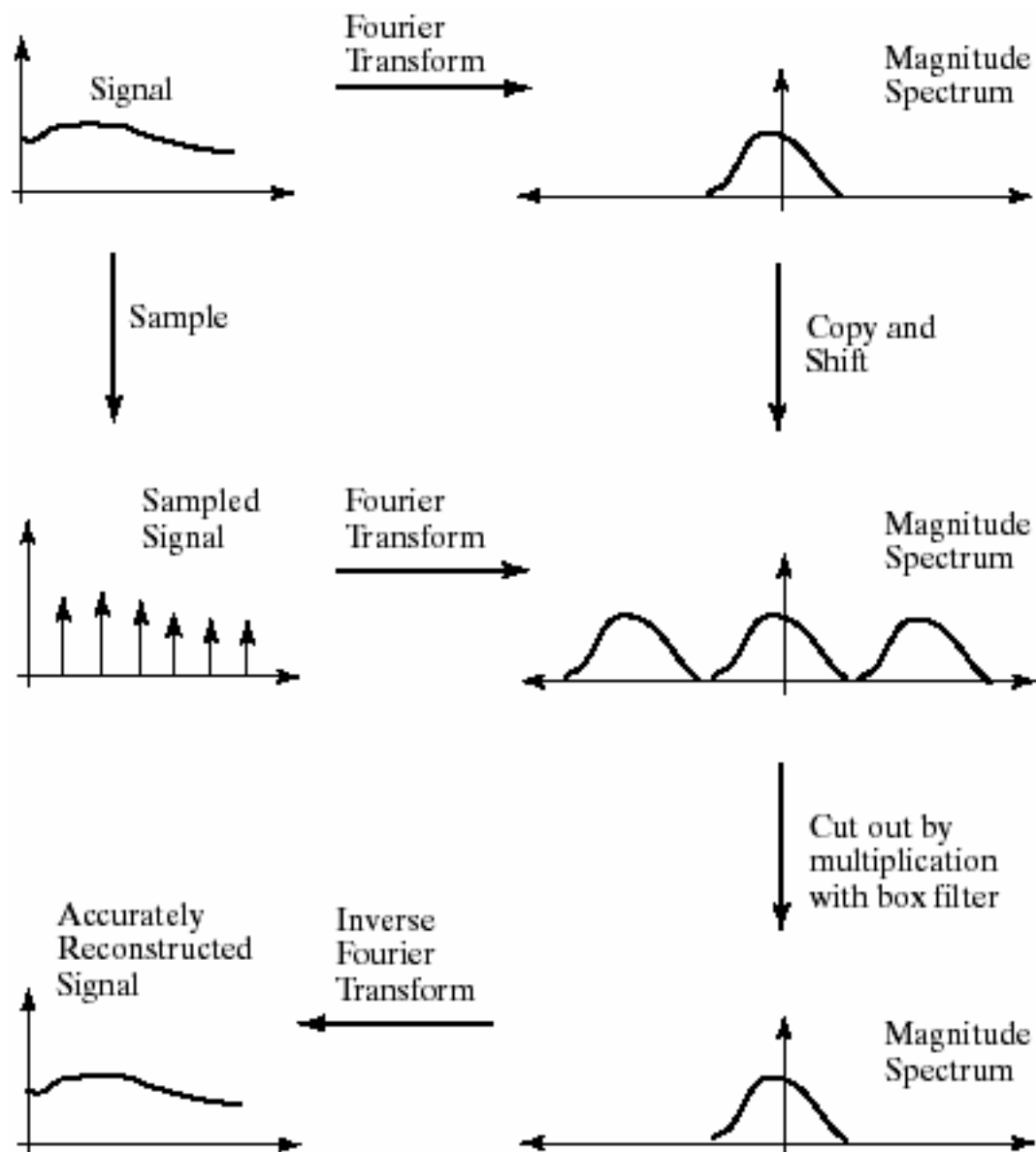
- Big bars (resp. spots, hands, etc.) and little bars are both interesting
 - Stripes and hairs, say
- Inefficient to detect big bars with big filters
 - And there is superfluous detail in the filter kernel
- Alternative:
 - Apply filters of fixed size to images of different sizes
 - Typically, a collection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

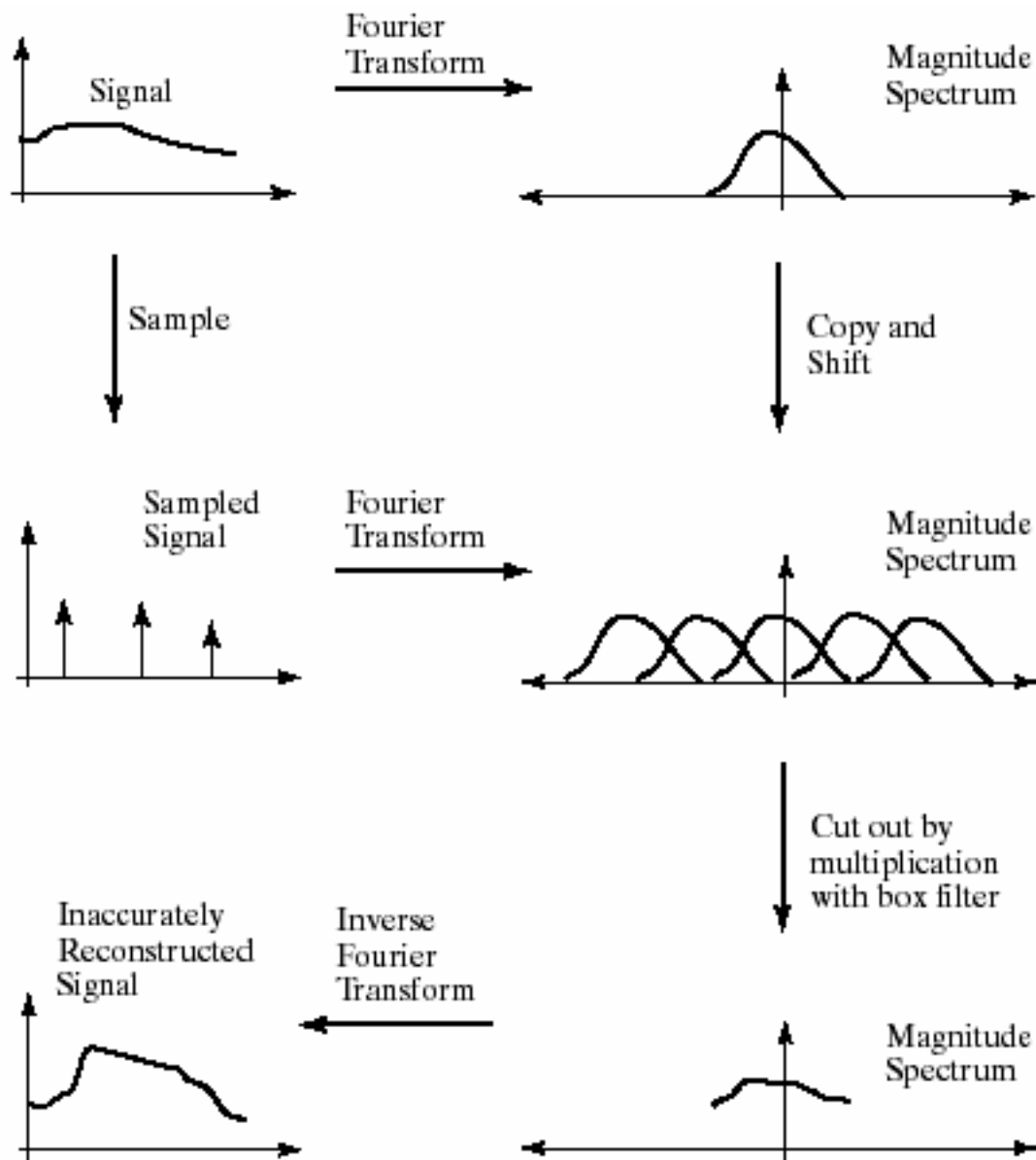
Example application: CMU face detector



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

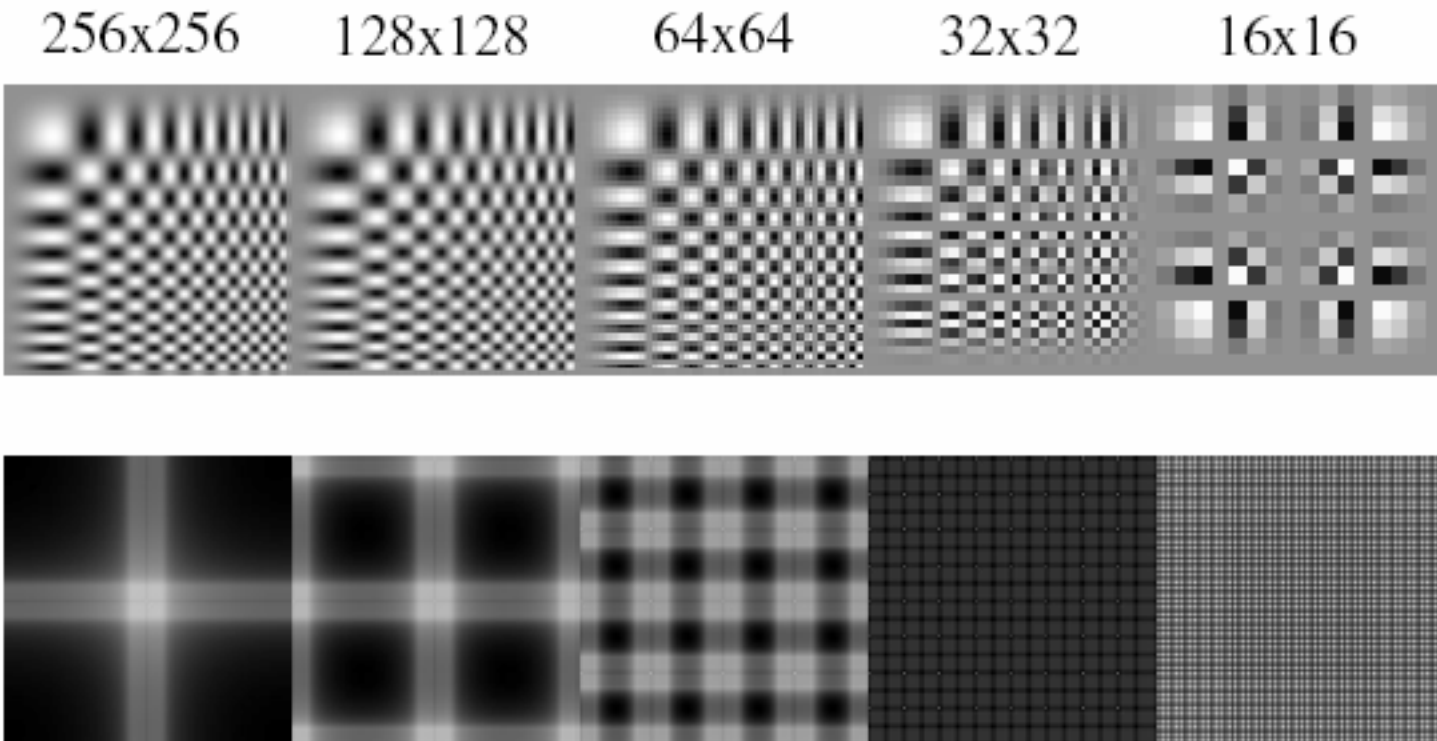




Smoothing as low-pass filtering

- 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.

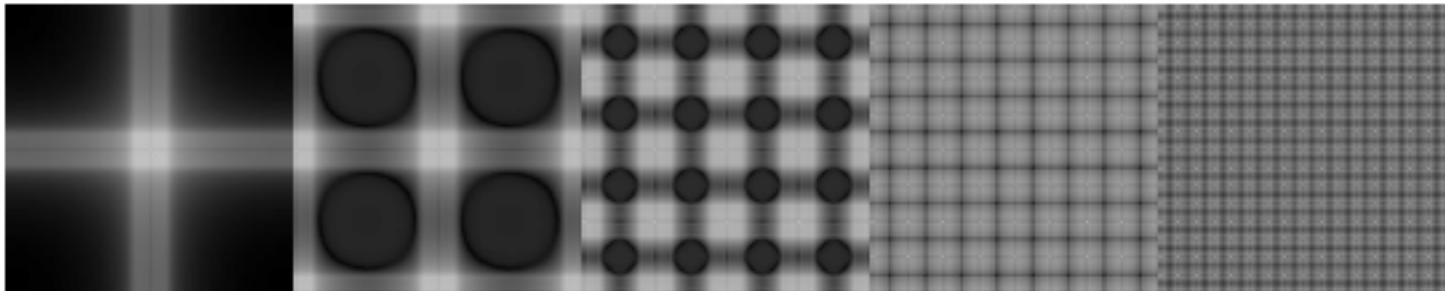
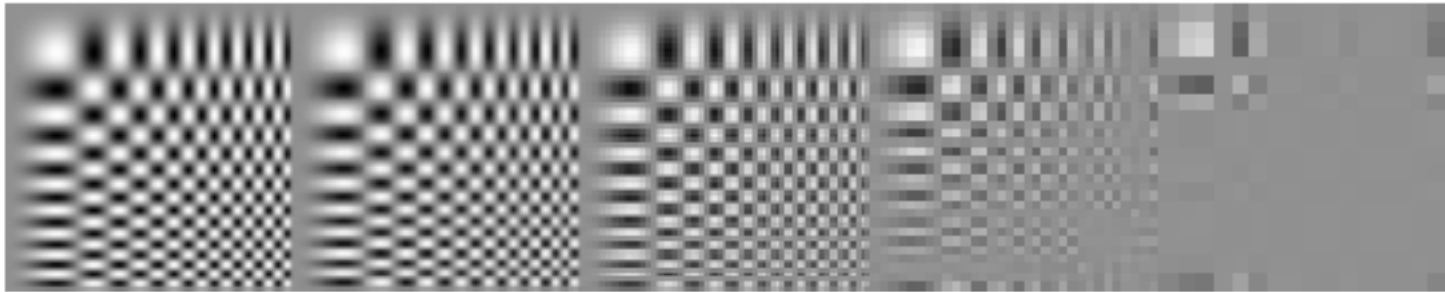
256x256

128x128

64x64

32x32

16x16



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

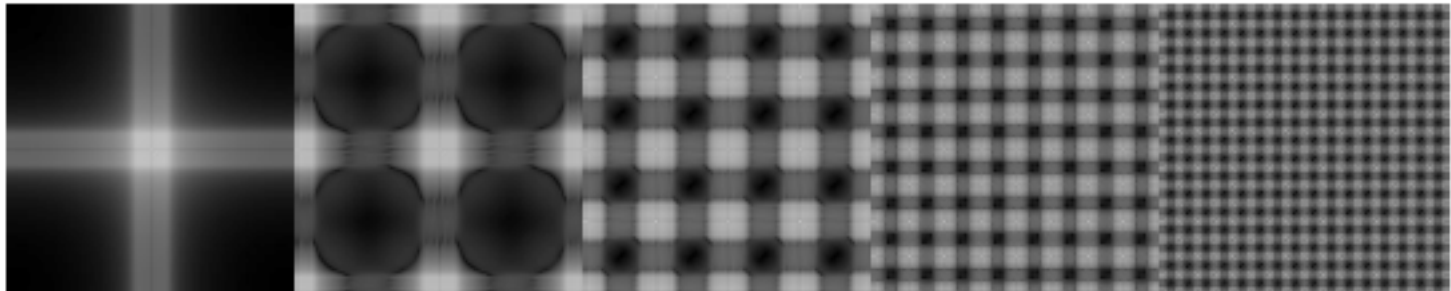
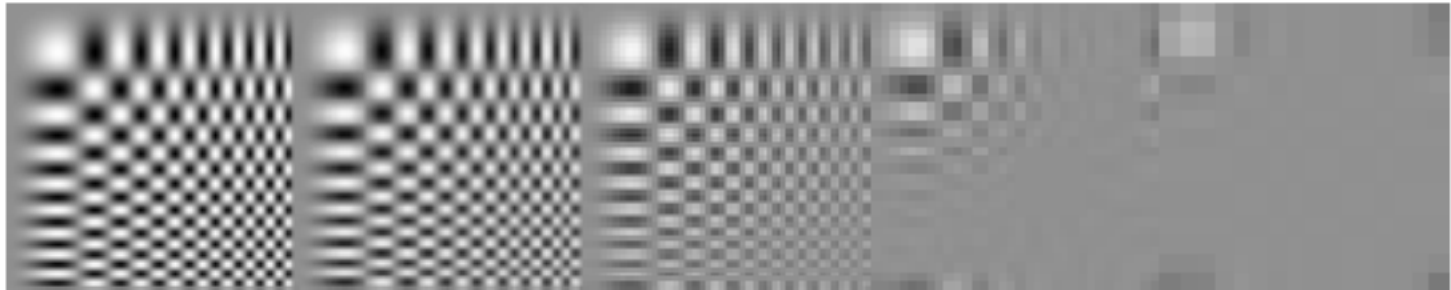
256x256

128x128

64x64

32x32

16x16



Matlab

Subsample image in matlab.

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

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

The computational advantage of pyramids

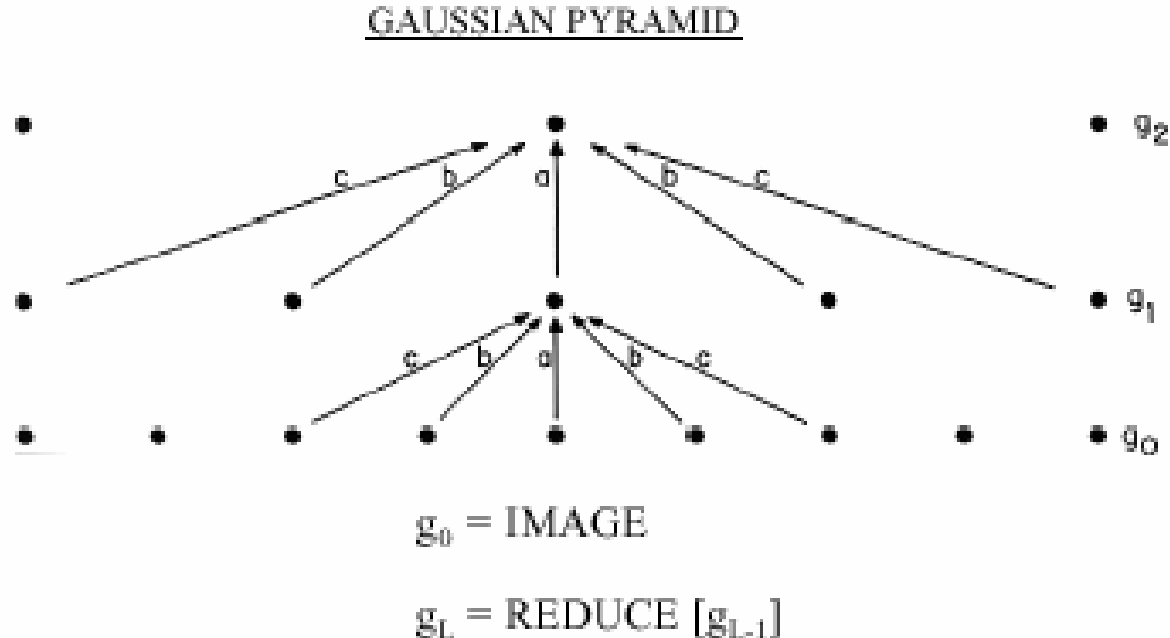


Fig 1. A one-dimensional graphic representation of the process which generates a Gaussian pyramid. Each row of dots represents nodes within a level of the pyramid. The value of each node in the zero level is just the gray level of a corresponding image pixel. The value of each node in a high level is 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.

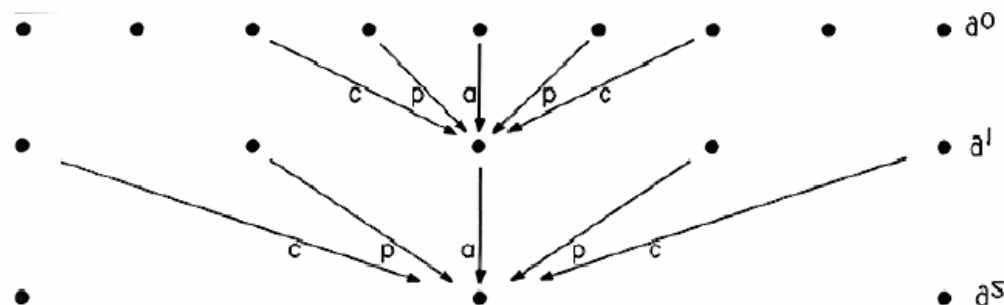
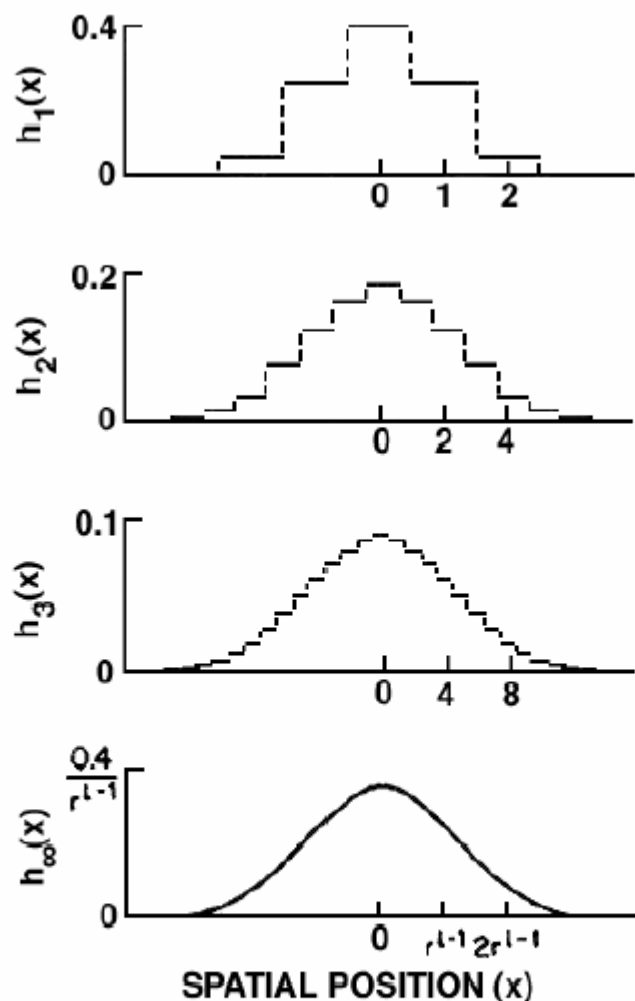


Fig. 2. The equivalent weighting functions $h_l(x)$ for nodes in levels 1, 2, 3, and infinity of the Gaussian pyramid. Note that axis scales have been adjusted by factors of 2 to aid comparison. Here the parameter a of the generating kernel is 0.4, and the resulting equivalent weighting functions closely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



0

GAUSSIAN PYRAMID



1



2



3



4



5

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



512

256

128

64

32

16

8



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



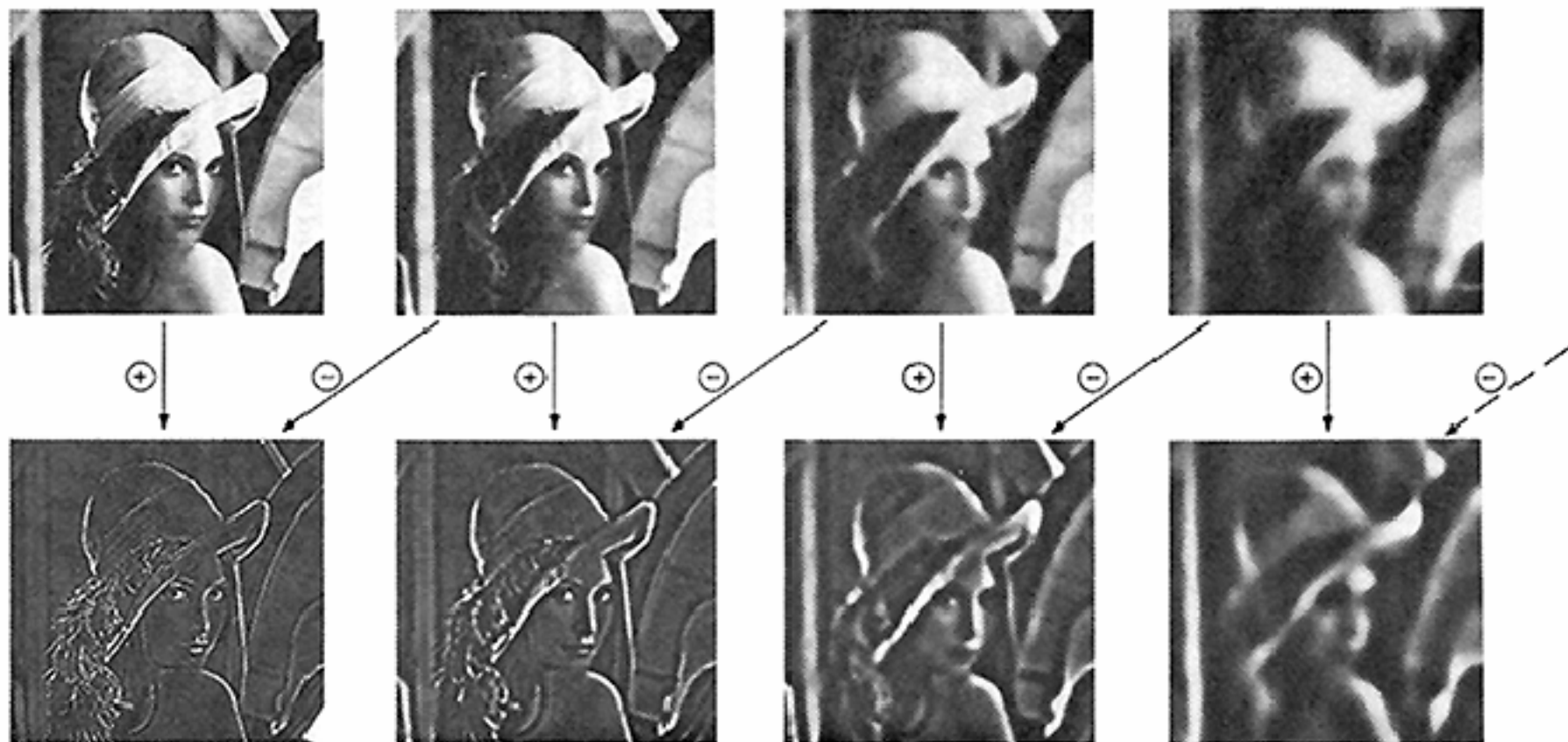


Fig. 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.



512

256

128

64

32

16

8





512

256

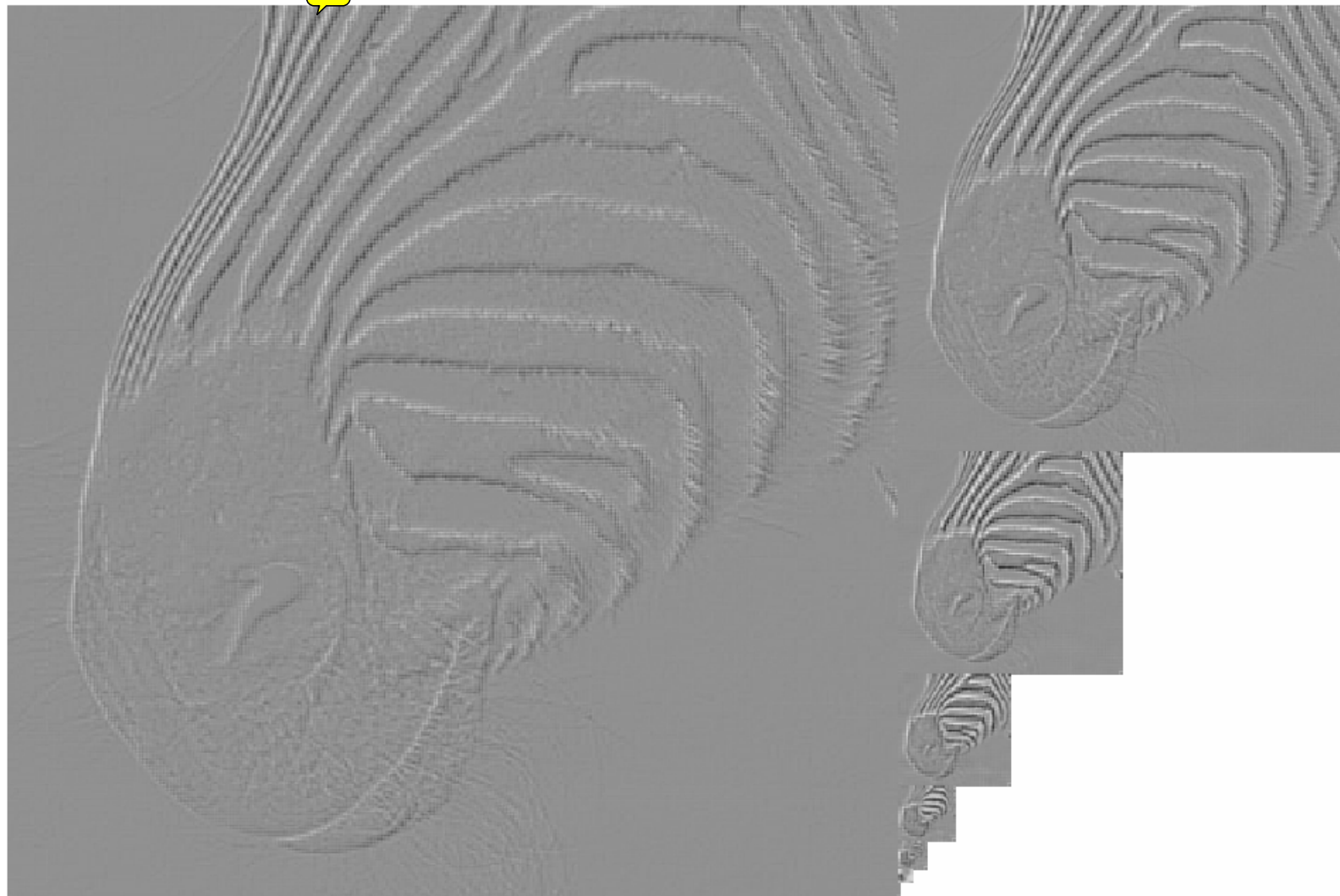
128

64

32

16

8



Application to image compression

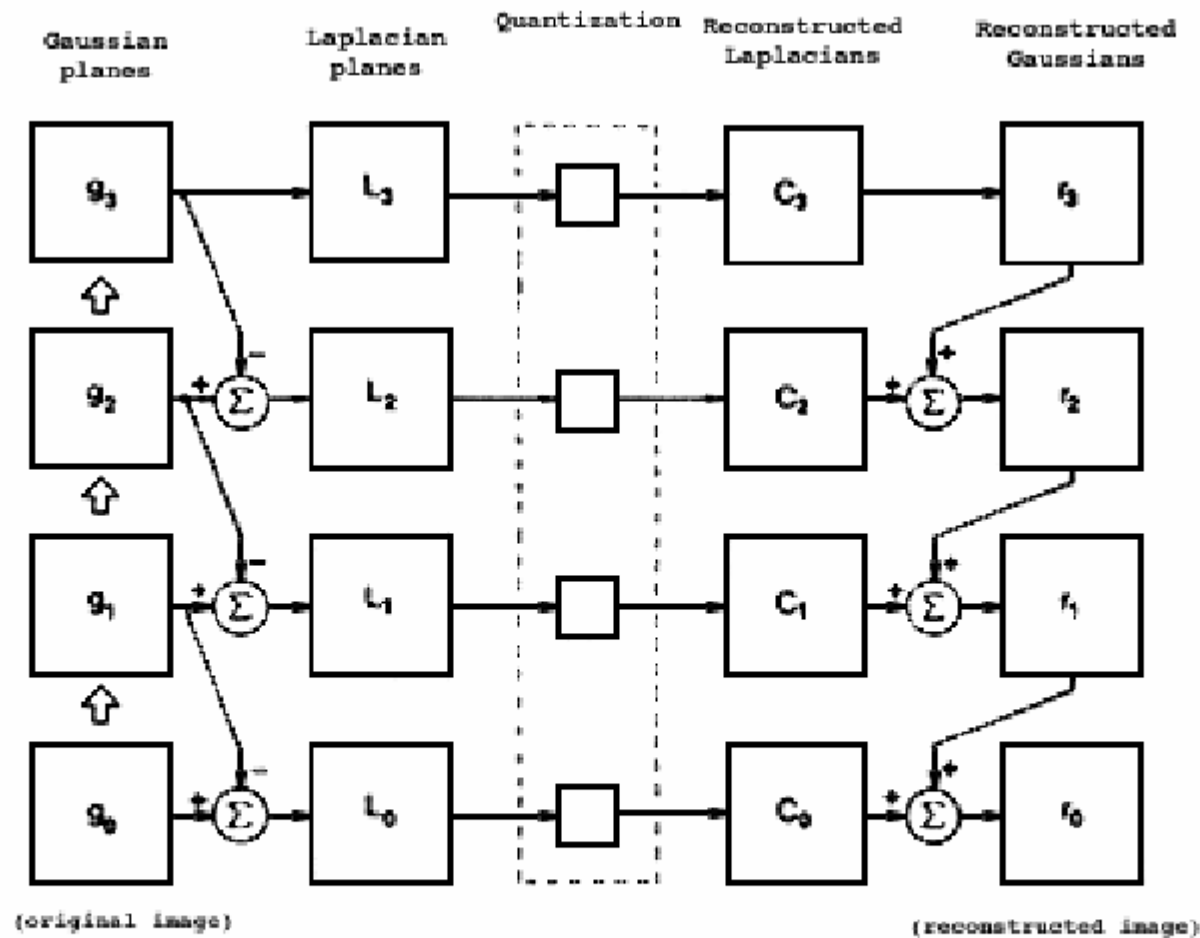


Fig. 10. A summary of the steps in Laplacian pyramid coding and decoding. First, the original image g_0 (lower left) is used to generate Gaussian pyramid levels g_1, g_2, \dots through repeated local averaging. Levels of the Laplacian pyramid L_0, L_1, \dots are then computed as the differences between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

Matlab manipulations with gaussian and laplacian pyramids

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

$$U =$$

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

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


```
>> inv(U)
```

```
ans =
```

```
0.5000    0.5000
```

```
0.5000   -0.5000
```

U =

$$\begin{array}{cccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{array}$$

```
>> 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

Matlab examples of Haar wavelet representation

- Frequency characteristics of the high and low-pass representations

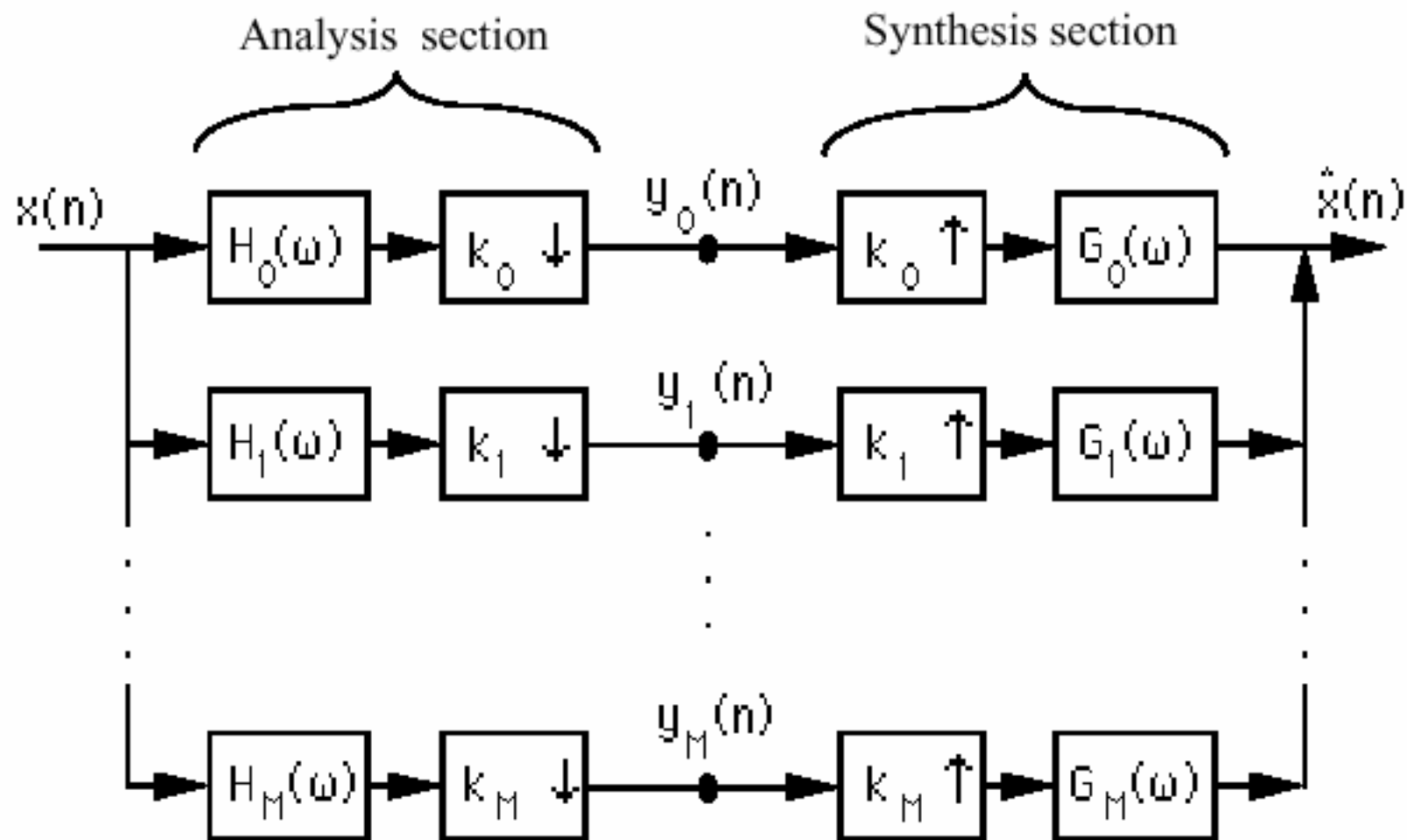


Figure 4.2: An analysis/synthesis filter bank.

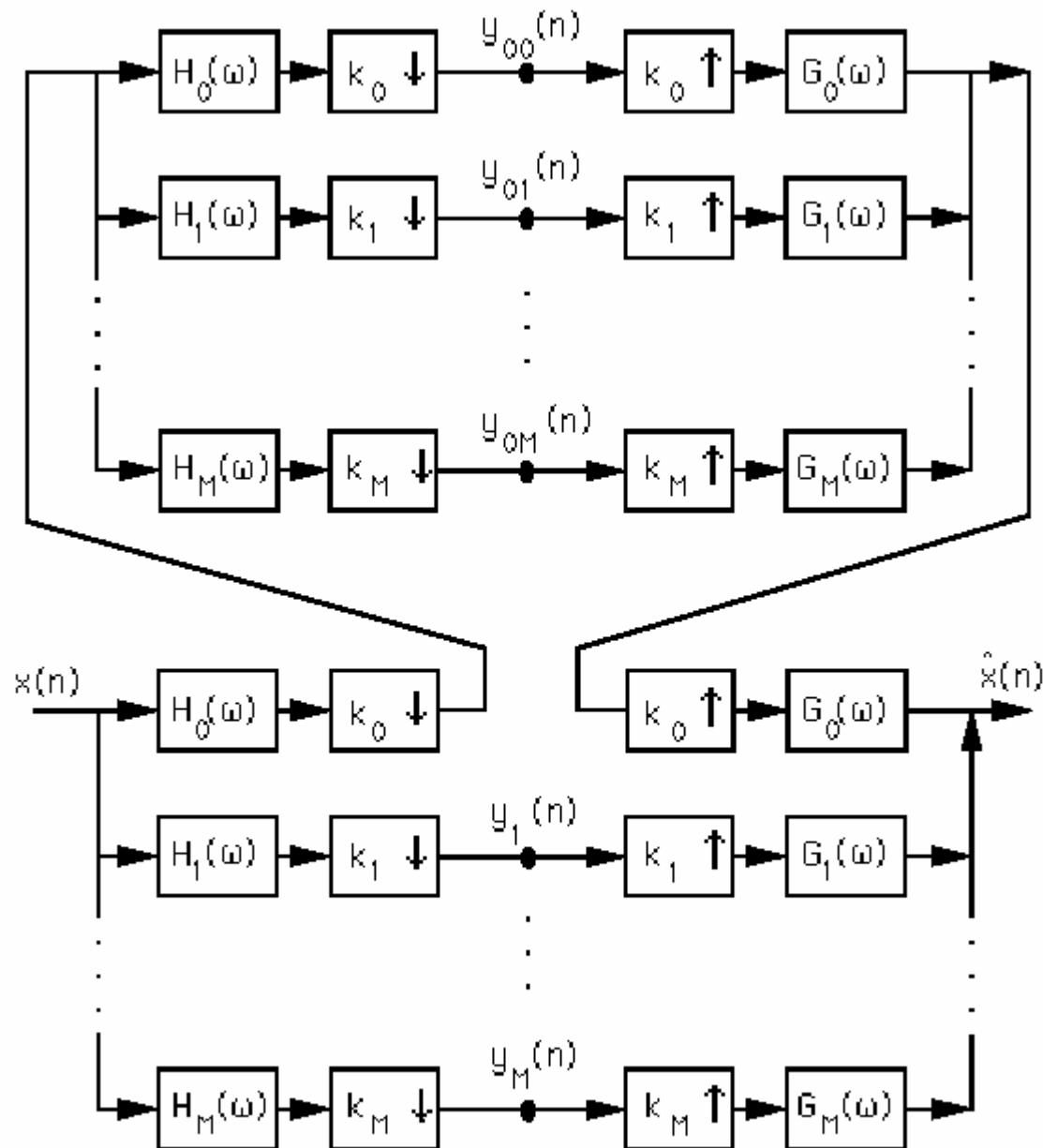


Figure 4.3: A non-uniformly cascaded analysis/synthesis filter bank.

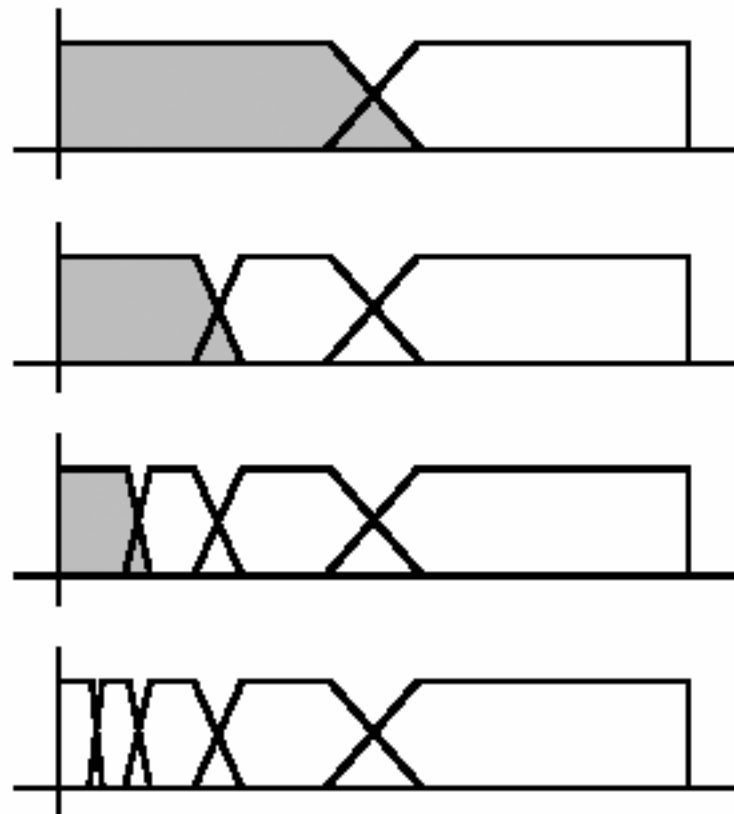
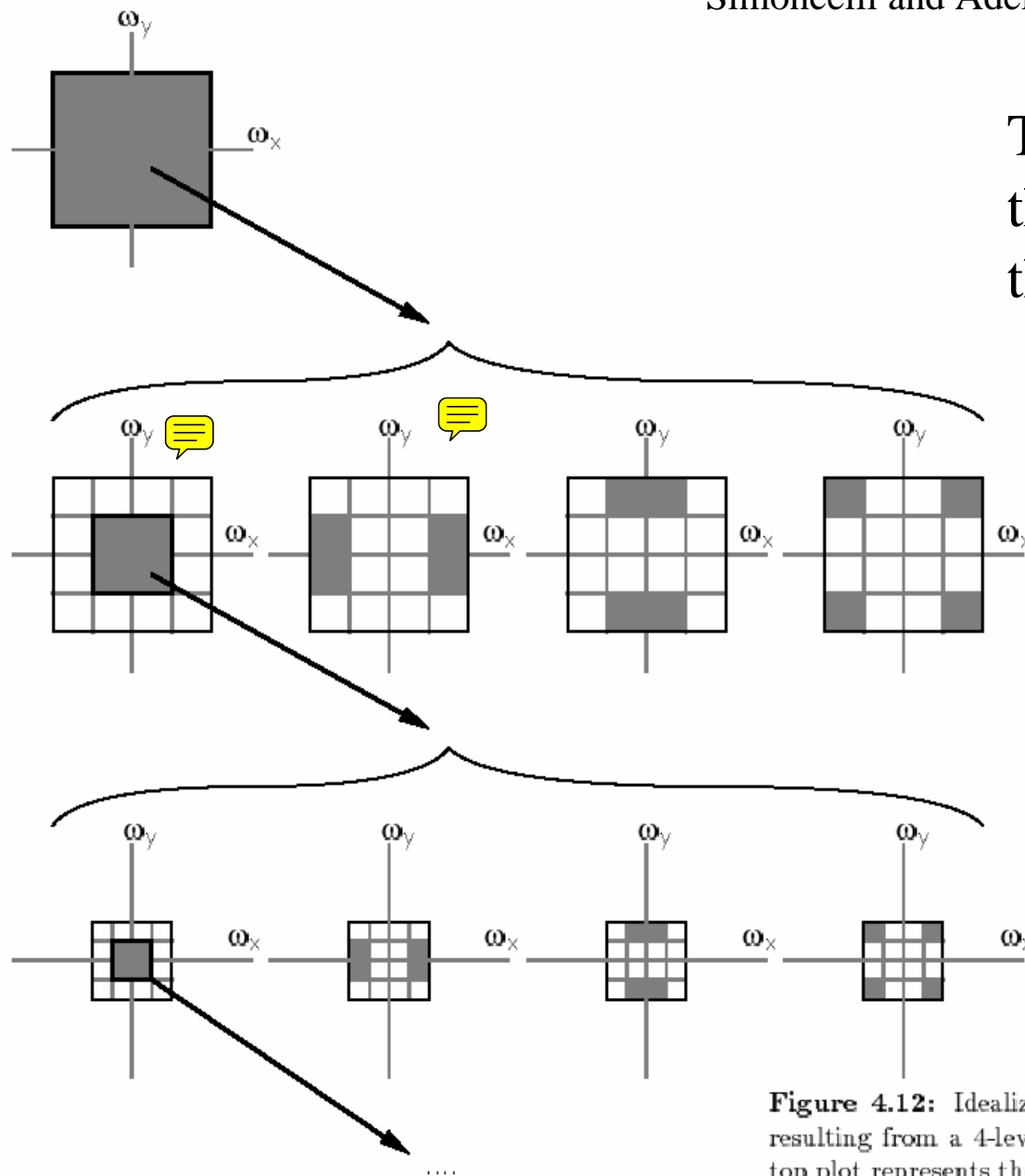


Figure 4.4: Octave band splitting produced by a four-level pyramid cascade of a two-band A/S system. The top picture represents the splitting of the two-band A/S system. Each successive picture shows the effect of re-applying the system to the lowpass subband (indicated in grey) of the previous picture.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

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

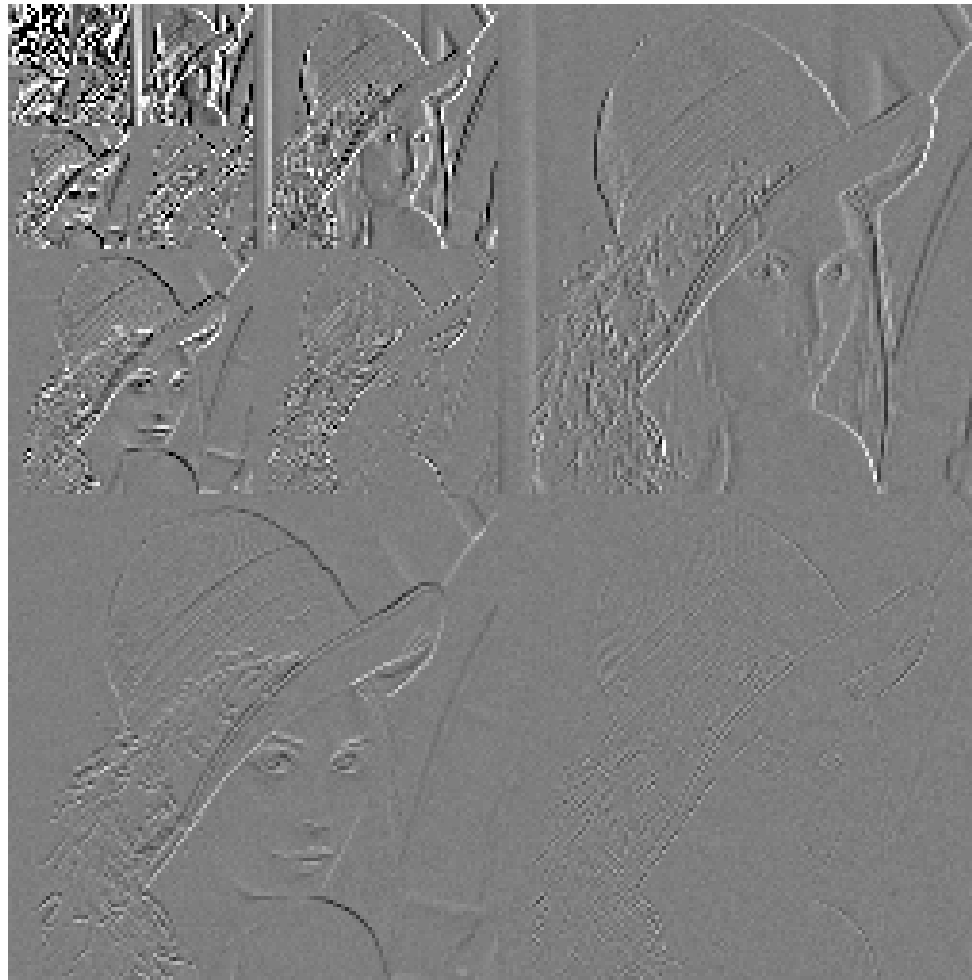
Table 4.1: Odd-length QMF kernels. Half of the impulse response sample values are shown for each of the normalized lowpass QMF filters (All filters are symmetric about $n = 0$). The appropriate highpass filters are obtained by delaying by one sample and multiplying with the sequence $(-1)^n$.



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

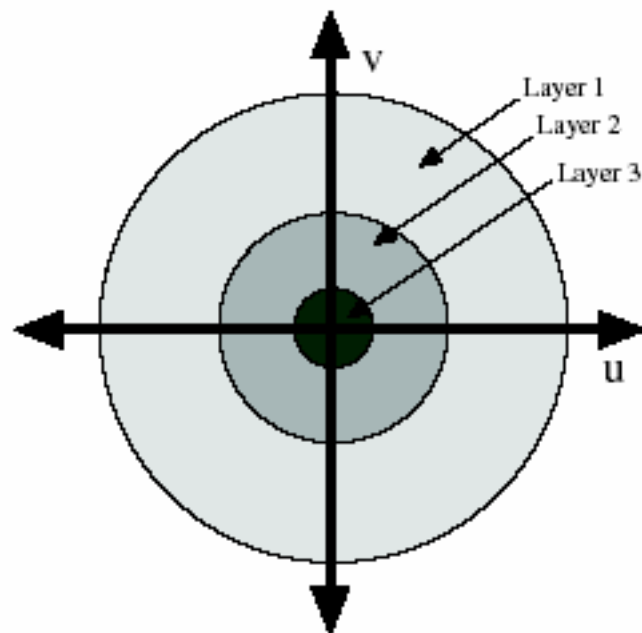
- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid

Steerable pyramids

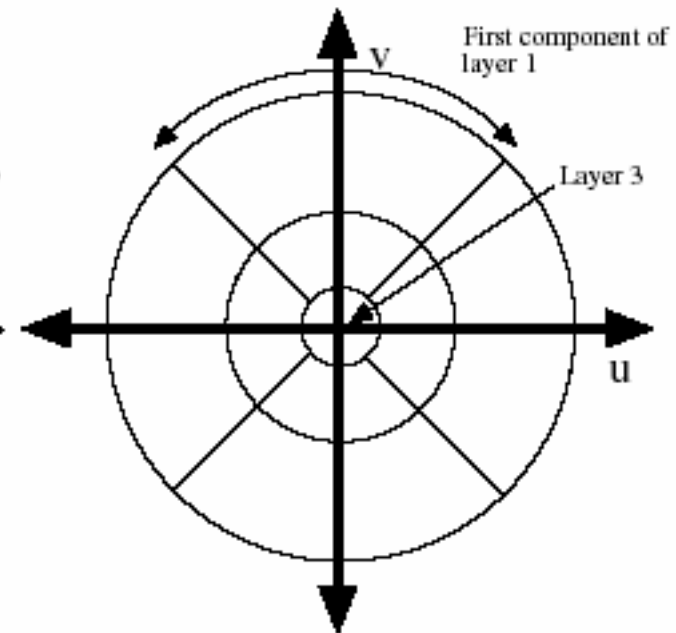
- Good:
 - Oriented subbands
 - Non-aliased subbands
 - Steerable filters
- Bad:
 - Overcomplete
 - Have one high frequency residual subband, required in order to form a circular region of analysis in frequency from a square region of support in frequency.

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.

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

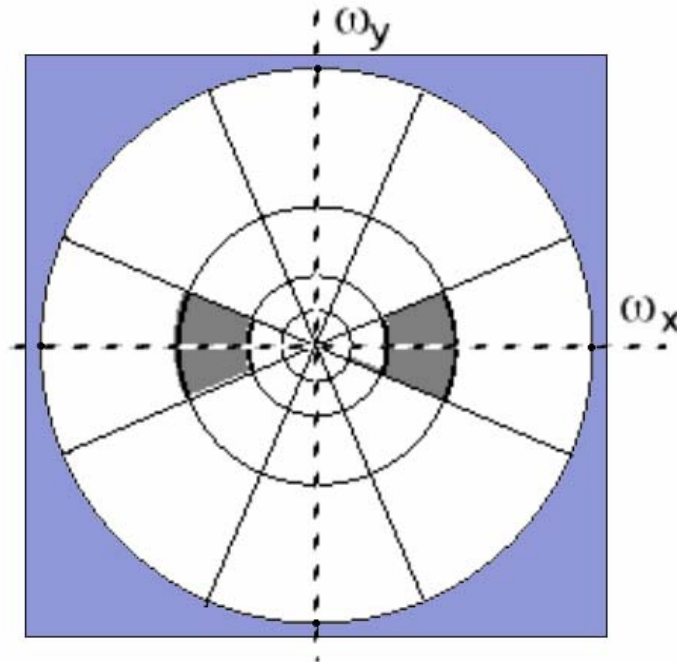
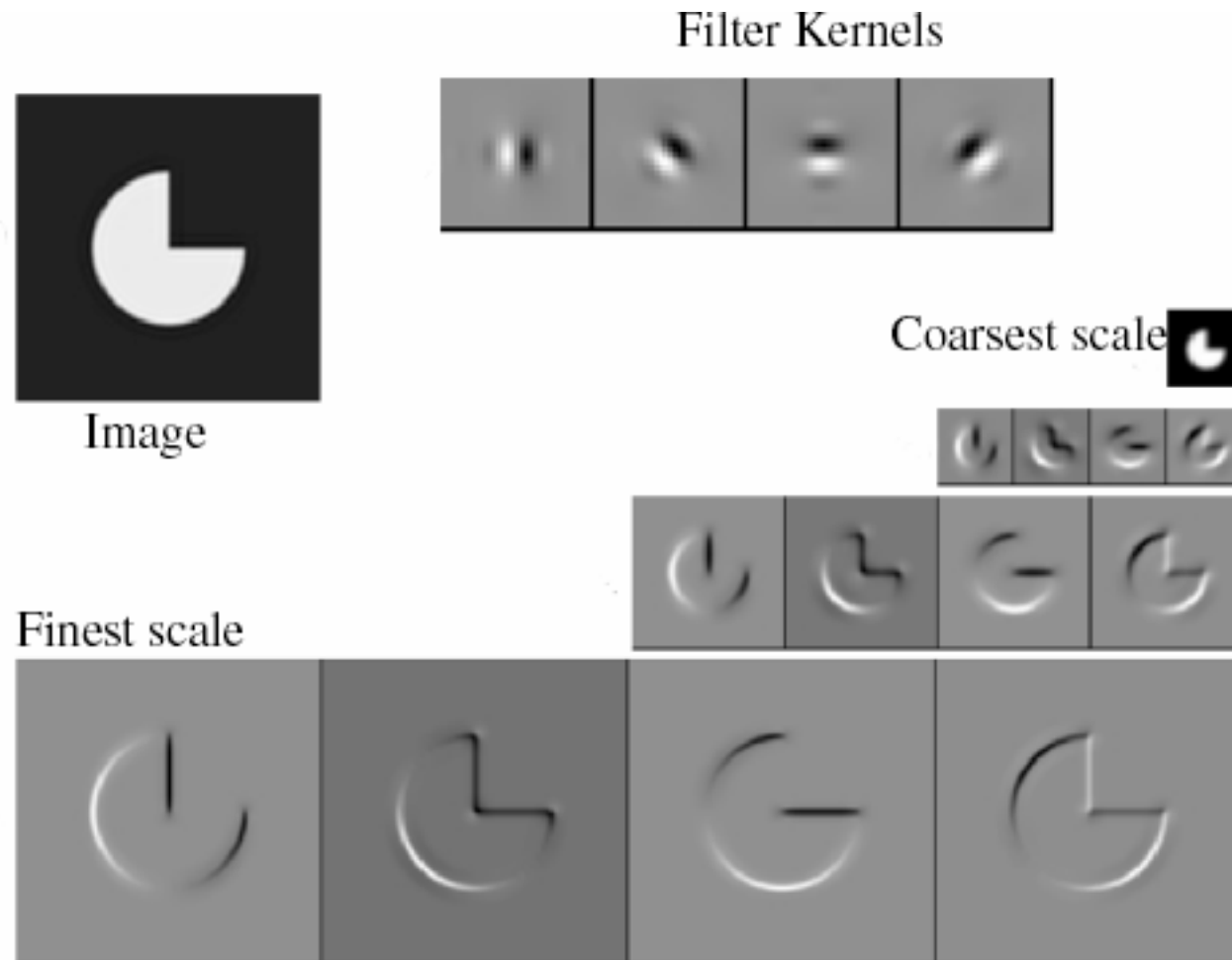


Figure 1. Idealized illustration of the spectral decomposition performed by a steerable pyramid with $k = 4$. Frequency axes range from $-\pi$ to π . The basis functions are related by translations, dilations and *rotations* (except for the initial highpass subband and the final low-pass subband). For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



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](#)

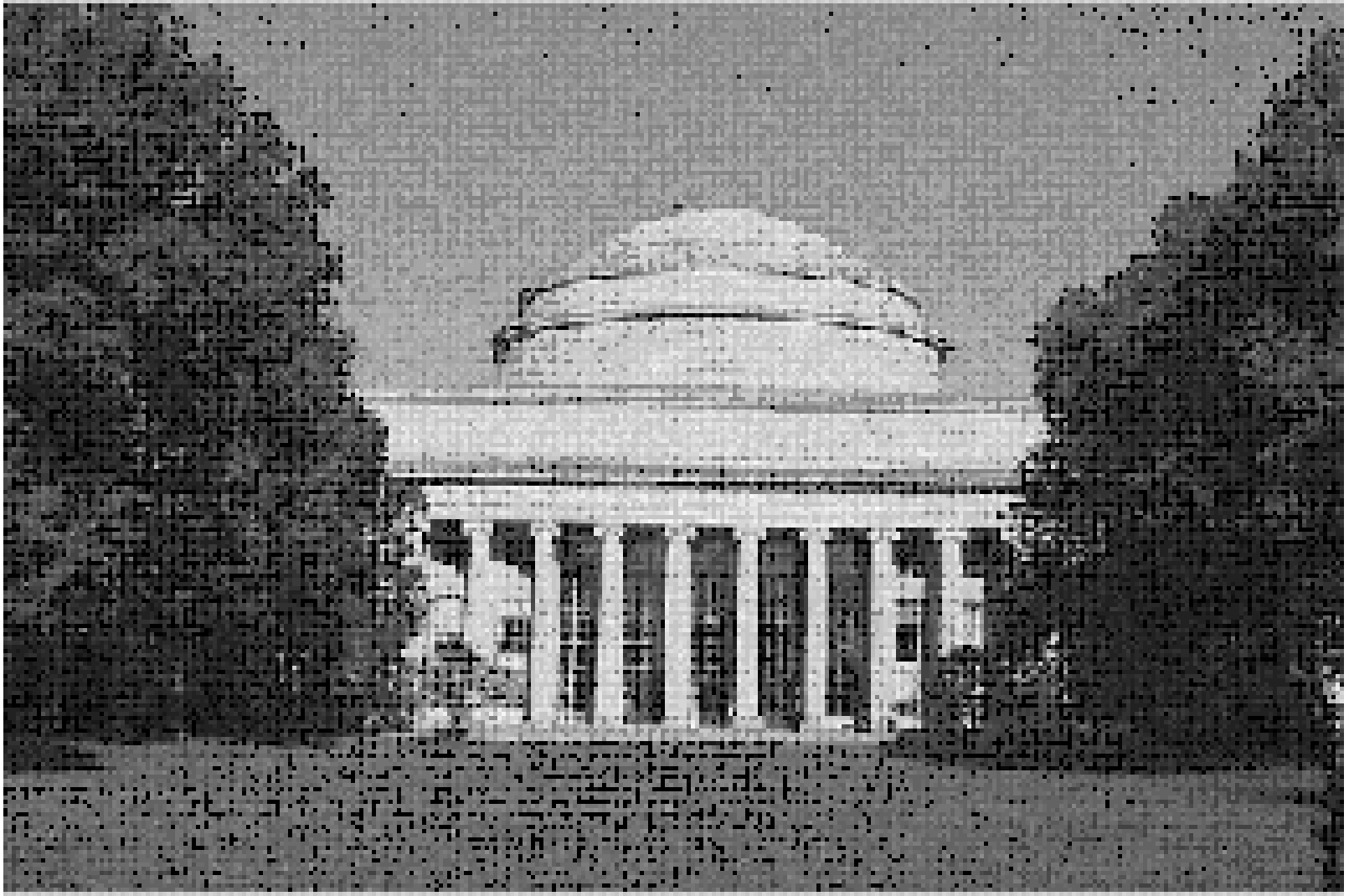
[Software](#)

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 \(119k\)](#).

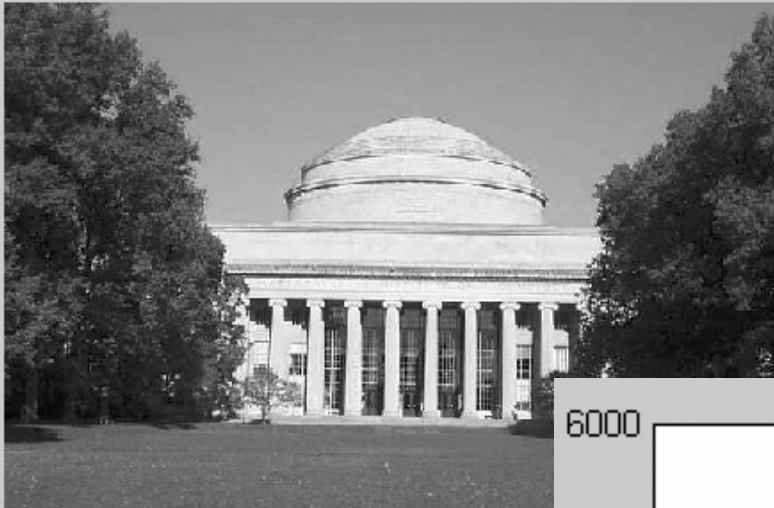
An application of image pyramids: noise removal

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



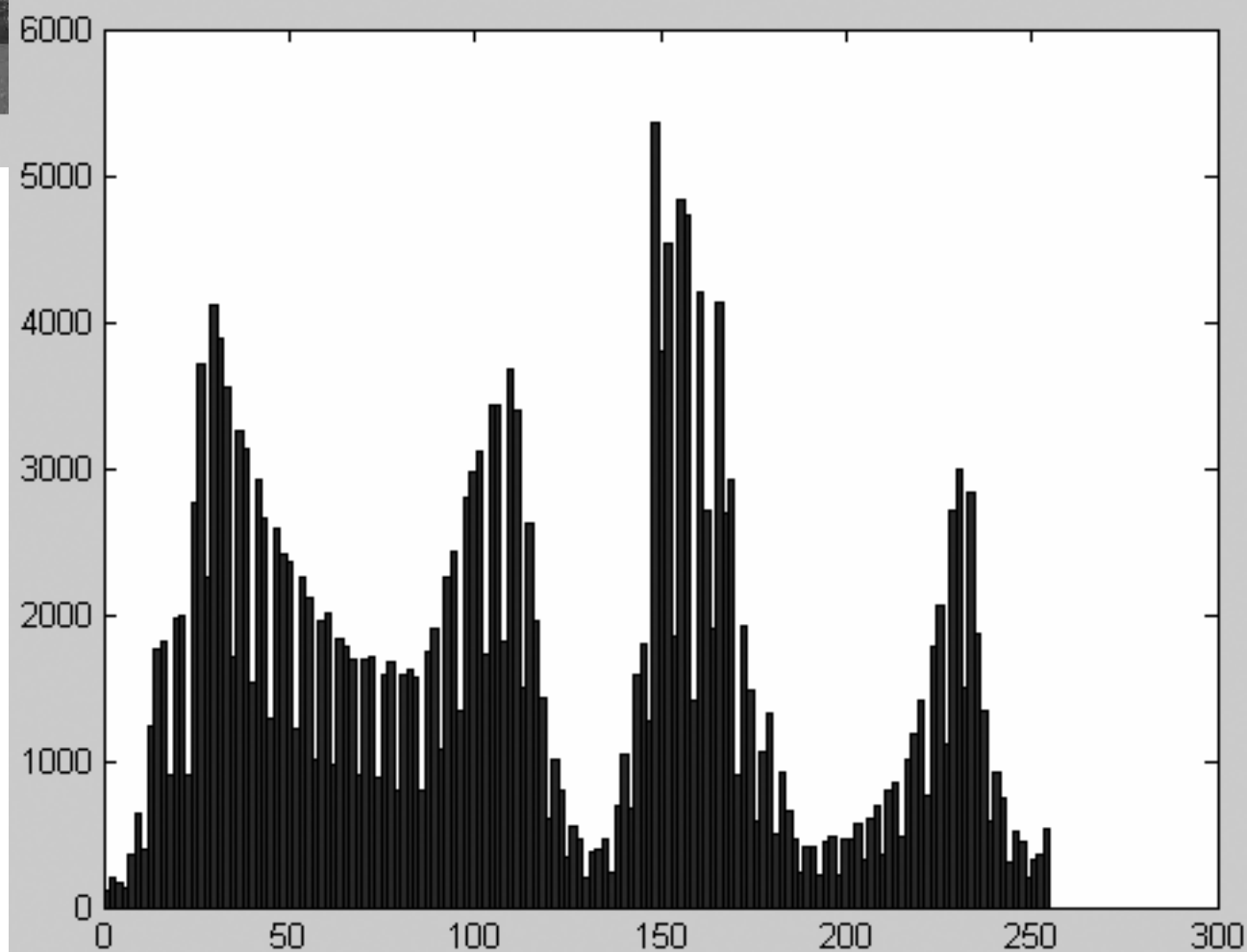


Range [0, 255]
Dims [394, 599]

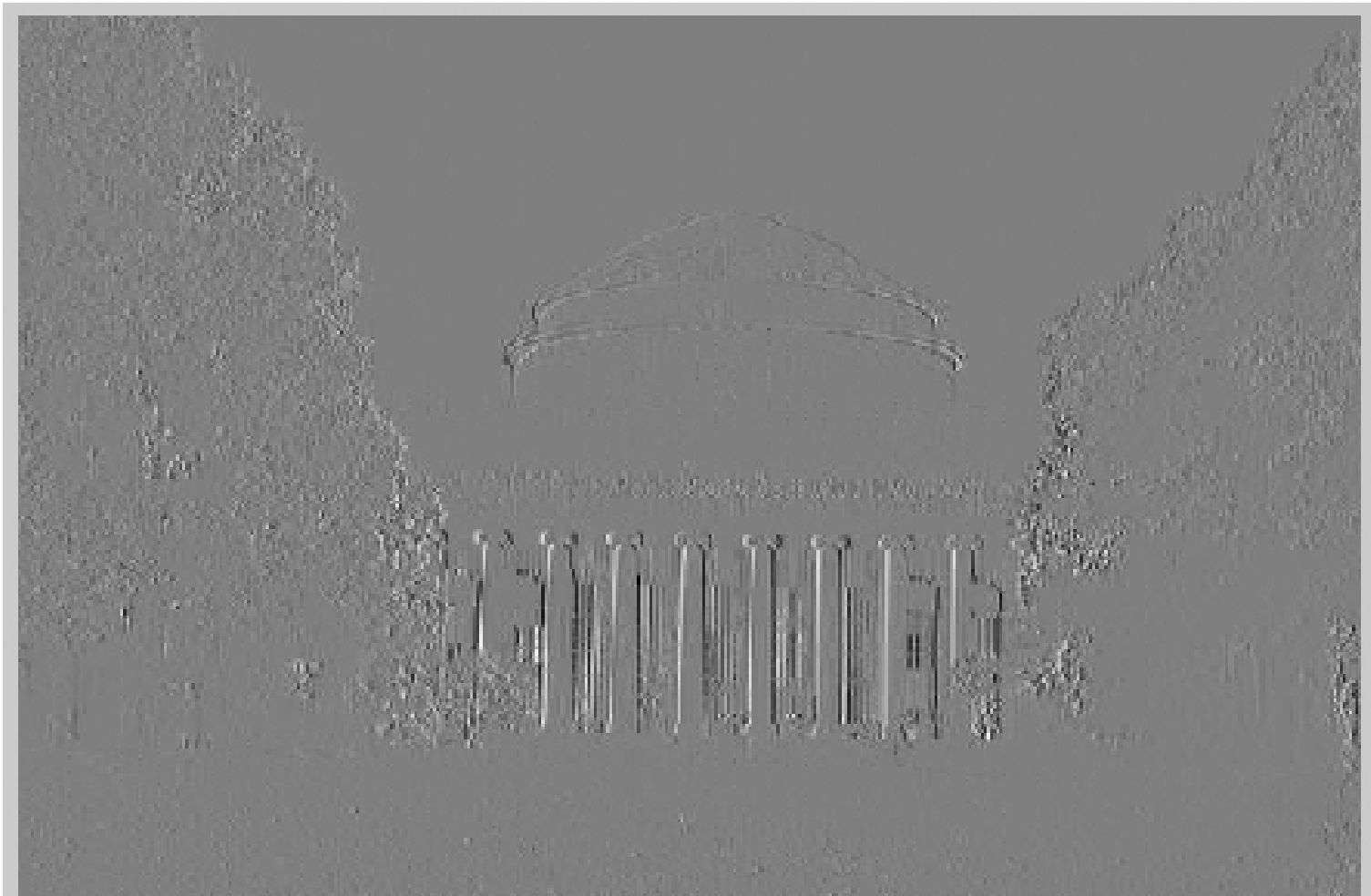


Range [0, 255]
Dims [394, 599]

Pixel representation image histogram



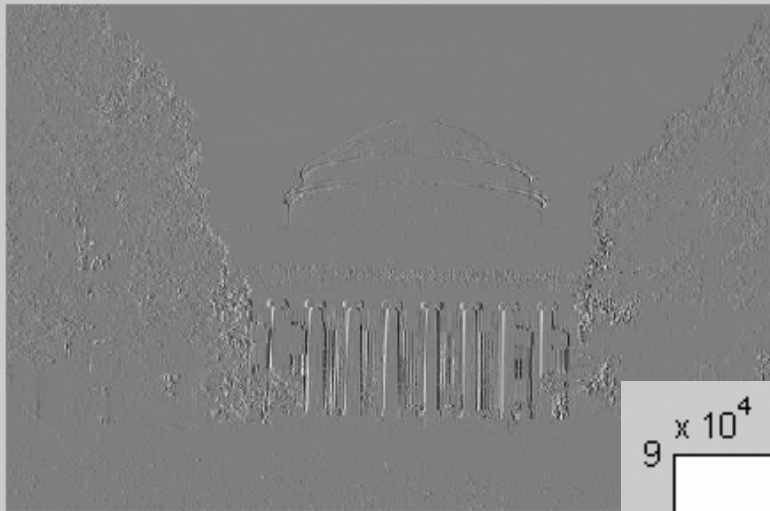
bandpass filtered image



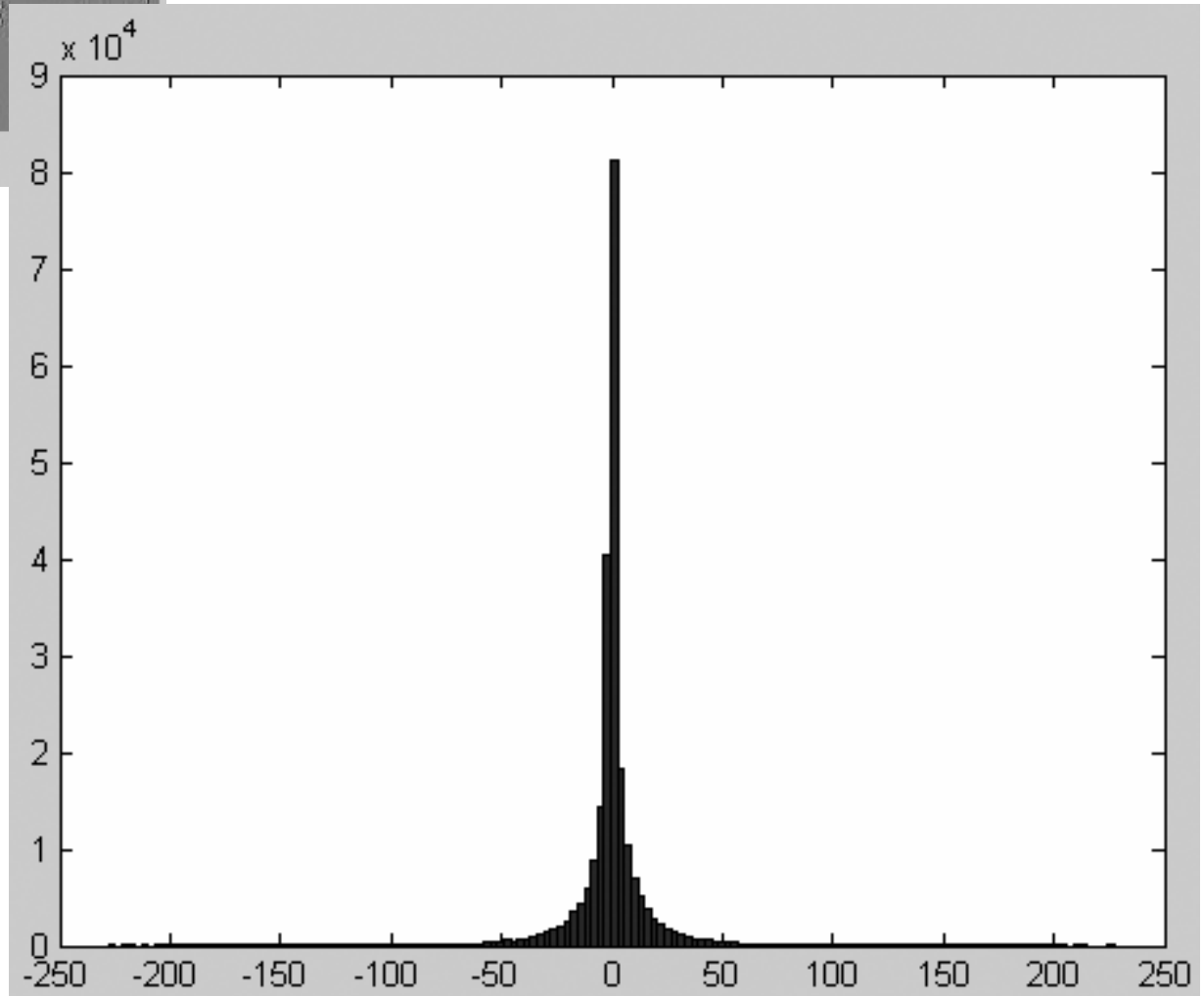
Range [-228, 227]

Dims [394, 598]

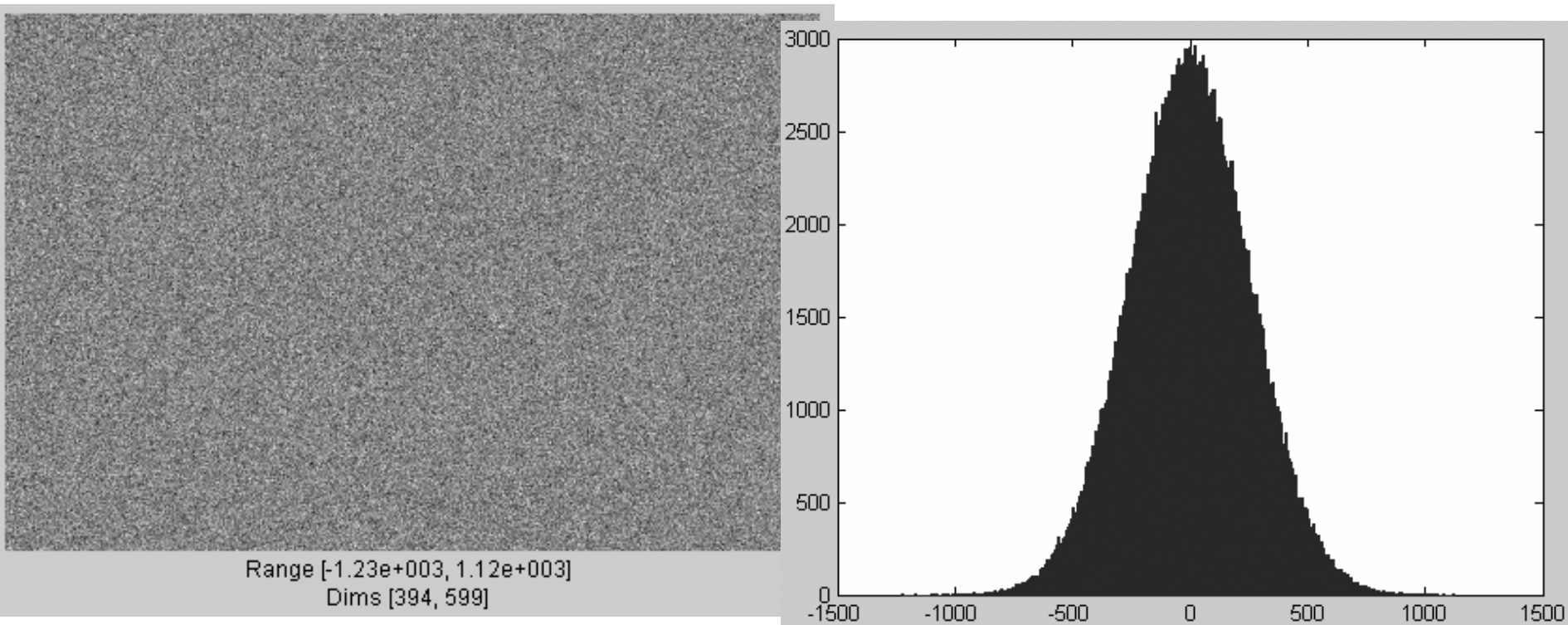
bandpassed representation image histogram



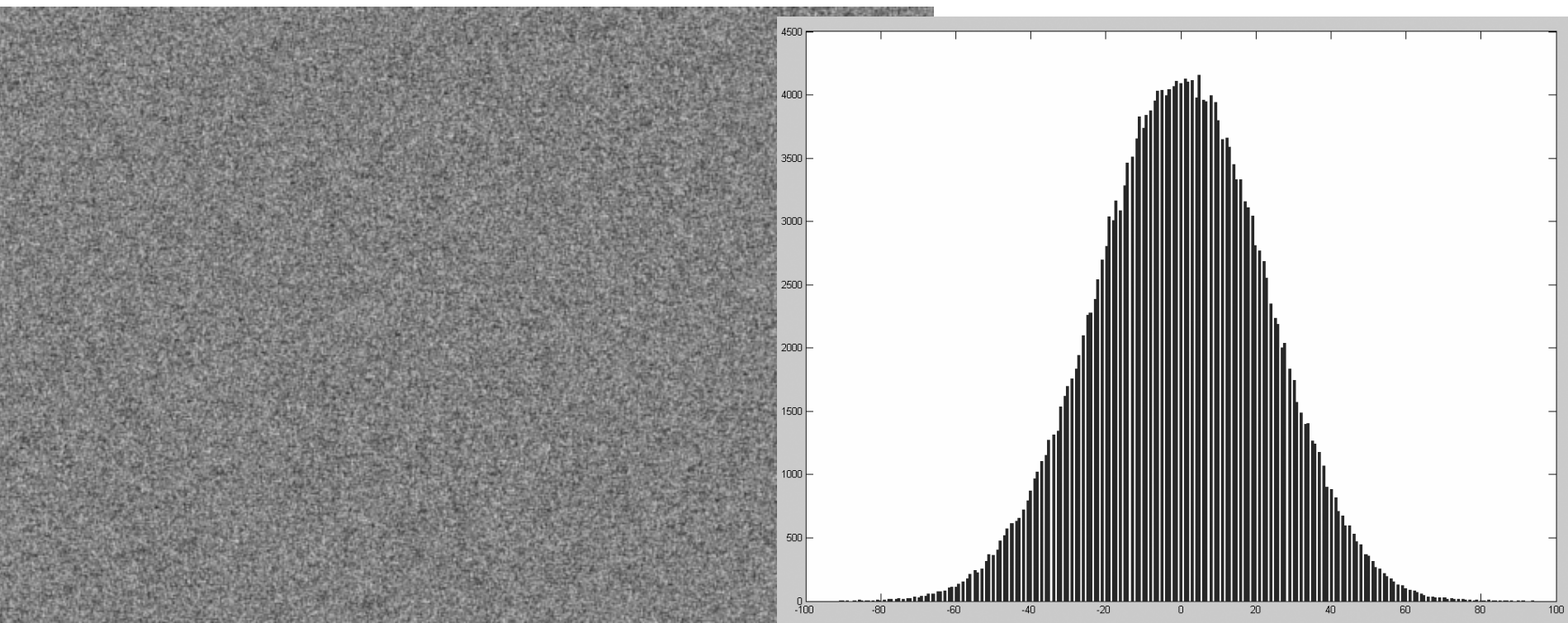
Range [-228, 227]
Dims [394, 598]



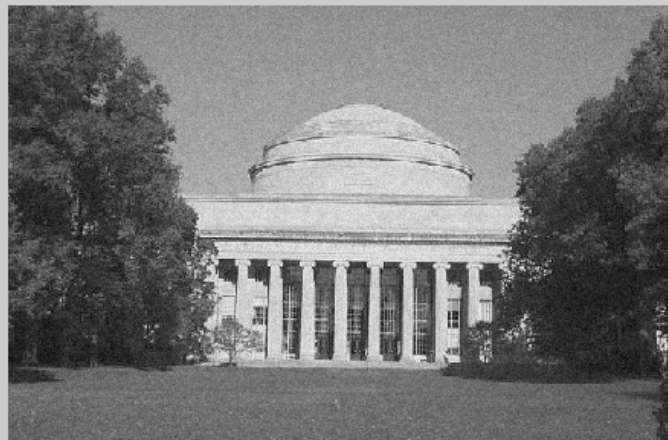
Pixel domain noise image and histogram



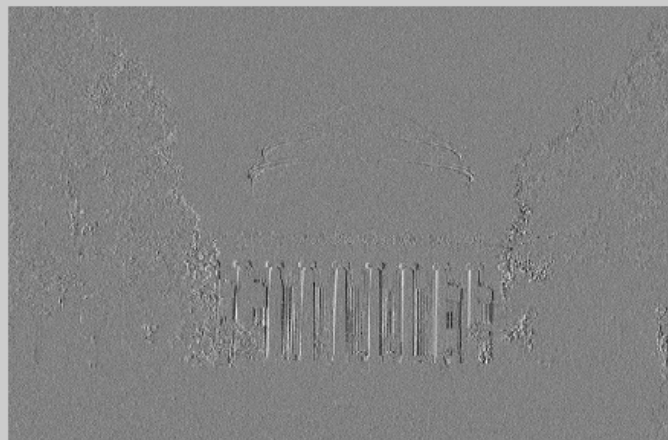
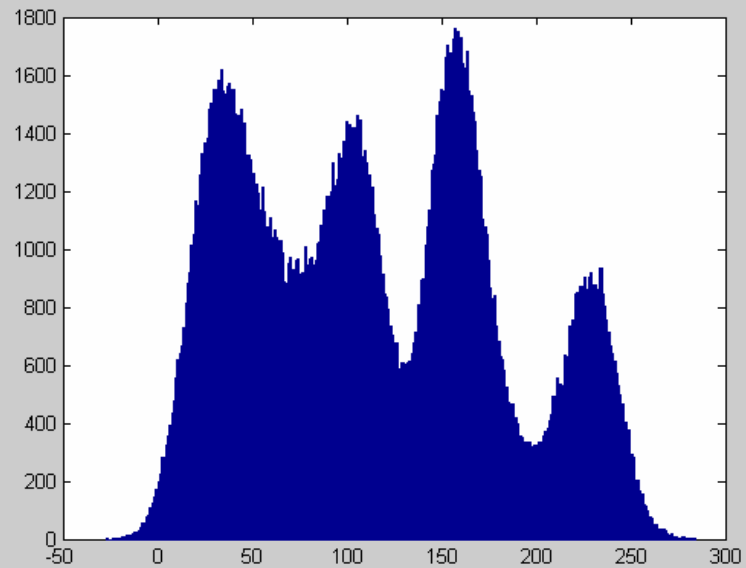
Bandpass domain noise image and histogram



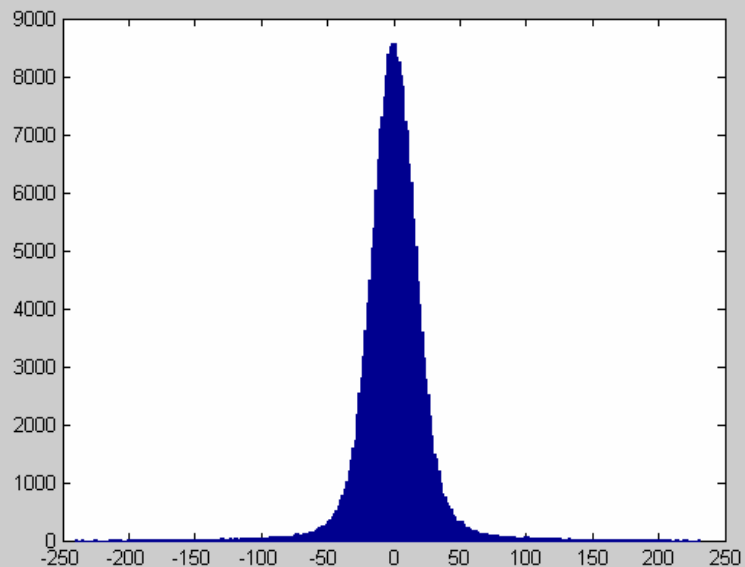
Noise-corrupted full-freq and bandpass images



Range [-27, 285]
Dims [394, 599]



Range [-240, 231]
Dims [394, 598]



Bayes theorem

$$P(\mathbf{x}, \mathbf{y}) = P(\mathbf{x}|\mathbf{y}) P(\mathbf{y})$$

so

$$P(\mathbf{x}|\mathbf{y}) P(\mathbf{y}) = P(\mathbf{y}|\mathbf{x}) P(\mathbf{x})$$

and

$$P(\mathbf{x}|\mathbf{y}) = P(\mathbf{y}|\mathbf{x}) P(\mathbf{x}) / P(\mathbf{y})$$

The parameters you
want to estimate

What you observe

Likelihood
function

Prior probability

Constant w.r.t.
parameters \mathbf{x} .

Bayesian MAP estimator for clean bandpass coefficient values

Let x = bandpassed image value before adding noise.

Let y = noise-corrupted observation.

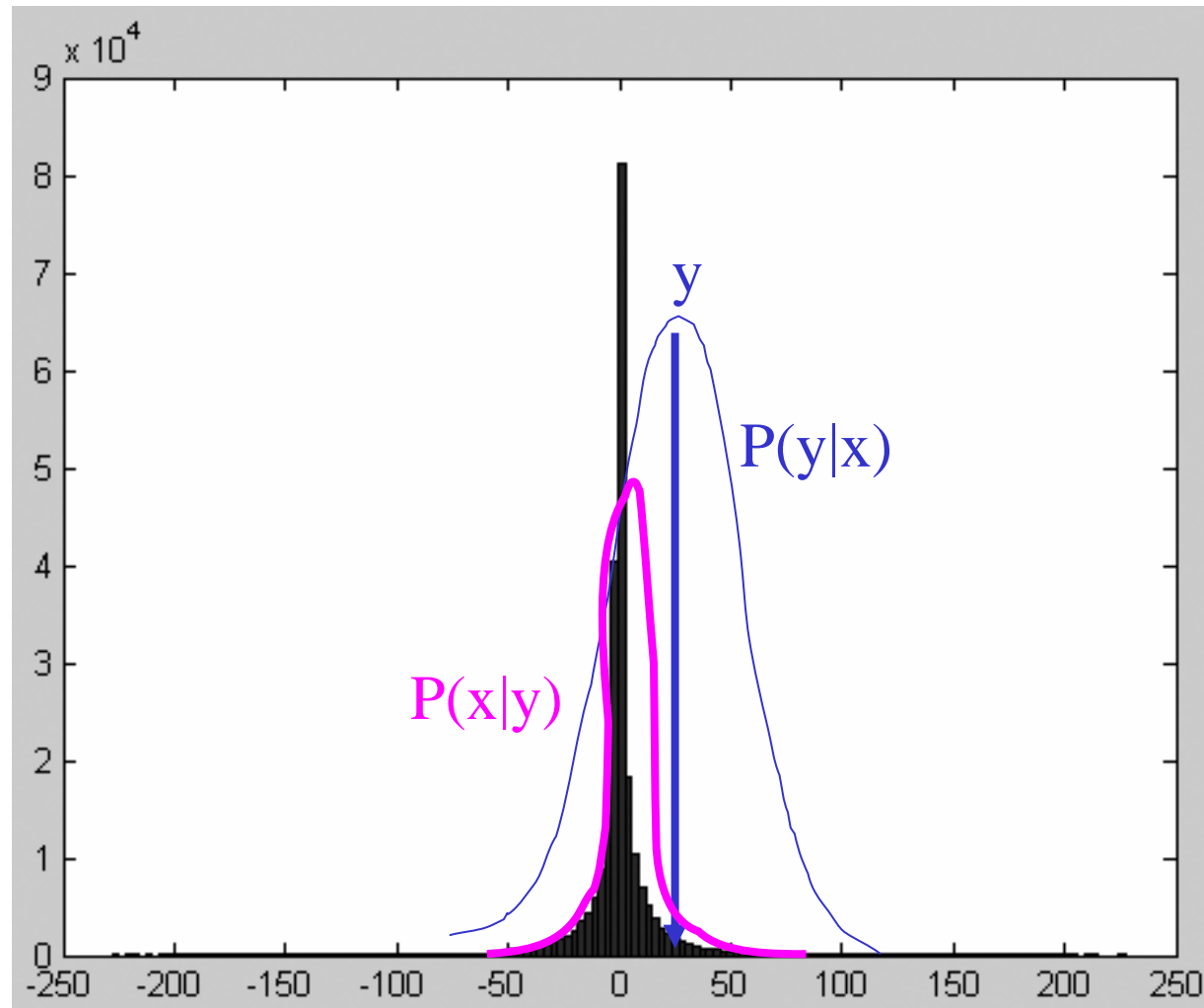
By Bayes theorem

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

$P(x)$

$P(y|x)$

$P(x|y)$



Bayesian MAP estimator

Let x = bandpassed image value before adding noise.

Let y = noise-corrupted observation.

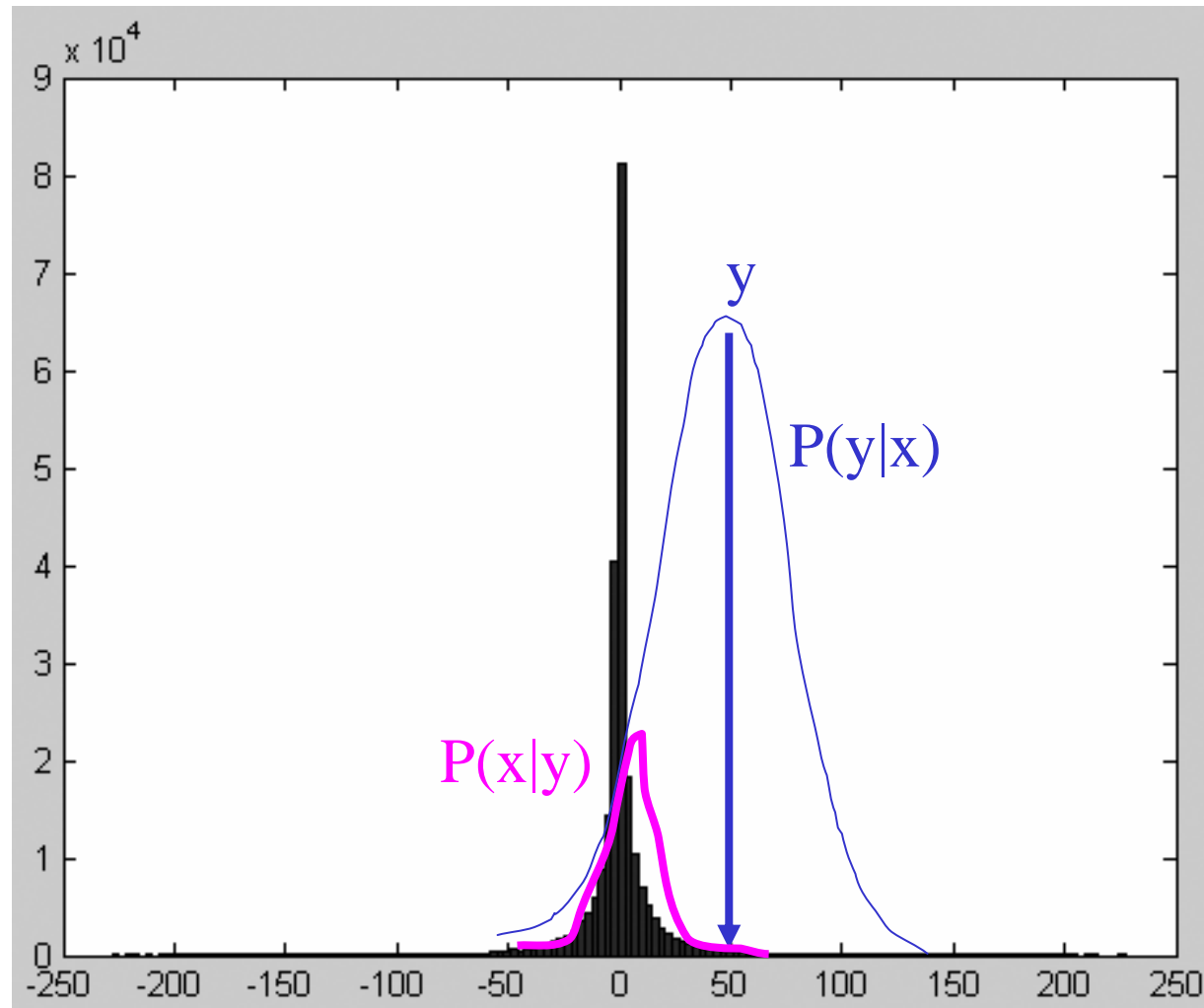
By Bayes theorem

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

$P(x)$

$P(y|x)$

$P(x|y)$



Bayesian MAP estimator

Let x = bandpassed image value before adding noise.

Let y = noise-corrupted observation.

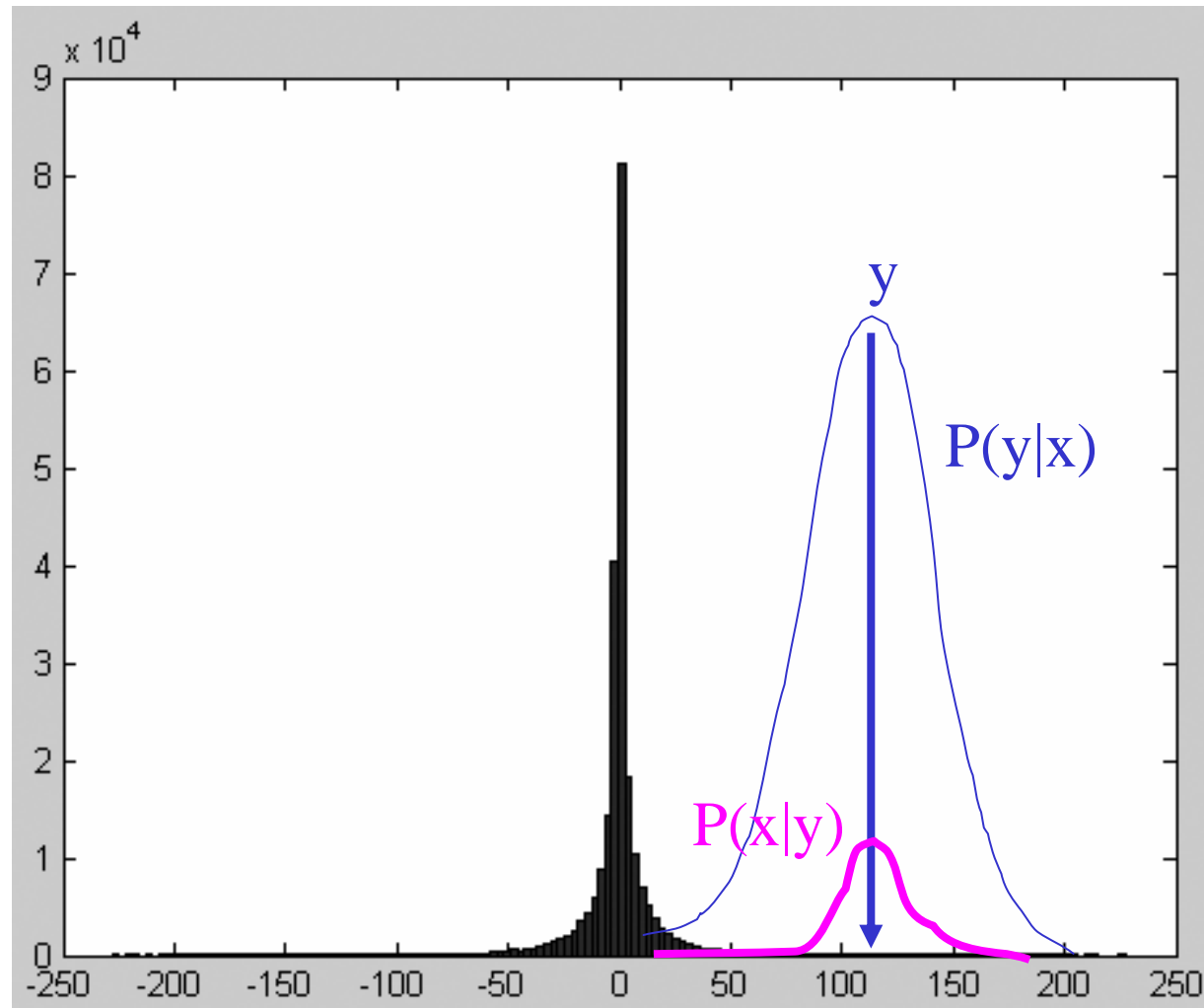
By Bayes theorem

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

$P(x)$

$P(y|x)$

$P(x|y)$



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

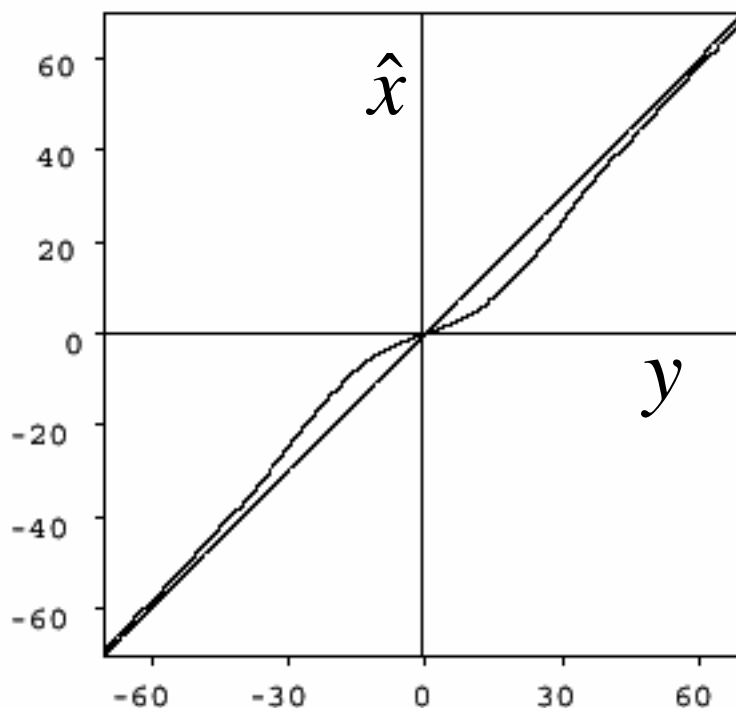


Figure 2: Bayesian estimator (symmetrized) for the signal and noise histograms shown in figure 1. Superimposed on the plot is a straight line indicating the identity function.

Noise removal results

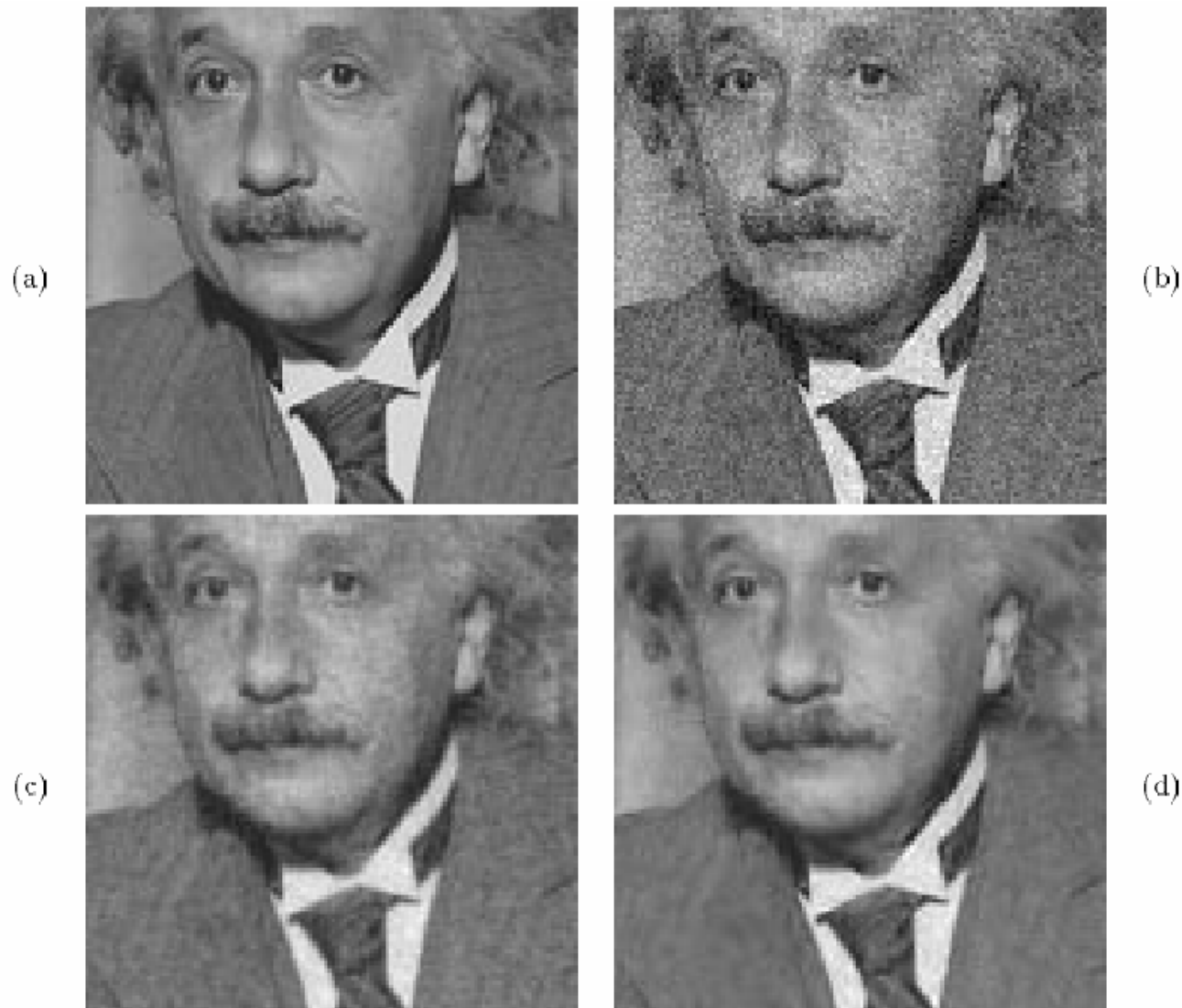


Figure 4: Noise reduction example. (a) Original image (cropped). (b) Image contaminated with additive Gaussian white noise ($\text{SNR} = 9.00\text{dB}$). (c) Image restored using (semi-blind) Wiener filter ($\text{SNR} = 11.88\text{dB}$). (d) Image restored using (semi-blind) Bayesian estimator ($\text{SNR}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

Simoncelli and Adelson, Noise Removal via Bayesian Wavelet Coring

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

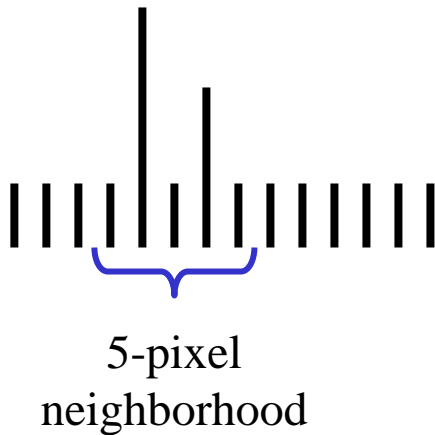
Non-linear filtering example

Median filter

Replace each pixel by the median over N pixels (5 pixels, for these examples).

Generalizes to “rank order” filters.

In:

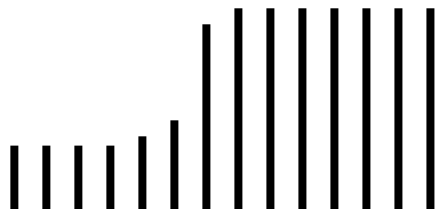


Out:

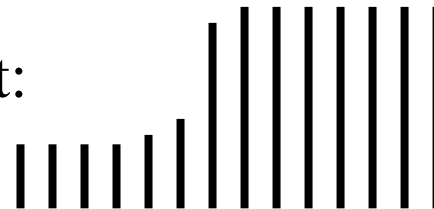


Spike
noise is
removed

In:

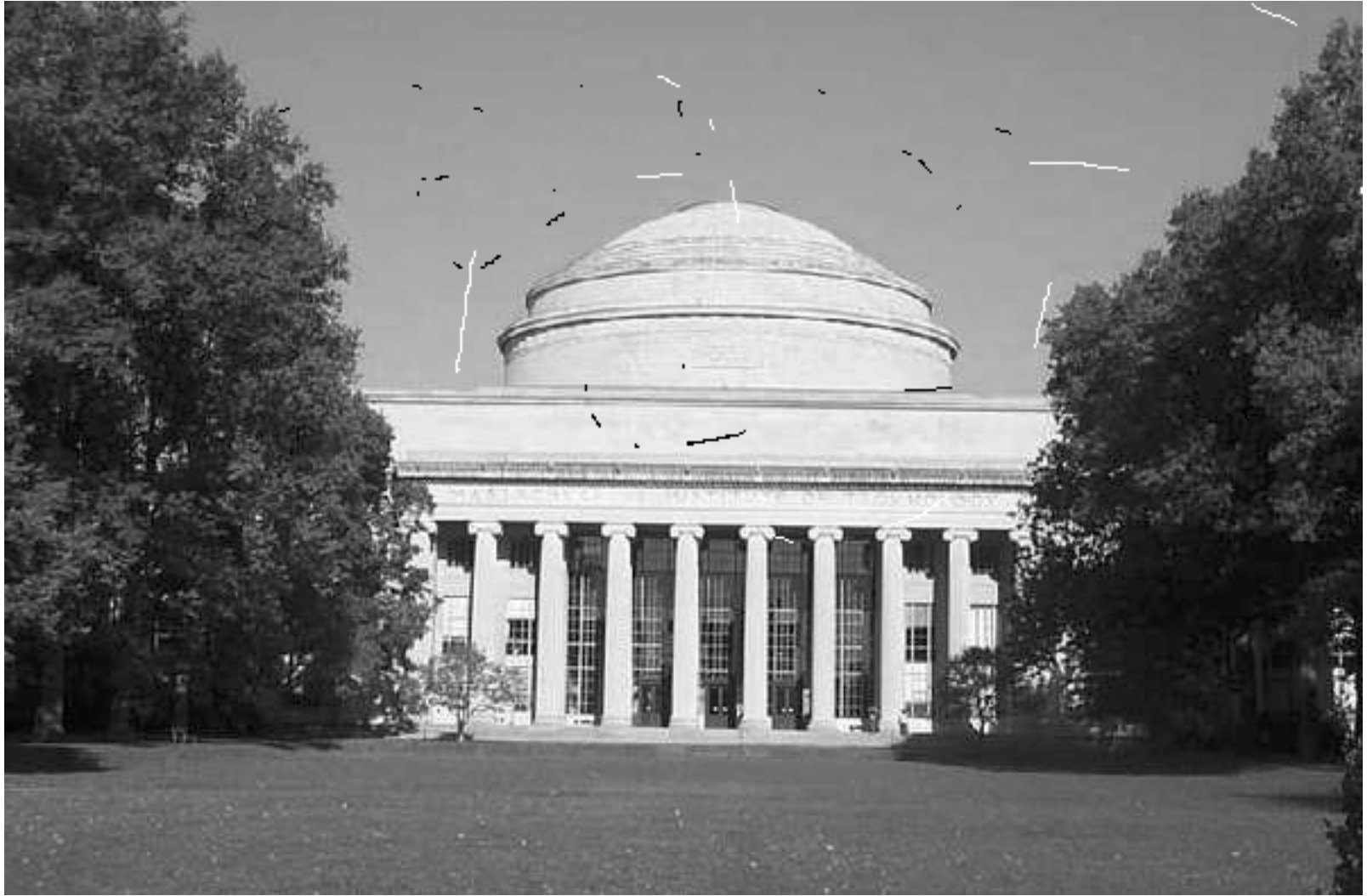


Out:



Monotonic
edges
remain
unchanged

Degraded image



Radius 1 median filter



Radius 2 median filter



CCD color sampling

Color sensing, 3 approaches

- Scan 3 times (temporal multiplexing)
- Use 3 detectors (3-ccd camera, and color film)
- Use offset color samples (spatial multiplexing)

Typical errors in temporal multiplexing approach

- Color offset fringes

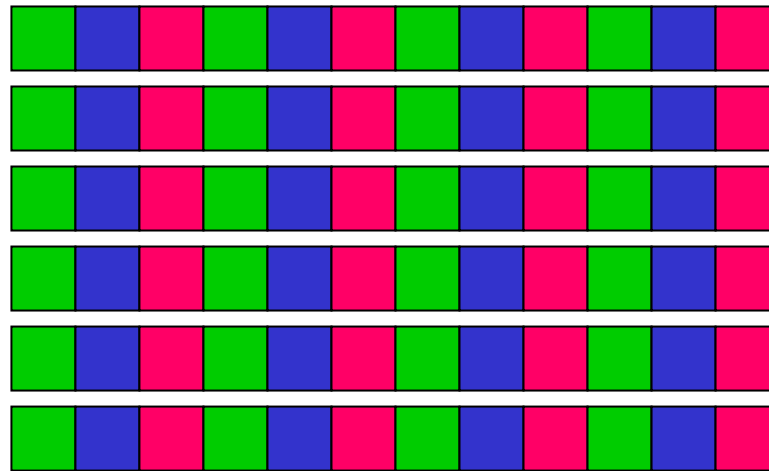


Typical errors in spatial multiplexing approach.

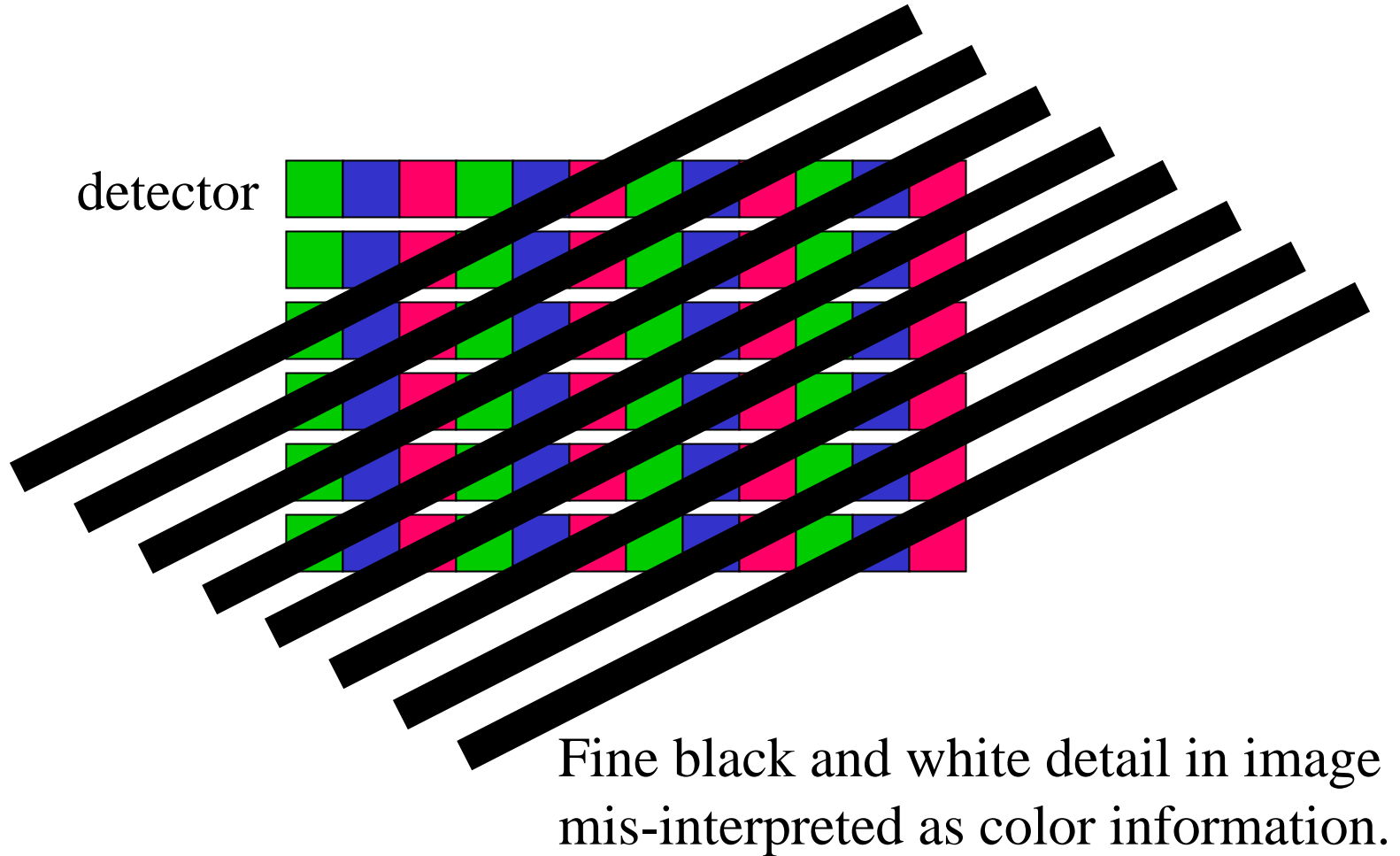
- Color fringes.

CCD color filter pattern

detector

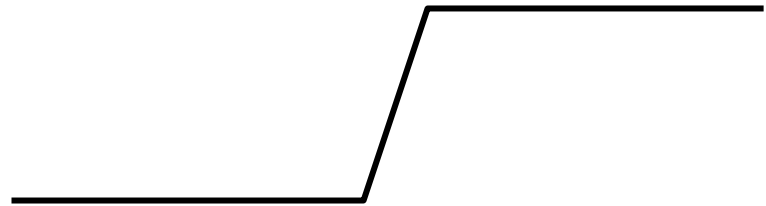


The cause of color moire



Black and white edge falling on color CCD detector

Black and white image (edge)

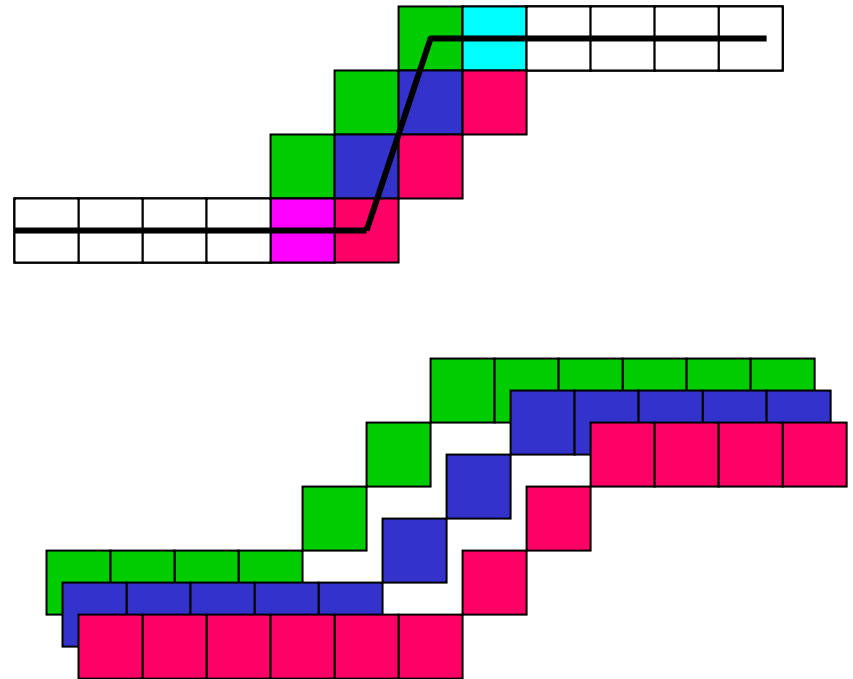


Detector pixel colors



Color sampling artifact

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



Typical color moire patterns

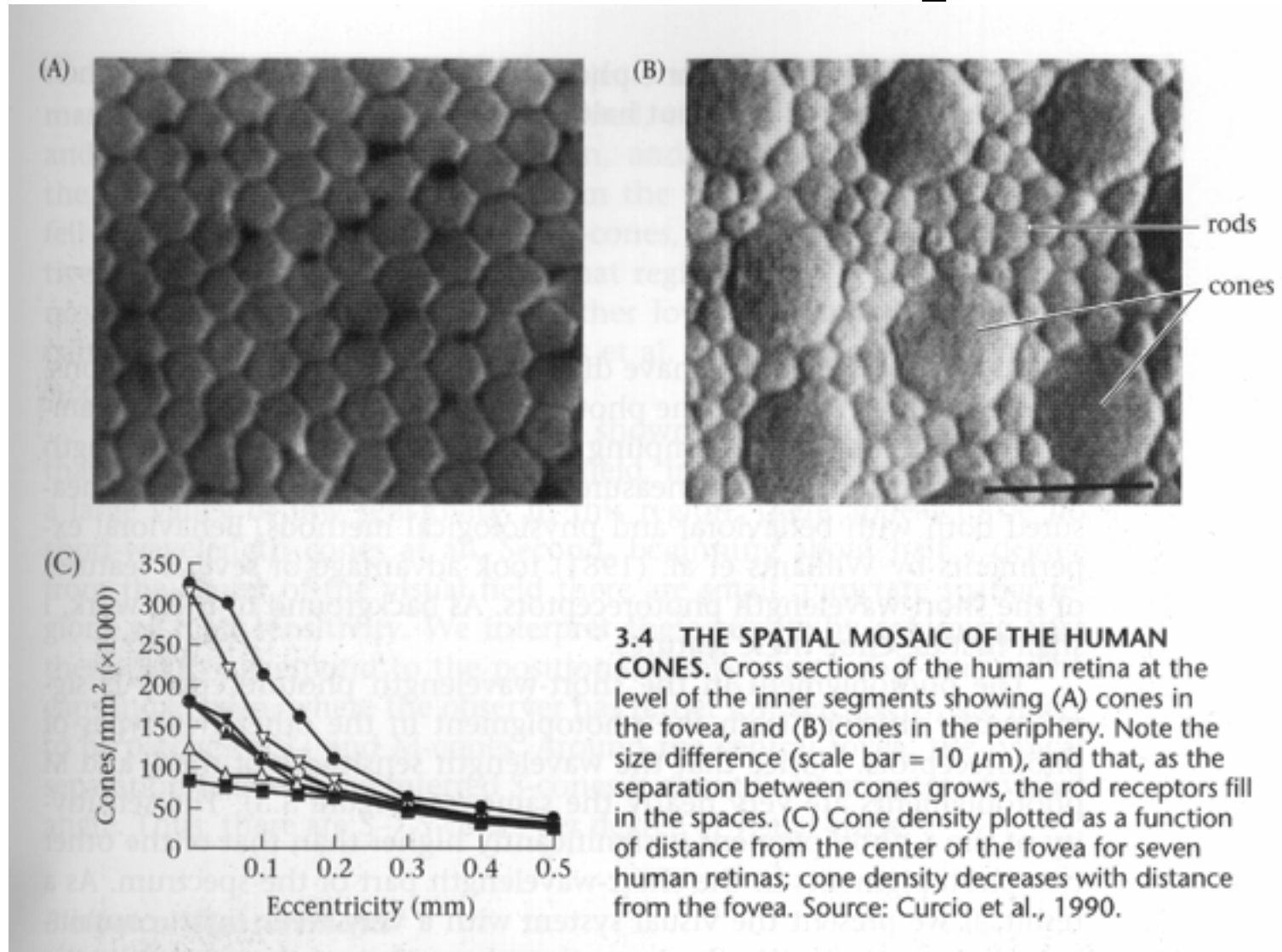


Blow-up of electronic camera image. Notice spurious colors in the regions of fine detail in the plants.

Color sampling artifacts



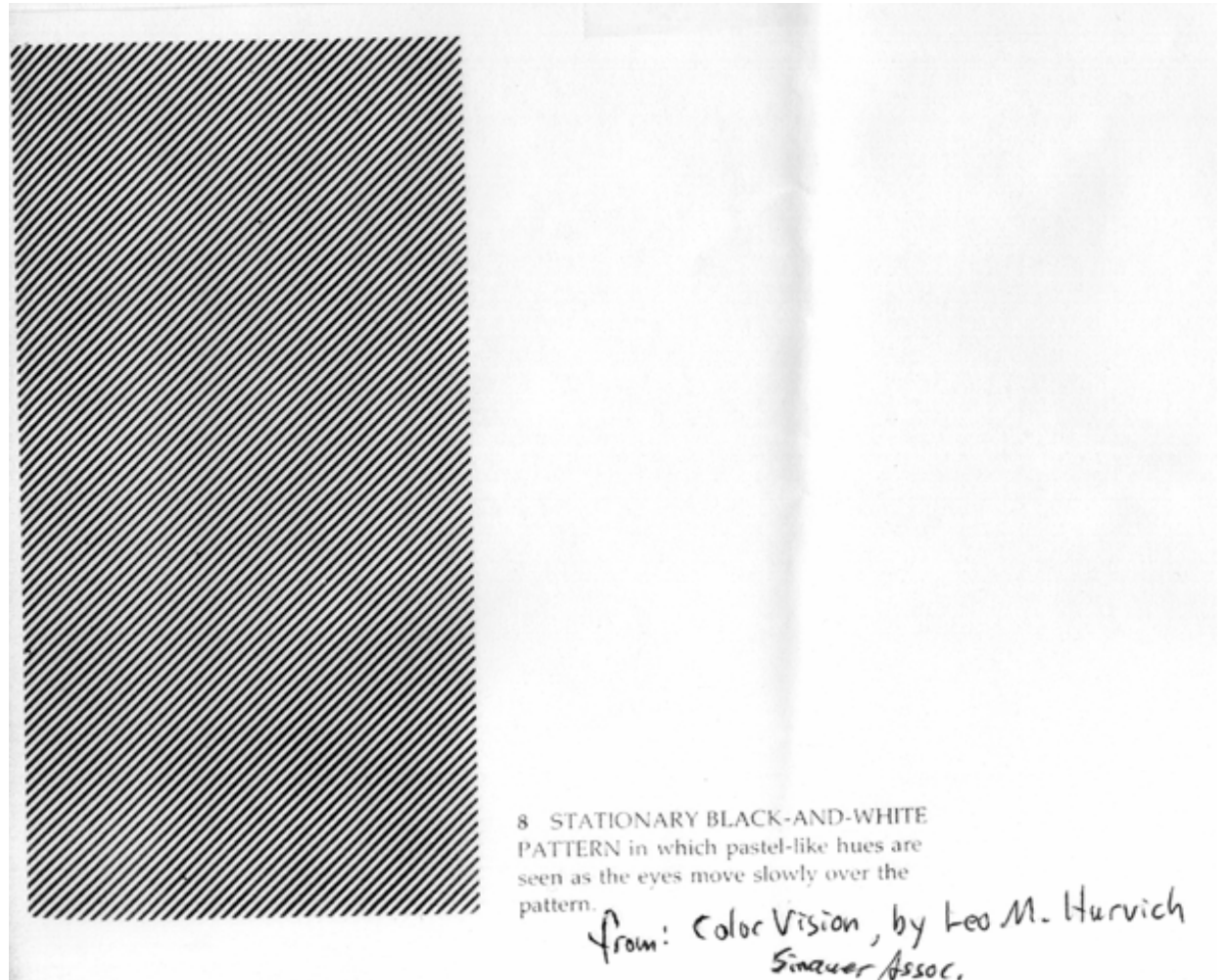
Human Photoreceptors



(From Foundations of Vision, by Brian Wandell, Sinauer Assoc.)

Brewster's colors example (subtle).

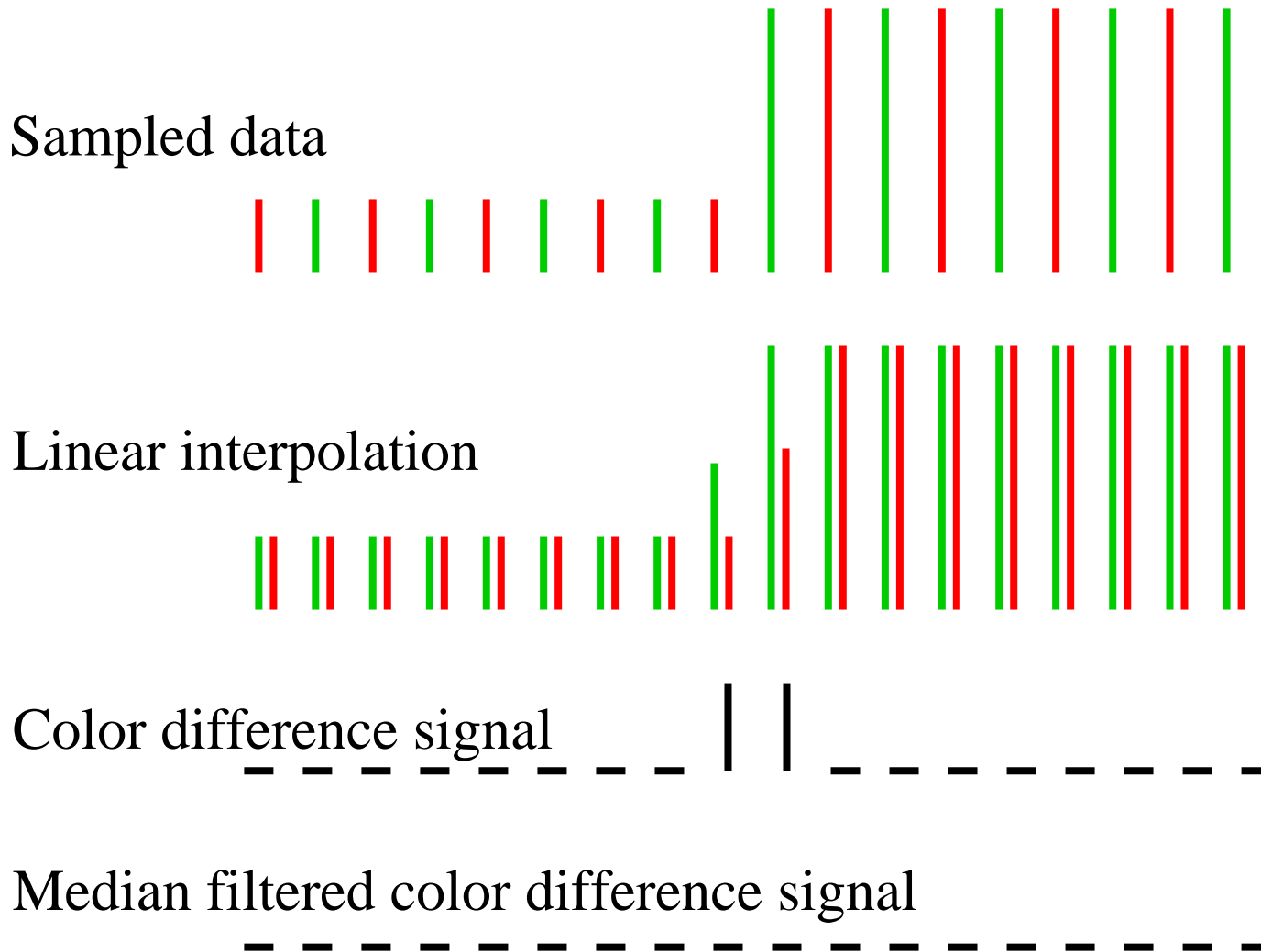
Scale relative
to human
photoreceptor
size: each line
covers about 7
photoreceptors.



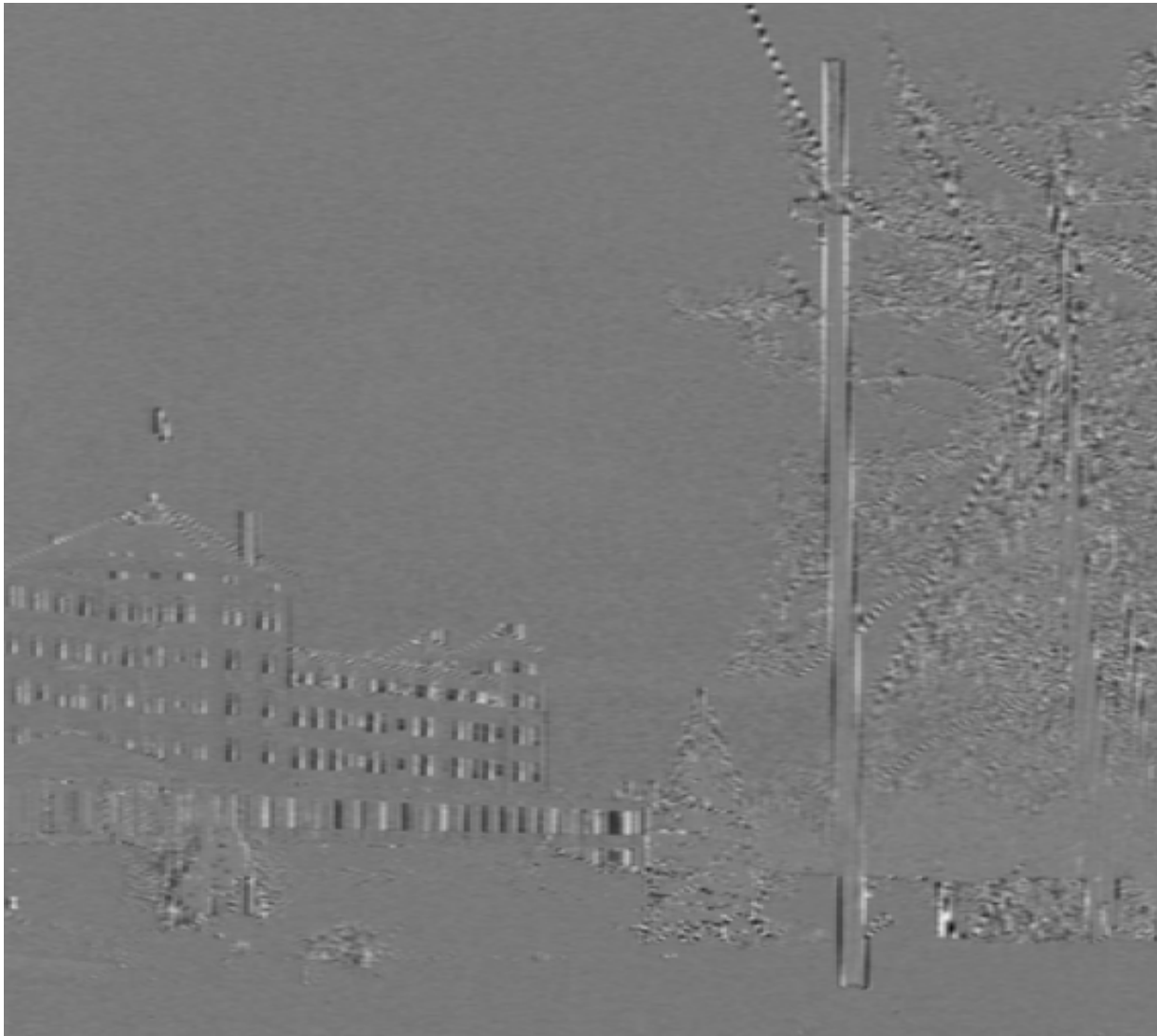
Median Filter Interpolation

- 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



R-G, after linear interpolation



R – G, median filtered (5x5)



Recombining the median filtered colors

Linear interpolation



Median filter interpolation



Didn't get a chance to show:

Local gain control.

- Summary of pyramid representations

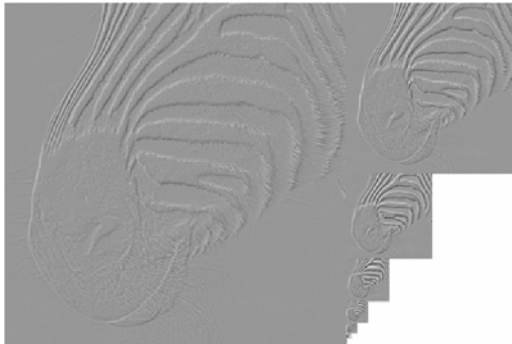
Image pyramids

- Gaussian



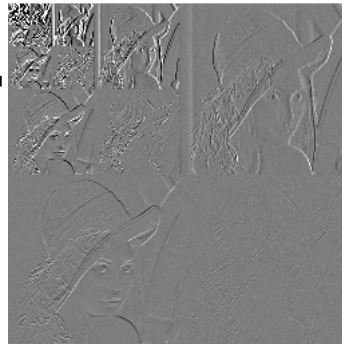
Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- Laplacian



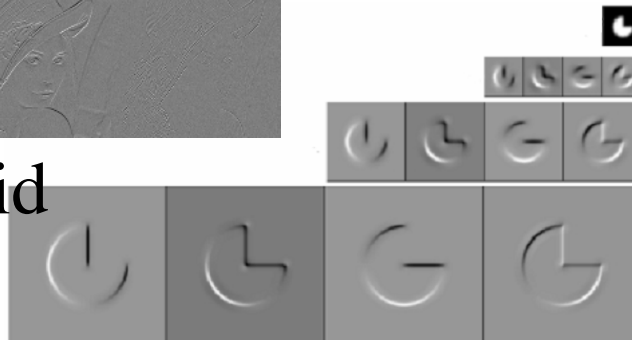
Shows the information added in Gaussian pyramid at each spatial scale. Useful for noise reduction & coding.

- 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

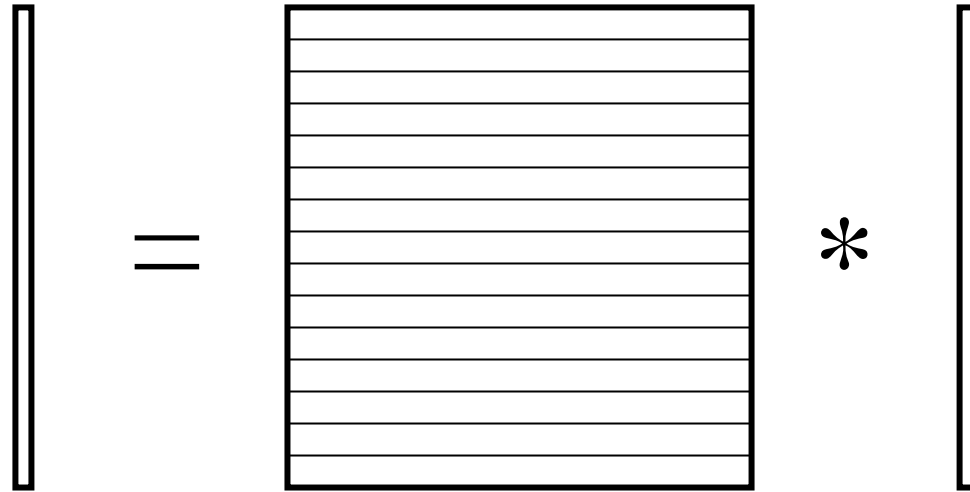
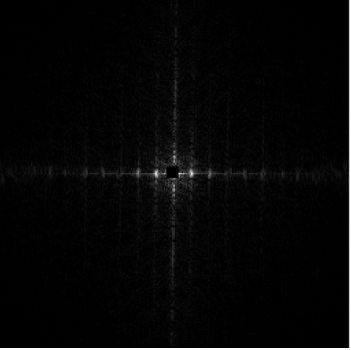
Fourier transform, or
Wavelet transform, or
Steerable pyramid transform

The diagram illustrates the equation $\vec{F} = U\vec{f}$. A blue arrow points from the text 'transformed image' to the vector \vec{F} . Another blue arrow points from the text 'Vectorized image' to the vector \vec{f} . A third blue arrow points from the text 'Fourier transform, or Wavelet transform, or Steerable pyramid transform' to the transformation matrix U .

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.

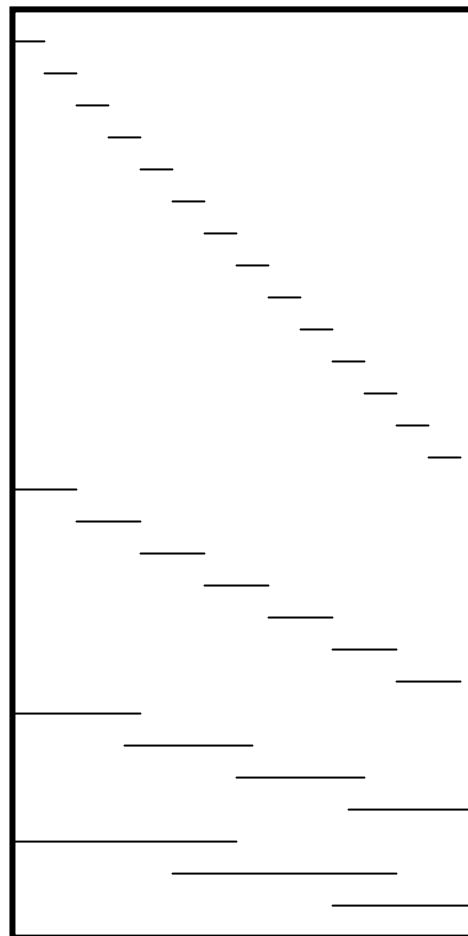
pixel domain
image



Gaussian pyramid

Gaussian
pyramid

=

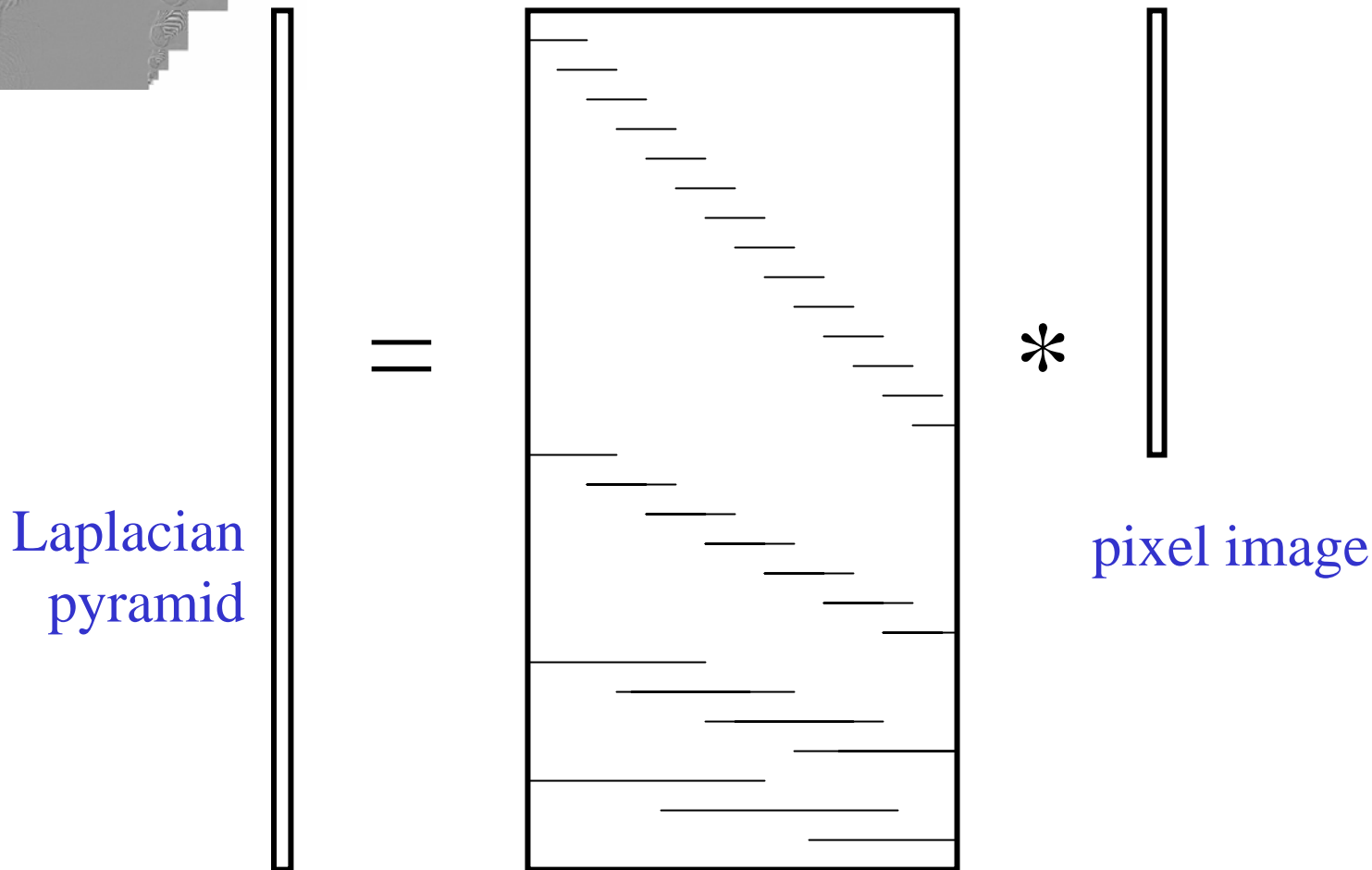


*

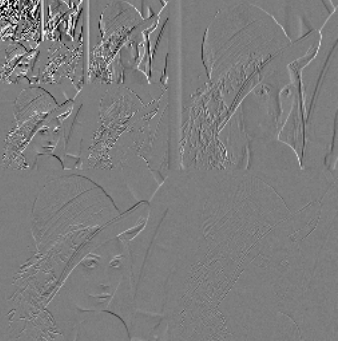
pixel image

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

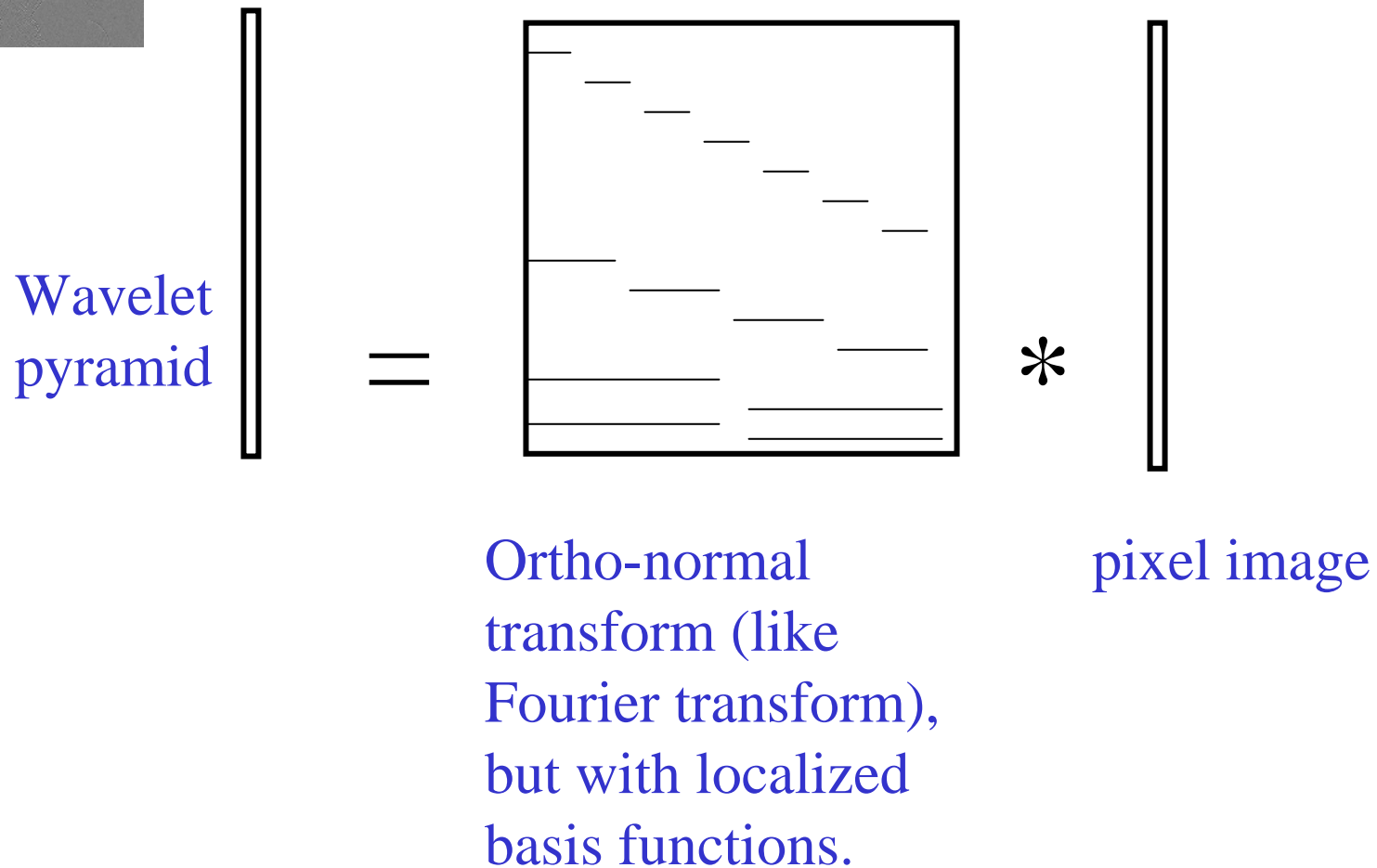
Laplacian pyramid



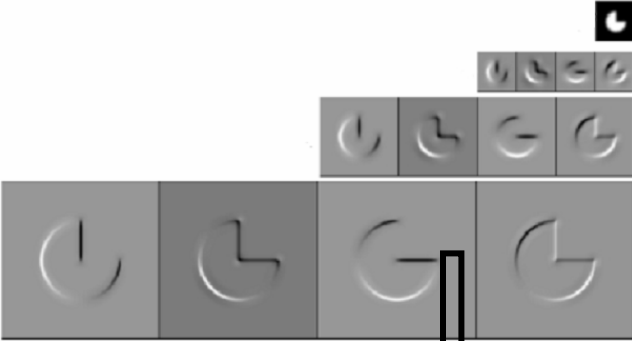
Overcomplete representation.
Transformed pixels represent
bandpassed image information.



Wavelet (QMF) transform



Steerable pyramid

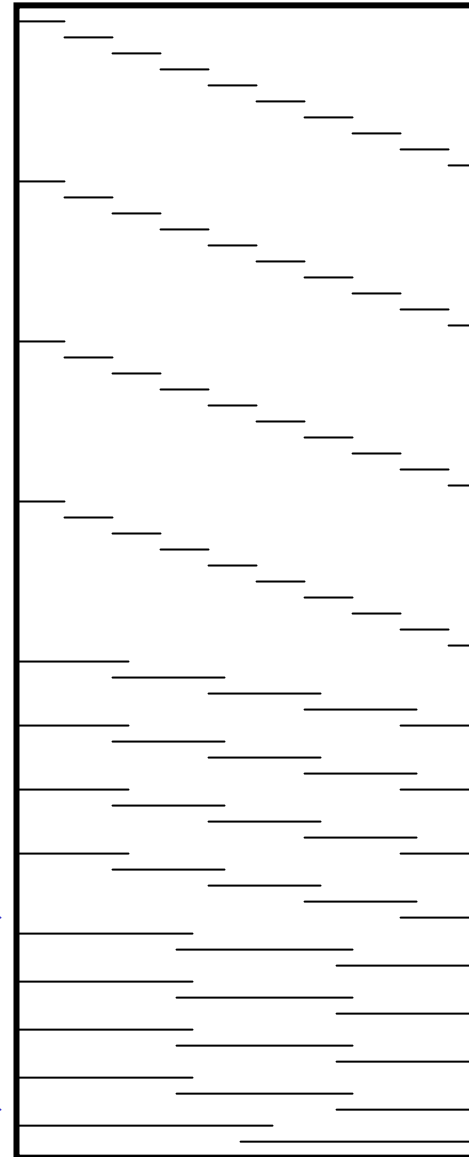


Steerable
pyramid

Multiple
orientations at
= one scale

Multiple
orientations at
the next scale

the next scale...



*

pixel image

Over-complete
representation,
but non-aliased
subbands.

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 \(119k\)](#).

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