Lecture 3

Image Pyramids
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.
The image through the Gaussian window

\[ h(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \]

Too much

Too little
The image through the Gaussian window

Too much

Too little

$$h(x,y) = e^{-\frac{x^2 + y^2}{2\sigma^2}}$$
The image through the Gaussian window

Too much

\[ h(x,y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \]

Too little

Probably still too little…
…but hard enough for now
Analysis of local frequency

Fourier basis:

\[ e^{j2\pi u_0 x} \]

\[ h(x, y; x_0, y_0) = e^{-\frac{(x-x_0)^2 + (y-y_0)^2}{2\sigma^2}} \]
Analysis of local frequency

Fourier basis:

$$e^{j2\pi u_0 x}$$

Gabor wavelet:

$$\psi(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} e^{j2\pi u_0 x}$$

$$h(x,y;x_0,y_0) = e^{-\frac{(x-x_0)^2 + (y-y_0)^2}{2\sigma^2}}$$
Analysis of local frequency

Fourier basis:

\[ e^{j2\pi u_0 x} \]

Gabor wavelet:

\[ \psi(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} e^{j2\pi u_0 x} \]

We can look at the real and imaginary parts:

\[ \psi_c(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]

\[ \psi_s(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \sin(2\pi u_0 x) \]
Gabor wavelets

\[ \psi_c(x,y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]

\[ u_0 = 0 \]
Gabor wavelets

\[ \psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]

\[ u_0 = 0 \]
Gabor wavelets

\[ \psi_c(x,y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]

\[ u_0 = 0 \]

\[ U_0 = 0.1 \]
Gabor wavelets

\[ \psi_c(x, y) = e^{\frac{x^2 + y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]

\( u_0 = 0 \)
\( U_0 = 0.1 \)
\( U_0 = 0.2 \)
Gabor wavelets

\[ \psi_c(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]

\[ \psi_s(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \sin(2\pi u_0 x) \]
Gabor wavelets

\[ \psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]

\[ \psi_s(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \sin(2\pi u_0 x) \]
<table>
<thead>
<tr>
<th>Structure</th>
<th>Operations</th>
<th>2D Fourier Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>World</strong></td>
<td>$I(x,y,t,\lambda)$</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>Optics</strong></td>
<td>Low-pass spatial filtering</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>Photoreceptor Array</strong></td>
<td>Sampling, more low-pass filtering, temporal low/bandpass filtering, $\lambda$ filtering, gain control, response compression</td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>LGN Cells</strong></td>
<td>Spatiotemporal bandpass filtering, $\lambda$ filtering, multiple parallel representations</td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>Primary Visual Cortical Neurons: Simple &amp; Complex</strong></td>
<td>Simple cells: orientation, phase, motion, binocular disparity, &amp; $\lambda$ filtering</td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td>Complex cells: no phase filtering (contrast energy detection)</td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
</tbody>
</table>

*FIGURE 1* Schematic overview of the processing done by the early visual system. On the left, are some of the major structures to be discussed; in the middle, are some of the major operations done at the associated structure; in the right, are the 2-D Fourier representations of the world, retinal image, and sensitivities typical of a ganglion and cortical cell.
Fig. 5. Top row: illustrations of empirical 2-D receptive field profiles measured by J. P. Jones and L. A. Palmer (personal communication) in simple cells of the cat visual cortex. Middle row: best-fitting 2-D Gabor elementary function for each neuron, described by (10). Bottom row: residual error of the fit, indistinguishable from random error in the Chi-squared sense for 97 percent of the cells studied.
Outline

• Linear filtering
• Fourier Transform
• Phase
• Sampling and Aliasing
• Spatially localized analysis
• Quadrature phase
• Oriented filters
• Motion analysis
• Human spatial frequency sensitivity
• Image pyramids
Quadrature filter pairs

A quadrature filter is a complex filter whose real part is related to its imaginary part via a Hilbert transform along a particular axis through origin of the frequency domain.
Contrast invariance!
How quadrature pair filters work

(a) Frequency response of even filter, $G$
(real)

(b) Frequency response of odd filter, $H$
( imaginary)

Figure 3-5: Frequency content of two bandpass filters in quadrature. (a) even phase filter, called $G$ in text, and (b) odd phase filter, $H$. Plus and minus sign illustrate relative sign of regions in the frequency domain. See Fig. 3-6 for calculation of the frequency content of the energy measure derived from these two filters.
How quadrature pair filters work

Figure 3-6: Derivation of energy measure frequency content for the filters of Fig. 3-5. (a) Fourier transform of $G \ast G$. (b) Fourier transform of $H \ast H$. Each squared response has 3 lobes in the frequency domain, arising from convolution of the frequency domain responses. The center lobe is modulated down in frequency while the two outer lobes are modulated up. (There are two sign changes which combine to give the signs shown in (b). To convolve $H$ with itself, we flip it in $f_x$ and $f_y$, which interchanges the $+$ and $-$ lobes of Fig. 3-5 (b). Then we slide it over an unflipped version of itself, and integrate the product of the two. That operation will give positive outer lobes, and a negative inner lobe. However, $H$ has an imaginary frequency response, so multiplying it by itself gives an extra factor of $-1$, which yields the signs shown in (b)). (c) Fourier transform of the energy measure, $G \ast G + H \ast H$. The high frequency lobes cancel, leaving only the baseband spectrum, which has been demodulated in frequency from the original bandpass response. This spectrum is proportional to the sum of the auto-correlation functions of either lobe of Fig. 3-5 (a) and either lobe of Fig. 3-5 (b).
Gabor filter measurements for iris recognition code

John Daugman, http://www.cl.cam.ac.uk/~jgd1000/
Iris codes are compared using Hamming distance.
Setting the Bits in an IrisCode

\[ h_{Re} = 1 \text{ if } \text{Re} \int_{\rho} \int_{\phi} e^{-i\omega(\theta_{0}-\phi)} e^{-(\tau_{0}-\rho)^2/\alpha^2} e^{-(\theta_{0}-\phi)^2/\beta^2} I(\rho, \phi) \rho d\rho d\phi \geq 0 \]

\[ h_{Re} = 0 \text{ if } \text{Re} \int_{\rho} \int_{\phi} e^{-i\omega(\theta_{0}-\phi)} e^{-(\tau_{0}-\rho)^2/\alpha^2} e^{-(\theta_{0}-\phi)^2/\beta^2} I(\rho, \phi) \rho d\rho d\phi < 0 \]

\[ h_{Im} = 1 \text{ if } \text{Im} \int_{\rho} \int_{\phi} e^{-i\omega(\theta_{0}-\phi)} e^{-(\tau_{0}-\rho)^2/\alpha^2} e^{-(\theta_{0}-\phi)^2/\beta^2} I(\rho, \phi) \rho d\rho d\phi \geq 0 \]

\[ h_{Im} = 0 \text{ if } \text{Im} \int_{\rho} \int_{\phi} e^{-i\omega(\theta_{0}-\phi)} e^{-(\tau_{0}-\rho)^2/\alpha^2} e^{-(\theta_{0}-\phi)^2/\beta^2} I(\rho, \phi) \rho d\rho d\phi < 0 \]
Outline

• Linear filtering
• Fourier Transform
• Phase
• Sampling and Aliasing
• Spatially localized analysis
• Quadrature phase
• **Oriented filters**
• Motion analysis
• Human spatial frequency sensitivity
• Image pyramids
Gabor wavelet:
\[ \psi(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} e^{j2\pi u_0 x} \]

Tuning filter orientation:
\[ x' = \cos(\alpha) x + \sin(\alpha) y \]
\[ y' = -\sin(\alpha) x + \cos(\alpha) y \]
Gabor wavelet:

\[ \psi(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} e^{j2\pi u_0 x} \]

Tuning filter orientation:

\[ x' = \cos(\alpha) x + \sin(\alpha) y \]
\[ y' = -\sin(\alpha) x + \cos(\alpha) y \]
Simple example

“Steerability”-- the ability to synthesize a filter of any orientation from a linear combination of filters at fixed orientations.

\[ G_\theta^1 = \cos(\theta)G_0^1 + \sin(\theta)G_{90}^1 \]

Filter Set:

- 0°
- 90°
- Synthesized 30°

Response:

Raw Image → 0° → 90° → Synthesized 30°

Steerable filters

Derivatives of a Gaussian:

\[
\begin{align*}
    h_x(x, y) &= \frac{\partial h(x, y)}{\partial x} = \frac{-x}{2\pi \sigma^4} e^{-\frac{x^2 + y^2}{2\sigma^2}} \\
    h_y(x, y) &= \frac{\partial h(x, y)}{\partial y} = \frac{-y}{2\pi \sigma^4} e^{-\frac{x^2 + y^2}{2\sigma^2}}
\end{align*}
\]

An arbitrary orientation can be computed as a linear combination of those two basis functions:

\[
h_\alpha(x, y) = \cos(\alpha) h_x(x, y) + \sin(\alpha) h_y(x, y)
\]

The representation is “shiftable” on orientation: We can interpolate any other orientation from a finite set of basis functions.
Steerable filters

Derivatives of a Gaussian:

$$h_x(x, y) = \frac{\partial h(x, y)}{\partial x} = \frac{-x}{2\pi \sigma^4} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

$$h_y(x, y) = \frac{\partial h(x, y)}{\partial y} = \frac{-y}{2\pi \sigma^4} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

An arbitrary orientation can be computed as a linear combination of those two basis functions:

$$h_\alpha(x, y) = \cos(\alpha) h_x(x, y) + \sin(\alpha) h_y(x, y)$$

The representation is “shiftable” on orientation: We can interpolate any other orientation from a finite set of basis functions.
Fig. 3. Steerable filter system block diagram. A bank of dedicated filters process the image. Their outputs are multiplied by a set of gain maps that adaptively control the orientation of the synthesized filter.
Steering theorem

Change from Cartesian to polar coordinates

\[ f(x,y) \leftrightarrow H(r,\theta) \]

A convolution kernel can be written using Fourier series in polar angle as:

\[
 f(r, \phi) = \sum_{n=-N}^{N} a_n(r) e^{in\phi}
\]

**Theorem:** Let \( T \) be the number of nonzero coefficients \( a_n(r) \). Then, the function \( f \) can be steer with \( T \) functions.
Steering theorem for polynomials

\[ f(x,y) = W(r) P(x,y) \]

**Theorem 3:** Let \( f(x,y) = W(r)P_N(x,y) \), where \( W(r) \) is an arbitrary windowing function, and \( P_N(x,y) \) is an \( N \)th order polynomial in \( x \) and \( y \), whose coefficients may depend on \( r \). Linear combinations of \( 2N + 1 \) basis functions are sufficient to synthesize \( f(x,y) = W(r)P_N(x,y) \) rotated to any angle. Equation (10) gives the interpolation functions \( k_j(\theta) \). If \( P_N(x,y) \) contains only even [odd] order terms (terms \( x^n y^m \) for \( n + m \) even [odd]), then \( N + 1 \) basis functions are sufficient, and (10) can be modified to contain only the even [odd] numbered rows (counting from zero) of the left-hand side column vector and the right-hand side matrix.

For an \( N \)th order polynomial with even symmetry \( N+1 \) basis functions are sufficient.
Steerability

Important example is 2\textsuperscript{nd} derivative of Gaussian

\[ G_2^0 = (4x^2 - 2)e^{-(x^2+y^2)} \]  
(~Laplacian):

\begin{align*}
G_{2a} &= 0.9213(2x^2 - 1)e^{-(x^2+y^2)} \\
G_{2b} &= 1.843xy e^{-(x^2+y^2)} \\
G_{2c} &= 0.9213(2y^2 - 1)e^{-(x^2+y^2)}
\end{align*}

\begin{align*}
k_a(\theta) &= \cos^2(\theta) \\
k_b(\theta) &= -2\cos(\theta)\sin(\theta) \\
k_c(\theta) &= \sin^2(\theta)
\end{align*}

Figure 16: X-Y separable basis filters for $G_2$, listed in Tables 3 and 4.

Table 3: X-Y separable basis set and interpolation functions for second derivative of Gaussian. To create a second derivative of a Gaussian rotated along to an angle $\theta$, use: $G_2^0 = (k_a(\theta) G_{2a} + k_b(\theta) G_{2b} + k_c(\theta) G_{2c})$. The minus sign in $k_b(\theta)$ selects the direction of positive $\theta$ to be counter-clockwise.

Two equivalent basis

These two basis can use to steer 2\textsuperscript{nd} order Gaussian derivatives

(a) $G_2$ Basis Set

(b) $G_2$ Amplitude Spectra

(c) $G_2$ X-Y Separable Basis Set
Approximated quadrature filters for 2\textsuperscript{nd} order Gaussian derivatives (this approximation requires 4 basis to be steerable)

(d) $H_2$ Basis Set

(e) $H_2$ Amplitude Spectra

(f) $H_2$ X-Y Separable Basis Set
Second directional derivative of a Gaussian and its quadrature pair

(a) Original image  
(b) real component of filtered image  
(c) imaginary component of filtered image  
(d) sum of the squares of (b) and (c)
Orientation analysis

Fig. 9. Test images of (a) vertical line and (b) intersecting lines; (c) and (d) oriented energy as a function of angle at the centers of test images (a) and (b). Oriented energy was measured using the $G_4$, $H_4$ quadrature steerable pair; (e) and (f) polar plots of (c) and (d).
Orientation analysis

High resolution in orientation requires many oriented filters as basis (high order gaussian derivatives).

Fig. 9. Test images of (a) vertical line and (b) intersecting lines; (c) and (d) oriented energy as a function of angle at the centers of test images (a) and (b). Oriented energy was measured using the $G_4$, $H_4$ quadrature steerable pair; (e) and (f) polar plots of (c) and (d).
Fig. 8. (a) Original image of Einstein; (b) orientation map of (a) made using the lowest order terms in a Fourier series expansion for the oriented energy as measured with $G_2$ and $H_2$. Table XI gives the formulas for these terms.
Fig. 10. Measures of orientation derived from $G_4$ and $H_4$ steerable filter outputs: (a) Input image for orientation analysis; (b) angular average of oriented energy as measured by $G_4, H_4$ quadrature pair. This is an oriented features detector; (c) conventional measure of orientation: dominant orientation plotted at each point. No dominant orientation is found at the line intersection or corners; (d) oriented energy as a function of angle, shown as a polar plot for a sampling of points in the image (a). Note the multiple orientations found at intersection points of lines or edges and at corners, shown by the florets there.
A contour detector
A contour detector

edge detector output

(a)
A contour detector

edge detector output

Local energy
A contour detector

(a) edge detector output

(b) Local energy

(c) Phase ~ 90

Phase ~ 0
Figure 3-8: The problem with using energy measures to analyze a structure of multiple orientations, and how to solve it (part one). (a) Horizontal line and (b) floret polar plot of $G_2$ and $H_2$ quadrature pair oriented energies as a function of angle and position. The same for a vertical line are shown in (c) and (d). Continued in Fig. 3-9.
Figure 3-9: The problem with using energy measures to analyze a structure of multiple orientations, and how to solve it (part two). (a) Cross image (the sum of Fig. 3-8 (a) and (c)). The oriented energy (b) of the cross is not the sum of the energies of the horizontal and vertical lines, Fig. 3-8 (b) and (d), due to an effect analogous to optical interference. Many of the florets do not show the two orientations which are present; several show angularly uniform responses. For comparison, (c) shows the sum of energies Fig. 3-8 (b) and (d). Floret polar plot of energies after spatial blurring, (d), are predicted to remove interference effects, as described in text. Note that the energy local maxima correspond to image structure orientations. These florets are nearly identical to the sum of blurred energies of the horizontal and vertical lines, (e), showing that superposition nearly holds. (The agreement is not exact because the low-pass filter used for the blurring was not perfect).
Figure 2-10: Example of a three-dimensional steerable filter. Surfaces of constant value are shown for the six basis filters of a second derivative of a three-dimensional Gaussian. Linear combinations of these six filters can synthesize the filter rotated to any orientation in three-space. Such three-dimensional steerable filters are useful for analysis and enhancement of motion sequences or volumetric image data, such as MRI or CT data. For discussions of steerable filters in three or more dimensions, see [59, 58, 33, 89]. (Martin Friedmann rendered this image with the Thingworld program).
Outline

- Linear filtering
- Fourier Transform
- Phase
- Sampling and Aliasing
- Spatially localized analysis
- Quadrature phase
- Oriented filters

- **Motion analysis**
- Human spatial frequency sensitivity
- Image pyramids
The space time volume

xyt- space-time volume
The space time volume

xt-slice

xyt- space-time volume
The space time volume

xt-slice

Static objects - vertical lines
Moving objects slanted lines, slope ~ motion velocity
Motion signals in space-time

space-time domain

spatio-temporal Fourier transform domain

\( \omega_t \)

\( \omega_x \)
Motion signals in space-time

space-time domain

spatio-temporal Fourier transform domain

\[ \omega_t \]

\[ \omega_x \]

note locus of energy

Wednesday, September 12, 12
Motion signals in space-time

space-time domain

spatio-temporal Fourier transform domain

spatio-temporal filters

cos phase filter

sin phase filter

note locus of energy

power in frequency domain

Wednesday, September 12, 12
Evidence for filter-based analysis of motion in the human visual system
Approximation to a square wave using a sequence of odd harmonics

Using Fourier series we can write an ideal square wave as an infinite series of the form

\[ x_{\text{square}}(t) = \frac{4}{\pi} \left( \sin(2\pi ft) + \frac{1}{3} \sin(6\pi ft) + \frac{1}{5} \sin(10\pi ft) + \cdots \right). \]

http://en.wikipedia.org/wiki/Square_wave
Space-time picture of translating square wave
Space-time picture of translating square wave

\[ \sin(w \cdot x) \]
Space-time picture of translating fluted square wave
Space-time picture of translating fluted square wave

1/3 \sin(3w x)
Translating Square Wave (phase advances by 90 degrees each time step)
Translating Fluted Square Wave (phase of lowest remaining sinusoidal component advances by 270 degrees (-90) each time step)
Outline

- Linear filtering
- Fourier Transform
- Phase
- Sampling and Aliasing
- Spatially localized analysis
- Quadrature phase
- Oriented filters
- Motion analysis
- Human spatial frequency sensitivity
- Image pyramids
Local image representations
Local image representations

A pixel

[r,g,b]
Local image representations

A pixel $[r,g,b]$

An image patch
Local image representations

A pixel
[r,g,b]

An image patch

Gabor filter pair in quadrature

Gabor jet


Local image representations

A pixel
[r,g,b]

An image patch

Gabor filter pair in quadrature

Gabor jet

V1 sketch: hypercolumns


Gabor Filter Bank

or = [4 4 4 4];
or = [12 6 3 2];
Image pyramids

- Gaussian pyramid
- Laplacian pyramid
- Wavelet/QMF pyramid
- Steerable pyramid
Image pyramids

- Gaussian pyramid
- Laplacian pyramid
- Wavelet/QMF pyramid
- Steerable pyramid
The Gaussian pyramid

• Smooth with gaussians, because
  – a gaussian*gaussian=another gaussian
• Gaussians are low pass filters, so representation is redundant.
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. 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.
Convolution and subsampling as a matrix multiply (1-d case)

\[ x_2 = G_1 x_1 \]

\[ G_1 = \]

\[
\begin{array}{cccccccccccccccccccc}
1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

(Normalization constant of 1/16 omitted for visual clarity.)
Next pyramid level

\[ x_3 = G_2 x_2 \]

\[
G_2 =
\begin{bmatrix}
1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \\
\end{bmatrix}
\]
The combined effect of the two pyramid levels

\[ x_3 = G_2 G_1 x_1 \]

\[ G_2 G_1 = \]

\[
\begin{array}{cccccccccccccccc}
1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 31 & 20 & 10 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 31 & 20 & 10 & 4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 30 & 16 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 25 & 16 & 4 & 0 & 0 & 0
\end{array}
\]
Fig. 2. The equivalent weighting functions $h_i(x)$ 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.
Gaussian pyramids used for

- up- or down- sampling images.
- Multi-resolution image analysis
  - Look for an object over various spatial scales
  - Coarse-to-fine image processing: form blur estimate or the motion analysis on very low-resolution image, upsample and repeat. Often a successful strategy for avoiding local minima in complicated estimation tasks.
1-d Gaussian pyramid matrix, for \([1 \ 4 \ 6 \ 4 \ 1]\) low-pass filter

full-band image, highest resolution

lower-resolution image

lowest resolution image
Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid
The Laplacian Pyramid

• Synthesis
  – Compute the difference between upsampled Gaussian pyramid level and Gaussian pyramid level.
  – band pass filter - each level represents spatial frequencies (largely) unrepresented at other level.
Laplacian pyramid algorithm
Laplacian pyramid algorithm

\[ x_1 \]
Laplacian pyramid algorithm

\[ x_1 \quad G_1 x_1 \]

\[ x_1 - G_1 x_1 \]

\[ \text{Repeatedly apply this process} \]
Laplacian pyramid algorithm

$x_1 \rightarrow G_1 x_1 \rightarrow - \rightarrow F_1 G_1 x_1$
Laplacian pyramid algorithm

\[ x_1 \rightarrow G_1 x_1 \rightarrow F_1 G_1 x_1 \rightarrow (I - F_1 G_1) x_1 \]
Laplacian pyramid algorithm

\[ x_1, \quad G_1 x_1 = x_2 \]

\[ F_1 G_1 x_1 \]

\[ (I - F_1 G_1) x_1 \]
Laplacian pyramid algorithm

\[ x_1 \]

\[ G_1 x_1 = x_2 \]

\[ \begin{array}{c}
\text{\( F_1 G_1 x_1 \)} \\
\text{\( \( I - F_1 G_1 \) x_1 \)} \\
\text{\( \( I - F_2 G_2 \) x_2 \)}
\end{array} \]
Laplacian pyramid algorithm

\[ x_1 \xrightarrow{G_1} x_2 \xrightarrow{F_2 G_1} (I - F_2 G_2) x_2 \xrightarrow{(I - F_3 G_3)} x_3 \]

\[ (I - F_1 G_1) x_1 \]
Upsampling

\[ y_2 = F_3 x_3 \]

Insert zeros between pixels, then apply a low-pass filter, \([1 \, 4 \, 6 \, 4 \, 1]\)

\[ F_3 = \begin{bmatrix} 6 & 1 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 4 \end{bmatrix} \]
Showing, at full resolution, the information captured at each level of a Gaussian (top) and Laplacian (bottom) pyramid.

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.
Laplacian pyramid reconstruction algorithm:
recover $x_1$ from $L_1$, $L_2$, $L_3$ and $x_4$

$G#$ is the blur-and-downsample operator at pyramid level #
$F#$ is the blur-and-upsample operator at pyramid level #

Laplacian pyramid elements:
$L_1 = (I - F_1 G_1) x_1$
$L_2 = (I - F_2 G_2) x_2$
$L_3 = (I - F_3 G_3) x_3$

$x_2 = G_1 x_1$
$x_3 = G_2 x_2$
$x_4 = G_3 x_3$

Reconstruction of original image ($x_1$) from Laplacian pyramid elements:
$x_3 = L_3 + F_3 x_4$
$x_2 = L_2 + F_2 x_3$

$x_1 = L_1 + F_1 x_2$
Laplacian pyramid reconstruction algorithm:
recover $x_1$ from $L_1$, $L_2$, $L_3$ and $g_3$
Gaussian pyramid
Laplacian pyramid
1-d Laplacian pyramid matrix, for \([1 4 6 4 1]\) low-pass filter

- high frequencies
- mid-band frequencies
- low frequencies
Laplacian pyramid applications

- Texture synthesis
- Image compression
- Noise removal
Image blending

(a)

(b)
Figure 3.42  Laplacian pyramid blending details (Burt and Adelson 1983b) © 1983 ACM.
The first three rows show the high, medium, and low frequency parts of the Laplacian pyramid.
Image blending

- Build Laplacian pyramid for both images: LA, LB
- Build Gaussian pyramid for mask: G
- Build a combined Laplacian pyramid: \( L(j) = G(j) \ LA(j) + (1-G(j)) \ LB(j) \)
- Collapse L to obtain the blended image
Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid
Linear transforms

\[ \vec{F} = U \vec{f} \quad \leftrightarrow \quad \vec{f} = U^{-1} \vec{F} \]

Note: not all important transforms need to have an inverse
Linear transforms

\[ \overrightarrow{F} = U\overrightarrow{f} \]

Pixels

\[
\begin{array}{cccc}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{array}
\]

\[ U = \]

\[ \]

\[
\]

Wednesday, September 12, 12
### Linear transforms

**Pixels**

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\mathbf{U} =
\]

**Derivative**

\[
\begin{bmatrix}
1 & -1 & 0 & 0 \\
0 & 1 & -1 & 0 \\
0 & 0 & 1 & -1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\mathbf{U} =
\]

\[
\mathbf{F} = \mathbf{U} \mathbf{f}
\]
Linear transforms

## Pixels

\[
U = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

## Derivative

\[
U = \begin{bmatrix}
1 & -1 & 0 & 0 \\
0 & 1 & -1 & 0 \\
0 & 0 & 1 & -1 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[\vec{F} = U \vec{f}\]

\[U^{-1} = \]
Linear transforms

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

Pixels

$$
\mathbf{U} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
$$

Derivative

$$
\mathbf{U} = \begin{bmatrix}
1 & -1 & 0 & 0 \\
0 & 1 & -1 & 0 \\
0 & 0 & 1 & -1 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
$$

$\mathbf{U}^{-1} = \begin{bmatrix}
1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}$
**Linear transforms**

\[ \vec{F} = U \vec{f} \]

### Pixels

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Derivative

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Integration

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- No locality for reconstruction
- Needs boundary
Haar transform

The simplest set of functions:

\[
U = \begin{pmatrix}
1 & 1 \\
1 & -1
\end{pmatrix}
\quad U^{-1} =
\]

\[
\vec{F} = U \vec{f}
\]
Haar transform

The simplest set of functions:

\[ U = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad U^{-1} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \]
Haar transform

The simplest set of functions:

$$U = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad U^{-1} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix}$$

To code a signal, repeat at several locations:

$$U = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix} \quad U^{-1} = \frac{1}{2}$$
Haar transform

The simplest set of functions:

\[ U = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad U^{-1} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \]

To code a signal, repeat at several locations:

\[ U = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \]

\[ \begin{align*} U^{-1} &= \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \end{align*} \]
Haar transform

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

\[ \mathbf{F} = \mathbf{Uf} \]
Haar transform

\[ \vec{F} = U f \]

Reordering rows

\[
\begin{array}{cc}
1 & 1 \\
1 & -1 \\
1 & 1 \\
1 & -1 \\
1 & 1 \\
1 & -1 \\
1 & 1 \\
1 & -1 \\
\end{array}
\]
### Haar Transform

The Haar transform is a type of wavelet transform that is particularly simple and useful for signal processing tasks. It is named after Alfred Haar, who introduced it in 1909.

The Haar wavelet basis functions are defined over the interval $[0, 1]$ and are piecewise constant. There are two types of Haar wavelets:

1. **Haar functions** ($f_n(x)$) are defined as
   
   $f_n(x) = \begin{cases} 
   1 & \text{if } 0 \leq x < \frac{1}{2} \\
   -1 & \text{if } \frac{1}{2} \leq x < 1 \\
   \end{cases}$

2. **Daubechies wavelets** ($\psi_n(x)$) are a generalization of the Haar wavelets and are defined using a symmetric, compactly supported function.

The discrete Haar transform can be computed as follows:

Given a discrete signal $x = (x_0, x_1, \ldots, x_{n-1})$, the Haar transform $F$ can be computed as:

$$F_k = \sum_{n=0}^{n-1} x_n \cdot 2^{-k/2}$$

where $k = 0, 1, \ldots, n-1$.

The discrete inverse Haar transform can be computed as:

$$x_n = \sum_{k=0}^{n-1} F_k \cdot 2^k$$

### Example

Consider the discrete signal $x = (1, 1, 1, 1, 1, 1, 1, 1)$. The Haar transform $F$ of $x$ is computed as:

$$F_0 = 1, F_1 = 2, F_2 = 3, F_3 = 4, F_4 = 5, F_5 = 6, F_6 = 7, F_7 = 8$$

The Haar transform $F$ is then reordered as follows:

$$F = (1, 3, 5, 7, 2, 4, 6, 8)$$

Reordering rows in the matrix representation of the signal $x$, we get:

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

This transformation is useful in various applications, including image compression and denoising.
**Haar transform**

\[ \tilde{F} = UF \]

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

Reordering rows:

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Low pass

High pass
Haar transform

Apply the same decomposition to the Low pass component:
Haar transform

\[ \widehat{F} = U \vec{f} \]

Apply the same decomposition to the Low pass component:
Haar transform

\[ \tilde{F} = UF \]

Apply the same decomposition to the Low pass component:

Reordering rows

Low pass

High pass
Haar transform

Apply the same decomposition to the Low pass component:

$$\vec{F} = \vec{Uf}$$
Haar transform

\[ \hat{F} = U f \]

Apply the same decomposition to the Low pass component:

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 \\
\end{array}
\]

= 

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & -1 & -1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]
Haar transform

Apply the same decomposition to the Low pass component:

And repeat the same operation to the low pass component, until length 1.
Haar transform

Reordering rows

Low pass

High pass

Apply the same decomposition to the Low pass component:

And repeat the same operation to the low pass component, until length 1.

Note: each subband is sub-sampled and has aliased signal components.
## Haar transform

The entire process can be written as a single matrix:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & 1 & -1 & -1 & & & & \\
1 & -1 & & & & & & \\
1 & -1 & & & & & & \\
1 & -1 & & & & & & \\
1 & -1 & & & & & & \\
1 & -1 & & & & & &
\end{bmatrix}
\]

- **Average**
- **Multiscale derivatives**

\[
\vec{F} = \vec{Uf}
\]
Haar transform

\[ \hat{F} = Uf \]

\[
U = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & 1 & -1 & -1 & & & & \\
& & & & 1 & 1 & -1 & -1 \\
1 & -1 & & & & & & \\
& & & & 1 & -1 & & \\
& & & & & & & 1 & -1 \\
& & & & & & & 1 & -1 \\
\end{bmatrix}
\]

\[
U^{-1} = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & 1 & -1 & -1 & & & & \\
& & & & 1 & 1 & -1 & -1 \\
1 & -1 & & & & & & \\
& & & & 1 & -1 & & \\
& & & & & & & 1 & -1 \\
& & & & & & & 1 & -1 \\
\end{bmatrix}
\]
Haar transform

\[ \hat{F} = U \vec{f} \]

\[ U = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & 1 & -1 & -1 & & & & \\
1 & -1 & & & & & & \\
1 & -1 & & & & & & \\
1 & -1 & & & & & & \\
1 & -1 & & & & & & \\
1 & -1 & & & & & & 
\end{bmatrix} \]

\[ U^{-1} = \begin{bmatrix}
0.125 & 0.125 & 0.25 & 0 & 0.5 & 0 & 0 & 0 \\
0.125 & 0.125 & 0.25 & 0 & -0.5 & 0 & 0 & 0 \\
0.125 & 0.125 & -0.25 & 0 & 0 & 0.5 & 0 & 0 \\
0.125 & -0.125 & 0 & 0.25 & 0 & 0 & 0.5 & 0 \\
0.125 & -0.125 & 0 & 0.25 & 0 & 0 & -0.5 & 0 \\
0.125 & -0.125 & 0 & -0.25 & 0 & 0 & 0 & 0.5 \\
0.125 & -0.125 & 0 & -0.25 & 0 & 0 & 0 & -0.5 
\end{bmatrix} \]
Haar transform

Properties:
• Orthogonal decomposition
• Perfect reconstruction
• Critically sampled
2D Haar transform

Basic elements:

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & -1 & 1 \\
\end{array}
\]
2D Haar transform

Basic elements:

\[
\begin{pmatrix}
 1 & 1 \\
 1 & -1 \\
\end{pmatrix}, \quad \begin{pmatrix}
 1 & 1 \\
\end{pmatrix}, \quad \begin{pmatrix}
 1 & -1 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
 1 & 1 \\
\end{pmatrix}, \quad \begin{pmatrix}
 1 & 1 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
 1 & 1 & 1 \\
 1 & 1 & 1 \\
\end{pmatrix}
\]

Low pass
2D Haar transform

Basic elements:

\[
\begin{array}{ccc}
1 & 1 \\
1 & 1 \\
-1 & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 \\
1 & 1 \\
1 & 1 \\
1 & -1 \\
\end{array}
\]

\[
\Downarrow 2
\]

Low pass
# 2D Haar transform

**Basic elements:**

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>-1</td>
</tr>
</tbody>
</table>

Low pass
2D Haar transform

Basic elements:

\[
\begin{array}{ccc}
1 & & 1 \\
1 & & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & -1 \\
\end{array}
\]

Low pass

\[
\begin{array}{ccc}
1 & & 1 \\
1 & & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & -1 \\
\end{array}
\]
2D Haar transform

Basic elements:

\[
\begin{array}{ccc}
1 & & 1 \\
1 & 1 & 1 \\
-1 & & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
-1 & -1 & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
-1 & -1 & -1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
-1 & -1 & -1 \\
\end{array}
\]

Low pass

High pass vertical

High pass horizontal

High pass diagonal

Wednesday, September 12, 12
2D Haar transform

Sketch of the Fourier transform

\[
\begin{pmatrix}
1 & 1 \\
1 & 1
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & -1 \\
1 & -1
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 1 \\
-1 & -1
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & -1 \\
-1 & 1
\end{pmatrix}
\]
2D Haar transform

Sketch of the Fourier transform

Horizontal low pass, Vertical low-pass
Horizontal high pass, vertical low-pass
Horizontal low pass, vertical high-pass
Horizontal high pass, vertical high pass

Wednesday, September 12, 12
Pyramid cascade

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 $\pi$. This is divided into four subbands at the next level. On each subsequent level, the lowpass subbands (outlined in bold) is subdivided further.
Wavelet/QMF representation

Same number of pixels!
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).
  – Separable computation, so it’s fast.
What is wrong with orthonormal basis?
What is wrong with orthonormal basis?

The representation is not translation invariant. It is not stable.
Shifttable transforms

The representation has to be stable under typical transformations that undergo visual objects:

Translation

Rotation

Scaling

...

Shiftability under space translations corresponds to lack of aliasing.

http://www.cns.nyu.edu/pub/eero/simoncelli91*reprint.pdf
Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid
Steerable Pyramid

2 Level decomposition of white circle example:

Images from: http://www.cis.upenn.edu/~eero/steerpyr.html
We may combine Steerability with Pyramids to get a Steerable Laplacian Pyramid as shown below.

**Decomposition**  
**Reconstruction**
Steerable Pyramid

We may combine Steerability with Pyramids to get a Steerable Laplacian Pyramid as shown below.

Images from: http://www.cis.upenn.edu/~eero/steerpyr.html
Steerable Pyramid

We may combine Steerability with Pyramids to get a Steerable Laplacian Pyramid as shown below

Images from: http://www.cis.upenn.edu/~eero/steerpyr.html
Figure 1. Idealized illustration of the spectral decomposition performed by a steerable pyramid with $k = 4$. Frequency axes range from $-\pi$ to $\pi$. 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.
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 $\pi$. 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.
There is also a high pass residual…
Dog or cat?
Almost no dog information
Steerable pyramids

• Good:
  – Oriented subbands
  – Non-aliased subbands
  – Steerable filters
  – Used for: noise removal, texture analysis and synthesis, super-resolution, shading/paint discrimination.

• 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.
<table>
<thead>
<tr>
<th></th>
<th>Laplacian Pyramid</th>
<th>Dyadic QMF/Wavelet</th>
<th>Steerable Pyramid</th>
</tr>
</thead>
<tbody>
<tr>
<td>self-inverting (tight frame)</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>overcompleteness</td>
<td>4/3</td>
<td>1</td>
<td>4k/3</td>
</tr>
<tr>
<td>aliasing in subbands</td>
<td>perhaps</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>rotated orientation bands</td>
<td>no</td>
<td>only on hex lattice [9]</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Table 1**: Properties of the Steerable Pyramid relative to two other well-known multi-scale representations.
• Summary of pyramid representations
Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid
Image pyramids

- Gaussian

- Laplacian

- Wavelet/QMF

- Steerable pyramid

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

- **Steerable pyramid**
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**
**Image pyramids**

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

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

- **Wavelet/QMF**
  - Shows components at each scale and orientation separately. Non-aliased subbands. Good for texture and feature analysis. But overcomplete and with HF residual.

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

\[ \vec{F} = U \vec{f} \]

Fourier transform, or Wavelet transform, or Steerable pyramid transform

transformed image

Vectorized image
Fourier transform

Fourier transform

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

pixel domain image
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.
Gaussian pyramid

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

Overcomplete representation. Transformed pixels represent bandpassed image information.
Laplacian pyramid

\[ \text{Laplacian pyramid} = \star \text{pixel image} \]

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

Wavelet pyramid
Wavelet (QMF) transform

Wavelet pyramid

= *

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

Over-complete representation, but non-aliased subbands.
Matlab resources for pyramids (with tutorial)

http://www.cns.nyu.edu/~eero/software.html
Matlab resources for pyramids (with tutorial)
http://www.cns.nyu.edu/~eero/software.html

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)
- **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]**:
- **CL-SHELL [Gnu Emacs <-> Common Lisp Interface]**:
  README / Change Log / Source Code (119k).
Matlab resources for pyramids (with tutorial)
http://www.cns.nyu.edu/~eero/software.html

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)
- **EPWC** - 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.
Why use these representations?

• Handle real-world size variations with a constant-size vision algorithm.
• Remove noise
• Analyze texture
• Recognize objects
• Label image features
• Image priors can be specified naturally in terms of wavelet pyramids.