Wavelets and Multiresolution Processing

Chapter 7

Prof. Vidya Manian
Dept. of Electrical and Computer Engineering
Overview

- Background
- Multiresolution expansion
- Wavelet transform in one dimension
- Fast wavelet transform
- Wavelet transforms in two dimensions
• Fourier basis functions are sinusoids, wavelets are based on small waves, of varying frequency and limited duration
• Fourier-only frequency information, no temporal information
• Wavelets: information of frequency when and where
• Signal processing and analysis using wavelets (Mallat[1987])
Multiresolution

- Subband coding from signal processing
- Quadrature mirror filtering from digital speech recognition
- Pyramidal image processing
- Features that might go undetected at one resolution may be easy to detect at another
Background

FIGURE 7.1
An image and its local histogram variations.
Image pyramids

- Representing images at more than one resolution
- Collection of decreasing resolution images arranged in the shape of a pyramid
- Base level $J$ is of size $2^J \times 2^J$ or $N \times N$
- $J = \log_2 N$, apex level $0$ is of size $1 \times 1$
- General level $j$ is of size $2^j \times 2^j$ where $0 \leq j \leq J$
- Restrict to $P$ reduced resolution approximations of the original image
FIGURE 7.2
(a) An image pyramid. (b) A simple system for creating approximation and prediction residual pyramids.
• Level j-1 approximation outputs are used to build approximation pyramid
• Level j prediction residual output is used to build complementary prediction residual pyramid
• Step 1: compute a reduced –resolution approximation of the level j input image. Place the resulting approximation at level j-1
• Step 2: create an estimate of level j input image from above approx. the resulting prediction image has same dimensions as the level j input image
• Step 3: compute difference between the prediction image of step 2 and input of step 1. Place this result in level j of the prediction residual pyramid
Example

FIGURE 7.3
Two image pyramids and their histograms:
(a) an approximation pyramid;
(b) a prediction residual pyramid.
Subband coding

• Image is decomposed into a set of bandlimited components called subbands
• The subbands can be reassembled to reconstruct the original image
FIGURE 7.4 (a) A digital filter; (b) a unit discrete impulse sequence; and (c) the impulse response of the filter.
**FIGURE 7.5** Six functionally related filter impulse responses: (a) reference response; (b) sign reversal; (c) and (d) order reversal (differing by the delay introduced); (e) modulation; and (f) order reversal and modulation.
FIGURE 7.6
(a) A two-band subband coding and decoding system, and (b) its spectrum splitting properties.
A two-dimensional, four-band filter bank for subband image coding.
Example

<table>
<thead>
<tr>
<th>$n$</th>
<th>$g_0(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.23037781</td>
</tr>
<tr>
<td>1</td>
<td>0.71484657</td>
</tr>
<tr>
<td>2</td>
<td>0.63088076</td>
</tr>
<tr>
<td>3</td>
<td>-0.02798376</td>
</tr>
<tr>
<td>4</td>
<td>-0.18703481</td>
</tr>
<tr>
<td>5</td>
<td>0.03084138</td>
</tr>
<tr>
<td>6</td>
<td>0.03288301</td>
</tr>
<tr>
<td>7</td>
<td>-0.01059740</td>
</tr>
</tbody>
</table>

**TABLE 7.1**
Daubechies 8-tap orthonormal filter coefficients for $g_0(n)$ (Daubechies [1992]).
FIGURE 7.8
The impulse responses of four 8-tap Daubechies orthonormal filters. See Table 7.1 for the values of $g_0(n)$ for $0 \leq n \leq 7$. 
FIGURE 7.9
A four-band split of the vase in Fig. 7.1 using the subband coding system of Fig. 7.7. The four subbands that result are the (a) approximation, (b) horizontal detail, (c) vertical detail, and (d) diagonal detail subbands.
The Haar transform

FIGURE 7.10
(a) A discrete wavelet transform using Haar $H_2$ basis functions. Its local histogram variations are also shown. (b)–(d) Several different approximations ($64 \times 64$, $128 \times 128$, and $256 \times 256$) that can be obtained from (a).
Haar scaling functions

\[ \varphi_{0,0}(x) = \varphi(x) \]

\[ \varphi_{0,1}(x) = \varphi(x - 1) \]

\[ \varphi_{1,0}(x) = \sqrt{2} \varphi(2x) \]

\[ \varphi_{0,1}(x) = \sqrt{2} \varphi(2x - 1) \]

\[ f(x) \in V_1 \]

\[ \varphi_{0,0}(x) \in V_1 \]
Haar wavelet functions

\[ \psi(x) = \psi_{0,0}(x) \]

\[ \psi_{0,2}(x) = \psi(x - 2) \]

\[ \psi_{1,0}(x) = \sqrt{2} \phi(2x) \]

\[ f(x) \in V_1 = V_0 \oplus W_0 \]

\[ f_0(x) = V_0 \]

\[ f_d(x) = W_0 \]
Wavelet series expansion of $y = x^2$ using Haar wavelets
CWT and Fourier spectrum

\[ f(x) \]

\[ |F(\mu)| \]

\[ W_s(\mu, \tau) \]
Fast wavelet transform analysis bank
2-scale FWT analysis bank and frequency splitting characteristics
Example: d2-scale FWT of \{1, 4, -3, 0\} using Haar scaling and wavelet vectors

\[ W_\varphi(2, n) = f(n) = \{1, 4, -3, 0\} \]

\[ W_\psi(I, n) = \{5/\sqrt{2}, -3/\sqrt{2}\} \]

\[ W_\varphi(1, n) = \{-3/\sqrt{2}, -3/\sqrt{2}\} \]

\[ W_\psi(0, 0) = \{4\} \]

\[ W_\varphi(0, 0) = \{1\} \]

\[ \{2.5, 1, -1.5\} \]

\[ \{2.5, 4, -1.5\} \]
Inverse FWT synthesis filter bank
2-scale FWT inverse synthesis bank
Example

\[ W_\phi(0, 0) = \{4\} \]

\[ W_\psi(0, 0) = \{1\} \]

\[ W_\phi(1, n) = \{-3/\sqrt{2}, 0, -3/\sqrt{2}\} \]

\[ W_\psi(1, n) = \{5/\sqrt{2}, -3/\sqrt{2}\} \]

\[ f(n) = W_\psi(2, n) = \{1, 4, -3, 0\} \]
**FIGURE 7.23** Time-frequency tilings for the basis functions associated with (a) sampled data, (b) the FFT, and (c) the FWT. Note that the horizontal strips of equal height rectangles in (c) represent FWT scales.
2D FWT
FIGURE 7.25
Computing a 2-D three-scale FWT: (a) the original image; (b) a one-scale FWT; (c) a two-scale FWT; and (d) a three-scale FWT.
Symlets or symmetrical wavelets

- Have least asymmetry and highest number of vanishing moments for a given compact support
- Low pass reconstruction filter $g_0(n) = h\phi(n)$ for $0 \leq n \leq 7$
- Wavelets in image processing
- Step 1. compute a 2D wavelet transform of an image
- Step 2. Alter the transform
- Step 3. Compute the inverse transform
FIGURE 7.26
Fourth-order symlets: (a)–(b) decomposition filters; (c)–(d) reconstruction filters; (e) the one-dimensional wavelet; (f) the one-dimensional scaling function; and (g) one of three two-dimensional wavelets, \( \psi(x, y) \). See Table 7.3 for the values of \( h_p(n) \) for \( 0 \leq n \leq 7 \).
Orthonormal 4\textsuperscript{th} order symlet filter coefficients (Daubechies[1992])

<table>
<thead>
<tr>
<th>$n$</th>
<th>$h_\psi(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0322</td>
</tr>
<tr>
<td>1</td>
<td>-0.0126</td>
</tr>
<tr>
<td>2</td>
<td>-0.0992</td>
</tr>
<tr>
<td>3</td>
<td>0.2979</td>
</tr>
<tr>
<td>4</td>
<td>0.8037</td>
</tr>
<tr>
<td>5</td>
<td>0.4976</td>
</tr>
<tr>
<td>6</td>
<td>0.0296</td>
</tr>
<tr>
<td>7</td>
<td>-0.0758</td>
</tr>
</tbody>
</table>
Wavelet based edge detection

• Delineate signal and background
• Zeroing horizontal details-isolate the vertical edges
Wavelet based edge detection

FIGURE 7.27
Modifying a DWT for edge detection: (a) and (c) two-scale decompositions with selected coefficients deleted; (b) and (d) the corresponding reconstructions.
Wavelet based noise removal

• Step 1. choose wavelet and number of levels (scales) $P$, for decomposition. Compute FWT of the noisy image

• Step 2. threshold detail coefficients. Select and apply threshold to detail coefficients from scales $J-1$ to $J-P$. Hard thresholding: setting to zero the elements whose absolute values are $<$ the threshold and then scaling the nonzero coefficients toward 0.

• Step 3. compute the IWT using original approx. coefficients at level $J-P$ and modified detail coefficients for levels $J-1$ to $J-P$
 FIGURE 7.28
Modifying a DWT for noise removal: (a) a noisy CT of a human head; (b), (c) and (e) various reconstructions after thresholding the detail coefficients; (d) and (f) the information removed during the reconstruction of (c) and (e). (Original image courtesy Vanderbilt University Medical Center.)
FWTs using the wavelet toolbox

- \([\text{loD, HiD, loR, HiR}] = \text{wfilters}(\text{wname})\)
- \([f1, f2] = \text{wfilters}(\text{wname, type})\)
- \(\text{Waveinfo(wfamily)}\)
- \([\phi, \psi, xval] = \text{wavefun}(\text{wname, iter})\)
- \([\text{loD, HiD, loR, HiR}] = \text{wfilters}('\text{haar}')\)
- \(\text{Waveinfo('haar')}\)
Manipulating transform decomposition vector

• \( F = \text{magic}(8); \)
• \([c1,s1]=\text{wavedec2}(f,3,’haar’);\)
• \( \text{Size}(c1) \)
• \( \text{approx} = \text{appcoef2}(c1,s1,’haar’); \)
• \( \text{horizdet2} = \text{detcoef2}(’h’,c1,s1,2); \)
• \( \text{newc1} = \text{wthcoef2}(’h’,c1,s1,2); \)
• \( \text{Newhorizdet2} = \text{detcoef2}(’h’,\text{newc1},s1,2); \)
• \([c,s]=\text{wavedec2}(x,n,\text{loD},\text{hiD})\)
• \([c,s]=\text{wavedec2}(x,N,\text{wname})\)
• \(F=\text{magic}(4)\)
• \([c1,s1]=\text{wavedec2}(f,1,\text{’haar’})\)
• \([c2,s2]=\text{wavedec2}(f,2,\text{’haar’})\)
Displaying wavelet decomposition coefficients

- F = imread('vase.tif')
- \([c, s] = \text{wavefast}(f, 2, 'db4')\)
- \(\text{Wavedisplay}(c, s, )\);
- figure; \(\text{wavedisplay}(c, s, 8)\);
- Figure; \(\text{wavedisplay}(c, s, -8)\);
Inverse FWT

• \( G = \text{waverec2}(c, s, \text{wname}) \);
• \( G = \text{waverec2}(c, s, \text{loR}, \text{hiR}) \)
Wavelets in image processing
wavelet directionality and edge
detection

• \( f = \text{imread('A.tif')} \);
• \( \text{imshow}(f) \);
• \([c,s]=\text{wavefast}(f,1,'sym4');\)
• \( \text{figure; wavedisplay}(c,s,-6); \)
• \([nc,y]=\text{wavecut('a',c,s)};\)
• \( \text{figure; wavedisplay}(nc,s,-6); \)
• \( \text{edges=abs(waveback(nc,s,'sym4'))}; \)
• \( \text{figure; imshow(mat2gray(edges))}; \)
Using wavezero to generate increasingly smoothed versions

• \( f = \text{imread(‘A.tif’);} \)
• \([c,s]=\text{wavefast}(f,4,’\text{sym4’});\)
• \( \text{Wavedisplay}(c,s,20); \)
• \([c,g8]=\text{wavezero}(c,s,1,’\text{sym4’});\)
• \([c,g8]=\text{wavezero}(c,s,2,’\text{sym4’});\)
• \([c,g8]=\text{wavezero}(c,s,3,’\text{sym4’});\)
• \([c,g8]=\text{wavezero}(c,s,4,’\text{sym4’});\)
Progressive reconstruction

- \( f = \text{imread("Strawberries.tif");} \)
- \( \left[ c, s \right] = \text{wavefast}(f, 4, 'jpeg9.0') \)
- \( \text{Wavedisplay}(c, s, 8); \)
- \( f = \text{wavecopy('a', c, s);} \)
- \( \text{figure; imshow(mat2gray(f));} \)
- \( \left[ c, s \right] = \text{waveback}(c, s, 'jpeg9.7', 1); \)
- \( f = \text{wavecopy('a', c, s);} \)
- \( \text{figure; imshow(mat2gray(f));} \)
- \( \left[ c, s \right] = \text{waveback}(c, s, 'jpeg9.7', 1); \)
- \( f = \text{wavecopy('a', c, s);} \)
- \( \text{figure; imshow(mat2gray(f));} \)
- \( \left[ c, s \right] = \text{waveback}(c, s, 'jpeg9.7', 1); \)
- \( f = \text{wavecopy('a', c, s);} \)
- \( \text{figure; imshow(mat2gray(f));} \)
- \( \left[ c, s \right] = \text{waveback}(c, s, 'jpeg9.7', 1); \)
- \( f = \text{wavecopy('a', c, s);} \)
- \( \text{figure; imshow(mat2gray(f));} \)
- \( \left[ c, s \right] = \text{waveback}(c, s, 'jpeg9.7', 1); \)
- \( f = \text{wavecopy('a', c, s);} \)
- \( \text{figure; imshow(mat2gray(f));} \)