Integral and Discrete Transforms in Image Processing

2^{nd} Generation of Wavelets – Lifting Scheme

David Svoboda

email: svoboda@fi.muni.cz
Centre for Biomedical Image Analysis
Faculty of Informatics, Masaryk University, Brno, CZ

November 6, 2022

Outline

Motivation

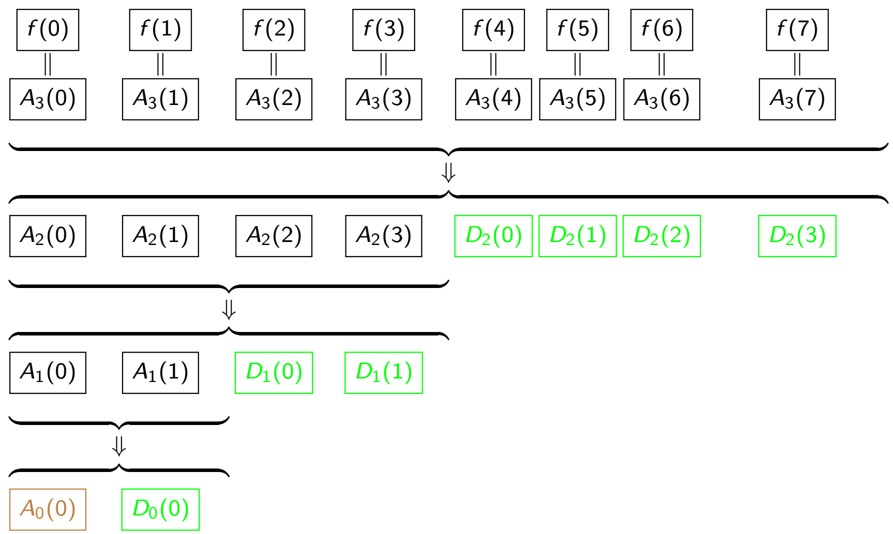
Practical use of DWT or FWT?

- Complexity of:
  - DWT: $O(n^2)$
  - FWT: $O(cn)$

  Can we do it faster?
- DWT/FWT are computed in floating point arithmetic.
  Can we restrict ourselves to integer number?
- DWT/FWT are computed *out-of-place*.
  Can we reduce the memory usage?

Analysis of FWT

Let $|f| = N = 8 = 2^3 = 2^J$ and $j_0 = 0$



Analysis of FWT

An example (Haar wavelets)

Let $f = A_j$ be an input signal:

$$A_{j-1} = \left(A_j * \left[\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right] \right) (\downarrow 2\times)$$
$$D_{j-1} = \left(A_j * \left[\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2} \right] \right) (\downarrow 2\times)$$

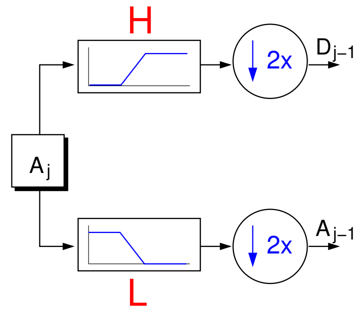
In each sample k, we perform one averaging and one difference:

$$A_{j-1}(k) = \left((A_j(k+1) + A_j(k)) \frac{\sqrt{2}}{2} \right) (\downarrow 2\times)$$
$$D_{j-1}(k) = \left((A_j(k+1) - A_j(k)) \frac{\sqrt{2}}{2} \right) (\downarrow 2\times)$$

Analysis of FWT

Fast Wavelet transform

- The filtering is followed by downsampling
- We throw away half of computed samples!



Optimization strategy

Let's flip the order of filtering and downsampling:

1. we start with downsampling
2. we proceed with filtering

Analysis of FWT

Downsampling in Z-domain

Let $f = [f_0, f_1, f_2, \ldots]$ be a signal and further let
$\Rightarrow f_{even} = [f_0, f_2, f_4, \ldots]$ and
$\Rightarrow f_{odd} = [f_1, f_3, f_5, \ldots]$ be its even and odd samples, respectively.
The same rule is applied to filter $g = [g_0, g_1, g_2, \ldots]$.

If $h = f * g$, then in Z-domain:

$$\begin{aligned} z^0 : h_0 &= f_0 g_0 \\ z^{-1} : h_1 &= f_1 g_0 + f_0 g_1 \\ z^{-2} : h_2 &= f_2 g_0 + f_1 g_1 + f_0 g_2 \\ z^{-3} : h_3 &= f_3 g_0 + f_2 g_1 + f_1 g_2 + f_0 g_3 \\ z^{-4} : h_4 &= f_4 g_0 + f_3 g_1 + f_2 g_2 + f_1 g_3 + f_0 g_4 \\ &\vdots \end{aligned}$$

Downsampling $H(z)$ by 2 removes each odd line, which results in:

$$H_{even}(z) = F_{even}(z) G_{even}(z) + z^{-1} F_{odd}(z) G_{odd}(z)$$

Analysis of FWT

Downsampling in Z-domain

Downsampling of A_j followed by filtering:

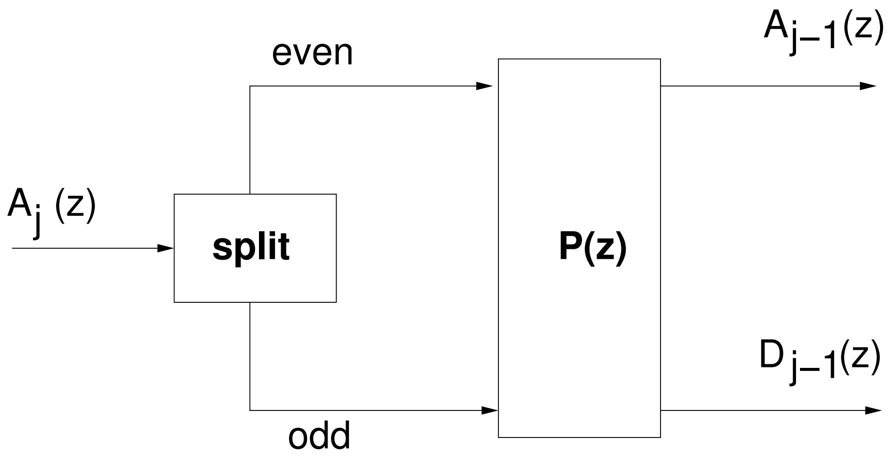
$$\begin{aligned} A_{j-1}(z) &= A_{j\,even}(z) L_{even}(z) + z^{-1} A_{j\,odd}(z) L_{odd}(z) \\ D_{j-1}(z) &= A_{j\,even}(z) H_{even}(z) + z^{-1} A_{j\,odd}(z) H_{odd}(z) \end{aligned}$$

In (polyphase) matrix notation:

$$\begin{bmatrix} A_{j-1}(z) \\ D_{j-1}(z) \end{bmatrix} = \begin{bmatrix} L_{even}(z) & L_{odd}(z) \\ H_{even}(z) & H_{odd}(z) \end{bmatrix} \cdot \begin{bmatrix} A_{j\,even}(z) \\ z^{-1} A_{j\,odd}(z) \end{bmatrix}$$
$$P(z) = \begin{bmatrix} L_{even}(z) & L_{odd}(z) \\ H_{even}(z) & H_{odd}(z) \end{bmatrix}$$

Analysis of FWT

The use of polyphase matrix



Lifting scheme

From Filters to Lifting

Algorithm invented by Wim Sweldens (1996):

1. Input: either 'LoD' and 'HiD' filters or φ and ϕ functions
2. Convert both filters 'LoD' and 'HiD' to Z-domain
3. Create *polyphase matrix* (2×2)
4. Factorize matrix into simple (lower and upper diagonal) matrices
   Each simple matrix corresponds either to one update or prediction step
5. Convert each matrix from Z-domain to time domain

Lifting scheme

Example: Haar Filters to Lifting

1. Low (l) and high (h) frequency decomposition filter ...

$$l = \left[\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right] \qquad h = \left[-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right]$$

with emphasizing reference positions:

$$l = \left[\boxed{\frac{\sqrt{2}}{2}}, \frac{\sqrt{2}}{2}\right] \qquad h = \left[\boxed{-\frac{\sqrt{2}}{2}}, \frac{\sqrt{2}}{2}\right]$$

2. Z-analysis of both filters:

$$L(z) = \frac{\sqrt{2}}{2}z^0 + \frac{\sqrt{2}}{2}z^{-1} \quad H(z) = -\frac{\sqrt{2}}{2}z^0 + \frac{\sqrt{2}}{2}z^{-1}$$

Lifting scheme

Example: Haar Filters to Lifting

2. Z-analysis of both filters (continued):

   Using the rule:

$$\mathrm{Filter}(z) = \mathrm{Filter}_e(z^2) + z^{-1}\mathrm{Filter}_o(z^2)$$

   we perform the separation of even and odd part of both filters:

$$L_e(z) = \frac{\sqrt{2}}{2}z^0 \quad L_o(z) = \frac{\sqrt{2}}{2}z^0$$

$$H_e(z) = -\frac{\sqrt{2}}{2}z^0 \quad H_o(z) = \frac{\sqrt{2}}{2}z^0$$

3. Creation of Polyphase matrix:

$$P(z) = \begin{bmatrix} L_e(z) & L_o(z) \\ H_e(z) & H_o(z) \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

Lifting scheme

Example: Haar Filters to Lifting

4. Factorization into lower diagonal matrix $\approx$ dual step:

$$P(z) = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} \overline{L_e}(z) & \frac{\sqrt{2}}{2} \\ \overline{H_e}(z) & \frac{\sqrt{2}}{2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \mathrm{Lower}(z) & 1 \end{bmatrix}$$

We get a set of equations:

$$\begin{aligned} \frac{\sqrt{2}}{2} &= \overline{L_e}(z) + \mathrm{Lower}(z)\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} &= \overline{H_e}(z) + \mathrm{Lower}(z)\frac{\sqrt{2}}{2} \end{aligned}$$

Solution (divison of polynomials – ambiguous!):

$$\mathrm{Lower}(z) = -1; \quad \overline{H_e}(z) = 0; \quad \overline{L_e}(z) = \sqrt{2}$$

$$P(z) = \begin{bmatrix} \sqrt{2} & \frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

Lifting scheme

Example: Haar Filters to Lifting

4. Factorization into upper diagonal matrix $\approx$ primal step:

$$P(z) = \begin{bmatrix} \sqrt{2} & \frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} \sqrt{2} & \overline{L_o}(z) \\ 0 & \overline{H_o}(z) \end{bmatrix} \cdot \begin{bmatrix} 1 & \mathrm{Upper}(z) \\ 0 & 1 \end{bmatrix}$$

We get a set of equations:

$$\begin{aligned} \frac{\sqrt{2}}{2} &= \sqrt{2} \cdot \mathrm{Upper}(z) + \overline{L_o}(z) \\ \frac{\sqrt{2}}{2} &= \overline{H_o}(z) \end{aligned}$$

Solution:

$$\mathrm{Upper}(z) = \frac{1}{2}; \quad \overline{L_o}(z) = 0$$

$$\begin{bmatrix} \sqrt{2} & \frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \cdot \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix}$$

Lifting scheme

Example: Haar Filters to Lifting

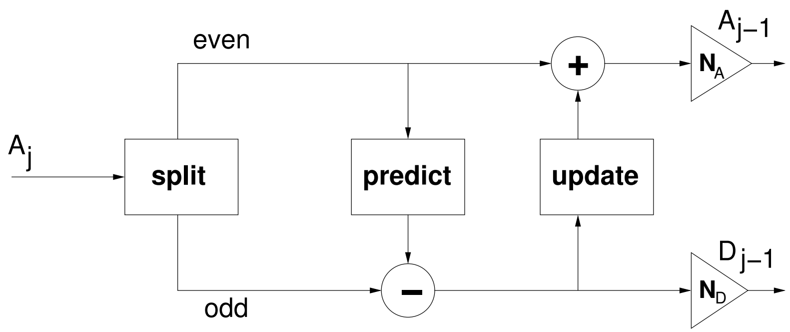
5. Finally

$$P(z) = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \cdot \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

$\begin{bmatrix} \sqrt{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \end{bmatrix}$	normalization	$N_A = \sqrt{2}, N_D = \frac{\sqrt{2}}{2}$
$\begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix}$	primal lifting step	$A_j \leftarrow A_j + \frac{1}{2}D_j$
$\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$	dual lifting step	$D_j \leftarrow D_j + (-1)A_j$

Lifting scheme

- Adopted idea of FastDWT
- The transition from level j to $j-1$ is computed efficiently
- Basic idea: split $\rightarrow$ predict $\rightarrow$ update $\rightarrow$ normalize

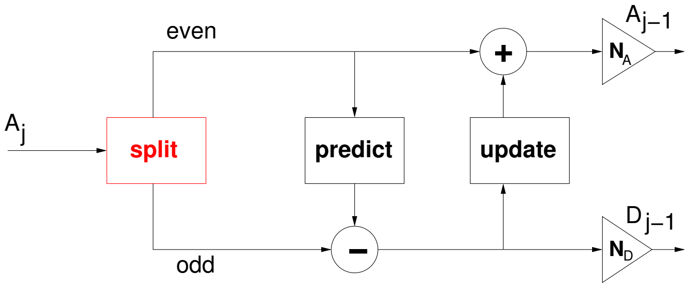


Lifting scheme

Split step

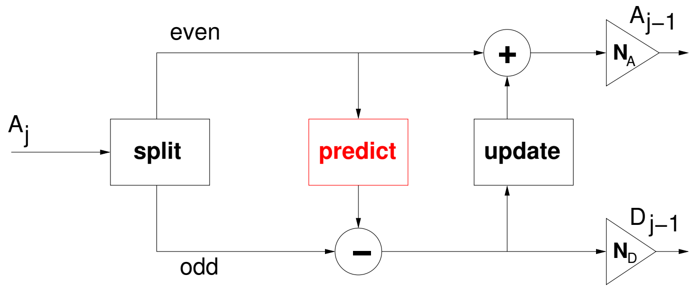
- Also known as *lazy wavelet*.
- The signal A_j is split into odd and even samples:

$$(A_{j-1}, D_{j-1}) \leftarrow split(A_j)$$



Lifting scheme

Prediction step

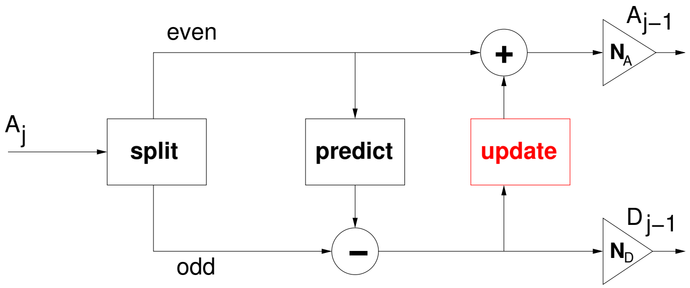


$$D_{j-1}(k) \leftarrow D_{j-1}(k) - predict(A_{j-1}(k))$$

- Also known as *dual lifting step*
- When using Haar wavelets the neighbouring samples are supposed to be equal, i.e. the predictor is simple: $predict_{Haar}(f(k)) = f(k)$
  $D_{j-1}(k) \leftarrow D_{j-1}(k) - A_{j-1}(k)$

Lifting scheme

Update step

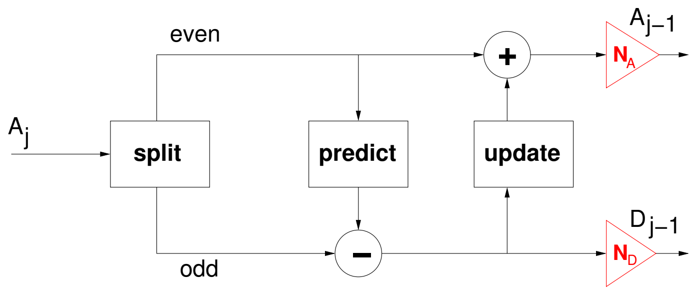


$$A_{j-1}(k) \leftarrow A_{j-1}(k) + update(D_{j-1}(k))$$

- Also known as *primal lifting step*
- Update step repairs the wrong estimate of the prediction step.
- When using Haar wavelet, we use: $update_{Haar}(f(k)) = \frac{1}{2}f(k)$
  $A_{j-1}(k) \leftarrow A_{j-1}(k) + \frac{1}{2}D_{j-1}(k)$

Lifting scheme

Normalization step



$$\begin{aligned} A_{j-1}(k) &\leftarrow A_{j-1}(k) \cdot N_A \\ D_{j-1}(k) &\leftarrow D_{j-1}(k) \cdot N_D \\ N_A \cdot N_D &= 1 \end{aligned}$$

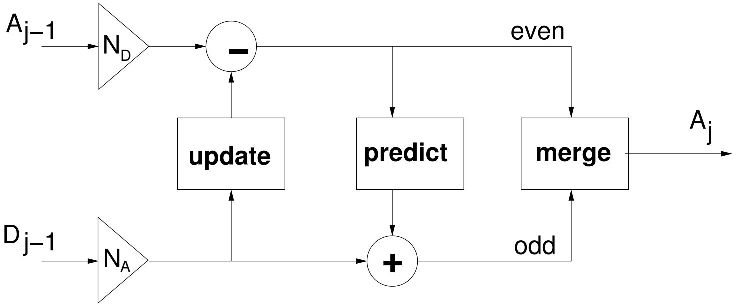
- The output signals are normalized to avoid boosting the signal.

Lifting scheme

Inverse lifting

- The forward algorithm can be simply inverted:

$$
\begin{aligned}
A_{j-1}(k) &\leftarrow A_{j-1}(k) - update(D_{j-1}(k)) \\
D_{j-1}(k) &\leftarrow D_{j-1}(k) + predict(A_{j-1}(k)) \\
A_j &\leftarrow merge(A_{j-1}, D_{j-1})
\end{aligned}
$$

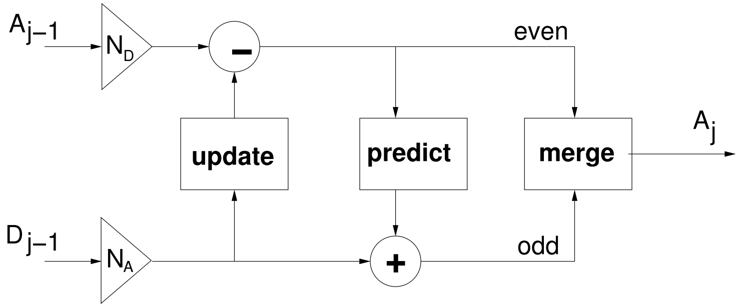


Lifting scheme

Inverse lifting (example)

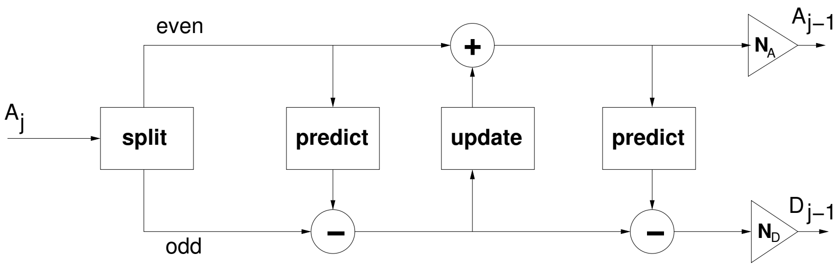
- Namely for Haar wavelets we get:

$$
\begin{aligned}
A_{j-1}(k) &\leftarrow A_{j-1}(k) \cdot N_D \\
D_{j-1}(k) &\leftarrow D_{j-1}(k) \cdot N_A \\
A_{j-1}(k) &\leftarrow A_{j-1}(k) - (D_{j-1}(k)/2) \\
D_{j-1}(k) &\leftarrow D_{j-1}(k) + A_{j-1}(k) \\
A_j &= merge(A_{j-1}, D_{j-1})
\end{aligned}
$$



Lifting scheme

From Filters to Lifting (Daubechies-2)



$$
\begin{aligned}
predict_1(f(k)) &= (-1.732)\, f(k) \\
update_1(f(k)) &= 0.433\, f(k) - 0.067\, f(k+1) \\
predict_2(f(k)) &= f(k-1)
\end{aligned}
$$

Lifting scheme

Technical/Implementation notes

Lifting ordering (for $N = 8$)

f(0)	f(1)	f(2)	f(3)	f(4)	f(5)	f(6)	f(7)
$A_2(0)$	$D_2(0)$	$A_2(1)$	$D_2(1)$	$A_2(2)$	$D_2(2)$	$A_2(3)$	$D_2(3)$
$A_1(0)$		$D_1(0)$		$A_1(1)$		$D_1(1)$	
$A_0(0)$				$D_0(0)$			

Notice: The computation is performed completedly *in-place*.

From filters to lifting scheme

- conversion 'LoD','HiD' → *update*(·), *predict*(·) always exists but is not unique
- conversion is performed in frequency domain via Z-transform (polyphase matrice factorization)

Lifting scheme

FWT and DWT comparison (examples)

Price of one decomposition level using DWT ($|f| = N$)

family of wavelets	multiplications	additions
Haar	$4N$	$2N$
Daubechies-2	$8N$	$6N$

Extra memory usage: one memory buffer of size $O(N)$ needed for convolution.

Price of one decomposition level using LS ($|f| = N$)

family of wavelets	multiplications	additions
Haar	$2N$	N
Daubechies-2	$3N$	$2N$

Extra memory usage: $\emptyset$

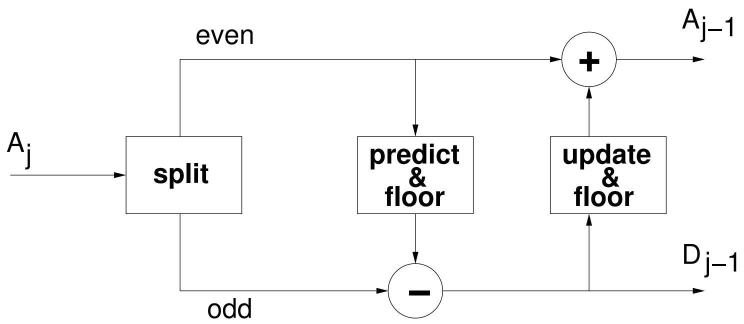
Integer Wavelet Transform

Basic idea & Properties

- IWT originates from lifting scheme (chain of predictions and updates).
- The fixed point arithmetic is guaranteed via `'floor'` function.
- The rounding error produced in forward transform is compensated by mirror `'floor'` in inverse lifting. They cancel out each other.
- The lifting is the same as the standard one but
  - each multiplication is followed but the truncation
  - no normalization is present

Integer Wavelet Transform

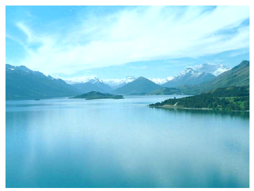
An example (Haar)



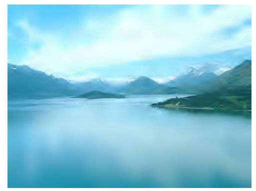
$$
\begin{aligned}
(A_{j-1}, D_{j-1}) &\leftarrow split(A_j) \\
D_{j-1}(k) &\leftarrow D_{j-1}(k) - floor\,(A_{j-1}(k)) \\
A_{j-1}(k) &\leftarrow A_{j-1}(k) + floor\,(D_{j-1}(k)/2)
\end{aligned}
$$

Applications

- Fusion of images with different resolution
- Image registration
- Edge detection
- Image compression:

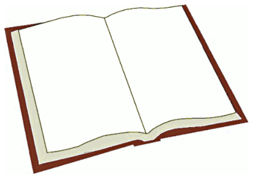

original (979 kB)


JPEG (6.21 kB)


JPEG2000 (1.83 kB)

Bibliography

- Li H., Manjunath B.S., Mitra S.K. Multisensor Image Fusion Using the Wavelet Transform, Graphical Models and Image Processing, Volume 57, Issue 3, May 1995, Pages 235-245, ISSN 1077-3169
- Jensen A., La Cour-Harbo A. Ripples in mathematics: the discrete wavelet transform, Springer, Berlin, 2001, ISBN 3-540-41662-5
- Sweldens W. The lifting scheme: A custom-design construction of biorthogonal wavelets. Applied and Computational Harmonic Analysis. 1996, Vol 3, Issue 2, pp 186–200



You should know the answers . . .

- Explain three principal phases of *lifting scheme*.
- Compare the time and spatial complexity of FWT and lifting scheme.
- Compute one level of wavelet transform (using Haar's basis) via lifting scheme for signal `f=[3 5 0 -1 4 2]`.
- What does *integer* wavelet transform mean?