Implementation of the DWT using Intel IA-32 SIMD Extensions

ENRIQUE P. MARTIN, JOSE SALVADO
Electrical Engineering Department
Escola Superior de Tecnologia - Instituto Politécnico de Castelo Branco
Av. do Empresário, S/N
PORTUGAL

Abstract: - This paper presents some results on the implementation of the DWT (DWT$^+$) through the lifting scheme by using general purpose processor SIMD extensions. We perform image analysis and reconstruction up to 3 levels of decomposition, using the DWT factored into lifting steps for the 9/7 wavelet filter pair. The algorithm was implemented in “C” code and evaluated in terms of performance and image degradation. Three approaches were used: floating-point representation, integer fixed-point representation and SIMD extensions integer code. The results obtained when compared to floating-point code implementation, indicate that the processing time for fixed-point is around 54% and SIMD extensions code is around 24.2%. The average PSNR results are also better for fixed-point and SIMD extensions than with floating-point code implementation.

Key-Words: - DWT, Lifting Scheme, SIMD Extension, Intel IA-32 GPP Architecture

1 Introduction

The DWT is a type of orthogonal transform used to decorrelate signals that registered a growing interest in the past few years, which has good localization properties in the time-frequency plan [1,2]. Another interesting feature of the DWT, and perhaps one of its most important ones, is its relation to signal multiresolution analysis and filter banks, which allows the construction of highly efficient embedded coding schemes like EZW, SPIHT and EBCOT; this later is used in JPEG2000 coding standard [3,4]. On the other hand, due to the growth in PC market and the demand for speed in media and graphics applications general purpose processors (GPP) adopted the concept of single instruction multiple-data (SIMD) parallel processing paradigm. Nowadays, the enhanced GPP architectures allow significant improvements in multimedia signal processing, namely for image and video signals, also for mobile applications [7–9].

This paper provides some results on the DWT (and DWT$^+$) implementation using different code implementation approaches, namely the SIMD extensions of today’s GPP processors namely the Intel IA-32 architecture.

2 Wavelet Analysis of Signals

In signal analysis, a signal $f(t)$ is represented as a weighted sum of building blocks of basis functions

$$f(t) = \sum_k c_k \psi_k(t)$$  \hspace{1cm} (1)

Since the basis functions are fixed, the information about the signal is carried by the coefficients. Choosing sinusoids as the basis functions in (1) yields the Fourier representation. To detect spikes or well localized high frequency components in the signal, one needs a representation which contains information about both, the time and frequency behaviour of the signal. However, resolution in time ($\Delta t$) and resolution in frequency ($\Delta\omega$) cannot be arbitrarily small at the same time as their product is lower bounded by the Heisenberg’s uncertainty principle.

$$\Delta t \cdot \Delta\omega \geq 1/2$$ \hspace{1cm} (2)

This means that one must trade off time resolution for frequency resolution, or vice versa. However, low-frequency events are usually spread in time (non local) and high-frequency events are usually concentrated (localized) in time. Therefore, it is possible to obtain good time-frequency information of a signal by choosing the basis functions to act as cascaded octave band-pass filters which repeatedly split the bandwidth of the signal in half. On the other hand sinusoids cannot provide information about the time behaviour of a signal as they have infinite support. The solution is to use basis functions that have finite (compact) support and different widths.

2.1 The Discrete Wavelet Transform

For a wavelet representation, the set of basis functions $\{\psi_k\}$ are scaled and translated versions of the same prototype function $\psi(t)$ known as mother wavelet. Scaling is achieved by multiplying $t$ by a
scale factor, normally a power of two, \( \psi(2^a t), a \in \mathbb{Z} \).

Since the prototype function has finite support, it can be scaled and translated to cover the all signal i.e. \( \psi(2^a t-b), b \in \mathbb{Z} \). The wavelet decomposition of a signal can be represented as

\[
f(t) = \sum_a \sum_b c_{ab} \psi_{ab}(t) \tag{3}
\]

Where \( \psi_{ab}(t) = 2^{a/2} \psi(2^a t - b) \) and \( c_{ab} \) are coefficients that can be computed through the wavelet transform.

The wavelet transform of a signal \( x(t) \) is defined as

\[
W_s(a,b) = \frac{1}{\sqrt{a}} \int_{\mathbb{R}} x(t) \psi^* \left( \frac{t-b}{a} \right) dt \tag{4}
\]

where \( a \in \mathbb{R}^+ \) is the scale or dilation parameter, \( b \in \mathbb{R} \) is the translation parameter, \( \psi^* \) denotes the complex conjugate of \( \psi(t) \) and \( 1/\sqrt{a} \) normalizes energy along scales. By taking the discrete values of the scale and translation parameters \( a = 2^{-j}, b = k 2^{-j} \) one can obtain the discrete wavelet transform (DWT) given by:

\[
w_{j,k} = W_s(2^{-j}, k 2^{-j}) = \frac{1}{\sqrt{2^{-j}}} \int_{\mathbb{R}} x(t) \psi^* \left( \frac{t-k 2^{-j}}{2^{-j}} \right) dt \tag{5}
\]

The basic idea of the wavelet transform is to exploit the high correlation structure present in most real life signals and build a sparse approximation. The correlation structure is typically local in space (or time) and frequency; the samples that are close each other are more correlated than ones that are far apart. Wavelet analysis has good scale-frequency and localization properties, allows the decomposition of signals with different levels of detail. For a multi-resolution decomposition scheme of a signal with \( p \) levels, one has to successively apply the analysis filters to the resulting approximation sub-sequence. This scheme can also be applied to images, through the lines and columns direction, thus resulting in a 2D hierarchical scheme of orthogonal sub-bands.

\[
\text{Fig. 1. Polyphase representation of the discrete wavelet transform.}
\]

### 2.2 The Lifting Scheme

Daubechies and Sweldens proposed and described how the discrete wavelet transform filters can be decomposed into a finite sequence of simple filtering steps and called this ladder structure Lifting Steps. Mathematically, this decomposition corresponds to a polyphase matrix representation of the wavelet (or sub-band) filters and its factorization into elementary matrices.

In the polyphase representation of the DWT of figure 1, in the analysis side, one has a pair of analysis filters \( \tilde{h} \) (low-pass) and \( \tilde{g} \) (high-pass) followed by sub-sampling by a factor of 2. On the synthesis side coefficients are first up-sampled and then passed through a pair of synthesis filters \( h \) and \( g \) (low and high pass, respectively). Consider that all filters are of FIR type (for more extensive sub-band transform). The polyphase representation of a filter \( h \) is given by

\[
h(z) = h_e(z^2) + z^{-1}h_o(z^2)
\]

Where \( h_e \) contains the even ordered coefficients, and \( h_o \) contains the odd ordered coefficients

\[
h_e(z) = \sum_k h_{2k}(z^{-k}) \quad \text{and} \quad h_o(z) = \sum_k h_{2k+1}(z^{-k})
\]

Then the polyphase matrix:

\[
P(z) = \begin{bmatrix} h_e(z) & g_r(z) \\ h_o(z) & g_o(z) \end{bmatrix}
\]

so that

\[
P(z^2) = \frac{1}{2} M(z) \begin{bmatrix} 1 & z \\ 1 & -z \end{bmatrix}
\]

One can define \( \tilde{P}(z) \) similarly.

The perfect reconstruction property is given by:

\[
P(z) \tilde{P}(z^{-1}) = \mathbf{I}
\]

\( P(z) \) and \( \tilde{P}(z) \) should contain only Laurent polynomials. Equation (11) implies \( \det P(z) \) and its inverse are both Laurent polynomials, which is only possible if \( \det P(z) \) is a monomial in \( z \).

The lifting scheme is an easy relationship between perfect reconstruction filter pairs \( (h,g) \) that have the same low-pass and high-pass filter. A filter pair \( (h,g) \) is complementary if the corresponding polyphase matrix with determinant one.

Any pair of complementary filters \( (h,g) \) can be factored into lifting steps using the Euclidean...
algorithm. First \( h_i(z) \) and \( h_n(z) \) have to be relatively prime because any common factor would also divide \( \det P(z) \) and we know that is 1. The Euclidean algorithm starts from \( h_i(z) \) and \( h_n(z) \) and the greatest common divisor (gcd) is a constant, \( k \):

\[
\begin{pmatrix}
  h_i(z) \\
  h_n(z)
\end{pmatrix} = \prod_{l=1}^{\infty} \begin{pmatrix}
  q_i(z) & 1 \\
  1 & 0
\end{pmatrix}^{k_l} \begin{pmatrix}
  1 & 0 \\
  0 & 1/k
\end{pmatrix}
\]  
(12)

Assume \( n \) is even (can work also with \( n \) odd, with little adjustment). Given a complementary filter pair \((h, g)\), then there always exist Laurent polynomials \( s_i(z) \) and \( t_i(z) \) for \( 1 \leq i \leq m \) and a constant \( k \) so we can rewrite (9) in the form:

\[
P(z) = \prod_{l=1}^{m} \begin{pmatrix}
  s_i(z) & 1 \\
  0 & 1
\end{pmatrix} \begin{pmatrix}
  t_i(z) & 1 \\
  0 & 1/k
\end{pmatrix}
\]  
(13)

In [5] Daubechies and Sweldens showed how to obtain the algorithms by factoring the wavelet transform into lifting steps for the orthogonal and bi-orthogonal wavelet families. They also showed that with the lifting scheme there is significant speed-up on DWT calculation over the standard implementation, ranging from around 54% for the Daubechies D4 or 64% for the 9/7 filter wavelet pair, up to 100% for the \((N, \tilde{N})\) interpolating.

### 3 SIMD Extensions for General Purpose Processors Architectures

By the mid 90’s, due to the growing on PC market and internet, and also because of the great demand for visual and graphic computing on multimedia applications, GPP manufacturers responded by incorporating the concept of SIMD parallelism into its RISC processors architectures, in order to improve multimedia signal processing capabilities [7–9]. Most general-purpose processors extended their Instruction-Set Architectures to include these features, e.g., Intel IA-32 (MMX and SSE/SSE2) and AMD K6/K7 (MMX and 3DNow!).

With SIMD technique the same operation is performed on multiple data elements, in parallel, thus allowing for significant speed-ups for many algorithms with inherent parallelism. On the other hand, the native data types in multimedia signals facilitate the data alignment requirement for SIMD operation.

To maintain full compatibility with other existing applications, GPP architectures use the floating-point registers as special registers for SIMD operations.

The data elements to work with those registers are commonly defined as packed (or compressed) data types. Each element within a packed data type is a fixed-point integer instead of floating-point data. The packed data types can have 64-bit (MMX) or 128-bit long for for the SSE/SSE2 and 3DNow! technologies. Thus, in SSE technology a 128-bit register can accommodate:

- 16 packed byte elements;
- 8 packed word elements (8 16-bit elements);
- 4 packed doubleword elements and
- 2 packed quadword (64-bit each);

The same applies for the MMX technology but reducing the register capabilities of packed data types to the half.

### 4 Lifting Scheme Implementation

In [5] the factored algorithm for the particular case of the 9/7 wavelet filter pair according to the lifting scheme, results into the steps shown in figure 2, for DWT and DWT\(^4\), respectively:

\[
a^{(0)}_{i} = x_{i1} \\
\]

\[
d^{(0)}_{i} = x_{i+1} \\
\]

\[
d^{(1)}_{i} = d^{(1)}_{i} + \alpha \cdot \left( a^{(0)}_{i} + a^{(0)}_{i+1} \right) \\
\]

\[
d^{(1)}_{i} = d^{(1)}_{i} - 1,5861 \cdot \left( a^{(0)}_{i} + a^{(0)}_{i+1} \right) \\
\]

\[
a^{(0)}_{i} = a^{(0)}_{i} + 0,8829 \cdot \left( a^{(1)}_{i} + a^{(1)}_{i+1} \right) \\
\]

\[
a^{(0)}_{i} = a^{(0)}_{i} + 0,4435 \cdot \left( a^{(2)}_{i} + a^{(2)}_{i+1} \right) \\
\]

\[
a^{(0)}_{i} = \frac{x_{i1}}{\varphi} \\
\]

\[
d^{(2)}_{i} = \frac{d^{(2)}_{i}}{\varphi} \\
\]

\[
d^{(2)}_{i} = 1,1496 \cdot d^{(2)}_{i} \\
\]

\[
a^{(0)}_{i} = \frac{x_{i1}}{\varphi} \\
\]

\[
d^{(2)}_{i} = \frac{d^{(2)}_{i}}{\varphi} \\
\]

\[
d^{(2)}_{i} = 1,1496 \cdot d^{(2)}_{i} \\
\]

\[
x_{i1} = a^{(0)}_{i} \\
\]

\[
x_{i+1} = d^{(0)}_{i} \\
\]

\[
\]

Fig. 2. Lifting scheme algorithm to calculate the: a) DWT and b) DWT\(^4\).
We implemented this algorithm in a “C” language, on a PC equipped with an Intel Pentium III processor at 800 MHz [12]. The algorithm was implemented with floating-point data representation and with integer types, using fixed-point data representation [10,11] and SIMD extensions.

To support the code development, namely for MMX/SSE intrinsic instructions support, we used Microsoft Visual Studio.Net 2005 IDE and measured code performance in terms of execution time with Intel® VTune™ Performance Analyzer v8.0 (integrated on Microsoft Visual Studio.Net 2005). We also evaluate the effects of each implementation on image the image using the peak signal to noise ratio (PSNR) given by:

$$\text{PSNR} = 10 \cdot \log_{10} \left( \frac{1}{JK} \sum_{i=1}^{K} \sum_{j=1}^{K} (F(j,k) - \hat{F}(j,k))^2 \right)$$  \hspace{1cm} (14)$$

where $F(j,k)$ is the original image and $\hat{F}(j,k)$ relates to the recovered image.

For image analysis the code of the 1D lifting DWT is expanded to 2D. For the sake of regularity this is first applied through lines and than trough columns, after matrix transposition. In order to give enough support to the algorithm, and also to avoid extra degradation in the region bounds, the matrix is symmetrically expanded, prior the DWT and DWT$^{-1}$ in each level.

5 Some Evaluation Results

Test results simply considered image analysis and reconstruction up to 3 levels of decomposition with the 9/7 filter DWT. We used the test images “barbara”, “boat” and “thorax_mri”, all in bitmap format, with spatial resolution 256×256 and 8-bits per pixel (grey-scale).

For the performance figure we considered 10 runs for each code implementation (floating-point, fixed-point and SIMD extensions) then discard the best and the worst results. The results obtained for each run are shown in table I. The results on image degradation, for different images, with 3 levels of analysis, are shown in table II. These tests simply include image analysis and reconstruction with 3 levels of decomposition, using the different code approaches, and measured the correspondent PSNR values (in dB).

From the results of table II one can notice that in general the image quality is better achieved for the fixed-point or SIMD implementations. One also notice that PSNR decreases as the number of levels increases. The difference in PSNR form floating to fixed point implementation is about 6 dB in each image, considering the same number of decomposition levels. This difference in PSNR is justified by the successive rounds from floating-point to integers in each level, and for the several lifting schemes phases, both in the DWT and in DWT$^{-1}$ as the lifting scheme is an in-place scheme, and therefore does not require additional memory.

5 Conclusions

This paper presents the implementation of the DWT factored into lifting steps using general purpose processors architectures features, namely the SIMD extensions, in order to improve performance. Test results show that the SIMD extensions approach allows a speed-up of 4 over the floating-point implementation and that the improvement is only around 1.8 for fixed-point implementation. The quality of the images, taken then PSNR measure, are also about 6 dB better for fixed-point or SIMD implementations for each image and for every level of decomposition.
Acknowledgment:
This work was carried out to fulfill the requirements for a graduation project at EST-IPCB, Portugal, under a EE.UU Socrates/Erasmus student mobility program.

References:


