Rodrigo Capobianco Guido

Practical and Useful Tips on Discrete Wavelet Transforms

The discrete wavelet transform (DWT) [1] is one of the most powerful tools for time-frequency signal analysis. Its applicability is extremely relevant in various areas of science, as exemplified in [2], with digital signal processing (DSP) as the most notable one. After teaching this topic for many years, I have noted that neither young DSP students nor experienced researchers have perceived several interesting aspects of the DWT from a practical point of view. Thus, the objective of this article is to construe such relevant aspects, providing useful tips to calculate the transform in one (1-D) and two (2-D) dimensions. The entire discussion is also valid to the discrete wavelet-packet transform (DWPT), which extends the decomposition carried out by the DWT so that a finer time-frequency analysis takes place, and also to the discrete shapelet transform (DST) [3], which extends the properties of the DWT and DWPT so that a joint time-frequency-shape signal analysis becomes possible.

THE TIPS

The tips given herein come after a short review, which is necessary to understand them.

1-D DWT CALCULATION

Mallat's algorithm [4] is the commonly used method to calculate the DWT of a certain discrete-time signal $(f_{[\cdot]})$ of length N. The procedure, consisting of a simple matrix multiplication, spans both the required convolutions of $f_{[\cdot]}$ with $h_{[\cdot]}$ and with $g_{[\cdot]}$, which represent, respectively, a low-pass filter and a high-pass filter, and the downsamplings by two, all possible due to the reduction of the signals bandwidths [1]. Filters $h_{[\cdot]}$ and $g_{[\cdot]}$ form the analysis filter pair, which have, most of the time, the same length M, and form a quadrature mirror filter (QMF) [4] set that presents an orthogonality condition and half-band cutoff frequencies. The complete process to transform $f_{[\cdot]}$ is shown in the boxed equation at the bottom of the page.

As the matrix $A_{[\cdot][\cdot]}$, formed by the filters coefficients, advances from the first pair of lines until the last one, a shift to the right becomes necessary so that $h_{[\cdot]}$ and $g_{[\cdot]}$ start to be written two positions ahead in each subsequent pair. In case some of the coefficients fall beyond the length of a row, e.g., at the bottom of matrix $A_{[\cdot][\cdot]}$, they are pushed back at the beginning of the same row and the remaining positions are set to zero. This is known as *wrap-around*. The procedure explained consists of the way the convolutions and the downsamplings were implemented.

After performing the calculations above, the resulting DWT corresponds to the concatenation of the subsignal formed by the even indexes of $r_{[\cdot]}$, designated *approximation* of length $(N/2)$, followed by that formed by the odd indexes of $r_{[\cdot]}$, designated *detail* of length $(N/2)$. Thus, DWT $(f_{[\cdot]}) = \{r_0, r_2, r_4, r_{N-2}..., r_1, r_3, r_5, ..., r_{N-1}\}$, registering the same length of the input signal, i.e., N. This corresponds to the first-level DWT $(f_{[\cdot]})$. If the detail subsignal is kept intact and the approximation subsignal is considered as being a new input to the same algorithm, then two other subsignals of length $(N/4)$ are obtained. Their concatenation, following the same process described above, generates one subsignal of length $(N/2)$ that replaces the original input of the same length. The new complete signal is of length N and corresponds to the second-level DWT $(f_{[\cdot]})$. The process can be repeated $(\log(N)/\log(2))$ times, i.e., until the length of the approximation subsignal at the current level becomes 1.

The aforementioned process is one we can easily find in literature, however, if the reader tries to calculate a DWT by hand, to check if he or she has learned the algorithm correctly, or even try to implement a computer software to perform the calculations, a problem may appear: as the level of decomposition advances, the approximation subsignal, used as input, reduces its length to half. How can one proceed with the calculations if the dimension of the input becomes lesser than the filters support-size? For example, letting $N = 8$ and $M = 4$, then, after the second-level

$$\underbrace{\begin{pmatrix} h_0 & h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & \cdots & h_{M-1} & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ g_0 & g_1 & g_2 & \cdots & \cdots & \cdots & \cdots & \cdots & g_{M-1} & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & \cdots & \cdots & \cdots & \cdots & \cdots & h_{M-1} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdots & \cdots & \cdots & \cdots & \cdots & g_{M-1} & 0 & \cdots & \cdots & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ h_2 & h_3 & \cdots & \cdots & \cdots & \cdots & h_{M-1} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & h_0 & h_1 \\ g_2 & g_3 & \cdots & \cdots & \cdots & \cdots & g_{M-1} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & g_0 & g_1 \end{pmatrix}}_{\text{matrix } A_{[\cdot][\cdot]}} \cdot \underbrace{\begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ \cdots \\ \cdots \\ \cdot \\ \cdot \\ \cdot \\ f_{N-2} \\ f_{N-1} \end{pmatrix}}_{\text{input}(f_{[\cdot]})} = \underbrace{\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ \cdots \\ \cdots \\ \cdot \\ \cdot \\ \cdot \\ r_{N-2} \\ r_{N-1} \end{pmatrix}}_{\text{output}(r_{[\cdot]})}.$$

Digital Object Identifier 10.1109/MSP.2014.2368586
Date of publication: 6 April 2015

decomposition, the input subsignal to be used in the third level has length two, i.e, the multiplication is not possible because matrix $A_{[\cdot][\cdot]}$ and the input are not compatible in terms of length. Some of us may rapidly respond: proceed with a zero-padding in the input so that it reaches, at least, the filters support-size. This solution is, however, equivalent to simply discarding some of the filters coefficients, because they will be multiplied by zero, therefore, it is incorrect. The right solution to this problem is to repeat the input so many times as necessary until it reaches the minimum required length so that the matrix multiplication becomes possible.

NUMERICAL EXAMPLE:

Let $f_{[\cdot]} = \{1, 2, 3, 4, 4, 3, 2, 1\}$, which is a discrete-time signal of length $N = 8$, and let $h_{[\cdot]} = \{(1+\sqrt{3})/(4\sqrt{2}), (3+\sqrt{3})/(4\sqrt{2}), (3-\sqrt{3})/(4\sqrt{2}), (1-\sqrt{3})/(4\sqrt{2})\}$ and $g_{[\cdot]} = \{(1-\sqrt{3})/(4\sqrt{2}), (-3+\sqrt{3})/(4\sqrt{2}), (3+\sqrt{3})/(4\sqrt{2}), (-1-\sqrt{3})/(4\sqrt{2})\}$, which correspond to the Daubechies-4 filter pair [4]. The first and second levels of decomposition produce, respectively, the results shown in the boxed equation at the top of the page.

Performing the third-level decomposition, by using the subsignal $\{(156+28\sqrt{3})/32, (164-28\sqrt{3})/32\}$ as input, yields the boxed equation at the bottom of the page, which corresponds to the correct procedure. The reader can note that, when this tip is applied, $A_{[\cdot][\cdot]}$ is not a squared matrix. Instead, it contains only the required number of lines to obtain the number of expected coefficients in the resulting signal, i.e., two in this example. The calculations above result in:

$$\mathrm{DWT}_{\mathrm{level3}} = \left\{5\sqrt{2}, \frac{7\sqrt{3}-1}{4\sqrt{2}}, \frac{4+28\sqrt{3}}{32}, \frac{-4-28\sqrt{3}}{32}, 0, \frac{2\sqrt{3}}{4\sqrt{2}}, 0, \frac{-2\sqrt{3}}{4\sqrt{2}}\right\}.$$

In case there is a need to invert the transformations at each level, the synthesis filter bank is adopted. It is composed of the filters $\bar{h}_{[\cdot]}$ and $\bar{g}_{[\cdot]}$, which are respectively defined as being $\bar{h}_k = h_{M-k-1}$ and $\bar{g}_k = (-1)^{k+1} h_k$. An easier way to implement the inverse DWT (IDWT) is to note that the matrix $A_{[\cdot][\cdot]}$ is orthogonal. Therefore, its inverse, which allows the process to be inverted, corresponds exactly to its transpose. Examples can be found in [4].

$$\mathrm{DWT}_{\mathrm{level1}} = \left\{\frac{20-4\sqrt{3}}{4\sqrt{2}}, \frac{30}{4\sqrt{2}}, \frac{20+4\sqrt{3}}{4\sqrt{2}}, \frac{10}{4\sqrt{2}}, 0, \frac{2\sqrt{3}}{4\sqrt{2}}, 0, \frac{-2\sqrt{3}}{4\sqrt{2}}\right\} \text{ and}$$

$$\mathrm{DWT}_{\mathrm{level2}} = \left\{\frac{156+28\sqrt{3}}{32}, \frac{164-28\sqrt{3}}{32}, \frac{4+28\sqrt{3}}{32}, \frac{-4-28\sqrt{3}}{32}, 0, \frac{2\sqrt{3}}{4\sqrt{2}}, 0, \frac{-2\sqrt{3}}{4\sqrt{2}}\right\}.$$

2-D-DWT CALCULATION

The definition of the 2-D-DWT can be found in [4]. According to it, an N by N matrix, usually representing a digital image, is used as input while the transformation output consists of another matrix with the same dimensions. This output is composed of the concatenation of four submatrices: $a_{[\cdot][\cdot]}$, $p_{[\cdot][\cdot]}$, $v_{[\cdot][\cdot]}$, and $d_{[\cdot][\cdot]}$, which correspond, respectively, to the approximations of the lines followed by the approximations of the columns, to the approximations of the lines followed by the details of the columns, to the details of the lines followed by the approximations of the columns, and to the details of the lines followed by the details of the columns. Assuming that $S_{[\cdot][\cdot]}$ is the $N \times N$ input matrix, its 2-D-DWT is

$$\begin{pmatrix} a_{[\cdot][\cdot]} & p_{[\cdot][\cdot]} \\ \left(\frac{N}{2} \times \frac{N}{2}\right) & \left(\frac{N}{2} \times \frac{N}{2}\right) \\ v_{[\cdot][\cdot]} & d_{[\cdot][\cdot]} \\ \left(\frac{N}{2} \times \frac{N}{2}\right) & \left(\frac{N}{2} \times \frac{N}{2}\right) \end{pmatrix}.$$

An interesting algorithm to calculate 2-D-DWT$(S_{[\cdot][\cdot]})$ directly, instead of calculating the four submatrices separately, is the following:

1) obtain $R_{[\cdot][\cdot]} = A_{[\cdot][\cdot]} S_{[\cdot][\cdot]} A_{[\cdot][\cdot]}^T$, being $A_{[\cdot][\cdot]}^T$ the transpose of $A_{[\cdot][\cdot]}$
2) rearrange the elements of $R_{[\cdot][\cdot]}$ so that
- $a_{[\cdot][\cdot]}$ is formed by the elements of $R_{[\cdot][\cdot]}$, which are on the even rows and even columns (starting at line and column zero)
- $p_{[\cdot][\cdot]}$ is formed by the elements of $R_{[\cdot][\cdot]}$, which are on the even rows and odd columns
- $v_{[\cdot][\cdot]}$ is formed by the elements of $R_{[\cdot][\cdot]}$, which are on the odd rows and even columns
- $d_{[\cdot][\cdot]}$ is formed by the elements of $R_{[\cdot][\cdot]}$, which are on the odd rows and odd columns.

The rearranged matrix corresponds to 2-D-DWT$(S_{[\cdot][\cdot]})$. Particularly, the aforementioned procedure corresponds to the first-level 2-D-DWT. To obtain the next levels, the submatrix $a_{[\cdot][\cdot]}$ of the current level is used as the new input and the entire process is repeated. In the calculations, matrix $A_{[\cdot][\cdot]}$ is exactly the same defined for the 1-D-DWT. As previously explained, wraparounds and repetitions of the input signal may be required.

The 2-D-IDWT can be obtained by the inverse algorithm, i.e.,

1) rearrange the elements of 2-D-DWT $(S_{[\cdot][\cdot]})$ so that:
- $a_{[\cdot][\cdot]}$ is now formed by the elements of 2-D-DWT$(S_{[\cdot][\cdot]})$, which are on the even rows and even columns
- $p_{[\cdot][\cdot]}$ is now formed by the elements of 2-D-DWT$(S_{[\cdot][\cdot]})$, which are on the even rows and odd columns
- $v_{[\cdot][\cdot]}$ is now formed by the elements of 2-D-DWT$(S_{[\cdot][\cdot]})$, which are on the odd rows and even columns

$$\underbrace{\begin{pmatrix} \frac{1+\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} & \frac{3-\sqrt{3}}{4\sqrt{2}} & \frac{1-\sqrt{3}}{4\sqrt{2}} \\ \frac{1-\sqrt{3}}{4\sqrt{2}} & \frac{-3+\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} & \frac{-1-\sqrt{3}}{4\sqrt{2}} \end{pmatrix}}_{\text{matrix } A_{[\cdot][\cdot]}} \cdot \begin{pmatrix} \frac{156+28\sqrt{3}}{32} \\ \frac{164+28\sqrt{3}}{32} \\ \frac{156+28\sqrt{3}}{32} \\ \frac{164+28\sqrt{3}}{32} \end{pmatrix} \begin{matrix} \left.\vphantom{\frac{1}{1}}\right\}\text{input} \\ \left.\vphantom{\frac{1}{1}}\right\}\text{input repeated once} \end{matrix}$$

$$R_{[\cdot][\cdot]}=\begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0\\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0\\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}}\end{pmatrix}\cdot\begin{pmatrix}4 & 6 & 10 & 12\\ 2 & 6 & 8 & 12\\ 1 & 4 & 6 & 7\\ 0 & 3 & 2 & 1\end{pmatrix}\cdot\begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0\\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0\\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}}\end{pmatrix}=\begin{pmatrix}9 & -3 & 21 & -3\\ 1 & 1 & 1 & 1\\ 4 & -3 & 8 & 0\\ 1 & 0 & 5 & -1\end{pmatrix}.$$

- $d_{[\cdot][\cdot]}$ is now formed by the elements of 2-D-DWT($S_{[\cdot][\cdot]}$), which are on the odd rows and odd columns

2) The rearranged matrix corresponds to $R_{[\cdot][\cdot]}$. The original signal is $S_{[\cdot][\cdot]}=A_{[\cdot][\cdot]}{}^{T}R_{[\cdot][\cdot]}A_{[\cdot][\cdot]}$.

NUMERICAL EXAMPLE

Let

$$S_{[\cdot][\cdot]}=\begin{pmatrix}4 & 6 & 10 & 12\\ 2 & 6 & 8 & 12\\ 1 & 4 & 6 & 7\\ 0 & 3 & 2 & 1\end{pmatrix},\ h[\cdot]=\left\{\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}}\right\}$$

and $g[\cdot]=\{1/\sqrt{2},-1/\sqrt{2}\}$. To obtain 2-D-DWT ($S_{[\cdot][\cdot]}$), we need to calculate $R_{[\cdot][\cdot]}=A_{[\cdot][\cdot]}S_{[\cdot][\cdot]}A_{[\cdot][\cdot]}{}^{T}$, i.e., the boxed equation at the top of the page.

When rearranging the elements of this last matrix, we get:

$$\begin{pmatrix}9 & 21 & -3 & -3\\ 4 & 8 & -3 & 0\\ 1 & 1 & 1 & 1\\ 1 & 5 & 0 & -1\end{pmatrix},$$

which corresponds to the 2-D-DWT ($S_{[\cdot][\cdot]}$). The inversion is directly based on the algorithm above.

CONCLUSIONS

This article offered and subsequently described tips on the 1-D and 2-D DWT calculations that were not yet been documented in literature in a practical way. The information presented herein can certainly help young students and experienced researchers in using this important tool for time-frequency signal analysis. I have developed a C/C++ source code that implements the DWT, the IDWT, the DWPT, and the inverse DWPT (both 1-D and 2-D) containing examples of usage, and it is freely available. Please send your request to guido@ieee.org. Additional tips I wrote on DWT and DWTP can be found in [5].

AUTHOR

Rodrigo Capobianco Guido (guido@ieee.org) is a professor at São Paulo State University in São José do Rio Preto, Brazil.

REFERENCES

[1] P. Addison, J. Walker, and R. C. Guido, "Time-frequency analysis of biosignals: A wavelet transform overview," *IEEE Eng. Med. Biol. Mag.*, vol. 28, no. 5, pp. 14–29, 2009.

[2] S.-H. Chen, R. C. Guido, T.-K. Truong, and Y. Chang, "Improved voice activity detection algorithm using wavelet and support vector machine," *Comput. Speech Lang.*, vol. 24, no. 3, pp. 531–543, 2010.

[3] R. C. Guido, S. Barbon Jr., L. S. Vieira, F. L. Sanchez, C. D. Maciel, J. C. Pereira, P. R. Scalassara, and E. S. Fonseca, "Introduction to the discrete shapelet transform and a new paradigm: Joint time-frequency-shape analysis," in *Proc. IEEE Int. Symp. Circuits and Systems (IEEE ISCAS 2008)*, Seattle, WA, vol. 1, pp. 2893–2896.

[4] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge Press, 1997.

[5] R. C. Guido, "A note on a practical relationship between filters coefficients and the scaling and wavelet functions of the discrete wavelet transform," *Appl. Math. Lett.*, vol. 24, no. 7, pp. 1257–1259, 2011.

[SP]