This is an expanded version of the 'Quick Study' in *Physics Today* magazine, October, 2007.

Wavelet Transforms — A Quick Study

Ivan W. Selesnick
Polytechnic University
Brooklyn, NY

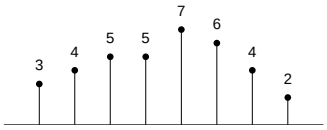
September 27, 2007

The wavelet transform has become a useful computational tool for a variety of signal and image processing applications. For example, the wavelet transform is useful for the compression of digital image files; smaller files are important for storing images using less memory and for transmitting images faster and more reliably. The FBI uses wavelet transforms for compressing digitally scanned fingerprint images. NASA's Mars Rovers used wavelet transforms for compressing images acquired by their 18 cameras. The wavelet-based algorithm implemented in software onboard the Mars Rovers is designed to meet the special requirements of deep-space communication. In addition, JPEG2K (the newer JPEG image file format) is based on wavelet transforms. Wavelet transforms are also useful for 'cleaning' signals and images (reducing unwanted noise and blurring). Some algorithms for processing astronomical images, for example, are based on wavelet and wavelet-like transforms.

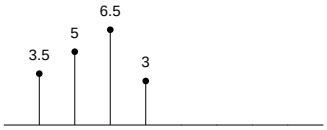
This Quick Study describes the wavelet transform, illustrates why it is effective for noise reduction, and briefly describes several improvements of the basic wavelet transform and basic noise reduction method used in the illustration. We describe what the wavelet transform is, and we describe algorithms for processing a signal after its wavelet transform has been computed. First we should point out that there are two basic types of wavelet transform. One type of wavelet transform is designed to be easily reversible (invertible); that means the original signal can be easily recovered after it has been transformed. This kind of wavelet transform is used for image compression and cleaning (noise and blur reduction). Typically, the wavelet transform of the image is first computed, the wavelet representation is then modified appropriately, and then the wavelet transform is reversed (inverted) to obtain a new image. The second type of wavelet transform is designed for signal *analysis*; for example, to detect faults in machinery from sensor measurements, to study EEG or other biomedical signals, to determine how the frequency content of a signal evolves over time. In these cases, a modified form of the original signal is not needed and the wavelet transform need not be inverted (it can be done in principle, but requires a lot of computation time in comparison with the first type of wavelet transform). In this Quick Study we will focus on those wavelet transforms that are easily invertible.

The most basic wavelet transform is the Haar transform described by Alfred Haar in 1910. It serves as the prototypical wavelet transform. We will describe the (discrete) Haar transform, as it

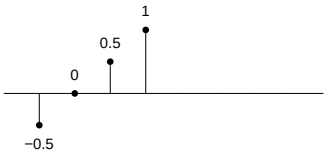
encapsulates the basic concepts of wavelet transforms used today. We will also see its limitation, which the newer wavelet transform (in 1998 by Ingrid Daubechies) resolves. The basis of the Haar transform is the decomposition of a signal, say the eight-point signal $x(n)$,



into two four-point signals. One being the average of pairs of signal values, $c(n)$:



The other signal being their differences, $d(n)$:



This decomposition can be written as

$$c(n) = 0.5\,x(2n) + 0.5\,x(2n+1)$$
$$d(n) = 0.5\,x(2n) - 0.5\,x(2n+1)$$

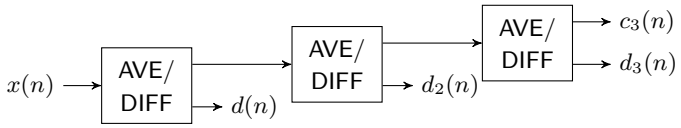
and represented by a block diagram:

It should be clear that this decomposition can be reversed. The original signal $x(n)$ can be reconstituted from the two shorter signals using

$$y(2n) = c(n) + d(n)$$
$$y(2n+1) = c(n) - d(n)$$

which we represent by a block diagram

When we repeat the simple AVE/DIFF signal decomposition procedure a number of times, each time on the average signal $c(n)$, we get the Haar transform. For our toy example eight-point signal, the AVE/DIFF decomposition can be repeated up to three times. In this case the Haar transform can be expressed clearly in block diagram form as



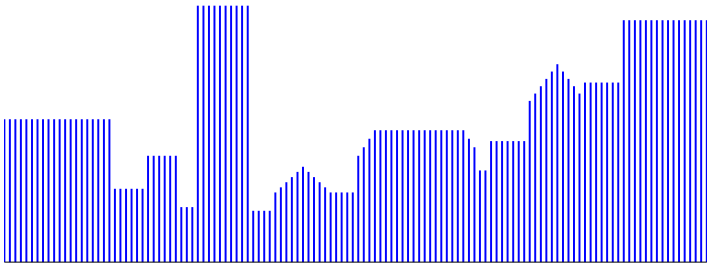
The Haar wavelet representation of the eight-point signal $x(n)$ is simply the set of four output signals produced by this three-level operation. Four our example, the four signals (or vectors) have lengths 1, 1, 2, and 4. Specifically,

$$
\begin{aligned}
c_3 &= [4.5] \\
d_3 &= [-0.25] \\
d_2 &= [-0.75, \quad 1.75] \\
d &= [-0.5, \quad 0, \quad 0.5, \quad 1]
\end{aligned}
$$

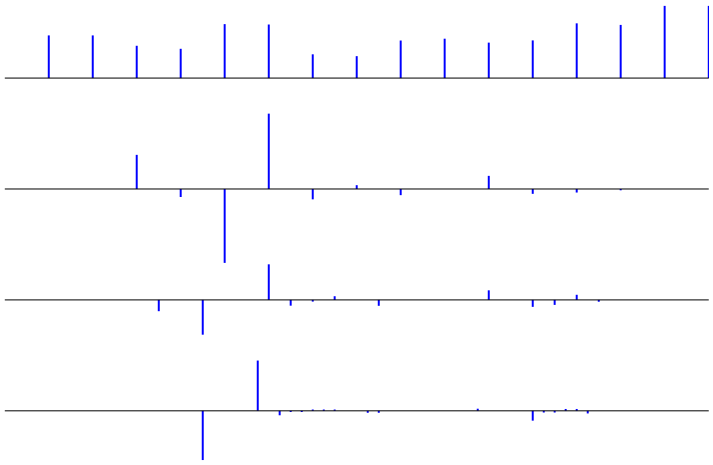
These values, taken together, are called the 'wavelet representation' of the signal x. Note that the single value c_3 is simply the average value of the original signal values, $x(n)$.

In general, if the original signal is of length 2^M, then the decomposition can be repeated up to M times. (In case the length of the original signal is not a power of two, there are several way to accommodate that.) The Haar transform can be very easily reversed by successive use of the INV block.

If we take the three-level Haar transform of the 128-point signal:



we obtain the following wavelet representation:



which can be reversed to retrieve the original signal. The three-level wavelet representation consists of four vectors of lengths 16, 16, 32, and 64. Notice that many of the values of the wavelet representation are small in value; we have what is called a *sparse* representation of the signal. That property is precisely what makes the wavelet transform useful for compressing images. For compression, the preponderance of zeros (in practice, small values) in the wavelet representation means that fewer bits need to stored in memory. Admittedly, the signal we have used here is chosen so as to exaggerate this property. Specifically, the signal we have used is constant over much its extent, and therefore the differencing part of the AVE/DIFF decomposition produces zeros. The large values in the wavelet representation are produced by the discontinuities in the signal and by those parts of the signal that are not constant. The Haar transform works well (provides a relatively sparse wavelet representation) for signals that are approximately piecewise constant. For more general (and more commonly encountered) piecewise-*smooth* signals (not necessarily piecewise-constant) one must use the newer (1988) wavelet transforms to obtain sparse wavelet representations.

In 1988 Daubechies constructed a family of easily implemented and easily invertible wavelet transforms that, in a sense, generalize the Haar transform. Like the Haar transform, the wavelet transform is implemented as a succession of decompositions. The only difference is that the AVE/DIFF decomposition is replaced by a new one. For the simplest of the Daubechies wavelet transforms, the new decomposition is given by

$$c(n) = h_0\, x(2n) + h_1\, x(2n+1) + h_2\, x(2n+2) + h_3\, x(2n+3)$$
$$d(n) = h_3\, x(2n) - h_2\, x(2n+1) + h_1\, x(2n+2) - h_0\, x(2n+3)$$

where the multipliers are:

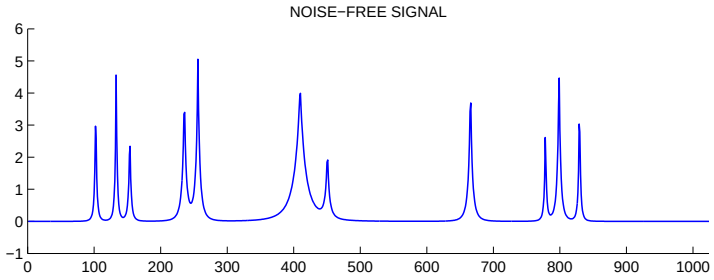
$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}.$$

The INV block that reverses this decomposition uses the same multipliers and is given by

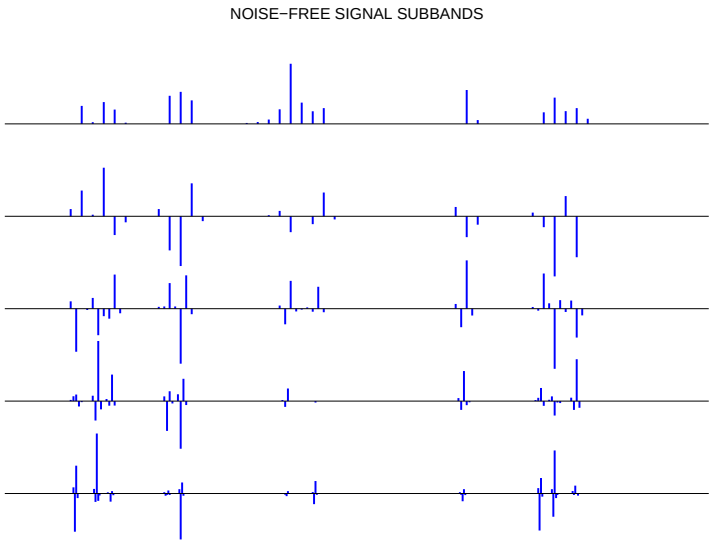
$$y(2n) = h_0\, c(n) + h_2\, c(n-1) + h_3\, d(n) + h_1\, d(n-1)$$
$$y(2n+1) = h_1\, c(n) + h_3\, c(n-1) - h_2\, d(n) - h_0\, d(n-1).$$

The new decomposition has the property that when the original signal $x(n)$ is a linear signal, $x(n) = a\,n + b$, then the output signal $d(n)$ will be identically zero. Moreover, for the linear signal, the output signal $c(n)$ will also be linear, and therefore, by induction, the wavelet representation will be *entirely* zero, for the exception of the $c(n)$ output signal of the last stage. (In contrast, for a linear signal, *no* values of the Haar representation will be zero.) Therefore, the wavelet transform using the Daubechies decomposition provides a sparse representation for piecewise-*linear* signals. In fact, Daubechies gave a method to construct new decompositions with this property for polynomial of *any* order. (The constant signal, for which the Haar transform is well suited, is a polynomial of degree 0, etc.) There are now, in addition to Daubechies' decomposition, a great number of other decompositions from which wavelet transforms can be built.

We now illustrate why a sparse representation is effective for noise reduction. Consider the 1024-point signal:

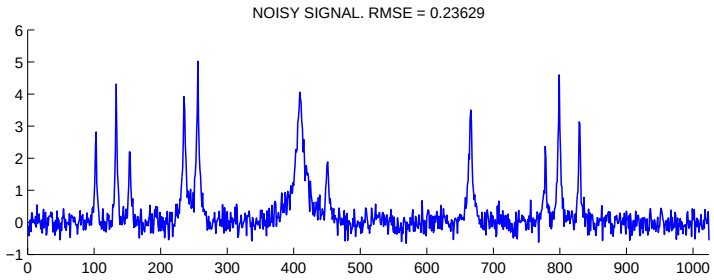


and its four-level wavelet representation:

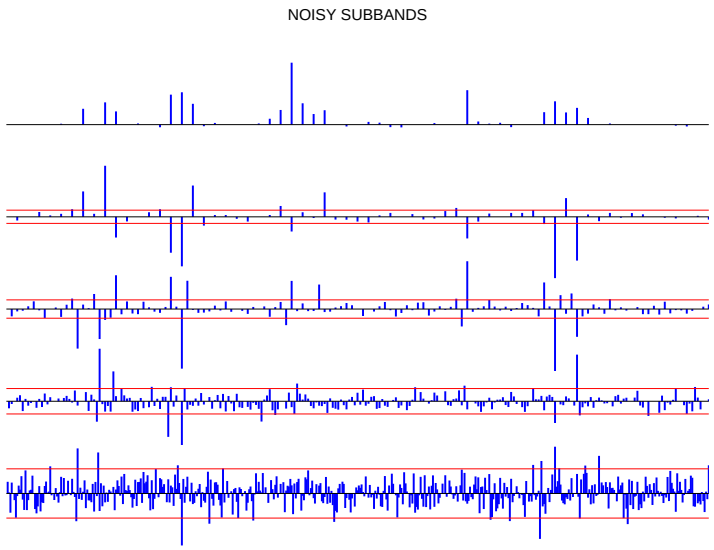


The wavelet representation illustrated here consists of five vectors of lengths 64, 64, 128, 256, and 512; and was computed with a Daubechies-like wavelet transform. Many of the values are small in absolute value; the wavelet representation is sparse, as desired. Note that each of the signals

illustrated here is individually scaled to maximize the visibility of the values. In case only a noisy version of the signal is available, we might wish to retrieve the original signal as far as is possible. For example, suppose we have only the following noisy signal, which has been obtained by adding a zero-mean Gaussian random variable, of standard deviation σ_n, to the original signal:

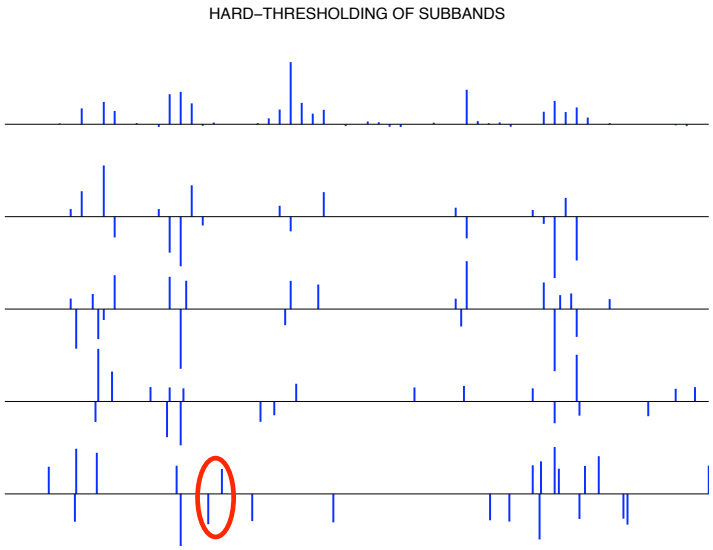


Then the wavelet representation of the noisy signal is itself noisy:

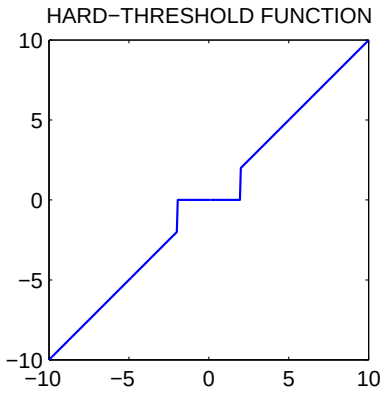


The addition of zero-mean Gaussian noise to the original signal has the effect of adding zero-mean Gaussian noise to the wavelet representation as well. Furthermore, because of the way we have implemented the wavelet transform here, the standard deviation of the noise in the wavelet representation is the same as the standard deviation of the noise that was added to the signal. The red lines displayed in this noisy wavelet representation indicates $\pm 2\,\sigma_n$. Note that relatively few values in the wavelet representation extend outside the red lines. To accomplish wavelet-based noise reduction, we would like to somehow modify the noisy wavelet representation so that it resembles, as closely as possible, the wavelet representation of the noise-free signal illustrated above. After such a modification, we could reverse the wavelet transform to obtain a less noisy signal. The most basic wavelet-based method for reducing signal noise is to simply set the small values in the wavelet representation to zero. For example, we might set those values within the red lines to zero, in which

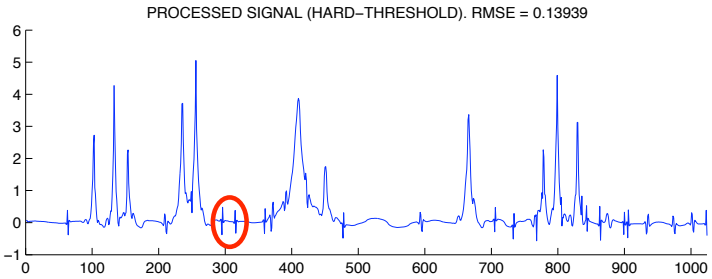
case, we obtain the following, modified wavelet representation:



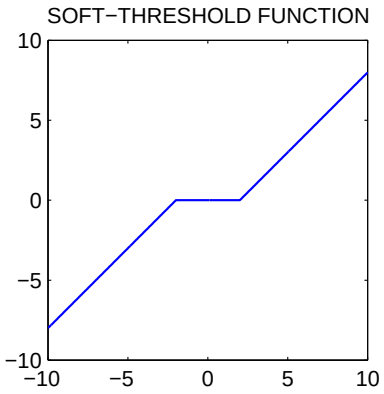
The new wavelet representation is relatively sparse. Comparing this wavelet representation to that of the original noise-free signal, we notice some values slightly exceeded the red line threshold (two are indicated) where in the noise-free wavelet representation the true value is almost zero. By the way, setting the small values to zero, can be represented by a nonlinear function:



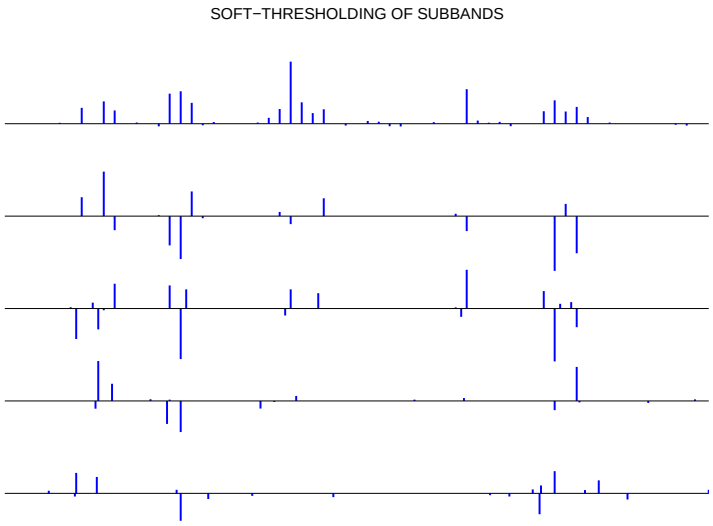
This function is commonly called the 'hard-thresholding' rule. We reverse the wavelet transform to obtained the noise-reduced signal:



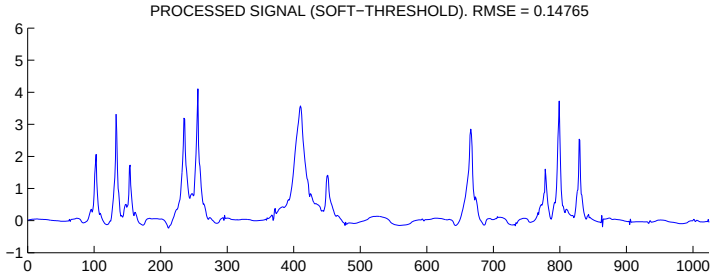
Comparing with the original noise-free signal, we see that much of the noise has been removed. However, there are some noise spikes remaining in the processed signal. These are due exactly to values in the noisy wavelet representation that slightly exceeded the red line threshold. To reduce these unwanted noise spikes, one option is to use a larger threshold so that more values are set to zero. However, increasing the threshold too high will cause other distortion artifacts in the processed signal. Another option is to use a different nonlinear threshold rule, for example the 'soft-threshold' rule:



which, in addition to setting small values to zero, also shrinks the remaining values towards zero, so that the nonlinearity is now a continuous function. Using this rule, any values in the noisy wavelet representation that slightly exceed the threshold, will be reduced in absolute value. If we process the noisy wavelet representation with the soft-threshold rule instead of the hard-threshold rule, we obtain the following modified wavelet representation:



Now the values that exceeded the red line threshold in the noisy wavelet representation are reduced. Reversing the wavelet transform produces the following noise-reduced signal:

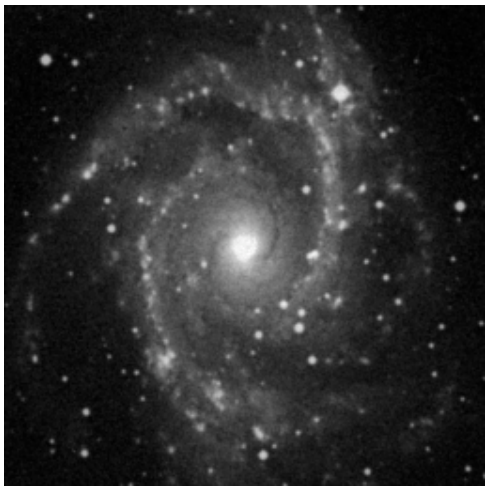


which suffers less from spurious noise spikes. On the other hand, soft-thresholding can sometimes lead to the loss of more detail than the hard-threshold rule. There are now many different nonlinearities that have been proposed for wavelet-based noise reduction. Some nonlinearities are derived from statistical models of the distribution of wavelet representation values; some are based on other mathematical models. The best performing nonlinearities are multivariate.

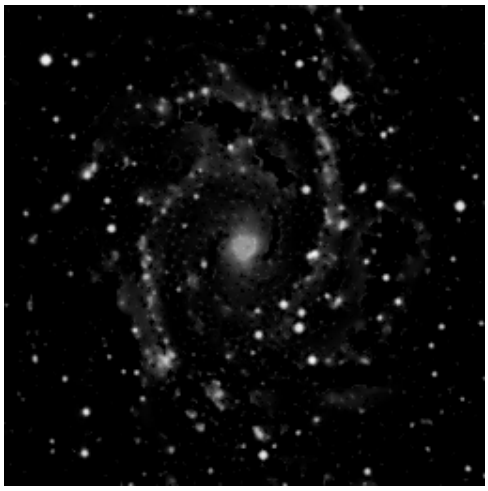
Wavelet-based nonlinear thresholding is effective for noise reduction only to the extent to which the wavelet representation of the noise-free signal is sparse. In general, the more zeros there are in the wavelet representation of a signal, the better compression, noise reduction, blur reduction, etc. can be accomplished. Certainly, for some signals, the wavelet transform does not provide a sparse representation at all. The most appropriate transform to use depends on the type of signal being processed. For some signals the Fourier transform will be superior. One avenue of research has been the development of algorithms that can automatically find a good transform for a specific signal; these are 'data-adaptive transforms'.

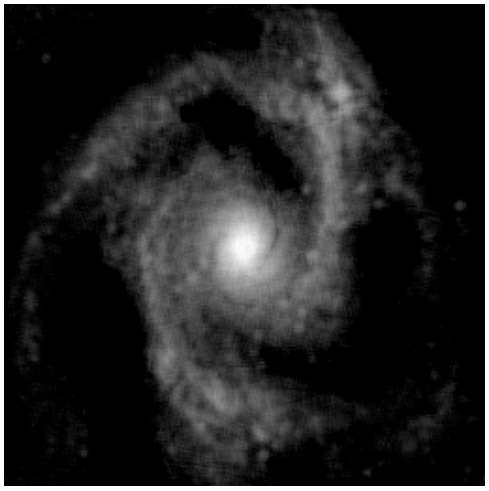
The development of additional related transforms and of specialized algorithms for processing, analyzing, and storing signals and images using such transforms has been an active topic in applied mathematics, statistics, science, and engineering. For example, the design of other signal decompositions using the theory of digital filter banks has been an active area in digital signal processing

research. Also, decompositions that expand the amount of data provide further improvements for noise reduction and other applications; such transforms are called *frames* in mathematics. One example of an effective 'frame' is the dual-tree complex wavelet transform which overcomes some of the limitations of the conventional wavelet transform. Additionally, it turns out that for images and higher-dimensional images, a different kind of decomposition can perform better than conventional wavelets. That is because the two-dimensional form of the conventional wavelet transform provides an optimally sparse representation for images with only point-like discontinuities, yet many images have discontinuities along curves (these are the edges of objects in the image). On the other hand, the curvelet transform, introduced by Candes and Donoho, is designed specifically to overcome this problem; it does provide sparse representations of two-dimensional curve-discontinuities. The curvelet transform has inspired contourlets, shearlets, and other transforms. For some kinds of image it is not so easy to obtain a sparse representation with any one transform. In this case, it can be useful to use several transforms together. For example, Jean-Luc Starck et al. used the isotropic wavelet, ridgelet, and curvelet transforms together to automatically separate the astronomical image of galaxy NGC 2997:



into two images, one with containing the stars, the other containing the vapor:





For further reading, a few of the many books on wavelets transforms and their applications are listed in the references.

References

[1] C. S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms.* Prentice Hall, 1997.

[2] I. Daubechies. *Ten Lectures On Wavelets.* SIAM, 1992.

[3] S. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, 1998.

[4] I. W. Selesnick, R. G. Baraniuk, and N. G. Kingsbury. The dual-tree complex wavelet transform - A coherent framework for multiscale signal and image processing. *IEEE Signal Processing Magazine*, 22(6):123–151, November 2005.

[5] J.-L. Starck, M. Elad, and D. L. Donoho. Redundant multiscale transforms and their application for morphological component analysis. *Advances in Imaging and Electron Physics*, 132, 2004.