Parallel Transform Method
for Lossless Compression of Analog Data

R. Crandall, Scientific Computation Group, and
M. Minnich, Sound and Music Group, NeXT Computer, Inc.
c. 1991 NeXT Computer Inc. All Rights Reserved.

Abstract: The Parallel Transform Method (PTM) is a compression scheme for analog data, especially sound signals. Upon sub-blocks of the input data stream, one allows a parallel battery of linear and non-linear transforms, together with a battery of compressors, to "compete" for superiority (compressibility). The compressed output stream consists of blocks of compressed data, together with tokens describing the winning transform/compressor combinations. The method is expandable - one may introduce additional linear and non-linear transforms in future implementations, with negligible effect on decompression times. PTM has been implemented for sound compression in NeXT Software Release 2.0.

1. General PTM scheme

There are many schemes for lossy compression of analog data, examples of such data being sounds, images, and scientific instrumentation data. In one wide class of lossy schemes, one may derive a transform (or "spectrum") of the data and truncate this spectrum, for example by removing high-frequency components or quantizing components according to some model of human perception.

But lossless compression is useful in its own right, because one might need to be assured of the absolute integrity of the compression-decompression chain. A good example would be that of compressing neurological data, for which a minute but critical pathology might be removed by a lossy scheme.

Lossless compression can also be important for high quality sound data. The NeXT Software Release 2.0 contains an implementation of PTM for 16-bit audio samples. Even a simple implementation of PTM can give about 2:1 lossless compression, with this ratio increasing as parallel transforms and compressor strategies are added in modular fashion.

Figure 1 shows the general PTM scheme. A block x of analog data (assumed to be a stream of integers) is fed to P different transforms, with each transform being tested over K different compressors. The output stream is
Parallel Transform Method

Figure 1: General Parallel Transform Method (PTM) scheme. The input block is tested over \( P \times K \) combinations of transforms and (lossless) compressors. The "winning pair" \((f, k)\) is fed along with the compressed stream \( C_k(x) \), as the output block.

symbolized by:

\[ f, k, C_k(x) \]  \hspace{1cm} (1.1)

meaning that transform \( T_f \) in conjunction with compressor \( C_k \) is the "winning combination," in the sense of minimizing the bit length of the output stream.

A primary advantage of PTM is that at any future time additional transforms and/or compressors can be added to the respective batteries. While such additions increase the compression time, there is only negligible effect on the decompression stage because decompression can be handled rapidly via a switch mechanism that obtains the \( f, k \) parameters and concentrates on just one decompressor.

2. Linear and non-linear transforms

Assume that signals are always decimated into blocks with a fixed number \( N \) of consecutive data in each block. The block data are denoted by \( x \):

\[ x = \{ x_j : j = 0, 1, ..., N-1 \} \]  \hspace{1cm} (2.1)

We assume further that each datum is an integer taken from some set \( S \), for example 16-bit signed audio data would have \( S = (-32768, ..., +32767) \).

ICMC 526
A transform we define for the present purposes as any bijective mapping

\[ T: \mathbb{S}^N \rightarrow \mathbb{S}^N \]  
(2.2)

That \( T \) must be invertible means that any transformed signal \( y = Tx \) can be used to recover \( x \) unambiguously:

\[ x = T^{-1}y \]  
(2.3)

It is convenient to think of \( T \) as an \( N \)-by-\( N \) matrix acting on an \( N \)-dimensional column vector \( x \). However, this can be misleading because the PTM allows non-linear transformations that do not admit of matrix representation.

Both linear and non-linear transforms can be considered in PTM. Examples of linear transforms include invertible linear filters, classical transforms such as the FFT, and modern wave-packet and wavelet transforms (many such filters can be decomposed into chains of very simple quadrature mirror filters [Strang 1989]).

An example of a non-linear transform is the logic filter. For example, an "exclusive-or" filter can be defined by:

\[ \begin{align*}
  y_0 &= x_0 \\
  y_1 &= x_0 \oplus x_1 \\
  y_2 &= x_0 \oplus x_2 \\
  &\vdots \\
  y_{N-1} &= x_{N-2} \oplus x_{N-1}
\end{align*} \]  
(2.4)

which is clearly invertible. The idea behind such a non-linear transform is that if the data in \( x \) are very similar, one might obtain most of the \( y \) data as very small values (having few bits). Other non-linear transforms include index permutations, value permutations, and number-theoretic filters.

In the actual NeXT Software Release 2.0, the exclusive-or filter is used together with difference filters of various orders. There is a singleton compressor, which uses a simple bit-packing scheme. The "winning" combination is therefore just the winning transform, which in this case wins by exhibiting the smallest supremum of bit-lengths computed across the \( y_{j, j} = 0, 1, \ldots, N-1 \).

3. Difference transforms

Difference transforms (invertible difference filters) are certainly attractive for sound compression. The difference technique was anticipated and analyzed in detail by [Moorer
1979]. A simple example is a second-difference filter, call it $D_2$, with matrix representation:

$$
\begin{align*}
1 & 0 & 0 & 0 & 0 & 0 & \ldots \\
0 & 1 & 0 & 0 & 0 & 0 \\
D_2 & = & 1 & -2 & 1 & 0 & 0 \\
0 & 1 & -2 & 1 & 0 & 0 \\
0 & 0 & 1 & -2 & 1 & 0 & \ldots 
\end{align*}
$$

(3.1)

with the bottom row ending in: $1 \ -2 \ 1$. By its tringular construction, the determinant of the transform is 1, and so $D_2$ is patently invertible. The transformed signal $y$ is computed:

$$
\begin{align*}
y_0 &= x_0 \\
y_1 &= x_1 \\
y_2 &= x_0 - 2x_1 + x_2 \\
\vdots \\
y_{N-1} &= x_{N-3} - 2x_{N-2} + x_{N-1}
\end{align*}
$$

(3.2)

and it is an easy matter to recover the $x_j$ from the $y_j$. Now define a general $d$-th order difference transform by:

$$
\begin{align*}
y_j &= x_j & : j = 0, 1, \ldots, d-1 \\
y_j &= \sum_{k=0}^{d} (-1)^k \binom{d}{k} \frac{d!}{k!(d-k)!} x_{j-k} & : j = d, \ldots, N-1
\end{align*}
$$

(3.3)

Again, this transform is invertible, and yields a discrete version of the $d$-th derivative of the input signal.

4. References

Moorer J A, AES preprint, Brussels, Belgium 82nd convention of the AES, 1979