Efficient implementation of discrete Pascal transform using difference operators

Q.-C. Zhong, A.K. Nandi and M.F. Aburdene

An alternative way is provided to define the discrete Pascal transform using difference operators to reveal the fundamental concept of the transform, in both one- and two-dimensional cases, which is extended to cover non-square two-dimensional applications. Efficient modularised implementations are proposed.

Introduction: Transformations are vital to any engineering subject, in particular, to electrical engineering. Fourier series, Laplace transform, Fourier transform and their counterparts in the discrete-time domain, are the fundamental tools for signal processing, communication and control engineering. Recently, the discrete Pascal transform (DPT) has been introduced to process discrete signals [1]. The transform has found applications in edge detection [2], bump detection and watermarking [1]. Various features of the DPT, such as computational complexity [3, 4], hardware implementation [5], interpolation using DPT [6] and the link between Poisson sequences and Laguerre polynomials [7], have been reported. The major contributions of this Letter are: (i) to provide an alternative definition of the transform using the difference operator, (ii) to provide efficient modularised implementations, and (iii) to extend the transform to non-square 2D applications.

1D DPT defined with difference operators: The DPT of a discrete-time signal \( y[n] \) with \( n = 0, 1, 2, \ldots, (N - 1) \) is given by (the use of notation \( y^T \) (\( Y \)) instead of \( y \) (\( Y \)) is to facilitate derivation and also to be consistent with the treatment in [1]):

\[
y^T = A y^T
\]

\[
Y^T = [y[0] \ y[1] \cdots y[N - 1]^T
\]

where

\[
y^T = [y[0] \ y[1] \cdots y[N - 1]^T
\]

and \( A = \begin{bmatrix} 1 & 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 1 & 0 & 0 \end{bmatrix} \) is an \( N \times N \) dimensional lower-triangular matrix constructed by placing the rows of Pascal’s triangle into the matrix form and alternating the signs of the columns. The elements of matrix \( A \) (note that the row and column indices here start from 0, not 1, i.e. the first element is \( a_{00} \) instead of \( a_{11} \)) are given by

\[
a_{jl} = \begin{cases} (-1)^{j-l} & \text{if } 0 \leq j \leq l \leq N, \\ 0 & \text{otherwise} \end{cases}
\]

The transformed signal values are:

\[
Y[0] = y[0] \\
\vdots
\]

where \( q \) is the shift operator satisfying \( q^{-1}y[n] = y[n - k] \). Hence, the discrete Pascal transform satisfies

\[
Y[k] = (-\Delta^ky[k], 0 \leq k < N
\]

From (2), the DPT can be implemented as shown in Fig. 1 as a series of \( \Delta \)-blocks with a single-pole multi-thrown switch to select the output of block \( k \) at step \( k \). Each \( \Delta \)-block consists of a shift register and an adder (subtracter) as \(-\Delta = q^{-1} - 1\). This implementation is highly modularised and very efficient. It requires \((N - 1)\) adders (subtracters) and \((N - 1)\) shifts. Since the hardware implementation cost depends on the number of adders (subtracters), this hardware implementation is much more efficient than the one proposed in [3], which, having \( 1/2N(N - 1) \) adders, is already very efficient in comparison to the original computational burden involving matrix multiplications. It is worth noting that these two schemes require the same number, \( 1/2N(N - 1) \), of additions in software implementation so the contribution here is in the savings of hardware implementation. In addition, more \( \Delta \)-blocks can be easily appended to the end of the chain if the number \( N \) increases. This offers the opportunities for real-time applications.

Two-dimensional case: Using (2), the transform can be rewritten in matrix form as

\[
Y = yQ
\]

where \( y \) and \( Y \) are the original and transformed signal row vectors, respectively, and

\[
Q = \text{diag}(1, -\Delta_p, (-\Delta_p)^2, \ldots)
\]

is a diagonal matrix involving the difference operator \( \Delta_p = 1 - q^{-1} \). As pointed out in [1], for a two-dimensional signal matrix \( z \), the DPT can be carried out to perform a one-dimensional transform to the rows of \( z \) and then another one-dimensional transform to the columns of the resulting matrix. In other words, the transformed signal matrix \( Z \) satisfies

\[
Z^T = (zQ)^T P \quad \text{or} \quad Z = PzQ
\]

where

\[
P = \text{diag}(1, -\Delta_p, (-\Delta_p)^2, \ldots)
\]

with the difference operator \( \Delta_p = 1 - q^{-1} \), defined along the column direction (while \( \Delta_p = 1 - q^{-1} \) is defined along the row direction). For example, for a \( 3 \times 3 \) signal matrix \( z = \begin{bmatrix} 2 & 0 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \), the transformed signal is
In general, the \((j, k)\)th element of \(Z\) is given by
\[
Z_{jk} = (-\Delta_p)^j(-\Delta_q)^k z_{jk}
\]
For example,
\[
Z_{11} = \Delta_p \Delta_q z_{11} = z_{11} - z_{10} - z_{01} + z_{00}
\]

\[\text{Fig. 2 Structure of two-dimensional DPT using difference operators}\]

It is worth noting that the dimensions of \(Q\) and \(P\) may not be the same, i.e. the two-dimensional signal matrix \(z\) may be non-square. This is different from [1], where the image has to be square. Assume that the dimension of \(z\) is \(M \times N\), then the two-dimensional DPT can be implemented as shown in Fig. 2. The image \(z\) is divided into \(M\) rows \(z(j, :)\), \(j = 0, 1, \ldots, (M - 1)\), to enter \(M\) chains of \((N - 1)\) \(\Delta\)-blocks from the left in series. At each step, \(M\) transformed values are generated. These values are stored and passed through a parallel-to-series (P/S) device to generate a row vector having \(M\) elements. There are in total \(N\) such row vectors. Each of these vectors then enters another chain of \((M - 1)\) \(\Delta\)-blocks from the left in series and the outputs are timed to form the final transformed signal \(z(m,:)\) for \(m = 0, 1, \ldots, (N - 1)\). Assume that the P/S block (including storing data, which normally takes one clock cycle, and P/S conversion) takes one step to send out a value, the \(th\) transformed column vector is available at step \(M\) and the whole process takes \((M + N - 1)\) steps. This implementation requires \(M(N - 1) + N(M - 1) = (2MN - M - N)\) blocks, with each block consisting of a subtractor and one shift. Again, this is very efficient and highly modularised. Interestingly, when \(N = 1\), it reduces to another implementation, which adds a P/S block to the implementation shown in Fig. 1 and parallel data inputs can be processed.

As to the two-dimensional inverse DPT, it has the same form as the DPT [1] and can be defined as:
\[
z = PZQ
\]
Note that, again, the image does not have to be square.

**Conclusion:** An alternative way to define the DPT using difference operators has been used to develop efficient modularised implementations for both 1D and 2D transforms. In addition, the approach has been extended to find non-square 2D DPT transforms.

© The Institution of Engineering and Technology 2007
28 September 2007
Electronics Letters online no: 20072712
doi: 10.1049/el:20072712
Q.-C. Zhong and A.K. Nandi (Department of Electrical Engineering and Electronics, The University of Liverpool, Brownlow Hill, Liverpool L69 3GJ, United Kingdom)
E-mail: q.zhong@liv.ac.uk
M.F. Aburdene (Electrical Engineering Department, Bucknell University, Lewisburg, PA 17837, USA)
M.F. Aburdene: Visiting Professor at the Department of Electrical Engineering and Electronics, The University of Liverpool.

**References**