fN(n) = \left( \frac{E[z^2(n)]}{E[z^2(n)]} \right)^2 + \left( \frac{C_2(z(n))}{C_2(z(n))} \right)^2 - 3 \left( \frac{C_1(z(n))}{C_2(z(n))} \right)^2 \tag{10}

We consider the lattice-form IIR filter proposed in [2] which comprises \( N + 1 \) sections, each of which is characterised by two identical reflection coefficients \( G_p, 0 \leq p \leq N \). The overall output \( z(n) \) is weighted sum of the backward residuals as follows:

\[
z(n) = \sum_{i=0}^{N} q_i(n)K_i(n) \tag{11}
\]

where \( \{q_i(n)\} \) and \( \{K_i(n)\} \) are the backward residuals and the feedforward coefficients, respectively. The IIR lattice filter can also be described by the following section input/output equations:

\[
q_m(n) = q_{m-1}(n-1) + G_m(n)f_{m-1}(n) \tag{12}
\]

\[
f_{m-1}(n) = f_m(n) - G_m(n)q_{m-1}(n-1) \tag{13}
\]

These equations require that \( f_{m}(n) = g_{m}(n) \) and the input is given by \( f_0(n) = y(n) \), where \( y(n) \) is the filter input. Define

\[
\Theta(n) = [G(n) K(n)]^T
\]

\[
\Theta(n) = [G_1(n) \cdots G_N(n) K_0(n) \cdots K_N(n)]^T \tag{14}
\]

\[
Q(n) = [q_0(n) \cdots q_N(n)]^T \tag{15}
\]

\[
\Psi(n) = [\gamma(n) K(n) Q(n)]^T \tag{16}
\]

where \( y(n) \) is a matrix composed of the derivatives of \( Q(n) \) with respect to \( G(n) \). Applying the cost function \( J_c \), we then obtain the lattice-form IIR blind algorithm described as

\[
\Theta(n) = \Theta(n-1) - \mu \left( 4\gamma_1 + 36\gamma_0 E[z^2(n)] 
- 24\gamma_0 E[z^4(n)] E[z^2(n)] - 3E[z^2(n)] z(n) 
+ 8\gamma_2 E[z^4(n)] - 3E[z^2(n)] - 12\gamma_2 \right) \Psi(n) \tag{17}
\]

where

\[
\gamma_{ij}(n) = \gamma_{i,j-1}(n-1) + G_j(n)\theta_{i,j-1}(n) + \delta_{i,j}f_{j-1}(n) \tag{18}
\]

\( \gamma(n) \) is the \( i \)th component of \( \gamma(n) \), \( \delta_{ij} \) is the Kronecker delta function and

\[
\theta_{i,j-1}(n) = \theta_{i,j}(n) - G_j(n)\gamma_{i,j-1}(n-1) - \delta_{i,j}\theta_{i,j-1}(n-1) \tag{19}
\]

These expressions require that \( \theta_0(n) = \gamma_0(n) \) and \( \theta_0(n) = 0 \).

Simulation results: In this Section we present some Monte-Carlo simulations of the proposed blind algorithms. Binary PSK data are transmitted. The channel defined below is used in the simulations:

channel : \( y(n) = d(n) + 0.9d(n-1) \)

The step size \( \mu \) is chosen to be \( 5 \times 10^{-4} \), the length \( N \) of the IIR equaliser is 6. The number of taps \( M \) for the FIR equaliser under comparison is 20. Fig. 1 shows the learning curves for the lattice-form IIR equaliser and its FIR version equaliser, respectively, for binary PSK data in the channel. These curves indicated that the proposed IIR blind algorithm not only has a faster convergence speed, but also yields a smaller steady state MSE than its FIR counterpart.

EBTC: An economical method for searching the threshold of BTC compression

Ching-Yung Yang and Ja-Chen Lin

Indexing terms: Data compression, Image processing, Image coding

An economical method for obtaining a nearly-optimal threshold for the block truncation coding (BTC) image compression algorithm is presented. Simulation results show that the PSNR performance of the proposed economical BTC (EBTC) method is very close to that of the optimised BTC (OBTC) algorithm, but EBTC only takes about half the computation time required by the OBTC algorithm.

Introduction: Block truncation coding (BTC) [1, 2] is a simple and fast compression technique for digitised images. To reduce the mean square error (MSE) further, Kamel et al. [3] presented an algorithm which used a (partially) optimal threshold to quantise the block. However, as was pointed out in [4], the MSE generated by the method of Kamel et al. did not obtain a minimum value because the quantised error was neglected. Chen and Liu [4] suggested the optimised BTC (OBTC) algorithm to minimise the MSE, using a new threshold value searching policy. The PSNR performance of the OBTC algorithm is slightly better than that of [3], and the computation speed is also faster than that of [3], although the computation cost of [4] is still a heavy burden. In this Letter we develop an economical BTC (EBTC) algorithm whose PSNR is high (MSE is minimised) but the computation time is reduced significantly. Experiments showed that there is an effective tradeoff between the PSNR and computation time. We note that the nearly-optimal threshold is obtained by only searching a small portion of the input data for each block.

EBTC algorithm: Partition the image into blocks of size \( n \times n \). For each block, let \( G = \{g|1, 2, \ldots, G\} \) be the \( G = n^2 \) given grey values to be split into two classes \( H = \{g \geq Q\} \) and \( L = \{g < Q\} \) where \( Q \) is a threshold value to be determined. The MSE of the block is defined by

\[
MSE = \sum_{g \in H} (g - \bar{h})^2 + \sum_{g \in L} (g - \bar{l})^2 \tag{1}
\]

where \( \bar{h} \) and \( I \) are the average grey values of \( H \) and \( L \), respectively. We try to obtain a \( Q \) for which the MSE is small. The procedure is as follows: The centroid \( O \) of \( G \) is evaluated by \( O = \frac{\sum_{g \in G} g}{|G|} \). The radius weighted mean [5] \( R \) of \( G \) is then evaluated by \( R = \frac{\sum_{g \in G} r(g)O}{\sum_{g \in G} r(g)} \) with \( r = |g - \bar{O}| \) for all \( g \). If \( O \) equals \( R \), then the value of \( R \) is assigned directly to the final threshold \( Q \) (that is, the threshold is obtained quickly without any searching operation). Otherwise, the data set is divided into two (temporary) subsets, say \( G_1 \) and \( G_2 \), by using the (temporary) threshold \( R \). Let \( G_1 \) \( 1870 \) ELECTRONICS LETTERS 26th September 1996 Vol. 32 No. 20

References


© IEE 1996
Fang-Biau Ueng (Department of Communication Engineering, National Chiao Tung University, Hsinchu, Taiwan, Republic of China)
but the processing speed of the proposed method is nearly twice
algorithm by a factor of 0.40-0.81dB. Conversely, the PSNR of
the proposed method is very close to that of the OBTC algorithm,
the black level in the grey levels in the block. The
and execution time produced by these three algorithms
with block sizes 4 x 4 and 8 x 8 are provided in Tables 1 and 2,
respectively. The MSEs (taking the average among all the blocks)
for these algorithms are also given in Table 3. (Although we did
not list the corresponding results of [3], we did observe that the
PSNR of [3] was a little less than, but very close to that of [4].
Also, the algorithm in [3] was observed to be slower than that in
4.)

Table 1: PSNR and execution time for different algorithms when
the block size is 4 x 4

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>33.85</td>
<td>s</td>
<td>34.47</td>
</tr>
<tr>
<td>Jet</td>
<td>32.65</td>
<td>0.48</td>
<td>33.44</td>
</tr>
<tr>
<td>Pepper</td>
<td>34.06</td>
<td>0.48</td>
<td>34.74</td>
</tr>
<tr>
<td>Scene</td>
<td>30.53</td>
<td>0.48</td>
<td>31.17</td>
</tr>
<tr>
<td>Monkey</td>
<td>27.66</td>
<td>0.48</td>
<td>28.10</td>
</tr>
</tbody>
</table>

Table 2: PSNR and execution time for different algorithms when
the block size is 8 x 8

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>30.60</td>
<td>s</td>
<td>31.29</td>
</tr>
<tr>
<td>Jet</td>
<td>29.55</td>
<td>0.41</td>
<td>30.43</td>
</tr>
<tr>
<td>Pepper</td>
<td>30.36</td>
<td>0.41</td>
<td>31.28</td>
</tr>
<tr>
<td>Scene</td>
<td>27.49</td>
<td>0.41</td>
<td>28.24</td>
</tr>
<tr>
<td>Monkey</td>
<td>25.87</td>
<td>0.41</td>
<td>26.27</td>
</tr>
</tbody>
</table>

Table 3: MSE for different algorithms with block sizes 4 x 4 and
8 x 8

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>26.80</td>
<td>36.66</td>
<td>23.23</td>
</tr>
<tr>
<td>Jet</td>
<td>35.29</td>
<td>72.17</td>
<td>29.48</td>
</tr>
<tr>
<td>Pepper</td>
<td>25.55</td>
<td>59.80</td>
<td>21.85</td>
</tr>
<tr>
<td>Scene</td>
<td>57.52</td>
<td>116.04</td>
<td>49.62</td>
</tr>
<tr>
<td>Monkey</td>
<td>111.43</td>
<td>168.16</td>
<td>100.63</td>
</tr>
</tbody>
</table>

From these Tables, it is observed that the PSNR performance
that the MSE produced by the proposed EBTC method is close to
that produced by the OBTC algorithm Moreover, we also found
that for ~73% of the total number of encoded blocks, the MSEs
generated by our method within these blocks are optimal instead
of nearly-optimal. We therefore suggest that the PSNR provided
by the proposed EBTC method is higher than that provided by the
OBTC algorithms, and is very close to that of the OBTC algorithm.
In addition, the proposed method takes only ~50% of the computation time required by the OBTC algorithm

Acknowledgments: This project was supported by the National Science
Council, Republic of China, under grant NSC85-2213-E009-111.

© IEE 1996

Fast algorithm for optimal bit allocation in a
rate-distortion sense

Woo Yong Lee and Jong Beom Ra

To improve the quality of reconstructed images for a given bit
rate constraint, the assigned bits must be distributed efficiently
using a set of admissible quantisers so that source distortion can
be minimised. However, the optimal bit allocation scheme is
usually not practical due to its large computational burden. The
authors propose a new fast algorithm which needs less computing
time than required by existing fast algorithms.

Introduction: Optimal bit allocation for source coding involves
minimising source distortion, subject to a given bit budget. In
image coding, the intention is to distribute the assigned bits effici-
cently among image blocks by using proper quantisers. To meet
this requirement, a dynamic programming method is suggested,
based on rate-distortion characteristics [1]. This optimal method,
however, requires large computational complexity. Even though
several fast algorithms [2-4] have been proposed to reduce the
computational complexity, they are still not fast enough for practi-
cal applications related to video coding.

In this Letter, we propose a new fast algorithm for optimal bit
allocation, which is based on the bi-directional prediction of the
Lagrange multiplier \( \lambda \) for a given bit rate \( R \). The proposed algo-
Algorithm reduces the computational complexity substantially,
compared with previous fast algorithms. It could be useful for
asymmetric coding applications such as CD-ROM, video on
demand (VOD) etc., where high coding performance is essential
and real-time encoding is not imperative.