INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO-IEC/JTC1/SC29/WG11
CODING OF MOVING PICTURES AND ASSOCIATED AUDIO INFORMATION

ISO-IEC/JTC1/SC29/WG11
MPEG92/
September 1992

CCITT
STUDY GROUP XV WP XV/1
Experts Group for ATM Video Coding
Tarrytown September 1992

Document # AVC-349

English Version

SOURCE: A. Reibman, K. Matthews, J. Mailhot, R. Aravind: AT&T Bell Laboratories
TITLE: Leaky prediction: Eliminating the limit cycle
PURPOSE: Informational, Proposal

1 Introduction

A difficulty arises when implementing leaky prediction that is caused by the finite precision of the digital filter used to generate the prediction error. This is independent of the multiplication method in the leak-multiplier. We describe the problem, propose two solutions that completely eliminate this difficulty, and recommend that the latter solution be incorporated into the MPEG-2 syntax.

Throughout the description, we ignore signal quantization effects. While the problem of finite precision remains for more general signals, we assume the original signal is constant in space and time for illustration purposes. Furthermore, we assume that there has been a channel change which resets the frame memory at the decoder to zero and that the frame memory at the encoder has converged to its correct value prior to the channel change.

Figure 1 shows a simplified block diagram of encoder and decoder prediction loops, assuming the value 128 has already been subtracted from the input signal x_t. The encoder generates the prediction error to be sent to the decoder by

$$y_t = x_t - T_n[\alpha x_{t-1}],$$

where $\alpha = 1 - 1/2^n$ is the leak factor, and $T_n[\cdot]$ denotes the truncation operation that eliminates the least significant n bits. Truncation (or rounding) is necessary because input pixels to the DCT can contain no more than 9 bits of precision per pixel. However, the output of the multiplication αx_{t-1} is $9+n$ bits, and the extra n bits cannot be transmitted to the decoder. The subscript t can be either a spatial or a temporal index. The decoder reproduces the signal

$$w_t = y_t + T_n[\alpha w_{t-1}],$$

to which it adds 128 to get the output signal.

If the input x_t is constant in the index t, then the prediction error y_t will also be constant. Therefore, the decoded signal w_t will build up until it reaches a steady-state value that depends only on y_t. As a result, a range of input values are all mapped into the same steady-state output value at the decoder. This is illustrated in Table 1 when $\alpha = 15/16$, both for floating-point multiplication with truncation as described in appendix L (core experiment 6) and fixed-point multiplication implemented with shift-and-subtract. (The limit cycle also exists if rounding is used instead of truncation.) The problem becomes more significant as α approaches 1, since 2^n levels are mapped into a single value.

Floating point			shift & subtract		
Input range $x_t + 128$	y_t	$w_t + 128$	Input range $x_t + 128$	y_t	$w_t + 128$
0 – 15	-8	0	0 – 15	-8	0
16 – 31	-7	16	16 – 31	-7	16
32 – 47	-6	32	32 – 47	-6	32
48 – 63	-5	48	48 – 63	-5	48
64 – 79	-4	64	64 – 79	-4	64
80 – 95	-3	80	80 – 95	-3	80
96 – 111	-2	96	96 – 111	-2	96
112 – 127	-1	112	112 – 127	-1	112
128	0	128	128 – 143	0	128
129 – 144	1	129	144 – 159	1	144
145 – 160	2	145	160 – 175	2	160
161 – 176	3	161	176 – 191	3	176
177 – 192	4	177	192 – 207	4	192
193 – 208	5	193	208 – 223	5	208
209 – 224	6	209	224 – 239	6	224
225 – 240	7	225	240 – 255	7	240
241 – 255	8	241			

Table 1: Limit cycles with $\alpha = 15/16$

The limit-cycle effect is most easily seen after a channel change in a still image with a leak factor fairly close to 1. For example, in still-flowergarden with $\alpha = 15/16$, there is significant contouring present in the sky, which remains even after 60 frames. This is demonstrated by D1 tape.

2 Two solutions to the limit cycle

We describe two solutions to the limit cycle, and compare their convergence given channel changing and their noise performance. Both methods are equivalent to adding a signal that varies between 0 and $(2^n - 1)/2^n$ after multiplication but before truncation. However, the methods differ in how they produce this auxiliary signal.

2.1 Solution 1

The first solution uses error spectrum shaping (ESS) to eliminate the limit cycle in the digital filter that generates the prediction error signal. (Note that this is similar to, but distinct from, error spectrum shaping used in quantization (DPCM) systems.) A simplified block diagram is shown in Figure 2.

ESS generates the auxiliary signal using the n least significant bits from the previous truncated signal. Therefore, at the encoder, the auxiliary signal is

$$r_t = R_n[\alpha x_{t-1} + r_{t-1}],$$

and the prediction error signal is

$$y_t = x_t - T_n[\alpha x_{t-1} + r_{t-1}],$$

where the operation $R_n[\cdot]$ keeps the remainder n bits. The decoder does the same basic operation, using its own auxiliary signal,

$$s_t = R_n[\alpha w_{t-1} + s_{t-1}],$$

with the reconstructed value

$$w_t = y_t + T_n[\alpha w_{t-1} + s_{t-1}].$$

ESS can eliminate the limit cycle even if $s_t \neq r_t$.

ESS requires memory in the feedback loop for the prediction. One method of implementation uses one n-bit memory location to store the remainder bits from the previous pixel. In this case, the subscript t refers to spatial location. However, this can cause dynamic difficulties. For example, with $\alpha = 15/16$, if the spatially and temporally constant DC input is 40, then the prediction error signal will alternate in value between -5 and -6. As a result, the 16*16 prediction error block will not only have a DC component, but also a high frequency component.

Alternately, we could implement ESS using enough n-bit memory locations to store the remainder bits for each pixel location in the image. However, this requires a significant amount of additional memory, so it is not recommended.

2.2 Solution 2

The second solution eliminates the limit cycle by using a pseudo-random auxiliary signal. It is equivalent to add an auxiliary signal that varies between 0 and $2^n - 1$ prior to the multiplication by α, as shown in Figure 3. At the encoder,

$$y_t = x_t - T_n[\alpha(x_{t-1} + b_t)],$$

and at the decoder

$$w_t = y_t + T_n[\alpha(w_{t-1} + b_t)].$$

For example, suppose the input value is constant at 40, so $x_t = x_{t-1} = -88$. Then, when $0 \leq b_t < 8$, $y_t = -6$, while when $8 \leq b_t < 16$, $y_t = -5$, if $\alpha = 15/16$. Without adding in the signal b_t, the prediction error $y_t = -6$ always (for truncation with either type of multiplication).

The signal b_t must be the same at the encoder and the decoder to ensure there are no limit cycles. Therefore, it should be sent in the picture layer immediately following the leak factor. This requires an additional 6 bits per picture since the maximum $n = 6$. Because b_t is constant throughout the picture, a DC input gets transmitted as successive DC values, with no additional high frequency components. The subscript t is a temporal index.

The choice of the signal b_t remains a subject of encoder design. However, a signal that works well is a ramp signal with the bits reversed (the most significant bit becomes the least significant bit, etc.).

The D1 tape demonstrates that contouring in the flowergarden sky is removed using this solution.

2.3 Comparison

Both solutions completely eliminate the limit cycle given a DC input. However, their dynamic behavior differs. We give two examples.

First, Table 2 shows the mean squared error between the coded image (at the encoder) and the recovered image at the decoder after a channel change, using a still-flowergarden. Solution 2 converges faster to the correct image than solution 1 particularly in the chrominance components, and has significantly lower MSE than the case with limit cycles.

Second, Table 3 shows the SNR of sequences coded at 4Mbps without channel errors or channel changes when coded using the three methods. Mobile and Flower were coded with $M = 3$ and $\alpha = 7/8$, while Bus and Hockey were coded with $M = 1$ and $\alpha = 15/16$. Both solutions improve the SNR with errors, although the SNR with solution 2 is marginally better than with solution 1.

Frames	With limit cycles	Solution 1	Solution 2
40	88.38 (y)	97.61 (y)	96.51 (y)
	31.7 (cb)	6.3 (cb)	3.5 (cb)
	24.6 (cr)	3.5 (cr)	1.1 (cr)
60	30.32 (y)	10.87 (y)	8.82 (y)
	28.7 (cb)	3.1 (cb)	0.5 (cb)
	23.6 (cr)	2.6 (cr)	0.2 (cr)

Table 2: Mean Squared Error after channel change

Sequence	With limit cycles	Solution 1	Solution 2
Mobile	28.02	28.06	28.07
Flower	28.91	28.93	28.94
Bus	28.35	28.44	28.44
Hockey	36.44	36.53	36.58

Table 3: SNR of sequences without channel effects

3 Recommendations

Because the limit cycle problem appears with both floating-point multiplication and shift-and-subtract multiplication, we suggest that the shift-and-subtract multiplication be adopted for ease of implementation. Furthermore, to eliminate the limit cycles, we suggest that solution 2 be used.

For core experiments, the signal b_t should be a bit-reversed ramp. The following pseudo-code can be used once per picture to generate the current value of b_t, where t is now a time index. The value of the variable ramp is initialized once to zero at the beginning of the encoding process.

```
ramp++;
ramp %= 64;
bt=0;
i=ramp;
for (k=0; k<n; k++) {
  bt = (bt<<1) + (i&0x01);
  i = i>>1;
}
```

The variables bt and i are unsigned characters and k and ramp are integers. The variable bt contains the value of the signal b_t. Note that while ramp varies from 0 to 63, bt will vary from 0 to $2^n - 1$.

The leak factor and value of the signal b_t should be transmitted in the picture layer for every P picture. One appropriate location is after the **chroma_postprocessing_type** bit in the picture layer, with the syntax shown below.

```
if (picture_coding_type == 2) {
  leak_factor                3   uimsbf
  leak_signal_b              6   uimsbf
}
```

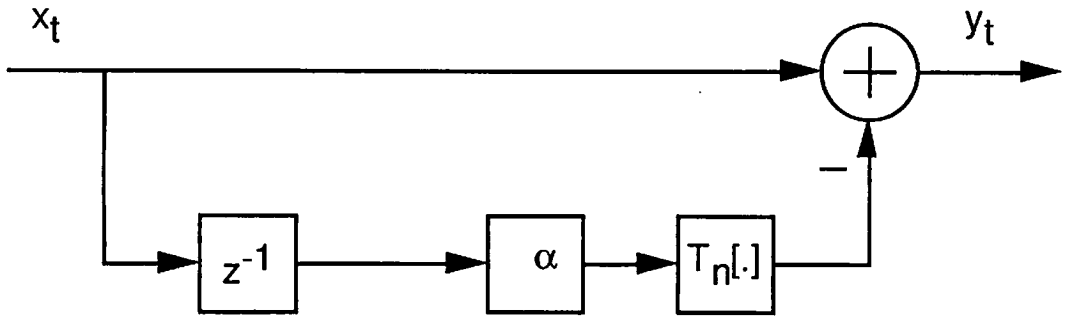

Figure 1(a). Encoder with limit cycles

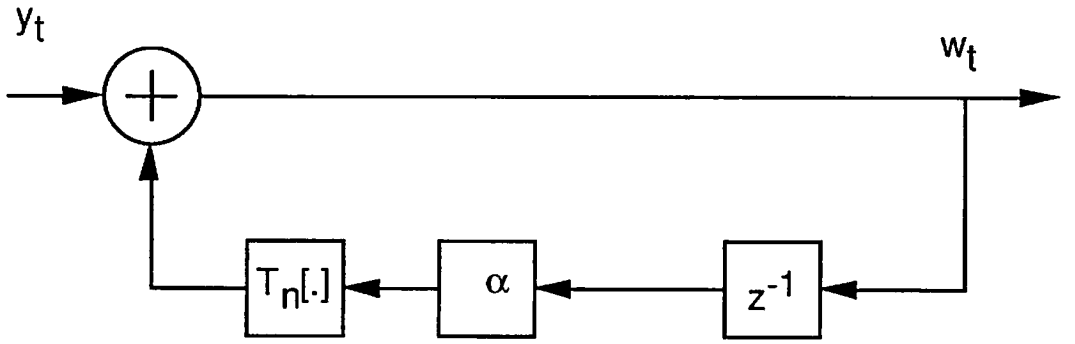

Figure 1(b). Decoder with limit cycles

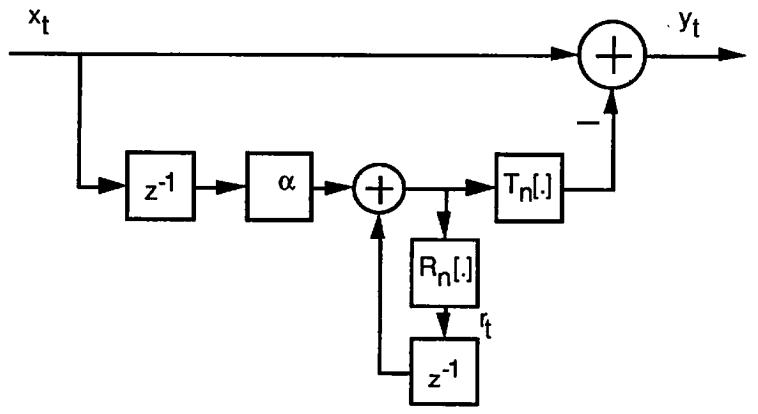

Figure 2(a). Encoder without limit cycles, solution 1

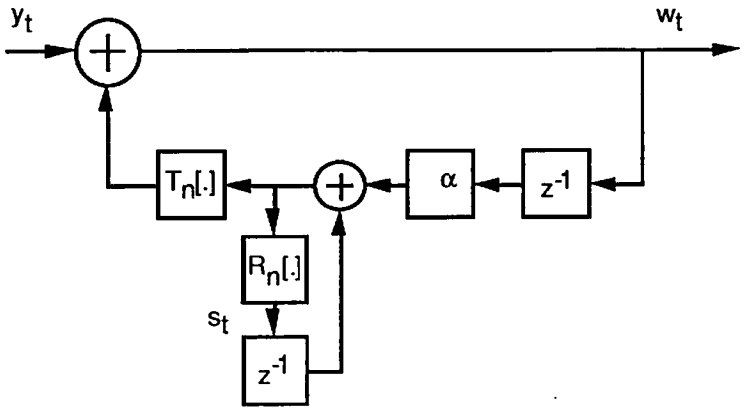

Figure 2(b). Decoder without limit cycles, solution 1

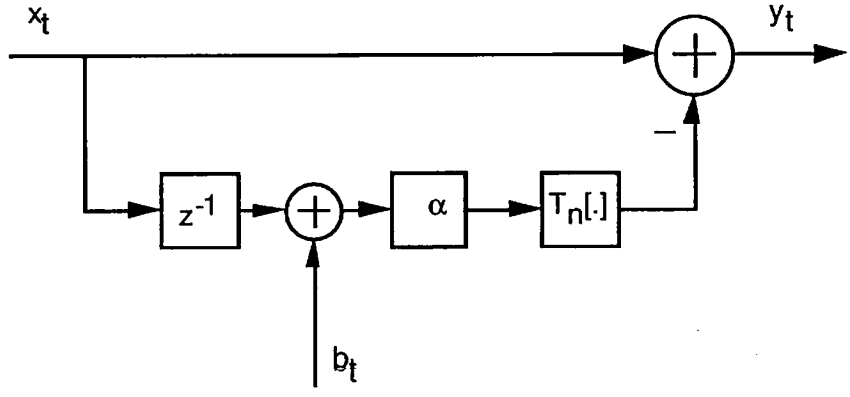

Figure 2(a). Encoder without limit cycles, solution 2

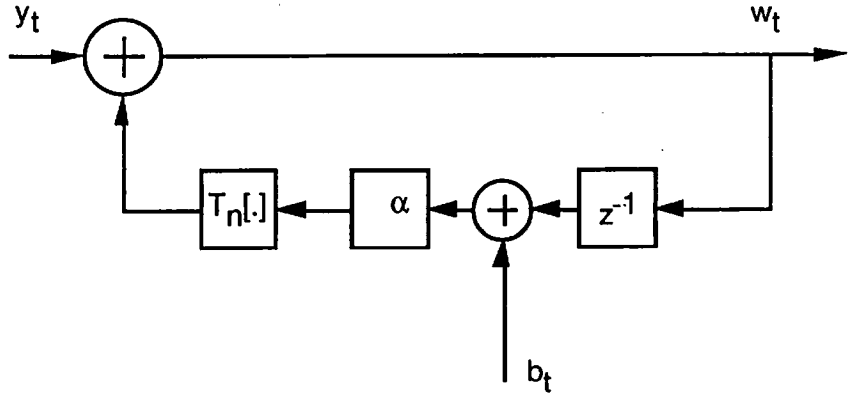

Figure 2(b). Decoder without limit cycles, solution 2