Lossless Image Compression
(Updated 10/28/2011)
[74, 75, 76, 77]
(PI access database from ftp/websites)
(7.3/ p.166)

CALIC (1994)

Context Adaptive Lossless Image Compression

Gray Scale Images  Binary Images

Uses both context & prediction. Two Schemes are:
1) Gray Scale Images  Discuss this only.
2) Bi-level Images

Gray Scale Images

<table>
<thead>
<tr>
<th></th>
<th>NN</th>
<th>NNE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Figure 7.1 labeling the neighbors of pixel X.

(N, W, NW, NE, NN, NE & NNE) previous neighborhood pels available at both encoder & decoder. Horizontal/ Vertical edges?

\[ d_h = |W - WW| + |N - NW| + |NE - N| \]

\[ d_v = |W - NW| + |N - NN| + |NE - NNE| \]

\[ d_h \gg d_v, \hat{x} = N \]

\[ d_v \gg d_h, \hat{x} = W \]

\( \hat{x} \) = weighted average of neighborhood pels
else \( |d_h - d_v| \) = small
Initial Prediction

if $d_h - d_v > 80$
\[ \hat{X} \leftarrow N \]
else if $d_v - d_h > 80$
\[ \hat{X} \leftarrow W \]
else
\[
\begin{align*}
\hat{X} & \leftarrow (N + W)/2 + (NE - NW)/4 \\
& \quad \text{if } d_h - d_v > 32 \\
& \quad \hat{X} \leftarrow (\hat{X} + N)/2 \\
& \quad \text{else if } d_v - d_h > 32 \\
& \quad \hat{X} \leftarrow (\hat{X} + W)/2 \\
& \quad \text{else if } d_h - d_v > 8 \\
& \quad \hat{X} \leftarrow (3\hat{X} + N)/4 \\
& \quad \text{else if } d_v - d_h > 8 \\
& \quad \hat{X} \leftarrow (3\hat{X} + W)/4
\end{align*}
\]

Prediction refinement

Quantify the info about the neighborhood form the vector.

\[ \left[ N, W, NW, NE, NN, W, 2N-N, 2W-W \right] \]
‘1’ or ‘0’

Compare each component of this vector with initial prediction $\hat{X}$. If the component $< \hat{X}$, replace with 1. Otherwise replace with 0. Eight component binary vector.

$256 = 2^8$ possible vectors. As the components are not independent, only 144 possible vectors $\binom{8}{2}$ combinations.

Compute $\delta = d_h + d_v + 2(N - \hat{N})$. This incorporates the vertical/horizontal edges and previous prediction error.

# of texture descriptors $= 144$

Divided range of $\delta$ into 4 intervals, (2 bits). Context for $X$ are $144 \times 4 = 576$. 
CALIC PP. 166-170
(Refinement of initial prediction)
Vector [N, W, NW, NE, NN, WW, 2N-NN, 2W-WW]
Eight component binary vector

Each component < \( \hat{X} \) set to 1
Each component > \( \hat{X} \) set to 0
Where \( \hat{X} \) = Initial prediction of X.

Consider \[ [N, NN, (2N-NN)] \]
Let 1 1 then (1 or 0)
Let 0 0 then (0 or 1)
Let 1 0 Can be 1 only
Let 0 1 Can be 0 only

Total 6 combinations. Similarly for [W, WW, (2W-WW)], i.e., 6 combinations, [NW, NE] has 4 combinations. Total # of combinations for the 8 component vector is
\[ 6 \times 6 \times 4 = 144 \]

8 component binary vector. Each component ‘1’ or ‘0’.
[N, W, NW, NE, NN, WW, 2N-NN] 256 possible vectors.
**CALIC**  PP.166/ Section 7.3

Vector \([N, W, NW, NE, NN, WW, 2N-NN, 2W-WW]\) 

\(\hat{X} \rightarrow \text{predicted value.}\)

Compare each component of the vector with initial prediction \(\hat{X}\).

If value of component \(< \hat{X}\), replace value with ‘1’ else replace value with ‘0’. 

**CASE 1**

```
N   NN   2N-NN
1   1    ‘1’ OR ‘0’ (2 Combinations)
```

When \(N = 1\) it means \(N < \hat{X}\)

\(NN = 1\) it means \(NN < \hat{X}\)

**Example 1)** \(N = 10, \hat{X} = 15, NN = 8\)

\(2N-NN = 20 - 8 = 12\)

\(2N-NN < 15 \rightarrow 2N-NN = 1\)

**Example 2)** \(N = 10, \hat{X} = 11, NN = 8\)

\(2N-NN = 20 - 8 = 12\)

\(2N-NN > 11 \rightarrow 2N-NN = 0\)

**CASE 2**

```
N   NN   2N-NN
0   0    ‘0’ OR ‘1’
```

**Example 1)** \(N = 10, \hat{X} = 8, NN = 15\)

\(2N-NN = 20 - 15 = 5\)

\(2N-NN < \hat{X} \rightarrow 2N-NN = 1\)

**Example 2)** \(N = 50, \hat{X} = 8, NN = 15\)

\(2N-NN = 100 - 15 = 85\)

\(2N-NN > \hat{X} \rightarrow 2N-NN = 0\)
CASE 3

\[
\begin{array}{ccc}
N & NN & 2N-NN \\
1 & 0 & \text{ONLY ‘1’}
\end{array}
\]

Example

\[
N = 10, \widehat{X} = 15, NN = 20 \\
2N-NN = 20 - 20 = 0 \\
2N-NN \text{ is always less than } \widehat{X} \rightarrow 2N-NN \text{ is assigned value 1}
\]

CASE 4

\[
\begin{array}{ccc}
N & NN & 2N-NN \\
0 & 1 & \text{ONLY ‘0’}
\end{array}
\]

Keep track of how much prediction error is generated in each context & offset initial prediction by that amount. This results in final predicted value. Code the residual in terms of the context.

\[
0 \leq \delta < q_1 \Rightarrow \text{Context 1} \\
q_1 \leq \delta < q_2 \Rightarrow \text{Context 2} \\
q_2 \leq \delta < q_3 \Rightarrow \text{Context 3} \\
q_3 \leq \delta < q_4 \Rightarrow \text{Context 4} \\
q_4 \leq \delta < q_5 \Rightarrow \text{Context 5} \\
q_5 \leq \delta < q_6 \Rightarrow \text{Context 6} \\
q_6 \leq \delta < q_7 \Rightarrow \text{Context 7} \\
q_7 \leq \delta < q_8 \Rightarrow \text{Context 8}
\]

\(q_1, \ldots, q_8\) can be prescribed by the user. If original pel values lie between 0 and (M-1), range of residuals is from \(-(M-1))\) to \((M-1)\). Follow description on pages 169 – 170.

Reference:

http://www.hpl.hp.com/loco (Web site for JPEG-LS)

7.4/p.170 JPEG-LS [78] (ISO/IEC standard)
Software in public domain.
HP LABS (LOCO-1) Low Complexity
(Lossless & lossy modes) Simpler than CALIC, Similar to CALIC

Lossless Mode (Initial Prediction)

\[
\text{if } NW \geq \max(W, N) \\
\hat{X} = \max(W, N) \\
\text{else} \\
\{
\text{if } NW \leq \min(W, N) \\
\hat{X} = \min(W, N) \\
\text{else} \\
\hat{X} = W + N - NW
\}
\]

<table>
<thead>
<tr>
<th></th>
<th>NN</th>
<th>NNE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NW</td>
<td>N</td>
<td>NE</td>
</tr>
<tr>
<td>WW</td>
<td>W</td>
<td>X</td>
</tr>
</tbody>
</table>

Pixel under prediction

Fig.7.1/p.166 Neighbors of pel X

This is variation of median adaptive predictor, MAP [79], i.e., prediction is median of N, W, & NW. Refine initial prediction using the average value of prediction error in that particular context.

<table>
<thead>
<tr>
<th>Contexts</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D₁ = NE - N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D₂ = N - NW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D₃ = NW - W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D₁, D₂, D₃ define a 3-component context vector Q₁(Q₁, Q₂, Q₃)
D₁, D₂, D₃ define a 3-component context vector Q (Q₁, Q₂, Q₃)

\[ i = 1,2,3 \quad \text{(p.171)} \]

\[ D_i \leq -T_3 \Rightarrow Q_i = -4 \]
\[ -T_3 < D_i \leq -T_2 \Rightarrow Q_i = -3 \]
\[ -T_2 < D_i \leq -T_1 \Rightarrow Q_i = -2 \]
\[ -T_1 < D_i \leq 0 \Rightarrow Q_i = -1 \]
\[ D_i = 0 \Rightarrow Q_i = 0 \]
\[ 0 < D_i \leq T_1 \Rightarrow Q_i = 1 \]
\[ T_1 < D_i \leq T_2 \Rightarrow Q_i = 2 \]
\[ T_2 < D_i \leq T_3 \Rightarrow Q_i = 3 \]
\[ T_3 < D_i \Rightarrow Q_i = 4 \]

T₁, T₂, & T₃ are positive Constants defined by the user.

# of possible contexts = 9*9*9 = 729.

This reduces these contexts to 365 (0 to 364). Replace any context vector Q whose first nonzero element is negative by \(-Q\). Whenever this happens, a variable SIGN is also set to -1. Otherwise it is set to +1. The vector Q is then mapped into a number between 0 and 364.

Original pel (8 bit PCM), Range: 0 to 255.

Range of prediction error: (-255 to 255)

Prediction error rₙ is mapped into an interval the same size as the range of (0 to M-1) the original pels. Mapping (eg. M = 256)

\[ r_n = -(M/2) \Rightarrow r_n \leftarrow r_n + M \]
\[ r_n > (M/2) \Rightarrow r_n \leftarrow r_n - M \]

Example: \( (M/2) = 128 \)

\[ r_n = -155 < -128 \]
\[ r_n = -155 + 128 = 01 \]
\[ r_n = 155 > 128 \]
\[ r_n = \#(155 - 256) = -101 \]

Code prediction errors based on Golomb Codes
JPEG-LS Combinations

\[ 9 \times 9 \times 9 = 729 \]
\[ Q \rightarrow (Q_1, Q_2, Q_3) \]

\( Q_1, Q_2, Q_3 \) can take on values.
\([-4, -3, -2, -1, 0, 1, 2, 3, 4]\) any of these 9 values.

4(+ve) values.
4(-ve) values.
And
1 zero.

**Case 1:** All zeros \((Q_1, Q_2, Q_3) = 000 \rightarrow 1 \text{ combination} \)

**Case 2**

\[
\begin{align*}
+ & \quad + & \quad + & \rightarrow 4 \times 4 \times 4 = 64 \\
+ & \quad + & \quad - & \rightarrow 64 \\
+ & \quad - & \quad - & \rightarrow 64 \\
+ & \quad - & \quad + & \rightarrow 64 \\
- & \quad + & \quad + & \rightarrow 64 \\
- & \quad - & \quad + & \rightarrow 64 \\
- & \quad + & \quad - & \rightarrow 64 \\
- & \quad - & \quad - & \rightarrow 64
\end{align*}
\]

\[ 64 \times 8 = 512 \text{ combinations} \]

Only 4 of the above 8 combinations are used \(512/2 = 256 \text{ combinations} \).
Case 3  Considering 0’s

\[
\begin{array}{ccc}
Q_1 & Q_2 & Q_3 \\
0 & & \\
1 & 8 & 8 & \rightarrow 64
\end{array}
\]

\[
\begin{array}{ccc}
0 & + & + & \rightarrow 16 \\
0 & + & - & \rightarrow 16 \\
0 & - & + & \rightarrow 16 \\
0 & - & - & \rightarrow 16
\end{array}
\]

\[
\begin{array}{ccc}
0 & 0 & \\
8 & 1 & 8 & \rightarrow 64 \\
8 & 8 & 1 & \rightarrow 64 \\
0 & 0 & \\
1 & 1 & 8 & \rightarrow 8 \\
0 & 0 & \\
1 & 8 & 1 & \rightarrow 8 \\
0 & 0 \\
8 & 1 & 1 & \rightarrow 8 \\
(64 * 3) + (8 * 3) = 216 \text{ combinations}
\]

Only (216/2) combinations are used = 108

Total useful combinations = 1 + 256 + 108 = 365

(0, 0, 0)
TABLE 7.3 Comparison of the file sizes obtained using new and old JPEG lossless compression standard and CALIC.

<table>
<thead>
<tr>
<th>Image</th>
<th>Old JPEG</th>
<th>JPEG - LS</th>
<th>CALIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sena</td>
<td>29,742</td>
<td>27,339</td>
<td>26,433</td>
</tr>
<tr>
<td>Sensin</td>
<td>32,429</td>
<td>30,344</td>
<td>29,213</td>
</tr>
<tr>
<td>Earth</td>
<td>32,137</td>
<td>26,088</td>
<td>25,280</td>
</tr>
<tr>
<td>Omaha</td>
<td>48,818</td>
<td>50,765</td>
<td>48,249</td>
</tr>
</tbody>
</table>

JPEG – Lossless selecting the best result out of eight predictors. Implement all JPEG predictors & select the best. JPEG – LS & CALIC are one pass algorithms. Their performance is similar. JPEG – LS outperforms JPEG – Lossless by 6% to 18%. p.165

TABLE 7.1 Compressed file size in bytes of the residual images obtained using the various JPEG prediction modes.

<table>
<thead>
<tr>
<th>Image</th>
<th>JPEG 0</th>
<th>JPEG 1</th>
<th>JPEG 2</th>
<th>JPEG 3</th>
<th>JPEG 4</th>
<th>JPEG 5</th>
<th>JPEG 6</th>
<th>JPEG 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sena</td>
<td>53,431</td>
<td>37,220</td>
<td>31,559</td>
<td>38,261</td>
<td>31,055</td>
<td>29,742</td>
<td>33,063</td>
<td>32,179</td>
</tr>
<tr>
<td>Sensin</td>
<td>58,306</td>
<td>41,298</td>
<td>37,126</td>
<td>43,445</td>
<td>32,429</td>
<td>33,463</td>
<td>35,965</td>
<td>36,428</td>
</tr>
<tr>
<td>Earth</td>
<td>38,248</td>
<td>32,295</td>
<td>32,137</td>
<td>34,089</td>
<td>33,570</td>
<td>33,057</td>
<td>33,072</td>
<td>32,672</td>
</tr>
<tr>
<td>Omaha</td>
<td>56,061</td>
<td>48,818</td>
<td>51,283</td>
<td>53,909</td>
<td>53,771</td>
<td>53,520</td>
<td>52,542</td>
<td>52,189</td>
</tr>
</tbody>
</table>

TABLE 7.2 Comparison of the file sizes obtained using JPEG lossless compression, GIF, and PNG.

<table>
<thead>
<tr>
<th>Image</th>
<th>Best JPEG</th>
<th>GIF</th>
<th>PNG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sena</td>
<td>29,742</td>
<td>51,085</td>
<td>31,577</td>
</tr>
<tr>
<td>Sensin</td>
<td>32,429</td>
<td>60,649</td>
<td>34,488</td>
</tr>
<tr>
<td>Earth</td>
<td>32,137</td>
<td>34,276</td>
<td>26,995</td>
</tr>
<tr>
<td>Omaha</td>
<td>48,818</td>
<td>61,341</td>
<td>50,185</td>
</tr>
</tbody>
</table>

GIF: Graphics Interchange Format
PNG: Portable Network Graphics uses predictive coding. Each row can be encoded using one of 4 possible predictors. GIF is dictionary based approach. LZ77, LZ78 & LZW.
(7.5/p.172) **Multiresolution Approaches**

**HINT (Hierarchical INTerpolation)**

<table>
<thead>
<tr>
<th>Δ</th>
<th>•</th>
<th>X</th>
<th>•</th>
<th>Δ</th>
<th>•</th>
<th>X</th>
<th>•</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>•</td>
<td>*</td>
<td>•</td>
<td>•</td>
<td>*</td>
<td>•</td>
<td>*</td>
<td>•</td>
<td>*</td>
</tr>
<tr>
<td>X</td>
<td>•</td>
<td>⋄</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>⋄</td>
<td>•</td>
<td>X</td>
</tr>
<tr>
<td>•</td>
<td>*</td>
<td>•</td>
<td>•</td>
<td>*</td>
<td>•</td>
<td>*</td>
<td>•</td>
<td>*</td>
</tr>
<tr>
<td>Δ</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>Δ</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>Δ</td>
</tr>
<tr>
<td>•</td>
<td>*</td>
<td>•</td>
<td>•</td>
<td>*</td>
<td>•</td>
<td>*</td>
<td>•</td>
<td>*</td>
</tr>
<tr>
<td>X</td>
<td>•</td>
<td>⋄</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>⋄</td>
<td>•</td>
<td>X</td>
</tr>
<tr>
<td>•</td>
<td>*</td>
<td>•</td>
<td>•</td>
<td>*</td>
<td>•</td>
<td>*</td>
<td>•</td>
<td>*</td>
</tr>
<tr>
<td>Δ</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>Δ</td>
<td>•</td>
<td>X</td>
<td>•</td>
<td>Δ</td>
</tr>
</tbody>
</table>

*Fig. 7.2 The HINT scheme for hierarchical prediction*

pels Λ: Obtain residuals using linear prediction & transmit.

pels X: estimate by linear interpolation and transmit the error.

pels ⋄: estimate from Δ and X and transmit the error.

pels * and 0: estimate from known neighbors & transmit error.

Reconstruction: Reversal of steps used for transmission.

7.5.1/p.173 PIT – Progressive Image Transmission

See [81-86]

FIGURE 7.4 Comparison between the received image using progressive transmission and using the standard raster scan order.

 Cena image coded using different block sizes for progressive transmission. Top row: block size 8 x 8 and block size 4 x 4. Bottom row: block size 2 x 2 and original image.
(1024 * 1024) Image. 8bpp transmit over 56 Kbps line.

Time to transmit an image = \[\frac{(1024 \times 1024 \times 8)}{(56 \times 10^3)}\] > 2.5 minutes

Btw. Browse thru a large # images thru a database (MRI, CT, CAT, etc.)

