Lossless Image Coding Using Binary Tree Decomposition of Prediction Residuals

Mortuza Ali*, Manzur Murshed*, Shampa Shahriyar†, and Manoranjan Paul‡
* Federation University Australia,
† Monash University, Australia,
‡ Charles Sturt University, Australia,

Emails: {mortuza.ali, manzur.murshed}@federation.edu.au
Email: shampa.shahriyar@monash.edu
Email: mpaul@csu.edu.au

Abstract—State-of-the-art lossless image compression schemes, such as, JPEG-LS and CALIC, have been proposed in the context adaptive predictive coding framework. These schemes involve a prediction step followed by context adaptive entropy coding of the residuals. It can be observed that there exist significant spatial correlation among the residuals after prediction. The efficient schemes proposed in the literature rely on context adaptive entropy coding to exploit this spatial correlation. In this paper, we propose an alternative approach to exploit this spatial correlation. The proposed scheme also involves a prediction stage. However, we resort to a binary tree based hierarchical decomposition technique to efficiently exploit the spatial correlation. On a set of standard test images, the proposed scheme, using the same predictor as JPEG-LS, achieved an overall compression gain of 2.1% against JPEG-LS.

I. INTRODUCTION

State-of-the-art lossless image compression algorithms, such as, JPEG-LS [1], CALIC [2], and EDP [3], have been proposed in the context-based adaptive predictive coding framework. The essentials components of a context-based adaptive predictive coding scheme is prediction followed by entropy coding of the residual. The framework relies on the observation that images can be modeled as some order Markov random processes, i.e., the value of a pixel depends on only neighboring pixels. These causal neighboring pixels can be used for prediction. Prediction decorrelates the image in the sense that the zero-order entropy of the residual is significantly lower than the zero order entropy of the original image. Therefore, encoding of the residuals, instead of the original values, using entropy codes, such as, Huffman codes, usually results in better compression.

Once the value of the pixel to be encoded is predicted, the residual needs to be entropy coded. However, entropy coding of these residuals requires the knowledge of their probabilistic distribution. To make the scheme adaptive, instead of using the same probability distribution throughout the entire image for residuals, a family of probabilistic distribution is used. During entropy coding, a specific distribution from this family is selected based on the image activity level, e.g., edginess, smoothness, around the pixel to be coded. The level of activity around the pixel is its context. The pixels which are used to determine the level of activity of the image in the vicinity of the current pixel form the casual neighborhood template for context modeling. The causal neighborhood template for prediction and context modeling need not be the same. Since straightforward use of the pixel values in the causal neighborhood template to label a context leads to context dilution problem [4], [5], [6], the number of contexts in these schemes is reduced by quantization.

It can be observed that for natural images, even after prediction, there exists significant spatial correlation among the residuals. In context-based adaptive coding this correlation is exploited by conditional selection of the probability distribution based on its context to drive the entropy coder. In this paper, we investigate on using an alternative hierarchical decomposition-based technique to exploit this spatial correlation. A hierarchical decomposition based lossless image compression has been proposed in [7], [8]. This scheme decomposes an original image using a binary tree. It then associate a bitmap with each of the nodes. Finally the bit maps are entropy coded using context adaptive binary arithmetic coding. In contrast, our proposed scheme operates on the residuals since after prediction significant spatial correlation can be observed in the residual images. The essential idea is to hierarchically partition the possible values of the residuals using a binary tree. We associate with each node of the tree a value and a bitmap. The bitmap represents whether the residual value corresponding to a pixel is larger than the value associated with the node. If there exists significant spatial correlation among the neighboring residuals, the bitmap is expected to exhibit strong clustering tendency. Central to the proposed scheme is an efficient bitmap encoder that adaptively partitions the bitmap into rectangular blocks. The partitioning induced by a binary tree aims at isolating large homogenous blocks that can be encoded efficiently. During the entropy coding of the resulting blocks, information from previously encoded bits are used to construct the template for context modeling.

The organization of the rest of the paper us as follows. In Section II we present an bitmap encoding scheme that can efficiently exploit the spatial correlation present in a binary image. By taking a hierarchical decomposition approach we propose a lossless image compression scheme using this
bitmap encoding algorithm in Section III. In Section IV, we present the experimental results to demonstrate the efficacy of the proposed scheme.

II. BITMAP ENCODING

Let \( B \) be a bitmap of size \( H \times W \) having significant spatial correlations. The objective is to efficiently encode the bitmap by exploiting the spatial correlation. In the following section, we argue that bitmaps having significant spatial correlation arise in the proposed hierarchical scheme for lossless image compression.

Spatial correlation in \( B \) is exploited by a rectangular partitioning of the bitmap using a binary tree such that the block of the bitmap covered by a leaf node is either highly correlated or highly uncorrelated. The tree is constructed by a simple divide-and-conquer algorithm with greedy optimization heuristic. Let the current block encompass \( N \) elements of \( B \) of that \( z \) are 0s. The ‘divide’ step first checks whether it contains only one kind of elements, i.e., \( z = 0 \) or \( z = N \). In that case, the block is classified as a Type I (all 0s) or Type II (all 1s) leaf node that requires no further information to encode. Otherwise, the block is split recursively into two sub-blocks along either horizontal-axis or vertical-axis such that 0-1 elements are maximally polarized by the split. This heuristic is expected to divide \( B \) into sub-blocks that are either highly polarized (i.e., almost all 0s or almost all 1s) or further sub-partitioning is unlikely to achieve any compression (i.e., almost random).

A block of size \( h \times w \) can be split in \( h - 1 \), and \( w - 1 \) ways along horizontal-axis and vertical-axis, respectively. Let a split \( s \) form two sub-blocks with \( N_{s,1} \) and \( N_{s,2} \) elements, where \( N_{s,1} + N_{s,2} = N \). Let \( z_{s,1} \) and \( z_{s,2} \) be the number of 0s in the two sub-blocks, respectively, where \( z_{s,1} + z_{s,2} = z \). Polarization of \( 0 - 1 \) elements by this split can be measured as

\[
P_s = \frac{z_{s,1}}{N_{s,1}} - \frac{z_{s,2}}{N_{s,2}}
\]

(1)

It may be noted that \( 1 - 0 \) polarization is same as the \( 0 - 1 \) polarization as

\[
\frac{N_{s,1} - z_{s,1}}{N_{s,1}} - \frac{N_{s,2} - z_{s,2}}{N_{s,2}} = \frac{z_{s,1}}{N_{s,1}} - \frac{z_{s,2}}{N_{s,2}} = P_s
\]

(2)

Then our objective is to find the split \( s^* \) that maximize the \( 0 - 1 \) polarization,

\[
s^* = \arg \max_s P_s
\]

(3)

The ‘conquer’ step of the algorithm discards those sub-partitioning decisions, made during the ‘divide’ step, that do not achieve better compression. If the bits required to independently encode the sub-blocks is greater than that required to encode the current block, the splitting decision is abandoned. In that case, the block is classified as a Type III leaf node, which is compressed by denoting the number of 0s or 1s in \( \log_2 N \) bits and then the context adaptive binary arithmetic coded bit-stream of the block. A split (non-leaf) node is classified into either Type X or Type Y depending on whether the optimal split is found along horizontal-axis or vertical-axis, respectively. The split positions are encoded using fixed-length codes. If the optimal split is found at position \( l \in \{1, \cdots, L\} \), where \( L \) is the range of the block’s dimension along the optimal axis, \( l \) is encoded in \( \lceil \log_2 L \rceil + 1 \) bits.

The type of each node of the tree is identified using Huffman codes shown in Table I that were generated by analyzing the probability of each type in a large set of test video sequences. The entire tree is encoded in the pre-order depth-first traversal sequence where the root of the tree is encoded first, then recursively the left sub-tree, and finally, recursively the right sub-tree.

Figure 1 presents the partitioning of a bitmap achieved by the proposed divide-and-conquer bitmap encoding algorithm. The figure demonstrate the efficiency of the proposed algorithm in partitioning of the bitmap into homogenous blocks.

III. THE PROPOSED SCHEME

The scheme presented in this section is based on the idea of hierarchical decomposition of a frame integers using a binary tree. Hence, the scheme can directly operate on the original images. However, our experimental results demonstrated that encoding of the prediction residuals is more efficient than that of the original images. This is due to the fact that, for natural images, zero-order entropy of the residuals is usually significantly lower than that of the original image. Besides, there exists significant spatial correlation among the residuals after prediction.

The scheme consists of two stages: prediction and hierarchical decomposition. In the prediction stage each of the pixels is predicted, in raster scan order, from previously encoded images.
neighborhood pixels. The resulting frame of residual is then compressed using the following hierarchical decomposition technique.

Let $I$ be a frame of integers of size $H \times W$. Let the elements of $I$ take values from the set $A = \{\alpha_0, \alpha_1, \cdots, \alpha_n\}$. Now $I$ can be decomposed using a binary tree as follows. Associate with each node of the tree a set of integers $S = \{s_1, \cdots, s_n\} \subseteq A$. Initially $S = A$ is associated with the root of the binary tree. At each node, find the integer $t \in S$ such that among all the elements in $I$ having values from $S$, $p$ fraction are less than $t$. Then assign with the node a bitmap $B$ where 0’s indicate that values in the corresponding positions in $I$ are in the range $[s_1, t]$ and 1’s indicate that they are in the range $[t + 1, s_n]$. In the next step, the node is expanded into two children nodes. The set of integers $S$ associated with the node is then partitioned into two subsets $S_l = \{s_1, \cdots, t\}$ and $S_r = \{t + 1, \cdots, s_n\}$ and are assigned to the left and right children nodes, respectively. The procedure is repeated at the children nodes until the cardinality of the set of non-negative integers associated with a node becomes less than 3. These leaf nodes are not expanded as the values of the elements associated with these nodes can be uniquely determined from the associated bitmap.

Now encoding of $I$ can be performed by a pre-order (root-left-right) traversal of the binary tree. During traversal, at each node, the encoder needs to transmit the value of $t$ followed by the encoding of the associated bitmap $B$. The value of $t$ can be encoded using $\lceil \log_2 n \rceil + 1$ bits. The bitmap $B$ can be encoded efficiently by the bitmap encoding algorithm presented in Section II. It may be noted that in bitmaps associated with the nodes, other than the root node, not all the positions are of interest. The bitmap $B$ at the root node represents information about all the elements in $I$. More specifically, if at the root node $A$ is split at $t$, then 0’s in $B$ denotes that the value of the element at the corresponding position is less than or equal to $t$ and 1’s indicates that they are larger than $t$. Then the bitmap $B_l$, associated with the left child node, needs to consider only the positions of 0’s in $B$. Similarly, only the positions of 1’s should be considered in the right child node. Thus, the information about the active positions of the bitmaps associated with a node that need to be coded is available from its parent node.

**Context modeling:** The bitmap encoder partitions the frame into homogenous rectangular blocks which are then entropy coded using context adaptive binary arithmetic coding. The bits in a block are encoded in raster scan order. Since there is significant spatial correlation within a block, three previously encoded neighboring bits are included in the template for context determination. Besides, the bitmap at the parent node also possesses significant correlation with the bitmap at a child node. Essentially they provide information about whether the value of a neighboring element is less than or greater than the element at the current position. In contrast to raster scan ordering, the bitmap from the parent node is available as a whole to the decoder before the start of encoding of the current bitmap at the child node. Thus, the template for context determination can incorporate bits from the all around the current position. To circumvent the context dilution problem, we incorporated four neighboring bits from the parent bitmap in the template for context determination. The context template which include three previously encoded bits from the current bitmap and four neighboring bits from the parent’s bitmap is depicted in Figure 2.

![Fig. 2: Template for context modeling. (a) three bits from the current bitmap and (b) four bits from the parent’s bitmap are included in the context template.](image)

## IV. Experimental Results

We evaluated the performance of the proposed coding scheme on the 24 8-bit gray scale version of Kodak images [9]. The images are of sizes $768 \times 512$ or $512 \times 768$. In the prediction stage of our scheme, we used the median edge detector (MED) predictor from JPEG-LS. Therefore, we compare our performance against JPEG-LS, to demonstrate the efficiency of the proposed scheme in exploiting the spatial correlation in the residuals.

In our implementation, we relied on the sign-magnitude representation of the residuals. The signs of the residuals were first encoded using the bitmap encoder proposed in Section II. In the absence of any parent’s bitmap, we used 7 previously encoded sign bits as context during arithmetic coding of the homogenous blocks resulting from the bitmap partitioning step. The absolute values of the residuals were then encoded using the hierarchical blocks resulting from the bitmap partitioning scheme presented in Section III. In our experiment, we used the value of $p = 0.5$ to find the point $t$ at which to split the set $S$ of integers associated with a nodes.

Ideally the ‘divide’ step of the bitmap encoding scheme may reach to individual residuals which are then merged during the ‘conquer’ step to form bigger blocks. Therefore, the ‘divide’ step can be constraint to a block size $M$ that prevents it from dividing a block of size smaller than $M$. For a reasonable value of $M$, this strategy can significantly reduce the encoding complexity with negligible loss in compression.

In Table II, we present the performance of the proposed scheme against JPEG-LS. In our experiment, we used HP Labs’ software implementation of JPEG-LS [10]. It follows from the table that the coding efficiency of the proposed scheme consistently increased with decreasing value of $M$. The smaller values of $M$ enabled the bitmap encoder to isolate more homogenous blocks which required only few bits to
Fig. 3: Images of the residuals for Image 10 and Image 11. In compressing Image 11, JPEG-LS achieved better compression than the proposed scheme due to its run-length coding mode. However, as the the value of \( M \) gets smaller, the performance of the proposed scheme approached that of JPEG-LS. In the absence of significant runs in the residual for Image 10, the proposed scheme performed better than JPEG-LS for all values of \( M \).

TABLE II: Compression efficiency (bits/pixel) of the proposed scheme against JPEG-LS on the Kodak gray scale images [9]. Since the proposed scheme also used the same predictor as JPEG-LS, the results demonstrate the efficacy of the proposed scheme in exploiting the spatial correlation.

<table>
<thead>
<tr>
<th>Image</th>
<th>Entropy Image</th>
<th>Entropy Residual</th>
<th>JPEG-LS (bit/pixels)</th>
<th>Proposed (bit/pixels)</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>( M = 512 \times 768 )</td>
<td>( M = 256 \times 256 )</td>
<td>( M = 128 \times 128 )</td>
</tr>
<tr>
<td>1</td>
<td>7.16</td>
<td>5.51</td>
<td>5.27</td>
<td>5.22</td>
<td>5.21</td>
</tr>
<tr>
<td>2</td>
<td>5.56</td>
<td>4.28</td>
<td>3.98</td>
<td>4.03</td>
<td>4.01</td>
</tr>
<tr>
<td>3</td>
<td>7.09</td>
<td>3.91</td>
<td>3.46</td>
<td>3.56</td>
<td>3.52</td>
</tr>
<tr>
<td>4</td>
<td>7.12</td>
<td>5.24</td>
<td>5.02</td>
<td>4.92</td>
<td>4.90</td>
</tr>
<tr>
<td>5</td>
<td>7.35</td>
<td>5.57</td>
<td>5.17</td>
<td>5.11</td>
<td>5.11</td>
</tr>
<tr>
<td>6</td>
<td>7.46</td>
<td>5.06</td>
<td>4.56</td>
<td>4.66</td>
<td>4.57</td>
</tr>
<tr>
<td>7</td>
<td>7.02</td>
<td>3.98</td>
<td>3.60</td>
<td>3.65</td>
<td>3.63</td>
</tr>
<tr>
<td>8</td>
<td>7.62</td>
<td>5.45</td>
<td>5.28</td>
<td>5.22</td>
<td>5.21</td>
</tr>
<tr>
<td>9</td>
<td>7.08</td>
<td>4.97</td>
<td>4.82</td>
<td>4.62</td>
<td>4.58</td>
</tr>
<tr>
<td>10</td>
<td>7.12</td>
<td>5.05</td>
<td>4.96</td>
<td>4.74</td>
<td>4.72</td>
</tr>
<tr>
<td>11</td>
<td>6.85</td>
<td>4.77</td>
<td>4.39</td>
<td>4.46</td>
<td>4.42</td>
</tr>
<tr>
<td>12</td>
<td>6.98</td>
<td>4.10</td>
<td>3.80</td>
<td>3.86</td>
<td>3.83</td>
</tr>
<tr>
<td>13</td>
<td>7.43</td>
<td>6.23</td>
<td>5.96</td>
<td>6.00</td>
<td>5.97</td>
</tr>
<tr>
<td>14</td>
<td>7.44</td>
<td>5.17</td>
<td>4.90</td>
<td>4.91</td>
<td>4.90</td>
</tr>
<tr>
<td>15</td>
<td>7.41</td>
<td>4.30</td>
<td>3.87</td>
<td>3.95</td>
<td>3.89</td>
</tr>
<tr>
<td>16</td>
<td>7.19</td>
<td>4.47</td>
<td>4.05</td>
<td>4.16</td>
<td>4.10</td>
</tr>
<tr>
<td>17</td>
<td>7.26</td>
<td>4.45</td>
<td>4.92</td>
<td>4.17</td>
<td>4.15</td>
</tr>
<tr>
<td>18</td>
<td>6.98</td>
<td>5.38</td>
<td>5.85</td>
<td>5.13</td>
<td>5.11</td>
</tr>
<tr>
<td>19</td>
<td>7.38</td>
<td>4.82</td>
<td>5.36</td>
<td>4.56</td>
<td>4.49</td>
</tr>
<tr>
<td>20</td>
<td>6.34</td>
<td>3.91</td>
<td>3.11</td>
<td>3.34</td>
<td>3.22</td>
</tr>
<tr>
<td>21</td>
<td>7.01</td>
<td>4.96</td>
<td>4.50</td>
<td>4.63</td>
<td>4.56</td>
</tr>
<tr>
<td>22</td>
<td>7.19</td>
<td>4.82</td>
<td>4.54</td>
<td>4.58</td>
<td>4.55</td>
</tr>
<tr>
<td>23</td>
<td>7.25</td>
<td>3.83</td>
<td>3.49</td>
<td>3.60</td>
<td>3.59</td>
</tr>
<tr>
<td>24</td>
<td>7.11</td>
<td>5.04</td>
<td>4.60</td>
<td>4.62</td>
<td>4.59</td>
</tr>
<tr>
<td>Average</td>
<td>7.10</td>
<td>4.80</td>
<td>4.56</td>
<td>4.49</td>
<td>4.45</td>
</tr>
</tbody>
</table>

 encode (see Figure 3). Out of 24 images, JPEG-LS performed better than the proposed scheme in compressing 11 images. However, for these images the performance of the proposed scheme was within less than 3% of that achieved by JPEG-LS. For 4 images the performance of the proposed scheme was same as that of JPEG-LS. For rest of the images, the proposed scheme outperformed JPEG-LS achieving a compression gain as high as 16.5%. On average, the compression achieved by the proposed scheme was 2.1% more than that achieved by JPEG-LS.

REFERENCES


