Optimization and Generalization of Lloyd’s Algorithm for Medical Image Compression

J. Subash Chandra Bose, B. Saranya and S. Monisha

Department of CSE, Professional Group of Institutions Palladam, Tirupur, Tamilnadu, India

Abstract: During the Optimization and Generalization of Lloyd’s algorithm for medical image Compression process the modified algorithm searches for empty cells i.e. the unused code vectors. Also the modified algorithm keeps track of the image vector producing maximum distortion corresponding to the code vector representing maximum number of image vectors. This maximum misrepresentation of image vectors are used to replace the unused code vectors thus removing the empty cells from the code book. As the repeated process takes place on the empty cells are completely removed and the final code book is free from empty cells. The objective of image compression is to reduce redundancy of the image in order to be able to store or transmit data in an efficient form. The performance measures can be compared using parameters such as MSE (Mean Square Error) and PSNR (Picture Signal to Noise Ratio). The Implementation has been done by using MATLAB 7.0.

Key words: Distortion • Lloyd’s algorithm • Cells • Image compression • Image vector

INTRODUCTION

The modified algorithm is used to overcome the Empty Cell Problem found in the LBG algorithm. The modified algorithm is very similar to the LBG Algorithm except for the optimization of the code vectors. As the additional code vectors are generated by adding offset to the existing code vectors they may not get mapped to any image vector and remain unused creating an Empty Cell.

The performance measures can be compared using parameters such as MSE (Mean Square Error) and PSNR (Picture Signal to Noise Ratio).

\[
MSE = \frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} [I(x,y) - I'(x,y)]^2 
\]

(3.1)

\[
PSNR = 20 \times \log_{10} \left( \frac{255}{\sqrt{MSE}} \right) 
\]

(3.2)

where \( I(x,y) \) is the original image, \( I'(x,y) \) is the reconstructed version and \( M,N \) are the dimensions of the images [5].

Related Work

Graphics Interchange Format (GIF): GIF files can be saved with a maximum of 256 colours. This makes it a poor format for photographic images. Because this can sometimes be tight, GIFs have the option to dither and will mix pixels of two different available colours to create a suggestion of another colour.

Portable Network Graphics (PNG): PNG’s main drawback is alpha-channels. Instead of the rudimentary transparency options in other formats (where a pixel is either transparent or opaque), an alpha channel can specify the opacity of any pixel from 0-255, where 0 is fully transparent and 255 is fully opaque.

Tagged Image File Format (TIF): The TIFF format can use one of five data compression schemes: Huffman, Pack Bits, LZW, Fax Group 3 and Fax Group 4. LZW compression is generally used for 24 bits per pixel images. Many applications that support TIFF do not support data compression; therefore it is advisable not to use data compression except for temporary interchange between applications known to support the required type of compression.

BMP/DIB/RLE: The Microsoft Windows 3.x environment on the PC employs a special family of formats for storage and display of bitmap images of 1, 4, 8 or 24 bits per pixel.
The Windows bitmap is supported by all Windows applications that support bitmaps. However it is poorly supported elsewhere. The BMP/DIB/RLE formats supersede the obsolete old and new MSP formats employed by previous versions of Windows for bit level bitmaps.

**LBG Algorithm:** There are many algorithms for codebook generation; a common approach for the generation of the code book is the use of popular k-mean algorithm proposed by Linde Buzo and Gray also known as GLA (Generalized Lloyd Algorithm). K-means algorithm is an iterative gradient descent algorithm that tries to minimize an average squared error distortion measure. Its performance is sensitive to the initialization of the code book. It may generate a code book with (few or some) code vectors, which are never used.

**Empty Cell Problem:** In this algorithm there is danger that an output point will never be used. A common approach to avoid this is to remove an output point that has no inputs associated with it and replace it with a point from the quantization region with most output points.

**Proposed Algorithm:** The modified algorithm is used to overcome the Empty Cell Problem found in the LBG algorithm. The modified algorithm is very similar to the LBG Algorithm except for the optimization of the code vectors. As the additional code vectors are generated by adding offset to the existing code vectors the may not get mapped to any image vector and remain unused creating an Empty Cell.

During the optimization process the modified algorithm searches for empty cells i.e. the unused code vectors. Also the modified algorithm keeps track of the image vector producing maximum distortion corresponding to the code vector representing maximum number of image vectors. This maximum distortion image vectors are used to replace the unused code vectors thus removing the empty cells from the code book. As the iteration goes on the empty cells are completely removed and the final code book is free from empty cells.

As the modified code book is free from empty cells it is more efficient than the LBG algorithm and produces a better picture quality. The performance measures can be compared using parameters such as MSE (Mean Square Error) and PSNR (Picture Signal to Noise Ratio).

\[
MSE = (1/MN) \sum_{y=1}^{M} \sum_{x=1}^{N} [I(x, y) - I'(x, y)]^2 \quad \text{(3.1)}
\]

PSNR = \[20 \cdot \log_{10} \left(\frac{255}{\sqrt{\text{MSE}}}\right)\] \quad \text{(3.2)}

where \(I(x,y)\) is the original image, \(I'(x,y)\) is the reconstructed version and \(M,N\) are the dimensions of the images.

The new image compression algorithm overcomes the Empty Cell Problem. It consists of two phases.

- Initialization of the code book.

**Code Book Initialization:** The original GLA algorithm uses splitting technique to initialize the codebook. This technique basically doubles the size of the codebook in all iteration.

This procedure starts with one code vector, \(C^{(0)}\) that is set to the average of all training vectors.

**Step 1:** In a general iteration there will be \(k\) code vectors in the code book. \(C^{(0)} = 1,2,..,k\).

Split each code vector into two code vectors \(C^{(0)}\) and \(C^{(0)} + r\), where \(r\) is a fixed perturbation vector. Set \(k = 2k\).

**Step 2:** If there are enough code vectors, stop the splitting process. The current set of \(k\) code vectors can now serve as the initial set \(C^{(0)}\) for the code book optimization phase. If more code vectors are needed, execute the optimistic algorithm on the current set of \(k\) entries, to converge them to a better set; then go to Step 1.

**Code Book Optimization:** Step 0: Select a threshold value ‘\(E\)’, set \(k = 0\) and \(D^{(0)} = 8\). Start with initial code book with code vectors \(C^{(0)}\) [where \(k\) is currently zero, but will be incremented in each iteration]. Training vectors are denoted as \(T_i\).

**Step 1:** For each code vector \(C^{(0)}\) find the set of all training vectors \(T_i\) that satisfies equation (1),

\[d(T_i, C^{(0)}) < d(T_j, C^{(0)})\quad \text{i \neq j}\]

This set (or cell) is denoted as \(P^{(0)}\). Repeat for all \(i\). It may happen that some cells will be empty [Empty Cell Problem].

**Step 2:** Calculate the distortion \(D^{(0)}\) between each code vector \(C^{(0)}\) and the set of training vectors \(P^{(0)}\) found for it in Step 1. Repeat for all \(i\), then calculate the average \(D(k)\) of all \(D^{(0)}\). A distortion \(D^{(0)}\) for a certain \(i\) is calculated.
Table 1: Training Images

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Training Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reface.jpg</td>
</tr>
</tbody>
</table>

Fig. 1: Training Images

Table 2: Performance Metrics

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Performance Metrics</th>
<th>Formula for computing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mean Square Error (MSE)</td>
<td>((1/MN) \sum_{y=1}^{M} \sum_{x=1}^{N} [I(x,y) - I(x,y)]^2)</td>
</tr>
<tr>
<td>2</td>
<td>Picture Signal to Noise Ratio (PSNR)</td>
<td>(10 \log (255^2 / \text{MSE}))</td>
</tr>
<tr>
<td>4</td>
<td>Compression Ratio</td>
<td>Bytes (Compressed Image) / Bytes (Original Image)</td>
</tr>
<tr>
<td>5</td>
<td>Compression Rate</td>
<td>Bytes (Ori Img) / Bytes (Comp.Img) / Bytes (Original Image)</td>
</tr>
</tbody>
</table>

Fig. 2: REface.jpg using 512 code vectors

by computing the distances \(d(C^{(k)}, T_m)\) for all training vectors \(T_m\) in the set \(P_i^{(k)}\), then calculating the average distance.

**Step 3:** If \((D^{(k+1)} - D^{(k)}) / D^{(k)} \leq \varepsilon\), stop.

Otherwise, continue

**Step 4:** \(k = k + 1\), find new code vectors \(C_i^{(k)}\) that are the average of training vectors in partition \(P_i^{(k-1)}\) that was computed in Step 1.

**Step 5:** Check for Empty Cells and replace with image vectors producing maximum distortion. Go to Step 1.
Table 3: Performance Analysis Comparison of Compression Ratio:

<table>
<thead>
<tr>
<th>Code Book size:</th>
<th>Training Images</th>
<th>LBG Algorithm</th>
<th>Proposed Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>REface 0.9930</td>
<td>0.9930</td>
<td>0.9894</td>
<td></td>
</tr>
<tr>
<td>Code vector size: 4 x 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REface 0.9979</td>
<td>0.9949</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code vector size: 8 x 8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Book size: 256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REface 4.9764</td>
<td>5.2938</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code vector size: 4 x 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Book size: 256</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REface 1.3601</td>
<td>1.4709</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code vector size: 8 x 8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Implementation Using MATLAB 7.0: The new image compression algorithm has been simulated using MATLAB 7.0. First the optimized code book is generated for training images given in Table 1 and shown in Fig. 1. Then the copy of the optimized code book is maintained at both the transmitter and the receiver.
Comparison Metrics: The following are the list of metrics used to analyze the performance of the new image compression algorithm in comparison with the existing LBG algorithm Shown in the Table 2.

CONCLUSION

As the modified code book is free from empty cells it is more efficient than the LBG algorithm and produces a better picture quality. The performance measures can be compared with LBG Algorithm and Proposed Algorithm. Simulation results provided by compressing the studied images validate the interest of the proposed approach shown in the Table 3 and the Figure 4. A good performance is obtained and the quality is preserved on the most important part, i.e. on the diagnostically important region. In future the addition research work is the region of interest which is lossily compressed in the proposed algorithm can be lossless compressed. This will result in a better image quality but the compression rate will be low.

REFERENCES