IMAGE CODING AT VERY LOW BITRATES WITH REGULAR TRIANGULAR MESH

Cesar S. Barreto[1], Gelson V. Mendonça[1,2], and Eduardo A. B. da Silva[1,2]

Universidade Federal do Rio de Janeiro
1. PEE/COPPE 2. DEL/EE
Caixa Postal 68504, CEP 21945-970
Rio de Janeiro, Brazil
barreto, gelson, eduardo@lps.ufrj.br

ABSTRACT

A method to code images by approximating them by a triangular mesh is presented. The coder encodes only the amplitudes of mesh nodes (triangle vertices) and the directions and shapes of mesh triangles. The decoder obtains the pixel amplitudes by interpolating the vertex amplitudes of the triangles they belong to.

Since the used mesh is regular, the chosen size of the mesh triangles influences the quality of the reconstructed image. Using 16×8-pixel triangles, the compression ratios that lead to the most competitive results are around 256:1 (very low bitrates). Results obtained hitherto are very promising, even when compared with state-of-the-art wavelet coders, thus motivating us in further explorations.

1. INTRODUCTION

Meshes are largely used to represent computer generated synthetic images. One advantage of meshes is the continuity of their elements: pixels in the mesh polygon boundaries are common to at least two polygons, and undesirable blocking effects do not occur. Mesh coding is directly applied to images without a domain transformation; this can be an advantage in video coding, because the same approach can be used in intra-frame and difference-frame coding. Additionally, meshes are well suited to motion estimation in video sequences. Such image representation structures have already been studied in the past few years, giving interesting results [1].

The proposed image coder approximates the images to a regular triangular mesh. Initially, the amplitudes of mesh nodes are chosen (as the mean value of the pixels around the node). Then, the best direction (horizontal or vertical) of triangles are chosen. After that, the diagonal arcs are swapped where the error is reduced. Then, the amplitudes of mesh nodes are optimized. Finally, the node amplitudes are quantized and coded in bit planes as in [2] and flags indicating arc directions and swappings are coded directly.

2. MESH REPRESENTATION

2.1. Mesh definition

To form a mesh, we first need to choose the type of polygon. In computer graphics, triangles, quadrilaterals and hexagons are used mostly. Since the triangle is the simplest polygon and is the most flexible one in a tridimensional space (we may change the position of only one vertex and it continues to be a triangle), we have chosen it.

After that, we have to make another decision: either to adapt the mesh to the image or to adapt the image to the mesh. In other words, either the mesh may change the position of its nodes to approximate the edges of the image or the image has its edges approximated by the fixed mesh.

The first choice seems to lead to better quality representation, but with higher coding cost. The second option, with fixed mesh position, is image independent and easier to be coded. We have chosen the second option. Therefore, we will adapt each image to a regular fixed triangular mesh.

The next step is to define the triangle shape and size. Thinking in coding optimization, we have chosen rectangular isosceles triangles with 16-pixel bases and 8-pixel heights, positioned to form a regular mesh, as shown in Figure 1(a).

2.2. Image approximation

The aim of our coder is to approximate images by a mesh and, then, to code only the amplitudes of mesh nodes (tri-

angle vertices). Then, triangle sides or arcs are repositioned to optimize image approximation.

After repositioning arcs, it is necessary to adjust node amplitudes. After adjusting node amplitudes, it is a good idea to iterate the process, repeating the arc swapping tests, and the adjustment of node amplitudes. This must be carried out until the required convergence is obtained. (Our tests showed that at most three iterations are needed.)

2.3. Pixel interpolation

Each image pixel is interpolated from the three vertices of the triangle containing it. The interpolation is linear, as in [1], that is, mesh surfaces are formed by several flat triangles. With such an approach, the resulting images are blurred and dented. That is so because the triangles are a bit large and in a few fixed positions.

2.4. Arc swapping

This limitation can be alleviated by swapping some triangles from horizontal to vertical position, as shown in Figure 1(b). This swapping is carried out whenever it leads to an improvement in the performance. Although this implies in an overhead of informing which triangles were swapped, there is an overall improvement in terms of rate$\times$distortion.

Another way to improve the image quality is by swapping "diagonal" arcs (the smaller sides of the triangles) from the original ± 45 degree directions to "nearly horizontal" or "nearly vertical" directions, as shown in Figure 1(c). Again, this is done wherever this leads to a reduction in the error. Also, it is necessary to inform which of the possible diagonal swaps were taken.

The direction or state of horizontal and vertical (H/V) arcs (which define whether the triangles are horizontal or vertical) sometimes prohibit diagonal swaps that would give considerable improvement in error reduction. Therefore, in these cases, we allow the swapping, and change the corresponding H/V arc directions.

Once a diagonal swap is carried out, the swapping of a number of surrounding arcs is prohibited. Therefore, it is important to first analyse the effect of each arc swapping and then execute those swaps in order of decreasing reduction of the squared error.

2.5. Node optimization

Node amplitudes are initially chosen as the mean value of neighboring pixels. After arc swapping, these values are changed in order to minimize the squared error in the regions affected by them. This is done by trial and error, incrementing and decrementing, separately, the initial value of each node. Since a change in one node amplitude interferes with the optimum value for neighboring nodes, this process must be iterated. (In our tests, three iterations were sufficient for the convergence.)

3. QUANTIZATION AND CODING

The value of node amplitudes and the state of arcs, swapped or not, are now ready to be encoded. In the case of amplitudes, it has been achieved using quantization by bit planes followed by arithmetic encoding [3].

The states of arcs are directly arithmetic encoded. The states of H/V arcs are encoded first, followed by the states of diagonal arcs (taking into account the positions where they have the permission to be swapped). Two models (histograms) are used in each one of both types of arcs, to exploit tendencies of neighboring arcs and, consequently, improve compression ratio.

3.1. Node interpolation

Instead of coding amplitudes of all nodes, we can substitute some of them by the mean value of amplitudes of a pair of nodes from the larger triangle (with four times its area) that contains it, as depicted in Figure 2. This can be done whenever the error introduced in node amplitudes is less than a predefined threshold.

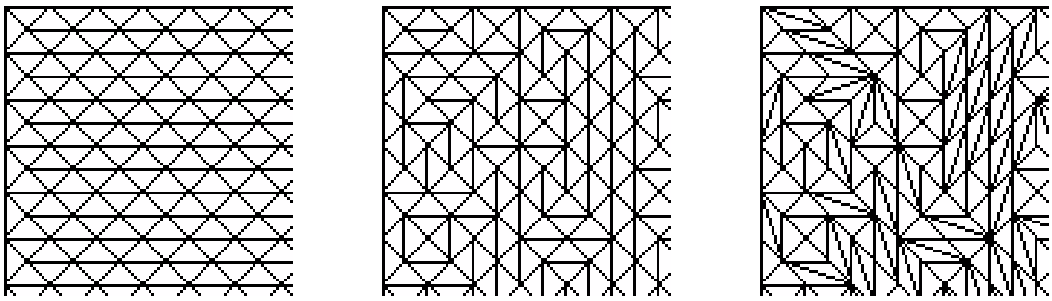
In such cases, we indicate it in a node map that is further encoded. To exploit the frequent cases where two or more adjacent triangles have their nodes interpolated, we use two histograms to code this map. If at least one adjacent, and already scanned, triangle has been interpolated, the used histogram is different from the one used in the opposite case.

When the error is greater than the threshold, we quantize and encode the node in bit planes.

3.2. Swapping thresholding

After node interpolation, flags used to indicate arc swappings begin to produce a large overhead in the whole bit stream.

Sometimes an arc swap does not considerably improve the image quality, although it contributes to increase the bitrate. In these cases, the reduction in the error is not worth the bits spent.



(a) (b) (c)

Fig. 1. (a) Regular triangular mesh structure, (b) H/V arc swapping, and (c) diagonal arc swapping.

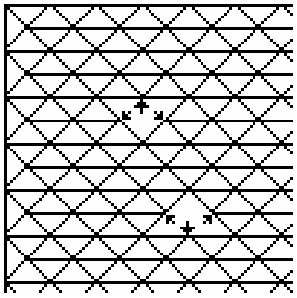


Fig. 2. Mesh structure pointing nodes that are not encoded, but are interpolated from node pairs of the larger triangle.

To solve this problem in a rate×distortion sense, we have tested some values of swapping thresholds and have taken only the swaps that produced such squared error reduction, at least.

4. EXPERIMENTAL RESULTS

Initially, we approximated the 512×512 Lena and Mandrill images with the mesh. Then, we quantized using 4 bit planes for the node amplitudes, and arithmetic coded them. After that, we tested combinations of node interpolation and/or swapping thresholding. The obtained results are presented in Tables 1 and 2, as follows:

1. 4-bit quantization, without node interpolation and swapping thresholding;
2. 4-bit quantization, with a 8000 squared error reduction swapping threshold;
3. 4-bit quantization, with a 8000 squared error reduction for node interpolation and swapping thresholds;
4. 4-bit quantization, with a 15000 squared error reduction for node interpolation threshold and 10000 for swapping threshold; and
5. 4-bit quantization, with a 20000 squared error reduction for node interpolation and swapping thresholds.

The images in part (a) of Figures 5 and 6 were mesh coded and correspond to the second last line of Tables 1 and 2, respectively. The images in part (b) of the same Figures were obtained using the wavelet-based SPIHT [4] coder, at the same coding rates, for comparison.

5. CONCLUSIONS AND FUTURE WORKS

Although the coded images have undesirable artifacts, that is because they are at very high compression ratios, nearly 256:1. With node interpolation and swapping thresholding, the PSNR are considerably close to the results obtained with wavelet codecs [4] for the same compression ratio, and the visual quality are comparable (mesh artifacts frequently seem to be less annoying than low bitrate wavelet ringing artifacts).

For further improvements, we should continue the optimization of the coarse mesh based coding (for example, differentially quantizing the nodes) and, then, hierarchically divide the mesh, as presented in [1], to better represent image high frequency details. Another planned future work is to substitute the linear pixel interpolation by a non-linear interpolation in order to smooth polygon edges, as used in computer graphics shading.

Table 1. Coding Results for Lena Image with Triangular Mesh and SPIHT.

Node Bits	Node Thr.	Swap Thr.	Code (bytes)	Rate (bpp)	Mesh PSNR (dB)	SPIHT PSNR (dB)
4	0	0	1416	0.0432	25.37	27.06
4	0	8000	1178	0.0359	25.20	26.47
4	8000	8000	1080	0.0330	25.16	26.21
4	15000	10000	1021	0.0311	25.02	25.96
4	20000	20000	914	0.0279	24.78	25.60

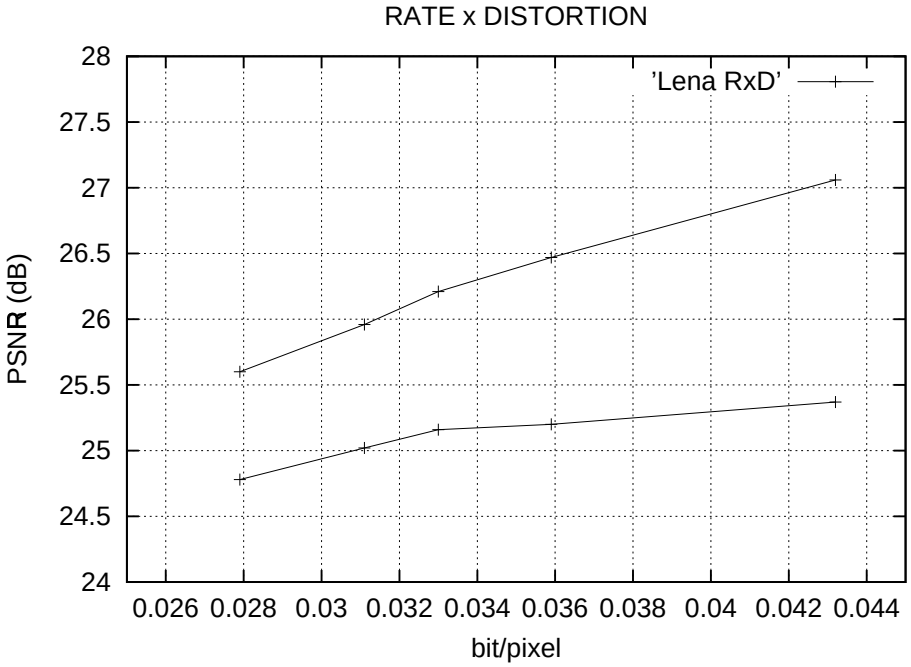


Fig. 3. Rate×distortion results for Lena image with triangular mesh (inferior curve) and SPIHT (superior curve).

Table 2. Coding Results for Mandrill Image with Triangular Mesh and SPIHT.

Node Bits	Node Thr.	Swap Thr.	Code (bytes)	Rate (bpp)	Mesh PSNR (dB)	SPIHT PSNR (dB)
4	0	0	1424	0.0435	20.06	20.22
4	0	8000	1227	0.0374	19.97	20.10
4	8000	8000	1182	0.0361	19.94	20.08
4	15000	10000	1070	0.0323	19.87	20.00
4	20000	20000	874	0.0263	19.74	19.86

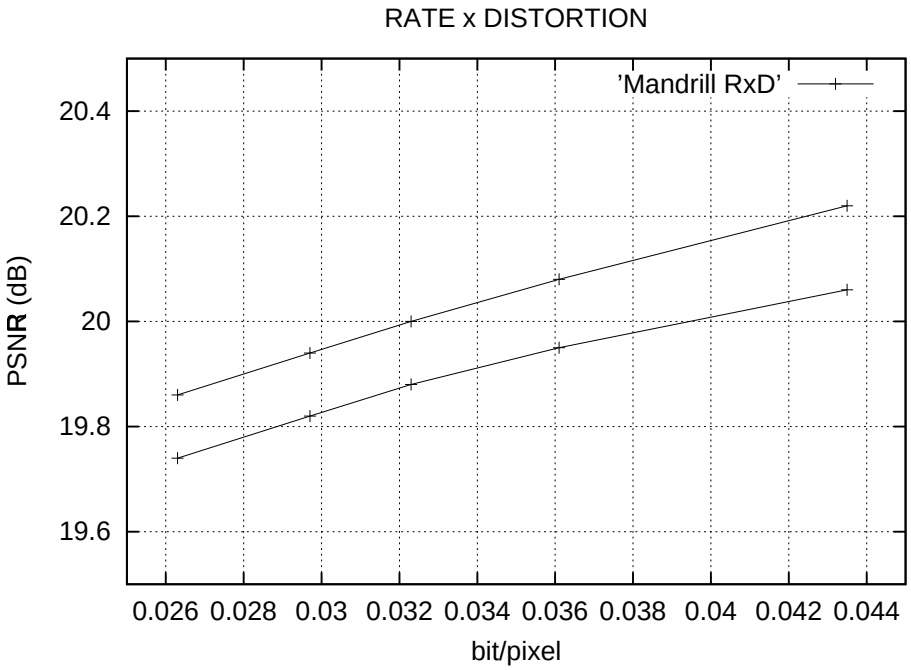


Fig. 4. Rate×distortion results for Mandrill image with triangular mesh (inferior curve) and SPIHT (superior curve).

(a) (b)

Fig. 5. Lena image coded at 0.0311 bpp with (a) triangular mesh (PSNR = 25.02 dB), and (b) SPIHT (PSNR = 25.96 dB).

(a) (b)

Fig. 6. Mandrill image coded at 0.0323 bpp with (a) triangular mesh (PSNR = 19.87 dB), and (b) SPIHT (PSNR = 20.00 dB).

6. REFERENCES

[1] P. Lechat, N. Laurent, and H. Sanson, "Scalable image coding with fine granularity based on hierarchical mesh," in *Proc. VCIP*, Jan. 1999.

[2] J. M. Shapiro, "Embedded image coding using zerotrees of wavelets coefficients," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.

[3] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, pp. 520–540, June 1987.

[4] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, June 1996.