l_∞ NORM BASED SECOND GENERATION IMAGE CODING

Marco Dalai and Riccardo Leonardi

Universitá degli Studi di Brescia
Dipartimento di Elettronica per l'Automazione
via Branze, 38 - I-25123 Brescia, Italy
Email: {marco.dalai, riccardo.leonardi}@ing.unibs.it

ABSTRACT

Many second generation image coding techniques have been studied in recent years. Most of these methods consider the l_2 norm of the error introduced in the coded image, while for the l_∞ case only predictive or transform based methods were considered up to now, focusing on near-lossless coding. In this paper we present a first scheme for l_∞ norm in the framework of second generation image coding. The image is adaptively segmented into rectangular regions of varying size leading to a binary tree decomposition. The grey levels of the pixels within every leaf are approximated by means of l_∞ sub-optimal bilinear surfaces.

1. INTRODUCTION

The field of image coding has received a lot of attention in recent years and different methods have been studied, ranging from transform based, to predictive based and segmentation based ones. Second generation coding techniques have demonstrated better performance at high compression ratios. The performance evaluation is in general determined using the SNR value, i.e. by evaluating the l_2 norm of the introduced error. The study of the l_∞ error measure, in fact, has been limited to near-lossless coding up to now. In this case one is interested in a very small error (typically a maximum error of 4 over 256 levels) and, for this task, transform based and prediction based techniques have been studied ([1, 2]). The use of the l_∞ norm for wider ranges of error value, and in the framework of second generation image coding, seems to have received little interest up to now. In this paper we present a simple segmentation based image coding method with l_∞ norm error control. The image is adaptively divided into rectangular areas. The result partition determines a binary tree. Each leaf of the tree is coded by using a bilinear l_∞ sub-optimal approximation to the pixel grey levels; the value of the bilinear surface at the corners of the rectangles are coded, so that in the decoding phase only bilinear interpolation of these values is required to reconstruct the image within each rectangle.

The paper is organized as follow. We first explain how to construct an optimal bilinear approximation of grey levels over a rectangular domain; we show then how a sub-optimal solution can be derived so as to gain in computational efficiency and address the image decomposition phase. The segmentation algorithm is then described, followed by a description of the coding strategy. In particular for each rectangle only the values of its four corner need to be specified within the original quantization resolution. Finally some experimental results in coding standard images are shown.

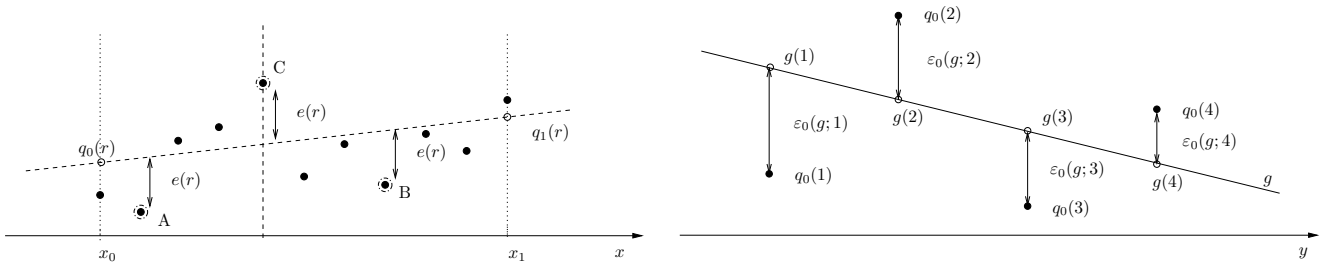
2. l_∞ BILINEAR APPROXIMATIONS

2.1. Stating the problem

Given a signal $s(x, y)$ defined over a discrete grid in a rectangular domain $D = \{(x, y) : x_0 \leq x \leq x_1, y_0 \leq y \leq y_1\}$ we aim at finding a function f, of the form $f(x, y) = axy + bx + cy + b$, that is a good approximation to s under the l_∞ norm. We start by saying that, as a, b, c and d parameters determine a linear space, finding their optimal values (according to a l_∞ criterion) can be reduced to solving a linear program in $\mathbb{R}^5$ ([3, 4]). By using recent linear programming techniques (see [5]) one can thus solve the problem in $O(n)$ expected operations, being n the number of image samples. Nevertheless, the use of this methods can be considered computationally expensive when applied to images. Furthermore, by using such techniques, one can only determine whether or not the signal can be approximated within the domain of interest below an error threshold, and one has no idea of what is the local behavior of the signal s inside D. This means that if it becomes necessary given the problem constraint (error threshold) to divide D in two subrectangles in an adaptive way, one has to choose a partition line by studying s in some other manner.

For these reasons we propose to adopt a separable approach in constructing a sub-optimal approximation f of s. Given that the function f is linear in x for every fixed y and *viceversa*, we perform optimal straight line approximation of s along every row and column of D respectively.

0-7803-8554-3/04/$20.00



a) Straight line approximation of a row

b) Straight line approximation of the $q_0(r)$ points

Fig. 1. Construction of the sub-optimal bilinear approximation.

However, such straight lines may not belong to a bilinear approximation, as their intersections with the image rectangle support boundaries do not necessarily lie on a straight line.

2.2. Sub-Optimal Approximations

Suppose the domain D has R rows and C columns. For every $r = 1 \dots R$ we compute the l_∞ optimal linear approximation of the r-th row of s; as explained in [6] this can be done in a very efficient way by means of geometric arguments (roughly speaking, 8 times faster than linear programming method). Such geometric approach determine at the same time the approximation error and a subset of three points of the row, which we call pivot points, where the approximation commits its maximum error (see fig. 1a, where A, B and C are the pivots). For every r we thus obtain a segment $l(r)$ which has an associated error $e(r)$; this segment intersect the planes $x = x_0$ and $x = x_1$ in two points that we call respectively $q_0(r)$ and $q_1(r)$. Let us focus on the points $q_0(r)$, $r = 1 \dots R$ (without loss of generality this applies also to the points $q_1(r)$ and their equivalent along each column). They all lie in the plane $x = x_0$ but, in the general case, they are not aligned. If we want the segments $l(r)$ to belong to a bilinear surface, it is necessary to vertically move the extremes $q_0(r)$ so as to align them. The corresponding line is not, in this case, necessarily the l_∞ optimal one. Suppose we approximate the points $q_0(r)$ with a line g which takes value $g(r)$ at the r-th row. We indicate with $\varepsilon_0(g; r)$ the approximation error of g over the point $q_0(r)$ that is $\varepsilon_0(g; r) = |g(r) - q_0(r)|$. Remember that the generic straight line $l(r)$ has a maximal error $e(r)$ over the row r; if we force its extremity to move from $q_0(r)$ to $g(r)$, an error less than or equal to $\varepsilon_0(g; r)$ is added to $e(r)$; so, we can state that the total error is at most $e(r) + \varepsilon_0(g; r)$. Thus, a good idea is to select as optimal straight line approximation of the points $q_0(r)$ the line h given by

$$h = \arg\min_g \left(\max_r (e(r) + \varepsilon_0(g; r)) \right). \qquad (1)$$

For every r, consider the points $q_0^+(r)$ and $q_0^-(r)$ obtained by moving respectively up and down the point $q_0(r)$ of a value $e(r)$. Then it is easy to see that

$$e(r) + \varepsilon_0(g; r) = \max(|g(r) - q_0^+(r)|, |g(r) - q_0^-(r)|).$$

Thus, the line h that minimizes (1) is the l_∞ optimal approximation of the set of points $\{q_0^+(r), q_0^-(r)\}_{r=1\dots R}$. The same arguments[1] hold in the plane $x = x_1$, so that we can construct a bilinear surface. It is obvious that the same scheme can be used by interchanging the roles of rows and columns, i.e. linear approximating the columns and then adjusting the extremities of the straight lines in the planes $y = y_0$ and $y = y_1$. In this way we find two generally different solutions, and obviously we choose the one with smallest error.

The main advantage of using this suboptimal approximation, in comparison with the optimal one given by the linear programming techniques, is that the execution time is about 10 times smaller, while the error of approximation is not substantially greater. Moreover, the suboptimal approximation provides an indication of the local approximations error on each row and column, which is a good hint for the segmentation task, as it is shown in the next section.

3. SEGMENTATION

Now, we show which criterion has been designed to build the decomposition of the image. The segmentation scheme is based on a binary tree structure. The image is initially approximated by a bilinear surface; if the error is larger than a given bound δ, the image is split in two smaller rectangles (not necessary equal) by dividing it along a vertical or horizontal line. The procedure is then applied recursively to each rectangle, until every leaf of the tree is approximated with a maximum error smaller than δ, or the area is so small that it is cheaper to represent every of its pixels rather than

[1] It is important to note that when approximating the points $q_1(r)$ with a line $j(r)$, the error to be considered is just $e(r) + \varepsilon_1(j; r)$ and not $e(r) + \varepsilon_0(g; r) + \varepsilon_1(j; r)$. This means that $\varepsilon_0(g; r)$ and $\varepsilon_1(j; r)$ must not be summed, because when moving the line $l(r)$ the total l_∞ error is actually at most $e(r) + \max(\varepsilon_0(g_0; r), \varepsilon_1(j; r))$.

constructing the optimal bilinear approximation of any further subdivision, in a rate distortion sense.

At every recursion level the partitioning line is chosen as follows. We identify the row or the column such that the error committed by its minmax straight line approximation ($e(r)$ or $e(c)$) is the largest with respect to all values of r and c. As mentioned previously, the maximum error is always occurring at one of the three pivots of the row or column pixels. We set then the partition line to be orthogonal to the selected row or column close to its central pivot point. This pivot point is left on one side or the other based on the collinearity with the two consecutive or preceding points. Once the partition line has been selected, the procedure is applied to each sub-rectangle.

It is important to note that, if for example the partition line is vertical, after dividing the domain one does not need to recompute the straight approximation of the columns, as there is no difference with respect to the preceding step. The same thing holds for horizontal partitions. This means that at every recursion level one has to compute either row approximations, or the column ones. This leads to a reduction of 50% in computation, which overall makes the algorithm about 20 times faster than using a linear programming approach.

4. CODING

Now that the segmentation method has been described, the following coding method has been designed. The tree structure is coded as usual; one bit is needed for every node, to specify whether it is a leaf or not. If the node is not a leaf, one bit is spent to code the direction, horizontal or vertical, of the partition line; in addition at most $\log_2(R)$ or $\log_2(C)$ bits are necessary to identify the position of the partition line (note that R and C are the dimensions of the rectangle that is being partitioned). If, instead, the node is a leaf, then it is necessary to code the bilinear surface approximation of the pixels. For this purpose it is sufficient to code the values of the surface on the corners of the rectangle, by quantizing them with the same precision as the one of the original image. The motivation is the following. Suppose that we want to code an image s with a maximum error δ, which is an integer value. We segment the image so as to obtain, in each rectangle, bilinear approximations with error less than δ. Let us call f the obtained surface in a given rectangle; when coding f we quantize the values reached at the corners of the rectangle. This means that, when decoding, we obtain[2] a different surface $\tilde{f}$. Clearly, within the rectangular domain, $\|f - \tilde{f}\|_\infty \leq 1/2$. Furthermore, when computing the real values of the pixel inside the rectangle, we have to quantize $\tilde{f}$ so as to obtain a integer-valued reconstruction $\tilde{s}$ of the original image. Again we have $\|\tilde{s} - \tilde{f}\| \leq 1/2$. Thus we can say that

$$\|s - \tilde{s}\| \leq \|s - f\| + \|f - \tilde{f}\| + \|\tilde{f} - \tilde{s}\| < \delta + 1,$$

so that, as $\|s - \tilde{s}\|$ must be an integer, $\|s - \tilde{s}\| \leq \delta$.
This means that if one constructs approximations with error smaller than the maximum value δ, then one does not need to care about the effect of quantization made at every corner.

In the coding phase, it is clearly possible to use prediction. If we code the image in a recursive way, starting with the top left rectangles first, we are sure that when any new rectangle is reached, all in left and above neighbors have been already coded. So, the values of the bilinear surface on its corners can be predicted from the pixel values above or to the left. A very simple first order prediction scheme has been selected in this work but a more accurate study of the involved statistics could improve further the results.

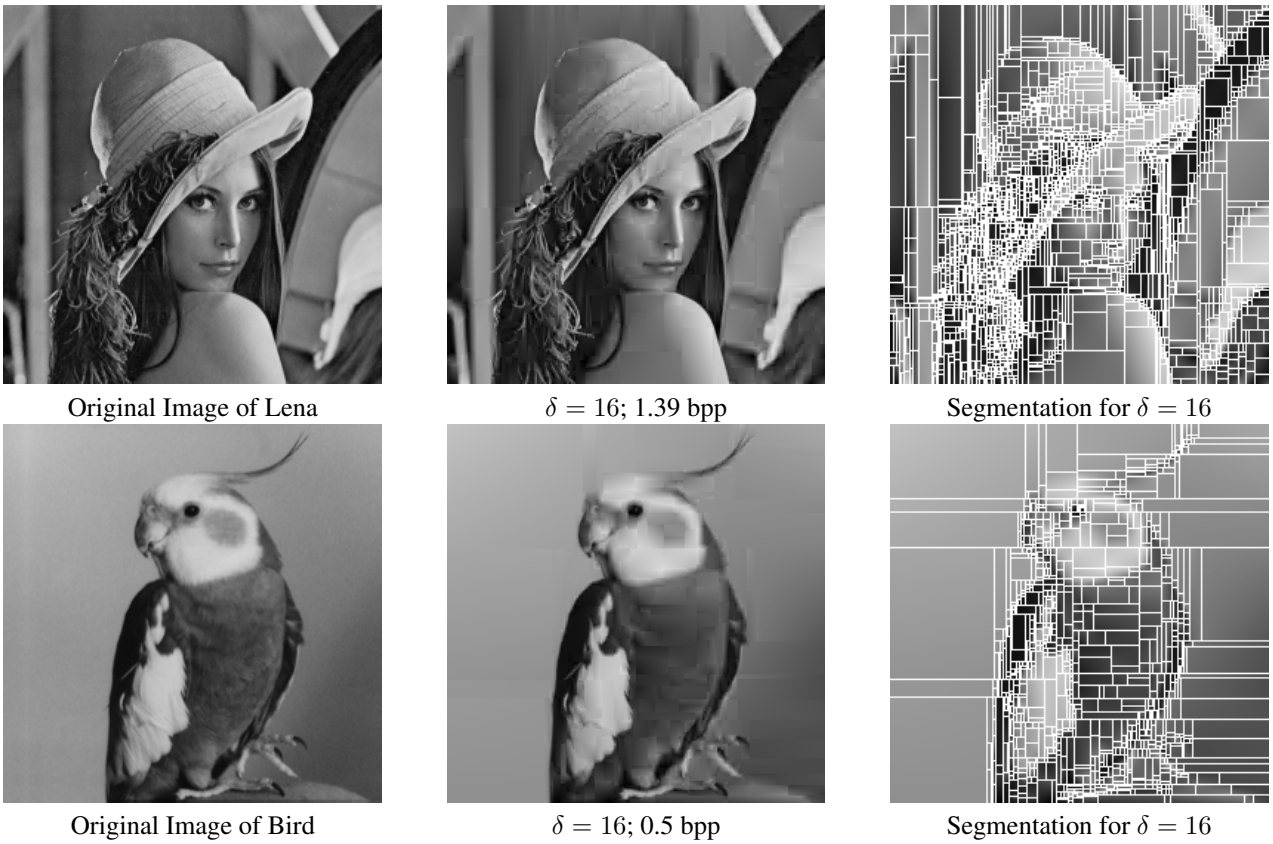
5. SIMULATIONS

For testing the proposed method, we demonstrate its performance in coding two standard images, namely Lena and Bird images, of 256×256 pixel, 8 bpp, available on the Wateloo BragZone site ([7]). The results are shown in table 1, where we report the computation time with a Pentium II 400 MHz processor and the compression rate for values of δ ranging from 4 to 16. Fig 2 shows the original images, the images compressed with $\delta = 16$ and the associated segmentation obtained in this case.
Even if, for small values of δ, the method cannot compete with other techniques for near lossless compression (for example with the JPEG-LS standard), the obtained results are very interesting for the whole range of values of δ. Many considerations can be made to improve the compression scheme. The objective of the work has however demonstrated the benefit of addressing second generation coding with the use of the l_∞ norm.

	Lena		Bird	
δ	bpp	time	bpp	time
4	3.57	1.72	1.68	1.36
6	2.73	1.54	1.12	1.09
8	2.26	1.48	0.89	0.96
10	1.95	1.35	0.75	0.91
12	1.72	1.29	0.65	0.88
14	1.54	1.23	0.57	0.85
16	1.39	1.19	0.50	0.83

Table 1. Computation time in seconds and bpp obtained in compressing Lena and Bird for different values of δ.

[2]Be aware of the fact that a bilinear surface is constructed by linearly interpolating the values on the four corners of a rectangle



Original Image of Lena | $\delta = 16$; 1.39 bpp | Segmentation for $\delta = 16$

Original Image of Bird | $\delta = 16$; 0.5 bpp | Segmentation for $\delta = 16$

Fig. 2. Results of simulations in coding the images Lena and Bird.

6. CONCLUSION

In this paper we have presented a simple second generation image coding technique with l_∞ norm error control. The method is based on segmentation of the image in rectangular region, in which near optimal bilinear approximations of the pixel values are used. We have shown how to use sub-optimal strategies so as to obtain a good compromise between the rate distortion performance of the coder and the computational complexity of the method.

7. ACKNOWLEDGMENTS

This material is based upon work partially supported by the IST programme of the EU in the project IST-2001-32795 SCHEMA.

8. REFERENCES

[1] L. Karry, P. Duhamel, and O. Rioul, "Image coding with an L^∞ norm and confidence interval criteria," *IEEE Trans. Image Proc.*, vol. 7, no. 5, pp. 621–631, 1998.

[2] A. Alecu, A. Munteanu, P. Schelkens, J. Cornelis, and S. Dewitte, "Wavelet-based fixed and embedded L-infinite-constrained coding," *Proc. IEEE Int. Conf. Digital Signal Processing*, pp. 509–512, 2002, Santorini, Greece.

[3] G. A. Watson, "Approximation in normed linear spaces," *J. Comput. Appl. Math.*, vol. 121, pp. 1–36, 2000.

[4] I. Borrodale and C. Phillips, "Solution of an overdetermined system of linear equations in the Chebyshev norm [F4] (algorithm 495)," *ACM Trans. Math. Software*, vol. 1, no. 3, pp. 264–270, 1975.

[5] R. Seidel, "Small-dimensional linear programming and convex hulls made easy," *Discrete Comput. Geom.*, vol. 6, pp. 593–613, 1991.

[6] M. Dalai and R. Leonardi, "Efficient (piecewise) linear minmax approximation of digital signals," *To appear in Proc. ICASSP'04*.

[7] "Waterloo BragZone and Fractals Repository," `http://links.uwaterloo.ca/bragzone.base.html`.