R-D Analysis of Adaptive Edge Representations

Rosa M. Figueras i Ventura, Lorenzo Granai and Pierre Vandergheynst
Signal Processing Institute
Swiss Federal Institute of Technology
1015 Lausanne, Switzerland
Email:{Rosa.Figueras, Lorenzo.Granai, Pierre.Vandergheynst}@epfl.ch

***Abstract*— This paper presents a Rate-Distortion analysis for a simple horizon edge image model. A quadtree with anisotropy and rotation is performed on this kind of image, giving a toy model for a non-linear adaptive coding technique, and its Rate-Distortion behavior is studied. The effect of refining the quadtree decomposition is also analyzed.**

I. Introduction

It is known that the Rate-Distortion (R-D) behavior of wavelets is not optimal for natural images because they are not capable of seeing the regularity of edges [1]. In fact the optimality of monodimensional wavelets is lost when using 2D separable basis, giving a R-D decay of $O(R^{-\frac{1}{2}})$ [2]. This motivates a lot of research in order to find a more efficient way for coding images [3], [4], [5], [6].

The goal of this work is to try to find an efficient technique to represent edges. To compute the R-D behavior we will use the "Horizon" model for piece-wise smooth images where the edge is also a smooth curve. For an image $I(x_1, x_2)$ defined on the unit square $[0,1]^2$, the horizon model defines the image as:

$$I(x_1, x_2) = 1_{x_2 \geq y(x_1)} \qquad 0 \leq x_1, x_2 \leq 1, \tag{1}$$

where $y(x_1) \in \mathbf{C}^p$ is p-times continuously differentiable and has finite length inside the unit square. A way to represent this image is through a quadtree decomposition, which is in fact a toy model for wavelets. Do, Dragotti, Shukla and Vetterli [7] already demonstrated that the R-D of this model decays as R^{-1}. To improve its R-D behavior, they introduced refinement in the isotropic quadtree technique, as further explained in Section II.

In this paper we do another step towards the understanding of the behavior of edges in compression. For this, anisotropy and rotation are directly introduced in the basic quadtree structure, obtaining a toy model for an adaptive non-linear image representation tool like bandelets [8]. Anisotropy makes the first derivative of the edge function appear in the R-D expression. Furthermore, when rotation is also included in the scheme, R-D is affected by the curvature of the edge, showing that the rate needed to represent a given contour is directly proportional to its geometrical complexity, which is coherent with the empirical and previous theoretical results.

II. Optimal quadtree based compression

The optimal quadtree is based on a dyadic division of the unit interval $[0,1]^2$ (see Fig. 1(a)). At each scale, the algorithm will keep on dividing the edge squares until the maximum number of iterations J has been reached. Finally, a refinement (coding with a certain number of bits where the edge crosses the square) will be performed. Then the edge will be represented by the lines that join these refinement points (so it will be represented as a piece-wise linear function). This approach gives a rate distortion decay of:

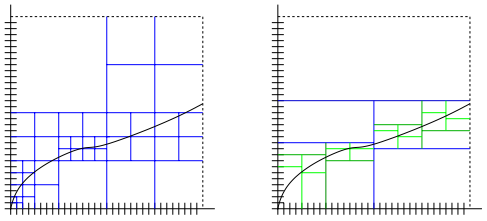
$$D(R) \sim \frac{\log R}{R^2}. \tag{2}$$

III. Anisotropic Quadtree

Let us now release the assumption of the original quadtree decomposition in order to bring adaptivity in the scheme. The difference between the dyadic quadtree and the anisotropic quadtree is the size of the partitions. The x axis will maintain the dyadic partition, but the y axis partition will never cross the edge, as can be seen in Fig. 1(b). After the first partition, the dyadic rectangle containing the edge will be divided into two along the x axis. This process will be repeated iteratively until the desired accuracy or bit-rate is reached (i.e. until the height or the width of the rectangle has the size 2^{-J}). In addition, a certain number of bits can be assigned to code the edge position in the division border, so that a straight line can approximate it. For this case, we supposed that the maximum slope of the edge inside the interval is one ($|y'(x)| \leq 1$), otherwise one could simply switch the axes.

Let J be the number of bits used for quantizing each of the two axes. N_{a_j} is the number of all the rectangles at iteration j and N_e the number of edge-rectangles:

$$N_{a_j} \sim 2^j, \quad N_e \leq 2^{J+\lceil \log_2 y'_{\max} \rceil}. \tag{3}$$



(a) Isotropic (b) Anisotropic

Fig. 1. Example of two different quadtree decompositions.

As next iteration rectangle will be inside previous iteration one, the number of bits needed to code the size of each rectangle will decrease with the iteration number as:

$$N_{\text{bits}} \leq J - (j-1) + \lceil \log_2 y'_{\max} \rceil. \quad (4)$$

Taking into account the bits needed to code whether a rectangle is black, white, edge or intermediate and the size and position, the total bit-rate needed to code the anisotropic quadtree will be given by:

$$R = (2 + N_{\text{bits}})N_a + 2M \cdot N_e, \quad (5)$$

where M is the number of bits used for the refinement. The first term counts the bits needed to describe the tree partition position and whether the partition is black, white, intermediate or edge, and the second term represents the bits needed to code the edge position in the finest partition. Merging (3) and (4) with (5), considering $M{=}J$ and high bit-rate and simplifying, we obtain:

$$R \sim J \cdot 2^{J+\lceil \log_2 y'_{\max} \rceil}. \quad (6)$$

The final distortion (at resolution 2^{-J}) is:

$$D(R) = \int_{[0,1]^2} (I - \hat{I})^2 \leq C \cdot 2^{-J-M}, \quad (7)$$

where $\hat{I}$ is the reconstruction of I. When considering $M{=}J$ and high bit-rate, it gives $D(R) \leq C2^{-2J}$, and from the previous expression and (6) we obtain:

$$D(R) \sim \frac{2^{2\lceil \log_2 y'_{\max} \rceil} \log^2 R}{R^2}. \quad (8)$$

In the above expression we see that the R-D of the irregular quadtree is similar to the one of the regular quadtree but with a factor $2^{2\lceil \log_2 y' \rceil}$ that comes from using anisotropy. The use of rectangles gives a factor $2^{\lceil \log_2 y' \rceil}$ and the coding of the position where the edge crosses the border of the rectangle gives the other $2^{\lceil \log_2 y' \rceil}$.

IV. Introducing Rotation

The anisotropic quadtree shows that the edge representation can be improved in a R-D sense if partitions follow the behavior of the edge. Developing this idea it is possible to use not only rectangular, but even rotated boxes. Let us take the curve in the unit interval $[0,1]^2$ and join the two extreme points with a line that represents its average slope. This line can be then moved up and down such that it does not cross the edge anymore in order to create the box. This procedure is repeated iteratively continuing to split the x axis inside the previous box in a dyadic way (see Fig. 2(b)). As in the anisotropic quadtree algorithm, the y axis partition will basically depend on the edge.

At each iteration j ($0 \leq j \leq J$) and for each box k the distortion is limited by the area of the box that encloses the edge (see Fig. 2(a)):

$$D_j^k \leq S_j^k H_j'^k = S_j^k H_j^k cos\theta = 2^{-j} H_j^k. \quad (9)$$

Inside every box the edge function will be approximated by its second order Taylor expansion at the central point of the partition, taking as initial partition the unit interval.

Defining x_- as the lowest point in the x axis which is inside the interval to analyse and x_+ as the highest one, the coordinates of the two extreme points of the curve quantized on a dyadic grid will be (x_-,$Q[y(x_-)]$) and (x_+, $Q[y(x_+)]$). The line which joins these two points is:

$$y_{LQ}(x) = Q[y(x_-)] + \frac{Q[y(x_+)] - Q[y(x_-)]}{x_+ - x_-}(x - x_-), \quad (10)$$

where $Q[\cdot]$ stands for uniform quantization. The parallelogram cannot cross the edge, therefore its superior and inferior distances to the line are:

$$\begin{array}{rcl} d_+ & = & \max\{0, \sup(y - y_{LQ})\} \geq 0 \\ d_- & = & \max\{0, \sup(y_{LQ} - y)\} \geq 0. \end{array} \quad (11)$$

Then the height of the parallelogram confining the edge will be:

$$H = Q\left[d_+ + d_-\right]. \quad (12)$$

Three cases have to be considered: d_+ and d_- are both bigger than zero, one of them is equal to zero, and finally $d_+ = d_- = 0$. The distortion is at most the area of the parallelogram that contains the edge, as already shown in (9). So, when the evolution of H with the number of bits is found, the evolution of the distortion as a function of the iteration number will be known as well.

A. Case $d_+ > 0$ and $d_- > 0$

1) Distortion: Let's first compute the distances d_+ and d_- in order to find out H. If x_{d_+} is the point in the x axis where $y_{LQ} - y$ is maximum, we get:

$$d_+ = y(x_{d_+}) - y_{LQ}(x_{d_+}). \quad (13)$$

The above expression, when approximating the curve and $y(x_-)$ of (10) by its second order Taylor expansion at the central point of the interval being analysed, turns to:

$$\begin{aligned} d_+ = & \left[y'\left(\tfrac{x_+ + x_-}{2}\right) - \tfrac{y(x_+) - y(x_-)}{x_+ - x_-}\right](x_{d_+} - x_-) \pm \\ & \pm 2^{-J}\left(\tfrac{x_{d+} - x_-}{x_+ - x_-}\right) \pm \tfrac{2^{-J}}{2} + \\ & + \tfrac{1}{2}y''\left(\tfrac{x_+ + x_-}{2}\right)\left[\left(x_{d_+} - \tfrac{x_+ + x_-}{2}\right)^2 - \left(\tfrac{x_+ - x_-}{2}\right)^2\right] + \\ & + O\left(\left(x_{d_+} - \tfrac{x_+ + x_-}{2}\right)^3\right) + O\left(\left(\tfrac{x_- - x_+}{2}\right)^3\right). \end{aligned} \quad (14)$$

It is possible to show that the first term in (14) is $O\left(2^{-3j}\right)$ [6] and that the 4^{th} term is bounded by $\frac{1}{2}\left|y''\left(\frac{x_+ + x_-}{2}\right)\right| 2^{-2(j+1)}$. This expression is related to the second derivative computed in the middle point of each interval k at each iteration j. From now on, to simplify the notation, it will be referred to as K_j^k.

$$K_j^k = \left|y''\left(\left(k + \frac{1}{2}\right)2^{-j}\right)\right|, \quad (15)$$

with $0 \leq k \leq 2^j - 1$. Since the curvature of a function is $\frac{y''(x)}{(1+y'^2)^{\frac{3}{2}}}$, K can be considered as its approximation. As the edge is a $\mathbf{C}^2$ curve, the set of K_j^k is bounded:

$$\beta = \max_{0 \leq k \leq 2^J - 1} K_j^k < \infty. \tag{16}$$

From (14) it follows that the asymptotic behavior of d_+ is given by:

$$d_+ \sim \frac{1}{2} K_j^k 2^{-J - log_2 \beta} + \frac{3}{2} \cdot 2^{-J}. \tag{17}$$

In fact the other terms are $O(2^{-3j})$ (one order of magnitude smaller), so they can be rejected when computing the asymptotic behavior. Finally d_- can be found with exactly the same method and has identical behavior.

The iterative algorithm is going to stop when the requested resolution is reached, i.e. when $H_j = 2^{-J}$. Substituting in (12), the number of iterations needed to reach this parallelogram height can be obtained as a function of the resolution and of the curvature of the edge:

$$j_{\text{stop}} = max \left\{ 0, \left\lceil \frac{1}{2} \left(J + \lceil log_2 \beta \rceil + 1 \right) \right\rceil \right\}. \tag{18}$$

Notice that now the parallelogram has the $\frac{a}{a^2}$ anisotropy present in curvelets [4]:

$$\frac{\text{width}}{\text{height}} = \frac{2^{-j_{\text{stop}}}}{2^{-J}} = 2^{-\log_2 \beta} \frac{2^{-\frac{J}{2}}}{2^{-J}} \sim \frac{\text{width}}{\text{width}^2}, \tag{19}$$

The final distortion ($j = j_{\text{stop}}$) is:

$$\begin{aligned} D &= \textstyle\sum_{k=0}^{2^{j_{\text{stop}}}-1} (d_+ + d_-)\, 2^{-j_{\text{stop}}} = \\ &= 2^{-2j_{\text{stop}}} \left(\textstyle\sum_{k=0}^{2^{j_{\text{stop}}}-1} K_{j_{\text{stop}}}^k 2^{-j_{\text{stop}}} + \sum_{k=0}^{2^{j_{\text{stop}}}-1} 3 \cdot 2^{-j_{\text{stop}}} \right). \end{aligned}$$

In the right-hand side of previous equation, the second sum gives a constant, while the first, when $j_{\text{stop}} \to \infty$, converges to the Riemann integral of the second derivative of the curve, which can be seen as an approximation of the total variation TV of the edge, with the only difference that we have a sum of K_j^k instead of the Riemann integral of the curvature. Calling it $\widetilde{\text{TV}}$ the final expression of the distortion turns to:

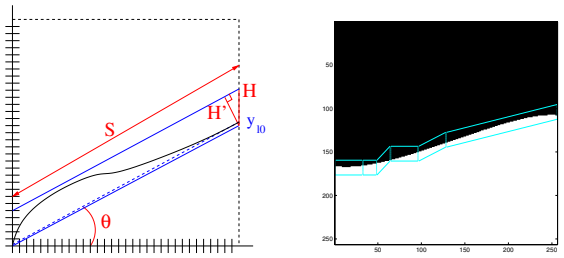
$$D \sim \left(\widetilde{\text{TV}} + 3 \right) \cdot 2^{-2j_{\text{stop}}} \sim \left(\widetilde{\text{TV}} + 3 \right) \cdot 2^{-J - log_2 \beta}. \tag{20}$$

2) Rate: Each rotated box is coded by means of H_j and a left and a right vertex. At iteration j, as at least two of the vertices of the following parallelogram will be inside the previous one, the number of bits needed to code one vertex of the box k will evolve as follows:

$$N_{\text{bits\_V}}^k = J - 2(j-2) + \left\lceil \log_2 \left(K_{j-1}^k \right) \right\rceil. \tag{21}$$

Therefore, the total rate will be:

$$R = \sum_{j=0}^{j_{\text{stop}}} (2 N_{\text{bits\_V}} + N_{\text{bits\_H}}) \cdot 2^j, \tag{22}$$



(a) Scheme (b) Final resolution

Fig. 2. Anisotropic quadtree with rotation.

where $N_{\text{bits\_H}} = N_{\text{bits\_V}}$ is the number of bits needed to code the height of each box. Simplifying:

$$R \leq 3J + 2 + \sum_{j=1}^{j_{\text{stop}}} ((J - 2(j-1) + \lceil log_2 \beta \rceil) \cdot 3 + 2) \cdot 2^j.$$

This is an arithmetico-geometrical progression, whose sum, when J big enough, can be approximated by [9]:

$$R \sim 2^{\frac{1}{2}(J + \lceil log_2 \beta \rceil)}. \tag{23}$$

3) Rate-Distortion: Combining this equation with (20) we obtain the asymptotic R-D behavior:

$$D(R) \sim \left(\widetilde{\text{TV}} + 3 \right) \cdot 2^{-2 log_2 R} \sim \left(\widetilde{\text{TV}} + 3 \right) \cdot R^{-2}. \tag{24}$$

B. *Case $d_+ > 0$, $d_- = 0$ or viceversa*

This case turns to have the same R-D than the previous one, because the evolution of the rectangle height is leaded by the distance bigger than zero.

C. *Case $d_- = d_+ = 0$*

This case is very favorable to our coding scheme, because it means that with just one iteration the minimum distortion requirement is reached. The parallelogram height will be $H = 2^{-J}$, and the rate will consist in the bits needed to code the two vertices and the box height, $R = 3J$. This makes a R-D behavior coherent with the results obtained in [7]:

$$D(R) = 2^{-\frac{R}{3}}. \tag{25}$$

V. Adding refinement

The anisotropic quadtree with rotation has a good R-D decay, but it has the drawback that the reconstructed edge may loose its original continuity. The introduction of refinement solves this problem. In this case the procedure is exactly the same that has been explained in the previous section but when the minimum resolution has been achieved, a refinement is performed inside the last resolution rectangle by splitting the x axis into intervals of size 2^{-J}.

The effect of adding refinement in the anisotropic quadtree with rotations does not change the slope of the R-D decay, but it allows a better PSNR given a certain

rate, shifting the R-D line to the left (see Fig. 3). Following the procedure that has been adopted previously, the distortion found for the case $d_+ > 0$ and/or $d_- > 0$ is:

$$D \sim \widetilde{\mathrm{TV}} \cdot 2^{-2J} + 3 \cdot 2^{-J-M}. \tag{26}$$

The rate now has to take into account the number of refinements performed inside each parallelogram, the number of parallelograms to refine and the number of bits to perform the refinement. Including the refinement bits in (23) and taking $M = J$, we find:

$$R = 2^{\frac{1}{2}(J+\log_2 \beta)} + M \cdot 2^J. \tag{27}$$

From (26) and (27), it is easy to deduce the final R-D expression:

$$D(R) \sim \widetilde{\mathrm{TV}} \cdot \frac{\log_2 R}{R^2}. \tag{28}$$

The R-D found in the case where both distances are 0 (the edge is a straight line) is very similar to the one obtained in the case without refinement:

$$D(R) = 2^{-\frac{R}{2}}. \tag{29}$$

VI. Simulation results

Some comparisons among the presented methods and wavelets (JPEG2000 [10]) for a polygonal edge are illustrated here (see Fig. 3). This results show that the anisotropic quadtree with rotation gives better approximations than the other methods discussed here. The fact that the slope for the anisotropic quadtree with or without refinement is almost the same in the graph is probably because at such low bit-rates the log factor has no influence. Even though JPEG2000 is not adapted to black and white images, its R-D behaviour shows that the isotropic quadtree and wavelets have really the same R-D slope.

Fig. 4 represents the rate distortion decay of four different curves with increasing Total Variation. It shows that the practical results, obtained with the anisotropic quadtree with rotation, are coherent with the theoretical behavior found: the lower the TV, the better the R-D. From left to right, the graph represents the R-D of: a straight line (TV=0), a parabola with TV=0.51, a cubic curve with TV=0.75 and a parabola with TV=0.89.

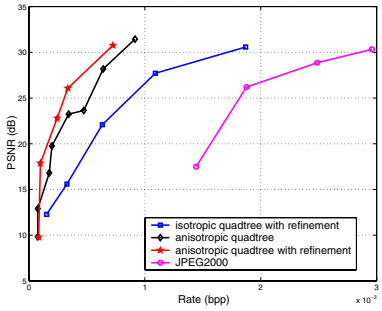


Fig. 3. Comparison among JPEG2000, isotropic quadtree with refinement and anisotropic quadtree with rotation for an image of 1024×1024 pixels.

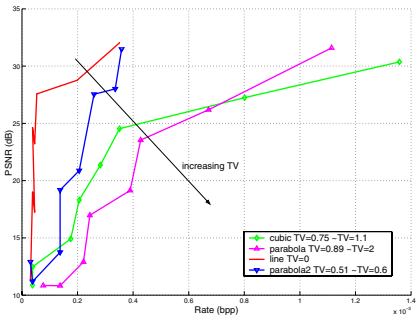


Fig. 4. Comparison of the R-D of different curves with different Total Variation.

VII. Conclusions

The inclusion of anisotropy and rotation in the quadtree improves the quality. The fact that an approximation of the TV appears in the R-D expression shows that geometrical complexity affects the capacity of compressing a given curve. As this anisotropic quadtree with rotation is a toy model for an adaptive non-linear image representation technique, it demonstrates that further research in adaptive image representation, taking into account the geometry, has to be done. Furthermore, the a/a^2 anisotropy that appears in [4] is also present here, mainly because we are using the second order Taylor expansion to approximate the edge. The theoretical results obtained here are coherent with existing ones, and the experimental data do not differ from the theoretical model.

An interesting development of this work will be to perform refinement inside the boxes introducing basis functions. This can allow us to extend the algorithm to natural images too. To insert the basis functions, the algorithm has to be slightly modified in order to have nested partitions.

References

[1] T. F. Chan and H. M. Zhou, "Adaptive ENO-wavelet transforms for discontinuous functions," in *12th International Conference on Domain Decomposition Methods*, T. Chan, T. Kako, H. Kawarada, and O. Pironneau, Eds., 2001, pp. 93–100.

[2] S. Mallat, *A wavelet tour of signal processing*, Academic Press, 1998.

[3] D. Donoho, "Wedgelets: Nearly-minimax estimation of edges," Tech. Rep., Dept. Of Statistics, Standford University, 1997.

[4] E. J. Candès and D. L. Donoho, "Curvelets - a surprisingly effective nonadaptive representation for objects with edges," in *Curve and Surface Fitting*, C.Rabut A.Cohen and L.L.Schmaker, Eds. Vanderbilt University Press., 1999.

[5] E. Candès and D. Donoho, "Ridgelets: a key to higher dimensional intermitency?," in *Phil Trans. R. Soc. Lond.*, 1999.

[6] Rosa M. Figueras i Ventura, Lorenzo Granai, and Pierre Vandergheynst, "A generalized rate-distortion limit for edge representation," Tech. Rep. 07.02, ITS-EPFL, http://ltspc4.epfl.ch/F/PAPERS/papers.shtml, May 2002.

[7] Mihn N. Do, Pier Luigi Dragotti, Rahul Shukla, and Martin Vetterli, "On the compression of two dimensional piecewise smooth functions," in *IEEE International Conference on Image Processing (ICIP)*, 2001.

[8] E. Le Pennec and S. Mallat, "Bandelet representation for image compression," in *International Conference on Image Processing. Proceedings*, 2001, vol. 1, pp. 12–15.

[9] I. S. Gradshteyn and Ryzhik I.M., *Table of Integrals, Series, and Products*, Academic Press, Inc., fifth edition, 1994.

[10] ISO/IEC 15444-1:2000, "Information technology - JPEG2000 image coding system," December 2000.