A generalized Rate-Distortion limit for edge representation

Rosa M. Figueras i Ventura, Lorenzo Granai and Pierre Vandergheynst

Abstract

This paper presents a rate-distortion analysis for a simple horizon edge image model. A quadtree with anisotropy and rotation is performed in 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 in the Rate-Distortion decay is also studied.

Keywords

Image coding, rate-distortion analysis, edge representation, wavelets, image analysis.

I. Introduction

It is known that the Rate-Distortion behavior of wavelets it 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 rate-distortion decay of $O\left(R^{-\frac{1}{2}}\right)$ [2]. This motivates a lot of research in order to find a more efficient way for coding images [3], [4], [5].

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

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

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 et al. [6] already demonstrated that the Rate-Distortion behavior of this model decays as R^{-1}. They introduced refinement in the isotropic quadtree technique to improve its distortion-rate behavior, 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. This anisotropic quadtree with rotations can be a good toy model for bandelets [7]. The fact of introducing anisotropy shows that the first derivative of the edge function appears in the rate-distortion expression. Furthermore, when rotation is also included in the scheme, rate-distortion is affected by the curvature of the edge, showing that the rate needed to represent a given contour is directly proportional to the geometrical complexity of this contour. The fact that geometrical complexity needs higher bit-rate to achieve a given distortion is coherent with empirical and previous theoretical results.

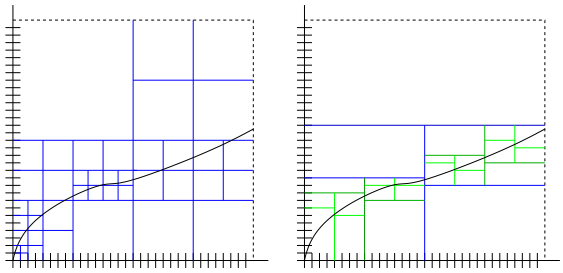
II. Optimal quadtree based compression

In [6], Do et al. showed that, in terms of rate-distortion, the optimal quadtree based compression gives better results than wavelets. This technique consists in employing a quadtree segmentation scheme and representing each sub-image by performing a rate distortion optimization.

The quadtree is based on a dyadic division of the unity interval $[0,1]^2$ (see Fig. 1(a)). At each scale, the algorithm will decide whether a given square is white, black, intermediate (there is an edge inside but the finest resolution has not been achieve) or edge. If it is a white or black square, it will not be divided any more. If it is an intermediate square, it will be further divided until the maximum number of iterations J has been

reached, when it will be labeled as edge square. The final division size will be $2^{-J} \times 2^{-J}$. Finally, when the edge square is achieved, a refinement (coding with a certain number of bits where the edge crosses the square edges) 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:

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



(a) Isotropic (b) Rectangular

Fig. 1. Example of isotropic and rectangular quadtree decomposition.

III. Anisotropic Quadtree

A. Algorithm

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 depend mainly on the edge. As can be seen in Fig. 1(b), the size of the partition height will be always chosen so that the superior or inferior limits of the partition are located as close to the edge as possible but without crossing it. After the first partition, the dyadic rectangle containing the edge will be divided into two along the x axis. The partitions will be refined and further divided. This process will be repeated 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.

B. R-D behavior

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

$$N_{a_j} \sim 2^j, \tag{3}$$

$$N_e \leq 2^{J+\lceil \log_2 y'_{max} \rceil}, \tag{4}$$

where J is the maximum number of decomposition levels allowed. As next iteration rectangle will be inside the previous iteration one, the number of bits needed to code the size of each rectangle will decrease with the iteration number as:

$$N_{\text{bits}} \leq J - (j-1) + \lceil \log_2 y'_{max} \rceil. \tag{5}$$

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:

$$R = (2 + N_{\text{bits}})N_a + 2M \cdot N_e, \tag{6}$$

where M is the number of bits used to code the position where the edge crosses the dyadic rectangle. 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 Eq. (3), (4) and (5) with Eq. (6) we obtain:

$$R \leq \sum_{j=0}^{J+\lceil \log_2 y'_{max} \rceil} [(2 + J - (j-1) + \lceil \log_2 y'_{max} \rceil) 2^j] + 2M \cdot 2^{J+\lceil \log_2 y'_{max} \rceil}. \quad (7)$$

By developing the sums as arithmetic series, previous equation turns to:

$$R \leq (2 + J + \lceil \log_2 y' \rceil)(2^{J+\lceil \log_2 y' \rceil + 1} - 1) - [2^{J+1+\lceil \log_2 y' \rceil}(J + \lceil \log_2 y' \rceil - 1) + 2] + 2^{J+1+\lceil \log_2 y' \rceil} M. \quad (8)$$

Simplifying, Eq. (8) gives:

$$R \leq 2(3 + M)2^{J+\lceil \log_2 y' \rceil} - J - \log_2 y' - 4, \quad (9)$$

and so:

$$R \leq 2(M + 3)2^J + \lceil \log_2 y' \rceil, \quad (10)$$

which, when considering $M{=}J$ and high bit-rate turns to:

$$R \sim J \cdot 2^{J+\lceil \log_2 y' \rceil}. \quad (11)$$

This result is equivalent to the quadtree result, but with a term $2^{\lceil \log_2 y' \rceil}$ which comes from the fact of using irregular y axis division. In terms of distortion, as the final resolution is 2^J (exactly as the regular quadtree) the situation basically is the same:

$$D(R, F) = \int_{[0,1]^2} (f - \hat{f})^2 \leq \max_{x_1 \in [0,1]} |y(x_1) - y_{J,K}(x_1)| \leq C 2^{-J-M}, \quad (12)$$

which again, when considering $M{=}J$ gives:

$$D(R, f) \leq C 2^{-2J}. \quad (13)$$

And from the previous expression and Eq. (11) we obtain:

$$D \sim \frac{2^{2\lceil \log_2 y' \rceil} \log^2 R}{R^2}. \quad (14)$$

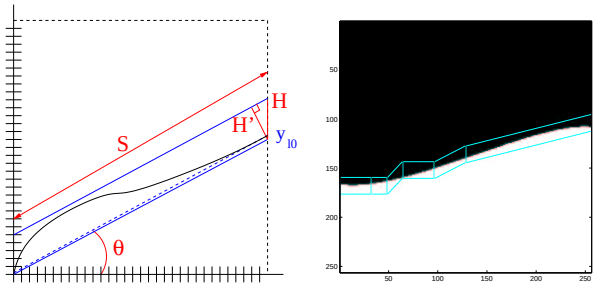
In the above expression we see that the rate-distortion of the irregular quadtree is similar to the one of the regular quadtree but with a factor $2^{2\lceil \log_2 y' \rceil}$ that comes from using anisotropy. The use of rectangles when dividing gives already a factor $2^{\lceil \log_2 y' \rceil}$ and 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 rate-distortion 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's take a 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. As in the anisotropic quadtree algorithm, the y axis partition will basically depend on the edge. Fig. 2(b) represents this edge analysis.

As in the previous case, it is imposed that the edge function $y(x) \in \mathbf{C^p}$, with $p \geq 2$ inside the unitary interval. 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 figure 2(a)):

$$D_j^k \leq S_j^k H_j'^k = S_j^k H_j^k cos\theta = 2^{-j} H_j^k. \quad (15)$$



(a) Scheme (b) Final resolution

Fig. 2. Anisotropic quadtree with rotation.

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 unitary interval:

$$y(x) = y\left(x_0\right) + y'(x_0)\left(x - x_0\right) + \tfrac{1}{2}y''(x_0)\left(x - x_0\right)^2 + O\left(\left(x - x_0\right)^3\right). \tag{16}$$

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 will be $(x_-,y(x_-))$ and $(x_+,\ y(x_+))$. Let y_L be the line that joins these two points:

$$y_L(x) = y(x_-) + \frac{y(x_+) - y(x_-)}{x_+ - x_-}(x - x_-). \tag{17}$$

The above expression does not take into account quantization. Considering that the vertices of the partitions must be on a dyadic grid, the real line will actually joint the points $(x_-,Q[y(x_-)])$ and $(x_+,\ Q[y(x_+)])$, where $Q[\cdot]$ stands for uniform quantization. Therefore, the new line is:

$$y_{LQ}(x) = Q[y(x_-)] + \frac{Q[y(x_+]) - Q[y(x_-)]}{x_+ - x_-}(x - x_-). \tag{18}$$

The parallelogram cannot cross the edge, therefore its superior and inferior distances to the line are bounded by:

$$\begin{aligned} d_+ &= \max\left\{0, \sup(y - y_{LQ})\right\} \geq 0 \\ d_- &= \max\left\{0, \sup(y_{LQ} - y)\right\} \geq 0. \end{aligned} \tag{19}$$

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

$$H = Q\left[d_+ + d_-\right]. \tag{20}$$

Three cases have to be considered: d_+ and d_- are both bigger than zero, one of them is equal to zero, and finally $d_+ = d_- = 0$. In the three cases, the distortion will be smaller or equal to the area of the parallelogram that contains the edge, as already shown in Eq. (15). 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 $d_- > 0$

A.1 Distortion

Let's first compute the distances d_+ and d_- in order to find out H and so the distortion. If x_{d_+} is the point in the x axis where $y_{LQ} - y$ is maximum, and approximating the curve by its second order Taylor expansion

in the middle point of the interval being analysed, we get:

$$\begin{array}{rcl} d_+ & = & y(x_{d_+}) - y_{LQ}(x_{d_+}) = \\ & = & y\left(\frac{x_+ + x_-}{2}\right) + y'\left(\frac{x_+ + x_-}{2}\right)\left(x_{d_+} - \frac{x_+ + x_-}{2}\right) + \\ & + & \frac{1}{2}y''\left(\frac{x_+ + x_-}{2}\right)\left(x_{d_+} - \frac{x_+ + x_-}{2}\right)^2 - \\ & - & y_{LQ}(x_-) - \frac{y_{LQ}(x_+) - y_{LQ}(x_-)}{x_+ - x_-}\left(x_{d_+} - x_-\right) + \\ & + & O\left(\left(x_{d_+} - \frac{x_+ + x_-}{2}\right)^3\right). \end{array} \tag{21}$$

The above expression, when substituting $y(x_-)$ by its Taylor expansion in the interval's middle point, turns to:

$$\begin{array}{rcl} d_+ & = & \left[y'\left(\frac{x_+ + x_-}{2}\right) - \frac{y(x_+) - y(x_-)}{x_+ - x_-}\right]\left(x_{d_+} - x_-\right) \pm 2^{-J}\left(\frac{x_{d+} - x_-}{x_+ - x_-}\right) + \\ & + & \frac{1}{2}y''\left(\frac{x_+ + x_-}{2}\right)\left[\left(x_{d_+} - \frac{x_+ + x_-}{2}\right)^2 - \left(\frac{x_+ - x_-}{2}\right)^2\right] \pm \\ & \pm & \frac{2^{-J}}{2} + O\left(\left(x_{d_+} - \frac{x_+ + x_-}{2}\right)^3\right) + O\left(\left(\frac{x_- - x_+}{2}\right)^3\right), \end{array} \tag{22}$$

It is possible to show that:

$$\left|y'\left(\frac{x_+ + x_-}{2}\right) - \frac{y(x_+) - y(x_-)}{x_+ - x_-}\right| \leq 2^{-2j} \tag{23}$$

and so

$$\left[y'\left(\frac{x_+ + x_-}{2}\right) - \frac{y(x_+) - y(x_-)}{x_+ - x_-}\right]\left(x_{d_+} - x_-\right) \sim O\left(2^{-3j}\right). \tag{24}$$

It is also easy to see that:

$$\left|\left(x_{d_+} - \frac{x_+ + x_-}{2}\right)^2 - \left(\frac{x_+ - x_-}{2}\right)^2\right| \leq 2^{-2(j+1)}, \tag{25}$$

which brings the following bound for the 3^{rd} term in Eq. (22):

$$\frac{1}{2}y''\left(\frac{x_+ + x_-}{2}\right)\left[\left(x_{d_+} - \frac{x_+ + x_-}{2}\right)^2 - \left(\frac{x_+ - x_-}{2}\right)^2\right] \leq \frac{1}{2}\left|y''\left(\frac{x_+ + x_-}{2}\right)\right| 2^{-2(j+1)}. \tag{26}$$

This expression is related to the second derivative computed in the middle point of each interval at each iteration. From now on, to simplify the notation, it will be referred to as K_j^k, where k refers to the label of the box and j to the iteration:

$$K_j^k = \left|y''\left(\left(k + \frac{1}{2}\right)2^{-j}\right)\right|, \tag{27}$$

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 considered to be a $\mathbf{C^2}$ curve, the set of K_j^k is bounded:

$$\beta = \max_{0 \leq k \leq 2^J - 1}\left|y''\left(\left(k + \frac{1}{2}\right)2^{-J}\right)\right| < \infty. \tag{28}$$

From equation (22) it follows that at iteration j and in box k the asymptotic behavior of d_+ is given by:

$$d_+ \sim \frac{1}{2}K_j^k 2^{-J - log_2\beta} + \frac{3}{2}2^{-J}. \tag{29}$$

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 an identical behavior.

The iterative algorithm is going to stop when the requested resolution is reached, i.e. when $H_j = 2^{-J}$. Substituting in (20), 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:

$$j_{\text{stop}} = max\left\{0, \frac{1}{2}\left(J + \log_2 \beta + 1\right)\right\}. \tag{30}$$

Now the parallelogram has width $2^{-j_{\text{stop}}}$ and height 2^{-J}. Notice that this makes the relation $\frac{\text{width}}{\text{height}}$ as:

$$\frac{\text{width}}{\text{height}} = \frac{2^{-j_{\text{stop}}}}{2^{-J}} = 2^{-\log_2 \beta}\frac{2^{-\frac{J}{2}}}{2^{-J}} \sim \frac{\text{width}}{\text{width}^2}, \tag{31}$$

giving the $\frac{a}{a^2}$ anisotropy present in curvelets [8].

The final distortion will be the distortion at the last resolution ($j = j_{\text{stop}}$), and so it takes the following form:

$$\begin{aligned} D &= \sum_{k=0}^{2^{j_{\text{stop}}}-1} \left(d_+ + d_-\right) 2^{-j_{\text{stop}}} = \\ &= \sum_{k=0}^{2^{j_{\text{stop}}}-1} \left(K^k_{j_{\text{stop}}} + 3\right) 2^{-3j_{\text{stop}}} = \\ &= 2^{-2j_{\text{stop}}}\left(\sum_{k=0}^{2^{j_{\text{stop}}}-1} K^k_{j_{\text{stop}}} 2^{-j_{\text{stop}}} + \sum_{k=0}^{2^{j_{\text{stop}}}-1} 3 \cdot 2^{-j_{\text{stop}}}\right). \end{aligned} \tag{32}$$

In the right-hand side of equation (32), 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:

$$\lim_{j\to\infty} \sum_{k=0}^{2^{j_{\text{stop}}}-1} K^k_{j_{\text{stop}}} \cdot 2^{-j_{\text{stop}}} = \int_0^1 \left|y''(x)\right| dx, \tag{33}$$

which can be seen as an approximation of the total variation TV of the edge (Riemann integral of the curvature), with the only difference that we have a sum of K^k_j instead of the integral of the curvature. Calling it $\widetilde{\text{TV}}$ the final expression for the distortion turns to:

$$D \sim \left(\widetilde{\text{TV}} + 3\right) \cdot 2^{-2j_{\text{stop}}} \sim \left(\widetilde{\text{TV}} + 3\right) \cdot 2^{-J - log_2 \beta}. \tag{34}$$

A.2 Rate

Each rotated box is coded by means of H_j and a left and a right vertex, so at the first iteration J bits are needed to code H_0 and J bits for the position of each vertex, which makes $3J$ bits. 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 depends on H_{j-1} as follows:

$$N^k_{\text{bits\_V}} = \log_2 \frac{H^{k'}_{j-1}}{2^{-J}} = J - 2(j-2) + \left\lceil \log_2\left(K^k_{j-1}\right)\right\rceil, \tag{35}$$

where k' is the index for the boxes of the previous iteration. As the number of bits needed to code all the boxes at iteration j is given by the bits for the two vertices and the height multiplied by the number of parallelograms, the total rate will be:

$$R = \sum_{j=0}^{j_{\text{stop}}} (2N_{\text{bits\_V}} + N_{\text{bits\_H}}) \times 2^j, \tag{36}$$

where $N_{\text{bits\_H}}$ is the number of bits needed to code the height of each box, which is the same as $N_{\text{bits\_V}}$. So the final rate turns out to be:

$$\begin{aligned} R &= 3J+2+\sum_{j=1}^{j_{\text{stop}}}\sum_{k=0}^{2^j-1}\left(\left(J-2(j-1)+\lceil log_2 K_{j-1}^k\rceil\right)\cdot 3+2\right)\leq \\ &\leq 3J+2+\sum_{j=1}^{j_{\text{stop}}}\sum_{k=0}^{2^j-1}((J-2(j-1)+\lceil log_2\beta\rceil)\cdot 3+2)= \\ &= 3J+2+\sum_{j=1}^{j_{\text{stop}}}((J-2(j-1)+\lceil log_2\beta\rceil)\cdot 3+2)\cdot 2^j. \end{aligned} \tag{37}$$

This is an arithmetico-geometrical progression, whose sum is [9]:

$$R \leq 28\cdot 2^{\frac{1}{2}(J+\lceil log_2\beta\rceil)} - 3J - 26 - 6\lceil log_2\beta\rceil. \tag{38}$$

Finally, for J big enough we find:

$$R \sim 2^{\frac{1}{2}(J+\lceil log_2\beta\rceil)}. \tag{39}$$

A.3 Rate-Distortion

Combining this equation with (34) we obtain the asymptotic Rate-Distortion behavior:

$$D(R) \sim \left(\widetilde{\text{TV}}+3\right)\cdot 2^{-2log_2 R} \tag{40}$$

and so:

$$D(R) \sim \left(\widetilde{\text{TV}}+3\right)\cdot R^{-2}. \tag{41}$$

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

This case turns to have the same distortion rate than the previous one, because the evolution of the distortion with the refinement does not change. Only some constants differ, but the general behavior is the same.

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. If $d_+ = d_- = 0$, the parallelogram height will be:

$$H = 2^{-J}, \tag{42}$$

and the rate will consist in the bits needed to code the two vertices and the box's height:

$$R = 3J. \tag{43}$$

This makes a Rate-Distortion behavior of:

$$D(R) = 2^{-\frac{R}{3}}, \tag{44}$$

which is coherent with the results obtained in [6].

V. Refinement in the anisotropic quadtree with rotation

The anisotropic quadtree with rotation has a good Rate-Distortion decay, but it has the drawback that the reconstructed edge may loose its original continuity. The introduction of refinement solves this problem. This refined version of the algorithm uses for iterations from 0 to j_{stop} the same approach than in the previous case. But with the difference that when the minimum resolution has been achieved, a refinement will be performed. This refinement will be done inside the last resolution rectangle splitting the x axis into intervals of size 2^{-J}.

The effect of adding refinement in the anisotropic quadtree with rotations does not improve the Rate-Distortion behavior, but it does improve the PSNR given a certain rate, as can be seen in the practical results section. Following the same procedure that has been taken all over the paper, the distortion found for the case $d_+ > 0$ and $d_- > 0$ is:

$$D = \widetilde{\mathrm{TV}} \cdot 2^{-2J} + 3 \cdot 2^{-J-M},$$

where J is, as usual, the number of bits allowed to code the vertices of the partitions and M are the number of refinement bits.

The rate now has to take into account the number of refinements performed inside each parallelogram (which is $\frac{2^{-j_{\mathrm{stop}}}}{2^{-J}} + 1$), the number of parallelograms to refine ($2^{j_{\mathrm{stop}}}$) and the number of bits to perform the refinement (M). Adding this to the rate needed to code the vertices of the parallelograms from Eq. (39) and taking $M = J$, we find:

$$R = 2^{\frac{1}{2}(J+\log_2 \beta)} + M \cdot 2^J. \tag{46}$$

And from (45) and (46) , it is easy to deduce the final Rate-distortion expression:

$$D(R) \sim \widetilde{\mathrm{TV}} \cdot \frac{\log_2 R}{R^2}. \tag{47}$$

As in the previous section, the case where $d_+ > 0$ or $d_- > 0$ does not change from the case where both distances are bigger than zero. For the case where both distances are 0 (the edge is a straight line), the rate-distortion found is very similar to the one found in the case without refinement:

$$D(R) = 2^{-\frac{R}{2}}. \tag{48}$$

VI. Practical results

Some comparisons among the anisotropic quadtree with rotation, the isotropic quadtree with refinement and wavelets (JPEG2000 [10]) are presented here (see Fig. 3). This results show that the anisotropic quadtree with rotation gives better approximations than any other method. 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. The JPEG2000 is far away to the right of the graph (it gives worse approximation), but this is due to the fact that it is adapted to natural images and not to black and white ones.

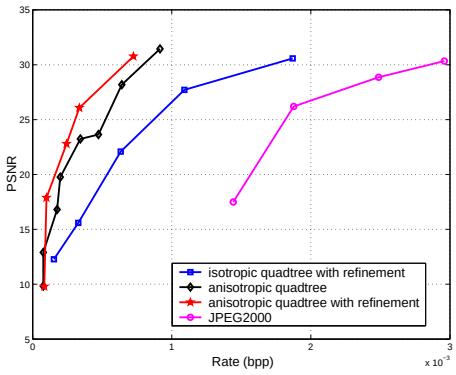


Fig. 3. Comparison among JPEG2000, isotropic quadtree with refinement and anisotropic quadtree with rotation for an image of 1024×1024 pixels.

Fig. 4 represents the rate distortion decay of four different curves with increasing Total Variation. It shows that the practical results are coherent with the theoretical behavior found: the lower the TV, the better the R-D. From left to right, the graph represents the rate-distortion 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.

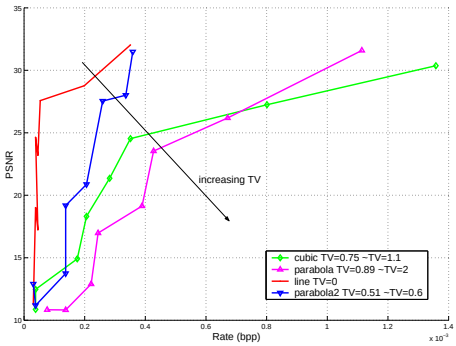


Fig. 4. Comparison of the bit-rate of different curves with different Total Variation.

VII. Conclusions

The inclusion of anisotropy and rotation in the quadtree gives an improvement of the approximation rate. The fact that an approximation of the TV appears at the R-D expression shows that geometrical complexity affects the capacity of compressing a given curve. As this anisotropic quadtree with rotations 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 [8] 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.

Acknowledgments

Many thanks to Diego Santa Cruz Ducci for his technical support.

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. Candès and D. Donoho, "Curvelets: A surprisingly effective nonadaptive representation of objects with edges," Tech. Rep., Department of Statistics, Standford University, 1999.

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

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

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

[8] 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., Saint-Malo, 1999.

[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.