Smoothlets — Multiscale Functions for Adaptive Representation of Images

Agnieszka Lisowska

Abstract

In this paper a special class of functions, called smoothlets, is presented. They are defined as a generalization of wedgelets and second order wedgelets. Unlike all known geometrical methods used in adaptive image approximation, smoothlets are continuous functions. They can adapt to location, size, rotation, curvature and smoothness of edges. The M-term approximation of smoothlets is $O(M^{-3})$. In this paper also an image compression scheme based on the smoothlet transform is presented. From the theoretical considerations and experiments, both described in the paper, it follows that smoothlets can assure better image compression than the other known adaptive geometrical methods, namely wedgelets and second order wedgelets.

Index Terms

smoothlets, wedgelets, multiresolution, adaptivity, approximation, compression.

EDICS: SMR-REP, COM-LOC.

Manuscript received April 21, 2010; revised August 12, 2010.

A. Lisowska is with the University of Silesia, Institute of Computer Science, ul. Bedzinska 39, 41-200 Sosnowiec, Poland, tel. +48 32 3689 746, e-mail: alisow@ux2.math.us.edu.pl .

I. INTRODUCTION

Finding an efficient representation of a digital image is the main issue in many image processing tasks. An efficient representation is understood as one which can capture the image content using as few basic functions as possible. Such an approach leads to a compact image representation which can be used for example in image compression or denoising. In fact, finding such a compact representation is not easy. So, new efficient representations are still needed.

Representations which have been proposed recently are multiresolution and geometrical, inspired from observations of neuropsychology [1]. Indeed, from the human visual system it is known [2] that the human eye is built to catch changes of location, scale and orientation. The most well-known wavelets are very efficient in catching changes of location and scale [3]. But they fail to catch changes of orientation, a consequence of their separable extension to two dimensions [4], [5], [6]. So, to efficiently catch changes of orientation many new, geometrical, representations have been proposed. They can be divided into two groups. The one is based on nonadaptive methods of computing, with the use of frames, like brushlets [7], ridgelets [8], curvelets [9], contourlets [10], and shearlets [11]. In the second group, the approximations are computed in an adaptive way. The majority of representations are based on dictionaries; examples include wedgelets [12], beamlets [13], second order wedgelets [14], [15], platelets [16], and surflets [17]. Recently, however, adaptive schemes based on bases have been proposed, like bandelets [18], grouplets [19], and tetrolets [20]. More and more "X-lets" have been defined.

All the cited families of functions are different but their approach to the approximation within a group is similar. One can note that these two groups are characterized by different properties. Generally speaking, the nonadaptive methods of approximation seem to be faster than the adaptive ones since no additional decision "how to choose the best function" is made. However, recent research in the field of computational complexity of the wedgelet transform [21], [22] has proved that also adaptive methods based on dictionaries can be run in real time. Additionally, adaptive methods of approximation are known to be more compact than the nonadaptive ones since only the best atoms from the dictionary are chosen adaptively during the approximation process. But the most important issue at this point is how to define the dictionary.

The dictionaries proposed so far, like wedgelets, second order wedgelets, platelets or surflets, were defined by discontinuous functions. They were designed to represent the step edges in efficient way. However, the majority of images contain both discontinuous and continuous edges (in other words, step edges and edges with a smooth transition between its two sides). Nonadaptive methods can naturally cope with this problem, but the dictionary-based adaptive methods proposed so far are not quite able to do so. The approximation of such a continuous edge is either not good enough (see Fig. 1 (b)) or contains too many basis functions (see Fig. 1 (c)). Of course, wedgelets were designed only for cartoon like images (e.g. images with only step discontinuities) [12] but in the great majority of papers related to wedgelets they are applied to still images in such areas of image processing as compression [15], [23], denoising [24], [25], image segmentation [26], [27] and edge detection [28]. This suggests that they can be used for images of different kinds. Moreover, some experiments of still image denoising [24], [25] show that adaptive methods (like wedgelets) assure better denoising results than nonadaptive ones (like curvelets) in spite of the drawbacks depicted above.

(a) (b) (c) (d)

Fig. 1. (a) The sample edge, (b) approximation by 1 wedgelet, (c) approximation by 22 wedgelets, (d) approximation by 1 smoothlet.

In this paper a new kind of function, called smoothlet, will be proposed for use in adaptive methods of geometrical approximation. The most important property of these functions is their continuity (only special cases of smoothlets are discontinuous). Thanks to this, smoothlets can adapt to both discontinuous and continuous edges (see Fig. 1 (d)). From neuropsychology it is known that the human eye is sensitive not only to location, scale and direction, but also to curvature. Additionally, it is well known that the human eye can catch changes of sharpness. All these aspects of image perception were used in the definition of smoothlets. The dictionary of smoothlets contains functions characterized by different locations, scales, directions, curvatures and smoothness. As follows from both the theoretical computations and the experiments performed, the proposed family of functions assures significantly more compact representation of an image than the adaptive methods known so far.

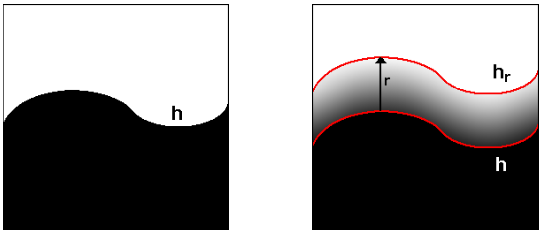
This paper is organized in the following way. In Section 2 the definition of smoothlets together with the theory related to the smoothlet transform is presented. In Section 3 the complete description of the smoothlet compression algorithm is presented, followed by theoretical computations of the Rate–Distortion dependency. Section 4 is devoted to experiments of image compression and their results. Section 5 summarizes this proposed approach to image representation.

II. SMOOTHLETS

Let us define an image domain $D = [0,1] \times [0,1]$. Next, let us denote a function $h(x)$ defined within D as the *horizon*, that is any smooth function defined on the interval $[0,1]$. Let us assume that $h \in C^\alpha$. Further, consider the characteristic function

$$H(x,y) = \mathbf{1}\{y \le h(x)\}\ , \qquad 0 \le x, y \le 1\,. \tag{1}$$

Then the function H is called the *horizon function* if h is a horizon. The function H models a black and white image with a horizon where the image is white above the horizon and black below (see Fig. 2 (a)). Let us define then a *blurred horizon* as the horizon function $H_B(x,y) : D \to [0,1]$ with a linear smooth transition (of length r) between black and white areas. An example of such a function is presented in Fig. 2 (b).



(a) (b)

Fig. 2. (a) Horizon function, (b) blurred horizon function (the blurring is parallel to the y-axis and does not depend on the horizon curvature).

A. *Definition of Smoothlet*

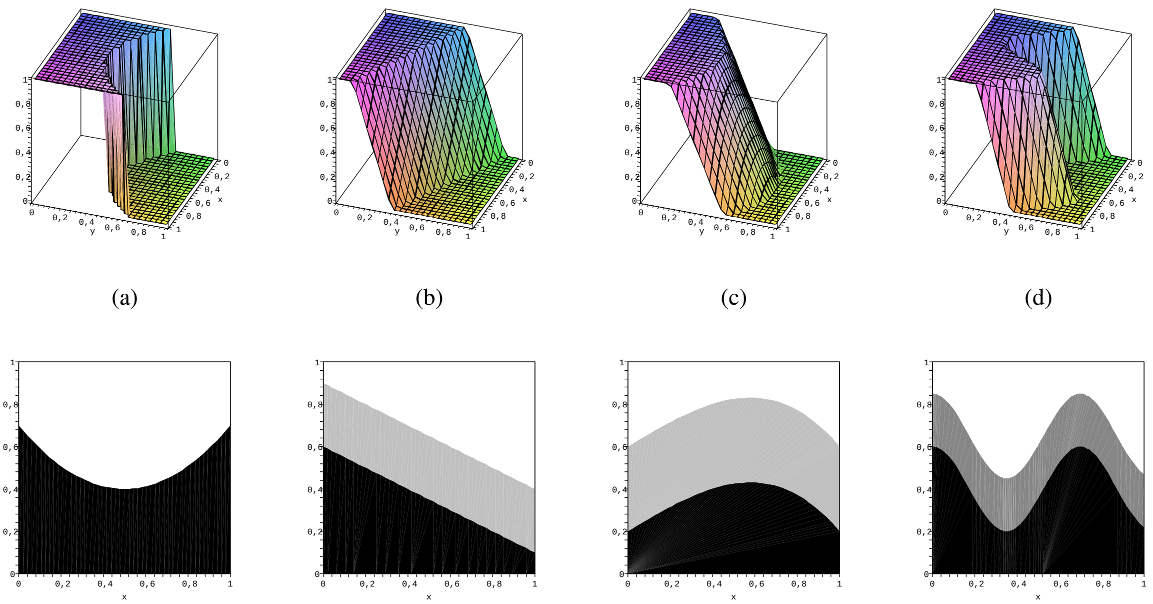
Consider any smooth function $b : [0,1] \to [0,1]$. Let us define the translation of $b(x)$ as $b_r(x) = b(x)+r$, for $r \in [0,1]$. Given these two functions, we can define an *extruded surface* represented by the following function

$$ext_{(b,r)}(x,y) = \frac{1}{r} b_r(x) - \frac{1}{r} y, \quad x, y \in [0,1],\ r \in (0,1]. \tag{2}$$

Strictly speaking, this function represents the surface which is the trace created by translating the function b in $\mathbb{R}^3$. Note that we can rewrite (2) in the following way:

$$r \cdot ext_{(b,r)}(x,y) = b_r(x) - y, \quad x, y \in [0,1],\ r \in [0,1]. \tag{3}$$

For $r = 0$ we obtain $b_r = b$ and $y = b(x)$. In that case the extruded surface is degenerate, it is a function b and is called a *degenerated extruded surface*.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 3. Examples of smoothlets: (a) $r = 0$, $b(x) = 1.2x^2 - 1.2x + 0.7$, (b) $r = 0.3$, $b(x) = -0.5x + 0.6$, (c) $r = 0.4$, $b(x) = -0.6x^3 + 0.6x + 0.2$, (d) $r = 0.25$, $b(x) = 0.2\cos(9x) + 0.4$; and (e)–(h) their corresponding projections, colors: black, white — areas of constant part, grey — area of linear part.

Let us then define a *smoothlet* as

$$S(x,y) = \begin{cases} 1, & \text{for } y \le b(x), \\ ext_{(b,r)}(x,y), & \text{for } b(x) < y \le b_r(x), \\ 0, & \text{for } y > b_r(x). \end{cases} \tag{4}$$

for $0 \le x, y \le 1$. In Fig. 3 the sample smoothlets for different functions b and different values of r, together with their projections on $\mathbb{R}^2$, are presented.

Let us examine the following specific examples of smoothlets.

Example 2.1: Assume that $r = 0$ and b is a linear function. We then obtain

$$S(x,y) = \begin{cases} 1, & \text{for } y \le b(x), \\ 0, & \text{for } y > b(x). \end{cases} \tag{5}$$

for $0 \le x, y \le 1$. It is a well known function called a *wedgelet* [12].

Example 2.2: Assume that $r = 0$ and b gives a segment of a parabola, ellipse or hyperbola. We then obtain $S(x,y)$ given by (5). It is a function called a *second order wedgelet* [14], [15]. An example is presented in Fig. 3 (a).

Example 2.3: Assume that $r = 0$ and b gives a segment of a polynomial. We then obtain $S(x, y)$ given by (5). It is a function called the two dimensional *surflet* [17].

Example 2.4: Assume that $r = 1$ and $b = 0$ is a special case of a linear function. We then obtain

$$S(x,y) = \begin{cases} ext_{(b,r)}(x,y), & \text{for } y \leq b_r(x), \\ 0, & \text{for } y > b_r(x). \end{cases} \tag{6}$$

for $0 \leq x, y \leq 1$. In this way we obtain a special case of a function called a *platelet* [16]. Indeed, in the definition of platelet in the place of the extruded surface in (6), any linear surface is possible.

B. Discrete Multiscale Smoothlets

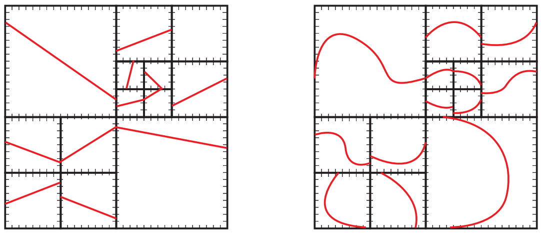
Consider an image domain $D = [0, 1] \times [0, 1]$. One can, in some sense, discretize it with different levels of multiresolution. Consider the dyadic square $D(i_1, i_2, j)$, as the two dimensional interval

$$D(i_1, i_2, j) = \left[i_1/2^j, (i_1 + 1)/2^j\right] \times \left[i_2/2^j, (i_2 + 1)/2^j\right], \tag{7}$$

where $0 \leq i_1, i_2 < 2^j$, $j \in \{0, \ldots, J\}$ and $i_1, i_2, J \in \mathbb{N}$. Note that $D(0, 0, 0)$ denotes the whole image domain D, that is the square $[0, 1] \times [0, 1]$. On the other hand $D(i_1, i_2, J)$ for $0 \leq i_1, i_2 < N$ denotes the appropriate pixels from $N \times N$ grid, where N is dyadic (it means that $N = 2^J$). From this moment on let us consider the domain of an image as such a $N \times N$ grid of pixels.

Having assumed that an image domain is the square $[0, 1] \times [0, 1]$ and that it consists of $N \times N$ pixels (or, more precisely, squares of size $1/N$) one can note that on each border of any square $D(i_1, i_2, j)$, $0 \leq i_1, i_2 < 2^j$, $0 \leq j \leq \log_2 N$, $i_1, i_2, j \in \mathbb{N}$, we may denote the vertices with distance equal to $1/N$. Every two such vertices in any fixed square may be connected by a segment of any, but fixed, function b. For example, when we assume that b is a linear function we obtain a linear segment, called a *beamlet* [13]. When we assume that b is a segment of any conic curve we obtain a *second order beamlet* [14], [15]. In general, when b is any function let us call b a *curvilinear beamlet*. So, taking into account curvilinear beamlets from all the above mentioned squares of different positions and scales, one obtains the dictionary of curvilinear beamlets. It consists of curvilinear segments of different positions, lengths, and orientations. The complexity of the dictionary for a fixed kind of beamlet is of order $O(N^2 \log_2 N)$ [12].

In Fig. 4, sample squares of different positions and scales together with their beamlets and curvilinear beamlets are presented. It is worth mentioning that since we deal with a discrete domain, the best way of drawing a curvilinear beamlet is to use the Bresenham algorithm [29].



(a) (b)

Fig. 4. Sample squares of different position and scale with their: (a) beamlets, (b) curvilinear beamlets.

In order to use curvilinear beamlets in image processing, there is a need to parameterize them in efficient way and define a dictionary of them. So, a *dictionary of curvilinear beamlets* can be defined in the following way:

$$B = \{b_{i,j,f} : i = 0, \ldots, 4^j - 1; j = 0, \ldots, \log_2 N; \\ \text{parameterization of } f\}, \tag{8}$$

where i denotes the position (the pair of subscripts i_1, i_2 can be replaced by the one subscript i such that $0 \leq i < 4^j$), j denotes the scale, and f denotes the kind of function together with its parameterization. It is important to mention that depending on the function used and the its kind of parameterization one may also need to take into account rotations. In general, depending on applications, the same class of functions can be parameterized differently. As an example, let us recall that beamlets are parameterized by two parameters θ, t which are the polar coordinates of the beamlet [23]. In the case of second order beamlets, three parameters are used: θ, t, d. The last one reflects the curvature of the second order beamlet [15].

Having defined the dictionary of curvilinear beamlets, we can define the *dictionary of smoothlets* in the following way:

$$\overline{S} = \{S_{i,j,b,r} : i = 0, \ldots, 4^j - 1; j = 0, \ldots, \log_2 N; \\ \text{parameterization of } b; r \in [0,1]\}, \tag{9}$$

where i denotes the position, j the scale, b the type of curvilinear beamlet together with its parameterization, and r denotes the length of the translation used in the definition of smoothlet given by (4).

C. Image Approximation by Smoothlets

In order to use smoothlets in image approximation there is a need to apply a grayscale smoothlet given by the formula

$$S_{(u,v)}(x,y) = \begin{cases} u, & \text{for } y \leq b(x), \\ ext_{(b,r)}^{(u,v)}(x,y), & \text{for } b(x) < y \leq b_r(x), \\ v, & \text{for } y > b_r(x). \end{cases} \tag{10}$$

for $0 \leq x, y \leq 1$ where

$$ext_{(b,r)}^{(u,v)}(x,y) = ext_{(b,r)}(x,y) \cdot (u-v) + v. \tag{11}$$

In the case of grayscale images $u, v \in \{0, \ldots, 255\}$. Let us note that $S_{(u,v)} = (u-v) \cdot S + v$.

Image approximation by smoothlets consists of two steps which are very similar to the wedgelets approximation [12]. The first one is the full smoothlet decomposition of an image with the help of the smoothlet dictionary. This means that for each square $D_{i,j}$, $0 \leq i < 4^j$, $0 \leq j \leq \log_2 N$, the best approximation in the MSE sense by a smoothlet is found. After the full decomposition (on all levels) the smoothlet coefficients are stored in the nodes of a quadtree. Then, in the second step, some kind of optimization algorithm, the bottom-up tree pruning algorithm [12], is applied to get a possibly minimal number of atoms in the approximation, ensuring the best image quality. Indeed, the following Lagrangian cost function is often minimized:

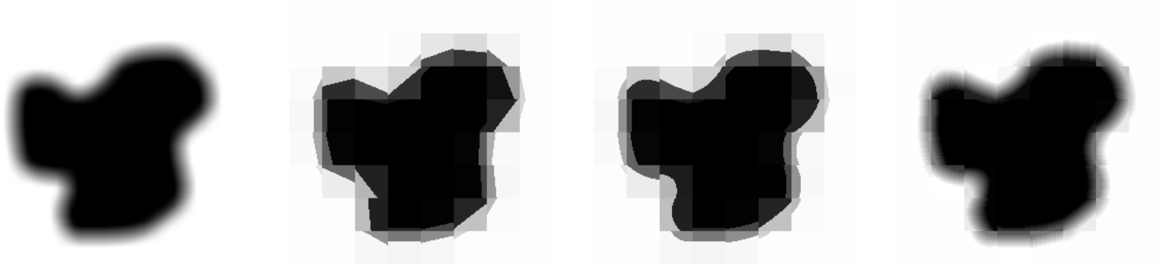
$$R_\lambda = min_{P \in QP} \{||F - F_S||_2^2 + \lambda^2 K\}, \tag{12}$$

where P is a homogenous partition of the image (elements of which are stored in the quadtree from the first step), F denotes the original image, F_S denotes its smoothlet approximation, K is the number of smoothlets used in the approximation and λ is the distortion rate parameter known as the Lagrangian multiplier. In case of an exact image approximation, the quality is determined and the reconstructed image is exactly like the original one.

In Fig 5 some examples of image approximation by smoothlets are presented. As one can see in case of the image with blurred edges the best approximation is given by smoothlets with $r > 0$, both visually and in the mean of Peak Signal-to-Noise Ratio given by formula $PSNR = 10 \log_{10}(255^2/MSE)$, where MSE denotes Mean Square Error.

III. Image Coding by Smoothlets

The smoothlets introduced in this paper can be used in efficient image processing, especially coding. Depending on the application, one can fix different kinds of function b. However, from the image coding



(a) (b) (c) (d)

Fig. 5. Examples of image approximation: (a) original image, (b) approximation by 64 wedgelets, $PSNR = 22.70dB$, (c) approximation by 64 second order wedgelets, $PSNR = 22.99dB$, (d) approximation by 64 smoothlets, $PSNR = 33.64dB$.

point of view, from the performed simulations it follows that the best choice for b is a conic curve. Such an approach assures the best Rate–Distortion (R–D) dependency. Indeed, functions of higher orders give better approximations than conic curves but the number of bits needed to code them is substantially larger, which thwarts the effectiveness of such coding. Additionally, we use in this paper paraboidal conic curves. From the simulations performed, it follows that the use of elliptical curves gives comparable results, both practically and theoretically.

A. Algorithm of Image Compression

There are many methods of image compression based on dictionary-based adaptive approaches [15], [23], [30], [31], [33]. Some of them are based on leaf coding, the others are based on scale dependency. The latter methods allow for progressive coding but the former ones are more applicable to smoothlets due to the fact that there is a poor dependency among r parameters in different scales in child–parent relation. A similar situation obtains in the case of the parameter d. So, the algorithm of image compression presented in the paper is based on leaf coding. It is worth mentioning that a method based on leaf coding can also lead to a progressive compression.

It is important to note that in the case of image compression by smoothlets a slightly modified version of (12) has to be used. In such a situation K denotes the number of bits needed to code a smoothlet instead of the number of smoothlets. And the algorithm finding the solution of the minimization problem is known as the generalized BFOS [32].

Approximation computing: It is a well known fact that the classical algorithm for wedgelet coding is time consuming. This follows from the step of the best matching wedgelet searching. Additional parameters d and r for curvature and smoothness, respectively, used in the definition of smoothlet, significantly lengthen the computation time. So, the proposed smoothlet approximation is performed in three steps:

1) Find the best wedgelet for a given square. This can be done quite efficiently according to [22].
2) Based on the best wedgelet, find the best smoothlet (with $d \neq 0$ and $r = 0$) in a fixed neighborhood of the wedgelet—try different ends of beamlets in proximity to the original one and different values of parameter d.
3) Based on the best smoothlet from the previous step find the best smoothlet (with $d \neq 0$ and $r \geq 0$) by trying different values of r—start from $r = 0$, as long as the approximation becomes better, increment r, otherwise stop.

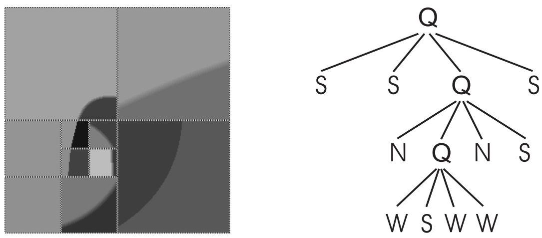
The proposed improvement of the best smoothlet computing enables the complete algorithm of smoothlet approximation to run in a reasonable time (less than 10 sec. for an image of size 256×256 pixels on an AMD Sempron 1.7 GHz processor). However, it should be pointed out that the obtained result is not necessarily the best one. Anyway, it is far better than can be assured by pure wedgelets.

Image coding: The coding scheme presented in this paper is similar to the ones presented in [15], [30]. The information is stored in a quadtree in the following way. In each node of the quadtree a node symbol is stored: Q—for further quadtree partitioning, N—for degenerated smoothlet (without any edge), W—for smoothlet with $d = 0$ and S—for the smoothlet with $d \neq 0$. Depending on the symbol the further information is stored (or not) in the appropriate node. So, the following convention is assumed:

- Q : no information,
- N : (N)(color),
- W : there are two cases:
  - when $r = 0$:
    (W)(number of beamlet)(color)(color)(0),
  - when $r > 0$:
    (W)(number of beamlet)(color)(color)(1)(r),
- S : there are two cases:
  - when $r = 0$:
    (S)(number of beamlet)(color)(color)(d)(0),
  - when $r > 0$:
    (S)(number of beamlet)(color)(color)(d)(1)(r).

An example of such a coding is presented below. In Fig. 6 there is shown a sample quadtree partition with smoothlets together with the appropriate quadtree with marked node symbols. In order to obtain the stream of data the quadtree is traversed in preorder mode. The final code related to it is as follows (the additional marks like commas, fullstops, etc., are used only for clarity):

Q, S:70457:162.63:8:1.2, S:63459:152.112:2:1.4, Q, N:147, Q, W:1679:21.145:0, S:4804:64.124:3:1.2, W:946:140.65:0, W:2307:64.187:1.2, N:144, S:13974:121.50:7:1.3, S:32794:62.88:18:0.



(a) (b)

Fig. 6. (a) Example of quadtree partition and (b) related quadtree applied in zig-zag mode.

In order to efficiently code the data from the quadtree to the bitstream it is sufficient to convert the decimal data to binary representation. The only issue to address is to fix how many bits we need to code a given parameter. The number of bits is additionally dependent on the square size. Indeed, for sufficiently small squares some simplification can be applied. In this paper the following conversion was made:

- For a square size larger than 2×2 pixels, the node symbols are translated as: Q – "00," N – "11," W – "01" and S – "10." When the square size equals 2×2 pixels it is sufficient to choose between degenerated wedgelet and wedgelet, so N – "1" and W – "0."
- The number of bits needed to code a beamlet is evaluated as $2j+3$ for a square of size $2^j \times 2^j$ pixels [30]. So, for a square size larger than 3×3 pixels, the number of bits needed to code a beamlet is $2j + 3$. When the square size equals 3×3 pixels it is sufficient to use 6 bits. And when the square size equals 2×2 pixels, it is 3 bits (only six possible beamlets are considered: two horizontal, two vertical and two diagonal ones).
- Color is stored using 8 bits.
- Parameter d is stored using j bits for a square of size $2^j \times 2^j$ pixels (this means $j - 1$ bits for the curvature and 1 bit for the sign). But it is possible only for the squares larger than 4×4 pixels.
- Parameter r is stored using $j - 1$ bits (applicable for squares larger than 2×2 pixels).

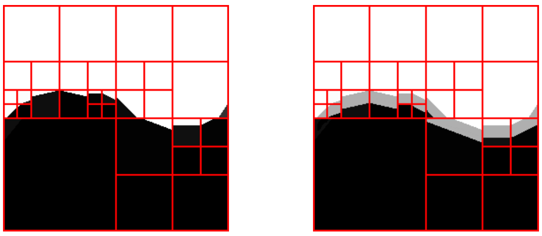
Theoretically, such bitstreams should be further compressed by an arithmetic coder in order to shorten the output. However, the proposed method of bitstream generation produces quite compact code. So, further stream processing is not necessary. On the other hand, an arithmetic coder does not give good results for short streams. From both these facts, it follows that the compression of the bitstream by an arithmetic coder produces a longer output than the input in the case of a short bitstream and insignificantly shorter output than the input in the case of longer bitstreams.

Postprocessing: It is a well known fact that during image compression based on a quadtree, blocking artifacts appear. The most commonly used method to remove this effect is to filter the obtained image along the block borders [34]. In this paper, a linear averaging method is used in order to improve the obtained results.

B. R-D Dependency

In the case of lossy coding the Rate–Distortion dependency is the best way of coding algorithm effectiveness evaluation.

Consider an image domain $D = [0,1] \times [0,1]$. As stated in Section II-B, it can be discretized with different levels of multiresolution. In this way we obtain $2^j \cdot 2^j$ elements of size $2^{-j} \times 2^{-j}$ for $j \in \{0,\ldots,J\}$. Consider then a horizon function defined on D. It can be approximated by a number of smoothlets, as presented in Fig. 7. In more detail, the edge presented in that image can be approximated by nearly 2^j elements of size $2^{-j} \times 2^{-j}$, $j \in \{0,\ldots,J\}$.



(a) (b)

Fig. 7. Example of horizon function approximation by smoothlets (a) $r = 0$, (b) $r > 0$.

Rate: According to the coding algorithm presented in the previous section, one needs the following number of bits to code a smoothlet:

- 2 bits for a node type coding and
- the following number of bits for smoothlet parameters coding:
  - 8 bits for degenerated smoothlet or

- $(2j+3)+16+1$ bits for smoothlet with $d=0$ and $r=0$ or
- $(2j+3)+16+j+1$ bits for smoothlet with $d>0$ and $r=0$ or
- $(2j+3)+16+j$ bits for smoothlet with $d=0$ and $r>0$ or
- $(2j+3)+16+j+j$ bits for smoothlet with $d>0$ and $r>0$.

So, the number R of bits needed to code a horizon function at scale j can be evaluated as follows:

$$R \leq 2^j \cdot 2 + 2^j((2j+3)+16+2j) \leq k2^j j. \tag{13}$$

Distortion: Consider a square of size $2^{-j} \times 2^{-j}$ containing an edge which is a C^α function. From the mean value theorem, it follows that it can be totally included between two straight lines with distance 2^{-2j} (see Fig. 8 (a)) [12]. So, the approximation distortion of edge h by straight line b can be evaluated as

$$\int_0^{2^{-j}} (b(x)-h(x))dx \leq k_1 2^{-j} 2^{-2j}. \tag{14}$$

Similarly, the edge can be totally included between two parabolas with distance 2^{-3j} (see Fig. 8 (b)). So, the approximation distortion of edge h by parabola b can be evaluated as

$$\int_0^{2^{-j}} (b(x)-h(x))dx \leq k_2 2^{-j} 2^{-3j}. \tag{15}$$

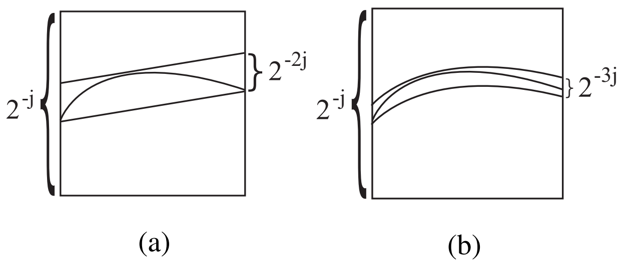


Fig. 8. (a) Distortion for beamlets, (b) distortion for second order beamlets.

Consider then a blurred horizon. The approximation distortion of blurred horizon H_B by smoothlet S can be computed as follows:

$$\int_0^{2^{-j}} \int_0^{2^{-j}} (S(x,y)-H_B(x,y))dydx = A+B+C, \tag{16}$$

where

$$A = \int_0^{2^{-j}} \int_0^{b(x)} (S(x,y)-H_B(x,y))dydx, \tag{17}$$

$$B = \int_0^{2^{-j}} \int_{b(x)}^{b_r(x)} (S(x,y)-H_B(x,y))dydx, \tag{18}$$

$$C = \int_0^{2^{-j}} \int_{b_r(x)}^{2^{-j}} (S(x,y) - H_B(x,y)) dy dx. \tag{19}$$

¿From the definition of S and H_B, (15), and direct computation, we obtain that

$$A \leq 2^{-3j}, \quad B \leq 2^{-j} 2^{-3j}, \quad C \leq 2^{-3j}. \tag{20}$$

So, the distortion of a blurred horizon approximation by a smoothlet is evaluated as follows:

$$||S - H_B|| \leq k 2^{-j} 2^{-3j}. \tag{21}$$

Taking into account the whole blurred edge defined on $[0,1] \times [0,1]$ and approximated by $k \cdot 2^j$ smoothlets, one obtains that the overall distortion D on level j is

$$D \leq k 2^{-3j}. \tag{22}$$

For comparison purposes let us note that applying, e.g., second order wedgelets (that is smoothlets with $r = 0$, called S_0) to a blurred horizon approximation, the distortion is

$$\int_0^{2^{-j}} \int_0^{2^{-j}} (S_0(x,y) - H_B(x,y)) dy dx \leq k 2^{-j} (2^{-3j} + \frac{r}{2}). \tag{23}$$

So, for the whole blurred edge the distortion D_0 on level j is

$$D_0 \leq k(2^{-3j} + \frac{r}{2}). \tag{24}$$

Since parameter r can be substantial, this distortion is larger than the one from the smoothlets. Maximally, r can equal 2^{-j}, so then $D_0 \leq k 2^{-2j}$.

Rate–Distortion: Rate–Distortion dependency relates the number of bits needed to code an image (parameter R) with distortion D. In the case of smoothlets the parameters R and D are evaluated by (13) and (22), respectively. So, we recall that

$$R \sim 2^j j, \quad D \sim 2^{-3j}. \tag{25}$$

By computing j from R and substituting it in D one obtains the following R–D dependency for smoothlet coding with constant k_s:

$$D(R) = k_s \frac{\log R}{R^3}. \tag{26}$$

For comparison purposes, recall that for wavelets $D(R) = k_v \frac{\log R}{R}$ [4] and for wedgelets $D(R) = k_w \frac{\log R}{R^2}$ [12].

M-term approximation: Another important parameter used in image approximation is the so-called M-term approximation. It is used in the case in which there is no need to code an image efficiently (e.g. image denoising). From the performed computations, it follows that each of 2^j elements of size $2^{-j} \times 2^{-j}$ generates distortion $k2^{-j}2^{-3j}$. So, a blurred horizon consisting of $M \sim 2^j$ elements generates a distortion of $D \sim 2^{-3j}$. It follows that $D \sim M^{-3}$. Repeating these considerations one can compute the M-term approximation for different horizon functions and for different smoothlets. These results are gathered in Table I.

TABLE I

M-TERM APPROXIMATION FOR DIFFERENT KINDS OF SMOOTHLETS.

Image	wedgelets	sec. ord. wedg.	smoothlets
horizon	$O(M^{-2})$	$O(M^{-3})$	$O(M^{-3})$
blurred horizon	$O(M^{-2} + \frac{r}{2})$	$O(M^{-3} + \frac{r}{2})$	$O(M^{-3})$

C. Computational Complexity

Adaptive methods of image coding are characterized by their computation time, which is dependent on the size of dictionary used. Because the dictionary of smoothlets is rather large, the expected computation time would seem to be substantial. But it can be reduced significantly by applying some tricks.

Let us recall that the computational complexity of the wedgelet transform of an image of size $N \times N$ pixels is $O(N^4 \log_2 N)$ [12]. However, it can be reduced by applying different improvements. One of them is based on efficient moment computation over polygonal domains by applying Green's theorem [22]. Its computational complexity is $O(N^3)$.

It is known also that the computational complexity of the second order wedgelet transform is the same as the wedgelet transform [15]. In the case of the smoothlet transform used in this paper, the only difference between that transform and the second order wedgelet transform is that there is a need to additionally test different values of parameter r for each partition square in order to compute the best smoothlet matching. Since for a square of size $N \times N$ pixels the maximal value of r is $N/2$, there is a need to compute $N/2$ smoothlet approximations. So, the overall computational complexity is $N/2 \cdot T$, where T is the computational complexity of the second order wedgelet transform. Because the best so far computational complexity for the wedgelet transform (and also for the second order wedgelet transform) is $O(N^3)$, the computational complexity of the smoothlet transform is $O(N^4)$.

IV. Experimental Results

In order to verify the theory presented in this paper, a number of numerical experiments have been performed. The smoothlets-based coder has been implemented in the C++ Builder environment. To prove the usefulness of the coder, the proposed method has been tested on the basis of benchmark still images. Some of them are presented in Fig. 9. These images were chosen so that some of them have rather sharp edges, some by edges with smooth transitions, and some by both kinds of edges.

Fig. 9. The tested images, namely: “Bird,” “Chromosome,” “Objects,” “Lena” and “Monarch.”

In Table II, the results of image coding by smoothlets are presented. For comparison purposes, the coding results by wedgelets and second order wedgelets are also shown. Because the images have been chosen with different kinds of edges, the results are also different. For images with a majority of sharp edges (“Monarch” and “Lena”) the average improvement is more or less 1.3%. For the image “Bird” it is 3%. The image “Objects” is characterized by smooth transitions across edges, so the improvement equals 5.5%. As one can expect, the best result is obtained for “Chromosome,” which has a strongly smooth character: it is over 10%. The average improvement for all presented images equals 3.8%. Of course it is pointless to apply the smoothlets to cartoon-like images which are characterized by sharp edges. But from the tests performed on still images, it follows that the smoothlets assure better coding results (both visually and in the mean of PSNR) than the other well known adaptive geometrical methods. It is worth mentioning here that it is difficult to compare the results to the ones from [31], [33] because there are neither available executable files nor source codes of those methods.

Additionally, in Fig. 10 there are presented the plots of the Rate–Quality dependency for rather non-smooth (“Lena”) and very smooth (“Chromosome”) images. In that way one can see the improvement. In Fig. 11 some more results are presented, namely images coded by smoothlets, wedgelets and JPEG2000

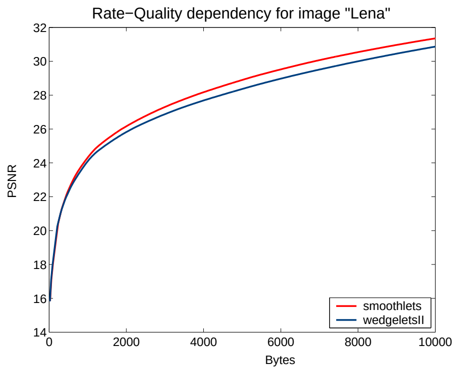
TABLE II

NUMERICAL RESULTS OF IMAGE CODING FOR DIFFERENT BITS PER PIXEL RATES (PSNR).

Image	bpp	Wedgelets	WedgeletsII	Smoothlets
Bird	0.2	31.253	31.280	32.360
	0.3	32.732	32.762	33.887
	0.4	33.889	33.913	34.939
	0.5	34.859	34.873	35.829
Chromosome	0.1	33.141	33.108	38.158
	0.2	36.048	35.969	40.487
	0.3	37.824	37.758	41.734
	0.4	39.230	39.194	42.590
Objects	0.3	29.288	29.311	31.329
	0.4	30.332	30.345	32.240
	0.5	31.248	31.252	32.990
	0.6	32.131	32.128	33.689
Lena	1.0	30.075	30.089	30.625
	1.1	30.442	30.455	30.970
	1.2	30.785	30.797	31.290
	1.3	31.105	31.116	31.577
Monarch	1.2	29.803	29.835	30.244
	1.3	30.221	30.245	30.663
	1.4	30.472	30.493	30.913
	1.5	30.822	30.840	31.251

[36]. As one can see, for the same file size, the smoothlets assure the best both PSNR and visual quality. However, it should be mentioned that for large bpp rates JPEG2000 assures better results than the smoothlets. So, the open problem to be solved is to either improve the smoothlet coder in high bpp rates or to build a connected coder which will use the smoothlets for low bpp and JPEG2000 for high bpp rates.

It should be mentioned that the results presented in the paper related to second order wedgelets are based on paraboidal wedgelets. It follows from the fact that such wedgelets are a special case of implemented paraboidal smoothlets (for $r = 0$), similar to wedgelets (for $r = 0$ and $d = 0$). So the overall scheme of image coding is exactly the same for all kinds of coding functions (namely wedgelets, second order wedgelets, and smoothlets). This assures that the improvements of image coding follows only from the nature of smoothlets and not from any tricks of coding. But, in order to compare the coding results with



(a)

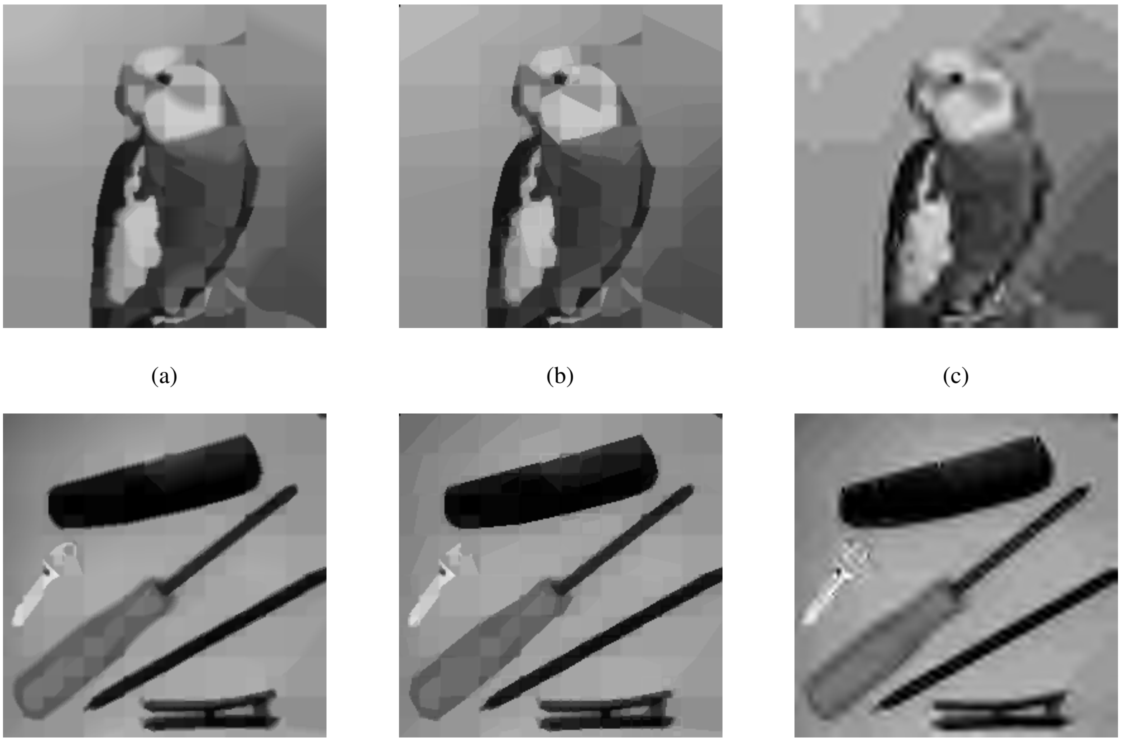
(b)

Fig. 10. Rate-Quality dependecy for image (a) “Lena,” (b) “Chromosome.”

the ones from the literature, in Fig. 12 there are shown the differences of coding between the paraboidal second order wedgelet coder applied in this paper and the elliptical second order wedgelet coder proposed in [15]. As one can see there is nearly no difference in efficiency between these two schemes. The plots were drawn for arbitrarily chosen images and the plots generated for the other tested images are similar.

It is important to note that the results of image coding presented in this section are based on the bitstream without a further compression step such as arithmetic coding [37]. This is due to two facts. The first one is that the produced code has a quite compact character, which means that further compression does not produce significant improvement. The second one is that the arithmetic coder does not support compression for short bitstreams (approximately up to 10kB). Indeed, for such bitstreams the output code is longer than the input one. To summarize—short bitstreams cannot be further compressed by an arithmetic coder and long bitstreams are compressed so little that it can be neglected (up to 1%). This tendency can be also seen in Table III, where some examples of arithmetic coding for arbitrary bitstreams of smoothlets are presented. It should be mentioned that it was used, the most promising, binary adaptive arithmetic coding. Based on these observations, none of the results presented in this paper were followed by arithmetic or any other additional coder.

The quadtree based image coders are characterized by so-called blocking artifacts. They cause visual image quality degradation. To overcome this problem, some postprocessing should be done. From the performed simulations it follows that the best choice of postprocessing for smoothlet-based coding is to average the pixel values lying on the borders. Unlike the averaging of true edge pixels which is adaptive (the parameter R can be different for different smoothlets) the averaging of the border pixels is fixed once. Assuming that the border is between pixels whose values are denoted by L and R, with $L \leq R$,



(a) (b) (c)

(d) (e) (f)

Fig. 11. Top: image "Bird" compressed by (a) smoothlets, size=545B, PSNR= 28.726 dB, (b) wedgelets, size=544B, PSNR= 27.674 dB, (c) JPEG2000, size=546B, PSNR=27.711 dB. Bottom: image "Objects" compressed by (d) smoothlets, size=904B, PSNR=28.576 dB, (e) wedgelets, size=904B, PSNR=26.436 dB, (f) JPEG2000, size=904B, PSNR=28.149 dB.

the new border pixel values are computed as $L_N = L + 1/3(R - L)$ and $R_N = R - 1/3(R - L)$. From the performed simulations it follows that smoothing wider than 2-pixels averaging does not give better results. ¿From the experiments it follows that applying postprocessing to the images coded by the smoothlets further improves the results by about 1%. In Fig. 13 there is presented such an example. The results presented in this paper are not postprocessed (except Fig. 11 (a),(d)) because they were compared to wedgelets which were presented in the literature without any postprocessing.

Another important part of an efficient coding algorithm is its computation time. According to the proposed way of the smoothlet transform computation, the algorithm consists of three steps: in each node of the quadtree the best wedgelet is found, given this wedgelet the best wedgelet with $d \neq 0$ is found, and based on this, the best smoothlet is found. In Table IV the computation times for different sizes of image "Lena" are presented for the smoothlet transform and for all consecutive steps. As one can see the additional searching (for d and r) takes about 29% of the overall computation time. The

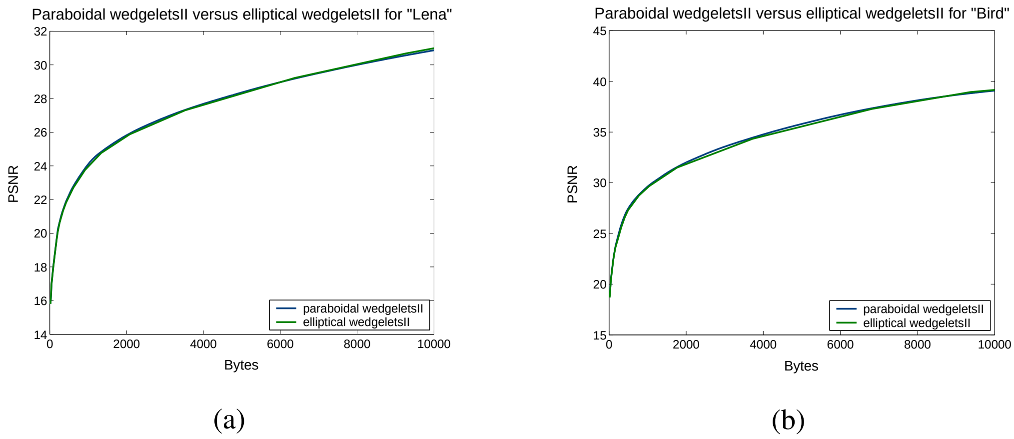


Fig. 12. Plots of Rate–Quality dependency for paraboidal (used in this paper) and elliptical (taken from [15]) second order wedgelets for image (a) "Lena," (b) "Bird."

TABLE III

SMOOTHLET BITSTREAM COMPRESSION BY THE BINARY ADAPTIVE ARITHMETIC CODING (+AC) FOR THE TESTED IMAGES (BYTES).

Image	Comp.	Length of the bitstream (bytes)					
Bird	stream	824	3300	7339	10172	13200	16539
	+AC	866	3326	7317	10116	13100	16373
Chrom.	stream	839	3317	7345	10197	13213	16627
	+AC	883	3350	7350	10184	13166	16502
Lena	stream	855	3303	7319	10183	13211	16570
	+AC	894	3340	7347	10199	13202	16513
Monarch	stream	832	3295	7340	10162	13210	16545
	+AC	874	3324	7362	10163	13191	16496
Objects	stream	822	3296	7345	10164	13205	16546
	+AC	859	3338	7363	10001	13075	16442

computations were performed on a Sempron 1.7 GHz processor unit. It should be mentioned that the best wedgelet is found according to the method that searches through every k-th beamlet and for the best one every beamlet from the neighborhood is checked. This is the simplest improvement of computation time. Some more improvements were proposed in [21], [22].

V. SUMMARY

(a) (b) (c) (d)

Fig. 13. Results of postprocessing for zoomed image "Bird," (a) no postprocessing, PSNR=29.837 dB, (b) 1-pixel averaging, PSNR=29.957 dB (c) 2-pixel averaging, PSNR=30.272 dB (d) 4-pixel averaging, PSNR=29.838 dB.

TABLE IV

COMPUTATION TIME (SEC.) OF THE SMOOTHLET TRANSFORM (ST) AND THE WEDGELET TRANSFORM (WT) PLUS BEST d SEARCHING PLUS BEST r SEARCHING.

Size	ST	WT	d	r
256	8.727	6.084	0.804	1.839
128	2.242	1.527	0.255	0.460
64	0.979	0.637	0.200	0.142
32	0.649	0.468	0.131	0.050
16	0.542	0.434	0.070	0.038

In this paper, a new family of adaptive geometrical functions, called smoothlets, has been presented. The concept of such a family of functions was based on the observation that the edges present in images are of different kinds of sharpness. So far, there was no adaptive geometrical method which was efficient in the approximation of such images. As is known, wavelets-based methods and non-adaptive geometrical multiresolution methods provide good representations only of smooth regions and edges with smooth transitions. On the other hand, adaptive geometrical multiresolution methods are good only for step discontinuities. Smoothlets are good at representating both kinds of edges.

The theory of smoothlets presented in this paper is quite general. Depending on the beamlet function used, one can obtain, for example, paraboidal smoothlets, elliptical smoothlets, polynomial smoothlets, sinusoidal smoothlets, etc. In the case of image compression the best choice seems to be the use of paraboidal smoothlets. This follows from the fact that it is easy to implement and a beamlet function of higher order leads to more parameters used in such a smoothlet representation. Such an approach will not assure good compression. But in the other image processing tasks, like for example denoising, there are no restrictions for the kind of a beamlet function. It can be fixed freely, even different kinds of functions can be used together in one image approximation.

The theoretical considerations and the performed experiments reported in this paper verified that smoothlets can give competitive results of approximation, in comparison to the other methods, leading to efficient compression of images, especially ones with smooth geometry. Additionally, since the proposed smoothlet transform is based on the wedgelet transform, recent research related to fast wedgelet transform assures that also the smoothlet transform can be computed in a reasonable time. This opens the doors for smoothlets to be used in real time applications.

REFERENCES

[1] G.W. Humphreys (ed.), *Case Studies in the Neuropsychology of Vision*, Psychology Press, UK, 1999.

[2] B.A. Olshausen and D.J. Field, "Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images," *Nature*, vol. 381, pp. 607–609, 1996.

[3] I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, 1992.

[4] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Academic Press, USA, 2009.

[5] M.J. Mohlenkamp and M.C. Pereyra, *Wavelets, Their Friends and What They Can Do for You*, European Mathematical Society, Germany, 2008.

[6] G.V. Welland (ed.), *Beyond Wavelets*, Academic Press, Netherlands, 2003.

[7] F.G. Meyer and R.R. Coifman, "Brushlets: A Tool for Directional Image Analysis and Image Compression," *Applied and Computational Harmonic Analysis*, vol. 4, pp. 147–187, 1997.

[8] E. Candès, *Ridgelets: Theory and Applications*, PhD Thesis, Departament of Statistics, Stanford University, Stanford, USA, 1998.

[9] E. Candès and D.L. Donoho, "Curvelets—A Surprisingly Effective Nonadaptive Representation for Objects with Edges," in: *Curves and Surface Fitting*, A. Cohen, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press, pp. 105–120, 1999.

[10] M.N. Do and M. Vetterli, "Contourlets," in: *Beyond Wavelets*, J. Stoeckler and G. Welland (eds.), Academic Press, San Diego, CA, pp. 83–105, 2003.

[11] D. Labate, W. Lim, G. Kutyniok and G. Weiss, "Sparse Multidimensional Representation Using Shearlets," *Proceedings of the SPIE*, vol. 5914, pp. 254–262, 2005.

[12] D.L. Donoho, "Wedgelets: Nearly-Minimax Estimation of Edges," *Annals of Statistics*, vol. 27, pp. 859–897, 1999.

[13] D.L. Donoho and X. Huo, "Beamlet Pyramids: A New Form of Multiresolution Analysis, Suited for Extracting Lines, Curves and Objects from Very Noisy Image Data," *Proceedings of SPIE*, vol. 4119, 2000.

[14] A. Lisowska, "Effective Coding of Images with the Use of Geometrical Wavelets," *Proceedings of the Decision Support Systems Conference*, Zakopane, Poland, (in Polish), 2003.

[15] A. Lisowska, *Geometrical Wavelets and their Generalizations in Digital Image Coding and Processing*, PhD Thesis, University of Silesia, Poland, 2005.

[16] R.M. Willet and R.D. Nowak, "Platelets: A Multiscale Approach for Recovering Edges and Surfaces in Photon Limited Medical Imaging," *IEEE Transactions on Medical Imaging*, vol. 22, pp. 332–350, 2003.

[17] V. Chandrasekaran, M.B. Wakin, D. Baron and R. Baraniuk, "Surflets: A Sparse Representation for Multidimensional Functions Containing Smooth Discontinuities," *IEEE International Symposium on Information Theory*, Chicago, 2004.

[18] E. Pennec and S. Mallat, "Sparse Geometric Image Representations with Bandelets," *IEEE Transactions on Image Processing*, vol. 14, no. 4, pp. 423–438, 2005.

[19] S. Mallat, "Geometrical Grouplets," *Applied and Computational Harmonic Analysis*, vol. 26, no. 2, pp. 161–180, 2009.

[20] J. Krommweh, "Image Approximation by Adaptive Tetrolet Transform," *International Conference on Sampling Theory and Applications*, Marseille, France, 2009.

[21] J. Romberg, M. Wakin and R. Baraniuk, "Multiscale Wedgelet Image Analysis: Fast Decompositions and Modeling," *IEEE International Conference on Image Processing*, vol. 3, pp. 585–588, 2002.

[22] F. Friedrich, L. Demaret, H. Führ and K. Wicker, "Efficient Moment Computation over Polygonal Domains with an Application to Rapid Wedgelet Approximation," *SIAM Journal on Scientific Computing*, vol. 29, no. 2, pp. 842–863, 2007.

[23] M. Wakin, J. Romberg, H. Choi and R. Baraniuk, "Image Compression Using an Efficient Edge Cartoon + Texture Model," *Data Compression Conference*, Snowbird, pp. 43–52, UT, USA, 2002.

[24] L. Demaret, F. Friedrich, H. Führ and T. Szygowski, "Multiscale Wedgelet Denoising Algorithms," *Proceedings of SPIE, Wavelets XI*, San Diego, CA, vol. 5914, 1-12, 2005.

[25] A. Lisowska, "Image Denoising with Second Order Wedgelets," *International Journal of Signal and Imaging Systems Engineering*, vol. 1, no. 2, pp. 90–98, 2008.

[26] R. Larsen, M.B. Stegmann, S. Darkner, S. Forchhammer, T.F. Cootes and B.K. Ersboll, "Texture Enhanced Appearance Models," *Computer Vision and Image Understanding*, vol. 106, pp. 20–30, 2007.

[27] S. Todorovic, M.C. Nechyba, "Detection of Artificial Structures in Natural-Scene Images Using Dynamic Trees," *Proceedings of the* 17^{th} *International Conference on Pattern Recognition*, vol. 1, pp. 35–39 2004.

[28] A. Lisowska, "Geometrical Multiscale Noise Resistant Method of Edge Detection," *Lecture Notes in Computer Science*, vol. 5112, pp. 182–191, Springer, Berlin, 2008.

[29] J.E. Bresenham, "Algorithm for Computer Control of a Digital Plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[30] X. Hou, J. Chen and D.L. Donoho, "JBEAM: Coding Lines and Curves via Digital Beamlets, *IEEE Preceedings of the Data Compression Conference*, Snowbird, USA, 2004.

[31] D. Alani, A. Averbuch and S. Dekel, "Image Coding With Geometric Wavelets," *IEEE Transactions on Image processing*, vol. 16, no. 1, 69–77, 2007.

[32] P.A. Chou, T. Lookabaugh and R.M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Transactions on Information Theory*, vol. 35, pp. 299–315, 1989.

[33] A.A. Kassim, W.S. Lee and D. Zonoobi, "Hierarchical Segmentation-Based Image Coding Using Hybrid Quad-Binary Trees," *IEEE Transactions on Image Processing*, vol. 18, no. 6, pp. 1284–1291, 2009.

[34] H. Reeve and J. Lim, "Reduction of Blocking Effects in Image Coding," *Optical Engineering*, vol. 23, pp. 34–37, 1984.

[35] G. Strang, *Calculus*, Wellesley-Cambridge Press, Wellesley, MA, 1991.

[36] C. Christopoulos, A. Skodras and T. Ebrahimi, "The JPEG2000 Still Image Coding System: An Overview," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 4, pp. 1103–1127, 2000.

[37] I.H. Witten, R.M. Neal and J.G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, vol. 30, no. 6, 1987.

Agnieszka Lisowska is a researcher-lecturer at the Institute of Computer Science, University of Silesia, where in 2006 she obtained her PhD with distinctions in the area of geometrical wavelets in Computer Science. In 2001 she completed her MSc in Mathematics at the same university. Her scientific research is concentrated on wavelets, geometrical multiresolution methods in image processing (including all kinds of ”X-lets”) and fractals. She collaborates with many scientists from all over the world. She is the author of several journal and conference papers published among others by Springer-Verlag or signed by IEEE.