J Math Imaging Vis (2011) 39: 180–192
DOI 10.1007/s10851-010-0233-3

Moments-Based Fast Wedgelet Transform

Agnieszka Lisowska

Published online: 28 October 2010

Abstract In the paper the moments-based fast wedgelet transform has been presented. In order to perform the classical wedgelet transform one searches the whole wedgelets' dictionary to find the best matching. Whereas in the proposed method the parameters of wedgelet are computed directly from an image basing on moments computation. Such parameters describe wedgelet reflecting the edge present in the image. However, such wedgelet is not necessarily the best one in the meaning of Mean Square Error. So, to overcome that drawback, the method which improves the matching result has also been proposed. It works in the way that the better matching one needs to obtain the longer time it takes. The proposed transform works in linear time with respect to the number of pixels of the full quadtree decomposition of an image. More precisely, for an image of size $N \times N$ pixels the time complexity of the proposed wedgelet transform is $O(N^2 \log_2 N)$.

Keywords Wedgelets · Moments · Fast wedgelet transform · Multiresolution

1 Introduction

From day to day image processing techniques become more and more efficient. Advanced multiresolution geometrical methods are seen nearly as a standard for modern algorithms of image processing such as denoising, segmentation, edge detection and compression. The most widely used wavelets theory has been replaced recently by geometrical wavelets. It follows from the fact that the methods based on them best reflect the way in which the human eye sees an image. Indeed, they properly catch changes of location, scale and orientation. And what follows, they properly reflect the geometry of an image. The theory of geometrical wavelets methods can be divided in two main groups. The first group is related to frames. In these techniques all the frame elements of different location, scale and orientation take part in an image approximation. The most known geometrical wavelets related to frames are ridgelets [2], curvelets [3], bandelets [16], etc. Whereas the second group is related to dictionaries like in the case of wedgelets [8], beamlets [9] or platelets [25]. Unlike in the previous group in the case of dictionary not all its elements take part in an approximation. The ones which do it are somehow adaptively chosen from the dictionary. Unfortunately, the "somehow" plays the most important role. Whereas frame based methods of approximation are relatively fast, the dictionary based methods are often slow. Too slow to build real time algorithms of image processing based on them.

Since the most important part of an image constitutes its geometry the techniques which can reflect it properly are widely used in many areas of image processing. In this case recently introduced wedgelets [8] are used with success in such tasks as compression [12, 13, 20, 23], denoising [7, 9, 12, 15], image segmentation [5, 22] or edge detection [14], to mention a few. Unfortunately, the computation time of wedgelet transform, which is used in all these applications, is rather slow. It causes that wedgelets-based methods may not be used in real time applications. Indeed, to compute wedgelet transform all wedgelet atoms from the dictionary have to be matched to each segment of quadtree partition of an image in order to choose the best one in MSE sense. Since the size of the dictionary is rather substantial and is dependent on the image size the computations take long time.

A. Lisowska (✉)
Institute of Informatics, University of Silesia, ul. Bedzinska 39, 41-200 Sosnowiec, Poland
e-mail: alisow@ux2.math.us.edu.pl

So, in order to improve the time complexity of the algorithm some modifications of it have been reported in the literature as, for example, top-down prediction among wedgelets [19] or the use of Green's theorem in wedgelet coefficients computation [10]. But, so far, there has not been any method which can work in a linear time.

In the paper the method of fast computation of the wedgelet transform has been proposed. It is based on moments computation what causes that wedgelet parameters are computed directly from the image instead of searching through the whole wedgelets' dictionary. In order to obtain wedgelet coefficients the method described in [18], which has rather theoretical character with poor practical application, has been improved. Additionally, since any wedgelet determined in such a way (reflecting edge presented in the image) is not necessarily the best one in the MSE sense, the additional improvement has been proposed in the paper. The improvement relates the time of computation with the quality of reconstructed image. It works in the way that the better quality one needs the longer the computation time should be. The theoretical proof and the experimental results, both presented in the paper, confirm that the proposed method of wedgelet transform is really fast, it works in linear time with respect to the size of the output data.

2 The Wedgelet Transform

At the beginning let us recall the theory related to the classical wedgelet transform. The majority of definitions presented in this section follows the work [8].

2.1 Dictionary of Wedgelets

Let us define an image domain $S = [0, 1] \times [0, 1]$. Next, let us denote function $h(x)$ defined within S as the "horizon", that is any continuous and smooth function defined on the interval $[0, 1]$. In practical applications it is sufficient to assume that the function h is of C^2 class.

Further, consider the characteristic function

$$H(x, y) = \mathbf{1}\{y \geq h(x)\}, \quad 0 \leq x, y \leq 1. \tag{1}$$

Then the function H is called the "horizon function" if h is "horizon". The function H models a black and white image with a horizon where the image is white above the horizon and black below.

Having an image domain $S = [0, 1] \times [0, 1]$ one can, in some sense, discretize it on different levels of multiresolution. Consider the dyadic square $S(j_1, j_2, i)$ as the two dimensional interval

$$S(j_1, j_2, i) = [j_1/2^i, (j_1+1)/2^i] \times [j_2/2^i, (j_2+1)/2^i], \tag{2}$$

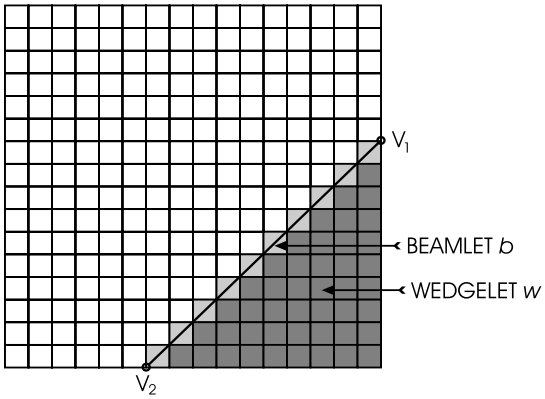


Fig. 1 Graphical representation of beamlet and wedgelet

where $0 \leq j_1, j_2 < 2^i$, $i \geq 0$ and $j_1, j_2, i \in \mathbb{N}$. Note that $S(0, 0, 0)$ denotes the whole image domain S, that is the square $[0, 1] \times [0, 1]$. On the other hand $S(j_1, j_2, I)$ for $0 \leq j_1, j_2 < N$ denote appropriate pixels from $N \times N$ grid, where N is dyadic (it means that $N = 2^I$). From this moment on let us consider a domain of image as such $N \times N$ 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 $S(j_1, j_2, i)$, $0 \leq j_1, j_2 < 2^i$, $0 \leq i \leq \log_2 N$, $j_1, j_2, i \in \mathbb{N}$, we may denote the vertices with distance equal to $1/N$. Every two such vertices in any fixed square may be connected to form a straight line—edge (also called *beamlet* after the work [9]).

Assume now that the considered edges are not degenerated, that is, they do not lie both at the border of the square. Then each such edge b splits any square S (we skip the subscripts denoting location and scale for a moment for better clarity) into two pieces. Let us consider one of the two pieces which is bounded by lines connecting in turn in clockwise direction, from the upper right corner, the first of the two edge vertices and then the second one. Let us define then the indicator function of that piece

$$W(x, y) = \mathbf{1}\{y \leq b(x)\}, \quad (x, y) \in S. \tag{3}$$

Such a function we call *wedgelet* defined by the beamlet b. The graphical representation of the wedgelet on S defined by the beamlet b is presented in Fig. 1.

It is obvious that on an arbitrary square S one can define many different wedgelets. Moreover, also the function which is the indicator function of the whole square S is taken as the wedgelet. So, more formally, one can define the set of wedgelets on any S as [8]

$$W(S) = \{\mathbf{1}(S)\} \cup \{\text{all possible } w \text{ defined on } S\}. \tag{4}$$

Additionally, within the whole image domain $S = [0, 1] \times [0, 1]$, the wedgelets are defined in different scales and lo-

cations (as stated in the case of beamlets). So, finally, one can define the wedgelets' dictionary W as the sum of all sets $W(S(j_1, j_2, i))$ of all dyadic squares $S(j_1, j_2, i)$, $0 \le j_1, j_2 < 2^i$, $0 \le i \le \log_2 N$, $j_1, j_2, i \in \mathbb{N}$.

Let us assume from now on that the pair of subscripts j_1, j_2 such that $0 \le j_1, j_2 < 2^i$ is replaced by the only subscript j such that $0 \le j < 4^i$. Such enumerations are equivalent, but the last one is more flexible. And let us denote $S(j_1, j_2, i)$ as $S_{i,j}$. Additionally, for the parameterization of direction (denoted so far by coordinates v_1, v_2) let us denote by m. Because in practical applications different parameterizations of direction are used such a general model seems to be more flexible. Indeed, the above assumptions allow to parameterize the wedgelets' dictionary using one parameter for scale i, one for location j and one for orientation m. More formally, we obtain the following definition.

Definition 1 The *Wedgelets' Dictionary* is defined as the following set:

$$\begin{aligned} W = \{&W_{i,j,m} : i = 0, \ldots, \log_2 N; \\ &j = 0, \ldots, 4^i - 1; \\ &m = 0, \ldots, M_W(S_{i,j}) - 1\}, \end{aligned} \tag{5}$$

where $M_W(S_{i,j})$ denotes the number of wedgelets on $S_{i,j}$.

In practical applications usually the parameterization based on polar coordinates is used. In such a case both beamlet and wedgelet are determined by angle θ between perpendicular to the beamlet and horizontal direction and by distance t from the beamlet to the center of the square. In grayscale images wedgelet function, additionally, is parameterized by two values h_1 and h_2 representing appropriate grayscale values.

Note that such a dictionary of wedgelets contains quite a large set of constant functions with discontinuities (seen as edges in images) along different locations, scales and orientations, which may be used in image representation. The total number of wedgelets for an image of size $N \times N$ is $O(N^2 \log_2 N)$ [8].

2.2 Wedgelet Transform

In adaptive methods of representation, the correlation of an image with all atoms of the dictionary must be assigned. In such a case the least squares projection is computed. So, the same occurs in the case of wedgelets' dictionary. Having defined such a dictionary and an image $F : S \to \mathbb{N}$, the Wedgelet Transform can be defined.

Definition 2 The *Wedgelet Transform (WT)* is defined by the following formula:

$$h_{i,j,m} = \frac{1}{T} \iint_S W_{i,j,m}(x, y) F(x, y)\, dxdy, \tag{6}$$

where

$$T = \iint_S W_{i,j,m}(x, y)\, dxdy \tag{7}$$

is the normalization factor and $S = [0, 1] \times [0, 1]$, $h_{i,j,m} \in \mathbb{R}$, $W_{i,j,m} \in W$, $0 \le i \le \log_2 N$, $0 \le j < 4^i$, $0 \le m < M(S_{i,j})$ and $i, j, m \in \mathbb{N}$.

From the practical point of view it means that the mean $h_{i,j,m}$ of all pixels (the values of which are denoted as $F(x, y)$) lying in the domain of the appropriate wedgelet $W_{i,j,m}$ is computed. In the discrete case the mean is taken from all pixels lying in the domain bounded by the border and the digital beamlet which is produced by Bresenham algorithm [1]. In the case of grayscale or colour images coefficients are additionally quantized to $h_{i,j,m} \in \{0, \ldots, 255\}$. Such coefficients denote mean grayscale intensities of the regions covered by appropriate wedgelets. In the case of binary images we use quantization such that $h_{i,j,m} \in \{0, 1\}$.

The wedgelet image representation is defined by the following formula:

$$F(x, y) = \sum_{i,j,m} h_{i,j,m} W_{i,j,m}(x, y). \tag{8}$$

But because W is a dictionary, not a basis, not all the wedgelets $W_{i,j,m}$ from the dictionary are used in the above representation (or in other words, some of the coefficients $h_{i,j,m}$ are equal to 0). The way in which they are chosen to represent F is described below.

2.3 Image Approximation

The base algorithm of image approximation with the use of WT is performed in two steps as presented in Fig. 2. The first one is the full wedgelet decomposition of an image with the help of WT. It means that for each square $S_{i,j}$, $0 \le j < 4^i$, $0 \le i \le \log_2 N$ the best approximation in MSE sense by wedgelet is found. It is done by searching the whole dictionary of wedgelets of scale i and by choosing the one with the smallest MSE. After the full decomposition (on all levels) all wedgelet coefficients are stored in the nodes of quadtree. Then, in the second step, some kind of optimization algorithm, the bottom-up tree pruning algorithm [8], is applied to get a possibly minimum number of atoms in approximation, ensuring the best image quality. Indeed, the following

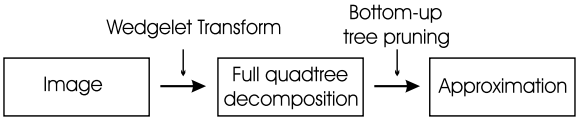


Fig. 2 The scheme of wedgelet image approximation

weighted sum is often minimized:

$$R_\lambda = \min_P \{\|F - F_W\|_2^2 + \lambda^2 K\}, \tag{9}$$

where P is homogeneous partition of an image (elements of which are stored in the quadtree from the first step), F denotes the original image, F_W its wedgelet approximation, K is the number of wedgelets needed to code the approximation and λ is the distortion rate parameter (also called penalization factor). In the case of exact image approximation the quality is determined and the reconstructed image is exactly like the original one. The two steps are described in more details in [8, 12].

2.4 Time Complexity

Because the algorithm of wedgelet approximation consists of two steps both of them should have relatively low time complexity in order to do fast approximation. Unfortunately, the most hard to compute is WT. Indeed, the tree pruning algorithm works in $O(N^2)$ time [8]. Whereas in the case of WT, since one must deal with $O(N^2 \log_2 N)$ wedgelet atoms to find best correlation with an image and every atom needs $O(N^2)$ computations, the following theorem is true.

Theorem 1 ([8]) *Consider an image of size $N \times N$ pixels. The time complexity of WT is $O(N^4 \log_2 N)$.*

So far, nobody proposed asymptotically linear method of wedgelet transform (with respect to the size of output data, that is $O(N^2 \log_2 N)$ since the number of result pixels is $N^2 \log_2 N$ for an image of size $N \times N$), though some attempts to improve the time complexity were undertaken.

Indeed, in [19] the method based on prediction was proposed. In this method in order to compute the parameters of wedgelets at coarse scales the parameters of wedgelets at finer scales are used. The time complexity of the algorithm is $O(MN^2)$ where M denotes the number of wedgelets for each dyadic square. But because $M = O(N^2)$ [8], the overall time complexity of the algorithm may be seen as $O(N^4)$. Additionally, it should be pointed out that the algorithm does not assure the best matching in the MSE sense for the predicted wedgelets.

Another improvement was proposed in [10]. The method is based on efficient moment computation over polygonal domains by applying the Green's theorem. The idea is based on the fact that in the discrete case integration of a function over two dimensional domain (needed in wedgelet coefficient computation) is replaced by integration of associated functions over the boundary of the domain. Since it causes reduction in dimension the overall algorithm of the wedgelet transform is fast and its time complexity is $O(N^2|\theta|)$, where $|\theta|$ denotes the number of angles used in computations. But in order to obtain results comparable with WT one needs that $|\theta| = O(N)$. So, the time complexity of the algorithm becomes $O(N^3)$.

In the next section we show that the time complexity of wedgelet transform can be further improved to its lower bound, that is $O(N^2 \log_2 N)$. The proposed method is based on moments computation.

3 The Fast Wedgelet Transform

Moments are widely used in wavelets theory, so it seems that introducing them to geometrical wavelets, especially wedgelets, should be also very attractive. Indeed, it is, as we show in this section. So, before we introduce the moments-based Fast Wedgelet Transform let us remind some facts about moments.

3.1 Moments

Moments are widely used in one dimensional wavelets construction in order to obtain the best possible properties of wavelet functions which are used then in function approximation [24]. In one dimensional case the moment is defined for any wavelet function ψ as

$$\int_s m(x)\psi(x)\,dx \tag{10}$$

where $m(x)$ is often a power function. Such moments are used in order to catch point discontinuities of a function.

However, in two dimensional images we deal with line discontinuities rather then point ones. So, to catch them properly, similarly as in the one dimensional case, moments may be used for two dimensional functions [11, 21]. In this case the moment is defined as

$$\iint_S M(x,y)W(x,y)\,dxdy. \tag{11}$$

Depending on the definition of function $M(x, y)$ one can construct different kinds of moments. For example the most commonly used ones are the Zernike [4], Tchebichef [17] or power moments. Additionally, in [18] a method of custom built moments construction has been proposed. Since the last method gives the best performance in approximation of functions such as wedgelets [18] it is used in the paper as the base method with the help of which the Fast Wedgelet Transform is defined.

3.2 The Moments-Based Wedgelet Transform

Consider any dyadic square S (we omit the subscripts i, j for a moment for better clarity) from a quadtree decomposition of any image. From the definition of wedgelet it follows that

the wedgelet approximation of the image $F : S \to \mathbb{N}$ which consists of two smooth areas with smooth step discontinuity between them, like in horizon function, is defined by the formula

$$F_W(x, y) = h_1 + (h_2 - h_1)W_m(x, y), \quad (x, y) \in S. \tag{12}$$

All we need to determine in order to compute WT from that simple example is the parameter m determining the beamlet position, which reflects the step edge present in the image F, and the wedgelets coefficients h_1 and h_2. As it can be parameterized in polar coordinates we use the pair (θ, t) instead of m. Such step edge may be detected in different ways. One of the most known method is the one based on the Radon Transform [6]. However, this method needs a rather large accumulator array and, what follows, is rather slow.

3.2.1 Beamlet Parameters Computation

To determine the parameters (θ, t) of the beamlet b approximating an edge on the image, as stated above, the following theorem is used.

Theorem 2 ([18]) *Let $K(x, y)$ be a continuously differentiable function, identically zero outside a bounded set S. Define*

$$A = \frac{\partial K}{\partial x}, \qquad B = \frac{\partial K}{\partial y}, \qquad C = \frac{\partial}{\partial x}(xK) + \frac{\partial}{\partial y}(yK)$$

and

$$\alpha = \iint_S A(h_1 + (h_2 - h_1)W_{\theta,t})\,dxdy,$$

$$\beta = \iint_S B(h_1 + (h_2 - h_1)W_{\theta,t})\,dxdy,$$

$$\gamma = \iint_S C(h_1 + (h_2 - h_1)W_{\theta,t})\,dxdy.$$

Then, all $(x, y) \in b(\theta, t)$ satisfy the equation

$$\alpha x + \beta y = \gamma.$$

To perform the practical computations of wedgelet parameters from S, it has been proposed to use the following function K [18]

$$K(x, y) = \begin{cases} (1 - x^2)(1 - y^2), & \text{if } (x, y) \in [-1, 1]^2, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

As shown in [18] this function assures the best approximation properties in comparison to the other known methods such as power or Zernike moments. So, in this paper also this function is used.

From the above theorem follows that we can compute the parameters of beamlet b which represents edge in the step function F_W given by formula (12) with the help of the moments α, β, γ. Since our approach is parameterization kind independent it is the fastest way to determine the beamlet and the related two wedgelets which approximate square segment S of the image. Indeed, one can simple transform the Cartesian parameterization to the polar one according to the formulae

$$\theta = \arctan\frac{\beta}{\alpha}, \qquad t = \frac{\gamma}{\sqrt{\alpha^2 + \beta^2}}. \tag{14}$$

Consider now any image $F : S \to \mathbb{N}$ with edge discontinuity. Because we do not know a priori the wedgelet function parameters h_1, h_2 in such a case we use the following formulae to determine α, β, γ instead of the ones from theorem 2:

$$\begin{aligned} \alpha &= \iint_S AF\,dxdy, \\ \beta &= \iint_S BF\,dxdy, \qquad \gamma = \iint_S CF\,dxdy. \end{aligned} \tag{15}$$

For images with well defined edges the computed parameters exactly reflect the edges which may be represented by appropriate beamlets. But in order to approximate the image by wedgelets one needs to compute additionally appropriate wedgelet coefficients (denoted as h_1, h_2).

3.2.2 Wedgelet Coefficients Computation

In order to compute grayscale intensities of wedgelet W and its complement W' within S, having the parameters of the beamlet, it has been proposed to use the following computations [18]

$$k = \iint_S (h_1 + (h_2 - h_1)W_{\theta,t})K\,dxdy, \tag{16}$$

$$U(a, b, c) = \iint_S W_{\theta,t}K\,dxdy, \tag{17}$$

$$\begin{aligned} h_2 - h_1 &= \left(\frac{\partial}{\partial c}U(a, b, c)\right)^{-1}, \\ h_1 &= \frac{k - (h_2 - h_1)\,U(a, b, c)}{\iint_S K\,dxdy}. \end{aligned} \tag{18}$$

However, these formulae, as computed directly according to Theorem 2, are useful only in the case of step functions given by formula (12). In real applications when we deal with any image F such approach does not give satisfactory results. Indeed, such computed values of parameters h_1, h_2 often do not reflect the real values of wedgelet coefficients represented by formula (6). So, to overcome that difficulty we propose to use the following formulae based on defini-

tion 2 instead of the above ones:

$$h_1 = \frac{\iint_S W'_{\theta,t} F\,dxdy}{\iint_S W'_{\theta,t}\,dxdy}, \qquad h_2 = \frac{\iint_S W_{\theta,t} F\,dxdy}{\iint_S W_{\theta,t}\,dxdy}. \tag{19}$$

This direct method is both simpler in implementation and faster than the one represented by formula (18). Additionally, what follows from experiments performed on a number of real images, approximations computed with the proposed modification give smaller MSE than the ones computed with the method proposed originally in [18].

3.2.3 The Moments-Based Fast Wedgelet Transform

In order to perform the moments-based Fast Wedgelet Transform we propose to proceed with the above procedure of wedgelet approximation (based mainly on formulae (15) and (19)) for all elements of quadtree partition of the image. More precisely, one has to determine the best wedgelet to every square of the quadtree image partition, that is to all the squares $S_{i,j}$ for $i = 0, \ldots, \log_2 N$, $j = 0, \ldots, 4^i - 1$. However, unlike in the classical WT where we must search within the whole dictionary of wedgelets in the moments-based version of wedgelet transform the parameters of wedgelets are computed directly from the image content. The main algorithm is presented below.

Algorithm 3.1: Fast Wedgelet Transform

1. **for** $i = 0$ to $\log_2 N$
2. **for** $j = 0$ to $4^i - 1$
3. compute α, β, γ according to formula (15);
4. compute θ, t according to formula (14);
5. compute h_1, h_2 according to formula (19);

From the above discussion one can conclude the following theorem.

Theorem 3 *Consider an image of size $N \times N$ pixels. The time complexity of the Fast Wedgelet Transform (FWT) is $O(N^2 \log_2 N)$.*

Proof Consider any square $S_{i,j}$, $0 \le i \le \log_2 N$, $0 \le j < 4^i$ from the quadtree partition of the image of size $N \times N$. The size of the square is $2^{n-i} \times 2^{n-i}$ pixels, where $n = \log_2 N$. For each such square in order to compute the beamlet parameters one needs three integration operations according to formula (15). Then, in order to compute wedgelet coefficients one needs two integration operations according to formula (19). Since the integration operation is performed in discrete domain and it denotes simple addition, the process is linear according to the number of pixels from the domain. Additionally, from the definition of quadtree partition follows that there are $2^i \cdot 2^i$ squares of size $2^{n-i} \times 2^{n-i}$ pixels. So, in order to integrate all squares from one level of decomposition one needs $(2^i \cdot 2^i) \cdot (2^{n-i} \cdot 2^{n-i})$ dominant operations. Because integration must be performed on all levels of decomposition one obtains the total number of integration operations as

$$\sum_{i=0}^{\log_2 N} (2^i \cdot 2^i) \cdot (2^{n-i} \cdot 2^{n-i}) = \sum_{i=0}^{\log_2 N} 2^n \cdot 2^n = N^2(1 + \log_2 N).$$

Since the integration is the dominant process in the algorithm we can conclude that its time complexity is $O(N^2 \log_2 N)$. □

As a direct result from the above considerations we obtain the following proposition.

Proposition 1 *Consider an image of size $N \times N$ pixels. Since the full quadtree decomposition of the image consists of $M = N^2(1 + \log_2 N)$ pixels the time complexity of Fast Wedgelet Transform is $O(M)$.*

The proposition states that the moments-based FWT works in linear time with respect to the number of pixels of the full quadtree decomposition. Moreover, since every pixel from this decomposition has to be computed, lesser time complexity of the algorithm is not possible. Whereas probably one can improve the speed of the proposed algorithm the time complexity remains still the same. That is the reason why the above proposed algorithm of decomposition has obtained the name Fast Wedgelet Transform (FWT). It is asymptotically the fastest algorithm of the all possible ones.

3.2.4 Practical Application

However, it should be pointed out the following problem related to FWT. Since one assumes that WT gives the best approximation of an image in MSE sense (and best edge adaptation as well), FWT not necessarily gives the best possible matching. An example of such a situation in the simple case of bird's wing fragment is presented in Fig. 3. As WT detected the edge properly (the white line), FWT detected it only nearby. It follows from the fact that we use the real image instead of ideal step function for which case the beamlet parameters were derived. It causes that the MSE of wedgelet matching with the help of FWT is not the best possible. On the other hand one can note that not necessarily MSE metric must be used in the approximation. It may be any other one. Indeed, it is true, wedgelet transform may be performed in any error metric because we are interested in

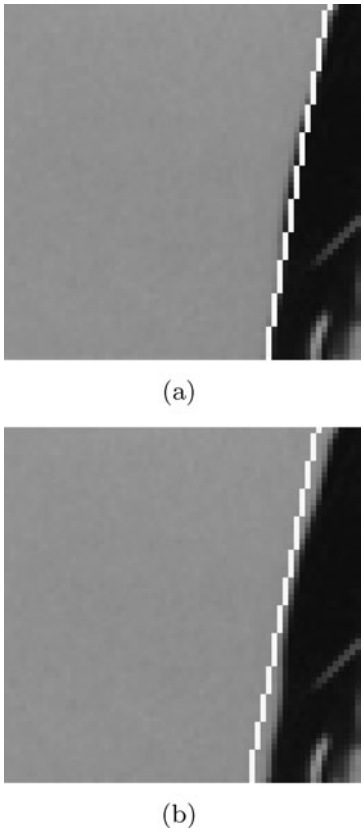


Fig. 3 Edge detection with the use of (**a**) WT, (**b**) FWT

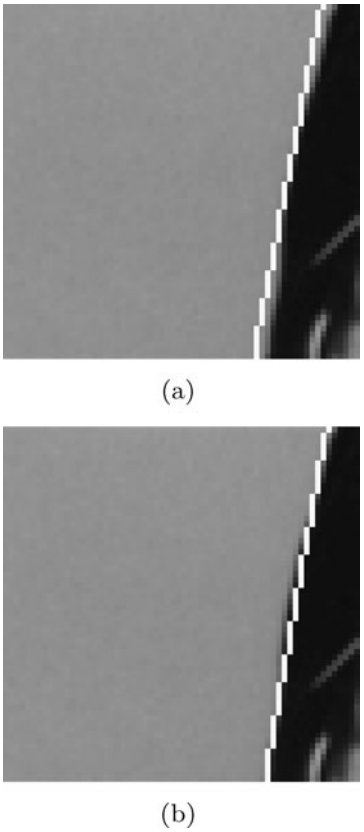


Fig. 4 Edge detection with the use of (**a**) FWT + 1, (**b**) FWT + 4

the best edge adaptation, not necessarily the best MSE adaptation. However, the example evidently shows that the edge was not matched properly. So, in the next section we show how the moments-based FWT may be improved.

3.3 From FWT to WT

From Fig. 3 and from a number of experiments performed follows that wedgelets determined by moments-based FWT are worse only a little than MSE best wedgelets. It means that such detected edges in great majority are situated in direct neighborhood of the proper edges. This fact may be used then to construct an improvement of the method.

Consider the beamlet parameterization as proposed originally in [8]. So, any beamlet may be parameterized by a pair of vertices (v_1, v_2). These vertices reflect the numbers of border pixels because in this model all border pixels are numbered consecutively starting from the upper right corner and proceeding in clockwise direction. Then, because we know that wedgelets determined by moments-based FWT are situated in a neighborhood of the original edge, we can try all wedgelets based on the neighboring beamlets with coordinates $(v_1 + k, v_2 + l)$ for $k, l \in \{-1, 0, 1\}$ and choose the one which gives better approximation in the MSE sense. If we want further to improve the precision of wedgelet matching we can try also all wedgelets based on beamlets with coordinates $(v_1 + k, v_2 + l)$ for $k, l \in \{-2, -1, 0, 1, 2\}$, etc. In general case we can check all wedgelets based on beamlets with coordinates $(v_1 + k, v_2 + l)$ for $k, l \in \{-R, \ldots, -1, 0, 1, \ldots, R\}$. Let us name the improved FWT with in range searching as FWT + R. Then in Fig. 4 one can see the result of FWT + 1 and FWT + 4. From the figure one can easily see that FWT + 1 gives better result of matching then FWT (see also Fig. 3(b)). Moreover, FWT + 4 gives exactly the same result as WT.

Note, additionally, that though the computation takes longer, the asymptotical time complexity still remains the same due to the fact that we perform additional computations a constant number of times. However, we can lengthen the computation time so much that we really compute classical WT. Indeed, the following theorem is true.

Theorem 4 *Consider an image of size $N \times N$ pixels and the time complexity of FWT as $O(N^2 \log_2 N)$. And let denote R as the range of best wedgelet searching for FWT + R. Then by tending with the range R to $3N - 5$ one obtains WT with time complexity $O(N^4 \log_2 N)$.*

Proof Note that for square of size $N \times N$ pixels and for arbitrary beamlet (v_1, v_2) from that square the possible max-

imal R equals $3N-5$ for v_i, $i=1,2$, if v_i is not situated at any corner of the square and it equals $2N-1$ if v_i lies at any corner. So, the overall maximal $R=3N-5$. For larger values of R the computations begin to repeat for the same beamlets. Similarly as in the proof of theorem 2 the dominant operation for a square $S_{i,j}$, $0\le i\le \log_2 N$, $0\le j<4^i$, is integration. Consider then the range R. For such range one needs to perform $(2R+1)^2$ integrations with the help of formula (19) since the range length is $2R+1$ and we must check beamlet connections for each pair (v_1,v_2), where $v_1,v_2\in\{-R,\ldots,0,\ldots,R\}$. So, note that if R tends to $3N-5$ the number of integrations for any square tends to

$$(2R+1)^2=(2(3N-5)+1)^2=(6N-9)^2.$$

So, for the all squares from the quadtree decomposition the number of dominant operations may be computed as follows

$$\begin{aligned}&\sum_{i=0}^{\log_2 N}(2^i\cdot 2^i)\cdot(2^{n-i}\cdot 2^{n-i})\cdot((6N-9)^2)\\&\quad=\sum_{i=0}^{\log_2 N}N^2(36N^2-108N+81)\\&\quad=(36N^4-108N^3+81N^2)(1+\log_2 N).\end{aligned}$$

From the above and from the fact that integration is dominant operation follows that the overall time complexity is $O(N^4\log_2 N)$. Finally, note that setting R as the maximal, $R=3N-5$, denotes checking the all possible beamlet connections and determining appropriate wedgelets. It is a way of working of the classical WT, what completes the proof. □

From the above theorem and from the definition of FWT + R follows directly the proposition.

Proposition 2 *Consider an image F of size $N\times N$ pixels. Denote F_{FWT+R} and F_{WT} as approximation of F by FWT+R and WT, respectively, for a fixed λ. If $R\to 3N-5$ then $\|F_{FWT+R}-F_{WT}\|\to 0$.*

From the above follows that by enlarging the range of best wedgelet computation one lengthens the time of computations but, on the other hand, from the construction of the algorithm follows that simultaneously one improves the quality of the approximation.

Note, finally, that in order to approximate an image one must perform the wedgelet transform first and then the tree pruning algorithm. So, in the first case, basing on the proposed approach, one needs to fix the parameter R (which relates time versus quality) and then, in the second case, one needs to fix the parameter λ (which relates compactness versus quality) in order to obtain wedgelet approximation of the image. So, this approach is very flexible. It is very convenient situation for the user who may decide what to choose: appropriately fast or exact approximation. In reality the choice depends on application.

4 Experimental Results

In order to test in practice the theory presented in the paper a number of numerical experiments have been performed with the use of Pentium IV 3 GHz processor. All the algorithms were implemented in the same programming environment, Borland C++ Builder 6, in order to ensure reliable results.

4.1 FWT Versus WT

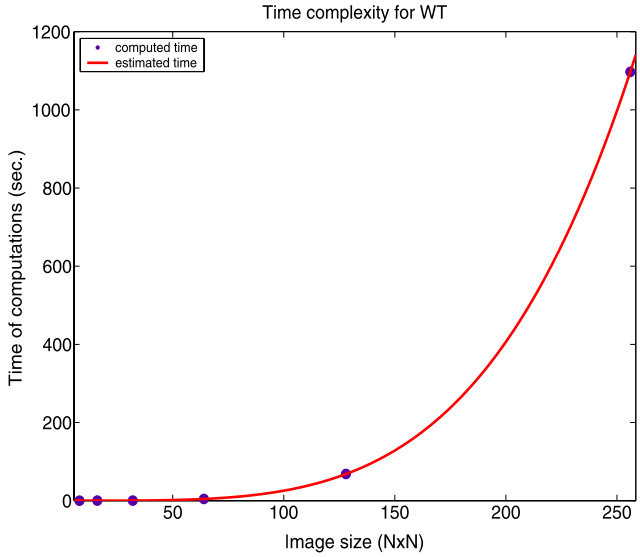
To test computation time of WT and FWT the test image "Lena" has been prepared in different sizes. Since the computation time is not dependent on the image content there is no need to present other examples. Indeed, for the other ones the computation time is practically the same. In Table 1 the computation times of WT and FWT for the test image of different sizes are presented.

As one can see from Table 1 the computation time for FWT falls down drastically in comparison to WT. Additionally, in Fig. 5 the theoretical versus practical computation times for WT and FWT are presented. The dots represent experimentally measured computation times whereas the solid lines constitute their approximation by functions of order $O(N^4\log_2 N)$ and $O(N^2\log_2 N)$ for WT and FWT, respectively. As follows from the plots practical computations match nearly exactly the theoretical estimations presented in the previous sections. Additionally, note that the plots are not comparable between each other directly since the axes representing time are in different scales. It is very difficult to draw the two plots in one coordinate system because the plot of FWT is too close to the horizontal axis to be well visible. However, it best reflects the power of the new method.

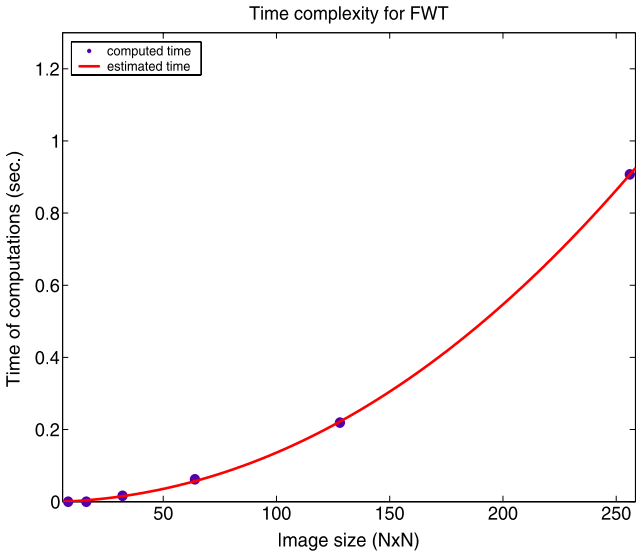
In practical applications also the memory size of the considered method plays an important role. In the case of FWT the memory occupancy is of the same order as of WT and equals $O(N^2\log_2 N)$. It follows from the fact that in

Table 1 Time of computation of WT and FWT (sec.) for image "Lena"

Size	WT	FWT
256	1097.156	0.907
128	68.203	0.219
64	4.344	0.062
32	0.265	0.016
16	0.015	< 0.001
8	< 0.001	< 0.001



(a)



(b)

Fig. 5 Computed and estimated time complexity for images of different sizes for (**a**) WT, (**b**) FWT

both cases the same kinds of coefficients are stored in the quadtree. In the case of WT the wedgelets' dictionary is nowhere stored, and the best matching wedgelet is found by checking all possible beamlet connections and computing MSE for everyone. In FWT the only difference is that such computations are performed only once per square segment for beamlet parameters obtained by integration (see formulae (15) and (14)).

4.2 From FWT to WT

As stated in the previous section the result of FWT is not exactly the same as the one of WT. However, it can be improved by the use of FWT + R. In Table 2 the PSNR quality of image decomposition on different levels by FWT + R and WT are gathered. In order to testify the independency of noise of the proposed method the artificially contaminated images have also been tested. The added contamination is Gaussian noise with zero mean and normalized variance depicted in the table. As follows from the gathered data the added noise does not disturb the efficiency of the proposed method. Additionally, one can see that depending on the decomposition level FWT + R assures the same results as WT for different values of R. Indeed, for sixth level FWT + 5 gives the same result as WT, for fifth level it is FWT + 10, etc.

Additionally, in Table 3 mean quotients (measured for all tested images) between PSNR values of the methods FWT + R and WT for different levels of decomposition are presented. From that table follows that FWT assures the image quality of more or less 94.7% of the quality given by WT. And then the longer the computations we perform the better quality we obtain. However, it is dependent on the level of decomposition. It means that, for example, FWT + 5 assures exactly the same decomposition on sixth level as WT, but only 96.42% of the quality of WT for second decomposition level. It is worth mentioning that in practical applications levels 4–7 are mainly used in approximations, so $R = 20$ can be treated as the upper bound for the practical use of FWT + R.

As follows from the experiments very often edges found by FWT are very close to the reference edges found by WT. To visualize that fact in Fig. 6 differences between WT and FWT + R for different values of R are presented. White areas denote exact matching and the darker the image the larger the error. From image (a) it is seen that in great majority of square segments the edges from FWT and WT are quite close. Applying FWT + R (images (b)–(d)) causes that appropriate edges are more and more close.

From the experiments performed on a number of images follows very important fact. In the areas where in the original image are presented well defined edges FWT assures the results very close to WT. On the other hand edges obtained from FWT are not so close to the ones given by WT mainly in the areas where there are no well defined original edges. In other words, original edges are detected very close by FWT whereas false edges are detected more differently. From practical point of view it is very reasonably approach since false edges detected by WT, anyway, are not practically used in many image processing tasks as, for example, edge detection. Indeed, the most important wedgelets are the ones which reflect original edges in an image.

In contrast to the quality improvement it should be presented how the computation time of FWT + R changes in practice. In Fig. 7 the computation time of FWT + R for different values of R are presented. And, similarly as in the

Table 2 Numerical results of FWT + R and WT decompositions on different levels for sample images with and without noise (PSNR)

Image	Level	FWT	FWT + 1	FWT + 5	FWT + 10	FWT + 20	FWT + 45	WT
Bird	6	33.04	34.31	34.65	34.65	34.65	34.65	34.65
(without	5	28.71	29.74	30.40	30.47	30.47	30.47	30.47
noise)	4	24.81	25.44	26.35	26.51	26.54	26.54	26.54
	3	21.24	21.49	22.07	22.46	22.79	23.03	23.03
	2	18.59	18.99	19.17	19.37	19.69	19.84	19.91
Peppers	6	28.63	30.27	30.73	30.73	30.73	30.73	30.73
(without	5	24.65	25.47	26.30	26.39	26.41	26.41	26.41
noise)	4	20.62	21.13	21.98	22.23	22.43	22.45	22.45
	3	17.46	17.70	18.19	18.45	18.70	18.92	18.92
	2	15.01	15.19	15.46	15.74	16.01	16.51	16.51
Ballons	6	22.98	23.61	23.74	23.74	23.74	23.74	23.74
(noise	5	20.73	21.22	21.82	21.90	21.90	21.90	21.90
$V = 0.005$)	4	18.86	19.16	19.69	19.86	19.94	19.94	19.94
	3	17.05	17.19	17.57	17.75	17.89	17.90	17.90
	2	15.62	15.67	15.81	15.99	16.10	16.23	16.30
Monarch	6	21.18	21.21	21.21	21.21	21.21	21.21	21.21
(noise	5	19.07	19.55	20.15	20.21	20.22	20.22	20.22
$V = 0.015$)	4	17.27	17.44	17.92	18.01	18.14	18.15	18.15
	3	16.57	16.63	16.83	16.95	17.01	17.02	17.02
	2	16.12	16.14	16.20	16.27	16.34	16.39	16.40

Table 3 Mean quotients between PSNR values of the methods FWT + R and WT for different levels of decomposition (%)

Level	FWT	FWT + 1	FWT + 5	FWT + 10	FWT + 20	FWT + 45	WT
6	96.29	99.24	100.00	100.00	100.00	100.00	100.00
5	94.13	96.91	99.66	99.97	100.00	100.00	100.00
4	93.77	95.54	98.67	99.43	99.96	100.00	100.00
3	94.28	95.15	97.25	98.45	99.42	100.00	100.00
2	95.05	95.48	96.42	97.48	98.57	99.79	100.00

Fig. 6 Difference between WT and: (**a**) FWT, (**b**) FWT + 1, (**c**) FWT + 5, (**d**) FWT + 10, for fragment of Bird on third level of decomposition

previous plot, the dots represent experimentally measured computation time whereas the solid lines represent their estimations. As one can see from the plot the larger the value of R the longer the computation time. Additionally, note that by changing the value of R to a fixed limit one obtains the plot from Fig. 5(a).

Finally, let us observe the main trend which follows from the previous section—the longer computation time the better quality of image approximation. The practical confirmation of the trend is presented in Fig. 8 for the fourth level of decomposition of test image "Monarch" (similar plots are generated for the other levels of decomposition and the other

images as well). The plot presents computation time versus MSE of image approximation for different values of R. As one can see the dependency is inversely proportional. In the presented example the result of FWT + 45 is exactly the same as the one of WT. From the shape of the plot evidently follows that addition of only little time to computations causes significant improvement of quality. On the other hand, after passing the saddle point of the plot the situation is inverse.

However, note that in order to make image approximation one needs to perform additional step—tree pruning. But for every approximation this fast procedure takes nearly the same time of computation, since it searches through the whole quadtree once. So, in the case of image approximation by wedgelets the plot of time complexity versus MSE is situated a little above the one from Fig. 8 and is similar in shape. But because we deal in the paper with time complexity of the wedgelet transform rather than image approximation the former plot has been omitted. Instead, as the last example, the practical use of FWT in image approximation is presented. In Fig. 9 the sample results of test image "Bird" are presented. Namely, in the consecutive subfigures the results of FWT, FWT + 5 and WT, followed by tree pruning (with $\lambda = 60$) are shown. Though the PSNR values are slightly different for the all methods the visual quality is nearly the same. Additionally, one can note that FWT + 5 improves the PSNR quality of the image in comparison to FWT. However, as one can expect, WT still assures the best result.

Finally, it is worth mentioning that all tests have been performed for squared dyadic images. For an image of arbi-

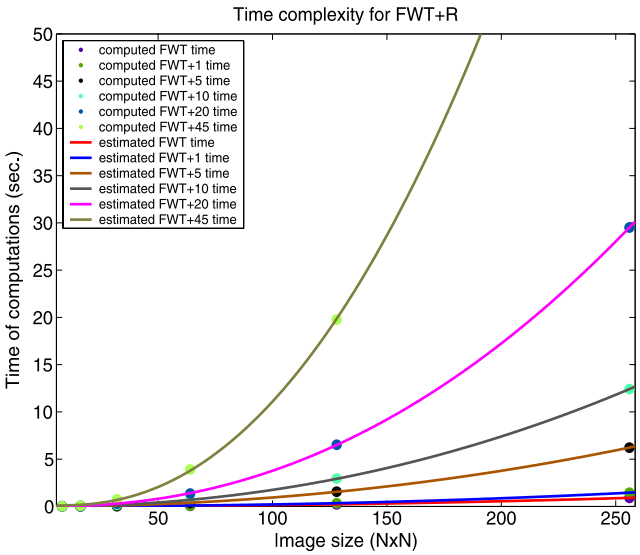


Fig. 7 Computed and estimated time complexity for images of different sizes for different ranges of additional searching of FWT + R

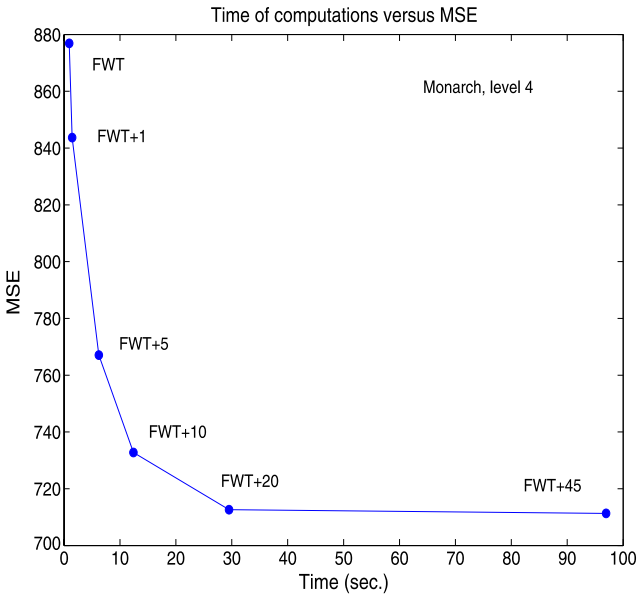


Fig. 8 Time complexity versus MSE for different searching ranges for fourth decomposition level of image "Monarch"

Fig. 9 Wedgelet approximation of image "Bird" for $\lambda = 60$ with the use of (**a**) FWT, PSNR = 29.97 dB, (**b**) FWT + 5, PSNR = 30.56 dB, (**c**) WT, PSNR = 30.76 dB

trary size there is a need to partition it on dyadic squares as is done in modern compression standards as, for example, JPEG2000.

5 Conclusions

In the paper the moments-based Fast Wedgelet Transform has been proposed. It has been proven that it works in asymptotically linear time with respect to the number of result pixels of the wedgelet transform. So far there has not been described in the literature any other method of wedgelet transform working in linear time. So the premise posed in [8] has been successfully fulfilled in this paper.

But, because the proposed method gives slightly different results than the classical WT, the improvement of the new method has been proposed in order to relate the time of computation versus the quality of approximation. In more details, in practice, depending on application, one can choose between fast computation and good reconstruction quality. In such a case the problem of wedgelet transform may be treated as two-criterial one. It is very important to note that in this case changes of quality are quite small while changes of computation time are quite substantial. So, in the proposed approach it is very little to lost while it is very much to win.

It must be mentioned, finally, that so far wedgelets-based algorithms were seen as usefulness in real time applications. But taking into account that the time complexity of the algorithm probably may be further improved by a constant and that the GPU parallel processing can be used (like, for example, NVIDIA CUDA) the proposed approach opens the door for the use of wedgelets in real time applications.

References

1. Bresenham, J.E.: Algorithm for computer control of a digital plotter. IBM Syst. J. **4**(1), 25–30 (1965)
2. Candès, E.: Ridgelets: Theory and applications. PhD Thesis, Department of Statistics, Stanford University, Stanford, USA (1998)
3. Candès, E., Donoho, D.: Curvelets—A surprisingly effective nonadaptive representation for objects with edges. In: Cohen, A., Rabut, C., Schumaker, L.L. (eds.) Curves and Surface Fitting, pp. 105–120. Vanderbilt University Press (1999)
4. Chong, C.W., Mukundan, R., Raveendran, P.: A comparative analysis of algorithms for fast computation of Zernike moments. Pattern Recognit. **36**, 731–742 (2003)
5. Darkner, S., Larsen, R., Stegmann, M.B., Ersboll, B.K.: Wedgelet enhanced appearance models. In: Proceedings of the Computer Vision and Pattern Recognition Workshops, p. 177. IEEE, New York (2004)
6. Deans, S.R.: The Radon Transform and Some of Its Applications. Wiley, New York (1983)
7. Demaret, L., Friedrich, F., Führ, H., Szygowski, T.: Multiscale wedgelet denoising algorithms. Proc. SPIE **5914**, 1–12 (2005)
8. Donoho, D.L.: Wedgelets: Nearly-minimax estimation of edges. Ann. Stat. **27**, 859–897 (1999)
9. Donoho, D.L., Huo, X.: Beamlet pyramids: A new form of multiresolution analysis, suited for extracting lines, curves and objects from very noisy image data. Proc. SPIE **4119** (2000)
10. Friedrich, F., Demaret, L., Führ, H., Wicker, K.: Efficient moment computation over polygonal domains with an application to rapid wedgelet approximation. SIAM J. Sci. Comput. **29**(2), 842–863 (2007)
11. Liao, S.X., Pawlak, M.: On image analysis by moments. IEEE Trans. Pattern Anal. Mach. Intell. **18**(3), 254–266 (1996)
12. Lisowska, A.: Geometrical wavelets and their generalizations in digital image coding and processing. PhD Thesis, University of Silesia, Sosnowiec, Poland (2005)
13. Lisowska, A.: Second order wedgelets in image coding. In: Proceedings of EUROCON '07 Conference, Warsaw, Poland, pp. 237–244 (2007)
14. Lisowska, A.: Geometrical multiscale noise resistant method of edge detection. In: Lecture Notes in Computer Science, vol. 5112, pp. 182–191. Springer, Berlin (2008)
15. Lisowska, A.: Image denoising with second order wedgelets. Int. J. Signal Imaging Syst. Eng. **1**(2), 90–98 (2008)
16. Mallat, S., Pennec, E.: Sparse geometric image representation with bandelets. IEEE Trans. Image Process. **14**(4), 423–438 (2005)
17. Mukundan, R., Ong, S.H., Lee, P.A.: Image analysis by Tchebichef moments. IEEE Trans. Image Process. **10**(9), 1357–1364 (2001)
18. Popovici, I., Withers, W.D.: Custom-built moments for edge location. IEEE Trans. Pattern Anal. Mach. Intell. **28**(4), 637–642 (2006)
19. Romberg, J., Wakin, M., Baraniuk, R.: Multiscale wedgelet image analysis: Fast decompositions and modeling. In: IEEE International Conference on Image Processing, vol. 3, pp. 585–588 (2002)
20. Romberg, J., Wakin, M., Baraniuk, R.: Approximation and compression of piecewise smooth images using a wavelet/wedgelet geometric model. In: IEEE International Conference on Image Processing, vol. 1, pp. 49–52 (2003)
21. Teh, C.H., Chin, R.T.: On image analysis by the methods of moments. IEEE Trans. Pattern Anal. Mach. Intell. **10**(4), 496–513 (1988)
22. Todorovic, S., Nechyba, M.C.: Detection of artificial structures in natural-scene images using dynamic trees. In: Proceedings of the 17th International Conference on Pattern Recognition, vol. 1, pp. 35–39 (2004)
23. Wakin, M., Romberg, J., Choi, H., Baraniuk, R.: Rate-distortion optimised image compression using wedgelets. In: IEEE International Conference on Image Processing, vol. 3, pp. 237–324 (2002)
24. Walker, J.S.: Fourier analysis and wavelet analysis. Not. Am. Math. Soc. **44**(6), 658–670 (1997)
25. Willet, R.M., Nowak, R.D.: Platelets: A multiscale approach for recovering edges and surfaces in photon limited medical imaging. IEEE Trans. Med. Imaging **22**, 332–350 (2003)

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.