BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 66, No. 2, 2018
DOI: 10.24425/122100

The realisation of selected signal processing functions by means of stack filters

Z. KUŚ* and A. NAWRAT

Silesian University of Technology, Institute of Automatic Control, 16 Akademicka St., Gliwice, Poland

Abstract. The goal of the paper was to shorten the calculation time by realising all used signal processing algorithms in the form of stack filters. The architecture of these filters allows us to process signals using the advantages of hardware processing and simultaneous signal processing. This paper elaborated on the synthesis of stack filters which realise median, averaging, opening and closing operations. The novel achievement was to develop the binary box method which allows us to obtain stack filters for more complex algorithms. This method consists of two stages and requires that we construct a spatial structure of the data. This structure allows us to examine the stacking property in two steps. Obtained in this way architecture of the function is predisposed to VLSI implementation. The authors devised this method transforming the averaging filter into stack filter; however, the invented binary box method allows us to synthesise stack filter which realises more complex signal processing algorithms. The only assumption which limits the class of acceptable algorithms is the fact that the algorithm has to satisfy the stacking property at each stage of the signal processing. The proposed approach allows us to convert well-known signal processing algorithms into realisation which guarantees significantly greater speeds of signal processing.

Key words: stack filters, median, opening and closing operations.

1. Introduction

Signal processing functions are essential elements of many practical applications, especially in image processing systems. The body of literature discusses widely image processing [1] because of its multiple applications [2–7].

One of the most popular application of the image processing systems is the system which can recognise the object's in unknown environment. An example one of such problem is the problem of the control in the systems which are made for object tracking, object observation or supervision of the object.

The location of the object may be used to steering the optic system into proper direction. Dynamical properties of the observed object, which are obtained from the localisation system, decide about the parameters of the control system which works with the optic system.

The problems of such a control system are discussed in [8–10]. We may state that the properties of the system which can recognise the object's movements have crucial influence on the problems with the synthesis of the control systems. Some of such a problems were presented in [8–10].

High quality of the system which can find object location results in high quality of the control system operation. The calculation time in the image processing system is one of the most important parameters for real time application. Realisation of the signal processing in the form of stack filter allows us to obtain very short time of calculation.

There are linear and non-linear functions which find their application in the field of image processing. However, a lot of these functions are numerically complicated and require a considerable amount of time for calculations. The most popular functions are e.g. median [17–19], average [1] functions or morphological operations [28]. The stack filters [22], which have been known since 80s, have great features when we need to speed up calculations. Unfortunately, the cost, we have to incur, is the hardware; however, the more expensive hardware results in the fact that the time of calculation may be extremely shortened. The only constraint which limits the class of functions, which may be represented in the form of stack filters, is the requirement that the transformed function has to satisfy the stacking property [22] at each stage of signal processing. We may find a lot of methods for stack filters synthesis which allow us to get stack filters based on minimising various indicators. In such a case, the stack filter is a solution of the optimisation problem. In this way, we obtain stack filter with specific features depending on requirements imposed by a given problem formulation. In this paper, similarly to many examples in the literature, we use the functions which are widely used in image filtering, segmentation or pattern recognition tasks. Our goal is to find the stack filter form for selected functions. This problem may be complicated by the constraint which causes that we are able to use only positive Boolean functions in the architecture of the stack filter [22]. The collected and perfected methods of creating stack filters will allow us to obtain the form of processing which is dedicated to VLSI implementation. Median and averaging filters, which are used for image filtering, will be examples of non-linear and linear functions respectively.

*e-mail: zkus@interia.pl

Manuscript submitted 2017-03-05, revised 2017-07–30, 2017-09-14, 2017-10-16 and 2017-12-10, initially accepted for publication 2017-12-20, published in April 2018.

The main body of this paper will be divided into two parts. The first one is devoted to basic formulation of the stack filter properties and the second one discusses the realisation of the used mathematical operations in the form of stack filters.

2. Literature review and the usefulness of stack filters

There is a wide body of literature which discusses the architecture of stack filters made in VLSI technology. We can find the examples of such deliberations in e.g. [35, 42, 43, 46–57]. All publications indicate on the fact that standard parallel stack filters architecture ([50] Fig. 2.) is the fastest whereas its disadvantages is the big size and cost of manufacturing the processor. However, the comparison of the calculation time can be made by analysing the operations which have to be conducted from the moment of entering data into the filter to the moment of obtaining the output result from the filter. Below, we will present the simplified analysis of the operations indispensable for carrying out stack filtering.

It is commonly known, that the signal calculation time is dependent on, among other things, the necessity of writing data into the register and reading data from the register, and performing mathematical operations. These mathematical operations may be conducted for the numbers of binary-type (0,1), integer-type numbers (integer numbers) or floating-point-type numbers (the approximate values of real numbers). The calculation time of the abovementioned operations for the data in the binary form is the shortest.

In the next paragraph, we will present the approximate analysis for the number of the clock cycles necessary for processing data contained in one window in order to obtain stack filter output for one pixel of the image.

2.1. Stack filter clock cycles. The stack filter conducts data processing in the following stages.

The first stage comprises putting the M-value signal in the filter input (*the first clock cycle*).

In all solutions the input data are written simultaneously – the number of input registers is equal to size of the filter window. The whole content of the window is processed simultaneously.

There is a close relation between the size of the window selecting pixels from an image to processing and the size of the processor's register. The advantage of such a solution is the fact that although the wider window results in a bigger register size (the cost of the microprocessor increases), it does not influence the calculation time.

The second stage comprises thresholding M-value signal into M-1 binary signals (*the second clock cycle*).

The content of all registers containing data from the window moving on the filtered image is thresholded simultaneously. In this way, at each threshold level we obtain the sequence of binary numbers which are the arguments of the Boolean function which defines stack filter. In this case, the following economic problem appears: the greater the number of grey levels is, the greater number of threshold levels. This results in a bigger size of the processor what is strictly connected with the bigger processor cost; however, it does not influence the calculation time.

The third stage comprises calculating the values of Boolean functions (*the third clock cycle*).

The calculations are carried out simultaneously at all threshold levels. In this case, the calculations come down to passing the binary data vector through the appropriate combination of the logic gates. At this stage, it is especially valuable to implement a stack filter as a dedicated microprocessor.

On the one hand, it is a constraint – one microprocessor provides one Boolean function and we may have only one type of stack filter, but on the other hand, we often do not need to modify stack filter which is well adjusted to the features of the noise. This situation is particularly common when we need fast filtering which may be applied online to the video stream.

The fourth stage comprises detection of the highest level of the binary output signal at which we met '1' (*the fourth clock cycle*).

Thanks to property of threshold decomposition and using positive Boolean functions, we obtain the output values of the Boolean functions which are ordered in the following way: for a given position of a window we obtain an output binary vector (the vertical column of binary numbers at the filter output). This column has such a form that there are only zeros at the top and there are only ones at the bottom. There is the one point where ones change in zeros. It may be observed in Fig. 1.

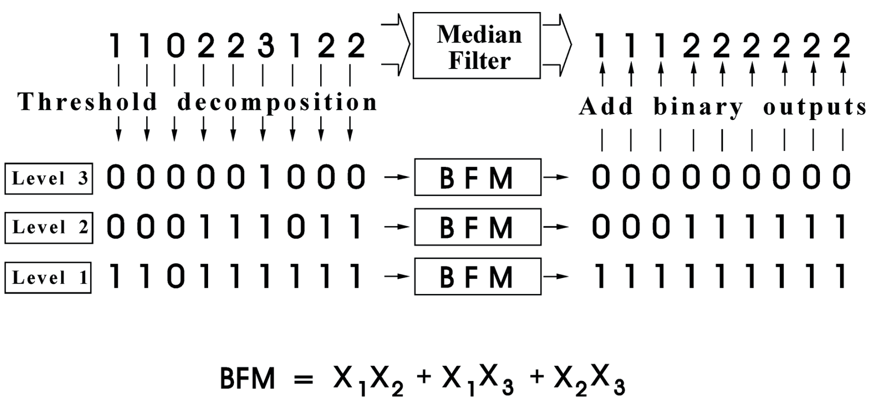

Fig. 1. Stack filter architecture on the example of median filter with three elements inside window

Due to this property, summing of the binary outputs may be replaced by the detection of the highest level where the last '1' is located. At this stage, the number of threshold levels does not influence the processing time for a single filter window.

In this way, we obtain the value of the M-value output signal. The key feature of this processing is the fact that the signals are processed simultaneously at all stages of binary signals processing what is equivalent to passing the signal through the logic gates. The number of such logic ways corresponds to the window width and the number of grey levels. The abovementioned results in the fact that the stack filter made as a dedicated processor in VLSI technology is the fastest realisation of the filter.

Each filter realisation based on programmable processors requires more clock cycles for writing to registers, reading from registers and mathematical calculations which altogether extends data processing time. We obtain such a result when we compare the stack filter and programmable processors made in the same technology.

If, as an alternative to the stack filter, we use the computer program, which was programmed in any computer language and compiled to the processor's language, it is necessary to take further steps. Firstly, the data is entered into a register; secondly, it is sent to the mathematical processor; then the calculations are carried out and the result is finally sent back to a register. The abovementioned four cycles of the processor, necessary for carrying out the calculations in the stack filter, need considerably shorter time to obtain a result. It is due to the fact that data is instantaneously sent to the logical gates which process the data. This result concerns all types of the mathematical operations which are discussed in this paper.

Concluding, we may state that the larger size of a window and the larger number of grey levels do not influence the time of window data processing. This is an obvious advantage of stack filters over the solutions which use non-dedicated programmable processors. The total time of image processing depends proportionally only on the image size. However, this problem concerns all types of image filtering systems.

2.2. The motivation of dealing with stack filters. Considering the motivation of dealing with stack filters nowadays, it is worth reminding two features, well known from the first publications, which make the problem still interesting to research.

The first feature, which stems from a big number of positive Boolean functions, is the diversity of influence a filtered signal, viz. the diversity of stack filters which have various features.

The second feature is to shorten data processing time. It is a particularly significant problem which has appeared together with the development of image processing applications – the images which often appear in the form of video streams. It is particularly worth quoting undermentioned deliberations indicating on the significance of this problem on account of the second feature.

The problem of time processing is particularly visible in the case when we use more than one kind of image for object tracking. Using thermal images together with a visual image results in double amount of data which have to be processed [2, 3]. In such a case, each way of shortening the processing time is extremely useful. Due to using stack filter, in such a case, the number of image types do not influence the processing time because operations for each image type may be conducted simultaneously.

The problem of time processing is crucial for the object tracking task. The papers described below present various methods which allow us to improve pattern recognition process. The key feature of these solutions is decreasing the calculation complexity and shortening calculation time. Additionally, if it was possible to realise the necessary operation as stack filters, then the methods presented below would allow us to decrease the sizes of the stack filter elements.

The method of developing the invariant functions vector for objects recognition from a given objects set is presented in [4]. This method allows us to select the proper set of the elements of the pattern vector – the set which takes into consideration the features of given tracked objects.

The method of guaranteeing the separation between the recognised object and background is presented in [5]. It reduces the size of the pattern vector without losing the quality of the recognition process.

The problem of minimizing the image resolution in order to increase the computing speed without losing the separation of the recognised patterns is presented in [6]. Presented results show that it is possible to significantly decrease the amount of processed data without losing the object.

Adjusting the thresholds to the recognised pattern in order to improve the separation between the recognised patterns is discussed in [7]. This solution is dedicated to realising calculations in stack filter architecture due to the fact that the lower the number of threshold levels is, the lower cost of the stack filter.

The papers [8, 9] and [10] elaborate on the problem of the synthesis control system for UAV and camera head which are used for object tracking. The main conclusion which may be drawn from these publications is the main role of data processing time for an object tracking task. Image processing system was not examined in these papers; however, we know that the position of the tracked object is calculated on the basis of the image processing system. In such a situation, the time of image processing has to be treated as death time in the model of the controlled plant. It can significantly worsen the operation of the control system. The considerations discussed in [8–10] and [11–13] convince us how important is the problem of shortening processing time.

Yet another example of such a situation when we aim at shortening the processing time is the problem of image retrieval [14–16]. This problem is particularly important when we need to enhance the quality of the image used to calculate UAV's trajectory in the object tracking task. In such a case, we have to guarantee high image quality and short calculation time.

The abovementioned papers do not discuss stack filtering, although in each of these problems it is visible how the problem of shortening data processing time is crucial in particular applications of the image processing system.

The first publications concerning stack filters [22, 23] were published in the 80s of the last century. Since then, we have been observing that publications concerning stack filters have been developing in two directions. One of them concerns stack filter implementation, whereas the second focuses on the features of stack filters based on different Boolean functions.

2.3. The synthesis of stack filters with different features. The papers presented below consider the synthesis of stack filters with different features which depend on the used Boolean function.

The problem of stack filters which minimise mean absolute error was presented in [24] and [25]. The problem of morphological filters and their relations to median, order-statistic, and stack filters were presented in [26] and [27]. Presented considerations allow us to construct popular basic filters in the form of stack filters. An algorithm that applies a stack filter simulating the mean curvature motion equation via a finite difference scheme is discussed in [32].

An unsupervised design of stack filters for recovering images from the impulse noise is proposed in [33]. The proposed method bases on the tree structure optimization.

Binary quantum-behaved particle swarm optimization (BQPSO) algorithm, and the design of stack filters for noise suppression using BQPSO algorithm was proposed in [34].

An ant colony-clonal selection algorithm for stack filters' optimizing is presented in [36]. The criterion used for minimising was the MMAE criterion,

The synthesis of adaptive stack filters is discussed in [37]. In this work, the behaviour of adaptive stack filters is evaluated for the classification of synthetic aperture radar (SAR) images, which are affected by speckle noise.

An exact algorithm for optimal MAE stack filter design is considered in [38]. The authors show that the duality of the integer programming formulation of the filter design problem is a minimum cost network flow problem. Next, they present a decomposition principle that can be used to break this dual problem into smaller sub-problems. Finally, they propose a specialization of the network Simplex algorithm based on column generation to solve these smaller sub-problems.

Another approach to optimizing stack filters is presented in [39]. As we know, the design of stack filters may be formulated as a course of optimizing positive Boolean functions (PBF). In this paper, the authors use clone selection algorithms (CSA) to optimize stack filters under the mean absolute error (MAE) criterion.

The authors of [40] indicate on the problems of using an improved immune memory clonal selection algorithm (IMCSA) for designing stack filters. The problem is the fact that the evaluation of each candidate solution involves the greatest computation complexity. In this paper, the authors propose representing the positive Boolean function (PBF) in objective function and a hybrid computation approach (software and hardware) for the calculation tasks. An analysis framework that interprets a Boolean vector to denote membership in a partition set is present in [41]. Using this framework a new algorithm to compute selection probabilities from positive Boolean functions is derived. Next, they present a method to synthesize a stack filter from specified selection probabilities.

The problem of improving the design of filters for enhancing colour images is discussed in [44]. The proposed technique uses an image fidelity measure based on models of the human visual system – such as the visible differences predictor (VDP). It is used in a nested loop training algorithm. In the inner loop of the algorithm, a stack filter is trained under a weighted mean absolute error (WMAE) Criterion to remove noise. In the outer loop, the VDP is used to train the weights in the WMAE criterion to ensure that the filter to which the algorithm converges is the one that produces output images that are as visually satisfying as possible.

The chapter in [45] concerns the comparison of various stack filters features and approaches. We can find there such subjects as minimum mean absolute error stack filtering, computational requirements of stack filters, an application of stack filters to noise reduction and edge detection.

The body of presented literature shows that the research on stack filters has not finished, especially in the last decades there has been a signficant development of the stack filters.

2.4. Stack filter implementations. The papers presented below consider stack filter implementation.

The bit-plane stack filter algorithm for focal plane processors is presented in [35]. This work discusses a novel parallel technique to implement stack morphological filters used for image processing.

Several field programmable gate array (FPGA) implementations of stack filters are presented in [42]. This design which was presented in the paper joins a modified version of the Bit-Serial method, already used with Stack Smoothers. The parallelism contained into the algorithms involved makes the FPGA, with its remarkable ability to implement complex digital systems, to be especially appropriate for this application. Owing to the fact that the FPGA contains the parallelism and has its remarkable ability to implement complex digital systems, the authors consider this solution especially appropriate for this application. What was found in this paper is the fact that the properties of one of the mathematical structures involved in the Stack Filter theory could help to significantly improve the filter's performance and to optimize its use of hardware resources.

The paper [43] discusses the dynamic non-linear threshold decomposition algorithm for implementing stack filters. This paper has introduced a new non-linear threshold decomposition architecture to reduce the complexity of the algorithm. Furthermore, by combining with the character of the image, a dynamic non-linear decomposition architecture is proposed to implement stack filters. This approach can shorten the running time in the filtering procedure and improve the quality of the output image simultaneously.

A new, however in the 90s, approach to the implementation of stack filtering was suggested in work [47]. Based on this approach, efficient algorithms and a new VLSI architecture were developed and presented in [46]. The presented solution is valuable because it connects two features: high speed of data processing and programmability. This fact allows us to support adaptive stack filtering in real time. The presented architecture of stack filter is based on the idea of information coding about order relations between input samples within the filter's window by binary arrays (matrices). This information coding is conducted in such a way that this information gives the possibility to find the output of a stack filter by applying simple tests. Moreover, it can be easily updated during the sliding of the window.

The main part of [48] concerns an algorithm for one-dimensional stack filter, along with its hardware structure. The proposed digital realization is simple and modular. The realization proposed by authors is based on a compare and swap circuit. Compared with other reported one-dimensional structures for stack filters, and based on VLSI implementation, the new structure has a better performance in terms of area and time delay. The new one-dimensional structure proposed in this paper has been extended to a separable two-dimensional stack filter.

Yet another architecture of rank order median filter is presented in [49]. The architecture processes all window samples in parallel in the bit-serial manner. Unlike related architectures, it neither sorts/swaps nor modifies the window samples. Due to this fact it requires less hardware resources. The authors showed that in their solution, to process k samples, each of N-bits in size, the architecture uses N shift registers of k bits each and a simple logic. It results in $N+1$ clock cycles independently to the window size.

A particularly interesting work, in the area of stack filer implementation in VLSI technology, is the paper [50] which contains an overview of the various solutions of the stack filter architecture problem. Presented approaches are aimed to maintain high speed for the data processing (due to the parallelism of signal processing) as well as reduction of the cost which has to be incurred for the production of the processor realising stack filter (due to the changes in the data processing structure).

According to the authors of [50], the basic stack filter structure presented in Fig. 2. in [50] is not economical in production due to the fact that it has a large number of threshold levels M (e.g. grey scale images).

According to [51] and [52], a single threshold decomposition unit is used for the input sample entering the window while the previously decomposed samples are stored in shift registers. Despite the theoretical interest of this very fast architecture, it is not practical for the large values of M because of the fact that the area of the processor is proportional to the number of PBF (positive Boolean function) units.

At the same time, the authors pay attention at the fact that the size of the window is usually smaller than the number of grey scale levels. This fact is the premise to decrease the number of grey scale levels to the size of the window. It was obtained by ordering input data in the window which is a kind of the compression of the input data. It is the main idea of the solution proposed in [50].

Another approximation is based on an algorithm of binary search [53] implemented in a bit-serial way. A stack filter realizes nonlinear function processing by a "stack" of $2^k - 1$ binary processing circuits, where k is the number of bits in the input signals. It is shown in this paper that the function of a stack filter can be realized in k-step recursive use of one binary processing circuit. This implementation uses only one PBF unit and $b = [log_2 M]$ calculations of the PBF in sequences. It is slow if b is large and the throughput of the structure may be unacceptable for high-speed applications.

Parametrizable decompositional algorithms and associated VLSI architectures which generalize [53] are introduced in [54] and [55]. They offer a series of tradeoffs between hardware complexity and the degree of parallelism. A computationally efficient algorithm for general stack filtering was presented in [54]. This algorithm was based on the use of Fibonacci p-codes. The resulting architecture allows compact VLSI implementations. A running stack filtering algorithm and architecture was developed for filters based on a subclass of PBF's. The bit-serial approach for stack filter design, presented in [55], is generalized to digit-serial processing for creating a class of pipeline-parallel architectures using the lexicographic representation of data. Threshold decomposition and bit-serial structures, as well as

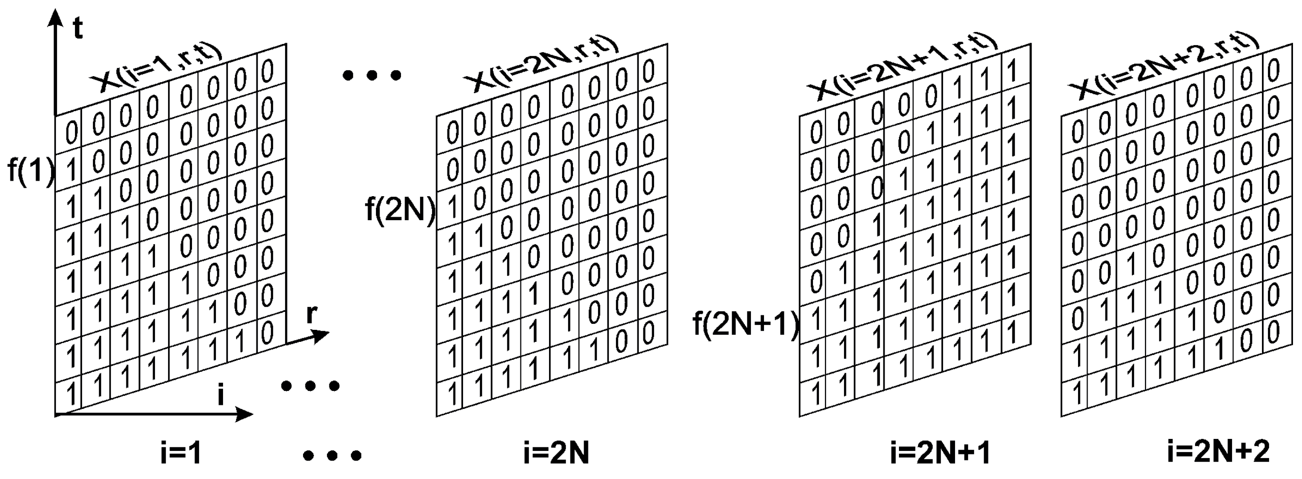

Fig. 2. The structure of argument tables in binary box

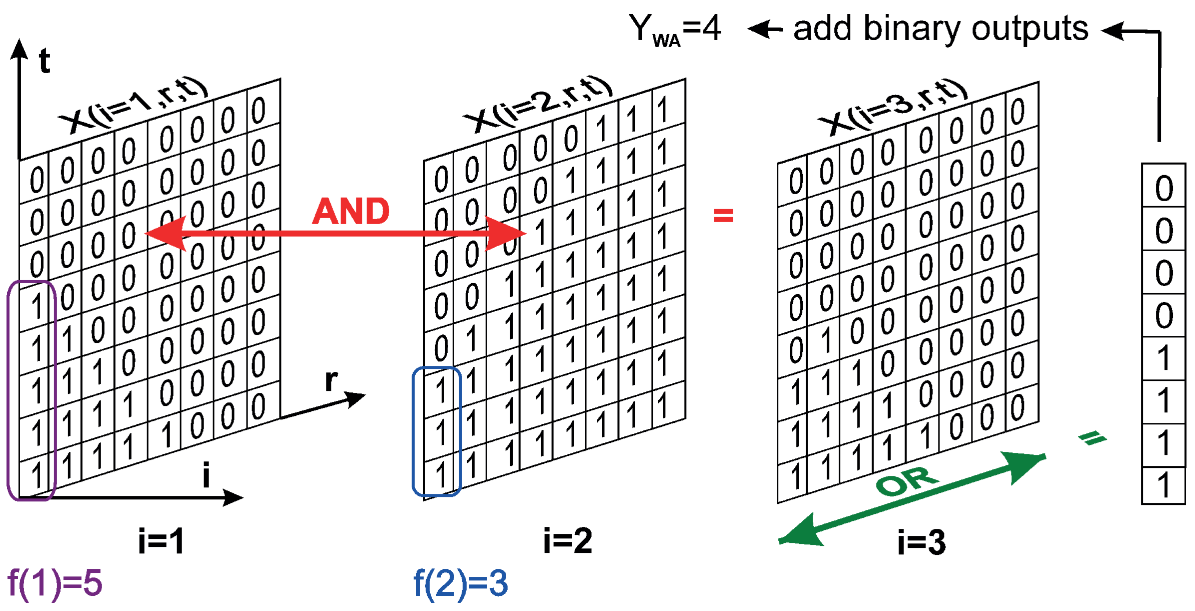

Fig. 3. The structure of argument tables in binary box which realises WA filtering for two input values $f(1) = 5$ and $f(2) = 3$

Fibonacci p-code based designs are particular cases of the proposed approach.

The architecture proposed in [56] can be seen as a bit-level pipelined version of the bit serial implementation of stack filters, thus leading to implementations with a latency which is too high for large values of M. That is, although less expensive in hardware, these solutions might be non-suitable for real time applications requiring parallel architectures.

A more efficient parallel architecture for stack filters uses an input compression algorithm [57]. It maps the sample points appearing in a window of size L onto the set of integers $\{0, 1, \ldots, L-1\}$ before any filtering takes places. By combining input compression with the stack architecture principle, the number of threshold levels is now $(L-1)$ instead of $(M-1)$, so $(L-1)$ binary filters are required instead of the previous $(M-1)$.

According to the abovementioned publications, stack filters area of science has been developed in the last few years. However, publications presented in this paragraph are not connected directly with the subject of this paper, but they show that there are a lot of problems which are still worth developing. It proves that stack filters are present in modern science.

3. Basic properties of the stack filters architecture

Stack filters constitute a class of nonlinear filters. These filters have two fundamental properties: threshold decomposition and stacking property [22]. A lot of nonlinear filters are included within the class of stack filters, especially such filters as: median filter [17], rank-order filters [20] and morphological filters [28, 21].

The stack filter architecture is illustrated in Fig. 1 where stack filter realises window-width three median filter for 1D signal [22–25]. Figure 3.1. presents the fact that stack filtering is based upon:

a) the threshold decomposition of the four-level input signal into three binary signals;

b) the binary median filtering of these binary signals;

c) summing the output binary signals which results in a four-level output signal.

In order to formalise the discussed terms, we will refer to the following definitions.

The definition of the threshold decomposition [22–25]. The threshold decomposition of an M-valued sequence R with integer elements $R(k)$ is the ordered set of $(M-1)$ binary sequences, called threshold sequences:

$$T_1, T_2, \ldots, T_{M-1} \tag{1}$$

which elements are denoted as follows:

$$T_i(j) = \begin{cases} 1 & \text{for} \quad R(j) \geq i \\ 0 & \text{for} \quad R(j) < i \end{cases} \tag{2}$$

□

The definition of the stacking property [22–25]. The ordered set M of binary vectors $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_M$ satisfies the stacking property if:

$$\bar{x}_1 \geq \bar{x}_2 \geq \ldots\ldots \geq \bar{x}_M. \tag{3}$$

The Boolean function $B(.)$ satisfies the stacking property if:

$$\bar{x} \geq \bar{y} \quad \Rightarrow \quad B(\bar{x}) \geq B(\bar{y}). \tag{4}$$

□

The necessary and sufficient condition of satisfying the stacking property by the Boolean function [22, 23].
The Boolean function satisfies the stacking property if and only if it can be expressed in the form of a Boolean expression containing no complements of the input variables. □

Taking into account the abovementioned definitions, we can observe the following property. Owing to the fact that thresholding results in a set of input binary sequences which satisfies the stacking property and the fact that the Boolean function used for binary filetring also satisfies the stacking property, we can state that the output binary sequences satisfy the stacking property.

Since zeros and ones are ordered in these sequences in such a way that for a given moment of time (a vertical column of zeros and ones) we have only zeros, moving from top to bottom, and after the first switch from zero to one we have only ones till the end of the column, we obtain the following simplification of the multi-valued output signal calculation.

This simplification consists in the fact that instead of summing ones for a given column, in order to obtain the multi-valued output for this element of the multi-valued output signal, we detect the threshold level for which the output value equal one occurs for the first time. During 2D grey scale image processing we assume the following. The number of threshold levels equals the number of grey levels minus one. The pixels of the input image are chosen by defining the window and assigning the particular window pixels to the variables of the Boolean function.

Therefore the part of the image which is covered by the window (for a given window location) is a multi-valued input sequence for the filter. Thresholding of this sequence gives us binary sequences which are the arguments of the Boolean function at each threshold level.

The values of the Boolean function for all threshold levels create a column consisting exclusively of zeros (at the top) and ones (at the bottom). The value of the pixel in the output image, in the place where the window was located, is obtained as a number of the threshold level where the value 1 is detected for the first time (moving from top to bottom).

The fact that stack filters use Boolean functions at each level of processing causes that the stack filters architecture is predisposed to implement this filters in VLSI technology.

The architecture of stack filters does not limit the size and shape of the filter window which is used for image processing. The example presented in Fig. 1 has data arranged in 1D sequence; however, this sequence may be obtained from the filter window of any shape and any size. It is necessary to define the way of ordering this data into the input sequence of the stack filter. This kind of data ordering is a part of a filter definition. It depends on the result we want to obtain. However, we may divide filters into two groups: the filters in which the result does not depend and depends on the arguments order. In both cases, the shape of the filter window has an unquestionable influence on the obtained result.

4. Stack filters architecture for selected linear and non-linear functions

The advantages of the hardware realisation of stack filters, especially in VLSI technology, cause that we will try to create stack filters which will realise well-known and very useful functions, such as the median filter, average filter, morphological filters or gradient operation. The functions which are selected as examples presented in the following subsections are chosen according to the criterion of usefulness in the image processing systems, e.g. [60–62].

4.1. Realising median filters as a stack filter. Median filter is an example of nonlinear filter with very useful properties. These properties constitute particularly edges preserve property and the fact that during filtering new values of signal do not appear. The median filters, on the whole, order statistics filters can be realised by using maximum/minimum mathematical operations as shown in [27]. It is known that in Boolean algebra the minimum (maximum) function corresponds to the expression of the logical product (sum) of the Boolean variables – it allows us to obtain an appropriate form of the Boolean function for the stack filter.

As far as the $k-th$ order statistic filter is concerned, it operates on the set A in this way that elements of A are arithmetically ordered and then the k-th element of this ordered sequence is the output of the filter. If the number of elements inside the window equals $2N+1$, then we can describe creating k-th order statistic filter as follows.

We choose all k-element combinations of $(2N+1)$ elements. The number of all elements equal $(2N+1)$ causes that the number of these combinations is $\binom{2N+1}{k}$. For each such combination, we calculate the minimum. Next, we calculate the maximum of all minimums calculated in the previous step. We can obtain a binary function when we replace max/min operations with logical operators and/or. An example of a binary function for $2N+1=3$ (3 elements inside the window) and $k=2$ (median) is shown in (5)

$$\begin{aligned} med &= \{x_1, x_2, x_3\} = \\ &= max\{min\{x_1, x_2\}, min\{x_1, x_3\}, min\{x_2, x_3\}\} \\ med &= \{x_1, x_2, x_3\} = x_1 \cdot x_2 + x_1 \cdot x_3 + x_2 \cdot x_3 \end{aligned} \tag{5}$$

where a $sum(product)$ corresponds to logical $or(and)$.

4.2. Realising average filters as a stack filter. Average filters are linear ones which are used to remove high frequency disturbances. One of the disadvantages of these filters is the fact that they blur the image. As it was presented in [26] and [27], averaging filters can be realised by using supremum/infimum (maximum/minimum) mathematical operations. It stems from relations between linear and morphological filters, cf [26]. Detailed assumptions, theorems and proofs are presented in [26]. The methodology of mean value calculation may be quoted from [26] where Section 4 presents relations between linear and morphological filters. Formula (40) from the theorem 9 in [26] is the basis for the method of calculating the average value for any window width, which is presented in (8). The example on page 1167 [26] presents the method for calculating the average value for two arguments using the formula (42) and the method for calculating the average value for three arguments using the formula (43). Presented methods are based on calculating average value by calculating a supremum of minima. Formulas (40), (42) and (43) in [26] allow us to derive (8) which presents the method for calculating the average value for any number of arguments. The deliberations presented between equations (42) and (43) in [26], used for calculating the mean value of two numbers, were used to derive (8) and (9), on the basis of which we can calculate the mean value for any number of numbers. We will present the formulation of the problem for discrete case. We assume that the number of elements inside the window is equal $2N+1$. The elements inside the window belong to the subset of integer numbers 0, 1, 2, …, 255 (e.g. the values of grey levels in grey images).

The size of the domain assumed for all calculations depends on the task the calculations are used for. In some applications it may be useful to decrease the domain size. For instance, for greyscale images it is common to use the domain equal $\{0, 1, 2, \cdots, 255\}$. All intermediate results of calculations have to be included in this assumed domain. It may result in an inaccuracy of the calculations. The assumed size of the domain has to guarantee that the inaccuracy will be acceptable.

In order to compute weighted average (WA), there is a factor $a_n \geq 0$ related to each window element. We assume that:

$$\sum_{i=1}^{i=2N+1} a_i = 1. \tag{6}$$

The elements of the input signal are denoted as $f(n)$. The output value of WA filter is defined as:

$$Y_{WA}(n+N+1) = \sum_{i=1}^{i=2N+1} a_i f(n+i). \tag{7}$$

According to the discussions presented in [26], the realisation of the abovementioned WA filter can be formulated as presented in (8)

$$\begin{aligned}\sum_{i=1}^{i=2N+1} a_i f(n+i) = \max_{r_i \in \{0,1,\ldots,255\}} \Bigg\{ \min \Bigg\{ f(n+1) - r_1,\\ f(n+2) - r_2, \ldots, f(n+2N) - r_{2N}, f(n+2N+1) + \\ + \frac{a_1 r_1 + a_2 r_2 + \ldots + a_{2N} r_{2N}}{1 - \sum_{i=1}^{i=2N} a_i} \Bigg\}\Bigg\}.\end{aligned} \tag{8}$$

The method of realisation WA filter formulated in (8), with using *max*/*min* operations, allows us to compute the approximation of a linear filter in the form of a nonlinear stack filter. However, the expression shown in (8) does not allow us to describe the creation of a stack filter in a simple manner. It was the reason that the novel method of stack filter creation was developed. The necessity of creating a data spatial structure is the reason why the method is called the Binary Box Method (BBM). BBM allows us to create the logical structure which is predisposed to VLSI implementation. Standard stack filtering, as it was presented in Fig. 1, consists of thresholding, binary filtering and summing. BBM allows us to realize more complex algorithms with the exploitation of advantages which result from stacking property. It causes that Stacking Property requirement has to be met at the each step of processing in Binary Box Method.

4.3. Binary box method for a WA filter. The construction of the stack filter which realises WA filtering described in (8) is equivalent to the following steps.

1) Creating 2-D argument tables.

In this case, these tables contain values for the elements $f(n+i) - r_i$ and last element from (8).

The tables in Fig. 2 contain the values of the signal (decreased, increased) thresholded with the threshold t. Firstly, the tables with the size of $(r_{max} \times r_{max})$ are filled with zeros. r_{max} is the number of the admissible values of $f(n)$, e.g. 256 grey levels.

Figure 2 presents forming the argument tables for $i = \{1, 2, \cdots, 2N\}$. For $r_i = 1$ we have $f(n+i)$ in the table. For $r_i > 1$ there are decreasing values $[f(n+i) - r_i]$ where $r_i = \{1, 2, \cdots, f(n-i) - 1\}$. Figure 2 presents also forming the argument table for $i = 2N+1$. For $r_{2N+1} = 1$ we have the value of $f(n+2N+1)$. For $r_{2N+1} > 1$ we have increasing values presented in (9)

$$f(n+2N+1) + \frac{r_{2N+1} \cdot \sum_{i=1}^{i=2N} a_i}{1 - \sum_{i=1}^{i=2N} a_i}. \tag{9}$$

Values a_i have to meet the condition (6) (the sum of all a_i is equal 1). Therefore the expression (9) may be transformed to the following form:

$$f(n+2N+1) + r_{2N+1} \cdot \frac{1 - a_{2N+1}}{a_{2N+1}}. \tag{10}$$

If the coefficient $\left(\frac{1-a_{2N+1}}{a_{2N+1}}\right)$ is denoted as α, then we can see that to the value of the function $f(n+2N+1)$ there are added in each step the values of r_{2N+1} that increase by 1 in each step and additionally they are multiplied by α. Therefore in the table for $i = 2N+1$ in Fig. 2 there will not be an increase by 1 in the columns for the subsequent values r, but there will be an increase by 1 times α. At the same time, we can see that irrespective of the non-integer coefficients a_i values the value α is always integer and if we represent a_i in the form of a fraction: $a_{2N+1} = {}^1\!/\!_\beta$, where β is integer, then we obtain $\alpha = \beta - 1$. For example, for each $a_i = \frac{1}{(2N+1)}$ we obtain $\alpha = 2N$. Therefore in this case, the rate of height increase for the columns of 'ones' in the table for $i = 2N+1$ will be twice greater than N for $N > 1$. For example, in the case from Fig. 3 we calculate the mean value for two numbers so we assume $a_1 = a_2 = 0.5$ and in this case $\alpha = 1$.

The terms in (8) correspond to the tables presented in Fig. 2. The element $[f(n+1) - r_1]$ corresponds to the table $X(i = 1, r, t)$. The element $[f(n+1) - r_2]$ corresponds to the table $X(i = 2, r, t)$. The penultimate element $[f(n+2N) - r_{2N}]$ corresponds to the table $X(i = 2N, r, t)$. The last element, presented in (10) corresponds to the table $X(i = 2N+1, r, t)$. When we consider the signs of variables r_1, r_2, etc., then we can notice that all variables r (except for the last r_{2N+1}) are subtracted from $f(\ldots)$, whereas the last r_{2N+1} is added to $f(n+2N+1)$. The variable r assumes values 0, 1, 2, …, 255. The range of these values results from the assumed domain. This domain corresponds to the grey levels in greyscale images. This assumption does not limit our deliberations. The abovementioned subtraction and addition decide about decreasing and increasing of the columns containing '1' in tables $X(i, r, t)$ in Fig. 2. In the tables for $i = 1$ to $2N$, the number of '1' in columns decreases for the subsequent values r whereas in the table for $i = 2N+1$ the number of '1' in columns increases.

Variable r_{2N+1} in (9) increases until the value of the expression (9) reaches r_{max}.

If the value of (9) achieves r_{max} for the value of variable $r_{2N+1} < r_{max}$, than the rest of the columns (the table for $i = 2N+1$ in Fig. 2 are filled with ones (two last columns of the '1' in the table for $i = 2N+1$ in Fig. 2.

The sequence of the above-created tables defines a finite discrete space (i, r, t) called binary box. The elements of this space will be denoted as $X(i, r, t)$.

2) Calculating 2-D table with the values of the first function operating on the tables from step 1.

In this case, the first function is a minimum. The elements of the table are denoted as $X(i = 2N+2, r, t)$ and calculated in (11)

$$X(i = 2N+2, r, t) = \bigcap_{i=1}^{i=2N+1} X(i, r, t). \tag{11}$$

3) Calculating the vector of values for the second function which operates on the rows (constant t) of the table calculated in step 2.

In this case, the second function is a maximum. The result is the vector with the thresholded output $Y_{WA}(n+N+1)$ of the WA filter. The elements of this vector for the thresholds t from 1 to $(r_{max} - 1)$ are denoted as $Y_{WA}(n+N+1)[t]$ and calculated according to (12)

$$Y_{WA}(n+N+1)[t] = \bigcap_{r=0}^{r=r_{max}} X(i = 2N+2, r, t). \tag{12}$$

Thanks to the stacking property, the calculation of the final result is nothing but the detection in $Y_{WA}(n+N+1)$ of the threshold level for which 1 occurs for the first time, moving from top to bottom. The number of this level constitutes the output of the WA filter.

Figure 4 presents the schematic diagram of the logical system which realises the average filter in the form of the stack filter. In the scheme, there was presented a division of the system into functional blocks; however, the realisation of the filter in the VLSI technology assumes making the system as a single processor.

Because of the necessity of carrying out a big number of connections between particular blocks, for the sake of figure's clarity, only selected connections were marked. Particularly, it concerns the 'AND' blocks which realise the operations described in (11) and 'OR' blocks which realise the operations described in (12).

To conclude the description of the variable $X(i, r, t)$ from the points 1, 2 and 3, we may state that the variable $X(i, r, t)$ is the block containing binary data which allow us to conduct the process of calculating Y_{WA}. The subsequent stages of this process are defined in the points 1, 2 and 3. In order to clarify the way we use the calculations described in (8) we will present two examples. The first of them presents an example for three input values, described in a general way, whereas the second one presents an example for two input values described in a great detail in the paper.

Example 1. In this example we assume that we will calculate the weighted average Y_{WA} for three input values with coefficients $a_1 = a_2 = a_3 = 1/3$. According to (8), we assume $N = 1$ what results in three input elements. Therefore we obtain

$$\begin{aligned} Y_{WA}(n+N+1) &= Y_{WA}(n+2) = \sum_{i=1}^{i=2N+1} a_i f(n+i) = \sum_{i=1}^{i=3} a_i f(n+i) = \max_{r_i \in \{0,1,\ldots,r_{max}\}} \Bigg\{ \min \Bigg\{ f(n+1) - r_1, \\ & f(n+2) - r_2, f(n+3) + r_3 \cdot \frac{\sum_{i=1}^{i=2N} a_i}{1 - \sum_{i=1}^{i=2N} a_i} \Bigg\}\Bigg\} = \max_{r_i \in \{0,1,\ldots,r_{max}\}} \Bigg\{ \min \Bigg\{ f(n+1) - r_1, f(n+2) - r_2, f(n+3) + \\ & + r_3 \cdot \frac{1/3 + 1/3}{1 - (1/3 + 1/3)} \Bigg\}\Bigg\} = \max_{r_i \in \{0,1,\ldots,r_{max}\}} \Bigg\{ \min \Bigg\{ f(n+1) - r_1, f(n+2) - r_2, f(n+3) + r_3 \cdot 2 \Bigg\}\Bigg\}. \end{aligned} \tag{13}$$

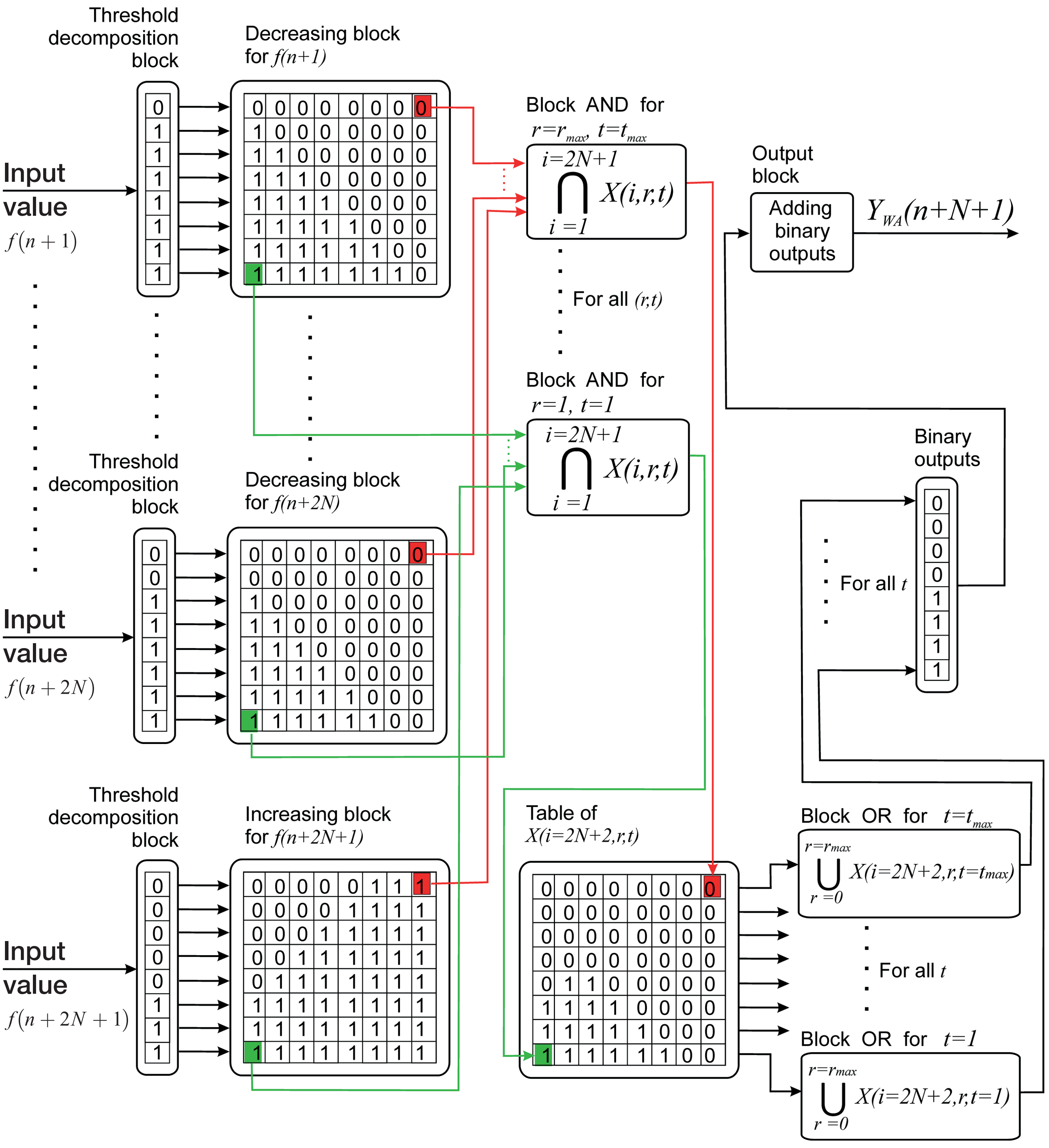
Threshold decomposition block
Decreasing block for f(n+1)
Input value $f(n+1)$
Block AND for $r=r_{max}$, $t=t_{max}$
$\bigcap_{i=1}^{i=2N+1} X(i,r,t)$
For all (r,t)
Block AND for $r=1$, $t=1$
Output block
Adding binary outputs
$Y_{WA}(n+N+1)$
Binary outputs
For all t
Threshold decomposition block
Decreasing block for f(n+2N)
Input value $f(n+2N)$
Threshold decomposition block
Increasing block for f(n+2N+1)
Input value $f(n+2N+1)$
Table of X(i=2N+2,r,t)
Block OR for $t=t_{max}$
$\bigcup_{r=0}^{r=r_{max}} X(i=2N+2,r,t=t_{max})$
For all t
Block OR for $t=1$
$\bigcup_{r=0}^{r=r_{max}} X(i=2N+2,r,t=1)$

Fig. 4. The schematic diagram of the logical system realising the average filter in the form of the stack filter

where r_1, r_2 and r_3 assume the subsequent values 0, 1, …, r_{max}. Therefore to find minimum $\{f(n+1)-r_1, f(n+2)-r_2, f(n+3)+ + r_3\cdot 2\}$ for $r_i \in \{0, 1, \dots, r_{max}\}$ we assume the following values of r_i and for them we obtain the values of minima $M(r_i)$:

$$\begin{aligned}
M(r_i=0) &= \min\Big\{f(n+1), f(n+2), f(n+3)\Big\}\\
M(r_i=1) &= \min\Big\{f(n+1)-1, f(n+2)-1,\\
&\qquad f(n+3)+1\cdot 2\Big\}\\
M(r_i=2) &= \min\Big\{f(n+1)-2, f(n+2)-2,\\
&\qquad f(n+3)+2\cdot 2\Big\}\\
M(r_i=3) &= \min\Big\{f(n+1)-3, f(n+2)-3,\\
&\qquad f(n+3)+3\cdot 2\Big\}\\
&\vdots \qquad\qquad \vdots\\
M(r_i=r_{max}) &= \min\Big\{f(n+1)-r_{max}, f(n+2)-r_{max},\\
&\qquad f(n+3)+r_{max}\cdot 2\Big\}.
\end{aligned} \tag{14}$$

On account of the fact that the domain for all used variables is equal $\{0, 1, \dots, r_{max}\}$, we assume that when $f(n+j)-r_i<0$, then we assume $f(n+j)-r_i=0$.

The same reason causes that when $f(n+3)+r_i\cdot 2>r_{max}$, then we assume $f(n+3)+r_i=r_{max}$.

In this way, we obtain $Y_{WA}(n+2)$ in the following form:

$$Y_{WA}(n+2) = \max_{r_i\in\{0,1,\dots,r_{max}\}}\Big\{M(r_i=0), M(r_i=1), \dots \dots, M(r_i=r_{max})\Big\}. \tag{15}$$

The next example presents the calculation of Y_{WA} for two input values.

Example 2. In this example we assume that we will calculate the weighted average Y_{WA} for two input values with coefficients $a_1=a_2=1/2$. We will use the simplified notation for input values and we assume that $f(n+1)$ and $f(n+2)$ will be denoted as $f(1)=5$ and $f(2)=3$ respectively.

The domain for all non-binary variables is equal $D=\{0, 1, \dots, r_{max}\}$ where $r_{max}=7$. The maximum value of variables in D is equal 7, which causes that the maximum value of the threshold t is equal 7 and $t\in\{1, 2, \dots, 7\}$. Owing to the fact that we calculate weighted average for two inputs, the variable i assumes $2+1$ values: 1, 2 and 3. The variable $X(i, r, t)$ is a block which contains in its cells binary values 0 or 1. The size of $X(i, r, t)$ is equal $3\times 8\times 7$. The tables for $i=1$ and $i=2$ are connected to $f(1)$ and $f(2)$ respectively. The table for $i=3$ is obtained as a result of calcutations in (11). The variables r_1 and r_2 assume subsequent values 0, 1, 2, …, 7. The variable t defines the threshold and it assumes subsequent values 1, 2, …, 7.

According to (8), for the abovementioned assumpions, we obtain the following expression for calculating the weighted average for two arguments.

$$\begin{aligned}
Y_{WA}\big(f(1), f(2)\big) &= a_1 f(1) + a_2 f(2) = \frac{f(1)+f(2)}{2} =\\
&= \max_{r_i\in\{0,1,\dots,r_{max}\}}\left\{\min\left\{f(1)-r_1, f(2)+r_2\cdot\frac{a_1}{1-a_1}\right\}\right\} =\\
&= \max_{r_i\in\{0,1,\dots,7\}}\Big\{\min\Big\{f(1)-r_1, f(2)+r_2\Big\}\Big\}.
\end{aligned} \tag{16}$$

where r_1 and r_2 assume the subsequent values 0, 1, …, r_{max}, $r_{max}=7$.

The procedure for calculating the value of $Y_{WA}\big(f(1), f(2)\big)$ is detailed in the following four steps.

1. Calculating $X(i=1, r, t)$.

The table $X(i=1, r, t)$ includes the element $f(1)-r_1$ from (16). We denote the variable on the axis r for $i=1$ as r_1. For the subsequent values of r_1, we obtain the values of $f(1)-r_1$ presented in (17).

$$\begin{aligned}
f(1)-r_1 &= 5 \quad \text{for} \quad r_1 = 0\\
f(1)-r_1 &= 4 \quad \text{for} \quad r_1 = 1\\
f(1)-r_1 &= 3 \quad \text{for} \quad r_1 = 2\\
f(1)-r_1 &= 2 \quad \text{for} \quad r_1 = 3\\
f(1)-r_1 &= 1 \quad \text{for} \quad r_1 = 4\\
f(1)-r_1 &= 0 \quad \text{for} \quad r_1 = 5\\
f(1)-r_1 &= 0 \quad \text{for} \quad r_1 = 6\\
f(1)-r_1 &= 0 \quad \text{for} \quad r_1 = 7.
\end{aligned} \tag{17}$$

As we can see in (17), for values $f(1)-r_1<0$ we assume $f(1)-r_1=0$. It is caused by the fact that the domain for all used variables is equal $\{0, 1, \dots, 7\}$.

Next, each value of $f(1)-r_1$ is thresholded with the thresholds t equal 1 to 7 and in this way we obtain the binary columns in $X(i=1, r, t)$. The first column corresponds to $f(1)-r_1$ for $r_1=0$, the second column corresponds to $f(1)-r_1$ for $r_1=1$ and so on up to $r_1=7$. It is presented in the Table 1.

We obtain the table $X(i=1, r, t)$ by combining the binary columns from the Table 1.

2. Calculating $X(i=2, r, t)$.

The table $X(i=2, r, t)$ includes the element $f(2)+r_2$ from (16). We denote the variable on the axis r for $i=2$ as r_2. For

Table 1
Thresholded values of $f(1) - r_1$

	Multi-value variables							
$f(1)$	5	5	5	5	5	5	5	5
r_1	0	1	2	3	4	5	6	7
$f(1) - r_1$	5	4	3	2	1	0	0	0
Binary values of thresholded $f(1) - r_1$								
$t = 7$	0	0	0	0	0	0	0	0
$t = 6$	0	0	0	0	0	0	0	0
$t = 5$	1	0	0	0	0	0	0	0
$t = 4$	1	1	0	0	0	0	0	0
$t = 3$	1	1	1	0	0	0	0	0
$t = 2$	1	1	1	1	0	0	0	0
$t = 1$	1	1	1	1	1	0	0	0

Table 2
Thresholded values of $f(2) + r_2$

	Multi-value variables							
$f(2)$	3	3	3	3	3	3	3	3
r_2	0	1	2	3	4	5	6	7
$f(2) + r_2$	3	4	5	6	7	7	7	7
Binary values of thresholded $f(2) - r_2$								
$t = 7$	0	0	0	0	1	1	1	1
$t = 6$	0	0	0	1	1	1	1	1
$t = 5$	0	0	1	1	1	1	1	1
$t = 4$	0	1	1	1	1	1	1	1
$t = 3$	1	1	1	1	1	1	1	1
$t = 2$	1	1	1	1	1	1	1	1
$t = 1$	1	1	1	1	1	1	1	1

the subsequent values of r_2, we obtain values of $f(2) + r_2$ presented in (18).

$$\begin{aligned}
f(2) + r_2 &= 3 \quad \text{for} \quad r_2 = 0\\
f(2) + r_2 &= 4 \quad \text{for} \quad r_2 = 1\\
f(2) + r_2 &= 5 \quad \text{for} \quad r_2 = 2\\
f(2) + r_2 &= 6 \quad \text{for} \quad r_2 = 3\\
f(2) + r_2 &= 7 \quad \text{for} \quad r_2 = 4\\
f(2) + r_2 &= 7 \quad \text{for} \quad r_2 = 5\\
f(2) + r_2 &= 7 \quad \text{for} \quad r_2 = 6\\
f(2) + r_2 &= 7 \quad \text{for} \quad r_2 = 7.
\end{aligned} \tag{18}$$

As we can see in (18), for values $f(2) + r_2 > 7$ we assume $f(2) + r_2 = 7$. It is caused by the fact that the domain for all used variables is equal $\{0, 1, \ldots, 7\}$.

Next, each value of $f(2) + r_2$ is thresholded with the thresholds t equal 1 to 7 and in this way we obtain the binary columns in $X(i = 2, r, t)$. The first column corresponds to $f(2) + r_2$ for $r_2 = 0$, the second column corresponds to $f(2) + r_2$ for $r_2 = 1$ and so on up to $r_2 = 7$. It is presented in the Table 2.

We obtain the table $X(i = 2, r, t)$ by combining the binary columns from the Table 2.

3. Calculating $X(i = 3, r, t)$.

The calculation of $X(i = 3, r, t)$, described in (11), corresponds to the operation *min* from (8). For the example 2, we obtain the form of (11) presented in (19)

$$X(i = 3, r, t) = \bigcap_{i=1}^{i=2} X(i, r, t). \tag{19}$$

4. Calculating Y_{WA}.

The general expression which is used to calculate Y_{WA} in the binary form is presented in (12). For the example 2, the equation (12) assumes a form shown in (20)

$$Y_{WA}(t) = \bigcap_{r=0}^{r=7} X(i = 3, r, t). \tag{20}$$

The way of carrying out these logical operations is presented in Table 3.

The multi-value form of $Y_{WA}(t)$ is obtained as a sum of binary outputs

$$Y_{WA} = \sum_{t=0}^{t=7} Y_{WA}(t) = 4. \tag{21}$$

Figure 3 presents the structure of argument tables in binary box which realises WA filtering for two input values $f(1) = 5$ and $f(2) = 3$ with coefficients $a_1 = a_2 = 1/2$.

Table 3
Calculating Y_{WA} in the binary form

$X(i = 3, r, t)$															binary $Y_{WA}(t)$
r = 0		r = 1		r = 2		r = 3		r = 4		r = 5		r = 6		r = 7	
0	or	0	or	0	or	0	or	0	or	0	or	0	or	0	= 0
0	or	0	or	0	or	0	or	0	or	0	or	0	or	0	= 0
0	or	0	or	0	or	0	or	0	or	0	or	0	or	0	= 0
0	or	1	or	0	or	0	or	0	or	0	or	0	or	0	= 1
1	or	1	or	1	or	0	or	0	or	0	or	0	or	0	= 1
1	or	1	or	1	or	1	or	0	or	0	or	0	or	0	= 1
1	or	1	or	1	or	1	or	1	or	0	or	0	or	0	= 1

The proposed Binary Box Method, applied to calculating weighted average, may be used to realise other signal processing functions in the case when it is necessary to make calculations in two steps. When the stacking property is guaranteed for both steps of calculation, then using BBM allows us to use all assets of stack filters.

4.4. Realising morphological filters as a stack filter. Other filters, realised as stack filters, which are often used for filtering are the morphological filters realising opening and closing operations. The morphological filters can be realised as stack filters with appropriate binary functions as shown in [26] and [27].

In morphological filtering, each input signal is presented as a set, and its geometrical features are modified by the morphological convolution of the signal with a structuring element. This structuring element is another set of a simple shape and size [28]. The shape and size of this element define which pixels of the input image are taken into account for calculating the value of the output pixel for a given window location. By adjusting appropriately the shape and size of the structuring element to the size and shape of the recognised object we can remove or enlarge the selected elements in the image.

According to the approach presented in [26], we may present the formulation of morphological filters which operate on: sets – set-processing filters (SP), functions – function-processing filters (FP) or function-and-set-processing filters (FSP). At the same time, we will consider only the class of upper semicontinuous (u.s.c.) functions.

Morphological filters may be analysed as SP filters which operate on input m-D sets by interacting them via Minkowski set addition or subtraction with structuring elements that are n-D sets $n \leq m$. The Minkowski set addition [29, 30] of the sets $A, B \subseteq R^m$ is the set shown in (22)

$$A \oplus B = \{a + b : a \in A,\ b \in B\} = \bigcup_{b \in B} A_b. \tag{22}$$

where $A_b = \{a + b : a \in A\}$ The Minkowski set subtraction [30] of B from A is the set defined in (23)

$$A \ominus B = \left(A^C \oplus B\right)^C \bigcap_{b \in B} A_b. \tag{23}$$

Let $B^S = \{-b : b \in B\}$ denote the symmetric set of B and $\emptyset$ denotes the empty set. The basic SP morphological filters are the erosion $X \ominus B^S$, opening X_B and closing X^B of X by B, defined in [26] as:

$$X \ominus B^S = \{z : B_z \subseteq X\} = \bigcap_{b \in B} X_{-b} \tag{24}$$

$$X \oplus B^S = \{z : B_z \cap X \neq \emptyset\} = \bigcap_{b \in B} X_{-b} \tag{25}$$

$$X_B = \left(X \ominus B^S\right) \oplus B \tag{26}$$

$$X^B = \left(X \oplus B^S\right) \ominus B. \tag{27}$$

In order to discuss the FSP morphological filters, we have to define herein, according to [26], the cross section term. We consider m-dimensional function $f(x)$ which domain is a subset of the domain space $D = Z^m$ or R^m (Z-integer numbers, R-real numbers). We assume the range of $f(x)$ values as a subset of the range space $V = R$ or Z. Signals can be represented either by functions or by sets. The set is the primary notion which causes that the main issue is to represent functions by sets. It will be conducted in the way described below. An m-D function can be represented by an ensemble of m-D sets called its cross sections. The set

$$X_t(f) = \{x \in D : f(x) \geq t\},\ t \in V \tag{28}$$

is called the cross section of $f(x)$ at the level t and is obtained by thresholding $f(x)$ at the level t. By considering all different levels t, we can associate $f(x)$ with a family of sets. These sets decrease monotonically as t increases.

According to [26], we can present FSP filters. 'm-D u.s.c. function' denotes m-dimensional function which is upper semi-continuous (m-dimensional function, D = Zm or D = Rm). It is defined in [56] and [59] together with the lower semi-continuous functions (1.s.c.). The input signals are m-D u.s.c. functions and the structuring elements are compact n-D sets with $n \leq m$. As presented in [26] they commute with thresholding. As an example we can present the erosion. Let Φ be the SP erosion filter by B then [28] defines an FSP erosion by B as follows. Since Φ is increasing (positive Boolean function) and u.s.c., for any input function f, the set class $\{\Phi[X_t(f)] = X_t(f) \ominus B^S : t \in V\}$ creates an output function h by setting $X_t(h) = \Phi[X_t(f)]$. This output function is the erosion of f by B, denoted by $f \ominus B^S$. Likewise, we can define the dilation $f \oplus B^S$

$$\begin{gathered} X_t\left(f \ominus B^S\right) = \\ = X_t(f) \ominus B^S \Leftrightarrow \left(f \ominus B^S\right)(x) = \inf_{y \in B_x} f(x) \end{gathered} \tag{29}$$

$$\begin{gathered} X_t\left(f \oplus B^S\right) = \\ = X_t(f) \oplus B^S \Leftrightarrow \left(f \oplus B^S\right)(x) = \sup_{y \in B_x} f(x). \end{gathered} \tag{30}$$

In this way, we obtain the erosion (dilation) description using the operation infimum (supremum) which operations for discrete compact closed sets are equal minimum (maximum). Thus, the erosion (dilation) of f by B at any point x is obtained by shifting the set B to location x and taking the minimum (maximum) of f inside this shifted set.

At the same time, it is known that in Boolean algebra calculating the minimum (maximum) for binary variables corresponds to the expression in the form of logical product (sum) of these variables. By using, at each threshold level, the Boolean functions which are product (sum) of variables corresponding to elements of the structural element B we obtain the operation erosion (dilation) in morphological processing.

4.5. Calculation of the gradient as a stack filter. Other filters, realised as stack filters, which we are often used for filtering are the morphological filters realising gradient as opening and closing operations [31]. One knows, that for function $F(x_1, x_2)$ of two variables x_1, x_2 with continuous partial derivatives the gradient $grad(F)$ is defined as 2-D vector presented in (31)

$$\left[\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}\right]. \tag{31}$$

Hence morphological gradient may be defined as [31]:

$$grad(F) = \lim_{r\to 0} \frac{1}{2r}\left[(F \oplus rB) - (F \ominus rB)\right] \tag{32}$$

where B – is the circle with the radius equal 1, r – the real number. For discrete images, it is necessary to formulate gradient term taking into consideration that r cannot be arbitrarily small. Therefore on a discrete plane we use the equation [31]:

$$F\nabla B = \frac{1}{2}\left[(F \oplus B) - (F \ominus B)\right] \tag{33}$$

In (33)[31] $F\nabla B$ corresponds to an absolute value of the gradient from (33)[31] B is a discrete approximation of a unit circle in the form of a square 3×3 pixels. These operations conducted in each threshold level guarantee stacking property. Therefore taking into consideration the abovementioned discussion about morphological filters as a stack filters, it is possible to make calculations of gradient as stack filtering.

The realisation of data processing in the form of stack filters results in shorter processing time which constitutes a crucial problem in many signal processing tasks, especially in many image processing tasks.

5. Conclusions

The presented paper focused on the method of refining signal processing by shortening the time of calculations. We proposed a modification bringing in realising various functions in the form of stack filters. The selected functions represent both linear and non-linear classes of functions. The examples of functions which were presented in the paper were selected because of their large usefulness in the image processing systems. These systems are ones in which the time of calculation is often the critical parameter and it decides about the usefulness of the proposed solution.

The paper collected and presented the methods of selected functions transformation into form of stack filter for the following functions: median, opening and closing morphological operations and gradient operation. In the body of literature, these functions are well-known and widely used for image processing.

The problems which appeared during the synthesis of the morphological filters as stack filters led to developing a novel method for the stack filter synthesis. This method was called by the authors the Binary Box Method (BBM). It is a two-stage method which allows us to present in the form of stack filter the more complicated mathematical operations. What we mean by "complicated" is the complex representation of a given function in the form of a stack filter. In this paper, the application of BBM was presented for the average filter. The only condition for the BBM which has to be met is the stacking property at all stages of the converted algorithm.

The developed method was proposed for the limited class of functions because of stack filters properties. The functions which we intend to convert have to satisfy stacking property requirement at each step of signal processing.

The proposed approach allows us to convert well-known signal processing algorithms into realisation which guarantees significantly greater speeds of signal processing.

References

[1] M. Sonka, V. Hlavac, and R. Boyle: Image Processing, Analysis and Machina Vision, 3rd ed. Thompson, (2008).

[2] Z. Kuś, "Dynamical Pattern Vector in Pattern Recognition with the Use of Thermal Images", 8th International Conference on Computer and Automation Engineering (ICCAE 2016), MATEC Web of Conferences, Volume 56, (2016).

[3] Z. Kuś, "The Fusion of the Visual and Thermal Images on the Basis of Determining the Image Fragments which Contain Essential Details", 8th International Conference on Computer and Automation Engineering (ICCAE 2016), MATECWeb of Conferences ,Volume 56, (2016).

[4] Z. Kuś and A. Nawrat, "The Method of Developing the Invariant Functions Vector for Objects Recognition from a Given Objects Set", Innovative Simulation Systems, Eds. A. Nawrat, K. Jedrasiak, ISBN 9783319211176 9783319211183, Springer, (2016).

[5] Z. Kuś and A. Nawrat, "The Method of Guaranteeing the Separa tion between the Recognised Object and Background", Innovative Simulation Systems, Eds. A. Nawrat, K. Jedrasiak, ISBN 9783319211176 9783319211183, Springer, (2016).

[6] Z. Kuś and A. Nawrat, "Minimizing the Image Resolution in order to Increase the Computing Speed without Losing the Separation of the Recognised Patterns", Innovative Simulation Systems, Eds. A. Nawrat, K. Jedrasiak, ISBN 9783319211176 9783319211183, Springer, (2016).

[7] Z. Kuś and A. Nawrat, "Adjusting the Thresholds to the Recognised Pattern in order to Improve the Separation Between the Recognised Patterns", Innovative Simulation Systems, Eds. A. Nawrat, K. Jedrasiak, ISBN 9783319211176 9783319211183, Springer, (2016).

[8] Z. Kuś, "The Analysis of Dynamical Properties of the Object Tracking System's Elements", *Bull. Pol. Ac.: Tech.* 64 (3), 479–489, ISSN (Online) 2300–1917, DOI: https://doi.org/10.1515/bpasts-2016–0053, September 2016

[9] Z. Kuś and A. Nawrat, "Camera head control system with a changeable gain in a proportional regulator for object tracking", Innovative control systems for tracked vehicle platforms. Ed. Aleksander Nawrat. Cham : Springer, (2014), (Studies in Systems, Decision and Control ; Vol. 2 2198–4182) ISBN: 978–3–319–04623–5 (Print) 978–3–319–04624–2 (Online) DOI 10.1007/978–3–319–04624–2

[10] Z. Kuś and A. Nawrat, "The limitation for the angular velocity of the camera head during object tracking with the use of the UAV", Innovative control systems for tracked vehicle platforms. Ed. Aleksander Nawrat. Cham : Springer, (2014), (Studies in Systems, Decision and Control ; Vol. 2 2198–4182) ISBN: 978–3–319–04623–5 (Print) 978–3–319–04624–2 (Online) DOI 10.1007/978–3–319–04624–2

[11] M. Blachuta and R. Grygiel, "Are anti-aliasing filters really necessary for sampled-data control?", American Control Conference, St Louis, 2009 American Control Conference, VOLS 1–9 Book Series: Proc. of the American Control Conf. Pages: 3200–3205, Jun 10–12, (2009).

[12] M. Blachuta and R. Grygiel, "On the Effect of Anti-aliasing Filters on Sampled-Data PID Control", 21st Chinese Control and Decision Conference, Guilin, Peoples R China, Jun 17–19, (2009).

[13] R. Bieda, M. Blachuta, and R. Grygiel, "High Performance PID Control of a Cascade Tanks System as an Example for Control Teaching", International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, Aug. 28-SEP 31, (2017).

[14] M. Luszczkiewicz-Piatek, "Which colour space should be chosen for robust colour image retrieval based on mixture modelling", Image Processing and Communications Challenges 5 (IP&C 2013), Series: Advances in Intelligent Systems and Computing, vol. 233, pp. 55–64, Springer International Publishing, (2014).

[15] M. Luszczkiewicz-Piatek and B. Smolka, "Robust image retrieval based on mixture modelling of weighted spatio-colour information", Image Processing and Communications Challenges 6 (IP&C 2014), Series: Advances in Intelligent Systems and Computing, vol. 313, pp. 85–93, Springer International Publishing, (2015).

[16] M. Luszczkiewicz-Piatek, "Image Similarity in Gaussian Mixture Model Based Image Retrieval", Image Processing and Communications Challenges 7 (IP&C 2015), Series: Advances in Intelligent Systems and Computing, vol. 389, pp. 87–95, Springer International Publishing, (2016).

[17] J.W. Tukey, "Nonseparable (nonsuperposable) methods for smoothing data", in Conf. Rec., EASCON, 1974, p. 673, and Exploratory Datu Analysis. Reading, MA: Addison-Wesley, (1977).

[18] N.C. Gallagher, Jr. and G.L. Wise, "A theoretical analysis of the properties of median filters", *IEEE Trans. Acoust. , Speech, Signal Processing*, vol. ASSP-29, pp. 1136–1141, Dec. (1981).

[19] S.G. Tyan, "Median-filtering: Deterministic properties", in Two- Dimensional Digital Signal Processing, II: Transforms and Median Filters, ch. 5. vol. 42, pp. 197–217, Topics in Applied Physics, T. S . Huang, Ed. New York: Springer-Verlag. (1981).

[20] T. Nodes and N.C. Gallagher. , "Median filters: Some modifications and their properties", *IEEE Trans. Acoust. , Speech, Signal Processing*, vol. ASSP-30, pp, 739–746. Oct. (1982).

[21] G. Matheron: Random Sets and Integral Geometry, New York: Wiley, 1975.

[22] P.D.Wendt, E.J. Coyle, and N.C. Gallagher. Jr., "Stack filters", *IEEE Trans. Acoust. Speech. Signal Processing*, vol. ASSP-34, pp. 898–911 , Aug. (1986).

[23] M. Gabbouj, "Optimal stack filter examples and positive Boolean functions", M.S. thesis, School Elec. Eng. Purdue Univ., West Lafayette, IN. Dec. (1986).

[24] J.H. Lin and E. Coyle, "Minimum Mean Absolute Error Estimation over the Class of Generalized Stack Filters", *IEEE Transactions On Acoustics, Speech And Signal Processing*, vol. 38, No. 4, April (1990).

[25] B. Zeng, M. Gabbouj, and Y. Neuvo, "Design of minimum MAE generalized stack filters for image processing", SPIE vol. 1606 Visual Communications and Image Processing '91: Image Processing, (1991).

[26] P. Maragos and R.W. Schafer, "Morphological filters–Part I: Their set-theoretic analysis and relations to linear shift-invariant filters", *Acoustics, Speech and Signal Processing, IEEE Transactions on*, Volume: 35, Issue: 8, August (1987).

[27] P. Maragos and R. W. Schafer, "Morphological filters–Part II: Their relations to median, order-statistic, and stack filters", *Acoustics, Speech and Signal Processing, IEEE Transactions on*, Volume: 35, Issue: 8, August (1987).

[28] I. Serra: Image Analysis and Mathematical Morphology, New York: Academic, (1982).

[29] H. Minkowski, "Volumen und Oberflache", *Math. Annalen*, vol. 57, pp.447–495, (1903).

[30] H. Hadwiger: Vorlesungen uber Inhalt, Oberflache und Isoperimetrie, Berlin, Germany: Springer-Verlag, (1957).

[31] M. Nieniewski: Mathematical Morphology for Image Processing, in Polish, AOW, Warszawa, (1998).

[32] M. Mondelli, "A Finite Difference Scheme for the Stack Filter Simulating the MCM", *Image Processing On Line*, ISSN 2105–1232 c 2013 IPOL & the authors CC-BY-NC-SA, (2013–07–11).

[33] T. Suzuki, Y. Hanada, and M. Muneyasu, "Unsupervised design of stack filters by tree structure optimization", 8th International Conference on Information, Communications & Signal Processing, (2011).

[34] K. Yan, Y. Hongwei, and F. Jiayin, "Design of optimal stack filters using QPSO", The 2nd International Conference on Information Science and Engineering, (2010).

[35] A. Frias-Velazquez and W. Philips, "Bit-plane stack filter algorithm for focal plane processors", IEEE International Conference on Image Processing, (2010).

[36] Y. Tian, Ch. Zhao, "Design of Optimizing Stack Filters by an Ant Colony – Clonal Selection Algorithm", Chinese Conference on Pattern Recognition, (2009).

[37] M. E. Buemi, M. Mejail, J. Jacobo, J. Gambini, "Improvement in SAR Image Classification using Adaptive Stack Filters", XX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2007). (2007).

[38] D. Dellamonica, P.J.S. Silva, C. Humes, N.S.T. Hirata, and J. Barrera, "An Exact Algorithm for Optimal MAE Stack Filter Design", *IEEE Transactions on Image Processing*, Volume: 16, Issue: 2, Pages: 453 – 462, August (1987).

[39] Ch. Zhao, Ch. Zhang, H. Ning, and Y. Cui, "Optimizing Stack Filters by Clone Selection Algorithm", 8th international Conference on Signal Processing, Volume: 1, (2006).

[40] G. Shi, W. Dong, and Z. Liu, "Design and implementation of stack filter based on immune memory clonal algorithms with hybrid computation", 48th Midwest Symposium on Circuits and Systems, (2005).

[41] M.K. Prasad, "Stack filter design using selection probabilities", *IEEE Transactions on Signal Processing*, Volume: 53, Issue: 3 Pages: 1025–1037, (2005).
[42] D. Diaz and J.L. Paredes, "FPGA implementation of a new family of stack filters", Proceedings of the Fifth IEEE International Caracas Conference on Devices, Circuits and Systems, Volume: 1, Pages: 152–157, (2004).
[43] S. Guangming, S. Liya, L. Honghua, and H. Daojun, "Dynamic nonlinear threshold decomposition algorithm for implementing stack filters", IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512) Volume:3, Pages: III – 641–4, Vol. 3, (2004).
[44] J. Huang and E.J. Coyle, "Using models of the Human Visual System in the design of stack filters for the enhancement of color images", 10th European Signal Processing Conference, (2000).
[45] C. Toumazou, N. Battersby, and S. Porta, "Stack Filters in Signal and Image Processing", Circuits and Systems Tutorials, Pages: 22 – 39, Wiley-IEEE Press eBook Chapters, (1996).
[46] D. Gevorkian, M. Hu, O. Vainio, and J. Astola, "VLSI architecture for stack filters", Digital Signal Processing Proceedings, 1997. DSP 97., 13th International Conference on, (1997).
[47] J. Astola, S. Agaian, K. Egiazarian, O. Vainio, and D. Gevorkian, "High-speed algorithms and an architecture for running stack filters", Proceedings of Int. Conf. on Digital Signal Processing, (1995).
[48] A. Hiasat, O. Hasan, "Bit-serial architecture for rank order and stack filters", *Integration, the VLSI Journal archive*, Volume 36 Issue 1–2,Pages 3–12, (2003).
[49] T. Yamamoto and V.G. Moshnyaga, "A New Bit-Serial Architecture of Rank-Order Filter", Conference Paper in Midwest Symposium on Circuits and Systems, (2009).
[50] M.J. Avedillo, J.M. Quintana, H. Alami, and A. Jimenez-Calderon, "A Practical Parallel Architecture for Stacks Filters", *Journal of VLSI signal processing systems for signal, image and video technology*, Volume 38, Issue 2, pp 91–100, (2004).
[51] C. Chakrabarti, "Efficient stack filter implementations of rank order filters", Circuits and Systems, ISCAS '93, IEEE International Symposium on, (1993).
[52] C. Chakrabarti and L. Lucke, "VLSI Architectures for Weighted order Statistic (WOS) Filters", Circuits and Systems, ISCAS '98. Proceedings of the 1998 IEEE International Symposium on, (1998).
[53] K. Chen, "Bit-Serial Realization of a Class of Nonlinear Filters Based on Positive Boolean Functions", *IEEE Transactions on Circuits and Systems*, Volume: 36, Issue: 6, Pages: 785–794, (Jun 1989).
[54] D.Z. Gevorkian, K.O. Egiazarian, S.S. Agaian, J.T. Astola, and O. Vainio, "Parallel algorithms and VLSI architectures for stack filtering using Fibonacci p-codes", *IEEE Transactions on Signal Processing*, Volume: 43, Issue: 1, Page(s): 286 – 295, Jan (1995).
[55] J. Astola, D. Akopian, O. Vainio, and S. Agaian, "New digit-serial implementations of stack filters", *Signal Processing*, Volume 61, Issue 2, Pages 181–197, September (1997).
[56] J.P. Fitch, "Software and VLSI algorithms for generalized ranked order filtering", *IEEE Transactions on Circuits and Systems*, Volume: 34, Issue: 5, Page(s): 553 – 559, May (1987).
[57] G.B. Adams, E.J. Coyle, L. Lin, L.E. Lucke, K.K. Parhi, "Input compression and efficient VLSI architectures for rank order and stack filters", *Signal Processing*, Volume 38, Issue 3, Pages 441–453, August (1994).
[58] R.G. Bartle: The Elements of Real Analysis, New York: Wiley, (1968).
[59] H.L. Royden: Real Analysis, New York: Macmillan Library Reference, ISBN 10: 0024041505 ISBN 13: 9780024041500 (1968).
[60] A. Kordecki, A. Bal, and H. Palus, "A smooth local polynomial model of vignetting", 22nd International Conference on Methods and Models in Automation and Robotics (MMAR), DOI: 10.1109/MMAR.2017.8046944, August (2017).
[61] A. Kordecki, A. Bal, and H. Palus, "Fast vignetting reduction method", 20th International Conference on Methods and Models in Automation and Robotics (MMAR), At Miedzyzroje, Poland, DOI: 10.1109/MMAR.2015.7284040, August (2015).
[62] A. Kordecki, H. Palus , and A. Bal, "Practical vignetting correction method for digital camera with measurement of surface luminance distribution", Signal Image and Video Processing 10(8), DOI: 10.1007/s11760–016–0941–2, License: CC BY 4.0, July (2016).