Efficient approach for OS-CFAR 2D technique using distributive histograms and breakdown point optimal concept applied to acoustic images

Sebastián A. Villar1,2*, Bruno V. Menna1, Sebastián Torcida2, Gerardo G. Acosta1
1INTELYMEC Group (UNCPBA) and CIFICEN (UNCPBA-CICPBA-CONICET), Av. del Valle 5737, B7400JWI Olavarría, Argentina
2Departamento de Matemática, Facultad de Ciencias Exactas (Campus), UNCPBA, Tandil, Argentina
E-mail: svillar@fio.unicen.edu.ar

Abstract: In this work, a new approach to improve the algorithmic efficiency of the order statistic-constant false alarm rate (OS-CFAR) applied in two dimensions (2D) is presented. OS-CFAR is widely used in radar technology for detecting moving objects as well as in sonar technology for the relevant areas of segmentation and multi-target detection on the seafloor. OS-CFAR rank orders the samples obtained from a sliding window around a test cell to select a representative sample that is used to calculate an adaptive detection threshold maintaining a false alarm probability. Then, the test cell is evaluated to determine the presence or absence of a target based on the calculated threshold. The rank orders allow that OS-CFAR technique to be more robust to the presence of the speckle noise, but requires higher computational effort. This is the bottleneck of the technique. Consequently, the contribution of this work is to improve the OS-CFAR 2D on-line computation with the distributive histograms and the optimal breakdown point optimal concept, mainly from the standpoint of efficient computation. The theoretical algorithm analysis is presented to demonstrate the improvement of this approach. Also, this novel efficient OS-CFAR 2D was contrasted experimentally on acoustic images.

1 Introduction

The underwater target detection from acoustic images represents a typical process required in different automatic applications such as archaeology, resources search, inspection and maintenance of pipelines, mines or waste detection, and other types of monitoring [1, 2]. Many approaches are currently available in an acoustic domain such as multi-fractal analysis [3], Markov random field (MRF) [4], local Fourier histograms [5], active contours [6], Gauss–MRF model [7], undecimated discrete wavelet transform [8], among others. These approaches require computationally expensive mathematical models to underwater target detection. On the other hand, the constant false alarm rate (CFAR) represents an adaptive technique able to perform accurate and robust target detection. This technique is commonly used in radar technology for detecting moving objects [9, 10] as well as in sonar technology applied in acoustic images from different sonar devices for multi-target detection [11–13], underwater pipeline detection on the seafloor [14, 15], acoustic segmentation of several types of regions [16], among others. CFAR calculates an adaptive detection threshold from interference power values to maintain an expected false alarm probability [9]. In the literature, there are numerous CFAR techniques such as cell averaging CFAR, order statistic CFAR (OS-CFAR), greatest of CFAR, smallest of CFAR, censored mean-level detector CFAR, trimmed mean CFAR, and other variations [9, 17–19].

Focusing on the OS-CFAR, for each test cell, this technique calculates the presence or absence of a target sorting the samples from a sliding window to select a representative sample that is used to calculate an adaptive detection threshold maintaining a false alarm probability [9]. The representative sample could be selected by setting a fixed-order statistics threshold, as suggested in [20], or automatically estimated, based on the application of the information theoretic criteria principle which does not require any prior information about the number of interfering targets [21].

In radar, OS-CFAR can maintain the robust performance of clutter suppression and does not suffer large detection loss in a non-stationary and non-uniform distribution clutter environment. Besides, OS-CFAR is suitable for being used in multi-target situations because of its high resolution [17–19, 22]. In sonar, OS-CFAR demonstrated to be more robust in multi-target situations and less sensitive to the presence of the speckle noise [13–16]. The main drawback of OS-CFAR is its computational effort because sorting is a time-consuming task. This computational effort prevents its use in real-time applications [11], and therefore the utility of OS-CFAR technique decreases. In addition, a two-dimensional (2D) sliding window is necessary to consider more contextual information and hence to improve detections. In this case, the computational effort increases considerably.

The bottleneck of OS-CFAR is the sorting problem. Many authors in the literature have tried to approach this sorting problem. A rather thorough comparison is offered in [23, 24], where the efficiency of several methods is reviewed in the worst case for a 1D array of \( N \) elements: insertion, with a \( O(N^2) \); selection, which is also \( O(N^2) \); bubble sort, \( O(N^2) \); bucket sort, decreasing complexity to \( O(N) \) when the distribution of elements is assumed constant, quick sort \( O(N \log N) \), merge sort \( O(N \log N) \), just to mention the most common approaches [25]. On the other hand, in the digital image processing context, numerous advances on the median filter (MF) computation have been made. The process of applying the MF is a non-linear smoothing one, best known for reducing impulsive or salt-and-pepper noise from a digital image while respecting its edges [26]. Briefly, the MF sliding-window visits each image element and places its centre on it. The intensity values within the window of radius size \( r \) are sorted, and the median intensity value is then used to replace the window’s centre in the filtered image. A rather thorough comparison of MF is offered in [25] using the classic sorting methods for a 1D array. The main reference of MF for a 2D array is the Huang et al. method [27], which was the first exhibiting in the worst case an \( O(r) \) per pixel algorithmic complexity (where \( r \) denotes the radius for a 2D array) using a single histogram. Different approaches have since been tried to break this linearity: the Weiss method [28] uses hierarchical histograms to reach an \( O(\log r) \) per pixel algorithmic complexity but losing simplicity, and the Gil and Werman method [29] has an \( O(\log^2 r) \) per pixel algorithmic complexity and it is...
The target detection problem in the acoustic image consists of detecting a target from the background and echoes of a target. Detection is usually done through analysing each echo signal with the purpose of detecting the presence or absence of a target. Detection is usually done through the contextual information analysis of each echo signal. In [12], two hypotheses were defined for this analysis: (i) the echo signal is the background (\(H_0\)), and (ii) the echo signal is a combination of background and echoes of a target (\(H_1\)).

If the detection system decides that \(H_0\) is validated (target is not present), then hypothesis \(H_0\) is stated. Otherwise, if the detection system decides that \(H_1\) is validated, then hypothesis \(H_1\) is stated, meaning that a target is present. Then the \(k\)th order statistic value \(x_{ki}\) is selected as a representative of the echo signal and a detection threshold \(T\) is estimated applying a scale factor \(\alpha_{OS}\)

\[
\hat{T} = \alpha_{OS} x_{k_i}.
\]

This scale factor \(\alpha_{OS}\) is a constant value determined from false alarm probability \(P_{fa}\). As OS-CFAR keeps on a constant false alarm probability, this detection threshold only varies depending on \(x_{ki}\) value. Therefore, this technique considers the contextual information of each cell under test \(x_{ki}\) to determine the adaptive detection threshold. Then, for each test cell, the detector system makes a decision according to the following decision strategy: For further details about this, please refer to [12, 15, 16]

\[
H_1 \quad x_{ki} \geq T,
H_0 \quad x_{ki} < T.
\]

### 3 Proposal for an efficient OS-CFAR 2D approach

The bottleneck of the OS-CFAR 2D technique is found in the sorting and selection stage of the \(k\)th order statistic value. To solve the sorting problem, in the literature, there are numerous approaches for 1D or 2D array that are classified based on simplicity, algorithmic complexity and objective computational metrics [23, 24, 27–30]. Among all these methods, the ones proposed in [27, 30] are of special interest as they claim to be the lowest algorithmic complexity to date. On the other hand, the selection of the \(k\)th order statistic value based on methods proposed in [27, 30] can be improved significantly using the optimal breakdown point [31]. These methods are really useful to enhance the algorithmic efficiency of the OS-CFAR 2D technique.

Fig. 1 shows the proposed architecture of the efficient OS-CFAR 2D approach using distributive histograms. A new approach to improve the algorithmic efficiency of the OS-CFAR 2D using distributive histograms and the breakdown point optimal concept is presented. This efficiency improvement is demonstrated performing a theoretical algorithm analysis, as well as with some final comments in Section 6.
The \( k \)th order statistic value \( x_i \) is selected using the optimal breakdown point described in Section 3.2, to estimate the threshold \( T \). The value of the test cell \( x_{ij} \) is compared with estimated threshold \( T \) to determine if the target is present. Finally, in Section 3.3, the complete procedure for a particular pixel of an image based on a flowchart is detailed.

### 3.1 Distributed histograms

To understand the sorting method proposed for OS-CFAR 2D, first consider the alternative of using a single kernel histogram \( H \) to store and update all the values from the current sliding window [27]. The kernel histogram \( H \) expresses a 1D array of size \( b \) that stores the frequencies \( f \) of the pixel values of the current sliding window (where \( f_0,\ldots,f_b \) represents the number of 0, \( \ldots, \) \( b \) pixel values). The computing of the \( k \)th order statistic value \( x_k \) is done by accumulating frequencies \( f \) of kernel histogram \( H \) from the top of the scale and stopping when the cumulative sum reaches the boundary \( kN \). The kernel histogram \( H \) is updated with new values as the sliding window scrolls through the image. Fig. 2 shows an example of a 2D array \( x \) representing an input image of size \( A \times B \) (rows by columns) with a squared window of radius equal to 2. When the window’s centre shifts one pixel to the right, e.g. from \( x_{i, j} \) to \( x_{i+1, j} \), two updates are performed: (a) the rightmost column histogram \( H \) is updated by adding removing and adding on top \( x_{i,b} \) and on bottom \( x_{i,a} \) pixel value, respectively; (b) the kernel histogram \( H \) is updated removing the leftmost column histogram \( H \) and adding the rightmost column histogram \( H \). Note that, in the update step (a), the pixel values of other rows are retained in the column histogram \( H \). In the same way, the computation of the \( k \)th order statistic value \( x_k \) is done by accumulating frequencies \( f \) of kernel histogram \( H \) from the top of the scale and stopping when the cumulative sum reaches the boundary \( kN \). The addition and subtraction of column histograms only depend on the number of histogram bins \( b \), itself a function of the image bit-depth.

In this way, to maintain the frequencies of cell values \( f_a,\ldots,f_b \) between rows and columns, one histogram \( h \) is updated removing and adding another pixel value \( x_{i,a} \), and to use the additive property of histograms [28]. This property establishes that the union of two sets of histograms \( R_1 \) and \( R_2 \) is simply the addition of their respective histograms

\[
H(R_1 \cup R_2) = H(R_1) + H(R_2).
\]  

(3)

The \( k \)th order statistic value \( x_k \) is selected using the optimal breakdown point described in Section 3.2, to estimate the threshold \( T \) applying a scale factor \( \alpha_{OS} \). Then, the value of the test cell \( x_{ij} \) is compared with estimated threshold \( T \) to determine if the target is present. Finally, in Section 3.3, the complete procedure for a particular pixel of an image based on a flowchart is detailed.

### 3.1 Distributed histograms

To understand the sorting method proposed for OS-CFAR 2D, first consider the alternative of using a single kernel histogram \( H \) to store and update all the values from the current sliding window [27]. The kernel histogram \( H \) expresses a 1D array of size \( b \) that stores the frequencies \( f \) of the pixel values of the current sliding window (where \( f_0,\ldots,f_b \) represents the number of 0, \( \ldots, \) \( b \) pixel values). The computing of the \( k \)th order statistic value \( x_k \) is done by accumulating frequencies \( f \) of kernel histogram \( H \) from the top of the scale and stopping when the cumulative sum reaches the boundary \( kN \). The kernel histogram \( H \) is updated with new values as the sliding window scrolls through the image. Fig. 2 shows an example of a 2D array \( x \) representing an input image of size \( A \times B \) (rows by columns) with a squared window of radius equal to 2. When the window’s centre shifts one pixel to the right, e.g. from \( x_{i, j} \) to \( x_{i+1, j} \), two updates are performed: (a) the rightmost column histogram \( H \) is updated by adding removing and adding on top \( x_{i,b} \) and on bottom \( x_{i,a} \) pixel value, respectively; (b) the kernel histogram \( H \) is updated removing the leftmost column histogram \( H \) and adding the rightmost column histogram \( H \). Note that, in the update step (a), the pixel values of other rows are retained in the column histogram \( H \). In the same way, the computation of the \( k \)th order statistic value \( x_k \) is done by accumulating frequencies \( f \) of kernel histogram \( H \) from the top of the scale and stopping when the cumulative sum reaches the boundary \( kN \). The addition and subtraction of column histograms only depend on the number of histogram bins \( b \), itself a function of the image bit-depth.

In this way, to maintain the frequencies of cell values \( f_a,\ldots,f_b \) between rows and columns, one histogram \( h \) is updated removing and adding another pixel value \( x_{i,a} \), and to use the additive property of histograms [28]. This property establishes that the union of two sets of histograms \( R_1 \) and \( R_2 \) is simply the addition of their respective histograms

\[
H(R_1 \cup R_2) = H(R_1) + H(R_2).
\]  

(3)

The \( k \)th order statistic value \( x_k \) is selected using the optimal breakdown point described in Section 3.2, to estimate the threshold \( T \). The value of the test cell \( x_{ij} \) is compared with estimated threshold \( T \) to determine if the target is present. Finally, in Section 3.3, the complete procedure for a particular pixel of an image based on a flowchart is detailed.

### 3.1 Distributed histograms

To understand the sorting method proposed for OS-CFAR 2D, first consider the alternative of using a single kernel histogram \( H \) to store and update all the values from the current sliding window [27]. The kernel histogram \( H \) expresses a 1D array of size \( b \) that stores the frequencies \( f \) of the pixel values of the current sliding window (where \( f_0,\ldots,f_b \) represents the number of 0, \( \ldots, \) \( b \) pixel values). The computing of the \( k \)th order statistic value \( x_k \) is done by accumulating frequencies \( f \) of kernel histogram \( H \) from the top of the scale and stopping when the cumulative sum reaches the boundary \( kN \). The kernel histogram \( H \) is updated with new values as the sliding window scrolls through the image. Fig. 2 shows an example of a 2D array \( x \) representing an input image of size \( A \times B \) (rows by columns) with a squared window of radius equal to 2. When the window’s centre shifts one pixel to the right, e.g. from \( x_{i, j} \) to \( x_{i+1, j} \), two updates are performed: (a) the rightmost column histogram \( H \) is updated by adding removing and adding on top \( x_{i,b} \) and on bottom \( x_{i,a} \) pixel value, respectively; (b) the kernel histogram \( H \) is updated removing the leftmost column histogram \( H \) and adding the rightmost column histogram \( H \). Note that, in the update step (a), the pixel values of other rows are retained in the column histograms \( h \). In the same way, the computation of the \( k \)th order statistic value \( x_k \) is done by accumulating frequencies \( f \) of kernel histogram \( H \) from the top of the scale and stopping when the cumulative sum reaches the boundary \( kN \). The addition and subtraction of column histograms only depend on the number of histogram bins \( b \), itself a function of the image bit-depth.
3.2 Breakdown point concept
Since OS-CFAR 2D uses a sorting method for selection of the kth order statistic value \( x_k \), this alternative is hence highly more reliable than the classical mean or average as utilised by other CFAR methods [9, 17–19]. In fact, the mean of a set of numbers changes when at least one of those numbers is changed or replaced. However, when using a sorting method, the result of selecting an order statistical value \( x_k \) does not vary substantially when those numbers are changed or replaced, always depending on the statistical order \( k \) established, e.g. for \( k = 0.5 \) assuming that \( n \) (odd) ordered numbers are at hand; if the \( \lfloor (n + 1)/2 \rfloor \) lowest numbers are replaced by other arbitrary numbers but keeping them below the \( (n + 1)/2 \) ranked number (the median), the new median will remain the same [31]. In general terms, the breakdown point concept denotes the percentage of data in a set that could be arbitrarily replaced without grossly modifying the value of estimation or computation [32]. Clearly, the mean has a 0% breakdown point, the median has almost a 50% breakdown point, and the other kth order statistic has almost a min \( (k; 1 - k/\% \) breakdown point.

Each time the sliding window shifts some of its values are removed and simultaneously replaced by new values, and a new kth order statistic is computed. The kth order statistic value computation typically uses a bottom-up or top-down accumulating strategy (e.g. for an 8-bit greyscale where values range from 0 to 255 or 255 to 0 the frequencies are accumulated) until the cumulative sum reaches the boundary \( k n_j \). Thereby, if the sliding window eventually processes an image region with most values near the top of the grey scale, the algorithm will get slower (an analogous problem would take place if frequencies were accumulated top-down and image regions with most of the low pixel values were eventually found). Besides, it seems rather inefficient not to take into account that successive windows share most of their values and thus resulting in similar kth order statistic. More precisely: the proportion of shared values between consecutive sliding windows of radius \( r \) is essentially \( |2r + 1| - 2(2r + 1) / (2r + 1)^2 \). The percentage of shared information between successive windows thus increases really fast with the radius: a 33% of shared values for a radius \( r = 1 \); a 60% of shared values for a radius \( r = 2 \), a 90% of shared values for a radius \( r = 10 \) and so on [31].

These inefficiencies can be overcome by making the most of the optimal breakdown point. In this way, the kth order statistic value from a new sliding window can be computed significantly faster by retaining the kth order statistic value from the previously processed window and updating it; in turn, this strategy enables a more efficient processing of those image regions with values in any extreme of the scale handling equally both cases. For this, it is necessary to consider the previous kth order statistic value \( P_{X_k} \), lower values of \( L_{X_k} \) than \( P_{X_k} \) and greater or equal values of \( G_{X_k} \) than \( P_{X_k} \). When the sliding window scrolls through the image, each new value that is added to the kernel histogram \( H \) is compared with \( P_{X_k} \); if the value is greater or equal than \( P_{X_k} \) or \( G_{X_k} \) is incremented in one and otherwise \( L_{X_k} \) is incremented in one. Also, each old value that is removed of the kernel histogram \( H \) is compared with \( P_{X_k} \); if the value is greater or equal than \( P_{X_k} \) or \( G_{X_k} \) is decremented in one and otherwise \( L_{X_k} \) is decremented in one. To calculate the new kth order statistic value, first it is needed to compute the threshold \( T_k \) based on the kth order statistic using \( T = k(2r + 1)^2 \); next, the threshold \( T_k \) is compared with \( L_{X_k} \); if \( T_k \leq L_{X_k} \), the new kth order statistic value will be lower than the previous one and it will be found moving downward from the current \( P_{X_k} \) bin in the kernel histogram \( H \); otherwise, the new kth order statistic value will be greater than the previous one and will be found moving upward from the current \( P_{X_k} \) bin in \( H \). The auxiliary variables \( L_{X_k} \) and \( G_{X_k} \) are accordingly updated in the process.

To further understand this method considers an example: a sliding window \( W \) of radius \( r = 1 \) with \( (2r + 1)^2 = 9 \) pixel values is given; then \( T_k = k(2r + 1)^2 + 1 = 7 \) with \( k = 0.75 \). Assume that \( W = [1, 3, 3, 4, 5, 7, 7, 7, 9] \) are current window’s ordered values, so \( P_{X_k} = 7, L_{X_k} = 6 \) and \( G_{X_k} = 3 \). Assume next that new values 1, 1 and 1 are added to \( W \) while old values 5, 5 and 1 are removed from it. This update of \( W \) results in \( W = [1, 1, 1, 3, 4, 7, 7, 9] \), which in turn updates \( L_{X_k} = 6 \) and \( G_{X_k} = 3 \) following the comparison with \( P_{X_k} = 7 \). Since \( T_k = L_{X_k} \) the kth order statistic value and no bin-displacement through \( H \) is required. On the other hand, consider that new values 10, 9 and 8 are added to \( W \) while old values 1, 1 and 1 are removed from it. This update of \( W \) results in \( W = [3, 3, 4, 7, 7, 8, 9, 9, 10] \) which in turn updates \( L_{X_k} = 3 \) and \( G_{X_k} = 6 \) following the comparison with \( P_{X_k} = 7 \). Since \( T_k = 7 > 3 = L_{X_k} \), the new kth order statistic value will be found moving forward from the \( P_{X_k} = 7 \) bin in the kernel histogram \( H \). Here, only three bin-displacements through \( H \) are needed to obtain the new kth order statistic value; the auxiliary variables are accordingly updated to \( P_{X_k} = 9, L_{X_k} = 6 \) and \( G_{X_k} = 3 \), respectively. Since \( T_k \) is now smaller than \( L_{X_k} \), the displacement will next start from \( P_{X_k} \) and backward.

3.3 Complete procedures of efficient OS-CFAR 2D approach
Fig. 4 shows the flowchart for the complete procedure applied to each pixel value \( x_{ij} \) (ith row and jth column) of an image \( X \) of size \( A \times B \) using the OS-CFAR 2D presented. Note that previously it must have created and initialised the kernel histogram \( H \), column histograms \( h(1 \leq j \leq B) \) and the previous kth order statistic value \( P_{X_k} \), lower values \( L_{X_k} \) than \( P_{X_k} \) and greater or equal values of \( G_{X_k} \) than \( P_{X_k} \). Essentially, the flowchart of Fig. 4 consists of six steps for each pixel value \( x_{ij} \).

3.3.1 Updating the column histogram \( h \): The rightmost column histogram \( h^{r+1} \) (where \( r \) is the radius size of the current sliding window) when the sliding window shifts to the right. A top pixel value and a bottom pixel value while old \( x_{i-1,r+j} \), \( x_{i,r+j} \) pixel value are, respectively, removed and added to the rightmost column histogram \( h^{r+1} \).

3.3.2 Updating the kernel histogram \( H \): For all the image bit-depth (since \( l = 0 \) to \( b \) ), those values from the sliding window leftmost column histogram \( h^{l-1} \) are removed while those from the new rightmost column histogram \( h^{l+1} \) are added to the kernel histogram \( H \).

3.3.3 Updating the variables \( L_{X_k} \) and \( G_{X_k} \): For all the image bit-depth (since \( l = 0 \) to \( b \) ) is compared with the kth order statistic value \( P_{X_k} \); if the new value is greater or equal than \( P_{X_k} \), \( G_{X_k} \) is updated removing the values of the leftmost column histogram \( h^{l-1} \) and adding the values of rightmost column histogram \( h^{l+1} \) and otherwise \( L_{X_k} \) it is updated in a similar way.

3.3.4 Computing the kth order statistic value \( x_k \): This step computes the new kth order statistic value by using the kernel histogram \( H \) and the variables \( L_{X_k} \), \( G_{X_k} \) and \( P_{X_k} \). First, the previous kth order statistic value \( P_{X_k} \) is stored in variable \( x_k \); next, the threshold \( T_k \) is computed \( T_k = k(2r + 1)^2 + 1/2 \) and compared with \( L_{X_k} \); if \( T_k \leq L_{X_k} \), the new kth order statistic value will be lower than the previous one and it will be found moving downwards from the current \( P_{X_k} \) bin in the kernel histogram \( H \); otherwise, the new kth order statistic value will be greater than the previous one and will be found moving upwards from the current \( P_{X_k} \) bin in \( H \). The auxiliary variables \( L_{X_k} \) and \( G_{X_k} \) are accordingly updated in the process.

3.3.5 Target detection: The new kth order statistic value \( x_k \) is stored in the variable \( P_{X_k} \) for the next calculation of the kth order statistic value and threshold \( T \) is estimated applying a scale factor
\( \alpha \text{OS}. \) Then, the value of the test cell \( x_{i,j} \) is compared with estimated threshold \( T \) to determine if the target is present.

### 4 Theoretical efficiency analysis

Table 1 exhibits computational metrics for the efficiency evaluation of the present OS-CFAR 2D approach. For this, a set of metrics based on computational time and space has been estimated: algorithmic complexity, dynamic memory, static memory, dynamic memory accesses, arithmetic operation, and logic comparisons [23]. Metrics algorithmic complexity, dynamic memory, and static memory are globally quantified while dynamic memory accesses, arithmetic operation and logic comparisons are quantified per pixel. Each operation has update column histogram, update kernel histogram, update \( L_x \) and \( G_x \), computing the \( k \)th order statistic value and target detection are evaluated in the worst, average and best case (see Fig. 4).

As shown in Table 1, the algorithmic complexity of the OS-CFAR 2D approach is \( O(b) \) per pixel where \( b = 2^{\text{image bit depth}} \). To compute the dynamic memory, the amount of memory required to allocate the input image of size \( A \times B \), the kernel histogram \( H \) of size \( b \times 1 \) and the \( B \) column histograms \( h^j \) \((1 \leq j \leq B)\) of size \( b \times 1 \) is determined. Besides, it is needed to allocate static memory for integer \((i,j,r,l,b,x_i,P_x,L_x\) and \(G_x)\) and real \((k,T,T=\alpha \text{OS})\) auxiliary variables. Focusing on the five operations (see Fig. 4):

i. Updating the column histogram \( h^j \) requires eight dynamic memory accesses (4 accesses for the input image \( x \) and column histogram \( h^j \), respectively) and 16 arithmetic operations between addition and subtraction (worst, average, and best case).

ii. Updating the kernel histogram \( H \) requires \( 4b \) dynamic memory accesses (2\( b \) accesses for the kernel histogram \( h^j \) and column histogram \( H \), respectively), \( 6b \) arithmetic operations for addition and subtraction and \( b \) logic comparisons for a loop since 0 to \( b \) (worst, average, and best case).

iii. Updating the variables \( L_x \) and \( G_x \) needs \( 2b \) dynamic memory accesses to column histogram \( h^j \), \( 5b \) arithmetic operations between addition and subtraction and \( b \) logic comparisons for the conditional \( l \geq P_x \) (worst, average, and best case).
iv. Computing the $k$th order statistic value $x_k$. First, the initialisation of the threshold $T_k$ requires six arithmetic operations and one logic comparison for the conditional $T_k \leq L_x$. Second, $3b$ dynamic memory accesses for histogram $H$, $5b$ arithmetic operations between addition and subtraction and $2b$ logic comparisons for the loop are required. Therefore, the worst case needs $3b$ dynamic memory accesses, $5b+5$ arithmetic operations and $2b+1$ logic comparisons. The average and best case require $(3b/2)$ and one dynamic memory access, $(5b+5/2)$ and 6 arithmetic operations, $b+1$ and three logic comparisons, respectively. Take into account that in the experimental results we will show that using the breakdown point optimal concept leads to the likelihood that the best case will be produced for the calculation of the $k$th order statistic value.

v. Target detection needs one arithmetic operation to multiply $x_k$ by $a_{OS}$ and one logic comparison for the conditional $T \geq x_{ij}$ (worst, average, and best case).

5 Experimental results

The proposed approach for OS-CFAR 2D technique using the distributive histograms and breakdown point optimal concept was developed in C++ code taking advantage of the data structure within OpenCV 2.3 [33]. The programming environment (IDE) was Nokia QtCreator for GNU/Linux implementation C++ code. The framework was executed on a PC with a 2.4 GHz Intel Core i7-3630 CPU and 8-GB RAM memory with Ubuntu 14.04 LTS (32 bits) operating system.

The upper part of Fig. 5 shows an acoustic image under test. This image was obtained with an Edgetech® company side scan sonar device (note the shadow in the right part of the underwater pipeline). The size of test images is $2000 \times 900$.

On the other hand, when the statistical order $k$ increases ($k = 0.5, 0.6, 0.75$ and 0.8) the resulting images become brighter due to the selected statistical order value $x_k$. In addition, greater separation between shadow and highlight zones of the acoustic image is produced.

Fig. 6, a comparison of the standard and the breakdown point approach to compute, for each order statistic $k(0.5, 0.6, 0.75$ and 0.8), the order statistic value $x_k$ using the column-average bin-displacements measurement (vertical axis) versus the column number (horizontal axis). For each order statistic $k$, the blue, green and red curves describe the effort when radius sizes $r = 1, 3$ and 5 are used, respectively.

In addition, Table 2 shows the minimum and maximum column-average bin-displacements from Fig. 6. Note that when the radius increases the curves corresponding to the breakdown point version stabilise in comparison with the standard version. This is caused by the greater number of processed values that allows calculating the order statistic value $x_k$ and therefore the effort to achieve it tends to be zero.

Table 1: Computational metrics of the proposed OS-CFAR 2D approach

<table>
<thead>
<tr>
<th>Algorithmic complexity</th>
<th>Dynamic memory</th>
<th>Static memory</th>
<th>Operation</th>
<th>Type case</th>
<th>Dynamic memory accesses</th>
<th>Arithmetic operations</th>
<th>Logic comparisons</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(b)$</td>
<td>$X$ of size $A \times H$ of size $b \times 1$, $h^j$ ($1 \leq j \leq B$) of size $b \times 1$</td>
<td>$i, j, r, l, b, k, x_k, T_k, \alpha_{OS}$</td>
<td>update column histogram</td>
<td>worst case, average case, best case</td>
<td>8</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>update kernel histogram</td>
<td>worst case, average case, best case</td>
<td>$4b$</td>
<td>$6b$</td>
<td>$b$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>update $L_{x_k}$ and $G_{x_k}$</td>
<td>worst case, average case, best case</td>
<td>$2b$</td>
<td>$5b$</td>
<td>$b$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>computing the $k$th order statistic value</td>
<td>worst case, average case</td>
<td>$3b$</td>
<td>$5b+5$</td>
<td>$2b+1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>best case, average case</td>
<td>$2b$</td>
<td>$5b+5$</td>
<td>$b+1$</td>
</tr>
</tbody>
</table>

Besides, the results represent a segmentation of three classes: highlight (white), shadow (black), and seafloor reverberation areas (grey) employing the detection of the multiclass strategy proposed in [16]. Note that the intermediate step of the selection of the $k$th
Fig. 5 Visualisation of the $k$th order statistic value $x_k$ for different radius size $r$ (1, 3 and 5) and order statistics $k$ (0.5, 0.6, 0.75 and 0.8) for sonar image with a pipeline on seafloor.
order statistic value $x_k$ is shown in Fig. 5. It also can be seen in Fig. 7, when the radius $r$ of the sliding window increases, greater number of pixel values is used to calculate the $k$th order statistic value $x_k$, and therefore the number of false detections decreases.

On the other hand, when the $k$th statistical order is increased, the amount of false detection is also decreased. The combination of the setting parameters ($r$, $k$ and $P_{fa}$) depends on the image resolution. Note that the selection of the parameters: radius size $r = 5$ (121 pixel values), statistical order $k = 0.8$ and $P_{fa} = 0.013$ allow to demonstrate a good trade-off with the detection results obtained. In addition, these parameters configuration allow differentiating clearly the inspection features: free span, rock dump, and reflective objects on the seafloor.

In Fig. 8, three different regions of interest (ROI) extracted from the test image (Fig. 5) are shown; their sizes are $200 \times 200$ ($A \times B$) pixel values in every case, and the Cartesian coordinates of the corresponding upper-left vertex used as reference are (50, 50), (460, 460), and (500, 1000), respectively. These three sampled regions were deliberately chosen to capture different inspection features described: (ROI-1) seabed reverberation; (ROI-2) underwater pipeline with rock dumping and (ROI-1) underwater pipeline deployed on the seafloor with free span.

Regarding the ROI-1, the seabed reverberation is presented as a noisy image (speckle noise) with variations of the pixel values in any range of the greyscale. In the ROI-2, the underwater pipeline with rock dumping is presented with acoustic highlight close to the maximum value of the greyscale range. Also, the presence of rock dumping shows a pronounced acoustic shadow greater than that of Free span...
the pipeline close to the minimum value of the greyscale range. In the ROI-3, the underwater pipeline deployed on the seafloor with free span is presented with acoustic reverberation between highlight and shadow. Note that the pipeline is not completely supported on the seafloor due to the presence of seabed reverberation between the pipeline and shadow.

Fig. 9 exhibits for each ROI previously introduced in Fig. 8, the following further details: the resulting image of the $k$th order statistic value $x_k$; the detection of image involved the OS-CFAR 2D technique; and the comparison of standard and breakdown point versions to calculate the order statistic value $x_k$ using the column-average bin-displacements measurement (vertical axis) versus the column number (horizontal axis). The parameter settings for OS-CFAR 2D were $k = 0.8$, $P_{fa} = 0.013$ and different radius sizes $r = 1, 3$ and $5$. As can be seen, when the radius size increases with...
a constant order statistic and false alarm probability the number of false detections decreases.

Considering the curves of ROI-1, it is shown for both versions (standard and breakdown point) that they are stable in the presence of seabed reverberation even with a large difference in the measurement of column-average bin-displacements.

ROI-2 and ROI-3 show that the standard version has peaks close to the maximum value of the greyscale range due to the presence of the pipe and rock dumping, then it decays to the minimum value of the greyscale range because of the shadow. Finally, it grows again due to the presence of seabed reverberation. Note that the rock dumping peak of the ROI-2 is much wider than the peak of pipeline in the ROI-3. On the other hand, the curves of breakdown point version are more stable in the presence of the pipeline and rock dumping, so they require a smaller amount of bin-displacements.

Towards the application of this approach in real-time scenarios, these results show that the computational effort makes it feasible. However, note that a fixed false alarm probability \( P_{fa} \) yields a fixed detection threshold \( T \) for OS-CFAR, and consequently, false alarms may appear, as it was presented in [15]. Future work on adaptive detection threshold selection, in the sense proposed in [21], is still pending.

Fig. 10 exhibits the experimental results of processing time against radius size for the common implementation of the five OS-CFAR 2D versions utilised: (i) Perreault and Herbert method [30] (blue line), (ii) Weiss method [28] (black line) (iii) bubble sort method [25] (cyan line) (iv) bucket sort method [25] (green line) and (v) proposed method (red line). Note that, only in this context, the processing time makes sense as a measure of experimental efficiency.

These methods were tested in Fig. 5 whose size is \( 2000 \times 900 \) \((A \times B)\) pixels from an 8-bit \((b = 256)\) greyscale. In Fig. 10, the vertical axis indicates the processing time (in seconds) while the horizontal axis indicates the MF window’s radius size. As is shown, the processing time varies exponentially with the radius size for the classical bubble sort and bucket sort methods; the processing time varies linearly with the radius size for the Weiss method; conversely, the Perreault and Herbert method and the proposed method, the processing time is constant due to the dependence on the number of scale levels \( (b = 2^{\text{image bit depth}}) \) and not on the radius size. It is worth noting that the processing time of the proposed method is reduced by using the distributive histograms and breakdown point optimal concept.

6 Conclusion

The OS-CFAR 2D technique is extensively used in radar and sonar technology to cope with different applications although it requires higher computational effort. This work presents a new approach to improve the algorithmic efficiency of the OS-CFAR 2D using the distributive histograms and the optimal breakdown point concept. This approach was evaluated using the theoretical algorithm analysis as well as its experimental results on acoustic images. These studies demonstrate the great improvement obtained using the novel proposal presented for the OS-CFAR 2D technique, mainly from the computation effort standpoint.
Fig. 9 ROI of Fig. 8: the $k$th order statistic value $x_k$ image; the detection image result applying the OS-CFAR 2D technique; and the comparison of standard and breakdown point versions to calculate the order statistic value $x_k$ using the column-average bin-displacements measurement (vertical axis) versus the column number (horizontal axis)
Fig. 10 Processing time versus radius size for OS-CFAR 2D using the (i) Perreault and Herbert method [30] (blue line), (ii) Weiss method [28] (green line) (iii) bubble sort method [25] (cyan line) (iv) bucket sort method [25] (green line) and (v) proposed method (red line)

7 References