A Deterministic Approach to Detect Median Filtering in 1D Data

Cecilia Pasquini, Giulia Boato, Naif Alajlan and Francesco G.B. De Natale

***Abstract*—In this work, we propose a forensic technique able to detect the application of a median filter to 1D data. The method relies on deterministic mathematical properties of the median filter, which lead to the identification of specific relationships among the sample values that cannot be found in filtered sequences. Hence, their presence in the analyzed 1D sequence allows excluding the application of the median filter. Owing to its deterministic nature, the method ensures 0% false negatives and, although false positives (not filtered sequences classified as filtered) are theoretically possible, experimental results show that the false alarm rate is null for sufficiently long sequences. Furthermore, the proposed technique has the capability to locate with good precision a median filtered part of 1D data and provides a good estimate of the window size used.**

***Index Terms*—Forensics, Detection, Median filter**

I. Introduction

As a result of globalization and worldwide connectivity, people from all over the planet are exchanging ever increasing amounts of information of whatsoever type and form, in the most diverse fields of human activity including science, economy, social relationships, news, entertainment. This poses a number of problems related to the reliability and trustworthiness of information, as well as its potential malevolent use. Forensics technologies provide powerful tools to verify the authenticity of data and their possible manipulation, either they refer to multimodal sensed signals, medical records, geophysical observations, marketing statistics or financial reports. In this framework, the detection of any operation that could have been employed to post-process a set of data, either for malicious purposes or simply to improve their content or presentation, turns out to be of interest for a comprehensive forensic data analysis.

In this work, we consider a widely known technique commonly used for data smoothing, namely, the median filter [1]. Thanks to its ability to effectively discard outliers while preserving relevant information, median filtering has been extensively adopted as a post-processing operator in different fields, including audio processing [2], [3], image processing [4], [5], geophysics [6], economics [7], biomedical signal processing [8], both for 1D and 2D signals. Several methods have been proposed for the forensic detection of median filtering, with particular attention to images. Most of them are based on a statistical characterization of the filtered signal in different domains, often relying on machine learning tools for the detection.

To the best of our knowledge, no specific methods for the forensic analysis on 1D data are available, as they usually focus on the two-dimensional case, although part of them can be easily conceived and adapted to the 1D domain and will represent a benchmark comparison in our experimental validation phase. In [9] Kirchner and Fridrich proposed a simple yet effective median detector that exploits the artifacts introduced by the filter. The ratio of histogram bins and the subtractive pixel adjacency matrix features in the first-order difference domain are used as traces to detect median filtering in bitmap images. The first-order difference map is employed in [10] by Cao et al. to compute the probability of zero-values in texture areas of the image. A more complex median detector was proposed by Yuan in [11], based on the idea that median filtering, applied to overlapping blocks, affects the pixels ordering in each block, thus introducing a strong dependence between median values originating from overlapping filter window. Kang et al. in [12] analyzed the statistical properties of the median filter residual by using an autoregressive model. In [13] Chen et al. proposed an effective median detector based on two sets of features, the cumulative distribution function of k-th order image difference (global probability) and the local correlations between different adjacent image difference pairs (local correlation). Recently, an effective median forensic algorithm was proposed in [14], where the second-order local ternary patterns are used to capture the changes of local textures due to median filtering. All of the above techniques rely on statistical classification as a final step of the detector, thus producing both false alarms and missed detections.

In this paper we propose a deterministic approach to detect the application of a median filter on 1D signals or data sequences. The method exploits some basic mathematical properties of the median filter, which are a consequence of its very definition and enforce specific relationships among the samples of the original and filtered sequences. Such properties lead to the identification of sets of 1D patterns (called in the following "unfeasible classes") that cannot be output by a median filter. While the problem of identifying roots of median filters (i.e., patterns that are certainly preserved after the filtering) has been widely addressed in the past [15], to the best of our knowledge this is the first work focusing on the study of patterns that are certainly not introduced by a median filter.

C. Pasquini, G. Boato and F.G.B. De Natale are with the Department of Information Engineering and Computer Science, University of Trento, Trento 38123, Italy (e-mail: cecilia.pasquini@unitn.it; boato@disi.unitn.it; denatale@disi.unitn.it).

N. Alajlan is with the College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia (najlan@ksu.edu.sa).

This work was supported by the Deanship of Scientific Research of the King Saud University through the International Research Group under Project IRG14-20.

This turns out to be particularly important for forensic purposes, as the subject signal can be scanned sequentially and the presence of such patterns can be checked by means of a simple algorithm: if they are detected, then the signal is classified as negative to median filtering; otherwise, it is classified as positive to median filtering. It is worth pointing out that the detection does not require any thresholding or training operation, as it is a direct result of the scanning procedure. Moreover, given the nature of the algorithm, the rate of false negatives is guaranteed to be null, as no unfeasible classes can be present in filtered sequences.

The effectiveness of the technique is proved by extensive experiments on different kinds of 1D data, including audio tracks, economical series, physiological signals. Besides confirming the absence of false negatives, experimental results demonstrate that in practical cases the method easily achieves 0% false positives, which would be possible in principle. Indeed, the occurrence of the unfeasible classes in common data originated from different sources is extremely frequent. Moreover, the detector is able to provide as a side-information the size of the applied median filter and, thanks to the capability of detecting the unfeasible classes throughout the entire sequence, the technique can be used to segment with a high precision the filtered subsequence in the case of local filtering. Although the proposed scheme is deterministic when the median filter is the very last process applied, we also explored the possibility to exploit the distribution of the unfeasible classes in the signal to detect median filtering even when a post-processing operation is applied, thus addressing robustness issues of the deterministic detector.

The rest of the paper is organized as follows: in Section II the rationale behind the proposed approach is explained in detail and the design of the final detectors is presented in Section III, while the results of the experimental tests are reported in Section IV. Finally, we draw the conclusions of our work in Section V.

II. Median filter detection and unfeasible sequences

In this work, we design a forensic detector of median filtering for 1D data based on deterministic properties of such processing, which can be applied to 1D signals or, in general, to any set of ordered one-dimensional data samples.

First, we introduce the theoretical background and the main rationale behind our method in Section II-A. Then, in Section II-B we propose an algorithmic procedure for the analysis of 1D data, that will be exploited in the following sections.

A. Theoretical background

We will represent the one-dimensional objects analyzed as numerical sequences $\{y_i\}_{i=1}^{\infty} \subset \mathbb{R}$, that we will simply denote as $\{y_i\}$ for the sake of brevity[1] . Then, the action of median filtering can be defined as follows:

[1]In practice the sequences to be analyzed will be finite, thus we will have that $y_i = 0$ when i is higher than a certain value.

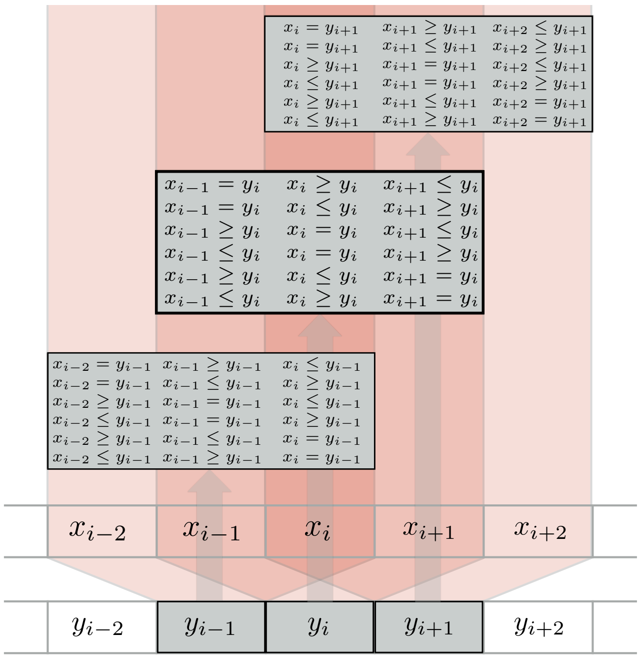

$x_i = y_{i+1}$ $x_{i+1} \geq y_{i+1}$ $x_{i+2} \leq y_{i+1}$
$x_i = y_{i+1}$ $x_{i+1} \leq y_{i+1}$ $x_{i+2} \geq y_{i+1}$
$x_i \geq y_{i+1}$ $x_{i+1} = y_{i+1}$ $x_{i+2} \leq y_{i+1}$
$x_i \leq y_{i+1}$ $x_{i+1} = y_{i+1}$ $x_{i+2} \geq y_{i+1}$
$x_i \geq y_{i+1}$ $x_{i+1} \leq y_{i+1}$ $x_{i+2} = y_{i+1}$
$x_i \leq y_{i+1}$ $x_{i+1} \geq y_{i+1}$ $x_{i+2} = y_{i+1}$
$x_{i-1} = y_i$ $x_i \geq y_i$ $x_{i+1} \leq y_i$
$x_{i-1} = y_i$ $x_i \leq y_i$ $x_{i+1} \geq y_i$
$x_{i-1} \geq y_i$ $x_i = y_i$ $x_{i+1} \leq y_i$
$x_{i-1} \leq y_i$ $x_i = y_i$ $x_{i+1} \geq y_i$
$x_{i-1} \geq y_i$ $x_i \leq y_i$ $x_{i+1} = y_i$
$x_{i-1} \leq y_i$ $x_i \geq y_i$ $x_{i+1} = y_i$
$x_{i-2} = y_{i-1}$ $x_{i-1} \geq y_{i-1}$ $x_i \leq y_{i-1}$
$x_{i-2} = y_{i-1}$ $x_{i-1} \leq y_{i-1}$ $x_i \geq y_{i-1}$
$x_{i-2} \geq y_{i-1}$ $x_{i-1} = y_{i-1}$ $x_i \leq y_{i-1}$
$x_{i-2} \leq y_{i-1}$ $x_{i-1} = y_{i-1}$ $x_i \geq y_{i-1}$
$x_{i-2} \geq y_{i-1}$ $x_{i-1} \leq y_{i-1}$ $x_i = y_{i-1}$
$x_{i-2} \leq y_{i-1}$ $x_{i-1} \geq y_{i-1}$ $x_i = y_{i-1}$
x_{i-2} x_{i-1} x_i x_{i+1} x_{i+2}
y_{i-2} y_{i-1} y_i y_{i+1} y_{i+2}

Fig. 1: Representation of the influence of $\{y_i\}$ on $\{x_i\}$ for the case of $N = 3$.

Definition *Given a sequence $\{x_i\}$ and a natural odd number N, the output of a median filter with size N applied to $\{x_i\}$ is a sequence $\{y_i\}$ such that*

$$y_i = median(X_i),$$
$$X_i := \{x_{i-\lfloor \frac{N}{2} \rfloor}, \dots, x_i, \dots, x_{i+\lfloor \frac{N}{2} \rfloor}\},$$

when $i > \lfloor \frac{N}{2} \rfloor$ and $y_i = 0$ otherwise.

In a forensic framework, we assume to analyze a 1D data and look for traces of previous median filtering. In other words, we deal with an “inverse” problem, where we are given a sequence $\{y_i\}$ and we need to determine whether it is the output of a median filter of a certain size N applied to an original unknown sequence $\{x_i\}$ or not.

In order to provide an answer to this question, we can now exploit the following consequence of the median filtering definition:

Property *Let $\{y_i\}$ be the output of a median filter of size N applied to $\{x_i\}$. Then, $\forall i > \lfloor \frac{N}{2} \rfloor$ the following facts hold:*

- *at least one value in X_i is equal to y_i*
- *among the other values of X_i, $\lfloor \frac{N}{2} \rfloor$ of them are equal to or greater than y_i and the remaining $\lfloor \frac{N}{2} \rfloor$ are equal to or lower than y_i.*

As an example, we consider the simple case where $N = 3$. In such case, if $\{y_i\}$ is the sequence to be analyzed and we suppose it is the output of a median filter with size 3 applied to an unknown sequence $\{x_i\}$, a generic value $y_i, i > \lfloor \frac{N}{2} \rfloor$ introduces some knowledge on the 3 elements of X_i, as illustrated in Fig. 1. In particular, we can observe that x_{i-1}, x_i and x_{i+1} must satisfy at least one of the 6 sets of conditions reported in the central grey block in Fig. 1, and the same

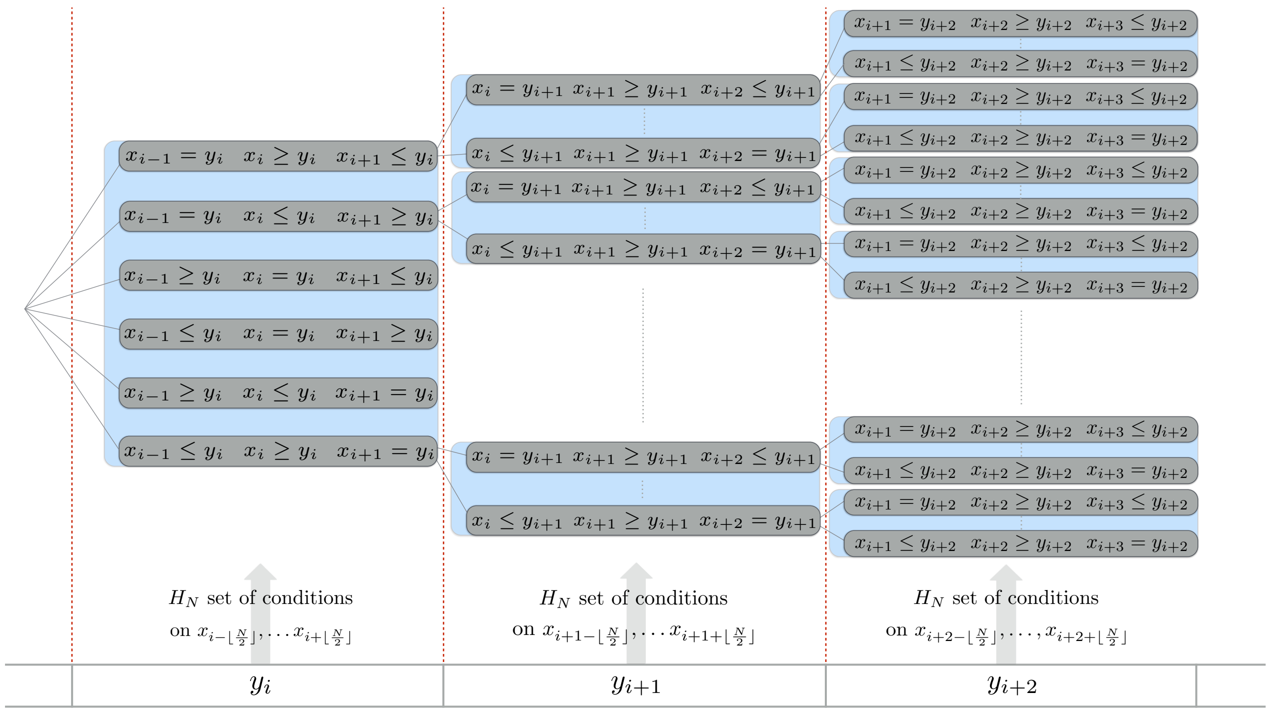

$x_{i-1} = y_i \quad x_i \geq y_i \quad x_{i+1} \leq y_i$
$x_{i-1} = y_i \quad x_i \leq y_i \quad x_{i+1} \geq y_i$
$x_{i-1} \geq y_i \quad x_i = y_i \quad x_{i+1} \leq y_i$
$x_{i-1} \leq y_i \quad x_i = y_i \quad x_{i+1} \geq y_i$
$x_{i-1} \geq y_i \quad x_i \leq y_i \quad x_{i+1} = y_i$
$x_{i-1} \leq y_i \quad x_i \geq y_i \quad x_{i+1} = y_i$
$x_i = y_{i+1} \quad x_{i+1} \geq y_{i+1} \quad x_{i+2} \leq y_{i+1}$
$x_i \leq y_{i+1} \quad x_{i+1} \geq y_{i+1} \quad x_{i+2} = y_{i+1}$
$x_{i+1} = y_{i+2} \quad x_{i+2} \geq y_{i+2} \quad x_{i+3} \leq y_{i+2}$
$x_{i+1} \leq y_{i+2} \quad x_{i+2} \geq y_{i+2} \quad x_{i+3} = y_{i+2}$
H_N set of conditions on $x_{i-\lfloor\frac{N}{2}\rfloor}, \dots x_{i+\lfloor\frac{N}{2}\rfloor}$
H_N set of conditions on $x_{i+1-\lfloor\frac{N}{2}\rfloor}, \dots x_{i+1+\lfloor\frac{N}{2}\rfloor}$
H_N set of conditions on $x_{i+2-\lfloor\frac{N}{2}\rfloor}, \dots, x_{i+2+\lfloor\frac{N}{2}\rfloor}$
y_i
y_{i+1}
y_{i+2}

Fig. 2: Hypothesis tree for a median filter $N = 3$.

holds for all the values in $\{y_i\}$: y_{i+1} enforces 6 possible sets of conditions involving the 3 elements x_i, x_{i+1}, x_{i+2}, as well as y_{i-1} for x_{i-2}, x_{i-1}, x_i, and so on. We can replicate the procedure for a generic filter size N, and we obtain that each y_i affects the N values in X_i by imposing a number of possible sets of conditions that is equal to

$$H_N = N \frac{(N-1)!}{\left(\frac{N-1}{2}\right)! \left(\frac{N-1}{2}\right)!}.$$

Clearly, H_N significantly increases with N: $H_N = 6, 30, 140, 630$ for $N = 3, 5, 7, 9$, respectively.

If we start from a value y_i and move forward in $\{y_i\}$, at each step we will have H_N possible systems introduced and the elements of $\{x_i\}$ must fulfill at least one of them. This can be represented by a tree, as in Fig. 2, where at each node one of the possible set of conditions is added to the ones cumulated in the previous steps along the corresponding path, thus obtaining an equality/inequality system at each node. Now, at each step the systems introduced and the ones of the previous step will share $N-1$ overlapping variables (the intersection of X_i and X_{i+1}). Hence, according to the values in $\{y_i\}$, at each node the system cumulated along the branch might contain conditions on the same variable that are not compatible: in such cases, the cumulated system has an empty feasibility region and we will define the branch as *unfeasible*; otherwise, we will denote it as *feasible*.

Clearly, if at a certain step j all the branches are unfeasible, it means that no sequence $\{x_i\}$ exists such that it could generate $\{y_i\}$ when median filtered with size N. In other words, we have the deterministic proof that $\{y_i\}$ *is not* the output of a median filter of size N and we obtain a response for the forensic problem we face. Although the specific framework and mathematical tools used are different, such approach can be compared to the ones proposed in [16] and [17], where a similar rationale is employed for the detection of resampling in signals and steganography in digital images, respectively.

B. Algorithmic checking procedure

In the light of the above, a possible approach for analyzing a given sequence $\{y_i\}$ could be to progressively scan it and determine at each step whether the tree generated by the samples contains at least one feasible branch, meaning that a median filtering with size N might have occurred. In this regard, we propose a recursive algorithmic procedure for the analysis of the sequence $\{y_i\}$ starting from a generic element at position i up to a certain number T_{max} of successive values. It is based on a recursive algorithm consisting of a depth-first visit of the tree and its pseudo-code is reported in Algorithm 1. In particular, it takes as input arguments the size N of the median filter, the sequence $\{y_i\}$, the index i of the first element of $\{y_i\}$ to be considered, the maximum number T_{max} of successive values to be scanned and the current level T of the tree (which is initially set to 1 and increases at each iteration up to T_{max}).

In other words, at each call of the CHECK function the algorithm creates all the H_N possible sets of conditions generated by the sample at the current location and check their consistency with the existing branches (created at the previous call) in a sequential order. When the first feasible condition is found, the function recursively launches itself and moves to the successive sample. As we observed, the number H_N substantially increases with N, thus leading to a higher computational complexity of the algorithm when raising N.

Algorithm 1

```
function CHECK($N$,$\{y_i\}$,$i$,$T$,$T_{max}$)
    Get $y_i$ from $\{y_i\}$
    Create the $H_N$ sets of conditions imposed by $y_i$
    for each set of conditions do
        Check conditions overlapping for that $T$
        if conditions are compatible then
            if $T = T_{max}$ then
                return $OK$
            else
                $R =$ CHECK($N$, $\{y_i\}$,$i+1$,$T+1$,$T_{max}$)
                if $R = OK$ then
                    return $OK$
                end if
            end if
        end if
    end for
    return $\neg OK$
end function
```

Fig. 3 shows an example referred to a practical case for the filter size $N = 3$. Here, we consider a sequence $\{y_i\}$ such that

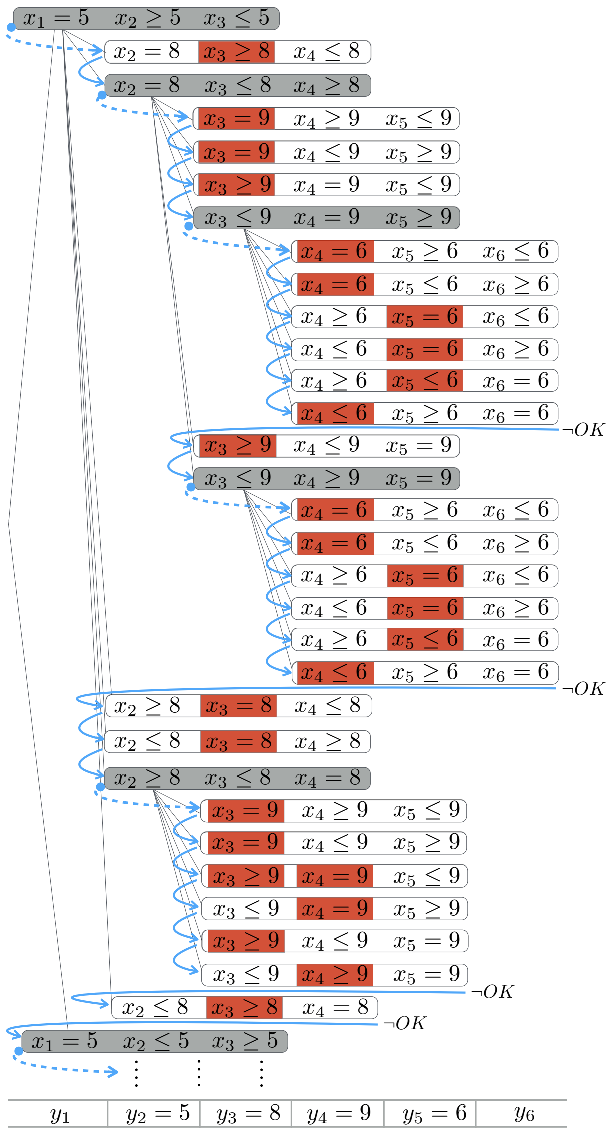
$$y_2 = 5, \quad y_3 = 8, \quad y_4 = 9, \quad y_5 = 6,$$

and illustrate the algorithmic procedure when CHECK(3,$\{y_i\}$,2,1,T_{max}) (where $T_{max} \geq 4$) is called. The analyzed sequence is reported at the bottom and the tree generated is represented above. The blue arrows show the branches sequentially analyzed by the algorithm, where the dotted ones indicate a new call of the CHECK function. Conditions on same elements of $\{x_i\}$ are aligned vertically and a red box indicates that an explicit inconsistency with respect to the previous set of conditions along the current branch is found. On the other hand, a grey box indicates that the current set of conditions is compatible and the CHECK function is recursively called with the third and fourth input arguments increased by a unit. For the sake of brevity, only the check on the first set of conditions generated by the first sample $y_2 = 5$ is represented (i.e., $x_1 = 5, x_2 \geq 5, x_3 \leq 5$); however, the following ones are treated in the same way and all of them turn out to be closed, thus showing that the analyzed sequence cannot be generated by a median filter of size 3.

Although a progressive scanning by means of such algorithmic procedure would represent a valid solution from a theoretical point of view, the actual application to data sequences of considerable length is computationally demanding, especially when N increases. However, we will see in the next section that we can exploit additional properties of the median filter and adopt smarter strategies in order to simplify the analysis from both a theoretical and a computational perspective.

III. UNFEASIBLE CLASSES AND $\mathcal{N}$-DETECTORS

A peculiar property of the median filter is the fact that it is based on a sorting operation and, because of that, the set of conditions on $\{x_i\}$ that we defined in the previous section are



$x_1 = 5$ $x_2 \geq 5$ $x_3 \leq 5$
$x_2 = 8$ $x_3 \geq 8$ $x_4 \leq 8$
$x_2 = 8$ $x_3 \leq 8$ $x_4 \geq 8$
$x_3 = 9$ $x_4 \geq 9$ $x_5 \leq 9$
$x_3 = 9$ $x_4 \leq 9$ $x_5 \geq 9$
$x_3 \geq 9$ $x_4 = 9$ $x_5 \leq 9$
$x_3 \leq 9$ $x_4 = 9$ $x_5 \geq 9$
$x_4 = 6$ $x_5 \geq 6$ $x_6 \leq 6$
$x_4 = 6$ $x_5 \leq 6$ $x_6 \geq 6$
$x_4 \geq 6$ $x_5 = 6$ $x_6 \leq 6$
$x_4 \leq 6$ $x_5 = 6$ $x_6 \geq 6$
$x_4 \geq 6$ $x_5 \leq 6$ $x_6 = 6$
$x_4 \leq 6$ $x_5 \geq 6$ $x_6 = 6$
$\neg OK$
$x_3 \geq 9$ $x_4 \leq 9$ $x_5 = 9$
$x_3 \leq 9$ $x_4 \geq 9$ $x_5 = 9$
$x_4 = 6$ $x_5 \geq 6$ $x_6 \leq 6$
$x_4 = 6$ $x_5 \leq 6$ $x_6 \geq 6$
$x_4 \geq 6$ $x_5 = 6$ $x_6 \leq 6$
$x_4 \leq 6$ $x_5 = 6$ $x_6 \geq 6$
$x_4 \geq 6$ $x_5 \leq 6$ $x_6 = 6$
$x_4 \leq 6$ $x_5 \geq 6$ $x_6 = 6$
$\neg OK$
$x_2 \geq 8$ $x_3 = 8$ $x_4 \leq 8$
$x_2 \leq 8$ $x_3 = 8$ $x_4 \geq 8$
$x_2 \geq 8$ $x_3 \leq 8$ $x_4 = 8$
$x_3 = 9$ $x_4 \geq 9$ $x_5 \leq 9$
$x_3 = 9$ $x_4 \leq 9$ $x_5 \geq 9$
$x_3 \geq 9$ $x_4 = 9$ $x_5 \leq 9$
$x_3 \leq 9$ $x_4 = 9$ $x_5 \geq 9$
$x_3 \geq 9$ $x_4 \leq 9$ $x_5 = 9$
$x_3 \leq 9$ $x_4 \geq 9$ $x_5 = 9$
$\neg OK$
$x_2 \leq 8$ $x_3 \geq 8$ $x_4 = 8$
$\neg OK$
$x_1 = 5$ $x_2 \leq 5$ $x_3 \geq 5$
$\vdots$ $\vdots$ $\vdots$
y_1 $y_2 = 5$ $y_3 = 8$ $y_4 = 9$ $y_5 = 6$ y_6

Fig. 3: Representation of the CHECK function with $N = 3$ for a specific sequence $\{y_i\}$ starting from the position 2.

composed of equalities and non-strict inequalities. By looking at Fig. 2, we have that the structure of the tree (the number and kind of conditions added at each step) is determined uniquely by the specific filter size N we are considering. On the other hand, the feasibility of each branch only depends on the values of the y_i's, as it is easy to observe that in an equality/inequality system the existence of a solution only depends on the order relations between all the constant terms involved, that in our case are represented by the elements of $\{y_i\}$. Indeed, in Section II-B we have shown how to verify that for $N = 3$ a sequence such that

$$y_2 = 5, \quad y_3 = 8, \quad y_4 = 9, \quad y_5 = 6,$$

generates a tree with only unfeasible branches, but we would have obtained the same result by applying the checking procedure to any sequence with the same 6 order relations between y_2, y_3, y_4 and y_5, i.e., such that

$$y_2 < y_3, \quad y_2 < y_4, \quad y_2 < y_5,$$
$$y_3 < y_4, \quad y_3 > y_5,$$
$$y_4 > y_5.$$

Let us now suppose to analyze a generic subsequence of $\{y_i\}$ composed of L elements in consecutive positions, regardless of its starting point in $\{y_i\}$, which can be seen as a vector of length L with values in $\mathbb{R}$. Thus, we can identify $\mathbb{R}^L$ as the set of all such possible objects and, consistently with the notation on trees, we can denote each vector $\mathbf{a}_L \in \mathbb{R}^L$ as *feasible* (*unfeasible*) for a filter size N if the corresponding tree, generated as described in Section II, contains at least one feasible branch (contains only unfeasible branches).

According to the previous observations, we can give the following definition and lemma:

Definition *Let $\sim$ be a binary relation over $\mathbb{R}^L$ such that for $\mathbf{a}_L, \mathbf{b}_L \in \mathbb{R}^L$*

$$\mathbf{a}_L \sim \mathbf{b}_L \quad \iff \quad \begin{array}{l}\textit{all the } L(L-1)/2 \textit{ order relations}\\ \textit{between the elements of } \mathbf{a}_L\\ \textit{hold also for the corresponding}\\ \textit{elements of } \mathbf{b}_L\end{array}$$

Example

$$(1, 3, 2, 4) \sim (100, 300, 200, 400),$$
$$(1, 3, 2, 4) \sim (-20, -1, -7, 50),$$
$$(1, 3, 2, 4) \nsim (-50, 0, 50, 100).$$

Lemma *If $\mathbf{a}_L \in \mathbb{R}^L$ is feasible (unfeasible) for a certain filter size N, then any $\mathbf{b}_L \in \mathbb{R}^L$ such that $\mathbf{b}_L \sim \mathbf{a}_L$ is feasible (unfeasible) for N.*

Proof: Let $\mathcal{A}$ and $\mathcal{B}$ be the trees generated for the fixed value of N from $\mathbf{a}_L$ and $\mathbf{b}_L$, respectively, as described in Section II-A (see Fig. 2). Let then $F_{\mathcal{A}}(i, l)$ be the cumulated system at the i-th node of the l-th level of $\mathcal{A}$, where $1 \leq i \leq (H_N)^l$ and $1 \leq l \leq L$; the same holds for $F_{\mathcal{B}}(i, l)$. Being fixed the value of N, $F_{\mathcal{A}}(i, l)$ and $F_{\mathcal{B}}(i, l)$ share the same variables and they are subject to the very same equality or inequality conditions with the exception of the constant terms, which are given by the elements of $\mathbf{a}_L$ and $\mathbf{b}_L$. By hypothesis, the corresponding elements of $\mathbf{a}_L$ and $\mathbf{b}_L$ have the same order relations and the systems $F_{\mathcal{A}}(i, l)$ and $F_{\mathcal{B}}(i, l)$ have either empty or not empty feasibility regions. □

In other words, we can limit the analysis to the classes on $\mathbb{R}^L$ (that we will denote as *L-classes*) defined by the relation $\sim$, since all the vectors of $\mathbb{R}^L$ with the same order relations between their components will be either feasible or unfeasible. This represents a crucial result, as such L-classes are in a finite number (for a given L they can be explicitly determined by means of combinatorics rules) and we can perform our analysis *a priori*, thus avoiding the application of the algorithmic procedure in II-B to the given sequence. As an example, we proved in Section II-B that $(5, 8, 9, 6)$ is unfeasible in $\mathbb{R}^4$ for $N = 3$. Given a generic sequence $\{y_i\}$, if we find along the sequence 4 values in a consecutive position which have the same order relations as $(5, 8, 9, 6)$, we can deterministically assert that such sequence (or such part of the sequence) has not been median filtered with $N = 3$.

To this extent, we can generate all the L-classes for any value of L and determine which of them are feasible or unfeasible for a given filter size N. We will then denote as $\mathcal{U}_L^N$ the set of L-classes that are unfeasible for a certain N and do not contain any L'-class, $L' < L$, that is itself unfeasible for N. By building such sets, we can obtain a set of patterns (meant as order relations between consecutive elements) to be sought in the given sequence $\{y_i\}$ in order to determine whether it has been filtered or not, as it is explored in the next section.

A. Identification of feasible and unfeasible classes

In order to identify *a priori* the feasible and unfeasible patterns for a certain filter size N, we exploited the algorithm proposed in II-B by analyzing the L-classes up to a certain length.

First, we observed that the filter size N determines a maximum length for its unfeasible classes, which is equal to $2N - 1$. Indeed, the following lemma holds:

Lemma *Let $\{x_i\}$ be a numerical sequence in $\mathbb{R}$ and $\{y_i\}$ the output of a median filter with size N applied to $\{x_i\}$. Then each y_i will be related to its adjacent values within a maximum window of size $2N - 1$.*

Proof: By definition of the median filter, each y_i depends uniquely on X_i. At the same time, X_i affects the values in the set $Y_i = \{y_k,\ k = i - \lfloor \frac{N}{2} \rfloor, \ldots, i + \lfloor \frac{N}{2} \rfloor\}$, thus establishing a mutual relationship between y_i and the elements of Y_i, whose cardinality is $2N - 1$. □

Hence, we have that the maximum length is given by 5, 9, 13 and 17 for $N = 3, 5, 7, 9$, respectively.

Then, we employed the function UNFEASIBLECLASSES, whose pseudocode is reported in Algorithm 2, to identify the unfeasible L-classes for a specific filter size N and for L starting from 2 up to $2N - 1$. In particular, at each step all the possible L-classes for the current L are created, the ones which contain a $(L-1)$-class identified as unfeasible at the previous iteration are discarded and, among the remaining ones, the CHECK function determines which ones are unfeasible, thus obtaining the set $\mathcal{U}_L^N$ for each length L.

In Table I, we report the cardinality of the $\mathcal{U}_L^N$ obtained, indicated as $|\mathcal{U}_L^N|$. The computational complexity of the procedure increases both with L (because of the total number of L-classes, which is reported in the second column) and with N (because of the number of possible sets of conditions). Thus, we limited the analysis to the classes up to $L = 11$ and in the hardest cases (marked with the symbol $*$), instead of all the possible L-classes, we considered only the ones where the relations between the L elements are strict inequalities, which can be seen as the $L!$ possible permutations of L elements (for $L = 11$ we have almost 40 million permutations). Although this does not provide an exhaustive analysis ($2N - 1$ would

Algorithm 2 Identification of unfeasible classes

```
function UnfeasibleClasses($N$)
    for $L = 2, \dots, 2N-1$ do
        Create all the possible $L$-classes
        for each $L$-class do
            if it does not contain an element of $\mathcal{U}_{L-1}^N$ then
                Choose an $\mathbf{a}_L \in \mathbb{R}^L$ in the current class
                Create the sequence $\{a_i\}$,
                whose first $L$ values are the elements of $\mathbf{a}_L$
                if Check($N$,$\{a_i\}$,1,1,$L$)= $\neg OK$ then
                    Save current class in $\mathcal{U}_L^N$
                end if
            end if
        end for
    end for
end function
```

TABLE I: Number of unfeasible L-classes derived for different values of N. The symbol $*$ means that only classes with strict inequality relations between the L elements have been considered. The last row reports the total time (in seconds) necessary to derive all the unfeasible classes of the corresponding value of N.

L	N. of L-classes	$\lvert\mathcal{U}_L^3\rvert$	$\lvert\mathcal{U}_L^5\rvert$	$\lvert\mathcal{U}_L^7\rvert$	$\lvert\mathcal{U}_L^9\rvert$
2	2	0	0	0	0
3	13	0	0	0	0
4	75	12	0	0	0
5	541	20	60	36	36
6	4683	0	468	270	222
7	47293	0	74	1712	980
8	545835	0	34	7666	5578
9	7087261	0	2	1802	31496
10	102247563	0	0	838	29776*
11	$\sim$ 1 billion	0	0	478	6510*
Total computation time		0.15s	580.3s	27320s	51960s

be higher for $N = 7, 9$), the number of unfeasible classes detected among the $L!$ considered is significant and sufficient to correctly detect the filter, as we will see in the next sections. It is to be pointed out that, although it is time consuming for higher values of N, the above operation can be performed off-line and once-for-all, and can be easily parallelized; moreover, aggregating all unfeasible classes detected among different values of L and N requires very limited memory requirements.

As an example, in Fig. 4 the 12 unfeasible 4-classes for $N = 3$ are graphically represented, where the distance between each value has been normalized to a common unit. We can notice that in most of the classes the values satisfy strict inequalities, while in 4 classes the first and the last elements are equal. Moreover, for each class its symmetric counterpart in the vertical and horizontal direction are included in $\mathcal{U}_4^3$. Indeed, it is easy to observe that in case of vertical or horizontal symmetry the building of the tree leads to the same results, thus suggesting that a further equivalence could be introduced in order to represent the unfeasible patterns in a more compact way.

Furthermore, it is interesting to visualize how much the unfeasible classes are similar for different values of N. To this end, we identified the unfeasible classes at a certain L that are shared by more than one value of N and the ones that are unfeasible for only one value of N. In Fig. 5, such analysis is graphically represented as Euler diagram, highlighting the different behaviour of the unfeasible classes when L varies. For instance, we can notice that the sets $\mathcal{U}_5^7$ and $\mathcal{U}_5^9$ coincide and, in general, the unfeasible classes for $N = 7$ and $N = 9$ have a significant overlap (which decreases by increasing L). Clearly, this intrinsically affects the detection performance of the method when a different window size is used in the filtering. Indeed, it is easy to predict that if a sequence has been median filtered with $N = 7$, the probability that it will contain classes that are unfeasible for $N = 9$ will strongly decrease, as most of them are unfeasible also for $N = 7$ and have been certainly removed.

B. $\mathcal{N}$-detectors

Once the sets $\mathcal{U}_L^N$ are identified, the definition of simple and fast detectors is straightforward. Indeed, differently as the solution proposed in Section II-B, it is now possible to simply scan the sequence progressively and check whether the elements of $\{y_i\}$ fulfill or not the order relations corresponding to the classes in the sets $\mathcal{U}_L^N$: if such classes are present in the sequence, we can deterministically classify $\{y_i\}$ as not filtered; if none of the unfeasible classes is found, we classify

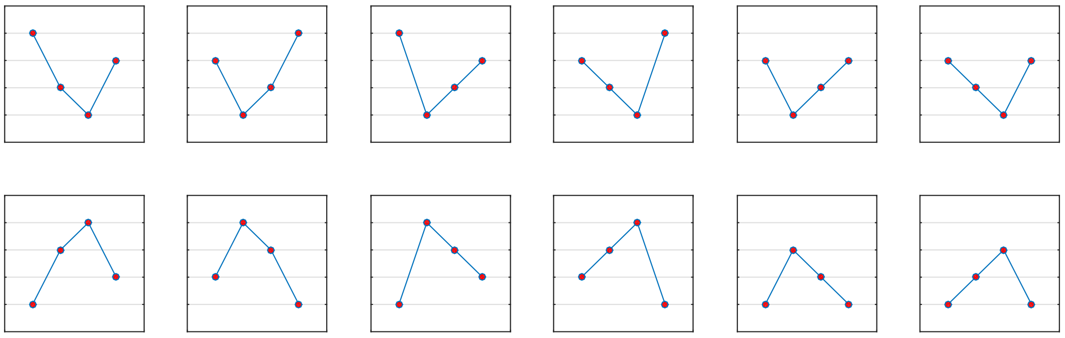


Fig. 4: Graphical visualization of the elements in $\mathcal{U}_4^3$.

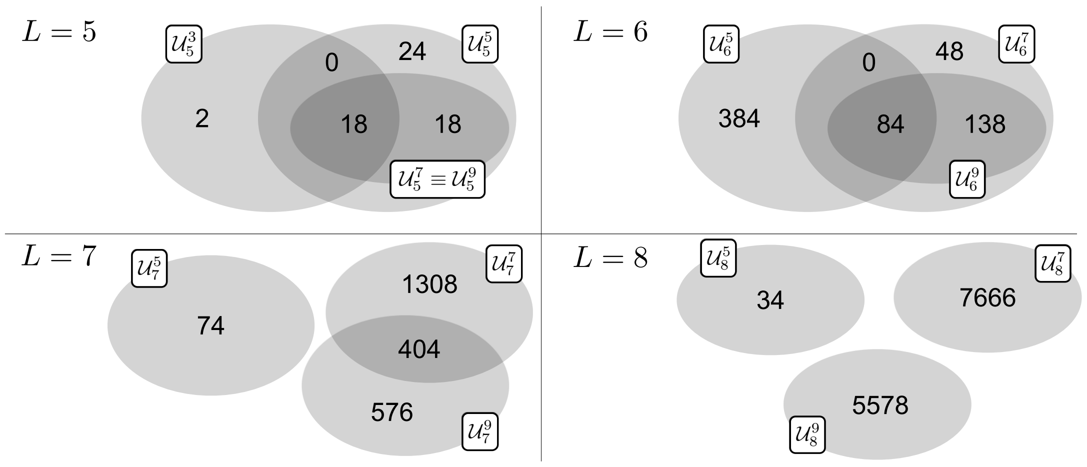


Fig. 5: Relative sharing of the unfeasible classes among different values of N and L. A number indicating the cardinality is placed in every set. Notice that for $N = 3$ the sets $\mathcal{U}_4^N$ and $\mathcal{U}_5^N$ are empty.

the sequence as filtered. Although this second assumption is not deterministic (unfeasible patterns might be missing also in a not filtered sequence), we will see that in practice the false alarm rate is basically null for sequences with a sufficient length, since pristine signals most likely contain unfeasible classes. A deeper analysis of such aspect is provided in Section IV-A, where we establish a relationship between the length of the analyzed sequence and the false alarm probability. Moreover, the implementation of such procedure leads to algorithms with a quite low computational complexity, consisting of a simple check on order relations.

Finally, the knowledge of the $\mathcal{U}_L^N$'s for the different values of N allows for a median filter detection targeted to one or more values of the filter size N. Precisely, considering a generic set of integer odd values $\mathcal{N}$, we can define its corresponding $\mathcal{N}$-detector: in this case, the presence of all the unfeasible classes in $\bigcup_{N\in\mathcal{N}} \mathcal{U}_L^N$ will be checked and the sequence will be considered as positive to median filtering detection if only feasible classes for at least one value of $N \in \mathcal{N}$ are detected; on the other hand, we consider the sequence as negative to median filtering detection if at least one unfeasible class for each value of $N \in \mathcal{N}$ is detected.

IV. Testing

The proposed detection method has been tested on various kinds of 1D data. In particular, we considered 4 different datasets:

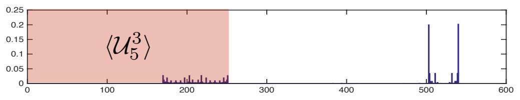
- **Music**: music audio clips of different length and sources have been taken from the entire version of the publicly available dataset used in [18], which includes 64 clips of 30 seconds and a genre collection composed by 1000 tracks of 30 seconds. They are stored in .wav and .au format and their sources vary from CD to radio and microphone recordings. The total number of clips is 1064, each one sampled at 22050Hz and consisting of 661500 samples.
- **Speech**: similarly, speech audio clips have been taken from [18] (where the technical specifications are the same as the music clips) and from the AMI Corpus [19], which includes 36 conference recordings of 20-60 minutes. From the latter, the first 20 millions samples (20 minutes) of each recording have been considered. The total duration of the dataset is 752 minutes.
- **Stock**: we downloaded historical stock data [20] of all the companies listed on NASDAQ stock exchange since at least 10 years. In particular, we considered the daily closing price of each company's security, thus obtaining a total number of 1299 sequences whose length varies from 3650 to about 9000 (depending on the date they entered the stock market).
- **Ecg**: Electrocardiogram (ECG) data (used in [8]) have been downloaded from the publicly available MIT-BIH Arrhythmia Database [21], the MIT-BIH Supraventricular Arrhythmia Database [22] and from the European ST-T Database [23]. The three datasets provide a total number of 216 ECG sequences containing on average 300000 samples.

A. *False alarm probability analysis*

As previously stressed, our detection method can guarantee a null rate of false negatives (median filtered data classified as not filtered), as filtered sequences will not present unfeasible classes by definition. On the other hand, not filtered sequences that do not contain unfeasible classes are possible in principle. In those cases, the detector would fail and classify the sequence as filtered, thus leading to false positives.

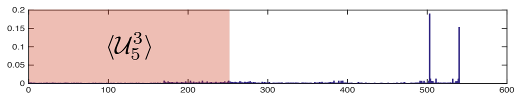
In order to quantify such false alarm probability, we performed a preliminary analysis on a subset of each dataset. Let us consider a value of N and a sequence with length equal to T. Then, we want to estimate $p_{FA}(T)$, the false alarm probability of the $\{N\}$-detector for a given dataset as a function of T, which can be seen as the probability of finding only feasible classes for N in sequences with T samples that have not been filtered. For the sake of simplicity, let L be such that $\exists L' \leq L$ with $\mathcal{U}_{L'}^N \neq \emptyset$, and let us suppose to consider only non-overlapping subsequences of length L contained in

MUSIC



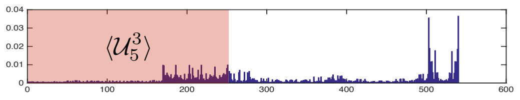
	$N=3$		$N=5$	
	p_U	T_{min}	p_U	T_{min}
ORIGINAL	0.40	50	0.32	84
MEDIAN 3	0	-	0.23	126
MEDIAN 5	$9.8 \cdot 10^{-2}$	2235	0	-
MEDIAN 7	$1.1 \cdot 10^{-2}$	2025	$5.1 \cdot 10^{-3}$	6307
MEDIAN 9	$1.1 \cdot 10^{-2}$	1180	$1.3 \cdot 10^{-2}$	2317

SPEECH



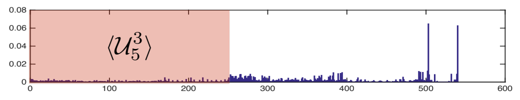
	$N=3$		$N=5$	
	p_U	T_{min}	p_U	T_{min}
ORIGINAL	0.47	40	0.39	70
MEDIAN 3	0	-	0.19	147
MEDIAN 5	$2.0 \cdot 10^{-2}$	1100	0	-
MEDIAN 7	$1.4 \cdot 10^{-2}$	1575	$1.5 \cdot 10^{-2}$	2037
MEDIAN 9	$1.2 \cdot 10^{-2}$	1825	$1.7 \cdot 10^{-2}$	1813

STOCK



	$N=3$		$N=5$	
	p_U	T_{min}	p_U	T_{min}
ORIGINAL	0.40	45	0.38	70
MEDIAN 3	0	-	0.12	252
MEDIAN 5	$1.7 \cdot 10^{-2}$	1315	0	-
MEDIAN 7	$1.0 \cdot 10^{-2}$	2110	$9.4 \cdot 10^{-3}$	3430
MEDIAN 9	$8.1 \cdot 10^{-3}$	2845	$7.8 \cdot 10^{-3}$	4144

ECG



	$N=3$		$N=5$	
	p_U	T_{min}	p_U	T_{min}
ORIGINAL	0.20	100	0.21	133
MEDIAN 3	0	-	$5.2 \cdot 10^{-2}$	602
MEDIAN 5	$5.3 \cdot 10^{-3}$	4300	0	-
MEDIAN 7	$6.0 \cdot 10^{-3}$	3810	$6.0 \cdot 10^{-3}$	5383
MEDIAN 9	$3.1 \cdot 10^{-3}$	7300	$4.0 \cdot 10^{-3}$	7959

Fig. 6: Pmf of the 5-classes and 7-classes in the different datasets. The histogram depicted refers to the ORIGINAL case for each dataset and the 252 classes belonging to $\langle \mathcal{U}_5^3 \rangle$ have been placed in the first bins. For the different processing (reported row wise), we computed the value of p_U and the value T_{min}, by imposing a false alarm probability upper bound equal to 0.01.

TABLE II: False alarm of the $\{3, 5, 7, 9\}$-detector.

	MUSIC	SPEECH	STOCK	ECG
ORIGINAL	0%	0%	0%	0%
MOVING AVERAGE	0%	0%	0%	0%
GAUSSIAN LOWPASS	0%	0%	0%	0%

TABLE III: Average computational time in seconds of the $\{3, 5, 7, 9\}$-detector for 100 samples.

	MUSIC	SPEECH	STOCK	ECG
ORIGINAL	$8.5 \cdot 10^{-6}$	$8.0 \cdot 10^{-6}$	$3.4 \cdot 10^{-4}$	$1.8 \cdot 10^{-5}$
MOVING AVERAGE	$8.3 \cdot 10^{-6}$	$7.9 \cdot 10^{-6}$	$3.5 \cdot 10^{-4}$	$1.7 \cdot 10^{-5}$
GAUSSIAN LOWPASS	$8.4 \cdot 10^{-6}$	$7.9 \cdot 10^{-6}$	$3.4 \cdot 10^{-3}$	$1.7 \cdot 10^{-5}$

the dataset, whose corresponding L-class is represented by a random variable C_L.

Then, we can assume that all the L-classes (meant as the L-classes corresponding to the subsequences of length L) contained in the dataset are independent realizations of C_L, and that we have an estimate of the probability mass function of C_L. Now, the sample space of C_L coincides with all the possible L-classes and is composed of two disjoint parts: the L-classes that do not contain nor are themselves unfeasible classes and the ones that either belong to $\mathcal{U}_L^N$ or contain a L'-class that belongs to $\mathcal{U}_{L'}^N$. We will denote the latter set as $\langle \mathcal{U}_L^N \rangle$ and by summing up the pmf values of all the elements in $\langle \mathcal{U}_L^N \rangle$ we can quantify the probability p_U that a realization of C_L is unfeasible.

Finally, these probability values provide some information on $p_{FA}(T)$ for a generic sequence. Indeed, a sequence of length T contains $\lfloor T/L \rfloor$ non-overlapping subsequences of length L and, thanks to the independence assumption and the knowledge of p_U, we can compute the probability of finding a given number of L-classes in $\langle \mathcal{U}_L^N \rangle$ among them. This is done be seeing the sequential analysis of the non-overlapping subsequences as a Bernoulli process (where the success is represented by the fact that the corresponding L-class belongs to $\langle \mathcal{U}_L^N \rangle$) and model the number of successes as a binomial distribution with parameters $\lfloor T/L \rfloor$ and p_U. We can then obtain the probability $p_{FA}^L(T)$ of finding only feasible L-classes among the $\lfloor T/L \rfloor$ non-overlapping subsequences in the sequence as follows:

$$\begin{aligned} p_{FA}(T) \le p_{FA}^L(T) &= \binom{\lfloor T/L \rfloor}{0} p_U^0 (1 - p_U)^{\lfloor T/L \rfloor} \\ &= (1 - p_U)^{\lfloor T/L \rfloor}. \end{aligned} \tag{1}$$

As stated in (1), $p_{FA}^L(T)$ is an upper bound of the global false alarm probability $p_{FA}(T)$ as it refers to non-overlapping subsequences only, while overlapping ones could belong to $\langle \mathcal{U}_L^N \rangle$ as well; moreover, unfeasible classes for N with a higher length might be present in the sequence. However, such a result allows us to find a value of T for which the false alarm probability is certainly lower than a desired value.

In Fig. 6, we report the results of such kind of analysis for two pair of N and L values: for $N = 3$ we considered $L = 5$ (which excludes the possibility of longer unfeasible sequences) and for $N = 5$ we considered $L = 7$, as higher values would imply a significantly higher computational cost. For each sequence of each dataset we considered its original version

TABLE IV: Percentage of positives for the $\{N\}$-detectors when a different window size is used in the filtering.

(a) **Music**

	$N=3$	$N=5$	$N=7$	$N=9$
Median 3	100%	0%	0.%	0%
Median 5	0%	100%	0.4%	100%
Median 7	0%	0%	100%	100%
Median 9	0%	0%	0.2%	100%

(b) **Speech**

	$N=3$	$N=5$	$N=7$	$N=9$
Median 3	100%	0%	7.0%	85.3%
Median 5	0%	100%	1.0%	78.0%
Median 7	0%	0%	100%	78.0%
Median 9	0%	0%	0%	100%

(c) **Stock**

	$N=3$	$N=5$	$N=7$	$N=9$
Median 3	100%	0.3%	0.1%	0%
Median 5	0.4%	100%	6.8%	100%
Median 7	0.3%	16.9%	100%	100%
Median 9	0.7%	25.7%	45.7%	0%

(d) **Ecg**

	$N=3$	$N=5$	$N=7$	$N=9$
Median 3	100%	0%	0%%	0%
Median 5	0.9%	100%	20.8%	100%
Median 7	0%	13.4%	100%	100%
Median 9	0.1%	11.5%	27.8%	0%

(denoted as Original) and we created four other sequences, resulting from the application of a median filter with a varying window size M, respectively denoted as Median M for $M = 3, 5, 7, 9$. Then, we estimated the pmf in each case by randomly collecting 500000 subsequences of the chosen length and creating a normalized histogram of the corresponding possible L-classes (see Fig. 6, where the pmf estimation for the Original sequences in the case $N = 3$ and $L = 5$ is represented for each dataset).

By knowing $\mathcal{U}_5^3$ and $\mathcal{U}_7^5$, we identified the 252 5-classes in $\langle\mathcal{U}_5^3\rangle$ and the 33232 7-classes in $\langle\mathcal{U}_7^5\rangle$. Then, we computed the value of p_U in each case and, by means of expression (1), we derived the length value T_{min} which is necessary to guarantee that $p_{FA}(T) \leq 0.01$. It is interesting to notice that, although the pmfs present some differences among the datasets due to the different nature of the data, the general shape of the histogram is quite similar. We also observe in any case a decrease of p_U when a window size $M = 5, 7, 9$ is used, while it is clearly null when $M = N$. However, the value of T_{min} turns out to be quite low (not higher than 8000 in each case) for both the values of N considered, and allows for acceptable results in terms of detection, as it will be explored in the following experiments.

B. Filter detection

After assessing the false alarm probability, we applied the $\mathcal{N}$-detectors (as described in Section III-B) for different $\mathcal{N}$ to the sequences of all the datasets (assuming to stop the search as soon as one unfeasible class for each element of $\mathcal{N}$ is found) and tested also the effectiveness of the proposed approach in identifying and discriminating median filtering with respect to other processing. In fact, in addition to the Original and Median M cases, for each sequence of each dataset we also created other two versions, resulting from the application of a moving average filter (with window size 3) and a Gaussian lowpass filter (with window size 3 and standard deviation equal to 0.5), respectively denoted as Moving Average and Gaussian Lowpass.

First, we applied a $\{3, 5, 7, 9\}$-detector, thus using all the unfeasible classes that we derived in Section III-A. As we specified in Section III-B, the algorithm checks the presence of unfeasible classes for $N = 3, 5, 7, 9$ and the sequence is classified as positive if only feasible classes for at least one value of N are detected, while it is classified as negative if at least one unfeasible class for each value of N is found. Clearly, the Original, Moving Average and Gaussian Lowpass cases should be negative to such detector, while the Median M cases should be positive. As expected from the theory, the rate of false negatives was null in all tests, so in Table II we report the results only on sequences that were not median filtered in terms of false alarm, which is the only type of error that might occur. By observing the results, we can notice that we also have a null rate of false positives both in the Original row (as we predicted in Section IV-A) and the Moving Average/Gaussian lowpass rows, thus showing the ability of the method in distinguishing between median filtering and other kind of processing.

Regarding the complexity and computational time of the detection, the search will be more demanding as N increases, as the number and the length of unfeasible classes increase as well (for $N = 9$, a total number of 74598 unfeasible classes need to be checked with a length up to 11), but the first unfeasible class is usually detected very soon. In Table III, we report the average computational time in seconds necessary to process 100 samples, showing the short time frame for the analysis of not filtered sequences (about 6 milliseconds for a 30 second long audio track).

As further analysis, we applied the detectors for a specific N to sequences that have been median filtered with a window size $M \neq N$, thus evaluating the ability of the technique to discriminate the size of the filter used. For instance, sequences filtered with $M = 3$ should be positive to a $\{3\}$-detector and negative to the $\{5\}$-, $\{7\}$- and $\{9\}$-detectors, and so on. However, we observed in Section III-A that the sets of unfeasible classes for N and M usually share a number of classes (in particular for $L = 5, 6$), which are certainly removed when a median filter with size M is applied. In addition, the action of a median filter with size $M \neq N$ generally decreases the frequency of occurrence of unfeasible classes for N even though they are feasible for M, as we observed in Section IV-A for $N = 3$. Because of that, the presence of unfeasible classes for N in sequences filtered with size M is less probable, thus affecting the performance of $\{N\}$-detectors. In Tables IV, we report the percentage of sequences classified as positives in the different cases. Similarly as before, the different values of M are reported row wise, while the values of N are placed column wise. We

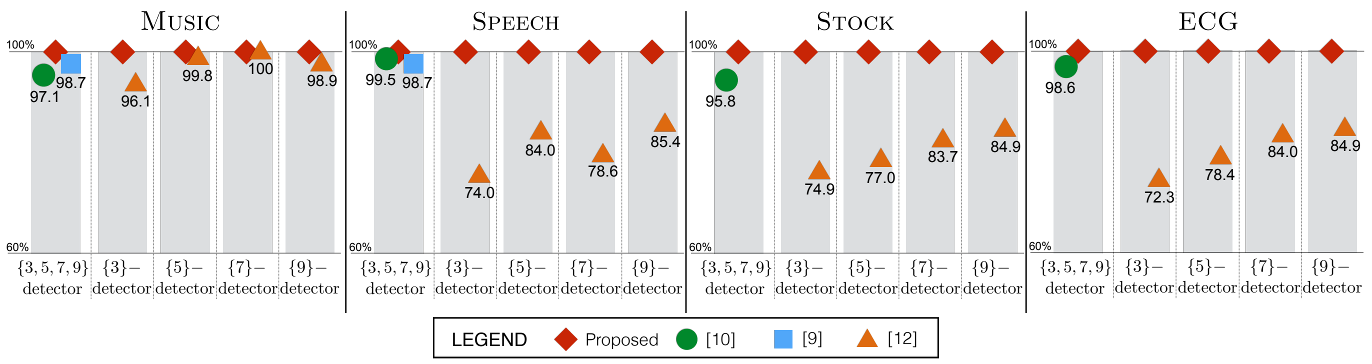


Fig. 7: Accuracies obtained by applying our approach and the existing state-of-the-art methods.

TABLE V: Average computational time in seconds of different methods for 100 samples.

	MUSIC	SPEECH	STOCK	ECG
[10]	$7.4 \cdot 10^{-3}$	$6.5 \cdot 10^{-3}$	$5.9 \cdot 10^{-3}$	$5.8 \cdot 10^{-3}$
[9]	$3.4 \cdot 10^{-3}$	$2.2 \cdot 10^{-3}$	$5.8 \cdot 10^{-3}$	$3.8 \cdot 10^{-3}$
[12]	$6.2 \cdot 10^{-4}$	$1.2 \cdot 10^{-4}$	$1.2 \cdot 10^{-4}$	$1.3 \cdot 10^{-4}$

can observe that the false alarm rate is generally lower than 1% for the $\{3\}$-detector, which is again coherent with the analysis performed in Section IV-A as the length of the sequence is in any case higher than the correspondent T_{min} values reported in Fig. 6. While the false alarm rate is generally acceptable up to $N = 7$, it substantially increases for the $\{9\}$-detector when a median filter with size $M = 5, 7$ is used, thus confirming that such filters tend to remove the unfeasible classes for $N = 9$. However, it is worth pointing out that for the sequences with a substantially higher length (i.e., part of the **SPEECH** dataset), such classes are generally found.

C. *Comparison with state-of-the-art techniques*

We also tested our method against some existing detection techniques. As we stressed in Section I, no specific methods for 1D data exists, thus we adapted three different state-of-the-art approaches that were originally conceived for images (2D median filtering), but their rationale can be applied also to 1D data. In particular, we considered the techniques proposed in [9], [10] and [12]. All of them require some training phase, as the former two are threshold-based and the latter employs an SVM classifier. Thus, we divided each dataset in two equal parts, one used for the training and the other one for the testing phase. For the methods in [9] and [10], the optimal threshold was determined by fixing a maximum false alarm rate of 5% and choosing the threshold value yielding the lowest false negative rate, while for the method in [12] we trained the SVM classifiers as suggested in the original paper.

Because of the different properties of each method, we could perform a comparison only for certain experimental scenarios. Indeed, the detectors [9] and [10] are not targeted to a specific filter size but they indicate that a generic median filter operation has been applied, thus they have been compared with the $\{3, 5, 7, 9\}$-detector. On the other hand, the technique

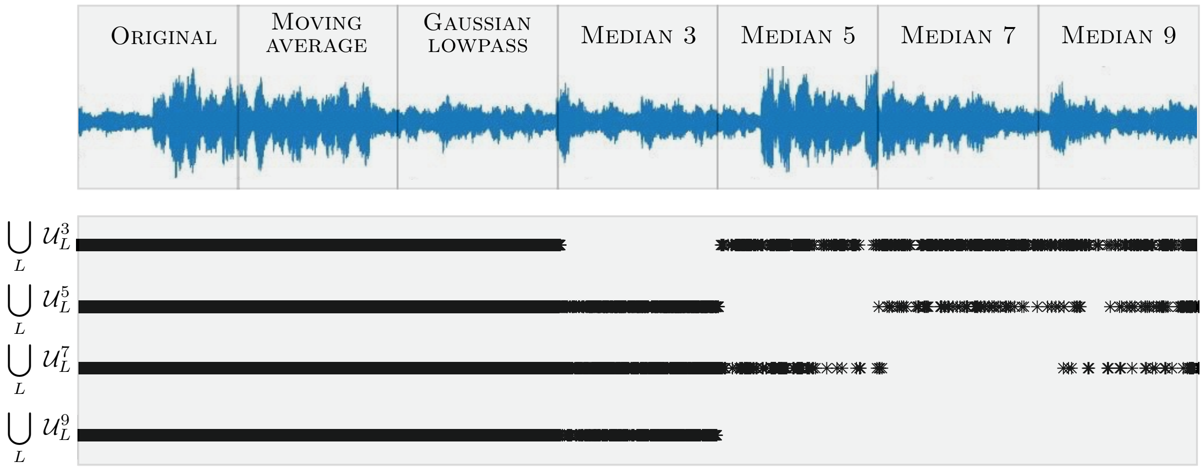


Fig. 8: Example of median filter localization with the $\{3, 5, 7, 9\}$-detector. The black asterisks are the unfeasible classes detected through the whole 1D signal.

in [12] is able to discriminate among different values of N and it has been compared with different $\{N\}$-detectors. Moreover, the detector in [9] can be applied only on data containing integer values, as it is based on an histogram bin ratio, thus in our experiments we could employ it only for the audio datasets. According to this considerations, in Fig. 7 we report the results obtained when applying the different detectors on non median filtered sequences (including original, average filtered, Gaussian lowpass filtered) and median filtered with the same window as the detector. Here, we report the accuracy value (meant as the percentage of sequences correctly classified as filtered or not filtered) in the different cases, as false negatives are also possible for the other methods. In this setting, we can observe that our technique achieves the maximum accuracy in any case. The performance of [10] and [9] is also good in the corresponding scenario, while we can notice that the technique in [12] has a different behaviour throughout the different datasets. Moreover, in Table V we report the average computational time that is necessary to each technique to process (i.e., extracting the features) 100 samples from the different datasets.

D. Tampering localization

In this section, we exploit the proposed detection algorithm to locate parts of the 1D data that have been median filtered. In particular, we partition each sequence into seven non-overlapping parts, each of them processed according to the different operations employed in the previous experiments (ORIGINAL, MOVING AVERAGE, GAUSSIAN LOWPASS, MEDIAN with window size $3, 5, 7, 9$, applied in this order). In this phase, we limited the analysis to a subset of the **MUSIC** and **SPEECH** dataset and applied each processing operation to a signal part of 30000 samples.

We then run the $\{3, 5, 7, 9\}$-detector through the whole sequence (i.e., not stopping at the first unfeasible class detected but analyzing the sequence entirely) and perform a local analysis by considering smaller non-overlapping blocks of 5000 samples. In particular, we classify each of them as positive to median filtering if only feasible classes for at least one values of N lie within it (i.e., blocks belonging to the first three parts are true negatives and the ones belonging to the last four parts are true positives). In Fig. 8, we graphically represent the behaviour of the detector by means of an exemplifying test sequence. The black asterisks below the signal are the unfeasible classes detected through the whole sequence for the different values of N; clearly, no unfeasible classes for a certain N are detected in the part of the sequence median filtered with window size N. Also in this case, we obtain 0% false negatives in every sequence, which means that at least one unfeasible class is detected in each one of the not median filtered blocks. Moreover, we can notice that the unfeasible classes for any N are dense in the first three parts, while they are more sparse when another filter size is used. In particular, coherently with the results in the previous section, no unfeasible classes for $N = 9$ are found in the last three parts, although this does not affect the performance of the $\{3, 5, 7, 9\}$-detector.

E. Robustness analysis

In this section, we deal with the issues of assessing the robustness of our method with respect to post processing, i.e., we consider the problem of detecting median filtering even when a further successive operation is applied after it. Firstly, we can observe that our technique is based on the order relationships between the samples and it is robust to any post-processing that preserve them, such as amplitude re-scaling, monotonic corrections, normalization, shifting. With regards to other operations, it is easy to state that when the median filter has not been the very last process, the method loses its deterministic nature since the post processing would potentially introduce unfeasible classes. However, we can notice that in such situation the effect of the previous median filtering is still visible in the distribution of the unfeasible classes detected in the sequence. In particular, we considered the sequences of the different datasets with no processing at all (denoted again as ORIGINAL) and we processed the same sequences by first applying a median filter with $N = 3$ or $N = 5$ followed by a post processing operation among moving average and Gaussian lowpass filter, thus obtaining four different scenarios. Then, we run the respective $\{N\}$-detector on all the sequences and we created an histogram of all the unfeasible classes for that value of N, normalized by the total number of unfeasible classes detected. In particular, for $N = 3$ the histogram has $|\mathcal{U}_4^3| + |\mathcal{U}_5^3| = 32$ bins and for $N = 5$ it has $|\mathcal{U}_5^5| + |\mathcal{U}_6^5| + |\mathcal{U}_7^5| + |\mathcal{U}_8^5| + |\mathcal{U}_9^5| = 638$ bins.

A possible approach is to use the histogram bin values as features and feed a classifier. Hence, as we did for the state-of-the-art approaches, we divided each dataset in two equal parts, one for training and one for testing, and we extracted features from each training set. Then, for each of the four scenarios we trained an SVM classifier with Gaussian kernel (optimal parameters have been derived by means of a grid search procedure) using features from all the datasets. Finally, we performed a classification for the testing set of each dataset in the different scenarios. In Table VI, we report the accuracies obtained. We can observe that the accuracy values are higher than 90% almost in any case and, as expected, the performance are better for the case of $N = 5$ and the Gaussian lowpass filter, which has less impact than the moving average filter. In order to have a comparison with state-of-the-art methods, we considered the approach in [12] (the only one which detects median filter with a specific value of N) and report in Table VII the results obtained by replicating the same experimental settings, showing a clear performance drop in case of post-processing and worse accuracies with respect to the proposed method.

V. CONCLUSIONS

We have proposed a forensic detector of median filtering on 1D data based on deterministic properties of such processing operation. According to a well defined theoretical rationale, a set of patterns that cannot be present in median filtered sequences have been computed offline and the final algorithm consists in searching such patterns in the test sequence. The proposed method has been tested on 1D signals and time series

TABLE VI: Classification accuracy by means of SVM with post-processing.

	MUSIC	SPEECH	STOCK	ECG
ORIGINAL vs MEDIAN $N = 3$ + MOVING AVERAGE	97.8%	93.7%	92.2%	73.1%
ORIGINAL vs MEDIAN $N = 5$ + MOVING AVERAGE	100%	100%	98.6%	94.4%
ORIGINAL vs MEDIAN $N = 3$ + GAUSSIAN LOWPASS	99.0%	96.8%	96.8%	78.2%
ORIGINAL vs MEDIAN $N = 5$ + GAUSSIAN LOWPASS	100%	100%	97.8%	98.2%

TABLE VII: Classification accuracy by means of method in [12] with post-processing.

	MUSIC	SPEECH	STOCK	ECG
ORIGINAL vs MEDIAN $N = 3$ + MOVING AVERAGE	55.0%	48.4%	50.7%	50.1%
ORIGINAL vs MEDIAN $N = 5$ + MOVING AVERAGE	50.0%	56.2%	74.4%	55.4%
ORIGINAL vs MEDIAN $N = 3$ + GAUSSIAN LOWPASS	62.3%	67.2%	56.7%	60.2%
ORIGINAL vs MEDIAN $N = 5$ + GAUSSIAN LOWPASS	66.2%	76.6%	69.4%	74.2%

coming from different sources, and proved to be extremely accurate in detecting and locating the occurrence of median filtering as well as identifying the size of the filter employed, which is a quite rare feature in existing techniques for 2D median forensics. Moreover, we also proved that the study of the unfeasible classes can be used to detect median filtering also in case of post-processing.

Such promising results open the way for future developments of this work in different directions. A natural step further would be to approach 2D median filtering, which is commonly applied to images. Unfortunately, treating bidimensional data and filters introduces significant problems, both in terms of theoretical results and computational complexity. Indeed, although the theoretical concepts can be easily extended, the derivation of the unfeasible classes presents two main issues: due to the distribution in the 2D domain, the actual overlapping variables at each step is reduced with respect to the 1D case (at most 6 overlapping variables for a 3×3 filter), together with the chance of encountering unfeasible branches; as a consequence, the number of possible branches in the tree (630 new ones are introduced at each step for a 3×3 filter) and the number of L-classes required is higher, leading to extremely demanding computational efforts. For this reasons, a significant optimization of the technique would be required and will be certainly subject of future work.

Moreover, much work can be developed regarding a further analysis of robustness issues of the proposed method, by designing more specific and advanced approaches allowing for the identification of the median filtering occurrence even after a wider range of post-processing operations.

REFERENCES

[1] J. W. Tukey, *Exploratory Data Analysis*. MA: Addison-Wesley, 1976.

[2] L. R. Rabiner, M. R. Sambur, and C. E. Schmidt, “Application of a nonlinear smoothing algorithm to speech processing,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 23, no. 6, pp. 552–557, 1975.

[3] T. Kasparis and J. Lane, “Adaptive scratch noise filtering,” *IEEE Transactions on Consumer Electronics*, vol. 39, no. 4, pp. 917–922, 1993.

[4] S. Akkoul, R. Lèdèe, R. Leconge, and R. Harba, “A new adaptive switching median filter,” *IEEE Signal Processing Letters*, vol. 17, no. 6, pp. 587–590, 2010.

[5] P. M. Narendra, “A separable median filter for image noise smoothing,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, no. 1, pp. 20–29, 1981.

[6] Y. Liu, C. Liu, and D. Wang, “A 1D time-varying median filter for seismic random, spike-like noise elimination,” *Geophysics*, vol. 74, no. 1, pp. 17–24, 2009.

[7] Y. Wen and B. Zeng, “A simple nonlinear filter for economic time series analysis,” *Economics Letters*, vol. 64, no. 2, pp. 151–160, 1999.

[8] T. Pander, “The class of M-filters in the application of ECG signal processing,” *Biocybernetics and Biomedical Engineering*, vol. 26, no. 4, pp. 3–13, 2006.

[9] M. Kirchner and J. Fridrich, “On detection of median filtering in images,” in *Proceedings of SPIE*, vol. 7541, 2010, pp. 101–112.

[10] G. Cao, Y. Zhao, R. Ni, and L. Yu, “Forensic detection of median filtering in digital images,” in *IEEE International Conference on Multimedia and Expo*, 2010, pp. 89–94.

[11] H. Yuan, “Blind forensics of median filtering in digital images,” *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 4, pp. 1335–1345, 2011.

[12] X. Kang, M. Stamm, A. Peng, and K. R. Liu, “Robust median filtering forensics using an autoregressive model,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 9, pp. 1456–1468, 2013.

[13] C. Chen, J. Ni, and J. Huang, “Blind detection of median filtering in digital images: a difference domain based approach,” *IEEE Transactions on Information Forensics and Security*, vol. 22, no. 12, pp. 4699–4710, 2013.

[14] Y. Zhang, S. Li, S. Wang, and Y. Q. Shi, “Revealing the traces of median filtering using high-order local ternary patterns,” *IEEE Signal Processing Letters*, vol. 21, no. 3, pp. 275–280, 2014.

[15] N. Gallagher and G. Wise, “A theoretical analysis of the properties of median filters,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 6, 1981.

[16] D. Vàzquez-Padìn, P. Comesana, and F. Pèrez-Gonzàlez, “Set-membership identification of resampled signals,” in *IEEE Workshop on Information Forensics and Security*, 2013, pp. 150–155.

[17] J. Fridrich, M. Goljan, and R. Dui, “Steganalysis based on JPEG compatibility,” in *SPIE Multimedia Systems and Applications*, 2001, pp. 275–280.

[18] G. Tzanetakis and P. Cook, “Musical genre classification of audio signals,” *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[19] (2006). [Online]. Available: http://groups.inf.ed.ac.uk/ami/corpus/

[20] [Online]. Available: http://www.stockhistoricaldata.com

[21] G. Moody and R. Mark, “The impact of the MIT-BIH Arrhythmia Database,” *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[22] S. Greenwald, “Improved detection and classification of arrhythmias in noise-corrupted electrocardiograms using contextual information,” Ph.D. dissertation, Harvard-MIT Division of Health Sciences and Technology, 1990.

[23] A. Taddei, G. Distante, M. Emdin, P. Pisani, G. Moody, C. Zeelenberg, and C. Marchesi, “The European ST-T Database: standard for evaluating systems for the analysis of ST-T changes in ambulatory electrocardiography,” *European Heart Journal*, no. 13, pp. 1164–1172, 1992.