Efficient computation of image moments for robust cough detection using smartphones

Hoyos Barceló, Carlos; Monge-Álvarez, Jesús; Pervez, Zeeshan; San-Jos´e-Revuelta, Luis M.; Casaseca, Juan Pablo

Published in:
Computers in Biology and Medicine

DOI:
10.1016/j.compbiomed.2018.07.003

Published: 01/09/2018

Document Version
Peer reviewed version

Efficient Computation of Image Moments for Robust Cough Detection using Smartphones

Carlos Hoyos-Barceló[a], Jesús Monge-Álvarez[a], Zeeshan Pervez[a], Luis M. San-José-Revuelta[b], Pablo Casaseca-de-la-Higuera[a,b,*]

[a] *School of Engineering and Computing, University of the West of Scotland, Paisley Campus, High Street, Paisley, PA1 2BE, Scotland, United Kingdom.*

[b] *ETSI Telecomunicación, Dpto. Teoría de la Señal y Comunicaciones e Ingeniería Telemática, 47011 Valladolid, Spain.*

Abstract

Health Monitoring apps for smartphones have the potential to improve quality of life and decrease the cost of health services. However, they have failed to live up to expectation in the context of respiratory disease. This is in part due to poor objective measurements of symptoms such as cough. Real-time cough detection using smartphones faces two main challenges namely, the necessity of dealing with noisy input signals, and the need of the algorithms to be computationally efficient, since a high battery consumption would prevent patients from using them. This paper proposes a robust and efficient smartphone-based cough detection system able to keep the phone battery consumption below 25% (16% if only the detector is considered) during 24h use. The proposed system efficiently calculates local image moments over audio spectrograms to feed an optimized classifier for final cough detection. Our system achieves 88.94% sensitivity and 98.64% specificity in noisy environments with a 5500× speed-up and 4× battery saving compared to the baseline implementation. Power consumption is also reduced by a minimum factor of 6 compared to existing optimized systems in the literature.

Keywords: Event detection, cough detection, mobile health, moment theory, optimization.

1. Introduction

Ever since the arrival of smartphones, many realized their potential as tools for delivery of healthcare services. *Mobile health* (*m-Health*) is an emerging field that advocates the use of mobile devices for health applications, aiming to provide assistance to patients from underdeveloped regions, and harnessing this technology for more efficient management of medical conditions [1]. Despite this potential, m-Health has failed to

*Corresponding author.

Email addresses: carlos.hoyos@uws.ac.uk (Carlos Hoyos-Barceló), jesus.monge@uws.ac.uk (Jesús Monge-Álvarez), zeeshan.pervez@uws.ac.uk (Zeeshan Pervez), lsanjose@tel.uva.es (Luis M. San-José-Revuelta), casaseca@lpi.tel.uva.es (Pablo Casaseca-de-la-Higuera)

live up to expectation in the context of respiratory illnesses, in part because of poor objective measures of symptoms, which could lead to early diagnosis or prevention [2].

Cough is the most common symptom in respiratory diseases, and an important clinical sign. By listening to cough, an expert physician may extract extremely valuable information for their treatment, both from quantitative (e.g. frequency or intensity) and qualitative (e.g. dry or wet cough) assessment.

However, patient-filled reports on cough frequency and intensity have proven to be grossly inaccurate, as chronic coughers become unaware of their coughs [3]; besides, listening to a recording to locate coughs by manual methods is a long and tedious task that raises privacy concerns. For this reason, there has been a call for automatic methods to speed it up. The medical community published guidelines for such systems, emphasizing the need for them to be automatic, unobtrusive, compact, private, and allow for full-day recordings [4].

It is in the context of remote monitoring as described above where m-health can play an important role, especially for patients with reduced mobility or chronic respiratory illnesses. An intelligent monitoring app would unobtrusively collect cough data by analysing the audio signal recorded with the smartphone, and provide medical experts with rich information on which to base their diagnosis. If this app runs seamlessly without interfering the regular use of the smartphone, patients would be less conscious of the medicalization of their lives.

An additional advantage of monitoring apps resides in their potential to decrease the cost of national health services. In the case of lung diseases, this cost has been estimated to be approximately 96.4 billion euros per year in the EU, plus another 283 billion euros in opportunity costs [5]. One reason for these steep costs is that patients only resort to physicians after their condition notably impairs their quality of life, requiring expensive care, and resulting in productivity loss [6]. Monitoring of early signs makes preemptive diagnosis possible, so physicians can prescribe simpler treatments while they are still effective [3].

However, m-Health monitoring apps have many challenges to overcome before they fulfill their potential. Namely, they need to process large streams of data in a timely fashion to achieve real-time performance; they need to be reliable in noisy environments, especially when audio signals constitute the information source; and they also need to sustain full-day monitoring sessions. For cough detection, the system has to process the input signal from the microphone, then extract a set of features that sets cough events apart from other sounds (the *feature vector*), and after that use a *pattern recognition* engine to classify those events as cough or non-cough events. This process is computationally expensive and leads to high battery consumption. Thus, smartphone implementations are usually basic and riddled with battery issues, with many systems off-loading to an external server to complete the task [1, 3, 7].

Most of the automatic cough detection systems proposed so far [8, 9, 10, 11] achieve acceptable sensitivity and specificity values in non-ambulatory, reduced noise environments. They usually rely on spectral features such as MFCC (*Mel Frequency Cepstral Coefficients*) [12] or GTCC (*GammaTone Cepstral Coefficients*) [13] typically used for audio event recognition. These feature sets provide a logarithmic characterization of the signal spectrum to emulate the response of the human hearing systems. Other methods are conceived to identify cough events from a series of pre-selected audio events rather than to detect coughs in real time. Murata et al. [14] employed six different smoothed

time-domain envelopes to identify sound events as coughs depending on their similarity to those templates. They achieved sensitivity and specificity values above 90% in a quite environment. Shin et al. [15] fed spectral (energy cepstral coefficients, ECC) and time features (signal envelope) into a hybrid model composed of an Artificial Neural Network and a Hidden Markov Model (HMM) to classify pre-selected sound events contaminated with pink noise with Signal to Noise Ratios (SNR) ranging from -10 to 30 dB.

The above mentioned systems are not optimized to perform in challenging noisy scenarios. Actually, portable cough detectors such as [16] have been shown to perform poorly in ambulatory environments. Our recent work [17] proposed a methodology to apply local invariant image moments to audio signal spectrograms for robust cough detection in such environments. We also demonstrated in [18] that local Hu Moments [19] outperformed conventional feature sets for cough detection even when the Signal-to-Noise-Ratio (SNR) was so low (-6 dB) that the amount of present noise hindered the signal of interest. Sensitivity and specificity values for those systems were respectively 86% and 99%.

There have also been some attempts to perform cough detection using smartphone apps. Some of them, such as *CoughSense* [3], or the cough detector implemented in *ResApp* [20, 21, 22] are not intended for continuous monitoring in noisy environments. Instead, the user would need to run the app and cough to it on a controlled measuring environment. In terms of computations, the former records the sound in a smartphone that the user wears around the neck, uploading the extracted features to a cloud server that performs cough detection. This dependence on a cloud server limits the app in places with no internet connection, plus any kind of offline processing requires storing all data on the device, which takes too much space to be practical for low-end smartphones.

On the other hand, systems like *ADAM*, that run on smartphones, report fast shortening of battery life [7]. This constitutes an issue since users will be reluctant to install and use it. One common method to reduce battery consumption is to implement a basic power threshold, a lightweight detector that discards those windows that do not contain audio events, so the use of the costly feature extraction and pattern recognition engines is minimized. One system implementing such filters is the *SymDetector* app, managing to extend the battery life to 20 hours [23]. However, to warrant seamless functionality with low battery drainage in regular use, the system also needs to be implemented in an efficient manner. Our work in [24] proposed an efficient implementation of the pattern recognition module proposed in [18] so that detection could also be performed in real time and battery consumption reduced. This paper focuses on the feature extraction part, which for local invariant image moments applied to cough detection, can significantly be optimized.

This paper proposes an optimized smartphone-based cough detection system able to keep the phone battery consumption below 25% (16% if only the detector is considered) during 24h use. The proposed system efficiently calculates local image moments over audio spectrograms to feed an optimized classifier for final cough detection. The contribution of the paper resides in the proposed optimizations for moment calculation, which enable fast and battery-friendly cough detection providing full functionality for long periods of activity even on low-end devices.

2. Materials

Our database consists of 78 ambulatory recordings from 13 patients of ages 45-72 with different respiratory conditions (Chronic Obstructive Pulmonary Disease –COPD–, bronchiectasis and asthma), acquired at the Outpatient Chest Clinic of the Royal Infirmary of Edinburgh (UK), adding up to 1,560 minutes of audio. The study was carried out in accordance with the Declaration of Helsinki and was approved by the NHS Lothian Research Ethics Committee (REC number: 15/SS/0234). Subjects provided their informed consent before the recordings. Patients were required to speak or read aloud, and their coughs would get recorded as they spontaneously arose. They were also asked to produce other sound events such as laughing, swallowing, throat clearing, or heavy breathing. The recording took place in three segments of parts, each one emulating a different real-life scenario. 2×20 minutes files were recorded in each part.

- **Part I**: Clear recordings in a low-noise environment. The patient is sitting and is requested to speak or read aloud. From time to time, we asked the patient to produce other foreground events such as throat clearing, swallowing (by drinking a glass of water), blowing nose, sneezing, breathless breathing or laugh.
- **Part II**: The second part emulated a noisy environment with a external source of contamination, i.e., the noisy background sounds are not produced by the patient. To do so, we repeated the same experiment as in part one with either a television set or radio player on. Besides, the door was left open so that noisy sounds from the corridor of the hospital were recorded as well. These interferences included trolleys, phones ringing, babble noise, typing noise, etc.
- **Part III**: Finally, the third part was designed to represent noisy environments where the own patients become also a source of contamination because of their movements and other activities. In this case, the patient could move freely around the room while we asked her to perform some activities like turning on/off the radio, opening/closing the window, opening/closing a drawer, moving a chair, washing hands, lying on the bed and standing up immediately, typing, putting on the coat and taking it off immediately, picking up something from the floor, etc. As in part two, the door was left open. Equally, while the patient was performing these activities, we requested her to produce other foreground events as in the first and second parts.

Signals were recorded at 44.1 kHz in *.wav* format, with 16 bits per sample. Patients placed the smartphone into their pocket or handbag for all three parts and were asked to carry it during the whole protocol. The percentage of cough samples across the entire dataset was 1.66%.

In order to obtain accurate annotations for the recordings, the technical research team manually annotated the audio recordings using Praat [25], a scientific software tool for analysing phonetic sounds and annotate them. The annotated recordings were subsequently reviewed by two members of the clinical team, a general practitioner, and a respiratory consultant. When ambiguous sounds required discussion among the researchers, if a conclusion could not be made, the whole event was labelled as unknown and excluded from further analysis.

3. Methods

3.1. Image moments

In image processing and computer vision, *Image moments* are weighted averages (moments) of the intensities of a given image's pixels or a function of such moments. They are useful to characterize objects after segmentation and are usually chosen so that they have a defined physical interpretation, capturing simple image properties such as areas, centroid, or total mass. The (i, j)-th order image moment is defined as

$$M_{ij} = \sum_x \sum_y x^i y^j g(x, y) \tag{1}$$

Equation (1) shows the function that computes the (i, j)-th order moment for a discrete image, where $g(x, y)$ represents the intensity value of the pixel at the (x, y) coordinate. The zeroth-order moment, M_{00}, represents the total mass (or power) of the image, whereas M_{20} and M_{02} describe the moments of inertia with respect to the coordinate axes [26]. Different image objects have different moment values, meaning that they can be used as basic descriptors.

One shortcoming of *Cartesian* (also called *raw* or *Geometric*) *moments* is that simple transformations of the image –such as translation, scaling, or rotation– result in different moment values; pattern recognition engines based on those will not treat transformed images as the same object. This led to the development of *Image moment invariants*, moment functions that are impervious to simple transformations:

- **Translation invariance** is obtained by setting the centroid of the image as the origin of coordinates; these are called *Central moments* and the discrete function is

$$\mu_{ij} = \sum_x \sum_y (x - \bar{x})^i (y - \bar{y})^j g(x, y) \tag{2}$$

  where $\bar{x} = \frac{M_{10}}{M_{00}}$ and $\bar{y} = \frac{M_{01}}{M_{00}}$ denote, for each axis, the points at where the image centroid is located.

- **Scale invariants** can be constructed by dividing the *central moments* by a properly scaled central moment, usually μ_{00} [26]. The moments computed according to the following expression are both invariant to translation and to proportional scaling:

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{1+\frac{i+j}{2}}} \tag{3}$$

- **Rotation invariance** was introduced by Hu, who presented a set of six moments that were invariant to translation, scale, and rotation [27], and demonstrated that his image moments could be effectively used in pattern recognition problems, even when noise was present in the image. He presented also a seventh one that was invariant to skew, which he used to detect mirror images. These moments are:

$$
\begin{aligned}
I_1 &= \eta_{20} + \eta_{02} \\
I_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \ldots \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\
I_6 &= (\eta_{20} - \eta_{02}) \left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \ldots \\
&\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] \ldots \\
&\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03}) \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]
\end{aligned}
$$

The first *Hu Moment* is equivalent to the *moment of Inertia* around the image's centroid, with rotation invariance being achieved by the rest.

3.2. Image moments for audio event recognition

Sun et al. were the first to propose the use of Image Moments as features for speech emotion recognition [19]. The rationale behind this was that the first Hu Moment Invariant appropriately characterizes energy distributions, and that the invariance of Hu Moments would make detection more robust to changes between speakers.

As Hu Moments are defined for 2D spaces, this re-purpose requires converting the audio signal to a 2D matrix. This is achieved using a time-frequency representation of the signal, in which one axis denotes time, and the other one frequency.

The audio signal is decomposed in M short segments with some overlap; each one assigned a row in the time axis. Then, these segments are converted to the frequency domain by means of the Fourier Transform, and passed through a series of N Mel filters to quantize the spectrum into N sub-bands, of which the Log Energy is computed to emulate the response of the human hearing system. This results in an $N \times M$ matrix of energy values at each time-frequency position.

Finally, this matrix is divided into square $W \times W$ blocks to calculate their Local Hu Moments coefficients, which characterize the time-frequency components of the audio events at local regions of the spectrogram. These coefficients form the *feature vector* that will be used in classification.

Our work in [17, 18, 28] adopted this approach to perform robust detection of cough events in noisy environments. However, moment computation is still a bottleneck in terms of efficiency, and this motivates our proposal to improve performance so they can be seamlessly implemented in mobile platforms.

3.3. Overall system overview

The pipeline of the basic system is composed of three modules, as shown in Fig. 1. The audio signal is downsampled at 8.82 kHz, which has shown appropriate for cough detection [29]. The samples are then grouped in 50 ms *windows* (441 samples) with 25ms

shift. The reason for doing this is that we are going to transform our non-stationary data locally to the frequency domain. By using overlapped chunks, smooth transitions that resemble time variations are generated. For each window, the input is smoothed

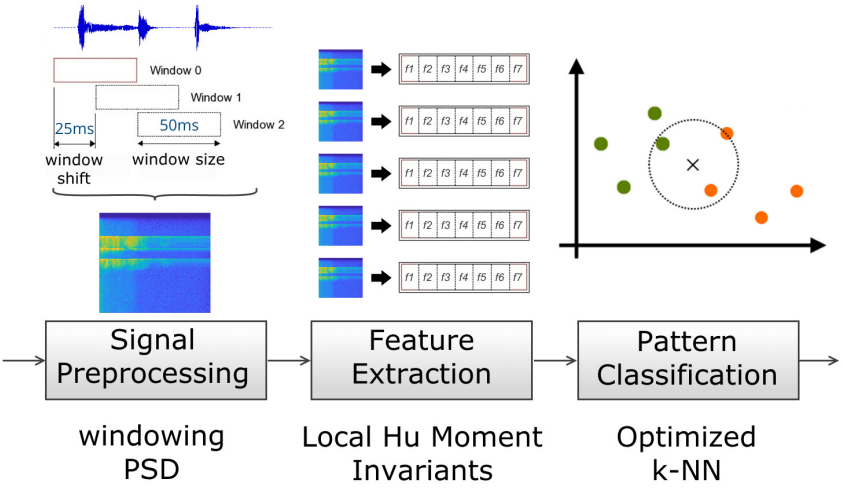

Figure 1: Overview of the proposed cough detector.

using a 441-length Kaiser window ($\beta = 3.5$) before using a 4096-point FFT (*Fast Fourier Transform*) to get the time-spectral representation, where the Power Spectral Density is calculated according to the Wiener-Khinchin-Einstein theorem [30]. Finally, the one-sided $PSD[f]$ is calculated and stored for $1 \leq f < 2049$ using the symmetry properties of the FFT. This $PSD[f]$ is calculated as

$$PSD[f] = \begin{cases} PSD[f] & f = 1 \\ 2 \cdot PSD[f] & 1 < f \leq 2048 \\ PSD[f] & f = 2049 \end{cases} \tag{4}$$

Next, this output is passed through 75 Mel Filter banks (see Fig. 2) to divide it in 75 energy sub-bands. Values are kept in a circular energy matrix E whose size is 75×5, which stores the logarithm of the spectral energies of the last 5 windows. Each new row m of matrix E is computed as

$$E_m(n) = \log\left(\sum_{f=f_{min}}^{f_{max}} PSD[f] \cdot Mel_n[f]\right), \quad 1 \leq n < 75 \tag{5}$$

where the f values correspond to 2049 discrete frequencies in the range $[f_{min}, f_{max}]$, with $f_{min} = 0$ and $f_{max} = 2$ kHz; $Mel_n[f]$ is the n-th filter bank in the Mel scale, which is defined as

$$Mel_n[f] = \begin{cases} \frac{2(f - C_{n-1})}{(C_{n+1} - C_{n-1})(C_n - C_{n-1})} & C_{n-1} \leq f < C_n \\ \frac{2(C_{n+1} - f)}{(C_{n+1} - C_{n-1})(C_n n+1 - C_n)} & C_n \leq f < C_{n+1} \\ 0 & f < C_{n-1}, f \geq C_{n+1} \end{cases} \tag{6}$$

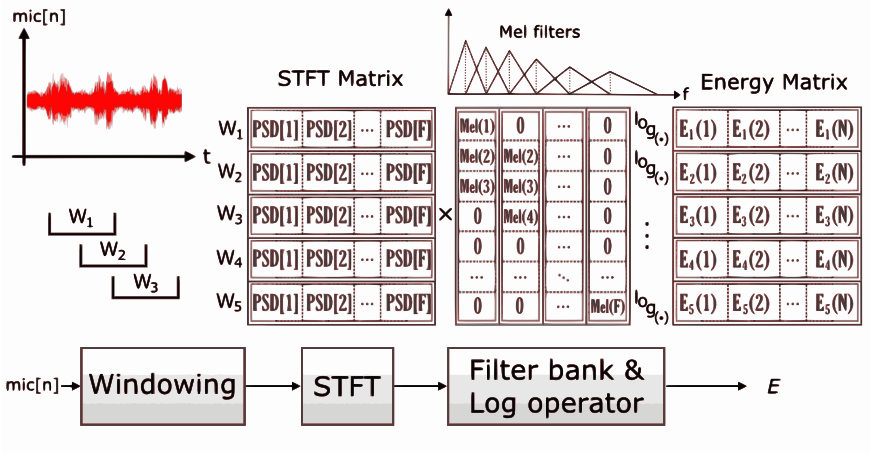

Figure 2: Illustration of the construction of the energy matrix.

where C_n, $0 \leq n < 75$, stands for the central frequencies for each filter in the filter bank, uniformly spaced in the Mel Scale. The following expressions can be used to convert from natural frequencies to Mel scale and vice versa:

$$f[Mel] = 2595 \cdot \log_{10}(1 + f[Hz]/700) \quad (7)$$

$$f[Hz] = 700\left(10^{\frac{f[Mel]}{2595}} - 1\right) \quad (8)$$

The energy matrix E is next segmented into 5×5 data blocks, from which we calculate the first invariant Hu Moment, resulting in a vector of 15 *local* Hu Moment invariants. The Discrete Cosine Transform (DCT-II) is then computed and coefficients $2nd$–$14th$ are kept to constitute the feature vector [18]. This feature vector gets passed as an input to an optimized k-NN classifier to detect cough events in real time [24].

3.4. Optimizations

3.4.1. Optimization of energy matrix calculation

Computing a new row of the energy matrix E requires passing the PSD array containing 2,049 elements through 75 Mel Filters given by Eq. (5), resulting in a sum of 153,675 multiplications. However, as most values in the Mel filters –calculated using Eq. (6)– are zero, we can speed-up this step by only iterating through the non-zero elements of the filters. By doing so, we only have to sum 1,754 multiplications corresponding to the band of the PSD overlapping with the non-zero range of the Mel-Filters.

3.4.2. Removal of unwanted invariances

Whereas invariance to translation, scaling, and rotation is desirable for 2D image detection, we deem it unnecessary and even detrimental for audio event detection. In the case of face recognition, finding a match at one end of a segmented image is as good as finding it at the other end of the spatial axis. However, in a spectrogram, an event producing a pattern near the origin of the frequency axis is different from the

one producing the same pattern at another frequency. So, if we use moments that are translation-invariant in that axis, they will erroneously assign the same value to different objects. To avoid this, Eq. (2) is simplified so as to remove $\bar{x}$ alignment by setting the order in the frequency axis to zero, since x denotes the frequency axis in our matrix:

$$\mu'_i = \sum_x \sum_y (y - \bar{y})^i g(x, y) \tag{9}$$

Besides, while it is common for images to appear scaled or rotated, the equivalent spectral transformations in a time-frequency matrix do not occur naturally [31] (see Figure 3). As such, these invariances rarely matter, and can be removed to improve efficiency. The only invariances of interest are *translation invariance in the time axis*, and *invariance to*

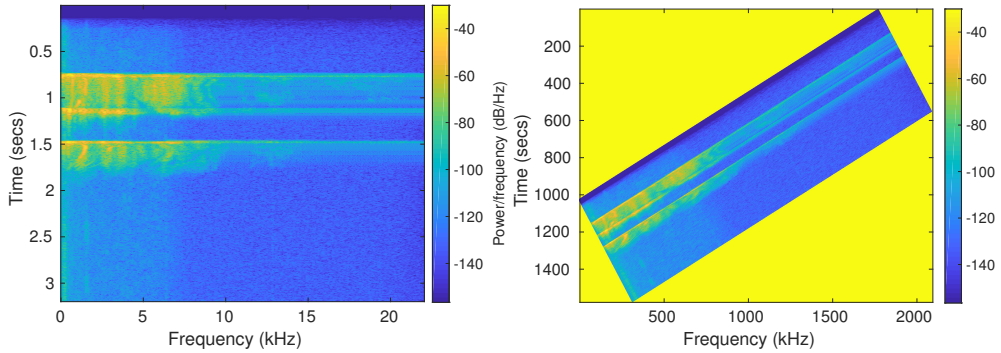

Figure 3: Illustration of spectrogram rotation.

contrast, which is reported to provide invariance against noise in gray-level images [32].

Contrast invariance can be achieved simply by normalizing each central moment by μ_{00} [26, 32]. Since $\mu_{00} = \mu'_0$, our proposed invariant becomes

$$I = \frac{\mu'_2}{\mu'_0} \tag{10}$$

3.4.3. Incremental computing of invariants

The 25ms shift between windows means that there is a 50% overlap of data between one window and the next. We apply incremental computing to calculate the invariants of new windows with values obtained from the previous ones, halving the amount of work required to complete the task.

The moment function is transformed by means of the *Pascal triangle* [33] in order to separate its variables, so that they can be incrementally computed between windows. The function to compute the i-th local moment for $1 \leq i \leq 15$ is given by

$$I_i = \frac{M_{i2} - \frac{M_{i1}^2}{M_{i0}}}{M_{i0}} \tag{11}$$

where $M_{ik} = \sum_x \sum_y y^k B_i(x, y)$ and $B_i(x, y)$ represents the i-th 5×5 block of the energy matrix E, with $y = 1$ being the oldest row timewise, and $y = 5$ the most recent one. Next,

the new values of the variables are incrementally computed using their previous values as a base. Updating M_{i0} is a matter of subtracting the sum of the oldest row, and adding the sum of the newest one –see Eq. (12). The remaining variables (given $M_{i0_{new}} \neq 0$, which would be an empty matrix) are calculated as

$$M_{i0_{new}} = M_{i0_{old}} - \sum_x B_{i_{old}}(x, 1) + \sum_x B_i(x, 5) \quad (12)$$

$$M_{i1_{new}} = M_{i1_{old}} - M_{i0_{old}} + 5 \sum_x B_i(x, 5) \quad (13)$$

$$M_{i2_{new}} = M_{i2_{old}} - 2M_{i1_{old}} + M_{i0_{old}} + 25 \sum_x B_i(x, 5) \quad (14)$$

$$I_{i_{new}} = \frac{M_{i2_{new}} - \frac{M_{i1_{new}}^2}{M_{i0_{new}}}}{M_{i0_{new}}} \quad (15)$$

For the sake of completeness, we also present the formulas for incremental computing of μ_{20} for the i-th block, which would be analogous to Eq. (11) but replacing M_{ik} by N_{ik}:

$$N_{ik} = \sum_x \sum_y x^k B_i(x, k) \quad (16)$$

$$N_{ik_{new}} = N_{ik_{old}} + \sum_x x^k B_i(x, 5) - \sum_x x^k B_{i_{old}}(x, 1) \quad (17)$$

Incremental computing is subject to floating-point rounding errors that accumulate and propagate between windows; this issue is solved by zeroing all partial factors when an empty matrix is detected. Checking for zeroness in IEEE 754 floating numbers is nontrivial, as operations inherently round to the closest representable value, which is itself a function of the values and the mantissa of both operands [34]. This means that simply checking if the value is 0.0 will fail in most scenarios, and the algorithm should rather check if the value falls inside of an interval $0.0 \pm \epsilon$.

Due to floating mantissa and error propagation issues, the value ϵ to use depends on the range of values of the operands, the operations to perform, and the *pu* (precision unit) of the machine [35]. In our current setup, we estimated empirically that $\epsilon = 1e-11$.

3.4.4. Mask uninteresting windows

We implement two adjustable thresholds, one based on dB energy levels, and the other on the *Zero-Crossing rate*, which is the number of times the signal changes sign inside the window:

$$ZCR = \frac{\sum_{i=2}^{N} |sign(s_i) - sign(s_{i-1})|}{N} \quad (18)$$

Windows that fall below any of the thresholds get marked as silent and zeroed, omitting the PSD calculation.

If the system detects 5 silent windows in a row, it will no longer call the feature extractor or the classifier until the next non-silent window arrives, as the energy matrix E is all zeros. This prevents the system from doing useless work. As coughs are particularly energetic and rare events, if thresholds are set properly, the majority of windows will be skipped without missing any cough events, saving a considerable amount of battery.

3.4.5. Exploitation of translation-invariance in time

Hu Moments were meant to be used in images with *masks* that set pixel values around objects to zero; without those masks, invariance properties do not manifest. The process of zeroing all values in the energy matrix E that are not part of a sound event is called *segmentation*. Segmentation is performed when the aforementioned filters zero the windows falling below the chosen thresholds. This way, masks are effectively created around the objects.

Proposed local moments are translation-invariant in the time axis; this means that they return the same values when applied to the same segmented object shifted in time. In this case, its computing can be avoided again, as they will not change; likewise, calling the classifier in this case is redundant, as the predicted class will be the same one as the unshifted object.

An object is shifted in time when the oldest row in the circular energy matrix is all zeros, and the new window that arrives is silent, as well. For the shortest type of events, this simple optimization removes 80% of computations.

The only updates required are $M_{i2} = M_{i2} - 2M_{i1} + M_{i0}$, and $M_{i1} = M_{i1} - M_{i0}$, to indicate a shift in time. Substituting variables, it is easy to verify that $I_{i_{new}} = I_{i_{old}}$.

3.5. Complexity analysis

Figure 4 shows the flowchart of the proposed optimized algorithm compared to the baseline implementation. For each stage of both algorithms, results of a complexity analysis are presented in terms of the different parameters involved to classify a window as cough or non-cough: window size ($W_{size} = 441$), # of FFT points ($N_{FFT} = 4096$), # of Mel filters ($k = 75$), # of local filter blocks ($b = 15$), size of each block ($b_{size} = 5$), size of the feature vector $v_{size} = 13$, and size of the database N_{DB}. The following improvements can be highlighted for individual stages:

- Exploiting window overlap halves the number of operations needed to check power or zero-crossing thresholds. $O(W_{size}) \rightarrow O(W_{size}/2)$[1].
- Similarly, PSD symmetry enables $O(N_{FFT}) \rightarrow O(N_{FFT}/2)$.
- The complexity of the Mel Bank step has been reduced from initial $O(k \cdot N_{FFT})$ to $O(round(N_{FFT}/2))$ thanks to the optimization of the energy matrix calculation described in section 3.4.1.
- The use of simplified moments (see section 3.4.2) and incremental computing (see 3.4.3) leads to an improvement from $O(b \cdot b_{size}^2)$ to $O(b \cdot b_{size})$ in the calculation of local Hu moments.

Table 1 compares the overall complexity that the baseline implementation requires to classify one window with the one that our optimized algorithm features in three different scenarios: 1) the window power and zero-crossing are above thresholds and no time-invariance is detected (full processing); 2) thresholds are exceeded, and time-invariance can be exploited as in section 3.4.5;and 3) window power or zero-crossings

[1]Even though in *big O* notation, $O(N)$ and $O(c \cdot N)$ with constant c are equivalent, we are showing constant improvements explicitly to highlight the achieved speed-ups.

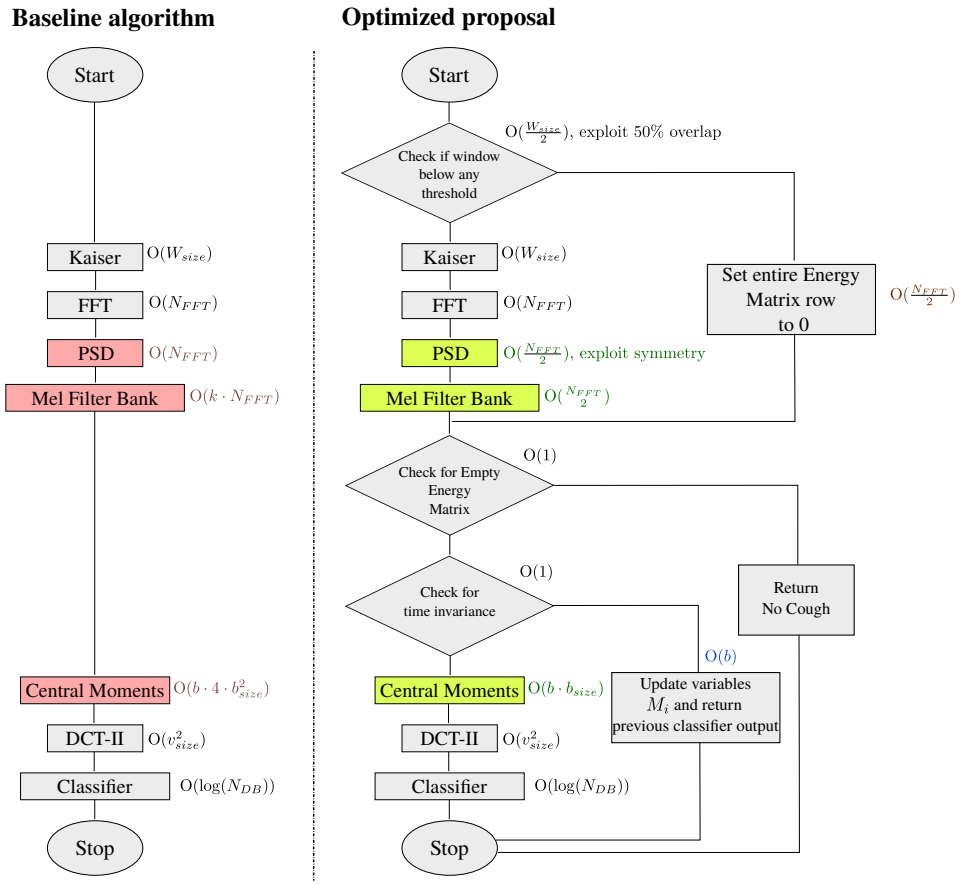

Figure 4: Flowchart for baseline and optimized algorithm where results from the complexity analysis for each stage are presented. Specific optimizations have been highlighted in colours.

are below their respective thresholds and non-cough is directly returned after detecting the empty energy matrix with no further processing (masks are applied as described in section 3.4.4).

Algorithm and scenario	**Complexity**
Baseline	$O(W_{size}) + O(k \cdot N_{FFT}) + O(b \cdot b^2_{size}) + O(v^2_{size}) + O(\log(N_{DB}))$
Optimized (full processing)	$O(W_{size}) + O(N_{FFT}) + O(b \cdot b_{size}) + O(v^2_{size}) + O(\log(N_{DB}))$
Optimized (exploiting time-invariance)	$O(W_{size}) + O(N_{FFT}) + O(b)$
Optimized (empty energy matrix)	$O(W_{size}) + O(N_{FFT})$

Table 1: Overall complexity analysis for the optimized algorithm in three different scenarios compared to baseline implementation. Optimizations are color-coded as in Figure 4.

4. Experiments

4.1. Experimental setup

We ran three tests on the data set. The first test evaluates the impact on performance of the proposed Invariant, given by Eq. (10). As metrics, we use *accuracy*, which is the ratio of correctly classified windows, plus *sensitivity* and *specificity*, which are respectively the ratio of detected coughs divided by the total number of cough windows in the testing set, and the ratio of non-cough windows correctly marked as such, divided by the total number of non-cough windows.

Each part of the protocol has been tested separately. Moments for each window are first generated in order to preserve time information, and then database is split in two groups respectively containing 60% of the data for training, and 40% for testing. The ratios of cough and non-cough windows were kept the same between groups. For validation, five repeated random splits were performed, using the same random seed for the two tested moments. For the tests, each dimension of the feature vector is normalized, and the vector is passed to a 3-NN classifier to predict the class. Afterwards, results are averaged.

For the second test, we pass the system a representative sound file of 11 seconds (440 windows), and compare the efficiency of the different optimizations in characterizing the signals. Efficiency is measured in terms of speed-up, which is the relative reduction in processing time, when compared to the base implementation. Optimizations were gradually implemented in the order they are described in this paper, so each one includes all the others.

For optimization IV (masking uninteresting windows), we set a conservative 50 dB level threshold for every window, which silences ambient sound but not human voice [36]. ZCR value is set to $90/441$, which in testing silenced most speech but not the cough events, as they are naturally turbulent regardless of their volume [37].

As optimizations IV and V (exploiting translation invariance in time) not only improve the feature vector calculation, but also allow skipping the classification step in some cases, we deemed relevant to also include performance figures for the system as a whole.

Finally, for the third test, the app was run on the phone for 24 hours, not running any other app or using it for other purposes, and then measured how much battery was depleted at the end.

The smartphone used for the tests is a Sony Xperia Z2, running Android 5.1.1. Computed times obtained in this study correspond to the average of five independent experiments.

4.2. Results

Table 2 presents average classification results in terms of sensitivity (SEN), specificity (SPE), accuracy (ACC), and positive predictive value (PPV), comparing the proposed implementation with the original moments as implemented in [17]. Metrics have been computed as follows:

$$\begin{aligned} \text{SEN} &= TP/(TP+FN) \\ \text{SPE} &= (TN)/(TN+FP) \\ \text{ACC} &= CP \cdot SEN + NCP \cdot SPE \\ \text{PPV} &= (TP)/(TP+FP) \end{aligned} \tag{19}$$

where TP (True Positives) refers to the number of cough windows predicted as such by the system, TN (True Negatives) is the number of non-cough windows that are correctly identified, and FP (False Positives) and FN (False Negatives) are the number of errors the system makes when a window is identified as a cough and non-cough respectively. CP is the percentage of cough windows in the recorded part of the protocol, and NCP is the percentage of non-cough windows. In our protocol, CP figures were 1.89% (1st part), 1.47% (2nd part) and 1.64% (3rd part).

	Original moments [17]				**Proposed moments**				**Proposed moments (postproc.)**			
Part	SEN (%)	SPE (%)	ACC (%)	PPV (%)	SEN (%)	SPE (%)	ACC (%)	PPV (%)	SEN (%)	SPE (%)	ACC (%)	PPV (%)
1st.	88.73	98.70	98.52	56.77	90.76	98.77	98.62	58.72	88.91	99.87	99.65	92.09
2nd.	86.73	98.83	98.65	52.54	89.38	98.92	98.78	55.22	84.90	99.89	99.67	92.09
3rd.	85.86	98.56	98.36	49.92	88.94	98.64	98.49	52.38	86.40	99.87	99.81	92.53

Table 2: Average classification results on patients signals. Performance figures are given in terms of Sensitivity (SEN), Specificity (SPE), Accuracy (ACC), and Positive Predictive Value (PPV).

Results show that the proposed moments, are not only simpler and faster to compute, but also improve all performance metrics in all noise environments, confirming our hypothesis that translation invariance in the frequency axis is not a desirable property for sound event recognition. Even though our system overcomes the one in [17] for all the measured figures, and despite the high SEN and SPE values, the PPV values for both compared systems are significantly low. The reason for this is that low occurence of cough events hinders the real number of FP, which turns into a high SPE value due to the high number of recorded samples. When measuring the PPV value, it turns out that there is still a high number of false positives, which is comparable to the correctly detected coughs, thus leading to values close to 50%. A high number of false positives constitutes a major problem in a system such as this. To overcome this issue, our system implementes a simple postprocessing step that ignores isolated cough-windows significantly reducing the FP values, and thus improving both SPE and PPV only at a little expense in SEN. Results for the final system including this postprocessing step are also presented in Table 2 and show the improvement in PPV, achieving values over 92%.

Table 3 shows computation times and speed-ups compared to the basic system achieved with the proposed moments. Optimizing the calculation of energy matrices by ignoring zero values from the Mel filters output achieves a 12.17× speed-up, which improves to 12.25× by removing frequency invariance. Incorporating incremental computing achieves 12.76× speed-up. This is further improved by the two subsequent optimizations namely, masking uninteresting windows using power and zero-crossing thresholds (5477.78× speed-up) and exploiting time-invariance (final speed-up: 5970.15×).

Table 4 is the counterpart of Table 3 considering the overall computation time, which includes not only feature extraction, but also classification. Similar improvements can be observed by adding the different optimization elements. However, in this case, the achieved improvement by exploiting time invariance, leads to a more significant gain (from 3084.35× to 5512.81×) compared to only feature extraction (from 5477.78× 5970.15×). The average time required to classify a window is approximately 3.1μs. Given that a new window is generated every 25ms, these results guarantee real-time

Time (ms)	Avg/window (ms)	Speed-up	Implementation
7551.86	17.1633	1×	Base
620.65	1.4106	12.17×	+Opt. Energy Matrix
616.33	1.4007	12.25×	+Simplified Moments
591.73	1.3448	12.76×	+Incremental Computing
1.38	0.0031	5477.78×	+Filtering silent
1.26	0.0028	5970.15×	+Exploit Invariance

Table 3: Computation time (ms per processed window) and speed-up (compared to baseline implementation) for the feature extraction part of the system.

performance even on low-end smartphones.

Time (ms)	Avg/window (ms)	Speed-up	Implementation
7627.37	17.33493	1×	Base
690.54	1.5694	11.05×	+Opt. Energy Matrix
688.44	1.5646	11.08×	+Simplified Moments
660.89	1.5020	11.54×	+Incremental Computing
2.47	0.0056	3084.35×	+Filtering silent
1.38	0.0031	5512.81×	+Exploit Invariance

Table 4: Computation time (ms per processed window) and speed-up (compared to baseline implementation) for the whole cough detection system.

Table 5 shows a comparison in terms of battery consumption of our proposed implementation vs. the baseline implementation of the system. The optimizations in processing time translate to 4× higher battery life.

System Implementation	Battery Consumption	Battery Efficiency
Baseline	96%	1×
Optimized	24%	4×

Table 5: Battery performance after 24 hours for proposed implementation and baseline.

Finally, a comparison with the energy consumption reported for other cough detectors in the literature (SymDetector [23] and CoughSense [3]) is presented in Table 6. Our consumption values are presented for two different functional modes. In the first one, the user GUI is only updated when a cough is detected. As for the second, the display is updated for every analyzed window. Consumption figures are reported as energy spent by each system on detecting a cough event, which corresponds to the worst case complexity scenario for our system. In addition, power consumed in average functional mode, which includes both cough and non-cough events, is presented for all the systems. A significant improvement from our proposal can be observed in both cases.

System	Average Energy Consumption per cough event (mJ)	Power Consumption in functional mode (mW)
Our system (display update only at cough detection)	250.4	95.2 (**59.0**)
Our system (display update at 40 fps)	460.4	108.8 (**67.4**)
SymDetector [23]	1250.0	399.0
CoughSense [3]	1930.0	536.1

Table 6: Average energy consumption per cough event and estimated power consumption of the proposed implementation compared to SymDetector [23] and CoughSense [3]. The power consumed by our app as reported by the Android profiler is shown in brackets (only 62% of the overall consumption was due to the cough detector).

5. Discussion

5.1. Complexity analysis and efficiency improvement

The complexity analysis summarized in Figure 4 and Table 1 shows that even in the worst case scenario, the proposed optimizations avoid complexity dependence with k, the number of filters in the Mel bank. In addition, moment calculation is simplified in terms of complexity from $O(b \cdot b_{size}^2)$ to $O(b \cdot b_{size})$. In more favourable scenarios, where either window power or zero-crossings are below thresholds or time invariance is detected, moment calculation and classification can be skipped, and complexity is significantly reduced. This is especially relevant for the k-NN classification step, since the term $O(\log(N_{DB}))$ can have a dramatic impact for large databases where the nearest neighbour is sought. This has a significant effect in the achieved speed-ups. By observing the last two rows in tables 3 and 4, the average required computation time to classify a window decreases two orders of magnitude.

5.2. Comparative analysis on energy consumption

The comparative analysis with SymDetector [23] and CoughSense [3] (see Table 6) shows that our proposal consumes $\approx$ 5 times less energy than the former in the worst case scenario (full window processing for cough detection) whereas the latter is overcome by a factor of $\approx$ 8. In terms of power consumption in normal functioning mode, where all optimizations are eventually applied, SymDetector consumes $\approx$ 6.77 more power than our optimal implementation. As for CoughSense, our system consumes $\approx$ 9 times less power during normal activity.

The *SymDetector* app also employed power thresholding to save energy. As reported in [23], the system managed to work at full functionality for 20 hours on a Samsung Galaxy S3 with a 3.8V, 2100mAh battery. Our implementation on a Sony Xperia Z2 with a 3.4V 3200mAh battery consumed only 24% of the battery life in 24 hours. This was reduced to 21% if the display was updated only when a cough was detected instead of every analyzed window at 40 frames per second. In both cases, the app was only responsible for 62% of that consumption. Our method guarantees that a phone running

the app can sustain 24h of continuous operation as requested by the medical community, and have more than enough battery left for the normal operation of the phone. Even for the baseline implementation (see Table 5), the system would be able to run for 24 hours. The optimized proposal would allow 4 times this duration. With the reported figures, SymDetector would not guarantee 24 hours of full functionality (6.77× higher consumption than our optimized version, (6.77/4)× higher than the baseline version which allows 24 hours on the Xperia Z2), nor would do CoughSense (9×).

5.3. Detection performance

Considering detection performance, the achieved SEN, SPE, ACC, and PPV values (see Table 2) overcome those achieved with the baseline implementation which was shown in [17] to outperform the most commonly used feature sets for cough detection in noisy environments. Especially, for PPV, values, our final proposal including preprocessing achieved values over 92.03% only at a little expense of SEN, which fell 1.8% below the one for the baseline only in the 2nd part of the protocol, while outperforming it for the rest of measures.

Other systems in the literature tested in noisy environments yielded poorer results. For instance, the accuracy of the method proposed in [15] was above 85% for high SNR values (>5dB). However, it significantly dropped for lower values below (78% for 0 dB, 67% for -5 dB, 57.5% for -10 dB). Sensitivity-wise, our proposal also outperforms SymDetector [23] for cough detection in similar experimental conditions (85.9%), which overcame both the Leicester Cough Monitor [8] and CoughSense [3] over the same database. The same is true for PPV values, where SymDetector only achieved 87.3%. The specificity and sensitivity values reported for CoughSense in [3] were higher ($\approx$ 90% and $\approx$ 99%, respectively). However, results for both SymDetector and CoughSense are provided on classification of audio events after a pre-filtering process in which cough events can be missed. Actually, the pre-filtering stage in SymDetector misses 4% of cough events [23], so the final reported figures should be adjusted to incorporate this. Other systems in the literature reporting higher performance figures such as [14] are also intended for pre-detected event classification. Our results are provided for online detection from the raw signal instead.

5.4. Limitations of the study and future work

The evaluation of the present study has been carried out on a limited population where 13 participants with three different respiratory conditions were recorded for approximately 1 day overall. The protocol included three parts where low, moderate and high (closer to reality) noise levels were emulated. The size of the subject group and the environment may not be deemed sufficiently representative of real environments and patient population. However, we demonstrated in our previous work [17] that the employed detection system succeeds to perform on a wide variety of real noise environments with low SNR levels. This paper focuses on how this system can be optimized to achieve detection with low battery consumption to enable continuous real-time time monitoring. Further validation of the detection system with longer recordings on a larger population is part of the future work and falls beyond the scope of this paper.

An additional limitation could be found on the dependence of the overall efficiency with the size of the database N_{DB} employed in the k-NN classifier. This becomes patent

from the dramatic improvements achieved when the classifier can be skipped. Optimization of the classifier has been addressed in our previous work [24] in order to enable real-time processing even in the worst case scenario. With the optimizations proposed in this paper, we achieve 12× speed-up, which allow real-time computation with significant battery savings even in those cases.

Our future work moves in two parallel directions. First, as stated above, we are currently working towards further validation of the detection system by running different studies involving more patients and longer acquisition protocols during real life activities. Secondly, an additional future line will explore on-line improvements of the system by enabling real-time updates of the database to achieve adaptive learning from detected samples.

6. Conclusions

We have proposed a robust and efficient cough detection system able to run on a smartphone with high sensitivity and specificity and low battery consumption. The system relies on an efficient implementation of local Hu moments that exploits signal properties to improve accuracy and energy efficiency. It is worth mentioning that some properties of the computed moments such as invariance to rotation, scaling and translation, have proven not to be advantageous for the problem at hand. The proposed implementation takes advantage of this fact and removes these invariances to more accurate detection with less calculations.

The proposed optimizations result in a system that is over five thousand times faster and consumes four times less battery than the baseline. Compared to other systems in the literature, the implemented system is 5 times more efficient in a worst case scenario and almost 7 times in an average one. The improved system is able to run in real-time during periods of four days of continuous use whereas other systems struggle to work for 24 hours without completely draining the battery.

Acknowledgments

Funding: This work was supported by the Digital Health & Care Institute Scotland as part of the Factory Research Project SmartCough/MacMasters. The authors would like to acknowledge support from University of the West of Scotland for partially funding C. Hoyos-Barceló and J. Monge-Álvarez studentships. UWS acknowledges the financial support of NHS Research Scotland (NRS) through Edinburgh Clinical Research Facility. Acknowledgement is extended to Cancer Research UK for grant C59355-A22878.

The authors would like to thank the SmartCough clinical team at the University of Edinburgh. Dr. Lucy McCloughan, Prof. Brian McKinstry, Dr. Hillary Pinnock, and Dr. Roberto Rabinovich, who provided valuable support in clinical matters. Additional thanks are given to Lorna Stevenson, Dave Bertin, and Jill Adams, from Chest Heart and Stroke Scotland, for setting up the patient panel for the SmartCough project.

Declarations of interest

None.

References

[1] S. Adibi, Mobile health: a technology road map, vol. 5, Springer, 2015.

[2] H. Pinnock, J. Hanley, L. McCloughan, A. Todd, A. Krishan, S. Lewis, A. Stoddart, M. van der Pol, W. MacNee, A. Sheikh, C. Pagliari, B. McKinstry, Effectiveness of telemonitoring integrated into existing clinical services on hospital admission for exacerbation of chronic obstructive pulmonary disease: researcher blind, multicentre, randomised controlled trial, BMJ 347.

[3] E. C. Larson, T. Lee, S. Liu, M. Rosenfeld, S. N. Patel, Accurate and privacy preserving cough sensing using a low-cost microphone, in: Proceedings of the 13th international conference on Ubiquitous computing, ACM, 375–384, 2011.

[4] J. Smith, A. Woodcock, New developments in the objective assessment of cough, Lung 186 (1) (2008) 48–54.

[5] E. R. S. (ERS), The economic burden of lung disease, European Lung White Book Ch. 2 (2015) 16–27.

[6] M. J. Fletcher, J. Upton, J. Taylor-Fishwick, S. A. Buist, C. Jenkins, J. Hutton, N. Barnes, T. Van Der Molen, J. W. Walsh, P. Jones, et al., COPD uncovered: an international survey on the impact of chronic obstructive pulmonary disease [COPD] on a working age population, BMC Public Health 11 (1) (2011) 612.

[7] M. Sterling, H. Rhee, M. Bocko, Automated cough assessment on a mobile platform, Journal of medical engineering 2014.

[8] S. S. Birring, T. Fleming, S. Matos, A. A. Raj, D. H. Evans, I. D. Pavord, The Leicester Cough Monitor: preliminary validation of an automated cough detection system in chronic cough, European Respiratory Journal 31 (5) (2008) 1013–1018, ISSN 0903-1936, doi:10.1183/09031936.00057407.

[9] E. Vizel, M. Yigla, Y. Goryachev, E. Dekel, V. Felis, H. Levi, I. Kroin, S. Godfrey, N. Gavriely, Validation of an ambulatory cough detection and counting application using voluntary cough under different conditions, Cough 6 (1) (2010) 3, doi:10.1186/1745-9974-6-3.

[10] T. Drugman, J. Urbain, N. Bauwens, R. Chessini, C. Valderrama, P. Lebecque, T. Dutoit, Objective Study of Sensor Relevance for Automatic Cough Detection, IEEE Journal of Biomedical and Health Informatics 17 (3) (2013) 699–707.

[11] Y. A. Amrulloh, U. R. Abeyratne, V. Swarnkar, R. Triasih, A. Setyati, Automatic cough segmentation from non-contact sound recordings in pediatric wards, Biomedical Signal Processing and Control 21 (Supplement C) (2015) 126 – 136.

[12] K. Tokuda, T. Kobayashi, T. Masuko, S. Imai, Mel-generalized cepstral analysis-a unified approach to speech spectral estimation, in: Third International Conference on Spoken Language Processing, 1994.

[13] J. M. Liu, M. You, G. Z. Li, Z. Wang, X. Xu, Z. Qiu, W. Xie, C. An, S. Chen, Cough signal recognition with Gammatone Cepstral Coefficients, in: 2013 IEEE China Summit and International Conference on Signal and Information Processing, 160–164, 2013.

[14] A. Murata, N. Ohota, A. Shibuya, H. Ono, S. Kudoh, New Non-invasive Automatic Cough Counting Program Based on 6 Types of Classified Cough Sounds, Internal Medicine 45 (6) (2006) 391–397.

[15] S. H. Shin, T. Hashimoto, S. Hatano, Automatic Detection System for Cough Sounds as a Symptom of Abnormal Health Condition, IEEE Transactions on Information Technology in Biomedicine 13 (4) (2009) 486–493.

[16] M. Krajnik, I. Damps-Konstanska, L. Gorska, E. Jassem, A portable automatic cough analyser in the ambulatory assessment of cough, BioMedical Engineering OnLine 9 (1) (2010) 17.

[17] J. Monge-Alvarez, C. Hoyos-Barcelo, P. Lesso, P. Casaseca-de-la Higuera, Robust Detection of Audio-Cough Events using local Hu moments, IEEE Journal of Biomedical and Health Informatics (in press).

[18] J. Monge-Álvarez, C. Hoyos-Barceló, K. Dahal, P. Casaseca-de-la Higuera, Audio-cough event detection based on moment theory, Applied Acoustics 135 (2018) 124–135.

[19] Y. Sun, G. Wen, J. Wang, Weighted spectral features based on local Hu moments for speech emotion recognition, Biomedical signal processing and control 18 (2015) 80–90.

[20] V. Swarnkar, U. R. Abeyratne, A. B. Chang, Y. A. Amrulloh, A. Setyati, R. Triasih, Automatic Identification of Wet and Dry Cough in Pediatric Patients with Respiratory Diseases, Annals of Biomedical Engineering 41 (5) (2013) 1016–1028.

[21] U. R. Abeyratne, V. Swarnkar, A. Setyati, R. Triasih, Cough Sound Analysis Can Rapidly Diagnose Childhood Pneumonia, Annals of Biomedical Engineering 41 (11) (2013) 2448–2462.

[22] K. Kosasih, U. R. Abeyratne, V. Swarnkar, R. Triasih, Wavelet Augmented Cough Analysis for Rapid Childhood Pneumonia Diagnosis, IEEE Transactions on Biomedical Engineering 62 (4) (2015) 1185–1194.

[23] X. Sun, Z. Lu, W. Hu, G. Cao, SymDetector: detecting sound-related respiratory symptoms using smartphones, in: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM, 97–108, 2015.

[24] C. Hoyos-Barceló, J. Monge-Álvarez, M. Z. Shakir, J.-M. Alcaraz-Calero, P. Casaseca-de-la Higuera, Effi-

cient *k*-NN Implementation for Real-Time Detection of Cough Events in Smartphones, IEEE Journal of Biomedical and Health Informatics (in press).
[25] P. Boersma, V. van Heuven, Speak and unSpeak with Praat, Glot International 5 (2001) 341–347.
[26] J. Flusser, T. Suk, B. Zitová, 3D Moment Invariants to translation, rotation, and scaling, 2D and 3D Image Analysis by Moments (2016) 95–162.
[27] M.-K. Hu, Visual pattern recognition by moment invariants, IRE transactions on information theory 8 (2) (1962) 179–187.
[28] J. Monge-Alvarez, C. Hoyos-Barceló, P. Lesso, J. Escudero, K. Dahal, P. Casaseca-de-la Higuera, Effect of importance sampling on robust segmentation of audio-cough events in noisy environments, in: Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the, IEEE, 3740–3744, 2016.
[29] P. Casaseca-de-la Higuera, P. Lesso, B. McKinstry, H. Pinnock, R. Rabinovich, L. McCloughan, J. Monge-Álvarez, Effect of downsampling and compressive sensing on audio-based continuous cough monitoring, in: Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE, IEEE, 6231–6235, 2015.
[30] S. Haykin, Communication Systems. New York: John Willey & Sons, 2001.
[31] B. A. Blesser, Perception of spectrally rotated speech., Ph.D. thesis, Massachusetts Institute of Technology, 1969.
[32] T. M. Hupkens, J. De Clippeleir, Noise and intensity invariant moments, Pattern Recognition Letters 16 (4) (1995) 371–376.
[33] B. Pascal, Traité du triangle arithmétique avec quelques autres petits traitez sur la mesme matière, 1665.
[34] D. Zuras, M. Cowlishaw, A. Aiken, M. Applegate, D. Bailey, S. Bass, D. Bhandarkar, M. Bhat, D. Bindel, S. Boldo, et al., IEEE standard for floating-point arithmetic, IEEE Std 754-2008 (2008) 1–70.
[35] B. N. Datta, Numerical linear algebra and applications, vol. 116, Society for Industrial and Applied Mathematic, 2010.
[36] IAC Acoustics, Comparative Examples of Noise Levels, 2018.
[37] K. Yatani, K. N. Truong, BodyScope: a wearable acoustic sensor for activity recognition, in: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, ACM, 341–350, 2012.