SURF: subject-adaptive unsupervised ECG signal compression for wearable fitness monitors

Mohsen Hooshmand[§], Davide Zordan[†], Tommaso Melodia[‡], Michele Rossi[†⋆]

***Abstract*—Recent advances in wearable devices allow non-invasive and inexpensive collection of biomedical signals including electrocardiogram (ECG), blood pressure, respiration, among others. Collection and processing of various biomarkers is expected to facilitate preventive healthcare through personalized medical applications. Since wearables are based on size- and resource-constrained hardware, and are battery operated, they need to run lightweight algorithms to efficiently manage energy and memory. To accomplish this goal, this article proposes SURF, a subject-adpative unsupervised signal compressor for wearable fitness monitors. The core idea is to perform a specialized lossy compression algorithm on the ECG signal at the source (wearable device), to decrease the energy consumption required for wireless transmission and thus prolong the battery lifetime. SURF leverages unsupervised learning techniques to build and maintain, at runtime, a *subject-adaptive* dictionary without requiring any prior information on the signal. Dictionaries are constructed within a suitable feature space, allowing the addition and removal of codewords according to the signal's dynamics (for given target fidelity and energy consumption objectives). Extensive performance evaluation results, obtained with reference ECG traces and with our own measurements from a commercial wearable wireless monitor, show the superiority of SURF against state-of-the-art techniques, including (i) compression ratios up to 90-times, (ii) reconstruction errors (RMSE) between 2% and 7% of the signal's range (depending on the amount of compression sought), and (iii) reduction in energy consumption of up to two-orders of magnitude with respect to sending the signal uncompressed, while preserving its morphology. SURF, with artifact prone ECG signals, allows for typical compression efficiencies (CE) in the range $\mathrm{CE} \in [40, 50]$, which means that the data rate of 3 kbit/s that would be required to send the uncompressed ECG trace is lowered to 60 bit/s and 75 bit/s for $\mathrm{CE} = 40$ and $\mathrm{CE} = 50$, respectively.**

I. Introduction

WEARABLES can be integrated into wireless body sensor networks (WBSN) [1] to update medical records via the Internet, thus enabling prevention, early diagnosis and personalized care. However, since they are required to be small and lightweight, they are also *resource constrained* in terms of energy, transmission capability, and memory.

In this article, we propose new data processing solutions for the long-term monitoring of quasi-periodic electrocardiography (ECG) signals. These biomedical traces are relatively easy to measure, but are at the same time extremely valuable for the aforementioned purposes. We consider the acquisition of such signals through wearable devices like smart watches or chest straps [2], [3] and are concerned with prolonging the battery time of these wearables through *lossy* signal compression. We consider scenarios where wireless transmission of ECG signals to some access point is required, so that the signal can be stored and made available through cloud servers to be analyzed by clinicians. Our approach consists of compressing the ECG time series right on the wearable device, prior to their transmission, so that the data to be stored and sent takes a small portion of its original space. As we quantify below, this leads to substantial energy savings (between one and two orders of magnitude) and to prolonged battery life.

The proposed compression algorithm is based on unsupervised neural maps for the construction of online dictionaries, whose codewords are utilized to match input patterns. The acquired biomedical signal, thanks to its quasi-periodic nature, is decomposed into *segments* made up of samples between consecutive signal peaks. We consider these segments as the signal's recurrent patterns. A preliminary training phase uses the incoming segments to learn the signal distribution of the actual subject. The synaptic weights of the neural maps become progressively and adaptively tuned to approximate such distribution, without any prior knowledge upon it. Note that these weights represent the codewords. Once the dictionary has been set up, each input segment is then encoded through a Vector Quantization (VQ) approach, which selects the best matching codeword and transmits its index to the receiving end in place of the full data. Moreover, each new data segment is also utilized to refine the dictionary in a real-time, online fashion. This is particularly appealing since it allows updating the dictionary to new subjects or to the same subject if and when their signal statistics undergoes major changes. We underline that, although our approach is here designed and tested against ECG traces, it can be applied to any quasi-periodic signal exhibiting recurrent patterns.

In a previous research paper [4], we devised a first subject-adaptive compressor that builds and maintains dictionaries *on the fly* using Time Adaptive Self Organizing Maps (TASOM), see [5]–[7]. As shown in that paper, these neural networks have excellent learning and adaptation capabilities and in general lead to high compression rates, reaching very good performance were other algorithms typically fail. In this work, we build on our previous research, and in particular on the TASOM-based algorithm of [4], by proceeding along two main axes: (i) first, we prove that TASOM-based dictionaries have some major limitations when new and sporadic patterns arise, such as in the presence of artifacts caused, for example, by motion of the wearer, (ii) given this, we design and validate a new algorithm, called SURF, for "Subject-

⋆Corresponding author. §Dept. of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI, US. †Dept. of Information Engineering, University of Padova, via Gradenigo 6/b, 35131, Padova, Italy. ‡Dept. of Electrical and Computer Engineering, Northeastern University, 02115, Boston, MA, US.

adpative Unsupervised signal compressor for weaRable Fitness monitors", which overcomes the limitations of TASOM-based schemes, while retaining their excellent performance in terms of compression ratio. This is an entirely new design that combines three different dictionaries with unsupervised learning techniques, successfully coping with artifacts.

Two possible usage models for SURF are shown in Fig. 1. The SURF compressor is run inside the wearable ECG monitor, taking as input the raw ECG sequence and generating a compressed bitstream. The latter is then sent over a wireless channel to an access point ("scenario a") or to a smartphone ("scenario b"). The SURF decompressor is used to reconstruct the original ECG signal and can either be implemented at an Internet server (see scenario a) or at the wireless receiver (e.g., at the smartphone for scenario b). The wireless channel is used to transport the compressed bitstream and to also update the dictionaries that are utilized at the decompressor to reconstruct the original ECG sequence.

SURF is based on the Growing Neural Gas (GNG) network of [8]. With GNG, the dictionary size can be dynamically adapted through the addition and removal of neurons. This allows the exploration of new regions in the data space without affecting the accuracy reached by the dictionary in the regions that have been already explored. Also, with SURF dictionaries are learned in a suitable *feature space*, with reduced dimensionality with respect to the dimensionality of the signal to be compressed. This makes it possible to further enhance the compression efficiency and reduce the cost of dictionary updates. In addition, an original dictionary management scheme is adopted, where one dictionary is used for compression, one for continuous learning and one for the assessment of previously unseen patterns. Before being used for compression, new codewords must pass the assessment phase. Patterns that are still under assessment are not encoded through the dictionary, but their compact (transform domain) representation is sent instead. This allows achieving small representation errors at all times, while effectively coping with artifacts (see Section VIII) and refining the dictionaries as the signal statistics change. Overall, SURF learns and maintains dictionaries in a totally unsupervised and online fashion, requiring less than $20\,\mathrm{kbytes}$ of memory space, while allowing for high compression ratios and small reconstruction errors (usually within 2 and 7% of the signal's peak-to-peak range, depending on the compression factor).

Several codebook-based solutions were proposed in the literature, see, e.g., the Gain-Shape Vector Quantization (GSVQ) method of [9]. GSVQ relies on *offline* learning, i.e., the codebook is attained from pre-collected datasets and is then used at runtime. With this approach, the codebook cannot be changed if the signal statistics changes significantly and a stream of *residuals* is transmitted to compensate for this. With GSVQ, the achievable reduction in size for ECG signals is up to 35-fold, as opposed to the much higher performance attained by SURF, which ranges from 50- to 96-fold (depending on the frequency of artifacts in the input data). Also, the performance of SURF is here compared against that of selected compression algorithms from the literature including neural networks [4], linear approximations [10], Fourier [11]–[13], Wavelet [14] transforms and compressive sensing (CS) [15], [16]. SURF surpasses all of them, achieving remarkable performance, especially at high compression ratios where the reconstruction error (Root Mean Square Error, RMSE) at the decompressor is kept below 7% of the signal's peak-to-peak amplitude, whereas the RMSE of other solutions becomes unacceptably high.

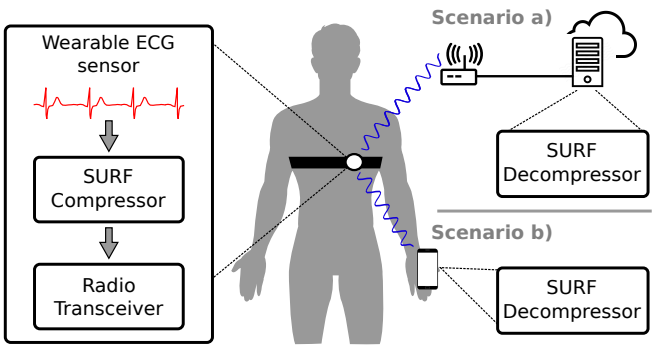


Fig. 1: Communication diagram.

A thorough numerical analysis of SURF, carried out on the PhysioNet public dataset [17] and our own collected ECG traces from a Zephyr Bioharness 3 device, reveals the following:

i) SURF's dictionaries gracefully and effectively adapt to new subjects or to their new activities,
ii) the size of these dictionaries is kept bounded within $20\,\mathrm{kbytes}$, making them amenable to implementation in wireless monitors,
iii) high compression efficiency is reached (reductions in the signal size from 50 to 96-fold),
iv) the original ECG time series are reconstructed at the receiver with high accuracy, i.e., obtaining peak-to-peak RMSEs within 7% and often smaller than 3% and,
v) compression allows saving energy at the transmitter, leading to reductions of up to two orders of magnitude at the highest compression ratios.

The reminder of this paper is structured as follows. In Section II, we discuss previous work on lossy compression for ECG signals. In Section III, we briefly review vector quantization, which we also exploit in our design. In Section IV, we introduce the self-organizing maps, and in Section V we describe an initial design based on them. This design is the same of [4] and is discussed here for the sake of completeness and for a better understanding of the more complex design of Section VI, where we describe in detail the SURF compression scheme. A thorough performance evaluation is carried out in Section VIII, comparing SURF against state-of-the art solutions for reference and own collected ECG traces. Conclusions are drawn in Section IX.

The following Section III introduces some preliminary notions on network quantization and Section IV summarizes the main operational principles of self organizing maps, on which the algorithms that are proposed in this paper rest. This material can be skipped by an expert reader on these matters.

II. Related Work

Compression algorithms for ECG signals can be grouped into three main categories: *Direct Methods*, *Transformation Methods* and *Parameter Extraction Methods*.

Direct methods, which include the *Lightweight Temporal Compression* (LTC) [10], the *Amplitude Zone Time Epoch Coding* (AZTEC) [18], and the *Coordinate Reduction Time Encoding System* (CORTES) [19], operate in the time domain and utilize prediction or interpolation algorithms to reduce redundancy in the input signal by examining subsequent time samples.

Transformation methods perform a linear orthogonal transformation. The most widely adopted techniques are *Fast Fourier Transform* (FFT) [11], *Discrete Cosine Transform* (DCT) [20], and *Discrete Wavelet Transform* (DWT) [14]. The amount of compression they achieve depends on the number of transform coefficients that are selected, whereas their representation accuracy depends on how many and which coefficients are retained. Although these algorithms can provide high compression ratios, their computational complexity is often too high for wearable devices. Also, as we quantify below, these methods are in general outperformed by linear and dictionary-based approaches at high compression ratios.

Parameter extraction methods use Artificial Neural Networks (ANNs), Vector Quantization (VQ), and pattern recognition techniques. This is a field with limited investigation that has recently aroused great interest from the research community. Unlike direct and transformation methods, the rationale is to process the input time series to obtain some kind of *knowledge* (e.g., input data probability distribution, signal features, hidden topological structures) and utilize it to get compact and accurate signal representations. The algorithm that we propose in this paper belongs to this class. Other representative algorithms are [4], [9], [21]–[24]. In [21], a direct waveform *Mean-Shape Vector Quantization* (MSVQ) is tailored for single-lead ECG compression. Based on the observation that many short-length segments mainly differing in their mean values can be found in a typical ECG signal, the authors segment the ECG into vectors, subtract from each vector its mean value, and apply scalar quantization and vector quantization to the extracted means and zero-mean vectors respectively. Differently from our approach, the segmentation procedure is carried out by fixing the vector length to a predetermined value. This avoids the computational burden of peak detection, but it does not take full advantage of the redundancy among adjacent heartbeats, which are in fact highly correlated. Moreover, in MSVQ, dictionaries are built through the *Linde-Buzo-Gray* (LBG) algorithm [25], without adapting its codewords at runtime. In [9], Sun et al. propose another vector quantization scheme for ECG compression, using the *Gain-Shape Vector Quantization* (GSVQ) approach. There, the ECG is segmented into vectors made up of samples between two consecutive signal peaks. Each extracted vector is normalized to a fixed length and divided by its *gain* to obtain the so called *shape* vector. A codebook for the shape vectors is generated using the LBG algorithm. After this, each normalized vector is assigned the index of the nearest codeword in the dictionary and a residual vector is encoded to compensate for inaccuracies. The original length of each heartbeat, the gain, the index of the nearest codeword and the encoded (residual) stream are sent to the decoder. For the signal reconstruction at the receiver, the decoder retrieves the codeword from its local copy of the dictionary, performs a denormalization using the gain and the length, and adds the residual signal. SURF resembles [9] in the way signal segments are defined and in the adoption of the GSVQ approach. Indeed, the main ECG peaks are used to extract segments, which then constitute the recurrent patterns to be learned. [22] distinguishes itself from the previous schemes because it defines a codebook of ECG vectors adaptable in *real-time*. The dictionary is implemented in a one dimensional array with overlapped and linearly shifted codewords that are continuously updated and possibly removed according to their frequency of utilization. In particular, an input vector that does not find a matching codeword is added to the codebook, triggering the removal of the codeword with the least number of matches. However, no details are provided on ECG segmentation nor on how ECG segments with different lengths are to be processed. A compression approach based on vector quantization, where dictionaries are built and maintained at runtime is presented in [4]. In this paper, time adaptive self organizing maps are utilized to reshape the dictionary as the signal statistics change. As we show in Section VIII, while this approach has excellent compression performance and gracefully adapts to non-stationary signals, it is not robust to artifacts, i.e., the quality of the dictionary degrades in the presence of sudden changes in the signal statistics or of previously unseen patterns. A compression scheme for quasi-periodic time series can be found in [24], where the authors target the lightweight compression of biomedical signals for constrained devices, as we do in this paper. They do not use a VQ approach but exploit sparse autoencoders and pattern recognition as a means to achieve *dimensionality reduction* and compactly represent the information in the original signal segments through shorter segments. Quantitative results assess the effectiveness of their approach in terms of compression ratio, reconstruction error and computational complexity. However, the scheme is based on a training phase that must be carried out *offline* and is thus not suitable for patient-centered applications featuring previously unseen signal statistics. A taxonomy describing most of these compression approaches, including their quantitative comparison, can be found in the survey paper [26].

Our present work improves upon previous research as neural network structures are utilized to build and adapt compact representations of biomedical signals at *runtime*, utilizing unsupervised learning. Our design uses multiple dictionaries to ensure robustness against artifacts, still obtaining very high compression ratios, and achieving small reconstruction errors at all times.

III. Preliminaries on Vector Quantization for Signal Compression

Vector quantization is a technique originally conceived for lossy data compression but also applicable to clustering,

pattern recognition and density estimation. VQ is a generalization of scalar quantization of a single random variable to quantization of a block (vector) of random variables [27]. Its motivation lies on the fundamental result of Shannon's rate-distortion theory, which states that better performance (i.e., lower distortion for a given rate or lower rate for a given distortion) can always be achieved by encoding vectors instead of scalars, even if the data source is memoryless or the data compression system is allowed to have memory. Let $\mathbf{x} = [x_1, x_2, ..., x_m]^T \in \mathbb{R}^m$ be an m dimensional input random vector. A **vector quantizer** is described by:

- A set of *decision regions* $I_j \subseteq \mathbb{R}^m, j = 1, \ldots, L$, such that $I_j \cap I_h = \emptyset, j, h = 1, \ldots, L$, $j \neq h$, and the union of I_j (with $j = 1, \ldots, L$) spans $\mathbb{R}^m$.
- A finite set of reproduction vectors (*codewords*) $\mathcal{C} = \{\boldsymbol{y}_j\}_{j=1}^L$, $\boldsymbol{y}_j = [y_{j1}, y_{j2}, \ldots, y_{jm}]^T \in I_j \subseteq \mathbb{R}^m$. This set is called *codebook* or *dictionary*. Each codeword $\boldsymbol{y}_j$ is assigned a unique index.
- A *quantization rule* $q(\cdot)$:

$$q(\boldsymbol{x}) = \boldsymbol{y}_j \quad \text{if} \quad \boldsymbol{x} \in I_j \,. \tag{1}$$

  This means that the j^{th} decision region I_j is associated with the j^{th} codeword $\boldsymbol{y}_j$ and that each vector $\boldsymbol{x}$ belonging to I_j is mapped by (1) into $\boldsymbol{y}_j$.

A compression system based on VQ involves an encoder and a decoder. At the encoder, the output samples from the data source (e.g., samples from a waveform, pixels from an image) are grouped into blocks (vectors) and each of them is given as input to the VQ. The VQ maps each vector $\boldsymbol{x}$ onto codeword $\boldsymbol{y}_j$ according to (1). Compression is achieved since the index j associated with $\boldsymbol{y}_j$ is transmitted to the decoder in place of the whole codeword. Because the decoder has exactly the same dictionary stored at the encoder, it can retrieve the codeword given its index through a table lookup. Note that, for correct decoding, the dictionary at the decoder shall be the same in use at the encoder at all times. We say that encoder and decoder are synchronized if this is the case and that are out-of-synchronism otherwise.

The quality of reconstruction is measured by the average distortion between the quantizer input $\boldsymbol{x}$ and the quantizer output $\boldsymbol{y}_j$. A common distortion measure between a vector $\boldsymbol{x}$ and a codeword $\boldsymbol{y}_j$ is the Euclidean distance $d(\boldsymbol{x}, \boldsymbol{y}_j) = \|\boldsymbol{x} - \boldsymbol{y}_j\|$. The average distortion is measured through the *root mean squared error* (RMSE):

$$E[d(\boldsymbol{x}, \boldsymbol{y}_j)] = \sum_{j=1}^{L} \int_{I_j} \|\boldsymbol{a} - \boldsymbol{y}_j\| f_{\boldsymbol{x}}(\boldsymbol{a}) d\boldsymbol{a} \,, \tag{2}$$

where $f_{\boldsymbol{x}}(\cdot)$ is the *probability density function* of the random vector $\boldsymbol{x}$. The design of an optimal VQ consists in finding the dictionary $\mathcal{C}$ and the partition of $\mathbb{R}^m$ that minimize the average distortion. It can be proved that an *optimal* VQ must satisfy the following conditions:

1) **Nearest Neighbor Condition (NNC).** Given the set of codewords $\mathcal{C}$, the optimal partition of $\mathbb{R}^m$ is the one returning the minimum distortion:

$$I_j = \{\boldsymbol{x} : d(\boldsymbol{x}, \boldsymbol{y}_j) \leq d(\boldsymbol{x}, \boldsymbol{y}_h), j \neq h\} \,. \tag{3}$$

   This condition implies that the quantization rule (1) can be equivalently defined as $q(\boldsymbol{x}) = \text{argmin}_{\boldsymbol{y}_j} d(\boldsymbol{x}, \boldsymbol{y}_j)$, i.e., the selected $\boldsymbol{y}_j$ is the *nearest codeword* to the input vector $\boldsymbol{x}$.
2) **Centroid Condition (CC).** Given the partition I_j, $j = 1, \ldots, L$, the codewords are the centroids of the decision regions.

Linde, Buzo and Gray, inspired by the *k-means method* for data clustering, provided an iterative algorithm (the LBG algorithm) to generate a vector quantizer that satisfies the above conditions [25]. It essentially defines an initial dictionary and proceeds by repeatedly computing the decision regions (according to NNC) and improving the codewords (according to CC) until the average distortion falls below a given threshold. It can be formulated to address known or unknown source statistics. In the last case, a large set of input vectors, called *training set*, must be used to build up the quantizer. In this paper, we adopt a VQ approach. Input vectors for the quantizer are determined by subdividing the original signal into segments between successive peaks. Dictionaries are obtained exploiting artificial neural networks and our primary interest is to obtain and update them in an *online* fashion as the signal statistics change. Note that the LBG algorithm does not natively support this, since it is conceived for time-invariant dictionaries.

Since our reference scenario is a wearable-based healthcare application, the proposed compression framework aims at being as energy-efficient as possible. A problem that arises with VQ is related to the search of the nearest codeword during the quantization process, i.e., the codeword $\boldsymbol{y}_{j^*} \in \mathcal{C}$ such that $\boldsymbol{y}_{j^*} = \text{argmin}_{\boldsymbol{y}_j} d(\boldsymbol{x}, \boldsymbol{y}_j)$. In fact, the number of operations performed in such phase affects the overall computational complexity performance and, in turn, power consumption. To speed up the search and thus save energy, we exploit the fast dictionary search algorithm devised by Wu and Lin [28]. The idea is to bypass those codewords which satisfy a kick-out condition without the actual computation of the distortion for the bypassed codewords. This is achieved by decomposing the Euclidean distance and using the Cauchy-Schwarz inequality, see [28].

IV. Unsupervised Dictionary Learning through Self-Organizing Maps

The Self Organizing Map (SOM) and its time-adaptive version (TASOM) are single layer feed-forward networks having an input layer of source nodes that projects directly onto an output layer of neurons. The SOM provides a structured representation of the input data distribution with the synaptic-weight vectors acting as prototypes. For its output layer, we consider a square lattice $\mathcal{A}$ with L neurons arranged in M rows and M columns. The input space $\mathcal{X}$ is m-dimensional with input vectors $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T \in \mathcal{X} \subset \mathbb{R}^m$. The SOM input layer has m source nodes, each associated with a single component of the input vector $\boldsymbol{x}$ and each neuron in the lattice is connected to all the source nodes. The links (synapses) between the source nodes and the neurons are weighted, such that the j-th

neuron is associated with a *synaptic-weight vector* denoted by $\boldsymbol{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jm}]^T \in \mathbb{R}^m$, $j = 1, \ldots, L$, where $L = M^2$ is the total number of neurons in $\mathcal{A}$. Training is unsupervised. Let $\{\boldsymbol{x}(n)\}_{n=0}^{N}$ be the training set of *unlabeled* examples (training input patterns), selected at random from $\mathcal{X}$. The learning process proceeds iteratively, from $n = 0$ to $n = N$, where N should be large enough so that self organization develops properly. At iteration n, the n-th training input pattern $\boldsymbol{x}(n)$ is presented to the SOM and the following three steps are performed [6].

Step 1: competition. The neurons compete among themselves to be selected as the winning neuron, i.e., the one whose synaptic-weight vector most closely matches $\boldsymbol{x}(n)$ according to the Euclidean distance. Its index $i(\boldsymbol{x})$ satisfies:

$$i(\boldsymbol{x}) = \text{argmin}_j \, \|\boldsymbol{x}(n) - \boldsymbol{w}_j(n)\| \, , \, j = 1, \ldots, L \, . \quad (4)$$

Step 2: cooperation. The winning neuron $i(\boldsymbol{x})$ identifies the center of a topological neighborhood of cooperating neurons modeled by the function $h_{ij}(n)$. If d_{ij} is the lateral distance between $i(\boldsymbol{x})$ and neuron j in $\mathcal{A}$, then $h_{ij}(n)$ is symmetric around $i(\boldsymbol{x})$ and its amplitude decreases monotonically with increasing lateral distance d_{ij}. Moreover, $h_{ij}(n)$ shrinks over time. We set $h_{ij}(n) = \exp(-d_{ij}^2/(2\sigma(n)^2))$ (unnormalized Gaussian function), where $\sigma(n)$ is the width of the topological neighborhood, that exponentially decreases with increasing time n [6].

Step 3: synaptic Adaptation. The synaptic-weight vector $\boldsymbol{w}_j(n)$ of neuron j at time n is adjusted through the equation:

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \eta(n) h_{ij}(n)(\boldsymbol{x}(n) - \boldsymbol{w}_j(n)) \, . \quad (5)$$

Equation (5) has the effect of moving the synaptic-weight vector $\boldsymbol{w}_{i(\boldsymbol{x})}$ of the winning neuron $i(\boldsymbol{x})$ (and the synaptic-weight vectors of the neurons in its topological neighborhood, through $h_{ij}(n)$) toward the input vector $\boldsymbol{x}$. The learning-rate parameter $\eta(n)$ starts at some initial value $\eta(0)$ and then exponentially decreases with increasing n through $\eta(n) = \eta_0 \exp(-n/\tau_2)$, $n = 0, 1, \ldots$, where τ_2 is a time constant. The synaptic weight adaptation process proceeds according to (5) and can be decomposed into two phases: an *ordering phase*, during which the topological ordering of the weight vectors takes place, followed by a *convergence phase*, which fine-tunes the feature map and therefore provide an accurate statistical quantification of the input space. As a general rule, the total number of iterations allowing the map to develop properly should be at least $N = 1000 + 500 \times L$ [6].

Once the SOM algorithm has terminated, a nonlinear transformation (*feature map*) $\Phi : \mathcal{X} \rightarrow \mathcal{A}$ is obtained as $\Phi(\boldsymbol{x}) = \boldsymbol{w}_{i(\boldsymbol{x})}$, where the index $i(\boldsymbol{x})$ is found using (4). $\Phi(\cdot)$ is a quantization rule as it approximates the input data space $\mathcal{X}$ with the finite set of weights (prototypes) $\boldsymbol{w}_j \in \mathcal{A}$. In fact, the same weight vector $\boldsymbol{w}_j$ is returned in response to all the input vectors $\boldsymbol{x}$ for which $\Phi(\boldsymbol{x}) = \boldsymbol{w}_j$. Thus, the SOM algorithm is a VQ algorithm. However, upon completion of the learning phase, the SOM map stabilizes and further learning / adaptation to new input distributions is difficult. With non-stationary signals, adaptive learning must be employed to update the feature map. The Time-Adaptive Self-Organizing Map (TASOM) achieves this by allowing the map to increase the learning rate when the signal's statistics changes and for this reason is a more appealing technique with non-stationary signals. The TASOM has been introduced in [7] improving upon the basic SOM and preserving its properties in stationary and non-stationary settings. In a TASOM, each neuron j, $j = 1, \ldots, L$, has a synaptic-weight vector $\boldsymbol{w}_j \in \mathbb{R}^m$ with its own learning-rate $\eta_j(n)$ and neighborhood width $\sigma_j(n)$, which are continuously adapted so as to allow a potentially unlimited training of the synaptic-weight vectors. The reader is referred to [7] for additional details.

V. A FIRST DESIGN: TASOM-BASED ECG COMPRESSION

In this section, we describe an initial design that uses the TASOM unsupervised learning algorithm. First, we identify as *ECG segments* the sequence of samples between consecutive ECG peaks and we use them to build a *dictionary* that stores typical segments and is maintained consistent and well representative through online updates. A diagram of the proposed technique is shown in Fig. 2. The ECG signal is first preprocessed through a third-order Butterworth filter to remove artifacts. Hence, the fast peak detection algorithm of [29] is employed to locate the signal peaks. Since the segments may have different lengths, after their extraction, a linear interpolation block resizes each segment from its actual length $r_{\boldsymbol{x}}(n)$ to a fixed length m. The resized segment is termed $\boldsymbol{x}(n) = [x_1(n), \ldots, x_m(n)]^T$, whereas

$$e_{\boldsymbol{x}}(n) = \frac{\sum_{k=1}^{m} x_k(n)}{m} \quad (6)$$

is its offset and

$$g_{\boldsymbol{x}}(n) = \left(\sum_{k=1}^{m} x_k(n)^2 / m \right)^{1/2} \quad (7)$$

is its gain. The normalization module applies the following transformation to each entry of $\boldsymbol{x}(n)$:

$$x_k(n) \leftarrow \frac{x_k(n) - e_{\boldsymbol{x}}(n)}{g_{\boldsymbol{x}(n)}} \, , \, k = 1, \ldots, m \, . \quad (8)$$

The normalized segment feeds the dictionary manager, which uses it to update the dictionary, and the pattern matching module, which returns the best matching codeword from the dictionary and outputs its index. The segment's original length, offset, gain and codeword index are then sent to the receiver in place of the original samples.

The dictionary manager is the key block of the TASOM-based compressor. We designed it thinking of a communication scenario entailing a transmitting wearable device and a receiver, such as a smartphone. At any time instant n, two dictionaries are maintained at the transmitter: the *current dictionary* $\mathcal{C}^c(n)$, which is used to compress the input signal, and the *updated dictionary* $\mathcal{C}^u(n)$, which undergoes updating at each time instant through the TASOM algorithm and is maintained to track statistical changes in the input signal's distribution. As for the dictionaries, we consider a TASOM with L neurons. When the compression

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2749758, IEEE Access

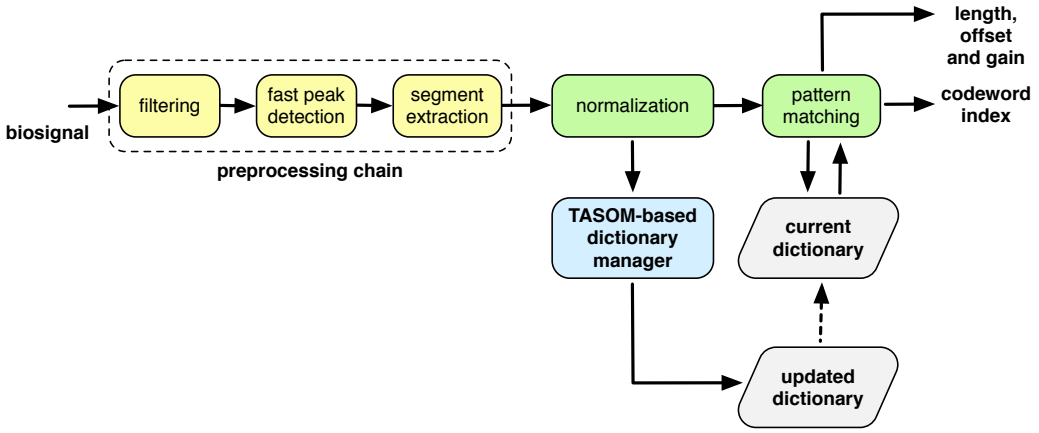


Fig. 2: Diagram of the TASOM-based compression algorithm.

scheme is used for the first time, a sufficient number N of signal segments shall be provided as input to the TASOM to perform a *preliminary training phase*. This training allows the map to learn the subject signal's distribution. This may be accomplished the first time the subject wears the device. After this, a first *subject-specific* dictionary is available. It can be used for compression and can also be updated at runtime as more data is acquired. Let us assume that time is reset when the preliminary training ends, and assume $n = 0$ at such point. The current and updated dictionaries are $\mathcal{C}^c(0) = \{\boldsymbol{c}_1^c(0), \ldots, \boldsymbol{c}_L^c(0)\}$ and $\mathcal{C}^u(0) = \{\boldsymbol{c}_1^u(0), \ldots, \boldsymbol{c}_L^u(0)\}$, respectively. Their codewords $\boldsymbol{c}_j^{c/u}(0)$ represent the synaptic-weight vectors of the corresponding neural (TASOM) maps. At time $n = 0$, we have $\boldsymbol{c}_j^c(0) = \boldsymbol{c}_j^u(0) = \boldsymbol{w}_j(0)$, $j = 1, \ldots, L$. Let also assume that the decompressor at the receiver is synchronized with the compressor, i.e., it owns a copy of $\mathcal{C}^c(0)$. From time 0 onwards, for any new segment $\boldsymbol{x}(n)$ $(n = 1, 2, \ldots)$ the following procedure is followed:

Algorithm 1 [TASOM-based compressor]:
1) Map $\boldsymbol{x}(n)$ onto the index of the best matching codeword in $\mathcal{C}^c(n)$, i.e., map $\boldsymbol{x}(n)$ onto the index $i_{\boldsymbol{x}}(n)$ such that

$$i_{\boldsymbol{x}}(n) = \text{argmin}_j \left\| \boldsymbol{x}(n) - \boldsymbol{c}_j^c(n) \right\|, \; j = 1, \ldots, L\,. \quad (9)$$

2) Let $d(n) = \|\boldsymbol{x}(n) - \boldsymbol{c}_i^c(n)\|$ be the distance between the current segment and the associated codeword, where we use index i as a shorthand notation for $i_{\boldsymbol{x}}(n)$. Use $\boldsymbol{x}(n)$ as the new input for the current iteration of the TASOM learning algorithm and obtain the new synaptic-weight vectors $\boldsymbol{w}_j(n)$, $j = 1, \ldots, L$.
3) Update $\mathcal{C}^u(n)$ by using the weights obtained in step 2, i.e., setting $\boldsymbol{c}_j^u(n) \leftarrow \boldsymbol{w}_j(n)$ for $j = 1, \ldots, L$.
4) Let $\varepsilon > 0$ be a tuning parameter. If $d(n)/\|\boldsymbol{x}(n)\| > \varepsilon$, then update $\mathcal{C}^c(n)$ by replacing it with $\mathcal{C}^u(n)$, i.e., $\mathcal{C}^c(n) \leftarrow \mathcal{C}^u(n)$ and, using (9), re-map $\boldsymbol{x}(n)$ onto the index $i_{\boldsymbol{x}}(n)$ of the best matching codeword in the new dictionary $\mathcal{C}^c(n)$.
5) Send to the receiver the segment's original length $r_{\boldsymbol{x}}(n)$, its offset $e_{\boldsymbol{x}}(n)$, gain $g_{\boldsymbol{x}}(n)$, and the codeword index $i_{\boldsymbol{x}}(n)$. If $\mathcal{C}^c(n)$ has been modified in step 4, then also send $\mathcal{C}^u(n)$ (that in this case is equal to the new $\mathcal{C}^c(n)$).

Step 2 makes it possible to always maintain an updated approximation of the input segment distribution at the transmitter. With step 4, we check the validity of the approximation provided by the current dictionary (the one used for compression, which is also known at the receiver). The tunable parameter ε is used to control the signal reconstruction fidelity at the decompressor: if $d(n)/\|\boldsymbol{x}(n)\| \leq \varepsilon$, codeword $\boldsymbol{c}_{i_{\boldsymbol{x}}(n)}^c(n)$ is considered suitable to approximate the current segment, otherwise $\mathcal{C}^c(n)$ is replaced with the updated dictionary $\mathcal{C}^u(n)$ and the encoding mapping is re-executed. Note that the higher ε, the higher the error tolerance and the lower the number of updates of the current dictionary. On the contrary, a small ε entails frequent dictionary updates: this regulates the actual representation error and also determines the maximum achievable compression efficiency.

At the receiver, the n-th ECG segment is reconstructed by picking the codeword with index $i_{\boldsymbol{x}}(n)$ from the local dictionary, performing renormalization of such codeword with respect to offset $e_{\boldsymbol{x}}(n)$ and gain $g_{\boldsymbol{x}}(n)$ and stretching the codeword according to the actual segment length $r_{\boldsymbol{x}}(n)$.

VI. The SURF compression scheme

In what follows, the TASOM-based compressor of the previous section is improved through the use of a more flexible neural network architecture, aiming at the following objectives.

O1) Objective 1 - "specializing the dictionary to new signal areas": we recall that the number of neurons in the TASOM map remains fixed as time evolves and this entails that some further refinement of the dictionary, whenever the signal statistics undergoes major changes and new behaviors arise, may not always be possible. In fact, from our experiments we have seen that, at times, additional neurons may be beneficial to specialize the dictionary upon the occurrence of new patterns, while at the same time preserving what previously learned. In that case, we do not want previous neurons to be involved in the refinement as we are exploring a new area in the input signal space, and we do not want to do this at the cost of getting lower accuracies in the portion of space that we have already inspected and successfully approximated. In our new design, we accomplish this through a GNG

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2749758, IEEE Access

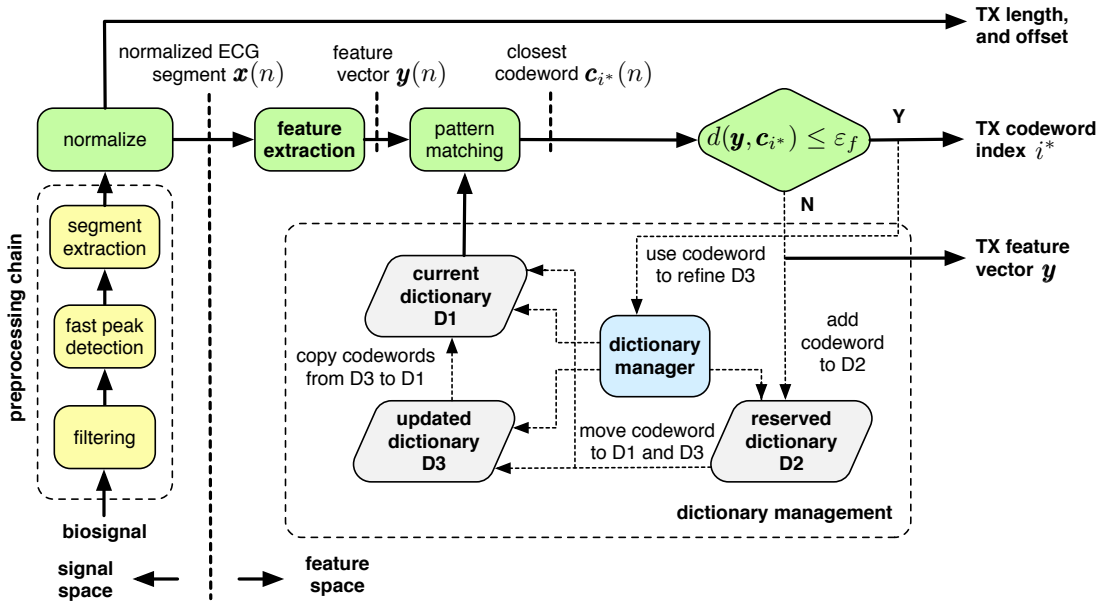


Fig. 3: Flow diagram of the SURF compression algorithm: the dictionaries are learned in the feature space. Codewords can be added or removed. When the distance between the best matching codeword in D1 and the current feature vector is higher than a threshold, a new codeword is added to D2. That codeword is in an *assessment* phase until it is either permanently added (to D1 and D3) or deleted (i.e., when no further matches occur). Dictionary D1 is used for (dictionary-based) compression, D3 for continuous learning. When a good match is found for an ECG segment, the compressor sends its length, offset and the index of the matching codeword. Otherwise, the segment's feature vector is sent along with its length and offset.

network [8]. This type of neural network incrementally learns the topological relations in a given input signal set, dynamically adding and removing neurons and connections, adapting itself to previously unseen input patterns.

O2) Objective 2 - "reducing overhead": we aim at further reducing the overhead associated with maintaining and transmitting the dictionary. This is achieved through two techniques: 1) working within a suitable *feature space*, where a number of features much smaller than the size of each ECG segment suffices for its accurate representation; 2) selective dictionary update. In the TASOM-based approach, a dictionary is *entirely* replaced whenever any of its codewords is no longer capable of approximating ECG segments belonging to a given signal's area within a preset accuracy. Instead, in our new design codewords are selectively replaced by new ones that better approximate the portion of signal space that they are responsible for.

O3) Objective 3 - "coping with artifacts": ECG signals gathered from wearable devices are prone to artifacts, caused, for example, by the body movements of the wearer. Dictionary-based approaches are particularly sensitive to artifacts as no existing codeword can adequately approximate them. Dictionary updates, attempting to bring the codewords closer to the new segments (i.e., the artifacts) are likely to result in a degraded representation accuracy for the recurring segments. However, these noisy segments must be accurately represented, as these may indicate anomalous behavior that could play an important role in the diagnosis of a disorder. Our new compressor successfully copes with this by: 1) sending features in place of full codewords whenever none of the current codewords provide a satisfactory match and 2) concurrently starting an *assessment* phase for the new pattern. In the *assessment* phase, a new neuron (codeword) is temporarily added to a local dictionary, which is only maintained at the source and is used for the evaluation of new (or anomalous) patterns. The permanent addition of such codeword to the main dictionary only occurs if further segments are found to match it, which means that the new segment has become recurrent.

Objectives O1, O2, and O3 are achieved through the SURF compression algorithm that we describe in detail next. It leverages a GNG neural structure to learn and maintain a set of prototypes in the signal's feature space in a totally unsupervised fashion. This neural network structure has a number $L(n)$ of neurons, where n is the (discrete) time index, which is updated as $n \leftarrow n+1$ each time a new ECG segment is processed.

A diagram of the SURF algorithm is shown in Fig. 3. The signal is at first preprocessed through the same chain of Fig. 2, involving filtering to remove artifacts, ECG peak detection and segment extraction. After this, ECG segments are normalized, resized and their offset is removed. As different ECG segments may have different lengths, linear interpolation is used to resize them to a fixed length m. Let $\boldsymbol{x}(n) = [x_1(n), \ldots, x_m(n)]^T$ be the resized m-length ECG segment at time n. Offset removal is achieved through:

$$x_k(n) \leftarrow x_k(n) - e_{\boldsymbol{x}}(n)\,,\; k = 1, \ldots, m\,, \qquad (10)$$

where $e_{\boldsymbol{x}}(n)$ is defined in (6). After this, the normalized

ECG segment $\boldsymbol{x}(n)$ is fed to a *feature extraction* block which reduces the dimensionality of $\boldsymbol{x}(n)$ through the computation of a number $f < m$ of features. This mapping is denoted by $\Psi : \mathbb{R}^m \rightarrow \mathbb{R}^f$ and we have: $\boldsymbol{y}(n) = \Psi(\boldsymbol{x}(n))$, where $\boldsymbol{y}(n) = [y_1(n), \ldots, y_f(n)]^T$. For our experimental results, this mapping corresponds to the DCT transform of $\boldsymbol{x}(n)$, by retaining the first (low-pass filtering) f coefficients in the transform (frequency) domain. We underline that our method is rather general and other transformation and coefficient selection methods can be applied.

At this point, the SURF dictionaries come into play. Differently from the TASOM approach, three dictionaries are maintained at the transmitter: D1) the *current dictionary* $\mathcal{C}^c(n) = \{c_1^c(n), \ldots, c_{L(n)}^c(n)\}$, D2) the *reserved dictionary* $\mathcal{C}^r(n) = \{c_1^r(n), \ldots, c_{R(n)}^r(n)\}$ and D3) the *updated dictionary* $\mathcal{C}^u(n) = \{c_1^u(n), \ldots, c_{L(n)}^u(n)\}$. D1 and D3 contain the same number of codewords at all times, whereas D2 contains $R(n)$ codewords, where in general $R(n) \ll L(n)$. D1 is used for compression at the source (transmitter) and has to be known by the decompressor at the receiver. This implies that any changes to D1 should be promptly communicated to the decompressor so that the dictionaries at source and at the receiver remain synchronized at all times. Instead, D2 and D3 only need to be maintained at the source (transmitter).

Dictionary D1: the current dictionary D1 contains the codewords which are currently in use. For each new feature segment $\boldsymbol{y}(n)$, the closest codeword $c_{i^*}^c(n)$ in D1 is fetched ("pattern matching" in Fig. 3) by minimizing the distance $d(\boldsymbol{y}(n), \boldsymbol{c}_j^c(n)) = \|\boldsymbol{y}(n) - \boldsymbol{c}_j^c(n)\|$ for all codewords $c_j^c(n) \in \mathcal{C}^c(n)$, i.e.,

$$i^* = \text{argmin}_j \, d(\boldsymbol{y}(n), \boldsymbol{c}_j^c(n)) \, , \, j = 1, \ldots, L(n) \, . \qquad (11)$$

If $d(\boldsymbol{y}(n), \boldsymbol{c}_{i^*}^c)$ is smaller than a preset error tolerance $\varepsilon_f > 0$,[1] the codeword $\boldsymbol{c}_{i^*}^c$ from D1 is deemed a good candidate to approximate the current ECG segment. In this case, we say that $\boldsymbol{y}(n)$ is *matched* by $\boldsymbol{c}_{i^*}^c$. Index i^* is thus sent to the receiver in place of the entire feature set $\boldsymbol{y}(n)$. At the receiver side, a copy of D1 is maintained at all times and is used to retrieve $\boldsymbol{c}_{i^*}^c$ from its index.

Dictionary D2: if $d(\boldsymbol{y}(n), \boldsymbol{c}_{i^*}^c) > \varepsilon_f$, none of the codewords in D1 adequately approximates the current feature vector, which is then termed *unmatched*. Note that this may be a consequence of changes in the signal statistics such as sudden variations in the subject's activity, to pathological (and often sporadic) ECG segments or to measurement artifacts. In these cases, we check for a match in the reserved dictionary D2 ($\mathcal{C}^r(n)$). If a match occurs, the *matching count* of the matching codeword in D2 is increased by one. Otherwise, a new codeword is added to D2. This is achieved by adding a neuron to dictionary $\mathcal{C}^r(n)$ and using feature vector $\boldsymbol{y}(n)$ to initialize its synaptic-weight vector. We stress that the codewords in D2 are not yet ready for use in the signal compression, but they have to go through an *assessment* phase. D2 behaves as a buffer with maximum size $L_{\max}$: if a codeword in D2 is matched γ times (with γ being a preset parameter), it is removed from D2 and added to D1. If instead D2 gets full and a new codeword has to be added to it for assessment, the oldest codeword from D2 is deleted and the new one takes its place. The rationale behind the *assessment* phase is that new codewords are added to explore a new portion of the signal's feature space, and this exploration is prompted by the measurement of previously unseen patterns. Now, if these patterns are very unlikely to occur again it does not make any sense to add them to dictionary D1 and it is better to send the feature vector $\boldsymbol{y}(n)$ for these isolated instances. In turn, $\boldsymbol{y}(n)$ will be utilized to reconstruct the pattern at the receiver. Instead, if after their first appearance, these become recurring patterns, it does make sense to add them to D1 (and D3 for their continuous refinement). Note that the combined use of D1 and D2 makes it possible to specialize the dictionary to new signal areas (new patterns, i.e., objective O1) and as well to cope with artifacts (objective O3).

Dictionary D3: this dictionary has the same number of neurons of D1 but its codewords are updated for each new *matched* ECG segment. That is, when $d(\boldsymbol{y}(n), \boldsymbol{c}_{i^*}^c) < \varepsilon_f$ the feature vector $\boldsymbol{y}(n)$ is also used to update dictionary $\mathcal{C}^u(n)$.

As stated above, dictionary D2 and D3 are continuously updated: D3 when a match occurs between $\boldsymbol{y}(n)$ and a codeword in D1, whereas D2 when no codeword in D1 matches $\boldsymbol{y}(n)$. In this case, if $\boldsymbol{y}(n)$ matches some codeword in D2, the corresponding matching count is increased, otherwise D2 is extended through the addition of a new codeword. Dictionaries D1 and D3 are initialized with $L(0)$ neurons, where $L(n)$ is always bounded, i.e., $L(n) \leq L_{\max}$ at all times, where $L_{\max}$ is a preset parameter to cope with memory constraints. At time 0, D2 is empty and the number of neurons therein is likewise bounded by $L_{\max}$. Similarly to the TASOM-based approach, when the compression scheme is activated for the first time, a sufficient number N of signal segments must be provided as input to perform a *preliminary training phase*. Such training allows the dictionaries to learn the subject signal's distribution. An observation is in oder. Basically, the just described approach dynamically switches the compression strategy between a dictionary-based technique and a standard transform-based one (i.e., sending a number of DCT coefficients for the current segment). The dictionary is used when it approximates well the current ECG pattern. Otherwise, a DCT compression approach is exploited. Note that this makes is possible to achieve high accuracy at all times, while adaptively (and automatically) tuning the instantaneous compression rate (as a function of the characteristics of the current segment). Also, this allows refining the main dictionary by only including those patterns that have become recurrent. As we shall see, this provides excellent accuracy performance, resilience against artifacts, while retaining most of the benefits of dictionary-based schemes (very high compression rates).

For the formal description of the SURF algorithm, let us assume that time is reset when the preliminary training

[1]Here, ε_f represents the error tolerance in the feature space, which must not be confused with that in the signal space ε, that was used for the TASOM-based compressor of Section V.

ends and assume $n = 0$ at such point. The codewords of D1 and D3, at time $n = 0$, $\mathcal{C}^c(0) = \{\boldsymbol{c}_1^c(0), \ldots, \boldsymbol{c}_{L(0)}^c(0)\}$ and $\mathcal{C}^u(0) = \{\boldsymbol{c}_1^u(0), \ldots, \boldsymbol{c}_{L(0)}^u(0)\}$ are set equal to the synaptic-weight vectors at the end of the initial training, i.e., $\boldsymbol{c}_j^c(0) = \boldsymbol{c}_j^u(0) = \boldsymbol{w}_j(0)$, $j = 1, \ldots, L(0)$. We also assume that the decompressor at the receiver is synchronized with the compressor, that is, it owns a copy of D1 ($\mathcal{C}^c(0)$). Also, for any codeword $\boldsymbol{c}$ belonging to any dictionary, if $d(\boldsymbol{y}(n), \boldsymbol{c}) < \varepsilon_f$ we say that $\boldsymbol{y}(n)$ is *matched* by $\boldsymbol{c}$. For the continuous update of the synaptic weight vectors (codewords) in dictionary D3, we apply the following **Algorithm 2**, which rests on the Hebbian learning theory in [30], [31].

Algorithm 2 [Synaptic weight vector update]:
At the generic time n, let $\boldsymbol{y}(n)$ and i^* respectively be the current feature vector and the index associated with the best matching codeword in D1, i.e.,

$$d(\boldsymbol{y}(n), \boldsymbol{c}_{i^*}^u(n)) \leq d(\boldsymbol{y}(n), \boldsymbol{c}_j^u(n)) \,,\; j = 1, \ldots, L(n) \,. \quad (12)$$

We have that i^* is the *winning* neuron in map (dictionary) D1 for this input (feature) vector $\boldsymbol{y}(n)$ and its synaptic weight vector is $\boldsymbol{w}_{i^*} = \boldsymbol{c}_{i^*}^u(n)$, with $\boldsymbol{w}_{i^*} \in \mathbb{R}^f$. The update rule for $\boldsymbol{w}_{i^*}$ is:

$$\boldsymbol{w}_{i^*}^{\text{new}} \leftarrow \boldsymbol{w}_{i^*} + \epsilon_b(\boldsymbol{y}(n) - \boldsymbol{w}_{i^*}) \,. \quad (13)$$

Moreover, when we have a match, an *edge* will be created in the neural map between i^* and i^{**}, where i^{**} is the second-closest neuron to the current input vector $\boldsymbol{y}(n)$. If i^* and i^{**} are already connected with an edge, no new edge will be added. After that, we update the synaptic weight vector of every neuron j that is a neighbor of i^*, i.e., that is connected to it with an edge:

$$\boldsymbol{w}_j^{\text{new}} \leftarrow \boldsymbol{w}_j + \epsilon_n(\boldsymbol{y}(n) - \boldsymbol{w}_j) \,, \quad (14)$$

where ϵ_b and ϵ_n are constant learning rates. The new weight vectors of (13) and (14) correspond to the updated codewords for dictionary D3.

Keeping the above definitions and update rules into account, from time 0 onwards, for any new feature segment $\boldsymbol{y}(n)$ ($n = 1, 2, \ldots$) the following procedure is executed:

Algorithm 3 [SURF]:
Step 1) for $\boldsymbol{y}(n)$, find the indices of the two closest codewords in D1 ($\mathcal{C}^c(n)$), which are respectively called $i_{\boldsymbol{y}}^*(n)$ and $i_{\boldsymbol{y}}^{**}(n)$, where

$$i_{\boldsymbol{y}}^*(n) = \text{argmin}_j \, d(\boldsymbol{y}(n), \boldsymbol{c}_j^c(n)) \,,\; j = 1, \ldots, L(n) \quad (15)$$

and $i_{\boldsymbol{y}}^{**}(n)$ is the index of the second-closest codeword in D1.

Step 2) Let $d(n) = d(\boldsymbol{y}(n), \boldsymbol{c}_{i^*}^c(n))$ be the distance between $\boldsymbol{y}(n)$ and the closest codeword $\boldsymbol{c}_{i^*}^c(n)$, where we use i^* as a shorthand notation for $i_{\boldsymbol{y}}^*(n)$. If $d(n) \leq \varepsilon_f$ move to **Step 3**, otherwise act as follows. Check the reserved dictionary D2 to see whether any of its codewords matches $\boldsymbol{y}(n)$. If this is the case, then increase by one unit the matching count for that codeword: if this count reaches γ, this codeword is removed from D2, added to D1 and D3 (increasing their size, i.e., $L(n) \leftarrow L(n) + 1$) and transmitted to the receiver. If no matching codeword exists in D2, the feature vector $\boldsymbol{y}(n)$ is sent to the receiver along with the length and offset of the corresponding signal segment. Also, a new codeword (neuron) is added to the reserved dictionary D2 and this neuron has a weight vector $\boldsymbol{w} = \boldsymbol{y}(n)$. Go to **Step 4**.

Step 3) Here, $d(n) \leq \varepsilon_f$. **3.1)** Use the weight vector of neuron i^* as the approximating vector for $\boldsymbol{y}(n)$. Hence, send index i^* to the receiver along with the length and offset of the signal segment associated with $\boldsymbol{y}(n)$. **3.2)** Use $\boldsymbol{y}(n)$ to update D3 through **Algorithm 2** above. **3.3)** For dictionary D3 do the following. Increase the age a_j of all the neighbors j of neuron i^*. Remove any edge with age $a_j \geq a_{\max}$, with $a_{\max}$ being a preset parameter. If this makes it so a neuron remains with no neighbors (no edges connecting it to other neurons in D3), then remove this neuron from both D1 and D3 and decrease their size, $L(n) \leftarrow L(n) - 1$. **3.4)** For dictionary D1 do the following. The distance between the input $\boldsymbol{y}$ and the nearest neuron i^* will be added to the local accumulated error of neuron i^*:

$$\text{error}(i^*)^{\text{new}} \leftarrow \text{error}(i^*)^{\text{old}} + d(\boldsymbol{y}(n), \boldsymbol{c}_{i^*}^c(n)) \,. \quad (16)$$

Step 4) Dictionary management. The following dictionary update procedure follows the growing neural gas network algorithm of [8]. Every λ time steps, we check the current dictionary D1 for its possible update as follows: **4.1)** Each two corresponding neurons in $\mathcal{C}^c(n)$ and $\mathcal{C}^u(n)$ will be considered. If their distance is greater than ε_f, the weight vector of the neuron (codeword) in $\mathcal{C}^c(n)$ will be replaced by the one of the corresponding neuron in $\mathcal{C}^u(n)$. The weight vectors (codewords) in $\mathcal{C}^c(n)$ that are updated as a consequence of this check are sent to the receiver. **4.2)** For dictionary D1, the neuron p (synaptic weight vector $\boldsymbol{w}_p$) with the maximum accumulated error is determined. A new neuron r (synaptic weight vector $\boldsymbol{w}_r$) is generated halfway between p and its neighbor q that has the largest accumulated error:

$$\boldsymbol{w}_r = 0.5(\boldsymbol{w}_p + \boldsymbol{w}_q) \,. \quad (17)$$

The new neuron r is then added to both D1 and D3 and is also transmitted to the receiver to update the decoder's dictionary D1. For both D1 and D3, remove the edge connecting neurons p and q (edge (p, q)) and add the two edges (p, r) and (r, q). Multiply the accumulated error of p and q by constant α and initialize the accumulated error of r with the new value of the accumulated error of p.

Step 5) All the accumulated errors will be multiplied by a second constant β. After this, go to **Step 1** for the next input segment.

In the above algorithm, **Step 2** checks whether the current segment is matched by one codeword in the current dictionary D1. If not, the current feature vector is tagged as an unknown pattern and is added to dictionary D2 to go through an assessment phase. If instead a matching codeword in D1 is found, this codeword is used in **Step 3** to approximate the current segment. This is achieved by sending the index associated with this matching codeword to the receiver, which owns a copy of

dictionary D1 and uses the index to retrieve the approximating codeword. With **Step 4**, we periodically perform a dictionary assessment, i.e., we check whether the current dictionary D1 is still well representative of the actual input distribution. This assessment is accomplished by checking the distance between each codeword in D1 and its corresponding codeword in D3: if this distance gets too large (namely, larger than the maximum error tolerance ε_f), the codeword in D1 is replaced by its counterpart in D3. Note that, the higher ε_f, the higher the error tolerance and the lower the number of updates that are carried out for the current dictionary D1. Conversely, a small ε_f entails frequent dictionary updates. This regulates the actual representation error and also determines the maximum achievable compression efficiency. Moreover, we stress that in **Step 4** the update procedure is solely applied to those neurons that need to be updated as opposed to our previous design of Section V, where the whole dictionary is updated. This helps reduce the overhead associated with the dictionary update operation (see objective O2).

At the receiver, when compression is achieved picking the closest codeword from dictionary D1 and sending the corresponding index i^*, the n-th segment is reconstructed by picking the codeword $\boldsymbol{y}^*$ with index i^* from the local dictionary, moving it into the time domain through the inverse feature map, i.e., $\boldsymbol{x}^* = \Psi^{-1}(\boldsymbol{y}^*)$. Instead, when the feature vector $\boldsymbol{y}(n)$ is transmitted, the decompressor directly applies $\boldsymbol{x}^* = \Psi^{-1}(\boldsymbol{y}(n))$ to the received feature vector. In both cases then, the offset $e_{\boldsymbol{x}}(n)$ is added back to $\boldsymbol{x}^*$ and the latter is resized to the actual segment length $r_{\boldsymbol{x}}(n)$. This returns the reconstructed segment $\hat{\boldsymbol{x}}(n)$.

VII. Hardware architecture, energy model and performance metrics

To evaluate the **energy consumption**, following the approach of [32], [33] we compute three metrics:

1) *compression energy*: the energy consumption to execute the compression algorithm on the wearable node,
2) *transmission energy*: the energy drained by the the transmission of the (either compressed or original) signal over a wireless channel, and the
3) *total energy*, given by the sum of the previous two metrics.

1) Compression energy. The compression energy has been evaluated by taking into account the number of operations performed by the Micro-Controller Unit (MCU), i.e., the number of additions, multiplications, divisions and comparisons. Then, according to the considered sensor hardware, we translated these figures into the corresponding number of clock cycles $N_{\rm cc}$ and, from there, we derived the energy expenditure, as in [32]. For the energy consumption plots of Section VIII-A, we considered a Cortex M4-90LP [34] processor, whose number of clock cycles per operation is detailed in Table 7-1 of [35]. As for the energy consumption per clock cycle, $E_{\rm cc}$, in active mode the Cortex M4-90LP consumes 10.94 μA with the MCU operating at 1 MHz and the supply voltage being $+3$ V:

$$E_{\rm cc} = 10.94\ \mu\text{A} \times 3\ \text{V}/1\ \text{MHz} = 32.82 \cdot 10^{-12}\ \text{J}\,. \quad (18)$$

1) Transmission energy. When ECG samples are measured using a Zephyr Bioharness 3 module [36], the sampling frequency is 250 Hz, and each ECG sample takes 12 bits. This amounts to a transmission rate for a continuously streamed (uncompressed) ECG signal of 3 kbit/s. This is the setup considered for the results in Section VIII-B, whereas in Section VIII-A the bitrate is of 3.96 kbit/s, as the sampling rate is higher (360 Hz with 11 bits per sample). The raw ECG signal is then compressed using SURF and transmitted through the wireless channel. Next, we detail how we estimated the energy consumption associated with the transmission of data packets as they travel from the wearable device to the data receiver. Towards this end, we consider the energy consumption figure of the Bluetooth LE Texas Instruments CC2541 radio [37], whose energy consumption per transmitted bit is $E_{\rm bit} = 300$ nJ/bit (18.2 mA at 3.3 V considering a physical layer bitrate of 2 Mbps and the radio in active mode). The procedure that we now describe can be applied to any other radio, by plugging the corresponding $E_{\rm bit}$.

The energy consumption for each transmitted packet is obtained as $E_{\rm packet} = E_{\rm bit} \times \texttt{packet\_size}$, where $\texttt{packet\_size} = \texttt{header\_size} + \texttt{payload\_size}$. No energy consumption is accounted for when the radio is in idle mode (between packet transmissions). The packet transmission process follows the Bluetooth LE protocol in the *connected mode* (in our case, a point-to-point connection between only one master and only one slave). In Bluetooth LE, a data packet consists of the following fields: preamble (1 byte), access address (4 bytes), link layer header (2 bytes), L2CAP header (4 bytes), which are followed by 3 bytes of ATT command type/ attribute ID, $L_{\rm data}$ information bytes (containing application data), and the CRC field (3 bytes), see [38]. This leads to a total protocol overhead of $\texttt{header\_size} = 17$ bytes. For our results, we picked a payload size of $L_{\rm data} = \texttt{payload\_size} = 105$ unencoded information bytes (leading to a protocol overhead of $(17/122) \times 100 = 13.9\%$), although the numerical results can be promptly adapted for any other value. Each side communicates with the other on a given period called Connection Interval (CI), whose minimum value is 7.5 milliseconds. Each communication instance between the master and the slave is called a communication event, subsequent communication events are separated by CI seconds and a maximum of $n_{\rm max}$ data packets can be transmitted within this period. The maximum number of packets per seconds $\texttt{PPS}_{\rm max}$ that can be exchanged between the two devices is thus: $\texttt{PPS}_{\rm max} = n_{\rm max}/\text{CI}_{\rm min}$, with $\text{CI}_{\rm min}$ expressed in seconds, and the maximum throughput is obtained as

$$\texttt{Thr}_{\rm max} = \texttt{PPS}_{\rm max} \times \texttt{payload\_size}\,. \quad (19)$$

Here, $n_{\rm max}$ depends on the operating system of the terminals, for example, at the time of writing, Android has $n_{\rm max} = 6$, whereas iOS has $n_{\rm max} = 4$. Using (19), the maximum throughput for a wireless ECG monitor connected with an Android terminal ($n_{\rm max} = 6$), is thus: $\texttt{Thr}_{\rm max} = \texttt{PPS}_{\rm max} \times \texttt{payload\_size} = (6/0.0075) \times 105 = 672$ kbit/s. This maximum throughput is more than enough to support the transmission of the raw ECG signal (3 to 4 kbit/s).

The number of transmitted packets is computed according to the number of information bits that are to be transmitted by the radio, segmenting the bitstream into multiple data packets according to a fixed payload length of 105 information bytes. The energy consumption associated with the transmission of a single data packet is obtained as E_{packet}, as per the above discussion. Finally, the total energy consumption is computed as the sum of *processing* and *transmission* energy.

Two additional metrics are considered in the performance analysis, i.e., the Compression Efficiency (CE) and the Root Mean Square Error (RMSE). CE has been computed as the ratio between the total number of bits that would be required to transmit the full signal divided by those required for the transmission of the compressed bitstream. The RMSE is used to represent the reconstruction fidelity, as is computed as the root mean square error between the original and the compressed signals, normalizing it with respect to the signal's peak-to-peak amplitude (p2p), that is

$$\text{RMSE} = \frac{100}{\text{p2p}} \left(\frac{\sum_{i=1}^{K} (x_i - \hat{x}_i)^2}{K} \right)^{1/2}, \qquad (20)$$

where K corresponds to the total number of samples in the ECG trace, x_i and $\hat{x}_i$ are the original sample i and that reconstructed at the decompressor (receiver side), respectively. The SURF default parameters have been set as follows: $\epsilon_b = 0.01$, $\epsilon_n = 0.005$, $\alpha = 0.5$, $\beta = 0.995$, $\gamma = 3$, $L_{\max} = 10$, $\lambda = 200$ and $a_{max} = 100$. These parameters were selected empirically and provide a good tradeoff between RMSE and overhead (memory and compression efficiency).

VIII. Numerical Results

In this section, we show quantitative results for the proposed signal compression algorithms, detailing their energy consumption, compression efficiency and reconstruction fidelity.

In Section VIII-A, we first assess the performance of the considered compression algorithms for the reference ECG traces from the PhysioNet MIT-BIH arrhythmia database [17]. In Section VIII-B, we extend our analysis to (artifact prone) ECG traces that we collected from a Zephyr BioHarness 3 wearable chest monitor.

A. PhysioNet ECG traces

For the first set of graphs, we considered the following ECG traces from the MIT-BIH arrhythmia database [17]: 101, 112, 115, 117, 118, 201, 209, 212, 213, 219, 228, 231 and 232, which were sampled at rate of 360 samples/s with $11-$bit resolution. Note that not all the traces in the database are usable (some are very noisy due to heavy artifacts probably due to the disconnection of the sensing devices) and an educated selection has to be carried out for a meaningful performance analysis, as done in previous work [17], [39]. The above performance metrics were obtained for these ECG signals and their average values are shown in the following plots.

As a first result, in Figs. 4 and 5 we show the compression efficiency and the total energy consumption of SURF, both plotted versus the RMSE by varying ε_f as a free parameter. In these plots we quantify the impact of the feature space size f, corresponding to the number of DCT coefficients that are retained and stored in the feature vector $\boldsymbol{y}(n)$. Two SURF variants were implemented:

1) SURF-TD. It is a time domain implementation of SURF dictionaries, i.e., the feature vectors $\boldsymbol{y}(n)$ that are inputted into the dictionary correspond to the original signal segments, i.e., $\boldsymbol{y}(n) = \boldsymbol{x}(n)$. If an input segment is unmatched, the corresponding DCT coefficients are transmitted to the receiver. So, in this case the DCT transform is only applied if a new pattern that the current dictionary is unable to approximate is detected. In this case, f of its DCT coefficients are sent to reconstruct it at the receiver ($f = 200$ is used for the SURF-TD curve in Fig. 4).

2) SURF. It is the feature domain implementation that we have described in Section VI, for which we considered the following values for the feature space size $f \in \{50, 75, 100, 150, 200\}$ (see Figs. 4, 5 and 7).

From Fig. 4, we see that SURF achieves the highest CE, up to 90-fold for the considered PhysioNet signals, whereas time domain processing allows for maximum efficiencies of 60-fold. As expected, increasing f entails a smaller RMSE at the cost of a smaller CE. However, we see that when f increases beyond 100 the RMSE performance gets affected and starts decreasing. In these cases, SURF behaves similarly to its time domain counterpart. This is because dictionary construction in feature space allows for more robustness and generalization capabilities than working in the time domain, which may lead to overfitting codewords to specific signal examples. This means that an optimal value of f can be identified, which in our case is around $f \simeq 100$. Fig. 5 shows the total energy consumption (adding up processing and transmission) and we see that savings of almost two orders of magnitude with respect to the case where the signal is sent uncompressed ("no-compression") are possible. This is further discussed below.

In Fig. 6, we plot RMSE *vs* CE for SURF, comparing it against the TASOM-based algorithm of Section V and selected lossy compression techniques from the literature based on DCT [20], DWT [14], linear approximation LTC [10], GSVQ [9], and schemes based on compressive sensing: BSBL [40] and SOMP [15]. At very low compression efficiencies LTC outperforms DCT in terms of RMSE. DWT does a much better job than DCT in terms of RMSE, especially at relatively small compression efficiencies, say, smaller than 30, but it is unable to reach higher CEs, for which LTC, TASOM and SURF are to be preferred. As for the CS-based algorithms, neither SOMP nor BSBL provides satisfactory performance. The compression efficiency of SOMP is rather small and the corresponding RMSE tends to diverge for, e.g., CE larger than 5. As we shall discuss shortly, although the BSBL compressor has the lowest energy consumption, its overall energy expenditure is high as this approach is less effective in terms of CE than other schemes such as dictionary

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2749758, IEEE Access

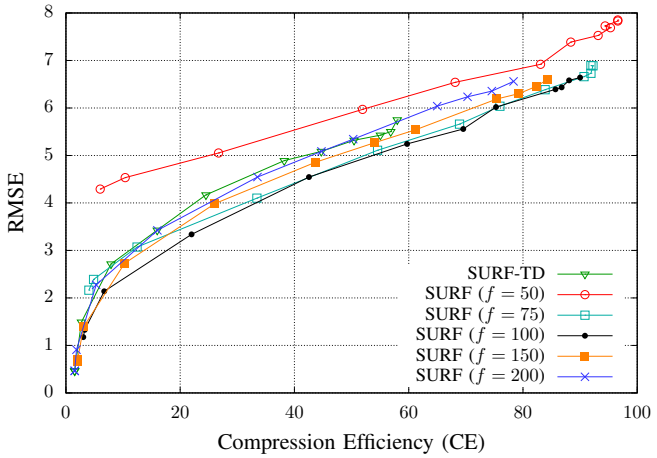


Fig. 4: SURF – RMSE *vs* compression efficiency.

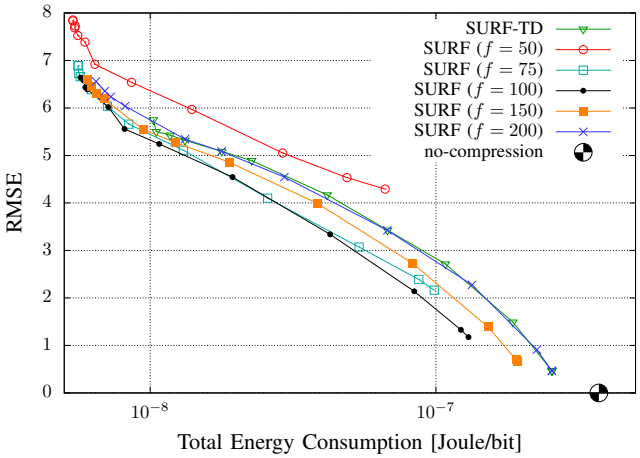


Fig. 5: SURF – RMSE *vs* total energy consumption.

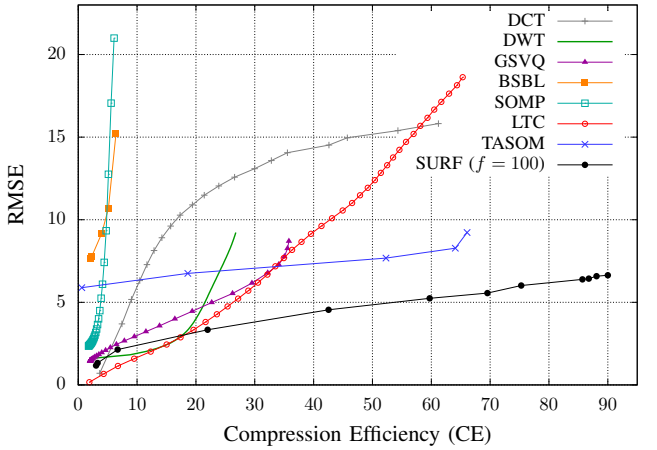


Fig. 6: RMSE *vs* CE – comparison of compression algorithms.

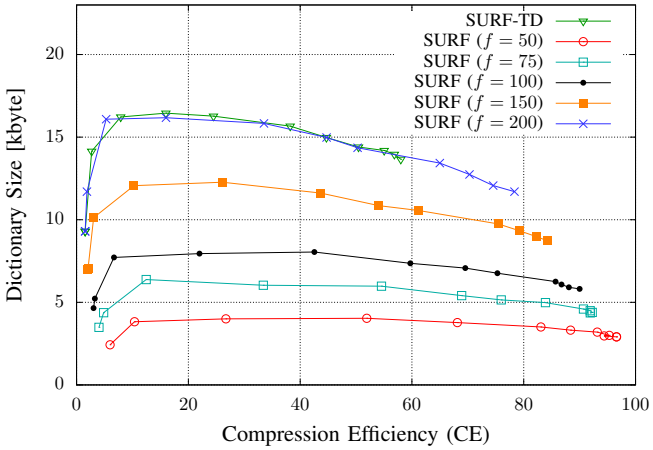


Fig. 7: SURF dictionary size *vs* CE. The tradeoff curves are obtained by varying the error tolerance ε_f.

based (GSVQ) and neural map based algorithms (TASOM and SURF). For GSVQ, we move along the RMSE *vs* CE curves by changing the threshold governing the number of bits that are encoded into the residual stream (residual encoding is the operation that affects the performance of GSVQ the most). Although not shown in the plot, with GSVQ one may think of not sending the residual encoding stream, so as to reach higher compression efficiencies. However, due to the use of a precomputed and *fixed* dictionary, this leads to a very high RMSE and is not a viable option. SURF offers very good performance both in terms of RMSE and CE, thus clearly outperforming the other algorithms. We also emphasize the substantial gap in both RMSE and CE that SURF achieves with respect to TASOM. The reasons for this are: i) SURF dictionaries more effectively represent new patterns and artifacts, ii) SURF works in the signal feature space, where the size of codewords is $f < m$ elements and iii) dictionary updates are selectively implemented only for those codewords that no longer meet the error tolerance ε_f.

For SURF, we also look at the size of dictionaries as a function of CE. In Fig. 7, we plot the total size of dictionaries D1, D2 and D3 and we see that it never exceeds $17\,\mathrm{kbytes}$. For this reason, the approach is deemed amenable to implementation on wearables. We also note that the size at first (small CE) increases up to a maximum and then starts decreasing for higher CE. This is because when the error tolerance ε_f is very small, the compressor often sends the full feature vector as none of the current codewords will match the new segment. Also, as a new pattern is detected and the corresponding feature vector is added to dictionary D2, this codeword will be put into use (moving it to D1 and D3) with small probability, as further "nearly exact" ($\varepsilon_f \to 0$) matches are difficult to occur for it. On the other hand, as ε_f increases, more codewords will be added to the dictionary and they will be used to encode multiple patterns each. However, as ε_f keeps increasing beyond a certain threshold, because of the relaxed accuracy requirement, a smaller codeword set will suffice to represent the input signal space and the dictionary size will correspondingly decrease. Again, $f = 100$ was found to be a good choice, requiring less than $10\,\mathrm{kbytes}$ of memory, while resulting in very high CEs. For this reason, $f = 100$ is used for SURF in the following graphs. Two further graphs, Figs. 8 and 9, quantify the impact of the maximum number of codewords in the dictionaries, $L_{\max}$, which corresponds to the number of neurons in the adopted GNG neural networks. The error tolerance ε_f is varied as an independent parameter

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2749758, IEEE Access

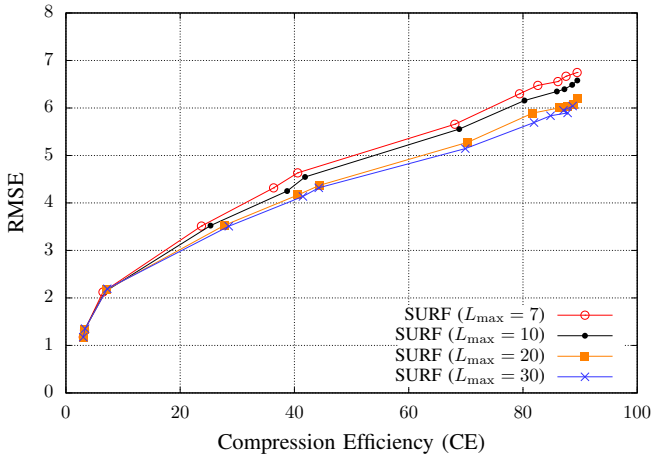


Fig. 8: SURF – RMSE *vs* compression efficiency.

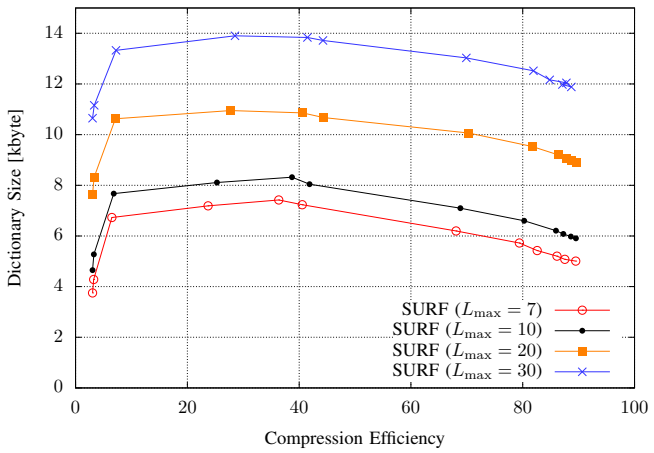


Fig. 9: SURF – Dictionary size *vs* compression efficiency.

in both plots. As expected, Fig. 8 shows that a higher $L_{\max}$ leads to a higher accuracy, i.e., the current dictionary more accurately represents the input signal. Nevertheless, we see that the accuracy increase is not very large, whereas there is a substantial difference in the overall memory space that is taken by the dictionaries as $L_{\max}$ increases. A number of neurons per dictionary between 15 and 20 appears to be a good choice, as further increasing $L_{\max}$ from 20 to 30 only leads to minor fidelity improvements (RMSE). Once $L_{\max}$ is fixed, the error tolerance can be used to tune the RMSE as desired.

In Fig. 10, we show the RMSE and the energy drained for compression (processing) at the transmitter, expressed in Joule per bit of the original ECG sequence. These tradeoff curves are obtained by varying the compression efficiency of each algorithm from the minimum to the maximum achievable (which is scheme specific, see Fig. 6). The RMSE increases with an increasing compression efficiency, whereas the compression energy depends weakly on CE. As expected, BSBL has the smallest energy consumption. This good performance is due to its lightweight compression algorithm, which just multiplies the input signal by sparse binary matrices. LTC is the second best, whereas SOMP, GSVQ and TASOM perform very close to one another and have the worst energy consumption for compression, although SURF consumes a slightly smaller amount of energy than them. We underline that the energy consumption of SURF, TASOM, SOMP and GSVQ is dominated by the preprocessing chain of Fig. 3 (as we quantify below through Tables I and II). In Fig. 10, we also show the performance of SURF by removing the contribution of this pre-processing chain (filtering, peak detection and segment extraction): the corresponding curve is referred to in the plot as "SURF NoPre". Note that filtering is always performed to remove measurement artifacts and peak detection is also very often utilized to extract relevant signal features. Given this, the energy consumption associated with the required pre-processing functions may not be a problem, especially if these functions are to be executed anyway.

In Fig. 11, we show the RMSE as a function of the *total energy consumption*, which is obtained adding up the energy required for compression to that for the subsequent transmis-

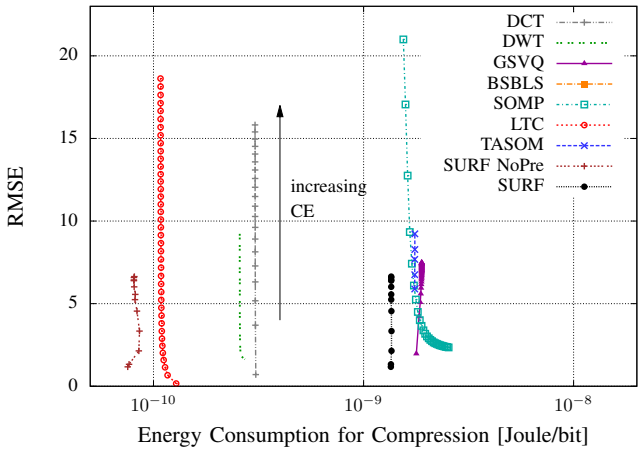


Fig. 10: RMSE *vs* compression energy obtained varying CE as the independent parameter.

sion of the compressed bitstream, as detailed in Section VII. This total energy is then normalized with respect to the number of bits in the original ECG signal. From this plot, we see that the total energy consumption is dominated by the transmission energy, which depends on the compression efficiency. In this respect, the best algorithms are LTC and SURF and the algorithm of choice depends on the target RMSE that, in turn, directly descends from the selected CE. As discussed above, an adaptive algorithm may be a good option, where for each value of CE the scheme that provides the smallest RMSE is used. In Fig. 11, the energy consumption when no compression is applied is also shown for comparison. We see that signal compression, and the subsequent reduction in size of the data to be transmitted, allows a considerable decrease in the total energy consumption. When the energy reduction is one order of magnitude, LTC and SURF both provide RMSEs smaller than 2%. The performance of SURF is particularly striking as it allows saving up to two order of magnitude in terms of energy consumption while still keeping the RMSE around 6%. Also, note that SURF's actual RMSE is automatically adjusted at runtime, by allowing slightly less accurate representations, and thus much higher compression, when no critical patterns

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2749758, IEEE Access

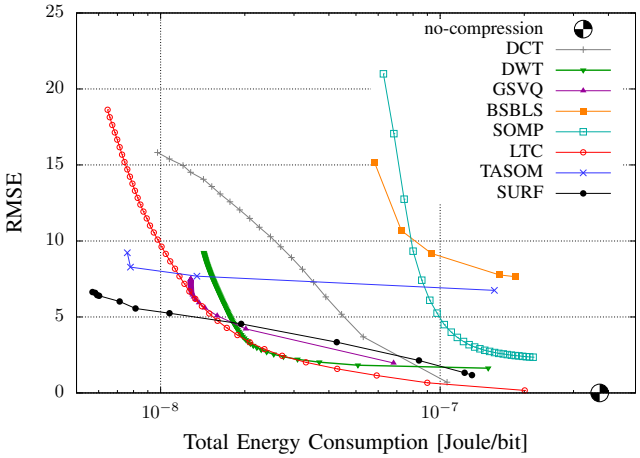


Fig. 11: RMSE *vs* total energy consumption.

occur.

A breakdown of the complexity and energy consumption figures for the considered algorithms is provided in Tables I and II. These metrics were obtained for the PhysioNet ECG signals and represent the average complexity (expressed in terms of number of operations) and energy consumption (Joules) for the compression and transmission of an ECG segment. From Table II, we see that SURF has a lower energy consumption with respect to TASOM for compression, transmission and in total. We also see that the peak detection block of TASOM and SURF accounts for 91% of the per-segment energy drainage. The same fact applies to the other segment-based approaches (SOMP, GSVQ).

The plots in Fig. 12 show original and reconstructed ECG temporal signals using LTC, TASOM and SURF, in the presence of anomalous ECG segments (toward the middle of the plots). Remarkably, although all algorithms have the same average RMSE, LTC heavily affects the ECG morphology. TASOM does a better job, but its dictionary is unable to effectively represent the new (anomalous) patterns. SURF provides the best results as it preserves the signal morphology, while achieving the highest CE, i.e., up to CE $= 53$.

B. Wearable ECG Signals

We now present some results for ECG signals that we acquired from a Zephyr BioHarness 3 wearable device [36]. To this end, we collected ECG traces from eleven healthy individuals, which were continuously recorded during working hours, i.e., from 8am to 6pm. These were sampled at a rate of 25 samples/s with each sample taking 12 bits.

The RMSE *vs* CE tradeoff for these signals is shown in Fig. 13 for the best performing compression algorithms. The results are similar to those of Fig. 6 with the main difference that in this case the ECG signals are prone to artifacts. Due to the artifacts and to the highly *non-stationary* behavior of the new traces, the resulting RMSE is higher and the CE performance is degraded for all schemes. DWT and LTC are good choices at low up to intermediate compression efficiencies, whereas SURF shows its superior performance at very high CEs, and especially its ability to gracefully adapt to artifact-prone and non-stationary signals. Although its maximum compression efficiency is affected, being lowered from 96 to 50, the RMSE remains within 6% and is much smaller than that achieved by all other schemes. SURF, with artifact prone ECG signals, allows for typical compression efficiencies in the range CE $\in [40, 50]$, which means that the data rate of 3 kbit/s that would be required to send the uncompressed ECG trace is lowered to 60 bit/s and 75 bit/s for CE $= 40$ and CE $= 50$, respectively. The energy consumption figures, although rescaled, have a very similar behavior as those obtained with the PhysioNet MIT-BIH traces and shown in Figs. 10 and 11. They are thus not shown in the interest of space.

In Figs. 14 and 15, we respectively show how the RMSE and CE evolve with time for LTC, TASOM and SURF, where these metrics are shown for each new ECG segment. For the RMSE (Fig. 14), we see that both TASOM and SURF provide excellent approximation accuracy. However, when artifacts occur, at around times 1100 and 2500 (at the end of the plot) we see that TASOM struggles to keep the RMSE low. SURF instead still provides satisfactory RMSE performance thanks to its adaptive mechanism by which feature vectors are transmitted in place of dictionary indices. From Fig. 15, we see that at times SURF's compression efficiency is reduced. This is either due to dictionary updates, which, with the considered SURF parameters, occur every $\lambda = 200$ time steps (ECG segments) or to artifacts, which in this figure are seen again around ECG segments 1100 or 1500 (the same portion of ECG signal is used for the last two figures).

In Fig. 16, we analyze the training behavior (RMSE versus time) for dictionary D3, which is continuously updated at the transmitter. Note that the current dictionary D1 is replaced with D3 when the distance among their codewords exceeds a given threshold. So, the evolution of D3, although at a coarser time scale, also represents that of D1. To obtain this plot, we ran the following experiment: we picked a first subject and we trained D3 with their ECG signal for the first 55 minutes, at which point, the input signal was swapped with that of a second subject. Two curves are shown in the figure, using $L_{\max} = 10$ and $L_{\max} = 30$ and keeping all the remaining parameters as specified at the beginning of the section. At time zero, the dictionary is initialized using random ECG segments from the first subject, whereas its subsequent training follows the GNG-based algorithms of Section VI. A few observations are in order. As expected, when the training starts the error is higher (the RMSE is higher than 4% for the first subject for $L_{\max} = 10$) but it decreases with time and converges to the steady-state error within 20 minutes. After 55 minutes, the signal is swapped with that of another subject and this may for example occur when the wireless ECG monitor is handled over to another patient. At this point, we observe a peak in the RMSE, which suddenly increases from 2.8% to 4.1%. However, D3 is retrained and in about 20 more minutes converges to the new steady-state RMSE for the second subject. This shows that SURF gracefully adapts to new wearers, progressively tuning its dictionaries to their ECG patterns. From this graph, we also see that the RMSE depends on the maximum number

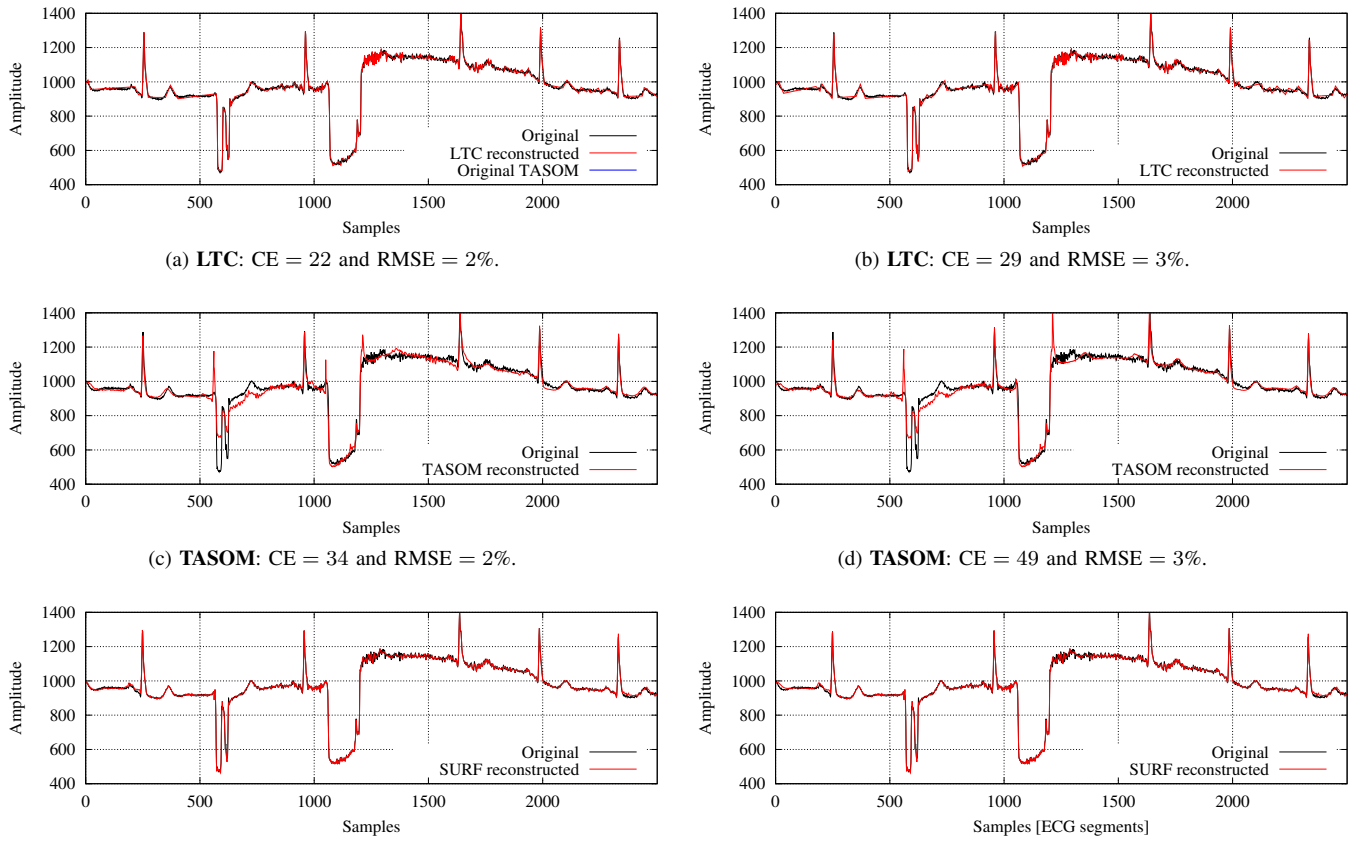
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2749758, IEEE Access

TABLE I: Energy breakdown [no. operations] and consumption [μJ] for TASOM. RMSE = 3.6% and CE = 20.92.

	Pass band filtering	**Peak detection**	**Segment extractor**	**Pattern matching**	**Codebook manager**	**Total**
Additions	5420	68136	592	2885	7103	84136
Multiplications	5032	67362	298	2747	4260	79699
Divisions	387.13	775.27	3	0	29	1193.4
Comparisons	0	580.7	0	193.14	22.51	796.35
Compression energy [μJ]	0.52	4.82	0.03	0.19	0.39	**5.95**
Transmission energy [μJ]	–	–	–	–	–	**75.3**
Total energy [μJ]						**81.25**

TABLE II: Energy breakdown [no. operations] and consumption [μJ] for SURF. RMSE = 3.6% and CE = 76.6.

	Pass band filtering	**Peak detection**	**Segment extractor**	**Pattern matching**	**Codebook manager**	**Total**
Additions	5413.4	68055	2497	1366.5	1204.1	78536
Multiplications	5027	67281	1498	1380.31	610.69	75797
Divisions	386.65	773.35	2	0	0	1162
Comparisons	0	580.01	0	11.45	116.64	708.1
Compression energy [μJ]	0.52	4.82	0.13	0.09	0.06	**5.62**
Transmission energy [μJ]	–	–	–	–	–	**20.6**
Total energy [μJ]						**26.12**



(a) **LTC**: CE = 22 and RMSE = 2%.

(b) **LTC**: CE = 29 and RMSE = 3%.

(c) **TASOM**: CE = 34 and RMSE = 2%.

(d) **TASOM**: CE = 49 and RMSE = 3%.

(e) **SURF**: CE = 43 and RMSE = 2%.

(f) **SURF**: CE = 53 and RMSE = 3%.

Fig. 12: Original and reconstructed signal in the presence of artifacts for LTC, TASOM and SURF.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2749758, IEEE Access

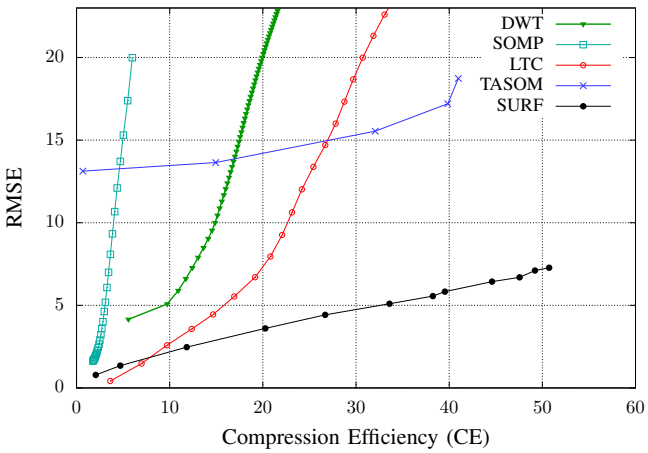


Fig. 13: RMSE *vs* CE for Bioharness ECG signals.

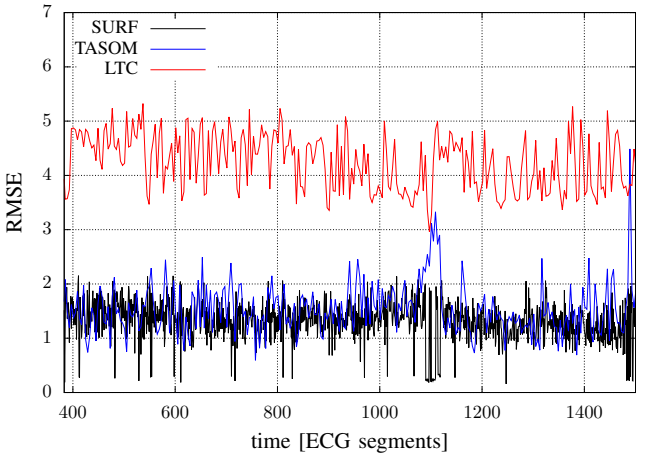


Fig. 14: RMSE as a function of time. CE $=$ 25 for all schemes, RMSE(LTC) $=$ 5%, RMSE(TASOM) $=$ 1.58% and RMSE(SURF) $=$ 1.29%.

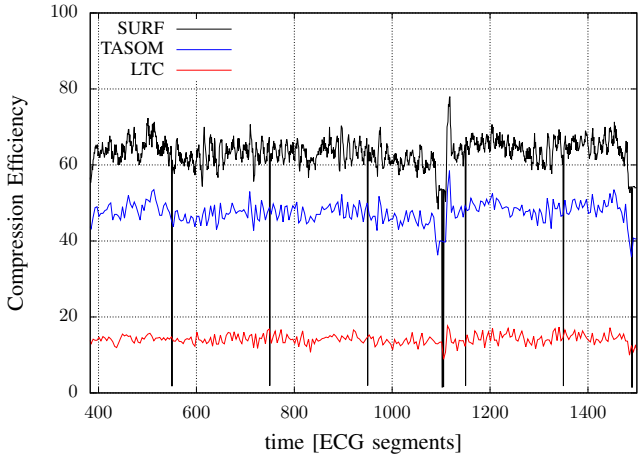


Fig. 15: CE as a function of time. RMSE $=$ 2% for all schemes, CE(LTC) $=$ 14, CE(TASOM) $=$ 42 and CE(SURF) $=$ 50.

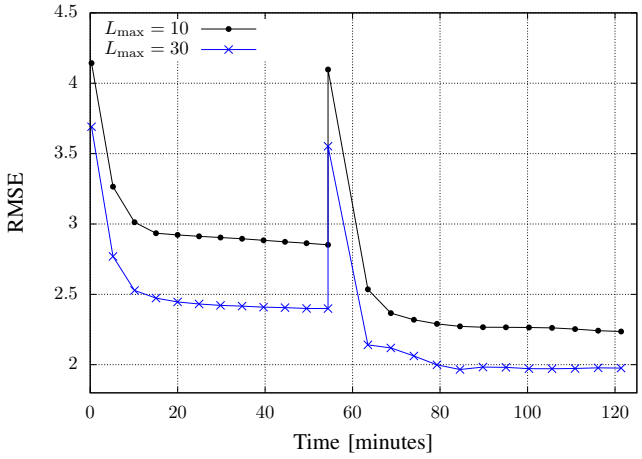


Fig. 16: Average normalized RMSE versus training time for the updated dictionary D3: the dictionary is trained on a first subject for the first 55 minutes. After that, the ECG trace of a different subject is used as the input time series. The dictionary at first produces high errors, but then quickly adapts and converges to the steady-state RMSE for the second subject.

of codewords in the dictionary, $L_{\max}$: an increasing $L_{\max}$ leads to higher accuracies. As a last remark, we recall that the RMSE in Fig. 16 corresponds to the representation error of SURF dictionaries, but the actual RMSE of the full SURF algorithm is always within the preset error tolerance. In fact, according to the algorithms of Section VI, when the RMSE is higher than a preset threshold the dictionary is not used, but the feature vector associated with the current segment is sent as the compressed representation. In other words, SURF automatically switches between dictionary-based compression and feature-based (e.g., DCT) compression, meeting the preset representation accuracy at all times.

Fig. 17 shows the energy consumption associated with radio transmission and processing, identifying the region where compression provides energy savings and it is therefore recommended. We obtained this plot as follows. Let B and $\hat{B}$ respectively be the number of bits to send over the channel when no compression is applied and those to be sent when the signal is compressed. With $N_{\rm cc}(B, \varepsilon_f)$ we indicate the number of clock cycles that are needed to run the compression algorithm, which depends on the number bits B in the original signal and on the compression error ε_f (which dictates a certain compression factor). Compression is convenient when the following inequality holds:

$$E_{\rm cc} N_{\rm cc}(B, \varepsilon_f) + E^0_{\rm tx} \hat{B} < E^0_{\rm tx} B \,, \tag{21}$$

which means that the energy for compression added to that for transmission of the compressed sequence (left hand side) must be smaller than the energy that would be required to send the uncompressed signal (right hand side). Solving this inequality for $E^0_{\rm tx}$, we find the minimum $E^0_{\rm tx}$ that allows compression to be energy efficient, that is:

$$E^{0,\min}_{\rm tx} = \frac{E_{\rm cc} N_{\rm cc}(B, \varepsilon_f)}{B - \hat{B}} \,. \tag{22}$$

The lines plotted in Fig. 17 correspond to $E^{0,\min}_{\rm tx}$ computed for several values of ε_f, which in turn imply different compression efficiencies (CE in the figure). The region in this plot where compression is advantageous (*energy efficient region*) is that

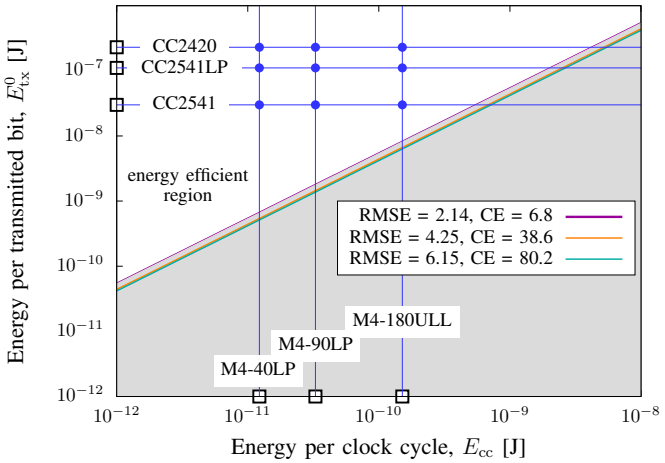


Fig. 17: Efficiency regions for SURF compression with different radios and MCUs. Radios: CC2420 (250 kbit/s, power 0 dBm), CC2541 (2 Mbit/s, at maximum power 0 dBm), CC2541LP (low rate 500 kbit/s and power −20 dBm). MCUs: Cortex-M4 versions 40LP, 90LP, 180ULL.

for which $E_{\text{tx}}^0 > E_{\text{tx}}^{0,\min}$, which corresponds to the region above the curves. As seen from the plot, the energy efficient regions weakly depend on the compression parameters as the number of clock cycles is almost constant for different settings (changing ε_f), B is also constant and it depends on the sampling rate of the ECG monitor, and the only variable that changes is $\hat{B}$. Most importantly, in the graph we have also reported the energy consumption figures (E_{tx}^0 and E_{cc}) of several radios and MCUs (each radio/MCU pair is indicated by a filled dot in the figure). All of them fall within the efficient region and, as expected, compression provides the highest gain when the radio is energy hungry (CC2420) and the processor is energy efficient (Cortex M4-40LP). Before applying the SURF algorithm to any architecture, one should make sure that the selected combination of radio and MCU operates within the energy efficient region of Fig. 17.

IX. Conclusions

In this paper, we have presented SURF, an original *subject-specific* and *time-adaptive* lossy compression algorithm for wearable fitness monitors. This algorithm is based upon dictionaries that are learned and maintained at runtime through the use of neural network maps. Our design utilizes unsupervised learning to accomplish the following objectives: i) dictionaries gracefully and effectively adapt to new subjects or their new activities, ii) the size of these dictionaries is kept bounded (i.e., within 20 kbytes), making them amenable to implementation in wireless monitors, iii) high compression efficiencies are reached, allowing for reductions in the signal size from 50- to 96-fold, depending on the frequency of artifacts in the sampled signal, iv) the original biometric time series are reconstructed at the receiver with high accuracy, i.e., within a peak-to-peak RMSE of 7% and often smaller than 3% and v) compression allows saving energy at the transmitter, lowering the total energy expenditure of almost two orders of magnitude. SURF outperforms the compression approaches that were proposed thus far. Although in this paper SURF has been designed and tested with ECG signals, it can be applied to other quasi-periodic signals as long as a reliable segment extraction technique is provided.

Acknowledgment

The work of Dadive Zordan, Mohsen Hooshmand and Michele Rossi was supported by Samsung Advanced Institute of Technology (SAIT), Korea, as part of its SAMSUNG Global Research Outreach (GRO) program and by the project IoT-SURF (CPDA 151221), funded by the University of Padova. The Work of Tommaso Melodia was supported by the US National Science Foundation under grants no. CNS-1253309 and CNS-1618731.

References

[1] P. Soh, G. Vandenbosch, M. Mercuri, and D.-P. Schreurs, "Wearable Wireless Health Monitoring: Current Developments, Challenges, and Future Trends," *IEEE Microwave Magazine*, vol. 16, no. 4, pp. 55–70, May 2015.

[2] M. Srivastava, T. Abdelzaher, and B. Szymanski, "Human-centric Sensing," *Phylosophical Transactions of Royal Society*, vol. 370, no. 1958, pp. 176–197, Jan. 2012.

[3] S. Riazul Islam, D. Kwak, M. Humaun Kabir, M. Hossain, and K.-S. Kwak, "The Internet of Things for Health Care: A Comprehensive Survey," *IEEE Access*, vol. 3, pp. 678–708, Jun. 2015.

[4] V. Vadori, E. Grisan, and M. Rossi, "Biomedical Signal Compression with Time- and Subject-adaptive Dictionary for Wearable Devices," in *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Vietri sul Mare, Salerno, Italy, Sep. 2016.

[5] T. Kohonen, "The Self-Organizing Map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.

[6] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, US: Prentice Hall, 1998.

[7] H. Shah-Hosseini and R. Safabakhsh, "TASOM: the time adaptive self-organizing map," in *Proceedings of the International Conference on Information Technology: Coding and Computing*, Las Vegas, NV, US, Mar. 2000, pp. 422–427.

[8] B. Fritzke, *A Growing Neural Gas Network Learns Topologies*. MIT Press, 1995.

[9] C. C. Sun and S. C. Tai, "Beat-based ECG compression using gain-shape vector quantization," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 11, pp. 1882–1888, Nov. 2005.

[10] T. Schoellhammer, B. Greenstein, M. W. E. Osterweil, and D. Estrin, "Lightweight temporal compression of microclimate datasets," in *Proceedings of the IEEE International Conference on Local Computer Networks (LCN)*, Tampa, FL, US, Nov. 2004.

[11] R. Shankara and S. M. Ivaturi, "ECG Data Compression Using Fourier Descriptors," *IEEE Transactions on Biomedical Engineering*, vol. 33, no. 4, pp. 428–434, Apr. 1986.

[12] V. Allen and J. Belina, "ECG data compression using the discrete cosine transform (DCT)," in *Proceedings of Computers in Cardiology*, Durham, NC, US, Oct. 1992, pp. 687–690.

[13] D. Zordan, B. Martinez, I. Villajosana, and M. Rossi, "On the Performance of Lossy Compression Schemes for Energy Constrained Sensor Networking," *ACM Transactions on Sensor Networks*, vol. 11, no. 1, pp. 15:1–15:34, Nov. 2014.

[14] B. A. Rajoub, "An Efficient Coding Algorithm for the Compression of ECG Signals Using the Wavelet Transform," *IEEE Transactions on Biomedical Engineering*, vol. 49, no. 4, pp. 355–362, Apr. 2002.

[15] L. Polania, R. Carrillo, M. Blanco-Velasco, and K. Barner, "Compressed sensing based method for ECG compression," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 761–764.

[16] G. D. Poian, R. Bernardini, and R. Rinaldo, "Gaussian dictionary for Compressive Sensing of the ECG signal," in *Proceedings of the IEEE Workshop on Biometric Measurements and Systems for Security and Medical Applications (BIOMS)*, Rome, Italy, Oct. 2014.

[17] M. Saeed and M. Villarroel and A.T. Reisner and G. Clifford and L. Lehman and G.B. Moody and T. Heldt and T.H. Kyaw and B.E. Moody and R.G. Mark, "Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): a public-access intensive care unit database," *Critical Care Medicine*, vol. 39, no. 5, pp. 952–960, May 2011.

[18] J. Cox, H. Fozzard, F. M. Nolle, and G. Oliver, "AZTEC: A preprocessing system for real-time ECG rhythm analysis," *IEEE Transactions on Biomedical Engineering*, vol. 37, no. 9, pp. 128–129, Apr. 1968.

[19] J. P. Abenstein and W. J. Tompkins, "A New Data-Reduction Algorithm for Real-Time ECG Analysis," *IEEE Transactions on Biomedical Engineering*, vol. 29, no. 1, pp. 43–48, Jan. 1982.

[20] H. Lee and K. M. Buckley, "ECG data compression using cut and align beats approach and 2-D transforms," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 5, pp. 556–564, May 1999.

[21] J. Cardenas-Barrera and J. Lorenzo-Ginori, "Mean-shape vector quantizer for ECG signal compression," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 1, pp. 62–70, Jan. 1999.

[22] S.-G. Miaou and J.-H. Larn, "Adaptive vector quantisation for electrocardiogram signal compression using overlapped and linearly shifted codevectors," *Medical and Biological Engineering and Computing*, vol. 38, no. 5, pp. 547–552, 2000.

[23] A. Chatterjee, A. Nait-Ali, and P. Siarry, "An input-delay neural-network-based approach for piecewise ecg signal compression," *IEEE transactions on biomedical engineering*, vol. 52, no. 5, pp. 945–947, 2005.

[24] D. Del Testa and M. Rossi, "Lightweight Lossy Compression of Biometric Patterns via Denoising Autoencoders," *IEEE Signal Processing Letters*, vol. 22, no. 12, pp. 2304–2308, Dec. 2015.

[25] Y. Linde, A. Buzo, and R. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, Jan. 1980.

[26] M. Hooshmand, D. Zordan, D. Del Testa, E. Grisan, and M. Rossi, "Boosting the Battery Life of Wearables for Health Monitoring through the Compression of Biosignals," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–16, Mar. 2017.

[27] A. Gersho and R. M. Gray, *Vector quantization and signal compression*. Springer Science & Business Media, 2012, vol. 159.

[28] K.-S. Wu and J.-C. Lin, "Fast vq encoding by an efficient kick-out condition," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 10, no. 1, pp. 59–62, 2000.

[29] M. Elgendi, "Fast QRS Detection with an Optimized Knowledge-Based Method: Evaluation on 11 Standard ECG Databases," *PLoS ONE*, vol. 8, no. 9, pp. 1–18, Sep. 2013.

[30] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "Neural-gas Network for Vector Quantization and Its Application to Time-series Prediction," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 558–569, Jul. 1993.

[31] B. Fritzke, "Growing Cell Structures: A Self-Organizing Network for Unsupervised and Supervised Learning," *Neural Networks*, vol. 7, no. 9, pp. 1441–1460, 1994.

[32] C. Karakus, A. C. Gurbuz, and B. Tavli, "Analysis of Energy Efficiency of Compressive Sensing in Wireless Sensor Networks," *IEEE Sensors Journal*, vol. 13, no. 5, pp. 1999–2008, May 2013.

[33] M. Hooshmand, M. Rossi, D. Zordan, and M. Zorzi, "Covariogram-based Compressive Sensing for Environmental Wireless Sensor Networks," *IEEE Sensors Journal*, vol. 16, no. 6, pp. 1716–1729, Mar. 2016.

[34] ARM The Architecture for the Digital World, "ARM Cortex-M4 Processor," 2015. [Online]. Available: http://www.arm.com/products/processors/cortex-m/

[35] ARM®, "Cortex-M4 Technical Reference Manual," 2010. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439b/DDI0439B_cortex_m4_r0p0_trm.pdf

[36] Zephyr Technology Corporation, "Bioharness 3 - Wireless Professional Heart Rate Monitor and Physiological Monitor," 2017. [Online]. Available: http://www.zephyranywhere.com/

[37] Texas Instruments, "CC2451: 2.4 GHz Low Energy and Proprietary System-on-Chip," 2015. [Online]. Available: http://www.ti.com/product/cc2541

[38] "Specification of the Bluetooth System v4.2," Bluetooth Core Specification Standard, Dec. 2014. [Online]. Available: https://www.bluetooth.com

[39] Y. Zigel, A. Cohen, and A. Katz, "ECG Signal Compression Using Analysis by Synthesis Coding," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 10, pp. 1308–1316, Oct. 2000.

[40] Z. Zhang, T.-P. Jung, S. Makeig, and B. D. Rao, "Compressed Sensing for Energy-Efficient Wireless Telemonitoring of Noninvasive Fetal ECG Via Block Sparse Bayesian Learning," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 2, pp. 300–309, Feb. 2013.