UNIVERSITÀ
DEGLI STUDI
DI PADOVA

SEDE AMMINISTRATIVA: UNIVERSITÀ DEGLI STUDI DI PADOVA
Dipartimento di Ingegneria dell'Informazione

DOTTORATO DI RICERCA IN : INGEGNERIA DELL'INFORMAZIONE
CURRICOLO: SCEINZA E TECNOLOGIA DELL'INFORMAZIONE
CICLO: XXIX

Sensing and Compression Techniques for Environmental and Human Sensing Applications

Coordinatore: Ch.mo Prof. Matteo BERTOCCO
Supervisore: Ch.mo Prof. Michele ROSSI

Dottorando: Mohsen HOOSHMAND

Academic Year 2016-2017

Abstract

In this doctoral thesis, we devise and evaluate a variety of lossy compression schemes for Internet of Things (IoT) devices such as those utilized in environmental wireless sensor networks (WSNs) and Body Sensor Networks (BSNs). We are especially concerned with the efficient acquisition of the data sensed by these systems and to this end we advocate the use of joint (lossy) compression and transmission techniques.

Environmental WSNs are considered first. For these, we present an original compressive sensing (CS) approach for the spatio-temporal compression of data. In detail, we consider temporal compression schemes based on linear approximations as well as Fourier transforms, whereas spatial and/or temporal dynamics are exploited through compression algorithms based on distributed source coding (DSC) and several algorithms based on compressive sensing (CS). To the best of our knowledge, this is the first work presenting a systematic performance evaluation of these (different) lossy compression approaches. The selected algorithms are framed within the same system model, and a comparative performance assessment is carried out, evaluating their energy consumption *vs* the attainable compression ratio. Hence, as a further main contribution of this thesis, we design and validate a novel CS-based compression scheme, termed covariogram-based compressive sensing (CB-CS), which combines a new sampling mechanism along with an original covariogram-based approach for the online estimation of the covariance structure of the signal.

As a second main research topic, we focus on modern wearable IoT devices which enable the monitoring of vital parameters such as heart or respiratory rates (RESP), electrocardiography (ECG), and photo-plethysmographic (PPG) signals within e-health applications. These devices are battery operated and communicate the vital signs they gather through a wireless communication interface. A common issue of this technology is that signal transmission is often power-demanding and this poses serious limitations to the continuous monitoring of biometric signals. To ameliorate this, we advocate the use of lossy signal compression at the source: this considerably reduces the size of the data that has to be sent to the acquisition point by, in turn, boosting the battery life of the wearables and allowing for fine-grained

and long-term monitoring. Considering one dimensional biosignals such as ECG, RESP and PPG, which are often available from commercial wearable devices, we first provide a throughout review of existing compression algorithms. Hence, we present novel approaches based on online dictionaries, elucidating their operating principles and providing a quantitative assessment of compression, reconstruction and energy consumption performance of all schemes. As part of this first investigation, dictionaries are built using a suboptimal but lightweight, online and best effort algorithm. Surprisingly, the obtained compression scheme is found to be very effective both in terms of compression efficiencies and reconstruction accuracy at the receiver. This approach is however not yet amenable to its practical implementation as its memory usage is rather high. Also, our systematic performance assessment reveals that the most efficient compression algorithms allow reductions in the signal size of up to 100 times, which entail similar reductions in the energy demand, by still keeping the reconstruction error within 4% of the peak-to-peak signal amplitude.

Based on what we have learned from this first comparison, we finally propose a new *subject-specific* compression technique called SURF "Subject-adpative Unsupervised ecg compressor for weaRable Fitness monitors". In SURF, dictionaries are learned and maintained using suitable neural network structures. Specifically, learning is achieve through the use of neural maps such as self organizing maps and growing neural gas networks, in a totally unsupervised manner and adapting the dictionaries to the signal statistics of the wearer. As our results show, SURF: i) reaches high compression efficiencies (reduction in the signal size of up to 96 times), ii) allows for reconstruction errors well below 4% (peak-to-peak RMSE, errors of 2% are generally achievable), iii) gracefully adapts to changing signal statistics due to switching to a new subject or changing their activity, iv) has low memory requirements (lower than 50kbytes) and v) allows for further reduction in the total energy consumption (processing plus transmission). These facts makes SURF a very promising algorithm, delivering the best performance among all the solutions proposed so far.

Contents

List of Figures

List of Tables

Abbreviations

AZTEC	**A**mplitude **Z**one **T**ime **E**poch **C**oding
BAN	**B**ody **A**rea **N**etwork
BSBL	**B**lock **S**parse **B**ayesian **L**earning
CB-DNS	**C**orrelation **B**ased **D**eterministic **N**ode **S**election
CC	**C**entroid **C**ondition
CE	**C**ompression **E**fficiency
CORTES	**C**oordinate **R**eduction **T**ime **E**ncoding **S**ystem
CPU	**C**entral **P**rocessing **U**nit
DNS	**D**eterministic **N**ode **S**election
DCT	**D**iscrete **C**osine **T**ransform
DFT	**D**iscrete **F**ourier **T**ransform
DSP	**D**igital **S**ignal **P**rocessing
DWT	**D**iscrete **W**avelet **T**ransform
ECB-DNS	**E**nhanced **C**orrelation **B**ased **D**eterministic **N**ode **S**election
ECG	**E**lectro**C**ardio**G**ram(**G**raphy
FFT	**F**ast **F**ourier **T**ransform
FPU	**F**loating **P**oint **U**nit
GNG	**G**rowing **N**eural **G**as
GSVQ	**G**ain-**S**hape **V**ector **Q**uantization
HR	**H**eart **R**ate
IDNN	**I**nput-**D**elay **N**eural **N**etwork
IoT	**I**nternet **o**f **T**hings
LBG	**L**inde-**B**uzo-**G**ray
LTC	**L**ightweight **T**emporal **C**ompression

LTE	**L**ong **T**erm **E**volution
MM	**M**athematical **M**orphology
MSE	**M**ean **S**quare **E**rror
MSVQ	**M**ean-**S**hape **V**ector **Q**uantization
NN	**N**eural **N**etworks
NNC	**N**earest **N**eighbor **C**ondition
PCA	**P**rincipal **C**omponent **A**nalysis
PLA	**P**iecewise **L**inear **A**pproximation
PPG	**P**hoto**P**lethysmo**G**ram(**G**rphy
PRD	**P**ercentage **R**oot mean square **D**ifference
RMSE	**R**oot **M**ean **S**quare **E**rror
RNS	**R**andom **N**ode **S**election
SOM	**S**elf **O**rganizing **M**aps
SOMP	**S**imultaneous **O**rthogonal **M**atching **P**ursuit
SURF	**S**ubject-adpative **U**nsupervised ecg compressor for wea**R**able **F**itness
TASOM	**T**ime-**A**daptive **S**elf **O**rganizing **M**aps
OD	**O**nline **D**ictionary
UMTS	**U**niversal **M**obile **T**elecommunications **S**ystem
VQ	**V**ector **Q**uantization
WiFi	**Wi**reless **Fi**delity
WSN	**W**ireless **S**ensor **N**etwork

Chapter 1

Introduction

1.1 A short introduction to IoT

In the last decade, a very large number of Internet of Things (IoT) technologies have emerged. These, have enabled the gathering and processing of data acquired through a variety of sensors, operating in many different fields (e.g., environmental, mobility-related, medical, etc.). The sheer amount of data collected from IoT systems calls for the development of applications that enable its organization, analysis and presentation, thus allowing for substantial improvements in human activities through intelligent data-driven systems.

Since its conception, this concept has grown and many technical solutions are now available for data gathering, data protection and analysis. IoT systems usually entail the use of a very large number of sensor devices with sensing and communication capabilities, which collect and transmit data to a so called *sink* node (i.e., an IoT gateway or a federation thereof), which collects data for further elaboration and transmission to Web-based servers and applications. According to the IoT paradigm, each everyday object can be possibly enhanced with a communication interface and some minimal sensing capabilities, which turn it into a "smart" object.

Nevertheless, IoT devices are often constrained in terms of computational power and energy. In fact, they are often powered with small batteries and their size can be very small (e.g., as

small as 2×2 mm^2). So, the energy they drain to carry out sensing and transmission plays an important role in their lifetime. Moreover, they have limited memory and processing capabilities. Thus far, there a lot of research work has appeared on IoT and in particular on data compression. However, previous performance assessments were mainly carried out considering quality of compression and reconstruction, whilst the energy consumption aspect was often neglected. This dissertation focuses on these subjects for several types of IoT devices. We stress and show that energy should be sparingly used by the software running on such devices, as they are often battery operated and, in turn, reduction in their energy consumption is a key consideration.

In the first part of this thesis, Chapters 2 and 3, we consider environmental wireless sensor networks and the temporal or spatio-temporal compression of the signal they acquire. In the second part, Chapters 4–6, we focus on lossy compression techniques for body sensor networks which gather one dimensional biosignals, such as electrocardiogram, respiration, etc.

The structure of this doctoral thesis is briefly described in the next two sections.

1.2 Environmental Wireless Sensor Networks

Wireless Sensor Network (WSN) are a major part of IoT. Lossy compression is a key functionality for them as it allows saving transmission power and prolonging the network lifetime. Moreover, for many applications some loss in the data representation accuracy is tolerable. On this matter, we underscore that a solid understanding of the suitability of different lossy compression techniques, along with their comparative performance evaluation, are still open research problems.

In Chapter 2, we consider the temporal and spatial correlations that are inherent to environmental WSN signals. WSNs are characterized by the dense deployment of sensor nodes that continuously observe some physical phenomenon. In practical scenarios, sensor observations are highly correlated in space. Furthermore, the nature of the physical phenomenon that

is being measured also implies some non-negligible temporal correlation. These facts, along with the collaborative nature of WSNs bring opportunities for the development of efficient compression/communication protocols. In this thesis, several key elements are investigated to capture and exploit the spatio-temporal correlation in WSNs for the design of efficient data gathering protocols. We focus on the spatio-temporal compression of readings from the sensor nodes. Our objective, for each data gathering round, is to collect the WSN readings from a small percentage of nodes (sparse sampling), while being able to recover with very good accuracy the entire dataset at the data collection point (the sink node). To do so, the spatio-temporal features of environmental WSN signals are exploited within suitable data compression tools including: Lightweight Temporal Compression (LTC), Distributed Source Coding (DSC), Discrete Fourier Transform (DCT) and Compressive Sensing (CS). The performance of these algorithms is compared considering realistic datasets and research directions are identified.

In Chapter 3, we improve the CS-based compression scheme of Chapter 2. In detail, the signal representation step of CS-based algorithms is ehnanced considering the concept of spatial covariograms and using it to refine the (estimated) covariance matrix at the data collector. Our covariogram-based estimation technique is executed online by estimating the signal statistics in space and time. As a result, the reconstruction error is substantially reduced, leading to more accurate representations of the signal. As an additional contribution, a novel spatial sampling strategy is presented to implement a more effective selection of the (subset of) nodes that transmit (i.e., sample the input signal) at each data collection round. The impact of covariogram estimation and sparse sampling is numerically assessed through a simulation tool developed in Matlab considering synthetic as well as real environmental signals.

1.3 Wearable Devices

Wearables technology has become more and more pervasive in the last few years. As noted in [3], the IoT paradigm has recently shifted from a scenario where sensors are solely placed

around humans (i.e., integrated within the surrounding environment) to one in which we, as humans, carry ourselves the sensor devices and actively participate in the sensing process.

In Chapters 4 and 5, we consider temporal correlation of biosignals acquired by wearable wireless monitors. These devices are becoming a natural and economic means to gather vital signs from end users, such as electrocardiogram, blood-pressure, blood-saturation (SpO2), body movement, respiration, etc., and own an immense potential for applications such as continuous monitoring for personalized healthcare or fitness applications. Wearables are however heavily constrained in terms of onboard memory, transmission capability and energy reserve, and this calls for dedicated, lightweight but still effective algorithms for data management. Lossy data compression techniques, whose aim is to minimize the amount of information that is to be stored on the onboard memory and subsequently transmitted over wireless interfaces, is the means that we investigate in this thesis to increase the data management and transmission efficiency.

Specifically, in Chapter 4 we analyze selected compression techniques for biometric signals, quantifying their complexity (energy consumption) and compression performance. Hence, we propose a new class of online dictionary compression algorithms, designed to be energy efficient, online and amenable to any type of signal exhibiting recurrent patterns (referred to as *quasi-periodic* signal).

In Chapter 5, we continue our analysis of compression algorithms for quasi-periodic biosignals by reviewing the most promising algorithms from the literature and carrying out a systematic performance evaluation of their complexity (energy cost, including processing and transmission), compression efficiency and reconstruction accuracy. One dimensional biosignals such as ECG, RESP and PPG, are considered as they are often available in commercial wearable IoT devices, using signals from public databases as well as signals that we gathered from a wireless Zephyr Bioharness 3 heart rate monitor. As we quantify, the most efficient schemes allow reductions in the signal size of up to 100 times, which entail similar reductions in their energy demand (with subsequent increase in their lifetime), by still keeping the reconstruction error within 4% of the peak-to-peak signal amplitude.

Based on the lessons we learned from the material of Chapters 4 and 5, Online Dictionary, due to its excessive memory demand, is not yet amenable to a practical implementation, it reveals some important performance tradeoffs that are considered for our more advanced compression algorithm of Chapter 6.

In Chapter 6 we propose an original *subject-specific* dictionary-based compression scheme for wearable wireless monitors. Suitable dictionaries are built and maintained in an online and unsupervised fashion to faithfully capture and represent input signal patterns. These dictionaries are then exploited in the compression of such signal through the transmission of indices in place of the entire codewords (each codeword represents a typical pattern in the input signal). Neural networks such as the Time Adaptive Self Organizing Map (TASOM) and the Growing Neural Gas (GNG) are utilized for the dictionary construction and maintenance. Our final algorithm is called SURF "Unsupervised ecg compressor for weaRable Fitness monitors" and uses GNG neural networks. The TASOM-based compressor also provides good results, i.e., reconstruction errors (RMSE) with only 9 neurons remain within 6% of the peak-to-peak signal amplitude while the signal size to be transmitted is reduced between 60 to 70 times (compression ratio). This amount of reduction can also be reached with small memory requirements. However, the TASOM has a static structure (i.e., the number of neurons is preset and fixed at all times), and this limits its performance in terms of reconstruction fidelity for rare sequences and its capability to adapt and explore new portions of the signal space as the device is worn by a new subject, new activities are carried out or rare events occur (which may be the symptoms of a disease and therefore, although rare, have to be properly tracked). Our final design (SURF) retains all the advantages of the TASOM-based compressor and also successfully deals with these aspects. Its reconstruction error is lowered to 4% and its compression is as high as 96-fold, outperforming all the previous (adaptive) approaches from the literature.

Finally, in Chapter 7 we summarize our main findings.

Chapter 2

Environmental WSN – a taxonomy and performance comparison of existing compression techniques

2.1 Introduction

Lossy compression is a key functionality for a Wireless Sensor Network (WSN) as it allows saving transmission power and prolonging the network lifetime. Moreover, for many applications some loss in the data representation accuracy is tolerable, think for example of environmental slow-varying data fields [4].

On this matter, we underscore that a solid understanding of the suitability of different lossy compression techniques, along with their comparative performance evaluation, are still open research problems. The objective of this chapter is to fill this gap, by providing a thorough quantitative analysis of lossy compression algorithms and, at the same time, proposing some improvements that work well when data is correlated in the spatial and temporal dimensions. Toward this end, we consider temporal compression schemes based on linear approximations and Fourier transforms, whereas spatial and/or temporal dynamics are exploited through

compression algorithms based on distributed source coding (DSC) [5] and several schemes based on compressive sensing (CS) [6].

Temporal lossy compression schemes for WSNs have been evaluated in [7], where the authors studied the energy-vs-compression tradeoffs when the sensor nodes independently compress their readings, by only exploiting the temporal correlation (TC) in the sampled signal. The best algorithms that have been identified in [7] are Discrete Cosine Transform (DCT) and Lightweight Temporal Compression (LTC) [8], with the former providing the best compression performance and the latter being the most energy efficient in terms of local computations at the sensors. The spatial correlation (SC) in the measured data is exploited by Distributed Source Coding (DSC). Practical DSC methods were first proposed in [9]. Briefly, compression is achieved by splitting the input data values according to so called *bins* and, for each reading, the bin identifier (`binID`) is sent in place of the actual data point. The rationale is that the `binID` needs fewer bits than the original data. Moreover, if the bins are properly designed, i.e., they have certain distance properties, reconstruction at the receiver is possible through the acquisition of some side information, i.e., an uncompressed (and spatially correlated) data sequence that is sent from another sensor. Data compression schemes belonging to this class will be considered and evaluated here. Recent advancements in the signal processing field have led to the development of the theory of Compressive Sensing (CS) [6, 10, 11]. Consider a network of sensors, where we are interested in collecting a vector of measurements (one reading per node) at a given time. CS-based compression relies on the idea that sampling from a small number of sensor nodes, which send their uncompressed readings, is sufficient to recover the whole vector of readings with good accuracy. CS is based on the premise that the data has a sparse representation in some domain, which means that in that domain it can be equivalently described by a small number of non-zero coefficients. More details on CS is provided in Section 2.5. This concept can be used to concoct lossy compression schemes that minimize the number of samples acquired from the sensor nodes, see [12].

Contribution: in this chapter we consider lossy compression schemes that rely on different

techniques, such as the exploitation of the temporal (DCT, LTC) and spatial (DSC) dynamics of the signal as well as recent CS schemes, that utilize the signal correlation in space and time. Notably, despite the large amount of research available on theoretical aspects, only recently (DISCUS has been the first practical approach to DSC) have researchers started to look at practical distributed compression techniques. Most importantly, approaches belonging to different fields such as signal processing (CS, DCT) or information theory (DSC) and networking (LTC) are seldom evaluated against one another. The aim of this chapter is to shed some light on the comparative performance of these algorithms.

To this end, the aforementioned schemes are adapted and integrated with practical aggregation and data gathering strategies under realistic WSN settings. Thus, they are compared in terms of their compression and energy consumption performance by varying relevant signal statistics and network parameters. The lessons that we have learned from this study are multifold. First, the best performing scheme depends on the signal statistics. CS-based compression is preferable across the entire range of temporal correlations when the spatial correlation is moderate. As the spatial correlation becomes high, DSC schemes perform best and when the temporal correlation is high, TC-based algorithms (DCT, LTC) perform almost as well as CS, providing a reasonable alternative to the latter. Notably, CS-based schemes show a non-negligible gap between their theoretical performance (a lower bound computed assuming perfect statistical knowledge of the signal at the sink) and the actual one. We believe this gap can be reduced and we identify how that can be done as part of a future research work.

Organization of this chapter: in Section 2.2, we start by describing the system model; this includes details on the network topology, the data gathering (routing) structure, the clustering procedure that we use for DSC, the energy consumption and the signal models (for synthetic and real data traces). In Section 2.3 we briefly review two temporal compression algorithms taken from the literature. In Section 2.4 and Section 2.5 we respectively describe compression schemes based on distributed source coding and the baseline compressive sensing algorithms. Our node selection strategies are presented in Section 2.6. We summarize our main findings and highlight future research directions.

2.2 System Model

2.2.1 Signal model

In this subsection, we first outline the adopted synthetic signal model. After that, we describe the environmental signals that are used at the end of the performance evaluation section.

Synthetic signals: we consider the method of [13], that has been validated against real signals and allows the generation of time-varying spatial fields with tunable correlation characteristics. This is especially useful to control the degree of correlation in space and time and numerically assess its impact on the performance of the selected compression schemes. Here, we target applications that sample the sensor field at regular intervals. Thus, time $t = 0, 1, 2, \ldots$ is slotted and the slot duration is fixed and equal to Δt. We denote the spatial and temporal domains by $\mathcal{D} = [-x_D, x_D] \times [-y_D, y_D]$ and by $\mathcal{T} = i\Delta t$, $i = 0, 1, 2, \ldots$, respectively. A point in space is indicated with $\boldsymbol{p} = (x, y) \in \mathcal{D}$. With $x(\boldsymbol{p}, t) : \mathcal{D} \times \mathcal{T} \to \mathbb{R}$ we indicate the generated signal, which is stationary in space and time, with mean $\mu_x = 0$, variance $\sigma_x^2 = 1$ and tunable correlation structure.[1]

As commonly assumed in the modeling of environmental spatio-temporal signals [13, 14], the correlation function is assumed to be separable in the space and time components:

$$\rho(\boldsymbol{p}_1, t_1, \boldsymbol{p}_2, t_2) = \rho_S(\boldsymbol{p}_1, \boldsymbol{p}_2)\rho_T(t_1, t_2) \ , \tag{2.1}$$

where $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are two points in $\mathcal{D}$, t_1 and $t_2 \geq t_1$ are two time instants, $\rho_S(\cdot)$ and $\rho_T(\cdot)$ respectively denote the spatial and temporal correlation functions. From the signal stationarity, (2.1) can be equivalently rewritten as a function of $d = \|\boldsymbol{p}_1 - \boldsymbol{p}_2\|_2$ and $\Delta t = t_2 - t_1$.

[1] Note that we can obtain any other values for the mean and variance by shifting and scaling the random field without affecting the correlation characteristics.

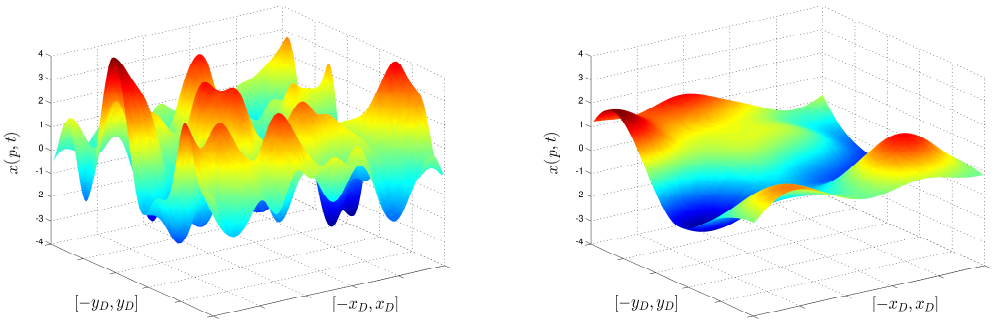


FIGURE 2.1: Signal example for $\gamma = 0.1$ (left, low spatial correlation) and $\gamma = 1$ (right, high spatial correlation).

Any suitable correlation function [15] can be used in the signal generation procedure. In our researches, for $\rho_S(d)$ we use a Gaussian function:

$$\rho_S(d) = \exp\left\{-\frac{d^2}{\gamma\alpha^2}\right\}, \tag{2.2}$$

where α is a scaling factor that depends on the field size and γ is a free parameter used to control the spatial correlation. In Fig. 2.1 we show a single time instant for two synthetic signals generated with this model, obtained with $\gamma \in \{0.1, 1\}$. For the temporal correlation, we use $\rho_T(\Delta t) = \rho \in [0, 1]$, referred to as the temporal correlation coefficient (this amounts to assuming an exponential correlation function).

Real signals: we consider the database from [2], which contains Global Historical Climatology Network and Legates and Willmott's meteorological station records of air temperature and precipitation. For our results, we used data from the first 100 measurement stations in the provided dataset for a time period ranging between 1950 and 1996.

2.2.2 Topology and Data Gathering

We consider a set $\mathcal{N}$ of sensor nodes with $|\mathcal{N}| = N$, deployed uniformly at random in the area $\mathcal{D}$, with $x_D = y_D = 50$ m. For the transmission range R of the nodes we adopt a unit disk model so that sensors can only communicate with those nodes placed at a distance shorter than or equal to R. We use $R = 2x_D\sqrt{5}/\sqrt{N}$ to guarantee that the structure is connected with high probability under any deployment. The sink is placed in the center of the WSN area. For the routing, each sensor considers as its next hop the node within its

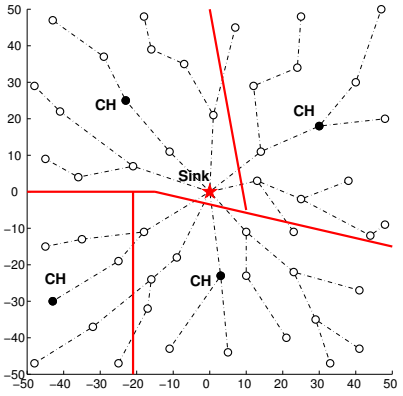


FIGURE 2.2: Network deployment and clustering example.

range that provides the largest advancement toward the sink (geographical forwarding), in Fig. 2.2 we show an example topology for $N = 50$ nodes.

For the data gathering, we assume that the monitoring application is delay-tolerant. Time $t = 0, 1, 2, \ldots$ is slotted. We say that a sensor "samples" at a certain time slot if it takes a reading from the onboard sensing device. Note that sensors do not necessarily sample in all time slots but suitable scheduling procedures, discussed in Section 2.6, are utilized to keep the number of sampling nodes per time slot low. Every T time slots (data collection round) the sink collects compressed data from the sensors. This is implemented to allow for the aggregation of the readings taken within a time window of T time slots into the payload of a single or multiple packets. As will be shown in Section 2.7, this effectively mitigates the overhead arising from the transmission of packet headers.

2.2.3 Clustering

For the DSC technique we have used the weighted clustering algorithm (WCA) of [16]. This makes it possible to group the sensors into a predetermined number N_c of clusters obtained so as to evenly divide the WSN area. Within each cluster, a clusterhead (CH) is elected, so that the CH will minimize the average distance with respect to the non-CH nodes within its cluster.

2.2.4 Energy Model

For every compression technique, we have counted the number of additions, multiplications, divisions and comparisons executed at the nodes. Thus, according to the considered sensor hardware, we have translated these figures into the related number of clock cycles and we have subsequently turned the latter into the corresponding energy expenditure. In addition, we have considered the energy consumption associated with the transmission and reception of each packet. In this thesis, for these figures we have considered a MSP 430 micro-processor and a CC2420 radio [7] from Texas Instruments.

2.3 Temporal Compression Techniques

For this class of algorithms we consider LTC and DCT, as these are the best performing algorithms from [7]. With them, each sensor node independently compresses a time series, exploiting the temporal correlation of the signal. Both algorithms work on time series of T subsequent (scalar) readings $\{x_1, x_2, \ldots, x_T\}$.

LTC: LTC is a low complexity algorithm that uses a linear model to approximate a time series, according to a preset error tolerance $\varepsilon \geq 0$. The algorithm works by approximating multiple readings through a single segment, so that the segment will be within the given error tolerance for all points, see [7]. This algorithm has a linear complexity in the number of readings in the time series. Also, a segment is described by four coefficients, so compression is achieved when the number of readings that are covered through it (that depends on ε) is larger than 4. It follows that LTC is effective when the signal exhibits a high temporal correlation, whereas in the case of uncorrelated readings its performance is worse than sending the time series uncompressed (as confirmed by the results of Section 2.7).

DCT: as observed in [7], DCT has a good energy compaction property, which means that the energy of the signal tends to be distributed within the first few DCT coefficients. Based on this, we implemented DCT by taking the time series $\{x_1, x_2, \ldots, x_T\}$, computing its DCT and retaining the first T' coefficients, where $T' \leq T$ determines the level of compression

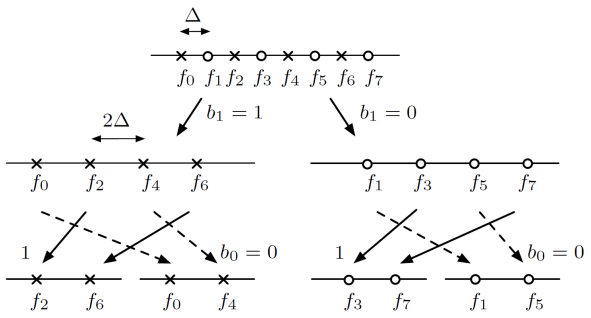


FIGURE 2.3: Graphical representation of the binning procedure using the Ungerboeck tree-based scheme [1]. In this example, the alphabet $\mathcal{F}$ contains 8 symbols $\{f_0, \ldots, f_7\}$, which are classified into bins according to (b_1, b_0) ($b_i \in \{0, 1\}$). For instance, $b_1 = 0$, $b_0 = 0$ identifies the bin in the bottom right of the figure (containing symbols f_1 and f_5). This mapping assures that the distance of the symbols contained in the same bin is maximized.

and, at the same time, the representation accuracy. In our implementation, also for DCT we considered a maximum error tolerance ε (see also [7]). This implies that the compressor has to evaluate the representation accuracy of the compressed sequence, adding additional coefficients until the tolerance ε is met.

Compression strategy: the compression is orchestrated according to cycles of T time slots. Each sensor independently collects a time series $\{x_1, x_2, \ldots\}$ and, when T readings are available, these are transformed according to the compression strategy in use (either DCT or LTC). The resulting coefficients are stored in the payload of a single(multiple) packet(s) that is(are) sent to the sink and the process is repeated.[2]

2.4 Spatial Compression Algorithms

The distributed source coding (DSC) algorithms that we implement in this research only exploit the spatial correlation of the signal. As suggested in [9] we use the Ungerboeck tree-based binning scheme [1], see Fig. 2.3 for an example. We consider the transmission of an analog reading, which is first quantized (uniform quantization is assumed in Fig. 2.3, with quantization step Δ) into a certain number of levels N_ℓ ($N_\ell = 8$ in the figure). We refer to n as the number of bits required to represent the quantized symbol. After this,

[2]Multiple packets may be needed to carry the compressed signal, depending on T, the compression process and the payload size.

bins are obtained by grouping the quantized symbol as shown in Fig. 2.3. In this figure, we use four bins $(B_0, \ldots, B_3)$, which means that $k = 2$ bits (b_1, b_0) are sufficient to represent the `binID`. Note that following this approach the symbols within each bin have maximum distance, whilst assuring an even distribution of the symbols in the bins. Also, the distance property is maintained for all bins. This procedure can be applied for any n and k.

Compression strategy: given the above splitting procedure, the DSC scheme works as follows. As said above, the WSN is divided into a number N_c of clusters. At each time slot t, each non-CH node in the cluster compresses its reading $x_t \in \mathbb{R}$ by first quantizing it into x_t^q according to Δ and N_ℓ.[3] Subsequently, the quantized symbol x_t^q is mapped into the corresponding bin according to the Ungerboeck procedure, using $k < n$ bits. At time t, the resulting `binID`, indicated by s_t (k bits), is stored at the sensor node in place of the quantized symbol x_t^q. The CH simply quantizes its current reading y_t and stores in its internal memory its quantized representation y_t^q (n bits). Every T time slots the memory of both non-CH and CH nodes is flushed and the data therein is sent to the sink. In detail, CH nodes store the last T quantized readings in a single packet (payload of nT bits), which is sent to the sink via multi-hop routing. Non-CH nodes store the last T compressed readings (the payload is now kT bits) into a single packet and send it to the sink via multi-hop routing.

For the procedure to be executed at the sink, let us consider a generic cluster. The decompressor first looks at the packet received from the corresponding CH, which contains T uncompressed readings y_t^q (side information). Consider now a non-CH node i in this cluster. For each `binID` s_t contained in the packet from this node, with $t \in \{1, \ldots, T\}$, we refer to B_{s_t} as the corresponding bin and the estimated (quantized) symbol for node i is obtained as:

$$\hat{x}_t^q = \text{argmin}_{x^q \in B_{s_t}} \left| x^q - y_t^q \right|, \tag{2.3}$$

where we look for the element in the bin B_{s_t} that has minimum Euclidean distance with respect to the side information y_t that is received from the CH.

[3] Quantization is performed according to a minimum distance criterion, as in [9].

From the bin construction procedure, it descends that $\hat{x}_t^q$ will be equal to the original reading x_t^q if the difference between x_t^q and the side information y_t^q is strictly smaller than $2^{k-2}\Delta$. Note that this depends on the (spatial) correlation properties of the signal. Also, we consider the quantization error as negligible; note that this is a good assumption in the considered scenario.[4] Also, it is possible to improve the performance of the above procedure through the use of coding for the generation of `binID` sequences (LDPC being a popular example). In this dissertation, we consider the memoryless procedure outlined above, noting that it is often preferred for WSNs due to its lightweight character, see, e.g., [17]. Extension to more complex (and burdensome) coding approaches, along with the evaluation of their additional complexity, are left as a future work.

2.5 Spatio Temporal Compression Algorithms

Within this class, we consider the compressive sensing algorithm. There, the covariance structure of the data is estimated on the fly and is exploited to reconstruct the entire sensor field, although sampling is performed from a subset of the nodes. Specifically, let $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]^\dagger \in \mathbb{R}^N$ be the signal at the N sensors at the generic time t.[5] Our aim is to have $M \ll N$ sensors sampling the signal and subsequently sending their readings to the sink. Suitable strategies for the selection of the M sensors are presented in Section 2.6.

Note that for compressive sensing we decouple the node selection procedure (dictating which nodes sample at each time slot) from the signal reconstruction algorithm that we use at the sink. Also, we note that transmitting at each time step is inefficient as the overhead due to packet headers dominates over the payload size. To overcome this, in Section 2.6 we present strategies that allow data aggregation from multiple time slots and the transmission of aggregated packets at a slower rate.

In this section we detail the data compression and reconstruction procedures involved in compressive sensing.

[4] We consider $n = 16$ bits per reading, with a signal range of ≈ 2 units. This leads to $\Delta = 3 \cdot 10^{-5}$, which is negligible compared to the reconstruction error, see Section 2.7.

[5] The notation $\boldsymbol{x}^\dagger$ indicates the transpose of vector $\boldsymbol{x}$.

2.5.1 CS-based Compression

As stated in the introduction, CS is a mathematical framework that makes it possible to retrieve the original signal $\boldsymbol{x}$ from a subset of the sensors' readings. In our problem, $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]^\dagger \in \mathbb{R}^N$, where the i-th element x_i represents the reading generated by sensor node i. Including the temporal dimension, we indicate the complete WSN signal at time t by $\boldsymbol{x}(t)$ (with the term "complete," we mean containing the readings from all the sensor nodes).

At any time t, CS can be used to approximate $\boldsymbol{x}(t)$ by using a subset of the readings, i.e., collecting a smaller vector $\boldsymbol{y}(t)$ of size $M \ll N$ [12]. Here, we assume that the signal $\boldsymbol{x}(t)$ is sparse in some domain and zero mean,[6] which means that there exists an invertible $N \times N$ transformation matrix $\boldsymbol{\Phi}(t)$ such that:

$$\boldsymbol{x}(t) = \boldsymbol{\Phi}(t)\boldsymbol{u}(t)\,, \tag{2.4}$$

where $\boldsymbol{u}(t) \in \mathbb{R}^N$ is M-sparse.[7] Assuming that $\boldsymbol{\Phi}(t)$ is known, $\boldsymbol{x}(t)$ can be retrieved from its sparse representation using (2.4).

Compression strategy: for illustration purposes, assume that at each time slot t the sink collects the readings from $M \ll N$ sensors, storing them into vector $\boldsymbol{y}(t)$.[8] Note that this is inefficient in terms of data collection overhead. To ameliorate this, practical sensor selection procedures, that promote data aggregation, are presented in Section 2.6.

We can write $\boldsymbol{y}(t) = \boldsymbol{R}(t)\boldsymbol{x}(t) + \boldsymbol{n}$, where $\boldsymbol{R}(t)$ is a sampling matrix of size $M \times N$ and $\boldsymbol{n} \in \mathbb{R}^N$ is the measurement noise (if any). $\boldsymbol{R}(t)$ is an all zero matrix, except for a single one in each row and at most a single one in each column, which means that $\boldsymbol{y}(t)$ is a sampled

[6] This assumption is not restrictive as the approach explained here can be applied to the signal after subtracting its (estimated) mean $\overline{\boldsymbol{x}}(t)$ [12].

[7] A vector is defined M-sparse when it has only M significant elements, whereas the amplitude of the remaining $N - M$ elements is negligible.

[8] Note that M can be varied across subsequent time slots.

version of $\boldsymbol{x}(t)$.[9] Using the previous definitions, we have:

$$\boldsymbol{y}(t) = \boldsymbol{R}(t)\boldsymbol{x}(t) + \boldsymbol{n} = \boldsymbol{R}(t)\boldsymbol{\Phi}(t)\boldsymbol{u}(t) + \boldsymbol{n} \stackrel{\text{def}}{=} \boldsymbol{\Psi}(t)\boldsymbol{u}(t) + \boldsymbol{n}\,, \tag{2.5}$$

which can be solved for $\boldsymbol{u}(t)$, once the sparsification matrix $\boldsymbol{\Phi}(t)$ (see Section 2.5.2) and the sampling matrix $\boldsymbol{R}(t)$ are known. Specifically, when the signal is sparse enough, it can be accurately recovered from its compressive measurement $\boldsymbol{y}$ by solving the quadratic program:

$$\hat{\boldsymbol{u}} = \text{argmin}\, \|\boldsymbol{u}\|_1 \text{ s.t. } \|\boldsymbol{y} - \boldsymbol{\Psi}\boldsymbol{u}\|_2 \leq \epsilon\,, \tag{2.6}$$

where ϵ is an upper bound on the ℓ_2 norm of the noise and $\|\cdot\|_1$ is the ℓ_1 norm. In the literature, many efficient solvers for this type of systems have been proposed, see, e.g., ℓ_1-MAGIC [18], Subspace Pursuit [19] and NESTA [20]. Once $\hat{\boldsymbol{u}}$ is computed, (2.4) is used to retrieve the real signal $\boldsymbol{x}(t)$. Note that the signal that we obtain through this procedure, referred to as $\hat{\boldsymbol{x}}(t)$, is an interpolated version of $\boldsymbol{x}(t)$.

2.5.2 Sparsification Basis

As demonstrated in [21], although standard transforms such as the Fast Fourier Transform (FFT), Wavelet or the DCT perform satisfactorily as sparsification bases for video sequences, they are rather ineffective for typical WSN signals. A solution to this entails the use of Principal Component Analysis (PCA) [22]. PCA is a statistical processing technique that uses the sample covariance matrix $\boldsymbol{\Sigma}$ to convert a correlated signal into a number of principal components that is usually smaller than the number of variables in the original signal $\boldsymbol{x}$. That is, PCA finds a smaller space where $\boldsymbol{x}$ can be projected by minimizing the loss of information that occurs from the original (size N) to the projected (size M) space. The projection basis is obtained by the M eigenvectors of the sample covariance matrix that better represent the signal (i.e., that capture most of its energy). Note that the projection basis corresponds to the sparsification basis $\boldsymbol{\Phi}(t)$ that we are looking for. A simple

[9] In the literature, this sampling is referred to as *transform coding*.

procedure to estimate matrix $\boldsymbol{\Phi}(t)$ is [12]:

Algorithm 1:

1. Use $\boldsymbol{y}(t)$ and the current $\boldsymbol{\Psi}(t)$ matrix to obtain $\boldsymbol{y}(t) \rightarrow \hat{\boldsymbol{x}}(t)$ (through (2.5), solved via ℓ_1-minimization).
2. Store $\hat{\boldsymbol{x}}(t)$ into a buffer, containing the T most recent (estimated) vectors $\mathcal{X} = \{\hat{\boldsymbol{x}}(t), \hat{\boldsymbol{x}}(t-1), \ldots, \hat{\boldsymbol{x}}(t-T+1)\}$.
3. Estimate $\boldsymbol{\Sigma}(t)$ (and $\overline{\boldsymbol{x}}(t)$) from the vectors in $\mathcal{X}$.
4. Compute the new $\boldsymbol{\Phi}(t+1)$ from $\boldsymbol{\Sigma}(t)$ using PCA.
5. Obtain $\boldsymbol{\Psi}(t+1) = \boldsymbol{R}(t+1)\boldsymbol{\Phi}(t+1)$.

In the next chapter, we present an original method to obtain an improved robust estimates of $\boldsymbol{\Phi}(t)$, that also works well when signals are non-stationary.

2.6 Sampling Strategies

In this section, we present random and deterministic sampling strategies to obtain matrix $\boldsymbol{R}(t)$, whose aim is to equally share the workload among the nodes, while sampling from the sensors that provide the highest benefit in terms of signal reconstruction performance at the sink. The sampling schemes are numerically evaluated at the end of the section.

2.6.1 Random node selection (RNS)

This technique has been proposed in [12]. At each time slot $t = 0, 1, 2, \ldots$, on average, a number M of nodes transmit their readings to the sink. A simple and distributed technique to achieve this is to define a transmission probability p_{tx} so that $M = Np_{\text{tx}}$. Hence, at time

t, each node independently decides whether it will be sampling and transmitting to the sink according to p_{tx}; in that case, its current reading is stored in the payload of a single packet, which is sent to the sink through multi-hop routing. If user $i \in \{1, \ldots, N\}$ transmits at time t, $\boldsymbol{R}(t)$ will have a (single) one in the i-th column.

2.6.2 Deterministic node selection (DNS)

Below, we present two node selection strategies to sample the incomplete data $\boldsymbol{y}$ at each collection round. Both approaches work in data collection rounds of T time slots. A predetermined and deterministic sampling strategy is computed by the sink and disseminated to all WSN nodes. In detail, exactly M nodes sample the signal at each time slot $t \in \{1, 2, \ldots, T\}$ and these nodes are selected according to one of the two selection strategies that we present below. The selected nodes over the T time slots define a so called *monitoring schedule*.

Thus, at time $t \in \{1, \ldots, T\}$, $\boldsymbol{R}(t)$ has a one in column $i \in \{1, \ldots, N\}$ if the monitoring schedule includes node i for transmission in this time slot. To improve the efficiency in the transmission of the sampled readings, for DNS we use the following *data aggregation* approach. During each collection round (time slots from 1 to T), each sensor will be sampling the signal in the time slots dictated by the monitoring schedule. In the last time slot T, the sensor will aggregate its readings storing them in the payload of a single (or multiple) packet(s), which is(are) sent to the sink for processing. The sink collects the incoming packets and, given its knowledge of the sampling matrix $\boldsymbol{R}(t)$, associates back the readings from the nodes to the corresponding time slots. This permits to obtain $\boldsymbol{y}(t)$ at the sink for all time slots $\{1, 2, \ldots, T\}$. This process is repeated for each collection round.

Next, we discuss two heuristic procedures for the computation of the monitoring schedule. These algorithms are based on the spatial correlation among sensors. The idea is to pick one node at a time and assign it to the monitoring schedule. In doing this, we seek for the node that brings the largest improvement in terms of reconstruction quality at the sink. To come up with fast and still tractable models, we assume that the underlying signal is Gaussian distributed. The algorithms will be then applied, in Section 2.7, to a general setup

and shown to be effective in the presence of non-Gaussian statistics as well.

Correlation based deterministic node selection (CB-DNS): to simplify our explanation, we assume that $T = N/M$ and consider the first collection round, i.e., time slots $\{1, 2, \ldots, T\}$.

To start with, let X and Y be two jointly Gaussian random variables, where $X \in N(\mu_X, \sigma_X)$ and $Y \in N(\mu_Y, \sigma_Y)$ (with μ and σ we respectively indicate mean and standard deviation). In this case, as shown, e.g., in Chapter 11 of [23], the conditional variance of Y, given that we have sampled X is constant and equal to $Var[Y|X = x] = \sigma_Y^2(1-\rho_{XY}^2) = \sigma_Y^2 - (\sigma_{XY}/\sigma_X)^2$, $\forall\, x$, where $\rho_{XY} = \sigma_{XY}/(\sigma_X\sigma_Y)$ is the correlation between X and Y and $\sigma_{XY} = E[(X - \mu_X)(Y - \mu_Y)]$. Generalizing this concept to all nodes in $\mathcal{N}$, for each $j \in \mathcal{N}$ we can compute its overall conditional variance m_j with respect to all the other sensors, that is:

$$m_j = \sum_{i \in \mathcal{N}} \sigma_i^2(1 - \rho_{ij}^2) = \sum_{i \in \mathcal{N}} \sigma_i^2 - \sum_{i \in \mathcal{N}} \frac{\sigma_{ij}^2}{\sigma_j^2}\,. \tag{2.7}$$

Note that the node $j^\star$ with the smallest m_j is found as:

$$\begin{aligned} j^\star &= \text{argmin}_{j \in \mathcal{N}} \left(\sum_{i \in \mathcal{N}} \sigma_i^2 - \sum_{i \in \mathcal{N}} \frac{\sigma_{ij}^2}{\sigma_j^2} \right) \\ &= \text{argmax}_{j \in \mathcal{N}} \left(\sum_{i \in \mathcal{N}} \frac{\sigma_{ij}^2}{\sigma_j^2} \right), \end{aligned} \tag{2.8}$$

where the second equality follows as the first sum in the first line does not depend on the index j. Given this, we can equivalently use the following metric m'_j:

$$m'_j = \left(\sum_{i \in \mathcal{N}} \frac{\sigma_{ij}^2}{\sigma_j^2} \right). \tag{2.9}$$

Note that m'_j is related to the amount of correlation between the value sampled at sensor j and the readings from all the other sensors. In a sense, m'_j tells us how suitable sensor j is to represent the other nodes in set $\mathcal{N}$.

The problem under consideration amounts to the selection of T disjoint sets containing M nodes each. Each set will be then associated with a single time slot in the collection round and the nodes therein will be sampling in that time slot. For CB-DNS we sort all the sensors in $\mathcal{N}$ in descending order of m'_j. Thus, we pick the T sensors with the highest metric in the ordered set and assign them to time slots $\{1, \ldots, T\}$ so that the best node will be assigned to time slot 1, the second best to time slot 2 and so on. Hence, we pick the second best T nodes and assign them to time slots $\{1, \ldots, T\}$ following the same approach. The process is iterated until all nodes are assigned. Each node will then sample in the assigned time slot.

The procedure can be easily generalized to the case where $T = \kappa N/M$ with κ and N/M integers. In that case, the monitoring schedule obtained from the previous assignment scheme is repeated κ times within the data collection round. When N/M is not an integer, some nodes will have to sample twice within $\lceil N/M \rceil$ time slots to ensure that there are M sampling nodes. The nodes that resample can be rotated to ensure fairness in their energy consumption.

Enhanced correlation based deterministic node selection (ECB-DNS): before detailing the full algorithm, we consider the simpler problem of selecting M sensors from $\mathcal{N}$ when $T = 1$, i.e., there is a single time slot per data collection round. In this case, we start initializing the set of nodes to be processed $\mathcal{S}_1 = \mathcal{N}$ and an empty set $\mathcal{S}_2$. Our objective is to select M nodes from $\mathcal{S}_1$ so that these will lead to the smallest reconstruction error for the sensors that are not selected. The algorithm is presented next.

Algorithm 2:

1. Consider the n-th step, with $n \in \{1, \ldots, M\}$, and let $\mathcal{S}_1(n)$ and $\mathcal{S}_2(n)$ respectively be the sets containing the nodes that are still to be processed and those selected in the previous $n-1$ steps.

2. The objective is to select an additional node $j^\star$ from $\mathcal{S}_1(n)$ so that this node has the highest correlation with respect to all the other sensors in $\mathcal{S}_1(n)$. In this way, we

maximize the informative value of $j^\star$ with respect to the sensors that are not selected. Note that this amounts to selecting $j^\star$ in such a way that the conditional variance with respect to the other sensors in $\mathcal{S}_1(n)$ is minimized. That is:

$$j^\star = \text{argmax}_{j\in\mathcal{S}_1(n)} \left(\sum_{i\in\mathcal{S}_1(n)} \frac{\sigma_{ij}^2}{\sigma_j^2} \right) . \tag{2.10}$$

After node $j^\star$ is identified, we set $\mathcal{S}_1(n+1) \leftarrow \mathcal{S}_1(n)\backslash\{j^\star\}$ and $\mathcal{S}_2(n+1) \leftarrow \mathcal{S}_2(n)\cup\{j^\star\}$.

3. If $n = M$ the algorithm stops, otherwise we set $n \leftarrow n+1$ and go back to step 2 above.

Note that the complexity of this algorithm is $\mathcal{O}(MN)$, whereas the optimal schedule is obtained through an exhaustive search, which entails checking $\binom{N}{M}$ monitoring schedules. Note also that the algorithm considers the same metric as CB-DNS with the difference that, in this case, every time a sensor is selected we recompute the relevant metrics conditioning on its removal from the main set. As it will be numerically shown below, this leads to a major performance improvement.

For a generic number of time slots per collection round, $T > 1$, Algorithm 2 can be generalized as follows. To simplify the exposition, we assume that N/M is integer and that $T = N/M$. In a single round, we have T time slots and we need to select exactly M nodes for each of them. The new algorithm starts with set $\mathcal{S}_1 = \mathcal{N}$ and outputs T sets $\mathcal{S}_2^t$, with $|\mathcal{S}_2^t| = M$, $\forall\, t \in \{1,\ldots,T\}$. The new algorithm is described next.

Algorithm 3:

1. Let $n \in \{1,\ldots,M\}$ be the n-th step and let $\mathcal{S}_1(n)$ and $\mathcal{S}_2^t(n)$, $t \in \{1,\ldots,T\}$ be the $T+1$ sets at this iteration. Use Algorithm 2, taking as input set $\mathcal{S}_1(n)$ and requiring the selection of T nodes from it.

2. Assign each of the nodes returned by Algorithm 2 to a different set $\mathcal{S}_2^t(n)$ for $t \in \{1,\ldots,T\}$. After this, these nodes are removed from $\mathcal{S}_1(n)$.

3. If $n = M$ stop, otherwise go to step 1.

In Algorithm 3, we use the same rationale as in Algorithm 2 by picking the best T sensors at each iteration and assigning them to the T disjoint sets $\mathcal{S}_2^t$. The idea is to evenly distribute the best nodes among the T selections and stop when all nodes are assigned, at which point all the selections contain exactly M nodes. The metric that we use is still based on (2.9) but each time a node is selected, it is removed from $\mathcal{S}_1$, which amounts to conditioning the statistics for the remaining nodes. For a generic T, the algorithm can be extended as we did for CB-DNS, requiring some of the nodes to resample in some slots and rotating them to balance their energy consumption.

2.6.3 Comparison of Sampling Strategies

In Fig. 2.4 we show the performance of the sampling strategies of Section 2.6 considering the synthetic signals of Section 2.2.1 for $\mu_x = 0$, $\sigma_x^2 = 1$. These graphs were obtained generating random fields $\boldsymbol{x}(t)$ for a sufficiently large number of time slots. For each spatial signal, N sensor nodes are distributed at random within the simulation area and sampling is performed from $M < N$ of them using one of the monitoring schedules from Section 2.6. As a benchmark, an *idealized* version of the CS algorithm is used to reconstruct the signal at the $N - M$ sensors that do not sample. In detail, we inverted (2.5) solving the quadratic program (2.6) by assuming a perfect knowledge of the signal's covariance matrix $\boldsymbol{\Sigma}$ and, in turn, using the most effective transformation matrix $\boldsymbol{\Psi}$ (obtained from $\boldsymbol{\Sigma}$ via the PCA technique).

In the plot, we show the Mean Square Error (MSE) between the actual and the estimated variance for the $N - M$ sensors that do not sample as a function of the number of sampling nodes M. Note that a small MSE entails a better estimation of the second order statistics of the signal at the non-sampling nodes and, in turn, a more accurate estimation of the covariance matrix $\boldsymbol{\Sigma}$. This, as will be shown in Section 2.7, leads to better reconstruction performance for all CS-based algorithms.

With the term "exhaustive" we indicate the optimal sampling procedure, obtained considering all possible $\binom{N}{M}$ selections and picking the one providing the best MSE performance.

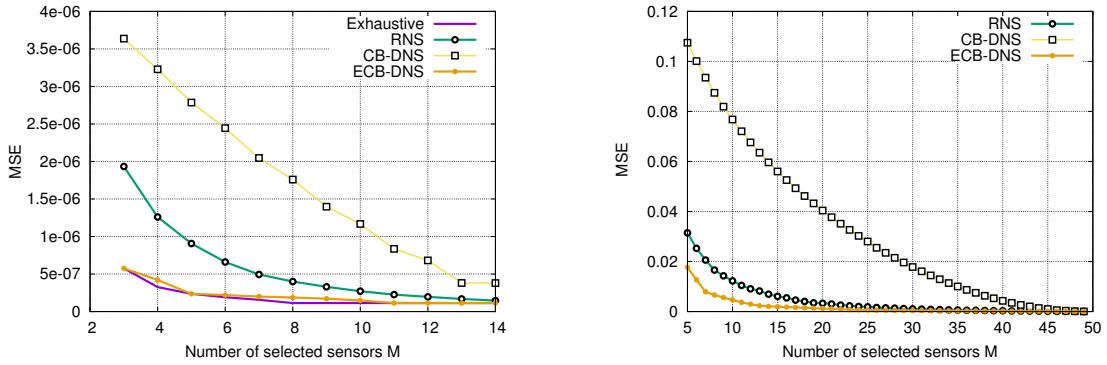


FIGURE 2.4: Comparison of sensor selection algorithms for (left, $N = 15$ and $\gamma = 10$) and $\gamma = 1$ (right, $N = 50$ and $\gamma = 1$).

Note that the optimal scheduling is shown for $N = 15$ (Fig. 2.4-left) but we could not obtain it for $N = 50$ (Fig. 2.4-right), due to the excessive computational burden in this case.

As shown in Fig. 2.4 (left, spatial correlation parameter $\gamma = 10$), CB-DNS performs unsatisfactorily and much worse than random selection (RNS). However, ECB-DNS performs much better and quite close to the optimal selection. This emphasizes the importance of conditioning and recomputing the involved statistics each time a new node is added to the monitoring schedule. The same fact is confirmed by the results of Fig. 2.4(right), which are obtained using a smaller spatial correlation parameter ($\gamma = 1$). Although not shown here in the interest of space, similar results were obtained across a wide range of correlation values $\gamma \in [0.1, 10]$ and for a higher number of nodes, $N = 50$ (although in this case the optimal selection could not be obtained to the excessive computational burden). This demonstrates the effectiveness of our conditioning approach (see Algorithms 2 and 3) and make ECB-DNS a good candidate for our CS-based algorithms of in this and next chapters.

2.7 Results

In this section, we compare temporal compression algorithms (LTC, DCT) against DSC and the discussed CS-based techniques. For the network setup, we consider $N = 50$ nodes with the signal model of Section 2.2 with varying spatial (γ) and temporal (ρ) correlation parameters and a data collection cycle length of $T = 50$ time slots. We compute the average reconstruction error at the sink $\xi \triangleq \mathrm{E}_t \left[\|\boldsymbol{x}(t) - \hat{\boldsymbol{x}}(t)\|_1 \right]$, where $\boldsymbol{x}(t)$ and $\hat{\boldsymbol{x}}(t)$ respectively

represent the original signal from the N sensors and the one reconstructed at the sink at time t, and the total average energy consumption per time slot, E, expressed in Joule (including the transmission and processing tasks performed by the WSN nodes). For the computation of the energy consumption figure, we considered the number of operations executed by the micro-controller to compress the signal and the number of bits sent to transmit it, as detailed in Section 2.2.4. Each sensor reading takes $n = 16$ bits of memory and for DSC we have used $k = 4$ bits to represent the bin identifiers.

In the graphs that follow, with CS-RNS we mean the technique of [12], which employs random sampling (see Section 2.6.1) together with the recovery scheme of Section 2.5.2, where the *sample* covariance matrix is exploited to obtain the PCA sparsification basis. CS-DNS uses the same CS-based approach, but considers the deterministic node selection technique of Section 2.6.2.

Next, we show tradeoff curves comparing ξ (y-axis) against E (x-axis), where the performance of DSC and CS schemes is obtained by varying an independent parameter as follows: **DSC:** the number of clusters N_c is varied between 5 and $N - 5$ in steps of 5; **CS-RNS:** p_{tx} goes from 0.1 to 1 in steps of 0.1; **CS-DNS:** M goes from 5 to N in steps of 5. A free parameter is not needed for DCT and LTC as our implementation of these two schemes is self-tuning, i.e., it takes the error ξ as the input parameter. As a benchmark, we also obtained a lower bound on the error recovery performance of CB-DNS by calculating the sample covariance matrix from the *complete* signal $\boldsymbol{x}(t)$, assuming that all the past samples (excluding the current one at time t) are available at the sink with no errors and at no additional cost with respect to the acquisition of the incomplete signal set. This idealized algorithm is referred to in the following plots as "Lower Bound" and has the same energy consumption as CB-DNS.

In Fig. 2.5 we look at the impact of the node selection strategy (DNS *vs* RNS) for $\rho = 0.5$ and $\gamma = 5$. CS-DNS substantially outperforms CS-RNS and the reason for this resides in 1) the aggregation strategy entailed by DNS and 2) the type of sampling (i.e., deterministic versus probabilistic). To understand the impact of each of these factors, we considered an

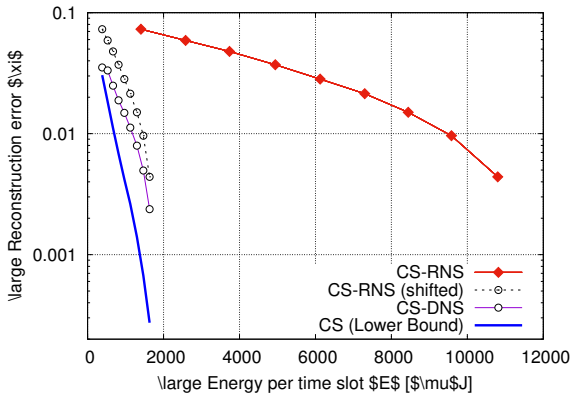


FIGURE 2.5: Reconstruction error ξ *vs* energy consumption E for $\rho = 0.5$ and $\gamma = 5$.

additional curve, obtained by left-shifting CS-RNS so that the energy consumption performance of the shifted curve is equal to that of CS-DNS. A direct comparison between this shifted curve and CS-DNS reveals the improvement, in terms of reconstruction error ξ, that is brought about by DNS (in fact, for this shifted curve the aggregation process does not affect ξ). As expected, this improvement is especially high when p_{tx} is small, i.e., when the signal is randomly sampled by a small number of sensors (top left of the plot). The reason behind this is that our DNS strategy allows for an improved estimation of the statistical properties of the signal at the sink (i.e., of its covariance matrix). Further, with DNS substantial energy savings are possible through data aggregation, as this alleviates the negative impact due to the transmission of large packet headers (assumed to be 13 bytes for the results in this thesis). DNS consistently outperforms RNS for all values of ρ and ξ when applied to all CS algorithms. For this reason, CS-RNS will no longer be considered in the rest of the chapter.

Next, we focus on the comparative analysis of CS, DSC, DCT and LTC, which is shown in Figs. 2.6– 2.9. From these figures, a few key observations can be made. **a)** When the signal is uncorrelated in space and time (Fig. 2.6), CS-DNS is the scheme of choice and performs close to its theoretical bound. Temporal correlation schemes should be avoided, as their energy consumption in this case is higher than that incurred in sending all the data uncompressed (given by the abscissa value where CS and DSC reach $\xi = 0$). Especially, LTC is rather inefficient as two coefficients (i.e., one segment) are sent for each data point for small values of ρ.**b)** As the signal correlation increases (see Fig. 2.7, with $\rho = 0.5$ and $\gamma = 5$),

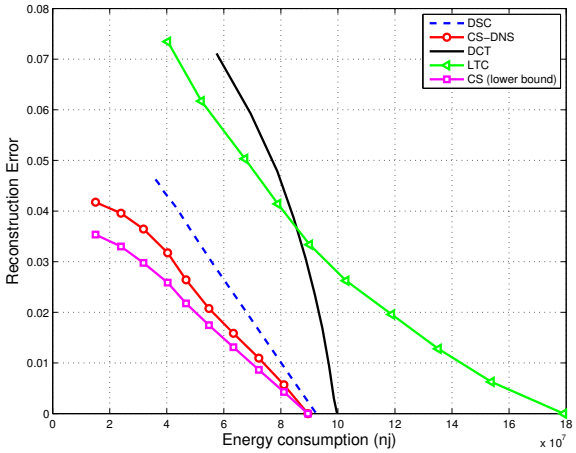


FIGURE 2.6: Reconstruction error ξ *vs* energy consumption E for $\rho = 0$ and $\gamma = 0.001$ (signal uncorrelated in time and space).

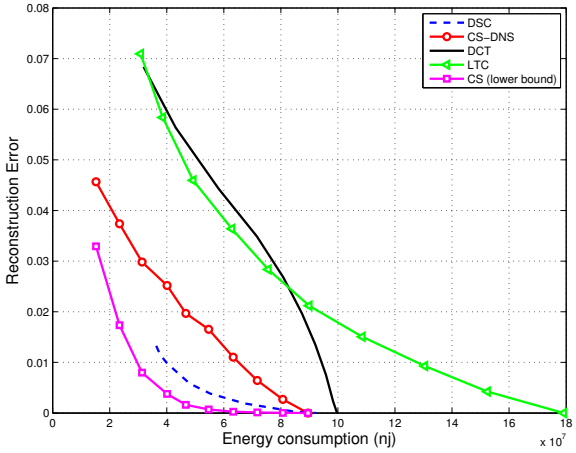


FIGURE 2.7: Reconstruction error ξ *vs* energy consumption E for $\rho = 0.5$ and $\gamma = 5$ (signal average correlated in time, but highly correlated in space).

DSC better uses the increased spatial correlation, outperforming CS as can be observed for $\xi \geq 1$. The performance of DCT and LTC is only marginally improved. **c)** For a temporal correlation as high as $\rho = 0.8$ (see Fig. 2.8), the performance of DCT and LTC reaches that of CS. **d)** For signals that are highly correlated in time but spatially uncorrelated, the situation is reversed, see Fig. 2.9. In this case, DCT and LTC both outperform CS, whereas DSC performs worst and its use is not recommended.

As a general remark, the gap between CS-DNS and its lower bound increases for increasing signal correlation. This means that the quality of the sample covariance matrix is highly impacted in this case. This, in turn, affects the accuracy of the related PCA transform, providing less accurate approximations of the signal. These results indicate that there is still some room for improvement for CS, whose performance can be ameliorated so as to

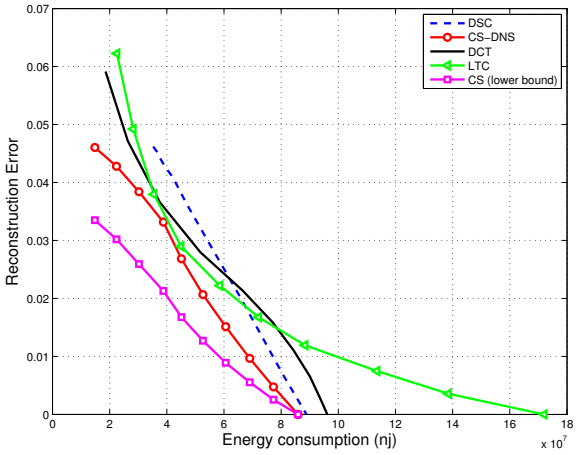


FIGURE 2.8: Reconstruction error ξ *vs* energy consumption E for $\rho = 0.8$ and $\gamma = 0.001$ (signal highly correlated in time, but uncorrelated in space).

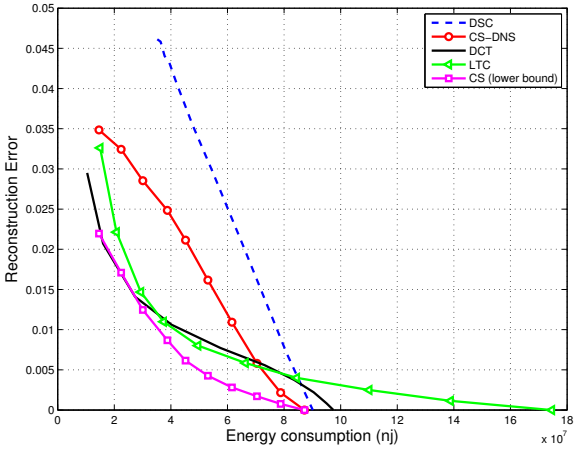


FIGURE 2.9: Reconstruction error ξ *vs* energy consumption E for $\rho = 0.98$ and $\gamma = 5$ (signal highly correlated in time, but uncorrelated in space).

approach that of DSC and temporal-compression schemes. To achieve this goal, we need to concoct improved covariance estimators for incomplete signal sequences. To this end, a sensible approach could be the use of spatial correlation estimators to filter the noise affecting the sample covariance. Future work has to be carried out in this direction.

All in all, however, if the correlation statistics are unknown, CS is deemed a valid compression approach as it often outperforms competing algorithms and, in the worst cases, it performs in between temporal and spatial correlation-based compression.

2.8 Chapter Conclusion

In this Chapter, we have performed a comparative performance evaluation of selected lossy compression schemes for WSN. This analysis can be seen as a preliminary study for a more in-depth research work, as it serves to pinpoint the pitfalls of the considered schemes. In fact, we found that no single algorithm performs best in all settings, and compressive sensing in general provides good performance that, in the worst case, lies in between that of temporal and spatial compression schemes. Most importantly, our analysis reveals that there are interesting avenues for more research. One of these consists in the improvement of CS schemes, and specifically in the design of improved estimators for the sparsification basis (obtained through PCA), which is strictly related to the covariance structure of the signal. Also, CS schemes can be implemented as well by exploiting a clustered WSN, i.e., by restricting the scope of the matrix inversion within each of the clusters.

Chapter 3

Environmental WSN – a new proposal, Covariogram-Based Compressive Sensing

3.1 Itroduction

In this chapter, we have improved the CS-based compression scheme of chapter 2. In chapter 2 we found that no single algorithm performs best in all settings, and compressive sensing in general provides good performance that, in the worst case, lies in between that of temporal and spatial compression schemes. In this chapter, the reconstruction step of CS-based algorithms has been improved considering the concept of variogram estimation and using it to refine the (estimated) covariance matrix at the data collector. As a result, the reconstruction error is substantially reduced, leading to more accurate representations of the signal. As an additional contribution, the sparse sampling strategy is improved to implement a more effective selection of the subset of nodes that are sampled at each data collection round. The impact of variogram estimation and sparse sampling is numerically assessed through a simulation tool developed in Matlab. The standard kriging reconstruction algorithm (from geo-statistics) is also considered and compared against CS-based reconstruction.

Our contribution: we consider lossy compression scheme that rely on our proposal which is an improved compressive sensing. So the first objective of this chapter is to shed some light on the comparative performance of existing algorithms, testing them for synthetic and real environmental signals. As a second major contribution, we introduce new sampling and covariance estimation strategies for CS. Our covariance estimation algorithm integrates the concept of covariogram estimation, which is a widely used tool in geostatistics [14]. The resulting CS technique, termed *covariogram-based compressive sensing*, outperforms all previous lossy compression schemes, showing an excellent capability to adapt to changes in the underlying correlation structure, while also providing compression-vs-energy tradeoffs that approach those of optimal CS algorithms (where the signal correlation structure is perfectly known at the receiver). For its design, we adopt the Kronecker compressive sensing framework of [24] which allows to jointly sparsify the input signal along the temporal and spatial dimensions. As a distinctive feature of our work, we compute the sparsification bases for CS on the fly and based on the statistical properties of the signal. We underscore that previous work [24][25] used signal independent Wavelet transforms, which work well for video signals but are often inadequate for non-stationary data from WSN fields [21]. The result is a CS spatio-temporal compression scheme that learns the best projections according to the signal features and provides excellent results in terms of adaptation capability in the presence of highly non-stationary signals, as will be shown in Section 3.3.

Organization of this chapter: our new covariogram-based compressive sensing scheme is presented in Section 3.2. In Section 3.3, we numerically evaluate the considered compression algorithms and, in Section 3.4, we summarize our main findings.

3.2 Covariogram-based Compressive Sensing (CB-CS)

In this section, we present an improved CS-based algorithm that embeds covariogram estimation techniques into the construction of the transformation bases $\mathbf{\Phi}(t)$. To develop our technique, we start discussing the main drawbacks of the CS reconstruction algorithm described in Section 2.5.2 of Chapter 2. The first drawback is that the sparse representation

$\boldsymbol{u}(t)$ for the current time slot t is obtained from (2.5) and the solution of this system *solely* depends on $\boldsymbol{y}(t)$ and $\boldsymbol{\Phi}(t)$ in time slot t. Hence, while the temporal correlation is accounted for in the estimation of matrix $\boldsymbol{\Phi}(t)$ (obtained from the sample covariance matrix, see Algorithm 1) in chapter 2, the CS algorithms that are used to invert (2.5) only exploit the spatial correlation of the signal of interest. This, in turn, means that CS only takes advantage of the signal's sparsity in the spatial domain, whereas sparsity in the temporal domain is not exploited. A second problem, that will be quantified in Section 3.3, is that the sample covariance matrix is often affected by large estimation errors and this has a considerable impact on the performance of CS reconstruction algorithms.

Here, the first problem is addressed by extending (2.5) in order to cover multiple time slots; this will be achieved adapting the Kronecker compressive sensing framework of [24], which was also adopted in [25]. The second problem is instead addressed using covariograms for the estimation of the sample covariance matrix $\boldsymbol{\Sigma}$.

Kronecker CS (Kron-CS): with Kron-CS we extend our vector representation of the spatial signal $\boldsymbol{x} \in \mathbb{R}^N$ so as to include W subsequent time slots. Hence, we consider a new vector $\boldsymbol{x}_W \in \mathbb{R}^{NW}$ that develops along two dimensions, where W is the number of time slots that are jointly considered by the CS inversion tools, and N is the size of the spatial input signal $\boldsymbol{x}$. Equation (2.5) and the corresponding CS framework need then to be extended to jointly account for the temporal and spatial dimensions. Next, we do this by calculating a new sampling matrix and a new sparsifying matrix. The Kronecker product of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of sizes $P \times Q$ and $R \times S$, respectively, is:

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \cdots & a_{1Q}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{P1}\boldsymbol{B} & \cdots & a_{PQ}\boldsymbol{B} \end{bmatrix}, \tag{3.1}$$

where a_{ij} is element (i, j) of matrix $\boldsymbol{A}$. In our specific case, we want to jointly operate along the temporal dimension (W subsequent time slots) and the spatial one (N spatial samples per time slot). Hence, if $\boldsymbol{\Phi}_{\mathrm{T}} \in \mathbb{R}^{W \times W}$ is the sparsifying basis that we use for the temporal

domain and $\mathbf{\Phi}_{\mathrm{S}} \in \mathbb{R}^{N \times N}$ is the sparsifying basis for the spatial domain, it can be proven that $\mathbf{\Phi}_W = \mathbf{\Phi}_{\mathrm{T}} \otimes \mathbf{\Phi}_{\mathrm{S}}$ with $\mathbf{\Phi}_W \in \mathbb{R}^{NW \times NW}$ is the joint sparsifying basis over time and space [24].

If t is the present time slot, the new signal is $\boldsymbol{x}_W(t) \in \mathbb{R}^{NW}$:

$$\boldsymbol{x}_W(t) \stackrel{\text{def}}{=} [\boldsymbol{x}(t-W+1)^\dagger, \ldots, \boldsymbol{x}(t-1)^\dagger, \boldsymbol{x}(t)^\dagger]^\dagger \,, \tag{3.2}$$

which covers a time window of W slots. Likewise, the new sampled signal is a vector of MW readings, i.e., M readings are acquired for each of the W subsequent time slots. That is:

$$\boldsymbol{y}_W(t) \stackrel{\text{def}}{=} [\boldsymbol{y}(t-W+1)^\dagger, \ldots, \boldsymbol{y}(t-1)^\dagger, \boldsymbol{y}(t)^\dagger]^\dagger \,, \tag{3.3}$$

with $\boldsymbol{y}_W(t) \in \mathbb{R}^{MW}$. At time t, the new transformation matrix is obtained applying the Kronecker product as $\mathbf{\Phi}_W(t) = \mathbf{\Phi}_{\mathrm{T}}(t) \otimes \mathbf{\Phi}_{\mathrm{S}}(t)$ and the new sampling matrix $\boldsymbol{R}_W(t) \in \mathbb{R}^{MW \times NW}$ is:

$$\boldsymbol{R}_W(t) = \begin{bmatrix} \boldsymbol{R}(t-W+1) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}(t-W+2) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{R}(t) \end{bmatrix} , \tag{3.4}$$

where $\boldsymbol{R}(\cdot) \in \mathbb{R}^{M \times N}$, $\mathbf{0} \in \mathbb{R}^{M \times N}$ is the all zero matrix and $\boldsymbol{R}_W(t) \in \mathbb{R}^{MW \times NW}$. Note that (3.4) has a block-diagonal structure as the entries of the sparse vector are grouped by signal, as done in [24, 25]. Hence, we have that $\boldsymbol{y}_W(t) = \boldsymbol{R}_W(t)\boldsymbol{x}_W(t)$ and (2.5) becomes:

$$\boldsymbol{y}_W(t) = \boldsymbol{R}_W(t)\mathbf{\Phi}_W(t)\boldsymbol{u}_W(t) + \boldsymbol{n} = \mathbf{\Psi}_W(t)\boldsymbol{u}_W(t) + \boldsymbol{n} \,, \tag{3.5}$$

where $\mathbf{\Psi}_W(t) = \boldsymbol{R}_W(t)\mathbf{\Phi}_W(t)$, $\mathbf{\Psi}_W(t) \in \mathbb{R}^{MW \times NW}$, $\boldsymbol{u}_W(t) \in \mathbb{R}^{NW}$ is the sparse representation of $\boldsymbol{x}_W(t)$ and $\boldsymbol{n}$ is the measurement noise.

Covariogram-based CS (CB-CS): for the present and the previous $W-1$ time slots we estimate the transformation matrix $\mathbf{\Phi}_{\mathrm{S}}$ using the covariogram theory. A typical measure of

statistical distance in spatial phenomena is provided by the experimental variogram (EV). Assume the random variable X defines the observed spatial phenomenon and $x(\boldsymbol{p})$ represents a sample of X at location $\boldsymbol{p} \in \mathcal{D}$. Then, for a given geographical distance h (referred to as *lag*), EV is defined as [26]:

$$\gamma(h) = \frac{1}{2N(h)} \sum_{\boldsymbol{p}' \text{ s.t. } \|\boldsymbol{p}-\boldsymbol{p}'\|_2=h} \left[x(\boldsymbol{p}) - x(\boldsymbol{p}')\right]^2 , \tag{3.6}$$

where $N(h)$ is the number of data pairs at distance $h > 0$.[1] The spatial correlation of X can be estimated through the computation of EV values for several distances h using available samples. Function $\gamma(h)$ can be experimentally evaluated. However, (3.6) is often insufficient to provide correlation information for CS signal reconstruction schemes. In fact, these often require correlation values between locations where no samples are available. To solve this issue, in geostatistics correlation models are established by fitting a curve on the computed EV values. This curve is then used to approximate the correlation of X at arbitrary locations. The best known covariogram functions follow the spherical, Gaussian and exponential models. These have been successfully used to describe a large number of environmental phenomena:

- Spherical:

$$\gamma_{\mathrm{s}}(h) = \begin{cases} n + s \times \left[1.5\left(\frac{h}{a}\right) - 0.5\left(\frac{h}{a}\right)^3\right] & h \leq a \\ n + s & h > a\,. \end{cases} \tag{3.7}$$

- Gaussian:

$$\gamma_{\mathrm{g}}(h) = n + s \times \left[1 - \exp\left(-\frac{3h^2}{a^2}\right)\right] . \tag{3.8}$$

- Exponential:

$$\gamma_{\mathrm{e}}(h) = n + s \times \left[1 - \exp\left(-\frac{3h}{a}\right)\right] . \tag{3.9}$$

n is the *nugget*, which represents the value of $\gamma(0)$, $n + s$ is the *sill*, which is related to the limit of $\gamma(h)$ as h grows to infinity, i.e., $n + s = \lim_{h\to\infty} \gamma(h)$, and a is the *range*, which

[1] When evaluating the EV of a randomly generated set of points, the lag is defined in a discrete set of contiguous intervals rather than as a continuous variable, for further details see [26]

represents the distance h for which $\gamma(h)$ converges (within a certain small tolerance) to the limiting value $n + s$. The spherical model is almost linear at small distances but flattens out as h increases. The Gaussian model shows a parabolic behavior near the origin and for this reason fits well extremely continuous phenomena. The exponential family provides excellent approximations for many classes of signals, such as meteorological data (as we show in Section 3.3).

The best covariogram fit is obtained as follows. Equation (3.6) is used to compute the experimental variogram for each pair of nodes $i, j \in \mathcal{N}$, i.e., every time the two nodes send their samples within a data collection round, we update the corresponding EV. Hence, we obtain a number of EV estimates according to the nodes that transmit their readings and we sort them according to the geographical distance h of the corresponding sensors. We consider that the spatial distribution is circularly symmetric, i.e., $\gamma(h)$ only depends on the distance h and not on the initial point $\boldsymbol{p}$. In this case, the nugget n is obtained as $n = 0.5 \sum_{i \in \mathcal{N}} \sigma_i^2 / N$, where σ_i^2 is the variance associated with the readings sampled at node i. To fully specify a covariogram function, we still have to compute the parameters s and a. Here, we obtained them using two nested Brent-Dekker minimizations. This is a fast and reliable algorithm using bisection, the secant method and inverse quadratic interpolation [27]. The algorithm has the reliability of bisection and is often as fast as less reliable methods.

For a given phenomenon, the just described procedure is run to identify the best suited covariogram model from (3.7), (3.8) and (3.9). The one that best matches the actual spatial correlation (represented by the EV values) is then adopted. Upon calculating $\gamma(h)$, we use it to estimate the covariogram matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{N \times N}$. From the theory in Chapter 12 of [26], for any given distance h, the covariogram $\gamma(h)$ is related to the covariance $C(h)$ through $C(h) = n + s - \gamma(h)$. Accordingly, element (i, j) of $\boldsymbol{\Sigma}$, $\sigma_{i,j}$, is computed as $\sigma_{i,j} = n + s - \gamma(h_{i,j})$, where $h_{i,j}$ is the geographical distance between the two nodes i and j.

Compression strategy: data samples are collected from the sensor nodes every T times slots. The ECB-DNS sampling strategy is used to obtain the monitoring schedule, which is computed at the sink as per Algorithm 3 of chapter 2 and disseminated to the sensor

nodes. The sensor nodes sample in their assigned time slot(s) and aggregate their samples in as few packets as possible, which are sent to the sink at the end of the data collection round. For the sake of illustration, let $T = W$ and let T be the current time slot. The signal reconstruction algorithm at the sink works as follows:

Algorithm 1:

1. At time $t = T$, the sink collects $\boldsymbol{y}_W$. The sampling matrix $\boldsymbol{R}_W$ for the present and the previous $W-1$ time slots is computed using (3.4). Note that, according to ECB-DNS, the monitoring schedule changes from slot to slot and for this reason matrices $\boldsymbol{R}(i)$ with $i \in \{1, \ldots, T\}$ differ.

2. The covariance matrix $\boldsymbol{\Sigma}$ is estimated through the covariogram-based technique in the previous paragraph, i.e., using the best covariogram fit at time slot T.

3. PCA is used with $\boldsymbol{\Sigma}$ to obtain the spatial transformation basis $\boldsymbol{\Phi}_{\mathrm{S}}$. For the temporal transformation basis, $\boldsymbol{\Phi}_{\mathrm{T}}$, we use DCT as it has a good energy compaction property, see, e.g., Chapter 4 of [28]. The joint transformation basis over time and space is obtained through the Kronecker product as $\boldsymbol{\Phi}_W = \boldsymbol{\Phi}_{\mathrm{T}} \otimes \boldsymbol{\Phi}_{\mathrm{S}}$.

4. Matrix $\boldsymbol{\Psi}_W$ is obtained as $\boldsymbol{\Psi}_W = \boldsymbol{R}_W \boldsymbol{\Phi}_W$ and the sparse signal estimate $\hat{\boldsymbol{u}}_W$ is retrieved solving (3.5) for $\boldsymbol{u}_W$.

5. The signal $\boldsymbol{x}_W$ is approximated as $\hat{\boldsymbol{x}}_W = \boldsymbol{\Phi}_W \hat{\boldsymbol{u}}_W$. With this approach, we obtain an approximation of $\boldsymbol{x}$ for each of the W time slots.

Algorithm 1 can be repeated every T time slots, i.e., signal recovery for an entire window occurs at the end of each data collection round. Alternatively, one could perform this procedure at every time slot. Considering slots T and $T+1$, at time T we obtain $\hat{\boldsymbol{x}}_W(T) = [\hat{\boldsymbol{x}}(1)^\dagger, \ldots, \hat{\boldsymbol{x}}(W)^\dagger]^\dagger$ and at time $T+1$ we retrieve $\hat{\boldsymbol{x}}_W(T+1) = [\hat{\boldsymbol{x}}(2)^\dagger, \ldots, \hat{\boldsymbol{x}}(W)^\dagger, \hat{\boldsymbol{x}}(W+1)^\dagger]^\dagger$. Note that the last two vectors overlap in $W-1$ positions, i.e., $W-1$ instances of the spatial signal $\boldsymbol{x}$ are reestimated at time $T+1$. This fact allows the refinement of the estimated multidimensional signal. In this chapter we have used the latter approach by

always replacing the old estimates with the new ones. Finally, we remark that unlike [24] and [25], we adaptively compute the spatial transform according to our combined covariogram-PCA estimation procedure. This allows tailoring the transformation basis on the non-stationary statistical structures of real WSN datasets. In the literature, the 2D Wavelet transform was used instead. This transform is independent of the data statistics and, while providing excellent results for video flows, does not necessarily perform well for non-stationary WSN data [21].

3.3 Results

In this section, we compare temporal compression algorithms (LTC, DCT) against DSC and the discussed CS-based techniques. For the network setup, we consider $N = 50$ nodes with the signal model of Section 2.2 of Chpater 2 with varying spatial (γ) and temporal (ρ) correlation parameters and a data collection cycle length of $T = 100$ time slots. We compute the average reconstruction error at the sink $\xi \triangleq \mathrm{E}_t \left[\|\boldsymbol{x}(t) - \hat{\boldsymbol{x}}(t)\|_1\right]$, where $\boldsymbol{x}(t)$ and $\hat{\boldsymbol{x}}(t)$ respectively represent the original signal from the N sensors and the one reconstructed at the sink at time t, and the total average energy consumption per time slot, E, expressed in Joule (including the transmission and processing tasks performed by the WSN nodes). For the computation of the energy consumption figure, we considered the number of operations executed by the micro-controller to compress the signal and the number of bits sent to transmit it, as detailed in Section 2.2.4 of Chpater 2. Each sensor reading takes $n = 16$ bits of memory and for DSC we have used $k = 4$ bits to represent the bin identifiers.

In the graphs that follow, with CS-RNS we mean the technique of [12], which employs random sampling (see Section 2.6.1 of Chapter 2) together with the recovery scheme of Section 2.5.2, where the *sample* covariance matrix is exploited to obtain the PCA sparsification basis. CS-DNS uses the same CS-based approach, but considers the deterministic node selection technique of Section 2.6.2. CB-CS is our new covariogram-based technique of Section 3.2, where with "(DCT)" we indicate that a DCT transform is used for the temporal dimension (matrix $\boldsymbol{\Phi}_\mathrm{T}$), whereas "(iid)" indicates an identity transform. Kron-CS indicates

the Kronecker CS technique of [24, 25] where DCT and the *sample* covariance matrix are considered for $\mathbf{\Phi}_{\mathrm{T}}$ and $\mathbf{\Phi}_{\mathrm{S}}$, respectively.

Next, we show tradeoff curves comparing ξ (y-axis) against E (x-axis), where the performance of DSC and CS schemes is obtained by varying an independent parameter as follows: **DSC:** the number of clusters N_c is varied between 5 and $N-5$ in steps of 5; **CS-RNS:** p_{tx} goes from 0.1 to 1 in steps of 0.1; **CS-DNS**, **CB-CS** and **Kron-CS:** M goes from 5 to N in steps of 5. A free parameter is not needed for DCT and LTC as our implementation of these two schemes is self-tuning, i.e., it takes the error ξ as the input parameter. As a benchmark, we also obtained a lower bound on the error recovery performance of CB-CS by calculating the sample covariance matrix from the *complete* signal $\boldsymbol{x}(t)$, assuming that all the past samples (excluding the current one at time t) are available at the sink with no errors and at no additional cost with respect to the acquisition of the incomplete signal set. This idealized algorithm is referred to in the following plots as "Lower Bound" and has the same energy consumption as CB-CS.

In Figs. 3.1 and 3.2 we compare the performance of the compression algorithms for three selected (ρ, γ) pairs. Note that, for this set of results, the signals were generated using the framework of Section 2.2.1 of Chpater 2 and their statistics is s*tationary*. As can be seen from these plots, temporal compression algorithms perform worse than those exploiting the spatial (DSC) or spatio-temporal (CS) features of the signal. Moreover, when the spatial correlation γ is high (Fig. 3.1), CS-DNS and Kron-CS perform worse than DSC. Instead, our CB-CS algorithm still performs very close to the lower bound and better than all the other schemes across the entire range of correlations.[2] This indicates that the proposed covariogram-based estimation technique of Section 3.2 is very effective in providing reliable and robust estimates of the covariogram matrix, although the signal collected at each round is incomplete. We also note that the gap between CB-CS (DCT) and CB-CS (iid) increases with an increasing temporal correlation, which can be observed as we go from Fig. 3.1 (uncorrelated in time and correlated in space) to a case where data is correlated in time and space (e.g., $\rho = 0.5$ and $\gamma = 5$, not shown in the interest of space). Remarkably,

[2]We checked this through an extensive simulation campaign, although we report here only the most significant results for the sake of space.

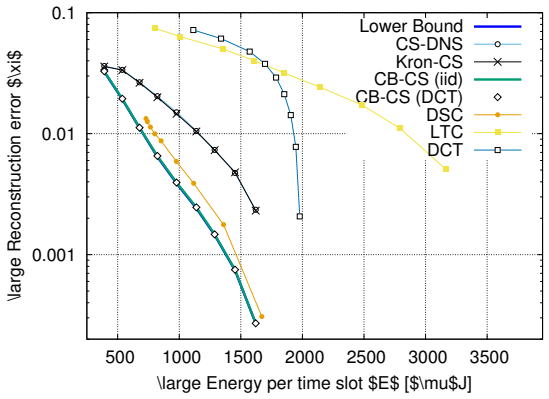


FIGURE 3.1: Reconstruction error ξ *vs* energy consumption E for $\rho = 0$ and $\gamma = 5$ (signal uncorrelated in time, but highly correlated in space).

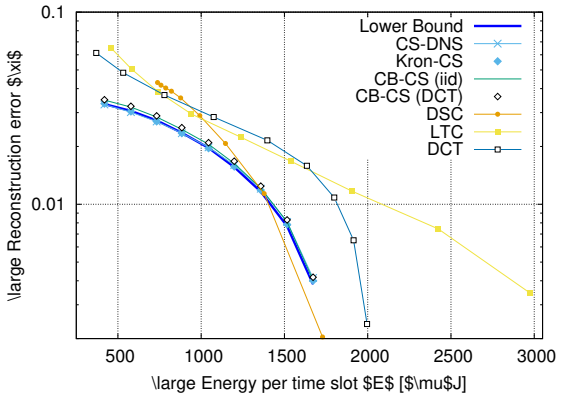


FIGURE 3.2: Reconstruction error ξ *vs* energy consumption E for $\rho = 0.8$ and $\gamma = 0.1$ (high temporal correlation and low spatial correlation).

in Fig. 3.2 CB-CS (DCT and iid), Kron-CS and CS-DNS perform similarly, as for this particular choice of the parameters (very high temporal correlation) the *sample* covariance matrix accurately matches the real covariance structure of the signal. As expected, when the number of sampling sensors is low, DSC performs worse than temporal compression schemes because the signal is uncorrelated in space. Moreover, the Kronecker framework alone does not provide substantial benefits and this holds across the entire range of correlations (we will see shortly that this is not the case for non-stationary signals). Another interesting observation, as ρ increases, the performance of DSC remains unchanged, whereas CB-CS takes advantage of the increased temporal correlation by lowering its reconstruction error across the entire range of M.

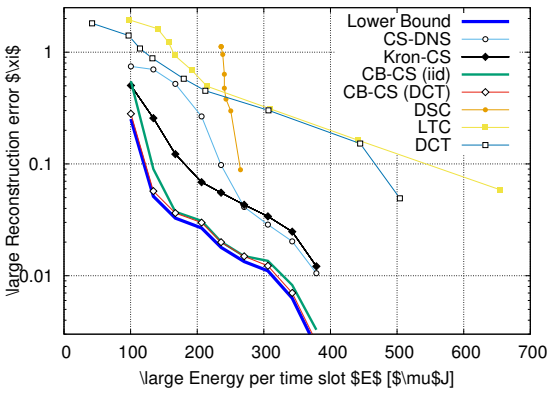


FIGURE 3.3: Results for experimental data: temperature dataset from [2].

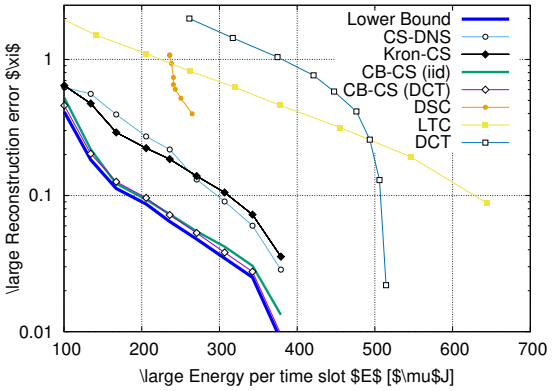


FIGURE 3.4: Results for experimental data: precipitation dataset from [2].

In Figs. 3.3 and 3.4 we show the reconstruction error *vs* energy tradeoff for real and *non-stationary* environmental data. Fig. 3.3 refers to a temperature dataset from [2] and provides further insights into the comparative performance of the selected schemes. Specifically, from this plot we observe a much bigger gap between Kron-CS and CS-DNS. This gap can also be observed, to a smaller extent, in Fig. 3.4 (precipitation dataset from the same database). The reason for this resides in the fact that for the temperature dataset the temporal correlation is higher and the spatial correlation is lower. The smaller spatial correlation is also confirmed by the fact that, in Fig. 3.3, DSC performs worse than temporal compression in terms of reconstruction error when, e.g., less than 50% of the nodes transmit. In this case, the impact of sparsifying over the temporal dimension, using DCT, is noticeable when M is small, as testified by the gap between CB-CS (iid) and CB-CS (DCT). Also, our CB-CS algorithm still performs close to the lower bound and outperforms the other schemes, even in the presence of non-stationary data.

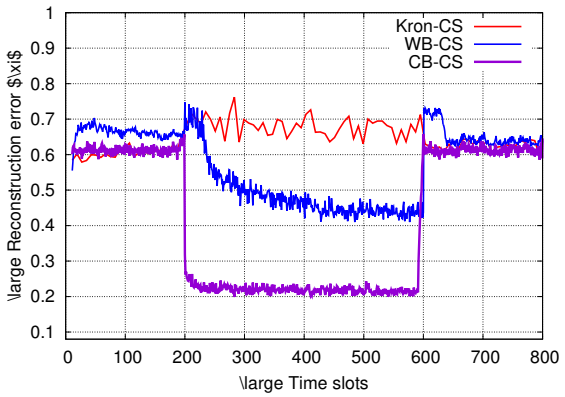


FIGURE 3.5: Results for a non-stationary data trace for $M = 15$ and $N = 50$, i.e., 30% of the nodes transmit in each collection round.

Finally, in Fig. 3.5 we look at the temporal behavior of CS-based solutions. Here, we perform an experiment considering a *non-stationary* dataset where in the first time period (between time slots 0 and 200) temporal and spatial correlations are set to $\rho = 0.8$ and $\gamma = 0.001$, respectively, to reproduce a scenario with uncorrelated spatial samples and high temporal correlation. At time 200, these values are changed to $\rho = 0.5$ and $\gamma = 5$ and they are finally restored to the initial values from time 600 onwards. In this plot we show a new scheme, termed Window Based Compressive Sensing (WB-CS). This algorithm uses DNS in conjunction with the covariogram-based covariance estimation of Section 3.2 and adopts the CS reconstruction approach of CS-DNS (see Section 2.5.2 of Chpater 2). That is, in every time slot the last time window (W time slots) is used to refine the estimated covariance matrix and reconstruct the original signal through the inversion of (2.5), but the Kronecker framework is not used. This means that WB-CS only sparsifies the data along the spatial dimension. On the other hand, note that Kron-CS sparsifies the data along space and time but the spatial transformation $\mathbf{\Phi}_S$ is computed from the *sample* covariance matrix. The gap between Kron-CS and WB-CS is due to the covariogram-based estimation procedure, which produces better estimates for $\mathbf{\Phi}_S$. The gap between WB-CS and CB-CS is instead due to using the Kronecker framework, i.e., performing the joint sparsification of the signal over time and space. Overall, the highest benefits are obtained by using the Kronecker framework combined with our covariogram-based estimation, as confirmed by the results of CB-CS. We finally observe that, right after time 200 and after time 600, WB-CS shows a rather long transient period before it stabilizes to the steady state performance for the new

correlation model. This shows the benefits brought about by the Kronecker approach in terms of adaptability to abrupt changes in the signal statistics.

3.4 Conclusion

In chapter 3, we have addressed the design of spatio-temporal compression schemes for environmental wireless sensor networks by, at the same time, delving into their performance comparison with respect to relevant compression algorithms from the literature. Our results revealed that CS is a powerful tool for the compression of correlated signals (in time, space or both) and that the joint sparsification along the temporal and spatial dimensions is key to achieve good performance and improve upon distributed source coding schemes, even when signals are highly correlated in space. In addition, a crucial role is played by the spatial transformation (sparsification) basis, which has to be adapted to the specific characteristics of the signal at hand. In this chapter, we did so via a covariogram-based algorithm for the estimation of the signal covariance matrix and its refinement as time evolves. Our quantitative results, obtained for synthetic and real signals, reveal that our final *covariogram based compressive sensing* scheme performs satisfactorily across all values of correlation, and is the algorithm of choice in terms of quality of reconstruction and energy consumption at the sensor nodes.

Chapter 4

Body sensor networks – Online Dictionary compression

4.1 Introduction

Internet of Things (IoT) technology enables objects to sense the physical environment and to seamlessly integrate the gathered data into sophisticated Internet applications that allow for substantial improvements of human activities at large. The focus of this chapter is on human sensing [29] through wearable IoT devices, such as smart watches, chest straps or wristbands, which can be used to help address the individual health and the fitness needs of the users [30].

For instance, wearables can be utilized to gather and share information about the status of outpatients, making it possible to collect, record and analyze new data streams faster and more accurately. This allows for an improved access to healthcare, an increase of its quality and ultimately, a reduction in its cost. Telehealth systems could deliver care to people in remote locations and provide streams of accurate data for making better care decisions (e.g., in terms of therapy adjustments or prompt interventions). In addition, these systems are expected to have a big impact on the field of rehabilitation where, for example, users may wear e-textile systems for remote, continuous monitoring of physiological

and movement data [31]. Through IoT technology, a large number of physiological signals can be monitored including oxygen saturation, blood pressure, heart rate, respiration rate, glucose level [30, 32] and user activities such as walking, standing, sleeping, etc., can be inferred [33]. A recent survey of wearable devices and their use is offered in [30], whereas rehabilitation systems are discussed in [31].

We look at an IoT scenario for e-health, where wearables are utilized to collect physiological signals, preprocess and transmit them over their wireless interface for their final storage and manipulation via backend server infrastructures. Within this context, we are concerned with the design of online signal compression algorithms, so that the gathered signals can be effectively stored in the limited memory space of wearables and conveniently transmitted over their radio interface. Ideally, we would like this software to adapt to the signals being sampled. This means that, high resolution should be provided when the user is up to some dynamic activity and wants to track that or when a critical behavior is detected. Toward this end, we advocate the use of lossy compression as a means to reduce the space taken by the collected biosignals and, at the same time, to save battery power through a reduced transmission time. This amounts to compressing the physiological data directly at its source.

As for the physiological signals of interest, we consider one dimensional and quasi-periodic biomedical signals as those provided by typical sensors in chest straps or wristbands, i.e., electrocardiography (ECG), photo-plethysmographic (PPG) and respiratory (RESP) signals. ECG is probably the most important among them for the diagnosis of heart malfunctions and IoT technologies are expected to be very useful to assess cardiac conditions within patient-monitoring applications. Commercial devices such as the BioHarness 3 from Zephyr Technology Corporation [34] can be utilized to measure this type of signal. RESP signals are also very relevant and can be obtained from chest straps [35] or rubber straps [36] placed around the abdomen to, e.g., assess the status of outpatients affected by chronic respiratory failure and allow monitoring them from home. PPG is often available in low-cost IoT devices for the consumer market (such as smart watches or wristbands designed for fitness applications), see the Angel sensor wristband [37]. PPG can be used to estimate heart-rate [38] and recent studies indicate that blood pressure can also be inferred [39].

We believe that, despite the focus and hype on wearable technology, research on data processing algorithms for wearable IoT devices is still in its infancy and most still has to be done to take full advantage of this portable technology. In past research, a large number of compression algorithms were proposed for ECG, but signal compression has never been applied to RESP or PPG. Moreover, performance assessments were only carried out for quality of compression and reconstruction, whereas the energy consumption aspect has often been neglected. Instead, we stress that energy should be sparingly used by the software running on wearables, as these devices are often battery operated and, in turn, their energy consumption is a key consideration. Also, to the best of the authors' knowledge, no quantitative comparison among existing solutions can be found in the literature and, due to this, it is unclear which algorithms are best suited for use in wearable devices.

In this chapter, we aim at filling these gaps. First, in Section 4.2 we present a general Idea of Internet of Things. then, we state about wearable devices and their uses in Section 4.2.1. After that, we talk about biosignals and types of the biosignals that we have considered for compression in Section 4.3.1. Section 4.4 we present a taxonomy of popular signal compression schemes from the literature, touching upon linear approximations [40, 41], Fourier [42], Wavelet [43] transforms and novel compression techniques based on *compressive sensing* [44, 45] and *denoising autoencoders* [46].

A novel compression architecture based on vector quantization and pattern recognition [47] is proposed in Section 4.5, where a suitable codebook (or dictionary) is built and maintained in an online fashion to efficiently represent data patterns. Compression is achieved as codebook indices are sent to the decompressor in place of the original time series. Despite its simplicity, this technique is found to be appealing due to its excellent performance in the high compression regime.

To summarize, the main contributions of this chapter are:

- Idea of Internet of Things and wearable devices as one of its main branches.
- Classification of existing signal compression schemes that are amenable to implementation on wireless wearable IoT devices.

- A simple but effective dictionary-based approach to the *online* compression of biosignals, along with its validation.

4.2 Internet of Things

In the last decade there has been the diffusion of a very large number of emerging technologies that enabled the gathering of data through a variety of sensors, operating in many different fields (e.g environmental, mobility-related, medical). The so collected data, as a result, gave rise to the development of applications able to organize it, analyze it a finally present it in a suitable way, thus allowing for substantial improvements in human activities at large.

This concept grew in what is now called the *Internet of Things* (IoT) paradigm: the scenario is that of a very large number of sensor devices with communication capabilities, connecting both in an ad-hoc fashion or to a sink node, which collects data for further elaboration. These sensor nodes would not have high capabilities in terms of computational power and self-sustainability. This framework, according to the IoT vision, will be realized by enhancing everyday objects with a communication interface and then exploiting their ability to sense and collect data.

Up to now, the development of the Internet of Things, has proceeded by proliferation of *islands*, i.e. groups of devices able to communicate and exchange information that are, due to multiple incompatibility issues, unable to communicate with devices of other IoT islands. In [48], the authors examine this fundamental problem and suggest directions that are to be taken in order to reach a *real* Internet of Things: enabling, not only the collection of the sensed data in the physical world, but also the exchange and elaboration of it in the digital domain.

This convergence should take start, again according to [48], from the design of a general architecture, able to integrate existing efforts and current technologies and thus overcoming the current fragmentation where "*many INTRAnets of things cannot operate in an*

well-integrated INTERnet of Things". Thus, it is not advisable to design from scratch a communication architecture that would, probably, just increase the number of different and non-interoperable communication architectures, but rather to provide a sort of *middleware* that would let, already operating technologies, to be interoperable.

4.2.1 Wearable Devices

A category of devices whose presence has become prevasive in the very last years is that of *wearable devices*: if we include in this class also *smartphones*, then it is clear that an immense potential was unleashed as these devices, originally made for communication, nowadays ship a large number of sensors. As noted in [3], the IoT paradigm has recently shifted from a scenario in which sensors are integrated in the environment to one in which we, as humans, carry ourselves the sensor devices and participate actively in the sensing process.

As stated in [3], it is observable that a conjunction in the spread of two technology advancements has boosted the development of applications for which the sensing process has a personal and social character: the first technology involves the update of the communication network and the second the diffusion of inexpensive sensing devices. The rapid improvements which the cellular network has undergone are in front of everyone and they are paralleled by the update of the core network. The terminals too, namely the now-called *smartphones*, besides powerful processors, carry, nowadays many sensing devices which targets different fields: imaging (cameras), position (GPS), movement (accelerometer and gyroscope) and geomagnetical field (magnetometer) among others. Also, through a number of applications, e.g. Twitter, Facebook, Whatsapp, etc., it is possible to collect and share massive amounts of data.

New small and lightweight wearable devices for the sensing of biomedical signals are also appearing in the market, thus pushing the interest of reseachers and posing new challenges. The first section of market to see a spread of adoption has been *fitness*: wristbands able to communicate to the mobile phones (usually via Bluetooth) started to integrate accelerometers that could give sufficiently accurate estimate of the number of steps walked, UV light

sensors that can measure how much solar radiation one have been exposed to and, most importantly, photodiodes able to generate a *photoplethysmogram* (PPG) signal. A possible second section of market, which is, according to the author, the next one which will go through rapid expansion, is that of biomedical devices for the monitoring of non critical patients such as elderly people or patients who have to undergo Holter exam: this, of course, entails a shift in requirements from energy-economy to reliability.

4.2.2 Fitness

As already noticed, the fitness market was the first to see the appearence of products directly involved in the measurements of biomedical quantities: most modern smartphones already implemeted some of these functions but dedicated devices let people not owning cutting-edge smartphones, to benefit of these technologies and to shape the device on the purpose. Dedicated fitness devices could be thus improved by adding more sensors to collect more quantities, and, most importantly, saving energy, given that they incorporate a separate battery.

The proliferation of these devices was spurred by factors similar to those that made smartphones so widespread: first of all the integration of reliable sensors that have become quite cheap and then the possibility to interface them with the internet, thus storing online, sharing and analyzing the signals they collect. Among others, the possibilty to generate PPG signals form *photodiodes* positioned on the wrist, as proven by some of the devices already in the market, has impressed a faster pace to their diffusion. The PPG signals generated this way led to the possibility of estimating an important parameter such as the *Heart Rate* (HR): the main issue that wearable devices have to overcome is motion noise, which, under intensive exercise could be very disrupting. This problem along with others, though not completely solved, is already analyzed and studied in works such as [49] [50].

A minor group of more sophisticated and advanced devices such as the BioHarness form Zephyr [51], are able to collect a proper ECG signal (single lead). These are devices that,

even though available on the market, are targeted to professionals and athletes at high levels and their cost confirms this assertion.

4.2.3 Medical

The hospital wards have different requirements from devices which retrieve biomedical signals, although some are shared with the fitness world. A lasting battery would be appreciated also by this last category but precision and reliability are of foremost importance, given that we are talking about unhealthy people. Thus the possible sources of errors should be bounded or errors should be reported instantly in some way. In fact, in the Hospitals, biomedical signals are gathered through machines that are often large, expensive and have very high computational capabilities (and are, implicitly, not portable); these devices were made with the purpose of being reliable and precise, given that they are mainly used in scenarios such as Intensive Care Units. Another important factor, which could seem unimportant at a first sight, is cost: these devices should be accessible to as much institutions as possible given that, in many Public Health Hospitals, their adoption will be directly influenced by the costs.

This work is targeted to health devices with the vision that, in the coming years, assistance to people affected by not serious patologies, could be automated through the use of portable and inexpensive devices to let, for example, these people continue living at their houses; another improvement could be done inside the Hospital wards where, very often, the staff is undersized. The challenge is, thus, to develop the fitness devices, already available, and to port them to a more technical level of application.

4.3 Signal Compression in wearable devices

This work is focused on temporal compression of biosignals, with the aim of allowing battery-operated devices to save energy: it is known, indeed, that the most part of the energy expenditure of a mobile device is given by its wireless communication interface [52]. This

fact led, in the past, to a common pitfall: the concern to compress as much as possible, very often obfuscated the fact that the processor too, when compressing, consumes energy; there exists thus the possibility for this energy to be more than that required for sending the data without any compression. A complete study of this possibility was accomplished in [52], where a number of lossy compression techniques, spanning a large number of different approaches, were tested in terms of compression capabilities and energy expenditure. The reconstruction capability of the algorithms was taken out of the study by implementing all of them in a way, that let the tolerance, i.e. the difference between original and reconstructed sequence, to be set as a parameter common to all. The results show that there is actually a threshold below which compression becomes inefficient, for it entails more energy than that needed for uncompressed data forwarding. It is remarked that, although compression can be *lossless* and *lossy*, this work only considers lossy techniques.

A group of appealing techniques are also advised in [52], as these techniques are beyond, or near to, the efficiency treshold: among these there is the *Discrete Cosine Transform* (DCT), for the transform based approach, and the *Lightweight Temporal Compression* (LTC), for the linear approximation approach (they will be explained in detail later). For this reason, these techniques are considered for comparison later in this work, even though in [52] a very constrained hardware architecture was considered, such as the MSP430, which is far a different choice from that made in this work.

4.3.1 Biomedical signals

Biomedical signals, or biosignals, are signals generated form the activity of the body during its everyday functions; these signals, collected as voltages or currents through some types of transducers, can have different nature: chemical, electrical, mechanical and magnetical and can be instrumental to the diagnosis process. Among the most commonly analyzed functions there are respiration, heart activity, brain activity and many others. A first dicotomy that can be made involves the presence of some repetition: signals such as ECG, PPG, Respiratory Signal and Arterial Blood Pressure (ABP) clearly present some sort of

repeatability given by the cycling activity of the organs involved; other type of signals, on the contrary, do not exhibit a high degree of repetition.

In this work only repetitive signals will be taken into consideration, for which the redundancy can be removed thus achieving compression. The word *repetitive* was used, rather than periodic, to highlight the fact that the repeatability occurs with some variations. We can formally define such a signal, allowing some variation, both in time and amplitude, in the definition of a periodic signal; so, a *quasi-periodic* signal can be mathematically defined as

$$x(t) = x(t + T + \Delta T) + \Delta x,\ t \in \mathbb{R}_+, \tag{4.1}$$

where ΔT and Δx are random variables that in general, but not necessarily, satisfy $\Delta T << T$ and $\Delta x << (\max x - \min x)$. Since we operate on discrete time signals, we can write

$$x(n) = x(n + T + \Delta T) + \Delta x, \quad \Delta T, \Delta x \in \mathbb{R}_+,\ n \in \mathbb{N}. \tag{4.2}$$

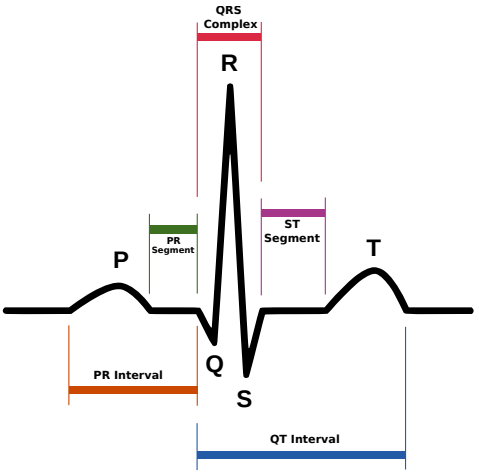


FIGURE 4.1: Basic structure of a typical QRS complex

4.3.2 Types of the considered biosignals

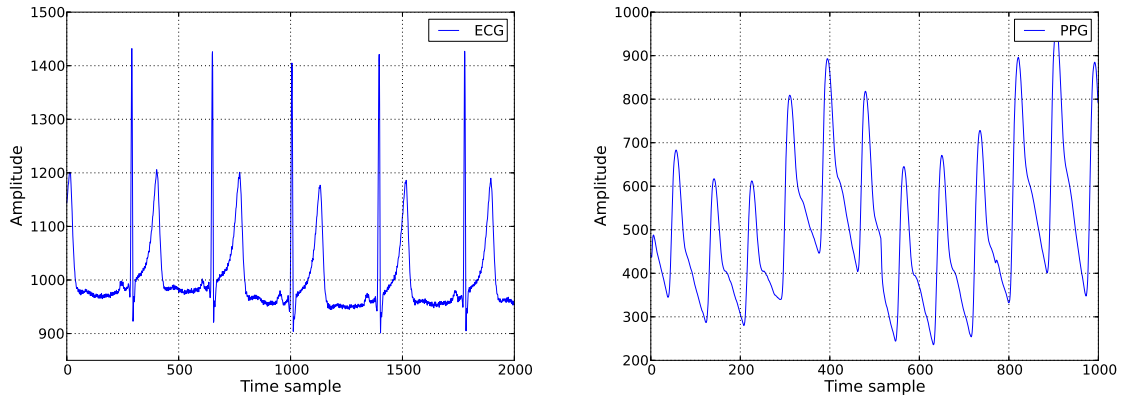
ECG The heart pumps blood through arteries and veins, permitting it to oxygenate organs and to carry out carbon-dioxide; to this aim, it follows a cycle in which, it generates voltage differences which activates its muscular structure and that are measurable through the skin immediately above it. The myocardial activity of the heart is induced by a signal which

structure has been deeply studied: it is formed by three groups of components and totally by five components: these are the P-, Q-, R-, S- and T-wave, among which the Q-R-S complex forms a special group. A typical P-QRS-T period of an ECG trace from a healthy person is shown in Figure 4.1.

We note that the most high-frequency component is the so-called *R peak*: it is indeed this peak that is used to keep track of the Heart Rate (HR); however all the other components still convey important information about the health state of a person's heart and are, thus, to be preserved in their shape.

As already mentioned, there exists a large amount of literature on efficient methods for locating the R peaks: among these the most widely used has certainly been the Pan-Tompkins algorithm [53]. Using a sequence of passband filtering, squaring, integrating and tresholding it proved to be a very effective procedure and it remained, until today, a very robust R peak detector. Lately, some works made large and exhaustive reviews on QRS complex detection algorithms, such as [54], while others already considered the scenario of wearable devices and made device-oriented reviews [55].

Machines able to collect the ECG signal were, in the past, only present in Hospitals and Healthcare structures, mainly in the form of *Intensive Care Unit* (ICU) bedside workstations. In the last decade a small number of new portable devices appeared in the market, which are able to collect a single lead ECG track such as the already mentioned Bioharness module [51].



(A) Typical ECG normal trace (B) Typical PPG normal trace

FIGURE 4.2: Sample traces for ECG and PPG

PPG Photoplethysmography (PPG) is a simple but effective technique used to measure volume changes in the microvascular bed of tissue: basically it works by measuring the light reflected, or refracted, by the capillaries. The PPG waveform, as can be seen in Figure 4.2b, comprises a pulsatile (AC) physiological waveform attributed to cardiac synchronous changes in the blood volume with each heart beat, and is superimposed on a slowly varying (DC) baseline with various lower frequency components. Photoplethysmography can also be used to asses oxygen saturation (SpO_2), blood pressure and cardiac output [56].

The pulse oximeter is a typical medical application of the photoplethysmography: it is a device used to monitorate, at the same time, blood oxygenation and heart rate. It is based on the physical principle according to which the colour of the blood depends on the amount of saturated hemoglobin (HbO_2). Saturated hemoglobin is red-coloured contrary to desaturated hemoglobin, which is blue-coloured. Hemoglobin, indeed, changes its structural configuration when it is involved in a chemical reaction; each configuration shows a different behavior in reflecting and refracting light: so, two different wavelengths are used to detect different saturation levels.

In recent years many fitness bands available on the market, started to integrate a photoplethysmograph able to collect PPG signals while positioned on the wrist: a LED paired with a photodiode are used to measure blood volume changes and, subsequently, to extract the Heart Rate (HR).

Other biosignals Other types of signals may include Arterial Blood Pressure (ABP) and Respiration Signal (RS). ABP is measured through a cuff, once put on the upper part of the arm and now also on the forearm, which once inflated, measures the pressure of the arteries by the *oscillometry* method. Respiration, on the other hand, must be derived, through some signal processing, from the previously described signals, namely ECG, PPG or both.

4.4 Lossy Compression Schemes' Classification

In the last few years, a great deal of work has been carried out on tools for the efficient analysis of ECG and other biosignals [57]. PPG is being intensively investigated for the estimation of the heart rate [38] and motion data is being used for activity detection [58]. Nevertheless, apart from ECG, little has been done regarding the compression of other signals, such as PPG, RESP, etc. We first focus on ECG and then elaborate on the use of compression for other signal types.

The two most important tasks to be accomplished in the ECG domain are 1) QRS complex detection and 2) signal compression. As per QRS detection, it is crucial to split the ECG time series into heart beat segments (one segment per beat) as this allows the fine-grained assessment of inter-beat signal features, which are useful to detect certain pathologies. Note that ECG can be efficiently split into beat segments as it is a quasi-periodic time series exhibiting recurrent patterns. As per signal compression, we emphasize that wearable devices are energy and memory constrained and, as such, minimizing the amount of data to store and send is an important consideration. As an example, a typical sampling rate of 250 samples per second with 12 bits per sample (e.g., from a Zephyr's BioHarness 3 device) leads to 32.4 Mbytes of data for a full day. As we shall see below, compression algorithms can easily reduce this number by 60 times to about 573 kbytes, leading to much higher transmission efficiencies. This is achieved through online dictionaries which take less than 10 kbytes, see Section 5.3 of Chapter 5.

1) QRS complex detection has been extensively studied in the literature. Several methods were proposed to detect QRS complexes and to enhance their features. The importance of QRS enhancement has been demonstrated to detect the QRS complex [59]. In particular, amplitude thresholding [60], first and second derivative methods [61], mathematical morphology [62, 63], filter banks [64], and wavelet transform techniques [65] are among the methods used for the enhancement of the QRS complex. The QRS detection is instead usually performed with a combination of techniques such as thresholding [60, 62], neural

networks [66], wavelet transform [67], matched filters [68]. These techniques are of foremost importance as they split the ECG time series into segments (i.e., the data points between subsequent heartbeats), which are then utilized for the subsequent estimation of the pulse, and for the compression of the ECG trace.

2) Signal compression. As we mentioned, most of the literature in the field of compression for biomedical signals, is devoted to compression schemes to be applied to the ECG: little was written for what concern other types of biosignals. The compression algorithms for ECG signals that have been developed so far, can be subdivided into three main families:

- **Time domain processing**: within this class we have AZTEC [69], CORTES [70] and Lightweight Temporal Compression (LTC) [40]. AZTEC and CORTES achieve compression by discarding some of the signal samples and applying a linear approximation, whereas LTC approximates the original time series through piecewise linear segments, where the two end points of a segment are sent in place of the points in between. As we show in Section 5.3 of Chapter 5, in spite of its simplicity, LTC closely matches the performance of Principal Component Analysis (PCA) [41, 71].

- **Transform based coding**: these exploit transformations such as Fast Fourier Transform (FFT) [42], Discrete Cosine Transform (DCT) [72] and Discrete Wavelet Transform (DWT) [43]. The rationale behind them is to represent the signal in a suitable transform domain and select a number of transform coefficients to be sent in place of the original samples. The amount of compression depends on the number of coefficients that are selected, the representation accuracy depends on how many and which coefficients are retained. Although the schemes belonging to this class have good compression capabilities, their computational complexity is often too high for wearable devices [73]. Lightweight implementations are possible and are considered in the present chapter. However, simpler linear and dictionary based algorithms have better performance in terms of reconstruction error as we show in Section 5.3 of Chapter 5.

- **Parametric techniques**: these schemes use neural networks [74], vector quantization [75], Compressed Sensing (CS) [44, 45, 76] and pattern matching [77]. Their rationale is to process the temporal series to obtain some kind of knowledge and use it to model the signal morphology. Recently, denoising autoencoders [46] have been proposed as universal approximators of biosignal patterns and have been shown to provide excellent compression performance and to have much smaller computational costs than competing algorithms. This is a field with limited investigation up to now. Also, these algorithms have promising capabilities for the extraction of signal features. The new technique that will be presented in this work belongs to the last category: it identifies recurrent patterns and builds a codebook (or dictionary) based on the most representative among them. Unlike previous solutions, our algorithm is not tailored to a specific type of biomedical signal, and can be applied to any quasi-periodic time series.

Despite these developments, we recall that no systematic comparison was carried out in the existing literature and, more than that, the proposed algorithms were not evaluated in terms of their energy expenditure. This is of course very important for wearables, which are battery operated and thus call for algorithms that are at the same time extremely effective and computationally cheap.

In addition, besides ECG, recent advances in technology for wearable devices have made it possible to efficiently collect and analyze other signals such as PPG, motion and respiration through body worn sensor technologies [78]. The PPG signal can be a powerful diagnostic tool due to simple, portable, and low-cost technology available for its fast, easy, and reliable acquisition and can be non-intrusively measured using wristbands or smart-watches. An increasing number of works in the literature deal with the extraction of physiological parameters from the PPG signal such as heart rate, blood pressure, blood oxigen saturation, and respiration [39, 79, 80]. Nevertheless, to the best of our knowledge no algorithms have been proposed so far for the compression of these signals. Note that with future application developments, besides the calculation of selected features or health indicators right on the

mobile devices, users or doctors may want to fully monitor the vitals, which could be sent to smartphones or control centers for further elaboration so as to provide a fine-grained assessment of the patient's condition, e.g., to assess the evolution or occurrence of a certain pathology. In this case, compressed but accurate representations of vital signals from heterogeneous sensor technology are expected to be very useful.

4.4.1 Fundamentals

The framework at the core of the proposed algorithm originates from two fundamental techniques which have already proven their effectiveness and whose capabilities has been exploited in various fields: *Motifs extraction* and *Vector Quantization*.

4.4.2 The concept of Motif

The concept of *Motif* originated from the field of time series *data mining* and *pattern recognition* to which the group of Keogh [81][82] contributed in a substatial way. The term (having as synonyms expressions such as *recurrent pattern* or *primitive shapes*) was taken from *computational biology* and aims at defining subportions of a signal which can be of variable dimension and appear with a certain degree of frequency (which, however, is expected to be high). In this section the basic concepts are given, which are necessary to understand what comes next.

The problem of efficiently finding a given pattern in a database was, in last decades, studied and analyzed to a great extent and can be considered, since long ago, a solved one. On the contrary, identifying the most recurrent patterns and their occurences in a given signal was not studied such deeply. A formal and complete definition of the problem is effectively given in [81], where an algorithm to discover the *Motifs* is proposed. The framework in [81] and it focuses, as we need, on time sequences. Given a time series of length N:

$$\mathbf{X} = \{x_1, x_2, ..., x_n, ..., x_N\}, \tag{4.3}$$

a subsequence, of length M ($M < N$, but usually $M << N$), of the given sequence is defined as

$$\mathbf{x} = \{x_m, x_{m+1}, ..., x_{m+M-1}\},\ \forall m \leq N - M + 1. \tag{4.4}$$

It is now clear that we are looking for those subsequences which occur with the highest frequency. To define the *occurrence* we account for a small difference among sebsequences, that is we define a function to formally express *dissimilarity*, $D(\mathbf{x}_i, \mathbf{x}_j)$ according to which we have a *match* if the so defined distance does not exceed a treshold ε:

$$D(\mathbf{x}_i, \mathbf{x}_j) \leq \varepsilon \Rightarrow \mathbf{x}_i \text{ match } \mathbf{x}_j \tag{4.5}$$

It si worth noting that this kind of *match* is a relation which enjoys the symmetric and reflexive properties, but not the transitive property.

Now, to discover the most frequent patterns, the authors of [81] define the *1-Motif*, $\mathbf{x}_1$, to be the pattern with the highest count of matches; the general *K-Motif*, $\forall K > 1$ is the pattern with the highest number of matches that also satisfy the relation $D(\mathbf{x}_i, \mathbf{x}_K) > 2\varepsilon,\ 1 < i < K - 1$: this condition is needed in order to correctly assign a match to a single *Motif*. Therefore the authors proceed to illustrate a brute-force algorithm to find the *1-Motif*.

4.4.3 Vector Quantization

The theory of Vector Quantization (VQ) can be found in the milestone by Gersho and Gray [83]. The theory comes from a generalization of scalar quantization to vectors, i.e. ordered sets of scalars. For what concern VQ, vectors can be formed by any sequence of samples of the signal, which can be, among others, a speech signal, a temperature dataset, an image or even a sequence of images, i.e. a video. First of all, we give some basic definitions upon which we will build the entire algorithm; we define a Vector Quantizer, Q, to be a mapping from the k-dimensional Euclidean space $\mathbb{R}^k$ to a finite set $\mathcal{C}$ containing N vectors (*codevectors* or *codewords*) of length k:

$$Q : \mathbb{R}^k \rightarrow \mathcal{C}. \tag{4.6}$$

The finite set $\mathcal{C}$, also called *codebook* or *dictionary*, contains vectors who also live in the Euclidean space $\mathbb{R}^k$ and can be formally defined as

$$\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N\}, \quad \mathbf{c}_i \in \mathbb{R}^k \quad \forall i \in \{1, 2, ..., N\}. \tag{4.7}$$

The codebook should be designed to be representative of the entire signal, i.e. the space where vectors live in (formally a subspace of $\mathbb{R}^k$), but it is used through the partition it induces on this space.

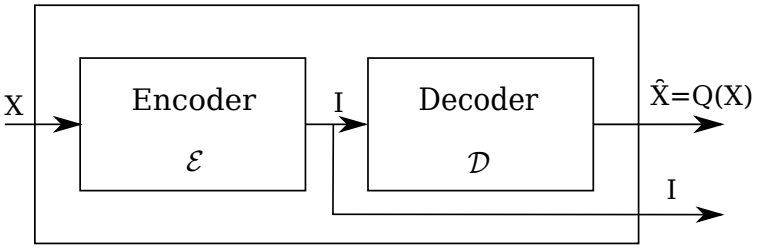


FIGURE 4.3: Encoder-Decoder structure of a Vector Quantizer

It could be more helpful to see the Vector Quantizer as a complete procedure including *encoding* and *decoding*. As shown in Fig. 4.3, the encoder, as already explained, associates to every input vector a codebook index, i; the decoder, on the other hand, uses this codebook index to retrieve the corresponding codeword which represents the input vector, $\hat{\mathbf{x}} = Q(\mathbf{x})$. The quantization error can thus be quantified through the distance between the input vector at time t, $\mathbf{x}_t$, and the assigned codeword $\mathbf{c}_{i\star}$: i.e. $D(\mathbf{x}_t, \mathbf{c}_{i\star}), \quad t \in \mathbb{R}_+$, where $c_{i\star} = Q(\mathbf{x}_t)$.

4.4.4 The time-invariant codebook

Having defined how a vector quantizer operates on input signals or, more correctly, on input vectors, it is time to describe how an efficient codebook can be built. The idea is to obtain a good representation of the input vectors using the smallest number possible of entries in the codebook. Such an optimization is obtained though a *partition* of the space where vectors live in, i.e. a set of N regions such that

$$\bigcup_i \mathcal{R}_i = \mathbb{R}^k, \quad \mathcal{R}_i \cap \mathcal{R}_j = \emptyset. \tag{4.8}$$

For each region a vector, or codeword, is chosen as representative for all the input vectors falling inside this region: this entails that, on the decoder side, this vector will be used instead of the input one. It is important to note, with reference to the encoder-decoder structure, that the partition of the space, the codebook, should be shared between the encoder and the decoder, i.e. it must be the same.

We recall, for future use, the definition of *convex set*: a convex set is a set in which, the points of any segment connecting any pair of points belong to the set too; thus a codebook formed by convex regions is said to be *regular* otherwise it is said to be *nonregular*. A subset of the regular quantizers, of particular interest, is constituted by the *polytopal* quantizers, where the regions, not only are convex subsets, but also *polytopes*. To quickly recall the polytope concept we highlight that polytopes are subsets of a space delimited by $(k-1)$-dimensional segments of hyper-planes.

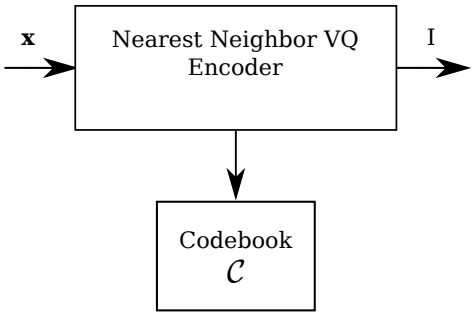


FIGURE 4.4: Structure of a Nearest Neighbor Vector Quantizer Encoder

The codebook construction will change if it is going to preserve a maximum error between input vectors and corresponding codewords or, else, if the requirement is for a maximum allowed number of codewords. The most common type of codebook design is the so-called *nearest neighbor VQ* or *Voronoi VQ*; in fact, very often, these types of VQs are simply referred to as *the* VQs. A Nearest Neighbor quantizer is formally defined as a VQ in which the cells are given by

$$\mathcal{R}_i = \{\, \mathbf{x} : d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j) \;\; \forall j \in \mathcal{I} \,\}, \tag{4.9}$$

where $\mathcal{I}$ is the set of indices labeling the cells. A simple adjustment permits to obtain the necessary mathematical correctness: when a vector lies exactly on a boundary between two cells, the one with the smallest index is chosen. In the Nearest Neighbor VQ any

computationally feasible distance measure $d(\cdot)$ con be used: anyway, the most used are the $k-$space usual norms such as L_2, L_1 or the L_∞-norm.

The structure of such a VQ relies completely on a codebook: in defining this codebook we can get rid of the regions and only store the codewords, given that we only need to compute a distance between them and the input vector: its structure can then be depicted as in Figure 4.4.

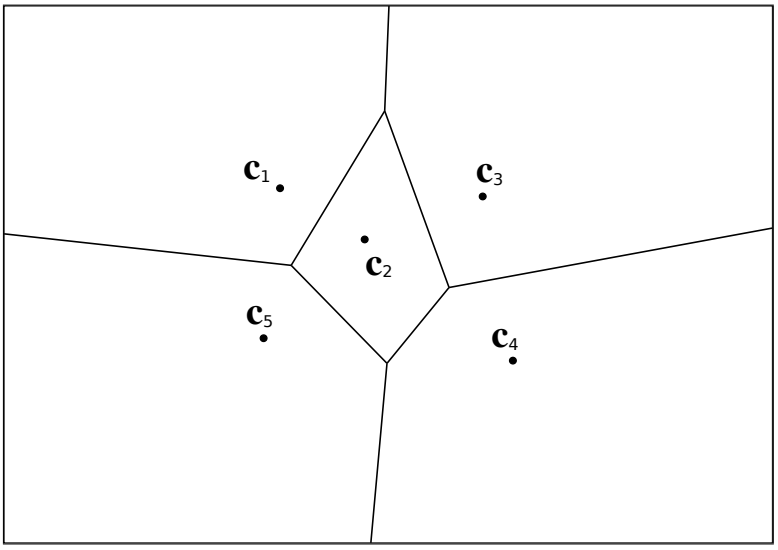


FIGURE 4.5: Partition given by a Nearest Neighbor VQ

If the Euclidean, or L_2, norm is used we can see that that the regions defined by a fixed set of codevectors are polytopes; in fact, we can rewrite the regions as an intersection of hyperplanes as

$$\mathcal{R}_i = \bigcap_{j, j \neq i} H_{ij}, \tag{4.10}$$

and the hyperplanes H_{ij} are, in this case:

$$H_{ij} = \{ \mathbf{x} : \|(\mathbf{x} - \mathbf{y}_i)\|^2 \leq \|(\mathbf{x} - \mathbf{y}_j)\|^2 \}. \tag{4.11}$$

An insight of how the parition is built is given by Figure 4.5, where the 2-dimensional case with 5 codevectors is repesented, and it is clear that a complete Voronoi diagram is generated.

4.4.5 RAZOR: Uniting Motifs and VQ

The work in [84][85], already exploits the two previous explained techniques, namely, Motifs extraction and Vector Quantization, to build an algorithm able to extract most frequent patterns, classify them and use this classification for compression in a Wireless Sensor Network (WSN): the approach has proven to be both lightweight in computational complexity and accurate in reconstruction.

The algorithm takes advantage of two important tools of VQ, namely *Shape-Gain* VQ and *Mean-Removed* VQ. These useful operations permit to obatin a normalized version of the input vectors, thus retaining only the important features. The two needed parameters, generally called *Offset(O)* and *Gain(G)*, are obtained from the input vector $\mathbf{x} = [x_1, x_2, ..., x_k,]$, through

$$O = \frac{\sum_{i=1}^{k} x_i}{k}, \quad G = \sqrt{\frac{\sum_{i=1}^{k} x_i^2}{k}}, \tag{4.12}$$

an thus the normalized version, $\bar{\mathbf{x}}$, of the vector $\mathbf{x}$ is given by

$$\bar{\mathbf{x}} = \frac{\mathbf{x} - O}{G}. \tag{4.13}$$

The vectors resulting from this normalizaton have thus zero mean and unit gain.

The algorithm specifies also the codebook build-up procedure, to be run on an initial portion, called *training set*, of the signal to be compressed: these initial samples are used to capture the behavior of the signal and then to construct a codebook optimized for that signal.

The resulting algorithm, called RAZOR, has two main parts: a first, in which the Motifs that will make up the codebook are selected, and a second one, in which, after the codebook has been shared with all the nodes, compression takes place.

Motif extraction First, a distance metric , $d(\cdot)$, is selected as in standard VQ, second a treshold d_{th}, on the distance, is set and finally a maximum allowed size of the codebook, K_{target}, is also set. From the training set sequence, of M_S samples, segments of fixed size k

are selected in this way:

$$S^{(i)} = [S_i, S_{i+1}, ..., S_{i+k-1}], \quad \forall i = 1, 2, ..., M_S - k + 1, \tag{4.14}$$

and the normalized version of each one is obtained, $\bar{S}^{(i)}$, as described earlier. A matrix, DM, is then computed, of which the generic element DM_{ij} is the distance, according to the selected distance measure, between the i-th and the j-th normalized segment:

$$DM_{ij} = d(\bar{S}^{(i)}, \bar{S}^{(j)}). \tag{4.15}$$

Algorithm 1 Codebook build-up procedure

```
1:  procedure RAZOR MOTIFS EXTRACTION
2:      for all m,n do
3:          MM_mn ← I(DM_mn < d_th)
4:      i ← 1
5:      while DM ≠ ∅ and k ≤ K_target do
6:          for all m do
7:              MC_m ← Σ_n MM_mn
8:          i ← arg max_m(MC_m)
9:          X(k) ← S̄^(i)
10:         Delete the l-th rows and columns in DM and MM for which MM_il = 1
11:         k ← k + 1
12:     K ← k
```

The Algorithm 1 is then applied, from which we obtain the codebook, $X(k), \ \ k = 1, 2, ..., K$, but also its size, that could be smaller than the limit K_{target}: this procedure is to be done on a high end machine, such as a *sink* operating in a WSN.

Encoding process After the codebook is determined, it is then shared among all the nodes before the proper encoding process, which is illustrated in Algorithm 2. The procedure simply takes as input vectors constituted by k signal samples and find which, among all the codebook entries, has the minimum distance from the current one, according to the selected dissimilarity measure.

Application of the algorithm proposed, as it is, has failed revealing what were the peculiarities and thus the needs of biomedical signals: the length of the pseudo-period does not let

Algorithm 2 VQ Compression

```
1: procedure RAZOR COMPRESSION
2:     d_min ← ∞
3:     for all X(k) ∈ CB do
4:         X_R(k) ← X(k)G + O
5:         if d(S^(i), X_R(k)) < d_min then
6:             X̂^(i) = X_R(k)
7:             d_min = d(S^(i), X_R(k))
8:     Return k
```

the procedure to build a dictionary (which has fixed size) able to be representative for the entire signal. In fact, the first part of the algorithm could, and should, be skipped given that we already know the recurrent patterns: at least where they begin and where they end.

4.5 Online Dictionary (OD) for biomedical signals

In this section, we propose a dictionary based compression algorithm based on the concept of *motif extraction* [86] and pattern recognition. Its building blocks are shown in Fig. 4.6 and explained in what follows. Although the scheme is simple (it consists of a single pass vector quantization without codeword reclustering) it provides excellent performance in the high compression regime and its analysis sheds some light on the desirable properties that a compression scheme should have, allowing the assessment of the pitfalls of offline dictionary based schemes and the identification of future research directions, as we discuss in next chapter.

The algorithm belongs to the inter-segment correlation class and can be applied to the biomedical signals exhibiting recurrent patterns such as ECG, photo-plethysmographic traces (PPG), arterial blood pressure (ABP), respiratory signals (RESP), etc. The idea is that recurrent patterns can be efficiently identified and used to construct, *at runtime*, a codebook (also referred to as dictionary). This codebook is built and maintained by the compressor at the transmitter side and has to be synchronized with that at the decompressor at the receiver. The compression of biosignals is achieved by sending, for each input pattern, the corresponding index in the codebook, in place of the original data points. We achieved

this through several processing functions, as shown in Fig. 4.6, namely: 1) a passband filter, 2) a peak detector, 3) a segment extractor, 4) pattern matching and 5) a codebook manager.

1) Passband filtering: as a first step, we use a passband filter to remove artifacts such as high frequency noise and the DC component. For ECG, this filter operates in the band $[8, 20]$ Hz, although these can be changed to best suit other signal types. Here, we implemented the third-order Butterworth filter of [87].

2) Peak detection: with this algorithm we detect the position of the main peaks in the time series. For ECG, these correspond to the heart beats. To this end, we have adopted the technique of [88], which has been conceived for ECG signals but can be easily modified to effectively work with PPG or respiratory traces. This technique is self-tuning and optimizes itself based on the input data sampling rate. We considered this scheme as it is fast and lightweight and thus suitable for use in wearable and energy constrained devices.

3) Segment extractor: once the peaks are detected, we consider the data samples between subsequent peaks. These constitute the input *segments* for our compressor algorithm. Note that, unlike the common practice of positioning the segments so that the peaks (heart beats) are in their center, we define a segment as the data points *between subsequent peaks*. Hence, all segments are normalized according to a predefined length of W samples, which is the same size of the codewords in the dictionary. This is accomplished by re-stretching the segment length to W samples through interpolation (this block is referred to as "period normalization" in Fig. 4.6). While in principle any interpolation technique can be used, such as quadratic or spline based, in our implementation we utilized a simple linear technique as we found it sufficiently accurate while also being computationally inexpensive. Working with such segments allows using machine learning algorithms for the construction of the codebook, as we detail shortly.

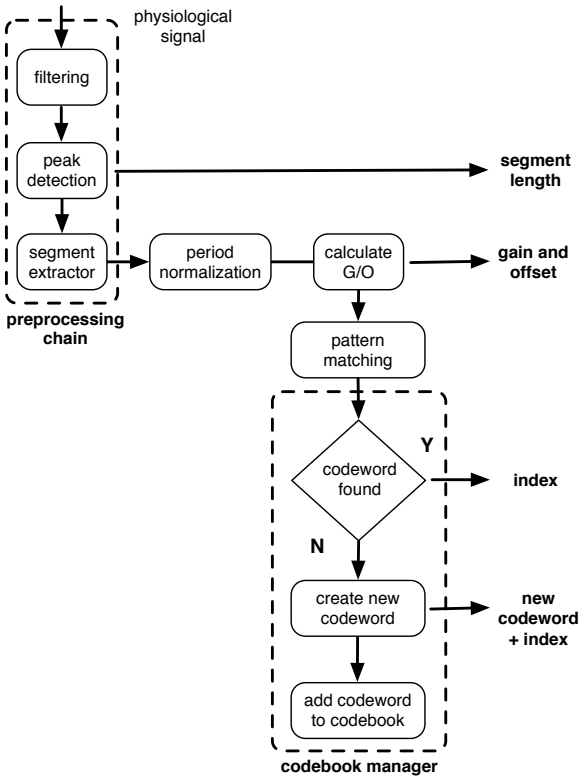


FIGURE 4.6: Online codebook-based compression scheme.

4) Pattern matching: this block takes the current input segment and checks whether this matches one of the codewords in the codebook (dictionary), which is built and maintained at runtime as we explain in point 5) below. Several matching criteria are possible. One of such criteria may be Dynamic Time Warping (DTW) [89], which has been extensively and successfully used in the literature to compare patterns of different length and can also be implemented in linear time [90]. However, we experimented with the DTW metric and we found it inadequate for ECG signals – the main problem is that this metric is by construction unable to preserve the position of the inner peaks in the compressed representations. Thus, in this work we resized each segment to a common length, as explained above, and checked for the best matching codeword using a suitable distance function, as we explain next.

5) Codebook manager: this block has a key role in the proposed online compression scheme. It is loosely based on *vector quantization* [91] and has two main functions: 1)

to *maintain* a consistent and representative codebook (dictionary) and 2) to *encode* input patterns into the corresponding indices from the codebook. Let $\boldsymbol{z}_t$ be the segment provided by the segment extraction block at the generic time $t = 0, 1, 2, \ldots$ (discrete time is assumed, corresponding to the arrival of a new segment). With $\mathcal{C}_t = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$ we indicate the codebook at time t, where $\boldsymbol{c}_i$, $i = 1, \ldots, N$, are the codewords therein. Segment $\boldsymbol{z}_t$ is remapped into a new segment $\boldsymbol{x}_t$ of length W samples as described above, where $\texttt{size}(\boldsymbol{c}_i) = \texttt{size}(\boldsymbol{x}_t) = W$, for $i = 1, \ldots, N$. The new segment $\boldsymbol{x}_t$ is obtained using linear resampling and removing offset o_t and gain g_t from $\boldsymbol{z}_t$ (see equations (5)–(7) of [86]). Thus, a suitable distance function $d(\boldsymbol{x}_t, \boldsymbol{c}_i)$ is evaluated for all codewords $\boldsymbol{c}_i$ in the codebook and the one with the minimum distance, with index $i^\star$, is picked. Now, if $d(\boldsymbol{x}_t, \boldsymbol{c}_{i^\star}) \leq \varepsilon$, codeword $\boldsymbol{c}_{i^\star}$ is deemed a good representative for the current segment $\boldsymbol{z}_t$, otherwise $\boldsymbol{x}_t$ is added to the codebook as a new codeword, where with $i^\star$ we mean the associated index. ε is a tunable parameter that we use to control the signal reconstruction fidelity at the decompressor. Finally, the index $i^\star$ is sent in place of the full segment, along with o_t, g_t and the original segment length, ℓ_t. The whole process is detailed in Fig. 4.6 (codebook manager block): if a match for $\boldsymbol{z}_t$ is found in the codebook (i.e., a codeword providing a sufficiently good accuracy, according to ε), then the corresponding index is sent over the transmission channel, along with the original segment length, its offset and gain parameters. These quantities correspond to the compressed bitstream, which is used at the decompressor to approximate the original time series by reversing each operation. Specifically, the decompressor applies three transforms to codeword $i^\star$ from the codebook: renormalization with respect to offset o_t and gain g_t and resampling according to the actual segment length ℓ_t. Otherwise, if no match is found for $\boldsymbol{z}_t$ at the compressor, this segment is added to the codebook as a new codeword and its normalized version (W samples) and the corresponding index are transmitted to the decoder so that the dictionary at the sender and that at the receiver remain synchronized at all times.

We remark that several distance functions can be used in the codebook manager, the L_∞-norm has been considered for the results in this chapter as it performed satisfactorily across a large range of signals.

According to our numerical results, as we show in Section 5.3 of Chapter 5, the number of codewords in the dictionary increases with decreasing ε but it tends to converge as time goes on. So the accuracy parameter ε also directly affects the dictionary size and, in turn, the memory requirements of the proposed algorithm. In case the codebook shall grow larger than the allowed memory space, the removal of codewords from the codebook can be implemented based on *last used* timestamps.

Chapter 5

Body sensor networks – a taxonomy of existing approaches and performance comparison against the Online Dictionary compression scheme

5.1 Introduction

In this chapter, we aim at continuing discussion of the previous chapter on biomedical signal compression. In the previous chapter, we have described the biosignal Compression in wearable devices and prposed a slightly new dictionary based method for an efficient compression. It is a novel compression architecture based on vector quantization and pattern recognition [47], where a suitable codebook (or dictionary) is built and maintained in an online fashion to efficiently represent data patterns. Compression is achieved as codebook

indices are sent to the decompressor in place of the original time series. Despite its simplicity, this technique is found to be appealing due to its excellent performance in the high compression regime.

Here we present the other selected algorithms from the literature which are detailed in Section 5.2 and a comparative performance evaluation of all the considered compression approaches is carried out in Section 5.3, where we quantify their compression efficiency, signal reconstruction fidelity and, most importantly, their energy consumption. Also, we estimate the energy savings due to the adoption of the discussed compression technology for continuous monitoring applications, which entail a longer battery life. Finally, our conclusions are presented in Section 5.4, along with a discussion of open research issues.

To summarize, the main contributions of this chapter are:

- A taxonomy of existing signal compression schemes that are amenable to implementation on wireless wearable IoT devices.
- A detailed performance evaluation of the considered compression schemes in terms of reconstruction error, energy consumption (isolating the energy required for compression and transmission) and compression efficiency when applied to ECG, RESP and PPG signals.
- A discussion of open areas for improvement and new research avenues.

5.2 Signal Compression Algorithms

In Chapter 4, we have presented a novel technique based on the online construction of a dictionary to represent input patterns. Here, we continue our discussion by detailing the selected signal compression algorithms for quasi-periodic biosignals from the literature. The compression methods that we describe below are based on differing paradigms. In fact, some use the degree of similarity (correlation) across subsequent patterns (referred to here as segments), whereas others consider the correlation within the same segment. We refer to the

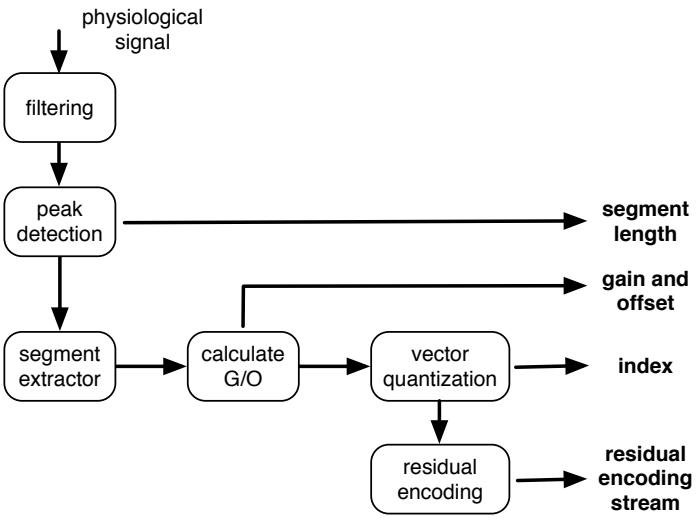


FIGURE 5.1: Diagram of the GSVQ compression technique.

former approach to as "inter-segment correlation" based compression, whereas for the latter we use the term "intra-segment correlation". The algorithms belonging to the inter-segment class are: online dictionary (which has been presented in the previous chapter), vector quantization and autoencoders, whereas algorithms based on principal component analysis, LTC, discrete cosine and wavelet transforms exploit intra-segment correlation properties. Two implementations of compressive sensing are considered, covering both classes.

5.2.1 Gain-Shape Vector Quantization (GSVQ)

In this section we review the Gain-Shape Vector Quantization (GSVQ) method of [75]. The rationale behind this algorithm is to exploit the information redundancy among adjacent heartbeats by segmenting the ECG signal into segments and normalizing the period to a fixed length and amplitude. The normalized heartbeats are then used to build a dictionary having a fixed number of codewords K, through the Linde-Buzo-Gray algorithm [92]. While the general compression principle (i.e., inter-segment correlation) is similar to that in our online dictionary based scheme, GSVQ builds the codebook through an *offline* training phase.

Once the dictionary is obtained, the method associates each normalized heartbeat with the closest codeword, and sends the codeword index in place of the original time series. The

algorithm also encodes the offset, the gain, and the length of the original segment, see Fig. 5.1. As a last step, the encoder calculates the residual, i.e., the difference between the current heartbeat (i.e., ECG segment) and the selected codeword, and uses the AREA algorithm [93], an adaptive sampling scheme for one dimensional signals, which obtains additional information to increase the quality of reconstruction. The principle behind the residual encoding phase is to encode and send a small number of significant points so as to bound the reconstruction error.

The decoder, upon receiving an encoded packet, retrieves the corresponding codeword from its local copy of the dictionary, performs a denormalization using the gain, the offset, and the length, and adds the residual stream to the reconstructed signal, see Fig. 5.1. As we shall see below, GSVQ performance predominantly depends on its residual encoding phase. The threshold used for residual encoding is in fact the main responsible for the amount of data to be transmitted, affecting the performance in terms of compression, reconstruction error, and energy efficiency.

5.2.2 Principal Component Analysis (PCA)

The goal of Principal Component Analysis (PCA) [41] is to shrink the information provided by a large set of correlated variables into a set of principal components with lower dimensionality. Each principal component is computed as a linear combination (linear transform) of the original variables, and the combination weights are chosen so that the components are mutually uncorrelated. This technique has been successfully applied in a multitude of applications, including ECG signal compression [71].

Before applying PCA, the biomedical signal goes through the preprocessing chain of Fig. 4.6, i.e., filtering, peak detection and segment extraction, where at time $t = 0, 1, 2, \ldots$ the last block normalizes each input segment $\boldsymbol{z}_t$ to a common length of W samples. The new segment is then stored into a vector $\boldsymbol{x}_t \in \mathbb{R}^W$ and is fed to the PCA encoder. Specifically, let $\boldsymbol{\mu_x} = E[\boldsymbol{x}_t]$ and $\boldsymbol{R_x} = E[\widetilde{\boldsymbol{x}}_t \widetilde{\boldsymbol{x}}_t^T]$ respectively be the mean of $\boldsymbol{x}_t$ and its covariance matrix, with $\widetilde{\boldsymbol{x}}_t = \boldsymbol{x}_t - \boldsymbol{\mu_x}$. PCA amounts to apply an orthonormal linear transformation

$\boldsymbol{\Psi} = [\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_W]$ to $\widetilde{\boldsymbol{x}}_t$, so that the elements $w_1, \ldots, w_W$ of the principal component vector $\boldsymbol{w} = \boldsymbol{\Psi}^T \widetilde{\boldsymbol{x}}_t = \boldsymbol{\Psi}^T(\boldsymbol{x}_t - \boldsymbol{\mu}_{\boldsymbol{x}})$ are mutually uncorrelated. It can be shown that the i-th principal component is obtained as $w_i = \boldsymbol{\psi}_i \widetilde{\boldsymbol{x}}_t$, where $\boldsymbol{\psi}_i$ is the eigenvector corresponding to the i-th largest eigenvalue of $\boldsymbol{R}_{\boldsymbol{x}}$, for $i = 1, \ldots, W$. The set of eigenvectors corresponding to the W principal components is obtained solving $\boldsymbol{R}_{\boldsymbol{x}}\boldsymbol{\Psi} = \boldsymbol{\Psi}\boldsymbol{\lambda}$ for $\boldsymbol{\Psi}$, where $\boldsymbol{\lambda}$ is a diagonal matrix containing the eigenvalues $\lambda_1, \ldots, \lambda_W$, placed in decreasing order. As the theoretical covariance matrix $\boldsymbol{R}_{\boldsymbol{x}}$ is difficult to compute, a matrix $\boldsymbol{X} \in \mathbb{R}^{W \times m}$ is built by stacking m successive ECG segments: their sample mean $\hat{\boldsymbol{\mu}}_{\boldsymbol{x}}$ and their sample covariance matrix $\hat{\boldsymbol{R}}_{\boldsymbol{x}} = (\boldsymbol{X}\boldsymbol{X}^T)/m \in \mathbb{R}^{W \times W}$ respectively replace $\boldsymbol{\mu}_{\boldsymbol{x}}$ and $\boldsymbol{R}_{\boldsymbol{x}}$ for the calculation of the eigenvectors.

According to the above discussion, we can write $\boldsymbol{x}_t = \boldsymbol{\mu}_{\boldsymbol{x}} + \boldsymbol{\Psi}\boldsymbol{w}$ and, if the signal is sufficiently correlated, only a fraction of the weights in $\boldsymbol{w}$ suffices to accurately describe the input vector $\boldsymbol{x}_t$. Compression is thus achieved by applying the PCA transform and sending the desired number h of principal components, i.e., the first h weights in $\boldsymbol{w}$, with $h \leq W$. In Section 5.2.3, we follow a similar rationale by using a particular neural network instance called autoencoder, which practically acts as a non-linear PCA [94].

5.2.3 Autoencoders (AE)

An autoencoder [95] is a neural network where input and output layers have the same dimension W, whereas the deepest hidden layer has a smaller dimension h, with $h < W$, as we show in Fig. 5.2. With $w_{ij}^{(1)}$ ($w_{ij}^{(2)}$) we indicate the autoencoder weights from neuron i to neuron j of the input (output) layer. Here, autoencoders are used as a non-linear dimensionality reduction technique to compactly represent the information in the original segments (of size W) into a much smaller space (ideally $h \ll W$ neurons).

The training of this neural network is accomplished through an unsupervised learning algorithm that uses a number of training examples $\boldsymbol{x} \in \mathbb{R}^W$ that are placed at the input of the autoencoder. Specifically, backpropagation is executed by setting the output $\boldsymbol{y} = \boldsymbol{x}$ so that the neural network weights $w_{ij}^{(1)}, w_{ij}^{(2)}$ are adjusted for the autoencoder to behave as

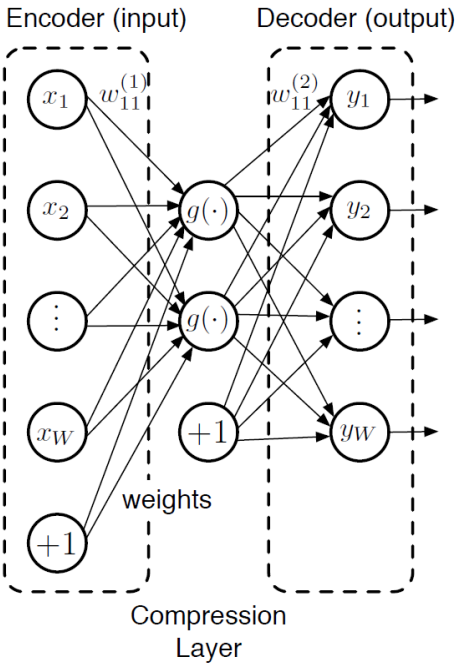


FIGURE 5.2: Graphical representation of an autoencoder: input and output layers have the same dimension W, whereas the compression layer has $h = 2$ neurons. $g(\cdot) : \mathbb{R} \to \mathbb{R}$ is assumed to be the logistic activation function $g(z) = (1 + \exp(-z))^{-1}$.

an identity function. In this work, we consider the approach of [46] where the authors use denoising autoencoders [96] to approximate the input biomedical patterns.

Once the autoencoder is trained to represent the input data, weights $w_{ij}^{(1)}$ fully specify the compressor (encoder), whereas $w_{ij}^{(2)}$ specify the decompressor (decoder), see Fig. 5.2. Signal compression is achieved by applying the preprocessing chain of Fig. 4.6, i.e., filtering, peak detection and segment extraction. Note that the last block also normalizes each segment to a common length of W samples. Each of such segments is inputted to the encoder section of the autoencoder, which returns the h values associated with the neurons in the compression layer. These h values correspond to the compressed representation of the current segment and are sent to the decompressor along with the original segment length. Finally, the decompressor at the receiver uses the values of these h inner neurons, along with weights $w_{ij}^{(2)}$, to obtain the reconstructed W-sample vector $\boldsymbol{y}$ through the decoder of Fig. 5.2. Vector $\boldsymbol{y}$ is thus resized to the original segment length. We remark that AE also belongs to the inter-segment correlation class of algorithms as it exploits the fact that patterns across different segments have a quasi-periodic behavior.

5.2.4 Compressive Sensing (CS)

Compressive sensing (CS) is a recently proposed theory [97][98] to efficiently acquire and reconstruct a signal, by solving ill-posed linear systems of equations. This technique is based upon the premise that the signal of interest is sparse in some transform domain. This means that, the original signal can be represented in a domain where only a few transform coefficients are required for its full description. To be more specific, let $\boldsymbol{x} \in \mathbb{R}^W$ be an W-sized vector and assume that this vector can be represented in a K-sparse domain through the sparse vector $\boldsymbol{s}$, where only $K \ll W$ elements of $\boldsymbol{s}$ are non-zero, i.e., vector $\boldsymbol{s}$ is K-sparse in this domain. If we refer to the sparsification basis as $\boldsymbol{\Psi} \in \mathbb{R}^{W \times W}$, we have that $\boldsymbol{x} = \boldsymbol{\Psi s}$. Now, let $\boldsymbol{\Phi} \in \mathbb{R}^{m \times W}$ be a sampling matrix. Note that, using this matrix to sense the full signal $\boldsymbol{x}$, we have $\boldsymbol{y} = \boldsymbol{\Phi x} + \boldsymbol{n}$, where $\boldsymbol{n} \in \mathbb{R}^m$ represents the measurement noise, $\boldsymbol{y} \in \mathbb{R}^m$ and $m < W$, which means that $\boldsymbol{x}$ is being subsampled.

CS tools allow the recovery of $\boldsymbol{x}$ from its subsampled version $\boldsymbol{y}$, where: $\boldsymbol{y} = \boldsymbol{\Phi x} + \boldsymbol{n} = \boldsymbol{\Phi \Psi s} + \boldsymbol{n}$. This is achieved solving for $\boldsymbol{s}$ the following equation:

$$\min \|\boldsymbol{s}\|_1 \quad \text{s.t.} \quad \|\boldsymbol{y} - \boldsymbol{\Phi \Psi s}\|_2 \leq \epsilon \,, \tag{5.1}$$

where ϵ represents a bound on the measurement noise. Numerically, a high number of techniques are available to solve (5.1); among them we cite ℓ_1-magic [18] subspace pursuit [19] and NESTA [20]. In this work, we consider two recent ECG compression frameworks from [44] and [76], which are based on CS. At the encoder, they exploit a standard CS matrix multiplication (sampling and sparsification), whereas at the decompressor the former exploits a technique called Simultaneous Orthogonal Matching Pursuit (SOMP) [44], whereas the latter uses Block Sparse Bayesian Learning (BSBL) [99]. The algorithms are discussed next.

5.2.4.1 Simultaneous Orthogonal Matching Pursuit (SOMP-CS)

this techniques first splits the ECG signal into a number of segments and then applies standard CS-sampling to R consecutive segments at a time. Recovery is based on Orthogonal Matching Pursuit.

SOMP-CS encoder:

- **Peak detection:** similarly to codebook-based schemes, a peak detection method is applied to the input signal to decompose it into segments $\boldsymbol{x}_t$, $t = 0, 1, 2, \ldots$.
- **Period normalization:** each segment $\boldsymbol{x}_t$ is normalized to a common length (W samples) using cubic-spline interpolation.
- **Sampling:** each R consecutive ECG segments are stored into a $W \times R$ matrix $\boldsymbol{X}$. A CS sampled matrix $\boldsymbol{Y}$ is then obtained as $\boldsymbol{Y} = \boldsymbol{\Phi X}$, where $\boldsymbol{\Phi} \in \mathbb{R}^{m \times W}$ is a suitable sampling matrix, with $m \ll W$. As assumed in [44], for matrix $\boldsymbol{\Phi}$ we use a dense Gaussian matrix (each element is independently sampled from a Gaussian pdf with zero mean and variance $1/m$, i.e., $\mathcal{N}(0, 1/m)$). $\boldsymbol{Y}$ and the corresponding original lengths are quantized and sent to the decoder. Note that this implementation of CS belongs to both the inter- and the intra-segment class as matrix $\boldsymbol{Y}$ spans across different adjoining segments.

Note that the CS encoder is extremely lightweight as it just implies the multiplication of the input time series by the sampling matrix $\boldsymbol{\Phi}$. The most computation intensive tasks are period normalization and peak detection, which are needed in all segment-based approaches. Under the assumption that the source data $\boldsymbol{X}$ can be rewritten as: $\boldsymbol{X} = \boldsymbol{\Psi S}$, where the matrix $\boldsymbol{S} \in \mathbb{R}^{W \times R}$ is sparse and the sparsification transform $\boldsymbol{\Psi}$ is the Daubechies wavelet db4 [100], the original data $\boldsymbol{X}$ can be retrieved solving problem (5.1) using Simultaneous Orthogonal Matching Pursuit. In our implementation, we have exploited the method in [101]

and the Matlab Uvi Wave tool [102] to represent the wavelet transform into an equivalent matrix form.

SOMP-CS decoder:

- **Simultaneous Orthogonal Matching Pursuit:** each segment is recovered from $\boldsymbol{Y}$ using the modified Simultaneous Orthogonal Matching Pursuit (with partially known support) of [103], which exploits the structure of the wavelet coefficients (through the knowledge of the support). This method solves for $\boldsymbol{S}$ the ill-posed system $\boldsymbol{Y} = \boldsymbol{\Phi\Psi S}$. Upon recovering $\boldsymbol{S}$, the original data is approximated through $\hat{\boldsymbol{X}} = \boldsymbol{\Psi S}$.
- **Period Recovery:** the reconstructed segments are re-interpolated to their original lengths.

Note that SOMP-CS considers a number R of subsequent segments ($R = 6$ in our implementation) and, in turn, also accounts for the "inter-segment" correlation structure of the ECG signal.

5.2.4.2 Block Sparse Bayesian Learning (BSBL)

BSBL exploits the fact that the ECG signal $\boldsymbol{x}$ is already sparse in the temporal domain, being composed of peaks spaced apart by an almost-flat signal. Hence, the input ECG signal is written as $\boldsymbol{y} = \boldsymbol{\Phi x} + \boldsymbol{n}$, where $\boldsymbol{y} \in \mathbb{R}^m$ is the compressed vector, $\boldsymbol{\Phi} \in \mathbb{R}^{m \times W}$ is a suitable sampling matrix ($m \ll W$), $\boldsymbol{x} \in \mathbb{R}^W$ is a sparse vector and $\boldsymbol{n} \in \mathbb{R}^m$ is the noise vector. Generally, vector $\boldsymbol{x}$ has additional structure and can be further represented as a concatenation of a certain number g of blocks x_i, possibly having different length d_i so that $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_g)^T$. Each block $\boldsymbol{x}_i \in \mathbb{R}^{d_i}$, $i = 1, \ldots, g$, is assumed to satisfy a parametrized multivariate Gaussian distribution $p(\boldsymbol{x}_i, \gamma_i, \boldsymbol{B}_i) \sim \mathcal{N}(0, \gamma_i \boldsymbol{B}_i)$ with the unknown parameters γ_i and $\boldsymbol{B}_i$. $\gamma_i \geq 0$ controls the block-sparsity of $\boldsymbol{x}_i$ and when $\gamma_i = 0$ the

i-th block becomes the all zero vector. Matrix $\boldsymbol{B}_i \in \mathbb{R}^{d_i \times d_i}$ is a positive definite matrix which captures the correlation structure within the i-th block. Assuming that the sub-blocks $\boldsymbol{x}_i$ are uncorrelated the prior of $\boldsymbol{x}$ is $p(\boldsymbol{x}, \{\gamma_i, \boldsymbol{B}_i\}) \sim \mathcal{N}(0, \boldsymbol{\Sigma}_0)$, where $\boldsymbol{\Sigma}_0 = \text{diag}\{\gamma_1 \boldsymbol{B}_1, \ldots, \gamma_g \boldsymbol{B}_g\}$. For the noise, it is assumed that $p(\boldsymbol{n}, \lambda) \sim \mathcal{N}(0, \lambda \boldsymbol{I})$, where $\lambda \in \mathbb{R}^+$ and $\boldsymbol{I} \in \mathbb{R}^{m \times m}$ is the identity matrix. The posterior of $\boldsymbol{x}$ (given the measured vector $\boldsymbol{y}$) is thus obtained as

$$p(\boldsymbol{x}|\boldsymbol{y}; \{\gamma_i, \boldsymbol{B}_i\}_{i=1}^g) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\Sigma}_{\boldsymbol{x}}) \tag{5.2}$$

where $\boldsymbol{\mu}_{\boldsymbol{x}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{x}}$ can be readily derived from λ, $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\Phi}$. Finally, the Maximum-A-Posteriori (MAP) estimate of $\boldsymbol{x}$, denoted by $\hat{\boldsymbol{x}}$, is given by [99]:

$$\hat{\boldsymbol{x}} = \boldsymbol{\Sigma}_0 (\boldsymbol{\Phi})^T \left[\lambda \boldsymbol{I} + \boldsymbol{\Phi} \boldsymbol{\Sigma}_0 (\boldsymbol{\Phi})^T\right]^{-1} . \tag{5.3}$$

Thus, the problem boils down to the estimation of the parameters λ and $\{\gamma_i, \boldsymbol{B}_i\}_{i=1}^g$. This is achieved using a Type II maximum likelihood procedure. Moreover, different techniques have been developed according to whether the block partition is known or not [99].

BSBL encoder: the ECG signal is split into a number of segments $\boldsymbol{x}$, each of which consists of W samples, where W is a tunable parameter not necessarily representing the number of samples in a segment. What the encoder does is to compute $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}$, which only entails a matrix multiplication. In our results, $\boldsymbol{\Phi}$ is the 0/1 matrix that was used in [76, 104].

We remark that the encoder is extremely lightweight as it does not have to split the ECG trace into segments, so peak detection and period normalization are not executed.

BSBL decoder: the decoder operates according to the above Bayesian estimation / maximum likelihood approach, see Eq. (5.3). Typical values for m and W are $m = 256$ and $W = 512$ [76] and, in turn, the maximum compression efficiency is given by $W/m = 2$ (in Section 5.3, we experiment with different (m, W) pairs).

We observe that BSBL accounts for the intra-block correlation without considering the correlation structure among subsequent ECG segments. We thus classify BSBL as an "intra-segment" compression scheme. For SOMP-CS, we have written our own encoder/decoder pair, whereas for BSBL-CS we used the code provided by the authors of [76]. The numerical results are discussed in Section 5.3.

5.2.5 Discrere Cosine Transform (DCT)

In the signal compression field, Discrete Cosine Transform (DCT) is often preferred to the Fourier Transform due to its superior energy compaction capabilities and the fact that it entails the use of real coefficients. Several ECG compression methods exploiting DCT have been proposed in the literature [105–110]. Basically, in all of the proposed algorithms DCT is used to reduce the amount of data to be sent through the transmission of a subset of transform coefficients, i.e., those which carry more information. Some solutions employ advanced techniques for the pre/post processing of the DCT coefficients that, however, for wearable devices are expected to be energetically prohibitive.

In this chapter, we consider two DCT based compression methods that differ in the adopted coefficient selection approach:

- **DCT-Cardinality Thresholding:** with this selection method the number of coefficients to be retained is given as input, and the coefficients are added starting from the lowest frequencies, i.e., the leftmost coefficient. Through this strategy the compression ratio can be finely tuned, but there are no guarantees on the reconstruction error at the decompressor.
- **DCT-Energy Thresholding:** with this method the coefficients are selected so as to meet an energy threshold constraint. The total energy of the DCT spectrum, E, is calculated and the coefficients that contain a predetermined fraction E_{th} of this energy are kept. The coefficients are selected again from the lowest to the highest frequencies, exploiting the energy compaction property of the DCT, so that their frequency position does not have to be encoded.

5.2.6 Discrere Wavelet Transform (DWT)

LTC [40] is a fast linear approximation technique working as follows. Let $z[n]$, $n = 0, 1, 2, \ldots$ be the input time series. The algorithm starts selecting $z[0]$ as the left endpoint of the current approximating segment. The following points $z[n]$ with $n > 0$ are transformed into vertical intervals $[z[n] - \varepsilon, z[n] + \varepsilon]$ where $\varepsilon > 0$ is an error tolerance on the reconstructed signal. When point $n > 1$ is considered, LTC evaluates the segment with extremes $(z[0], z[n])$ and checks whether this segments falls within *each* of the previously obtained vertical intervals around $z[1], z[2], \ldots, z[n-1]$. If this is the case, the algorithm obtains the vertical interval for the current point n and performs the check for the next point $n + 1$. Otherwise, the algorithm stops, taking $z[n-1]$ as the right endpoint of the current segment. Thus, 1) $z[0]$ and $z[n-1]$ are sent as the left and right endpoints of the current segment as an approximation to values $\{z[0], z[1], \ldots, z[n-2], z[n-1]\}$ and 2) the algorithm reiterates with a new approximating segment, taking $z[n-1]$ as its left endpoint.

5.2.7 Lightweight Temporal Compression (LTC)

LTC [40] is a fast linear approximation technique working as follows. Let $z[n]$, $n = 0, 1, 2, \ldots$ be the input time series. The algorithm starts selecting $z[0]$ as the left endpoint of the current approximating segment. The following points $z[n]$ with $n > 0$ are transformed into vertical intervals $[z[n] - \varepsilon, z[n] + \varepsilon]$ where $\varepsilon > 0$ is an error tolerance on the reconstructed signal. When point $n > 1$ is considered, LTC evaluates the segment with extremes $(z[0], z[n])$ and checks whether this segments falls within *each* of the previously obtained vertical intervals around $z[1], z[2], \ldots, z[n-1]$. If this is the case, the algorithm obtains the vertical interval for the current point n and performs the check for the next point $n + 1$. Otherwise, the algorithm stops, taking $z[n-1]$ as the right endpoint of the current segment. Thus, 1) $z[0]$ and $z[n-1]$ are sent as the left and right endpoints of the current segment as an approximation to values $\{z[0], z[1], \ldots, z[n-2], z[n-1]\}$ and 2) the algorithm reiterates with a new approximating segment, taking $z[n-1]$ as its left endpoint.

5.3 Results

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

For the energy consumption, following the approach of [111, 112] we compute three metrics: 1) the energy consumption for the execution of the compression algorithms in the node (termed *compression energy*), 2) the energy drained by the the transmission of the (either compressed or original) signal over a wireless channel (*transmission energy*) and 3) the total energy, which is given by the sum of the previous two metrics. 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. These were then translated into the corresponding number of MCU cycles and, in turn, into the energy consumption in Joule per bit considering a Cortex M4 [113] processor, see also [111]. For the transmission energy, we took a Texas Instruments CC2541 low-energy Bluetooth system-on-chip [114], which is widely adopted for IoT devices.

The **Compression Efficiency (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. For the reconstruction fidelity, we computed the **Root Mean Square Error (RMSE)** between the original and the compressed signals normalizing it with respect to the signal's peak-to-peak amplitude, that is:

$$\text{RMSE} = \frac{100}{\text{p2p}} \sqrt{\frac{\sum_{i=1}^{L} (x_i - \hat{x}_i)^2}{L}}\,, \tag{5.4}$$

where L corresponds to the total number of samples in the trace, x_i and $\hat{x}_i$ are the original sample and the one reconstructed after the decompressor in position i, respectively. p2p is the average peak-to-peak signal's amplitude. We observe that other metrics such as the Percentage Root mean square Difference (PRD) are also possible. As pointed out in [115], PRD can mask the real performance of compression algorithms since it depends on the

mean value of the original signal, whilst our RMSE metric allows one to immediately gauge the error against the signal's range. For this reason, we use (6.15) as our preferred metric throughout the chapter.

In Section 5.3.1 we first assess the performance of the considered compression algorithms for the standard test ECG traces from the PhysioNet MIT-BIH arrhythmia database [116]. In Section 5.3.2, we extend our analysis to ECG traces that we collected from a Zephyr BioHarness 3 wearable chest strap. In Section 5.3.3, we consider PPG and RESP signals.

5.3.1 PhysioNet ECG traces

In the first set of graphs, we show results for ECG signals. To this end, we considered the following traces from the MIT-BIH arrhythmia database [116]: 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 of 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 [116, 117]. The above performance metrics were obtained for these traces and their average values are shown in the following plots.

In Figs. 5.3, 5.4 and 5.5 we show the RMSE *vs* CE performance for all compression algorithms. Fig. 5.3 shows the performance of standard compression approaches, namely, DCT, DWT and linear approximation (LTC), Fig. 5.4 presents that of the codebook-based schemes (GSVQ and OD), whereas in Fig. 5.5 we show results for dimensionality reduction algorithms, namely, BSBL-CS, SOMP-CS, PCA and AE. Our online dictionary (OD) scheme is plotted in all figures for comparison. The tradeoff curves of OD have been obtained by varying the representation accuracy ε as a free parameter.

In Fig. 5.3, we consider the energy thresholding version of DCT (DCT-ET). We also experimented with its cardinality thresholding (CT) variant and we found its performance to be very similar to that of DCT-ET in every respect (RMSE, compression efficiency and energy). Thus, implementation convenience will dictate which of the two variants is to be

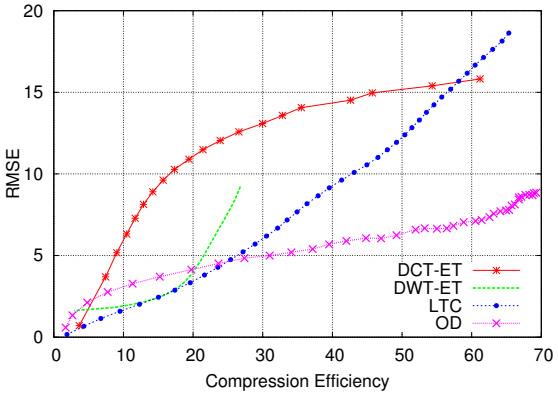


FIGURE 5.3: RMSE *vs* compression efficiency for ECG signals: DCT, DWT, LTC and OD.

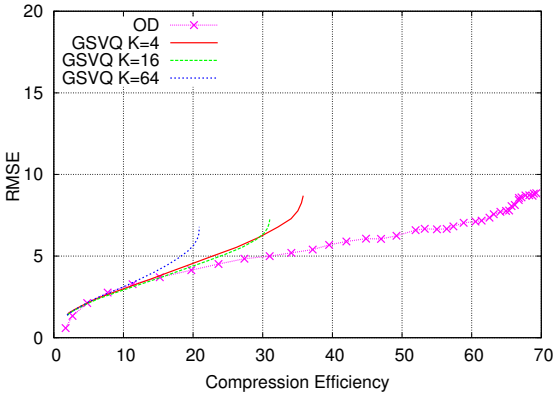


FIGURE 5.4: RMSE *vs* compression efficiency for ECG signals – comparison of codebook-based compression schemes: GSVQ and OD. K is the (fixed) size of the GSVQ dictionary.

preferred. LTC outperforms DCT-ET in terms of RMSE and CE; although, we remark that this is not always the case. For example, in [73] a DCT implementation that considerably surpasses LTC in terms of RMSE is proposed, but this comes at the price of a much higher computational complexity. This is possible through a more sophisticated selection of the coefficients, which requires performing inverse transforms for every ECG segment. This DCT variant is however not considered here as it is not deemed appropriate for wearable devices, due to its high computational cost. 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 compression efficiencies, for which LTC and OD are to be preferred. At small compression efficiencies, adaptive algorithms may be a valuable option – for instance, one may switch between LTC and OD as a function of the required compression level. For OD, we also look at the number of codewords in the dictionary as a function of

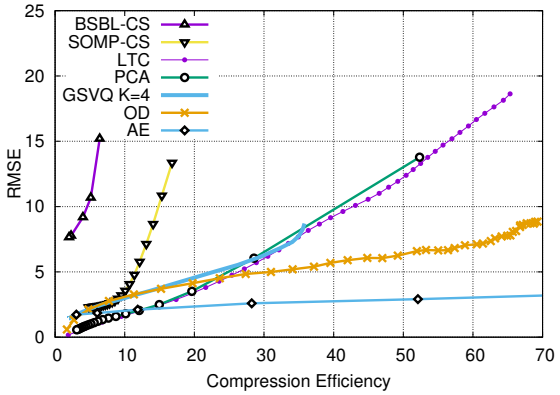


FIGURE 5.5: RMSE*vs* compression efficiency for ECG signals: BSBL-CS, SOMP-CS, PCA, LTC, AE and OD.

the compression efficiency, see Fig. 5.6. From that plot, we see that using OD is especially convenient at high compression efficiencies, i.e., higher than 45. In this region, the size of the dictionary is in fact reasonably small (smaller than 35 codewords) and it is thus feasible to store it in the limited memory of wearables. Specifically, for the considered setup the size of each codeword is $(W \times 11)/8 = 275$ bytes, where $W = 200$ is the codeword length [1] and 11 is the number of bits to represent the signal samples from the MIT-BIH database. This means that a dictionary of 35 codewords takes $35 \times 275 = 9.625$ kbytes of memory space.

A comparison for the codebook-based algorithms is presented in Fig. 5.4. For GSVQ we move along the RMSE *vs* CE curves by changing the threshold that governs the number of bits that are encoded into the residual stream. As discussed in Section 5.2.1, residual encoding is the operation that affects the most the performance of GSVQ. The dictionary size K affects the maximum achievable compression but the maximum CE is always smaller than that of OD, where the dictionary adapts to the signal in an online fashion. Although not shown in the plot, one may be thinking 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 recommended.

[1] A codeword represents one full ECG segment. We experimented with different values of W, varying it from 150 to 350. In the considered databases, the number of samples in an ECG segment ranges between 190 and 318. The value $W = 200$ was chosen as it leads to a good tradeoff between accuracy, complexity and compression efficiency.

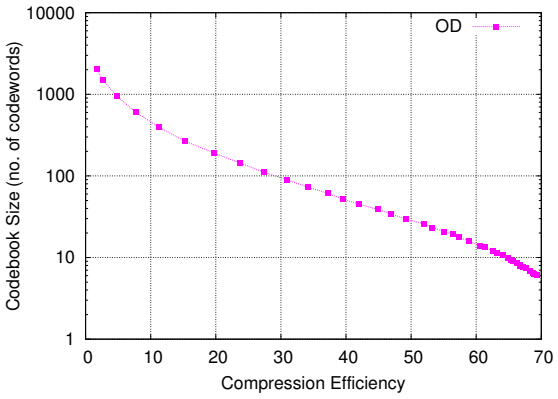


FIGURE 5.6: Online dictionary compression: codebook size as a function of the compression efficiency. The tradeoff curve is obtained by varying the representation accuracy parameter ε.

PCA is shown in Fig. 5.5. From this graph we see that the performance of PCA closely matches that of LTC, which is plotted in the same figure for the sake of comparison. This is quite interesting and non trivial – although both algorithms rely on linear approximations, PCA is rather involved, whereas LTC has a much lighter computational cost, as we show shortly. Also, in Fig. 5.5 the tradeoff curve for PCA is obtained by varying the number of principal components h from 100 (leftmost point in the figure) down to 5 (rightmost point) in steps of 5, whereas the performance of LTC is plotted varying ε within a continuous interval. Overall, LTC permits a fine-grained control of the RMSE *vs* compression tradeoff, whereas this is not possible with PCA, especially at high compression efficiencies (small h). Finally, LTC provides a means to precisely control the maximum reconstruction error, through the parameter ε, whereas the number of retained principal components h does not provide any guarantee in terms of reconstruction accuracy.[2]

AE is shown in Fig. 5.5, where the number of inner neurons h is varied as a free parameter in $\{100, 50, 25, 10, 5, 2\}$: $h = 100$ is represented by the leftmost point in the graph, whereas the rightmost one corresponds to $h = 2$. AE obtains the best performance both in terms of RMSE and CE. We underline that this algorithm entails an *offline* training phase which has two drawbacks: 1) usually, this phase is computationally demanding and requires a representative dataset, 2) although autoencoders have excellent generalization capabilities,

[2]With PCA, an inverse transform at the compressor is required to assess the reconstruction error provided by a certain value of h.

TABLE 5.1: Average complexity [no. operations] and energy consumption [μJ] per ECG segment. RMSE is 7.5% for all algorithms except AE for which the average RMSE is 3.75% (the highest with AE, obtained with $h = 2$ neurons in the inner compression layer).

	DCT-ET	**DWT-ET**	**GSVQ**	**BSBL-CS**	**SOMP-CS**	**LTC**	**OD**	**AE**
Additions	13463	7809.9	98114	3747	97302	1258.8	84411	82439
Multiplications	9089.8	6883.5	82788	0	95805	0	82817	81535.7
Divisions	0	323.2	1137.2	0	1045	634.1	940.2	938.6
Comparisons	30.4	7723.1	475.6	0	522	1231.1	987	462.7
Compression energy	0.74	0.88	6.47	**0.12**	6.84	0.37	5.95	5.83
TX energy	124.22	45.85	35.82	639.03	272.36	37.90	20.37	**13.45**
Total energy	124.96	46.73	42.29	639.15	279.20	38.27	26.32	**19.28**

if the statistics of the underlying signal changes substantially, there are no guarantees that autoencoders trained with the old data will still provide good approximations for the new signals. However, the RMSE performance achieved by AE is striking and spurs the use of neural networks within this domain. A note on the AE compression efficiency is in order. For OD the compressed bitstream comprises the following fields, which are sent for each new ECG segment: the codeword index, the original segment length, the gain and the offset. Thus, when no updates of the dictionary occur, 4 parameters are to be sent for each new segment. The maximum compression efficiency of OD is thus obtained as[3] $CE_{\text{OD}}^{\max} = \texttt{SamplesPerSeg}/4$, where `SamplesPerSeg` corresponds to the number of samples in a segment. With AE, for each segment we only send the h values associated with the inner neurons, see Fig. 5.2, and the length of the original segment. Two additional offsets are sent only once, when the compression starts. Thus, the maximum CE is approximated as $CE_{\text{AE}}^{\max} \simeq \texttt{SamplesPerSeg}/(h+1)$. For an average segment size of 318 samples, we get $CE_{\text{OD}}^{\max} = 80$ and $CE_{\text{AE}}^{\max} = 106$, which explain the results in Fig. 5.2. Note that the compression efficiency of OD is actually smaller than 80 and this is due to the new codewords that must be sent to update the dictionary at the decompressor.

For AE, the memory occupation is fixed and amounts to the memory needed to store the weights of the encoder in Fig. 5.2 and the output values generated by the inner layer, i.e., $((W+1) \times h \times 11)/8$ bytes, where $W = 200$, h is the number of inner neurons and 11 is the number of bits to represent a floating point (either a weight or an encoder output). Given this, with, e.g., $h = 8$ the memory footprint of AE is 2.211 kbytes.

[3]For the sake of clarity, we assume that input samples and parameters are represented through the same number of bits. If this is not the case, the following equation should consider the different precision.

TABLE 5.2: Energy breakdown [no. operations] and energy consumption [μJ] for the OD processing blocks. RMSE is 7.5%.

	Pass band filtering	**Peak detection**	**Segment extractor**	**Pattern matching**	**Codebook manager**	**Total**
Additions	4373	78723.8	396	399.8	518.3	84411
Multiplications	4061	78099	198	200	259	82817
Divisions	312.2	625	1	2	0	940.2
Comparisons	0	468.6	0	0	518.4	987
Compression energy	0.42	5.45	0.02	0.021	0.043	5.95

As for the CS-based algorithms, neither SOMP-CS nor BSBL-CS provides satisfactory performance. The compression efficiency of SOMP-CS is rather small and the corresponding RMSE tends to diverge for, e.g., CE larger than 5. As we shall see shortly below, the overall energy performance of SOMP-CS is unsatisfactory when compared to that of other algorithms and the compression strategy of BSBL-CS has the lowest energy consumption, but its intra-segment approach is less effective in terms of CE than that of other inter-segment schemes such as dictionary based algorithms (GSVQ, OD) and AE. Although the results that we show here for SOMP-CS and BSBL-CS were respectively obtained using the theory from [44] and [76], we found similar CE figures in other papers [45]. In these studies, the compression efficiency is defined as $\text{CE}' = ((W - m)/W) \times 100$, with W being the number of original samples and m the number of compressed samples that are transmitted to the receiver. With this definition, CS schemes achieve maximum efficiencies of 80-90%. We observe that these figures correspond to a CE ranging from 5 to 10 according to the definition that we use in the present chapter, i.e., $\text{CE} = W/m$.

In Fig. 5.7, we show the RMSE and the energy drained for compression 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 1 to the maximum achievable (which is scheme-specific, see Fig. 5.9). The RMSE increases with an increasing compression efficiency, whereas the compression energy depends weakly on CE. As expected, BSBL-CS has the smallest energy consumption. LTC is the second best, whereas OD, SOMP-CS and AE perform very close to one another and have the worst energy consumption for compression. The good performance of BSBL-CS is due to its lightweight compression algorithm, which just multiplies the input signal by sparse binary matrices, with entries in $\{0, 1\}$. We underline that the energy consumption of OD, SOMP-CS and AE is dominated

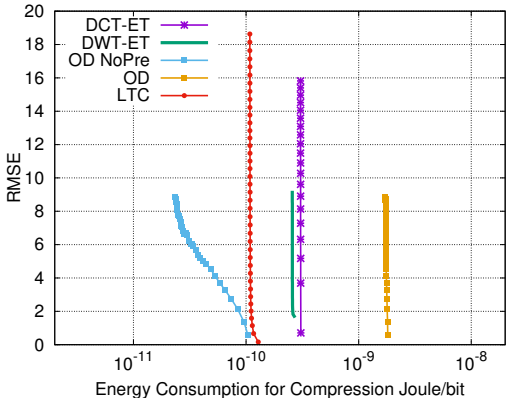


FIGURE 5.7: RMSE *vs* compression energy obtained varying CE as the independent parameter.

by the preprocessing chain of Fig. 4.6 (as we quantify below through the measurements in Table 5.2). In this plot, we also show the performance of OD and AE by removing the contribution of the pre-processing blocks: the corresponding curves are referred to in the plot as "OD NoPre" and "AE NoPre", respectively. 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. Although not shown for the sake of readability of the plot, PCA and GSVQ have nearly the same energy consumption of OD.

In Fig. 5.8, we show the RMSE as a function of the *total energy consumption*, which is obtained summing the energy required for compression to that for the subsequent transmission of the compressed bitstream over a CC2541 Bluetooth low-energy wireless interface. 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, OD and AE 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 providing the smallest RMSE is used. In Fig. 5.8, the energy consumption when no compression is applied is also shown for comparison. We see that signal compression, and the subsequent

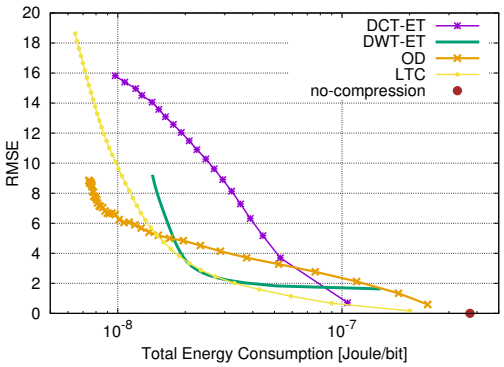


FIGURE 5.8: RMSE as a function of the total energy consumption (compression plus transmission) of ECG signals.

reduction in size of the data to be transmitted, allows a considerable decrease in the total energy consumption. Specifically, AE and OD respectively enable energy savings of about one order of magnitude while respectively providing RMSEs smaller than 2% and 4%. The performance of AE is particularly striking as it allows saving up to two order of magnitude in terms of energy consumption by still keeping the RMSE around 4%. This motivates the use of signal compression techniques for continuous monitoring applications for IoT devices. Note that the actual RMSE can be dynamically tuned at runtime, by allowing slightly less accurate representations (and thus much higher compressions) when no critical patterns are detected. Also, for AE a visual inspection reveals that a RMSE smaller than 4% entails excellent approximations to the original biosignals, and that the error is mainly due to smoothing out spurious oscillations that are introduced and that are not filtered by the preprocessing chain of Fig. 4.6.

A breakdown of the complexity and energy consumption figures for the considered algorithms is provided in Tables 5.1 and 5.2 (for the same RMSE of 7.5%). 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 a single ECG segment of data. From Table 5.1 we see that BSBL-CS is the most energy efficient in terms of compression, LTC is the second best, whereas SOMP-CS utilizes more energy as the ECG signal has to be segmented prior to performing the CS sampling, see Section 5.2.4. Other algorithms such as OD and AE are more computationally demanding, but their maximum compression efficiency is much higher. Since the

transmission energy dominates that needed for data processing, AE and OD represent the best alternatives when all the sources of energy consumptions are added up. In Table 5.2, we show a breakdown of the energy demand for the various processing blocks of OD. Interestingly, we see that peak detection accounts for 91% of the per-segment energy drainage. The same fact applies to all the segment-based approaches (e.g., AE and COMP-CS). We thus recommend working on lightweight peak detection algorithms.

5.3.2 Wearable ECG Signals

We now present some results for ECG signals that we acquired from a Zephyr BioHarness 3 wearable device [34]. 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 250 samples/s with each sample taking 12 bits.

The RMSE *vs* CE tradeoff for these signals is shown in Fig. 5.9 for the best performing compression algorithms. The results are similar to those of Figs. 5.3–5.5 with the main difference that in this case the ECG signals have more artifacts and a higher variability. As such, the resulting RMSE is also higher for all schemes and the compression performance is degraded. The general trends and recommendations remain unchanged, i.e., SOMP-CS and LTC are good choices at low up to intermediate compression efficiencies, whereas OD and AE perform better at higher CEs. However, we remark that the OD's compression efficiency is impacted with respect to that in Fig. 5.5 as the non-steady data of wearables requires more frequent dictionary updates. AE is still very effective, providing the highest compression efficiencies and the smallest RMSE.

The energy consumption figures of all schemes, although slightly rescaled, have a totally similar behavior as those obtained with the PhysioNet MIT-BIH traces (see Figs. 5.7 and 5.8) and are thus not shown in the interest of space. In fact, the energy consumption is marginally affected by the non-steady behavior of wearable signals, which impacts more on the RMSE and compression performance.

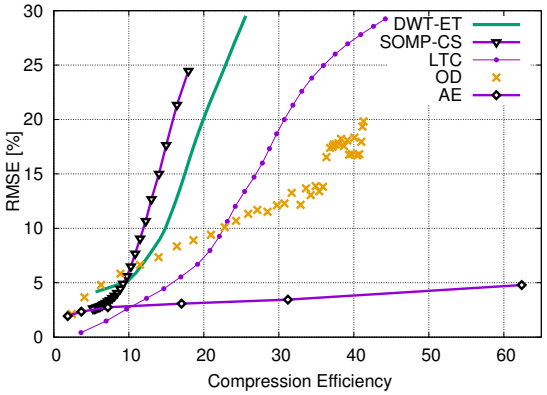


FIGURE 5.9: RMSE *vs* compression efficiency for ECG signals: DWT-ET, SOMP-CS, LTC, OD and AE.

As an illustrative example, in Figs. 5.10 and 5.11 we respectively look at the per segment RMSE and CE performance of LTC, OD and AE. In Fig. 5.10 we fix the compression efficiency to CE = 28 for all schemes and we show the RMSE for each segment considering 12 minutes of ECG readings from one of the subjects. Overall, OD performs satisfactorily, providing an average RMSE of 4%, LTC settles around an RMSE of 11% and AE achieves the best accuracy, i.e., RMSE = 2.6%. Artifacts and the non-steady behavior of the BioHarness ECG traces require more frequent dictionary updates for OD, which then entail some major variability in its RMSE performance, as can be clearly seen in the range $[600, 700]$ segments in both plots. Specifically, when the current dictionary is no longer representative of the input data, at first the RMSE increases and then it sharply decreases due to the consequent dictionary update. Fig. 5.11 shows the per segment compression efficiency for the same ECG trace by operating LTC, OD and AE so that their average RMSE is 3%. From this plot we see that AE reaches much higher CEs, delivering strikingly good performance. Besides, the CE of OD sometimes drops to 1 (no compression) and this happens when the dictionary is updated (see again the range $[600, 700]$ seg. in Fig. 5.11). Note that the first update occurs at time zero, as OD has no dictionary at the beginning of the ECG sequence.

5.3.3 PPG and RESP Signals

As a final result, in Figs. 5.12 and 5.13 we respectively show the RMSE *vs* CE performance for PPG and respiratory (RESP) signals from the PhysioNet MIMIC-II waveform

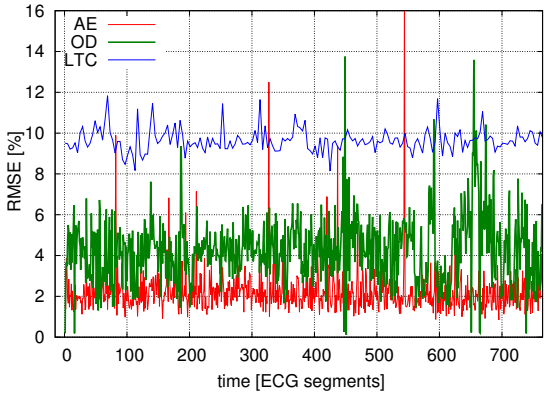


FIGURE 5.10: RMSE as a function of time. CE = 28 for both schemes, RMSE(LTC) = 11%, RMSE(OD) = 4% and RMSE(AE) = 2.6%.

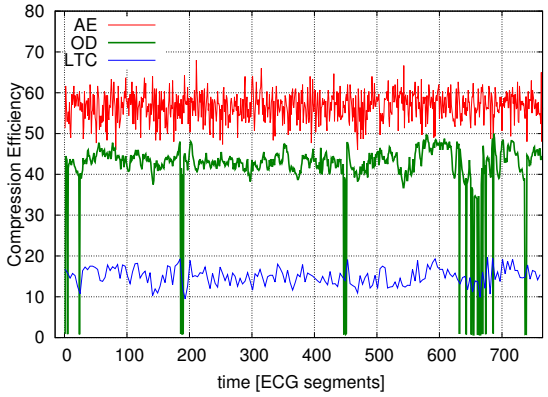


FIGURE 5.11: CE as a function of time. RMSE = 3% for all schemes, CE(LTC) = 15, CE(OD) = 19 and CE(AE) = 56.

database [116]. In these graphs, we only show the performance of the three best algorithms, namely, LTC, OD and AE. AE is plotted considering the number of inner neurons $h \in \{100, 50, 25, 10, 5, 2\}$, where $h = 100$ is represented by the leftmost point, whereas $h = 2$ by the rightmost, and outperforms all the remaining schemes for $h \leq 5$. Clearly, OD and AE are still effective for these signal types. For respiratory signals, LTC performs best for compression efficiencies up to 40, OD is to be preferred for intermediate efficiencies between 40 and 80. The compression efficiency obtained for PPG signals is smaller than that achieved for ECG and RESP but this is due to the lower sampling rate in the PPG traces. For all signals, the RMSE of AE never exceeds 4%, while its CE respectively reaches 56 and 156 for PPG ad RESP when just two inner neurons ($h = 2$) are utilized. These results are impressive and motivate further research, especially to make the AE learning phase online and subject-adaptive, as we do for OD.

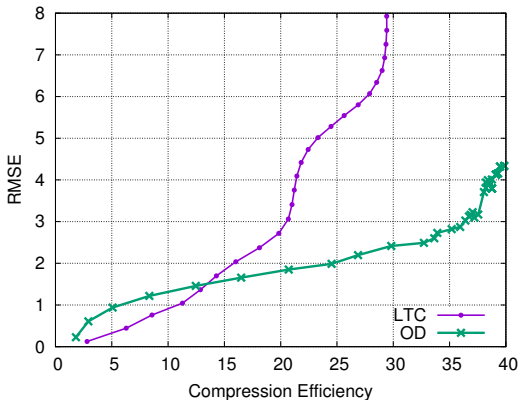


FIGURE 5.12: RMSE *vs* compression efficiency for PPG signals.

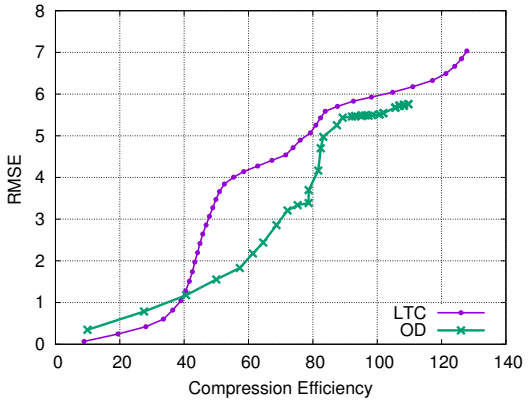


FIGURE 5.13: RMSE *vs* compression efficiency for RESP signals.

5.4 Lesson Learned and Open Issues

In this chapter, we advocated the use of lossy compression as a means to boost the battery life of wireless wearable devices for health monitoring. As a first contribution, we presented an original dictionary based technique, where compression is achieved by building and maintaining at runtime a dictionary. This dictionary is subsequently used to approximate signal sequences transmitting codeword indices in place of the original samples. This technique is found to be very effective, showing excellent approximation capabilities and very high compression efficiencies at the cost of a reasonably small amount of computation. We then considered compression algorithms based on linear approximations. Despite their inherent simplicity, we found them to be quite effective and, when the required compression efficiency is not too high, they represent the best option among competing solutions. We also

found that a recent scheme belonging to this class, called lightweight temporal compression, very closely matches the performance of principal component analysis, at a much smaller computational cost and additionally providing inbuilt guarantees on the maximum approximation error at the decompressor. Thus, we explored the performance of recent approaches based on autoencoders. These neural network architectures are found to be extremely effective, leading to the highest compression efficiencies at a reasonable computational cost. Their performance is striking especially at very high compression rates, where just two inner neurons are utilized to represent input patterns comprising several hundreds of points, still providing very small approximation errors (usually the RMSE remains bounded within 4%). The performance of these algorithms was numerically evaluated against that of the most prominent schemes from the literature, i.e., Fourier and Wavelet transforms, compressive sensing and vector quantization techniques.

Open research areas. From the numerical analysis that we have carried out in this chapter, we have identified several avenues for future research. We have seen that the most promising means to reach high compression efficiencies is to exploit inter-segment correlation. Dictionary based algorithms belong to this category and do a very good job in all respects. Nevertheless, the online scheme proposed in this chapter uses too much memory space at relatively small compression efficiencies, say, smaller than 40. Autoencoders also have a main drawback. In fact, these networks need to go through an offline training phase, during which their weights are shaped utilizing a representative dataset. Although they have excellent generalization capabilities, they will be nevertheless unable to satisfactorily represent input patterns that sharply differ from those in the dataset used for training. Hence, a desirable contribution would be to concoct a new neural network based algorithm with the following properties, both very relevant from a practical standpoint: a) we would like the size of the dictionary not to grow with a diminishing RMSE or, equivalently, with a decreasing compression efficiency. Ideally, the dictionary size should be kept constant. b) We would also like the training phase to be carried out in an online fashion. In this way, the dictionary will adapt to the specific signal statistics of the subject wearing the device. Another interesting subject for future investigations is the joint compression of

multiple vitals, including respiratory rate, electrocardiogram, plethysmograph, and data from motion sensors.

Chapter 6

Body sensor networks – SURF: Subject-adpative Unsupervised ecg compressor for weaRable Fitness monitors

6.1 Introduction

In this chapter, we continue our processing solutions for the long-term monitoring of quasi-periodic electrocardiography (ECG) signals.

We achieve this through the use of unsupervised neural maps for the construction and update of online dictionaries, whose codewords are utilized to match input patterns. Specifically, the acquired biomedical signal is decomposed into *segments* made up of samples between two consecutive signal peaks[1] and we consider these segments as the signal's recurrent patterns. A preliminary training phase uses the incoming segments to learn the actual subject signal distribution. The synaptic weights of the neural maps become progressively and adaptively tuned to approximate such distribution, without any prior knowledge upon it. Note that

[1]This is possible thanks to the quasi-periodic nature of the considered biomedical time series.

these weights represent the codewords. Each new (unseen) segment is thus encoded through a vector quantization 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 further train these algorithms in an online fashion, so as to maintain a representative dictionary at all times. This last feature of our algorithms are particularly appealing as it allows updating the dictionary to new subjects or to the same subject as their signal statistics changes.

In a first design, dictionaries are built using Time Adaptive Self Organizing Maps (TASOM), see [118–120]. These have excellent learning and adaptation capabilities but are found to have limitations when new and sporadic patterns arise or in the presence of artifacts due to, e.g., the motion of the wearer. In our final design, we exploit the Growing Neural Gas network of [121]. Here, the dictionary size can be dynamically adapted through the addition and removal of neurons, and this allows the exploration of new regions in the data space without affecting the current accuracy reached by the dictionary in the region that was already explored. Also, in this last approach dictionaries are learned in a suitable feature space, with reduced dimensionality. This makes it possible to further enhance the compression efficiency and reduce the cost of dictionary updates.

The final compression algorithm is called SURF, for "Subject-adpative Unsupervised ecg compressor for weaRable Fitness monitors". We stress that, although the considered processing tools may seem sophisticated, SURF is rather simple, requiring dictionaries that take less than 20kbytes and that entail computationally light operations at each update.

With SURF, the achievable compression ratios for ECG signals are in the range from 50- to 96-fold (depending on the frequency of artifacts in the data). Also, the performance of SURF is compared against that of selected compression algorithms from the literature involving linear approximations [40], Fourier [42, 73, 108], Wavelet [43] transforms and compressive sensing (CS) [44, 45]. Our approach 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 between 2% and 7% of the signal peak-to-peak amplitude.

A thorough numerical analysis of SURF, carried out for the PhysioNet public dataset [116] and own collected ECG traces from a Zephyr Bioharness 3 device, reveal the following facts: *i)* its dictionaries gracefully and effectively adapt to new subjects or their new activities, *ii)* the size of these dictionaries is kept bounded (i.e., within 20kbytes), making them amenable to their implementation in wireless monitors, *iii)* high compression efficiencies are reaches (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., within a peak-to-peak RMSE within 7% and often smaller than 3% and *v)* compression allows saving energy at the transmitter, making it almost two orders of magnitude smaller.

A compression scheme for quasi-periodic time series can be found in [46], where the authors target the lightweight compression of biomedical signals for constrained devices, as we do in this chapter. 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. Our approach is instead specifically designed to cope with this type of learning, we lose some of the compression and accuracy performance of [46] but we gain in terms of adaptability to the patient that wears the device, learning and adapting the compression and pattern recognition functionalities over time through the collection of measurements. The capability of another type of ANN called *Input-Delay Neural Network* (IDNN) to capture the dynamic characteristics of the input signals is used in [122] for a piecewise ECG compression approach. However, a *different* IDNN is used for each ECG segment of duration 10 seconds and this implies the need for a dedicated training phase for each segment, which entails a too high power consumption for a wearable-based scenario.

Our present work improves upon previous research as neural network structures are utilized to build compact representations of biomedical signals at runtime. Differently from previous

techniques, our dictionaries are built and adapted at runtime to represent at best the signal statistics of the subject that wears the device, following an unsupervised learning approach.

This chapter is structured as follows. In Section 6.2, we introduce the self-organizing maps and in Section 6.3 we describe a first design based on them. The SURF compression algorithm is presented in Section 6.4 and its performance is evaluated in Section 6.5, comparing it against state-of-the art solutions. Our conclusions are drawn in Section 6.6.

6.2 Unsupervised Dictionary Learning through Self-Organizing Maps

The 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 rectangular lattice $\mathcal{A}$ with L neurons arranged in M rows and M columns. The input space is m-dimensional, i.e., $\mathcal{X} \subset \mathbb{R}^m$ with input vectors $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T \in \mathcal{X}$. 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 [119].

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

to the Euclidean distance. Its index $i(\boldsymbol{x})$ satisfies:

$$i(\boldsymbol{x}) = \text{argmin}_j \left\| \boldsymbol{x}(n) - \boldsymbol{w}_j(n) \right\|, \; j = 1, \ldots, L \,. \tag{6.1}$$

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. In this work, we set $h_{ij}(n) = \exp(-d_{ij}^2/(2\sigma(n)^2))$, where $\sigma(n)$ is the width of the topological neighborhood, exponentially decreasing with increasing time n [119]. These properties are reflected by the unnormalized Gaussian function: $h_{ij}(n) = \exp(-d_{ij}^2/(2\sigma(n)^2))$, where $\sigma(n)$ is the width of the topological neighborhood at iteration n. $\sigma(n)$ decreases with time according to an exponential decay function.

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

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \eta(n) h_{ij}(n)(\boldsymbol{x}(n) - \boldsymbol{w}_j(n)) \,. \tag{6.2}$$

(6.2) 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, according to $h_{ij}(n)$) 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 time n.

$$\eta(n) = \eta_0 \exp\left(-\frac{n}{\tau_2}\right), \; n = 0, 1, \ldots \tag{6.3}$$

where τ_2 is another time constant. The adaptive process of the synaptic weights in the network, computed in accordance with Eq. (6.2) , may 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 $N = 1000 + 500 \times L$ [119].

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 according to (6.1). $\Phi(\cdot)$ 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 and we use it to design a subject-adaptive dictionary for biomedical signals. However, upon completion of the learning phase, the SOM map stabilizes and further learning / adaptation to new input distributions is difficult. In the presence of non-stationary signals, adaptive learning must be employed to update the feature map. The time-adaptive self-organizing map (TASOM) does 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 [120] as an extended version of the basic SOM and preserves 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. This feature enables the TASOM to be more flexible and to approximate the input data space distribution as it evolves. For more details on the TASOM algorithm, the reader is referred to [120].

6.3 A first design: TASOM-based ECG compression

We leverage the quasi-periodic nature of the considered vital signs to develop a lossy compression technique with subject-adaptive dictionary. First, we identify as *motifs* the sequence of samples between consecutive peaks (which we call *segments*) and we use them to build a *dictionary* that stores typical segments and is maintained consistent and well representative through online updates. The TASOM unsupervised learning algorithm is utilized to construct and manage the dictionary. A diagram of the proposed technique

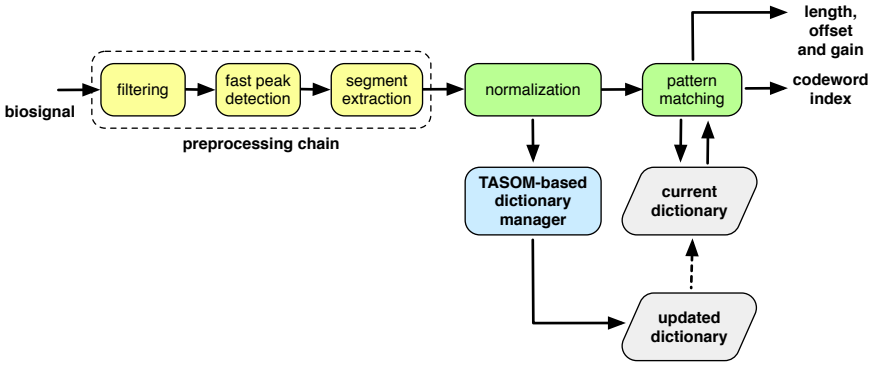


FIGURE 6.1: Diagram of the TASOM-based compression algorithm.

is shown in Fig. 6.1. The physiological signal is first preprocessed through a third-order Butterworth filter to remove artifacts. Then the fast and simple peak detection algorithm exposed in [88] is employed to locate the signal peaks. The segment extractor splits the signal into segments made up of samples between subsequent peaks and inputs them to the normalization module. Since the segments may have different lengths, linear interpolation resizes the current segment with length $r_{\boldsymbol{x}}(n)$ to a fixed length m. We refer to the resized segment as $\boldsymbol{x}(n) = [x_1(n), \ldots, x_m(n)]^T$, whereas

$$e_{\boldsymbol{x}}(n) = \frac{\sum_{k=1}^{m} x_k(n)}{m} \tag{6.4}$$

is its offset and

$$g_{\boldsymbol{x}}(n) = \left(\sum_{k=1}^{m} x_k(n)^2 / m \right)^{1/2} \tag{6.5}$$

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

$$x_k(n) \leftarrow \frac{x_k(n) - e_{\boldsymbol{x}}(n)}{g_{\boldsymbol{x}(n)}}, \; k = 1, \ldots, m \tag{6.6}$$

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 design it thinking of a communication scenario consisting of a transmitting wearable device and a

receiver, such as a PDA or 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 scheme is activated for the first time, a sufficient number N of signal segments are provided as input to the TASOM to perform a *preliminary training phase*. Such 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 assume that time is reset when the preliminary training ends and assume $n = 0$ at such point. Both $\mathcal{C}^c(0) = \{\boldsymbol{c}_1^c(0), \ldots, \boldsymbol{c}_L^c(0)\}$ and $\mathcal{C}^u(0) = \{\boldsymbol{c}_1^u(0), \ldots, \boldsymbol{c}_L^u(0)\}$ are defined as the dictionaries whose codewords $\boldsymbol{c}_*^*(0)$ are equal to the synaptic-weight vectors of the TASOM. 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, that is, 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

$$i_{\boldsymbol{x}}(n) = \text{argmin}_j \left\| \boldsymbol{x}(n) - \boldsymbol{c}_j^c(n) \right\|, \; j = 1, \ldots, L\,. \tag{6.7}$$

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 re-map $\boldsymbol{x}(n)$ onto the index of the best matching codeword in the new dictionary $\mathcal{C}^c(n)$, i.e., map $\boldsymbol{x}(n)$ onto the index $i_{\boldsymbol{x}}(n)$ obtained through (6.7) using 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 the current dictionary has been changed in step 4, then also send $\mathcal{C}^u(n)$.

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 $\boldsymbol{c}^c_{i_{\boldsymbol{x}}(n)}(n)$ is assumed to be a suitable representation of 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 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 $r_{\boldsymbol{x}}(n)$.

6.4 The SURF compression scheme

The TASOM-based compressor described in the previous section is now improved through the use of a more flexible neural network architecture in several respects.

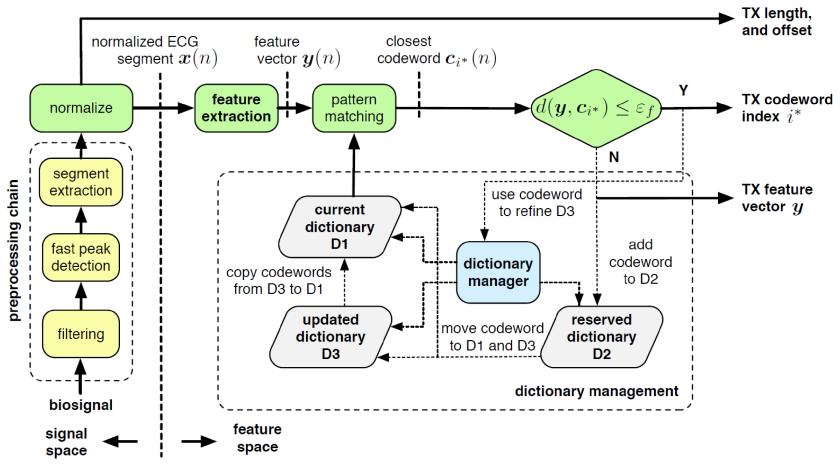


FIGURE 6.2: Flow diagram of the SURF compression algorithm: the dictionary is learned in the feature space. Codewords can be added or removed. When the distance between the best matching codeword and the current feature vector is higher than a threshold, a new codeword is added. That codeword is in an *assessment* phase until it is either permanently added or deleted (i.e., when no further matches occur). 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.

O1) **Objective 1** - "specializing the dictionary to new signal areas": we remark that the number of neurons in the TASOM map remain fixed as time evolves and this entails that any further refinement of the dictionary, whenever the signal statistics undergoes changes and new (unseen) behaviors arise is not always 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 of 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 covered (and successfully approximated). This is accomplished through the integration of a growing gas neural network [121] into our new design.

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) the first one consists in 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) The second technique is about the way in which the

dictionary is updated. In the TASOM-based approach the old dictionary is entirely replaced whenever it is no longer capable of approximating ECG segments within a preset accuracy. Instead, in our new approach codewords are selectively updated, replacing them with 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 due to the various body movements of the wearer. Dictionary-based approaches are particularly impacted by them as the dictionary manager, may try to match them with current codewords and then it may also adapt the dictionary either adding new codewords or updating existing ones. This is of course to be avoided as it results in a degraded dictionary. However, these noisy segments must be isolated and differentiated from genuine ECG segments containing anomalous patterns as these may have detected and learned. Our new compressor successfully copes with this by: 1) sending features in place of codewords whenever none of the current codewords provide a satisfactory match and 2) simultaneously starting an *assessment* phase for the new pattern. During that phase a new neuron is temporarily added to the dictionary. Its permanent addition to the dictionary only occurs if multiple matches occur within a predetermined time frame.

The three objectives of above are achieved through the SURF compression algorithm that we detail next. It exploits 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 and $n \leftarrow n + 1$ each time a new ECG segment is processed.

A diagram of the SURF algorithm is shown in Fig. 6.2. The signal is at first preprocessed through the same chain of Fig. 6.1, involving filtering to remove artifacts, ECG peak detection and segment extraction. After this, ECG segments are normalized through resizing and offset removal. Since different ECG segments may have different lengths (number of samples), 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:

$$x_k(n) \leftarrow x_k(n) - e_{\boldsymbol{x}}(n)\,,\; k = 1, \ldots, m\,, \tag{6.8}$$

where $e_{\boldsymbol{x}}(n)$ is defined in (6.4). 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 is accomplished by computing the DCT transform of $\boldsymbol{x}(n)$ and retaining the first (low-pass filtering) f coefficients in the transform (frequency) domain. We underline that our method is rather general and any other mapping 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 $R(n) \ll L(n)$.

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 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.,

$$i^* = \text{argmin}_j\, d(\boldsymbol{y}(n), \boldsymbol{c}_j^c(n))\,,\; j = 1, \ldots, L(n) \tag{6.9}$$

If $d(\boldsymbol{y}(n), \boldsymbol{c}_{i^*}^c)$ is smaller than a preset error tolerance $\varepsilon_f > 0$,[2] 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

[2]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 6.3.

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 can adequately approximate the current feature vector, which is then termed *unmatched*. Note that this may be due to 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, what we do is to 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 ready for use in the signal compression, but they rather go through an *assessment* phase: they are then added to the dictionary D1 only if they are matched at least γ times within Δ time slots after their addition to D2, with γ and Δ being preset parameters. The rationale behind such an assessment is that these 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 the dictionary 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 allows the specialization of 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 n, 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. Let assume that time is reset when the preliminary training ends and assume $n = 0$ at such point. Both $\mathcal{C}^c(0) = \{\boldsymbol{c}_1^c(0), \ldots, \boldsymbol{c}_{L(0)}^c(0)\}$ and $\mathcal{C}^u(0) = \{\boldsymbol{c}_1^u(0), \ldots, \boldsymbol{c}_{L(0)}^u(0)\}$ are defined as the dictionaries whose codewords $\boldsymbol{c}_j^c(0)$ and $\boldsymbol{c}_j^u(0)$, $j = 1, \ldots, L(0)$, are equal to the synaptic-weight vectors at the end of the initial training. Hence, at time $n = 0$ we have $\boldsymbol{c}_j^c(0) = \boldsymbol{c}_j^u(0) = \boldsymbol{w}_j(0)$, $j = 1, \ldots, L(0)$. 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 is based on the Hebbian learning theory in [123, 124].

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., $d(\boldsymbol{y}(n), \boldsymbol{c}_{i^*}^u(n)) \leq d(\boldsymbol{y}(n), \boldsymbol{c}_j^u(n))$, $j = 1, \ldots, L(n)$. We have that i^* is the *winning* neuron in map (dictionary) D1 for this input (feature) vector 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:

$$\boldsymbol{w}_{i^*}^{\text{new}} \leftarrow \boldsymbol{w}_{i^*} + \epsilon_b(\boldsymbol{y}(n) - \boldsymbol{w}_{i^*})\,. \tag{6.10}$$

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 created. 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 through an edge:

$$\boldsymbol{w}_j^{\text{new}} \leftarrow \boldsymbol{w}_j + \epsilon_n(\boldsymbol{y}(n) - \boldsymbol{w}_j)\,, \tag{6.11}$$

where ϵ_b and ϵ_n are constant learning rates. The new weight vectors of (6.10) and (6.11) 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 $\boldsymbol{y}(n)$ $(n = 1, 2, \dots)$ 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

$$i^*_{\boldsymbol{y}}(n) = \text{argmin}_j\, d(\boldsymbol{y}(n), \boldsymbol{c}^c_j(n))\,,\ j = 1, \dots, L(n) \tag{6.12}$$

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

Step 2) Let $d(n) = d(\boldsymbol{y}(n), \boldsymbol{c}^c_{i^*}(n))$ be the distance between $\boldsymbol{y}(n)$ and the closest codeword $\boldsymbol{c}^c_{i^*}(n)$, where we use i^* as a shorthand notation for $i^*_{\boldsymbol{y}}(n)$. If $d(n) > \varepsilon_f$ act as follows, otherwise move to **Step 3**. 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 exceeds γ in the Δ time slots following its addition to D2, then 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 this neuron has a weight vector $\boldsymbol{w} = \boldsymbol{y}(n)$.

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^*:

$$\text{error}(i^*)^{\text{new}} \leftarrow \text{error}(i^*)^{\text{old}} + d(\boldsymbol{y}(n), \boldsymbol{c}_{i^*}^c(n))\,. \tag{6.13}$$

Step 4) Dictionary management. The following dictionary update procedure follows the growing neural gas network algorithm of [121]. Every λ time steps, we check the current dictionary 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 with 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 q (synaptic weight vector $\boldsymbol{w}_q$) with the maximum accumulated error is determined. A new neuron r (synaptic weight vector $\boldsymbol{w}_r$) is generated halfway between q and its neighbor f that has the largest accumulated error:

$$\boldsymbol{w}_r = 0.5(\boldsymbol{w}_q + \boldsymbol{w}_f)\,. \tag{6.14}$$

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 q and f (edge (q, f)) and add the two edges (q, r) and (r, f). Multiply the accumulated error of q and f by constant α and initialize the accumulated error of r with the new

value of the accumulated error of q.

Step 6) All the accumulated error 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 tolerable error ε), the codeword in D1 is replaced with 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 6.3, where the whole dictionary is updated. This helps reduce the overhead associated with the dictionary update operation (see objective O2).

At the receiver, 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., $\boldsymbol{x}^* = \Psi^{-1}(\boldsymbol{y}^*)$, adding back offset $e_{\boldsymbol{x}}(n)$ and resizing the codeword to the actual segment length $r_{\boldsymbol{x}}(n)$.

6.5 Numerical Results

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

For the **energy consumption**, following the approach of [111, 112] we compute three metrics: 1) the energy consumption for the execution of the compression algorithms in the node (termed *compression energy*), 2) the energy drained by the the transmission of the (either compressed or original) signal over a wireless channel (*transmission energy*) and 3) the total energy, which is given by the sum of the previous two metrics. 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. These were then translated into the corresponding number of MCU cycles and, in turn, into the energy consumption in Joule per bit considering a Cortex M4 [113] processor, see also [111]. For the transmission energy, we took a Texas Instruments CC2541 low-energy Bluetooth system-on-chip [114], which is widely adopted for IoT devices.

The **Compression Efficiency (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. For the reconstruction fidelity, we computed the **Root Mean Square Error (RMSE)** between the original and the compressed signals normalizing it with respect to the signal's peak-to-peak amplitude, that is:

$$\text{RMSE} = \frac{100}{\text{p2p}} \left(\frac{\sum_{i=1}^{K} (x_i - \hat{x}_i)^2}{K} \right)^{1/2}, \tag{6.15}$$

where K corresponds to the total number of samples in the trace, x_i and $\hat{x}_i$ are the original sample and the one reconstructed after the decompressor in position i, respectively. p2p is the average peak-to-peak signal's amplitude. The SURF parameters have been set as follows: $\epsilon_b = 0.01$, $\epsilon_n = 0.005$, $\alpha = 0.5$, $\beta = 0.995$, $\lambda = 200$ and $a_{max} = 100$.

In Section 6.5.1 we first assess the performance of the considered compression algorithms for the standard test ECG traces from the PhysioNet MIT-BIH arrhythmia database [116].

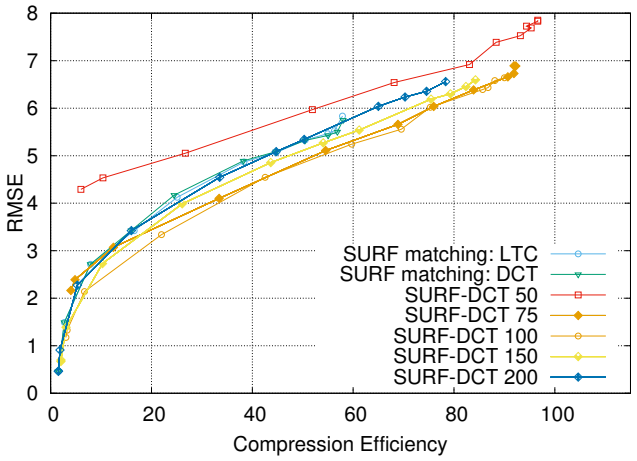


FIGURE 6.3: RMSE *vs* compression efficiency for ECG signals – comparison of SURF compression schemes.

In Section 6.5.2, we extend our analysis to ECG traces that we collected from a Zephyr BioHarness 3 wearable chest monitor.

6.5.1 PhysioNet ECG traces

In the first set of graphs, we show results for ECG traces. To this end, we considered the following traces from the MIT-BIH arrhythmia database [116]: 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 of 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 [116, 117]. The above performance metrics were obtained for these traces and their average values are shown in the following plots.

As a first result, in Figs. 6.3 and Fig. 6.4 we show the compression efficiency and the total energy consumption of SURF, both plotted versus the RMSE. In these plots we quantify the impact of feature space size f, which corresponds to the number of DCT coefficients that are retained and stored in the feature vector $\boldsymbol{y}(n)$. Three SURF variants were implemented:

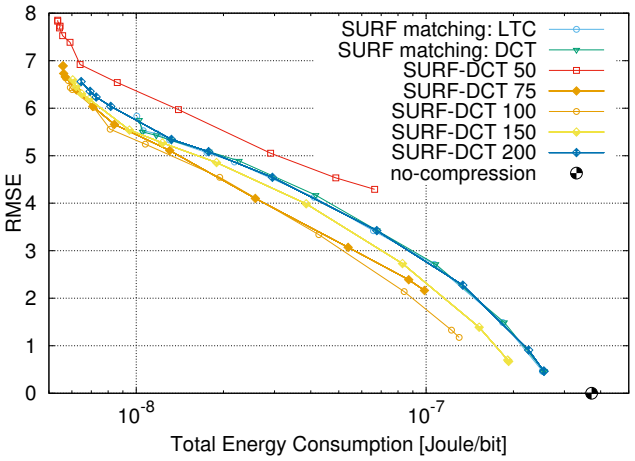


FIGURE 6.4: RMSE as a function of the total energy consumption (compression plus transmission) of ECG signals.

1) SURF matching: LTC. It refers to the time domain implementation, i.e., the feature map is an identity matrix and the feature vectors $\boldsymbol{y}(n)$ correspond to the original signal segments, i.e., $\boldsymbol{y}(n) = \boldsymbol{x}(n)$. Whenever an input segment is unmatched, the corresponding LTC coefficients are transmitted to the receiver and the segment normalization block is skipped (as LTC does not require it).

2) SURF matching: DCT. It is another time domain implementation of SURF where, 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.

3) SURF-DCT. It is the feature domain implementation of Section 6.4, for which we considered the following values for the feature space dimensionality $f \in \{50, 75, 100, 150, 200\}$.

From Fig. 6.3, we see that SURF-DCT 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 decreasing CE. However, we see that when f increases beyond 100 the RMSE performance gest affected and starts decreasing. In these cases, SURF-DCT behaves similarly to its time domain

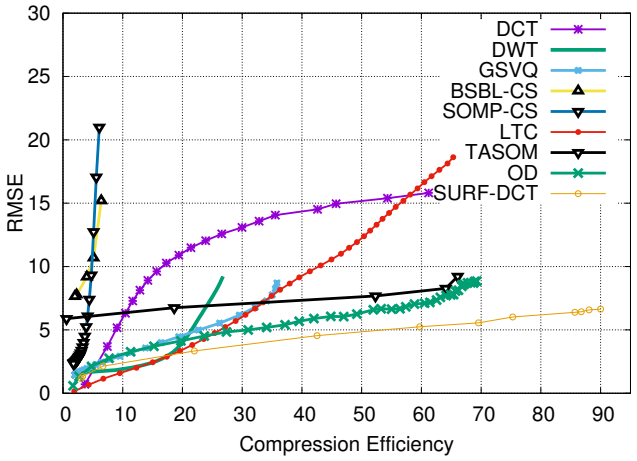


FIGURE 6.5: RMSE *vs* compression efficiency for ECG signals: comparison of compression algorithms.

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. 6.4 shows the total energy consumption (adding up processing and transmission) and we see that savings of almost two orders of magnitude are possible.

In Fig. 6.5, we plot RMSE *vs* CE for all the considered compression algorithms, namely, techniques based on DCT, DWT and linear approximation LTC, the GSVQ and OD codebook-based schemes, the BSBL-CS and SOMP-CS dimensionality reduction algorithms, the TASOM-based algorithm of Section 6.3 and SURF-DCT. 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 a higher CE, for which LTC, TASOM, OD and SURF are to be preferred. As for the CS-based algorithms, neither SOMP-CS nor BSBL-CS provides satisfactory performance. The compression efficiency of SOMP-CS is rather small and the corresponding RMSE tends to diverge for, e.g., CE larger than 5. As we shall see shortly below, although the BSBL-CS compressor has the lowest energy consumption, the overall energy expenditure is high as this approach is less effective in terms of CE than other schemes such as dictionary based (GSVQ, OD) and neural map based algorithms (TASOM and SURF). For GSVQ, we move along the RMSE *vs* CE curves by changing the threshold that governs the

number of bits that are encoded into the residual stream (residual encoding is the operation that affects the most the performance of GSVQ). There, the dictionary size affects the maximum achievable compression but the maximum CE is always smaller than that of OD, where the dictionary adapts to the signal in an online fashion. Although not shown in the plot, one may be thinking 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 recommended. SURF-DCT offers very good performance both in terms of RMSE and CE, resulting the best algorithm. We also emphasize the substantial gap in both performance metrics that SURF-DCT achieves with respect to the TASOM-based approach. The reasons for this are that: *i)* SURF dictionaries more effectively deal with new patterns and artifacts, *ii)* SURF works in the signal feature space, where the size of codewords is reduced to f elements and dictionary updates are selectively implemented only for the codewords that no longer meet the error tolerance.

For SURF, we also look at the number of codewords in the dictionary as a function of the CE, see Fig. 6.6 and 6.7. From those plots, we see that the dictionary size never exceeds 17kbytes and that for this reason the approach is amenable to implementation on wearables. We also note that the dictionary 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. 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 increases beyond a certain threshold, a smaller codeword set will suffice to represent the input signal space and the dictionary size will correspondingly start decreasing.

In Fig. 6.8, 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

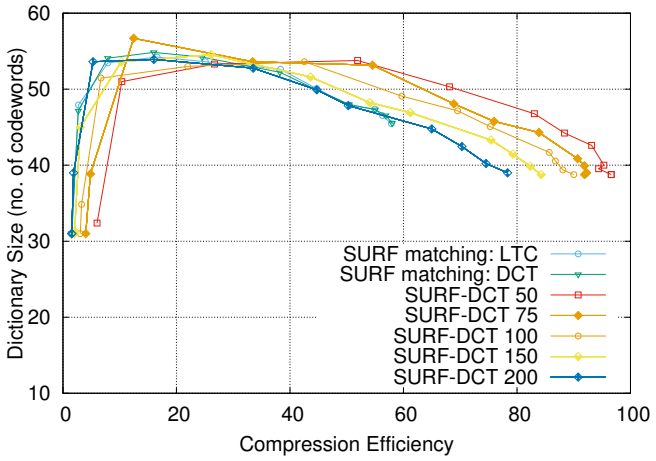


FIGURE 6.6: SURF compression: codebook size as a function of the compression efficiency. The tradeoff curve is obtained by varying the representation accuracy parameter ε_f.

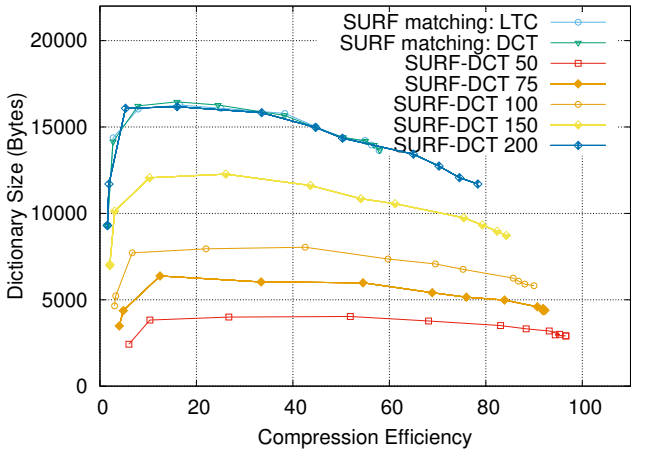


FIGURE 6.7: SURF compression: codebook size as a function of the compression efficiency. The tradeoff curve is obtained by varying the representation accuracy parameter ε_f.

curves are obtained by varying the compression efficiency of each algorithm from the minimum to the maximum achievable (which are scheme-specific, see Fig. 6.11). The RMSE increases with an increasing compression efficiency, whereas the compression energy depends weakly on CE. As expected, BSBL-CS 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, with entries in $\{0, 1\}$. LTC is the second best, whereas OD, SOMP-CS, GSVQ and TASOM perform very close to one another and have the worst energy consumption for compression. SURF consumes a smaller amount of energy than them. We underline that the energy consumption of SURF, TASOM, OD, SOMP-CS and GSVQ is dominated by the preprocessing chain of Fig. 6.2 (as we quantify below through

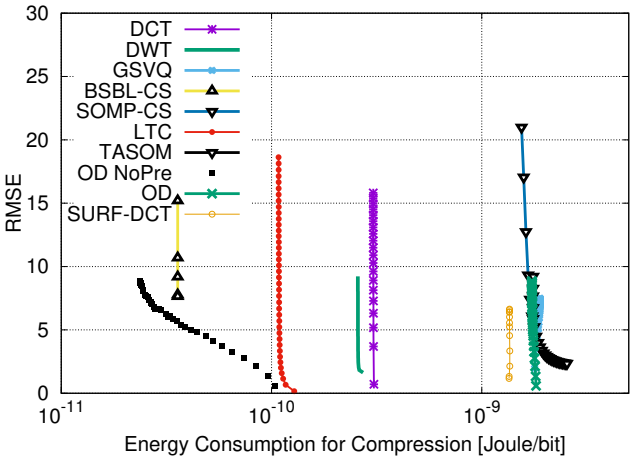


FIGURE 6.8: RMSE *vs* compression energy obtained varying CE as the independent parameter.

Tables 6.1 and 6.2). In Fig. 6.8, we also show the performance of OD by removing the contribution of the pre-processing blocks (filtering, peak detection and segment extraction): the corresponding curve is referred to in the plot as "OD 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. The same decrease in processing energy is obtained by removing these contributions from SURF-DCT.

In Fig. 6.9, we show the RMSE as a function of the *total energy consumption*, which is obtained summing the energy required for compression to that for the subsequent transmission of the compressed bitstream over a CC2541 Bluetooth low-energy wireless interface. 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, OD 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 providing the smallest RMSE is used. In Fig. 6.9, 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

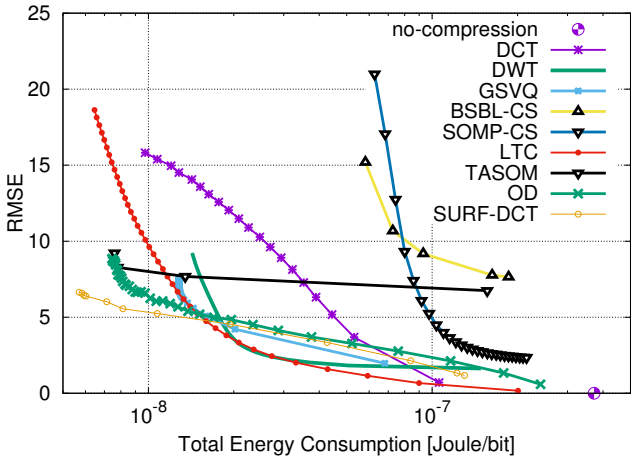


FIGURE 6.9: RMSE as a function of the total energy consumption (compression plus transmission) of ECG signals.

energy consumption. When the energy reduction is of one order of magnitude, LTC, OD and SURF respectively provide RMSEs smaller than 2%, 4% and 2%. The performance of SURF is particularly striking as it allows saving up to two order of magnitude in terms of energy consumption by still keeping the RMSE around 6%. This motivates the use of signal compression techniques for continuous monitoring applications for wearable devices. Also, note that SURF's actual RMSE is automatically adjusted at runtime, by allowing slightly less accurate representations, and thus much higher compressions, when no critical patterns occur.

A breakdown of the complexity and energy consumption figures for the considered algorithms is provided in Tables 6.1 and 6.2. 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 a single ECG segment of data. From Table 6.2 we see that SURF has a lower energy consumption with respect to TASOM for compression, transmission and in total. From these results, 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 all the segment-based approaches (e.g., OD and COMP-CS). We thus recommend working on lightweight peak detection algorithms.

The plots in Fig. 6.10 show original and reconstructed ECG temporal signals using LTC,

TABLE 6.1: Energy breakdown [no. operations] and consumption [μJ] for the TASOM compressor. RMSE = 5.64%.

	Pass band filtering	**Peak detection**	**Segment extractor**	**Pattern matching**	**Codebook manager**	**Total**
Additions	4874.89	61284.37	592	3142.99	7105.42	77001.68
Multiplications	4526.69	60587.96	298	3009.81	4260	72682.46
Divisions	348.21	696.41	3	0	29	1076.62
Comparisons	0	522.31	0	194.15	26.87	743.33
Compression energy [μJ]	0.47	4.34	0.03	0.21	0.39	**5.43**
Transmission energy [μJ]	–	–	–	–	–	**117.23**
Total energy [μJ]						**122.66**

TABLE 6.2: Energy breakdown [no. operations] and consumption [μJ] for SURF. RMSE = 6.39%.

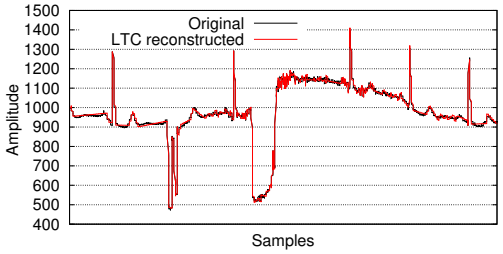
	Pass band filtering	**Peak detection**	**Segment extractor**	**Pattern matching**	**Codebook manager**	**Total**
Additions	4843.93	60895.16	2497	1462.29	1136.80	70835.19
Multiplications	4497.94	60203.17	1498	1477.06	577.75	68253.19
Divisions	346	691.99	2	0	0	1039.99
Comparisons	0	518.99	0	11.27	137.01	667.27
Compression energy [μJ]	0.47	4.31	0.13	0.1	0.06	**5.06**
Transmission energy [μJ]	–	–	–	–	–	**16.3**
Total energy [μJ]						**21.36**

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 as part of its approximation. TASOM does a better job, but its dictionary is unable to effectively represent the new patterns. SURF provides the best results as it preserves the signal morphology, while achieving the highest compression efficiencies, i.e., up to CE = 53.

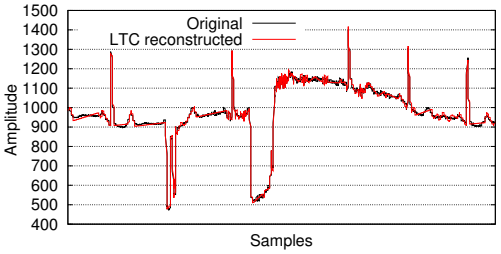
6.5.2 Wearable ECG Signals

We now present some results for ECG signals that we acquired from a Zephyr BioHarness 3 wearable device [34]. 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 250 samples/s with each sample taking 12 bits.

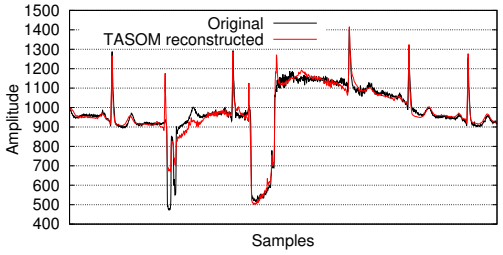
The RMSE *vs* CE tradeoff for these signals is shown in Fig. 6.11 for the best performing compression algorithms. The results are similar to those of Fig. 6.5 with the main difference that in this case the ECG signals are prone to artifacts and have a higher non-stationarity. As such, the resulting RMSE is also higher and the compression performance is degraded for



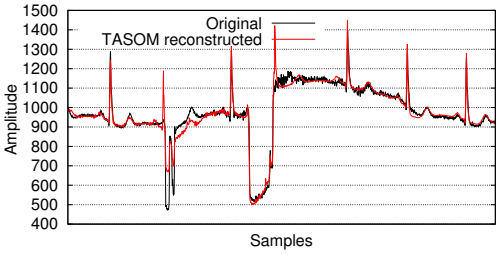
(A) **LTC**: CE = 22 and RMSE = 2%.



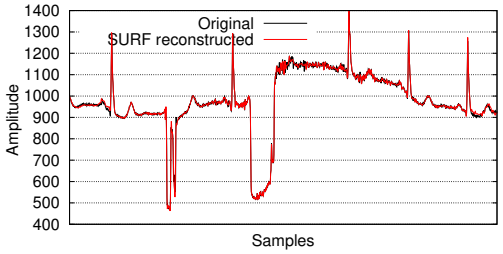
(B) **LTC**: CE = 29 and RMSE = 3%.



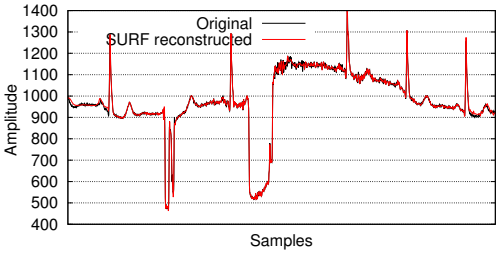
(C) **TASOM**: CE = 34 and RMSE = 2%.



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



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



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

FIGURE 6.10: Original and reconstructed signal in the presence of artifacts for LTC, TASOM and SURF.

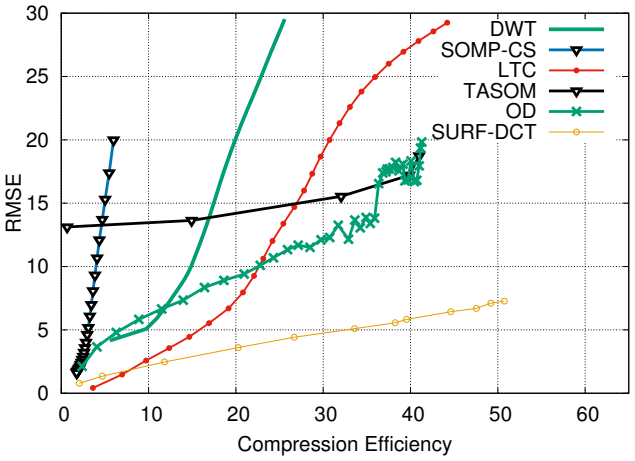


FIGURE 6.11: RMSE *vs* compression efficiency for Bioharness ECG signals

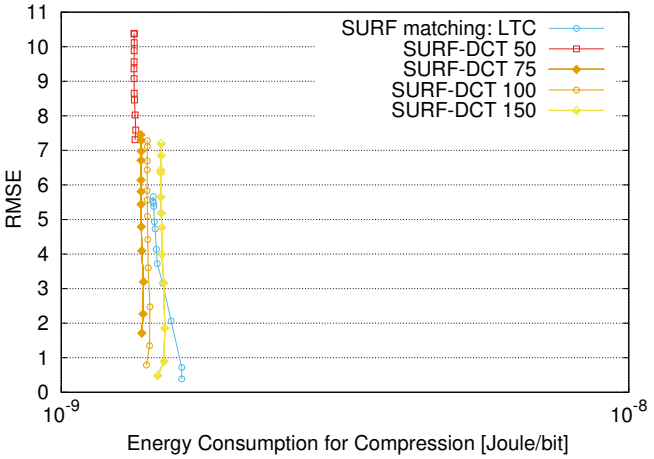


FIGURE 6.12: RMSE vs Energy for Computation for Bioharness traces

all schemes. The general trends and recommendations remain unchanged, i.e., SOMP-CS and LTC are good choices at low up to intermediate compression efficiencies, whereas OD perform better at higher CEs. Here, SURF shows its superior performance and especially its ability to gracefully adapt to artifact-prone and non-steady 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.

The energy consumption figures, although rescaled, have a totally similar behavior as those obtained with the PhysioNet MIT-BIH traces and shown in Figs. 6.8 and 6.9. The energy consumption associated with onboard processing and the total energy budget for the Zephyr Bioharness ECG signals are respectively shown in Figs. 6.12 and 6.13.

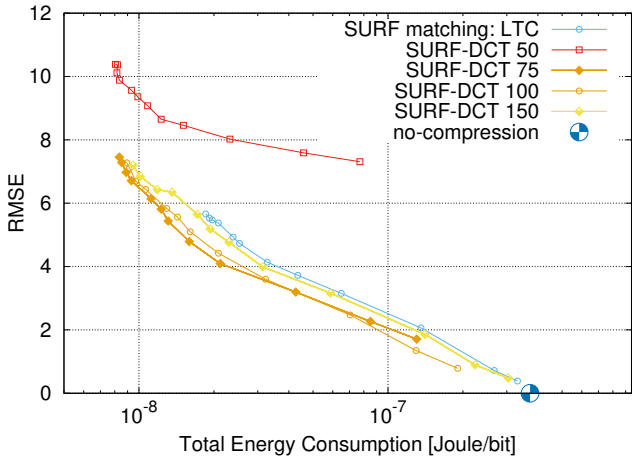


FIGURE 6.13: RMSE vs Total Energy for Bioharness traces

6.6 Conclusions

In this chapter, we have presented SURF, a Subject-adpative Unsupervised ecg compressor for weaRable Fitness monitors. Our idea is to use lossy compression on the ECG signal at the wearable device, to decrease the energy consumption entailed in its wireless transmission and thus prolong the battery lifetime. SURF exploits unsupervised learning techniques to build and maintain, at runtime, a *subject-adaptive* dictionary without requiring any prior information on the signal itself. 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). It exploits Growing Neural Gas (GNG) networks. Quantitative 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: compression ratios of up to 90-times are generally achievable, reconstruction errors (RMSE) remain often within 3%, energy consumption is reduced up to two-orders of magnitude with respect to sending the signal uncompressed and the input signal morphology is well preserved.

We have identified, from the numerical analysis, several avenues for future research that we have carried out in this chapter. Firstly, the major part of the energy consumption related to computation belongs to the filtering and peak detection sections. Implementation of light-weight peak detection method will highly improve the efficiency. Moreover, we have

started a feature domain compression. We have used DCT for this purpose which has a great effect on performance improvement. Hence, a desirable subject for future research is to study and analyze the feature domain more deeply.

Chapter 7

Conclusion

In this doctoral thesis, we have addressed the design and the performance evaluation of signal compression schemes for wireless and battery operated sensor networks and body sensor networks (wearable devices). The objective of these techniques is to perform signal compression (either temporal or spatio-temporal) at the source(s) so as to reduce the amount of data to transmit. This has the twofold beneficial effect of: i) decreasing the energy entailed in the data transmission and ii) reducing the overall energy consumption (signal processing plus transmission) at the source. These facts lead to obvious advantages in terms of memory requirements and overall energy budget, making it possible to prolong the network or device lifetime.

In the second part of the thesis, we have addressed the design of spatio-temporal compression schemes for environmental wireless sensor networks by, at the same time, delving into their performance comparison with respect to relevant compression algorithms from the literature. Among other schemes, our results revealed that compressive sensing (CS) can be effectively used for the compression of correlated signals (in time, space or both) and that the joint sparsification along the temporal and spatial dimensions is key to achieve good performance and improve upon distributed source coding schemes, even when signals are highly correlated in space. In addition, a crucial role is played by the spatial transformation (sparsification) basis, which has to be adapted to the specific characteristics of the signal at hand. In

our work, we did so via an online covariogram-based algorithm for the estimation of the signal covariance matrix and its refinement as time evolves. The main idea behind our CS schemes is to estimate the signal sparsification basis on the fly, by analyzing the statistical properties of the input signal. This allows for projections in a space where the signal is with good approximation sparse and only a few of its coefficients suffice to represent it. Our quantitative results, obtained for synthetic and real signals, reveal that our final *covariogram-based compressive sensing* scheme performs satisfactorily across all values of correlation, and is the algorithm of choice in terms of quality of reconstruction and energy consumption at the sensor nodes.

In the second part of this thesis, we advocated the use of lossy compression as a means to boost the battery life of wireless wearable devices for health monitoring. As a first contribution, we presented an original dictionary based technique, where compression is achieved by building and maintaining at runtime a dictionary. This dictionary is subsequently used to approximate signal sequences transmitting codeword indices in place of the original samples. This technique is found to be very effective, showing excellent approximation capabilities and very high compression efficiencies at the cost of a reasonably small amount of computation. We then considered compression algorithms based on linear approximations. Despite their inherent simplicity, we found them to be quite effective and, when the required compression efficiency is not too high, they represent the best option among competing solutions. We also found that a recent scheme belonging to this class, called lightweight temporal compression, very closely matches the performance of principal component analysis, at a much smaller computational cost and additionally providing inbuilt guarantees on the maximum approximation error at the decompressor. Thus, we explored the performance of recent approaches based on autoencoders. These neural network architectures are found to be extremely effective, leading to the highest compression efficiencies at a reasonable computational cost. Their performance is striking especially at very high compression rates, where just two inner neurons are utilized to represent input patterns comprising several hundreds of points, still providing very small approximation errors (usually the RMSE remains bounded within 4%). The performance of these algorithms was numerically evaluated against that of the most

prominent schemes from the literature, i.e., Fourier and Wavelet transforms, compressive sensing and vector quantization techniques.

In the final chapter, we have presented an original *subject-specific* lossy compression algorithm for wearable fitness monitors, called SURF. 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 20kbytes), making them amenable to their implementation in wireless monitors, *iii)* high compression efficiencies are reaches, allowing for 50 to 96-fold reduction in the signal size, 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, making it almost two orders of magnitude smaller. These facts make SURF a compelling algorithm, that outperforms the compression approaches that were proposed thus far for the considered vital signs.

Bibliography

[1] G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals," *IEEE Transactions on Information Theory*, vol. IT-28, no. 1, pp. 55–67, 1982.

[2] C. J. Willmott, K. Matsuura, and D. R. Legates, "Global Air Temperature and Precipitation: Regridded Monthly and Annual Climatologies," 2012. Center for Climatic Research, Department of Geography, University of Delaware, US.

[3] M. Srivastava, T. Abdelzaher, and B. Szymanski, "Human-centric Sensing," *Philosophical Transactions of Royal Society*, vol. 370, pp. 176–197, November 2012.

[4] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki, "Practical data compression in wireless sensor networks: A survey," *Elsevier Journal of Network and Computer Applications*, vol. 35, pp. 37–59, Jan. 2012.

[5] P. L. Dragotti and M. Gastpar, *Distributed Source Coding: Theory, Algorithms and Applications*. Academic Press, 2009.

[6] Y. C. Eldar and G. Kutyniok, eds., *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

[7] 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, pp. 15:1–15:34, Nov. 2014.

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

[9] S. S. Pradhan and K. Ramchandran, "Distributed Source Coding Using Syndromes (DISCUS): Design and Construction," *IEEE Transactions on Information Theory*, vol. 49, pp. 626–643, Mar. 2003.

[10] E. Candes and T. Tao, "Near optimal signal recovery from ran-dom projections: Universal encoding strategies?," *IEEE Transactions on Information Theory*, vol. 52, pp. 5406–5425, Dec. 2006.

[11] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Birkhauser (Series: Applied and Numerical Harmonic Analysis), 2013.

[12] G. Quer, R. Masiero, G. Pillonetto, M. Rossi, and M. Zorzi, "Sensing, Compression and Recovery for WSNs: Sparse Signal Modeling and Monitoring Framework," *IEEE Transactions on Wireless Communications*, vol. 11, pp. 3447–3461, Oct. 2012.

[13] D. Zordan, G. Quer, M. Zorzi, and M. Rossi, "Modeling and Generation of Space-Time Correlated Signals for Sensor Network Fields," in *IEEE GLOBECOM*, (Houston, Texas, US), Dec. 2011.

[14] M. Umer, L. Kulik, and E. Tanin, "Spatial interpolation in wireless sensor networks: localized algorithms for variogram modeling and Kriging," *Geoinformatica*, vol. 14, no. 1, pp. 101–134, 2010.

[15] P. Abrahamsen and N. Regnesentral, *A review of Gaussian random fields and correlation functions*. Norwegian Computing Center, Oslo, 1997.

[16] M. Chatterjee, S. K. Das, and D. Turgut, "An On-Demand Weighted Clustering Algorithm (WCA) for Ad Hoc Networks," in *IEEE GLOBECOM*, (San Francisco, CA, US), Nov. 2000.

[17] J. E. Barceló-Lladó, A. M. Pérez, and G. Seco-Granados, "Enhanced Correlation Estimators for Distributed Source Coding in Large Wireless Sensor Networks," *IEEE Sensors*, vol. 2, pp. 2799–2806, Sept. 2012.

[18] E. Candès and J. Romberg, "ℓ_1-MAGIC." Matlab routines for solving CS problems, 2005.

[19] W. Dai and O. Milenkovic, "Subspace Pursuit for Compressive Sensing Signal Reconstruction," *IEEE Transactions on Information Theory*, vol. 55, pp. 2230–2249, May 2009.

[20] S. Becker, J. Bobin, and E. J. Candès, "NESTA: A Fast and Accurate First-Order Method for Sparse Recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, 2011.

[21] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, "On the interplay between routing and signal representation for compressive sensing in wireless sensor networks," in *IEEE Workshop on Information Theory and Applications (ITA)*, (San Diego, CA, US), Feb. 2009.

[22] C. R. Rao, "The Use and Interpretation of Principal Component Analysis in Applied Research," *Sankhyā: The Indian Journal of Statistics*, vol. 26, pp. 329–358, Dec. 1964.

[23] N. Balakrishnan and C.-D. Lai, *Continuous Bivariate Distributions*. Springer, 2009.

[24] M. F. Duarte and R. G. Baraniuk, "Kronecker Compressive Sensing," *IEEE Transactions on Image Processing*, vol. 21, no. 2, pp. 494–504, 2012.

[25] M. Leinonen, M. Codreanu, and M. Juntti, "Sequential Compressed Sensing with Progressive Signal Reconstruction in Wireless Sensor Networks," *IEEE Transactions on Wireless Communications*, vol. 14, pp. 1622–1635, March 2015.

[26] E. H. Isaaks and R. M. Srivastava, *An Introduction to Applied Geostatistics*. Oxford University Press, 1990.

[27] R. P. Brent, *Algorithms for Minimization without Derivatives*. Dover Publications, 2013.

[28] D. Salomon, *Data Compression: The Complete Reference*. Springer, 2004.

[29] M. Srivastava, T. Abdelzaher, and B. Szymanski, "Human-centric Sensing," *Philosophical Transactions of Royal Society*, vol. 370, pp. 176–197, Jan. 2012.

[30] 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, June 2015.

[31] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, "A Review of Wearable Sensors and Systems with Application in Rehabilitation," *Journal of Neuroengineering and Rehabilitation*, vol. 9, pp. 1–13, Apr. 2012.

[32] R. Istepanian, S. Hu, N. Philip, and A. Sungoor, "The potential of Internet of m-health Things "m-IoT" for non-invasive glucose level sensing," in *IEEE International Conference of the Engineering in Medicine and Biology Society (EMBC)* , (Boston, MA, US), pp. 5264–5266, Aug. 2011.

[33] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, "Fusion of Smartphone Motion Sensors for Physical Activity Recognition," *MDPI Sensors*, vol. 14, pp. 10146–10176, June 2014.

[34] Zephyr Technology Corporation, "BioHarness 3 - Wireless Professional Heart Rate Monitor and Physiological Monitor," 2015.

[35] M. Hesse, P. Christ, T. Hormann, and U. Ruckert, "A Respiration Sensor for a Chest-Strap Based Wireless Body Sensor," in *Proceedings of IEEE Sensors*, (Valencia, Spain), pp. 490–493, Nov. 2014.

[36] Thought Technology Ltd, "Respiration Sensor: Model SA9311M," 2015.

[37] Seraphim Sense Ltd., "Angel open sensor wristband," 2015.

[38] Z. Zhang, Z. Pi, and B. Liu, "TROIKA: A General Framework for Heart Rate Monitoring Using Wrist-Type Photoplethysmographic Signals During Intensive Physical Exercise," *IEEE Transactions on Biomedical Engineering*, vol. 62, pp. 522–531, Feb. 2015.

[39] R. Mukkamala, J.-O. Hahn, O. T. Inan, L. K. Mestha, C.-S. Kim, H. Toreyin, and S. Kyal, "Toward Ubiquitous Blood Pressure Monitoring via Pulse Transit Time: Theory and Practice," *IEEE Transactions on Biomedical Engineering*, vol. 62, pp. 1879–1901, Aug. 2015.

[40] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow, and D. Estrin, "Lightweight Temporal Compression of Microclimate Datasets," in *Proceedings of the IEEE International Conference on Local Computer Networks (LCN)*, (Tampa, FL, US), pp. 516–524, Nov. 2004.

[41] C. R. Rao, "The Use and Interpretation of Principal Component Analysis in Applied Research," *Sankhyā: The Indian Journal of Statistics*, vol. 26, pp. 329–358, Dec. 1964.

[42] R. Shankara and I. S. N. Murthy, "ECG Data Compression Using Fourier Descriptors," *IEEE Transactions on Biomedical Engineering*, vol. 33, pp. 428–434, Apr. 1986.

[43] B. A. Rajoub, "An Efficient Coding Algorithm for the Compression of ECG Signals Using the Wavelet Transform," *IEEE Transactions on Biomedical Engineering*, vol. 49, pp. 355–362, Apr. 2002.

[44] L. F. Polania, R. E. Carrillo, M. Blanco-Velasco, and K. E. 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), pp. 761–764, May 2011.

[45] 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), pp. 80–85, Oct. 2014.

[46] D. Del-Testa and M. Rossi, "Lightweight Lossy Compression of Biometric Patterns via Denoising Autoencoders," *IEEE Signal Processing Letters*, vol. 22, pp. 2304–2308, Dec. 2015.

[47] R. Francescon, M. Hooshmand, M. Gadaleta, E. Grisan, S. K. Yoon, and M. Rossi, "Toward Lightweight Biometric Signal Processing for Wearable Devices," in *Proceedings of the International Conference of the IEEE Engineering in Medicine and Biology Society*, (Milan, Italy), Aug. 2015.

[48] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi, "From Today's INTRAnet of Things to a Future INTERnet Of Things: A Wireless- and Mobility- Related View," *Wireless Communications, IEEE*, vol. 17, pp. 44–51, December 2010.

[49] R. Yousefi, M. Nourani, S. Ostadabbas, and I. Panahi, "A Motion-Tolerant Adaptive Algorithm for Wearable Photoplethysmographic Biosensors," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, pp. 670–681, March 2014.

[50] Z. Zhang, Z. Pi, and B. Liu, "TROIKA: A General Framework for Heart Rate Monitoring Using Wrist-Type Photoplethysmographic Signals During Intensive Physical Exercise," *IEEE Transactions on Biomedical Engineering*, vol. 62, pp. 522–531, February 2015.

[51] "Zephyr BioHarness3." `http://zephyranywhere.com/products/bioharness-3/`.

[52] D. Zordan, B. Martinez, I. Vilajosana, and M. Rossi, "On the Performance of Lossy Compression Schemes for Energy Constrained Sensor Networking," *ACM Transactions on Sensor Networks*, vol. 11, pp. 1–34, November 2014.

[53] J. Pan and W. J. Tompkins, "A Real-Time QRS Detection Algorithm," *IEEE Transactions on Biomedical Engineering*, vol. 32, pp. 230–236, March 1985.

[54] B.-U. Kohler, C. Hennig, and R. Orglmeister, "The Principles of Software QRS Detection," *IEEE Engineering in Medicine and Biology Magazine*, vol. 21, pp. 42–57, January 2002.

[55] M. Elgendi, B. Eskofier, S. Dokos, and D. Abbott, "Revisiting QRS Detection Methodologies for Portable, Wearable, Battery-Operated, and Wireless ECG Systems," *PloS ONE*, vol. 9, p. e84018, January 2014.

[56] J. Allen, "Photoplethysmography and its application in clinical physiological measurement," *IOP Physiological Measurement*, vol. 28, pp. 1–39, February 2007.

[57] A. K. Jain, R. Chellappa, S. C. Draper, N. Memon, P. J. Phillips, and A. Vetro, "Signal Processing for Biometric Systems," *IEEE Signal Processing Magazine*, vol. 24, pp. 146–152, Nov. 2007.

[58] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity Recognition using Cell Phone Accelerometers," *ACM SIGKDD Explorations Newsletter*, vol. 12, pp. 74–82, Dec. 2010.

[59] M. Elgendi, B. Eskofier, S. Dokos, and D. Abbott, "Revisiting QRS Detection Methodologies for Portable, Wearable, Battery-Operated, and Wireless ECG Systems," *PLoS ONE*, vol. 9, p. e84018, Jan. 2014.

[60] P. Morizet-Mahoudeaux, C. Moreau, D. Moreau, and J. Quarante, "Simple microprocessor-based system for on-line e.c.g. arrhythmia analysis," *Medical and Biological Engineering and Computing*, vol. 19, pp. 497–500, Jul. 1981.

[61] M. L. Ahlstrom and W. J. Tompkins, "Automated High-Speed Analysis of Holter Tapes with Microcomputers," *IEEE Transactions on Biomedical Engineering*, vol. 30, pp. 651–657, Oct. 1983.

[62] Y. Chen and H. Duan, "A QRS Complex Detection Algorithm Based on Mathematical Morphology and Envelope," in *Proceedings of the International Conference of the IEEE Engineering in Medicine and Biology Society*, (Shanghai, China), pp. 4654–4657, Jan. 2005.

[63] F. Zhang and Y. Lian, "QRS Detection Based on Multiscale Mathematical Morphology for Wearable ECG Devices in Body Area Networks," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 3, pp. 220–228, Aug. 2009.

[64] V. Afonso, W. J. Tompkins, T. Nguyen, and S. Luo, "ECG Beat Detection Using Filter Banks," *IEEE Transactions on Biomedical Engineering*, vol. 46, pp. 192–202, Feb. 1999.

[65] H. Dinh, D. Kumar, N. Pah, and P. Burton, "Wavelets for QRS Detection," in *Proceedings of the International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 2, (Istanbul, Turkey), pp. 1883–1887, Oct. 2001.

[66] Q. Xue, Y. H. Hu, and W. J. Tompkins, "Neural-Network-Based Adaptive Matched Filtering for QRS Detection," *IEEE Transactions on Biomedical Engineering*, vol. 39, pp. 317–329, Apr. 1992.

[67] L.-Y. Shyu, Y.-H. Wu, and W. Hu, "Using Wavelet Transform and Fuzzy Neural Network for VPC Detection from the Holter ECG," *IEEE Transactions on Biomedical Engineering*, vol. 51, pp. 1269–1273, Jul. 2004.

[68] D. T. Kaplan, "Simultaneous QRS Detection and Feature Extraction using Simple Matched Filter Basis Functions," in *Proceedings of IEEE Computers in Cardiology*, (Chicago, IL, US), pp. 503–506, Sep. 1990.

[69] 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. 15, pp. 128–129, Apr. 1968.

[70] J. P. Abenstein and W. J. Tompkins, "A New Data-Reduction Algorithm for Real-Time ECG Analysis," *IEEE Transactions on Biomedical Engineering*, vol. 29, pp. 43–48, Jan. 1982.

[71] F. Castells, P. Laguna, L. Sörnmo, A. Bollmann, and J. M. Roig, "Principal Component Analysis in ECG Signal Processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. 1–21, Feb. 2007.

[72] 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, pp. 556–564, May 1999.

[73] 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, pp. 15:1–15:34, Nov. 2014.

[74] N. Maglaveras, T. Stampkopoulos, K. Diamantaras, C. Pappas, and M. Strintzis, "ECG pattern recognition and classification using non-linear transformations and neural networks: A review," *International Journal of Medical Informatics*, vol. 52, pp. 191–208, Oct. 1998.

[75] C. C. Sun and S. C. Tai, "Beat-based ECG compression using gain-shape vector quantization," *IEEE Transactions on Biomedical Engineering*, vol. 52, pp. 1882–1888, Nov. 2005.

[76] 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, pp. 300–309, Feb. 2013.

[77] J. Cardenas-Barrera and J. Lorenzo-Ginori, "Mean-Shape Vector Quantizer for ECG Signal Compression," *IEEE Transactions on Biomedical Engineering*, vol. 46, pp. 62–70, Jan. 1999.

[78] 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, pp. 55–70, May 2015.

[79] A. Johansson, P. Öberg, and G. Sedin, "Monitoring of Heart and Respiratory Rates in Newborn Infants Using a New Photoplethysmographic Technique," *Journal of Clinical Monitoring and Computing*, vol. 15, pp. 461–467, Dec. 1999.

[80] K. Chon, S. Dash, and K. Ju, "Estimation of Respiratory Rate From Photoplethysmogram Data Using Time–Frequency Spectral Estimation," *IEEE Transactions on Biomedical Engineering*, vol. 56, pp. 2054–2063, Aug. 2009.

[81] J. Lin, E. Keogh, S. Lonardi, and P. Patel, "Finding motifs in time series," in *Proceeding of the 2nd Workshop on Temporal Data Mining*, SIGKDD '02, (New York, NY, USA), pp. 53–68, ACM, 2002.

[82] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A Symbolic Representation of Time Series, with Implications for Streaming Algorithms," in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, (New York, NY, USA), pp. 2–11, ACM, 2003.

[83] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, Massachusetts: Kluwer Academic Publishers, 1992.

[84] M. Danieletto, *Design and Evaluation of Compression, Classification and Localization Schemes for Various IoT Applications*. PhD thesis, University of Padova, January 2014.

[85] M. Danieletto, N. Bui, and M. Zorzi, "RAZOR: A Compression and Classification Solution for the Internet of Things," *Sensors*, vol. 14, no. 1, pp. 68–94, 2013.

[86] M. Danieletto, N. Bui, and M. Zorzi, "RAZOR: A Compression and Classification Solution for the Internet of Things," *Sensors*, vol. 14, pp. 68–94, Jan. 2014.

[87] N. M. Arzeno, Z.-D. Deng, and C.-S. Poon, "Analysis of First-Derivative Based QRS Detection Algorithms," *IEEE Transactions on Biomedical Engineering*, vol. 55, pp. 478–484, Feb. 2008.

[88] M. Elgendi, "Fast QRS Detection with an Optimized Knowledge-Based Method: Evaluation on 11 Standard ECG Databases," *PLoS ONE*, vol. 8, pp. 1–18, Sep. 2013.

[89] E. Keogh and C. A. Ratanamahatana, "Exact Indexing of Dynamic Time Warping," *Knowledge and Information Systems*, vol. 7, pp. 358–386, Mar. 2005.

[90] S. Salvador and P. Chan, "Toward Accurate Dynamic Time Warping in Linear Time and Space," *Journal of Intelligent Data Analysis*, vol. 11, pp. 561–580, Oct. 2007.

[91] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA, US: Kluwer Academic Publishers, 1991.

[92] Y. Linde, A. Buzo, and R. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. 28, pp. 84–95, Jan. 1980.

[93] S. C. Tai, *Adaptive Sampling of One Dimensional Digital Signal.* No. 083501, Jan. 1997.

[94] J. Karhunen and J. Joutsensalo, "Generalizations of Principal Component Analysis, Optimization Problems, and Neural Networks," *Neural Networks*, vol. 8, pp. 549–562, Jan. 1995.

[95] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, pp. 1–127, Jan. 2009.

[96] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, Dec. 2010.

[97] E. Candès and J. Romberg, "Sparsity and Incoherence in Compressive Sampling," *Inverse Problems*, vol. 23, pp. 969–985, Nov. 2007.

[98] R. Baraniuk, "Compressive Sensing [Lecture Notes]," *IEEE Signal Processing Magazine*, vol. 24, pp. 118–121, Jul. 2007.

[99] Z. Zhang and B. D. Rao, "Extension of SBL Algorithms for the Recovery of Block Sparse Signals with Intra-Block Correlation," *IEEE Transactions on Signal Processing*, vol. 61, pp. 2009–2015, Apr. 2013.

[100] A. Jensen and A. la Cour-Harbo, *Ripples in Mathematics: the Discrete Wavelet Transform.* Springer, 2001.

[101] J. Yan, "Wavelet matrix." Technical Report, University of Victoria, Nov. 2009.

[102] N. G. Prelcic and O. W. M. Flrez, "UVI WAVE: the ultimate toolbox for wavelet transforms and filter banks," in *Workshop on Intelligent Methods for Signal Processing and Communications*, (Bayona, Vigo, Spain), pp. 224–227, June 1996.

[103] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Simultaneous Sparse Approximation via Greedy Pursuit," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, (Philadelphia, PA, US), Mar. 2005.

[104] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, "Compressed Sensing for Real-Time Energy-Efficient ECG Compression on Wireless Body Sensor Nodes," *IEEE Transactions on Biomedical Engineering*, vol. 58, pp. 2456–2466, Sept. 2011.

[105] S. Lee, J. Kim, and J.-H. Lee, "A Real-Time ECG Data Compression and Transmission Algorithm for an e-Health Device," *IEEE Transactions on Biomedical Engineering*, vol. 58, pp. 2448–2455, Sep. 2011.

[106] H. Lee and K. M. Buckley, "Heart Beat Data Compression Using Temporal Beats Alignment and 2-D Transforms," in *Conference Record of the Thirtieth Asilomar Conference on Signals, Systems and Computers*, vol. 2, (Pacific Grove, CA, US), pp. 1224–1228, Nov. 1996.

[107] V. Lukin, M. Zriakhov, A. Zelensky, K. Egiazarian, and A. Varri, "Lossy Compression of Multichannel ECG Based on 2-D DCT and Pre-processing," in *Proceedings of International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science*, (Lviv-Slavske, Ukraine), pp. 159–162, Feb. 2008.

[108] V. Allen and J. Belina, "ECG Data Compression Using the Discrete Cosine Transform (DCT)," in *Proceedings of Computers in Cardiology*, (Durham, NC, US), pp. 687–690, Oct. 1992.

[109] M. Alam and N. Rahim, "Compression of ECG Signal Based on its Deviation from a Reference Signal Using Discrete Cosine Transform," in *Proceedings of the International Conference on Electrical and Computer Engineering*, (Dhaka, Bangladesh), pp. 53–58, Dec. 2008.

[110] E. Anas, M. Hossain, M. Afran, and S. Sayed, "Compression of ECG Signals Exploiting Correlation between ECG Cycles," in *Proceedings of the International Conference on Electrical and Computer Engineering*, (Dhaka, Bangladesh), pp. 622–625, Dec. 2010.

[111] C. Karakus, A. C. Gurbuz, and B. Tavli, "Analysis of Energy Efficiency of Compressive Sensing in Wireless Sensor Networks," *IEEE Sensors Journal*, vol. 13, pp. 1999–2008, May 2013.

[112] M. Hooshmand, M. Rossi, D. Zordan, and M. Zorzi, "Covariogram-based Compressive Sensing for Environmental Wireless Sensor Networks," *IEEE Sensors Journal*, vol. 16, pp. 1716–1729, Mar. 2016.

[113] ARM The Architecture for the Digital World, "ARM Cortex-M4 Processor," 2015.

[114] Texas Instruments, "CC2451: 2.4 GHz Low Energy and Proprietary System-on-Chip," 2015.

[115] M. Blanco-Velascoa, F. Cruz-Roldána, J. I. Godino-Llorenteb, J. Blanco-Velascoa, C. Armiens-Aparicioa, and F. López-Ferreras, "On the Use of PRD and CR Parameters for ECG Compression," *Elsevier Medical Engineering & Physics*, vol. 27, pp. 798–802, Nov. 2005.

[116] 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, pp. 952–960, May 2011.

[117] Y. Zigel, A. Cohen, and A. Katz, "ECG Signal Compression Using Analysis by Synthesis Coding," *IEEE Transactions on Biomedical Engineering*, vol. 47, pp. 1308–1316, Oct. 2000.

[118] T. Kohonen, "The Self-Organizing Map," *Proceedings of the IEEE*, vol. 78, pp. 1464–1480, Sept. 1990.

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

[120] 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), pp. 422–427, Mar. 2000.

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

[122] 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.

[123] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'Neural-gas' Network for Vector Quantization and Its Application to Time-series Prediction," *Trans. Neur. Netw.*, vol. 4, pp. 558–569, July 1993.

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