ECG Compression Based on Successive Differences

M. Gusev*+, M. Jovanov*, A. Shekerov*, G. Temelkov+
* Sts Cyril and Methodius University in Skopje, Faculty of Computer Science and Engineering
+ Innovation Dooel, Skopje, North Macedonia
E-mail: {marjan.gushev, mile.jovanov}@finki.ukim.mk, andrej.shekerov@students.finki.ukim.mk,
goran.temelkov@innovation.com.mk

***Abstract*—This paper aims to reduce the amount of physical space the electrocardiogram data occupies and needs to be transferred, designing an efficient algorithm that can run on hardware commonly found in a medical device setting. The algorithm needs to be compatible with real-time data transfer and be fast and lossless. We introduce an algorithm inspired by pulse code modulation in telecommunication theory based on successive differences between neighboring electrocardiogram samples. The research hypothesis addressed in this paper finds if the new algorithm can reduce the data size by a meaningful amount and efficiently. Further on we tackle the research questions to find the level of distortion, compression, and processing speed of compression and decompression, along with the analysis of benefits and disadvantages. We reduced the storage space or bits to be transferred up to 46% reaching a compression ratio of 2.17 for an electrocardiogram benchmark dataset of up to 100 Mbits.**

***Keywords*—ECG, Sampling, Analog to digital conversion, ADC, Compression, Consecutive differences**

I. Introduction

Advances in wearable technologies enable small medical devices to measure an electrocardiogram (ECG) in real time. The convenient sampling frequency to provide sufficient and relevant data for medical experts and automated Artificial Intelligence (AI) algorithms are in the range between 125 and 500 Hz, although some professional devices use sampling frequencies up to 1000 Hz, while the bit resolution is in the range between 10 and 16 bits.

The problems associated with systems for data transfer and real-time processing are classified in the Big Data domain, since data comes in high velocity generating large volumes. A simple calculation shows that such a system produces from 10,800,000 samples or 108 Mbits per day up to 86,400,000 samples or 1.38 Gbits per day. Considering that date/time stamp maybe accompanied to ECG data, the amount of data to be transferred and processed is from 20 MB up to 180 MB per day.

CardioHPC [1] is a project to realize an experiment for High Performance Computing (HPC) data processing center for thousands of real-time ECG streams. Such a system requires huge storage capacity and any savings with lossless compression are beneficial and highly appreciated. In addition, this method allows transferring of a smaller number of bits of the end-user device (ECG sensor) to the nearby smartphone or edge/dew device, which consumes less energy and saves the battery life. The related application [2] is a software solution [3] to detect arrhythmia or other dangerous heart conditions.

In this paper, we present an algorithm to compress ECG data based on pulse code modulation concepts from telecommunication theory encoding the differences between successive ECG samples. The idea is to build an algorithm that will efficiently store only successive differences in a specific way without losing data.

The purpose of this algorithm is to help with the transfer, storage, and real-time practical application of ECG data since the amount of storage space that needs to be allocated for even a day's worth of data can be overwhelming. With our algorithm, the storage space required to store this data can be reduced by more than half in some cases, and by 46% on average.

We realize an experimental method to find if the newly developed algorithm can efficiently reduce the data size by a meaningful amount. In addition, we aim at finding the level of distortion, compression, and processing speed of compression and decompression, along with the analysis of benefits and disadvantages, specifying optimization as the research question.

Related work is presented in Section II and methods in Section III with explanation of the algorithm the experiment and evaluation methodology. Implementation issues, benefits and disadvantages, along with a comparison to other tools are discussed in Section V. Finally, conclusions are presented in Section VI.

II. Related work

The need for compressing ECG data has been around for a long time. The earliest methods make use of linear interpolation and Huffman encoding, and some more modern methods apply approaches such as neural networks, wavelets, and compatibility with portable devices.

A. Overview

One of the earliest papers that tackled ECG data compression are reported when tape-less ECG recorders were starting to become feasible. Moody et al. [4] introduce compression by reducing the amount of redundant information which arise from a lack of statistical independence between symbols. Another relevant topic discussed is regarding "lossless" compression, where the accuracy of

the compression depends on the clinical interpretation; a change in the data can range from insignificant to extremely significant, depending on the context.

Singh et al. [5] review a few ECG data compression techniques, including direct, transform, parameter extraction, 2D transformation, etc. The authors introduce several evaluation parameters (compression ratio, percent mean square, difference, etc.) to compare and evaluate the different approaches. A key observation form their study is that even though there are a lot of ECG compression techniques, not so many are utilized in real monitoring systems and telemedicine, mainly due to fear of recovery from the distortions.

Another comprehensive review [6] presents various ECG compression implementations, using similar evaluation parameters. In addition, the authors analyze some effective algorithms (such as the wavelet transform method) and conclude that, due to their complexity, they haven't been implemented in real-life scenarios.

An overview of ECG compression methods suitable for specific scenarios is analyzed [7] from different evaluation perspectives. They conclude that threshold-based algorithms require low computational cost, transform-based methods are good for scenarios with a lot of noise due to their insensitiveness, etc.

A methodological review of different ECG data compression techniques [8], is reported with implementation of real-world data to evaluate the analyzed methods by several evaluation metrics. An interesting conclusion is that although simple direct methods achieve a smaller compression ratio, due to their simplicity and ease of implementation are most suitable for Internet of Things (IoT) health-care applications.

ECG compression techniques for wireless ECG devices have been analyzed in [9] with a focus on two direct data compression methods: delta coding and Huffman coding, as well as their variations. The experiments were checked on a smaller proprietary dataset and selected MITDB records achieving a compression ratio of 1.6. A combination of a second-order delta encoding and run-length encoding (RLE) based data compression is proposed [10] for ECG and PPG compression after quantizing using optimal quantization level to achieve a lesser number of representation bits.

Variable-length code was applied in developing an electronic circuit [11] on determined four regions of an ECG (initial region, QRS region, flat region and unrest region) adaptively predicting (detecting) unique waveform features, followed by a variable length code.

These methods differ from our approach which uses delta coding and variable-length code on successive differences.

B. Specialized techniques

An overview of a few known compression methods [12] states the need and benefits of standardization in terms of ECG compression evaluation methods. Due to a lack of any standardized way to make comparisons between different algorithms and manufacturer equipment, the clinical engineering community have no way of direct comparison of equipment and services.

There are several algorithms that utilize wavelets to compress the data [13]–[16] and belong to the category of transform methods, which are generally considered to be very effective. Among a variety of wavelet types, notably scalar wavelets and multiwavelets are generally more optimal.

Specialized techniques are often utilized when designing a new compression algorithm, and 2D transformation [17] has been used to compress ECG data in an efficient manner. Additionally, it attempts to highlight the current and future issues regarding the development of suitable ECG data compression techniques.

A more modern take on the problem - a compression algorithm that is based both on wavelets and on neural networks, and manages to achieve a very respectable compression ratio using these techniques [18]. This algorithm uses self-learning to gradually improve it's accuracy and reliability, and manages to achieve results that exceed other wavelet-based methods.

One can use direct methods [19] with the goal of finding an algorithm that is most suitable for telemedicine. The algorithms presented in this paper need to be transferred over a network reliably and efficiently, so they had additional requirements of security and the goal of a higher network transfer rate.

Jacobi polynomials have also been used in an ECG compression algorithm [20] to achieve interesting results compared to wavelet compression methods. The results of this algorithm could be greatly improved, since the procedure shown here is subject to rounding errors that pile up and eventually give modified results.

C. Special purpose

Development of ECG compression algorithms for portable devices [21] and Bluetooth are two specialized scenarios that require additional attention. These algorithms are generally suitable for portable devices and embedded systems with limited computing power. Some of the more interesting pre-processing methods listed here include filtering, down-sampling and peak-detection. An analysis [22] is given of some existing methods to find ones which are suitable to be used in an remote environment that uses Bluetooth to transfer the data. Important properties of the algorithms are latency, error-tolerance and power-consumption. Due to the limited processing and transmission power of small devices, these algorithms should not be very complex.

Lossless ECG data compression algorithms [23] are preferable to lossy methods due to the importance of the data that is being transmitted; medical regulatory board worldwide advocate for lossless compression. Since a

change in the original data may have grave consequences on the well-being of patients, ECG compression algorithms should aim to be lossless, or to at least have an acceptable amount of distortion.

An evaluation of a few known ECG compression methods [24], based on a few parameters. The authors mention the need for a standard way for comparing compression algorithms, and give a brief overview and recommendation on how to measure the performance on new methods.

III. Methods

Further on, we present the new compression algorithm and describe the experimental and evaluation methodology.

A. Algorithm

The ECG data stream contains a series of integer samples. Each ECG sample is compared to the preceding sample to determine the successive difference. The successive difference represents only an increase or decrease of an ECG sample from the previous sample. The sampling frequency determines the magnitude of the differences, and it determines the code for compression. The idea is that the code will be chosen to represent the average magnitude of the differences (or to encompass the most of cases).

The new ECG compression based on successive differences takes advantage of the fact that successive ECG values are usually similar, so it is better to store the difference between successive values instead of the values themselves. Still an open question arises to detect the efficient algorithm and the optimal number of bits to allocate for each value. Determination of the optimal number of bits to store the difference is not the only issue, we need to find a method to deal when the difference is bigger than the space we have allocated for it.

To find the optimal number of bits, we conduct an experiment by specifying test cases with different number of allocated bits and observe the achieved compression ratio. The testing domain is in the range from 2 bits to 7 bits.

This concept is implementation of the differential pulse-code modulation which encodes the differences between successive samples into n-bit data streams used in telecommunications or in analog-to-digital conversion, where the code modulation with 1-bit data stream is also known as the Delta-sigma $\Delta\Sigma$ modulation. This or other forms of pulse code modulations simply reject cases when successive difference is larger than the selected code, while in our algorithm we implement a solution to cope with these differences and do not lose any data in the compression.

The code to be selected is supposed to contain the successive difference and also the sign considering that the difference may be positive or negative. The minimal code to be used is 2 bits, one is the sign and the other for the difference. In a general case, the code of n bits will be able to store successive differences in the range from $-2^{(n-1)}$ up to $+2^{(n-1)}$.

The method to decide how to deal with successive differences is based on the no data loss concept, so we develop an algorithm that uses a special code to write a normal ECG sample instead of a successive difference.

The algorithm starts with a predefined zero value (the mid point in the selected bit-resolution). Then, a repetitive procedure analyzes the ECG values one by one in a sequence, to calculate the difference between the values of the current sample and the previous sample. If the calculated difference is small enough to fit into the space determined by the code, then it is stored with the sign and absolute value. The cases when the calculated difference is out of the range determined with a magnitude of $(n-1)$ bits, then a special code is stored, usually interpreted as negative zero (-0), and then store the real ECG sample instead of the successive difference.

B. Experiment

In our experiment, we use 48 ECG records from the MITDB ECG benchmark dataset. This collection of 48 ECG records totals 46.8 MB for 650000 samples per record each stored as a 12-bit sample. We have excluded the four records for patients with paced beats, so our dataset contains only 44 ECG records. Note that although the original declaration of the MITDB is 12 bit analog-to-digital conversion, still data are presented in 11-bit resolution.

Besides conducting the first *Experiment A* on the original dataset that uses the original sampling frequency of 360 Hz and bit resolution of 11 bits we have resampled the ECG records from 360 to 125 Hz, and rescaled from 11 to 10 bits and conduct an *Experiment B* on a second smaller dataset with a total of 13.5 MB. This resampling and rescaling achieves a compression ratio of 3.456. Although this method loses specific data features, still it is proven [25], [26] to be efficient in detecting the main heartbeat features and classifying different arrhythmia. Note that a very small number of values in the MITDB were found out of a 10-bit range, for specific signal inytervals of a pysically active person, with a lot of noise and artefacts, and these values were simply truncated.

The starting point in experiment A is the mid value of 1023 (midpoint of 11-bit resolution) and for the Experiment B the starting point is 511 (midpoint of 10-bit resolution). Experiment A stores original data of 11 bits after specifying a special code (-0) when the successive difference is larger then the selected code (number of bits), and Experiment B stores only 10 bits.

Test cases cover codes (number of bits) from 2 up to 7 bits. The testing environment that the algorithm is executed and tested is a system running with g++ version 12.1.0, Intel® Core™ i7-9750H processor, 16GB of RAM.

Additionally, to make sure that the compression is correct, the algorithm does a self-check to decompress the

data and compare it to the initial input. This is also used to measure decoding time.

C. Evaluation methodology

Additionally, this algorithm is lossless, meaning that the data should remain identical to the original after decompression. Due to this there is no need to measure the distortion. Considering the nature of lossless algorithms, the evaluation metrics [5] that compare different compression methods based on the magnitude of errors, such as Percent Mean Square Difference (PRD); Percent Root Mean Square Difference Normalized (PRDN); Quality Score (QS); Root Mean Square Error (RMS), etc. are not applicable. Our compression algorithm does not generate an error due to compression, these metrics are PRD=0, PRDN=0, RMS=0 and $QS \rightarrow \infty$.

A valid evaluation measure is the *compression ratio*, calculated as a ratio between the uncompressed size and compressed size. In our experiments, we will measure the the sizes of the benchmark dataset and compare to the size of the compressed data. In addition, we will express the reciprocal value of the compression ratio to reveal the capacity of the reduced size.

Additionally, we measure the time the algorithm takes in milliseconds for each test case and for each number of bits. This time includes reading the data, compressing the data, writing the data to a file, and then verifying the compression by reading the compressed data from a file and comparing it to the initial values we had for the test case. In practice the verification could be excluded to save on I/O and computing time, but here it is included to make sure that the algorithm gives us correct values. To make a real comparison this test is also executed with uncompressed data (original dataset).

IV. Results

Table I and Table II contain details on the number of analyzed bits as a sum and average per record, results of achieved capacity in percentage of the number of bits in the uncompressed dataset, and the compression rate (total, minimum, maximum, average and standard deviation) for selected 44 records from MITDB without records with paced beats. Table I refers to Experiment A for data sampled on the original 360 Hz sampling frequency and 11-b resolution, while Table II to Experiment B for data resampled to a 125 Hz sampling frequency and rescaled to 10-bits.

The best compression rate is achieved for 4 bits regardless the sampling frequency (applied for both experiments), reaching a value of 2.17 and 1.87 for Experiments A and B correspondingly.

Processing time (measured in milliseconds) is presented in Table III. The fastest compression is again achieved by using 4 bits, and the average time this per case is 123.85ms, whereas without compression the data is processed with an average of 108.96ms per case.

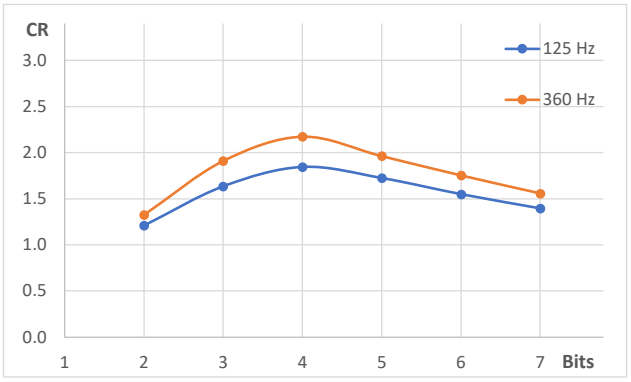

Fig. 1: Measured compression ratio in the experiments using data sampled at 360 Hz and 125 Hz

V. Discussion

Even though the algorithm works similarly regardless of the number of bits to allocate the space for successive differences, the compression achieved depends on this number. We have found that using 4 bits for this compression is the optimal number of bits in most cases. There are a few exceptions where 3 bits or 5 bits are better by a small margin, and even a case where using 6 bits is optimal, but the 4-bit compression has the lowest overall average and lowest total sum across all our tests. Additionally, the compression always performs better than the non-compressed data, except for 2-bit compression, which only falls behind on 4% of cases. The compression does take extra time to compute, but on average the fastest compression (depending on bits we use) is only 13.71 milliseconds behind the non-compressed data.

Many other methods of ECG compression use bit manipulation as part of their algorithm to help reduce the size of the data. Here we have provided a simple yet effective way to reduce the data in a lossless way. The data generated from this method could then be further compressed by some of the other methods instead of the raw ECG data.

The benefits of using this algorithm over others is the fact that our algorithm has no distortion; the original signal can always be recovered, and quite efficiently too as seen from the results section. The time it takes the algorithm to compress and decompress the data only slightly exceeds the time it takes for the raw data to be processed. Additionally, the algorithm is very simple and doesn't require much processing power to run, being able to be developed in a low-level language such as C, C++ or other similar languages.

Figure 1 presents the dependence of the achieved compression ratio versus different code (number of selected bits) in our algorithm.

Analysing the original 360 Hz sampling frequency (Experiment A dataset) and the sampling frequency of 125 Hz for resampled data (Experiment B dataset) we confirm the theoretical conclusion that the higher the sampling

TABLE I: Achieved compression rates for 44 MITDB ECG records in Experiment A with data sampled on 360 Hz with a 11-bit resolution.

	Total bits			**Compression Rate (CR)**				
Method	Sum	Avg	Capacity	Total	MIN	MAX	AVG	STD
No C.	314600000	7150000						
2 Bits	237733276	5403029	75.57%	1.32	1.07	1.70	1.34	0.15
3 Bits	164798403	3745418	52.38%	1.91	1.31	2.56	1.96	0.33
4 Bits	144762464	3290056	46.01%	2.17	1.80	2.46	2.19	0.17
5 Bits	160336044	3644001	50.97%	1.96	1.76	2.14	1.97	0.08
6 Bits	179489937	4079316	57.05%	1.75	1.60	1.83	1.75	0.05
7 Bits	202024306	4591461	64.22%	1.56	1.50	1.57	1.56	0.02

TABLE II: Achieved compression rates for 44 MITDB ECG records in Experiment B with data resampled on 125 Hz with a 10-bit resolution.

	Total bits			**Compression Rate (CR)**				
Method	Sum	Avg	Capacity	Total	MIN	MAX	AVG	STD
No C.	99305360	2256940						
2 Bits	81843372	1860077	82.42%	1.21	0.97	1.55	1.23	0.13
3 Bits	60081008	1365477	60.50%	1.65	1.06	2.26	1.70	0.29
4 Bits	53088234	1206551	53.46%	1.87	1.31	2.14	1.89	0.20
5 Bits	57243710	1300993	57.64%	1.73	1.56	1.87	1.74	0.08
6 Bits	64025726	1455130	64.47%	1.55	1.43	1.66	1.55	0.05
7 Bits	71216762	1618563	71.71%	1.39	1.30	1.43	1.39	0.03

TABLE III: Processing time for compression of 44 MITDB ECG records in Experiment B with data sampled on 125 Hz (measured in milliseconds)

Name	Sum	Avg
No comp.	5230	108.96
2 Bits	6878	143.29
3 Bits	6428	133.92
4 Bits	5945	123.85
5 Bits	6093	126.94
6 Bits	6115	127.40
7 Bits	6380	132.92

frequency is, the smaller differences between neighbours and the higher the compression rate is achieved.

The standard deviation calculated on different records shows small deviations, not just in absolute, but also in relative value, in the range from 2.0% (for 7 bits) - 17.6% (for 3 bits) for Experiment B (dataset sampled on 125 Hz), and from 1.2% (for 7 bits) - 17.1% (for 3 bits) for Experiment A (dataset sampled on 360 Hz). The best compression achieved for a 4-bit code results in 10.5% and 7.8% relative standard deviation correspondingly for Experiments B and A.

Note that several published approaches, including simple successive difference and Huffman coding [9] or their variations, or a combination of delta encoding and RLE [10] are different from our approach. Merdjanovska et al. [9] use successive differences followed by Huffman coding as a very popular lossless coding allocating specific codes to repeating data values. The authors do not expose the applied code table and conclude that a combination of Huffman and delta coding is worth exploring. In this paper, we use a variant of variable-length coding based on allocating a fixed code for the most frequent values (for example, successive differences of 1, 0, and -1) and if the successive difference is outside the values covered with the code then the successive difference is stored. This is quite different from Huffman code, which builds a binary tree for all the values. In our case, we do not build a binary or other tree and use an n-bit code to store both positive and negative numbers and the code associated to a negative zero is the special code to represent the successive difference.

Banerjee et al. [10] use successive differences of second order (difference of differences) instead of our approach using only first order differences. The authors apply quantization to different numbers of bits and in our approach we use conversion from 11 to 10 bits for the MITDB example. Repetition of zeros (a sequence of several zeros) is eliminated using the RLE, and in. our approach, we use a completely different approach to variable-length code. The proposed combination of second-order delta encoding is limited by the space allocated to represent the maximum absolute value (MAV) which was estimated to be 6 bits, out of which one bit is for a sign, meaning that the quantization was done to a smaller number of bits, generating representation errors.

These approaches generate a modified (distorted) signal within the compression algorithm and thus produce $PRD > 0$. Therefore, a direct comparison of our algorithm to other algorithms is not possible since our algorithm does not distort the signal (PRD=0). We use successive differences since their magnitude is much smaller than the ECG value, and the histogram of their representation follows a normal Gaussian distribution, such that our code that maps successive differences to a variable number of bits, allows compression with zero error (lossless data compression) and still be read back symbol by symbol.

Some of the following features might be considered as disadvantages:

- Misinterpretation of a code might generate a nonrecoverable error, solvable by an integration of time

stamps within each new data chunk.
- Variable bandwidth is expected with such a code to achieve higher compression against fixed bandwidth.
- The worst case scenario of sending noise with maximal alternating successive differences (not an ECG signal) will result in $CR \leq 1$.

VI. Conclusion

There are many ways in which ECG data can be compressed, and depending on the needs of the situation different algorithms have different strengths and weaknesses. We introduced a new compression algorithm for ECG measurements, based on pulse code modulation techniques exploited in telecommunication theory, focusing on successive differences between neighbouring ECG samples. In addition to the conventional approach in the pulse code modulation methods, we implement a new code that will save the original value of the successive difference instead of cutting the value to fit to the pulse code value. This approach allows a suitable implementation and flexibility without loosing in quality or distorting the data values that represent the signal.

Besides requiring a simple implementation our algorithm is effective and efficient (CR=1.6 and PRD=0) suitable in any scenario where the ECG data needs to be transferred or stored without any data loss. This is especially important to save energy for end-user devices and sensors which need to save energy and prolong battery life. Our future work addresses the dependence of the achieved compression rate with a wide range of different sampling frequencies, evaluating not only the compression achieved with our algorithm but also with reducing the sampling frequency.

Acknowledgement

The experiment "CardioHPC - Improving DL-based Arrhythmia Classification Algorithm and Simulation of Real-Time Heart Monitoring of Thousands of Patients" has received funding from the European High-Performance Computing Joint Undertaking (JU) through the FF4EuroHPC project under grant agreement No 951745. The JU receives support from the European Union's Horizon 2020 research and innovation program and Germany, Italy, Slovenia, France, and Spain.

References

[1] Innovation Dooel, "CardioHPC project "Real-time heart monitoring of thousands of patients"," online, 2023, as seen on 01.02.2023. [Online]. Available: http://cardiohpc.innovation.com.mk/

[2] M. Gusev, A. Stojmenski, and A. Guseva, "ECGalert: A heart attack alerting system," in *ICT Innovations 2017: Data-Driven Innovation. 9th International Conference, ICT Innovations 2017, Skopje, Macedonia, September 18-23, 2017, Proceedings 9*. Springer, 2017, pp. 27–36.

[3] E. Domazet and M. Gusev, "Improving the qrs detection for one-channel ECG sensor," *Technology and Health Care*, vol. 27, no. 6, pp. 623–642, 2019.

[4] G. B. Moody, K. Soroushian, and R. G. Mark, "ECG data compression for tapeless ambulatory monitors," *Computers in Cardiology*, vol. 14, pp. 467–470, 1987.

[5] B. Singh, A. Kaur, and J. Singh, "A review of ECG data compression techniques," *International journal of computer applications*, vol. 116, no. 11, 2015.

[6] S. B. Kale and D. H. Gawali, "Review of ECG compression techniques and implementations," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. IEEE, 2016, pp. 623–627.

[7] S. O. Rajankar and S. N. Talbar, "An electrocardiogram signal compression techniques: a comprehensive review," *Analog Integrated Circuits and Signal Processing*, vol. 98, no. 1, pp. 59–74, 2019.

[8] C. K. Jha and M. H. Kolekar, "Electrocardiogram data compression techniques for cardiac healthcare systems: A methodological review," *IRBM*, 2021.

[9] E. Merdjanovska, M. Mohorcic, M. Depolli, A. Rashkovska, and T. Javornik, "Data compression for wireless ecg devices." in *BIOSIGNALS*, 2022, pp. 15–21.

[10] S. Banerjee and G. K. Singh, "A new real-time lossless data compression algorithm for ecg and ppg signals," *Biomedical Signal Processing and Control*, vol. 79, p. 104127, 2023.

[11] K. Li, Y. Pan, F. Chen, K.-T. Cheng, and R. Huan, "Real-time lossless ecg compression for low-power wearable medical devices based on adaptive region prediction," *Electronics Letters*, vol. 50, no. 25, pp. 1904–1906, 2014.

[12] S. M. Jalaleddine, C. G. Hutchens, R. D. Strattan, and W. A. Coberly, "ECG data compression techniques-a unified approach," *IEEE transactions on Biomedical Engineering*, vol. 37, no. 4, pp. 329–343, 1990.

[13] R. Besar, C. Eswaran, S. Sahib, and R. Simpson, "On the choice of the wavelets for ECG data compression," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, vol. 6. IEEE, 2000, pp. 3614–3617.

[14] M. S. Manikandan and S. Dandapat, "Wavelet-based electrocardiogram signal compression methods and their performances: A prospective review," *Biomedical Signal Processing and Control*, vol. 14, pp. 73–107, 2014.

[15] J. Chen, S. Itoh, and T. Hashimoto, "ECG data compression by using wavelet transform," *IEICE TRANSACTIONS on Information and Systems*, vol. 76, no. 12, pp. 1454–1461, 1993.

[16] N. V. Thakor, Y.-c. Sun, H. Rix, and P. Caminal, "Multiwave: a wavelet-based ECG data compression algorithm," *IEICE TRANSACTIONS on Information and Systems*, vol. 76, no. 12, pp. 1462–1469, 1993.

[17] R. Kumar, A. Kumar, and G. K. Singh, "Electrocardiogram signal compression based on 2d-transforms: A research overview," *Journal of Medical Imaging and Health Informatics*, vol. 6, no. 2, pp. 285–296, 2016.

[18] B. Zhang, J. Zhao, X. Chen, and J. Wu, "Ecg data compression using a neural network model based on multi-objective optimization," *PloS one*, vol. 12, no. 10, p. e0182500, 2017.

[19] V. Kumar, S. C. Saxena, and V. Giri, "Direct data compression of ECG signal for telemedicine," *International Journal of Systems Science*, vol. 37, no. 1, pp. 45–63, 2006.

[20] D. Tchiotsop, D. Wolf, V. Louis-Dorr, and R. Husson, "ECG data compression using jacobi polynomials," *IEEE engineering in medicine and biology magazine*, vol. 1, p. 1863, 2007.

[21] D. C. Pandhe and H. Patil, "Review and enhancement of ecg data compression and reconstruction method for portable devices," in *2015 International Conference on Energy Systems and Applications*. IEEE, 2015, pp. 490–495.

[22] B. Yu, L. Yang, and C.-C. Chong, "ECG monitoring over bluetooth: data compression and transmission," in *2010 IEEE Wireless Communication and Networking Conference*. IEEE, 2010, pp. 1–5.

[23] A. Tiwari and T. H. Falk, "Lossless electrocardiogram signal compression: A review of existing methods," *Biomedical Signal Processing and Control*, vol. 51, pp. 338–346, 2019.

[24] A. Němcová, R. Smíšek, L. Maršánová, L. Smital, and M. Vítek, "A comparative analysis of methods for evaluation of ECG signal quality after compression," *BioMed research international*, vol. 2018, 2018.

[25] E. Ajdaraga and M. Gusev, "Analysis of sampling frequency and resolution in ECG signals," in *2017 25th Telecommunication Forum (TELFOR)*. IEEE, 2017, pp. 1–4.

[26] M. Gusev and E. Domazet, "Optimizing the impact of resampling on QRS detection," in *ICT Innovations 2018. Engineering and Life Sciences: 10th International Conference, ICT Innovations 2018, Ohrid, Macedonia, September 17–19, 2018, Proceedings 10*. Springer, 2018, pp. 107–119.