Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2019) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 16th International Conference on Ambient Systems, Networks and Technologies (ANT)
April 22-24, 2025, Patras, Greece

Toward Digital-Twin-Assisted Data Acquisition in Underwater Acoustic Sensor Networks

Ivan Spajić*, Lars Michael Kristensen, Volker Stolz

Department of Computer Science, Electrical Engineering, and Mathematical Sciences, Western Norway University of Applied Sciences

Abstract

Data transfer from underwater acoustic sensor networks (UASNs) to cloud services is challenging due to the severe bandwidth constraints of acoustic communication and energy consumption constraints of battery-powered sensor devices. To mitigate this problem, various forms of time series compression algorithms, both lossless and lossy, have been proposed in the literature. These proposals, however, focus on large time series whose in-memory buffering is not feasible in a UASN context. Furthermore, especially in oceanographic data acquisition, buffering large time series sacrifices the Age of Information. To address this gap, we evaluate a modern lossy compression algorithm, Mix-Piece, on small-sized time series. Motivated by our findings, we have developed the Custom-Piece algorithm, an adaptation of Mix-Piece that yields higher data compression at the expense of increased execution times. Our experimental evaluation of the Mix-Piece/Custom-Piece algorithms on real-world oceanographic data shows that compression ratios are highly dependent on the configuration of the algorithms due to the emerging characteristics of the underlying data sets. To facilitate practical use of the Mix-Piece/Custom-Piece algorithms, we outline how a digital twin may dynamically run simulations on available past data to support end-users in developing sensor configurations according to data acquisition requirements.

Keywords: emerging networking; wireless sensor networks and applications; optimization for effective communications; lossy compression;

1. Introduction

There are multiple constraints in the domain of underwater acoustic sensor networks (UASNs) that make the sampling, acquisition, and transfer of oceanographic data particularly challenging. Not only are bandwidths significantly lower than those of traditional radio frequency (RF) terrestrial networks, but energy consumption during data transmission is also considerably higher [1]. Naturally, a way of reducing energy consumption would be to compress the data before transmission. Of the two main types of compression [2, 3], lossless compression is usually reserved for

E-mail address: ispa@hvl.no

contexts where data exactness is of utmost importance (such as transmitting executable programs). Lossy compression, on the other hand, tends to yield higher compressibility, albeit at the cost of accuracy of the decompressed data. Given that there is a strong focus on energy consumption in UASNs and that in the domain of oceanography, due to limited sensor accuracy and measurement uncertainty, sampled data already contains an element of inexactness, there is a natural alignment toward the application of lossy compression.

Compared to domains such as healthcare, data sampling of oceanographic parameters (e.g., sea temperature or salinity), occurs on much larger timescales since these are comparably slower-changing. Sampling more data points before transmitting increases the probability of finding data patterns, which ultimately reduce energy consumption per sampled bit by increasing compressibility. Conversely, due to the larger sampling intervals used by underwater sensors, buffering many data points would result in long waiting times before transmission. For example, if data is sampled every 30 minutes, buffering 1000 data points would result in 500 hours or almost 21 full days of waiting between data transfers, making the data far less current. These conditions altogether indicate that with the application of lossy compression in UASNs, there exist trade-offs between energy consumption, data accuracy, and Age of Information (AoI) [4].

The contribution of this paper is two-fold: *(i)* we present an architecture for a cloud-based, digital twin (DT) that can assist in analyzing and evaluating the aforementioned lossy compression trade-offs, subsequently providing domain experts and end-users with support for fine-tuning their UASN-based, Internet of things (IoT) systems through the use of past-data simulations; and *(ii)* we present a method based on a state-of-the-art, lossy compression algorithm [5] to further increase compression ratios for small-sized time series. Our findings show that, compared to the state-of-the-art, our new algorithm yields higher compressibilities at the cost of longer execution times.

Our work has been undertaken as part of the SFI Smart Ocean project [6] which is aimed at developing a smart and wireless underwater sensor network for fact-based management of marine resources. Our prototype is linked to a pilot demonstrator of a UASN that has been deployed at the Austevoll marine research site located off-shore from south-western Norway. The pilot demonstrator is comprised of multiple anchored buoys whose mooring lines contain various types of configurable, battery-powered, underwater sensors measuring oceanographic parameters such as salinity, oxygen, temperature, and current. Using acoustic modems, the sensors transmit data to a 4G surface gateway which forwards it to cloud-based services. The experimental evaluation of the lossy compression approach proposed in this paper is based on the real-world data sources from the Austevoll marine research site.

The rest of this paper is structured as follows. In section 2, we provide a high-level overview of a cloud-based DT system architecture and how it operates in the context of UASNs. In section 3, we revisit the original Mix-Piece compression algorithm and discuss its shortcomings in online scenarios where AoI is important. Based on the design ideas used in Mix-Piece, we introduce our prototypical implementation of an extension called Custom-Piece. Furthermore, in section 4, we illustrate the performance of the Mix-Piece algorithm for small buffer sizes, and following that, we show how to explore the parameter space (e.g., error threshold) of Custom-Piece. Section 5 concludes and provides directions for future work.

2. Digital Twin-Assisted System and Software Architecture

As per Fitzgerald et al. [7], a digital twin (DT) is a digital representation of a real-world entity called the physical twin (PT). Both twins can communicate with one another, allowing the DT to maintain a known level of fidelity of the PT it represents. Through services, the DT adds value to the PT without compromising its operation.

Our overall approach to DT-assisted data acquisition is illustrated in Figure 1 in the context of the Austevoll marine research site. Here, the battery-powered underwater sensors (having the role of PTs) transmit data to a cloud-based DT which undertakes simulations based on the received data. The main feature of the DT is the simulator component facilitating experts (end-users) in choosing configurations for their domains based on past-data simulations and analyses. To produce these analyses, the simulator component utilizes the same compression algorithm as the one deployed on the remote, battery-powered device. The dashboard of the DT provides an interactive interface for viewing energy consumption, data accuracy, and AoI while allowing for adjustments of the directly related and configurable parameters; compression ratio, error threshold, and maximum number of buffered data points. With the DT's simulation abilities, trade-off strategies between the parameters can be presented to help domain experts in

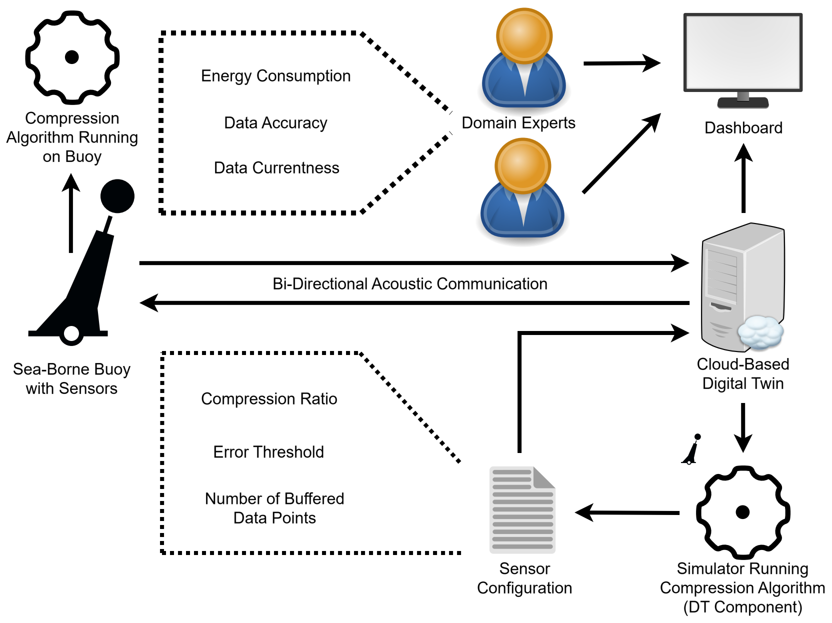

Compression Algorithm Running on Buoy
Energy Consumption
Data Accuracy
Data Currentness
Domain Experts
Dashboard
Bi-Directional Acoustic Communication
Sea-Borne Buoy with Sensors
Cloud-Based Digital Twin
Compression Ratio
Error Threshold
Number of Buffered Data Points
Sensor Configuration
Simulator Running Compression Algorithm (DT Component)

Fig. 1: An architectural overview of the DT-assisted UASN system.

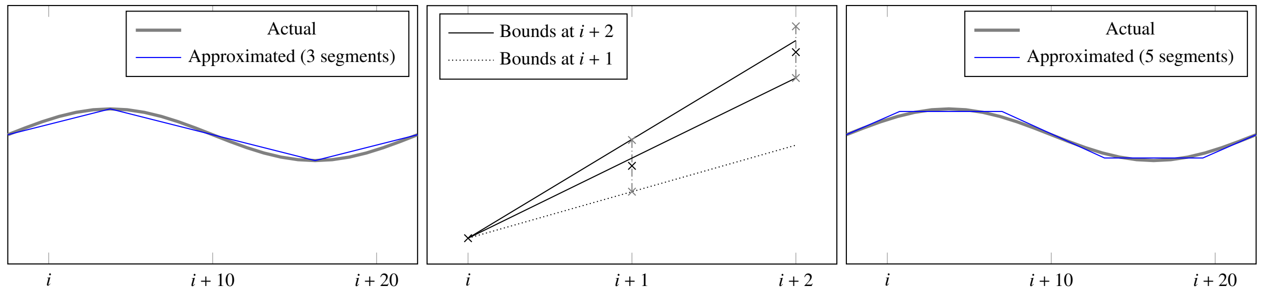
configuring their data acquisition system according to requirements. For example, users may configure data accuracy thresholds or energy budgets as constraints with which the DT could present fitting system configuration options. Based on the results from simulations in the DT, sensor and algorithm configurations are transmitted to the PT in order to optimize and adapt data acquisition in accordance with domain expert requirements.

It should be noted that our approach is not restricted to the setting of the Austevoll marine research site being used in our validation. It generalizes to other resource-constrained, IoT sensing devices where energy consumption is a focus and where the connection to cloud services involves a low-bandwidth network such as a UASN. We do, however, assume that the battery-powered sensing device is capable of some degree of data pre-processing as a main aim is to ultimately reduce transmission costs by means of data compression algorithms deployed on it.

Unlike relatively unpredictable data sets, such as those of stock market information, oceanographic data has more predictable trends. For example, it is generally known that weather patterns over a sea area affect the water in various ways. More windy or stormy weather tends to mix the water, typically cooling it down (at a given depth). In contrast, still waters, together with sunny weather, typically increase its temperature. Moreover, specific tide level fluctuations are known for specific areas, so, although the exact values cannot be predicted, the expected trends (directions of fluctuation) at a given time (e.g., hour of day or season of year) certainly can. Knowledge like this may give additional indication about the expected oceanographic properties and could thus increase our prediction confidence, at least for some data sets. Given sufficient prediction confidence, the cloud-hosted DT could, in this respect, act as a supporting system to battery-powered, sea-borne devices.

3. Algorithms and Implementations

Sensor data is generally sampled in specific and continuous intervals. New values are recorded with a timestamp, transmitted, and added to a growing collection of data points, typically known as a time series. Due to the often large amounts of data involved, various compression techniques have been considered for both transmission and storage. In this section, we first review some existing approaches and then present our own contribution based on a state-of-the-art, lossy compression algorithm intended for large time series.



(a) Piece-wise linear approximation concept. (b) Segment construction process [5]. (c) Different distribution on the Figure 2a curve.

Fig. 2: Piece-wise linear approximation.

3.1. Data Compression and Mix-Piece

Of the known lossy compression techniques for time series data, one of the more popular methods is Piece-wise linear approximation (PLA) [8, 9, 5]. Figure 2a shows how this method approximates data on a curve by constructing linear segments such that each segment covers multiple consecutive points on it. Given that each segment can be represented with a starting point and a gradient value, and given every segment covers at least a few data points on the curve, the amount of information needed to represent the approximated curve becomes reduced. The compressed result is a collection of linear segments that approximate the values of their covered points within a pre-configured deviation range determined by a maximally-allowed error threshold.

Roy and Nikolaidis [10] present a PLA-based algorithm that prioritizes a fixed data rate, but no implementation is directly available. With respect to the trade-offs discussed in section 2, their implementation specifically aims to optimize for accuracy with a given data size, whereas we propose a system where domain experts freely choose which parameters to constrain. Kitsios et al. [5] present two novel PLA algorithms, namely Sim-Piece and Mix-Piece, which outperform similar competing techniques. Our own work is based on Mix-Piece since, according to the authors' experiments and our own test trials, Mix-Piece predictably (generally) outperforms Sim-Piece with respect to data compressibility, especially for increasing error thresholds.

Mix-Piece, much like Sim-Piece, works in two main phases. Figure 2b illustrates how phase 1 begins by traversing over the original time series data points and constructing linear segments for them. Each segment begins construction with an initial point $\langle t_i, v_i \rangle$ by recording its timestamp t_i and quantizing its value v_i. The quantization of the starting-point value of a segment is discussed further in phase 2. Next, the following point, $\langle t_{i+1}, v_{i+1} \rangle$, is read, and the pre-configured error threshold, ε, is added to and subtracted from v_{i+1} to determine the upper- and lower-bound gradients between the points, respectively. The range of gradients between the upper and lower bounds determines all possible gradients of the segment where all of the covered points' values are within $\pm\varepsilon$. Construction continues by reading the next point, $\langle t_{i+2}, v_{i+2} \rangle$, and checking whether v_{i+2} lies within the upper and lower bounds of the segment. If it is above the upper bound or below the lower bound, it is outside of the bounds of the current segment and therefore cannot be included as part of it. In that case, the current segment's construction ends, and a new segment's construction begins. However, if v_{i+2} is within bounds, the values of the upper- and lower-bound gradients must be updated to stay within ε of v_{i+2}. More intuitively, with each new point, the upper and lower bounds are *squeezed* together to keep that point within the maximum allowed deviation of ε. This process continues as long as each new point's value stays within the previously updated bounds or until the end of the time series is reached.

In phase 2, Mix-Piece exploits similarities between the constructed segments to further compress the data. As each segment contains a range of gradients between its upper and lower bounds, overlap between any two segments' gradient ranges would mean that both segments' points could be represented by that overlapping range. More formally, assume that segment s_1 covers its points with some gradient range $s_1.lower \le m \le s_1.upper$ and that segment s_2 covers its points with a gradient range $s_2.lower \le n \le s_2.upper$. Given $s_1.upper > s_2.upper$, $s_1.lower > s_2.lower$, and $s_1.lower < s_2.upper$, there exists an overlapping range such that $s_1.lower \le o \le s_2.upper$, which contains the range of gradients covering both the points in s_1 and s_2. After all possible overlaps have been found, Mix-Piece takes the averages of the resulting upper and lower bounds to further reduce data. Additionally, the quantization formula used for the segments' starting point values ensures they may be equal to other starting point values while staying within $\pm\varepsilon$ of their original one. Matching quantization values and overlapping gradient bounds ultimately produce information which can be used to represent multiple segments simultaneously, thereby further compressing time series data. Mix-Piece optimizes overlaps and exploits other cases of similarities between linear segments, but the reader is referred to the original paper [5] for the remaining details.

3.2. Custom-Piece: Trading Computation for Compression

Thus far, PLA-based data compression has been discussed in the literature in the general context of time series which may be very large, sometimes containing hundreds of thousands, if not millions of data points. Algorithms like Mix-Piece are therefore highly suitable in scenarios of storage space or network bandwidth constraints. Our experiments with Mix-Piece show that the longer the time series is, the higher the compressibility achieved (discussed more in section 4), which is to be expected as a longer time series provides more opportunities for more matching segments. In the context of data sampling with underwater sensor devices, however, it is not feasible to buffer large amounts of data before transmission, both due to memory constraints of the devices themselves as well as typical requirements of a lowest appropriate AoI. Therefore, particularly in the context of oceanographic data sampling, time series data should be processed in a more *online* fashion.

In order to determine more reasonable waiting times, a naturally-arising question is how small an input time series can be before Mix-Piece no longer produces smaller-than-original output sizes. Experiments on our data sets with reasonable ε's revealed that the minimum number of data points is around 13 to 16, with the number being highly dependent on the data set and error threshold. Subsection 4.2 discusses the results in more detail. Experiments also revealed that Mix-Piece does not necessarily always achieve higher compressibilities for higher error thresholds. Somewhat counter-intuitively, some lower ε's against the same time series achieve better compressibilities for some data sets. This observation gives the first insight into our custom implementation of the Mix-Piece algorithm, namely Custom-Piece.

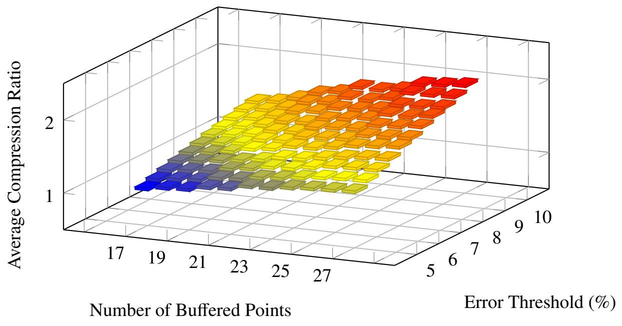
The main difference between lower and higher ε's is in the way that it affects segment construction. Higher ε's will create more range between a segment's upper and lower gradient bounds, thus increasing the chance for a subsequent point to be included. Compared to segment construction with lower ε's, this typically increases the number of points a segment covers, ultimately extending its length (in terms of points) on the time series curve. All subsequent segment constructions now become *deferred* by the additional points that got included in the previous segment(s). Therefore, as the distribution of segments on the time series curve is changed, there is a chance that, for some data sets, a higher ε produces a comparably less compressible linear segment combination.

Since it is intended for large data sets, Mix-Piece's greedy approach does not explore all of the possible segment combinations on a time series curve for obvious reasons. However, given the minimal sizes of time series in our context, finding different combinations (e.g., Figures 2a and 2c) becomes feasible. Our new Custom-Piece algorithm therefore alters the approach of phase 1 of Mix-Piece by constructing a tree of all possible segment combinations for a given time series curve. It then subjects all possible root-to-leaf paths (linear segment combinations) to phase 2 to yield a certain compression ratio with respect to the original data before picking the most optimal one. As a result, it consistently achieves either equal or greater compression ratios when compared to standard Mix-Piece, although at the cost of longer execution times.

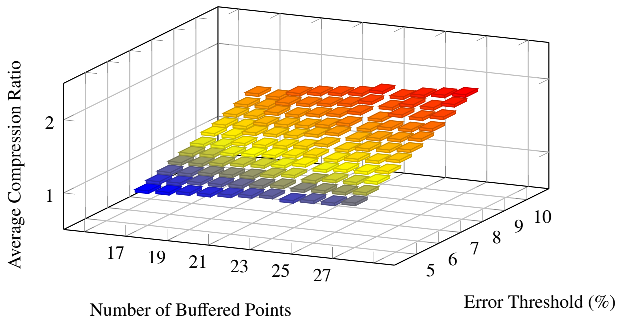
3.3. Algorithm Implementation and Data Sets

Sim-Piece, Mix-Piece, and Custom-Piece were all implemented[1] as part of a C# library to closely match the original authors' pseudo-code and are testable through the accompanying xUnit test project. The test project contains unit tests verifying the correctness of the algorithms, as well as various executable methods to facilitate conducting experiments and exporting their results. The original Austevoll time series data sets are available as CSV files, each of which contains two columns: an incrementing `int64` (`long`) timestamp used to facilitate compression and a `float64`

[1] Our code, data sets, and experimental results are available at `https://doi.org/10.5281/zenodo.14298565`



(a) Turbidity Data Set (n=8662)



(b) Temperature Data Set (n=8662)

Fig. 3: Trade-offs between the numbers of buffered points, error thresholds, and average compression ratios in Mix-Piece.

(`double`) sample value. All testing data set paths, varying time series portion (bucket) sizes, and varying compression error thresholds are included in a separate testing class for easier and more modular configuration.

4. Experiments and Evaluation

We conducted multiple experiments to evaluate the two algorithms against one another and across various data sets and ε values. Similarly to Kitsios et al. [5], our ε's were calculated by finding the difference between the maximum and minimum values of a given time series and calculating percentages of it, with ours ranging from 0.5% to 10%. This approach keeps the absolute value of ε proportional to the value fluctuations in the data set, although it should be noted that it is vulnerable to outliers. For example, if a sea area has water temperatures typically ranging from 4°C to 6.5°C, the difference in maximum and minimum values is 2.5°C. However, if a data set from the area contains a single outlier point of 15°C (e.g., due to sensor malfunction or biological interference), the difference in values for the time series grows to 11°C, which results in a disproportionately large ε. This problem can, however, be avoided in practice by performing data quality assessments on the time series prior to compression.

Moreover, we calculate sizes of the original and compressed time series by assigning each part, such as a quantized value or a gradient, a proportional number of bytes. From this, it is then possible to calculate a simple ratio between the original and the compressed time series, whereby results >1 indicate compression and thus less energy consumption during transmission. The exact details of this are precisely annotated within the code base, and there is no additional, general-purpose compression used.

4.1. *Energy Consumption vs. Data Accuracy vs. Age of Information*

As discussed in section 1, upon applying lossy compression to the domain of UASNs, it becomes relevant to consider the trade-offs between energy consumption, data accuracy, and AoI. These properties can be translated into compression ratios, error thresholds or ε's, and numbers of buffered points or bucket sizes, respectively. Figures 3a and 3b show trade-offs between all three properties after compressing through the entire turbidity and temperature data sets with Mix-Piece. The chosen bucket sizes and ε's are the lowest suitable values where Mix-Piece produces compression ratios >1, although this is investigated in more detail in subsection 4.2.

Generally, three parameter relations are clear: *(i)* for a given bucket size n, increasing ε's yield increasing compression ratios, *(ii)* for a given ε, increasing bucket sizes also yield increasing compression ratios, and *(iii)* for a given compression ratio, increasing bucket sizes require decreasing ε's and vice versa. It is, however, also evident that each data set contains unique properties such that the three relations do not necessarily always hold true or, in the very least, proportionally. For example, for the turbidity data set, if one picks an ε of 8% or more, the increase in compressibility from buffering 24 points compared to 23 is negligible (0.017), however, the increase in compressibility from buffering 25 points compared to 24 is certainly more notable (0.077) in comparison. Another interesting example is that, for any given ε on the temperature data set, compressibility actually decreases if one buffers 24 points compared to 23. Furthermore, it is also clear that the unique data set properties cause discontinuities (differences in the jumps) between increasing ε's, ultimately making some additional sacrifices in data accuracy more worthy than others.

Table 1: The average compression ratios depend highly on the ε's used as well as the underlying time series data. For every combination, Custom-Piece will either match or outperform Mix-Piece at the cost of considerably longer execution times.

		Turbidity (size=8662)				Salinity (size=27219)				Temperature (size=8662)			
		Mix-Piece		Custom-Piece		Mix-Piece		Custom-Piece		Mix-Piece		Custom-Piece	
n	ε%	Avg. Comp.	Avg. ms per Bucket	Avg. Comp.	Avg. ms per Bucket	Avg. Comp.	Avg. ms per Bucket	Avg. Comp.	Avg. ms per Bucket	Avg. Comp.	Avg. ms per Bucket	Avg. Comp.	Avg. ms per Bucket
10	3	0.770	1.826	0.796	9.595	0.795	1.680	0.817	11.411	0.753	2.181	0.764	11.404
10	5	0.877	1.528	0.923	11.805	0.932	1.885	0.971	15.517	0.887	1.864	0.912	13.665
10	7	0.983	1.771	1.056	11.889	1.065	1.638	1.130	16.207	1.023	1.700	1.073	18.114
13	3	0.798	1.432	0.851	108.605	0.834	1.907	0.870	176.164	0.778	1.781	0.799	119.958
13	5	0.944	1.447	1.031	177.368	1.001	2.198	1.072	290.283	0.931	1.502	0.980	197.716
13	7	1.092	1.509	1.219	235.087	1.171	1.576	1.278	352.588	1.096	1.584	1.181	346.227
15	3	0.827	1.463	0.886	1,044.683	0.870	1.594	0.919	1,346.872	0.796	2.208	0.821	884.433
15	5	0.993	1.506	1.088	1,698.269	1.068	1.855	1.162	2,516.834	0.976	2.269	1.042	1,920.423
15	7	1.141	1.529	1.279	2,740.924	1.266	1.650	1.408	3,297.335	1.170	2.076	1.283	3,347.416
16	3	0.873	1.719	0.942	3,702.769	0.891	1.589	0.947	3,435.389	0.800	2.011	0.829	2,501.301
16	5	1.048	1.555	1.167	6,259.301	1.105	1.531	1.212	8,598.029	0.978	2.279	1.054	6,869.982
16	7	1.220	2.102	1.381	9,241.039	1.313	1.661	1.474	11,946.387	1.184	2.022	1.305	11,438.762

In this experiment, AoI is the time to acquire n data points to fill a bucket plus the (average) time to compress it. Hence, if we pick the highest possible sampling frequency, the resulting AoI becomes at least twice the compression time for a given bucket. In oceanography, however, sampling frequencies are in minutes/hours, ultimately being orders of magnitude larger than our average compression times (discussed further in sub-section 4.2).

The kind of simulation and analysis performed in this experiment could ultimately be performed by a DT simulator component in the context of UASNs or other bandwidth-limited domains where lossy compression is tolerable. This component would thus provide value to its stakeholders by running past-data simulations whose results would facilitate users in choosing the right parameter values for their specific use cases. It should, however, be noted that these simulations were done on raw (sampled) data and not decompressed data, as it would be received from the remote device. In either case, domain experts could provide constraints, such as upper limits on bucket sizes and ε's as well as a lower limit on compression ratios, to have the DT yield an optimal configuration if possible. Another example could be to supply ranges of tolerated bucket sizes and ε's as well as a compression ratio difference threshold which would dictate if reducing data currentness or accuracy would be worth it to the domain expert.

4.2. Mix-Piece vs. Custom-Piece Compression Ratios

As outlined in Section 3, we experimented to find out how small a time series could be before Mix-Piece no longer produced smaller-than-original output sizes. According to the results shown in Table 1, this size range is roughly between 13 and 16 points for oceanographic data sets, with the result being highly dependent on the ε used. For example, results for $n = 15$ and $\varepsilon = 7\%$ show that Mix-Piece's average compression ratio for the turbidity data set is 1.141 with an average execution time of 1.529ms. In comparison, for the same parameters, Custom-Piece's average compression ratio for the same data set is 1.279 with an execution time of ~2.7s. It is evident that an ε of 3% consistently produces compression ratios <1, even for larger bucket sizes, whereas an ε of 7% produces compression ratios >1 for almost all parameter combinations. Furthermore, executing Custom-Piece with the same input parameters produces comparably higher compression ratios, albeit at the cost of drastically increasing execution times. This indicates that executing Custom-Piece on remote underwater devices becomes less feasible with a growing number of buffered points.

Two certainties should also be noted: firstly, answering the minimal time series size question relies on the possibility of using ε's in the range of 3% to 7%, which may not be acceptable for all use cases, although we leave this to the domain experts (Roy and Nikolaidis compared 1%, 2%, 5% and 10% [10], and Kitsios et al. used ε's from 0.5% to 5% in increments of 0.5% [5]). Secondly, observing average algorithm execution times may determine the highest acceptable sampling frequencies for some sensor devices. For example, depending on system limitations, a sensor device executing Custom-Piece with a configuration where bucket size $n = 15$ and $\varepsilon = 7\%$, may need up to 3.5s of processing time for the buffered bucket before sampling the next data point. In case of such limitations, this ultimately sets an upper frequency limit on sampling, whereby it could only safely occur at minimal intervals of ~4s.

5. Conclusion and Future Work

In this paper, we present analyses of oceanographic data sets in terms of data accuracy, energy consumption, and AoI, which are all relevant elements in data sampling with resource-constrained sensor devices, especially when applying lossy compression to the restrictive context of UASNs. Our findings highlight unique properties of real-world, oceanographic data sets, subsequently demonstrating that different parameter values, such as those of time series sizes or data error thresholds, are differently suitable for different oceanographic properties. We propose making use of such parameter analyses in DT-based simulations where the results could facilitate stakeholders in making informed decisions on optimizing their systems for their specific use cases. Naturally, such DTs would not only be applicable to UASNs but also to other similarly-constrained domains. Our planned future work in this regard includes a cloud-hosted implementation of such a DT as a proof of concept for our Austevoll marine research site. We would also like to investigate how cloud-based simulations running on decompressed data can be used to fine-tune configurations of remotely deployed compression algorithms running on raw (sampled) as well as sanitized data. Furthermore, to more accurately convey energy savings, we plan to show the necessary energy consumption for communication as well as the consumption from the computational load of running compression algorithms.

Additionally, we investigated the applicability of a state-of-the-art, PLA-based, lossy compression algorithm, Mix-Piece, on small-sized time series and introduced our own similar approach, Custom-Piece, for achieving even higher compressibility. Due to Custom-Piece finding optimal combinations of linear segments on a curve, it consistently achieves equal or higher compression ratios compared to those of Mix-Piece. However, due to the combinatorial method used, Custom-Piece's execution times increase drastically for increasing bucket sizes, making it less feasible for remote IoT devices. Although this problem is somewhat mitigated by the fact that oceanographic data sampling occurs at larger intervals, future work would still involve developing a more heuristic approach, whereby Custom-Piece could dynamically adapt to the data set and find optimal segment combinations without having to construct all possibilities in a trial-and-error fashion. Moreover, instead of waiting for a given bucket size before compressing, Custom-Piece can be made more *online* by running the segment construction procedure during data point buffering. This would altogether significantly reduce post-buffering execution times, subsequently increasing the maximum possible sampling frequency.

Acknowledgements

This work was supported by SFI Smart Ocean NFR Project 309612/F40 and Bergens Rederiforening.

References

[1] A. F. Harris, M. Stojanovic, M. Zorzi, Idle-time energy savings through wake-up modes in underwater acoustic networks, Ad hoc networks (2009). doi:10.1016/j.adhoc.2008.07.014.

[2] G. Chiarot, C. Silvestri, Time Series Compression Survey, ACM Computing Surveys (2021). doi:10.1145/3560814.

[3] J. D. A. Correa, A. S. R. Pinto, C. Montez, Lossy Data Compression for IoT Sensors: A Review, Internet of Things (2022). doi:10.1016/j.iot.2022.100516.

[4] N. Yazdani, D. E. Lucani, Online Compression of Multiple IoT Sources Reduces the Age of Information, IEEE Internet of Things Journal 8 (19) (2021) 14514 – 14530. doi:10.1109/JIOT.2021.3064511.

[5] X. Kitsios, P. Liakos, K. Papakonstantinopoulou, Y. Kotidis, Flexible grouping of linear segments for highly accurate lossy compression of time series data, The VLDB journal (2024). doi:10.1007/s00778-024-00862-z.

[6] SFI Smart Ocean, https://sfismartocean.no/, accessed: 2024-11-24 (2024).

[7] J. Fitzgerald, C. Gomes, P. G. Larsen (Eds.), The Engineering of Digital Twins, Springer, 2024.

[8] S. H. Cameron, Piece-wise linear approximations, Tech. Rep. CSTN-106, IIT Research Institute (1966). doi:10.21236/ad0647190.

[9] G. Luo, K. Yi, S.-W. Cheng, Z. Li, W. Fan, C. He, Y. Mu, Piecewise linear approximation of streaming time series data with max-error guarantees, IEEE International Conference on Data Engineering (2015). doi:10.1109/icde.2015.7113282.

[10] S. K. Roy, I. Nikolaidis, Limited Size Lossy Compression for WSNs, IEEE Conference on Local Computer Networks (2022). doi:10.1109/lcn53696.2022.9843510.