Linearity-based Sensor Data Online Compression Methods for Environmental Applications

Olli Väänänen
School of Technology
JAMK University of Applied Sciences
Jyväskylä, Finland
ORCID: 0000-0002-7211-7668

Timo Hämäläinen
Faculty of Information Technology
University of Jyväskylä
Jyväskylä, Finland
ORCID: 0000-0002-4168-9102

Abstract— **Environmental monitoring is a typical Internet of Things (IoT) application. Environmental monitoring plays a significant role, for example, in smart farming and smart city applications. Environmental magnitudes are usually measured using wireless sensor nodes, which are often battery-powered, and the number of sensing nodes can be large. One effective method for reducing the energy consumption of a sensor node is to use data compression to reduce the amount of data required for transmission via a wireless connection. Compressing the sensor data means fewer transmission periods, and thus, lower energy consumption. Compression methods should be effective for compressing environmental magnitudes and be computationally light to be suitable for constrained sensor nodes. A compression algorithm should be able to compress an online data stream. In this paper, we review some compression algorithms suitable for environmental monitoring and present two new versions of those algorithms. The algorithms were evaluated, tested, and compared. The main parameters used for the comparisons were compression ratio, root mean square error, and inherent latency. The simulation results obtained using real datasets demonstrate that simple linearity-based compression algorithms are effective and suitable for compressing environmental data. Two new compression algorithm versions proved to be effective for compressing sensor data with reasonable compression quality and predictable inherent latency.**

Keywords—compression algorithm, data compression, edge computing, Internet of Things, sensor data

I. INTRODUCTION

In environmental monitoring the wireless sensor nodes can be located in wide area and the number of nodes can be large. Wireless sensor nodes are often battery powered and replacing empty batteries can be costly, as the nodes may be located in a wide area and thus require manpower to complete the replacement. Thus, minimizing the energy consumption and lengthening the lifetime of the sensor node can be a cost-effective solution. Compressing the sensor data stream in online mode can reduce the transmission periods needed via wireless connection. Wireless transmission is known to be the most energy consuming operation in wireless sensor node. Between sensing and transmission phases the node can be in sleep mode.

In this paper, some basic linearity-based compression algorithms are presented, and two new versions are developed and evaluated. The algorithms are compared to each other by compression ratio, root mean square error and algorithm inherent latency. The remainder of this paper is organized as follows: present algorithms are presented in Section II. New algorithm versions are presented in Section III. Compression algorithms inherent latency considerations are in Section IV. Algorithms' ability to compress environmental datasets is in Section V. Section VI is the summary of the results and finally section VII presents the conclusions.

II. LINEARITY-BASED TEMPORAL COMPRESSION METHODS FOR SENSOR DATA

A. Linear Regression based Temporal Compression

Linear Regression based Temporal Compression (LRbTC) algorithm is based on basic linear regression and it is designed to compress the sensor data in online mode. Thus, the dataset is not already available, but as a function of time, the new data values come in sequence with constant frequency, and the LRbTC algorithm compresses the data value by value [1]. The algorithm waits until the first N measurement values are available and then the regression line described by the N values is calculated. These N values are expected to predict the future values of the sensor data. If the data behave linearly, then the regression line can predict the consecutive measurement values with a certain error bound (ε) allowed. N has a minimum of three values, but four and five values were also tested in [1].

Calculating the linear regression of N values yields the linear line that best fits the N values used. The calculation of the regression line is based on the least-squares method, which minimizes the sum of squares of the deviation between data points. After calculating the regression line of the first N measured values, the algorithm stores and/or sends the starting point of the regression line. The inherent latency of the algorithm is $(N\text{-}1)\Delta t$ at this point when the regression line is calculated. Then when a new measured value is achieved after one measurement interval (Δt), the algorithm compares the value to the regression line value at that timestamp. If the value is within one error bound from the line, the algorithm waits for the next measured value and makes a new comparison. When the new measured value falls off from the regression line prediction (differs more than one error bound from the line), the algorithm stores and/or sends the linear regression line value in one timestamp before (last timestamp when the measured value was still within one error bound from the line) as an end point of the linear segment. Then the algorithm waits for N-1 new measured values (because one value is already available; the one that was in more than one error-bound distance from the line and ended the linear segment). After calculating the new regression line, the algorithm stores and/or sends the new regression line starting point.

The weakness of this basic form of LRbTC is the possibility that the values used to calculate the regression line may differ more than one error bound (ε) from the regression line. Thus, the values derived from the compressed dataset may differ more than one error bound from the original values and therefore the error bound requirement is not guaranteed [1]. A modified version was developed to solve the aforementioned weakness in the basic version. The modified LRbTC (M-LRbTC) is analogous to LRbTC, except that the comparison between the raw values and regression line is also made for the values used to calculate the regression line [1]. If the difference between the calculated regression line and the raw value or values is larger than the error bound, then the first two raw values are retained (stored/sent), and a new regression line is calculated when the next two new values are available. The algorithm works if $N = 3$ or more [1],[2].

The output of the algorithm can be presented as a compressed dataset, $M\text{-}LRbTC(S) = <(c_1, \tau_1), (c_2, \tau_2),\ldots, (c_k, \tau_k)>$. The compressed data values (c_i, τ_i) are either the starting points or end points of the linear segments, or the raw data values if the difference has been too big between the line and the raw values that were used to calculate the line.

One drawback of M-LRbTC (and in basic LRbTC) is that two data pairs are required for each linear segment; starting point and end point. Another drawback of this algorithm is the inherent latency. Latency is predicted when the new regression line is calculated, and it is determined by *N*. After the regression line is calculated, the latency is not known and is not predictable. The better the regression line predicts future values, the longer is the latency. The drawback of unpredictable latency can be overcome by sending the regression line parameters *a* (slope) and *b* (base) with the line starting point timestamp. Thus, in this case, the compressed data can be represented as $LRbTC(S) = <(a_1,b_1,\tau_1),(c_2,\tau_2),(a_3,b_3,\tau_3),(c_4,\tau_4),\ldots,(a_{k-1},b_{k-1},\tau_{k-1}),(c_k,\tau_k)>$. The latency in this version is the N–1 measurement intervals when calculating the regression line. When the line parameters are sent, the receiver knows that the values follow the line with one measurement interval latency until the line-end parameters (c_i and τ_i) are received [2].

B. Real-Time Linear Regression Based Temporal Compression

RT-LRbTC was presented in [2] as a modification of M-LRbTC to achieve a predictable and shorter inherent latency. RT-LRbTC uses already available sensor values to calculate a new regression line. Thus, the inherent latency is only one measurement interval, Δt. A flowchart of the RT-LRbTC is presented in Fig. 1. Initially, the algorithm works as an M-LRbTC, and the inherent latency is N–1 measurement intervals long, which is $(N\text{-}1)\Delta t$. In step 6, the algorithm stores and/or sends the regression line parameters and the timestamp of the regression line starting point instead of the line starting point value. In step 7, when the algorithm is in the linear section, the inherent latency is one measurement interval (Δt). If the difference in step 8 is larger than the error bound allowed, the new regression line is calculated in step 9, but the line is calculated using the already available values. The compressed dataset is presented as $RT\text{-}LRbTC(S) = <(a_1,b_1,\tau_1), (a_2,b_2,\tau_2),\ldots, (a_k,b_k,\tau_k)>$, where a_i and b_i are the regression line parameters and τ_i is the line beginning timestamp. When the new line parameters with the timestamp are received, it is known that the previous line ended one measurement interval earlier. Thus, the inherent latency of the algorithm was described by the measurement frequency [2].

The compression efficiency is dependent on the data characteristics, and in most cases, RT-LRbTC has a lower inherent compression ratio (*CR*) than M-LRbTC [2]. As RT-LRbTC generally has a lower *CR*, it means that there are more linear regression lines after compression with RT-LRbTC than with M-LRbTC. An advantage of RT-LRbTC is that the line parameters must be sent only once for each linear section, thus resulting in a better compression ratio compared to M-LRbTC [2]. In the basic version of M-LRbTC, the starting and endpoint values with timestamps need to be sent for each linear segment. RT-LRbTC benefits from the fact that, compared to $(N\text{-}1)\Delta t$ latency with M-LRbTC, there is no inherent latency when the new regression line is calculated.

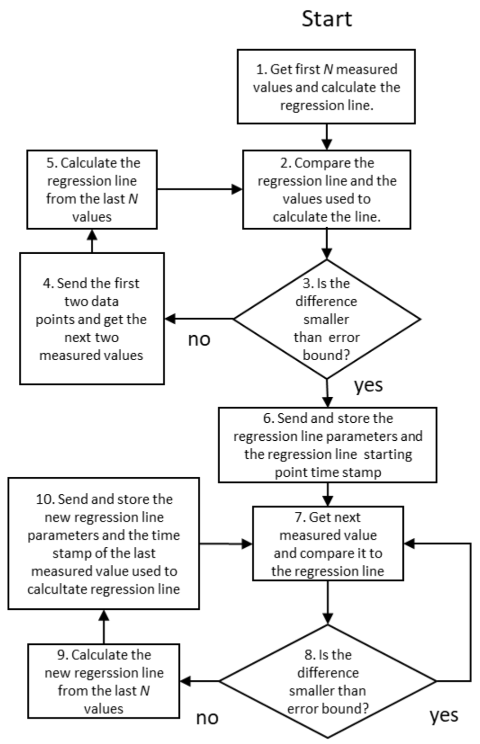
Start
1. Get first N measured values and calculate the regression line.
2. Compare the regression line and the values used to calculate the line.
3. Is the difference smaller than error bound?
no
yes
4. Send the first two data points and get the next two measured values
5. Calculate the regression line from the last N values
6. Send and store the regression line parameters and the regression line starting point time stamp
7. Get next measured value and compare it to the regression line
8. Is the difference smaller than error bound?
no
yes
9. Calculate the new regerssion line from the last N values
10. Send and store the new regression line parameters and the time stamp of the last measured value used to calcultate regression line

Fig. 1. RT-LRbTC flowchart

C. Lightweight Temporal Compression

LTC is a well-known and simple compression algorithm. It was first presented in [3], but a similar algorithm, called Fan, was actually presented before in [4] for electrocardiogram (ECG) data. LTC is a very effective compression algorithm, especially for environmental data that behave rather linearly when the observation time window is short. The compression ratio depends on the data characteristics and error bound used. LTC can achieve a compression ratio as high as 20 when compressing the environmental temperature data with a 10-minute measurement interval and 1.0 °C error bound [1].

The LTC has an unpredictable latency and is dependent on each linear section length. Thus, the higher the *CR*, the longer the latency. If the data behave very linearly, a long latency is derived. When the new linear segment starts, the starting point is known, but the direction of the following values remains unknown until the linear segment ends, and the end point is stored in the compressed dataset. Due to unpredictable latency, LTC is not suited for real-time applications. In this study, LTC has been used as a comparison for the other algorithms.

Some slight variations of the original LTC algorithm have been developed. In [5], a modification of the LTC algorithm was used. Other variations include Adaptive Lightweight Temporal Compression [6], Refined Lightweight Temporal Compression (RLTC) [7], multidimensional extension of the LTC method [8], Direct Lightweight Temporal Compression (DLTC) [9] and DFan [10]. These modified versions were developed either to minimize the data reconstruction error or improve the compression efficiency.

III. New Versions of the RT-LRbTC Algorithm

In this study, two new versions of the RT-LRbTC algorithm were developed. One variation of the basic RT-LRbTC is the RT-LRbTC with 2Δ*t* inherent latency in the new regression line calculation (RT-LRbTC-2Δ*t*). This version is the same as the basic RT-LRbTC (with *N* = 3), but the values used to calculate the new regression line are the last point in the previous linear section, the first value that fell off from the previous section, and one new measurement value. The need to wait for one new value adds the inherent latency to 2Δ*t* when the information of the previous line ends, and new line parameters are obtained. In Fig. 1, this means that in step 9, there is a need to wait for one measurement interval and then calculate the new regression line from the last three values. This new version is a compromise between M-LRbTC and RT-LRbTC; having the inherent latency between those two algorithms.

Another variation of the basic RT-LRbTC and RT-LRbTC-2Δ*t* is the use of weighted linear regression instead of ordinary linear regression. This version of the algorithm is called RT-WLRbTC (Real-Time Weighted Linear Regression-based Temporal Compression) with 2Δ*t* inherent latency (RT-WLRbTC-2Δ*t*). Weighted linear regression (or weighted least-squares, *WLS*) is used in statistics and data analysis instead of simple linear regression when the variation in the samples (values) is not constant. This heterogeneous nature of the values can be addressed by *WLS* using heterogeneous weights w_i in the normal linear regression equations [11]. The sum of squares of the deviation with weights is [12]:

$$S_w = \sum_{i=1}^{n} w_i[y_i - (ax_i + b)]^2 \tag{1}$$

The *WLS* estimates of *a* and *b* (line parameters) were obtained by minimizing (1). In this study, the idea of *WLS* is utilized as some of the samples are used more than once while calculating the linear regression to give more weight to a specific individual value (single values used twice means double weight). RT-WLRbTC-2Δ*t* is similar to RT-LRbTC-2Δ*t*, except that the last value used to calculate the linear regression line is used twice, thus having a weight value of 2 compared to 1 for the other two values. Hence, the computational complexity is similar to that of the M-LRbTC with *N* = 4. The main idea behind using this type of weighted linear regression is that when the linear section ends and the new regression line is calculated, it is expected that the direction of the values is changing. Thus, the latest value is expected to predict future values better than other values used to calculate the regression line, and thus the latest value has a larger effect on the linear regression line calculation. Different versions of weighted linear regression can be developed and used; however, only this one example was tested in this study.

IV. Temporal Compression Methods' Inherent Latency

In Table I, all the presented methods are compared in the order of the algorithm's inherent latency. This comparison does not consider the latency caused by the computational time. Because the measurement interval in typical environmental applications is rather long (minutes or even hours in some cases), the time needed for calculations is negligible, even with the most constrained end devices.

LTC and M-LRbTC (basic version) are not well suited for compressing sensor data value by value in the online mode. LTC has unpredictable inherent latency, which is dependent on how well the values fit in the linear section. When the linear section ends, the endpoint and the new linear section starting point are at the same point. That information is achieved in one measurement interval after the linear section ends. The basic version of the M-LRbTC has an inherent latency of (*N*-1)Δ*t* in the beginning, when the algorithm waits until there are *N* measurement values to be used to calculate the regression line. The starting point line value is stored and/or sent, but it is not known in which direction the values are moving since this until the line ends and the end point value is stored and/or sent. If the linear regression line parameters are sent, then the inherent latency is constant Δ*t* in the linear section. Only M-LRbTC[b] (Table I) and the three RT-LRbTC-based algorithms have fixed and predictable latencies. Of the presented algorithms, RT-LRbTC has the shortest overall latency, Δ*t*, in the linear section, and no latency in calculating a new line. The new versions have double inherent latency (2Δ*t*) and no latency in calculating a new regression line.

V. Linearity-based Methods' Compression Quality and Ability to Compress Environmental Datasets

It was demonstrated in [13] that the average absolute change between consecutive measurements (*AC*) can be used to predict the selected linearity-based algorithm's ability to compress datasets (compression ratio, *CR*). *AC* is defined as:

$$AC = \frac{\sum_{i=1}^{n-1}|x_{i+1} - x_i|}{n-1} \tag{2}$$

Additionally, the standard deviation (*SD*) of the change between consecutive measurements can also be used to predict the *CR*, but the *AC* provides a better estimation [13]. *SD* is defined as (3):

$$SD = \sqrt{\frac{\sum_{i=1}^{n-1}\left((x_{i+1} - x_i) - (\overline{x_{i+1} - x_i})\right)^2}{n-2}} \tag{3}$$

where,

$$(\overline{x_{i+1} - x_i}) = \frac{1}{n-1}\sum_{i=1}^{n-1}(x_{i+1} - x_i) \tag{4}$$

TABLE I. Compression Algorithms' Latencies in Different Phases of Compression

Phase of the Compression	**Compression Algorithm**					
	LTC	***M-LRbTC***[a]	***M-LRbTC***[b]	***RT-LRbTC***	***RT-LRbTC-2Δt***	***RT-WLRbTC-2Δt***
At the beginning	0	(*N*-1)Δ*t*	(*N*-1)Δ*t*	(*N*-1)Δ*t*	(*N*-1)Δ*t*	(*N*-1)Δ*t*
In linear section	Length of the linear section	Length of the linear section	Δ*t*	Δ*t*	2Δ*t*	2Δ*t*
Calculating new line	NA	(*N*-1)Δ*t*	(*N*-1)Δ*t*	0	0	0

[a]M-LRbTC: The linear regression line start and endpoint values are sent.
[b]M-LRbTC: The linear regression line parameters are sent with the starting timestamp. The endpoint of the linear line is sent when the first value falls off from the linear segment.

The suitability of a compression algorithm depends on the characteristics of the sensor data. Many environmental magnitudes are quasi-linear in a short time window, and some compression algorithms are more suitable and effective for this type of linearly behaving data than for other types of data. In this study, the *AC* and *SD* values of the datasets were used to compare the datasets' characteristics and to estimate the compression algorithms' ability to compress those datasets effectively.

The common parameters used to compare different compression algorithms are the compression ratio (*CR*) and the root mean square error (*RMSE*). The compression ratio is calculated as *CR = (original data)/(compressed data)* and the root mean square error [14]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_i - c_i)^2} \tag{5}$$

where v_i is the raw data value and c_i is the corresponding value derived from the compressed dataset. The compression ratio indicates how effectively the algorithm reduces the size of the original data and is most widely used to express the efficiency of the compression algorithm. The *RMSE* indicates the compression quality. Because the methods presented in this paper are lossy, the reconstructed data differ from the original data. Thus, some information is lost. The *RMSE* provides information about how much the data reconstructed from the compressed data differs from the original data. The smaller the *RMSE* value, the smaller is the deviation from the original data.

A. The Selected Compression Algorithms' Efficiency to Compress Real Environmental Datasets

LTC, M-LRbTC, and three versions of RT-LRbTC algorithms were tested with real datasets and compared with each other. All tested algorithms are suitable for constrained IoT devices. In [15] and [16], LTC and RT-LRbTC (with *N* values 3 and 4) were implemented on a LoRa sensor node and no significant energy consumption from the algorithm calculations was discovered. It was discovered that using these lossy compression algorithms led to significant reduction on energy consumption and thus it can be effective solution for lengthening the battery powered sensor node lifetime. The energy saving was due to reduction of the wireless transmitting periods [15], [16].

In this study, the M-LRbTC algorithm was tested using *N* values of 3, 4, and 5. The RT-LRbTC algorithm was tested with the original version and two newly developed versions: RT-LRbTC-2Δ*t* and RT-WLRbTC-2Δ*t*. All algorithms were programmed and tested using MATLAB. The datasets were already available, and thus, this situation does not correspond to the situation for compressing the real-time sensor data stream. This testing situation demonstrates how the algorithm would have compressed the data when the dataset was collected. As the temperature at the same geological location behaves rather similarly year by year, this testing indicates how the algorithm could possibly compress the data in that situation. Similar behavior at certain geological locations for other environmental magnitudes is also expected.

The environmental magnitudes were temperature, air pressure and wind speed. The datasets used were obtained from the Finnish Meteorological Institute's open data service [17]. The datasets tested were Salla Naruska measurement station data and Hanko Tulliniemi measurement station data. All datasets were for the full year 2019 with 10-minute measurement intervals. The temperature was measured in degrees Celsius with 0.1 degrees resolution, air pressure was measured in hPa with 0.1 hPa resolution and wind speed was measured in 10-minute average value with 0.1 m/s resolution.

The Salla Naruska measurement station is in the eastern part of Finnish Lapland. It is one of the coldest locations in Finland. The Hanko Tulliniemi measurement station is in the southernmost part of Finland, 100 m from the sea. These two locations have very different climates. The Salla Naruska dataset was also used in [2] but the Hanko Tulliniemi dataset is a new experiment. In [2] LTC, M-LRbTC, and RT-LRbTC compression ratios with different error bound values were tested using the Salla Naruska dataset (temperature, air pressure, and wind speed). In this study, the same compression ratio simulations for the Salla Naruska datasets were repeated, but the RT-LRbTC algorithm's MATLAB version was further developed to give the *CR* value calculated as only line parameters are needed for each linear section and possible single values (not included in any linear segment) were taken into account. In [2] the same situation was achieved by doubling the *CR* values achieved from the compressed datasets, which included the start and end points of the linear segments. The method in [2] gives the same values for *CR* when the *CR* is high but gives erroneous results if the compressed dataset also includes single values that do not belong to linear segments. In this study, the *RMSE* values for each algorithm were calculated using different error-bound values. In addition, the *CR* and *RMSE* values were compared, and two new algorithms were tested for each environmental dataset.

1) Temperature Datasets

The Salla Naruska dataset contains 52 463 values and the Hanko Tulliniemi dataset 51 961 measurement values with 10-minute measurement intervals. For the full year, the dataset should contain 52 560 values but both datasets have some periods with missing values. The individual missing values and short periods with missing values in the original datasets were linearly interpolated. Longer periods with missing values were removed from the dataset. The temperature values in the Salla Naruska dataset varied between -37.2 °C and +30 °C. The temperature values in the Hanko Tulliniemi dataset varied between -12.3 °C and +27.6 °C.

Table II presents a comparison between these two temperature datasets. Both the *AC* and *SD* values were higher for the Salla Naruska temperature dataset than for the Hanko Tulliniemi dataset. It indicates a lower compression ratio for Salla Naruska dataset.

TABLE II. TEMPERATURE DATASETS

	Salla Naruska	***Hanko Tulliniemi***
Number of values	52 463	51 961
Average change (AC)	0.2077	0.1282
Standard deviation (SD)	0.3473	0.2221

The results achieved in [13] can be used to estimate the compression ratios with error bound of 0.5 °C for the LTC and M-LRbTC algorithms when the *AC* and *SD* values are known. With *AC* values from Table II the estimation gives *CR* = 10.4 (Salla Naruska dataset) and *CR* = 15.8 (Hanko Tulliniemi dataset) for LTC when the error bound is 0.5 °C. The real *CR* values achieved in this paper were 10.2 and 14.1 respectively. For M-LRbTC ($N = 3$) the estimation gives *CR* values 4.0 and 5.8 compared to the results achieved here which are 4.0 and 5.4 respectively. The estimations from [13] are close to the real compression ratios achieved in this study.

Fig. 2 shows the performance results of the different compression algorithms. Fig. 2 (a) and (b) show the compression ratios as a function of error bound (from 0.1 to 1.0 degrees Celsius). The LTC has a superior compression ratio compared to the linear regression-based algorithms. The difference between the different linear-regression-based algorithms is not very large. The compression performance difference between M-LRbTC $N = 3$ and $N = 4$ was slightly larger than that between $N = 4$ and $N = 5$. RT-LRbTC has the lowest compression ratio in terms of linear lines (start and end points), but it benefits from the fact that only one transmission period is required for each linear line. In Fig. 2 (a) and (b), for different RT-LRbTC (RT-LRbTC, RT-LRbTC-2Δt, and RT-WLRbTC-2Δt) algorithms, the compression ratio values are presented as only the line parameters are stored/sent at the beginning of each linear line.

The new variations in RT-LRbTC (RT-LRbTC-2Δt and RT-WLRbTC-2Δt) have a slightly better compression performance than the basic RT-LRbTC. This can be observed in Fig. 2 (a) and (b) for the compression ratio. In general, all the tested compression algorithms performed better for the Hanko Tulliniemi dataset than for the Salla Naruska dataset, as indicated by the *AC* and *SD* values in Table II. A typical error bound for temperature data in many applications can be approximately ± 0.5 degrees Celsius. One possibility to choose the error bound is to use the margin of error of the temperature sensor, which can be found in the sensor's data sheet [3].

In Fig. 2 (c) and (d) the *RMSE* values with different error bounds can be seen. The LTC has the largest *RMSE* values, which means that the drawback of a better compression ratio is lower compression quality. Among linear-regression-based methods, the results are similar between them. As RT-LRbTC has a higher compression ratio, it also has a lower compression quality than the M-LRbTC methods. The advantage of RT-LRbTC is its shorter latency and moderate compression ratio, but its drawback is the larger average reconstruction error after compression than that of M-LRbTC.

The compression quality measurements (*RMSE*) are similar level with all linear regression-based algorithms, as can be seen in Fig. 2 (c) and (d). RT-WLRbTC-2Δt does not show any better performance than RT-LRbTC-2Δt in any measurements for these temperature datasets. In Fig. 2 (e) and (f), the *RMSE* values are compared in terms of the compression ratio. In this comparison, the LTC has the best performance, even though the previous comparisons in (c) and (d) indicate a lower compression quality. The LTC benefits from its superior compression ratio compared to the other tested methods.

2) *Air Pressure Datasets*

The same algorithms were tested for the air pressure datasets. The datasets are listed in Table III. The characteristics of both datasets were very similar, indicating

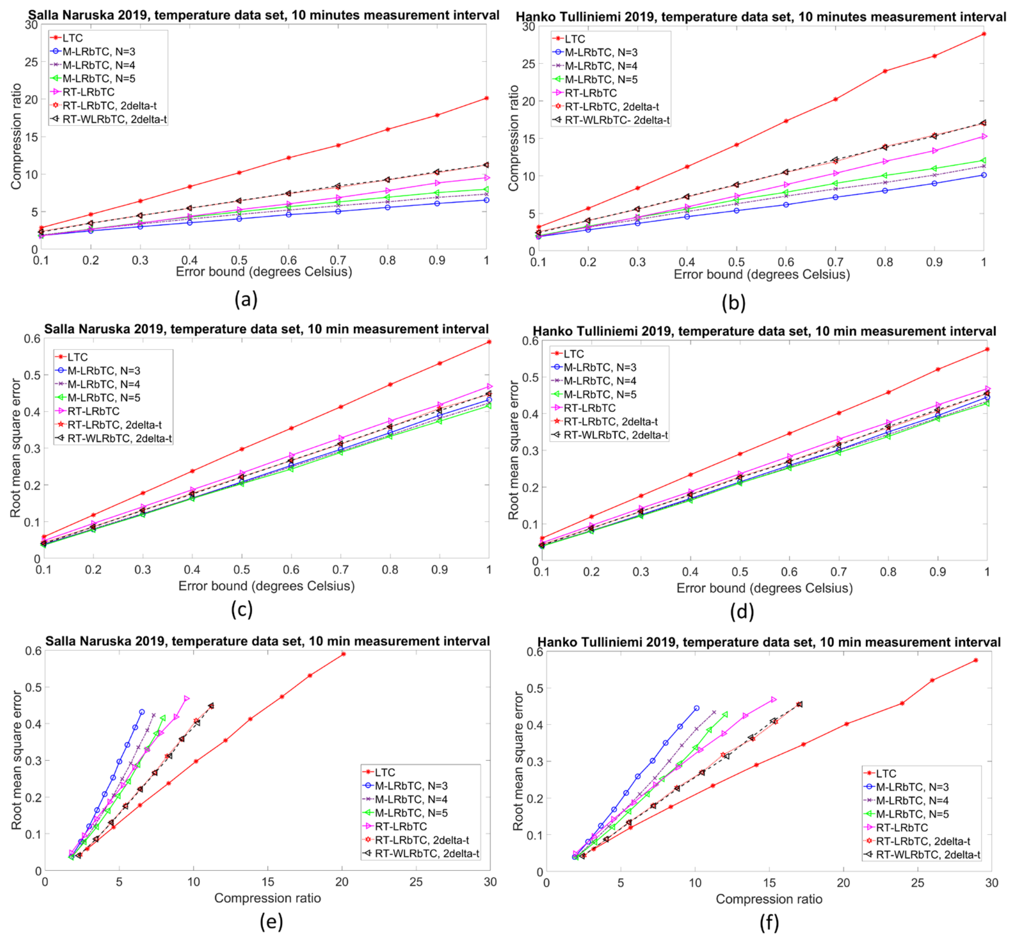

Fig. 2. Compression algorithms' performance for temperature datasets

very similar behavior for both datasets. There was only a small difference between the *AC* and *SD* values. The air pressure values in the Salla Naruska dataset varied between 967.2 hPa and 1039.5 hPa. In Hanko Tulliniemi dataset the values were between 970.5 hPa and 1043.1 hPa.

TABLE III. AIR PRESSURE DATASETS

	Salla Naruska	***Hanko Tulliniemi***
Number of values	52 463	51 961
Average change (AC)	0.0856	0.0836
Standard deviation (SD)	0.1234	0.1249

In Fig. 3, all the results for compressing the two tested air pressure datasets are shown. Both datasets are full-year datasets with 10-minute measurement intervals.

As shown in Fig. 3 (a) and (b), the compression ratios were significantly higher than for the temperature data (Fig. 2). These are not directly comparable, but as both magnitudes are measured with 0.1 (degree and hPa) resolution, the error bounds of 0.1 – 1.0 can thus be compared to each other sufficiently. In general, the air pressure data changes rather slowly; thus, it is well suited for linearity-based compression algorithms. The *AC* and *SD* values in Table III are significantly lower than those for the temperature datasets in Table II, thus indicating better compression performance. From the results obtained in [13] it is possible to estimate the compression ratios for the LTC and M-LRbTC algorithms. The data from [13] estimated the compression ratio with 0.5 hPa error bound for LTC to be 27.9 for the Salla Naruska dataset and 28.3 for the Hanko Tulliniemi dataset. The results from MATLAB simulations provided *CR* values 29.6 for the Salla Naruska dataset and 29.8 for the Hanko Tulliniemi dataset. These results are very close to the estimations.

The results for RT-LRbTC-2Δ*t* and RT-WLRbTC-2Δ*t* were very close to the RT-LRbTC values in terms of the compression ratio (Fig. 3 (a) and (b)). All linear regression-based algorithms are very close to each other in terms of the quality metrics (*RMSE*), as can be seen in Fig. 3 (c) and (d). Again, the LTC has a superior compression ratio but also the largest average construction error. The difference between the various linear regression-based algorithms is small in terms of the quality metrics and compression ratio. The three different RT-LRbTC algorithms presented the best compression performance among the linear regression-based algorithms. In Fig. 3 (e) and (f), the *RMSE* values of the different algorithms are compared in terms of compression ratio. The performance order between the algorithms is quite similar to that of the temperature data, as seen previously.

3) *Wind Speed Datasets*

The two wind speed datasets have different characteristics. As the measurement stations are located in very different places, one close to the sea and the other in Lapland in the lowland, the wind conditions are different. The *AC* and *SD* values are listed in Table IV. The *AC* and *SD* values were higher for the Hanko Tulliniemi dataset, which indicates lower compression ratios for that dataset. The wind speed was measured as 10-minute average values. Otherwise, the wind speed is gusty if measured in instantaneous values, and thus,

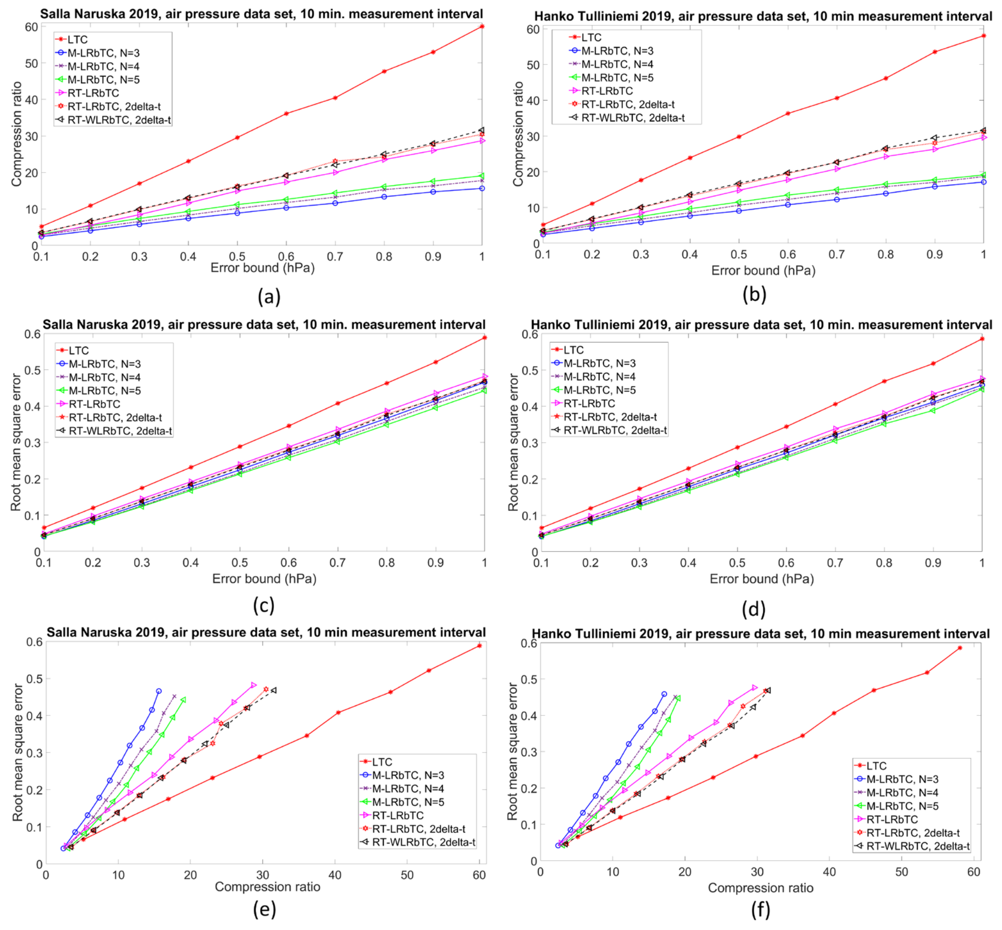

Fig. 3. Compression algorithms' performance for air pressure datasets

it does not exhibit linear behavior. The values in the Salla Naruska dataset were 0 m/s - 9.3 m/s. The values in the Hanko Tulliniemi dataset were 0 m/s - 21.8 m/s.

TABLE IV. WIND SPEED DATASETS

	Salla Naruska	*Hanko Tulliniemi*
Number of values	52 463	51 961
Average change (AC)	0.2844	0.4188
Standard deviation (SD)	0.4125	0.5917

The compression ratios for the Salla Naruska dataset are significantly higher than those for the Hanko Tulliniemi dataset, as can be seen in Fig. 4 (a) and (b). For example, LTC with error bound 0.5 m/s achieves *CR* = 5.54 for Salla Naruska dataset and *CR* = 3.98 for Hanko Tulliniemi dataset. The estimations obtained from the data in [13] provide the compression ratio estimations of 5.5 for the Salla Naruska dataset and 3.6 for Hanko Tulliniemi dataset for LTC with the same error bound. These estimations were close to the actual compression ratios achieved. The estimations for M-LRbTC, *N* = 3 give *CR* = 2.7 for Salla Naruska data and *CR* = 2.1 for Hanko Tulliniemi data. The MATLAB simulation yielded the same values. The compression ratios are clearly higher for the Salla Naruska dataset than for the Hanko Tulliniemi dataset, as indicated by the *AC* and *SD* values of the datasets. RT-LRbTC-2Δ*t* and RT-WLRbTC-2Δ*t* exhibited the higher *CR* values compared with the other linear regression-based methods.

The *RMSE* results are shown in Fig. 4 (c) and (d). The performance results with these quality measurements were at the same level for each linear-regression-based method. The differences were very limited. LTC has a significantly lower compression quality (higher *RMSE*), as can be seen in Fig. 4 (c) and (d). The different characteristics of the datasets did not appear to affect the quality metrics. The *RMSE* results were very similar for both the datasets. When comparing the *RMSE* values as a function of compression ratio (Fig. 4 (e) and (f)), the performances of the different algorithms differ less from each other than with temperature and air pressure datasets.

The new algorithms, RT-LRbTC-2Δ*t* and RT-WLRbTC-2Δ*t*, showed better overall performance than the other linear regression-based algorithms for wind speed datasets. Thus, these new algorithms are potential methods for compressing wind-speed data if the additional inherent latency is acceptable.

VI. SUMMARY OF THE RESULTS

LTC had the best compression efficiency for all datasets, but at the same time, it had the largest *RMSE* values with a certain error bound. The M-LRbTC algorithm benefits from increasing the *N* value from 3 to 4 or 5; however, it makes the algorithm more complex and increases the inherent latency. The new versions (RT-LRbTC-2Δ*t* and RT-WLRbTC-2Δ*t*) have slightly better compression performance than RT-LRbTC. These new algorithms have the same benefit as the RT-LRbTC in that only one transmitting period is needed for each linear segment. The weighted linear regression did not improve the compression performance compared with RT-LRbTC-2Δ*t*. Among the tested algorithms, RT-LRbTC is the best algorithm if a short inherent latency is required, as shown in Table I. If the predicted latency is required, then different versions of RT-LRbTC or M-LRbTC (when regression line parameters are sent) are suitable algorithms. LTC has superior compression performance, but unpredictable latency; thus, it

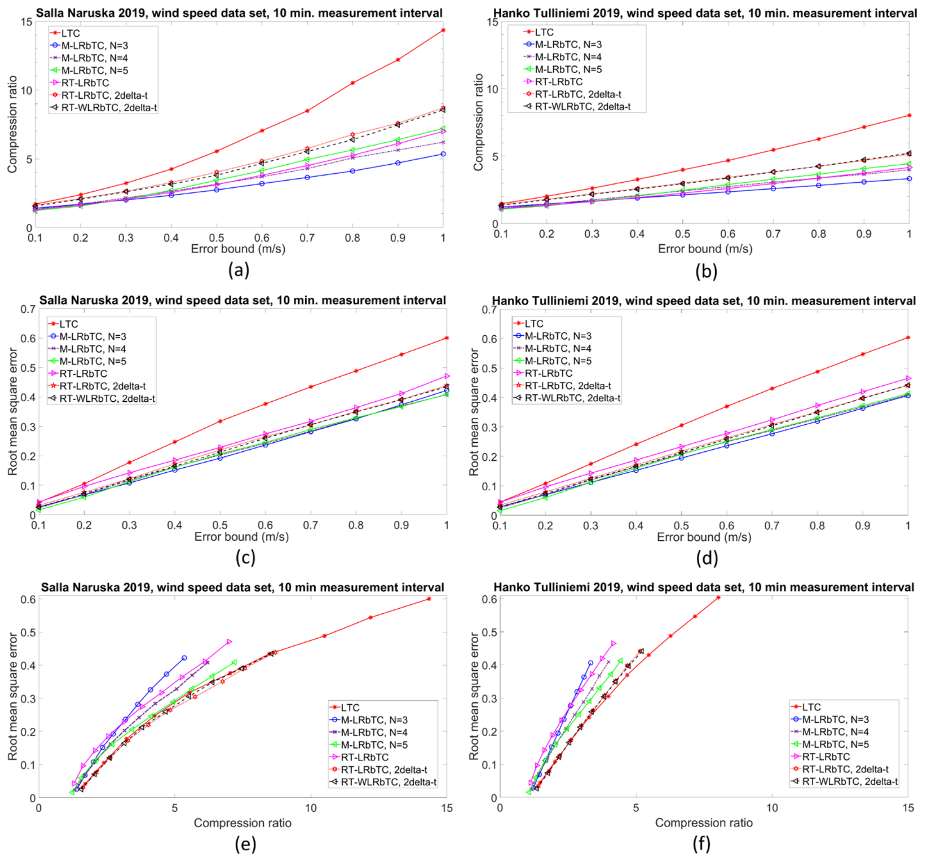

Fig. 4. Compression algorithms' performance for wind speed datasets

is not well suited for online data stream compression or near real-time applications. With a certain error bound, the LTC had the largest *RMSE* values, and thus had the lowest compression quality. Linear regression-based algorithms have very similar *RMSE* values, and thus have a similar performance in terms of compression quality. The two new algorithms proved to be suitable for compressing online environmental sensor data streams with a predictable but slightly longer inherent latency than the original RT-LRbTC.

VII. Conclusions

Different linearity-based sensor data compression algorithms were presented, and their efficiency to compress different environmental microclimate datasets was tested with real datasets. The environmental magnitudes tested were temperature, air pressure and wind speed. Algorithms were compared using the compression ratio (*CR*) and quality measurements as the root mean square error (*RMSE*). Inherent latency was used as a feature to compare the ability of different algorithms to compress data in the online mode.

The datasets used were real datasets acquired from the Finnish Meteorological Institute's Open Data Service. The datasets were used retrospectively and not in the online mode. Thus, the dataset characteristics can be used to compare the compression results of different compression algorithms. The datasets were measurement values from the year 2019, but the characteristics of those datasets can be used to predict the performance of different algorithms in the future in that measurement station and measurement setup. In different places (microclimates), different magnitudes have a behavior typical for that place. Thus, the available dataset for that specific location can be used to predict the characteristics of future datasets and thus predict the performance of the different algorithms.

Two new versions of the RT-LRbTC are presented in this paper. These new versions were tested and compared with other algorithms. LTC has a superior compression ratio compared to other methods but as a disadvantage, LTC has unpredictable inherent latency. The quality measurements demonstrate that the LTC also has the largest reconstruction error when a certain error bound is used. The quality measurements between the different versions of M-LRbTC and RT-LRbTC are very close to each other.

The new versions of the RT-LRbTC algorithm (RT-LRbTC-2Δt and RT-WLRbTC-2Δt) present better compression ratios than the basic version of the RT-LRbTC, but at the cost of a larger inherent latency. In addition, the quality measurements are slightly better for the new versions. Using weighted linear regression to calculate the regression line, weighting the last value used to calculate the line, did not yield any better results than the regular linear regression line. Thus, it only adds the computational complexity of the compression algorithm, without any significant benefits.

All the presented and tested linearity-based compression algorithms are very simple and thus suitable for use in constrained wireless sensor nodes to reduce the overall energy consumption and extend the battery lifetime. These compression methods can significantly reduce the number of wireless transmission periods and, consequently, lower the energy consumption of the sensor node.

References

[1] O. Väänänen and T. Hämäläinen, "Compression Methods for Microclimate Data Based on Linear Approximation of Sensor Data," *NEW2AN 2019, Lecture Notes in Computer Science*, vol 11660. Springer, Cham. https://doi.org/10.1007/978-3-030-30859-9_3

[2] O. Väänänen and T. Hämäläinen, "Sensor Data Stream on-line Compression with Linearity-based Methods," *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2020, pp. 220-225, doi: 10.1109/SMARTCOMP50058.2020.00049.

[3] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow and D. Estrin, "Lightweight temporal compression of microclimate datasets [wireless sensor networks]," *29th Annual IEEE International Conference on Local Computer Networks*, 2004, pp. 516-524, doi: 10.1109/LCN.2004.72.

[4] S. M. S. Jalaleddine, C. G. Hutchens, R. D. Strattan and W. A. Coberly, "ECG data compression techniques-a unified approach," in *IEEE Transactions on Biomedical Engineering*, vol. 37, no. 4, pp. 329-343, April 1990, doi: 10.1109/10.52340.

[5] D. Parker, M. Stojanovic, and C. Yu, "Exploiting temporal and spatial correlation in wireless sensor networks," *2013 Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 2013, pp. 442-446, doi: 10.1109/ACSSC.2013.6810315.

[6] J. Azar, A. Makhoul, R. Darazi, J. Demerjian, and R. Couturier, "On the performance of resource-aware compression techniques for vital signs data in wireless body sensor networks," *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*, Jounieh, 2018, pp. 1-6, doi: 10.1109/MENACOMM.2018.8371032.

[7] O. Sarbishei, "Refined Lightweight Temporal Compression for Energy-Efficient Sensor Data Streaming," *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, Limerick, Ireland, 2019, pp. 550-553, doi: 10.1109/WF-IoT.2019.8767351.

[8] B. Li, O. Sarbishei, H. Nourani, and T. Glatard, "A multi-dimensional extension of the Lightweight Temporal Compression method," *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 2918-2923, doi: 10.1109/BigData.2018.8621946.

[9] L. Klus *et al.*, "Direct Lightweight Temporal Compression for Wearable Sensor Data," In *IEEE Sensors Letters*, vol. 5, no. 2, pp. 1-4, Feb. 2021, doi: 10.1109/LSENS.2021.3051809.

[10] S. Lu, Q. Xia, X. Tang, X. Zhang, Y. Lu and J. She, "A Reliable Data Compression Scheme in Sensor-Cloud Systems Based on Edge Computing," in *IEEE Access*, vol. 9, pp. 49007-49015, 2021, doi: 10.1109/ACCESS.2021.3068753.

[11] W. W. Piegorsch, Statistical Data Analytics: Foundations for Data Mining, Informatics, and Knowledge Discovery. Chichester, West Sussex: Wiley, 2015.

[12] S. Chatterjee and A. S. Hadi, Regression Analysis by Example, John Wiley & Sons, 2012.

[13] O. Väänänen, M. Zolotukhin, and T. Hämäläinen, "Linear Approximation Based Compression Algorithms Efficiency to Compress Environmental Data Sets," In *Web, Artificial Intelligence and Network Applications. WAINA 2020. Advances in Intelligent Systems and Computing,* vol 1150. Springer, Cham. doi: 10.1007/978-3-030-44038-1_11

[14] N. Q. V. Hung, H. Jeung, and K. Aberer, "An Evaluation of Model-Based Approaches to Sensor Data Compression," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2434-2447, Nov. 2013, doi: 10.1109/TKDE.2012.237.

[15] O. Väänänen and T. Hämäläinen, "LoRa-Based Sensor Node Energy Consumption with Data Compression," *2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT)*, 2021, pp. 6-11, doi: 10.1109/MetroInd4.0IoT51437.2021.9488434.

[16] O. Väänänen and T. Hämäläinen, "Efficiency of temporal sensor data compression methods to reduce LoRa-based sensor node energy consumption", In *Sensor Review*, Vol. 42 No. 5, pp. 503-516, 2022, https://doi-org.ezproxy.jyu.fi/10.1108/SR-10-2021-0360

[17] Finnish Meteorological Institute's open data–service. [Online] Available: https://en.ilmatieteenlaitos.fi/open-data