Compression Methods for Microclimate Data Based on Linear Approximation of Sensor Data

Olli Väänänen[1][0000-0002-7211-7668] and Timo Hämäläinen[2][0000-0002-4168-9102]

[1] Industrial Engineering, School of Technology, JAMK University of Applied Sciences, Jyväskylä, Finland
olli.vaananen@jamk.fi
[2] Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland
timo.t.hamalainen@jyu.fi

Abstract. Edge computing is currently one of the main research topics in the field of Internet of Things. Edge computing requires lightweight and computationally simple algorithms for sensor data analytics. Sensing edge devices are often battery powered and have a wireless connection. In designing edge devices the energy efficiency needs to be taken into account. Pre-processing the data locally in the edge device reduces the amount of data and thus decreases the energy consumption of wireless data transmission. Sensor data compression algorithms presented in this paper are mainly based on data linearity. Microclimate data is near linear in short time window and thus simple linear approximation based compression algorithms can achieve rather good compression ratios with low computational complexity. Using these kind of simple compression algorithms can significantly improve the battery and thus the edge device lifetime. In this paper linear approximation based compression algorithms are tested to compress microclimate data.

Keywords: Edge Computing, Internet of Things, Compression algorithm.

1 Introduction

Edge computing has been one of the most significant research topics in the field of Internet of Things during these years. The edge computing means that part of the data analysis is carried out in so-called edge devices. The edge devices are devices located on the edge of the network. Wireless sensor nodes are one example of typical edge devices. The edge devices are often computationally constrained and light devices [1].

Edge computing is not going to substitute the cloud computing but it is more like a supplement concept in the IoT field. Most of the data analysis has been carried out in the cloud and this will be probably the case in the future as well. As the amount of the data from the sensing devices is increasing all the time and these sensing devices are often battery or energy harvesting powered, the energy efficiency of those so called edge devices has become very important. It is known that transmitting data wirelessly from the edge device is the most energy-consuming task in these devices. It is more energy efficient to conduct some light data analysis or pre-processing locally and thus reduce the amount of data needed to send to the cloud. One possible pre-processing task for the sensor data is to filter clearly erroneous data and to compress the sensor data. The edge computing approach can also help to solve privacy and security issues concerning IoT data and offer minimized latency and improve the quality of service (QoS) [2].

There are different sensor data compression methods available. The suitability and efficiency of different methods depend on the data characteristics. Different methods differ in computational complexity, which is an important aspect in edge computing. This paper presents basic and light compression algorithms based on data linearity. Many environmental values are near linear in small time window. These compression algorithms' compression efficiency is tested for microclimate datasets. Datasets are temperature, air pressure and wind speed measurements from the Finnish Meteorological Institute's open data service.

The microclimate data is often nearly linear in short time window. For example, temperature normally changes slowly and if the measurement sampling rate is fast enough, the consecutive measurements cannot deviate much from each other [3]. Air pressure normally also changes slowly. Only approaching low pressure such as a thunderstorm, the air pressure can drop quickly [4]. Wind speed is slightly different because it can stay zero rather long periods. The wind speed also varies quite quickly and it is also quite abrupt in nature [3]. In this paper, the wind speed dataset is averaged data and thus represents more linear type data.

The microclimate data is very important for example in different agricultural applications. Agricultural applications for example for crop protection and to maximize crop production [5, 6] have been presented; however, microclimate measurements are important also in urban environment [7].

2 Lightweight Compression Methods for Sensor Data

In constrained edge devices, it is crucial to optimize resource usage. This means to optimize computational capacity, energy consumption and bandwidth usage [8]. These devices are often connected to the internet via wireless connection. Wireless transmitting is known to be the most energy-consuming task in these devices, thus it is in many cases more energy efficient to carry out data pre-processing and lightweight data analytics locally and thus reduce the amount of data needed to send via wireless link. A very simple method for reducing the amount of data is to compress the data. The other method is simply to reduce the sampling frequency of the sensor [3]. The drawback here is that information is lost between sampling points. Sampling a sensor is quite low energy operation compared to the energy consumption in radio transmission [3]. By using an effective and low computational complexity compression algorithm it is possible to keep radio transmitting rate low and thus keep the energy consumption on a low level, yet at the same time keep the accuracy of the higher sampling rate.

Typically, a simple edge device is a sensor node measuring some environmental magnitudes. Typical environmental magnitudes are for example temperature, humidity, air pressure and lightness. The measured values are then sent to the cloud and in the cloud, the data is combined with other data (for example open data) and together used for decision processing.

2.1 Lossy Methods and Lossless Methods

Sensor data compression methods are divided in lossy and lossless methods. Many different algorithms are presented for sensor data compression [9, **Error! Reference source not found.**. The suitability of the compression algorithm is dependent on the sensor data characteristics. For example, many environmental magnitudes are nearly linear in short time scale, and thus some compression algorithms are more suitable for this kind of data. Some other type of data may require different types of compression algorithms.

If the reconstruction error accepted is more than zero, it is possible to use lossy compression algorithm. Compression ratio is dependent on accepted reconstruction error. Thus, the lossy compression algorithm will lead to loss of the information [11]. The advantages of lossy compression algorithms are the effective reduction of the data and in many cases, the computational simplicity. The compression and reduction of the data is done by eliminating some of the original information [11]. The accepted level of reconstruction error is very application dependent. In general, the lossy compression algorithms have higher a compression ratio together with lower computational complexity than lossless algorithms [12].

Many lossy algorithms have some latency and thus are not suitable for real-time applications. There are also lossy zero-latency compression algorithms. These compression methods are based on predictive filters (e.g. Kalman filter), which predict the data values from previous samples. In this method, the same filter is used in both sides of the network (sensor node and the user node where the data is analyzed further), thus the same estimation is used in both sides, and the new data is sent only if the value differs from the predicted value more than the tolerance level [8].

Lossless algorithms are able to reconstruct the original data without an error. The lossless methods perform two steps: the statistical model is first generated and then the second step uses this statistical model to map the input data to the bit sequences. In these bit sequences, the frequently occurred data generates a shorter output than infrequently occurred data. The two main encoding algorithms used are Huffman coding and arithmetic coding. The Huffman coding is computationally simpler and faster; however, it gives poor results in compression. Arithmetic coding is more efficient in compression but more complex. In many cases, the lossless algorithms are not suitable because the compression ratio is poor and computational complexity is higher than in lossy algorithms [13].

2.2 Lossy Compression Algorithms Based on Linear Approximation

Lossy data compression algorithms analyzed in this paper are based mainly on piecewise linear approximation. Piecewise linear approximation based compression algorithms are based on the fact that many environmental phenomena are near linear in short time window [3]. These kinds of phenomena are for example temperature, humidity, air pressure and wind speed.

A simple linear compression model is based on a regression line, which is calculated on the minimum of the first three measured values [13]. Least-squares regression line is used to approximation of discrete data [14]. In the least-squares regression line, the linear model is set to fit a set of data points. The least-squares method minimizes the sum of squares of the deviation between the data points and the fitting line thus gives a best fit to the data points. This is called a linear regression. A linear function $y = ax + b$ has two free parameters, a and b [14]. The general sum of squares of the deviation is [14]:

$$S = \sum_{k=1}^{N} [y_k - (ax_k + b)]^2 \tag{1}$$

Minimizing this equation and solving for a and b give [14]:

$$a = \frac{\sum_{k=1}^{N} x_k \sum_{k=1}^{N} y_k - N \sum_{k=1}^{N} x_k y_k}{\left(\sum_{k=1}^{N} x_k\right)^2 - N \sum_{k=1}^{N} x_k^2} \tag{2}$$

$$b = \frac{\sum_{k=1}^{N} x_k \sum_{k=1}^{N} x_k y_k - \sum_{k=1}^{N} x_k^2 \sum_{k=1}^{N} y_k}{\left(\sum_{k=1}^{N} x_k\right)^2 - N \sum_{k=1}^{N} x_k^2} \tag{3}$$

The parameters a and b give the best line fit to the N data points. To use these formulas it is needed to sum x_k, x_k^2, y_k, $x_k y_k$, and square the sum of x_k [14]. If the regression line is calculated from the first three measurements, then N is 3.

If the data is nearly linear, this regression line gives the prediction for the following measured data points with a certain error bound e. When the measured data point falls out of the error bound $\pm e$, then the new regression line is calculated. Hence, the data will be presented in piecewise linear segments.

There are several different versions of this kind of algorithm presented in literature. The algorithm is named here as Linear Regression based Temporal Compression (LRbTC). The algorithm is as follows:

1. Get the next three measured values and calculate the regression line to fit those three values.
2. Store (send) regression line point at time moment of the first measured value used to calculate regression line.
3. Get next measured value and compare it to the regression line.
4. If the difference is under the error bound e, then go to 3. Else, continue onto the next step.
5. Store (send) the regression line point when the measured value was last time under the error bound and go to 1.

Fig. 1 shows an example of this linear regression based compression for sensor data. Original temperature data is marked in blue circle. The regression line is calculated from the first three measured values (20, 20.3 and 20.1). Then following measured values are compared to the regression line value on that time moment. Regression line continues until the difference between measured value and regression line exceed the error bound *e*, which is in this example set to 0.5. Regression line is the green line in Fig. 1. At time moment 11 the difference between regression line and measured value exceeds 0.5, thus the first regression line is set to end at time moment 10. From time 1 to 10, the compressed data includes only the starting point of the regression line and the end point of that line. The next regression line is calculated from the measured values in time moments 11 to 13. At time moment 15, the difference exceeds the error bound and thus the new line is calculated from the values at time moments 15-17. In 20, the difference exceeds again the error bound. The first 19 measured values (time moments 1 to 19) are compressed to 6 values (three regression lines).

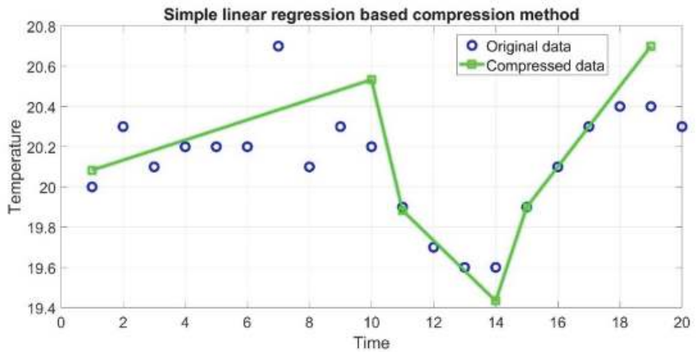

Fig. 1. Linear Regression based Temporal Compression (LRbTC) algorithm example.

In the example in Fig. 1, the error bound was set to 0.5. Thus, the measured value and regression line value should not exceed 0.5. This can anyway happen in time moments that are used to calculate the regression line.

The modified version of the LRbTC (M-LRbTC) algorithm corrects the problem if the difference between regression line and the data values used to calculate this regression line exceed the error bound *e*. The modified version of the algorithm is as follows:

1. Get the next three measured values and calculate the regression line to fit those three values.
2. Compare the regression line and three values used to calculate the line.
3. If the difference is greater than error bound *e,* then store (send) the first two data points and get the next two measurement values and calculate new regression line and go to 2, else continue onto the next step.
4. Get the next measured value and compare it to the regression line.
5. If the difference is under the error bound *e*, then go to 4. Else, continue onto the next step.
6. Store (send) the last regression line point when the measured value was under the error bound and go to 1.

Lightweight temporal compression (LTC) was introduced in [3]. It is simple and very efficient compression algorithm for microclimate type data in a small enough time window. LTC's effectivity to compress data depends on the data characteristics. For linear type environmental data, it can obtain up to 20-to-1 compression ratio [3]. Compression ratio is also dependent on error bound used. It is recommended to use the sensor manufacturer's specified accuracy value as the error bound in LTC algorithm [3]. For example if the sensor used is a temperature sensor with 0.5 degrees accuracy, it is reasonable to use 0.5 as the error bound.

The LTC algorithm is explained in detail in [3, 11, 15, 16] and a modified version in [17]. The linear model starts with the first data value as a starting point. The lower line and upper line (limit lines) are drawn from the starting point to the next measured value ±*e* as seen in Fig. 2 a. The limit lines are tightened from the following values when error bound extreme or extremes are inside the previous limit lines as in Fig. 2 b. and c. The measured data is discarded from the linear model if the measurement cannot fit inside upper line and lower line determined by the previous data with the error bound ±*e*. Then the new linear model starts using as a starting point the middle point of the upper line and lower line in last time moment included in the linear segment. This procedure of the algorithm can be seen in Fig. 2 d.

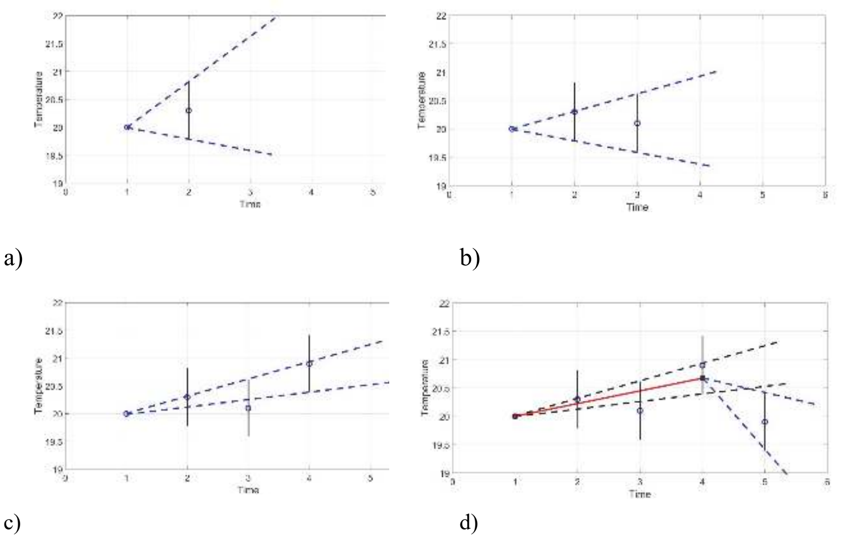

Fig. 2. Lightweight temporal compression (LTC) algorithm.

The reconstruction error never exceeds the error bound *e* in the LTC algorithm. The LTC algorithm has low computational complexity and thus it is suitable for constrained edge devices such as sensor nodes [17]. In Fig. 3, the LTC is compared to previously presented linear regression based algorithm.

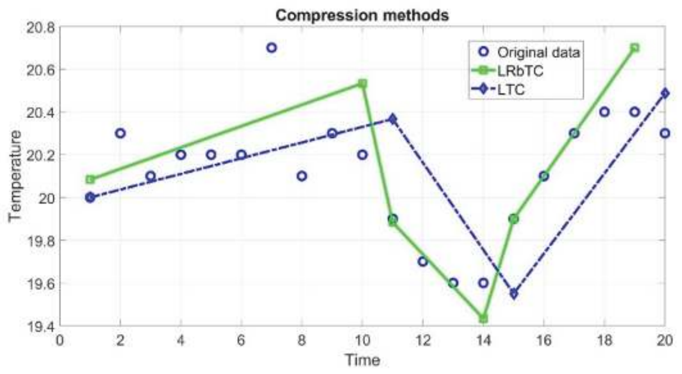

Fig. 3. LTC compared to the basic linear regression based algorithms.

The disadvantage of the LTC is that it is not well suited for real-time applications [8] and its suitability in general is very application dependent. LTC uses linear interpolation to represent the original signal, and the linear interpolation model is known only when the both extremes of the linear part is known. This introduces significant latency for the model [8]. The linear regression based algorithms presented previously suffer from the same problem.

2.3 Transform Based Compression Methods

Discrete Fourier Transform (DFT) is a well-known transform based algorithm. It is simple to use for compression by using the Fast Fourier Transform (FFT) algorithm [18]. The FFT algorithm expresses the time-series signal in frequency representation. By removing the coefficients with less energy, it is possible to reduce the amount of data and still keep the information to rebuild the time series data with reasonable reconstruction error. When the FFT is taken over a window of *N* samples and the first sample and last sample differ a lot, the information of discontinuity is spread across the frequency spectrum. To prevent this discontinuity it is possible to overlap the windows [18].

Another well-known transform based algorithm is Discrete Cosine Transform (DCT) and Modified Discrete Cosine Transform (MDCT) [12, 18, 19]. It has several advantages compared to FFT algorithm [18]. The DCT coefficients are real numbers; thus there is no need to deal with complex numbers. This saves memory and is less complex. The DCT also has the information concentrated to the few low-frequency components and the DCT does not suffer the edge discontinuity problem like FFT. The DCT is a well known and widely used compression algorithm for example in image compression and for time series type sensor data.

3 Testing the Algorithms with Real Microclimate Data

The linear approximation based compression algorithms are tested for microclimate type data and compared to the DCT algorithm. The datasets tested here are gathered from the Finnish Meteorological Institute's open data service [20]. Finnish Meteorological Institute has about 400 observation stations in Finland. Not all the stations have the same measured variables. For this research, Salla Naruska station's data from year 2018 in 10 minutes time sampling rate was chosen. The variables chosen were temperature, air pressure and wind speed. Salla Naruska measurement station is located in eastern Lapland and known as one of the coldest places in Europe. The exact situation of the station is: latitude 67.16226, longitude 29.17766 in decimal degrees. The temperature is in Celsius degrees (ºC), air pressure in hectopascals (hPa) and wind speed in meters per second (m/s). Th wind speed is measured in 10 minutes average. All variables are measured with one decimal resolution.

One year measurements in 10 minutes time interval mean 51,961 measurements for each variable. Some data was missing; however, the missing points were linearly interpolated. In air pressure data in total 102 points were missing, in temperature data 101 points were missing and in wind speed data 1,077 points were missing. For comparison, also the same data in one-hour measurement interval was used. This one-hour interval data for the whole year 2018 includes 8,761 measurements points for each magnitude. The missing values were also linearly interpolated.

The compression algorithms chosen were simple linear regression based approximation algorithm (LRbTC), modified linear regression based algorithms (M-LRbTC) and lightweight temporal compression (LTC). Basic discrete cosine transform (DCT) was used for comparison.

The algorithms were tested with MATLAB simulation. LRbTC, M-LRbTC and LTC algorithms were programmed in MATLAB by using mainly functions *polyfit* and *polyval*. *Polyfit* function was used for linear regression in LRbTC, and M-LRbTC and to create upper and lower lines in LTC instead of equations 2 and 3 [21].

Discrete cosine transform (DCT) was tested by using the MATLAB built-in function *dct*. In this example, the DCT was used with window of five measured values to calculate DCT. It was then tested with different threshold values to cancel the smallest coefficient values. After rebuilding the signal, the maximum difference (variation) from the original values was calculated.

Algorithms were compared to each other by compression ratio versus reconstruction error. The compression ratio (*CR*) was calculated by:

$$CR = \frac{original\ data}{compressed\ data} \tag{5}$$

Thus, the bigger the *CR* value is, the more efficient the compression algorithm is. The *CR* varies significantly according the error bound *e* used.

Temperature data was tested first. In the total 51,961 measured values the highest temperature was +30.4 ºC and the lowest temperature -33.8 ºC. With 10-minute measurement interval, the temperature data is mostly near linear; however, in some extremes the temperature changes quite a lot between consecutive measurements.

LRbTC algorithm showed very big reconstruction errors for tested temperature data, which is because when measuring the regression line, the values used to calculate may differ from the regression line more than the error bound *e* used. The data used is with 10-minute interval and in extremes, the values may differ significantly from measurement moment to the next moment. Higher measurement sampling rate would help the situation.

The modified version of the basic linear regression based algorithm (M-LRbTC) works as it is intended. It was tested with different quantity of measurement values to calculate the regression line. In Fig. 4, the red line is used with three values to calculate the regression line; cyan line is with four values and magenta with five values.

Fig. 4 illustrates the results for M-LRbTC, LTC and DCT algorithms for temperature data. With typical error bound $e = 0.5$ ºC, the M-LRbTC algorithm can achieve 3.9-4.8 compression ratio. LTC is significantly more effective with $CR = 9.5$. DCT was tested with five values time window to calculate DCT. DCT compression ratio is significantly lower compared to the other tested algorithms. DCT suffers from the small window used and it can achieve higher compression ratios with bigger window used. Small time window was chosen to be more realistic for sensor data stream.

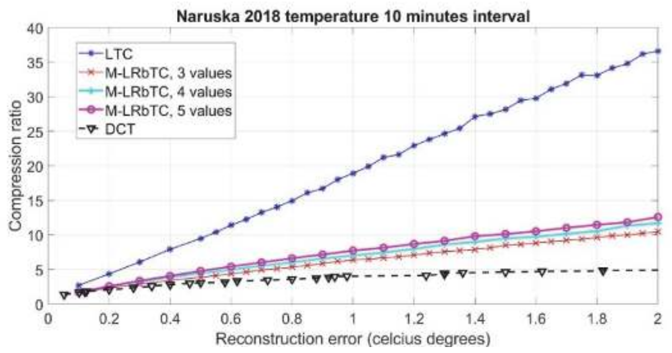

Fig. 4. Linear approximation based compression algorithms and DCT for temperature data.

Fig. 5 shows the results for the air pressure data. Air pressure values varied between 976.4 hPa – 1056.4 hPa. The variation is nearly linear in short time window, however, the data includes few clear errors. Three times the air pressure value changes from measurement to the next more than it is normally possible. The biggest difference in consecutive measurements is 12.7 hPa (in 10 minutes), which is clearly an error. Normally the air pressure can change up to 5 hPa/hour and only in some very quickly progressing low pressure it can be more than 5 hPa/hour [4]. The quick changes in air pressure data can be due to clear measurement error or for example due to sensor calibration. The results for the air pressure data are similar to the temperature data, except the compression ratios are much higher. This indicates that the air pressure data is behaving very linearly with the 10-minute measurement interval. The LTC algorithm can achieve high compression ratios.

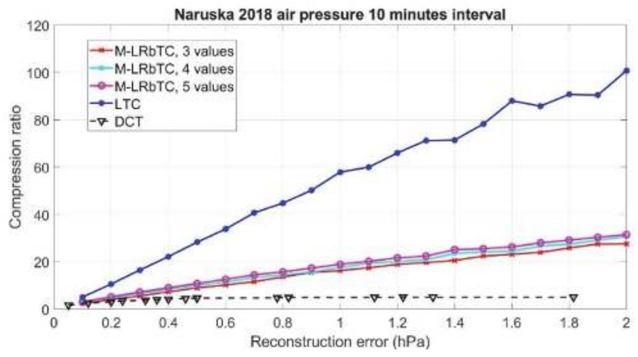

Fig. 5. Linear approximation based compression algorithms and DCT for air pressure data.

In Fig. 6 are the results of the wind speed measurements. Wind speed is measured in 10-minute average values. Wind speed is a different characteristic compared to the temperature and air pressure data. Wind speed can remain quite a long period in 0 m/s. Wind speed can also change quickly and quite significantly; however, here the 10-minute average measurement averages the results significantly. The compression ratios for wind speed data are on the same level as for the temperature data.

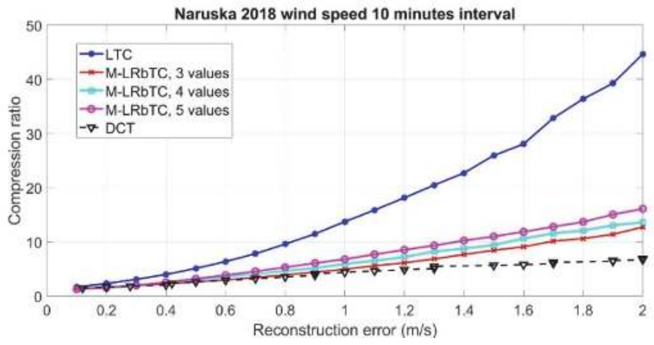

Fig. 6. Linear approximation based compression algorithms and DCT for wind speed data.

In every comparison, it can be seen that LTC is the most effective compression method. M-LRbTC also works well and it is a very simple algorithm and easy to apply. Table 1 illustrates a comparison of the compression algorithms between two different datasets with error bound *e* set to 0.5 for each quantity. The datasets are the same 10-minute interval sets as used previously and also with 1 hour measurement interval. It can be seen in table 1 that all compression algorithms are significantly more effective for 10-minute sampling rate data. This is because with 10-minute sampling rate, the data behaves more linearly.

Table 1. Comparison of the compression ratios for 10 min and 1 hour interval datasets.

Compression algorithm	Temperature (e = 0.5 °C)		Air pressure (e = 0.5 hPa)		Wind speed (e = 0.5 m/s)	
	10 min	1 hour	10 min	1 hour	10 min	1 hour
M-LRbTC, 3 values	3.9	1.85	8.94	3.05	2.62	1.88
M-LRbTC, 4 values	4.46	1.95	9.94	3.45	3.01	2.04
M-LRbTC, 5 values	4.78	1.86	10.75	3.79	3.18	1.97
LTC	9.49	3.19	28.22	8.28	5.09	3.03
DCT	3.07	1.75	4.63	2.72	2.6	1.8

The disadvantage in these linear approximation based algorithms is the latency. These methods are not directly suitable for real-time applications. LRbTC based methods are possible to modify to work better for almost real-time operations: After calculating the new regression line, the first point of the line and line coefficients can be sent. The receiver can use that information until the new point and line are received. Thus, the latency is in maximum when the new regression line is calculated, and it depends on how many point data is used for calculating the regression line.

4 Conclusions and Future Work

Compression algorithms were tested with some real measurement data. In this case, the environmental microclimate data such as temperature, air pressure and wind speed were used. Many environmental quantities are near linear in nature at least if the observation window is short. Linear approximation based compression algorithms benefit from this environmental data behavior. In this research, it was shown that these simple compression algorithms are rather efficient for this kind of data. The performance of compression algorithms for compression compared to reconstruction error was the main property to compare. The next step will be to test these algorithms in edge devices and to take into account the computational complexity of the algorithms.

References

1. Väänänen, O., Hämäläinen, T.: Requirements for Energy Efficient Edge Computing: A Survey. In: The 18th International Conference on Next Generation Wired/Wireless Advanced Networks and Systems NEW2AN 2018, August 29 - 31, 2018, St.Petersburg, Russia. doi: 10.1007/978-3-030-01168-0_1
2. Alrowaily, M., Lu, Z.: Secure Edge Computing in IoT Systems: Review and Case Studies. In: 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, pp. 440-444. (2018). doi: 10.1109/SEC.2018.00060
3. Schoellhammer, T., Osterwein, E., Greenstein, B., et al.: Lightweight temporal compression of microclimate datasets. In: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks IEEE Computer Society, pp. 516-524. (2004)
4. Finnish Meteorological Institute, https://ilmatieteenlaitos.fi/ilmanpaine

5. Norbu, J., Pobkrut, T., Siyang, S., Khunarak, C., Namgyel, T. and Kerdcharoen, T.: Wireless Sensor Networks for Microclimate Monitoring in Edamame Farm. In: 2018 10th International Conference on Knowledge and Smart Technology (KST), Chiang Mai, pp. 200-205. (2018) doi: 10.1109/KST.2018.8426200
6. Muhammad, A. R., Setyawati, O., Setyawan R. A. and Basuki, A.: WSN Based Microclimate Monitoring System on Porang Plantation. In: 2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), Batu, East Java, Indonesia, pp. 142-145. (2018) doi: 10.1109/EECCIS.2018.8692849
7. Rathore, P., Rao, A. S., Rajasegarar, S., Vanz, E., Gubbi J. and Palaniswami, M.: Real-Time Urban Microclimate Analysis Using Internet of Things. In*:* IEEE Internet of Things Journal*,* vol. 5, no. 2, pp. 500-511, April 2018. doi: 10.1109/JIOT.2017.2731875
8. Giorgi, G.: A Combined Approach for Real-Time Data Compression in Wireless Body Sensor Networks. In*:* IEEE Sensors Journal*, vol*. 17, no. 18, pp. 6129-6135, 15 Sept.15, 2017.
9. Bose, T., Bandyopadhyay, S., Kumar, S., Bhattacharyya, A., Pal, A.: Signal Characteristics on Sensor Data Compression in IoT - An Investigation. In*:* 2016 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops*)*, pp. 1-6. London (2016)
10. Ying, Y. B.: An energy-efficient compression algorithm for spatial data in wireless sensor networks. In*:* 2016 18th International Conference on Advanced Communication Technology (ICACT)*, P*yeongchang, pp. 161-164. (2016) doi: 10.1109/ICACT.2016.7423312
11. Fallah, S. A., Arioua, M., El Oualkadi, A. and El Asri, J.: On the performance of piecewise linear approximation techniques in WSNs. In*:* 2018 International Conference on Advanced Communication Technologies and Networking (CommNet)*,* Marrakech, pp. 1-6. (2018)
12. Jaafar K. Alsalaet, Abduladhem A. Ali, Data compression in wireless sensors network using MDCT and embedded harmonic coding, ISA Transactions, Volume 56, Pages 261-267, (2015) ISSN 0019-0578, https://doi.org/10.1016/j.isatra.2014.11.023.
13. Aggarwal, Charu C.: Managing and Mining Sensor Data. Springer. 2013. DOI: 10.1007/978-1-4614-6309-2
14. Lopez, Robert J. Advanced Engineering Mathematics. Addison-Wesley. United States of America (2001). ISBN: 0-201-38073-0
15. Sharma, R.: A data compression application for wireless sensor networks using LTC algorithm. In*:* 2015 IEEE International Conference on Electro/Information Technology (EIT)*, Dekalb,* IL, pp. 598-604. (2015) doi: 10.1109/EIT.2015.7293435
16. Azar, J., Makhoul, A., Darazi, R., Demerjian, J. and Couturier, R.: On the performance of resource-aware compression techniques for vital signs data in wireless body sensor networks. In: 2018 IEEE Middle East and North Africa Communications Conference (MENACOMM), Jounieh, pp. 1-6. (2018) doi: 10.1109/MENACOMM.2018.8371032
17. Parker, D., Stojanovic, M. and Yu, C.: Exploiting temporal and spatial correlation in wireless sensor networks. In*:* 2013 Asilomar Conference on Signals, Systems and Computers*, Pacific G*rove, CA, pp. 442-446. (2013) doi: 10.1109/ACSSC.2013.6810315
18. Zordan, D., Martinez, B., Vilajosana, I. and Rossi, M.: On the Performance of Lossy Compression Schemes for Energy Constrained Sensor Networking. In*:* ACM Trans. Sen. Netw. 11, 1, Article 15, 34 pages. (2014). doi: http://dx.doi.org.ezproxy.jyu.fi/10.1145/2629660
19. Tan, L.: Digital Signal Processing: Fundamentals and Applications. Academic Press, Elsevier, United States of America (2008), ISBN: 978-0-12-374090-8
20. Finnish Meteorological Institute's open data –service. https://en.ilmatieteenlaitos.fi/open-data
21. Matlab polyfit function documentation. https://se.mathworks.com/help/matlab/ref/polyfit.html