

A novel data compression method using improved JPEG-LS in wireless sensor networks

Chih-Chung Lin, Chi-Cheng Chuang, Chien-Wen Chiang, Ray-I Chang

Department of Engineering Science, National Taiwan University, Taipei, 10617, Taiwan

{R97525035, D95525005, R97525037, rayichang}@ntu.edu.tw

Abstract—Wireless Sensor Network (WSN) consists of a group of energy constrained sensor nodes. Since the sensor nodes are often equipped with the battery that usually cannot be replaced or recharged, the way to prolong the lifetime of WSN becomes a critical issue. The utilization of radio communication is the main cause of power consumption. If the data size can be reduced, for instance, through the data compression, the lifetime prolongation can be achieved. In this paper, we propose a novel WSN architecture using improved JPEG-LS data compression. Improved JPEG-LS combines the data pre-collection with modified JPEG-LS predictor. We treat each sensing data as the raw data of a pixel. These raw data are grouped and collected for composing an image. Then the image will be compressed through the improved JPEG-LS. We take the user-acceptable data error defined by the user into consideration and use the error for near-lossless compression in JPEG-LS. The simulation result reveals that our mechanism can reach a good compression ratio with user-acceptable errors. We also make a detailed comparison with the previously proposed methods.

Keywords—Wireless Sensor Network, Data compression, Huffman coding, JPEG-Lossless, Predictor.

I. INTRODUCTION

These sensor nodes are usually equipped with limited resources like power, memory, computation, and have been widely used in the environmental monitoring, surveillance, disaster management and, emergency response [1][2], etc. They are usually deployed in unattended or dangerous areas doing long-time measurements. In these situations, in which recharging or replacing the battery is inconvenient for us, the power saving becomes increasingly important. The utilization of data processing on microcontroller or wireless communication module is the main cause of power consumption. In the comparison of the microcontroller computation cost and wireless transmission cost, the power used to transmit a single bit is equal to the consumption amount of processing thousands of commands [3]. For these reasons, it is the first priority to decrease the quantity of data or the frequency of transmission.

In the WSN, there are several techniques to prolong the lifetime of networks. These techniques are listed as the following.

- Synchronization mechanism [4]: Since the power consumption in idle state of a wireless communication module is almost equal to transmission or reception, it uses the sleep/wakeup scheduling algorithm to make the sensor nodes work and sleep at the appropriate time.
- The shortest path [4]: Since the WSN is a multi-hop network, and the power consumption will be proportional to the hops sensing data passed by. This manner will make a power saving by finding the shortest path to the sink node.
- In-network processing [3], [5], [9]-[12]: Using the data compression or data aggregation techniques to remove the redundancy by exploiting the spatial or temporal correlation in the sensing data.

In this paper, we are only interested in the in-network processing. We will introduce the related researches on the in-network processing in Section 2. Section 3 describes the algorithm and architecture of the improved JPEG-LS. The simulation results are depicted in Section 4. Section 5 is our future work. Conclusions are summarized in Section 6.

II. RELATED WORKS

We introduce the architecture of the JPEG-LS used in this paper and related studies of the in-network processing like data representation and codebook techniques to reduce the size of data which are transmitted by wireless module in the WSN.

JPEG-LS: The core of JPEG-LS is based on the LOCO-I (Low Complexity Lossless Compression for Images) which is developed by the Hewlett-Packard (HP) Laboratories [6]. The JPEG-LS is a mode of operation of JPEG [7]. The main difference between the lossy JPEG and lossless JPEG is the coder. The lossy JPEG uses the discrete cosine transform (DCT) for lossy coding while the lossless JPEG uses the Golomb and Run Length coder for lossless coding.

The architecture of JPEG-LS is illustrated in Figure 1. The modeler samples the image and then predicts it through the predictor. Finally the sampling will be compressed through the lossless coder.

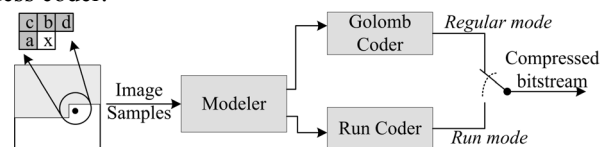


Figure 1. The architecture of JPEG-LS.

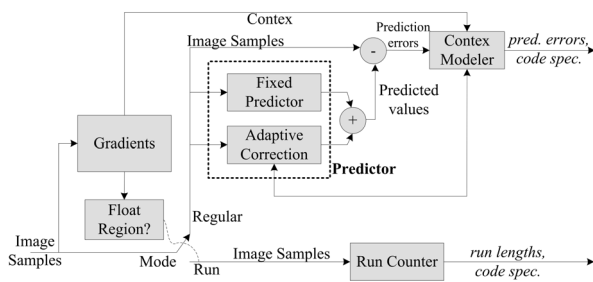


Figure 2. The Modeler in JPEG-LS.

In Figure 2, the image samples will be calculated the $g_1 = d - a$, $g_2 = b - c$ and $g_3 = c - a$ through the gradient, and these differences represent the local gradient. The samples are determined as in the float region while $g_1 = g_2 = g_3 = 0$. Then the samples will be sent into Run mode. On the contrary, the samples will be sent into Regular mode. The fixed predictor is depicted in (1). An edge will be detected by a , b and c . When $c \geq \max(a, b)$, a horizontal edge is detected by $\max(a, b) = a$ and a vertical edge by $\max(a, b) = b$. Similarly, when $c \leq \min(a, b)$, a horizontal edge is detected by $\min(a, b) = a$ and a vertical edge by $\min(a, b) = b$.

$$\hat{x}_{MED} \triangleq \begin{cases} \min(a, b) & \text{if } c \geq \max(a, b) \\ \max(a, b) & \text{if } c \leq \min(a, b) \\ a + b - c & \text{otherwise} \end{cases} \quad (1)$$

Besides, the JPEG-LS also provides the lossy compression which is called near lossless compression. The JPEG-LS uses the Differential Pulse Code Modulation (DPCM) for coding the difference between the sample and predicted value in the Modeler. It is tantamount to allowing that there is an error Δ between the samples and the original data. The specification [8] treats the lossless mode as a special case of near-lossless compression with $\Delta=0$.

Pipelined In-Network Compression: Arici, et al. [9] proposed the Pipelined In-Network Compression. The proposed method extends the time of sensing data stay in the aggregation node. In this duration, the sensing data are combined into one packet and the redundancy will be removed. The sensing data before compressed in the WSN has the following formation: <sensing data, node ID, timestamp>. Then the compressed packet has another formation: <shared prefix, suffix list, node ID list, timestamp list>. Shared prefix is the common part of the sensing data while the suffix list is the remained part. Node ID list is the conjunction of sensor ID. Timestamp is the conjunction of the time when the sensing data has been measured. In Figure 3, the total bits are reduced from 33 bits to 27 bits. Since the more common parts the sensing data have, the shorter the compressed packet will be. The Pipelined In-Network Compression obviously works very efficiently when the data have high correlation. But when the sensing data have low correlation, the common part would no longer be short and lead to an inefficient compression ratio.

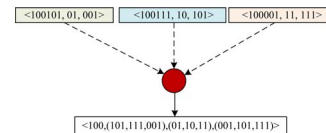


Figure 3. Pipelined In-Network Compression [5].

Static Huffman coding: Marcelloni, et al. [10] proposed a simple compression algorithm for analog-to-digital (ADC) converter which is installed on a sensor node to sense data. This method used the DC coefficients (brightness component) codebook in JPEG baseline algorithm as the data compression. He considered the sensing data to be the continuous and high correlated data. Then he calculated the difference d_i between the former and the latter sensing data. Since the DC coefficients codebook is known before coding with the probabilities of the sensing data, the code book can compress the d_i with high compression ratio. This method is simple and easy to implement. But there are some deficiencies when the statistics of the data is unknown. The algorithm is not suitable for real time sensor data since the statistics of the sensing data must be known beforehand in order to assign the probabilities for Huffman coding in the DC coefficients code book.

Modified Adaptive Huffman coding: Tharini, et al. [11] proposed a modified Adaptive Huffman coding. The proposed method improved the Static Huffman coding in [10] with adaptive Huffman coding [12] and did not require the knowledge of statistics of the sensing data for assigning the probabilities in advance. Adaptive Huffman coding has two procedures when there is no prior knowledge of the probabilities. The binary tree is constructed in the processing of coding. In the beginning, there is only one node named Not Yet Transmitted (NYT), and the binary tree is constructed from NYT. In the long run, the code consists of two parts:

- Prefix indicates the code by traversing the tree which includes data.
- Suffix indicates the binary representation of the sensing data.

The sensor node just transmits the code by combing the prefix and suffix part. This brings a problem where the initial data has less probability to get a shorter code since the initial probability is assigned after the binary tree first constructed.

Modified Adaptive Huffman coding is combined with both the advantages in the Static Huffman coding and Adaptive Huffman coding. The leaves of the tree are assigned with a set of symbols, rather than one symbol in [10]. The code is changed and finally made up of two parts:

- Prefix indicates the code by traversing the tree which includes the data.
- Suffix indicates the array index of the data appears in the set of the symbol.

The simulation result showed the modified Adaptive Huffman coding has better compression ratio in the medially correlated data. But it still has poor compression ratio in the high correlated data in comparison with the Static Huffman coding in [10].

Both Marcelloni [10] and Tharini [11] proposed the methods for data compression in the WSN. Although the Static Huffman

coding has effective compression ratio, it could not be suitable for the low correlated data. The sensing data is not complete in high correlation but sometimes low correlation in some harsh environments. The modified Adaptive Huffman coding solved the problem in [10] and obtained a better compression ratio in low correlated data. In the high correlated data, it still had lower compression ratio.

The compression ratio of Marcelloni and Tharini in the different data type is classified in Table 1.

Table 1. Compression Ratio of Marcelloni and Tharini in the different data type.

Data type	Method	Compression Ratio
High Correlated data	Marcelloni[10]	Good
	Tharini[11]	Poor
Low Correlated data	Marcelloni[10]	Poor
	Tharini[11]	Good

III. PROPOSED METHOD

Since the sensor nodes are deployed under various circumstances, the correlation of data is unpredictable. Our anticipated goal is to reach good compression ratio in the high correlated data and low correlated data. Since the application of sensor nodes is mostly used to report the status of interested area to users [13][14], it is more intuitive to report with an image. Users can learn about the measurements or data distribution from the image. Also, users can even specify some area for further information. We expect that the image will not be produced in the beginning through video camera [15]. It will increase the power consumption while the sensor node is equipped with a video camera.

For these reasons, we propose a novel WSN architecture using improved JPEG-LS data compression. The improved JPEG-LS consists of the data pre-collection and modified JPEG-LS predictor. The compression ratio will be improved after a period of collection. We treat the sensing data as the raw data of a pixel. These raw data will be collected to form an image and then compressed through the improved JPEG-LS.

A. Data Pre-collection

The purpose of the data pre-collection is for transmitting a shorter symbol after some time interval. The architecture of the improved JPEG-LS is shown in Figure 4, we divide the sensor network into three layers.

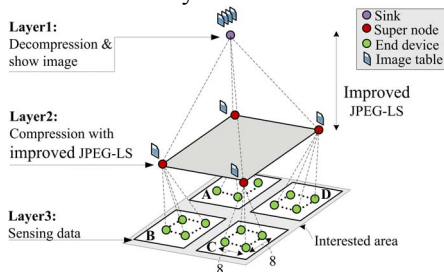


Figure 4. The architecture of the improved JPEG-LS.

- **Layer 3:** We divide the interested area into pieces of sub-regions. For example, in Figure 4, it is divided into four sub-regions (A, B, C and D). There are 64 end devices in each sub-region. The end devices are responsible for sensing the data and transmitting it back to the super node in the layer 2.
- **Layer 2:** The super node is responsible for processing and forwarding the data to sink. The super nodes integrate the sensing data from layer 3 to form an image and then compress it through the improved JPEG-LS. There are two stages for compressing data, the initial stage and the compression stage. In Figure 5, it shows the mechanism of data pre-collection. The super node will make a subtraction of P_{k+n} and P_{t_i} after t_k in the compression stage.

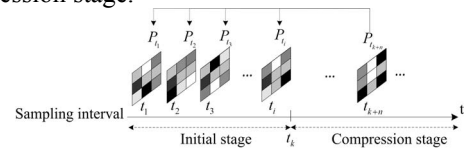


Figure 5. The mechanism of data pre-collection.

Figure 6 depicts the procedure of the initial stage. The system will take some time to collect the lossless image. If there is no identical P_{t_i} in the image table, the super node will store it into the image table. The image table is shown as Table 2. There are two columns of index and lossless image. The $x1, x2 \dots xi$ represent any integer which is smaller than k . After t_k , the system will start the compression stage. The time length of t_k is determined by the correlation of sensing data. It takes a shorter t_k with the high correlated data while the low correlated data needs a longer t_k .

Table 2. Image table on the super node and the sink in the initial stage.

Index	Lossless Image
t_{x1}	$P_{t_{x1}}$
t_{x2}	$P_{t_{x2}}$
$\vdots$	$\vdots$
t_{xi}	$P_{t_{xi}}$

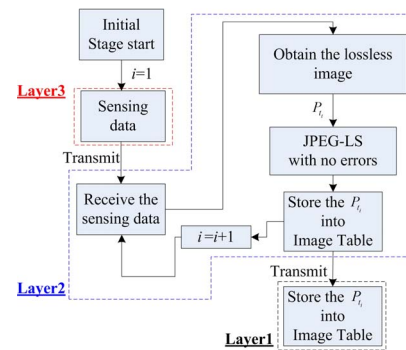


Figure 6. The procedure of initial stage.

The detailed process of compression stage is described in Figure 7. We consider the user-acceptable error Δ

which is defined by the user. The error Δ can be distributed into every sensing data evenly by adding the error into JPEG-LS. It will be more inaccurate by making a subtraction of $P_{t_{k+n}}$ and P_{t_i} without adding the error to the JPEG-LS. When the layer 2 obtain the lossless image $P_{t_{k+n}}$, the $P_{t_{k+n}}$ will be compressed with lossy and lossless compression. The lossless image will be stored into the image table temporary.

If we can find an image in the image table within an error Δ after making the subtraction of samples and lossless images in the image table, the corresponded index will be transmitted. The lossless image will be transmitted back to the layer 1 if there is no error whin the user-acceptable range.

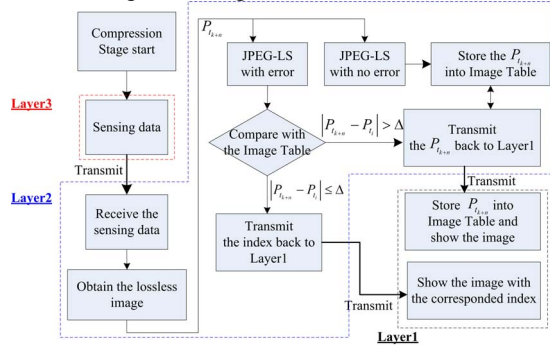


Figure 7. The procedure of compression stage.

- **Layer 1:** The data which the sink node received from the layer 2 will be two cases. If the sink node receives the index, it will display the image with the corresponded index in the image table. If the sink node receives the lossless image $P_{t_{k+n}}$, the sink node will store it into the image table and display it.

B. JPEG-LS with Modified Predictor

We add the user-acceptable error Δ and modify the predictor in JPEG-LS for compression ratio improvement. For instance, in Figure 8, we can make the compressed data with an error Δ by using the gradient computation. If $g_1 \leq \Delta$, $g_2 \leq \Delta$ and $g_3 \leq \Delta$, the subtractions between every pixel on the image will be less than or equal to error Δ . The image which satisfies (2) will be the user-acceptable image [6].

$$|P_{t_{k+n}}(x, y) - P_{t_i}(x, y)| \leq \Delta \quad (2)$$

Figure 8. The subtractions between each pixel on the image.

The subtractions between $P_{t_{k+n}}(x, y)$ and $P_{t_i}(x, y)$ will be greater than error Δ with some sudden drastic changes in the environment which makes the sensing data increase or decrease in a short time. In these situations, the lossless

image stored temporary in the image table in layer 2 will be transmitted to the image table of the layer 1 for updating.

Generally, the sensor nodes are used to measure temperature, humidity or other climate-related figures [16], [17]. Therefore, we modified the predictor in the JPEG-LS for better compression ratio under the temperature measurement.

We modified a more efficient predictor for the image produced by our topology. We keep the edge detection in the predictor. In Figure 9, since the formation of the images in the layer 2 are grid-patterned. If we detect the cross area of four separated pixel, we return the sample value x directly.

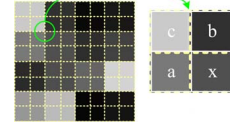


Figure 9. The image obtained by super node in the layer 2.

IV. SIMULATION RESULT

We use the Network Simulator 2 (NS2) to implement the simulation. The topology of the network is based on Figure 4. The compression ratio is calculated as (3).

$$\text{Compression Ratio} = \left(1 - \frac{\text{Compressed data}}{\text{Original data}}\right) \quad (3)$$

Our simulation uses the temperature data on MARS acquired from Planetary Atmospheres Node of the Planetary Data System (PDS) in the NASA [18]. We consider two sets of data with significant differences. One set is low correlated data while the other is high correlated. The autocorrelation of the data is illustrated in Figure 10, computed by the PAST software [19]. The simulation parameters are set in Table 3.

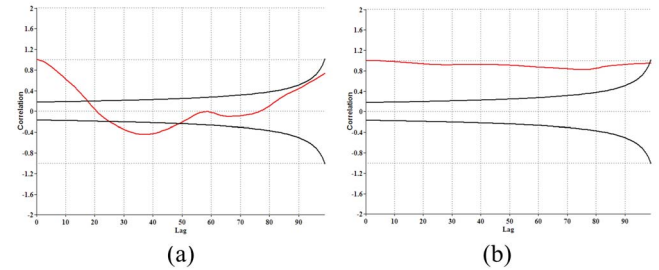


Figure 10. Autocorrelations of the (a) low and (b) high correlated temperature data on MARS.

Table 3. Simulation parameters.

Attribute	Value
Topology	As shown in Figure 4
Protocol	IEEE 802.15.4 [20]
Nodes	Layer 3: 256(End device) Layer 2: 4(Super node) Layer 1: 1(Sink)
Simulation time	200 min
Compression Ratio for Index transmission	85%
Compression Ratio for lossless image transmission	52.4%

User-acceptable error(°C)	0, 1, 2, 3, 4, 5
-----------------------------	------------------

We make the end devices sense the data once per minute. The user-acceptable error Δ is determined by the real image which is received in layer 1. When the error is bigger than 5°C , there will be a significant error on the image.

In Figure 11, the simulation result shows the relations between the time and compression ratio under the varied error Δ in the high correlated data. We observe the time interval 0~125min for obvious changes. In the beginning, transmitting the lossless image at initial stage leads to the compression ratio maintained at 52.4%.

After a time interval with data collection, the compression ratio can be improved. The time will be reduced gradually for reaching the same compression ratio when the user-acceptable error Δ becomes larger.

The more the data collected over time, the higher the probability of transmitting index will be. The compression ratio will approach about 85% in the long run.

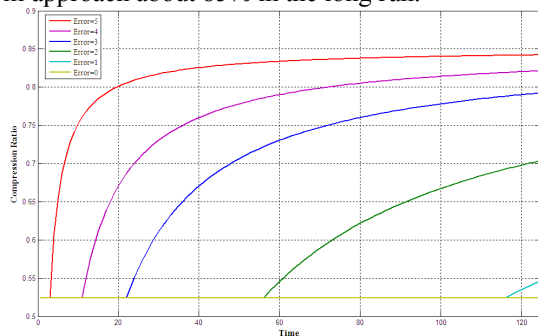


Figure 11. Time over compression ratio under the varied error Δ in the high correlated data.

In Figure 12, the simulation result shows the relations between the time and compression ratio under the varied error Δ in the low correlated data. We observe the time interval 80~125min for obvious changes.

The compression ratio will be improved at 83min and reached 63.2% at 125min under the user-acceptable error 5. By contrast, the compression ratio will be improved at 117min and reached only 53.6% at 125min under the user-acceptable error 1.

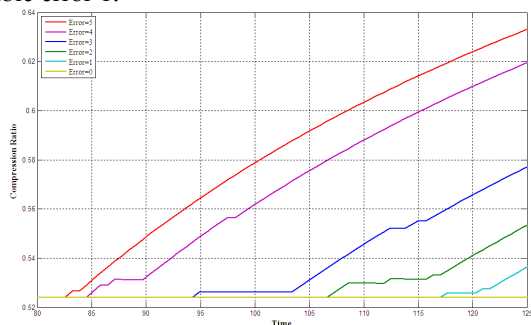


Figure 12. Time over compression ratio under the varied error Δ in the low correlated data.

The compression ratio will be the lowest 52.4% in the two data types under the user-acceptable error 0.

We compare our method with the methods of Marcelloni[10] and Tharini[11] in the same simulation environment. We analyze the relations between the time and compression ratio under the user-acceptable error 5 in the high correlated data and low correlated data.

In Figure 13, our method obtain a better compression ratio than Static Huffman Coding after 10~15min in the high correlated data. If the system collects more information, our method will improve the Static Huffman Coding by 13% and Modified Adaptive Huffman Coding by 45% when the compression ratio reaches about 85%.

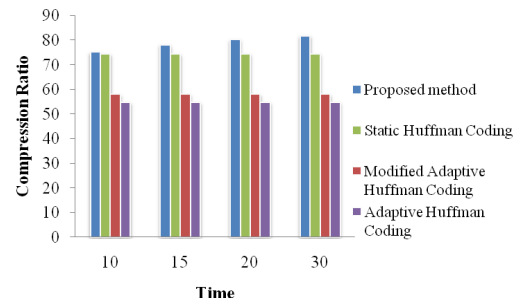


Figure 13. The time over compression ratio with different methods in the high correlated data.

In Figure 14, our method takes a longer time 85~90min to obtain a better compression ratio than Modified Adaptive Huffman Coding in the low correlated data. If the system collects more information, our method will improve the Static Huffman Coding by 45.3% and Modified Adaptive Huffman Coding by 25.9% when the compression ratio reaches about 67%.

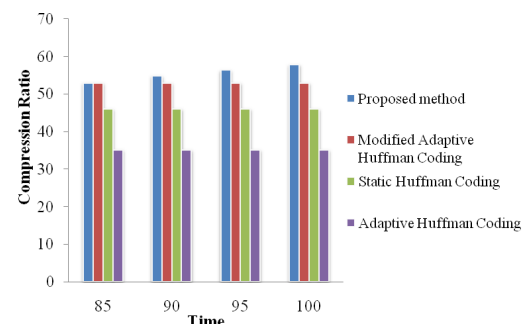


Figure 14. The time over compression ratio with different methods in the low correlated data.

In Table 4, we make a statistics chart of required time to reach the same compression ratio of the previously proposed methods in [10], [11] and [12]. It provides a reference for the system design in the future.

Table 4. The relations between time and user-acceptable error in the two data types.

Data type	User error	S.H.C.[10] (74.3%)	M.A.C.[11] (57.9%)	A.H.C.[12] (54.6%)
High Correlated data	1	>200min	136min	124min
	2	161min	67min	60min
	3	67min	26min	23min
	4	34min	13min	11min

