technologies

MDPI

Article

Lossless Compression of Aldebaran-I Telemetry Data Using the On+ Algorithm

Flávio Barros [1,*], Letícia Correia [1], Caio Magno [1], Christian Diniz [1], Gean Sousa [1], Allan Kardec Barros [1] and Luis Claudio Silva [2]

[1] Department of Electrical Engineering, Federal University of Maranhão, São Luís 65080-040, Maranhão, Brazil; leticia.correia@ifma.edu.br (L.C.); magno.caio@ufma.br (C.M.); christiandiniz@gmail.com (C.D.); gean.sousa@ufma.br (G.S.); allan.kardec@ufma.br (A.K.B.)

[2] Department of Aerospace Engineering, Federal University of Maranhão, São Luís 65080-040, Maranhão, Brazil; claudio.luis@ufma.br

* Correspondence: flavio.barros@ifma.edu.br

Abstract

Lossless compression of telemetry data in satellites is essential due to the stringent limitations of bandwidth and onboard storage. Traditional methods based on information theory and entropy coding, such as Huffman and Arithmetic coding, exploit statistical redundancy but still present opportunities for improvement when applied to data with low redundancy, large alphabets, and near-uniform symbol distributions. This study proposes On+, a novel lossless compression algorithm for satellite telemetry data. Using real telemetry data captured by the Aldebaran-1 CubeSat satellite, a dataset consisting of 600 binary files was created. The performance of the proposed algorithm was evaluated in comparison with classical methods (Huffman and Arithmetic coding) and several commercial compressors (.rar, .zip, .7z, .xz, and .gz). The On+ algorithm achieved an average compression rate of 29.19%, with a standard deviation of 1.26 and a median of 29.09%, outperforming the traditional Huffman coding and Arithmetic coding methods in terms of compression efficiency. Furthermore, it exhibited superior performance compared with all commercial solutions evaluated, many of which resulted in file expansion (negative compression rates). These results demonstrate the effectiveness and viability of the On+ algorithm for optimizing telemetry data compression in satellites.

Keywords: compression algorithm; lossless compression; entropy coding; cubesat satellites; telemetry data

Check for updates

Academic Editors: Shizhe Hu and Zhengzheng Lou

Received: 11 May 2026
Revised: 29 May 2026
Accepted: 6 June 2026
Published: 12 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The process of reducing the size of data transmitted from a device, such as a satellite, to a ground station is known as telemetry stream compression. For data storage or transmission, compressing telemetry streams is crucial, especially in situations where energy [1,2] and bandwidth are limited [3]. Through compression, reduced data volume leads to lower transmission costs, as both transmission time and bandwidth usage are often expensive [4].

According to [5], monitoring the health of a satellite is fundamental to ensuring its proper operation. This process involves the continuous collection of data on its status and performance, encompassing parameters such as battery charge levels, solar panel efficiency, the temperature of critical components, as well as the performance of propulsion and communication systems, including their efficiency and integrity [6]. Additionally, navigation and attitude control data, such as the satellite's position and orientation, are also

monitored. In this context, the transmission of telemetry data by satellites, as discussed by [7–10], plays a crucial role, as it enables maintenance, efficient operation, and the success of space missions [6,11,12].

Telemetry data is essential for the daily operation of a satellite, as they support mission operations [5] by enabling the monitoring and control of the satellite, as well as orbit adjustments, orientation changes, and payload management [7,8]. Consequently, telemetry monitoring is crucial to ensure that the satellite operates within safe parameters [4,13], preventing potential failures such as overheating or even loss of control [14].

In various scenarios, lossless compression may not provide substantial benefits, as evidenced by certain existing methods. In some cases, the process may even lead to an increase in file size due to the incorporation of control information or the additional overhead associated with the compression process [15]. This overhead can arise from multiple factors, including the execution of auxiliary processing steps, the inclusion of control data, and, in some instances, the need to satisfy temporary storage requirements. Consequently, the time and computational resources required to perform these operations are considered integral components of the overall compression overhead [16]. Control information typically includes headers, metadata, and other auxiliary data necessary to accurately reconstruct the original data structure.

This study aims to create a lossless compression algorithm specifically designed for satellite telemetry data, with the purpose of achieving higher compression rates compared with those offered by currently available methods. To achieve this goal, the work proposes a structured approach following three key stages. Firstly, it seeks to conduct a thorough study and detailed analysis of lossless compression algorithms, exploring their characteristics, limitations, and optimization potential. Subsequently, the analysis focuses on the development of a specific algorithm designed for the lossless compression of telemetry data from the Aldebaran-1 satellite, with particular attention to the unique characteristics and technical requirements inherent to this type of data. Finally, the study plans to perform a series of tests and validations to compare the proposed algorithm's performance with existing methods in the literature, assessing its effectiveness and superiority in real-world scenarios.

2. Theoretical Framework and Analysis of Related Work

Several studies have investigated lossless data compression methods for satellite telemetry data. These studies focus on techniques that optimize the compression process, including self-learning [11], prediction based on neural networks and entropy coding [12], algorithms based on decision trees [14] and the use of artificial intelligence resources [3].

The work presented in [11] conducted a study on satellite telemetry data classification based on machine learning. Four basic classes of telemetry data were suggested and studied using time series features and information entropy analysis.

As well, [12] presented a prediction based on recurrent neural networks to improve satellite telemetry compression. The proposed method consists of a decorrelation step and an entropy coding step.

In [14], a satellite fault detection and diagnosis based on data compression and an improved decision tree is proposed. The method based on data compression and the improved decision tree algorithm improved the accuracy and efficiency of satellite fault diagnosis and detection.

Also in 2022, a preliminary study was carried out with the aim of verifying the effectiveness of one of the most representative AI-based lossless telemetry compression algorithms on three different NASA datasets [3].

In the context of data compression, the most widely used algorithms are Huffman Coding [17–19] and Arithmetic Coding [20–22].

The Huffman technique, developed by David Huffman, assigns prefix codes to the data to be encoded based on two premises: (I) symbols that occur more frequently should have shorter codes, while less frequent symbols should have longer codes; and (II) the two least frequent symbols must have codes of equal length. In the decoding process, the generated dictionary of symbols and codes is used to recover the original sequence, since in its prefix code [19,23], each symbol has a unique code, ensuring no ambiguity during decoding.

Arithmetic coding is a technique in which the encoding and decoding processes are carried out through continuous mathematical operations over numerical intervals, resulting in a representation of the entire data sequence as a single real number within an interval. In certain situations, arithmetic coding can outperform the Huffman technique, especially when the alphabet is small and the probability distributions are skewed [17,22].

Both Huffman and arithmetic coding require reading the entire dataset in advance to calculate the probabilities associated with each symbol [24]. When the dataset is small, this requirement is not a major concern. However, as the data stream increases, two critical aspects must be considered: (I) the additional time needed to read the data and compute the probabilities, and (II) the assumption that the calculated probabilities will remain constant throughout the encoding process.

Still in this context, Huffman coding and Arithmetic coding are methods based on entropy [22,25,26] and exploit statistical redundancy in the data to achieve high compression rates. However, the average number of *bits* per message symbol cannot be less than the entropy:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i), \tag{1}$$

where the data distribution p_i is known in advance.

Furthermore, entropy-based coding is independent of the specific characteristics of the medium [27], since this method starts by initially counting the frequency of each symbol according to its occurrence in the file [28]. This way, it is independent of the type of information being compressed. For symbols declared in the original file, these new symbols are fixed and do not depend on the file contents. The length of these new symbols is variable and varies with the frequency of inclusion of symbols from the original file [29]. Thus, entropy encoders exploit the estimated statistical properties of their message to get closer to the theoretical compression limit [26].

In this scenario, compression algorithms can be evaluated by various criteria, including speed, computational complexity, memory usage, compression ratio, and others [30,31].

Equation (2) provides a measure of the ratio between the original file and the compressed file. In other words, the higher the Compression Factor (CF), the better the encoding algorithm. The second quality measure presented in Equation (3) indicates the amount of data processed per unit of time. The Transfer Rate (TR) shows the algorithm's efficiency in terms of bandwidth occupancy (throughputs) on communication channels.

$$CF = \frac{\text{Original file in bits}}{\text{Compressed file in bits}} \tag{2}$$

$$TR = \frac{\text{File size in bits}}{\text{Total Processing Time (sec)}} \tag{3}$$

Equation (4) presents the Compression Rate (CR):

$$CR = \left(1 - \frac{\text{Compressed file in bits}}{\text{Original file in bits}}\right) * 100 \tag{4}$$

Compression Rate indicates the percentage of space reduced after compression. Thus, the higher the CR value, the more efficient the lossless compression algorithm is at reducing the data volume.

3. Proposed Compression Algorithm

In this section, we present the methodology used for the development and experimentation of the lossless compression algorithm aimed at satellite data, with a specific focus on housekeeping data, i.e., data related to the monitoring and control of onboard systems essential for ensuring the safe and efficient operation of the spacecraft. The objective of the algorithm is to reduce the volume of hexadecimal data without compromising its integrity. Furthermore, we detail the formal notation and entropy model of the proposal, present an illustrative example for better understanding, describe the compression and decompression steps, and provide a preliminary analysis of the computational complexity of the method. Finally, we discuss the metrics used and the benchmark employed to evaluate the effectiveness of the lossless compression algorithms.

3.1. On+ Proposed Algorithm

Figure 1 presents the encoding and decoding flow of the technique proposed in this work, named On+.

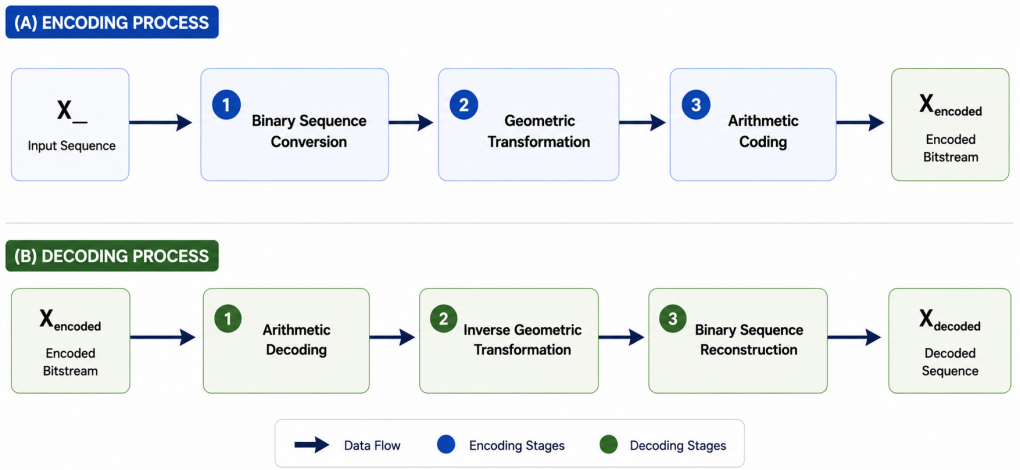


Figure 1. Proposed encoding and decoding pipeline. (**A**) Encoding process. (**B**) Decoding process.

The encoding process (A) of the On+ algorithm starts with the input sequence X, which is sequentially subjected to three main steps: conversion to a binary sequence, geometric transformation, and arithmetic coding, resulting in the compacted bitstream X_{encoded} with reduced size.

The decoding process (B), in turn, performs the inverse operations. Starting from the encoded bitstream X_{encoded}, arithmetic decoding is executed, followed by the inverse geometric transformation and, finally, the reconstruction of the original binary sequence, recovering the decoded sequence X_{decoded}.

The proposed algorithm leverages the statistical redundancy intrinsic to the data to enable lossless compression. However, when the data exhibits low statistical redundancy after reading, associated with a large alphabet and uniformly distributed probabilities among symbols, the On+ algorithm performs preprocessing aiming to transform the probability distribution in order to explore its asymmetry and skewness. This scenario, characterized by lower entropy and greater variability in symbol frequencies, creates a favorable environment for the application of entropy-based coding methods, such as the one proposed.

3.1.1. Notation

We present a formal notation for the description of the On+ algorithm. Let the vectors be:

- $X = \{x_1, x_2, x_3, \ldots, x_n\}$, a binary vector of size n, where $x_i \in 0, 1$, and $x_n = 1$.
- $Y = \{y_1, y_2, y_3, \ldots, y_m\}$, a conversion vector of size m, where $y_j \in \mathbb{N}_{>0}$.

Let i be the index of the elements in vector X, with $i = 1, \ldots, n$, and let j be the index of the elements in vector Y, with $j = 1, \ldots, m$. We define k, initially set to 1, as the number of consecutive *zeros* between $x_i = 0$ and $x_{i+k} = 1$.

The following encoding rule defines the conversion from X to Y:

$$y_j = \begin{cases} 1, & if \;\; x_i = 1 \\ \\ k+1, & if \;\; x_i = 0 \;\; and \;\; x_{i+k} = 1 \end{cases} \tag{5}$$

To illustrate the transformation proposed in this section, Figure 2 presents an example of applying the geometric transformation to a hypothetical sequence. We consider the binary vector $X = [0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1]$, with the goal of transforming it into vector Y. To do this, we sequentially traverse each element of vector X.

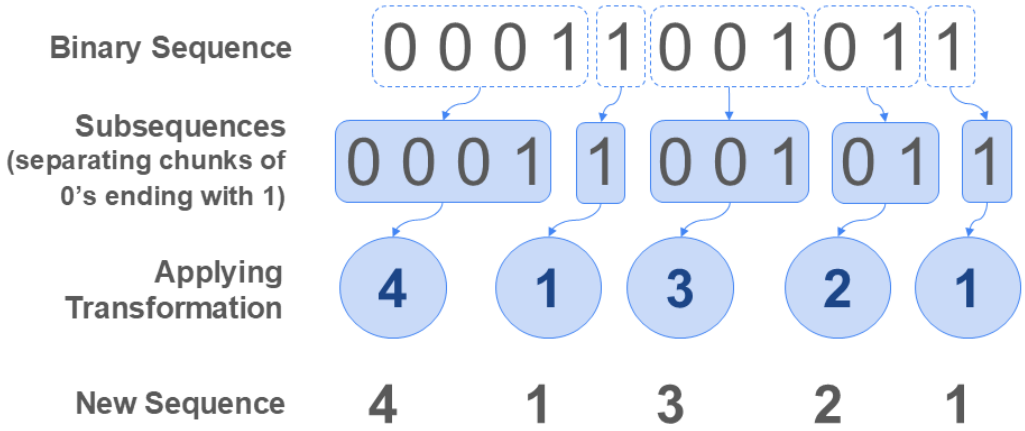


Figure 2. Example of applying geometric transformation to a hypothetical sequence.

The result of the transformation of the binary vector $X = [0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1]$ is the vector $Y = [4, 1, 3, 2, 1]$. Each symbol $Y \in \Sigma$ has an associated probability $P(Y)$, estimated based on the frequency of occurrence of the symbols in vector Y. This vector is then directly submitted to the Adaptive Arithmetic coding method [17,28]. Therefore, after processing all symbols, we obtain a single number n, which represents the entire sequence.

For the decoding process of the conversion rule from Y to X, we have, by definition, the concatenation rule $A \bigoplus B = [a_1, a_2, \ldots, a_n] \bigoplus [b_1, b_2, \ldots, b_m] = [a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m]$, where $\bigoplus$ is the concatenation operator and $\bigotimes$ is the repetition operator.

Considering the concatenation rule as described in the conversion rule for decoding:

$$X = \begin{cases} X \bigoplus [1], & if \;\; y_j = 1 \\ \\ X \bigoplus (([0] \bigotimes (y_j - 1)) \bigoplus [1]) \;, & if \;\; y_j > 1 \end{cases} \tag{6}$$

where, for each element y_j in vector Y, a bit 1 is included in vector X when $y_j = 1$. Otherwise, $(y_j - 1)$ zeros are concatenated, followed by a bit 1.

3.1.2. Entropy Model Estimation

We present a formal model for the description of the On+ algorithm. The steps are as follows:

(i) Given a Bernoulli sequence $X = \{x_1, x_2, x_3, \ldots, x_{N_X}\}$, a binary vector of length N_X where $x_{N_X} = 1$, with each $x_i \in \{0,1\}$ and $p(x_i = 1) = p$, the entropy H_X of this sequence is given by:

$$H_X = -[(1-p)\log_2(1-p) + p\log_2 p] \tag{7}$$

The average length L_X of sequence X, in bits, is given by:

$$L_X = N_X \cdot H_X \tag{8}$$

(ii) Transformation of sequence X into Y: Sequence X is transformed into a sequence $Y = \{y_1, y_2, y_3, \ldots, y_{N_Y}\}$, a vector of size N_Y, where $N_Y < N_X$ and $y_i \sim \text{Geom}(p)$ follows a geometric distribution with the same parameter p as sequence X. The entropy H_Y is given by the equation:

$$H_Y = -\left[\frac{(1-p)\ log_2\ (1-p) + p\ log_2\ p}{p}\right] = \frac{H_X}{p} \tag{9}$$

The average length L_Y of sequence Y, in bits, is given by:

$$L_Y = N_Y \cdot H_Y = N_Y \cdot \frac{H_X}{p} \tag{10}$$

From the comparative analysis between sequences X and Y, it is first concluded that the entropy of Y is always greater than or equal to that of X ($H_Y \geq H_X$), given that $p \in [0,1]$. Secondly, it is verified that the distribution of X differs from the distribution of Y, which takes the form of a geometric distribution (Geo(p)). This distortion of the original distribution of X occurs due to the fact that the symbols of Y are formed from variable-length partitions of the binary sequence of X.

The relationship between the size, in bits, of the transformed sequence Y and the original sequence X is given by:

$$R = \frac{L_Y}{L_X} = \frac{N_Y H_Y}{N_X H_X} = \frac{\frac{N_Y H_X}{p}}{N_X H_X} = \frac{N_Y}{N_X} \cdot \frac{1}{p} \tag{11}$$

Due to the fact that each symbol of the sequence Y is formed by a partition of X that contains only a single 1 bit, the ratio $\frac{N_Y}{N_X} = p$ and the relation R reduces to

$$R = p \cdot \frac{1}{p} = 1 \tag{12}$$

It is observed that the process does not change the file size in bits. It is verified that there is no effective compression during the transformation, even though the entropy of Y is greater than that of X ($H_Y > H_X$), but the number of symbols in Y is smaller ($N_Y < N_X$), maintaining the ratio $R = 1$. However, the asymmetric distribution assumed by Y favors the performance of entropy coders, such as the arithmetic coding algorithm, which can exploit this characteristic to achieve more efficient compression.

3.1.3. Encoding

Algorithm 1 presents the encoding process in detail. In this step, the original data is analyzed and represented in a more compact way. The On+ algorithm plays a crucial role in replacing repetitive patterns with shorter symbols, thereby reducing redundancy in data and eliminating unnecessary information or representing it more efficiently.

Algorithm 1: Encoding

Input: X $(binary\ \ vector)$
Output: $encoded$ $(decimal\ \ vector)$
begin
  $X \leftarrow [X\ \ 1]$
  $X_{geometric} \leftarrow$ [ ] $(empty\ \ vector)$
  $count0 \leftarrow 0$
  for $i \leftarrow 1$ ***to*** $size(X)$ **do**
    if $X(i) = 0$ **then**
      $count0 \leftarrow count0 + 1$
    else
      if $count0 > 0$ **then**
        append $(count0 + 1)$ to the end of $X_{geometric}$ vector
      else
        append 1 to the end of $X_{geometric}$ vector
      $count0 \leftarrow 0$
  $encoded \longleftarrow AdaptativeAritmethicEncoding(X_{geometric})$

To ensure that the stopping criterion of the encoding process is always satisfied, the algorithm initially adds a bit equal to 1 at the end of the input binary sequence, as shown in the initialization line of the vector $X \leftarrow [X\ \ 1]$. This additional bit acts as an end-of-sequence marker, ensuring that the last sequence of zeros is correctly processed and encoded. During the decoding stage, this bit is removed at the end of the process to exactly restore the original binary sequence.

3.1.4. Decoding

The lossless decoding process of the file generated by the encoding method is presented in detail in Algorithm 2. Decoding consists of reconstructing the original data from the compressed sequence by reversing the operations performed during the encoding stage. In this phase, the On+ algorithm restores the original binary vector from the encoded file by applying the inverse operations of those executed in Algorithm 1.

Initially, the algorithm performs adaptive arithmetic decoding to recover the vector $X_{geometric}$. Next, each element of this vector is used to reconstruct the original binary sequence by adding $(X_{geometric}(i) - 1)$ zeros followed by a bit equal to 1. At the end of the process, the last bit of the reconstructed sequence is removed. This bit corresponds exactly to the additional bit inserted during the encoding stage in Algorithm 1, which was used to ensure the stopping criterion of the encoding process. Therefore, its removal does not cause any information loss but simply restores the original binary sequence correctly.

Algorithm 2: Decoding

Input: *encoded* (*decimal vector*)
Output: *decoded* (*binary vector*)
begin
  $X_{geometric} \longleftarrow AdaptativeAritmethicDecoding(encoded)$
  for $i \leftarrow 1$ ***to*** $size(X_{geometric})$ **do**
    append $(X_{geometric}(i) - 1)$ zeros to the end of *decoded*
    append 1 to the end of *decoded*
  Remove the last bit of *decoded*

3.2. Performance and Efficiency Evaluation of the On+ Algorithm

The evaluation of the performance and efficiency of the On+ algorithm will be conducted through a structured analysis encompassing computational complexity, the definition of specific metrics, and the execution of comparative benchmarks. Initially, the complexity analysis will be performed to determine the algorithm's computational resource consumption as a function of input size, following the approach proposed by [32]. This analysis will consider two main dimensions: time complexity, expressed using the asymptotic notation O(n), which quantifies the worst-case execution time, and space complexity, which assesses the additional memory usage required to process the input data [33]. This step will be crucial for understanding the efficiency of On+ and predicting its behavior across different scenarios, including worst-case, best-case, and average conditions, enabling comparison with alternative algorithms, the selection of appropriate improvements, and ensuring adequate performance at scale.

To quantify the performance of On+, the compression ratio (CR) was adopted as one of the main metrics, calculated according to Equation (4). This ratio expresses the percentage reduction in file size: positive values indicate successful compression, while negative values denote expansion (increase in size) relative to the original file. This metric was chosen due to its relevance in the context of telemetry data, where computational efficiency and resource savings are priorities.

Finally, to validate the efficiency of On+ compared with established methods, a benchmark will be conducted using widely recognized compression techniques in the literature, such as Huffman coding and Arithmetic coding. The On+ algorithm, presented in Section 3.1, will undergo tests that evaluate both the compression ratio and execution time, enabling a robust comparative analysis.

4. Results

In this section, we evaluate the performance of the On+ algorithm in comparison with other lossless compression methods. Initially, we present the satellite telemetry datasets used in the experiments and describe the experimental platform, highlighting its main characteristics and advantages. Next, we conduct benchmark tests using a predefined set of parameters, comparing the proposed method with classical compression techniques, such as Huffman and Arithmetic Coding, as well as widely used commercial compressors, including .rar, .zip, .7z, .xz, and .gz.

We also perform an analysis of the entropy of the files in the repository, as well as the entropy associated with the success probability (p). Finally, we analyze the time and space complexity of the On+ encoding and decoding algorithms, together with the Arithmetic and Huffman compression methods, including a comparative table that summarizes the main results obtained.

4.1. Repository and Experimental Platform

For the experiments, real telemetry data from the Aldebaran-1 satellite, a 1U CubeSat, were used. This CubeSat was developed by the Laboratory of Electronics and Space Embedded Systems (LABESEE/UFMA) in partnership with SpaceLab/UFSC, with the main objective of assisting in the rescue of fishermen in emergency situations along the Brazilian coast, by receiving alert signals transmitted by vessels facing adverse ocean conditions [34].

The repository contains 600 binary files (.bin) captured from the real environment of the Aldebaran-1 satellite. These files have sizes ranging from 0.164 kB (minimum) to 0.166 kB (maximum). After data extraction, the files were subjected to lossless compression techniques, including the traditional Huffman and Arithmetic methods, as well as commercial compressors in .rar, .zip, .7z, .xz, and .gz formats.

Figure 3 presents the engineering model of the Aldebaran-1 satellite and the output of its telemetry transceiver processor. The satellite transceivers are based on LoRa (Long Range) radios [35], used for signal downlink and uplink operations.

To prevent transmission distortions, LoRa modulation includes native error detection/correction at the physical layer. It uses Forward Error Correction (FEC) to enhance signal resilience against interference, which is crucial for Long Range communications with low Earth orbit satellites. LoRa adds redundant bits to each packet, enabling bit error detection and correction without retransmission; this is configurable for noisy environments. To counter burst interference, LoRa employs interleaving to spread errors across codewords, improving the effectiveness of its Hamming code correction. Additionally, LoRa uses a cyclic redundancy code to ensure data integrity in the message payload.

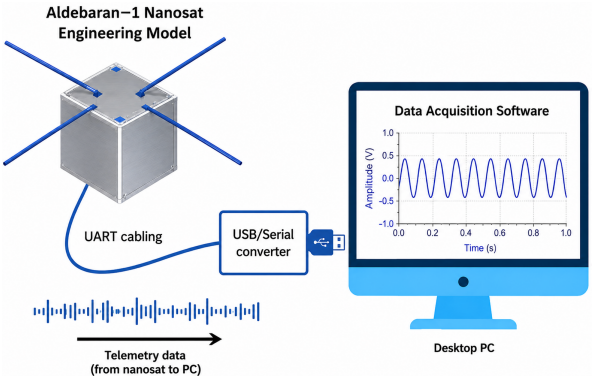


Figure 3. Telemetry transmission is performed using the UART interface.

The Aldebaran-1 project was designed to operate as a data receiver from collection platforms, ensuring efficient integration of the acquired information. The engineering model used in the experiments was developed to collect and transmit telemetry data through its UART interface. The transmitted data is captured by a UART-to-USB Serial converter and displayed on a personal computer for analysis. The received data is then processed by acquisition software and subsequently used in compression tests.

4.2. Benchmark Experimental

To evaluate the performance of the On+ algorithm, a benchmark was conducted using telemetry data from the Aldebaran-1 satellite, comparing it with the Huffman method and arithmetic coding. The objective was to analyze the compression efficiency using the following metrics: average compression rate (%), and sample standard deviation (%). Additionally, the maximum and minimum compression rates (%) were obtained.

Table 1 presents the results of the lossless compression evaluation for the classical Huffman and Arithmetic methods, as well as the commercial compressors .rar, .zip, .7z, .xz,

and .gz. It is observed that the .rar, .zip, and .7z methods resulted in file size expansion (negative rates). The On+ algorithm achieved an average compression rate of 29.19%, with a standard deviation of 1.26 and a median of 29.09%.

Table 1. Statistical summary of compression performance (%).

Statistic	On+	Huffman	Arithmetic	RAR	ZIP	7Z	XZ	GZ
Mean	29.19	18.23	8.11	−6.28	−64.32	−69.72	20.77	26.30
Standard Deviation	1.26	3.83	0.32	1.90	2.40	2.98	1.76	1.76
Median	29.09	17.45	7.93	−6.10	−64.63	−70.73	20.00	25.60

Figure 4 is a boxplot comparing the compression rate of the evaluated traditional lossless methods. It is observed that the minimum value of the On+ method exceeds the non-outlier maximum of the Huffman and Arithmetic methods, i.e., even the worst case of On+ is superior to the best typical case of the competitors. Additionally, On+ presents a shorter box (smaller interquartile range) and a higher mean (and median), indicating greater consistency and better average compression performance. On the other hand, Huffman exhibits upper outliers that reach exceptionally high compression rates, occasionally surpassing On+. These events, however, are rare and constitute exceptions to the general rule, not compromising the superiority of On+ in typical scenarios.

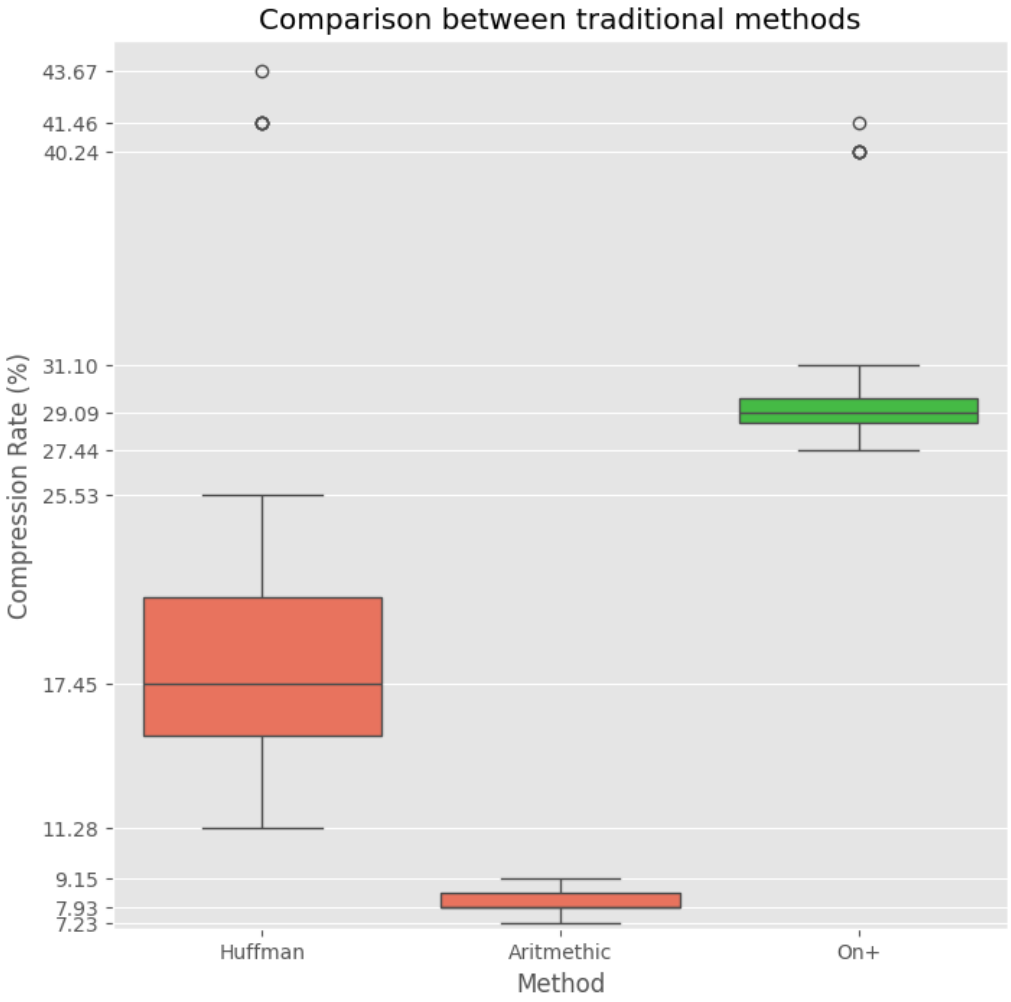


Figure 4. Performance graph of the Huffman, Arithmetic, and On+ algorithm methods, using data from the Aldebaran-1 satellite.

Figure 5 presents a boxplot with the compression rate results of the On+, .xz, and .gz methods, which were the only ones to show effective compression. Although the .xz and .gz methods achieved superior results in some specific files, these results were not statistically representative.

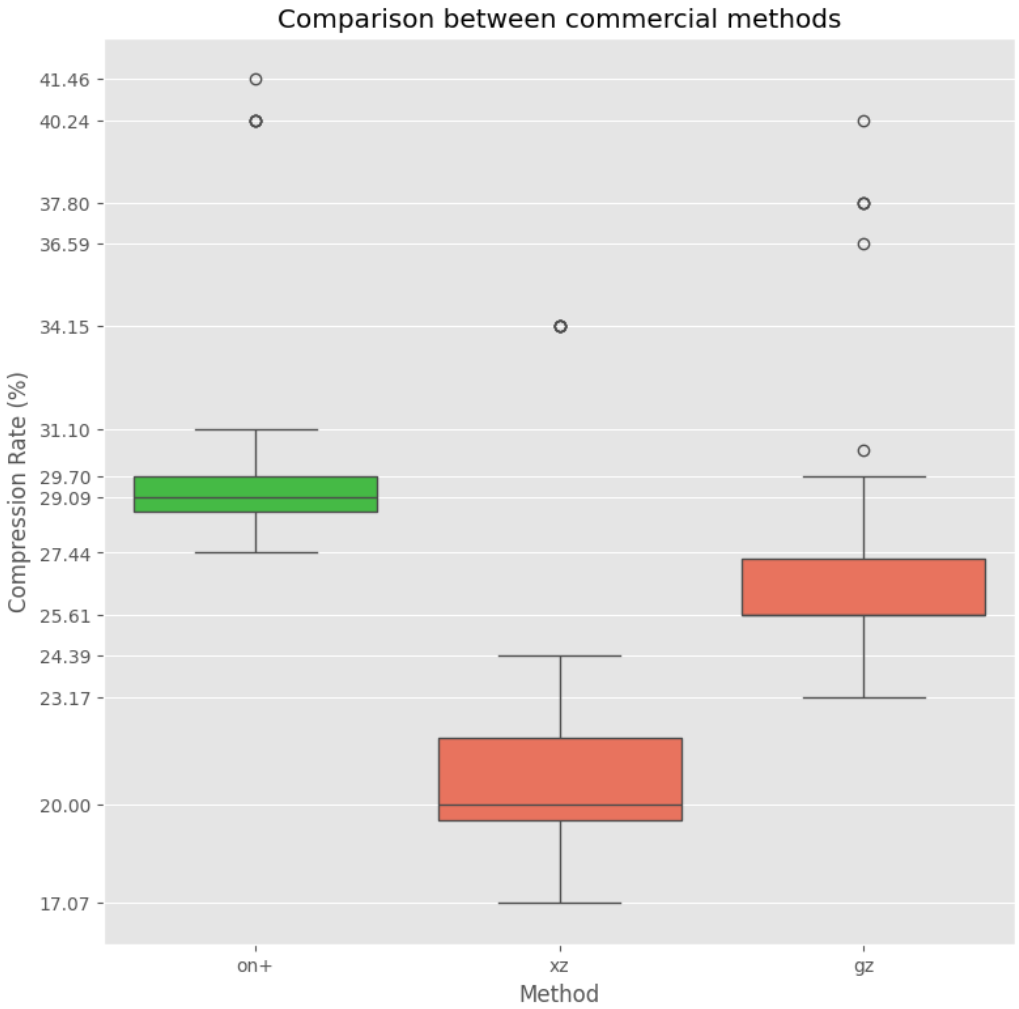


Figure 5. Performance graph of the On+ method and commercial compressors that did not expand.

4.3. Analysis of Entropy and File Compression

We analyzed the results obtained by the On+ algorithm, exploring its relationships with various factors, such as the maximum and minimum compression rates, the average compression rate, and the entropy calculation. Additionally, we present the behavior of the On+ algorithm for different values of p (probability of occurrence of bit 1).

Table 2 presents the compression rates that quantify the effectiveness of the algorithms in reducing data size, using 10 randomly selected files as an example. The results reveal significant variations between extreme cases, with values ranging from a minimum of 27.44% to a maximum of 41.46% (including outliers).

Figure 6 presents the results of the On+ algorithm's behavior for different values of p (the probability of occurrence of bit 1). Binary entropy quantifies the degree of uncertainty in the binary vector X, while the probability p directly influences the efficiency of the On+ algorithm.

Table 2. Comparison of size, compression ratio, and entropy of original and compressed files using the On+ algorithm, based on 10 randomly selected files.

Description	Original Size (kB)	On+ (kB)	On+ Cr. (%)	Original Entropy	Entropy On+
File 1	0.164	0.096	41.46	0.875	1.000
File 2	0.164	0.098	40.24	0.876	1.000
File 3	0.164	0.098	40.24	0.875	0.999
File 4	0.164	0.098	40.24	0.876	0.999
File 5	0.164	0.098	40.24	0.877	1.000
File 6	0.165	0.114	30.91	0.887	0.998
File 7	0.164	0.118	28.05	0.891	1.000
File 8	0.164	0.119	27.44	0.900	1.000
File 9	0.164	0.119	27.44	0.894	0.997
File 10	0.164	0.119	27.44	0.892	0.999

Legend: kB, kBytes; On+ Cr., On+ Compression ratio.

It is observed that, in the limiting condition where $\frac{L_Y}{L_X} = 1$, compression is ineffective or nonexistent, meaning that the average length of the compressed sequence (L_Y) is not shorter than the average length of the original sequence (L_X).

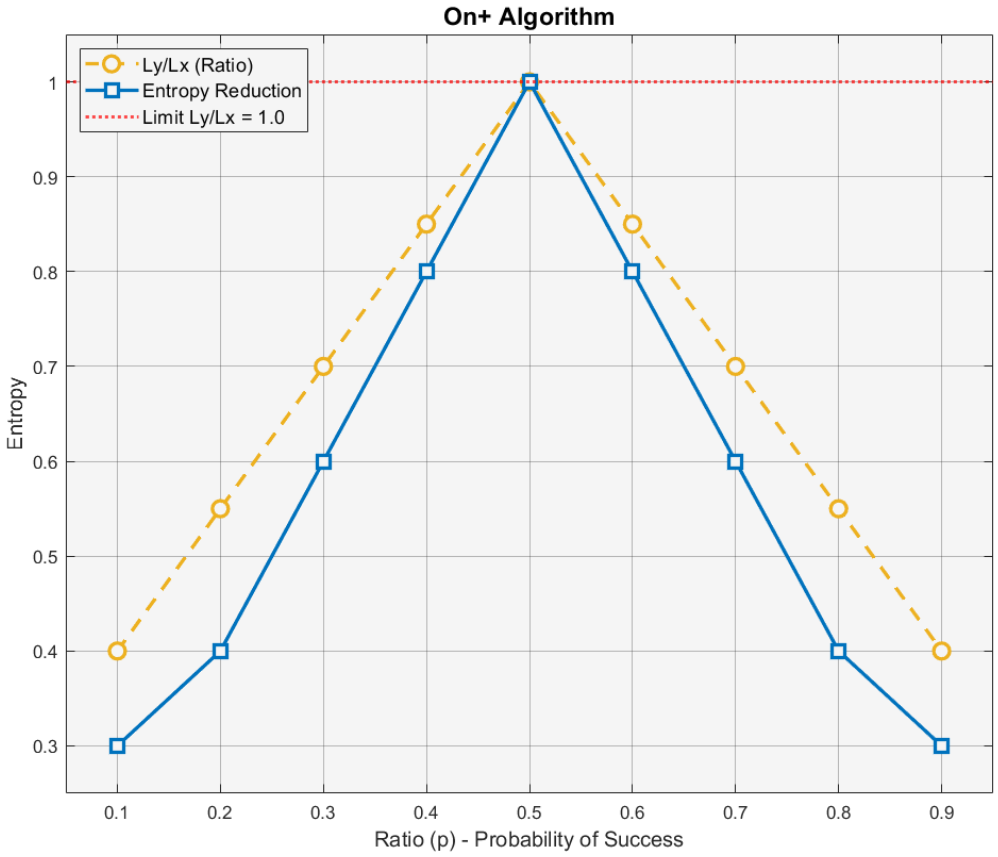


Figure 6. Relationship between entropy, the ratio L_y/L_X, and the success probability (p).

4.4. Time and Space Complexity Analysis

We evaluate the time and space complexities of the proposed encoding and decoding algorithms, comparing them with traditional Adaptive Arithmetic and Huffman compression methods, highlighting their efficiencies and limitations. This analysis allows for a detailed understanding of the required computational resources, contributing to the assessment of the algorithm's efficiency in practical satellite telemetry applications.

Table 3 presents a comparative summary of the complexity analysis of the lossless compression algorithms investigated for satellite telemetry data.

Table 3. Comparative Summary of Complexity Analysis for On+, Arithmetic, and Huffman Compression Algorithms.

Algorithm	Encoding Time	Encoding Space	Decoding Time	Decoding Space
On+ encoding	n	n	-	-
On+ decoding	-	-	n	n
Arithmetic	n	n	n	n
Huffman	$n\ logn$	n	n	n

Legend: For simplicity, we are using n instead of $O(n)$ to represent time and space complexity.

The analysis of the time complexity of the encoding process, presented in Algorithm 1, considered variable initialization, conditional instructions, the main iteration loop, and the adaptive arithmetic coding step. Initially, the algorithm traverses the input binary vector X only once to count consecutive zero sequences and build the vector $X_{geometric}$. Since each element of the vector is processed exactly once and the operations performed within the loop have constant cost, the preprocessing step has linear complexity.

After constructing the vector $X_{geometric}$, the symbols are encoded by the *AdaptiveArithmetic* function, which also processes the data sequentially. Thus, the total time complexity of the encoding algorithm grows linearly with respect to the size of the input vector n, resulting in an asymptotic complexity of $O(n)$.

The analysis of the space complexity of Algorithm 1 was performed considering local variables, the auxiliary vector $X_{geometric}$, and the output vector *encoded*. The largest memory consumption is associated with storing these vectors, whose sizes are proportional to the input size. Since no additional data structures with superlinear growth are used, the total space complexity of the encoding algorithm is $O(n)$.

The analysis of the time complexity of the decoding process, described in Algorithm 2, considered variable initialization, reading the *encoded* vector, executing the *AdaptiveArithmetic(encoded)* function, and the main loop responsible for reconstructing the original binary sequence. Initially, the arithmetic decoding function generates the vector $X_{geometric}$, which contains the lengths of the encoded sequences. Then, the algorithm traverses each element of this vector to reconstruct the binary vector *decoded*.

During each iteration of the main loop, the algorithm adds $(X_{geometric}(i) - 1)$ zeros to the *decoded* vector and subsequently appends a bit equal to 1. Although the number of zeros inserted varies with each iteration, the total number of bits added to the reconstructed vector is proportional to the final size of the original binary sequence. After the process ends, the last bit is removed to correctly restore the original data.

Thus, both the *AdaptiveArithmetic* function and the bit insertion operations into the *decoded* vector perform operations proportional to the total number of processed elements. Consequently, the total time complexity of the decoding algorithm is also linear, given by $O(n)$, where n represents the size of the processed sequence.

The analysis of the space complexity of Algorithm 2 considered local variables, the intermediate vector $X_{geometric}$ generated by the *AdaptiveArithmetic(encoded)* function, and the *decoded* vector, responsible for reconstructing the original binary sequence. Memory consumption is dominated by the storage of these vectors, while operations such as reading the encoded vector, calculating the sequence size, and removing the last bit do not require significant additional memory.

During the reconstruction process, the *decoded* vector is dynamically expanded through the insertion of zeros and bits equal to 1, according to the values stored in $X_{geometric}$. Since the final size of the reconstructed vector grows proportionally to the size of the processed input, the total memory consumption also exhibits linear growth. Therefore, the total space complexity of the decoding algorithm is $O(n)$.

Arithmetic compression [21] has a linear time complexity for encoding relative to the input size, with a complexity of $O(n)$, meaning that the algorithm processes the input in proportion to its size. The encoding space complexity is also $O(n)$, indicating that the amount of memory required is proportional to the input size and the compressed code. Regarding decoding, the time required is also linear, or $O(n)$, reflecting that the decoding is performed in a time proportional to the size of the compressed input. Similarly, the decoding space complexity is $O(n)$, implying that the memory required to store the compressed input and the decoded vector is proportional to the size of the compressed input.

Huffman compression [18] has a time complexity of $O(n \log n)$ for encoding due to the need to construct a Huffman tree, which involves a logarithmic cost in building the code. This process is more complex compared with algorithms with linear time complexity. In terms of encoding space, the Huffman method has a complexity of $O(n)$, corresponding to the storage of the code table and the compressed input. For decoding, the time complexity is linear, or $O(n)$, since the Huffman tree is used to efficiently decompress the data. Finally, the decoding space complexity is also $O(n)$, required to store the Huffman tree and the decoded input.

5. Discussion

The choice of a compression algorithm for satellite telemetry data is essential due to the stringent requirements for processing and storage efficiency, especially in environments with limited resources and the need for high-speed data communication. Our objective was to discuss the implications of choosing among the On+ coding algorithm, arithmetic coding, and Huffman coding, taking into account the specific context of satellite telemetry data.

Table 2 shows that the entropy of the compressed files approaches 1.000, a value close to the ideal theoretical limit, where each symbol carries approximately 1 bit of information. This result confirms the efficiency of the On+ algorithm, which eliminates redundancies and produces a uniform distribution of bits, demonstrating successful and near-ideal compression.

Figures 4 and 5 show that the On+ algorithm achieved a median rate of 29.09%, surpassing the Huffman method (17.45%), Arithmetic coding (7.93%), and the commercial compressors .xz and .gz. The On+ method presented a maximum compression rate of 31.10% (excluding outliers) and a minimum rate of 27.44%. Even this minimum rate is higher than the typical maximum rate observed for Huffman methods of 25.48% (excluding outliers) and arithmetic coding of 9.15%, confirming the ability of On+ to exploit patterns in highly structured data. The performance gain of the On+ algorithm over arithmetic and Huffman algorithms can be attributed to the transformation of the input probability distribution into a geometric distribution. The latter exhibits an asymmetry that can be effectively exploited by entropy encoders.

Regarding Figure 6, it can be observed that the entropy varies as a function of p. High values of p produce a geometric distribution with a short tail, while low values of p produce a distribution with a long tail. In both cases, the resulting geometric distribution exhibits asymmetry, a characteristic that is advantageous for entropy coders. When p approaches 0.5, the geometric distribution tends toward a uniform distribution, which is disadvantageous for entropy coders such as Huffman and arithmetic coding.

Table 2 shows that the tendency of the entropy of the compressed files to approach (1.000) suggests that the algorithm is operating efficiently to maximize compression. However, the success of the compression depends directly on the structure of the original data, since highly random patterns reduce the effectiveness of the encoding.

Table 3 shows that the On+ algorithm offers an efficient solution, with linear complexity $O(n)$ for both encoding and decoding. This algorithm is particularly advantageous in telemetry scenarios, where simplicity and speed are crucial. In space environments, characterized by limited bandwidth and processing resources, the temporal and spatial efficiency of On+ represents a significant advantage. The linear complexity ensures that the compression and decompression processes do not overload the satellite's resources, consolidating On+ as a practical and effective solution.

6. Conclusions

Satellites collect a wide range of data, such as temperature, pressure, battery level, and position. Due to limited onboard resources, optimizing communication efficiently is essential. In this study, we demonstrate the feasibility of applying geometric transformation compression to satellite telemetry data, highlighting its compression rate performance compared with traditional (Huffman and Arithmetic) and commercial (rar, zip, 7z, xz, and gzip) methods.

The comparative analysis revealed that, although each algorithm has its own advantages, the On+ algorithm stood out in terms of compression efficiency under the telemetry conditions of the Aldebaran-1 satellite, based on LoRa modulation, providing a significant reduction in data packet size.

The On+ method achieved compression in all 600 files extracted from the Aldebaran-1 satellite, with a maximum compression rate of 31.10% (excluding outliers), a minimum rate of 27.44%, and a median of 29.09%. The results show that the On+ method is superior to the Huffman and Arithmetic methods both in terms of average compression rate and consistency. It was observed that, in the worst cases, On+ achieves a compression rate higher than the best cases of the alternative methods analyzed, except for outliers. The On+ method exhibited a low standard deviation compared with the others, with the exception of the Arithmetic method, revealing the good consistency of the proposed method.

As perspectives for future work, we plan to deploy the On+ algorithm onboard the Aldebaran-1 satellite in future missions, as a compression step preceding the transmission of telemetry data. We intend to improve the On+ method by investigating the feasibility of prediction techniques and context modeling, with an emphasis on exploiting similarity between consecutive packets [4]. We also plan to conduct experiments with larger data packets and from different satellites to verify their impact on compression performance. This evaluation will strengthen our experimental analysis and increase the robustness and generalizability of the results.

The results of this study indicate that the On+ algorithm is a promising solution for lossless compression of telemetry data, with applications in satellites, spacecraft, and data collection devices in the Internet of Things (IoT).

Author Contributions: Conceptualization, L.C. and C.D.; methodology, F.B.; algorithm, F.B.; validation, F.B. and L.C.; formal analysis, F.B., G.S. and C.M.; investigation, F.B., G.S. and L.C.; resources, L.C.S. and C.D.; writing—original draft preparation, F.B.; writing—review and editing, C.M., L.C.S. and A.K.B.; visualization, F.B., L.C. and A.K.B.; supervision, A.K.B.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the conclusions of this article are available upon request from the corresponding author.

Acknowledgments: The authors thank the reviewers for their valuable suggestions. The visual representations of the data (graphs), as well as the implementation and tests included in this study, were performed using MATLAB R2020a software under a license provided by the Federal University of Maranhão. In addition, the platforms DeepSeek-V4, Grok 4.20 Beta 2, and GPT-5.3 Instant were used as linguistic adapters for refining technical terms, idiomatic expressions, and providing alternative versions of ambiguous texts, with the objective of preserving the original meaning and content.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CF	Compression Factor: ratio between the size of the original file and the compressed file
CR	Compression Rate
Downlink	Data download from the satellite to the ground station
IoT	Internet of Things
kB	kBytes
LoRa	Long Range
NASA	National Aeronautics and Space Administration
On+	Lossless Compression Algorithm
$O(n)$	Complexity Notation: linear time
TR	Transfer Rate: amount of data processed per unit of time
UART	Universal Asynchronous Receiver/Transmitter
Uplink	Data upload from the ground station to the satellite

References

1. Guojun , L.; Jian, S.; Running, Z. Lossless Data Compression Algorithm for Satellite Packet Telemetry Data. In Proceedings of the 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC), Shenyang, China, 20–22 December 2013; pp. 2756–2759.
2. Ketshabetswe, L.K.; Zungeru, A.M.; Lebekwe, C.K.; Mtengi, B. Energy-Efficient Algorithms for Lossless Data Compression Schemes in Wireless Sensor Networks. *Sci. Afr.* **2024**, *23*, e02008. [CrossRef]
3. Ciaparrone, G.; Benedetto, V.; Gissi, F. A Preliminary Study on AI for Telemetry Data Compression. In Proceedings of the International Conference on Deep Learning, Artificial Intelligence and Robotics, Salerno, Italy, 12–19 December 2022; pp. 134–143.
4. Rakhmanov, A.; Wiseman, Y. Compression of GNSS data with the aim of speeding up communication to autonomous vehicles. *Remote Sens.* **2023**, *15*, 2165. [CrossRef]
5. Meß, J.G.; Schmidt, R.; Fey, G. Adaptive Compression Schemes for Housekeeping Data. In Proceedings of the 2017 IEEE Aerospace Conference, Big Sky, MT, USA, 4–11 March 2017; pp. 1–12.
6. Evans, D.; Labrèche, G.; Marszk, D.; Bammens, S.; Hernández-Cabronero, M.; Zelenevskiy, V.; Shiradhonkar, V.; Starcik, M.; Henkel, M. Implementing the New CCSDS 124.0-B-1 Housekeeping Data Compression Standard (Based on POCKET+) on OPS-SAT-1. In Proceedings of the Small Satellite Conference, Logan, UT, USA, 6–11 August 2022.
7. Anmireddy, V.; Vasudevan, R.; Anand, D.; Rao, T.V.; Kapardhi, B.V.N.; Trivedi, D.; Manchanda, R.K. Telemetry, Telecommand and Safety Sub-systems for Scientific Ballooning from Hyderabad. *Adv. Space Res.* **2010**, *46*, 960–967. [CrossRef]
8. Meß, J.G.; Schmidt, R.; Fey, G.; Dannemann, F. On the Compression of Spacecraft Housekeeping Data Using Discrete Cosine Transforms. In Proceedings of the 2016 International Workshop on Tracking, Telemetry and Command Systems for Space Applications (TTC), Noordwijk, The Netherlands, 13–16 September 2016; pp. 1–8.
9. Xu, Z.; Cheng, Z.; Tang, Q.; Guo, B. An Encoder-Decoder Generative Adversarial Network-Based Anomaly Detection Approach for Satellite Telemetry Data. *Acta Astronaut.* **2023**, *213*, 547–558. [CrossRef]
10. Xu, Z.; Cheng, Z.; Guo, B. A Hybrid Data-Driven Framework for Satellite Telemetry Data Anomaly Detection. *Acta Astronaut.* **2023**, *205*, 281–294. [CrossRef]

11. Wan, P.; Zhan, Y.; Jiang, W. Study on the Satellite Telemetry Data Classification Based on Self-Learning. *IEEE Access* **2019**, *8*, 2656–2669. [CrossRef]
12. Shehab, A.F.; Elshafey, M.A.; Mahmoud, T.A. Recurrent Neural Network Based Prediction to Enhance Satellite Telemetry Compression. In Proceedings of the 2020 IEEE Aerospace Conference, Big Sky, MT, USA, 7–14 March 2020; pp. 1–11.
13. Hernández-Cabronero, M.; Evans, D.; Bartrina-Rapesta, J.; Aulí-Llinàs, F.; Blanes, I.; Serra-Sagristà, J. Resiliency and Efficiency of the CCSDS 124.0-B-1 Telemetry Compression Standard. *IEEE Access* **2024**, *12*, 36702–36711. [CrossRef]
14. Hao, Y.; Zhang, C.; Chai, S.; Li, Z.; Liu, X. Satellite Fault Detection and Diagnosis Based on Data Compression and Improved Decision Tree. In Proceedings of the 2020 Chinese Automation Congress (CAC), Shanghai, China, 6–8 November 2020; pp. 1686–1691.
15. Jeong, S.; Jeong, S.; Woo, S.S.; Ko, J.H. An Overhead-Free Region-Based JPEG Framework for Task-Driven Image Compression. *Pattern Recognit. Lett.* **2023**, *165*, 1–8. [CrossRef]
16. Wang, Z.; Wen, M.; Xu, Y.; Zhou, Y.; Wang, J.H.; Zhang, L. Communication Compression Techniques in Distributed Deep Learning: A Survey. *J. Syst. Archit.* **2023**, *142*, 102927. [CrossRef]
17. Sayood, K. *Introduction to Data Compression*; Morgan Kaufmann: Burlington, MA, USA, 2018.
18. Tian, J.; Rivera, C.; Di, S.; Chen, J.; Liang, X.; Tao, D.; Cappello, F. Revisiting Huffman Coding: Toward Extreme Performance on Modern GPU Architectures. In Proceedings of the 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Portland, OR, USA, 17–21 May 2021; pp. 881–891.
19. Wiseman, Y. High-Speed Architecture for Hybrid Arithmetic–Huffman Data Compression. *Technologies* **2025**, *13*, 585. [CrossRef]
20. Langdon, G.G. An Introduction to Arithmetic Coding. *IBM J. Res. Dev.* **1984**, *28*, 135–149. [CrossRef]
21. Said, A. Introduction to Arithmetic Coding–Theory and Practice. *arXiv* **2023**, arXiv:2302.00819.
22. Auli-Llinas, F. Fast and Efficient Entropy Coding Architectures for Massive Data Compression. *Technologies* **2023**, *11*, 132. [CrossRef]
23. Adel, M.; El-Naggar, M.; Darweesh, M.S.; Mostafa, H. Multiple Hybrid Compression Techniques for Electroencephalography Data. In Proceedings of the 2018 30th International Conference on Microelectronics (ICM), Sousse, Tunisia, 16–19 December 2018; pp. 124–127.
24. Ma, Z.; Zhu, H.; He, Z.; Lu, Y.; Song, F. Deep Lossless Compression Algorithm Based on Arithmetic Coding for Power Data. *Sensors* **2022**, *22*, 5331. [CrossRef] [PubMed]
25. Barros, A.K. Entropy as a Geometric Consequence of Higher Dimensions. *Technologies* **2025**, *13*, 563. [CrossRef]
26. Brian, K. Lossless Compression with Asymmetric Numeral Systems. 2020. Available online: https://bjlkeng.io/posts/lossless-compression-with-asymmetric-numeral-systems/ (accessed on 12 September 2025).
27. Khandwani, F.I.; Ajmire, P.E. A Survey of Lossless Image Compression Techniques. *Int. J. Electr. Electron. Comput. Sci. Eng.* **2018**, *5*, 39–42.
28. Blelloch, G.E. *Introduction to Data Compression*; Computer Science Department, Carnegie Mellon University: Pittsburgh, PA, USA, 2001; Volume 54.
29. Patel, H.; Itwala, U.; Rana, R.; Dangarwala, K. Survey of Lossless Data Compression Algorithms. *Int. J. Eng. Res. Technol.* **2015**, *4*, 926–929.
30. Cayoglu, U.; Tristram, F.; Meyer, J.; Schröter, J.; Kerzenmacher, T.; Braesicke, P.; Streit, A. Data Encoding in Lossless Prediction-Based Compression Algorithms. In Proceedings of the 2019 15th International Conference on eScience (eScience), San Diego, CA, USA, 24–27 September 2019; pp. 226–234.
31. Jayasankar, U.; Thirumal, V.; Ponnurangam, D. A Survey on Data Compression Techniques: From the Perspective of Data Quality, Coding Schemes, Data Type and Applications. *J. King Saud-Univ. Comput. Inf. Sci.* **2021**, *33*, 119–140. [CrossRef]
32. Phalke, S.; Vaidya, Y.; Metkar, S. Big-O Time Complexity Analysis of Algorithm. In Proceedings of the 2022 International Conference on Signal and Information Processing (IConSIP), Pune, India, 25–27 August 2022; pp. 1–5.
33. Vaz, R.; Shah, V.; Sawhney, A.; Deolekar, R. Automated Big-O Analysis of Algorithms. In Proceedings of the 2017 International Conference on Nascent Technologies in Engineering (ICNTE), Vashi, India, 27–28 January 2017; pp. 1–6.
34. Marques, J.; Santos, J.; Azulay, L.; Junior, C.; Filho, E.; Junior, J.; Silva, L. Thermal Simulation of a High Altitude Balloon. 2022. Available online: https://cubesat.ufsc.br/2022/cb.pdf (accessed on 21 November 2025).
35. Elzeiny, S.; Edward, P.; Elshabrawy, T. LoRa performance enhancement through list decoding technique. In Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.