LiDAR Data Compression in IoT and Drone Systems: A Comparative Study of Domain-Specific and General-Purpose Algorithms

Lucas Silva Souza, Rafaella L. Dias, Samuel B. Mafra, Hugerles S. Silva and Felipe A. P. de Figueiredo

***Abstract*— LiDAR systems generate large point cloud datasets, posing significant storage and transmission challenges in resource-constrained environments like IoT networks and drone-based systems. This study conducts a comparative analysis of lossless compression algorithms for LiDAR data in LAS format, evaluating their performance in balancing compression efficiency, speed, and data integrity. Using real-world airborne LiDAR datasets from Washington, DC, we test specialized (e.g., LASzip) and general-purpose (e.g., 7-Zip, BSC) algorithms. Results demonstrate that LASzip achieves the highest average compression ratio (84.67%), reducing files to 16.33% of their original size, while BSC excels in speed, compressing data 3-4 times faster than alternatives. Integrity validation via MD5 checksums and visual inspections confirms lossless preservation of point cloud structure and attributes. The findings highlight a clear trade-off: LASzip and NanoZip optimize storage efficiency, whereas BSC prioritizes rapid processing. This work provides actionable insights for selecting compression methods tailored to application needs, whether maximizing storage savings in environmental monitoring or enabling real-time performance in autonomous systems.**

***Keywords*— Data compression algorithms, LiDAR, IoT, Drones.**

I. Introduction

LiDAR (Light Detection and Ranging) remote sensing has established itself as one of the leading technologies for acquiring high-precision three-dimensional data in environmental, urban, and engineering studies. The technique is based on the emission of laser pulses and the measurement of their return time after striking surfaces, enabling the generation of point clouds that accurately represent the morphology of the terrain and objects within the area of interest. This level of detail has driven significant advancements in fields such as topographic mapping, forest inventory, and urban modeling. Beyond these traditional applications, LiDAR is increasingly integral to emerging technologies such as IoT-enabled environmental monitoring networks and drone-based systems. For instance, in IoT ecosystems, distributed LiDAR sensor arrays are deployed for real-time flood prediction, pollution tracking, and smart city infrastructure management, where granular spatial data informs adaptive urban planning [1]. Similarly, drone-mounted LiDAR systems revolutionize precision agriculture by enabling high-resolution crop health assessment, infrastructure inspection, and disaster response, where rapid 3D mapping enhances decision-making in dynamic environments [2], [3].

However, the high density of points collected in LiDAR surveys results in large volumes of data, often organized into multiple large files. Each recorded point stores detailed information, such as spatial coordinates, return intensity, and, in some cases, spectral attributes, substantially increasing the generated files' size. The fragmentation of these data into numerous files aims to facilitate processing and analysis but may also pose challenges related to storage, manipulation, and data integrity. These challenges are exacerbated in IoT and drone applications: energy-constrained IoT devices require efficient bandwidth utilization for real-time data transmission, while drones face limitations in onboard storage and computational capacity during prolonged missions. Furthermore, applications like drone-based obstacle avoidance or IoT-driven emergency response demand low-latency processing, where uncompressed data streams can hinder system performance [4].

In this context, it becomes crucial to discuss the benefits of LiDAR and the inherent challenges in managing the large volumes of data it produces, particularly in IoT and drone use cases. Understanding these aspects is essential for adopting effective storage, processing, and analysis strategies, thereby contributing to the optimized use of this technology across various research and application contexts [4], [5].

Given the substantial computational requirements associated with LiDAR data processing and the necessity of preserving an optimal level of detail, the application of data compression techniques must carefully account for the computational resources involved in both compression and decompression processes, while ensuring the preservation of data integrity throughout [3], [5]. For IoT networks, lightweight compression algorithms are critical to minimize energy consumption during data transmission, enabling edge devices to operate sustainably in remote or resource-limited settings. In drone systems, adaptive compression methods that balance resolution retention with file size reduction can enhance real-time navigation and mapping capabilities, ensuring efficient use of limited onboard storage and processing power [5], [6]. By addressing these domain-specific constraints, LiDAR data compression not only mitigates storage and transmission bottlenecks but also unlocks new possibilities for scalable, real-time applications across smart infrastructure, autonomous systems, and environmental

Lucas Silva Souza, Rafaella L. Dias, Samuel B. Mafra, and Felipe A. P. de Figueiredo are with INATEL, e-mail: lucas.souza@mtel.inatel.br, rafaella.dias@mtel.inatel.br, samuelbmafra@inatel.br, and felipe.figueiredo@inatel.br; Hugerles S. Silva is with UnB, email: hugerles.silva@unb.br. This work was supported by CNPq (311470/2021-1, 403827/2021-3, and 306199/2025-4), by the projects XGM-AFCCT-2024-2-5-1, XGM-AFCCT-2024-5-1-2, and XGM-AFCCT-2024-9-1-1 supported by xGMobile - EMBRAPII-Inatel Competence Center on 5G and 6G Networks, with financial resources from the PPI IoT/Manufatura 4.0 from MCTI (052/2023), by RNP, with resources from MCTIC (No. 01245.020548/2021-07), under the Brazil 6G project of the Radiocommunication Reference Center of INATEL, Brazil, SEMEAR Project supported by FAPESP (Grant No. 22/09319-9), SAMURAI Project supported by FAPESP (Grant 20/05127-2), and by FAPEMIG (APQ-04523-23, APQ-05305-23, and APQ-03162-24, APQ-01558-24, RED-00194-23) and xGMobile (Grant PPE-00124-23). This work has also been supported by master scholarships from FAPEMIG.

monitoring [7]–[10].

Therefore, we evaluate lossless compression methods for LiDAR point cloud data, addressing the challenges posed by large file sizes in resource-constrained environments. We compare specialized algorithms (e.g., LASzip) and general-purpose tools (e.g., 7-Zip, BSC) using real-world airborne LiDAR datasets from Washington, DC. Key contributions include identifying LASzip as the most efficient in achieving an average compression ratio of 84.67%, significantly reducing storage needs while preserving data integrity. The study also highlights BSC as the fastest algorithm, compressing files 3-4 times quicker than other methods, albeit with a moderate compression ratio. Through integrity validation using MD5 checksums and visual inspections, we confirm the lossless nature of these techniques. Their analysis provides actionable insights for selecting algorithms based on application priorities, optimal storage (LASzip/NanoZip) versus rapid processing (BSC), and advancing efficient LiDAR data management in domains like environmental monitoring and autonomous systems.

This article is organized into sections: Section I provides an overview of the work; Section II presents the motivations that guided the research. The methodology and the experiments conducted are described in Section III, while the results are detailed in Section IV. Finally, Section V discusses the findings and the opportunities opened by this research.

II. Related Work

The work in [2] presents a comparative analysis between specialized and general-purpose algorithms for lossless compression of airborne LiDAR data. The study evaluates three LiDAR-specific compression algorithms (LASzip, LASComp, and LiDAR Compressor) and three general-purpose compression algorithms (7-Zip, WinZip, and WinRAR), using real LiDAR point cloud datasets from the city of Washington, DC.

The algorithms were compared in terms of compression ratio, compression time, and bits per point. The results indicate that LASzip, a LiDAR-specific algorithm, achieved the best overall performance, with an average compression ratio of 16.63% of the original file size and an average compression time of 16.65 seconds. Among the general-purpose algorithms, WinRAR achieved better results than LiDAR Compressor in terms of compression ratio (20.24% of the original file size). However, the LiDAR-specific algorithms outperformed the general-purpose ones regarding compression efficiency.

In [4], the authors present a framework for compressing and transmitting LiDAR-generated point clouds in automotive scenarios, using semantic segmentation to guide the process. The point cloud is divided into semantically coherent groups, which are individually compressed with lossy algorithms and transmitted with parameters dynamically adjusted according to each group's relevance to the application. The proposed CACTUS method prioritizes maintaining the semantic integrity of critical data segments, ensuring that essential information is preserved despite compression losses.

In [5], the authors propose a deep learning-based approach for compressing LiDAR-generated point clouds, employing a convolutional autoencoder that processes raw data directly, without relying on discretization structures. The encoder extracts compact local feature descriptors to form a latent representation, while the decoder reconstructs dense geometry using a novel deconvolution technique, reducing quantization errors and memory overhead. Experimental results show that the method surpasses traditional compression techniques in reconstruction quality at comparable bit rates, achieving high compression ratios and efficient data handling in large-scale outdoor scenarios.

III. Methodology

Several steps were followed to conduct the comparative study, as shown in Figure 1. Initially, the data were standardized to ensure consistency and comparability of results. Subsequently, the selected compression algorithms were applied to the standardized files. The data collected during the compression process and the characteristics of the compressed files were analyzed to establish performance metrics. Finally, the files with the best compression results were decompressed, and their integrity was verified by comparing them to the original file.

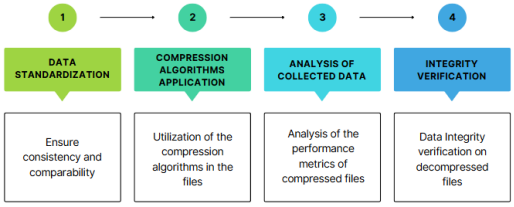


Fig. 1: Methodology Flowchart.

A. Dataset

To perform the compressions, data from real point cloud datasets obtained by Airborne LiDAR systems over the city of Washington, DC (District of Columbia) were used [6]. Each dataset represents a LiDAR point cloud, that is, a collection of millions of three-dimensional points (x, y, z) representing the Earth's surface, buildings, vegetation, and other objects captured during the flight of an airborne LiDAR system [6].

The data are in the standard LAS format of the ASPRS (American Society for Photogrammetry and Remote Sensing). Each point typically includes, in addition to the spatial coordinates, attributes such as laser return intensity, object classification (ground, vegetation, building, etc.), pulse number, timestamp, among other metadata relevant for applications in remote sensing, 3D modeling, Digital Elevation Model (DEM) generation, among other uses [6]. Table I shows each dataset's file size in bytes.

TABLE I: Group of LiDAR Data for Testing

	File name	File Size (Bytes)
1	1120.las	1.236.304.420
2	1121.las	1.087.954.270
3	1122.las	1.101.359.710
4	1317.las	1.106.791.720
5	1318.las	849.407.590
6	1319.las	1.006.267.960

B. Data Compression Algorithms

Data compression reduces file size by eliminating redundancy through statistical or dictionary-based methods, optimizing storage and transmission efficiency. Techniques are lossy (irreversible data removal, yielding approximate reconstructions) or lossless (exact reconstruction via redundancy reduction). For LiDAR applications, lossless compression is critical to preserve the geometric and radiometric precision of point cloud data in '.las' files, which contain dense spatial coordinates, intensity values, and metadata [2].

ZIP and GZIP are foundational lossless compression formats widely used for general-purpose data. ZIP employs the DEFLATE algorithm, which combines LZ77 (a dictionary-based method for detecting repeated sequences) with Huffman coding (entropy encoding to assign shorter codes to frequent symbols). ZIP also supports file archiving, enabling multiple files to be bundled into a single compressed archive. GZIP, based on the same DEFLATE algorithm, is optimized for single-file compression and is commonly used in Unix-based systems and web servers for reducing HTTP payload sizes. While ZIP includes metadata for file structure, GZIP focuses on streaming efficiency, making it ideal for log files and network transmission.

Symbol Ranking Version 2 (sr2) is a context-modeling, lossless compressor optimized for speed over maximum compression ratios. It operates as a single-file compressor with a fixed 6 MB memory footprint. sr2 employs a hybrid context model, combining an order-4 context that tracks the last three observed bytes to predict the next symbol (maintaining a dynamic ranking of recently seen symbols) with an order-1 fallback model for symbols not predicted by the higher-order context. Both contexts feed into an adaptive arithmetic coder, which assigns shorter codes to higher-ranked symbols. While sr2 sacrifices some compression efficiency for speed, its lightweight design suits real-time applications or memory-constrained systems [7].

LASzip, a lossless, open-source compressor tailored for LiDAR data, was developed by Martin Isenburg to convert '.las' files to '.laz' with 80–93% size reduction while retaining full data integrity. It employs component-wise encoding to separate LiDAR attributes such as coordinates, intensity, and GPS time for targeted compression. For example, delta encoding reduces integer magnitude and entropy by storing differences between consecutive point coordinates, while run-length encoding (RLE) compresses repeated values like classification flags. Residual values are further compressed using range coding, a variant of arithmetic coding. LASzip also implements chunk-based parallelism, dividing data into blocks for multi-core processing. The resulting '.laz' files remain backward-compatible with LAS-supporting software via integrated decompression libraries, solidifying LASzip as the de facto standard for LiDAR compression [8].

NanoZip (NZ), an experimental high-ratio archiver by Sami Runsas, combines statistical modeling and dictionary-based techniques. It uses context mixing (CM) to blend predictions from diverse models, such as Burrows-Wheeler Transform (BWT) and LZ77 predictors (LZP), alongside LZ77 variants for detecting repeated byte sequences. Multi-threading splits files into chunks for parallel processing on modern CPUs, prioritizing compression efficiency over speed. This modular design allows users to select algorithms based on data type, making NanoZip suitable for archival purposes where maximum compression is prioritized [9], [10].

Block Sorting Compressor (bsc), developed by Ilya Grebnov, is a parallelized lossless compressor leveraging the Burrows-Wheeler Transform (BWT). The BWT rearranges data into reversible, entropy-optimized blocks, which are then processed by move-to-front (MTF) encoding to enhance Huffman or arithmetic coding efficiency. Independent blocks are compressed concurrently across multiple CPU cores, enabling linear performance scaling. While bsc achieves high compression ratios on repetitive data like genomic sequences or logs, its reliance on BWT requires significant memory for transformation tables [11].

7-Zip, developed by Igor Pavlov, is a versatile open-source tool supporting the LZMA (Lempel-Ziv-Markov chain Algorithm) and LZMA2 formats. LZMA combines a sliding-window LZ77 parser with a range coder, utilizing a 4 GB dictionary to detect long-distance redundancies. LZMA2 enhances this with multi-threaded chunk processing and improved handling of incompressible data. Additional preprocessing filters, such as BCJ (branch call jump), optimize the compression of executable files. While its graphical interface is Windows-exclusive, the command-line port ('p7zip') supports Linux and macOS. 7-Zip's balance of speed and compression ratio has made it a popular choice for general-purpose use [12].

C. Performance Metrics

Several characteristics were analyzed to evaluate each compression algorithm's performance, including compression time, resulting file size, and compression ratio, which is calculated according to Eq. (1).

$$\text{compression\_ratio} = \left(1 - \frac{\text{size\_after}}{\text{size\_before}}\right) \times 100. \quad (1)$$

The equation indicates that the compression ratio represents a comparison between the file size before and after applying a compression method. For example, a 1GB file compressed with a 65% compression ratio will result in a final size of approximately 350MB.

D. Integrity Analysis

An integrity validation procedure was conducted in two stages to ensure that the compression and subsequent decompression processes did not alter the original data. Initially, visual inspections were performed using point cloud visualization software to qualitatively assess any perceptible differences between the original and decompressed files. This step aimed to detect distortions or loss of data structure, particularly in spatial distribution and scalar field values.

Subsequently, MD5 checksum verifications were carried out on the original and decompressed files. This cryptographic hash function generates a unique fingerprint for each

TABLE II: File Size After Compression in Mega Bytes (MB)

File	7-Zip	Zip	GZip	LASzip	NZ	BSC	sr2
1120.las	294	421	421	194	224	384	388
1121.las	255	367	367	167	195	338	339
1122.las	254	366	366	166	194	341	341
1317.las	257	374	374	167	195	345	347
1318.las	201	290	290	133	157	265	265
1319.las	234	339	339	153	179	317	316

file, allowing for a precise binary-level comparison. Identical MD5 values indicate that the file content remains unchanged throughout the compression cycle, thus confirming the lossless nature of the process. Discrepancies in hash results were further analyzed to determine whether they stemmed from differences in compression strategies (e.g., streaming formats or metadata alterations) or actual loss of point cloud data.

IV. Results and Discussions

For the compression process and the evaluation of compression rates and processing times, the general-purpose algorithms WinZip, 7-Zip, and Gzip were used, through the software 7-Zip Manager, commonly known simply as 7-Zip [13] and a domain-specific compression algorithm, LASzip [14]. In addition, three lossless compression algorithms available on Matt Mahoney's "Large Text Compression Benchmark" page were used: Nanozip, BSC, and sr2.

The tests were conducted on a computer with the following specifications: Dell Inc. Inspiron 14 5440, equipped with 32GB of RAM, an Intel Core i7-150U (12 cores), and both Intel RPL-U integrated graphics and an NVIDIA GeForce MX570 A GPU. The operating system used was Ubuntu 24.04.2 LTS.

The results obtained from the six compression algorithms in terms of compression ratio (calculated based on Eq. (1)) on the test dataset are presented in Tables II and III.

The results obtained from the six update algorithms in terms of update time (on the test dataset) are presented in Table IV.

The algorithms exhibited distinct performances in terms of compression ratio and processing time. LASzip stood out with the highest efficiency in file size reduction, achieving an average compression ratio of 84.67%, surpassing all other methods. This result suggests that algorithms specialized for LiDAR data, such as LASzip, are more suitable for lossless compression in this context, as they leverage the specific structure of LAS files. Nanozip achieved the second-best compression ratio (82.08%), outperforming general-purpose algorithms such as 7-Zip (76.61%), Zip (66.24%), and GZip (66.23%).

TABLE III: Average Compression Ratio by Algorithm

	Algorithm	Average Compression Ratio
1	7-Zip	76,61%
2	Zip	66,24%
3	GZip	66,23%
4	Nanozip	82,08%
5	BSC	68,83%
6	LASzip	84,67%
7	sr2	68,75%

TABLE IV: Compression Time (in seconds)

File	7-Zip	Zip	GZip	LASzip	NZ	BSC	sr2
1120.las	60	90	93	34	72	23	67
1121.las	55	81	79	17	98	20	25
1122.las	87	83	80	15	35	21	27
1317.las	62	81	89	18	83	20	26
1318.las	56	61	60	25	30	15	19
1319.las	69	75	73	23	82	17	23
Average	**64,83**	**78,50**	**79,00**	**22**	**66,7**	**19,33**	**31**

Regarding compression time, BSC proved to be the fastest, with an average of 19.33 seconds—approximately 3.4 times faster than 7-Zip (64.83 seconds) and 4.1 times faster than Nanozip (66.7 seconds). However, this speed came with a moderate compression ratio (68.83%), lower than that of LASzip and Nanozip. The sr2 algorithm partially balanced these factors, with an average time of 31 seconds and a compression ratio of 68.75%, similar to BSC.

The file-level analysis reinforced these trends: for the file 1120.las, LASzip achieved a size of 185 MB (85% reduction), while 7-Zip reached 280 MB (76.6%). BSC compressed the same file to 366 MB (68.8%) in just 23 seconds, compared to 60 seconds for 7-Zip. These results indicate an inverse relationship between compression efficiency and speed, requiring choices based on application priorities—optimized storage (LASzip/Nanozip) or faster processing (BSC).

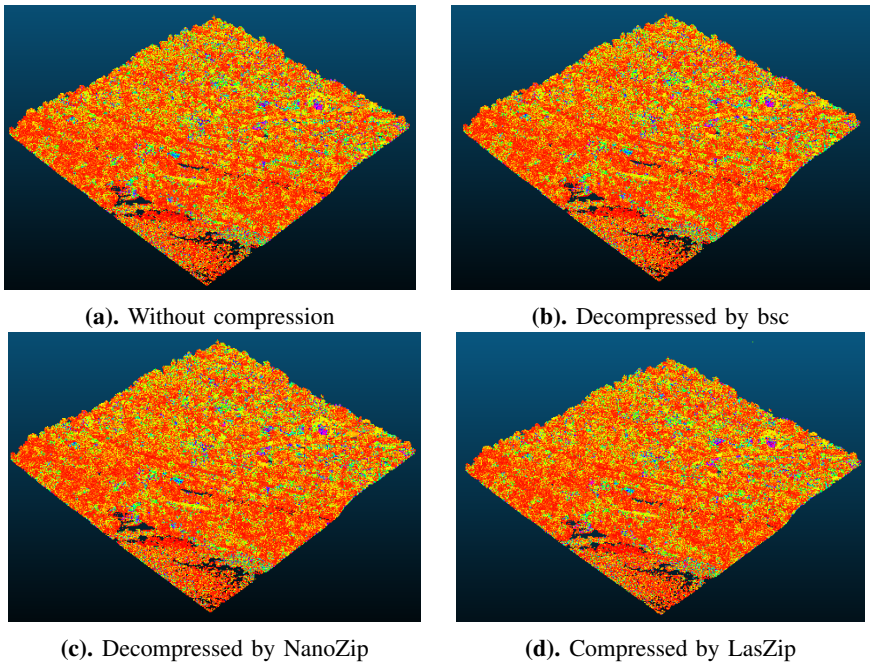
Figures 2a, 2b, 2c, and 2d show visualizations of the point cloud from file 1319.las: in its original form, after decompression using the BSC method, after decompression using NanoZip, and from the .laz file generated by LASzip, respectively. The visual similarity among the four point clouds and the preservation of scalar field values verified through visualization software indicate that none of the compression methods introduce perceptible alterations to the original data. Thus, it can be concluded that these compression and decompression processes preserve the integrity of the point cloud.

A comparison between the original 1319.las file and its decompressed versions from NanoZip and BSC were performed using an MD5 checksum verifier. Both methods returned identical checksums, confirming the preservation of the original file's data. A similar test was conducted with the .laz file generated by LASzip, which initially produced a different checksum. This discrepancy arises because .laz files are optimized for direct use without explicit decompression, resulting in a different binary structure despite representing the same point cloud. Notably, the original .las file and the .laz file contain the same number of points, totaling 33,540,692. However, when the .laz file is fully decompressed back into a .las format, the resulting checksum matches the original, confirming the lossless nature of LASzip compression.

A test involving the compression of a .las file to .laz using LASzip, alongside another compression method, yielded unsatisfactory results, as the average compression ratio was below 1%. However, the compression time remained low, averaging only a few seconds.

V. Conclusions

The comparative analysis of compression algorithms for LiDAR point cloud data underscores that the optimal method



(a). Without compression **(b).** Decompressed by bsc

(c). Decompressed by NanoZip **(d).** Compressed by LasZip

Fig. 2: Comparing the original and compressed versions of file 1319.las across different methods.

depends on the application's priorities. LASzip emerged as the most storage-efficient solution, achieving an average compression ratio of 84.67% (reducing files to 16.33% of their original size), making it ideal for scenarios where maximizing storage savings is critical, such as archival or bandwidth-constrained IoT and drone systems. In contrast, BSC excelled in processing speed, compressing data 3-4 times faster than alternatives (average: 19.33 seconds), positioning it as the preferred choice for real-time applications like drone-based mapping or autonomous navigation.

While LASzip and NanoZip outperformed general-purpose tools in efficiency, BSC's compatibility with Linux-based systems (e.g., Ubuntu, Raspberry Pi) extends its utility to embedded, drone, and IoT devices, enabling field-deployable solutions. Notably, the study confirmed the lossless integrity of all algorithms through MD5 checksums and visual inspections, ensuring fidelity for precision-critical applications like topographic modeling or environmental monitoring.

LASzip remains the top performer for compression ratio, while BSC leads in speed. NanoZip offers a balance but does not surpass LASzip in compression efficiency. General-purpose tools (e.g., 7-Zip) offer a middle ground but are less optimized for LiDAR's unique data structures.

Future work should explore machine learning-driven adaptive compression, such as autoencoder-based models, to dynamically prioritize semantically critical LiDAR data segments (e.g., urban infrastructure vs. vegetation) while maintaining acceptable fidelity in lossy contexts. Additionally, edge-optimized lightweight compression methods must be developed to address IoT devices' and drone systems' stringent energy and real-time processing constraints, ensuring efficient deployment in resource-limited scenarios.

In summary, this study provides actionable insights for selecting algorithms based on workflow needs, LASzip for storage optimization, BSC for speed, while highlighting open challenges for future research in scalable, intelligent compression systems.

REFERENCES

[1] NOAA Office for Coastal Management, "Lidar 101: An Introduction to Lidar Technology, Data, and Applications," disponível em: https://coast.noaa.gov/data/digitalcoast/pdf/lidar-101.pdf. Accessed on: May 3, 2025.

[2] A. Kotb, S. Hassan, e H. Hassan, "A comparative study among various algorithms for lossless airborne LiDAR data compression," *Proceedings of the 2018 14th International Computer Engineering Conference (ICENCO)*, pp. 17–21, 2018.

[3] G. Rivera, R. Porras, R. Florencia e J. P. Sánchez-Solís, "LiDAR applications in precision agriculture for cultivating crops: A review of recent advances," *Computers and Electronics in Agriculture*, vol. 207, p. 107737, 2023.

[4] D. Mari, E. Camuffo e S. Milani, "CACTUS: Content-Aware Compression and Transmission Using Semantics for Automotive LiDAR Data," Sensors, vol. 23, n. 12, art. 5611, 2023. DOI: 10.3390/s23125611. Disponível em: https://www.mdpi.com/1424-8220/23/12/5611. Accessed on: May 3, 2025.

[5] L. Wiesmann, A. Milioto, X. Chen, C. Stachniss e J. Behley, "Deep Compression for Dense Point Cloud Maps," IEEE Robotics and Automation Letters, vol. 6, n. 2, pp. 2060–2067, abr. 2021. DOI: 10.1109/LRA.2021.3059633.

[6] AWS Open Data, "District of Columbia 2015 LiDAR," disponível em: https://github.com/awslabs/open-data-docs/tree/main/docs/dc-lidar-2015. Accessed on: May 3, 2025.

[7] M. Mahoney, "Data Compression Programs," available at: https://www.mattmahoney.net/dc/#sr2. Accessed on: May 3, 2025.

[8] M. Isenburg, "LASzip: Lossless compression of LiDAR data," *Photogrammetric Engineering and Remote Sensing*, vol. 79, n. 2, pp. 209–217, 2013.

[9] M. Mahoney, "Large Text Compression Benchmark," disponível em: https://www.mattmahoney.net/dc/text.html#1493. Accessed on: May 3, 2025.

[10] D. Pukhkaiev, "Energy-efficient benchmarking for energy-efficient software," 2016.

[11] I. Grebnov, "libbsc: High Performance Block-Sorting Data Compression Library," disponível em: https://github.com/IlyaGrebnov/libbsc. Accessed on: May 3, 2025.

[12] 7-Zip, "7-Zip: Open-source file archiver with high compression ratio," disponível em: https://www.7-zip.org/. Accessed on: May 3, 2025.

[13] 7-Zip, "7-Zip official website," disponível em: https://www.7-zip.org/. Accessed on: May 3, 2025.

[14] Martin Isenburg, "LASzip: lossless compression of LiDAR data," disponível em: https://laszip.org/. Accessed on: May 3, 2025.