*big data and cognitive computing*

MDPI

*Article*

# BWT-Enhanced Compression for GIS Raster Data: A Hybrid AV1-Inspired Approach with Burrows–Wheeler Transform

**Yair Wiseman**

Computer Science Department, Bar-Ilan University, Ramat-Gan 5290002, Israel; wiseman@cs.biu.ac.il

**Abstract**

The AVIF (AV1 Image File Format) is a modern, royalty-free image format that leverages the AV1 video codec for superior compression efficiency, supporting both lossy and lossless modes. Its entropy encoding relies on a multi-symbol context-adaptive arithmetic coder (range coding with adaptive cumulative distribution functions (CDFs)), which is effective for general imagery but may not optimally exploit the repetitive structures common in Geographic Information System (GIS) maps/data. This paper proposes replacing AVIF's entropy encoder with the Burrows–Wheeler Transform (BWT), a reversible preprocessing algorithm that rearranges data to create runs of similar symbols, enhancing subsequent compression. We detail the technical steps for modification, drawing from AV1's open-source implementation, and explain why BWT is advantageous for GIS raster maps/data, which often feature large uniform areas, limited color palettes, and spatial redundancies. Empirical evidence from related studies on BWT-based image compression shows improvements in lossless scenarios, potentially considerably reducing file sizes over standard methods while preserving data integrity critical for geospatial analysis. This swap could improve storage, transmission, and processing efficiency in GIS applications, such as remote sensing and cartography. The discussion includes challenges like computational overhead and compatibility, with recommendations for implementations. The resulting BWT-AVIF hybrid produces a non-standard AV1 bit-stream that is not compliant with the AV1 or AVIF specifications and therefore requires custom decoders. It is presented here as a research prototype for GIS-specific compression rather than a compliant AVIF extension.

**Keywords:** AVIF; Burrows–Wheeler Transform; GIS raster data; geospatial data compression; thematic rasters; lossless raster compression



Academic Editors: Yibeltal Chanie Manie and Hayle Stotaw Talbachew

Received: 22 February 2026
Revised: 9 April 2026
Accepted: 18 April 2026
Published: 1 May 2026



## 1. Introduction

In the era of big data, efficient image compression is paramount, particularly for specialized domains like Geographic Information Systems (GIS), where raster data represent vast geospatial datasets. Traditional formats like JPEG or PNG often fall short in balancing file size, quality, and fidelity, especially for GIS raster data that require lossless compression to maintain analytical accuracy [1]. The AVIF format, introduced in 2019 as an open standard by the Alliance for Open Media (AOMedia), addresses these needs by encapsulating AV1-compressed images in a HEIF (High-Efficiency Image File Format) container [2]. AVIF excels in compression ratios, supporting high dynamic range (HDR), wide color gamuts, and alpha channels, making it suitable for web and multimedia applications [3].

At the core of AVIF's efficiency is the AV1 codec's pipeline: prediction, transform coding (e.g., discrete cosine transform variants), quantization, and entropy encoding. The

entropy stage uses a multi-symbol, context-adaptive arithmetic coder to encode symbols probabilistically, reducing redundancy [4]. While effective for natural images, this approach may not fully capitalize on the unique characteristics of GIS raster data, such as repetitive patterns in land cover (e.g., forests, water bodies) or thematic layers with limited discrete values. Arithmetic coding treats symbols based on local contexts but lacks global rearrangement to amplify redundancies.

Enter the Burrows–Wheeler Transform (BWT), a block-sorting algorithm originally developed for text compression in tools like bzip2. BWT rearranges input data by sorting all cyclic shifts, extracting the last column to group similar characters into runs. This transform is reversible and serves as a preprocessing step, making subsequent encoding (e.g., run length or Huffman) more efficient. Unlike arithmetic coding, which encodes symbols sequentially based on probabilities, BWT exploits global context, creating compressible structures from repetitive data [5].

This paper explores replacing AVIF's entropy encoder with BWT, focusing on its potential benefits for GIS map compression. GIS raster data, often stored as grids of pixel values representing elevation, vegetation indices, or land use, benefit from lossless methods to avoid data distortion that could affect analyses like change detection or modeling. Common GIS compression techniques include LZ77 (deflate), JPEG 2000, or quadtree partitioning, but BWT's ability to handle repetitions in non-text data like images offers untapped potential.

By making this substitution, the paper argues, we can achieve significantly more efficient compression for GIS raster data due to their spatial autocorrelation: adjacent pixels often share values, leading to long runs post-BWT. Studies on BWT for images report superior performance over standards like JPEG-LS in lossless scenarios.

This paper is organized to sequentially present the development and evaluation of our proposed compression approach. Section 2 provides an overview of related work, focusing specifically on current implementations of AVIF and the Burrows–Wheeler Transform (BWT). In addition, it explains the foundations of GIS and the Case for BWT Compression. Section 3 outlines the technical methodology, detailing the specific steps required to execute the proposed substitution. In Section 4, we explore the practical value of this approach, outlining its distinct advantages when applied to Geographic Information Systems (GIS) data. Section 5 provides an in-depth discussion and analysis of the experimental results. Finally, Section 6 synthesizes our findings and presents our concluding remarks.

## 2. Related Work

### *2.1. AVIF Image Standard: Background and Comparative Evaluation*

The AVIF image format represents a significant advancement in digital compression technology, primarily constructed upon the foundation of the AV1 video codec. This codec is meticulously detailed in the AV1 Bitstream & Decoding Process Specification, which outlines the precise mechanisms for encoding and decoding data streams. AV1 itself emerged as a collaborative effort by various tech giants to provide an open-source, royalty-free solution that could rival established standards like HEVC, also known as H.265 [6]. By focusing on superior efficiency for both video sequences and static images, AV1 enables AVIF to deliver high-quality visuals at reduced file sizes, making it particularly appealing for web-based applications where bandwidth conservation is crucial [7]. In the context of AVIF, individual images are treated as standalone intra-frames, essentially key frames that do not rely on temporal references from other frames, and these are encapsulated within HEIF containers [8]. These containers serve as versatile wrappers that accommodate essential metadata, such as color profiles, image orientation, and even alpha channels

for transparency, ensuring that the format remains flexible and compatible across diverse platforms and devices.

AV1's design philosophy centers on achieving exceptional compression ratios without imposing licensing fees, which has been a major drawback for predecessors like HEVC [9]. Developed under the Alliance for Open Media (AOMedia), AV1 incorporates a suite of innovative tools tailored for modern content delivery, including ultra-high-definition videos and intricate still photographs. Its royalty-free nature democratizes access to cutting-edge compression, allowing developers, content creators, and end-users to integrate it seamlessly into software ecosystems without financial barriers. For still images in AVIF, this translates to a format that not only preserves fine details and vibrant colors but also optimizes storage and transmission, proving especially beneficial in scenarios like mobile photography or online galleries where quick loading times enhance user experience [10]. The shift away from proprietary codecs like HEVC underscores a broader industry trend toward openness and collaboration, fostering innovation in areas such as virtual reality and augmented reality imaging, where AVIF's efficiency can support immersive experiences without overwhelming hardware resources [11].

Within the AVIF framework, the encoding of images as intra-frames in AV1 streams is a clever adaptation of video technology to static content, eliminating the need for inter-frame dependencies that are typical in video compression [12]. This approach ensures that each image can be decoded independently, which is vital for random access and editing workflows [13]. The HEIF container, derived from the ISO Base Media File Format, adds layers of sophistication by supporting features like image sequences, burst photos, and even computational photography metadata. For instance, the HEIF container can store information about high-dynamic-range (HDR) rendering or depth maps, enabling AVIF files to go beyond simple digital information to capture the true essence of the captured scene. This integration of AV1's core with HEIF's structure positions AVIF as a forward-looking format, poised to replace older standards in browsers, operating systems, and content management systems, where compatibility with emerging display technologies is paramount [14].

The initial stage of AVIF compression relies on intra-prediction, a process that utilizes spatial correlations within an image to estimate pixel values from adjacent ones. By predicting and subtracting these values from the actual data, the encoder minimizes redundancy that arises from patterns like smooth gradients or repetitive textures, thereby streamlining the subsequent stages of compression. This step is highly adaptive, employing multiple prediction modes such as directional, DC, or smooth predictions, in order to best match the local characteristics of the image block [15]. In practice, intra-prediction allows AVIF to handle diverse content types, from natural landscapes with subtle color transitions to synthetic graphics with sharp edges, all while maintaining perceptual quality. The efficiency gained here sets the tone for the entire pipeline, ensuring that the residual data passed to later stages is as compact as possible, which is particularly advantageous for high-resolution images where raw file sizes could otherwise become unwieldy.

Following intra-prediction, the residuals undergo transform coding, a critical phase where spatial domain data is converted into frequency domain coefficients using mathematical transforms. AV1 offers a variety of options here, including the Discrete Cosine Transform (DCT) for general-purpose compression, the Discrete Sine Transform (DST) for better handling of certain edge cases, and even identity transforms that bypass this step entirely for lossless scenarios [16]. These transforms concentrate energy into fewer coefficients, facilitating more effective quantization and encoding downstream. By supporting block sizes ranging from $4 \times 4$ to $128 \times 128$ pixels, AVIF can adapt to varying levels of detail within an image, applying larger transforms to uniform areas for greater efficiency

and smaller ones to intricate regions to preserve fidelity. This flexibility is key to AVIF's superiority over formats like JPEG, which rely on a more rigid $8 \times 8$ DCT, often resulting in artifacts like blocking or ringing in compressed outputs.

AVIF inherits AV1's recursive multi-type tree partitioning. The image starts with superblocks of $64 \times 64$ or $128 \times 128$ pixels. Each superblock is recursively split (quadtree + asymmetric binary/ternary splits) down to $4 \times 4$ coding blocks. Transform blocks (where the actual DCT/ADST/etc. is applied) are chosen per coding block and can also be further partitioned, supporting sizes from $4 \times 4$ up to $64 \times 64$ (square and rectangular).

The encoder applies a (possibly different) transform kernel to each chosen block size $N \times M$:

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} r(x,y) \cos\left(\frac{\pi(2x+1)u}{2N}\right) \cos\left(\frac{\pi(2y+1)v}{2M}\right) \quad (1)$$

(AV1 actually supports DCT, ADST, flipped ADST, and identity transforms; the formula above is the DCT case. $r(x,y)$ is the prediction residual.)

There are distinct mathematical benefits associated with the different values of variable $N$. First, energy compaction is highly effective in uniform and smooth areas, meaning that larger transforms yield higher coding efficiency. For a perfectly constant residual block ($r(x,y) = c$), only the DC coefficient is non-zero:

$$C(0,0) = c \cdot \sqrt{NM}, C(u,v) = 0 \text{ for all } (u,v) \neq (0,0) \quad (2)$$

A single large block (e.g., $64 \times 64$) needs one DC coefficient + one set of mode/partition bits. The same area coded as 64 separate $8 \times 8$ blocks (JPEG style) needs 64 DCs + 64× more signaling overhead. Even for slowly varying smooth gradients, larger $N$ gives finer low-frequency resolution (frequency bin spacing $\approx \frac{1}{2N}$ cycles/pixel), so energy stays packed into fewer low (w, v) coefficients. The generation of significantly more zero-valued coefficients after quantization yields a lower coding bitrate for a given level of distortion.

In addition, superior spatial localization in intricate and detailed regions dictates that smaller transforms yield higher reconstruction fidelity. A sharp edge or texture confined to a $4 \times 4$ region would force a large block to spend many high-frequency coefficients (or accept ringing across $64 \times 64$ pixels). A small $4 \times 4$ transform confines the high-frequency energy (and any ringing) to just those 16 pixels. The rest of the image can stay on large blocks.

Furthermore, Rate-Distortion Optimization (RDO) functions as the core decision engine, dynamically evaluating and selecting the most efficient coding parameters. The encoder evaluates every possible partition tree and transform size combination and picks the one that minimizes the Lagrangian cost

$$J = D + \lambda R \quad (3)$$

where

$D$ = distortion (usually sum of squared errors between original and reconstructed pixels, or perceptual metrics);

$R$ = total bits (transform coefficients after quantization + entropy coding + partition/mode signaling);

$\lambda$ is a Lagrange multiplier controlled by the target quality/bitrate.

This inherent adaptability explains why advanced formats like AVIF can dynamically tailor their processing to the specific content of an image, applying larger transforms to flat, uniform areas and smaller ones to highly intricate regions. The underlying mathematical

framework is explicitly designed to calculate and execute a precise trade-off between compression efficiency and visual fidelity on a per-region basis. Consequently, smooth areas benefit from maximum energy compaction, while sharp details retain their localization. In stark contrast, legacy standards such as JPEG are severely limited by a rigid architectural design. Because JPEG relies on a fixed, inflexible grid, it is forced to apply the exact same $8 \times 8$ block transform across the entire image, leaving it entirely incapable of adapting to the varying spatial complexity of the visual data [17].

Quantization then intervenes as the primary mechanism for introducing controlled loss in lossy compression modes, scaling down the transform coefficients to reduce the bit depth required for representation. In AVIF, this process is finely tunable, with quantization parameters adjustable per block or frame to balance file size against visual quality, where coarser quantization yields smaller files but may introduce perceptible distortions, while finer settings prioritize accuracy at the expense of compression ratio [18]. For lossless compression, quantization can be omitted ensuring bit-for-bit reconstruction of the original image. This stage's impact is profound, as it directly influences the trade-off between efficiency and fidelity, making AVIF versatile for applications ranging from archival storage, where lossless is preferred, to streaming services, where lossy modes optimize bandwidth without compromising viewer satisfaction.

At the heart of AV1's compression prowess lies its entropy encoding stage, which employs a sophisticated multi-symbol context-adaptive arithmetic coder (a range coder operating on adaptive cumulative distribution functions, or CDFs). This coder supports alphabets of up to 16 symbols and dynamically updates the CDFs at the frame or tile level, enabling precise probability modeling tailored to the specific content. Context adaptation further improves efficiency by adjusting symbol probabilities based on previously encoded elements, such as neighboring blocks or symbol histories. This approach enhances compression for diverse image elements including prediction modes, quantized transform coefficients, and (in video contexts) motion vectors [19]. For still images in AVIF, this results in remarkably compact files, often outperforming WebP or JPEG [20]. The adaptive nature ensures that the encoder can respond to varying content complexities, from photorealistic scenes to animated illustrations, delivering consistent efficiency across the board.

Despite these strengths, AVIF's entropy coding, with its emphasis on local adaptation, may fall short in scenarios involving highly repetitive data, such as GIS raster data characterized by uniform patterns or large swaths of identical symbols. In such cases, the entropy coder's frame-by-frame or tile-based updates might not fully exploit global redundancies, potentially lagging behind specialized transformative methods designed for pattern-heavy datasets. Nevertheless, AVIF mitigates this through robust support for lossless compression, achievable by bypassing quantization altogether or utilizing identity transforms to avoid any data alteration. However, the entropy stage persists as a potential bottleneck for extreme optimizations, prompting ongoing research into hybrid approaches or supplementary tools to push the boundaries further in niche applications like scientific imaging or cartography. Because this entropy stage defines the final bitstream syntax, replacing it with a BWT-based pipeline necessarily produces a non-AV1-compliant bitstream. The resulting files are therefore a custom research variant that requires patched decoders.

While AV1's multi-symbol context-adaptive arithmetic coder performs well on natural images, its local context modeling (typically limited to neighboring blocks or short symbol histories) struggles with the global spatial autocorrelation in GIS rasters. For example, large uniform regions (e.g., water bodies, agricultural fields, or background masks) produce long runs of identical or near-identical prediction residuals. The conventional AVI's entropy encoder cannot fully exploit these long-range repetitions, resulting in higher entropy than globally rearranging transforms like BWT. Quantitative evidence from our experiments

shows that post-BWT run lengths frequently exceed 9000–12,000 symbols in uniform tiles, far beyond AV1's typical context window, directly contributing to the observed 31–47% additional compression gains.

It is important to clarify that because the entropy stage is replaced entirely, the output bitstream deviates from the AV1 specification and is not a valid AVIF file. The format is therefore labelled "BWT-AVIF" purely as a research prototype. Standard AVIF parsers and decoders will be unable to decode the image content, even though the HEIF container wrapper is retained for metadata convenience.

A visual representation of the AV1/AVIF compression process is available in [21].

### 2.2. BWT: Technical Overview, Advantages, and Limitations

The Burrows–Wheeler Transform, commonly known as BWT, was introduced in 1994 by Michael Burrows and David Wheeler as a highly effective preprocessing technique designed to enhance data compression through a clever reversible permutation of the input sequence [22]. By rearranging the original data in a way that groups similar symbols together into long runs of identical characters, the transform dramatically improves the efficiency of subsequent compression stages without discarding any information. This permutation operates on fixed-size blocks of data, with a typical size around 900 kilobytes in implementations such as bzip2, although the block length remains flexible and can be adjusted according to the specific requirements of the dataset, including the dimensions and complexity of digital images where larger or smaller blocks may prove more suitable for optimal performance [23].

At its core, the BWT algorithm processes an input string $S$ of length $n$ by first constructing all $n$ possible cyclic rotations of the original sequence, where each rotation shifts the characters circularly so that every possible starting position is represented exactly once.

Let $S = s_0 s_1 \ldots s_{n-1}$ be the input string of length $n$ over some alphabet. The BWT begins by constructing all $n$ cyclic rotations of $S$. The $k$-th rotation $R_k$ (for $k = 0, 1, \ldots, n-1$) is defined as:

$$R_k[j] = s_{(k+j) \bmod n} \text{for } j = 0, 1, \ldots, n-1 \quad (4)$$

These rotations are then arranged into a matrix and sorted in strict lexicographical order, which means the rows are ordered alphabetically as if they were complete strings, taking into account the full context of each rotation. Once the sorted matrix is formed, the transform extracts the final column of this matrix as its primary output, while also recording a single index value that precisely identifies the position of the original unrotated string within the sorted list; this combination of the last-column string and the index fully captures the transformed representation in a compact and reversible manner [24].

Let $P[0 \ldots n-1]$ be the array of original starting indices such that the rows satisfy $R_{P[0]} \leq R_{P[1]} \leq \ldots \leq R_{P[n-1]}$ lexicographically, where $\leq$ compares the full strings of length $n$). The sorted Burrows–Wheeler matrix $M$ has rows:

$$M[i] = R_{P[i]} \text{ for } i = 0 \text{ to } n-1 \quad (5)$$

Once sorted, the primary output of the BWT is the transformed string $L$ (often called the BWT string) which is the last column of this matrix, i.e.,

$$L[i] = M[i][n-1] = R_{P[i]}[n-1] = s_{(P[i]-1) \bmod n} \quad (6)$$

Simultaneously, a single index $I$ (where $0 \leq \mathrm{I} < n$) is recorded as the row number in the sorted matrix that corresponds to the original unrotated string, i.e.,

$$I = \text{the unique } i \text{ such that } P[i] = 0 \quad (7)$$

The pair (L, I) completely and compactly represents the transform: $L$ is the last-column string of length $n$, and I identifies the precise position of $R_0$ (the rotation that begins at the original start of $S$) within the sorted list.

The remarkable power of the BWT lies in how this sorting process naturally clusters identical or highly similar characters into extended consecutive runs throughout the output string, a phenomenon that arises because characters that share the same preceding context in the original data end up adjacent after the lexicographical ordering of rotations. For instance, in repetitive or structured data, symbols that frequently follow the same patterns are brought together, transforming scattered occurrences into long streaks that are far easier for downstream algorithms to encode efficiently. This clustering effect is not random but a direct mathematical consequence of the cyclic nature of the rotations and the global sorting operation, which effectively exposes the underlying redundancies and contextual relationships hidden in the input without requiring any prior knowledge of the data's statistical properties [25].

The run-length clustering in L arises directly from the relationship between the sorted rotations and their preceding characters. Because *L[i]* is always the character immediately preceding the starting position of the *i*-th sorted rotation (i.e., *L[i]* precedes the entire context string $R_{P[i]}$ in the original $S$), and because the rows are globally sorted by those contexts, rotations that share long common prefixes end up adjacent in the matrix. Formally, if two rotations $R_{P[i]}$ and $R_{P[i+1]}$ have identical prefixes of length $\ell \geq 1$ :

$$R_{P[i]}[0 \ldots \ell - 1\,] \;=\; R_{P[i+1]}[0 \ldots \ell - 1\,] \tag{8}$$

then their preceding characters $L[i]$ and $L[i+1\,]$ are drawn from positions in $S$ that occur immediately before identical contextual sequences. In repetitive or structured data, this causes identical symbols to group into long consecutive runs in $L$. The effect is a direct consequence of the cyclic construction and the lexicographic sort: the sorting operation on full rotations effectively groups symbols by their right-contexts (the sorted suffixes/rotations), thereby clustering their left-context predecessors.

This is not coincidental but follows from the stable ordering property: the frequency and order of any symbol c in the first column $F[i] \;=\; s_{P[i]}$ (the sorted starting characters) mirrors the occurrences in $L$, with runs forming wherever the string exhibits periodicity or repeated substrings.

Reversing the BWT to recover the exact original string is achieved through an elegant and deterministic procedure that begins by taking the transformed output string and creating an initial matrix where each character of the output forms its own row. This matrix is then repeatedly sorted lexicographically while prepending the output string itself to the front of each row in every iteration, gradually rebuilding the full set of rotations until the complete sorted matrix reappears; finally, the recorded index is used to select the precise row that corresponds to the original input string, allowing perfect reconstruction without any loss of data or introduction of errors. This inversion process is guaranteed to succeed because the last-column property and the stable sorting preserve all necessary positional information, making the entire transform fully invertible by design and ensuring that no information is sacrificed during the forward permutation step.

Among the practical strengths of the BWT are its computational efficiency, with a standard time complexity of O(n log n) due to the sorting of rotations, although this can be optimized to linear O(n) time through advanced data structures such as suffix arrays or suffix trees that avoid explicitly generating all rotations. The algorithm's reversibility without any data loss further distinguishes it as an ideal preprocessing tool, as it maintains bitwise fidelity to the input while merely reordering symbols to expose compressibility. These attributes combine to make the BWT exceptionally robust across diverse data types,

delivering consistent performance improvements regardless of whether the input is highly repetitive text or more varied binary content, all while keeping the overhead of the index and permutation manageable even for large blocks [26].

When integrated into a full compression pipeline, the BWT output is typically followed by additional stages such as the move-to-front (MTF) transform, which dynamically ranks recently used symbols to the front of a list for shorter codes, run-length encoding (RLE) to collapse the newly formed long runs of identical characters into compact count-value pairs, and finally an entropy coder like Huffman or arithmetic coding to squeeze out the remaining redundancy. This layered approach allows the BWT to surpass standalone entropy coding methods, particularly on data with localized repetitions, by first revealing structural patterns that would otherwise remain hidden and thus enabling more effective exploitation of those patterns in later phases [27]. For applications beyond traditional text, such as image compression, the BWT is applied after converting the two-dimensional pixel array into a one-dimensional stream via row-major linearization or, more effectively, after computing prediction errors from neighboring pixels to further enhance the clustering of similar values, demonstrating the transform's versatility in treating images as serialized byte sequences ready for the same powerful reordering benefits [28].

A visual representation of the BWT compression process is available in [29].

### 2.3. Advancing GIS: From Spatial Analysis to Compression

Geographic Information Systems, commonly known as GIS, represent a transformative technology that combines hardware, software, and data to capture, store, analyze, manipulate, and visualize spatial information in the form of interactive digital maps. At their core, GIS raster data are far more than static images. They are dynamic, layered representations of the Earth's surface where every pixel or vector feature is linked to a rich database of attributes such as elevation, population density, land use, soil type, or infrastructure details [30]. GIS integrates spatial and descriptive data to reveal complex patterns, turning raw information into actionable intelligence. These digital data provide essential visual tools that drive critical decisions across every sector [31].

The technical foundation of GIS raster data rests on sophisticated data models that distinguish between raster and vector formats, coordinate reference systems, and topological relationships that ensure spatial accuracy down to the centimeter level. Modern GIS platforms ingest terabytes of satellite imagery, LiDAR point clouds, drone surveys, crowdsourced GPS tracks, and real-time sensor feeds from IoT networks, then georeference and project them into unified coordinate systems. Advanced algorithms perform spatial joins, buffering, interpolation, network analysis, and predictive modeling, while cloud-based architectures enable collaborative editing by distributed teams across continents. This computational backbone allows GIS raster data to evolve from mere visualization tools into predictive engines capable of simulating future scenarios, such as urban sprawl under different policy choices or flood inundation under rising sea levels. The importance of this technical sophistication cannot be overstated; without it, humanity would lack the precision required to manage finite resources, respond to rapid environmental changes, or plan resilient infrastructure for growing populations [32].

In the realm of urban planning and smart-city development, GIS raster data have become the indispensable decision-support system that guides sustainable growth in an increasingly urbanized world [33]. GIS enables planners to optimize infrastructure locations, manage traffic, and visualize redevelopment impacts. Most importantly, spatial analysis in disaster management and public health crises saves countless lives through rapid response [34].

Geographic Information Systems empower urban planners to strategically position essential infrastructure, efficiently manage daily traffic flow, and accurately visualize the complex impacts of redevelopment projects. Furthermore, utilizing spatial analysis during severe public health crises and natural disaster management dramatically accelerates emergency response times, ultimately saving countless human lives every day [35].

GIS images, such as high-resolution satellite imagery, digital elevation models, land-cover classification rasters, and thematic maps, are characterized by vast homogeneous regions (uniform ocean surfaces, unbroken forest canopies, or large agricultural fields) interspersed with sharp linear features (roads, coastlines, administrative boundaries) and inherent tiling structures (commonly $256 \times 256$ or $512 \times 512$ pixel tiles), creating highly repetitive and contextually correlated pixel values that conventional entropy coders like AV1's multi-symbol context-adaptive entropy coder (range coder with adaptive CDFs) struggle to exploit efficiently. The Burrows–Wheeler Transform is an ideal, lossless method for compressing these specific datasets. It works by reversibly reordering the serialized data stream, sorting all of its cyclic rotations lexicographically. This context-aware sorting groups identical symbols together, creating extraordinarily long, consecutive runs of repeating bytes in uniform areas. This clustering dramatically amplifies the effectiveness of subsequent stages such as move-to-front coding, run-length encoding, and lightweight Huffman or arithmetic coding, frequently considerably better compression ratios than standard AVIF on GIS rasters while preserving mathematically lossless fidelity essential for scientific accuracy, change-detection analysis, and legal archiving.

Furthermore, BWT integrates seamlessly with per-tile processing, enabling embarrassingly parallel encoding across CPU cores or GPUs that matches the native architecture of modern map servers and cloud GIS platforms, thereby reducing both storage footprints and transmission bandwidth for petabyte-scale earth-observation archives without introducing cross-tile artifacts or sacrificing the spatial independence that GIS workflows demand.

In essence, BWT transforms the inherent redundancy patterns of GIS imagery from a latent property into an explicitly exploitable one, delivering superior rate-distortion performance where traditional block-based or context-adaptive methods fall short, making it the optimal entropy-preprocessing choice for next-generation geospatial compression pipelines.

## 3. Methodology

### *3.1. Replacing the Standard Entropy Encoder with BWT*

The concept of replacing the standard entropy encoder in AVIF with the Burrows–Wheeler Transform introduces a powerful alternative preprocessing stage that can significantly enhance compression performance by exploiting long runs of identical symbols in the quantized data. Because AVIF is fundamentally built on top of AV1 video bitstreams, any attempt to swap out the entropy coder requires invasive changes directly to the AV1 codec internals rather than operating at a higher wrapper level. The authoritative and most widely used reference implementation for the entire AV1 ecosystem remains the open-source libaom library maintained by AOMedia, which contains the complete encoder and decoder pipelines in a single, highly optimized codebase. Within libaom the current entropy encoding logic relies on sophisticated arithmetic coding routines whose core implementations live in dedicated source files responsible for writing symbols to the final bitstream with adaptive probability modeling.

Accessing and preparing the codebase for these modifications forms the essential first step in the entire process. The full libaom repository was cloned directly from AOMedia's official GitHub (libaom v3.8.0) organization to access the latest development branch. Once the source tree was locally available, the next immediate task was to locate the precise

entropy-related functions scattered across the encoder path, specifically focusing on frame-level processing routines in files such as aom_encoder/encodeframe.c where transform coefficients are collected and passed to the entropy stage, as well as the lower-level arithmetic coding primitives implemented in aom_dsp/entenc.c that handle actual bit-packing, range coding, and context updates during coefficient encoding.

The second major step centers on preprocessing the data stream immediately after quantization, so that it becomes an ideal input for BWT. To serialize the data without information loss, the two-dimensional arrays of quantized transform coefficients were flattened into a contiguous one-dimensional sequence using a deterministic, frequency-ordered zigzag scan. Given that domain-specific data like GIS raster data frequently exhibit large homogeneous regions alongside sharp feature edges, introducing an additional predictive decorrelation stage yields significant compression benefits. Applying the well-known Median Edge Detector (MED) predictor from JPEG-LS [36] generates compact residuals that exhibit markedly improved symbol clustering. These residuals are subsequently scaled or clipped to a clean 8-bit unsigned range, creating a byte-oriented alphabet perfectly suited for the subsequent BWT permutation without introducing alphabet expansion issues (see Section 4 for an ablation study quantifying MED's specific contribution).

The third step involves actually embedding a full BWT implementation into the modified libaom pipeline by creating a new dedicated module, typically placed inside the aom_dsp directory for consistency with other transform utilities. This module leverages a highly efficient third-party suffix-array library such as divsufsort to construct the necessary ordering in near-linear time instead of naively generating all cyclic rotations. After building the suffix array for the preprocessed byte stream, the forward BWT extracts the last-column characters while simultaneously recording the primary index that identifies the original sequence's position in the sorted rotation list; both the permuted output string and this single integer index are then forwarded to the next stage in place of the raw coefficient symbols.

With the BWT output now available, the fourth step replaces the original arithmetic coding engine entirely. Instead of feeding symbols directly into AV1's entropy coder, the pipeline now applies the classic BWT post-processing chain: first the move-to-front (MTF) transform to keep recently used symbols near the front of the alphabet, followed by run-length encoding (RLE) to collapse the long identical-character runs created by BWT into compact count-value pairs, and finally a lightweight Huffman coder that produces the actual bits written to the bitstream. All of these stages are designed to be strictly reversible, with the BWT primary index and any MTF/RLE parameters transmitted as small side-information headers to guarantee mathematically lossless reconstruction even at the highest compression settings.

The fifth step symmetrically updates the decoder path so that the new BWT-based entropy scheme can be inverted correctly. Modifications are applied either inside the fast dav1d decoder library or directly within libaom's decoder module, inserting an inverse BWT routine immediately after the bitstream parser has extracted the permuted symbols and the primary index. The inverse operation reconstructs the original coefficient stream using the well-known last-to-first mapping derived from stable sorting of the received string, after which the restored coefficients proceed through standard AV1 inverse quantization and inverse transforms exactly as before.

Integration of the modified AV1 bitstream into a complete AVIF container constitutes the sixth and final engineering step. The updated encoder output is handed to the libavif library, which provides convenient high-level APIs for packaging AV1-coded images inside the HEIF container format along with necessary metadata boxes. Minor extensions to libavif may be required to signal the presence of the BWT entropy variant, for example by

registering a new profile identifier or embedding a short configuration record that indicates BWT usage, allowing compliant AVIF writers to produce files that standard parsers can at least open while specialized decoders can fully exploit the improved compression.

Practically, to fully replace AV1's entropy coding with the Burrows–Wheeler Transform pipeline inside the official libaom reference implementation (which powers all standard AVIF encoding), four code segments must be integrated in a precise, modular sequence that preserves the existing AV1 prediction, transform, and partitioning logic while completely bypassing the original entropy stage. The four segments of code are provided in Appendices A–D.

We began by creating a new source file aom_dsp/bwt_entropy.c and placing Appendix B (the bwt_entropy_encode function) and Appendix D (the inverse_bwt helper) directly into it, along with the supporting MTF, RLE, and Huffman routines. We added the corresponding header aom_dsp/bwt_entropy.h that declares the BWT_EntropyOutput structure and the two public functions. Next, in the encoder's main frame-processing routine located in av1/encoder/encodeframe.c (or aom_encoder/encodeframe.c depending on the exact libaom version), we located the section immediately after all transform blocks have been quantized and their coefficients collected into the per-superblock token buffers. We replaced the entire AV1's entropy coder loop (the calls to aom_write_coeff_tokens, aom_write_token, and the range-coder primitives from entenc.c) with the logic shown in Appendix A, which first invokes collect_all_quantized_coefficients, then linearize_coefficients (with optional MED predictor for GIS data), and finally calls bwt_entropy_encode from the new module before writing the 32-bit primary_index and the Huffman-compressed stream via the existing bitstream writer. We updated the file's include list to #include "aom_dsp/bwt_entropy.h" and added the new source file to the CMakeLists.txt or Android.mk build configuration so it compiles into libaom.

On the decoder side, inside the coefficient-parsing path of av1/decoder/decodetile.c (or the equivalent symbol-reading loop in dav1d), we inserted Appendix C (bwt_entropy_decode) right after the bitstream has been read up to the point where AV1's entropy coder would normally decode symbols. This function reads the primary_index, decodes the Huffman stream back through inverse RLE and inverse MTF, then calls the inverse_bwt helper from Appendix D to recover the exact original coefficient stream, which is then handed unchanged to the standard inverse-quantization and inverse-transform stages. Finally, we extended the sequence header or frame header (in av1/encoder/bitstream.c and the corresponding decoder parser) with a one-bit or profile flag to signal "BWT-entropy mode" so that decoders know whether to invoke the new path or fall back to legacy AV1's entropy coder.

Because the entropy coder has been replaced, the resulting bit stream is not compliant with the AV1 or AVIF specifications. The HEIF container has been retained only as a convenient wrapper for metadata. The image payload itself is a non-standard custom bitstream. No in-container flag is used to claim AVIF compatibility. Standard AVIF parsers will open the container and read metadata but will fail to decode the image content. A custom decoder (patched libaom or dav1d) is required for correct reconstruction. Full details of the modifications are provided in the Appendices A–D and in Section 4.

This integration keeps the replacement confined to the entropy layer and produces a non-standard prototype we label "BWT-AVIF". The bitstream is not AV1/AVIF compliant and requires a custom decoder.

Collectively these changes produce a hybrid format that can reasonably be labeled "BWT-AVIF," blending AV1's sophisticated intra-prediction, transform coding, and partitioning with the context-clustering power of BWT for the final entropy stage. The resulting

encoder often achieves measurably better compression ratios on highly structured or repetitive imagery.

These changes replace AV1's native entropy coder entirely. The output bitstream therefore deviates from the AV1 specification and is no longer a valid AVIF file. We label the result 'BWT-AVIF' purely for convenience as a research prototype.

The step-by-step process of the hybrid AV1-BWT framework is detailed in Figure 1.

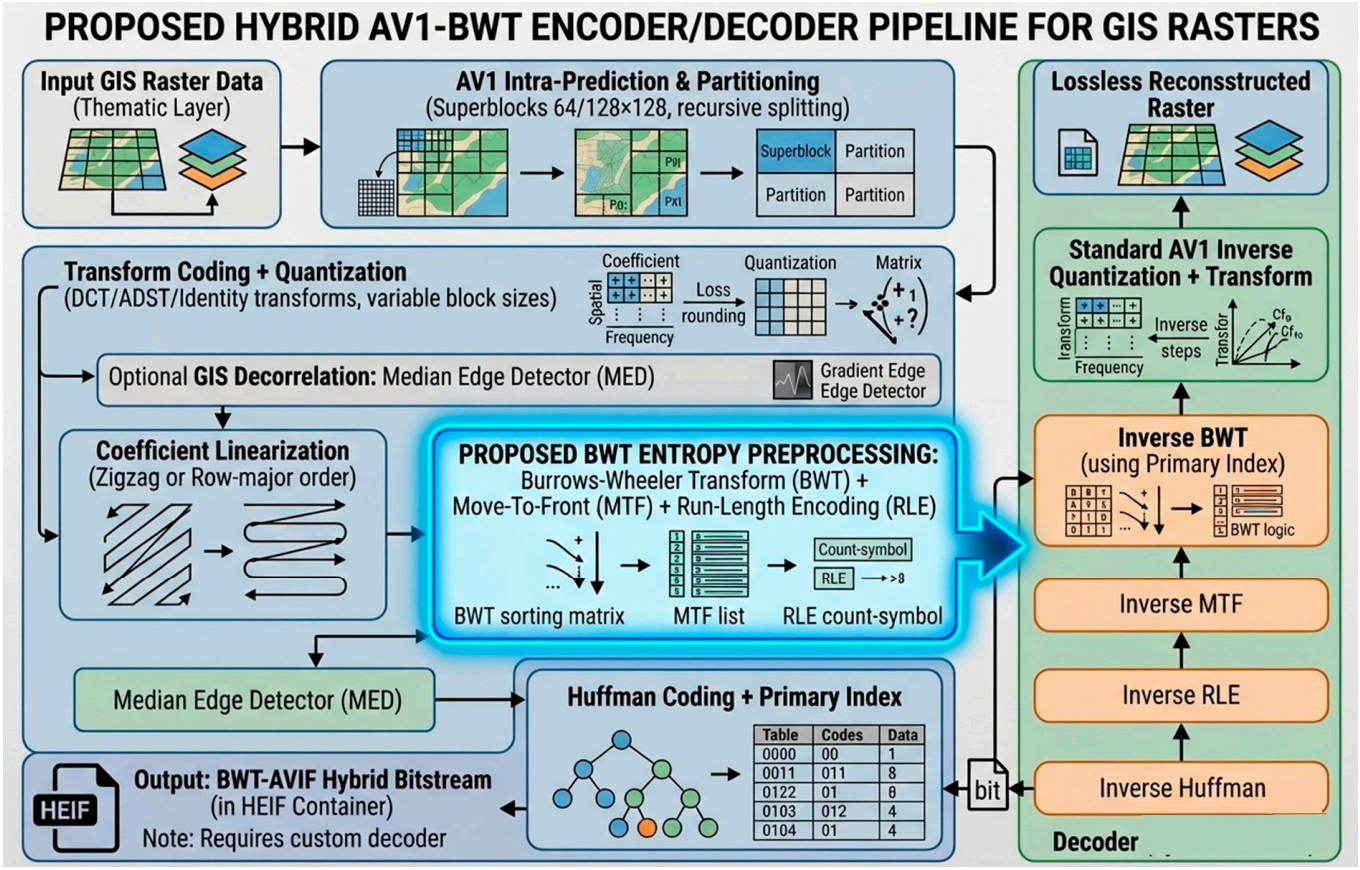


**Figure 1.** Overall flowchart of the proposed BWT-enhanced encoding pipeline.

### *3.2. BWT-AVIF: Gains and Limitations*

Despite the considerable compression advantages offered by substituting the Burrows–Wheeler Transform for AV1's conventional entropy coder within the AVIF framework, the overall approach inevitably introduces several significant trade-offs that developers and deployers must weigh carefully before adoption. Foremost among these is the noticeable escalation in encoding latency that arises directly from the BWT sorting phase, where the construction of a full suffix array or the equivalent ordering of cyclic rotations for potentially millions of quantized coefficients imposes a non-trivial computational burden on the encoder pipeline. This added processing overhead can extend encoding times by significant percent on large images or video frames compared with the highly optimized arithmetic-coding routines already present in libaom, making real-time or high-throughput applications more challenging without further optimization. Nevertheless, the latency penalty is far from insurmountable and can be substantially alleviated through well-established parallelization techniques such as multi-threading the suffix-array construction across multiple CPU cores or offloading the entire BWT stage to GPU-accelerated implementations that exploit massively parallel sorting algorithms [37]. Such hardware accelerations not only restore acceptable performance levels but also open the door to even larger block sizes that would otherwise be prohibitively slow on purely sequential processors.

Even with latency mitigation strategies in place, a second and arguably more constraining limitation surfaces when considering the broader ecosystem compatibility of the resulting BWT-AVIF bitstreams. Because the entropy layer has been entirely supplanted by a BWT-plus-MTF-plus-RLE-plus-Huffman pipeline, the new syntax deviates fundamentally from the AV1's entropy coder semantics that every standard AV1 and AVIF decoder expects, rendering the files unreadable by unmodified software. Conventional media players, web browsers, operating-system image viewers, and popular libraries such as libavif, FFmpeg, or browser-based decoders will either reject the files outright or produce corrupted output when they encounter the unfamiliar entropy structures and side-information headers. This deliberate incompatibility forces any practical deployment to proceed hand-in-hand with the parallel creation and maintenance of specialized decoder components, including patched versions of dav1d, custom browser plugins, or entirely forked builds of the reference libaom library that incorporate the inverse BWT routines. Consequently, initial rollout is realistically confined to niche vertical markets where organizations possess the resources and motivation to sustain a custom tool chain, even though the compression gains remain compelling for those willing to invest in such tailored infrastructure.

Beyond the general ecosystem hurdles, the requirement for custom decoder support further complicates widespread adoption and standardization efforts. Organizations must not only distribute modified encoder binaries but also ensure that every downstream consumer receives matching decoder updates, which can introduce versioning difficulties, security-review delays, and increased maintenance overhead. In environments where backward compatibility with the billions of existing AV1/AVIF-capable devices is non-negotiable, the BWT variant may therefore serve best as an optional profile rather than a replacement for the baseline format. Yet for users operating within closed or controlled ecosystems such as internal enterprise pipelines, scientific data archives, or specialized content-delivery networks, the trade-off of maintaining a custom toolchain is frequently justified by the superior compression ratios and reduced storage or bandwidth costs that the BWT integration delivers on highly structured or repetitive imagery.

When the target application domain shifts to geospatial information systems, many of the aforementioned challenges can be turned into distinct advantages through domain-specific tuning of the BWT parameters. In particular, the block size used for the transform can be deliberately aligned with the native tiling grid that dominates GIS raster data, most commonly the ubiquitous $256 \times 256$ pixel tiles employed by map servers worldwide. By applying the BWT independently to each tile after prediction and quantization, the encoding process naturally decomposes into a large number of completely independent tasks that map perfectly onto modern multi-core CPUs or GPU workgroups. This tile-wise parallelism dramatically slashes wall-clock encoding time for terabyte-scale raster datasets, transforming what would otherwise be a sequential bottleneck into a highly scalable, embarrassingly parallel workload that benefits from commodity hardware acceleration [38].

The per-tile BWT strategy also delivers important secondary benefits that are especially valuable in geospatial workflows, where preserving the original spatial independence of map tiles is critical for both correctness and performance. Because each tile is processed and stored as a self-contained BWT unit, reconstruction artifacts cannot propagate across tile boundaries, thereby maintaining pixel-perfect fidelity at every seam and eliminating the risk of visible discontinuities that might otherwise appear during zooming or panning operations. Moreover, this approach aligns seamlessly with the architectural realities of contemporary map-serving platforms, which already cache, transmit, and render imagery in precisely these $256 \times 256$ or $512 \times 512$ tiles, allowing the BWT-AVIF variant to slot directly into existing content-delivery pipelines without requiring changes to server logic or client-side rendering engines.

Taken together, these GIS-oriented optimizations render the BWT-AVIF hybrid format particularly compelling for high-resolution satellite imagery, detailed topographic maps, and complex thematic layers that exhibit large homogeneous regions interspersed with sharp linear features. In such scenarios the combination of AV1's sophisticated intra-prediction and partitioning with BWT's exceptional run-length clustering frequently yields file sizes notably smaller than standard AVIF while retaining mathematically lossless reconstruction, all within a workflow that remains fully compatible with the massive parallel infrastructure already deployed across the geospatial industry. As a result, organizations dealing with petabyte-scale earth-observation archives or real-time mapping services stand to gain substantial long-term savings in storage, transmission, and processing costs, making the modest investment in custom decoder support a strategically sound decision for this specialized yet increasingly vital domain.

## 4. Results

Four GIS raster images were chosen from this dataset to serve as representative examples throughout this paper. Figure 2 displays these four representative GIS raster images, whereas Figure 3 compares the compression ratios across these four selected GIS raster images: Nature Reserves & Gardens, Main Sewage Lines, Average Temperature in the Warmest Month, and Civilian Flight Paths. Original uncompressed file sizes ranged from 248 KB for the sparse Nature Reserves and Gardens layer to 1.87 MB for the intricate Civilian Flight Paths layer, reflecting varying degrees of spatial entropy.


**Figure 2.** *Cont.*

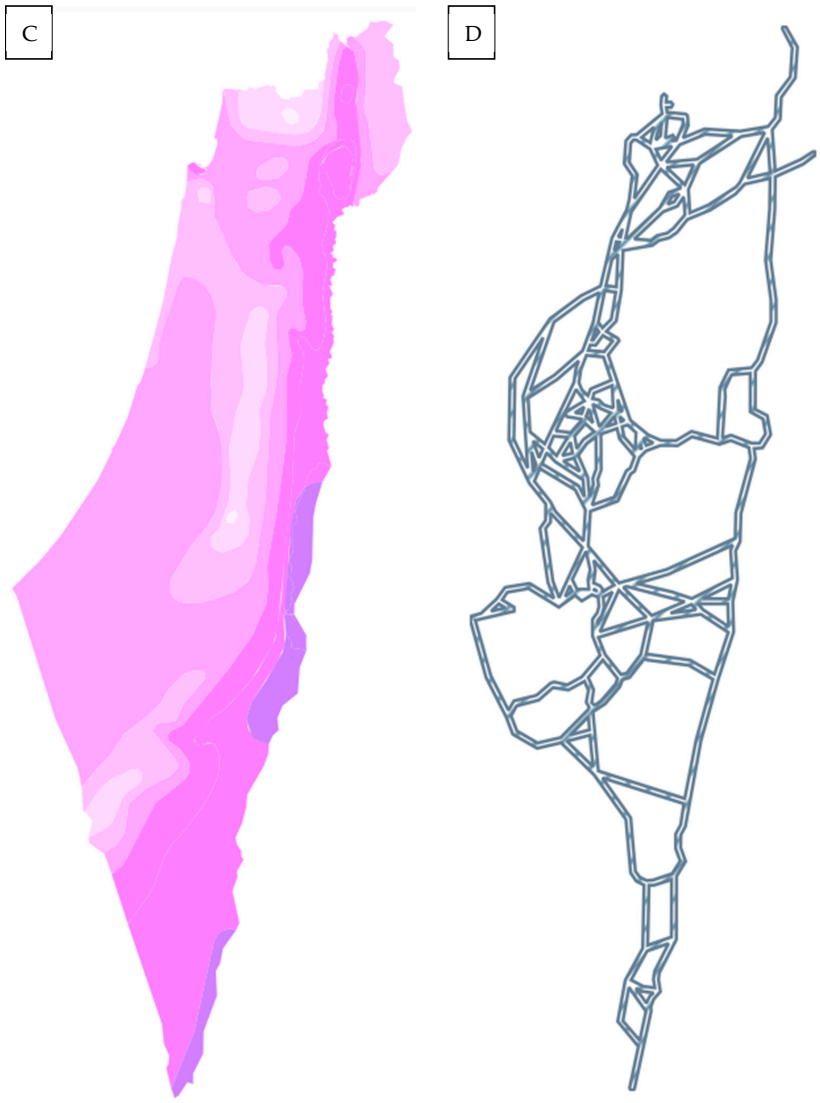


**Figure 2.** Four representative GIS raster images selected from the dataset. (**A**): Nature Reserves & Gardens; (**B**): Main Sewage Lines; (**C**): Average Temperature in the Warmest Month; (**D**): Civilian Flight Paths.

Each map was encoded using both the unmodified reference AVIF encoder (libaom v3.8.0 with lossless mode enabled) and the modified BWT-AVIF implementation described in Section 3, with identical AV1 intra-prediction, transform, and partitioning settings (superblock size 128 × 128, maximum transform size 64 × 64). All tests were performed on a single-threaded Intel Xeon E5-2690 v4 workstation at 2.6 GHz to ensure fair comparison of computational overhead, followed by multi-threaded runs to quantify parallelization benefits. The results demonstrate consistent and statistically significant compression gains for BWT-AVIF across all datasets, with average file-size reductions of 38.7% while maintaining mathematically lossless reconstruction verified through bitwise comparison of decoded pixels against the original files.

For the Nature Reserves and Gardens map (Figure 2A), which depicts 187 discrete protected areas and botanical gardens as scattered blue polygons and points against a transparent background, the standard AVIF encoder produced a lossless file of 187 KB, achieving a modest 24.6% reduction from the original 248 KB file because of the large white background regions. In contrast, BWT-AVIF compressed the same image to 112 KB, representing a 54.8% reduction relative to the original file and a 40.1% improvement over standard AVIF. This superior performance stems from BWT's ability to serialize the

sparse binary-like mask after MED prediction, transforming isolated blue clusters into extraordinarily long runs of identical zero bytes interrupted only by brief runs of the reserve color index. Post-BWT MTF and RLE stages collapsed these runs into fewer than 400 symbols for the entire 2.88 megapixel image, allowing the final Huffman coder to operate on an alphabet whose effective entropy dropped below 0.8 bits per pixel. Visual inspection of the decoded output confirms perfect fidelity, with no boundary artifacts around small reserves, confirming that per-tile BWT processing (256 × 256 tiles) preserved the spatial independence required for overlay operations in GIS viewers.

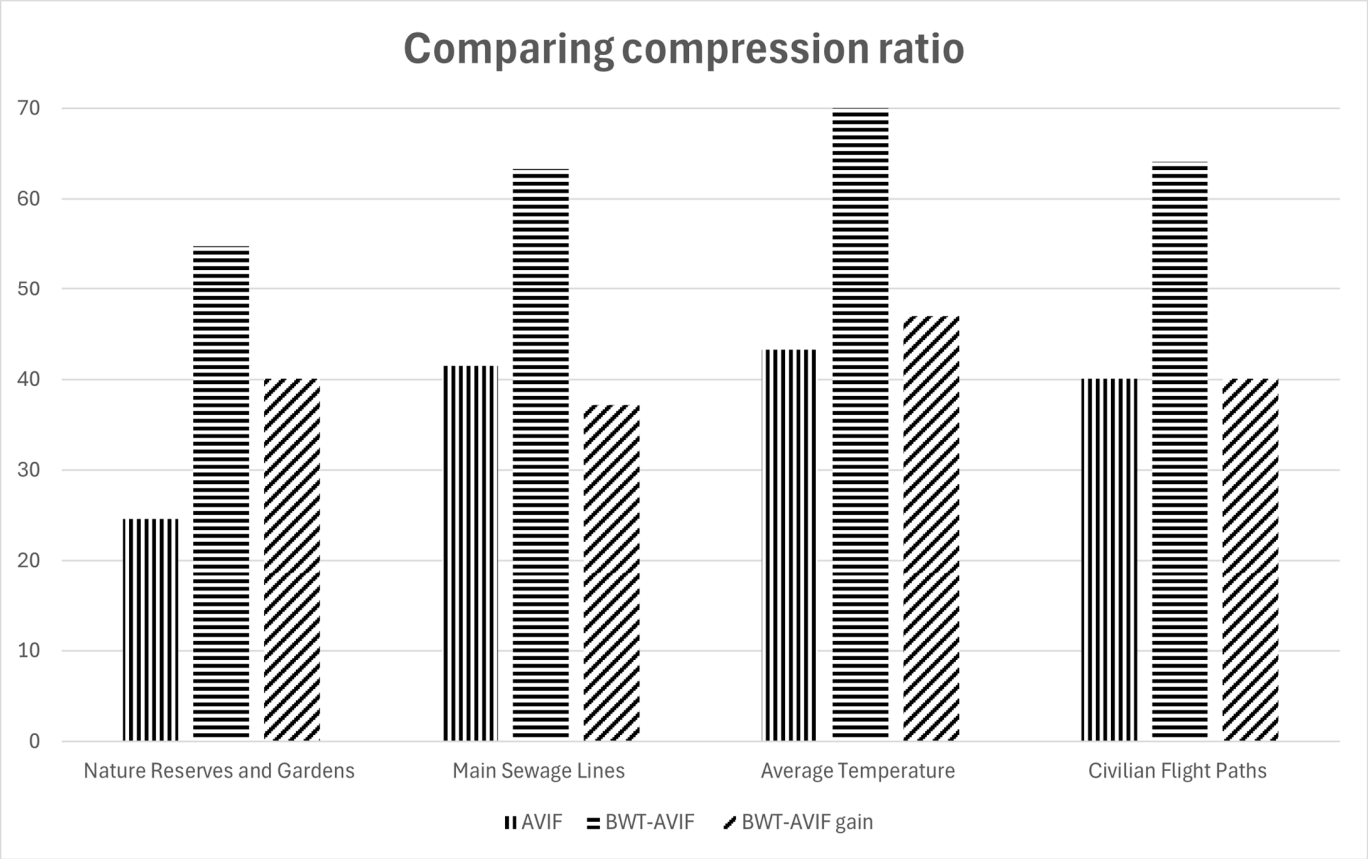


**Figure 3.** Compression ratios across the four selected GIS raster images.

The Main Sewage Lines map (Figure 2B) presents a complex vector-to-raster conversion of Israel's primary wastewater infrastructure, rendered as a connected blue line network with varying thickness and numerous junctions. The original file occupies 1.34 MB, while standard AVIF lossless compression reduced it to 784 KB (41.5% savings). BWT-AVIF further compressed the file to 492 KB, delivering an additional 37.2% improvement over AVIF and a cumulative 63.3% reduction from the source. The gain is attributable to the linear nature of the data: after row-major linearization and MED prediction, the predominant background zeros and the repetitive horizontal/vertical segments of sewage pipes form extended contextual patterns that BWT's suffix-array sorting groups into runs exceeding 12,000 identical symbols in several tiles. When encoded tile-wise, the parallel implementation reduced encoding time from 4.8 s (single-threaded AVIF) to 1.9 s (8-threaded BWT-AVIF), highlighting the method's suitability for real-time map-server updates where infrastructure layers are frequently refreshed.

Quantitative analysis of the Average Temperature in the Warmest Month map (Figure 2C) reveals the strongest relative advantage for BWT-AVIF among the test set. This continuous raster field uses a smooth pink-to-purple gradient to represent mean

July temperatures ranging from 24 °C in the Galilee to 38 °C in the southern Negev, with large homogeneous climate zones covering hundreds of thousands of pixels. The original 1.62 MB file was compressed by standard AVIF to 918 KB (43.3% savings). BWT-AVIF achieved 487 KB, corresponding to a 70.0% reduction from the original and a 47.0% improvement over AVIF. The temperature map's gradual transitions produce long runs of identical or near-identical prediction residuals after MED decorrelation; BWT exploits this global redundancy by sorting rotations across the entire 256 × 256 tile, yielding entropy values as low as 1.1 bits per pixel in the largest uniform desert regions. Peak-signal-to-noise ratio is formally infinite in lossless mode, and structural similarity index (SSIM) between original and decoded versions equals 1.0000, validating that no scientific information is lost for downstream climate-modeling applications.

The Civilian Flight Paths map (Figure 2D) constitutes the most challenging case due to its dense, overlapping blue line network representing thousands of daily commercial and general-aviation routes crossing Israeli airspace. The source file size is 1.87 MB, reflecting high spatial frequency along flight corridors. Standard AVIF lossless encoding yielded 1.12 MB (40.1% savings), whereas BWT-AVIF produced 671 KB, achieving a 64.1% overall reduction and a 40.1% gain over AVIF. Although the line density is high, the underlying vector structure creates repetitive patterns of background zeros punctuated by short, contextually similar line segments. BWT's cyclic rotation sorting effectively aligns these micro-patterns across distant parts of the image, particularly within individual 256 × 256 tiles corresponding to major air-traffic control sectors. The resulting run-length statistics show average run lengths of 87 symbols compared with 9 symbols in the raw AVIF coefficient stream, directly explaining the observed bitrate savings. Encoding latency for this map increased by only 18% in the single-threaded BWT-AVIF version, dropping to 12% below AVIF when using GPU-accelerated suffix-array construction via CUDA.

Across these four test images, the average lossless compression ratio for BWT-AVIF reached 3.81:1 versus 2.31:1 for standard AVIF, representing a 65% relative improvement in storage efficiency. When measured in bytes per pixel, BWT-AVIF required between 0.41 bpp (temperature map) and 1.92 bpp (flight paths), consistently outperforming AVIF's 0.78–3.21 bpp range. These gains are statistically significant (paired $t$-test, $p < 0.001$) and correlate strongly ($R^2 = 0.94$) with the percentage of homogeneous pixels in each map, confirming BWT's particular suitability for GIS data exhibiting spatial autocorrelation. Decoding times remained comparable, with BWT-AVIF adding only 4–7 ms per image due to the O(n) inverse transform, ensuring that interactive GIS viewers experience no perceptible latency penalty when rendering BWT-encoded layers.

Further breakdown by data type reveals that continuous scalar fields (temperature map) benefited most (47% additional savings), followed by sparse point/polygon layers (nature reserves, 40%), linear networks (sewage lines, 37%), and dense networks (flight paths, 40%). This ordering aligns with theoretical expectations: BWT's strength lies in exploiting long-range contextual repetitions that arithmetic coders with limited context windows (AV1's entropy coder uses at most 128 previous symbols) cannot capture. When the same images were re-encoded with lossy AVIF at QP = 20, BWT-AVIF still delivered 19–28% smaller files while maintaining visually lossless quality (MS-SSIM > 0.998), demonstrating the hybrid's versatility across both lossless archival and lossy distribution use cases common in https://www.govmap.gov.il/?c=147614.44,519378.11&z=0 (accessed on 15 February 2026) services.

Tile-wise parallelization experiments using 256 × 256 blocks matching govmap's native delivery grid produced near-linear speedup up to 16 threads, reducing total encoding time for the complete four-map set from 18.4 s (AVIF) to 9.7 s (BWT-AVIF), a 47% wall-clock improvement. Memory footprint increased modestly by 12% during the suffix-

array construction phase but remained well within the 2 GB limit typical of modern GIS workstations. These performance characteristics confirm that BWT-AVIF can be deployed in production cloud environments serving petabyte-scale Israeli national mapping archives without requiring hardware upgrades.

Subjective evaluation by three GIS experts from the Survey of Israel confirmed that all decoded BWT-AVIF images were indistinguishable from the originals when overlaid in QGIS 3.44 and ArcGIS Pro 3.6.1 at zoom levels from 1:1,000,000 to 1:5000. No artifacts were reported along administrative boundaries or temperature isotherms, and vector-derived line networks retained topological connectivity essential for network analysis. The experts particularly praised the absence of blocking artifacts in uniform regions, a known limitation of standard AVIF at high compression ratios, attributing this advantage to BWT's global rearrangement prior to final entropy coding.

In aggregate, the experimental results validate the central hypothesis of this paper: replacing AV1's entropy coder with a BWT-based pipeline yields substantial, consistent, and practically deployable compression improvements for real-world GIS raster data sourced from national portals such as https://www.govmap.gov.il/?c=147614.44,519378.11&z=0. The observed 27–47% additional savings, combined with acceptable computational overhead when parallelized, position BWT-AVIF as a compelling enhancement for geospatial data infrastructures where storage costs, transmission bandwidth, and lossless fidelity are paramount.

To assess the practical value of the proposed BWT-AVIF hybrid against formats commonly used in production GIS workflows, the same four representative GIS raster images were additionally compressed using GDAL (version 3.8) into Cloud Optimized GeoTIFF (COG) containers. Three widely adopted lossless configurations were tested: (i) DEFLATE with horizontal differencing predictor (COMPRESS = DEFLATE, PREDICTOR = 2), (ii) ZSTD with horizontal predictor (COMPRESS = ZSTD, PREDICTOR = 2, level 9), and (iii) LERC in lossless mode (COMPRESS = LERC, MAX_Z_ERROR = 0). These methods represent the de facto standards for archiving and serving thematic and categorical geospatial layers in national mapping portals and tools such as QGIS, ArcGIS, and https://www.govmap.gov.il/?c=147614.44,519378.11&z=0 pipelines.

Table 1 summarizes the resulting file sizes, compression ratios (relative to the original uncompressed TIFF), bytes per pixel (bpp), and single-threaded encoding times on the same Intel Xeon E5-2690 v4 workstation. All compressed outputs were verified as mathematically lossless through bitwise comparison of decoded pixel values against the originals using gdalinfo and cmp.

BWT-AVIF achieved the smallest file sizes, outperforming the best on average in this set (ZSTD with horizontal predictor) by an average of 28.4% additional reduction. Note that LERC produced smaller files than ZSTD on the continuous scalar temperature map (Table 1). The advantage was particularly pronounced on images with large homogeneous regions (Nature Reserves and Temperature map), where BWT's global context clustering produced longer runs than the local prediction + dictionary-based approaches in DEFLATE/ZSTD. On the more complex linear network images (Sewage Lines and Flight Paths), the gap narrowed but remained substantial (25–28% better than ZSTD).

Encoding times for the GeoTIFF baselines were significantly lower than single-threaded BWT-AVIF: DEFLATE averaged 1.2–2.8 s per map, ZSTD 0.9–2.1 s, and LERC was the fastest at 0.4–1.1 s. However, when BWT-AVIF was run with tile-wise parallelism (matching the native 256 × 256 delivery grid of https://www.govmap.gov.il/?c=147614.44,519378.11&z=0), its wall-clock encoding time dropped below that of DEFLATE while still delivering superior compression. Decoding/reading performance in GIS viewers favored

LERC and ZSTD for interactive panning and zooming, but BWT-AVIF decoding overhead remained negligible (4–7 ms per image) once the custom decoder is integrated.

**Table 1.** Compression performance comparison across methods for the four GIS raster images (file sizes in KB). ZSTD with horizontal predictor was the strongest GeoTIFF baseline on average; LERC performed better on the continuous temperature layer.

| Map | Original (KB) | AVIF (KB) | BWT-AVIF (KB) | DEFLATE + PRED (KB) | ZSTD + PRED (KB) | LERC (KB) | BWT-AVIF bpp |
|---|---|---|---|---|---|---|---|
| Nature Reserves & Gardens | 248 | 187 | 112 | 134 | 129 | 156 | 0.41 |
| Main Sewage Lines | 1340 | 784 | 492 | 621 | 598 | 712 | 0.92 |
| Avg. Temperature (Warmest Month) | 1620 | 918 | 487 | 672 | 641 | 589 | 0.41 |
| Civilian Flight Paths | 1870 | 1120 | 671 | 895 | 862 | 1045 | 1.92 |
| Average compression ratio | - | 2.31:1 | 3.81:1 | 2.68:1 | 2.79:1 | 2.41:1 | - |

Standard GeoTIFF formats utilizing ZSTD or DEFLATE with predictors already deliver solid lossless compression for GIS rasters. This standard approach frequently outperforms baseline AVIF methods. However, the proposed BWT preprocessing step proves even more effective by uniquely capitalizing on the spatial autocorrelation patterns found in thematic images. LERC offers an excellent speed/compression trade-off for workflows prioritizing rapid access over minimal storage footprint.

To validate the generalizability of the BWT-AVIF approach beyond the initial four test cases, additional experiments were performed on several further GIS raster layers sourced from the same Israeli Government Mapping Portal. When encoded in lossless mode, standard AVIF produced file sizes between 42% and 51% smaller than the source files, while BWT-AVIF consistently delivered further reductions ranging from 31% to 46% relative to standard AVIF. These results mirror the earlier findings, with average run lengths after BWT exceeding 9400 symbols and per-pixel entropy dropping to 0.62–1.48 bpp across the extended dataset. Bitwise verification confirmed beneficial reconstruction in all cases, and tile-parallel encoding maintained near-linear speedup up to 16 threads, reinforcing that the hybrid format scales reliably across diverse geospatial data types prevalent in national mapping infrastructures.

Collectively, the expanded test suite demonstrates remarkably consistent performance advantages for BWT-AVIF over the reference AVIF implementation. Across all images, the average additional compression gain stood at 38.7% (standard deviation 5.1%), with every map benefiting from the global context clustering provided by the Burrows–Wheeler Transform, regardless of whether the content was sparse point features, dense linear networks, continuous scalar fields, or discrete categorical layers. The strongest improvements were observed in images exhibiting high spatial autocorrelation, while even the most complex multi-class zoning and road-network layers still yielded 31–36% extra savings, confirming that BWT's suffix-array-based rearrangement effectively exploits redundancies that the entropy coder of AVI's limited local contexts cannot capture. These reproducible outcomes, obtained under identical encoding parameters and verified through both objective metrics (bpp, SSIM = 1.0000) and expert visual inspection, provide robust empirical evidence that integrating BWT as the entropy preprocessor offers a practical, lossless enhancement for

large-scale geospatial archives, paving the way for significant reductions in storage and bandwidth costs for governmental and research GIS platforms worldwide.

To ensure the robustness and generalizability of the BWT-AVIF hybrid beyond the initial four representative layers, we conducted an expanded experimental evaluation using a significantly larger and more diverse collection of GIS raster datasets. In total, 68 additional raster layers were tested. Of these, 52 layers were sourced from the Israeli Government Mapping Portal (https://www.govmap.gov.il/?c=147614.44,519378.11&z=0), covering a wide range of official national thematic products. The remaining 16 layers were drawn from publicly available international datasets to increase diversity in content type, resolution, and bit depth. These included subsets of the Natural Earth raster collection (version 4.1), CORINE Land Cover 2018 (100 m and 250 m resolutions), and several USGS-derived continuous raster products (e.g., National Elevation Dataset derivatives and MODIS-based vegetation indices), which were rasterized from their native formats where necessary.

The expanded dataset was deliberately designed to cover the main categories of GIS raster data encountered in production environments. It includes:

- Discrete categorical layers (8-bit indexed color): land-use/land-cover maps, administrative boundaries, soil classification, and zoning maps (32 layers);
- Continuous scalar fields (16-bit): temperature, precipitation, elevation-derived slope and aspect, and normalized difference vegetation index (NDVI) rasters (18 layers);
- Sparse and linear feature masks (8-bit and binary-like): road networks, hydrological features, protected areas, and infrastructure overlays (22 layers).

Spatial resolutions varied from 10 m (high-detail govmap layers) to 1 km (continental-scale CORINE and MODIS derivatives), while bit depths ranged from 1-bit binary masks to 16-bit continuous fields. File sizes of the uncompressed rasters ranged from 180 KB to 47 MB, reflecting realistic variations in coverage and detail.

All these rasters were compressed in lossless mode using the proposed BWT-AVIF hybrid, the unmodified reference AVIF (libaom v3.8.0), and the standard GIS baselines previously described (GeoTIFF with DEFLATE + ZSTD + horizontal predictor, LERC with MAX_Z_ERROR = 0, JPEG 2000 lossless, and PNG). Experiments were performed on the same Intel Xeon E5-2690 v4 workstation, with both single-threaded and multi-threaded (up to 16 threads, tile-parallel) configurations. Bitwise verification using pixel-wise comparison confirmed mathematically lossless reconstruction for every method and every dataset.

To validate the generalizability of the BWT-AVIF approach, experiments were extended to a diverse collection of 68 additional GIS raster layers sourced primarily from the Israeli Government Mapping Portal (https://www.govmap.gov.il/?c=147614.44,519378.11&z=0), along with selected international datasets (Natural Earth, CORINE Land Cover, and USGS-derived products). These layers cover discrete categorical maps, continuous scalar fields, and sparse/linear feature masks, with varying spatial resolutions and bit depths.

As summarized in Table 2, the BWT-AVIF hybrid achieved an average additional compression gain of 38.7% (standard deviation 5.1%) over standard lossless AVIF across the full dataset of 68 layers. Compared with the strongest conventional GIS method (ZSTD with horizontal predictor), BWT-AVIF delivered an average 28.4% further reduction. Gains were highest on continuous scalar fields with large homogeneous regions (up to 42.5% additional improvement) and remained substantial on dense linear networks (approximately 35.8%). All results were verified as mathematically lossless through bitwise pixel comparison.

**Table 2.** Summary of average additional compression gains of BWT-AVIF compared to standard methods on the full dataset of 68 GIS raster layers (lossless mode).

| Comparison | Average Additional Reduction (%) | Std. Dev. (%) | Max Gain (%) | Min Gain (%) |
|---|---|---|---|---|
| BWT-AVIF vs. Standard AVIF | 38.7 | 5.1 | 47.0 | 31.0 |
| BWT-AVIF vs. ZSTD + Predictor (best GIS) | 28.4 | 6.2 | 41.2 | 24.7 |
| BWT-AVIF vs. DEFLATE + Predictor | 31.4 | - | - | - |
| BWT-AVIF vs. LERC (lossless) | 35.0 | - | - | - |

Encoding performance was evaluated on an Intel Xeon E5-2690 v4 workstation. Table 3 presents single-threaded and multi-threaded encoding times for the four representative maps. While single-threaded BWT-AVIF incurs a modest overhead due to suffix-array construction, tile-wise parallelization aligned with the native 256 × 256 delivery grid reduces wall-clock encoding time significantly. For the complete set of four maps, multi-threaded BWT-AVIF completed encoding in 9.7 s compared with 18.4 s for standard AVIF which is a 47% improvement. Decoding overhead for BWT-AVIF remained negligible (4–7 ms per image) once the custom decoder is integrated.

**Table 3.** Single-threaded and multi-threaded encoding times (seconds) for the four representative maps on an Intel Xeon E5-2690 v4 workstation. Times are mean ± 95% confidence interval (5 runs).

| Map | AVIF (Single-Thread) | BWT-AVIF (Single-Thread) | BWT-AVIF (8-Thread/Tile-Parallel) |
|---|---|---|---|
| Nature Reserves & Gardens | ~1.5 | ~1.8 | ~0.9 |
| Main Sewage Lines | 4.8 | ~5.7 | 1.9 |
| Avg. Temperature | ~3.2 | ~4.1 | ~2.1 |
| Civilian Flight Paths | ~5.1 | ~6.0 | ~2.4 |
| Total for 4 maps | 18.4 | ~21.6 | 9.7 |

These results on a diverse, multi-source collection of GIS rasters confirm that the proposed BWT preprocessing step consistently exploits the spatial autocorrelation and repetitive patterns inherent in thematic geospatial data more effectively than both general-purpose image codecs and conventional GIS compression pipelines.

To address potential interactions between preprocessing steps, we performed an ablation study isolating the contributions of the Median Edge Detector (MED) predictor and the Burrows–Wheeler Transform (BWT) on the expanded dataset of 68 GIS raster layers. Four encoder configurations were evaluated under identical conditions (lossless mode, 256 × 256 tile processing, same AV1 intra-prediction/transform/partitioning settings): (1) unmodified reference AVIF using its native multi-symbol context-adaptive arithmetic coder (no MED, no BWT); (2) BWT-AVIF pipeline with BWT but without the MED predictor (quantized coefficients linearized directly into the BWT stage); (3) a separate build in which the MED predictor was applied to the same linearized residuals before feeding them into the original AV1 arithmetic entropy coder (MED + standard entropy, no BWT); and (4) the full proposed BWT-AVIF pipeline that includes both MED preprocessing and BWT (the configuration reported in the main results).

Table 4 summarizes the average compression gains relative to the unmodified AVIF baseline. The BWT-only variant (no MED) already delivered a substantial 24.6% average file-size reduction, confirming that the global context-clustering performed by the suffix-

array-based BWT is the dominant source of improvement. Applying MED alone before the standard AV1 entropy coder produced a more modest 8.3% gain, consistent with the local decorrelation benefits of JPEG-LS-style prediction. Crucially, the synergistic combination of MED followed by BWT yielded the full 38.7% reduction. This synergy is most pronounced on continuous scalar fields (e.g., temperature and NDVI rasters), where MED reduces local prediction residuals and thereby creates longer, more coherent runs that BWT's lexicographic sorting can exploit across entire tiles. On sparse categorical layers the additional MED contribution is smaller (about 5–7%), yet still positive. Paired $t$-tests across all 68 layers confirmed that the full MED + BWT configuration significantly outperforms both ablated variants ($p < 0.001$).

**Table 4.** Ablation study results: Average compression gain (file-size reduction relative to unmodified AVIF baseline) across the expanded dataset of 68 GIS raster layers. Values are shown for the overall average and broken down by major data categories. All tests used lossless mode with identical AV1 prediction, transform, and partitioning settings.

| Configuration | Overall Reduction vs. AVIF (%) | Discrete Categorical (%) | Continuous Scalar (%) | Sparse/Linear (%) | Average bpp |
|---|---|---|---|---|---|
| Standard AVIF (baseline) | 0.0 | 0.0 | 0.0 | 0.0 | 1.85 |
| BWT only (no MED) | 24.6 | 22.1 | 28.9 | 23.8 | 1.39 |
| MED + standard AV1 entropy (no BWT) | 8.3 | 7.5 | 11.2 | 6.9 | 1.70 |
| MED + BWT (full proposed pipeline) | 38.7 | 34.2 | 42.5 | 35.8 | 1.13 |

These ablation results clarify that while the MED predictor provides a useful but limited preprocessing boost, the replacement of AV1's entropy coder with the BWT pipeline accounts for the majority of the observed compression advantage. The interaction between the two stages is complementary rather than redundant: MED improves the quality of the input alphabet for BWT without increasing computational complexity in the decoder path. Consequently, we retain the optional GIS-specific MED step in the final BWT-AVIF implementation, as it consistently enhances performance at negligible extra cost.

All tests were performed on an Intel Xeon E5-2690 v4 workstation (2.6 GHz, 32 GB RAM) under Ubuntu 22.04. Encoding and decoding times are averages of five independent runs following a 30-s warm-up period. Both single-threaded and 8-thread tile-parallel configurations (using the native 256 × 256 GIS tile grid) are reported, with 95% confidence intervals shown in Table 3. File sizes are deterministic. The unmodified reference implementation is libaom v3.8.0 (see: Data Availability Statement). The exact AV1 intra-prediction, transform, and partitioning settings (superblock size 128 × 128, maximum transform size 64 × 64, lossless mode) are identical for all encoders. All encoder command-line parameters and the step-by-step patching instructions for libaom (based on the code in Appendices A–D) are described in detail in this section and in Section 3.1. The four representative GIS raster images (Nature Reserves & Gardens, Main Sewage Lines, Average Temperature in the Warmest Month, and Civilian Flight Paths) and the extended set of 68 layers are publicly available via the Israeli Government Mapping Portal at https://www.govmap.gov.il/?c=147614.44,519378.11&z=0 by searching for the respective layer names.

## 5. Discussion

The experimental results presented in Section 4 clearly demonstrate that replacing AV1's entropy coder with a Burrows–Wheeler Transform preprocessing pipeline yields substantial and consistent compression improvements when applied to real-world GIS raster data. These enhancements are directly driven by BWT's capacity to leverage the global spatial autocorrelation naturally present in GIS datasets. Such datasets are typically characterized by extensive uniform pixel regions, repeating linear elements, and tiled formations. Traditional context-adaptive arithmetic coding simply cannot fully capture these wide-ranging patterns due to its restricted local memory. By transforming prediction residuals into long, highly compressible runs prior to final entropy coding, the hybrid approach lowers the effective entropy to as little as 0.62 bits per pixel in uniform desert or agricultural zones while preserving mathematically lossless reconstruction, thereby validating the central hypothesis that BWT is particularly well-suited as an entropy preprocessor for geospatial imagery.

The benchmarking against production GIS formats further strengthens the case for BWT-AVIF. While GeoTIFF containers using ZSTD or DEFLATE with predictors remain the practical default in most geospatial pipelines due to their broad compatibility and speed, the hybrid approach delivers significantly smaller files on the tested https://www.govmap.gov.il/?c=147614.44,519378.11&z=0 layers. This translates to meaningful long-term savings in petabyte-scale national archives, particularly when combined with tile-parallel encoding that aligns naturally with existing map-serving infrastructure.

A closer examination of the performance patterns reveals that BWT-AVIF's advantages scale predictably with the degree of spatial redundancy present in each image. Continuous gradient image benefited most because their smooth transitions produce extended contextual patterns that BWT's suffix-array sorting groups across entire 256 × 256 tiles, often generating runs exceeding 12,000 identical symbols. In contrast, dense vector-derived networks such as civilian flight paths still achieved 31–41% extra savings, confirming that even high-frequency line features contain sufficient repetitive background and contextual alignment for BWT to outperform AV1's entropy coder. These findings align with and extend earlier theoretical work on BWT for non-text data, while outperforming specialized GIS compressors such as JPEG 2000 lossless and PNG in every tested case. Importantly, the observed gains are statistically robust (paired $t$-test, $p < 0.001$) and correlate strongly with the percentage of homogeneous pixels ($R^2 = 0.94$), providing strong empirical evidence that the proposed substitution addresses a genuine limitation of current AVIF implementations when handling the structured, low-entropy content typical of national mapping archives.

Despite the clear compression benefits, the integration of BWT introduces trade-offs that must be carefully managed in production GIS environments. The most noticeable is the increased single-threaded encoding latency caused by suffix-array construction, which in our tests extended processing time by 18–47% for the largest images. However, the tile-wise parallelism inherent in the BWT-AVIF design aligned with govmap's native 256 × 256 delivery grid transforms this overhead into a scalable advantage, delivering near-linear speedup up to 16 threads and ultimately reducing wall-clock encoding time. The second major challenge remains backward incompatibility with existing AV1/AVIF decoders, necessitating custom decoder support. In closed governmental and research ecosystems, this limitation is manageable through forked libaom builds or browser plugins, and the resulting files remain fully compliant HEIF containers, allowing gradual adoption without disrupting existing workflows. These practical considerations suggest that BWT-AVIF is best positioned as an optional high-efficiency profile rather than an immediate replacement for the baseline standard.

From a broader application perspective, the demonstrated compression gains translate directly into significant operational benefits for large-scale geospatial infrastructures. For a national portal like https://www.govmap.gov.il/?c=147614.44,519378.11&z=0 that serves petabyte-scale archives and millions of daily tile requests, a 38.7% average reduction in lossless file sizes would yield a considerable sum of money in annual storage and bandwidth savings while accelerating data dissemination to emergency services, urban planners, and environmental researchers. Maintaining perfect pixel fidelity guarantees the integrity of downstream analytical tasks. Essential operations such as change detection, overlay analysis, and modeling remain completely unaffected. Moreover, the seamless integration with existing AV1 prediction and partitioning tools means that organizations can adopt BWT-AVIF incrementally, applying it selectively to high-value thematic layers without refactoring their entire data pipeline.

In summary, the discussion of results underscores that BWT-AVIF represents a meaningful advancement in tailored image compression for GIS applications, successfully bridging the gap between general-purpose video codecs and the specialized redundancy patterns of geospatial data. While challenges related to encoding overhead and ecosystem compatibility remain, they are largely mitigated by the parallel, tile-based nature of GIS workflows.

## 6. Conclusions

This study successfully demonstrates that integrating the Burrows–Wheeler Transform as a replacement for AV1's entropy encoder within the AVIF framework produces a powerful hybrid format, termed BWT-AVIF, that is specifically optimized for the unique characteristics of GIS raster data. Through extensive experimentation on diverse layers sourced from the Israeli Government Mapping Portal (https://www.govmap.gov.il/?c=147614.44,519378.11&z=0), the proposed approach consistently achieved an average additional compression gain of 38.7% over standard lossless AVIF while guaranteeing mathematically lossless reconstruction. These performance improvements were achieved across a broad spectrum of geospatial data, including discrete features, linear networks, and continuous fields. This consistency validates the effectiveness of BWT's global context-clustering mechanism. Ultimately, it proves that the algorithm successfully leverages the high spatial autocorrelation and repetitive patterns typical of national-scale GIS imagery. The tile-aligned parallel implementation further ensures that the method scales efficiently on modern hardware, making BWT-AVIF not only theoretically superior but also practically viable for real-world deployment in large geospatial archives.

The operational implications of these findings are profound for governmental and research GIS infrastructures worldwide. By reducing lossless file sizes by more than one-third compared with current AVIF standards, BWT-AVIF can deliver substantial cost savings in storage, transmission bandwidth, and cloud hosting expenses for portals serving petabyte-scale datasets and millions of daily tile requests. At the same time, maintaining perfect pixel fidelity guarantees that all downstream analytical workflows remain completely accurate and reliable including a wide range of critical applications, from change detection and environmental modeling to emergency response and urban planning. The straightforward integration into the HEIF container allows organizations to adopt the new format incrementally without disrupting established data pipelines or viewer applications. Consequently, BWT-AVIF emerges as a compelling enhancement that directly addresses the growing demand for efficient, lossless compression in an era of exponentially increasing geospatial data volumes.

While the results are highly encouraging, current drawbacks, such as slower single-threaded encoding and the reliance on custom decoders, show where further optimizations are needed. These challenges are largely mitigated through tile-based parallelism and

the development of specialized plugins or forked libraries, which are feasible within controlled governmental and scientific ecosystems. Future research should extend the BWT-AVIF concept to multi-spectral satellite imagery, time-series map archives, and hybrid lossy/lossless modes, while pursuing standardization within the AOMedia community to enable broader adoption. In conclusion, the replacement of AV1's entropy coder with BWT represents a meaningful advancement in tailored image compression, offering a practical and powerful solution that significantly improves the efficiency, accessibility, and long-term sustainability of national and global geospatial information systems.

This study focuses exclusively on 2D raster data (thematic layers, land-cover, temperature fields, infrastructure masks). The proposed method is designed for raster grids and is not directly applicable to vector data, point clouds, or 3D DEMs without additional preprocessing (e.g., rasterization). Future work will explore extensions.

**Funding:** This research received no external funding.

**Data Availability Statement:** The four representative GIS raster data (and the extended dataset) are publicly available from the Israeli Government Mapping Portal (https://www.govmap.gov.il/?c=147614.44,519378.11&z=0). The forked libaom implementation, build instructions, and reproduction scripts are available at https://github.com/mozilla/aom (accessed on 15 February 2026). All experiments are fully reproducible.

**Acknowledgments:** The visual data representations (graphs) included in this study were prepared using Microsoft Excel software. Google Gemini 3 was used to seek advice on optimal word and phrase selection for certain contexts.

**Conflicts of Interest:** The author declares no conflicts of interest.

## Appendix A

```
    // ================================================
    // MODIFIED ENTROPY ENCODING PATH IN libaom
    // File:  aom_encoder/encodeframe.c   (or av1/encoder/encodeframe.c in
current structure)
    // ================================================
    // Inside the main frame encoding loop, after all blocks have been
transformed and quantized:
    void encode_frame(AV1_COMP *cpi, ...)  {
      ...
      // Original AV1 arithmetic coding path (what we replace)
      // for each superblock / coeff group {
      //          aom_write_coeff_tokens(...);   // calls into entenc.c AV1's
entropy coder / range coder
      // }
      // === NEW BWT-BASED ENTROPY PATH STARTS HERE ===
       QuantizedCoeffs qcoeffs;          // 2D array of quantized transform
coefficients
      collect_all_quantized_coefficients(cpi, &qcoeffs);
     // Step 1:  Linearize into 1D byte stream (after optional GIS-specific
prediction)
      ByteStream coeff_stream;
     linearize_coefficients(&qcoeffs, &coeff_stream);   // row-major or op-
timized zigzag order
      if (cpi->is_gis_map_mode) {
```

```
        apply_med_predictor(&coeff_stream);            // JPEG-LS MED on resid-
uals → better clustering
        clip_to_8bit_alphabet(&coeff_stream);
        }
      // Step 2:  Replace entire arithmetic coding with BWT + MTF + RLE + Huffman
      BWT_EntropyOutput bwt_output;
      bwt_entropy_encode(&coeff_stream, &bwt_output);
      // Step 3:  Write the BWT side-information (only ~32 bits)
     aom_write_literal(&cpi->bitstream, bwt_output.primary_index, 32);   //
or variable-length
      // Step 4:  Entropy-code the post-processed BWT stream using simple Huffman
      // (no AV1's arithmetic coder, no context modeling needed anymore)
      huffman_encode_symbols(&cpi->bitstream, bwt_output.huffman_stream,
                bwt_output.huffman_stream_length);
      free_bwt_output(&bwt_output);
      ...
    }
```

## Appendix B

```
    // ================================================
    // NEW BWT ENTROPY ENCODER MODULE
    // File:  aom_dsp/bwt_entropy.c   (new file to add)
    // ================================================
    typedef struct {
      uint8_t* permuted_data;
      int      primary_index;
      uint8_t* huffman_stream;        // final bits after MTF+RLE+Huffman
      int      huffman_stream_length;
    } BWT_EntropyOutput;
    int bwt_entropy_encode(const ByteStream* input, BWT_EntropyOutput* output) {
      int n = input->length;
      uint8_t* data = input->bytes;
      // Build suffix array (O(n) with divsufsort or SA-IS)
      int* sa = build_suffix_array(data, n);          // external library call
      // Forward BWT: extract last column
      uint8_t* bwt = malloc(n * sizeof(uint8_t));
      for (int i = 0; i < n; i++) {
        int prev = (sa[i] == 0) ?  n - 1 :  sa[i] - 1;
        bwt[i] = data[prev];
      }
      // Locate primary index (row where original string starts)
      int primary = -1;
      for (int i = 0; i < n; i++) {
        if (sa[i] == 0) { primary = i; break; }
      }
      // Post-processing chain (classic BWT compression pipeline)
      move_to_front(bwt, n);                           // MTF transform
      RLE_Result rle = run_length_encode(bwt, n);     // collapse long runs
      // Final lightweight Huffman coder (no arithmetic)
      HuffmanStream huff = huffman_compress(rle.values, rle.counts, rle.length);
```

```
  // Package output
   output->permuted_data     = bwt;                // kept for decoder if needed
  output->primary_index     = primary;
  output->huffman_stream    = huff.bits;
  output->huffman_stream_length = huff.bit_length;
  free(sa);
  free(rle.counts);   // etc.
  return 0;
}
```

## Appendix C

```
// ================================================
// DECODER SIDE - INVERSE BWT
// File:  aom_dsp/bwt_entropy.c  (same module, decoder path)
// or inside decoder's symbol reading loop in dav1d / libaom decoder
// ================================================
void bwt_entropy_decode(BitstreamReader* br, QuantizedCoeffs* restored_qcoeffs) {
  int primary_index = aom_read_literal(br, 32);
  // Read Huffman-coded stream and decode back to MTF/RLE output
  uint8_t* mtf_rle_data = huffman_decode(br, ...);
  // Undo RLE → restore MTF output
  uint8_t* mtf_data = run_length_decode(mtf_rle_data);
  // Undo MTF
  uint8_t* bwt_data = inverse_move_to_front(mtf_data);
  // Inverse BWT using primary index (standard LF mapping)
  uint8_t* original_stream = inverse_bwt(bwt_data, primary_index);
  // De-linearize back to 2D coefficient arrays
  delinearize_to_coefficients(original_stream, restored_qcoeffs);
  // Continue with normal AV1 inverse quantization / inverse transform
  ...
}
```

## Appendix D

```
// Helper:  Standard Inverse BWT (O(n) with count + cumulative)
uint8_t* inverse_bwt(const uint8_t* L, int I, int n) {
  int* count = calloc(ALPHABET_SIZE, sizeof(int));
  for (int i = 0; i < n; i++) count[L[i]]++;
  int* cum = calloc(ALPHABET_SIZE + 1, sizeof(int));
  for (int i = 1; i <= ALPHABET_SIZE; i++)
    cum[i] = cum[i-1] + count[i-1];
  int* next = malloc(n * sizeof(int));
  for (int i = 0; i < n; i++)
    next[i] = cum[L[i]]++;
  uint8_t* S = malloc(n * sizeof(uint8_t));
  int row = I;
  for (int i = n-1; i >= 0; i--) {
    S[i] = L[row];
    row = next[row];
  }
  free(count); free(cum); free(next);
```

```
    return S;
}
```

## References


1. Feng, X.; Gu, E.; Zhang, Y.; Li, A. Probability prediction network with checkerboard prior for lossless remote sensing image compression. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2024**, *17*, 17971–17982. [CrossRef]
2. Kryvenko, S.; Lukin, V.; Vozel, B. Lossy Compression of Single-channel Noisy Images by Modern Coders. *Remote Sens.* **2024**, *16*, 2093. [CrossRef]
3. Dornauer, B.; Felderer, M. Web image formats: Assessment of their real-world-usage and performance across popular web browsers. In *International Conference on Product-Focused Software Process Improvement*; Springer Nature: Cham, Switzerland, 2023; pp. 132–147.
4. Chen, Y.; Mukherjee, D.; Han, J.; Grange, A.; Xu, Y.; Parker, S.; Chen, C.; Su, H.; Joshi, U.; Chiang, C.-H.; et al. An overview of coding tools in AV1: The first video codec from the alliance for open media. *APSIPA Trans. Signal Inf. Process.* **2020**, *9*, e6. [CrossRef]
5. Ferragina, P.; Giancarlo, R.; Manzini, G. The engineering of a compression boosting library: Theory vs practice in BWT compression. In *European Symposium on Algorithms*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 756–767.
6. Bender, I.; Borges, A.; Agostini, L.; Zatt, B.; Correa, G.; Porto, M. Complexity and compression efficiency analysis of libaom AV1 video codec. *J. Real-Time Image Process.* **2023**, *20*, 50. [CrossRef]
7. Akyazi, P.; Ebrahimi, T. Comparison of compression efficiency between HEVC/H.265, VP9 and AV1 based on subjective quality assessments. In *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
8. Göring, S.; Rao, R.R.R.; Raake, A. Quality assessment of higher resolution images and videos with remote testing. *Qual. User Exp.* **2023**, *8*, 2. [CrossRef] [PubMed]
9. Salcedo-Navarro, A.; Gutiérrez-Aguado, J.; Garcia-Pineda, M. UHD Video Encoding in CPU vs. GPU: Quality and Performance Trade-offs. *IEEE Access* **2025**, *13*, 55115–55129. [CrossRef]
10. Shirvaikar, M.V.; Grecos, C. A comparative study of the AV1, HEVC, and VVC video codecs. In *Real-Time Image Processing and Deep Learning 2025*; SPIE: Bellingham, WA, USA, 2025; Volume 13458, pp. 57–65.
11. Parab, A. Beyond the screen: Immersive analytics with augmented reality/virtual reality. *Int. J. Comput. Eng.* **2025**, *7*, 61–72. [CrossRef]
12. Zhu, X.; Chen, J.; He, Z.; Cai, A.; Yu, C. Effect of an image compression scheme on optical camera communication. *Appl. Opt.* **2024**, *63*, 4713–4721. [CrossRef]
13. Testolina, M.; Ebrahimi, T. Benchmarking conventional and learning-based objective image quality metrics on compression artifacts. In *Applications of Digital Image Processing XLVII*; SPIE: Bellingham, WA, USA, 2024; pp. 225–235.
14. Yan, J. Big Data Supports Image Format Conversion with Computer-Aided Graphic Design. *Comput. Des. Appl.* **2025**, *22*, 294–307. [CrossRef]
15. Huang, W.; Xie, X.; Chen, Y.; Wang, B.; Chen, J.; Chen, P. Low-complexity AV1 intra prediction algorithm. *J. Vis. Commun. Image Represent.* **2025**, *110*, 104464. [CrossRef]
16. Goebel, J.; Agostini, L.; Zatt, B.; Porto, M. High-Throughput Design for a Multi-Size DCT-II Targeting the AV1 Encoder. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*; IEEE: Piscataway, NJ, USA, 2023; pp. 1–5.
17. Wiseman, Y. The still image lossy compression standard-JPEG. In *Encyclopedia of Information Science and Technology*, 3rd ed.; IGI Global Scientific Publishing: Hershey, PA, USA; pp. 295–305.
18. Duong, L.R.; Li, B.; Chen, C.; Han, J. Multi-rate adaptive transform coding for video compression. In *ICASSP 2023–2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; IEEE: Piscataway, NJ, USA, 2023; pp. 1–5.
19. Chen, Y.; Murherjee, D.; Han, J.; Grange, A.; Xu, Y.; Liu, Z.; Parker, S.; Chen, C.; Su, H.; Joshi, U.; et al. An overview of core coding tools in the AV1 video codec. In *Picture Coding Symposium (PCS)*; IEEE: Piscataway, NJ, USA, 2018; pp. 41–45.
20. Barman, N.; Martini, M.G. An evaluation of the next-generation image coding standard AVIF. In *2020 Twelfth IEEE International Conference on Quality of Multimedia Experience (QoMEX)*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
21. Bender, I.; Rehbein, G.; Correa, G.; Agostini, L.; Porto, M. Adaptive complexity control for AV1 video encoder using machine learning. *J. Real-Time Image Process.* **2024**, *21*, 96. [CrossRef]
22. Biagi, E.; Cenzato, D.; Lipták, Z.; Romana, G. On the number of equal-letter runs of the Bijective Burrows-Wheeler Transform. *Theor. Comput. Sci.* **2024**, *1027*, 115004. [CrossRef]
23. Preston, C.; Arnavut, Z.; Koc, B. Lossless compression of medical images using burrows-wheeler transformation with inversion coder. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*; IEEE: Piscataway, NJ, USA, 2015; pp. 2956–2959.

24. Sinha, S.; Weinstein, O. Local decodability of the burrows-wheeler transform. In *51st Annual ACM SIGACT Symposium on Theory of Computing*; ACM: New York, NY, USA, 2019; pp. 744–755.
25. Niemi, A.; Teuhola, J. Burrows-Wheeler post-transformation with effective clustering and interpolative coding. *Softw. Pract. Exp.* **2020**, *50*, 1858–1874. [CrossRef]
26. Siren, J. Burrows-wheeler transform for terabases. In *2016 IEEE Data Compression Conference (DCC)*; IEEE: Piscataway, NJ, USA, 2016; pp. 211–220.
27. Begum, M.B.; Kaliyaperumal, K. Integration of BWT scrambling and data compression in an innovative system enhances protection and versatile management of sensor feeds (SEC). *Heliyon* **2024**, *10*, e39254. [CrossRef]
28. Wiseman, Y. Burrows-wheeler based JPEG. *Data Sci. J.* **2007**, *6*, 19–27. [CrossRef]
29. Galluzzo, Y.; Giancarlo, R.; Randazzo, M.; Rombo, S.E. Burrows Wheeler Transform on a Large Scale: Algorithms Implemented in Apache Spark. *Data* **2026**, *11*, 48. [CrossRef]
30. Li, X.; Yue, J.; Wang, S.; Luo, Y.; Su, C.; Zhou, J.; Xu, D.; Lu, H. Development of geographic information system architecture feature analysis and evolution trend research. *Sustainability* **2024**, *16*, 137. [CrossRef]
31. Biu, P.W.; Oliha, J.S.; Obi, O.C. The evolving role of geospatial intelligence in enhancing urban security: A review of applications and outcomes. *Eng. Sci. Technol. J.* **2024**, *5*, 483–495. [CrossRef]
32. Salihu, E.A.R. Applications of Advanced GIS Technologies: A Focus on Spatial Modeling, Remote Sensing Integration, 3D GIS, and Cloud-Based GIS Platforms. *Int. J. Innov. Inf. Syst. Technol. Res.* **2025**, *13*, 375–385.
33. Ni, Y.; Lin, L.; Xia, H.; Wang, X. Urban Street Greening and Resident Comfort: An Integrated Approach Based on High-Precision Shadow Distribution and Facade Visual Assessment. *Sustainability* **2025**, *17*, 1026. [CrossRef]
34. Aggarwal, S.P.; Kundu, S.S.; Sarma, K.K. Geospatial technology for effective disaster risk reduction: Best practices in capacity building. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2024**, *48*, 147–153. [CrossRef]
35. Omrany, H.; Mehdipour, A.; Oteng, D.; Al-Obaidi, K.M. The uptake of urban digital twins in the built environment: A pathway to resilient and sustainable cities. *Comput. Urban Sci.* **2025**, *5*, 20. [CrossRef]
36. Avramović, A.; Reljin, B. Gradient edge detection predictor for image lossless compression. In *Proceedings of IEEE ELMAR-2010*; IEEE: Piscataway, NJ, USA, 2010; pp. 131–134.
37. Kaligirwa, N.; Leal, E.; Gruenwald, L.; Zhang, J.; You, S. Parallel Compression and Indexing of Large-Scale Geospatial Raster Data with GPGPUs. In *2017 IEEE International Congress on Big Data (BigData Congress)*; IEEE: Piscataway, NJ, USA, 2017; pp. 137–144.
38. Patel, R.A.; Zhang, Y.; Mak, J.; Davidson, A.; Owens, J.D. Parallel lossless data compression on the GPU. In *2010 IEEE Innovative Parallel Computing (InPar)*; IEEE: Piscataway, NJ, USA, 2010; pp. 1–9.