



Article

Real-Time Multiresolution Management of Spatiotemporal Earth Observation Data Using DGGS

Amir Mirzai Golpayegani ^{*,†} , Mahmudul Hasan and Faramarz F. Samavati [†]

Department of Computer Science, University of Calgary, Calgary, AB T2N 1N4, Canada; mhasan@ucalgary.ca (M.H.); samavati@ucalgary.ca (F.F.S.)

* Correspondence: amir.mirzaigolpayega@ucalgary.ca

† These authors contributed equally to this work.

Abstract: The effective management of spatiotemporal Earth observation data is a significant challenge due to their growing size and scale, geometric distortion, temporal gaps, and restricted access. In this article, we introduce a novel methodology utilizing a Discrete Global Grid System (DGGS) to address a set of challenges related to spatiotemporal data storage with a live updating mechanism, the multiresolution processing of an arbitrary region of interest (ROI) in real time, and the approximation of missing data in a smooth, continuous manner. We use reverse Chaikin subdivision and B-spline curve fitting to handle temporal data gaps, allowing for real-time updates. Additionally, our work presents a triangular wavelet scheme to incorporate a flexible, tensor-based multiresolution storage scheme for spatiotemporal raster data. The case study we present uses data from the RADARSAT Constellation Mission (RCM) of the Canadian Space Agency (CSA). Our system enables the dynamic retrieval and visualization of time-varying data for a user-defined ROI. The obtained results demonstrate that our method ensures high data fidelity while making spatiotemporal data more accessible across various practical applications in Earth observation.

Keywords: satellite imagery; multiresolution; triangular wavelets; Discrete Global Grid Systems (DGGSs); temporal data approximation; geospatial data infrastructure



Academic Editor: Nicola Pergola

Received: 16 December 2024

Revised: 29 January 2025

Accepted: 4 February 2025

Published: 7 February 2025

Citation: Mirzai Golpayegani, A.; Hasan, M.; Samavati, F.F. Real-Time Multiresolution Management of Spatiotemporal Earth Observation Data Using DGGS. *Remote Sens.* **2025**, *17*, 570. <https://doi.org/10.3390/rs17040570>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the realm of Earth observation, satellite imagery plays a pivotal role, providing critical data for environmental monitoring—such as precision agriculture—urban development, and disaster management. As the demand for the accurate and timely observation of the Earth's surface intensifies, the efficient management of the vast volumes of raster data provided by satellites becomes increasingly crucial [1,2]. However, traditional data management systems often fall short in handling the complex, dynamic nature of spatiotemporal data, which in turn inhibits effective decision making and analysis [3,4].

Among the fundamental challenges in the field of Earth observation are the real-time processing and updating of the data, ensuring integrity and reliability. Because data also have a time-varying nature, handling this type of spatiotemporal data in both the spatial and temporal dimensions adds another layer of complexity to the data management process. Moreover, the multiresolution storage and accessibility of high-resolution satellite data are essential, as specific applications may require data at varying levels of detail and different sizes for their ROI. In addition, the accurate approximation of missing data, such as gaps in satellite time-series data due to cloud cover or sensor malfunctions, is crucial to maintaining data continuity and usability. The challenge is to have a method for filling the gaps in

satellite time-series data in a continuous manner that is also computationally efficient. This is particularly critical when the database is supposed to be updated in real time. We need a local data fitting method, which means that the approximation of the missing data at any timestamp should be dependent on only the data points in the vicinity of the missing data. This avoids the need to retrieve the entire time series for approximating the missing data, which is computationally expensive. These challenges are further compounded by the geometric distortions inherent in satellite imagery, which pose the need for preprocessing operations to ensure data consistency. Thus, the need for a robust, scalable, and efficient data infrastructure that can handle the complexities of spatiotemporal data is paramount.

Current methodologies, including those employed by systems like RADARSAT, often exhibit limitations, particularly in their approach to storing Synthetic Aperture Radar (SAR) raster data [5]. Their method involves storing the raw satellite images in large chunks, which lacks consistency across the globe. As opposed to a consistent data structure, which ensures uniform coverage of the Earth's surface without gaps or overlaps, RADARSAT's system stores data in large tiles, which essentially correspond to the original raw images acquired by the satellite. This approach not only makes it difficult to manage data at different resolutions but also results in inefficiencies in data retrieval, as users must manually access and process large tiles to extract the desired ROI at a specific resolution. Thus, the need for a standardized grid, in which the cells are defined uniformly across various resolution levels, ensuring that both coverage and resolution are maintained across the entire globe, is essential. Additionally, for having a multiresolution data structure, such uniformity in the grid is required. In addition, missing data approximation is another challenge that needs to be addressed. Also, the raw data provided by some satellite imagery sources, including RADARSAT, require preprocessing to remove noise and compensate for geometric distortions, which can be time-consuming and require specialized expertise in remote sensing. These evident shortcomings underscore the need for solutions that can fundamentally transform the approach to satellite imagery management.

This article introduces a novel approach that leverages the hierarchical structure of Discrete Global Grid Systems (DGGs) [6,7]. A DGG is integrated with the proposed triangular wavelet scheme, aimed at improving the multiresolution management of raster-based satellite imagery. Discrete wavelet transforms (DWTs) are applied multiple times on 2D images [8,9], decomposing them into coarse and detail coefficients to represent various resolution levels. Similarly, wavelet schemes are adaptable to quad grids by using 1D wavelet transformations, such as Haar wavelets. For Aperture 3 DGGs supporting hexagonal cells, Mahdavi-Amiri utilized a wavelet scheme at alternating resolutions to facilitate rhombus (quad grid) one-to-nine refinement [6]. Unlike quad grids, which can utilize tensor-product grid structures, triangular grids do not easily conform to this method. Here, we introduce a Haar-like wavelet scheme adapted for triangular grids.

We also derive a tile-based approach to store raster data by introducing the concept of tensors, which are low-resolution DGG cells used for representing blocks of data. The flexibility of the tensor resolution and size while adhering to the consistent DGG structure is a key feature. For example, in urban planning and precision agriculture applications, the ROIs are usually small, where using high-resolution tensors is feasible, unlike those in environmental monitoring. We use a hierarchical run-length indexing scheme based on a hierarchical space-filling curve on a triangular grid to store the tensor data. This approach leads to a multiresolution storage model, allowing for flexible and scalable access at various resolutions.

Moreover, our approach incorporates a data approximation method using local B-spline curve fitting [10], which effectively fills the gaps in satellite time-series data. We leverage reverse Chaikin subdivision to generate a set of control points that represent

the original data points [11]. The local support of the B-spline curve, together with the localized least squares optimization inherent in reverse Chaikin subdivision, facilitates a real-time updating mechanism that ensures data reliability while maintaining continuous availability in storage. Our methodology is designed to address the challenges associated with managing large volumes of spatiotemporal data, providing a flexible solution for Earth observation applications.

In summary, the main contributions of this article are as follows:

- Consistent, DGGS-based data infrastructure: Utilizing a DGGS to provide a consistent, hierarchical data organizer, enabling the enhanced management of large-volume, high-resolution satellite data; addressing the inefficiencies of the large-tile storage method; and facilitating flexible access to any ROI at various resolutions.
- Integration of a triangular wavelet scheme in the DGGS grid: Facilitating the multiresolution management of raster-based satellite data for efficient storage and retrieval at various levels of detail by using a tensor-based approach for dividing the grid into manageable image-like chunks.
- Missing data approximation and real-time updating mechanism: Utilizing reverse Chaikin subdivision and local B-spline curve fitting to approximate missing data in a continuous manner, ensuring data continuity; plus, incorporating a method for updating and accessing the time-series data in real time.
- RADARSAT case study: Demonstrating the effectiveness of our methodology by using data from the RCM.

The remainder of this article is organized as follows: Section 2 provides an overview of the foundational concepts and prior work that underpin our methodology. Section 3 outlines the methodology for the data storage and data retrieval components, including the preprocessing, encoding, and compression of satellite imagery data. Section 4 presents a case study using data from the RCM, demonstrating the effectiveness of our methodology in managing spatiotemporal data. Finally, Section 5 concludes the article and discusses future work in this area.

2. Background and Prior Work

In this section, we overview the foundational concepts and prior work crucial to our methodology, highlighting the key concept of the DGGS and its role in managing spatiotemporal Earth observation data. We review related work on DGGS-based data management, multiresolution techniques, time-series analysis using B-spline curve fitting and reverse subdivision rules, and real-time storage systems, forming the basis for our contributions.

2.1. Discrete Global Grid System (DGGS)

DGGSs have revolutionized how spatial data are structured, queried, and visualized, providing a consistent and hierarchical framework ideal for a wide array of geospatial applications [12]. Initially conceptualized from Platonic solids, geodesic DGGSs set a foundational paradigm for spatial data management, enabling enhanced computational efficiency and structured data integration across various scales and dimensions [13]. Figure 1 illustrates an example of the refinement of an icosahedral DGGS grid from resolution 0 to 5. The foundational concept of the DGGS, thoroughly reviewed and expanded in seminal works [7,14], underpins modern digital Earth platforms that integrate DGGSs to support complex Earth observation data infrastructures [7,14,15]. Furthermore, the adoption of the Open Geospatial Consortium (OGC) standards for DGGSs ensures uniformity and interoperability among diverse geospatial systems, promoting widespread application and development [16].

Building on this solid foundation, DGGs address challenges posed by the volume, variety, and velocity of big geospatial data, where traditional methods struggle [17]. The structured, multiscale nature of DGGs is effectively utilized in applications such as spatiotemporal modeling [18], enhancing data handling and analytical processes. Sherlock et al. introduced a novel web-based multiresolution Digital Earth system that leverages GPU acceleration for the real-time rendering and dynamic styling of large multiscale geospatial datasets, enabling efficient multilevel focus+context visualization and interactive data styling [19]. In a similar vein, Mahdavi-Amiri et al. developed the Atlas of Connectivity Maps (ACM), optimizing the management of semiregular models by organizing connectivity within DGGs, thus supporting a range of multiresolution frameworks [20]. Additionally, DGG applications in environmental monitoring [21], agricultural management [22], and low-altitude flight conflict detection [23] showcase their broad utility and versatility in contemporary geospatial analysis.

Further enhancements in DGGs are driven by the integration of big Earth data with cloud and high-performance computing technologies, which significantly boost real-time data processing capabilities [4,24]. In the context of big geospatial data, parallelized data storage is recommended, with DGG cell indexing supporting efficient spatial positioning and enabling distributed storage and parallel computing. DGG integration with cloud-based high-performance computing enhances their ability to handle large-scale data analysis effectively [25]. Developments within aperture 3 hexagonal DGGs include advanced indexing and wavelet transforms, improving data retrieval and transmission efficiency [6]. Additionally, addressing Geographic Information science (GIScience) challenges through robust data integration strategies is crucial to advancing DGG frameworks [26]. This collective body of work emphasizes the adaptability of DGGs across different configurations, significantly enhancing the management and analysis of complex geospatial datasets [12].

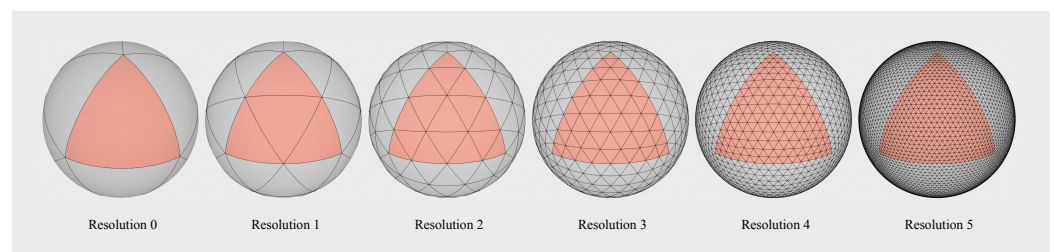


Figure 1. The refinement of the icosahedral DGG grid from resolutions 0 to 5. The selected triangle is one of the base cells of the DGG.

2.2. Spatiotemporal and Big Data Management

By integrating spatiotemporal data management within DGGs, Yao et al. and Xia et al. proposed a unified framework that enhances the handling of big Earth observation data (BEOD) by utilizing cloud computing and optimizing DGG capabilities [2,27]. This framework, as discussed in [27], refines DGG models and enhances cloud computing performance, improving data retrieval through an efficient spatiotemporal indexing structure that integrates spatial and temporal data. By addressing further technical challenges, Zhou et al. innovatively proposed a new encoding method for time within DGGs that resolves issues of non-uniform time scales and code redundancy [18], building on the foundational work by Tong et al. [28]. Furthermore, Sahr et al. explored using regular polyhedra for creating hierarchical, multiresolution partitions, using icosahedrons to minimize distortion through effective spherical transformations [13]. Huang et al. detailed a framework for managing agricultural remote sensing data to improve precision agriculture, focusing on optimizing the storage and analysis of high-resolution satellite imagery [1].

The practical applications of these theoretical advancements are demonstrated in diverse settings. Koubarakis et al. utilized semantic web technologies to enhance data accessibility and integration for applications such as precision farming [29]. Simultaneously, Rathore et al. and Pekturk et al. emphasized the importance of real-time and offline big data processing architectures in remote sensing, applying advanced computational strategies like filtration, load balancing, and parallel processing [3,30]. These strategies facilitate immediate data analysis and decision making, crucial to dynamic environmental monitoring, showcasing the broad utility and adaptability of DGGs in enhancing the efficiency and responsiveness of Earth observation systems.

2.3. Multiresolution Analysis and Wavelets

The integration of wavelets into DGGs and image processing underscores significant advancements in spatial data management and analysis. Hebert et al. presented a method using two-dimensional Haar wavelets on triangular grids for image encoding [31]. This facilitates efficient compression and rendering, crucial to enhancing the multiresolution representation of satellite imagery in DGGs grids. Similarly, Fujinoki et al. enhanced this approach by introducing two-dimensional nonseparable biorthogonal wavelets on a triangular lattice, offering isotropic image representations with hexagonal symmetry that improve image processing efficiency and data compression [32].

Moreover, the foundational work by Stollnitz et al. highlights the adaptability of wavelets in computer graphics, facilitating efficient geospatial data processing [33]. This is further supported by Chakrabarti et al. and Garofalakis et al., who explore the utility of wavelets for approximate query processing and creating probabilistic synopses, respectively, thus enhancing data management efficiency [34,35]. Additionally, by addressing the challenge of missing data in time-series satellite imagery due to factors like cloud cover, Samavati et al. utilized reverse subdivision combined with least squares data fitting to approximate missing data in satellite time series, a method that refines control points for B-spline curves and consequently improves data handling [10]. The work by Samavati et al. on local filters for B-spline wavelets aligns with these efforts, offering an innovative approach to maintaining smoothness and efficiency in multiresolution operations [36]. Li et al. and Vitter et al. extended these applications into data mining and the management of sparse data by using the data cube operator, advancing data analysis within relational databases and DGGs frameworks [37–39].

Building on advanced wavelet techniques, our methodology, applied within an icosahedral DGGs's triangular grid, introduces a novel multiresolution storage strategy. This approach supports dynamic data analysis, allowing for the seamless integration of various resolution levels. Thus, our method highlights the potential of multiresolution techniques to enhance data representation and analysis in applications from environmental monitoring to urban planning. This significantly improves the usability and accessibility of Earth observation data, making them more adaptable to diverse applications.

2.4. Missing Data and Temporal Approximation

Missing data in time-series satellite imagery, often due to cloud cover or sensor malfunctions, necessitate temporal approximation techniques like B-spline curve fitting to ensure continuous representation of the Earth's surface. Amirfakhrian et al. presented a variational-based method to refine cloud-contaminated satellite images by using spatiotemporal Poisson blending, offering strategies for addressing missing data across varying scales [40].

This research builds on foundational works such as the constrained and Biarc curve optimization methods by Rogers et al. and Ong et al. [41,42], which enhance model accu-

racy across varied applications. Piegler et al.'s methods for seamless data transitions and Ateshian's surface fitting techniques show flexibility in computer-aided design and geospatial analysis [43,44]. Additionally, the integration of temporal and spatial data management benefits from the TIMESAT program by Jönsson et al. [45], which boosts operational capabilities. The innovative use of reverse subdivision by Bartels et al. [11] and its practical applications by Sadeghi et al. [46] underpin our methodological approach. Additionally, Samavati et al. utilized reverse subdivision combined with least squares data fitting to approximate missing data in time series, a method that refines control points for B-spline curves and consequently improves data handling [10]. Recent studies like those by Lan et al. and Zhou et al. further underscore the challenges and advances in time-series data reconstruction, particularly through the use of deep learning and other advanced computational techniques [47,48]. However, while machine learning techniques provide innovative solutions for data reconstruction, they often require access to the entire time-series dataset to approximate a single missing point, which can be inefficient. In contrast, local approximation methods offer a more streamlined approach, enabling reconstructions based on just a few recent data points. This local focus significantly enhances efficiency, especially in scenarios where real-time data processing and minimal computational overhead are crucial. Thus, our method for missing data approximation is specifically based on the local reverse Chaikin subdivision scheme presented by Bartels et al. [11]. While the original work focuses on deriving regular local reverse subdivisions for curve and surface fitting, providing some visual examples, it does not explore specific practical use cases that showcase the benefits of the locality of these curve-fitting schemes. In this article, we present an important practical case study that leverages this local optimization method to reduce the number of retrievals for a single missing data point, which is crucial to real-time data processing [49].

2.5. Satellite Imagery Storage Systems

Satellite imagery storage systems are designed to manage large volumes of raster data collected by Earth observation satellites. These systems are essential to storing, indexing, and retrieving satellite imagery, enabling users to access and analyze data for various applications.

Satellite imagery from missions like the Earth Observing System (EOS), the Copernicus Programme, the Sentinel series, and the RADARSAT Constellation Mission (RCM) has been foundational in Earth observation. These missions provide critical data across various domains, such as climate monitoring, urban development, and disaster response. For instance, the RCM continues the advancements of its predecessor, RADARSAT, enhancing capabilities in flood mapping, sea ice classification, and wetland monitoring [5,50–53]. Similarly, missions like the EOS and the Copernicus Programme have been pivotal to providing data for environmental and land monitoring studies, as detailed in recent analyses [54,55].

Satellite data platforms such as NASA Earthdata, Sentinel Explorer, Earth Observation Data Management System (EODMS), and Google Earth Engine (GEE) have significantly advanced access to and the analysis of Earth observation data. NASA Earthdata offers a comprehensive archive with robust capabilities for data searching and access, facilitating research and application development. Sentinel Explorer provides focused access to Sentinel mission data, enhancing user interaction with satellite imagery and supporting systematic observations of the Earth system through the Sentinel-1, -2, and -3 missions, which are vital to monitoring the ocean, cryosphere, and land [56]. The EODMS manages RADARSAT data, storing and handling large, inconsistent tiles that require manual preprocessing to address noise and geometric distortions. Similarly, GEE revolutionizes geospatial analysis with its cloud-based platform, providing extensive access to satellite imagery and analytical

tools. GEE employs a tile-based storage system using 256×256 pixel tiles for efficient data management and utilizes multiresolution pyramids to downsample images for rapid visualization, though it also focuses heavily on time-series data crucial to monitoring temporal changes. The RADARSAT Constellation Mission (RCM) and its data handling via the EODMS are pivotal to providing imagery crucial to a variety of monitoring and analytical tasks [51,57,58].

However, these platforms often exhibit limitations in terms of multiresolution storage, time-series data management, and consistency in data representation and storage, which are critical to effective data handling. More specifically, the large-tile storage method lacks consistency and efficiency, making it challenging to manage data at different resolutions and access data for specific ROIs at varying levels of detail. Also, the gaps and overlaps in data coverage due to the large-tile storage method, like the one used by RADARSAT or GEE, make it difficult to ensure data consistency across the globe and at various resolutions. Additionally, the manual preprocessing required for RADARSAT data to address noise and geometric distortions is time-consuming and requires specialized expertise in remote sensing. These limitations underscore the need for a more efficient, scalable, and consistent data infrastructure that can handle the complexities of spatiotemporal Earth observation data. Our work introduces a novel methodology that leverages the hierarchical structure of Discrete Global Grid Systems (DGGs) to provide a consistent, hierarchical data infrastructure, addressing these limitations and enhancing the management of spatiotemporal Earth observation data.

3. Methodology

In this section, we describe the comprehensive methodology employed in this work to store and manage spatiotemporal Earth observation data by using a DGGs-based data infrastructure. We start by explaining the main pipeline for storing data with the acquisition and preprocessing of raw satellite imagery. Next, we elaborate on the preprocessing operations applied on the data so that they are converted into a DGGs-based format. Afterwards, we have a detailed discussion on the encoding process, which involves mapping the image data into the DGGs cells, as containers of data values. Next, we go over how the time-series data are represented in our spatiotemporal data storage, taking advantage of reverse Chaikin subdivision and B-spline curve fitting. Next, we cover the details of our triangular wavelet scheme, which is used for representing the data at various resolutions. Finally, we see how the resulting wavelet coefficients are compressed and stored in our database for later retrieval and access. The following subsections describe the methodology in detail.

3.1. Data Acquisition

In this work, we focus on the acquisition and preprocessing of Earth observation data from the EODMS platform. However, to support a wide range of applications, our methodology is designed to be adaptable, capable of handling data from multiple satellite data sources, including but not limited to the RCM. This mission provides high-resolution SAR data, essential to monitoring environmental changes, land use, and natural disasters. The RCM comprises three satellites, each equipped with a variety of imaging modes and bands with different spatial resolutions and penetration capabilities, suitable for diverse applications in Earth observation [51].

By automating the data acquisition process, we guarantee a consistent and regular supply of data, which is crucial to time-series analysis. Requesting the new data available on a daily basis is essential since satellite image providers usually update their data repositories regularly. This process involves setting up a data pipeline that automatically

downloads the latest data from the satellite providers and preprocesses them. By aligning these processes with the specific needs of our data infrastructure, we ensure that the data are always current and readily available, leading to a real-time updating mechanism that is essential to dynamic Earth observation applications.

3.2. Raw Data Preprocessing

Earth observation data, regardless of the source, often require several preprocessing steps to adjust for spatial and temporal inconsistencies and enhance data quality for further analysis. These preprocessing operations are tailored based on the specific characteristics of the data source. To apply these, we leverage the Sentinel Application Platform (SNAP) of the European Space Agency (ESA), an open-source toolbox equipped with Python APIs for automating the preprocessing steps, making the process scalable and adaptable to different data sources, including the RCM [59]. For the SAR data from the RCM, preprocessing involves the following key operations:

- Radiometric calibration: We apply this operation to correct for sensor noise and inherent inconsistencies in signal strength across the image, ensuring uniform brightness and enhancing data reliability [60].
- Speckle noise reduction: We use this operation since it is essential to improving image clarity and mitigates the grainy appearance that is characteristic of coherent radar images, making the data more visually interpretable [61].
- Geometric/terrain correction: This operation aligns the imagery with geographical coordinates to correct positional distortions [62]. This part of our preprocessing is crucial for accurate mapping into a DGGS-based format later in the pipeline.
- Reprojection: Lastly, we use this operation to transform the data from the sensor's native projection to a standard map projection, ensuring consistency across different datasets and facilitating integration with other geospatial data encoded into the DGGS. We use the WGS84 projection as the destination projection since it is the most commonly used projection in geospatial applications.
- Polarimetric decomposition: We apply this operation to extract detailed metrics such as entropy and anisotropy, essential for accurate land cover classification and environmental assessments. This operation is crucial to enhancing the data's utility for advanced applications [63].

After preprocessing, the preprocessed data are stored in the GeoTIFF format, which is a widely used format for storing geospatial raster data [64], ready to be encoded into the DGGS grid cells, as the next step in our methodology. The preprocessing operations are essential to ensuring data consistency, quality, and compatibility with the DGGS-based data infrastructure, enabling the efficient storage and retrieval of Earth observation data.

3.3. Encoding Data into DGGS

This process involves mapping the preprocessed satellite imagery onto the hierarchical cell structure of the DGGS. Our DGGS uses an icosahedron as the base polyhedron, with triangular faces that are subject to one-to-four refinement at each level. The benefit of using an icosahedral DGGS with triangular cells is that it provides a simple and consistent representation of the Earth's surface [65,66]. Also, triangles generally offer several advantages: they support efficient indexing and congruent refinement and are well suited for compatibility with rendering software and hardware [12]. This configuration provides a simple, efficient, consistent grid structure with immediate hardware support that is ideal for the efficient management of massive high-resolution spatiotemporal data. The cells are indexed by using a barycentric coordinate system, which provides a unique identifier for each cell across all the resolutions, which allows for efficient triangular operations [12]. We define

target resolution as the resolution of our grid at which we encode the data without losing any information. This resolution is chosen based on the spatial resolution of the original satellite imagery, ensuring that the data are encoded at an appropriate level of detail.

The main challenge of managing large volumes of raster data across the globe at different resolutions is the need for a consistent database structure. On top of the consistent, hierarchical cell structure of the DGGS, we need to have a method for dividing the cells into large-enough manageable chunks, called *tensors*. These tensors facilitate data organization and wavelet compression in storage in later steps. Tensors, depending on the application, can be defined at different resolutions, which we call the *tensor resolution*. The tensor resolution is chosen based on the target resolution. The *tensor size* is determined by the number of cells at the target resolution contained in each tensor. It is crucial that we choose a tensor resolution that results in an appropriate tensor size, ensuring efficient storage and retrieval of the data. If the tensor size is too large, it may lead to slower retrieval times. On the other hand, if the tensor size is too small, it may result in numerous tensors, which can also impact storage and retrieval efficiency. Figure 2 illustrates two different examples of tensor choices for the same target resolution. Depending on the size of our ROI and the application, we have the flexibility to adjust the tensor resolution to optimize the storage and retrieval performance. For instance, for an environmental monitoring application with a large ROI, we may choose a lower tensor resolution to reduce the number of tensors and improve retrieval efficiency. In contrast, for a precision agriculture application with a small ROI, we may choose a higher tensor resolution to ensure that the data are divided into reasonably sized tensors.

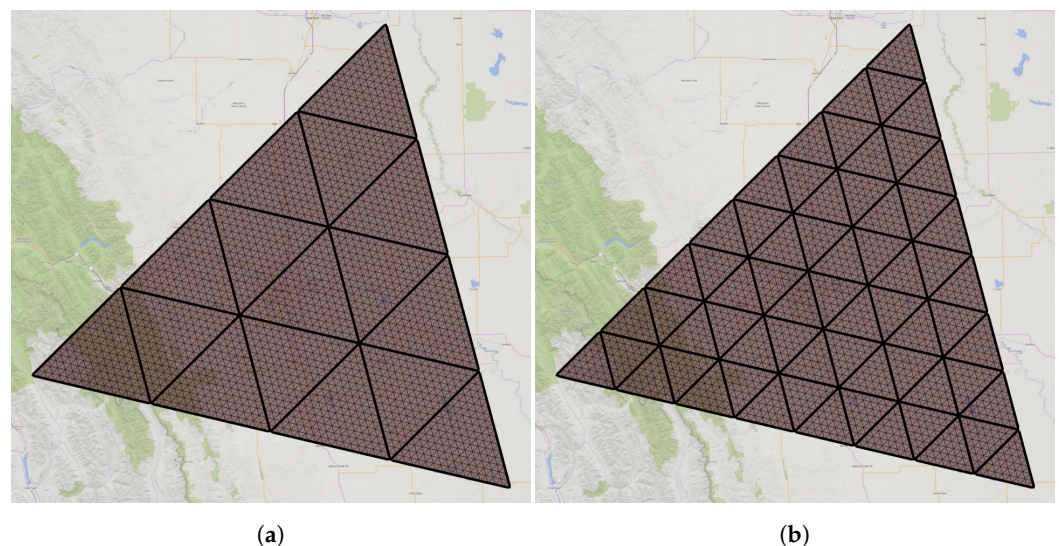


Figure 2. A large region in Alberta, Canada, covering the city of Calgary and surrounding areas, encoded at target resolution 11, with two different examples of tensor choices: (a) the tensor resolution is 7, which results in a tensor size of 256, and (b) the tensor resolution is 8, which results in a tensor size of 64.

Inspired by how images are stored and compressed, we aim to use tensors to divide the DGGS cells into smaller chunks, treating each tensor as a block of data. Within each of these blocks, we separately apply the required processes for encoding, compression, and storage. This tensor-based approach allows us to organize data into manageable chunks, facilitating structured storage and retrieval. While the concept of tensors can generally be defined in the context of DGGS-based data storage, processing, and retrieval, specific implementation details may vary depending on the grid type and application. In our methodology, we define triangular tensors within the triangular grid of the DGGS we use, which is an icosahedral DGGS with one-to-four refinement. However, for other grid types,

such as hexagonal grids, tensor implementation may vary. This tensor-based approach is a key component of our methodology, enabling the division of data into well-defined, dynamic, manageable chunks.

We summarize some definitions and terminologies used in our tensor-based approach in our methodology as follows:

- ROI: Region of interest, a specific area in the DGGS grid that is of interest for analysis. This region is initially defined by the user as a boundary specified by a polygon or vector data on the Earth's surface.
- Target cells: Triangle cells in the DGGS grid, identified by their unique DGGS index, representing an ROI.
- Target resolution: The resolution, in the DGGS grid, at which the data are encoded into the DGGS grid. This resolution is chosen based on the spatial resolution of the original data to be encoded.
- Tensor: A cell of relatively low resolution in the DGGS grid used for representing blocks of data. Each tensor consists of a set of DGGS cells at the target resolution. Since the tensors are cells in the DGGS grid, they are also identified by their unique DGGS index.
- Tensor resolution: The resolution, in the DGGS grid, at which the data are divided into tensors.
- Tensor size: The number of cells at the target resolution contained in each tensor.
- Query resolution: The resolution, in the DGGS grid, at which the data are retrieved from the DGGS grid. This resolution is chosen based on the level of detail required for analysis.

Sampling the preprocessed satellite imagery is the main step in the encoding process. Depending on the target resolution, we have a set of DGGS cells that cover the region defined by the boundary of each preprocessed GeoTIFF image. We call these cells the *target cells*. As the result of the sampling process, we want to have a mapping of target cells to values at the sampling points. This mapping forms the basis for the encoding process, allowing us to efficiently store the data in a DGGS-compatible format. For this purpose, we first find the set of target cells by applying a geocoding method. The geocoding method transforms pixelated image data into the discrete triangular grid of the DGGS. More specifically, given the boundary of the preprocessed image, represented as a spherical polygon or vector data, we find all the DGGS cells, identified by their unique DGGS indices, that cover the region defined by that boundary. In this case, the cells are found at the target resolution.

Next, as illustrated in Figure 3, we sample the preprocessed image at the centroid of the triangles in the target cells, assigning the value at each sample to the corresponding target cell. Since the preprocessed image is in a latitude–longitude domain, i.e., the WGS84 coordinate system, we transform the sampling points from a spherical 3D coordinate system to the latitude–longitude coordinate system. This transformation is then followed by another transformation into the 2D texture coordinates of the preprocessed image. These two transformations are shown in Figure 3. By using nearest-neighbor interpolation, we assign the value at each sample to the corresponding target cell. This process is repeated for all the target cells, resulting in a mapping of target cells to values at the sampling points, forming the basis for the encoding process. The pseudocode for this process is shown in Algorithm 1.

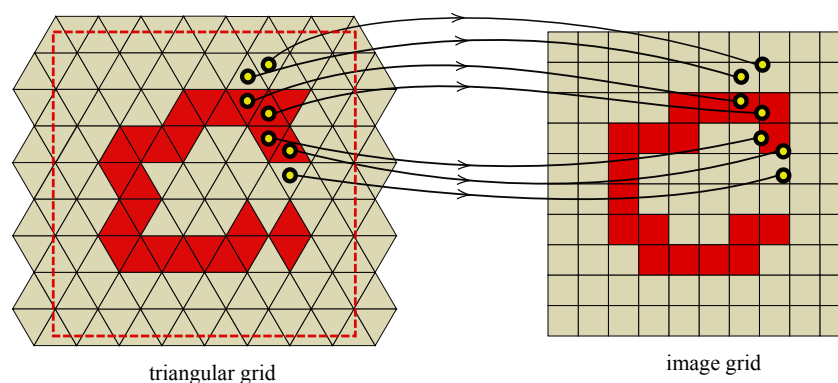


Figure 3. An example sampling for the target cells at the target resolution. The sampling points are picked by finding the centroids of the triangles in the triangular grid. Next, each sampling point is transformed into its corresponding pixel in the image grid. Finally, the values of the sampled pixels are assigned to their corresponding triangles.

Algorithm 1 Sampling the preprocessed image for encoding into DGGS cells.

- 1: **Input:** *preprocessedImage* and *targetCells*
 - 2: **Output:** Mapping of *targetCells* to values at sampling points
 - 3: Rasterize the *preprocessedImage* region to find the *targetCells*.
 - 4: **for** *targetCell* in *targetCells* **do**
 - 5: Find the centroid of the triangle in the *targetCell*.
 - 6: Transform the centroid to latitude–longitude coordinates.
 - 7: Transform the latitude–longitude coordinates to texture coordinates.
 - 8: Assign the value at the texture coordinates to the *targetCell*.
 - 9: **end for**
-

3.4. Triangular Wavelet Scheme

In this section, we introduce the triangular wavelet scheme, which is a key component of the multiresolution management of the data in our DGGS-based data infrastructure. First, we provide an overview of the wavelet decomposition and reconstruction steps. Next, we describe the hierarchical run-length indexing scheme used for applying the wavelet transform to the data. Finally, we discuss the storage and compression strategy.

3.4.1. Wavelet Decomposition and Reconstruction

Wavelet transformations are straightforward to derive for Cartesian grids, such as those used in standard 2D image processing [67]; however, adapting these transformations to triangular grids, such as those employed in our DGGS, poses significant challenges. The irregularity and hierarchical structure of triangular grids complicate the application of standard wavelet techniques, which are typically optimized for orthogonal coordinate systems. Addressing this, our article introduces a novel wavelet scheme tailored for triangular grids. This scheme efficiently compresses data while preserving the inherent hierarchy of the grid, allowing for effective storage and retrieval of data at multiple resolutions. We adapt the simplicity of the Haar wavelet mask to suit triangular cells in our hierarchical, DGGS-based grid, employing a one-to-four refinement strategy, where each cell is divided into four sub-cells at each level of refinement.

Figure 4 shows the wavelet decomposition and reconstruction steps for two resolution levels, r and $r + 1$, where r is an integer number starting from zero. Starting with the higher level of resolution, $r + 1$, we have four cells that are the children of a single cell at the lower level of resolution, r . The values assigned to these four cells are denoted by f_1 , f_2 , f_3 , and

f_4 . For representing these grid values at a lower level of detail or resolution, we apply the decomposition step to obtain one coarse value, c , and three detail values, d_1 , d_2 , and d_3 :

$$\begin{aligned} c &= \frac{f_1 + f_2 + f_3 + f_4}{4}, \\ d_1 &= f_1 - c, \\ d_2 &= f_2 - c, \\ d_3 &= f_3 - c, \end{aligned} \quad (1)$$

where c is the coarse value and is the average of the four initial fine values and each detail value, d_i , is the difference between the coarse value and one of the fine values. The resulting coarse value, c , also called the scaling coefficient, represents the original data, f_1 , f_2 , f_3 , and f_4 , at a lower level of resolution. By intuition, that is the average of the four fine values, which is a good approximation of the original data at a lower level of resolution. Also, the detail values, d_1 , d_2 , and d_3 , are called the wavelet coefficients. The coarse and the detail values form this wavelet transform.

The wavelet scheme is proven to be invertible, meaning that we can reconstruct the original values from the wavelet coefficients by applying the reconstruction step. This way, we do not lose any information in the compression process. The reconstruction step, as shown in the equations

$$\begin{aligned} f_1 &= c + d_1, \\ f_2 &= c + d_2, \\ f_3 &= c + d_3, \end{aligned} \quad (2)$$

and $f_4 = c - d_1 - d_2 - d_3$,

simply computes the values of the four fine cells based on the coarse and detail values. The equations are derived from the decomposition equations by solving them for the fine values.

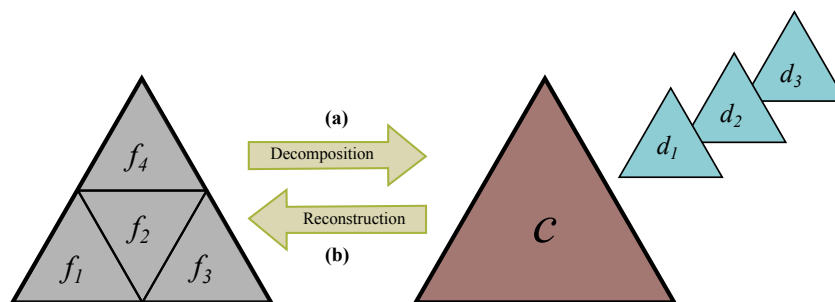


Figure 4. The DGS-based triangular wavelet scheme shown in one level of decomposition and reconstruction between the higher level of resolution, $r + 1$, and the lower level of resolution, r . The decomposition step (a) is applied to the cells in the higher level of resolution, f_1 , f_2 , f_3 , and f_4 , to obtain the coarse value, c , and the detail values, d_1 , d_2 , and d_3 . The reconstruction step (b) is applied to the coarse and detail values to retrieve the original values.

As Figure 4 shows, for any two consecutive resolution levels in the grid, we can apply the decomposition step on the cells at the higher resolution level to obtain the coarse approximation and wavelet coefficients and then apply the reconstruction step to retrieve the original values. By decomposing the data into coarse approximation and details, we manage to store the data in a multiresolution format, in the form of a wavelet transform. Therefore, we can store the data at different levels of resolution in the same tensor without having redundant data. This way, we can efficiently retrieve the data at the desired resolution by applying the reconstruction step to the wavelet transform for the required number of levels.

How exactly the values are stored in the database requires an array-like representation of the wavelet transform for each tensor cell. We are inspired by the way images are stored in a raster format, where the values are stored in a row-major order. We store the wavelet transform in a similar fashion, where the coarse value is stored first, followed by the detail values. To apply this decomposition step multiple times, at each level, we store all the coarse values first in the array, followed by all the detail values. For the next level, we apply the same process to the coarse values, and so on. Figure 5 shows this process applied twice to a tensor of size 16. This way, we can easily retrieve the values in the same order, and apply the reconstruction step to retrieve the original values. The scaling and wavelet coefficients are then compressed by using a lossless compression algorithm, in the TIFF format [68], which leverages efficient compression techniques to reduce storage requirements and enable faster retrieval of the data.

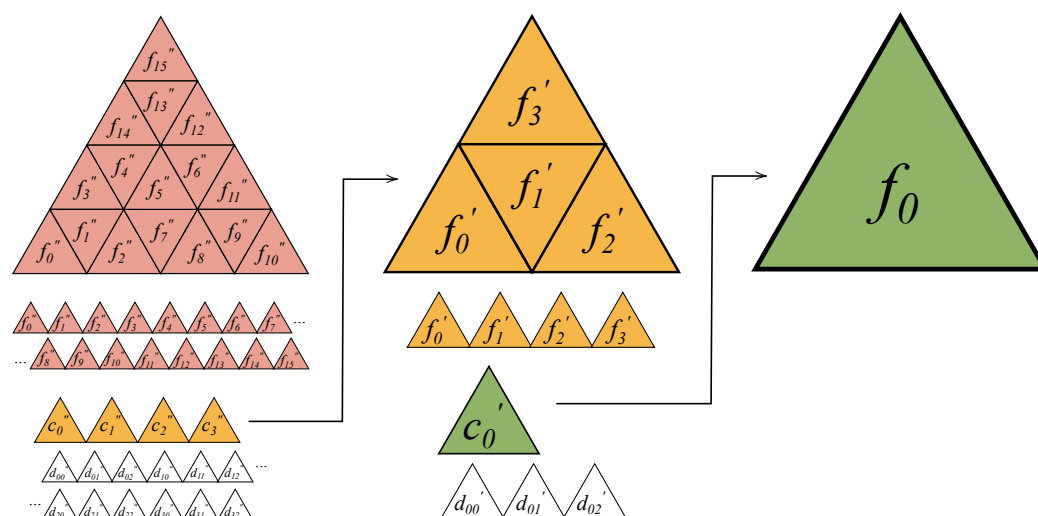


Figure 5. An example of applying two levels of wavelet decomposition to a tensor of size 16. After the first level, we have 4 coarse values and 12 detail values. By using Equation (1), each four consecutive cells are turned into one coarse value and three detail values, e.g., f_0'' , f_1'' , f_2'' , and f_3'' are turned into c_0'' , $d_{0,0}'$, $d_{0,1}'$, and $d_{0,2}'$. The coarse value c_0'' is also interpreted as the value of the parent cell at the lower level of resolution, i.e., f_0' . The same process is repeated for the second level of decomposition, resulting in 1 coarse value and 3 detail values.

3.4.2. Hierarchical Run-Length Indexing

The advantage of having tensor-based storage is recognized as we utilize a hierarchical run-length indexing scheme for the cells inside each tensor. We notice how a space-filling curve defined on a two-dimensional Cartesian grid, as shown in Figure 6, can help us order all the pixels in a fixed specific order. Similarly, we can define a space-filling curve on the triangular grid of the DGGS, which passes through all the cells in a fixed order. This space-filling curve can be similar to the Hilbert curve on a quad grid. As explored in Chapter 6 of the thesis by Mahdavi-Amiri [69], Goodchild et al. utilized Dutton's indexing method in a triangular grid, with a one-to-four refinement scheme, by modifying the ordering of children [70,71]. We utilize the same hierarchical indexing scheme to achieve the hierarchical space-filling curve. We can benefit from the hierarchical property of such indexing scheme. This hierarchical property facilitates applying our wavelet scheme efficiently by ordering the triangle cells in a one-dimensional sequence where each four consecutive cells are the children of the same parent cell at the next lower level of resolution. Furthermore, the run-length property of this indexing scheme enables us to store the indices of the cells in a more compact way, which is crucial for better compression ratios. More specifically, space-filling curves enhance image data compression and storage compactness

by preserving the locality of data points, thereby improving compression efficiency and reducing storage space [72].

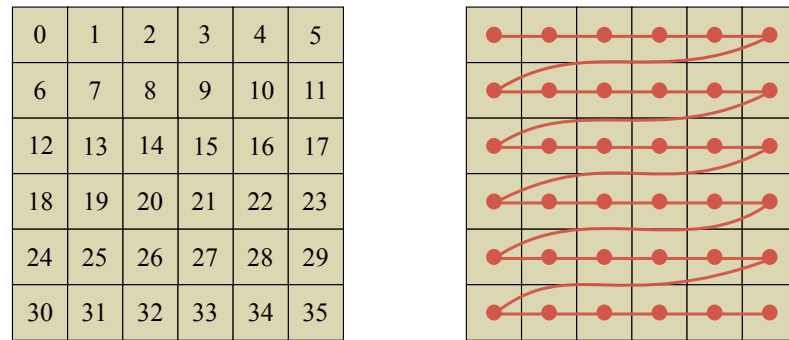


Figure 6. An example of a space-filling curve that passes through the pixels of a 2D image, i.e., a Cartesian grid. The curve is defined in a way that it passes through all the pixels in a row-major order.

This hierarchical run-length indexing is defined for each tensor separately, which results in a fixed order of the cells in the tensor. We aim to treat each tensor as a block of data, like a block of pixels in a two-dimensional image, as shown in Figure 6. By using the space-filling curve, we can store the ordered cell values in the tensor instead of keeping the indices of the DGGS cells in that tensor, avoiding storage overhead. Also, since the definition of this indexing scheme for each tensor is fixed, we can retrieve the values in the same order, making the process invertible.

As outlined in Algorithm 2, to apply wavelet decomposition on the target cells within a given tensor, first, we order all the children cells based on the hierarchical run-length indexing. As a result, we have an array of values sorted in the same order in which the corresponding space-filling curve passes through these cells. Since this indexing scheme is hierarchical, we can conclude that each four consecutive cells based on this ordering are the children of the same parent cell at the next lower level of resolution. The next step is to apply wavelet decomposition to these cells, i.e., each four cells in this order are turned into one coarse value and three detail values. To store the resulting coarse and detail values, we simply store the coarse value first, followed by the detail values. See Figure 5 for an example of applying two levels of wavelet decomposition to a tensor of size 16.

Algorithm 2 Applying wavelet decomposition on the target cells in a DGGS tensor to obtain a single coarse value for the tensor.

```

1: Input: targetCells in a tensor, in resolution  $res_{target}$ 
2: Output: Wavelet transform for the tensor, in resolution  $res_{tensor}$ 
3:  $res_{current} \leftarrow res_{target}$ 
4:  $coarseValues, detailValues \leftarrow []$ 
5:  $orderedTargetCells \leftarrow ORDERHIERARCHICALRUNLENGTH(targetCells)$ 
6: while  $res_{current} > res_{tensor}$  do
7:   for  $f_1, f_2, f_3, f_4 \leftarrow$  each group of four consecutive cells in  $orderedTargetCells$  do
8:      $c, d_1, d_2, d_3 \leftarrow APPLYWAVELETDECOMPOSITION(f_1, f_2, f_3, f_4)$ 
9:     Append  $c$  to the end of  $coarseValues$ .
10:    Append  $d_1, d_2, d_3$  to the end of  $detailValues$ .
11:   end for
12:    $res_{target} \leftarrow res_{target} - 1$ 
13:    $targetCells \leftarrow coarseValues$ 
14: end while

```

3.4.3. Compression and Storage

Figure 5 demonstrates an example of applying two levels of wavelet decomposition on a tensor of size 16. The number of levels of decomposition, n , is determined by the difference between the target resolution and the tensor resolution,

$$n = res_{target} - res_{tensor}. \quad (3)$$

As a result of applying wavelet decomposition on each tensor for n levels, we end up with an array of size 4^n . This array consists of two parts: the first element in the array is a coarse value that represents the average value of all the target cells in the tensor, and the rest of the elements are the detail values for all the resolutions from $res_{target} - 1$ to res_{tensor} . As seen in Figure 5, starting with a tensor of size 16, after one level of wavelet decomposition, we obtain 4 coarse values and 12 detail values for resolution $res_{target} - 1$. Next, after the second level of decomposition, we obtain 1 coarse value and 3 detail values for resolution $res_{target} - 2$, i.e., res_{tensor} . This array is eventually saved as a file in the TIFF format, which is a common format for storing raster data. While there are good alternatives for the file format, we use the TIFF format for its simplicity and compatibility with most image processing libraries. The file is then compressed by using a lossless compression algorithm, DEFLATE [68], to reduce the storage requirements and enable faster retrieval of the data. The compressed TIFF file is then stored in our local file-system database for later retrieval and access. The possibility of using a cloud-based storage system or distributed file system is also considered for scalability.

It is crucial to note that each tensor file is assigned a unique identifier (ID) in the file-system database, which is based on the tensor's index in the DGGS. For storing time-series data, in addition to the tensor index, the timestamp of the source satellite image is also considered when assigning the tensor file ID. Instead of specifically encoding the temporal dimension within the tensor ID, timestamps are mapped to tensor IDs by incorporating a timestamp integer that represents the date and time of image acquisition. For example, a tensor file corresponding to a specific DGGS index might have an ID such as Tensor_12_5756665_1672531200, where 12_5756665 represents the tensor's DGGS index, and 1672531200 is the integer timestamp. This approach ensures that both spatial and temporal dimensions are uniquely identified while keeping the ID format efficient and straightforward.

3.5. Temporal Approximation Method

To deal with the time axis of satellite data, it is desired to approximate the values at the missing timestamps while staying close to the original data values at the existing timestamps. For example, for the specific region covering Lake Minnewanka in Alberta, Canada, we have five RCM satellite images available within a month-long time range at these timestamps: 17 October, 26 October, 3 November, 11 November, and 19 November. If we need an approximation of the data on 7 November, a simple method is linear interpolation. This method is efficient in terms of retrieval, since we only require the values at the two timestamps of 3 November and 11 November, i.e., the two closest data points to 7 November, the one before and the one after it. However, the problem with this method is that it does not provide any compression and the resulting curve is not a smooth approximation of the original data points. Thus, more advanced curve-fitting methods are used to achieve a smooth approximation and a compressed representation of the time-series data. A global least squares problem can be solved to fit a quadratic B-spline curve to the data points. However, this method is computationally expensive in retrieving and updating the data, as it requires to update all the control points for the B-spline curve when adding a new data point. To overcome this limitation while still providing a smooth

and compressed representation of the data, we propose a temporal approximation method using reverse Chaikin subdivision. This subdivision scheme is a technique that finds a set of control points which will generate the second-degree (third-order) B-spline curve that best approximates the original data points. That is in fact the solution to a local least squares problem for fitting a quadratic B-spline curve to the data points [10]. This method not only provides a smooth representation of the data, minimizing the local least squares error, but also allows us to compress the data in the temporal domain as well. By reducing the number of initial data points to the number of resulting control points, we can efficiently store the data in a more compact format.

Algorithm 3 shows the steps for applying reverse Chaikin subdivision to a set of initial data points. The algorithm starts with the initial data points as an array of size $2n - 2$. By applying the reverse Chaikin subdivision mask to the initial data points, we obtain the control points as an array of size n . The mask is defined as a set of weights that are applied to the data points. Figure 7 also shows an example of applying reverse Chaikin subdivision on some time-series data.

Algorithm 3 Applying reverse Chaikin subdivision to a set of initial data points.

```

1: Input: Initial data points  $f_0, f_1, \dots, f_{2n-3}$ 
2: Output: Control points  $c_0, c_1, \dots, c_{n-1}$ 
3:  $c_0 \leftarrow f_0$ 
4:  $c_1 \leftarrow -\frac{1}{2}f_0 + f_1 + \frac{3}{4}f_2 - \frac{1}{4}f_3$ 
5: for  $i = 2$  to  $n - 3$  do
6:    $c_i \leftarrow -\frac{1}{4}f_{2i-2} + \frac{3}{4}f_{2i-1} + \frac{3}{4}f_{2i} - \frac{1}{4}f_{2i+1}$ 
7: end for
8:  $c_{n-2} \leftarrow -\frac{1}{4}f_{2n-6} + \frac{3}{4}f_{2n-5} + f_{2n-4} - \frac{1}{2}f_{2n-3}$ 
9:  $c_{n-1} \leftarrow f_{2n-3}$ 

```

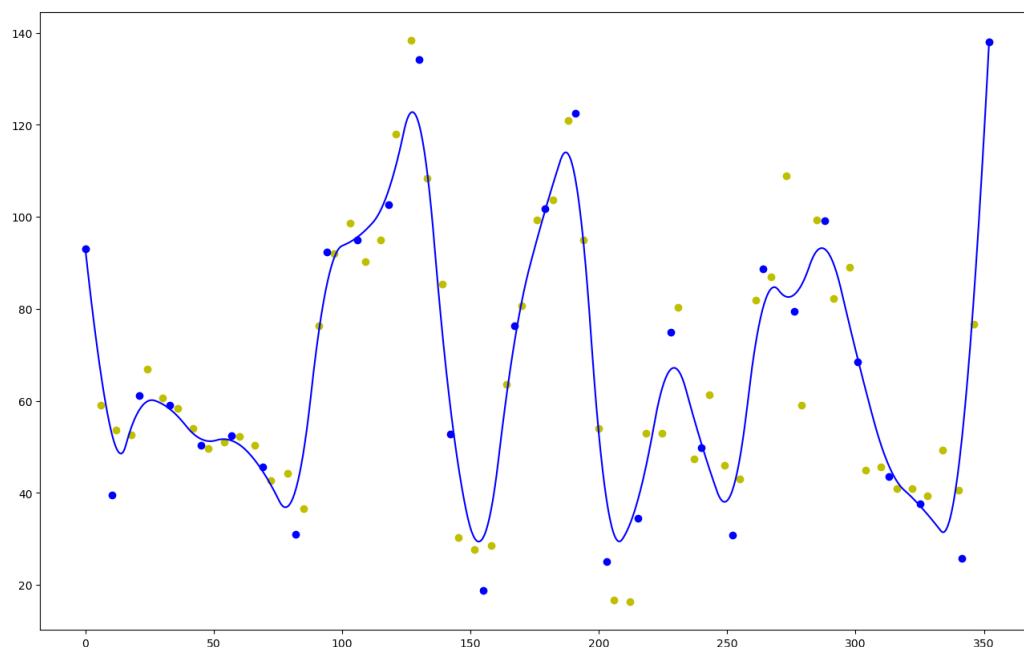


Figure 7. An example plot of applying reverse Chaikin subdivision on a set of yellow points which represent the initial data points. The y-axis represents the intensity of a certain pixel in a satellite image from the RCM throughout the year 2024. The x-axis represents the day of the year for each data point in the time series. By applying reverse Chaikin subdivision, we obtain the blue points, which represent the control points for the quadratic B-spline curve that best approximate the initial data points, i.e., the control points represent the solution to the local least squares problem for fitting a quadratic B-spline curve to the data points.

In addition to the benefits stated above, because reverse Chaikin subdivision solves a local optimization problem, for adding a new data point to the time series, we only need to update the last few control points in the local neighborhood of the new data point. Moreover, the B-spline curve has the local support property, which is inherent in the definitions of its basis functions. This means that the value of a point on the B-spline curve is only dependent on the values of the control points in a local neighborhood of that point. For a quadratic B-spline curve, the value of a point on the curve is only dependent on the values of the three closest control points to that point. This property is crucial to our application, as it allows us to approximate the values at the missing timestamps without having to retrieve the values of all the control points for the B-spline curve, representing the whole time series. This way, we can efficiently approximate the values at the missing timestamps and provide a continuous representation of the data for the user without having to inefficiently load the data for the whole time series going all the way back to the first timestamp for processing any query.

3.6. Real-Time Data Storage Pipeline

As we discussed in the previous subsections, there are several components in our data storage model, including data acquisition, preprocessing, DGGS encoding, triangular wavelet decomposition, and temporal approximation. These components are all used and integrated into a data infrastructure. In this subsection, we present our pipeline for storing the data in this infrastructure for two different scenarios: single-image encoding and time-series data encoding.

3.6.1. Single-Image Encoding

Figure 8 shows the detailed pipeline for encoding the data representing each satellite image in our DGGS-based data infrastructure. First, the image is geocoded to the DGGS cells by using the geocoding method. The image is then sampled at the target resolution to assign values to each of the target cells. Next, the target cells are divided into the tensors according to the tensor resolution. Consequently, based on a run-length hierarchical indexing scheme, the target cell values in each tensor are ordered in a sequence. Afterwards, the wavelet decomposition step is applied on each tensor's ordered values. Finally, the resulting scaling and wavelet coefficients are compressed by using TIFF compression schemes and stored in the database.

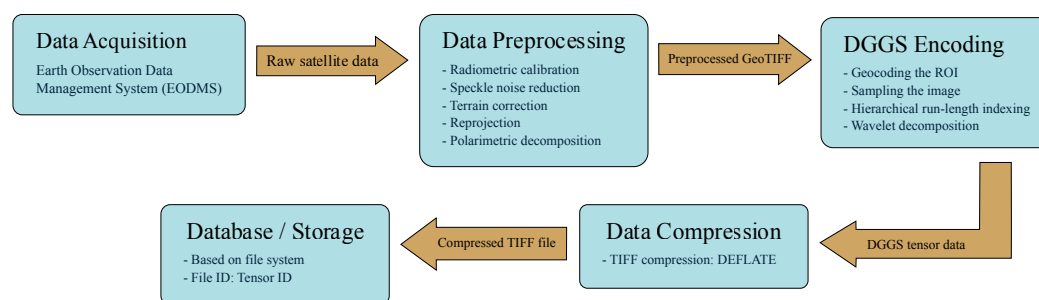


Figure 8. The data storage pipeline consists of five units, starting with the image downloader, which downloads the raw satellite imagery, followed by the preprocessing unit, which preprocesses the data; the DGGS-based encoder, which encodes the data into the DGGS cells by using the sampling method and applying our wavelet decomposition scheme; the data compression unit, which compresses the data by using common TIFF compression schemes; and finally the data storage unit, which stores the compressed data in a database.

As mentioned earlier, the proposed methodology for storing the RCM satellite data in a DGGS-based data infrastructure is designed to work in real time, meaning that the data should be updated regularly at the same rate as they are provided by the satellite data

provider. For that matter, we make sure that the encoding process runs in real time, meaning that it does not cause a delay in the process of data acquisition, that is, the acquired data are encoded and stored in the database before the next instance of data becomes available for encoding. The results in Section 4 support this claim.

3.6.2. Time-Series Data Encoding

If we consider time-series data, the encoding process is slightly more complex than encoding a single image. The main difference is that we need to consider the temporal dimension of the data. The temporal dimension is represented by the timestamps of the satellite images. As shown in Figure 7, we start by making the control points by using Algorithm 3 for representing our B-spline curve that best approximates the initial data points, i.e., each of the initial data points is, in fact, a mapping of target cells to values at each existing timestamp, and each control point is a mapping of target cells to values at some other timestamps derived by the reverse Chaikin algorithm. By applying one level of reverse Chaikin subdivision on the initial data points, we obtain the control points. Next, since each control point is basically a mapping of target cells to values at some timestamp, we can apply the same encoding process as we did for a single image. Thus, the data at each control point are encoded separately into the DGGS-based data infrastructure, as demonstrated in Figure 8 and in Section 3.6.1.

3.7. Data Retrieval

As discussed in the previous subsections, the proposed data infrastructure is designed to store and manage large volumes of satellite imagery data in a DGGS-based format, consisting of several components. These components were integrated into a real-time data storage pipeline. Similar to how, in Section 3.6, we presented the pipeline for storing the data, we now present the pipeline for retrieving the data from the database. We explain two different scenarios: retrieving the data at an available timestamp and retrieving the data at an unavailable timestamp.

3.7.1. Available Timestamp Retrieval

As we have shown that every step in the data storage pipeline, including wavelet decomposition and run-length hierarchical indexing, is invertible, we can retrieve and reconstruct the stored data in the original format. In other words, the data retrieval pipeline is the reverse of the data storage pipeline. A query consists of a spatiotemporal ROI, defined by a spatial region and a timestamp. A query resolution is also specified, which is the resolution of the data to be returned. The spatial region is represented as a set of DGGS cells at the target resolution, which can be obtained by geocoding the boundary of the region the same way we do in the storage pipeline. Next, the tensors that cover the region are identified by using a parent access operation applied on the target cells. The parent access operation is a common operation in the DGGS, which finds the parent cell of a given cell at a lower level of resolution. Consequently, we load the files associated to the identified tensors and the queried timestamp from the database and apply the wavelet reconstruction step to obtain the original values of the target cells. At this stage, we also have the option of applying the wavelet reconstruction step as many times as needed to obtain the cell values at the query resolution, which can be as high as the target resolution or as low as the tensor resolution. Lastly, the reconstructed values are mapped to the cells in the query resolution by using the same space-filling curve that was used in the encoding process. The pseudocode for this process is provided in Algorithm 4.

Algorithm 4 Data retrieval.

- 1: **Input:** Spatial region, Timestamp, Query resolution
- 2: **Output:** Mapping of target cells to values at the query resolution
- 3: Rasterize the spatial region to find the target cells.
- 4: Find the tensors that cover the region.
- 5: **for** tensor in tensors **do**
- 6: Load the tensor file from the database.
- 7: Apply wavelet reconstruction step on the tensor file to obtain the original values.
- 8: **end for**
- 9: Map the reconstructed values to the cells in the query resolution.

3.7.2. Unavailable Timestamp Retrieval

In the case of a missing timestamp, we need to approximate the values at the missing timestamp by using the control points of the B-spline curve, as illustrated in Figure 7. Finding the control points that contribute to the pixel value on the curve at the missing timestamp t is a crucial step. First, we need to determine the span in the knot vector to which t belongs. The *knot vector* is a sequence of parameter values that divides the curve into segments, and the *span* refers to the interval between two consecutive knots in the knot vector. This span is determined by the timestamps of the two closest control points to t , which we have stored in the database. Algorithm 5 shows the steps for approximating the values at the missing timestamp.

Algorithm 5 Approximating the values at the missing timestamp by using B-spline curve fitting.

- 1: **Input:** Knot vector $\{u_0, u_1, \dots, u_{n+3}\}$, control points $\{P_0, P_1, \dots, P_m\}$, parameter u
- 2: **Output:** Value of the curve $Q(u)$ at u
- 3: **Step 1: Identify the knot span**
- 4: Find i such that $u_i \leq u < u_{i+1}$.
- 5: **Step 2: Initialize basis functions**
- 6: Set $N_{j,1}(u) = 1$ if $u_j \leq u < u_{j+1}$, otherwise $N_{j,1}(u) = 0$ for all j .
- 7: **Step 3: Compute basis functions recursively for $k = 2$ and $k = 3$:**
- 8: **for** $k = 2$ to 3 **do**
- 9: **for** $j = i - k + 1$ to i **do**
- 10: Compute:

$$N_{j,k}(u) = \frac{u - u_j}{u_{j+k-1} - u_j} N_{j,k-1}(u) + \frac{u_{j+k} - u}{u_{j+k} - u_{j+1}} N_{j+1,k-1}(u).$$

- 11: **end for**
- 12: **end for**
- 13: **Step 4: Compute the curve value**
- 14: Evaluate:

$$Q(u) = \sum_{j=i-2}^i N_{j,3}(u) P_j.$$

- 15: **Return:** $Q(u)$

4. Results and Discussion

In this section, we present the results obtained from implementing our novel DGGS-based data infrastructure designed for the multiresolution management of time-series satellite imagery in real time. The evaluation focuses on our system's performance in terms of data storage, data retrieval, and real-time processing capabilities, in addition to our system's feature for temporal approximation. We detail the outcomes of our approach in managing large volumes of SAR data from the RCM, highlighting the advantages of

our methods, such as reverse Chaikin subdivision and B-spline curve fitting for temporal data approximation and triangular wavelet schemes for multiresolution data access. These results underscore the effectiveness of the proposed method for the management of satellite imagery in diverse applications. It is crucial to note that the novelty of this work lies in the integration of various features into a comprehensive data infrastructure, including our DGGS-based method for encoding data, our triangular wavelet scheme for the multiresolution storage and retrieval of data, our temporal approximation method using reverse Chaikin subdivision and B-spline curve fitting, a real-time updating mechanism for managing time-series data, and a data retrieval pipeline for accessing the data of any arbitrary ROI at any timestamp. Thus, in this section, we aim to demonstrate the effectiveness of our methodology in managing large volumes of satellite data in real time and show the advantages of our method in terms of data storage, retrieval, and temporal approximation.

Moreover, since there are no direct analogues to our comprehensive approach in the current scientific and technological landscape, including all the contributions that we present in this article, we can only compare some specific aspects of our methodology with some existing satellite data management systems, such as Sentinel Hub and Google Earth Engine (GEE) [4,57,73].

All results presented in this section were obtained on a desktop computer featuring a 12th Gen Intel® Core™ i7-12700K processor, an NVIDIA GeForce RTX 3050 graphics card, and 64 GB of main memory, running a Ubuntu 64-bit operating system.

First, we describe the configuration of the DGGS we use as the core of our data infrastructure. We use an icosahedron as the base polyhedron, with triangular faces that are subject to one-to-four refinement at each level of resolution. The cells are indexed by using a barycentric coordinate system [7], which provides a unique identifier for each triangular cell across all the resolutions via fast DGGS operations, such as hierarchical access, neighbor access, and point-to-cell operation, i.e., geocoding. As explained in Section 3.3, this indexing scheme allows for efficient triangular operations [12]. Figure 9 demonstrates this configuration at resolutions 0, 1, and 2 for a single base cell, which corresponds to a single face of the icosahedron. Although the choice of the projection does not affect the methodology presented in this article, in our implementation, we use the gnomonic projection to map the spherical surface of the Earth to the faces of the icosahedron. We utilize this projection in our geocoding operation, which is used by our rasterization operation, when we want to find the set of DGGS cells covering a given ROI. This operation is a crucial part of our sampling and retrieval process, as explained in Algorithms 1 and 4. As shown by Wecker et al., this projection makes the geocoding operation, which is an important part of our encoding process, considerably faster, compared with an equal-area operation [12]. For interoperability between a hexagonal DGGS and the triangular DGGS we employ, one can implement hierarchical grid conversion methodologies [74].

This section is structured as follows: First, we evaluate the performance of our methodology in managing a large volume of satellite data for a single timestamp. The purpose is to focus on the encoding and multiresolution storage and retrieval capabilities that our method provides. Secondly, we extend our evaluation to time-series data, focusing on the temporal approximation unit and the real-time updating mechanism that our method offers.

To gain a more comprehensive understanding of how our methodology works in practice, refer to the video provided as a Supplementary Material. The video demonstrates various use cases of our real-time spatiotemporal DGGS-based infrastructure at 1x speed. In the video, we first highlight tensor flexibility in terms of resolution and size, which is helpful for handling various scales of ROIs in different applications. Next, we illustrate how an ROI, whether as large as the state of Alberta, Canada, or as small as a city park, is geocoded at

appropriate resolutions. Finally, we demonstrate our system's multiresolution data retrieval feature for an ROI encompassing the west coast of Lake Minnewanka, Alberta, Canada.

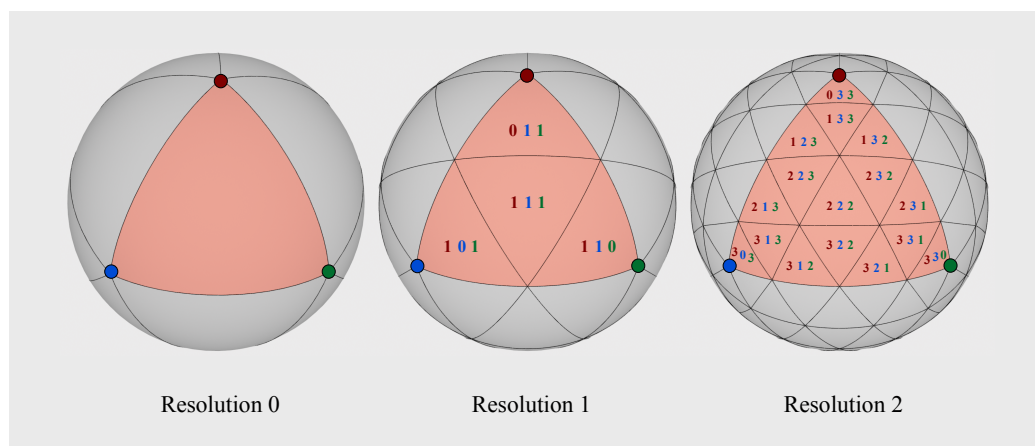


Figure 9. The configuration of the DGGS we use in our data infrastructure. The base polyhedron is an icosahedron with triangular faces, refined in a one-to-four scheme at each level of resolution. The cells are indexed by using a barycentric coordinate system, providing a unique identifier for each cell across all the resolutions. The vertices of a base cell, e.g., the highlighted triangle, are colored in red, green, and blue. Each digit of the index of a child cell is also shown in colors corresponding to its distance from the vertex of the same color.

4.1. Single-Timestamp Data Encoding

In this subsection, we aim to justify our claim that our methodology is capable of managing large volumes of satellite data at a global scale. We choose the region of Alberta, Canada, which is considered a large region, with an approximate area of 661,848 km². Thus, to evaluate the performance of our methodology, the results we provide in this subsection demonstrate the capabilities of our method in processing and storing large volumes of satellite data effectively. More specifically, since our method uses a tensor-based processing and storage approach, it is also applicable to larger volumes of data at a global scale. For taking this even further, utilizing distributed computing and database management systems makes it possible to scale our method to a global level.

For this case study, we acquire the RCM satellite data, which provides high-resolution SAR imagery of our ROI. We also consider a month-long time range, January 2024, to select a single timestamp. This choice allows us to have a dataset that fully covers our ROI. The data are downloaded from the EODMS platform. The images we download are tiles in Alberta in the specified time range and with a large size varying between 500 MB and 2 GB each.

The spatial resolution of the images we download for this case study is 30 m. Therefore, for encoding the data, we choose the target resolution of 18. According to Table 1, at this resolution, the triangular cells most accurately match the pixels of the original satellite images we encode with in terms of the area they represent. Generally, the tensor resolution can be chosen flexibly in the range of 0 to 18, depending on the application requirements. Most importantly, the ROI to be encoded is a key factor in determining the tensor resolution, i.e., larger ROIs require larger tensors. Also, it is inefficient to store a small ROI with a large tensor that covers areas outside the ROI and does not effectively subdivide the region. In addition to this point, a higher tensor resolution results in smaller tensors, which can lead to faster storage and retrieval times, due to faster processing times of a smaller data volume. On the other hand, a lower tensor resolution results in larger tensors, which can lead to more efficient storage, due to the reduced overhead of storing the tensor metadata and better compression ratios. The results in Table 2 confirm this trade-off between storage

space and time efficiency. According to these results, as the tensor resolution is decreased, the storage compression is improved, but at the same time, due to the larger file size, the process for both storing and retrieving each tensor's data becomes slower. To show the data compression improvement, we calculate a metric by dividing the total storage space by the number of cells in the tensor.

Table 1. Comparison of the image pixel area and the approximate DGGS cell area, and the appropriate DGGS resolution equivalent to each image resolution.

Satellite Image Resolution (m)	Pixel Area (m ²)	Appropriate DGGS Resolution	DGGS Cell Area (m ²)
1	1	Res. 22	0.89
5	25	Res. 20	14.33
10	100	Res. 19	57.32
20	400	Res. 18	229.36
30	900	Res. 18	229.36
50	2500	Res. 17	917.88
100	10000	Res. 16	3674.98

Table 2. Storage performance results for a tensor at different DGGS resolutions. The target resolution is set to 18 for these results.

Tensor Resolution	#Target Cells	Storage Time (s)	Storage Space (MB)	Storage Space per Cell (B)
Res. 5	67,108,864	354.109	132.2	1.970
Res. 6	16,777,216	89.740	38.6	2.302
Res. 7	4,194,304	23.523	12.1	2.887
Res. 8	1,048,576	7.280	3.6	3.441
Res. 9	262,144	3.229	1.2	4.561
Res. 10	65,536	2.247	0.317	4.846

Figure 10 also shows different choices for the tensor resolution for representing the region of Alberta, Canada. For the case study, based on the results in Table 2, we choose the tensor resolution of 7, resulting in a good balance between fast tensor processing times and high compression ratios. At this resolution, we have 502 tensors covering our ROI, dividing it into many smaller regions. Each tensor approximately covers an area of 1312 km², and the tensor size is 4,194,304 cells. For downloading the data, we use the EODMS API. More specifically, by sampling the boundary of each tensor, as shown in Figure 11, we can make a shapefile that is passed as an argument to the API as the ROI. At this point, we download the image that has the most overlap with the tensor within our time range. Afterwards, the data are preprocessed by using the SNAP toolbox. The preprocessing operations we apply in this case study are polarimetric decomposition, geometric correction, speckle noise filtering, and reprojection into the standard WGS84 projection.

The performance of our method in terms of storage time is also evaluated in Table 2. The column *Storage Time* shows the time it takes to encode the data of a single tensor at each resolution. This encoding process, as demonstrated in Figure 8, consists of several steps, including the sampling process, the run-length indexing, the wavelet decomposition, and the file compression and storage. Table 3 shows the time it takes for each of these steps for encoding a single tensor. The results show that the sampling process is the most time-consuming step, followed by run-length indexing and wavelet decomposition. It is crucial to note that most of the functionalities in these processes are dependent on the fundamental operations of the DGGS, such as hierarchical access, neighbor access, and point-to-cell encoding, which are implemented efficiently in our DGGS base code [12].

However, further optimizing those operations can lead to faster processing times for the encoding process. All in all, the results in Table 2 show that our method is capable of encoding the data of a single tensor in a reasonable time, which leads to the real-time updating of the database on a regular basis and the management of global-scale data.

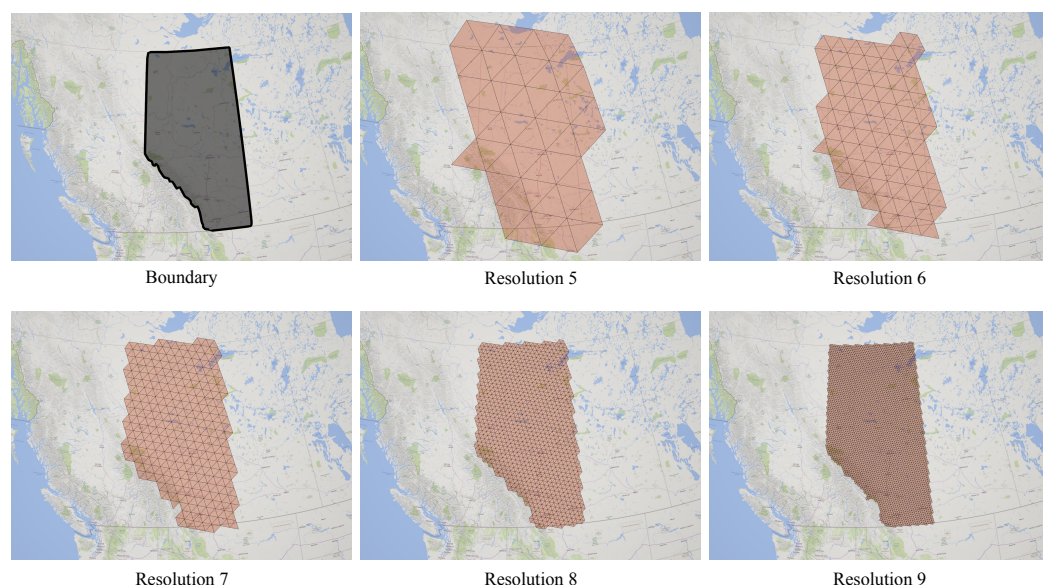


Figure 10. Different choices for the tensor resolution for representing the region of Alberta, Canada.



Figure 11. Subfigure (a) shows an example of sampling the boundary of a tensor at resolution 2 of the DGGs to download the satellite image that has the most overlap with the tensor. The shapefile is used as the region of interest for downloading the data from the EODMS platform. Subfigure (b) shows a zoomed-in view of the sampling points on the boundary of the tensor.

Table 3. The average percentage of the time spent on each encoding step for a single tensor at resolution 7. The target resolution is set to 18 for these results.

Encoding Step	Average Time (%)
Reading the GeoTIFF tile	3.9
Obtaining the DGGs cells	16.3
Sampling	47.1
Run-length indexing	22.8
Wavelet decomposition	7.5
TIFF compression and storage	2.4

Table 4 demonstrates the performance of our method in terms of retrieval time, focusing on the efficiency of data access for various queries. The retrieval time is a crucial factor for providing real-time access to users. For our retrieval queries, we choose two different regions in Alberta, Canada, which are of varying sizes, one small region and one large

region, which have an overlap with more than one tensor. The query resolution is also varied between 16 and 18. Figure 12 also shows the visualized results of the entropy band after the polarimetric decomposition of the SAR images for the region of Lake Minnewanka, Alberta, Canada, at five different zoom levels. The triangle cells of the DGGS are colored based on the entropy values, where yellow cells represent low entropy values and blue cells represent high entropy values. These cells are clearly visible at the fifth zoom level, which corresponds to the highest query resolution, which is 18 in this case.

Table 4. Retrieval times for different query resolutions and for two different regions in Alberta, Canada, with varying sizes. The tensor resolution is set to 7, and the target resolution is set to 18 for these results.

Query Resolution	#Tensors	#Target Cells	Retrieval Time (s)	Region Area (km ²)
Res. 18	3	483,767	122.155	818.93
Res. 17	3	112,030	5.546	818.93
Res. 16	3	28,205	0.478	818.93
Res. 18	2	24,215	0.377	78.01
Res. 17	2	6221	0.086	78.01
Res. 16	2	1654	0.055	78.01

4.1.1. Distributed Computing and Database

A potential advantage of a DGGS is that we could significantly improve performance in both storage and retrieval pipelines by integrating distributed computing and database management systems. Distributing the encoding and retrieval processes across multiple nodes, and thus parallelizing these operations, can lead to faster processing times. While distributed computing is not employed in the current case study, we contend that this approach would be highly advantageous for scaling the method to a global level, improving both efficiency and speed.

Specifically, the hierarchical structure of the DGGS and the tensor-based approach presented in this work facilitate the distribution of both data and processing tasks across multiple nodes. One practical method of distribution involves storing the data corresponding to each tensor, or each set of tensors, on a separate node. Similarly, the encoding and retrieval processes can be distributed across nodes, allowing for parallel execution. This parallelization can lead to substantial reductions in processing times.

Based on current advancements in processing power and distributed computing technologies, we estimate that the processing time for encoding the entire Alberta region of Canada—represented by tensors at resolution 7, which, according to Table 2, currently requires approximately 23.523 s per tensor and involves 502 tensors (totaling around 3 h)—could be reduced to mere minutes. This improvement would also extend to retrieval times, offering further performance gains.

To empirically validate our claim, we conduct a simple experiment aimed at understanding the potential gains from parallelizing data encoding across multiple processing units. In this scenario, we encode 10 tensors at resolution 7, which typically requires 23.523 s per tensor on a single processing unit, as demonstrated by Table 2. By distributing this task across 10 separate processing units, we observe a noticeable decrease in processing time to approximately 26.44 s for all tensors combined, including a minor overhead for task distribution. These results, presented in Table 5, clearly illustrate the significant efficiency improvements achievable through parallel processing, supporting our claim that distributed computing can drastically enhance encoding speeds in large-scale geospatial data systems. We run this experiment by using multi-threading in C++, dividing the task among different CPU cores. Figure 13 shows a diagram for distributing the encoding process of 10 tensors across 10 processing units.

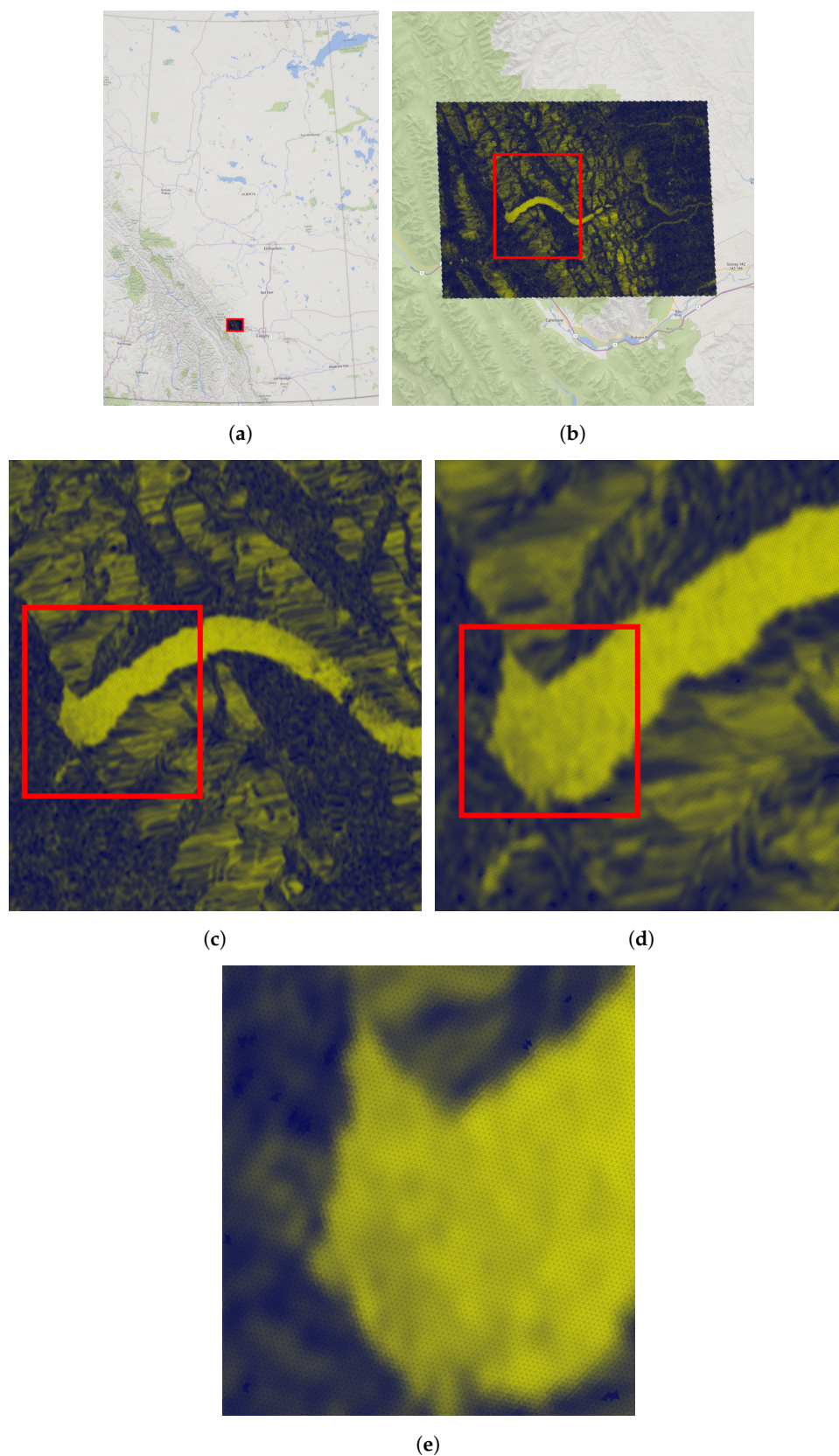


Figure 12. Visualized result of retrieving the entropy band after the polarimetric decomposition of a SAR image from (a) a region in Alberta, Canada, demonstrated at four different zoom levels in (b–e). At each step, the area inside the red rectangle is shown in the next zoom level. Yellow cells represent low entropy values, while blue cells represent high entropy values. The large distinctive yellow region corresponds to Lake Minnewanka, Banff, Alberta, Canada.

Table 5. Encoding time of 10 tensors at resolution 7 in two different scenarios: using a single processing unit and parallelizing the task across 10 processing units. The target resolution is set to 18 for these results.

Processing Units	Encoding Time (s)
1	233.26
10	26.44

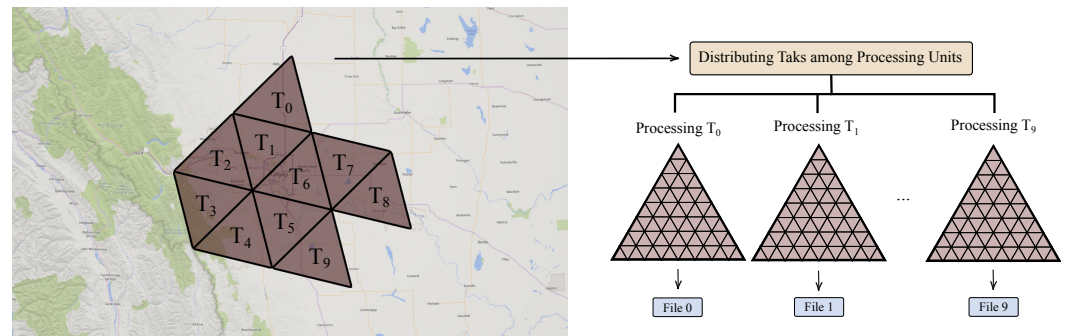


Figure 13. Diagram for distributing the encoding process of 10 tensors across 10 processing units.

4.1.2. Comparison with Existing Systems

In the rapidly evolving field of satellite data management, established platforms such as Sentinel Hub and GEE provide robust solutions for real-time data encoding and retrieval. While these systems offer significant advantages in terms of accessibility and functionality, this subsection aims to highlight how our novel DGGS-based data infrastructure addresses some of the limitations observed in these platforms. By comparing specific elements of our methodology with these existing systems, we underscore the enhancements our approach brings to satellite data management.

First, in contrast to Sentinel Hub, which employs a tile-based method using WGS84 and Universal Transverse Mercator (UTM) coordinate systems for managing geospatial data, our system utilizes a DGGS. Sentinel Hub organizes data into a grid that divides the Earth into variably sized tiles based on specific latitudes and longitudes. This method facilitates regional data handling and supports various grid sizes and resolutions, tailoring to diverse user needs. However, Sentinel-2 data experiences spatial overlaps due to its tiling system, from intersecting projections under the UTM zones and enlarged MGRS tiles, leading to redundancy and spatial overhead [75]. This complicates data handling by creating multiple grid representations for a single location. Our DGGS-based infrastructure reduces distortion errors commonly seen in traditional grid systems, thereby ensuring higher fidelity and consistency in satellite imagery processing. This is particularly beneficial for applications requiring detailed, high-resolution data analysis across multiple scales. Figure 14 illustrates the difference between the Sentinel Hub tiles used for Sentinel-2 data and the triangular tensors of DGGS, which have more flexibility in terms of resolution and size over the Earth's surface.

Secondly, GEE handles multiresolution data through its pyramiding policy, which involves creating and storing multiple resolution levels of an image, tailored for both continuous and categorical data types. This method ensures faster data access across different scales by employing aggregation policies like Mean, Min, Max, or Mode depending on the nature of the data. While this approach offers rapid access to data at various resolutions, it inherently requires more storage space by maintaining separate copies of the data at each resolution level. In contrast, our wavelet-based method utilizes discrete wavelet transforms, tailored to the triangular grid structure of our DGGS, to store data in a way that allows for multiple resolutions to be represented within a single dataset without

any redundancy. This greatly conserves storage space. Unlike GEE’s method, which can result in significant storage overhead, our approach ensures geographic consistency and data integrity across scales with greater efficiency and fewer storage requirements. Table 6 provides a comparison of the storage space required for the same data in both GEE’s image pyramid method and our wavelet-based method. The results show that our method greatly reduces the storage space and is very close in terms of storage time but is slightly slower in terms of retrieval time compared with GEE’s method.

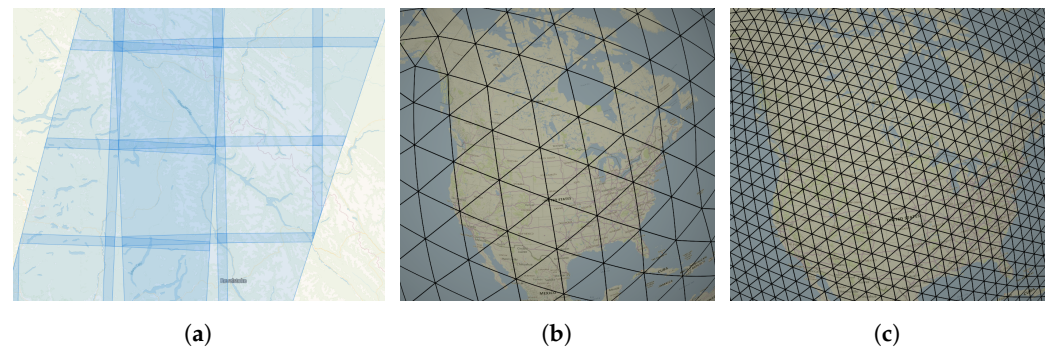


Figure 14. Comparison of (a) the Sentinel Hub tiles used for Sentinel-2 data and (b), (c) the triangular tensors of the DGGS at resolutions 3 and 5, respectively. The Sentinel Hub tiles are fixed, based on specific latitudes and longitudes with overlaps, while our DGGS tensors have more flexibility in terms of resolution and size over the Earth’s surface. The Sentinel Hub tiles are sourced from the Sentinel Hub EOBrowser <https://apps.sentinel-hub.com/eo-browser/>, accessed on 26 January 2025.

Table 6. Comparison of storage space, storage time, and retrieval time at query resolutions 16, 15, and 7 for the same tensor data in GEE’s image pyramid method and our wavelet-based method. We encoded the data for a single tensor at resolution 7, with the target resolution set to 16. For GEE’s image pyramid method, we use the mean aggregation policy and store the data at all the resolution levels from 7 to 16. In contrast, our wavelet-based method stores the wavelet coefficients after applying 9 levels of decomposition.

Method	Storage Space (MB)	Storage Time (s)	Retrieval Time (s) at Resolution 16	Retrieval Time (s) at Resolution 15	Retrieval Time (s) at Resolution 7
GEE	1.73	6.151	24.015	1.579	0.008
Ours	1.30	6.410	24.061	1.582	0.008

Finally, in traditional systems such as those used by the RCM, accessing data for a specific ROI requires downloading and processing multiple large tiles, which then need to be clipped and integrated to extract the desired data. This method can be particularly cumbersome and resource-intensive, especially in applications such as precision agriculture, where the ROIs can be very small and scattered. In contrast, our system significantly streamlines this process through the use of an adaptive data retrieval mechanism. This mechanism allows users to directly access only the data pertinent to their specified ROI, bypassing the need to handle extraneous data. This direct access is facilitated by our DGGS-based data structure, which dynamically adjusts to the scale and location of the interest area, thereby enhancing the efficiency of data processing and significantly reducing the operational overhead. In Figure 15, we illustrate the inefficiency of the EODMS platform in processing a small ROI, which is the city of Calgary, Alberta, Canada. As shown in the figure, for acquiring the data for this relatively small ROI, multiple tiles, almost 20 times larger than the ROI, need to be downloaded and processed. This process is not only time-consuming but also resource-intensive, as we need to extract the data that belong to the ROI from each tile, then integrate them to form the complete data for the ROI. This is in

contrast to our method, which provides more efficient and direct access to the data for any given ROI by utilizing the consistent grid structure of our DGGS.

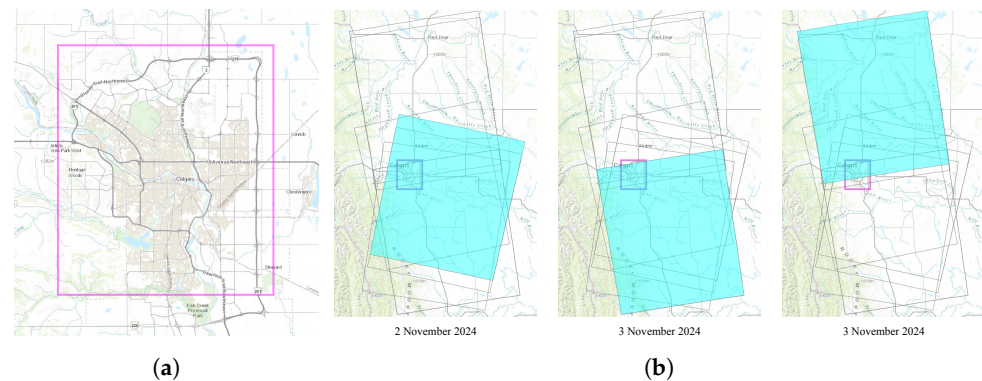


Figure 15. (a) The pink rectangle is the bounding box of the ROI, which is the city of Calgary, Alberta, Canada, and (b) the three tiles that are available in the time range of 2 November 2024 to 3 November 2024, which cover the ROI, are shown in blue. These images are screenshots from the EODMS platform at <https://www.eodms-sgdot.nrcan-rncan.gc.ca/index-en.html>, accessed on 26 January 2025.

4.2. Time-Series Data

Building on the results from the previous section, we explore the application of our DGGS-based framework specifically for managing time-series satellite imagery, especially, the temporal approximation method. Utilizing the data from the RCM, for another case study, we select a specific region, which is the west coast of Lake Minnewanka, in Alberta, Canada. We focus on the change in this fixed region over time. As seen in Figure 16, we acquire five images of this region of an area of 77.68 km² representing the change in the region at five different timestamps: 17 October 2024, 26 October 2024, 3 November 2024, 11 November 2024, and 19 November 2024. Our system preprocesses the data in the same way as the previous case study, with the same target and tensor resolutions, 18 and 7, respectively. The only difference is that we now have multiple images for different timestamps that we have to use the reverse Chaikin subdivision scheme explained in Section 3.

Table 7 presents the retrieval performance in the context of time-series analysis, highlighting our method's ability to deliver continuity in the data change to users. The results show that the queried timestamp does not affect the retrieval times. Also, the visualized results in Figure 17 demonstrate the effectiveness of our temporal approximation method in providing a continuous representation of the data change over time. We use a high temporal resolution of *one* day to retrieve these images from the database, which could be representative of the smooth and continuous change in these data over time.

The results confirm that our methodology not only is capable of handling high-frequency updates but also provides a dependable framework for tracking environmental changes as they occur. This capability is crucial to integrating time-series satellite data into a responsive and adaptable DGGS framework.

Furthermore, we compare the performance of our retrieval pipeline, specifically the local updating mechanism, with the naive approach of retrieving data for the whole time series, i.e., solving a global least squares problem. In this experiment, we focus on only a small region in one tensor at resolution 7 and encode its data for the whole year of 2024 in 48 timestamps. According to Table 7, the performance of our retrieval method is significantly better than the naive approach. Thus, having to retrieve the data for the whole time series for each query is not only inefficient but also impractical for real-time applications.

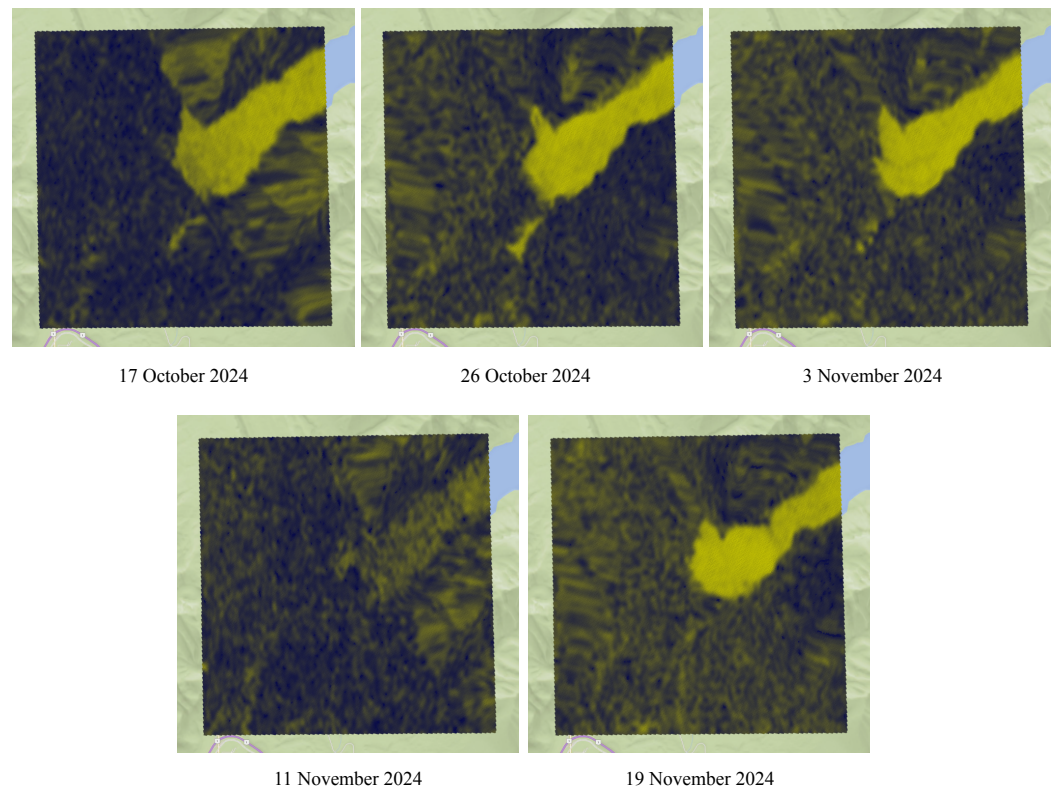


Figure 16. The west coast of Lake Minnewanka, in Alberta, Canada, represented at five different timestamps. The center of the region is at the coordinates 51.25°N, 115.45°W.

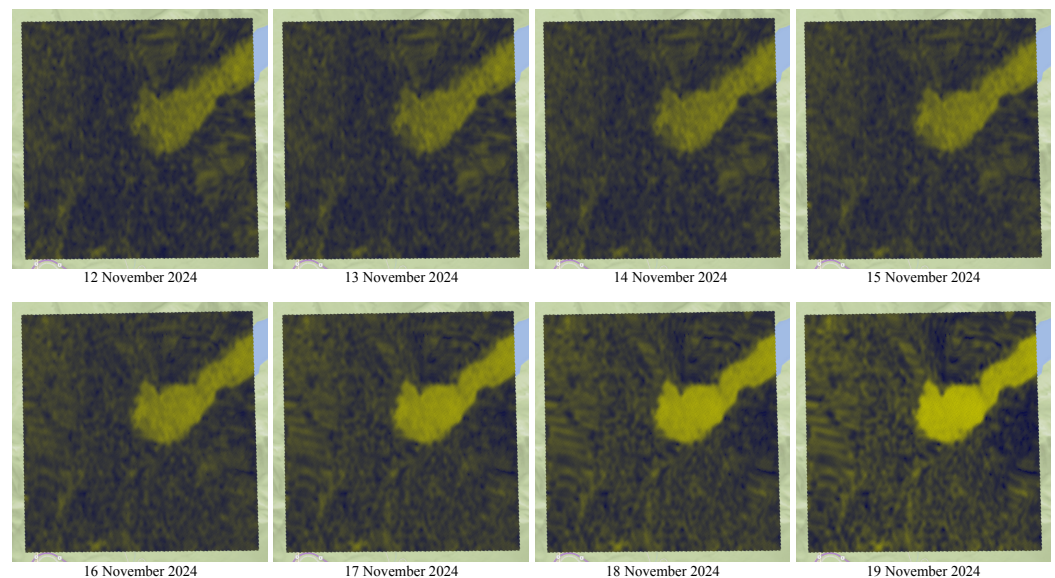


Figure 17. The west coast of Lake Minnewanka, in Alberta, Canada, represented in a continuous manner over time. The center of the region is at the coordinates 51.25°N, 115.45°W. As shown in the figure, the data are represented in a continuous manner at day-level temporal resolution from 12 November 2024 to 19 November 2024 by using our B-spline curve-fitting method.

Finally, to compare our temporal approximation method with a linear interpolation method, it is important to note that the linear method performs better in terms of retrieval efficiency, as it requires retrieving only two data points for any given queried timestamp. This is also shown in Table 7. However, the approximation curve produced by the linear method is significantly less smooth compared with a second-degree B-spline curve. Compared with the linear interpolation method, the quality of the approximation in our SAR dataset, sourced from the RCM, is notably superior with the B-spline curve-fitting method.

This is due to the smoothness of the resulting curve. Figure 18 shows the visualized results of the west coast of Lake Minnewanka, in Alberta, Canada, represented in a continuous manner over time by using the linear interpolation method. Compared with the results in Figure 17, the linear method produces a less smooth curve. The noisy effect on 11 November 2024 is more pronounced in the linear method, especially at the timestamps 12, 13, and 14 November 2024. This is due to the linear method's inability to smooth out the noise in the data, which is not the case with the B-spline curve-fitting method. In fact, the B-spline curve-fitting method provides a clearer representation of the lake and manages to smooth out the noise in the data. In summary, we recommend the local B-spline curve-fitting method for noisy datasets.

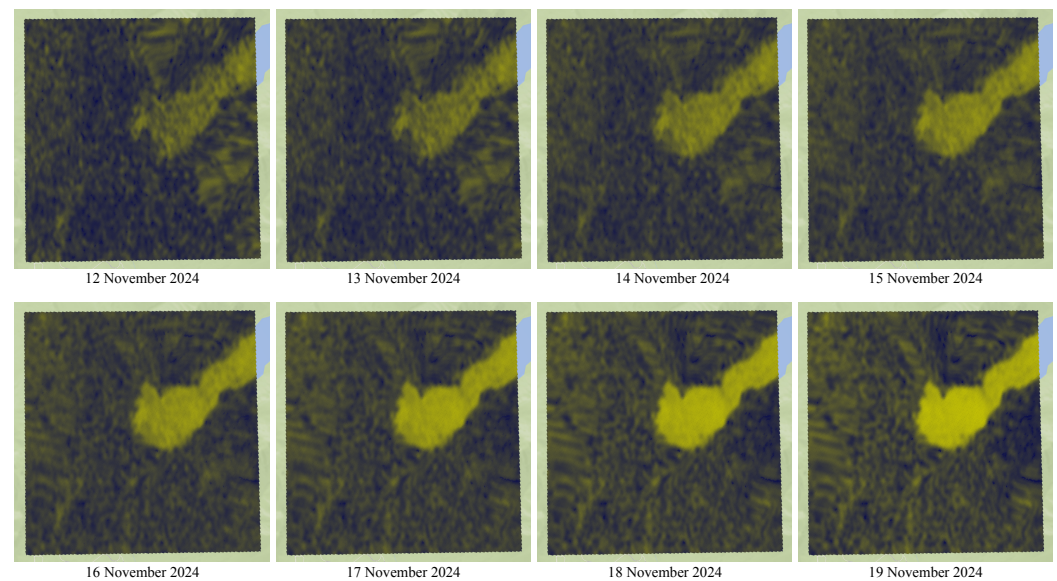


Figure 18. The west coast of Lake Minnewanka, in Alberta, Canada, represented in a continuous manner over time. The center of the region is at the coordinates 51.25°N, 115.45°W. As shown in the figure, the data are represented in a continuous manner at day-level temporal resolution from 12 November 2024 to 19 November 2024 by using the linear interpolation between the data from 11 November 2024 and the data from 19 November 2024.

Regarding the potential use of more complex methods, such as machine learning-based approaches for time-series prediction, it is essential to consider the trade-off between accuracy and performance. Increasing the complexity of the approximation may yield more accurate results, but it comes at the cost of reduced retrieval performance. Therefore, striking a balance between performance and approximation quality is crucial. We believe that our B-spline curve-fitting method achieves an optimal balance between these two factors.

Table 7. Retrieval times for three different arbitrary timestamps for the time-series data of the west coast of Lake Minnewanka, in Alberta, Canada. The encoded time-series data for this region consist of 48 timestamps for the whole year of 2024. The tensor resolution, target resolution, and the query resolution are set to 7, 18, and 18, respectively. We retrieve 105,489 target cells for each query. We compare our *local method* with a *global optimization* approach which requires retrieving the data for the whole time series for each query.

Arbitrary Timestamp	Our Local Method (s)	Global Optimization (s)	Linear Interpolation (s)
10 October 2024	13.92	222.72	9.17
30 October 2024	13.94	220.19	9.16
14 November 2024	13.91	220.56	9.20

5. Conclusions and Future Work

In this article, we introduced a real-time spatiotemporal Earth observation data management system utilizing a DGGs. Our methodology leverages the hierarchical structure of the DGGs in a flexible, tensor-based, multiresolution storage scheme and a real-time updating mechanism, which benefits from the locality of reverse Chaikin subdivision and B-spline curve fitting for temporal data approximation. Furthermore, we employed a triangular wavelet scheme to enable flexible, scalable access to spatiotemporal data at various resolutions.

The results we presented demonstrate that our system greatly enhances the accessibility and efficiency of the real-time processing of satellite raster data. Our system effectively caters to many practical applications in Earth observation by supporting dynamic data retrieval across user-defined ROIs. The presented performance characteristics underscore the system's capability to meet the growing demands for rapid and reliable geospatial data analysis.

More specifically, our methodology presents notable advantages. One key strength lies in the system's ability to manage large volumes of satellite imagery data, as demonstrated in our case studies. According to the results provided in Section 4, our methodology is shown to be effective in encoding, storing, and retrieving data in real time, even at a scale as large as the entire province of Alberta, Canada. The results also highlight the system's capability in managing time-series data, showcasing the effectiveness of our temporal approximation method in providing continuous data representation over time. This feature is implemented in such a way that we can also benefit from a real-time updating mechanism, which is crucial to time-sensitive applications.

However, the potential hurdles and limitations of our approach should not be overlooked. One significant improvement area lies in optimizing the system's performance, particularly in terms of storage and retrieval times, by employing a distributed computing and database management system. As explained in Section 4, for practical global-scale applications, the system's performance can be greatly enhanced by distributing the data processing and storage across multiple nodes, possibly assigning each tensor to a separate node, benefiting from our tensor-based methodology. Another area of improvement is the temporal approximation method, which can be further optimized to provide more accurate approximations of the data at missing timestamps. Spatiotemporal methods such as the one presented by Amirfakhrian et al. [40] can be employed to use both the spatial and temporal axes to predict the missing data values more accurately.

The progressive reconstruction of target images on the client side by using data transformed by wavelets and stored on the server side represents another exciting direction for future development. By leveraging wavelet coefficients, which inherently support multiresolution analysis, we can enable an incremental refinement of image quality during data transfer. This approach minimizes bandwidth usage and reduces latency, making it well suited for interactive applications where quick initial previews are followed by progressively detailed imagery.

Moreover, looking ahead, optimizing the system for more extensive data volumes and broader application scopes is a key focus for future work. This includes enhancing the system's capabilities to include 3D data visualization and analysis, extending beyond the Earth's surface to various altitudes. These improvements aim to further refine our system's efficiency and expand its applicability across more diverse geospatial and environmental monitoring tasks.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/rs17040570/s1>, Video demonstration of the developed real-time spatiotemporal DGGS-based infrastructure.

Author Contributions: Conceptualization, A.M.G. and F.F.S.; Methodology, A.M.G. and F.F.S.; Software, A.M.G.; Validation, A.M.G., M.H. and F.F.S.; Formal analysis, A.M.G. and F.F.S.; Writing—original draft, A.M.G.; Writing—review and editing, A.M.G., M.H. and F.F.S.; Visualization, A.M.G.; Supervision, F.F.S. All authors have read and agreed to the published version of the manuscript.

Funding: We acknowledge the support of Vivid Theory (BigGeo), Mathematics of Information Technology and Complex Systems (MITACS), and the Natural Sciences and Engineering Research Council of Canada (NSERC).

Data Availability Statement: RADARSAT data were obtained from the Earth Observation Data Management System (EODMS) platform, at <https://www.eodms-sgdot.nrcan-rncan.gc.ca/index-en.html>, accessed on 8 December 2024.

Acknowledgments: Our appreciation goes to the Vivid Theory (BigGeo) team for their collaboration, especially for providing the DGGS base code. Their support greatly shaped the research work presented in this article. We also thank the Graphics, Interaction, and Visualization (GIV) research group at the University of Calgary for their invaluable support and feedback throughout the research process. Special thanks to Lakin Wecker for his review of the results and his assistance with setting up the BigGeo DGGS base code. The efforts and dedication of all involved have significantly enriched this project.

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

References

- Huang, Y.; Chen, Z.x.; Yu, T.; Huang, X.z.; Gu, X.f. Agricultural Remote Sensing Big Data: Management and Applications. *J. Integr. Agric.* **2018**, *17*, 1915–1931. [\[CrossRef\]](#)
- Yao, X.; Li, G.; Xia, J.; Ben, J.; Cao, Q.; Zhao, L.; Ma, Y.; Zhang, L.; Zhu, D. Enabling the Big Earth Observation Data via Cloud Computing and DGGS: Opportunities and Challenges. *Remote Sens.* **2019**, *12*, 62. [\[CrossRef\]](#)
- Rathore, M.M.U.; Paul, A.; Ahmad, A.; Chen, B.W.; Huang, B.; Ji, W. Real-Time Big Data Analytical Architecture for Remote Sensing Application. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 4610–4621. [\[CrossRef\]](#)
- Gomes, V.C.F.; Queiroz, G.R.; Ferreira, K.R. An Overview of Platforms for Big Earth Observation Data Management and Analysis. *Remote Sens.* **2020**, *12*, 1253. [\[CrossRef\]](#)
- Thompson, A.A. Overview of the RADARSAT Constellation Mission. *Can. J. Remote Sens.* **2015**, *41*, 401–407. [\[CrossRef\]](#)
- Mahdavi Amiri, A.; Alderson, T.; Samavati, F. Geospatial Data Organization Methods with Emphasis on Aperture-3 Hexagonal Discrete Global Grid Systems. *Cartogr. Int. J. Geogr. Inf. Geovisualization* **2019**, *54*, 30–50. [\[CrossRef\]](#)
- Mahdavi-Amiri, A.; Alderson, T.; Samavati, F. A Survey of Digital Earth. *Comput. Graph.* **2015**, *53*, 95–117. [\[CrossRef\]](#)
- Jawerth, B.; Sweldens, W. An Overview of Wavelet Based Multiresolution Analyses. *SIAM Rev.* **1994**, *36*, 377–412. [\[CrossRef\]](#)
- Jia, R.Q.; Shen, Z. Multiresolution and Wavelets. *Proc. Edinb. Math. Soc.* **1994**, *37*, 271–300. [\[CrossRef\]](#)
- Samavati, F.F.; Bartels, R.H. Multiresolution Curve and Surface Representation: Reversing Subdivision Rules by Least-Squares Data Fitting. *Comput. Graph. Forum* **1999**, *18*, 97–119. [\[CrossRef\]](#)
- Bartels, R.H.; Samavati, F.F. Reversing Subdivision Rules: Local Linear Conditions and Observations on Inner Products. *J. Comput. Appl. Math.* **2000**, *119*, 29–67. [\[CrossRef\]](#)
- Wecker, L.; Hall, J.; Samavati, F.F. Constructing Efficient Mesh-Based Global Grid Systems with Reduced Distortions. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 373. [\[CrossRef\]](#)
- Sahr, K.; White, D.; Kimerling, A.J. Geodesic Discrete Global Grid Systems. *Cartogr. Geogr. Inf. Sci.* **2003**, *30*, 121–134. [\[CrossRef\]](#)
- Guo, H.; Goodchild, M.F.; Annoni, A. (Eds.) *Manual of Digital Earth*; Springer: Singapore, 2020. [\[CrossRef\]](#)
- Alderson, T.; Purss, M.; Du, X.; Mahdavi-Amiri, A.; Samavati, F. Digital Earth Platforms. In *Manual of Digital Earth*; Guo, H., Goodchild, M.F., Annoni, A., Eds.; Springer: Singapore, 2020; pp. 25–54. [\[CrossRef\]](#)
- Purss, M. *Topic 21: Discrete Global Grid Systems Abstract Specification*; Open Geospatial Consortium: Arlington, VA, USA, 2017.
- Kmoch, A.; Vasilyev, I.; Virro, H.; Uuemaa, E. Area and Shape Distortions in Open-Source Discrete Global Grid Systems. *Big Earth Data* **2022**, *6*, 256–275. [\[CrossRef\]](#)

18. Zhou, J.; Ben, J.; Liang, Q.; Huang, X.; Ding, J. A General Modeling Scheme for Spatiotemporal DGGS with Emphasis on Encoding and Operating Multiscale Time Grids. *Trans. GIS* **2024**, *28*, 1130–1155. [\[CrossRef\]](#)
19. Sherlock, M.J.; Hasan, M.; Samavati, F.F. Interactive Data Styling and Multifocal Visualization for a Multigrid Web-Based Digital Earth. *Int. J. Digit. Earth* **2021**, *14*, 288–310. [\[CrossRef\]](#)
20. Mahdavi-Amiri, A.; Samavati, F. Atlas of Connectivity Maps. *Comput. Graph.* **2014**, *39*, 1–11. [\[CrossRef\]](#)
21. Fichtner, F.; Mandery, N.; Wieland, M.; Groth, S.; Martinis, S.; Riedlinger, T. Time-Series Analysis of Sentinel-1/2 Data for Flood Detection Using a Discrete Global Grid System and Seasonal Decomposition. *Int. J. Appl. Earth Obs. Geoinf.* **2023**, *119*, 103329. [\[CrossRef\]](#)
22. Kazemi, M.; Samavati, F.F. Automatic Soil Sampling Site Selection in Management Zones Using a Multi-Objective Optimization Algorithm. *Agriculture* **2023**, *13*, 1993. [\[CrossRef\]](#)
23. Miao, S.; Cheng, C.; Zhai, W.; Ren, F.; Zhang, B.; Li, S.; Zhang, J.; Zhang, H. A Low-Altitude Flight Conflict Detection Algorithm Based on a Multilevel Grid Spatiotemporal Index. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 289. [\[CrossRef\]](#)
24. Guo, H.; Liu, Z.; Jiang, H.; Wang, C.; Liu, J.; Liang, D. Big Earth Data: A New Challenge and Opportunity for Digital Earth's Development. *Int. J. Digit. Earth* **2017**, *10*, 1–12. [\[CrossRef\]](#)
25. Li, M.; Stefanakis, E. Geospatial Operations of Discrete Global Grid Systems—A Comparison with Traditional GIS. *J. Geovisualization Spat. Anal.* **2020**, *4*, 26. [\[CrossRef\]](#)
26. Hojati, M.; Robertson, C.; Roberts, S.; Chaudhuri, C. GIScience Research Challenges for Realizing Discrete Global Grid Systems as a Digital Earth. *Big Earth Data* **2022**, *6*, 358–379. [\[CrossRef\]](#)
27. Xia, J.; Yang, C.; Li, Q. Building a Spatiotemporal Index for Earth Observation Big Data. *Int. J. Appl. Earth Obs. Geoinf.* **2018**, *73*, 245–252. [\[CrossRef\]](#)
28. Tong, X.; Cheng, C.; Wang, R.; Ding, L.; Zhang, Y.; Lai, G.; Wang, L.; Chen, B. An Efficient Integer Coding Index Algorithm for Multi-Scale Time Information Management. *Data Knowl. Eng.* **2019**, *119*, 123–138. [\[CrossRef\]](#)
29. Koubarakis, M.; Kyzirakos, K.; Nikolaou, C.; Garbis, G.; Bereta, K.; Dogani, R.; Giannakopoulou, S.; Smeros, P.; Savva, D.; Stamoulis, G.; et al. Managing Big, Linked, and Open Earth-Observation Data: Using the TELEIOS\LEO Software Stack. *IEEE Geosci. Remote Sens. Mag.* **2016**, *4*, 23–37. [\[CrossRef\]](#)
30. Pekturk, M.K.; Unal, M. A review on real-time big data analysis in remote sensing applications. In Proceedings of the 2017 25th Signal Processing and Communications Applications Conference (SIU), Antalya, Turkey, 15–18 May 2017; pp. 1–4. [\[CrossRef\]](#)
31. Hebert, D.J.; Kim, H. Image Encoding with Triangulation Wavelets. In Proceedings of the SPIE's 1995 International Symposium on Optical Science, Engineering, and Instrumentation, San Diego, CA, USA, 1 September 1995; Laine, A.F., Unser, M.A., Eds.; SPIE: Bellingham, WA, USA, 1995; pp. 381–392. [\[CrossRef\]](#)
32. Fujinoki, K.; Vasilyev, O.V. Triangular Wavelets: An Isotropic Image Representation with Hexagonal Symmetry. *EURASIP J. Image Video Process.* **2009**, *2009*, 248581. [\[CrossRef\]](#)
33. Stollnitz, E.J.; DeRose, T.; Salesin, D. *Wavelets for Computer Graphics: Theory and Applications*; Morgan Kaufmann: Burlington, MA, USA, 1996.
34. Chakrabarti, K.; Garofalakis, M.; Rastogi, R.; Shim, K. Approximate Query Processing Using Wavelets. *VLDB J.* **2001**, *10*, 199–223. [\[CrossRef\]](#)
35. Garofalakis, M.; Gibbons, P.B. Wavelet Synopses with Error Guarantees. In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, SIGMOD '02, New York, NY, USA, 3–6 June 2002; pp. 476–487. [\[CrossRef\]](#)
36. Samavati, F.; Bartels, R. Local Filters of B-spline Wavelets. In Proceedings of the International Workshop on Biometric Technologies (BT'04), Hong Kong, China, 15–17 July 2004; pp. 105–110.
37. Li, T.; Li, Q.; Zhu, S.; Ogihara, M. A Survey on Wavelet Applications in Data Mining. *SIGKDD Explor. Newsl.* **2002**, *4*, 49–68. [\[CrossRef\]](#)
38. Vitter, J.S.; Wang, M. Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. In Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, SIGMOD '99, New York, NY, USA, 31 May–3 June 1999; pp. 193–204. [\[CrossRef\]](#)
39. Vitter, J.S.; Wang, M.; Iyer, B. Data Cube Approximation and Histograms via Wavelets. In Proceedings of the Seventh International Conference on Information and Knowledge Management, Bethesda, MD, USA, 2–7 November 1998; pp. 96–104. [\[CrossRef\]](#)
40. Amirfakhrian, M.; Samavati, F.F. Variational-Based Spatial–Temporal Approximation of Images in Remote Sensing. *Remote Sens.* **2024**, *16*, 2349. [\[CrossRef\]](#)
41. Rogers, D.; Fog, N. Constrained B-spline Curve and Surface Fitting. *Comput. Aided Des.* **1989**, *21*, 641–648. [\[CrossRef\]](#)
42. Ong, C.; Wong, Y.; Loh, H.; Hong, X. An Optimization Approach for Biarc Curve-Fitting of B-spline Curves. *Comput. Aided Des.* **1996**, *28*, 951–959. [\[CrossRef\]](#)
43. Piegl, L.; Tiller, W. Least-Squares B-Spline Curve Approximation with Arbitrary End Derivatives. *Eng. Comput.* **2000**, *16*, 109–116. [\[CrossRef\]](#)

44. Ateshian, G.A. A B-Spline Least-Squares Surface-Fitting Method for Articular Surfaces of Diarthrodial Joints. *J. Biomech. Eng.* **1993**, *115*, 366–373. [\[CrossRef\]](#)
45. Jönsson, P.; Eklundh, L. TIMESAT—A Program for Analyzing Time-Series of Satellite Sensor Data. *Comput. Geosci.* **2004**, *30*, 833–845. [\[CrossRef\]](#)
46. Sadeghi, J.; Samavati, F.F. Smooth Reverse Subdivision. *Comput. Graph.* **2009**, *33*, 217–225. [\[CrossRef\]](#)
47. Lan, S.; Dong, Z. Incorporating Vegetation Type Transformation with NDVI Time-Series to Study the Vegetation Dynamics in Xinjiang. *Sustainability* **2022**, *14*, 582. [\[CrossRef\]](#)
48. Zhou, Y.; Wang, S.; Wu, T.; Feng, L.; Wu, W.; Luo, J.; Zhang, X.; Yan, N. For-Backward LSTM-based Missing Data Reconstruction for Time-Series Landsat Images. *GIScience Remote Sens.* **2022**, *59*, 410–430. [\[CrossRef\]](#)
49. Bartels, R.H.; Beatty, J.C.; Barsky, B.A. *An Introduction to Splines for Use in Computer Graphics & Geometric Modeling*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1987.
50. Cote, S.; Lapointe, M.; De Lisle, D.; Arsenault, E.; Wierus, M. The RADARSAT Constellation: Mission Overview and Status. In Proceedings of the EUSAR 2021; 13th European Conference on Synthetic Aperture Radar, Online, 29–31 April 2021; pp. 1–5.
51. Dabboor, M.; Olthof, I.; Mahdianpari, M.; Mohammadimanesh, F.; Shokr, M.; Brisco, B.; Homayouni, S. The RADARSAT Constellation Mission Core Applications: First Results. *Remote Sens.* **2022**, *14*, 301. [\[CrossRef\]](#)
52. Dabboor, M.; Geldsetzer, T. Towards Sea Ice Classification Using Simulated RADARSAT Constellation Mission Compact Polarimetric SAR Imagery. *Remote Sens. Environ.* **2014**, *140*, 189–195. [\[CrossRef\]](#)
53. Dabboor, M.; Montpetit, B.; Howell, S. Assessment of the High Resolution SAR Mode of the RADARSAT Constellation Mission for First Year Ice and Multiyear Ice Characterization. *Remote Sens.* **2018**, *10*, 594. [\[CrossRef\]](#)
54. Dede, M.; Widiawaty, M.A. Utilization EOS Platform as Cloud-Based GIS to Analyze Vegetation Greenness in Cirebon Regency, Indonesia. *J. Inf. Technol. Its Util.* **2020**, *3*, 1–4. [\[CrossRef\]](#)
55. Borgogno-Mondino, E.; Fissore, V. Reading Greenness in Urban Areas: Possible Roles of Phenological Metrics from the Copernicus HR-VPP Dataset. *Remote Sens.* **2022**, *14*, 4517. [\[CrossRef\]](#)
56. Malenovsky, Z.; Rott, H.; Cihlar, J.; Schaepman, M.E.; García-Santos, G.; Fernandes, R.; Berger, M. Sentinels for Science: Potential of Sentinel-1, -2, and -3 Missions for Scientific Observations of Ocean, Cryosphere, and Land. *Remote Sens. Environ.* **2012**, *120*, 91–101. [\[CrossRef\]](#)
57. Gorelick, N.; Hancher, M.; Dixon, M.; Ilyushchenko, S.; Thau, D.; Moore, R. Google Earth Engine: Planetary-scale Geospatial Analysis for Everyone. *Remote Sens. Environ.* **2017**, *202*, 18–27. [\[CrossRef\]](#)
58. Mutanga, O.; Kumar, L. Google Earth Engine Applications. *Remote Sens.* **2019**, *11*, 591. [\[CrossRef\]](#)
59. European-Space-Agency. *Sentinel Application Platform (SNAP)*; European-Space-Agency: Paris, France, 2022.
60. Luo, S.; Jiang, X.; Yang, K.; Li, Y.; Fang, S. Multispectral Remote Sensing for Accurate Acquisition of Rice Phenotypes: Impacts of Radiometric Calibration and Unmanned Aerial Vehicle Flying Altitudes. *Front. Plant Sci.* **2022**, *13*, 958106. [\[CrossRef\]](#)
61. Choi, H.; Jeong, J. Speckle Noise Reduction Technique for SAR Images Using Statistical Characteristics of Speckle Noise and Discrete Wavelet Transform. *Remote Sens.* **2019**, *11*, 1184. [\[CrossRef\]](#)
62. Loew, A.; Mauser, W. Generation of Geometrically and Radiometrically Terrain Corrected SAR Image Products. *Remote Sens. Environ.* **2007**, *106*, 337–349. [\[CrossRef\]](#)
63. Yamaguchi, Y.; Moriyama, T.; Ishido, M.; Yamada, H. Four-Component Scattering Model for Polarimetric SAR Image Decomposition. *IEEE Trans. Geosci. Remote Sens.* **2005**, *43*, 1699–1706. [\[CrossRef\]](#)
64. Devys, E.; Habermann, T.; Heazel, C.; Lott, R.; Rouault, E. *OGC GeoTIFF Standard*; Open Geospatial Consortium: Arlington, VA, USA, 2019.
65. Lin, B.; Zhou, L.; Xu, D.; Zhu, A.X.; Lu, G. A Discrete Global Grid System for Earth System Modeling. *Int. J. Geogr. Inf. Sci.* **2018**, *32*, 711–737. [\[CrossRef\]](#)
66. White, D.; Kimerling, A.J.; Sahr, K.; Song, L. Comparing Area and Shape Distortion on Polyhedral-Based Recursive Partitions of the Sphere. *Int. J. Geogr. Inf. Sci.* **1998**, *12*, 805–827. [\[CrossRef\]](#)
67. Antonini, M.; Barlaud, M.; Mathieu, P.; Daubechies, I. Image Coding Using Wavelet Transform. *IEEE Trans. Image Process.* **1992**, *1*, 205–220. [\[CrossRef\]](#) [\[PubMed\]](#)
68. TIFF, Revision 6.0. 2024. Available online: <https://www.loc.gov/preservation/digital/formats/fdd/fdd000022.shtml> (accessed on 10 December 2024).
69. Mahdavi-Amiri, A. Atlas of Connectivity Maps. Doctoral Thesis, University of Calgary, Calgary, AB, Canada, 2015.
70. Goodchild, M.F.; Shiren, Y. A Hierarchical Spatial Data Structure for Global Geographic Information Systems. *CVGIP Graph. Models Image Process.* **1992**, *54*, 31–44. [\[CrossRef\]](#)
71. Dutton, G.H. *A Hierarchical Coordinate System for Geoprocessing and Cartography: Working Through the Scales*; Springer: Berlin/Heidelberg, Germany, 1999.
72. Moghaddam, B.; Hintz, K.J.; Stewart, C.V. Space-Filling Curves for Image Compression. In Proceedings of the Automatic Object Recognition, Orlando, FL, USA, 1 April 1991; SPIE: Bellingham, WA, USA, 1991; Volume 1471, pp. 414–421. [\[CrossRef\]](#)

73. Amani, M.; Ghorbanian, A.; Ahmadi, S.A.; Kakooei, M.; Moghimi, A.; Mirmazloumi, S.M.; Moghaddam, S.H.A.; Mahdavi, S.; Ghahremanloo, M.; Parsian, S.; et al. Google Earth Engine Cloud Computing Platform for Remote Sensing Big Data Applications: A Comprehensive Review. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *13*, 5326–5350. [[CrossRef](#)]
74. Mahdavi-Amiri, A.; Harrison, E.; Samavati, F. Hierarchical Grid Conversion. *Comput. Aided Des.* **2016**, *79*, 12–26. [[CrossRef](#)]
75. Bauer-Marschallinger, B.; Falkner, K. Wasting Petabytes: A Survey of the Sentinel-2 UTM Tiling Grid and Its Spatial Overhead. *ISPRS J. Photogramm. Remote Sens.* **2023**, *202*, 682–690. [[CrossRef](#)]

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