Geospatail Data Organization Methods with Emphasis on Aperture 3 Hexagonal Discrete Global Grid Systems

April 30, 2018

Abstract

Digital Earth frameworks deal with data sets of different types collected from various sources. In order to effectively store, retrieve, and transmit these data sets, efficient multiscale data representations that are compatible with the underlying structure of the Digital Earth framework are required. In this paper, we describe several such techniques and their properties; namely, how to represent data in the multiscale cell hierarchy of a DGGS or in the multiscale hierarchy of a customized wavelet transform. We also discuss how these techniques can be tuned to be applicable to the A3H DGGS.

1 Introduction

The Digital Earth provides a representation of the Earth on which data sets of different types and from different sources can be integrated, analyzed and visualized [21]; and is commonly implemented using a Discrete Global Grid System (DGGS) [41]. In a DGGS, the Earth is typically approximated using a spherical polyhedron that is iteratively refined in order to generate a multiresolution hierarchy [21]. Through this process, the surface of the Earth is discretized into a hierarchical set of cells, where the area of a refined/child cell as compared to a parent cell is described by the factor (or aperture) of the refinement. These cells are then projected to the sphere using one of a variety of spherical projection methods, although equal area projections are usually more desirable when data analysis is emphasized [59, 22].

DGGSs consisting primarily of hexagonal cells are particularly popular [40, 53, 52], due to the superior sampling behavior of hexagonal tilings. Especially common are those DGGSs, collectively known as Aperture 3 Hexagonal (A3H) DGGSs, that consist of hexagonal cells refined by factor of three (aperture 3), which provides a smooth transition between resolutions [36, 54, 24]. See Figure 1 for an illustration.

As the core functionality of any Digital Earth lies in the efficient management of geospatial data (primarily imagery and elevation data, vector data, and quantitative data), techniques for the representation of geospatial data merit discussion. One of the core obstacles

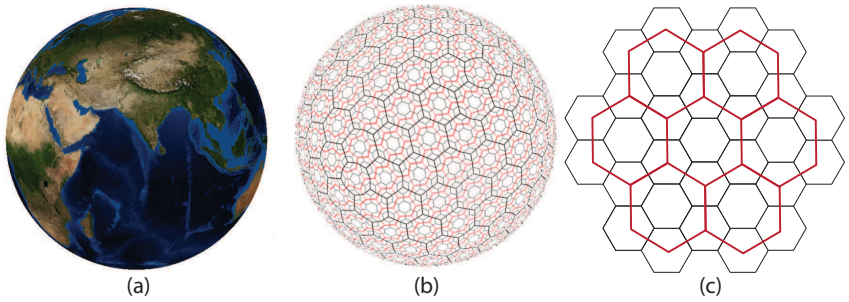


Figure 1: (a), (b) The PYXIS Digital Earth, which is an A3H DGGS, and its cell structure. (c) The 1-to-3 hexagonal refinement (aperture 3).

inherent to this topic lies in the immense scale of and storage requirements for many geospatial data sets, which continues to grow with improvements to data capturing technologies. As a result, geospatial data often cannot be stored locally on a single machine, and must be distributed across a set of servers. Client machines with limited memory may then connect to these servers in order to request data or initiate geospatial queries, ranging from requests for specific images in a given area to statistical analyses on quantitative data.

The issue of scale additionally requires that data representations be multiscale. The immense difference in the scope of a city neighborhood against that of an entire country makes it often impractical to visualize or process a data set at its native resolution. While the cell hierarchy of a DGGS offers a natural multiscale representation for different data sets, alternative representations — such as wavelet transforms — can offer additional benefits to certain operations. In particular, appropriate multiscale representations can be used to counteract the adverse effect of data size increases on transmission and query processing times, and ensure that queries are handled quickly and accurately.

In this paper, we describe different methods that may be used to represent, in a multiscale manner, the three main types of geospatial data, with particular emphasis on client-server Digital Earth architectures supported by an A3H DGGS. We present a number of existing and novel methods that represent data with wavelet transforms or convert data into a wavelet-supported form, largely for the purposes of efficient data transmission or querying. Such methods allow for an initial visualization or query result to be produced with approximate data, which may then be refined to the correct result using loss-less wavelet techniques as additional data arrives via transmission or finishes processing.

In Section 3, we briefly overview how geospatial data may be stored in and retrieved from a DGGS. The primary mechanism underlying these processes is a DGGS indexing system. Two such indexing systems for A3H DGGSs, which assign a unique index to each cell of the DGGS, are described in detail.

Section 4 then describes how imagery and elevation data sets can be extracted from the cell hierarchy of an A3H DGGS for the purposes of transmission or rendering. This is then

followed by discussions on how these data may be loss-lessy compressed for transmission to client machines using, e.g., the Haar wavelet transform. A novel wavelet transform that is compatible with an A3H DGGS — the integer ternary Haar wavelet — is additionally described.

Following this, in Section 5 we describe approaches to representing vector data in a DGGS. These approaches include rasterization, hierarchical cell representation, and a geometry-based spherical vector representation. We introduce here a generalization of the hierarchical cell representation of [44] (which works with quadtrees) to more general types of DGGS.

Finally, Section 6 discusses the management of quantitative data sets. Statistics pertaining to each cell may be transmitted and analyzed before the transmission completes (with a known, progressively decreasing error) using wavelet histograms and range trees. We present a novel modification of nLT trees that allows storage requirements to be reduced, and a modification of the wavelet histogram that approximates the data using a piece-wise linear function. By applying the wavelet histogram to the difference between this function and the original data, the error from analyzing the incomplete data can be controlled and reduced.

2 Related Work

Digital Earths, their applications (data analysis being an important example), and the different methods used to build, visualize, and index them have been studied extensively in the literature [21]. There are three important types of data sets for Digital Earths that are considered in this paper: imagery/elevation, vector, and quantitative data. In the following, we describe each of these data sets and note some of the methods that address analysis and representation in geospatial applications.

2.1 Imagery and Elevation Data Sets

Geospatial imagery data sets are often beneficial to the visualization and analysis of locations, and are often used as textures for the Earth's cells or used to provide special views of the Earth (e.g. spherical panoramic views). Data assignment to the cells of a DGGS is most commonly performed via rasterization, where each cell of the DGGS is treated as a pixel-like entity, and attributes such as color or height are assigned to these cells [40].

DGGSs induce a natural multiresolution representation for images, as each resolution of a DGGS contains a set of images that represent the Earth or specific regions at that particular resolution. However, alternative multiresolution representations for images can also be beneficial. For instance, mip-mapping [60] has been used in, e.g., [15] to establish a continuous transition between different correlated images. In addition, wavelet transforms (which are described in [51]) can be used to produce a low-resolution version of an image that can be efficiently transmitted through a network and gradually restored to its original

resolution on a client as data arrives. A simple and useful wavelet transform, which we make use of throughout the paper, is the Haar wavelet. Refer to Section 4.2 for a description.

Elevation data sets are often very similar to images, as Data Elevation Models are uniform grids with values for heights instead of colors. As a result, their rasterization and wavelet transforms are very similar to those employed for images; the only difference being that there is no need to use integer wavelets for elevation data sets. This is because they are typically represented using floating point values as opposed to integers.

2.2 Vector Data Sets

Vector data sets are available in the forms of points, polylines, or polygons. These points are usually connected by spherical or ellipsoidal arcs and represent networks (such as road or river networks) or region boundaries (such as those belonging to continents, cities, etc). Vector data can be produced via ground surveying, LIDAR, and photogrammetry [20]. Features in LIDAR data sets can be detected and vectorized using different techniques [3, 39, 31, 8], although imagery data alone can also be used to extract vector data sets [31, 9, 47, 10, 30, 5].

Three main methods are available to visualize vector data sets: texture-based, geometry-based, and shadow volume-based [63]. Texture-based approaches rasterize the vector data into textures that are then mapped onto the terrain surface [16]. Geometry-based approaches consider the geometry of the vector data separate from the geometry of the terrain [46, 45, 1, 38, 1, 57], and can be used with wavelet transforms [2] to provide multiscale rendering. Shadow volume-based approaches extrude the vector geometry into polyhedrons that are rendered via the stencil buffer to distinguish visible and invisible parts of the scene [17].

As the individual grids of a DGGS each provide a rasterization of the Earth, vectors are often converted into raster images for storage in a DGGS, particularly on the web as in [14, 32]. Alternative approaches include the hierarchical cell representation, in which a vector is represented as an ordered set of cells at each resolution of the DGGS. Treating the cells of the DGGS as "buckets" allows data storage techniques similar to those used by quadtrees to be employed [43, 44], and can be used to store the actual vector geometry to address rasterization artifacts. Example works that use the approach include [37], which uses the cells of a A3H DGGS to store vectors; and [63], which uses a multiresolution vector pyramid to store vectors in a QQM (quaternary quadrangle model) DGGS.

2.3 Quantitative Data Sets

Quantitative data sets provide statistical and other numerical information related to locations. These statistical data can be environmental, biological, or demographic in nature (e.g. average income of a particular location). They are usually collected by sampling regions through various means (e.g. surveys or sensors) [62] and are assigned to the cells

of a DGGS as attributes related to that particular location.

In order to obtain useful information from these data sets, techniques from the field of information processing may be used. Though not specific to DGGS, of particular importance is that query processing times on such data sets can become prohibitively expensive in many situations (including data mining [19] and multidimensional aggregate computation [55]), whether due to the complicated nature of the query or the size of the data set. As a result, it can be useful to run the queries on simplified versions of the data to provide fast initial results, with the possibility to refine the data and query result a needed over time. In order to accomplish this, wavelets can again prove useful, and have been used for this purpose in [19, 55, 7, 56, 11].

3 Data Storage and Retrieval

In a DGGS, geospatial data are assigned to and retrieved from cells. In order to support this functionality, each cell requires a unique identifier, called a cell index. Using these indices as references into a database, folder structure, or file (e.g. a spreadsheet) allows cells to be associated with data stored in any of these formats [28]. For instance, one possibility is to utilize a nested folder structure, in which folders are associated with cells and contain files for data sets belonging to the cell (e.g. textures, elevation data). The folder for a child cell is contained within the folder for its parent, allowing the folder structure to be traversed similarly to the DGGS cell structure. Of course, this is not the only method for storing geospatial data sets on disk. Interested readers may refer to [50] for a DGGS implementation and [61] for information on spatial database implementation.

Hence, an essential component of any DGGS is its associated cell indexing system. There are three main types of indexing systems employed within different DGGSs: hierarchy-based, coordinate-based, and space filling curve (SFC) based [28]. We describe in this section two indexing systems that have been proposed for A3H DGGSs: one hierarchy-based and one coordinate based.

In order to define a hierarchy-based indexing system on the hexagonal cells of an A3H DGGS, a hierarchical relationship among the cells is needed [36, 54, 40]. However, as hexagonal refinements are not congruent, this hierarchical relationship is not straightforward to define. The PYXIS hierarchy presents one possibility for defining hierarchical relationships between cells at different resolutions [36, 54]. In this hierarchy, cells are categorized into two types — A and B — generating different fractal shapes called *tiles* throughout the resolutions that fit together to cover the entire surface of the sphere (see Figure 2). Type B cells with index b have children with indices bi ($0 \leq i \leq 6, i \in N$) while a type A cell has only one child with index $a0$. $a0$ and $b0$ are considered to be of type B, while the other bi cells are of type A [24, 54]. We have used this type of indexing system and the tree structure resulting from the parent-child relationships between cells in order to define a hierarchical cell representation for feature vectors (as discussed in Section 5.2).

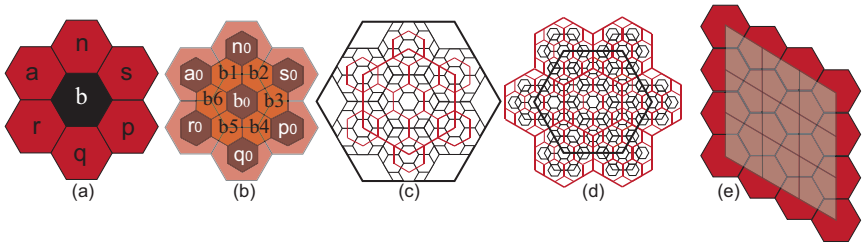


Figure 2: (a), (b) PYXIS hierarchical indexing method. (c), (d) Fractal tiles created from type A and B cells. (e) A diamond covering a set of hexagonal cells. The vertices of the refined diamond are aligned with the centroids of the hexagonal cells.

Hierarchy based indexing systems are efficient at addressing hierarchical queries, but neighborhood queries (i.e. finding the neighbors of a cell) cannot be performed in constant time and have $O(r)$ time complexity (where r is the resolution). To overcome this issue, a second, coordinate-based indexing system can be defined on the hexagonal cells of a given A3H DGGS [24]. One such system is based on the duality relationship between hexagons and triangles (see Figure 2). Once the centroids of the hexagons are connected to form triangles, and pairs of triangles are combined into diamonds [23], the axes of a coordinate system can be defined such that they align with the edges of the diamonds. Using the resulting coordinate system, the vertices of the diamonds — and their associated hexagonal cells — may be indexed. This coordinate-based indexing system can be used to arrange hexagonal cells into a quadrilateral shape (i.e., diamonds) that are compatible with standard graphics pipelines (e.g. OpenGL) and are easy to sample and render. For more details, see Section 4.1.

However, in order to provide interoperability and retrieve data from an external DGGS, a conversion between the cell indices of the different DGGSs is required. In essence, given a cell in the destination DGGS, a corresponding cell that represents approximately the same area in the target DGGS must be determined. This requires a correspondence between the resolutions of the two DGGSs, which can be established based on, e.g., the number of cells that each DGGS provides at specific resolutions. In the absence of an existing conversion, a cell index from the destination DGGS can be converted to an index in the target DGGS using a common intermediate DGGS or geospatial coordinate system (e.g. latitude/longitude) for which an existing conversion is known. See [28] for more details on converting between two Digital Earth frameworks.

4 Imagery and Elevation Data Sets

Imagery and elevation data sets are important to any Digital Earth, and hence need to be integrated, analyzed, and visualized efficiently. In an A3H DGGS, cells are hexagonal and

store information about the region each encompasses. This includes imagery or elevation data, which must be assigned to these cells in order to be represented within the DGGS.

In this section, we describe methods to assign imagery data to the cells of an A3H DGGS and to transmit it through a network. The benefits to visualization offered by the mechanism used to convert the data — namely, the dual conversion — are also discussed. Due to similar structure, elevation data sets are handled similarly.

4.1 Converting from DGGS Cells to Images

Assigning imagery data to DGGS cells requires a correspondence to be established between the cells of the DGGS and the pixels of the image. This can be accomplished using standard cartographic projection techniques, such as Snyder's projection [49]. However, for some purposes, we may wish to convert the DGGS cells back into imagery. For instance, in order to efficiently transmit imagery data to a client DGGS or to render a textured 3D globe, it is important to note that standard image compression techniques and graphics libraries (such as OpenGL) are designed to work with triangular or quadrilateral polygons rather than the hexagonal cells of an A3H DGGS.

The basic idea behind this method is to use a dual conversion, forming triangles that connect the centroids of neighboring hexagonal cells [23]. In essence, the vertices of a triangle each correspond to a hexagonal cell. When these triangular cells are created, they are packed into congruent quadrilaterals, or diamonds (see Figure 3), that can be used for transmission or to render cells efficiently on the GPU [24, 27, 25]. Note that cracks or gaps can appear between diamonds from different resolutions, but can be connected by a set of "zippers" to patch the final surface of the Earth [26].

Given this conversion of hexagonal cells into diamonds compatible with standard compression and rendering techniques, we can serialize hexagonal cells into quadrilateral images by sampling the DGGS at each vertex of the diamond. As the aperture of the cells in an A3H DGGS is three, the resulting sampled images have dimensions of $3^n \times 3^n$ (where n depends on the resolution of the DGGS cells). Figure 4 illustrates an example of image sampling using diamonds.

4.2 Image Data Transmission: The Haar Wavelet

With a client-server DGGS architecture, imagery data sets are sampled and assigned on a server, then sent over a network to clients. However, sending the entire data set can be quite slow due to the potentially large volume of the data. As a result, compression techniques — such as the wavelet-based GML in JPEG 2000 standard [35] — can be very useful.

Wavelets can also be used to design loss-less compression techniques, which send an initial approximation of the data that may be corrected as additional information arrives over time. One of the simplest, yet most efficient, wavelet transforms is the Haar wavelet.

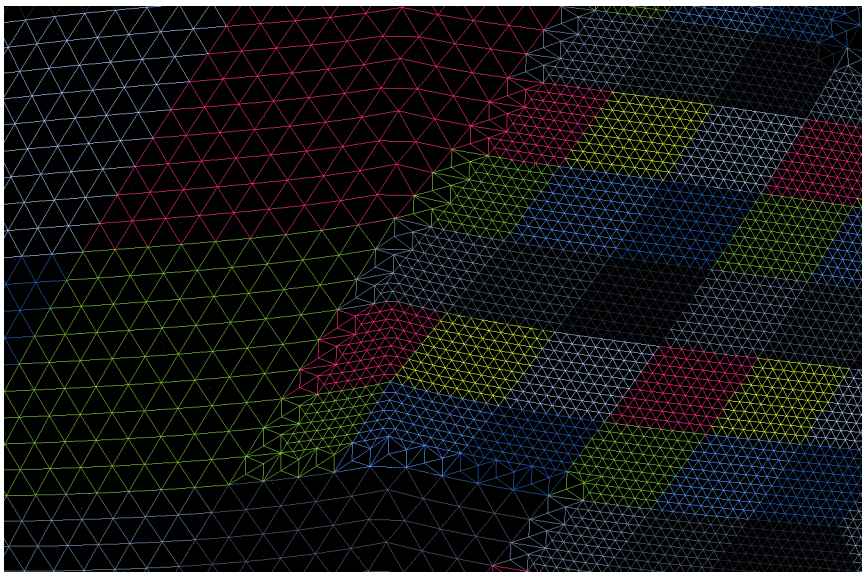


Figure 3: A zoomed-in view of the Earth rendered using diamonds.

Figure 4: (a) A globe textured using diamonds. One of the diamonds has been textured differently to display its relationship to the original hexagonal cells. (b) A view of neighboring textured diamonds. The red lines highlight the boundary of one of the diamonds. (c) A close-up look at the hexagonal sampling of the textures.

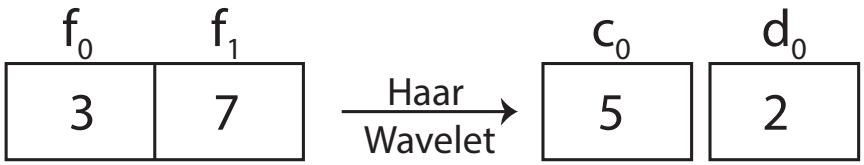


Figure 5: Two fine points f_0 and f_1 are averaged to obtain c_0 in the Haar wavelet transform. The difference between f_0 and c_0 is the detail d_0.

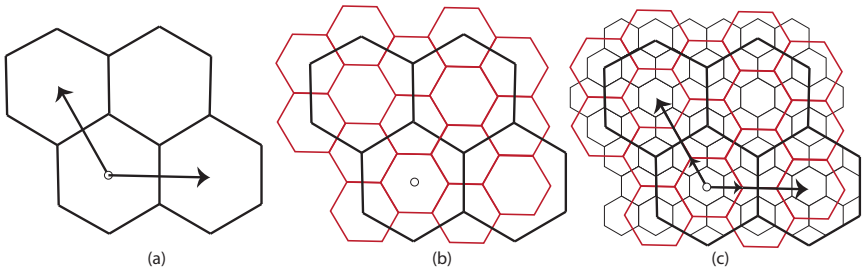


Figure 6: (a) A set of hexagons and a coordinate system defined on them. (b) The coarse hexagons in (a) are refined using 1-to-3 refinement. (c) After two iterations of 1-to-3 refinement, the fine hexagons are aligned with the coarse hexagons in (a).

In this wavelet transform, low resolution data c_i are built by averaging two consecutive high resolution data values f_{2i} and f_{2i+1} in a process called *decomposition*, which may be applied to images by performing the decomposition both row-wise and column-wise. The details corresponding to the decomposition are found as the difference between the low resolution data and the high resolution data $d_i = c_i - f_{2i}$. Using the low resolution data and details, the high resolution data can be reconstructed as $f_{2i} = c_i - d_i$ and $f_{2i+1} = c_i + d_i$ (see Figure 5). As is readily apparent, the dimension of the coarse data and its associated details is the same as that of the original high resolution data set (i.e. no additional information/storage space is needed).

The Haar wavelet is a binary wavelet, as the dimension of the low resolution data is half that of the high resolution data. However, this transform is not readily compatible with an A3H DGGS, for which a 1-to-3 refinement is employed. Two iterations of the 1-to-3 refinement, however, provides a ternary refinement in which the number of cells is tripled along the two main axes defined for cells (see Figure 6). As a result, we suggest a ternary version of Haar that is compatible with an A3H DGGS [34].

4.3 Ternary Haar Wavelet

In the ternary Haar wavelet transform [34], low resolution data c_i are built by averaging three consecutive high resolution data values f_{3i}, f_{3i+1}, and f_{3i+2} in the decomposition

Figure 7: Applying the ternary Haar wavelet to an image two times.

process, i.e. $c_i = \frac{f_{3i}+f_{3i+1}+f_{3i+2}}{3}$. To perfectly reconstruct the high resolution data, we associate two details d_{2i} and d_{2i+1} with c_i, instead of just one, that are individually defined as $d_{2i} = c_i - f_{3i}$ and $d_{2i+1} = c_i - f_{3i+1}$. In the reconstruction process, high resolution data values are reconstructed via $f_{3i} = c_i - d_{2i}$, $f_{3i+1} = c_i - d_{2i+1}$, and $f_{3i+2} = c_i + d_{2i} + d_{2i+1}$. Algorithms 1 and 2 outline the decomposition and reconstruction processes, respectively, of the ternary Haar wavelet. In general, much as the ternary Haar wavelet is appropriate for an A3H DGGS, an n-ary Haar wavelet ought to be used with a DGGS of aperture n. A description (and proof of correctness) for general n-ary Haar wavelets is provided in supplementary material.

Algorithm 1 Decomposition of a single row of an image $\mathbf{f}$ (of n pixels) into $\mathbf{c}$ and $\mathbf{d}$ via the ternary Haar wavelet.

for $i = 0$ to $\frac{n}{3}$, step 3 **do**
  $\mathbf{c}_i = \frac{\mathbf{f}_{3i}+\mathbf{f}_{3i+1}+\mathbf{f}_{3i+2}}{3}$
  $\mathbf{d}_{2i} = \mathbf{c}_i - \mathbf{f}_{3i}$
  $\mathbf{d}_{2i+1} = \mathbf{c}_i - \mathbf{f}_{3i+1}$
end for

Note that in the Haar wavelet transform, if the high resolution values are integers, there is no guarantee that the obtained low resolution values will also be integers after the averaging process. While this is not a problem for elevation data (as such data are naturally available in floating point format), imagery data sets are represented in integer

Algorithm 2 Reconstruction of a single row of an image $\mathbf{f}$ (of n pixels) using $\mathbf{c}$ and $\mathbf{d}$ via the ternary Haar wavelet.

for $i = 0$ to $\frac{n}{3}$, step 3 **do**
  $\mathbf{f}_{3i} = \mathbf{c}_i - \mathbf{d}_{2i}$
  $\mathbf{f}_{3i+1} = \mathbf{c}_i - \mathbf{d}_{2i+1}$
  $\mathbf{f}_{3i+2} = \mathbf{c}_i + \mathbf{d}_{2i} + \mathbf{d}_{2i+1}$
end for

format. Therefore, when applying the Haar wavelet transform, the data must be truncated in order to obtain integer values. In this scenario, however, we cannot perfectly reconstruct the high resolution data due to truncation error. To avoid this problem, we can employ integer ternary Haar wavelets.

4.4 Integer Ternary Haar Wavelet

As proposed in [58], integer wavelets restrict the high resolution data, low resolution data and details to remain as integers throughout the entire multiresolution process. To define an integer ternary Haar wavelet, we modify the ternary Haar wavelet in such a way that all details and low resolution data values become integers. Consider $d_{2i} = c_i - f_{3i}$ and $d_{2i+1} = c_i - f_{3i+1}$. If we substitute $c_i = \frac{f_{3i}+f_{3i+1}+f_{3i+2}}{3}$ into these relations, we get $d_{2i} = \frac{-2f_{3i}+f_{3i+1}+f_{3i+2}}{3}$ and $d_{2i+1} = \frac{f_{3i}-2f_{3i+1}+f_{3i+2}}{3}$. Since c_i might not be an integer, instead of saving c_i, we save $\tilde{c}_i = \lfloor c_i \rfloor$. In this case, d_{2i} and d_{2i+1} are also not integers, so instead of d_{2i} and d_{2i+1}, we save $\tilde{d}_{2i} = -2f_{3i} + f_{3i+1} + f_{3i+2}$ and $\tilde{d}_{2i+1} = f_{3i} - 2f_{3i+1} + f_{3i+2}$. In the reconstruction process, f_{3i} and f_{3i+1} are reconstructed quite simply as $f_{3i} = \tilde{c}_i - \left\lfloor \frac{\tilde{d}_{2i}}{3} \right\rfloor$ and $f_{3i+1} = \tilde{c}_i - \left\lfloor \frac{\tilde{d}_{2i+1}}{3} \right\rfloor$. Finally, given f_{3i}, f_{3i+1} and $\tilde{d}_{2i}$, f_{3i+2} is reconstructed as $\tilde{d}_{2i} + 2f_{3i} - f_{3i+1}$. These decomposition and reconstruction processes are respectively presented in Algorithms 3 and 4. Using this wavelet transform, we can compress imagery data sets in a manner compatible with an A3H DGGS (see Figure 7).

Algorithm 3 Decomposition of a single row of an image $\mathbf{f}$ (of n pixels) into $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{d}}$ via the integer ternary Haar wavelet.

for $i = 0$ to $\frac{n}{3}$, step 3 **do**
  $\tilde{\mathbf{c}}_i = \left\lfloor \frac{\mathbf{f}_{3i}+\mathbf{f}_{3i+1}+\mathbf{f}_{3i+2}}{3} \right\rfloor$
  $\tilde{\mathbf{d}}_{2i} = -2\mathbf{f}_{3i} + \mathbf{f}_{3i+1} + \mathbf{f}_{3i+2}$
  $\tilde{\mathbf{d}}_{2i+1} = -2\mathbf{f}_{3i+1} + \mathbf{f}_{3i} + \mathbf{f}_{3i+2}$
end for

To examine the behavior of our integer ternary Haar wavelet, we compared the Peak Signal to Noise Ratio (PSNR) of our integer ternary Haar wavelet with the PSNR of the

Algorithm 4 Reconstruction of a single row of an image $\mathbf{f}$ (of n pixels) using $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{d}}$ via the integer ternary Haar wavelet.

for $i = 0$ to $\frac{n}{3}$, step 3 **do**
  $\mathbf{f}_{3i} = \tilde{\mathbf{c}}_i - \left\lfloor \frac{\tilde{\mathbf{d}}_{2i}}{3} \right\rfloor$
  $\mathbf{f}_{3i+1} = \tilde{\mathbf{c}}_i - \left\lfloor \frac{\tilde{\mathbf{d}}_{2i+1}}{3} \right\rfloor$
  $\mathbf{f}_{3i+2} = \tilde{\mathbf{d}}_{2i} + 2\mathbf{f}_{3i} - \mathbf{f}_{3i+1}$
end for

Figure 8: Test pictures used in Table 1.

integer binary Haar wavelet. In both cases, we only used eighty percent of the original data for reconstruction. The comparison shows that the integer ternary Haar wavelet is comparable with integer binary Haar wavelet, as the PSNR of both methods are very close. Table 1 provides the PSNR of both methods for the images shown in Figure 8.

Table 1: PSNR for the images in Figure 8 under integer ternary and binary Haar wavelets. The three rows per method provide the PSNR of each color channel: red, green, and blue.

Method	a	b	c	d	e	f
Ternary	34.2025	29.8641	25.4580	22.9446	25.9309	29.3136
	34.1923	29.8623	25.6380	22.7123	25.9668	29.3894
	34.1824	30.2141	25.7534	22.9040	25.9336	29.3652
Binary	36.8128	32.7858	28.6623	25.9970	28.6422	32.2297
	36.8058	32.7972	28.7183	26.0223	28.6733	32.2598
	36.7980	33.0755	28.8164	26.0532	28.6650	32.2405

It is possible to extend the ternary integer Haar wavelet to an arbitrary n-ary integer Haar wavelet for use with DGGSs of aperture n (refer to supplementary material). The extension to n-ary integer Haar wavelets and their proof of correctness are provided in supplementary material.

5 Vector Data Sets

Vector (or feature) data sets are defined as poly-lines, points, or polygons that describe geospatial features, such as road networks or the boundaries of countries and cities. Typically, the points making up these vectors exist on a sphere and are connected with geodesic arcs (i.e., great circle arcs), as geodesic arcs traverse the shortest path between two points on the surface of the sphere.

Since these data sets may be very large — consisting, for example, of millions of points — it is necessary to utilize a representation of these data sets that supports the efficient handling of relevant queries, such as buffering or data transmission. In the context of a DGGS, such a representation can be provided by rasterizing the vector into an image (rasterization), by association with cells at different resolutions (hierarchical cell representation), or by using wavelet transforms on the feature curves (spherical vector representation). In the following section, we describe each method in the context of an A3H DGGS and its benefits to different applications.

5.1 Rasterization

One approach that can be used to represent, store, transmit, and visualize vector data sets is to rasterize them into an image, and is similar to texture-based approaches for attaching vector data to a terrain [16]. Like the images described in Section 4.1, the pixels of these images each share a one-to-one correspondence with a cell in the DGGS, and can be assigned to the DGGS using the dual conversion. Furthermore, as in Section 4), the resulting images can then be overlaid on the globe for visualization, and can be effectively compressed and transmitted through the network using known techniques for image compression and transmission [48].

In order to generate these images such that they have the same dimensions as the cells in an A3H DGGS, the vectors are sampled using hexagons in a $3^n \times 3^n$ diamond-shaped tiling that is large enough to contain the features (where n depends on the resolution of the corresponding DGGS cells). Those cells that contain parts of the feature receive a color, while other cells are made to be transparent (see Figure 9). This technique can be used with DGGSs of different apertures if the dimensions of the images are modified accordingly.

While the cells representing a feature can usually be made fine enough to provide a good approximation of the feature, one problem with this representation is that the accuracy of the representation for the feature is fixed. Once the image is created using cells with a specific resolution, providing a more accurate representation of the feature requires reacquisition of the whole feature curve and creation of a different image.

5.2 Hierarchical Cell Representation

In a hierarchical cell representation, the cell structure of a DGGS is directly used to obtain a representation for vector data sets. The basic approach can be illustrated using quadtrees,

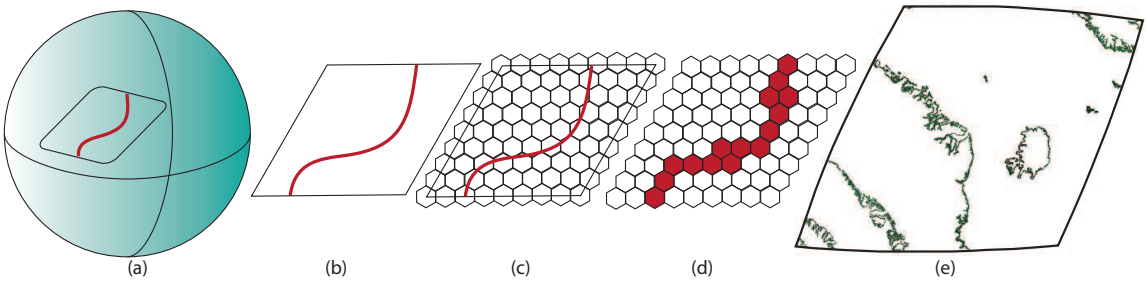


Figure 9: (a) Red feature and the diamond enclosing this feature on the sphere. (b) Projecting the feature and the diamond to the 2D diamond. (c) Sampling the feature using hexagonal cells. (d) Cells sharing a feature are colored. (e) A real example of a rasterized geospatial feature.

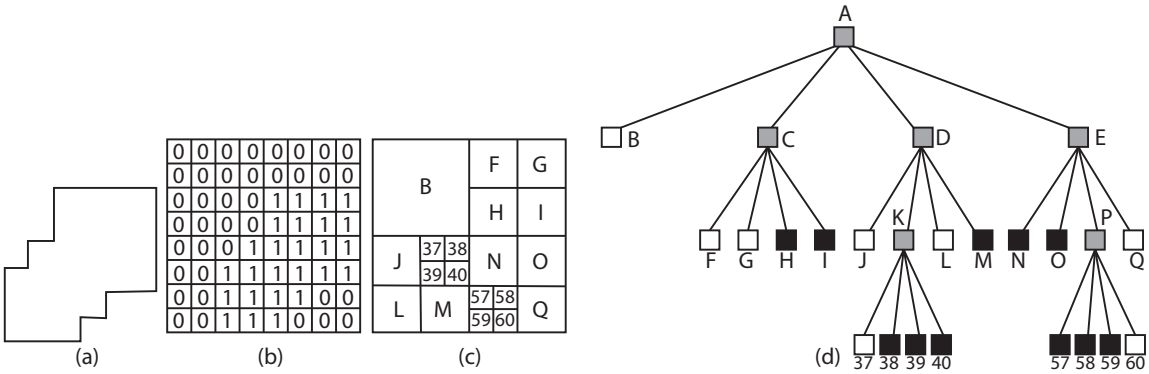


Figure 10: (a) A simple feature. (b) Cells that are inside or outside of the feature are assigned a number, 1 or 0, respectively. (c) Refinement of cells is used to approximate the feature. (d) Trees and coloring of the nodes. Image is a reproduction of an image in [44].

which are widely used to approximate features on an image [44]. Given a quadtree, its set of quad cells may be recursively refined until a good approximation of the feature is achieved. A cell is refined if the cell is intersected by the feature, such as the grey cell in Figure 10. If the cell is fully inside or outside of the feature, or if the cell size shrinks past a particular threshold, it is not refined any more.

We instead use a hierarchical tree structure compatible with PYXIS indexing (refer to Section 3). While the construction of the nodes of the tree is very similar to the quadtree method of [44], the refinement is instead a hexagonal 1-to-3 refinement (see Figure 11) with incongruent parent and child cells.

Deciding whether a cell is completely inside or outside a feature is more challenging, as the children of type A and B cells create a fractal coverage throughout the resolutions.

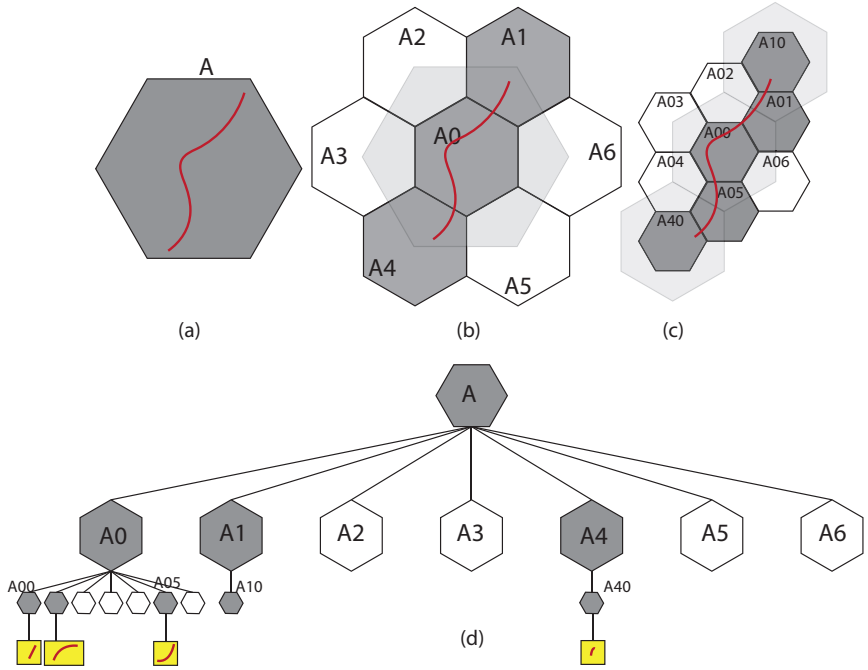


Figure 11: (a) Red feature on a coarse hexagon with index A. (b), (c) Cells are refined to approximate the feature. (d) Hierarchical tree associated with the refinement process. Leaves store the geometry of the curves if they contain points less than a threshold.

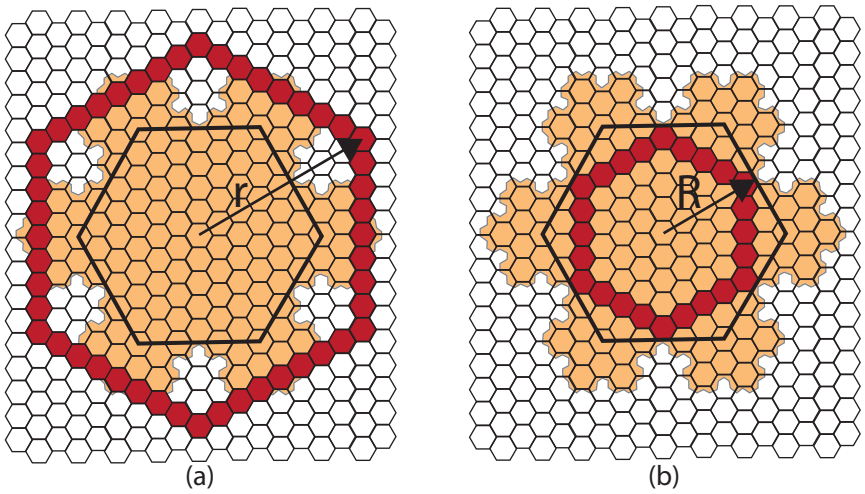


Figure 12: Minimum and maximum radius for the fractal coverage of type B cells.

To simplify this decision, the coverage of type A and B cells are approximated using two circles with different radii (see [40] for more details). Figure 12 illustrates the maximum and minimum radii for type B cells. If the circle associated with a cell lies completely outside or inside the feature, the cell is not refined anymore. Otherwise, it is considered to be a grey cell and refinement continues. Refer to Algorithm 5 for pseudo-code that describes this method.

This method produces rasterizations of the vector data, similarly to the rasterization approach described above, but at all resolutions of the DGGS instead of at a single resolution. However, no raster approach can perfectly capture the original vector geometry; hence inaccuracies will still result. In order to avoid this, the geometry of the vectors may be stored at the leaves of the tree. Then, if a portion of a high resolution curve needs to be visualized at its native resolution, only those feature points in the area of interest need to be retrieved while the remaining portions can be rendered in raster form by cutting the tree at a coarse depth.

5.3 Multi-scale Spherical Representation

A third approach to representing a feature curve is to provide a multi-scale representation/wavelet transform of the curve itself, rather than build a separate hierarchical structure on top of the feature curve. This is a general approach that can be used with any type of DGGS as an independent data structure, and is not specific to A3H DGGS.

One natural approach is to project the feature curve into an intermediate 2D domain (e.g. a map addressed using latitude/longitude coordinates (ϕ, θ)) using a spherical projection χ (see Figure 13). In this way, a feature curve can be projected to a 2D curve with

Algorithm 5 Hierarchical cell rasterization of a feature F into the cells of a DGGS. The maximum and minimum radii associated with a cell c are denoted $r(c)$ and $R(c)$, respectively.

```
for each cell c_0 at the coarsest resolution do
  cellsToCheck = {c_0}
  doubleCheck = {}
  for each cell c in cellsToCheck do
    Remove c from cellsToCheck
    d = the distance from F to the centroid of c
    if d > r(c) then
      if the centroid of c is in F then
        Mark c as black (inside the feature)
      else
        Mark c as white (outside the feature)
      end if
    else
      if d ≤ R(c) then
        Mark c as grey (intersected by the feature)
      else
        Add c to doubleCheck
      end if
      if c has children then
        Add the children of c to cellsToCheck
      else
        (Optional) points = the points in F that are contained in c
        (Optional) Store points in c
      end if
    end if
  end for
end for
for each cell c in doubleCheck do
  if any descendant of c is not white then
    Mark c as grey (intersected by the feature)
  else
    Mark c as white (outside the feature)
  end if
end for
```

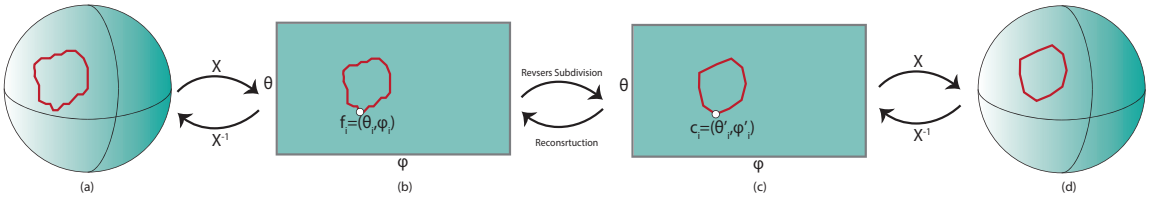


Figure 13: (a) A feature curve located on the sphere. (b) Projecting the sphere and the curve to the 2D domain (ϕ, θ) using function χ. (c) Using a wavelet decomposition, we can generate a low resolution version of the curve with vertices c. Using reconstruction, we can reconstruct the high resolution curve f. (d) The low resolution curve can be inverse projected to the sphere to obtain a coarse feature curve using function χ^{-1}.

coordinates in (ϕ, θ), on which we may apply a standard wavelet transform (e.g. the Haar wavelet) [42]. Such a wavelet transform maps fine points f (the feature curve) to coarse points c, where $f_i \in f$ has coordinates (θ_i, ϕ_i) and $c_i \in c$ has coordinates $(\acute{\theta}_i, \acute{\phi}_i)$. To obtain a coarse spherical curve, we can inverse project the coarse points c to the spherical domain using the inverse projection χ^{-1}.

However, as linear operations on geodetical coordinates (ϕ, θ) do not take into account the non-linear domain in which the coordinates reside, undesired distortions are produced. In particular, the shortest path between two points on the Earth generally maps to a curve in the (ϕ, θ) domain, and their midpoint on the Earth does not necessarily map to their midpoint in the (ϕ, θ) domain. This is especially true for features that cover large areas of the Earth, features whose points are located at the singularities of χ, or features that cross the boundaries of the (ϕ, θ) domain.

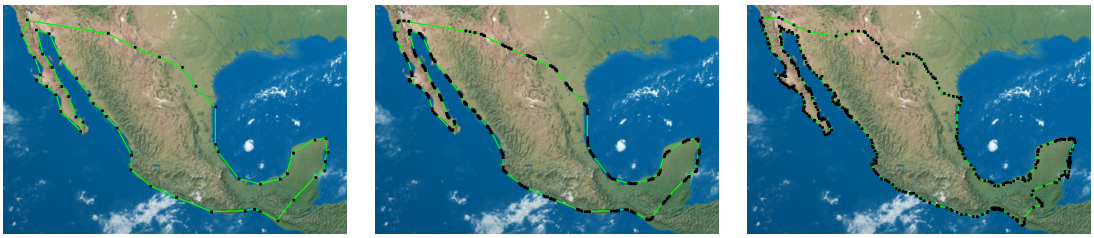
To avoid such artifacts, we can use spherical wavelet transforms to decompose and reconstruct feature curves directly in the spherical domain. In [2], a simple geometric construction for wavelet transforms on feature curves was introduced that is based on a modified Lane-Riesenfeld algorithm [18]. As the construction is composed entirely of SLERP (**s**pherical **l**inear int**erp**olation) operations, it is possible to increase or decrease the resolution of spherical curves without using intermediate domains (see Figure 14), and thus avoid any distortions due to projection mappings. A simple spherical wavelet transform (a spherical Haar wavelet) is described in Algorithms 6 and 7.

Algorithm 6 Decomposition of a feature curve $\mathbf{f}$ (of n points) into $\mathbf{c}$ and $\mathbf{d}$ via the spherical Haar wavelet.

for $i = 0$ to $\frac{n}{2}$, step 2 **do**
  $\mathbf{c}_i = SLERP(\mathbf{f}_{\in\rangle}, \mathbf{f}_{\in\rangle+\infty}, \frac{1}{2})$
  $\mathbf{d}_i =$ the rotation from $\mathbf{f}_{2i}$ to $\mathbf{c}_i$
end for

Algorithm 7 Reconstruction of a feature curve $\mathbf{f}$ (of n points) using $\mathbf{c}$ and $\mathbf{d}$ via the spherical Haar wavelet.

for $i = 0$ to $\frac{n}{2}$, step 2 **do**
  $\mathbf{f}_{2i} = \tilde{\mathbf{c}}_i$ rotated by $\mathbf{d}_i^{-1}$
  $\mathbf{f}_{2i+1} = \tilde{\mathbf{c}}_i$ rotated by $\mathbf{d}_i$
end for



(a) (b) (c)

Figure 14: Progressive refinement of a geospatial feature curve. (a) Coarse feature curve. (b), (c) Subdivision of the feature in (a), without and with details.

(a) (b)

Figure 15: Adaptive refinement of a geospatial feature curve. (a) Coarse feature curve. (b) The curve can be locally reconstructed to full detail as necessary.

As with other wavelet transforms, this approach is loss-less and can be used to perfectly reconstruct a feature vector after decomposition. Furthermore, as it operates on local neighborhoods, one can build a good coarse approximation of the feature and then reconstruct portions of the curve in full detail on demand. It is therefore useful for rendering spherical curves on top of a textured globe; both for displaying the overall shape of the feature at a distance and a portion of the feature in detail when zoomed in (see Figure 15). Such a multi-scale representation of geospatial vector curves has the potential to additionally prove beneficial in handling queries on the vector data.

5.4 Selecting a Representation

Each of these representations carries its own advantages and disadvantages, and therefore selecting an appropriate vector data representation depends a great deal on the queries one expects to encounter. For instance, rasterization provides image representations of feature vectors, which are compatible with the cell structure of a DGGS and can make use of the efficient transmission and visualization techniques that exist for images. However, rasterized vectors have a fixed resolution and performing geometric queries on these representations can be difficult and inaccurate. Specifically, the error can be as large as the radius of the largest cell at the target resolution. This error additionally propagates to line representations (see [4] and [33] for line drawing algorithms in quadrilateral and hexagonal grids, respectively).

Cell-based representations, in comparison, can efficiently support vectors up to any resolution supported by a given Digital Earth. Theoretically, this representation offers arbitrarily small error, but as the resolution of a Digital Earth is usually capped at a maximum in practice, the error equates to that of rasterization at the maximum resolution (unless the vector geometry is additionally stored). Furthermore, this representation is very sensitive, as a slight modification to the feature vector has the potential to completely alter the structure of the hierarchical cell tree.

The geometry-based multi-scale spherical representation of features is not sensitive to perturbations and is not resolution dependent. Furthermore, spherical wavelet transforms support efficient data transmission, geometric query handling, and adaptive reconstruction. When fully reconstructed, this representation provides the same error as that of the original vector data, however the coarse approximations are constructed based on a spherical shape for the Earth that is less accurate than conventional ellipsoidal representations such as WGS 84. Additionally, this representation exists independently of the underlying DGGS cell structure; more research is needed to integrate this representation into the DGGS so that one may fully benefit from the DGGS structure and coordinate system.

6 Geospatial Quantitative Data Sets

Quantitative data sets encompass a variety of data sets that are usually presented as numerical numbers. For instance, the average age of the population living in a region, the number of endangered species in a region, and the rates of rain in certain cities are examples of quantitative data sets.

An important distinction that exists between quantitative data sets and imagery or vector data sets is in the number of data values assigned to a given cell. Whereas imagery and vector data sets associate a constant number of data values with each cell that depends on the representation (e.g. four color channel values in the case of an image), the number of data values per cell associated with a quantitative data set is variable and depends on the data set itself (e.g. one data value per endangered species). In other words, while imagery and vector data are solely distributed across space, quantitative data are both distributed across space and across a variety of non-spatial categories.

While it is possible to represent each of these data values as a separate image (for visualization purposes, especially), there are some statistical queries one may wish to run over the entire data set that do not benefit from multiscale representations over the spatial dimensions. One such query is the range query, in which the number of data values within a particular range is requested. Hence, it can be useful to consider multiscale representations for this additional non-spatial dimension of the data, which may be applied in conjunction with or separately from the spatial dimensions. While the Haar wavelet may be used for this purpose, an important goal in defining such a representation is to allow queries to be asked at the data's coarse scale without significant loss in accuracy, so that a user may be presented with a fast estimate while awaiting the final result.

In this section, we discuss some methods to represent quantitative data sets in a multiscale manner that are specifically tailored towards addressing range queries at different scales. These methods are general and can be used for any DGGS, but can be adapted for a DGGS with a particular refinement (e.g. an A3H DGGS) to establish a correspondence between the DGGS resolutions and the method structure. Note that as the methods outlined in this section are based on the Haar wavelet, using an n-ary Haar wavelet (see supplementary material) allows us to extend this adaptation to DGGSs of other apertures.

6.1 Wavelet Histogram

The first approach to managing quantitative data sets we discuss is the wavelet histogram, which has been used in data cubes [29]. Suppose that we are given a data set outlining how many people are of a certain age — ranging between age one and eight — and we are interested to ask queries such as "How many people are between the ages of one and six?". To answer such queries, in [29], the total number of people having an age less than or equal to a certain age n is calculated for each age n. This number is called the cumulative frequency (CF). Formally, the cumulative frequency of age n is found as $\sum_{i=1}^{n} f_i$ in which

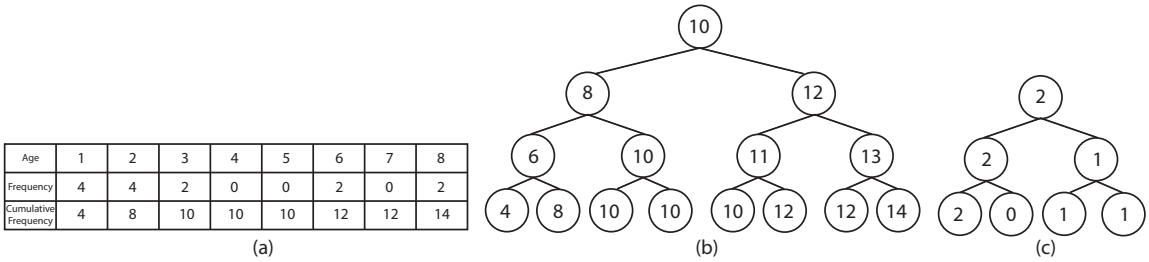


Figure 16: (a) Age data set and cumulative frequency. (b) Haar wavelet histogram. (c) Detail tree associated with the wavelet histogram in (b).

f_i is the frequency related to each age i (see Figure 16). Given the cumulative frequency, the answer to the query "How many people are between the ages of one and six?" can be found by subtracting the CFs of ages six and one.

Since the range of the data might be quite large, a tree structure called a *wavelet histogram* is built over the range of the data set to efficiently address these queries. The lowest level of the tree is built by placing CF values at each node. The higher levels can be created using wavelet decomposition. For example, if using the Haar wavelet, higher level nodes are formed by averaging two consecutive nodes.

Given such a tree, range queries can be estimated without accessing the entire data set (for instance, if the data set has not finished transmitting to a client). In the wavelet histogram, each node at any level of the tree provides an estimation for a range of data. Consider a data set with range $0 \leq i \leq n$ for which a wavelet histogram has been built. In general, the ith node at the kth level of this wavelet histogram ($0 \leq k \leq \log_2 n$ and $0 \leq i \leq 2^k - 1$) provides an estimate for all values in the range $(\frac{i \times n}{2^k}, \frac{(i+1) \times n}{2^k})$. For example, consider a situation in which we have only the second level of the tree in Figure 16 and we want to answer the query above: "How many people are between the ages of six and one?". Since six belongs to the right node of the tree at the second level; $(4 \leq 6 \leq 8)$ and one belongs to the left node $(0 \leq 1 \leq 4)$, we can consult the second level of the tree and estimate the answer as $12 - 8 = 4$. Naturally, estimates carry a certain amount of error; in this example, for which the correct answer is $12 - 4 = 8$, the error is equal to 4.

Note that it is not necessary to store the whole tree in order to reconstruct the data, as we can store the detail tree associated with the Haar wavelet decomposition, whose dimension is equal to the dimension of the initial data set. The detail tree is built using the nodes of the wavelet tree. If N is a non-leaf node of the wavelet tree, it has two children: N_l on the left and N_r on the right. The nodes of a (Haar) detail tree are defined as $N - N_l$. Since the leaves of a wavelet tree do not have any children, detail trees are one level shorter than wavelet histograms. Given only the root of the wavelet histogram alongside the complete detail tree, the entire data set can be reconstructed (see Figure 16 (c)).

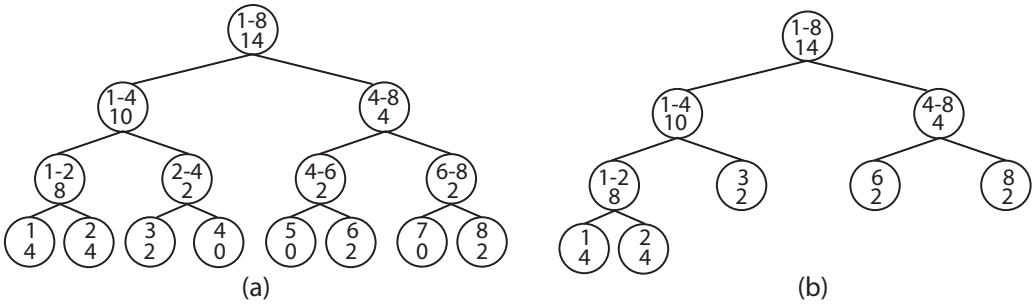


Figure 17: (a) Bisection in a nLT histogram. (b) Range tree after optimization.

Although wavelet histograms are powerful, their usefulness is limited by four main factors. First, if the data set consists of floating point numbers, the wavelet histogram cannot be applied to the data set. Second, if the range of the data is not a power of two, a complete tree cannot be made. Third, since wavelet histograms are created based on an specific range of numbers, combining two wavelet histogram is not possible without making a new tree. Fourth, the tree does not provide any information about the error and the error is not controllable.

6.2 Range Trees

An alternative method for classifying quantitative data sets and addressing range queries is known as the range tree, which is a modified versions of the nLT histogram [6]. In each node of a nLT histogram, three numbers are saved. Two of the numbers record the range of the data (i.e. the min and max of the range) and the last records the frequency of the data. As demonstrated in Figure 17 (a), the root of the tree describes the range of the entire data set and the sum of all frequencies. Each child node bisects the range of its parent and contains the frequency over its bisected range. Range trees differ from nLT histograms in one respect: in order to save memory, if the frequency of a node is zero or one, or if all of the numbers associated with a node are the same, bisection is not applied to the node (see Figure 17 (b)). To further optimize the structure, if a node is missing either its left or right branch, the node is replaced by its only child.

While this structure can support floating point numbers (unlike wavelet histograms), each node stores two floating point numbers and one integer, in comparison to the one float per node required by a wavelet histogram and associated detail tree. Hence, the amount of data used by range trees is higher than that used by wavelet histograms. A comparison between the amount of memory required by range trees and wavelet histograms is illustrated in Figure 18. For the data sets tested (with ranges (0, 1024) to (0, 8192)), the wavelet histograms required much less memory than the range trees.

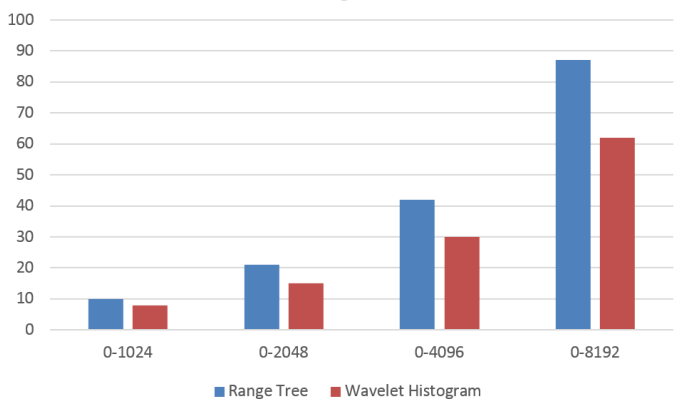


Figure 18: A comparison between the memory sizes of range trees and wavelet histograms for different data set ranges. It is clear that the wavelet histogram requires less space to store the whole data structure.

6.3 Modified Wavelet Histogram

With appropriate modifications, we can address the four issues that reduce the usefulness of wavelet histograms. The first two issues can be solved by binning. Suppose that the range of a data set is between min and max. We can distribute the data set into 2^n bins with $bin\_size = \frac{max-min}{2^n}$. This way, both integer and floating point numbers can be handled, and the number of bins is always a power of two. In order to solve the third problem, we can fix min, max, and bin_size based on the properties of the data set.

There are two strategies for choosing min and max. One is to choose an appropriate min and max based on the type of the data. For instance, given an age distribution data set, a sensible min and max could be zero and 100, as the majority of people fall within this age range. The second strategy is to set the maximum and minimum based on the data itself. For instance, given a data set of heights — $(135.5, 157.6, 165, 190.5, 199.5)$ — (see Figure 19), we can take $max = 199.5$, $min = 135.5$, and $n = 3$.

Both approaches suffer from some problems, however. For the first approach, it is possible to select an unnecessarily large range beyond what the actual data requires. For the second approach, if the data set changes, the bins may need to be resized and the data redistributed. As a result, additional properties of the data, such as how dynamic/prone to change it is, should be considered when selecting min and max.

Choosing a value for bin_size should also be based on properties of the data set. As a rule of thumb, bin_size should be selected in such a way that all the data located in a single bin can be processed and transmitted efficiently without the need for a multiresolution approach. However, it must always be a power of two, so that a complete wavelet histogram can be built. For example, in Figure 19, $n = 3$, $bin\_size = 8$, hence 8 bins are created.

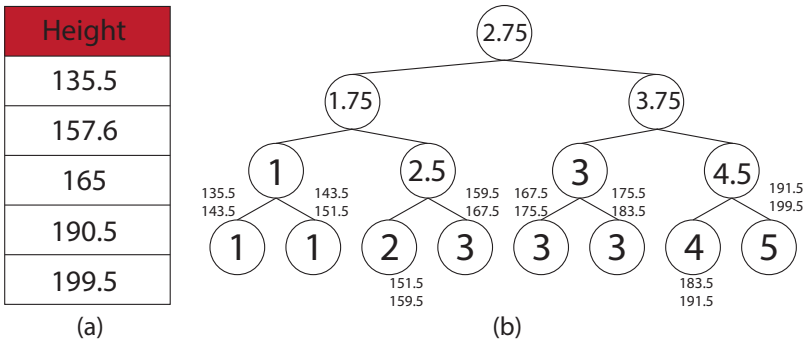


Figure 19: (a) Data set for the height of five people. (b) The wavelet tree created by the forming the bins with size 8. The range of each bin is shown beside each leaf node.

Although the Haar wavelet histogram is simple and can been improved upon in this way to better meet the needs of data representation in a Digital Earth framework, it does not provide any information about the error at any level of the tree, nor can the error be controlled. Here, the error — since each node of the tree summarizes all of its descendants — is the maximum difference between the value of the parent node and the values of its leaf descendants. (For instance, the error of the node with value 2.5 in Figure 19, whose children have values of 2 and 3, is 0.5.) In order to overcome this issue, we have developed a novel method called the Least Squares Wavelet.

6.4 Least Squares Wavelet

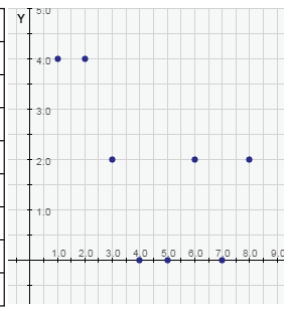
In the Least Squares Wavelet, we first approximate data with a piece-wise linear function with a known and reasonable error and then progressively improve the estimate by reconstructing the data using the known error. This methodology is especially useful in data transmission (an important process in any Digital Earth), allowing queries to be estimated with a known maximum error and then refined as the individual error values are transmitted.

Consider n data points (f_i, $0 \leq i < n$) that need to be retrieved or transmitted. We propose that these data points be approximated with piecewise linear functions (connected by control points p_i, $0 \leq i < m$) by solving a least squares system (see Figure 22). Consider a system of equations $\Phi\mathbf{p} = \mathbf{f}$ in which $\mathbf{p}$ and $\mathbf{f}$ are vectors composed of p_i and f_i and Φ is an $n \times m$ matrix with linear basis functions. The solution of this system can be found using the normal equation: $\mathbf{p} = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{f}$ [13]. Figure 20 illustrates a simple example, featuring a single-piece linear function to approximate the data set from Figure 16.

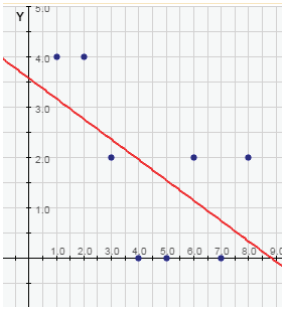
The error (or residual) of this approximation is given by $\mathbf{r} = \mathbf{f} - \Phi\mathbf{p}$. For example, in Figure 20, solving a least squares system for two points ($m = 2$) produces $p_0 = 3.166$ at age 1 and $p_1 = 0.337$ at age 8, which form a line passing through the data set. To

Age	Frequency
1	4
2	4
3	2
4	0
5	0
6	2
7	0
8	2

(a)



(b)



(c)

Age	Estimation
1	3.166
2	2.761
3	2.356
4	1.951
5	1.546
6	1.142
7	0.736
8	0.337

(d)

Age	Erro(r_i)
1	0.834
2	1.239
3	-0.356
4	-1.951
5	-1.546
6	0.858
7	-0.736
8	1.663

(e)

Figure 20: (a) Data set for the age of eight people. (b) Plot of the data set in (a). (c) The least squares line passing through the data set. (d) Estimation of the data set in (a) by evaluating the linear function in (c). (e) The error, or residual, of the data evaluation.

evaluate the error, we can subtract the data set values from corresponding values on the linear function $l(t) = (1-t)p_0 + tp_1$. Therefore, the error for the ith data value, f_i $(0 \leq i \leq 7)$, is $r_i = f_i - l(\frac{i}{7})$ (see Figure 20 (e)). To evaluate the total error, we can use different norms, but the least squares solution minimizes norm-2, $\|\mathbf{r}\|_2 = \sqrt{r_0^2 + ... + r_n^2}$. By increasing the number of control points, m, we can reduce the total error and obtain a better approximation of the data. Therefore, by evaluating $\|\mathbf{r}\|_2$, we can gauge the error of our approximation and decrease it as necessary, as opposed to setting up the histogram on the actual data without control over the error.

Now we must determine how to gradually add more information to the data set and reduce the error. To do so, we propose applying the wavelet histogram to the residuals instead of the data itself (see Figure 21). We can then gradually send nodes of the tree at different levels and improve the initial results obtained from the least squares solution.

Formally, suppose that we are given the residual $\mathbf{r}$ and the operators of a known wavelet transform: the decomposition operators (a reverse subdivision operator A and detail calculation operator B) and the reconstruction operators (a subdivision operator P and detail restoration operator Q). For the sake of convenience, let P^i, Q^i, A^i, and B^i be, respectively, the matrices corresponding to i applications of the operators P, Q, A, and B. We can decompose $\mathbf{r}$ into a coarse residual $\mathbf{c}^{[k]} = A^k\mathbf{r}$ and details $\mathbf{d}^{[i]} = B^i\mathbf{r}$ $(i = k, k-1, ..., 0$; see Algorithm 8 for the case of the Haar wavelet).

Note that the residuals can be approximated at different resolutions using $\mathbf{c}^{[i-1]} = P\mathbf{c}^{[i]} + Q\mathbf{d}^{[i]}$ (with $\mathbf{c}^{[0]} = \mathbf{r}$) (see Algorithm 9 for the reconstruction of $\mathbf{r}$ and $\mathbf{f}$). Although the coarse residuals $\mathbf{c}^{[i]}$ (for $i \neq 0$) have fewer entries than $\mathbf{r}$, $\mathbf{c}^{[i]}$ can be repeatedly subdivided to obtain a residual approximation $\hat{\mathbf{r}} = P^i\mathbf{c}^{[i]}$. As the root of the Haar wavelet histogram, $\mathbf{c}^{[k]}$, is the average of the residuals (which are always zero), $\mathbf{c}^{[k]}$ is not needed to reconstruct the residuals.

An important property resulting from using Haar wavelets to generate the residual tree

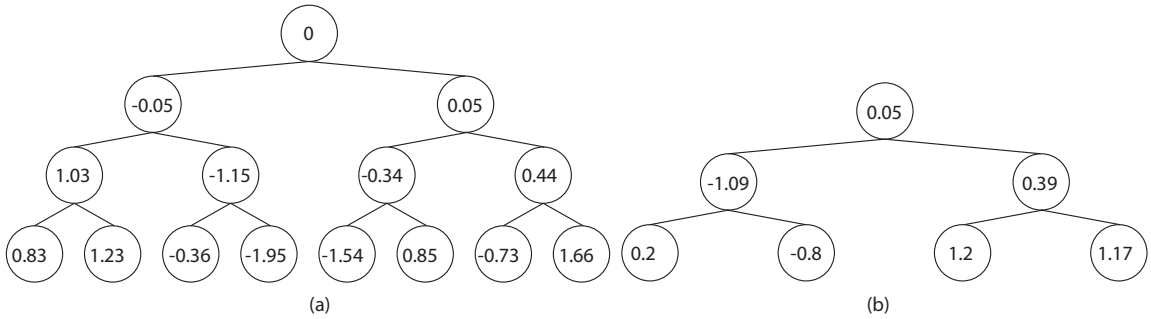


Figure 21: (a) Haar wavelet tree for residuals. (b) Detail tree for residuals.

Algorithm 8 Decomposition of $\mathbf{r}$ into $\mathbf{d}_k, \mathbf{d}_{k-1}, ..., \mathbf{d}_0$ ($\mathbf{c}_k = 0$).

for $j = 0$ to k, step 1 **do**
  $\mathbf{temp}$ = new vector of size $\frac{sizeof(\mathbf{r})}{2}$
  for $i = 0$ to sizeof($\mathbf{r}$), step 2 **do**
    $\mathbf{temp}(\frac{i}{2}) = \frac{\mathbf{r}(i)+\mathbf{r}(i+1)}{2}$
    $\mathbf{d}_j(\frac{i}{2}) = \mathbf{r}(i) - \mathbf{temp}(\frac{i}{2})$
  end for
  $\mathbf{r} = \mathbf{temp}$
end for

Algorithm 9 Reconstruction of $\mathbf{f}$ after receiving $\mathbf{d}_k, \mathbf{d}_{k-1}, ..., \mathbf{d}_0$.

for $j = k$ to 0, step -1 **do**
  $\mathbf{temp}$ = new vector of size $2 \times sizeof(\mathbf{r})$
  for $i = 0$ to sizeof($\mathbf{r}$), step 1 **do**
    $\mathbf{temp}(2i) = \mathbf{r}(i) - \mathbf{d}(i)$
    $\mathbf{temp}(2i+1) = \mathbf{r}(i) + \mathbf{d}(i)$
  end for
  $\mathbf{r} = \mathbf{temp}$
end for
$\mathbf{f} = \Phi\mathbf{p} + \mathbf{r}$

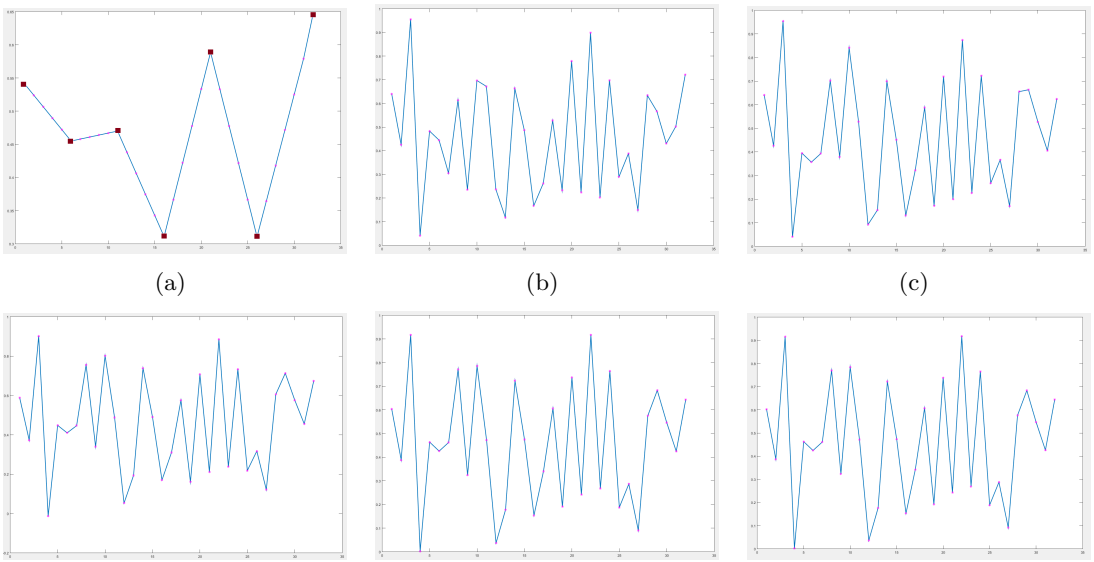


Figure 22: (a) Estimating the data set using piecewise linear functions. Control points are shown as squares. (b), (c), (d), (e) Adding a portion of the residual to the piecewise linear functions using a Haar wavelet tree, until the data set is perfectly reconstructed in (f).

is that the error of the approximation is bounded. That is, for a given $0 \leq i \leq k$ (with $\hat{\mathbf{r}} = P^i \mathbf{c}^{[i]}$), $\|\mathbf{r} - \hat{\mathbf{r}}\| \leq \|\mathbf{r}\|$. To prove this, consider (without loss of generality) the first entry of $\mathbf{c}^{[i]}$, which we will denote as c. By construction of Haar multiresolution, c is the average of the first 2^i entries of $\mathbf{r}$, and the first 2^i entries of $\hat{\mathbf{r}}$ are duplicates of c. Now, we can see that

$$\begin{aligned} c^2 &\leq 2c^2 \\ c^2 &\leq 2c\frac{r_0 + r_1 + ... + r_{2^i}}{2^i} \\ 2^i c^2 &\leq 2c \cdot r_0 + 2c \cdot r_1 + ... + 2c \cdot r_{2^i} \\ (c^2 - 2c \cdot r_0) + (c^2 - 2c \cdot r_1) + ... + (c^2 - 2c \cdot r_{2^i}) &\leq 0. \end{aligned}$$

By adding $r_0^2 + r_1^2 + ... + r_{2^i}^2$ to both sides, this becomes

$$(r_0 - c)^2 + (r_1 - c)^2 + ... + (r_{2^i} - c)^2 \leq r_0^2 + r_1^2 + ... + r_{2^i}^2.$$

As this was found without loss of generality, it holds for all entries of $\mathbf{c}^{[i]}$, hence for all entries of $\hat{\mathbf{r}}$ and $\mathbf{r}$. Therefore, $\|\mathbf{r} - \hat{\mathbf{r}}\| \leq \|\mathbf{r}\|$. Table 2 illustrates an example in which the error has been reduced by adding the nodes of the Haar residual tree to the data set (the data is the same as in Figure 22).

Table 2: Adding the nodes of the Haar residual tree at different resolutions reduces the error. There is no change at the first resolution since the average of the residuals is always zero.

Resolution	Error
0	1.3461
1	1.3461
2	1.3391
3	1.3181
4	1.2497
5	0

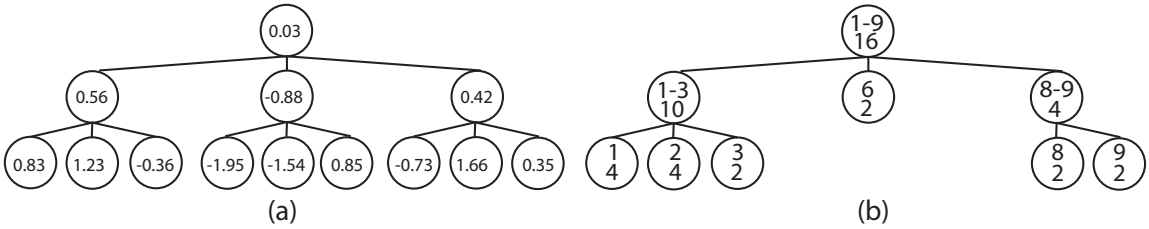


Figure 23: (a) Ternary Haar wavelet tree. (b) Ternary range trees.

6.5 Refinement Extension

The wavelet histogram tree and its associated binning method as discussed above are based on a binary Haar wavelet. Although a wavelet histogram can be built on the data independent from the underlying DGGS, it is also possible to adopt the tree and cell structure of the DGGS. In this case, the same factor of refinement as the DGGS can also be used for the wavelet histogram tree. For instance, given an A3H DGGS, a ternary Haar wavelet (in which each coarse node is found as the average of its nine children) can be used to build a wavelet histogram tree or range tree that is compatible with the DGGS cell structure (see Figure 23). For DGGSs with different refinements (aperture n), it is also possible to use the appropriate n-ary Haar wavelet to construct the wavelet tree (see supplementary material for the general Haar wavelet) or a range tree. Establishing a similar structure between these trees and the DGGS cell hierarchy supports the possibility of forming a correspondence between the resolution of the DGGS and the accuracy of the approximated data.

7 Conclusion and Future Work

In this paper, we reviewed different techniques — both new and pre-existing — to represent data sets in an A3H DGGS. Some of these methods (such as the least square wavelet,

modified wavelet histogram, and spherical vector representation) are general and can be applied to any DGGS. Our discussions of data representation were divided among different types of data — imagery/elevation, vector, and quantitative data sets — and began with an overview of data storage and retrieval using cell indices.

There are many potential directions that can be explored to improve upon the methods presented here. For instance, one may wish to determine the performance of these methods in combination with different queries. Finding the intersection or union of two vector data sets as represented in any of the forms described above, or the use of wavelet histograms for queries other than range queries, are two such examples. In addition, combining different representations of vector data sets, such as a combined multi-scale spherical representation and cell based representation, is an interesting research path. Finally, designing new multiresolution frameworks besides Haar that can support the compression of quantitative data sets while preserving the quality of the data (i.e. with minimal error) could be interesting for future work.

References

[1] Anupam Agrawal, M. Radhakrishna, and R. Joshi. Geometry-based mapping and rendering of vector data over LOD phototextured 3D terrain models. In *Proc. of WSCG '06*, 2006.

[2] Troy Alderson, Ali Mahdavi-Amiri, and Faramarz Samavati. Multiresolution on spherical curves. *Graphical Models*, 86:13 – 24, 2016.

[3] Aleksey Boyko and Thomas Funkhouser. Extracting roads from dense point clouds in large scale urban environment. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6):S2–S12, 2011.

[4] J. E. Bresenham. Seminal graphics. chapter Algorithm for Computer Control of a Digital Plotter, pages 1–6. 1998.

[5] M. A. Brovelli, M. Cannata, and U. M. Longoni. Managing and processing LIDAR data within GRASS. In *Proc. of the GRASS Users Conference*, volume 29, 2002.

[6] Francesco Buccafurri and Gianluca Lax. Fast range query estimation by n-level tree histograms. *Data & Knowledge Engineering*, 51(2):257 – 275, 2004.

[7] Kaushik Chakrabarti, Minos Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Approximate query processing using wavelets. *The VLDB Journal, The International Journal on Very Large Data Bases*, 10(2-3):199–223, 2001.

[8] Simon Clode, Franz Rottensteiner, Peter Kootsookos, and Emanuel Zelniker. Detection and vectorization of roads from LIDAR data. *Photogrammetric Engineering & Remote Sensing*, 73(5):517–535, 2007.

[9] A Fortier, D Ziou, C Armenakis, and S Wang. Survey of work on road extraction in aerial and satellite images. Technical report, Département de mathématiques et dínformatique, Université de Sherbrooke, 1999.

[10] M.-F. Auclair Fortier, D. Ziou, C. Armenakis, and S. Wang. Automated correction and updating of road databases from high-resolution imagery. *Canadian Journal of Remote Sensing*, 27(1):76–89, 2001.

[11] Minos Garofalakis and Phillip B Gibbons. Wavelet synopses with error guarantees. In *Proc. of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 476–487. ACM, 2002.

[12] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, 3rd edition, 1996.

[13] Google Inc. Google Earth. `http://earth.google.com`, 2017.

[14] Charles Han and Hugues Hoppe. Optimizing continuity in multiscale imagery. *ACM Transactions on Graphics*, 29(6):171:1–171:10, 2010.

[15] Oliver Kersting and Jürgen Döllner. Interactive 3D visualization of vector data in GIS. In *Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*, pages 107–112. ACM, 2002.

[16] M. Lambers and A. Kolb. Ellipsoidal cube maps for accurate rendering of planetary-scale terrain data. In *Proc. of the Pacific Conference on Computer Graphics and Applications*, PG '12, pages 5–10, 2012.

[17] Jeffrey M Lane and Richard F Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):35–46, 1980.

[18] Tao Li, Qi Li, Shenghuo Zhu, and Mitsunori Ogihara. A survey on wavelet applications in data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):49–68, 2002.

[19] P. Longley. *Geographic Information Systems and Science*. Wiley, 2nd edition, 2005.

[20] Ali Mahdavi-Amiri, Troy Alderson, and Faramarz Samavati. A survey of digital Earth. *Computers & Graphics*, 53, Part B:95 – 117, 2015.

[21] Ali Mahdavi-Amiri, Faraz Bhojani, and Faramarz F. Samavati. One-to-two digital Earth. In *Proc. of the International Symposium on Visual Computing*, ISVC '13, pages 681–692, 2013.

[22] Ali Mahdavi-Amiri, Erika Harrison, and Faramarz Samavati. Hierarchical grid conversion. *Computer-Aided Design*, 79:12 – 26, 2016.

[23] Ali Mahdavi-Amiri, Erika Harrison, and Faramarz F. Samavati. Hexagonal connectivity maps for digital Earth. *International Journal of Digital Earth*, pages 1–20, 2014.

[24] Ali Mahdavi-Amiri and Faramarz F. Samavati. Connectivity maps for subdivision surfaces. In *Proc. of GRAPP/IVAPP*, pages 26–37, 2012.

[25] Ali Mahdavi-Amiri and Faramarz F. Samavati. Adaptive atlas of connectivity maps. In *Proc. of the 8th International Conference on Curves and Surfaces*, Lecture Notes in Computer Science. Springer, 2014.

[26] Ali Mahdavi-Amiri and Faramarz F. Samavati. Atlas of connectivity maps. *Computers & Graphics*, 39:1 – 11, 2014.

[27] Ali Mahdavi-Amiri, Faramarz F. Samavati, and Perry Peterson. Categorization and conversions for indexing methods of discrete global grid systems. *ISPRS International Journal of Geo-Information*, 4:320–336, 2015.

[28] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-based histograms for selectivity estimation. In *Proc. of the 2002 ACM SIGMOD International Conference on Management of Data*, volume 27, pages 448–459. ACM, 1998.

[29] Helmut Mayer, Ivan Laptev, and Albert Baumgartner. Multi-scale and snakes for automatic road extraction. In *Computer Vision — ECCV'98*, volume 1407 of *Lecture Notes in Computer Science*, pages 720–733. Springer Berlin Heidelberg, 1998.

[30] J. B. Mena. State of the art on automatic road extraction for GIS update: a novel classification. *Pattern Recognition Letters*, 24(16):3037–3058, 2003.

[31] Microsoft Corporation. Bing Maps - Directions, trip planning, traffic cameras & more. `https://www.bing.com/maps`, 2017.

[32] Lee Middleton and Jayanthi Sivaswamy. *Hexagonal Image Processing: A Practical Approach*. Advances in Computer Vision and Pattern Recognition. Springer-Verlag London, 2005.

[33] Susanna Minasyan, Radomir Stankovic, and Jaakko Astola. *Computer Aided Systems Theory - EUROCAST 2009: 12th International Conference, Las Palmas de Gran Canaria, Spain, February 15-20, 2009, Revised Selected Papers*, chapter Ternary Haar-Like Transform and Its Application in Spectral Representation of Ternary-Valued Functions, pages 518–525. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[34] OGC. GML in JPEG 2000 for geographic imagery encoding — OGC. `http://www.opengeospatial.org/standards/gmljp2`, 2017.

[35] Perry Peterson. Close-packed, uniformly adjacent, multiresolutional, overlapping spatial data ordering. US Patent 8,400,451 (issued March 19, 2013), 2004.

[36] PYXIS innovation Inc. PYXIS Studio. `https://www.pyxisglobe.com/`, 2017.

[37] Zhiyuan Qiao, Jingnong Weng, Zhengwei Sui, Heng Cai, and Xuzhao Zhang. A rapid visualization method of vector data over 3D terrain. In *Proc. of the 19th International Conference on Geoinformatics*, pages 1–5, 2011.

[38] F. Rottensteiner. Automatic generation of high-quality building models from LIDAR data. *IEEE Computer Graphics and Applications*, 23(6):42–50, 2003.

[39] Kevin Sahr. Location coding on icosahedral aperture 3 hexagon discrete global grids. *Computers, Environment and Urban Systems*, 32(3):174–187, 2008.

[40] Kevin Sahr, Denis White, and A. Jon Kimerling. Geodesic discrete global grid systems. *Cartography and Geographic Information Science*, 30(2):121–134, 2003.

[41] Faramarz F. Samavati and Richard H. Bartels. Multiresolution curve and surface representation: reversing subdivision rules by least-squares data fitting. *Computer Graphics Forum*, 18:97–119, 1999.

[42] Hanan Samet. Using quadtrees to represent spatial data. In *Computer Architectures for Spatially Distributed Data*, pages 229–247. Springer, 1985.

[43] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[44] Arne Schilling, Jens Basanow, and Alexander Zipf. Vector based mapping of polygons on irregular terrain meshes for web 3D map services. In *Proc. of the 3rd International Conference on Web Information Systems and Technologies*, WEBIST '07, pages 198–205, 2007.

[45] Martin Schneider, Michael Guthe, and Reinhard Klein. Real-time rendering of complex vector data on 3D terrain models. In *Proc. of the 11th International Conference on Virtual Systems and Multimedia*, pages 573–582, 2005.

[46] A.K. Shackelford and C.H. Davis. Fully automated road network extraction from high-resolution satellite multispectral imagery. In *Proc. of the IEEE International Geoscience and Remote Sensing Symposium, 2003*, volume 1 of *IGARSS '03*, pages 461–463, 2003.

[47] Jaya Shukla, Manoj Alwani, and Anil Kumar Tiwari. A survey on lossless image compression methods. In *Proc. of 2nd International Conference on Computer Engineering and Technology*, 2010.

[48] J. P. Snyder. An equal area map projection for polyhedral globes. *Cartographica*, 29:10–21, 1992.

[49] Southern Terra Cognita Laboratory. DGGRID software - Discrete global grids. `http://www.discreteglobalgrids.org/software/`, 2017.

[50] Eric J. Stollnitz, Tony D. Derose, and David H. Salesin. *Wavelets for computer graphics: theory and applications*. Morgan Kaufmann Publishers Inc., 1996.

[51] Xiaochong Tong, Jin Ben, Ying Wang, Yongsheng Zhang, and Tao Pei. Efficient encoding and spatial operation scheme for aperture 4 hexagonal discrete global grid system. *International Journal of Geographical Information Science*, 27(5):898–921, 2013.

[52] A. Vince. Indexing the aperture 3 hexagonal discrete global grid. *Journal of Visual Communication and Image Represention*, 17(6):1227–1236, 2006.

[53] A. Vince and X. Zheng. Arithmetic and Fourier transform for the PYXIS multiresolution digital Earth model. *International Journal of Digital Earth*, 2(1):59–79, 2009.

[54] Jeffrey Scott Vitter and Min Wang. Approximate computation of multidimensional aggregates of sparse data using wavelets. In *ACM SIGMOD Record*, volume 28, pages 193–204. ACM, 1999.

[55] Jeffrey Scott Vitter, Min Wang, and Bala Iyer. Data cube approximation and histograms via wavelets. In *Proc. of the 7th International Conference on Information and Knowledge Management*, pages 96–104. ACM, 1998.

[56] Zachary Wartell, Eunjung Kang, Tony Wasilewski, William Ribarsky, and Nickolas Faust. Rendering vector data over global, multi-resolution 3D terrain. In *Proc. of the Symposium on Data Visualisation*, VISSYM '03, pages 213–222, 2003.

[57] Thomas Wendler. *Verfahren für die hierarchische Codierung von Einzelbildern in medizinischen Bildinformationssystemen*. PhD thesis, Aachen, 1987. Aachen, Techn. Hochsch., Diss., 1987.

[58] Denis White, Jon A. Kimerling, and Scott W. Overton. Cartographic and geometric components of a global sampling design for environmental monitoring. *Cartography and Geographic Information Science*, 19(1):5–22, 1992.

[59] Lance Williams. Pyramidal parametrics. *ACM SIGGRAPH Computer Graphics*, 17(3):1–11, 1983.

[60] Albert K. W. Yeung and G. Brent Hall. *Spatial Database Systems: Design, Implementation, and Project Management*. Springer Netherlands, 2007.

[61] C. Zhang. *Fundamentals of Environmental Sampling and Analysis*. Wiley, 2007.

[62] Mengyun Zhou, Jing Chen, and Jianya Gong. A virtual globe-based vector data model: quaternary quadrangle vector tile model. *International Journal of Digital Earth*, (ahead-of-print):1–22, 2015.