One-to-two Digital Earth

Ali Mahdavi Amiri, Faraz Bhojani, Faramarz Samavati

University of Calgary, Department of Computer Science

Abstract. The digital Earth framework is a multiresolution 3D model used to visualize location-based data. In this paper, we introduce a new digital Earth framework using a cube as its underlying polyhedron. To create multiresolution, we introduce two types of 1-to-2 refinement. Having a smaller factor of refinement enables us to provide more resolutions and therefore a smoother transition among resolutions. We also suggest two indexing methods specifically designed for quadrilateral cells resulting from 1-to-2 refinement. We finally discuss the equal area spherical projection that we are using in this framework to model the Earth as a sphere partitioned to equal area cells.

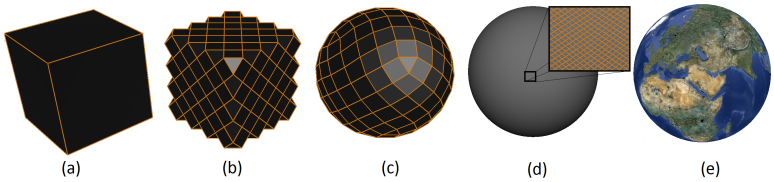

Fig. 1. (a)A cube (b) The cube after applying dual 1-to-2 refinement three times. (c) The refined cube is projected to the sphere using an equal area spherical projection. (d) The spherical cube at a high resolution. (e)The resulting spherical polyhedron is textured using a blue marble image. The blue marble texture is taken from [1].

1 Introduction

The digital Earth is a framework which represents the Earth as a multiresolution 3D model, and visualizes location-based data. Through this framework, users are able to zoom-in, zoom-out and analyze the data at different levels of detail. This framework has many applications in various fields such as computer science, cartography, and Geo-information systems. Google Earth is a particular well known and developed example of a digital Earth framework [2]. In this framework, the Earth is represented using traditional latitude longitude discretization of the Earth. As highlighted in [3], this representation creates issues regarding accuracy, replicability, and documentation.

It is possible to address these issues with a more regular representation of the Earth using a Geodesic Discrete Global Grid System (GDGGS) [4]. In GDGGS, the Earth can be approximated by a polyhedron whose faces are refined and projected to the sphere (Fig 1). These projected faces are called cells. Using a data structure, location-based data such as borders of countries, height maps, and texture data, are associated with these cells. In fact, a GDGGS has five fundamental parts: the base polyhedron, type of cells, type of refinement, spherical projection, and the employed data structure.

There are several viable polyhedrons that can be used to represent the Earth. In our framework, we use a cube as the underlying polyhedron to represent the Earth. This is reasonable, since a cube is the only polyhedron that can provide quadrilateral cells. Quadrilateral cells are more compatible with hardware and display devices and they are also compatible with familiar Cartesian coordinates.

It is common to use 1-to-4 refinement to support multiresolution or a hierarchical representation for the cells of a cube. In this refinement, each cell is divided into four cells by inserting a vertex in the midpoint of edges. As a result, the number of cells grow exponentially by a factor of four from one resolution to the next. However, choosing a refinement with smaller factor results in a more gradual change in the resolution. In our framework, we use 1-to-2 refinement as it is the minimum factor of refinement. This results in a smooth transition between resolutions, creating an efficient representation for geographic features.

In order to have a spherical representation of the Earth, we need to project faces of a refined polyhedron to the Earth using a spherical projection. Spherical projections may create angular and areal distortion. Equal area projections preserve the area. This projection is commonly used for the digital Earth frameworks since it eases the analysis of data associated with cells. As a result, our framework incorporates an equal area spherical projection that is designed specifically for the grids forming on faces of a cube (see Section 3.3 for more details) [5].

To associate data with cells and to handle adjacency and hierarchical queries, a spatial data structure is needed. In digital Earth frameworks, data can be assigned at very high resolutions. As a result common spatial tree structures might not be efficient [6]. Indexing methods are designed to replace the tree structure. In this paper, we suggest two indexing methods that are adapted for 1-to-2 refinements of a cube.

Our contribution is to provide a new digital Earth framework using a spherical cube as the Earth's representation. We introduce 1-to-2 refinements to provide multiresolution among the cells with the slowest factor of growth. We then provide two indexing methods for the cells, resulting from the refinement, and we suggest an equal area projection.

In Section 2, we discuss the related work of our method. Our proposed framework is introduced in Section 3. We then provide some results and discussion in Section 4. Conclusion and future work are presented in Sections 5.

2 Related Work

The Earth's surface can be partitioned into regular cells using a method called Geodesic Discrete Global Grid System (GDGGS)[7]. As we discussed earlier, GDDGS are distinguished based on their underlying polyhedron, type of cells, refinement, projection and their employed data structure [7]. We highlight some work related to each of these elements.

Underlying Polyhedrons: Different polyhedrons including platonic solids and truncated icosahedron have been used as an approximation of the Earth [8–11]. Among these polyhedra, spherical cubes have been traditionally used as the Earth representation [12] and they are still commonly used for spherical representations. Cubes are very popular since they can provide regular quadrilateral cells that are hardware-efficient, and adaptable with display devices, existing data structures, and familiar Cartesian coordinates. As a result, they are used as the base for a sphere in many applications such as terrain rendering, environmental mapping, game design, surface modeling, [13–16, 8] as well as the Earth representation [17, 8]. In this paper, we use a cube as an underlying polyhedron for a digital Earth framework.

Type of Cells: Cells of GDGGS can be hexagons, quads or triangles. For example, Dutton uses the triangular faces of an octahedron [10]. Some Digital Earth frameworks also employ hexagonal cells as the base cells (see [18] for a complete survey). However, as mentioned earlier, quadrilateral cells have been used more commonly. In this paper, we also use squares as the base cells of our proposed digital Earth framework.

Type of Refinement: Refinements are mostly applied to subdivision surfaces to create smooth graphical objects [19]. Refinement methods are also used to create levels of detail for a digital Earth framework. A 1-to-n refinement divides a cell with an area of A into cells with an area of $\frac{A}{n}$ where n is the *factor* of the refinement or *aperture* of the digital Earth framework [4].

It is common, for digital Earth frameworks that use a cube as the underlying polyhedron, to use 1-to-4 refinement. However, refinements with smaller factors are desirable for the digital Earth framework, since it is possible to provide more resolutions under the fixed maximum number of cells. 1-to-2 refinement provides the smallest factor of refinement among the refinements, therefore, it creates smooth transition among cells. In this paper, we introduce 1-to-2 refinement for constructing the resolutions.

Data Structure: To assign data to cells and handle necessary inquiries, a spatial data structure is required. The quadtree is one of the most common data structure for quadrilateral cells. To make quadtrees more efficient at high resolutions, some indexing methods have been proposed [20, 6]. Indexing methods can be constructed based on space filling curves, hierarchy of cells at successive resolutions, or a coordinate system defined for the cells [13, 20, 21]. We suggest two kinds of indexing methods based on the hierarchy and a defined coordinate system of cells resulting from 1-to-2 refinement on quadrilateral cells.

Spherical Projection: Digital Earth frameworks vary based on their employed spherical projection. Different projections such as conformal, gnomonic,

or equal area can be used to represent the Earth [22]. When a spherical projection is used, two types of distortion might appear: angular distortion and areal distortion. The projection that preserves the area is called equal area. This projection has been widely used for representing the Earth [17, 23, 5, 11]. Some equal area projections may reveal specific disadvantages. For example, they may create singularities at specific points [17], or iterative techniques, that slow down the handling of inquiries, are used to find its inverse relations [23, 24]. We suggest to use a spherical area projection that is specifically designed for cubes and has a closed form for both projection and inverse projection mappings [5].

3 Framework

In previous sections, we described the elements necessary for a digital Earth framework modeled by a Geodesic Discrete Global Grid System. As we discussed earlier, the underlying polyhedron of our framework is a cube and the cells are quadrilaterals. In this section, we describe other elements of our proposed framework. We first describe 1-to-2 refinement and explain some of its properties. Then, we discuss two possible indexing methods for such a framework. We eventually describe the projection that we use in our proposed digital Earth framework.

3.1 Refinement

Using the concept of lattices to analyze the behavior of refinement methods is very common [25]. To describe the refinement of our proposed method, we also use lattices. Consider a square regular lattice L_0 (see Fig 2(a)). In L_0, each vertex is connected by four edges to its nearest neighbors.

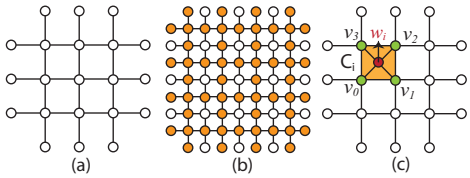

Fig. 2. (a) A portion of square regular lattice L_0. (b) 1-to-4 refinement applied on L_0. Orange circle are newly inserted vertices in L_0. (c) Vertices v_0, v_1, v_2, and v_3 create cell C_i. w_i is inserted in the midpoint of C_i.

The traditional 1-to-4 refinement is very common for quadrilateral cells (Fig 2(b)). However, as discussed earlier, having a refinement with lower factor is desirable. 1-to-2 refinements for quadrilateral cells have the smallest factor of refinement. Two types of 1-to-2 refinement are defined for L_0. Consider vertex v_0 and three vertices v_1, v_2 , and v_3 making cell C_i (Fig 2(c)). We insert a vertex w_i in the midpoint of C_i and connect w_i to its nearest vertices v_0, v_1, v_2 , and v_3 (Fig 2(c)) and then discard the old edges. If this refinement is applied on all cells, lattices illustrated in Fig 3 are obtained. This refinement is used by $\sqrt{2}$ subdivision for smoothing graphical objects [26].

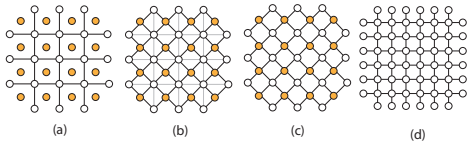

Fig. 3. (a) Inserting midpoints in all cells of L_0. (b) Connecting midpoints to their closest vertices. (c) Removing old edges. (d) The lattice after two iterations of 1-to-2 refinement.

The other 1-to-2 refinement for quadrilaterals is defined by inserting new vertices in the midpoint of edges. Again consider lattice L_0 and cell C_i with vertices v_0, v_1, v_2, and v_3. These vertices form edges e_0, e_1, e_2 , and e_3 as illustrated in Fig 4 (a). To refine cell C_i, vertex w_i is inserted in the midpoint of edge e_i ($0 \leq i \leq 3$). Afterwards, w_i is connected to w_{i+1} (w_3 is connected to w_0) in order to make a new cell. Finally, all edges e_i and vertices v_i are discarded. Fig 4 illustrates steps of refining cell C_i.

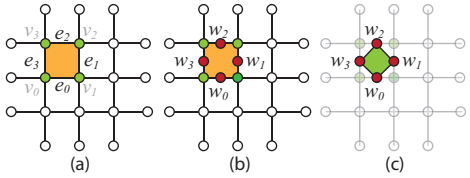

Fig. 4. (a) Edges creating cell C_i. (b) Inserting midpoints w_i at the midpoint of edges. (c) Connecting new vertices and creating a new cell.

Figure 5 illustrates application of 1-to-2 refinement on all cells of L_0. This type of 1-to-2 refinement is also created if we apply simplest subdivision method on a regular grid [27]. If we apply this refinement twice the lattice illustrated in Fig 5(d) is obtained.

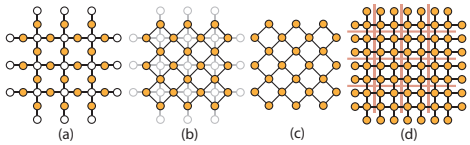

Fig. 5. (a) Inserting vertices at the midpoint of edges. (b) Connecting new vertices. (c) Discarding old vertices and edges. (d) Refining L_0 twice using 1-to-2 refinement. L_0 is illustrated in red.

To distinguish these 1-to-2 refinements, we call the first one *primal* (Fig 3) and the second one *Dual* (Fig 5). In dual 1-to-2 refinement, cells at two successive resolutions have the same midpoints. Such refinements are called *aligned* or *central place* [4]. These types of refinements have some advantages in compared to the alternative.

In GDGGS, the midpoints of cells at resolution r denoted by m_r are interpreted as a sample point of the entire cell. Accuracy, here means the error for representing points by cells, where error is the distance between m_r and a given point p. One of the advantages of dual 1-to-2 refinement is that increasing the resolution enhances the accuracy since the distance between m_r and p is decreased by increasing the resolution. This means $d_r \leq d_{r+1}$ where $d_r = \|p - m_r\|_2$ How-

ever, this characteristic is not guaranteed in some other refinements such as the common 1-to-4 refinement. Fig 6 illustrates this scenario and compares primal 1-to-4 refinement and dual 1-to-2 refinement.

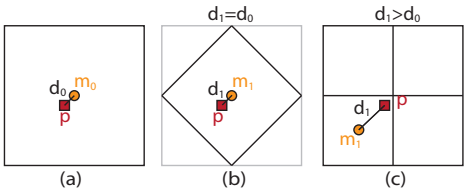

Fig. 6. (a) d_0 is the distance between m_0, and point p illustrated by the red square. (b) After one iteration of dual 1-to-2 refinement, the accuracy is at least as the previous resolution. (c) After one iteration of 1-to-4 refinement, d_1 is bigger than d_0. Increasing the resolution may decrease the accuracy in primal 1-to-4 refinement.

3.2 Indexing

Using indexing methods for handling spatial queries is common. These queries include determining the location of a cell, its neighbors and its position in the hierarchy of cells. Here, we suggest two possible indexing methods for dual 1-to-2 refinements. These indexing methods can be used for the primal 1-to-2 refinement with slight modifications. The first indexing method is similar to the hierarchy-based indexing designed for quadtrees [20] and hexagonal cells [4, 28]. The second proposed indexing is adopted based on the coordinates of cells at successive resolutions [29, 21, 4]. We call the first method, hierarchy-based indexing and the second one, coordinate-based indexing.

Hierarchy-based Indexing Method Hierarchy-based indexing methods are defined based on the relationships between cells at successive resolutions after the refinement. These indexing methods are very efficient in handling hierarchical queries. In our proposed indexing method, faces of the initial spherical cube are considered as the cells in resolution 0. In dual 1-to-2 refinement, for a cell C_r at resolution r, there is a cell C_{r+1} with the same midpoint (Fig 7(a)). C_{r+1} is called the *midpoint child* of C_r and C_r is called the parent of C_{r+1}. C_r has four other children whose midpoints are aligned with vertices of C_r (Fig 7(b)). These children are called *vertex children*. If cell C_r has index α at resolution r, its midpoint child has index $\alpha 0$ and four other children have indices αi, $1 \leq i \leq 4$, based on their position with respect to C_r. Fig 7(c) illustrates such an indexing for a cell at resolution r.

In hierarchy-based indexing, hierarchical access is handled by appending digits to a cell's index to access its children, or truncating a part of its index to access its parents. Neighbors of cells are found through a look-up table determining the algebra of such 1D indices. Vince [28] has provided such a look-up table for a similar indexing for hexagons resulting from refining an icosahedron. We can adapt it for the quadrilaterals resulting from the cube. In [28], an index is

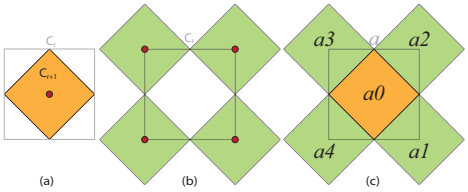

Fig. 7. (a) Cells C_r and C_{r+1} share the midpoint. (b) Vertex children of C_r are illustrated in green. (c) If C_r has index α, its children have indices as illustrated.

created using the elements of set $\Lambda = \{0, 1, 2, 3, 4, 5, 6\}$ since it indexes hexagonal cells with neighbors in 6 directions. However, in our method this set is changed to $\Lambda = \{0, 1, 2, 3, 4\}$ and therefore the look-up table is modified accordingly. Note that the singularities of an icosahedron are pentagons but singularities of a cube are triangles; both are located at the vertices of the initial polyhedron. Triangles produced at the corners are indexed similar to quadrilaterals but they have three neighbors.

Coordinate-based Indexing Method Coordinate based indexing is another type of indexing method in which faces of polyhedrons are typically unfolded onto the plane. An index is then defined for each cell by snapping the vertices or midpoint of cells on the integer coordinates [21, 11, 29]. In this section, we introduce coordinate-based indexing for quadrilaterals resulting from dual 1-to-2 refinement applied on the faces of a cube.

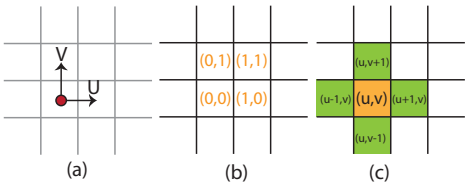

Fig. 8. (a) Coordinate system defined for cells of L_0 (b) Indices of cells. (c) Cell (u, v) (orange) and its neighbors (green).

To describe this indexing, we again use the concept of lattices for representing the connectivity of cells. Consider L_0 as previously defined in Section 3.1. We take two vectors with a 90° difference connecting midpoints of two adjacent cells as the unit vectors (U and V) making a coordinate system for cells (Fig 8). We can allow the midpoint of any cell to be the origin of the coordinate system. With respect to this coordinate system, midpoints of cells can get integer coordinates. These coordinates (u, v) are considered as the index of cells.

In this indexing, neighborhood finding queries are handled by simple algebraic operations. A cell with index (u, v), has neighbors $(u + 1, v)$, $(u, v + 1)$, $(u, v-1)$, and $(u-1, v)$ (Fig 8(c)). As well as neighborhood finding, the indexing scheme must be capable of handling hierarchical access queries. Therefore, we need to index cells at various resolutions and find a hierarchical access relation between cells at various resolutions.

We use L_r for the lattice obtained from applying r times of 1-to-2 refinement on L_0. To index L_1, we take the same origin as the one chosen for L_0. We define a coordinate system for L_1 by considering the vectors connecting midpoints of two adjacent cells as the main vectors of the coordinate system (Fig 9(a)). We index cells according to this new coordinate system. To distinguish cells at different resolutions, we use a subscript indicating the resolution. As a result, a cell with index $(u, v)_r$ is at resolution r and is u and v steps away from the origin, in the direction of the U and V axes respectively.(see Fig 9(b)).

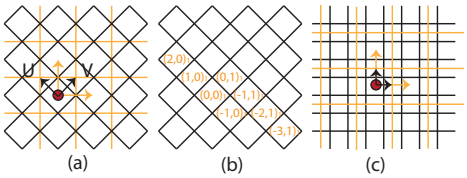

Fig. 9. (a) Coordinate system defined for cells of L_1. L_0 is illustrated in orange. (b) Indices of cells. (c) L_2 and its coordinate system. L_0 and its coordinate system are illustrated in orange. ● is the origin.

As shown in Fig 9, it is possible to choose coordinate systems for L_2 aligned with L_0. Similarly, we can choose coordinate systems for L_3 aligned with L_1. We can extend this property for further resolutions and pick a coordinate system for L_r aligned with L_0 or L_1 depending on if r is even or odd respectively. This property enables us to establish simple hierarchical relationships.

To establish a hierarchical relationship, we explain the transition from a parent to its children. To do this, we find the index of the midpoint child of a cell with index $(u, v)_r$. In fact, this task is equivalent to finding corresponding vectors of $(u, v)_r$ at resolution $r + 1$. Therefore, we can find the corresponding vectors of $(1, 0)_r$ and $(0, 1)_r$ at resolution $r + 1$ and multiply by u and v respectively.

Assume that r is even, therefore as illustrated in Fig 10 (a), $(1, 0)_r = (-1, 1)_{r+1}$ and $(0, 1)_r = (1, 1)_{r+1}$. Thus, $(u, v)_r = (v - u, u + v)_{r+1}$. For odd, we have the same relations (Fig 10(b)). Note that for any r, $(1, 0)_r = (2, 0)_{r+2}$ and $(0, 1)_r = (0, 2)_{r+2}$. Therefore $(u, v)_r = (2u, 2v)_{r+2}$. As a result, we can generalize hierarchical relationships for transitioning from resolution r to resolution $r + k$ by using Equation 1. $(s, t)_{r+k}$ denotes the index of the midpoint child of cell $(u, v)_r$ after k iterations of refinement.

$$(s,t)_{r+k} = \begin{cases} 2^k(u,v)_r & \text{if } k \text{ is even} \\ 2^{k-1}(v-u,u+v)_r & \text{if } k \text{ is odd} \end{cases} \tag{1}$$

The transition from a fine cell to a coarse cell is simply the reverse of the process. To apply this indexing method on a cube, we can index each face of the cube individually and handle the connectivity queries of boundaries using an edge based method that captures the connection of faces [29]. When a cube is refined by a dual 1-to-2 refinement, a triangle is formed at each corner. These triangles are indexed the same as quadrilaterals. The only difference is that they have three neighbors and three vertex children instead of four in the case

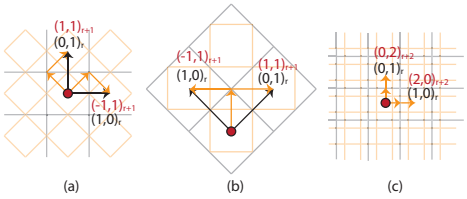

Fig. 10. (a) Equality of vectors when r is even. (b) Equality of vectors when r is odd. (c) Equality of vectors after two resolutions and r is even. ● is the origin.

of quadrilaterals. We can also flatten the polyhedron and index all faces in a 2D domain [21]. This way, triangles and boundary faces are traceable at all resolutions using hierarchical operations. Both methods are applicable for this application. We have implemented both and the results are comparable.

3.3 Projection

To form the spherical cells of our digital Earth framework, we use a spherical equal area projection. An equal area projection is a mapping from a domain Ω to another domain Δ while preserving the area. In our employed projection, Ω and Δ represent a cube and a unit sphere respectively, where both are centered at the origin and have the same area (the cube's edge has length $a = \sqrt{2\pi/3}$). In this projection, Δ is divided into six equal partitions by finding the intersection of planes $z = \pm x$, $z = \pm y$, and $x = \pm y$ with Δ. Fig 11(c) illustrates one partition of Δ in black.

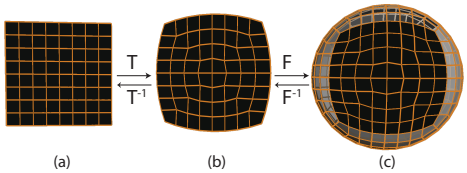

Fig. 11. Steps of the spherical projection. Points on a face f of Ω (a) are projected onto a curved square (b) and then projected onto a partition of the unit sphere Δ (c).

The main idea of the projection is to map each face f of Ω to a partition of Δ. To this end, an intermediate domain, called curved square, is used. As a result the projection has two steps. First f is projected to a curved square on the tangent plane of Δ, parallel to f, using an equal area bijection called T. Afterwards, the curved square is mapped to a partition of Δ using inverse Lambert Azimuthal equal area projection called F (Fig11). For the detailed discussion and derivations of mappings, you can refer to [5].

This projection is suggested due to its property of area preservation and its closed form definitions. One can use a different projection with different properties (such as conformality) based on application needs. Our proposed refinement and indexing methods are not dependent on the employed projection.

4 Results and Discussion

In this Section, we present some results of our framework. We illustrated the results of applying dual 1-to-2 refinement on the cube in Fig 1. Primal 1-to-2 refinement can also be used for our proposed framework (see Fig 12).

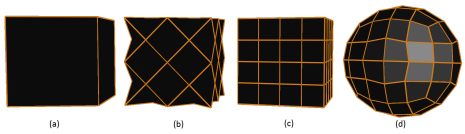

Fig. 12. (a) A cube. (b), (c) Applying primal 1-to-2 refinement three and four times respectively. (d) Projection of (c) to the sphere.

A digital Earth framework should support visualizing raster datasets such as images (Fig 1). In addition, it should support vector data. Vector datasets are coordinate-based data models that represent geographic features. Such features are typically provided as points, lines, and polygons. Fig 13(a) illustrates a vector dataset representing the boundaries of different countries. In GDGGS, each point is approximated by a cell enclosing the given point. Therefore, a feature can be represented by a sequence of cells.

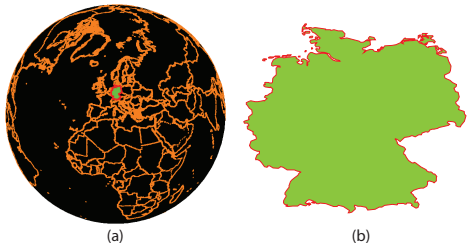

Fig. 13. (a) Vector data representing boundary of countries. Germany is highlighted in Green. (b) Boundary of Germany is zoomed. Data is taken from [30] and rendered in our framework.

To see the details of geographic features, data must be represented at different resolutions. For example, the boundary of Germany is highlighted in green in Fig 13 (a). However, as illustrated in Fig 13 (b) many details of this feature is not visible in the low resolution model. As a result, we need to have a resolution dependent accuracy. Accuracy, again means minimal error for representing feature points by cells.

In comparison to other frameworks [31, 28, 17], our framework provides an efficient representation for features by using the minimal factor of refinement. Consider that we want to have cells representing the boundary of Germany with an error less than $\approx 28Km$. Fig 14(a) illustrates cells representing the boundary with an error $\approx 35Km$ in red. To get the desired accuracy, our framework at the next resolution can approximate the feature points with 114 cells and an error $\approx 25Km$ which is good enough for our purpose (Fig 14(b)). However, if we use a 1-to-4 refinement (Fig 14(c)), although we have satisfied the required accuracy (error less than $\approx 28Km$), we need to save and render 162 cells (48 more cells

than 114.) As a result, using 1-to-2 refinement, we can save 42% of the cells for representing the boundary of Germany as an example of a geographic feature.

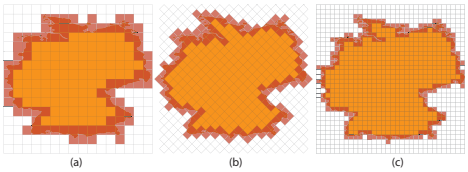

Fig. 14. Cells representing the boundary of germany with errors (a) $\approx 35Km$, (b) $\approx 25Km$, and (c) $\approx 17.5Km$

5 Conclusion and Future Work

In this paper, we have introduced a new Digital Earth framework modeled by a Geodesic Discrete Global Grid System. This framework uses a cube as its base polyhedron. We use two types of 1-to-2 refinement, which is the minimum factor of refinement. These refinements are used to create cells to discretize the cube's surface. To project such cells, we suggest a projection with closed forms, both for projection and its inverse. We also provide two types of indexing methods to handle hierarchical and neighborhood finding operations.

This framework can be used as the base of many other applications aiming to visualize location based information. Therefore, in this aspect, there are many directions for enhancing the features of the framework. However, there are two future works relevant to the framework's structure. The spherical projection that we are using should be compared with other alternatives such as Snyder projection in terms of time to report the exact difference. Moreover, the angular distortion of the used projection should be calculated and compared with other projections and polyhedrons.

References

1. : Map Projections. (`www.progonos.com/furuti`)
2. : Google Earth. (`http://earth.google.com`)
3. Goodchild, M.F., et al.: Next-generation digital earth. Proceedings of the National Academy of Sciences (2012)
4. Sahr, K., White, D., Kimerling, A.J.: Geodesic discrete global grid systems. Cartography and Geographic Information Science **30** (2003) 121–134
5. Rosca, D., Plonka, G.: Uniform spherical grids via equal area projection from the cube to the sphere. J. Computational Applied Mathematics **236** (2011) 1033–1041
6. Samet, H.: 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)
7. Goodchild, M.F.: Discrete global grids for digital earth. In: Proceedings of 1 st International Conference on Discrete Global Grids, March, 2000. (2006)

8. Cozzi, P., Ring, K.: 3D Engine Design for Virtual Globes. 1st edn. CRC Press (2011)
9. Wickman, F.E., Elvers, E., Edvarson, K.: A system of domains for global sampling problems. Geografiska Annaler. Series A, Physical Geography **56** (1974) 201–212
10. Dutton, G.H.: A Hierarchical Coordinate System for Geoprocessing and Cartography. Lecture Notes in Earth Sciences Series. Springer Verlag (1999)
11. Sahr, K.: Location coding on icosahedral aperture 3 hexagon discrete global grids. Computers, Environment and Urban Systems **32** (2008) 174–187
12. F.Chan, E.O'neill: Feasibility study of a quadrilateralized spherical cube earth data base. Technical report, EPRF, Silver Spring, Md. Computer Sciences Corporation. Environmental Prediction Research Facility. (1976)
13. Cignoni, P., Ganovelli, F., Gobbetti, E., Marton, F., Ponchio, F., Scopigno, R.: Planet-sized batched dynamic adaptive meshes (p-bdam). In: Proceedings of the 14th IEEE Visualization. VIS '03, IEEE Computer Society (2003) 147–155
14. Greene, N.: Environment mapping and other applications of world projections. Computer Graphics and Applications, IEEE **6** (1986) 21 –29
15. Compton, K., Grieve, J., Goldman, E., Quigley, O., Stratton, C., Todd, E., Willmott, A.: Creating spherical worlds. In: ACM SIGGRAPH 2007 sketches. SIGGRAPH '07, ACM (2007)
16. Grimm, C.M.: Simple manifolds for surface modeling and parameterization. In: Shape Modeling International, 2002. Proceedings. (2002) 237 –244
17. Alborzi, H.: Geometric issues in spatial indexing. Master's thesis, UNIVERSITY OF MARYLAND, COLLEGE PARK (2006)
18. K.Sahr: Hexagonal discrete global grid systems for geospatial computing. Archives of Photogrammetry, Cartography and Remote Sensing **22** (2011) 363–376
19. Cashman, T.J.: Beyond catmull-clark? a survey of advances in subdivision surface methods. Comput. Graph. Forum **31** (2012) 42–61
20. Gargantini, I.: An effective way to represent quadtrees. Commun. ACM **25** (1982) 905–910
21. Mahdavi-Amiri, A., Samavati, F.: Connectivity maps for subdivision surfaces. In: GRAPP/IVAPP. (2012) 26–37
22. W.Grafarend, E., W.Krumm, F.: Map projections: cartographic information systems. Springer (2006)
23. Snyder, J.P.: An equal area map projection for polyhedral globes. Cartographica **29** (1992) 10–21
24. Harrison, E., Mahdavi-Amiri, A., Samavati, F.: Analysis of inverse snyder optimizations. Transactions on Computational Science **16** (2012) 134–148
25. P.Ivrissimtzis, I., A.Dodgson, N., A.Sabin, M.: A generative classification of mesh refinement rules with lattice transformations. Comput. Aided Geom. Des. **21** (2004) 99–109
26. Li, G., Ma, W., Bao, H.: $\sqrt{2}$ subdivision for quadrilateral meshes. Vis. Comput. **20** (2004) 180–198
27. Peters, J., Reif, U.: The simplest subdivision scheme for smoothing polyhedra. ACM Trans. Graph. **16** (1997) 420–431
28. Vince, A., ZHENG, X.: Arithmetic and fourier transform for the pyxis multiresolution digital earth model. Int. J. Digital Earth **2** (2009) 59–79
29. Peters, J.: Patching catmull-clark meshes. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. SIGGRAPH '00 (2000) 255–258
30. : Thematic Mapping API. (`http://thematicmapping.org/`)
31. : PYXIS Innovation. (`http://www.pyxisinnovation.com`)