Implementing Atlas of Connectivity Maps for ICON Grid

Mohammad Imrul Jubair, *Student*

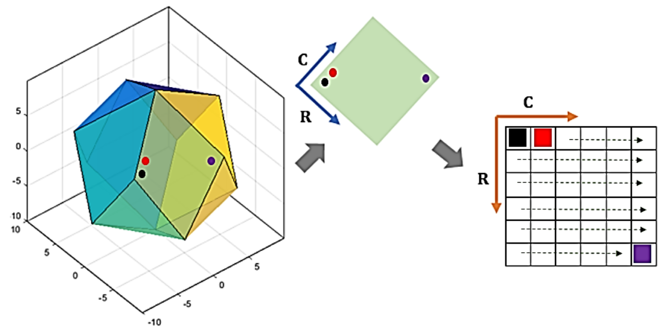


Fig. 1. mapping coordinate information of ICON grid into an array using ACM

Abstract— Mapping connectivity information of vertices and faces of geometric layout into a separate two dimensional domain is a popular data structure in the research area of Digital Earth. Such arrangement of connectivity provides a great advantages to work with geospatial data. In this paper, an approach is presented that captures the connectivity information of the vertices of ICON grid into a 2D domain. The paper implements the mothodology of Atlas of Connectivity Maps (ACM) where ICON grid is taken as an input and it generates an array where connectivity information of vertices are recorded. An user interface is also developed that demostrates the results and effectiveness of the implementation.

Index Terms— Digital Earth, ICON Grid, Atlas of Connectivity Maps, Geographic Coordinates

Introduction

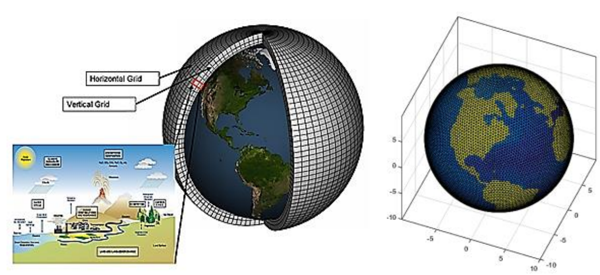
Computer-based globe model has become an important subject in the field of Meteorology to understand the behaviour of climate, e.g. prediction of climate performance for future. Research on climate typically depends on the large amount of geospatial data obtained from various kind of data acquisition process. *Fig. 2* shows an overview of data acquisition from different source of Earth and representation of the data on a digitized grid. Main challenge of computer-based globe system is to generate appropriate framework that presents this data in a structured manner. In most of the cases, such frameworks deal with discretizing Earth's surface into different geometric objects that are used to assign data. The cells represent areas that contains geospatial information related to the point of interest. Data in different cells can be visualized using proper colormap for further exploration [1] [2].

Different digital earth system use different geometric layout for discretized globe surface. Although there are established data structure to represent the geometric component, applying connectivity mapping on the vertices delivers more benefits, e.g. smooth traversal, accessing neighbourhoods etc. Atlas of Connectivity Maps [3] is an efficient approach where a data structure is introduced that maps the connectivity of semiregular models onto a 2D domains storing the connectivity of regular patches in semiregular models obtained from an arbitrary refinement. The connections between vertices and faces are captured by these 2D domains and their interconnections [3].

• *University of Calgary. E-mail: mohammadimrul.jubair@ucalgary.ca*

CPSC 601 Fall 2014 - Project Report
Submision date: 20 December 2014

In this paper, the ACM methodology is implemented on ICON Grid that reproduce the vertices on the globe surface into a data structure where coordinate information of each vertex is assigned in 2D array(*see Fig. 1*). The next sections of this paper discuss about the background study, explanation of ICON Grid connectivity information and the approach of implementing ACM on this grid. The outcome of the mapping and its usefulness is also discussed later. The paper presents pictorial demonstration via a user interface that visualize data based on the data structure obtained from connectivity mapping. Finally it concludes with future work followed by conclusion.



(a) (b)

Fig. 2. (a) shows an idea of geospatial data acquisition for cells on a computer-based globe surface. And (b) illustrates visualization of data on the grid

1 Background Study

Geometric entities are the most important matter in representing geospatial data on digitized earth surface. Different types of geometric components can be used to represent the earth grid, e.g. vertices, triangles, hexagons etc. As a result different framework stores the geospatial data based on different geometric objects; i.e. data can be stored in the centre of the cells, or in the edge vertices.

Fig. 3 shows example of different geometric components and the concept of reserving data on them.

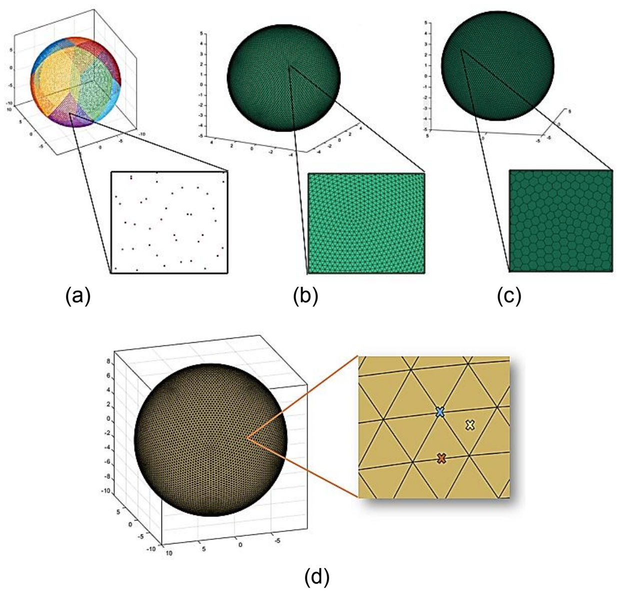


Fig. 3. Example of geometric components used to hold data in digitized Earth Grids. Here (a), (b) and (c) shows three different grid with vertices, triangle cells and hexagonal cells respectively. Additionally, (d) shows an example where data is reserved in vertices (blue cross), in center of the triangle (yellow) and the edge midpoint (red)

As this paper deals with ICON globe model, a study on its grid arrangement will be the topic of discussion here. ICON (ICOsahedral Non-hydrostatic) models is developed jointly by German Weather Service (DWD) and Max-Planck-Institute for Meteorology (MPI-M), and is used for numerical weather prediction as well as for future climate predictions. The dataset of ICON is a collection of *variables*, *dimensions* and *attributes*. The *variable* holds data and the geographic coordinates (latitude and longitude) of different vertices of different *dimensions*. The connectivity and interrelation among different variables are defined by *attributes*. There are six types of different vertices which are used to cover ICON globe grid, they are – centre vertices of the triangle cells, three vertices of triangle, edge midpoint vertices, four neighbour of the edge midpoint (which forms a rectangular shape), centre vertices of hexagonal cells (which actually lie on the centre of the triangle) and six vertices of the hexagon. Among those six, data reserving vertices are the centre of the triangle, edge midpoint and the centre of three hexagons. Fig. 4 illustrates the concept geometric and data storage arrangement for ICON Grid. The exploration of these vertices and data provides an initial concept for implementing Atlas of Connectivity Maps (ACM) on ICON's vertices. The next section of this paper describes the importance of applying ACM on ICON followed by the methodology of the implementation.

2 Atlas of Connectivity Maps for Icon Grid

In [3], the procedure to set up the atlas of connectivity maps for handling connectivity queries of quadrilateral semiregular models is presented in detail. The authors also describe how to modify the method to support triangular meshes. Modification concept is useful for this paper as the vertices on the ICON grid is generated by subdividing the initial icosahedron into several triangular cells, where their vertices of the triangles basically represent the geometric layout of the grid. Fig. 5 shows a brief overview of ACM for semiregular models for both quadrilateral and triangular model collected from [3].

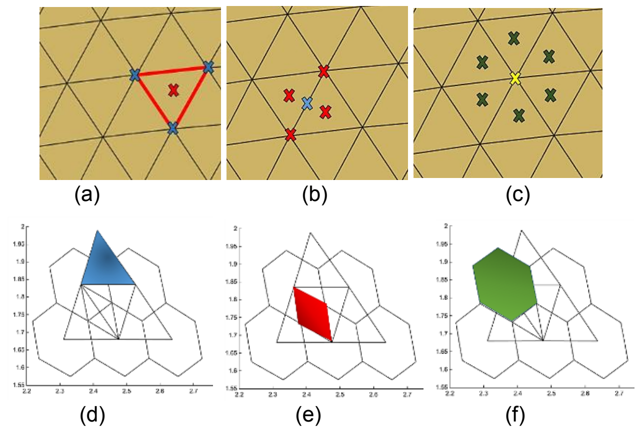


Fig. 4. Vertices used in ICON Grid - (a) center of the triangle (red cross) and three edge vertices (blue), (b) edge midpoint (light blue) and four neighbors as a rectangle (red), and (c) center of the hexagon (yellow) and six vertices of hexagons (green). Besides, (d) to (e) shows data storage at triangle, rectangles and hexagons respectively.

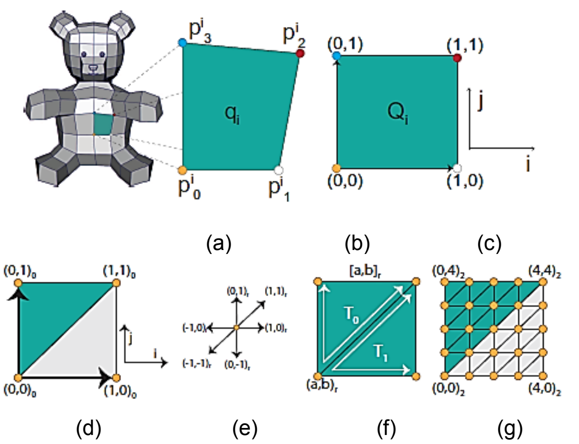


Fig. 5. (a) a quad mesh with M quads. (b) q_i is a quad in 3D. (c) q_i is mapped to its connectivity map Q_i and a coordinate system is defined for Q_i. Moreover, (d) Coordinate system for triangular connectivity maps. (b) Neighborhood vectors for triangles. (c) Form of triangular faces. (d) Applying two steps of 1-to-4 refinement on a triangular connectivity map.

2.1 Importance of Applying ACM on ICON

Atlas of Connectivity Map produces a 2D domain where the vertex information is assigned. The recording of three dimensional geographic coordinates to an array basically creates a two dimensional representation of the discretised Earth surface where each element in the index points the latitude and longitude of a vertex (see Fig. 6). This type of conversion creates a data structure that allows user to traverse through the vertices and to search for specific vertex information using some data structure operations. Besides, we can access neighbourhood information from the 2D domain. Moreover, working with multi-resolution, e.g. down-sampling and up-sampling becomes easier for this kind of connectivity mapping. Fig. 6 also illustrates the significance of ACM on ICON.

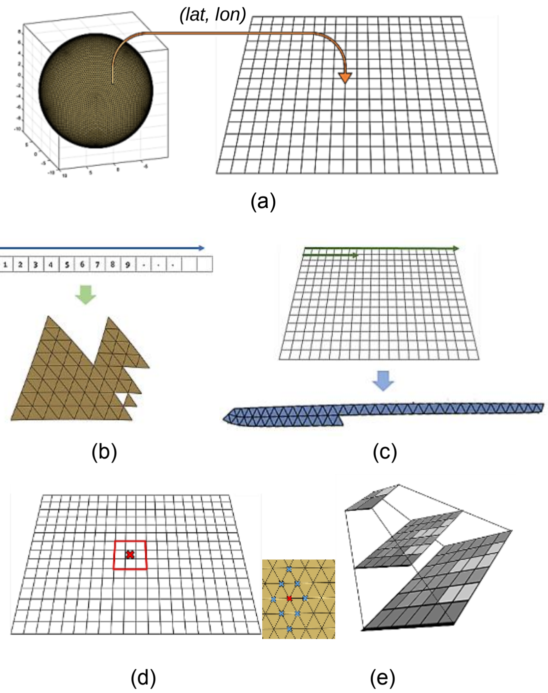


Fig. 6. (a) shows a conceptual representation of ICON Grid to 2D domain mapping. (b) to (e) illustrates the importance of ACM on ICON Grid. Here, (b) is the traversal through the vertices before performing mapping and (c) shows traversal on 2D array. (d) exhibits the neighborhood accessing. Finally, (e) shows the advantage of 2D domain in terms of muti-resolution.

2.2 Implementation

Methodology of ACM is implemented on ICON Grid goes through a pipeline. At first, regular icosahedron is generated from grid. The adjacent triangles of icosahedron are merged to form diamonds. Preliminary diamond is processed to map the first 2D domain. After that, a procedure is followed to implement the connectivity mapping for rest of the diamonds. Overall pipeline is presented in Fig. 7. In this section of the paper, the approach to perform connectivity aping on ICON is described in detail.

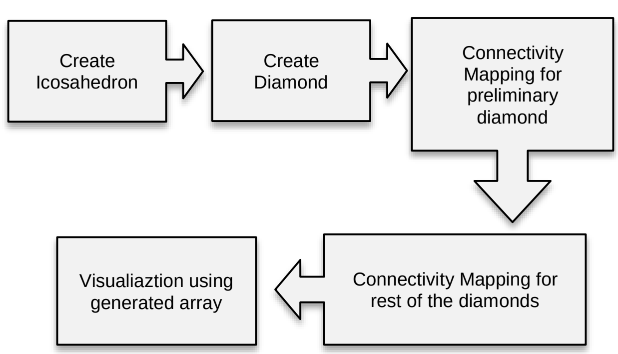


Fig. 7. Pipeline of implemeting ACM on ICON Grid

Building Icosahedron And Creating Diamonds: The procedure of building icosahedron depends on the hexagonal cells of ICON gird. While covering the Earth sphere with hexagons, irregular hexagons (polygons) is formed. There are 12 such polygons an each polygon refers to the *12* vertices of regular icosahedron. In order to find those *12* pentagons the variable that holds the *6* coordinates of hexagon vertices are examined. Outcomes of examination indicate that, the *5th* and *6th* coordinates are repeated for those pentagons and first five coordinates provides the five vertices of the pentagon. By using these pentagons, regular icosahedron is created. Two adjacent triangles are merged to form a diamond which is rectangle in shape [2]. Thus, ten diamonds are generated from the icosahedron and each diamond is treated as a region on the Earth surface that covers a collection of vertices. The target is to apply ACM on these vertices to keep their connectivity information into a two dimensional array. Fig. 8 illustrates the ocncept.

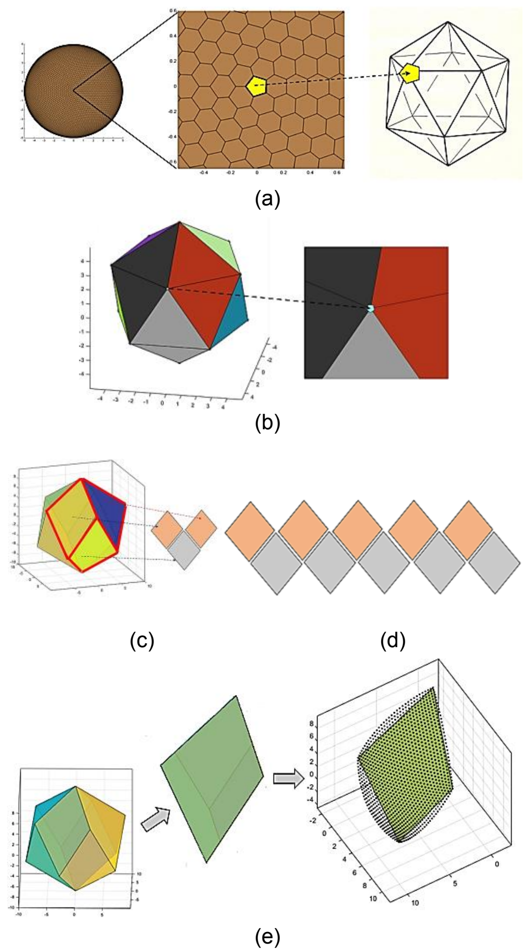


Fig. 8. (a) pentagons on the Earh sphere while covering by hexagons. (b) pentago refers to a vertex of regular icosahedron. (c) and (d) formation of 10 diamonds. (e) - each diamon is aregion on the Earth sphere coveirng a collection of vertices

CM for preliminary diamond: After generating the diamonds, the next step is to perform connectivity mapping for preliminary diamond. The preliminary dimond can be chosen arbitrarily. In this step an 2D array will be filled up with the vertices information (latitude, longitude) as shown in Fig. 1. The method of finding the corresponding vertices for a specific index on the array is illustrated in Fig.9. The initializtion of this method depends on the orientation of the pentagons. But in this case, the pentagons found for the icosahedron is not taken into account. Here, the triangles that shares the same icosahedron vertices are obtained and their vertices actually forms another pentagons which is different from the previously located ones. The newly esablished pentagons are actually rotated and scaled version of the previous ones. Fig. 10 shows the difference between these two kinds of pentagons.

An arbitrary pentagon on the preliminary icosahedron is selected. In this step, four initial vertices information are necessary to initiate the maping process. Basically, they are the vertices of any two adjacent triangls and are labeled as – *origin vertex (O), diagonal vertex (D), column vertex (C)* and *row vertex (R)*. Fig. 11 shows the indication of these vertex on a pentagon. At this stage, these four vertices are essential to start mapping a 2D array. If we need a $m \times n$ array D_{init} to map, then *origin vertex* is located on $D_{init}(0,0)$, *diagonal vertex* is in $D_{init}(1,1)$; and *column vertex* and *row vertex* have their entries in $D_{init}(0,1)$ and $D_{init}(1,0)$. Here, integer coordinates are used, and *m* and *n* are the rows and column numbers of the array. However, the origin of the coordinates is at upper left corner of the array.

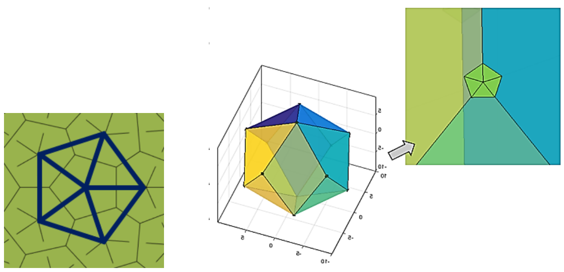


Fig. 10. Pentagons with blue borders *(left)* are newly formed pentagons from five triagles sharing the same vertices of

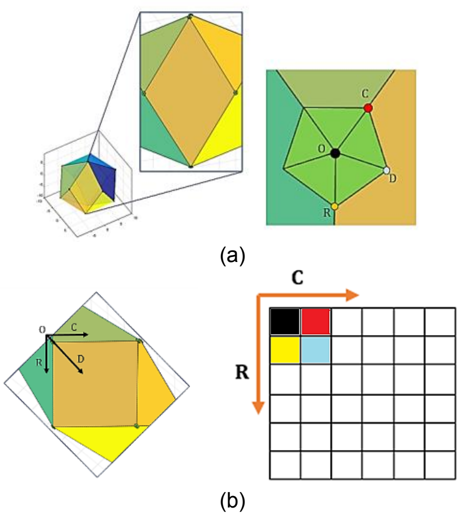


Fig. 11. Four initial vertices to start connectivity mapping. (a) shows their position in the pentagon and (b) shows their location in the 2D array. *Origin vertex* is denoted with *O* and marked with black and its situated in *(0,0)* of the array. *D* represents *diagonal vertex* (light blue) and it occupies *(1,1)* in the array. Similarly, *C* and *R* stands for *column vertex* and *row vertex* and their respective positions in the array are *(0,1)* and *(1,0)*

Next step is to initialize rows and columns of the array with corresponding coordinates of vertices. To do that, a simple iterative procedure is followed. Fig. 12 shows few iterations of the procedure as an example. At first, three vertices *O, C* and *D* are taken into account. Then the algorithm tries to find a triangle that shares *C* and *D,* but not *O*. The 3rd vertex of such triangle will give us next vertiex *V1*. The vertex *O* is marked as '*used'* and *V1*'s geogrpahic coordinate information is put into $D_{init}(1,2)$. Now *V1, C* and *D* can be used to find *V2* by assigning the searching condition where a triangle has edge with *C* and *V1*, but not *D*. The result is *V2* and $D_{init}(0,2)$ is filled up with its latitude longitude information and *D* is marked as '*used'*. According to the same fashion, $D_{init}(1,3)$ will have *V3*'s information that can be found using *V1, V2* and *C*. This itertive method can move on untile it reaches *n*, the number of column in D_{init}, and by doing this first to two rows of D_{init} is initialized which are $D_{init}(0,0)$ *to* $D_{init}(0,n)$, and $D_{init}(1,0)$ to $D_{init}(1,n)$.

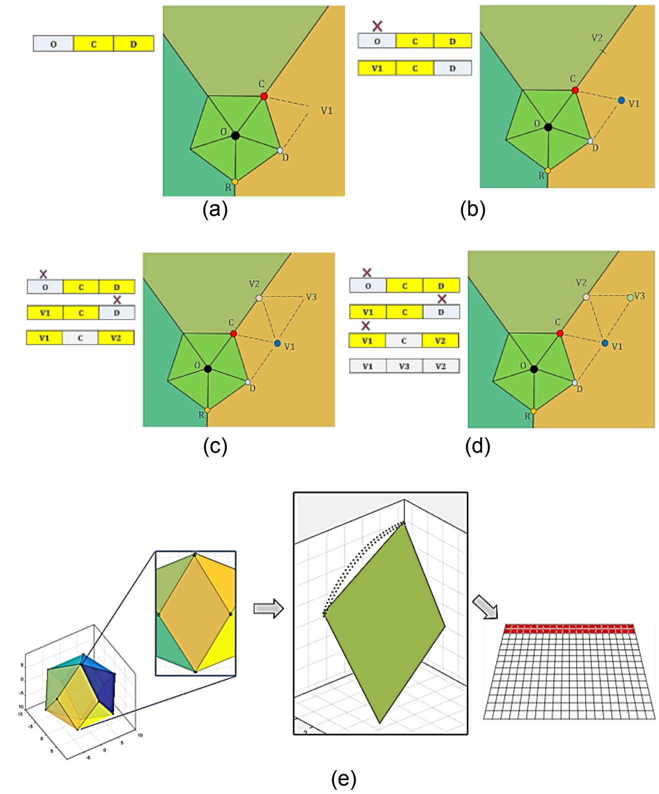


Fig. 12. (a) to (d) shows some steps of initializing first two rows of preliminary diamond. Elements with yellow color represent the vertices which are shared by same triangle. The cross signs indicate that the vertices are marked as *'used'*. (e) shows corresponding array of the diamond.

The first two columns of D_{init} *(0,0)* are established by applying similar iterative approach (see Fig. 13). But in this case it starts with *O, R* and *D*. However, the columns that are mapped with connectivity are D_{init} *(0,0)* to D_{init} *(m,0)* and D_{init} *(0,1)* to D_{init} *(m,1)* where m is the number of rows in D_{init}.

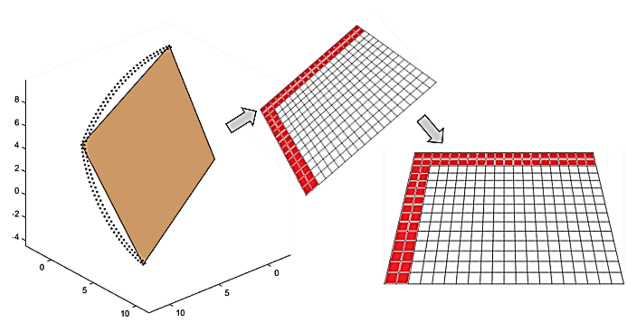
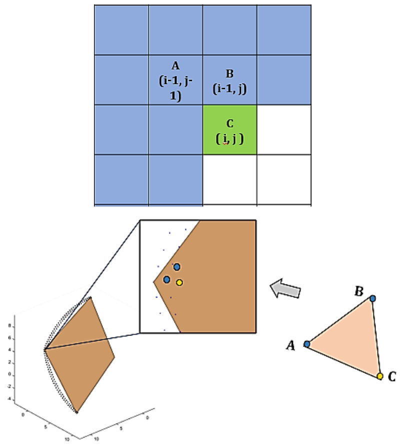


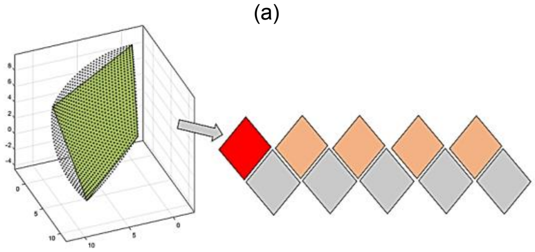
Fig. 13. Initialization of first two columns of the preliminary diamond (*left*). Accomplished two rows and columns in connectivity map is indicated with red color (*middle* and *right*)

After initializtion, it is necessary to complete the entire array (see Fig. 14). In order to find the vertex information for *C* which is in D_{init} *(i,j)*, D_{init} *(i-1, j-1)* and D_{init} *(i-1, j)* can be used, labeled as *A* and *B* . The algorithm will search for a triangle that shares coordinates of two vertices *A* and *B*, but the third vertex has no entry in D_{init} so far.

This is the way to fill each $D_{init}(i,j)$ and it continues mapping until it reaches $D_{init}(m,n)$.



(a)



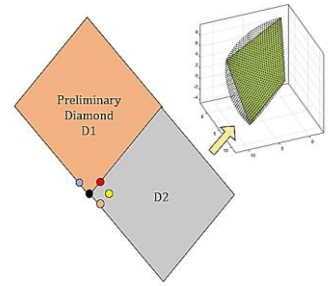
(b)

Fig. 14. Completion of connectivity mapping for preliminary diamond. (a) shows finding the connectivity information of *C* at $D_{init}(i,j)$ using *A* at D_{init} *(i-1, j-1)* and *B* at D_{init} *(i-1, j)*. Blue color indicates the completed initial rows and columns in the connectivity array. Green indicates the current array location that needs to be occupied. Moreover, (b) shows completion of connectivity mapping D_{init} for preliminary diamond with red color.

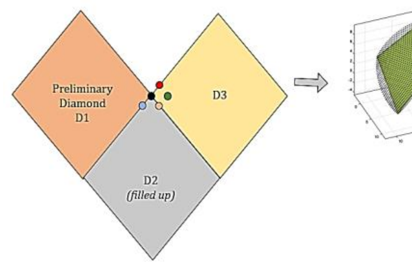
CM for rest of the diamonds: To perform connectivity mapping for other nine diamond above procedure is followed, but is not necessary to select *O, D, R* and *C* for all of these diamonds to initiate iterative method. Fig. 15 illustrates the procedure. From D_{init}, we can pick D_{init} *(m,0)* which will work as *O* for next diamond D_1. The we can select D_{init} *(m,1)* which is assigned as *C*. Now we need another vertex *P* from D_{init} which is at *(m-1,0)* position. So again the algorithm will search for a triangle that shares *O* and *C*, but not *P*. The other vertex of that triangle will be treated as *D*. From *O, C* and *D*, we can simply get coordinates of *R*. Using *O, D, C* and *R*, the previous methods can be applied for D_1. This approach works for every odd diamons. For even diamonds D_k, we will look at D_{k-1} and assign *O* as $D_{k-1}(1,n)$, *R* as $D_{k-1}(2,n)$ and *P* as $D_{k-1}(0,n-1)$. Using this connectivity, *D* and *C* can be easily found and the array for D_k is completed.

3 Results

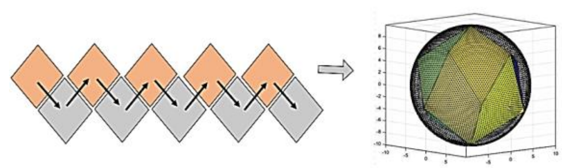
An user interface is developed to visualize data of ICON on MATLAB. The interface is called *visICON* (*vis*ualizing *ICON*) and takes the connectivity map from the pipeline described above and visualize data using those geographic coordinates. In order to visualize, the data stored in center of the triangles are used in this paper. In addition, downsampling is performed on the arrays to present an application of implementing ACM on ICON. To utilize the advantage of index operation of MATLAB, the indices of the vertices from ICON is also stored in an additional array along with their conectivity information. Fig. 16 – 20 shows some screeshots and outputs using the *visICON* interface.



(a)

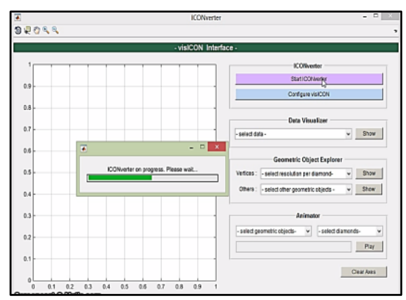


(b)

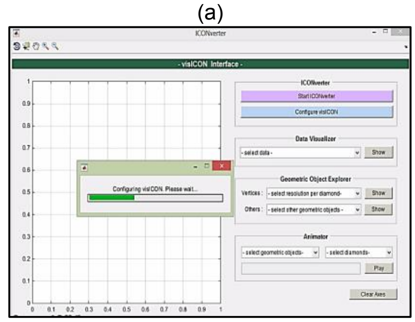


(c)

Fig. 15. (a) connectivity mapping procedure for odd diamonds. *O, D, C* and *R* is marked as black, yellow, red and orange circles. (b) shows CM for even diamonds. Here *O, D, C* and *R* is marked as black, green, red and orange. (c) presents the processing of all the diamonds.

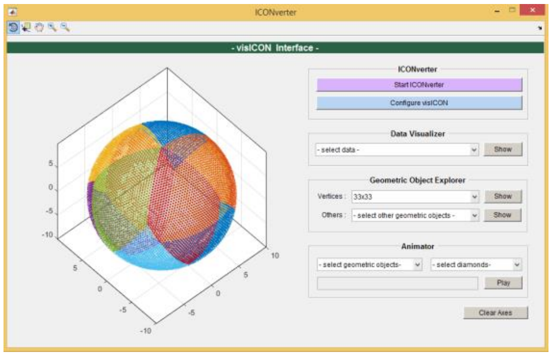


(a)

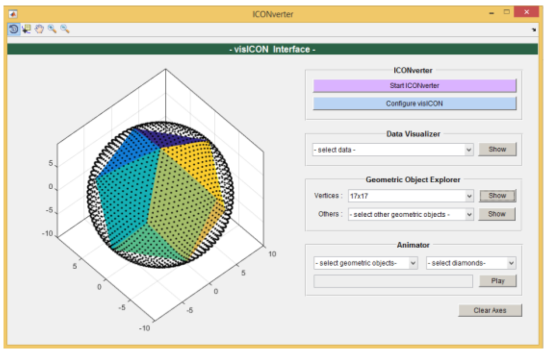


(b)

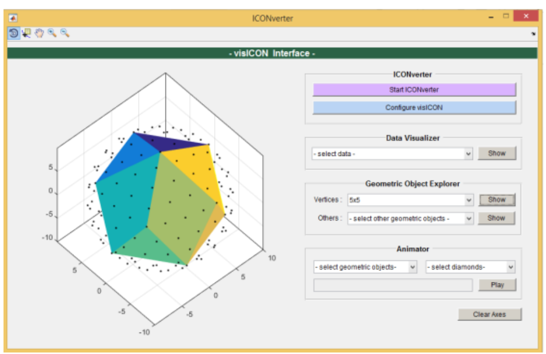
Fig. 16. (a) screenshot of *visICON* while the connectivity mapping is on progress. The process of converting the 3D coordinates to 2D array for ICON is labeled as *ICONverter* (*ICON*+*con*verter) in the GUI. (b) screeshot while MATLAB index operation is being configured



(a)



(b)



(c)

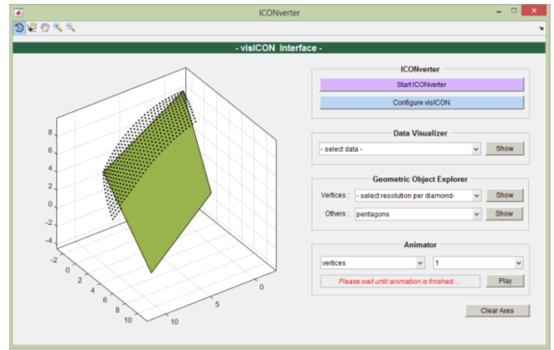
Fig. 17. Shows visualization of vertices of ICON grid with 33 × 33 vertices per diamond in (a), 17 × 17 per diamond in (b) and 5 × 5 per diamond in (c)

4 Future Work

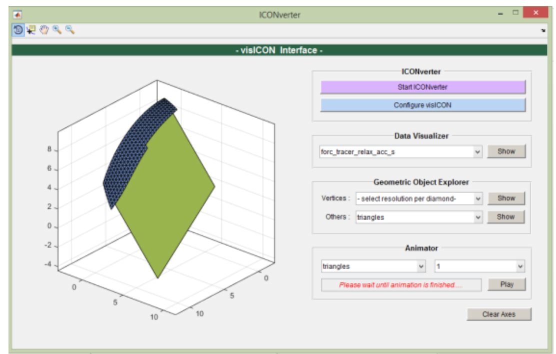
Implementation of Atlas of Connectivity Maps for ICON grid provides a flexible data structure on which several operations can be performed. The future plan is to extend this work to multi-resolution, hierarchical traversal between different level, neighborhood traversal etc.

5 Conclusion

In this paper, the vertex information of ICON grid is mapped totwo dimensional array using Atlas of Connectivity Mapping technique. Visualization and results exhibits that the connectivity information is preserved in 2D domain and the information remain unchanged. The output of the procedure presented in this paper can be used furter for more deeper research on ICON dataset.

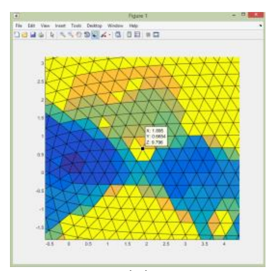


(a)

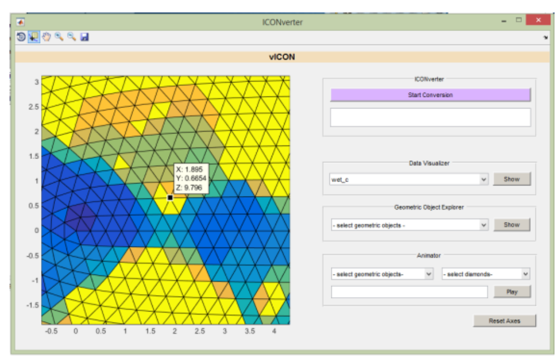


(b)

Fig. 18. The *visICON*'s option for visualizing the animation of generating vertices (a) and triangles (b) while scanning through an array corresponding to a diamond.



(a)



(b)

Fig. 19. shows preserving of geographic coordinates aftter applying ACM. (a) shows a vertex befor applying ACM, while (b) shows the same vertex after applying. Coordinates remain unchanged after connectivity mapping

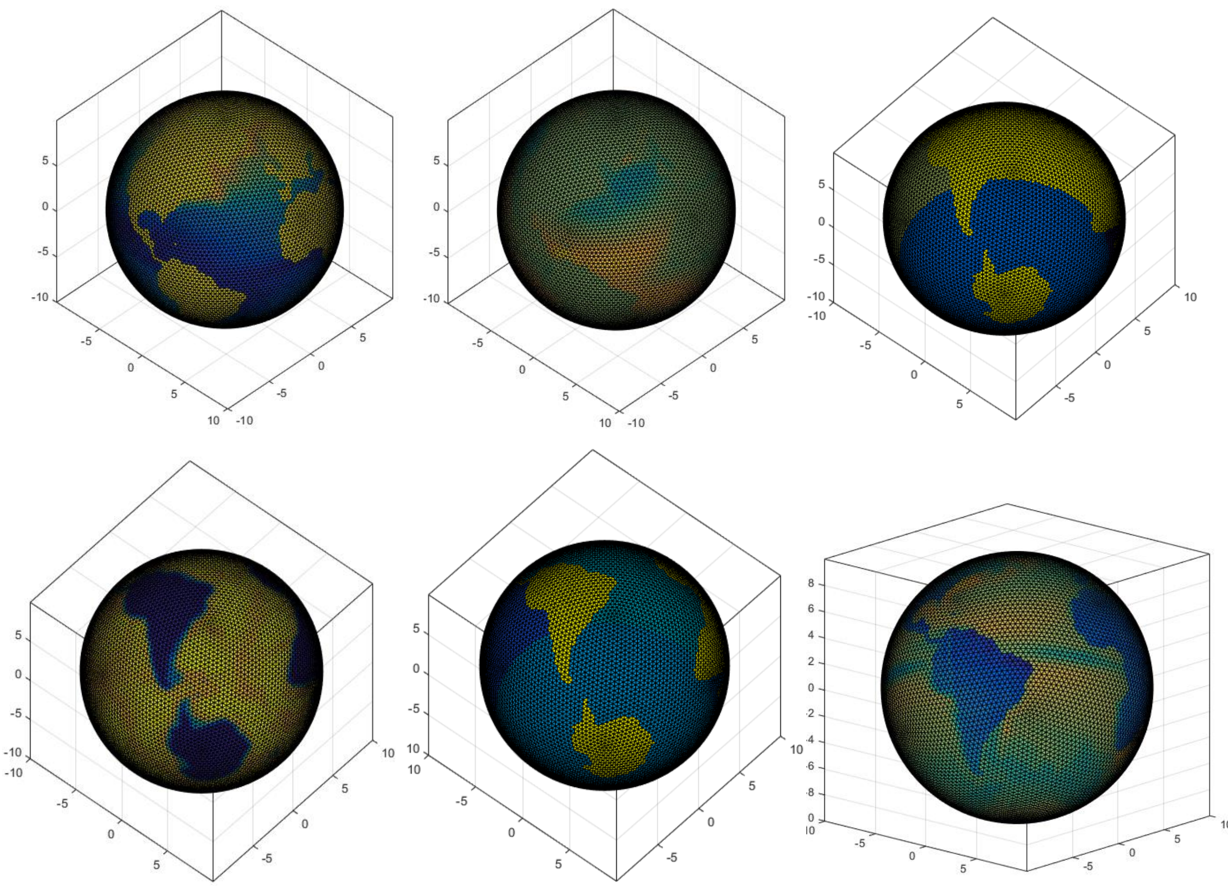


Fig. 20. Visualization of different types of data from ICON dataset after applying Atlas of Connectivity Maps

ACKNOWLEDGMENTS

I wish to thank Dr. Usman Raza Alim for his continuous supervision on this work, and to Ali Mahdavi-Amiri for his guidance and support.

REFERENCES

[1] Jubair, Mohammad Imrul., Project Proposal Report, CPSC 601, Fall 2014

[2] Mahdavi-Amiri, Ali., Harrison, Erika., and Samavati, Faramarz. Hexagonal Connectivity Maps for Digital Earth. International Journal of Digital Earth 2014

[3] Mahdavi-Amiri, Ali., Samavati, Faramarz. ACM: Atlas of Connectivity Maps for Semiregular Models. Proceedings of Graphics Interface 2013