Geodesic Self-Organizing Map

Yingxin Wu[a] and Masahiro Takatsuka[b*]

[a] The School of Information Technologies, The University of Sydney, Australia;
[b] ViSLAB, The University of Sydney, Australia
NICTA[#]

ABSTRACT

Self-Organizing map (SOM) is a widely used tool to find clustering and also to visualize high dimensional data. Several spherical SOMs have been proposed to create a more accurate representation of the data by removing the "border effect". In this paper, we compare several spherical lattices for the purpose of implementation of a SOM. We then introduce a 2D rectangular grid data structure for representing the geodesic dome. This new approach improves the neighborhood searching process in the spherical gird. The new Geodesic SOM and its data structure are tested using socio-demographic data. In the experiments, we try to create a notion of direction in the Geodesic SOM. The direction facilitates more consistent visual comparison of different datasets as well as to assist viewers building their mental maps.

Keywords: Self-Organizing Maps, sphere tessellation, geodesic dome, direction

1. INTRODUCTION

1.1 Data Visualization

Recent rapid advances in modern computing and graphics technologies[1] suggest that various scientific researches dealing with massive and complex datasets can be enhanced by the increase use of data visualization. An excellent visualization technique could significantly improve the process of Exploratory Data Analysis by engaging human experts in much deeper data analysis through visual inspection. Moreover, there are now great demands for carrying out such tasks in real time in order to test many hypotheses with different parameters, allowing analysts to quickly gain insights into complex models.[2]

1.2 Self-Organizing Map

Self-Organizing Map[3] is one of the Artificial Neural Networks and has been greatly appreciated in analyzing and visualizing high-dimensional data. The applications of the SOM can be found not only in the fields of engineering but also in other areas such as the medical, agricultural and social science fields[1,4]. This neural network tries to map a high-dimensional real number space onto a low-dimensional (typically 2D or 3D) integer space. It exhibits the following characteristics during its non-linear mapping process: 1) multidimensional scaling, 2) topological mapping and 3) unsupervised clustering.

In SOM, the neighborhood relationship, which is used to describe the topological information, is usually defined by a 2D rectangular or hexagonal lattice such that every grid unit has 4 or 6 neighbors (Figure 1). Each grid unit is used as a neuron and is associated with a weight vector. During the training phase, all neurons compete with each other for the input signals. The winner and its neighbors update their connection weights. At the end of the training, all samples are equally represented by neurons, and the neighboring units in the grid tend to model similar regions of the data space. This topological mapping is achieved due to the neighborhood relationships. However, the grid units at the boundary of

* Further author information: (Send correspondence to Yingxin Wu.)
Yingxin Wu: E-mail: chwu@it.usyd.edu.au, Telephone: +61-2-83745583
Masahiro Takatsuka.: E-mail: masa@vislab.net, Telephone: +61-2-9351-5903

National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council

the SOM have fewer neighbors than the units inside the map, which often leads to the notorious "border effect"—the weight vectors of these units "collapse to the center of the input space".[5]

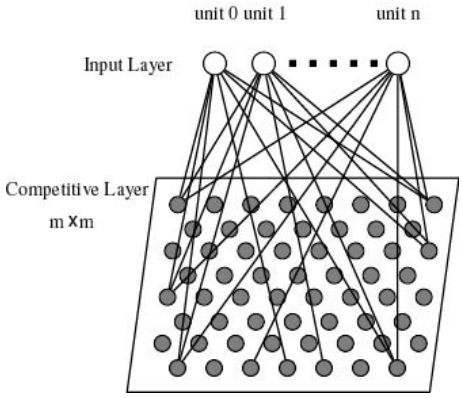

Figure 1. A hexagonal array of neurons in the SOM**.**

2. REVIEW OF PREVIOUS WORKS

Several approaches have been suggested to eliminate the "border effect", such as the heuristic weighting rule method by Kohonen[3] and the local-linear smoothing by Wand and Jones[6]. Besides the mathematical solutions, we can also implement the SOM on a spherical lattice so that all the boundaries are eliminated. Ritter[7] suggested the use of tessellated platonic polyhedra as the underlying lattices. He also pointed out that the spherical SOM would be very suitable for data with underlying directional structures. Since then, some SOMs based on different spherical grids have been implemented and applied to different types of dataset

2.1 Polar Coordinated SOM

Oyabu[8] developed a spherical SOM based on the latitude/longitude grid system which is commonly used in representations of the Earth's geometry. Their RGB color spherical SOM maps the black and white colors to the two poles and all the other colors in between (Figure 2). This SOM is able to achieve the multidimensional scaling by mapping a RGB color (3D volumetric) space onto a continuous spherical surface. However, the surface areas associated with each neuron varies significantly. In SOM, it's desirable to have all neurons receive equal geometrical treatment. In other words, every neuron should occupy an equal surface area.

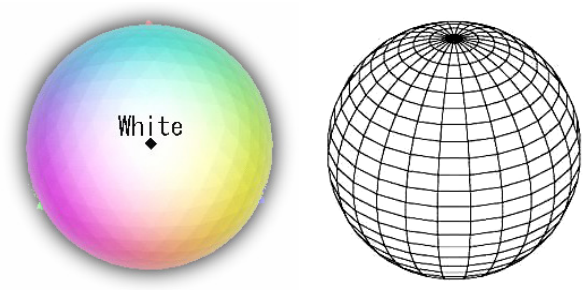

Figure 2. The spherical RGB SOM by Oyabu[8]

2.2 SOM using the tessellated Platonic Polyhedra

Sangole and Knopf [9] implemented another spherical SOM based on Ritter's idea. In their work, colors were used to reflect the magnitude of the weight vectors, and the shape deformation of the sphere was introduced to visualize the weight vectors' variance (see Figure 3). However, we would like to point out that this work considered the grid units on the tessellated platonic polyhedra to be uniformly distributed on the surface of the sphere. A uniform distribution of

points means that the distances between every point and its neighbors are identical. It is a fact that such uniformity can not be achieved on the sphere except in the cases of the five platonic polyhedra (tetrahedron, cube, octahedron, icosahedron and dodecahedron).

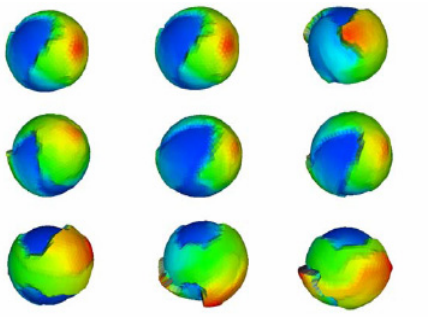

Figure 3. Sangole and Knopf 's spherical SOM[9]

In the next section, we will discuss in detail how we select the grid lattice for our Geodesic SOM. The main difficulty in implementing the GeoSOM is the choice of a data structure which facilitates neighborhood searching. Updating the winning neuron's neighborhood is a computationally intensive task during the SOM's training process. Then in section 4, we will introduce a neighborhood searching algorithm based on the proposed data structure. In section 5, we go further to implement a directional Geodesic SOM.

3. SPHERICAL LATTICE FOR THE GEODESIC SOM

For the 2D SOM lattice, the hexagonal grid is considered to be preferable over the rectangular one because every grid unit has the same number of direct neighbors, and the distances between a unit and its six direct neighbors are the same. We will use these two criteria in choosing the spherical lattice for our SOM. However, as mentioned before, there are only five platonic polyhedra that can satisfy both of the criteria[10]. Note that this would not hold if they were further tessellated to create more grid units. So what we can do is to choose a spherical lattice which has the least variance in the number of neighbors and has approximately the same distance between direct neighbors. In order to search for the most appropriate spherical lattice, we have examined results from various mathematical and cartographic researches. We now make comparisons of several spherical lattices.

Uniformly distributing large number of points on the sphere has been a problem of great interest to the mathematical researchers because it has important applications in the field of computation. But their definition of uniformity is usually to minimize the external energies[11] so that their solutions are not suitable to our purpose. Firstly, whether the generated points have same number of neighbors or not is not taken into consideration under these mathematical solutions. Secondly, the nearest neighbors' distances vary greatly when only a few hundreds of points are needed. For example, Rakhmanov[12] introduced a simple algorithm to distribute an arbitrarily number of points on the sphere in an asymptotically uniform manner. The spherical coordinate (θ_i, φ_i) of each point is calculated by:

$$\theta_i = \arccos(h_i)\,,\ \varphi_i = (\varphi_{i-1} + \frac{3.6}{\sqrt{N}}\frac{1}{\sqrt{1-h_i^2}})(\mathrm{mod}\,2\pi)$$

$$\text{where } h_i = -1 + \frac{2(i-1)}{(N-1)},\ 1 \le i \le N \text{ and N is the number of points}$$

We used these formulas to generate 162 points on the sphere and calculated the distance between each vertex and its nearest neighbors. The distances vary from 0.15779 to 0.30069 with the biggest nearest neighbor distance on the equator and the smallest distance at the two poles.

The latitude/longitude grid system used by Oyaby[8] has the advantage that each grid unit can be indexed by spherical coordinates. The positions of the grid units can thus be easily stored in a 2D array which makes the searching for neighbors as easy as in the normal rectangular grid. But the distances between a grid unit and its direct neighbors vary

greatly from the equator to the two poles. This would greatly exaggerate the distances between the weight vectors along the equator and compress them at the two poles.

The grid systems obtained from tessellating the five platonic polyhedra are also commonly used in cartography. They are considered to be superior to the latitude/longitude grid system because the shapes of the grid cells are much more regular[13]. In fact, they are also very suitable to become the underlying lattice of the spherical SOM. Firstly, they can be tessellated up to any frequency to create the desire number of grid units; secondly, most of the vertices have the same number of neighbors and the edge lengths are similar to each other. The tessellation process introduced by Fuller[14] is shown in Figure 4. Each triangular face of the polyhedron is subdivided into a series of smaller triangles by lines running parallel to the original edges of the triangles[10]. For example, to create a two frequency triangle face, firstly calculate the mid point of each edge. Then connect the 3 mid points using straight lines which subdivide the triangle into 4 smaller triangles. This method is only applicable to the faces which are triangles. If we want to tessellate the cube and the dodecahedron, we need to triangulate their faces first. The spherical polyhedron obtained from this method is called the geodesic dome.

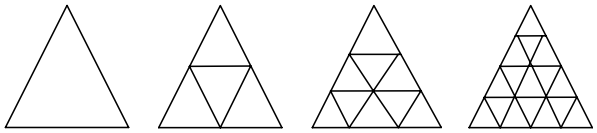

Figure 4. From left to right: one-frequency, two frequency, three frequency and four frequency tessellation

The lattices obtained from tessellating the five platonic polyhedra are quite different from each other. Among the five, the icosahedron is most similar to the sphere, so after tessellation, the edges always have the smallest variance in length compared to the others (see Table 1). Based on the above observation, we decided to choose the geodesic dome based on the icosahedron as our spherical lattice to implement the SOM.

Table 1. Edge lengths of the geodesic domes based on different platonic polyhedra[10].The radius of the circumscribing sphere is 1.

		Tetrahedron	Octahedron	Cube	Dodecahedron	Icosahedron
1 frequency	No. of different edge lengths	1	1	2 (after triangulation)	2 (after triangulation)	1
	Edge length variation	1.63299	1.14121	0.91940—1.15736	0.71364—0.64085	1.05146
2 frequency	No. of different edge lengths	2	2	5	5	2
	Edge length variation	0.91940—1.41421	0.76537—1.00000	0.47313—0.65012	0.32473—0.37668	0.54653—0.61803
3 frequency	No. of different edge lengths	3	3	9	10	3
	Edge length variation	0.50914—0.97784	0.45951—0.67142	0.38164—0.45883	0.21349—0.25260	0.34862—0.41241
4 frequency	No. of different edge lengths	5	6	14	14	6
	Edge length variation	0.33742—0.999998	0.32037—0.57735	0.22393—0.35141	0.15847—0.19199	0.25319—0.31287

4 DATA STRUCTURES AND NEIGHBORHOOD SEARCHING

Finding the neighbors within a certain distance to a neuron is a very important step in the training of a SOM. In the traditional SOM algorithm, the training will go for several thousand epochs in order for the weight vectors to converge and correctly represent the data. In each epoch, the input data points are fed into the network one by one, and the neuron

with the weight vector closest to a specific input vector is updated together with its neighbors. Using the rectangular or hexagonal lattice, grid units can be stored in a 2D array. This means that the neighbors of the winning neuron can be found using simple addition/subtraction arithmetic, see Figure 5.

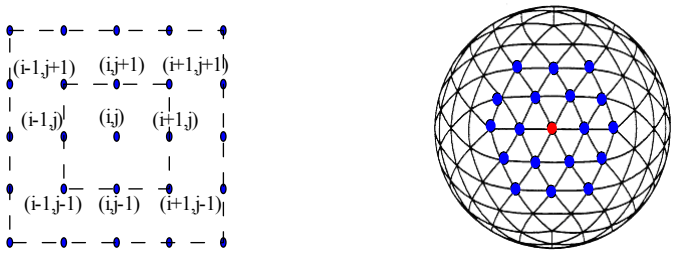

Figure 5. Left: the neighborhood of a winning neuron on the rectangular lattice; Right: the neighborhood of the winning neuron on the 4 frequency Icosahedron

Storing the lattice of the geodesic dome is more complex compared to the 2D grid. The data structure should support fast vertex indexing otherwise the neighborhood searching would become the bottleneck in the SOM training process. The basic idea is to treat the spherical lattice as a 3 dimensional graph. The distance between two vertices is defined to be the length of the shortest path. Given a specific vertex, we first find out its directly connected neighbors. Then go on to search the further away neighbors based on the direct neighbors and so on, up to a specified distance parameter (the update radius). There are two common methods to store a graph: one is to calculate the adjacency matrix of all the vertices. The alternative is to store all the vertices into an array with each array element containing pointers to its direct neighbors. The first method takes $O(n^2)$ space, where n is the number of grid units. Therefore this method is not suitable for SOMs with a large number of neurons. The second method is clumsy and error prone because we need to carefully maintain a large number of pointers. The main drawback of these two methods is that if we need to further tessellate the geodesic dome to create more neurons, the adjacency matrix or all the direct neighbors' pointers need to be updated. We introduce a data structure which can address these problems and uses only O(n) space.

4.1 Data structure for an icosahedron based geodesic dome

The lattice of the geodesic dome can be unwrapped into a two-dimensional map, as shown in Figure 6. We created two axes U and V in the map such that each vertex in the map can be uniquely denoted by the two-dimensional coordinate pair (u, v). By rotating all the vertices in the map such that the two axes U, V are orthogonal to each other, we can see that all the vertices of an icosahedron could be stored in a two-dimensional matrix (Figure 7). In our program, vertices sharing the same u coordinate are stored in a one dimensional array, called a v_array, in order of ascending v coordinate. Every v_array is contained within another one dimensional array, called the u_array, in order of ascending u coordinate

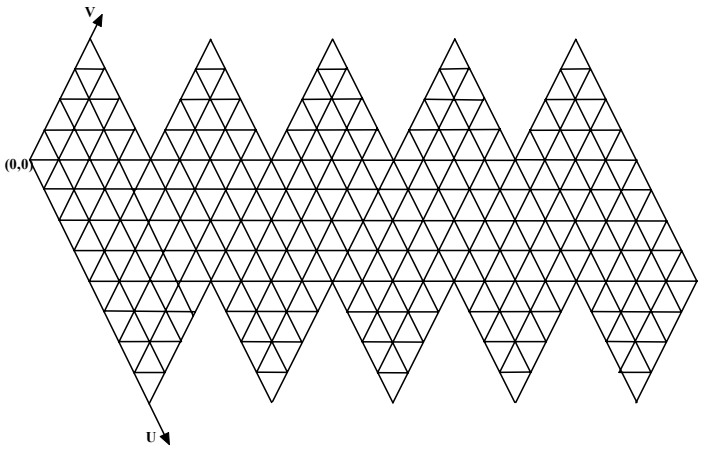

Figure 6. Unwrapped 4-frequency Icosahedron

Note that all the boundaries of the 2D map are joined on the sphere. Therefore there would be some duplicated points in the map. For example, in Figure 7, point A and point C are two poles on the geodesic dome. They each have four duplicated points (A' to A'''', C' to C''''). B and B' as well as D and D' are also duplicated. Our data structure maintains a list of the duplicated points. Note that only the points at the boundary may be duplicated, therefore the number of duplicated points will be very small compared to the total number of grid units. Using this data structure, only O(n) space is needed to store the geodesic dome, where n is the number of vertices on the dome

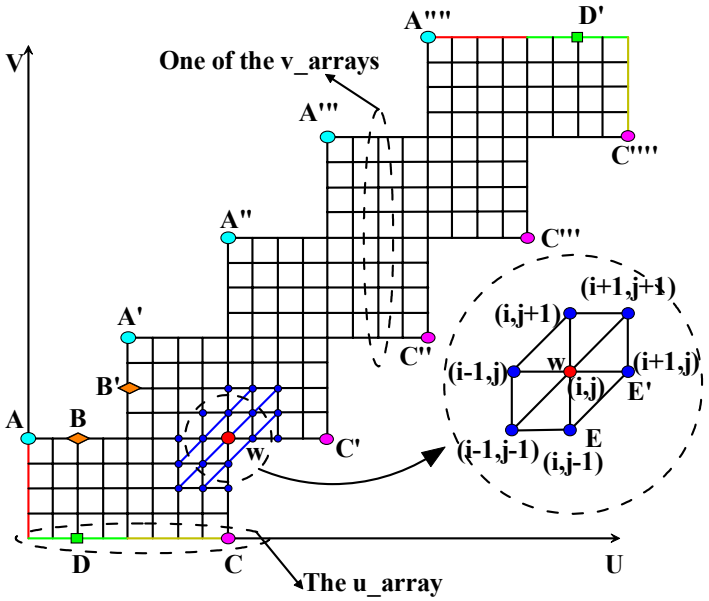

Figure 7. The winning neuron and its neighbors in the data structure

4.2 Searching for neighbors in the data structure

The neighbors of a grid unit are grouped into levels. For the point W in Figure 7, its first level neighbors are the vertices that connect to it by only one edge in the geodesic dome. The second level neighbors are the vertices that connect to W by 2 edges and so on. In our current algorithm, in order to get the n^{th} level neighbors, we need to search from the 1^{st} level neighbors until the target level is reached. By the first thought, this method would be slow in finding the n^{th} level neighbors. However, in the training of SOM, all the neighbors from the 1^{st} level to the target level of the winning neuron need to be updated. Therefore using this algorithm, searching for neighbors can be done efficiently.

Each point in the 2D geodesic dome map has 6 first level neighbors. The first level neighbors of W(i, j) can be calculated as: (i, j+1), (i+1,j+1), (i+1, j), (i-1,j-1), (i-1,j), (i, j-1). As mentioned above, there are duplicated points in the map's boundaries and they have the same weight vector. A vertex's duplicate point should not be put into the neighbor list or the weight vector they represent will be updated several times. For example, in Figure 7, point E and E' are the same, so only one of them will be included. In the case when the winning neuron has duplicated points, such as point A in Figure 7, all the neighbors of its duplicated points (A' to A'''') need to be searched

4.3 Increasing the frequency of the Geodesic Dome

The number of neurons needed in the SOM depends on the dataset size. The Icosahedron only has 12 vertices. In order to increase the amount of neurons, the dome can be tessellated into a different frequency. The vertices number of the icosahedron-based geodesic dome can be calculated by: $|V| = 10 \times f^2 + 2$, where f is the frequency number[15]. Figure 8 gives an illustration of how to obtain a 3 frequency geodesic dome from an icosahedron using the above data structure. Dome of other frequencies can be calculated under the same scheme.

Suppose the icosahedron's circumscribing sphere is a unit sphere with its center at the origin. ABD and BCD are two adjacent spherical triangles on the sphere. In order to tessellate the two triangles into 3 frequency, firstly break each

edge of these two triangles into 3 equal length segments. This is equivalent to inserting 8 new vertices E, F, I, J, G, H, K and L between the old spherical arcs. Given the coordinates of A, B, C and D, it is not difficult to calculate the coordinates of the newly generated vertices geometrically. Then we can go on to calculate and insert vertices O and P which are the mid points of the arc FJ and KG respectively. All the other spherical triangles in the geodesic dome can be tessellated using this method.

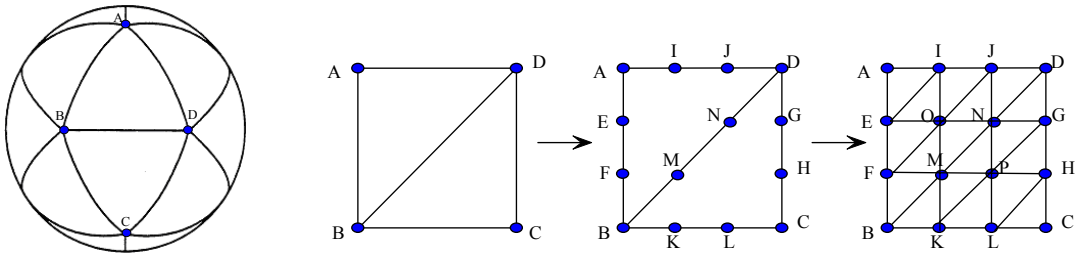

Figure 8. Two spherical triangles and the tessellation process.

Note that based on our data structure, we only need to calculate the new vertices when increasing the frequency. No additional adjacency matrix or neighbor pointers need to be updated. Hence, it is possible to apply the principal of the Growing SOM[19]. Some new duplicated points will be created along the boundaries of the map. Therefore we need to scan along the map boundaries to find all the duplicated points.

5 EXPERIMENTS

We carried out several experiments to test our proposed data structure using datasets from different domains. In each experiment, the average distances between every weight vector and its direct neighbors are calculated and coded with colors. The colors are linearly changing from blue, cyan, yellow to orange. Blue denotes the smallest variance between the weight vectors, and orange the largest.

We also introduce a direction in the Geodesic SOM by fixing two extreme data points to the North and South Pole. We have not yet fully investigated the accuracies of the direction and how to choose the two best points to be fixed. But as we will see in the following experiments, the direction helps us to compare SOM trained with different datasets, which would be difficult using normal SOMs. Moreover, we believe that the introduction of the directional information to our Geodesic SOM will help viewer building their mental maps.

5.1 The Poverty and Inequlity Data Set

This dataset [16, 17] contains the statistical data of 91 countries in 1990. Each country is described by its birth rate, death rate, infant death rate, men/women life expectancies and Gross National Production, which means each sample has 6 dimensions. In the experiment, the data are normalized before training so that each dimension varies from 0 to 1. A lattice of the six frequency geodesic dome is used in the experiment. It contains 362 grid units. In order to create a direction on the map, we first calculate the two data points which have the biggest Euclidean distance — Japan and Malawi. Then during the training, they are fixed on the North Pole and South Pole respectively. The spherical self-organizing map is trained for 500 epochs. In the first 20 epochs, only two data samples representing Japan and Malawi are fed into the SOM to create an initial orientation. After the training, neurons are labeled according to the data mapped on to them. If several data points are mapped on to the same neuron, the one which is closest to the neuron's weight vector is chosen in labeling.

From Figure 9 and Table 2, we can see that all the rich countries are mapped onto the north hemisphere with the poorest countries at the south hemisphere. Rich countries tend to have lower birth rates, lower infant death rate and longer life spans, which indicating better living environment; on the other hand, people in the poor countries such as Afghanistan live a hard life for they have higher birth rate and death rates. The people's life spans are also much shorter. The experiment result shows that because of the chosen fixed points, the order of countries in the SOM form a direction pointing from the rich countries to the poor countries.

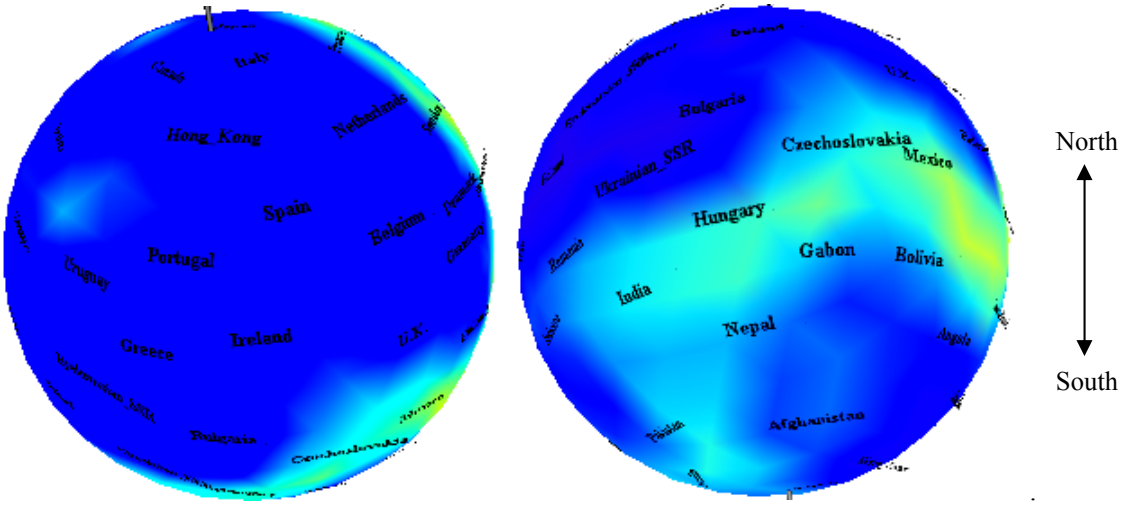

Figure 9. The Geodesic SOM trained with the poverty and inequality dataset. Japan and Malawi are fixed at the North Pole and South Pole. Training parameters: 362 neurons, total Iteration=500, initial learning rate α=0.5, initial update radius r=10 level, $\alpha(t) = 1 - t / totalIteration$, $r(t) = \exp(-40 \times t / totalIteration)$, t is the iteration number.

Table 2. Some sample data from the poverty and inequality dataset[18]

Country	Birth Rate (%)	Death Rate (out of 1000)	Infant Death Rate (out of 1000)	Males Life Span	Women Life Span	Gross National product per capita (USD)
Italy	9.7	9.1	8.8	72	78.6	16830
Hong Kong	11.7	4.9	6.1	74.3	80.1	14210
Spain	10.7	8.2	8.1	72.5	78.6	16100
Portugal	11.9	9.5	13.1	66.5	72.4	7600
Hungary	11.6	13.4	14.8	65.4	73.8	2780
Gabon	39.4	16.8	103	49.9	53	390
Nepal	39.6	14.8	128	50.9	48.1	170
Afghanistan	40.4	18.7	181.6	41	42	168

5.2 The Illiteracy Data Set

Two datasets were used in this experiment. They contain the illiteracy information of 125 countries in year 1990 and year 2000 respectively[18]. Hungary is the most literate country and Niger is the least literate country in both of the datasets. An eight frequency geodesic dome is used as the spherical lattice. Again we fix the most literate and illiterate country to the two poles and label the neurons according to the data mapped on to it. The Geodesic SOM is trained for 800 epochs. Same as the former experiment, in the first 20 epochs, only two data representing Hungary and Niger are fed into the SOM to form an initial orientation.

From Figure 10, we can see that fixing two extreme data points at the two poles creates a sense of direction in the Geodesic SOM. The countries are approximately ordered from North Pole to South Pole according to their literacy standards. This provides a mechanism to compare 1990 dataset and the 2000 dataset. Take three countries — Lao, Uganda and Omen — for example. Lao's illiteracy rate was the lowest among the three countries in 1990. However, in

2000, Oman reduced its illiteracy rates and became the most literate country among the three. Lao also improved its education standards but obviously left behind the other countries. However, we should point out that the direction is roughly. If some data are too similar to each other, their order in the map may not strictly follow the direction.

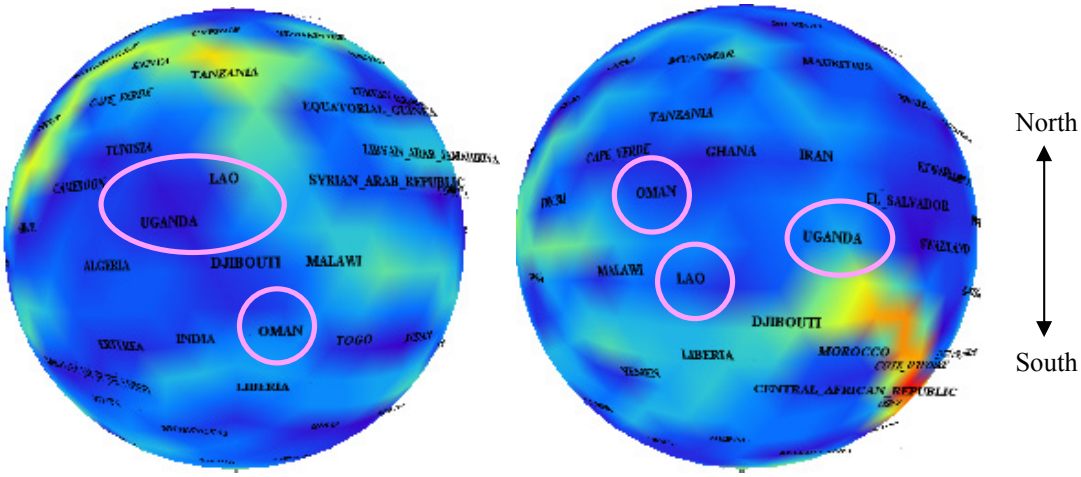

Figure 10. Left: the Geodesic SOM trained with the year 1990 dataset. Right: the Geodesic SOM trained with the year 2000 dataset. Hungary and Niger are fixed onto the two poles. Parameters: 642 neurons, total Iteration=800, initial learning rate α=0.6, initial update radius r=16 level, $\alpha(t) = 1 - t / totalIteration$, $r(t) = \exp(-30 \times t / totalIteration)$, t is the iteration number.

Table 3. Illiteracy Data for the three countries in 1990

Country	Total Illiteracy rate. 15 years and older (%)	Males Illiteracy rate. 15 years and older (%)	Females Illiteracy rate. 15 years and older (%)
Lao	43	30	57
Uganda	44	31	57
Oman	45	33	62

Table 4. Illiteracy Data for the three countries in 2000

Country	Total Illiteracy rate. 15 years and older (%)	Males Illiteracy rate. 15 years and older (%)	Females Illiteracy rate. 15 years and older (%)
Lao	35	24	37
Uganda	33	23	43
Oman	28	20	38

6 CONCLUSION AND FUTURE WORK

The spherical SOMs have attracted many researcher's interests because they can remove the notorious "border effect" of the normal 2D SOM without using complex mathematical tools. However, it is impossible to uniformly distribute an arbitrarily large number of points onto the sphere. Therefore the distances between direct neighbors in the spherical grid are not as uniform as the 2D rectangular/ hexagonal grid. We examined several possible spherical lattices based on mathematical and cartographic studies. The grid system of the geodesic dome based on Icosahedron is considered to be the best among the other spherical lattices. We implemented our Geodesic SOM using a 2D rectangular grid data structure. This data structure is space efficient and facilitates the fast searching of neighbors on the spherical lattice.

In the experiments, we visualized two different socio-demographic datasets. Two extreme points were mapped on to the poles to impose the directional information in the Geodesic SOM. We believe that this directional information would help a viewer to build his/her mental map. Furthermore, the directional information provides reference points, which enable users to compare different results. However, choosing the two fixed data still relies on the domain knowledge. We hope to investigate the statistical based methods in selecting the reference points to be fixed.

REFERENCES

1. Brown, J. R. et al., *Visualization: Using Computer Graphics to Explore Data and Present Information*, John Wiley & Sons, Inc. NY，1995
2. Schwan, K. et al., , *Integrating Program Steering, Visualization, and Analysis in Parallel Spectral Models of Atmospheric Transport*, *Science Information Systems Newsletter*, issue 36, 1995
3. T. Kohonen, *Self-Organizing Maps,* Third Edition*,* Springer, 2001.
4. Tokutaka, H., et al., *Application of Self-Organizing Map (SOM) to Chemical Data Analysis*, Journal of Surface Analysis, vol. 5, no. 1, pp. 102--105
5. Scarle, W.S., *SOM FAQ,* ftp://ftp.sas.com/pub/neural/FAQ.html, 2002
6. Wand, M.P., and Jones, M.C., *Kernel Smoothing,* London: Chapman & Hall, 1995
7. Ritter H., *Self-Organizing Maps on Non-Euclidean Spaces*, In: Oja E, Kaski S, editors. Kohonen maps. Amsterdam. The Netherlands: Elesevier BV; 1999. p.97-109
8. Oyabu, Fukuoka and Nakatsuka, *Clustering using a Self-Organizing Map*, http://www2.kanazawa-it.ac.jp/oyabu/som/somint.htm
9. Sangole, A., Knopf, G.K., *Visualization of Randomly Ordered Numeric Data Sets Using Spherical Self-Organizing Feature Maps*, Computer & Graphics 27, p963-976, 2003
10. Pugh, A., *Polyhedra—a Visual Approach,* University of California Press, Ltd, 1976
11. E. B. Saff and A. B. J. Kuijlaars, *Distributing Many Points on a Sphere,* The Mathematical Intelligencer, Springer-Verlag Volume 19, Number 1, 1997
12. Rakhmonv E. A., Saff E. B., and Zhou Y. M., *Minimal discrete energy on the sphere*. Math, Res. Lett. 1, 647-662, 1994
13. Sahr, K., et. al, *Geodesic Discrete Global Grid Systems*, Cartography and Geographic Information Science, Vol. 30, No. 2, pp. 121-134, 2003
14. Fuller, R. B., *Synergetics*. New York, New York: MacMillan. 1975
15. Hoffmann,G., *Sphere Tessellation by Icosahedron Subdivision*, 2002
16. Day, A. (ed.), *The Annual Register 1992*, pp. 234, London: Longmans, 1992
17. *U.N.E.S.C.O. 1990 Demographic Year Book*, New York: United Nations, 1990
18. *Public Reports: Illiteracy rate and Illiterate Population*, The UNESCO Institute for Statistics, http://stats.uis.unesco.org/eng/ReportFolders/Rfview/explorerp.asp
19. Fritzke, B. (1995), *Growing Grid - a self organizing network with constant neighborhood range and adaptation strength*, Neural Processing Letters, Vol.2, No. 5, page 9-13