Spherical Layout implementation using Centroidal Voronoi Tessellations

Martín Larrea[1,2], Dana Urribarri[1,2], Sergio Martig[1], and Silvia Castro[1]

[1]Universidad Nacional del Sur
Departamento de Ciencias e Ingeniería de la Computación
Laboratorio de Investigación y Desarrollo
en Visualización y Computación Gráfica.
Avenida Alem 1253, Bahía Blanca, Buenos Aires, Argentina. CP 8000
[2]Consejo Nacional de Investigaciones Científicas y Técnicas
(CONICET)
Avenida Rivadavia 1917, Ciudad de Buenos Aires,
Argentina. CP C1033AAJ
`{mll,dku,srm,smc}@cs.uns.edu.ar`
`http://vyglab.cs.uns.edu.ar`

Abstract. The 3D tree visualization faces multiple challenges: the election of an appropriate layout, the use of the interactions that make the data exploration easier and a metaphor that helps in the process of information understanding. A good combination of these elements will result in a visualization that effectively conveys the key features of a complex structure or system to a wide range of users and permits the analytical reasoning process. In previous works we presented the Spherical Layout, a technique for 3D tree visualization that provides an excellent base to achieve those key features. The layout was implemented using the Tri-Sphere algorithm, a method that discretized the spheres's surfaces with triangles to achieve a uniform distribution of the nodes. The goal of this work was centered in a new algorithm for the implementation of the Spherical layout; we called it the Spherical Centroidal Voronoi Tessellations (SCVT). In this paper we present a detailed description of this new implementation and a comparison with the TriSphere algorithm.

Key words: Tree Layout. 3D Tree Visualization. Information Visualization. Voronoi Diagrams. Voronoi on Spheres

1 Introduction

Information Visualization is a very young research field, but has grown very fast as a rich and interdisciplinary research field. The last advances in Visualization, and particularly in Information Visualization, also highlight fundamental research issues. Nowadays, it is currently a challenging task for designers to find out the strategies and tools available to visualize a particular type of information. The data characteristics and their organization are essential aspects at the adequate visual representation selection. The creation of adequate visual representations is a big challenge. A visual representation is able to convey relationships among many elements in parallel and provides the user with a tool to explore the data in an effective way. Visual representations are essential aids to human cognitive tasks to the extent that they provide stable and external reference points upon which dynamic activities and thought processes may be calibrated and upon which models and theories can be tested and confirmed. The interaction with visual representations makes many complex and intensive cognitive tasks feasible. Visual representations and interaction techniques must allow the users to see, explore and understand large amounts of information at once and are essential to the analytical reasoning process in order to gain insight in the data. Information Visualization has become a large field and tree visualization has emerged as an important subfield applicable when there is a hierarchical relation among the data elements to be visualized. Tree visualization has many areas of application, and many domains require the manipulation and comprehension of complex hierarchical dataset; to address this field, it is necessary to support interactive representations of large trees. The 3D tree visualization faces multiple challenges: the election of an appropriate layout, the support of the interactions that make the data exploration easier and a metaphor that helps with the information understanding. A good combination of these elements will result in a visualization that effectively conveys the key features of a complex structure or system to a wide range of users and permits the analytical reasoning process.

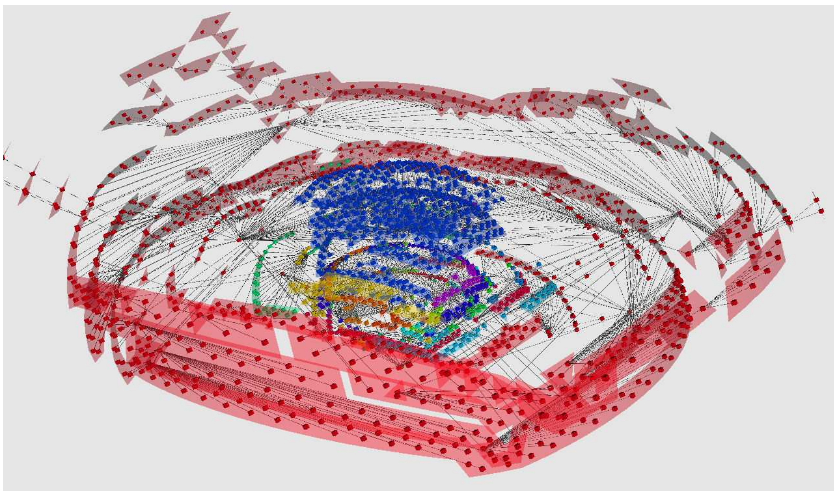
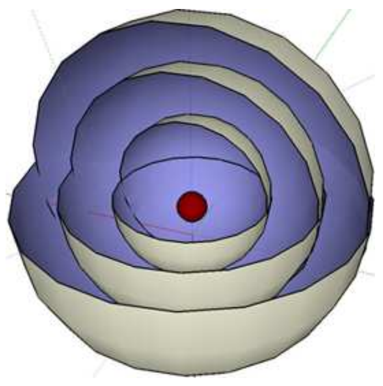


Fig. 1. A file hierarchy visualization using Spherical Layout implemented with the TriSphere algorithm ([1]).

The Spherical Layout ([1], [2]) (Figure 1) is a 3D extension of the 2D radial layout, it was created to allocate a larger number of nodes than the Radial Layout providing an intuitive set of interactions in a 3D environment. Although the Spherical Layout does fit a large number of nodes, its distribution in the space, by the TriSphere algorithm, is not optimal. In this paper, we present a new approach for the distribution of the nodes in the space under the Spherical Layout; by using a Spherical Centroidal Voronoi Tessellations (SCVT) we can distribute any number of nodes on the sphere maximizing its surface usage. This paper is organized as follows: in the next section we present a brief description of the Spherical Layout and its problems. We continue in Section 3 with a description of the Spherical CVT. Afterwards, in Section 4, we compare both algorithms and provide the conclusions. Section 5 ends this paper with the future work.

2 Spherical Layout

We presented the Spherical Layout as an extension of the Radial Layout to three dimensions. Such extension is no straightforward, because the nodes must be placed on a surface instead of an arc and this adds many possibilities. In this section we give a brief presentation of the Spherical layout; as we said at the beginning of this paper; a good association of layout and interactions will result in an effective visualization. For the Spherical Layout we had defined a set of interactions, for details of these interactions and more please refer to [1].


(a) The concentric spheres are the basis of the Spherical Layout.

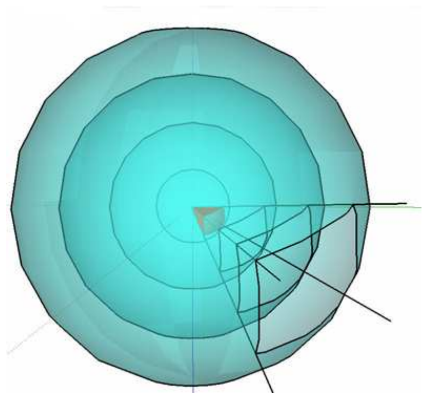

(b) In Spherical Layout the region assigned to a node is defined by a pyramid.

Fig. 2.

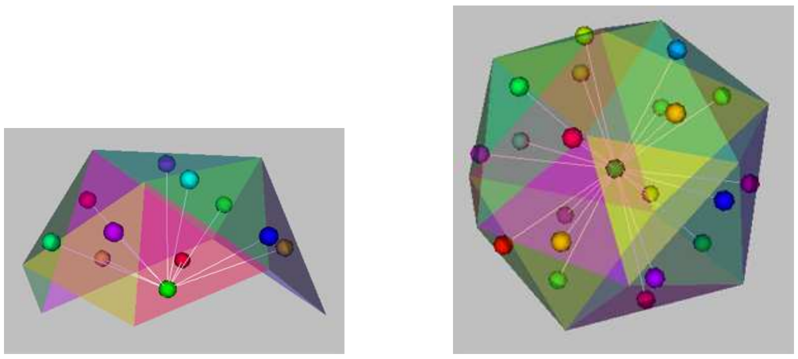
The basis of the Radial Layout are the concentric circles where the nodes are placed; the first step to a 3D generalization is to map these circles into a 3D space. To achieve this goal we consider concentric spheres on which surfaces the nodes are going to be placed (Figure 2(a)). In the Radial Layout each node, except the root, is allocated in a 2D sector within the sector assigned to its parent; in Spherical Layout we replace the 2D plane with a 3D region and the nodes are allocate within the surfaces defined by it (Figure 2(b)). The implementation presented in [1] and [2] is describe as follow: In this implementation the nodes are uniformly distributed on the spheres surfaces; to achieve this goal we first discretized the surfaces of the spheres with triangles and place the nodes in the center of them. We create as many concentric spheres as levels the tree has, all with the same number of triangles. The Spherical Layout discretized uniformly the surfaces of the spheres with triangles; in order to achieve this, we start with an icosahedron, 20 triangles or faces. If the amount of nodes to be allocated is smaller or equal to 20, we place each node in each of the icosahedron's faces. To allocate more than 20 nodes we increased the number of triangles in our sphere. To maintain a uniform distribution each triangle is divided into four triangles; so the number of triangles increases by a factor of 4. Starting from 20 triangles, the next division will result in 80, then 320 and so on. The hierarchical relations of the tree are present here through the projection of the region and subregion from one sphere to the next inner sphere. The execution time of this algorithm is in the order of the number of leaves by the depth of the tree.

2.1 TriSphere Distribution Problem

As we can see in figure 3(a), if the number of nodes is significantly smaller that 20 the result is not the best distribution. To allocate more than 20 nodes we increase the number of triangles in our sphere. If the number of nodes has the form 20×4^i, where i is a positive integer, the resulting distribution will be perfect (Figure 3(b)). However, any number different from this will result in a non optimal distribution. Let us assume we have to allocate 100 nodes; a sphere with 20 triangles is not enough, neither with 80, we must create a sphere with 320 triangles. From those 320 triangles only 100 will be used, which means that 220, more than 65% of the surface, will not be used.

3 Weighted Spherical CVT

Based on the Weighted Centroidal Voronoi Tessellation presented in [3] we developed a Weighted Spherical Centroidal Voronoi Tessellation to make the most of the entire surface of the sphere at the moment of distributing nodes on it. Several works have been done on Spherical Voronoi Tessellation ([9]) and Spherical Centroidal Voronoi Tessellation ([7], [6], [8]), including Constrained Centroidal Voronoi Tessellation, but it seems that nothing on Weighted Centroidal Voronoi Tessellation on the surface of the sphere has been done. In contrast to Constrained Centroidal Voronoi Tessellation, which associates a density function to the surface, Weighted Centroidal Voronoi Tessellation associates values (weights) to the generators: a greater weight means a bigger generated region, independently of the position of the generator on the surface.


(a) Ten nodes distributed on the sphere's surface.

(b) .Twenty nodes distributed on the sphere's surface.

Fig. 3. TriSphere distribution problem. Because we used an icosahedron as the smallest sphere, 20 triangles, half the surface is not used in figure 3(a). In 3(b) because the number of nodes is equal to the number of triangles in the surface we achieve a perfect distribution.

Given a set of points (generators) $P = \{p_1, p_2, \ldots, p_n\}$ in R^m, a Voronoi Tessellation is a set of n regions $V(p_i)$, where a point $q \in R^m$ lies in region $V(p_i)$ if and only if $distance(p_i, q) < distance(p_j, q)$ for each $p_i, p_j \in P, i \neq j$. A Spherical Voronoi Tessellation is a Voronoi Tessellation of the surface of a sphere. In this case the set P is a set of points lying on a surface $\mathcal{S} = \{(x, y, z) \in R^3 : x^2 + y^2 + z^2 = 1\}$, and the regions $V(p_i)$ are the points $q \in \mathcal{S}$ which satisfy $distance(p_i, q) < distance(p_j, q)$ for each $p_i, p_j \in P, i \neq j$. A Weighted Spherical Voronoi Tessellation (WSVT) is a Spherical Voronoi Tessellation where each generator p_i has associated a weight w_i, and the distance between a point q and a generator p_i is the weighted distance $w\text{-}distance(a, w_a, x) = |a - x|^2 - w_a$, where $|.|$ is the euclidean distance.

The general idea of the algorithm used to calculate the WSVT of points P on the sphere is detailed in algorithm 1. The weighted circumcenter of a spherical triangle $\triangle abc$ where w_a, w_b and w_c are the weights of a, b and c respectively, is defined as follows. Let x be the point coplanar to a, b and c which satisfies

$$w\text{-}distance(a, w_a, x) = w\text{-}distance(b, w_b, x) = w\text{-}distance(c, w_c, x).$$

Then, the weighted circumcenter of $\triangle abc$ is $\frac{x}{|x|}$.

A Centroidal Voronoi Tessellation (CVT) [5] is a particular Voronoi Tessellation where the generator of each Voronoi region is the center of mass (centroid) of its own region. A CVT with weighted distance is appropriate to divide a surface into subareas where the size of each one depends on the generator itself and not on the generator's position on the surface. To calculate the Weighted Spherical Centroidal Voronoi Tessellation (WSCVT) it is necessary to introduce the definition of centroid of a spherical triangle and centroid of a spherical polygon. For the next formulae we are considering triangles and polygons on a unitary sphere,

Algorithm 1 Weighted Spherical Voronoi Tessellation (WSVT)

Input: A set of points $P = \{p_1, \ldots, p_n\}$ and a set of weights $W = \{w_1, \ldots, w_n\}$ where w_i is the weight of p_i.

Output: The WSVT $\mathcal{V}$ of P and W on the spherical surface $\mathcal{S}$.

```
1: Let $\mathcal{H}$ be the Convex Hull of $P$ in $R^3$. It represents the Delaunay Triangulation of
   $P$ on $\mathcal{S}$. Note that $\mathcal{H}$ is not weighted.
2: Let $\mathcal{V}$ be the Voronoi Tessellation constructed as the dual graph of $\mathcal{H}$: for each
   triangle in $\mathcal{H}$ its weighted circumcenter is a vertex in $\mathcal{V}$. If two triangles in $\mathcal{H}$ are
   neighbors then their weighted circumcenters are linked by an edge in $\mathcal{V}$.
3: return $\mathcal{V}$.
```

therefore the radius is 1 in all of them. The centroid of a spherical triangle $◬abc$ is $\frac{a+b+c}{|a+b+c|}$. The centroid of a spherical polygon $v_0, \ldots, v_n$ [4] is

$$\frac{\sum_{i=1}^{n-1} \text{area}(◬v_0 v_i v_{i+1}) \text{ centroid}(◬v_0 v_i v_{i+1})}{\sum_{i=1}^{n-1} \text{area}(◬v_0 v_i v_{i+1})},$$

where the area of triangle $◬abc$ is equals to its spherical excess E

$$E = 4 \arctan \sqrt{\tan(\frac{1}{2}s) \tan[\frac{1}{2}(s-A)] \tan[\frac{1}{2}(s-B)] \tan[\frac{1}{2}(s-C)]},$$

being A, B and C the side lenghts, and s the semiperimeter.

Our WSCVT algorithm is based on the one presented in [3] to compute Weighted CVTs in planar surfaces. The general idea of the algorithm is to construct the WSVT of a set of generators and then, replace each generator with the centroid of its corresponding Voronoi region, until a desired error has been reached. To control the size of each region, the weighted distance is not enough, it is necesary to adjust the weight of each generator in every iteration: if after one iteration a region size results bigger than the desired size value, the weight of that generator might be decreased for the next iteration, analougsly, if the region size results smaller, the weight might be increased.

It is important to note that the size values are not areas, but percentages. Making an association between the weights of the generators and the surface of the sphere, the area of a region G_i must represent the same percentage of the entire spherical surface as the weight w_i represents of the overall sum of weights. Then, the desired size value d_i and the actual size value a_i of a region G_i are

$$d_i = \frac{w_i}{\sum w_i} \qquad a_i = \frac{\text{area}(G_i)}{\text{area}(\mathcal{S})}.$$

The algorithm stops when the difference between the desired size value and the actual size value of every region is below a given error ε. The adjusted weight of a generator p_i of current weight w_i, desired size value d_i and actual size value a_i is

$$w_i(1 + \frac{a_i - d_i}{d_i}) \text{ if this value is greater than } \delta\text{, otherwise } \delta$$

where δ is a positive value close to 0, for instance 10^{-6}, which avoids sites with null weight. Taking into account the weight adjusting and the size value measure, the algorithm to generate a WSCVT is outlined in algorithm 2. Figure 4 shows the resultant Voronoi Diagram for 100 generators with weights from 1 to 100.

Algorithm 2 Weighted Spherical Centroidal Voronoi Tessellation (WSCVT)

```
                                          ▷ Place nodes on the surface of a sphere
Input: A set of weights $W = \{w_1, \ldots, w_n\}$.
Output: A set of points $P = \{p_1, \ldots, p_n\}$, each point corresponds to the position of
    a node distributed according $W$.
1: Let $P$ be a initial tentative point distribution on a unitary sphere $\mathcal{S}$.  ▷ Possibly a
   random distribution
2: Let $D$ be the desired values ($d_i = \frac{w_i}{\sum w_i}$)
3: while $\varepsilon_{\max} > \varepsilon$ do          ▷ it has not achieve the desired threshold
4:     Let $\mathcal{V}$ be the WSVT of $P$.
5:     Let $\varepsilon_{\text{actual}}$ be the maximum (or average) difference between desired size values
       ($d_i$) and actual size values ($\frac{\text{area}(G_i)}{\text{area}(\mathcal{S})}$) of the regions of $\mathcal{V}$.
6:     for all region $G_i$ of $\mathcal{V}$ do
7:         Let $p_i \in P$ the generator of $G_i$.
8:         Replace $p_i$ with the spherical centroid of $G_i$.
9: return $P$.
```

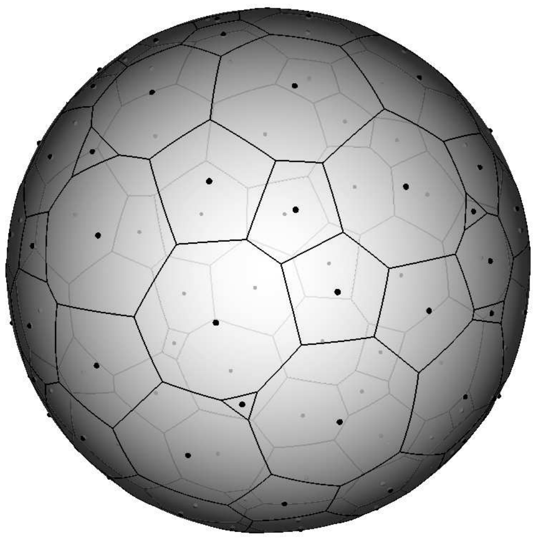
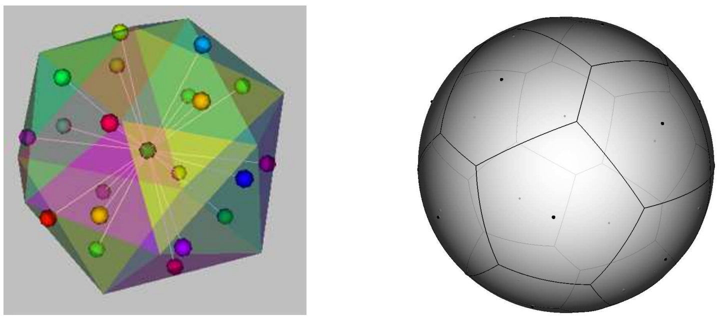


Fig. 4. The Voronoi Diagram of 100 generators, with weights from 1 to 100, with an error less than 5×10^{-4}. Transparence has been added to show the back of the sphere.

WSCVT can be applied to the first level of the Spherical Layout if the weight of each root's child measures how wide is the representation of the subtree rooted on it. Then each Voronoi region is projected to the outer spheres (Figure 2(b)) to define the pyramid that delimits the regions to place the descendant nodes.

4 TriSphere and WSCVT: A comparison and conclusions

In this section, we present a brief comparison between the TriSphere and the Weighted Spherical CVT algorithm. For each one, we generated 4 trees with different size; 20, 50, 1000 and 1500 nodes each. In figure 5, because the number of nodes has the form 20×4^i, where i is a positive integer, the resulting distribution using the TriSphere algorithm is perfect; as well as the WSCVT one. In figure 6 in order to allocate 50 nodes using the TriSphere algorithm, the icosahedron must be divided once by four. The resulting figure has 80 faces, which means than 37.5% of the sphere's surface is not used. Figures 7 and 8 also shown how the TriSphere algorithm cannot achieve an optimal distribution (21.875% of the surface is not used in the first case and 70.70% in the second one), whereas the WSCVT algorithm always does.



(a) TriSphere algorithm. (b) WSCVT algorithm.

Fig. 5. Same tree in both cases, 20 leaves to distribute on the surface of the sphere.

5 Future Work

WSCVT algorithm can still be highly improved. Due to the weighted part of the algorithm being based on a non-weighted one, some wrong edges may exist, and then some overlapping regions may appear, specially when the differences between the weights are considerable (the weight set has a high variance). One easy solution, but not always effective, is to traverse the edges of the Delaunay Triangulation, and swap every wrong edge. Let abc and acd be two adjacent triangles, with weighted circumcenters m_b and m_d respectiverly, the edge ac is wrong if

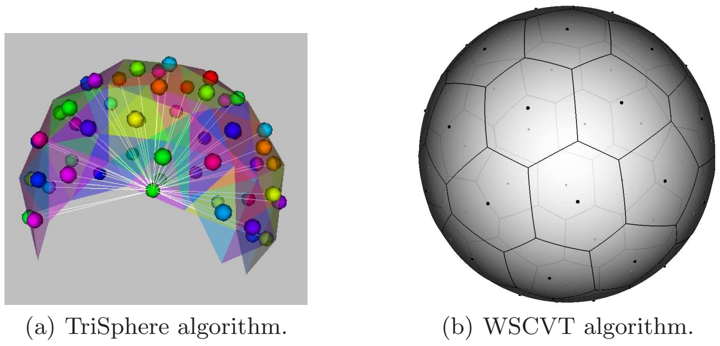

(a) TriSphere algorithm. (b) WSCVT algorithm.

Fig. 6. Same tree in both cases, 50 leaves to distribute on the surface of the sphere.

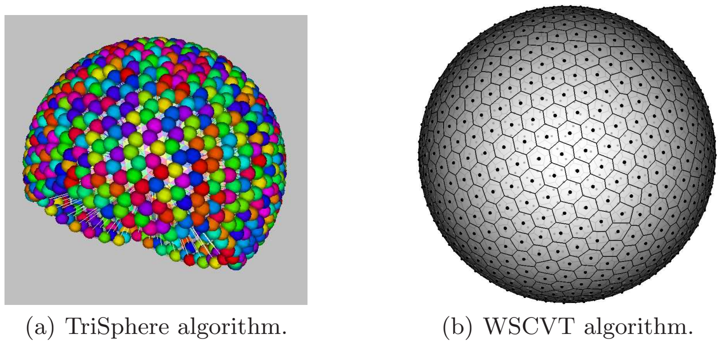

(a) TriSphere algorithm. (b) WSCVT algorithm.

Fig. 7. Same tree in both cases, 1000 leaves to distribute on the surface of the sphere.

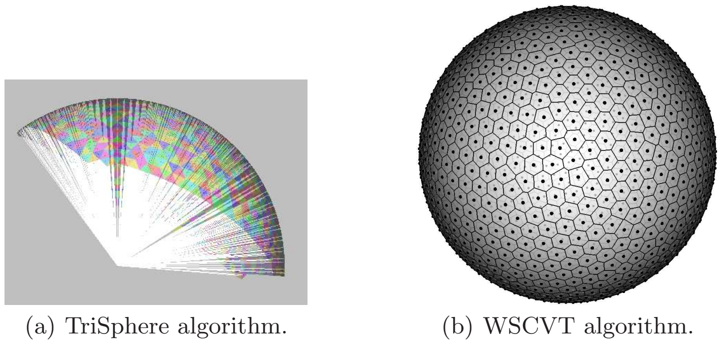

(a) TriSphere algorithm. (b) WSCVT algorithm.

Fig. 8. Same tree in both cases, 1500 leaves to distribute on the surface of the sphere.

$w\text{-}distance(d, m_b) < w\text{-}distance(b, m_b)$ or $w\text{-}distance(b, m_d) < w\text{-}distance(d, m_d)$, and it must be changed by the edge *bd*. This procedure reduces the existance of wrong edges, althought it cannot always eliminate all wrong edges, because it may happend that the edge *bd* is also a wrong edge. It can be added to the condition in the `while` sentence in algorithm 2 that the Delaunay Triangulation has no wrong edges, to avoid the few cases where swapping edges is not enough. This is a time consuming solution to the problem and not an adequate one, therefore we are working on designing a complete weighted algorithm. Furthermore, we are looking into the possibility of applying this algorithm to just a portion of the surface.

Acknowledgments. This work was partially supported by the PGI 24/N020, 24/ZN12 and 24/ZN19, Secretaría General de Ciencia y Tecnología, Universidad Nacional del Sur, Bahía Blanca, Argentina.

References

1. Martín Larrea, Sergio Martig and Silvia Castro, 2007. “Spherical Layout: Layout for 3D Tree Visualization”. IADIS 2007 Multi Conference on Computer Science and Information System, 2(1), pp.91–98. ISBN:978-972-8924-39-3.
2. Martín Larrea, Sergio Martig and Silvia Castro, 2006. “Thesis Overview: Spherical Layout for 3D Graph Visualization”. Journal of Computer Science and Technology, 7(1), pp.112–113. ISSN:1666-6038.
3. Michael Balzer, Oliver Deussen, 2005. “Voronoi Treemaps”. IEEE Symposium on Information Visualization (InfoVis 2005), pp. 49–56, ISSN: 1522-404X.
4. Jeff S. Jenness 2008. “Calculating areas and centroids on the sphere”. Poster presented at Arizona/New Mexico Wildlife Society Meeting. Albuquerque, New Mexico, USA (2008) and 28th Annual ESRI International User Conference. San Diego, California, USA.
5. Qiang Du, Vance Faber and Max Gunzburger. 1999. “Centroidal Voronoi Tessellations: Applications and Algorithms”. SIAM Review, 41(4) pp. 637–676.
6. Qiang Du, Max D. Gunzburger and Lili Ju. 2003. “Voronoi-based finite volume methods, optimal Voronoi meshes and PDEs on the sphere”. Computer methods in applied mechanics and engineering, 192 pp. 3933–3957.
7. Geoffrey A. Womerdorff. 2008. “Spherical Centroidal Voronoi Tessellations: point generation and density functions via images”. Master of Science Tesis, Florida State University.
8. Qiang Du and Lili Ju. 2005. “Finite Volume Methods on Spheres and Spherical Centroidal Voronoi Meshes”. SIAM J. Numer. Anal., 43(4) pp. 1673–1692, ISSN: 0036-1429
9. Hyeon-Suk Na, Chung-Nim Lee and Otfried Cheong. 2002. “Voronoi diagrams on the sphere”. Comput. Geom. Theory Appl., 23(2) pp. 183–194, ISSN: 0925-7721.