International Scholarly Research Network
ISRN Computer Graphics
Volume 2012, Article ID 609234, 10 pages
doi:10.5402/2012/609234

Research Article

Hyperbolic Wheel: A Novel Hyperbolic Space Graph Viewer for Hierarchical Information Content

Ho-Ching Lam[1] and Ivo D. Dinov[2]

[1] *Department of Computer Science, University of California, Los Angeles, CA 90095, USA*
[2] *The SOCR Resource (BH 9432), Department of Statistics, University of California, 8125 Mathematical Sciences Building, Los Angeles, CA 90095-1554, USA*

Correspondence should be addressed to Ivo D. Dinov, dinov@stat.ucla.edu

Received 13 September 2012; Accepted 1 October 2012

Academic Editors: Y. Okada and G. Patanè

Tree and graph structures have been widely used to present hierarchical and linked data. Hyperbolic trees are special types of graphs composed of nodes (points or vertices) and edges (connecting lines), which are visualized on a non-Euclidean space. In traditional Euclidean space graph visualization, distances between nodes are measured by straight lines. Displays of large graphs in Euclidean spaces may not utilize efficiently the available space and may impose limitations on the number of graph nodes. The special hyperbolic space rendering of tree-graphs enables adaptive and efficient use of the available space and facilitates the display of large hierarchical structures. In this paper we report on a newly developed advanced hyperbolic graph viewer, Hyperbolic Wheel, which enables the navigation, traversal, discovery and interactive manipulation of information stored in large hierarchical structures. Examples of such structures include personnel records, disc directory structures, ontological constructs, web-pages and other nested partitions. The Hyperbolic Wheel framework provides an intuitive and dynamic graphical interface to explore and retrieve information about individual nodes (data objects) and their relationships (data associations). The Hyperbolic Wheel is freely available online for educational and research purposes.

1. Introduction

Graph structures are frequently used for presenting hierarchical and linked data [1, 2]. One special type of graphs is hyperbolic trees, which are composed of nodes (points or vertices) and edges (connecting lines), and are visualized on a non-Euclidean space [3, 4]. Graph views in Euclidean spaces show distances between nodes as straight lines proportional to corresponding edge weights [5]. However, visualizations of large graphs in such Euclidean spaces do not utilize efficiently the available space and impose limitations on the number of nodes for the graph [6]. The special cases of large tree-graphs can be efficiently displayed in hyperbolic spaces. Here, we report on a newly developed advanced hyperbolic graph viewer, Hyperbolic Wheel, which provides the infrastructure for navigation, traversal, discovery, and interactive manipulation of information stored in such large hierarchical structures. Examples of such data structures include personnel records, disc directory structures, ontological systems, web-pages, and other nested partitions. The Hyperbolic Wheel provides an intuitive and dynamic graphical interface to explore and retrieve information about individual nodes (data objects) and their relationships (data associations). The entire Hyperbolic Wheel project is open-source and the Java code is available under LGPL license to the entire community.

There is a diverse array of data visualization techniques [7]. The choice of a computational, algorithmic, or visualization technique for processing or navigating large datasets is typically based on the intrinsic characteristics of the data. Information visualization of independent objects or unstructured datasets is more challenging as little or no dimensionality reduction may be possible in these cases. On the flip side, exploration of data with well-defined structure may be significantly simplified by employing graphical visualization techniques [8–10]. Tree-maps [11] employ

color-coded space-filling rectangles to represent (tree) hierarchical structures. Force-directed graph visualization [12] utilizes an edge-bundling method and a self-organizing approach to model node connectivity using flexible (contracting/expanding) springs. This approach does not require hierarchical data and reduces clutter by visibly exposing the high-level structural patterns in the data. Multiscale graph visualization allows interactive representations of hierarchical information [13] and provides scalable visualization with abstraction of detail and clutter at different scales [14]. Semantic maps introduce a new visualization technique based on semantically annotated data and the implicit impartation of knowledge [15].

There is a constant push to increase the amount and complexity of information that can be coherently, efficiently, and interactively displayed on a drawing canvas. The hyperbolic graph framework [16, 17] allows for displaying and manipulating large hierarchical datasets via smooth transformations of location and focus. Conventional graph display approaches map hierarchical data into large regions and provide scrolling or panning functionality to navigate the larger display canvas. This approach may hide node relationships between visible and invisible portions of the graph. Another interesting graph layout technique for drawing directed acyclic graphs is the H3 3D hyperbolic graph [18–20]. The H3 graph viewing mechanism utilizes the hierarchical graph organization to display node-cluster specific information using spanning trees. As the volume of 3D hyperbolic spaces increases exponentially in the periphery (in terms of the hyperbolic space metric), H3 provides a mechanism for displaying large number of nodes and edges. This 3D hyperbolic navigation has been tested on graphs of the order of 20,000 nodes and facilitates a Focus + Context view of hierarchical structures and minimizes visual clutter. Other 3D graph visualization techniques use hyperbolic geometry on spherical surfaces and enable dynamic fly-through-the-graph functionality avoiding node/edge boundaries [21].

Hierarchy visualization may also be achieved using radial, space-filling methods, for example, DocuBurst [22], InterRing [23], coaxial viewer [24], and Sunburst [25, 26]. These offer the advantages of efficient utilization of the drawing canvas and manual/automated multifocus selection. The ASK-GraphView is a large graph visualization framework [27], which allows scalable clustering and rendering of hierarchical and weighted undirected graph structures. This approach trades off interactivity to gain significant scalability enabling visualization of graphs containing millions of edges. The graph navigation is driven by expanding or collapsing individual clusters. Focus + Context techniques, like document lens, perspective wall, and fish-eye, facilitate the rendering of complex tree structures [17, 28, 29]. However, there are many alternatives to display tree structures, for example, zoomable adjacency matrices, hierarchical edge bundles, geometric edge clustering [30–32], and so forth. Some challenges of these approaches may include inadequate spacing between leaf nodes, obscuring of the hierarchy level, lack of context display, or inconsistent spacing between nodes. Some of these problems are addressed by the new Hyperbolic Wheel viewer, which also provides a dynamic mechanism for linking nodes with external web-pages.

2. Materials and Methods

This section describes the design and implementation details of the Hyperbolic Wheel architecture and its practical utilization to interactively render large hierarchical graph structures.

2.1. Layout of the Unit Plane. The Hyperbolic Wheel layout is a unit disc with polar coordinates which represents a 2D hyperbolic space. "Unit disk" indicates that the radius of all nodes is always less than or equal to 1. A root of a tree is referred to as the origin and all other nodes are placed radially about the root. All the nodes' coordinates are determined by their (radial) level and angle θ. The radii and angles are determined using a recursive algorithm. The radial level is simply the depth of a node and is determined by the distance between the root and the node:

$$\text{Radius} = \frac{\text{Level}}{\text{TreeHeight}}, \tag{a}$$

where TreeHeight is the total number of levels of the tree structure (i.e., the maximal length between all nodes and the root).

The radii of all nodes are ≤ 1. For a particular node, to compute the angle, θ, we need to know the starting radian and the angular quota of that node. The starting radian is a property of a sector (more details about sector structure are available in the Structure of sectors section). When we draw a sector, we need to know where to start and where to end. For example, if a root has four children, the first sector will have angular-range from 0 to $\pi/2$, the second sector will have a range from $\pi/2$ to π, and so on. Thus, the starting angular measures of the first and the second child are 0 and $\pi/2$, respectively. For a given parent node, the starting angular measure for the Nth child is given by:

$$\text{Starting Radian} = N \times \frac{\text{Parent's Angular Quota}}{\text{Number of Siblings}}. \tag{b}$$

The parent's angular quota represents how much radial space is available to be assigned to its children. Obviously, the root has angular quota of 2π and all other (deeper level) nodes have progressively smaller angular quotas. We used the following straightforward angular quota assignment of each node:

$$\text{Angular quota} = \frac{\text{Parent's Angular Quota}}{\text{Number of Siblings}}. \tag{c}$$

This assignment ensures that all children of a particular parent utilize, but not exceed, their parent's quota and there is no overlap with children from different parents. This approach assigns equal quotas to all the children of the same parent, which results in a visually appealing graph. Once we have the starting radial and the angular quotas, the actual angle for a node, θ, can be determined as follows:

$$\theta = \text{Starting Radian} + \frac{\text{Angular Quota}}{2}. \tag{d}$$

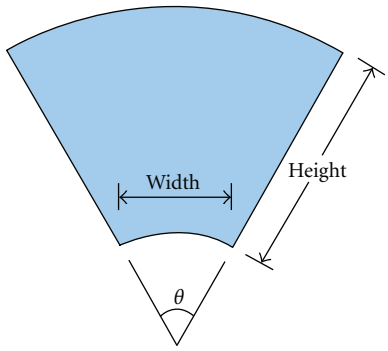

Figure 1: Schematic of a sector.

According to (d), a node is located in the center of its corresponding sector. In fact, the angular quota assignment, (c), is only a primitive equation. A child can get only a portion of its parent's quota, up to the entire parent quota, if it is the only child. In general, the quota may be exhausted quickly even after 3 or 4 levels in the hierarchy. Exhausting the quota means that the angular quota of a parent node is split between the children into too many (smaller) fragments. This may result in some of the children being invisible. The quota may be exhausted when the number of nodes grows exponentially as the depth level of the graph increases. As we navigate to a deeper level in the hierarchical structure, we have a limited partial quota assigned to each node. Even at 3rd or 4th levels, we may encounter this problem when the number of children is large. Therefore, we need a more sophisticated equation to boost the children-node visibility. To do that, we need to revise (c) and consider all the successors of a node rather than just its immediate siblings

$$\begin{aligned}&\text{Node's Angular Quota}\\&\quad=\text{Number of Successors}\\&\qquad\times\frac{\text{Parent's Angular Quota}}{\text{Number of Parent's Successors}}.\end{aligned}\tag{e}$$

In this equation, the smallest nodal angular quota will be assigned sufficient angular space for all of its successors. The more successors a node has, the larger its assigned quotas will be (see the Results section).

2.2. Structure of Sectors. A sector is like a coat of a node. It is the wrapper interface that enables the user to see and interact with the node. A node may be considered as a pair of a location (center) point and a sector, and rendered in an area (sector) covering that node. In the Hyperbolic Wheel viewer, sectors can be turned on or off (all together, but not individually). Users may select whether they want to see the hyperbolic disk, the tree structure or both in the same display.

There are 3 kinds of nodes: the Root, a nonLeaf or a Leaf node. They are all constructed as sectors. We compute a number of anchor points around a node and draw a sector by connecting all those points together using quadratic Bézier curves [33]. The number of anchor points varies for each node. A sector is formed by four curves: top, bottom, left, and right curves (Figure 1).

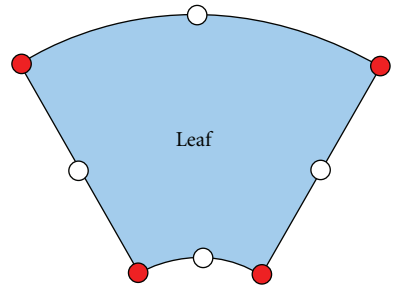

Figure 2: Hyperbolic leaf.

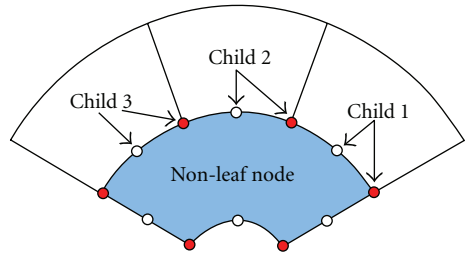

Figure 3: C-node structure.

A Leaf has the simplest structure among the three kinds of nodes (Figure 2). It has 8 points and all leaves have exactly the same structure and number of points. There are 4 endpoints (red dots) and 4 control points (white dots) of the Bézier curve. A nonleaf, child node, or C-node, on the other hand, has the most complicated structure (Figure 3).

When a node's angular quota is small, drawing the node 4 borders using quadratic Bézier curves requires good approximations of the curves in the hyperbolic plane. If C-nodes have a large number of children and their angular quotas are too small for the approximation empty space may appear between parent and children. To solve this challenge, we use a dynamic number of points on a C-node's top curve. The idea is to reuse the bottom curve of the node's children to form the node's top curve and therefore avoid gaps between parents and children. In practice we use the following equation to compute the number of points of a C-node is depending on the number of its children:

$$\text{Number of points} = 2\times(\text{number of children}) + 6.\tag{f}$$

Figure 3 illustrates the construction of the left, right, and bottom curves for a C-node by taking 2 points for every child and adding another 6 more points. Figure 4 shows the number-of-points allocation for the root and any other node with angular quota of 2π. Note that in a degenerate case when a root has only one child, that child will inherit the root's angular quota of 2π. There are two different possibilities for these full angular quota nodes. The first one is similar to the C-nodes except that they don't have the left, right, and bottom curves. From Figure 3, we see that when the angular quota is 2π, the sector is no longer a sector shape but rather a complete track. The second possibility is a special case which occurs when a node has angular quota of 2π and has only one child. In the first case, a C-node will only take 2 points

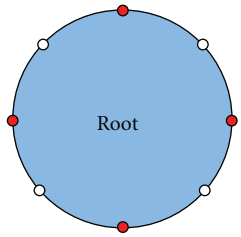

Figure 4: Root node.

from each child, and 2 points are insufficient to form a circle-like shape of the C-node. So, we compute the sector point separately in order to approximate a circle. In these cases, we use 8 points and we divide the radian 2π in 8 equal segments. By using constant angles and constant radii, we can compute those 8 anchor points.

2.3. Mapping. Below are some details about the Hyperbolic Wheel viewer. *First*, the unit hyperbolic disc is only used as a conceptual idea to help us develop an understanding of the hyperbolic space layout. The coordinates of a node are stored in a Java hash-table indexed by the nodes and we compute the shape and locations of all nodes on the fly when we render the entire graph as a planar Euclidean structure. The concept of the unit hyperbolic disc facilitates the understanding of how the user interacts with the graph. Thus, we employ the unit disc for better conceptual explanation. *Second*, all the coordinate computations and angular assignments from Sections 1 and 2 occur in the unit hyperbolic disc. Although points are computed in polar coordinates, they are actually stored as Euclidean coordinates. The reason for this choice is to decrease the computational complexity of the hyperbolic graph rendering engine. Therefore, all functions used in the calculations are based on objects' (node's or edge's) Euclidean coordinates. *Third*, every point on the unit disc is relatively fixed. This means that the position of each node is fixed relative to the other node's coordinates. As an analogy, a unit disc is like a paper circle where computing object coordinates is like drawing points on the 2D disc. If a user moves or rotates the paper disc in 3D, the individual points remain in their initial *relative* positions. *Fourth*, users interact with the hyperbolic graph by moving the unit disc in space-that is, altering the scene observation point. Conceptually, the hyperbolic graph we render on the screen is the shadow of the real graph and we can only move the shadow by moving the unit disc. The reason for separating the coordinates of the unit disc and the screen is to improve the efficiency of the calculations. Also, the arithmetic in Euclidean coordinates enables adding of novel future mappings to the viewer. *Finally*, mapping is applied to point coordinates only. All lines, arcs and nodes in the hyperbolic graph are drawn directly on the screen using their pure Euclidean space coordinates.

In summary, these mappings are used for converting points between the unit disc and the screen canvas. So far, we are using two specific mappings: Poincaré disc model [34] and Klein model [35]. Poincaré disc model is a conformal

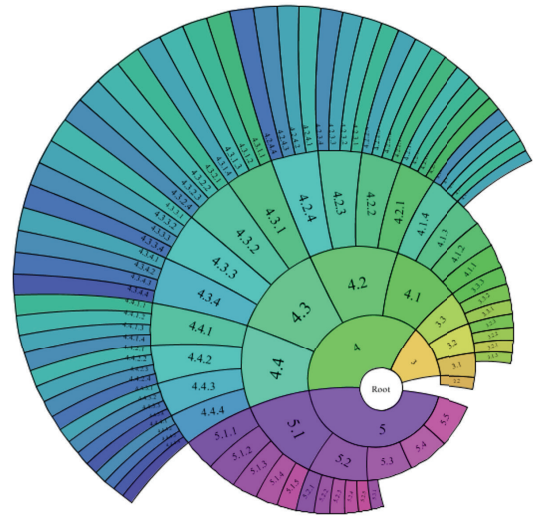

Figure 5: Poincaré disc model.

model, that is, an angle-preserving model, where lines are represented by segments of circles perpendicular to the disc's boundary [36]. Circles that exclude the origin are shrunk by transformations between the Poincaré hyperbolic disc space and the Euclidean disk. The forward and reverse transformations are given by the complex variable mappings:

$$T(z) = \frac{\theta z + P}{1 + \overline{P}\theta z}, \qquad T'(z) = \frac{\theta' z + P'}{1 + \overline{P}'\theta' z}, \tag{1}$$

where P and $\overline{P}$ are complex conjugates ($|P| < 1$, and $P\overline{P} = 1$), P encodes the radial panning of the graph between the origin and P, and $|\theta| = 1$ represents the angular rotation. The inverse transformation maps the Euclidian display coordinates to the (Poincaré) hyperbolic space and is given by an analogous Möbius transformation, $T'(z)$, where $P' = -\bar{\theta}P$ and $\theta' = \overline{\theta}$ [37].

The Klein model, on the other hand, is a projective model where Euclidian lines are represented as chords [17]. The forward and reverse mappings between the Poincaré and the Klein disks are also Möbius transformations:

$$T(z) = \frac{2z}{1 + \overline{z}z}, \qquad T'(z) = \frac{1 - \sqrt{1 - \overline{z}z}}{\overline{z}z} z. \tag{2}$$

Both models are trying to spread and expand nodes when a user navigates the hyperbolic graph (e.g., by clicking and dragging the mouse) [38]. The further a node is from the root-node, the less screen real-estate is available for the rendering of that node in the periphery of the graph. This spreads apart nodes in the center of the hyperbolic disc, and contracts nodes towards the disc boundary far from the disc center. Graphically speaking, the Poincaré disc model provides a more appealing graph rendering than the Klein model [39] because the former separates naturally and non-linearly the graph nodes (Figures 5 and 6). In the Poincaré disc model, the angle between lines is preserved, so when we spread the nodes, the shape of the area close to the root-node shows little distortion. In the Klein model, however, as

the nodes are spread out, the shape of that area near the root-node will be significantly distorted.

2.4. Hyperbolic Wheel Functionality

2.4.1. Text Drawing. Text displays are used to convey information regarding each of the graph nodes. A simple horizontally displayed text is no longer suitable for this hyperbolic viewer, as the node names may fall beyond the boundary of their corresponding sectors. We developed a new schema to display the node names according to the orientation of the node (and its sector). First, we adjust the text to match the angle of the corresponding node-sector, relative to the position/orientation of the root-node. A simple use of the angular properties of the sector does not provide a naturally appearing node name display within the boundaries of the node sector. As we navigate the hyperbolic graph, the angle of each sector with respect to the root-node constantly changes. So, the angle of the node-name text must be obtained dynamically from the corresponding sector screen coordinates. We use the root's coordinates as an origin of a local coordinate system and compute the relative angle of a sector with its coordinates. Then, we adjust the font size to ensure that the entire node-name fits within the node-sector. Originally, we used the width (Figure 1) of a sector as the font size, however, this approach only works for sectors with small angular quotas. The main challenge comes from the fact that the shapes of some sectors are close to rectangles, where the ratio of width/height is small, whereas some sectors are extremely nonrectangular. Sectors with large angular quotas may have a ratio of width/height larger than 1, which makes the text excessively large. To solve this problem, we designed a function to determine the font size by using the width and the height of a sector as independent variables. Specifically, we defined the font size by:

$$\text{font size} = \begin{cases} \dfrac{\text{width}}{(1 + (\text{width}/\text{height}) \times 1.5)}, & \text{if width} < \text{height}, \\ \dfrac{\text{height}}{(1 + (\text{width}/\text{height}) \times 1.5)}, & \text{if width} \geq \text{height}. \end{cases} \tag{3}$$

When the ratio of width/height is small, which is the case for nodes with small angular quota, the font size is close to the width, otherwise, it will be a fraction of the height. This function produces visually appealing results for both small and large sectors. However, when the ratio is larger than 1, we need another modification to fit the text in the allotted sector space. To curve the text following the sector shape we first compute the equation of the quadratic Bézier curves by using three points: one control point from each of the left and right curves (Figure 1) and the coordinate of the node. We print each character of the node name along the curve by using that Bézier equation, which produces more visually consistent test in the graph, Figure 9.

2.4.2. Color Scheme. We designed a new color scheme to enhance the visibility of the relations between different nodes and branches of nodes in the hyperbolic graph. Colors and glyphs are helpful for discriminating different nodes and node relations. There are two essential requirements that a good color scheme must have: the ability to clearly identify parent and child nodes, and depict neighboring nodes descending from different branches of the hyperbolic tree. We used the HSB (HSL) color model [40] to assign node sector color based on a pseudo function call Color (<*Hue*>, <*Saturation*>, <*Brightness*>). In our color scheme, the root and the nodes in the 1st hierarchy level are treated differently. For nodes in higher hierarchy levels, we decided to lower the brightness (dim) of the node color proportional to the distance of the node/sector level from the root. Thus, nodes further away from the root (higher hierarchy levels) have reduced color brightness, that is, Color ~ *(*<*Hue*>, <*Saturation*>, *Level)*. The idea is to determine the Hue by the angle of a node sector, however, purely angular values will produce similar colorings, whereas we need the neighboring nodes to have distinct colors. Instead of using the angular properties, each node will inherit a Hue from its parent and then add its own Hue, which is determined by its number of children. Therefore, Color ~ *(parent's Hue + node's Hue,* <*Saturation*>, *Level)*. This requires that we have an initial Hue at the beginning, but all nodes cannot inherit the Hue from root-node as the root Hue will propagate to all children. Thus, the initial source of Hue comes from the 1st level of the hierarchy. The angular properties will only be used in determining the Hue of nodes in the 1st level. Hence, the angle is actually a perfect reference of the range of sector colors. The full range of the color can be referenced as a real value in the interval from 0 to 2π. Notice that the number of nodes in the 1st hierarchy level is usually small and their angular quotas will be significantly large, relative to the nodes in deeper hierarchy levels. Thus, the difference between the colors determined by the angular properties in the 1st hierarchy level will be obvious. The sector color saturation is simply a constant and the final pseudo color

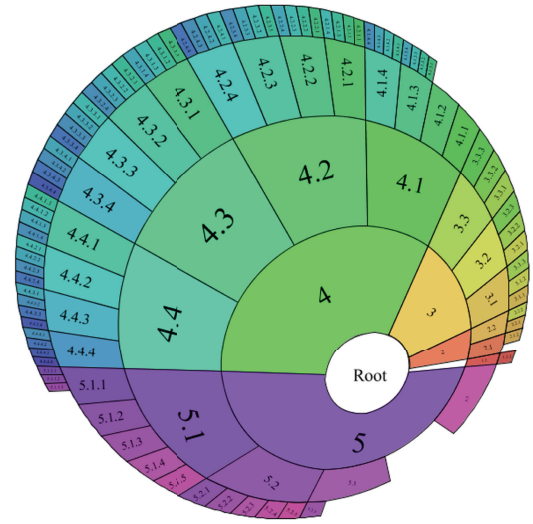

Figure 6: Klein disc model.

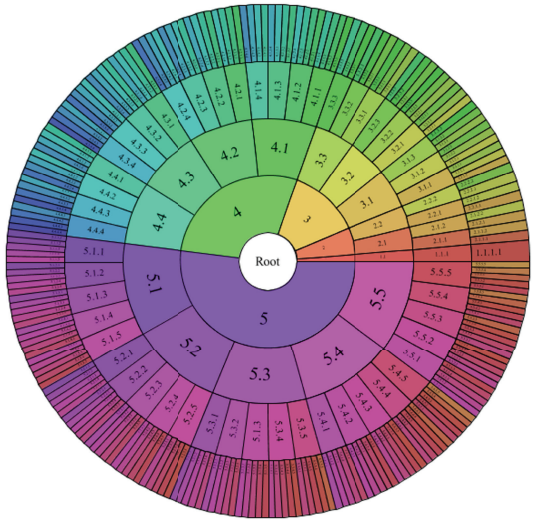

Figure 7: Sector coloring schema.

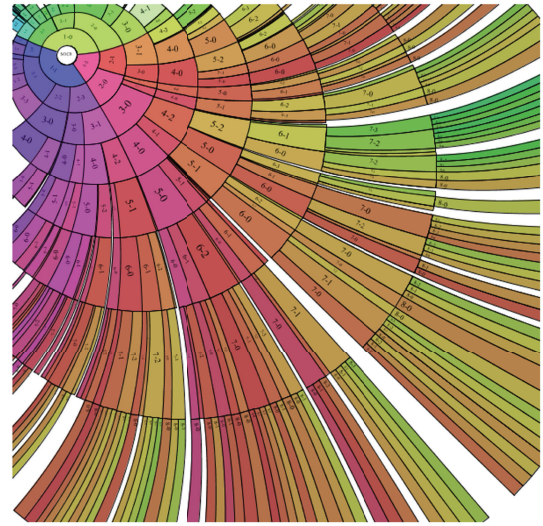

Figure 8: Visualization of 8-level deep nested hierarchical structures.

function is given by Color $\sim$ *(parent's Hue + node's Hue, Constant, Level)*, Figure 7. This coloring schema allows quick visual cues to the node's ancestry-branch (parents). The rainbow color-spectrum spans the angular range $[0 : 2\pi]$ and the color brightness is a direct indicator of the radial distance between the node and the tree-root. As a side note, the structure-based coloring proposed with InterRing [23] provides a more advanced mechanism of assigning colors to nodes based on their radial and angular coordinates.

3. Results

3.1. Graphs with Different XML Structure. The viewer takes a XML file as input and generates the graph in a browser. Any acyclic graph can be rendered by the Hyperbolic Wheel viewer, however the quality of the results may vary depending on the hierarchical data structure.

3.1.1. Ideal Structure (Uniformly Distributed). The ideal graph is a balanced graph in which the nodes within a same level have about the same number of children independently from their parents. In other words, whether the nodes are from the same parent or not, once they are at the same level, they should have about the same number of children. Generally speaking, for any two nodes at the same level, the number of children of one node should not be 10 times more than the other. Figure 7 depicts an example of a preferable graph. In Layout of the unit plane section, we mentioned that we adopted a different approach to assign the angular value so that every nodes of high-levels graphs are still visible and Figure 8 demonstrate a 8-level graph which is also randomly generated with a condition that each nonleaf has 0$\sim$5 children.

3.1.2. Web Site Examples (SOCR Site). This is the application of Hyperbolic Wheel on the website Statistics Online Computational Resource (SOCR). Figure 9 demonstrating

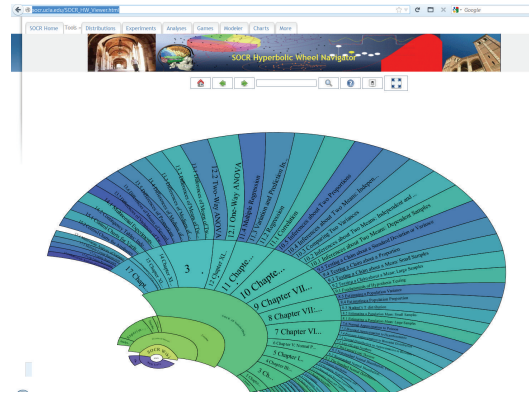

Figure 9: Visualization of SOCR EBook table of contents.

the SOCR's EBook (http://wiki.stat.ucla.edu/socr/index.php/EBook) and we see the chapters and sections are nicely displayed.

3.1.3. Show More Results with the Data in These XML Hyperbolic Viewer Structures

Case 1. Figure 10 shows that if the root has only one child then it will inherit the entire root's angular quota and become a track instead of a sector. This will continue if every node has only one successor all the way down the tree. This diagram shows the resource of the Neuroscience Information Framework (http://socr.ucla.edu/test/NIF_TR_Resources.xml).

Case 2. Figure 11 shows a graph structure of the core National Science Foundation resources that the Hyperbolic Wheel renders imperfectly because the graph is heavily skewed and unbalanced and most of the successors belong to one parent. The reason for the strange appearance of the graph is that all the curves in the Hyperbolic Wheel are not arcs from a perfect circle but quadratic Bézier curves which

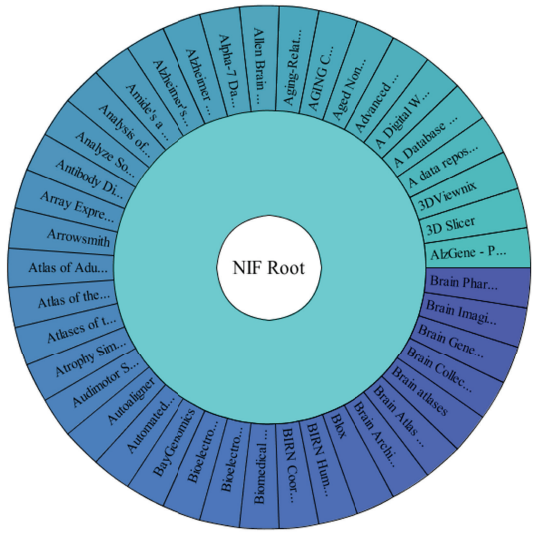

Figure 10: One-level deep hierarchies (NIF).

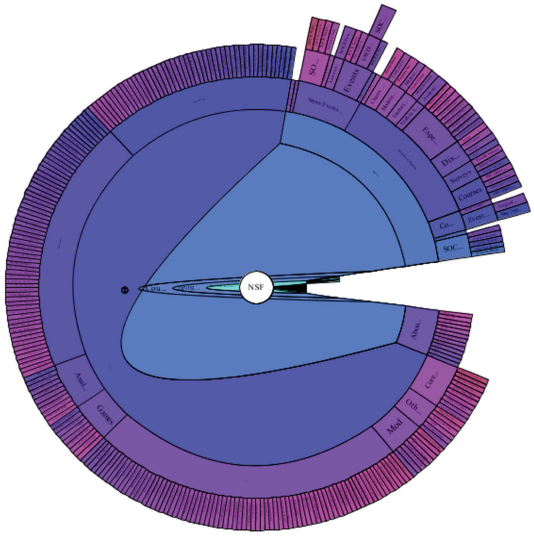

Figure 11: Imperfect hyperbolic wheel structure rendering (NSF).

depend on only one control point, more details are available in the section Structure of sectors (http://socr.ucla.edu/test/NSF_TR_Resources.xml).

Case 3. This example renders well, however, since some of the nodes have only one child the viewer constrains the angular space for descendent nodes when they are moved in focus. Here we rendered the hierarchy of iTools resources [41]. Detailed explanation of this case is included in the section *Limitations*. Figure 12 is a similar but more realistic example of Figure 16 which focuses only on this particular problem (http://socr.ucla.edu/test/iTools_TR_Resources.xml).

Case 4. Figure 13 shows an extremely unbalanced hierarchical structure where some parents have too many children and some do not have children at all. This hierarchy

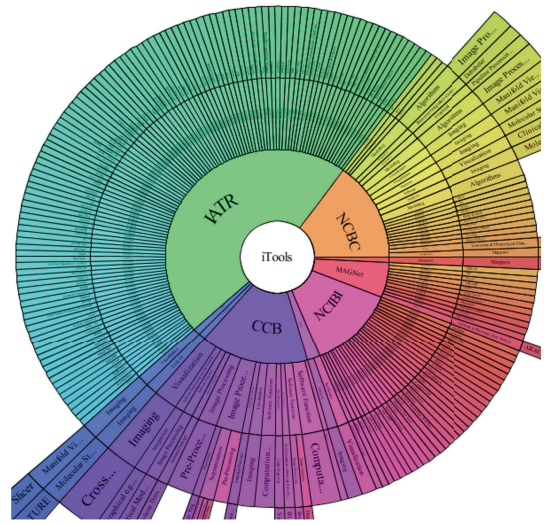

Figure 12: Hyperbolic wheel rendering of nodes with few children (iTools).

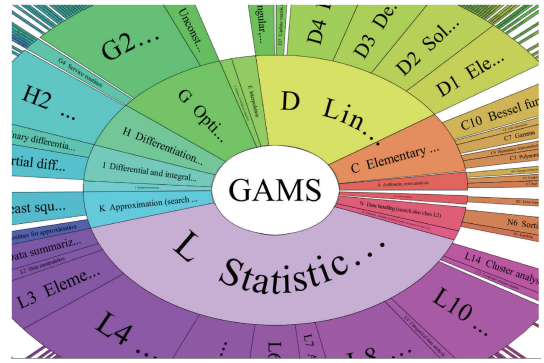

Figure 13: Hyperbolic wheel rendering of nodes with too many children (GAMS).

represents the NIST Guide to available mathematical software resources. Zooming in to nodes which are too small relative to the whole graph will generate a nice display of the local node information (http://socr.ucla.edu/test/GAMS_NIST_Resources.xml).

Case 5. This is an example having a huge number of nodes. This hierarchical structure represents the entire UMLS Knowledge Source Server (UMLSKS) database. The graph can be possibly moved and zoomed but it is difficult to practically navigate. Figure 14 shows this data structure where most of the nodes names are visually too small due to the adopted font size calculation (http://socr.ucla.edu/test/UMLS_HyperTree.xml).

Case 6. Figure 15 has an even larger graph than Figure 14. This structure contains about 100,000 nodes including the entire Math Genealogy database. Although some of the nodes are readable, the graph has poor interactive performance. This example is difficult to interactively navigate with

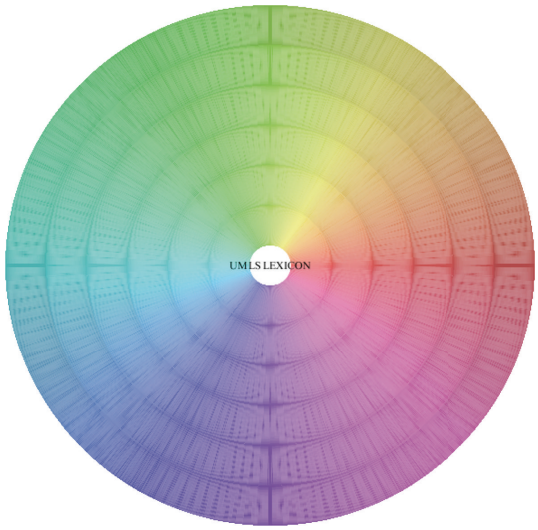

Figure 14: Hyperbolic wheel rendering of structures with huge number of nodes (UMLS).

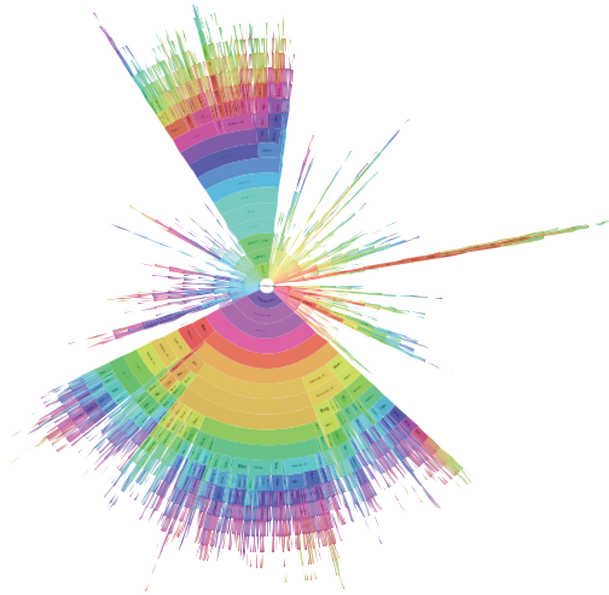

Figure 15: Hyperbolic wheel rendering of 100,000 nodes (Mathgenealogy).

due to excessive computation. For browser configuration using 300 MB JVM RAM, it takes about 135 seconds to draw the whole graph on an Intel Core2 with 1.80 GHz and 768 MB RAM (http://socr.ucla.edu/test/MathGenealogy_SOCR_April26_2007.xml).

4. Discussion/Conclusion

The new Hyperbolic Wheel viewer efficiently utilizes space by constructing a radial metric on a disc, assigning a sector-size for each node and morphing the shape of each sector based on a hyperbolic space metric. A node is no longer a vertex—it becomes a sector in the hyperbolic disc space. This enables users to interactively explore the information stored in the hierarchical structure. Each node, and its vicinity, is rendered, using shape and color, to indicate the position of the node, its relations to other branches of the data hierarchy and its distance from other nodes. The Hyperbolic Wheel also allows additional node information to be added into its sector (e.g., using pop-ups or glyphs) without increasing the complexity of the graph-structure or the graph-display. For instance, each of the nodes (sectors) in the Hyperbolic Wheel is a clickable object that links to an appropriate web-site providing additional information about the selected node. The Hyperbolic Wheel viewer is part of the statistical and visualization library supported by the Statistics Online Computational Resource (http://www.socr.ucla.edu/) [42, 43].

There are some similarities and some differences between the proposed Hyperbolic Wheel display and prior work on graph visualization using hyperbolic geometry. The focus + context based methods in 2D [17, 35] and 3D [19, 20] provide scalable visualization for a node-and-link based hierarchical tree structures. The Hyperbolic Wheel technique could be extended to 3D, however complex volume rendering of solid balls, or any 3 manifolds, are difficult and very time consuming, especially at high resolution. The radial

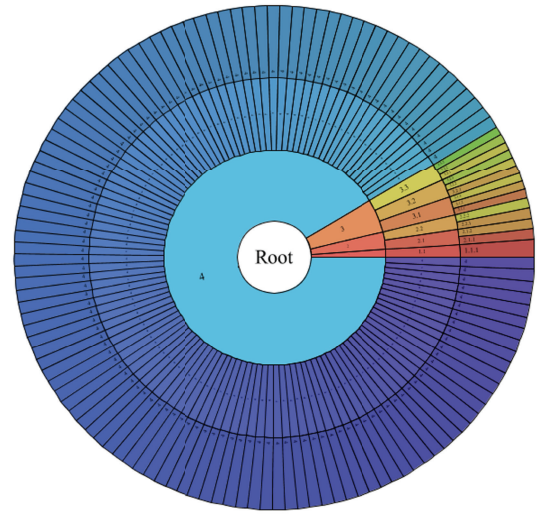

Figure 16: Hyperbolic wheel rendering of structures with number of parent nodes approximately equal to the number of their children.

space-filling Hyperbolic Wheel visualization of hierarchical tree relations may be most useful in 2D for enhanced view representing a mixture of layered, hyperbolic geometry-based and color-cues display of the information.

4.1. Features

4.1.1. Performance. In the framework of the Hyperbolic Wheel viewer, performance refers to how quickly and efficiently a hyperbolic graph can be rendered, and how smooth is the response of the Hyperbolic Wheel viewer. We used several methods to enhance this performance. First, we avoid

rendering nodes that are too small to be seen at the current resolution (zoom) level. The font size is a good predictor in determining which node may be visible, and therefore needs to be displayed. Even if a user can see the sector of a small node, it may be pointless to draw the node as the node name may be too small to read. In fact, small nodes may often represent a large portion of the graph in the hyperbolic space. Thus, hiding small nodes significantly improves performance.

4.1.2. Hiding Text. We hide most of the text when the graph is being dragged, moved, or manipulated. Text drawing is an expensive operation because every single character is warped (transformed, rotated, and adjusted), as mentioned in Text Drawing section, in order to fit it into a particular sector shape. So, hiding the text provides essential speed gain. Completely hiding the node name, however, may cause a user to lose their focus and orientation while traversing the hyperbolic graph. Our solution to this problem compromises between speed and content by only showing names of the node dragged by the user and its immediate children. All other node names are not rendered until the completion of the mouse manipulation step. This functionality enables a smooth uninterrupted interaction with the hyperbolic graph without loss of orientation/focus.

4.1.3. Drawing Fewer Edges. Another way to enhance performance is to draw fewer lines and curves. In most cases, nodes will be surrounded by other nodes in all directions: top, bottom, left and right. All the edges of these nodes are overlapping with the edges of their neighbors. Roughly half of the lines or curves drawn are redundant and therefore performance can be improved by avoiding duplicate lines or curves. However, this method can only be applied to rendering the outlines of the node sectors or the hyperbolic graph without colors. All four edges are required for filling a sector with a correct color and so this improvement does not apply to rendering the color graph nodes.

4.2. Limitations. The Hyperbolic Wheel viewer is designed to visualize large data structure having tree-like organization. This framework does not support cyclic graphs. In general, this may or may not be a problem depending on the data information and graph structure. A graph of a web site, for example, can be converted into an acyclic graph. Links from child to parent pages are usually for convenience purpose, they are not essential parts of the data structure. In the hyperbolic viewer, the user does not need "links" to "go back", as simple mouse drags enable graph navigation. However, if cycles are part of a data structure, then the Hyperbolic Wheel may not be an appropriate infrastructure for visualizing the data. For example, cycles in a social networking graph may be unavoidable.

Another potential limitation is that parent nodes of lower hierarchical levels (other than leaf nodes) may be drawn separate from their neighbors when the number of parents is about the same as the number or children (Figure 16). The reason for that artifact is that upper hierarchy levels distort the hyperbolic space the most. So, when the numbers of nodes in different hierarchical levels are about the same, the nodes in lower levels will be assigned significantly less screen space than the leaf nodes. Finally, the number of levels (associated with number of nodes) of graph is not scalable. This is a different limitation from the first two and its manifestation is machine dependent. More hierarchical levels may make the viewer extremely slow and difficult to navigate.

4.3. Future Improvements. A future improvement of the Hyperbolic Wheel will focus on displaying multiple metadata about the node (in addition to its name). With limited screen space, extra information will be displayed using alternative glyphs. For example we can use pop-ups to contain all the details of a particular node. Another improvement is to design a more sophisticated interface that contains an advanced panel or control center allowing users to manipulate graph setting, such as color scheme, performance control, and the graph I/O. Finally, improving the computational complexity and optimizing the rendering calculations using multithreading will significantly enhance user experiences.

Acknowledgments

The authors are indebted to the SOCR faculty, students, and staff for their motivation, encouragement and support for the development, deployment, and validation of the Hyperbolic Wheel viewer. These SOCR efforts are funded in part by National Science Foundation DUE Grants 0716055, 1023115, and 0442992, and by the National Institutes of Health Grants P41 RR013642 and U24-RR025736.

References

[1] H. J. Schulz and H. Schumann, "Visualizing graphs—a generalized view," in *Proceedings of the International Conference on Information Visualization (IV '06)*, pp. 166–173, July 2006.

[2] S. E. Kruck, F. Teer, and W. A. Christian, "GSLAP: a graph-based web analysis tool," *Industrial Management and Data Systems*, vol. 108, no. 2, pp. 162–172, 2008.

[3] J. A. Walter, "H-MDS: a new approach for interactive visualization with multidimensional scaling in the hyperbolic space," *Information Systems*, vol. 29, no. 4, pp. 273–292, 2004.

[4] J. Ontrup and H. Ritter, "Large-scale data exploration with the hierarchically growing hyperbolic SOM," *Neural Networks*, vol. 19, no. 6-7, pp. 751–761, 2006.

[5] F. Van Ham and J. J. Van Wijk, "Interactive visualization of small world graphs," in *Proceedings of IEEE Symposium on Information Visualization (InfoVis '04)*, pp. 199–206, October 2004.

[6] C. Chen, "Top 10 unsolved information visualization problems," *IEEE Computer Graphics and Applications*, vol. 25, no. 4, pp. 12–16, 2005.

[7] I. Herman, G. Melancon, and M. S. Marshall, "Graph visualization and navigation in information visualization: a survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 24–43, 2000.

[8] D. A. Keim, "Information visualization and visual data mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 1–8, 2002.

[9] S. Card, J. D. MacKinlay, and B. Shneiderman, "Information visualization," in *Human-Computer Interaction: Design Issues, Solutions, and Applications*, p. 181, 2009.

[10] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2011.

[11] B. Johnson and B. Shneiderman, "Tree-maps: a space-filling approach to the visualization of hierarchical information structures," in *Proceedings of IEEE Conference on Visualization*, pp. 284–291, 1991.

[12] D. Holten and J. J. Van Wijk, *Force−Directed Edge Bundling for Graph Visualization*, Wiley Online Library, 2009.

[13] N. Elmqvist and J. D. Fekete, "Hierarchical aggregation for information visualization: overview, techniques, and design guidelines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 3, pp. 439–454, 2010.

[14] T. Munzner, "A nested model for visualization design and validation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 921–928, 2009.

[15] K. Nazemi, M. Breyer, and C. Hornung, "SeMap: a concept for the visualization of semantics as mapsuniversal access," in *Human-Computer Interaction*, C. Stephanidis, Ed., Applications and Services, pp. 83–91, Springer, Heidelberg, Germany, 2009.

[16] J. Lamping and R. Rao, "Laying out and visualizing large trees using a hyperbolic space," in *Proceedings of the 7th annual ACM symposium on User Interface Software and Technology (UIST '94)*, pp. 13–114, 1994.

[17] J. Lamping and R. Rao, "The hyperbolic browser: a focus 1 context technique for visualizing large hierarchies," *Journal of Visual Languages and Computing*, vol. 7, no. 1, pp. 33–55, 1996.

[18] T. Munzner, "H3: laying out large directed graphs in 3D hyperbolic space," in *Proceedings of IEEE Symposium on Information Visualization*, pp. 2–10, October 1997.

[19] T. Munzner, "Exploring large graphs in 3D hyperbolic space," *IEEE Computer Graphics and Applications*, vol. 18, no. 4, pp. 18–23, 1998.

[20] T. Munzner, *Drawing Large Graphs With H3viewer and Site Manager*, Springer, 1998.

[21] J. Weeks, "Real-time rendering in curved spaces," *IEEE Computer Graphics and Applications*, vol. 22, no. 6, pp. 90–99, 2002.

[22] C. Collins, S. Carpendale, and G. Penn, "DocuBurst: visualizing document content using language structure," *Computer Graphics Forum*, vol. 28, no. 3, pp. 1039–1046, 2009.

[23] Y. Jing, M. O. Ward, and E. A. Rundensteiner, "InterRing: an interactive tool for visually navigating and manipulating hierarchical structures," in *Proceedings of IEEE Symposium on Information Visualization (INFOVIS '02)*, pp. 77–84.

[24] J. Zhang, Y. Zhang, L. Fu et al., "Coaxial interactive viewer: a multi-dimensional data visualization with spatial distortional views," in *Proceedings of the Visual Information Communication—International Symposium (VINCI '11)*, ACM, New York, NY, USA, 2011.

[25] J. Stasko and E. Zhang, "Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations," in *IEEE Symposium on Information Visualization (INFOVIS '00)*, pp. 57–65, October 2000.

[26] C. Stab, M. Breyer, K. Nazemi et al., "SemaSun: visualization of semantic knowledge based on an improved sunburst visualization metaphor," in *Proceedings of the World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-Media)*, pp. 911–919, 2010.

[27] J. Abello, F. Van Ham, and N. Krishnan, "ASK-GraphView: a large scale graph visualization system," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 669–676, 2006.

[28] E. Pietriga and C. Appert, "Sigma lenses: focus-context transitions combining space, time and translucence," in *Proceedings of the 26th Annual CHI Conference on Human Factors in Computing Systems (CHI '08)*, pp. 1343–1352, ACM, April 2008.

[29] C. Tominski, J. Abello, F. Van Ham, and H. Schumann, "Fisheye tree views and lenses for graph visualization," in *Proceedings of the Information Visualization (IV '06)*, pp. 17–24, July 2006.

[30] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li, "Geometry-based edge clustering for graph visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1277–1284, 2008.

[31] N. Elmqvis, T. N. Do, H. Goodell, N. Henry, and J. D. Fekete, "ZAME: interactive large-scale graph visualization," in *Proceedings of the Pacific Visualization Symposium (PacificVis '08)*, pp. 215–222, March 2008.

[32] D. Holten, "Hierarchical edge bundles: visualization of adjacency relations in hierarchical data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 741–748, 2006.

[33] D. S. Kim, I. K. Hwang, and B. J. Park, "Representing the Voronoi diagram of a simple polygon using rational quadratic Bézier curves," *Computer-Aided Design*, vol. 27, no. 8, pp. 605–614, 1995.

[34] A. A. Ungar, "The hyperbolic pythagorean theorem in the Poincaré disc model of hyperbolic geometry," *American Mathematical Monthly*, vol. 106, no. 8, pp. 759–763, 1999.

[35] L. John and R. Ramana, "Laying out and visualizing large trees using a hyperbolic space," in *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology*, ACM, Marina del Rey, Calif, USA, 1994.

[36] J. W. Anderson, *Hyperbolic Geometry*, Springer, 2005.

[37] J. Lamping and R. Rao, "The hyperbolic browser: a focus+context technique for visualizing large hierarchies," in *Readings in Information Visualisation Using Vision To Think*, K. C. Stuart, D. M. Jock, and S. Ben, Eds., pp. 381–408, 1999.

[38] M. Jin, F. Luo, and X. Gu, "Cornputing surface hyperbolic structure and real projective structure," in *Proceedings of the ACM Symposium on Solid and Physical Modeling (SPM '06)*, pp. 105–116, June 2005.

[39] Y. Shavitt and T. Tankel, "Hyperbolic embedding of internet graph for distance estimation and overlay construction," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 25–36, 2008.

[40] J. Brusey and L. Padgham, "Techniques for obtaining robust, real-time, colour-based vision for robotics," in *RoboCup 1999*, vol. 1856 of *Lecture Notes in Artificial Intelligence*, Springer, 2000.

[41] I. D. Dinov, D. Rubin, W. Lorensen et al., "iTools: a framework for classification, categorization and integration of computational biology resources," *PLoS ONE*, vol. 3, no. 5, Article ID e2265, 2008.

[42] I. Dinov and N. Christou, "Web-based tools for modelling and analysis of multivariate data: California ozone pollution activity," *International Journal of Mathematical Education in Science and Technology*, vol. 42, no. 6, pp. 789–7829, 2011.

[43] N. Christou and I. D. Dinov, "Confidence interval based parameter estimation-a new SOCR applet and activity," *PLoS ONE*, vol. 6, no. 5, Article ID e19178, 2011.