A Study of Layout, Rendering, and Interaction Methods for Immersive Graph Visualization

Oh-Hyun Kwon, *Student Member, IEEE*, Chris Muelder, *Member, IEEE*, Kyungwon Lee, *Member, IEEE*, and Kwan-Liu Ma, *Fellow, IEEE*

Abstract—Information visualization has traditionally limited itself to 2D representations, primarily due to the prevalence of 2D displays and report formats. However, there has been a recent surge in popularity of consumer grade 3D displays and immersive head-mounted displays (HMDs). The ubiquity of such displays enables the possibility of immersive, stereoscopic visualization environments. While techniques that utilize such immersive environments have been explored extensively for spatial and scientific visualizations, contrastingly very little has been explored for information visualization. In this paper, we present our considerations of layout, rendering, and interaction methods for visualizing graphs in an immersive environment. We conducted a user study to evaluate our techniques compared to traditional 2D graph visualization. The results show that participants answered significantly faster with a fewer number of interactions using our techniques, especially for more difficult tasks. While the overall correctness rates are not significantly different, we found that participants gave significantly more correct answers using our techniques for larger graphs.

Index Terms—Graph visualization, virtual reality, immersive environments, head-mounted display

1 Introduction

One of the key freedoms of information visualization design is the opportunity to map arbitrary information to screen space at will, allowing for great control over how the data is presented. As most displays or representations are two-dimensional, most information visualization techniques limit themselves to two-dimensional mappings. That is, information visualization approaches traditionally eschew 3D techniques; 3D visualizations projected onto 2D displays often lead to occlusion and clutter issues. Better 2D representations generally avoid this problem. However, such a limitation is only applicable to 2D displays. The onset of ubiquitous, consumer-level stereoscopic displays has prompted questions of the potential effectiveness of 3D for information visualization techniques.

Whereas 3D displays used to be rare and cost-prohibitive, recent advances have made them ubiquitous enough that they can be assumed to be available for general visualization approaches. The most common type of these displays currently exists in the form of standard displays paired with eyewear that reveals different images per eye, which is either active (where an LCD over each eye alternates views in sync with the display) or passive (polarization of light differs per eye). However, even these displays limit the user's view to a small rectangular window encapsulated by the display area. While numerous information visualization techniques exist to visualize large data with limited display space, screen area still presents a concrete limit, particularly since the human eye is capable of utilizing a much larger field of view (FOV).

Larger displays such as powerwall displays or CAVE systems [1] aim to make use of the full range of human vision, and enable nigh unlimited immersive techniques, but these systems are often too expensive and bulky space-wise to build, as they require a large number of displays, and massive amounts of processing power to drive them all.

Head-mounted displays (HMDs) have recently become a popular alternative. In recent years, a number of HMDs are released or announced to be released including Oculus Rift [2], HTC Vive [3], Sony PlayStation VR [4], Google Cardboard [5], Samsung Gear VR [6], and Microsoft Hololens [7]. With these devices, a small, but high resolution display is placed directly in front of the user's eyes, such that each eye's view can be tightly controlled. Thus, it is unnecessary to show the entire scene; only the user's view needs to be rendered. Due to recent rapid improvements in features such as pixel density, high refresh rate, and low latency head tracking, HMDs have quickly become both feasible and affordable. Consumer-grade HMDs are quickly nearing a point where they can be treated as commodity hardware. These devices create immersive environments at a fraction of the cost of large display systems. The ubiquity of such devices enables a multitude of visualization possibilities.

Immersive scientific visualization techniques have been well explored, as 3D environments are ideally suited for 3D immersion. As such, scientific visualization has been a driving force behind the development of virtual reality systems. However, investigation into the applicability of such immersive environments for non-spatial data has been extremely sparse.

- *O.-H. Kwon is with the Department of Computer Science, University of California, Davis, Davis, CA 95616, and the Ajou University, Suwon, Korea. E-mail: kw@ucdavis.edu.*
- *C. Muelder and K.-L. Ma are with the Department of Computer Science, University of California, Davis, Davis, CA 95616. E-mail: cwmuelder@ucdavis.edu, ma@cs.ucdavis.edu.*
- *K. Lee is with the Department of Digital Media, Ajou University, Suwon, Korea. E-mail: kwlee@ajou.ac.kr.*

Manuscript received 28 July 2015; revised 14 Jan. 2016; accepted 17 Jan. 2016. Date of publication 21 Jan. 2016; date of current version 1 June 2016. Recommended for acceptance by T. Dwyer, S. Liu, G. Scheuermann, S. Takahashi, and Y. Wu. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TVCG.2016.2520921

1077-2626 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Even the concept of employing stereoscopy for non-spatial data has had limited exploration [8], [9]. In fact, previous research has found that stereoscopic representations are particularly helpful for graph visualization in certain tasks [10], [11], [12], [13], though these approaches limit themselves to traditional rectangular displays.

Following a preliminary study of ours [14] which introduced the layout and edge routing approaches that are the basis of this work, we are further concerned with the applicability of immersive virtual reality environments to graph visualization. In this paper, we present a novel study of layout, rendering, and interaction methods for immersive graph visualization. Specifically, the primary contributions of our work are:

- More complete description and discussion of the layout methods and the edge routing techniques compared to our preliminary study [14].
- Rendering techniques for enhancing clarity and highlighting techniques to better support interaction with the graph visualization.
- A user study for comparing different layout methods using a number of common graph exploration tasks in an immersive environment.

The results of the user study show that traditional 2D graph visualization are ill suited for immersive environments. Methods specifically designed for immersive graph visualization like ours are in need. In this manner, we have not only established an improved understanding of the effectiveness of the immersive graph visualization, but also posited general guidelines for future immersive visualization approaches.

2 Related Work

Virtual reality and stereoscopic techniques have a long history of being used in scientific visualization [15], [16], [17], applied to GIS data [18], volume data [19], and medical imaging data [20]. When the data has a 3D spatial attribute, such techniques are a natural fit.

The use of 3D in information visualization has been demonstrated for a long time [8], [21]. However, most existing 3D information visualizations are displayed on 2D displays with the monocular depth cues such as perspective, occlusions, motion parallax, and shading [22], [23]. Stereoscopic 3D displays provide one additional important depth cue: *binocular disparity*. Stereoscopy has been shown to be beneficial for some information visualization tasks [9]. Notably, stereoscopy has been shown in multiple user studies to be effective for graph visualization tasks [10], [11], [12], [13].

The earliest study on using stereoscopy for graph visualization was done by Ware and Franck [24]. They considered general low-level tasks for graph analysis such as identifying paths between two highlighted nodes, and found that stereoscopy with head tracking outperforms those in 2D condition. About ten years later, Ware and Mitchell [10], [11] conducted similar studies but in a higher resolution display. They found that participants showed less error rates with both motion parallax and stereoscopic depth cues, and stereoscopic viewings showed faster response times, regardless of the motion parallax. A more recent study done by Alper et al. [12] showed the effectiveness of stereoscopic highlighting techniques for 2D graph layout. While there was no significant difference between the stereoscopic highlighting and static visual highlight (color), participants performed better when both highlighting were used together. Halpin et al. [25] used an immersive system, but they still used a standard euclidean layout and employed the stereoscopy just for highlighting. Barahimi and Wismath [26] used HMD to view a 3D layout. In most of these studies, the graph is generally looked at from outside so that they often requires lots of viewpoint navigation and cluttered.

To keep the user's viewpoint equidistant to most of the display area, immersive environments can be modeled very naturally with a spherical or a cylindrical paradigm, with the user at the center. While there are lots of 3D graph layout approaches [27], few graph layouts work in a spherical space [28], [29], [30], [31], [32] because most existing graph layouts perform their calculations in the standard euclidean space. Kobourov and Wampler [32] introduced a force-directed method to calculate a graph layout in an arbitrary Riemannian geometry. Hyperbolic graph layout works in a spherical space [33], but nodes/edges would go through the user's view in the center of the sphere. Wu and Takatsuka [31] used a self-organizing map to place a small graph on a sphere, but this is based on multidimensional node properties. While these methods work in the spherical space, they are not designed to be looked at from the inside of the sphere.

In graph visualization, one challenge is how to effectively handle large number of edges. Edge bundling techniques [34], [35], [36] are common methods for routing edges to avoid clutter. While 3D layouts can eliminate many edge crossings by utilizing the additional dimension, conveying the depth of the edges presents an additional challenge. The studies of vector field visualization and diffuse tensor imaging techniques have explored this same problem. One basic technique is to illuminate the lines [37], [38]. To add the effect of global illumination, ambient occlusion may be used as done by LineAO [39]. It has also been shown adding halos can help enhance perception of relative depth between lines [40], [41], [42], though halos do not work for dense lines.

3 Methods

In an immersive environment, the visible scene is arranged omnidirectionally around the user's viewpoint. As much of the scene will be out of the user's view at any time, this naturally evokes exploration through head motion. Performing such navigation comfortably is often limited by the range of motion of the user's neck, which is often relatively fixed positionally, but flexible angularly (e.g., in a seated usage). Therefore, visibility of the scene from the perspective of the user's viewpoint is vital. However, traditional 3D graph layout often requires lots of spatial navigation in order for the user to find viewpoints that are good for perceiving depth and comprehending the structure of the graph [12]. To improve on this, it is important to find or create an ideal viewpoint that does not require the user to perform such heavy navigation [13]. Our approach targets this by using graph layouts on the surface of a sphere and by placing the user's viewpoint at the center of the sphere (Fig. 1), so that all nodes are equally visible to the user through angular

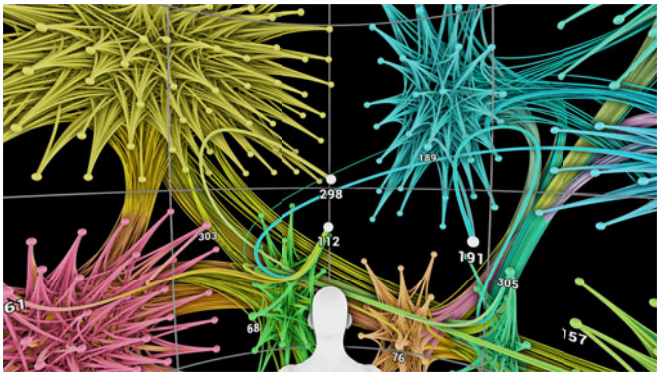


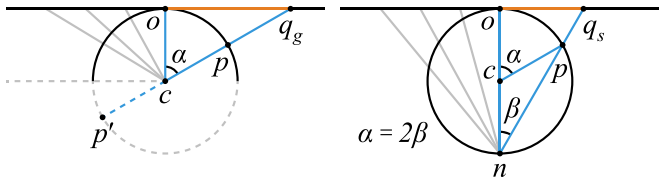
Fig. 1. The viewer is placed at the center of the sphere, on which the graph is laid out.

motion alone. In order to reduce occlusions, edges are routed outside the sphere in a bundled fashion (Fig. 2c). Lastly, as graph rendering involves rendering large numbers of lines, we utilized dense line rendering techniques from scientific visualization works (Fig. 2d).

3.1 Spherical Graph Layout

It is non-trivial to calculate the layout of a graph on the surface of a sphere as it is non-euclidean space. While several studies [28], [29], [30], [31], [32] introduced methods that can calculate a graph layout on the surface of a sphere, most existing graph layout algorithms are designed for 2D euclidean space. Thus, while dedicated spherical graph layouts are possible, this paper focuses on methods of effectively mapping existing 2D graph layout approaches to the surface of a sphere, in order to maximize the utilization of existing graph works.

While the surface of a sphere is also a 2D space, it is impossible to map a 2D euclidean geometry to the surface of a sphere without distortion. The field of cartography has explored map projections from a sphere to a plane fairly extensively [43]. To map a 2D graph layout to the surface of a sphere, we need a process that calculates the inverse of such map projections (i.e., that maps locations on a plane to



(a) Gnomonic projection geometry (b) Stereographic projection geomotry

Fig. 3. Spherical projection geometry. Linear distances in the plane are not linear on the surface of the sphere (gray lines).

locations on the surface of a sphere). Of the numerous kinds of map projections, we focus on gnomonic and stereographic projections, because they provide continuous mappings to arbitrary planes, and have well defined distortions that can be compensated for. We also used spherical coordinates and a cubed sphere as additional ways to map 2D layouts onto the sphere's surface.

3.1.1 *Gnomonic Projection*

In a gnomonic (or rectilinear) projection, a hemisphere is projected from its center radially outward onto a plane that is tangent to the hemisphere at a point. One advantage to this projection is that great circles are projected to straight lines on the plane, i.e., a straight line on the plane is a geodesic arc on the sphere. Gnomonic projection (Fig. 3a) projects a point p on the surface of a sphere from the center c of the sphere to point q_g on a tangent plane of a point o [46]. This projection can only be used to project one hemisphere at a time because it projects antipodal points p and p' to the same point q_g on the plane.

Let us consider a unit sphere in $\mathbb{R}^3$. There is no distortion at the tangent point o, but distortion increases the further a point is away from o. The geodesic arc $\widehat{op}$ on the sphere corresponds the line segment $\overline{oq_g}$ on the plane. By considering the points o, c, and q_g as a right triangle, the length of $\overline{oq_g}$ is $\tan\alpha$. The result of this is that in a naive, direct mapping the points from the plane to the surface of a sphere, the points further away from o become compacted on the

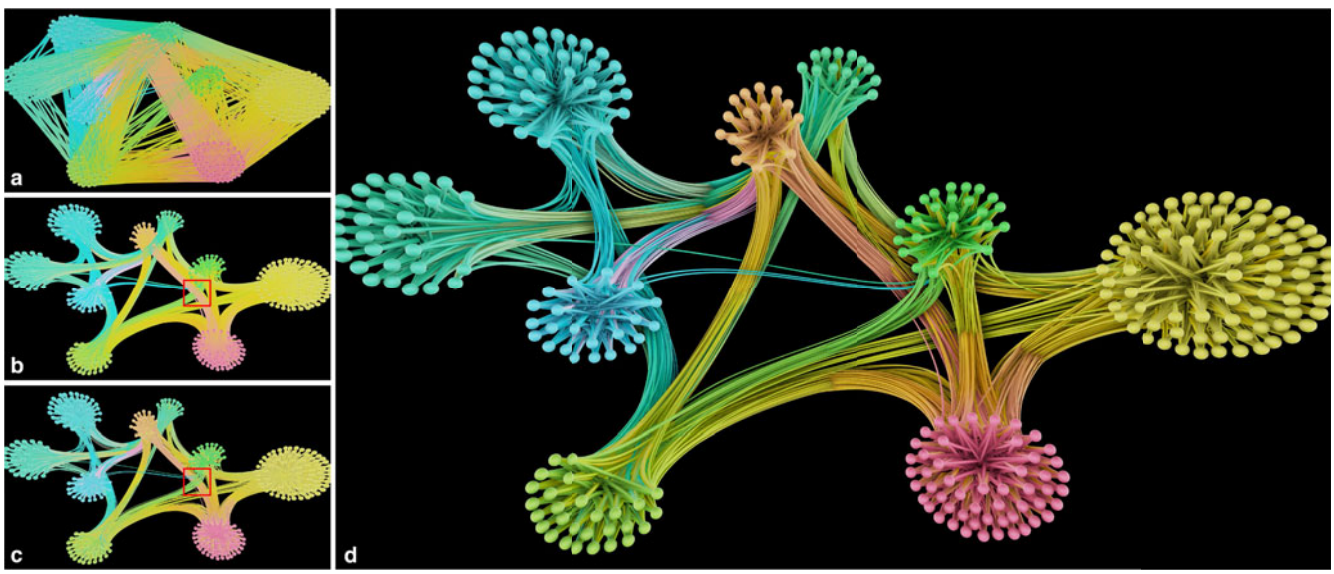
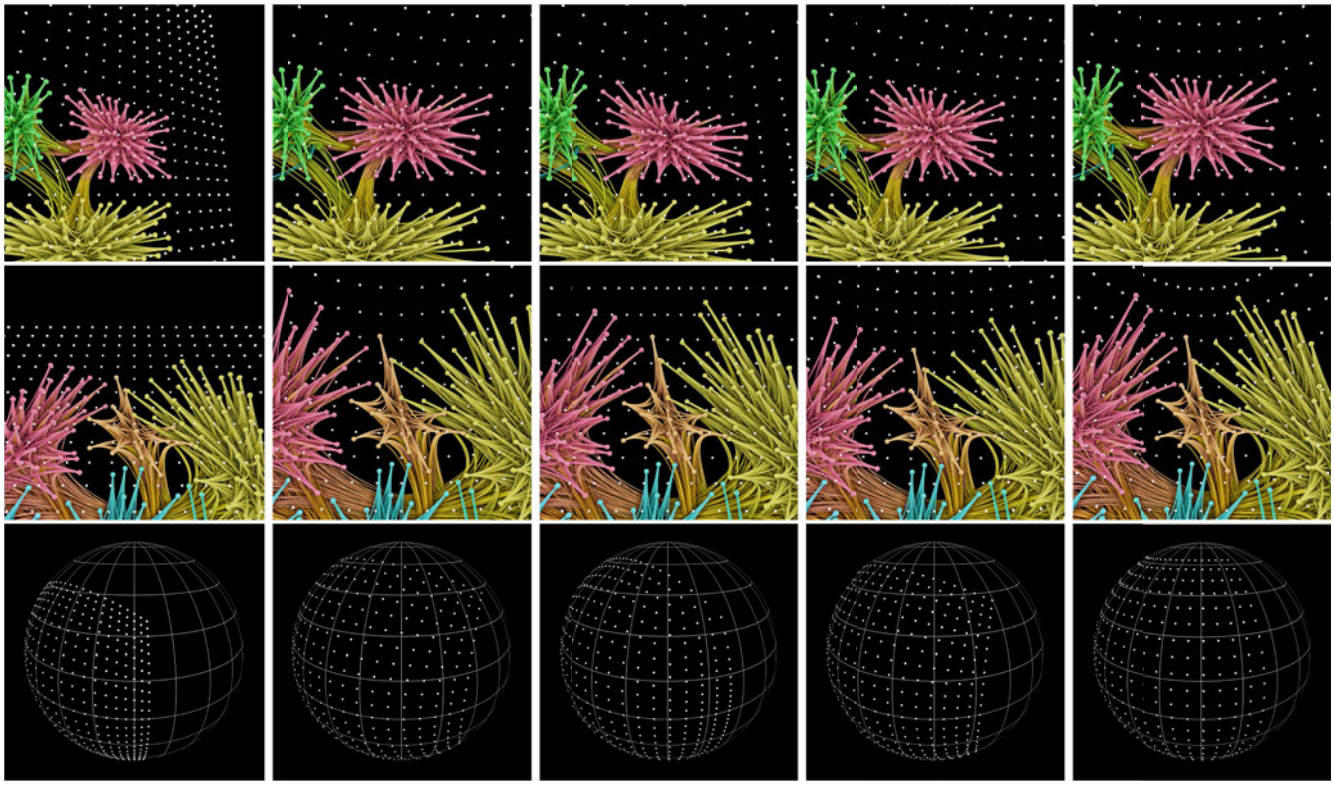


Fig. 2. Layout and rendering strategies. (a) place nodes on the surface of a sphere; (b) employing spherical edge bundling; (c) adding depth routing; (d) adding illumination. The detail views of the red rectangle can be found in Fig. 8.



(a) Direct Gnomonic Projection (b) Radially Compensated Gnomonic Projection (c) Independently Compensated Gnomonic Projection (d) Direct Stereographic Projection (e) Spherical Coordinates

Fig. 4. Mapping 2D graph layouts to the surface of a sphere. The upper two rows of images are what the user sees, and the bottom row of images show the mapped grid points on the sphere surface. The top row of images show the corner area of treemap-based layout [44]. The middle row of images show Gosper curve based layout [45] as an example of roughly circular layout.

sphere, as can be seen in Fig. 4a. Therefore, before mapping a point from the plane onto the sphere, we compensate for this by normalizing the point and then distorting them radially according to $q'_g = o + \tan\|\mathbf{u}\| \cdot \hat{\mathbf{u}}$, where $\mathbf{u} = q_g - o$. While this is good at preserving radial distances, one limitation of direct application of this mapping is that it does not preserve angles or straight lines. This becomes especially apparent with rectangular input layouts (such as a treemap-based layout [44]). This angular distortion can be seen in the corners in Fig. 4b. Also, the FOV available for the graph is limited because corners get wrapped much further around the sphere than the sides, so this mapping is limited by extrema in the layout, such as corners or outliers. Thus, this specific mapping works best with roughly circular layouts. If the layout is more circular and less dependent on straight-line boundaries (as in many force directed layouts or the Gosper curve based layout [45]), the angular preservation is less of an issue.

As a compromise, we also employ the option of warping the x and y dimensions independently as $x' = \tan x$ and $y' = \tan y$ before mapping, as demonstrated in Fig. 4c, In this manner, corners are not overly extended. Also, horizontal or vertical lines are preserved as perceptually linear arcs along the surface. However, distortions in area will increase as the FOV increases. Notably, the area of regions near the bounds of a rectangular area would shrink to zero as they near the hemisphere boundary, meaning this approach is only really useful for relatively narrow FOV. In practice, this approach is best suited to a limit of 120 degree FOV for the graph.

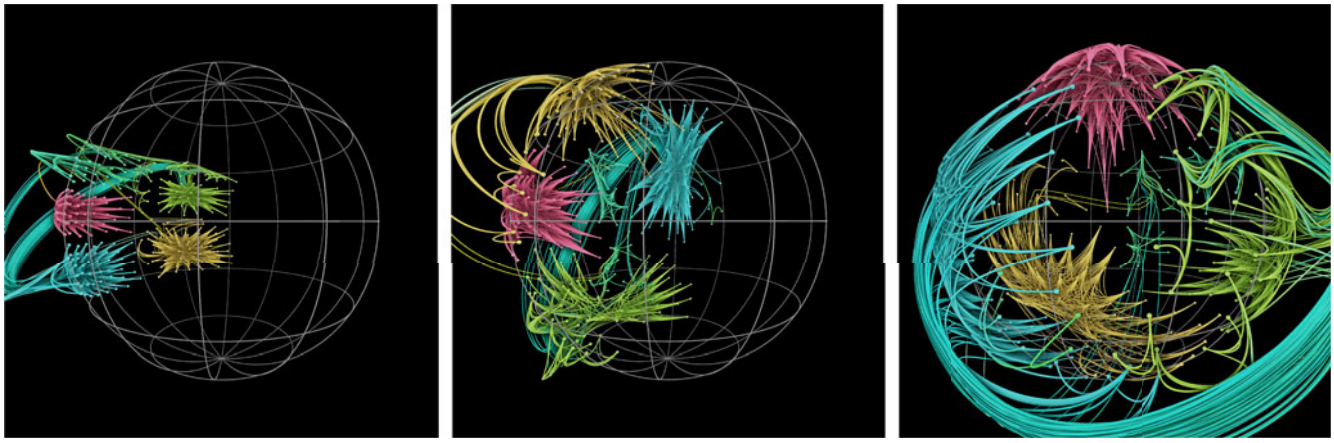
3.1.2 Stereographic Projection

Stereographic projection [46] projects a point p on the surface of a sphere to the point q_s on the plane tangent to the sphere at point o along a ray originating from the antipodal point n opposite o, as illustrated in Fig. 3b. Similar to gnomonic projection, stereographic projection offers very predictable and analyzable distortion. Unlike gnomonic projection, stereographic projection is good at preserving angular properties. Notably, circles on the sphere that do not pass through the point n are projected to circles on the plane and vice-versa.

As in gnomonic projection, regularly spaced points in a euclidean space would be skewed when projected to spherical space, as can be seen in Fig. 4d. Unsurprisingly, this follows the same tangential law as gnomonic projection, so the same correctional options are applicable. However, one big difference between gnomonic projection and stereographic projection is that the angle β in stereographic projection is half of the angle α in gnomonic projection. Thus, we can either warp the plane radially by $q'_s = o + \tan\frac{\|\mathbf{v}\|}{2} \cdot \hat{\mathbf{v}}$, where $\mathbf{v} = q_s - o$, or warp the x and y dimensions independently as $x' = \tan\frac{x}{2}$ and $y' = \tan\frac{y}{2}$.

3.1.3 Spherical Coordinates

Another straightforward method is mapping 2D coordinates (x, y) to spherical coordinates (ϕ, θ). The major difference between this and the projection-based approaches is that in spherical coordinates, all points along the equator are undistorted, but the distortion increases toward the



(a) 120° ×67.5° FOV (b) 180° FOV (c) 360° FOV

Fig. 5. Spherical graph layouts: 2D layouts can be mapped to the sphere with some amount of distortion. Preserving angles with independent axis corrective mapping (a) is appropriate for rigid, rectangular structures, but is limited in FOV. Radial corrective mapping (b) works well for roughly circular layouts on a hemisphere. For full immersion (c), we use a space filling curve defined on a cubed sphere to cover the entire surface.

poles. As shown in Fig. 4e, points near the poles are greatly compacted. Unlike in previous approaches, this distortion is not as simple to account for. However, by limiting the range of the vertical angular axis, the consequences of this distortion can be minimized. While this limits the vertical range of space that is utilized, it still allows the use of up to the full 360 degree horizontal range.

3.1.4 Cubed Sphere Mapping

For a completely immersive layout, the nodes should be distributed roughly evenly in all directions, not just on the front hemisphere. One of the best ways we found to do this was to employ a cubed sphere, where a graph layout is calculated onto a cube, and then each face of the cube is mapped onto the sphere. As the gnomonic projection, the grid cells of each cube face are warped with a tangent function. Because of the rigid boundaries, it is imperative to preserve straight lines, as the interfaces between the faces should be contiguous. Thus, we warp the x and y dimensions independently forming what is commonly referred to as an equiangular cubed sphere [47].

The space-filling curve layout then maps very nicely to this approach, as a space-filling curve can be defined on the surface of a cube as six planar space-filling curves, one each face, that are still contiguous as one single curve [47]. To compute this layout, we break the graph into 6 sections, lay out the nodes for each section in a separate plane, warp their x and y values according to a tangent function, and finally map each plane onto the appropriate face of the cubed sphere. This ensures an even distribution of nodes over the entire surface, while preserving cluster locality.

3.2 Field of View Variation

The one thing all the spherical mappings have in common is that there is flexibility in the range of the sphere to utilize. That is, the layout algorithm has control of how much FOV to utilize for the graph.

One important aspect to consider is the range of motion of the user's neck (cervical spine). Particularly when the users seated, they often look forward, and to the left and right without moving their torso. Also, the visual density of a graph should be considered to determine appropriate FOV for the graph. Too wide of an FOV can be overly strenuous to the user, and can hide parts of the graph in regions the user might not even see. Conversely, too narrow FOV can be overly cluttered and less immersive. So it is generally better to limit the FOV of the layout to be within the average range of motion of neck, unless the graph is complex enough to benefit from wide FOV.

There are several criteria to measure this range. We used what is referred to as the active range of motion (the range of movement through which a person can actively move the joint without any assistance). Several studies [48], [49] show that the vertical range of the motion of is roughly 50 degree upwards (extension) and 60 degree downwards (flexion), and the horizontal range left and right are around 80 degree each. Thus, the maximum range to limit ourselves to should be around 160 degree horizontal by 100–110 degree vertical. We confirmed this during our pilot study, where we found that the users were not comfortable near or over these limits. So, in our evaluation of the effect of FOV within these limits we used layouts at three FOVs: with horizontal FOVs of 150, 120, and 90 degree, each with a 16×9 aspect ratio.

3.3 Edge Bundling

Edge bundling techniques [34], [35], [36] are frequently used to improve the legibility of dense or complicated node-link diagrams. The main difference between these techniques lies in how the curves of the edges are calculated. These techniques use curves such as Bezier curves, B-splines, or Catmull-Rom splines to route edges along control points. We apply a variant of hierarchical edge bundling [34] to our immersive graph visualization.

Direct application of edge bundling to a spherical layout using computations in 3D euclidean space would not work well for our purposes, as the bundles would run through the inside of the sphere close to the viewer, obscuring the nodes of the graph. Rather, just like the layout of the nodes, we route the edges around the surface of the sphere instead. In addition, we also route edges away from the outside of the sphere according to the clustering hierarchy, with the edges that has longer path along the clustering hierarchy routed further away from the sphere than short ones, in order to improve visibility when viewed from the inside of the sphere.

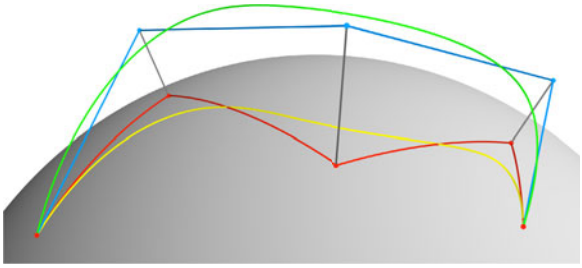


Fig. 6. Hierarchical edge bundling routes edges with splines that follow the clustering hierarchy. Each control point corresponds with a cluster node in the hierarchy. We compute the edge spline in two stages. A spherical spline (yellow) is calculated with a spherical B-spline according to the control points on the surface (red). Then, we calculate the 1D depth spline according to the height of the corresponding cluster nodes in the clustering hierarchy (blue). Finally, the spherical spline is extended radially (green) by applying the depth spline.

3.3.1 Spherical Edge Bundling

The first step is to route the edge on the surface of the sphere. That is, given the start point, end point, and a set of intermediate control points, all on the surface of the sphere, we want to compute a cubic B-spline that also lies on the surface. Most spline calculation algorithms are defined using linear interpolation in euclidean space, such as de Boor's algorithm [50]. In the euclidean space, a B-spline is calculated using de Boor's algorithm as:

$$p_i^j(t) = \begin{cases} \mathrm{LERP}(p_{i-1}^{j-1}(t), p_i^{j-1}(t), r_i^j) & \text{if } j > 0 \\ p_i & \text{if } j = 0 \end{cases} \quad \text{where } r_i^j = \frac{t - t_i}{t_{i+k-j} - t_i}, \quad \mathrm{LERP}(p_0, p_1, t) = (1-t)p_0 + tp_1. \tag{1}$$

However, linear interpolation in euclidean space would fail to compute correctly on the surface of a sphere, as would linear interpolation in spherical coordinates, as the surface of a sphere is non-euclidean. In this work, we used a modification of de Boor's algorithm that use *s*pherical *l*inear int*erp*olation (SLERP [51]) instead of euclidean linear interpolation to calculate splines on the surface of a sphere, which is defined as:

$$\mathrm{SLERP}(p_0, p_1, t) = \frac{\sin(1-t)\theta}{\sin\theta} p_0 + \frac{\sin t\theta}{\sin\theta} p_1, \quad \text{where } \theta = \arccos(p_0 \cdot p_1). \tag{2}$$

Replacing the euclidean linear interpolation with the spherical linear interpolation in de Boor's algorithm yields a spherical B-spline:

$$p_i^j(t) = \begin{cases} \mathrm{SLERP}(p_{i-1}^{j-1}(t), p_i^{j-1}(t), r_i^j) & \text{if } j > 0 \\ p_i & \text{if } j = 0 \end{cases} \quad \text{where } r_i^j = \frac{t - t_i}{t_{i+k-j} - t_i}. \tag{3}$$

In this manner, the spline is smooth from the perspective of the user in the center of the sphere. While several studies [52], [53] have introduced more complex methods, we found de Boor's algorithm with SLERP to be sufficient, and it is relatively simple and computationally efficient enough to recompute upon interaction. An example of such a spline is shown as the yellow curve in Fig. 6, which is computed according to the red control points.

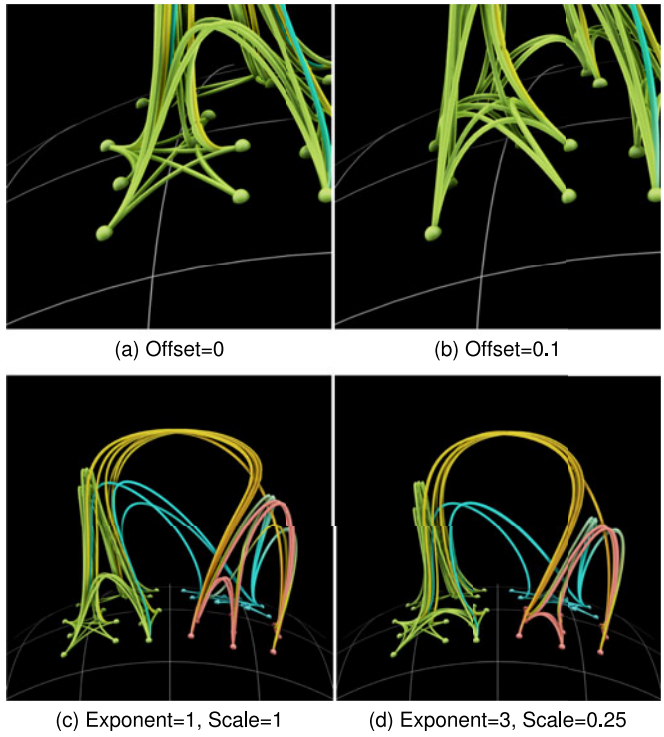


Fig. 7. Depth routing parameters. By varying the offset, the scale value, and the exponent value, we can fine-tune the edge bundling to improve the clarity of the visualization.

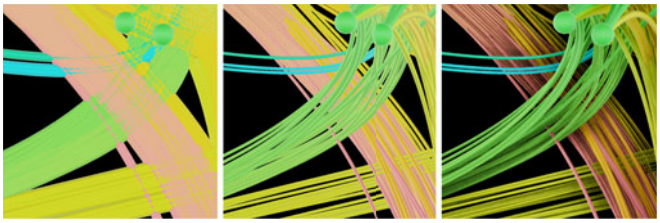
3.3.2 Depth Routing

Bundling edges using spherical splines improves the legibility of the spherical graph layout equivalent to edge bundling in traditional 2D layouts. But it places all edges at the same distance from the user's viewpoint, which not only causes edge crossings which could be avoidable, but also fails to take advantage of depth perception capabilities of stereoscopy.

So in addition to the spherical spline computation, we also calculate a 1D depth spline to route the edges depthwise by raising the spline off the surface of the sphere by modulating the distance of the sample points with the samples of the depth spline. Each control point corresponds with a node in the clustering hierarchy, so we calculate the depth spline according to the height of the corresponding cluster node (i.e., distance in the cluster hierarchy from the cluster node to its deepest leaf node).

We calculate the distance from the center of the sphere to a each control point of the depth spline u in the clustering hierarchy as $d_u = r(1 + o + s \cdot h_u^p)$, where r is the radius of the sphere, o is an offset value, s is an scale value, p is an exponent value, and h_u is the height of the node u in the hierarchy. The offset value o (Figs. 7a and 7b) determines how much deep to route edges between nodes within the same cluster. Figs. 7c and 7d shows the effect of the exponent value p. By varying the offset value o, the scale value s and the exponent value p, we can fine-tune the edge bundling to improve the clarity of the visualization.

In this manner, an edge is routed up the clustering hierarchy, moving further away from the user's viewpoint up



(a) without depth routing (b) with depth routing (c) with depth routing and illumination

Fig. 8. Depth routing and rendering techniques.

to the least common ancestor cluster, and then back toward the user as it traverses down the other side of the clustering hierarchy. The result of this is that the longer, inter-cluster bundles are also the furthest away from the user, while shorter, more intricate structures remain close to the viewpoint. While closer edges occlude other edges from the user's viewpoint, the edges with the depth routing (Fig. 8b) are longer, and bundled, making them easier to perceive. Also, the legibility is improved even further with illumination techniques, as shown in Fig. 8c.

3.4 Rendering Techniques

Rendering dense bundles of lines has been well studied for scientific visualization, such as diffusion tensor imaging or streamline visualization. As such, there are numerous techniques available that could be applied to graph visualization, such as illuminated lines [37], [38], halos [40], [41], or LineAO [39].

LineAO [39] is particularly effective since it is both utile and efficient. It emphasizes both local structure and global structure using multiple hemispheres of different radius for ambient occlusion sampling. To enable maximum frame rate of our implementation, we simplified LineAO to using two hemispheres of two radius (local and global). Fig. 8 shows our simplified LineAO result.

Standard line rendering produces unintuitive perceptual cues at close distances. Specifically, any strand-like object in the real world will appear larger at close distances and smaller when far away. In order to convey this, simple line rendering is not nearly sufficient. As such, we have to employ tube rendering techniques, so that the width and shading will correspond correctly with depth. In order to maintain efficiency, pumping large amounts of geometry through the rendering pipeline would be counterproductive. So instead, we utilize geometry shaders to expand lines into depth-dependent quads, and use pixel shaders (or fragment shaders) to compute the appropriate normals. For distant (sub-pixel) edges, standard line rendering is still used both to increase the performance, as lines are simpler to compute, and to avoid sub-pixel rasterization issues, such as culling the polygons of the edges entirely.

While a graph visualization should benefit from any number of these techniques, there are critical performance limitations to consider. Current HMD rendering techniques require to render two views separately, one for each eye, which leads to significant performance drop. For a HMD, high frame rate and low latency tracking system are both of critical importance in order to avoid inducing nausea in the user. As such, it is imperative for a visualization to run with frame rate as close to the display's refresh rate limit as possible. The current generation of Oculus Rift, (DK2), runs at 75 Hz, and future HMDs will run at 90 Hz or higher. To get a high frame rate while still attaining good rendering quality, we used Unreal Engine 4 [54] to employ real time rendering technologies which have been extensively optimized by the game industry.

3.5 Interaction Techniques

Interfaces within HMD environments is an area of many open challenges and opportunities. Since the user loses direct sight of their hands from within the device, traditional mouse and keyboard interaction is limited. While hand tracking devices such as Leap Motion [55] are gaining popularity in conjunction with virtual reality environments, we found that the currently available generation of the hand tracking devices were not well suited to the tasks of our user study, and users are not as familiar with them as they are with traditional input devices. Thus, we still wanted to use a basic cursor paradigm, as most users are familiar with mouse interaction. In all cases, the 3D selection follows a ray from the center of the view through the cursor.

We considered three methods of controlling a cursor within a HMD environment:

- Cursor follows center of the user's view, without mouse control
- Cursor follows relative to the user's view, with mouse control
- Cursor does not follow the user's view (stay in the world), mouse control only

The first (and simplest) of these options is to lock the cursor to the center of the user's view. Then, the cursor is moved entirely via head tracking. This is the simplest method since it eliminates the need for additional mouse control, other than simple button interactions. Also, the cursor will always be in the user's view. However, there is no freedom to change the cursor position while maintaining a particular view, and precision requires precise neck motion which can tire the user.

The second method option is to still have the cursor track with the user's view, but to use the mouse to move the cursor within the view. This offers the advantages of keeping the cursor always in the user's view, while allowing the user to fine tune the mouse with a traditional mouse input. However, in our pilot study the users found this method to be confusing. When having the cursor's position depend on both head motion and mouse movement, the cursor to move a lot, often in unintuitive ways. Also, when user changes their view while the cursor is periphery of the view, many unexpected and distracting interactions occur.

The last option is to control the cursor via the mouse relative to the fixed spherical space, regardless of the user's view direction. This is the most similar to the 2D displays that users are used to, because in traditional desktop environments, the cursor keeps its position on the 2D screen space even when user looks at different portions of the screen. However, the downside to this approach is that the cursor can leave or be left outside of the user's view. To counter this, when the cursor is outside of view frustum, we show an arrow that pointing cursor's position, and we added a key shortcut to reset the cursor's position to the center of the user's view. Our pilot study found this

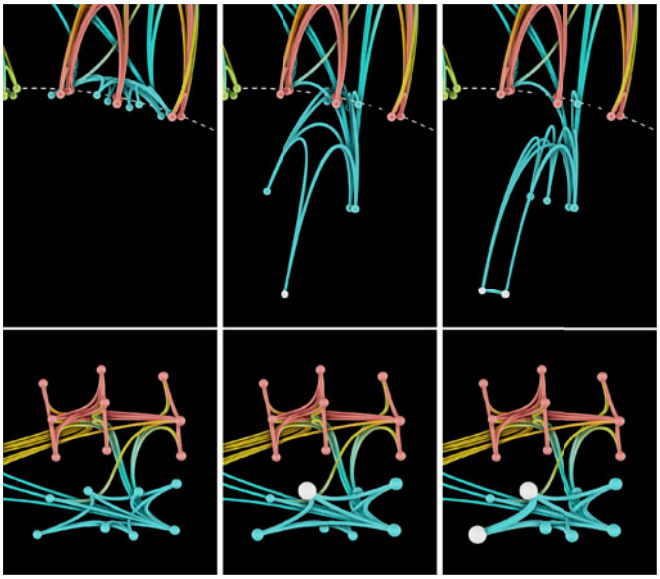


Fig. 9. Highlighting technique; the upper row of figures are external views for illustration and the lower row of figures are user's views. (a) Before highlighting, nodes are laid out on the sphere's surface. (b) On highlighting, a node and its neighbors are brought closer to the user, with the highlighted node closer than its neighbors. (c) If two nodes are highlighted, an edge between them is also brought in to the closest focal depth.

approach to be the most effective, so we used this option for our user studies.

3.6 Interactive Highlighting

The cursor provides a method of interacting with the graph, but there is also flexibility in how to represent interactions such as inspection, highlight, or selection. We utilize a modified variant of an existing stereoscopic highlighting technique [12] combined with our depth routing techniques and smooth animated transition to better utilize stereoscopic depth cues.

As shown in Fig. 9a, the edges are routed outside of the sphere before highlighting. When a node is highlighted (Fig. 9b), the highlighted node is moved closer to the user's viewpoint which is the center of the sphere. Also, its adjacent nodes are brought closer to the user too, but no more than the highlighted node. The edges that have a highlighted node are also brought closer to differentiate them from other edges. To do this, we transform the depth spline for edge routing of the highlighted edges to be in between the default layout sphere and the focal sphere to bring the edges closer naturally, reducing occlusion or edge crossings for the highlighted edge. The overall effect is that the highlighted nodes/edges are easily distinguished from the remainder of the graph, and the immediate connections are easy to visually follow to their destinations. To further accentuate this, we can apply a halo technique to the highlighted edges.

Multiple nodes/edges can be highlighted and brought into the focal sphere, allowing the user to explore any potentially interesting paths or relations that they find. To emphasize the edges between the highlighted nodes, we remove the depth routing and set the depth to be the same as the highlighted nodes, so that the entire edge spline is on the focal sphere (Fig. 9c). We implemented these depth routing changes with smooth animated transitions for user can keep the track of nodes and edges.

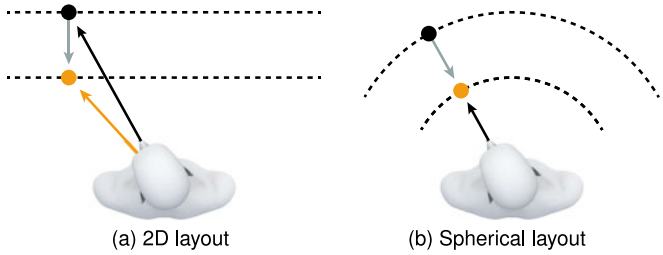


Fig. 10. Stereoscopic highlighting technique [12] on 2D graph layout (a) and spherical graph layout (b).

We brought a node closer to the user when they are highlighted it (see Fig. 10). In spherical layouts (Fig. 10b), the direction of 'getting closer' is same as the direction from the node to the viewpoint. Thus, any other interactions on the highlighted node can be performed without moving the cursor again. In a 2D layout (Fig. 10a), however, if you highlight a node then the position of the node changes in the user's view. So the user needs to move the cursor each time they re-interact with the highlighted node. In fact, the highlighted node in a 2D layout can move to the outside of the user's view.

4 User Study

The main purpose of our user study is to evaluate the use of spherical graph layouts and depth routing techniques in immersive graph visualization. The findings from a preliminary pilot study helped set up this study. For example, we found that for many participants, some tasks were either too simple or difficult preventing us from obtaining meaningful results. We also learned that cursor movement for selection should follow mouse movement alone and not depend on head movement. Furthermore, the preliminary study helped find how best to properly label the graph during viewing. These findings enabled us to remove unwanted factors that would impact the performance of the participants in the new study. The pilot study also helped us narrow the scope of the new study; we dropped the comparison of different displays [56] and focused on the specifics of using a HMD.

4.1 Experiment Design

For our study, we designed a within-subjects experiment: 3 *visualization conditions* × 3 *graph sizes* × 4 *tasks*. We evaluated three dependent variables in the study: *task completion time*, *correctness rate*, and *number of interactions*. *Task completion time* does not include the time to read the task description. *Correctness rate* is the percentage of tasks correctly answered. *Number of interactions* counts the number of pointing (mouse over on a node), number of highlighting (left click on a node), and number of selecting (right click on a node).

4.1.1 Visualization Conditions

We considered three visualization conditions:

C1: 2D graph layout. The graph is laid out on the plane.
C2: Spherical graph layout without depth routing. The graph is laid out on the surface of a sphere.

C3: Spherical graph layout with depth routing. The graph is laid out on the surface of a sphere.

All conditions used our modified stereoscopic highlighting [12], and the same rendering techniques (Section 3.4). All participants used a HMD, which is an Oculus Rift DK2. Graph labels always face toward user's viewpoint with the same size.

With the 2D graph layout, we decided not to use depth routing for several reasons. First, depth routed edges are skewed (or narrowed) toward the view direction as a result of perspective distortion. This can be resolved by calculating the depth routing in view space, but if the view changes then the shape of depth routed edges is also changed, which introduces more confusion to the users. Also, the shape of depth routed edges seen by the participant would not be natural if the view direction is not perpendicular to the 2D graph. If we used orthographic projection to render, the participant would lose the depth perception.

In conditions without depth routing (C1 and C2), we assigned the control points of the 1D depth spline for edge routing (i.e., clusters along a path) by linear interpolation of the depth of the two nodes of the edge.

4.1.2 Tasks

We used four tasks in the study:

T1: Find common neighbors. *Select all nodes that are common neighbors of two given nodes, not just one or the other.* The participant can find common neighbors by first highlighting the two given nodes, then look at which nodes have edges connected to the given nodes.

T2: Find the highest degree node. *Among four given nodes, select the node with the highest degree (most neighbors).* User can count the neighbors of each given node by highlighting it.

T3: Find a path. *Find an ordering of a given set of nodes which forms a path along edges from node A to node B, visiting each node once.* The participant was given a start node, an end node, and three other labeled nodes, and instructed that there exists a path that goes through them. The participant was then asked to find the order of the nodes in this path.

T4: Recall node locations. *Find the start node and the end node used in the previous task*. To compare the spatial memory in different visualization conditions, we asked users to remember the locations of the start node and the end node given in T3.

The labels of given nodes are always shown during the tasks, the labels of other nodes are hidden unless they are pointed (mouse over), highlighted (left click), or selected (right click).

4.1.3 Graph Size and Field of View

We used three different graphs with different data sizes (i.e., the number of nodes and number of edges) and spatial sizes (FOVs) for the experiment. One additional graph was used for the training session. To keep the visual density of the graph visualizations, we assigned the FOV of a graph according to its size.

It is not necessary to follow the aspect ratio of the HMD for the aspect ratio of the graph layout because the virtual world has infinite space; however, we used 16×9 aspect ratio for all graph layouts according to the estimated range of motion of the participant as discussed in Section 3.2.

The graph visualization was located at 10 meter away from the participant's viewpoint in the virtual world. Naturally, the default view direction is the direction from viewpoint to the center of the graph layout. Also, the 2D graph layout (C1) is perpendicular to the default view direction. The distance from the viewpoint to a node is equidistant in a spherical graph layout (C2 and C3), and is closest at the center of a 2D graph layout while the distance increases for nodes away from the center. The geodesic width of a spherical graph layout (C2 and C3) is $r\theta$ and the width of a 2D graph layout (C1) is $2 \cdot tan(\frac{\theta}{2})$, where θ is the horizontal FOV of the graph.

The four datasets are:

D0: This graph [57] consists of 34 nodes and 78 edges. The FOV of this graph is $60^\circ \times 33.75^\circ$. This graph was used in the training session.

D1: The *small* graph [58] consists of 77 nodes and 254 edges. The FOV of this graph is $90^\circ \times 50.625^\circ$.

D2: The *medium* graph [59] consists of 116 nodes and 615 edges. The FOV of this graph is $120^\circ \times 67.5^\circ$.

D3: The *large* graph [60] consists of 297 nodes and 2359 edges. The FOV of this graph is $150^\circ \times 84.375^\circ$.

4.2 Participants

We recruited 21 (12 males and 9 females) participants in our user study. The age of the participants ranged from 21 to 34, with the mean age of 25.71 years (SD = 3.96). The group consists of 11 undergraduate students and 10 graduate students. Each participant completed the experiment in about 60 minutes including initial setup, training session, and questionnaires.

Nine participants have normal vision. Ten participants wear eye glasses. Two participants wear contact lenses. All participants are not color-blind. We adjusted HMD for each participant. The participants wearing glasses used HMD without glasses. However, they had no other vision condition than nearsightedness which can be resolved by adjusting focal distance of HMD. None of the reported differences in vision was statistically significant.

Only one participant had previous experience with HMD. Nineteen participants had previous experience with stereoscopic viewing from television or movie theater. Fourteen participants indicated that they had seen a graph visualization (e.g., a node-link diagram) before, and knew what it is, but had never used it. Seven participants had no experience with graph visualization.

4.3 Apparatus and Implementation

For a fair comparison of different FOVs for graph visualization, it is necessary to maximize the use of the given area for graph layout. Therefore, the graph layout was calculated using a treemap-based approach [44] in all visualization conditions. To avoid learning effects between tasks, we randomized the graph layout by randomly reordering each level of the hierarchical clustering and randomly orienting intermediate levels of the treemap by multiples of 90 degree. Nodes were labelled with a three-digit unique

random number to remove any possibility of the participants answering the questions from meaningful node labels. The participants used a standard computer mouse as an input device with the spherical cursor technique described in Section 3.5.

The participants were seated in front of a desk and used Oculus Rift DK2 [2]. The Oculus Rift DK2 has a 1920 × 1080 px (split to 960 × 1080 px per eye) OLED panel with a 75 Hz refresh rate. NVIDIA GTX 980 graphics card was used to render the visualization. The rendering was maintained at about 75 frames per second. The positional tracking was enabled only in C1 because the spherical graph layouts (C2 and C3) do not require the positional tracking.

4.4 Procedure

Prior to the beginning of the experiment, we informed the participants of possible issues (e.g., eyestrain, disorientation, or sickness) and right to exit the experiment at any time. The participants had no time limit for the tasks.

Before each experiment, we adjusted the HMD (for the distance between the pupils of each eye and the distance between HMD lens and the cornea) for each participant. We showed a demo scene about 3 minutes to the participants to identify any issue (e.g., lack of depth perception, eyestrain, disorientation, or sickness) with the HMD. None of the participants had any issue with the HMD.

4.4.1 Training

We first instructed the participants about how to use three interaction techniques, pointing (mouse over), highlighting (left click), and selecting (right click). The participants was asked to point, to highlight/dehighlight, to select/deselect specific nodes that the experimenter asked to do so. Each participant first conducted 12 training trials (3 *visualization conditions* × 4 *tasks*) using D0 to be familiar with the experimental setup and procedure. We then instructed the participants about the strategies for finding a correct answer for each task. During the training, we indicated to the participant whether his/her answer was correct.

4.4.2 Experiment

The tasks were presented to the participants in the same repeated order from T1 to T4. The order of *visualization conditions* × *datasets (graph sizes)* combinations was counterbalanced.

The participant was allowed to rest and could continue to the next trial when ready. Before each trial, we reset the participant's head position and view direction in both physical world and virtual world to remove effect from bad calibrations. The participant read the task description without the graph visualization. After the participant understood the task, then the graph visualization appeared and the task description became hidden to maximize available display area. The task descriptions were shown again when the participant requested it. The participant answered the question by selecting nodes. The participant was asked to talk loud for proceeding to the next step rather than through a GUI to avoid uncontrolled effect from the GUI. During experiment, we did not give any indication to the participant on whether an answer was correct or not.

After all trials, the participant was asked to comment on the advantages and disadvantages of the visualization conditions and point out any specific features that he/she liked or disliked in experiment.

4.5 Hypotheses

Based on the setting of our user study, we expected to obtain three main results:

- H1: For all tasks, spherical graph layouts (C2 and C3) would outperform 2D graph layouts (C1) in immersive environments.
- H2: Spherical graph layouts with depth routing (C3) would outperform a spherical graph layout without depth routing (C2).

4.6 Results

Overall, C2 and C3 often outperform C1, and are never outperformed substantially by C1, confirming H1. Similarly, C3 often outperforms C2, as in H2.

4.6.1 Task Completion Time

On average, each *task* took 58.39s (*SD* = 77.86) to complete.

A repeated measures ANOVA with a Greenhouse-Geisser correction (ε = .691) shows a significant effect of *visualization condition* on *task completion time* ($F_{1.38,27.63}$ = 16.05, $p < .001$). Average *task completion times* (Fig. 11a) are 75.77s for C1 (*SD* = 103.19), 56.22s for C2 (*SD* = 67.02), and 43.20s for C3 (*SD* = 50.55). Post-hoc tests using the Bonferroni correction indicate that C3 is significantly faster than both C1 ($p < .0001$) and C2 ($p < .05$), and that C2 is also significantly faster than C1 ($p < .05$).

We found a significant effect of *task* on *task completion time* ($F_{3,60}$ = 10.23, $p < .001$). Unsurprisingly, the participants required more time to answer for more complex tasks. Average *task completion times* (Fig. 11b) are 53.06 s for T1 (*SD* = 80.06), 44.86 s for T2 (*SD* = 53.89), 76.89 s for T3 (*SD* = 91.75), and 58.77s for T4 (*SD* = 77.85).

When we analyzed the results for each *task* separately, there are significant effects of *visualization condition* on *task completion time* for T1 ($F_{1.21,24.15}$ = 6.01, $p < .05$, ε = 0.604, using Greenhouse-Geisser correction) and T3 ($F_{2,40}$ = 12.68, $p < .0001$), but not T2 or T4. For T1, both C3 ($p < .01$) and C2 ($p < .05$) are significantly faster than C1. For T3, C3 is significantly faster than both C1 ($p < .001$) and C2 ($p < .05$), and C2 is also significantly faster than C1 ($p < .05$).

Our analysis shows a significant effect of *dataset (graph size)* on *task completion time* ($F_{2,40}$ = 3.45, $p < .05$). As would be expected, the participants required more time to answer for larger graphs. Average *task completion times* (Fig. 11c) are 52.30 s for D1 (*SD* = 80.60), 56.54 s for D2 (*SD* = 71.75), and 66.34 s for D3 (*SD* = 80.54).

4.6.2 Correctness Rate

On average, 87.70 percent of the answers were correct (*SD* = 17.02) for all 21 participants.

While we did not observe a significant effect of *visualization condition* on *correctness rate* ($F_{2,40}$ = 2.08, p = 0.14), C3 slightly outperforms the others about 3.5 percent on average. Average *correctness rate* are 86.51 percent for C1, 86.51

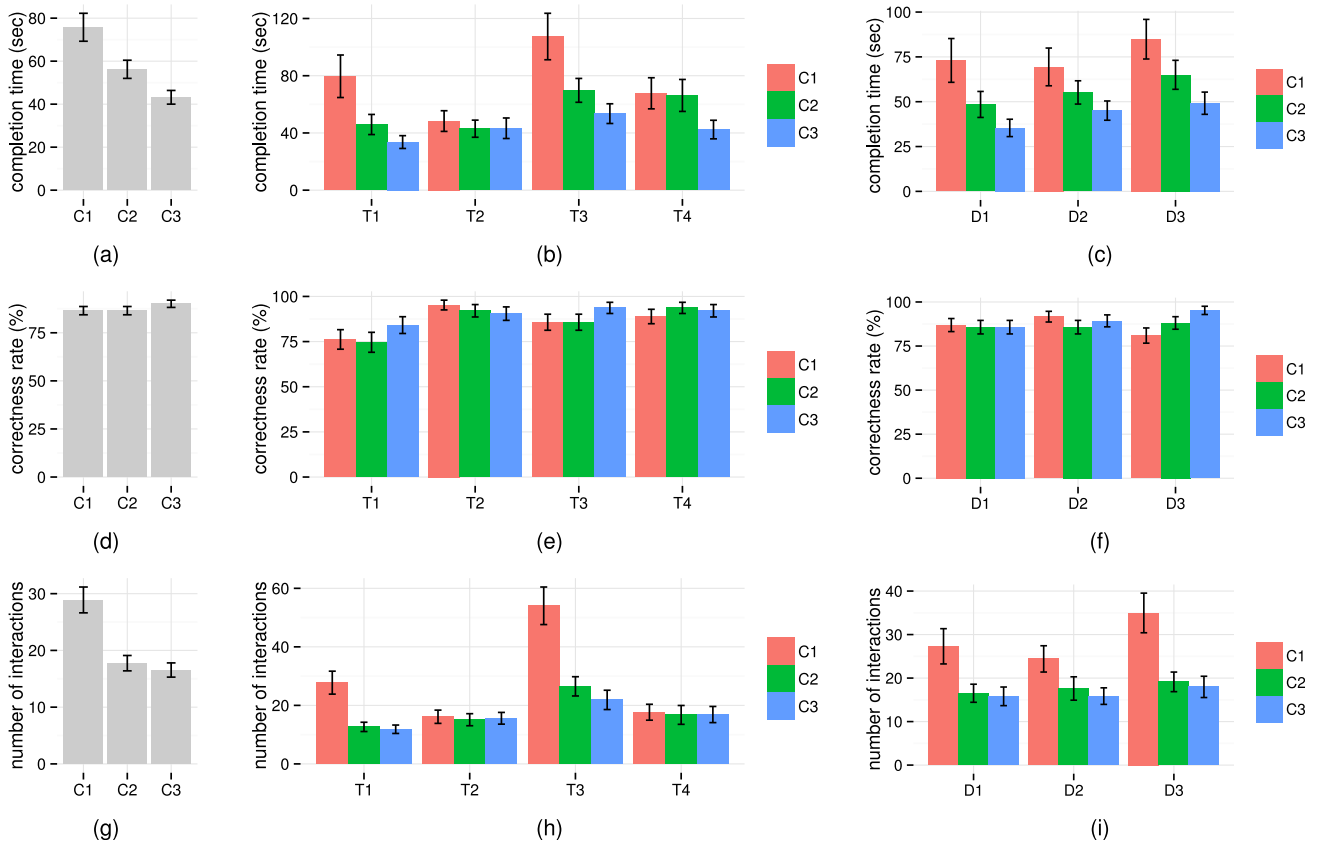


Fig. 11. Results of the experiment. (a) shows the overall *task completion time* in each *visualization condition*, which is then broken down by *tasks* (b) and *datasets (graph sizes)* (c). Similarly, (d-f) show *correctness rate* and (g-i) show *number of interactions* with the same breakdowns. Overall, using C2 and C3 outperforms C1 in terms of *task completion time* and *number of interactions*. While the overall *correctness rate* is not significantly different between *visualization conditions*, using C3 shows significantly higher *correctness rate* than C1 for the largest graph we tested (D3).

percent for C2, and 90.08 percent for C3 (Fig. 11d). However, there is a significant effect of *visualization condition* on *correctness rate* ($F_{2,40} = 6.92$, $p < .01$) only for the largest graph we tested (D3). For D3, average *correctness rate* are 80.95 percent for C1, 88.1 percent for C2, and 95.24 percent for C3 (Fig. 11f). Post-hoc tests using the Bonferroni correction show C3 is significantly different from C1 ($p < .05$) only for D3.

We found that *correctness rates* significantly differ between *tasks* ($F_{2.09,41.78} = 7.39$, $p < .01$, $\varepsilon = .696$ using Greenhouse-Geisser correction). Unsurprisingly, the participants showed higher *correctness rate* for easier *tasks*. Average *correctness rate* are 78.31 percent for T1, 92.59 percent for T2, 88.36 percent for T3, and 91.53 percent for T4.

We did not find a significant effect of *dataset (graph size)* on *correctness rate* ($F_{2,40} = 1.23$, $p = .30$). Average *correctness rate* are 88.61 percent for D1, 88.89 percent for D2, and 88.1 percent for D3.

4.6.3 Number of Interactions

We measured the *number of interactions* as the sum of number of pointing (mouse over on a node) + number of highlighting (left click on a node) + number of selecting (right click on a node). On average, each *task* took 21.06 interactions ($SD = 27.39$) to complete.

A repeated measures ANOVA with a Greenhouse-Geisser correction ($\varepsilon = .603$) shows a significant effect of *visualization conditions* on *number of interactions* ($F_{1.21,24.13} = 25.63$, $p < .0001$). As shown in Fig. 11g, average *number of interactions* are 28.89 for C1 ($SD = 36.15$), 17.75 for C2 ($SD = 21.5$), and 16.55 for C3 ($SD = 19.87$). Post-hoc tests using the Bonferroni correction show C1 is significantly different from C2 ($p < .0001$) and C3 ($p < .0001$).

We found a significant effect of *task* on *number of interactions* ($F_{1.58,31.55} = 19.77$, $p < .0001$, $\varepsilon = .526$, using Greenhouse-Geisser correction) As shown in Fig. 11h, average *number of interactions* are 17.42 for T1 ($SD = 21.65$), 15.6 for T2 ($SD = 16.68$), 34.14 for T3 ($SD = 38.84$), and 17.09 for T4 ($SD = 22.95$).

When we analyzed the results for each *task* separately, there are significant effects of *visualization condition* on *number of interactions* for T1 ($F_{1.23,24.68} = 14.42$, $p < .001$, $\varepsilon = 0.617$, using Greenhouse-Geisser correction) and T3 ($F_{2,40} = 28.63$, $p < .0001$), but not T2 or T4. For both T1 and T3, both C3 ($p < .001$) and C2 ($p < .001$) show significantly fewer *number of interactions* than C1.

There is a significant effect of *dataset (graph size)* on *number of interactions* ($F_{2,40} = 4.56$, $p < .05$). Average *number of interactions* are 19.87 for D1 ($SD = 27.04$), 19.29 for D2 ($SD = 23.83$), and 24.03 for D3 ($SD = 30.73$).

4.6.4 User's Feedback

After the experiment, the participants were asked to answer a post-questionnaire and comment on the experiment freely.

The participants were asked to choose their overall preference between the *visualization conditions* (or *no preference*). All participants except one (who chose *no preference*) preferred C3 over C2 over C1.

All ratings in the questionnaire were measured on a 7 point Likert scale, where toward 1 meant the negative rating (e.g., difficult, not confident), and toward 7 was the positive rating (e.g., easy, confident). We use the Kruskal-Wallis H test (one-way ANOVA on ranks) to test the ratings.

We did not observe a significant difference in confidence ratings (1: no confidence, 7: complete confidence) between the *visualization conditions* ($p = .16$). Mean confidence ratings were 2.95 ($IQR = 2$) for C1, 3.43 ($IQR = 1$) for C2, and 3.62 ($IQR = 2$) for C3.

We found a significant difference in ease-of-use ratings (1: very difficult, 7: very easy) between the *visualization conditions* ($\chi^2(2) = 21.21$, $p < .0001$), with mean ease -use ratings on *visualization conditions* were 2.43 ($IQR = 1$) for C1, 4.33 (IQR = 3) for C2, and 4.67 ($IQR = 3$) for C3.

The participants responded T3 was the most difficult task and T2 was the easiest task. We found a significant difference in task difficulty ratings (1: very difficult, 7: very easy) between the *tasks* ($\chi^2(3) = 27.68$, $p < .0001$), with mean task difficulty ratings were 3.29 ($IQR = 3$) for T1, 5.38 ($IQR = 2$) for T2, 2.62 ($IQR = 1$) for T3, and 4.52 ($IQR = 2$) for T4. The results of *correctness rate* also indicate that T1 and T3 are more difficult than T2 or T4.

There is a significant difference in ease-of-use ratings (1: very difficult, 7: very easy) between the *datasets (graph sizes)* ($\chi^2(2) = 6.04$, $p < .05$), with mean ease-of-use ratings were 4.14 ($IQR = 3$) for D1, 4.43 ($IQR = 2$) for D2, and 3.33 ($IQR = 3$) for D3.

The participants offered many positive comments for C2 and C3 over C1, such as:

- "It was easier to navigate with C2 and C3",
- "It was hard to control the cursor in C1",
- "The nodes near corner of C1 were too far from me",
- "C1 looks like inclined",
- "When I highlight a node in C1 it looks like new edges are appeared",
- "It was difficult to follow paths in C1".

Six participants asked "can I use zoom feature?" when they are using C1 during the experiment. But no one asks it when they using C2 or C3.

For the depth routing (C3) in particular, some user comments were:

- "It was much less cluttered in C3",
- "I can feel richer depth perception with C3",
- "C1 and C2 look flat",
- "C3 looks like coral, C2 looks like broken egg shells",
- "C3 looks more neat".

Three participants explicitly mentioned about the edge crossings when using the visualization without depth routing (C1 and C2). It is natural because the edges without depth routing are laid on same depth so it cause more edge crossings and depth fighting (z-fighting) problem.

We also asked to the participants to report if they had any kind of discomfort (e.g., eyestrain, disorientation, or sickness). Two participants (one wore contact lenses, another one had normal vision) reported they had light eyestrain, dry eye, and felt pressure on near eye area caused by HMD. No one reported disorientation or sickness. Two participants reported they can clearly see pixel grid of the HMD (also known as the screen door effect) which caused by limited pixel-fill-factor of current HMD. However, imminent hardware improvements are addressing this in the next generation of HMDs.

4.7 Discussion

The results of the user study consistently indicate that the spherical graph layouts (C2 and C3) are generally preferred over the 2D graph layout (C1), and the depth routing (C3) helps in most of the tasks. More specifically, using C2 and C3 shows significantly shorter *task completion time* and a significantly fewer *number of interactions* than C1 for more difficult tasks (T1 and T3). Also, using C3 shows significantly higher *correctness rate* than C1 for the largest graph (D3).

Positional tracking is enabled when the 2D graph layout was used. However, participants did not move very much to fully utilize this tracking feature. We suspect that it is because they were seated. This suggests in our future study we might want to include the conditions that participants are standing throughout the experiment.

As shown in Fig. 10, when using the 2D layout (C1), stereoscopic highlighting moves the node such that it no longer coincides with the cursor. A participant would have to move the cursor to the new position of the highlighted node to remove highlight, which increases the *number of interactions* and thus the time required to complete a task. A larger FOV incurs a greater movement of the highlighted node. For example, using FOV of 150 degree would lead to poorer performance of participants than using FOV of 90 degree or FOV of 120 degree. Nevertheless, smaller FOVs would result in more clutter. There is no such a problem with spherical layouts.

While the depth routing noticeably increases the apparent legibility of the graph (as seen in Fig. 8), our user study found that C3 only slightly outperforms C2. The highlighted edges look very similar in C2 and C3 when spatially isolated from the rest of the graph, while the non-highlighted ones look very different between C2 and C3. That is, we found that overall legibility was not as important with respect to user performance as the interactive highlighting techniques, probably because interactive highlighting was more relevant to the specific tasks considered. A future work is thus to study the effects on user performance with techniques (such as depth routing) for improving the aesthetics or legibility of the overall graph by using tasks, for example, on larger scale structures or trends in the graph.

There are possibly other layouts that could work well in an immersive environment, which we intend to explore next. Many traditional layouts can be computed in a 3D space and augmented with fisheye or other distortion techniques. There are some graph visualizations designed for a spherical space already, and they could map well to an immersive environment, such as hyperbolic layouts [33].

We only conducted our study with HMDs because we had to limit the scope of the study. In our following study, we plan to compare HMDs with other display facilities such as 2D monitors, wall-size displays, 3D TVs, CAVE, etc.

We have implemented a basic selection interaction method, but there is a vast range of possible interaction methods that are yet to be explored. While long known about, six degree of freedom (6 DOF) input devices have generally been too imprecise, impractical, or expensive for

general use [61]. But just as the price has gone down and the quality gone up for precise tracking for HMDs, so too has the cost and quality (respectively) of precisely tracked 6 DOF input devices, raising their viability and popularity especially in combination with immersive environments. This suggests another direction of the investigation, as visibility within HMDs limits the usage of traditional input devices. Having a dedicated region of space within which the user can introduce selected foci and directly investigate an interaction is another such possibility.

5 Conclusion

While stereoscopic viewing has previously been shown effective for graph visualization, ours is the first work that achieves immersive graph visualization using a HMD with proven effectiveness. The results of the user study show that participants performed better using our techniques than using traditional 2D graph visualization in an immersive environment, especially for more difficult tasks and larger graphs. Our study with a limited scope only addresses a few particular design considerations. Even though we have conducted our study with the use of a HMD, our design should also be effective in other immersive virtual reality environments. Our work is an early foray into exploring virtual reality techniques for graph visualization. More research is clearly needed before immersive 3D visualization can begin to benefit real-world information visualization tasks. We hope our work and findings will encourage others to join this exciting area of study so they can help accelerate the development of usable technologies to meet the growing demand of more effective tools for examining and analyzing large complex data.

Acknowledgments

This research was supported in part by the National Research Foundation of Korea via BK21 PLUS, by the U.S. National Science Foundation via NSF DRL-1323214 and NSF DE-FC02-12ER26072, and by UC Davis's RISE program.

References

[1] A. Febretti, A. Nishimoto, T. Thigpen, J. Talandis, L. Long, J. D. Pirtle, T. Peterka, A. Verlo, M. Brown, D. Plepys, D. Sandin, L. Renambot, A. Johnson, and J. Leigh, "CAVE2: A hybrid reality environment for immersive simulation and information analysis," in *Proc. Eng. Reality Virtual Reality*, article 864903, 2013.

[2] (2012). Oculus VR, LLC. Oculus Rift. [Online]. Available: https://www.oculus.com

[3] (2015). HTC Corporation. Vive. [Online]. Available: http://www.htcvr.com

[4] (2015). Sony Computer Entertainment Inc. PlayStation VR. [Online]. Available: https://www.playstation.com/en-us/explore/project-morpheus

[5] (2014). Google Inc. Google Cardboard. [Online]. Available: https://www.google.com/get/cardboard

[6] (2014). Samsung Electronics Co., Ltd. Gear VR. [Online]. Available: http://www.samsung.com/global/microsite/gearvr

[7] (2015). Microsoft. Hololens. [Online]. Available: https://www.microsoft.com/microsoft-hololens

[8] R. Brath, "3D InfoVis is here to stay: Deal with It," in *Proc. IEEE VIS Int. Workshop 3DVis*, 2014, pp. 25–31.

[9] J. P. McIntire and K. K. Liggett, "The (possible) utility of stereoscopic 3D displays for information visualization: The good, the bad, and the ugly," in *Proc. IEEE VIS Int. Workshop 3DVis*, 2014, pp. 1–9.

[10] C. Ware and P. Mitchell, "Reevaluating stereo and motion cues for visualizing graphs in three dimensions," in *Proc. Symp. Appl. Perception Graph. Vis.*, 2005, pp. 51–58.

[11] C. Ware and P. Mitchell, "Visualizing graphs in three dimensions," *ACM Trans. Appl. Percept.*, vol. 5, no. 1, article 2, 2008.

[12] B. Alper, T. Hollerer, J. Kuchera-Morin, and A. Forbes, "Stereoscopic highlighting: 2D graph visualization on stereo displays," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 12, pp. 2325–2333, Dec. 2011.

[13] N. Greffard, F. Picarougne, and P. Kuntz, "Beyond the classical monoscopic 3d in graph analytics: An experimental study of the impact of stereoscopy," in *Proc. IEEE VIS Int. Workshop 3DVis*, 2014, pp. 19–24.

[14] O.-H. Kwon, C. Muelder, K. Lee, and K.-L. Ma, "Spherical layout and rendering methods for immersive graph visualization," in *Proc. IEEE Pacific Vis. Symp.*, 2015, pp. 63–67.

[15] S. Bryson, "Virtual reality in scientific visualization," *Commun. ACM*, vol. 39, no. 5, pp. 62–71, 1996.

[16] F.P. Brooks, "What's real about virtual reality?" *IEEE Comput. Graph. Appl*, vol. 19, no. 6, pp. 16–27, Nov. 1999.

[17] A. van Dam, A. Forsberg, D. Laidlaw, J. LaViola, and R. Simpson, "Immersive VR for scientific visualization: A progress report," *IEEE Comput. Graph. Appl*, vol. 20, no. 6, pp. 26–52, Nov./Dec. 2000.

[18] R. Bennett, D. J. Zielinski, and R. Kopper, "Comparison of interactive environments for the archaeological exploration of 3D landscape data," in *Proc. IEEE VIS Int. Workshop 3DVis*, 2014, pp. 67–71.

[19] C. H. B. Weyers, B. Hentschel, and T. W. Kuhlen, "Interactive volume rendering for immersive virtual environments," in *Proc. IEEE VIS Int. Workshop 3DVis*, 2014, pp. 73–74.

[20] K. Mirhosseini, Q. Sun, K. C. Gurijala, B. Laha, and A. E. Kaufman, "Benefits of 3D immersion for virtual colonoscopy," in *Proc. IEEE VIS Int. Workshop 3DVis*, 2014, pp. 75–79.

[21] A. Teyseyre and M. Campo, "An overview of 3D software visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 1, pp. 87–105, Jan./Feb. 2009.

[22] C. Ware, *Information Visualization: Perception for Design*. San Mateo, CA, USA: Morgan Kaufmann, 2004.

[23] J. E. Cutting and P. M. Vishton, "Perceiving layout and knowing distances: The integration, relative potency, and contextual use of different information about depth," in *Perception of Space and Motion*, ser. Handbook of Perception and Cognition, W. Epstein and S. Rogers, Eds. Orlando, FL, USA: Academic, 1995, pp. 69–117.

[24] C. Ware and G. Franck, "Evaluating stereo and motion cues for visualizing information nets in three dimensions," *ACM Trans. Graph.*, vol. 15, no. 2, pp. 121–140, 1996.

[25] H. Halpin, D. J. Zielinski, R. Brady, and G. Kelly, "Exploring semantic social networks using virtual reality," in *Proc. Int. Semantic Web Conf.*, 2008, pp. 599–614.

[26] F. Barahimi and S. Wismath, "3D graph visualization with the oculus rift," in *Proc. 22nd Int. Symp. Graph Drawing*, 2014, pp. 519–520.

[27] I. Herman, G. Melancon, and M. Marshall, "Graph visualization and navigation in information visualization: A survey," *IEEE Trans. Vis. Comput. Graph.*, vol. 6, no. 1, pp. 24–43, Jan.–Mar. 2000.

[28] T. Munzner, "Exploring large graphs in 3D hyperbolic space," *IEEE Comput. Graph. Appl*, vol. 18, no. 4, pp. 18–23, Jul. 1998.

[29] T. Hughes, Y. Hyun, and D. Liberles, "Visualising very large phylogenetic trees in three dimensional hyperbolic space," *BMC Bioinformat.*, vol. 5, no. 1, 2004.

[30] T. Sprenger, M. Gross, A. Eggenberger, and M. Kaufmann, "A framework for physically-based information visualization," in *Visualization in Scientific Computing*. New York, NY, USA: Springer, 1997, pp. 71–83.

[31] Y. Wu and M. Takatsuka, "Visualizing multivariate network on the surface of a sphere," in *Proc. Asia-Pacific Symp. Inf. Vis.*, 2006, pp. 77–83.

[32] S. G. Kobourov and K. Wampler, "Non-Euclidean spring embedders," *IEEE Trans. Vis. Comput. Graphics*, vol. 11, no. 6, pp. 757–767, Nov./Dec. 2005.

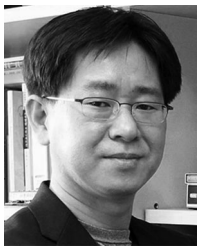
[33] T. Munzner, "H3: Laying out large directed graphs in 3D hyperbolic space," in *Proc. IEEE Symp. Inf. Vis.*, 1997, pp. 2–10.

[34] D. Holten, "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data," *IEEE Trans. Vis. Comput. Graphics*, vol. 12, no. 5, pp. 741–748, Sep./Oct. 2006.

[35] O. Ersoy, C. Hurter, F. Paulovich, G. Cantareiro, and A. Telea, "Skeleton-based edge bundling for graph visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2364–2373, Dec. 2011.
[36] D. Holten and J. J. van Wijk, "Force-directed edge bundling for graph visualization," in *Proc. Eurographics/IEEE-VGTC Conf. Vis.*, 2009, pp. 983–998.
[37] O. Mallo, R. Peikert, C. Sigg, and F. Sadlo, "Illuminated lines revisited," in *Proc. IEEE Vis.*, 2005, pp. 19–26.
[38] D. C. Banks and C.-F. Westin, "Global illumination of white matter fibers from DT-MRI data," in *Visualization in Medicine and Life Sciences*. New York, NY, USA: Springer, 2008, pp. 173–184.
[39] S. Eichelbaum, M. Hlawitschka, and G. Scheuermann, "LineAO—Improved three-dimensional line rendering," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 3, pp. 433–445, Mar. 2013.
[40] M. H. Everts, H. Bekker, J. B. T. M. Roerdink, and T. Isenberg, "Depth-dependent halos: Illustrative rendering of dense line data," *IEEE Trans. Vis. Comput. Graphics*, vol. 15, no. 6, pp. 1299–1306, Nov./Dec. 2009.
[41] M. H. Everts, H. Bekker, J. B. T. M. Roerdink, and T. Isenberg, "Flow visualization using illustrative line styles," in *Proc. Nat. ICT.Open/SIREN Workshop*, article 44, 2011.
[42] M. Luboschik and H. Schumann, "Illustrative halos in information visualization," in *Proc. Working Conf. Adv. Vis. Interfaces*, 2008, pp. 384–387.
[43] B. Jenny, "Adaptive composite map projections," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 12, pp. 2575–2582, Dec. 2012.
[44] C. W. Muelder and K.-L. Ma, "A treemap based method for rapid layout of large graphs," in *Proc. IEEE Pacific Vis. Symp.*, 2008, pp. 231–238.
[45] C. W. Muelder and K.-L. Ma, "Rapid graph layout using space filling curves," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, no. 6, pp. 1301–1308, Nov./Dec. 2008.
[46] H. S. M. Coxeter, *Introduction to Geometry*. New York, NY, USA: Wiley, 1969.
[47] J. M. Dennis, "Partitioning with space-filling curves on the cubed-sphere," in *Proc. Parallel Distrib. Process. Symp.*, 2003, pp. 6–11.
[48] (1995). NASA. Space flight human-system standard volumes 1. [Online]. Available: http://msis.jsc.nasa.gov/volume1.htm
[49] J. W. Youdas, T. R. Garrett, V. J. Suman, C. L. Bogard, H. O. Hallman, and J. R. Carey, "Normal range of motion of the cervical Spine: An initial goniometric study," *Phys. Therapy*, vol. 72, no. 11, pp. 770–780, 1992.
[50] C. de Boor, "On calculating with B-splines," *J. Approximation Theory*, vol. 6, no. 1, pp. 50–62, 1972.
[51] K. Shoemake, "Animating rotation with quaternion curves," *ACM SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 245–254, 1985.
[52] S. R. Buss and J. P. Fillmore, "Spherical averages and applications to spherical splines and interpolation," *ACM Trans. Graph.*, vol. 20, no. 2, pp. 95–126, 2001.
[53] M.-J. Kim, M.-S. Kim, and S. Y. Shin, "A general construction scheme for unit quaternion curves with simple high order derivatives," in *Proc. Ann. Conf. Comput. Graph. Interactive Techn.*, 1995, pp. 369–376.
[54] (2012). Epic Games. Unreal Engine 4. [Online]. Available: https://www.unrealengine.com
[55] (2010). Leap Motion, Inc. Leap Motion. [Online]. Available: https://www.leapmotion.com
[56] Prabhat, A. Forsberg, M. Katzourin, K. Wharton, and M. Slater, "A comparative study of desktop, fishtank, and cave systems for the exploration of volume rendered confocal data sets," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, no. 3, pp. 551–563, May/Jun. 2008.
[57] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropolog. Res.*, vol. 30, pp. 452–473, 1977.
[58] D. E. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Computing*. Reading, MA, USA: Addison-Wesley, 1993.
[59] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, 2002.
[60] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.
[61] S. Zhai, "User performance in relation to 3D input device design," *ACM SIGGRAPH Comput. Graph.*, vol. 32, no. 4, pp. 50–54, 1998.

Oh-Hyun Kwon received the Bachelor's degree in digital media from the Ajou University in 2013. He is currently working toward the PhD degree at the University of California, Davis and the Ajou University. His research interests include information visualization, computer graphics, and virtual reality. He is a student member of the IEEE.

Chris Muelder received the PhD degree in computer science from the University of California at Davis in 2011. He then was a postdoctoral researcher with the VIDI Labs at UC Davis during 2011-2014. His research interests include information visualization and visual analytics. He is currently with Google as a software engineer. He is a member of the IEEE.



Kyungwon Lee received the MFA degree in computer graphics and interactive media from the Pratt Institute in 2002. He is a professor in the Department of Digital Media and the director of Integrated Design Lab at the Ajou University. His research interests include information visualization, human-computer interaction, and media art. He is a member of the IEEE.

Kwan-Liu Ma received the PhD degree in computer science from the University of Utah in 1993. He is a professor of computer science and the chair of the Graduate Group in Computer Science (GGCS) at the University of California, Davis. He leads VIDI Labs and and the UC Davis Center for Visualization. His research interests include visualization, high-performance computing, and user interface design. He received the US National Science Foundation (NSF) PECASE award in 2000, and the IEEE VGTC 2013 Visualization Technical Achievement Award. He has served as a papers chair for SciVis, InfoVis, EuroVis, and PacificVis, and also as an associate editor of IEEE TVCG (2007-2011) and the *Journal of Computational Science and Discoveries* (2009-2014). He is a founder of PacificVis, Ultravis, and LDAV. He presently serves on the editorial boards of the *IEEE CG&A* and the *Journal of Visualization*. He is a fellow of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.