iSphere: Focus+Context Sphere Visualization for Interactive Large Graph Exploration

Fan Du
University of Maryland
fan@cs.umd.edu

Nan Cao
Tongji University
nan.cao@gmail.com

Yu-Ru Lin
University of Pittsburgh
yurulin@pitt.edu

Panpan Xu
Hong Kong University of Science and Technology
pxu@cse.ust.hk

Hanghang Tong
Arizona State University
hanghang.tong@asu.edu

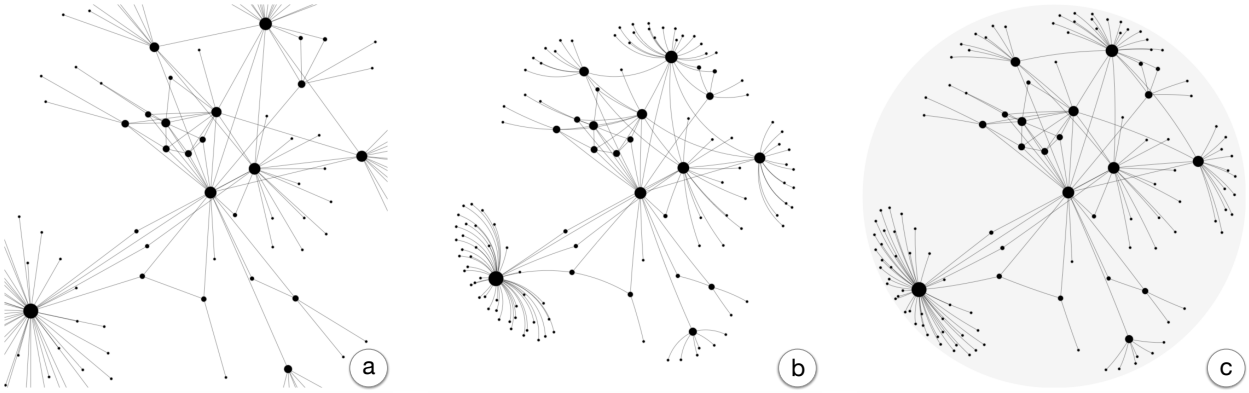

Figure 1. The node-link diagram shown in (a) the zoomable plane and the focus+context displays, (b) hyperbolic display and (c) iSphere display.

ABSTRACT

Interactive exploration plays a critical role in large graph visualization. Existing techniques, such as zoom-and-pan on a 2D plane and hyperbolic browser facilitate large graph exploration by showing both the details of a focal area and its surrounding context that guides the exploration process. However, existing techniques for large graph exploration are limited in either providing too little context or presenting graphs with too much distortion. In this paper, we propose a novel focus+context technique, iSphere, to address the limitation. iSphere maps a large graph onto a Riemann Sphere that better preserves graph structures and shows greater context information. We conduct extensive experiment studies on different graph exploration tasks under various conditions. The results show that iSphere performs the best in task completion time compared to the baseline techniques in link and path exploration tasks. This research also contributes to understanding large graph exploration on small screens.

CHI 2017, May 06–11, 2017, Denver, CO, USA
 ISBN 978-1-4503-4655-9/17/05...$15.00
DOI: http://dx.doi.org/10.1145/3025453.3025628

Author Keywords

Graph visualization; graph exploration; focus+context.

ACM Classification Keywords

H.5.2 User Interfaces: Graphical user interfaces (GUI)

INTRODUCTION

Interactive exploration is one of the major approaches for navigating through large graph data, which is widely adopted by many graph visualization systems [10, 12, 40, 42]. It has been used for, for example, navigating through a big road network on Google Maps or finding relational patterns such as communities or critical paths between communities in a large social network [10]. Various visualization and interaction techniques have been developed for supporting *large* graph exploration, that is, the exploration of a large graph on a relatively *small* screen. In particular, users can query to find and visualize the parts of interest of a large graph in a node-link diagram [10–12], explore an aggregated graph via semantic zooming [2, 6, 19, 39, 47, 50], use brushing on a graph overview to show the detailed structures in a separate view [2, 27], or directly perform zooming and panning on a graph in a focus+context display [42, 46].

However, most of the above techniques have limitations for exploring a large graph. For example, the query based approach is only useful when the properties of the desired structure are

known, so that queries can be formulated. Semantic zoom is usually inefficient when the hierarchy of the aggregated graph is deep. Overview+detail based approaches break the spatial continuity of a graph representation, making it inefficient for, for example, tracing a long path in the graph. Focus+context displays circumscribe the aforementioned issues by showing both the focal details and overview context smoothly and continuously within the same view, and they have been the core of recent hybrid techniques that aim to use a small screen more efficiently [38, 56]. Two techniques in this category, plane (Fig. 1(a)) and hyperbolic [31, 42] (Fig. 1(b)) displays, have been widely used for graph exploration, but also have limitations. The first technique often shows too little context to guide the exploration while the second one tends to introduce too much distortion when displaying the graph links, making path tracing difficult [31, 57].

To address the above issues, we introduce a novel focus+context display, iSphere (Fig. 1(c)), for supporting interactive large graph exploration. It maps a node-link diagram onto a Riemann Sphere [3] and orthogonally projects the sphere onto a 2D plane. The produced focus+context view, similar to the hyperbolic display, has a focal area in the middle surrounded by context suppressed at the periphery of the circular display. We compared iSphere with the plane and hyperbolic displays in two controlled user studies, based on three graph exploration tasks, which were designed for exploring three fundamental graph elements: nodes, links, and paths. In particular, *Study I* investigated iSphere's performance in a full-sized window on a regular laptop computer and *Study II* investigated iSphere's performance in smaller windows, mimicking the screen sizes of the mainstream personal-computing devices, i.e., tablets, mobile phones, and smart watches.

The study results showed iSphere had the best performance under several testing conditions. In particular, it significantly outperformed the hyperbolic display in path exploration tasks and significantly outperformed the plane display in both link and path exploration tasks in smaller windows. These results suggested the potential of using iSphere on mobile devices for exploring the routes inside a large graph. Overall, this paper has the following contributions:

- We propose a novel focus+context display for large graph exploration, which shows more context when compared to the plane display and better preserves graph structures when compared to the hyperbolic display.
- We report the first, to the best of our knowledge, comprehensive user studies, comparing two different focus+context techniques as well as the zoomable plane, while varying exploration tasks, screen sizes, and graph sizes.
- We provide extensive analysis and discussion of the study results that give insight to when, why, and how different graph displays are useful or have limitations. We further discuss the benefit of applying iSphere on smaller screens (tablets, mobile phones, smart watches) and how it helps users make sense of a large graph during the exploration, which provides insight into the trade-offs between the amount of distortion and contextual details on small displays.

GRAPH EXPLORATION TECHNIQUES

In this section, we review and compare interaction and visualization techniques developed for exploring large-scale graphs shown in node-link diagrams. We focus on the most generic and related techniques, including (1) query based approach, (2) semantic zoom in hierarchical graphs, (3) zooming and panning, (4) overview+detail, and (5) focus+context. More comprehensive and general surveys can be found in [40, 51].

Interaction Techniques

There are three generic interaction techniques that are developed for exploring a large graph, including (1) the query based approach, (2) semantic zoom in hierarchical graphs, and (3) zooming and panning.

Query Based Approach. Query (or search) is the simplest approach for exploring a large graph, which has been adopted by many graph visualization systems. Via this approach, users can query the labels of nodes or links to find and visualize a graph portion of their own interests. For example, FacetAtlas [11] was designed for exploring a word co-occurrence graph extracted from documents by querying on different topics. g-Miner [10] supports the exploration of large multivariate graphs via querying on both the graph structure and node attributes. However, query based graph exploration is only efficient when users' tasks are clear so that queries can be formulated. In many real world applications, users explore the graph without a specific goal, which cannot be accomplished with query based approaches.

Semantic Zoom in Hierarchical Graph. Semantic zoom is commonly used in hierarchical graph visualizations [2,6,19,39, 47, 50], in which the nodes in a large graph are hierarchically aggregated into so-called "meta-nodes" and the graph structure with respect to the meta-nodes is displayed. It reduces the total number of nodes and links to be visualized at a time [20]. Usually, the first level of the hierarchy is shown by default. When users zoom into the next level, the graph at the specified level is visualized. In contrast to the ordinary graphical zoom, semantic zoom not only changes the graphical representation, but also updates the data to be displayed [54].

When compared to the query based approach, hierarchical graph visualizations provide additional context to guide the data navigation procedure. The meta-nodes show potential options to be zoomed into. However, important information such as how a leaf connects to other leaves under different branches is still missing. Users have to zoom back and forth to navigate through different branches. When the hierarchy is deep or has a large fan-out, semantic zoom is inefficient.

Graphical Zooming and Panning. One solution to avoid the inefficient semantic zoom is to flatten or even remove the hierarchy. However, the *presentation problem*, i.e., the lack of space for showing a large dataset [18], makes node-link diagrams incapable of representing a large graph without any aggregation. Zooming and panning [44] are the most commonly used interactions that address this issue. They are often used together to help users control the exploration. Many zoomable user interfaces [9,44] have been developed, and they are also frequently used for exploring large graph data [48]. We adopted these interactions in iSphere to support a continuous exploration in a focus+context display.

Visualization Techniques

Two generic visualization design principles, overview+detail and focus+context, are also used for exploring large graphs.

Overview+Detail. An overview+detail user interface of a large graph consists of multiple views with an overview (overall view) illustrating the entire graph and other views showing the details of the graph from different perspectives. A typical implementation of this technique combines the two views together based on two different approaches: (1) the overview is displayed in a small inspection window that floats on top of another big window with a selected portion of the graph shown in detail (or in the opposite) [27], and (2) the details are visualized in a small inspection lens which floats on top of the overview window [30, 34]. The overview and the detail view are linked together by interactions. Users can choose to visualize the details of any part by brushing on the overview in the first design or moving the lens in the second design.

Although simple to implement and widely used, showing overview and details in two separated views breaks the spacial continuity of the visual representation, making this technique difficult for a user to, for example, trace or recall a long path in a large graph.

Focus+Context. Focus+context [7] is another generic visualization design principle that is commonly used for big data exploration. It enables viewers to see the objects of primary interest presented in full details while at the same time get an overview-impression of the surrounding context. A focus+context display guarantees the spacial continuity and enables a smooth data navigation via zooming and panning. Although many techniques have been developed [13], here, we only focus on those designs for representing graphs.

The simplest way for representing a large graph is to directly show the graph in a zoomable and pannable window, in which only a small portion of the data is shown in a focal viewport. Users can navigate through the entire data by zooming and panning the viewport. Distortion oriented techniques [37] have been developed to improve this design by magnifying the focal area while showing more surrounding context without increasing the display size or losing the spacial continuity. These techniques, such as the polyfocal projection [29], the Bifocal Display [5], and various types of graphical fisheye views [22, 37, 46] and topological fisheye views [1, 14, 25], produce similar results in which the context regions are suppressed, leaving space for showing more details in the focal area. However, given the limited distortion rate, these approaches are incapable of transforming the entire visualization plane into a focus+context display [41]. Besides, past user studies on distortion techniques indicate that introducing distortion will impair users' performance as they must remain aware about the distortions in shape and position and mentally undo them when needed [23]. Therefore, the distortion oriented techniques are often used as an interactive lens to magnify a small focal region inside a large display [49].

There have also been hybrid focus+context techniques that focused on how to use a small screen more efficiently. For example, to better allocate space to display details of a user's interest, Li and Takatsuka proposed a context-based filtering technique that combines the degree of interest of Logical Fisheye View and geometric distortion [38]. ViSizer [56] automatically resizes a visualization to fit any display by combining multi-focus+context and grid deformation to avoid feature congestion. Such hybrid techniques are complementary as they can be used to augment existing focus+context methods.

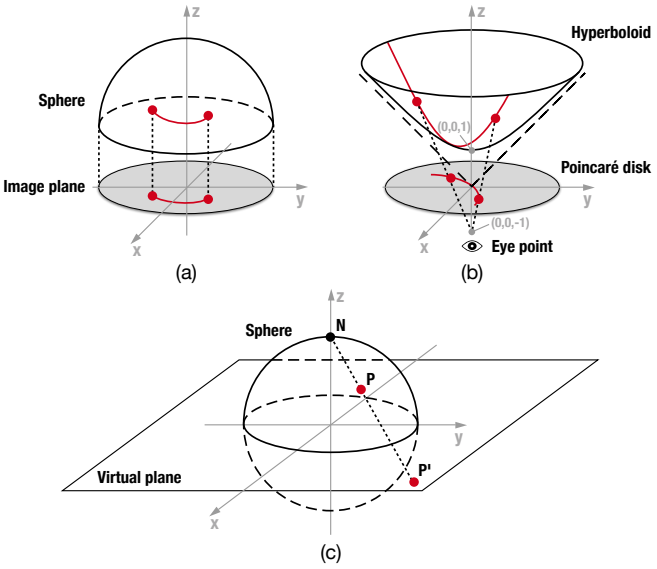

Figure 2. Illustrations of (a) the orthogonal projection from a sphere to a Euclidean plane, (b) the perspective projection from a hyperboloid surface to the Poincaré disk, and (c) the conformal mapping from the Riemann Sphere to a virtual plane.

Non-Euclidean geometry was used for producing a focus+context display with a much higher distortion rate. In three dimensions, there are two types of non-Euclidean geometries, namely, elliptic geometry and hyperbolic geometry [55]. Elliptic geometry can be visualized as the surface of a sphere in the three dimensional space on which "lines" on a Euclidean plane are shown as big circles. When the sphere is shown in a 2D display on the screen based on the orthogonal projection, it automatically generates a focus+context display (Fig. 2(a)). Using this technique, Kobourov and Wampler [31] extended the traditional force directed layout to directly lay out a graph on the surface of a sphere. Ito et al. [28] also proposed algorithms for visualizing a bipartite graph on a sphere. However, these techniques are limited by the "presentation problem" as the surface area of a sphere is limited. Recently, Kwon et al. [33] applied spherical graph layout in an immersion system, but its usage on a 2D plane remains unclear.

Hyperbolic geometry can be represented by the Poincaré disk model [21] in two dimensions. As shown in Fig. 2(b), this model compresses the infinite surface of the forward sheet (S^+) of a two-sheet hyperboloid into a circular disk based on a prospective projection at (0,0,-1). In this way, the point (0,0,1) on the hyperboloid surface is projected onto the center of the Poincaré disk. The points that are infinitely far away are projected onto the boundary of the Poincaré disk. This approach provides an extremely high distortion rate that suppresses the surface of S^+ into a unit disk which can be displayed in an arbitrary-sized window. In addition, when compared to other models [4], the Poincaré disk model, also known as the conformal disk model, preserves angles, which is an important property for representing structures and shapes. These nice properties attracted attention in the field of information visualization and the Poincaré disk model has been extensively used to produce a focus+context view for exploring large tree structures [35], graphs [31, 42] (in both 2D and 3D), and multi-dimensional data [16, 53]. Besides, Kreuseler et al. [32] extend this model by introducing additional degrees of freedom for

exploring complex hierarchical graphs, which allows users to change the focal area by moving the center of the projection. These are the state-of-the-art focus+context techniques that are most related to iSphere. However, as discussed in [31], although the Poincaré disk model preserves angles, it distorts lines, which makes path tracing in a graph difficult [31, 57].

iSphere circumscribes the aforementioned issues by introducing a focus+context display with a moderate distortion rate that provides more context than the plane display and also reduces the distortion of links when compared to the hyperbolic display. Our user studies showed that iSphere was efficient for supporting graph exploration tasks and was significantly better than the hyperbolic display in many situations.

ISPHERE VISUALIZATION

In this section, we discuss the design philosophy and technical details about the implementation of iSphere.

Design Philosophy

iSphere was designed for visualizing and exploring large graphs. We attempted to develop a generic technique that can be used for all types of graphs (directed/undirected, unipartite/bipartite/multipartite, planar/non-planar) of different sizes that are visualized in a node-link diagram but is *independent* of any type of node-link graph layouts. To this end, we introduced a projection based approach that transforms the graph visualization into a focus+context display as a whole.

Specifically, we first visualize a graph on a virtual (or conceptual) plane ($\mathbb{P}$). It has an infinite size so that graphs in any scale can be clearly shown. We then map $\mathbb{P}$ onto a unit sphere $\mathbb{S}$ based on the inverse of the stereographic projection [15]. After that, $\mathbb{S}$ is scaled and rendered into a two-dimensional view port based on the orthogonal projection, which transforms $\mathbb{P}$ into a focus+context display. We describe the details of each step in the rest of this section.

Placing a Graph in a Virtual Plane

The vertices of a graph can be placed onto the virtual plane ($\mathbb{P}$) by using any of the existing graph layout algorithms. This procedure usually runs offline as most of the current layout algorithms are inefficient when the graph is large. In our implementation, stress majorization [24] is applied.

Mapping the Virtual Plane onto the Riemann Sphere

The virtual plane, with the graph visualized on it, is then mapped onto the surface of a unit sphere, also known as the Riemann Sphere ($\mathbb{S}$), based on the inverse of the stereographic projection. This projection was first introduced for producing a map, which projects the surface of a sphere (e.g., the Earth) onto a plane (e.g., a map). It is obtained by projecting a point $P(x,y,z)$ on sphere $\mathbb{S}$ from the sphere's north pole N to a point $P'(X,Y)$ that intersects sphere at the equator (Fig. 2(c)). The projection is given by the following formula:

$$(X,Y) = \left(\frac{x}{1-z}, \frac{y}{1-z}\right)$$

In iSphere, we perform the inverse of this projection to map the virtual plane onto the surface of $\mathbb{S}$, formally described as:

$$(x,y,z) = \left(\frac{2X}{X^2+Y^2+1}, \frac{2Y}{X^2+Y^2+1}, \frac{X^2+Y^2-1}{X^2+Y^2+1}\right)$$

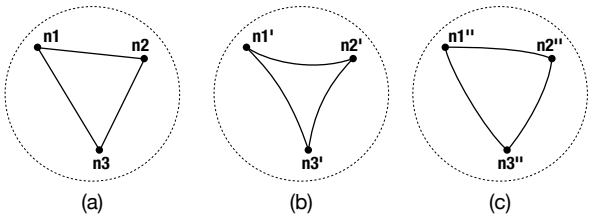

Figure 3. An illustration of a triangle displayed on (a) a Euclidean plane, (b) a Poincaré disk, and (c) a Riemann Sphere.

This is a *conformal* mapping, through which angles are preserved. As shown in Fig. 2(c), in this mapping, the plane region inside the sphere is mapped onto the surface of the south hemisphere, thus being magnified and becoming the focus of the view. The plane region outside the sphere is suppressed and mapped onto the north hemisphere as the context. Particularly, all the points infinitely far away from the plane center are compressed into the north pole.

Rendering

Usually, the south hemisphere of $\mathbb{S}$ is rendered in front of users based on orthogonal projection as it displays the focal area in the above mapping model. In our system, each node of the graph is mapped on the sphere. However, the mapping procedure can be parallelized at the pixel level while rendering on a GPU, making it efficient and general enough for representing any visualization shown on $\mathbb{P}$ beyond node-link diagrams.

When graph links are arbitrary curves (e.g., bundled arcs) on $\mathbb{P}$, they have to be rendered at the pixel level by mapping each pixel (or densely sampling some of the pixels) in the curve onto the sphere and connecting them by an interpolating spline as an approximation. This procedure can also be accelerated by a GPU. When graph links are displayed as straight line segments on $\mathbb{P}$, the rendering can be simplified based on a property of the stereographic projection, i.e., a straight line is mapped to a large circle on $\mathbb{S}$. Therefore, a link when visualized on $\mathbb{S}$ will be a circular arc segment, which can be determined by the mapping results of the link's endpoints and midpoint. We implemented the second approach assuming that all the links are straight lines on $\mathbb{P}$.

Interactions

Zooming and panning are also implemented in iSphere for navigating through the virtual plane. Users can zoom in or out by decreasing or increasing the radius of $\mathbb{S}$ to exclude or include the content inside the focal area (i.e., the south hemisphere). They can also pan the graph on $\mathbb{S}$ based on the Möbius transformation [17]. Another equivalent implementation of these interactions is to keep $\mathbb{S}$ unchanged and simply zooming and panning on the virtual plane. The mapping is refreshed in each mouse operation. Both methods provide a smooth and continuous transforming effect. Especially, the effect of panning on iSphere looks like the rotation of a sphere, which was preferred by most of the participants in our user studies.

Comparing to the Hyperbolic Display

A hyperbolic based focus+context display [31,35] shares many similar properties and functions with iSphere. First, both of them distort the original Euclidean plane into a circular focus+context display in which the center region is magnified for showing the focus and the surrounding region is suppressed

for the context. Second, they both provide a conformal transformation, so that the graph structures are preserved on these displays. Third, they both map an infinite information space (i.e., S^+ on a Poincaré disk and $\mathbb{P}$ on a Riemann Sphere) into a unit display that can be visualized in windows of any size.

They also have many distinct properties. First, the hyperbolic display has a much higher distortion rate. Intuitively, it maps points infinitely far away in the information space onto the edge of a Poincaré disk, making the entire space visible to users. In this case, high-level topology tasks such as identifying clusters may be easier. However, the heavy distortion impacts user interaction as small movements can trigger big visual changes. In contrast, iSphere maps the information space onto a 3D Riemann Sphere, which is orthogonally projected onto a 2D display, thus only showing half of the sphere (i.e., information space) and hiding nodes that are far away from the focus. Second, on a Poincaré disk, a hyperbolic line is an arc of a Euclidean circle contained within the disk that is perpendicular to the boundary of the disk, which is different from the line shown on a Riemann sphere as discussed above. Third, hyperbolic geometry has a constant negative sectional curvature, whereas the Riemann Sphere's is always positive. These properties lead to different representations of a structure. As illustrated in Fig. 3, a triangle on a Euclidean plane is shown differently on a Poincaré disk (curving inward) or on a Riemann Sphere (curving outward).

In the following sections, we will investigate how these properties affect users' performance in graph exploration tasks.

EXPERIMENT DESIGN

In this section, we introduce the design of the user studies and provide rationales for the key design decisions based on our pilot studies and literature survey.

Baselines

In the studies, we compare the different displays introduced above (Fig. 1), including the plane display (D_p), the hyperbolic display (D_h), and the proposed iSphere display (D_s). All of them support zooming and panning (Fig. 4) and were implemented in Java. In particular, developing a two dimensional hyperbolic display for showing a generic graph visualization is non-trivial as the original system was introduced for representing and exploring a tree or a tree-like graph [35] based on a radial layout. To ensure a fair comparison, we kept the layout method (i.e., the stress majorization [24]) unchanged in all three displays. We implemented the projection techniques introduced in [31] and implemented zooming by adjusting distances between nodes based on the Poincaré metric [8] and panning based on the Möbius transformation [17].

Tasks

Following the task taxonomy of graph visualizations [36], we designed three graph exploration tasks in our studies that respectively focused on exploring nodes, links, and paths:

T1: *Finding target(s) in the neighbors of a given node (node exploration).* Given a node A, find a target node B with the highest degree in A's neighborhood.

T2: *Finding target(s) in the common neighbors of two given nodes (link exploration).* Given two nodes A and B, find a target node C with the highest degree among the common neighbors of A and B.

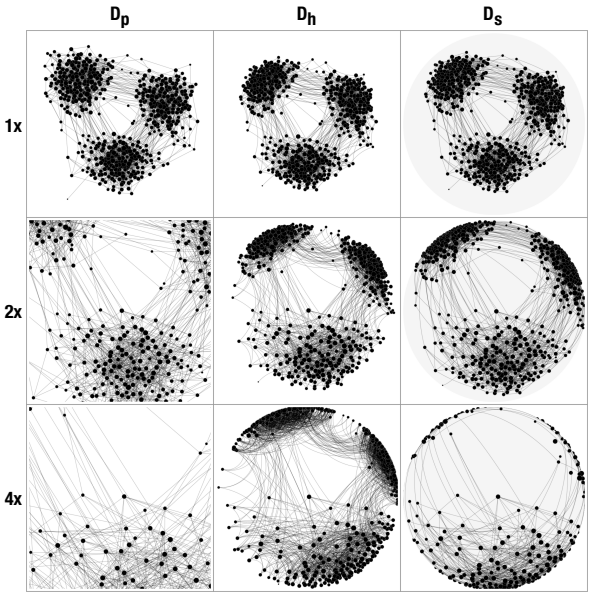

Figure 4. Illustrations of the zooming effect of D_p, D_h, and D_s. The graph contains 512 nodes and 4,096 links.

T3: *Finding target(s) on a given path (path exploration).* Given a path from node A to node B, find a target node C with the highest degree among the nodes on the path.

All these tasks were designed to investigate a wide-range exploration of the graph via zooming and panning in both the plane display and focus+context displays, so that the displays' pros and cons can be revealed. For example, to find the most connected neighbor of a given node A in a large graph, users must zoom into details and pan around to see and compare different neighbors of A. A distortion based focus+context display might be helpful as it shows more context that guides the navigation when compared to the plane display. Here, users were asked to select one single target instead of multiple targets (e.g., all the neighbors of a node) in order to minimize their operations. Fixing the target number also ensures a fair comparison of the task completion time in all cases, as selecting different amounts of targets will cost different length of time which could confound the study results. We did not include high-level topology tasks (e.g., identifying clusters) since our pilot users were able to complete them easily through zooming functions. When the zooming factor is small enough, all three techniques look quite similar, which makes such tasks not differentiable and less informative.

Variables

The primary variable to test in the user studies was the three different displays shown in Fig. 1. In addition, a display's capability of supporting the exploration of a large graph can be affected by the size and community structure of the graph, as well as the size of the display window—these are additional variables used for generating different testing conditions in our studies. In particular, we applied the graph data model introduced by Sah et al. [45] to produce undirected random graphs whose sizes and community structures were precisely controlled for different testing conditions. We also chose to use four different window sizes in the studies. In addition, a pilot study with 4 users was conducted to determine many other conditions that may affect users' performance.

We did not include distortion lens based techniques in the experiments because such techniques alone are often insufficient to cover a relatively large virtual plane. To make the study conditions comparable, we have to use the lens on top of a zooming and panning display, but this setting confounds with other factors—it is hard to tell if the changes in performance are caused by the distortion lens or the zooming and panning. Therefore, we focused on techniques that are able to display an infinitely large virtual plane directly. We believe a distortion lens is more like a widget instead of a display, which can be attached to all three types of displays described in this paper.

Graph Size. We controlled the graph size via two parameters: the number of nodes and the average degree of the graph. We tested a wide range of different choices in the pilot study and selected the parameters that best differentiated users' performance. In particular, we chose graphs with at most $2,048$ nodes to ensure that all the tasks can be completed by users in approximately 30 seconds. To the best of our knowledge, this is the largest graph scale that has ever been tested in a controlled user study. However, such graph size is still far from "big." To investigate the scalability of each display while keeping the tests to be completed within a bearable time for users, we proposed to conduct a scale-up experiment to illustrate the performance trends when the size of the graph is exponentially increased. Specifically we tested users' performance based on small graphs ($2^7 = 128$ nodes), medium graphs ($2^9 = 512$ nodes), and large graphs ($2^{11} = 2,048$ nodes) in a row. In addition, we chose 8 as the average degree as it produced the most feasible tasks according to our pilot users—neither too dense nor too sparse. Accordingly, the number of links of a small/medium/large graph was set to $1,024/4,096/16,384$.

Graph Structure. We controlled the community structure of a graph based on modularity [43], which is a measurement designed to estimate the strength of the division of a graph into clusters. Intuitively, a graph with a high modularity shows clear modular structures (i.e., clusters) (Fig. 5(a)), whereas, a graph with a low modularity shows no clear structures (Fig. 5(b)). We tested a wide range of modularity scores in the pilot study and selected 0.3 and 0.6 as low and high modularity scores, respectively. We also fixed the number of clusters to be three in the testing graphs to produce testing cases with a moderate difficulty.

Window Size. We displayed the generated graphs inside areas of four different sizes to mimic the screen sizes of the mainstream personal-computing devices, i.e., laptop computers, tablets, mobile phones, and smart watches. Particularly, we chose the window size for mimicking laptop computers to be $1,366 \times 768$ pixels (361×203 mm on a standard 96 dpi display), which is the most common screen size according to the latest display statistics [52]. We also chose to use iPad Air, iPhone 6, and iWatch to represent tablets, mobile phones, and smart watches, whose screens are 148×197 mm, 58×104 mm, and 36×42 mm in dimension, respectively.

In the pilot study, we also found that despite the graph size, graph modular structure, and window size, the difficulty of a task also heavily depends on the number of nodes needed to be explored and their distributions in the testing graph. To control the difficulty, we fixed these factors when producing the testing data. In particular, we selected a node A whose degree was 15 and neighbors were scattered inside different clusters for *T1*, selected two nodes A and B belonging to different clusters and sharing exactly three common neighbors for *T2*, and selected a path with the length of 5 and all its containing nodes scattered in three different clusters for *T3*. These parameters were determined in the pilot study to ensure a moderate difficulty of each task.

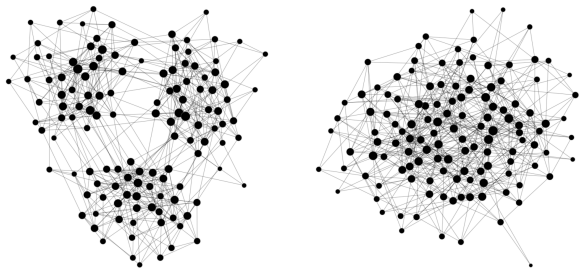

Figure 5. Examples of (a) a high-modularity graph and (b) a low-modularity graph, both of size 128 nodes and 1,024 links.

Study Conditions

A full investigation of the above independent variables and tasks leads to a total of 648 testing trials[1] for each participant, which takes about five hours to finish, thus making the study impossible to be conducted. To reduce the fatigue issue and ensure the quality of the results, instead of testing all the conditions at once, we designed two within-subject studies[2] in a row to cover different testing conditions.

Study I tests how different techniques (D_p, D_h, D_s), graph sizes (small, medium, large), and modularity settings (low, high) affect a user's performance on exploring a graph. We used a full-sized window on a regular laptop computer (i.e., $1,366 \times 768$ pixels). In this study, each participant needed to perform 54 study trials three times, once for each of the three tested graph exploration techniques, yielding a total of $2,916$ ($18 \times 3 \times 54$) trials as summarized in Table 1.

Study II tests how different window sizes (i.e., tablet, mobile phone, smart watch) affect a user's performance on exploring a large graph using D_s or D_p. Here, we chose to only use large graphs to increase the difficulty of the tasks. We also eliminated the cases of using D_h given its poor performance on small screens according to our pilot study (several users even failed to complete the tasks using D_h when the screen is of the size of a smart watch). In this way, as summarized in Table 1, each participant needed to perform 54 study trials twice, once for each tested graph exploration technique, yielding a total of $1,944$ ($18 \times 2 \times 54$) trials.

	Study I	***Study II***
Techniques	D_p, D_h, D_s	D_p, D_s
Tasks	T1, T2, T3	T1, T2, T3
Window Sizes	Laptop Computer	Tablet, Phone, Watch
Graph Sizes	Small, Medium, Large	Large
Graph Modularities	Low, High	Low, High
Repetitions	3	3
Participants	18	18
Total trials	2,916	1,944

Table 1. The design of the study trials.

[1] 3 (techniques) $\times$ 3 (tasks) $\times$ 4 (window sizes) $\times$ 3 (graph sizes) $\times$ 2 (modularities) $\times$ 3 (repetitions) $=$ 648 trials.

[2] The two studies were designed and conducted separately. *Study II* was designed following the suggestions from *Study I*'s participants.

Hypotheses

In the studies, we were interested in comparing the effectiveness of the plane display and two different focus+context displays in supporting large graph exploration. We believed that when the graph is small or the window is large, they would make little difference. However, when the size of the graph increases or the size of the window decreases, preserving the graph context properly becomes critical as it can guide users' navigation and help them find the target more efficiently.

As discussed previously, the three tested displays preserve different amounts of context in the periphery of the view ports in different ways. D_h has the highest distortion rate, in which the entire graph is shown in the display with a part in the center as the focus surrounded by the remaining graph suppressed as the context. D_s has a lower distortion rate, in which only the related part is shown as the context, surrounding the focal area. D_p has no distortion, showing the least amount of context. On one hand, a higher distortion rate increases the amount of context that can guide the data exploration. On the other hand, it also leads to more curved graph links and distorted graph structures, and unfavorably impacts interaction by triggering big visual changes, which may affect users' performance. Therefore, an effective graph exploration lies in balancing these two factors. In the following, we hypothesize the impact of different techniques (D_p, D_h, D_s in *Study I* and D_p, D_s in *Study II*) on graph exploration with different graph conditions and window sizes.

Among the three displays, D_h and D_s provide more contextual information (D_h the most) than D_p, thus better facilitating the node exploration task. Hence, we hypothesize:

H1(a): D_h is more effective than D_s and D_p in performing *T1* when graph size increases in *Study I*.

H1(b): D_s is more effective than D_p in performing *T1* when window size decreases in *Study II*.

Meanwhile, D_s shows more context than D_p and its links are less distorted when compared to D_h. Hence, we hypothesize that for the link and path exploration tasks, it will perform better than the other two techniques:

H2(a): D_s is more effective than D_h and D_p for *T2* when graph size increases in *Study I*.

H2(b): D_s is more effective than D_p for *T2* when window size decreases in *Study II*.

H3(a): D_s is more effective than D_h and D_p for *T3* when graph size increases in *Study I*.

H3(b): D_s is more effective than D_p for *T3* when window size decreases in *Study II*.

Besides graph size, in *T3*, users' performance will also be affected by graph topological structure: when the modular structure in the testing graph is clear, D_p will perform the best as it best preserves the structure without any distortion. When the modular structure is unclear, users will benefit from the focus+context displays with distortion and we believe D_s will perform the best given its moderate distortion rate:

H4: D_p is more effective than D_h and D_s in performing *T3* when modularity is high.

H5: D_s is more effective than D_h and D_p in performing *T3* when modularity is low.

USER STUDY

We conducted the user studies designed above with a total of 36 participants. In this section, we will describe the study procedure and results, and discuss our findings.

Participants and Apparatus

We recruited two groups of 18 volunteers to participate in *Study I* (12 males, 6 females; aging from 23 to 34, $M = 26.55$, $SD = 3.03$), and *Study II* (11 males, 7 females; aging from 19 to 25, $M = 20.44$, $SD = 1.54$), respectively. All participants were students or researchers in computer science with normal or corrected-to-normal vision.

The experiments were conducted on a laptop computer equipped with a mouse, a keyboard, and a 15.4-inch display with $1,440 \times 900$ pixels and 60 Hz refreshing rate. Graph visualizations were displayed in a window with a white background. Graph nodes were black dots. The size of each node reflected its degree, formally defined as $radius = 2 \times \log_2(degree + 1)$ pixels. Graph links were black lines of one pixel in width. Our study system supports picking and selection mechanisms [26] to assist users' exploration. In particular, when the mouse hovers over a node in the graph, the node and its adjacent links are highlighted in blue. When the mouse clicks on the node, the highlighting will persist until the next click. As suggested in [26], these interactive enhancements can help users focus on graph elements and reduce the frustrations caused by getting lost in a graph.

Procedure

At the beginning of each study, we introduced the tested graph exploration techniques to the participants and showed them how the same graph is displayed differently using each technique (Fig. 1). This helped them get familiar with the visualizations and get some intuition about how the graph display is distorted in each technique. Then, we explained the three graph exploration tasks to the participants and showed them how to perform the tasks using our study system. In particular, users can zoom and pan the display to navigate through a graph and can select a node to highlight its adjacent links. In each task, we guided users' exploration by highlighting a node whose neighbors needed to be explored (*T1*), two nodes whose common neighbors needed to be explored (*T2*), or a path whose containing nodes needed to be explored (*T3*).

Before the formal study, the users needed to perform practice trials to ensure a full understanding of all the tasks. The trials covered the three tasks and the tested graph exploration techniques, using three small-size graphs (128 nodes, $1,024$ links, 0.3 modularity). We encouraged the participants to ask questions and provided solutions of the practice trials.

After the training and practice, we conducted the formal study. We counterbalanced the order of the techniques using a 3×3 or 2×2 Latin Square in *Study I* and *Study II*, respectively. The order of the study trials for each technique was randomized. We reused the same set of graphs for all three techniques to ensure a fair comparison. We mirrored and rotated a graph before reusing it, which ensured that the participants were unable to memorize the correct answers. Each study trial had a 40-second limit and each formal study session took about 60 minutes. Participants can take a break after completing the trials for each technique. We conducted *Study I* and *Study II* separately at different times and *Study I* was conducted first.

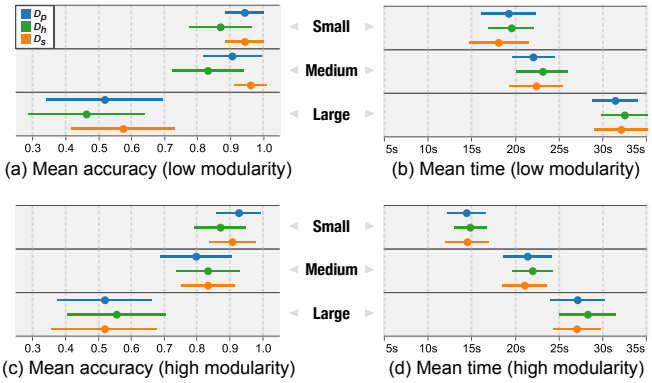

Figure 6. Node exploration task (T1) results of *Study I*. Error bars represent 95% confidence intervals (CIs).

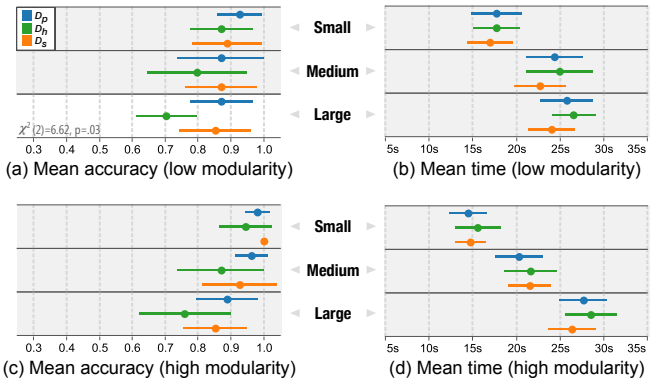

Figure 7. Link exploration task (T2) results of *Study I*. Error bars are 95% CIs. Rows with significant results are in a white background.

Collected Data and Statistical Analysis

We recorded the task completion time and accuracy (1 if the target node was found; otherwise, 0) to measure users' performance in each of the study tasks. We normalized the completion times with a log-transformation and applied the Repeated Measures ANOVA test to compare the means across multiple techniques, with paired t-test for pairwise comparisons. Since the accuracy results did not follow a normal distribution, we analyzed them using the Friedman test and pairwise Wilcoxon test. The normality of the data was determined by Shapiro-Wilk test. All tests used a significance level of .05.

Results of *Study I*

The study results were summarized in Fig. 6-8. We broke down the results by graph modularity (low = 0.3, high = 0.6) and graph size (small, medium, large).

T1: Node Exploration Task

Accuracy: There was no significant difference in accuracy among the displays in all the conditions (Fig. 6(a,c)).

Completion Time: We also did not find significant difference in task completion time among different techniques in all the testing conditions (Fig. 6(b,d)).

Overall, these results rejected **H1(a)**. The three displays had similar performances in *T1*.

T2: Link Exploration Task

Accuracy: In low-modularity condition (Fig. 7(a)), significant differences were detected when the graph size was large

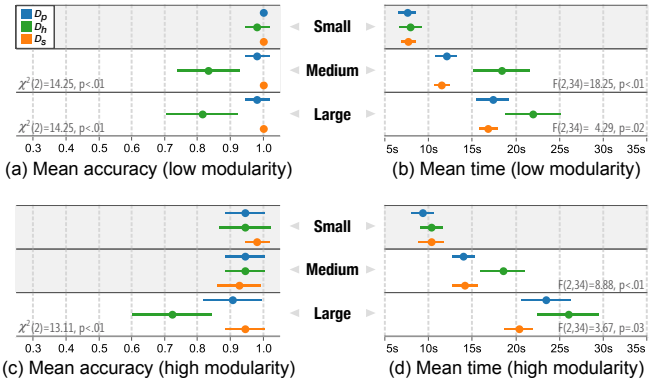

Figure 8. Path exploration task (T3) results of *Study I*. Error bars are 95% CIs. Significant rows are highlighted in a white background.

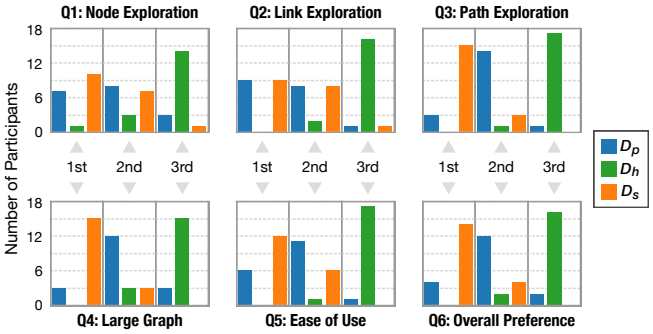

Figure 9. Ranking results for the three graph exploration techniques.

($\chi^2(2) = 6.62$, $p = .03$). Post-hoc analysis showed that D_s ($M = .85$) and D_p ($M = .87$) were significantly more accurate than D_h ($M = .70$). In high-modularity condition (Fig. 7(c)), the three displays had similar accuracies.

Completion Time: We did not find any significant difference in all the graph conditions in completion time.

Overall, D_s was more accurate than D_h in *T2* when the modularity was low and graph was large, partially supporting **H2(a)**. No significant differences were found between D_s and D_p.

T3: Path Exploration Task

Accuracy: As shown in Fig. 8(a,c), when the graphs were large, significant differences existed in both low ($M(D_p) = .98, M(D_h) = .81, M(D_s) = 1$; $\chi^2(2) = 14.25$, $p < .01$) and high ($M(D_p) = .90$, $M(D_h) = .72$, $M(D_s) = .94$; $\chi^2(2) = 13.11$, $p < .01$) modularity conditions. In the case of medium graphs, it only existed when the modular structure in the testing graphs was unclear with a low modularity ($M(D_p) = .98$, $M(D_h) = .83$, $M(D_s) = 1$; $\chi^2(2) = 14.25$, $p < .01$). No significance was found in all conditions with small graphs.

Completion Time: The completion time results were reported in Fig. 8(b,d). In the low-modularity condition, we found significant differences among the three displays when the graphs were medium ($M(D_p) = 12.03s$, $M(D_h) = 18.35s$, $M(D_s) = 11.49s$; $F_{2,34} = 18.25$, $p < .01$) or large ($M(D_p) = 17.35s$, $M(D_h) = 21.96s$, $M(D_s) = 16.81s$; $F_{2,34} = 4.29$, $p = .02$). In the high-modularity condition, significant differences also exist in graphs that were medium ($M(D_p) = 13.96s$,

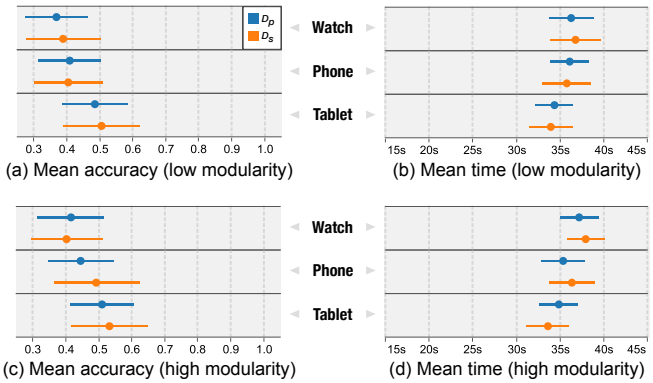

Figure 10. Node exploration task (T1) results of *Study II*. Error bars are 95% confidence intervals (CIs).

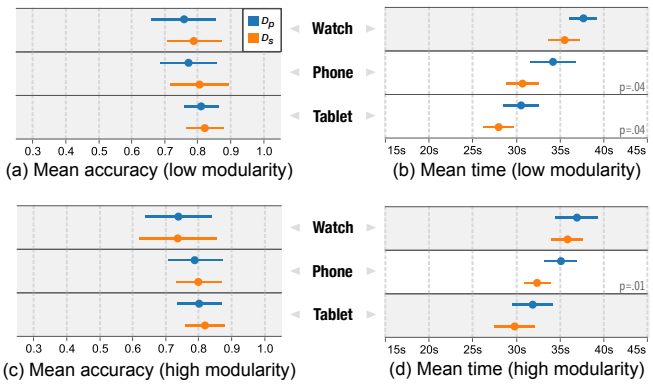

Figure 11. Link exploration task (T2) results of *Study II*. Error bars are 95% CIs. Rows with significant results are in a white background.

$M(D_h) = 18.51s$, $M(D_s) = 14.11s$; $F_{2,34} = 8.88$, $p < .01$) or large ($M(D_p) = 23.44s$, $M(D_h) = 26.03s$, $M(D_s) = 20.34s$; $F_{2,34} = 3.67$, $p = .03$). Post-hoc analysis showed that in both modularities D_s and D_p were significantly faster than D_h in medium graphs, and only D_s was significantly faster than D_h in large graphs.

Overall, D_s and D_p had better performances than D_h in *T3*, especially when the graph size increases. The above findings partially supported **H3(a)** and **H4-5**.

Post-Study Questionnaire

Users were asked to complete a questionnaire for ranking the three different displays according to their effectiveness in supporting (Q1) node exploration, (Q2) link exploration, (Q3) path exploration, and (Q4) large graph exploration as well as according to their (Q5) ease of use and (Q6) the users' overall preference, regarding the conditions tested in *Study I*. Overall, all the results ranked D_s as the best and D_h as the worst. The detailed results were summarized in Fig. 9.

Results of *Study II*

Fig. 10-12 summarized the results of *Study II*. We broke down the results by graph modularity (low = 0.3, high = 0.6) and window size (watch, phone, tablet), and controlled the graph size to be large (2,048 nodes, 16,384 links) in all study trials.

T1: Node Exploration Task

Accuracy: We found no significant difference in accuracy between D_p and D_s in all testing conditions (Fig. 10(a,c)).

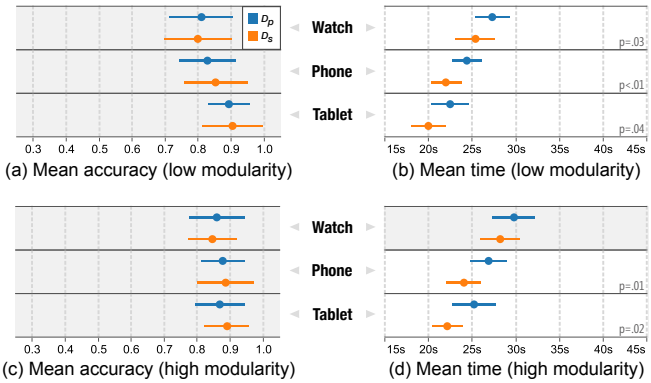

Figure 12. Path exploration task (T3) results of *Study II*. Error bars represent 95% CIs. Significant rows are highlighted in a white background.

Completion Time: No significant difference was found in task completion time in all testing conditions (Fig. 10(b,d)).

Overall, the results rejected **H1(b)** and showed D_p and D_s had similar performance in *T1* in windows of different sizes.

T2: Link Exploration Task

Accuracy: No significant difference in accuracy was observed between the two displays in all conditions (Fig. 11(a,c)).

Completion Time: When the graph modularity was low (Fig. 11(b)), the mean completion times of D_s were 30.71s and 27.99s when displayed in a window of phone size or tablet size, which was significantly faster than D_p whose mean times were 34.19s and 30.54s, respectively. In the condition of high modularity and phone-size window (Fig. 11(d)), D_s ($M = 32.37s$) also significantly outperformed D_p ($M = 35.10s$).

Overall, D_s was faster than D_p in *T2*, especially when displayed in smaller windows, which supported **H2(b)**.

T3: Path Exploration Task

Accuracy: We did not observe significant differences in accuracy in the path exploration task (Fig. 12(a,c)).

Completion Time: As shown in Fig. 12(b,d), when the graph modularity was low, study trials using D_s cost significantly shorter time than D_p for all window sizes: watch ($M(D_s) = 25.36s$, $M(D_p) = 27.34s$), phone ($M(D_s) = 22.00s$, $M(D_p) = 24.39s$), and Tablet ($M(D_s) = 19.90s$, $M(D_p) = 22.44s$). When the modularity was high, D_s also performed significantly better than D_p when using windows of phone ($M(D_s) = 24.02s$, $M(D_p) = 26.88s$) and tablet sizes ($M(D_s) = 22.11s$, $M(D_p) = 25.19s$).

Overall, these results indicated that D_s was faster than D_p in *T3* under both modularity conditions, which partially supported **H3(b)** and **H5**, but contradicted **H4**.

Post-Study Questionnaire

A post-study questionnaire was also conducted after *Study II*, in which we asked the users to select the more effective technique between D_p and D_s when they were displayed in windows of different sizes. More users chose D_s to be more effective for windows of watch size (10 out of 18) and phone size (11 out of 18). However, for windows of tablet size, 12 out of 18 users thought D_p was more effective.

DISCUSSION

Our user study results provided valuable insights into when, why, and how different focus+context techniques and the zoomable display are useful or have limitations in different situations, which will be discussed in this section.

Why was D_h considered as the worst technique in Study I?

In the questionnaire, D_h was considered as the least effective tool in supporting large graph exploration (Q4). One user mentioned that *"it is very difficult to traverse all neighbor nodes in hyperbolic layout because the relative distance between the nodes keeps changing."* Another user said *"the shape of the graph changes greatly even [after] applying minor panning interaction."* Some other users also complained that *"the [distorted] links confused me; it is hard to predict the direction I am moving to."* All these complaints were due to the same issue—the severe distortion in D_h that brought too much visual clutter, especially during the graph navigation. Just like some users suggested: *"Hyperbolic is poor for interactions; it might be useful if the graph is static (not interactive)."*

Why was D_s only conditionally superior than D_p in Study I?

It was a surprising finding in *Study I* that D_s showed no significantly better performance when compared to D_p. To investigate the reasons, we informally interviewed the users who ranked D_p as the most effective in Q1-3. All of them felt that *"tracing on a straight line is more intuitive and effective than tracing on a curve."* It seems that this benefit made up D_p's limitation of showing too little context when the exploration tasks were simple (e.g., *T1* and *T2*). However, according to the study results, D_p was not scalable when users were required to trace a long path. As shown in Fig. 8, there was a clear trend of dropping performance when the graph size increased. This was also confirmed by users' comments: *"It seems I need to spend more time on tracing on a long path in a large graph [on D_p]; when the path is too long, I cannot remember the previous candidates, ..., showing more context helps me to memorize."* Therefore, we believe D_s will outperform D_p when exploring a large graph in real world situations.

Why was D_s the most popular among users in Study I?

In *Study I*, although D_s only performed slightly better than D_p, it was a technique that provided much better user experience and favored by most of our users (16 out of 18). One user said that *"sphere's magnification [of the focal area] helps me see details at current focus; for the plane display, I need to zoom in to achieve the same level of detail, ..., for large dataset, the sphere looks less dense than the other two [techniques], so it makes the exploration easier."* Some users also mentioned that *"sphere shows more things in the limited space."* One user said *"[when using iSphere] less interactions were needed especially for finding common neighbors and tracking paths."* In addition, many users particularly favored the effect of the panning operation on D_s, which looked like sphere rotation and was considered to be easier and more interesting to operate by our users. The users also believed it is more efficient: *"The center view of the sphere automatically expands the distances between nodes; I only have to rotate the sphere without zooming in [to see the details]."*

Why did D_s outperform D_p on smaller windows in Study II?

After *Study II*, we informally interviewed the users who considered D_s to be more effective than D_p in the questionnaire questions. One commonly reported reason was that *"sphere needs less drag and zoom operations,"* as a user explained *"you need to drag and zoom a lot if the screen cannot show many nodes at a time, ..., sphere can show more nodes on the screen, so it requires less operations."* Another potential reason was raised that *"sphere looks less crowded than plane."* One user mentioned *"the sphere is less distracting because it makes the screen seem larger,"* and another said that *"it was more efficient to use sphere for tracing paths because it can avoid being too crowded around the paths you are tracing."* Besides, a user was concerned about large screens and commented that *"if all nodes can be displayed clearly, follow links in a plane display is easier because the lines are straight."*

When should iSphere (D_s) be used?

We believe when producing a focus+context display for exploring graphs, choosing a technique with a proper distortion rate is important. A high distortion rate will capture more context information but will also severely distort the information space, which significantly harms users' perception. According to the above user study and questionnaire results, we believe iSphere is a proper solution, which assists in exploring graphs especially for performing path-tracing related tasks when the screen is small or the graph is large and has unclear modular structures (i.e., when there are more details to memorize). Our findings suggested that iSphere can be used as a better alternative to the traditional plane based graph visualization techniques (D_p) as it has a similar or better performance, provides better user experiences, and has a better scalability.

CONCLUSION

In this paper, we have presented a novel focus+context technique, iSphere, to facilitate large graph exploration. Specifically, iSphere maps a node-link diagram onto a Riemann Sphere and orthogonally projects the sphere into a 2D plane. The produced focus+context view, similar to the hyperbolic display, has a focal area in the middle surrounded by context suppressed at the periphery of the circular display. We conducted two comprehensive controlled user studies to compare iSphere with two state-of-the-art baseline techniques, plane display and hyperbolic display, based on three graph exploration tasks. The user study results showed that iSphere significantly outperformed the hyperbolic display in path exploration tasks and the plane display in link and path exploration tasks in smaller windows. Based on the results, we have discussed when, why, and how different focus+context techniques and the zoomable display are useful or have limitations.

Our study results revealed great potentials of using iSphere in mobile devices. Besides, many promising future directions worth pursing, including (1) comparison of iSphere with other focus+context techniques in a wider range of graph exploration tasks, (2) study of iSphere when its graph links are not distorted and drawn as straight lines, and (3) study of the generalizability of iSphere and applying it to improve visualizations beyond graph explorers.

ACKNOWLEDGMENTS

Nan Cao is the corresponding author. We thank all the study participants and reviewers for their comments. This work is a part of the research supported by NSFC grant No.61602306, NSF grant No.1637067, DTRA grant: HDTRA1-16-0017, ARO grant: W911NF-16-1-0168, and the IBM SUR Award.

REFERENCES

1. James Abello, Stephen G Kobourov, and Roman Yusufov. 2004. Visualizing large graphs with compound-fisheye views and treemaps. In *International Symposium on Graph Drawing*. 431–441.
2. James Abello, Frank Van Ham, and Neeraj Krishnan. 2006. ASK-graphview: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 669–676.
3. Lars Valerian Ahlfors and Leo Sario. 2015. *Riemann Surfaces*. Princeton University Press.
4. James Anderson. 2006. *Hyperbolic Geometry*. Springer Science & Business Media.
5. Mark D Apperley, I Tzavaras, and Robert Spence. 1982. A bifocal display technique for data presentation. In *Proceedings of Eurographics*, Vol. 82. 27–43.
6. Michael Balzer and Oliver Deussen. 2007. Level-of-detail visualization of clustered graph layouts. In *6th International Asia-Pacific Symposium on Visualization*. 133–140.
7. Patrick Baudisch, Nathaniel Good, and Paul Stewart. 2001. Focus plus context screens: Combining display technology with visualization techniques. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*. 31–40.
8. Alan F Beardon and Ch Pommerenke. 1978. The Poincaré metric of plane domains. *Journal of the London Mathematical Society* 2, 3 (1978), 475–483.
9. Benjamin B Bederson, James D Hollan, Ken Perlin, Jonathan Meyer, David Bacon, and George Furnas. 1996. Pad++: A zoomable graphical sketchpad for exploring alternate interface physics. *Journal of Visual Languages & Computing* 7, 1 (1996), 3–32.
10. Nan Cao, Yu-Ru Lin, Liangyue Li, and HangHang Tong. 2015. g-Miner: Interactive visual group mining on multivariate graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 279–288.
11. Nan Cao, Jimeng Sun, Yu-Ru Lin, David Gotz, Shixia Liu, and Huamin Qu. 2010. FacetAtlas: Multifaceted visualization for rich text corpora. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 1172–1181.
12. Duen Horng Chau, Aniket Kittur, Jason I Hong, and Christos Faloutsos. 2011. Apolo: Interactive large graph sensemaking by combining machine learning and visualization. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 739–742.
13. Andy Cockburn, Amy Karlson, and Benjamin B Bederson. 2009. A review of overview+detail, zooming, and focus+context interfaces. *Comput. Surveys* 41, 1 (2009), 2:1–2:31.
14. Aurélie Cohé, Bastien Liutkus, Gilles Bailly, James Eagan, and Eric Lecolinet. 2016. SchemeLens: A content-aware vector-based fisheye technique for navigating large systems diagrams. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 330–338.
15. Harold Scott Macdonald Coxeter. 1969. *Introduction to Geometry*. Wiley.
16. Andrej Cvetkovski and Mark Crovella. 2016. Multidimensional scaling in the Poincaré disk. *Applied Mathematics & Information Sciences* 10, 1 (2016), 125–133.
17. Arnold Douglas and Jonathan Rogness. 2009. Möbius transformations revealed. *Notices of American Mathematical Society* 55, 10 (2009), 1226–1231.
18. Juan C. Dürsteler. 2002. Focus+context. www.infovis.net/printMag.php?num=85&lang=2. (2002). [Online; accessed 1-Sept-2016].
19. Peter Eades and Qing-Wen Feng. 1997. Multilevel visualization of clustered graphs. In *Graph Drawing*. 101–112.
20. Niklas Elmqvist and Jean-Daniel Fekete. 2010. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics* 16, 3 (2010), 439–454.
21. Werner Fenchel. 1989. *Elementary Geometry in Hyperbolic Space*. Walter de Gruyter.
22. G W Furnas. 1986. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 16–23.
23. George W Furnas. 2006. A fisheye follow-up: Further reflections on focus+context. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 999–1008.
24. Emden R Gansner, Yehuda Koren, and Stephen North. 2005a. Graph drawing by stress majorization. In *Graph Drawing*. 239–250.
25. Emden R Gansner, Yehuda Koren, and Stephen C North. 2005b. Topological fisheye views for visualizing large graphs. *IEEE Transactions on Visualization and Computer Graphics* 11, 4 (2005), 457–468.
26. Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. 2004. A comparison of the readability of graphs using node-link and matrix-based representations. In *IEEE Symposium on Information Visualization*. 17–24.
27. Kasper Hornbæk, Benjamin B. Bederson, and Catherine Plaisant. 2002. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM Transactions on Computer-Human Interaction* 9, 4 (2002), 362–389.
28. Takao Ito, Kazuo Misue, and Jiro Tanaka. 2009. Sphere anchored map: A visualization technique for bipartite graphs in 3D. In *International Conference on Human-Computer Interaction*. 811–820.
29. Naftali Kadmon and Eli Shlomi. 1978. A polyfocal projection for statistical surfaces. *The Cartographic Journal* 15, 1 (1978), 36–41.

30. P. Karnick, D. Cline, S. Jeschke, A. Razdan, and P. Wonka. 2010. Route visualization using detail lenses. *IEEE Transactions on Visualization and Computer Graphics* 16, 2 (2010), 235–247.
31. Stephen G Kobourov and Kevin Wampler. 2005. Non-Euclidean spring embedders. *IEEE Transactions on Visualization and Computer Graphics* 11, 6 (2005), 757–767.
32. Matthias Kreuseler, Norma Lopez, and Heidrun Schumann. 2000. A scalable framework for information visualization. In *IEEE Symposium on Information Visualization*. 27–36.
33. Oh-Hyun Kwon, Chris Muelder, Kyungwon Lee, and Kwan-Liu Ma. 2015. Spherical layout and rendering methods for immersive graph visualization. In *IEEE Pacific Visualization Symposium*. 63–67.
34. Eric LaMar, Bernd Hamann, Kenneth Joy, and others. 2001. A magnification lens for interactive volume visualization. In *Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*. 223–232.
35. John Lamping, Ramana Rao, and Peter Pirolli. 1995. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 401–408.
36. Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. 2006. Task taxonomy for graph visualization. In *Proceedings of the AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*. 1–5.
37. Ying K Leung and Mark D Apperley. 1994. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction* 1, 2 (1994), 126–160.
38. Wanchun Li and Masahiro Takatsuka. 2004. Adding filtering to geometric distortion to visualize a clustered graph on small screens. In *Proceedings of the Australasian Symposium on Information Visualisation*, Vol. 35. 71–79.
39. Zhiyuan Lin, Nan Cao, Hanghang Tong, Fei Wang, U Kang, and Duen Horng Polo Chau. 2013. Demonstrating interactive multi-resolution large graph exploration. In *IEEE 13th International Conference on Data Mining Workshops*. 1097–1100.
40. Kwan-Liu Ma and Chris W Muelder. 2013. Large-scale graph visualization and analytics. *Computer* 46, 7 (2013), 39–46.
41. Tomer Moscovich, Fanny Chevalier, Nathalie Henry, Emmanuel Pietriga, and Jean-Daniel Fekete. 2009. Topology-aware navigation in large networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2319–2328.
42. Tamara Munzner. 1998. Exploring large graphs in 3D hyperbolic space. *IEEE Computer Graphics and Applications* 18, 4 (1998), 18–23.
43. M. E. J. Newman. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* 103, 23 (2006), 8577–8582.
44. Ken Perlin and David Fox. 1993. Pad: An alternative approach to the computer interface. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. 57–64.
45. Pratha Sah, Lisa Singh, Aaron Clauset, and Shweta Bansal. 2014. Exploring community structure in biological networks with random graphs. *BMC Bioinformatics* 15, 1 (2014), 220.
46. Manojit Sarkar and Marc H. Brown. 1994. Graphical fisheye views. *Commun. ACM* 37, 12 (1994), 73–83.
47. Lei Shi, Nan Cao, Shixia Liu, Weihong Qian, Li Tan, Guodong Wang, Jimeng Sun, and Ching-Yung Lin. 2009. Himap: Adaptive visualization of large-scale online social networks. In *IEEE Pacific Visualization Symposium*. 41–48.
48. Christian Tominski, James Abello, and Heidrun Schumann. 2009. CGV—an interactive graph visualization system. *Computers & Graphics* 33, 6 (2009), 660–678.
49. C Tominski, S Gladisch, U Kister, R Dachselt, and H Schumann. 2016. Interactive lenses for visualization: An extended survey. *Computer Graphics Forum* (2016).
50. Frank Van Ham and Jarke J van Wijk. 2004. Interactive visualization of small world graphs. In *IEEE Symposium on Information Visualization*. 199–206.
51. Tatiana Von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke J van Wijk, Jean-Daniel Fekete, and Dieter W Fellner. 2011. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum* 30, 6 (2011), 1719–1749.
52. w3schools. 2015. Browser display statistics. `w3schools.com/browsers/browsers_display.asp`. (2015). [Online; accessed 1-Sept-2016].
53. Jörg A Walter. 2004. H-MDS: A new approach for interactive visualization with multidimensional scaling in the hyperbolic space. *Information Systems* 29, 4 (2004), 273–292.
54. InfoVis Wiki. 2014. Semantic zoom. `infovis-wiki.net/index.php/Semantic_Zoom`. (2014). [Online; accessed 1-Sept-2016].
55. Harold E Wolfe. 2012. *Introduction to Non-Euclidean Geometry*. Dover Publications.
56. Yingcai Wu, Xiaotong Liu, Shixia Liu, and Kwan-Liu Ma. 2013. ViSizer: A visualization resizing framework. *IEEE Transactions on Visualization and Computer Graphics* 19, 2 (2013), 278–290.
57. Kai Xu, Chris Rooney, Peter Passmore, Dong-Han Ham, and Phong H Nguyen. 2012. A user study on curved edges in graph visualization. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2449–2456.