DOI: 10.1111/cgf.12381
Eurographics Conference on Visualization (EuroVis) 2014
H. Carr, P. Rheingans, and H. Schumann
(Guest Editors)

Volume 33 (2014), Number 3

A Gaze-enabled Graph Visualization to Improve Graph Reading Tasks

Mershack Okoe, Sayeed Safayet Alam, and Radu Jianu

Florida International University, USA

Abstract

Performing typical network tasks such as node scanning and path tracing can be difficult in large and dense graphs. To alleviate this problem we use eye-tracking as an interactive input to detect tasks that users intend to perform and then produce unobtrusive visual changes that support these tasks. First, we introduce a novel fovea based filtering that dims out edges with endpoints far removed from a user's view focus. Second, we highlight edges that are being traced at any given moment or have been the focus of recent attention. Third, we track recently viewed nodes and increase the saliency of their neighborhoods. All visual responses are unobtrusive and easily ignored to avoid unintentional distraction and to account for the imprecise and low-resolution nature of eye-tracking. We also introduce a novel gaze-correction approach that relies on knowledge about the network layout to reduce eye-tracking error. Finally, we present results from a controlled user study showing that our methods led to a statistically significant accuracy improvement in one of two network tasks and that our gaze-correction algorithm enables more accurate eye-tracking interaction.

Keywords: Eye tracking, gaze contingent graph visualization.

1. Introduction

Network analysis plays an important part in domains such as neuroscience [BS09], genomics and proteomics [CCNS08], software engineering [GN00], or social sciences [BMBL09]. Interaction is instrumental in allowing users to weed through the scale, complexity, and clutter inherent to visualizations of real-life networks. Here we explore the use of eye tracking as an interactive input to detect users' intentions and support them by slight changes in the visualization. The use of eye tracking as an input has been explored in the human computer interaction (HCI) community [Duc02], but there are few results in the visualization domain.

Specifically, we introduce three types of interactions. First, we reduce clutter by using a novel fovea-based filtering that dims edges that pass through the user's view focus but have their endpoints far outside of the user's fovea. Second, we increase the saliency of edges that users are viewing or have recently viewed. Third, we keep track of nodes that were recently viewed and increase the salience of their neighborhood. All visual responses are gradual, incremental rather than binary, and visually subtle.

Thus, by design, our interactions are gaze-contingent [Duc02]. We use gaze coordinates to infer users' task intentions and to visually support these tasks as unobtrusively as possible, so as to minimize distraction. This approach also relates to attentive interfaces [Duc02, Sel04] and multimodal interfaces [Ovi03] but contrasts with early HCI efforts to use eye-tracking in ways analogue to manual pointing and clicking. Merely connecting eye-tracking input to otherwise conventional network interactions is limited by particularities of eye-movements and eye-tracking technology. Specifically, as noted by [ZMI99], the eyes are not a control organ, eye-tracking input is generally low resolution and inaccurate, and the absence of a trigger command is difficult to compensate [Jac90].

We also contribute a gaze-correction algorithm that uses knowledge of the visualization layout to reduce eye-tracking error. Insufficient calibration sometimes leads to screen regions in which gaze input is offset from the users' real viewing point. Our algorithm relies on the known visual positioning of nodes on the screen to detect nodes that are likely to be viewed.

We evaluated our gaze-enabled network visualization in a

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

within-subject user study with twelve participants. First, we asked participants to perform two types of tasks: (i) identify whether there is a direct connection between two nodes; and (ii) identify the shortest path between two nodes. In a third task designed to evaluate our gaze correction algorithm, users selected as many nodes as possible in a given time by looking at them. Our results showed a 30% improvement in the direct connection task ($p = 0.02$), a 25% improvement in the node selection task ($p = 0.01$), and were not significant in the path task.

Given the unavoidable connection between eyes and data visualization, the fact that people's gazes are linked to tasks they are performing [YR67, SCC13], and that eye-tracking is on its way to becoming a component of regular work stations [Duc07, JK03], we hypothesize that visualization research can benefit from exploring the use of eye-tracking as an input channel.

2. Related Work

The fovea, a small area in the center of the retina, is responsible for our high resolution vision. The larger part of our field of view (i.e. parafoveal and peripheral region) is low resolution. The illusion of full high definition vision is created by an unconscious scanning process: the fovea performs quick translations, called saccades between short moments of focus, called fixations. Eye-tracking technology allows us to locate users' points of gaze [WM87, Jac91].

Most often, gaze tracing is used for data collection in offline, post hoc analyses of human visual perception [Duc07]. In data visualization, Huang et al. used eye tracking to investigate the cognitive processes involved in reading graph visualization [HEH08], Pohl et al. used it to understand how user performance is affected by network layout [PSD09], Burch et al. investigated visual exploration behavior and task solution strategies for hierarchical tree layouts [BAA*13, BKH*11], and Tory et al. used eye tracking to analyze the effectiveness of visualization designs that combine 2D and 3D views [TAK*05]. In a different approach, Andrienko et al. identified visual analytics methods applicable to eye tracking data analysis [AABW12], while Steichen et al. notes user and task characteristics that can be inferred from eye tracking data [SCC13]. Unlike these works, we use eye tracking data as an input source to change visualizations in real time.

The appeal of the eye's speed led HCI researchers to explore gaze as an actuatory input in ways analogue to manual input. This approach has met with limited success due to several reasons. First, while very fast, gaze-input comes with disadvantages such as low accuracy, jitter, drift, offsets, and calibration needs [Duc07, JK03, KPW07]. Second, finding a gaze equivalent of a trigger command is not trivial and leads to the "Midas touch" phenomenon - the inability of the interface to reliably distinguish between looking and controlling [Jac91]. Ultimately, the duration of a fixation, or dwell time, has been established as the most effective way to trigger commands [WM87, Jac91]. However, low dwell thresholds amplify the Midas touch problem by triggering commands inadvertently, while high dwell thresholds offset the speed advantage of gaze input.

The current consensus is that eyes are not suited for interface control [Jac91, JK03, ZMI99, Zha03]. Instead, Jacob proposed that interfaces should use gaze as an indicator of intention and should react with gradual, unobtrusive changes [Jac91, JK03], a view formalized by the concept of attentive interfaces [ADS05, Ver02, VSCM06, VS08, HMR05, RHN*03]. The research described here aligns with this paradigm and also draws inspiration from work in gaze-contingent rendering [OHM*04, DÇ07, ODH02], where scenes are drawn in high resolution only in foveated screen areas to reduce computational costs.

In the visualization domain the use of eye tracking as an interactive input is minimal. Streit et al. [SLMS09] use gaze-information to enlarge visualization regions of interest and to navigate or manipulate 3D scenes. This work fits in the HCI control paradigm. Our work differs through the adoption of the attentive interface approach by which we produce unobtrusive visual responses that minimize distraction and are complementary to traditional manual control. A further contribution over previous work is the gaze correction method described in the following section.

3. Implementation

Our implementation focuses on two issues: improving gaze accuracy and providing interactive visual responses. The interactive responses are: (i) a novel fovea based filtering that dims out edges with endpoints far removed from a user's view focus; (ii) highlighting edges that are being traced at any given moment or have been the focus of recent attention; (iii) tracking recently viewed nodes and increasing the saliency of their neighborhoods. We detail these techniques in the following sections.

3.1. Gaze-correction

Due to calibration limitations, gazes reported by eye-tracking are sometimes offset from real gaze coordinates. We alleviate this problem by leveraging the known network layout. We use the eye-tracker API to compute fixations from individual gaze samples. We match subsequent long fixations (200-300*ms*) to proximal nodes that have a relative pairwise positioning similar to that of the fixations. We then assume these nodes were likely the target of the user's attention and compute offsets between them and the fixations. We aggregate these offsets over time, gradually constructing and adjusting an offset map over the screen space. This offset map is then used to correct the coordinates of all incoming gaze-samples (Fig. 1).

The specific implementation is shown in Algorithm 1. For

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

the last three fixations delivered by the eye-tracker API, we find the closest three network nodes to each fixation and we construct 27 possible combinations. We then find the combination that minimizes a score that averages two components: proximity of the gazes to their corresponding nodes; and how well the relative positioning of the nodes matches the relative positioning of the fixations. We consider that the current three fixations map to those three nodes. We then integrate the offset vectors into the existing offset map. The map is a partitioning of the screen space into cells of 10×10 pixels, where each cell contains a two-dimensional offset vector that will be applied to gazes landing in that cell. Fig. 2 illustrates the results of the algorithm.

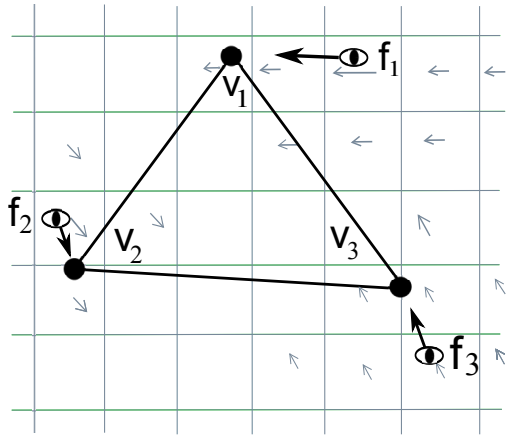

Figure 1: *Gaze correction: fixations $f_{1..3}$ do not perfectly overlap the nearest vertices $v_{1..3}$ but their relative positioning matches that of the vertices. We conclude that $f_{1..3}$ were fixations on $v_{1..3}$. We compute displacement offsets between f and v and incorporate them into a grid of displacement vectors that we apply to all gaze samples.*

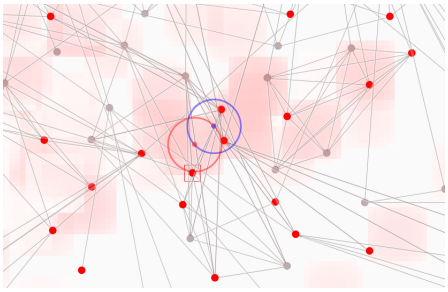

Figure 2: *Gaze correction in our system. The blue circle represents the corrected gaze while the red one matches the raw gaze sample. The magnitude of vectors in the offset map is displayed as a red heatmap.*

We also note that correction happens in screen space rather than visualization space and as such the computed offset map is bounded in size. This means it can be computed relatively quickly and then adjusted on the fly as users interact with the visualization.

Algorithm 1 Correct Gaze

Require: $f_{1..3}$, three most recent fixations
(w,h), size of display window
$cells[w/10,h/10]$, grid over screen space
$v_{i,1..3} \leftarrow$ closest three vertices to f_i, $i = 1..3$
$v \leftarrow []$, $minScore \leftarrow \infty$
for $(j,k,l)|j = 1..3,k = 1..3,l = 1..3$ **do**
$score \leftarrow avg(|f_1 - v_{1,j}|,|f_2 - v_{2,k}|,|f_3 - v_{3,l}|) \times$
$avg(|(f_1 - f_2) - (v_{1,j} - v_{2,k})|,$
$|(f_1 - f_3) - (v_{1,j} - v_{3,l})|,$
$|(f_2 - f_3) - (v_{2,k} - v_{3,l})|)$
if $score < minScore$ **then**
$score \leftarrow minScore$, $v \leftarrow [v_{1,j}, v_{2,k}, v_{3,l}]$
end if
end for
for $i|i = 1..3$ and $score < threshold$ **do**
$disp \leftarrow v[i] - f_i$
$(cx,cy) \leftarrow$ cell of f_i
for $(x,y)|cx-5 < x < cx+5, cy-5 < cy+5$ **do**
$d \leftarrow (|(cx,cy) - (x,y)|)$
$cell[x,y] \leftarrow disp/(d+1)$
end for
end for
STATE apply cell[x,y] to all gaze samples landing in that cell
**all distance are computed in screen space

We note that our approach resembles work by Salvucci et al. [SA00] who use Markov and Bayesian models to predict gaze targets based on probable behavior, to that of MacKenzie and Zhang [MZ08] who use letter and word prediction to improve their eye-typing system, and finally to work interpreting fixations as part of "gaze gestures" [DS07,DDLS07].

3.2. Gaze-enabled network interactions

We implemented three types of interactions. The next three sub-sections describe a series of node and edge scores that are computed from gaze data. The fourth sub-section describes how these scores are combined and used during rendering to create visual responses. The last sub-section discusses a few implementation details that apply to all interactions.

3.2.1. Gaze-enabled filtering

Gaze-enabled edge filtering dims edges that pass through the user's fovea but have both endpoints far removed from it. To achieve this, given a current gaze point, we compute filtering scores (S_{ef}) for all edges according to the diagram in Figure 3 and using Formula 1. Specifically, if the edge's closest endpoint to the gaze is within the smaller circle then

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

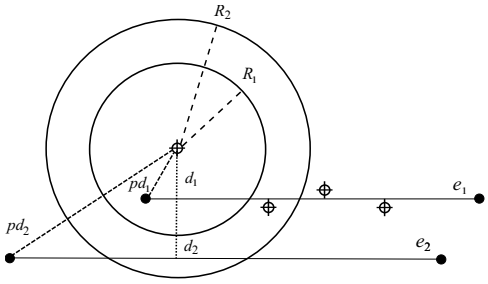

Figure 3: *Edges running through users' fovea but with both endpoints far away from its center are filtered out.*

the score will be 1 ($f(pd) = 1$) and the edge visible. If the edge runs through the user's fovea (i.e., d is small) and the closest endpoint is not within view ($pd > R2, f(pd) = 0$) the score is null and the edge filtered. The two circles centered at the user's focus point create a gradual transition between those two cases.

Current edge scores are combined with previous scores to ensure gradual changes (1..2 seconds) in this measure. Finally, S_{ef} is combined with other saliency measures and used to render edges , as will be described in section 3.2.4.

$$\begin{aligned} S_{ef} &= f(pd) + (1 - f(pd)) \times min(1, \frac{d}{R_2}) \\ \text{where } f(pd) &= max(0, min(1, 1 - \frac{pd - R_1}{R_2 - R_1})), \\ pd &= |\text{gaze} - \text{closest edge endpoint}| \\ d &= |\text{gaze} - \text{edge}| \end{aligned} \quad (1)$$

3.2.2. Detecting viewed edges

We also aim to detect edges as users are visually tracing them and increase their saliency. To this end we compute edge viewing scores (S_{ev}).

We first divide edges into segments of equal lengths. A score is maintained for segment endpoints to indicate whether recent gazes landed nearby. Each gaze sample landing close to an edge segment endpoint will increase the endpoint's score by a factor inversely proportional to the distance between the gaze and the endpoint. At the same time all scores are gradually decreased each time a new gaze is processed.

This essentially creates a "histogram" along edges. Figure 4 shows a segmented edge with histogram bars arranged horizontally. To compute the edge viewing score, we sum over the histogram, divide by a constant, in our case 500, and cap the value to 1. This step ensures edge scores are between 0 and 1 and gives preference to edges close to or longer than 500 pixels.

An improvement was introduced to account for a behavior observed during testing. People seemed to require shorter fixations and longer saccades when tracing edges that were fairly isolated or travelling through empty space, but required longer fixations with shorter saccades between them when tracing edges in dense areas. To account for this, we compute a density score for each segment endpoint by adding up the number of other endpoints that lie within a certain distance from it. We use this density score to extend longer bars from low density endpoints and shorter bars from high density endpoints. This ensures that we easily detect views of isolated edges yet at the same time reduce false positives in dense areas where two random fixations could easily match an edge.

Figure 4: *Viewing histograms along edges are computing by dividing the edge in segments and monitoring gaze-count in each segment.*

Two types of edge viewing scores are computed using this methodology. The first is a short-term score (S_{ev}) that captures edges that are currently viewed. This score can change between its minimum and maximum values within a few hundred milliseconds. The second is a long-term score (S'_{ev}) that captures edges of interest, those that have been viewed repeatedly in the last several seconds. We use the first score to increase or decrease the value of the second score by a constant that depends on the desired life-span of the second score. The process of computing S_{ev} and S'_{ev} is formalized in Formula 2.

$$\begin{aligned} &Seg_{e,k}, \text{ segment } k \text{ of edge } e \\ Density(Seg_{e,k}) &= count(Seg_{e',k'}) \text{ where } e' \text{ != } e \text{ and} \\ &|Seg_{e',k'} - Seg_{e,k}| < 100px \\ Score(Seg_{e,k}) &= \frac{1}{|Gaze - Seg_{e,k}|} \\ S_{ev}(e) &= sum(Score(Seg_{e,k}) \times \frac{1}{Density(Seg_{e,k})}) \\ S'_{ev}(e) &= S'_{ev}(e) + (S_{ev}(e) - 0.5)/10, \\ &S'_{ev} \text{ capped to [0,1]} \end{aligned} \quad (2)$$

3.2.3. Detecting sub-networks of interest

We also increase the saliency of nodes of current interest and their neighborhoods. To achieve this, we first compute an interest score for each node in a way analogue to edge segment endpoint scores. If a user's gaze lingers close to a node, its score will be increased by a factor inversely proportional to the distance between the fixation and the node. As for edges, we decay all node scores each time a new gaze is processed. Once node scores are updated according to a new gaze, we

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

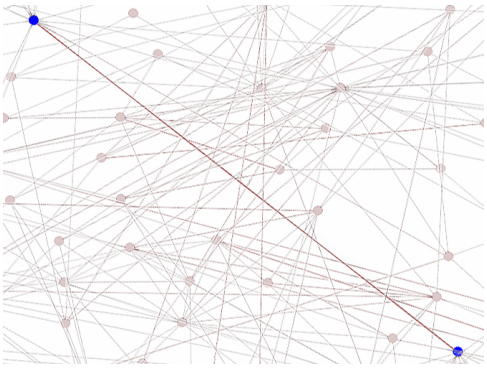

Figure 5: *Viewed edge highlighting. An edge visually traced by the user is highlighted (red).*

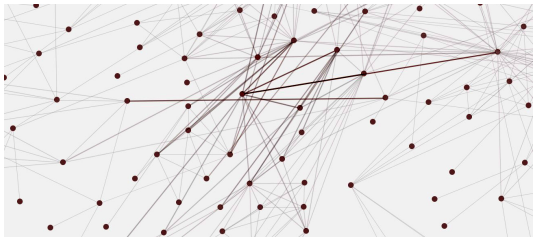

Figure 6: *Highlighting a sub-network of interest. Recently viewed nodes and edges between them become slightly highlighted. The effect is also diffused to their immediate neighborhoods.*

diffuse them across their neighborhood using the process illustrated in Algorithm 2. Similar to edge scores, we keep two scores for each node, a short term score S_v computed as described above and a long term score S'_v that is computed from S_v. The process of computing S_v and S'_v is formalized in Formula 3.

$$\begin{aligned} S_v(n) &= \frac{1}{|Gaze - n|} \\ S'_v(n) &= S'_v(n) + (S_v(n) - 0.5)/10, \\ & S'_v \text{ capped to } [0,1] \end{aligned} \tag{3}$$

Algorithm 2 Diffuse node scores

```
do three times
for all edges e(s,d) do
    score(s) ← max(score(s), score(d)/4)
    score(d) ← max(score(d), score(s)/4)
end for
```

3.2.4. Rendering

On rendering, we combine the previously described scores and link them to visual properties such as color and alpha blending. In our implementation we have used the mapping described below. However, other mappings can be explored as long as they are gradual and unobtrusive.

First, node scores S_v and S'_v are factored into the node color and opacity using formula 4. The constants R,G,B,A determine the base color for unviewed nodes, the saliency for viewed nodes, and weight of each score into the visual response. The values chosen in our evaluated implementation, given color components between 0 and 255, were $R_{base} = 50$, $G_{base} = 50$, $B_{base} = 50$, $A_{base} = 50$, $R_{Sv} = 30$, $R_{S'v} = 90$, $A_{Sv} = 30$, $A_{S'v} = 120$.

$$\begin{aligned} RGB(v) &= (R_{base} + S_v \times R_{Sv} + S'_v \times R_{S'v}, \\ & \qquad G_{base}, B_{base}) \\ Alpha(v) &= A_{base} + S_v \times A_{Sv} + S'_v \times A_{S'v} \\ & R_{base} + R_{Sv} + R_{S'v} \leq 255 \\ & A_{base} + A_{Sv} + A_{S'v} \leq 255 \end{aligned} \tag{4}$$

Similarly, when drawing edges we also varied opacity and the red color component to highlight interesting edges, as shown in formula 5. Moreover, we rendered edges in two layers: a short-term layer that highlights edges currently viewed and a long-term layer to highlight edges that have been of interest in the last several seconds or more.

$$\begin{aligned} RGB(u,v) &= (R_{base} + S_{ev} \times R_{Sv} + S'_{ev} \times R_{S'ev} \\ & \quad + S_{endp} \times R_{endp}, G_{base}, B_{base}) \\ Alpha(u,v) &= S_{ef} \times (A_{base} + S_{ev} \times A_{Sev}) \\ & \quad + (1 - S_{ef}) \times S_{ev} \times A_{Sev} \\ Alpha'(u,v) &= (S_{ef} + (1 - S_{ef}) \times A_{notef}) \\ & \quad \times (S'_{ev} \times A_{S'ev} + S_{endp} \times A_{endp}) \\ & \text{where } S_{endp} = \frac{(S'_u + S'_v)}{2} \\ & \text{and } R_{base} + R_{Sv} + R_{S'ev} + R_{endp} \leq 255 \\ & \qquad A_{base} + A_{Sev} \leq 255 \\ & \qquad A_{S'ev} + A_{endp} \leq 255 \end{aligned} \tag{5}$$

In the color computation, R_{base} indicates the base component of unviewed edges (125 in our implementation). R_{Sv},

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

$R_{S'ev}$, and R_{endp} represent weights given to the three scores they precede.

Blending edge filtering with edge interest scores is not trivial. If edge filtering scores are simply multiplied to the interest scores, then a user tracing a long edge may find that the edge disappears while their gaze reaches the edge midpoint. Thus, viewed edges should be "exempt" from filtering.

To achieve this, the computed alpha values combine two components: what happens when edges are not filtered ($S_{ef} = 1$) and what happens when they are filtered ($1 - S_{ef} = 1$). Thus, for short term alphas ($Alpha$), in the case of non-filtered edges, we combine a base component (A_{base}) with a component determined by (S_{ev}). In the case of filtered edges the score is determined by S_{ev}. As such viewed edges will be visible even when they are filtered to a degree determined by A_{Sev}.

The long-term alpha ($Alpha'$) does not have a base component since the highlighting created by $Alpha'$ is only visible for edge of interest. $A_{S'ev}$ and A_{endp} are weights of the two scores they precede. Finally, analogue to A_{Sev} in the previous paragraph, A_{notef} indicate the degree to which filtered out edges should be highlighted if they are of interest.

3.3. Implementation Notes

Our gaze correction relies on fixations provided by the eye-tracking API. These fixations are computed by the API from a stream of gaze-samples acquired at a frequency of 120Hz. As such, they are a discretization of the actual data stream. Conversely, the gaze interactions in section 3.2 work directly with individual gaze samples. These small but frequent bits of information are aggregated over different time scales to indicate whether nodes and edges are being viewed. The advantage of this continuous approach is that responses can be gradual rather than binary, that errors are smaller and less noticeable, and that visual responses can be produced while a fixation is in progress.

Distance thresholds involved in gaze-based interactions are defined in screen space. When computations are done in visualization space, these thresholds are adjusted by the zoom level. For example, the 500 pixel threshold mentioned in section 3.2.2 refers to lengths on the screen rather than lengths in visualization space. Thus, since edge "histogram" values are computed in model space, we adjust the 500 pixel threshold by a factor proportional to the zoom level.

All data related to viewing scores are maintained in data objects – nodes and edges. As such, interactions that change the network layout or the viewpoint do not impact the gaze-interaction in any way. The data objects will continue aggregating viewing scores computed based on the new configuration, and the visualization will evolve smoothly.

Finally, while computations may seem complex, they generally involve simple searches and run in linear time. Our current implementation processes gaze coordinates at 120Hz for the data described in section 4.1 without lag. Moreover, significant optimizations could be made to accommodate larger datasets. Since all computations are a function of gaze coordinates, decomposing the visualization using a quad-tree structure or even a simple grid would enable fast retrieval of closest nodes, edges, and other network properties. Such structures would need updating upon interactions that affect the graph layout but are likely to be feasible within timeframes typical for manual input.

4. Evaluation

The study was designed with two goals in mind: (1) test the potential of gaze interactions for improving network tasks and (2) demonstrate the effectiveness of the gaze correction algorithm. To this end we tested two hypotheses: (H1) network data reading tasks are aided by gaze enabled interactions in terms of speed and accuracy; (H2) the gaze correction algorithm increases the system's accuracy in detecting intended view-targets.

4.1. Study Design

We performed a within-subjects user study to evaluate our gaze-enabled network visualization (eye tracking condition) against the same visualization without gaze interaction (control condition). The study lasted approximately one hour and involved 12 participants.

We first tested H1 by asking users to perform two tasks in one condition and then again in the other condition. To reduce learning effects, half of the participants started the study in the eye-tracking condition while the other half started with the control condition. A three minute break was introduced between conditions. We then tested H2 by asking users to perform a third task with and without eye-tracking support. Again, the two conditions were alternated.

The actual study was preceded by a training session. First, we familiarized subjects with the concepts of node-link diagrams and the tasks they were going to complete in a short whiteboard presentation. They were then shown the visualization, several instances of tasks with correct answers, and were allowed to use the controls to provide answers and advance through the study. Users were also informed on how the gaze-enabled visualization reacts to their view.

A questionnaire at the end of the study captured users' preference between eye tracking and conventional visualization, a $1-5$ rating of eye tracking appeal (1=not-appealing, 5=very-appealing), a $1-5$ rating of the helpfulness of eye tracking (1=not-helpful, 5=very-helpful), and an indication of whether the visual responses were obtrusive or not (yes/no).

The design of the study was informed by a smaller scale

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

study described in [OAJ13] and a small pilot study with two users that helped remove a few errors of implementation.

Participants: We recruited 12 participants, 9 male and 3 female, most of which were graduate students in our department. Their ages ranged between 24 and 30 years. None of the participants reported vision deficiencies or color blindness. Reimbursement was set at $10 with an additional $5 awarded to the user with the best aggregated accuracy.

Apparatus and Calibration: We used a RED120HZ eye-tracker from Sensory Motor Instruments (SMI). The eye tracker was calibrated for each subject at the beginning of the eye tracking condition using nine-point calibration. Calibration was considered successful when differences between reported location and view target was at most 0.5cm.

Stimuli: The dataset used for the study was a book recommendation network dataset. The network had approximately 900 nodes and 2500 edges and had been drawn using the *neato* algorithm [EGK*02].

Tasks: To test H1, our participants solved two types of network tasks. In *edge* tasks, users determined the existence of direct connections between pairs of highlighted nodes. Tasks were limited to 3 seconds after which the screen faded out. Users answered the questions by pressing 'Y' or 'N' within the 3 seconds or anytime after the screen faded out. An answer would also prompt the study to advance to the next question. For each task, the view was centered at the midpoint between the two nodes and users were not allowed to use the mouse. We showed 175 instances of *edge*, the same ones in both conditions.

In *path* tasks, users looked for shortest paths between pairs of highlighted nodes. Tasks were limited to 35 seconds after which the study would advance automatically. To provide answers, users had to click on the path nodes. Within the last five seconds of each task, the screen faded slightly indicating to users that time draws to an end and they should provide an answer before the next question would replace the current one. Tasks involved paths of length three or four. Views were again centered between the two nodes and users were allowed to pan but not zoom. We showed 20 such questions, the same ones in both conditions.

A third task, *correction*, was used to test H2. With eye-tracking support enabled, users were shown a view of the graph in which a randomly selected quarter of nodes (225) were gray while all others were red. They were then asked to turn as many nodes as possible from gray to red by looking at them. They were allowed two minutes with gaze correction active and two minutes with gaze correction inactive. In this task users could pan but not zoom.

4.2. Results

Figure 8 summarizes the quantitative results of our user study. First, a Shapiro-Wilk test removed the possibility that the data was not normally distributed (all $p > 0.1$). We then analyzed the data using a paired t-test and found statistically significant accuracy improvements in the *edge* task of approximately 30% and in the *correction* task of approximately 25%. No difference was found for the *path* task. The full results are listed below:
***edge*:** $t(12) = 2.67$, $p = 0.021$, mean-difference= 12.5, effect size (Cohen's d) = 0.77;
***path*:** $t(12) = 0.74$, $p = 0.47$, mean-difference= 0.58, effect size (Cohen's d) = 0.21;
***correction*:** $t(12) = 3.10$, $p = 0.01$, mean-difference= 14.17, effect size (Cohen's d) = 0.89.

Qualitatively, all users preferred the eye-tracking enabled system, rated it as helpful or very helpful, and most of them rated it as appealing or very appealing (Figure 8). While not listed, none of the users found eye-tracking to be obtrusive.

Several reasons may have contributed to the lack of meaningful results for the *path* task. First, the high error rate indicates that the task was difficult. This is likely to have introduced a significant variability which our small user sample size could not capture. Second, while striving to minimize obtrusiveness we may have reduced the visual effect to the point that it was no longer helpful. Third, the higher cost of interaction over visual search may have washed out any performance difference introduced by the eye-tracker. Finally, the technique itself works better in some case than others. For example, in highly connected networks, entire regions "light up" around viewed nodes, thereby rendering any highlighting advantage void.

While gaze correction was shown to be working, the magnitude of the improvement is dependent on factors such as eye tracking calibration, lighting, or user particularities. In ideal settings, when eye-tracking works well, the correction effect would be negligible, such as in the case of two of our users. In poor conditions however, the effect can be significant. For example, just two users performed better on the *edge* task in the control condition. Those users also happen to have two of the most significant improvements in the *correction* task, suggesting that the eye-tracking was not functioning properly. Conversely, users that show only mild improvements in the *correction* task generally showed significant improvements in the the *edge* task.

5. Discussion

Data visualization is a suited application area for gaze interactions because of the inherent interplay between eyes and displays and between people's gazes and tasks they perform [YR67]. By interpreting users' intentions from gaze data, we can reduce perceptual overload and fit visualizations to users' tasks and intentions, we can reduce the overhead of manual interaction, and ultimately create visualizations that participate proactively in the analytic process.

While our results demonstrated the effectiveness of eye-

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

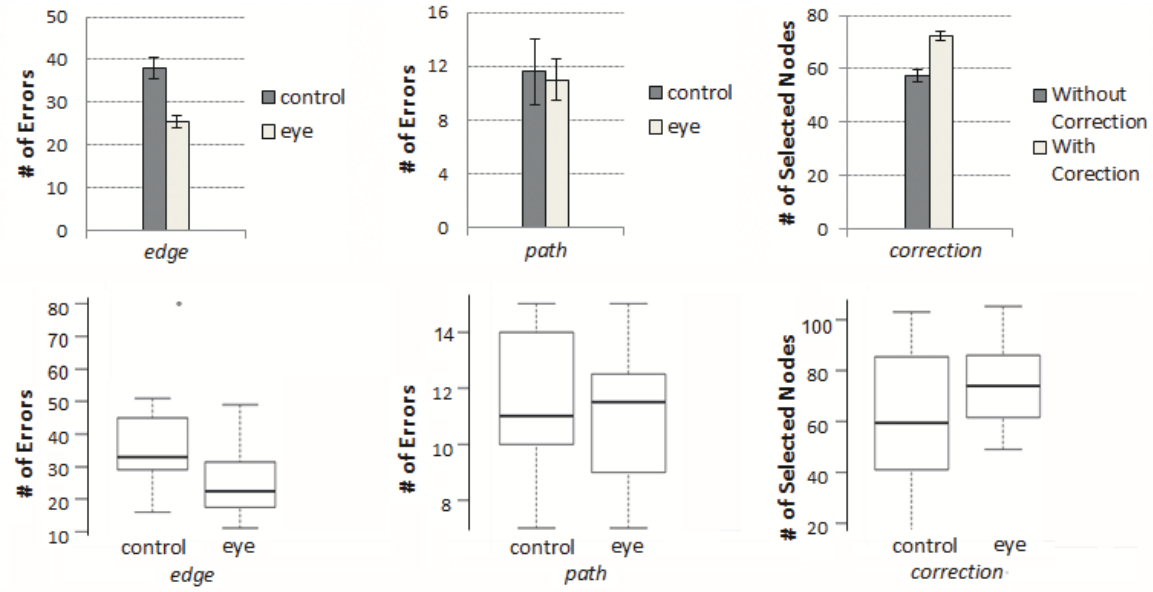

Figure 7: *Our quantitative results show that subjects using eye-tracking more accurately determined if two nodes are connected (left, $p = 0.02$) but showed no improvement finding shortest edges between nodes (middle). Our gaze-correction algorithm (section 3.1) increases the ability of eye-tracking users to fixate nodes (right, $p = 0.01$).*

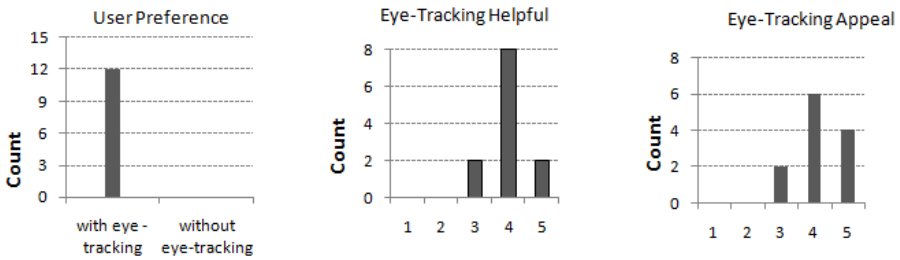

Figure 8: *Our evaluation questionnaire showed that subjects preferred eye-tracking interaction to the control condition, thought eye-tracking was helpful, and liked it.*

tracking in supporting low level tasks, they failed to reveal benefits in the more high-level task of path detection. We believe this was in part due to the design of our methods but also due to the difficulty of evaluating complex tasks quantitatively. However, given the reliability in detecting viewed nodes and edges, the strong effects in the short perceptual task, and the effectiveness of our gaze correction, we hypothesize that our methods can be used as a foundation for further exploration of high level interactive and analytic metaphors.

While we did not formally investigate the interplay between gaze interactions and network clutter, we noticed the following. When network density is very low, gaze interaction does not improve perception since users have little trouble finishing tasks before the system can produce a response. Moreover, with low densities users rely more on their peripheral view and do not always fixate objects even though they register their presence. In cluttered displays the performance of our system depends on the visual similarity of the traced object to other objects around them. For example, the system cannot differentiate between edges as long as they run alongside each other. However, it excels at highlighting long edges (i.e., 300+ pixels in screen space) running through regions dense with edges dissimilar in size or orientation. If we were to apply this technique to parallel coordinate views our expectation is that we couldn't visually select individual poly-lines, but that we could highlight clusters of poly-lines that run along our visual trajectory.

We also hypothesize that we can further improve on our detection of visual targets and user tasks. In sections 3.1 and 3.2.2 we mentioned several assumptions about people's gaze

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

patterns: long fixations for nodes, shorter fixations for edge (section 3.1); longer fixations in regions of high density, shorter fixations in regions of low density (section 3.2.2). These assumptions were made from informal observations during design and do not rely on data or models of visual perception. While such gaze models exist for specific areas such as reading, they have not been generalized for visual objects and layouts specific to our field. A better understanding of how visual parameters correlate to gaze measures such as fixation duration, saccade distance, or revisitation would enable us to detect viewed object in a more principled way.

In a complete analysis system, additional information could be leveraged. For instance, since eye-movement and fixation precedes motor movement and action [Duc07], mouse patterns can be used to confirm and adjust data about fixations. Using our gaze-correction in an environment with multiple views, one visualization could correct offsets for another visualization that may occupy the same screen space.

Modeling higher level analytic tasks specific to data visualization would allow us to detect and support tasks that go beyond perception and data reading. We hypothesize that gaze-support can be provided at multiple temporal scales, across multiple visualizations and datasets, and across multiple users. For instance, current interest would help with immediate perceptual tasks, recent interest could be used to achieve a relevance filtering effect to unclutter visualizations, and historical interest could be used to restore analysts memory, perhaps in conjunction with visualization history methods. Interest in visualization elements immediately translates to interest in data objects and can be transferred across multiple visualizations that share overlapping subsets of data. Moreover, interest harvested from groups of users could be used to direct or inform analysis of other groups of users with similar research interests. Such analysis could prove useful in domains such as genomics or proteomics were many users analyze overlapping subsets of the common space of genes, proteins, and interactions between them.

Finally, our work doesn't investigate the interplay between mouse interaction and gaze-interaction. Users may outperform the eye-tracking enabled visualization if allowed to select nodes and highlight their outgoing edges. One of the reasons for limiting *edge* tasks to three seconds was our desire to capture the effect of eye-tracking on short, perceptual tasks in which interaction is a significant overhead. We hypothesize that there is a class of interactive queries that would be cumbersome to specify deliberately but would be easy to compute based on users' gaze patterns. *Path* tasks tried to capture such a query. Asking the user to provide and continuously update information about nodes of interest would be unfeasible. Using eye-tracking, the same process could be done automatically.

6. Conclusion

We introduced techniques for using eye-tracking as interactive input in network visualizations and demonstrated their effectiveness in a controlled user study. Specifically, we dimmed out edges with endpoints outside users' view focus, we highlighted edges that were visually traced, and increased the saliency of sub-networks around nodes viewed often. We also described an algorithm that improved eye-tracking accuracy by leveraging the known layout of the network. In a user study with twelve participants, we showed that these techniques allow users to more accurately determine whether two nodes are connected. We also demonstrated the effectiveness of the gaze correction technique. Given the reliability in detecting viewed nodes and edges, the strong effects in the connectivity task, the success of the gaze correction technique, and the privileged role of gaze in data visualization, we hypothesize that further exploration of gaze-enabled interactions for visualization will be valuable.

Acknowledgments

This work was supported by Florida International University seed award AWD000000003432.

References

[AABW12] ANDRIENKO G., ANDRIENKO N., BURCH M., WEISKOPF D.: Visual analytics methodology for eye movement studies. *Visualization and Computer Graphics, IEEE Transactions on 18*, 12 (2012), 2889–2898. 2

[ADS05] ASHMORE M., DUCHOWSKI A. T., SHOEMAKER G.: Efficient eye pointing with a fisheye lens. In *Proceedings of Graphics interface 2005* (2005), Canadian Human-Computer Communications Society, pp. 203–210. 2

[BAA*13] BURCH M., ANDRIENKO G., ANDRIENKO N., HOFERLIN M., RASCHKE M., WEISKOPF D.: Visual task solution strategies in tree diagrams. In *Visualization Symposium (PacificVis), 2013 IEEE Pacific* (2013), IEEE, pp. 169–176. 2

[BKH*11] BURCH M., KONEVTSOVA N., HEINRICH J., HOEFERLIN M., WEISKOPF D.: Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study. *Visualization and Computer Graphics, IEEE Transactions on 17*, 12 (2011), 2440–2448. 2

[BMBL09] BORGATTI S. P., MEHRA A., BRASS D. J., LABIANCA G.: Network analysis in the social sciences. *science 323*, 5916 (2009), 892–895. 1

[BS09] BULLMORE E., SPORNS O.: Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience 10*, 3 (2009), 186–198. 1

[CCNS08] CHIN G., CHAVARRIA D., NAKAMURA G., SOFIA H.: Biographe: high-performance bionetwork analysis using the biological graph environment. *BMC bioinformatics 9*, Suppl 6 (2008), S6. 1

[DÇ07] DUCHOWSKI A. T., ÇÖLTEKIN A.: Foveated gaze-contingent displays for peripheral lod management, 3d visualization, and stereo imaging. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP) 3*, 4 (2007), 6. 2

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

[DDLS07] DREWES H., DE LUCA A., SCHMIDT A.: Eye-gaze interaction for mobile phones. In *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology* (2007), ACM, pp. 364–371. 3

[DS07] DREWES H., SCHMIDT A.: Interacting with the computer using gaze gestures. In *Human-Computer Interaction–INTERACT 2007*. Springer, 2007, pp. 475–488. 3

[Duc02] DUCHOWSKI A. T.: A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers 34*, 4 (2002), 455–470. 1

[Duc07] DUCHOWSKI A. T.: *Eye tracking methodology: Theory and practice*, vol. 373. Springer, 2007. 2, 9

[EGK*02] ELLSON J., GANSNER E., KOUTSOFIOS L., NORTH S. C., WOODHULL G.: Graphviz–open source graph drawing tools. In *Graph Drawing* (2002), Springer, pp. 483–484. 7

[GN00] GANSNER E. R., NORTH S. C.: An open graph visualization system and its applications to software engineering. *Software Practice and Experience 30*, 11 (2000), 1203–1233. 1

[HEH08] HUANG W., EADES P., HONG S.-H.: Beyond time and error: a cognitive approach to the evaluation of graph drawings. In *Proceedings of the 2008 Workshop on BEyond time and errors: novel evaLuation methods for Information Visualization* (2008), ACM, p. 3. 2

[HMR05] HYRSKYKARI A., MAJARANTA P., RÄIHÄ K.-J.: From gaze control to attentive interfaces. In *Proceedings of HCII* (2005). 2

[Jac90] JACOB R. J.: What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1990), ACM, pp. 11–18. 1

[Jac91] JACOB R. J.: The use of eye movements in human-computer interaction techniques: what you look at is what you get. *ACM Transactions on Information Systems (TOIS) 9*, 2 (1991), 152–169. 2

[JK03] JACOB R. J., KARN K. S.: Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. *Mind 2*, 3 (2003), 4. 2

[KPW07] KUMAR M., PAEPCKE A., WINOGRAD T.: Eyepoint: practical pointing and selection using gaze and keyboard. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2007), ACM, pp. 421–430. 2

[MZ08] MACKENZIE I. S., ZHANG X.: Eye typing using word and letter prediction and a fixation algorithm. In *Proceedings of the 2008 symposium on Eye tracking research & applications* (2008), ACM, pp. 55–58. 3

[OAJ13] OKOE M., ALAM S. S., JIANU R.: Using eye-tracking as interactive input enhances graph visualization. *IEEE Visualization 2013 Poster Compendium* (2013). 7

[ODH02] O'SULLIVAN C., DINGLIANA J., HOWLETT S.: Eye movements and interactive graphics. *Hyona, J. Radach, R. and Deubel, H. (eds.) The Mind's Eyes: Cognitive and Applied Aspects of Eye Movement Research* (2002), 555–571. 2

[OHM*04] O'SULLIVAN C., HOWLETT S., MORVAN Y., MCDONNELL R., O'CONOR K.: Perceptually adaptive graphics. *Eurographics state of the art reports 4* (2004). 2

[Ovi03] OVIATT S.: Multimodal interfaces. *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications* (2003), 286–304. 1

[PSD09] POHL M., SCHMITT M., DIEHL S.: Comparing the readability of graph layouts using eyetracking and task-oriented analysis. In *Proceedings of the Fifth Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging* (2009), Eurographics Association, pp. 49–56. 2

[RHN*03] RUDDARRAJU R., HARO A., NAGEL K., TRAN Q. T., ESSA I. A., ABOWD G., MYNATT E. D.: Perceptual user interfaces using vision-based eye tracking. In *Proceedings of the 5th international conference on Multimodal interfaces* (2003), ACM, pp. 227–233. 2

[SA00] SALVUCCI D. D., ANDERSON J. R.: Intelligent gaze-added interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2000), ACM, pp. 273–280. 3

[SCC13] STEICHEN B., CARENINI G., CONATI C.: User-adaptive information visualization: using eye gaze data to infer visualization tasks and user cognitive abilities. In *Proceedings of the 2013 international conference on Intelligent user interfaces* (2013), ACM, pp. 317–328. 2

[Sel04] SELKER T.: Visual attentive interfaces. *BT Technology Journal 22*, 4 (2004), 146–150. 1

[SLMS09] STREIT M., LEX A., MÜLLER H., SCHMALSTIEG D.: Gaze-based focus adaption in an information visualization system. In *IADIS International Conference Computer Graphics, Visualization, Computer Vision and Image Processing* (2009), pp. 303–307. 2

[TAK*05] TORY M., ATKINS M. S., KIRKPATRICK A. E., NICOLAOU M., YANG G.-Z.: Eyegaze analysis of displays with combined 2d and 3d views. In *Visualization, 2005. VIS 05. IEEE* (2005), IEEE, pp. 519–526. 2

[Ver02] VERTEGAAL R.: Designing attentive interfaces. In *Proceedings of the 2002 symposium on Eye tracking research & applications* (2002), ACM, pp. 23–30. 2

[VS08] VERTEGAAL R., SHELL J. S.: Attentive user interfaces: the surveillance and sousveillance of gaze-aware objects. *Social Science Information 47*, 3 (2008), 275–298. 2

[VSCM06] VERTEGAAL R., SHELL J. S., CHEN D., MAMUJI A.: Designing for augmented attention: Towards a framework for attentive user interfaces. *Computers in Human Behavior 22*, 4 (2006), 771–789. 2

[WM87] WARE C., MIKAELIAN H. H.: An evaluation of an eye tracker as a device for computer input2. In *ACM SIGCHI Bulletin* (1987), vol. 17, ACM, pp. 183–188. 2

[YR67] YARBUS A. L., RIGGS L. A.: *Eye movements and vision*, vol. 2. Plenum press New York, 1967. 2, 7

[Zha03] ZHAI S.: What's in the eyes for attentive input. *Communications of the ACM 46*, 3 (2003), 34–39. 2

[ZMI99] ZHAI S., MORIMOTO C., IHDE S.: Manual and gaze input cascaded (magic) pointing. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1999), ACM, pp. 246–253. 1, 2

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.