Analyzing Eye-Tracking Information in Visualization and Data Space: from Where on the Screen to What on the Screen

Sayeed Safayet Alam*, *Member, IEEE,* and Radu Jianu†, *Member, IEEE*

Abstract—Eye-tracking data is currently analyzed in the image space that gaze-coordinates were recorded in, generally with the help of overlays such as heatmaps or scanpaths, or with the help of manually defined areas of interest (AOI). Such analyses, which focus predominantly on where on the screen users are looking, require significant manual input and are not feasible for studies involving many subjects, long sessions, and heavily interactive visual stimuli. Alternatively, we show that it is feasible to collect and analyze eye-tracking information in data space. Specifically, the visual layout of visualizations with open source code that can be instrumented is known at rendering time, and thus can be used to relate gaze-coordinates to visualization and data objects that users view, in real time. We demonstrate the effectiveness of this approach by showing that data collected using this methodology from nine users working with an interactive visualization, was well aligned with the tasks that those users were asked to solve, and similar to annotation data produced by five human coders. Moreover, we introduce an algorithm that, given our instrumented visualization, could translate gaze-coordinates into viewed objects with greater accuracy than simply binning gazes into dynamically defined AOIs. Finally, we discuss the challenges, opportunities, and benefits of analyzing eye-tracking in visualization and data space.

Index Terms—Eye-tracking, Area of Interest Analysis, Usability Analyis, Evaluation

1 Introduction

Eye-tracking allows us to locate where users are looking on a computer screen [1], [2], and is used widely in psychology and cognitive science to help researchers understand thought and affect mechanisms [3], and in data visualization and human computer interaction (HCI) to explain how people use visual interfaces [4]. To date, eye-tracking data is collected and interpreted as gaze-coordinates in the space of rendered visual stimuli that gazes were recorded for. Relating this data to the semantic content of the stimuli is generally done offline by analysts who inspect gaze heatmaps visually, or define area of interest (AOIs) manually. This process is inefficient for studies involving many subjects, long sessions, and interactive content.

This paper explores a different approach: for visual content that a computer generates (e.g., data visualizations), the layout of the content is known at rendering time. Thus, gaze-coordinates provided by an eye-tracker can be related to rendered visual objects in real-time, yielding an account of which objects a user views at any given time. Specifically, our approach relies on a visualization to communicate to an object-detection module what content it draws on the screen (e.g., shapes and positions of glyphs), and for the module to match incoming gazes to that content. For interactive visualizations, this communication needs to persist throughout their run-time, as any visual changes (e.g., moving a glyph) need to be reported to the object detection module. Programmatically, we implement this communication by inserting calls to the object detection-module directly in the visualization's source code. For example, a rendering instruction used to add or reposition an object is mirrored by a call that informs the object-detection module of this change.

The approach is limited to visualizations for which the source code is accessible, and involves the overhead of instrumenting visualizations at a low level. These limitations are balanced by several advantages. First, once instrumented, a visualization can be reused to collect eye-tracking data from different datasets and multiple usage scenarios, unlike traditional analyses, which would require new AOIs to be defined for new data. Second, data collection is automatic and independent of a user's interactions, thus enabling experimenters to collect data from interactive visualizations in long studies. Third, the collected data is highly granular (e.g., a small glyph or label, rather than an entire screen region), and is derived directly from the visualization's underlying data (e.g., protein GRB2 in a protein interaction network), thus having semantic meaning without the need for additional coding. We posit that such data can be particularly well suited to explain how people forage for, analyze, and integrate information in complex visual analytics systems and workflows, as opposed to the traditional role of eye-tracking in studying visual perception.

Our paper contributes to establishing this method of instrumentation in data visualization. First, we introduce and evaluate an algorithm for viewed-object detection that is tailored to visualization content and is more accurate than just binning gazes into AOIs. We also describe a recipe and guidelines for instrumenting visualizations in practice. We build on results by Salvucci, who showed that viewed object detection is more accurate if gazes are "interpreted intelligently", by leveraging that users don't view content randomly, but in patterns influenced by their tasks [5],

* *S. S. Alam is with the School of Computing and Information Sciences, Florida International University, Miami, FL, 33199. E-mail: salam011@cis.fiu.edu*

† *R. Jianu is with the School of Computing and Information Sciences, Florida International University, Miami, FL, 33199. E-mail: rdjianu@cis.fiu.edu*

Manuscript received April 19, 2005; revised September 17, 2014.

[6]. For example, as shown in Section 4, we found that users look predominantly at highlighted items that are connected to what they already viewed previously.

Second, we show that our instrumentation approach is feasible, can be used to collect data over long sessions involving open-ended tasks and interactive content, and produces meaningful results. We instrumented an interactive visualization of IMDB data and collected viewed-object data from nine subjects. We show that viewed objects we detected automatically were tightly correlated with the tasks we asked users to do. We also show that differences between manual annotations of five coders who were asked to analyze the raw gaze data, and our automatically collected data, are not greater than differences between the coders' annotations. Moreover, we show that our novel algorithm for detecting viewed objects outperforms two naïve implementations.

Third, we demonstrate novel analyses that this instrumentation facilitates. Specifically, we show how the collected data allows us to demonstrate the existence of viewing patterns in visualization, and we discuss the range of applications that the method enables.

2 Related Work

Eye-tracking can locate users' gazes on a computer screen [1], [2] and is a technology that is becoming increasingly accurate, fast, and affordable [4], [7]. Eye-tracking is often used to record peoples' gazes while performing tasks that involve visual stimuli, and to analyze the data off-line to gain insight into how people interpreted the stimuli and solved the tasks [4]. For example, eye-tracking was used to understand how people perceive faces [8], [9], to study how attention changes with emotion [10], to understand changes in perception that are caused by disease [11], and to gain insight into how students use visual content to learn [12], [13], [14], [15]. Within the field of data visualization, examples of eye-tracking studies include but are not limited to work by Pohl et al. and Huang et al. on graph readability [16], [17], [18], Burch et al.'s work on tree-drawing perception [19], [20], and work by Kim et al. on evaluating an interactive decision making visualization [21].

Eye-tracking data is traditionally interpreted in the space of rendered visual stimuli that gazes were recorded for, using one of two analysis paradigms: point based or area of interest (AOI) based [22]. Point based analyses treat gaze samples or fixations as independent points while AOI analyses aggregate gazes into areas of interest and operate at this higher level of abstraction. Most often, experimenters define AOIs manually, but gaze clustering algorithms are also available to automatically define AOIs based on eye-tracking data [23], [24], [25].

Our work is close to a few methods that seek to automatically relate gazes to the semantics of computer generated content. Several papers allude to the fact that AOIs could be defined dynamically for such cases [26], [27], but none formalize an approach or quantify feasibility. For dynamic stimuli with known 3D structure, researchers have explored the concept of objects of interest (OOI), in which gazes are automatically assigned to 3D objects in the scene [28]. More recently, Bernhard et al. looked at similar gaze-to-object mapping in the context of understanding what people were looking at in virtual 3D environments [29]. Our work presents a more detailed account of how gazes can be automatically assigned to content typical of 2D information visualizations and evaluate how effective this can be.

Moreover, our method of detecting viewed objects improves over naïve methods by leveraging Salvucci's "intelligent gaze interpretation" paradigm [5], [6]. Specifically, Salvucci found that simply assigning gazes to an object if the gazes' coordinates are within the bounds of the object is insufficient, and that leveraging the semantics of visual content can significantly improve our ability to predict which objects are viewed. More recently, Okoe et al. found similar results, albeit using a different method [30], [31]. We extend on such work by presenting an intelligent gaze interpretation' algorithm that is tailored to content typical of data visualization and by evaluating it.

Finally, visualization researchers proposed a plethora of visual analytics tools for both point-based and AOI based eye-tracking analysis. Blascheck et al. provides a comprehensive review of such methods [22]. Most relevant to our work are methods for AOI visualization, since our data is in essence a highly granular and annotated AOI data. Popular examples of such techniques include scarf plots [32], AOI rivers [33], and AOI transition matrices [34]. Also relevant are visual analytics principles and systems for analyzing AOI data, such as work by Andrienko et al. [35], Weibel et al. [36], and Kurzhals et al. [27]. However, the data our instrumentation allows us to collect differs from regular AOI data through its high granularity, connection with the underlying data of visualizations, and uncertainty about whether an object AOI was truly viewed. The focus of the work we present here is not in proposing novel techniques of analyzing data collected in visualization and data space, but on whether and how this can be done accurately.

3 Approach

3.1 Overview

Our general approach is illustrated in Figure 1. For visualizations with accessible code, gaze coordinates from an eye-tracker can be mapped to visual objects displayed on the screen automatically, since the layout of the visual content is known during rendering. A visualization instrumented with our approach will not only draw visual objects on the screen (e.g., nodes in a graph), but will also inform a viewed-object detection algorithm about the position and shape of such objects. The detection algorithm uses this information to map 2D gaze coordinates to visualization objects rendered on the screen. Since interactive visualizations change dynamically at run-time, should the visualization be transformed (e.g., zoomed, panned), its content altered (e.g., visual objects added or removed), or individual visualization components moved (e.g., dragging a node), the detection algorithm is informed about these changes as soon as the visualization is redrawn, so it can map subsequent gaze samples to the new visual layout.

Next, we describe an algorithm for mapping gaze samples to objects that users are viewing. As shown in Figure 1, we will assume that gaze samples have already been transformed from screen space into model space. As such, the basic input of our algorithm is a stream of gaze samples in model space, and a list of visual primitives drawn on the screen, together with their shape and position. The algorithm outputs a stream of viewed visualization and data primitives (e.g., nodes, labels) in real-time, as users are viewing them in the instrumented visualization. Our approach is limited in that it cannot be applied to already rendered images or videos and requires that the visualization's source code be altered. We discuss these limitations in Section 6.2.

We describe our algorithm incrementally in the following three sections, starting from a naïve approach that simply draws AOIs dynamically around visualization objects, to a predictive one that

detects objects more accurately by using knowledge about how specific visualizations are typically used. A comparative evaluation of these detection algorithms is presented in Section 4.4.1.

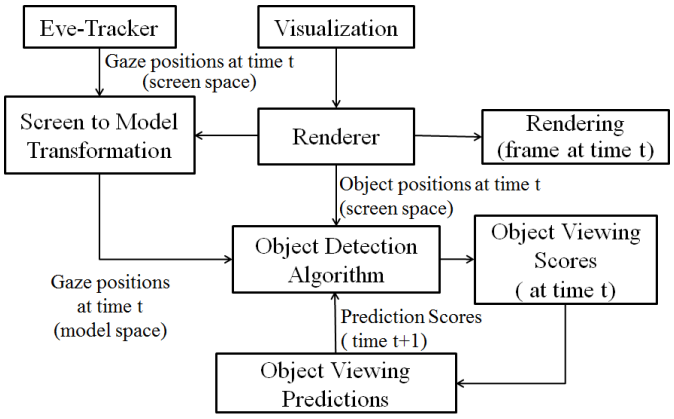


Fig. 1: Real-time detection of viewed objects in generative visualizations.

3.2 AOI-based viewed object detection

A naïve approach is to treat object shapes as dynamic AOIs and determine that a viewed object is that with the most recent fixation landing in its AOI. Manually drawn AOIs are typically used in the same manner in offline eye-tracking data analysis, and the similar concept of objects of interest (OOIs) has been proposed already by Stellmach et al. [28] for generative 3D content.

The problem with this approach is that for highly granular visual content, such as individual nodes or labels, users often fixate in the vicinity of the object rather than on the object itself. A potential solution is to pad object AOIs to be slightly larger than the objects. However, larger AOIs may lead to overlap in cluttered visualizations. We demonstrate and quantify these observations in Section 4. Ultimately, the problem lies with an inability to determine with absolute certainty what a user is looking at, and is described in more detail in the next section.

3.3 A probabilistic approach to viewed object detection

Unlike mouse input, eye-tracking can only indicate a small screen region that a user is fixating, rather than a particular pixel. Typically, such regions are about one inch in diameter, though specific values depend on viewing conditions. As such, it is generally impossible to tell with certainty which object a user is viewing, if the user is fixating in the vicinity of multiple close objects (Figure 2(a)). This is not a significant problem for traditional AOI analyses, which generally use large AOIs. Conversely, we aim to detect the viewing of granular visual content, such as network nodes or glyphs, in cluttered visualizations.

We advocate for a fuzzy interpretation of gaze data and detect likelihoods that objects are viewed rather than certainties. To this end, we can compute object gaze scores gs (for all objects i in a visualization, and at all times t) that range between zero- the object is not viewed, and one- the object is certainly viewed, as shown in Figure 3 and Formula 1.

$$gs_{i,t} = 1 - \min(1, (\frac{d}{R})) \quad (1)$$

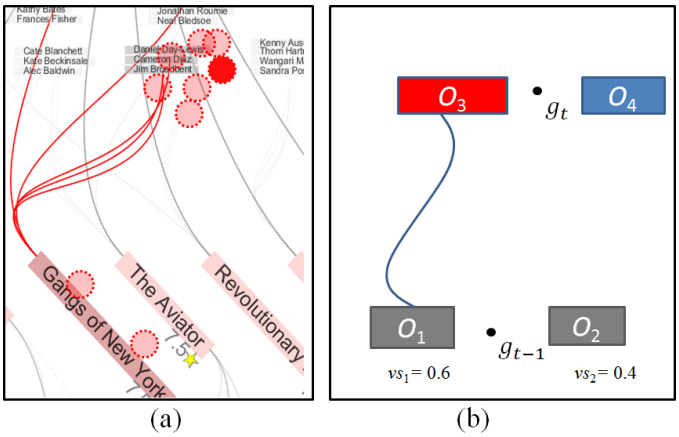


Fig. 2: (a) A real visualization example in which a user fixates in the vicinity of multiple close object groups (red dot). (b) predictive method: even though the latest gaze sample falls equidistantly between visual objects O_3 and O_4, we suspect that O_3 is the more likely viewing target given that it is highlighted and connected to O_1, which is likely to have been the object that the user viewed previously ($vs_1 = 0.6 > vs_2 = 0.4$).

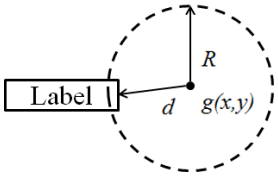


Fig. 3: Calculating gaze score gs for a gaze sample landing near an object. d is the distance from the object to the gaze sample, and R approximates the size of the user's foveated region.

The region of radius R used in the formula is analogue to the user's foveated region, and as such needs to be constant in screen space. Thus, if the view is zoomed in or out, R needs to be scaled accordingly in model space to remain constant in screen space. A more detailed discussion about choosing an appropriate R is given in Section 6. Similar approaches were used by Salvucci et al. [6] and Okoe et al. [30].

Finally, we note that the object scores (gs) do not directly equate to probabilities. The distinction is important because our implementation can detect two objects as being viewed simultaneously ($gs_1 = 1$ and $gs_2 = 1$). We think this is appropriate since a person can in fact visually parse multiple objects at the same time if they fall within the user's foveated region, and even think of multiple objects as a unit for specific task purposes. We discuss this in more detail in Section 6.4.

3.4 A predictive algorithm for viewed object detection

Salvucci and Anderson described the concept of "intelligent gaze interpretation" in the context of a gaze-activated interface [6]. They more accurately detected which interface control a user was gazing at, by integrating both the proximity of the gaze to the control, and the likelihood that the control was the target of a gaze-interaction, based on the current state and context of the interface. Formally, their algorithm identified the most likely currently viewed item i_{viewed} by solving

$$i_{viewed} = \underset{i \in I}{\operatorname{argmax}}[Pr(g|i) \cdot Pr(i)]$$

where $Pr(g|i)$ is the probability of producing a gaze at location g given the intention of viewing item i, and $Pr(i)$ is the prior probability of an item i being the target of a gaze-interaction. Salvucci and Anderson based these prior probabilities on assumptions about how an interface might be used, and hardcoded them into their system.

We adapt Salvucci and Anderson's paradigm to solve the ambiguous case when a gaze-sample lands close to multiple objects (e.g., Figure 2(a)). For example, in a network visualization we may assume that a user who has just viewed a node n will more likely view one of n's neighbors than a random other node, perhaps especially if the user previously highlighted node n and its outgoing edges. In Section 4 we show quantitatively that this assumption holds for one tested visualization.

We consider the simplified scenario in Figure 2(b): four visual objects ($O_{1...4}$), two of which are connected (O_1 and O_3), and one of which is highlighted (O_3), are shown on the screen. A new gaze sample registers between O_3 and O_4 at time t. Intuitively, it is more likely that O_3 was viewed since it is highlighted. Moreover, if we knew that O_1 was viewed just before the current moment and assume that users generally view neighboring nodes together, then this likelihood becomes stronger.

Formally, we compute $vs_{i,t}$ (i.e., the viewing score vs of object i at time t) by weighing the gaze score $gs_{i,t}$ described in Section 3.3 by a prediction score $ps_{i,t}$ that object i is a viewing target at time t:

$$vs_{i,t} = gs_{i,t} \times ps_{i,t} \tag{2}$$

This prediction score is computed based on the likelihood that an object is viewed if another object (e.g., a node's neighbor) was viewed just before it. Specifically, ps is derived from a viewing transition function T between objects: $T(j,i)$ gives the likelihood that object i is viewed after object j is viewed. We will assume that $T(j,i)$ is given as input to our algorithm. Concrete examples of what $T(j,i)$ could be linked to are whether objects i and j are somehow connected or related, or whether they are part of a special group (e.g., highlighted elements). Moreover, connections could be either visual, such as an explicit edge or leader line or an implicit sharing of similar visual attributes (e.g., color, shape), or semantic (e.g., both nodes are actors). More examples of $T(j,i)$ functions, and means of defining them, are described throughout the paper.

To compute ps, we could consider $ps_{i,t} = T(j,i)$ but that would involve knowing j, the previously viewed object, with absolute certainty. As exemplified in Figure 2(b), we often cannot unequivocally determine which item was viewed at a given time: O_1's previous viewing score ($vs_{1,t-1} = 0.6$), is just slightly larger than O_2's viewing score ($vs_{2,t-1} = 0.4$), and thus an absolute choice of O_1 over O_2 as previously viewed element would be rather arbitrary. In other words, we cannot say with absolute certainty which of the two objects was viewed before because the user fixated between them.

In more general terms, our computation of $ps_{i,t}$ must account for multiple items j that may have been viewed before. These items j are those with a previous visual score $vs_{j,t-1}$ that is greater than 0. As such, we compute $ps_{i,t}$ as a weighted average of all transition probabilities from objects j with $vs_{j,t-1} > 0$, to our current item i. The weights are given by the likelihood that an object j was viewed before - in other words by its previous viewing score $vs_{j,t-1}$. This computation is captured by Formula 3.

$$ps_{i,t} = \frac{\sum_j vs_{j,t-1} \times T(j,i)}{\sum_j vs_{j,t-1}} \text{ , where } \begin{array}{l} 0 \le i \le n \text{ and } gs_{i,t} > 0 \\ 0 \le j \le n \text{ and } vs_{j,t-1} > 0 \\ 0 \le j \le n \text{ and } gs_{j,t} = 0 \end{array} \tag{3}$$

Finally, an important constraint needed to be added to Formula 3. Intuitively, our approach means that previously viewed objects j act as referees with varying degrees of influence (i.e., previous visual scores) in a competition between currently viewed items i. This analogy provides the intuition for the necessary constraint: an object should not referee a competition that it is part of. For example, in our simplified scenario, using O_3 as a previous element in a competition between itself and O_4 would result in an open feedback-loop and should be avoided. This restriction is reflected in Formula 3 by the 3rd inequality. The algorithm pseudocode is provided below.

Algorithm 1 Viewed Object Detection Algorithm

1: **Inputs:**
   $O_{i,\ldots,n}$= tracked visualization objects (shapes, positions)
   $g(x,y)$ = gaze sample in model space (time t)
   $T(i,j)$ = viewing transition function ($T(i,j) \in [0,1]$)
2: **Outputs:**
   $vs_{i,t}$ = momentary viewing scores of all objects ($i = 1,\ldots,n$).
3: **for** $i \leftarrow 1$ to n **do**
4: Compute $gs_{i,t}$ using Formula 1
5: $max \leftarrow 0$
6: **for** $i \leftarrow 1$ to n **do**
7: **if** $gs_{i,t} > 0$ **then**
8: Compute $ps'_{i,t}$ using Formula 3
9: **if** $ps'_{i,t} > max$ **then**
10: $max \leftarrow ps'_{i,t}$
11: **for** $i \leftarrow 1$ to n **do**
12: $vs_{i,t} \leftarrow gs_{i,t} \times \frac{ps'_{i,t}}{max}$

Last, we note that to optimize for speed, we only compute prediction scores for objects with non-zero gazes (Algorithm 1, line 7). Also, we compute viewing scores for every gaze sample, rather than every fixation. We believe that doing so leads to results that are less dependent on how fixations are computed and more robust. Since our eye-tracker's sampling rate is 120Hz, the scores $vs_{j,t-1}$ were computed just 8ms ago, an interval generally shorter than the time it takes for people to shift their attention to a new object. As such, instead of using the raw $vs_{j,t-1}$ score, we use an average of the last several viewing scores, and, for all practical purposes, the term $vs_{j,t-1}$ should be replaced in the previous formulas by $\sum_{k=1}^{k=15} vs_{j,t-1}$, which, given our eye-tracker's 120Hz temporal resolution, averages samples over approximately 125ms, a time window we observed to be close to an average fixation duration. However, we note that our algorithm can take as input fixations rather than individual gaze samples, in which case this step would not be necessary. Moreover, additional smoothing and filtering such as those summarized by Kumar et al. [37] could be used to clean gazes before feeding them into our algorithm. We tried removing gaze samples with high velocity as they are likely to be part of saccades, but observed no discernable improvement in our algorithm's output.

Performance analysis: The algorithm swifts through all objects (n) to find those in the proximity of a gaze sample or fixation (k_t). Then, to compute ps for each of the k_t potentially viewed elements, the algorithm iterates over k_{t-1} objects with non-zero viewing scores from the previous iteration. The algorithm is linear if we consider the number of objects that a user can view at any time to be a constant. This is not true for example if the visualization is zoomed out too much and falls entirely within the algorithm's R radius. However, in such cases the output of the algorithm would be meaningless anyway and the algorithm should be aborded.

4 EVALUATION

4.1 Overview

We instrumented Doerk's interactive PivotPaths visualization of multifaceted data [38]. The visualization was linked to the popular internet movie database (IMDB) and is shown in Figure 4. We collected data from 9 subjects, each using our instrumented visualization for 50 minutes on a series of structured and unstructured tasks. We used this data data to test the validity and effectiveness of our approach in two ways.

First, we compared the output of the predictive algorithm to human annotations. We found that data collected automatically was on average as similar to human annotations, as human annotations were similar to each other. We conducted this analysis for all three viewed detection algorithms described in Sections 3.2 to 3.4 and found that the AOI algorithm performs poorly compared to the other two, and that the predictive algorithm improves detection accuracy by about 5% (Figure 5).

Second, we showed that our instrumentation method provides relevant information that can be leveraged in novel ways. We showed both qualitatively and quantitatively that viewed objects detected automatically were closely correlated to tasks people were asked to do, and that data collected automatically from many users could answer novel questions about how people use visualizations (Figures 6 and 7). We also demonstrated quantitatively that the viewing-biases our predictive algorithm exploits exist and are significant: our users were much more likely to look at objects that were highlighted and connected to each other (Table 2).

4.2 Instrumenting a sample visualization

Our PivotPaths visualization of IMDB data renders movies in the center of the screen, actors on top, and genres and directors at the bottom (Figure 4). Actors, directors, and genres are connected by curves to the movies they associate with, and are larger, and their connections more salient, if they are associated with multiple movies. Actors, genres, and directors are colored distinctively, which is particular important for genres and directors since they occupy the same visual space. Such views are created in response to users' searches for specific movies, actors, and directors, and show only data that is most relevant to the search. As shown in Figure 4, users can hover over visual elements to highlight them and their connections. Users can also click on visual elements to transition the view to one centered on the select element. Finally, users can freely zoom and pan.

We opted to instrument this visualization for three reasons. First, it is highly interactive and would be significantly difficult to analyze using traditional analyses. Second, it contains visual metaphors, graphic primitives, and interactions typical of a wide range of visualizations. Third, movie data is familiar to a broad range of users.

To choose transitions functions T underlying our predictive algorithm, we made informal assumptions about how the visualization is used, an approach also employed by Salvucci [6]. We assumed that transitions between connected items would occur more often than between unconnected items. We also assumed that highlighted elements are more likely to be viewed than those that are not. We translated these assumptions into specific weights, as exemplified in Table 1. We show in Section 4.4.1 that these assumptions hold for the instrumented visualization and the subjects that used it in our study. We describe more principled ways of choosing T functions in section 5.

Assumed visual and transition weights	
Movie to unconnected actor	1
Movie to connected actor	3
Movie to unconnected genre	1
Movie to connected genre	3
Movie to unconnected director	1
Movie to connected director	3

TABLE 1: Example transition probabilities in our instrumented visualization (assumed).

Finally, as part of instrumentation, our system collected screen shots, interactive events (e.g., zooming, panning), raw gaze samples captured at a rate of 120Hz, and visual elements that users viewed. For each viewed element we recorded the type (i.e., movie, actor, director, genre), its label, its gaze score (gs), its prediction score (ps), and the aggregated viewing score (vs). All recorded data was time stamped.

4.3 Study Design

Setup: We used the IMDB visualization described above, and an SMI RED-120Hz connected to a 17" monitor. Subjects were seated approximately 30″ away from the display.

Subjects: We collected data from 9 graduate and undergraduate students aged between 20 and 30 years. Six subjects were male and three were female. All were paid $10 for their participation.

Protocol: Subjects were first given a description of the study's purpose and protocol. They were then introduced to the visualization and asked to perform a few training tasks. This introductory part lasted on average 10 minutes. The main section of the study followed, involved multiple instances of four types of tasks, and lasted approximately 50 minutes.

Tasks: Subjects completed four types of structured and unstructured tasks. To solve structured tasks, subjects had to consider data that was better defined and less variable than in unstructured tasks. This made it easier for us to test the degree to which object-detection was aligned with the task associated data. On the other hand, data collected in unstructured tasks may be more ecologically valid. We limited the time we allowed subjects to spend on each task for two reasons: to manage the total duration of the study, and to make results comparable across users.

- **Task1 (structured):** Finding four commonalities between pairs of movies. The tasks were limited at three minutes each, and subjects solved the following four instances of this task: (a) Goodfellas and Raging Bull; (b) Raiders of the Lost Ark and Indiana Jones and the Last Crusade; (c) Invictus and Million Dollar Baby; (d) Inception and The Dark Knight Rises.

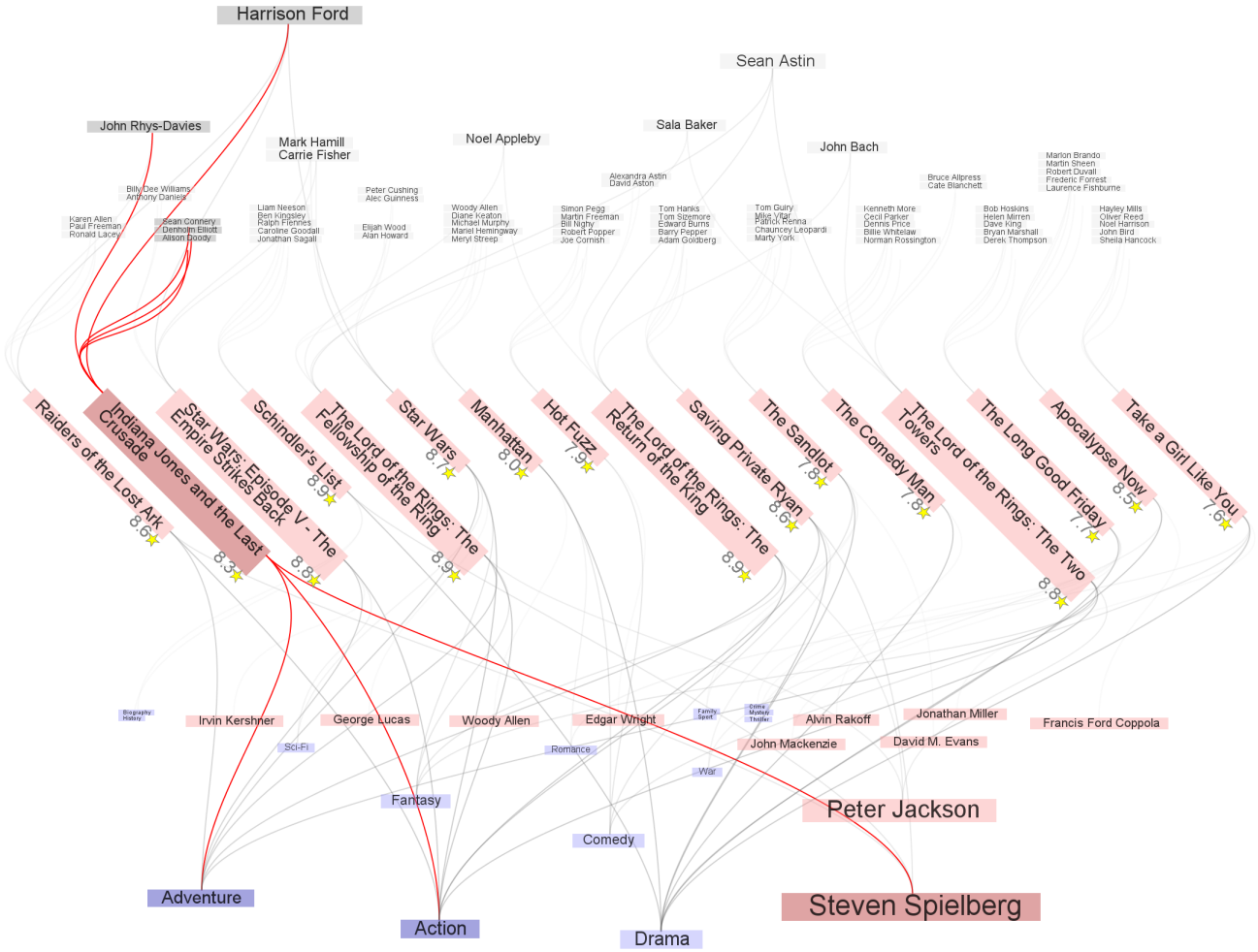


Fig. 4: PivotPaths visualization of IMDB data. Movies are displayed in the center of the screen, actors at the top, and directors and genres share the bottom space. Actors, directors, and genres associated to movies are connected through curves. Users can highlight objects and their connected neighbors by hovering over them.

- **Task2 (structured):** Ranking collaborations between a director and three actors (2 minutes, 4 instances): (a) Ang Lee; (b) Tim Burton; (c) James Cameron; (d) David Fincher.
- **Task3 (semi-structured):** Given three movies, subjects were asked to recommend a fourth (5 minutes, 3 instances): (a) Catch Me If You Can, E.T. the Extra-Terrestrial, and Captain Phillips; (b) To Kill a Mockingbird, The Big Country, and Ben-Hur; (c) Inglourious Basterds, The Avengers, and Django Unchained.
- **Task4 (unstructured):** Given a brief and incomplete description of the "Brat Pack", a group of young actors popular in the 80's, subjects were asked to find additional members and movies they acted in. Subjects solved one such task, in approximately 5 minutes.

4.4 Results

4.4.1 Data collected automatically is similar to that of human annotators

We tested whether the outputs of the three algorithms described in Sections 3.2 to 3.4 (AOI, probabilistic, and predictive) are comparable to annotation data obtained from human coders who inspected screen-captures with overlaid gaze samples and manually recorded what subjects looked at. We included in our analysis the AOI algorithm version which uses padded AOIs (Section 3.2). As shown in Figure 5, we found that the overlap between human annotations and the predictive algorithm's output is similar to the overlap within the set of human annotations, and that the predictive algorithm outperforms the others.

We enlisted the help of five coders and asked them to annotate eye-tracking data corresponding to one task of approximately three minutes, for each of six subjects. The task was the same for all coders - task 1b. The six subjects were selected randomly and were the same for all five coders. Coders spent approximately one hour per subject completing their annotation. This long duration meant it was unfeasible to code data from more users or more tasks. Four coders completed all six assigned annotation tasks, while one was able to annotate the data of only three subjects.

Coders used an application that allowed them to browse through screen captures of a users' activity with overlaid gaze coordinates. We asked coders to advance through the videos in 100ms time-steps, determine what visual objects their assigned subjects were viewing, and record those objects along with the

start and end time of their viewing. If unsure which of multiple objects was viewed, coders were allowed to record all of them.

We transformed each coder's annotation into temporal vectors with 100ms resolution. These vectors contained at each position one or several objects that were likely viewed by the subject during each 100ms time-step. We then created similar representations from our automatically collected data. Finally, we defined a similarity measure between two such vectors as the percentage of temporally aligned cells from each vector that were equal. Equality between vector cells was defined as a non-empty intersection between their contents.

For each algorithm, we computed the similarity of its output for each subject's data to all available human annotations of the same data. This yielded 4 coders $\times$ 6 subjects $+$ 1 coder $\times$ 3 subject $=$ 27 similarities per algorithm. We averaged these similarities and plotted them as the first four bars in Figure 5. Then, we compared each coder's annotation of a subject's data to all other available annotations of the same data. Since we had five annotations for three subjects, yielding 3 subjects $\times$ 10 annotation pairs $=$ 30 similarities, and four annotations for the remaining subjects, yielding 3 subjects $\times$ 6 annotation pairs $=$ 18 similarities, we obtained 48 similarities, which we averaged and plotted as the last bar of Figure 5.

The data we collected allowed us to perform this analysis for all three algorithms described in Section 3, as well as for the padded version of the AOI method. If we only consider gaze scores *gs* that are equal to one (Section 3.2) and no predictive component, we essentially have the output of the AOI algorithm. If we limit the analysis to *gs* scores alone, without the prediction component described in Section 3.4, we have the output of the probabilistic approach described in Section 3.3.

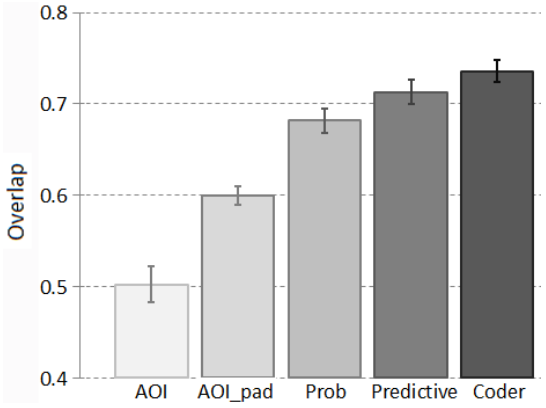


Fig. 5: Comparison between automated and manual viewed object detection. The first four bars show the overlap between the outputs of the three algorithms described in Section 3 (padded-AOI approach included), and annotation results of human coders. The last bar shows the overlap within the set of human annotations. Values correspond to averages over multiple tasks, multiple subject data sets, and multiple annotators, and are computed as described in Section 4.4.1. Error bars extend by one standard error.

4.4.2 *Data collected automatically is relevant and useful*

We used two analyses to show that data collected automatically is tightly correlated with the tasks that users had to do. We chose this evaluation for two reasons. First, it provides evidence that our instrumentation approach can be used to solve the inverse problem: an observer or analyst who is unfamiliar with a subject's intentions can determine what these are by looking at the subject's visual interest data.

Second, it demonstrates how the automated collection of eye-tracking data can facilitate novel insights into how visualizations are used. Our approach allowed us to quantify that a users' interest in a visual item present on the screen decays exponentially with a decrease in the items' relevance to a task. While it was generally known that users follow "information scent" when solving tasks visually [39], we were able to quantify this effect.

First , we created heatmap representations from our collected data (Figure 6) to illustrate qualitatively the strong connection between the tasks our subjects performed and the data we collected. We listed viewed objects vertically, discretized viewing scores by averaging them over 500*ms* intervals, and arranged them horizontally. Thus, time is shown horizontally, viewed objects vertically, and intensity of heatmap cells indicate the degree to which an object was viewed at a given time. The viewed objects listed vertically were colored based on their type (movie, actor, director, genre) and could be sorted by either first time they were viewed, amount of viewing activity, or type.

Figure 6 shows the data collected from a subject performing task 1b: finding commonalities between two Indiana Jones movies. We notice that elements viewed often are tightly connected to the subjects' task. Moreover, we can distinguish a temporal pattern: the movies featured in the task description were viewed throughout the analysis, actors were considered early on, followed by genres, then directors, and ultimately a quick scan of other movies. We observed this pattern for most subjects and think it was caused by the ordering used in the task's phrasing: we asked subjects to determine actors, genres, and directors that were common between the two movies.

Second, we formalized the relevance of each visual item to a particular task and plotted this relevance against the amount of interest that each item attracted (Figure 7). These plots quantify the degree to which tasks determine users' interest in visual items, and demonstrate that our instrumentation captures relevant data.

We formalized the relevance of a visual item to a task as Relevance $= 1/(1+d)$, where d is the shortest graph distance between that item and items mentioned directly in the task description. To exemplify, the relevance of Goodfellas and Ranging Bull to task 1a is 1 as they are the focus of the task, that of Martin Scorsese is $1/2$ because he directed both movies, while that of other movies directed by Scorsese is $1/3$. This definition is not fully accurate as items might be relevant to a task even though they are not directly mentioned in the description. For instance, items that eventually constitute a user's answer will elicit more attention.

Figure 7 facilitates several insights. First, even though many items were shown to subjects during their tasks, only very few were viewed for significant periods of time, and many were not viewed at all. Second, the types of data that users focus on, correlate with the particularities of each task. For example, task 3 involved movie recommendations and Figure 7 illustrates that genres and directors were viewed significantly more than in task 4, which involved determining the identity of a group of actors and seemed to drive users' attention towards actors.

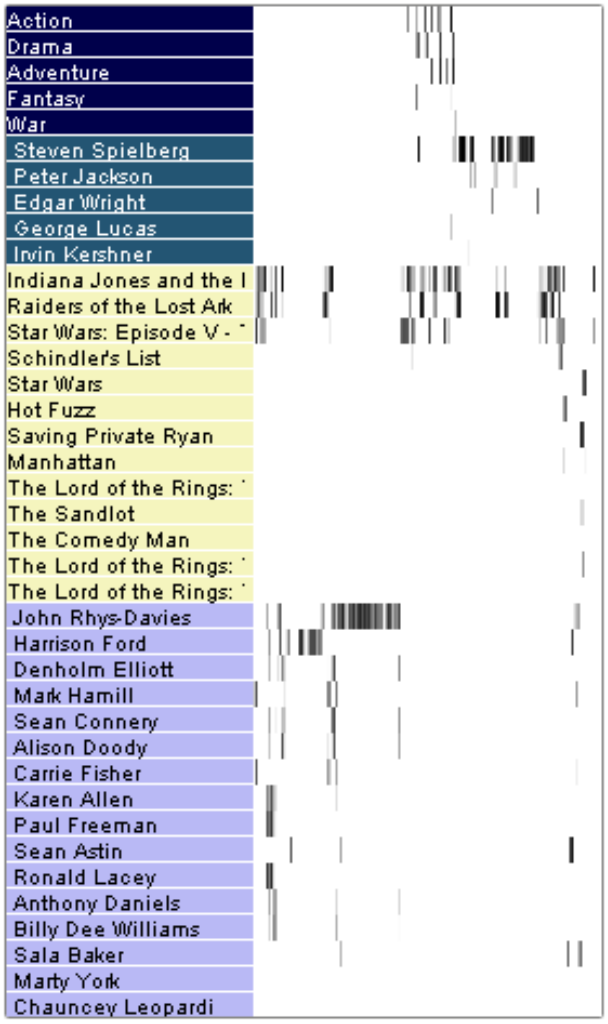


Fig. 6: Heatmap views of one subject's activity on task 1b; time, in 500ms increments, is shown horizontally; viewed objects are viewed vertically; cell darkness indicates viewing intensity (black: high; white: low); viewed items are ordered by category (genre, director, movie, actor).

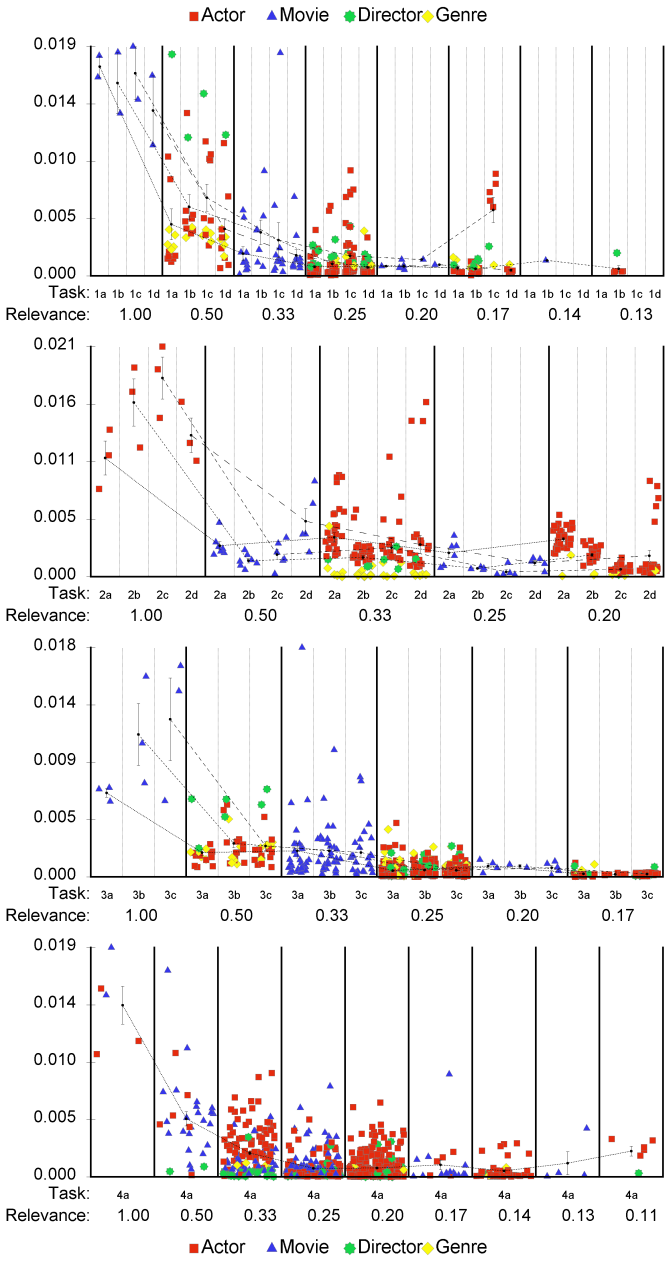


Fig. 7: Users' interest in data objects, in relation to each objects' relevance to a task, for twelve tasks of four types. Each individual task is plotted in its type's corresponding chart as a subdivision across multiple relevance categories. Relevance was computed as described in Section 4.4.2, and plotted for all objects that were visible to subjects during each task. The average interest in objects with the same task relevance are linked by separate polylines for each individual task; errors bars extend from the averages by one standard error.

4.4.3 *Viewing transition biases exist and are significant*

We performed a quantitative analysis of our subjects' viewing-transition patterns, using the data we collected during our study, and found that the informal assumptions we made in Section 3.4 were correct: our users showed strong preferences to view objects that were highlighted or connected to previously viewed objects. The last three columns in Table 2 compare the probability with which our users viewed one object category after another (e.g., viewed a highlighted actor after a movie) as computed from data we collected, to a null hypothesis in which users pick at random which items to view next. The quantitative results show for instance that after viewing a movie, our users were four times more likely to look at an actor that was highlighted ($Ratio = 4.081$), and eleven times more likely to look at an actor that was both highlighted and connected to the previously viewed movie ($Ratio = 11.484$), than if users were viewing items at random.

To reach these results, we first discarded the prediction component from our data, since it represents exactly the assumption we seek to evaluate. We then counted direct viewing transitions between all types of objects (sources) to all other types of objects (targets) and divided them into categories based on whether targets were highlighted, connected to the sources, or both (Table 2). For example, after looking at a movie, our users looked at an actor that was unconnected to that movie and unhighlighted 793 times, and at an actor that was connect to the movie and highlighted 616 times. Since in our visualization connections existed only

between movies and actors, genres, and directors, transitioning to connected targets was only possible to and from movies.

These counts were translated into observed transition probabilities by normalizing them by the total number of transitions from each type of source to each type of category. For example, our users transitioned in total 1784 times from a movie to an actor, of which 147 transitions were from a movie to a highlighted actor, yielding an observed transition probability of $147/1784 = 0.082$.

However, interpreting these observed probabilities by themselves can be misleading. For example, we observed 793 transitions from a movie to an unconnected actor, and just 147 to a connected one. This however does not indicate a preference for viewing actors that are not highlighted, but happened because users had many more opportunities to view unhighlighted actors than they had to view highlighted ones. Intuitively, when a user transitions their gaze from a source to a target, the visualization typically contains many more targets that are not highlighted and are not connected to the source, than those that are.

Thus, observed transitions should be compared to the default case, which assumes that users treat all visual objects equally. Assume the following simplified case: a movie is connected to two of ten actors shown in a visualization. We observe that of ten transitions from that movie to one of the actors, five were to a connected actor, while five were to unconnected actors. The two observed probabilities, to connected and unconnected actors, would in this case be equal at $5/10 = 0.5$. However, if there was no transitioning preference, the probability of transitioning to any actor would be equal to 0.1, that of transitioning to a connected actor 0.2, while that of transitioning to an unconnected actor 0.8. Thus, our observed transition probability from a movie to a connected actor is $0.5/0.2 = 2.5$ times higher than the default, unbiased probability, while our observed transition from a movie to an unconnected actor is a fraction ($0.5/0.8 = 0.625$) of the unbiased one.

To compute unbiased probabilities, every time we counted a transition from a source to a target, we also counted all target options available to the subject at that point, given the state and structure of the visualization at the time of transition. Reverting to our simplified example, for each of our ten observed transitions we would count two possible transitions to connected actors and eight possible transitions to unconnected actors, ending up with 20 counts for connected actors, and 80 counts for unconnected actors. These numbers allow us to compute the two unbiased probabilities as $20/(20+80)$ and $80/(20+80)$.

Movie to		No. of transitions	Observed trans. prob.	Unbiased trans. prob.	Ratio $\frac{\text{Observed}}{\text{Unbiased}}$
Actor	-	793	0.445	0.898	0.495
	H	147	0.082	0.02	4.081
	C	228	0.128	0.052	2.473
	CH	616	0.345	0.03	11.484
Movie	-	5727	0.761	0.899	0.846
	H	1798	0.239	0.101	2.376
Director	-	304	0.537	0.887	0.606
	H	37	0.065	0.021	3.088
	C	51	0.09	0.055	1.647
	CH	174	0.307	0.038	8.176
Genre	-	193	0.33	0.792	0.417
	H	40	0.068	0.033	2.045
	C	69	0.118	0.102	1.159
	CH	282	0.483	0.072	6.693

Actor to		No. of transitions	Observed trans. prob.	Unbiased trans. prob.	Ratio $\frac{\text{Observed}}{\text{Unbiased}}$
Actor	-	4711	0.685	0.962	0.713
	H	2164	0.315	0.038	8.207
Movie	-	839	0.469	0.82	0.572
	H	213	0.119	0.058	2.046
	C	386	0.216	0.076	2.843
	CH	352	0.197	0.046	4.284
Director	-	68	0.701	0.959	0.731
	H	29	0.299	0.041	7.271
Genre	-	43	0.524	0.931	0.563
	H	39	0.476	0.069	6.918

Director to		No. of transitions	Observed trans. prob.	Unbiased trans. prob.	Ratio $\frac{\text{Observed}}{\text{Unbiased}}$
Actor	-	71	0.747	0.958	0.78
	H	24	0.253	0.042	5.964
Movie	-	271	0.494	0.792	0.623
	H	55	0.1	0.04	2.478
	C	130	0.237	0.108	2.198
	CH	93	0.169	0.06	2.841
Director	-	384	0.706	0.93	0.759
	H	160	0.294	0.07	4.216
Genre	-	256	0.522	0.899	0.581
	H	234	0.478	0.101	4.708

Genre to		No. of transitions	Observed trans. prob.	Unbiased trans. prob.	Ratio $\frac{\text{Observed}}{\text{Unbiased}}$
Actor	-	61	0.656	0.9791	0.67
	H	32	0.344	0.021	16.47
Movie	-	229	0.118	0.261	0.453
	H	46	0.024	0.008	3.001
	C	172	0.089	0.093	0.956
	CH	138	0.071	0.013	5.288
Director	-	282	0.591	0.973	0.608
	H	195	0.409	0.027	15.174
Genre	-	348	0.398	0.943	0.422
	H	526	0.602	0.057	10.627

TABLE 2: Transitions from a source object to a target object, divided by: (i) type of source and target; (ii) whether the target was highlighted (H); (iii) whether the target was highlighted and connected to the source (HC); (iv) and whether source and target were neither highlighted nor connected. Columns show: (i) the number of direct transitions for the source/target combination; (ii) the observed transition probability from the source to that target; (iii) the (unbiased) probability of transition between source and target if all elements had equal probability to be viewed; (iv) the ratio between observed and unbiased transition probabilities.

5 INSTRUMENTING VISUALIZATIONS IN PRACTICE

To assess the feasibility of using our approach in practice, we tested if the predictive algorithm could be packaged as a reusable object-detection library, and used to instrument interactive visualizations without considerable effort. We chose to target three visualizations developed using the popular D3 library [40], but note that our approach is platform independent. We were able to instrument the visualizations by adding to their code between 10 and 20 instructions (or $5-10\%$ of code). Such efforts, excluding the preliminary writing or understanding of a visualizations code, required approximately 20-30 minutes per visualization. All added instructions were generally straightforward, and directly mirrored the visualizations rendering and interaction commands. An example is shown in Figure 8.

We implemented the object-detection library as a lightweight Java HTML server which accepted calls from visualizations as HTML requests, and gaze samples from an eye-tracker via network protocols. The visualization, the object-detection server, and the eye-tracker ran on the same machine. This architecture allows any visualization with networking capabilities to use the instrumentation library, regardless of programming language or visualization environment. Our D3 visualizations contacted the object-detection library through AJAX calls.

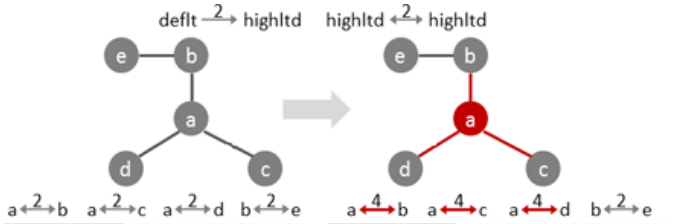
Our case study involved the instrumentation of three visualizations: a network that could be zoomed and panned, and whose nodes could be dragged and highlighted; a collapsible tree visualization from D3's examples repository; and a bar graph. Our evaluation was limited to detecting glyph-like objects (i.e., small objects that can be perceived with one or two fixations, such as nodes or labels), a limitation which we discuss in section 6.5.

Instrumenting visualizations involved augmenting their source code with three types of calls to the object-detection library (Figure 8). First, the library needed to translate gaze coordinates from screen space to the visualizations model space. The visualization code was augmented to detect when a visualizations window was moved or resized, and when the visualization itself was zoomed or panned, and to notify the object-detection library of such changes.

Second, the library needed to know the positions and sizes of objects that the visualization drew on the screen. Instrumentation code was added to mirror instructions used to add, remove, and update visual objects. We note that this workflow integrates particularly well with the add-remove-update pattern of D3.

Third, the library allowed visualizations to define transition probabilities between objects and manage them dynamically at run-time. We achieved this by creating groups of objects and specifying transition probabilities between those groups. Visual objects could be added or removed from a group, and transition probabilities between groups could change, at run-time. The top part of Figure 8 exemplifies the changes in transition probabilities once a user selects a node in a network visualization.

Identifying and quantifying appropriate transition probabilities may be difficult for some visualizations, given that a solid understanding of how users parse and interpret visualizations does not yet exist. Our instrumentations relied on informal assumptions (e.g., a highlighted object is more likely to be viewed than an un-highlighted one) and follow the similar approach of Salvucci et al. We have also shown in section 4 that our assumptions held for the IMDB visualization we analyzed. A more principled way to determine typical viewing patterns and sequences in a specific visualization is to run a pilot study using no transition probabilities. Preliminary data could then be used to determine typical usage patterns and help refine viewed object detection by informing the choice of appropriate transition factors. Section 4 shows how such an analysis could be done.



a 2 b a 2 c a 2 d b 2 e

obj	groups		a	b	c	d	e
a	a, deflt	a	-	2	2	2	1
b	b, deflt	b	2	-	1	1	2
c	c, deflt	c	2	1	-	1	1
d	d, deflt	d	2	1	1	-	1
e	e, deflt	e	1	2	1	1	-

a 4 b a 4 c a 4 d b 2 e

obj	groups		a	b	c	d	e
a	a, highltd	a	-	2x4	2x4	2x4	1
b	b, highltd	b	2x4	-	2	2	2
c	c, highltd	c	2x4	2	-	2	1
d	d, highltd	d	2x4	2	2	-	1
e	e, default	e	2	2x4	2	2	-

```
//transitions between highlighted and un-highlighted
//objects; any transition not set explicitly
//(e.g., default->default) is by default '1'
eyeTrack("setTransition_default_highlighted_2",
     "setTransition_highlighted_highlighted_2");

//svg is zoomed/panned; new transform is 't'
d3.select("#vis").attr("transform", t);
eyeTrack(["transform_",t]);

//create nodes
var nodes = g.selectAll(".node").data(nodeData)
 .enter().append("g")
   //... node properties are set
   .attr("id", function(d){
      //track the object, add it to its own group,
      //add it to the group of un-highlighted objects
      eyeTrack([["addObj",d.id,d.x,d.y,1010],
                ["setGroup",d.id,d.id],
                ["setGroup",d.id,"default"]]);
      return d.label;})

//create edges
var edges = g.selectAll(".link").data(edgeData)
    .enter().append("line")
      //... edge properties are set
      .style("stroke", function(d){
        //set a higher than default transition between
        //endpoints of an edge (or rather their groups)
        eyeTrack(["setTransition",d.start.id,d.end.id,2]);
        return "gray"; })

//a node is dragged
d3.selectAll("#"+node.id)
   .attr("cx", function(){return nod.x})
   .attr("cy", function(){return nod.y});
eyeTrack(["changeObj",nod.id,nod.x,nod.y,10,10]);

//hover a node
d3.select(this).style("fill","red");
d3.selectAll(".link").style("stroke", function(e){
    if (e.start == node || e.end == node){
      //once an edge highlighted: the transition between
      //its endpoints increases from 2 to 4; both its
      //endpoints are marked as highlighted (new transitions
      //will apply)
       eyeTrack([["setTransition",e.start.id,e.end.id,4],
          ["changeGroup",e.start.id,"default","highlighted",
          ["changeGroup",e.end.id,"default","highlighted"]);
       return "red";
    }
    else
        //not shown: reverting an edge to its normal state
        return "gray"; })})
```

Fig. 8: Instrumenting a D3 graph visualization. (Top) Changing transition-probabilities at run-time. Selecting a node marks it and its direct neighbors as highlighted; transitions between these nodes, and from un-highlighted nodes to these nodes, become more likely ($1 \rightarrow 2$), as do transitions on highlighted edges ($2 \rightarrow 4$). These rules translate to transitions between pairs of specific nodes; when multiple rules apply to a pair of nodes, their effects are compounded through multiplication. (Bottom) Three types of instrumentation calls mirror the visualizations rendering and interaction code to capture: view changes (|), addition and updating of visual objects (||), and management of transition probabilities (|||).

6 Discussion

6.1 Benefits and Applications

It took each of our coders about six hours to code eighteen minutes of user data (6 subjects $\times$ 3 minutes). Our instrumentation can reduce this overhead and makes it feasible to analyze eye-tracking data from many subjects, using highly interactive content, in long analysis sessions. Since manual coding is not required, data can be analyzed immediately after or even as it is collected. Moreover, this data has semantic meaning tied to the underlying

data of the visualization and can support broader analyses than traditional eye-tracking data. Such analyses, which we exemplified in Sections 4.4.2 and 4.4.3, focus on data and concepts, and differ from current eye-tracking analyses, which generally focus on understanding low-level visual perception in static visualizations.

As such, our approach can be useful in understanding how users forage for, integrate, and hypothesize about data using interactive visualizations. Visual analytics applications could be instrumented to facilitate the exploration of domain expert workflows, of how expertise influences data search and analysis patterns, and of visual strategies associated with successful hypothesis testing. Similarly, instrumenting visual learning environments could lead to insights into how students learn and what makes some learners more effective than others.

Data produced by our instrumentation can help explain how visualizations support analysis, discovery, and learning, and how they may be changed to be more efficient. Given a particular domain, it would allow us to quantify which data best answers which questions, what types of data are often used together, and how visual widgets are viewed in an analysis process. Section 4 exemplifies these types of quantitative and qualitative analyses.

Viewing data collected automatically during a user's session could be transformed into summaries that capture the user's activity during a day, week, or month. Such summaries may help refresh the user's memory at a later time, communicate progress to peers or supervisors, and provide useful hints to other users or analysts exploring similar questions in similar data-sets.

Finally, real-time viewed-object detection opens up two distinct opportunities. First, eye-tracking could be used in teaching. Using instrumented learning environments, instructors could track students' progress in lab assignments in real-time to detect students that are not tending to elements crucial for solving assigned problems, and to provide proactive help. Second, it would allow us to create a new generation of gaze-contingent visualizations that could detect in real-time data that is of interest to a user and make recommendations of unexplored data with similar attributes. The decreasing cost of eye-trackers makes it conceivable that they could be included in regular work stations and make such applications real.

6.2 Limitations

Our approach is restricted to visualizations with open source code and cannot be used to automate the full range of eye-tracking studies (e.g., analysis of real imagery or of commercial systems). This problem is inherent to any type of instrumentation. Capturing an application's interaction data, a website's activity, or a network's throughput requires privileged access to those systems. Our approach is intended primarily for creators or owners of data visualizations who wish to understand how their visualizations are used and how to make them more efficient.

Instrumenting a visualization by altering its source code and defining viewing-transition probabilities involves an overhead. This is also a general instrumentation problem that requires a case by case decision, by considering the tradeoff between the overhead of instrumenting a specific system and the benefits of collecting data from it. We showed in section 5 that bundling the predictive algorithm into a reusable instrumentation library reduces the cost of instrumentation.

Finally, as discussed in section 5, picking the right parameters to our predictive algorithm may currently be difficult and require pilot studies. However, we believe visualizations are rarely used in a random fashion and that specific tasks and visual outputs elicit certain gaze patterns. First, most visualizations have a mechanism for highlighting specific elements through interactions or queries, and we showed that such highlighting impacts how people view objects. Second, while we studied connected elements in the context of graphs, establishing visual connections between elements is also employed in brushing and linking or when using leader lines. Third, data in most real-life visualizations can be divided into semantic groups (e.g., actors, movies, directors; protein kinases, protein receptors; conference papers, journal papers) and we hypothesize that viewing transitions between and within such groups are also not random. Fourth, we showed that users identify data connected to their task and then shift their attention repeatedly and almost exclusively within that data group. We hypothesize further research can lead to a more general understanding of such transition probabilities and provide prescriptive guidelines for choosing the parameters that our predictive algorithm relies on. Moreover, our conceptual framework facilitates exactly this type of unexplored questions in ways previously not possible.

6.3 Performance gains by using a predictive approach

While in our evaluation the gain from using the predictive approach was relatively small (5%), we think that benefits are highly dependent on the type of visualizations that are instrumented, and that some visualizations will benefit more from the predictive approach. This is suggested by results shown in Table 2, which reveal very strong biases in how people use visualizations (e.g., up to 11 times more likely to view highlighted objects connected to previously viewed object, than to view random objects).

The degree to which such viewing patterns can and need to be leveraged predictively depends on the visualization. For example, in our particular case study, the different data categories (i.e., movies, directors, actors, genres) were spatially separated in different panels. As such, if a gaze landed between multiple data objects, these were generally of the same type and our algorithm could not use object category as a discriminator. Instead, in a traditional node link diagram, multiple types of nodes share the same space, and are distinguishable by specific visual attributes or semantic meaning (e.g., proteins in a protein interaction network can be kinases, receptors, etc.). In such a case, an algorithm could use knowledge that a user is currently scanning for a particular type of node, to distinguish between the viewing of nodes that are co-located but from different groups.

Generally, a visualization will benefit from our predictive approach if cluttered heterogeneous content shares the same space, and the visualization provides visual and semantic cues that allow users to select subsets of data that are relevant to particular tasks. Such visualizations are fairly common in real-life analytic applications. If the visual content is sparse and well separated, then computing gaze scores alone is likely sufficient.

6.4 Evaluating viewed object detection

The dependence on visualization type makes it hard to assess the impact of the predictive method. Moreover, comparing the output of the predictive algorithm to human annotations is questionable since, if coders look at momentary gaze positions rather than try to understand what users aim to do more broadly, their annotation may be closer to our our simpler, probabilistic detection. This raises an issue about whether human coders can provide a robust

ground truth for evaluating techniques such as ours, and whether such ground truths could be improved if eye-tracking data was collected in conjunction with a think-aloud protocol.

First, we note that the quantitative evaluation described in Section 4 is one against the state-of-the-art rather than against a ground truth. In other words, we don't claim that our method produces results that are accurate with respect to what people actually looked at, but claim that our method allows us to analyze the data in the same way a human could, only much faster.

Second, we believe that striving towards a reliable ground truth is slightly misguided in the context of evaluating eye-tracking instrumentation. People often view elements without consciously realizing it, since vision is by-and-large a subconscious process [4]. People are also able to register multiple objects in a fixated region, while not fixating any one object specifically. For example, while reading people often skip short words or syllables, while still registering that they are there. Moreover, for specific tasks, people may think about multiple objects as single data units of analysis. For example, a user of a graph visualization might think in terms of nodes for some tasks (e.g., are two nodes connected?) but may reason in terms of clusters of nodes or cliques for other tasks (e.g., what is the largest clique in the graph?). As such, we believe that we can generally capture only what subjects see, and that obtaining a ground truth of what subjects actually look at is either unattainable or, if obtained through think allowed protocols or constrained tasks, would not be ecologically valid. As shown by Ogolla, concurrent think-aloud protocols change visual patterns, especially for exploratory tasks [41].

6.5 Future Work

Our work demonstrates and quantifies the existence of gaze transition biases in visualization, but exploring their prevalence remains an open question that is beyond the scope of this paper. Section 6.2 lists multiple potential gaze transition patterns that future research can explore, and our framework provides the necessary tools for such work. This research would both deepen our understanding of how visualizations are used and provide guidelines for choosing appropriate inputs to our algorithms.

More work is also needed to understand the impact of different parameters involved in viewed object detection. For example, how far away from an item can a user fixate and still be considered to be viewing the item? The parameter that captures this in our algorithm is R, and, while we use a constant R for all items, this is unlikely the best approach. Based on qualitative observations in the data we collected, and knowledge of the interplay between peripheral vision and the fovea [42], we believe users fixate close to items if they are surrounded by clutter, but exhibit significantly more variability if items are isolated. Thus, we hypothesize that R should adjust itself dynamically based on the clutter of the region that a user is fixating.

Additional work is also needed to understand how to detect visual objects other than nodes or labels, such as for instance polylines in a parallel coordinate plot, contours in a group or set visualization, or cells in a heatmap. Since distances from gaze samples to objects can be computed even for objects with irregular shapes, Formula 1 could be used to compute gaze scores (gs) in such cases as well. However, it is unclear whether this is the best approach, since it is not known exactly how people parse large or complex objects. Do they fixate uniformly across the shape's entire area, and if so, how many fixations are enough to deem an object viewed? Are more fixations needed for irregular shapes as compared to common shapes like ovals or rectangles? For example, we showed previously that we could detect the viewing of edges in a network visualization, but that this required a more complex way of computing gs scores, which relied on an understanding of how users parse lines visually [30].

7 Conclusion

In visualizations that are open to instrumentation, gaze information provided by an eye-tracker can be used to automatically detect what visual objects users are likely to be viewing. Such detection can provide results that are almost as accurate as annotations created by human coders, provided that detection is done "intelligently", by using gazes together with a prediction of which objects are likely to be viewed at a given time. Data collected in this way is highly granular and has semantic content because it is linked to the data underlying the visualization. For this reason, and because it does not require any human pre-processing, object viewing data can be collected and analyzed efficiently for many subjects, using interactive visualizations, for long analytic session, and could be used in studies that explore how analysts hypothesize about data using complex visual analytics systems.

References

[1] C. Ware and H. H. Mikaelian, "An evaluation of an eye tracker as a device for computer input2," *ACM SIGCHI Bulletin*, vol. 18, no. 4, pp. 183–188, 1987.

[2] R. J. Jacob, "The use of eye movements in human-computer interaction techniques: what you look at is what you get," *ACM Transactions on Information Systems (TOIS)*, vol. 9, no. 2, pp. 152–169, 1991.

[3] K. Rayner, "Eye movements and cognitive processes in reading, visual search, and scene perception," *Studies in Visual Information Processing*, vol. 6, pp. 3–22, 1995.

[4] A. T. Duchowski, *Eye tracking methodology: Theory and practice*. Springer, 2007, vol. 373.

[5] D. D. Salvucci, "Inferring intent in eye-based interfaces: tracing eye movements with process models," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 1999, pp. 254–261.

[6] D. D. Salvucci and J. R. Anderson, "Intelligent gaze-added interfaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2000, pp. 273–280.

[7] L. Sesma, A. Villanueva, and R. Cabeza, "Evaluation of pupil center-eye corner vector for gaze estimation using a web cam," in *Proceedings of the symposium on eye tracking research and applications*. ACM, 2012, pp. 217–220.

[8] K. Guo, H. Shaw *et al.*, "Perceiving facial expression from varying face orientations: an eye-tracking study," 2014.

[9] J. R. Shasteen, N. J. Sasson, and A. E. Pinkham, "Eye tracking the face in the crowd task: why are angry faces found more quickly?" *PloS one*, vol. 9, no. 4, p. e93914, 2014.

[10] T. Vervoort, Z. Trost, K. M. Prkachin, and S. C. Mueller, "Attentional processing of other's facial display of pain: An eye tracking study," *Pain*, vol. 154, no. 6, pp. 836–844, 2013.

[11] S. Kim, L. J. Lombardino, W. Cowles, and L. J. Altmann, "Investigating graph comprehension in students with dyslexia: An eye tracking study," *Research in developmental disabilities*, vol. 35, no. 7, pp. 1609–1622, 2014.

[12] A. M. Zawoyski, S. P. Ardoin, and K. S. Binder, "Using eye tracking to observe differential effects of repeated readings for second-grade students as a function of achievement level," *Reading Research Quarterly*, 2014.

[13] R. E. Mayer, "Unique contributions of eye-tracking research to the study of learning with graphics," *Learning and instruction*, vol. 20, no. 2, pp. 167–171, 2010.

[14] T. van Gog and K. Scheiter, "Eye tracking as a tool to study and enhance multimedia learning," *Learning and Instruction*, vol. 20, no. 2, pp. 95–99, 2010.

[15] C. Conati, V. Aleven, and A. Mitrovic, "Eye-tracking for student modelling in intelligent tutoring systems," *Design Recommendations for Intelligent Tutoring Systems*, vol. 1, p. 229, 2013.

[16] M. Pohl, M. Schmitt, and S. Diehl, "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*. Eurographics Association, 2009, pp. 49–56.

[17] W. Huang, P. Eades, and S.-H. Hong, "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*. ACM, 2008, p. 3.

[18] W. Huang and P. Eades, "How people read graphs," in *proceedings of the 2005 Asia-Pacific symposium on Information visualisation-Volume 45*. Australian Computer Society, Inc., 2005, pp. 51–58.

[19] M. Burch, N. Konevtsova, J. Heinrich, M. Hoeferlin, and D. Weiskopf, "Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 2440–2448, 2011.

[20] M. Burch, G. Andrienko, N. Andrienko, M. Hoferlin, M. Raschke, and D. Weiskopf, "Visual task solution strategies in tree diagrams," in *Visualization Symposium (PacificVis), 2013 IEEE Pacific*. IEEE, 2013, pp. 169–176.

[21] S.-H. Kim, Z. Dong, H. Xian, B. Upatising, and J. S. Yi, "Does an eye tracker tell the truth about visualizations?: findings while investigating visualizations for decision making," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 12, pp. 2421–2430, 2012.

[22] T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, and T. Ertl, "State-of-the-art of visualization for eye tracking data," *EuroVis2014*, 2014.

[23] C. M. Privitera and L. W. Stark, "Algorithms for defining visual regions-of-interest: Comparison with eye fixations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 9, pp. 970–982, 2000.

[24] A. Santella and D. DeCarlo, "Robust clustering of eye movement recordings for quantification of visual interest," in *Proceedings of the 2004 symposium on Eye tracking research & applications*. ACM, 2004, pp. 27–34.

[25] G. Drusch, J. C. Bastien, and S. Paris, "Analysing eye-tracking data: From scanpaths and heatmaps to the dynamic visualisation of areas of interest," *Advances in Science, Technology, Higher Education and Society in the Conceptual Age: STHESCA*, vol. 20, p. 205, 2014.

[26] B. Steichen, G. Carenini, and C. Conati, "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*. ACM, 2013, pp. 317–328.

[27] K. Kurzhals, F. Heimerl, and D. Weiskopf, "Iseecube: visual analysis of gaze data for video," in *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 2014, pp. 351–358.

[28] S. Stellmach, L. Nacke, and R. Dachselt, "3d attentional maps: aggregated gaze visualizations in three-dimensional virtual environments," in *Proceedings of the international conference on advanced visual interfaces*. ACM, 2010, pp. 345–348.

[29] M. Bernhard, E. Stavrakis, M. Hecher, and M. Wimmer, "Gaze-to-object mapping during visual search in 3d virtual environments," *ACM Transactions on Applied Perception (TAP)*, vol. 11, no. 3, p. 14, 2014.

[30] M. Okoe, S. S. Alam, and R. Jianu, "A gaze-enabled graph visualization to improve graph reading tasks," in *Computer Graphics Forum*, vol. 33, no. 3. The Eurographics Association and Blackwell Publishing Ltd., 2014, pp. 251–260.

[31] ——, "Using eye-tracking as interactive input enhances graph visualization," in *IEEE Vis Poster Track*, vol. 2, 2013, p. 8.

[32] D. C. Richardson and R. Dale, "Looking to understand: The coupling between speakers' and listeners' eye movements and its relationship to discourse comprehension," *Cognitive science*, vol. 29, no. 6, pp. 1045–1060, 2005.

[33] M. Burch, A. Kull, and D. Weiskopf, "AOI rivers for visualizing dynamic eye gaze frequencies," in *Computer Graphics Forum*, vol. 32, no. 3pt3. Wiley Online Library, 2013, pp. 281–290.

[34] J. H. Goldberg and X. P. Kotval, "Computer interface evaluation using eye movements: methods and constructs," *International Journal of Industrial Ergonomics*, vol. 24, no. 6, pp. 631–645, 1999.

[35] G. Andrienko, N. Andrienko, M. Burch, and D. Weiskopf, "Visual analytics methodology for eye movement studies," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 12, pp. 2889–2898, 2012.

[36] N. Weibel, A. Fouse, C. Emmenegger, S. Kimmich, and E. Hutchins, "Let's look at the cockpit: exploring mobile eye-tracking for observational research on the flight deck," in *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 2012, pp. 107–114.

[37] M. Kumar, J. Klingner, R. Puranik, T. Winograd, and A. Paepcke, "Improving the accuracy of gaze input for interaction," in *Proceedings of the 2008 symposium on Eye tracking research & applications*. ACM, 2008, pp. 65–68.

[38] M. Dork, N. Henry Riche, G. Ramos, and S. Dumais, "Pivotpaths: Strolling through faceted information spaces," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 12, pp. 2709–2718, 2012.

[39] J. Nielsen, "Information foraging: Why google makes people leave your site faster," http://www.nngroup.com/articles/information-scent/, 2003, [Online; accessed 30-June-2003].

[40] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 2301–2309, 2011.

[41] J. A. Ogolla, "Usability evaluation: Tasks susceptible to concurrent think-aloud protocol," Master's thesis, Linkoping University, 2011.

[42] B. Balas, L. Nakano, and R. Rosenholtz, "A summary-statistic representation in peripheral vision explains visual crowding," *Journal of vision*, vol. 9, no. 12, pp. 13.1–18, 11 2009.

Sayeed Safayet Alam is a PhD candidate at Florida International University. He received B.Sc. in Computer Science and Engineering from Bangladesh University of Engineering and Technology. His research interests are in applications of eye-tracking to data visualization, and visual analytics.

Radu Jianu received The PhD degree in computer science from Brown University. He is currently an Assistant Professor in the School of Computing and Information Sciences at Florida International University. His research interests are in visualization evaluation, applications of eye-tracking to data visualization, and applications of visualization to disaster management and mitigation.