A gaze-directed lens for touchless analytics

Abhishek Chander
Computer Laboratory
University of Cambridge
ac839@cam.ac.uk

Advait Sarkar
Computer Laboratory
University of Cambridge
advait.sarkar@cl.cam.ac.uk

Abstract

We present the design and implementation of a gaze-directed lens tool for the interactive exploration of quantitative chart visualisations. In view of visual analytics as end-user programming, we present lenses which enable the selective display of labels, as well as interaction with parameterised uncertainty. We describe an algorithm for smoothing the movement of the lens. We report a controlled, within-subjects experiment on 11 participants demonstrating that participants were as adept at moving the lens with their gaze as they were using a mouse, and furthermore, that the gaze-directed interface promoted greater inspection of the key regions of the graphs.

1. Introduction

Visual analytics can be viewed as an instance of end-user programming. Here, the "programs" being written are not represented as textual source code, but rather the instances of charts, graphs, and other visualisations which arise as (often transient) results of exploring a dataset using a visualisation tool. The process of interacting with the tool is analogous to the act of programming, in the sense that each transient visualisation embodies a procedure for transforming data.

Traditional mouse-and-keyboard interfaces, or even touchscreen interfaces, are problematic in a number of situations. For instance, shared public display walls, where each individual user cannot be given mice and keyboards, and touchscreens cannot be implemented for reasons of cost and robustness; or operating theatres, where sterility is a primary concern (O'Hara et al., 2014; O'hara, Harper, Mentis, Sellen, & Taylor, 2013; Perry, Beckett, O'Hara, & Subramanian, 2010); or where the target end-user has impaired motor skills which prevents them from using conventional input mechanisms. In the era of ubiquitous analytics, similar situations demand tools for the visual exploration of data. In the touchless scenarios previously described, eye movement-based interaction provides a promising solution (Jacob, 1990). *Lenses* are interactive overlays which serve a variety of functions for data analytics, including zooming, filtering, manipulation of representation, and changing the level of detail (Tominski, Gladisch, Kister, Dachselt, & Schumann, 2014). Consequently, it seems appropriate to apply eye-tracking to the movement of lenses for visual analytics.

Eye-tracking is an important approach to investigating the psychology of programming. For instance, it has been used to assess program comprehension (Bednarik & Tukiainen, 2006), syntax highlighting (Sarkar, 2015), and visual attention (Bednarik & Tukiainen, 2004) amongst other things. With respect to the view of visual analytics as end-user programming, and that eye-tracking is an important methodology for program comprehension understanding, this work makes the following contributions:

1. We present a purely gaze-directed lens tool, designed such that eye movements can be multiplexed for use either to inspect the graph or to move the lens.
2. We describe an algorithm for dynamically smoothing raw eye-tracking coordinates which provides results superior to simple exponential averaging or PID control.
3. We report the evaluation of our tool through a controlled study of 11 participants, showing that it is as easy to manipulate as a mouse-based lens and promotes inspection of key regions.

2. Related work

Much previous work has applied eye-tracking to the movement of lenses, albeit not specifically for visual analytics. A central problem from which gaze-directed analytical tools suffer is the *gaze multiplexing problem*, sometimes known as the *Midas touch* problem: eye tracking applications have to guess whether the user is reading, intends for a lens to move, or engage a selection, etc., and act accordingly. It is tricky to infer intent from eye movements alone; for instance, when trying to read a label placed along the edge of a gaze-driven lens, the user might inadvertently move the lens itself because the centre of their gaze has shifted.

To address the problem of gaze multiplexing, state-of-the-art solutions often rely on temporal multiplexing (i.e., changing modes based on dwell time) to infer intent, at the cost of interactional fluidity (Ashmore, Duchowski, & Shoemaker, 2005; Lankford, 2000). Others still use multimodal input (e.g., through head position, touch, and keyboard) to improve the experience and reliability of lens tools (Stellmach, Stober, Nürnberger, & Dachselt, 2011; Stellmach & Dachselt, 2013; Spindler, Büschel, & Dachselt, 2012; Lehmann, Schumann, Staadt, & Tominski, 2011; Kumar, Paepcke, & Winograd, 2007), at the cost of touchless interaction.

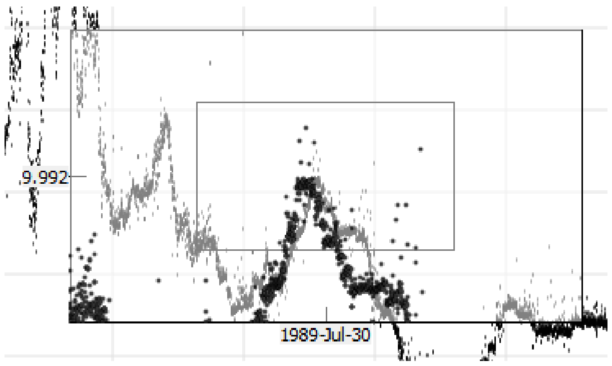


Figure 1 – Flat magnification lens.

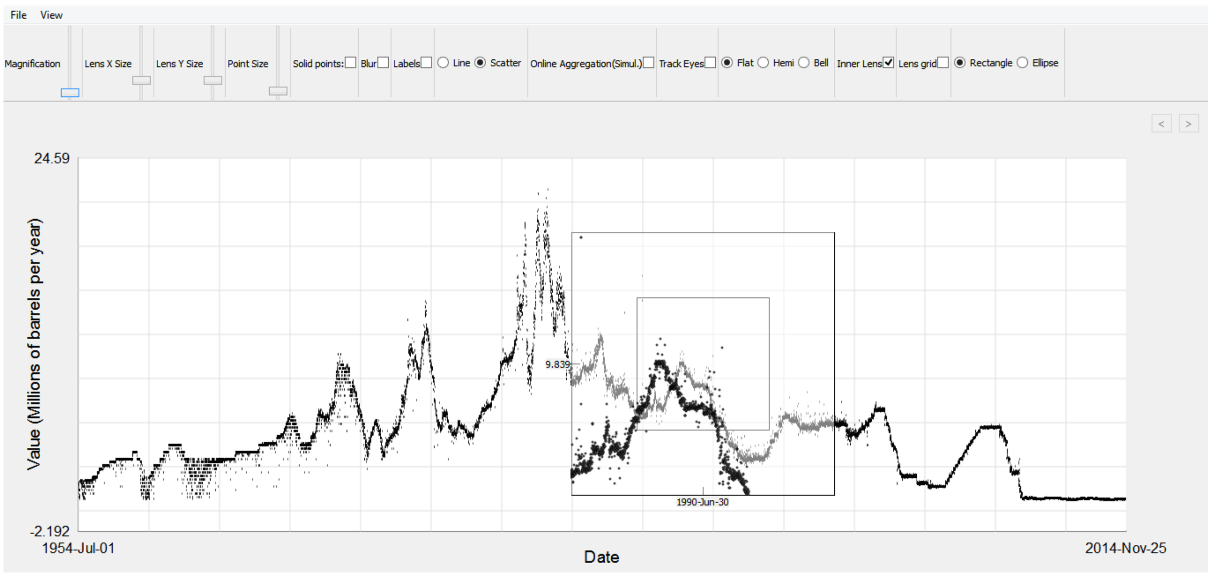


Figure 2 – Our prototype, showing a 2x magnification lens. The toolbar, which can be hidden, shows various lens parameters.

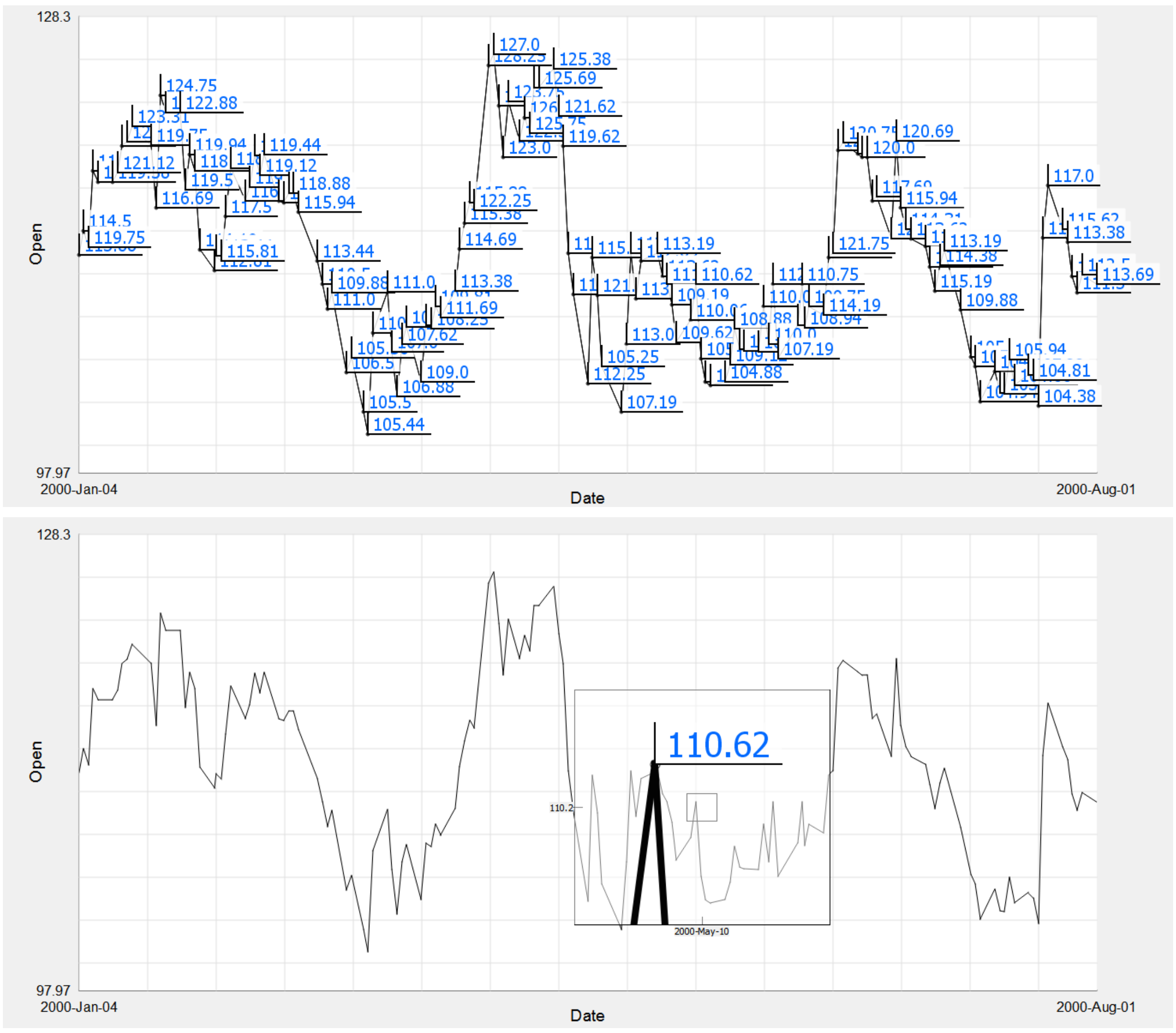


Figure 3 – A magnifying lens for reducing label clutter. Top: all labels turned on. Bottom: Labels shown only within lens.

3. Flat Lens Design

Figure 1 shows a magnification lens. Our magnification lens is flat; i.e., it linearly scales the underlying axes. Thus, it avoids the interpretability issues caused by fisheye lenses, which warp the underlying axes and are thus less suitable for scientific analysis (Carpendale, 1999). However, this also loses a key advantage of fisheye lenses, namely that no part of the graph is obscured, despite extreme warping at the lens edge. Apart from magnification, we also implemented a label lens (Figure 3) which only shows labels for points within the lens, reducing clutter in dense plots.

To achieve linear magnification without warping or obscuring parts of the graph, we use a two-part lens: an inner box indicates what part of the graph is being magnified, and a translucent outer box contains the magnified graph. The lens is necessarily translucent, otherwise parts of the graph would be obscured. The underlying graph is also visually de-emphasised by using a grey stroke so that the magnified graph, which uses a black stroke, is more prominent.

It is also necessary for these boxes to be contained in one other and to have a common centre, because otherwise trying to inspect parts of the magnified graph would move the lens, changing the region being magnified. With a common centre, gaze location signals the user's unambiguous intent to magnify a region *and* inspect it, greatly alleviating the gaze multiplexing problem.

4. Gaze data smoothing

The eye tracking data is extremely jittery and unusable in its raw form, even if the calibration is good. When the user is staring at a fixed point on the screen, deviations in the received gaze coordinates can have an amplitude of up to 50 pixels. A lens configured to be centred around the raw coordinates in real time would be unusable.

Exponential averaging is a well-known technique for smoothing jittery eye-tracking data (Wojciechowski & Fornalczyk, 2014). However, basic exponential smoothing was unable to meet our requirements as low values of α, the memory coefficient, produced smooth outputs, but incurred a large latency when moving the lens across the graph. Higher values of α caused jittery output when attempting to focus within the lens. Our implementation of a Proportional-Integral-Derivative controller with Ziegler-Nichols tuning (Ziegler & Nichols, 1942) also produced unsatisfactory results.

4.1. Dynamic Exponential Smoothing (DES)

We modified our approach to use a dynamic value for α, scaled by the difference between consecutive values of the gaze location. In the following, x_t denotes the x-coordinate received from the eye-tracker at timepoint t, and s_t denotes the output of our smoothing algorithm, on which the lens is centred:

$$\beta_t = \frac{|x_t - s_{t-1}|}{chart\ width} \tag{1}$$

$$\alpha_t = \begin{cases} \beta_t & \text{if } \beta_t < 0.05 \\ min(1.0, f \cdot \beta_t) & \text{otherwise.} \end{cases} \tag{2}$$

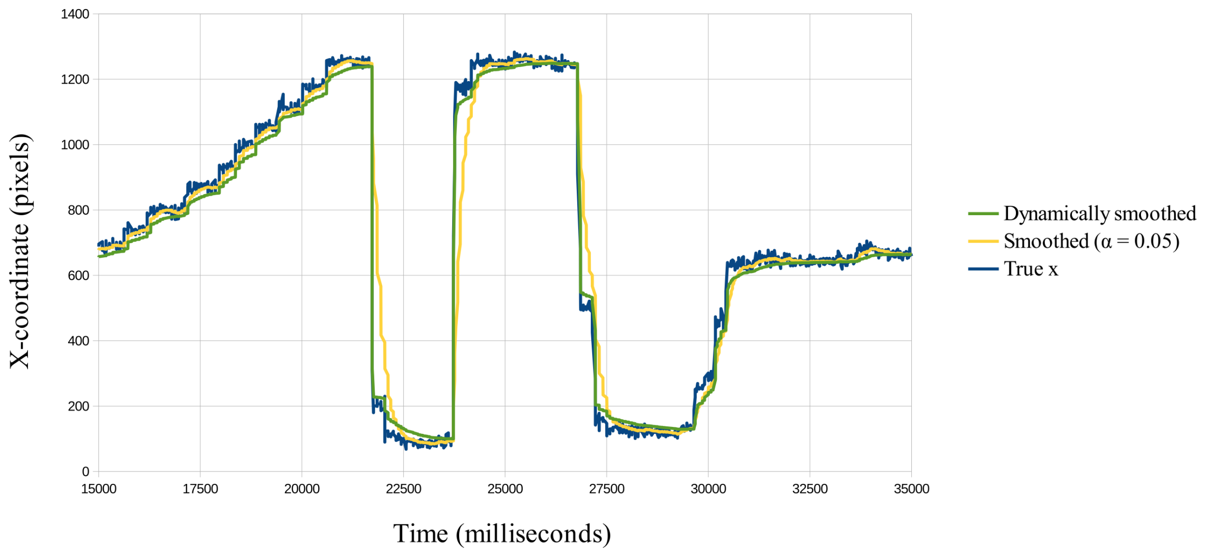
$$s_0 = x_0 \tag{3}$$

$$s_t = \alpha x_t + (1-\alpha)s_{t-1}, t > 0 \tag{4}$$

A similar set of equations (with β dependent on chart height instead of width) determines the smoothed y-coordinate. The variable f is an "expansion factor," which controls how quickly α grows as the gaze location moves further from the current smoothed location. This technique results in smoothing behaviour which is much better than basic exponential averaging or PID control in terms of maintaining smoothness, as well as quickly acquiring distant targets.

Figure 4 shows the effects of different smoothing strategies on a trace of the x-coordinate of users' gaze when looking at points on the left, centre and right of the screen. Observe how DES catches up with large shifts in user's gaze (i.e., the user is looking at another part of the graph and so the lens must move) while maintaining stability during small shifts (i.e., the user is inspecting different parts of the lens contents but does not wish the lens to move). Simple exponential smoothing suffers either from lag or excess jitter, and PID control sufferers from both lag as well as "overshooting" the mark for large shifts in gaze location.

DES, like PID, is based on the principle that large shifts in the quantity being controlled should cause the controller to *accelerate* faster than small shifts (i.e., second-order control), but unlike PID, DES is altered to better fit the specific requirements of gaze data smoothing. Stellmach et al. (Stellmach & Dachselt, 2013) present a similar idea, where lenses have invisible concentric 'zones'—the lens is not moved when gaze coordinates lie within the innermost zone, allowing uninterrupted examination of the lens contents; within the intermediate zone, the lens is moved faster as the gaze drifts further away from the innermost zone; within the outermost zone, the lens is positioned absolutely. DES is an evolution of this idea, but framed in terms of the user's gaze location instead of in terms of lens coordinates, and also provides smooth, continuous control rather than shifting control based on hard boundaries, which can introduce unpredictable threshold effects for the user.



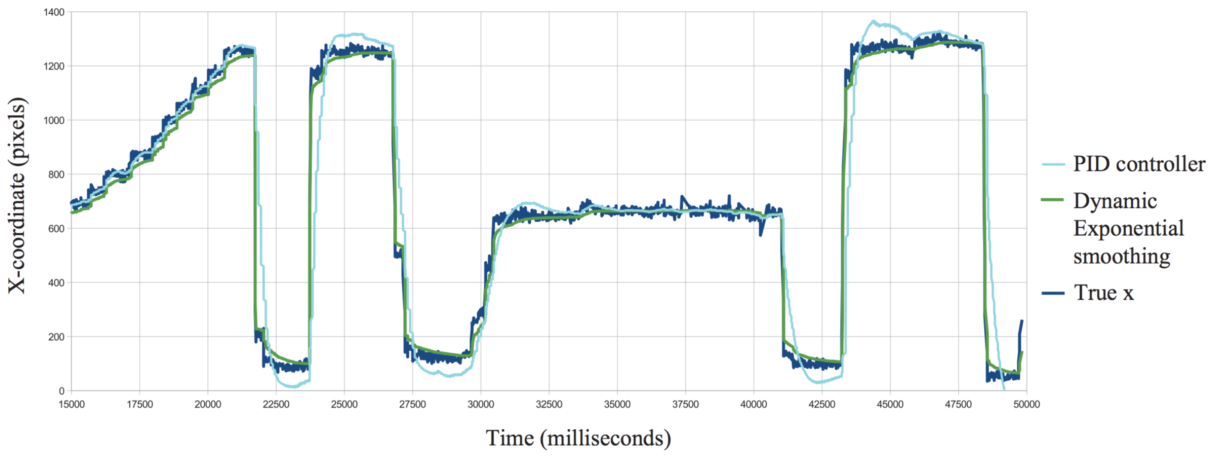


Figure 4 – Comparison of DES against basic exponential smoothing (top) and PID control (bottom).

5. A lens for interacting with approximate computations

In previous work, we have shown how it is possible to enable users to interact with and control uncertainty associated with individual data points, where that uncertainty has arisen out of an approximate computation process, such as sampling (Sarkar, Blackwell, Jamnik, & Spott, 2014). The motivation is that it can be much faster, and consequently more interactive, to rapidly render a graph based on approximate values. However, it is an open question as to how these approximate values may be interactively refined. For instance, we have previously proposed that data points may be displayed with error bars, which can themselves be dragged to adjust the level of uncertainty associated with a point; reducing the size of an error bar triggers recomputation to a higher degree of accuracy (Sarkar, Blackwell, Jamnik, & Spott, 2015).

This section describes a lens built as a tool for interaction with simulated uncertainty. The simulation can be used on any data set, because the inaccurate initial values are generated randomly from the true data. The core idea here is that the position of the lens directly indicates user attention; points on which the user's gaze dwells are recomputed with lower errors.

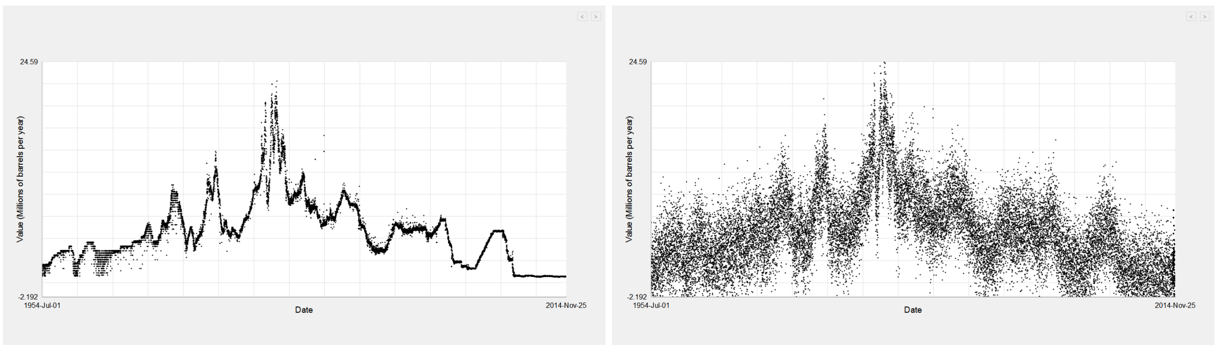
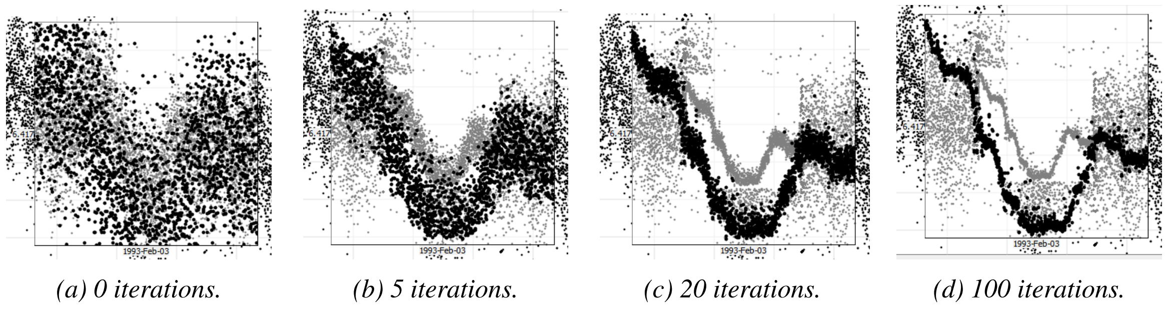


Figure 5 – A graph of 20,000 data points (L). Gaussian error applied to the y-values (R).



(a) 0 iterations. *(b) 5 iterations.* *(c) 20 iterations.* *(d) 100 iterations.*

Figure 6 – Error is reduced over time (left to right).

5.1. Error model

We add Gaussian noise to create errors in the true y-values.

$$e_i \sim \text{Gaussian}(0,\, \delta(y_{\max} - y_{\min})) \tag{5}$$

$$\hat{v}_i = v_i + e_i \tag{6}$$

where $y_{\min}$ and $y_{\max}$ are the minimum and maximum y-values in the dataset respectively, which scale δ to the current dataset; $\hat{v}_i$ is the approximation to v_i, the true i^{th} value. The value δ, which defaults to 0.1, can be varied to increase/decrease the maximum error throughout the dataset. This produces a “noisy” version of the dataset as shown in figure 5.

Points within the lens are recomputed iteratively. Error is reduced exponentially; in each iteration we set

$$\hat{v}_i := \theta \hat{v}_i + (1 - \theta) v_i \tag{7}$$

where θ is set to some fraction between 0.9 and 0.99, with higher values corresponding to slower convergence. We stop iterating and set our approximate value to be equal to the true value after 100 iterations, or after the onscreen distance between approximate and true positions of the data point becomes smaller than half a pixel. The visual effect of this procedure can be seen in Figure 6.

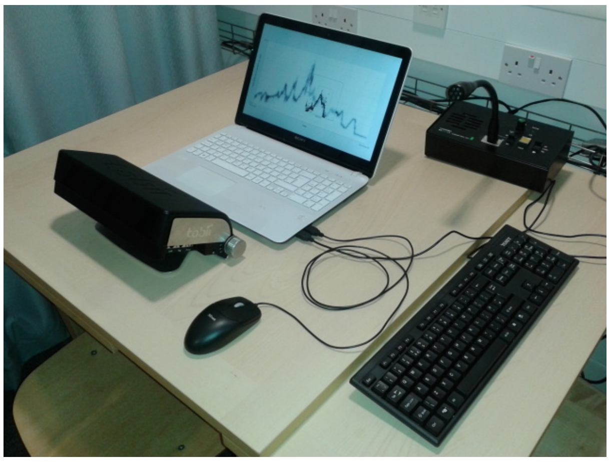


Figure 7 – Experimental setup.

6. Evaluation

We designed a controlled experiment to compare the usability of our gaze-directed lens tool versus a conventional mouse-driven interface. Our primary questions were, in the context of exploring graphs using lens interfaces:

1. Is it feasible (i.e., not inferior in terms of time required) to use a gaze directed lens as a direct replacement for a mouse?

2. Does an eye-tracking system improve exploration of a graph in terms of the proportion of time spent inspecting key regions?

6.1. Experimental design and procedure

To answer these questions, we conducted a study of 11 users, undergraduate students at the University of Cambridge, while they performed analytical exploration tasks, each separately using the mouse and eye tracker to guide the lens. The user studies were conducted using a Tobii[1] X120 remote eye tracker (Figure 7). The user sat in front of the eye-tracker. The experimenter sat to the side of the user, using an external keyboard to navigate through the questions.

We selected 30 openly-available datasets of univariate time series (Government, 2010; Quandl, 2015). Using our datasets, we framed unambiguous questions such as “What is the highest ever price of oil?”, “When did the market crash?”, etc. The participant was expected to seek peaks, troughs, and inflexion points on the graph in order to answer our questions. These questions could only be answered using the lens tool, as they either required a label value to be read off with the label lens, or a subtle feature to be discerned with the magnification lens.

We annotated each graph with “key regions,” such as peaks, troughs, and inflection points, which the user would need to consider in order to answer the question (Figure 8). We logged when gaze/mouse coordinates fell within these key regions, so as to compare investigation of key regions with the eye tracker versus with the mouse.

The level of difficulty was consistent across questions, and the question order was randomised for each participant. Each participant answered 30 questions; 15 using the eye-tracker to guide the lens, and the other 15 using the mouse. Half of the users began with the eye tracker; the other half with the mouse. This allowed us to perform a within-subjects comparison. The participants were instructed to read the question, investigate the graph and speak out their answer, upon which the experimenter would proceed to the next question.

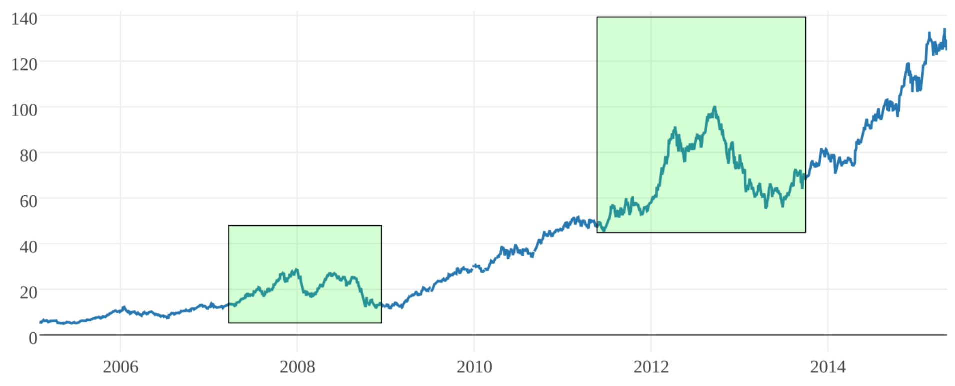


Figure 8 – An example chart highlighting some key regions (e.g., peaks, troughs, points of inflexion).

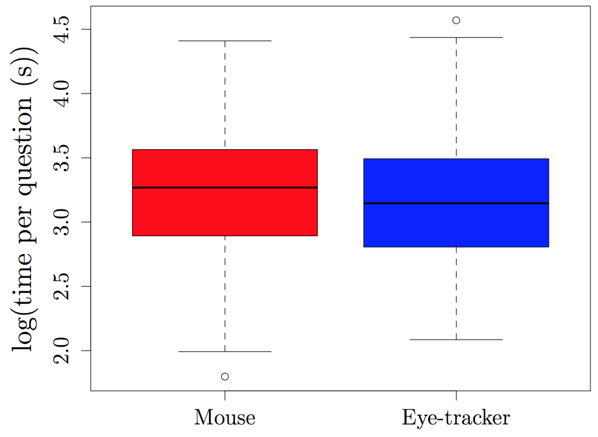


Figure 9 – Comparing time per question.

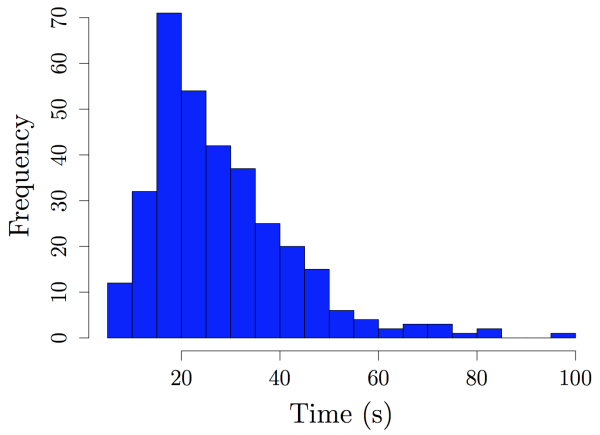


Figure 10 – Time per question (5s bins).

6.2. Results

Our timing data was log-normally distributed (Figure 10). Using log-normalised times, a non-inferiority test (two one-sided test) gives a confidence level of 94.4% using an epsilon value of 0.2 (for testing equivalence, a 90% confidence level yields a 0.05 significance level). This demonstrates that the gaze-directed interface is strictly not inferior in efficiency when compared to using a mouse (Figure 9). In fact, on average, participants spent 0.115s *less* per question with the eye-tracker than with the mouse.

Participants spent a median of 2.56s more time investigating key regions with the eye tracker than with the mouse (Wilcoxon rank sum test, $p = 8.63 \cdot 10^{-5}$). Thus, the eye-tracker promotes investigation of the the key-regions of the graph using the lens. However, the total time per question was lower using the eye tracker, indicating that when using an eye tracker, a greater proportion of the time spent answering the question was investigating the key regions. The median proportion of time spent investigating the regions using the mouse was 35.2%, whereas using the eye-tracker, this increased to 48%.

6.3. Evaluation of uncertainty reducing lens

To evaluate our lens for interacting with approximate computations, we conducted a think-aloud study with each user after they had completed the primary study. The users investigated using the uncertainty reducing lens on one of the datasets (namely, the dataset in Figure 5), without any prior information about the function of the lens. The users were requested to investigate the lens tool until they had formed a hypothesis regarding the function of the lens.

[1]`http://www.tobii.com`

Findings

The audio recordings from 11 users was transcribed and the average time the users spent on the dataset was 3 minutes and 51 seconds. The responses of the users are categorised as follows:

1. "Error": 5 out of the 11 users correctly stated that the tool is reducing the error, or "noise".

2. "Averaging": 4 out of the 11 users mentioned that the tool is performing some sort of averaging ("mean" or "aggregate") of the points. This might be due to the fact that the points were distributed bidirectionally using a normal distribution (section 5), and upon investigation, converged to their true values which on average could mean going up or down, as in figure 5.

3. The remaining 2 users gave physical descriptions (e.g., "the points are moving around") of what they observed as the lens was used on the data points, but were unable to give precise hypotheses of the cause of the movement of data-points.

Of our 11 participants, 5 (45%) of the users correctly identified the purpose of the lens without any prior description of approximate computation.

7. Conclusion

We view visual analytics as a form of end-user programming. We have presented the design for a gaze-directed lens for data exploration, which uses concentric, flat, translucent lenses to overcome the warping associated with fisheye lenses, which render them unsuitable for scientific analysis. We presented a design for how a lens might be used to control uncertainty arising from approximate computation. We have also presented dynamic exponential smoothing, a smoothing technique for gaze data which surpasses exponential averaging and basic PID control.

We conducted a user study of 11 participants, and found that our gaze-directed interface is as efficient as a mouse-based interface for analytical inspection of graphs. Furthermore, use of the eye-tracker promoted interactivity with the lens tool, by increasing the proportion of time spent by the user investigating the key regions of the graph. Finally, evidence from the think-aloud study suggests that the lens can be used as an intuitive interface for interacting with parameterised uncertainty.

8. References

Ashmore, M., Duchowski, A. T., & Shoemaker, G. (2005). Efficient eye pointing with a fisheye lens. In *Proceedings of graphics interface 2005* (pp. 203–210). School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society. Retrieved from `http://dl.acm.org/citation.cfm?id=1089508.1089542`

Bednarik, R., & Tukiainen, M. (2004). Visual attention and representation switching in java program debugging: A study using eye movement tracking. In *Proceedings of the 16th annual workshop of the psychology of programming interest group* (pp. 159–169).

Bednarik, R., & Tukiainen, M. (2006). An eye-tracking methodology for characterizing program comprehension processes. In *Proceedings of the 2006 symposium on eye tracking research & applications* (pp. 125–132).

Carpendale, M. S. T. (1999). *A framework for elastic presentation space*. Unpublished doctoral dissertation, Simon Fraser University.

Government, U. (2010). *Data.gov.uk*. `http://data.gov.uk`. (Last accessed July 1, 2016)

Jacob, R. J. (1990). 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* (pp. 11–18).

Kumar, M., Paepcke, A., & Winograd, T. (2007). Eyepoint: Practical pointing and selection using gaze and keyboard. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 421–430). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/1240624.1240692` doi: 10.1145/1240624.1240692

Lankford, C. (2000). Effective eye-gaze input into windows. In *Proceedings of the 2000 symposium on eye tracking research & applications* (pp. 23–27).

Lehmann, A., Schumann, H., Staadt, O., & Tominski, C. (2011). Physical navigation to support graph exploration on a large high-resolution display. In *Advances in visual computing* (pp. 496–507). Springer.

O'Hara, K., Gonzalez, G., Sellen, A., Penney, G., Varnavas, A., Mentis, H., … others (2014). Touchless interaction in surgery. *Communications of the ACM*, *57*(1), 70–77.

O'hara, K., Harper, R., Mentis, H., Sellen, A., & Taylor, A. (2013). On the naturalness of touchless: putting the "interaction" back into nui. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *20*(1), 5.

Perry, M., Beckett, S., O'Hara, K., & Subramanian, S. (2010). Wavewindow: public, performative gestural interaction. In *Acm international conference on interactive tabletops and surfaces* (pp. 109–112).

Quandl. (2015). *Quandl financial and economic data.* `https://www.quandl.com/`. (Last accessed July 1, 2016)

Sarkar, A. (2015). The impact of syntax colouring on program comprehension. In *Proceedings of the 26th annual conference of the psychology of programming interest group (ppig 2015)* (pp. 49–58).

Sarkar, A., Blackwell, A. F., Jamnik, M., & Spott, M. (2014). Hunches and sketches: rapid interactive exploration of large datasets through approximate visualisations. In *The 8th international conference on the theory and application of diagrams, graduate symposium (diagrams 2014)* (Vol. 1).

Sarkar, A., Blackwell, A. F., Jamnik, M., & Spott, M. (2015). Interaction with uncertainty in visualisations. In E. Bertini, J. Kennedy, & E. Puppo (Eds.), *Eurographics/IEEE VGTC Conference on Visualization (EuroVis 2015).* The Eurographics Association. doi: 10.2312/eurovisshort.20151138

Spindler, M., Büschel, W., & Dachselt, R. (2012). Use your head: Tangible windows for 3d information spaces in a tabletop environment. In *Proceedings of the 2012 acm international conference on interactive tabletops and surfaces* (pp. 245–254). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2396636.2396674` doi: 10.1145/2396636.2396674

Stellmach, S., & Dachselt, R. (2013). Still looking: Investigating seamless gaze-supported selection, positioning, and manipulation of distant targets. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 285–294). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2470654.2470695` doi: 10.1145/2470654.2470695

Stellmach, S., Stober, S., Nürnberger, A., & Dachselt, R. (2011). Designing gaze-supported multimodal interactions for the exploration of large image collections. In *Proceedings of the 1st conference on novel gaze-controlled applications* (p. 1).

Tominski, C., Gladisch, S., Kister, U., Dachselt, R., & Schumann, H. (2014). A survey on interactive lenses in visualization. *EuroVis State-of-the-Art Reports*, 43–62.

Wojciechowski, A., & Fornalczyk, K. (2014). Exponentially smoothed interactive gaze tracking method. In *Computer vision and graphics* (pp. 645–652). Springer.

Ziegler, J. G., & Nichols, N. B. (1942). Optimum settings for automatic controllers. *trans. ASME*, *64*(11).