EuroVis Workshop on Visual Analytics (2024)
M. El-Assady and H.-J. Schulz (Editors)

# Transient Visual Analytics

Hans-Jörg Schulz[1] and Chris Weaver[2]

[1]Department of Computer Science, Aarhus University, Denmark
[2]School of Computer Science, University of Oklahoma, USA

**Abstract**

*Visual Analytics often utilizes progression as a means to overcome the challenges presented by large amounts of data or extensive computations. In Progressive Visual Analytics (PVA), data gets chunked into smaller subsets, which are then processed independently, and subsequently added to a visualization that completes over time. We introduce Transient Visual Analytics (TVA), which complements this incremental addition of data with progressive removal of data as it becomes outdated, starts to clutter the visualization, and generally distracts from the data that is currently relevant to visual analysis. Through combinations of various progressive addition and removal strategies, and supported by suitable analogies for the analyst and the software engineer, TVA captures a variety of visual analysis scenarios and approaches that are not well captured by PVA alone.*

**CCS Concepts**

• ***Human-centered computing*** → ***Visual Analytics;*** • ***Computing methodologies*** → ***Progressive computation;***

## 1. Introduction

Progressive Visual Analytics (PVA) is a means of visualizing and analyzing large datasets one data chunk at a time [ASSS18, FFNE18, UAF*23]. By breaking up the data into smaller chunks that are easier to handle in terms of computation time and required memory, PVA is a suitable means to overcome long-running computational processes and memory limitations, so as to nevertheless facilitate fluid visual-interactive data analysis. PVA is a formidable way to funnel large amounts of input data into a visual analytics pipeline that delivers output data to views for interactive exploration even as the input data continues to arrive, the pipeline continues to process it, and/or the output data continues to accumulate. The existing literature reports some convincing examples that PVA indeed works well in practice, for example to query a large flight database [FPDs12] or to view large genomic datasets [CKBE19].

Yet there is a caveat: The graphical space available in a view can only accommodate a limited amount of data before it becomes visually cluttered. Similarly, the underlying data structure holding that data in memory, such as a DOM tree for an SVG output, can only grow so large before it becomes unwieldy for storage, transmission, and/or computation. As a result, most progressive visualizations use some form of data and/or visual aggregation—transforming it into, e.g., bar graphs [PRJ*23] or binned scatter plots [BEF17]. The aggregation can be tuned to show more or less detail while remaining within desired memory or screen space bounds, for example by adjusting bin size. The price of this strategy is that analysis tasks requiring access to the individual, disaggregated data items can no longer be performed. Such tasks are common and include detecting outliers in quantitative data, identifying critical nodes (i.e., articulation points) in network data, and discovering rare patterns or peaks in time series. In addition, foundational techniques commonly used in interactive visual analyses, such as linking & brushing between multiple coordinated views, become cumbersome to realize, as the individual data items are no longer present that would allow determining which parts of a chart to link with the corresponding parts of another chart.

In response to this challenge, this paper proposes a form of PVA that does not accumulate data without limits at the end of a progressive visual analytics pipeline in some buffer, be it a data structure, a view, or a combination of both. Instead, once a capacity threshold is reached, the progressive process adding more and more new data items is complemented by another progressive process that removes data items—a *regressive* process so to speak. The dataflow enacted by these two processes is shown in Figure 1. We call this combination of incremental progression and decremental regres-

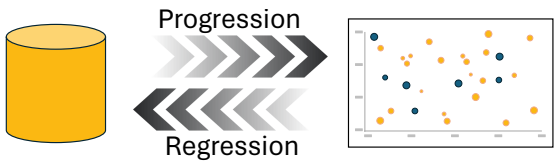


**Figure 1:** *The general dataflow between backend and frontend in Transient Visual Analytics. The progression of data added from the database to the view is complemented by an inverse process, the* regression*, that removes data from the view and "returns" them to the database.*

sion *Transient Visual Analytics (TVA)* as data items "pass through" the view but do not necessarily stay around indefinitely.

In this paper, we define the main idea of TVA (Sec. 2), look at different strategies for progression and regression (Sec. 3), propose analogies as possible mental models for TVA (Sec. 4 and Sec. 5), and consider the combination of PVA and TVA (Sec. 6).

## 2. Transient Visual Analytics in a Nutshell

We define Transient Visual Analytics (TVA) as the combined mechanism of (a) progressively adding data to a view for its interactive visual analysis and (b) progressively removing data from that view—for example, if the view becomes overcrowded or the data becomes stale. Removed data are "added back" to the database (or marked within the database as no longer visible) so that they can be added again later, should they once more become relevant to the analysis. In accordance with PVA, we call the loading/adding process *progression*, whereas we call the clearing/removing process *regression* to better differentiate them as two opposite yet complementary modes of data flow.

Given a large enough dataset that does not fit the available screen space or memory, the transient view of the data will thus always be intermediate and never complete, as the data are removed at some point to make space for other data. This is the most important difference between TVA and PVA: Whereas PVA aims to provide the human analyst with a gradually maturing view that converges towards the full view of all data over time, TVA aims to provide the right portions of the data at the right times throughout an analysis.

In doing so, TVA acknowledges that real-world analysis workflows are constantly in flux and shifting from one hypothesis to the next as new observations are made and insights are gained. To support this unpredictable nature of analyses in which different portions of the data are relevant at different stages, TVA shifts with the meandering analysis workflow by constantly adding data items currently deemed relevant and removing irrelevant ones that clutter the view without purpose or benefit at pertinent points in the workflow. Although steerable PVA [WM04, CKBE19, HASS22] is capable of adding relevant data with varying notions of relevance over the course of an analysis, it is not able or even intended to "garbage collect" data items after use. As a result, the view either clogs up with individual data items or, in anticipation of this problem, is designed to show aggregates.

The TVA process is governed by three factors: (1) the progression strategy by which data items are added, (2) the regression strategy by which data items are removed, and (3) the thresholding approach that determines when regression is triggered. These factors can change mid-analysis to account for a shift in user interest, e.g., switching from one progression strategy to another.

Possible progression strategies include, for example, uniform random sampling (e.g., for data to be shown in a scatter plot), reverse sequential order (e.g., for time series data starting with the most recent time point and then going backward to provide historical context), user-determined (e.g., through interactive steering [CKBE19, HASS22]), or something more involved like a custom sampling [HS24] or fetching data based on a degree-of-interest function [HMPS23].

Possible regression strategies are, for example, first-in, first-out (i.e., removal by age), user-determined (e.g., like the inverse to the steering approaches mentioned above, where regions of the data space that are furthest away from zoomed-in or selected regions get removed first), or a degree-of-invisibility (e.g., data items that are not visible due to overplotting by other data items). The regression can also remove items based on data characteristics (e.g., low data quality or high uncertainty). After all, if only a tiny sample of all data can fit the screen space, we want this sample to contain the least error-prone and most reliable data points to base subsequent analytic and real-world decisions on.

Possible thresholds at which the regression is triggered can be, for example, a limit of displayable items (i.e., a *visual entity budget* [EF10]), a limit of allowable clutter or data density (e.g., as captured by metrics such as those introduced by Bertini and Santucci [BS04] or Ellis and Dix [ED06]), or a time limit for how long a data item can be actually or effectively out of view (e.g., offscreen due to panning or at subpixel resolution due to zooming) before it gets removed.

A complicating factor for thresholds in TVA is how oscillations may occur around a threshold as the overall progression shifts back and forth between progressive adding and regressive removal. To smooth out these oscillations and reduce their influence on interaction and analysis, one can introduce a hysteresis in the form of two thresholds: a $limit_{min}$ below which the progression adds more data items and a $limit_{max}$ above which the regression removes data items, with $limit_{max} - limit_{min} > chunksize$. More generally, a threshold approach can apply start, pause, continue, and stop thresholds to progressive adding and/or regressive removal, and do so either independently or in concert.

## 3. Strategies for Progression and Regression

To unlock the full potential of TVA, it is important to note that the strategy that determines the (current) irrelevance of a data item does not necessarily have to be the inverse of the strategy that is used to determine its (current) relevance. In essence, different strategies for progression and regression can be combined to create a wide variety of transient behaviors. To illustrate this point, we provide four examples of such behaviors in the following list.

**Sliding Window** is a well-known transient pattern that combines a linear progression of chunking and processing data in sequence (e.g., temporal sequence) with a first-in, first-out regression strategy that removes the "oldest" data items to make space for new data items. The direction in which the window slides over the data can be user-steerable, so that a pan to the left or to the right adds data in that direction of the sequence and removes data items from the respective other end of the window. While by itself such a sliding window may not be notable, it is not far-fetched to imagine combining multiple sliding windows to create powerful progressive analysis setups like progressive *ChronoLenses* [ZCPB11]. One could even create "smart" sliding windows in which one or both ends are pinned or interactively snap to progression or regression threshold crossing points in the data processing sequence.

**Reactive Visualization** is a progressive visualization that combines a progression of levels of detail (e.g., hierarchical data be-

ing linearized and chunked in breadth-first order) with a last-in, first-out regression strategy that removes the "latest"—i.e., the most detailed—data items first. Such a (Level-of-Detail) LoD-based progression has already been suggested for *Progressive Treemaps* [RH09] to output exactly the right amount of information that the available drawing space can sensibly show, and then stop the progression. By adding the regression to this procedure, the progression can be *temporarily* halted once the drawing space is sufficiently full. It can later be restarted to progressively add more levels of detail whenever the user enlarges the view or to regressively remove the previously added layers of detail whenever the user shrinks the view down. Thus, the LoD of the displayed information will dynamically adapt to the available drawing space. Another scenario in which this mechanism would be useful is a progressive overview & detail visualization design in which two views—a small minimap (the overview) and a larger cutout (the detail view)—are both provisioned with data by the same progression, but have different thresholds for how much data can be shown. The threshold for each view could be yoked relative to the progression and regression trigger thresholds for the data itself. A series of intermediate views showing multiple levels of detail could have fixed or interactively adjustable thresholds that are distributed between the thresholds of the overview and detail views.

**Local Dampening** is a combination of a random uniform progression and a regression that removes data items from very dense regions in the view. Together, they can be used to address a standing problem in PVA: Certain regions of the view (e.g., a scatter plot) may fill up and stabilize sooner than other regions—yet there is no means to halt the progression locally in the stable regions while keeping it running in the others. If a user decides to halt/terminate the progression, it is always globally for the whole view, in which some regions may still be very vaguely developed while others are already overly dense with no more visible changes occurring. By adding the regression, we can keep the progression running while keeping the overly dense regions "in check" by removing data items from them if needed. This way, view regions that take longer to develop into clear patterns have the opportunity to do so without regions already exhibiting a pattern starting to deteriorate again due to increased overplotting. To indicate the regionally different sampling rates, these rates can, for example, be color-coded in the background.

**Transient Snapshots** capture moments of analytic interest as a visualization progresses and regresses. Which moments to snapshot can be triggered by onsets, offsets, or transitions between progression and regression strategies, when crossing thresholds of interest chosen by the analyst, or at regular or irregular wall clock offsets relative to other triggers. Particularly useful are snapshots that behave as fully interactive copies of the visualization at the captured points of progression. Taken in series, such snapshots can serve to record and present the visual analytic provenance [GS06, RESC16] of progression and regression, show the history of convergence from detail to overview, and support coordinated interaction such as parallel navigation and selection. They can even be subject to progression and regression themselves in a multiple view layout that adds them to one end and removes them from the other end as the "window of transience" catches up to, passes, and leaves behind snapshot trigger points. In ways like this, visualization designs can utilize TVA at the level of entire views and layouts as well as at the level of individual and aggregated data items within views.

## 4. Frontend Analogy: Spraying

Working with any form of PVA can be overwhelming, as in addition to handling the data analysis itself, the user now also has to handle (parameterize, steer, or branch) the process of how the data are incrementally loaded and computed. With TVA now adding yet another progressive process to be managed—the regression—data analysis might become even more daunting to the end user. To overcome this hurdle, we suggest the use of an intuitive analogy that provides a mental model for the progressive processes running "under the hood" and that makes their results foreseeable and the effect of adjusting them predictable.

When talking to nontechnical visualization users about the mechanism of transient visualization, we found the analogy of "spraying data items onto a canvas" quite fitting and understandable. In this analogy, the progression is described as spraying data onto the visualization in the same way as a garden hose sprays water onto a clear vertical surface. The surface becomes more visible where the spray lands on it. The more drops that land at a spot, the more they accumulate there. When enough drops have accumulated, the excess water runs down the surface, so that again only some drops remain. If the garden hose is directed somewhere else, over time the sprayed region dries up. After a while only the clear surface remains. This analogy to TVA is illustrated in Figure 2.

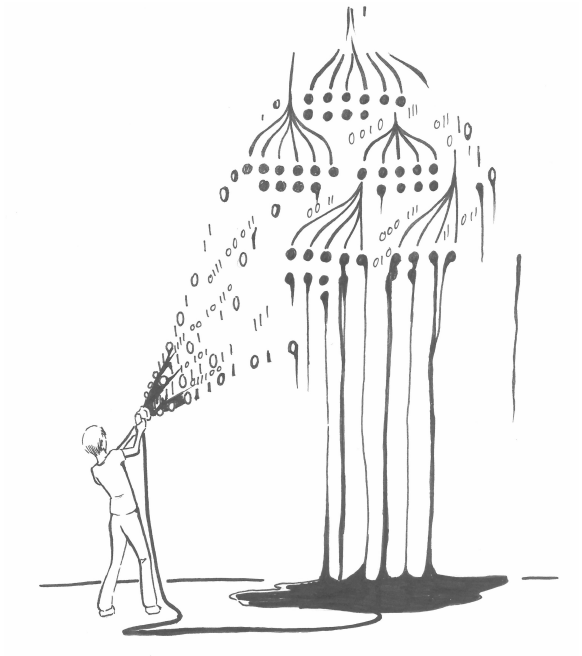

**Figure 2:** *In the analogy of "spraying data onto the visualization", the spray from the garden hose signifies the progression of adding data items to the view, while the runoff stands for the regression of removing excess data from the view.*

The analogy immediately makes clear what to expect from the progressive data stream and how it will behave. Opening the valve (i.e., increasing the chunk size or throughput of the progression) will produce and add more data items/water droplets over time. Setting the nozzle of the garden hose to *mist* will produce a uniform random addition of more data items/droplets at a wide range – possibly to the entire visualization at once. However, setting it to *jet* will produce a very targeted stream that adds data/water to a small region of the visualization; for example, in an attempt to finish the progression in that area sooner. Thanks to the regression, excess water/data runs off, so that the small area cannot drown in water/data and get completely overplotted. A practical use case of this "jet" setting would be to point it at a suspected outlier to increase the progressive sampling around that outlier and then observe if it remains a singleton or not—that is, to test if it is a true outlier or whether it is just an artifact of the progression not having progressed far enough.

Along the lines of *spray rendering* [PS93], which has been used in early incremental visualizations [PA94], it is likewise quite intuitive to consider the utility for TVA of "additives" in the water stream, much like the *smart particles* used in spray rendering. These could be used, for example, to add derived data particles to represent cluster centroids, imputed data, or extrapolated/forecasted data that do not exist in the dataset itself.

Depending on the setting of the described scenario, other mechanisms can be emulated. For example, the sprayed liquid may not be described as water but as a sticky glue-like substance that does not run off the surface but accumulates instead. This plausibly captures an effect like *Visual Sedimentation* [HVF13]. Likewise, the clear surface that is being sprayed may not be described as vertical but as horizontal, so that the water does not run off but pools around spots that receive more data items/water than others. This could serve as a sound model for something akin to *jitter* (e.g., [Cle93, TGC03]) that randomly displaces densely plotted data items to nearby locations, or *Scatterplots with circular pixel placement* [JHM*13] in which data items are not overplotted but instead displaced outward to the next available free pixel, effectively creating something like a "splotch" or "puddle" around dense local areas.

## 5. Backend Analogy: Caching

PVA/TVA is not only a challenge for the end user to manage the additional layer of progression while performing an analysis of the incoming stream of data chunks. It is also a challenge for the engineer who has to develop the TVA system. Mapping the concept of TVA—what it is and what it does—to a well-known existing mechanism can again help to lower the hurdles of realizing TVA.

In our own ongoing work to build a software framework for TVA, we found it helpful to think of it in analogy to a "read-through caching mechanism". In this analogy, the progression would *insert* data items into the cache (i.e., the view) when these are needed, e.g., when they should become visible as the user may have panned the view and they are no longer off-screen. The regression would then be the *eviction mechanism* that removes data items from the cache/view to make space for new data items.

In this analogy, we can even have multiple cache levels – for example, a larger cache that represents the data structures in memory combined with a smaller cache that contains only those data items from the larger cache that are currently visible. Layering the caches like this is useful for making view navigation smoother, as not every little pan and zoom operation will require fetching new data from the database, but can be fulfilled with data still available in the next-level cache. At the same time, when the trajectory of a panning operation is known, data lying in that direction can be prefetched from the database [AW12].

Note that a layered architecture like this is almost a natural extension of existing non-progressive layers in the visualization pipeline. Whether it is layers of filters as introduced by Tominski et al. [TAS09, Sec. 5.3], or layers of zoom levels as used in *Stacked Zooming* [JE13], it is already commonplace in visualizations to utilize layering in which subsequent layers contain fewer data items that can thus be displayed at higher levels of detail. Extending these layers with a "cache-like" functionality that retains and removes data according to different strategies builds onto an existing conceptual understanding of the VA system, instead of asking the developer to rethink and overwrite their current "mental map".

## 6. Combining PVA and TVA

Having criticized that PVA only supports analytic tasks on data aggregates, TVA seems to go from one extreme to the other. What is really needed is a well-balanced combination of the two mechanisms: running PVA in areas of low interest to create crude aggregates that support overview tasks, provide orientation, and aid navigation; while at the same time running TVA in areas of high interest to create detailed per-data item mappings that support direct and indirect look-ups, comparisons, and details on demand.

Figure 3 shows an example of such a combination as generated by an early prototype of our TVA software framework that we are currently developing. The shown progressive visualization is geared to support finding outliers, i.e., one of the tasks we mentioned in the introduction for which PVA is not well suited. Sparse regions are handled by TVA, with additional data items being added individually. The moment a given density threshold is surpassed in a particular region, all data items are removed, and the region is handed over to a PVA process that shows aggregate counts through color coding. This makes sense, as denser regions do not contain outliers and thus regions become increasingly uninteresting as their density increases.

The effect of this combination is that the cluster boundaries, where the data density is low, are processed by TVA at the highest level of detail, while the center of each cluster is treated by PVA and shown in compacted form, effectively freeing memory to keep the regions of interest at a per-item resolution. As the view builds up, empty regions become sparsely populated, sparse regions become densely populated, the cluster boundaries move outward, and the TVA regions follow—i.e., they shift with the analysis interest as needed.

But it is not only the computational and memory efficiency of this combined approach that is notable. The upcoming roadmap and research agenda for Progressive Data Analysis explicitly calls for "mechanisms for attention management" as an important research

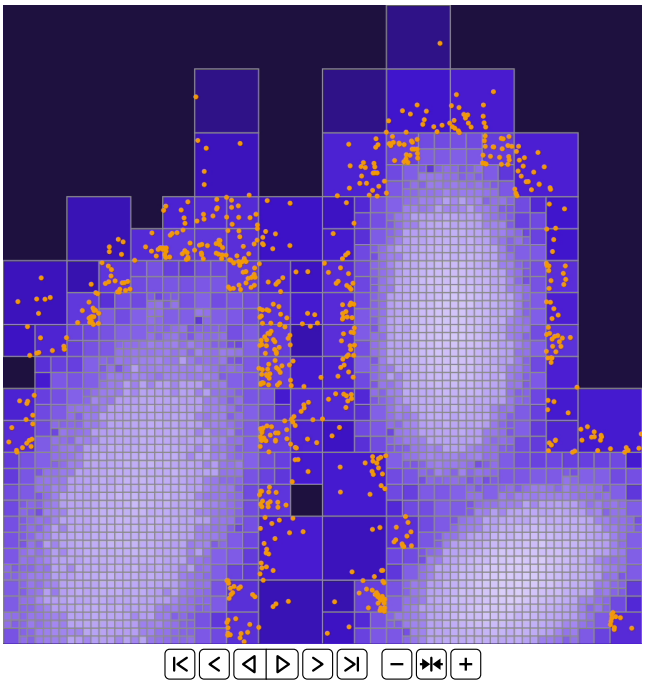

**Figure 3:** *A preview of a combination of PVA and TVA. This example uses PVA to aggregate data in areas of lesser interest (here in the cluster centers) and TVA to placing individual data items in areas of high interest (here at the cluster boundaries). This allows searching for outliers in the sparse regions of the plot even as these regions shift over time with additional data coming in. (To play the animation, use a stand-alone PDF viewer like Adobe Acrobat.)*

challenge [FFS24, chapter 5.5]. We believe that the combination of PVA and TVA, as illustrated in Figure 3, is a viable candidate for providing such a mechanism, as it directs the analyst's attention to the visually salient parts that are (currently) controlled by TVA, while the PVA-controlled parts provide the necessary context.

## 7. Conclusion

For now, transient VA is a concept proposed to alleviate some of the inherent challenges of incremental PVA. As outlined in the previous section, we do not see TVA as a "competitor" to PVA, but as a complementary mechanism that can handle some analysis scenarios that are not a good fit for PVA, and vice versa. While TVA still needs to be further developed—both in theory and practice—we believe it to be the combination of PVA and TVA that will provide a next-generation visual analytics approach for very large data.

## Acknowledgments

The authors are grateful to Remco Chang and Marius Hogräfer who gave valuable input to the idea of Transient Visual Analytics along the way. Special thanks go to Stephan Ohl for providing screenshots from our early TVA software framework for the animation in Figure 3, as well as to Maja Dybboe for her artistic rendition of TVA in Figure 2. Work on this project was funded in part by the Independent Research Fund Denmark (DFF) through project *ArtiPlex*, grant no. 3105-00117B.

## References

[ASSS18] Angelini M., Santucci G., Schumann H., Schulz H.-J.: A review and characterization of progressive visual analytics. *Informatics 5*, 3 (2018), 31:1–31:27. doi:10.3390/informatics5030031. 1

[AW12] Ahmed Z., Weaver C.: An adaptive parameter space-filling algorithm for highly interactive cluster exploration. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)* (Seattle, WA, October 2012), Ward M., Santucci G., (Eds.), pp. 13–22. doi:10.1109/VAST.2012.6400493. 4

[BEF17] Badam S. K., Elmqvist N., Fekete J.-D.: Steering the craft: UI elements and visualizations for supporting progressive visual analytics. *Computer Graphics Forum 36*, 3 (2017), 491–502. doi:10.1111/cgf.13205. 1

[BS04] Bertini E., Santucci G.: Quality metrics for 2D scatterplot graphics: Automatically reducing visual clutter. In *Proceedings of the International Symposium on Smart Graphics* (2004), Springer, pp. 77–89. doi:10.1007/978-3-540-24678-7_8. 2

[CKBE19] Cui Z., Kancherla J., Bravo H. C., Elmqvist N.: Sherpa: Leveraging user attention for computational steering in visual analytics. In *Proceedings of the Symposium on Visualization in Data Science* (2019), IEEE, pp. 48–57. doi:10.1109/VDS48975.2019.8973384. 1, 2

[Cle93] Cleveland W. S.: *Visualizing Data*. Hobart Press, Summit, NJ, 1993. 4

[ED06] Ellis G., Dix A.: The plot, the clutter, the sampling and its lens: occlusion measures for automatic clutter reduction. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (2006), ACM, pp. 266–269. doi:10.1145/1133265.1133318. 2

[EF10] Elmqvist N., Fekete J.-D.: Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics 16*, 3 (2010), 439–454. doi:10.1109/TVCG.2009.84. 2

[FFNE18] Fekete J.-D., Fisher D., Nandi A., (Eds.) M. S.: Progressive data analysis and visualization. *Dagstuhl Reports 8*, 10 (2018), 1–40. doi:10.4230/DagRep.8.10.1. 1

[FFS24] Fekete J.-D., Fisher D., Sedlmair M. (Eds.): *Progressive Data Analysis: Roadmap and Research Agenda*. Eurographics, 2024. to appear. 5

[FPDs12] Fisher D., Popov I., Drucker S., schraefel m.: Trust me, I'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems* (2012), ACM, pp. 1673–1682. doi:10.1145/2207676.2208294. 1

[GS06] Groth D. P., Streefkerk K.: Provenance and annotation for visual exploration systems. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (Nov. 2006), 1500–1510. doi:10.1109/TVCG.2006.101. 3

[HASS22] Hogräfer M., Angelini M., Santucci G., Schulz H.-J.: Steering-by-example for progressive visual analytics. *ACM Transactions on Intelligent Systems and Technology 13*, 6 (2022), 96:1–96:26. doi:10.1145/3531229. 2

[HMPS23] Hogräfer M., Moritz D., Perer A., Schulz H.-J.: Combining degree of interest functions and progressive visualization. In *Short Paper Proceedings of the IEEE Conference on Visualization & Visual Analytics* (2023), IEEE, pp. 251–255. doi:10.1109/VIS54172.2023.00059. 2

[HS24] Hogräfer M., Schulz H.-J.: Tailorable sampling for progressive visual analytics. *IEEE Transactions on Visualization and Computer Graphics* (2024), 1–13. to appear. doi:10.1109/TVCG.2023.3278084. 2

[HVF13] HURON S., VUILLEMOT R., FEKETE J.-D.: Visual sedimentation. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2446–2455. doi:10.1109/TVCG.2013.227. 4

[JE13] JAVED W., ELMQVIST N.: Stack zooming for multifocus interaction in skewed-aspect visual spaces. *IEEE Transactions on Visualization and Computer Graphics 19*, 8 (2013), 1362–1374. doi:10.1109/TVCG.2012.323. 4

[JHM*13] JANETZKO H., HAO M. C., MITTELSTÄDT S., DAYAL U., KEIM D.: Enhancing scatter plots using ellipsoid pixel placement and shading. In *Proceedings of the 46th Hawaii International Conference on System Sciences* (2013), IEEE, pp. 1522–1531. doi:10.1109/HICSS.2013.197. 4

[PA94] PANG A., ALPER N.: Mix&match: a construction kit for visualization. In *Proceedings of Visualization* (1994), IEEE, pp. 302–309. doi:10.1109/VISUAL.1994.346305. 4

[PRJ*23] PATIL A., RICHER G., JERMAINE C., MORITZ D., FEKETE J.-D.: Studying early decision making with progressive bar charts. *IEEE Transactions on Visualization and Computer Graphics 29*, 1 (2023), 407–417. doi:10.1109/TVCG.2022.3209426. 1

[PS93] PANG A., SMITH K.: Spray rendering: Visualization using smart particles. In *Proceedings of Visualization* (1993), IEEE, pp. 283–290. doi:10.1109/VISUAL.1993.398880. 4

[RESC16] RAGAN E. D., ENDERT A., SANYAL J., CHEN J.: Characterizing provenance in visualization and data analysis: An organizational framework of provenance types and purposes. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (January 2016), 31–40. doi:10.1109/TVCG.2015.2467551. 3

[RH09] ROSENBAUM R., HAMANN B.: Progressive presentation of large hierarchies using Treemaps. In *Proceedings of the International Symposium on Visual Computing* (2009), Springer, pp. 71–80. doi:10.1007/978-3-642-10520-3_7. 3

[TAS09] TOMINSKI C., ABELLO J., SCHUMANN H.: CGV—an interactive graph visualization system. *Computers & Graphics 33*, 6 (2009), 660–678. doi:10.1016/j.cag.2009.06.002. 4

[TGC03] TRUTSCHL M., GRINSTEIN G., CVEK U.: Intelligently resolving point occlusion. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)* (Seattle, WA, October 2003), IEEE, pp. 131–136. doi:10.1109/INFVIS.2003.1249018. 4

[UAF*23] ULMER A., ANGELINI M., FEKETE J.-D., KOHLHAMMER J., MAY T.: A survey on progressive visualization. *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–18. to appear. doi:10.1109/TVCG.2023.3346641. 1

[WM04] WILLIAMS M., MUNZNER T.: Steerable, progressive multidimensional scaling. In *Proceedings of the IEEE Symposium on Information Visualization* (2004), IEEE, pp. 57–64. doi:10.1109/INFVIS.2004.60. 2

[ZCPB11] ZHAO J., CHEVALIER F., PIETRIGA E., BALAKRISHNAN R.: Exploratory analysis of time-series with ChronoLenses. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2422–2431. doi:10.1109/TVCG.2011.195. 2