Drawing *in* the Flow: A Data-Aware Mixed-Reality Sketching Interface for Illustrative 3D Flow Visualization

Walter Sands* Sean J Dorr† Kiet Tran‡ Daniel F. Keefe§

Department of Computer Science & Engineering
University of Minnesota - Twin Cities

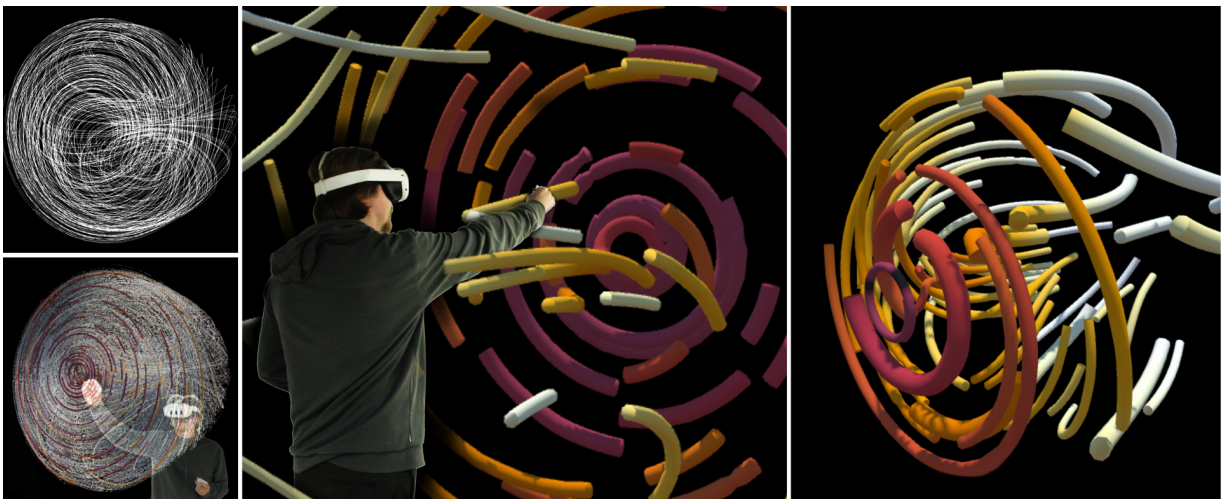

Figure 1: *Upper-Left:* 3D spatial visualization tools, like Paraview [3], work well for calculating and visualizing streamlines, but it is challenging to refine these visualizations in ways that a skilled illustrator would (e.g., by emphasizing specific lines or portions of lines). *Bottom-Left:* The Drawing *in* the Flow starts by importing a dense 3D line set (e.g., generated in Paraview with streamlines seeded on a regular 3D grid or randomly). These define an underlying 3D data "canvas", which can optionally be visualized to provide context. *Middle:* Users then work with a bimanual 3D sketching interface to draw lines that (roughly) follow the flow, controlling the length, density, color, and other stylistic properties as with any digital painting tool. "Ink-data settling" morphs these hand-drawn lines to match the underlying vector flow data, and "lazy data binding" can be used to tie color and line width to underlying scalar data (e.g., speed, pressure). *Right:* The result is an accurate semi-immersive 3D multivariate flow visualization customized without programming or scripting to emphasize particular features for teaching or other illustrative purposes.

Abstract

We present "Drawing *in* the Flow", a mixed-reality 3D user interface for authoring illustrative, multivariate 3D flow visualizations by sketching on, or better stated, *in*, 3D data. The approach interprets hand-drawn 3D strokes relative to an underlying data "canvas" and applies animated "ink-data settling" to ensure the strokes accurately reflect vector field data (i.e., 3D streamline paths). Color and other visual properties are interpreted relative to scalar data variables with "lazy data binding" to help users prioritize creative visual design tasks. Results include example 3D illustrations of multiple flow fields. The work is significant because of the ability to make authoring accurate 3D mixed reality data visualizations accessible to stakeholders without programming or scripting experience and because it demonstrates a novel approach to balancing the tradeoff between accuracy and expression in 3D data visualization.

Index Terms: Mixed Reality, Sketching, Flow Visualization.

*e-mail: sands224@umn.edu
†e-mail: dorr0024@umn.edu
‡e-mail: tran0563@umn.edu
§e-mail: dfk@umn.com

1 Introduction

While visualization techniques for analyzing and explaining complex spatial phenomena (e.g., fluid flows) have made great strides, the tools needed to create these 3D visualizations, which are often viewed in virtual or mixed reality (MR) displays, remain complex and, generally, not accessible to stakeholders who do not have advanced training in programming or data processing. This contributes to two challenges that are longstanding in the field. First, it limits contributions to visualization from visual artists, designers, illustrators and others who, if they had the right tools, might be able to apply their deep visual training and creativity to improve these hard-to-design spatial visualizations. Second, the high effort required to create a visualization means that we gravitate toward creating visualization techniques that work in a wide variety of general cases rather than custom domain-and-task-specific visualizations. Of course visualizations that work well in many cases are important, valuable, and a great starting point, but they are not necessarily the best end point. A medical illustrator will fine-tune their drawing to emphasize some features and de-emphasize others; a thermodynamics professor sketching a fluid flow on a whiteboard will selectively emphasize the flow features that relate to the current lesson; a climate scientist drawing a picture of changing patterns in ocean currents will always desire an accurate picture, but they will simultaneously adjust that picture to best suit the audience (e.g., elementary school students, scientific colleagues, policy makers).

Our work is, therefore, motivated by the goal of giving all stakeholders who work with 3D spatial visualizations the power to create and customize these visualizations with an interface that is as natural as sketching. We are not the first researchers to explore this direction. Prior work has found that sketching interfaces can be a valuable way to design custom data visualizations for both 2D non-spatial data [13, 16] and, as in our case, 3D spatial data [11]. However, a number of open challenges remain.

This short paper contributes to, perhaps, the most important theme and research question in sketch-based visualization authoring: *How to balance the inherent tradeoff between accuracy and expressiveness?* From prior research, we know the answer depends upon the style of visualization. The TimeSplines technique [15], for example, does an exceptional job of finding this balance for 2D personalized, curved timeline visualizations–within this style, the tool is exceptionally expressive. Similarly, Visualization-by-Sketching [21] finds the right balance between accuracy and expressiveness for multi-layered 2D scalar and vector fields visualized using hand-drawn animated glyphs and hand-painted color maps.

Our contribution is specific to working with 3D spatial data visualized with a perspective-tracked stereoscopic display. Although there is a long, successful history of 3D sketching and painting in virtual reality [9], we do not yet understand the user-interaction metaphors and data binding techniques needed to find a similar "right balance" between accuracy and expressiveness in this context. To this end, as shown in Fig. 1, we introduce a bimanual user interface for 3D sketching while *inside* 3D fluid flow data, and we demonstrate how to interpret "ink-data settling" [20] and "lazy data binding" [13], two important concepts from prior 2D sketch-based visualization authoring tools, to the 3D sketching and 3D spatial data context.

2 Related Work

2.1 Authoring Visualizations without Programming

A variety of visualization tools have been developed to enable users to construct charts and visualizations without programming. Work in this direction relies on 2D graphical user interfaces to facilitate dragging and dropping data fields [23], suggestion-directed visualization [26, 27], and constraint-based and direct manipulation [17, 19]. All of these tools have the common objective of reducing the technical barrier to expand the set of users involved in creating data visualizations. Continuing with this objective, AI-powered authoring tools have been used to automate data transformation and chart generation [25], as well as to alter charts via user prompts [22]. Across these approaches, there are varying degrees of expressivity, learnability, and effectiveness, leading toward more powerful yet accessible visualization authoring solutions for non-programmers. Although generally effective in reducing technical barriers to creating data visualizations, there is potential to do even more in this style, for example, moving beyond the keyboard and mouse, to create styles of user interaction that better match how traditionally trained visual designers work.

2.2 Sketching for Non-Spatial Data Visualization

Sketching on paper has always been an important way to quickly explore an idea and iteratively refine it [24], and researchers have long sought to translate these powerful properties of sketching into computer-based tools. For non-spatial data visualized using 2D vector-based graphics, Data Illustrator [13] focuses on flexible data-to-design mappings that enable users to embed data into lines, shapes and their anchor points. Data Illustrator also introduces the concept of "lazy data binding" for non-spatial data, which has influenced our work and other follow-on systems, like TimeSplines [15] and DataGarden [16]. DataGarden specifically brings in the ability to attach the data to sketched glyphs allowing for more expression in the shapes that represent the data.

2.3 Sketching for Spatial 2D Data Visualization

Our approach is geared toward visualizing spatial data, and the prior work here is also informative. Drawing with the Flow [20] introduced a sketch-based interface for visualizing 2D vector fields by implementing a concept called "Ink-data settling", where strokes gently morph to match the underlying vector field data after they are drawn. Our work builds on this by moving into the mixed reality space with 3D vector and scalar data. A closely related follow-up is Visualization-by-Sketching [21], which extends the gestural sketching method to animated mulitvariate geospatial data with colormap adjustments and animated streaklets. From the user's standpoint, the way that Visualization-by-Sketching infers color maps from the digital paint applied by artists feels much like the "lazy data binding" concept because the user's first actions and focus are on creating the visual, and the details of the data mapping come later. These works have directly informed our new 3D sketching interface.

2.4 Sketching for Spatial 3D Data Visualization

Sketching has been extensively studied as a 3D data selection technique (e.g., [1, 5]). Sketching has also been used to more intuitively design volume rendering transfer functions and manipulate volume data (e.g., [6]). However, these systems employed 2D sketching input on a tablet or monitor displaying 3D graphics. Our specific interest is in exploring the potential of 3D sketching in perspective-tracked, stereoscopic virtual and mixed reality since this is an increasingly common medium for data visualization.

3D sketching is "a type of technology-enabled sketching where: (1) the physical act of mark making is accomplished off-the-page in a 3D, body-centric space, (2) a computer-based tracking system records the spatial movement of the drawing implement, and (3) the resulting sketch is often displayed in this same 3D space, for example, via the use of immersive computer displays, as in virtual and augmented realities (VR and AR) [2]." 3D sketching has been used previously to prototype hard-to-design virtual reality visualizations via a process called "Scientific Sketching" [11]. The design process has been used for many years, most recently to prototype visualizations of physics simulations for paleontologists [14]. Its benefits are that it is accessible to visual artists and designers and that the prototypes can be critiqued in the target medium of VR. However, it has the major shortcoming that the sketches produced are not bound to data (i.e., they are designs only, not data-driven visualizations).

On the other extreme, Artifact-Based Rendering [10] introduced using physical materials from nature and traditional artistic media (sketching, drawing, painting, sculpting). All of the visual elements are hand-crafted by artists and then digitally scanned and finally bound to data using a puzzle-piece-style visual programming interface [7]. The expressiveness of the resulting visualizations is outstanding, even motivating research on affect in visualization [18, 28], but the approach does not have the immediacy and intuition of working with sketch-based interfaces directly in a visualization environment. Our work seeks to fill this gap, introducing a 3D sketching interface for designing 3D spatial data visualizations directly within the target medium.

3 Drawing *in* the Flow

This section introduces the data processing, user interface, and algorithms for our conception of Drawing *in* the Flow.

3.1 Data Pre-Processing

We use Paraview [3], an open-source scientific visualization application, to load and process multi-field volumetric data and our own custom application written with the Unity Game Engine to implement the interface and data visualization. The system runs on the Meta Quest 3 platform. For larger datasets (over 900 streamlines), we tether the Quest to a MSI Stealth GS77 laptop with NVidia Geforce RTX 3060 GPU and 12th Gen Intel Core i7-12700H CPU.

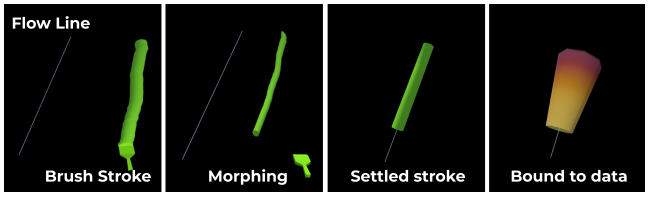

Figure 2: User drawn stroke morphing from where it was drawn to a flow line then data is bound to the stroke.

The pre-processing in Paraview consists of generating a dense streamline set for the data using the Stream Tracers filter with an adaptive Runge-Kutta 4-5 order integrator. We export the resulting line geometries to a simple data file format that contains a series of (x, y, z) points with an integration time value and, optionally, co-located values for multiple scalar data variables at these locations. This data file is then read into Unity, where we render smooth lines from the sampled positions as extruded splines.

3.2 Bimanual User Interface

Users then enter the immersive environment, which can run in either a fully-immersive VR mode or a see-through MR mode, to explore the flow data. Upon loading, the flow lines are visualized as thin white lines that are intended to function as a virtual "underpainting layer" to provide context. The visibility of this layer is toggled with the primary button (A) on the right-hand controller.

Users walk around to view the visualization, including going inside the flow field. By doing so, they can view any part of the flow field and access specific flow lines. In addition, holding the left controller's trigger button will "grab" the artwork and rotate or translate it to view it from other angles using input from the controller's six degree-of-freedom tracking. While grabbing the artwork with the left hand, holding the right controller's trigger button activates a bimanual scaling mode, where the artwork's scale is adjusted dynamically as the controllers are moved nearer or further from each other, scaling down or up, respectively. The goal of these grabbing features is to make it possible to quickly view and/or access with their hands any location in the 3D flow field. The color of the brush is adjusted by holding the primary button on the left controller (X), which activates a double-sided cone 3D Hue, Saturation and Value (HSV) color picker widget modeled on the one in CavePainting [12]. The brush size can be adjusted in a similar method to scaling the artwork. If the two controllers are held close together (within a set proximity of each other) pressing the left controller's trigger will instead scale the brush in proportion to the distance between the hands. The entire bimanual interface is implemented following the strategy described in VR Developer Gems Chapter 14 [7]. Specifically, we implemented a finite-state machine that changes state in response to various input events, and this makes it possible to accomplish features like overloading the interpretation of the trigger buttons on the controllers, which helps the interface feel immediate, gestural, and fluid–similar to sketching [7].

The most important user interface action is to sketch flow lines. Again, this is accomplished by holding the right-hand controller's trigger button. As seen in the left panel of Fig. 2, the user's stroke is visualized using a tube-shaped triangle mesh that is generated dynamically, adding one new segment to the tube each graphics frame, to follow the trajectory (position and orientation) of the user's hand movements until the trigger is released.

3.3 Drawing within Data Fields

Our key innovation is in interpreting the strokes drawn with the interface relative to the underlying 3D data field, and we have several modes and techniques for accomplishing this. Our current implementation supports binding data to the position, color, and size visual channels of the hand-drawn strokes. The line data originate from a vector field in the original dataset, and our approach is to always bind the position of the users' strokes to these data using an "ink-data settling" technique that gently morphs each stroke after it is drawn to match the underlying data (details described in the next section). The color and size channels (size is the radius of the tube geometry) are bound to scalar fields in the underlying data. The specific data variable to map to each channel does not need to be defined before drawing so users can focus entirely on visual experimentation if they wish (i.e., lazy data binding). These bindings can be set, changed, or reset to the originally drawn size and color at any point during the working session using a menu virtually "carried" on the non-dominant hand like a palette. In all cases, whenever a mapping is changed, the drawing responds by updating with a smooth animated transition. The palette menu in the non-dominant hand includes controls for these mappings as well as an annotation mode where the user's strokes are not interpreting relative to the data. Finally, the palette menu also includes options to load preset color maps; however, another way to define the color and size data-mappings is to simply sketch lines of the desired colors and widths and ask the system to "infer" the mapping (details described in Sec. 3.5).

3.4 Ink-Data Settling

With ink-data settling, each stroke is interpreted as a depiction of the underlying flow field, and to make sure these depictions are accurate, after each stroke is drawn, it is smoothly morphed to align with the most similar streamline in the dense, pre-computed streamline set. To compute the similarity of the drawn lines to the precomputed streamlines, we use a metric that computes a weighted sum of differences in distance and direction. For each point along the drawn stroke (p_1), we find the nearest point for each streamline (p_2). Then, we lookup the direction (tangent) of the two lines at these points ($\vec{d}_1$ and $\vec{d}_2$). The similarity metric is then calculated using the following equation with weights $w_{dist} = 10$ and $w_{dir} = 1$ in our implementation.

$$S = \sum_{i=1}^{n} w_{dist} * distance(p_1, p_2)^2 + w_{dir} * |\vec{d}_1 \cdot \vec{d}_2| \qquad (1)$$

Computing this metric on a large dataset of streamlines can be computationally expensive and prevent the fast, smooth framerates required for head-tracked displays. We have implemented two strategies to overcome this. First, the metric described above is computed incrementally as each stroke is drawn. Second, we use a simple spatial data structure to reduce the number of lines to check. During program initialization, we divide the volumetric data space into a regular 3D grid cubes and, for each cube, create a list of the id numbers of each flowline spline that passes through the cube. This increases performance because the similarity metric only needs to be computed for the flow lines that pass through the same cubes as the user's drawn stroke. Once the most similar flow line is determined by minimizing similarity calculation, a segment of that flow line is found based, again, on the similarity but this time adding the absolute difference in line length with the segment being compared as a constraint. The calculation becomes:

$$S = \sum_{i=1}^{n} (w_{dist} * distance(p_1, p_2)^2 + w_{dir} * |\vec{d}_1 \cdot \vec{d}_2|) + w_{len} * |l_1 - l_2| \qquad (2)$$

We want only a segment because some flow lines can be long and only a fraction of the line is necessary to give the impression of the entire line.

Finally, as shown in Fig. 2, a smooth morph to the line's new location is accomplished with linear interpolation on the positions,

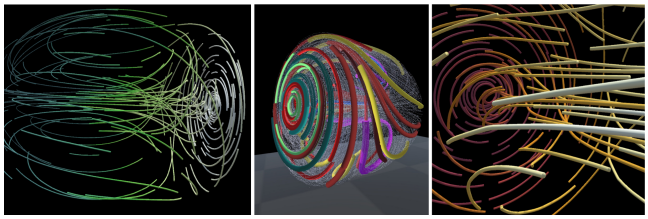

Figure 3: Additional 3D illustrations of the heated and spinning disk dataset show the range of visual styles that can be created using the 3D sketching interface.

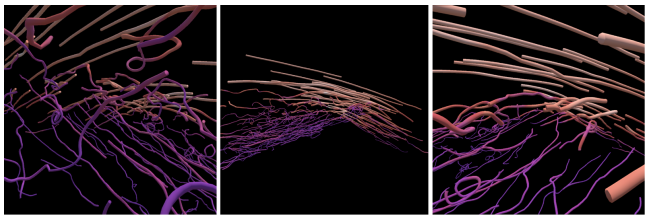

Figure 4: Perspective views from walking around a hand-drawn 3D illustration of a forest fire simulation, with color mapped to velocity magnitude (tan=low, magenta=high).

orientations, (optionally) colors and (optionally) widths of each segment of the stroke. This animation occurs over three seconds.

3.5 Inferring Scalar Data Bindings

Inspired by the 2D Visualization-by-Sketching system [21], color maps and size scales to apply consistently across the 3D dataset can be inferred directly from the visual properties of the hand-drawn strokes. Since the size channel assumes a linear mapping from a minimum radius to a maximum radius, we use a simple heuristic to infer this mapping. First, we search all of the drawn strokes to determine the minimum and maximum sizes for the lines in the user's current illustration style. Then, we examine the scalar data value for whatever data variable is currently bound to size at the 3D locations where these minimum and maximum sized strokes were drawn to determine the direction of the mapping (i.e., should size increase when the data values increase or the reverse). Finally, the inferred mapping is applied consistently to all strokes using, as always, a smooth animated transition.

The strategy for inferring color is only slightly more complex. The algorithm identifies all the unique colors drawn by the user. Then, it calculates the mean value for whatever data variable is currently bound to color across all of the locations of these strokes, averaging across all of the sample points that make up a stroke and multiple strokes (if there are multiple strokes that use the same color). Finally, it creates a new color map with one control point for each unique color used in the illustration assigned to a the mean data value calculated for that color. For perceptual accuracy, all color interpolation is calculated in the Lab color space.

4 Results

Thus far, we have applied "Drawing *in* the Flow" to interactively sketch 3D illustrations of two different representative datasets. Fig. 1 and Fig. 3 show illustrations of the Paraview example dataset disk_out_ref.ex2. In pre-processing, we used a point cloud seeding strategy to generate a streamline set with 900 streamlines with an average of 50 data points per line. Each data point includes the point's (x, y, z) position as well as values for the following data variables at that location: AsH3, CH4, GaMe3, H2, Pressure, Temperature, Velocity in x, y, z (converted to Velocity Magnitude), Vorticity in x, y, z (converted to Vorticity Magnitude), Rotation, Angular Velocity, and Normal in x,y,z.

For this dataset, the spatial grid search reduced the computational time when checking similarity of a stroke to the flow data from 0.2 to 0.01 seconds. This is sufficient to support running the calculations in real-time while the stroke is being drawn. The illustrations in Fig. 3, created by one of the paper co-authors, provide some documentation of the variety of styles that can be produced on the same dataset. Each variation took about 30 minutes to make from an initial blank visualization.

Fig. 4 shows an illustrative visualization for timestep 46000 of the "headcurve 40" simulation from the of the FireTec ensemble dataset [4] provided by Los Alamos National Laboratory in support of the IEEE VIS 2022 SciVis Contest. In pre-processing, we generated a streamline set for these data that contains 800 streamlines with an average of 400 samples per line. This illustration, created by a second one of the paper co-authors, took approximately 45 minutes to design and create.

5 Conclusion, Limitations, and Future Work

Our impression from using the system is that it has potential to be an engaging and effective tool for visualizing 3D flow data. However, this is just an early impression, and we plan to conduct a user study to better understand the strengths and weaknesses of the approach. Since the primary goals of the tool are to enable rapid experimentation with a wide variety of 3D line illustration styles, we believe a reproduction study will be informative. Although they are not common across all areas of visualization research, reproduction studies have been used with success recently to understand the expressive capabilities of a variety of visualization authoring systems [21, 15, 16]. In this type of study, we would plan, for example, to ask participants to recreate successful example visualizations from prior work, such as the IEEE VIS 2022 SciVis Contest from which we have already used data (e.g. Fig. 4) [4].

One limitation, and the most obvious avenue for future work, is to expand the richness of the 3D drawing and painting engine used in our initial implementation. This should include supporting multiple brush types and textures, as well as using brush pressure sensors to add tapering to strokes. A gesture-based interface similar to the 2D one in Drawing with the Flow could be another valuable future addition, making it possible to crop, extend, delete, and edit visualization marks via pen-based gestures that fit naturally into a sketching process.

Although our early testing has not suggested a direct need, we can imagine situations where additional tools might be useful to help users work in extremely dense regions of the flow data, where occlusion might limit visibility. One potential extension to help with this would be a magic lens that hides flow lines that cross between the user's head and their pen, e.g., similar to the approach in Force Brushes [8], but extended to MR. Another alternative, potentially complementary, design would hide all lines except for the ones that pass through a sphere surrounding the user's pen, similar to the local flow preview in Drawing with the Flow [20].

Finally, we plan to add support for creating animated flow visualizations, first for steady flows, where the work will focus on the user interface and metaphors for defining some objects as animated and others as static. Then, we also plan to support time-varying flow data, where the work will need to explore what it means to sketch on/in a 3D data canvas that changes over time, including extending the underlying data structures to accommodate time and pathlines as opposed to streamlines.

Acknowledgments

The work was conducted in the context of a larger effort to advance the technologies and teaching of 3D Drawing in Extended

Reality with co-investigator Dr. Bret Jackson of Macalester College, supported in part by the National Science Foundation (Awards #2326998 and #2326999). The authors also thank members of the Spring 2025 CSCI-8980 3D Drawing in eXtended Reality course, where this work originated as a final course project, and the University of Minnesota ArTeS Collaborative, which supported the course.

REFERENCES

[1] D. Akers. Cinch: a cooperatively designed marking interface for 3d pathway selection. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, p. 33–42. Association for Computing Machinery, New York, 2006. doi: 10.1145/1166253.1166260 2

[2] R. Arora, M. Machuca, P. Wacker, D. Keefe, and J. Israel. *Introduction to 3D sketching*, pp. 151–177. River Publishers, Denmark, Dec. 2022. 2

[3] U. Ayachit. *The paraview guide: a parallel visualization application*. Kitware, Inc., 2015. 1, 2

[4] D. Banesh, R. Linn, and J. Patchett. Vorticity-driven lateral spread ensemble data set, 2021. Los Alamos. 4

[5] L. Besançon, M. Sereno, L. Yu, M. Ammi, and T. Isenberg. Hybrid touch/tangible spatial 3D data selection. vol. 38, pp. 553–567. Wiley Online Library, 2019. Issue: 3. 2

[6] S. Bruckner and M. Groller. Volumeshop: an interactive system for direct volume illustration. In *VIS 05. IEEE Visualization, 2005.*, pp. 671–678, 2005. doi: 10.1109/VISUAL.2005.1532856 2

[7] B. Herman, F. Samsel, A. Bares, S. Johnson, G. Abram, and D. F. Keefe. Printmaking, puzzles, and studio closets: Using artistic metaphors to reimagine the user interface for designing immersive visualizations. In *2020 IEEE VIS Arts Program (VISAP)*, pp. 19–28, 2020. doi: 10.1109/VISAP51628.2020.00009 2, 3

[8] B. Jackson, D. Coffey, and D. F. Keefe. Force Brushes: Progressive Data-Driven Haptic Selection and Filtering for Multi-Variate Flow Visualizations. In M. Meyer and T. Weinkaufs, eds., *EuroVis - Short Papers*. The Eurographics Association, 2012. doi: /10.2312/PE/EuroVisShort/EuroVisShort2012/007-011 4

[9] B. Jackson and D. Keefe. Chapter 14: From Painting to Widgets, 6-DOF and Bimanual Input Beyond Pointing. In *VR Developer Gems*, p. 26. AK Peters/CRC Press, Taylor and Francis Group, 1st edition ed., 2019. 2

[10] S. Johnson, F. Samsel, G. Abram, D. Olson, A. J. Solis, B. Herman, P. J. Wolfram, C. Lenglet, and D. F. Keefe. Artifact-based rendering: Harnessing natural and traditional visual media for more expressive and engaging 3d visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):492–502, 2020. doi: 10.1109/TVCG.2019.2934260 2

[11] D. F. Keefe, D. Acevedo, J. Miles, F. Drury, S. M. Swartz, and D. H. Laidlaw. Scientific sketching for collaborative VR visualization design. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):835–847, 2008. doi: 10.1109/TVCG.2008.31 2

[12] D. F. Keefe, D. A. Feliz, T. Moscovich, D. H. Laidlaw, and J. J. LaViola. Cavepainting: a fully immersive 3d artistic medium and interactive experience. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D '01, p. 85–93. Association for Computing Machinery, New York, 2001. doi: 10.1145/364338.364370 3

[13] Z. Liu, J. Thompson, A. Wilson, M. Dontcheva, J. Delorey, S. Grigg, B. Kerr, and J. Stasko. Data illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, p. 1–13. Association for Computing Machinery, New York, 2018. doi: 10.1145/3173574.3173697 2

[14] J. Novotny, J. Tveite, M. L. Turner, S. Gatesy, F. Drury, P. Falkingham, and D. H. Laidlaw. Developing virtual reality visualizations for unsteady flow analysis of dinosaur track formation using scientific sketching. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):2145–2154, 2019. doi: 10.1109/TVCG.2019.2898796 2

[15] A. Offenwanger, M. Brehmer, F. Chevalier, and T. Tsandilas. Timesplines: Sketch-based authoring of flexible and idiosyncratic timelines. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):34–44, 2024. doi: 10.1109/TVCG.2023.3326520 2, 4

[16] A. Offenwanger, T. Tsandilas, and F. Chevalier. Datagarden: Formalizing personal sketches into structured visualization templates. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1268–1278, 2025. doi: 10.1109/TVCG.2024.3456336 2, 4

[17] D. Ren, B. Lee, and M. Brehmer. Charticulator: Interactive construction of bespoke chart layouts. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):789–799, 2019. doi: 10.1109/TVCG.2018.2865158 2

[18] F. Samsel, G. Abram, S. Zeller, and D. Keefe. Affective palettes for scientific visualization: Grounding environmental data in the natural world. In *2021 IEEE VIS Arts Program (VISAP)*, pp. 20–34, 2021. doi: 10.1109/VISAP52981.2021.00009 2

[19] A. Satyanarayan and J. Heer. Lyra: An Interactive Visualization Design Environment. *Computer Graphics Forum (Proc. EuroVis)*, 2014. doi: 10.1111/cgf.12391 2

[20] D. Schroeder, D. Coffey, and D. Keefe. Drawing with the Flow: A Sketch-Based Interface for Illustrative Visualization of 2D Vector Fields. In M. Alexa and E. Y.-L. Do, eds., *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. The Eurographics Association, 2010. doi: /10.2312/SBM/SBM10/049-056 2, 4

[21] D. Schroeder and D. F. Keefe. Visualization-by-sketching: An artist's interface for creating multivariate time-varying data visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):877–885, 2016. doi: 10.1109/TVCG.2015.2467153 2, 4

[22] V. Setlur, S. E. Battersby, M. Tory, R. Gossweiler, and A. X. Chang. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, p. 365–377. Association for Computing Machinery, New York, 2016. doi: 10.1145/2984511.2984588 2

[23] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002. doi: 10.1109/2945.981851 2

[24] T. Strothotte, B. Preim, A. Raab, J. Schumann, and D. R. Forsey. How to Render Frames and Influence People. *Computer Graphics Forum*, 1994. doi: 10.1111/1467-8659.1330455 2

[25] C. Wang, J. Thompson, and B. Lee. Data formulator: Ai-powered concept-driven visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):1128–1138, 2024. doi: 10.1109/TVCG.2023.3326585 2

[26] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):649–658, 2016. doi: 10.1109/TVCG.2015.2467191 2

[27] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, p. 2648–2659. Association for Computing Machinery, New York, 2017. doi: 10.1145/3025453.3025768 2

[28] S. Zeller, F. Samsel, and L. Bartram. Affective, hand-sculpted glyph forms for engaging and expressive scientific visualization. In *2022 IEEE VIS Arts Program (VISAP)*, pp. 127–136, 2022. doi: 10.1109/VISAP57411.2022.00025 2