Visualization-by-Sketching: An Artist's Interface for Creating Multivariate Time-Varying Data Visualizations

David Schroeder, *Member, IEEE*, and Daniel F. Keefe, *Senior Member, IEEE*

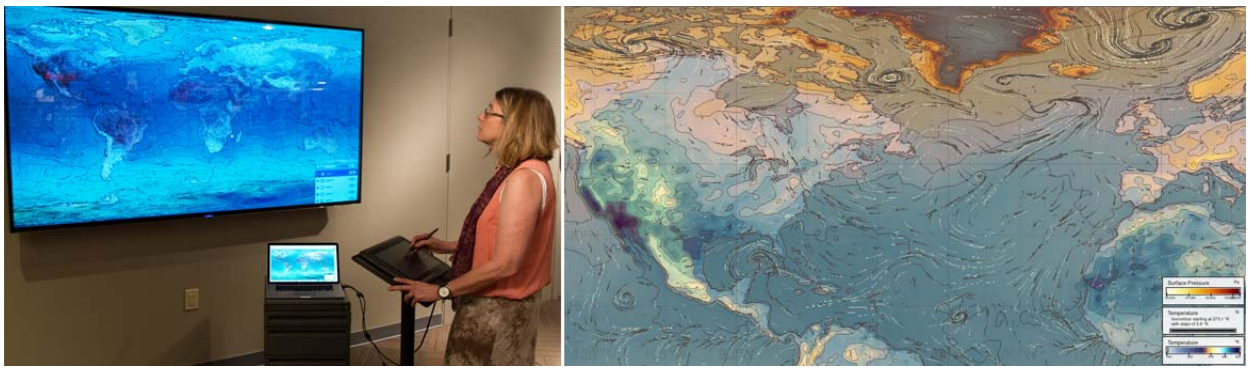

Fig. 1: Left: Visualization-by-Sketching enables artists and other visual experts to create accurate and expressive data visualizations by painting on top of a digital data canvas, sketching data glyphs, and arranging and blending together multiple layers of animated 2D graphics. Right: A variety of expressive visual styles can be created, such as this depiction of wind speed, surface pressure, and temperature using color, iso-contours, contextual graphics (e.g., land masses), and glyphs.

Abstract— We present Visualization-by-Sketching, a direct-manipulation user interface for designing new data visualizations. The goals are twofold: First, make the process of creating real, animated, data-driven visualizations of complex information more accessible to artists, graphic designers, and other visual experts with traditional, non-technical training. Second, support and enhance the role of human creativity in visualization design, enabling visual experimentation and workflows similar to what is possible with traditional artistic media. The approach is to conceive of visualization design as a combination of processes that are already closely linked with visual creativity: sketching, digital painting, image editing, and reacting to exemplars. Rather than studying and tweaking low-level algorithms and their parameters, designers create new visualizations by painting directly on top of a digital data canvas, sketching data glyphs, and arranging and blending together multiple layers of animated 2D graphics. This requires new algorithms and techniques to interpret painterly user input relative to data "under" the canvas, balance artistic freedom with the need to produce accurate data visualizations, and interactively explore large (e.g., terabyte-sized) multivariate datasets. Results demonstrate a variety of multivariate data visualization techniques can be rapidly recreated using the interface. More importantly, results and feedback from artists support the potential for interfaces in this style to attract new, creative users to the challenging task of designing more effective data visualizations and to help these users stay "in the creative zone" as they work.

Index Terms—Visualization design, multivariate, art, sketch, color map, glyph

1 Introduction

Today, in the era of web-based graphics and data-intensive workflows, more people than ever before are paying attention to data visualization, and the data we wish to visualize are more complex than ever. As these trends continue, one of the critical challenges that visualization practitioners will face is that of effective "visual design". By *visual design* we mean a process similar to the type of ideation, exploration, iterative refinement, and critique that artists, illustrators, graphic designers, architects, or other design professionals practice as an essential part of their creative processes in other domains.

For data visualization, the goal of this process is to create a mapping from data to visuals (sometimes called a "digital visual metaphor" [6]) that is accurately readable, insightful, and (often) aesthetically pleasing for a given audience. A simple example of creating this mapping (described from the standpoint of a designer's thoughts) is as follows: *Let me try displaying variable A using a red glyph; how does this look? Good. I see, this red color makes A "pop out" relative to the color map I have applied in the background. But, wait, A is now too visually prominent, and this is interfering with my ability to read variable B, which I have displayed with a more subtle color. Maybe the whole visualization will read better if I change the shape of A's glyph to make it thinner, let me quickly try that...* When we encounter new data types and complex datasets (e.g., containing multiple variables that must be understood together, with data changing over time) designing an effective visual mapping can be the most challenging aspect of creating the visualization.

While a variety of powerful toolkits and applications have been developed to support the technical aspects of creating data visualizations [3, 13, 25], current tools are limited in their ability to support creative visual designers and design processes:

- Many current tools require knowledge of programming, scripting, or the meaning of algorithmic parameters that goes beyond what is reasonable to expect of traditionally trained artists and designers, making these tools inaccessible to an important community of creative visual thinkers.
- Other tools are accessible to a wider range of users, but do not have the needed expressive power; for example, the tool may provide only an ability to tweak pre-canned visualizations.

- *David Schroeder conducted this work at the University of Minnesota and is now with Gonzaga University. E-mail: schroederd@gonzaga.edu.*
- *Daniel F. Keefe is with the Department of Computer Science & Engineering at the University of Minnesota. E-mail: keefe@cs.umn.edu.*

Manuscript received 31 Mar. 2015; accepted 1 Aug. 2015; date of publication 20 Aug. 2015; date of current version 25 Oct. 2015. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org. Digital Object Identifier no. 10.1109/TVCG.2015.2467153

- Finally, compared to results within the growing body of literature on creativity support tools (e.g., [2, 4, 10, 26]) and even existing software for other common visual design tasks (e.g., digital painting, photo editing, 3D modeling), current tools for creating data visualizations rarely guide users toward workflows that incorporate sketching, exemplars, analogy, and/or other activities that we know are correlated with creative processes and results.

In this paper, we argue for the importance of new user interfaces to address these limitations. We share a common motivation with early work using sketching to design data visualizations [15, 16] but unlike the "prototype visualizations" and "Wizard of Oz" user interfaces that resulted from that work, we utilize painting and sketching metaphors to create real, animated, data-driven visualizations.

We introduce a user interface called Visualization-by-Sketching (Fig. 1) with the goal of enabling the designer to work creatively with full attention on the design task. When working with Visualization-by-Sketching, an artist should be able to react naturally to a visualization with a color map. If she notices that an important feature in the data is not as visually salient as it should be and that it would "pop" much more if the red color at that location were more saturated, then, rather than taking her focus away from the visualization to identify and modify a mathematically-defined control point in the color map, she should simply be able to paint a more saturated red color onto the visualization at this location. Then, the system should figure out how to modify the underlying mathematical constructs in order to create the visual picture she desires. This operation should be so quick and natural that she can immediately move on to the next visual refinement. Likewise, to explore various designs for data glyphs, she should be able to simply sketch a variety of alternative shapes. Rather than editing density and size functions for placing these glyphs, she should be able to quickly explore and critique a variety of exemplars created automatically based on the current data.

We imagine that such an interface could, eventually, be added on top of many existing lower-level visualization toolkits, but defining the interface in the first place requires developing appropriate new metaphors, interaction techniques, and data-aware algorithms. The technical focus of this paper is on addressing these challenges. We accomplish this in the context of visualizing time-varying, multivariate data, a type of visualization that is particularly challenging to design from a perceptual standpoint [22].

The contributions of this paper include:

- An approach to generating data visualizations that is designed specifically to match the creative mindset and is accessible to visual experts with traditional, non-technical training.
- A set of user interface and graphics algorithms to interpret painterly input on top of a data visualization as a method of refining a mathematically accurate color map, including a real-time method to distinguish the intention to make global vs. local edits.
- The design of an interactive sketch-based glyph editor with auto-generated side-view exemplars.
- An application of "layers", as used in image editing software, to visualization, and a discussion of nine layer types that were developed and tested.
- An extension of the core Visualization-by-Sketching system to support scripting so as to enable storytelling.
- A preliminary evaluation of the expressiveness of the interface conducted by adapting a series of previously published multivariate visualization styles to work with new datasets.
- Example results and user feedback from traditionally trained artists.

2 Related Work

Our work builds upon prior research that has also focused on the visual design aspects of visualization, as well as recent work in direct-manipulation interfaces and multivariate data visualization.

2.1 A Focus on Visual Design

Although every visualization requires some form of visual design, there are relatively few published research results that directly target the goal of enhancing the human task of visual design that is required early in the visualization process. The most prominent work in this area, like ours, recognizes the valuable role that artists might play in this process. This can be traced back to early efforts to form artist-scientist-technologist "renaissance teams" [5]. In practice, the design tools needed to accomplish this interdisciplinary collaboration are limited; traditional sketching and painting on paper is still used for much design work. One exciting exception uses a combination of expert human feedback and parameter optimization to refine multivariate visualizations [22]. A form of 3D sketching within virtual reality has also been developed, motivated by a need to design scientific visualizations directly within the medium in which they would be viewed [15, 17]. Traditional sketching is quick and expressive (two important criteria for any design tool) [4], but traditional sketching and even the digital 3D sketching explored previously suffer from a major problem in that they are detached from data. Prototyping visualization ideas in a *purely* visual sense, without any real data-driven graphics, can only be carried so far before the question is raised, how well will this work with real data?

Critique, as practiced in art and design disciplines, is another method adopted by visualization practitioners who wish to focus on visual design [19]. In fact, it has been demonstrated that traditionally trained visual experts, such as illustrators, can rate visualization designs during a critique session in ways that are congruent with results obtained through quantitative user studies [1]. We take this as strong motivation to develop design tools that make it possible for more experts outside of the technical disciplines to experiment with creating data-driven visualizations of their own.

2.2 Direct-Manipulation User Interfaces

A few visualization applications provide early inspiration for the type of visual design in the context of data that we believe is possible. Like Visualization-by-Sketching, these all utilize a direct manipulation interface, where the user manipulation occurs (via mouse or pen input) directly on top of the data visualization and in direct response to the visuals displayed there. In this style, *Drawing with the Flow* enables users to position, crop, edit, and delete streamlines by sketching on top of fluid flow visualizations [24], and *WYSIWYG Volume Rendering* addresses the difficult problem of defining effective transfer functions by enabling users to "paint" color directly onto 3D graphics [11]. These and similar systems (e.g., painting on slicing planes [27]) are in contrast with the more common approach of adjusting visualization parameters via scripting, buttons, sliders, or other widgets.

These more direct interfaces appear to have two key advantages: First, they hide unneeded technical complexity. For example, in volume rendering, a visualization designer does not need to know the fact that a multi-dimensional, texture-based transfer function is used; she simply needs to know that if she wants to change the color of the bone displayed in the image, then she should paint a new color on top of it. Second, they make it possible for the user to focus attention on the visual task at hand. When a visual designer sees an image, she should be able to "riff" off of it, reacting immediately to explore a variety of visual edits and refinements; thus, there is a benefit to keeping the focus on the visualization, not a scripting window or panel of input parameters to the side.

2.3 Multivariate Visualization

Research on multivariate visualization has a rich history and is still active today. Researchers have addressed this problem through a variety of means, including color weaving [28], interactive techniques, and glyphs, and have drawn inspiration for the visual designs they create from sources as diverse as naturally occurring textures [14] and brushstrokes in oil painting [12, 18, 20]. Our goal is not to introduce a single new multivariate visualization technique to add to this literature but rather to utilize the difficulty of this design problem for inspiration and to create a tool with which users can contribute new ideas to

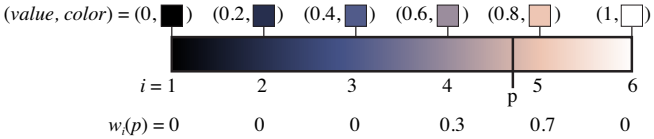

Fig. 2: Schematic of color maps as utilized in Visualization-by-Sketching. The function $w_i(p)$ returns for a given pixel, p, within the data visualization the weight (influence) that control point i has on the color of that pixel. An example of the weights for a pixel with data value = 0.74 is shown.

address the problem. Multivariate visualization is applicable to a number of different datasets, but most research has aimed to determine the most effective methods to display a combination of one or more scalar fields along with one or more vector fields, each defined over the same spatial domain.

3 Visualization-by-Sketching

This section describes the main components of the Visualization-by-Sketching interface. The style of direct-manipulation visual design motivated thus far is best exemplified by the *paintable color maps* feature. We begin the discussion with an overview of this technique. This is followed by a description of the data glyph editing interface. In Visualization-by-Sketching, color maps, glyphs, and other graphics are arranged in layers. Following the model utilized in most current image editing tools, the order of layers can be adjusted interactively as can the transparency and the method used for blending adjacent layers (e.g., normal compositing, multiply, screen). To date, we have implemented nine data-aware layer types. We conclude the section by describing the additional layer types, a scripting interface that enables Visualization-by-Sketching to be used for simple storytelling, and some optimizations that enable our implementation to work with multivariate datasets as large as one terabyte.

3.1 Paintable Color Maps

Color maps are effective at conveying the values in a scalar field, allowing for both local comparisons and for determining the value at an arbitrary location [30]. Fig. 2 provides a schematic of a simple color map and introduces key terminology used in our discussion. Color maps are defined by a series of *control points* that are (*data-value, color*) pairs. The example in Fig. 2 contains six evenly spaced control points, and we have assumed here that the data are normalized to fall within the range 0.0 (min) to 1.0 (max). To determine the color for a particular data value (e.g., $temperature = 0.74$, as indicated by the line labeled p in Fig. 2), the two closest control points are identified (here, control points number 4 and 5), and an interpolated version of the corresponding colors is returned.[1] Conventional color map design interfaces work by creating and deleting control points and/or editing the data-value, color pairs at each control point.

Following the motivation set forth earlier, we argue that a creative visual design may be better facilitated by an interface that is instead based upon painting color directly on top of the data visualization. The resulting interface works at the brush stroke level. A digital "airbrush" tool is used to paint a brush stroke on top of the data visualization using a variety of settings that can be adjusted interactively (color, radius, hardness, opacity, flow, and blend mode). The algorithm then interprets the brush stroke relative to the underlying data, which triggers an update in the color map. This yields a new data visualization, which fades into view as soon as the stroke is complete. The reason for the fade is that we must ensure that the final visualization is accurate with respect to the underlying data, and we are not guaranteed that the user's painterly input will result in an accurate representation. Thus, the algorithm aims to balance the goal of artistic freedom via direct painting with the need for accurate data visualization.

[1] Our implementation uses a fixed number of 20 evenly spaced control points that are initialized to a linear ramp from black to white, and all color interpolation is performed in CIE $L^*a^*b^*$ space.

3.1.1 From a Single Pixel to Strokes

To understand how paintable color maps work, let us begin by considering a single pixel p in the image. Assume, as above, that we are visualizing a temperature scalar data field, and the temperature value directly under p is 0.74. Imagine we change the color of this pixel by painting it completely orange, and we want this painting operation to update the color map accordingly. Since the pixel is not affected by control points 1, 2, 3, and 6 the colors at these control points do not need to change. Control points 4 and 5 do affect the color the pixel, but not equally (30% for control point 4 and 70% for control point 5). So, the painting operation should make the colors associated with these two control points "more orange" in a way that somehow accounts for this proportion. As diagrammed at the bottom of Fig. 2, for a specific pixel p, we use $w_i(p)$ to denote the weight that control point i has on determining p's color. The algorithm will use $w_i(p)$ as one factor in the way that the control point colors are updated.

We next need to know *how significantly* to change the color of each of the control points that have non-zero values for $w_i(p)$. Should the color change to become completely orange, or just 10% more orange? This is an important but challenging aspect of the interface because the user's intent for each brushstroke is not always the same. For example, we observed that a common first step is to "rough out" a color scheme using large, broad strokes that cover wide ranges of data. Then, over time, users shift back and forth between the goals of making refinements by painting over smaller regions of data, highlighting important specific local features, and returning to more global operations, such as brightening all the colors in wide range of the color map.

For global-style brush strokes it makes sense to adjust the color map in proportion to the amount of data covered (e.g., if the artist painted blue over 90% of the regions of low temperature, then it is clear that she wants blue to signify low temperature). However, this interpretation does not always work. For example, consider highlighting a local feature. It is natural to paint right over a local feature with the intent *make this feature orange*, but if we interpret this globally, we may find that even though we have completely painted over the entire local feature, we have only painted over 20% of the visible data with similar values. If the feature were to become only "20% more orange" after this operation, then users would be very frustrated, as the result would not match their intent. Our approach is, therefore, to characterize each stroke based upon the degree to which it should be interpreted globally vs. locally and then to adjust *how significantly* we change the color of each control point based upon this interpretation.

3.1.2 Global Interpretation

When brush strokes fit a more global style, we determine how much to adjust each control point's color based upon how completely the user has covered the data that "belongs" to each control point. First, we identify, for each control point, the total weight that it has on the displayed image. T_i, the total weight for the i-th control point, is calculated as the sum of $w_i(p)$ for each pixel in the image:

$$T_i = \sum_p w_i(p). \tag{1}$$

Then, we make a similar calculation, but just adding together the weights for the pixels that have been painted. One nuance here is that the brush is designed to have a soft edge and adjustable flow rate in order to support color blending, which makes the paint semi-transparent. Thus, a brush mask is calculated to store the degree to which each pixel has been painted as a floating point value (0.0 to 1.0). This mask is illustrated in Fig. 3. Since it is defined for each pixel in the image, we refer to it as $b(p)$. The total weight of just the painted pixels calculated for each control point can then be written as:

$$P_i = \sum_p w_i(p) b(p). \tag{2}$$

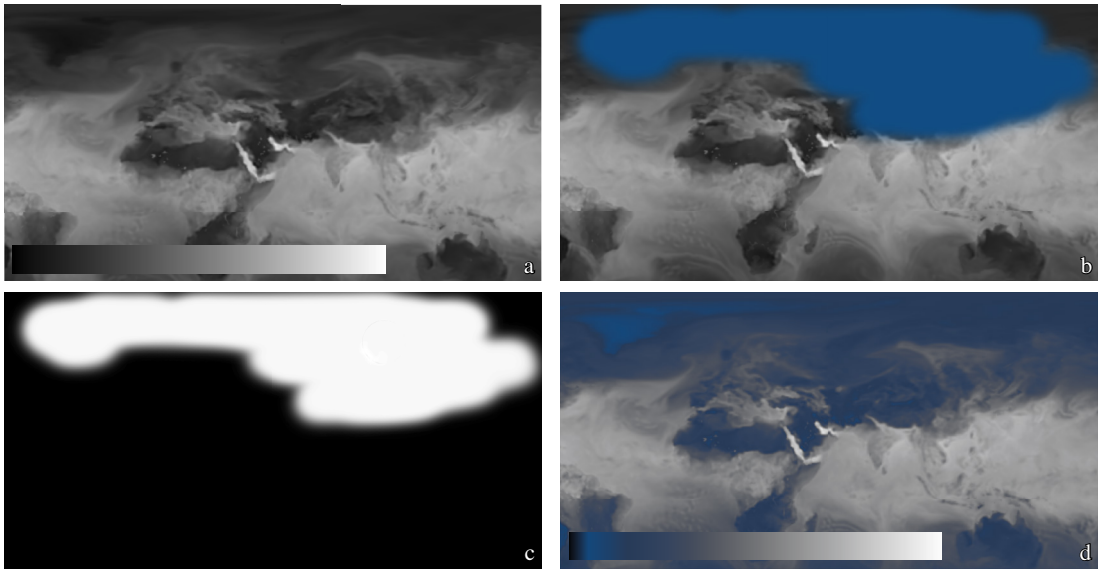

Fig. 3: An example of interpreting a brush stroke that fits a global style. (a) The original data visualization with a simple grayscale color map applied to climate data. (b) The user paints over a portion of the visualization in blue. (c) The "amount of paint" on each pixel is stored in a brush mask image. (d) After using the brush mask and color map weighting functions to determine which color map control points refer to the painted pixels, the color map is updated to incorporate the blue color of the brush stroke, and an updated data visualization fades into view.

The ratio of total weights for painted pixels relative to total weights for all pixels produces an appropriate measure of how much to adjust each control point for strokes that fit the global style:

$$G_i = P_i / T_i. \tag{3}$$

This global interpretation is balanced with the local interpretation described next.

3.1.3 Combining with Local Interpretation

Users switch naturally between global and local operations as they work, using keyboard shortcuts (e.g., Command+Plus, Command+Minus) to zoom in and out of the digital canvas and changing the size of the brush using an interactive tool palette. When brush strokes fit a more local style, we determine how much to adjust each control point's color based upon the degree to which the brush stroke succeeds at capturing a local feature. The approach is illustrated in Fig. 4. Again, the user's brush stroke is converted to a mask. But, in addition to interpreting this mask relative to the original underlying data, the mask is also interpreted relative to a local feature map designed to accentuate local features at the scale of the current brush. The local feature map is a filtered version of the original data. The filter has a high-frequency cutoff at 25% of the brush's current radius (to eliminate noise) and a low-frequency cutoff at 100% of the brush's current radius (to eliminate details much larger than the brush).[2] The local feature map (Fig. 4d) can be interpreted as visualizing the extent to which each pixel refers to a local maximum (white) or minimum (black), where the locality is defined based on the current size of the user's brush.

Using the feature map value at each pixel, $f(p)$, a local weighting scheme that parallels the global one described earlier can be computed. Mirroring equation 2, F_i is a sum of control point weights from the painted pixels, but this time modulated by the degree to which each of the painted pixels represents a local feature:

$$F_i = \sum_p w_i(p) b(p) f(p). \tag{4}$$

[2]Using a common image processing technique, the local feature map is computed by smoothing the data with the difference of two isotropic Gaussians [8, p. 199]. This computation is initiated in a background thread as soon as brush parameters are set so that the map is ready to utilize when the stroke is completed.

Mirroring equation 3, L_i is computed as a ratio, but this time rather than evaluating how much of the global data has been covered by paint, the ratio is based on how much of the painted area corresponds to a local feature:

$$L_i = F_i / P_i. \tag{5}$$

Finally, a parameter, α, is used to evaluate the degree to which each stroke should be interpreted locally (high α) vs. globally (low α), and a final set of weights, W_i, for each control point are computed based on this interpretation.

$$\alpha = |\sum_i F_i / \sum_i P_i| \tag{6}$$

$$W_i = \alpha(L_i) + (1-\alpha)(G_i). \tag{7}$$

The color for each control point is then adjusted based on these weights and the current brush color and brush blend mode. For example, with the default "normal blend" mode selected:

$$color_i = W_i * brushcol + (1 - W_i) * color_i. \tag{8}$$

The blend mode can also be set to screen, multiply, divide, lighten, or darken. Lighten and darken are particularly powerful. Using the lighten mode and a large brush, an artist can quickly paint over an entire image to lighten all of the colors in the current color map.

3.2 Interactive Glyph Design with Side Views

In addition to color maps, glyphs and particles (i.e., small, simple glyphs) have been used extensively to visualize multivariate data, especially vector fields. Glyphs can be used in both animated and static visualizations, and, as might be expected, the most appropriate design may change significantly based upon the decision to include animation or not.

The glyph design interface is pictured in Fig. 5. The current implementation, guided by design decisions discussed in the literature [9, 22, 31], assumes that glyphs have a single color, are symmetric about the direction of motion, and fade out opposite their direction of motion to a user-controlled end opacity, ranging from fully transparent to fully opaque.

The interface is divided into two regions. The glyph sketching canvas is on the left. Here, artists experiment with various shape profiles for the glyphs by sketching (with mouse or pen input) directly on top

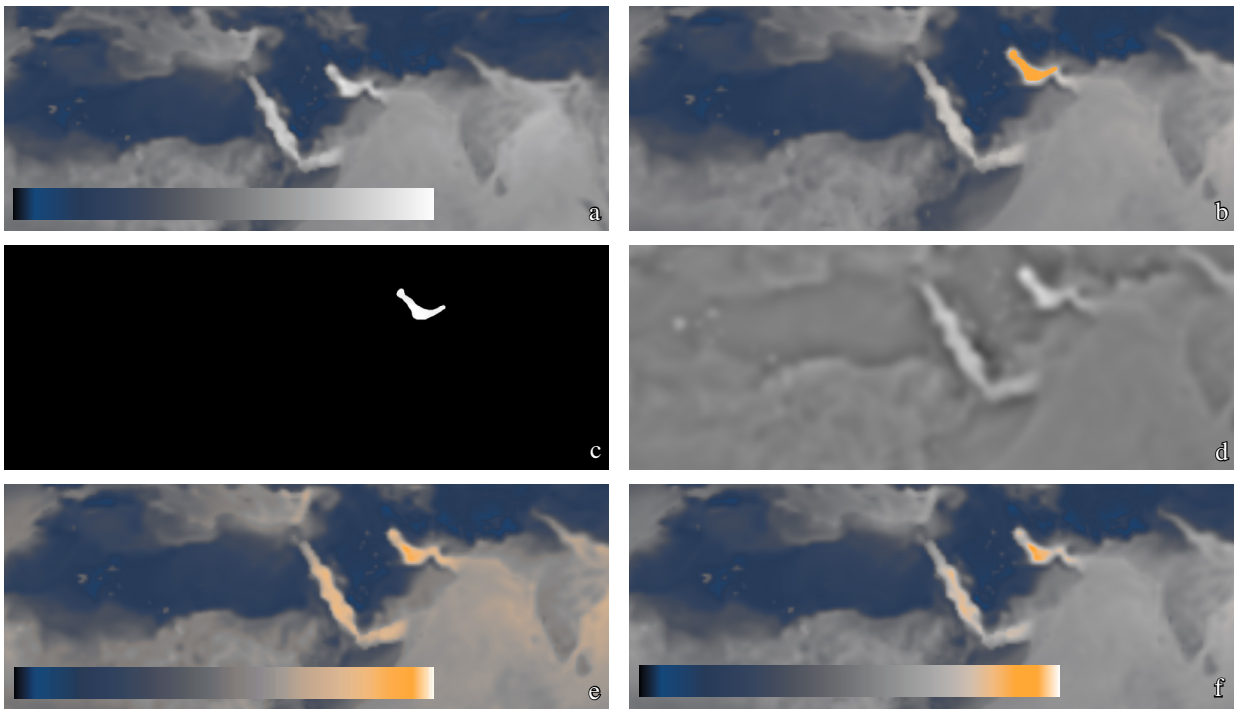

Fig. 4: An example of interpreting a brush stroke that fits a local style. (a) The original data visualization. (b) Using an orange brush, the user paints a stroke, where the intent is clearly to highlight this local feature. (c) As in the global interpretation, a brush mask is computed. (d) In addition, a local feature map is computed. Note, this bandpass filtered version of the original data contains both positive and negative values; in the image, 50% gray represents a value of zero. (e) The strong correlation between the user's brush stroke and a local feature leads to a local-style interpretation of the brush stroke and produces this image. (f) In contrast, if purely global interpretation were used, this is what the resulting image would look like. Notice that the effect of the local interpretation is to amplify the strength of the painting operation.

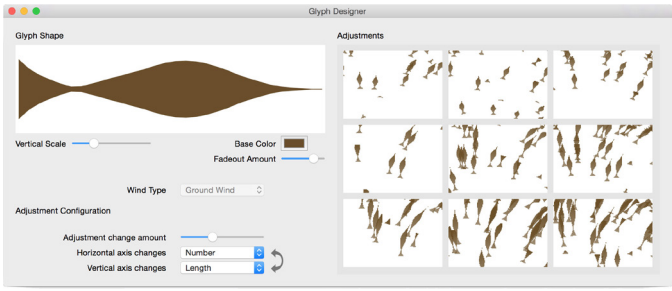

Fig. 5: The glyph design interface with sketching canvas on the left and animated previews on the right.

of the current profile. Since the profiles are symmetric, a single curve drawn on the top half of the canvas will be automatically mirrored on the bottom half as it is drawn. In addition, to support a range of possible widths, the vertical scale of the glyph sketching canvas can be adjusted via a slider. The base color, degree of fade out, and variable attached to the glyph can all be edited interactively as shown in Fig. 5.

The right side of the interface aims to support creative exploration of design alternatives by presenting artists with automatically generated exemplars. As in early work on supporting creativity in open ended tasks via Side Views [26], the intent is to "support the serendipitous discovery of viable alternatives" by presenting alternatives at the right time and enabling experimentation. This is accomplished by presenting a grid of design variations similar to the parameter spectrums in Side Views and to the choices available in Design Galleries [21], but adapted here to display animated, data driven graphics. The exemplars demonstrate how the current glyph would appear when animated and viewed with different settings for graphics rendering parameters, such as speed of advection, particle density, particle length, and particle lifetime. A preview based upon the current visualization settings is shown in the center, and variations based on two other parameters selected interactively by dropdown box (one tied to the horizontal axis of the grid and one the vertical) are automatically computed and displayed in the surrounding cells. The degree of change between the current visualization settings and the automatically generated variations can be set interactively via a slider. Clicking on the cell of one of the variations selects the associated parameters and updates all of the cells accordingly.

In all of the glyph-based visualizations, glyphs are seeded randomly in space and time. The seed points are distributed within a region centered on the visible area on screen, but expanded by 20% in each direction to ensure that glyphs do not appear to be sparser near the perimeter of the screen. Each glyph is integrated through the vector field using Runge-Kutta integration for a configurable number of timesteps. The interface includes a widget for adjusting the currently displayed timestep, entering a play mode to advance through time automatically, or pausing. Animation is available in all modes. In the play mode, the glyphs advect following the time-varying vector field, and when the timestep is fixed, the glyphs advect based on the static vector field at the current timestep.

3.3 Additional Layers

Visualization-by-Sketching supports a number of additional layers beyond the previously mentioned color map layer and particle layer. The layers are organized into three categories (Fig. 6).

Data layers, such as the color maps and glyphs discussed thus far, are used to depict data from the underlying dataset. Other data layers include an isocontour layer that can display any data variable as a set of isocontours with user-controllable spacing. A measurement layer enables users to select specific spatial locations where the value of a scalar variable will be displayed as text. The locations are specified simply by clicking on the visualization. A graph layer can be used to display a small graph of a variable either over time or with respect

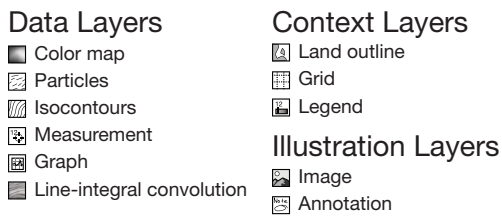

Fig. 6: Visualization-by-Sketching layer types.

to another variable. A line-integral convolution layer provides either a static or animated visualization of a vector field, using the line-integral convolution technique [7].

Context layers provide information about the visualization as a whole, and do not display specific data variables. These layers include a land outline layer, a grid layer, and a legend layer. The grid layer provides both a geographic grid showing lines of longitude and latitude, as well as a data grid, showing the size of the individual data cells. Lines of longitude and latitude fade in and out as the display zooms in order to maintain a user-specified density, and data cell lines only appear when each individual cell is over a user-specified size.

Illustration layers add additional information to the visualization. Image layers can be used creatively to add elements that provide context for what is currently being shown, such as photographs, arrows, or other visuals from outside sources. Images can be positioned relative to the data, or can be positioned in screen space. Annotation layers are used to draw colored lines, text, or other hand-drawn annotations on top of the visualization.

3.4 Story Telling

We have also extended the core Visualization-by-Sketching interface to support scripting. Scripts are made up of scenes, and scenes are defined by a set of active layers, a geographical position and zoom level, a duration to control how long the scene is visible, and parameters to control a transition to the next scene. Using scripts it is possible to create short animated sequences to highlight multiple aspects of the data, a feature we envision being useful for story telling and conveying findings to a broad audience via animation (e.g., Fig 7).

3.5 Data Access and Management

Data files are processed prior to being loaded to convert them into regularly gridded files of IEEE floats that are then used for direct memory mapping as implemented via the operating system's virtual memory subsystem. This eliminates parsing overhead and means that data can be accessed as if already resident in RAM, with the operating system transparently loading and caching the data as needed.

4 Implementation and Application Datasets

To test Visualization-by-Sketching, we have applied it first to the challenge of visualizing multivariate climate data. The examples presented in this paper come from two National Oceanic and Atmospheric Administration (NOAA) datasets. The first is the Climate Forecast System Reanalysis (CFSR), which provides hourly data over the entire Earth from 1979 until 2009 [23]. We visualize a 1.2 TB subset of the data that contains 8 scalar fields (geopotential height, humidity, precipitable water, precipitation, sea-level pressure, surface pressure, temperature, wind speed (10m)) and one vector field (wind velocity (10m)) for 11 years of observation at one-hour intervals and 0.5 degree resolution across the earth.

The second dataset is the North American Mesoscale Forecast[3]. We visualize a 360 MB subset of the data that contains 11 scalar fields (cloud cover, geopotential height, humidity, precipitable water, precipitation, pressure (jet stream), sea-level pressure, surface pressure,

[3]http://www.ncdc.noaa.gov/data-access/model-data/model-datasets/north-american-mesoscale-forecast-system-nam

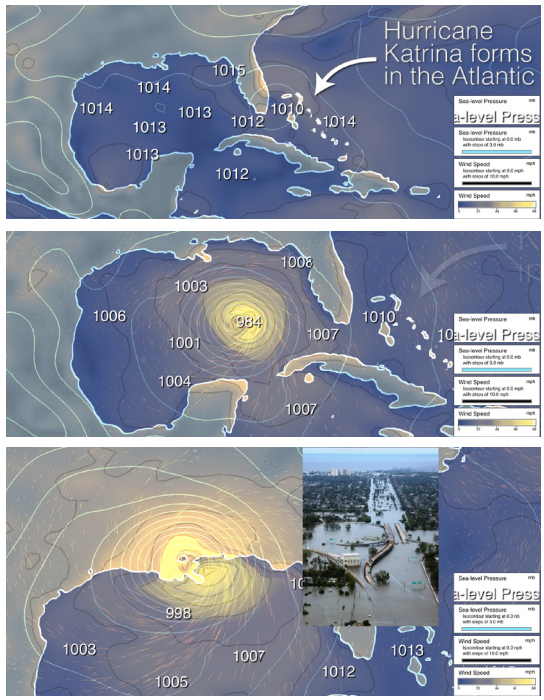

Fig. 7: Three Visualization-by-Sketching scenes are combined to tell a story about Hurricane Katrina.

The current implementation runs on Mac OS X systems and has been tested successfully on several different computers. A typical configuration used in our lab is an iMac purchased in late 2012 (OS X 10.10.3, Screen Resolution 2560x1440, 2.9 GHz Intel Core i5, 16 GB RAM, NVIDIA GeForce GTX 660M with 512 RAM), an external 2 TB hard drive, and a Wacom Cintiq pen display tablet. This configuration supports interactive access to eleven years of global climate data for the CFSR dataset described above. The current implementation is not limited to working with climate data specifically, but new multivariate datasets may require preprocessing to be defined on a regular grid and stored as IEEE-format floating-point numbers. The visual designs and scenes generated can be saved, loaded, and shared with other users via a custom XML-based file format. Visualizations can be viewed within the software or saved as images or movies using screen capture software.

5 Results and Discussion

We present two types of results. First, as a preliminary evaluation of the expressive capability of the current set of features, we used Visualization-by-Sketching to recreate a series of previously published multivariate visualization styles. Second, we report on results and feedback from artist users.

5.1 Recreating Multivariate Visualization Styles

To test Visualization-by-Sketching we created a series of visualizations of the NOAA datasets in styles that mimic some multivariate visualization techniques that have previously appeared in the literature [18, 20, 28, 29, 31]. We selected this set of five examples based upon the wide range of visual attributes evident in them (e.g., black and white vs. extensive use of color, complex glyphs vs. simple particles, animated vs. static). Fig. 8 visually summarizes the results. Here, we comment on the degree to which each of these visualization styles can be expressed using the current feature set of Visualization-by-Sketching and discuss lessons learned.

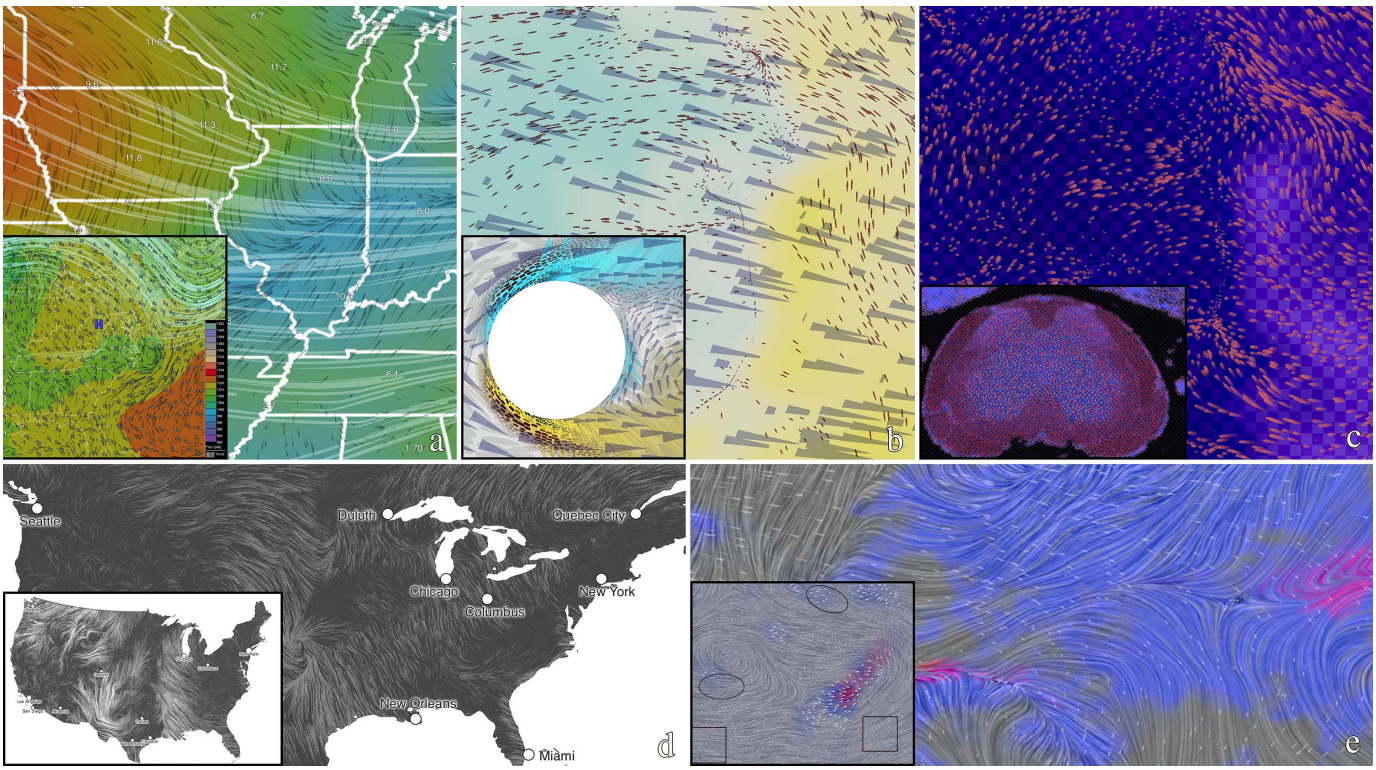

Fig. 8: A series of climate visualizations with visual styles inspired by previously published multivariate visualizations of both climate and other multivariate datasets. Inset images show the original inspiration for each visualization design. Inset images reused with permission. Inset (a) copyright Ware 2015 using Ware and Plumlee 2013 [31]. Inset (b) copyright Kirby et al. and IEEE 1999 [18]. Inset (c) copyright Laidlaw et al. and IEEE 1998 [20]. Inset (d) copyright Viégas and Wattenberg 2012 [29]. Inset (e) copyright Urness et al. and IEEE 2006 [28].

Fig. 8a follows the style of an improved weather map designed by Ware and Plumlee that depicts multiple variables at once: temperature, barometric pressure, and surface and jet stream wind [31]. The several weather map designs introduced in the original work are carefully crafted and detailed with many alternatives considered during development and testing and strong perceptual grounding for each visual design decision. As with each of the examples in this section, we discovered both successes and limitations in our attempt to recreate a similar visualization using Visualization-by-Sketching. Here, several features of the original are beyond the capabilities of Visualization-by-Sketching, including intelligent strategies to filter slower particles out of view, automatically place text, non-linearly scale particle advection speeds, and automatically identify highs and lows. Nevertheless, with Visualization-by-Sketching, we can mimic, just through the sketch-based interface, some of the key elements of this style. We began by adding the color map layer. Initial, large strokes were used to get the general structure of the color map, and smaller, more detailed strokes were used to refine it. This produced a sequence of colors similar to the original work, but Visualization-by-Sketching does not yet support banded color maps, an important perceptually-motived feature of the original design. We then added two glyph layers, setting one to depict the ground-level winds and the other to represent jet-stream winds. We adjusted the shape, speed, and count of these glyphs to achieve a similar appearance to the Ware and Plumlee visualization. Then, the white outlines for each state were added using an image layer. Finally, a measurement layer was added to provide the textual data displays at specific locations identified by hand. From start to finish, the visualization pictured here took the lead author of the paper 44 minutes to create.

Fig. 8b is in the style of a flow visualization developed by Kirby et al. that depicts several data fields using techniques inspired by traditional oil painting [18]. The original visualization depicts nine values in one image. Visualization-by-Sketching can support several of the same mappings to produce a multivariate visualization in a similar visual style; however, we note that Visualization-by-Sketching does not yet support varying glyph color, width, or opacity in response to a changing scalar field. Our triangular glyphs are a simpler version of the original. In order to completely recreate the original style, Visualization-by-Sketching would also need to support working with 2D tensor values (depicted by the ellipses in the original work). This result took the lead author of the paper 15 minutes to create.

Fig. 8c is in the style of a visualization of diffusion-tensor imaging of a mouse spinal cord created by Laidlaw et al. [20]. We were able to achieve a visual style (e.g., color scheme for background and glyphs, underlying data grid) that is clearly motivated by the original data-to-visual mappings. However, the two datasets here are quite different. The original data include 3D tensor values, whereas our visualization is based upon the 2D vector and scalar fields available in our test datasets. Adding support for tensor data would be an exciting and challenging direction for future work. This result took the lead author of the paper just 6 minutes to create.

Fig. 8d is in the style of a recent art project by Viégas and Wattenberg [29]. The simplified color palette and choice to display wind data only over the land masses are clearly intentional. Rather than more complex, colored glyphs, the flow patterns here reveal themselves through the use of animation and varying density. There is an interesting use of an image layer in this example. To place markers and labels for key cities, we simply used an online map viewer to navigate to a similar vantage point of the continent, captured an image of the screen with cities highlighted, then used an image editor to create a mostly transparent image with just a bit of text and circles in the correct places. The result was then imported into Visualization-by-Sketching as an image layer and positioned correctly to match the data visualization. This visualization took the lead author of the paper 13 minutes to create.

Fig. 8e is in the style of a flow visualization introduced by Urness et

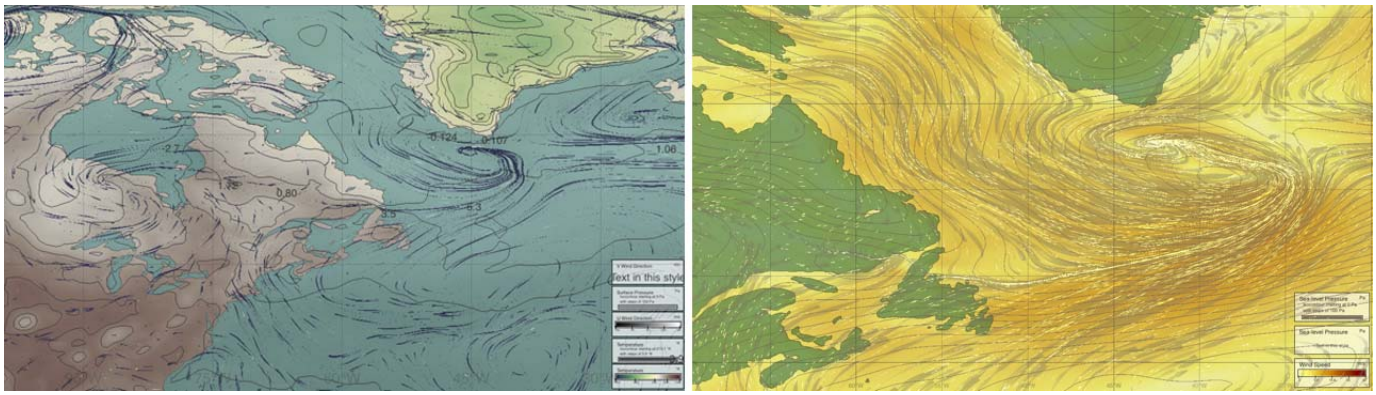

Fig. 9: Two additional results created by artist collaborator, Francesca Samsel.

al. that uses "color weaving", particles, and line integral convolution to display multiple scalar and vector fields together [28]. Visualization-by-Sketching can be used to create a result with a similar color scheme and combination of line integral convolution and glyphs/particles. The color weaving introduced in the original work requires a level of sophistication that goes beyond simple color blending, so this aspect cannot currently be recreated in Visualization-by-Sketching. This result took the lead author of the paper 37 minutes to create.

Although none of these results is a perfect match with the original multivariate visualization algorithm, what we find interesting about these results is that in less than two hours we were able to: 1. look at an inspirational image of a multivariate data visualization style, 2. using only sketching, painting, and layering create an accurate data visualization of a different dataset using (or at least motivated by) the original style, and 3. repeat this process five times. We also learned that supporting just a few more glyph customization options, blend modes, and data types would significantly improve the expressive power.

5.2 Evaluative Feedback and Visual Designs from Artists

We are eager to get Visualization-by-Sketching into the hands of artists and other visual experts and have done this in two capacities thus far. First, over the course of the past two years, as we have iteratively developed the interface, we have sought feedback from professors and students at local colleges of art and design. These have typically been short sessions of demonstrations and discussions, but they have been valuable in shaping the design of the interface and have included feedback from an art professor specializing in new media and a design professor specializing in infographics.

Second, we deployed two successive versions of Visualization-by-Sketching to an expert artist collaborator, Francesca Samsel, who used the tool on her own mac computer, first to provide additional evaluative feedback and then to develop the animated visualizations pictured in Figs. 1 and 9. It is worth noting that Samsel is just the type of user that Visualization-by-Sketching aims to support. As a visual artist she specializes in science-related large-scale outdoor public art. In the last five years she has been working with visualization scientists at the Texas Advanced Computing Center and Los Alamos National Labs, using her artistic skills to create visualizations tailored to the needs of domain scientists.

5.2.1 The Need for Visual Design Tools

In our discussions, all of the artists and designers have immediately provided the same high-level feedback: tools in this style are needed. The need for rapid visual exploration is consistently cited, and it is emphasized that, regardless of the domain, design is an iterative process. Thus, the ability to rapidly create and explore potential visualizations is critical. Samsel notes a similarity to thumbnail sketches:

> Visualization-by-Sketching provides a means of creating thumbnail sketches with the data of the specific visualization problem, which I can review with the domain scientists to provide the best solution for their needs. What Visualization-by-Sketching provides is the ability to try out dozens of solutions. This is the most critical step in the creative design process. It is the ability to look at options, improve on them and experiment in multiple directions that enables the creative leaps.

5.2.2 Visualizations Created, Feedback on Specific Features

Figs. 1 and Fig. 9 show visualizations designed by Samsel during her recent use of the tool. The image in Fig. 9 (right) demonstrates an interesting experiment. Here, two layers of glyphs, each of different shape and color, are used in combination, but both to depict the same variable. The result seems to have a textural effect that could be useful for supporting a variety of analysis tasks. This is an example of the type of discovery that we believe will be common when using Visualization-by-Sketching but is uncommon with current tools. It was serendipitous that the artist noticed this effect while experimenting with different options. Since this design breaks a common assumption about the use of glyphs, it is unlikely this design would even have been attempted if she had been using a tool that required more effort in order to experiment.

As a result of her experiences, Samsel rated the glyph design tool as the single highest impact feature in Visualization-by-Sketching. Specifically, she wishes to utilize this tool to design what she calls "visual sets", combinations of colormaps and multiple glyphs that enable examination of multiple variables within one visualization. Being familiar with powerful scientific visualization toolkits, such as Paraview [13], it is interesting to note as we interpret this feedback that what we find with Visualization-by-Sketching is not that the interface technically enables a particular shape of glyph or data mapping that would not be possible to create with other tools. Rather, the novelty and potential for impact comes from the method used to create visualizations, which, through the use of sketching, exemplars, and direct manipulation, enables a new level of visual experimentation with data without the need for programming, scripting, or deep knowledge of graphics algorithms. Her feedback is also clear that a higher level of control over shape is needed in this interface in order to finely tune the details of the designs.

In collaborations with artists and designers, our experience was that the paintable color maps were such a departure from the way we usually approach color map design that this feature took considerable time to explain. It was useful in these discussions to introduce the concept of using direct painting on the data canvas to "nudge" the colormap toward a desired result. When this was grasped, there was excitement about the interface, which can feel almost magical. Users also acknowledged, that there can be advantages to a more traditional mode of color map design. For example, in early demonstrations of the tool, we received the comment, "I like being able to use these brushes to edit color directly... [but] if I want to make a specific [data] value red, it is hard to do so."

5.3 Limitations and Future Work

As is often the case when working with expert users, some of the most valuable bits of feedback have come in the form of suggestions for future work. A first theme in this feedback is a desire for even more aesthetic control. One designer specifically asked for "more typographic freedom". For users trained in graphic design or art, it is natural to analyze and critique an image at this level of visual detail. A similar bit of feedback is the desire for more control over glyph sketching and the details of text, line weights, and composition on the page for legends and graphs.

Artists and designers have been interested in using the tool to visualize other datasets, for example, specific requests were made for stock market and volumetric datasets. Finally, there was a suggestion to support interactive data querying.

Our uses of Visualization-by-Sketching have so far been limited to 2D multivariate visualizations. The results indicate flexibility in terms of visual designs supported, but also limitations in terms of the ability to recreate some specific advanced visualization features. Some examples are documented in Fig. 8, another is support for multidimensional transfer functions. In the future, we plan to adapt Visualization-by-Sketching to work with volumetric data with a specific emphasis on medical visualization. As we expand to new datasets, we anticipate a need for artist-friendly tools for processing and formatting raw data to work with Visualization-by-Sketching, something that was handled for artists by pre-loading datasets in our current implementation.

Our planned future work also includes a more sophisticated evaluation of the tool, including feedback from a broader range of artist and scientist users, as supported by open distribution of the software.

6 Conclusion

Visualization-by-Sketching contributes a new approach to data visualization designed to match the iterative, creative, hands-on style of working taught in traditional design disciplines. This work provides evidence that visualization design tools that are accessible to artists and other visual experts with traditional, non-technical training can support the creation of difficult-to-design multivariate data visualizations. We hope that interfaces in this style not only lead to better visualization designs but also an opportunity for more creative visual thinkers to become involved in visualizing complex datasets.

Acknowledgments

Thanks to artist Francesca Samsel for valuable feedback on Visualization by Sketching and assistance creating example data visualizations and to students and professors at the Minneapolis College of Art and Design and the University of Minnesota College of Design for feedback on early versions of the interface. Public data provided by NOAA and the National Centers for Environmental Information. This work was supported in part by the National Science Foundation (IIS-1218058).

References

[1] D. Acevedo, C. Jackson, F. Drury, and D. Laidlaw. Using visual design experts in critique-based evaluation of 2D vector visualization methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):877–884, 2008.

[2] P. André, M. Schraefel, J. Teevan, and S. T. Dumais. Discovery is never by chance: Designing for (un)serendipity. In *Proceedings of the Seventh ACM Conference on Creativity and Cognition*, C&C '09, pages 305–314, New York, NY, USA, 2009. ACM.

[3] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, Dec. 2011.

[4] B. Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, 2007.

[5] D. Cox. Using the supercomputer to visualize higher dimensions: An artist's contribution to scientific visualization. *Leonardo*, 21(3):233–242, 1988.

[6] D. Cox. Metaphoric mappings: The art of visualization. In P. Fishwick, editor, *Aesthetic computing*, pages 89–114. MIT Press, Cambridge, MA, 2006.

[7] G. Erlebacher, B. Jobard, and D. Weiskopf. Flow textures: High-resolution flow visualization. In C. D. Hansen and C. R. Johnson, editors, *The Visualization Handbook*, pages 279–293. Elsevier, 2005.

[8] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Pearson, 2nd edition, 2011.

[9] D. Fowler and C. Ware. Strokes for representing univariate vector field maps. *Graphics Interface 89*, pages 249–253, 1989.

[10] J. S. Gero and M. L. Maher, editors. *Modeling Creativity and Knowledge-Based Creative Design*. L. Erlbaum Associates Inc., 1993.

[11] H. Guo, N. Mao, and X. Yuan. WYSIWYG (what you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2106–2114, 12 2011.

[12] C. G. Healey, L. Tateosian, J. T. Enns, and M. Remple. Perceptually based brush strokes for nonphotorealistic visualization. *ACM Transactions on Graphics*, 23(1):64–96, Jan. 2004.

[13] A. Henderson. *ParaView Guide, A Parallel Visualization Application*. Kitware, Inc., 2007.

[14] V. Interrante. Harnessing natural textures for multivariate visualization. *IEEE Computer Graphics and Applications*, 20(6):6–11, Nov. 2000.

[15] 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.

[16] D. F. Keefe, D. A. Feliz, T. Moscovich, D. H. Laidlaw, and J. J. LaViola Jr. CavePainting: A fully immersive 3D artistic medium and interactive experience. In *Proceedings of I3D 2001*, pages 85–93, 2001.

[17] D. F. Keefe, D. B. Karelitz, E. L. Vote, and D. H. Laidlaw. Artistic collaboration in designing VR visualizations. *IEEE Computer Graphics and Applications*, 25(2):18–23, 2005.

[18] M. Kirby, H. Marmanis, and D. H. Laidlaw. Visualizing multivalued data from 2D incompressible flows using concepts from painting. In *Proceedings of IEEE Visualization 1999*, pages 333–340, 1999.

[19] R. Kosara. Visualization criticism – the missing link between information visualization and art. In *Information Visualization, 2007*, pages 631–636, 2007.

[20] D. H. Laidlaw, E. T. Ahrens, davidkremers, M. J. Avalos, C. Readhead, and R. E. Jacobs. Visualizing diffusion tensor images of the mouse spinal cord. In *Proceedings of IEEE Visualization 1998*, pages 127–134, 1998.

[21] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of ACM SIGGRAPH '97*, pages 389–400, 1997.

[22] P. Mitchell, C. Ware, and J. Kelley. Investigating flow visualizations using interactive design space hill climbing. In *IEEE International Conference on Systems, Man and Cybernetics, 2009*, pages 355–361, Oct 2009.

[23] S. Saha, S. Moorthi, H.-L. L. Pan, X. Wu, J. Wang, S. Nadiga, P. Tripp, R. Kistler, J. Woollen, and D. Behringer. The NCEP climate forecast system reanalysis. *Bulletin of the American Meteorological Society*, 91(8):1015–1057, 2010.

[24] D. Schroeder, D. Coffey, and D. F. Keefe. Drawing with the flow: A sketch-based interface for illustrative visualization of 2D vector fields. In *Proceedings of ACM SIGGRAPH/Eurographics Sketch-Based Interfaces and Modeling 2010*, pages 49–56, 2010.

[25] W. J. Schroeder, L. S. Avila, and W. Hoffman. Visualizing with VTK: A tutorial. *IEEE Computer Graphics and Applications*, 20(5):20–27, Sept. 2000.

[26] M. Terry and E. D. Mynatt. Side views: Persistent, on-demand previews for open-ended tasks. In *Proceedings of ACM Symposium on User Interface Software and Technology 2002*, pages 71–80, 2002.

[27] F.-Y. Tzeng, E. Lum, and K.-L. Ma. An intelligent system approach to higher-dimensional classification of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):273–284, May 2005.

[28] T. Urness, V. Interrante, E. Longmire, I. Marusic, S. O'Neill, and T. W. Jones. Strategies for the visualization of multiple 2D vector fields. *IEEE Computer Graphics and Applications*, 26(4):74–82, 7 2006.

[29] F. Viégas and M. Wattenberg. Wind map. `http://hint.fm/wind/`, 2012.

[30] C. Ware. Color sequences for univariate maps: Theory, experiments and principles. *IEEE Computer Graphics and Applications*, 8(5):41–49, 1988.

[31] C. Ware and M. D. Plumlee. Designing a better weather display. *Information Visualization*, 12(3-4):221–239, 2013.