Drawing with the Flow: A Sketch-Based Interface for Illustrative Visualization of 2D Vector Fields

D. A. Schroeder, D. Coffey, and D. F. Keefe
Department of Computer Science and Engineering, University of Minnesota

Abstract

This paper presents Drawing with the Flow, a sketch-based interface for illustrating 2D vector fields. Drawing with the Flow explores the problem of making scientific visualization tools accessible to artists and illustrators, who have a finely tuned visual design sense, but do not typically have the programming or mathematical background required to work with modern visualization algorithms and tools. Using a sketch-based interface that is accessible to illustrations, Drawing with the Flow makes it possible for illustrators to explore new visual designs for streamline placement in order to create illustrations of 2D vector fields, such as simulated fluid flows. The interface includes a method for interpreting hand-drawn marks relative to an underlying vector field, utilizing an “ink-data settling” procedure to subtly maintain an image consistent with the underlying data. A variable-density interactive streamline seeding algorithm tied to sketch-based input is also introduced, and design lessons learned and limitations are discussed. Several example illustration results have been created during sessions ranging from five to twenty-five minutes. The results presented demonstrate different styles of illustration for describing simulated data for 2D flows past a cylinder and 2D flows that include several different types of critical points.


1. Introduction

Scientific visualization has a rich, lengthy history of drawing upon artistic and illustrative techniques to represent complex data. The work of Leonardo DaVinci is perhaps the best classic example of combining art/illustration and science. His pen and ink studies of water [LS05] are timeless examples of the power that an illustrator’s eye (and pen) can bring to visualizing complex scientific phenomena, including fluid flows.

Today, the modern field of scientific visualization continues this tradition: non-photorealistic rendering is employed to create volume rendering techniques that mimic pen and ink illustration [RE01], multi-variate data are visualized using techniques inspired by oil painting [KKL05, HE02], and many other compelling visualization strategies draw inspiration from art and illustration (e.g. [LM02, HMCM]). One limitation of the current generation of art and illustration inspired tools is that they do not, aside from a few noted exceptions [KAM”08, IEGC08], utilize interfaces that make it possible for illustrators (the artists themselves) to contribute to the process of visualization. Medical illustrators, artists, and visual designers train for years to develop highly-refined visual skills, yet, for the most part, current tools and algorithms attempt to replicate the traditional insights of illustrators rather than engage with current visual thinkers to solve new challenging visualization problems. Our work is motivated by the belief that if we can develop new strategies that make it possible for medical/scientific illustrators and other visual thinkers to work with data-driven visualization software, then this can lead to a new ability to create more effective and engaging visualizations that can be used for data analysis, teaching, and disseminating scientific results.

To this end, this paper introduces a new sketch-based interface we call “Drawing with the Flow”. The interface targets a specific visual design problem in the area of 2D vec-
Figure 1: An illustration of simulated 2D fluid flow past a cylinder created using Drawing with the Flow.
pen-and-ink styles of illustration, including methods for increasing emphasis in certain portions of an image by adding detail via additional pen strokes [RE01]. Many other examples build on related illustration+visualization themes, e.g. [SJE05, HMC3].

Somewhat surprisingly, given the wealth of recent research in this area, there are only a few examples of research in art-inspired visualization that include real artists and illustrators in the process. Notable examples include the work of Donna Cox [Cox88] and collaborative work at Brown University and the Rhode Island School of Design, which has demonstrated the potential of using illustrators for expert critiques of scientific visualizations [JAL03] and several virtual reality interfaces for applying artistic 3D modeling to visualization design [KFM01, KZL07, KAM08]. Our work builds on these recent results and attempts to connect research in illustrative visualization to sketch-based interfaces for illustration and design. Ultimately, the aim of our work is to provide the right interface to enable artists and illustrators to contribute to solving visual design problems in scientific domains, such as flow visualization.

2.2. Optimized Streamline Placement for Flow Visualization

Streamlines and their derivatives (streaklines, pathlines) are one of the most fundamental features that can be used to describe a fluid flow or other vector field. Since streamlines are constructed by integrating through a vector field starting from a seed point, an infinite number of streamlines can be created for a given flow field by starting from different seed points. However, including too many streamlines in a single image can clutter and confuse it, thus, the topic of picking the best set of streamlines to show in order to most accurately depict the important features of a flow is one that continues to receive much attention in the literature.

Turk and Banks introduced an image-based technique to create images with an even density of streamlines by low-pass filtering the image generated and optimizing streamline placement to reduce variation in the image [TB96]. The visual results from this work were reproduced by Jobard and Lefer using a more deterministic algorithm with better runtime characteristics. This algorithm starts with an initial streamline and iteratively adds new streamlines at a specified distance from existing streamlines until the image is filled, ensuring a consistent density [JL97]. We introduce an extension to this algorithm to support variable density automatic streamline seeding. Variations in density can be useful for illustrating flows. Automatic algorithms have attempted to leverage this by seeding streamlines in specific patterns around critical points [VKB00] or by only displaying streamlines that are quite different from each other [LHS08]. Both of these algorithms are motivated by illustration techniques, in that they seed streamlines non-uniformly. Our interactive tool can be used to create similar results, where streamlines are placed precisely and deliberately, but through an interactive process controlled by the illustrator.

2.3. Sketch-Based Interfaces in Visualization and Design

Our work builds upon a number of recent advances in sketch-based interfaces, most notably the 3D modeling and design system, ILoveSketch [BBS08], which includes a notion of ink drying that has a similar visual aesthetic to our ink-data settling procedure. In general, our approach strives for a similar, fluid sketching, user experience. As in the Lineogrammer system [ZKF08], many of the marks drawn by the user in our system are interpreted based upon an underlying constraint, however, in our case, these constraints come from underlying 2D vector fields.

Sketch-based interfaces have been applied to visualization applications before [Ake06, SLD09]. Most closely related to our work is the exploratory flow visualization system created by Isenberg et al. [IEG08], which also supports drawing on top of fluid flows, but with the aim of exploring flow datasets through coupling animation with loose, freehand drawing. In contrast, our work strives to enable artist-refined, illustrative visualizations to convey information to other viewers. Our current implementation works with steady flows (static 2D vector fields) and creates a single image as a result. In the future, an interface similar to ours might be used to create animated illustrations, which may be particularly useful in describing how unsteady flows evolve over time or making steady flow visualizations even more engaging.

3. Drawing with the Flow

Drawing with the Flow aims to make it as easy as possible for an illustrator to explore a variety of illustration styles.
D. A. Schroeder & D. Coffey & D. F. Keefe / Drawing with the Flow: A Sketch-Based Interface for Illustrative Visualization of 2D Vector Fields

Figure 3: The illustrator can refine existing strokes in two ways. In a, the illustrator draws an extension (in blue) to the existing stroke, which the ink-data settling algorithm refines to the image shown in b. In c, the illustrator indicates that the bottom stroke should take a different path. The program first combines the original stroke and the refinement stroke, and then settles the stroke to match the underlying data.

There are three main components to Drawing with the Flow, which are described in the following sections: 1. a sketch-based interface, 2. ink-data settling, and 3. illustration with automatic streamlines.

3.1. A Sketch-Based Interface for Illustrators

Drawing with the Flow utilizes a pen-based interface. The current implementation uses a 21-inch Wacom Cintiq tablet display. When the application first starts, it displays a subtle line-integral convolution (LIC) [CL93] visualization of the vector field as an underpainting on the display in order to provide the illustrator with a visual cue for the underlying data, see background image in Figure 2. The LIC underpainting provides information for every point in the flow without including sharp, emphasized flow lines, which makes it well suited to the task of providing the illustrator with a visual cue for the underlying data without dominating the illustration that is being designed. In general, LIC has been shown to be an effective visual style for evoking the sense of a flow, although other visualization methods are preferred for many data analysis tasks [LKJ’05].

To augment the underpainting display, a local flow preview widget was developed to provide the illustrator with a finer sense of the flow immediately under the stylus, in the style of a traditional magic lens. This lens draws a small number of streamlines seeded in the vicinity of the stylus point and updates interactively as the stylus is moved on the display. The streamlines in this widget are colored to indicate the direction of the flow (orange=forward flow, blue=backward flow), which is something that is not readily conveyed in LIC-style visualizations. A number of other important design decisions are included in this widget. The seeding positions of the streamlines are arranged such that the direction from one seeding point to the next is perpendicular to the flow direction, ensuring that the streamlines adequately capture twists and turns in the flow. The size of the cursor also changes in reaction to the local speed of the flow; slower speeds are represented by shorter streamlines.

Given some understanding of the structure of the underlying vector field provided by the underpainting and the local flow preview widget, the illustrator is ready to begin designing a custom streamline visualization. This is done through sketching. When the illustrator draws a stroke, it is interpreted as either a gesture, a new stroke to add to the visualization, or a refinement of an existing stroke. A refinement stroke can be either an extension of an existing stroke or a re-routing of an existing stroke, as demonstrated in Figure 3.

Figure 4: Three gestures are supported: crop, more, and delete. In a, the illustrator draws a crop line (in blue) through the top stroke, which gets cropped to the result in b. In c, the illustrator performs the more gesture, which instructs the program to add more streamlines to the illustration, resulting in d. In e, the user scratches out a stroke, deleting it from the visualization, shown in f.

An extension of an existing stroke is indicated by drawing a line out of the end of the existing stroke, while a re-routing is indicated by overdrawing.

Several important gestures are recognized by the system, as shown in Figure 4. A stroke drawn through an existing mark in a direction roughly perpendicular to the existing mark is recognized as a crop gesture and divides the stroke at the point of intersection, deleting the smaller half. A scribble-out gesture, as used in a number of previous sketch-based interfaces, e.g. [BBS08], is useful for quickly deleting marks. Finally, “more” and “less” operations that automatically add or delete streamlines in the illustration (described
3.3. Illustration with Automatic Streamlines

While generating an illustration in this style, one tedious task that can arise is filling in a region of the drawing with “similar” streamlines. This section describes an algorithm to assist the illustrator in this task. The technique is controlled via the “more” and “less” gestures shown in Figure 4. The intent is to utilize the context provided by the illustrator’s recent actions to determine a region and style (length, density) for new streamlines to automatically add to the illustration. From the illustrator’s viewpoint, the interface should feel intuitive, e.g., “give me more of what I just did.”

3.3.1. Review of Constant Density Streamline Seeding

Our algorithm extends Jobard and Lefer’s constant density automatic streamline seeding algorithm [JL97]. The original algorithm is reviewed briefly here, our extensions are described in detail in the next section. The algorithm iteratively attempts to add new streamlines to gaps in the image given the constraint that new streamlines must be a specified distance \(d_{sep}\) away from all existing lines. The algorithm begins with a single original streamline added to a queue, then iterates as follows:

- Pop the next streamline \(S\) from the queue.
- For each sample point along \(S\), find the candidate seed points \(p_1\) and \(p_2\) that are a perpendicular distance \(d_{sep}\) from the streamline on both sides of the streamline.

(Continued)
For each point $p_1$ and $p_2$, if the distance from the point to any other existing streamline is $< d_{sep}$, discard the point.

- Otherwise, create a new streamline passing through the candidate point, integrating both forward and backward through the vector field until the streamline either exits the flow or comes too close to an existing streamline. (Too close is usually defined as within $0.5 * d_{sep}$ of another streamline.)

- Add the newly created streamline to the queue.

### 3.3.2. Extension to Support Variable Density Seeding

We introduce an extension to this algorithm to support variable spacing between streamlines, as it is an important aspect of many hand-drawn flow illustrations. Rather than a global $d_{sep}$, our approach varies $d_{sep}$ throughout the image. Since streamlines can be long and travel through a large portion of the image, setting the separation distance at the streamline level is not sufficient, thus, we store a local value for $d_{sep}$ for each sample in each streamline. This local value is used to find the candidate points $p_1$ and $p_2$ in the original algorithm.

To drive this variable density seeding strategy, the local $d_{sep}$ for each sample on each streamline needs to be computed. This value should be representative of the local separation of neighboring streamlines, thus, the algorithm sets the local $d_{sep}$ value for each streamline sample to be the distance from that sample to the closest streamline.

### 3.3.3. Automatic Streamline Generation

Figure 6 demonstrates the use of the variable density streamline seeding algorithm within a flow illustration. Several streamlines have already been drawn by hand and are shown in black. The illustrator then performs a “more” gesture, as described earlier, which initiates the automatic seeding and displays a small circular widget to indicate the gesture was recognized. Now, as the stylus continues to move in a circular pattern around the widget, a single new streamline is automatically added to the image for each quarter revolution of the pen around the circle. Since just a single streamline is added at a time, there is no need store a queue of streamlines as described in the original seeding algorithm. Instead, drawn streamlines and sample points are selected at random until a new successful candidate streamline is found.

The automatically generated streamlines can be removed by simply reversing the direction of rotation around the circle widget. Returning to the forward direction of rotation will begin adding new streamlines again, this time picking new candidate seed points. This allows illustrators to scrub back and forth using the circle widget to explore different seeding possibilities.

As shown in Figure 6, recently drawn lines are used to infer attributes for the automatically generated streamlines. The length of new streamlines is constrained to the average length of the last two lines that were drawn by the illustrator. This constraint is removed if the last two drawn lines were longer than half of the total width of the image. The system interprets this as meaning that the illustrator intends to draw long lines that likely exit the flow field on both ends.

Automatically generated lines are also constrained to appear within a region of recent activity, if such a region can be identified from the illustrator’s recent drawing. If the last two drawn lines are within a circular region with a radius of twenty percent of the image size, only candidate points inside this region will be considered.

### 4. Results

Figures 7–9 show result illustrations created using Drawing with the Flow on an Intel Core 2 Duo 2.66 GHz Mac Mini with a GeForce 9400 and a Cintiq 21UX Wacom Tablet.

Figure 7 shows two illustrations that together demonstrate how 2D simulated flow past a cylinder changes depending upon the Reynolds number used in the simulation (Re=100 vs. Re=500). Each illustration was completed in approximately twenty minutes. The illustrative style utilized adds detail to selected areas in each illustration to highlight the differences between the two cases.

Figure 8 demonstrates the variety of illustrative styles that can be achieved, even when working with the same dataset. The minimalist style on the left is similar to what a fluid mechanics professor might draw on paper to illustrate this canonical flow case to a student. The short lines style utilized the “more” gesture to quickly populate the background of the image. Each of these illustrations took from five to ten minutes to create. The final two illustrations demonstrate...
Figure 7: Two illustrative visualizations comparing flow past a cylinder with Reynolds number 100 (left) and 500 (right). The illustrator has attempted to emphasize the differences between the two flows.

Figure 8: Illustrative visualizations of the same flow in four distinct styles, varying streamline density, length, and positioning.

additional styles made possible by selectively including or excluding detail from the images. These took approximately twenty minutes to create.

Figure 9 shows illustrations of the type of 2D vector fields that are often used to evaluate automatic streamline seeding algorithms. Each of these fields contains at least one critical point. A smart selection of streamlines should make these critical points easy to identify, while a poor selection of streamlines can make them difficult to find. Each of these visualizations took approximately ten minutes to complete; the illustrator decided which streamlines to draw in each of these cases based on the goal of making critical points in the flow easy to identify.

5. Conclusion

In this paper, we presented a sketch-based system for creating illustrative visualizations of 2D vector fields which is accessible to illustrators and provides several tools to facilitate creating visualizations in a variety of illustrative styles, including styles that adjust visual emphasis to clearly depict specific features in the data. Although the approach supports a great deal of artistic freedom, it also maintains the constraint that the marks displayed are accurate to the underlying vector field data. In addition to addressing the very specific visual design problem of placing streamlines to illustrate 2D vector fields, we believe this sketch-based interface can also serve as an example to more broadly motivate future work in the area of making data-driven visualization tools accessible to illustrators, artists, and other visual thinkers.

6. Acknowledgments

The authors thank Mike Kirby and David Laidlaw for making available the vector field datasets used in this paper.

References


Figure 9: Illustrations of 4 fields containing different sets of critical points. Detail was added to highlight critical points, whereas areas of little change are left relatively open.