Quantitative Comparative Evaluation of 2D Vector Field Visualization Methods

David H. Laidlaw*
Department of Computer Science

R.M. Kirby*
Division of Applied Mathematics

J. Scott Davidson, Timothy S. Miller, Marco da Silva*
Department of Computer Science

William H. Warren, Michael Tarr†
Department of Cognitive and Linguistic Sciences

Brown University, Providence, RI 02912

Abstract

We present results from a user study that compared six visualization methods for 2D vector data. Two methods used different distributions of short arrows, two used different distributions of integral curves, one used wedges located to suggest flow lines, and the final was line-integral convolution (LIC). We defined three simple but representative tasks for users to perform using visualizations from each method: 1) locating all critical points in an image, 2) identifying critical point types, and 3) advecting a particle.

Results show different strengths and weaknesses for each method. We found that users performed better with methods that: 1) showed the sign of vectors within the vector field, 2) visually represented integral curves, and 3) visually represented the locations of critical points.

These results provide quantitative support for some of the anecdotal evidence concerning visualization methods. The tasks and testing framework also provide a basis for comparing other visualization methods, for creating more effective methods, and for defining additional tasks to further understand tradeoffs among methods. They may also be useful for evaluating 2D vector on 2D surfaces embedded in 3D and for defining analogous tasks for 3D visualization methods.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Evaluation/methodology I.3.8 [Computer Graphics]: Applications J.2 [Computer Applications]: Physical Sciences and Engineering

Keywords: Scientific Visualization, User Study, Line-integral Convolution, Two-dimensional Vector Fields, Streamlines, Iconic Textures, Image-guided Streamlines, Jittered Grid Icons, Critical Point, Advection, Fluid Dynamics, Fluid Flow

1 Introduction

Scientific visualization strives to display measurements of physical quantities so the underlying physical phenomena can be interpreted accurately, quickly, and without bias. In experimental sciences, great care is taken in choosing where the measurements will be made so that inferences about the underlying phenomena will be correct. How important is it to craft visualizations analogously, carefully placing arrows, curves, or other visual icons that display the data? What are the best ways to craft visualizations?

Questions like these about how to best design visualizations have been addressed in many references [1, 2, 3] with qualitative or anecdotal advice. For example, Ware [3] suggests that vectors placed on a regular grid are less effective than vectors placed in a streamline-like fashion. Despite such rules of thumb, however, quantitative studies of visualization methods are still very limited.

Our quantitative study of these questions began with a hypothesis of the form "When visualizing 2D vector fields, arrows distributed using method X are more effective than arrows distributed using method Y." We proposed to test the hypothesis with a user study. To perform the test we needed to define "more effective," "method X," and "method Y." We defined "more effective" via the performance of users on a set of three tasks described in Sec. 2. If users could perform the tasks more accurately and quickly using one of the methods, we would consider that method more effective. "X" and "Y" were initially the first two methods in the list below, but as we designed the experiment, we realized that broader coverage of the existing methods would be more valuable. We converged on the following six visualization methods:

1. GRID: icons on a regular grid,
2. JIT: icons on a jittered grid [5],
3. LIT: icons using one layer of a visualization method that borrows concepts from oil painting [6],
4. LIC: line-integral convolution [7],
5. OSTR: image-guided streamlines [8], and
6. GSTR: streamlines (integral curves) seeded on a regular grid [8].

We refer to each visualization method by its abbreviated name throughout the paper.

Because they form the kernel of the experiments, we first discuss the experimental tasks in Sec. 2. Details of the different visualization methods follow in Sec. 3. Sec. 4 presents the experimental design, including details of the setup and stimuli. Results are presented and discussed in Sec. 5, and conclusions drawn in Sec. 6.

* {dhl,rmk,jsdavids,tsm,mds}@cs.brown.edu
† {William_Warren,Michael_Tarr}@brown.edu

Presented at IEEE Visualization 2001
October 21 - October 26, 2001
San Diego Paradise Point Resort, San Diego

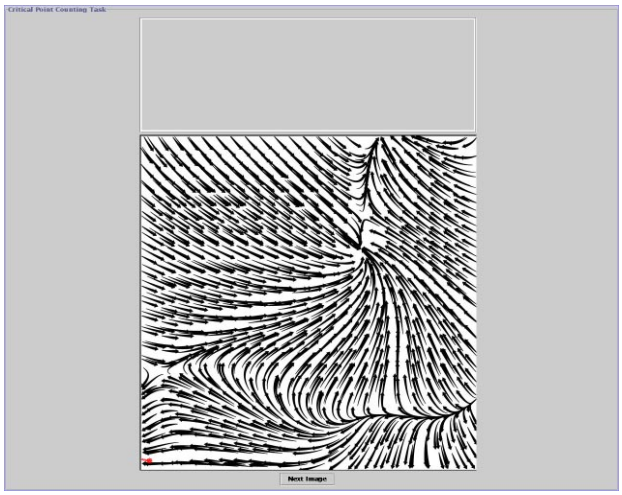

Figure 1: The experimental setup for the critical point location task. The user must choose all of the critical points in a given vector field. The LIT method is shown, but each user sees all six methods during the course of the experiment.

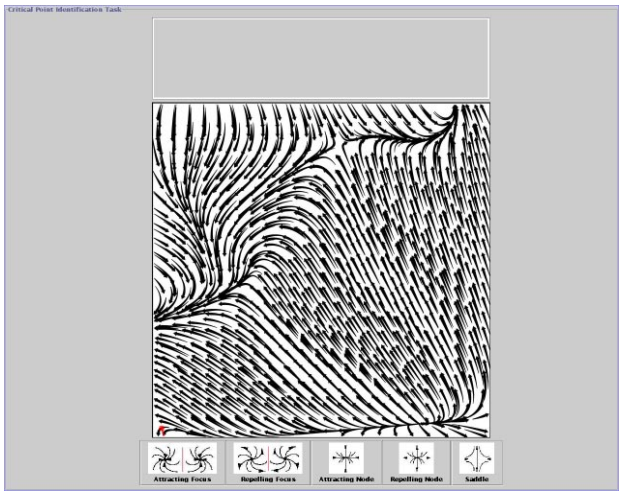

Figure 2: The experimental setup for the critical point type identification task. A red dot appears for one half second before the visualization indicating the critical point to identify. The user chooses the type for that critical point from the list of icons at the bottom.

2 2D Vector Tasks

The tasks that we defined to evaluate the effectiveness of visualization methods were carefully constructed to satisfy two main criteria. First, they needed to be representative of what users of a visualization typically do. Second, they had to be testable – they had to be simple and quick enough that a user could perform them enough times for us to calculate meaningful statistics. That implies that there exist quantitative measures of accuracy for each task, and that it be possible to generate sufficient instances of each.

We searched the literature and interviewed fluids researchers to identify representative tasks. Two of the tasks, locating critical points and identifying their types, were derived from the motivation for the development of many of the visualization methods that we are testing. Critical points are the salient features of a flow pattern; given a distribution of such points and their types, much of the remaining flow field and its geometry and topology can be deduced, since there is only a limited number of ways that the streamlines can be joined. Beyond their import for the interpretation of vector fields, these tasks are testable: we can measure how accurately a user determines the number, placement, and type of a collection of critical points in a given image.

Fig.1 shows an example stimulus for locating all the critical points in a vector field. The LIT method is used in this example. A user picks each critical point with the mouse and presses return when finished. Chosen points may not be deleted or moved because editing operations tend to significantly increase the variability of response time, making statistical comparisons more difficult. For a task that does not permit editing, mistakes will, instead, reduce accuracy. We felt that this was an appropriate tradeoff. The locations of all chosen points and the time to choose them is recorded.

Fig. 2 shows an example stimulus for identifying the type of a critical point in a vector field. A pre-image with a red dot appears for one half second before the visualization, indicating the location of a critical point. The user then selects the type of critical point from the five choices at the bottom of the display: attracting focus, repelling focus, attracting node, repelling node, saddle. The type and time to choose it are recorded.

In addition to these critical-point tasks, we identified a third task

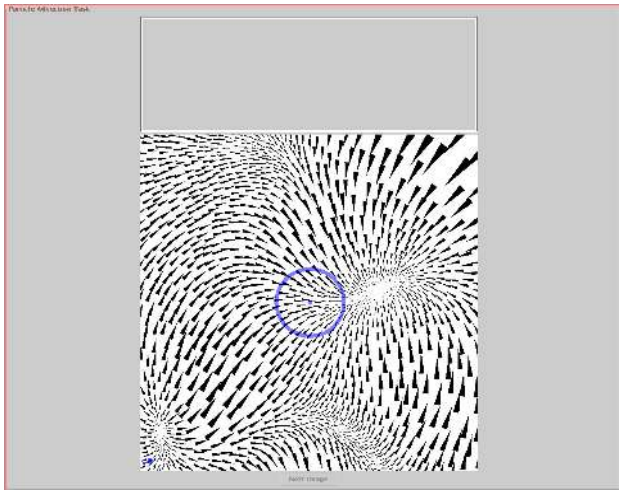

Figure 3: The experimental setup for the advection task. The user must choose the point on the circle where a particle advected from the center will end up.

that is different in character from the other tasks yet important in interpreting 2D vector fields. This task is motivated by an implicit criterion sometimes mentioned when qualitatively examining visualization methods: the ability of a method to show the flow direction. We attempted to quantify this criterion by introducing an advection task. In this task, the user must identify the location on a circle that a particle dropped in the center will end up.

Fig.3 shows an example stimulus for performing the advection task using OSTR. The user chooses the point where a particle advected from the center dot will intersect the circle and then presses return on the keyboard to move to the next stimulus. The chosen point and the elapsed time from presentation to pressing return are recorded. A small red icon in the lower left corner indicates the signed direction of the vector. This is needed for LIC, which does not capture the sign of the vector field; it is included in all of the

methods to avoid biasing results.

In summary, the three tasks are:

- choosing the number and location of critical points in an image,
- identifying the type of a critical point, and
- predicting where a particle starting at a specified point will advect.

The tasks that we have chosen are testable and, we believe, representative of many of the the real uses of these visualizations. As such, they are potentially predictive of the performance of real users in using these kinds of visualizations. Of course, these three tasks do not encompass all possible tasks for which a fluids scientist would use vector visualization. For example, combinations or modified versions of these tasks may be more or less difficult than straightforward generalizations would predict. However, performance on these tasks seems reasonably likely to generalize to performance on other similar or aggregate tasks.

3 Visualization Methods and Data

To accomplish the user study, we required a controlled set of stimuli. We first generated approximately 500 2D vector fields and then created six visualizations for each field as stimuli, one for each visualization method.

We used matlab [10] to generate the 2D vector fields. Each field was represented by a regular grid of 700 by 700 vectors and was generated in the following manner: nine random locations uniformly distributed on the interval $[0,1] \times [0,1]$ were chosen. At each random location, a vector was generated such that both components of each random vector were chosen from a uniform random distribution between -1.0 and 1.0 The x and y components of these nine vectors along with a regular grid of 700 by 700 uniformly spaced points were input to the matlab function **griddata** using the 'v4' option (for matlab's spatial interpolating function), which in turn provided x and y vector components for the interpolated vector field at each of the 700 by 700 grid points.

To calculate the user accuracy on the critical-point tasks, we needed to know the correct locations and types of all critical points within these vector datasets. The critical points were located in each vector field using a 2D Newton-Raphson method. We used 150 random initial positions for each field. Once a critical point was located, matlab routines formed the local Jacobian of the field at the critical point and determined the eigenvalues of the Jacobian, which determine the type of a critical point. The method was verified against the TOPO module of the FAST visualization environment [11] for several of the fields, and showed no errors. Data fields were discarded if they contained fewer than one or more than four critical points.

Six visualizations were generated for each vector field, one for each visualization method. The visualizations for one vector field are shown in Fig. 4. Both GRID and JIT were generated using standard matlab plotting routines. LIC [7] and LIT [6] were implemented locally. OSTR and GSTR images were made with code from Turk and Banks [8].

Each of the visualization methods has parameters that influence the images that are create, for example, the path integration length in LIC, or the grid spacing in GRID. For five values in a range for each parameter, we created three test images for each method. Each of the authors independently and subjectively estimated which value of a parameter would be best for performing each of the tasks. We then viewed them as a group and came to a consensus value for each parameter based on the tasks that we were planning. We were generally in accord on the best parameter value for a given task, but that setting sometimes differed across tasks. We tried to choose a compromise value that would work as well as possible for all three tasks. The following paragraphs describe the parameter values we chose.

For GRID, a uniformly-spaced lattice of 29×29 points was used to span $[-1,1] \times [-1,1]$. To find the x and y values of the vector at each of the given points in the lattice, matlab's 'interp' routine with 'spline' setting was used to interpolate down from the 700x700 point data set to the 29x29 point data set. The vectors were created by giving the x, y, v_x, v_y arrays to the matlab routine 'quiver,' which graphically displays vector icons. The automatic scaling provided by quiver was used; no special parameters where passed to quiver.

For JIT, a uniformly-spaced lattice of 35×35 points was used to span $[-1,1] \times [-1,1]$. For each point, (x,y), an offset was computed in both the x and y directions. The offset was uniformly distributed in $[-\frac{\delta}{2}, \frac{\delta}{2}] \times [-\frac{\delta}{2}, \frac{\delta}{2}]$ where δ denotes the spacing between uniformly-spaced grid points. Once a jittered grid was created, both the interp and quiver functions were used as in the uniform grid case to interpolate and graphically represent the vectors.

For LIT a triangle shaped wedge with a base one quarter its length represented the flow at points in the field. The area of each wedge was proportional to the speed at its center, and the wedges were placed using a uniform random distribution such that they would overlap at most 40% along their long direction and would maintain clear space between wedges of at least 70% of their width. Wedges that would not have satisfied this spacing were not kept. Strokes were placed until 250 consecutive attempts failed the spacing criteria. The overall size of the wedges was scaled so that there would be about 2000 strokes in each image.

For LIC we used a box-shaped convolution kernel of width 20 pixels. The convolution was performed on a noise image where each pixel value was set to a uniform random value in the interval $[0,1]$. To correct for loss of contrast due to the convolution, we applied an intensity mapping that took intensity I to $I^{(4/(I+1)^5)}$.

For OSTR and GSTR, the code from reference [8], version 0.5, was first modified to allow batch running without a graphical display and then to have the optimization process stop after 60 seconds without requiring manual intervention. OSTR was invoked with `opt 0.017` given to the `stplace` program, GSTR was invoked with `square 23 .2` (streamlines 20% of the image width each centered on a square grid of 23 points in each direction), and both were plotted with 'fancy arrows.' All other options to OSTR and GSTR were left as the defaults.

4 Experimental Design

4.1 Timing and Training

Fig. 5 shows the timing of the study. Users first see a text display for general training describing the goals of the experiment and the three tasks in general terms. There follow three parts of the experiment, one for each task. Within each part, an initial text display describes the task in more detail. The user is then shown a stimulus image and performs an instance of the task. Additional stimuli are presented, grouped by visualization method. In a pilot study, two users were found to converge to reasonable accuracy after 8 examples. Therefore, within each group for a given method, an initial untimed subgroup of 8 stimuli provides an opportunity to learn the task for that method. For each of these untimed cases, the correct answer is provided after the user completes the instance so

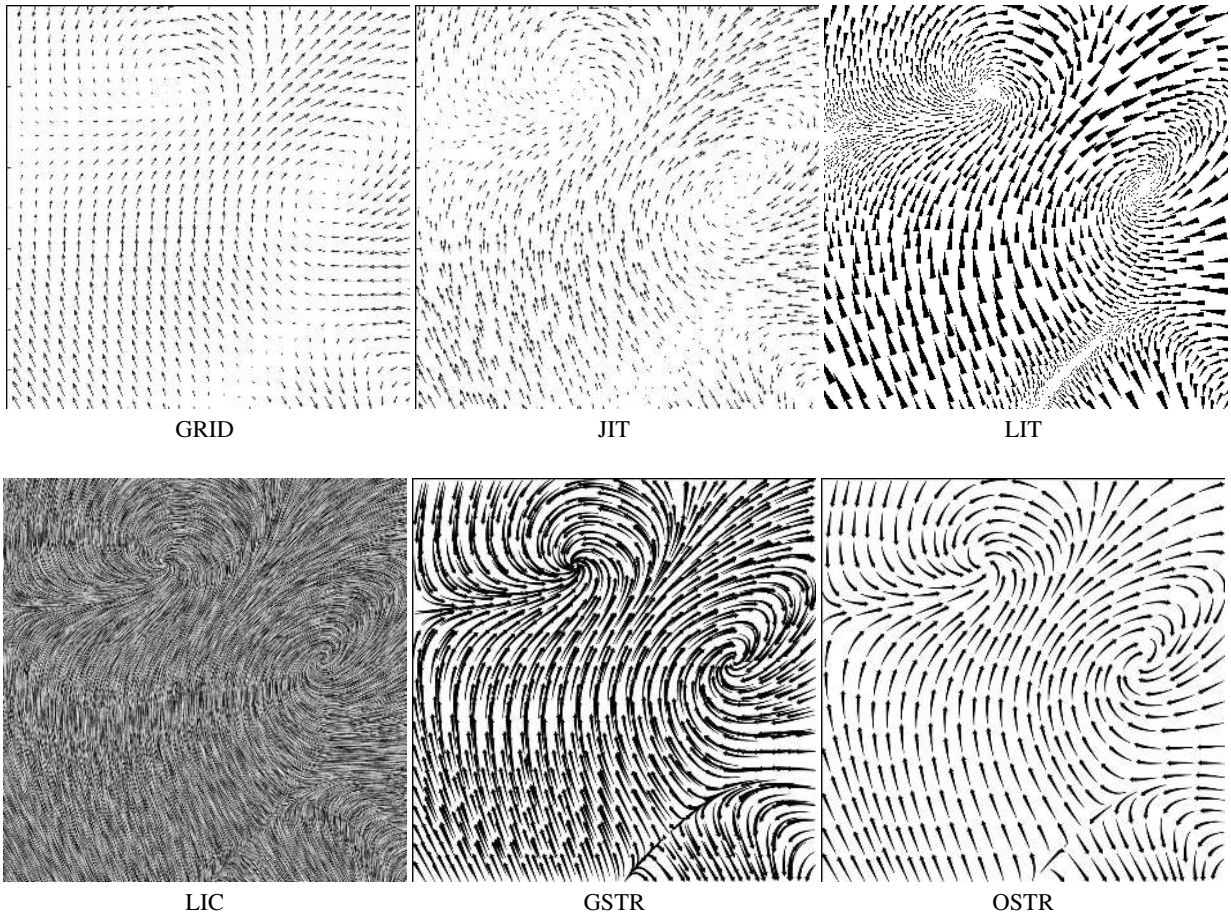

Figure 4: One of the approximately 500 vector field visualized with each of the six visualization methods.

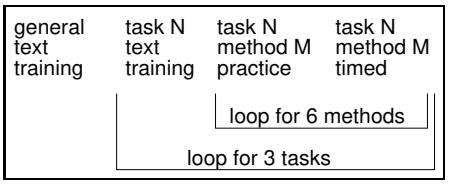

Figure 5: Ordering of tasks in the experiment.

that users are consistently trained before the timed tasks. After the training period, the user performs 20 timed instances.

A java program written specifically for this experiment presented the stimuli and recorded the data. The program pre-loaded all images at the beginning of a block so that timing would be consistent for each stimulus. The several-second pause before each block, however, did cause some small problems we discuss later.

To avoid biasing results, the ordering of tasks and of visualization methods within the tasks were each counterbalanced with a replicated randomized Latin square design [12]. For the testing (timed and recorded) phase of each task, 120 images were generated; each block of 20 images within that 120 was assigned to a visualization type per user, counterbalanced with a randomized Latin square design.

4.2 Subject Pool

Subjects were undergraduate science majors. We wanted subjects that might use such tools in the future for their work, but who had not yet started to do so. All subjects had previously studied applied math but had not studied fluid mechanics.

We deliberately chose not to use fluids researchers because we felt they might perform with a bias toward tools similar to those they already use. Future testing of this expert population, however, is likely to provide additional useful insight.

The study was designed for multiples of six subjects; data for twelve subjects was successfully acquired and is reported here. Users were paid.

5 Results and Discussion

Graphs here show the results of the data analysis. They are organized so that higher values on the vertical axes indicate greater error or slower performance (i.e., are worse). The horizontal axis shows the six visualization methods. Mean values are shown with error bars that are plus or minus one standard error. In some cases, the statistics were calculated on logarithms, to result in geometric means, and so the error bars will not appear symmetric.

Some discussion and details of the analysis, including thresholds and significance, follow. F and p values are shown in Table 1.

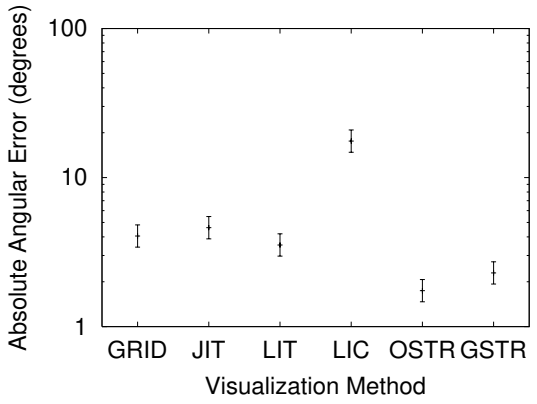

Figure 6: Geometric mean absolute angular error for advection task.

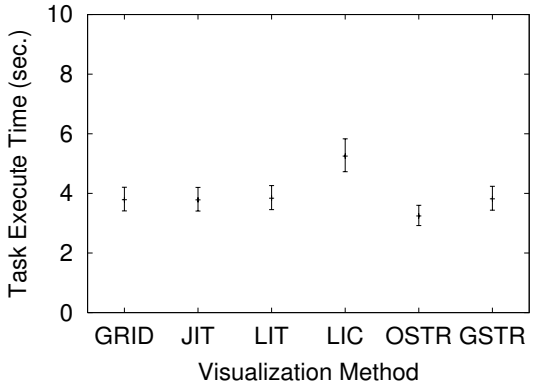

Figure 7: Geometric mean time to perform advection task.

5.1 Advecting a Particle

For the advection task, error was measured as the absolute angle error between the user-chosen intersection point and the correct intersection. Statistics were calculated on the log of the absolute angle error to normalize the distribution against the observed floor effect. Error results are shown in Fig. 6.

Error was highest with LIC. We conjecture that LIC images suffered because they do not display the sign of the vector field. The single icon in the corner is too difficult to propagate across the image to correct for this. Task accuracy appears to be slightly better for OSTR than for some of the other methods. This may be because the uniform distribution of integral curves throughout the field offers a single integral curve to follow for many advection cases. Most of the other methods require chaining together many icons to perform advection.

Tasks performance times are shown in Fig. 7. Statistics for this measure were also calculated on the log of the time to normalize against the observed floor effect. LIC was slowest, perhaps again due to directional ambiguities. OSTR was fastest, perhaps due to its uncluttered streamlines.

For this task, OSTR appears to be both accurate and fast compared to the other methods, with GSTR comparably accurate, but a bit slower.

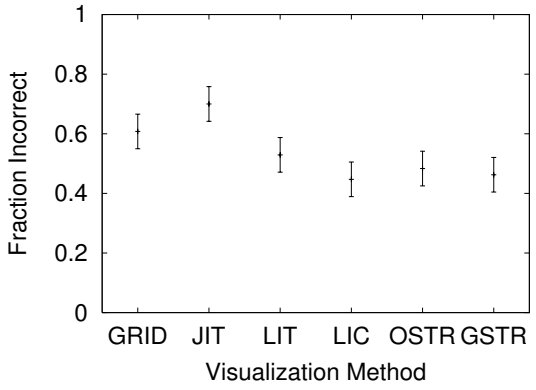

Figure 8: Fraction of images with incorrect number of critical points identified.

5.2 Locating Critical Points

Fig. 8 shows the fraction of images with an incorrect number of critical points identified. GRID and JIT are generally less accurate by this measure, but it is notable how inaccurate all of the methods are. We postulate that this may be because critical points near the edge of a visualization are difficult to identify as inside or outside. A critical point slightly outside the image may be identified as inside, or vice versa. In some ways, this task is also an aggregate of several location tasks, which makes it subject to more error.

A second error measure for this task is the distance from the chosen critical points to the actual critical points (see Fig. 9). Statistics for this distance were calculated on the log of the distance as a normalizing transform. Statistics in the top graph were calculated for cases where the number of critical points chosen matched the number of critical points present in the image. The lower graph shows statistics for all cases, with critical points matched to chosen points as closely as possible. In both cases a least-squares fit was used to find the closest critical points.

For both sets of statistics, GRID and JIT were less accurate. The other methods were comparable.

Fig. 10 shows performance times for the six methods; statistics were calculated on the log of the time due to the observed floor effect and were taken over all images. LIC and LIT are relatively fast, with GSTR a bit slower for this task. The timing statistics calculated using only images with the correct number of critical points identified were not statistically significant.

For this task, LIT and LIC appear to be fast and accurate relative to the other methods; GSTR is a bit slower, but relatively accurate.

5.3 Identifying Critical Point Type

Statistics on the fraction of critical points identified incorrectly are shown in Fig. 11. As might be expected, the methods that show some continuity (LIT, OSTR, and GSTR) work somewhat better than those that do not. Once again, error rates for LIC are likely higher because it does not show the sign of the vector.

GSTR was relatively fast, with OSTR and LIC next, as shown in Fig. 12.

For this task, GSTR is both quick and accurate, relative to the other methods, with OSTR and LIT second and third, respectively.

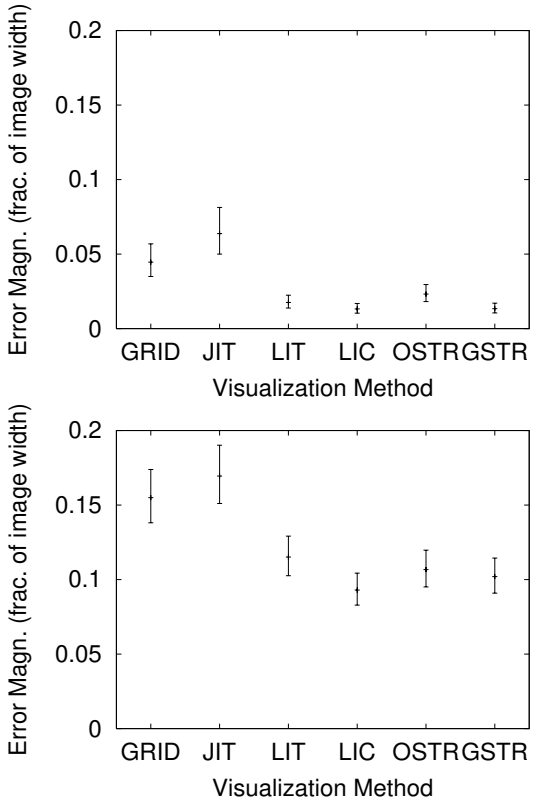

Figure 9: Geometric mean error magnitude for locating critical points task. Error magnitude is the distance between a user-chosen point and the nearest critical point. The top graph shows only cases where the chosen number of points matched the actual number of critical points. The bottom graph shows all cases; as many chosen points as possible were matched with critical points.

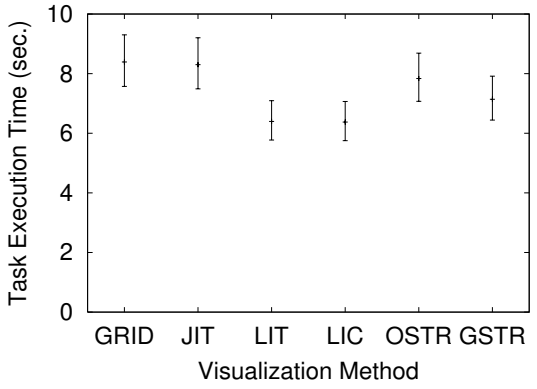

Figure 10: Geometric mean time to count and locate critical points. This computation was done over all images, not just those answered correctly.

5.4 Analysis Details

Statistics were computed using all the results (after any transformation such as the logarithm) of a given user for a given visualization type and task. The base F degrees of freedom were 5 and 55; the Geisser-Greenhouse correction, also known as Box $\hat{\varepsilon}$, was then applied [12]. Statistics are presented in Table 1.

We removed 5 observations from the analysis because it appeared that users mistakenly clicked multiple times. This conjecture is supported by the fact that the cases occurred at the beginning of a group of images for a task/method pair, where there was a pause for images to load. For the counting task, two users each had two successive images with a 0 response time recorded and no critical points selected (for LIC and GRID); in the type identification task one user had one image with a 0 response time (for LIC), Those 5 observations were dropped from all subsequent analysis.

For the critical point location task, the number of critical points ranged from 1 to 4 with a median of 3 and mean of 2.683. For the advection task angular error, Batschelet [13] states that circular statistics are unnecessary for angular differences if the sign of the angular difference does not matter, and therefore standard linear statistics were used on the absolute value of the angular error.

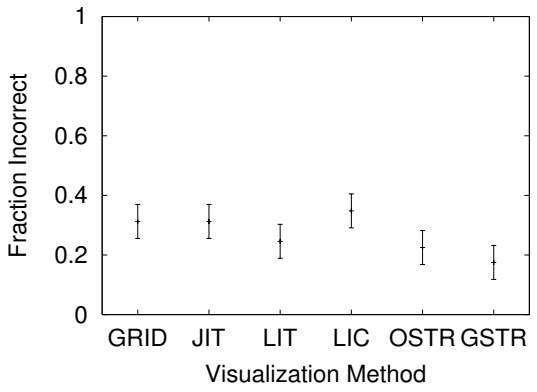

Figure 11: Fraction of critical points with type misidentified.

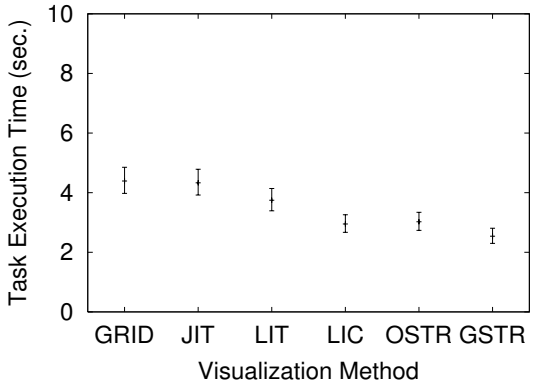

Figure 12: Geometric mean time to identify critical point type.

5.5 Normalizing Visualization Methods

We attempted to normalize the visualization methods by setting their parameters to optimal values for our tasks. However, the normalization might have been done more rigorously. With different tuning of the methods, results might have been different. We did attempt to balance the parameters to perform as well on all of the tasks as possible. A more formal study for each method could have more objectively measured performance of users with each method

Advection $\ln|\theta_{\text{err}}|$: $F_{2.35,25.8} = 43.5$, $p = 1.73 \times 10^{-9}$
Advection, no LIC $\ln|\theta_{\text{err}}|$: $F_{3.55,39.0} = 20.5$, $p = 5.52 \times 10^{-7}$
Advection ln time: $F_{2.96,32.6} = 4.55$, $p = 0.00924$
Locating (correct n) ln distance: $F_{2.68,29.5} = 14.4$, $p = 1.04 \times 10^{-5}$
Locating (all) ln distance: $F_{3.33,36.6} = 8.79$, $p = 0.000103$
Locating (correct n) ln time: $F_{2.72,29.9} = 2.09$, $p = 0.128$
Locating (all) ln time: $F_{3.31,36.4} = 2.90$, $p = 0.0437$
Locating fraction errors: $F_{2.98,32.8} = 5.71$, $p = 0.00298$
Type ID ln time: $F_{2.71,29.8} = 10.1$, $p = 0.000141$
Type ID fraction errors: $F_{3.01,33.1} = 2.62$, $p = 0.0669$

Table 1: Statistics computed for the various comparisons.

using different parameter settings and perhaps chosen the parameters more objectively.

We considered an alternative normalization as well: creating images that had a comparable "density." However, we were not able to define density for all methods and also found that the optimal parameter settings for different methods produced images with densities that were quite different. As a simple example, the GRID and the JIT methods were very similar, and yet the optimal number of icons different by 45%. Given the difficulties in specifying this approach, we opted for the optimal parameter setting described above.

5.6 Subject Pool

Our choice of naive subjects presents another issue. Visualization tools are typically used by experts. Experts have different preferences, abilities, and styles, and these will all influence their performance. Further testing using experts would likely provide more insight into the relative strengths and weaknesses of the different methods.

5.7 Data

Our randomly constructed 2D vector datasets were not drawn from experimental or computational fluid flow examples. However, they are 2D vector fields, well sampled, with good continuity, and with moderate numbers of critical points. They also contain the different types of critical points in the ratios one would expect for fluid flow data. Fluid researchers found their appearance representative. While an alternative construction or source of data might be interesting to test, particularly if it was representative of incompressible flow or some other constrained type of vector field, we felt that the construction we used was a reasonable compromise.

6 Conclusions

Conclusions are difficult to draw from a study like for a number of reasons. First, the differences between some methods on some tasks were not statistically significant. Second, the causes of significant performance differences are difficult to identify conclusively. Further, the results are limited to the specific simple tasks tested, making them difficult to generalize. Our original hypothesis about the distribution of arrows remains unevaluated – GRID and JIT performed indistinguishably, except for critical point locating, where one was a little faster and the other a little more accurate. The two arrow-based methods, however, did not perform well against the other more-continuous methods.

Assuming roughly equal importance for all three tasks both in accuracy and timing, GSTR was the best overall performer. Accuracy was relatively high for all tasks, and performance time low for all but the critical point location task, where it was in the middle of the pack. OSTR and LIT could probably be classed as second and third, respectively. All three had roughly equivalent accuracy. LIT was slower for advection and typing than GSTR, although it was faster for locating critical points. OSTR was slower for locating and typing critical points, but slightly faster for advection.

The good performance of GSTR on these tasks is interesting because it consists of integral curves seeded on a regular grid. While some sources suggest that that seeding will introduce biases into the visualization, those biases don't seem to have hindered performance in this case. Perhaps the fact that the streamlines are significantly longer than the grid spacing hides a bias that might otherwise be introduced.

While the performance of specific methods is interesting, it is perhaps more valuable to look at common aspects of the methods that may explain good performance. Using such information, it may be possible to modify or combine the methods to increase performance. Several factors seemed to be correlated with methods that performed well. First, methods that had some visual representation for integral curves support better performance on all tasks. These included OSTR, GSTR, LIC, and LIT. Second, methods that showed the sign of the vector permitted better performance on the advection and critical point type identification tasks. LIC was the only method that did not clearly indicate vector sign. Third, methods that had some visual indication of critical point location performed better on the critical point location task. These included LIC, where critical points were indicated by the anomalous structure of the flow near the critical points, LIT, where the area of the wedge shapes shrink to leave clear blank regions around critical points, and GSTR, where the overlapping streamlines tend to cluster near the critical points.

In summary, we have presented comparative performance results for three 2D vector visualization tasks using six visualization methods. The choice of tasks clearly influences the results. The tasks seem representative of the types of tasks that fluids researchers want to be able to perform from visualizations, although they could clearly be augmented. Our results show differences among the methods and suggest that the methods that show directionality, indicate critical points, and attempt to visually represent integral curves support better performance.

7 Acknowledgments

Thanks to J. J. Braider for his helpful comments on the paper. This work was partially supported by NSF (CCR-0086065). Opinions expressed in this paper are those of the authors and do not necessarily reflect the opinions of the National Science Foundation.

References

[1] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut, 1983.

[2] William S. Cleveland. *The Elements of Graphing Data*. Wadsworth, 1985.

[3] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, New York, 2000.

[4] Inc. Amtec Engineering. *Tecplot*. Amtec Engineering, Inc., Bellevue, WA 98005, 1988-1998.

[5] Mark A. Z. Dippé and Erling Henry Wold. Antialiasing through stochastic sampling. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 69–78, July 1985.

[6] R. Michael Kirby, H. Marmanis, and David H. Laidlaw. Visualizing multivalued data from 2d incompressible flows using concepts from painting. In *Proceedings Visualization '99*. IEEE Computer Society Press, 1999.

[7] Brian Cabral and Leith (Casey) Leedom. Imaging vector fields using line integral convolution. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 263–272, August 1993.

[8] Greg Turk and David Banks. Image-guided streamline placement. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 453–460. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[9] David J. Field, Anthony Hayes, and Robert T. Hess. Contour integration by the human visual system: Evidence for a local 'association field'. *Vision Research*, 33(2):173–193, 1993.

[10] Inc. Mathworks. *Matlab*. Mathworks, Inc., Natick, MA, 1999.

[11] A. Globus, C. Levit, and T. Lasinski. A tool for visualizing the topology of three-dimensional vector fields. In *Visualization '91*, pages 33–40, 1991.

[12] Scott E. Maxwell and Harold D. Delaney. *Designing Experiments and Analyzing Data: A Model Comparison Perspective*. Wadsworth Publishing Company, Belmont, CA, 1990.

[13] Edward Batschelet. *Circular Statistics in Biology*. Academic Press, 1981.