Interactive Exploration and Tracking of Ensemble Viscous Fingers

Andrew Burks* Cassiano Sugiyama† Timothy Luciani* Jonathan Komperda* G. Elisabeta Marai*

Electronic Visualization Lab
University of Illinois at Chicago

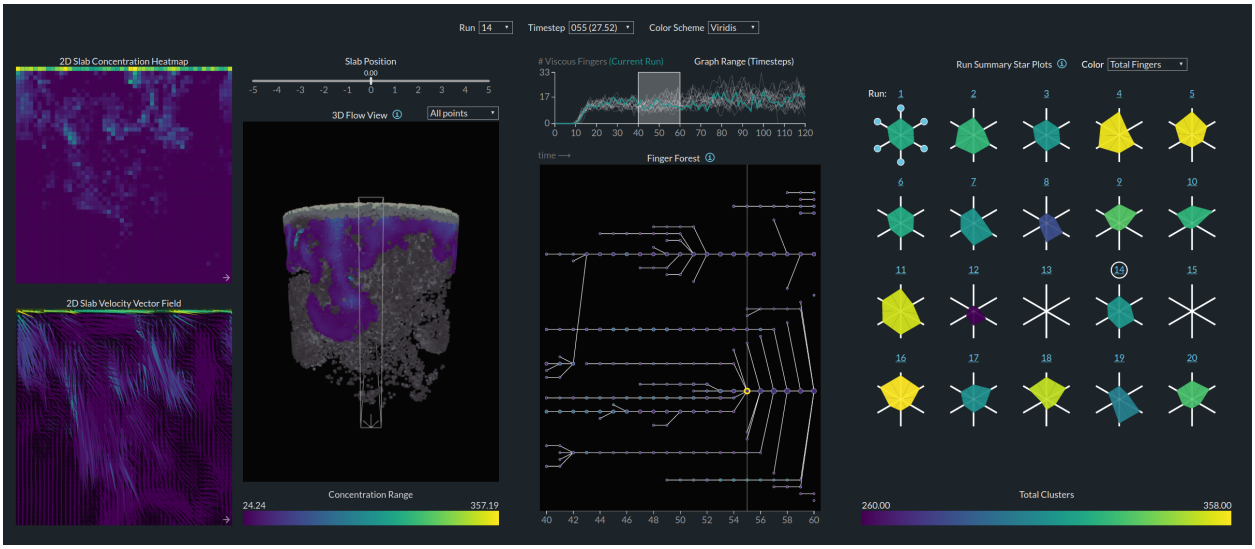

Figure 1: Our application supports the interactive, web-based exploration of multi-run ensemble simulations, through the use of three linked panels: 1) a flow panel comprising two 2D slabs and a 3D flow view (left), 2) a finger panel comprising a run-tracking time chart and a viscous finger forest (center), and 3) an ensemble panel comprising a small multiple of Kiviat diagrams (right). Linked interaction and a computational back-bone allow users to identify fingers and track their evolution over time, and to analyze the data at multiple levels of detail: ensemble, fingers, and particles.

Abstract

We present a condensed version of our submission to the SciVis Data Contest 2016. Our submission was a web-based tool for the visual analysis of multi-run fluid flow simulation data. The tool enables the summarization and comparison of ensemble simulation runs. The back-end implements an algorithm for the identification and tracking of features (viscous fingers and their properties) in transient, fluid flow simulations. The front-end allows users to explore 3D simulation run data using linked 3D views and vertical slabs for specific timesteps, a merge-tree view linked to these spatial representations, and Kiviat diagrams for exploring run properties. We evaluate the tool with a computational fluid dynamics expert and report his feedback.

Index Terms: K.6.1 [Ensemble Simulation Visualization]: Computational Fluid Flow—Feature Extraction and Tracking

*e-mail: {aburks3,tlucia2,jonk,gmarai} @uic.edu
†e-mail: c.sugiyama1@gmail.com

1 Introduction

This work presents an open source, web-based tool to provide interactive visualization of a Finite Pointset Method (FPM) simulation that consists of a cylinder filled with water with an infinite salt source at the top. The tool enables the analysis of the simulation at both low and high levels of abstraction by allowing the user to browse the data in each timestep, inspect properties of each run over time and view summaries of entire runs.

2 Methods

2.1 Task 1: Visualization and Browsing of Point Data

How do you achieve (near-)interactive visualization and browsing of the point data? How do you represent points? How do you provide the context of the simulation domain? What steps do you take to address cluttering issues?

We represent points as OpenGL vertices in a web-based 3D Flow View which uses a color scheme adapted from Matplotlib colormaps [5].

A 3D vertical slab (2D cutting plane with depth) can be used to carve out a subset of points for in-depth analysis to alleviate cluttering. Further analysis of the slab through a linked panel shows the concentration heat map and velocity vector field of the data it contains.

2.2 Task 2: Visualization of Viscous Fingers

Which approach do you use to identify and visually represent the viscous fingers? Does your approach offer (near-)interactive visualization? What forms of pre-processing are required before visualization?

We implement a back-end module to enable finger identification (Task 2), tracking (Task 3), and summarization (Task 3). To identify features within the data using the definition of a viscous finger (a contiguous area of high concentration), a custom clustering algorithm was run on the data.

Moving the slab onto the viscous finger allows the user to view the concentration heatmap and velocity vector field of the slab containing the finger. Selecting a particular tree node in the Finger Forest (described below) highlights the finger in color.

2.3 Task 3: Evolution of Viscous Fingers

How do you quantify and visualize finger properties across time within an ensemble member? How do you represent the evolution across time of these properties? How do you allow to interactively focus the visualization on fingers with particular properties?

The output of the clustering algorithm is fed into a custom, two-stage feature tracking algorithm.

To assign to each cluster a unique ID and also to find the ID of the cluster in the next timestep, several properties of each cluster are calculated and used. The first property is an average position of the points within the cluster, weighted by the concentration of the points. Second, we calculate the average velocity of the points in the cluster, also weighted by the concentration of the points.

In addition to animation, Finger evolution is captured and represented through our Finger Forest visualization (middle panel in teaser figure). The Finger Forest view contains one horizontal, time-aligned tree for each finger in the run, as well as a Time Chart view of the finger count over time for each run. The trees may merge or split according to the finger evolution over time. Each node in a tree represents a finger at that timestep. Finally, we color and size the nodes by the average concentration and number of the points in the finger, respectively. We perform a tree balancing operation to minimize edge-crossings.

A Time Chart can be used to select the range of timesteps which are then graphed in the Finger Forest.

2.4 Task 4: Ensemble summarization

How do you summarize and visualize the temporal evolution of properties of the viscous fingers across the ensemble? How do you allow to interactively use the summary visualization to focus on ensemble members (and possibly even individual fingers) with particular properties?

The Time Chart of the Finger Forest shows the temporal evolution of the unique viscous finger count. The Kiviat diagrams [2] summarize other properties of each run and enable comparison through the shape and color of each diagram. We compute six properties of the runs at the initial launch of the application, one of which is a property of the run as a whole, and five more properties for each timestep, and averaged over the duration of the run

Details on demand (pop-up box) show how each property was computed. Through our run summarization Kiviat diagrams, these properties can be compared between ensemble members. The user may then look at that particular run, and, using the Finger Forest, can examine specific fingers within an individual run at a specific timestep.

2.5 Task 5: Tying Everything Together

How do you design the visualization interface to accommodate the different analysis tasks? How interactive is your system? What is the largest ensemble size that you have successfully applied your visualization to, and what are the current barriers to moving to larger ensembles?

Our top-level design uses a linked-view paradigm to support multiple possible workflows through the ensemble data. The 3D view, the two slab views, the finger forest view, the runtime chart, and the ensemble view are tied together through brushing, linking and filtering. A typical analysis may start by 1) selecting and loading a specific run from the ensemble (top control boxes at interface top) and then exploring the finger data for that run; by 2) browsing the ensemble summaries (star plot view), selecting a run, and then exploring the finger data for that run; or by 3) browsing the ensemble finger count over time (time chart view), selecting a run, then exploring the finger data for that run.

To implement this web-based application, we used the d3.js [1] and three.js [3] Javascript libraries.

3 EVALUATION, DISCUSSION AND CONCLUSION

We developed the application using a parallel prototyping [4] approach, which included 1) exploring encodings, 2) evaluation with a computational fluid dynamics (CFD) expert, and 3) winnowing a variety of measures as well as encodings (including parallel coordinate plots and scatterplots).

The CFD expert evaluation shows that the tool can track viscous finger behavior over time. By analyzing Finger Forest clusters and highlighting them in the 3D view, it is possible to identify fingers generated by boundary conditions and observe how they merge with existing fingers at a given timestep then disperse later on. The vertical slab views proved to be visually engaging and useful to explore finger subsets of points from the simulation. The Kiviat diagrams allow for the summarization and comparison of simulation runs. The Time Chart shows a specific run in the context of other runs, captures the trend in the number of fingers in each timestep over the course of the run, and also serves as a control allowing the user to focus the Finger Forest on a specific time window.

In conclusion, we introduced a novel tool to visualize and analyze concentration transport ensemble simulation data. Users can interactively explore the point cloud data, track viscous fingers and their properties over time, and compare properties between multiple runs within the ensemble to find similarities between multiple runs. The concept of a web-based analysis tool which allows any user to explore sets of data interactively may be applied to other areas of engineering and science.

ACKNOWLEDGEMENTS

This publication is based on work supported in part by the National Science Foundation (NSF), awards NSF CBET-1250171, NSF DMS-1557559 and NSF IIS-0952720-CAREER. We thank the NSF REU program for supporting Andrew, the Brazilian Scientific Mobility Program for supporting Cassiano's summer at EVL, and the NSF Graduate Research Fellows program for supporting Tim. We thank the SciVis 2016 contest organizers and the San Diego Supercomputing Center for sharing their datasets. We further thank our colleagues at EVL for their support and help.

REFERENCES

[1] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

[2] J. M. Chambers. *Graphical methods for data analysis*. 1983.

[3] B. Danchilla. Three. js framework. In *Beginning WebGL for HTML5*, pages 173–203. Springer, 2012.

[4] S. P. Dow, A. Glassco, J. Kass, M. Schwarz, D. L. Schwartz, and S. R. Klemmer. Parallel prototyping leads to better design results, more divergence, and increased self-efficacy. In *Design Thinking Research*, pages 127–153. Springer, 2012.

[5] Matplotlib. Matplotlib colormaps.