Viscous Fingers: A topological Visual Analytic Approach

Garrett Aldrich*
Univeristy of California
Davis, USA
Los Alamos National Lab
Los Alamos, USA

Jonas Lukasczyk†
University of Kaiserslautern
Germany

Michael Steptoe‡
Arizona State University
Tempe, U.S.A.

Ross Maciejewski§
Arizona State University
Tempe, U.S.A.

Heike Leitte¶
University of Kaiserslautern
Germany

Bernd Hamann‖
University of California
Davis, U.S.A.

ABSTRACT

We present a web-based visual analytics framework to visualize viscous fingers that combines particle systems, direct volume rendering, and graphs. In our first step, we apply a Gaussian filter to the particle dataset to obtain a voxel representation for the entire computational domain. Next, we extract and track the viscous fingers using Reeb graphs, where the fingers are treated as level sets above a user-specified salt concentration. We provide linked views to compare, browse, and analyze the ensembles in real time. The major contribution of this technique is an interactive graph which visualizes the evolution of the fingers over time, i.e., when fingers are formed, split, merge, and dissolve.

1 INTRODUCTION

The goal of the proposed technique is to visualize viscous fingers which are areas within the can volume with increased salt concentration. Since it is not obvious how to choose a proper threshold concentration value, our approach is based on the method proposed by Lukasczyk et al. [2] which enables users to vary this threshold to extract different subsets (the fingers) which exceed a defined concentration value. The advantage of varying this threshold is that the users can effectively peal through the layers implied by the concentration values and explore the dataset from different points of view. For instance, a high threshold filters insignificant fingers while a low threshold also detects fingers with slightly increased salt concentration levels. We track the extracted subsets and visualize their evolution and computed statistics through graphs.

Specifically, our web-based framework consists of two main windows: the cinema view (Figure 1) and the detail view (Figure 2). The user starts in the cinema view which provides overviews across ensembles and enables users to find interesting datasets by performing queries. Specifically, users can search for fingers above a given salt concentration, their number, volume, bounding boxes, integrals, and lifespans. The browser provides a table of pre-rendered images of relevant datasets and time steps following the cinema concept [1]. The images can be linked, so that changing the camera in one picture also updates the other images. If users want to explore datasets in detail, they will be able to double-click images to open up the detail view in a new browser-tab. The detail view consists of two elements: the 3D rendering (Figure 5) and an interactive graph (Figure 8). The 3D rendering can show the detected viscous fingers as iso-surfaces of the volume data (5a), the raw data rendered as a particle system (Figure 5b), and the *spatial Reeb graph* of the fingers (5b). The major contribution of this technique is the interactive graph which visualizes the evolution of the fingers over time, i.e., when fingers are formed, split, merge, or dissolve.

2 DATA PREPROCESSING

To identify structure from the particle data set generated by the simulation we apply a preprocessing step to estimate the density distribution for salt over the domain. Each particle in the simulation represents the local salt concentration at that point in the domain. There are several options for estimating the density distribution from particle data, however, choosing an optimal method would require more input from domain experts. Given the constraints of the challenge, we choose a common method for estimating the density that adequately demonstrates the effectiveness of our method. We first represent the domain with a regular voxel grid and average the salt concentration in each cell of the grid by averaging the salt density values of all particles in the cell. To deal with gridding artifacts and reduce high-frequency noise in the density estimate, we apply low-pass filtering using a Gaussian kernel. The choices made in both the number of grid cells used when decomposing the domain, and the amount of smoothing affect the results of our analysis. In a general sense, a higher-resolution gridding and lower level of smoothing allows us to identify small-detail structures, at the cost of an increase in noise which affects the tracking of structures over time. Instead of choosing single values, we vary these parameters (32-256 for the grid resolution and 0.5-3.0 for the Gaussian kernel) for several datasets and include them in the ensemble analysis. This allows the domain expert to compare and contrast the effects of these parameters. We obtained the best quantitative results using a mesh resolution between 64 and 128 grid cells (Figure 9), and a smoothing kernel between 1.0 and 2.0 (Figure 10). Input from domain experts and more user studies are needed to evaluate the best parameters to use, but the values we have chosen demonstrate the effectiveness of our topological analysis and capability of extraction of structures representing the viscous fingers. Another advantage of using a voxel representation is that the size of the voxel data does not increase with increasing particle number.

*e-mail: aldrich@lanl.gov
†e-mail: jl@jluk.de
‡e-mail: msteptoe@mainex1.asu.edu
§e-mail: rmacieje@asu.edu
¶e-mail: leitte@cs.uni-kl.de
‖e-mail: hamann@cs.ucdavis.edu

3 Data Analysis

3.1 Identifying Fingers

Figure 3 shows the outline of our detection algorithm. To identify fingers within the voxel data we start by simply filtering voxels which are below the user-specified salt concentration level (Figure 3a). To filter voxels which are considered noise or belong to the salt supply we derive the *spatial Reeb graph* (Figure 3b) with the method proposed by Lukasczyk et al. [2]. This graph represents the evolution of the fingers in vertical direction (z-direction) for one single timestep, i.e., the graph is a topological skeleton of the fingers and as such captures when fingers are formed, dissolve, merge, and break into parts while sinking. We use this graph to decompose the voxel volume in z-direction and subsequently check for each component of the graph whether its associated voxels have to be filtered out. Specifically, components are identified as part of the salt supply when their xy-bounding boxes roughly approximate a square and cover more than half of the can. Valid fingers also have to be at least two grid cells high to filter out small bumps. As shown in Figure 4, although the *spatial Reeb graph* detects bumps at t_0 and t_2, they only get classified as fingers when they become significantly large enough in time steps t_1 and t_4, respectively. Those criteria, however, can easily be adjusted according to the interest of the domain scientists. After filtering noise and the salt supply (Figure 3c), we determine the connected components within the remaining voxels and assign each voxel group a unique, positive integer ID across the entire simulation (Figure 3d). Voxels not belonging to any group have the voxel group ID -1. Furthermore, for each group we store the integral, number of voxels, bounding box, min/max value, and center of mass.

In oder to implement this approach efficiently, the concentration values are represented as a 1D floating-point array and the voxel group ID of each voxel is stored in an additional integer array of the same size, also known as the *level set matrix (LSM)* [2]. Both arrays are passed as *data textures* to a *WebGL shader* which performs direct volume rendering. The shader renders for a given salt concentration the iso-surfaces and colors them according to the *LSM* (Figure 5a). To enhance spatial perception we also render iso-lines in z-direction and an outline of the computational domain. It is also possible to view and superimpose the *spatial Reeb graph* and the underlying particle dataset. The rendering of the particles is interactive, i.e., the analyst can filter particles by salt concentration, adjust the particle sizes, and choose the used color map (Figure 6). Moreover, the analyst can see what particles belong to which extracted finger by color coding them according to the *LSM*. Figure 5 shows the original particle dataset next to the fingers we extracted with our approach. By varying the concentration threshold for the finger detection the analyst is able to examine the impact of said threshold in real time and choose a threshold that seems appropriate to partition the particles into fingers, thus performing real visual analytics. To enhance spatial perception the particle system is also rendered with screen-space ambient occlusion.

The computation of the *LSM*, the direct volume rendering (512^2 pixels), and the rendering of the particles (512^2 pixels for up to one million particles) is done at interactive framerates even on devices with integrated graphic cards. In order to browse through time at interactive framerates the particle datasets (especially files larger in size than 30MB) are pre-loaded asynchronously from a web server by prioritizing files close to the current timestep. If the prototype system is running on a machine which has direct access to the data then the loading time will be negligible. Since we only visualize the positions and concentrations of the particles it would also be possible to further reduce the file size of the original data by removing unused values such as velocity. Furthermore, the rendering of the particles is optional and thus not required to explore the extracted fingers and their evolution. The only resources required for our approach are the voxel volumes and the *spatio-temporal Reeb graphs*, which both can be loaded in real time.

3.2 Tracking Fingers

In order to track fingers over time we extended the method of Lukasczyk et al. [2] to derive the *Spatio-Temporal Reeb Graph* in 4D space. Hence, nodes in the graph represent 3D volumes (the fingers) and nodes of adjacent time steps are connected by an edge iff their associated 3D volumes overlap. Critical nodes within this graph are nodes with either no previous nodes (birth nodes), no successor nodes (death nodes), multiple predecessor nodes (merge nodes), or multiple successor nodes (split nodes). Hence, a birth node represents the first time a finger is being detected and a death node marks the last time step before a finger dissolves. Merge nodes represent the event of multiple fingers joining, while split nodes indicate that fingers break apart.

This graph can be visualized in different ways. We render the *Spatio-Temporal Reeb Graph* as a line chart where the x-axis is time and the y-axis and the line thickness are used to represent some metric of the fingers, e.g., average z-position, number of voxels, integral, or finger length. For instance, Figure 7 shows a graph where the y-axis and line thickness encode average z-position and finger length, respectively. Birth and death nodes are shown as discs, while merge and split nodes are shown as diamonds. For better readability the nodes are labeled with their corresponding time step and each finger has its own color. If fingers merge the new finger will inherit the color of its largest predecessor. If a finger splits the largest successor will inherit the color of its predecessor and the other fingers will be assigned new colors. This makes it easy to follow the history of a finger and links it to the 3D rendering. Although this projection is useful to see trends, e.g., that all fingers sinking down, this graph is highly occluded, especially when the number of tracked fingers increases. Therefore, it is possible to click on an edge to highlight all components which are connected to that edge and blur the rest. We also provide a projection which uses the y-axis to minimize edge crossings in order to clearly show the evolution of fingers over time by reducing occlusion (Figure 8). The user can easily switch between projections and metrics. Furthermore, by clicking on nodes and edges the user can view information about the associated finger.

3.3 Summarizing and Comparing Ensembles

We compare and contrast the ensemble members by leveraging a visualization database framework constructed using the *ParaView Cinema* concept [1]. In this image-based approach, we create a database by analyzing and rendering the dataset with the methods described in the previous sections. Specifically, we store images of the particle dataset and the iso-surfaces for varying camera angles, concentration thresholds, time steps, voxel grid resolutions, and smoothing parameters. In contrast to the WebGL based rendering in the detailed view, the pre-rendered images are generated by ray tracing which allows the use of more advanced lighting models to better reveal structure and depth.

To explore the image database, we implemented a webbased viewer that stores and displays the image database (Figure 1). Each image in the viewer represents a single simulation, analyzed for a fixed concentration threshold. The camera of an image can be rotated by dragging it with the mouse, emulating real time volumetric interaction. The user can also use a slider to advance the current view over time. For comparison purposes, we also allow the user to sync both the view and the simulation time across all images.

The user can browse through the massive image database by using an intuitive query interface. Each image in the database is linked to the parameters used to generate that image and statistical metrics we calculate based on the *spatio-temporal Reeb graph* such as the average density, number, or length of fingers. More meaningful metrics could be described and included by domain experts, however; we chose a few simple ones to demonstrate our methodology. Once the user has discovered a dataset of interest from within the ensemble, they can explore the simulation in the detail view.

4 Conclusion

As a conclusion we directly address the capabilities of our approach relative to the specific tasks to be performed for the contest.

4.1 TASK 1: Visualization of Viscous Fingers: Basic Visualization and Ensemble Browsing

How do you achieve (near-)interactive visualization and browsing of the point data?

- By pre-rendering snapshots which can be browsed within the cinema viewer (Figure 1) as well as asynchronous pre-loading of the particle data for the 3D rendering.

How do you represent points?

- As a particle system where each particle is rendered as a ball where its size and color encode either its salt concentration or its membership according to the *LSM* (Figures 5b and 6).

How do you provide the context of the simulation domain?

- We render the outline and the backside of the can in 3D space (Figure 3).

What steps do you take to address cluttering issues?

- We apply a low-pass filter by performing Gaussian smoothing, thus already filtering noise. Moreover, the user can focus on a specific part of the *Spatio-Temporal Reeb Graph* by clicking on an edge or node, query the cinema database, and peal through the particle system by filtering particles below a salt concentration threshold (Figure 6).

4.2 TASK 2: Visualization of Viscous Fingers

Which approach do you use to identify and visually represent the viscous fingers?

- We represent the can as a voxel volume and fingers are sub-volumes which have a salt concentration above a user-specified threshold. The viscous fingers are either visualized via direct volume rendering (Figure 5a) or by superimposing the membership and the *Spatial Reeb Graph* on the particles by color (Figure 5b).

Does your approach offer (near-)interactive visualization?

- Due to the nature of the cinema approach, the intelligent pre-loading, and the fast rendering we can achieve interactive framerates. Nevertheless, if a huge particle file is requested that is not pre-loaded yet then it will be likely that it takes a few seconds to load.

What forms of pre-processing are required before visualization?

- In the first step we apply a Gaussian filter on the particle data, have to pre-compute the *spatio-temporal Reeb graphs* for each run, and pre-render snapshots for the cinema database.

4.3 TASK 3: Visualization of Viscous Fingers

How do you quantify and visualize finger properties across time within an ensemble member?

- We calculate the *Spatio-Temporal Reeb Graph* (Figures 7 and 8) which visualizes the evolution of fingers and uses the y-axis as well as the line thickness to encode finger metrics such as size, lifespan, number of voxels, average z-position, and integral.

How do you represent the evolution across time of these properties?

- The evolution across time is also visualized through the *Spatio-Temporal Reeb Graph*.

How do you allow to interactively focus the visualization on fingers with particular properties?

- The user is able to click on edges and nodes of the *Spatio-Temporal Reeb Graph* to focus on the evolution of a specific finger. Moreover, the user can query the cinema database to browse through datasets to meet specific search criteria.

4.4 TASK 4: Ensemble Summarization

How do you summarize and visualize the temporal evolution of properties of the viscous fingers across the ensemble?

- The *Spatio-Temporal Reeb Graph* also summarizes the temporal evolution of properties of the fingers by visualizing them as a compact and comprehensible line chart (Figures 7 and 8).

How do you provide a comparative visualization of different point cloud resolutions?

- Analysts can compare overviews of different cloud resolutions in the cinema database and perform a detailed comparison in the detail view.

How do you allow to interactively use the summary visualization to focus on ensemble members (and possibly even individual fingers) with particular properties?

- The cinema view enables searching for ensembles, properties of their members, and even individual fingers by performing queries on the cinema database in real time.

4.5 TASK 5: Tying Everything Together

How do you design the visualization interface to accommodate the different analysis tasks?

- Our prototype provides two views: the cinema (Figure 1) and the detail view (Figure 2). The cinema view provides an overview of ensembles and enables users to browse through the data by performing queries. The detail view is used to examine one simulation where the data is rendered in real time and the user is able to browse through time and different concentration thresholds.

How interactive is your system?

- Except for pre-loading cache misses the system performs at interactive framerates.

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?

- The only barrier is the number of pre-loaded particle datasets when the analyst wishes to directly compare the extracted fingers with the underlying particle data. However, we emphasize that rendering of the particle data is optional. To render the fingers and visualize their properties and evolution it is only necessary to load the voxel data and *Spatio-Temporal Reeb Graph*, which both can be loaded at interactive framerates.

In case you pre-process or resample the data, what are the details of your approach? Which errors may result from this?

- Since we apply a Gaussian filter we automatically filter particle anomalies such as particles with a high salt concentration within a cloud of particles with low salt concentration values. Our approach currently aims to explore the density of the salt concentrations rather than isolated particles. In the future a more sophisticated density estimation should be used to also correct the density estimation at the boundaries of the domain.

References

[1] J. P. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen. An image-based approach to extreme scale in situ visualization and analysis. *SuperComputing 2014*, pages 424–434, 01 2014.

[2] J. Lukasczyk, R. Maciejewski, C. Garth, and H. Hagen. Understanding Hotspots: A Topological Visual Analytics Approach. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '15, pages 36:1–36:10. ACM, 2015.

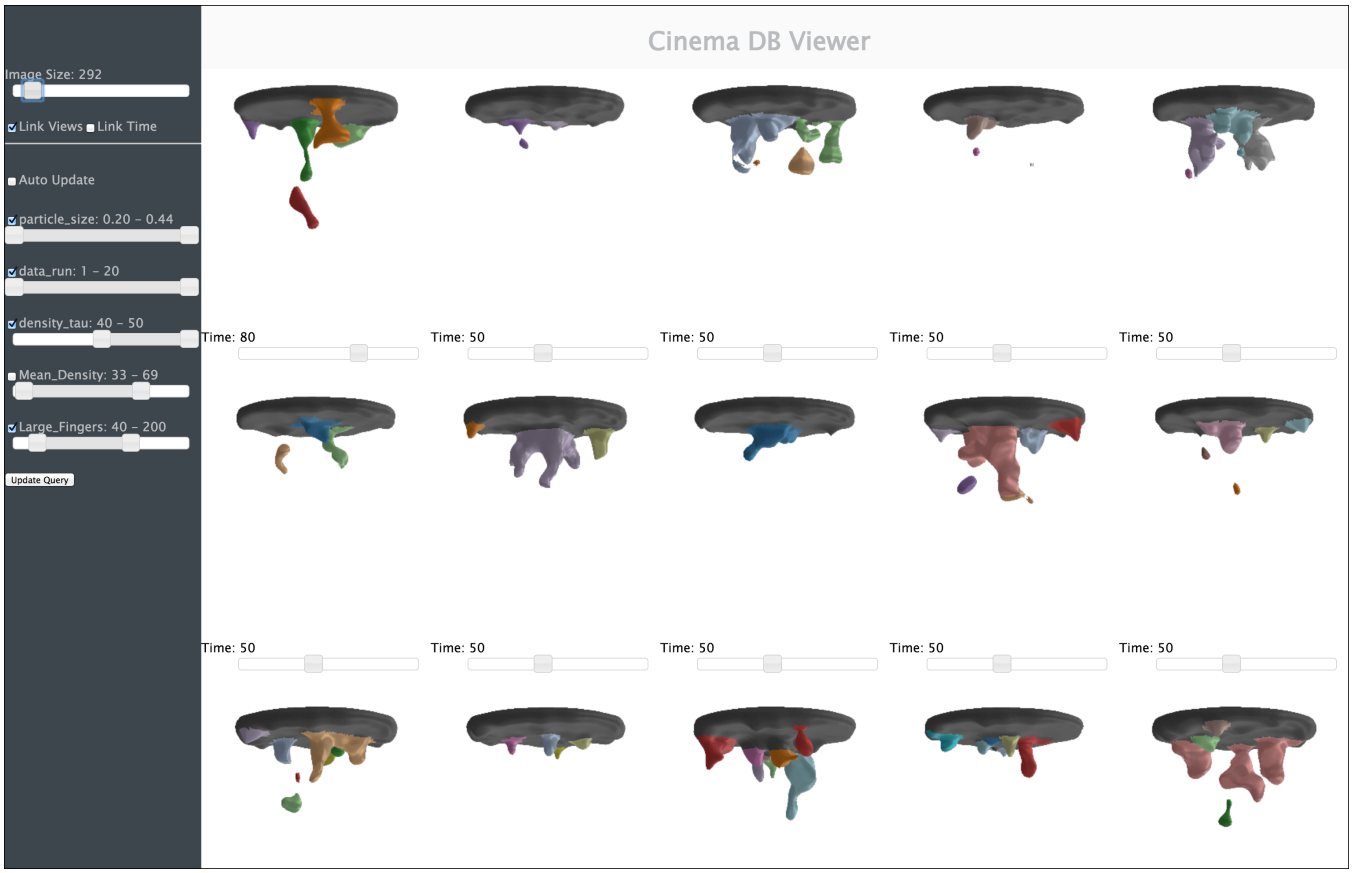


Figure 1: Cinema view showing a set of images meeting the users search criteria.

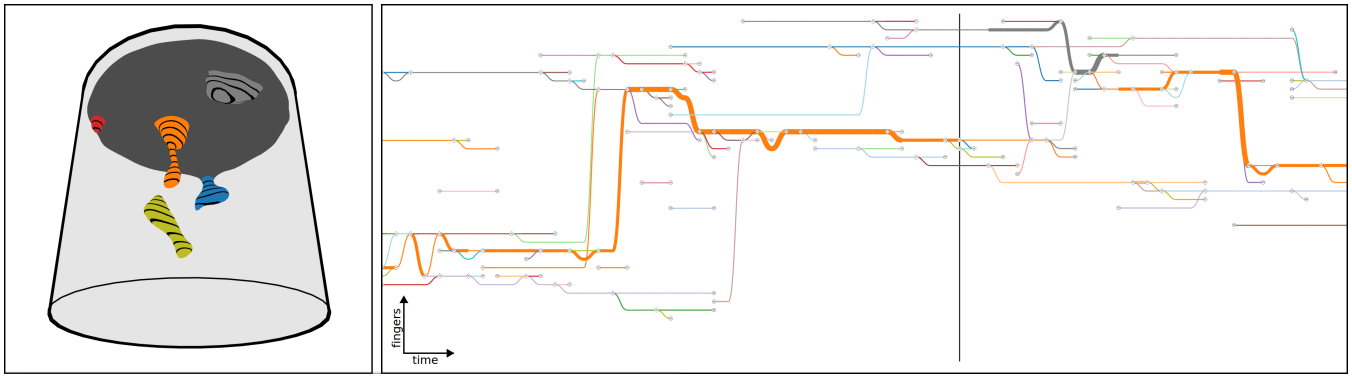
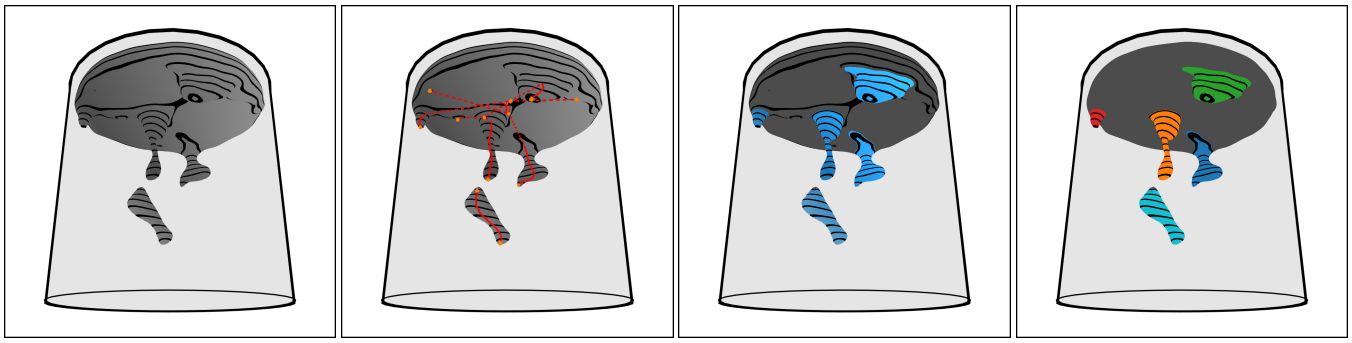
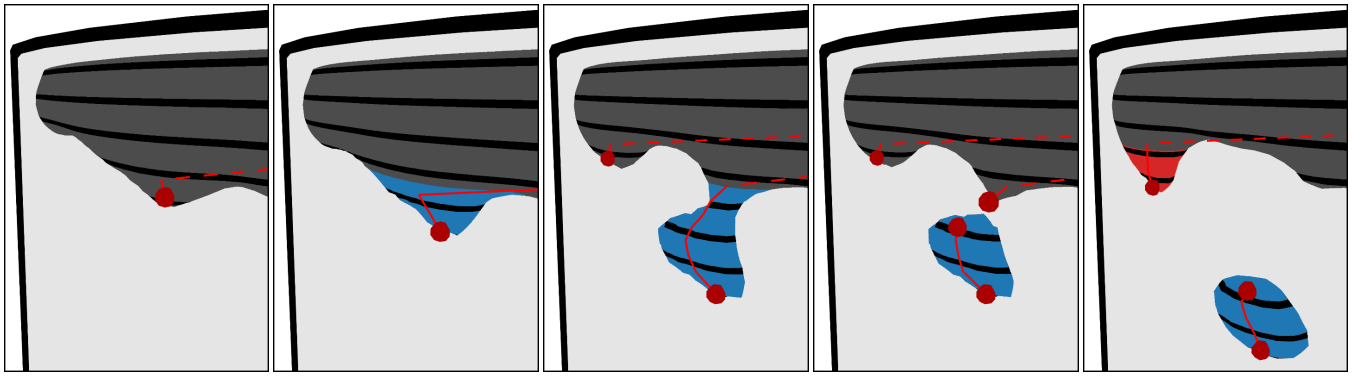


Figure 2: The detail view consisting of the 3D rendering and the *spatio-temporal Reeb graph*.



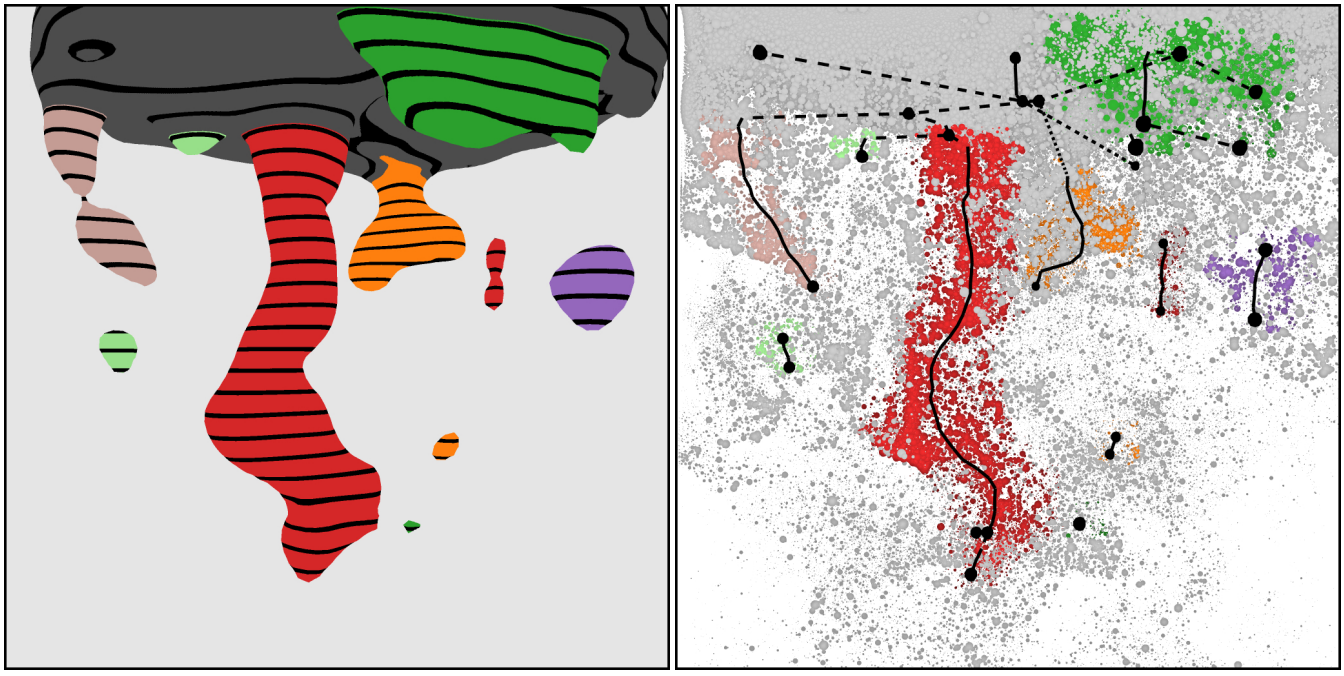
a) b) c) d)

Figure 3: The pipeline of identifying fingers: starting with the voxel volume exceeding the concentration threshold (a), we compute the spatial Reeb graph of said volume (b), then filter the salt supply and noise (c), and finally identify connected components (d).



t_0 t_1 t_2 t_3 t_4

Figure 4: We use information stored in the spatial Reeb graph to filter noise and only identify fingers which are large enough.



a) b)

Figure 5: Extracted finger volumes rendered via direct volume rendering (a), and particle system with superimposed finger membership and *spatial Reeb graph* (b).

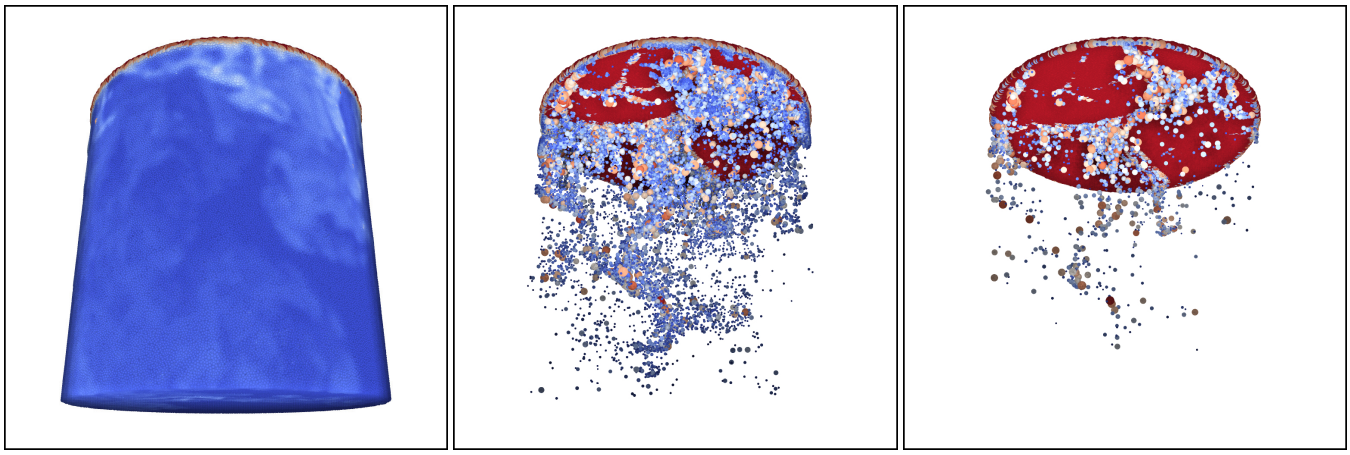


Figure 6: Particle system with 1,900,000 points rendered with a diverging color map filtered by salt concentration values: all particles (left), particles above concentration 100 (middle), and particles above concentration 200 (right).

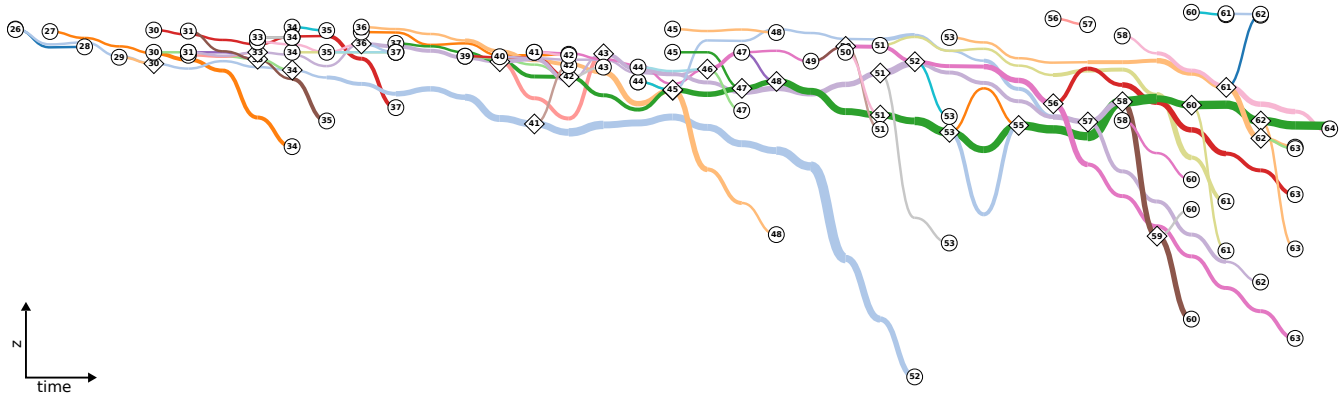


Figure 7: *Spatio-Temporal Reeb Graph* projected as a line chart where the x-axis is time, the y-axis is the average z-position of a finger, and line thickness encodes finger length. This projection is used to explore trends and identify outliers while also visualizing the evolution of fingers, i.e., the time they appear (discs), disappear (discs), merge (diamonds), and split (diamonds). For better readability the nodes are also labeled with their corresponding time step.

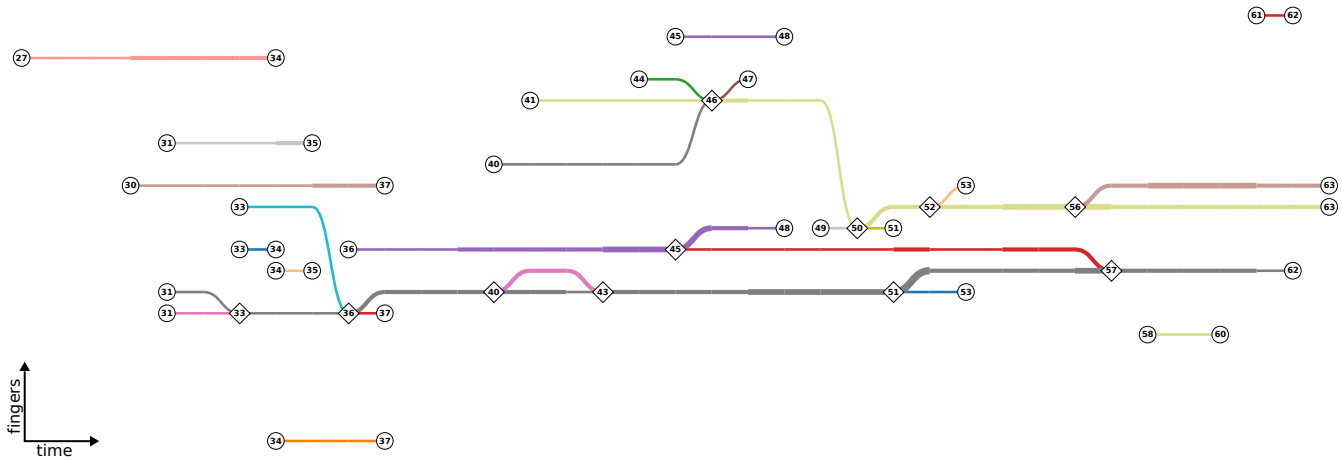


Figure 8: *Spatio-Temporal Reeb Graph* projected as a line chart where the x-axis is time, the y-axis is used to minimize edge crossings, and the line thickness encodes finger length. This projection is used to visualize the evolution of fingers by reducing occlusion.

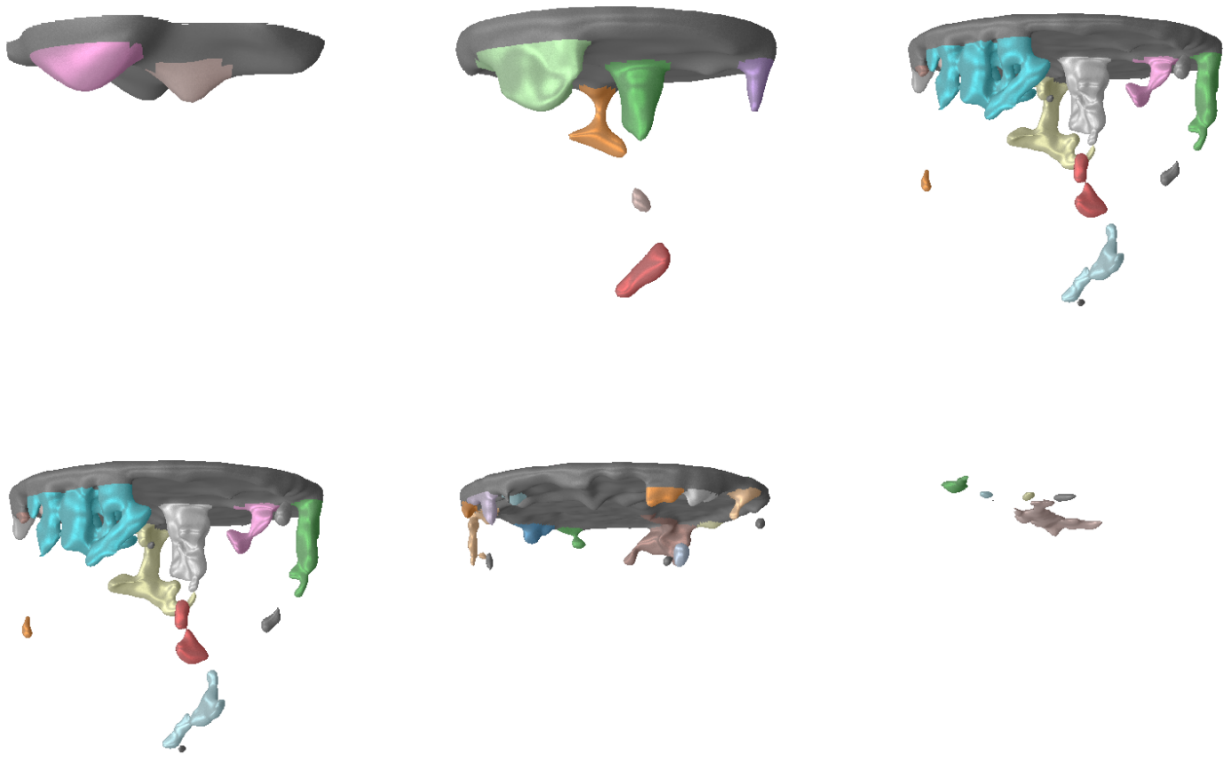


Figure 9: Top row: The effect of choosing different grid sizes for estimating the density distribution. A voxel grid with resolution 32^3 (left), 64^3 (middle), and 128^3 (right) for a single simulation of the .20 particle data ensemble. The higher resolution voxel grid produces smaller, but more detailed structures. Bottom row: The effect of the grid resolution on different particle sizes. A 128^3 resolution is used to estimate the density distribution for a single run from the simulation with .20 particle resolution (left), .30 particle resolution (middle), and .44 particle resolution (right). The largest particles require a smaller mesh resolution as the number of particles is not adequate to estimate the density function in the higher resolution grid.

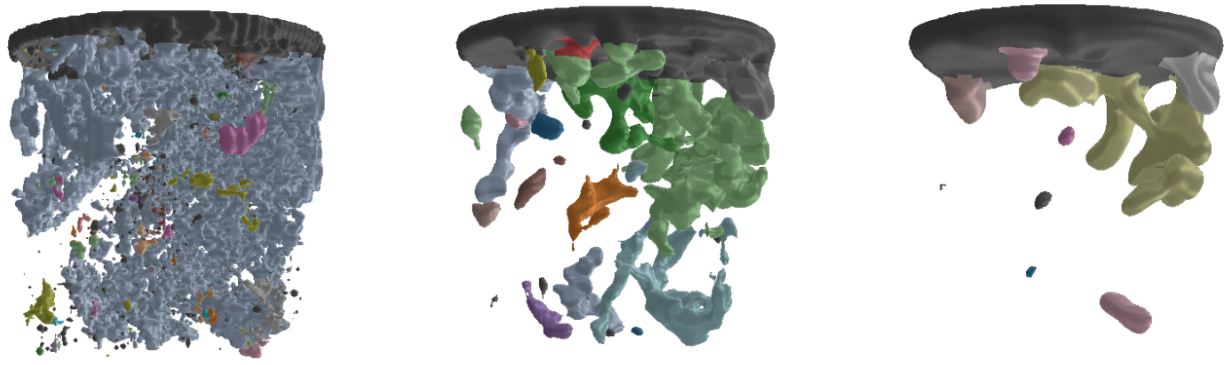


Figure 10: The effect of Gaussian smoothing on the density estimation. In (left) no smoothing is applied, in (middle) a smoothing kernel of size 1.0 is used, and in (right) a smoothing kernel of size 2.0. The low-pass filter is necessary for removing noise, however applying too large of a kernel reduces the structural detail of the resulting fingers.