Visualization Research Center

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Comparative visualization across physical and parameter space

Adrian Zeyfang

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Thomas Ertl
Supervisors:	Alexander Straub, M. Sc. Dr. Steffen Frey Dr. Guido Reina
Commenced:	February 2, 2022
Completed:	September 16, 2022

Abstract

We designed and developed an interactive visualization approach for exploring and comparing image sequences in the context of porous media research. Our tool facilitates the visual analysis of two-dimensional image sequence datasets captured during fluid displacement experiments in a porous micromodel. The images are aggregated into a single graph-based representation, allowing for an experiment to be visualized across its entire temporal domain. This graph is generated from the viscous flow patterns of the invading fluid, reducing the need for manual image masking and clean-up steps. The Node-Link representation of the graph is superimposed onto the raw images, creating a composite spatio-temporal view of the dataset. We demonstrate the functionality of our implementation by evaluating its output and performance on a collection of related datasets. We found that separate experiments in the same porous medium yield topologically different, yet visually similar flow graphs with comparable node positions.

Kurzfassung

Wir haben eine interaktive Visualisierungsmethode für die Exploration und den Vergleich von Bildsequenzen im Kontext der Erforschung poröser Medien entworfen und entwickelt. Unser Werkzeug unterstützt die visuelle Analyse zweidimensionaler Bildsequenzdaten, die aus Experimenten zur Flüssigkeitsverdrängung in porösen Mikromodellen gewonnen wurden. Die Bilder werden in eine einheitliche Graph-basierte Repräsentation aggregiert, die es ermöglicht, ein Experiment über seine gesamte Zeitspanne zu visualisieren. Dieser Graph wird aus den viskosen Strömungsmustern der eindringenden Flüssigkeit erzeugt, wodurch die Notwendigkeit manueller Bildmaskierungs- und Säuberungsschritte reduziert wird. Die Node-Link-Repräsentation dieses Graphen wird mit den Rohbildern überlagert, um eine zusammengesetzte räumliche und zeitliche Ansicht des Datensatzes zu erhalten. Wir demonstrieren die Funktionalität unserer Implementierung, indem wir ihre Ausgabe und ihr Laufzeitverhalten mittels einer Sammlung zusammenhängender Datensätze evaluieren. Wir stellten fest, dass separate Experimente im selben porösen Medium topologisch unterschiedliche, aber visuell ähnliche Flussgraphen mit vergleichbaren Knotenpositionen ergeben.

Contents

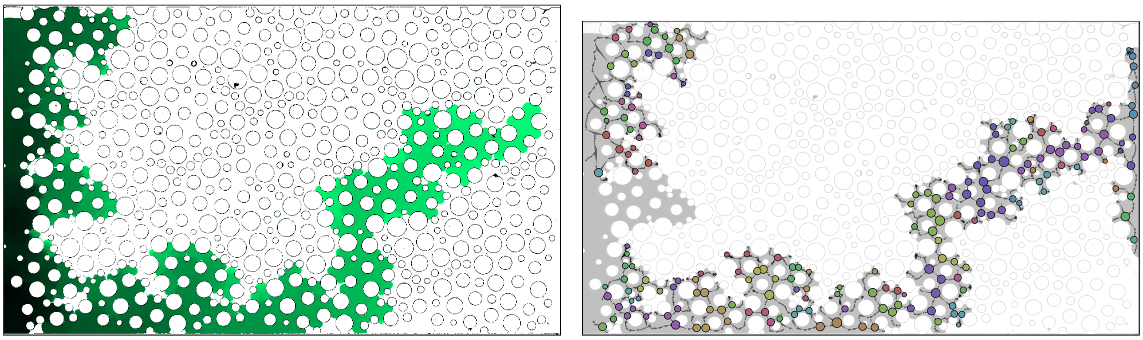
List of Figures

1 Introduction

The field of *Poromechanics* studies the physical properties of *porous media*: solid materials with an interconnected network of narrow voids. The influence of porous media on the behavior of fluids occupying or flowing through their pore space is of particular interest, as saturated porous materials are often encountered across various industrial and scientific fields, such as geology [GDB04], biology [Kac08] and materials science [UCH04].

Modeling the physical processes that lead to observable macroscopic behavior is a primary objective in poromechanics. One such process is *fluid displacement*, in which a fluid flows through an already saturated porous medium, pushing out the existing fluid as it permeates the pores within the material. Despite significant advances in porous media research, many of the underlying processes are not yet fully understood [DiC13] [ASY22].

To gain insights into displacement processes at a microscopic scale, experiments with artificial micromodels are conducted, simulating the conditions of a porous material in a controlled environment while allowing for the direct observation of flow behavior [FSK+21] [WDH+22]. These experiments yield results in the form of image sequences, capturing the evolution of the invading fluid's path taken through the pore network. Tool-assisted interactive visual analysis and comparison of these time-oriented datasets can lead to the identification of patterns and behaviors that emerge under certain conditions.



Temporally aggregated images of a fluid displacement process. Flow timestamps are mapped to a color gradient between dark green (early) and light green (late)

Flow graph superimposed onto snapshot from dataset

Figure 1.1: Visualization of auxiliary dataset A (see section 4.2).

In this work, we describe an approach to interactively visualize datasets derived from fluid displacement experiments. At its core, our technique constructs a flow graph from a sequence of grayscale images, aggregating the dataset over its temporal domain (see figure 1.1). This graph is then used as an additional intermediate data source, making use of superimposition and interactive temporal selection to present an enriched view of the dataset. We develop a full visualization pipeline based on our approach, focusing on scalability to large datasets. This implementation is built as a plug-in for the visualization framework *MegaMol*, taking advantage of its graphical user interface for configuring the visualization pipeline and its parameters as a directed acyclic graph. Finally, we evaluate our approach and its implementation by applying it to a collection of datasets and examining the results.

1.1 Analysis goals

Our primary research goal is the development of a compact visualization method for two-dimensional fluid flow within a porous medium. This visualization method is designed to facilitate answering our main research question within the context of experimental flow analysis: *Do fluid displacement processes in a porous medium exhibit similar flow patterns when occurring under similar viscosity ratios and capillary numbers?*

In order to answer this question, a suitable approach for visualizing and comparing time-dependent datasets in the context of porous media research is designed, implemented and applied. This includes the aggregation of the datasets to allow the full temporal domain to be analyzed, transformation into an intermediate graph-based representation, collection of relevant metrics from the dataset to characterize different types of flow across parameter space, and a series of final mapping and rendering steps to display the combined results across one or more datasets.

As part of our work, a full visualization pipeline is implemented within the MegaMol visualization framework, providing an interactive tool for the visual analysis of image series datasets. The technical foundations laid by the MegaMol plug-in developed within the scope of this work are designed to be reusable for future research tasks on image series. To evaluate our approach and implementation, we use these newly developed tools to visualize multiple datasets collected from fluid displacement experiments in prior studies.

2 Related work

Prior work in the field of porous material visualization commonly focuses on static three-dimensional datasets, applying volume rendering approaches to visualize scanned and generated porous media. The problems addressed by these studies include data acquisition and storage [JBT+18], real-time rendering performance [GRZ+10] and occlusion issues for pore space and solid space geometry [NBK13]. Multiple software applications and frameworks facilitating the visualization of porous media are available, sharing many of the research goals outlined above.

PoreSpy [GKT+19] provides a collection of Python modules for porous media research, each containing various high-level functions for operating on 3D images. It supports the import and generation of porous volume datasets, a range of filters to preprocess and transform datasets, and various aggregation functions to produce metrics, networks and 2D visual representations. Due to its nature as a library, the construction of a visualization pipeline using PoreSpy is achieved by invoking the provided functions from Python code.

Voxie [Kie16] is an interactive application for importing, processing and visualizing volume datasets. It includes a suite of porous media analysis and visualization tools, allowing for datasets to be explored in a three-dimensional iso-surface view, presented as 2D slices, or reduced to a set of metrics generated from the dataset. The visualization pipeline is constructed and configured via a graph-based user interface, with filter backends written primarily in Python and C++. The porous media filters primarily focus on identifying, measuring and visualizing fully isolated *blind pores* embedded in solid objects, which have been scanned using an industrial computer tomography process.

While these software tools incorporate a wide range of functionality for processing three-dimensional single-image volumes, analysis and visualization options for two-dimensional time-oriented datasets depicting fluid flow are more limited. In a prior study, Frey et al. [FSK+21] describe and evaluate a technique for visualizing and comparing multiple flow experiments in a 2D micromodel. By first generating a pore space graph (referred to as a *Transport Network* by the authors) from the experimental setup, then tracking the fluid's behavior by matching it to the graph's nodes, the datasets are mapped to a more readily comparable form, which is then presented visually. Upon evaluating this technique by visually analyzing and comparing the recordings of 11 experimental parameter combinations, the authors find that their approach allows domain experts to gain useful insights into the underlying physical processes within porous media.

3 Fundamental concepts

Several key concepts across the fields of fluid mechanics and information visualization serve as the theoretical foundation for our work.

3.1 Porous media

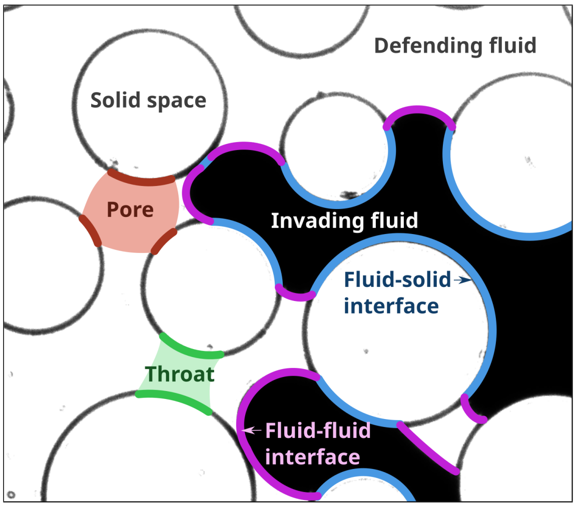


Figure 3.1: Annotated illustration of a cropped section in a porous medium currently undergoing a fluid displacement process.

A *porous medium* is characterized by a combination of solid material and void spaces, forming a composite medium through which fluids can flow. Depending on the geometry of these voids and their interconnections, a range of physical effects can emerge during fluid flow processes.

3.1.1 Terminology

Throughout this work, the following terminology is used in the context of studying porous media and their interactions with contained fluids during displacement processes.

Pores and throats In an interconnected porous medium, *pores* denote larger spaces in the solid material, whereas *throats* are thin, elongated gaps connecting pores together (see figure 3.1). Together, pores and throats form the *pore space* of a porous medium, while the complementary solid material is defined as *solid space*.

Porosity is the overall ratio between the pore space's volume and the porous medium's total volume, given as a dimensionless quantity between 0 and 1. A greater porosity corresponds to a larger number and size of spaces within the solid material.

Displacement When studying the properties of porous materials, a particular area of interest is the behavior of fluids as they permeate through such a medium. Displacement experiments are conducted by first filling the pore space of a porous medium with a wetting ("defending") fluid, followed by the forced external injection of dyed non-wetting ("invading") fluid at a constant rate.

Two-dimensional models of porous media constructed from transparent materials can be experimentally studied in a laboratory setting, allowing the flow behavior of fluids to be observed directly and recorded using a standard optical camera.

The side of the porous medium into which fluid is injected is called the *inlet*, while the opening through which the displaced fluid can escape on the opposite side is called the *outlet*.

Interface The boundary separating two distinct phases of matter is referred to as an *interface*. This can be further classified into the *fluid-fluid interface* which separates the non-wetting and wetting fluid phases within the porous medium and moves throughout the experiment, and the *fluid-solid interface*, which represents the contact surface between the porous medium and the contained fluids.

Viscosity ratio Viscous forces are generated by friction between a fluid's outer layers and solid walls, affecting the inner layers via shear stress and altering the overall flow of fluid [FSK+21]. The susceptibility to such effects is an inherent property of the fluid, and is quantified by its *viscosity*.

In the context of a fluid displacement experiment, the term *viscosity ratio* M refers to a dimensionless quantity defined by the ratio between the injected fluid and the fluid being displaced.

Capillary number Capillary forces are generated by adhesive effects between the fluid and solid phases, causing an increase in flow rate through thin throats. As such, these forces largely depend on the local geometry of the porous medium [FSK+21].

The *capillary number* Ca is defined as the ratio between viscous forces and capillary forces. A lower capillary number indicates flow driven primarily by capillary forces, whereas a higher capillary number indicates the dominance of viscous forces.

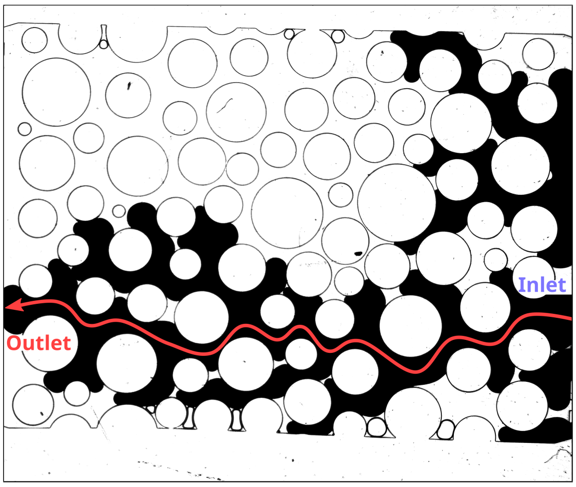


Figure 3.2: Breakthrough occurring during a fluid displacement experiment. The red curve indicates the uninterrupted fluid connection between inlet and outlet. Data source: [FSK+21]

Breakthrough In a fluid displacement experiment, *breakthrough* is reached when an uninterrupted path through the porous medium is established by the injected fluid, connecting the inlet and the outlet (see figure 3.2). Once this connection is established, the observable flow behavior of the injected fluid changes significantly [ASY22].

Due to these differences, this work's scope focuses primarily on *pre-breakthrough* flow behavior, using breakthrough as a reference point for temporally aligning datasets with experiment durations spanning several orders of magnitude [FSK+21].

3.2 Visualization of time-oriented data

When collecting data over a period of time, each point in the dataset is implicitly or explicitly associated with the timestamp of its recording. These timestamps can be saved alongside the data as an additional variable, yielding a *time-oriented dataset*. To visualize these datasets, a suitable visual metaphor should be chosen to represent the temporal dimension. Time-oriented visualization techniques can be classified into *static* and *dynamic* approaches [Koc19].

Static time-oriented visualization techniques map the temporal dimension to a visual variable (often the position of visual primitives [Koc19]), resulting in time-to-space mappings, such as *Stacked Graphs* [BW08] or *Space Time Cubes* [BDA+16]. This allows data from more than one timestep to be displayed at once, making these techniques suitable for providing an overview of a time-oriented dataset across the temporal domain.

Dynamic time-oriented visualization techniques instead treat the temporal dimension as a display parameter, which is used for selecting a temporal slice of the dataset to be visualized. This selection can be shifted automatically via a time-to-time mapping, resulting in an animated view of the dataset that retains the temporal aspect of the original data source. Alternatively, the temporal selection can be manipulated interactively using a timeline control, such as *SmoothScroll* [WE13], which supports layered temporal selections at multiple granularities. Dynamic techniques allow for individual snapshots within the dataset to be viewed independently, freeing up a visual variable that would otherwise represent the temporal dimension. However, they are prone to causing *change blindness*, a cognitive limitation in the ability to identify changes between images that are shown sequentially [SL97].

These techniques can be combined to form an ensemble visualization, using a temporally aggregated overview to allow for selections to be made, and a linked dynamic view to provide details on demand based on interactive user input. *LiveRAC* is an implementation of this hybrid approach, which nests multiple static viewports within a larger interactive overview [MMKN08].

3.3 Graph visualization

When visualizing graphs, the choice of technique depends on graph's characteristics, such as the number of vertices in the graph, the distribution of edges across the graph's vertices and the presence of scalar attributes or labels on vertices or edges [TKE12].

Within the scope of this work, image sequence datasets captured during displacement experiments are transformed into graphs with vertices with positional and scalar attributes and unweighted, directed edges. These graphs are then visualized in a single-dataset or comparative multi-dataset context.

3.3.1 Node-link diagrams

In a *node-link diagram*, vertices in a graph are each mapped to distinct visual elements, such as circles or rectangles, which are then connected to each other using lines or curves to represent edges between pairs of vertices (see figure 3.3).

For graph datasets containing vertices without spatial information, the position of these nodes can be chosen via a graph layout algorithm to highlight the graph's topology and improve the diagram's readability by minimizing visual clutter.

In some graphs, each vertex is inherently associated with a specific position in space, allowing the node-link diagram's mapping to preserve this spatial relationship by letting

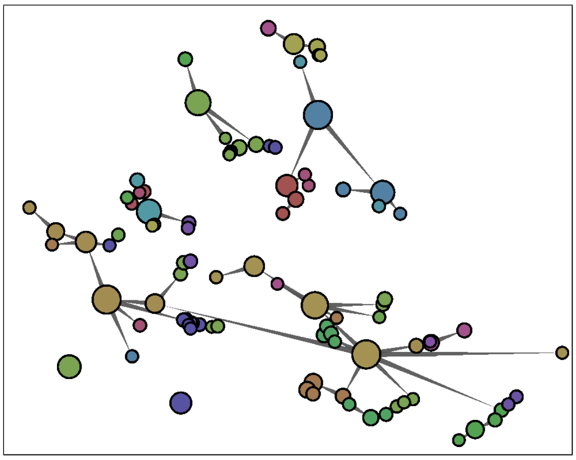


Figure 3.3: Node-Link Diagram representation of a small directed graph, using tapered lines to indicate edge directions. Extract of a flow graph generated from dataset $Ca = 10^{-2}, M = 1$ (see section 4.1).

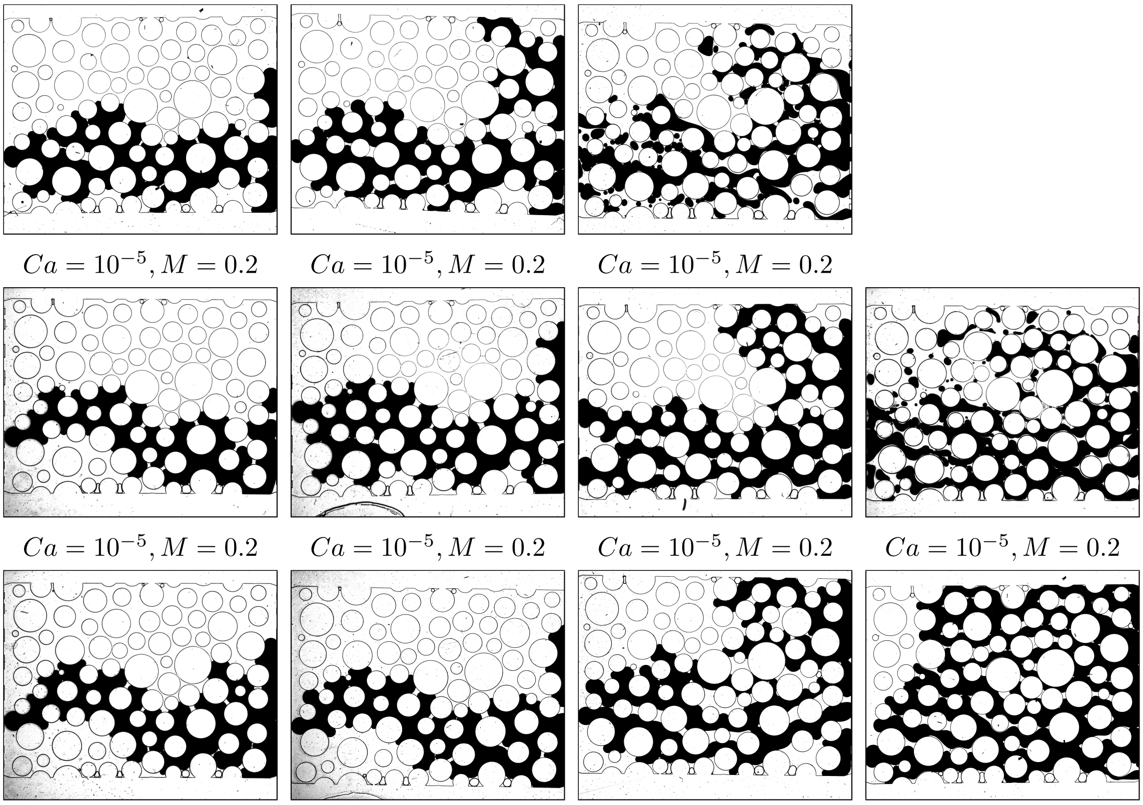
the positional attribute influence or decide the position of the visual primitive representing the vertex.

While the node-link diagram is an intuitive technique for displaying small or medium-sized sparse graphs, its scalability is limited by visual clutter for graphs with very large vertex counts or a high density [BBDW16].

4 Data

Throughout our work, two groups of two-dimensional grayscale image sequence datasets serve as the main data source. Both groups were captured as part of a series of fluid displacement experiments conducted in a micromodel, which serves as an artificial porous medium. Between datasets in each group, the experimental setup and thus the geometry of the porous medium was preserved, while the invading fluid's composition and injection rate are varied between experiments to enable comparisons.

4.1 Fluid displacement experiments



$Ca = 10^{-5}, M = 0.2$ $Ca = 10^{-5}, M = 0.2$ $Ca = 10^{-5}, M = 0.2$

$Ca = 10^{-5}, M = 0.2$ $Ca = 10^{-5}, M = 0.2$ $Ca = 10^{-5}, M = 0.2$ $Ca = 10^{-5}, M = 0.2$

$Ca = 10^{-5}, M = 0.2$ $Ca = 10^{-5}, M = 0.2$ $Ca = 10^{-5}, M = 0.2$ $Ca = 10^{-5}, M = 0.2$

Figure 4.1: Still frames from each experiment at the time of breakthrough [FSK+21]

The primary collection of datasets consists of 11 image sequences, each of which captures the displacement process of one fluid by another in an artificial two-dimensional porous medium (see figure 4.1). These experiments were conducted by Frey et al. as part of a prior study [FSK+21].

The porous medium, a transparent micromodel constructed using soft lithography with Polydimethylsiloxane, consists of a hollow cuboid measuring 20 mm $\times$ 15 mm $\times$ 100 μm, with solid cylinders of varying radii distributed within it. The cylinders cover the entire height of 100 μm, forming a connected network in the model's pore space with an overall porosity of 0.44 and a mean pore size of 410 μm [FSK+21].

For each experiment, the porous medium was filled with Fluorinert FC-43, which was displaced via injection of a mixture of dyed water and glycerol. This displacement process was captured optically, yielding a sequence of grayscale images with uniformly distributed timestamps. Capture rates between 1 and 15 frames per second were chosen for each experiment based on the speed of the process. This process was repeated with different ratios between glycerol and water to achieve viscosity ratios $M \in \{0.2, 1, 10\}$, each of which was tested with different injection rates to achieve capillary numbers $Ca \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$, yielding a total of 12 parameter combinations for Ca and M. The combination $Ca = 10^{-2}, M = 0.2$ was excluded due to technical challenges [FSK+21].

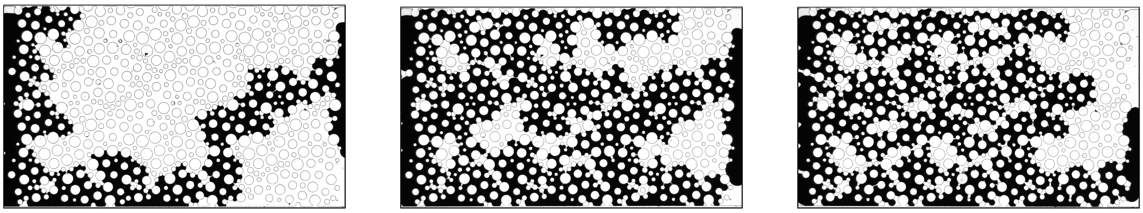
4.1.1 Data format and scale

The datasets are originally provided as uncompressed TIFF files with a single 8-bit grayscale channel, with a resolution of 2448×2050 pixels across all images. The number of images in each dataset ranges from 55 to 2009, for a total of 8122 images occupying 38.5 GB of storage space.

To reduce disk read overhead while visualizing these datasets, the files are converted into 8-bit single-channel PNG images using ImageMagick for batch conversion. The compressed dataset, now totalling 7.9 GB, serves as the main data source for which visualization techniques are designed and implemented in this work.

4.2 Secondary dataset

In an as of yet unpublished follow-up work to a study on permeable porous media by Weinhardt et al. [WDH+22], three additional experiments have been performed, similar in setup to those described in section 4.1. Between these experiments, the flow rate at which the dyed invading fluid is injected varies, while the experimental setup and the viscosity ratio remains the same. Within the context of this work, the resulting datasets are labeled A, B and C, in order of increasing flow rate (see figure 4.2).



Secondary dataset A Secondary dataset B Secondary dataset C

Figure 4.2: Still frames from each of the secondary displacement datasets post-breakthrough (see section 4.2)

Similar to the primary dataset, this secondary collection of image sequences is originally provided as uncompressed single-channel 8-bit TIFF files. Each dataset is captured at a slightly different resolution, varying between 2141×1266 for dataset A, 2171×1271 for dataset A and 2173×1271 for dataset C. These differences in resolution must be taken into account when designing and implementing visualization tools for comparing these datasets. The uncompressed files of this secondary dataset occupy 12.9 GB of disk space. By applying the preprocessing steps described in 4.1.1, the scale was reduced to 9.5 GB of PNG image data.

Unlike the primary experiments, two displacement processes were performed in succession in each trial. After the non-wetting fluid achieved breakthrough, wetting fluid was injected through the same inlet to displace the non-wetting fluid. Within the scope of this work, only the initial displacement process is analyzed, with all image sequences being cut off from the inlet pump's shut down after the initial breakthrough. The final number of images is 312 for dataset A, 564 for dataset B and 326 for dataset C.

5 Visualization

This chapter outlines the design choices for suitable image series visualization techniques, given the characteristics of the datasets described in chapter 4.

5.1 Visualization pipeline architecture

Recording the displacement of one fluid by another in an experimental model of a porous medium produces large amounts of raw image data. Datasets capturing longer experiments can contain several thousand high-resolution images, grouped together as a time-dependent image series.

Visualizing time-dependent data presents a particular challenge due to the limitations in the ability of humans to accurately identify differences and trends across images [SL97]. Animation-based approaches preserve the original temporal mapping inherent to the original experiment, but limit the amount of data shown on-screen at once and fail to offer a complete overview of the experiment's evolution over time. On the other hand, agglomeration-based approaches incur a loss of information during the mapping step. Capturing the temporal component of the image sequence datasets is essential for comparing datasets with similar spatial structures, but differing behaviors over time [Koc19].

5.1.1 Modeling flow as a graph

Depending on the chosen experimental parameters for capillary number and viscosity, and thus the observable behavior of the fluid during the experiment, it is possible to reduce the amount of data required to extract suitable metrics and visualize the dataset across its full temporal domain.

The main approach for dimensionality reduction discussed in this work is based on aggregation. The time-dependent dataset is transformed into a simplified representation that encodes areas of interest at each timestep while omitting invariant data from other areas of the dataset.

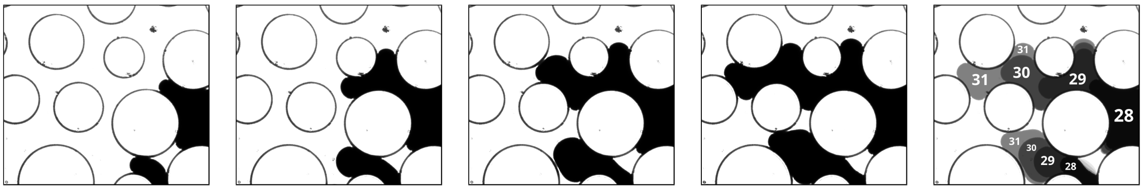
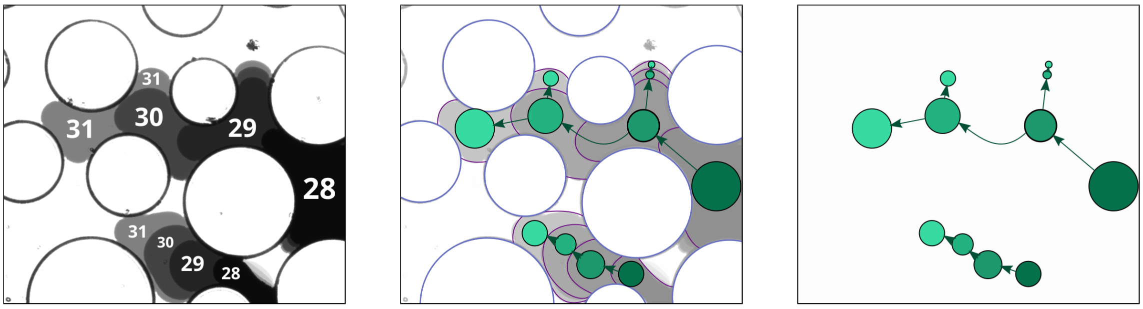


Figure 5.1: Left to right: Four subsequent frames of a cropped region extracted from the experiment performed at $Ca = 10^{-3}, M = 1$ at frames $[28; 31]$.
The rightmost image represents the timestamp map of this dataset, with each pixel now storing at which frame number the fluid-fluid interface first passed through it. Numeric labels have been superimposed for illustrative purposes.

Viscous flow

For experiments performed under predominantly viscous conditions, the non-wetting (invading) fluid's behavior is stable, with several fronts simultaneously displacing the wetting (defending) fluid at a comparable velocity across different regions within the porous medium [RSK20] [YKZS21]. Once the wetting fluid has been displaced from a pore or throat, it is subsequently pushed towards the outlet by the non-wetting fluid, which does not recede significantly once a space has been invaded. As a result, the fluid-fluid interface can be represented as a growing boundary, defined by a two-dimensional mapping from each spatial location in the dataset to the first timestamp at which the wetting fluid is displaced by the non-wetting fluid (see figure 5.1). This assumes that the injected fluid does not recede by a significant amount, which can indeed be observed in the available datasets from experiments performed in the viscous regime.



Partial timestamp map for $Ca = 10^{-3}, M = 1$.

Connected equal-timestamp regions and their neighbors.

Resulting flow graph.

Figure 5.2: Construction of a flow graph from a timestamp map.

To further reduce the dataset into a more compact representation, a graph is constructed from this timestamp-valued aggregate image (see figure 5.2). Each node in this graph represents a connected region of non-wetting fluid displaced at a specific timestep. The framerate at which photographs are taken during the experiment provides a natural means of quantization, allowing regions to be associated with each other by timestamp equality. Edges are established between temporally and spatially adjacent regions, creating a correspondence between chains of nodes in the graph and the path taken by the fluid-fluid interface through a specific throat in the experimental setup.

The set of nodes in the graph with a specific timestep corresponds directly to the active fingers flowing through the porous material at that same timestep.

Capillary flow

When analyzing a fluid displacement experiment in which capillary forces are dominant, the interface between the fluids behaves more chaotically compared to viscous flow [BKL+86] [YKZS21]. Isolated regions of non-wetting fluid permeate through the porous material without necessarily leaving behind a trail [ST58].

As a result, reducing the dataset by mapping each pixel to a single timestamp at which displacement occurred would result in a significant loss of information. Instead, connected regions must be labeled and analyzed for each frame. Regions are identified and tracked across multiple frames by checking for overlapping pixels in previous or subsequent frames.

A graph-based representation of a capillary flow experiment is constructed by first mapping each connected region in each frame to a node. Then, pairs of temporally adjacent images are compared, establishing a link between all pairs of nodes whose regions overlap between the two images. The resulting graph represents moving regions as chains of nodes with one incoming and one outgoing edge each, while splits and merges are represented by corresponding junctions in the graph's topology.

This approach requires the experiment to be captured at a sufficiently high framerate for the moving regions enclosed by the fluid-fluid interface to overlap between subsequent frames. If a region moves at a faster rate, it is no longer possible to track its motion through the porous medium and the graph's topology no longer corresponds to fluid flow.

5.1.2 Graph visualization

Taking advantage of the properties listed in section 5.1.1 allows for the reduction of the whole image sequence dataset into a more compact, temporally aggregated representation.

Applying the approaches described in section 5.1.1 yields a directed, acyclic graph with unweighted edges. Each vertex is inherently associated with spatial position corresponding to the center of mass of all pixels contained in a region, as well as an integer timestamp corresponding to the frame index the region was computed from.

In datasets generated from viscous flow experiments, each pixel is uniquely associated with at most one displacement timestamp, resulting in a strong correlation between the temporal and spatial attributes, as each pixel can only be part of a region's expansion in at most one timestep. This reflects the observable tendency of the injected fluid to flow from the inlet towards the outlet during the experiment.

The majority of vertices in a flow graph have a degree of 2, with exactly one incoming edge and one outgoing edge. In a viscous flow experiment, such vertices correspond to a finger moving forward through a throat or expanding to fill a pore in the porous material. When fluid passes through a pore and flows out through multiple throats, the node representing the flow at this pore has a higher degree, with at least two outgoing edges and one incoming edge.

As a result, the graph's edges are sparse, with only few junction points at which significant topological features can occur. Barring artifacts due to noise or algorithmic inaccuracies, these junctions correspond directly to a subset of pores in the material, and are therefore dictated primarily by the local structure of the porous medium. In particular, repetitions of the same experiment during which fluid flows through the same porous medium should yield similar positions for junction nodes, provided that the injected fluid traverses a similar path across these repetitions.

In the context of a comparative visualization between multiple datasets, the spatial position of each node is a suitable attribute for matching nodes together across experiment repetitions. As long as the porous medium is not rearranged or moved between experiments, the relationship between position and the local pore geometry, which dictates flow direction, velocity and behavior, is preserved.

Additionally, the flow graph's edges only connect nodes across subsequent timesteps, making it a *layered graph*, with the timestamp of each node denoting its layer. The set of edges can be fully decomposed together with their respective nodes into a series of bipartite subgraphs, connecting adjacent layers of nodes for each subsequent pair of frames [FFK+98].

Node-link diagrams

As each vertex in a flow graph is located at a meaningful position in two-dimensional space, corresponding to a physical point on the experimental setup, and its edges only connect to vertices within its immediate spatial and temporal proximity, the resulting graph can be naturally mapped to a node-link diagram. Due to the requirement for spatial adjacency in the original image sequence in order for an edge to be established in the flow graph, most edges do not cross each other when using the center of mass of each vertex as the node's position between which edges are drawn.

The node-link diagram of a flow graph yields a reduced visual representation of the path taken by the fluid throughout the experiment, encompassing the dataset's entire spatial domain. Due to the direct correspondence between node positions and the coordinate system of the input images, it is possible to overlay the flow graph on top of the original image sequence, combining a temporally aggregated representation of the whole dataset with a detailed view of individual frames in the series.

5.2 Multi-dataset comparison

When performing and recording experiments with different parameters, or multiple runs of the same experiment with the identical parameters, the resulting datasets can be visualized and analyzed in a comparative view to gain insights into the similarities and variances in the underlying physical processes.

The experimental datasets analyzed within the scope of our work span a variety of temporal and spatial resolutions, ranging from rapid fluid displacement on the order of milliseconds, to viscous flow processes observed for several minutes. These significant differences in the temporal domain, along with potential spatial misalignments between datasets due to camera movement between runs and variations in the paths taken by the invading fluid even across multiple runs of the same experiment, limit the direct comparability of the raw image sequences.

In order to compare datasets across this wide range of characteristics and visualize their differences, they are first transformed into flow graphs (see section 5.1.1) to reduce the reliance on spatial and temporal similarity. The nodes within these resulting graphs can then be matched across datasets to identify similarities and differences, based on their attributes and their topological neighbors.

5.3 Time-dependent metrics

Over the course of each experiment, the interactions between the fluids and the porous medium containing them undergo changes based on the local geometry. These interactions are reflected in quantifiable metrics, which can be derived from the dataset during preprocessing. To study changes in the underlying physical processes over time, space and across parameters, multiple such metrics should be collected and attached to nodes within the flow graph. Such metrics include the number of fingers, rate of displacement, the lengths of boundaries between fluid and solid phases, and local flow velocity.

5.3.1 Number of fingers

Viscous fingering describes a range of distinctive patterns formed by a fluid under the effects of predominantly viscous forces as it permeates through a porous medium and displaces another fluid [Hom87]. These patterns are characterized by the appearance of multiple fluid-fluid interface fronts propagating forward through the porous medium, splitting, merging and stopping according to the local geometry of the solid material surrounding the fluids [DiC13]. Each of these individual interface fronts is called a *finger*, and the number of actively moving fingers varies significantly throughout the displacement process.

Conventionally, when analyzing viscous flow, the difference between each pair of temporally adjacent binarized images is computed [DNS86]. The resulting difference images are then each segmented into distinct regions by labeling connected components, with each such component corresponding to a finger.

This approach relies on pairs of images differing in cohesive, connected regions that correspond directly to fingers. As a result, the choice of framerate relative to the flow velocity impacts the output quality. In a high-framerate recording of an experiment, slow-moving fluid-fluid interfaces may show few or no pixel differences between adjacent frames, thus vanishing from the output of a naive algorithm based on subsequent frames.

This can be mitigated by sampling pairs of frames across a longer timestep interval, causing the resulting difference images to contain broader fronts for each moving finger. However, this mitigation causes the pixel area of difference regions to no longer correspond to the flow rate at each finger, and delays the point in time at which a fast-moving finger is split into two separate regions when passing through a junction. The optimal choice for the sampling interval also depends on the flow velocity, which differs according on the local geometry of the porous medium.

To avoid these issues and the drawbacks associated with their mitigation, we propose an alternative algorithm for tracking fingers in viscous flow across varying fluid-fluid interface velocities. This algorithm takes an aggregated time-valued image as an input to iteratively propagate fluid-fluid interfaces over time and space, yielding a graph and a labeled image as outputs. This approach is described in further detail in section 6.2.4.

5.4 Performance requirements

To ensure the interactive usability of our implementation, we require all computationally heavy processing steps to execute asynchronously, allowing the interface to continue responding to user input.

All intermediate results should be cached and memoized, allowing modules further down the visualization pipeline to reuse previous results if the respective parameters did not change, instead of needing to recompute all previous steps. This allows the results of parameter adjustments to be seen much more quickly, improving the user experience.

All algorithms involved in the visualization approach must be scalable to arbitrary data set sizes by limiting their memory usage to a fixed upper bound. In particular, the tool must correctly process datasets that exceed the amount of available random access memory on the machine.

6 Implementation

Based on the design outlined in chapter 5, an interactive visualization tool has been developed. It implements a full pipeline for processing and visualizing two-dimensional image sequence datasets in the context of porous media flow analysis, while meeting performance and scalability requirements imposed by the primary datasets (see chapter 4).

6.1 Architecture

The open-source visualization framework *MegaMol* [GKM+15], in development at the University of Stuttgart, serves as the foundation for our implementation, providing a graphical user interface for building, reconfiguring and executing visualization pipelines, referred to as *Projects* within MegaMol. A plug-in framework provides an application programming interface for extensions, which can be included or excluded in a particular build configuration at compile time.

MegaMol includes a range of built-in plug-ins, providing functionality for performing various Scientific Visualization and Information Visualization tasks. Its current functionality is primarily tailored around the visualization of particle-based datasets in the context of molecule research.

Within the scope of this work, we develop the *Image Series* plug-in as an extension to MegaMol, containing several configurable modules that each implement a step in the visualization process and can be combined to form a full pipeline. The *image series data 2D* data type is defined by this plug-in, and serves as the primary data carrier for inter-module communication.

6.1.1 Technology

Due to its nature as a MegaMol plug-in, the software is primarily written in C++, using the C++17 standard. Library usage is limited to the C++ standard library, MegaMol's own public functions, as well as *VISlib*, a utility library for visualization which is developed and maintained as part of MegaMol. All display operations are implemented in OpenGL, using GLSL shaders to apply color maps when rendering image sequences.

Additionally, an auxiliary graphical tool is used for visualizing the graph data emitted by the plug-in's filter modules. This external visualization tool, tentatively named *LuaVis*, was implemented as part of this work. It is written in Lua, a dynamic scripting language which was chosen for its support for live reloading using a custom state-preserving script loader. This enables rapid prototyping of visualization techniques without requiring a lengthy compilation step after each code change. The *LuaJIT* virtual machine is used instead of the standard Lua interpreter, due to significant performance gains when visualizing larger amounts of data.

6.1.2 Streaming

In order to process datasets at scales exceeding the amount of random-access memory available on the machine while allowing the user interface to remain responsible, an asynchronous, streaming-based approach for data acquisition, processing and visualization has been implemented on top of MegaMol's graph-based architecture.

Data acquisition

The Image Series Loader module serves as the primary method of importing image series datasets into MegaMol. When supplied with a file system path containing image files and a regular expression for filtering the series to specific file name patterns, an asynchronous image series object is produced containing metadata for the dataset as a whole. Attributes such as the resolution, channel count and bit depth of the images contained within the series are computed by parsing the first image file in the series, under the assumption that these properties are shared by all images in the series.

Requesting an image from the series via the module's output slot immediately returns a reference-counted asynchronous image data object and submits an image-loading task to a multi-threaded worker queue. The caller immediately receives the image object and can pass it through asynchronous filters.

Hashing

As the image object's memory address remains constant across repeated invocations, even after the data is loaded, the image data pointer itself is usable as a key for cache lookups and change detection. Additionally, a 64-bit hash is computed for each image in a series, with the file path serving as the hash function's input for imported images. The hash does not depend on the image contents themselves, allowing it to be computed for all images in the series ahead of time without needing to load and process them first.

Caching

Each data source and filter in the visualization pipeline stores references to its output images in a Least-Recent-Used cache, using the image's hash as the key for cache entries. These cache entries are created immediately when an image from the series is requested, even if its computation occurs asynchronously. Requests for an image that has not yet been processed are answered with a shared pointer to the same image instance, without causing another work item to be added to the asynchronous processing queue.

If the parameters of a source or filter module are changed, the cache is cleared immediately, dropping all references to the image objects produced according to the module's old parameters. For images that are no longer in use, this results in the immediate release of memory associated with them. Existing asynchronous image processing tasks are permitted to finish their work, dropping the reference to the stale input and output data objects upon termination.

Filter chains

Image series filter modules provide one or more input slots, allowing the user to establish a connection with a corresponding output slot of a data source or another filter. These connections are displayed as curved paths between the modules in MegaMol's user interface, providing a visual representation of the visualization pipeline's data flow.

When data is requested through a filter's output slot, the filter propagates the request to the module connected to its input slot. As all image series modules immediately return asynchronous data objects with consistent hashes regardless of the image's processing state, the filter can launch its own asynchronous operations on this object right away. The asynchronous image data object provides functions to wait until the data is fully processed. If no worker thread is currently processing the data object, a work-stealing operation is performed and the waiting thread immediately begins performing the image's required processing steps. This can trigger further preemptive computations, traversing the dependency chain and ensuring that none of the thread pool's workers are idle.

If the data object entering a filter via one of its input slots differs as a result of upstream parameter modifications, the difference in the object's hash results in a different cache slot being selected, propagating the change to any modules connected to the filter's output slots and triggering asynchronous recomputations along the visualization pipeline.

6.2 Visualization pipeline components

Each component in the visualization pipeline is implemented as a *module* in MegaMol. A module defines a set of *caller slots* (inputs), *callee slots* (outputs) and *parameter slots* (customizable arguments to parametrize the module's operations).

The listed components have been developed within the scope of our work, with MegaMol's existing codebase providing the framework for modules, as well as the user interface for connecting and configuring them.

6.2.1 Image registrator

When recording several experiments in sequence using the same porous medium, subtle changes in camera position will cause the resulting image sequences to vary in alignment. To compensate for these variations, an image series registration module has been implemented, which can be inserted into the visualization pipeline's filter chain. Two image series are passed to the module: a moving image series to be transformed by the filter, and a reference image series to serve as an alignment target. The first images of both sequences are used for alignment, under the assumption that the first frame in a dataset captures of the experiment's solid structure without any visible dyed fluid.

To align the image sequences to each other, the moving image is first preprocessed using a Gaussian Blur filter to reduce the impact of noise on the alignment. This preprocessed image's gradient along both axes is then computed via the Sobel Operator and stored as a two-channel intermediate image. An affine transformation matrix, first initialized to the identity matrix, is used to transform the moving image via progressive adjustments.

In each iteration, the intensity value at each pixel of the reference image is sampled. The pixel's coordinates are transformed by the current affine matrix, which are used for a sub-pixel lookup on the moving image. To reduce aliasing artifacts, bilinear interpolation is used for points that fall between integer pixel coordinates. The square difference between the sub-pixel intensity value of the moving image and the original image's intensity at the corresponding position is computed and summed up across all pixels.

A derivative matrix for the current pixel is constructed by sampling the preprocessed gradient image at the transformed position and multiplying the original pixel coordinates by the resulting vector. This local derivative matrix is multiplied by the previously computed square error between the moving and reference images, and likewise summed up across all pixels.

The resulting sum of weighted derivative matrices is multiplied with a customizable convergence rate and added to the current transformation matrix. This gradually translates, rotates, scales and shears the moving image towards the reference image in each step. Once the error between the moving and reference images is below a customizable threshold, the algorithm terminates and the registration is complete.

At this point, the module's output slot provides a modified view of the input image series, with the computed affine transformation matrix applied to each image, such that the output aligns with the reference image series.

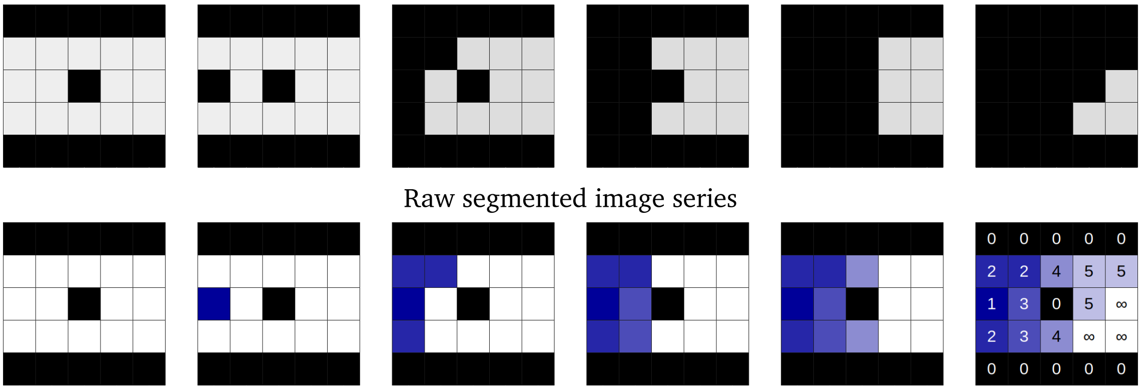
6.2.2 Preprocessor

The grayscale image sequence is passed through a preprocessing module, which applies a series of additional filtering steps to each image passing through it, in order to prepare the dataset for further processing stages.

The primary application of this module is the *segmentation* step, which maps all pixel values in the image to 0 or 255 using a step function with a customizable threshold. By default, all pixels with an intensity of up to and including 127 are mapped to 0, while all pixels with values of 128 and above map to 255. Both the cut-off threshold and the step direction (inverting or non-inverting) can be configured as filter parameters. The resulting monochrome image serves as the input for further processing steps.

The module also provides optional filters for deinterlacing input images by applying a customizable horizontal offset to odd-numbered rows of pixels, and for applying a per-pixel mask to all images in the sequence.

6.2.3 Displacement Time Filter



Raw segmented image series

0	0	0	0	0
2	2	4	5	5
1	3	0	5	∞
2	3	4	∞	∞
0	0	0	0	0

Incremental construction of timestamp map

Figure 6.1: Step-by-step conversion of a raw segmented image series to a timestamp map, in which each pixel encodes the index of the frame at which the segmentation threshold was crossed, indicating that the fluid-fluid interface has reached the corresponding location.
Bottom right: final timestamp map with numeric labels. Background pixels that never cross the threshold are labeled by the maximum applicable pixel value for the image format.

During displacement of a fluid in a porous medium via the injection of another fluid, the interface between the fluids propagates forward in the direction of the flow. This behavior has been observed across all test datasets produced by recording viscous fluid displacement. Taking this observation into account, the segmented time-dependent dataset can be reduced from a monochrome image sequence to a single aggregate image. Each pixel of the resulting image stores the timestamp of the first frame in which this pixel crossed the segmentation threshold. This timestamp map is populated gradually by iterating through the input image sequence in ascending order, and marking all segmented pixels in the output image with the timestamp of the current frame (see section 6.1).

6.2.4 Flow Graph Builder

The input to this filter is a single-channel image as produced by the displacement time filter (see section 6.2.3), with each pixel containing the earliest timestamp at which a pixel's intensity crosses the configured segmentation threshold. This image serves as a compact representation of the fluid's path as displacement occurs, limiting the filter's memory requirements to a constant multiple of the dataset's spatial resolution, regardless of the number of frames within the dataset.

During the filter's execution, an image-sized output buffer is allocated, mapping each pixel to a unique numeric label from 5 to 255, with an initial value of 0 for all pixels, indicating the absence of a label. These labels correspond to connected regions with the same displacement timestamp. Adjacent regions with differing displacement timestamps may be marked with the same label value, as each region's unique identifier consists of the combination of the label and timestamp. The label's numeric value is only used to distinguish physically separate regions of newly displaced pixels at any given timestamp, numbering the fingers of the fluid's branching flow.

Inlet discovery

To discover suitable starting points for generating the flow graph, an inlet discovery step is performed on the whole input image. During this step, a set of all locally minimal regions is built, with the pixels of each region having the same segmentation thresholding crossing timestamp, and no neighboring region having a lower timestamp.

From each pending pixel of the input image above a specific initial timestamp threshold, a speculative four-neighbor flood fill is initiated. The initial threshold is customizable and serves as a filter for misdetecting solid boundaries as fluid inlets. All connected pixels with the same timestamp as the starting pixel are marked and added to a queue, propagating the flood fill operation in an upcoming iteration. If a pixel with a lower timestamp (above the initial threshold) is found, the region is not a local minimum and the flood fill is aborted. In this case, all marked pixels are reverted. If a pixel with a greater timestamp is found, the pixel is marked as a front-facing interact, but not added to the flood fill queue. This interface mark is only retained after the fully processed region is a local minimum and has a sufficiently large area.

After all connected pixels have been filled and no lower-timestamp neighbor is present, the total number of pixels is compared to a configured inlet area threshold. If this threshold is met or exceeded, the region is marked as a valid inlet and can be used as the flow graph's starting point. All marked front-facing interface pixels are added to a queue for further processing. If the number of pixels in the area is lower than the specified inlet area threshold, or if the region borders a lower-timestamp neighbor and thus loses its status as a local minimum, all marked pixels are reverted to a neutral label and any front-facing interface pixels are discarded.

Masking

A pixel exceeding the segmentation threshold can fall under several categories.

Dyed fluid: Assuming a displacement of an initial clear fluid by an injected dyed fluid, these pixels start out below the threshold towards the dataset's initial frame, with pixels of subsequent frames changing in value over the course of the experiment.

Solid boundaries: As the experimental setup is stationary and the camera is not moved while capturing, the porous material through which fluids flow throughout the experiment is visible in all frames across the dataset. Solid pixels are characterized by being present at the same locations throughout the entire temporal range of the dataset.

Noise: Dust, impurities on the camera's lens, lighting conditions or other artifacts can lead to pixels crossing the segmentation threshold without being directly related to the experiment. Noise caused by physical impurities will be present at the same pixel locations throughout the dataset, while electronic sources of noise are often only temporarily visible on some frames.

If the dataset's initial frames contain only undyed fluid within the porous medium, a mask of solid boundary pixels can be automatically generated from these images to be used for further processing, without requiring the manual creation and cleanup of such an image.

In order to generate the mask, a loop iterates over each pixel in the input image: if the earliest timestamp at which the pixel's segmentation threshold is crossed lies before the minimum flow interface timestamp (see section 6.2.4), it is marked as a mask pixel in the label array. Otherwise, the pixel's initial neutral label is retained.

Flow tracing

To analyze the flow paths taken through the porous material, the interface between the two fluids is traced iteratively. Starting from the timestamp corresponding to the lowest eligible local minimum detected during inlet discovery (see section 6.2.4), all queued front-facing interface pixels are first split into connected regions. Interface regions are allowed to connect across gaps of 2 pixels horizontally or vertically while still being combined into the same label. This limits the effects of noise, preventing the same physical fluid front from

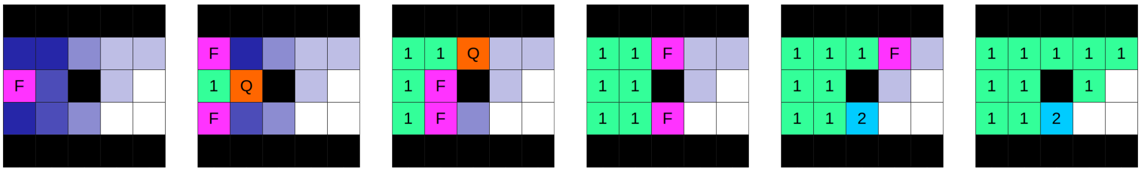


Figure 6.2: Flow tracing process across a range of 6 frames, overlaid on the previous example timestamp map (figure 6.1). Pixels marked with **F** represent the current fluid-fluid interface being traced. **Q** represents pixels in the pending interface queue, with a timestamp greater than the current iteration. Numbers indicate the final output labels, uniquely distinguishing connected regions within a single timestamp.

being incorrectly split into two disjoint fingers by the algorithm. Each connected interface is assigned an sequential label value, unique to its timestamp index. From the interface's set of starting pixels, an iterative breadth-first flood fill algorithm is performed. For each equal-timestamp pixel on the input image, the corresponding index in the label buffer is populated with the current region's numeric label value, similar in function to a Connected Component Labeling filter. If a lower-timestamp pixel is encountered, it is ignored: the pixel is not queued for further flood filling, and the label buffer is not modified. If an adjacent mask pixel is encountered, it is marked as a fluid-solid interface and counted separately for the current region. the pixel is not queued for further flood filling. If a higher-timestamp pixel is encountered, it is added to the pending front-facing interface queue. the pixel is not queued for further flood filling, and the label buffer is not modified.

Once all pending interfaces for the current timestamp have been processed, the timestamp is increased by one and the propagation loop is started once again. Pending interface pixels with a greater timestamp than the propagation algorithm's current iteration index are postponed until their respective timestamp is reached, preserving their interface pixels. Upon reaching the maximum timestamp index, equal to the number of frames in the dataset, the algorithm terminates, yielding a labeled image of all distinct connected flow regions per timestamp.

Graph generation

While computing the connected regions at each timestep, a graph structure is built up incrementally alongside the labeled image. Each node represents a region of adjacent pixels that were displaced from one frame to the next, corresponding to a finger of the fluid's flow. Each fluid-fluid interface between adjacent regions forms an edge between two nodes. Edges are always established between nodes of neighboring timestamps, tracing the movement of each finger as the fluid flows through the porous medium. When a junction within the material is reached, a single connected region will split into multiple front-facing interfaces, causing the number of fingers to increase from one frame to the next. Each connected region in the output image, when combined with the timestamp index in the input image, corresponds directly to a node within the graph. For each node, several metrics are computed and attached as metadata based on the region's properties. Along with basic attributes for the node's unique numeric identifier, timestamp index and axis-aligned bounding box spanning all contained pixels, several other metrics can be derived from the region's shape. The number of connected pixels that crossed the segmentation threshold within the current frame corresponds to the rate of displacement within a specific region. Each node is also associated with a spatial position via its center of mass, consisting of the mean position of all pixels contained within the region. For intermediate fluid fronts that do not contain any new threshold-crossing pixels for a given timestamp, the mean position of queued interface pixels is used to compute the center of mass instead. The interface length (in pixels) between the non-wetting and wetting fluid, as well as between the non-wetting fluid and the solid boundary, is measured and tracked at each timestep. Solid and fluid interfaces are distinguished by comparing each pixel's label value during flood fill propagation. If a pixel was previously marked as "solid" in the pre-processing step, by means of crossing the segmentation threshold at an early timestamp, any adjacent fluid pixel contributes to the fluid-solid interface length for that region.

Velocity computation

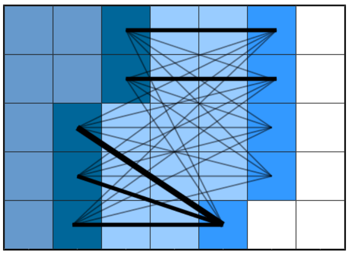


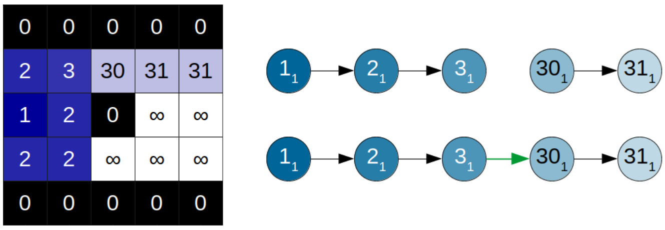
Figure 6.3: Computation of the Hausdorff Distance, measuring how far a flow front propagated from one frame to the next.

Whenever a link between two temporally adjacent fluid-fluid interface nodes is established, the *Hausdorff Distance* between their sets of interface points is used as an approximation for the interface's velocity [HKR93]. For each point of the first interface, the square distance to the nearest point in the second interface is computed, the maximum of which is tracked (see figure 6.3). The square root of the resulting value is used as a measure for the distance traveled by the interface from one frame to the next. It is stored alongside the graph node of the region bordered by the two interfaces.

6.2.5 Flow Graph Postprocessor

After the flow graph is built, it is written to the file system in a comma-separated value format, which can be read and processed by other tools. When imported into *LuaVis* (see section 6.1.1), several post-processing steps are applied.

Broken path patching



0	0	0	0	0
2	3	30	31	31
1	2	0	∞	∞
2	2	∞	∞	∞
0	0	0	0	0

Figure 6.4: Left: example timestamp map of a flow that halts for 27 timesteps before resuming.
Right: resulting initial flow graph (top), and flow graph with added edge after broken path patching (bottom).

During flow processing, timestamp gradients exceeding a customizable steepness threshold are ignored. By default, neighboring regions that are more than 20 frames apart prevent the fluid interface from propagating. To correctly track the flow even after the movement of a finger pauses or reverses temporarily, unlinked pairs of source and sink nodes in close spatial proximity are joined by an artificial edge (see figure 6.4).

For a pair of unlinked nodes to be eligible for patching, the source node must be located within 30 pixels of the sink node, measured by Euclidean distance. Additionally, the source node's timestamp must be strictly greater than the sink node's timestamp. If multiple sink nodes are eligible for linking with a specific source node, the closest sink is chosen, and can no longer by connected to any other source node.

Junction normalization

To simplify the resulting graph's structure, junctions at which a node splits into three or more descendants are normalized to be represented by multiple consecutive two-way splits. For each such junction, the child node with the greatest area is selected from the node's descendants. The edge between this child node and its parent is retained, whereas all other child nodes are re-parented to a newly inserted "dummy node", which itself is connected to the parent node. The dummy node retains the original parent node's timestamp and center of mass, but its area and interface lengths are set to zero. An auxiliary attribute is set for each dummy node, causing it to be excluded from aggregate metrics over the whole dataset and thus preventing any bias from being introduced. If a dummy node has more than two child nodes, the process is repeated iteratively until the final dummy node in the subgraph only has two outgoing edges. The resulting graph can be readily stored as a flat list of nodes, requiring only two columns for the outgoing nodes' numeric identifiers. Neither a variable-size column of links on each node, nor a separate list of edges is required, allowing the graph to be exported as a single comma-separated value table without the loss of edge information.

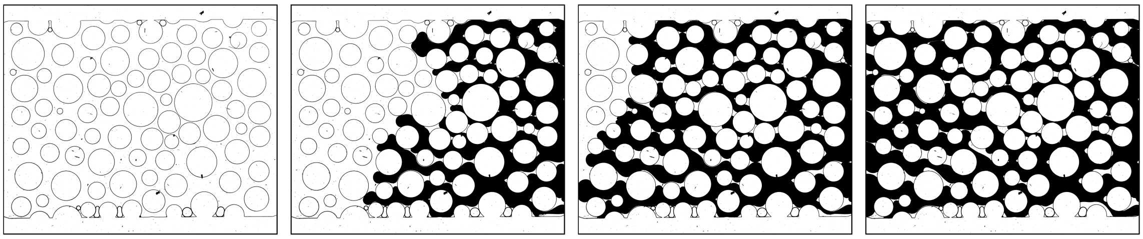
Graph simplification

As a finger slows down and stops, its active flow front gradually shrinks in size, eventually splitting into multiple disjoint regions. These regions do not represent multiple distinct paths taken by the fluid through the porous material, as they are not separated by solid boundaries. To prevent them from being counted as individual fingers, an iterative simplification step is repeated on the completed graph. For each sink node with no outgoing edges, its area in pixels is compared to a customizable threshold. If the area falls under the threshold, the node's attributes are merged into its parent node. The nodes' areas are added together, the bounding box is expanded to encompass both nodes, the center of mass, velocity and interface lengths are updated to the weighted average according to the ratio between the nodes' areas. If the merged node lacks any more outgoing edges and its area falls under the threshold, the process is repeated, gradually eliminating the artifacts resulting from slow fluid displacement. As this step is performed after Broken Path Patching (see section 6.2.5), temporary sections of reduced flow rate with continued movement after a delay are not affected by this step, as the patching step establishes edges between nodes that were originally sinks, making them ineligible for culling unless their descendant subgraph as a whole falls under the merging threshold.

7 Results

To demonstrate the functionality of the visualization pipeline in a practical setting, the techniques implemented in chapter 6 are applied to the experimental data described in chapter 4. The resulting visual representations are evaluated for suitability in the context of studying porous media, particularly when comparing datasets from multiple experiments to identify similarities.

7.1 Flow visualization



(a) Initial state **(b)** Mid-displacement **(c)** Breakthrough **(d)** Final state

Figure 7.1: Still images extracted from dataset for $Ca = 10^{-2}, M = 10$ at varying points in time [FSK+21].

Loading the dataset corresponding to the experiment with parameters $Ca = 10^{-2}, M = 10$ into MegaMol via the *Image Series Loader* module yields a lazy-loaded image sequence (see figure 7.1). Attaching an *Image Series Renderer* and connecting it to a 2D view module displays a preview of the raw dataset in MegaMol's user interface, loading image data from the file system on demand based on the timestamp chosen in the view.

In order to aggregate this dataset along its temporal domain, an *Image Series Timestamp Filter* module is added, producing a single output image from the complete input sequence. The resulting image is processed by the *Image Series Flow Labeler* module, which produces an indexed image containing numeric labels for each pixel, as well as an auxiliary graph data object that is saved to a separate file for external analysis and visualization. The full pipeline configuration up to this point is shown in figure 7.2. The outputs of the *Image Series Timestamp Filter* and *Image Series Flow Labeler* modules can be viewed directly in MegaMol using the *Image Series Renderer* module's *Timestamp* and *Label* modes, which apply alternate color maps corresponding to the filters' output formats (see figure 7.3).

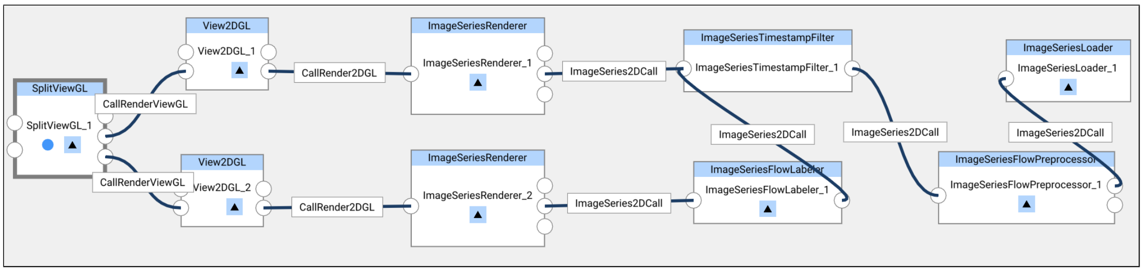
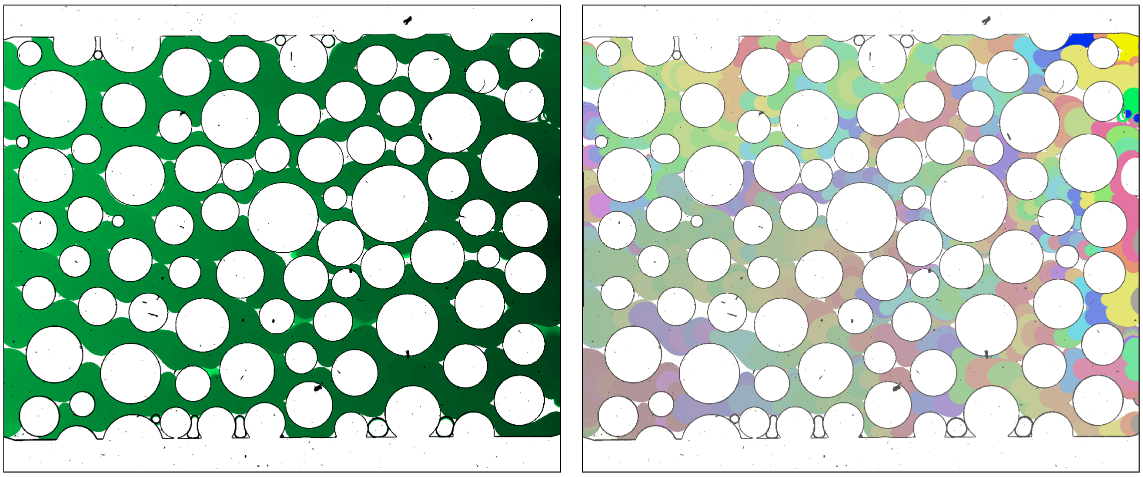


Figure 7.2: Screenshot of MegaMol's project graph user interface, depicting the visual pipeline for producing the output in figure 7.3. Control flow propagates from left to right, starting at the *View* modules and traversing the call network, represented by curved connectors between the modules. Data flows from right to left, starting at the *Image Series Loader* module.

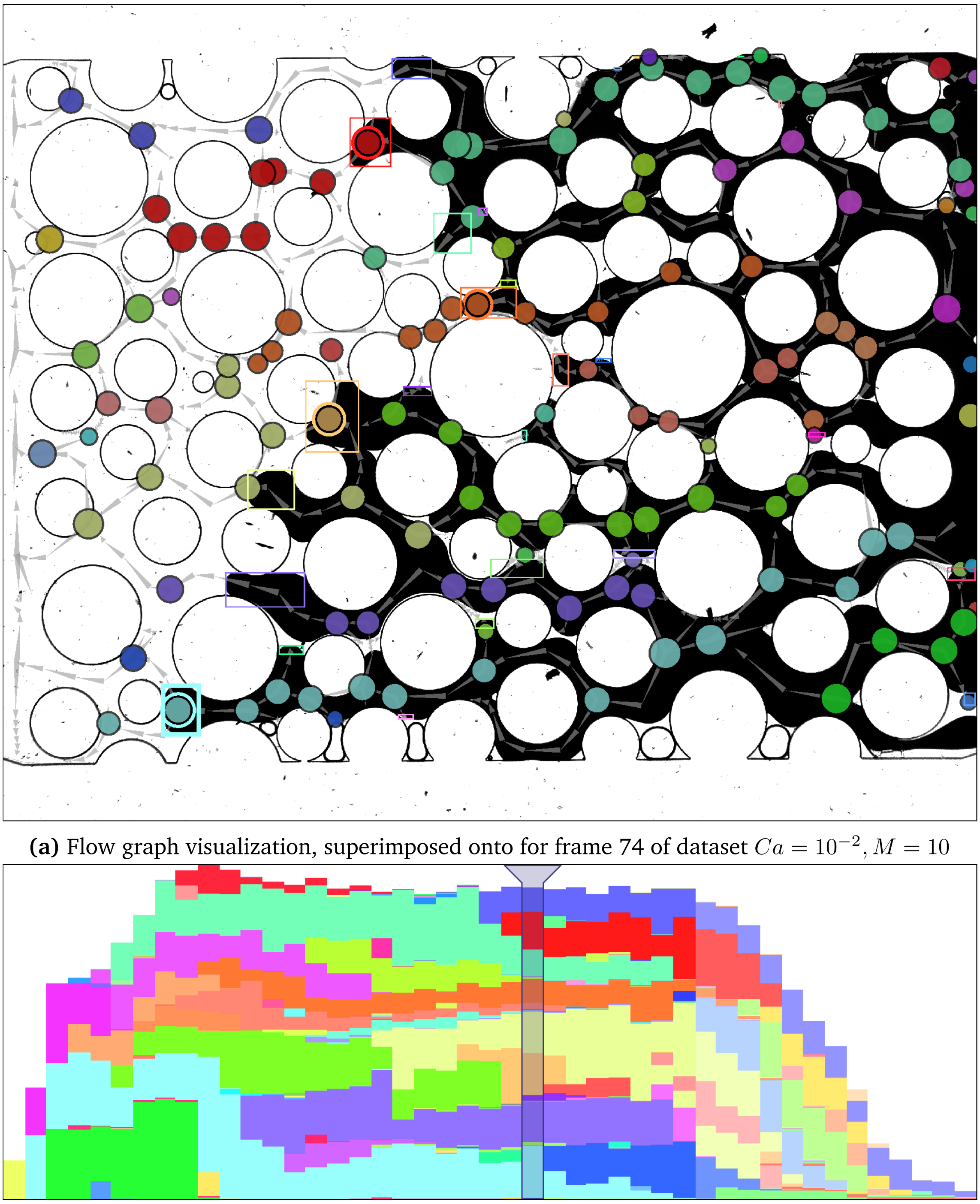


Timestamp map, with a linear color scale between dark green (low timestamps) and light green (high timestamps).

Colorized label map for the same dataset, assigning a unique label for every connected region at each timestep.

Figure 7.3: Intermediate visual representations of the dataset for $Ca = 10^{-2}$, $M = 10$ after processing.

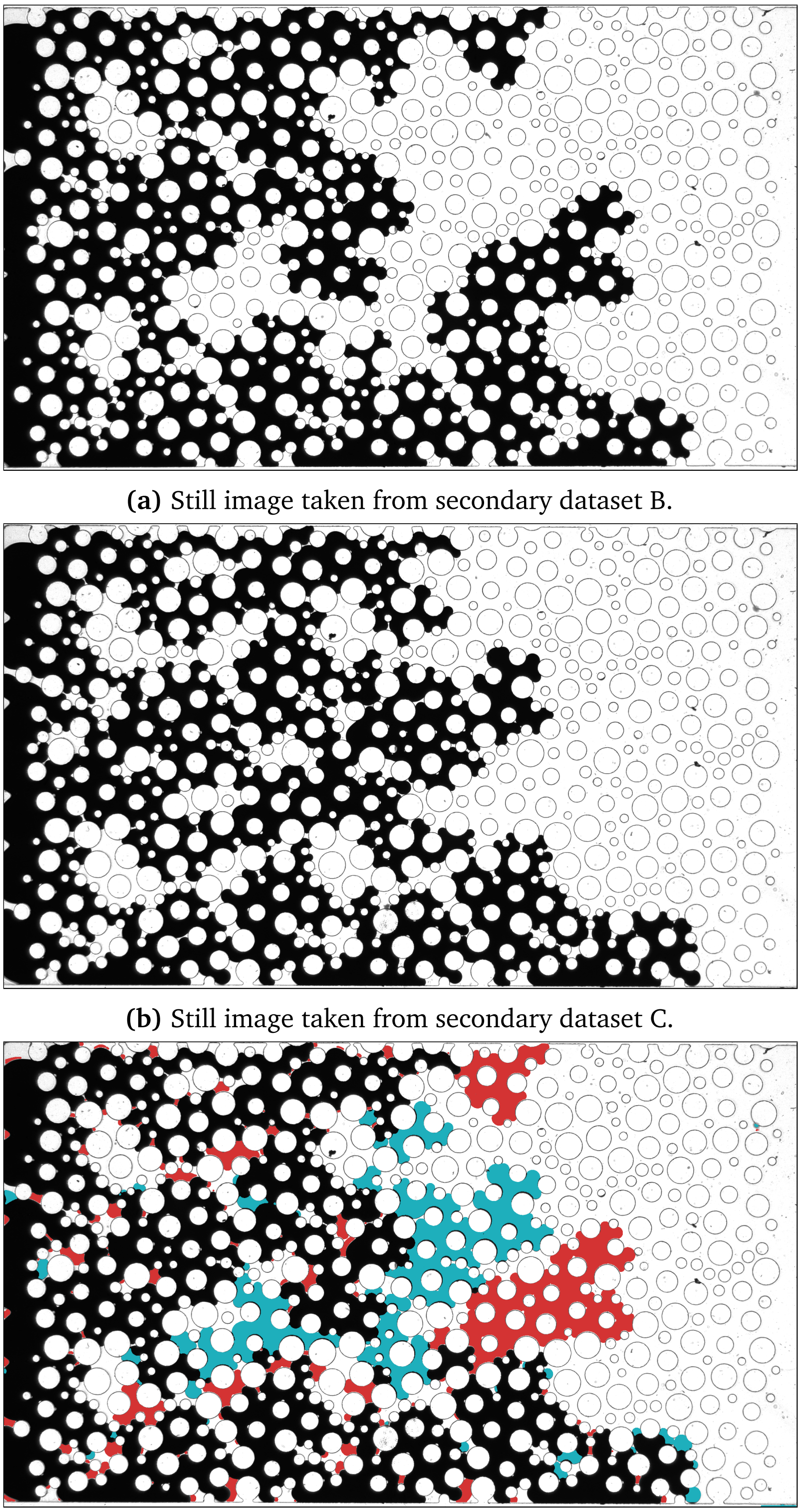
Using an external visualization tool developed within the scope of this project, the raw graph data file generated by the *Image Series Flow Labeler* module can be loaded and explored interactively. By superimposing a node-link diagram of the graph on top of a still image from the dataset (see figure 7.4), the paths taken by the invading fluid throughout the experiment can be viewed in the context of the surrounding porous material's geometric layout. The interactive temporal navigation features and dynamic highlights give a simultaneous view of the overall displacement process and details on individual frames of interest.



(a) Flow graph visualization, superimposed onto for frame 74 of dataset $Ca = 10^{-2}, M = 10$

(b) Stream chart plotting the displaced area for each timestep. The frame selection for detail view is marked by a semi-transparent cursor.

Figure 7.4: Visualization of the flow graph generated from $Ca = 10^{-2}, M = 10$. Colored borders highlight the graph nodes corresponding to the current frame, as well as the bounding boxes of their displaced area. Nodes on the same path within the graph are colored equally across the node-link view and the stream chart.

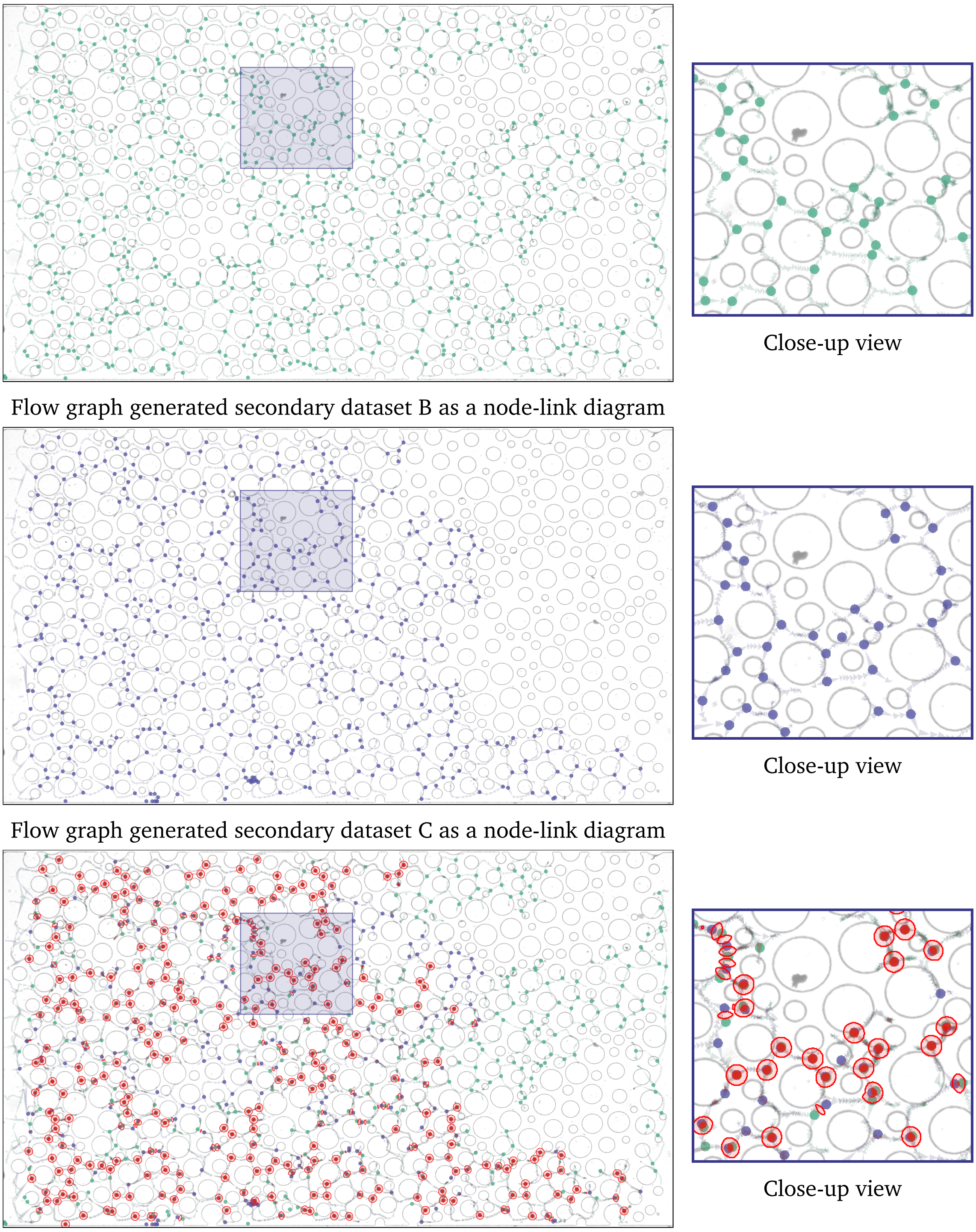


(a) Still image taken from secondary dataset B.

(b) Still image taken from secondary dataset C.

(c) Colorized difference image between the two input images, using red and cyan to highlight differing regions.

Figure 7.5: Raw image comparison across two datasets, highlighting variances between their flow paths. Data source: see section 4.2



Flow graph generated secondary dataset B as a node-link diagram

Flow graph generated secondary dataset C as a node-link diagram

(a) Unified view of both flow graphs, with matching junction vertices highlighted in red.

Figure 7.6: Individual and comparative views of flow graphs generated from two datasets. The graphs have been superimposed onto of the experimental setup. (Due to a limitation in the visualization tool, this background image is not fully aligned with the graphs)

7.2 Flow graph comparison

Two superficially dissimilar image sequence datasets from different displacement experiments initially appear to exhibit variances between their flow paths (see figure 7.5). Additionally, the datasets have been recorded at varying temporal resolutions, spanning 564 and 326 frames from the start of the experiment until breakthrough, respectively.

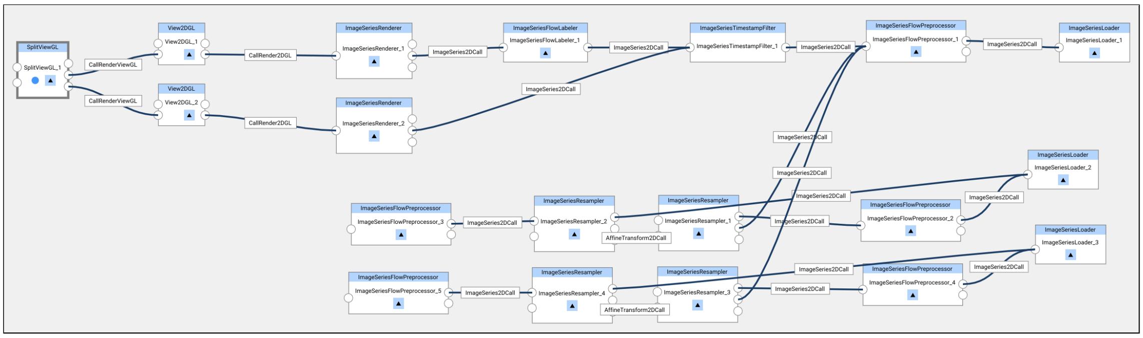


Figure 7.7: Screenshot of MegaMol's project graph user interface, depicting the visual pipeline used to generate the graphs in figure 7.6. Multiple datasets are loaded simultaneously, and image registration filters are used to automatically align datasets to each other.

To compare these datasets, the preprocessing and filtering steps described in chapter 6 are applied to both of the image sequences. To compensate for slight variances in camera positions between the two experiments, which would lead to inaccurate node coordinates, the image registration module (see 6.2.1) applies an affine transformation that aligns the second dataset to the first one.

The full visualization pipeline is constructed interactively using MegaMol's project graph (see figure 7.7), consisting primarily of filter modules operating on image sequences, which have been implemented within the scope of this work.

In contrast to the apparent differences between the raw input datasets, the resulting flow graphs show similarities between the datasets when viewed as node-link diagrams, as seen in figure 7.6. The points at which fingers split and merge, when measured by the center of mass of the area covered by the invading fluid in a single frame, matches precisely across both datasets in all locations marked in red. In this composite view, nodes with similar, but not identical positions between the two datasets are displayed as lens-shaped circle intersections, representing spatial uncertainty for partial matches. Nodes that are only present in one of the datasets retain their original cyan and blue coloring from the single-graph views.

In this visualization, only junction nodes are displayed explicitly. Trivial vertices with one incoming and one outgoing edge are hidden, leaving only their links visible. Non-linear flow paths that follow the surrounding pore geometry can be traced visually by following the curvature of these intermediate links.

7.3 Performance

In order to evaluate the scalability of the visualization approach from a performance standpoint, the run times of the most computation-heavy filters are measured for each invocation and summed across all frames within a dataset. Due to the visualization pipeline's asynchronous, parallelized architecture, these measurements are taken via thread-local high-resolution clocks, measuring the effective duration of each individual step, excluding any prerequisite filters for data dependencies.

Therefore, the cumulative times represent how long these steps would have taken on a single-core system. On a multi-core system, the actual performance (excluding disk reads) at run time is much faster due to the use of multiple threads.

The *Segmentation* and *Timestamp mapping* steps must be performed once per dataset, converting an image sequence into binary images by applying a threshold, then marking each pixel coordinate with the index of the first frame at which this pixel's value exceeds the segmentation threshold, producing a timestamp map. Upon completion, the results of these steps are cached for future computations until the parameters of the respective modules are changed. Only the *Graph generation* step, which converts the timestamp map to a flow graph, must be re-evaluated when changing most graph generation parameters.

To avoid timing inconsistencies due to caching, while still allowing the use of the in-application cache for memoization, the application is restarted between each measurement. Image file read times are excluded from this benchmark due to variations in file system access performance. All performance benchmarks were conducted on a 12-core *Intel Core i7-8700K* CPU clocked at 3.70 GHz with 32 GB of RAM.

7.3.1 Benchmark Results

Table 7.1 lists the total cumulative computation time to process all frames within each of the main datasets, with all times given in milliseconds.

Dividing each measurement by the number of frames in the input dataset yields metrics on the data processing rate. Table 7.2 lists these adjusted measurements per dataset in milliseconds per frame.

Dataset		**Size**	**Processing time** (cumulative)		
Ca	M	**Frames**	**Segmentation**	**Timestamp map**	**Graph gen.**
10^{-2}	1	55	109.382 ms	239.320 ms	46.422 ms
10^{-2}	10	295	646.956 ms	1307.141 ms	128.230 ms
10^{-3}	0.2	619	1326.541 ms	2577.005 ms	84.602 ms
10^{-3}	1	233	490.766 ms	1045.163 ms	117.122 ms
10^{-3}	10	321	675.688 ms	1373.779 ms	171.555 ms
10^{-4}	0.2	265	542.009 ms	1125.217 ms	78.266 ms
10^{-4}	1	627	1287.843 ms	2653.129 ms	285.225 ms
10^{-4}	10	1362	2539.616 ms	5718.646 ms	369.390 ms
10^{-5}	0.2	1038	2161.513 ms	4394.350 ms	262.852 ms
10^{-5}	1	832	1716.724 ms	3420.883 ms	300.602 ms
10^{-5}	10	2009	4021.489 ms	8159.216 ms	171.740 ms

Table 7.1: Cumulative processing time

Dataset		**Size**	**Processing time** (per frame)		
Ca	M	**Frames**	**Segmentation**	**Timestamp map**	**Graph gen.**
10^{-2}	1	55	1.989 ms/f	4.351 ms/f	0.844 ms/f
10^{-2}	10	295	2.193 ms/f	4.431 ms/f	0.435 ms/f
10^{-3}	0.2	619	2.143 ms/f	4.163 ms/f	0.137 ms/f
10^{-3}	1	233	2.106 ms/f	4.486 ms/f	0.503 ms/f
10^{-3}	10	321	2.105 ms/f	4.280 ms/f	0.534 ms/f
10^{-4}	0.2	265	2.045 ms/f	4.246 ms/f	0.295 ms/f
10^{-4}	1	627	2.054 ms/f	4.231 ms/f	0.455 ms/f
10^{-4}	10	1362	1.865 ms/f	4.199 ms/f	0.271 ms/f
10^{-5}	0.2	1038	2.082 ms/f	4.233 ms/f	0.253 ms/f
10^{-5}	1	832	2.063 ms/f	4.112 ms/f	0.361 ms/f
10^{-5}	10	2009	2.002 ms/f	4.061 ms/f	0.085 ms/f

Table 7.2: Processing time per frame

8 Discussion

Porous media and their effects on fluids passing through them are an important research target, with the overarching goal of understanding the physical processes that lead to the broad spectrum of observable flow behaviors across different fluids and porous materials. Visualization plays a key role in identifying patterns in experiments that could lead to new discoveries, as well as improving simulations to more accurately model porous flow behavior by comparing simulated processes to experimental data from their real-world counterparts.

Our work builds upon the concept of *Transport Networks* for modeling fluid flow introduced by Frey et al., and is evaluated on the experimental datasets collected during their study [FSK+21]. The visualization techniques we developed focus on achieving comparable results with an alternate approach that minimizes required preprocessing steps, such as the creation and cleanup of mask images. Similar to the existing work, we reduce large image series datasets to a more compact graph-based intermediate representation. Instead of creating a static transport network based on the experiment's pore space as a prerequisite step, the flow graphs in our approach trace the paths taken by the invading fluid throughout the displacement process.

Between multiple experiments dominated by viscous forces, key nodes at which merges and splits occur are located at similar positions within the porous medium based on the local pore space geometry, as long as the same medium is used throughout all experiments. This is the case even across minor variances in experiment parameters, allowing for a comparative visualization of multiple flow graphs. The topology of each flow graph follows the path taken by each finger of the invading fluid over both space and time, opening up the possibility of visualizing the graph not only by mapping its nodes to their original position within the micromodel's physical space, but using alternate graph layout algorithms to explore the flow graph's topology independently of the porous medium's structure.

The visualizations generated from the datasets under study highlight the applicability of our approach to viscous fluid displacement processes in porous media, providing a compact graphical representation of each experiment across its temporal domain as an overview, while enabling the interactive selection of timestamps to view details on demand at a specific moment during the process. Additional metrics derived from the experiments are attached to each node, allowing variables such as displaced area, flow velocity and fluid-fluid interface length to be tracked over time and per-interface.

As part of this work, the described visualization approaches were implemented as an extension to the MegaMol visualization framework developed at the Visualization Research Center (VISUS) at the University of Stuttgart. A full visualization pipeline for handling image series datasets has been implemented, resulting in a mix of general-purpose image streaming components for use beyond this project, and specialized modules to achieve specific preprocessing and analysis goals in the context of porous media research and viscous flow behavior. The source code is available on GitHub [1].

By planning for asynchronous processing, parallelization and heavy use of caching from the project's early architectural stages, the performance goal of keeping the application interactive and responsive throughout any workload combination involving the newly implemented modules was met successfully. Even large datasets consisting of multiple gigabytes of image data can be analyzed within minutes. Additionally, the introduction of reference-counted data objects into MegaMol's data flow architecture, in conjunction with a per-module result cache, ensures that computationally intensive processing steps do not need to be repeated after downstream parameter changes, improving interactivity by reducing waiting times after user input. The streaming architecture also allows the pipeline to operate on datasets larger than the available random access memory on the machine, as each cache has a fixed upper bound on the memory it may allocate.

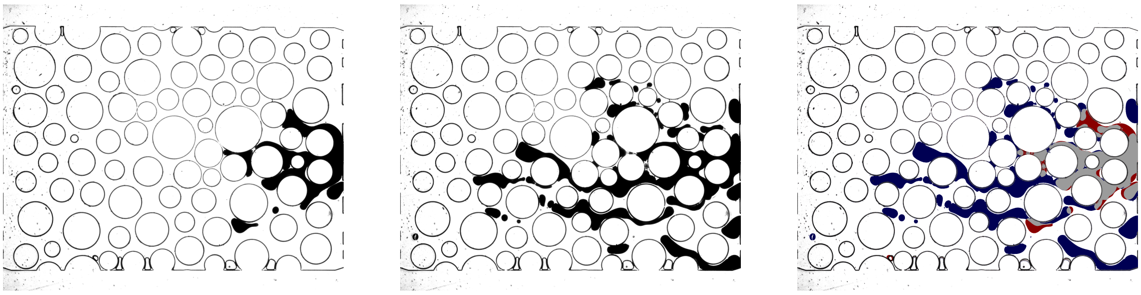
In order to conclusively answer the research question on whether fluid displacement processes under similar viscosity ratios and capillary numbers visually resemble each other (see section 1.1), additional quantitative evaluation and comparison of the flow graphs and their associated metrics is required. However, the results obtained from our work demonstrate the potential of this dynamic approach to flow graph generation for two-dimensional experimental datasets, and our implementation can serve as the basis for future research projects in the field of porous media visualization.

8.1 Problems and limitations

When applying the visualization techniques to some of the datasets from the primary group of experiments, such as $Ca = 10^{-2}, M = 1$, the limitations of the approach become evident. As demonstrated in figure 8.1, experiments conducted under high-velocity capillary flow produce isolated pockets of invading fluid that move at greater distances than their own size in the duration of a single frame. Even when applying the algorithmic modifications for capillary flow described in section 5.1.1, this rapid movement prevents edges from being established in the flow graph between occurrences of the same fluid region over time, as there are no overlapping pixels between subsequent images.

While the omission of manual masking and clean-up steps reduces the amount of labor required to analyze experimentally captured data, it also leads to a drop in output quality when significant impurities or artifacts are present in the dataset. Figure 8.2 highlights an example of noise-induced artifacts manifesting in the flow graph's topology. While adjusting the segmentation threshold eliminates the noise, this also prevents the micromodel's solid

[1] Image Series plug-in for MegaMol on GitHub: https://github.com/Marukyu/megamol/tree/feature/image-sequence-comparison/plugins/imageseries

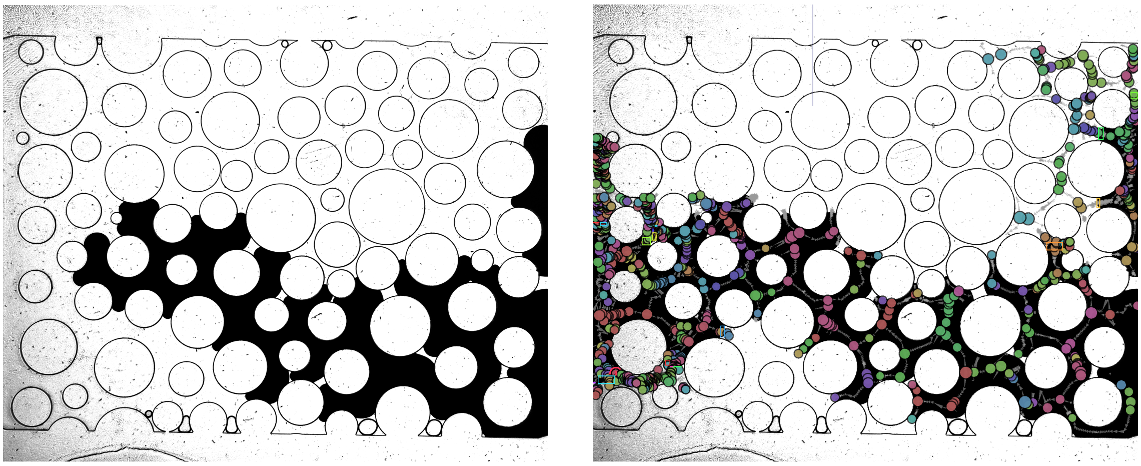


Frame 6

Frame 7

Colorized difference image between frames 6 and 7

Figure 8.1: In an experiment performed at $Ca = 10^{-2}, M = 1$, two subsequent frames in the dataset taken roughly 67 milliseconds apart show significant differences across a large area due to the high flow velocity and comparatively low framerate. Several regions between the two frames can no longer be directly associated to each other, as there is no overlap between them. Data source: [FSK+21]



Raw image from dataset

Superimposed flow graph

Figure 8.2: In the experiment performed at $Ca = 10^{-4}, M = 10$, various impurities are visible throughout the experimental setup, particularly near the image boundaries. These impurities lead to numerous false-positive junction nodes throughout the flow graph. Data source: [FSK+21]

cylinders and boundaries from being detected, resulting in inaccurate values for the fluid-solid interface length metric. In such cases, more extensive clean-up steps than the automatic masking functionality implemented in our work are required in order to produce useful results.

8.2 Future work

The dynamic approach to flow graph generation and visualization described in this work has been evaluated on several experimental datasets capturing flow displacement processes in porous media. To further explore its applicability, it can be applied to new datasets in broader contexts, such as comparisons between simulated processes and real-world experiments with equal conditions.

For datasets that currently produce suboptimal results due to noise, future iterations of these algorithms could include alternate preprocessing steps that attempt to distinguish solid experiment boundaries from noisy artifacts, based on shape recognition and by taking the motion of the invading fluid into account.

The visualization pipeline implemented in this project forms a flexible foundation for further image series processing projects built into MegaMol. The generic caching and asynchronous computation components can be extracted and applied to other subprojects within MegaMol, improving performance and interactivity in unrelated parts of the codebase.

8.2.1 Improved flow tracing

Viscous flow is currently tracked progressively, by following the gradient left behind by fluid-fluid interfaces on the timestamp map (see section 6.2.4). While this algorithm correctly tracks existing fingers, it requires an initial set of *inlet regions* to be constructed first, which are local minima in the timestamp map that serve as the starting point for tracing the fluid's flow along one of its paths through the porous medium. This approach, described in section 6.2.4, works well for experiments with a single, clearly defined inlet point from which fluid starts flowing into the porous medium. However, if another finger emerges at a significantly later point in time, it may not be recognized by the inlet discovery step, thus leading to the subsequent tracing step failing to follow this finger.

While it is possible to work around this issue in the current implementation by increasing the time threshold by which pending flow fronts are allowed to remain in the queue, this comes with the tradeoff of increasing the likelihood of mask pixels being misdetected as invading fluid pixels, and thus inaccurate values for the displaced area and interface length metrics.

In a future revision, inlets detected via local timestamp minima could be placed on a separate queue without a time threshold, allowing high-timestamp regions to remain in this queue until the algorithm has reached their corresponding frame index, at which point the regular threshold for moving fingers is used.

8.2.2 Visualization of capillary flow

The algorithms and visualization techniques discussed in this work are primarily designed around flow dominated by viscous forces. Capillary flow displays significant differences in its characteristics, violating the assumptions made by some of the preprocessing steps about the flow of the fluid. In particular, reducing each pixel coordinate to a single scalar timestamp (see section 6.2.3) across the entire temporal domain covered by the dataset induces significant data loss when the fingers formed by the injected fluid detach from the inlet and form isolated regions that move independently through the porous medium.

While algorithms to handle these cases are discussed briefly in section 5.1.1, they have been omitted from the implementation, as the available datasets with capillary flow were recorded at an insufficient framerate to track fluid movement via spatial overlap across subsequent frames.

To handle such datasets in future improvements upon this algorithm, a heuristic approach could be employed, constructing local feature vectors for all regions in each frame, then matching nodes with similar features across adjacent frames. For instance, detached regions of non-wetting fluid retain their total visible area while moving, which makes the area a suitable feature to perform such a matching within the same dataset.

Bibliography

[ASY22] B. An, D. Solorzano, Q. Yuan. "Viscous Fingering Dynamics and Flow Regimes of Miscible Displacements in a Sealed Hele-Shaw Cell." In: *Energies* 15.16 (Aug. 2022), p. 5798. DOI: 10.3390/en15165798. URL: https://doi.org/10.3390/en15165798 (cit. on pp. 9, 15).

[BBDW16] F. Beck, M. Burch, S. Diehl, D. Weiskopf. "A Taxonomy and Survey of Dynamic Graph Visualization." In: *Computer Graphics Forum* 36.1 (Jan. 2016), pp. 133–159. DOI: 10.1111/cgf.12791. URL: https://doi.org/10.1111/cgf.12791 (cit. on p. 17).

[BDA+16] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, S. Carpendale. "A Descriptive Framework for Temporal Data Visualizations Based on Generalized Space-Time Cubes." In: *Computer Graphics Forum* 36.6 (Apr. 2016), pp. 36–61. DOI: 10.1111/cgf.12804. URL: https://doi.org/10.1111/cgf.12804 (cit. on p. 15).

[BKL+86] D. Bensimon, L. P. Kadanoff, S. Liang, B. I. Shraiman, C. Tang. "Viscous flows in two dimensions." In: *Reviews of Modern Physics* 58.4 (Oct. 1986), pp. 977–999. DOI: 10.1103/revmodphys.58.977. URL: https://doi.org/10.1103/revmodphys.58.977 (cit. on p. 25).

[BW08] L. Byron, M. Wattenberg. "Stacked Graphs - Geometry and Aesthetics." In: *IEEE Transactions on Visualization and Computer Graphics* 14.6 (Nov. 2008), pp. 1245–1252. DOI: 10.1109/tvcg.2008.166. URL: https://doi.org/10.1109/tvcg.2008.166 (cit. on p. 15).

[DiC13] D. A. DiCarlo. "Stability of gravity-driven multiphase flow in porous media: 40 Years of advancements." In: *Water Resources Research* 49.8 (Aug. 2013), pp. 4531–4544. DOI: 10.1002/wrcr.20359. URL: https://doi.org/10.1002/wrcr.20359 (cit. on pp. 9, 27).

[DNS86] G. Daccord, J. Nittmann, H. E. Stanley. "Radial viscous fingers and diffusion-limited aggregation: Fractal dimension and growth sites." In: *Physical Review Letters* 56.4 (Jan. 1986), pp. 336–339. DOI: 10.1103/physrevlett.56.336. URL: https://doi.org/10.1103/physrevlett.56.336 (cit. on p. 27).

[FFK+98] A. Fiat, D. P. Foster, H. Karloff, Y. Rabani, Y. Ravid, S. Vishwanathan. "Competitive Algorithms for Layered Graph Traversal." In: *SIAM Journal on Computing* 28.2 (Jan. 1998), pp. 447–462. DOI: 10.1137/s0097539795279943. URL: https://doi.org/10.1137/s0097539795279943 (cit. on p. 26).

[FSK+21] S. Frey, S. Scheller, N. Karadimitriou, D. Lee, G. Reina, H. Steeb, T. Ertl. "Visual Analysis of Two-Phase Flow Displacement Processes in Porous Media." In: (2021). DOI: 10.48550/ARXIV.2103.17197. URL: https://arxiv.org/abs/2103.17197 (cit. on pp. 9, 11, 14, 15, 19, 20, 41, 49, 51).

[GDB04] Y. Guéguen, L. Dormieux, M. Boutéca. "Fundamentals of Poromechanics." In: *Mechanics of fluid-saturated rocks* (2004), pp. 1–54 (cit. on p. 9).

[GKM+15] S. Grottel, M. Krone, C. Muller, G. Reina, T. Ertl. "MegaMol—A Prototyping Framework for Particle-Based Visualization." In: *IEEE Transactions on Visualization and Computer Graphics* 21.2 (Feb. 2015), pp. 201–214. DOI: 10.1109/tvcg.2014.2350479. URL: https://doi.org/10.1109/tvcg.2014.2350479 (cit. on p. 29).

[GKT+19] J. Gostick, Z. Khan, T. Tranter, M. Kok, M. Agnaou, M. Sadeghi, R. Jervis. "PoreSpy: A Python Toolkit for Quantitative Analysis of Porous Media Images." In: *Journal of Open Source Software* 4.37 (May 2019), p. 1296. DOI: 10.21105/joss.01296. URL: https://doi.org/10.21105/joss.01296 (cit. on p. 11).

[GRZ+10] S. Grottel, G. Reina, T. Zauner, R. Hilfer, T. Ertl. "Particle-based rendering for porous media." In: *Proceedings of SIGRAD 2010: Content aggregation and visualization; November 25–26; 2010; Västerås; Sweden*. 052. Linköping University Electronic Press. 2010, pp. 45–51 (cit. on p. 11).

[HKR93] D. Huttenlocher, G. Klanderman, W. Rucklidge. "Comparing images using the Hausdorff distance." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15.9 (1993), pp. 850–863. DOI: 10.1109/34.232073. URL: https://doi.org/10.1109/34.232073 (cit. on p. 37).

[Hom87] G. M. Homsy. "Viscous Fingering in Porous Media." In: *Annual Review of Fluid Mechanics* 19.1 (Jan. 1987), pp. 271–311. DOI: 10.1146/annurev.fl.19.010187.001415. URL: https://doi.org/10.1146/annurev.fl.19.010187.001415 (cit. on p. 27).

[JBT+18] M. Jiřik, M. Bartoš, P. Tomášek, A. Malečková, T. Kural, J. Horáková, D. Lukáš, T. Suchý, P. Kochová, M. H. Kalbáčová, M. Králičková, Z. Tonar. "Generating standardized image data for testing and calibrating quantification of volumes, surfaces, lengths, and object counts in fibrous and porous materials using X-ray microtomography." In: *Microscopy Research and Technique* 81.6 (Feb. 2018). Ed. by G. Perry, pp. 551–568. DOI: 10.1002/jemt.23011. URL: https://doi.org/10.1002/jemt.23011 (cit. on p. 11).

[Kac08] M. Kaczmarek. "Bio-Poromechanics. Problems Of Modelling Tissues And Biomaterials." In: *Selected Topics of Contemporary Solid Mechanics* (2008), p. 4 (cit. on p. 9).

[Kie16] S. Kieß. *Voxie voxel viewer*. June 2016. URL: https://github.com/voxie-viewer/voxie (cit. on p. 11).

[Koc19] S. Koch. *Lecture notes: Information Visualization and Visual Analytics*. Jan. 2019 (cit. on pp. 15, 23).

[MMKN08] P. McLachlan, T. Munzner, E. Koutsofios, S. North. “LiveRAC: Interactive Visual Exploration of System Management Time-Series Data.” In: *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI ’08*. ACM Press, 2008. DOI: 10.1145/1357054.1357286. URL: https://doi.org/10.1145/1357054.1357286 (cit. on p. 16).

[NBK13] D. Naumov, L. Bilke, O. Kolditz. “Rendering Technique of Multi-layered Domain Boundaries and its Application to Fluid Flow in Porous Media Visualizations.” In: *Workshop on Visualisation in Environmental Sciences (EnvirVis)*. Ed. by O. Kolditz, K. Rink, G. Scheuermann. The Eurographics Association, 2013. ISBN: 978-3-905674-54-5. DOI: 10.2312/PE.EnvirVis.EnvirVis13.043-046 (cit. on p. 11).

[RSK20] S. Roman, C. Soulaine, A. R. Kovscek. “Pore-scale visualization and characterization of viscous dissipation in porous media.” In: *Journal of Colloid and Interface Science* 558 (Jan. 2020), pp. 269–279. DOI: 10.1016/j.jcis.2019.09.072. URL: https://doi.org/10.1016/j.jcis.2019.09.072 (cit. on p. 24).

[SL97] D. J. Simons, D. T. Levin. “Change blindness.” In: *Trends in Cognitive Sciences* 1.7 (Oct. 1997), pp. 261–267. DOI: 10.1016/s1364-6613(97)01080-2. URL: https://doi.org/10.1016/s1364-6613(97)01080-2 (cit. on pp. 16, 23).

[ST58] P. G. Saffman, G. I. Taylor. “The penetration of a fluid into a porous medium or Hele-Shaw cell containing a more viscous liquid.” In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 245.1242 (June 1958), pp. 312–329. DOI: 10.1098/rspa.1958.0085. URL: https://doi.org/10.1098/rspa.1958.0085 (cit. on p. 25).

[TKE12] R. M. Tarawaneh, P. Keller, A. Ebert. “A General Introduction To Graph Visualization Techniques.” In: *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering - Proceedings of IRTG 1131 Workshop 2011*. Ed. by C. Garth, A. Middel, H. Hagen. Vol. 27. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 151–164. ISBN: 978-3-939897-46-0. DOI: 10.4230/OASIcs.VLUDS.2011.151. URL: http://drops.dagstuhl.de/opus/volltexte/2012/3748 (cit. on p. 16).

[UCH04] F. Ulm, G. Constantinides, F. Heukamp. “Is concrete a poromechanics materials? A multiscale investigation of poroelastic properties.” In: *Materials and Structures* 37.1 (Jan. 2004), pp. 43–58. DOI: 10.1007/bf02481626. URL: https://doi.org/10.1007/bf02481626 (cit. on p. 9).

[WDH+22] F. Weinhardt, J. Deng, J. Hommel, S. V. Dastjerdi, R. Gerlach, H. Steeb, H. Class. “Spatiotemporal Distribution of Precipitates and Mineral Phase Transition During Biomineralization Affect Porosity–Permeability Relationships.” In: *Transport in Porous Media* 143.2 (May 2022), pp. 527–549. DOI: 10.1007/s11242-022-01782-8. URL: https://doi.org/10.1007/s11242-022-01782-8 (cit. on pp. 9, 20).

[WE13] M. Wörner, T. Ertl. “SmoothScroll: A Multi-scale, Multi-layer Slider.” In: *Communications in Computer and Information Science*. Springer Berlin Heidelberg, 2013, pp. 142–154. DOI: 10.1007/978-3-642-32350-8_9. URL: https://doi.org/10.1007/978-3-642-32350-8_9 (cit. on p. 16).

[YKZS21] A. Yiotis, N. K. Karadimitriou, I. Zarikos, H. Steeb. “Pore-scale effects during the transition from capillary- to viscosity-dominated flow dynamics within microfluidic porous-like domains.” In: *Scientific Reports* 11.1 (Feb. 2021). DOI: 10.1038/s41598-021-83065-8. URL: https://doi.org/10.1038/s41598-021-83065-8 (cit. on pp. 24, 25).

All links were last followed on August 29, 2022.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature