Visualizing Time-Varying Topological Structures

A PROJECT REPORT
SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
Master of Technology
IN
Faculty of Engineering

BY

Toshit Bazaz

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

July, 2021

Declaration of Originality

I, **Toshit Bazaz**, with SR No. **04-04-00-10-42-19-1-17396** hereby declare that the material presented in the thesis titled

Visualizing Time-Varying Topological Structures

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2020-21**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: 10-07-2021 **Student Signature**

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Vijay Natarajan **Advisor Signature**

DEDICATED TO

The Student Community

who can use and reuse this template to glory

Acknowledgements

I am filled with gratitude to all those who helped me in completion of the project. The most pleasant point of presenting a project is the opportunity to thank those who have contributed to it. The list of all those to whom I extend my sincere thanks and acknowledgement is too long. Indeed, this page of acknowledgement shall never be able to touch the horizons of generosity of those who tendered their help to me.

First and foremost, I would like to express my sincerest gratitude and indebtedness to Prof. Vijay Natarajan, for his kindness in allowing me to introduce the present topic and his inspiring guidance, constructive criticism and valuable suggestion throughout the project. I am sincerely thankful to him for his guidance and pain-staking effort in improving my understanding of this project.

I would also like to thank Somenath Das for his unrelenting support.

Thanks are due to all the supporting staff of the department who have at all times during the project work in particular and degree programme in general provided me their required support.

Last, but not the least, my sincere thanks to all my friends who have patiently extended all sorts of help for accomplishing this undertaking.

Toshit Bazaz
SR No.: 04-04-00-10-42-19-1-17396
M.Tech in Computer Science Automation

Abstract

Time varying structure visualization focuses on the challenge of representing the evolution of relationships between entities in readable, scalable, and effective diagrams. In dynamic graph visualization, changes to mental map should be minimal. To this end, the position of nodes is tried to be kept stable, which is called dynamic stability or drawing stability. In this project, we present a methodology to visualize, interact and query time varying graphs efficiently. We use time varying extremum graph and time varying temporal correspondence graph for visualization. We use result of the queries to find some interesting information from time varying graphs like longest track/(s) in data, missing edges, formation of fingers etc.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

The motivation for this project is to have an interactive tool that tracks maxima and outputs longest tracks and finding missing tracks. Graph visualization is concerned with visual representations of graph or network data. Effective graph visualization reveals structures that may be present in the graphs and helps the users to understand and analyze the underlying data. Dynamic graph visualization is used for representing the evolution of relationships between entities in readable, scalable, and effective diagrams. In dynamic graph visualization, changes to mental map should be minimal. To this end, the position of nodes is tried to be kept stable, which is called dynamic stability or drawing stability.

Mental map refers to the abstract structural information a user forms by looking at the layout of a graph.

Extremum graph [3] is a natural way to connect critical points. It represents ridge-like connections between extrema (maxima in our case) as given by the Morse complex. Even in its abstract form the extremum graph preserves much of the local structure and overall shape of the original features. Time varying extremum graph is a graph that contains extremum graph for all the time frames of a dynamic graph.

Viscous fingering[7] is an instability phenomenon that occurs in porous media at the interface between two fluids of distinct viscosity. In particular, it appears when a less viscous fluid is injected within a more viscous one. The problem statement is to create a full fledged tool for visual exploration of time varying graph with support for overview visualization, querying, interactive exploration.

In this project, we introduce a tool for visualization of raw data file, time varying extremum graph and time varying correspondence graph for time varying data. The tool also has support for tracking maxima(s), longest track(s), finding missing edges and visualizing fingers.

1.2 Related Work

Graphs in general form one of the most important data models in computer science because many problems and domains can be modeled as graph structures.In most of the applications, temporal development can be observed and needs to be considered to fully understand the respective problem. Visualization is a particular means for exploratively comprehending and analyzing this data. A graph consists of objects or entities, usually referred to as vertices, and relationships between them, called edges. Other publications have already partly reviewed the field of visualizing dynamic graphs. In 2011, Correa [3] introduced the concept of topological spines, a way to represent extremum graphs. In 2003, Erten [5] introduced the concept a graphAEL, a Graphical User Interface for interactive visualization of evolving graphs like citation graphs, topic graphs and collaboration graphs. In 2001, Diel [4] introduced two algorithms Graph Animation Partitioning(GAP) and Reduced Graph Animation Partitionings(RGAP) for partitioning data and preserving mental map using a foresighted layout. In 2016, Beck [1] did a survey for different visualizations for dynamic graphs. In 2005, Tzeng [8] introduced a machine learning model for intelligent feature extraction and they show the results using cumulative histograms. A neural network is used for generating the adaptive transfer function with a set of inputs and desired outputs. They apply the model on argon bubble data and swirling flow data set. In 2010, Bremer [2] introduced an algorithm for tracking burning structures in hydrogen flames. They analyze the burning structues and create reeb graphs. They use these reeb graphs to track burning. In result they shows graphs showing relation between number of cells and diferent parameters like time, persistance etc. In 2015, Wathsala Widanagamaachchi [9] a method to track extinction in turbulent surface. They use isovolumes of turbulent surface to create merge trees. They apply temporal artifact reduction on merge trees and use the resulting graphs for tracking purpose. In this tool, we provide support for visualizing time varying graphs. We introduce the algorithm for tracking maxima and computing different longest paths in the graph.

Chapter 2

Algorithm for longest Path

In this chapter, we are introducing the algorithm used for computing longest paths. Time complexity is O(n^2). This algorithm tracks all the paths and then selects longest path among them. We are need two arrays for computing the paths. One that holds visited maxima and one that holds their lifetime. We select that maxima from maxima array that has largest lifetime at the end of algorithm.

```
Create an empty array maxima = []
Create an empty array lifetime = []
Initialize i = 0
Initialize max lifetime = 0
Initialize max lifetime id = 0
Initialize max lifetime dead = 0
Initialize max lifetime dead id = 0
#Open file temporal i
file = open(temporal i)
while True:
  #To read maxima from file
  line = file.readline()
  if not line:
    break
  maxima.append(line)
  lifetime.append(1)
max lifetime += 1
for i in range(1,n):
  #Open file temporal i
  file = open(temporal i)
  while True:
    #To read maxima from file
    line = file.readline()
    if not line:
      break
    #If there is no predecessor in last time frame
    If maxima takes birth in this time frame:
      maxima.append(line)
      lifetime.append(1)
    #If there is no successor in next time frame
    else if maxima dies in this frame:
      if lifetime[line] ≥ max lifetime:
        if lifetime[line] ≥ max lifetime dead:
          max lifetime dead = lifetime[line]
          max lifetime dead id = line
      Remove maxima from maxima array and
      lifetime array
    #If maxima has predecessor and successor
    else:
      Increment lifetime predecessor by one
      Update max lifetime
      Update max lifetime id
    if max lifetime > max lifetime dead:
      max lifetime dead = 0
      max lifetime dead id = 0
if max lifetime ≥ max lifetime dead:
  Create longest track with max lifetime id as start
  and of lifetime max lifetime
else:
  Create longest track with max lifetime dead id as
  start and of lifetime max lifetime dead
```

Figure 2.1: Algorithm for Longest Track

Chapter 3

Framework

In the chapter, we are describing he framework created for the project. RenderView1 is used to display raw data files. RenderView3 is used to display time varying extremum graph. RenderView4 is used to display time varying temporal correspondence graph. RenderView4 is used to display result of all the queries discussed in interaction section. For queries of maxima selection, the user has to select the maxima from RenderView4.

3.1 Gaussian data

The framework for gaussian data is as shown in figure 3.1.

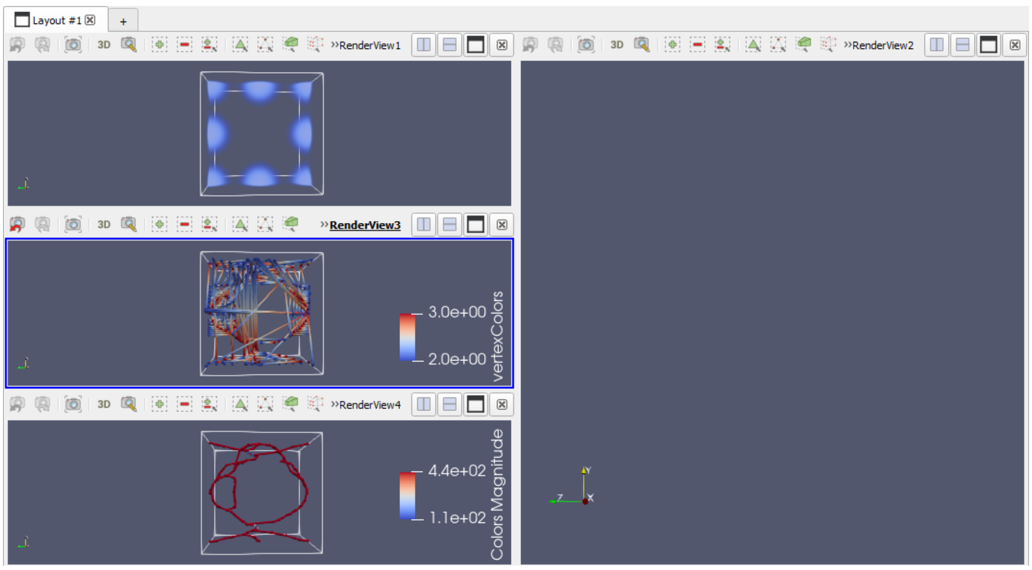

Figure 3.1: Framework for gaussian data

3.2 Viscous finger data

The framework for viscous finger data is as shown in figure 3.2.

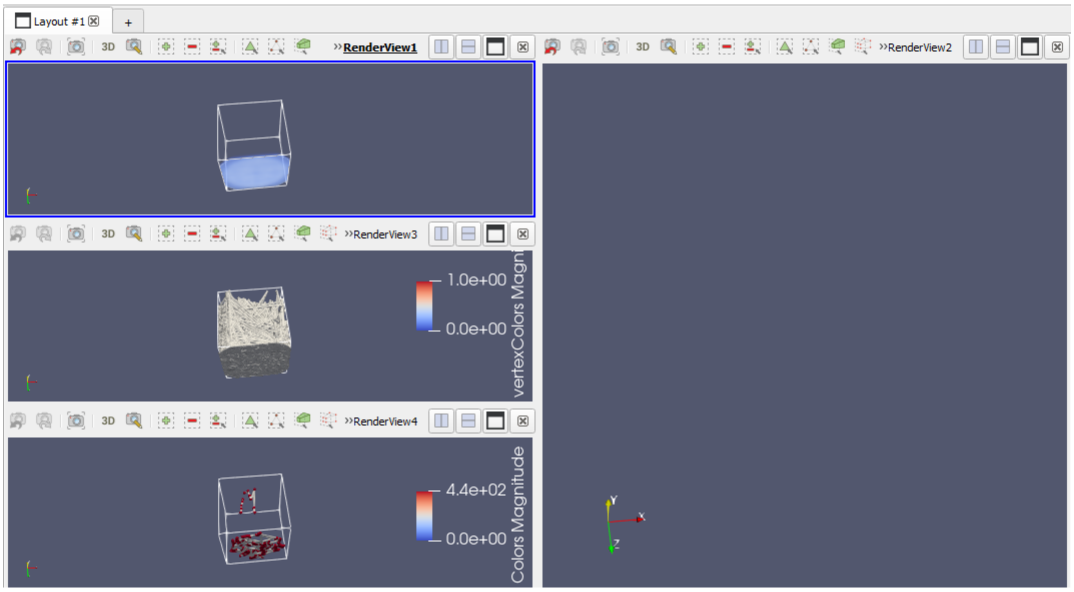

Figure 3.2: Framework for viscous finger data

Chapter 4

Data

In this chapter, we are describing the data along with the pre- processing done on the data and how we are visualizing the processed data for the project.

4.1 Gaussian data

This is synthetic data that we use to illustrate and test the visualization techniques. The Gaussian data is defined within a domain of dimension $128 \times 128 \times 128$. Within the domain eight Gaussian blobs of equal volume are defined on the z=0 plane. These blobs are moved along four parabolic splines. These paths are defined using cubic splines [6]. Each of the 50-time steps are result of these Gaussian blobs moving along these splines one step. The dataset is 3D imagefiles of the gaussian data.
Outline filter is used on data. Volume rendering for first frame is shown in Figure 4.1.

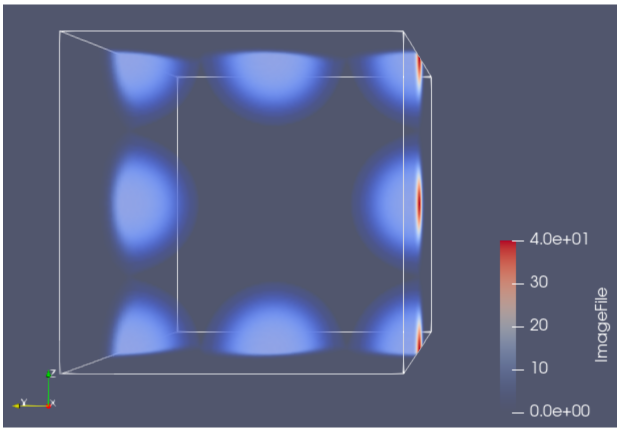

Figure 4.1: Volume rendering of first frame of the gaussian data

The datatype is unsigned int and data byte order is little endian. The color map used is cool to warm(or cyan to red; where cyan corresponds to 0 value and red corresponds to 40 value).

4.1.1 Extremum graph

This is synthetic data that contains information about the extremum points(maxima and saddles) of gaussian data. The dataset is 50 text files each corresponding to extremum graph for one frame of gaussian data.Each text file contains 4 columns: First 3 columns corresponds to x, y and z coordinates of maximas and saddles and 4th column contains an identifier specifying if the coordinate is a maxima or a saddle.
A python code is used to convert these text files into vtp files. The surface representation for first frame is shown in Figure 4.2.
The vertices here represent maxima and saddles and the edges are connecting a maxima to saddle that is connected to another maxima. The color map used is similar to gaussian data with cyan corresponding to saddle(2.0e+00) and to red corresponding to maxima(3.0e+00). The points are render as spheres of size 5px and lines are rendered as tubes of width 3px.Each edge connects a maxima and a saddle.

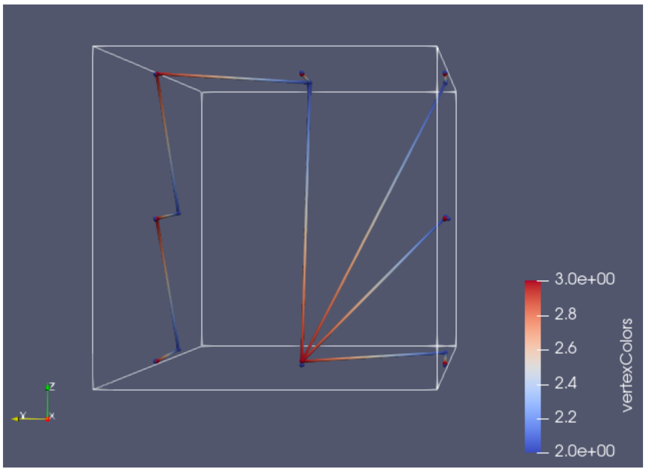

Figure 4.2: Extremum graph corresponding to first frame of Gussian Data

4.1.2 Time varying extremum graph

Group time steps filter is used on extremum graph to show extremum graphs for all 50 frames in a single frame. The surface representation for time varying extremum graph is shown in Figure 4.3.

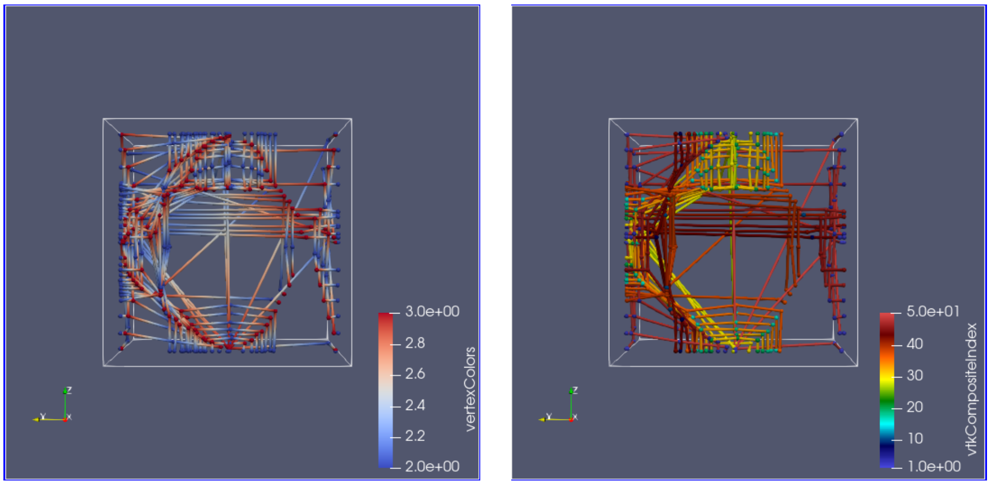

Figure 4.3: Time varying extremum graphs with different color maps (a)The graph on left is time varying extremum graph using cool to warm color map. (b)The graph on right is time varying extremum graph using rainbow desaturated color map.

The color map for graph on left varies from cyan corresponding to saddle(2.0e+00) and to red corresponding to maxima(3.0e+00). The points are render as spheres of size 5px and lines are rendered as tubes of width 3px.

The color map for graph on right represents range of diferent time frames; blue corresponding to extremum graphs for frames 1 to 10 and cyan corresponding to extremum graphs for frames 11 to 20 and so on. The points are render as spheres of size 5px and lines are rendered as tubes of width 3px.

4.1.3 Temporal correspondence

This is synthetic data that contains information about the temporal correspondence edges between adjacent frames. We are only considering correspondence edges between maxima so that we can track the progress of maxima about time.

The data provided is 49 text files each containing details of temporal correspondence edges between adjacent frames. Each text file contains 4 columns: First 3 columns contain x, y and z coordinates of maximas and 4th column contains an identifier specifying which frame the maxima belongs to. A python code is used to convert these text files into vtp files. The surface representation for first correspondence frame is shown in Figure 4.4.

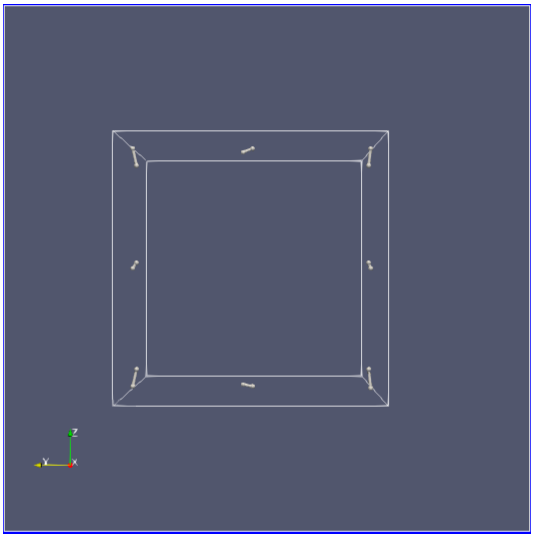

Figure 4.4: Temporal correspondence between frame 1 and 2 of gaussian data

The vertices here represent maxima and the edges are temporal edges that are connecting maxima from one frame to maxima of adjacent frame. The points are render as spheres of size 5px and lines are rendered as tubes of width 3px.

4.1.4 Time varying temporal correspondence

Group time steps filter is used on temporal correspondence graph to show temporal edges for all 49 frames in a single frame. The surface representation for time varying temporal correspondence is shown in the Figure 4.5.

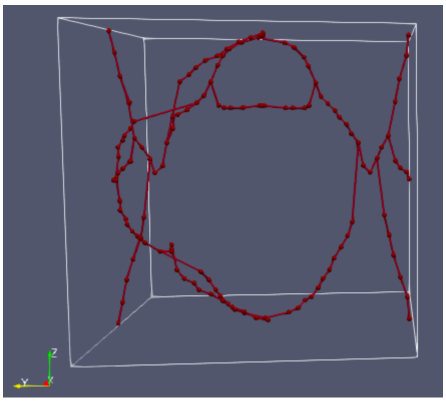

Figure 4.5: Time varying temporal correspondence graph

The points are render as spheres of size 5px and lines are rendered as tubes of width 3px.

4.2 Viscous Finger data

This is synthetic data that we use to illustrate and test the visualization techniques. The Viscous Finger data is defined within a domain of dimension $101 \times 101 \times 101$. The data shows the formation of viscous fingers in data. These formations are similar to what are defined in [6]. Each of the 120-time steps are result of formtion of different fingers. The dataset is 3D imagefiles of the viscous finger data.
Outline filter is used on data. Volume rendering for first frame is shown in Figure 4.6.
The datatype is float and data byte order is little endian. The color map used is cool to warm.

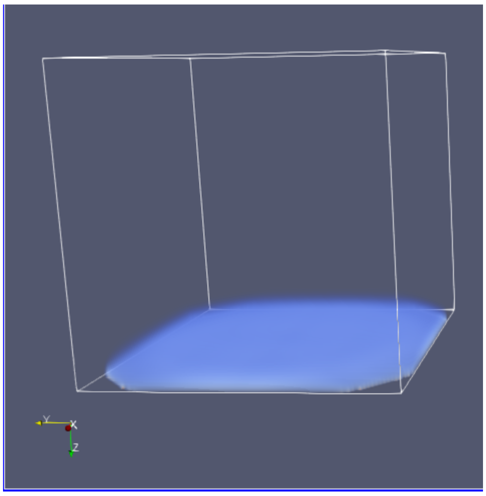

Figure 4.6: Volume rendering of first frame of the viscous data

4.2.1 Extremum graph

This is synthetic data that contains information about the extremum points(maxima and saddles) of viscous finger data. The dataset is 120 text files each corresponding to extremum graph for one frame of viscous finger data. Each text file contains 4 columns: First 3 columns corresponds to x, y and z coordinates of maximas and saddles and 4th column contains an identifier specifying if the coordinate is a maxima or a saddle.
A python code is used to convert these text files into vtk files. The surface representation for first frame is shown in Figure 4.7.
The vertices here represent maxima and saddles and the edges are connecting a maxima to saddle that is connected to another maxima. The points are render as spheres of size 5px and

lines are rendered as tubes of width 3px.Each edge connects a maxima and a saddle.

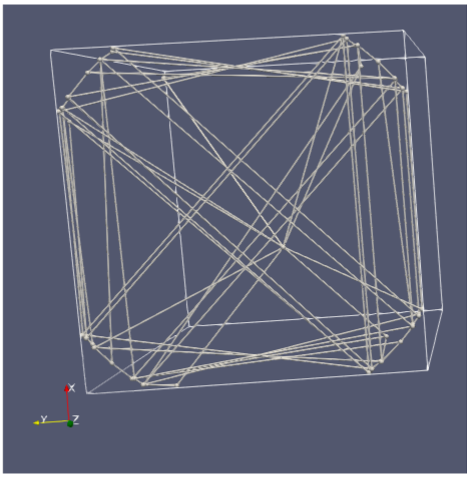

Figure 4.7: Extremum graph corresponding to first frame of Viscous Finger Data

4.2.2 Time varying extremum graph

Group time steps filter is used on extremum graph to show extremum graphs for all 120 frames in a single frame. The surface representation for time varying extremum graph is shown in Figure 4.8.

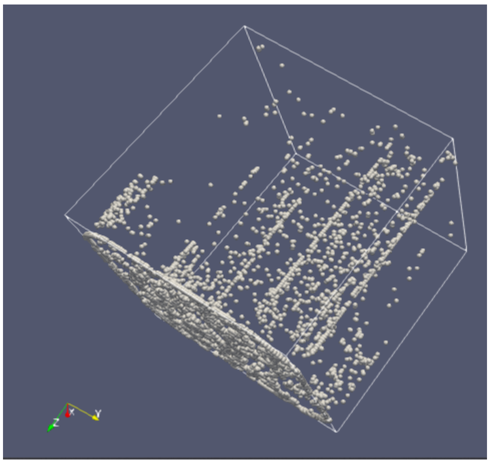

Figure 4.8: Time varying extremum graph

The points are render as spheres of size 5px and lines are rendered as tubes of width 3px.

4.2.3 Temporal correspondence

This is synthetic data that contains information about the temporal correspondence edges between adjacent frames. We are only considering correspondence edges between maxima so that we can track the progress of maxima about time.

The data provided is 119 text files each containing details of temporal correspondence edges between adjacent frames. Each text file contains 4 columns: First 3 columns contain x, y and z coordinates of maximas and 4th column contains an identifier specifying which frame the maxima belongs to. The values vary from 0.5 to 99.5 in both x and y direction. The concentration of maxima is very high at the ends. So, only the set of maxima lying within the circle of radius 43 with (50,50) as center on x,y plane are used for temporal correspondence. A python code is used to get these maxima to convert them into vtk files. The surface representation for first correspondence frame is shown in Figure 4.9.

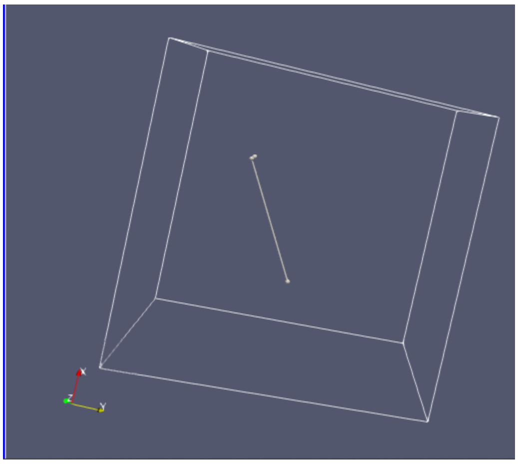

Figure 4.9: Temporal correspondence between frame 12 and 13 of viscous finger data

The vertices here represent maxima and the edges are temporal edges that are connecting maxima from one frame to maxima of adjacent frame. The points are render as spheres of size 5px and lines are rendered as tubes of width 3px.

4.2.4 Time varying temporal correspondence

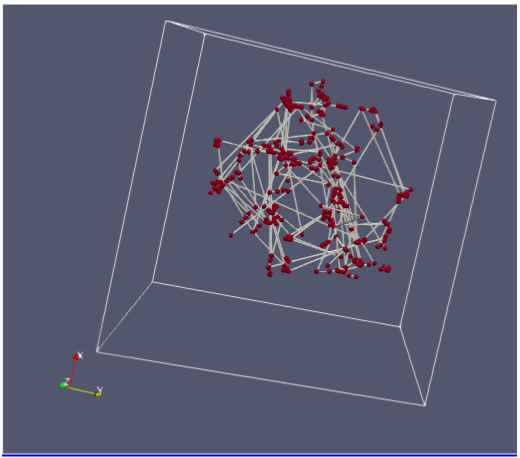

Figure 4.10: Time varying temporal correspondence graph

Group time steps filter is used on temporal correspondence graph to show temporal edges for all 119 frames in a single frame. The surface representation for time varying temporal correspondence is shown in the Figure 4.10.

The points are render as spheres of size 5px and lines are rendered as tubes of width 3px.

Chapter 5

Interaction

In this chapter, we are describing the different types of user interactions possible in the project

5.1 Single maxima selection

In this mode, a user can press on any maxima and the maxima will be tracked. The extremum graph of the time frame the maxima belong to and longest track that starts from the maxima will be displayed in a new renderview. For this, we will search the coordinates of the maxima in the temporal correspondence data that will return the time frame this maxima belong to along with the temporal track. We store the temporal correspondence in an array. We will stop if we either reach the last frame or if there is no temporl track for the maxima.

5.1.1 Single maxima selection in gaussian data

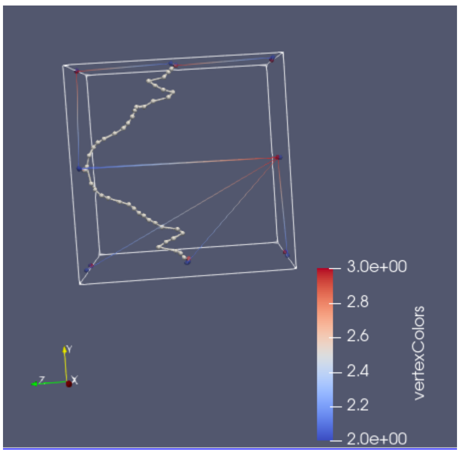

Figure 5.1: Single maxima selection in gaussian data

The representation for one Single maxima selection in gaussian data is shown in Figure 5.1.

5.1.2 Single maxima selection in viscous finger data

The representation for one Single maxima selection in viscous finger data is shown in Figure 5.2.

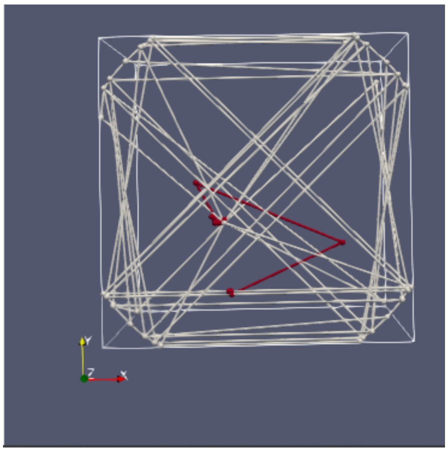

Figure 5.2: Single maxima selection in viscous finger data

5.2 Multiple maxima selection

In this mode, a user can select a set of maxima by creating a quadrilateral and all the maxima that lie in the quadrilateral will be tracked. The extremum graph/(s) corresponding to these maxima and the longest tracks starting from each maxima will be displayed in a new renderview. For this, we will search the coordinates of the maxima in the temporal correspondence data that will return the time frame each of these maxima belong to along with the temporal tracks for each maxima. We store the temporal correspondence in a 2d array. We will stop if we either reach the last frame or if there is no temporal track left for the any maxima.

5.2.1 Multiple maxima selection in gaussian data

The representation for one mutiple maxima selection in gaussian data is shown in Figure 5.3.

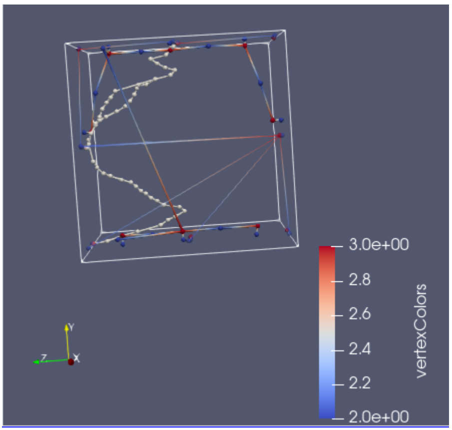

Figure 5.3: multiple maxima selection in gaussian data

5.2.2 Multiple maxima selection in viscous finger data

The representation for Multiple maxima selection in viscous finger data is shown in Figure 5.4.

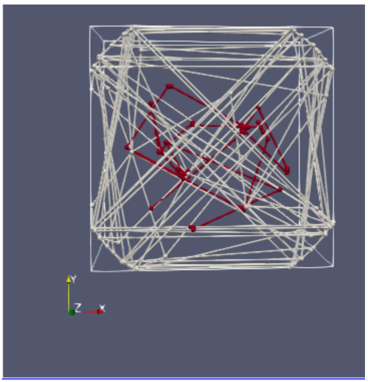

Figure 5.4: Multiple maxima selection in viscous finger data

5.3 Frame selection

In this mode, a user will be asked to input a time frame number and the exremum graph for the time frame will be displayed along with the temporal tracks for all the maxima belonging to the selected time frame for next five time steps. For this, we will select input frame from extremum graph data and temporal correspondence data and we will get coordinates of the all maxima belonging to the frame along with the temporal tracks for each maxima. We store the temporal correspondence in a 2d array. We will stop if we either reach the last frame or if there is no temporal track left for the any maxima or after five time steps.

5.3.1 Frame selection in gaussian data

The representation for frame selection in gaussian data is shown in Figure 5.5.

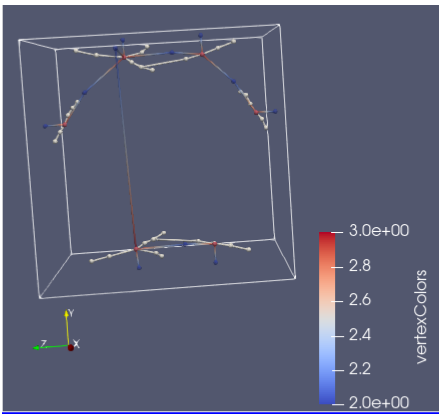

Figure 5.5: Frame selection in gaussian data

5.3.2 Frame selection in viscous finger data

The representation for Frame selection in viscous finger data is shown in Figure 5.6.

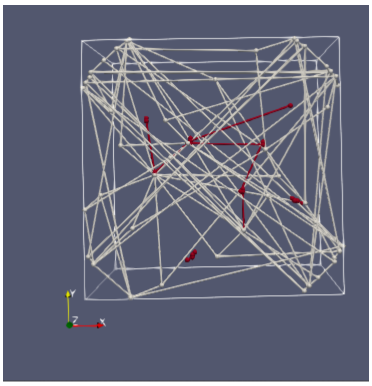

Figure 5.6: Frame selection in viscous finger data

5.4 Longest tracks

In this mode, the longest track/(s) among all the tracks will be displayed in new renderview. For this, we will start tracking all the maxima from frame one of temporal correspondence data. We store the lifetime of each maxima in an array along with max lifetime among all alive maxima. We will stop if we reach the last frame.

5.4.1 Longest track in gaussian data

The representation for longest in gaussian data is shown in Figure 5.7.

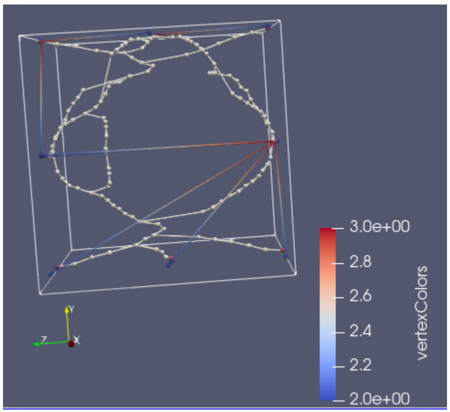

Figure 5.7: Longest track in gaussian data

5.4.2 Longest Track in viscous finger data

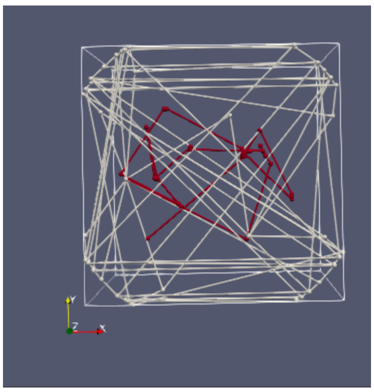

Figure 5.8: Longest track in viscous finger data

The representation for longest track in viscous finger data is shown in Figure 5.8.

5.5 Longest track after some time frame

In this mode, the user will be asked for a time frame number and the longest track/(s) among all the tracks that appear this time frame will be displayed in new renderview. For this, we will start tracking all the maxima from the input time frame of temporal correspondence data. We store the lifetime of each maxima in an array along with max lifetime among all alive maxima. We will stop if we reach the last frame.

5.5.1 Longest track after some time frame in gaussian data

The representation for longest after some time frame in gaussian data is shown in Figure 5.9.

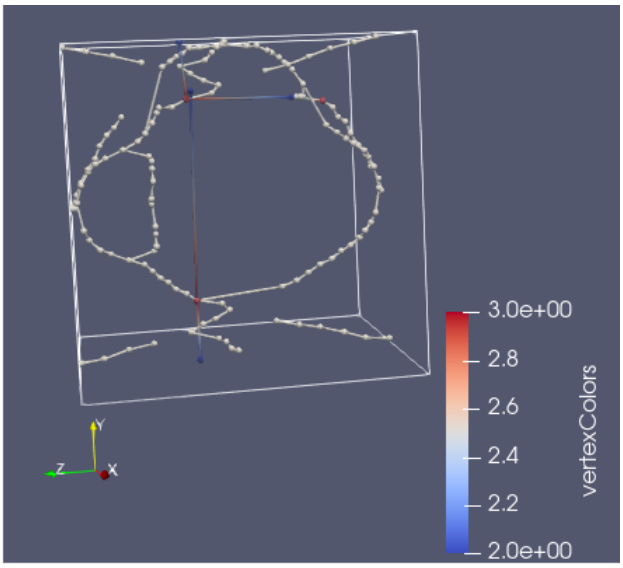

Figure 5.9: Longest track after some time frame in gaussian data

5.5.2 Longest Track after some time frame in viscous finger data

The representation for longest track after some time frame in viscous finger data is shown in Figure 5.10.

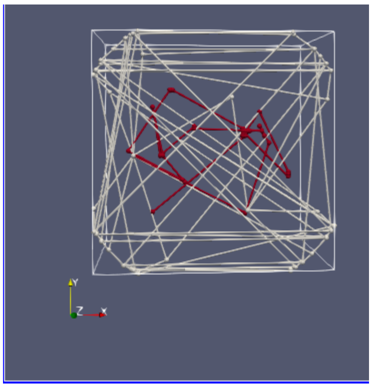

Figure 5.10: Longest track after some time frame in viscous finger data

5.5.3 Longest track in some range of time frame

In this mode, the user will be asked two time frame number, one corresponding to the starting and other corresponding to the ending of range and the longest track/(s) among all the tracks that appear in this time range will be displayed in new renderview. For this, we will start tracking all the maxima from the first input time frame of temporal correspondence data. We store the lifetime of each maxima in an array along with max lifetime among all alive maxima. We will stop if we reach the second input time frame.

5.5.4 Longest track in some range of time frame in gaussian data

The representation for longest in some range of time frame in gaussian data is shown in Figure 5.11.

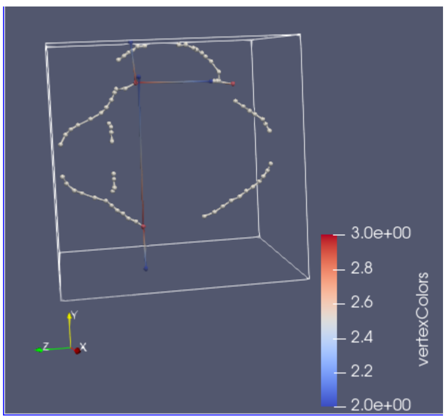

Figure 5.11: Longest track in some range of time frame in gaussian data

5.6 Fingers only

In this mode, only the fingers will be shown in a new renderview. For this, we will start from frame one of extremum graph for viscous finger data and display all the extrema,along with their edges, that do not lie on xy plane. We will stop if we reach the last frame.

5.6.1 Fingers only for viscous finger data

The representation for fingers only for viscous finger data is shown in Figure 5.12.

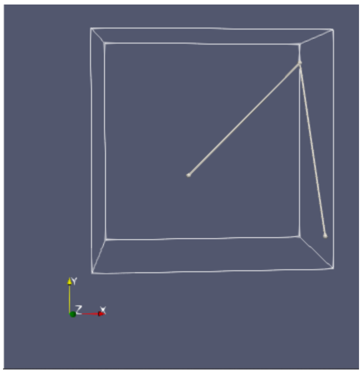

Figure 5.12: Fingers only for viscous finger data

5.7 Missing edges

Due to using only a subset of maxima in temporal correspondence data for viscous fingers data, there are some frames that do not have any temporal correspondence edges. Due to which there are disconnections in time varying temporal correspondence graph for viscous finger data. So in this mode, the time temporal correspondence graph along with missing correspondence edges for viscous finger data will be shown in a new renderview. For this, we will start from frame one of temporal correspondence data for viscous finger and display all the temporal edges. In case there is no temporal edge, we will add edges from all maxima in last frame to all maxima in next frame. We will stop if we reach the last frame.

5.7.1 Missing edges only for viscous finger data

The representation for missing edges only for viscous finger data is shown in Figure 5.13.

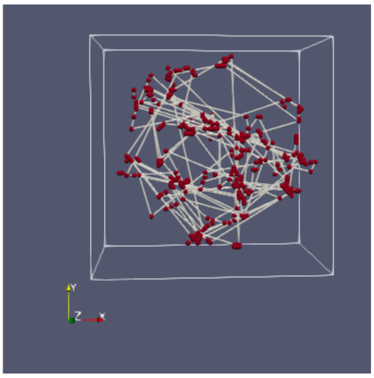

Figure 5.13: Missing edges for viscous finger data

Chapter 6

Implementation

Paraview 5.8.1 is used for visualizing data.
ParaView is an open-source, multi-platform data analysis and visualization application. ParaView users can quickly build visualizations to analyze their data using qualitative and quantitative techniques. The data exploration can be done interactively in 3D or programmatically using ParaView's batch processing capabilities.
ParaView was developed to analyze extremely large datasets using distributed memory computing resources. It can be run on supercomputers to analyze datasets of petascale size as well as on laptops for smaller data, has become an integral tool in many national laboratories, universities and industry, and has won several awards related to high performance computation.

Python 3.7.4 is used for coding purposes.
Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.
Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection

of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Paraview uses vtk file format to visualize data. vtk 9.0.1 is used to convert text file format into vtk file format. The Visualization Toolkit (VTK) is open source software for manipulating and displaying scientific data. It comes with state-of-the-art tools for 3D rendering, a suite of widgets for 3D interaction, and extensive 2D plotting capability.
VTK is part of Kitware's collection of supported platforms for software development. The platform is used worldwide in commercial applications, as well as in research and development. For examples, please see VTK in Action.

All the codes and data are available at Github

Chapter 7

Conclusion

This is a full fledged tool with the ability to visualize any time varying graph. This tool has support for interactive exploration like maxima selection. This tool also has support for queries like longest path, finding missing edges, finding viscous fingers, partitioning of data on the basis of time frames as well as spatial selection.

Bibliography

[1] Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. A taxonomy and survey of dynamic graph visualization. *Comput. Graph. Forum*, 36(1):133–159, January 2017. ISSN 0167-7055. doi: 10.1111/cgf.12791. URL https://doi.org/10.1111/cgf.12791. 2

[2] Peer-Timo Bremer, Gunther Weber, Valerio Pascucci, Marc Day, and John Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):248–260, 2010. doi: 10.1109/TVCG.2009.69. 2

[3] C. Correa, P. Lindstrom, and P. Bremer. Topological spines: A structure-preserving visual representation of scalar fields. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1842–1851, 2011. doi: 10.1109/TVCG.2011.244. 1, 2

[4] Stepahn Diel, Carsten Görg, and Andreas Kerren. Preserving the mental map using foresighted layout. In *Proceedings of the 3rd Joint Eurographics - IEEE TCVG Conference on Visualization*, EGVISSYM'01, page 175–184, Goslar, DEU, 2001. Eurographics Association. ISBN 3211836748. 2

[5] Cesim Erten, Philip Harding, Stephen Kobourov, Kevin Wampler, and Gary Yee. Graphael: Graph animations with evolving layouts. volume 2912, pages 98–110, 09 2003. ISBN 978-3-540-20831-0. doi: 10.1007/978-3-540-24595-7_9. 2

[6] Gerald E Farin and Gerald Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002. 6, 10

[7] Jonas Lukasczyk, Garrett Aldrich, Michael Steptoe, Guillaume Favelier, Charles Gueunet, Julien Tierny, Ross Maciejewski, Bernd Hamann, and Heike Leitte. Viscous fingering: A topological visual analytic approach. *Applied Mechanics and Materials*, 869:9–19, 08 2017. doi: 10.4028/www.scientific.net/AMM.869.9. 1

[8] Fan-Yin Tzeng and Kwan-Liu Ma. Intelligent feature extraction and tracking for visualizing large-scale 4d flow simulations. In *SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, pages 6–6, 2005. doi: 10.1109/SC.2005.37. 2

[9] Wathsala Widanagamaachchi, Jackie Chen, Pavol Klacansky, Valerio Pascucci, Hemanth Kolla, Ankit Bhagatwala, and Peer-Timo Bremer. Tracking features in embedded surfaces: Understanding extinction in turbulent combustion. In *2015 IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 9–16, 2015. doi: 10.1109/LDAV.2015.7348066. 2