5-2020

High Dimensional Event Exploration Over Multiple Simulations

Steven Deron Scott
Utah State University

HIGH DIMENSIONAL EVENT EXPLORATION OVER MULTIPLE SIMULATIONS

by

Steven Deron Scott

Capstone submitted in partial fulfillment of the requirements for graduation with

University Honors

with a major in

Computer Science

in the Department of Computer Science

Approved:

Capstone Mentor Dr. John Edwards	**Departmental Honors Advisor** Dr. Dan Watson
Committee Member [optional: type name or N/A] N/A	**University Honors Program Director** Dr. Kristine Miller

UTAH STATE UNIVERSITY
Logan, UT

Spring 2020

Acknowledgements

Special thanks to Dr. John Edwards his mentorship throughout this project.

Special thanks to Jaxon Willard for his help, particularly his work on z-ordering and Hilbert curves.

Special thanks to Dan Watson for overseeing the honors students in the Computer Science Department.

Thanks also to the Computer Science Department, Honors Program, College of Science, and Utah State University.

Abstract

In this project, we introduce a visualization technique to analyze event simulation data. In particular, we allow the user to discover families of events based on the topological evolution of discrete events across simulations. Discovering how events behave across runs of a simulation has applications in financial market analysis, military simulations, physical mechanics, and other settings. Our approach is to use established methods to produce a linearized tour through parameter space of arbitrary dimension and visualize events of interest in two dimensions, where the first dimension is the tour ordering and the second dimension is usually time. This paper presents our approach and gives examples in the context of a magnet dynamics simulation.

Contents

High Dimensional Event Exploration Over Multiple Simulations

April 27, 2020

1 Introduction

1.1 Simulations With Discrete Events

Computer simulations are used in many contexts to model how a system will act under a set of circumstances. For instance, climate simulations show how earth systems might respond to increased levels of greenhouse gases. Stock market simulations show how a particular trading algorithm would have performed during a period of historical data. Over time, simulations have been increasing in complexity, with more factors considered in the simulation, more runs of each simulation, greater output resolution, and greater output complexity. Throughout this project, we used a simulation of two magnets interacting in a frictionless environment.

Different simulations will produce different types of output. Some simulations will produce a system state - for instance, a climate model may show how much sea ice remains under a certain warming scenario. Other simulations produce a series of discrete events occurring over time. For example, a stock market trading simulation would likely produce a series of buy and sell events over the course of a market simulation. Our visualization is designed to analyze simulations that produce discrete events, like a market trading algorithm. Other applications include defense system analysis and physics simulations. Many approaches exist for analyzing and visualizing data from scalar functions, but we know of no existing approaches for exploring discrete event families across simulations.

1.2 Input Parameters and Event Families

Comparing simulations is only useful if the simulations vary in some meaningful way. This usually means that the input parameters vary, producing varied output. These input parameters are application-specific – for a momentum-based trading algorithm, an input parameter might be the percentage change in stock price necessary to trigger a sell event. In our magnet simulation, which simulates 2D interactions between a "fixed magnet" and a magnet that is allowed

to move, the input parameters are the initial polar coordinate (r, θ) of the movable magnet relative to the fixed magnet, the starting rotational angle of the movable magnet's dipole axis (β), and the initial momentum for each of these three dimensions. In this context, one example of an event is the free magnet bouncing off the fixed magnet in an inelastic collision. Each simulation will have a set of timestamped events. In the magnet simulation, small variations in the input parameters will produce relatively small variations in the first few events of the simulation. These variations are heteroskedastic: over increased amounts of time, the small variations in input parameters can cause the simulation to devolve into chaos or to enter a stable cycle of events.

Event families may be produced when small variations in inputs yield small variations in outputs. An event family is an event that is produced in multiple simulations. An event family death occurs when a threshold is crossed, causing a previously existent event to stop appearing in later simulations. An event family birth is the opposite - a previously unseen event begins appearing in later simulations. Bifurcations occur when an event family splits into two separate event families. A merger is the inverse, where two event families merge into one.

In this work, we explore high-dimensional event data across simulations. In particular, we seek to create a visualization system where a user could explore how changes to input parameters cause event family births, deaths, bifurcations, and merges. Using our previous example, suppose a military analyst is considering how a series of missile strikes will affect key assets. Pursuant to this, they run simulations of missiles launched toward various cities and explore which simulations resulted in the missiles being detected by friendly assets. We cast the problem of exploring this data in terms of visualizing the simulation runs in series and discovering problem simulations where threats were not detected. A second example relates to financial markets: while stock market trading algorithms may use machine learning or some other technique to determine how much weight is given to different events in determining whether to buy or sell, it may be difficult for the analysts to explain why the algorithm chooses to buy or sell at a certain time. By tweaking the input weights slightly for each simulation of the algorithm, the analyst can gain a clearer understanding of which thresholds and events are most impactful on the output set.

Using the taxonomy given in [12], our contribution is two-fold. The first contribution is data transformational, using z-order linearization to provide a 1-dimensional ordering of simulations. The second contribution, given an ordering of simulations, is a visual mapping of events that allows the user to interactively discover topological features of the events across simulations. After discussing related work (Sec. 2), we describe our linearization approach (Sec. 3.1) followed by our visualization technique with examples (Sec. 3.2). We conclude with implications of our work and future directions (Sec. 4).

2 Related work

Simulation data is typically analyzed in terms of scalar functions in the spatiotemporal domain and other temporally-driven domains across a single simulation. Dimension reduction techniques (e.g. PCA [11]) reduce the number of attributes to a quantity that is easier to visualize. Clustering (e.g. k-means clustering [1]) is an unsupervised learning technique that partitions samples into meaningful groups. Subspace clustering (e.g. [3]) clusters dimensions, often along with clustering of samples. Topological data analysis (TDA) [5, 12, 13, 16] allows users to analyze data using topological constructions that both provide compact descriptions of the data and reveal features. There is a body of work that compares features across simulations, such as comparing clusterings [9] and topological merge trees [2], but none to our knowledge that directly compares discrete events.

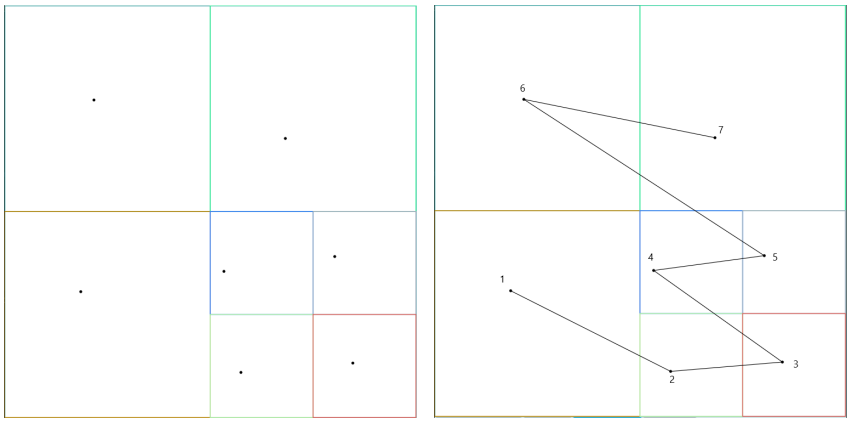
Our goal is a level of abstraction away from typical methods. Rather than analyzing scalar or multi-dimensional, quantitative output functions, we are interested in families of discrete, temporal events across simulations. To our knowledge, no visualization technique of this type of data has been published. Approaches have been suggested for exploring and predicting single events such as extinction and reignition in a combustion engine [10] and successful binding of an antibody to a noxious molecule [14], but these approaches do not analyze the topological structure of multiple events across time across simulations.

3 Description of the visualization

We describe our visualization approach to allow users to explore event families in high-dimensional simulations. We demonstrate the techniques using the *MagPhyx* simulation software [8]. *MagPhyx* is a software package that simulates spatial interactions between pairs of spherical dipole magnets. In our examples here, we vary combinations of three input parameters, β, θ, and r [8].

3.1 Linearization

One of the key challenges in exploring event families across simulations is placing simulations in an order that allows the user to find patterns, points of interest, or divergences. In order to visualize event families in a meaningful way, we need a method of linearizing (i.e. reducing to a single-dimensional ordering) the simulations. For input sets following a grid pattern, the ubiquitous and intuitive row- and column-major orderings are simple but have poor spatial locality [4]. That is, the average distance in parameter space between two consecutive simulations is large compared to distances between consecutive points in other, better orderings. In order to traverse the data so as to minimize large jumps between simulations that are displayed as consecutive, we use z-ordering as constructed using a quadtree [15] . While this approach occasionally results in a large jump, it usually will place simulations that are close together in input space next to



(a) (b)

Figure 1: (a) A quadtree subdivides until there are one or zero points in each cell. (b) Z-Order Traversal. The algorithm starts at the bottom-left leaf of every node and continues to the bottom-right, top-left, and top-right leaves until every point is visited.

each other in the simulation display. After recognizing some of the spatial locality problems with z-ordering, we began using Hilbert curves to reorder the data. Hilbert curves did improve the spatial locality of the simulations.

Quadtrees (Fig. 1a) are data structures commonly used in collision detection and shape modeling. The structure is recursive, with each non-leaf cell containing exactly four children. Leaf nodes contain exactly one or zero points. In our implementation, a cell will subdivide if it already contains a point and a new point is inserted into it.

Z-ordering is a graph traversal technique used to order data in arbitrary dimensions into a one-dimensional ordering while preserving spatial locality. While a z-order traversal can be calculated by interleaving binary coordinates of the data, a depth-first traversal of a quadtree will produce the same output. In our implementation, the traversal (Fig. 1b) starts at the bottom-left cell that contains four sub-children and visits the bottom-left, bottom-right, top-left, and top-right sub-children. It then moves up a level recursively and repeats itself until every cell in the graph is visited.

This technique will produce a one-dimensional representation of all the points on the graph. The benefit of linearized simulations is that we can see the effects of many small changes in input parameters. This allows identification of event births, deaths, merges, and bifurcations in multidimensional simulations.

3.2 Visualization

Our visualization includes three charts: a scatter plot of input configurations (Parameter Space chart), a scatter plot of all events, and timeline chart of events. We describe each chart in detail.

1. Parameter Space chart (Fig. 2) - This first visualization accomplishes several purposes. First, it allows the user to see the ordering of simulations used in the Event Timeline chart. Second, in our *MagPhyx* setting, clicking any point in the Parameter Space chart will replay that simulation, allowing the user to see detail on the simulation behavior. Finally, hovering over a point will highlight all events in the other two charts that were produced by that particular set of parameters, linking the charts together. Note that this visualization only shows the variation in two dimensions. This is not a problem if only two input dimensions are varied or if two input dimensions are the primary focus; however, most applications will vary many input dimensions, exceeding the capacity of this chart.

2. All Events chart (Fig. 3) - This visualization displays all output events from all simulations, charted on a scatter plot of theta and phi. This allows the user to see clusters of events or correlations in the data. Hovering over any of these points will highlight the point in the Parameter Space chart. Similar to the Parameter Space chart, the All Events Chart is limited to displaying two dimensions.

3. Event Timeline chart (Figs. 4 - 6a) - This visualization allows the user to explore patterns in the simulation data caused by variations in input parameters. Each row of the chart corresponds to a single simulation. The rows are ordered according to the order displayed in the parameter space component.

Fig. 4 shows an example of the Event Timeline chart. This particular dataset has some interesting periodicity in its output, as indicated by the arrows at X1, X2, and X3. The periodicity is not an artifact of linearization in this case, but rather, is inherent in the physical phenomenon [6, 7].

Fig. 5 displays several interesting phenomena. At (a), we can see an event family death. At that point, a threshold of some point had been reached, causing that event to stop occurring. At (b), we see an event family birth (in this case, a re-birth). This is essentially the inverse of a death. At (c), we see examples of bifurcations, where an event family divides into two distinct event families.

With this visualization, we can see the necessity of good linearization among simulations. Consider Fig. 6a, which used column-major order to linearize its simulations. Every 10 simulations, a shift occurs. These shifts draw the attention of the user but don't correspond to any phenomena in the data. Instead, these shifts are caused by large jumps in the parameter space. This is not ideal, as the shifts distract from meaningful patterns in the data.

Contrast Fig. 6a with Fig. 6b. This figure uses the same data as Fig. 6a, but uses z-ordering instead of column-major ordering. This change makes the

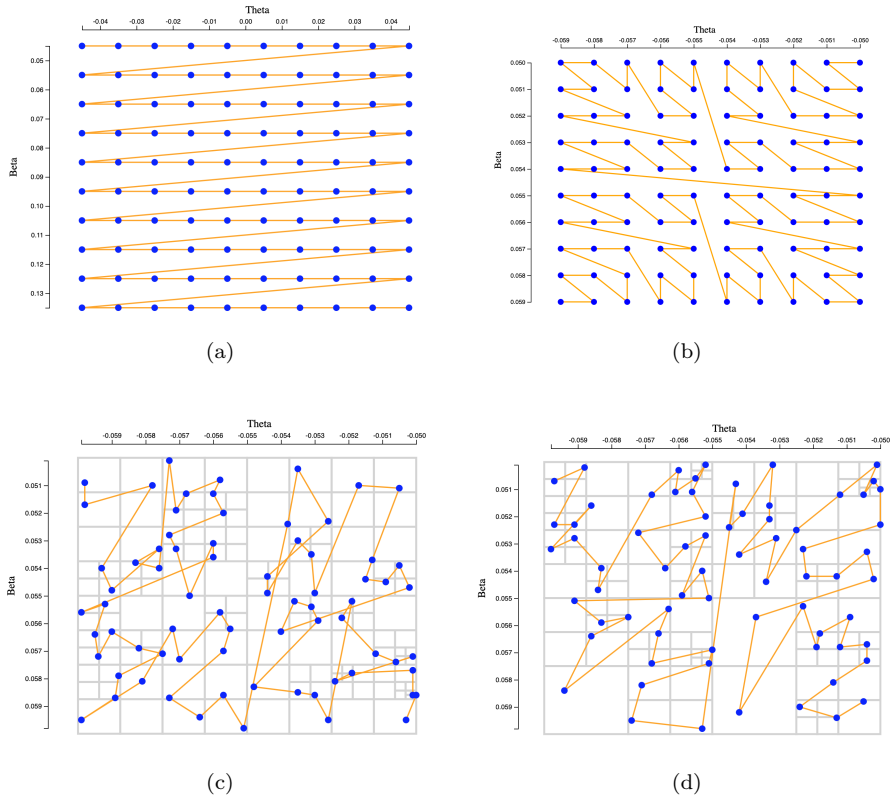


Figure 2: These examples of a Parameter Space Chart show how the linearization of simulations is displayed to the user. (a) Row Major Order. (b) Z-Ordering. Note the decrease in large jumps from row-major order. (c) and (d) Z-Ordering on randomly sampled parameter space. In both cases, the data were produced by randomly varying θ and β from -0.05 through -0.06 and 0.05 through 0.06, respectively. (c) shows a potential limitation of z-ordering, as several large jumps between consecutive simulations can be seen. (d) was created through the same process, but has fewer large jumps between consecutive simulations.

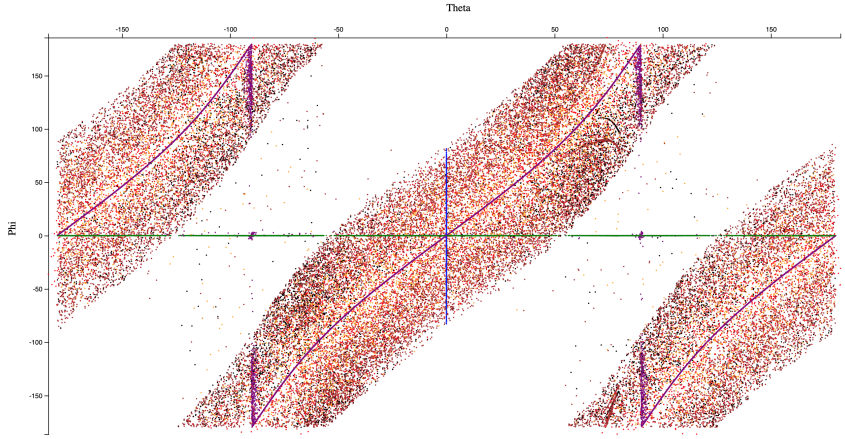


Figure 3: All Events Chart

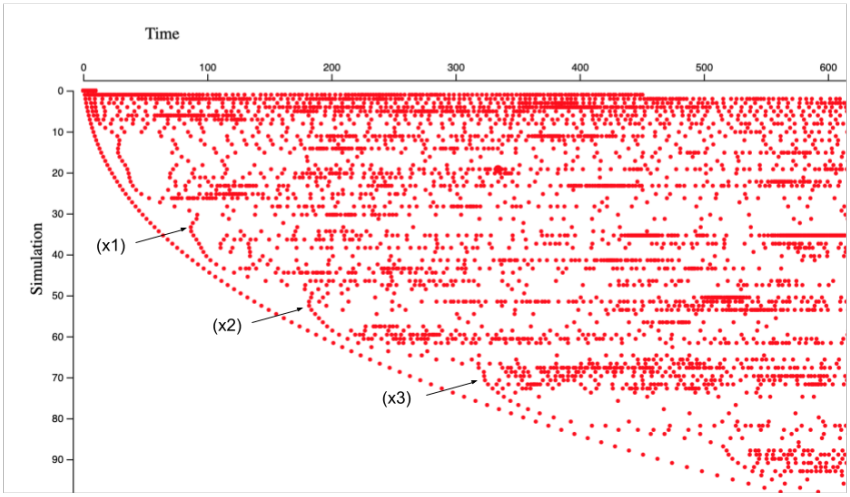


Figure 4: This Event Timeline shows periodicity in event families. The data were produced by linearly varying the starting radius from 1 to 10 and keeping beta and theta at 0. Each simulation is given a starting momentum angle of 80 degrees and a starting location, relative to the origin, of 80 degrees. The simulations are ordered based on the starting radius. Collisions are shown.

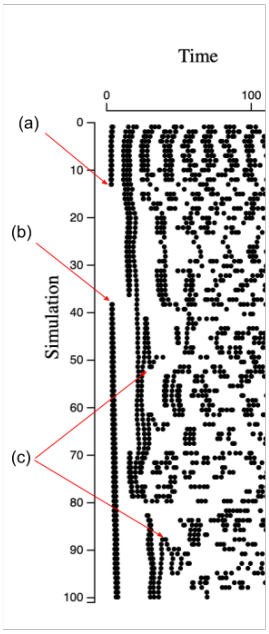
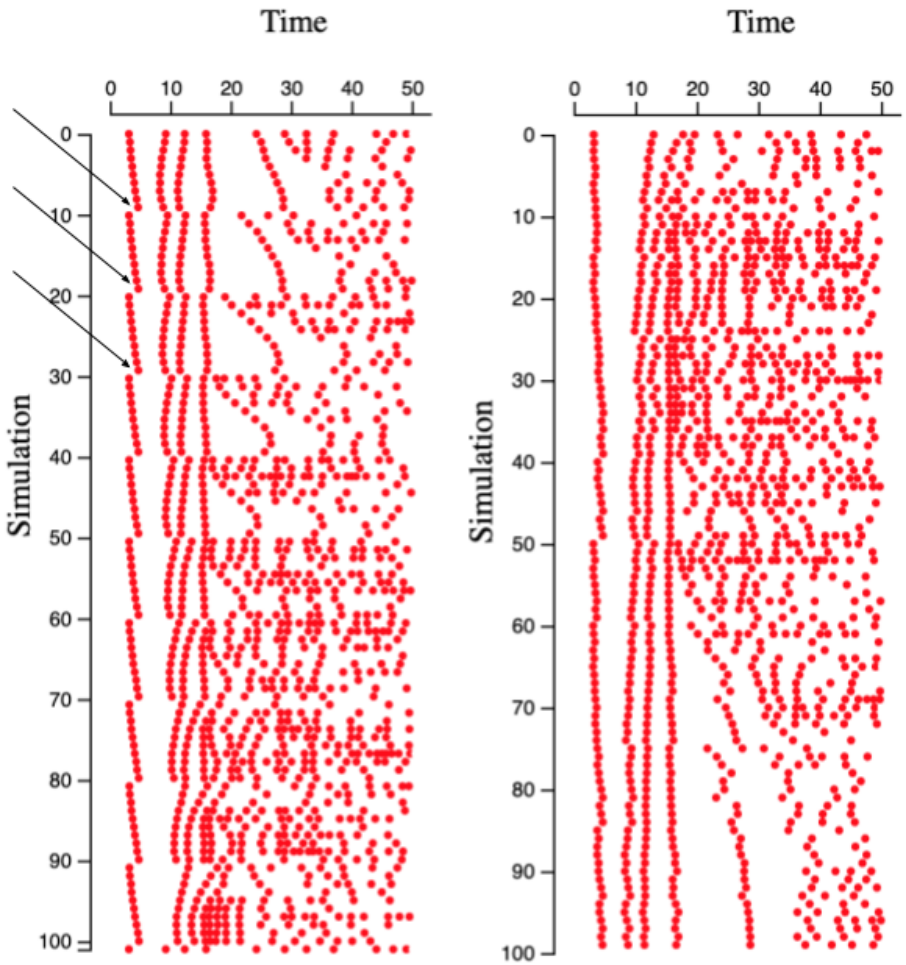


Figure 5: This Event Timeline displays several interesting phenomena, including an event family death (a), an event family birth (b), and several bifurcations (c). These simulations were produced by varying the starting radius from 1 to 2, varying theta from 0.045 to 0.055, and varying beta from -0.045 to -0.055. As the simulations were produced in a linear manner, no linearization was necessary. Events are defined as zero crossings of the magnet's angular moment.

output far more continuous, avoiding most of the unwanted shifts seen in Fig. 6a.

The structure of events as shown in the event timeline chart is highly dependent on the presentation order of the simulations. As can be seen in Fig. 7a, the first three events are consistent across all simulations and the fourth event appears in many simulations, but once event times start to diverge and event families are born or die based on their input parameters, the ordering becomes vital to clustering like-simulations together in terms of their event similarities. In Fig. 7b we see that using an ordering (z-ordering) with better spatial locality results in events families that are easily parsed visually. We also see event families much later in Fig. 7b than was possible to distinguish in Fig. 7a.

Because of the ease of visual parsing of event families in the timeline chart, the analyst can identify thresholds causing births and deaths of events of interest and therefore isolate dimensions (input attributes) causing the topological changes of event families.



(a) (b)

Figure 6: (a) This Event Timeline displays a problem arising from poor linearization. The shifts occurring every 10 simulations are from the use of row-major order, not from any phenomena in the data. The large jumps occur between the simulation at the end of one row and the start of the next row. (b) This Event Timeline uses the same data as is used in 6a, but uses z-ordering to improve the linearization. This change to linearization removes the unwanted shifts. These data were produced by varying theta from 0.045 to 0.055 and beta from -0.045 to -0.055, producing a grid pattern. Radius was set to 1.5, starting momentum angle set to 80 degrees, and angle from the origin set to 80 degrees. The output events are collisions between the magnets.

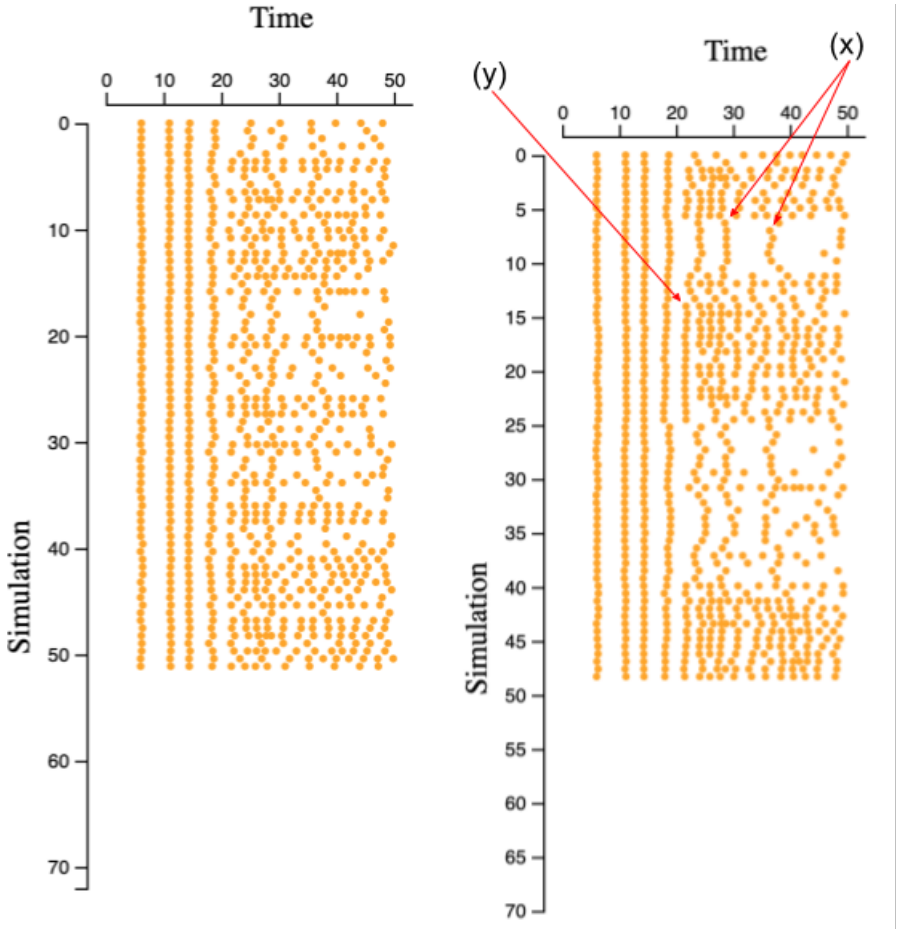


Figure 7: These two examples of the Event Timeline Chart show how z-ordering can improve pattern recognition with randomly produced simulations. Both show the same data, which are the same data seen in 2d, and show events where the magnet's momentum was zero. Fig. (a) shows the visualization with z-ordering turned off. Fig. (b) shows the same data in z-order. Several event families that were not clear in (a) can be seen in Fig. (b), including two at (x). While both Event Timelines show the same first four event families, a fifth event family emerges in Fig. (b) at (y). Thus, we see it is much easier to find event families and patterns of events in Fig. (b). Note that the same set of simulations was removed from both of these charts because the simulations' input parameters were extremely close together.

4 Conclusions

In this project, we created a novel visualization approach to discovering event patterns across simulations and applied this visualization technique to data from a magnet dynamic simulation. We found that, while z-ordering did provide better linearization than row major order or unordered simulations, z-ordering created large jumps that are problematic for the visualization. Hilbert curves offer a potential improvement in linearization. This work serves as a starting point for future efforts – we plan to explore additional methods of linearizing data, use the visualization for additional data sets including stock market data, and look for methods of evaluating the effectiveness of the visualization. Ideally, we would compare a human-annotated gold standard dataset against future empirical results. This will only be possible as we continue to refine the definition of the problem. Needs in this area also relate to the visualization technique itself, including providing user awareness of the distance between simulations in the event timeline chart.

5 Reflections

5.1 Capstone Experience:

This project was the capstone of my undergraduate education because it combined concepts from my majors in Computer Science and Economics and my minors in Mathematics and Anticipatory Intelligence. The project mainly focused on computer science, as did my class load and work experience during my undergraduate degree.

In particular, this project drew heavily from CS 5890 Data Science and Data Visualization. We used the visualization library d3, as taught in CS 5890, to display our visualization. Since Dr. Edwards, Jaxon, and I all worked on the code, we used practices learned in CS 3450 Software Engineering - particularly the use of the version control system Git. Here are several other classes and the skills that were applied to this project:

- HTML, CSS, and JavaScript - Intro to Web Development (CS 2610)
- Advanced JavaScript techniques and debugging - Game Development (CS 5410)
- Event handling and User Interface design - Intro to Event-Driven Programming and GUIs (CS 2410)
- End-user considerations and project importance - The Art and Science of Anticipation (CAI 5300)
- Statistical methods - Intro to Probability (Math 5710), Intro to Econometrics (ECN 4330)

5.2 Contribution to Overall Education and Future Goals:

The project provided me with an excellent opportunity to prepare for graduate school, one of my future goals. The skills needed to prepare an academic paper, submit for peer review, and present research are foundational for success in graduate school.

5.3 Relationship with a Mentor:

Dr. Edwards has been a fantastic mentor for this project. He has done a great job of helping Jaxon and I explore intriguing concepts and find ways to quickly implement those ideas into the visualization. When we would get stuck on a task, he would help us resolve it; otherwise, he would focus his attention on the big-picture of the project. Additionally, Dr. Edwards has encouraged me to continue my education by attending graduate school and offered advice on how to navigate that process.

5.4 Research Experience Within Computer Science:

As a sophomore, I worked as a research assistant for Dr. Chad Albretch. Our research primarily focused on business strategy and corporate fraud. We published an article in the Journal of Financial Crime and I helped review several academic papers. While this work was deeply interesting, it was not related to computer science or closely related to economics. This capstone project helped take some of the lessons learned from that work and apply them more directly to my computer science major.

5.5 Critical Thinking About Topics in Computer Science:

As this project continued, refactoring and refinement became increasingly important for adapting the code to our new ideas and new features. This aspect of the project was more similar to my experience working at Space Dynamics Lab than my experience completing projects for classes. Projects for computer science classes tend to be relatively short, lasting a couple of weeks at most. Having a much longer-term project highlighted the importance of making code reusable and maintainable. For most commercial software, reusability and maintainability are absolutely crucial.

5.6 Experience Across Disciplines:

While this capstone project was primarily related to computer science, the applied nature of the visualization lent itself to ideas from other disciplines. Thanks to my second major in Economics and my minors in Mathematics and Anticipatory Intelligence, I was able to utilize my strengths in other disciplines to improve this project.

Economics and Mathematics helped both develop the visualization and find applications for the technique. We standardized the parameter data to make it comparable across dimensions - a common practice in statistics and econometrics. We considered several different distance algorithms from mathematics to judge how similar simulations are, including Euclidian distance, Manhattan distance, and Hausdorff distance. Near the end of the project, we began working on expanding to data sets beyond our magnet simulation data. Using my background in Economics, I reached out to a master's student in the Finance department with an idea about creating simulations for our visualization based on stock market trading algorithms.

Anticipatory intelligence mainly highlighted the need for good visualization techniques. In CAI 5300, the Art and Science of Anticipating the Future, we frequently talked about how complex problems are difficult to reason about, in part because they are difficult to visualize. This provided an extra motivation to work on the project - figuring out ways to visualize high-dimensional data could provide extra insight into complex problem sets.

5.7 Impact of Work:

Dr. Edwards has talked several times about the potential for this project, after some more work, to make it into the VIS conference - the premiere conference in data visualization. This is an exciting prospect, both because of the chance to show our work more broadly and the prestige of presenting at the VIS conference.

5.8 Key Transition Points During the Project:

When we began this project, we were focused on applying for an STR grant from the Missile Defense Agency (MDA). Due to insufficient communication with our partner company, we had to scramble to be ready to present for the grant consideration. After the presentation, it took us a few weeks to figure out how to proceed while we waited to hear back. During this period, the project began to evolve significantly. We began generating data from MagPhyx magnet simulation and Jaxon Willard joined the project.

Once we implemented a few initial ideas using the MagPhyx data, the project became more general and more interesting. Instead of focusing on how the sample data from the MDA could be visualized, we began trying to visualize any type of high-dimensional data from contrasting simulations. Furthermore, we recognized linearization as a key challenge in this capstone project. Depending on how well the simulations are put in order, patterns either appear or disappear.

During this phase, the project became more intriguing and meaningful for me. Instead of just applying past ideas to the new problem or speculating on theoretical solutions, we were exploring new ideas and successfully implementing those ideas. We frequently ran into truly hard problems and sometimes found a way around them.

We did not receive the grant from the MDA; however, since the project was going well, we decided to continue work on the project.

5.9 Lesson on Planning:

Throughout this project, we made many plans that did not end up working out. The grant did not work out, various timelines had to be changed, and the scope and comprisal of the project changed over time. By the end of the project, the continual adaptation became the normal mode of research. I enjoyed the flexibility, although it was sometimes difficult to communicate the overall plan for the project.

6 Author Bio

Steven Deron Scott is an undergraduate student at Utah State University. He is a dual major in Computer Science and Economics, with minors in Mathematics and Anticipatory Intelligence. During his time at USU, he participated in the Huntsman Scholar Program, Go Global Program, Society for the Advancement of Ethical Leadership, and Anticipatory Intelligence Program. With Huntsman Scholars, Steven traveled throughout Western Europe studying international business, ethics, and the politics of the European Union. During the Go Global Program, Steven studied international business and entrepreneurship in Chile and Peru. In the Anticipatory Intelligence Program, Steven worked with students from various majors across campus to research local, national, and global security concerns. As a member of the leadership council for the Society for the Advancement of Ethical Leadership, Steven selected ethics-themed books for club members to read and discuss. In 2019, Steven received the USU Outstanding Junior in Computer Science award. After graduation, Steven plans to work in software engineering and eventually attend graduate school.

References

[1] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. An efficient k-means clustering algorithm. 1997.

[2] Kenes Beketayev, Damir Yeliussizov, Dmitriy Morozov, Gunther H Weber, and Bernd Hamann. Measuring the distance between merge trees. In *Topological Methods in Data Analysis and Visualization III*, pages 151–165. Springer, 2014.

[3] Chun-Hung Cheng, Ada Waichee Fu, and Yi Zhang. *Entropy-based subspace clustering for mining numerical data*. PhD thesis, Citeseer, 1999.

[4] Peter J. Denning. The locality principle. *Communications of the ACM*, 2005.

[5] Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci. Morse-smale complexes for piecewise linear 3-manifolds . 2003.

[6] B. F. Edwards and J. M. Edwards. Periodic nonlinear sliding modes for two uniformly magnetized spheres. *Chaos*, 27(5), 2017.

[7] B. F. Edwards, B. A. Johnson, and J. M. Edwards. Periodic bouncing modes for two uniformly magnetized spheres I: Trajectories. *Chaos*, in press.

[8] Boyd F Edwards and John M Edwards. Dynamical interactions between two uniformly magnetized spheres. *European Journal of Physics*, 38(1):015205, 2016.

[9] Chris Fraley and Adrian E Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal*, 41(8):578–588, 1998.

[10] Samuel Gerber, Peer-Timo Bremer, Valerio Pascucci, and Ross Whitaker. Visual exploration of high dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1271–1280, 2010.

[11] Ian Jolliffe. *Principal component analysis*. Springer, 2011.

[12] Shusen Liu, Dan Maljovec, Bei Wang, Peer-Timo Bremer, and Valerio Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE transactions on visualization and computer graphics*, 23(3):1249–1268, 2016.

[13] Monica Nicolau, Arnold J Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011.

[14] Richard A Norman, Francesco Ambrosetti, Alexandre MJJ Bonvin, Lucy J Colwell, Sebastian Kelm, Sandeep Kumar, and Konrad Krawczyk. Computational approaches to therapeutic antibody design: established methods and emerging trends. *Briefings in Bioinformatics*, 2019.

[15] Hanan Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260, 1984.

[16] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *SPBG*, pages 91–100, 2007.